

一种数据库系统的事务处理方法、装置、电子设备、计算机可读存储介质及计算机程序产品

相关申请的交叉引用

5 本申请基于申请号为 202110546080.0、申请日为 2021 年 05 月 19 日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此引入本申请作为参考。

技术领域

本申请涉及数据库技术领域，涉及一种数据库系统的事务处理方法、装置、电子设备、计算机可读存储介质及计算机程序产品。

10 背景技术

随着数据库技术的发展，在数据库系统中如何识别数据异常成为了一个关键问题。目前有两种识别数据异常的方法，其一是利用封锁技术，依赖锁的互斥机制，来识别数据异常，其二是利用依赖图技术，在由并发事务构成的依赖图中确认是否存在环，如果存在环，则说明存在数据异常。然而，封锁技术限制了数据库系统的并发度，导致事务处理效率低下，而依赖图技术又需要遍历每个并发事务以识别出环的存在，导致事务处理效率较低。

发明内容

本申请实施例提供了一种数据库系统的事务处理方法、装置、电子设备、计算机可读存储介质及计算机程序产品，能够提高数据库系统的事务处理效率。

20 本申请实施例提供了一种数据库系统的事务处理方法，该方法包括：

确定目标事务的并发事务，所述并发事务与所述目标事务之间包括作用于同一变量的读写操作，所述目标事务为待提交事务；

获取所述目标事务的读集和所述并发事务的写集之间的第一交集，并获取所述目标事务的写集和所述并发事务的读集之间的第二交集；

25 当所述第一交集和所述第二交集至少有一项为非空数据集、且所述目标事务与所述并发事务冲突时，基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列表，确定数据异常类型，所述目标变量列表包括所述目标事务和所述并发事务所作用的同一变量。

本申请实施例提供了一种事务处理装置，该装置包括：

30 第一确定模块，配置为确定目标事务的并发事务，所述并发事务与所述目标事务之间包括作用于同一变量的读写操作，所述目标事务为待提交事务；

第二确定模块，配置为获取所述目标事务的读集和所述并发事务的写集之间的第一交集，并获取所述目标事务的写集和所述并发事务的读集之间的第二交集；当所述第一交集和所述第二交集至少有一项为非空数据集、且所述目标事务与所述并发事务冲突

35 时，基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列

表，确定数据异常类型，所述目标变量列表包括所述目标事务和所述并发事务所作用的同一变量。

本申请实施例提供了一种电子设备，该电子设备包括处理器和存储器，该存储器用于存储至少一条计算机程序，该至少一段计算机程序由该处理器加载并执行时，实现本申请实施例提供的数据库系统的事务处理方法。

本申请实施例提供了一种计算机可读存储介质，该计算机可读存储介质中存储有至少一条计算机程序，该至少一条计算机程序由处理器加载并执行时，实现本申请实施例提供的数据库系统的事务处理方法。

本申请实施例提供了一种计算机程序产品，该计算机程序产品包括计算机程序或指令，该计算机程序或指令被处理器执行时，实现本申请实施例提供的数据库系统的事务处理方法。

本申请实施例提供的技术方案带来的有益效果至少包括：在确定目标事务的并发事务后，在该目标事务的读集和该并发事务的写集之间的第一交集和该目标事务的写集和该并发事务的读集之间的第二交集中，存在非空数据集的情况下，对目标事务与并发事务是否冲突进行判断，若目标事务与并发事务冲突，则根据这两个事务所作用的相同变量以及相同变量的版本变更次数，来确定数据异常类型；如此，能够在目标事务的执行过程中，确认目标事务与并发事务是否真正构成冲突，另外通过两个事务所作用的相同变量来确定数据异常类型，能够识别数据库系统中的各种数据异常，保证数据状态的一致性，从而提高数据库系统的事务处理效率。

附图说明

为了更清楚地说明本申请实施例中的技术方案，下面将对本申请实施例描述中需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

- 图 1 是根据本申请实施例提供的一种事务处理方法的实施环境示意图；
- 图 2 是根据本申请实施例提供的一种冲突环图的示意图；
- 图 3 是根据本申请实施例提供的另一种冲突环图的示意图；
- 图 4 是根据本申请实施例提供的又一种冲突环图的示意图；
- 图 5 是根据本申请实施例提供的一种事务处理方法的流程图；
- 图 6 是根据本申请实施例提供的另一种事务处理方法的流程图；
- 图 7 是根据本申请实施例提供的一种单元数据异常 CCG 图的示意图；
- 图 8 是根据本申请实施例提供的一种双元数据异常 CCG 图的示意图；
- 图 9 是根据本申请实施例提供的一种多元数据异常 CCG 图的示意图；
- 图 10 是根据本申请实施例提供的一种获取目标冲突环图的示意图；
- 图 11 是根据本申请实施例提供的一种事务处理装置的结构示意图；
- 图 12 是根据本申请实施例提供的一种服务器的结构示意图。

具体实施方式

为使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施

例中所描述的实施方式并不代表与本申请相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本申请的一些方面相一致的装置和方法的例子。

本申请中术语“第一”“第二”等字样用于对作用和功能基本相同的相同项或相似项进行区分，应理解，“第一”、“第二”、...、“第七”之间不具有逻辑或时序上的依赖关系，也不对数量和执行顺序进行限定。还应理解，尽管以下描述使用术语第一、第二等来描述各种元素，但这些元素不应受术语的限制。

这些术语只是用于将一个元素与另一个元素区别开。例如，在不脱离各种示例的范围的情况下，第一事务能够被称为第二事务，并且类似地，第二事务也能够被称为第一事务。第一事务和第二事务都可以是事务，并且在某些情况下，可以是单独且不同的事务。

其中，至少一个是指一个或一个以上，例如，至少一个事务可以是一个事务、两个事务、三个事务等任意大于等于一的整数个事务。而多个是指两个或者两个以上，例如，多个事务可以是两个事务、三个事务等任意大于等于二的整数个事务。

在介绍本申请实施例之前，需要引入一些云技术领域内的基本概念。

云技术 (Cloud Technology): 是指在广域网或局域网内将硬件、软件、网络等系列资源统一起来，实现数据的计算、储存、处理和共享的一种托管技术，也即是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台技术和应用技术等总称，可以组成资源池，按需所用，灵活便利。本申请实施例所提供的数据库系统的事务处理方法（以下简称事务处理方法），可以是基于云技术实现的。

云存储 (Cloud Storage): 是在云计算概念上延伸和发展出来的一个新的概念，分布式云存储系统（以下简称存储系统）是指通过集群应用、网格技术以及分布存储文件系统等功能，将网络中大量各种不同类型的存储设备（存储设备也称之为存储节点）通过应用软件或应用接口集合起来协同工作，共同对外提供数据存储和业务访问功能的一个存储系统。本申请实施例所提供的事务处理方法，可以应用于云存储中，以对云存储中的异常进行识别。

数据库系统 (Database): 一种电子化的文件柜，也即是存储电子文件的处所，用户可以对文件中的数据进行新增、查询、更新和删除等操作。所谓“数据库系统”是以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。本申请实施例所提供的事务处理方法，可以应用于数据库系统中，以对数据库系统中的异常进行识别。

本申请实施例所涉及的数据库系统，可以是单机数据库系统、单机以事务为主的数据库系统、单机以分析型为主但需要事务处理能力的数据库系统，可以是泛指非关系型数据库 (Non-relational SQL, NoSQL) 体系，还可以是分布式数据库系统、分布式大数据处理系统，对于分布式数据库系统而言，当不同变量分布存储在不同的物理节点上时，对应于数据状态一致性模型中存在两个及两个以上变量的情况，数据状态一致性模型以及相应的事务处理流程均将在后续描述中进行详细说明。

在数据库系统中可以包括至少一个节点设备，每个节点设备的数据库中存储有数据表，每个数据表可以用于存储一个或多个数据项（也称为变量版本）。其中，节点设备的数据库可以为任一类型的分布式数据库，可以包括关系型数据库或者非关系型数据库中至少一项，例如结构化查询语言 (Structured Query Language, SQL) 数据库、NoSQL、泛指各种新式的可拓展/高性能数据库 (NewSQL) 等，在本申请实施例中对数据库的类型不作限定。

在一些实施例中，还可以应用于一种基于区块链技术的数据库系统（以下简称为“区块链系统”），上述区块链系统在本质上属于一种去中心化式的分布式数据库系统，采用

共识算法保持区块链上不同节点设备所记载的账本数据一致，通过密码算法保证不同节点设备之间账本数据的加密传送以及不可篡改，通过脚本系统来拓展账本功能，通过网络路由来进行不同节点设备之间的相互连接。

在区块链系统中可以包括一条或多条区块链，区块链是一串使用密码学方法相关联产生的数据块，每一个数据块中包含了一批次网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

区块链系统中节点设备之间可以组成点对点（Peer To Peer, P2P）网络，P2P 协议是一个运行在传输控制协议（Transmission Control Protocol, TCP）协议之上的应用层协议。在区块链系统中，任一节点设备可以具备如下功能。

1）路由，节点设备具有的基本功能，用于支持节点设备之间的通信；

2）应用，用于部署在区块链中，根据实际业务需求而实现特定业务，记录实现功能相关的数据形成账本数据，在账本数据中携带数字签名以表示数据来源，将账本数据发送至区块链系统中的其他节点设备，供其他节点设备在验证账本数据来源以及完整性成功时，将账本数据添加至临时区块中；其中，应用实现的业务可以包括钱包、共享账本和智能合约等；

3）区块链，包括一系列按照先后的时间顺序相互接续的区块，新区块一旦加入到区块链中就不会再被移除，区块中记录了区块链系统中节点设备提交的账本数据。

在一些实施例中，每个区块中可以包括本区块存储交易记录的哈希值（本区块的哈希值）以及前一区块的哈希值，各区块通过哈希值连接形成区块链；另外，区块中还可以包括有区块生成时的时间戳等信息。

由于数据库系统中事务并发控制的正确程度可以通过一致性和隔离性来描述，因此在介绍本申请实施例之前，下面先对一致性和隔离性进行解释说明。

一、隔离性；事务隔离级别通过能否规避某种数据异常而进行定义，可能涉及到的数据异常包括：A、脏读，指一个事务读取到了另一事务尚未提交的数据项；B、不可重复读，指一个事务对同一数据项重复读取两次却得到了不同的结果；C、幻读，指事务在操作过程中进行两次谓词查询（范围查询），第二次查询的结果包含了第一次查询的结果中未出现的数据项或者缺少了第一次查询的结果中出现的的数据项。

基于能够解决上述 ABC 三种数据异常，数据库国际标准国家标准学会（A-National Standards Institute, ANSI）SQL 提出四种隔离级别，以对上述 ABC 三种已知的数据异常加以区分，以在允许某些数据异常存在的情况下提高事务处理效率。

另外，还包括四种隔离级别，分别包括：a、读未提交：允许如上 ABC 三种数据异常发生；b、读已提交：不允许脏读发生；c、可重复读：不允许脏读、不可重复读发生；d、可串行化：如上 ABC 三种数据异常均不能发生。易知，这四种隔离级别均不允许脏写异常发生，脏写异常是指两个未提交事务修改了同一个数据项。在 ANSI SQL 制定标准时已知的数据异常种类不多，后续不断有新的数据异常被发现，除了上述几种数据异常之外，数据异常还包括：丢失更新异常、读偏序异常、写偏序异常、读写偏序异常、锯齿波写偏序异常、串行并发现象（Serial-Concurrent-Phenomenon）异常、交叉现象（Cross-Phenomenon）异常、因果丢失异常、因果反转异常、旧读异常和未来读异常等。

二、一致性；在数据库技术中，“一致性”一词可以表达两重含义，一是指事务一致性，二是指分布式一致性。

数据库的事务一致性为：在事务的操作下，数据库的数据状态从一个一致的状态变迁为另一个一致的状态。上述“一致的状态”是指满足数据库系统预先定义的一些规则的数据状态，比如，这些规则可以包括约束、级联、触发器以及三者之间任意的组合（属于数据的逻辑语义）；写偏序异常违反的就是特定数据之间的约束，这里的约束属于用

户语义所限定的数据的一致性。

数据库的事务处理技术中的并发访问控制技术,用于发现和解决事务并发操作在数据项上是否会造成数据异常、以及如何消除数据异常的问题,ANSI SQL 提出了四种数据异常和隔离级别,各种并发访问控制算法也得以发展;例如,基于封锁的并发访问控制
5 控制技术、基于时间戳的并发访问控制技术、基于多版本并发控制 (Mutil-Version Concurrency Control, MVCC) 的并发访问控制技术、基于乐观并发控制 (Optimistic Concurrency Control, OCC) 的并发访问控制技术等。

数据库的分布式一致性也称为共享数据对象操作一致性,是针对整个数据库系统的系统级含义,是指要想保证数据在数据库系统中保持一致,还要求数据库系统符合两个
10 特性,一个是可串行性 (Serializability),另一个是可恢复性 (Recoverability)。可串行性也即是说上述在隔离性中的可串行化隔离级别,可串行性保证了并发操作的正确性,而可恢复性是指已经提交的事务未曾读过被回滚的事务写过的数据 (指不会发生脏读异常),可恢复性保证了事务被回滚后数据回到之前的一致状态,被回滚的事务不会对数据的一致性造成影响,数据库系统的一致性状态是可恢复的。

基于上述概念可知一致性和隔离性均与数据异常是息息相关的,故需要在数据库系统中识别并规避数据异常。这里,涉及到的数据库异常可以包括:脏读数据异常、脏写
15 数据异常、不可重复读数据异常、幻读数据异常、丢失更新数据异常、读偏序数据异常、写偏序数据异常、串行并发现象数据异常、交叉现象数据异常等。

一般来说,在识别数据异常时依赖于可串行化原理,通常可以采用两种识别数据异常的策略。一是利用封锁技术,依赖锁的互斥机制来避免数据异常的发生,比如 DB2、
20 Informix 合 MySQL 等数据库系统的可串行化隔离级别都是使用封锁技术;二是利用依赖图技术,通过在并发事务所构成的依赖图中确认是否存在环,如果存在环则需要打破环的存在从而消除潜在的数据异常。

然而,封锁技术严重限制了数据库系统的并发度,导致事务处理效率低下,而依赖
25 图技术则需要在并发事务之间遍历每个事务以识别出环的存在,导致事务处理效率较低;而为了识别出环是否存在,不仅需要借助各种衍生出的算法,而且需要复杂的数据结构来表示并发事务与被操作的数据项 (也即是变量版本) 之间的关联关系,提高了数据库系统的复杂性。

此外,为了进行事务处理,还可采用基于依赖图和冲突可串行化技术 (该技术基于
30 可串行化原理) 的简化实现方式,比如数据库 (PostgreSQL) 的快照隔离 (Snapshot Isolation, SSI) 技术,SSI 技术中能够提前识别出可能会造成依赖图中存在环的必要不充分条件:一个事务存在一条读写冲突入边和读写冲突出边;如果任一事务满足上述条件,则认为存在数据异常。

另外,上述识别数据异常的方法还存在如下问题。

35 第一,认知割裂不统一:封锁技术、依赖图技术和 SSI 技术,均割裂了每一种数据异常的识别方式,使得在识别数据异常时只能逐个去认知是否符合每一种数据异常;也即是说,不能用统一的角度去识别、认知这些不同种类的数据异常,导致针对数据异常的认知思路复杂且不清晰统一,复杂度较高。

第二,认知有限不全面:封锁技术、依赖图技术和 SSI 技术,均不能穷尽是否还有
40 未知的新的数据异常。而当数据异常存在时,将会为整个数据库系统带来损失;比如,当数据库系统处于非可串行化隔离级别下,如有新的数据异常,那么会导致系统中出现数据不一致错误。

第三,识别原理不精准:可串行化是数据存在异常的充分不必要条件,如此在数据库系统实现中虽然可避免出现新的数据异常,但是也有可能会把一些不存在数据异常的

情况也当作了数据异常,相当于对于数据异常进行了误判;比如事务1写入了变量版本X1且读取了变量版本Y1,事务2读取了变量版本X1且写入了变量版本Y1,上述一系列的数据库操作可以记为“W1(X1) R2(X1) W2(Y1) R1(Y1) C2 C1”,这一系列的操作是不可串行化的,但并不存在数据异常。

- 5 有鉴于此,本申请实施例提供一种事务处理方法,给出了在事务处理过程中识别数据异常并规避数据异常的方式,提供了一种通用形式的冲突环图来定义数据异常,在确保数据异常定义的正确性的前提下,使得数据异常定义形象化;并且,对数据异常进行了细化分类,能够提高数据库系统的事务处理效率,下面将进行详述。

10 图1是本申请实施例提供的一种事务处理方法的实施环境示意图。参见图1,本申请实施例可以应用于数据库系统中的分布式数据库系统,该系统中可以包括网关服务器101(示例性地示出了三个网关服务器)、全局时间戳生成集群102、分布式存储集群103以及分布式协调系统104(例如ZooKeeper);在分布式存储集群103中可以包括节点设备1至m(m为大于或等于1的正整数),每个节点设备包括可以是数据节点设备,还可以是协调节点设备。

15 其中,网关服务器101用于接收外部的读写请求,并将读写请求对应的读写事务分发至分布式存储集群103;比如,用户在登录终端上的应用客户端之后,触发应用客户端生成读写请求,调用分布式数据库系统提供的应用程序编程接口(Application Programming Interface, API)将该读写请求发送至网关服务器101;其中,API可以是任意一种关系型数据库系统提供的API,比如,该API可以是MySQL API。

20 在一些实施例中,该网关服务器101可以与分布式存储集群103中的任一个数据节点设备或任一协调节点设备可以是同一个物理机,也即是,让某个数据节点设备或协调节点设备兼具网关服务器101的功能。

全局时间戳生成集群102用于生成全局事务的全局提交时间戳(Global Timestamp, Gts),该全局事务又称为分布式事务,是指涉及到多个数据节点设备的事务;例如全局读事务可以涉及到对多个数据节点设备上存储数据的读取,又例如,全局写事务可以涉及到对多个数据节点设备上的数据写入。全局时间戳生成集群102在逻辑上可以视为一个单点,也可以通过一主三从(分别是一个主设备和与该主设备对应的三个辅助设备)的架构来提供具有更高可用性的服务,采用集群的形式来实现该全局提交时间戳的生成;当一主三从架构中的一个设备故障时,可以通过一主三从架构中的其他设备继续实现该全局提交时间戳的生成,从而可以防止单点故障,也就规避了单点瓶颈问题。

30 在本申请实施例中,全局提交时间戳是一个在分布式数据库系统中全局唯一且单调递增的时间戳标识,能够用于标志每个事务全局提交的顺序,以此来反映出事务之间在真实时间上的先后关系(事务的全序关系),全局提交时间戳可以采用物理时钟、逻辑时钟、混合物理时钟或者混合逻辑时钟(Hybrid Logical Clock, HLC)中至少一项,本申请实施例不对全局提交时间戳的类型进行限定。

示例性地,全局提交时间戳可以采用混合物理时钟的方式生成,全局提交时间戳可以由八字节组成,每字节包括八位,共64位;其中,前44位可以为物理时间戳的取值(也即Unix时间戳,精确到毫秒),这样共计可以表示 2^{44} 个无符号整数,因此理论上

40 一共可以表示约为 $557.8 \left(\frac{2^{44}}{1000 \times 60 \times 60 \times 24 \times 365} = 557.8 \right)$ 年的物理时间戳;其中,后20位可以为在某一毫秒内的单调递增计数,这样每毫秒有 2^{20} 个(约100万个)计数,基于上述数据结构,如果单机(任一数据节点设备)的事务吞吐量为10w/s,理论上可以支持包含1万个节点设备的分布式存储集群103;同时,全局提交时间戳的数量代表了系统理论上所能支持的总事务数,基于上述数据结构,理论上系统可以支持 $(2^{44}-1) \times 2^{20}$ 个事务。这里是对一种全局提交时间戳的定义方法的示例性说明,根据业务需求的不同,

可以对全局提交时间戳的位数进行扩展,以满足对更多的节点数、事务处理数的支持,本申请实施例不对全局提交时间戳的定义方法进行限定。

在一些实施例中,该全局时间戳生成集群 102 可以是物理独立的,也可以和分布式协调系统 104 合并到一起。

5 其中,分布式存储集群 103 可以包括数据节点设备和协调节点设备,每个协调节点设备可以对应于至少一个数据节点设备,数据节点设备与协调节点设备的划分是基于事务的不同而划分的;以某一全局事务为例,全局事务的发起节点可以称为协调节点设备,全局事务所涉及的其他节点设备称为数据节点设备,数据节点设备或协调节点设备的数量可以是一个或多个,本申请实施例不对分布式存储集群 103 中数据节点设备或协调节点设备的数量进行限定。由于本申请实施例所提供的分布式数据库系统中缺乏全局事务管理器,因此在该系统中可以采用组织分布式事务规范(eXtended Architecture, XA)/二阶段提交(Two-Phase Commit, 2PC)技术来支持跨节点的事务(全局事务),保证跨节点写操作时数据的原子性和一致性;此时,协调节点设备相当于 2PC 算法中的协调者,而该协调节点设备所对应的各个数据节点设备相当于 2PC 算法中的参与者。

15 在本申请实施例中,每个数据节点设备或协调节点设备可以是单机设备,也可以采用主备结构(也即是为了一主多备集群);如图 1 所示,以节点设备(数据节点设备或协调节点设备)为一主两备集群为例进行示意,每个节点设备中包括一个主机和两个备机,每个主机或备机都对应配置有代理(Agent)设备,代理设备可以与主机或备机是不同的物理设备,当然,代理设备还可以作为主机或备机上的一个代理模块;以节点设备 1 为例,节点设备 1 包括一个主数据库及代理设备(主 Database+Agent,简称主 DB+Agent),此外还包括两备数据库及代理设备(备 Database+Agent,简称备 DB+Agent)。

20 示例性地,每个节点设备所对应的主机或备机的数据库实例集合称为一个集合(SET);例如,假设某一节点设备为单机设备,那么该节点设备的 SET 仅为该单机设备的数据库实例,假设某一节点设备为一主两备集群,那么该节点设备的 SET 为主机数据库实例以及两个备机数据库实例的集合;此时可以基于云数据库的强同步技术来保证主机的数据与备机的副本数据之间的一致性;这里,每个 SET 可以进行线性扩容,以应付大数据场景下的业务处理需求,在一些金融业务场景下,全局事务通常是指跨 SET 的转账。

30 分布式协调系统 104 可以用于对网关服务器 101、全局时间戳生成集群 102 或者分布式存储集群 103 中至少一项进行管理;在本申请实施例中,可以通过终端上的调度器(Scheduler)访问该分布式协调系统 104,从而基于前端的调度器来控制后端的分布式协调系统 104,实现对各个集群或服务器的管理。例如,可以通过调度器来控制 ZooKeeper 将某一个节点设备从分布式存储集群 103 中删除,也即是使得某一个节点设备失效。

35 上述图 1 提供了一种轻量级的全局事务处理的架构图,是一种类分布式数据库系统。整个分布式数据库系统可以看作是共同维护一个逻辑上的数据表,这个数据表中存储的数据通过主键被打散到分布式存储集群 103 中的各个节点设备中,每个节点设备上存储的数据是独立于其他节点设备的,从而实现了节点设备对逻辑数据表的水平切分。由于在上述系统中能够将各个数据库中各个数据表水平切分后进行分布式地存储,因此,这种系统也可以称为具有“分库分表”的架构。

40 在上述分布式数据库系统中,已经基于 XA/2PC 技术实现了写操作时数据的原子性和一致性,如图 1 中示出的分布分表架构,为一个局部事务管理器,不是全局的,故还存在读操作时数据的一致性问题;而本申请实施例提供的事务处理方法,能够实现分布式事务处理,并且通过构造轻量的、去中心化的分布式事务处理机制,能够为分布式数据库系统提供水平扩展等能力,提升分布式数据库系统的应用范围,进而能够解决写操

作时数据的原子性和一致性、以及读操作时数据的一致性问题,从而提升事务处理效率。

本申请实施例提供的事务处理方法,可以应用于上述采用了分库分表架构的分布式数据库系统中;例如,该分布式数据库系统为分布式事务型数据库系统,当然也可以是分布式关系型数据库系统;此外,本申请实施例提供的事务处理方法也可以应用于一些
5 单机数据库系统中;当本申请实施例提供的事务处理方法应用于分布式数据库系统时,能够实现分布式事务处理,且能够通过灵活地隔离级别,提高事务处理效率。

在一些实施例中,上述网关服务器 101、全局时间戳生成集群 102、分布式存储集群 103 以及分布式协调系统 104 所构成的分布式数据库系统,可以视为一种向用户终端提供数据服务的服务器,该服务器可以是独立的物理服务器,也可以是多个物理服务器
10 构成的服务器集群或者分布式系统,还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、内容分发网络(Content Delivery Network, CDN)、以及大数据和人工智能平台等基础云计算服务的云服务器。这里,上述用户终端可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表等,但并不局限于此。终端以及服务器可以通过有线或无线通信方式进行直接
15 或间接地连接,本申请在此不作限制。

下面将对数据库系统中所涉及到的一些基本术语以及符号表示进行介绍。

事务:是数据库管理系统在执行操作的过程中的一个逻辑单位,由一个有限的数据库操作序列构成,是数据库系统操作的最小执行单位。

变量,事务是数据库关系系统中的一个数据单位,变量是数据库操作的作用者(或者说被操作的数据对象);在一些实施例中,变量也称为数据项。其中,一个变量可以是一个元组(Tuple)或者记录(Record),也可以是一个页面(Page)或者一个表(Table)对象等。一个变量可以包含若干变量版本(简称为“版本”),每当事务对变量进行更新时,则会添加新的变量版本,变量的各个变量版本可以以自然数作为版本号标识,版本号越大,表示变量版本越新。
20

操作:一个数据库操作由操作类型、事务和变量版本三部分构成;其中,操作类型可以包括读(Read, R)和写(Write, W)两种。例如,事务 T_i 读取变量 x ,生成了变量 x 的新版本 x_m ,上述读操作可以记为 $R_i[x_m]$;又例如,事务 T_j 对变量 y 进行更新,生成了变量 y 的新版本 y_n ,上述写操作可以记为 $W_j[y_n]$;其中, i 和 j 为自然数,分别为对应操作所属的事务号, m 和 n 为自然数,分别为对应变量的版本号。另外,在无需
25 区别 R 和 W 时,也可以用 O 来代表一个操作。

可串行化:可串行化是一个调度,也可以理解为多个事务之间的执行方式;多个事务之间的执行有先后顺序,如果事务之间没有共同操作的变量,则事务之间的执行顺序前后置换是没有关系的;但是如果事务间存在共同操作的变量,则事务间先后执行的顺序则需要区分;对于存在共同操作的变量的多个并发执行的事务,如果其执行结果“等价”于某个“串行化调度”,则这个调度为“可串行化的调度”。满足“可串行化的调度”
30 则具有了可串行化属性。因此,可串行化保证了多个事务并发时的执行顺序对数据的一致性没有影响。

数据库系统中的每个事务拥有两个事务数据集合,分别为事务的写集和事务的读集,其含义为:写集 $DSW(T)$ 用于存储事务 T 所写入的新的数据版本,读集 $DSR(T)$ 用于存储事务 T 所读取的数据版本。
40

本申请实施例所提供的事务处理方法,是在数据状态一致性模型(也可以简称为“数据状态模型”)的基础上运行的,本申请实施例基于冲突可串行化理论,对数据异常进行了统一,并提供了一种通用形式的冲突环图,用于表示事务之间的关联关系以及事务对应的读写操作所作用的变量,下面对本申请涉及的基础概念进行介绍。

1、调度；一个调度是指由一组事务的多个操作组成的序列，记为 s 。一个调度 s 中所有的事务记为 $TS(s)$ ，一个调度 s 中所有的操作记为 $O(s)$ ，一个调度 s 中所有的带有版本号标识变量的集合记为 $VS(s)$ 。例如，调度 $s = R_1[x_0] R_2[y_0] W_3[x_1] W_4[y_1] R_2[x_1] R_1[y_1]$ ，表示该调度 s 的操作序列来自 4 个事务，即 $TS\{T_1, T_2, T_3, T_4\}$ ，涉及 2 个变量 x 和 y ，且变量 x 和 y 分别有两个版本。

示意性地，如果一个调度 s 中，任意两个事务的操作各自以事务为单位相邻，也即是，当执行完一个事务的全部操作后再执行下一个事务，则称该调度 s 为串行调度。例如，一个串行调度 s 表示为 $s = R_1[x_0] R_1[y_0] R_2[y_0] R_2[x_0] W_3[x_1] W_4[y_1]$ 。

2、冲突 (Conflict)；冲突也称为冲突操作，当有两个操作满足如下三个条件，则称这两个操作为冲突操作。其中，三个条件为：1) 两个操作属于不同的事务；2) 至少一个操作为写操作；3) 两个操作所操作的对象为同一个对象（读同一个对象或写同一个对象），在本申请实施例中，对象也称为变量。

示意性地，设 $O_i[x_m] O_j[x_n]$ ($i \neq j, m < n$) 为调度 s 中的两个事务的操作，分别为事务 T_i 和 T_j 作用于同一个变量 x 上的两个操作。若 $O_i[x_m] O_j[x_n]$ 属于 $R_i[x_m]W_j[x_{m+1}]$ 、 $W_i[x_m]R_j[x_m]$ 和 $W_i[x_m]W_j[x_{m+1}]$ 三种冲突中的任一种，则称 $O_i[x_m] O_j[x_n]$ 为冲突，冲突在 O_i 和 O_j 之间构成偏序关系，将包含有调度 s 中全部冲突的集合记为 $conf(s)$ 。在本申请实施例中，上述三种冲突可以依次分别简称为 RW、WR 以及 WW。

3、冲突等价 (Conflict Equivalence)；对于不同的调度 s 和 s' ，若满足如下两个条件，则称 s 和 s' 为冲突等价。其中，两个条件为：1) 调度 s 和 s' 包括同样的事务集合，也即是每个事务中的操作的顺序是固定的，不能在不同的调度下发生变化；2) 调度 s 和 s' 包括同样的冲突集合。

示意性地，对于不同的调度 s 和 s' ，若 $O(s) = O(s')$ 且 $conf(s) = conf(s')$ ，则称 s 和 s' 为冲突等价，记为 $s \approx s'$ 。

例如，以下为两个冲突等价的调度。

$$s = R_1[x_0] R_1[y_0] W_2[x_1] W_1[y_1] R_2[z_0] W_1[x_2] W_2[y_2] ;$$

$$s' = R_1[y_0] R_1[x_0] W_1[y_1] W_2[x_1] W_1[x_2] R_2[z_0] W_2[y_2] .$$

4、冲突可串行化 (Conflict Serializability)；若一个调度 s 存在一个串行化调度 s' ，使得这两个调度为冲突等价（也即是满足 $s \approx s'$ ），则称 s 为冲突可串行化调度。

5、冲突环图 (Conflict Cycle Graph, CCG)；示意性地，在一个调度 s 中，存在多个事务 $t \in TS(s)$ ，若多个事务 t 构成的图满足如下三个条件，则称该环图为冲突环图，其中， $CCG(s) = (V, E)$ 。下面对三个条件进行说明。

1) $V = VS(s)$ ，表示为 CCG 图的顶点集，该顶点集中的元素称为顶点，一个顶点对应一个事务；其中，一个顶点 T_i 满足 $T_i \in TS(s)$ ；

2) E 为 $V \times V$ 的子集，也可以称为边集，该边集中的元素称为有向边，记为 $\langle T_i, T_j \rangle$ ，也可以简称为边，其中， $\langle T_i, T_j \rangle \in E \Leftrightarrow i \neq j \wedge (\exists O_i \in T_i) (\exists O_j \in T_j) (O_i, O_j) \in conf(s)$ ；因此，有向边也可以记为与上述“冲突”相似的形式，例如，将有向边记为 $W_i[x_m]W_j[x_{m+1}]$ ，本申请实施例对此不作限定。

3) $\langle T_i, T_j \rangle \in E, \langle T_j, T_k \rangle \in E, \dots, \langle T_p, T_i \rangle \in E, i \neq j \neq k \neq \dots \neq p$,
则 CCG 图为一个有向环构成的图。

例如, 参考图 2, 图 2 是本申请实施例提供的一种冲突环图的示意图。如图 2 所示,
图中 T_1 和 T_2 分别为冲突环图 2-1 的顶点, 且 T_1 和 T_2 之间通过两条边相连, 构成冲突
5 环图; 其中, 两条边分别表示为 $R_1[x_0]W_2[x_1]$ 和 $W_2[x_1]R_1[x_1]$ 。

在本申请实施例中, 基于冲突环图, 得到下述三种推论, 下面分别进行说明。

1) 一个冲突环图至少有两条边。

示意性地, 如果冲突环图只有一条边, 则冲突环图中仅存在两个事务, 且这两个事
务是串行事务, 因此不会构成有向环; 如果有两条边构成有向环, 例如, 冲突环图中的
10 第一条边使得 T_i 在 T_j 之前先发生, 冲突环图中的第二条边使得 T_j 在 T_i 之前先发生, 也
即是不存在等价的可串行化调度, 则这两个事务 T_i 和 T_j 冲突。

2) 一个冲突环图至多有无数条边。

示意性地, 一个调度 s 对应的 $TS(s)$ 中可以存在无数个事务, 相应地, 一个冲突
环图中可以存在无数个顶点, 也即可以存在无数条边。

3) 一个冲突环图如果去掉任何一条边, 则不再构成环。

示意性地, 根据上述冲突环图的第三个条件得知, 在冲突环图构成的有向环中, 存
在 $i \neq j \neq k \neq \dots \neq p$, 也即是任意两条边是不同的, 因此对于一个冲突环图, 若去
掉任何一条边, 则不再构成环。

6、冲突一致性状态图 (Conflict Consistency State Graph, CCSG); 示意性地, 在一
20 个调度 s 中, 存在多个事务 $t \in TS(s)$, 若多个事务 t 构成的图满足如下四个条件, 则称
该图为冲突一致性状态图, 其中, $CCSG(s) = (V, E)$ 。下面对四个条件分别进行说明。

1) CCSG 图中调度 s 的定义满足上述 CCG 图中调度 s 的定义, 即 $CCG(s)$ 和 $CCSG(s)$
之间可以等价转换。

2) 对于 $CCSG(s)$, $V = VS(s)$ 称为 CCSG 图的顶点集, 其元素称为顶点, 一个顶点 v_i
25 满足 $v_i \in VS(s)$, 该顶点由两种类型的数据构成, 一是事务标识 (Identification, ID), 二
是一个列表 (List), 该列表用于存放变量的不同版本。

3) E 为 $V \times V$ 的子集, 也可以称为边集, 该边集中的元素称为有向边, 在 CCSG 图
中, 边分为下述两种, 对于以下两种边, 统一表达为 $\langle V_i, V_j \rangle$, 这样的边源自 CCG 图的
边的等价变形, 故属于冲突操作的变形。下面对两种边分别进行说明。

30 第一种边为一致性状态边, 记为 $\langle x_m, y_n \rangle$, 简称事务边 E_T 。该类型的边包括同一个
事务的操作, 因此该事务内的数据状态满足一致性要求。其中, $x=y$ 且 $m \neq n$; 或者 $x \neq$
 y 。由于 CCSG 图等价于 CCG 图, 故一致性状态边也可表示为 $R_i[x_m]R_i[y_n]$, E_T 可以
作为该边的标签。

35 第二种边为状态变迁边, 记为 $\langle x_i, x_{i+1} \rangle$, 简称状态边 E_S 。该类型的边是由某个写
事务写某个变量, 致使数据状态发生变迁而构成, 因此变量的版本发生变化。由于 CCSG
图等价于 CCG 图, 故一致性状态边也可表示为 $W_j[x_{m+1}]$ 。

4) $\exists V_1 \in V, V_2 \in V \dots, V_k \in V$, 也 $\exists E_1 \in E, E_2 \in E \dots, E_k \in E$, 使得 CCSG 图为一个
有向环构成的图。也即是: $V_1 \rightarrow E_1 \rightarrow V_2 \rightarrow E_2 \dots V_k \rightarrow E_k \rightarrow V_1$ 为一个有向环构成的图。

7、CCG 图向 CCSG 图转换的等价转换规则; 其中, CCG 图向 CCSG 图的等价转
40 换规则包括下述四条, 下面分别进行说明。

第一条为边拆分。其中, CCG 图的任何一条边, 由两个操作即一个冲突构成, 其
可以拆分为两个单独的边。示意性地, 边的拆分方式包括如下三种, 每一种拆分出两个
子边, 下面对三种拆分方式进行说明。

1) 例如, 边表示为 $R_1[x_0]W_2[x_1]$, 对该边进行拆分, 得到 $R_1[x_0]$ 和 $W_2[x_1]$, 同时得到变量 x 的两个版本 x_0 和 x_1 。

2) 例如, 边表示为 $W_2[y_1]R_3[y_1]$, 对该边进行拆分, 得到 $W_2[y_1]$ 和 $R_3[y_1]$, 同时得到变量 y 的一个版本 y_1 。

5 3) 例如, 边表示为 $W_3[z_1]W_1[z_2]$, 对该边进行拆分, 得到 $W_3[z_1]$ 和 $W_1[z_2]$, 同时得到变量 z 的两个版本 z_1 和 z_2 。

第二条为顶点转换。其中, 将变量的版本作为 CCSG 图的顶点, 也即是把上一步的版本转换为顶点; 这里, 以上述第一条举例得到的顶点为例, CCSG 图的顶点包括 x_0 、 x_1 、 y_1 、 z_1 以及 z_2 。同一个变量的多个版本纵向排列, 被同一个事务作用过的不同变量的版本之间横向排列, 其中, 纵向可以是指与屏幕显示信息对应的水平方向, 还可以是指与屏幕显示信息对应的水平方向垂直的方向, 等等, 本申请实施例对此不作限定; 但
10 同一个变量的多个版本的排列方向, 与被同一个事务作用过的不同变量的版本之间的排列方向是相对的。需要说明的是, 此处顶点的排列方式并非特定要求, 在一些实施例中, 还可以按照其他排列方式进行排列, 本申请实施例对此不作限定。

15 第三条为在相邻版本的同一变量对应的顶点之间添加边。其中, 以上述第二条得到的新顶点之间的变迁关系为 CCSG 图的新边。例如, 前述的事务 T_2 的相关子边包括: $W_2[x_1]$ 和 $W_2[y_1]$, 则把 x_1 和 y_1 水平并列, 从而 x_0 和 x_1 之间存在一条 $W_2[x_1]$ 边。

第四条为在同一事务所作用的变量对应的顶点之间添加边。其中, 在同一个事务之间, 构造 CCSG 图的一致性状态边。例如, 继续以前述顶点为例, 在 x_1 和 y_1 之间添加
20 边, 其边的标签为事务 T_2 。

8、CCSG 图向 CCG 图转换的等价转换规则; 其中, CCSG 图向 CCG 图的等价转换规则主要包括下述三条, 下面分别进行说明。

第一条为顶点合并。其中, 对于 CCSG 图, 将同一个事务操作过的变量合并为一个顶点, 并将该顶点命名为该事务的事务标识, 该顶点的表示方式, 等价于前述 CCG 图
25 的顶点表示方式, 这里不再重复描述。

第二条为在相邻顶点之间添加边; 包括以下两点, 第一点是保留变量的状态变迁。例如, 由边 $R_1[x_0]W_2[x_1]$ 可知, 变量的状态变迁由 T_1 指向 T_2 , 因此添加的新边为 $T_1 \rightarrow T_2$ 。第二点是如果没有写操作发生, 则按照原始的 CCG 图中发生的冲突关系构造新边。需要说明的是, 应用于存在 WR 冲突关系的情况, 在存在 WR 冲突关系的情况下,
30 对应的边由 W 所在的事务指向 R 所在的事务。

第三条为得到的 CCG 图存在环, 且可以忽略其有向的属性, 也即是对 CCSG 进行转换得到的 CCG 图为一个无向有环图。

需要说明的是, 在本申请实施例中, CCSG 图即为数据状态一致性模型的基础。也即是, 以变量的版本为顶点, 以事务的读写操作为状态变迁的边或以事务的一致性状态
35 边为边, 构成的图, 即为数据状态一致性模型。在该数据状态一致性模型中, 若存在环则存在数据异常, 如不存在环, 则不存在数据异常。

9、数据异常 (Data Anomalies, DA); 一个调度 s 中, 若存在一个子调度 s' , 也即是 $\exists s' \subseteq s$, 且 s' 存在冲突环图, 则称调度 s 存在数据异常, s' 即为某一个具体的数据异常。

40 另外, 基于上述描述的冲突环图以及数据异常, 得到下述三种推论, 下面分别进行说明。

1) 数据异常有无数个。基于冲突环图中的第二条推论得知, 冲突环图中可以存在

无数个顶点和无数条边,因此,数据异常有无数个。

2) 存在无数个细分类的数据异常。由于冲突环图中的顶点不同则数据异常不同,冲突环图中的边不同则数据异常不同,冲突环图中的顶点和边中的至少一种的个数不同则数据异常不同,因此存在无数个细分类的数据异常。

5 3) 采用形式化表达式对每个细分类的数据异常用表达,并为该细分类的数据异常命名。

应理解,一个数据异常是对应存在冲突环图的调度 s 的子集,因此一个细粒度的数据异常可以由构成冲突环图的该调度 s 的子集部分的表达式来表示。

10 例如,以丢失更新异常为例,该丢失更新异常对应的冲突环图参考图 3,图 3 是本申请实施例提供的另一种冲突环图的示意图。如图 3 所示,在该冲突环图 3-1 中, $V=\{T_1, T_2\}$, $E=\{<R_1[x_0]W_2[x_1]>, <W_2[x_1]W_1[x_2]>\}$, 则该丢失更新异常的形式化定义为 $R_i[x_m]...W_j[x_{m+1}]...W_i[x_{m+2}]$ 。

15 又例如,以读偏序异常为例,该读偏序异常对应的冲突环图参考图 4,图 4 是本申请实施例提供的又一种冲突环图的示意图,如图 4 所示,在该冲突环图 4-1 中, $V=\{T_1, T_2\}$, $E=\{<R_1[x_0]W_2[x_1]>, <W_2[y_1]R_1[y_1]>\}$, 则该读偏序异常的形式化定义为 $R_i[x_m]...W_j[x_{m+1}]...W_i[y_n]...R_i[y_n]$ 。

需要说明的是,此处以上述两种数据异常为例,结合对应的冲突环图,对数据异常的形式化进行了举例说明,其他数据异常对应的冲突环图以及数据异常的形式化定义将在后续进行介绍。

20 10、回滚率;当利用封锁技术来抑制并发并避免数据异常产生时,使得原本可以提交的事务被回滚,也就是假回滚。下面对在本申请实施例所涉及的回滚的信息分别进行说明。

1) 数据异常回滚;对于调度 s ,当由于数据异常出现而导致的事务的回滚,称为数据异常回滚,也称为真回滚(True Rollback, TR)。

25 2) 算法回滚;对于调度 s ,当没有数据异常存在时,通过并发访问算法确定可能发生数据不一致,而致使的事务回滚,称为算法回滚,也称为假回滚(False Rollback, FR)。

3) 真回滚率;假设 P 个调度中,如果 U 个调度中每个调度都存在真回滚,则 U/P 称为真回滚率。真回滚率反应了应用背景下数据异常出现的概率。

30 4) 假回滚率;假设 P 个调度中,如果 V 个调度中每个调度都存在假回滚,则 V/P 称为假回滚率。假回滚率反应了并发事务的并发执行能力,假回滚率越低则并发度越高,且计算资源的利用率越高,事务并发效率越高。

35 以上对本申请实施例涉及的基础概念进行了介绍,下面将对本申请的事务处理方法进行说明。在本申请实施例中,以上述基础概念为理论基础,利用事务与事务之间构成的冲突环图来进行数据异常的识别,能够有效识别出事务在执行过程中出现的数据异常,从而提高事务处理的效率。

参考表 1,表 1 描述了本申请实施例涉及到的一些符号及对应含义。

表 1

符号	含义
CCG	基于冲突的有向环图,也称为冲突环图
s	一个调度
$s \approx s'$	两个调度 s 和 s' 冲突等价
$TS(s)$	调度 s 中事务的集合

$VS(s)$	调度 s 中带有某个版本号标识变量的集合
$O(s)$	一个调度的操作集合
$O_i[x_m] O_j[x_n]$	事务 T_i 和 T_j 作用于同一个变量 x 上的两个操作，称为一个冲突； 其中，一个特例可以表达为 $R_i[x] W_j[x]$
$O_i O_j - O_j O_i$	两个冲突之间组合，构成一个环，是构成数据异常的必要条件， 同时也是数据异常的形式化表达式
x_m	变量 x 的第 m 个版本值
$R_i[x_m]$	事务 T_i 的读操作 R 读取变量 x 的第 m 个版本值
$W_i[x_m]$	事务 T_i 的写操作 W 写入变量 x 的第 m 个版本值
$RW/WR/WW$	三种冲突的简写形式
$R_i[x_m] \dots W_j[x_{m+1}]$ $\dots W_i[y_n] \dots R_i[y_n]$	(读偏序) 数据异常的形式化表达式

基于上述描述的基础概念以及表 1 所示的符号和对应含义，下面参考图 5，对本申请实施例提供的一种事务处理方法进行说明。图 5 是本申请实施例提供的一种事务处理方法流程图，如图 5 所示，本申请实施例提供的事务处理方法应用于数据库系统中任一节点设备，当数据库系统为单机数据库系统时，该节点设备可以是单机设备；当数据库系统为分布式数据库系统时，该节点设备可以是协调节点设备或者数据节点设备；本申请实施例包括步骤 501 和步骤 502，下面对各步骤分别进行说明。

步骤 501、节点设备确定目标事务的并发事务，并发事务与该目标事务之间包括用于同一变量的读写操作，目标事务为待提交事务。

在本申请实施例中，该目标事务是全局事务或局部事务；其中，该全局事务是指涉及到跨节点操作的事务，全局事务也称为分布式事务；而局部事务则是指仅涉及单个节点操作的事务，局部事务也称为本地事务；本申请实施例对于目标事务的类型不进行限定。这里，读写操作是指读操作和写操作中的至少一种。

其中，节点设备在执行目标事务时，通过事务读集和写集，并对至少一个候选并发事务进行筛选，以从筛选出的至少一个候选并发事务中确定出目标事务的并发事务（确定并发事务的具体实施方式将在后续进行介绍）。这里，目标事务可以由终端发起，此时终端与节点设备建立用于处理目标事务的会话，终端向节点设备发送目标事务的执行请求，节点设备响应于目标事务的执行请求，开始执行该目标事务。在一些实施例中，如果终端与节点设备已经建立过一个会话，那么无需在两者之间建立新的会话，只需要复用当前已建立的会话即可。

步骤 502、节点设备获取目标事务的读集和并发事务的写集之间的第一交集，并获取目标事务的写集和并发事务的读集之间的第二交集，当第一交集和第二交集的至少一项为非空数据集，且目标事务与并发事务冲突时，基于目标事务和并发事务所作用的同一变量的版本变更次数和目标变量列表，确定数据异常类型，目标变量列表包括目标事务和并发事务所作用的同一变量。

在本申请实施例中，目标事务与并发事务冲突是指目标事务与并发事务中存在至少两个操作为冲突操作，若此时提交目标事务，将导致数据库系统出现数据异常。节点设备在确定上述第一交集和第二交集中存在至少一项为非空数据集，且目标事务与并发事务冲突时，基于该版本变更次数和该目标变量列表，确定数据异常类型并将数据异常类

型进行上报。其中，非空数据集是指不为空集的数据集，即为包括集合元素的数据集。

在本申请实施例提供的事务处理方法中，节点设备在确定目标事务的并发事务后，在该目标事务的读集和该并发事务的写集之间的第一交集和该目标事务的写集和该并发事务的读集之间的第二交集中，存在至少一项不为空的情况下，对目标事务与并发事务是否冲突进行判断，若目标事务与并发事务冲突，则根据这两个事务所作用的相同变量以及相同变量的版本变更次数，来确定数据异常类型。这种方式能够在目标事务的执行过程中，确认目标事务与并发事务是否真正构成冲突；并且，通过两个事务所作用的相同变量来确定数据异常类型，能够基于可串行化理论，识别出数据库系统中各种各样的数据异常，保证了数据状态的一致性，也提高了数据库系统的事务处理效率。

上述图 5 是对本申请提供的事务处理方法的简要说明，下面结合图 6，对本申请实施例提供的事务处理方法进行详细介绍。图 6 是本申请实施例提供的另一种事务处理方法流程图，如图 6 所示，本申请实施例提供的事务处理方法应用于数据库系统中任一节点设备，当数据库系统为单机数据库系统时，该节点设备可以是单机设备；当数据库系统为分布式数据库系统时，该节点设备可以是协调节点设备或者数据节点设备；本申请实施例包括下述步骤 601 至步骤 609，下面对各步骤分别进行说明。

步骤 601、节点设备响应于目标事务的执行指令，对目标事务的读集和写集进行初始化。

在本申请实施例中，节点设备响应于目标事务的执行指令，将该目标事务的读集和写集初始化为空集，并为目标事务分配全局唯一的事务 ID，也即是事务标识。这里，不同事务的事务 ID 按照节点设备接收到相应执行指令的时间顺序整数递增，例如，节点设备在 t1 时刻接收到目标事务的执行指令，将该目标事务的事务 ID 命名为 1000，若节点设备在 t2 时刻接收到另一事务的执行指令，则节点设备将该另一事务的事务 ID 命名为 1001，本申请实施例对此不作限定。

在本申请实施例中，在该初始化过程中，节点设备在数据库系统启动时，节点设备的操作系统分配一块内存空间，该内存空间用于维护至少一个事务的读集和写集；当目标事务开始执行时，节点设备从该内存空间中申请一块内存，该内存用于管理该目标事务的读集和写集，从而在节点设备上完成了目标事务的读集和写集的创建，并且，将创建出的读集和写集均初始化为空集。

下面对目标事务的读集和写集的更新策略进行介绍。

在完成读集和写集进行初始化后，节点设备根据目标事务的执行语句，确定该目标事务是否涉及对变量的更新操作，当确定涉及对变量的更新操作时，节点设备响应于该目标事务对任一变量进行更新，将该变量添加至该目标事务的写集中，并为该变量分配版本号。例如，当某一事务 T 对变量 x 进行更新时，将变量 x 添加至事务 T 的写集中，并为变量 x 分配版本号。不同事务对同一个变量的写操作的版本号按整数递增；此时，节点设备执行读操作即可获得变量的版本号。即数据库系统为每个变量维护一个递增的版本号。另外，节点设备根据目标事务的执行语句，对任一变量进行读取时，所读取的变量是最新的且满足读已提交规则的变量。

上述更新策略也可以理解为目标事务的读集和写集随着事务操作的进行而实时维护。若发生读操作，则被读取的变量进入读集；若发生写操作，则被读取的变量进入写集。同一个事务写同一个变量多次则有多个不同的新版本，在写集中用最新的变量版本替代旧的变量版本。

需要说明的是，本步骤 601 也即是上述步骤 501 中涉及的事务读集和写集形成的过程，为读写阶段事务读写集合的形成步骤。

步骤 602、节点设备响应于目标事务中每个操作的执行指令，获取目标事务的至少

一个候选并发事务。

在本申请实施例中，对于目标事务的每一个操作（该操作可以是读操作也可以是写操作），在该操作发生时，进入准备阶段，节点设备进入临界区，遍历该目标事务的读集和写集，以及获取目标事务的至少一个候选并发事务（比如所有并发事务）的事务 ID，

5 并将该至少一个候选并发事务的事务 ID 存入到一个链表（Typescript，TS）中。

在本申请实施例中，节点设备在获取至少一个候选并发事务的过程中，内存中的每个变量上都记载曾经读写过该变量但尚未提交的事务的事务 ID（简称为活跃事务 ID），每个变量上记载的活跃事务 ID 可以是一个，也可以是多个，还可以为空。节点设备通过获取读集和写集中各个变量上所记载的各个活跃事务 ID，即可得到该至少一个候选并发事务的事务 ID。

10

步骤 602 也可以理解为节点设备在目标事务的每个操作发生时，开始做准备工作，进入目标事务的准备阶段；其中，准备阶段是为事务 T 是否可以提交做好事务并发访问控制异常验证前的准备工作，节点设备在临界区中找到至少一个候选并发事务的事务 ID 之后，退出临界区。

15

需要说明的是，步骤 602 也即是上述步骤 501 中的准备阶段，用于筛选出至少一个候选并发事务。

步骤 603、节点设备从至少一个候选并发事务中确定目标事务的并发事务，并发事务与目标事务之间包括作用于同一变量的读写操作。

20

在本申请实施例中，节点设备在目标事务的每个操作发生时，获取该目标事务的至少一个候选并发事务；对于至少一个候选并发事务中的任一个候选并发事务，节点设备根据该候选并发事务、以及目标事务的读集和写集，对该候选并发事务是否为目标事务的并发事务进行判断；若该候选并发事务与目标事务之间包括作用于同一变量的读写操作，则将该候选并发事务确定为该目标事务的并发事务，执行步骤 604 至步骤 607；否则，将该候选并发事务重新加入链表 TS 的尾部。

25

其中，节点设备基于可串行化理论，以该至少一个候选并发事务和目标事务之间是否构成冲突环图，来进行一致性检验。下面以链表 TS 中的任一个候选并发事务为例，对本步骤 603 的实施过程进行详细说明；这里，步骤 603 的实施过程包括步骤 6031 至步骤 6035，下面对各步骤分别进行说明。

30

步骤 6031、节点设备获取目标事务的读集和候选并发事务的写集之间的第一候选交集、目标事务的写集和候选并发事务的读集之间的第二候选交集、以及目标事务的写集和候选并发事务的写集之间的第三候选交集。

35

其中，当目标事务为 T，候选并发事务为 TL 时，节点设备将目标事务 T 与该候选并发事务 TL 之间的有向边的数量置为 0，将目标事务与候选并发事务所作用的相同变量的版本变更次数置为 0，将目标变量列表置为空集。其中，该有向边的数量用于表示目标事务与该候选并发事务之间是否冲突；该版本变更次数用于表示目标事务与该候选并发事务所作用的相同变量的数据状态；该目标变量列表用于存储目标事务与该候选并发事务所作用的相同变量。

40

在本申请实施例中，节点设备用动边交叉值（Dynamic Line Intersection，DLI）来表示有向边的数量，用变量状态值（Variable State，VS）来表示版本变更次数，用变量列表（Variable List，VL）来表示目标变量列表，也即令 $DLI=0$ ； $VS=0$ ； $VL=空集$ 。

在本申请实施例中，第一候选交集表示为“ $S1=DSR(T) \cap DSW(TL)$ ”；第二候选交集表示为“ $S2=DSR(TL) \cap DSW(T)$ ”；第三候选交集表示为“ $S3=DSW(T) \cap DSW(TL)$ ”。

在本申请实施例中，节点设备在获取第一候选交集、第二候选交集以及第三候选交集时，可以使用哈希表存储目标事务和候选并发事务各自的读写集合，从而能够在线性

时间复杂度内得到对应的交集。

步骤 6032、若第一候选交集、第二候选交集以及第三候选交集之间的并集为空集，则节点设备将候选并发事务重新加入链表 TS 的尾部；若第一候选交集、第二候选交集以及第三候选交集中存在至少一项不是空集，则节点设备执行下述步骤 6033 至步骤 6035。

其中，若第一候选交集、第二候选交集以及第三候选交集之间的并集为空集（即第一交集、第二交集以及第三交集均为空集），说明该候选并发事务并不是目标事务的并发事务。若第一候选交集、第二候选交集以及第三候选交集中存在至少一项不是空集，说明该候选并发事务为目标事务的并发事务，则节点设备将该候选并发事务确定为目标事务的并发事务，也即是本步骤 603 所涉及的并发事务，执行下述步骤 6033 至步骤 6035；另外，当节点设备确定候选并发事务为目标事务的并发事务时，第一候选交集即为第一交集，第二候选交集即为第二交集，第三候选交集即为第三交集。

下面以该候选并发事务为确定好的并发事务为例，对步骤 6033 至步骤 6035 进行说明。

步骤 6033、若该第一交集不为空集，节点设备对有向边的数量进行增量处理，并将第一交集集中的变量添加到目标变量列表中；若第一交集中存在版本不同的变量，节点设备对版本变更次数进行增量处理。

其中，增量处理是指将对应的值自增一。示意性地，以有向边的数量为 DLI，版本变更次数为 VS 为例，上述两种增量处理分别表示为 $DLI=DLI++$ 和 $VS=VS++$ 。另外，节点设备将该第一交集集中的变量添加到该目标变量列表中是指按照预设顺序，将变量有序添加到该目标变量列表中。

步骤 6034、若第二交集不为空，节点设备对有向边的数量进行增量处理，将第二交集集中的变量添加到目标变量列表中；若第二交集中存在版本不同的变量，节点设备对版本变更次数进行增量处理。

其中，步骤 6034 与上述步骤 6033 在实现过程上类似，故在此不再重复描述。

步骤 6035、若第三交集不为空集，节点设备基于并发事务的提交状态，确定数据异常类型；若第三交集为空集，节点设备执行后续步骤 604。

其中，若第三交集不为空集，说明目标事务与并发事务之间出现了写写冲突，而写写冲突是数据一致性状态模型下导致数据错误的原因之一，也即为禁止脏写数据异常；因此，在这种情况下，节点设备基于该并发事务的提交状态，确定数据异常类型，并报告写写异常发生，结束当前流程，对目标事务进行回滚。而当第三交集为空集时，表明写写冲突未发生，从而节点设备执行后续步骤 604。

在一些实施例中，若该并发事务未提交且该目标事务的目标参数 T.no_committed 为一，则节点设备确定数据异常类型为脏写异常；其中，该目标参数 T.no_committed 用于表示该目标事务的读写集合所对应的已提交事务数量；由于在循环过程中，如果上一个并发事务与目标事务之间不存在数据异常，那么在执行本次循环之前，需要将上一个并发事务与目标事务进行合并，并将合并后得到的新的事务确定为本次循环的目标事务，而合并可通过对两事务的读写集合进行合并实现；因此在任一次循环中，目标事务的读写集合有可能是经过合并后的读写集合，在目标事务的读写集合中包括已经提交的事务的读写集合的成分的情况下，目标参数 T.no_committed 用于描述读写集合中对应的已提交事务数量。

在一些实施例中，若该并发事务已提交且该第一交集与该第三交集之间的交集不是空集，确定数据异常类型为丢失更新异常。

示意性地，以第一交集、第二交集以及第三交集分别表示为 S1、S2 以及 S3 为例，

若 $S3 \neq \emptyset$, 当满足如下任一条件时, 节点设备确定数据异常类型, 报告写写异常发生, 循环终止; 其中, 待满足的条件包括: 1) 该并发事务没有提交完成且该目标事务的目标参数 $T.no_committed=1$, 此时构成写写冲突, 数据异常类型为脏写异常; 2) 该并发事务已经提交, 且 $S1 \cap S3$ 不是空集, 此时数据异常类型为丢失更新异常。

5 需要说明的是, 节点设备可以同步执行上述步骤 6031 至步骤 6035, 还可以按照任意顺序执行上述步骤 6031 至步骤 6035, 本申请实施例对此不作限定。

经过上述步骤 603, 节点设备从至少一个候选并发事务中确定出目标事务的并发事务, 若目标事务与并发事务之间的第三交集为空集, 则节点设备执行下述步骤 604。

10 步骤 604、若第一交集和第二交集中存在至少一项不为空集, 且目标事务与并发事务冲突, 则节点设备基于目标事务和并发事务所作用的不同变量的版本变更次数和目标变量列表, 确定数据异常类型, 目标变量列表包括目标事务和并发事务所作用的不同变量。

其中, 该第一交集和该第二交集中, 存在至少一项不为空集包括以下三种情况: 1) 第一交集不为空集, 第二交集为空集; 2) 第一交集为空集, 第二交集不为空集; 3) 第一交集不为空集, 第二交集不为空集。若第一交集和第二交集满足上述三种情况中的任一种, 且该目标事务与该并发事务冲突, 则节点设备基于版本变更次数和目标变量列表, 来确定数据异常类型。其中, 节点设备基于目标事务与并发事务之间的有向边的数量, 来确定目标事务与并发事务是否冲突, 也即是, 若目标事务与该并发事务之间的有向边的数量大于或等于二, 则确定目标事务与并发事务冲突。

20 下面将对步骤 604 中确定数据异常类型的过程进行说明。首先通过以下四点, 对本申请实施例所涉及的数据异常类型的所属类别、子类别、以及隔离级别分别进行介绍。

第一点、数据异常类型的所属类别。

25 在本申请实施例中, 数据异常类型的所属类别包括写异常类别 (Write Anomaly Type, WAT)、读异常类别 (Read Anomaly Type, RAT) 以及交叉异常类别 (Intersect Anomaly Type, IAT)。

其中, 若一个调度中存在作用于不同版本的同一变量的至少两个 (称为第一指定数量) 写操作 (称为满足写异常条件), 则该数据异常类型的所属类别为写异常类别; 若一个调度中不存在作用于不同版本的同一变量的至少两个写操作 (称为不满足写异常条件), 且该调度中存在作用于相同版本的同一变量的读操作和写操作 (称为满足读异常条件), 则该数据异常类型的所属类别为读异常类别; 若一个调度中不存在作用于不同版本的同一变量的至少两个写操作 (称为不满足写异常条件), 且该调度中不存在作用于相同版本的同一变量的读操作和写操作 (称为不满足读异常条件), 且该调度中存在作用于不同版本的同一变量的读操作和写操作 (称为满足交叉异常条件), 则该数据异常类型的所属类别为交叉异常类别。

35 下面结合上述表 1 所示的符号及对应含义, 当一个调度 s 对应的 CCG 图中 $\exists T_i, T_j \in TS(s)$, 且 $i \neq j$, 则数据异常类型的所属类别可以表示为下述三种形式, 下面分别进行说明。

1、写异常类别。

40 若 $\exists \langle T_i, T_j \rangle \in E = W_i[x_m]W_j[x_{m+1}]$, 则该调度 s 存在数据异常, 且该数据异常类型的所属类别为写异常类别。

2、读异常类别。

若 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]W_j[x_{m+1}]$, 且 $\exists \langle T_i, T_j \rangle \in E = W_i[x_m]R_j[x_m]$, 则该调度 s 存在数据异常, 且该数据异常类型的所属类别为读异常类别。

3、交叉异常类别。

若 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]W_j[x_{m+1}]$ ，且 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]R_j[x_m]$ ，
且 $\exists \langle T_i, T_j \rangle \in E = R_i[x_m]W_j[x_{m+1}]$ ，则该调度 s 存在数据异常，且该数据异常类
型的所属类别为交叉异常类别。

5 第二点、数据异常类型的子类别。

在本申请实施例中，第一点描述了根据冲突的差异，将数据异常类型分为写异常类
别、读异常类别以及交叉异常类别三类；这里，为了提升数据异常识别的精细度，继续
对数据异常的类型进行划分，得到子类别。

其中，若一个数据异常发生同一个变量上，则数据异常类型在所属类别中的子类别
10 属于单元数据异常（Single-Meta Data Anomalies, SDA）；若一个数据异常发生两（称为
第三指定数量）个变量上，则数据异常类型在所属类别中的子类别属于二元数据异常
（Double-Meta Data Anomalies, DDA）；若一个数据异常不属于上述两种子类别，则数
据异常类型在所属类别中的子类别属于多元数据异常（Multi-Meta Data Anomalies,
MDA）。

15 示意性地，假设一个数据异常（Data Anomalies, DA，简称为 da）中变量的个数表
示为 NV，事务的个数表示为 NT，将 NV 和 NT 的组合称为数据异常的特征，记为
Cda=(NV, NT)，则上述数据异常类型的子类别表示为下述三种情况，下面分别进行说
明。

1、单元数据异常。

20 若 Cda=(1, 2)，则数据异常发生在两个事务一个变量上，则数据异常类型在所属类
别中的子类别属于单元数据异常。

2、二元数据异常。

若 Cda=(2, 2)，则数据异常发生在两个事务两个变量上，则数据异常类型在所属类
别中的子类别属于二元数据异常。

25 3、多元数据异常。

若 Cda $\neq$ (1, 2)，且 Cda $\neq$ (2, 2)，则数据异常类型在所属类别中的子类别属于多元
数据异常。

第三点、数据异常类型的细分类。

30 在本申请实施例中，按照上述第一点（数据异常类型的所属类别）和第二点（数据
异常类型的子类别）所示的分类，给出了每种细分类的数据异常的所属类别、名称、对
应的形式化表示、对应的 CCG 图以及对应的子类别。具体如表 2 以及图 7 至图 9 所示；
其中，图 7 为本申请实施例提供的一种单元数据异常 CCG 图的示意图；图 8 为本申请
实施例提供的一种二元数据异常 CCG 图的示意图；图 9 为本申请实施例提供的一种多
元数据异常 CCG 图的示意图。

35 表 2

所属类别	数据异常名称	形式化表示	CCG	子类别
WAT	丢失更新异常	$R_i[x_m] \dots W_j[x_{m+1}] \dots W_i[x_{m+2}]$	图 7 中 (b)	SDA
	丢失自更新异常	$W_i[x_m] \dots W_j[x_{m+1}] \dots R_i[x_{m+1}]$	图 7 中 (c)	SDA
	全写入异	$W_i[x_m] \dots W_j[x_{m+1}] \dots W_i[x_{m+2}]$	图 7 中 (d)	SDA

	常			
	读写偏序异常 1	$R_i[x_m] \dots W_j[x_{m+1}] \dots W_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (f)	DDA
	读写偏序异常 2	$W_i[x_m] \dots W_j[x_{m+1}] \dots R_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (g)	DDA
	重复写偏序异常 1	$W_i[x_m] \dots W_j[x_{m+1}] \dots W_j[y_n] \dots R_i[y_n]$	图 8 中 (h)	DDA
	重复写偏序异常 2	$W_i[x_m] \dots R_j[x_m] \dots W_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (i)	DDA
	全写入偏序异常	$W_i[x_m] \dots W_j[x_{m+1}] \dots W_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (c)	DDA
	台阶式写异常	$\dots W_i[x_m] \dots W_j[x_{m+1}] \dots$	图 9 中 (a)	MDA
RAT	不可重复读异常	$R_i[x_m] \dots W_j[x_{m+1}] \dots R_i[x_{m+1}]$	图 7 中 (a)	SDA
	中间读异常	$W_i[x_m] \dots R_j[x_m] \dots W_i[x_{m+1}]$	图 7 中 (e)	SDA
	读偏序异常 1	$R_i[x_m] \dots W_j[x_{m+1}] \dots W_j[y_n] \dots R_i[y_n]$	图 8 中 (d)	DDA
	读偏序异常 2	$W_i[x_m] \dots R_j[x_m] \dots R_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (e)	DDA
	写读偏序异常	$W_i[x_m] \dots R_j[x_m] \dots W_j[y_n] \dots W_i[y_n]$	图 8 中 (b)	DDA
	台阶式读异常	$\dots W_i[x_m] \dots R_j[x_m] \dots$ 且不包括 $\dots W_{ii}[x_m] \dots W_{jj}[x_{m+1}] \dots$	图 9 中 (b)	MDA
IAT	写偏序异常	$R_i[x_m] \dots W_j[x_{m+1}] \dots R_j[y_n] \dots W_i[y_{n+1}]$	图 8 中 (a)	DDA
	台阶式交叉异常	$\dots R_i[x_m] \dots W_j[x_{m+1}] \dots$ 且不包括 $\dots W_{ii}[x_m] \dots W_{jj}[x_m] \dots$ 和 $\dots W_{iii}[x_m] \dots W_{jjj}[x_{m+1}] \dots$	图 9 中 (c)	MDA

需要说明的是, 图 7 至图 9 所示的 CCG 图的示意图的相关描述, 参见图 2 至图 4 对应的描述文段, 本申请实施例在此不再重复描述。

需要说明的是, 相比于单元数据异常和双元数据异常, 多元数据异常的发生概率较小。下面对图 9 所示的多元数据异常 CCG 图对应的调度表达式进行示意性说明。

5 如图 9 中 (a) 图所示, 冲突环图 9-1 中为台阶式写异常, 台阶式写异常的调度表达

式为 $s = R_1[x_0] W_2[x_1] W_2[y_1] W_3[y_2] R_3[z_0] W_1[z_1]$ ，该调度中包含了 WW 冲突。

如图 9 中 (b) 图所示，冲突环图 9-2 中为台阶式读异常，台阶式读异常的调度表达式为 $s = R_1[x_0] W_2[x_1] W_2[y_1] R_3[y_1] R_3[z_0] W_1[z_1]$ ，该调度中包含了 WR 冲突，但没有包含 WW 冲突。

如图 9 中 (c) 图所示，冲突环图 9-3 中为台阶式交叉异常，台阶式交叉异常的调度表达式为 $s = R_1[x_0] W_2[x_1] R_2[y_0] W_3[y_1] R_3[z_0] W_1[z_1]$ ，该调度中包含了 RW 冲突，但没有包含 WW 冲突和 WR 冲突。

第四点、数据异常类型的隔离级别。

在本申请实施例中，按照上述第三点所示的数据异常的细分类，提供了两种隔离级别，分别为无读写异常 (No Read-Write Anomalies, NRW) 和无任何异常 (No Anomalies, NA)；其中，隔离级别的分级依据是数据异常出现的允许程度，通过隔离级别，能够降低并发访问控制处理的复杂度。其中，NRW 是指不允许出现 WAT 和 RAT 类数据异常，但不能消除 IAT 类数据异常；NA 是指不允许任何数据异常发生，等价于 ANSI-SQL 标准定义的可串行化隔离级别。因此，NRW 级别低于 NA 级别。

下面参考表 3，描述了本申请实施例提供的隔离级别。

表 3

类别	数据异常名称	NRW	NA
WAT	丢失更新异常	不允许	不允许
	丢失自更新异常	不允许	不允许
	全写入异常	不允许	不允许
	读写偏序异常 1	不允许	不允许
	读写偏序异常 2	不允许	不允许
	重复写偏序异常 1	不允许	不允许
	重复写偏序异常 2	不允许	不允许
	全写入偏序异常	不允许	不允许
	台阶式写异常	不允许	不允许
RAT	不可重复读异常	不允许	不允许
	中间读异常	不允许	不允许
	读偏序异常 1	不允许	不允许
	读偏序异常 2	不允许	不允许
	写读偏序异常	不允许	不允许
	台阶式读异常	不允许	不允许
IAT	写偏序异常	允许	不允许
	台阶式交叉异常	允许	不允许

需要说明的是，节点设备基于表 3 中描述的隔离级别的相关信息，确定并发访问控

制处理。

在本申请实施例中，按照表 3 中描述的隔离级别，节点设备能够通过设置规则避免出现数据异常，达到 NRW 级别。例如，采用禁止 WW 边出现的规则来避免出现 WAT 异常；又例如，采用读已提交，快照读等方式来避免出现 RAT 异常。如此，能够减少
5 隔离级别的层级个数，提高并发访问控制处理的效率。例如，采用可串行化快照隔离 (Serializable Snapshot Isolation, SSI) 技术，通过识别连续的 2 个 RW 边是否出现异常，即能达到 NA 级别，因此能够提高并发访问控制处理的效率。

经过上述四点，对本申请实施例所涉及的数据异常类型的所属类别、子类别、以及隔离级别进行了描述，下面以上述四点所描述的信息为基础，对本步骤 604 的实现过程
10 进行详细说明；其中，步骤 604 包括步骤 6041 至步骤 6043，下面对各步骤分别进行说明。

步骤 6041、节点设备基于目标事务与并发事务中作用于同一变量的读写操作，确定所属类别。

其中，所属类别是指数据异常类型所属的类别；步骤 6041 包括下述三种情况，下面
15 对该三种情况分别进行说明。

情况一、若目标事务与该并发事务中存在作用于不同版本的同一变量的第一指定数量（比如为两个）的写操作，则节点设备确定所属类别为写异常类别。

示意性地，当该目标事务为 T_i ，该并发事务为 T_j ，若 $\exists \langle T_i, T_j \rangle \in E = W_i[x_m]W_j[x_{m+1}]$ ，则节点设备确定所属类别为写异常类别。

情况二、若该目标事务与该并发事务中不存在作用于不同版本的同一变量的两个写操作，且该目标事务与该并发事务中存在作用于相同版本的同一变量的读操作和写操作，
20 则节点设备确定所属类别为读异常类别。

示意性地，假设该目标事务为 T_i ，该并发事务为 T_j ，若 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]W_j[x_{m+1}]$ ，且 $\exists \langle T_i, T_j \rangle \in E = W_i[x_m]R_j[x_m]$ ，则该

25 节点设备确定所属类别为读异常类别。

情况三、若该目标事务与该并发事务中不存在作用于不同版本的同一变量的两个写操作，且该目标事务与该并发事务中不存在作用于相同版本的同一变量的读操作和写操作，
且该目标事务与该并发事务中存在作用于不同版本的同一变量的读操作和写操作，
则节点设备确定所属类别为交叉异常类别。

示意性地，假设该目标事务为 T_i ，该并发事务为 T_j ，若 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]W_j[x_{m+1}]$ ，且 $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m]R_j[x_m]$ ，且 $\exists \langle T_i, T_j \rangle \in E = R_i[x_m]W_j[x_{m+1}]$ ，则节点设备确定所属类别为交叉异常类别。
30

步骤 6042、节点设备基于数据异常类型的所属类别以及目标变量列表，确定在所属类别中的子类别。

其中，由上述步骤 603 中可知，目标变量列表中存储有该目标事务与该并发事务所
35 作用的相同变量，也即是，节点设备基于目标变量列表，获取该目标事务与该并发事务所作用的相同变量的数量，从而结合上述三种子类别的描述，确定在所属类别中的子类别。

示意性地，本步骤 6042 包括下述三种情况，下面对该三种情况分别进行说明。

情况一、若该目标变量列表中的变量数量为第二指定数量（比如为一），则节点设备
40 确定该在所属类别中的子类别属于单元数据异常。

情况二、若该目标变量列表中的变量数量为第三指定数量（比如为二），则节点设备
确定该在所属类别中的子类别属于二元数据异常。

情况三、若该目标变量列表中的变量数量为第四指定数量，其中，第四指定数量与第二指定数量和第三指定数量均不同；也即是，该在所属类别中的子类别既不属于该单元数据异常，也不属于该双元数据异常；则节点设备确定该在所属类别中的子类别属于多元数据异常。

5 步骤 6043、节点设备基于在所属类别中的子类别、版本变更次数以及并发事务与目标事务中作用于同一变量的读写操作，确定数据异常类型。

其中，由上述步骤 603 可知，版本变更次数用于表示该目标事务与该并发事务所作用的相同变量的数据状态；另外，又由上述表 2 可知，不同的数据异常对应的版本变更次数以及两个事务作用于同一变量的读写操作不同；因此，在本步骤 6043 中，节点设备基于版本变更次数以及该并发事务与该目标事务中作用于同一变量的读写操作，来确

10 定该数据异常类型。

示意性地，下面以八种场景为例，对本步骤 6043 的实施过程分别进行说明。

场景一、若所属类别为写异常类别，该子类别为单元数据异常，则本步骤 6043 包括下述三种情况，下面对该三种情况分别进行说明。

15 1) 若该版本变更次数为一，则节点设备确定该数据异常类型为丢失自更新异常。

2) 若该版本变更次数为二，该目标事务与该并发事务中存在作用于同一变量的第五指定数量（比如三个）的写操作，则节点设备确定该数据异常类型为全写入异常。

3) 若该版本变更次数为二，该目标事务与该并发事务中存在作用于同一变量的两个写操作和第六指定数量（比如一个）的读操作，则节点设备确定该数据异常类型为丢失更新异常。

20

场景二、若该数据异常类型的所属类别为写异常类别，该子类别为双元数据异常，则本步骤 6043 包括下述三种情况，下面对该三种情况分别进行说明。

1) 若该版本变更次数为一，该目标事务与该并发事务中存在作用于不同版本的第一变量的写操作和读操作（称为满足第一条件）、以及存在作用于不同版本的第二变量的两个写操作（称为满足第二条件），则节点设备确定该数据异常类型为读写偏序异常；其中，满足第一条件是指目标读写操作包括作用于不同版本的第一变量的写操作和读操作，满足第二条件是指目标读写操作包括作用于不同版本的第二变量的第一指定数量的写操作。

25

2) 若该版本变更次数为一，该目标事务与该并发事务中存在作用于相同版本的第一变量的写操作和读操作（称为满足第三条件）、以及存在作用于不同版本的第二变量的两个写操作（称为满足第二条件），则节点设备确定该数据异常类型为重复写偏序异常；其中，满足第三条件是指目标读写操作包括作用于同一版本的第一变量的写操作和读操作。

30

3) 若该版本变更次数为一，该目标事务与该并发事务中存在作用于不同版本的第一变量的两个写操作（称为满足第四条件）、以及存在作用于不同版本的第二变量的两个写操作（称为满足第二条件），则节点设备确定该数据异常类型为全写入偏序异常；其中，满足第四条件是指目标读写操作包括作用于不同版本的第一变量的第一指定数量的写操作。

35

场景三、若该数据异常类型的所属类别为读异常类别，该子类别为单元数据异常，则本步骤 6043 包括下述两种情况，下面对该两种情况分别进行说明。

40

1) 若该版本变更次数为一，该目标事务与该并发事务中存在作用于同一变量的两个读操作和一个写操作，则节点设备确定该数据异常类型为不可重复读异常。

2) 若该版本变更次数为一，该目标事务与该并发事务中存在作用于同一变量的两个写操作和一个读操作，则节点设备确定该数据异常类型为中间读异常。

场景四、若该数据异常类型的所属类别为读异常类别，该子类别为双元数据异常，则本步骤 6043 包括下述两种情况，下面对该两种情况分别进行说明。

1) 若该版本变更次数为第七指定数量（比如为零），则节点设备确定该数据异常类型为写读偏序异常。

5 2) 若该版本变更次数为一，则节点设备确定该数据异常类型为读偏序异常。

场景五、若该数据异常类型的所属类别为交叉异常类别，该子类别为双元数据异常，则本步骤 6043 包括：若该版本变更次数为一，则节点设备确定该数据异常类型为写偏序异常。

10 场景六、若该数据异常类型的所属类别为写异常类别，该子类别为多元数据异常，则本步骤 6043 包括：节点设备确定该数据异常类型为台阶式写异常。

场景七、若该数据异常类型的所属类别为读异常类别，该子类别为多元数据异常，则本步骤 6043 包括：节点设备确定该数据异常类型为台阶式读异常。

场景八、若该数据异常类型的所属类别为交叉异常类别，该子类别为多元数据异常，则本步骤 6043 包括：节点设备确定该数据异常类型为台阶式交叉异常。

15 需要说明的是，节点设备是按照步骤 6041 至步骤 6043 的顺序执行的，也即是，节点设备先确定数据异常类型的所属类别，再确定子类别，最后确定细分类的数据异常类型。在一些实施例中，上述步骤 6041 至步骤 6043 替换为下述步骤一至步骤三。

步骤一、节点设备基于该目标变量列表，确定子类别。

20 步骤二、节点设备基于该目标事务与该并发事务中作用于同一变量的读写操作，确定所属类别；

步骤三、节点设备基于在所属类别中的子类别、该版本变更次数以及该并发事务与该目标事务中作用于同一变量的读写操作，确定该数据异常类型。

也即是，节点设备先确定数据异常类型的子类别，再确定所属类别，最终确定细分类的数据异常类型。

25 在一些实施例中，节点设备还可以同步确定数据异常类型的所属类别和子类别，本申请实施例对此不作限定。

经过上述步骤 604，节点设备确定数据异常类型后，执行下述步骤 605 至步骤 607。需要说明的是，若该第一交集和该第二交集中，存在至少一项不为空集，且该目标事务与该并发事务不冲突，则节点设备执行下述步骤 608。

30 步骤 605、节点设备从目标事务和并发事务中，确定待回滚的事务。

在本申请实施例中，经过上述步骤 604，节点设备确定出现数据异常，需要对该目标事务或该并发事务进行回滚，从这两个事务中确定出待回滚的事务。

下面对本步骤 605 所涉及的两种可选实施方式进行说明。

35 方式一、节点设备按照事务优先级，将该目标事务和该并发事务中事务优先级最低的事务确定为该待回滚的事务。

在本申请实施例中，事务优先级为预先设置的优先级，数据库系统预先为每类事务分配事务优先级，例如，按照事务的处理时长、事务的操作类型和所作用的变量数量等中的至少一种，确定每个事务的事务优先级，本申请实施例对于事务优先级的设置方式不作限定。

40 示意性地，下面以四种事务优先级的设置方式为例，对本步骤 605 的实施方式进行说明。

第一种、若该目标事务和该并发事务中的当前事务的处理时长小于其他事务的处理时长，则节点设备将该当前事务确定为事务优先级最低的事务；从而，该待回滚的事务即为该当前事务。

其中，事务优先级是按照事务的处理时长来确定的。处理时长大于参考时长的事务为长事务，处理时长小于或等于参考时长的事务确定为短事务，长事务的事务优先级高于短事务的事务优先级。这里，当前事务为目标事务和并发事务中的任一事务，其他事务为目标事务和并发事务中除当前事务之外的事务。

5 示意性地，对于目标事务来说，以并发事务的处理时长为参考时长，若目标事务的处理时长小于该并发事务，则节点设备将该目标事务确定为待回滚的事务；否则，将并发事务确定为待回滚的事务。

第二种、若该目标事务和该并发事务中任一事务为读事务，则节点设备将该读事务确定为事务优先级最低的事务；从而，待回滚的事务即为该读事务。

10 其中，事务优先级是按照事务的操作类型来确定的，写事务的事务优先级高于读事务的事务优先级；这里，事务的操作类型包括写事务类型和读事务类型，写事务类型的事务为写事务，读类型的事务为读事务。

示意性地，若目标事务为写事务，并发事务为读事务，则节点设备将该目标事务确定为待回滚的事务。

15 第三种、若该目标事务和该并发事务中的当前事务为写事务，且该写事务的处理时长小于参考时长，则节点设备将该写事务确定为事务优先级最低的事务；从而，待回滚的事务即为该当前事务。

其中，事务优先级是结合事务的处理时长以及操作类型来确定的，也即是，当写事务为短事务时，其事务优先级低于长的读事务。示意性地，若该目标事务为写事务，且
20 该目标事务的处理时长小于参考时长，则节点设备将该目标事务确定为待回滚的事务。

需要说明的是，在一些实施例中，事务优先级的设置方式还可以有其他形式，例如，短事务的事务优先级高于长事务的事务优先级，又例如，读事务的事务优先级高于写事务的事务优先级，等等，本申请实施例对此不作限定。

可以理解的是，通过这种基于事务优先级来确定待回滚的事务的方式，能够有针对性地回滚事务，保证重要的事务优先处理，从而提高了计算资源的利用率，提高了事务
25 处理的效率。

第四种、若该目标事务与该并发事务中的当前事务所作用的变量数量大于其他事务所作用的变量数量，则节点设备将该当前事务确定为事务优先级最低的事务；从而，待回滚的事务即为该当前事务。

30 在本申请实施例中，节点设备遍历目标事务与并发事务的目标冲突环图（有关目标冲突环图的介绍会在后文步骤 607 中介绍，在此不再重复描述），在当前事务所作用的变量数量最多时，将该当前事务确定为待回滚的事务。

通过这种可选地实施方式，将涉及变量数量较多的事务进行回滚，能够减少后续事务处理过程中与其他事务冲突的可能性，从而减少了事务处理的冲突率，提高了事务处
35 理的效率。

方式二，节点设备将从目标事务与并发事务中随机选择的事务，确定为待回滚的事务。

步骤 606、节点设备对该待回滚的事务进行回滚。

需要说明的是，通过上述步骤 601 至步骤 606，节点设备确定了目标事务与并发事
40 务是否构成冲突，只有在存在数据异常的情况下，才确定待回滚的事务。

可以理解的是，节点设备在确定待回滚的事务时是按照一定的策略来确定的，而不是直接将目标事务进行回滚，这样能够有效降低假回滚率，从而提高计算资源的利用率，提高事务处理的效率。

步骤 607、节点设备删除目标冲突环图中待回滚的事务所作用的变量以及对应的读

写操作,其中,目标冲突环图中的顶点用于表示事务对应的事务标识以及事务对应的读写操作所作用的变量,目标冲突环图中的边用于表示事务之间的关联关系以及变量之间的版本变化。

在本申请实施例中,目标冲突环图是一种细节冲突环图(Detail Conflict Cycle Graph, DCCG),是等价于CCG图和CCSG图的一种冲突环图,也是本申请实施例中数据状态一致性模型的实现基础。在经过上述步骤606,将待回滚的事务进行回滚后,节点设备删除目标冲突环图中待回滚的事务所作用的变量以及对应的读写操作,来维护目标冲突环图的结构。

下面将对目标冲突环图的获取方式进行详细阐述,包括如下步骤一至步骤三,下面对各步骤分别进行说明。

步骤一、获取该目标事务与该并发事务之间的第一冲突环图,该第一冲突环图中的顶点用于表示事务对应的事务标识,该第一冲突环图中的边用于表示事务之间的关联关系以及事务对应的读写操作所作用的变量。

其中,第一冲突环图为该目标事务与该并发事务之间的CCG图。关于第一冲突环图中的顶点以及边的含义请参考前述对CCG图的顶点和边的介绍,此处不再重复描述。

步骤二、对该第一冲突环图进行转换,得到第二冲突环图,该第二冲突环图中的顶点用于表示多个不同版本的变量,该第二冲突环图中的边用于表示变量之间的版本变化以及同一事务所作用的变量。

其中,第二冲突环图为该目标事务与该并发事务之间的CCSG图。关于第二冲突环图中的顶点以及边的含义请参考前述对CCSG图的顶点和边的介绍,此处不再重复描述。基于前述对CCG图和CCSG图的介绍可知,CCG图和CCSG图等价,下面以CCG图向CCSG图的等价转换规则为基础,对本步骤二的实施方式进行简要说明,包括下述步骤A和步骤B,下面对各步骤分别进行说明。

步骤A、对该第一冲突环图的边进行拆分,得到该多个不同版本的变量。

步骤B、将该多个不同版本的变量作为顶点,并在相邻版本的同一变量对应的顶点之间添加边,以及在同一事务所作用的变量对应的顶点之间添加边,得到该第二冲突环图。

步骤三、对该第二冲突环图进行转换,得到该目标冲突环图。

其中,基于前述对CCG图和CCSG图的介绍可知,CCG图和CCSG图等价,CCSG图能够等价转换为CCG图,该目标冲突环图是通过将CCSG图进行转换得到的新的CCG图。在上述第一冲突环图中,该第一冲突环图中的顶点用于表示事务对应的事务标识,而在目标冲突环图(也即是新的CCG图)中,该目标冲突环图的顶点信息相比于第一冲突环图中的顶点信息更为细化,既包括了事务对应的事务标识,也包括了事务对应的读写操作所作用的变量;另外,该目标冲突环图的顶点对应的变量保留了构成冲突环的必要变量,因为一个事务所操作过的变量可能很多,但不是所有的变量及其版本都是构成冲突环图的充分条件。

下面以前述CCSG图向CCG图的等价转换规则为基础,对本步骤三的具体实施方式进行说明,包括下述步骤C至步骤E,下面对各步骤分别进行说明。

步骤C、将该第二冲突环图中同一事务所作用的变量对应的顶点进行合并,并将同一事务对应的事务标识确定为合并后的顶点。即节点设备将合并后的顶点命名为同一事务对应的事务标识。

步骤D、在合并后的第二冲突环图的相邻顶点之间添加边,得到第三冲突环图。

步骤E、若该第三冲突环图的任意两个相邻顶点中存在相同版本的变量,将该任意两个相邻顶点合并,并删去合并后的顶点所对应的该相同版本的变量,得到该目标冲突

环图；若该第三冲突环图的任意两个相邻顶点中存在不同版本的同一变量，将该任意两个相邻顶点合并，得到该目标冲突环图。

其中，该第三冲突环图即为对CCSG图按照等价转换规则进行转换得到的CCG图，在该第三冲突环图中，若任意两个相邻顶点中存在相同版本的变量，则该第三冲突环图适用于约简规则，从而节点设备对这两个顶点进行合并，且删去相同版本的变量；若任意两个相邻顶点中存在不同版本的同一变量，则该第三冲突环图适用于合并规则但不适用于约简规则，从而节点设备对这两个顶点进行合并；若任意两个相邻顶点中不存在相同的变量，则节点设备直接将该第三冲突环图作为目标冲突环图。

下面对上述约简规则以及合并规则进行介绍。

第一、约简规则。

对于一个冲突环图中任意两个相邻顶点 V_i 和 V_{i+1} ，若 $V_i \cap V_{i+1} = \{x_m, y_n, \dots\} \neq \emptyset$ ，即存在至少一个相同版本的变量，则 V_i 和 V_{i+1} 可以合并为一个顶点；该合并出的新顶点中， x_m 和 y_n 等可以被去除，保留两个顶点中不相交的部分，也即是 $V_i \cup V_{i+1} - V_i \cap V_{i+1}$ 。即约简后的新顶点保留 V_i 和 V_{i+1} 这两个集合的差集。

基于上述对目标冲突环图的介绍以及该约简规则，能够得到以下三条推论，下面对各推论分别进行说明。

推论 1、若一个调度 s ， $\exists \text{dccg} \in \text{DCCG}$ ，该 dccg 图的任意两个相邻顶点的交集为空集，则该 dccg 称为最小 DCCG 图，也即是该 dccg 图中满足约简规则的顶点完成约简后，得到的 DCCG 图最精简。

推论 2、一个最小 DCCG 图等价于上述表 2 中的某一种数据异常。

推论 3、若一个调度 s 中，任意两个并发事务之间存在 WR 关系，则这两个并发事务可适用约简规则。

第二、合并规则。

对于一个冲突环图中任意两个相邻顶点 V_i 和 V_{i+1} ，若 $V_i \cap V_{i+1} = \emptyset$ ，则这两个顶点不可约简；若 $V_i \cap V_{i+1} = \{x, y, \dots\} \neq \emptyset$ ，即存在至少一个相同的变量，且不适用于约简规则，则这两个相邻顶点适用于合并规则；也就是说，将这两个相邻顶点对应的事务称为可合并事务，将 V_i 和 V_{i+1} 合并为一个顶点，该合并出的新顶点对应的事务的读集为 V_i 和 V_{i+1} 对应的读集的合集，该合并出的新顶点对应的事务的写集为 V_i 和 V_{i+1} 对应的写集的合集。

另外，基于上述对目标冲突环图的介绍以及该合并规则，能够得到以下推论：一个调度 s 的 dccg 图所表示的数据异常，在通过对可合并事务进行合并后，不影响原 dccg 图等价的数据异常。

以上对目标冲突环图的获取方式以及相关规则进行了介绍，下面参考表 4 和图 10，结合示例，对目标冲突环图的获取方式进行举例说明；其中，图 10 是本申请实施例提供的一种获取目标冲突环图的示意图；一个调度 s 包括三个事务 T1、T2 以及 T3，其执行序列如表 4 所示。

表 4

时刻	T ₁	T ₂	T ₃
t ₀	$R_1[x_0]$		
t ₁		$W_2[x_1]$	
t ₂		$W_2[y_1]$	
t ₃			$R_3[y_1]$

t_4			$W_3[z_1]$
t_5	$W_1[z_2]$		

如图 10 中 (a) 所示, 该调度 s 的 CCG 图 (也即是第一冲突环图) 中的边用于表示事务之间的关联关系以及事务对应的读写操作所作用的变量, 包括以下三条边: $E_1 = (T_1 \rightarrow T_2, R_1[x_0]W_2[x_1])$, $E_2 = (T_2 \rightarrow T_3, W_2[y_1]R_3[y_1])$, $E_3 = (T_3 \rightarrow T_1, W_3[z_1]W_1[z_2])$; 对应三个变量 x , y 和 z 。

5 如图 10 中 (b) 和 (c) 所示, 该 CCG 图中包括三个变量 x , y 和 z , 分别处于边 E_1 , E_2 以及 E_3 , 以变量为顶点, 对三条边进行拆分。例如, E_1 表示为 $R_1[x_0]W_2[x_1]$, 对该边进行拆分, 得到 $E_{11} = R_1[x_0]$ 和 $E_{12} = W_2[x_1]$; E_2 表示为 $W_2[y_1]R_3[y_1]$, 对该边进行拆分, 得到 $E_{21} = W_2[y_1]$ 和 $E_{22} = R_3[y_1]$; E_3 表示为 $W_3[z_1]W_1[z_2]$, 对该边进行拆分, 得到 $E_{31} = W_3[z_1]$ 和 $E_{32} = W_1[z_2]$ 。由此, 删除 CCG 图中的顶点。

10 如图 10 中 (d) 所示, 在经过上述边拆分的过程后, 得到变量 x 的两个版本 x_0 和 x_1 , 变量 y 的一个版本 y_1 以及变量 z 的两个版本 z_1 和 z_2 。将这些变量的版本作为 CCSG 图的顶点, 同一个变量的多个版本纵向排列, 被同一个事务作用过的不同变量的版本之间横向排列 (应理解, 图 10 中 (d) 中还包括变量 z 的初始版本 z_0 , 这样才能与边 E_{31} 对应)。这里, 在相邻版本的同一变量对应的顶点之间添加边; 在同一事务所作用的变量对应的顶点之间添加边。另外, 基于前述对 CCSG 图的描述可知, CCSG 图为一个有向环构成的图, 则边 $W_3[z_1]$ 和 $W_2[y_1]$ 以及变量版本 z_0 和 y_0 对构成有向环没有作用, 因此删去这两条边和这两个变量版本, 得到如图 10 中 (e) 所示的 CCSG 图。

20 如图 10 中 (e) 所示, 该 CCSG 图 (也即是第二冲突环图) 中的顶点用于表示多个不同版本的变量, 该第二冲突环图中的边用于表示变量之间的版本变化以及同一事务所作用的变量。示意性地, 顶点 $V = \{x_0, x_1, y_1, z_1, z_2\}$; $E = \{W_2[x_1], R_2[x_1]R_2[y_1], R_3[y_1]R_3[z_1], W_1[z_2], R_1[x_2]R_1[x_0]\}$; 其中, $E_T = \{R_2[x_1]R_2[y_1], R_3[y_1]R_3[z_1], R_1[x_2]R_1[x_0]\}$; $E_s = \{W_2[x_1], W_1[z_2]\}$ 。

25 如图 10 中 (f) 所示, 该图 10 中 (f) 是通过将图 10 中 (e) 所示的 CCSG 图进行转换得到的新的 CCG 图 (也即是第三冲突环图)。转换的过程包括: 节点设备将 CCSG 图中同一个事务涉及的变量版本合并到同一个顶点里面, 得到该图 10 中 (f)。如图 10 中 (f) 所示, 每个顶点重新变为以事务为顶点, 示出了事务之间的关联关系。需要说明的是, 在一些实施例中, 该图也可以为一个无向的环图。

30 如图 10 中 (g) 所示, 顶点 V_2 和 V_3 满足上述约简规则, 也即是在这两个顶点中, 均包括相同版本的变量 y_1 , 因此, 对顶点 V_2 和 V_3 进行合并, 删去合并后的顶点中的变量 y_1 , 得到图 10 中 (h) 所示的 DCCG 图 (也即是目标冲突环图)。

以上通过举例的方式对目标冲突环图的获取方式进行了介绍, 也可以理解为确定并发事务之间的有向环图的过程。在本步骤 607 中, 节点设备在确定了待回滚的事务后, 对该目标冲突环图的结构进行维护, 便于后续执行下一次循环, 也即是对至少一个候选并发事务中的其他候选并发事务进行一致性检验。

35 下面将通过步骤 608, 对目标事务和并发事务不冲突的情况进行说明。

步骤 608, 节点设备对目标事务和并发事务进行合并, 得到合并后的目标事务, 再次执行上述步骤 603, 直至链表 TS 为空。

在本申请实施例中, 经过上述步骤 604, 第一交集和第二交集中, 存在至少一项不

为空集，且该目标事务与该并发事务不冲突，也可以理解为该目标事务与该并发事务之间不构成目标冲突环图，则目标事务与并发事务不存在数据异常，节点设备对这两个事务进行合并，得到合并后的目标事务，继续从链表 TS 中选取候选并发事务，直至链表 TS 为空。

5 下面以目标事务为 T，并发事务为 TL 为例，对本步骤 608 的实施方式进行说明。

节点设备将该并发事务 TL 的读集并入目标事务 T 的读集中，将该并发事务 TL 的写集并入目标事务 T 的写集中，以实现事务合并，得到合并后的新事务 T-new；此外，如果并发事务 TL 没有提交，则对 T-new 的目标参数（no_committed）进行增量处理，也即令 T-new.no_committed++（自增 1），表示合并得到的新事务中拥有某个已经提交的事务的读写集合的成分。基于此，节点设备将 T-new 确定为新的目标事务 T'（即为 T' = T-new），从链表 TS 中取出第二个候选并发事务定为 TL'，执行步骤 603（判断新的 T-new 与 TL' 之间是否冲突）。

步骤 609、若链表 TS 中最后一个候选并发事务与该目标事务之间不存在数据异常，则节点设备提交该目标事务。

15 在本申请实施例中，若链表 TS 中最后一个候选并发事务与该目标事务之间不存在数据异常，则表明目标事务可提交。这里，如果隔离级别参数=S，也即隔离级别是可串行化级别，此时即可满足可串行化级别，否则，满足前述表 3 所设定的隔离级别。

上述所有可选技术方案，可以采用任意结合形成本申请实施例，在此不再一一描述。

图 11 是根据本申请实施例提供的一种事务处理装置的结构示意图。该事务处理装置用于执行上述事务处理方法执行时的步骤，参见图 11，该事务处理装置 1100 包括：第一确定模块 1101 和第二确定模块 1102。

第一确定模块 1101，配置为确定目标事务的并发事务，所述并发事务与所述目标事务之间包括作用于同一变量的读写操作，所述目标事务为待提交事务；

25 第二确定模块 1102，配置为获取所述目标事务的读集和所述并发事务的写集之间的第一交集，并获取所述目标事务的写集和所述并发事务的读集之间的第二交集；当所述第一交集和所述第二交集的至少一项为非空数据集，且所述目标事务与所述并发事务冲突时，基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列表，确定数据异常类型，所述目标变量列表包括所述目标事务和所述并发事务所作用的同一变量。

30 在本申请实施例中，该事务处理装置 1100 还包括第三确定模块和回滚模块；第三确定模块，配置为从该目标事务和该并发事务中，确定待回滚的事务；回滚模块，配置为对该待回滚的事务进行回滚。

在本申请实施例中，第三确定模块，配置为基于事务优先级，将所述目标事务和所述并发事务中事务优先级最低的事务确定为所述待回滚的事务。

35 在本申请实施例中，该第三确定模块，配置为在所述目标事务和所述并发事务中，在当前事务的处理时长小于其他事务的处理时长时，将所述当前事务确定为所述事务优先级最低的事务，所述当前事务为所述目标事务和所述并发事务中的任一事务，所述其他事务为所述目标事务和所述并发事务中除所述当前事务之外的事务；在所述目标事务和所述并发事务中，当所述当前事务为读事务时，将所述当前事务确定为所述事务优先级最低的事务；在所述目标事务和所述并发事务中，当所述当前事务为写事务、且所述当前事务的处理时长小于参考时长时，将所述当前事务确定为所述事务优先级最低的事务；在所述目标事务和所述并发事务中，当所述当前事务所作用的变量数量大于所述其他事务所作用的变量数量时，将所述当前事务确定为所述事务优先级最低的事务。

40 在本申请实施例中，该事务处理装置 1100 还包括删除模块，配置为删除目标冲突

环图中该待回滚的事务所作用的变量以及对应的读写操作,其中,该目标冲突环图中的顶点用于表示事务对应的事务标识以及事务对应的读写操作所作用的变量,该目标冲突环图中的边用于表示事务之间的关联关系以及变量之间的版本变化。

在本申请实施例中,该事务处理装置 1100 还包括获取模块,配置为获取该目标事务与该并发事务之间的该目标冲突环图。

在本申请实施例中,该获取模块包括子获取模块、第一转换模块和第二转换模块;获取模块,配置为获取该目标事务与该并发事务之间的第一冲突环图,该第一冲突环图中的顶点用于表示事务对应的事务标识,该第一冲突环图中的边用于表示事务之间的关联关系以及事务对应的读写操作所作用的变量;第一转换模块,配置为对该第一冲突环图进行转换,得到第二冲突环图,该第二冲突环图中的顶点用于表示多个不同版本的变量,该第二冲突环图中的边用于表示变量之间的版本变化以及同一事务所作用的变量;第二转换模块,配置为对于该第二冲突环图进行转换,得到该目标冲突环图。

在本申请实施例中,该第一转换模块,配置为对该第一冲突环图的边进行拆分,得到该多个不同版本的变量;将该多个不同版本的变量作为顶点,并在相邻版本的同一变量对应的顶点之间添加边,以及在同一事务所作用的变量对应的顶点之间添加边,得到该第二冲突环图。

在本申请实施例中,该第二转换模块,配置为将所述第二冲突环图中同一事务所作用的变量对应的顶点进行合并,并将同一事务对应的事务标识确定为合并后的顶点;在合并后的所述第二冲突环图的相邻顶点之间添加边,得到第三冲突环图;当所述第三冲突环图的两个相邻顶点中包括同一版本的变量时,将所述两个相邻顶点合并,并删去合并后的顶点中的同一版本的变量,得到所述目标冲突环图;当所述第三冲突环图的两个相邻顶点中包括不同版本的同一变量时,将所述两个相邻顶点合并,得到所述目标冲突环图。

在本申请实施例中,该事务处理装置 1100 还包括第一处理模块、第二处理模块和第三处理模块;第一处理模块,配置为当所述第一交集为所述非空数据集时,对所述目标事务和所述并发事务之间的有向边的数量进行增量处理,并将所述第一交集中的变量添加到所述目标变量列表中,以及当所述第一交集中包括版本不同的变量时,对所述版本变更次数进行增量处理,所述有向边的数量用于指示所述目标事务与所述并发事务之间是否冲突;第二处理模块,配置为当所述第二交集为所述非空数据集时,对所述有向边的数量进行增量处理,并将所述第二交集中的变量添加到所述目标变量列表中,以及当所述第二交集中包括版本不同的变量,对所述版本变更次数进行增量处理;第三处理模块,配置为当第三交集为所述非空数据集时,基于所述并发事务的提交状态,确定所述数据异常类型,所述第三交集为所述目标事务的写集和所述并发事务的写集之间的交集。

在本申请实施例中,该第二确定模块 1102 包括第一子确定模块、第二子确定模块和第三子确定模块;第一子确定模块,配置为基于所述目标事务与所述并发事务中作用于同一变量的目标读写操作,确定所属类别;第二子确定模块,配置为基于所述目标事务与所述并发事务中作用于同一变量的目标读写操作,确定所属类别中的子类别;第三子确定模块,配置为基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种,确定所述数据异常类型。

在本申请实施例中,该第一子确定模块,还配置为当所述目标事务与所述并发事务中的所述目标读写操作满足写异常条件时,确定所述所属类别为写异常类别,满足所述写异常条件是指所述目标读写操作包括作用于不同版本的同一变量的第一指定数量的写操作;当所述目标读写操作不满足所述写异常条件,且满足读异常条件时,确定所述

所属类别为读异常类别,满足所述读异常条件是指所述目标读写操作包括作用于同一版本的同一变量的读操作和写操作;当所述目标读写操作不满足所述写异常条件和所述读异常条件,且满足交叉异常条件时,确定所述所属类别为交叉异常类别,满足所述交叉异常条件是指所述目标读写操作包括作用于不同版本的同一变量的读操作和写操作。

5 在本申请实施例中,该第二子确定模块,还配置为当所述目标变量列表中的变量数量为第二指定数量时,确定在所述所属类别中的所述子类别为单元数据异常;当所述目标变量列表中的变量数量为第三指定数量时,确定在所述所属类别中的所述子类别为二元数据异常;当所述目标变量列表中的变量数量为第四指定数量时,确定在所述所属类别中的所述子类别为多元数据异常,其中,所述第四指定数量与所述第二指定数量和所述第三指定数量均不同。

10 在本申请实施例中,当所述所属类别为写异常类别,且所述子类别为单元数据异常时,则该第三子确定模块,还配置为当所述版本变更次数为第二指定数量时,基于所述写异常类别和所述单元数据异常,确定所述数据异常类型为丢失自更新异常;当所述版本变更次数为第三指定数量,且所述目标读写操作为第五指定数量的写操作时,基于所述写异常类别和所述单元数据异常,确定所述数据异常类型为全写入异常;当所述版本变更次数为所述第三指定数量,且所述目标读写操作为第一指定数量的写操作和第六指定数量的读操作时,基于所述写异常类别和所述单元数据异常,确定所述数据异常类型为丢失更新异常。

15 在本申请实施例中,当所述所属类别为写异常类别,且所述子类别为二元数据异常,以及所述版本变更次数为第二指定数量时,第三子确定模块,还配置为当所述目标读写操作满足第一条件和第二条件时,基于所述写异常类别、所述二元数据异常和所述第二指定数量,确定所述数据异常类型为读写偏序异常,满足所述第一条件是指所述目标读写操作包括作用于不同版本的第一变量的写操作和读操作,满足所述第二条件是指所述目标读写操作包括作用于不同版本的第二变量的第一指定数量的写操作;当所述目标读写操作满足第三条件和所述第二条件时,基于所述写异常类别、所述二元数据异常和所述第二指定数量,确定所述数据异常类型为重复写偏序异常,满足所述第三条件是指所述目标读写操作包括作用于同一版本的所述第一变量的写操作和读操作;当所述目标读写操作满足第四条件和所述第二条件时,基于所述写异常类别、所述二元数据异常和所述第二指定数量,确定所述数据异常类型为全写入偏序异常,满足所述第四条件是指所述目标读写操作包括作用于不同版本的所述第一变量的所述第一指定数量的写操作。

20 在本申请实施例中,当所述所属类别为读异常类别,且所述子类别为单元数据异常,以及所述版本变更次数为第二指定数量时,第三子确定模块,还配置为当所述目标读写操作包括作用于同一变量的第一指定数量的读操作和第六指定数量的写操作,基于所述读异常类别、所述单元数据异常和所述第二指定数量,确定所述数据异常类型为不可重复读异常;当所述目标读写操作包括作用于同一变量的所述第一指定数量的写操作和所述第六指定数量的读操作,基于所述读异常类别、所述单元数据异常和所述第二指定数量,确定所述数据异常类型为中间读异常。

25 在本申请实施例中,当所述所属类别为读异常类别,且所述子类别为二元数据异常时,第三子确定模块,还配置为当所述版本变更次数为第七指定数量时,基于所述读异常类别和所述二元数据异常,确定所述数据异常类型为写读偏序异常;当所述版本变更次数为第二指定数量时,基于所述读异常类别和所述二元数据异常,确定所述数据异常类型为读偏序异常。

30 在本申请实施例中,第三子确定模块,还配置为当所述所属类别为交叉异常类别,且所述子类别为二元数据异常,以及所述版本变更次数为第二指定数量时,确定所述数

据异常类型为写偏序异常。

在本申请实施例中，该第三子确定模块，还配置为当所述所属类别为写异常类别，且所述子类别为多元数据异常时，确定所述数据异常类型为台阶式写异常；当所述所属类别为读异常类别，且所述子类别为所述多元数据异常时，则确定所述数据异常类型为台阶式读异常；当所述所属类别为交叉异常类别，且所述子类别为所述多元数据异常时，确定所述数据异常类型为台阶式交叉异常。

需要说明的是，上述实施例提供的事务处理装置在处理事务时，是以上述各功能模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能模块完成，即将装置的内部结构划分成不同的功能模块，以完成以上描述的全部或者部分功能。另外，本申请实施例提供的事务处理装置与事务处理方法属于同一构思，实现过程详见方法实施例，这里不再重复描述。

本申请实施例还提供了一种电子设备（比如计算机设备），该电子设备包括处理器和存储器，该存储器用于存储至少一条计算机程序，该至少一段计算机程序由该处理器加载并执行以实现本申请实施例中的事务处理方法。

在一些实施例中，本申请实施例所涉及的计算机程序可被部署在一个计算机设备上执行，或者在位于一个地点的多个计算机设备上执行，又或者，在分布在多个地点且通过有线网络或无线网络互连的多个计算机设备上执行，分布在多个地点且通过有线网络或无线网络互连的多个计算机设备可以组成区块链系统。

以计算机设备为服务器为例，图 12 是根据本申请实施例提供的一种服务器的结构示意图，该服务器 1200 可因配置或性能不同而产生比较大的差异，能够包括一个或一个以上处理器（Central Processing Units, CPU）1201 和一个或一个以上的存储器 1202，其中，该存储器 1202 中存储有至少一条计算机程序，该至少一条计算机程序由处理器 1201 加载并执行以实现上述各个方法实施例提供的事务处理方法。当然，该服务器还能够包括有线或无线网络接口、键盘以及输入输出接口等部件，以便进行输入输出，该服务器还能够包括其他用于实现设备功能的部件，在此不再重复描述。

本申请实施例还提供了一种计算机可读存储介质，该计算机可读存储介质应用于计算机设备，该计算机可读存储介质中存储有至少一条计算机程序，该至少一条计算机程序由处理器加载并执行以实现本申请实施例的事务处理方法。

本申请实施例还提供了一种计算机程序产品，包括计算机程序或指令，该计算机程序或指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机程序或指令，处理器执行该计算机程序或指令，使得该计算机设备执行本申请实施例提供的事务处理方法。

本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成，也可以通过程序来指令相关的硬件完成，所述的程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，磁盘或光盘等。

可以理解的是，在本申请实施例中，涉及到用户信息等相关的数据，当本申请实施例运用到具体产品或技术中时，需要获得用户许可或者同意，且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。

以上所述仅为本申请的可选实施例，并不用以限制本申请，凡在本申请的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。

权利要求书

- 1、一种数据库系统的事务处理方法，所述方法由电子设备执行，所述方法包括：
确定目标事务的并发事务，所述并发事务与所述目标事务之间包括作用于同一变量的读写操作，所述目标事务为待提交事务；
- 5 获取所述目标事务的读集和所述并发事务的写集之间的第一交集，并获取所述目标事务的写集和所述并发事务的读集之间的第二交集；
当所述第一交集和所述第二交集至少有一项为非空数据集，且所述目标事务与所述并发事务冲突时，基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列表，确定数据异常类型，所述目标变量列表包括所述目标事务和所述并发事务所作用的同一变量。
- 10 2、根据权利要求1所述的方法，其中，所述方法还包括：
从所述目标事务和所述并发事务中，确定待回滚的事务；
对所述待回滚的事务进行回滚。
- 15 3、根据权利要求2所述的方法，其中，所述从所述目标事务和所述并发事务中，确定待回滚的事务，包括：
基于事务优先级，将所述目标事务和所述并发事务中事务优先级最低的事务确定为所述待回滚的事务。
- 4、根据权利要求3所述的方法，其中，所述基于事务优先级，将所述目标事务和所述并发事务中事务优先级最低的事务确定为所述待回滚的事务之前，所述方法还包括下述任一项：
- 20 在所述目标事务和所述并发事务中，在当前事务的处理时长小于其他事务的处理时长时，将所述当前事务确定为所述事务优先级最低的事务，所述当前事务为所述目标事务和所述并发事务中的任一事务，所述其他事务为所述目标事务和所述并发事务中除所述当前事务之外的事务；
- 25 在所述目标事务和所述并发事务中，在所述当前事务为读事务时，将所述当前事务确定为所述事务优先级最低的事务；
在所述目标事务和所述并发事务中，在所述当前事务为写事务、且所述当前事务的处理时长小于参考时长时，将所述当前事务确定为所述事务优先级最低的事务；
在所述目标事务与所述并发事务中，在所述当前事务所作用的变量数量大于所述其他事务所作用的变量数量时，将所述当前事务确定为所述事务优先级最低的事务。
- 30 5、根据权利要求2至4任一项所述的方法，其中，所述对所述待回滚的事务进行回滚之后，所述方法还包括：
删除目标冲突环图中所述待回滚的事务所作用的变量以及对应的读写操作，其中，所述目标冲突环图中的顶点用于表示事务对应的事务标识以及事务对应的读写操作所作用的变量，所述目标冲突环图中的边用于表示事务之间的关联关系以及变量之间的版本变化。
- 35 6、根据权利要求5所述的方法，其中，所述方法还包括：
获取所述目标事务与所述并发事务之间的第一冲突环图，所述第一冲突环图中的顶点用于表示事务对应的事务标识，所述第一冲突环图中的边用于表示事务之间的关联关系以及事务对应的读写操作所作用的变量；
40 对所述第一冲突环图进行转换，得到第二冲突环图，所述第二冲突环图中的顶点用于表示多个不同版本的变量，所述第二冲突环图中的边用于表示变量之间的版本变化以及同一事务所作用的变量；

- 对所述第二冲突环图进行转换,得到所述目标冲突环图。
- 7、根据权利要求6所述的方法,其中,所述对所述第一冲突环图进行转换,得到第二冲突环图,包括:
- 对所述第一冲突环图的边进行拆分,得到所述多个不同版本的变量;
- 5 将所述多个不同版本的变量作为顶点,并在相邻版本的同一变量对应的顶点之间添加边,以及在同一事务所作用的变量对应的顶点之间添加边,得到所述第二冲突环图。
- 8、根据权利要求6所述的方法,其中,所述对所述第二冲突环图进行转换,得到所述目标冲突环图,包括:
- 将所述第二冲突环图中同一事务所作用的变量对应的顶点进行合并,并将同一事务
- 10 对应的事务标识确定为合并后的顶点;
- 在合并后的所述第二冲突环图的相邻顶点之间添加边,得到第三冲突环图;
- 当所述第三冲突环图的两个相邻顶点中包括同一版本的变量时,将所述两个相邻顶点合并,并删去合并后的顶点中的同一版本的变量,得到所述目标冲突环图;
- 当所述第三冲突环图的两个相邻顶点中包括不同版本的同一变量时,将所述两个相
- 15 邻顶点合并,得到所述目标冲突环图。
- 9、根据权利要求1至4任一项所述的方法,其中,所述方法还包括:
- 当所述第一交集为所述非空数据集时,对所述目标事务和所述并发事务之间的有向边的数量进行增量处理,并将所述第一交集集中的变量添加到所述目标变量列表中,以及
- 20 当所述第一交集集中包括版本不同的变量时,对所述版本变更次数进行增量处理,所述有向边的数量用于指示所述目标事务与所述并发事务之间是否冲突;
- 当所述第二交集为所述非空数据集时,对所述有向边的数量进行增量处理,并将所述第二交集集中的变量添加到所述目标变量列表中,以及当所述第二交集集中包括版本不同的变量,对所述版本变更次数进行增量处理;
- 当第三交集为所述非空数据集时,基于所述并发事务的提交状态,确定所述数据异常类型,所述第三交集为所述目标事务的写集和所述并发事务的写集之间的交集。
- 25 10、根据权利要求1至4任一项所述的方法,其中,所述基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列表,确定数据异常类型,包括:
- 基于所述目标事务与所述并发事务中作用于同一变量的目标读写操作,确定所属类别;
- 30 基于所述所属类别以及所述目标变量列表,确定在所述所属类别中的子类别;
- 基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种,确定所述数据异常类型。
- 11、根据权利要求10所述的方法,其中,所述基于所述目标事务与所述并发事务中作用于同一变量的目标读写操作,确定所属类别,包括下述任一项:
- 35 当所述目标事务与所述并发事务中的所述目标读写操作满足写异常条件时,确定所述所属类别为写异常类别,满足所述写异常条件是指所述目标读写操作包括作用于不同版本的同一变量的第一指定数量的写操作;
- 当所述目标读写操作不满足所述写异常条件,且满足读异常条件时,确定所述所属类别为读异常类别,满足所述读异常条件是指所述目标读写操作包括作用于同一版本的
- 40 同一变量的读操作和写操作;
- 当所述目标读写操作不满足所述写异常条件和所述读异常条件,且满足交叉异常条件时,确定所述所属类别为交叉异常类别,满足所述交叉异常条件是指所述目标读写操作包括作用于不同版本的同一变量的读操作和写操作。

12、根据权利要求 10 所述的方法，其中，所述基于所述所属类别以及所述目标变量列表，确定在所述所属类别中的子类别，包括下述任一项：

当所述目标变量列表中的变量数量为第二指定数量时，确定在所述所属类别中的所述子类别为单元数据异常；

5 当所述目标变量列表中的变量数量为第三指定数量时，确定在所述所属类别中的所述子类别为二元数据异常；

当所述目标变量列表中的变量数量为第四指定数量时，确定在所述所属类别中的所述子类别为多元数据异常，其中，所述第四指定数量与所述第二指定数量和所述第三指定数量均不同。

10 13、根据权利要求 10 所述的方法，其中，当所述所属类别为写异常类别，且所述子类别为单元数据异常时，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括下述任一项：

当所述版本变更次数为第二指定数量时，基于所述写异常类别和所述单元数据异常，确定所述数据异常类型为丢失自更新异常；

15 当所述版本变更次数为第三指定数量，且所述目标读写操作为第五指定数量的写操作时，基于所述写异常类别和所述单元数据异常，确定所述数据异常类型为全写入异常；

当所述版本变更次数为所述第三指定数量，且所述目标读写操作为第一指定数量的写操作和第六指定数量的读操作时，基于所述写异常类别和所述单元数据异常，确定所述数据异常类型为丢失更新异常。

20 14、根据权利要求 10 所述的方法，其中，当所述所属类别为写异常类别，且所述子类别为二元数据异常，以及所述版本变更次数为第二指定数量时，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括下述任一项：

25 当所述目标读写操作满足第一条件和第二条件时，基于所述写异常类别、所述二元数据异常和所述第二指定数量，确定所述数据异常类型为读写偏序异常，满足所述第一条件是指所述目标读写操作包括作用于不同版本的第一变量的写操作和读操作，满足所述第二条件是指所述目标读写操作包括作用于不同版本的第二变量的第一指定数量的写操作；

30 当所述目标读写操作满足第三条件和所述第二条件时，基于所述写异常类别、所述二元数据异常和所述第二指定数量，确定所述数据异常类型为重复写偏序异常，满足所述第三条件是指所述目标读写操作包括作用于同一版本的所述第一变量的写操作和读操作；

35 当所述目标读写操作满足第四条件和所述第二条件时，基于所述写异常类别、所述二元数据异常和所述第二指定数量，确定所述数据异常类型为全写入偏序异常，满足所述第四条件是指所述目标读写操作包括作用于不同版本的所述第一变量的所述第一指定数量的写操作。

40 15、根据权利要求 10 所述的方法，其中，当所述所属类别为读异常类别，且所述子类别为单元数据异常，以及所述版本变更次数为第二指定数量时，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括下述任一项：

当所述目标读写操作包括作用于同一变量的第一指定数量个读操作和第六指定数量的写操作，基于所述读异常类别、所述单元数据异常和所述第二指定数量，确定所述数据异常类型为不可重复读异常；

当所述目标读写操作包括作用于同一变量的所述第一指定数量的写操作和所述第六指定数量的读操作，基于所述读异常类别、所述单元数据异常和所述第二指定数量，确定所述数据异常类型为中间读异常。

- 16、根据权利要求 10 所述的方法，其中，当所述所属类别为读异常类别，且所述子类别为二元数据异常时，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括下述任一项：

当所述版本变更次数为第七指定数量时，基于所述读异常类别和所述二元数据异常，确定所述数据异常类型为写读偏序异常；

- 10 当所述版本变更次数为第二指定数量时，基于所述读异常类别和所述二元数据异常，确定所述数据异常类型为读偏序异常。

17、根据权利要求 10 所述的方法，其中，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括：

- 15 当所述所属类别为交叉异常类别，且所述子类别为二元数据异常，以及所述版本变更次数为第二指定数量时，确定所述数据异常类型为写偏序异常。

18、根据权利要求 10 所述的方法，其中，所述基于在所述所属类别中的所述子类别、所述版本变更次数以及所述目标读写操作中的至少一种，确定所述数据异常类型，包括下述任一项：

- 20 当所述所属类别为写异常类别，且所述子类别为多元数据异常时，确定所述数据异常类型为台阶式写异常；

当所述所属类别为读异常类别，且所述子类别为所述多元数据异常时，则确定所述数据异常类型为台阶式读异常；

当所述所属类别为交叉异常类别，且所述子类别为所述多元数据异常时，确定所述数据异常类型为台阶式交叉异常。

- 25 19、一种事务处理装置，所述装置包括：

第一确定模块，配置为确定目标事务的并发事务，所述并发事务与所述目标事务之间包括作用于同一变量的读写操作，所述目标事务为待提交事务；

- 30 第二确定模块，配置为获取所述目标事务的读集和所述并发事务的写集之间的第一交集，并获取所述目标事务的写集和所述并发事务的读集之间的第二交集；当所述第一交集和所述第二交集的至少一项为非空数据集，且所述目标事务与所述并发事务冲突时，基于所述目标事务和所述并发事务所作用的同一变量的版本变更次数和目标变量列表，确定数据异常类型，所述目标变量列表包括所述目标事务和所述并发事务所作用的同一变量。

- 35 20、一种电子设备，所述电子设备包括处理器和存储器，所述存储器用于存储至少一条计算机程序，所述至少一条计算机程序由所述处理器加载并执行时，实现权利要求 1 至权利要求 18 中任一项所述的数据库系统的事务处理方法。

21、一种计算机可读存储介质，所述计算机可读存储介质中存储有至少一条计算机程序，所述至少一条计算机程序由处理器加载并执行时，实现权利要求 1 至权利要求 18 中任一项所述的数据库系统的事务处理方法。

- 40 22、一种计算机程序产品，包括计算机程序或指令，所述计算机程序或指令被处理器执行时，实现权利要求 1 至权利要求 18 中任一项所述的数据库系统的事务处理方法。