(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111143389 B
(45)授权公告日 2022.08.05

(21)申请号 201911381740.3
(22)申请日 2019.12.27
(65)同一申请的已公布的文献号
申请公布号 CN 111143389 A
(43)申请公布日 2020.05.12
(73)专利权人 腾讯科技(深圳)有限公司
地址 518057 广东省深圳市南山区高新区科技中一路腾讯大厦35层
专利权人 中国人民大学
(72)发明人 李海翔 卢卫 赵展浩 杜小勇 潘安群
(74)专利代理机构 北京三高永信知识产权代理有限责任公司 11138
专利代理师 郭新禹
(51)Int.Cl.
G06F 16/23(2019.01)
(56)对比文件
CN 109739935 A,2019.05.10
CN 106598992 A,2017.04.26
CN 110019530 A,2019.07.16
US 2015317349 A1,2015.11.05
CN 105389161 A,2016.03.09
审查员 孙国辉

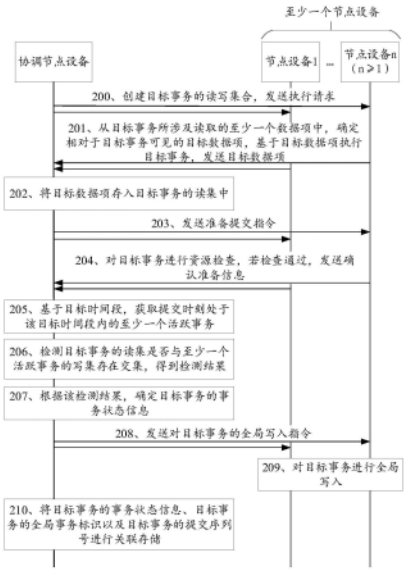
权利要求书4页 说明书23页 附图5页

(54)发明名称

事务执行方法、装置、计算机设备及存储介质

(57)摘要

本申请公开了一种事务执行方法、装置、计算机设备及存储介质，属于数据库技术领域。本申请基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务，检测该目标事务的读集是否与该至少一个活跃事务的写集存在交集，得到检测结果，根据检测结果，确定该目标事务的事务状态信息，该事务状态信息用于表示与该检测结果对应的执行状态，在全局写入该目标事务时，存储该目标事务的事务状态信息，仍然会对写写冲突进行检测，但不论检测结果表示是否存在冲突，都不影响目标事务的全局写入，提升了数据库系统中事务执行的并发度，提升了数据库系统中的事务执行效率。



CN 111143389 B

1.一种事务执行方法,其特征在于,所述方法包括:

基于目标时间段,获取提交时刻处于所述目标时间段内的至少一个活跃事务,所述目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段;

检测所述目标事务的读集是否与所述至少一个活跃事务的写集存在交集,得到检测结果,所述读集为所述目标事务所读取的数据项集合,所述写集为所述至少一个活跃事务所写入的数据项集合;

根据所述检测结果,确定所述目标事务的事务状态信息,所述事务状态信息用于表示与所述检测结果对应的执行状态;

在全局写入所述目标事务时,存储所述目标事务的事务状态信息。

2.根据权利要求1所述的方法,其特征在于,所述根据所述检测结果,确定所述目标事务的事务状态信息包括:

若所述检测结果为所述读集与所述写集存在交集,将所述事务状态信息确定为已回滚状态;或,

若所述检测结果为所述读集与所述写集不存在交集,将所述事务状态信息确定为已提交状态。

3.根据权利要求1所述的方法,其特征在于,所述存储所述目标事务的事务状态信息包括:

将所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号进行关联存储。

4.根据权利要求3所述的方法,其特征在于,所述将所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号进行关联存储包括:

基于键值对的形式进行关联存储,其中,以所述目标事务的全局事务标识作为键,以目标事务的所述事务状态信息以及所述目标事务的提交序列号作为值;或,

基于事务状态元组的形式进行关联存储,其中,所述事务状态元组包括所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号。

5.根据权利要求1所述的方法,其特征在于,所述在全局写入所述目标事务时,存储所述目标事务的事务状态信息之后,所述方法还包括:

若所述目标事务的事务状态信息为已回滚状态,在所述目标事务所生成的数据项中仅记录被修改的属性值。

6.根据权利要求1所述的方法,其特征在于,所述检测所述目标事务的读集是否与所述至少一个活跃事务的写集存在交集之前,所述方法还包括:

从所述目标事务所涉及读取的至少一个数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存入所述目标事务的读集中。

7.根据权利要求6所述的方法,其特征在于,所述从所述目标事务所涉及读取的至少一个数据项中,确定相对于所述目标事务可见的目标数据项包括:

对所述至少一个数据项中任一数据项,基于生成所述数据项的事务的全局事务标识,查询所述事务的事务状态信息;

若查询不到所述事务的事务状态信息,且所述事务的全局事务标识与所述目标事务的全局事务标识不相等,确定所述数据项不为目标数据项;

若查询得到所述事务的事务状态信息，基于所述事务的事务状态信息以及所述事务的提交序列号，确定所述数据项是否为目标数据项。

8.根据权利要求7所述的方法，其特征在于，所述基于所述事务的事务状态信息以及所述事务的提交序列号，确定所述数据项是否为目标数据项包括：

若所述事务的全局事务标识等于所述目标事务的全局事务标识，或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号，确定所述数据项为目标数据项；否则，确定所述数据项不为目标数据项。

9.根据权利要求7所述的方法，其特征在于，所述基于所述事务的事务状态信息以及所述事务的提交序列号，确定所述数据项是否为目标数据项包括：

若所述事务的全局事务标识等于所述目标事务的全局事务标识，或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号，或所述事务的事务状态信息为已回滚状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号，确定所述数据项为目标数据项；否则，确定所述数据项不为目标数据项。

10.根据权利要求1所述的方法，其特征在于，所述在全局写入所述目标事务时，存储所述目标事务的事务状态信息之后，所述方法还包括：

在存储所述目标事务所生成的数据项时，若内存的剩余空间小于空间阈值，将所述内存中的已有数据项转存至磁盘中，将所述目标事务所生成的数据项存入所述内存。

11.根据权利要求10所述的方法，其特征在于，所述将所述内存中的已有数据项转存至磁盘中包括：

将所述内存中已有的数据项与磁盘中的数据项按照键序进行合并，在合并过程中，若检测到生成任一数据项的事务的事务状态信息为已回滚状态，跳过对所述数据项的合并操作。

12.根据权利要求1所述的方法，其特征在于，所述方法还包括：

每间隔目标时长，获取当前正在执行的事务中的最小开始序列号；

删除提交序列号小于所述最小开始序列号且事务状态信息为已回滚状态的事务所操作的数据项。

13.一种事务执行装置，其特征在于，所述装置包括：

获取模块，用于基于目标时间段，获取提交时刻处于所述目标时间段内的至少一个活跃事务，所述目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段；

检测模块，用于检测所述目标事务的读集是否与所述至少一个活跃事务的写集存在交集，得到检测结果，所述读集为所述目标事务所读取的数据项集合，所述写集为所述至少一个活跃事务所写入的数据项集合；

确定模块，用于根据所述检测结果，确定所述目标事务的事务状态信息，所述事务状态信息用于表示与所述检测结果对应的执行状态；

存储模块，用于在全局写入所述目标事务时，存储所述目标事务的事务状态信息。

14.根据权利要求13所述的装置，其特征在于，所述确定模块用于：

若所述检测结果为所述读集与所述写集存在交集，将所述事务状态信息确定为已回滚状态；或，

若所述检测结果为所述读集与所述写集不存在交集，将所述事务状态信息确定为已提

交状态。

15.根据权利要求13所述的装置,其特征在于,所述存储模块包括:

存储单元,用于将所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号进行关联存储。

16.根据权利要求15所述的装置,其特征在于,所述存储单元用于:

基于键值对的形式进行关联存储,其中,以所述目标事务的全局事务标识作为键,以目标事务的所述事务状态信息以及所述目标事务的提交序列号作为值;或,

基于事务状态元组的形式进行关联存储,其中,所述事务状态元组包括所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号。

17.根据权利要求13所述的装置,其特征在于,所述装置还用于:

若所述目标事务的事务状态信息为已回滚状态,在所述目标事务所生成的数据项中仅记录被修改的属性值。

18.根据权利要求13所述的装置,其特征在于,所述装置还包括:

确定存入模块,用于从所述目标事务所涉及读取的至少一个数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存入所述目标事务的读集中。

19.根据权利要求18所述的装置,其特征在于,所述确定存入模块包括:

查询单元,用于对所述至少一个数据项中任一数据项,基于生成所述数据项的事务的全局事务标识,查询所述事务的事务状态信息;

确定单元,用于若查询不到所述事务的事务状态信息,且所述事务的全局事务标识与所述目标事务的全局事务标识不相等,确定所述数据项不为目标数据项;

所述确定单元,还用于若查询得到所述事务的事务状态信息,基于所述事务的事务状态信息以及所述事务的提交序列号,确定所述数据项是否为目标数据项。

20.根据权利要求19所述的装置,其特征在于,所述确定单元用于:

若所述事务的全局事务标识等于所述目标事务的全局事务标识,或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,确定所述数据项为目标数据项;否则,确定所述数据项不为目标数据项。

21.根据权利要求19所述的装置,其特征在于,所述确定单元用于:

若所述事务的全局事务标识等于所述目标事务的全局事务标识,或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,或所述事务的事务状态信息为已回滚状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,确定所述数据项为目标数据项;否则,确定所述数据项不为目标数据项。

22.根据权利要求13所述的装置,其特征在于,所述存储模块还用于:

在存储所述目标事务所生成的数据项时,若内存的剩余空间小于空间阈值,将所述内存中的已有数据项转存至磁盘中,将所述目标事务所生成的数据项存入所述内存。

23.根据权利要求22所述的装置,其特征在于,所述存储模块还用于:

将所述内存中已有的数据项与磁盘中的数据项按照键序进行合并,在合并过程中,若检测到生成任一数据项的事务的事务状态信息为已回滚状态,跳过对所述数据项的合并操作。

24.根据权利要求13所述的装置,其特征在于,所述装置还用于:

每间隔目标时长，获取当前正在执行的事务中的最小开始序列号；

删除提交序列号小于所述最小开始序列号且事务状态信息为已回滚状态的事务所操作的数据项。

25.一种计算机设备，其特征在于，所述计算机设备包括一个或多个处理器和一个或多个存储器，所述一个或多个存储器中存储有至少一条程序代码，所述至少一条程序代码由所述一个或多个处理器加载并执行以实现如权利要求1至权利要求12任一项所述的事务执行方法。

26.一种存储介质，其特征在于，所述存储介质中存储有至少一条程序代码，所述至少一条程序代码由处理器加载并执行以实现如权利要求1至权利要求12任一项所述的事务执行方法。

事务执行方法、装置、计算机设备及存储介质

技术领域

[0001] 本申请涉及数据库技术领域，特别涉及一种事务执行方法、装置、计算机设备及存储介质。

背景技术

[0002] 随着数据库技术的发展，为了能够适应大数据、云计算等业务场景，分布式数据库系统逐渐变得普及。在分布式数据库系统中采用的并发控制算法普遍不允许写写冲突(写写冲突是指两个不同事务对同一数据项执行写操作)的发生，并发控制算法会采用回滚或者延迟的机制，来阻塞写写冲突的事务提交。由于在高并发的场景下写写冲突普遍存在，因此并发控制算法对写写冲突的约束机制，会大大影响事务执行的并发度，导致分布式数据库系统内大量事务回滚，影响分布式数据库系统的事务执行效率。有鉴于此，亟需一种能够提升分布式数据库系统的事务执行效率的方法。

发明内容

[0003] 本申请实施例提供了一种事务执行方法、装置、计算机设备及存储介质，能够提升分布式数据库系统的事务执行效率。该技术方案如下：

[0004] 一方面，提供了一种事务执行方法，该方法包括：

[0005] 基于目标时间段，获取提交时刻处于所述目标时间段内的至少一个活跃事务，所述目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段；

[0006] 检测所述目标事务的读集是否与所述至少一个活跃事务的写集存在交集，得到检测结果，所述读集为所述目标事务所读取的数据项集合，所述写集为所述至少一个活跃事务所写入的数据项集合；

[0007] 根据所述检测结果，确定所述目标事务的事务状态信息，所述事务状态信息用于表示与所述检测结果对应的执行状态；

[0008] 在全局写入所述目标事务时，存储所述目标事务的事务状态信息。

[0009] 一方面，提供了一种事务执行装置，该装置包括：

[0010] 获取模块，用于基于目标时间段，获取提交时刻处于所述目标时间段内的至少一个活跃事务，所述目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段；

[0011] 检测模块，用于检测所述目标事务的读集是否与所述至少一个活跃事务的写集存在交集，得到检测结果，所述读集为所述目标事务所读取的数据项集合，所述写集为所述至少一个活跃事务所写入的数据项集合；

[0012] 确定模块，用于根据所述检测结果，确定所述目标事务的事务状态信息，所述事务状态信息用于表示与所述检测结果对应的执行状态；

[0013] 存储模块，用于在全局写入所述目标事务时，存储所述目标事务的事务状态信息。

[0014] 在一种可能实施方式中，所述确定模块用于：

[0015] 若所述检测结果为所述读集与所述写集存在交集，将所述事务状态信息确定为已

回滚状态;或,

[0016] 若所述检测结构为所述读集与所述写集不存在交集,将所述事务状态信息确定为已提交状态。

[0017] 在一种可能实施方式中,所述存储模块包括:

[0018] 存储单元,用于将所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号进行关联存储。

[0019] 在一种可能实施方式中,所述存储单元用于:

[0020] 基于键值对的形式进行关联存储,其中,以所述目标事务的全局事务标识作为键,以目标事务的所述事务状态信息以及所述目标事务的提交序列号作为值;或,

[0021] 基于事务状态元组的形式进行关联存储,其中,所述事务状态元组包括所述目标事务的事务状态信息、所述目标事务的全局事务标识以及所述目标事务的提交序列号。

[0022] 在一种可能实施方式中,所述装置还用于:

[0023] 若所述目标事务的事务状态信息为已回滚状态,在所述目标事务所生成的数据项中仅记录被修改的属性值。

[0024] 在一种可能实施方式中,所述装置还包括:

[0025] 确定存入模块,用于从所述目标事务所涉及读取的至少一个数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存入所述目标事务的读集中。

[0026] 在一种可能实施方式中,所述确定存入模块包括:

[0027] 查询单元,用于对所述至少一个数据项中任一数据项,基于生成所述数据项的事务的全局事务标识,查询所述事务的事务状态信息;

[0028] 确定单元,用于若查询不到所述事务的事务状态信息,且所述事务的全局事务标识与所述目标事务的全局事务标识不相等,确定所述数据项不为目标数据项;

[0029] 所述确定单元,还用于若查询得到所述事务的事务状态信息,基于所述事务的事务状态信息以及所述事务的提交序列号,确定所述数据项是否为目标数据项。

[0030] 在一种可能实施方式中,所述确定单元用于:

[0031] 若所述事务的全局事务标识等于所述目标事务的全局事务标识,或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,确定所述数据项为目标数据项;否则,确定所述数据项不为目标数据项。

[0032] 在一种可能实施方式中,所述确定单元用于:

[0033] 若所述事务的全局事务标识等于所述目标事务的全局事务标识,或所述事务的事务状态信息为已提交状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,或所述事务的事务状态信息为已回滚状态且所述事务的提交序列号小于或等于所述目标事务的开始序列号,确定所述数据项为目标数据项;否则,确定所述数据项不为目标数据项。

[0034] 在一种可能实施方式中,所述存储模块还用于:

[0035] 在存储所述目标事务所生成的数据项时,若内存的剩余空间小于空间阈值,将所述内存中的已有数据项转存至磁盘中,将所述目标事务所生成的数据项存入所述内存。

[0036] 在一种可能实施方式中,所述存储模块还用于:

[0037] 将所述内存中已有的数据项与磁盘中的数据项按照键序进行合并,在合并过程

中，若检测到生成任一数据项的事务的事务状态信息为已回滚状态，跳过对所述数据项的合并操作。

[0038] 在一种可能实施方式中，所述装置还用于：

[0039] 每间隔目标时长，获取当前正在执行的事务中的最小开始序列号；

[0040] 删除提交序列号小于所述最小开始序列号且事务状态信息为已回滚状态的事务所操作的数据项。

[0041] 一方面，提供了一种计算机设备，该计算机设备包括一个或多个处理器和一个或多个存储器，该一个或多个存储器中存储有至少一条程序代码，该至少一条程序代码由该一个或多个处理器加载并执行以实现如上述任一种可能实现方式的事务执行方法所执行的操作。

[0042] 一方面，提供了一种存储介质，该存储介质中存储有至少一条程序代码，该至少一条程序代码由处理器加载并执行以实现如上述任一种可能实现方式的事务执行方法所执行的操作。

[0043] 本申请实施例提供的技术方案带来的有益效果至少包括：

[0044] 通过基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务，该目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段，检测该目标事务的读集是否与该至少一个活跃事务的写集存在交集，得到检测结果，该读集为该目标事务所读取的数据项集合，该写集为该至少一个活跃事务所写入的数据项集合，根据该检测结果，确定该目标事务的事务状态信息，该事务状态信息用于表示与该检测结果对应的执行状态，在全局写入该目标事务时，存储该目标事务的事务状态信息，可以看出，仍然会对写写冲突进行检测，但不论检测结果表示是否存在冲突，都不影响目标事务的全局写入，而是为目标事务配置与检测结果对应的事务状态信息，在写入阶段同时存储该事务状态信息，提升了数据库系统中事务执行的并发度，提升了数据库系统中的事务执行效率。

附图说明

[0045] 为了更清楚地说明本申请实施例中的技术方案，下面将对实施例描述中所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。

[0046] 图1是本申请实施例提供的一种事务执行方法的实施环境示意图；

[0047] 图2是本申请实施例提供的一种事务执行方法的交互流程图；

[0048] 图3是本申请实施例提供的一种数据项的数据结构的逻辑示意图；

[0049] 图4是本申请实施例提供的一种并发控制方法的流程图；

[0050] 图5是本申请实施例提供的一种磁盘分层管理的原理性示意图；

[0051] 图6是本申请实施例提供的一种事务执行装置的结构示意图；

[0052] 图7是本申请实施例提供的一种计算机设备的结构示意图。

具体实施方式

[0053] 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。

[0054] 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n”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。

[0055] 本申请中术语“至少一个”是指一个或多个，“多个”的含义是指两个或两个以上，例如，多个第一位置是指两个或两个以上的第一位置。

[0056] 在介绍本申请实施例之前，需要引入一些数据库技术中的基本概念：

[0057] 一、事实数据库

[0058] 本申请实施例涉及的数据库系统中可以包括节点设备和协调节点设备，节点设备或协调节点设备的数据库中可以存储有多个数据表，每个数据表可以用于存储一个或多个数据项(也称为元组)。其中，节点设备或协调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SQL(Structured Query Language，结构化查询语言)数据库、NoSQL(Non-relational SQL，泛指非关系型数据库)、NewSQL(泛指各种新式的可拓展/高性能数据库)等，在本申请实施例中对数据库的类型不作具体限定。

[0059] 在一些实施例中，本申请实施例涉及一种新型的数据库模型，可以称为“事实数据库”，事实数据库是事实数据表的延伸，下面对事实数据表进行介绍：在业务系统中常常会使用事实数据表来存放业务的操作记录等数据，比如存放现金登记事务所产生的数据等，这些数据表通常数据量较大，包含了大量的数据行，事实数据表的主要特点是保护数字信息(俗称为“事实数据”)，并且这些数字信息可以汇总，从而提供给有关单位进行查询历史记录、审计等操作，例如，可以汇总某一时间段内公司每日营业额情况，从而方便监管部门对企业进行监管。

[0060] 基于事实数据表的概念，提供一种事实数据库，事实数据库是对事实数据表的扩充：数据库自动对业务的操作记录等数据进行追踪，从而对已发生的事件进行持久化存储(无论是提交还是回滚的数据均被持久化存储)，做到业务中所有操作都可以被追溯。与事实数据表的区别在于，事实数据库不需要业务来对操作记录进行维护，而是由数据库进行自动维护，从而大大降低了业务系统的实现复杂度，同时，事实数据库在数据库层面的操作追踪，具有更高的准确性，所有事务的回滚和提交操作均会被数据库发现，并记录称为“事实”，从而可以更好地为金融风险管控、审计、决策支持等应用提供服务。

[0061] 在一个示例性场景中，比如若采用事实数据库来支持信用卡支付系统，数据库将自动追踪信用卡的每笔交易，并记录每一笔交易在数据库中精确的状态和操作时间，基于这些被记录下来的数据可以轻松实现对信用卡交易的审计。同时，系统中被回滚的交易也被数据库记录下来，可以用来事后对这些异常情况进行精确地原因分析，从而确认是否存在发现信用卡盗刷而导致交易回滚等异常情况的发生。

[0062] 二、数据的全态

[0063] 上述数据库系统中的数据项，基于状态属性的不同，可以划分为三种状态：当前态、过渡态和历史态，该三种状态合称为“数据的全态”，简称全态数据，全态数据中的各个不同状态属性，可以用于标识数据在其生命周期轨迹中所处的状态。

[0064] 当前态(Current State)：最新版本的数据项，是处于当前阶段的数据项。

[0065] 历史态(Historical state):数据项在历史上的一个状态,其值是旧值,不是当前值。多个历史态数据项可以对应于同一主键标识,反映了具有该主键标识的各个数据项的状态变迁的过程。处于历史态的数据项,只能被读取而不能被修改或删除。

[0066] 过渡态(Transitional State):不是当前态数据项也不是历史态数据项,处于从当前态向历史态转变的过程中,这种处于过渡态的数据也称为半衰数据。

[0067] 基于上述名词解释,不同的数据项可以具有相同的主键标识(primary key,PK),此时,具有相同主键标识的各个数据项可以构成一个全态数据集,该全态数据集内的各个数据项在本质上用于表示全态数据,也即是说,在对具有该主键标识的初始数据项进行多次修改(或删除)的过程中,由于修改(或删除)时刻不同而产生的多个不同的版本,即可构成一个全态数据集。在一个全态数据集中,有的数据项处于当前态,有的数据项处于过渡态,有的数据项处于历史态数据。这里的全态数据集是指一个抽象的、虚拟的集合概念,同一个全态数据集内的各个数据项可以分布式地存储在不同的物理机上。数据库系统在存储各个数据项时,可以采用指针将对应于同一主键标识的各个数据项按照时序链接起来,便于查询全态数据的生命周期轨迹。

[0068] 三、数据项的可见性

[0069] 数据项的可见与否(数据项的可见性)是针对于事务而言的,某个数据项可能针对一些事务可见,针对一些事务不可见。此外,在一些实施例中,数据项的可见与否还与用户的权限相关,比如普通用户的权限较低,无法访问处于已回滚状态的数据项,而超级用户(比如系统的管理用户)的权限较高,可以访问处于已回滚状态的数据项,具体判断数据项可见性的算法将在后文的实施例中进行详述。

[0070] 图1是本申请实施例提供的一种事务执行方法的实施环境示意图。参见图1,本实施例可以应用于分布式数据库系统,该系统中可以包括网关服务器101、全局序列号生成集群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和协调节点设备。

[0071] 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API(Application Programming Interface,应用程序编程接口)将该读写请求发送至网关服务器101,比如,该API可以是MySQL API(一种关系型数据库系统提供的API)。

[0072] 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中的任一个节点设备或任一协调节点设备合并在同一个物理机上,也即是,让某个节点设备或协调节点设备充当网关服务器101。

[0073] 全局序列号生成集群102用于生成全局事务的全局序列号,该全局事务可以是指涉及到多个节点设备的事务,例如全局读事务可以涉及到对多个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节点设备上的数据写入。采用使用集群的形式来实现该全局序列号的生成,可以防止单点故障。

[0074] 可选地,全局序列号是一个在分布式数据库系统中全局唯一且单调递增的标识信息,事务的全局序列号可以包括开始序列号(start_tn)或者提交序列号(commit_tn)中至少一项,开始序列号用于表示事务执行过程的开始时刻,提交序列号用于表示事务执行过

程的提交时刻,开始序列号和提交序列号由事务执行的协调节点设备向全局序列号生成集群102申请,全局序列号可以采用物理时钟、逻辑时钟或者混合物理时钟中至少一项。全局序列号能够标识出每个事务开始/提交的顺序,以此可以反映出事务的全序关系。

[0075] 在一个示例性场景中,全局序列号可以采用混合物理时钟的方式生成,全局序列号由八字节组成,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,因此理论上一共可以表示约为557.8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}-1个(约100万个)计数,基于上述数据结构,如果单机(任一节点设备)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,序列号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$(2^{44}-1)*2^{20}$个事务。

[0076] 在一些实施例中,全局序列号生成集群102可以对外提供如下接口:获取当前序列号的接口以及获取最大事务提交序列号(max_commit_gts)的接口。协调节点设备可以通过调用上述至少一种接口来进行与全局序列号生成集群102之间的交互。

[0077] 可选地,当任一协调节点设备调用获取当前序列号的接口时,由于全局序列号的一部分用于表示物理时间戳,因此全局序列号生成集群102可以基于当前物理时间戳生成一个当前序列号,将该当前序列号返回至协调节点设备,协调节点设备可以将该当前序列号作为事务的提交序列号,或者,在一些涉及到冲突验证的场景中,还可以将该全局序列号作为事务的验证序列号,具体情况可以视协调节点设备的业务需求而定。

[0078] 可选地,当任一协调节点设备调用获取max_commit_gts的接口时,全局序列号生成集群102还可以确定当前时刻数据库系统内已全局提交(或已全局回滚)的最大事务的提交序列号max_commit_gts,将该max_commit_gts返回至协调节点设备,协调节点设备可以将获取到的max_commit_gts作为事务的开始序列号(start_tn),进一步地,在该事务进行全局提交(或全局回滚)时,协调节点设备还需要与全局序列号生成集群102进行一次交互,以获取该事务的提交序列号,比如协调节点设备可以通过调用上述获取当前序列号的接口来获取提交序列号,,全局序列号生成集群102获取到为该事务分配提交序列号之后,由于该事务在当前时刻处于待提交(或待回滚)状态,说明该事务为当前时刻的最大事务,因此将已有的max_commit_gts更新为上述事务的提交序列号,使得全局序列号生成集群102完成了对max_commit_gts的维护与更新,具体过程将在下个实施例中进行详述。

[0079] 在一些实施例中,该全局序列号生成集群102可以是物理独立的,也可以和分布式协调系统104(例如ZooKeeper)合并到一起。

[0080] 其中,分布式存储集群103可以包括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节点设备,节点设备与协调节点设备的划分是针对不同事务而言的,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至少一个节点设备,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中节点设备或协调节点设备的数量进行具体限定。

[0081] 由于本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XA(eXtended Architecture,X/Open组织分布式事务规范)/2PC(Two-Phase Commit,二阶段提交)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,协调节点设备用于充当2PC算法中的协调者,而该协调节点设备所对

应的各个节点设备用于充当2PC算法中的参与者。

[0082] 可选地，每个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(也即是为一主多备集群)，如图1所示，以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，可选地，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物理独立的，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)，此外还包括两个备数据库及代理设备(备database+agent，简称备DB+agent)。

[0083] 在一个示例性场景中，每个节点设备或协调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SET(集合)，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，可选地，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。

[0084] 分布式协调系统104可以用于对网关服务器101、全局序列号生成集群102或者分布式存储集群103中至少一项进行管理，可选地，技术人员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，技术人员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。

[0085] 上述图1仅是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大表，这个大表中存储的数据通过主键标识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大表的水平切分。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，这种系统也可以形象地称为具有“分库分表”的架构。

[0086] 在上述分布式数据库系统中，已经基于XA/2PC算法实现了写操作时数据的原子性和一致性，而读操作的数据一致性问题，需要通过构造一个轻量的、去中心化的分布式事务处理机制来改善，从技术的角度来看，分布分表架构缺乏一个全局事务管理器，也就缺乏分布式事务处理能力，通过构造上述轻量的、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，并且保证分布式数据库系统简单易推广、事务处理效率更高，必将对传统并发控制方式所设计的分布式数据库架构产生极大冲击，具体的分布式事务处理机制将在下个实施例中进行详述。

[0087] 在一些实施例中，本申请实施例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。

[0088] 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。

[0089] 上述实施环境中所涉及的网关服务器101、全局序列号生成集群102、分布式存储集群103(包括节点设备或协调节点设备)以及分布式协调系统104均可以作为区块链系统中的节点设备,使得本申请实施例提供的事务执行方法可以搭建在区块链系统中。

[0090] 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、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。

[0091] 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,区块中还可以包括有区块生成时的时间戳等信息,比如还可以包括本申请实施例提供的事务状态信息。

[0092] 本申请实施例可以应用于上述实施环境中的分布式数据库系统或上述基于区块链技术的数据库系统,通过设计一种新型的允许“写写冲突”发生的并发控制算法,能够提高分布式事务的并发控制效率,从而能够在分布式数据库系统(SQL、NoSQL、NewSQL,关系型、非关系型)、分布式大数据处理系统等大型分布式事务型系统中,提升系统内事务执行的整体效率,使得分布式事务处理机制能够适应当前主流的系统架构,从而规避系统的性能瓶颈,提升数据处理系统的性能,此外,还可以保证分布式数据库系统中较为严格的可串行化隔离级别,且不影响原有隔离级别的正常使用,较大程度的防止数据异常的发生。

[0093] 图2是本申请实施例提供的一种事务执行方法的交互流程图。参见图2,该实施例应用于数据库系统,该数据库系统包括节点设备和协调节点设备,具体可以包括下述步骤:

[0094] 200、协调节点设备创建目标事务的读写集合,向目标事务所涉及的至少一个节点设备发送执行请求。

[0095] 其中,目标事务可以是全局事务,也可以是局部事务,本申请实施例以目标事务为全局事务为例进行说明。

[0096] 其中,目标事务的读写集合包括目标事务的读集或者目标事务的写集中至少一项,目标事务的读集是指目标事务所涉及读取的数据项集合,目标事务的写集是指目标事务所涉及写入的数据项集合,需要说明的是,由于在对数据项执行写入操作时,首先要将数据项读取到读集中,因此目标事务的写集在本质上是目标事务的读集的一个子集合。

[0097] 在本申请实施例中,仅以协调节点设备(coordinator)为目标事务的发起节点、至少一个节点设备(participants,或cohort)为目标事务所涉及的节点设备(也称为参与节点设备)为例进行说明,可选地,除了目标事务的发起节点之外,协调节点设备也可以是上

述实施环境中的网关服务器，还可以是分布式存储集群中的任一个节点设备，该至少一个节点设备可以是目标事务或者目标事务的并发事务中至少一项所涉及的节点设备，还可以是分布式存储集群中的所有节点设备，本申请实施例不对协调节点设备与至少一个节点设备的数量和类型进行具体限定。

[0098] 需要说明的是，当协调节点设备为目标事务的发起节点时，由于不同的目标事务通常具有不同的发起节点，因此对不同的目标事务而言协调节点设备或者至少一个节点设备并非是固定不变的，也即是说，同一节点设备有可能对一些目标事务而言属于协调节点设备，对另一些目标事务而言属于至少一个节点设备中的一个节点设备。

[0099] 本申请实施例所涉及的数据库系统，可以是上述实施环境中提到的基于事实数据库的分布式系统，还可以是其他的关系型或者非关系型的数据库系统，本申请实施例不对数据库系统的类型进行具体限定。

[0100] 在上述步骤200中，协调节点设备在数据库系统启动时，可以向操作系统申请一块内存空间，该内存空间用于进行至少一个事务的读写集合的维护，当目标事务开始执行时，协调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写集合，此时在协调节点设备上创建了目标事务的读写集合，相当于对目标事务进行初始化操作。在后续至少一个节点设备分别执行目标事务时，协调节点设备将至少一个节点设备中局部的读写集合同步到自身全局的读写集合中，从而能够进行集中式的冲突验证，识别出系统内与目标事务存在冲突的并发事务。

[0101] 在一些实施例中，协调节点设备还可以在初始化操作中申请目标事务的开始序列号，可选地，协调节点设备可以调用获取max_commit_gts的接口，向全局序列号生成集群发送用于获取max_commit_gts的第一获取请求，全局序列号生成集群响应于该第一获取请求，确定当前时刻数据库系统内已全局提交(或已全局回滚)的最大事务的提交序列号max_commit_gts，向协调节点设备发送max_commit_gts，协调节点设备接收max_commit_gts之后，将max_commit_gts作为该目标事务的开始序列号，若将目标事务记为T，那么目标事务的开始序列号T.start_tn＝max_commit_gts。

[0102] 可选地，协调节点设备可以将T.start_tn封装在执行请求中发送至该至少一个节点设备，T.start_tn可以采用物理时钟、逻辑时钟或者混合物理时钟中任一项方式生成，例如，T.start_tn的数据结构可以如上述实施环境中所介绍的由八字节组成，前44位为物理时间戳值，后20位为在某一毫秒内的单调递增计数，这里不做赘述。

[0103] 201、至少一个节点设备响应于该执行请求，从目标事务所涉及读取的至少一个数据项中，确定相对于目标事务可见的目标数据项，基于目标数据项执行目标事务，将目标数据项发送至协调节点设备。

[0104] 在上述过程中，对至少一个节点设备中任一节点设备，该节点设备响应于目标事务的执行请求，执行目标事务的事务逻辑中包含的读写操作，在执行过程中，需要针对数据项进行可见性判断，确定相对于目标事务可见的目标数据项，将目标事务所涉及操作(或读或写)的目标数据项发送至协调节点设备，以便于协调节点设备维护目标事务的全局的读写集合。

[0105] 若事务逻辑包括查询操作(读取操作)，节点设备需要读取该目标事务所涉及读取的数据项，若事务逻辑包括更新或删除操作(写入操作)，节点设备仍然需要读取目标事务

所涉及更新或删除的数据项，也即是说，判断数据项的可见性本质上是判断数据项的读取可见性，因此在上述步骤201中，仅以数据读取操作为例进行说明，下面将对数据项的读取可见性判断算法进行介绍。

[0106] 在一些实施例中，对该至少一个数据项中任一数据项，该数据项所在的节点设备可以通过下述子步骤来判断该数据项是否为目标数据项(也即是判断该数据项是否相对于目标事务可见)：

[0107] 2011、节点设备基于生成该数据项的事务的全局事务标识，查询该事务的事务状态信息。

[0108] 对该至少一个数据项中任一数据项，节点设备可以在该数据项的数据结构(具体的数据结构在下述步骤209中进行详述)中获取到产生该数据项的事务的全局事务标识，进一步地，节点设备会对每个事务均会维护一个事务状态信息，并将事务的全局事务标识与事务状态信息进行关联存储，因此，节点设备可以基于产生该数据项的事务的全局事务标识，查询产生该数据项的事务的事务状态信息，但由于事务状态信息是在事务执行完毕之后才能够确定的，而该事务有可能在当前时刻并未执行完毕，因此有可能会查询不到该事务的事务状态。

[0109] 其中，该事务状态信息用于表示与事务在冲突验证阶段的检测结果相对应的执行状态，例如，该事务状态信息可以包括已提交状态(committed)或者已回滚状态(aborted)中至少一项，在一些实施例中，该事务状态信息还可以包括正在执行状态(running)，具体如何确定检测结果以及如何确定事务状态信息将在进行下述步骤206-207详述，这里不做赘述。

[0110] 2012、若查询不到该事务的事务状态信息，且该事务的全局事务标识与该目标事务的全局事务标识不相等，节点设备确定该数据项不为目标数据项。

[0111] 在上述过程中，由于目标事务只有在执行完毕之后才能获取到事务状态信息，因此，若查询不到事务状态信息，说明产生该数据项的事务尚未完成，但由于目标事务自身产生的数据项相对于自身是可见的，因此还需要保证该事务的全局事务标识与目标事务的全局事务标识不相等，也即是保证该事务不为目标事务的基础上查询不到该事务的事务状态信息，那么该数据项一定不是目标数据项。

[0112] 为了方便描述，假设用T表示目标事务，用T.tid表示目标事务的全局事务标识，用v.tid表示产生数据项v的事务的全局事务标识(也即产生版本v的事务号)，那么节点设备获取到数据项v之后，将数据项v的数据结构中事务ID字段获取为v.tid，通过v.tid来查询与v.tid对应存储的事务状态信息，若查询不到与v.tid对应的事务状态信息，且v.tid不等于T.tid，说明该事务还未完成，则确定数据项v不可见，数据项v不为目标数据项，根据数据项v的roll_ptr指针链接到该数据项的前一数据项(也即是前一版本)v-1，继续判断前一数据项v-1的可见性。

[0113] 2013、若查询得到该事务的事务状态信息，节点设备基于该事务的事务状态信息以及该事务的提交序列号，确定该数据项是否为目标数据项。

[0114] 在上述过程中，若获取到上述事务的全局事务标识对应的事务状态信息，由于事务的事务状态信息与事务的提交序列号也是互相关联存储的，因此节点设备可以根据事务的事务状态信息查询得到对应的提交序列号，从而能够基于该事务的事务状态信息以及提

交序列号，判断该数据项是否相对于目标数据项可见，确定该数据项是否为目标数据项。

[0115] 在一些实施例中，针对事实数据库的数据读取操作而言，按照发起用户类型的不同，可以将数据读取操作划分为常规读操作和事实读操作，相应地，针对不同类型的读操作，用于判断数据项是否可读的可见性判断算法可以分为两种，分别是常规读可见性判断算法以及事实读可见性判断算法。

[0116] 对于常规读操作而言，被成功提交的数据项可读，被回滚的数据项不可读，对应于上述常规读可见性判断算法；对于事实读操作而言，事实数据库中特殊读命名为事实读操作，是指具备超级用户身份并且为系统管理员时发起的读操作，此时允许超级用户读被回滚的数据项，对应于上述事实读可见性判断算法，下面进行分类讨论：

[0117] 一、常规读可见性判断算法

[0118] 针对常规读操作，节点设备可以通过下述方式来判断数据项的可见性：若该事务的全局事务标识等于该目标事务的全局事务标识，或该事务的事务状态信息为已提交状态且该事务的提交序列号小于或等于该目标事务的开始序列号，节点设备确定该数据项为目标数据项；否则，节点设备确定该数据项不为目标数据项。

[0119] 为了方便描述，假设用T表示目标事务，用T.tid表示目标事务的全局事务标识，用T.start_tn表示目标事务的开始序列号，用v.tid表示产生数据项v的事务的全局事务标识(也即产生版本v的事务号)，用v.gts表示产生数据项v的事务的提交序列号，用v.status表示产生数据项v的事务的事务状态信息，那么可以根据上述各参数判断数据项v的可见性，若满足如下条件中至少一项，数据项v相对于目标事务可见：1)v.tid＝T.tid，表示该事务为目标事务时目标事务产生的数据项相对于自身可见，数据项v为目标数据项；2)v.status＝committed且v.gts≤T.start_tn，表示最近已提交的数据项相对于目标事务可见，数据项v为目标数据项。反之，若上述两个条件均不满足，数据项v相对于目标事务不可见，数据项v不是目标数据项。

[0120] 二、事实读可见性判断算法

[0121] 针对事实读操作，节点设备可以通过下述方式来判断数据项的可见性：若该事务的全局事务标识等于该目标事务的全局事务标识，或该事务的事务状态信息为已提交状态且该事务的提交序列号小于或等于该目标事务的开始序列号，或该事务的事务状态信息为已回滚状态且该事务的提交序列号小于或等于该目标事务的开始序列号，确定该数据项为目标数据项；否则，确定该数据项不为目标数据项。

[0122] 为了方便描述，假设用T表示目标事务，用T.tid表示目标事务的全局事务标识，用T.start_tn表示目标事务的开始序列号，用v.tid表示产生数据项v的事务的全局事务标识(也即产生版本v的事务号)，用v.gts表示产生数据项v的事务的提交序列号，用v.status表示产生数据项v的事务的事务状态信息，那么可以根据上述各参数判断数据项v的可见性，若满足如下条件中至少一项，数据项v相对于目标事务可见：1)v.tid＝T.tid，表示该事务为目标事务时目标事务产生的数据项相对于自身可见，数据项v为目标数据项；2)v.status＝committed且v.gts≤T.start_tn，表示最近已提交的数据项相对于目标事务可见，数据项v为目标数据项；3)v.status＝aborted且v.gts≤T.start_tn，表示最近已回滚的数据项相对于目标事务可见，数据项v为目标数据项。反之，若上述三个条件均不满足，数据项v相对于目标事务不可见，数据项v不是目标数据项。

[0123] 上述步骤2011-2013示出了节点设备判断任一数据项是否为目标数据项的过程，节点设备重复执行上述步骤2011-2013，从而能够判断出至少一个数据项中的所有目标数据项，进而基于目标数据项执行目标事务，将目标数据项同步到协调节点设备。

[0124] 需要说明的是，若目标事务涉及事实读操作，在目标事务执行完毕返回给用户时，可以将事务状态信息v.status拼在数据项上，从而标识出本数据项是提交事务产生的还是回滚事务产生的。

[0125] 在一些实施例中，节点设备还可以为目标事务分配本地事务标识，将该本地事务标识同步至协调节点设备，使得协调节点设备能够在哈希表中关联目标事务的全局事务标识和本地事务标识。

[0126] 202、协调节点设备将目标数据项存入目标事务的读集中。

[0127] 在上述过程中，协调节点设备接收该至少一个节点设备发送的目标数据项之后，若事务逻辑包括查询操作(读取操作)，协调节点设备可以将目标事务所涉及读取的目标数据项存入到目标事务的读集中，可选地，若事务逻辑包括更新或删除操作(写入操作)，节点设备可以将目标事务所涉及更新或删除的目标数据项分别存入到目标事务的读集以及写集中。

[0128] 需要说明的是，若协调节点设备上也存储有目标事务所涉及读取的至少一个数据项，协调节点设备也可以执行与至少一个节点设备类似的操作，从该至少一个数据项中确定相对于目标事务可见的目标数据项，然后直接将该目标数据项存入目标事务的读集中。

[0129] 203、协调节点设备向至少一个节点设备发送准备提交指令。

[0130] 在至少一个节点设备将目标事务中所有读写操作都完成后，目标事务进入2PC算法内的准备提交(prepare)阶段，协调节点设备与目标事务所涉及的至少一个节点设备进行通信，通过发送准备提交指令来通知所有相关的节点设备准备提交，该至少一个节点设备接收到准备提交指令之后，执行下述步骤204。

[0131] 204、至少一个节点设备响应于准备提交指令，对目标事务进行资源检查，若检查通过，向协调节点设备发送确认准备信息。

[0132] 在上述过程中，至少一个节点设备接收到准备提交指令后，在本地进行资源检查，当检查无误后，向协调节点设备发送确认准备信息，该确认准备信息可以是一个ACK(acknowledge character，确认字符)否则，若检查存在异常，向协调节点设备发送准备失败信息，该准备失败信息可以是一个错误码。

[0133] 在一个示例性场景中，至少一个节点设备在资源检查时还可以维护局部读写集合，在这种情况下，除了协调节点设备对目标事务维护全局的读写集合之外，至少一个节点设备还可以各自在本地对目标事务维护局部的读写集合，至少一个节点设备可以直接将局部的读写集合同步至协调节点设备，协调节点设备汇总各个节点设备的局部读写集合之后，整理得到全局读写集合。

[0134] 205、若该至少一个节点设备均返回确认准备信息，协调节点设备基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务，该目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段。

[0135] 在上述过程中，在所有的节点设备均返回确认准备信息之后，协调节点设备进入对目标事务的冲突验证阶段，此时的冲突验证算法又称为全局验证算法，协调节点设备可

以调用获取当前序列号的接口，向全局序列号生成集群发送第二获取请求，该第二获取请求用于获取当前时刻的全局序列号，全局序列号生成集群响应于该第二获取请求，基于当前物理时间戳生成一个当前序列号，将该当前序列号发送至协调节点设备，协调节点设备将接收到的当前序列号作为目标事务的提交序列号(commit_tn)，需要说明的是，若目标事务最终被确认需要进行全局回滚，此时的提交序列号也可以称为回滚序列号。

[0136] 在获取到提交序列号之后，协调节点设备可以基于目标事务的开始序列号以及提交序列号来确定目标时间段，从而将提交序列号位于该目标时间段内的其他事务确定为至少一个活跃事务。否则，若存在任一个节点设备返回准备失败信息，将目标事务的事务状态信息置为已回滚状态，全局回滚该目标事务，存储该目标事务的事务状态信息。

[0137] 在上述过程中，假设用start_tn表示开始序列号，用commit_tn表示提交序列号，对任一事务而言，若该事务的提交序列号大于目标事务的开始序列号且小于或等于目标事务的提交序列号，说明该事务的commit_tn位于目标事务的(start_tn,commit_tn]范围内，将该事务确定为一个活跃事务。

[0138] 206、协调节点设备检测目标事务的读集是否与至少一个活跃事务的写集存在交集，得到检测结果，该读集为该目标事务所读取的数据项集合，该写集为该至少一个活跃事务所写入的数据项集合。

[0139] 由于在对数据项执行写入操作时，首先要将数据项读取到读集中，因此目标事务的写集在本质上是目标事务的读集的一个子集合，也即是说，目标事务的读集包括了目标事务的写集，因此在上述步骤206中，判断目标事务与至少一个活跃事务的读写集合是否存在交集时，可以仅判断目标事务的读集是否与至少一个活跃事务的写集存在交集，而无需判断目标事务的写集是否与至少一个活跃事务的写集存在交集。

[0140] 在上述过程中，协调节点设备可以遍历目标事务的读集以及至少一个活跃事务的写集，从而确定该读集与该写集中是否包括相同数据项，若该读集与该写集中不包括任何相同数据项，将检测结果确定为该读集与该写集不存在交集，否则，若该读集与该写集中包括任一相同数据项，将检测结果确定为该读集与该写集存在交集。

[0141] 207、协调节点设备根据该检测结果，确定目标事务的事务状态信息。

[0142] 在上述过程中，若该检测结果为该读集与该写集存在交集，协调节点设备可以将该事务状态信息(status)确定为已回滚状态(aborted)；或，若该检测结构为该读集与该写集不存在交集，协调节点设备可以将该事务状态信息(status)确定为已提交状态(committed)。

[0143] 上述步骤205-207中，相当于提供了一种在协调节点设备上进行的全局验证算法，全局验证算法的执行逻辑可以使用下述伪代码进行表示：

[0144] T.commit-tn:＝tnc;

[0145] T.status:＝committing;

[0146] for t from start-tn+1to commit-tn do//判断本事务(目标事务)读集和其他并发事务(至少一个活跃事务)的写集是否相交(存在交集)

[0147] **if** (write set of transaction with commit_tn=t intersects read set or write set)

```
            then T.status := aborted;
        else
            then T.status := committed;
        goto write-phase; // 执行写入阶段
```

[0148]

[0149] 208、协调节点设备向至少一个节点设备发送对目标事务的全局写入指令。

[0150] 在相关技术中，并发控制算法需要通过临界区的方式来规避验证，以避免在验证和提交阶段发生新的写写、读写冲突，临界区是指每次仅允许一个事务进行验证和写入阶段，导致数据库系统的并发度较低，并发事务处理性能差，事务执行效率低。

[0151] 可选地，上述全局写入指令可以包括提交(commit)指令或者回滚(abort)指令中至少一项，若事务状态信息为committed，则协调节点设备对应发送commit指令，若事务状态信息为aborted，则协调节点设备对应发送abort指令。

[0152] 在上述过程中，不论目标事务的检测结果是否为存在交集，也即是不论事务状态信息(T.status)为已提交状态(committed)还是已回滚状态(aborted)，目标事务都将继续进入写入阶段，从而去除了临界区的使用，通过允许写写冲突发生的方式，提供了允许写写冲突发生的并发控制算法，这种新型的分布式事务处理方案，极大地提升了数据库系统的并发度，优化了数据库系统的事务并发控制性能，并且在分布式数据库系统中具有较好地适用性。

[0153] 209、至少一个节点设备响应于该全局写入指令，对目标事务进行全局写入。

[0154] 对该至少一个节点设备中任一节点设备，协调节点设备通过全局写入指令，通知该节点设备进行数据写入，该节点设备上会提交目标事务对应的局部事务，清理局部事务的上下文信息(比如局部事务的读写集合)，将原本维护在缓存中的待写入数据进行数据落盘操作，当每个节点设备均完成数据落盘时，目标事务完成写入阶段，执行下述步骤210。

[0155] 在一些实施例中，由于存在写写冲突的数据项需要被持久化地保存下来，那么本申请实施例提供的事务处理机制有可能会增大系统的I/O(Input/Output，输入/输出)开销，为了降低额外的I/O开销对系统性能的影响，本申请实施例分别针对磁盘型存储引擎和键值(key-value)型存储引擎提出了各自的多版本数据存储机制，下面进行详述：

[0156] 一、Key-value模式存储的数据库系统

[0157] 在上述Key-value模式存储的数据库系统中，数据项是按照键(key)有序存放的，因此，每次内存与磁盘的交互单位不再固定为一个数据页面，基于这一特点，数据项的存储格式可以更为灵活，这种按键序存放的数据项形式，使得数据项的写入速度大大提高(采用了append方式，减少了传统数据库系统随机写的写放大问题)，可以规避由于写入数据量增加而导致的负载问题。

[0158] 对于Key-value模式存储引擎，每个数据项(每个版本)均可以通过一条kv(键值对)的形式进行存放。本申请实施例提供的kv数据结构可以如表1所示，请参考表1，key(键)中主要包括表空间ID(identification，标识)，主键标识PK、事务ID、版本号等，从而能够通过key唯一标识一个数据项(一个版本)，可选地，节点设备还可以在key中维护一个写写冲突标志位(表中未示出)，用于标识是否存在写写冲突，此外，value中包含数据项(版本)的其余属性和属性值，例如，对版本V0进行更新操作，将姓名(Name)为James的余额(Bal)从50

更新到100,产生版本V1,那么在版本V1中,value(值)为Name:James,Bal:100。

[0159] 表1

[0160]

	key（键）					value（值）
版本	表空间ID	PK	事务ID	版本号	…	列：列值
V0	tbl	a	100	0	…	Name: James， Bal: 50
V1	tbl	a	120	1	…	Name: James， Bal: 100
V2	tbl	a	130	2	…	Name: James， Bal: 200

[0161] 在一些实施例中,若该目标事务的事务状态信息为已回滚状态,在该目标事务所生成的数据项中可以仅记录被修改的属性值。也即是说,节点设备对回滚事务所产生的数据项,仅维护其被修改的属性值,能够节省多版本数据的存储开销,优化多版本数据存储机制。

[0162] 请参考表2,示出了本申请实施例提供的一种kv数据结构,假设存在某一事务将版本V1修改为版本V3,该事务要求将James的余额修改为300,但该事务最终被回滚,则V3的value值可以仅记录{Bal:300},而{Name:James}则被忽略,在高并发场景下,由于写写冲突大量存在,会导致大量的回滚事务,通过对回滚事务仅记录被修改的属性值,而忽略记录未被修改的属性值,能够大量减少针对回滚事务所产生的数据项的存储开销。

[0163] 表2

[0164]

	key（键）					value（值）
版本	表空间ID	PK	事务ID	版本号	…	列：列值
V0	tbl	a	100	0	…	Name: James， Bal: 50
V1	tbl	a	120	1	…	Bal: 100
V2	tbl	a	130	2	…	Bal: 200

[0165] 在相关技术中,事务处理流程需要与存储紧密绑定在一起,比如,Undo(撤销)数据结构维护在存储引擎中,而事务引擎在事务处理流程中需要对Undo数据结构进行操作,导致事务引擎与存储引擎不能做到解耦。

[0166] 在上述多版本数据存储机制中,事务处理流程能够与存储过程相解耦,事务处理的流程(即解决和发现冲突的方法)与数据的读写操作得到分离,事务处理流程中不必在缓存中再维护额外的数据结构,比如无需再度维护Undo结构,使得事务引擎与存储引擎之间的交互较为简单。

[0167] 在上述过程中,通过采用key-value模式进行事务状态管理,在数据页面中管理各个数据项和各个事务状态信息(关于事务状态信息的管理请参见下述步骤210),能够去除事务处理流程中对Undo数据结构的访问,简化了事务处理流程和存储流程。

[0168] 二、页面模式存储的数据库系统

[0169] 由于关系模型普遍采用段页式的数据维护结构,因此,针对基于关系模型的数据库系统(也即是页面模式存储的数据库系统)中数据项来说,该数据项的数据结构可以如图3所示,请参考图3,该数据项的数据结构300可以包括:记录头、记录指针、主键标识、事务ID

(transaction identification,也即是TID,为产生该数据项的事务的事务标识,包括全局事务标识和本地事务标识,通常为6个字节)、roll_ptr(指向该主键标识所对应的前一数据项的指针,通常为7个字节)以及主键外的其他非空字段,这种数据结构在例如MySQL/InnoDB中被采用。

[0170] 基于上述数据结构的数据项,在某一数据项被修改之后,旧版本数据项与新版本数据项统一存放在数据页中,并基于指针进行链接,具有相同主键标识的不同数据项(俗称为一个数据项的多个版本)均管理在数据页面中,也即是说,一个数据页面中可能存放着某一数据项的多个版本(相当于某一主键标识对应的全态数据集)。在这种存储机制下,数据项或者元组形式的事务状态信息(在下述步骤210中进行详述)均可以采用页面模式进行维护,从而能够复用数据页面的相关逻辑,减少额外的代码开发开销。

[0171] 在相关技术中,传统的多版本数据是基于Undo结构进行存储的,某一数据项被修改后,旧版本数据项会被存在Undo结构中,在Undo结构中,同一事务产生的Undo记录需要通过链表进行维护,每次进行插入、更新或者删除操作时,均需要记录一条新的Undo记录,并且Un记录中需要保护数据信息、地址指针等大量的属性信息,导致Undo结构需要经过大量的代码优化,Undo结构的开发成本和维护成本巨大,影响了数据库系统的整体性能。

[0172] 而在本申请实施例中,通过分别对key-value模式和页面模式存储的数据库系统进行数据结构的优化,能够在上述两种数据库系统内均消除Undo等复杂数据结构的维护,使得并发控制逻辑变得简单,极大地降低了程序实现所带来的性能内耗。

[0173] 210、在全局写入该目标事务时,协调节点设备将目标事务的事务状态信息、目标事务的全局事务标识以及目标事务的提交序列号进行关联存储。

[0174] 在上述步骤210中,协调节点设备在目标事务的写入阶段中,存储目标事务的事务状态信息,而通过将事务状态信息、全局事务标识以及提交序列号关联存储,能够方便后续针对目标事务所产生数据项进行可见性判断的流程,比如,方便了与上述步骤2011类似的基于全局事务标识查询事务状态信息,还方便了与上述步骤2012类似的基于事务状态信息查询提交序列号,这里不做一一赘述。

[0175] 在上述过程中,通过对目标事务的事务状态信息进行维护,能够维护目标事务的执行结果(是被提交还是被回滚),从而实现了对事实数据库中事实数据进行维护。在本申请实施例中分别针对磁盘型存储引擎和键值(key-value)型存储引擎提出了各自的事务状态信息维护方式,下面进行详述:

[0176] 一、Key-value模式存储的数据库系统

[0177] 在上述Key-value模式存储的数据库系统中,协调节点设备可以基于键值对的形式进行关联存储,可选地,协调节点设备以该目标事务的全局事务标识作为键(key),以目标事务的该事务状态信息以及该目标事务的提交序列号作为值(value),从而对每个目标事务可以构建一个键值对,通过该键值对来维护目标事务的事务状态信息。

[0178] 在一个示例性场景中,对任一目标事务T,该目标事务的全局事务标识、事务状态信息以及提交序列号可以视为一个事务状态元组{TID,Status,Gts},该事务状态元组是一个三元组,其中,TID表示事务ID(全局事务标识),对系统内的所有事务进行唯一表示,TID在事务初始化时进行分配;Gts表示事务全局提交或者全局回滚完毕时的序列号(提交序列号),由全局序列号生成集群提供,保证全局单调递增有序;Status表示事务的最终状态(事

务状态信息),可以采用committed代表事务已经全局提交完毕,可以采用aborted代表事务已经全局回滚完毕,可选地,Status可以是一个比特(bit)位,比如bit位为1时代表committed,bit位为0时代表aborted,可选地,在一些数据库系统中还可以维护一个正在执行状态(running,表示事务正在运行中),此时Status可以是两个bit位,比如bit位为11时代表committed,bit位为00时代表aborted,bit位为01表示running,本申请实施例不对Status的类型进行具体限定。

[0179] 在Key-value模式存储的数据库系统中,每一条事务状态元组可以作为一条kv(键值对)存入key-value数据库中存储,其中,TID作为key,Status和Gts作为value,因此不论是数据项还是事务状态元组,在存储时均作为一条kv存在。

[0180] 需要说明的是,事务状态元组的记录是在目标事务全局提交或全局回滚时进行的,当目标事务产生的数据项写入时,同时将该目标事务对应的事务状态元组以kv形式写入存储引擎中,进行持久化维护。

[0181] 二、页面模式存储的数据库系统

[0182] 在上述页面模式存储的数据库系统中,协调节点设备可以基于事务状态元组的形式进行关联存储,其中,该事务状态元组包括该目标事务的事务状态信息、该目标事务的全局事务标识以及该目标事务的提交序列号。

[0183] 也即是说,在上述页面模式存储的数据库系统中,采用段页式管理事务状态元组(也称为事务状态日志),内外存之间交换的最小单元是一个数据页面,协调节点设备可以在内存中开辟专门的事务状态缓冲区,在目标事务全局提交或全局回滚时,随着目标事务产生的数据项的写入操作,同时将目标事务的事务状态元组(三元组){TID,Status,Gts}写入到事务状态缓冲区中。

[0184] 在一些实施例中,协调节点设备可以保证事务状态元组在数据页面中按照TID从小到大进行有序存放,使得在读取事务状态元组时可以通过TID换算出数据页面ID和对应的页内偏移量,从而能够节省页面扫描的开销,提升数据读取操作的性能。

[0185] 在上述过程中,不仅创新性地提出了一种允许写写冲突发生的分布式事务处理方案,使得事务处理流程与数据读写流程解耦,而且提出了允许写写冲突的并发控制算法(Write-allow Concurrency Control,W-CC算法),能够获取更好的事务并发控制性能,在分布式数据库系统中具有良好的适用性,此外,实现了一种事实数据库,能够全量存储数据库系统中真实发生的事件相关数据(事实数据),使得一切业务操作能够被回溯。

[0186] 上述所有可选技术方案,可以采用任意结合形成本公开的可选实施例,在此不再一一赘述。

[0187] 本申请实施例提供的方法,通过基于目标时间段,获取提交时刻处于该目标时间段内的至少一个活跃事务,该目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段,检测该目标事务的读集是否与该至少一个活跃事务的写集存在交集,得到检测结果,该读集为该目标事务所读取的数据项集合,该写集为该至少一个活跃事务所写入的数据项集合,根据该检测结果,确定该目标事务的事务状态信息,该事务状态信息用于表示与该检测结果对应的执行状态,在全局写入该目标事务时,存储该目标事务的事务状态信息,可以看出,仍然会对写写冲突进行检测,但不论检测结果表示是否存在冲突,都不影响目标事务的全局写入,而是为目标事务配置与检测结果对应的事务状态信息,在写入阶段同时存储

该事务状态信息，提升了数据库系统中事务执行的并发度，提升了数据库系统中的事务执行效率。

[0188] 此外，在上述事务处理机制下，提供了对应的数据项可见性判断算法，能够基于存储的事务状态信息来判断数据项的可见性，保证由于写写冲突导致回滚的事务对于普通用户是不可见的，当然，在一些场景下，具有超级用户权限以及身份为系统管理员的用户，能够基于事实读操作来读取回滚事务。进一步地，分别提出了基于key-value以及基于关系模型(页面模式存储)的新型多版本数据存储机制，消除了Undo等复杂数据结构的维护，使得并发控制逻辑变得简单，极大地降低了程序实现所带来的性能内耗。

[0189] 为了能够更加直观地说明本申请实施例的事务执行过程，图4是本申请实施例提供的一种并发控制方法的流程图，请参考图4，在基于本申请实施例的事务处理机制进行事务并发控制时，事务的执行可以划分为读取阶段401、验证阶段402以及写入节点403，下面进行详述：

[0190] 一、读取阶段401

[0191] 协调节点设备首先创建目标事务的读写集合(全局读写集合)，然后向节点设备1和节点设备2发送执行请求，节点设备1和节点设备2分别执行事务逻辑中包含的读写操作。

[0192] 以节点设备1执行读事务为例，协调节点设备向节点设备1发送携带SQL查询语句的执行请求，节点设备1基于可见性判断算法获取读事务涉及的数据(目标数据项)，节点设备1向协调节点设备返回查询结果(发送目标数据项)，协调节点设备将该查询结果存入全局读集。以节点设备2执行写事务为例，协调节点设备向节点设备2发送携带SQL更新语句的执行请求，节点设备2将写事务的数据项存入到本地缓存，节点设备2向协调节点设备返回待更新的数据项，协调节点设备将该待更新的数据项存入全局写集。

[0193] 二、验证阶段402

[0194] 对任一目标事务，协调节点设备获取至少一个节点设备发送的目标事务涉及读取的至少一个数据项，对该至少一个数据项进行全局验证，检测目标事务的全局读集是否与至少一个活跃事务的全局写集存在交集，得到检测结果，不管检测结果表示是否存在交集，目标事务均会进入写入阶段403。

[0195] 三、写入阶段403

[0196] 协调节点设备基于检测结果，确定目标事务的事务状态信息，记录目标事务全局写入，向至少一个节点设备发送全局写入指令，根据事务状态信息的不同，全局写入指令也不尽相同，比如若事务状态信息为committed，全局写入指令为commit指令，若事务状态信息为aborted，全局写入指令为abort指令，各个节点设备提交该目标事务对应的局部事务，清理事务的上下文信息，将原本维护在缓存中的待写入数据进行数据落盘操作，向协调节点设备返回确认提交信息，从而完成目标事务的全局写入。当所有节点设备均写入完毕时，协调节点设备清空全局读写集合，持久化地维护目标事务的事务状态信息。

[0197] 上述实施例提供一种允许写写冲突的并发控制算法，用于对并发事务进行管理，负责协调并发事务的执行，防止事务之间相互干扰而得到错误的结果。下面针对传统的并发控制算法与本申请实施例提供的允许写写冲突的并发控制算法进行对比分析。

[0198] 在传统的并发控制算法中，不允许写写冲突发生，当某一写写冲突不被并发控制算法允许时，并发控制算法会采用回滚或延迟的机制，阻塞发生写写冲突的事务提交。在基

于传统的并发控制算法进行事务处理时，由于在高并发场景下，写写冲突普遍存在，而由于传统的并发控制算法普遍通过回滚或等待的方式对写写冲突进行约束，导致系统内大量事务被回滚，影响系统的并发度以及事务执行效率。尤其是在分布式数据库系统中，回滚或等待的通信代价将会被放大，因此分布式数据库系统内的性能损耗相较于单机数据库系统而言更大，这是由于分布式数据库系统中数据项会被分布在不同的物理节点(节点设备)上，如果发生回滚操作，事务的协调节点设备与相关的所有节点设备进行通信，发送回滚指令，这会带来大量的网络传输开销，每个节点设备上执行的回滚操作也需要进行协调，有可能会因为较大的回滚代价而造成系统不可用。

[0199] 本申请实施例提供的方法，提供一种允许写写冲突的新型并发访问控制算法，在写写冲突发生时，不会导致目标事务的回滚，而是允许目标事务继续执行，但在该机制下，写写冲突仍然会被识别出来，并记录对应的事务状态信息，但不会导致事务的回滚或等待，无需阻塞写写冲突的事务提交，从而构建了以上述事务处理机制为核心的事实数据库系统。并且，由于不需要按照某一确定顺序对事务进行排序之后再一一进入临界区执行写入操作，这样能够避免写写冲突造成的事务回滚影响，可以良好地适用于分布式系统中。

[0200] 在一些实施例中，针对上述各实施例提供的事实数据库系统，还可以向目标用户(具备超级用户身份的系统管理员)开放事实数据管理模式，目标用户可以通过系统参数来设置当前数据库系统是否启用上述事实数据管理模式。

[0201] 在事实数据管理模式下，普通用户无法对事实数据进行清理操作，只有目标用户才具备对处于事实数据管理模式下的数据项进行清理操作的权限，清理操作可以由目标用户手动触发，数据库系统不会主动对处在事实数据管理模式下的数据进行清理，能够将事实数据全量保存下来，由于事实数据的体量较大，可以通过分布式文件系统提供无限存储能力，保证事实数据库系统的性能。

[0202] 需要说明的是，目标用户还可以通过系统参数设置事实数据库的维护粒度，具体包括：1)全局维护粒度。数据库系统中所有的数据表中的所有数据项均会被保留下来，适用于全数据库系统审计等场景。2)表维护粒度。目标用户可以通过指定表名的形式，对部分表应用上述事务处理机制进行事实数据的维护，适用于对部分重点业务进行监控的场景，具有更好的灵活性。

[0203] 在非事实数据管理模式下，对于因写写冲突而产生的回滚数据，可以基于一种垃圾清理机制对这些回滚得到的垃圾数据(无效数据)进行清理，在本申请实施例中针对key-value模式和页面模式分别提供了不同的垃圾清理机制，下面进行详述：

[0204] 一、Key-value模式存储的数据库系统

[0205] 对任一当前写入完毕的目标事务而言，节点设备(或协调节点设备)在存储目标事务所生成的数据项时，若内存的剩余空间小于空间阈值，可以将该内存中的已有数据项转存至磁盘中，将该目标事务所生成的数据项存入该内存。其中，该空间阈值可以是任一大于0的数值，本申请实施例不对空间阈值的取值进行具体限定。

[0206] 在一些实施例中，节点设备(或协调节点设备)在将内存中已有的数据项刷入磁盘时，还可以将该内存中已有的数据项与磁盘中的数据项按照键序进行合并，在合并过程中，若检测到生成任一数据项的事务的事务状态信息为已回滚状态，跳过对该数据项的合并操作。

[0207] 在上述过程中，数据项在kv存储结构中首先在内存中维护，当内存写满之后才会将数据项刷入磁盘，在刷入磁盘时还可以触发合并机制，保证磁盘中的数据项按照键序有序排列，通过跳过对回滚事务所产生数据项的合并操作，能够清理掉回滚事务所产生的数据项。

[0208] 在一个示例性场景中，在磁盘中kv存储引擎可以按照分层方式来排列kv对，图5是本申请实施例提供的一种磁盘分层管理的原理性示意图，请参考图5，当内存中的数据刷入磁盘后，该数据会首先与第1层501先合并，如果第1层501内文件达到一定数量后，则将多余的文件写到下一层也即是第2层502，以此类推。比如，当第1层501数据满之后，向下一层(第2层502)进行扫描，如果发现key范围重叠的数据，则将数据归并之后一起写入到第2层502。每一次内数据key按序存放，保证了层内数据key没有重叠，这样的存储机制使得在查找key范围的时候很方便，可以使用二分法快速查找定位到key范围。

[0209] 基于上述kv存储的分层结构，在数据项按照键序进行数据合并操作时，同时出发垃圾清理操作，操作机制也即是：在内存中的数据刷入磁盘时，扫描当前待合并内存数据块(内存中已有的数据项)，当发现产生该数据项的事务的事务状态信息为已回滚状态(aborted)，不对该数据项执行合并操作，跳过该数据项。这样的垃圾清理机制，仅在数据项进行合并操作时才会被触发，属于一种被动触发的方式，可以减少垃圾清理操作所占用的处理资源。

[0210] 二、页面模式存储的数据库系统

[0211] 在页面模型存储的数据库系统(基于关系模型的数据库系统)中，可以每间隔目标时长，获取当前正在执行的事务中的最小开始序列号，删除提交序列号小于该最小开始序列号且事务状态信息为已回滚状态的事务所操作的数据项。其中，目标时长为任一大于0的数值，本申请实施例不对目标时长的取值进行具体限定。

[0212] 在上述过程中，垃圾清理机制的触发方式为定时触发，在一个示例性场景中，每间隔目标时长，由协调节点设备发起数据清理，协调节点设备向其余所有的协调节点设备通信，获取所有正在执行事务中最小开始序列号(start_tn)，代表当前全部活跃事务中正在使用的最小的读取序列号，将该最小开始序列号记为min，协调节点设备将获取到的min值发送给所有的节点设备，触发各个节点设备各自进行数据清理，节点设备对各个事务的事务状态元组进行扫描，获取到commit_tn＜min且status＝aborted的元组集合U，遍历元组集合U，通过各个事务状态信息对应存储的TID(全局事务标识)，查询到各个回滚事务所操作的数据项，将上述数据项进行清理，当遍历完成后，将元组集合U丢弃，垃圾清理操作完成。

[0213] 上述所有可选技术方案，可以采用任意结合形成本公开的可选实施例，在此不再一一赘述。

[0214] 本申请实施例提供的方法，向目标用户开发了事实数据库系统的管理模式，目标用户可以选择是否开启事实数据管理模式，在事实数据管理模式中全量存储所有的事实数据，系统不主动进行垃圾清理，但目标用户可以手动触发进行垃圾清理，并且在开启事实数据管理模式的情况下，还可以向目标用户提供调整维护粒度的服务，使得目标用户指定全局维护粒度或者表维护粒度，使得事实数据库系统的管理更加个性化、灵活性更高。此外，在非事实数据管理模型下，针对key-value模式存储的数据库系统和页面模式存储的数据

库系统分别提供了垃圾清理机制，能够通过被动触发或者定时触发的逻辑，数据库系统自动进行垃圾清理，及时释放回滚事务所产生的数据项占用的存储空间，提升了数据库系统的性能。

[0215] 图6是本申请实施例提供的一种事务执行装置的结构示意图，请参考图6，该装置包括：

[0216] 获取模块601，用于基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务，该目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段；

[0217] 检测模块602，用于检测该目标事务的读集是否与该至少一个活跃事务的写集存在交集，得到检测结果，该读集为该目标事务所读取的数据项集合，该写集为该至少一个活跃事务所写入的数据项集合；

[0218] 确定模块603，用于根据该检测结果，确定该目标事务的事务状态信息，该事务状态信息用于表示与该检测结果对应的执行状态；

[0219] 存储模块604，用于在全局写入该目标事务时，存储该目标事务的事务状态信息。

[0220] 本申请实施例提供的装置，通过基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务，该目标时间段为从目标事务的开始时刻至提交时刻所确定的时间段，检测该目标事务的读集是否与该至少一个活跃事务的写集存在交集，得到检测结果，该读集为该目标事务所读取的数据项集合，该写集为该至少一个活跃事务所写入的数据项集合，根据该检测结果，确定该目标事务的事务状态信息，该事务状态信息用于表示与该检测结果对应的执行状态，在全局写入该目标事务时，存储该目标事务的事务状态信息，可以看出，仍然会对写写冲突进行检测，但不论检测结果表示是否存在冲突，都不影响目标事务的全局写入，而是为目标事务配置与检测结果对应的事务状态信息，在写入阶段同时存储该事务状态信息，提升了数据库系统中事务执行的并发度，提升了数据库系统中的事务执行效率。

[0221] 在一种可能实施方式中，该确定模块603用于：

[0222] 若该检测结果为该读集与该写集存在交集，将该事务状态信息确定为已回滚状态；或，

[0223] 若该检测结构为该读集与该写集不存在交集，将该事务状态信息确定为已提交状态。

[0224] 在一种可能实施方式中，基于图6的装置组成，该存储模块604包括：

[0225] 存储单元，用于将该目标事务的事务状态信息、该目标事务的全局事务标识以及该目标事务的提交序列号进行关联存储。

[0226] 在一种可能实施方式中，该存储单元用于：

[0227] 基于键值对的形式进行关联存储，其中，以该目标事务的全局事务标识作为键，以目标事务的该事务状态信息以及该目标事务的提交序列号作为值；或，

[0228] 基于事务状态元组的形式进行关联存储，其中，该事务状态元组包括该目标事务的事务状态信息、该目标事务的全局事务标识以及该目标事务的提交序列号。

[0229] 在一种可能实施方式中，该装置还用于：

[0230] 若该目标事务的事务状态信息为已回滚状态，在该目标事务所生成的数据项中仅记录被修改的属性值。

[0231] 在一种可能实施方式中，基于图6的装置组成，该装置还包括：

[0232] 确定存入模块，用于从该目标事务所涉及读取的至少一个数据项中，确定相对于该目标事务可见的目标数据项，将该目标数据项存入该目标事务的读集中。

[0233] 在一种可能实施方式中，基于图6的装置组成，该确定存入模块包括：

[0234] 查询单元，用于对该至少一个数据项中任一数据项，基于生成该数据项的事务的全局事务标识，查询该事务的事务状态信息；

[0235] 确定单元，用于若查询不到该事务的事务状态信息，且该事务的全局事务标识与该目标事务的全局事务标识不相等，确定该数据项不为目标数据项；

[0236] 该确定单元，还用于若查询得到该事务的事务状态信息，基于该事务的事务状态信息以及该事务的提交序列号，确定该数据项是否为目标数据项。

[0237] 在一种可能实施方式中，该确定单元用于：

[0238] 若该事务的全局事务标识等于该目标事务的全局事务标识，或该事务的事务状态信息为已提交状态且该事务的提交序列号小于或等于该目标事务的开始序列号，确定该数据项为目标数据项；否则，确定该数据项不为目标数据项。

[0239] 在一种可能实施方式中，该确定单元用于：

[0240] 若该事务的全局事务标识等于该目标事务的全局事务标识，或该事务的事务状态信息为已提交状态且该事务的提交序列号小于或等于该目标事务的开始序列号，或该事务的事务状态信息为已回滚状态且该事务的提交序列号小于或等于该目标事务的开始序列号，确定该数据项为目标数据项；否则，确定该数据项不为目标数据项。

[0241] 在一种可能实施方式中，该存储模块604还用于：

[0242] 在存储该目标事务所生成的数据项时，若内存的剩余空间小于空间阈值，将该内存中的已有数据项转存至磁盘中，将该目标事务所生成的数据项存入该内存。

[0243] 在一种可能实施方式中，该存储模块604还用于：

[0244] 将该内存中已有的数据项与磁盘中的数据项按照键序进行合并，在合并过程中，若检测到生成任一数据项的事务的事务状态信息为已回滚状态，跳过对该数据项的合并操作。

[0245] 在一种可能实施方式中，该装置还用于：

[0246] 每间隔目标时长，获取当前正在执行的事务中的最小开始序列号；

[0247] 删除提交序列号小于该最小开始序列号且事务状态信息为已回滚状态的事务所操作的数据项。

[0248] 上述所有可选技术方案，可以采用任意结合形成本公开的可选实施例，在此不再一一赘述。

[0249] 需要说明的是：上述实施例提供的事务执行装置在执行事务时，仅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，即将计算机设备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，上述实施例提供的事务执行装置与事务执行方法实施例属于同一构思，其具体实现过程详见事务执行方法实施例，这里不再赘述。

[0250] 图7是本申请实施例提供的一种计算机设备的结构示意图，该计算机设备700可因配置或性能不同而产生比较大的差异，可以包括一个或一个以上处理器(Central

Processing Units,CPU)701和一个或一个以上的存储器702,其中,该存储器702中存储有至少一条程序代码,该至少一条程序代码由该处理器701加载并执行以实现上述各个实施例提供的事务执行方法。当然,该计算机设备700还可以具有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,该计算机设备700还可以包括其他用于实现设备功能的部件,在此不做赘述。

[0251] 在示例性实施例中,还提供了一种计算机可读存储介质,例如包括至少一条程序代码的存储器,上述至少一条程序代码可由终端中的处理器执行以完成上述实施例中事务执行方法。例如,该计算机可读存储介质可以是ROM(Read-Only Memory,只读存储器)、RAM(Random-Access Memory,随机存取存储器)、CD-ROM(Compact Disc Read-Only Memory,只读光盘)、磁带、软盘和光数据存储设备等。

[0252] 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,该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。

[0253] 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。

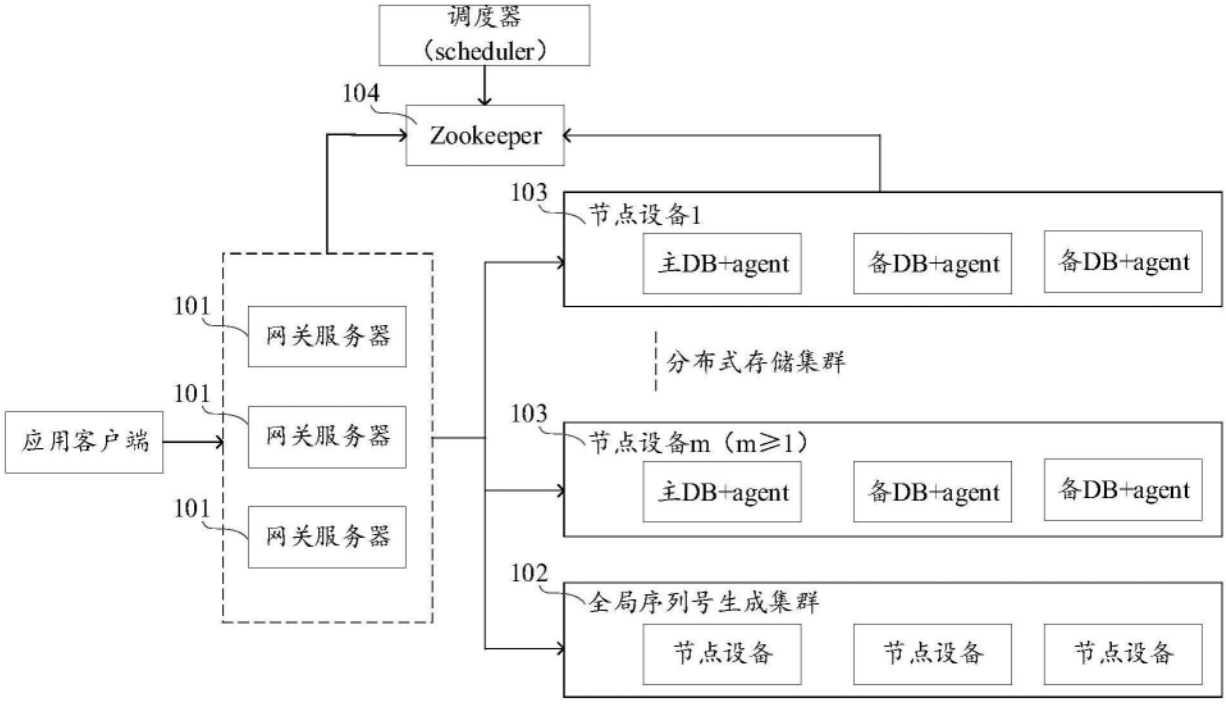

调度器
（scheduler）
104
Zookeeper
103
节点设备1
主DB+agent
备DB+agent
备DB+agent
101
网关服务器
101
网关服务器
101
网关服务器
应用客户端
分布式存储集群
103
节点设备m（m≥1）
主DB+agent
备DB+agent
备DB+agent
102
全局序列号生成集群
节点设备
节点设备
节点设备

图1

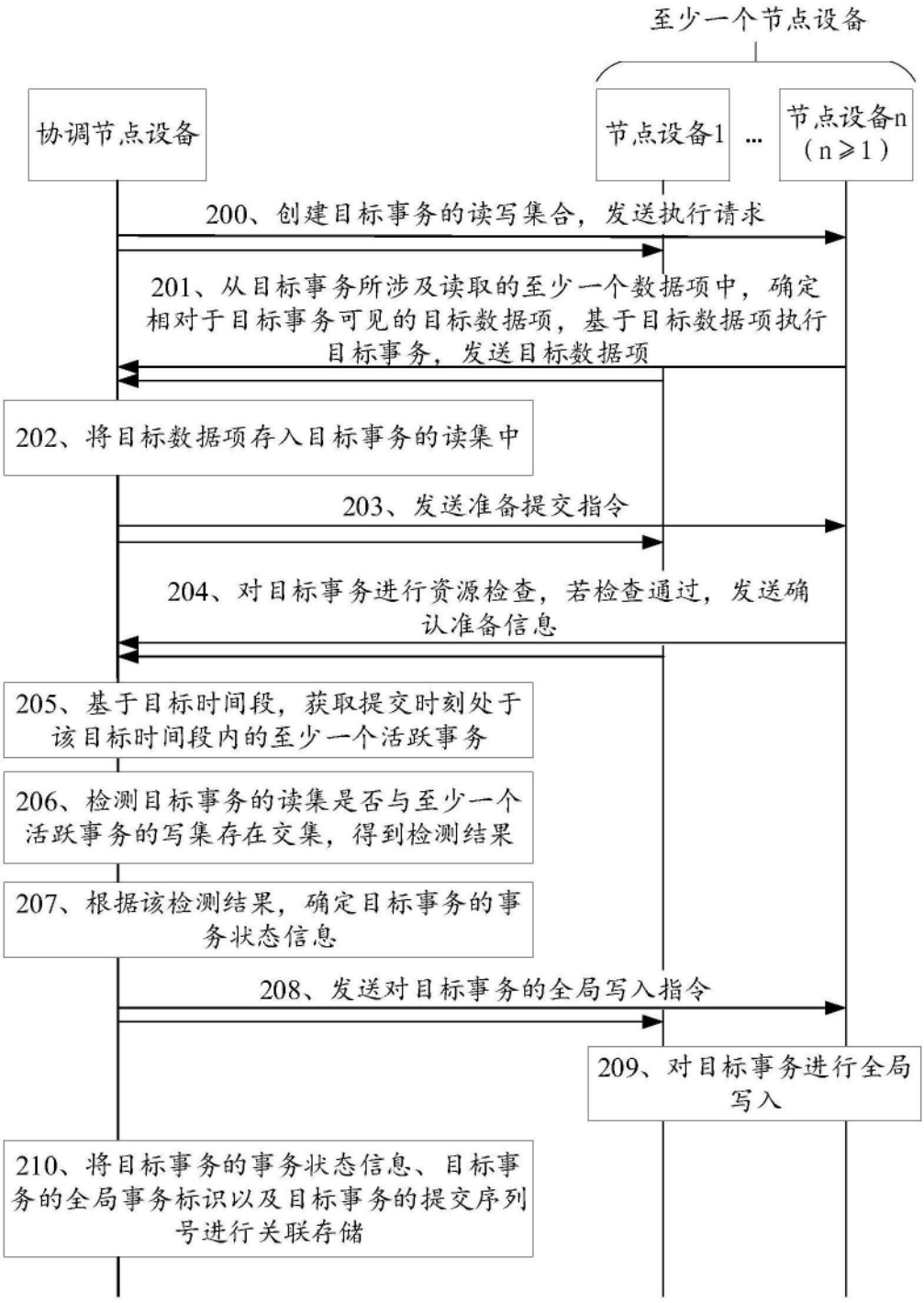

至少一个节点设备
协调节点设备
节点设备1
...
节点设备n
（n≥1）
200、创建目标事务的读写集合，发送执行请求
201、从目标事务所涉及读取的至少一个数据项中，确定相对于目标事务可见的目标数据项，基于目标数据项执行目标事务，发送目标数据项
202、将目标数据项存入目标事务的读集中
203、发送准备提交指令
204、对目标事务进行资源检查，若检查通过，发送确认准备信息
205、基于目标时间段，获取提交时刻处于该目标时间段内的至少一个活跃事务
206、检测目标事务的读集是否与至少一个活跃事务的写集存在交集，得到检测结果
207、根据该检测结果，确定目标事务的事务状态信息
208、发送对目标事务的全局写入指令
209、对目标事务进行全局写入
210、将目标事务的事务状态信息、目标事务的全局事务标识以及目标事务的提交序列号进行关联存储

图2


300
记录头
主键标识
事务ID
（6个字节）
roll_ptr
（7个字节）
主键以外的其他非空字段
记录指针

图3

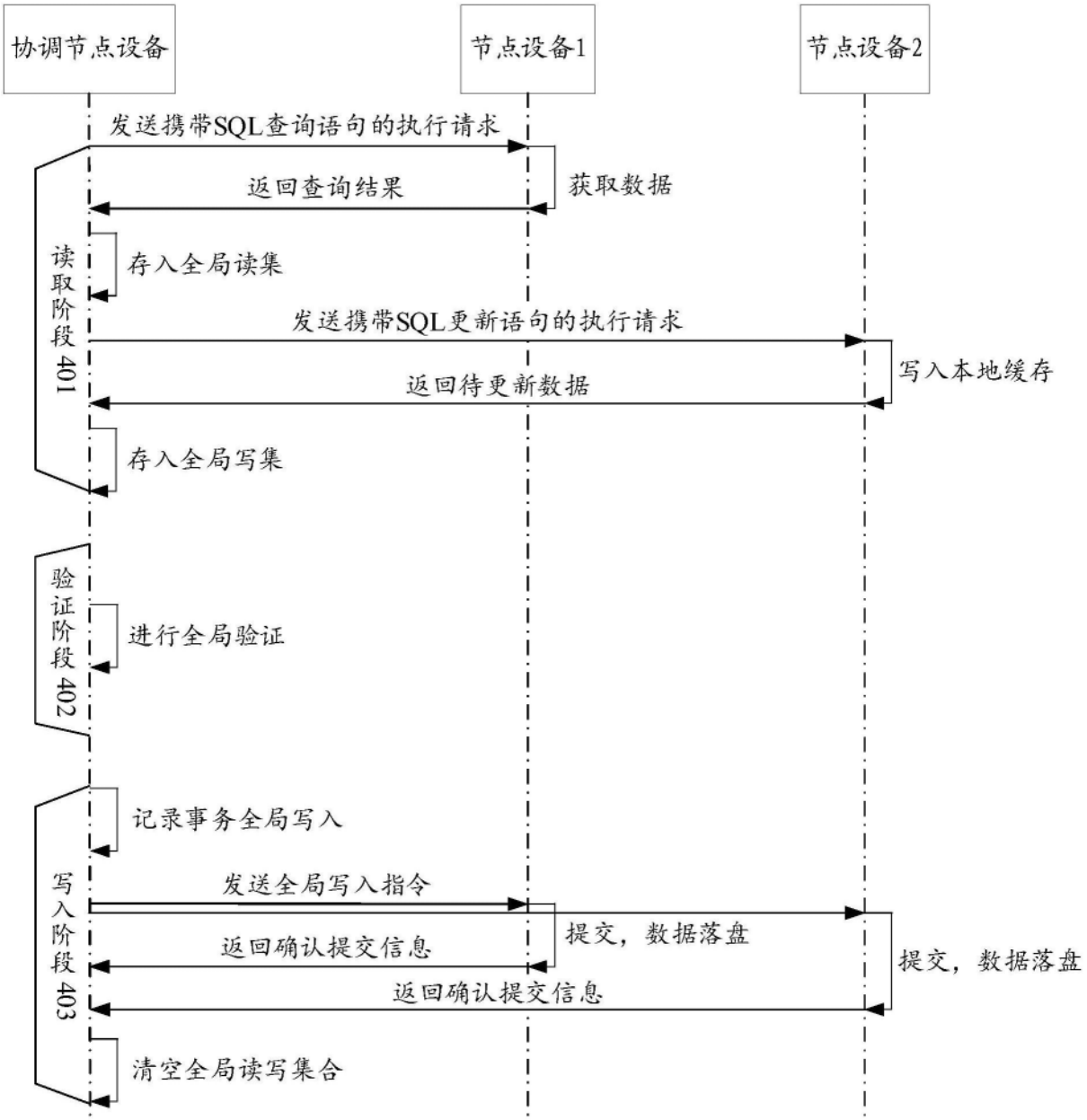

协调节点设备
节点设备1
节点设备2
发送携带SQL查询语句的执行请求
返回查询结果
获取数据
存入全局读集
读取阶段401
发送携带SQL更新语句的执行请求
写入本地缓存
返回待更新数据
存入全局写集
验证阶段402
进行全局验证
记录事务全局写入
写入阶段403
发送全局写入指令
提交，数据落盘
返回确认提交信息
提交，数据落盘
返回确认提交信息
清空全局读写集合

图4

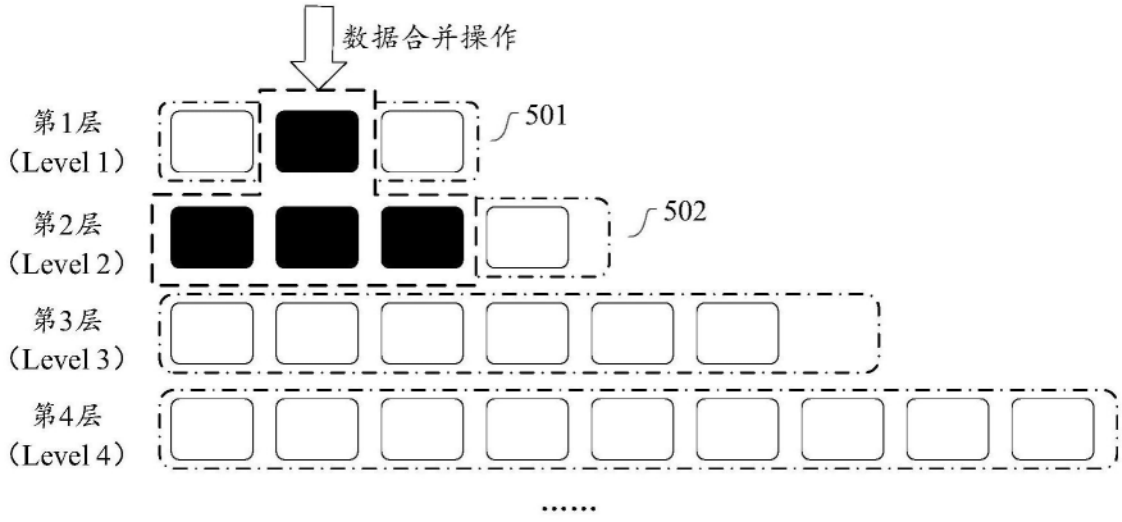

数据合并操作
第1层
(Level 1)
501
第2层
(Level 2)
502
第3层
(Level 3)
第4层
(Level 4)
.....

图5

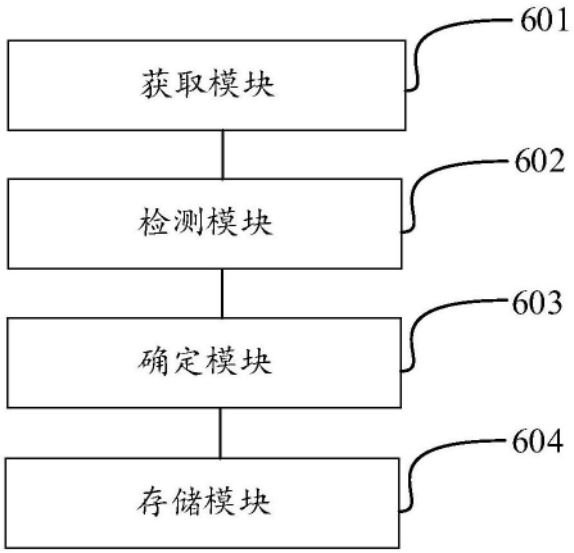

601
获取模块
602
检测模块
603
确定模块
604
存储模块

图6

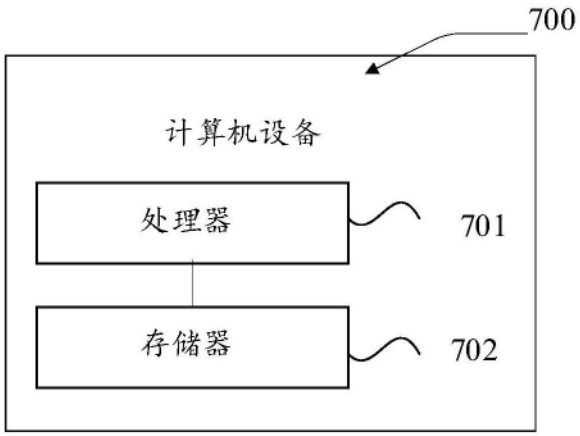


图7