一种乐观嵌套工作流事务模型

董云卫[1] 郝克刚[2]

(西北工业大学计算机学院 西安 710072)[1] (西北大学计算机科学系 西安 710069)[2]

摘 要 为解决目前工作流事务管理的不足和存在的问题,本文基于多数据版本、三阶段执行的并发控制理论提出了一种乐观嵌套工作流事务模型,该事务模型借用时间戳的概念,通过对不同事务中活动类型的分类,较好地解决了长执行事务和协同事务的可靠性和正确性问题,提高了工作流处理的效率。乐观嵌套事务模型把嵌套事务、工作流模型和并发控制协议有机地结合在一起,定义了较为完整的事务操作原语及其语义。本文还给出了乐观事务模型到工作流模型的映射,使得事务工作流执行过程中,其操作原语和乐观事务模型的操作原语是一致的,工作流活动的转移控制与乐观嵌套事务模型的子嵌套事务的生成过程及其表示方式也是一致的。

关键词 乐观嵌套事务模型,工作流事务,扩展信牌驱动分布式工作流计算模型

An Optimistic Nested Transaction Model in Workflow

DONG Yun-Wei[1] HAO Ke-Gang[2]

(School of Computer Science, Northwest Polytechnic University, Xi'an710072)[1]

(Department of Computer Science, Northwest University, Xi'an710069)[2]

Abstract An optimistic nested transaction workflow model is presented to solve some problem which is existed among workflow transaction management in this paper. The optimistic nested transaction workflow model adopts many technologies, such as multi-data version, three-phase concurrent control protocol and timestamp. It sorts workflow activities to assure reliability and correctness of long-running transaction and/or collaborative transaction, and to get a high level efficiency of system ability of computing. The optimistic nested transaction workflow mode defines completely primitives and semantic transaction operator. A mapping is also presented from optimistic nested transaction workflow mode to Extend Xinpai driven Distributed Workflow Model in this paper. The optimistic nested transaction workflow model is consistent with Extend Xinpai driven Distributed Workflow Model in workflow process running-time and building-time.

Keywords Optimistic nested transaction model, Workflow transaction, Extend xinpai driven distributed workflow model

1 前言

事务处理的概念起源于数据库系统,在以数据为中心的系统中,事务作为操作的基本单元必须满足 ACID 属性,即原子性(Atomicity)、一致性(Consistency)、孤立性(Isolation)和持久性(Durability)。而工作流事务管理是以过程为中心,保证工作流应用的可靠性和正确性。早期工作流建模方法是通过放松事务的基本特性来实现的,这样的工作流事务模型通常称为扩展事务模型或高级事务模型。

在工作流事务模型设计中,要考虑的问题一般要分两个层次:一是实现每一个操作的正确性和可靠性,这是数据库系统中关注的主题;二是实现每一个活动中许多操作之间流的管理。因此,表示过程活动执行的事务与传统的数据库事务具有显著不同的特点是[1~4]:

1)长时间执行:活动事务的执行时间长,事务执行的周期和工作量在许多情况下难以估计,可能几天、几月,甚至数年,称为长执行事务(long running transaction);

2)协同性:工作的事务中的活动执行的结果互相影响,并且,这些活动可能分布在异构环境中不同的系统中,称为协同事务(Collaboration transaction)。

3)可见性:中间和最终结果为多个合作者间共享和交换,是工作流事务必需的。它常常要求未提交的数据提前开放,并同时保证数据的一致性和正确性。

在工作流应用中,人们为了满足工作流事务管理的需要,

董云卫 博士,副教授,研究兴趣:工作流技术、面向方面的开发方法、软件测试技术。

流技术成功推广应用的重要意义。通过引入工作流网,工作流日志,工作流事件排序关系等相关概念,提出了一种基于工作流日志工作流重构算法。该算法能够处理干扰数据,因而能适应于大多数实际的工作流。重构算法包括:构建依赖-频率表;推导依赖-频率图;工作流网构建三个部分。在实验中,该算法能够对大多数业务流程的进行重构。当然,该算法仍有局限性,特别在识别工作流模型中复杂模式时,仍存在不完整性,有待进一步的研究。

参 考 文 献

1 Baldan P. Modelling concurrent computations: From contextual Petri nets to graph grammars:[Ph. D. thesis. TD-1/100]. Dipartimento di Infomatica University of Pisa, 2000

2 Weijters T, van der Aalst W M P. Process Mining: Discovering Workflow Models from Event-Based Data. In: Krse, B. et. al, eds. Proc. 13th Belgium-Netherlands Conf. on Artificial Intelligence (BNAIC01), 25-26 October 2001, Amsterdam, The Netherlands, 2001. 283~290

3 van der Aalst W M P, Desel J, Oberweis A, eds. Business Process Management: Models. Techniques, and Empirical Studies, volume 1806 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2000

4 Herbst J. A Machine Learning Approach to Workflow Management. In: 11th European Conf. on Machine Learning, volume 1810 of Lecture Notes in Computer Science, Springer, Berlin, Germany, 2000. 183~194

设计了一些事务模型，如 Sagas 模型、Flexibility 模型和 ConTract 模型等，这些模型为工作应用提供了较好的事务管理机制，但还存在以下缺点：

• 大多数工作流都是基于补偿的原理实现对事务管理的可见性和长时间执行的处理，它存在一个致命的假设：所有补偿事务都能够正确执行，实际上，补偿事务与一般事务一样也存在失败的可能；另外，有的事务可能就不能补偿或不存在补偿事务。

• 有的工作流的事务处理是基于代理机制，它需要用户为每个活动设计代理或补偿活动，这在实际应用中往往不可行，因为用户不是软件工程师，它不能较好地理解事务的语义和语法。

• 现有工作流事务模型都是基于锁协议，而在工作流事务管理中死锁的监测与工作流活动的控制交织在一起，是一件非常困难的事；

• 不同事务模型之间不能实现互操作，因为，在不同模型中的补偿事务的定义和代理机制的设置是不一样的。

为满足工作流事务管理的需要本文提出乐观嵌套事务模型，以解决现存的工作流事务模型的缺点。所谓的“乐观”是指事务不需要通过“等待或回滚”等“悲观”方式（如锁机制）实现事务的调度执行的机制，下面对该模型的思想、语义及规则进行论述。

2 乐观嵌套事务

在工作流应用中多数并发事务的操作是不会发生冲突的，一方面事务读操作与读操作是不会冲突，不对相同资源竞争的事务也是不会发生冲突的；另一方面事务操作的大部分工作会分布到不同的客户机上执行（特别是包含人机交互的事务），并且事务的执行时间会很长，在对相同资源竞争的并发事务之间，无论是读操作和写操作之间还是读操作与读操作之间，只有在对数据状态要改变的阶段（写阶段）才会导致系统状态的不一致，因此，可以采用乐观并发控制协议，把事务分为两个或三个阶段执行，不但能够大大提高系统的并发能力，还能够保持系统的正确性和一致性[5,6]。

工作流过程定义具有嵌套的特点，工作流事务模型要求嵌套事务的机制。在乐观嵌套事务模型中一子事务成功结束时，它将把其执行结束通知给自己的父事务，并根据父母事务的依赖关系由父事务决定该事务是提交成功还是失败。子事务不等同于一般的事务，它不需要完全满足事务的 ACID 属性。从父母事务的角度看是具有原子性的，从他们实现的局部功能看是保持一致性的，都孤立于其父母事务内部和外部的事务，但是其提交规则不满足持久性原则[7]，同时，子事务访问结果可对其父事务和非并发子事务块中的兄弟事务开放，从其父母或非并发的兄弟事务的角度看具有可见性。嵌套事务的顶层事务完全满足 ACID 属性，从事务的外部看，顶层事务的中间结果是不可见的，因而是原子的，它实现的功能也保持一致性和孤立性，与子事务不同的是顶层事务是持久的，如果事务提交，它的执行结果会持久保存在系统中。

3 事务控制

乐观嵌套事务模型整体上采用乐观并发控制协议，由于读操作不会影响数据的完整性，因此对读操作不做任何控制，事务的读操作能够自由访问数据，但在事务提交前需要进行有效性检查，以判断它提交的值是否正确（已读取的数据是否“过时”）；写事务分为三个阶段：读阶段、有效性检查阶段和写阶段。

对于事务访问的数据组织采用多版本控制协议，为每一个事务建立一个它或它的子事务需要访问的数据版本。事务的内部结构采用嵌套事务机制，事务所访问的数据项也嵌套处理。每一级事务在对某数据第一次读、写操作时，都根据上一级的该数据版本，建立属于本级事务的该数据版本；如果上一级的该数据版本不存在，还要根据上上一级的该数据版本，建立上一级和本级事务的该数据版本，依次类推。每一级事务的读、写访问都是在本级事务数据版本中进行。当子事务成功提交时，用该事务的数据版本更新其父事务数据版本中的相应数据；子事务失败后不进行提交，从而其父事务的数据版本不变，相当于进行了恢复。

为了明确地表示每一个事务 T_i 的并发执行的控制情况，我们给每一个事务指定一个唯一的事务号，并且在事务执行时赋予事务进行有效性验证阶段的时间戳 Validation(T_i)。对事务进行 T_j 有效性验证就是要求任何满足 Validation(T_i) < Validation(T_j) 的事务 T_i 必须满足下列条件之一：

i. 事务 T_i 完成写阶段在事务 T_j 读阶段开始之前；

ii. 事务 T_i 所写操数据项集合与事务 T_j 所读数据的集合不相交，并且事务 T_i 完成写阶段在事务 T_j 写阶段开始之前；

iii. 事务 T_i 所写的数据项集合与事务 T_j 所读的数据项集合和所写的数据项集合均不相交，并且事务 T_i 完成读阶段在事务 T_j 读阶段完成之前；

i)说明事务 T_i 在事务 T_j 之前完成；ii)说明事务 T_i 不影响事务 T_j 的读阶段，并且事务 T_i 在事务 T_j 开始写操作之前完成写操作；iii)说明事务 T_i 不影响事务 T_j 的写阶段和读阶段，而且事务 T_j 不影响事务 T_i 的读阶段和读阶段。因此，时间戳排序机制通过进行事务的有效性验证可实现事务串行化执行，当 Validation(T_i)< Validation(T_j) 且满足有效性验证的条件时，所产生的任何调度必定等价于事务 T_i 出现在 T_j 之前的某个串行调度。

4 事务原语

在乐观嵌套事务模型中，一个事务有 8 个状态可选：开始、读、确认（有效性验证）、写、请求提交、提交、失败和结束，如图 1 所示。一个正在被执行的事务它将处于 8 个状态中的其中一个。我们设计了乐观嵌套事务模型的操作原语，如表 1 所示。

表 1 乐观嵌套事务的基本原语表

	事务原语	原语语义
1	事务开始 Begin_Transaction	表示一个事务开始。
2	事务结束 End_Transaction	表示一个事务结束。
3	事务读 Read [a]	事务读取数据 a，并建立自己的一个私有数据版本 $a'=a$。
4	事务确认 Validation_ Transaction	进行事务有效性验证。
5	事务写 Write [a]	事务把其私有数据版本的操作结果回写到公共数据版本中。
6	事务发通知 Send_Signal	事务写阶段执行成功后向读取了旧数据，且未提交的事务发送一个通知。
7	请求提交 Reqest_Commit	子事务执行成功结束时调用，其目的子事务向它的父母事务请求提交。该原语之后跟随的是提交成功或失败原语。
8	事务提交 Commit	事务回写成功结束。
9	事务失败 Abort	读事务失败没有返回值或写事务回写失败，删除其私有数据版本。

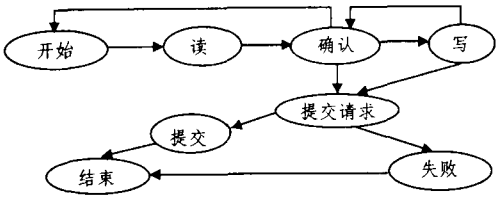

图 1 乐观事务模型中事务执行状态图

一般地，工作流控制逻辑可以归纳为 20 种模式[8]，无论是事务模型还是工作流模型，都需要有刻画这些控制逻辑模式的能力，西北大学郝克刚教授等人通过对 Petri Nets 理论进行简化，结合工作流的特点提出了信牌驱动分布式计算工作流模型[9]。为了使信牌驱动分布式计算工作流模型的规则与工作流国际组织 WfMC 发布的标准 WPDL 相一致，对信牌驱动分布式计算工作流模型进行扩展[10]，该流模型具有更加丰富的语义和直观的描述能力，能够描述各种业务流程的控制逻辑。乐观嵌套事务模型是以信牌驱动分布式计算工作流模型作为工作流成建模的基点来考虑的，在乐观嵌套事务模型中，除了表 1 所列举的事务原语是父事务与子事务都能够执行的基本操作外，还需为父事务定义控制子事务执行的事务原语，称为扩展事务原语，如表 2 所示。

表 2 乐观嵌套事务模型的扩展事务原语

	事务原语	原语语义
10	创建并发子事务块 Block_Par_Create $[T_i, \cdots, T_j]$	一个事务创建一个子事务块 Tset = { $T_i, \cdots, T_j$ } 时调用，Tset 中的事务并发执行。
11	创建串行事务块 Block_Seq_Create $[T_i, \cdots, T_j]$	一个事务创建一个子事务块 Tset 时调用，并且这些子事务按顺序 $T_i < \cdots < T_j$ 串行执行。
12	创建选择执行事务块 Block_select_Create $[T_i, \cdots, T_j]$	一个事务创建一个子事务块 Tset 时调用，并且这些子事务只有一个执行成功
13	创建占优执行事务块 Block_dominant_Create $[T_i, \cdots, T_j]$	一个事务创建一个子事务块 Tset 时调用，并且这些子事务有序关系 $T_i < \cdots < T_j$，只有关系前面的子事务执行失败后后面的事务才能执行，并且前面的事务执行成功提交，后面的事务将不再执行。
14	同意事务提交 Grant_commit $[T_i]$	一个事务同意其子事务 T_i 提交时调用。
15	强迫事务失败 Force_Abort $[T_i]$	一个事务强迫其子事务 T_i 失败时调用。
16	Dummy (op)	Op 是操作原语 10-13 中操作的一种或多种组合。

前面四个原语是用来创建新事务的事务原语，它们反映了工作流事务中兄弟嵌套事务之间的内部依赖关系以及它们与父母事务之间的依赖关系，后面两个事务原语是用来同意子事务的提交或强制失败。

在工作流事务中，一些事务在流程定义中是不相干的，执行时却有依赖关系。为此，我们定义了一个哑元事务(Dummy Transaction)原语，把这些事务当作是哑元事务的子事务，把事务间的依赖关系转变成为事务内的依赖关系。哑元事务只表示事务间的依赖关系不执行任何的操作，当哑元事务中嵌套的事务按照相互之间的依赖关系提交时，哑元事务也将提交，否则，哑元事务失败，所有被嵌套事务的事务都失败。

5 工作流事务原语与工作流的活动控制操作的对应关系

在乐观事务模型中，每一个事务都是以原语操作 Begin_Transaction 开始，以原语操作 End_Transaction 结束。事务的执行就是对应用数据和相关数据的操作(执行操作原语 Read [a]、Validation_ Transaction、Write [a]等)和事务执行的管理(执行操作原语 Send_Signal 、Reqest_Commit Commit 和 Abort)。事务可以是一个弱平面事务，也可以是一个子嵌套事务，对于嵌套事务或子事务，还要考虑子事务的构成和执行依赖关系，它们通过调用扩展事务原语操作来实现的。

在扩展信牌驱动分布式工作流计算模型中，一个工作流程定义表示工作流的静态结构，而工作流的执行就是它被实例化后的一次执行。工作流活动的执行也像工作流的执行一样，它需要先实例化，然后才能启动。所谓实例化就是为工作流或活动的执行所需的各种变量设定初始值。工作流活动执行的条件是其前信牌箱中有信牌，执行时从前信牌箱中取走相应数量的信牌，并执行活动中所包含的操作(如对数据库的查询、修改等)，活动成功结束时向其后信牌箱中放信牌。执行失败时，不向后信牌箱放信牌，并进行异常处理。

从上面看工作流活动与乐观嵌套事务的执行是一致的，一个叶子事务对应了一个工作非复合流活动，而一个嵌套事务则对应了工作流复合活动。从执行的过程来看乐观嵌套事务模型的处理机制除了满足工作流活动执行要求外，还要考虑并发操作的冲突可串行化(分阶段执行、加时间戳)；从执行的结果看乐观事务模型的事务成功提交与工作流活动成功结束是一样的，当不成功时，事务通过执行恢复原语操作保证数据的一致性，而没有加事务处理的工作流系统是不能做到的。因此，事务管理原语覆盖了工作活动执行的原语，其映射关系如表 3 所示。

表 3 乐观嵌套事务模型与扩张信牌驱动模型的工作流活动间操作原语的对应关系

乐观嵌套事务模型	扩张信牌驱动模型	备注
事务开始 Begin_Transaction	活动启动 Start	活动实例化，取信牌
事务结束 End_Transaction	活动结束 Finish	活动结束，传递信牌
事务读 Read [a]	读操作 或写前读操作	对象关数据和应用数据、控制数据读操作或为写而读
事务确认 Validation_ Transaction	无	
事务写 Write[a]	写操作	保存对象关数据和应用数据、控制数据读操作
事务发通知 Send_Signal	无	
请求提交 Reqest_Commit	无	
事务提交 Commit	无	
事务失败 Abort	异常处理	日志管理

在乐观嵌套事务模型中支持不同事务的并发执行，并且利用乐观并发控制协议实现事务的串行化调度，嵌套子事务之间也支持事务并发和串行执行，它们之间的依赖关系是通

过生成嵌套子事务的原语操作实现的。其对应关系如表4所示。

从表3,4可以看出,乐观嵌套事务模型中的事务原语不但能够满足工作流活动执行的操作之外,还对活动之间的依赖关系的表示进行了扩展和补充,在保证工作流系统的正确性基础上,增强工作流程建模的描述能力,也提高了工作流系统的协同处理能力。

表4 工作流复合活动状态转移原语与嵌套子事务生成原语的对应关系

工作流原语	乐观嵌套事务模型	备注
Seq-Tran	创建串行事务块 Block_Seq_Create[T_i,…,T_j]	相同
ALL-SPLIT	创建并发子事务块 Block_Par_Create[T_i,…,T_j]	相同,表示方式一
ALL-JOINT	创建并发子事务块 Block_Par_Create[T_i,…,T_j]	相同,表示方式二
XOR-SPLIT	创建选择执行事务块 Block_select_Create[T_i,…,T_j]	相同,表示方式一
XOR-JOINT	创建选择执行事务块 Block_select_Create[T_i,…,T_j]	相同,表示方式二
AND-SPLIT	Block_Par_Create[T_i,…,T_j] Block_select_Create[T_i,…,T_j]	二者组合应用
AND-JOINT	Block_Par_Create[T_i,…,T_j] Block_select_Create[T_i,…,T_j]	二者组合应用
LOOP	不支持	嵌套事务有序层次性决定了在一个事务树中不能有循环路径
	创建占优执行事务块 Block_dominant_Create[T_i,…,T_j]	对串行执行的一个补充,简化工作流活动转移关系的表示
	同意事务提交 Grant_commit[T_i]	是实现活动间协同执行的补充
	强迫事务失败 Force_Abort[T_i]	支持部分活动的执行
	Dummy(op)	实现工作流活动执行过程中的动态协同

6 模型应用举例及后续工作

图2(a)所示的旅行计划工作流事务是表述工作流事务管理的特点的典型例子。在个该事务中包含三个子活动事务:设计旅行线路、购买飞机票和预订酒店。该工作流应用要求在设计旅行线路时,需要预订酒店和购买飞机票,由于酒店预订后可以取消,但机票购买后不能退(否则收取退票费),机票购买成功,预定的酒店才有效,否则取消酒店预订,只有酒店预订和机票购买都成功,旅行线路设计才有效,整个事务成功提交。需要注意的是这三个子活动是分布在三个不同业务流程(系统)中的。传统的方法是在机票购买失败后执行酒店的补偿事务,取消预定酒店,但是如果补偿事务失败的,系统的状态就不正常了。

在乐观嵌套事务模型的运行机制中,利用原语 Dummy(op)把设计旅游 T_1、购买机票 T_2 和预定酒店 T_3 三个子活动事务定义成一个工作流事务 T_0,如图2(b)所示,事务的执行顺序是先设计旅游线路(如选择要入住的酒店、搭乘的航班),然后预定酒店、预订了酒店之后接着预订飞机票,事务提交的顺序酒店和航班选择后子事务 T_1 向 T_0 事务内提交,结果对其兄第事务 T_2 和 T_3 开放;事务 T_3 作为事务 T_2 的子事务,先购买飞机票,但它真正执行前要执行其子事务 T_3:预定酒店,T_3 完成后,也向其父事务 T_2 提交(并没有真正意义上提交到系统中),此时开始执行购买飞机票事务 T_2,事务 T_2 成功提交则 T_3 与事务 T_2 一道完成提交,否则,事务 T_2 和 T_3 都失败。同理,事务 T_0 成功提交时,事务 T_1、T_2 和 T_3 也成功提交,否则,事务 T_1、T_2 和 T_3 也提交失败。在事务 T_0 执行过程中,事务 T_1 和 T_2 的相互协同执行关系是通过事务 T_0 实现的,它们之间的操作原语都是 Block_Seq_Create[T_i,…,T_j]。同时,在每一层级中的兄弟事务的执行结构都相互开放,并且在事务 T_0 提交前,所有子事务的结果都没有真正提交到系统中。

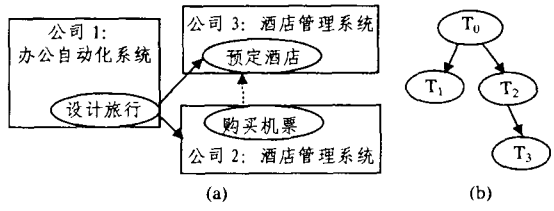

图2 工作流协同事务

本文就乐观嵌套工作流事务模型的思想和语义进行了阐述,该模型提供了较为完整的是操作原语,克服了目前广泛用于工作流事务管理的扩展事务模型的不足,既实现了事务之间和事务内部的依赖关系控制流管理的正确性,又在保证数据一致性的基础上极大地提高了事务的并发处理能力,但是由于篇幅的原因,本文并未就该模型的正确性进行论证与分析,该部分内容在另一篇论文《乐观嵌套工作流事务模型的形式化描述中》进一步论证。

参 考 文 献

1 Schuldt H, Alonso G, Beeri C, Schek Han-Jorg. Atomicity and Isolation for Transaction Processes, ACM Transaction on Database Systems, March, 2002, 27(1): 63～116

2 Wachter H, Reuter A. The ConTract Model, Database Transaction Models for Advanced Applications. Edited by A. H. Elmagarmid, Morgan Kaufmann, San Jose, 1992. 123～158

3 丁柯,金蓓弘,冯玉琳. 事务工作流的建模和分析. 计算机学报, 2003, 26(10)

4 范玉顺,等. 工作流管理技术基础. 清华大学出版社, 2001

5 Thomasian A. Distributed Optimistic Concurrency Control Methods for High- Performance Transaction Processing, IEEE Transaction on Knowledge and Data Engineering, 1998, 10(1)

6 Kung H T, Robinson J T. On Optimistic Methods for Concurrency Control, ACM Transaction on Database Systems, 1981, 6(2): 213～226

7 Eliot J, Moss B. Nested Transaction: an Approach to Reliable Distributed Computing, April 1981, MIT/LCS/RT-260

8 van der Aalst1 W M P, ter Hofstede2 A H M, Kiepuszewski2 B, Barros A P. Workflow Patterns. http://tmitwww.tm.tue.nl/research/patterns/patterns.htm, 2004

9 岳晓丽,杨斌,郝克刚. 信牌驱动分布式计算工作流模型. 计算机研究与发展, 2000, 12: 1513～1519

10 郝克刚,王斌君,安贵. WPDL 中的 JOIN 语义问题和分区解决方案. 计算机科学, 2003

11 龚晓庆,葛玮,郝克刚. 信牌驱动模型中的工作流模式. 西北大学学报, 2004