

[19] 中华人民共和国国家知识产权局

[51] Int. Cl⁷

G06F 9/46

G06F 11/14



[12] 发 明 专 利 说 明 书

[21] ZL 专利号 99812302.1

[45] 授权公告日 2003 年 11 月 19 日

[11] 授权公告号 CN 1128405C

[22] 申请日 1999. 10. 13 [21] 申请号 99812302. 1

[30] 优先权

[32] 1998. 10. 19 [33] GB [31] 9822807. 5

[86] 国际申请 PCT/GB99/03400 1999. 10. 13

[87] 国际公布 WO00/23890 英 2000. 4. 27

[85] 进入国家阶段日期 2001. 4. 18

[71] 专利权人 国际商业机器公司

地址 美国纽约

[72] 发明人 伊恩·J·米切尔 史蒂文·鲍威尔

审查员 袁丽颖

[74] 专利代理机构 中国国际贸易促进委员会专利

商标事务所

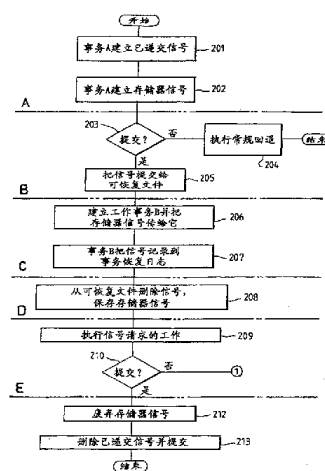
代理人 王以平

权利要求书 2 页 说明书 9 页 附图 3 页

[54] 发明名称 事务处理系统及方法

[57] 摘要

一种用于处理一些由多个成员事务组成的应用的事务处理方法、系统和计算机程序产品。一成员事务的事务失败，通过可用于一个或多个其它事务的一个失败指示符，造成一个可靠的失败许可。在一可恢复存储器中存储该失败指示符，使它的作用可以在系统失败和恢复下存在。



1. 一种用于处理具有多个成员事务的应用的事务处理系统，包括：
用于检测失败成员事务的失败的检测装置；
用于响应所述检测装置而回退所述失败成员事务的回退装置；
用于响应所述检测装置而指示所述失败成员事务的失败的失败指示符装置；以及

用于存储失败指示符的可恢复存储装置，其中可使所述失败指示符可靠地用于所述多个成员事务中的其它一个或多个事务。

2. 如权利要求1所述的事务处理系统，其中能可靠地用于所述多个成员事务中的所述一个或多个事务的失败指示符能够触发所述其它的一个或多个事务。

3. 如权利要求1所述的事务处理系统，还包括：
用于建立已递交信号的建立装置，
用于从所述已递交信号建立存储器中的存储器信号的建立装置；
其中所述失败指示符装置通过修改所述存储器信号中的一个或多个信号生成所述失败指示符。

4. 如权利要求3的事务处理系统，还包括：
用于记录修改的存储器信号的记录装置；
用于在所述记录装置的操作期间暂停所述回退装置的操作的暂停装置；以及
用于在所述记录装置的操作之后恢复所述回退装置的操作的恢复装置。

5. 如前面任一权利要求所述的事务处理系统，还包括：

一个可处理一个或多个失败指示符的同步点管理器。

6. 一种用于处理具有多个成员事务的应用的方法，包括以下步骤：

检测第一事务的失败；

响应所述检测失败的步骤而回退所述第一事务；

可恢复地存储一失败指示符，从而在系统失败并恢复后它是可恢复

的；以及

可靠地使所述失败指示符可用于其它事务，从而所述其它事务可依赖所述失败指示符。

7. 如权利要求6所述的方法，其中可由所述能可靠地用于其它事务的失败指示符能触发所述其它事务。

8. 如权利要求6或7所述的方法，还包括步骤：

建立一个或多个已递交信号；以及

从所述一个或多个已递交信号建立存储器中的一个或多个存储器信号；

其中通过修改所述存储器信号中的一个或多个信号而生成所述失败指示符。

9. 如权利要求8所述的方法，还包括步骤：

记录修改的存储器信号；

在所述记录步骤期间暂停所述回退步骤；以及

在所述记录步骤后恢复所述回退步骤。

事务处理系统及方法

技术领域

本发明涉及用于数据处理系统的事务处理领域，尤其涉及由多个成员事务组成的应用的处理。

背景技术

在事务处理领域中，事务通常需要具有所谓的 ACID 特性。ACID 是代表四个事务特性的首字母缩写，即，原子性、一致性、隔离性和持久性。原子性要求对事务的所有操作都发生或者所有操作都不发生。一致性要求事务保持数据库的一致性。隔离性要求事务不读取另一事务的中间结果。持久性要求承诺事务的结果是持久的。有关 ACID 特性的背景信息，参见 Transaction Processing: Concepts and Techniques, J. Gray and A. Reuter，其收录为本文的参考资料。

利用联机事务处理(OLTP)系统实现的大企业应用典型地使用一些独立的 ACID 事务来实现企业认为是复杂的但却协调地更新商业数据的任务。由于 OLTP 环境中的数据和资源共享要求，不可能简单地用单个 ACID 事务实现企业事务。

随着应用发展成越来越复杂，在事务处理系统的事务的概念和企业分析家的事务的概念之间的失配变得更大。这在协调事务的收集以使它们能按企业分析家的要求来运行方面，给应用开发人员带来更多的复杂性。

事务之间的协调主要是利用事务处理系统的应用编程接口(API)通过应用设计人员的创造力予以处理。应用需要保持一些状态数据以记录事务收集中的进展，并且每个独立的事务需要一种激发下一个事务从而实现前进的手段。这种激发是以信令为形式的。

可以用几种方式管理应用前进状态数据。例如，可以通过在为应用

选择的数据库中存储它而使它是可恢复的。把该应用的“工作存储”和寿命更长的企业数据(例如顾客记录)隔离开是一个好习惯。

在主要由最终用户行为驱动应用中的前进的情况下, 可把一系列的事务称为“伪会话”。把来自最初用户终端的输入作为用于下个要启动的事务的信号。常规地, 把一些状态数据传给新启动的事务。在众所周知的 IBM 的 CICS 事务处理系统(CICS 是国际商业机器公司的一个注册商标)中, 状态数据只是一段和该用户终端相关的虚拟存储(“伪会话 COMMAREA”)。

由于由多个成分事务组成的企业应用要求即使一个成员事务失败仍要继续前进, 故可用于应用的、使信号有计划地通过一系列的事务的工具, 可能是非常有限的, 并且每个事务的 ACID 特性是抑制性的; 这是因为隔离特性意味着失败事务的所有影响必须被颠倒, 或者“回退”。

典型地, 事务处理系统的 API 包括一个使得执行另一个事务的命令, 并且应用程序员试图克服上述限制的一种方法是使用该 START 命令。在 CICS API 中存在着 START 命令。START 命令允许在一个事务中执行的一个应用程序建立另一个事务并且开始执行它。该命令建立一个运行一个事务的请求; 该请求称为 ICE (区间控制元素)。数据可以从起源事务传送到新事务。START 命令允许 PROTECT(保护)参数, 该参数用于规定该命令的作用的恢复特征。

受保护的 START 具有 ACID 特性: 仅在发出它的事务成功地向前提交期间才提交它的作用; 传送到新事务的 ICE 和数据被存储在可恢复的资源中。提交的并受保护的 START 请求保证是会发生的而且只发生一次。而不受保护的 START 不遵守 ACID 事务语义。新事务被建立并且立即开始执行, 有可能和起源事务并行地执行。ICE 和数据不存储在可恢复的资源中。不保证: 不受保护的 START 命令会发生并且只发生一次。

利用 START 命令刺激应用的前进是有限制的。应用程序人员试图回避这些限制, 但这是特别困难和不可靠的。使用不受保护的 START 命令通常不能达到和事务应用相关的保证水平。例如, 可利用不受保护的 START 激发应用中的下个步骤, 但起源事务会异常终止并回退它的可恢

复的更新。接着由该不受保护 START(启动)引起的事务必须在考虑到前面某步骤失败下确定应用的状态。系统的重新启动会丢失未决的不受保护 START 请求,并且从而丢失该应用中的线程,使用受保护 START 命令确保前进是和各事务的成功同步的并且举止合理地越过系统故障。但正是给出这些特征的 ACID 特性导致出应用程序人员必须克服的一个不同问题。若起源事务执行回退(例如在应用失败的情况下),则受保护的 START 不产生任何结果。从而异常终止的事务不可能主动促使应用继续。

应用程序人员可试图通过建立一个作为“后退停止”的延迟 START 请求以检测出某具体步骤的失败来应付这种情况。在事务提交的情况下,该后退停止请求是多余的并被废弃,而在回退情况下该时间段过去后执行该后退停止事务一意味着先前出现过回退。该技术的局限在于它增加应用前进管理的复杂性,它在系统出故障下不是完全可靠的,由于后退停止事务需要对付各个可能要执行它的时刻并且要对付主事务成功或者不成功,其编程特别困难,以及难以选择后退停止事务的延迟时间并且这还限制了应用进展速率。

过去,已提出各种事务处理模型,包括各种带有嵌套子事务的长期运行、扩展的事务模型。一种试图处理多步骤事务应用的模型是 ConTract 模型,该模型由 Waechter 和 Reuter 在 A. K. Elmagarmid 编辑的

“Database Transaction Model for Advanced Application”一书中的“The ConTract Model”文章里说明,该文作为本文的参考资料。ConTract 模型提供一个用于商业事务的理论框架,其涉及使用一组流控制以及在 ConTract 管理实体的控制下的诸步骤之间的依从关系。该文明确提出:

“为了在处理各 Contract 期间构建系统的工作,需要专用类型的全局嵌套事务”。这种类型的层次结构具有把一个成本高的额外控制层引入到系统中的缺点。每个这样的层负面地影响系统的性能并增加系统的复杂性。在该模型中定义二个编程层:第一个层是各个步骤的编程,第二个层是控制“脚本”的编程以管理事务流和依从性。引入这种附加的编程层同样不利地增加系统的复杂性。

发明内容

于是，本发明在第一方面中，提供一种用于处理具有多个成员事务的应用的事务处理系统，其包括：检测装置，用于检测失败成员事务发生的失败；回退装置，用于响应所述检测装置以回退所述失败的成员事务；失败指示装置，用于响应所述检测装置以指示所述失败的成员事务的失败；以及可恢复存储装置，用于存储失败指示符，从而可靠地使所述失败指示符可用于所述多个成员事务中又一个或多个事务。

换言之，用于处理由多个成员事务组成的事务的事务处理系统可恢复地存储成员事务的失败的指示，并和该失败事务的回退处理无关地使该指示可用于其它成员事务。

从而，本发明使后面的事务能传送来自前面的失败的可靠的事务的失败的许可，这种对事务处理的传统范围的扩展在商业应用的处理中是有用的，在商业应用中要求即使前面的活动失败仍应继续处理某些活动。采用可恢复的存储装置使本事务处理系统能有益地在系统失败和恢复后保存失败指示符。

本发明的第一方面的优选特征是具有一个如上所描述的事务处理系统，其中能可靠地用于所述多个成员事务中的所述一个或多个事务的所述失败指示符能触发所述又一个或多个事务。

本发明的第一方面的另一优选特征是具有一个如上所描述的事务处理系统，其还包括：用于建立已递交信号的建立装置；用于从所述已递交信号中建立存储器的存储器信号的建立装置；其中所述失败指示符装置通过修改一个或多个所述存储器信号而生成所述失败指示符。该事务处理系统最好还包括：用于记录修改的存储器信号的记录装置；用于在所述记录装置的操作期间暂停所述回退装置的操作的暂停装置；以用于在所述记录装置的操作后恢复所述回退装置的操作的恢复装置。

本发明的第一方面的另一个优选特征是具有一个如上所描述的事务处理系统，其还包括：一个能处理一个或多个失败指示符的同步点管理器。

本发明在第二方面，提供一种用于处理具有多个成员事务的应用的

方法，该方法包括步骤：检测第一事务的失败；响应所述检测失败的步骤回退所述第一事务；可恢复地存储一个失败指示符，使得它在系统失败和恢复后是可恢复的；以及可靠地使所述失败指示符可用于其他事务，从而所述又一事务可依赖所述失败指示符。

本发明的第二方面的优选特征是，该方法使得所述失败指示符能够触发所述其他事务。

本发明的第二方面的其它优选特征是，该方法还包括步骤：建立一个或多个已递交信号；以及从所述一个或多个已递交信号建立存储器中的一个或多个存储器信号；其中通过修改所述存储器信号中的一个或多个信号而生成所述失败指示符。

本发明的第二方面的其它优选特征是，该方法还包括步骤：记录修改的存储器信号；在所述记录步骤期间暂停所述回退步骤；以及在所述记录步骤后恢复所述回退步骤。

附图说明

现参照各附图以示例的方式说明本发明的一优选实施例，附图是：

图 1 示出依据本发明的数据处理系统。

图 2 是依据本发明的处理方法的一部分的流程图。

图 3 是依据本发明的处理方法的另一部分的流程图。

具体实施方式

图 1 中示出一个具有事务处理监视器(101)、主存储器(102)、同步点管理器(103)、日志(104)和可恢复文件(105)的数据处理系统(100)。还示出第一事务(106)和另一事务(107)。同步点管理器(103)具有一个确信的信号管理器部件(108)。

在这种类型的常规系统中，事务的 ACID 特性确保若某事务例如事务(106)被回退，则它不对可恢复资源，例如可恢复文件(105)，产生外部可见的影响。本实施例通过把可恢复文件(105)用作信号储存库而打破这种约定；由于这在系统失败上呈现鲁棒性(robustness)，它是非常需要

的。

在本实施例中,在提交情况和回退情况二者下,从事务,例如事务(106)发出确信的信号。提交情况正和上面说明的受保护 START 请求类似。允许在回退期间发出信号有益地使得有可能在所有情况下保持应用流,使得应用程序人员不需要使用其它手段检测失败;现在回退能够激发复杂的事务前进。本文中使用术语“确信的”以和常规事务行为相区别,“确信的”行为的形式和事务的常规行为接近,但其允许回退具有持久的结果。这种对常用隔离特性的打破限于在事务之间传送的信号。本实施例的对确信的信号的控制是在同步点管理器(103)内的确信的信号管理器部件(108)中实现的,并且按和可恢复资源管理器相同的方式使用同步点管理器的记录功能。因此可把确信的信号管理器看成是一种专用型的资源管理器。

在该语境下,信号是关于要完成的某工作,即请求,的信息。请求含有足够的用于事务处理系统(101)的信息,以建立并开始执行新事务。例如,在 CICS 术语下,该信息包含要附着的 TRANID 以及在其安全属性下执行该事务的 USERID。除该信息之外,在本实施例中,该请求还携带一个发布事务的状态的指示符:指示该事务是否回退。

在本实施例中,确信的信号可能有三个地点可驻留。每个地点可含有略有不同的数据并且由此对信号给出不同的名字。它们是“已递交信号”、“存储器信号”和“已记录信号”。已递交信号驻留在可恢复文件(105)中。这意味着对已递交信号的所有更新是原子的并且对于系统中的任何其它事务是隔离的。具体地,若该事务失败并执行回退,则从任何其它事务的观点来看,对该已递交信号作出的任何改变也回退。已递交信号为确信信号而构成基储存库,确信信号的内容可通过其它类型的信号扩大。每个已递交信号具有一个足以在该可恢复文件中定位它的标识符;其它类型的信号也含有一个已递交信号标识符。

事务是通过其完成事务处理系统中的任何持久工作的手段,从而通过事务完成已递交信号的每次改变。具体地,事务起源于一已递交信号,并且某事务(通常是另一个事务)会删除或者“消耗”该信号。在本实施例

的事务处理系统中还在主存储器(102)中保存信号的拷贝。它们被称为“存储器信号”。每个存储器信号和一个正运行的事务相关，并且除回退情况之外它通常保持和对应的已递交信号同步。

每个事务具有日志(104)中的记录，它们由同步点管理器(103)管理。当一事务执行由某确信信号要求的工作时，把该信号的一个拷贝(称为“记录的信号”)写入到用于该事务的日志中。

现参照图 2 的流程图，当事务 A 建立一个信号时，以常规方式把已递交信号写入(201)可恢复文件，并建立一个对应的存储器信号(202)。若建立该信号的事务 A 提交(203)，则对可恢复文件提交(205)该信号并且建立(206)一个新事务 B(称为工作事务)以执行该信号涉及的工作。然而，若建立该信号的事务 A 执行回退(204)，则该信号不在可恢复文件中出现并且系统的其余部分不考虑它。事务 B 不被建立。该实施例的优点是不需要对可恢复文件资源语义作出改变。

指定事务 B 来执行由该信号请求的工作。把存储器信号传送(206)给该工作事务以指示要完成该工作。该事务会消耗该已递交信号或者安排成使它在和该事务同步的其它事务中消耗。该信号被记录(207)，即，在事务恢复日志上建立一个记录信号，从而信号处理可保存系统失败。接着通过删除(208)从该可恢复文件(通过可恢复动作)消耗掉该已递交信号；而存储器信号被保留。接着执行由该信号请求的工作(209)。随后，事务 B 或者因为它的处理成功而提交(210)，或者它在失败情况下或在废弃该工作的慎重应用决策下执行回退。这二种情况在同步点处理期间造成不同的动作。

在提交处理的情况下，废弃该存储器信号(212)并且不需要其它动作。在常规方式下将提出(213)该已递交信号的删除动作并且从该可恢复文件永久性地去掉该已递交信号：即可见于其它事务的持久更新。该信号不产生其它事务，但可把它的删除事实用作其它复杂事务处理的启动点。

参照图 3 的流程图，在事务失败的回退处理情况下，以用于可恢复文件的常规方式取消该已递交信号的可恢复删除动作(301)，并且不删除该已递交的信号。该已递交信号的内容和它建立时的内容是相同的，并

且它不包括任何有关现行事务 B 失败的任何信息。为了激发依从于该事务 B 失败的工作，修改该存储器信号以记录失败并且启动一个“失败事务” C(302)来执行记录该失败的工作。在该实施例 中，该存储器信号还记录其它有关该失败的信息，例如错误代码和报文。失败事务 C 和该工作事务的回退处理(303)同步启动。不废弃该存储器信号而是把它传送(304)到失败事务 C 供进一步处理。在可恢复文件回退处理结束之后和回退处理结束之前暂停该工作事务(305)。当通过该失败事务恢复(307)该工作事务时，在常规方式下它可终止(308)并结束同步点处理。

在暂停该工作事务的同时，通过记录(306)修改的存储器信号(即，建立一个记录信号，其是事务恢复日志中的存储器信号的拷贝)启动失败事务 C，从而它可保存系统失败。该行为类似于上面说明的消耗处理的行为。此刻恢复发出该失败事务 C 的该工作事务 B(307)。通过用可恢复动作删除(310)已递交信号和执行失败处理工作，继续失败事务 C。这通常包括：把该工作标记为具有故障，给出原因(311)，并且然后提交。

本实施例有益地提供信号的可靠处理，如进一步参照图 2 和图 3 所示出的那样，在图 2 和图 3 中通过用图左侧的字母 A、B 等标志的线指出可能的系统故障点。

在图 2 中，可在用 A、B、C、D 和 E 标记的逻辑时刻出现系统故障。依次研究每个这样的点以解释本实施例如何保留足够的信息以便重试工作或者继续前进，仿佛未曾中断似的。

- A 常规事务和系统恢复会确保起源事务不起作用；尤其已递交信号(放在可恢复文件中)不会在回退后出现在该文件中。
- B 在起源事务中决定承诺后，系统恢复过程后的系统故障会造成把已递交信号重新存储到可恢复文件中。现在有可能在系统恢复正常工作时检索该信号并执行它在通常方式下指出的工作。
- C 在工作事务已记录该信号之后的系统故障会确保在系统恢复期间找到已递交信号和已记录信号，它们会匹配并且已记录的信号实际上会被忽略。如情况 B 中那样，处理继续。
- D 在删除已递交信号之后但在提交工作事务之前的系统故障仍会造成

由情况 B 和 C 指示的系统恢复。该信号保存在可恢复文件中(系统故障过程中的工作事务将回退, 并且激发删除), 并且在恢复正常系统工作时可起作用。

- E 在承诺工作事务之后的系统故障将造成兑现已递交信号删除的系统恢复。(用于该事务的)事务日志是可废弃的, 并消耗掉该信号。

参照图 3, 可在用 F、G、H 和 J 标记的逻辑时刻出现系统故障。依次研究每个这样的点以解释本实施例如何保留足够的信息以便重试工作或者继续前进, 仿佛未曾中断似的。

- F 在工作事务作出回退决策之后可出现第一个新系统故障点。在系统恢复下将回退和该事务相关的工作。从系统日志恢复已记录信号并且该信号和可恢复文件中的信号相关, 而且会恢复回退处理。在和以前相同的方式下重新启动该失败事务。

- G 和 F 相同地处理该故障点。通过系统恢复也许能或者也许不能识别该失败事务的存在, 但在不论那种情况下会完全激发它。将不会废弃该工作事务, 并且从而它将从系统事务日志完整恢复该信号以及记录故障的责任。

- H 在记录失败信号后有可能已经废弃工作事务, 从而已记录信号是仅有的记录故障的责任记录。在系统恢复时总是通过恢复已记录信号并且在恢复常规处理之前使已记录信号和已递交信号相关, 使这一点得到承认, 在已记录信号和本实施例的工作事务之间的重叠可防止在系统故障的情况下丢失不可缺少的信息。

- J 在故障事务已删除已递交信号之后的系统故障将造成失败事务的回退(在该情况下不消耗掉该信号并且当恢复常规系统功能时重试该失败事务), 或者造成提交该失败的事务(在该情况下可能以常规方式消耗该信号)。常规事务和系统恢复机制处理这些情况。

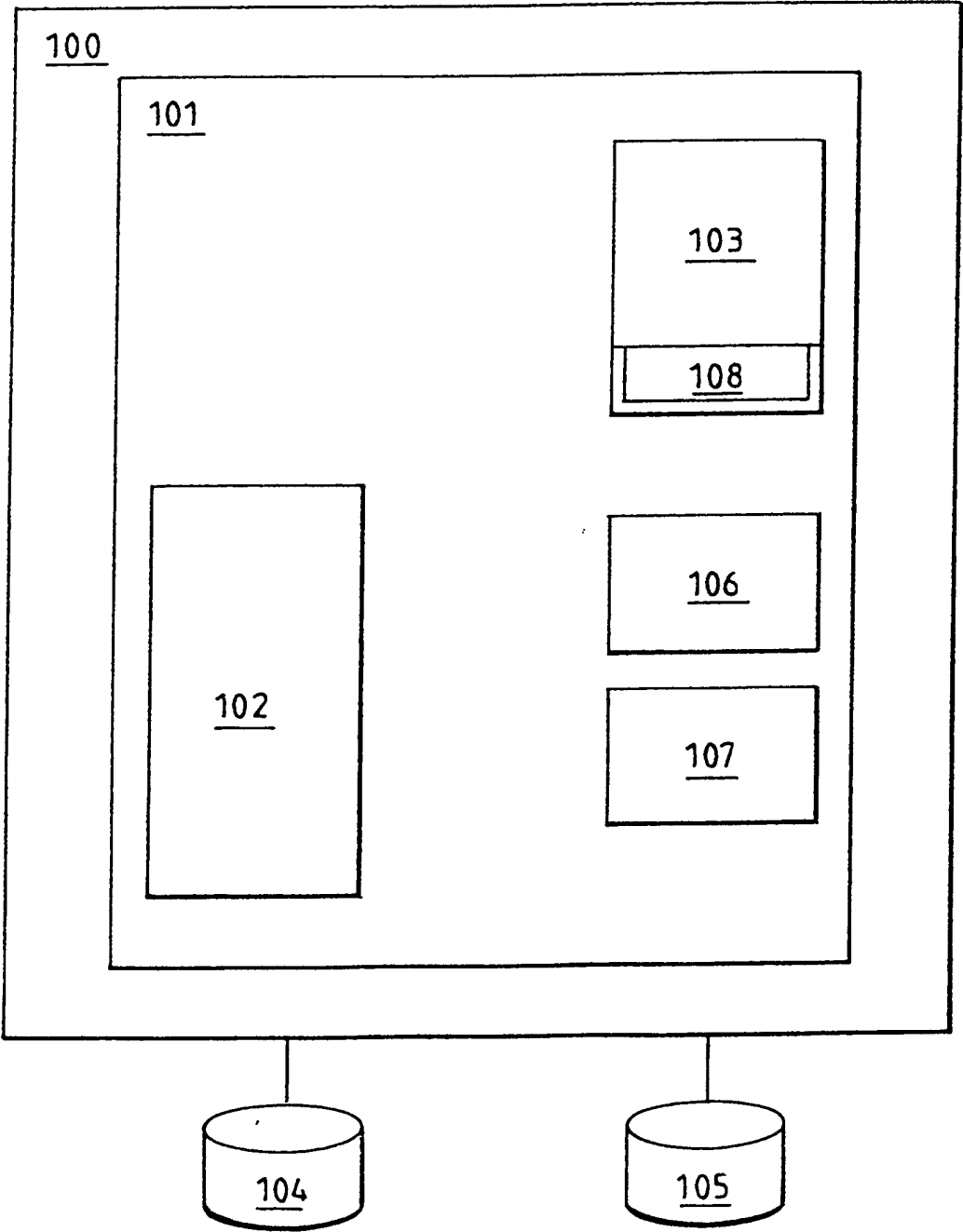


图1

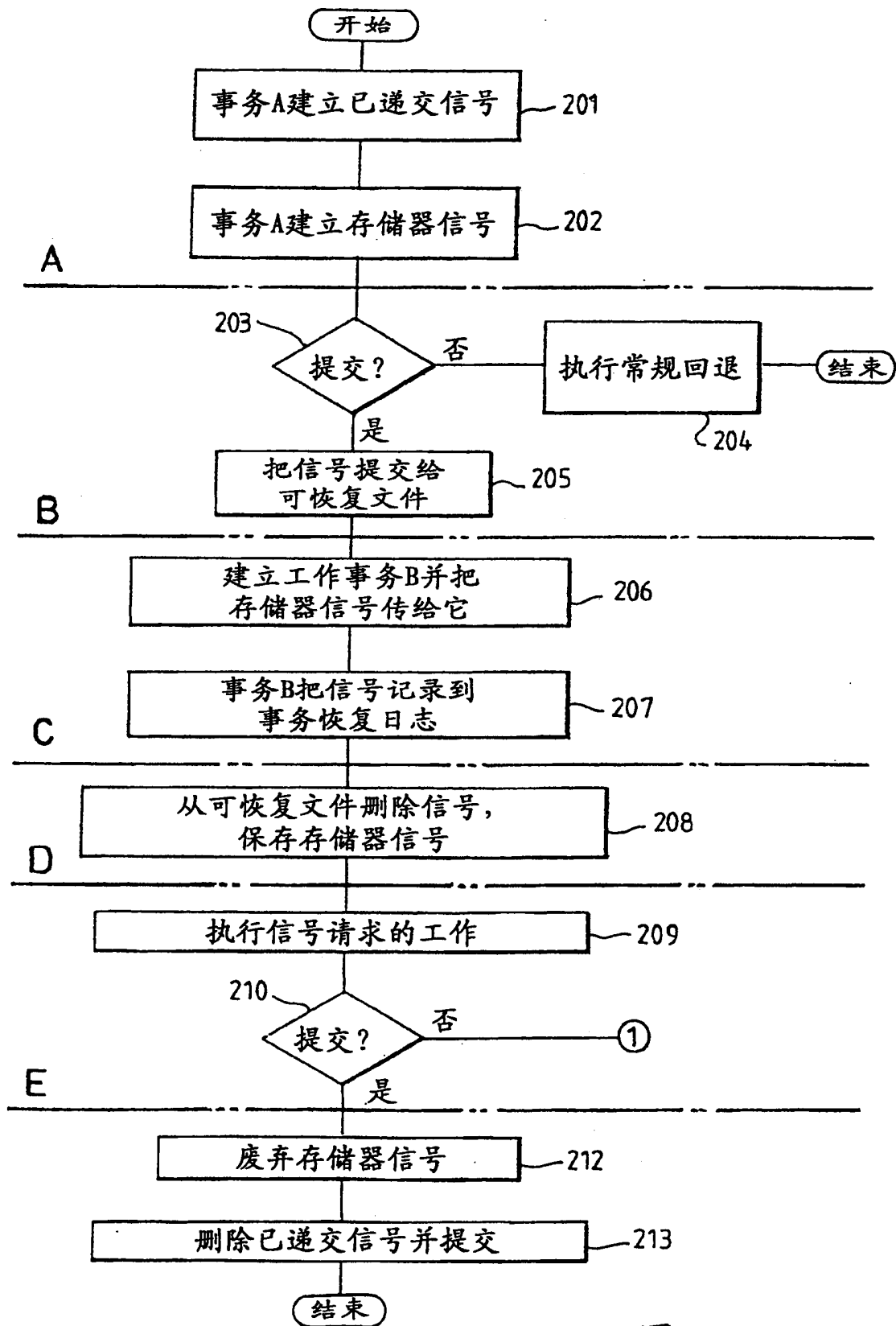


图2

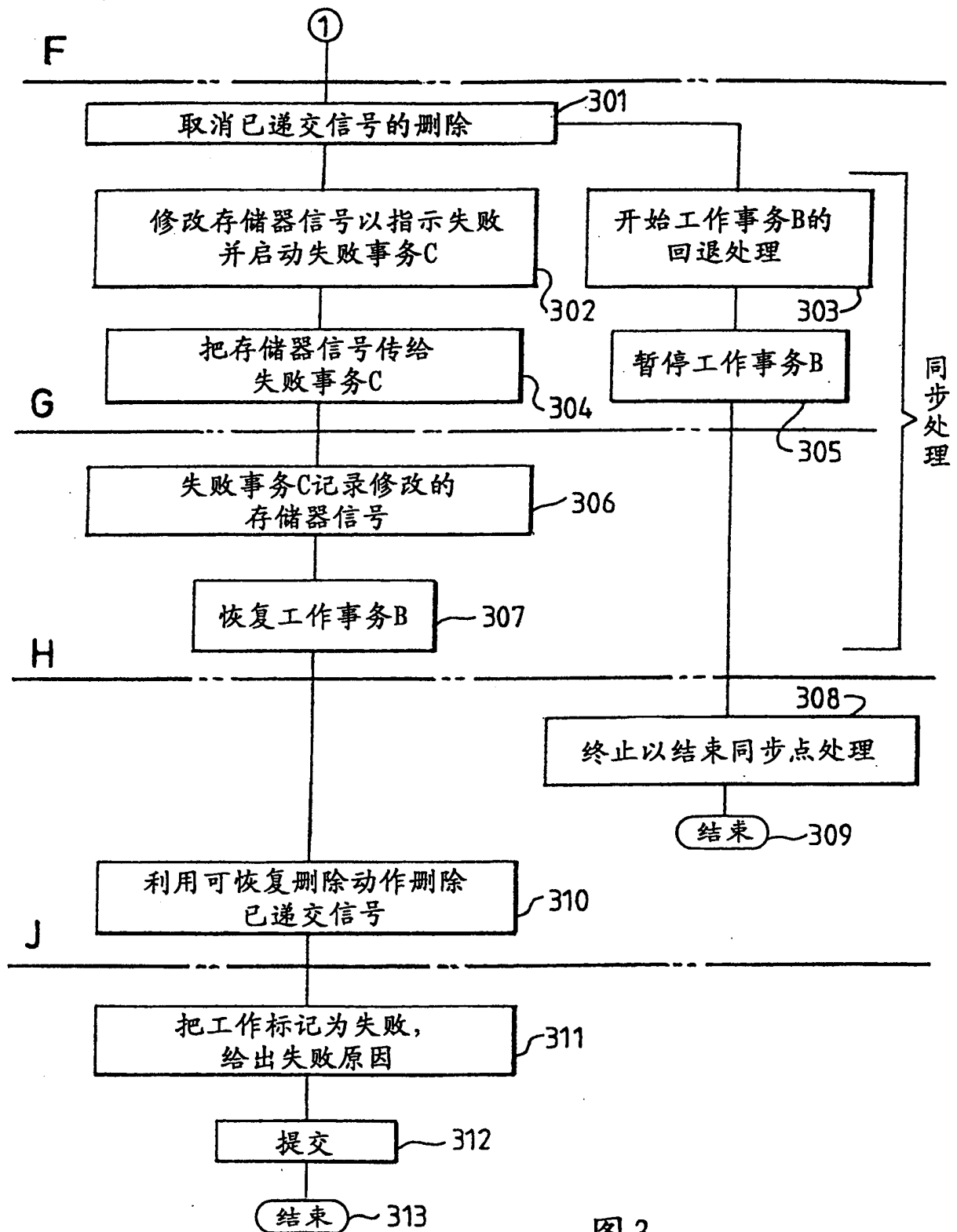


图 3