Web服务事务中的补偿机制研究与实现*)

许 峰 徐碧云 黄 皓 谢 立

(软件新技术国家重点实验室 南京大学计算机科学与技术系 南京 210093)

摘 要 可靠的Web服务事务机制是面向服务的架构(SOA)中不可缺少的要素之一,它本身也需要有一定的恢复机制,其中非常重要的技术就是事务补偿。结合Web服务事务的特点,以及数据库系统中的触发器技术,可以为Web服务的事务处理模型提供一个半自动的事务补偿机制。

关键词 Web服务,事务,事务补偿,触发器

Research and Implement on Compensation in Web Service Transaction

XU Feng XU Bi-Yun HUANG Hao XIE Li

(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Reliable Web service transaction mechanism is one of the essential parts in the Service Oriented Architecture (SOA), and transaction it's self needs a mechanism for failure recovery. A pattern which combination with the feature of Web service transaction and the technique of trigger in database systems, is able to facilitate the application of Web service transaction process model.

Keywords Web service, Transaction, Transaction compensation, Trigger

1 概述

事务是进行可靠的应用程序处理最基本的概念之一。Web服务作为新一代的分布式计算方式,需要一个灵活的分布式事务处理模型来对请求和输出结果进行控制。现有的协议和模型一般把Web服务的事务分为两类或者三类(见第2节),当需要事务回滚或者子事务提交后撤销时,就使用一些额外的操作来还原或者部分地还原事务执行的效果。

2 Web服务事务处理和补偿

Web服务有其自身的特点,比如说资源的自治性高,潜在故障多,可能长期运行[9]等等,因此Web服务的事务需要有自己独特的协调和实现机制,现有的事务处理模型和协议包括BTP[6]、WS-Transaction和WS-Coordination[5]以及刚推出不久的WS-CAF系列协议[8]。虽然模型中的细节有所区别,但是基本上一致地把Web服务的事务分为原子事务(Atoms)和业务事务(Business Transaction)两类,而WS-CAF由于试图在一个框架中协调多个事务模型,故比其他事务处理模型多了一种事务,但本质上并没有什么不同。

原子事务具有ACID属性,可能采用嵌套的事务模型,参与事务的操作采用两阶段提交(2PC)来协调事务结果。业务事务则是原子事务的组合,有不确定的事务参与者集合,可能是长期运行的事务(Long-lived Transaction, LLT),采用开放嵌套(open-nested)事务模型或Sagas[3]模型,将事务划分为子事务的序列。每个子事务都包含一个可以被触发的补偿事务,在适当的时候被触发来撤销已经提交事务的影响。

3 补偿事务

定义 补偿事务是指用来撤销已提交的子事务所产生影响的事务[1]。

在当前的许多传统事务系统中,应用了补偿机制的实现作为正常操作的一部分。而作为取消事务效果的业务逻辑,也常常被作为完整的应用系统不可缺少的一部分来实现。

事务补偿不是一个新出现的概念,一般有两种传统的办法:(1)许多的事务系统中有独立的补偿模块来实现提交事务效果的取消,其实现大多通过补偿业务逻辑的执行来完成。(2)有的分布式数据库系统事务处理模型使用补偿事务的做法,利用触发器监控数据库表的操作,记录敏感操作和数据,用于子事务的恢复。

但是在Web服务的情形下,传统的作法显然不可取,因为让服务的使用者实现服务提供者端的补偿逻辑不太现实,而且分散的补偿逻辑不利于应用策略,所以本文使用了在每个Web服务端的数据库中实现补偿逻辑的方法。

3.1 触发器驱动的补偿

触发器驱动的补偿是基于主动数据库触发器概念的补偿事务产生和执行的方法,也就是以前所谓的事件-条件-行为规则(Event-Condition-Action rules, ECA rules)。

在Web服务领域应用这个方法,需要将Web服务构建于支持触发器的数据库系统之上。而现在所有主要的数据库系统基本上都遵从SQL:1999标准,支持触发器和传统事务。

支持触发器是主动式数据库系统一个非常出色的功能,主动式数据库系统监视着系统的运行,一旦符合条件的事件发生,触发器就被触发,一些与预定义的操作就会执行。触发器的形式一般如下:

```
on Event
if Condition
do Action
```

补偿事务通过三个步骤产生:首先,产生补偿操作的规则

*)本课题得到国家自然科学基金(60473091)和国家"八三六"高技术研究发计划项目基金(2003AA142010)资助。许 峰 博士研究生,主要研究方向为软件构件。徐碧云 大士,主要研究方向为软件构件。黄 皓 教授,博士生导师,主要研究领域为软件自动化。谢 立 教授,博士生导师,主要研究领域为分布式计算。

由 Web 服务开发者根据应用逻辑事先定义。其次，当子事务执行的时候，数据库系统根据规则产生补偿操作。最后，当子事务提交时，子事务的补偿操作结合进补偿事务中。

3.2 补偿操作的产生

所有改变数据库状态的子事务操作类型(更新，删除，插入等等)都需要定义补偿规则，补偿规则用数据库中的触发器定义。触发器由数据库模式设计器或者 Web 服务开发者使用 DBMS 的触发器语言定义。

产生补偿操作采用事件驱动机制，只有那些影响应用系统的数据或状态的操作才会产生相应的补偿操作。驱动事件包括：数据修改操作(如对数据库的 insert，update 以及 delete 操作)；事务协调消息(如 begin，commit 和 abort 等)；用户自定义的事件；系统事件(如系统或机器重启)。

根据补偿规则，补偿事务产生器在事件驱动下产生补偿操作。例如，当某子事务删除一个数据元素时，删除事件产生相应的补偿操作：

```
on Delete
do compensationOperation('Insert',data)
```

根据这个规则，子事务中的每一个 delete 操作将导致一次 insert 被删除的数据元素的补偿操作。函数 compensationOperation 可以被任何嵌入在商务逻辑中用于补偿的功能模块所替代。例如给某客户发一份电子邮件。补偿操作是在子事务的执行过程中动态产生的，假如子事务失败了，则在该子事务执行过程中所产生的所有补偿操作也被放弃。图 1 除了补偿操作就像其它常规的数据元素一样被存放在数据库中。

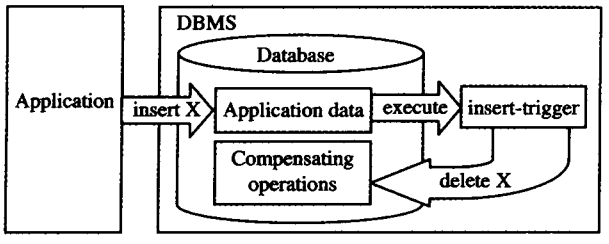

图 1 生成并存储补偿操作

3.3 补偿事务的产生与执行

前面描述了触发器驱动的补偿操作产生，现在对补偿事务的全局操作进行说明。触发器驱动的方法利用了触发器来定义若干的事件，来描述补偿事务。除了数据的改变和事务操作事件，触发器还定义了系统事件(比如：系统重起)，用于启动恢复(如图 2)。用户定义的事件可能被触发，并且得到一个当前执行的子事务的标识符。这个标识符可能之后被用于为这个制定的子事务初始化恢复活动。

在这个方法中，当前的 Web 服务中内建的 DBMS 事务管理功能不会受到任何影响，子事务以及补偿子事务将作为传统的事务而被执行，并且可以被 DBMS 作为一般的事务执行。而且，DBMS 事务管理能够自动地处理所有传统事务的恢复。因为子事务以及补偿事务都是传统 ACID 事务，所以都能够被 DBMS 通过传统的基于 Undo/Redo 的方法自动执行。

下列算法用于撤销一个提交的子事务 T，Abort[T]：

1)使用 DBMS 回滚设施撤销当前执行的子事务。

2)设置子事务 T 的执行状态为“撤销”。

3)取得所有存储的未执行的补偿操作形成补偿事务。执行下面 A 到 D 步来进行补偿事务：

A. 开始一个新事务，按照存储器中指定的顺序执行补偿操作。

B. 设置补偿事务的执行状态为“已执行”。

C. 提交补偿事务。

D. 如果事务在 A 到 C 步中失败，执行回滚并重复 A 到 C 操作 N 次。如果子事务仍然为失败，设置顶层的事务状态为 Failed(恢复失败)，并且通知系统管理员补偿事务失败，然后不执行补偿事务的剩余步骤，直接转到第 4 步。

4)回复已经成功完成撤销。

聚合事务通过 Cancel 消息或时间服务的超时触发来启动一个补偿事务。

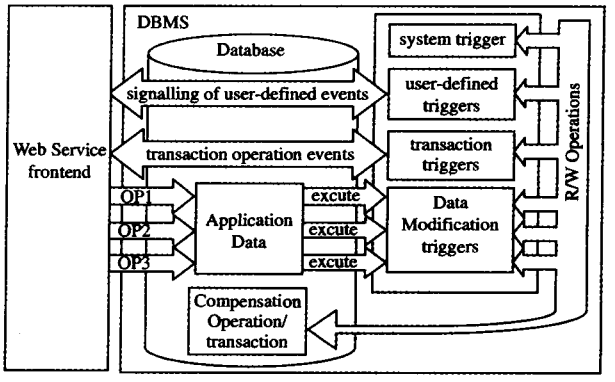

图 2 触发器驱动的事务补偿结构

4 事务补偿的实现

触发器驱动的事务补偿机制，可以通过 Oracle 9i RDBMS 来实现，Oracle 9i 提供了相对丰富的触发器机制，是 SQL:1999 比较完善的实现。Oracle 9i 也提供了很高级别的数据库存储过程扩展叫做 PL/SQL，完全可以用于应用触发器逻辑，当然要指出的是，任何支持触发器的数据库系统都完全能胜任这个工作，而并不是本文提出的模型需要对数据库系统有特殊的要求。

本节原型系统中事务补偿的核心是补偿服务，如图 3 中右下角所示，此处的补偿服务是通过 Oracle 9i 的 PL/SQL Package 来实现的，叫做 DBMS_COMPENSATE。

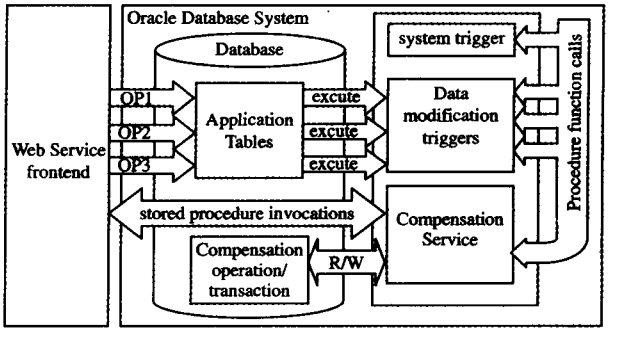

图 3 补偿服务的结构

为了触发一个特定模式的补偿服务(如图 3)，模式设计者必须调用一个叫作 DBMS_COMPENSATE. installService 的子程序，这将在当前的模式中插入系统重起事件触发器，这个触发器实现部分的恢复协议。同时，需要新建一些数据库表来保存补偿事务和补偿服务生成的内部结构信息。

模式设计者通过手工编写触发器来定义补偿事务，触发器用于监视相关库表的行和命令层面的数据管理事件，随时检查比对新旧数据库的状态，来抽取数据信息用于补偿，或者

在补偿无法进行时拒绝操作。补偿服务有两个接口类型，一个应用接口和一个模式设计接口。

4.1 应用和模式定义接口

Web服务可以调用应用接口，来实现补偿事务的开始，结束或者重新开始一个事务，并且可以设置保存点（save point），本接口用数据库系统的存储过程来实现，表1列出了接口提供的方法。

表1 应用接口列表

方法	描述
beginTransaction	开始一个新的会话事务
commitTransaction	提交事务
abortTransaction	撤销事务
continueTransaction	在某种失败之后继续执行事务
getTransactionStatus	获得事务的状态

应用程序可以通过使用 DBMS_COMPENSATE. abortTransaction 来开始恢复事务，如果事务已经有或者有多个子事务已经提交，那么需要执行补偿事务来取消已经提交结果。这时使用所提出的 Abort[T]算法。

提供模式设计接口是为了较少存储补偿事务的复杂性，模式设计者可以使用这里提供的方法来安装保存有补偿事务产生规则的触发器，同时安装与补偿事务结合在一起的补偿操作。方法列表见表2。

表2 模式定义接口说明

方法	描述
isTransactionValid	确认会话中是否有活动和有效的事务
addOperation	保存一个补偿操作，并把操作结合到相对应的补偿事务中
installTriggers	给特定的表格安装默认的触发器，以产生补偿操作

4.2 失效与异常的恢复

在基于触发器的补偿事务应用过程中，可能需要处理一些不同的出错情况。比如说多个补偿事务可能在执行前被检测到有死锁，补偿事务就会拒绝执行，这样的话需要在死锁发生时捕捉 Oracle 9i 系统抛出的异常，然后重新执行补偿操作。

如果补偿事务在重新执行 N 次之后还未能成功（N 可以通过补偿服务来设置，首先需要设定一个初值），则将被视为补偿失败，需要在日志中进行记录。同时，全局事务将设为 Failed 状态，并需要用某种方式（比如发送邮件）通知管理员处理恢复失效。下面是由 PL/SQL 实现的部分存储过程代码：

```
PROCEDURE recover_tran
    Failed_recovery EXCEPTION;
BEGIN
    n:=0;
    LOOP
      n=n+1;
      IF compensate_tran THEN EXIT;
      IF n=N+1 THEN
      RAISE Failed_recovery;
      EXIT;
      END IF;
    END LOOP;
END;

PROCEDURE compansating
BEGIN
    Recover_tran;
EXCEPTION WHEN Failed_recovery THEN
    BEGIN
      transaction_state = Failed;
      send_mail;
      EXIT;
    END;
    transaction_state = Cancelled;
END;
```

通过调用这两个存储过程，实现了补偿事务的执行过程，实质性的部分是 compensate_tran 中按照 XID 和 transaction. log 中的日志等恢复出来的补偿事务。如果子事务无法恢复，则标记事务失败，并调用 send_mail 存储过程来发送邮件给管理员。

日志记录的实现中一个非常重要的问题是如何得到执行的 SQL 语句 S 的反语句（inversed）S′，反语句 S′用来撤销语句 S 的执行效果。这里简单地以插入操作为例子讲一下这个过程：

```
insert into products(pno,name,price)
values(17,'chair',150)
```

这个 SQL 语句的反语句为：

```
delete from products
where pno=17 and name='chair' and price=150
```

如果“pno”是主键的话，“where pno=17”就可以了。接下来使用触发器来捕捉插入事件，假定日志要记录在 compansate. log 中，那么用 Oracle 9i 的 PL/SQL 语法，创建一个 insert-trigger 来生成删除操作：

```
CREATE TRIGGER compensate_opr_gen ON products
 AFTER INSERT ON products
BEGIN
 INSERT INTO comansate. log SELECT
  'DELETE products WHERE pno='+ || pno || +""+
  'AND name="'+ name + '"' +
  'AND price='+ || price ||
 FROM inserted
   END
```

日志中还需要记录其他一些信息来协助完成恢复过程，比如说事务的 XID，事务结束时的状态，甚至操作系统的状态等等。当然，有的时候仅仅生成反语句来生成补偿子事务是不行的，比如说月初结算上个月的销售量，如果在结算后上个月的某笔交易撤销了，那仅仅删除交易本身的记录是不对的。还要重新生成财务报表。这可以通过应用更复杂的存储过程来实现。

结束语 本文提出的基于触发器的 Web 服务事务补偿模式，能够比较好地处理事务撤销时的恢复问题，并能够以半自动的方式提供服务。而且，由于触发器的逻辑在数据库——也就是对 Web 服务而言的底层系统来实现事务的补偿，有利于集中利用补偿策略，具有一定的可扩展性。

参考文献

1 Strandenæs T, Karlsen R. Transaction Compensation in Web Services. In: Proc. NIK'2002, Norsk Informatikk Konferanse, Trondheim, November 2002

2 Wiekum G, Schek H-J. Concepts and applications of multilevel transactions and open nested transactions. In: A Elmagamid, ed. Database Transaction Models for Advanced Applications, 1992

3 Garcia-Molina H, Salem K. SAGAS. In: Stonebraker M, ed. Readings in database systems, San Francisco, California, 1987. 290～300

4 Limthanmaphon B, Zhang Yanchun. Web Service Composition Transaction Management. In: Conferences in Research and Practice in Information Technology - ADC. 2004, 27

5 Cabrera F, et al. Web Services Transaction (WS-Transaction). August, 2002. http:// www. ibm. com/developerworks/library/ws-transpec/. 2005. 3

6 Ceponkus A, et al. Business Transaction Protocol V1. 0. http:// www. oasis-open. org/committees/download. php/1184/2002206203. BTP-cttee-spec-1. 0. pdf. 2005. 3

7 Bunting D, Chapman M. Web Services Composite Application Framework (WS-CAF). http://www. oasis-open. org/committees/tc_home. php? wg_abbrev=ws-caf. 2005. 3

8 唐飞龙，李明禄. 一个 Web 服务事务处理模型：结构、算法和事务补偿. 电子学报，2003(12)：2074～2079