文章编号:1001 - 9081(2006)07 - 1520 - 03

一种基于扩充相容性封锁的多版本两阶段封锁协议

李陶深,甘秋玲,宋庆祯
(广西大学 计算机与电子信息学院,广西 南宁 530004)
(tshli@gxu.edu.cn)

摘 要:分析了现有的分布式并发控制机制的不足,提出一种基于有序相容性多粒度的多版本两阶段封锁协议,并对其正确性进行了证明。分析表明,该协议可有效提高协作设计事务的并发度,比较适合于开放式的协同设计环境。

关键词:两阶段封锁协议;多版本;并发控制;扩充相容性封锁;协作设计
中图分类号: TP311.52 **文献标识码**:A

Multi-version two-phase locking protocol based on extended compatibility locking

LI Tao-shen, GAN Qiu-ling, SONG Qing-zhen
(*School of Computer and Electronic Information, Guangxi University, Nanning Guangxi* 530004, *China*)

Abstract: The defects of current distributed concurrency control protocol were analyzed. Then, a multi-version two-phrase locking protocol was proposed based on ordered sharing, and the correctness of this protocol was proofed. The analysis results show that the protocol can enhance the concurrency of cooperative design transaction and is suitable for open cooperative design environment.

Key words: 2PL (two-phase locking) protocol; multi-version; concurrency control; extended compatibility lock; cooperative design

0 引言

现代协作设计要求建立一个协同工作环境,以改善人们信息交流的方式,消除或减少人们在时间和空间上的相互分隔的障碍,节省工作人员的时间和精力,提高群体工作质量和效率[1]。为更好地支持协作设计过程和管理设计数据,需要有一个强大的数据库及事务处理系统来支撑。

毫无疑问,协作设计环境是一种多用户环境,作为协作设计系统重要组成部分的分布式工程数据库,其任务之一就是要协调多个用户同时对共享数据对象的访问,而这一任务主要由事务管理中的协调机制来完成。支持协作设计的事务管理的协调机制主要包括并发控制、冲突消解以及版本控制,本文着重讨论协作设计事务的并发控制机制。

传统的并发控制机制有两阶段封锁(Two-Phase Locking, 2PL)、多粒度封锁、乐观的并发控制机制、时间戳排序、多版本并发机制等[2]。但是,与传统事务要求的冲突互斥访问共享对象不同,协作事务要求对设计对象进行协作访问,即遵循某一协作规则的设计活动(如处于同一协作小组中)可同时对同一对象进行操作。传统的并发控制方法极大地限制了协作的开展,并降低了系统的并发度,而协作环境下设计活动对同一对象同时进行操作的可能性大大增加,其引起的频繁的操作冲突更加剧了上述问题[3]。因此有必要对传统并发控制进行改进,以使其能支持协作设计。

文献[4]提出了一种基于有序相容性多粒度封锁,它被证明比较适合于分布式工程数据库的并发控制,具有较好的协作能力。在开放式环境中,用户查询是很频繁的,且希望能够迅速得到查询结果,但在基于有序相容性多粒度封锁机制中没有对只读事务和更新事务进行区分,只读事务与更新事务的冲突导致查询难以快速得到结果,很难达到用户的要求,而且将会严重影响更新事务的执行效率。

文献[2]认为多版本并发机制比较适合于长事务,但多版本 2PL 采用强 2PL,而在开放式环境下,用户互操作非常频繁,且协作事务大多是长事务,因此它具有与传统的 2PL 同样的缺点,严格限制了协作设计事务的并发性,不能很好地适合于开放式的协作环境。本文提出一种基于扩充相容性多粒度锁的多版本 2PL 协议,比较适合于开放式的协作环境。

1 扩充相容性多粒度锁模型

简单读写锁模型只区分两种锁类型:共享锁(S/R 锁)和排它锁(X/W 锁)。由于工程设计事务的特殊性,这种简单的读写锁模型不能很好地适应工程设计的需要。为此,我们增加了升级锁(U 锁),并根据多粒度封锁机制增加了 IR,IU,IW 三种意向锁[5]。各种锁及其工作方式定义如下:

R - 读锁。用于设计事务读设计对象。

U - 升级锁。用于设计事务先读取设计对象然后对其进行修改的操作,它与 R 锁有序相容,即允许其他事务读此对象。

W - 写锁。用于设计事务独占访问设计对象。

IR - (Intend Read)。用于对要上 R 锁的设计对象的所有祖先节点进行封锁。

IU - (Intend Upgrade)。用于对要上 U 锁的设计对象的

收稿日期:2006 - 01 - 08;**修订日期**:2006 - 04 - 06
基金项目:广西"新世纪十百千人才工程"专项基金项目(桂人字 2001213 号);广西自然科学基金项目(桂科自 0229008)
作者简介:李陶深(1957 -),男,广西邕宁人,教授,主要研究方向:分布式数据库、计算机网络、CAD; 甘秋玲(1975 -),女,广西玉林人,助教,硕士,主要研究方向:分布式数据库; 宋庆祯(1979 -),女,陕西咸阳人,助教,硕士,主要研究方向:分布式数据库.

所有祖先节点进行封锁。

IW－(Intend Write)。用于对要上 W 锁的设计对象的所有祖先节点进行封锁。

基于上述几种锁以及设计需要,相容矩阵定义如表 1 所示。

表 1　扩充相容性多粒度锁模型的相容矩阵

行	列					
	R	U	W	IR	IU	IW
R	+	⊕	－	+	⊕	－
U	－	⊕	－	－	⊕	+
W	－	－	－	－	－	－
IR	+	⊕	－	+	+	+
IU	－	⊕	－	+	+	+
IW	－	－	－	+	+	+

+表示相容,－表示不相容,⊕表示有序相容。

基于这一相容性矩阵,扩充的相容性多粒度锁模型可以提供以下几种访问 CAD 数据的方式:1)当读取的数据直接影响设计结果时可以选择 R 锁;2)当允许被授权事务读取或协作修改尚未提交事务的已修改数据时,可以选择 U 锁;3)当事务欲对数据对象进行长期独占访问时可以选择 W 锁。此锁模型不仅允许被授权事务协作读尚未提交事务修改的数据,而且允许被授权事务协作修改尚未提交事务已修改的数据,能够较好地支持工程设计中的用户协作设计,提高协作能力。各种意向锁的设定可防止对节点的外层节点加上不相容的锁,有利于提高协作事务的并发度。

扩充的相容性多粒度锁模型允许一个对象在加了“U”锁后,仍有可能被其他事务读取,为了使上述模型能够保证事务执行的可串化,该锁模型还应该遵守文献[6]给出的有序相容性加锁规则和有序相容封锁释放规则。遵守了这两条规则,可以证明多粒度锁模型能保证事务执行的可串化[6]。

从扩充相容性封锁的特点来看,它已具备了一定的支持协作的能力,因此可用它来扩充已有的多粒度封锁机制,从而设计出可以支持协作的适用于工程设计的封锁机制。

2　基于扩充相容性多粒度锁的多版本 2PL 协议

基于扩充相容性多粒度锁的多版本 2PL 协议的基本思想是:将只读事务和更新事务分别处理,只读事务在执行时遵守多版本时间戳排序协议,而更新事务的执行将按基于两阶段有序相容性封锁的多粒度封锁协议进行;同时,为数据项的每个版本配置一个时间戳,该时间戳是一个计数器(称为 *tcounter*)用于在事务提交时增加计数。假设 $TS(T_i)$ 表示事务 T_i 的时间戳,基于扩充相容性多粒度锁的多版本 2PL 协议的描述如下:

1)在只读事务开始执行前,取 *tcounter* 的当前值作为该事务的时间戳。只读事务在执行读操作时遵从多版本时间戳排序协议。当只读事务 T_i 发出读操作 read(V) 时,返回值是时间戳小于 $TS(T_i)$ 的最大时间戳的版本的内容。

2)更新事务按基于两阶段有序相容性封锁的多粒度封锁协议进行操作。创建事务新版本时,新版本时间戳最初置为 ∞。事务 T_i 提交时,T_i 将它创建的每一版本的时间戳设为 *ts-counter* 的值加 1,然后每一次只允许有一个更新事务提交。假设数据对象为 X,更新事务操作时应遵守以下封锁协议:

a)更新事务的所有操作在访问数据对象 X 前必须对数据对象加锁;

b)加锁操作应先对数据对象 X 的所有祖先节点按从上到下的顺序加相应的意向锁,成功之后才能给数据对象加相应的锁;

c)当事务对数据对象 X 申请的锁与该对象上已持有的锁相容时,则同意加锁;

d)当事务对数据对象 X 申请的锁与该对象上已持有的锁不相容时,如果它申请的锁与已持有的锁有序相容,则以有序相容方式加锁;

e)一个事务只有在提交或夭折后才能释放锁,释放锁时从数据对象 X 开始到其所有祖先依次释放锁;

f)如果事务 T_j 以有序相容性封锁方式从事务 T_i 处获得数据对象 X 的锁,则在事务 T_i 释放前,T_j 不能释放任何锁;

g)如果一个事务 T_j 并发地修改尚未提交事务 T_i 的修改值,则事务 T_j 必须在 T_i 提交之后才能提交;

h)事务 T_i 开始释放锁之后,它不能再获得新锁。

3)在事务 T_i 的 *ts-counter* 值增加后启动的只读事务将可看到被 T_i 更新的值,而那些在 T_i 的 *ts-counter* 值增加之前就启动的只读事务看到的是被 T_i 更新之前的值。无论哪种情况,只读事务不必等待。

4)假设某数据项的两个版本 V_k 和 V_j 的时间戳都小于系统中最老的只读事务的时间戳,则 V_k 和 V_j 中较旧的版本将不再被使用,可将其删除。

3　协议的正确性证明

定义 1　假设一个数据库是一个对象的集合,则一个事务 T_i 是一个次序对 $(\Sigma_i, <_i)$,其中 Σ_i 是事务 T_i 的操作集,$<_i$ 是这些操作的执行次序。T_i 的读或写记为 $r_i[x]$ 或 $w_i[x]$。

定义 2　事务的描述为:一个只读事务只包含读操作,而一个更新事务包含读和写操作。事务以一个提交或夭折结束。

定义 3　假设 $T = \{T_1, T_2, \cdots, T_n\}$ 是一个事务集,则 T 的历史是一个偏序 $(\Sigma, <_H)$,其中 Σ 是事务执行的操作集,$<_H$ 是这些操作的执行次序。

定义 4　两个历史 H_1 和 H_2 是冲突的等价于:

1)它们被定义在同一事务集上;

2)若 O_1 和 O_2 是两个冲突操作且 $O_1 <_{H1} O_2$,则有 $O_1 <_{H2} O_2$。

定义 5　如果历史 H_s 中每两个事务 T_i 和 T_j 中存在这样的关系:要么 T_i 所有的操作在 T_j 之前,要么在 T_j 之后,则称历史 H_s 是一个串行历史。如果一个历史 H 等价于一个串行历史,则历史 H 是可串化的。

历史 H 的可串行性的判定一般使用串行图,历史 H 的串行图记为 $SG(H)$。这是一个有向图,图中的节点由 H 中已提交的事务构成。如果 T_i 的一个操作在 T_j 的一个操作之前且与之冲突,则有一条从 T_i 指向 T_j 的有向边(记为 $T_i \rightarrow T_j$)。

定义 6　若一个历史 H 的串行图是非循环的,则历史 H 是可串化的。

在一个多版本数据库中,一个对象 x 的每个写操作产生一个 x 的新版本。这样,对于每个对象 x,有一个版本的列表 $x_1, x_2, \cdots, x_n$,其中 x_i 表示被事务 T_i 创建的版本。

定义 7　一个多版本历史(MV)是已被事务提交的对象版本上的一系列操作。一个写操作 $w_i[x]$ 映射成 $w_i[x_i]$,每个读操作 $r_i[x]$ 映射成 $r_i[x_k]$,对某个 k。若在 H 中,$r_j[x_i] \in H$,则一个事务 T_j 从事务 T_i 读 x。

定义 8　如果一个 MV 历史等价于在单一版本数据库上执行的同一事务集的串行历史,则它是 1 拷贝可串化的。

一个 MV 历史 H 的多版本串行图($MVSG(H)$)是一个有向图,图中的节点代表已提交的事务。对每个对象 x,有一个总次序(记为 $\ll_x$)在所有写 x 的事务上。如果出现下列情况,则在 $MVSG(H)$ 中存在一条从 T_i 指向 T_j 的边($T_i \rightarrow T_j$):

1) $r_j[x_i] \in H$,对某个 x(即,事务 T_j 从事务 T_i 读 x);

2) 对某个 x 和 T_k,T_i 从 T_k 读 x 且 $x_k \ll_x x_j$;

3) 对某个 x 和 T_k,T_k 从 T_j 读 x 且 $x_i \ll_x x_j$。

第一条边称为 $SG(H)$ 边,后两条边称为版本次序边。若一个 MV 历史的多版本串行图是非循环的,则 MV 历史是 1 拷贝可串化的。

定理 如果一组事务的调度遵守基于扩充相容性多粒度的多版本 2PL 协议,则该调度是 1 拷贝可串化的。

证明 该定理等价于如果一组事务的调度遵守基于扩充相容性多粒度的多版本 2PL 协议,则由这组事务构成的多版本串行图 $MVSG(H)$ 是非循环的。我们通过证明这组事务构成的多版本串行图 $MVSG(H)$ 是非循环的来证明此定理。

每个事务分配一个唯一的时间戳 $ts(T_i)$,c_i 表示提交操作。

1) $T_i \rightarrow T_j$ 是 $SG(H)$ 的一条边,即 $w_i[x_i] <_H r_i[x_i]$。

这样,根据有序相容性多粒度 2PL 的规则,我们有 $c_i < c_j$,因此 $MVSG(H)$ 不会产生循环。

2) $T_i \rightarrow T_j$ 是一条版本次序边,因为对某个 x 和 T_k,$r_i[x_k] \in H$ 且 $x_k \ll_x x_j$。对每个 $w_j[x_j]$ 和 $r_i[x_k]$,如果 $w_j[x_j] <_H r_i[x_k]$,则 $c_j < c_k$,如果 $r_i[x_k] <_H w_j[x_j]$,则 $c_i < c_j$。根据 $\ll$ 的定义,$c_k < c_j$,所以 $c_j < c_k$ 是不可能的,因此,我们有 $c_i < c_j$。

3) $T_i \rightarrow T_j$ 是一条版本次序边,因为对某个 x 和 T_k,$r_k[x_j] \in H$ 且 $x_i \ll_x x_j$。因为 $x_i \ll_x x_j$,根据 $\ll$ 的定义,我们有 $c_i < c_j$。因此,$MVSG(H)$ 是非循环的。

证毕。

4 实验与分析

4.1 实验环境与参数

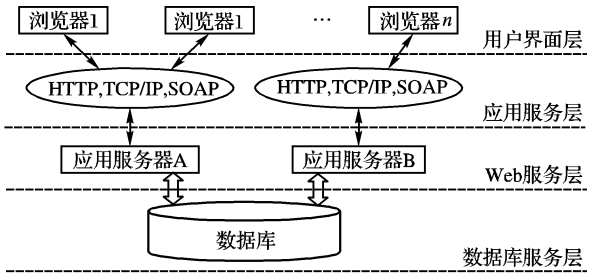

图 1 开放式环境下的协同设计实验系统结构

为了验证基于有序相容性多粒度的多版本 2PL 协议的正确性和可行性,我们构建了一个四层结构的协同设计实验环境,进行了一些模拟实验。该实验环境的体系结构如图 2 所示,包括用户界面层、Web 服务层、应用服务层和数据库服务层。其中,用户界面层用于为用户提供友好的接口;Web 服务层负责将用户的请求用 SOAP 协议请求格式传给应用服务层;应用服务层完成对用户请求的处理和将结果传送给用户。其中的协同事务处理部分包括并发控制机制、事务提交机制、事务恢复机制、检入/检出机制、消息收发机制等;数据库服务层用于存储协同设计的数据。

该实验系统的 Web 服务器和应用服务器均采用 Windows Server,应用层采用简单的浏览器,数据服务层采用 MS SQL Server 2000。系统为用户提供只读、读、更新、写四种操作。数据库放有三个数据对象:usera,userb 和 userc。其中 usera 是 userb 和 userc 的父对象。

我们用 Microsoft Visual Studio. net 中提供的 Application Center Test 测试进行大规模访问时 Web Service 执行的情况。实验将把基于有序相容性多粒度的多版本 2PL 协议与文献[4]提出的基于有序相容性封锁的多粒度封锁机制作比较。用作比较的参数如表 2 所示。

表 2 实验参数的基本设置

变量名	变量含义	取值
Nt	事务的个数	100
Nb	浏览器每秒连接数	5
t	单个事务的单独执行时间	15s/0.5s

4.2 实验结果及分析

我们启动 100 个事务,浏览器每秒同时的连接数为 5 个,单个事务单独的执行时间为 15s(事务较长)。事务完成率为成功完成的事务数与启动的事务数的比率。

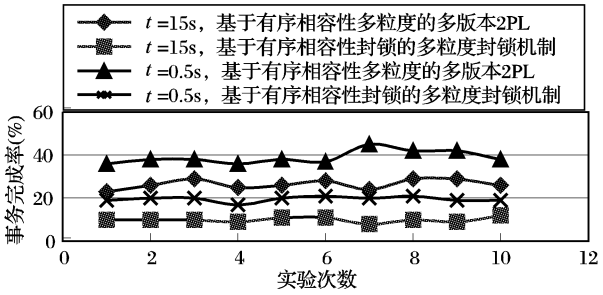

图 2 事务完成率的比较

从图中可以看出,在相同的实验环境下,不仅基于有序相容性多粒度的多版本 2PL 协议的较短事务(t =0.5s)的事务完成率比基于有序相容性封锁的多粒度封锁机制的事务完成率要高,而且其较长事务(t =15s)的事务完成率也比基于有序相容性封锁的多粒度封锁机制的事务完成率高。因此,实验表明新协议确实比较适合于上述开放式的协同设计环境。

5 结语

传统封锁机制的锁模型过于简单,无法表达协作设计丰富的语义信息,而基本两阶段封锁协议无法实现设计中的协作要求。本文对简单锁模型进行了扩充,使用了多粒度分层封锁,并结合对协作具有一定支持能力的有序相容性封锁规则作为封锁协议的基础,提出了一个基于扩充相容性多粒度的多版本 2PL 协议。该协议主要是从工程协作设计事务的角度来考虑的,可以为协作读事务和协作修改事务提供有效的支持:

对协作读事务的支持 该模型允许更新事务在修改某数据时,可授权多个事务读取该数据。

对协作修改事务的支持 工程设计过程中对同一产品的合作修改可根据协作的程度分为对同一产品的不同组件进行修改、对同一组件的不同零件进行修改和对同一产品的相同组(零)件进行修改。该模型的分层封锁机制保证了每次封锁都是选择最佳封锁粒度进行封锁,直接支持了允许对同一产品的不同组件或对同一组件的不同零件同时进行修改这两种协作设计。

经过证明,我们的提出的扩充相容性多粒度封锁机制能够保证事务执行的可串化。实验结果分析表明,该机制确实比较适合于开放式协作环境下的协作事务的并发控制。

(下转第 1538 页)

间路由器简化为两条独立的信道,一条用于发送数据,另外一条用于发送 ACK 包。

4)忽略数据包头部的其他字段,只考虑头部中的序列号字段。

4.2 CPN 模型

SUNA 可靠数据传输机制 CPN 模型如图 5 所示。模型包括三个部分:发送方、接收方以及传输数据的信道。发送方包括两个位置:“S_win”和“NextSend”以及三个变迁:“SendData”,“timeout”和“ReceiveAck”。接收方包括一个位置“R_win”和一个变迁“ReceiveData”。信道包括两个位置“DataChannel”和“AckChannel”。

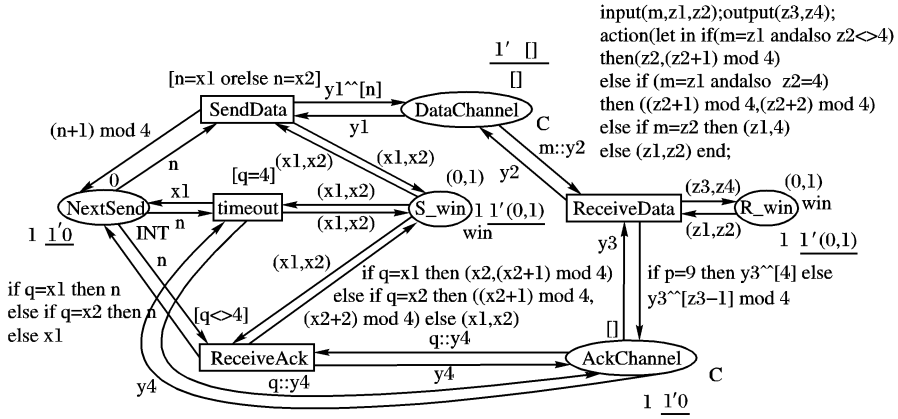

图 5 SUNA 可靠数据传输机制 CPN 模型

模型的左边描述发送方行为。位置“S_win”表示发送窗口的状态,它的初始标识是 1′(0,1),表示会话开始时发送窗口的下界是 0,上界是 1。位置“NextSend”表示发送方发送的下一个数据包,由 ACK 的值和当前发送窗口共同决定。位置“NextSend”的初始标志是 1′0,表示发送方发送的第一个数据包是 0#数据包。变迁“SendData”将数据包发送到信道上,并且返回下一个数据包的包号。变迁“timeout”通知发送方重传丢失的数据包。变迁“ReceiveAck”从信道接收 ACK,接着改变滑动窗口上、下界,并且返回下一个数据包的包号。

模型的右边描述接收方行为。位置“R_win”表示接收窗口的状态,它的初始标识是 1′(0,1),表示会话开始时接收窗口可以接收 0#数据包和 1#数据包。变迁“ReceiveData”从信道接收数据包,接着改变接收窗口,并且向信道发送 ACK。

模型信道部分包括两个位置。位置“DataChannel”的 Token 值表示正在从发送方向接收方传输的数据包序列号,位置“AckChannel”的 Token 值表示正在从接收方向发送方传输的 ACK 包序列号。

5 状态空间分析

CPN Tools提供的一个重要工具就是状态空间分析工具。通过状态空间报告可以对系统的有界性(Boundedness)、回归性(Home)、活性(Liveness)以及公平性(Fairness)进行分析。

表 1 SUNA 可靠数据传输机制 CPN 模型状态空间分析的部分报告

Statistics	Liveness Properties	Home Properties
Nodes: 749 Arcs: 1648 Secs: 0 Status: Full	Dead Markings: None Dead Transitions Instances: None Live Transitions Instances: All	Home Markings: All

表中第 1 列描述了状态空间的统计信息。数据显示,本模型的状态空间包括 749 个节点和 1 648 条弧;生成状态空间所用的时间不到 1s。第 2 列描述了状态空间的活性。本模型没有死标识和死变迁,所有变迁都是活的。因为当发送数据序列号为滑动窗口上界时,发送方继续发送序列号为滑动窗口下界的数据,这是一个不中断的循环发送过程。第 3 列表明所有标识都是回归标识,因为整个传输过程是一个循环过程,经过若干步后总会回到当前标识。以上报告显示,SUNA 可靠数据传输机制是完全正确、可靠的,满足系统的有界性、回归性以及活性等方面的要求。

参考文献:

[1] TENNENHOUSE D, WETHERALL D. Towards an Active Network Architecture[J]. Computer Communication Review, 1996,26(2).

[2] LAZAR A. Programming Telecommunication Networks [A]. Proceedings of 5th International Workshop on Quality of Service [C]. New York, USA, 1997.3-24.

[3] BOECKING S. 面向对象的网络协议[M]. 严伟,译. 北京:机械工业出版社, 2000.

[4] BOECKING S, SEIDEL V, VINDEBY P. CHANNELS - A Run-Time System for Multimedia Protocols[A]. 4th International Conference on Computer Communications and Networks[C]. Las Vegas, USA, 1995.178-185.

[5] BRADEN B, FABER T, HANDLEY M. From Protocol Stack to Protocol Heap-Role-Based Architecture[A]. First Workshop on Hot Topics in Networking[C], 2002.

[6] 曾家智,徐洁,吴跃,等. 服务元网络体系结构及其微通信元架构[J]. 电子学报, 2004, 32(5).

[7] JENSEN K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume1 - Basic Concepts[M]. Springer-Verlag, 1992.

[8] JENSEN K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume3 - Practical Use[M]. Springer-Verlag, 1997.

(上接第 1522 页)

参考文献:

[1] 史美林,向勇,杨光信,等. 计算机支持的协同工作理论与应用[M]. 北京:电子工业出版社,2000.

[2] SIBERSCHATZ A, KORTH HF, SUDARSHAN S. Database System Concepts(Fourth Edition)[M]. 北京:机械工业出版社,2003.

[3] 邵佩英. 分布式数据库系统及其应用[M]. 北京:科学出版社,2000.

[4] CHEN G, LI T, LIAO G. A Multi - granularity Locking Protocol Based on Ordered Sharing Locks in Engineering Databases that Supports Cooperative Design[A]. 2004 International Symposium on Distributed Computing and Applications to Business, Engineering and Science(DCABES 2004)[C], 2004. 552-558.

[5] 李陶深,廖国琼. 支持工程合作设计事务的扩充分层封锁机制[J]. 计算机应用与工程,2002,38(7):47-49.

[6] 齐进,张家明,周伯鑫,等. 工程数据库中一种支持合作设计事务的并发控制机制[J]. 计算机研究与发展,1998,35(11):987-990.