基于向后恢复的多层工作流事务管理模型
Multi-Layer Workflow Transaction Model Based on Backward Recovery

邓珍荣
Deng Zhenrong
(桂林电子工业学院计算机系,广西桂林 541004)
(Dept. of Comp. Sci., Guilin University of Electronic Technology, Guilin, Guangxi, 541004, China)

摘要:分析工作流语义故障的向后恢复技术,在改进 LOL 确认机制的基础上,整合补偿模型与原流程模型,建立基于补偿的多层工作流事务模型及其构建规则,使事务粒度可由用户灵活定制,提高系统的执行效率。给出 2 个补偿模型的实例。

关键词:工作流 多层事务 向后恢复 补偿 确认
中图法分类号:TP311

Abstract: A compensation model is generated based on compensation and confirmation, and integrated into the original process model. The resultant model could support multilayer transaction management and bear the capability of fault tolerant and flow forward. By this way, the efficiency of workflow transaction can be greatly improved.

Key words: workflow, multi-layer transaction, backward-recovery, compensation, confirmation

当前业界推出各式各样的工作流产品,但实践表明,工作流管理系统在业界的成功案例不多,其中一个重要原因就是其作为一个底层的平台很难提供类似于大型数据库管理系统的可靠度,系统缺乏对事务的支持[1]。改进后的扩展事务模型(Advanced Transaction Models,简称 ATMs)可以支持"事务工作流"[2]。然而 ATMs 的一些要求是 WfMS (Workflow Management System)无法满足的,如两阶段提交协议等等。

文献[3]提出一种说明工作流应用中原子性需求的方法,但这要求对隐含在流程中的逻辑关系进行深入的分析,加大了实现的复杂性。但是文献[3]并没有给出在事务失败时如何恢复的方法。文献[4]提出的事务模型只支持平面事务,不支持多层事务,而多层事务有利于提高性能。本文拟把补偿模型与原流程模型合并在一起来建立事务模型,并采用改进的 LOL 确认机制以支持多层的事务模型,使事务粒度可以由用户灵活定制,提高系统执行效率。

2004-04-20 收稿,2004-05-31 修回。

1 工作流语义故障的向后恢复技术

工作流的故障主要包含系统故障和语义故障。向后恢复的技术可用来处理语义故障。文献[5,6]介绍一种原型 FlowBack 及其实现 IBM MQ Workflow。该法根据 MQ Workflow 创建时输出的流程定义文件生成一个补偿流程。FlowBack 的方法是把补偿模型与原流程模型相分离,一旦原流程执行发生故障,就立即启动补偿流程。但这个方法存在如下严重问题:原流程一旦发生执行故障,就要全部作废,等补偿流程补偿之后重来。如果流程比较复杂,执行时间长,那么流程执行成功的几率就会很低,也将导致不断的重新执行,这种代价是高昂的。文献[7]提出一种叫"确认"的机制(下称 LOL 确认机制),可把许多不可补偿的任务改写为可补偿的任务,并且通过数据隔离和推迟提交,给向后恢复提供有力的支持。而 LOL 确认机制不直接支持多层事务,所以需要作一些改进以支持多层事务。

在一个流程当中可以包含若干个事务,但并不是其中一个事务执行发生故障就要全盘推翻。一般说来,如果失败事务对流程的其它部分无重大影响,从用户的角度,更希望流程能够尽可能多地完成既

定的任务。

2 基于向后恢复的工作流事务模型

在传统的事务模型中，允许存在一个共享数据的多次读写和读脏数据的情况，操作之间的数据依赖非常复杂，操作之间也容易产生冲突，从而导致事务的连环回滚。而工作流流程具有执行时间长的特点，如果事务执行成功率过低，则会严重影响整个系统的效率，这在实际情况中是无法接受的。本文通过减少读写次数、补偿和向后恢复来实现高效的细粒度事务管理。

实现补偿主要有静态的和动态的2种方法。静态方法假设工作流补偿模型的创建是在设计阶段。补偿模型可以被设计成原模型的一部分[8]或者一个分开的模型。补偿路径的获得会比较复杂，尤其是补偿模型作为原模型的一部分来实现时。动态方法能较好的解决这个问题，它是对用户最透明最优雅的解决方法。但是在许多特定的情形下，补偿模型的自动生成是十分困难的。本文采用半动态即静态与动态相结合的方法，于创建态在原模型的内部扩充补偿模型，在运行态辅以日志工具提供原模型的执行路径，即可精确的进行补偿，并且流程的某个事务执行失败，不至于整个流程都作废，而是尽可能的向前执行。

2.1 改进的LOL确认机制

LOL确认机制的思想是，把事务中的活动分成正常操作、确认和补偿三部分，事务先执行正常操作部分，如果执行成功，则在事务结束之前最后进行执行确认，否则执行补偿。对于事务中访问的共享数据 d，在事务的正常操作部分中并不直接操作 d，而是操作 d 的一个副本 d'，这样就实现了隔离。在事务结束前再进行“确认”，即把 d' 的值写到 d 中去。

为了支持多层事务从而提高并发性，需要对LOL确认机制进行改进。改进后的确认机制用如下3种规则描述：

(1)共享访问规则：在事务中访问共享数据必须采用确认机制，即在事务的正常操作部分不应该写原共享数据，而是对一个复制的副本来进行操作。

(2)分层确认规则：不同的事务层次对共享数据的不同副本进行操作，同一层次的不同事务对共享数据的不同副本进行操作。

(3)持久规则：确认活动应该只有写持久数据的功能，而不应该再包含其它的功能，并且确认应该作为事务的最后一个步骤来完成，一旦确认，本事务即宣告结束。

2.2 事务模型

工作流流程是由控制流和数据流控制的按照预先设定的先后次序执行的多个活动构成的。流程中的一些相邻活动密切相关，如果希望这些活动要么全部都被执行，要么在效果上等同于全部都不被执行，那么可以把这些活动组合成一个事务。

定义1 设 ρ 是一序列的操作，α、β 分别是任意的一序列操作，如果 $\alpha\cdot\rho\cdot\beta$ 与 $\alpha\cdot\beta$ 的执行效果相同，则称 ρ 是无影响的(effect-free)。

定义2 工作流事务 $T=\langle A,R\rangle$，其中，A 是一个活动的集合，R 是 A 上的二元关系。

定义3 如果 T_1 和 T_2 是事务，且 $T_2.A\subseteq T_1.A$，$T_2.F\subseteq T_1.F$，则称 T_1 包含 T_2，记为 $T_2\subseteq T_1$。

定义4 如果工作流事务 $T_2\subseteq T_1$，且不存在工作流事务 T_3 使 $T_2\subseteq T_3\subseteq T_1$ 成立，则 T_1 称为 T_2 的父事务，T_2 称为 T_1 的子事务。

定义5 如果对于活动 a，$\neg\exists\ T(a\in T.A)$，则称 a 位于工作流的第0层事务。如果事务 T 没有父事务，则 $\forall\ a\in T.A$ 称为位于工作流的第1层事务。如果 $\forall\ a_1\in T_1.A$ 位于工作流的第 i 层事务($i\geqslant 1$)，$T_1=\mathrm{Parent}(T_2)$，则称 $\forall\ a_2\in T_2.A$ 位于工作流的第 $i+1$ 层事务。

事务中活动可分为哑活动、可补偿的活动和不可补偿的活动。

定义6 对于不是哑活动的 $a\in TA$，如果存在另一个活动 a'，使得 $a\cdot a'$ 是无影响的，则称 a 是可补偿的活动，用 ca 表示，这种活动的集合用 S_{ca} 表示；否则，a 称为不可补偿的活动，其集合用 S_{uca} 表示。

2.3 事务模型的构建规则

事务模型建立在流程原模型的基础之上，并与流程原模型有机结合，以获得在故障发生时高效的部分回滚然后继续向前流动的能力。事务的补偿模型是流程原模型的逆操作，所以，其结构应该与原模型一致，只是对应的部分之间的执行次序正好与原模型相反。补偿模型应该与原模型紧密结合，要在2个模型之间通过合理的迁移来建立控制关系。事务模型要遵循改进的LOL确认机制。事务模型遵循以下8种规则来建立.

规则1 事务中的每个可补偿的活动都生成一个对应的补偿活动。原活动执行失败时马上转到其补偿活动上进行补偿。

规则2 “确认”活动利用传统的ACID事务来实现。

规则 3 补偿活动在补偿原活动之后，控制流回流，补偿先前已经执行的操作。

规则 4 事务中的每一个不可补偿的活动都应该有个关系为“或”的分裂分支，一个是原活动执行成功的执行路线，另一个则是原活动执行失败的执行路线。

规则 5 每个子事务都要定义一个事务故障处理活动，该活动有 2 个关系为“或”的分裂分支，一个流向本层的下一个活动（或事务），另一个则继续回流；第一层事务的故障处理活动则只有一个到下一个活动或事务的迁移。

规则 6 在事务执行中发生故障而回滚之后，最终转移到本事务的故障处理活动上来。

规则 7 在事务执行失败时，要记录故障所在的层次。在事务故障处理活动执行之后，如果本事务所处层数不比故障所在层数小，则要回流，否则应该继续向前流动。

规则 8 在回滚时，按照日志记录的事务中执行活动的逆序来执行补偿。

2.4 补偿模型例子

图 1 和图 2 是上节中所述规则的 2 个实例，(a) 是原事务模型，(b) 是扩充了补偿模型后的事务模型。从图 1 可知，在只有可补偿的活动的情况下，补偿模型与原模型很相似，只是补偿活动之间的依赖关系与原活动之间的依赖关系正好相反。另外，每一个原活动都有一个都对应补偿活动的迁移，使得当该活动发生故障的时候能够马上迁移到其补偿活动去进行补偿并向后回滚。图 2(a) 是多层事务的情况，每个事务和不可补偿的活动都有一个出错处理活动。

2.5 运行时的向后恢复

以图 1 为例，在图 1(a) 中，如果循环执行到第二次的时候，在 ca_2 处发生故障，则日志将有以下记录：ca_1-ca_2-ca_3-ca_1-ca_2，于是，马上转入 ca'_2 并按照 ca'_2-ca'_1-ca'_3-ca'_2-ca'_1 的次序来进行补偿。在图 2 (a) 中，假设流程执行到 ca_4 发生故障（日志中的记录是 uca-ca_1-ca_2-ca_4）则将转入 ca'_4 并按照 ca'_4-ca'_2-ca'_1-$fsta$ 的次序来进行补偿，并在 $fsta$ 执行之后转到 ca_5，因为发生故障的层次已经补偿完毕，所以可以继续向前流动。而如果流程执行到 ca_5 发生故障，日志中的记录是 uca-ca_1-ca_2-ca_4-ca_5，则将转入 ca'_5 并按照 ca'_5-ca'_4-ca'_2-ca'_1-$fsta$-$fuca$-fta 的次序来进行补偿。因为故障发生在第 $i+1$ 层事务，所以层次不小于 $i+1$ 的事务都要回滚，直到在 fta 执行之后，第 $i+1$ 层事务被补偿，然后从 fta 继续向前执行。

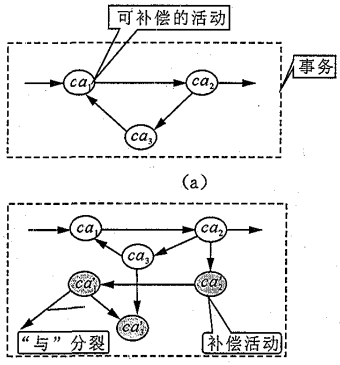

图 1 补偿模型的例子 1

(a)用户定义的事务片段——循环；(b)增加了补偿模型的事务片段——循环

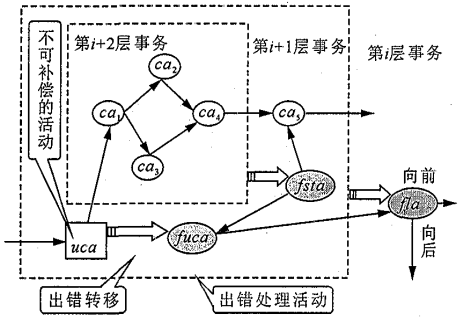

图 2 补偿模型的例子 2

(a)用户定义的事务片段——嵌套事务；(b)增加了补偿模型的事务片段——嵌套事务

3 结束语

本文通过使用补偿和确认，在创建态把补偿模

型与流程原模型整合在一起，并提供对多层事务的支持，以使用户可以灵活定制事务的粒度。在运行态利用日志提供向后恢复的路径的半动态方法而建立起来的事务模型，该模型具有以下优点：

(1)效率高。克服传统事务模型中数据依赖和冲突关系的复杂性，从而严重降低系统并发执行性能的缺陷。利用改进的LOL确认机制，实现事务数据的有效分离，不会出现读脏数据的情况，把事务期间对数据的读写操作次数降到最低，大大降低复杂度，提高了效率；

(2)易于管理和控制。把原模型与补偿模型整合在一起，故障发生点自然获得，相关数据易于传递；

(3)可精确回滚。利用日志提供向后恢复的路径，即使事务中包含如循环等复杂的结构也不会造成干扰；

(4)支持嵌套事务，提高并行性和减小事务颗粒度，提供更加灵活的控制，满足用户多样化的需求。

在向后恢复的路径方面，其实并不是一定要严格按照执行的次序的逆序来进行的，因为有些活动并没有依赖关系，如果能使这些没有依赖关系的补偿活动能够并行执行，则效率会更高，如何实现并行的向后恢复是需要进一步研究的问题。

参考文献：

1 Georgakopoulos D, Hornick M, Sheth A. An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases, 1995, 3: 119～153.

2 Sheth A, Rusinkiewicz M. On transactional workflows. In: Special Issue on Workflow and Extended Transaction Systems IEEE Computer Society, Washington D C, 1993.

3 Derks W, Dehnert J, Grefen P, et al. Customized atomicity specification for transactional workflows, cooperative database systems for advanced applications 2001. The Proceedings of the International Symposium, 2001, (s): 140～147.

4 Ding K, Jin B H, Wei J, et al. New model and scheduling protocol for transactional workflows, computer software and applications conference 2002. Proceedings 26th Annual International, 2002, (s): 920～927.

5 Mühlberger R, Orlowska M E, Kiepuszewski B. Backward Step: the right direction for production workflow systems. Distributed Systems Technology Centre Technical Report, 1998.

6 Kiepuszewski B, Mühlberger R, Orlowska M. FlowBack: providing backward recovery for workflow systems. Proceedings of 1998 ACM SIGMOD International Conference on Management of Data, 1998.

7 Liu C, Orlowska M, Lin X, et al. Improving backward recovery in workflow systems. In: 7th International Conference on Database Systyems for Advanced Applications, 2001.

8 Alonso G, Agrawal D, A El Abbadi, et al. Advanced transaction models in workflow contexts. In: 12th International Conference on Data Engineering, New Orleans, 1996.

(责任编辑：黎贞崇)

(上接第158页)

参考文献：

1 任承业. 校园信息系统中数据挖掘的研究与应用. 广州：暨南大学(硕士学位论文). 2003.

2 傅国强. 基于数据仓库的校园管理与决策支持系统的设计. 微机发展, 2003, 1: 82～84.

3 陶 兰, 王保迎, 吕建军. 数据挖掘技术在高等学校决策支持中的应用. 中国农业大学学报, 2003, 8(2): 39～41.

4 Jiawei Han, Micheline Kamber. 数据挖掘概念与技术. 范明、孟小峰, 等译. 北京：机械工业出版社, 2002. 223～259.

5 Knorr E M, Ng R Tucakov V. Distance-Based outliers: algorithms and applications. VLDB Journal Very Large Databases, 2000. 237～253.

6 Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. Proceedings of the ACM SIGMOD Conference, 2000. 473～438.

7 Bay S D, Schwabacher M. Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. Washington D C: SIGKDD'03, USA, 2003.

8 Breunig M M, Kriegel H P, Ng R T, et al. LOF: Identifying density-based local outliers. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, 2000. 93～104.

9 Aggarwal C C, Yu P S. Outlier detection for high dimensional data. In: Proceedings of the ACM SIGMOD International Conference on Management of data, 2001.

(责任编辑：黎贞崇)