

数据库事务处理方法、装置、服务器及存储介质

本申请要求于 2020 年 06 月 10 日提交中国专利局、申请号为 202010520559.2、名称为“数据库事务处理方法、装置及服务器”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

5 技术领域

本申请涉及数据库技术领域，更具体地，涉及一种数据库事务处理方法、装置、服务器及存储介质。

背景

元数据（metadata）是用于描述数据的数据，例如，用于描述数据库中数据对象的组织和结构的数据可以视为一种元数据。分布式数据库系统包括全局元数据存储和多个工作节点。全局元数据存储中维护有全局元数据信息，其包括系统中的所有元数据。每个工作节点的本地缓存中存储有部分或全量的全局元数据信息。

工作节点在执行针对数据对象的事务或操作语句之前，通常需要基于该数据对象的元数据进行一些处理。而工作节点首先会从本地缓存获取该元数据，在获取失败的情况下再从全局元数据存储中获取。因此，确保各工作节点缓存的元数据的一致性非常重要。

技术内容

本申请实施例提供了一种数据库事务处理方法，应用于分布式数据库系统的工作节点，该方法包括：在开启目标事务时，获取目标事务的事务时间戳及分布式数据库系统当前的全局最新版本信息，其中，目标事务包括针对目标数据对象的至少一个操作语句，全局版本信息是分布式数据库系统存储的各个元数据中最新生成的元数据的版本信息；根据当前的全局版本信息确定目标数据对象的最新版本的元数据，以及根据事务时间戳确定目标事务的待访问用户数据；基于目标数据对象的最新版本的元数据对待访问用户数据执行目标事务中的操作语句。

本申请实施例提供了一种数据库事务处理装置，应用于分布式数据库系统的工作节点，该装置包括：获取模块、确定模块及事务处理模块。其中，获取模块用于当工作节点开启目标事务时，获取目标事务的事务时间戳及所述分布式数据库系统当前的全局最新版本信息，其中，所述目标事务包括针对目标数据对象的至少一个操作语句，所述全局最新版本信息是所述分布式数据库系统存储的各个元数据中最新生成的元数据的版本信息。确定模块用于根据当前的全局最新版本信息确定目标数据对象的最新版本的元数据，以及根据事务时间戳确定目标事务的待访问用户数据。事务处理模块用于基于目标数据对象的最新版本的元数据对待访问用户数据执行目标事务中的操作语句。

本申请实施例提供的数据库事务处理装置还包括变更模块。变更模块用于：接收针对任一数据对象的元数据的变更指令，根据变更指令生成该数据对象的变更后的元数据；

向全局元数据存储提交变更后的元数据；当变更后的元数据成功提交时，将全局最新版本信息更新为变更后的元数据的版本信息。

5 本申请实施例提供的数据库事务处理装置中，变更模块还用于：在向全局元数据存储提交变更后的元数据之前，当生成变更后的元数据时，向全局时间戳管理器发送第一时间戳分配请求；接收全局时间戳管理器基于第一时间戳分配请求返回的时间戳，并将该时间戳确定为变更后的元数据的版本信息。

10 本申请实施例提供的数据库事务处理装置中，全局最新版本信息存储于全局时间戳管理器中，变更模块将全局最新版本信息更新为变更后的元数据的版本信息的方式为：向全局时间戳管理器发送提交成功通知，使全局时间戳管理器根据提交成功通知将存储的全局最新版本信息更新为变更后的元数据的版本信息。

本申请实施例提供的数据库事务处理装置中，工作节点的本地缓存存储有数据对象的本地最新元数据的版本信息，变更模块还用于：当变更后的元数据被成功提交时，将变更后的元数据写入本地缓存；在本地缓存中，将元数据发生变更的数据对象的本地最新元数据的版本信息更新为变更后的元数据的版本信息。

15 本申请实施例提供的数据库事务处理装置中，确定模块根据当前的全局最新版本信息确定目标数据对象的最新版本的元数据的方式为：从本地缓存中获取目标数据对象的本地最新元数据的版本信息；对比目标数据对象的本地最新元数据的版本信息与当前的全局最新版本信息是否相同；若相同，从本地缓存存储的元数据中，获取目标数据对象的本地最新元数据作为目标数据对象的最新版本的元数据。

20 本申请实施例提供的数据库事务处理装置中，确定模块根据当前的全局最新版本信息确定目标数据对象的最新版本的元数据的方式为：若目标数据对象的本地最新元数据的版本信息与当前的全局最新版本信息不相同，从本地缓存和全局元数据存储中存储的目标数据对象的各版本的元数据中，查找是否存在目标元数据，查找到的目标元数据的版本信息比目标数据对象的本地最新元数据的版本信息新、且不比当前的全局最新版本信息新；若存在，将所述查找到的目标元数据中版本信息最新的目标元数据确定为目标数据对象的最新版本的元数据。

25 本申请实施例提供的数据库事务处理装置中，确定模块还用于：在将版本信息最新的目标元数据确定为目标数据对象的最新版本的元数据之后，将版本信息最新的目标元数据更新至本地缓存；将目标数据对象的本地最新元数据的版本信息，更新为版本信息最新的目标元数据对应的版本信息。

30 本申请实施例提供的数据库事务处理装置中，确定模块根据当前的全局最新版本信息确定目标数据对象的最新版本的元数据的方式为：若不存在所述目标元数据，将目标数据对象的本地最新元数据确定为目标数据对象的最新版本的元数据，以及将目标数据对象的本地最新元数据的版本信息更新为当前的全局最新版本信息。

35 本申请实施例提供的数据库事务处理装置中，全局最新版本信息存储于所述分布式数据库系统的全局时间戳管理器中，获取模块获取目标事务的事务时间戳及分布式数据库系统当前的全局最新版本信息的方式为：向全局时间戳管理器发送目标事务对应的第二时间戳分配请求；接收全局时间戳管理器基于第二时间戳分配请求返回的响应信息，该响应信息包括目标事务的事务时间戳及当前的全局最新版本信息。

40 本申请实施例提供的数据库事务处理装置中，事务处理模块基于目标数据对象的最新版本的元数据对待访问用户数据执行目标事务中的操作语句的方式为：根据目标数据

对象的最新版本的模式信息，对目标事务中的操作语句进行解析；根据解析结果对待访问用户数据进行处理。

另一方面，本申请实施例提供了一种服务器，包括：一个或多个处理器；存储器；一个或多个程序，其中所述一个或多个程序被存储在存储器中并被配置为由所述一个或多个处理器执行，所述一个或多个程序配置用于执行上述的方法。

另一方面，本申请实施例提供了一种计算机可读存储介质，其上存储有程序代码，该程序代码可被处理器调用执行上述的方法。

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 示出了本申请实施例提供的一种分布式数据库系统的架构示意图。
- 图 2 示出了本申请实施例提供的一种数据库事务处理方法的流程示意图。
- 图 3 示出了图 2 所示步骤 S201 的子步骤示意图。
- 图 4 示出了本申请实施例提供的数据库事务处理方法的另一流程示意图。
- 图 5 示出了本申请实施例提供的数据库事务处理方法的又一流程示意图。
- 图 6 示出了图 2 所示步骤 S202 的子步骤示意图。
- 图 7 示出了本申请实施例提供的数据库事务处理方法的再一流程示意图
- 图 8 示出了本申请实施例提供的数据库事务处理方法在一个具体示例中的交互流程图。
- 图 9 示出了本申请实施例提供的数据库事务处理方法在另一个具体示例中的交互流程图。
- 图 10 示出了本申请实施例提供的数据库事务处理方法在又一个具体示例中的交互流程图。
- 图 11 示出了本申请实施例提供的数据库事务处理方法在再一个具体示例中的交互流程图。
- 图 12 示出了本申请实施例提供的数据库事务处理装置的框图。
- 图 13 是本申请实施例的用于执行根据本申请实施例的数据库事务处理方法的服务器的框图。
- 图 14 是本申请实施例的用于保存或者携带实现根据本申请实施例的数据库事务处理方法的程序代码的存储单元。

实施方式

为了使本技术领域的人员更好地理解本申请方案，下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述。

请参照图1，图1是本申请实施例提供的一种分布式数据库系统10的架构示意图。分布式数据库系统10包括分布式存储系统200、全局元数据存储300及全局时间戳管理器

(Global Timestamp Manager, GTM) 400以及至少两个工作节点, 例如图1所示的工作节点110、120和130。

其中, 分布式存储系统200可以包括多个物理存储节点(例如, 存储服务器、主机等), 用于存储用户数据。分布式存储系统200还可以包括多个数据存储(Data Store, DS)节点, 每个DS节点与所述多个物理存储节点组成的存储资源池中的一部分存储资源对应, 用于管理该部分存储资源所存储的用户数据。

工作节点也可以称为计算节点(Computation Node, CN), 用于接收并执行数据库用户(user)提交的事务请求及事务中的操作语句的进程。这里的数据库用户可以理解为分布式数据库系统中的一个数据库对象, 其也可以称为用户对象, 可以视为用户登录信息与用户数据之间的连接桥梁。

本申请实施例中, 如果分布式数据库系统10采用的是存储与计算分离的架构, 则工作节点和DS节点通常是不同的进程。如果分布式数据库系统10没有采用存储与计算分离的架构, 则工作节点和DS节点只是逻辑上的模块, 实际上可以属于同一个进程。

上述的工作节点和DS节点可以运行在分布式存储系统200中的物理存储节点上, 也可以运行于分布式存储系统200之外的其他物理服务器上, 本申请实施例对此没有限制。

全局元数据存储中存储有分布式数据库系统10中的所有元数据, 每个元数据可以有至少两个版本。每个工作节点都具有本地缓存, 比如, 工作节点110具有本地缓存111, 工作节点120具有本地缓存121, 工作节点130具有本地缓存131。这里的本地可以理解为工作节点所在的物理设备。对应地, 工作节点的本地缓存可以是工作节点所在的物理服务器上的存储空间, 该存储空间可供工作节点缓存分布式数据库系统10的部分或者全量的全局元数据信息。本申请实施例中, 全局元数据存储例如可以是全局数据字典(Global Data Dictionary, GDD), 工作节点的本地缓存则可以是数据字典缓存(Data Dictionary Cache, DDC)。

在数据库系统中, 常见的一种元数据是模式(schema)信息。schema信息是指数据库系统中全体数据的逻辑结构和特征的描述信息, 其可以理解为数据库用户的逻辑数据视图。在数据库系统中, 每个数据对象可以具有对应的schema对象, 这里的schema对象可以理解为数据对象的schema信息按照一定数据结构组合而成的集合。

数据对象是指数据库系统中存储的数据, 随着数据库系统类型的不同, 数据对象也有所不同。比如, 一些数据库系统以数据表(Data Table)为存储的基本对象, 则这些数据库系统中的数据对象可以理解成数据表。又比如, 一些数据库系统以文档(Document)为存储的基本对象, 则这些数据库系统中的数据对象可以理解成文档。以数据表的schema对象为例, 其中通常包括数据表的定义信息, 例如表名、表类型、表中的属性名、属性类型、索引(index)、数据约束等。

以元数据是schema信息为例, 在分布式数据库系统10中, 工作节点在对数据对象进行任何的数据查询和数据操作之前, 通常需要先访问该数据对象的schema对象, 以确定所述数据查询和数据操作的目标数据是否存在。比如, 工作节点110需要从表t1中删除列C1的数据, 则需要访问表t1的schema对象, 以检查表t1中是否存在列C1。详细地, 工作节点101首先会从本地缓存111查找表t1的schema对象, 在无法从本地缓存111中查找到表t1的schema对象时, 再从全局元数据存储300查找表t1的schema对象, 以根据该schema对象检查表t1中是否存在列C1。当通过检查确定表t1中是否存在列C1时, 再执行从表t1中删除列C1的数据的操作。

由此可见，当一个工作节点对某一数据对象O1的schema对象进行变更之后，如果其他工作节点无法及时获知该变更，则其他工作节点在对数据对象O1执行数据库事务或操作语句时，所访问的schema对象可能仍然是数据对象O1的变更前的schema对象，从而可能导致数据库事务或操作语句的处理出现异常。也即，当某一工作节点对数据对象的元数据进行变更后，其他工作节点难以及时获知该元数据的变更，容易导致各工作节点在处理事务时使用的元数据不一致，从而出现事务处理异常。

本申请实施例中，事务是由针对数据库系统中用户数据的一个或多个操作语句（如，用于进行增、删、查或者改的操作语句）组成的一个逻辑工作单元。事务通常具有ACID特性，ACID分别是Atomicity（原子性）、Consistency（一致性）、Isolation（隔离性）、Durability（持久性）的缩写。其中，原子性是指事务是一个不可分割的工作单位，事务中的操作语句所指示的操作要么都发生，要么都不发生。一致性是指事务前后数据的完整性需要保持一致，也可以理解为事务使得数据库系统从一个正确的状态转换到另一个正确的状态。此处的正确的状态可以理解为数据库系统中的数据满足预订的约束条件。隔离性是指当多个用户并发访问数据库系统时，数据库系统为每一个用户开启的事务相互之间不会发生干扰。持久性则是指事务一旦被提交（commit），则该事务对数据库系统中数据的改变就是永久性的。

上述的在元数据变更无法及时被各个工作节点获知时出现的异常，可能有多种。

一个例子中，在分布式数据库系统10中，假如当前表t1的schema对象存在V1和V2两个版本。此时，工作节点110对表t1的schema对象进行了变更，比如，工作节点110在删除表t1中C1列之后，从表t1的schema对象中删除了基于C1列建立的索引。如此，将会产生表t1的新版本的schema对象，即V3版本的schema对象。当V3版本的schema对象还没有被同步至工作节点120时，工作节点120开启了一个事务T1，事务T1依次包括以下三个操作语句：1. 查询表t1有哪些列；2. 向表t1写入一行数据；3. 再向表t1写入一行数据。如果在事务T1中的操作语句2被执行后，V3版本的schema对象被同步至工作节点120。那么，将出现这样的情况下：工作节点120在处理事务T1中的操作语句1和2的过程中，访问的表t1的schema对象是V2版本的，其中存在列C1的索引；在处理事务T1的操作语句3的过程中，访问的表t1的schema对象是V3版本的，其中不存在列C1的索引。也就是，在同一个事务的处理过程中出现了元数据不一致的情况。

分布数据库系统中的工作节点为了使得保持并发事务的隔离性，常采用快照隔离技术来对事务进行处理。快照隔离的基本思想是在数据项发生增、删、改时创建数据项的新版本，并基于当前时间为该新版本分配一个时间戳作为版本信息（如，版本号）。这里的时间戳通常是利用时间戳函数对当前时间进行处理而生成的数值，可以随着时间的递增而递增。工作节点在执行任一事务的操作语句时，可以利用该事务的事务时间戳取得一个用户数据快照（snapshot），这个用户数据快照的版本信息即为该事务时间戳。只有版本信息小于或等于用户数据快照的版本信息的用户数据对该事务是可见的，即，可以作为该事务的待访问用户数据。

在上述的例子中，工作节点120在执行操作语句1和2的过程中，待访问用户数据中，表t1已经不包括C1列的数据。而在其访问的V2版本的schema对象中，则存在C1列的索引。换句话说，当元数据的变更无法及时地被各个工作节点获知的情况下，同一事务或操作语句对用户数据和元数据的隔离级别是不同的，再换句话说，就是工作节点在事务或操作语句的处理过程中可见的用户数据和元数据可能是不对应的。这可能导致最终的处理

结果出错。

因此，当工作节点对数据对象的元数据进行变更后，其他工作节点能够及时地获知该变更是非常重要的。基于此，一些实施方式中，分布式数据库系统会采用基于两阶段提交（Two-phase Commit, 2PC）等事务提交协议来实现schema对象的变更和同步。

5 详细地，工作节点可以将schema对象的变更作为一种特殊的事务，发起schema变更事务的工作节点作为事务提交的协调者，其他需要同步变更的schema对象的工作节点则作为schema变更事务的参与者。采用这种方案的前提是 可以知晓并连接分布式数据库系统中的其他所有节点（即，参与者），并且协调者和各个参与者都需要维护各自的节点状态并记录日志，以供事务提交协议使用。也就是说，这种方式中，工作节点必须是有状态的，并且在schema变更和同步的过程中不能有工作节点退出和加入系统，否则变更和同步就会失败。

10 然而，目前分布数据库领域为了达到更好的可伸缩性和性价比，在架构设计方面大都采用了存储与计算分离的架构和无状态工作节点。在这类分布式数据库系统中，无状态的工作节点与数据存储层是解耦的，只负责接收用户的数据操作请求，根据数据操作请求从存储层访问数据并执行处理任务，无状态的工作节点本身并不保存任何持久的数据和状态。且无状态的工作节点可以因故障、运维、扩容和缩容等原因随时退出或新增。因此，基于两阶段提交等事务提交协议的元数据变更和同步方式通常不适用于这类分布式数据库系统。

15 另一种实施方式中，可以通过定时方式实现schema对象的变更和同步。详细地，进行schema对象变更的工作节点可以将变更的schema对象写入全局元数据存储中，由其他各个工作节点定时去全局元数据存储中读取schema对象以实现同步。

20 再一种实施方式中，可以利用基于租约和中间状态的方式来实现对schema对象的变更。这种方式中，每个工作节点上的scheme对象看作是schema对象的一种状态（在此称为“schema状态”），对于schema对象的变更和同步则可以视为是对schema状态的变更和同步。比如，如果对表t1的schema对象所做的变更是添加某一列的索引，则没有添加索引的schema对象可以视为absent状态，而添加了索引的schema对象可以视为public状态。

25 根据租约机制，每一个schema状态都有一个租期（lease），每个工作节点必须在租约结束（即，租期到期）之前续约。工作节点续约时，需要检查全局元数据存储中是否存在新的schema状态，若存在，则需要将该新状态加载至本地缓存。如果某个工作节点因故障或异常情况无法完成续约，则不再对数据库用户提供服务。租约机制保证了任何一个可对用户提供服务的工作节点在一个租期内都可以获得最新的schema状态。为了防止不同工作节点在某一时刻持有不同的schema状态造成数据的不一致，这类方案采用了schema的中间状态，schema的中间状态只对特定类型（如，删除、插入、更新等）的操作语句有效。如果采用这种方式，在上述的例子中，只有当工作节点上的schema对象处于public状态时，工作节点上表t1的schema对象中添加的索引才是对外可见的，而当工作节点上的schema对象处于absent状态或其他中间状态时，所述添加的索引要么并不存在，要么并不对外可见。

30 可见，上述的定时方式、基于租约和中间状态的方式中，当一个工作节点变更了schema对象后，其他各个工作节点都是在一段时间后才能够同步变更后的schema对象，即无法及时地获知变更后的元数据，从而可能出现上文描述的，同步恰好发生在事务执行过程中，使得属于同一事务的不同操作语句可见的schema信息不同，或是使得同一事

务可见的用户数据和schema信息不对应。

基于此，本申请实施例提出了一种数据库事务处理方法、装置及服务器，可以确保在目标事务开启之前变更的元数据及时地被需要使用该元数据的工作节点获知。下面对该内容进行阐述。

5 请参照图2，图2是本申请实施例提供的一种数据库处理方法的流程示意图，该方法可以应用于图1所示的任意一个工作节点。详细描述如下。

 S201，在开启目标事务时，获取所述目标事务的事务时间戳及分布式数据库系统当前的全局最新版本信息，其中，所述目标事务包括针对目标数据对象的至少一个操作语句，所述全局最新版本信息是所述分布式数据库系统存储的各个元数据中最新生成的元数据的版本信息。

 其中，开启目标事务可以是指目标事务开始被执行。每一个事务的事务时间戳可以是全局时间戳管理器400在该事务开启时基于当前时间信息生成的数值。例如，可以在事务开启时，通过一时间戳函数对当前时间信息进行处理，从而得到一个函数值，该函数值可以作为该事务的事务时间戳。在一些实施例中，时间戳函数通常是单调递增的，对应地，事务的事务时间戳也随着事务开启时间的递增而递增。

 本实施例中，工作节点在获得任一数据对象的任意一个版本的元数据时，可以请求全局时间戳管理器400为该元数据分配一个版本信息。详细地，元数据的版本信息也可以是全局时间戳管理器400利用一时间戳函数对当前时间信息进行处理而生成的数值。本实施例中，全局时间戳管理器400可以通过同一时间戳函数来生成元数据的版本信息和事务的事务时间戳，在此情况下，元数据的版本信息和事务的事务时间戳属于同一时间戳序列。全局时间戳管理器400也可以通过不同的时间戳函数来生成元数据的版本信息和事务的事务时间戳，在此情况下，元数据的版本信息和事务的事务时间戳属于不同的时间戳序列。本实施例对此没有限制。

 在元数据的版本信息是基于时间信息生成的时间戳的情况下，同一元数据的版本信息之间存在时间先后顺序，不同元数据的版本信息之间也存在时间先后顺序。

 比如，分布式数据库系统10中存在元数据md1和md2，其中元数据md1有两个版本，分别为V11和V12，元数据md2有两个版本V21和V22。则，V11、V12、V21、V22之间是有时间先后顺序的，该时间先后顺序可以通过V11、V12、V21和V22之间的大小顺序来体现，在时间戳函数是单调递增函数的情况下，版本信息越大的元数据生成时间越晚，数据就越新。下文中，在未作特殊说明的情况下，均以时间戳函数单调递增为例进行描述。

 那么，在分布式数据库系统10存储的所有元数据的版本信息中，势必存在一个最大的版本信息，具有这个版本信息的元数据就是最新生成的元数据。比如，当分布式数据库系统10只存储有元数据md1和md2的情况下，如果V22是最大的版本信息，则版本信息是V22的元数据md2就是最新生成的元数据。

 全局时间戳管理器400中可以存储有一全局最新版本信息，所述全局最新版本信息即为分布式数据库系统10的所有元数据中最新生成的元数据的版本信息，例如上述例子中版本信息是V22的元数据md2。因此，分布式数据库系统10的全局最新版本信息是随着系统10中的元数据的变更而变化的。当系统10中有新的元数据产生时，全局最新版本信息就会被更新。对应地，S201中描述的当前的全局最新版本信息是指，工作节点在开启目标事务时，从全局时间戳管理器获取到的全局最新版本信息，其表示的是，在目标

事务开启时刻之前分布式数据库系统10中产生的最新元数据的版本信息。

S202, 根据所述当前的全局最新版本信息确定所述目标数据对象的最新版本的元数据, 以及根据所述事务时间戳确定所述目标事务的待访问用户数据。

本实施例中, 工作节点可以基于S201中获得的全局最新版本信息, 即所述当前的全局最新版本信息, 访问全局元数据存储300, 从而获得一个元数据快照。如此, 相当于将版本信息小于或等于所述当前的全局最新版本信息的元数据作为待访问元数据。然后, 工作节点可以从待访问元数据中查找目标数据对象的最新版本的元数据, 即目标数据对象的各个版本的元数据中版本信息最大的一者。

此外, 工作节点还可以用目标事务的事务时间戳做用户数据快照, 从而可以确定目标事务可见的用户数据, 也就是版本信息小于或等于事务时间戳的用户数据。所确定的目标事务可见的用户数据即为所述待访问用户数据, 也即处理目标事务的过程中可以访问的用户数据。

S203, 基于所述目标数据对象的最新版本的元数据对所述待访问用户数据执行所述目标事务中的操作语句。

实施过程中, 工作节点可以访问目标数据对象的最新版本的元数据(如, schema信息), 根据目标数据对象的最新版本的元数据, 对目标事务中的操作语句进行解析, 并根据解析结果对待访问用户数据进行处理。这里的解析例如可以是检查目标事务中的操作语句的处理对象(如, 数据表、数据表中的行、列、主键等)是否存在等, 当解析结果为存在时, 则可以继续执行该操作语句。

通过图2所示流程, 由于分布式数据库系统中任意元数据的变更都可以触发全局版本信息的更新, 而工作节点在开启目标事务时, 是基于当前的全局最新版本信息来确定目标数据对象的最新版本的元数据的, 因而, 可以确保在目标事务开启之前发生变更的元数据可以及时地被工作节点获知。换句话说, 如果在目标事务开启前, 发生了元数据变更, 而工作节点在执行目标事务中恰好需要使用该元数据, 通过本方案可以确保工作节点在整个目标事务的执行过程中使用的是变更后的元数据。

进一步地, 由于本方案可以确保工作节点在整个目标事务的执行过程中使用的是变更后的元数据, 因而, 避免了执行同一事务的不同操作语句的过程, 访问的元数据不一致的情况。此外, 本方案中, 工作节点在一开始拿到的就是变更后的元数据, 避免了前述例子中, 由于元数据变更无法及时可见所导致的同一事务可见的用户数据和元数据不对应的问题。

请一并参照图2和图3, 下面将对图2所示流程做进一步的阐述。详细地, S201中获取所述目标事务的事务时间戳及所述分布式数据库系统当前的全局最新版本信息的步骤可以通过图3所示流程实现。

S201-1, 向全局时间戳管理器发送目标事务对应的第二时间戳分配请求。

其中, 第二时间戳分配请求是指针对某一事务发出的时间戳分配请求, 用于请求全局时间戳管理器400为该事务分配事务时间戳。可以理解, 这里的第二时间戳分配请求仅仅是为了和后文的第一时间戳分配请求进行区分, 而并非通过“第一”、“第二”对时间戳请求的重要性进行限制。

S201-2, 接收全局时间戳管理器基于第二时间戳分配请求返回的响应信息, 该响应信息包括目标事务的事务时间戳及所述当前的全局最新版本信息。

本实施例中, 全局时间戳管理器400在接收到第二时间戳分配请求时, 可以基

于当前时间信息生成一时间戳作为事务时间戳，并获取当前的全局最新版本信息，填充至第二时间戳分配请求对应的响应信息中，并将该响应信息发送给工作节点。值得说明的是，生成事务时间戳和获取当前的全局最新版本信息这两个操作具有原子性，换句话说，这两个操作要么都发生，要么都不发生。实施过程中，可以通过为这两个操作设置原子锁的方式实现其原子性。当然，也可以通过其他方式实现所述两个操作的原子性，本实施例对此没有限制。

通过图3所示流程，可以确保目标事务的事务时间戳和当前的全局最新版本信息基本与同一时间点对应，对应地，基于事务时间戳确定的待访问用户数据和基于当前的全局最新版本信息确定的目标数据对象的最新版本的数据是相互对应的版本。

本实施例中，全局时间戳管理器400 维护的全局最新版本信息可以通过图4所示的流程来更新。

S401，接收针对任一数据对象的元数据的变更指令，根据所述变更指令生成该数据对象的变更后的元数据。

这里的变更指令可以是能够使元数据发生变化的任意指令，如，用于实现增、删、改等操作的指令。变更指令可以由用户或数据库管理员输入，也可以由针对数据对象的操作而触发，本实施例对此没有限制。

分布式数据库系统10中的每个工作节点在接收到针对任意一个数据对象的元数据的变更指令时，可以对该数据对象的当前版本的元数据做一个快照，该快照的版本信息可以是基于当前时间信息生成的时间戳。然后，对该数据对象的元数据执行变更指令，得到该数据对象新版本的元数据，即所述变更后的元数据。

此时，可以请求全局时间戳管理器400为所述变更后的元数据分配一个版本信息。基于此，本实施例提供的数据库事务处理方法还可以包括图5所示的步骤。

S501，当生成变更后的元数据时，向全局时间戳管理器发送第一时间戳分配请求。

S502，接收所述全局时间戳管理器基于所述第一时间戳分配请求返回的时间戳，并将该时间戳确定为所述变更后的元数据的版本信息。

本实施例中，第一时间戳分配请求是指针对元数据发送的时间戳分配请求，基于第一时间戳分配请求返回的时间戳是元数据的版本信息。

全局时间戳管理器400在接收到工作节点发送的第一时间戳分配请求时，可以基于当前时间信息生成一个时间戳，并将时间戳携带在第一时间戳分配请求对应的响应信息中返回给工作节点。工作节点接收该响应信息，提取该响应信息中的时间戳，并将提取的时间戳确定为所述变更后的元数据的版本信息。

S402，向全局元数据存储提交所述变更后的元数据。

本实施例中，图5所示的步骤可以在S402之前执行。实施过程中，工作节点可以将变更后的元数据及其版本信息一并提交给全局元数据存储300，变更后的元数据及其版本信息将被写入全局元数据存储300，写入成功后全局元数据存储300可以返回一个提交确认信息。此时，变更后的元数据及其版本信息将被持久化地存储于全局元数据存储300中。

S403，当所述变更后的元数据成功提交时，将所述全局最新版本信息更新为所述变更后的元数据的版本信息。

工作节点如果接收到全局元数据存储300返回的提交确认信息，可以认为所述变更

后的元数据已经提交成功。因此，工作节点可以在接收到全局元数据存储300返回的提交确认信息时，向全局时间戳管理器400发送提交成功通知，示例性地，提交成功通知可以包括变更后的元数据的版本信息，从而使全局时间戳管理器400可以根据接收的提交成功通知，将其存储的全局最新版本信息修改为所述变更后的元数据的版本信息。

5 详细地，全局时间戳管理器400在接收到提交成功通知时，可以基于提交成功通知中携带的变更后的元数据的版本信息，确定具有该版本信息的元数据已经成功提交至全局元数据存储300，进而可以将存储的全局最新版本信息更新为提交成功通知中携带的该版本信息，即变更后的元数据的版本信息。

本实施例中，在变更后的元数据提交不成功的情况下，工作节点可以重复执行S402。
10 可以理解，工作节点在进行元数据变更的过程中维护有持久化的日志或任务队列，所述持久化的日志或任务队列记录了元数据变更的进度。如果在元数据变更的过程中，执行元数据变更的工作节点出现故障，该工作节点或者其他工作节点可以在恢复后基于所述持久化的日志或任务队列继续执行元数据变更，直至变更后的元数据提交成功。

本实施例中，全局最新版本信息可以理解为一个变量，而对全局最新版本信息的更新可以理解成对该变量的值的更新。为了与后文描述的其他变量区分，本文中，全局最新版本信息可以称为第一变量。

通过图4所示流程，可以确保全局时间戳管理器400存储的全局最新版本信息是分布式数据库系统10中最新生成的元数据的版本信息。

为了便于描述，在此将目标事务所针对的目标数据对象约定为第一数据对象。如上文描述的，任意一个工作节点对元数据的变更都会触发全局时间戳管理器400中的全局最新版本信息的更新。因此，可能存在这样的情况，在目标事务开启之前，目标数据对象并没有新版本的元数据产生，或是新版本的元数据已经被同步到开启目标事务的工作节点的本地缓存中。在此情况下，工作节点本地缓存中的目标数据对象的最新元数据，已经是目标数据对象在分布式数据库系统10中的最新元数据。此时，工作节点在执行
25 S202的过程中并没有必要访问全局元数据存储300。

因此，为了避免工作节点对全局元数据存储300进行不必要的访问，从而减少网络IO（Input/Output，输入/输出）次数，减少对网络带宽的占用，提升系统性能，本实施例中，图2所示的S202可以通过图6所示的流程实现。

S202-1，从本地缓存中获取目标数据对象的本地最新元数据的版本信息。

30 其中，本地缓存是指开启目标事务的工作节点的本地缓存。每个工作节点的本地缓存中可以维护有每个数据对象的本地最新元数据的版本信息，这里描述的本地最新元数据的版本信息可以理解成一个变量，在此称为第二变量。第二变量的值是可被更新的，在每个工作节点的本地缓存中，存在每个数据对象对应的第二变量。

比如，在工作节点120的本地缓存121中，存在分布式数据库系统10中每个数据对象的第二变量，第二变量的值表示该数据对象在本地缓存121中的最新元数据的版本信息。以该数据对象是数据表t1为例，本地缓存121存储有数据表t1的各版本元数据，数据表t1在本地缓存121中的第二变量的值表示的是所述各版本元数据的版本信息中最新的一个。

以本实施例提供的数据库事务处理方法应用于图1所示的工作节点120为例，如果工作节点120开启了一个事务T2，事务T2包括针对数据表t2的至少一个操作语句。那么，
40 事务T2可以视为目标事务，工作节点202-1可以视为开启并用于处理目标事务T2的节点，数据表t2可以视为目标数据对象（或，第一数据对象），工作节点120的本地缓存121可以

视为S202-1中的本地缓存。

实施过程中，工作节点120可以从本地缓存121中查找数据表t2的第二变量，获取该第二变量的当前值，该当前值即为数据表t2在本地缓存121中的最新元数据的版本信息。

5 S202-2，对比所述目标数据对象的本地最新元数据的版本信息与所述当前的全局最新版本信息是否相同。若是，执行S202-3；若否，可以执行S202-4。

S202-3，从所述本地缓存存储的元数据中，获取所述目标数据对象的本地最新元数据作为所述目标数据对象的最新版本的元数据。

10 本实施例中，为了确定本地缓存中的目标数据对象的本地最新元数据，是否是目标数据对象的全局最新元数据，可以对通过S202-1获得的目标数据对象的本地最新元数据的版本信息与通过S201获得的所述当前的全局最新版本信息进行比较。

如果两者相同，表示本地缓存中的目标数据对象的本地最新元数据就是分布式数据库系统10中的全局最新元数据，可以直接从本地缓存获取该本地最新元数据，而不必访问全局元数据存储300。

15 S202-4，从所述本地缓存和所述全局元数据存储中存储的所述目标数据对象各版本的元数据中，查找是否存在目标元数据，查找到的目标元数据的版本信息比所述目标数据对象的本地最新元数据的版本信息新、且不比所述当前的全局最新版本信息新。若是，可以执行S202-5；若否，可以执行S202-6。

S202-5，将所述查找到的目标元数据中版本信息最新的目标元数据确定为所述目标数据对象的最新版本的元数据。

20 如果本地缓存中的目标数据对象的本地最新元数据的版本信息和所述当前的全局最新版本信息不相同，表示在目标事务开启前，已经发生了一次或多次元数据变更，所述一次或多次元数据变更触发了全局最新版本信息的更新，所述一次或多次元数据变更中可能存在目标数据对象的元数据变更，也可能不存在。

25 如果所述一次或多次元数据变更中存在目标数据对象的元数据变更，则在本地缓存或者全局元数据存储300中势必存在目标元数据，也就是版本信息处于第一数据对象的本地最新元数据的版本信息与所述当前的全局最新版本信息之间的元数据。这是因为，全局最新版本信息的更新，是在触发该变更后的元数据被成功提交至全局元数据存储300后才执行的，因此，具有当前的全局最新版本信息及其之前的版本信息的元数据均已被写入全局元数据存储300中，可以被查找到。

30 在目标数据对象各版本元数据中，版本信息最新的目标元数据是最后一次变更产生的，也就是目标数据对象在分布式数据库系统10中最新版本的元数据。对应地，工作节点可以将版本信息最新的目标元数据更新至本地缓存，以使本地缓存中存储有目标数据对象的最新版本的元数据，并将目标数据对象的本地最新元数据的版本信息更新为该版本最新的目标元数据的版本信息（即，目标数据对象的最新版本35 的元数据的版本信息）。

S202-6，将所述目标数据对象的本地最新元数据确定为所述目标数据对象的最新版本的元数据。

40 如果上述的一次或多次元数据变更中不存在目标元数据的变更，那么，目标数据对象的本地最新元数据就已经是目标数据对象在分布式数据库系统10中最新版本的元数据。换句话说，本地缓存或者全局元数据存储300存储的目标数据对象的元数据中不会存在目标元数据。因此，当工作节点无法通过S202-4查找到目标元数

据时，可以直接将目标数据对象的本地最新元数据作为目标数据对象的最新版本的元数据。

可以理解，S202、S202-3、S202-5以及S202-6中描述的目标数据对象的最新版本的元数据，均是指目标数据对象在整个分布式数据库系统10中的最新版本的元数据。

本实施例中，在执行S202-6之后，工作节点还可以将本地缓存中的目标数据对象的本地最新元数据的版本信息（即，第二变量的值）更新为当前的全局最新版本信息（即，第一变量的当前值）。

此外，本实施例中，变更后的元数据除了需要写入全局元数据存储300，还需要同步到各个工作节点。其中，产生变更后的元数据的工作节点可以通过图7所示的流程，对该变更后的元数据进行同步。对应地，产生变更后的元数据的工作节点可以对元数据发生变更的数据对象的第二变量进行更新。详细描述如下。

S701，当变更后的元数据被成功提交时，将所述变更后的元数据写入本地缓存。

实施过程中，工作节点在向全局元数据存储300提交变更后的元数据，并接收到全局元数据存储300返回的提交确认信息时，可以将变更后的元数据写入本地缓存中。

S702，在所述本地缓存中，将元数据发生变更的数据对象的本地最新元数据的版本信息更新为所述变更后的元数据的版本信息。

其中，元数据发生变更的数据对象是指所述变更后的元数据所属的数据对象。当工作节点成功将变更后的元数据写入本地缓存后，可以将该变更后的元数据所属的数据对象的本地最新元数据的版本信息（即，第二变量的值）修改为该变更后的元数据的版本信息。

为了使本领域技术人员更加清楚地理解本方案，下面结合图1所示场景给出一些具体示例，以对本实施例提供的数据库事务处理方法进行详细阐述。

假设初始状态下，分布式数据库系统10中存在数据表t1、t2和t3，其中，数据表t1的元数据为V10版本的模式（schema）对象O1，数据表t2的元数据为V20版本的schema对象O2，数据表t3的元数据为V30版本的schema对象O3。

全局元数据存储300存储有V10版本的schema对象O1、V20版本的schema对象O2以及V30版本的schema对象O3，并且全局元数据存储300维护有第一变量GLSV（global largest schema version，全局最大模式版本号），GLSV的当前值为V20，表示分布式数据库系统10中当前最大的版本信息是V20，也即，最新的版本信息是V20。

本地缓存111、121以及132均缓存有V10版本的schema对象O1、V20版本的schema对象O2及V30版本的schema对象O3，并且均维护有数据表t1对应的第二变量EGSV-1（equivalent global schema version-1，等效全局模式版本号-1）、数据表t2对应的第二变量EGSV-2以及数据表t3对应的第二变量EGSV-3。其中，EGSV-1的当前值是V10，EGSV-2的当前值为V20，EGSV-3的当前值为V30。

一个示例中，本实施例提供的数据库事务处理方法的流程可以包括图8所示的流程。

S1，工作节点110接收到针对数据表t1的schema对象O1的变更指令，根据变更指令生成变更后的schema对象O1。

S2，工作节点110向全局时间戳管理器400发送第一时间戳分配请求r1。

S3，全局时间戳管理器400根据当前时间信息生成一时间戳V11。

S4，全局时间戳管理器400向工作节点110发送时间戳V11。

S5, 工作节点110将时间戳V11作为变更后的schema对象O1的版本信息, 得到V11版本的schema对象O1。

S6, 工作节点110向全局元数据存储300提交V11版本的schema对象O1。

5 S7, 全局元数据存储300在存储V11版本的schema对象O1之后, 返回一提交确认信息给工作节点110。

S8, 工作节点110在接收到提交确认信息时, 向全局时间戳管理器400发送通知, 通知携带有版本信息V11。

S9, 全局时间戳管理器400根据通知将GLSV的当前值更新为V11。

10 S10, 工作节点110在接收到提交确认信息时, 将V11版本的schema对象O1写入本地缓存111, 并将本地缓存111中的EGSV-1的当前值更新为V11。

其中, S8和S10可以是并行执行的。

此时, 如果工作节点110开启事务T1, 事务T1包括针对数据表t1的操作语句, 则可以按照如下流程处理。

15 S11, 工作节点110在开启事务T1时, 向全局时间戳管理器400发送第二时间戳分配请求r2。可以理解, 这里的事务T1即为目标事务。

S12, 全局时间戳管理器400接收r2, 基于当前时间信息生成一时间戳V40, 并获取GLSV的当前值V11, 将V40和V11填充到r2对应的响应信息中。

S13, 全局时间戳管理器400将r2对应的响应信息返回给工作节点110。

20 S14, 工作节点110从本地缓存111中获取EGSV-1的当前值V11, 确定EGSV-1的当前值V11和GLSV的当前值V11相同, 从本地缓存111中获取版本信息为V11的schema对象O1。

S15, 工作节点110将V40作为事务T1的事务时间戳, 基于V40获得用户数据快照, 将版本信息小于或等于V40的用户数据确定为事务T1的待访问用户数据data1。

S16, 根据V11版本的schema对象O1对事务T1中的操作语句进行解析, 并依据解析结果对data1执行事务T1中的操作语句。

25 另一个示例中, 如果工作节点110是在S9被执行之后、S10还没有被执行时开启的, 那么, 本实施例提供的数据库事务处理方法首先可以按照S11-S13进行处理, 在S13之后, 则可以按照图9所示流程处理。

30 S17, 工作节点110从本地缓存111中获取EGSV-1的当前值V10, 确定EGSV-1的当前值V10与GLSV的当前值V11不相同, 则从本地缓存111和全局元数据存储300存储的schema对象O1中, 查找版本信息大于V10、且小于或等于V11的schema对象O1 (即, 目标元数据)。

S18, 工作节点110从全局元数据存储300中查找到V11版本的schema对象O1, 将其确定为最新版本的schema对象O2, 并将V11版本的schema对象O1同步至本地缓存111中, 将本地缓存111中的EGSV-1的当前值更新为V11。

35 S19, 工作节点110基于事务时间戳V40获得用户数据快照, 将版本信息小于或等于V40的用户数据确定为事务T1的待访问用户数据data1。

S20, 工作节点110基于V11版本的schema对象O1对事务T1中的操作语句进行解析, 并依据解析结果对data1执行事务T1中的操作语句。

40 在上述示例的基础上, 工作节点110在时刻2接收到针对数据表t2的schema对象O2的变更指令, 则可以按照上述的S1-S10的类似流程进行处理, 完成所述处理后, 全局元数据存储300中新增了新版本 (V21) 的schema对象O2, GLSV的当前值是V21。工作节点

110的本地缓存111中新增了V21版本的schema对象O2，且本地缓存111中的EGSV-2的当前值为V21。而工作节点120和130上不存在V21版本的schema对象O2，且本地缓存121和131中的EGSV-2的当前值仍为V20。

5 此时，如果工作节点120开启事务T2，事务T2包括针对数据表t2的操作语句。在此情况下，本实施例提供的数据库事务处理方法可以按照图10所示流程处理：

S21，工作节点120开启事务T2时，向全局时间戳管理器400发送第二时间戳分配请求r3。这里的事务T2可以视为目标事务。

S22，全局时间戳管理器400接收r3，基于当前时间信息生成一时间戳V41，并获取GLSV的当前值V21，将V41和V21填充到r3对应的响应信息中。

10 S23，全局时间戳管理器400将响应信息返回给工作节点120。

S24，工作节点120从本地缓存121中获取EGSV-2的当前值V20，确定EGSV-2的当前值V20和GLSV的当前值V21不相同，则从本地缓存121和全局元数据存储300中存储的schema对象O2中，查找版本信息大于V20、且小于或等于V21的schema对象O2（即，目标元数据）。

15 S25，工作节点120从全局元数据存储300中查找到V21版本的schema对象O2，将其确定为最新版本的schema对象O2，并将V21版本的schema对象O2同步至本地缓存121中，将本地缓存121中的EGSV-2的当前值更新为V21。

20 S26，工作节点120基于事务时间戳V41确定事务T2的待访问用户数据data2，基于V21版本的schema对象O2解析事务T2中的操作语句，并基于解析结果对data2执行事务T2中的操作语句。

在上述示例的基础上，工作节点130在时刻3开启事务T3，事务T3包括针对数据表t3的操作语句。则本实施例提供的数据库事务处理方法可以按照图11所示流程处理：

S27，工作节点130开启事务T3时，向全局时间戳管理器400发送第二时间戳分配请求r4。这里的事务T3可以视为目标事务。

25 S28，全局时间戳管理器400接收r4，基于当前时间信息生成一时间戳V42，并获取GLSV的当前值V21，将V42和V21填充至r4对应的响应信息中。

S29，全局时间戳管理器400将该响应信息返回给工作节点130。

30 S30，工作节点130从本地缓存131中获取EGSV-3的当前值V30，确定EGSV-3的当前值V30与GLSV的当前值V21不相同，则从本地缓存131和全局元数据存储300存储的schema对象O3中，查找版本信息大于V30、且小于或等于V21的schema对象O3（即，目标元数据）。

S31，工作节点130没有查找到目标元数据，则将V30版本的schema对象O3确定为最新版本的schema对象O3，并将本地缓存131中的EGSV-3的当前值更新为V21。

35 S32，工作节点130基于事务时间戳V42确定事务T2的待访问用户数据data3，基于V21版本的schema对象O3（即，V30版本的scheme对象O3）解析事务T2中的操作语句，并基于解析结果对data3执行事务T3中的操作语句。

40 本申请提供的技术方案，分布式数据库系统维护有全局最新版本信息，该全局最新版本信息是指系统的所有元数据中版本信息最新的元数据的版本信息。工作节点在开启针对目标数据对象的目标事务时，可以获取事务时间戳以及当前的全局最新版本信息，从而根据当前的全局最新版本信息确定目标数据对象的最新版本元数据，根据事务时间戳将目标事务可见的用户数据确定为待访问用户数据，并基于目标数据对象的最新版本

元数据对目标事务的待访问用户数据执行目标事务中的操作语句。由于对分布式数据库系统中任意元数据的变更均可以触发全局版本信息的更新，而工作节点在开启目标事务时，是基于当前的全局版本信息来确定目标数据对象的最新版本元数据的，因而可以确保在目标事务开启之前变更的元数据能够及时地对需要使用该元数据的工作节点可见。

5 此外，本申请的技术方案易于在已支持多版本并发控制的主流数据库管理系统中实现和应用，并且无显著的额外性能开销，不影响分布式数据库系统可用性和扩展性。

请参阅图 12，其示出了本申请实施例提供的一种数据库事务处理装置 1200 的结构框图。该装置 1200 可以包括：获取模块 1210、确定模块 1220 以及事务处理模块 1230。

10 其中，获取模块 1210 用于当所述工作节点开启目标事务时，获取所述目标事务的事务时间戳及所述分布式数据库系统当前的全局最新版本信息，其中，所述目标事务包括针对目标数据对象的至少一个操作语句，所述全局最新版本信息是所述分布式数据库系统存储的各个元数据中最新生成的元数据的版本信息。

确定模块 1220 用于根据所述当前的全局最新版本信息确定所述目标数据对象的最新版本的元数据，以及根据所述事务时间戳确定所述目标事务的待访问用户数据。

15

事务处理模块 1230 用于基于所述目标数据对象的最新版本的元数据对所述待访问用户数据执行所述目标事务中的操作语句。

所述装置 1200 还可以包括变更模块。变更模块用于：接收针对任一数据对象的元数据的变更指令，根据所述变更指令生成该数据对象的变更后的元数据；向全局元数据存储提交所述变更后的元数据；当所述变更后的元数据成功提交时，将所述全局最新版本信息更新为所述变更后的元数据的版本信息。

20

所述变更模块还可以用于：当生成所述变更后的元数据时，向全局时间戳管理器发送第一时间戳分配请求；接收所述全局时间戳管理器基于所述第一时间戳分配请求返回的时间戳，并将该时间戳确定为所述变更后的元数据的版本信息。

25

工作节点的本地缓存存储有数据对象的本地最新元数据的版本信息。所述变更模块还可以用于：当所述变更后的元数据被成功提交时，将所述变更后的元数据写入本地缓存；在所述本地缓存中，将元数据发生变更的数据对象的本地最新元数据的版本信息更新为所述变更后的元数据的版本信息。

30 所述确定模块 1220 根据所述当前的全局最新版本信息确定所述目标数据对象的最新版本的元数据的方式可以为：从本地缓存中获取所述目标数据对象的本地最新元数据的版本信息；对比所述目标数据对象的本地最新元数据的版本信息与所述当前的全局最新版本信息是否相同；若相同，从所述本地缓存存储的元数据中，获取所述目标数据对象的本地最新元数据作为所述目标数据对象的最新版本的元数据。

35

所述确定模块 1220 根据所述当前的全局最新版本信息确定所述目标数据对象的最新版本的元数据的方式还可以为：若所述目标数据对象的本地最新元数据的版本信息与所述当前的全局最新版本信息不相同，从所述本地缓存和所述全局元数据存储中存储的所述目标数据对象的各版本的元数据中，查找是否存在目标元数据，查找到的目标元数据的版本信息比所述获取的本地最新版本信息新、且不比所述当前的全局最新版本信息新；若存在，将所述查找到的目标元数据中版本信息最新的目

40

标元数据确定为所述目标数据对象的最新版本的元数据。

所述确定模块 1220 还可以用于：在将所述查找到的目标元数据中版本信息最新的目标元数据确定为所述目标数据对象的最新版本的元数据之后，将所述版本信息最新目标元数据更新至所述本地缓存；将所述目标数据对象的本地最新元数据的版本信息，更新为所述版本信息最新的目标元数据对应的版本信息。

所述确定模块 1220 根据所述当前的全局最新版本信息确定所述目标数据对象的最新版本的元数据的方式还可以为：若不存在所述目标元数据，将所述目标数据对象的本地最新元数据确定为所述目标数据对象的最新版本的元数据，以及将所述目标数据对象的本地最新元数据的版本信息更新为所述当前的全局最新版本信息。

全局最新版本信息存储于所述分布式数据库系统的全局时间戳管理器中。获取模块 1210 获取所述目标事务的事务时间戳及所述分布式数据库系统当前的全局最新版本信息的方式可以是：向所述全局时间戳管理器发送所述目标事务对应的第二时间戳分配请求；接收所述全局时间戳管理器基于所述第二时间戳分配请求返回的响应信息，该响应信息包括所述目标事务的事务时间戳及所述当前的全局最新版本信息。

事务处理模块 1230 基于所述目标数据对象的最新版本的元数据对所述待访问用户数据执行所述目标事务中的操作语句的方式可以是：根据所述目标数据对象的最新版本的模式信息，对所述目标事务中的操作语句进行解析；根据解析结果对所述待访问用户数据进行处理。

所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，上述描述装置和模块的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

在本申请所提供的几个实施例中，所显示或讨论的模块相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或模块的间接耦合或通信连接，可以是电性，机械或其它的形式。

另外，在本申请各个实施例中的各功能模块可以集成在一个处理模块中，也可以是各个模块单独物理存在，也可以两个或两个以上模块集成在一个模块中。上述集成的模块既可以采用硬件的形式实现，也可以采用软件功能模块的形式实现。

请参考图 13，其示出了本申请实施例提供的一种服务器 1300 的结构框图。该服务器 1300 可以是上述的工作节点所在的服务器，其可以是独立的服务器或多个物理服务器组成的服务器集群，还可以是用于执行云计算、大数据和人工智能平台等基础云计算服务的云服务器。本申请中的服务器 1300 可以包括一个或多个如下部件：处理器 1310、存储器 1320、以及一个或多个应用程序，其中一个或多个应用程序可以被存储在存储器 1320 中并被配置为由一个或多个处理器 1310 执行，一个或多个程序配置用于执行如前述方法实施例所描述的方法。

处理器 1310 可以包括一个或者多个处理核。处理器 1310 利用各种接口和线路连接整个服务器 1300 内的各个部分，通过运行或执行存储在存储器 1320 内的指令、程序、代码集或指令集，以及调用存储在存储器 1320 内的数据，执行服务器 1300 的各种功能和处理数据。可选地，处理器 1310 可以采用数字信号处理 (Digital Signal Processing, DSP)、现场可编程门阵列 (Field-Programmable Gate Array, FPGA)、可编程逻辑阵列 (Programmable Logic Array, PLA) 中的至少一种硬件形式来实现。处理器 1310 可集成中央处理器 (Central Processing Unit, CPU)、图像处理器 (Graphics

Processing Unit, GPU) 和调制解调器等中的一种或几种的组合。其中, CPU 主要处理操作系统、用户界面和应用程序等; GPU 用于负责显示内容的渲染和绘制; 调制解调器用于处理无线通信。可以理解的是, 上述调制解调器也可以不集成到处理器 1310 中, 单独通过一块通信芯片进行实现。

- 5 存储器 1320 可以包括随机存储器 (Random Access Memory, RAM), 也可以包括只读存储器 (Read-Only Memory)。存储器 1320 可用于存储指令、程序、代码、代码集或指令集。存储器 1320 可包括存储程序区和存储数据区, 其中, 存储程序区可存储用于实现操作系统的指令、用于实现至少一个功能的指令 (比如触控功能、声音播放功能、图像播放功能等)、用于实现下述各个方法实施例的指令等。
- 10 存储数据区还可以存储服务器 1300 在使用中所创建的数据 (比如元数据) 等。

请参考图 14, 其示出了本申请实施例提供的一种计算机可读存储介质的结构框图。该计算机可读介质 1400 中存储有程序代码, 所述程序代码可被处理器调用执行上述方法实施例中所描述的方法。

- 15 计算机可读存储介质 1400 可以是诸如闪存、EEPROM (电可擦除可编程只读存储器)、EPROM、硬盘或者 ROM 之类的电子存储器。在一些实施例中, 计算机可读存储介质 1400 包括非瞬时性计算机可读介质 (non-transitory computer-readable storage medium)。计算机可读存储介质 1400 具有执行上述方法中的任何方法步骤的程序代码 1410 的存储空间。这些程序代码可以从一个或者多个计算机程序产品中读出或者写入到这一个或者多个计算机程序产品中。程序代码 1410 可以例如以适当形式进行压缩。
- 20

最后应说明的是: 以上实施例仅用以说明本申请的技术方案, 而非对其限制; 尽管参照前述实施例对本申请进行了详细的说明, 本领域的普通技术人员当理解: 其依然可以对前述各实施例所记载的技术方案进行修改, 或者对其中部分技术特征进行等同替换; 而这些修改或者替换, 并不驱使相应技术方案的本质脱离本申请各实施例技术方案的精神和范围。

25