第 7 卷第 2 期
1996 年 2 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 2
Feb. 1996

电子 CAD 框架系统的长事务处理和网络版本管理*

潘雪增　寿宇澄　赵余平　平玲娣

(浙江大学计算机系　杭州　310027)

摘要　用面向对象方法管理框架系统中的工程和设计数据已成为大多数人的共识. 本文对该系统中的 Server/Client 结构和并发控制、长事务管理、网络上的对象版本管理等进行了论述,提出了想法和实现. 该系统已应用到电子 CAD 框架系统中.

关键词　计算机辅助电子设计,面向对象,框架系统,长事务处理,网络版本管理.

工程设计数据管理系统(EDDMS)是电子 CAD 框架系统的基础. 根据工程实际需求,EDDMS 提供的主要功能有:

(1) 设计对象的版本管理,记录设计数据演变的历史. 同一对象的多个版本自动地保存,而且及时地记录版本的导出关系.

(2) 数据库分类管理,把数据库分为 3 类:标准库、设计库和工作库. 标准库是公用库,存放着经过验证的单元对象. 设计库存放设计对象. 工作库存放事务处理过程中的设计对象.

(3) 配置管理,提供配置定义和访问路径. 用户能方便地指定某一设计的构成成分为一个配置. 随着设计的进展,该配置的版本信息都被记录.

(4) 存取权限控制,保证设计对象的安全访问. 未经授权的用户禁止访问,检索或修改特定的设计对象.

(5) 并发控制,提供平行访问功能可以获得最大的数据共享. 但是为保持数据的一致性和完整性,必须加以严格控制.

(6) 设计数据的追踪管理,记录全部操作环境信息. 这样,便于对存储的信息追踪检查,对后来的查询与分析十分有用.

国内外近年来在这一领域研究相当活跃. 文献[1]提出版本保存模型,文献[2,3]提出配置概念,文献[4]中介绍了事务处理方法,文献[5,6]详细介绍 OODB 的原理及在各有关应

---

* 本研究受国家 863 高技术的资助.作者潘雪增,1944 年生,副教授,主要研究领域为 ECAD,工程数据库,电子 CIMS、电子 CAD 框架. 寿宇澄,1968 年生,讲师,主要研究领域为工程数据库,人工智能,智能 CAD 等. 赵余平,1967 年生,1995 年硕士毕业于浙江大学,主要研究领域为 CAD/CAM 技术,工程数据库,框架技术. 平玲娣,女,1946 年生,副教授,主要研究领域为 CAD/CAM 集成技术,设计自动化(EDA),CIMS.

本文通讯联系人:潘雪增,杭州 310027,浙江大学计算机系

本文 1994－12－30 收到修改稿

用领域的实施实例.本文的目的在于解决EDDMS中的几个关键问题:长事务管理、Server/Client结构以及网络上的版本管理.

1 长事务管理

1.1 系统的Server/Client体系

本系统在网络上实现,采用的是Server/Client模型,以TCP/IP为通信协议.本系统的Server/Client上的空间和进程管理有自己的考虑.

数据空间并不是全都由Server进行统一管理的,而是分为Public DB和Private DB两部分.由Server负责管理Public DB,存放设计成熟信息,由Client自行管理Private DB,存放设计过程中的信息.Public DB和Private DB间用Check_in和Check_out操作相联系.

采用这种空间管理形式的主要目的是为了适应工程长事务的需要.在工程应用中,如何减轻网络负担是一个十分重要的问题.工程中的长事务可能历时很长,如果经常频繁地从Public DB中获取数据,那么网络上的负担就太重了,也可能引起系统事实上的无法运行.Private DB存放当前Client所需的信息,而不是将所有空间都统一由Server管理,只需在事务开始/结束时用Check_out/Check_in操作将所需信息从Public DB取出/放入,在设计过程中只要进行少量的信息交换,从而减少网络负担.现在的硬件环境也完全能满足这种要求.

采用这种方式的另一个好处是简化事务处理中的并发机制,因为不必在Client端考虑并发问题.由于是在TCP/IP协议下进行通信,因此需要制定自己的传输格式.在Check_out时,首先检查出模式的合法性,包括用户的权限和锁冲突的检查,然后将申请输出的信息打成包,包的格式是事先定义好的,通过网络将信息从Server端传入Client端,在Client端进行解包,放入Private DB.各种Client端的信息操作均在Private DB上进行.当一个设计过程完成时,通过Check_in操作将信息从Client端传回Server端.在Check_in操作中,首先进行合法性检查,然后进行信息过滤,对那些在本次事务中进行过修改的信息打包,通过网络传到Server端,解包,存入Public DB,并做必要的信息通报.由于面向对象数据独立性好,控制简单,因而可以保证在网络上传递的信息绝大部分为工程必需信息,冗余信息也较少,使网络上的负担降到最低限度.

1.2 长事务处理

事务的基本含义是:在事务开始和结束时,保证数据库的一致性和完整性.事务管理的两个主要功能是:并发和恢复.

工程事务过程具有以下3个特点:

(1) 工程设计过程是长时间的活动,一个工程设计事务经常要持续几小时,几天,乃至更长,而传统商用事务处理只持续几秒钟.

(2) 工程设计过程是群体协作的活动,一个工程设计项目往往分成若干子项目,而每个子项目又可以分成若干项目组……,子项目、项目组成员之间需要交换信息.

(3) 工程设计过程是试探性的活动,设计人员往往不是沿着一条线性的路径进行设计,而是有许多的反复试验,以便寻找最佳设计.

为支持这些特点,必须摒弃商用事务处理中简单的等待和放弃.既不能要求用户对合理

申请进行长时间的等待,也不能简单地对失败的操作进行放弃,而是应建立一套新的锁机制和事务恢复机制以支持工程事务的长期性和试探性特点.

本系统从传统 DBMS 的事务管理继承了它的日志功能和 recovery 机制. 为了适应工程设计的需求,又加入了 savepoint 概念和 rollback 机制,这是对传统 DBMS 的事务管理概念的扩充. 正是在此基础上构成了具有自己特征的事务处理模型. 事务处理模型如图 1 所示. 由于每个对象都拥有其自身的版本进化信息,所以在 savepoint 中要复制的信息只是各对象的当前版本号.

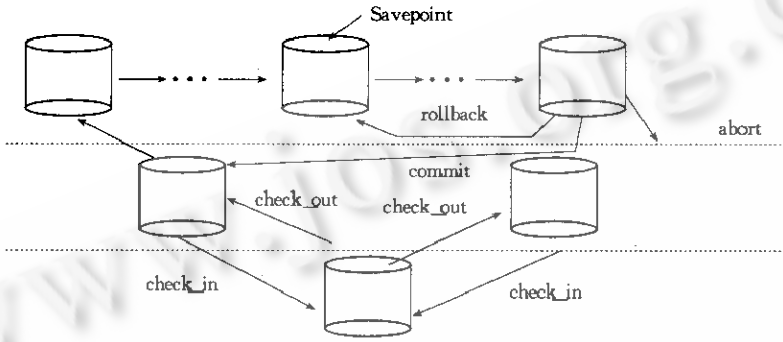

图1 事务处理模型

另一个重要的机制是锁机制. 这是合法性检查和冲突解决的基础.

长事务所持有的锁应是持久锁,在实现上应随时在外存保存最新锁状态.

本系统以对象为信息独立单位,上锁的范围以对象版本为最小单位. 不过对于复合对象,其上锁范围还包括其下属的对象,即该对象的闭包.

基本锁模式有两种:Read 和 Update.

锁的操作有 5 种:对对象上锁、检查是否可对对象上锁、释放一持久锁、检查某一事务是否对某一对象上了锁、将某一事务的所有锁释放.

对于网上用户来说,一个很关键的问题是冲突的处理. 一般的锁冲突有读－写冲突和写－写冲突两种. 对于写－写冲突,由于在工程设计中很少会出现同一数据允许不同用户同时设计修改的情况,因此判定后申请者申请失败是合理的. 而对于读－写冲突,则允许用户继续工作,但当事务提交时,应向读事务通报最新修改. 由于事务提交后信息将放入 Public DB,因此读事务中的内容更新是较易实现的. 只须根据通报信息将最新版本的内容取到 Private DB 并调整当前版本号即可.

1.3　长事务的恢复机制

如前所述,事务处理可以分为二部分,一部分位于 Server 中,另一部分在 Client 端. Server 部分主要处理主数据库的并发访问及锁机制,事务识别,数据 check_in/check_out 控制等.

Client 端的事务管理模块主要向用户提供记日志和 recovery 功能. 日志(log)管理为对象的每次改变记日志,recovery 支持 UNDO 和 REDO 操作,以实现系统崩溃时的恢复工作. 此外,为了适应对工程设计过程中长事务的特点和探索性的工程设计,又增加了 savepoint,rollback 和 pause,resume 功能.

Client 的一个长事务由多个子事务构成. 有关子事务的操作如下:

(1) transaction_begin:开始一个子事务
(2) transaction_abort:放弃一个子事务
(3) transaction_commit:提交一个子事务
(4) savepoint:建立一个保存点
(5) rollback:回退到一个保存点
(6) pause/resume:事务暂停与重启
(7) update,create,delete:基本数据库操作
(8) undo/redo:消除一个操作/重做一个操作
(9) recovery:数据库的恢复

其中的 recovery 用于系统的恢复;而 pause/resume 是为了提供更良好的用户环境,允许用户暂时停下手头的工作,休息后再重新 resume,恢复原先的状态.这对于长事务来说也是必须的.undo/redo 操作的定义是为了支持 rollback 和 recovery,这二个操作对应用程序是"透明的".利用以上定义的操作,就可以构成一个完整的事务处理过程模型.

整个事务处理过程可以用一棵树来表示,树的根是事务的开始,每次 savepoint 操作对应于树上的一个结点.rollback 可以使用户在各个结点之间自由转换,但当前的结点只有一个,代表了目前所处的状态.通过这种树状态事务,系统可以支持工程设计中的探索性特点.

传统的事务处理模块主要提供 2 项功能:一项是恢复,另一项是并发控制.由于在 client 端,数据是由用户独占的,因此不用考虑并发机制,这大大简化了事务处理的实现.但是 server 中的事务管理模块管理的是公用的数据库,这部分是必须要考虑并发和冲突的.

在事务处理过程中,每一个操作都遵循"先写日志"协议,即在对数据库进行操作前,应先将所作的修改记入日志中,以备将来恢复和 rollback 时使用这些信息.这样,一旦系统崩溃,事务管理模块可以根据记在日志文件中的信息,恢复到崩溃前的状态.而 rollback 也可以利用 log 日志文件记入的信息,使用户在不同的结点之间进行状态转换.

因此,日志文件是 Client 端事务处理的核心,所有的操作都是围绕日志文件进行的.

在使用过程中,对事务试探性的支持集中表现在 savepoint(事务状态结点)的使用上.

在实际的工程设计过程中,设计可能一种方案一次成功,也可能通过多种方案权衡和取舍,最后选出比较合适的一种方案.这种设计过程决定了工程设计过程不是线性过程,而是一个试探、摸索的过程.虽然最终往往只有一个结果,但在没有最后选择之前,任何一种可能的信息都应该完整地保存,以便于选择修改.

对于单个对象(包括复合对象)来说,可以由版本管理来进行信息保存,而对于整个数据库,就要用事务处理来记录设计过程.

该系统是支持 CAD/CAM 等工程设计的,因此也必须与实际的工程设计过程相适应.反映在事务处理上,就产生了探索性的事务处理过程.

于是,在事务处理过程上,提出 savepoint 和 rollback 概念,以支持探索性事务处理过程.savepoint 事务状态结点是事务处理过程中一个相对稳定的状态,这种状态下,数据库内,日志文件内部数据与日志文件之间的数据是完整的,并且是一致的,它对应于工程设计过程中相对完整的一个设计方案,如一种可能的布局方案.由于它具有一定的稳定性和完整性,可以由 rollback(滚回)操作在不同的结点之间进行变换.因此,它是 rollback 和recovery

操作的目标和基础,并且 rollback 只能滚回至结点处,而不能是事务处理过程中的其他状态,这是它相对于 rollback 和 recovery 操作的原子性. 由于 savepoint 是事务处理过程中的一个相对稳定、完整的状态,可以使用户在这样一些的状态之间进行 rollback 类型的状态转换,转换之后的状态就如同用户直接进行到此状态,而没有作过其他的操作一样. 事务处理就是以这种方式支持工程设计所要求的探索性特点. 值得强调的是,在 savepoint 中保存的是对象的版本信息,而不是复制对象本身.

一个结点可以有父结点和子结点. 它的父结点表示它的前导状态;它的子结点则是继承了它的状态,继续进行了一系列的事务操作之后设定的状态结点. 因此,一个结点只可能有一个确定的父结点,而它可以有很多个子结点,各个子结点都继承了它的状态,但在此基础上,各自又进行了不同的操作.

整个系统的状态结点构成了一棵树.

事务的 rollback 和 recovery 操作是数据库事务处理部分的关键之一. recovery 操作是传统数据库系统所必须具备的功能. OODB 作为 OOP 与 DB 技术的结合,也必须具有这种功能. rollback(滚回)则是在传统数据库处理方面的扩展. 因为传统的数据库是一个线性的过程,因此它的滚回也是一个线性的过程. 本系统的事务处理过程由于支持工程设计的要求,因而具有非线性的特点,整个事务处理过程是用一棵树来表示的,并增加了状态结点的概念. 事务的状态能在各结点之间转换. rollback 操作可以提供在任意两个已存在结点之间的转换. 转换后系统的状态,就如同从根开始直接做到目标结点的状态. 又因为 rollback 以 savepoint 结点为最小操作单位,而不是事务处理定义的操作,因此,在 savepoint 之后和 rollback 之前的操作将被丢弃不能恢复. 由于事务状态的这种树形结构突破了以往的线性结构,带来了支持长事务,探索性设计等方面的优点.

1.4　长事务的其它机制

长事务的其它机制包括:开始和结束、暂停和重启,以及事务放弃. 其中暂停和重启是长事务所特有的机制. 由于长事务可能出现一次事务需要很长的时间完成,所以有必要提供 pause/resume 机制,以便于设计人员休息和重新设计思考. 其实现机制与 savepoint 和 rollback 是类似的.

2　网络版本管理

由于采用 Server/Client 体系结构,因此除了普通的版本管理功能外,还特别要考虑网络上的版本功能.

Server 上保持着具有全局性的公用数据库,存放成熟的信息,可被 Client 上的私有数据库共享,而 Client 上的数据库存放的是一个用户在一次长事务过程中的信息,在一次长事务开始时,用户通过 Check_out 操作从 Server 上的公用数据库中提取感兴趣的,也是允许授权操作的数据子集,传送到 Client 上;这次长事务完成时,他要将这次事务中修改的信息经审批后提交给 Server,版本管理器的任务是在获得授权的前提下,对指定的对象版本树的信息进行打包和解包,同时必须保证信息的完整性和一致性.

在进行 Check_out 操作时,系统允许用户用以下两种方式从公用数据库中提取数据.

(1) 指定一个对象,即提供这个对象的标识,这种提取方式将该对象的所有版本及其所

有成员对象的全部内容传送到用户所在的 Client 上;

(2) 指定一个对象的一个版本,这种方式只是将指定对象版本的内容以及其构型中包括的成员对象的相应版本的内容传送到 Client.

为了实现在网络上的传送,必须将上述内容打成一个二进制数据包提交给网络传输模块. 第一种 Check_out 的方式,事实上传送了指定对象的一份拷贝,它的打包单元是一个对象的全部版本内容,其成员对象的内容是另外分别打包的,再由事务管理和网络传输模块来实现信息传送的完整性,每一个打包单元的数据包格式为:

数据体的开始部分为对象管理信息,来自对象管理模块的一部分必要信息,如对象标识,版本数等.

紧接着这个对象的全部版本内容,按创建时间排序,每一个版本的内容由两部分组成,首先是该版本的基本信息,包括以下内容:

```
VERSION_NUM num;          /* 该版本号 */
VERSION_NUM father;       /* 父版本号 */
unsiged short len;        /* 版本数据体长度 */
unsiged short status;     /* 状态标记 */
unsiged short n_config;   /* 复合对象的构型登记锁数 */
unsiged short config[];   /* 这是维数为 n_confing 的变体数组 */
```

紧接的内容是这个版本的数据体. 由于组建版本时版本数据体的内容不发生变化,因此,在一个对象的版本树里,可能有多个版本共享一个数据体. 在传输打包时,要识别这种情况. 若该版本的 status 置有 VS_DUPDATA,则"数据体"仅是一版本号,它指明和哪个版本的数据体共享,否则,将该版本的实际数据体打入包.

第二种传输方式的数据包格式和第一种也很相近,只是在内容上对象管理信息不再是直接的拷贝,且紧跟着的是指定的版本.

两种方式的 Check_out 操作的打包总体过程是一样的,首先完成指定的对象(或对象版本)自身内容的打包,同时得到全部成员对象(或成员对象版本),再逐项进行打包操作.

Check_out 在 Client 上的解包操作是根据这次操作传输过来的全部信息,包括指定的对象(或对象版本),及其成员对象的内容. 各个对象自身的信息可以直接解包而得,完成这些分立操作后,需要根据对象版本中的构型建立引用关系.

Check_in 操作是用户在完成一次长事务前,将此次事务的操作结果提交给 Server 的操作,因此,它在对对象的处理上和 Check_out 有些不同,主要表现在:

(1) Check_in 操作的形式只定义指定对象的方式,即 Check_out 中的第一种方式,数据包的格式也一样;

(2) 在进行打包工作以前,先对对象的版本做一次筛选,那些从 Server 通过 Check_out 传送过来的版本是没必要重新送回的,另外,被删除的版本也不需送到 Server;

(3) 对一个对象数据包里的版本树进行解包时,若这个对象在 Server 的公用数据库上是已经存在的,则必须将两个版本树溶合到一起,具体的步骤是:

(a) 对数据包里的版本树进行解包时,先不建立和公用数据库上对应对象版本树的时间序关系,但要建立父子关系;

(b) 传输进来的版本进行版本号的重新分配；

(c) 建立和原版本树的时间序关系.

(4) 在打包操作之前，若发现存在未组建的版本，则先进行组建新版本的操作.

3 结束语

长事务和网络上的版本管理是工程设计数据管理系统的两项关键技术. 本文提出了对这两项技术的设计和实现. 它们已应用到 EDDMS 和电子 CAD 框架系统中. 随着系统的深入应用，该系统还将在实践中进一步得到完善.

参考文献

1 Silva Mario, Gedye David, Kata Randy *et al*. Protection and versioning for OCT. 26th ACM/IEEE DAC Las Vegas, 1989. 264～269.

2 Babich Wayne A. Software configuration management. Addison Wesley, 1986.

3 Vanks Steve. A configuration management system in a data management framework. 28th ACM/IEEE DAC, 1991.

4 Gray J *et al*. Transaction processing: concepts and technology. Morgan Kaufman, San Mateo, CA, 1993.

5 Kemper A *et al*. Object oriented database management system. Prentice Hall Inc., 1994.

6 Gupta Rajiv, Horowitz E. Object—oriented databases with applications to CASE networks and VLSI CAD. Pretice Hall Inc., 1991.

LONG TRANSACTION PROCESSING AND NETWORK VERSION MANAGEMENT WITHIN A ECAD FRAMEWORK SYSTEM

Pan Xuezeng　Shou Yucheng　Zhao Yuping　Ping Lingdi

(*Department of Computer Science and Engineering　Zhejiang University　Hangzhou* 310027)

Abstract　It is well known that object—oriented data management will play a key role in engineering design data management system which is the base of a CAD framework. In this paper, the authors present their design and implementation of long transaction management, concurrency control, version management on network, etc. They have practically applicate them to their ECAD framework.

Key words　Computer aided electronic design, object—oriented, framework system, long transaction procession, network version management.