一种层次式的事务工作流失效恢复算法

任 怡,吴泉源,贾 焰
(国防科技大学计算机学院博士生队,湖南长沙 410073)

摘 要: 部分补偿和完全补偿是事务工作流失效恢复的传统补偿方法,对于长期运行且结构复杂的事务工作流,失效时需要补偿至静态定义的某一活动甚至起始活动,因此代价较大.给出了事务工作流的概念模型,提出了一种层次式的失效恢复算法.与传统方法相比,允许事务性或者非事务性子过程同时存在;将工作流看作由嵌套结构的子过程组成,而不是一个平坦流图;基于执行历史而不是工作流定义进行补偿,可简化循环结构的恢复.该算法针对活动的恢复策略进行层次式的向上规约处理,动态确定补偿终止点,可有效减小补偿域.证明了算法的正确性,性能分析表明其性能通常优于非层次的失效恢复算法,讨论了算法的实现.

关键词: 事务工作流; 失效恢复; 层次式; 补偿
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2005) 02-0317-05

A Hierarchical Failure Recovery Algorithm for Transactional Workflows

REN Yi, WU Quan-yuan, JIA Yan
(*School of Computer Science, National University of Defence Technology, Changsha, Hunan* 410073, *China*)

Abstract: Partial compensation and complete compensation are two common failure recovery methods for transactional workflows. For complex-structured long running transactional workflows, the compensation cost may be unacceptable since the workflow must be reversed to a previously defined activity or even to the start point to ensure semantic atomicity. We presented a conceptual model for transactional workflow. Based on the model, we proposed a hierarchical failure recovery algorithm. Different from traditional methods, it supports both transactional and non-transactional sub processes, and sub processes of a workflow are nestedly structured. The algorithm determines the end compensation point dynamically in a hierarchical bottom-up manner, so the compensation sphere is confined and compensation costs are reduced. We proved the correctness of the proposed algorithm. Performance analyses showed that it is more preferable than non-hierarchical methods. The algorithm has been applied into InforFlow, a workflow management system prototype. The implementation issues were illustrated.

Key words: transactional workflow; failure recovery; hierarchical; compensation

1 引言

工作流是一类可完全或部分自动完成的业务过程,某些应用领域要求一个工作流作为一个整体要么正确提交,要么回滚退出,即工作流具有事务特性,称为事务工作流[1].对于长期运行的具有特定语义的工作流,"非全则无"的原子性要求过于严格.因此需要放松原子性要求,通过灵活的失效恢复机制避免完全撤销已完成任务.

文献[2]提出了补偿的概念,用于维持长期运行的数据库应用的事务特性.目前事务工作流的放松原子性的保证主要基于该方法完成,这将涉及某些活动的撤消,开销较大,因此需要减少这类活动.已有的失效恢复方法缺乏算法一级的研究及其正确性验证;且常见的部分补偿和完全补偿机制[3]通常将具有复杂嵌套结构的工作流看作平坦流图,失效时事务工作流需要部分回滚至事先定义的一致点,甚至完全回滚到工作流起始点.

本文提出了一种面向复杂事务工作流的层次式失效恢复算法 HFR(Hierarchical Failure Recovery).由于工作流多个层次间具有基于应用语义的依赖关系,较低层次中出现的失效必要时需要传播到较高层次.HFR 为具有不同恢复特征的子过程及活动提供了灵活的处理机制,通过事先配置的子过程的恢复模式和活动的恢复策略,动态确定补偿终止点,将失效尽可能限制在低层的 block 中,可减少不必要的补偿,从而提高失效恢复的效率.该算法基于工作流实例的执行历史动态产生补偿图,可避免环形结构在回退处理中可能引起的问题.

收稿日期:2003-10-16;修回日期:2004-12-20
基金项目:国家自然科学基金项目(No. 90104020);国家 863 重点课题(No. 2001AA113020)

2 保证语义原子性的事务工作流模型

定义 1 一个工作流可定义为一个二元组 $W = (G, O)$. G 是一个有向图, $G = (A, L)$, 其中 $A = \{a_1, a_2, \cdots, a_{n1}\}$ 为结点集, $L = \{l_1, l_2, \cdots, l_{n2}\}$ 为连接弧集, $l_i = \langle a_j, a_k \rangle$ 为从 a_j 到 a_k 的连接弧, $a_j, a_k \in A$; $O = \{o_1, o_2, \cdots, o_{n3}\}$, 表示 A 中结点访问的抽象数据对象组成的集合.

定义 2 工作流的一个子过程记为 $P = (G', O')$. $G' = (A', L')$ 是 G 的一个有向子图, $A' \subseteq A$, $L' \subseteq L$, O' 为 A' 访问的抽象数据对象的集合, $O' O$. 子过程也是工作流.

定义 3 当 G' 为平凡图时, P 所定义的子过程只含一个结点, 将这种子过程称作工作流的活动.

定义 4 工作流 w 满足语义原子性, 若 w 的每次执行满足条件之一: (1) 成功完成; (2) 产生失效而不能成功完成时, 其执行结果可被补偿.

子过程可看作特殊的工作流, 基于定义 4, 给出:

定义 5 若工作流 w 的子过程 p 需要满足语义原子性, 则称 p 是 w 的一个块(block). 将块 b 的起始点表示为 S_b.

定义 6 工作流的执行历史是指一个工作流实例从起始点到结束点所经历的轨迹, 记为 $H = (A', \prec)$, A 为工作流定义结点集, $\prec$ 是 A 生的一个偏序关系.

以活动为结点, 根据时序建立结点间的有向边, 可产生一个有向图 $G_H = (A', E)$, 称为工作流执行历史图, $A' \subseteq A$, E 为 G_H 中有向边的集合.

3 失效处理范围和循环处理问题

图 1 是旅行预约服务的工作流定义. a_8 是同步点, 仅当其所有前驱结点执行完毕方可执行; a_2、a_3、a_4 和 a_8 形成 block, a_6 和 a_7 是一个循环. 假设 a_3 失败, a_5 已完成, a_6 和 a_7 正执行. 若采用完全补偿, 则要回退到 a_1; 即使采用部分补偿, 也要回退至 a_2 (通常会定义 a_2 为一致点), 于是并行执行的 $a_{5\sim7}$ 也在补偿范围内, 造成不必要的回退和重新执行. 本文提出的 HFR 算法将动态确定补偿域, 可有效减小补偿范围.

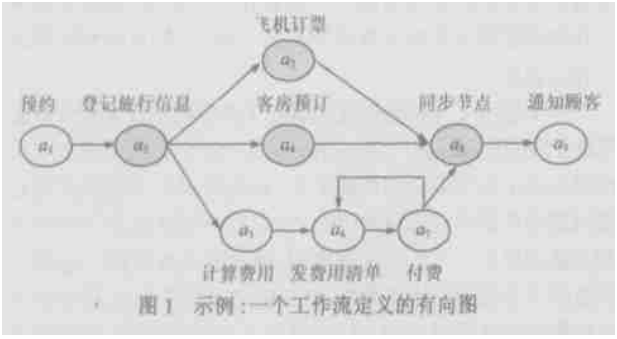

图 1 示例: 一个工作流定义的有向图

循环是工作流管理系统实现中的另一个难点, 复杂循环的补偿存在一定难度, 因为每次执行循环体时, 系统处于不同的状态, 仅根据工作流定义的补偿活动进行补偿是不合适的. 补偿需要借助执行历史和日志信息, 根据具体情况进行. 本文方法基于工作流的执行历史而不是工作流定义. 执行历史是对定义的解释执行, 其结构相对简单, 不存在环结构和或分支/或汇合结构, 从而简化了循环的补偿问题.

下面将基于第 2 节的模型提出一个有效的失效恢复算法. 在该算法下, 图 1 的例子只需补偿 a_3, 且依据执行历史图进行动态的补偿.

4 HFR 算法

4.1 恢复模式和恢复策略

事务工作流的子过程可能作为一个原子整体, 也可能是非事务性的实体; 工作流活动由外部应用程序和资源完成, 执行环境、业务逻辑等的差异使其具有不同的恢复特点. 因此, 本节首先基于文[4]给出描述活动恢复特性的相关定义, 然后提出恢复模式 (RM) 和恢复策略 (RP), 用于声明子过程和活动的恢复属性.

定义 7 设 $\delta = \langle a_{i1}, a_{i2}, \cdots, a_{ik} \rangle$ 是一个活动执行序列, $a_{ij} \in A$, $j = 1, 2, \cdots, k$. 若对于任意活动执行序列 μ 和 ν, $\langle \mu\delta\nu \rangle$ 与 $\langle \mu\nu \rangle$ 的执行效果相同, 则称 δ 是无影响的. 若活动 $a \in A$ 无影响, 则称 a 可忽略 (non-vital), 否则称其为不可忽略的.

定义 8 对于某个工作流的活动 a_i, 若存在 a_j, 使得 $\delta = \langle a_i, a_j \rangle$ 无影响, 则称 a_j 是 a_i 的补偿活动, 记作 a_i^{-1}, 并称 a_i 可补偿 (compensable), 否则称 a_i 不可补偿.

定义 9 若活动 $a \in A$ 不可补偿, 且不可忽略, 则称 a 为关键的 (critical).

定义 10 对于 $\forall a \in A$, 若 $\exists m \in N$, 即使对所有 $1 \leq k < m$, $a^{(k)}$ 均被放弃, $a^{(m)}$ 仍能确保提交, 则称 a 是可重试的 (retriable). $a^{(k)}$ 表示 a 的第 k 次执行.

定义 11 对于 $\forall a \in A$, 若存在活动 $a' \neq a$, 且 a 与 a' 按应用语义具有相同的功能, 则称 a 是可替换的 (replaceable), a' 是 a 的替换活动.

定义 12 对于块 b, b 的结点数大于 1, 若存在 $b' \neq b$ 且 $S_b = S_{b'}$, b' 与 b 按应用语义具有相同的功能, 则称 b' 具有可选路径, b' 是 b 的可选路径.

若因为包含关键结点且无可选路径, 造成子过程 p 不能自动恢复, 则称 p 不可恢复, 否则是可恢复的.

工作流的子过程和活动往往具有不同的恢复属性, 这与具体业务逻辑相关, 因此工作流失效恢复不可避免地需要业务层的支持. 为此扩展子过程和活动已有属性, 提出 RM 和 RP 的概念.

定义 13 RM 是表示子过程 p 原子性的属性. $RM(p) = \{\text{atomic}, \text{non-atomic}\}$, atomic 表示该子过程最终要满足语义原子性; non-atomic 指该子过程的执行无语义原子性要求.

$RM(p) =$ atomic 的子过程 p 是 block. 活动的 RM 值缺省为 atomic.

定义 14 RP 是定义活动 a 恢复特点的属性. $RP(a) = \{\text{non-vital}, \text{compensable}, \text{critical}\}$, 其中 compensable 包括子策略 retriable 和 replaceable.

RP 为 compensable 的活动, 需要定义子策略. 图 1 中 a_3、a_4、a_6 和 a_7 的 RP 为 compensable, a_3 和 a_4 的子 RP 为 replaceable, a_6 和 a_7 的子 RP 为 retriable. 为提高系统自动处理失效的能力, 需要在工作流定义中预先配置 RM 和 RP, 并声明补

偿/替换活动,有可选路径的子过程应提供其可选路径.事务工作流应用开发者通过预先定义这些属性间接参与失效恢复.

4.2　层次式的失效恢复算法

HFR 算法基于以下约束条件:

①补偿活动必须正确提交;可选路径必须正确完成.

②block 的子过程不能是 non-atomic 的.

③若产生失效的 block 中存在关键活动,那么该关键活动不能是失效结点与补偿终止点之间已完成的活动.

约束 ①可避免失效的无终止处理;约束 ②界定最高层块为一个原子域;约束 ③避免补偿关键活动情况的发生. HFR 针对工作流结构的层次特点,由系统、工作流、块和活动四个不同粒度、相互协调的子算法组成.不妨称子过程的直接上层子过程为其父过程;若块 b 的父过程 p 是 atomic 的,则 p 是 b 的父块.

4.2.1　异常的分发 ExceptionDispensation　在出现失效时,该算法作为顶层算法被激活.对于已知类型的失效,激活事先注册的用于解决该类型异常事件的异常处理器;对于不能处理或者不存在相应类型异常处理器的情况,调用工作流失效恢复算法 FlowRecovery.

4.2.2　工作流的失效恢复 FlowRecovery　FlowRecovery 处理异常的同时保证语义原子性.它首先中止正在执行的活动,通过调用 FAPreHandling 确定失效活动是否是可恢复的.若可恢复,则根据 FAPreHandling 预处理后的定义重新初始化该活动,否则通过 BCGGenerating 计算父块(如果存在)的补偿图.若父块可恢复,工作流回滚至父块的起始点并执行可选路径,否则继续计算更高层块的补偿图,直至工作流可恢复或者到达顶层.若顶层块不可恢复,通知系统管理员进行人工干预.

算法　工作流的失效恢复算法 FlowRecovery

算法的输入:失效活动 ID

算法的基本步骤:

step1　NonRecoverable := FALSE;

step2　放弃正在执行的活动;

step3　初始化当前补偿图为空;

step4　调用 FAPreHandling,检查 NonRecoverable 标记值;

step5　若为 FALSE,则按预处理后的定义重新执行,退出 FlowRecovery,否则设置当前失效活动为当前失效块;

step6　若当前失效块存在有父过程 p,且 p 是 atomic 的,

step6.1　则以当前补偿图为输入,调用 BCGGenerating,获得 p 的补偿图.若 p 有可选路径,

step6.1.1　则 NonRecoverable := FALSE,根据补偿图回滚至,执行可选路径,返回.

step6.1.2　否则将 p 设为当前失效块,goto 6.

step6.2　否则回滚至块起始点,通知系统管理员,返回.

FlowRecovery 通过补偿支持向后恢复,通过可选路径和重试支持向前恢复.存在可选路径的块的起始点作为逆向补偿终止点(向前恢复起始点).该算法直到可恢复的块或者顶层块才结束.一旦块可恢复,就不再处理更高层的块.因此,补偿终止点是在自底向上的过程中动态确定的,补偿域被限制在尽可能的较低层中.

4.2.3　活动失效预处理 FAPreHandling　该算法进行失效恢复的预处理.若失效活动是可忽略的,则重新定义该活动为空操作;若为可补偿的,则执行补偿活动(可替换的失效活动的定义被其替换活动所替代);若为关键的,则设置 NonRecoverable 标记为 TRUE,并返回该值.

4.2.4　块补偿图的生成 BCGGenerating

补偿图用于向后恢复.该算法的目标是根据工作流执行历史计算当前失效块(见 4.2.2)的父块的补偿图.父块补偿图的计算是以当前块的补偿图为输入的.遍历父块的所有结点及其相关结点,构造对应执行历史图的补偿图.按照补偿图执行,可撤销已完成活动的结果.

4.3　算法的正确性

由约束 ②,工作流 w 的任何 block 必然含有 n 层 block 结构($n \in N$).不妨假设由下至上分别称为集合 B_{H0}、B_{H1}、…、B_{Hn-1}.其中,B_{Hi}的元素是若干块和活动,B_{Hi+1}的元素是包括 B_{Hi}元素组成的块的若干块和活动,则由 B_{Hn-1}定义的 w 的一个原子域中的任意活动在并仅在某一个 B_{Hi}中,$i = 0, 1, \cdots, n-1$.

定理 1　HFR 能保证工作流实例在某个失效产生情况下的语义原子性.

证明　块的失效最终会归结到结点失效,因此,下面只分析活动的失效处理的正确性.

对于任意引起系统失效的活动 $a \in B_{Hi}$,有:若 a 是 non-vital 的,则 a 不会对 B_{Hi}及工作流 w 产生影响;若 a 是 compensable 的,则工作流显然可恢复;若 a 是 critical 的,由 FlowRecovery 知,存在以下可能:

(1)若 a 存在可恢复的父块,则由父块补偿图将系统回滚至父块起始点,并利用可选路径向前恢复;

(2)若 a 存在父块,但父块不可恢复,则向上直至可恢复的父块 $b \in B_{Hj}$, $i < j \leq n$,处理同(1);或直到 B_{Hn-1}工作流仍不可恢复,则生成 B_{Hn-1}的补偿图,系统回滚至原子域执行前的状态,并请求人工处理.

因此在满足约束的条件下,HFR 要么使工作流回滚至原子域执行前的状态,要么在原子域的某个子域内得到恢复.得证.

定理 2　HFR 保证的是放松的原子性.

证明　由定理 1 的证明知,若 B_{Hn-1}定义的原子域中出现失效且 $B_{Hi}(0 \leq i < n)$可恢复时,工作流并不在整个原子域中回滚,而是在 B_{Hi}内部通过补偿、重试、替换、执行可选路径实现恢复.因此,HFR 保证的不是"非全则无"的原子性,而是放松的原子性.

定理 3　对于 non-atomic 的子过程,HFR 也可保证其语义正确性.

证明　$\forall p(RM(p) = \text{non-atomic})$,无需保证 p 执行的原子性,只要能处理失效即可.由约束 ②,p 不包含在任何块中.p 的失效都源于它的某个活动 a 的失效:若 a 属于某个 p 的子块,由定理 1 知,可得到正确处理;否则是原子域为一个活

动的特殊情况,无需保证原子性,只需恢复该活动即可,易证 p 可被正确处理.

4.4 算法的性能分析

HFR 算法的一个优点在于限制了失效补偿的范围,从而减少不必要的补偿;但同时由于层次式处理方式增加了相应的开销.下面将 HFR 与一般非层次的补偿方法的回滚代价进行比较.

(1) 整个原子域不可恢复时,两类方法需补偿活动数相同.由于复杂工作流活动的执行时间较长,补偿活动作为其逆操作,运行时间也相应较长.与补偿活动的执行时间相比,由层次式处理方式增加的开销可忽略.

(2) 在原子域可恢复的情况下,HFR 能够有效减少补偿活动的个数,而补偿活动运行时间相应较长,一般情况下,HFR 的性能较优.

由于工作流应用的特点,失效通常可在原子域内得到恢复,HFR 算法性能一般较非层次的方法更高.

4.5 算法的实现

如图 2,HFR 算法已用于 InforFlow 工作流原型系统中,基于 HFR 的事务性失效恢复管理器是其重要组成部分.工作流引擎将异常事件转发给异常管理器,后者负责异常的处理与分发,不能处理的异常类型,将移交事务性失效恢复管理器.工作流引擎和失效恢复管理器通过定义的公共访问接口维护并读写执行历史信息.由工作流引擎向工作流管理和监控部件发送处理信息,必要时接收人工干预.实现中考虑的问题包括:

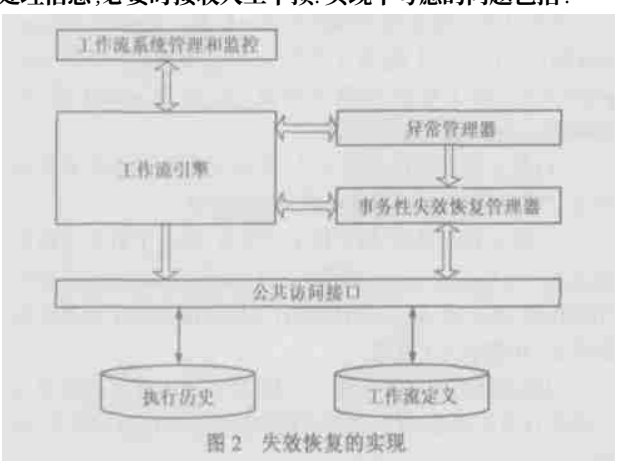

图 2 失效恢复的实现

(1) 执行历史图的维护.静态工作流定义和动态工作流实例相分离,两者都保存在持久存储器中.执行历史图的构造、维护和存储与动态工作流实例紧密相关.

(2) 失效检测和处理.异常处理器是系统定义的特殊活动,只有当活动出现语义失效时,该活动被激活.异常处理器被处理不同异常事件的子失效处理器所继承.它处理注册的各种异常,并将不能处理的异常转发给事务性失效恢复管理器.

(3) 工作流定义语言扩展.扩展已有的工作流定义语言以支持复杂过程的事务特性.WfSubProcess 和 WfActivityNode 被扩展,以包含 *RM* 和 *RP* 属性.在工作流定义实例化后,这些恢复特征信息可被使用.

5 相关工作

事务工作流的失效恢复的研究源于高级事务模型(ATM: Advanced Transaction Model),如嵌套事务[5]、开放式嵌套事务[5]、Sagas[2]、Flexible 事务模型[6]等.尽管 ATM 的研究已经取得了一定成果,但并不完全适合事务工作流,原因在于 ATM 通常面向数据集中的应用,事务结构模式固定,事务工作流的失效恢复要比 ATM 复杂.

ConTracts、FlowMark、WAMO、OPERA、WIDE 等工作流系统借鉴 ATM 思想,采用基于平坦流图的事务工作流失效恢复机制,即完全补偿或部分补偿的静态确定补偿范围的方法[3,7~10].与 HFR 相比,完全补偿的方法从失效结点开始,逆向回滚至工作流起始点,所以补偿域是整个工作流,也包括块的最上层;部分补偿的方法事先确定补偿终止点,失效时逆向回溯至该点,若该点落在发生失效的 block 中,仅回滚到该点是不能保证语义原子性的,若落在发生失效的 block 之外,则会回滚整个 block,于是将回滚至子过程的顶层.因此,HFR 所确定的补偿域通常小于基于平坦流图的失效恢复机制确定的补偿范围,可有效减少补偿活动的个数.

以上工作为失效恢复问题提供了不同角度的研究,但都未考虑如何减少补偿代价.HP 实验室的研究[11]提供了一种声明补偿域的灵活机制,试图减少补偿的数量,但并未给出如何通过补偿保证语义原子性;且其终止补偿点是固定的,可能会引起不必要的补偿或不正确的补偿.Q Chen 的研究[12]中,事务组织为层次结构,但并不区分事务性或者非事务性任务,这并不符合实际工作流的特点.

已有研究缺乏算法一级的描述,因此失效恢复的自动化支持程度相对较低.本文采用层次方法,给出了事务工作流失效恢复的算法一级的解决方案.

6 结束语

HFR 面向失效恢复代价问题,支持区分事务性和非事务性子过程,基于执行历史计算补偿图动态确定终止补偿点,将失效补偿范围限制到尽可能低层次的 block 内部,能够有效减小补偿域.该算法具有通用性,适用于集中或者分布的工作流管理系统,还可用于具有事务特性的其它面向过程的自动化系统.

参考文献:

[1] A Sheth, M Rusinkiewicz. On transactional workflows[J]. IEEE Data Engineering Bulletin, Special Issue on Workflow Systems, 1993, 16(2):34 - 40.

[2] H Garcia-Molina, K Salem Sagas. In proceedings of the ACM SIGMOD Conference on Management of Data[C]. San Francisco, California, USA, May 1987. 249 - 259.

[3] P Grefen, et al. Global transaction support for workflow management systems: from formal specification to practical implementation[J]. The VLDB Journal, 2001, 10(4):316 - 333.

[4] H Schuldt, et al. Atomicity and isolation for transactional processes[J].

ACM Transactions on Database Systems ,2002 ,27(1) :63 - 116.

[5] A Elmagarmid. Database Transaction Models for Advanced Applications [M]. San Francisco :Morgan Kaufmann Publishers ,1992.

[6] A Zhang ,et al. Ensuring relaxed atomicity for flexible transactions in multidatabase systems[A]. In proceedings of the ACM SIGMOD Conference[C]. Minneapolis ,Minnesota ,USA ,May 1994. 67 - 78.

[7] F Schwenkreis. A formal approach to synchronize long-lived computations[A]. In proceedings of the 5th Australasian Conference on Information Systems[C]. Melbourne ,Australia ,September 1994. 273 - 284.

[8] F Leymann. Supporting business transactions via partial backward recovery in workflow management systems[A]. In Proceedings of BTW' 95[C]. Dresden ,Germany ,March 1995. 51 - 70.

[9] J Eder ,W Liebhart. The workflow activity model WAMO[A]. In proceedings of the Third International Conference on Cooperative Information Systems[C]. Vienna ,Austria ,May 1995. 87 - 98.

[10] C Hagen ,G Alonso. Flexible exception handling in the OPERA process support system[A]. In Proceedings of the 18th International Conference on Distributed Computing Systems[C]. Amsterdam ,The Netherlands ,May 1998. 526 - 533.

[11] W Du ,et al. Flexible Compensation for Workflow Processes[R]. Technical Report HPL-96-72(R. 1) ,HP Labs. http :/ / www. hpl. hp. com/ techreports/ 96/ ,1997.

[12] Q Chen ,U Dayal. Failure handling for transaction hierarches[A]. In proceedings of 13th International Conference on Data Engineering[C]. Birmingham ,UK ,April 1997. 245 - 254.

作者简介:

任　怡　女,1977 年 4 月生于陕西宝鸡,国防科技大学计算机学院在读博士生,主要研究方向为数据库和分布计算技术. E-mail :renxiaoyi @ 21cn. com.

吴泉源　男,1942 年 2 月生于上海,教授,博士生导师,主要研究方向为人工智能与专家系统、网络计算中间件和分布计算软件环境.

贾　焰　女,1960 年 4 月生于四川成都,教授,博士生导师,主要研究方向为数据库、分布计算和人工智能.