(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2021 年 3 月 4 日 (04.03.2021)

WIPO | PCT

(10) 国际公布号
WO 2021/036768 A1

(51) 国际专利分类号:
G06F 16/2458 (2019.01)

(21) 国际申请号: PCT/CN2020/108100

(22) 国际申请日: 2020 年 8 月 10 日 (10.08.2020)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201910797634.7 2019年8月27日 (27.08.2019) CN

(71) 申请人: 腾讯科技（深圳）有限公司 (TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED) [CN/CN]; 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。

(72) 发明人: 李海翔(LI, Haixiang); 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。 卢卫(LU, Wei); 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。 杜小勇(DU, Xiaoyong); 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。 赵展浩(ZHAO, Zhanhao); 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。 潘安群(PAN, Anqun); 中国广东省深圳市南山区高新区科技中一路腾讯大厦35层, Guangdong 518057 (CN)。

(74) 代理人: 北京三高永信知识产权代理有限责任公司(BEIJING SAN GAO YONG XIN INTELLECTUAL PROPERTY AGENCY CO., LTD.); 中国北京市

WO 2021/036768 A1

(54) **Title:** DATA READING METHOD, APPARATUS, COMPUTER DEVICE, AND STORAGE MEDIUM

(54) 发明名称: 数据读取方法、装置、计算机设备及存储介质

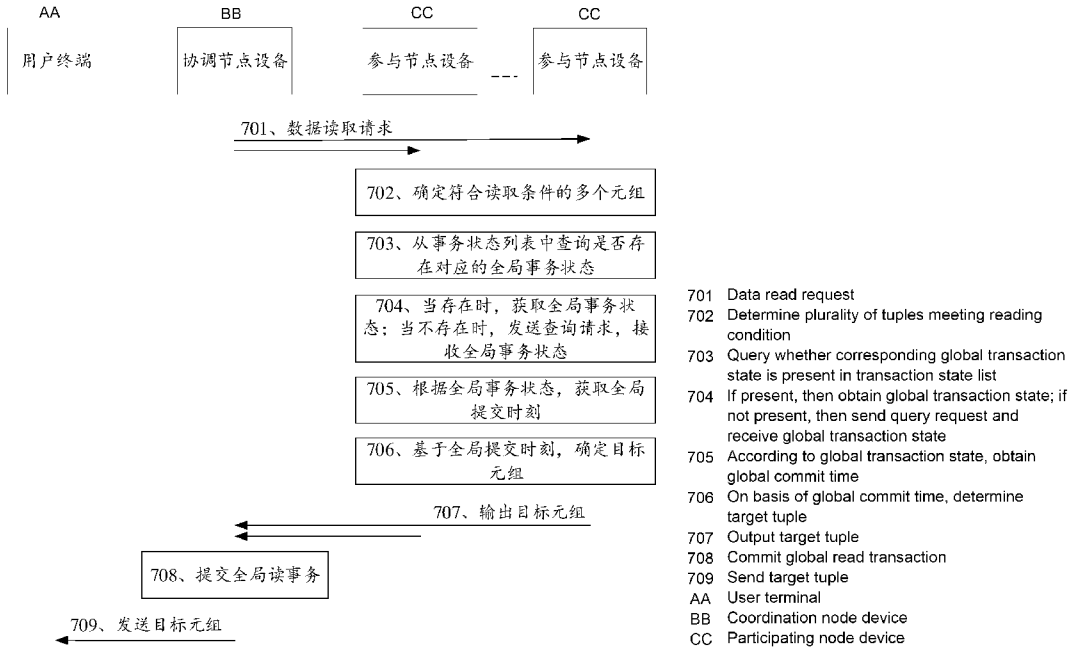

图 7

(57) **Abstract:** Provided are a data reading method, apparatus, computer device, and storage medium, belonging to the technical field of databases; the described method uses a read condition carried in a data read request to determine a plurality of tuples which meet the reading condition (702); the global transaction status of each global transaction is maintained in a database system; thus a global transaction status meeting a consistency condition is obtained, and, according to the global transaction status, a global commit time is obtained; on the basis of the global commit time of the plurality of global transactions, a target tuple is determined from the plurality

[见续页]

海淀区学院路蓟门里和景园A座1单元102室, Beijing 100088 (CN)。

(81) 指定国(除另有指明，要求每一种可提供的国家保护)：AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) 指定国(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

根据细则4.17的声明：

— 关于申请人有权要求在先申请的优先权(细则4.17(iii))

本国际公布：

— 包括国际检索报告(条约第21条(3))。

of tuples. In the described method, since different global commit times are assigned to different global transaction states, the visibility of tuples is determined directly on the basis of global commission, also preventing the occurrence of DRCC exceptions and ensuring the transaction consistency of global read operations in the database system.

(57) 摘要：一种数据读取方法、装置、计算机设备及存储介质，属于数据库技术领域，上述方法通过数据读取请求携带的读取条件，确定符合该读取条件的多个元组（702），由于数据库系统中维护了各个全局事务的全局事务状态，从而获取到符合一致性条件的全局事务状态，从而根据全局事务状态，获取全局提交时刻，基于该多个全局事务的全局提交时刻，从该多个元组中确定目标元组。上述方法由于对不同的全局事务状态赋予不同的全局提交时刻，直接基于全局提交来判断元组的可见性，也就避免了发生DRCC异常，保证了数据库系统中全局读操作的事务一致性。

数据读取方法、装置、计算机设备及存储介质

本公开要求于 2019 年 08 月 27 日提交的申请号为 201910797634.7、发明名称为“数据读取方法、装置、计算机设备及存储介质”的中国专利申请的优先权，其全部内容通过引用结合在本公开中。

技术领域

本公开涉及数据库技术领域，特别涉及一种数据读取方法、装置、计算机设备及存储介质。

背景技术

目前的分布式数据库系统中，有很多可以支持跨节点的写操作，也即是，对于某一个写操作来说，可能涉及到对分布式数据库系统中多个节点设备的写入过程，由此，可能会产生读取数据的事务一致性问题。

例如：当实现跨节点的写操作时，假设有两个节点设备，已经完成了事务提交的准备阶段，本事务可以提交，第一个节点设备提交完成，第二个节点设备正在提交过程中却尚未提交，此时，分布式数据库系统新来了一个全局读操作，第一个节点设备已提交的数据被读取，但是第二个节点设备因尚未完成数据提交，而导致其正在提交的数据无法被读取，这种不一致现象，称为分布式读半已提交异常（Distributed Read Committed-Committing anomaly，简称 DRCC 异常），因此，目前的数据读取过程中，不能保证读取的数据处于事务一致的状态。

发明内容

本公开实施例提供了一种数据读取方法、装置、计算机设备及存储介质。该技术方案如下：

一方面，提供了一种数据读取方法，该方法包括：

当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，确定符合所述读取条件的多个元组；

获取所述多个元组所对应的多个全局事务的全局事务状态；

根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

一方面，提供了一种数据读取方法，该方法包括：

当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

一方面，提供了一种数据读取装置，该装置包括：

第一确定模块，用于当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，

确定符合所述读取条件的多个元组；

第一获取模块，用于获取所述多个元组所对应的多个全局事务的全局事务状态；

第二获取模块，用于根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

第二确定模块，用于基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

一方面，提供了一种数据读取装置，该装置包括：

发送模块，用于当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

提交模块，用于当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

所述发送模块，还用于当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

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下操作：

当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，确定符合所述读取条件的多个元组；

获取所述多个元组所对应的多个全局事务的全局事务状态；

根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

在一种可能实施方式中，当所述数据读取请求中携带指定时刻时，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如下操作：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为已提交状态或正在提交状态，当所述任一全局事务的事务完成时刻大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值；当所述任一全局事务的事务完成时刻不大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述事务完成时刻。

在一种可能实施方式中，当所述数据读取请求中携带指定时刻时，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如下操作：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值。

在一种可能实施方式中，当所述数据读取请求中携带指定时间段时，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如下操作：

基于所述指定时间段的起始时刻，执行获取所述多个全局事务在所述起始时刻下的全局提交时刻的操作；

基于所述指定时间段的终止时刻，执行获取所述多个全局事务在所述终止时刻下的全局

提交时刻的操作。

在一种可能实施方式中，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如下操作：

当所述数据读取请求中携带指定时刻时，对所述多个元组中任一元组，如果产生所述任一元组的全局事务的全局提交时刻小于所述指定时刻，且修改所述任一元组的全局事务的全局提交时刻大于所述指定时刻，将所述任一元组确定为一个目标元组。

在一种可能实施方式中，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如下操作：

当所述数据读取请求中携带指定时间段时，对所述多个元组中任一元组，如果产生所述任一元组的全局事务的全局提交时刻小于所述指定时间段的终止时刻，且修改所述任一元组的全局事务的全局提交时刻大于所述指定时间段的起始时刻，将所述任一元组确定为一个目标元组。

在一种可能实施方式中，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如下操作：

对所述多个元组中的任一元组上记录的全局事务标识，从本地的事务状态列表中，查询是否存在与所述全局事务标识所对应的全局事务状态；

当存在时，获取与所述全局事务标识所对应的全局事务状态；

当不存在时，向所述全局事务标识对应的多个节点设备发送查询请求，接收所述多个节点设备中任一节点设备返回的全局事务状态。

在一种可能实施方式中，每个全局事务对应于一个事务状态元组，所述事务状态元组包括所述每个全局事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项。

在一种可能实施方式中，所述至少一条程序代码还由所述一个或多个处理器加载并执行如下操作：

当任一个全局写事务提交完成时，向所述任一个全局写事务对应的目标节点设备发送数据同步请求，接收所述目标节点设备返回的全局事务状态，所述目标节点设备用于存储所述任一个全局写事务的全局事务状态；或，

如果当前执行的事务涉及到所述任一个全局写事务对应的所述目标节点设备，与所述目标节点设备之间进行增量同步；或，

每间隔目标时长，与所述目标节点设备之间进行增量同步。

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下操作：

当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

在一种可能实施方式中，所述至少一条程序代码还由所述一个或多个处理器加载并执行如下操作：

对任一全局写事务，向所述任一全局写事务涉及的多个节点设备发送数据准备请求；

当接收到所述多个节点设备中每个节点设备所返回的准备成功响应时，向所述多个节点设备发送提交指令，通知目标节点设备记录所述任一全局写事务的事务完成时刻，并通知所述目标节点设备将所述任一全局写事务的全局事务状态置为已提交状态；

当接收到所述多个节点设备中任一节点设备所返回的准备失败响应时，向所述多个节点设备发送回滚指令，通知所述目标节点设备将所述任一全局写事务的全局事务状态置为正在回滚状态，当接收到所述多个节点设备中每个节点设备所返回的回滚完成响应时，通知所述目标节点设备记录所述任一全局写事务的事务完成时刻，并通知所述目标节点设备将所述任一全局写事务的全局事务状态置为已回滚状态。

在一种可能实施方式中，所述至少一条程序代码还由所述一个或多个处理器加载并执行如下操作：

从所述任一全局写事务涉及的多个节点设备中随机选取所述目标节点设备；或，

将所述任一全局写事务涉及的多个节点设备中第一个被访问的节点设备确定为所述目标节点设备。

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数据读取方法所执行的操作。

附图说明

为了更清楚地说明本公开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公开的一些实施例，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 是本公开实施例提供的一种数据读取方法的实施环境示意图；

图 2 是本公开实施例提供的一种 HTAC 系统的架构示意图；

图 3 是本公开实施例提供的一种 DRCC 异常的原理性示意图；

图 4 是本公开实施例提供的一种数据写入方法的交互流程图；

图 5 是本公开实施例提供的一种传统数据库系统元组结构的示意图；

图 6 是本公开实施例提供的一种分布式数据库系统元组结构的示意图；

图 7 是本公开实施例提供的一种数据读取方法的交互流程图；

图 8 是本公开实施例提供的一种事务一致性点的示意图；

图 9 是本公开实施例提供的一种数据读取装置的结构示意图；

图 10 是本公开实施例提供的一种数据读取装置的结构示意图；

图 11 是本公开实施例提供的计算机设备的结构示意图。

具体实施方式

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

本公开实施例涉及的数据库存储有多个数据表，每个数据表可以用于存储元组，一个元组也即是数据表中存储的一个或多个数据项。其中，该数据库可以为基于 MVCC

（Multi-Version Concurrency Control，多版本并发控制）的任一类型的数据库。在本公开实施例中，对该数据库的类型不作具体限定。需要说明的是，上述数据库中的数据基于状态属性，可以包括三种状态：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。

当前态（Current State）：元组的最新版本的数据，是处于当前阶段的数据。处于当前阶段的数据的状态，称为当前态。

过渡态（Transitional State）：不是元组的最新的版本也不是历史态版本，处于从当前态向历史态转变的过程中，处于过渡态的数据，称为半衰数据。

历史态（Historical state）：元组在历史上的一个状态，其值是旧值，不是当前值。处于历史阶段的数据的状态，称为历史态。一个元组的历史态，可以有多个，反映了数据的状态变迁的过程。处于历史态的数据，只能被读取而不能被修改或删除。

需要说明的是，在 MVCC 机制下，数据的上述三种状态均存在，在非 MVCC 机制下，数据可以只存在历史态和当前态。在 MVCC 或封锁并发访问控制机制下，事务提交后的数据的新值处于当前态。以 MVCC 机制为例，当前活跃事务列表中最小的事务之前的事务生成的数据，其状态处于历史态。在封锁并发访问控制机制下，事务提交后，提交前的数据的值变为历史态的值，即元组的旧值处于历史态。而被读取的版本上尚有活跃事务（非最新相关事务）在使用，而由于最新相关事务修改了元组的值，其最新值已经处于一个当前态，被读取到的值相对当前态已经处于一个历史状态，因此，其数据状态介于当前态和历史态之间，所以称为过渡态。

例如，MVCC 机制下，User 表（用户表）的 A 账户余额从 10 元充值变为 20 元，然后消费了 15 元变为 5 元，此时金融 B 机构读取数据做检查事务一直进行中，A 之后又充值 20 元变为 25 元，则 25 元为当前态数据，B 正在读取到的 5 元为过渡态，其余的两个值 20、10 是历史上存在过的状态，都是历史态数据。

基于上述名词解释，需要说明的是，在 MVCC 机制下，不同的元组可以具有相同的主键（key），此时，具有相同主键的各个元组可以构成一个元组集合，该元组集合内的各个元组，在本质上是用于表示一个全态数据，也即是说，在对具有该主键的初始元组进行多次修改（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，即可构成一个元组集合。因此，在一个元组集合中，有的元组内记载了当前态数据，有的元组内记载了过渡态数据，有的元组内记载了历史态数据。这里的元组集合是指一个抽象的、虚拟的集合概念，同一个元组集合内的各个元组可以存储在不同的物理机上。

此外，还需要声明的是，在本公开实施例中，元组相对于数据读取请求可见，是指该元组针对与该数据读取请求对应的全局读事务可见，也即是说，元组的可见与否（元组的可见性）是针对于事务而言的，某个元组可能针对一些事务可见，针对一些事务不可见。

图 1 是本公开实施例提供的一种数据读取方法的实施环境示意图。参见图 1，该图 1 所提供的实施环境可以为分布式数据库系统，该系统中可以包括网关服务器 101、全局事务标识生成集群 102、分布式存储集群 103 以及分布式协调系统 104（例如 ZooKeeper）。

其中，网关服务器 101 用于接收读写请求，并基于读写请求所对应的读事务或写事务是否为全局事务，来为该读写请求对应的读事务或写事务向全局事务标识生成集群 102 申请唯

一的全局事务标识，以保证数据读写在整个分布式数据库系统中的一致性。

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合并在同一个物理机上，也即是，让某个参与读操作的节点设备充当网关服务器 101。

全局事务标识生成集群 102 用于生成全局事务标识，即 gxid，来标识全局事务，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。而采用使用集群的形式来实现该全局事务标识的生成，可以防止单点故障。当有全局事务发生的时候，可以通过网关服务器 101 向该全局事务标识生成集群 102 申请一个全局唯一的标识值。

在一些实施例中，该全局事务标识生成集群 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，对各个网关服务器 101 提供全局的全局事务标识生成服务。

其中，分布式存储集群 103 可以包括多个节点设备，且每个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。

分布式协调系统 104 可以用于对网关服务器 101、全局事务标识生成集群 102 或者分布式存储集群 103 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。

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。

基于上述实施环境，在一些实施例中，上述分布式数据库系统可以为 HTAC（hybrid transaction / analytical cluster，混合事务/分析集群）系统。图 2 是本公开实施例提供的一种 HTAC 系统的架构示意图，参见图 2，在 HTAC 系统内可以包括 TP（transaction processing，事务处理）集群 201 和 AP（analytical processing，分析处理）集群 202。

其中，该 TP 集群 201 可以用于提供事务处理服务，在 TP 集群 201 中可以包括多个 TP 节点设备，在事务处理过程中，各个 TP 节点设备用于处理当前态数据，其中，每个 TP 节点设备可以是单机设备，也可以是一主多备集群，本公开实施例不对 TP 节点设备的类型进行具体限定。

其中，该 AP 集群 202 可以用于存储历史态数据并提供历史态数据的查询及分析服务，在 AP 集群 202 中可以包括上述实施例中的全局事务标识生成集群 102 和分布式存储集群 103。在分布式存储集群 103 中可以包括多个 AP 节点设备，当然，各个 AP 节点设备可以是单机设备，也可以是一主多备集群，本公开实施例不对 AP 节点设备的类型进行具体限定。

在上述架构中，每个TP节点设备所对应主机或备机的数据库实例集合称为一个SET（集合），例如，如果某一TP节点设备为单机设备，那么该TP节点设备的SET仅为该单机设备的数据库实例，如果该TP节点设备为一主两备集群，那么该TP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（cloud database）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。

可选地，各个AP节点设备可以在本地数据库中TP集群201产生的存储历史态数据，还可以通过存储接口接入分布式文件系统203，从而可以通过该分布式文件系统203对TP集群201产生的历史态数据提供无限存储功能，例如，该分布式文件系统203可以是HDFS（Hadoop distributed file system，Hadoop分布式文件系统）、Ceph（一种Linux系统下的分布式文件系统）、Alluxio（一种基于内存的分布式文件系统）等。

在一些实施例中，由于该多个TP节点设备可以提供事务处理服务，当任一事务提交完成时，在生成新的当前态数据的同时，也会生成与该当前态数据所对应的历史态数据，而由于历史态数据会占用较多存储空间，而历史态数据又具有保存价值，因此该多个TP节点设备可以通过预定义的历史态数据迁移策略，将产生的历史态数据原子化地迁移至AP集群202，AP集群202中各个AP节点设备基于本地执行器（local executor，LE）实现历史态数据的转储，并将每次数据迁移的元信息注册到元数据（metadata，MD）管理器中，从而便于AP集群202基于MD管理器来统计已储备数据的元信息。

在一些实施例中，用户可以基于SQL路由（structured query language router，SQL Router，简称SR）层中提供的查询语句、查询操作的语义和元数据，路由查询到TP集群201或AP集群202内存储的任一数据，当然，TP集群201主要提供对当前态数据的查询服务，AP集群202则主要提供对历史态数据的查询服务。

需要说明的是，在本公开实施例提供的HTAC系统中，TP集群201和AP集群202之前可以分别配置有上述实施例中的网关服务器101，并且上述实施例中的分布式协调系统104可以用于管理整个HTAC系统，也即是由分布式协调系统104来调度TP集群201或AP集群202中的至少一项。

基于上述HTAC系统，从涉及到节点设备的个数来看，TP集群201或AP集群202的数据库内所执行的事务可以分为两类，第一类是仅涉及某一个节点设备（TP节点设备或AP节点设备）的事务，称为局部事务（也称本地事务），第二类是涉及到多个节点设备的事务（跨节点的事务），称为全局事务。当然，从事务的操作类型来看，上述事务也可以分为读事务和写事务两类，读操作对应于读事务，写操作对应于写事务。

以执行某一个全局写事务为例进行说明，HTAC等分布式数据库系统需要基于分布式一致性算法来执行该全局写事务，以保证该全局写事务所涉及到的所有TP节点设备在执行写操作时数据的原子性和一致性。可选地，该分布式一致性算法可以是2PC（two-phase commit，二阶段提交）算法，可以是XA（eXtended Architecture，X/Open组织分布式事务规范）规范（本质上也是2PC算法）等。

在上述过程中，由于分布式数据库系统中缺乏全局事务管理器，尽管通过XA/2PC技术仅能够解决分布式数据库系统中全局写事务的事务一致性，然而并不能保证全局读事务的事务一致性，下面针对全局读事务过程中可能出现的DRCC异常（Distributed Read

Committed-Committing anomaly，分布式读半已提交异常）进行详述：

假设某一个全局写事务 T，需要两个节点设备上执行写操作，以完成从节点设备 1 向节点设备 2 的转账操作，具体地，该全局写事务 T 可以通过下述语句来实现：

BEGIN GLOBAL; //申请一个全局事务标识 gxid，假设 gxid 为 20

UPDATE user_account SET my_wallet= my_wallet －10 WHERE key=100; //节点设备 1，本地事务标识 lxid 为 18

UPDATE user_account SET my_wallet= my_wallet + 10 WHERE key=900; //节点设备 2，本地事务标识 lxid 为 22

COMMIT;

节点设备 1 上的 key 为 100 的元组，其元组上的事务标识为一个二元组：{gxid，lxid}={20，18}。

节点设备 2 上的 key 为 900 的元组，其元组上的事务标识为一个二元组：{gxid，lxid}={20，22}。

通过上述事务标识，就可以识别来自不同节点设备的数据是否是同一个全局事务操作的数据，即是否属于同一个事务。而如果节点设备 1 下一个事务是全局事务，则事务标识为{gxid，lxid}={21，19}；再下一个事务是局部事务，则事务标识为{gxid，lxid}={0，20}；再下一个事务是全局事务，则事务标识为{gxid，lxid}={22，21}，依次类推。

图 3 是本公开实施例提供的一种 DRCC 异常的原理性示意图，参见图 3，假设在上述全局写事务 T 的提交过程中，产生了一个涉及到节点设备 1 和节点设备 2 的全局读事务 Q。在全局读事务 Q 进行到节点设备 1 时，由于节点设备 1 上的全局写事务 T 处于已提交状态，因此读取到的节点设备 1 上 key=100 元组 X 所对应的账号额度已经减去了 10 元（读到了 X-10），但是全局读事务 Q 进行到节点设备 2 时，由于节点设备 2 上的全局写事务 T 处于正在提交状态，也即是节点设备 2 上全局写事务 T 还未完成提交，导致读取到的节点设备 2 上 key=900 元组 Y 所对应的账号额度还未增加 10 元（读到了 Y），使得基于全局读事务 Q 读取的数值进行核对时，会发现转账数值不平衡，产生事务读取不一致的现象，这种由于分布式数据库系统采取了并发控制方法而可能引发的不一致现象，即被称为 DRCC 异常。

示意性地，在一些基于分布式数据库系统处理金融计费业务的场景，例如在执行对账业务的对账操作（本质上是一个全局读事务）时，如果对账操作与转账操作（本质上是一个全局写事务）并发执行，DRCC 异常就会普遍发生，由于读取到的元组所处的状态不同，在对账时就会发现双方账户余额之和异常，导致总账金额异常，也就是对账业务整体异常。

有鉴于此，本公开实施例提供了一种数据读取方法，能够解决分布式数据库系统中存在的以 DRCC 异常为例的全局读事务的不一致问题，保证了在分布式数据库系统中对任意时刻（或任意时间段）下全态数据执行全局读操作时的事务一致性，将在下述实施例中进行详述。

本公开实施例可以应用于上述 HTAC 等分布式数据库系统中，在执行全局写事务（也即跨节点的写操作）的过程中，可以对以 2PC 算法为例的分布式一致性算法进行改进，当协调节点设备接收到该全局写事务涉及的各个参与节点设备返回的确认提交相应时，即将该全局写事务的全局事务状态置为已提交状态，从而能够实现对全局写事务进行去中心化的事务状态管理，以便于根据各个全局事务的全局事务状态，执行本公开实施例提供的数据读取方法。

图 4 是本公开实施例提供的一种数据写入方法的交互流程图。参见图 4，在介绍数据读取方法之前，首先对数据写入方法进行说明，以作为该数据读取方法的实现前提，本公开实施例的数据写入方法可以包括下述步骤。

401、对任一全局写事务，协调节点设备向该任一全局写事务涉及的多个参与节点设备发送数据准备请求。

可选地，按照是否为该任一全局写事务的发起节点，可以将节点设备分为协调节点设备（coordinator）和参与节点设备（participants，或 cohort），其中，该协调节点设备可以是指发起该任一全局写事务的节点设备，该参与节点设备可以是指该任一全局写事务所涉及的、除了发起该全局写事务的节点设备之外的节点设备，需要说明的是，协调节点设备或参与节点设备并非是固定不变的，也即是说，对同一个节点设备而言，可能对一部分全局写事务而言是协调节点设备，对一部分全局写事务而言是参与节点设备。

在一些实施例中，应用于分布式数据库系统时，该协调节点设备可以是网关服务器，此时以协调节点设备为网关服务器为例进行说明，上述步骤 401 可以包括下述子步骤：

4011、当任一写事务涉及到跨节点操作时，网关服务器将该任一写事务确定为全局写事务，向全局事务标识生成集群发送生成请求。

在上述步骤 4011 中，数据写入业务的请求方（例如某个请求转账的金融机构）可以通过终端上的应用客户端向网关服务器发送事务的操作语句，当网关服务器接收到任一个事务的操作语句（例如 SQL 语句，英文全称：Structured Query Language，中文全称：结构化查询语言）时，网关服务器作为数据库的高级计算层可以对该事务的操作语句进行解析，当该事务的操作语句携带指定关键字时，该网关服务器可以将该事务确定为全局事务。

例如，该指定关键字可以为“GLOBAL”，用于指示该操作语句的操作对象包括数据库系统中的所有数据，也即是覆盖数据库系统中的所有节点设备，则当该操作语句中包括“GLOBAL”，代表该写事务涉及到跨节点操作，将该事务确定为全局事务。由于全局事务中可以包括全局读事务和全局写事务，在本公开实施例中仅对全局写事务进行说明，在下个实施例中将针对全局读事务的读取方法进行详述。

当然，在一些实施例中，网关服务器还可以根据该任一写事务的操作语句，确定所要写入的数据是否在同一个节点设备上，如果确定不在同一个节点设备上，则确定该任一写事务为全局写事务。具体地，网关服务器可以根据该任一写事务的操作语句所要写入数据的范围以及该范围内的元数据，确定所要写入的数据是否存储于两个及两个以上的节点设备，当确定存储于该两个及两个以上的节点设备时，代表该任一写事务涉及到跨节点操作，将该任一写事务确定为全局写事务。例如，由于元数据中会记载有数据目前的存储设备，网关服务器可以根据元数据，为每个事务的每条操作语句（例如 SQL 语句）确定所访问的节点设备，网关服务器记录所确定的节点设备，当统计到被访问的不同节点设备数大于或等于 2 时，确定该任一写事务为全局写事务。

上述某一写事务确定是否为全局写事务的过程，可以总结为基于指定关键字的识别以及由网关服务器执行的自动识别，如果某一写事务涉及到跨节点操作，则将该写事务确定为全局写事务。当网关服务器确定了某一写事务为全局写事务后，可以生成一个用于获取全局事务标识的生成请求，向全局事务标识生成集群发送该生成请求。

当然，在一些实施例中，如果事务的操作语句只涉及单个节点设备，则属于局部事务，又称为本地事务，则无需申请全局事务标识，仅为该事务分配局部事务标识即可。

4012、当全局事务标识生成集群接收到该生成请求后，全局事务标识生成集群为该全局写事务生成事务开始时刻和全局事务标识，将该事务开始时刻和全局事务标识发送至该网关服务器。

其中，全局事务标识生成集群可以是采用多副本机制，来避免发生单点故障，例如，该全局事务标识生成集群可以部署三副本、五副本或者更多副本，随着副本数量的增多，能够提升全局事务标识生成集群的可靠性，降低全局事务标识生成集群发生数据丢失的可能性。

在本公开实施例中，该全局事务标识生成集群所生成的全局事务标识的赋值随时间单调递增，其本质上是个时间戳，全局事务标识的赋值大小可以表示该全局写事务的发生时间，全局事务标识的赋值越大，其全局写事务的发生时间在已提交的全局写事务中的时序越靠后。例如，该全局事务标识可以是数值类型或者时间类型或者字符类型等能够代表时间戳的任一种形式。

4013、当接收到该事务开始时刻和全局事务标识后，网关服务器将该全局事务标识作为该全局写事务的事务标识，通知目标节点设备记录该全局写事务的全局事务标识和事务开始时刻，并通知目标节点设备将该全局写事务的全局事务状态置为正在执行状态。

在上述过程中，该目标节点设备用于存储该全局写事务的全局事务状态，该目标节点设备可以是全局写事务涉及的任一节点设备，例如，该目标节点设备为协调节点设备（例如网关服务器），当然，该目标节点设备也可以为任一个参与节点设备。

在一些实施例中，网关服务器可以通过随机选取的方式来确定目标节点设备：网关服务器从该全局写事务涉及的多个参与节点设备中随机选取目标节点设备。

例如，全局写事务 T 涉及到对三个参与节点设备 Na、Nb 和 Nc 执行写操作，则由网关服务器在这三个参与节点设备中随机选取一个作为目标节点设备，此后将由该目标节点设备来存放该全局写事务的全局事务状态，并向其他的参与节点设备同步该全局事务状态。

在上述过程中，由于在各个参与节点设备中随机指定了目标节点设备，因此能够避免各个参与节点设备之间的数据倾斜问题，均衡各个参与节点设备之间的负载。

在一些实施例中，网关服务器还可以通过指定规则来确定目标节点设备：网关服务器将该全局写事务涉及的多个参与节点设备中第一个被访问的参与节点设备确定为该目标节点设备。

例如，全局写事务 T 涉及到对三个参与节点设备 Na、Nb 和 Nc 执行写操作，则由网关服务器在将第一个被访问的参与节点设备 Na 确定为目标节点设备，此后将由该参与节点设备 Na 来存放该全局写事务的全局事务状态，并向其他的参与节点设备同步该全局事务状态。

在上述过程中，由于将第一个被访问的参与节点设备确定为目标节点设备，能够及时地获取到全局事务的提交状态，降低更新全局事务状态时可能产生的延时。

在上述两种指定目标节点设备的方式中，可以看出，不管是随机选取还是按照指定规则选取，相较于将网关服务器作为目标节点设备这种中心化的管理机制而言，将全局事务状态下放到某个参与节点设备中进行维护，能够使得分布式数据库系统在管理全局事务状态时呈现出一种去中心化的管理机制，实现了对全局事务状态管理机制的优化。而正是基于这种去中心化的管理机制，才使得后续执行全局读事务时，在建立全局读事务的全局事务快照的过程中，由于各个参与节点设备之间会同步全局事务状态，也就无需向网关服务器申请集中式的全局事务状态，避免了频繁地向网关服务器获取全局事务状态而产生的网络信令开销，而是转而从任一同步后的参与节点设备中均能够获取到全局事务状态，也就能够提升全局读事

务的性能。

在确定目标节点设备之后，网关服务器可以向目标节点设备发送第一事务状态更新指令，该第一事务状态更新指令中可以携带该全局写事务的全局事务标识、事务开始时刻和当前事务状态，其中，该事务开始时刻和全局事务标识由全局事务标识生成集群所返回，该当前事务状态也即是正在执行状态。进一步地，当目标节点设备接收到该第一事务状态更新指令，记录全局事务标识和事务开始时刻，并将全局事务状态置为正在执行状态。

在一些实施例中，在目标节点设备中，可以通过事务状态元组来维护上述全局事务标识、全局事务状态和事务开始时刻。也即是，在目标节点设备中，每个全局事务可以对应于一个事务状态元组，该事务状态元组包括该每个全局事务的全局事务标识 gxid、事务开始时刻 start_time、事务完成时刻 finish_time、全局事务状态 status 或者节点设备标识 nodes 中的至少一项。在上述情况中，当事务状态元组包括上述所有的五项时，该事务状态元组是一个五元组{gxid，start_time，finish_time，status，nodes}。

其中，gxid 已经在上述步骤 4012 中介绍过了，这里不做赘述。

其中，start_time 表示全局事务的开始时间戳，start_time 可以采用逻辑时钟、物理时钟或者混合逻辑时钟等，只要保证全局有序递增即可，在全局事务开始时记录事务的开始时间。

其中，finish_time 表示全局事务完成提交/完成回滚的时间戳，同样可以采用逻辑时钟、物理时钟或者混合逻辑时钟等，只要保证全局有序递增即可，在全局事务完成提交/完成回滚时记录事务的完成时间。

其中，status 表示全局事务当前所处的状态，包括正在执行、正在准备、正在提交、已提交、正在回滚和已回滚这六种状态，下面进行详述。

正在执行（running）状态：表示全局事务正在执行中，也即是上述步骤 4011-4013 中所处的阶段，全局写事务已经开始执行，但协调节点设备还未发送数据准备请求。

正在准备（preparing）状态：表示全局事务正处于准备阶段，也即是下述步骤 4014 中所处的阶段，协调节点设备已经发送了数据准备请求，但并未收到所有参与节点设备返回的准备成功响应或准备失败响应。

正在提交（committing）状态：表示全局事务已经完成准备阶段并正处于提交阶段，也即是下述步骤 403 中所处的阶段，协调节点设备已经发送了提交指令，但并未收到所有参与节点设备返回的提交成功响应。

已提交（committed）状态：表示全局事务已经完成全局提交，也即是所有参与节点设备都已经完成提交，协调节点设备接收到了所有参与节点设备返回的提交成功响应。

正在回滚（aborting）状态：表示全局事务正处于回滚阶段，也即是当接收到任一个参与节点设备返回的准备失败响应时，协调节点设备向所有的参与节点设备发送回滚指令，但并未收到所有参与节点设备返回的回滚成功响应。

已回滚（aborted）状态：表示全局事务已经完成了全局回滚，也即是所有参与节点设备都已经完成回滚，协调节点设备接收到了所有参与节点设备返回的回滚成功响应。

其中，nodes 表示当前全局事务所涉及到的各个节点设备（包括协调节点设备和参与节点设备）。

基于上面介绍的事务状态元组，当目标节点设备接收到第一事务状态更新指令时，可以在本地的事务状态列表中为该全局写事务创建一个事务状态元组，并将第一事务状态更新指令中携带的全局事务标识和事务开始时刻记录在该事务状态元组中，将全局事务状态置为正

在执行状态。可选地，在第一事务状态更新指令中还可以携带节点设备标识，从而目标节点设备可以在事务状态元组中记录节点设备标识。当然，由于该全局写事务尚未完成，因此在事务状态元组中可以暂且将事务完成时刻置为空或置为初始值。

4014、网关服务器向该全局写事务涉及的多个参与节点设备发送数据准备请求，通知目标节点设备将该全局写事务的全局事务状态置为正在准备状态。

图 5 是本公开实施例提供的一种传统数据库系统元组结构的示意图，如图 5 所示，对于数据库系统来说，传统的数据库系统（例如 MySQL/InnoBD 等）中的元组结构中可以包括记录头、主键、事务 ID（transaction identification，也即是事务标识，通常为 6 个字节）、roll_ptr（指向该主键所对应的前一元组的指针，通常为 7 个字节）以及主键外的其他非空字段。

图 6 是本公开实施例提供的一种分布式数据库系统元组结构的示意图，在本公开实施例中的分布式数据库系统中，如果支持 MVCC 技术，可以对数据准备请求中传输的元组进行结构调整，调整后的元组结构如图 6 所示，在一个元组中包括系统字段和用户字段，在系统字段中可以包括 trx_min（表示产生该元组的事务信息）、trx_max（表示将该元组修改为历史态元组的事务信息）、info_bit（表示该元组的操作状态）以及 roll_ptr（指向元组所对应主键的前一元组的指针），在用户字段中可以包括 col.1 至 col.n 共 n 个由用户定义的数据项，其中，n 为任一大于或等于 1 的整数。

在上述调整后的元组结构中，元组上记录的事务信息（不管是 trx_min 还是 trx_max）可以由一个三元组{gxid，lxid，node}来表示，其中，gxid 为全局事务标识，lxid 为本地事务标识，node 用于表示为事务信息当前存储在哪些节点设备上，从而用来快速定位出事务信息在分布式数据库系统中的存储位置。其中，本地事务标识和全局事务标识的赋值可以都是单调递增的。

基于上述元组结构，在上述步骤 4014 中，网关服务器可以将全局事务标识生成集群返回的全局事务标识，写入元组结构中 trx_min 所对应三元组事务信息中的 gxid，并向该全局写事务涉及的多个参与节点设备发送数据准备请求，等待各个参与节点设备返回准备成功响应或准备失败响应，从而可以根据返回的响应来判断是否需要发起该全局事务的提交流程。需要说明的是，该数据准备请求是根据全局写事务对各个参与节点设备所要求的写操作来确定的，不同参与节点设备通常对应于不同的数据准备请求。

当网关服务器向所有的参与节点设备均发送完毕数据准备请求后，可以向目标节点设备发送第二事务状态更新指令，该第二事务状态更新指令中可以携带全局事务标识和当前事务状态（也即是正在准备状态），当目标节点设备接收到第二事务状态更新指令，响应于该第二事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中将全局事务状态从正在执行状态置为正在准备状态。

在上述步骤 4011-4014 中，以协调节点设备为网关服务器为例，协调节点设备向各个参与节点设备发送数据准备请求之前，如何基于去中心化的管理机制，在目标节点设备上维护全局事务状态（例如以事务状态元组的形式来维护）进行了阐述，但是需要说明的是，本公开实施例虽然是以全局写事务为例进行说明，但在分布式数据库系统中，不仅对全局写事务可以维护全局事务状态，还可以对全局读事务也维护全局事务状态，从而能够完整地对所有的全局事务的全局事务状态进行记录。

在一些实施例中，分布式数据库系统中各个目标节点设备可以仅对全局写事务的全局事务状态进行永久维护，而对全局读事务的全局事务状态只在事务执行过程中进行维护，当全

局读事务提交完毕/回滚完毕后，删除全局读事务的全局事务状态。在这种情况中，由于只有写事务会对数据库中数据的状态进行改变，从而省略了对全局读事务的全局事务状态的永久维护，而只是在事务执行过程中进行暂时性地维护，能够节约各个目标节点设备的本地存储空间。

402、当各个参与节点设备接收到该数据准备请求时，对该任一全局写事务进行数据写入的准备操作，当准备完毕后，各个参与节点设备向协调节点设备返回准备成功响应。

以多个参与节点设备中任一参与节点设备为例进行说明，该参与节点设备接收到数据准备请求（prepare 请求）时，进入该全局写事务的准备阶段，对该全局写事务所要求写入的数据，将该数据的 Undo（回退）信息和 Redo（重做）信息写入本地临界区的事务日志中，但不提交该事务，当将 Undo 信息和 Redo 信息写入成功时，向协调节点设备返回准备成功响应。当然，在一些实施例中，如果将 Undo 信息和 Redo 信息写入失败时，可以向协调节点设备返回准备失败响应。

403、当接收到该多个参与节点设备中每个参与节点设备所返回的准备成功响应时，协调节点设备向该多个参与节点设备发送提交指令，通知目标节点设备记录该任一全局写事务的事务完成时刻，并通知目标节点设备将该任一全局写事务的全局事务状态置为已提交状态。

在上述过程中，协调节点设备接收各个参与节点设备返回的准备成功响应或准备失败响应，当所有的参与节点设备均返回了准备成功响应时，可以视为全局准备成功，协调节点设备可以通过执行上述步骤 403，向所有的参与节点设备发送提交指令（commit 指令），从而进入 2PC 算法的提交阶段，同时向全局事务标识生成集群申请一个事务完成时刻（finish_time，本质上是一个时间戳），向目标节点设备发送第三事务状态更新指令，该第三事务状态更新指令中携带全局事务标识、事务完成时刻和当前事务状态（也即是已提交状态），当目标节点设备接收到该第三事务状态更新指令，响应于该第三事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中记录该第三事务状态更新指令中携带的事务完成时刻，并将全局事务状态从正在准备状态置为已提交状态。

在一些实施例中，当接收到该多个参与节点设备中任一参与节点设备所返回的准备失败响应时，协调节点设备可以向该多个参与节点设备发送回滚指令（abort 指令），通知目标节点设备将该任一全局写事务的全局事务状态置为正在回滚状态，当接收到该多个参与节点设备中每个参与节点设备所返回的回滚完成响应时，通知目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已回滚状态。

在上述过程中，只要有任一个或一个以上的参与节点设备返回了准备失败响应，就可以认为全局准备失败，协调节点设备执行上述发送回滚指令的操作，同时可以向目标节点设备发送第四事务状态更新指令，该第四事务状态更新指令中携带全局事务标识和当前事务状态（也即是正在回滚状态），当目标节点设备接收到该第四事务状态更新指令，响应于该第四事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中将全局事务状态从正在准备状态置为正在回滚状态。

进一步地，等待各个参与节点设备返回回滚完成响应，当所有的参与节点设备均返回了回滚完成响应时，向全局事务标识生成集群申请一个事务完成时刻，向目标节点设备发送第五事务状态更新指令，该第五事务状态更新指令中携带全局事务标识、事务完成时刻和当前

事务状态（也即是已回滚状态），当目标节点设备接收到该第五事务状态更新指令，响应于该第五事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中记录该第五事务状态更新指令中携带的事务完成时刻，并将全局事务状态从正在回滚状态置为已回滚状态。

404、当对任一个全局写事务提交完成时，各个参与节点设备向该任一个全局写事务对应的目标节点设备发送数据同步请求，接收该目标节点设备返回的全局事务状态，该目标节点设备用于存储该任一个全局写事务的全局事务状态。

在上述过程中，对任一参与节点设备，每当该参与节点设备对一个全局写事务提交完成，即向该全局写事务对应的目标节点设备发送数据同步请求，由于目标节点设备可以通过事务状态元组维护全局事务状态，当目标节点设备接收到数据同步请求后，可以根据该数据同步请求在本地的事务状态列表中查询对应的事务状态元组，向该参与节点设备返回该事务状态元组，该参与节点设备接收该事务状态元组，并对该事务状态元组进行持久化存储，例如可以存储到本地磁盘中，也可以存储到云端数据库中。

上述步骤 404 中，各个参与节点设备当事务提交完成时，立即执行全局事务状态的实时同步，能够缩短全局事务状态同步过程的网络延时，使得各个参与节点设备快速地、实时地获取到已完成事务的全局事务状态。

在一些实施例中，各个参与节点设备还可以在事务提交完成时，不会立即执行全局事务状态的同步，而是通过下述方式实现全局事务状态的延时同步：对任一参与节点设备，如果当前执行的事务涉及到该任一个全局写事务对应的该目标节点设备，该参与节点设备与该目标节点设备之间进行增量同步；或，每间隔目标时长，该参与节点设备与该目标节点设备之间进行增量同步。

在前一种延时同步的方式中，参与节点设备在与目标节点设备之间产生必要通信时，参与节点设备将当前执行的事务（或读或写）之前已提交的、与该目标节点设备对应的全局写事务的全局事务状态同步至本地，相应地，由于参与节点设备也有可能是另一全局写事务的目标节点设备，因此目标节点设备也会将参与节点设备中存储的一部分全局事务状态同步至本地。

也即是说，当参与节点设备与目标节点设备之间产生必要通信时，双方之间互相发送待同步的事务状态元组，从而实现了参与节点设备与目标节点设备之间批量且双向的数据同步。可选地，在每次同步完成后，可以设定检查点（checkpoint），从而在下次执行数据同步时可以进行增量同步，这样的同步方式中，由于全局事务状态的增量同步是随着当前执行的事务一起发送的，也就能够避免在事务的提交阶段产生一轮额外的通信，降低了全局事务状态在同步过程的网络信令开销。

在后一种延时同步的方式中，参与节点设备与目标节点设备之间每间隔目标时长进行一次增量同步，其中，目标时长可以任一大于 0 的数值，该目标时长可以由技术人员通过调度器来进行设置，增量同步的方式与上一种方式类似，这里不做赘述。通过这种定时轮询的延时同步方式，能够定期触发全局事务状态的增量同步，能够避免当参与节点设备与目标节点设备之间长时间没有产生必要通信时，导致上一个全局事务的全局事务状态迟迟无法同步到本地的延时过长问题。

在上述各种方式中，由于事务全局提交后，一开始是只有目标节点设备上存储有该全局写事务的全局事务状态，为了让各个参与节点设备上均存储该全局事务状态，不管是通过实

时同步还是通过延时同步方式，各个参与节点设备均能够将该全局事务状态同步至本地，也即是说，对于已经全局提交的事务，通过同步机制能够使得每个参与节点设备中均存储有上述已经全局提交的事务的全局事务状态。从而在后续执行全局读事务的时候，可以从任一个参与节点设备上获取到该全局事务状态，能够在局部的参与节点设备的本地加速事务的并发判断，避免额外的网络信令开销。

例如，全局写事务 T 涉及到参与节点设备 Na、Nb、Nc、Nd，Na 为目标节点设备，当该全局写事务 T 全局提交后，目标节点设备 Na 将全局写事务 T 的事务状态元组（其中包括全局写事务 T 的全局事务状态）同步至参与节点设备 Nb、Nc 和 Nd，技术人员基于调度器可以定义节点设备间的同步策略，比如通过参数控制来定义为实时同步或者某种延时同步等。

需要说明的是，对任一参与节点设备，当该参与节点设备将某一全局事务的全局事务状态同步至本地后，刚同步得到的全局事务状态立即生效，该生效后的全局事务状态可以立即应用于进行事务的并发判断。在一些实施例中，如果待并发判断的事务的全局事务状态处于并未完成同步的状态（也即是本地查询不到相关的全局事务状态），那么需要该参与节点设备向远程的待同步节点设备获取该事务的全局事务状态，在下个实施例中会以待并发判断的事务为全局读事务为例进行详细介绍。

在上述步骤 401-404 中，可以看出，在全局写事务完成全局提交之前，对全局事务状态的更新是一个单向同步写入的过程，由协调节点设备单向的、同步的通知目标节点设备对全局事务状态进行更新。

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。

在传统的 2PC 算法中，当所有参与节点设备均向协调节点设备返回准备成功响应时，该全局写事务会进入正在提交（committing）状态，而当所有参与节点设备均向协调节点设备返回提交完成响应时，该全局写事务才进入已提交（committed）状态。

而在本公开实施例中，对传统的 2PC 算法进行了改进，当所有参与节点设备均向协调节点设备返回准备成功响应时，协调节点设备即直接认为该全局写事务进入已提交（committed）状态，从而通知目标节点设备将该全局写事务的全局事务状态置为已提交状态，向各个参与节点设备下发提交指令，在这样的算法逻辑下，在事务的 2PC 提交阶段完成后，事务所操作的数据就已经被写入了各个参与节点设备中并实现持久化存储，因为提前将全局事务状态置为已提交状态是可行的。通过上述改进方式，能够将 2PC 算法的提交阶段转化为一个异步的过程，简化了事务的提交流程，提高了事务的处理效率，有利于后续保障全局读事务的数据一致性。

本公开实施例提供的方法，对任一个全局写事务，当所有的参与节点设备均向协调节点设备返回准备成功响应时，协调节点设备向全局事务标识生成集群申请事务完成时刻，并通知目标节点设备记录事务完成时刻、将全局事务状态置为已提交状态，从而将 2PC 算法的提交阶段转化为一个异步的过程，简化了事务的提交流程，提高了事务的处理效率，有利于后续保障全局读事务的数据一致性。

进一步地，当事务完成全局提交后，目标节点设备与各个参与节点设备之间进行全局事务标识的同步，能够在后续执行全局读事务的时候，从任一个参与节点设备上获取到该全局

事务状态，能够在局部的参与节点设备的本地加速事务的并发判断，避免额外的网络信令开销。

进一步地，在实时同步的情况下，能够缩短全局事务状态同步过程的网络延时，使得各个参与节点设备快速地、实时地获取到已完成事务的全局事务状态。

进一步地，在延时同步的情况下，如果在产生必要通信时进行增量同步，能够避免在事务的提交阶段产生一轮额外的通信，降低了全局事务状态在同步过程的网络信令开销。或者，如果每间隔目标时长进行增量同步，能够避免当参与节点设备与目标节点设备之间长时间没有产生必要通信时，导致上一个全局事务的全局事务状态迟迟无法同步到本地的延时过长问题。

进一步地，协调节点设备在各个参与节点设备中指定一个目标节点设备，从而将全局事务状态下放到该目标节点设备中进行维护，能够使得分布式数据库系统在管理全局事务状态时呈现出一种去中心化的管理机制，实现了对全局事务状态管理机制的优化，使得后续执行全局读事务时，在建立全局读事务的全局事务快照的过程中，避免了频繁地向网关服务器获取全局事务状态而产生的网络信令开销，能够提升全局读事务的性能。

进一步地，协调节点设备在各个参与节点设备中随机指定目标节点设备，能够避免各个参与节点设备之间的数据倾斜问题，均衡各个参与节点设备之间的负载。或者，协调节点设备将第一个被访问的参与节点设备确定为目标节点设备，能够及时地获取到全局事务的提交状态，降低更新全局事务状态时可能产生的延时。

进一步地，在目标节点设备中，以事务状态元组的方式来维护全局事务状态，在事务状态元组中记录全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项，大大方便了对某一个全局事务的各项信息进行统一的管理，优化了对全局事务所对应的各项信息的存储形式。

在上述实施例中，对分布式数据库系统中的数据写入过程进行说明，基于上述实施例中对全局写事务执行的改进后的 2PC 算法，以及上述实施例中去中心化的全局事务状态的管理机制，在这种前提下，对本公开实施例提供的数据读取方法进行详述。

图 7 是本公开实施例提供的一种数据读取方法的交互流程图，参见图 7，本实施例包括下述步骤。

701、当获取到全局读事务时，协调节点设备向该全局读事务涉及的多个参与节点设备发送数据读取请求。

其中，这里的协调节点设备是指发起该全局读事务的节点设备，例如，该协调节点设备可以为分布式数据库系统中的网关服务器。这里的参与节点设备可以是指该全局读事务所涉及的、除了发起该全局读事务的节点设备之外的节点设备。

需要说明的是，在本公开实施例中的协调节点设备或参与节点设备，仅仅是因为针对某一个全局读事务的身份不同而进行的区别称呼，分布式数据库系统中的任一个节点设备都可能成为协调节点设备或参与节点设备，不同的全局事务对应于不同的协调节点设备或参与节点设备，同一个节点设备也可以既是一些事务的协调节点设备，又是另一些事务的参与节点设备。另外，由于本公开实施例是以任一个全局读事务为例进行说明，因此，本公开实施例的协调节点设备与上述实施例中的协调节点设备并不要求一定是同一个节点设备，参与节点设备也并不要求一定是相同的多个节点设备，由事务本身的性质来决定。

上述步骤 701 与上述步骤 401 类似，也即是说，该步骤 701 可以执行下述子步骤：

7011、当任一读事务涉及到跨节点操作时，协调节点设备将该任一读事务确定为全局读事务，向全局事务标识生成集群发送生成请求。

以协调节点设备为网关服务器为例，数据读取业务的请求方（例如某个请求查账的金融机构）通过终端上的应用客户端向网关服务器发送读事务的操作语句，当网关服务器接收到任一个读事务的操作语句时，对该读事务的操作语句进行解析，当该读事务的操作语句携带指定关键字时，将该读事务确定为全局读事务。例如，该指定关键字可以为“GLOBAL”。

可选地，网关服务器还可以通过下述方式确定出全局读事务：网关服务器根据该读事务的操作语句所要读取数据的范围以及该范围内的元数据，确定所要读取的数据是否存储于两个及两个以上的节点设备，当确定存储于该两个及两个以上的节点设备时，代表该读事务涉及到跨节点操作，将该读事务确定为全局读事务。

上述步骤 7011 中发送生成请求的过程与上述步骤 4011 中发送生成请求的过程类似，这里不做赘述。

7012、当全局事务标识生成集群接收到该生成请求后，全局事务标识生成集群为该全局读事务生成事务开始时刻和全局事务标识，将该事务开始时刻和全局事务标识发送至该协调节点设备。

上述步骤 7012 与上述步骤 4012 类似，这里不做赘述。

7013、当接收到该事务开始时刻和全局事务标识后，协调节点设备将该全局事务标识作为该全局读事务的事务标识，通知目标节点设备记录该全局读事务的全局事务标识和事务开始时刻，并通知目标节点设备将该全局读事务的全局事务状态置为正在执行状态。

上述步骤 7013 与上述步骤 4013 类似，这里不做过多的阐述。

以协调节点设备为网关服务器为例，网关服务器可以通过随机选取的方式确定目标节点设备，从而避免各个参与节点设备之间的数据倾斜问题，均衡各个参与节点设备之间的负载。或者，网关服务器还可以通过指定规则来确定目标节点设备，例如将第一个被访问的参与节点设备确定为目标节点设备，从而能够及时地获取到全局事务的提交状态，降低更新全局事务状态时可能产生的延时。

同理，目标节点设备上也可以以事务状态元组的方式来维护全局读事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项，大大方便了对某一个全局事务的各项信息进行统一的管理，优化了对全局事务所对应的各项信息的存储形式。

同理在确定目标节点设备之后，网关服务器可以向目标节点设备发送第六事务状态更新指令，该第六事务状态更新指令中可以携带该全局读事务的全局事务标识、事务开始时刻和当前事务状态，其中，该事务开始时刻和全局事务标识由全局事务标识生成集群所返回，该当前事务状态也即是正在执行状态。

进一步地，当目标节点设备接收到该第一事务状态更新指令，在本地的事务状态列表中为该全局读事务创建一个事务状态元组，并将第一事务状态更新指令中携带的全局事务标识和事务开始时刻记录在该事务状态元组中，将全局事务状态置为正在执行状态。

可选地，在第一事务状态更新指令中还可以携带节点设备标识，从而目标节点设备可以在事务状态元组中记录节点设备标识。当然，由于该全局读事务尚未完成，因此在事务状态元组中可以暂且将事务完成时刻置为空或置为初始值。

7014、协调节点设备向该全局写事务涉及的多个参与节点设备发送数据准备请求，通知目标节点设备将该全局写事务的全局事务状态置为正在准备状态。

其中，数据准备请求中传输的元组也可以具有如图 6 所示调整后的元组结构。

上述步骤 7014 与上述步骤 4014 类似，这里不做过多的阐述。

以协调节点设备为网关服务器为例，网关服务器可以将全局事务标识生成集群返回的全局事务标识，写入元组结构中 trx_min 所对应三元组事务信息中的 gxid，并向该全局读事务涉及的多个参与节点设备发送数据准备请求，当发送完毕后，向目标节点设备发送第七事务状态更新指令，该第七事务状态更新指令中可以携带全局事务标识和当前事务状态（也即是正在准备状态），当目标节点设备接收到第七事务状态更新指令，响应于该第七事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中将全局事务状态从正在执行状态置为正在准备状态。

702、当接收到该数据准备请求时，各个参与节点设备根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组。

其中，该读取条件用于指示待读取的数据，例如，该读取条件可以是读取某一个用户表（user 表）中的所有数据，又比如，该读取条件可以是读取某一个主键区间内的数据，例如读取 key 位于[100,1000]范围内的数据。

在上述过程中，各个参与节点设备对数据读取请求进行解析，得到读取条件，从而在数据库中查询符合该读取条件的多个元组。例如在 HTAC 架构中，TP 集群维护当前态数据或过渡态数据对应的元组，AP 集群维护历史态数据对应的元组，协调节点设备分别会向 TP 集群和 AP 集群的参与节点设备发送数据读取请求，各个集群的各个参与节点设备各自查询符合上述读取条件的多个元组。

703、对该多个元组中的任一元组上记录的全局事务标识，各个参与节点设备从本地的事务状态列表中，查询是否存在与该全局事务标识所对应的全局事务状态。

在上述过程中，如图 6 所示，可以看到每个元组结构的系统字段上会记录有两个事务的事务信息，一个是产生该元组的事务信息 trx_min，一个是将该元组修改为历史态元组的事务信息 trx_max，各个事务的事务信息分别以{gxid，lxid，node}三元组来表示。由于对当前态元组而言，还未被修改为历史态元组，因此元组结构中的 trx_max 可能暂且为空，但每个元组上会记录有至少一个事务的事务信息。

有鉴于此，对各个元组上记录的至少一个事务中任一事务而言，各个参与节点设备可以从该事务的事务信息中，获取该事务的全局事务标识，从本地的事务状态列表中，以该事务的全局事务标识为索引，查询是否能够命中该事务状态列表中的任一事务状态元组，执行下述步骤 704。

需要说明的是，当该事务的全局事务标识为 0 时，说明该事务不是全局事务，那么将不会引发全局读操作中可能出现的 DRCC 异常，则无需执行下述各个步骤，直接按照传统的单机数据库系统的可见性判断算法进行可见性判断即可，这里不做赘述，在本公开实施例中，仅聚焦于该事务为全局事务的情况。

704、当存在时，各个参与节点设备获取与该全局事务标识所对应的全局事务状态；当不存在时，各个参与节点设备向该全局事务标识对应的多个节点设备发送查询请求，接收该多个节点设备中任一节点设备返回的全局事务状态。

在上述过程中，当能够命中该事务状态列表中的任一事务状态元组时，认为存在与该全

局事务标识对应的全局事务状态，各个参与节点设备从该事务状态元组中，将 status 的当前值获取全局事务状态。

当然，如果不能命中该事务状态列表中的任一事务状态元组，则认为不存在与该全局事务标识对应的全局事务状态，各个参与节点设备可以从该事务的事务信息三元组{gxid，lxid，node}中，将 node 中记载的节点设备确定为该全局事务标识对应的多个节点设备，向该多个节点设备发送携带该全局事务标识的查询请求。当该多个节点设备接收到该查询请求后，执行类似的查询过程，以该全局事务标识为索引，在本地的事务状态列表中查询是否能够命中任一事务状态元组，当命中时，向查询请求的发送方返回该事务状态元组，当不能命中时，向查询请求的发送方返回查询失败响应或者不返回任何响应。

在一种可能实施方式中，各个节点设备本地的事务状态列表中，可以按照全局事务标识从小到大的顺序进行有序地存储，因此在查询时可以通过二分查找等算法来进行查询，以提升在事务状态列表中的查询效率。

对于各个参与节点设备而言，由于本地的事务状态列表中不存在对应的事务状态元组，存在两种可能产生上述结果的情况，一种情况是该全局事务标识所指示的全局事务还未全局提交，另一种情况则是该全局事务标识所指示的全局事务已经完成了全局提交，但是当前的参与节点设备还没有完成数据同步。因此，通过向各个节点设备发送查询请求，能够判断出该全局事务标识对应的全局事务究竟属于上述两种中的哪一种情况。

进一步地，对于各个参与节点设备而言，无需等待所有的节点设备均返回全局事务状态（或事务状态元组），而是只要有任一个节点设备返回了全局事务状态，那么即可执行下述步骤 705。这是因为一旦有任一个节点设备能够返回全局事务状态，即证明该事务已经完成全局提交并在该节点设备上进行了数据同步，也就不必等待后续其他节点设备返回全局事务状态。

在一些实施例中，还有一种可能是事务尚未全局提交，此时仅有事务所对应的目标节点设备会向参与节点设备返回全局事务状态，其他节点设备均不会返回全局事务状态（未全局提交之前不会执行数据同步），依然执行下述步骤 705。

在一些实施例中，如果所有的节点设备均没有返回全局事务状态，则各个节点设备需要重启该全局读事务，直到有某个节点设备返回了全局事务状态。

在上述过程中，参与节点设备向多个节点设备广播（或组播）查询请求，从而获取到了全局事务的全局事务状态，能够避免因某个节点设备宕机而产生的故障，当然，在一些实施例中，参与节点设备还可以每次仅向各全局事务的目标节点设备发送查询请求，当目标节点设备宕机时，再执行向所有节点设备广播查询请求的操作。

在上述步骤 703-704 中，是以某一个全局事务为例进行说明，各个参与节点设备对每个元组上记录的每个全局事务重复执行上述步骤 703-704，从而能够在去中心化的全局事务状态的管理机制下，获取该多个元组所对应的多个全局事务的全局事务状态。

可以看出，在上述步骤 703-704 中，各个参与节点设备是从分布式数据库系统中的一个或多个节点设备上获取到全局事务状态，能够避免频繁地网络信令开销，在一些实施例中，也可以由网关服务器来集中式的存储各个全局事务的全局事务状态，这种中心化的管理机制下则能够实现简化获取全局事务状态的流程。

705、各个参与节点设备根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻。

在上述过程中，数据读取请求中可以携带一个指定时刻，也可以携带一个指定时间段。当数据读取请求中携带指定时刻时，该数据读取请求用于查询在该指定时刻符合一致性的元组，例如在数据读取请求中以 SYSTEM_TIME AS OF time_expr 来标识该指定时刻。当数据读取请求中携带指定时间段时，该数据读取请求用于查询从起始时刻至终止时刻的时间段内符合一致性的元组，例如在数据读取请求中以 SYSTEM_TIME BETWEEN time_expr AND time_expr2，来标识从起始时刻 time_expr 至 time_expr2 的时间段。

在本公开实施例中，将对指定时刻或指定时间段这两种情况下，如何获取全局提交时刻（commit_Ts）分别进行说明。需要声明的是，这里获取的 commit_Ts 是一个以供可见性判断算法来判断元组可见性的数值，根据全局事务的全局事务状态，在不同的情况下为 commit_Ts 进行不同的赋值，每个全局事务唯一对应于一个 commit_Ts。

1、当该数据读取请求中携带指定时刻时，对该多个元组中任一元组所对应的任一全局事务，各个参与节点设备中任一参与节点设备均可以通过下述方式获取全局提交时刻：

（1）如果该任一全局事务在该指定时刻的全局事务状态为已提交状态或正在提交状态，当该任一全局事务的事务完成时刻大于该指定时刻时，参与节点设备将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值；当该任一全局事务的事务完成时刻不大于该指定时刻时，参与节点设备将该任一全局事务的全局提交时刻置为该事务完成时刻。

在上述过程中，例如，假设用 Ts 表示数据读取请求中携带的指定时刻，该参与节点设备获取到该全局事务的事务状态元组后，确定该事务状态元组{gxid，start_time，finish_time，status，nodes}内 status 的当前值为 committed 状态或 committing 状态，该参与节点设备判断是否 finish_time＞Ts。

这里执行判断逻辑，是因为 status 的当前值仅能代表对当前时刻而言的全局事务状态是已提交状态，而指定时刻 Ts 并不一定是当前时刻（Ts≤当前时刻），因此需要判断在执行时刻 Ts 时该全局事务处于什么样的全局事务状态。

如果 finish_time＞Ts，说明该全局事务在指定时刻 Ts 时，实际上正处于 running 状态或者 preparing 状态，因此在指定时刻 Ts 时该全局事务尚未全局提交，则该参与节点设备将 commit_Ts 置为 Ts+n，其中，n 为任一正数。

反之，如果 finish_time≤Ts，说明该全局事务在指定时刻 Ts 时，实际上正处于 committing 状态或者 commited 状态，由于在上个实施例中对 2PC 算法进行了改进，也即是在全局写事务的 committing 阶段提取确认了全局提交完毕，因此只要 finish_time≤Ts，即认为该全局事务在指定时刻 Ts 时全局提交完毕，则将 commit_Ts 置为 finish_time。

（2）如果该任一全局事务在该指定时刻的全局事务状态为正在执行状态或正在准备状态，参与节点设备将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值。

基于上述示例中的假设，如果事务状态元组{gxid，start_time，finish_time，status，nodes}内 status 的当前值为 running 状态或者 preparing 状态，说明该全局事务在当前时刻都处于 running 状态或者 preparing 状态，因此在指定时刻 Ts 时一定没有全局提交完毕，参与节点设备直接将 commit_Ts 置为 Ts+n，其中，n 为任一正数。

（3）如果该任一全局事务在该指定时刻的全局事务状态为已回滚状态或正在回滚状态，参与节点设备将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数

值。

基于上述示例中的假设，如果事务状态元组{gxid，start_time，finish_time，status，nodes}内 status 的当前值为 aborting 状态或者 aborted 状态，该参与节点设备同样判断是否 finish_time＞Ts。其中，执行判断逻辑的理由和上述（1）中类似，这里不做赘述。

如果 finish_time＞Ts，说明该全局事务在指定时刻 Ts 时，实际上正处于 running 状态或者 preparing 状态或者 aborting 状态，因此在指定时刻 Ts 时该全局事务尚未全局提交，则该参与节点设备将 commit_Ts 置为 Ts+n，其中，n 为任一正数。

反之，如果 finish_time≤Ts，说明该全局事务在指定时刻 Ts 时，实际上正处于 aborted 状态，也即是该全局事务在指定时刻 Ts 时全局回滚完毕，则仍然将 commit_Ts 置为 Ts+n，其中，n 为任一正数。

经过上面的分析可以看出，上述（2）和（3）可以总结为，如果该全局事务在指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，参与节点设备均将该全局事务的全局提交时刻置为指定时刻与任一正数相加所得到的数值。

在上述（1）-（3）所描述的方式中，对 status 可能取值的全部的六种状态均说明了如何为 commit_Ts 赋值。由于在分布式数据库系统中，如果某一全局写事务没有全局提交，那么该全局写事务的全局事务状态可能会在全局读事务的执行过程中被更新，从而会引发 DRCC 异常，而通过上述任一种方式，能够在分布式数据库系统中找到在指定时刻 Ts 下处于符合一致性条件的全局事务状态，并通过为 commit_Ts 赋值来将这种在指定时刻 Ts 下符合一致性条件的全局事务状态进行量化，以供后续可见性判断算法直接通过比较数值大小的方式来判断元组的可见性，也即能够保证全局读操作的事务一致性。

2、当该数据读取请求中携带指定时间段时，对该多个元组中任一元组所对应的任一全局事务，各个参与节点设备中任一参与节点设备基于该指定时间段的起始时刻和终止时刻，分别执行获取各自对应的全局提交时刻的操作。

可选地，该参与节点设备可以基于该指定时间段的起始时刻，执行获取该多个全局事务在该起始时刻下的全局提交时刻的操作；基于该指定时间段的终止时刻，执行获取该多个全局事务在该终止时刻下的全局提交时刻的操作。

具体地，在获取起始时刻下的全局提交时刻时，可以将指定时间段的起始时刻视为上述过程中的“指定时刻 Ts”，从而通过执行上述（1）-（3）所提供的方式，获取到该起始时刻下 commit_Ts 的赋值，同理，将指定时间段的终止时刻视为上述过程中的“指定时刻 Ts”，从而通过执行上述（1）-（3）所提供的方式，获取到该终止时刻下 commit_Ts 的赋值。

在上述情况中，由于各事务的全局事务状态是一个有向的变化过程，因此可以将获取整个指定时间段内 commit_Ts 的复杂操作，转化为获取指定时间段的起始时刻下 commit_Ts 以及终止时刻下 commit_Ts 的简单操作，通过判断事务在指定时间段的起始时刻和终止时刻下的 commit_Ts，即能够标识出事务在该指定时间段内符合一致性条件的全局事务状态，对该指定时间段内符合一致性条件的全局事务状态进行量化，以供后续可见性判断算法直接通过比较数值大小的方式来判断元组的可见性，也即能够保证全局读操作的事务一致性。

在上述两种情况中，分别从按时间点（指定时刻）读取数据以及按时间段（指定时间段）读取数据的角度，介绍了各个参与节点设备如何根据全局事务状态获取全局提交时刻，通过全局提交时刻对符合一致性条件的全局事务状态进行量化，相当于提供了一种分布式数据库系统下的事务状态提取算法，基于该事务状态提取算法能够提取出某一时间点或某一时间段

下的全局事务状态（以commit_Ts的赋值来表示），进一步地，各个参与节点设备通过执行下述步骤706，利用一种去中心化的全时态可见性判断算法，判断出多个元组中的目标元组。

706、各个参与节点设备基于该多个全局事务的全局提交时刻，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。

其中，元组相对于数据读取请求可见，是指该元组针对与该数据读取请求对应的全局读事务可见，也即是说，元组的可见性是针对于事务而言的，也即是说，某个元组可能针对一些事务可见，针对一些事务不可见，因此，该目标元组是指针对上述步骤701中的数据读取请求可见的元组。

在本公开实施例中，将对指定时刻或指定时间段这两种情况下，各个参与节点设备中任一参与节点设备如何判断任一元组的可见性进行分别说明：

1、当该数据读取请求中携带指定时刻时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时刻，该参与节点设备将该任一元组确定为一个目标元组。

在上述过程中，由于某一元组对某一指定时刻可见，相当于要求该指定时刻能够落入该元组的有效时间范围内，也即是说，对任一元组来说，判读该元组相对于指定时刻Ts可见需要满足两个条件：（1）产生该元组的事务对于指定时刻已经全局提交；（2）修改该元组的事务对于指定时刻尚未全局提交。

在上述步骤705中，对各个元组所对应的各个事务的commit_Ts已经完成了赋值，因此，上述两个条件可以相应地转化为下述两个不等式。

条件（1）：v.trx_min.commit_ts＜Ts，即确保产生该元组的事务的全局提交时刻小于指定时刻Ts，从而保证产生该元组的事务对于指定时刻已经全局提交。

条件（2）：v.trx_max.commit_ts＞Ts，即确保修改该元组的事务的全局提交时刻大于指定时刻Ts，从而保证修改该元组的事务对于指定时刻尚未全局提交。

在上述过程中，当产生该元组的事务trx_min的commit_Ts和修改该元组的事务trx_max的commit_Ts同时满足上述两个不等式所构成的不等式组时，即可认为元组v相对于指定时刻Ts可见，将该元组v确定为目标元组。也即是说，各个参与节点设备根据指定时刻和元组上各事务的事务信息，判断出了元组是否为目标元组。

2、当该数据读取请求中携带指定时间段时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时间段的终止时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时间段的起始时刻，该参与节点设备将该任一元组确定为一个目标元组。

在上述过程中，由于某一元组对某一指定时间段可见，相当于要求该指定时间段能够与该元组的有效时间范围相交（存在交集），也即是说，对任一元组来说，判断该元组相对于指定时间段Ts1~Ts2可见需要满足两个条件：（1）产生该元组的事务对于指定时间段的终止时刻Ts2已经全局提交；（2）修改该元组的事务对于指定时间段的起始时刻Ts1尚未全局提交。

在上述步骤705中，对各个元组所对应的各个事务的commit_Ts已经完成了赋值，因此，上述两个条件可以相应地转化为下述两个不等式。

条件（1）：v.trx_min.commit_ts＜Ts2，即确保产生该元组的事务的全局提交时刻小于指定时间段的终止时刻Ts2，从而保证产生该元组的事务对于指定时间段的终止时刻Ts2已经全局提交。

条件（2）：v.trx_max.commit_ts＞Ts1，即确保修改该元组的事务的全局提交时刻大于指定时间段的起始时刻Ts1，从而保证修改该元组的事务对于指定时间段的起始时刻Ts1尚未全局提交。

在上述过程中，当产生该元组的事务trx_min的commit_Ts和修改该元组的事务trx_max的commit_Ts同时满足上述两个不等式所构成的不等式组时，即可认为元组v相对于指定时间段Ts1~Ts2可见，将该元组v确定为目标元组。也即是说，各个参与节点设备根据指定时间段和元组上各事务的事务信息，判断出了元组是否为目标元组。

在上述两种情况中，分别从按时间点（指定时刻）读取数据以及按时间段（指定时间段）读取数据的角度，介绍了各个参与节点设备如何根据元组所对应的两个事务的全局提交时刻commit_Ts来判断元组可见性，也即是从多个元组中确定出目标元组。因此，相当于提供了一种分布式数据库系统下的全时态可见性判断算法，该全时态可见性判断算法可以作用于分布式数据库系统的任一个节点设备中，并且保证该节点设备读取到的元组满足全局的事务一致性，能够避免出现DRCC异常。

进一步地，上述全时态可见性判断算法可以用下述伪代码来进行描述：

Algorithm 1：v.visible（Ts，Ts1，Ts2，T）

Input ：Ts：Timestamp

Input ：Ts1，Ts2：Time interval

Input ：T：Query type

Output：visibility

if T = AS_OF **then**

　　if（v.trx_min.commit_ts＜Ts ∧ v.trx_max.commit_ts＞Ts）**then**

　　return visible;

if T = BETWEEN_AND **then**

　　if（v.trx_min.commit_ts＜Ts2 ∧ v.trx_max.commit_ts＞Ts1）**then**

　　return visible;

return unvisible;

需要说明的是，在本公开实施例中，各个时间点的取值范围如下：Ts≤当前时刻，Ts2≤当前时刻，Ts1＜Ts2。

在上述步骤705-706中，在按照指定时刻读取时，按照指定时刻的取值不同，可以将全时态数据的一致性读操作划分为下述两种策略：

当前一致读策略：跨节点的全局读事务指定该全局读事务的事务开始时刻作为要求的一致性点，也即是Ts = start_time，从而可以基于上述步骤705提供的事务状态提取算法和上述步骤706提供的全时态可见性判断算法，进行按照“当前时间点”的数据读取，分别获取到分布式数据库系统中每个节点设备上的可见元组（也即目标元组），保证了全局读取数据过程符合事务一致性。

历史一致读策略：跨节点的全局读事务指定某一个早于该全局读事务的事务开始时刻的时间点作为要求的一致性点，也即是Ts＜start_time，从而可以基于上述步骤705提供的事务状态提取算法和上述步骤706提供的全时态可见性判断算法，进行按照“历史时间点”的数据读取，分别获取到分布式数据库系统中每个节点设备上的历史可见元组（也即目标元组），也即找到了历史上符合一致性点的元组集合，保证了全局读取数据过程符合事务一致性。

707、各个参与节点设备向协调节点设备输出目标元组。

在上述过程中，由于整个分布式数据库系统对各个全局事务的全局事务状态，采用了去中心化的管理机制，因此在执行全局读事务的时候，各个参与节点设备直接在本地执行去中心化的事务状态提取算法和全时态可见性判断算法，能够直接在本地判断并输出目标元组，从而仅需要一轮通信，协调节点设备就能够直接从各个参与节点设备中获取到符合一致性条件的目标元组，无需繁琐地获取集中式的全局事务状态，再通过多轮通信判断出目标元组，能够缩短整个数据库系统的响应延时，节约全局读事务的网络信令开销。

708、当接收到该多个参与节点设备中每个参与节点设备所返回的目标元组时，协调节点设备提交该全局读事务，该目标元组由每个参与节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得。

上述步骤 708 中全局读事务的提交流程，与上述实施例的步骤 403 中全局写事务的提交流程类似，这里不做过多的阐述。

也即是说，当接收到所有参与节点设备返回的目标元组后，协调节点设备向所有参与节点设备发送提交指令，同时向全局事务标识生成集群申请一个事务完成时刻，向目标节点设备发送第八事务状态更新指令，该第八事务状态更新指令中携带全局事务标识、事务完成时刻和当前事务状态（也即是已提交状态），当目标节点设备接收到该第八事务状态更新指令，响应于该第八事务状态更新指令，根据全局事务标识，从事务状态列表中查询与该全局事务标识对应的事务状态元组，在该事务状态元组中记录该第八事务状态更新指令中携带的事务完成时刻，并将全局事务状态从正在准备状态置为已提交状态。

当然，如果某一个参与节点设备发生异常（例如本地没获取到全局事务状态，广播查询请求后也没有任何节点设备返回全局事务状态），那么需要通知协调节点设备重启该全局读事务，因此由协调节点设备向各个参与节点设备发送回滚指令，并执行与上述步骤 403 中类似的 2PC 算法的回滚流程，这里不做赘述。当回滚完毕后，协调节点设备重新执行上述步骤 701-708 所执行的操作。

709、当该全局读事务提交完成时，协调节点设备发送该每个参与节点设备所返回的目标元组。

在上述过程中，当所有参与节点设备均对全局读事务提交完成后，协调节点设备对各个参与节点设备发送的目标元组进行汇总，向用户（数据读取业务的请求方）终端发送汇总后的所有目标元组。

在一些实施例中，在发送目标元组完毕后，协调节点设备还可以向各个参与节点设备发送携带该全局读事务的全局事务标识的删除指令，该删除指令用于指示删除该全局读事务的全局事务状态，当各个参与节点设备接收到该删除指令，可以根据该全局读事务的全局事务标识，在本地的事务状态列表中查询与该全局事务标识对应的事务状态元组，删除查询到的事务状态元组，从而能够省略了对全局读事务的全局事务状态的永久维护，仅仅在全局读事务的执行过程中进行暂时性的维护，节约了各个参与节点设备的本地存储空间。

在另一种情况中，还可以不执行上述步骤 707-709，而是每当各个参与节点设备判断一个目标元组后，直接实时地将该目标元组输出至用户对应的终端，在这种情况下，全局读事务会受到各个参与节点设备的宕机影响。例如，某个参与节点设备在读取了一部分数据后发生宕机，那么该参与节点设备在恢复后，需要向用户对应的终端通知返回的目标元组无效。

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。

本公开实施例提供的方法，通过当接收到数据读取请求时，根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组，由于数据库系统中维护了各个全局事务的全局事务状态，从而获取该多个元组所对应的多个全局事务的全局事务状态，这些获取到的全局事务状态都是对数据读取请求而言符合一致性条件的全局事务状态，从而根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻，也即在不同的全局事务状态下，对全局提交时刻进行不同的赋值，相当于对各个符合一致性条件的全局事务状态进行量化，从而能够基于该多个全局事务的全局提交时刻，对各个元组进行可见性判断，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。正是由于数据库系统中维护了全局事务状态，因此在找到符合一致性条件的全局事务状态后，为了能够方便地进行可见性判断，对不同的全局事务状态赋予不同的全局提交时刻，从而直接基于全局提交来判断元组的可见性，也就避免了发生 DRCC 异常，保证了数据读取过程中所读取的数据处于事务一致的状态，也即是保证了数据库系统中全局读操作的事务一致性。

进一步地，当数据读取请求携带指定时刻时，提供了当全局事务状态处于六种状态下时分别对全局提交时刻的赋值逻辑，通过这样的赋值逻辑，使得全局提交时刻能够精准的用数值刻画出符合一致性条件的全局事务状态，简化了后续可见性判断算法的处理逻辑。

进一步地，当数据读取请求携带指定时间段时，基于该指定时间段的起始时刻和终止时刻，分别执行获取各自对应的全局提交时刻的操作，由于各事务的全局事务状态是一个有向的变化过程，从而通过上述方式可以将获取整个指定时间段内全局提交时刻的复杂操作，转化为获取指定时间段的起始时刻下全局提交时刻以及终止时刻下全局提交时刻的简单操作，简化了获取某一指定时间段内全局提交时刻的复杂度。

进一步地，当数据读取请求携带指定时刻时，根据指定时刻、产生各个元组的事务的全局提交时刻、修改各个元组的事务的全局提交时刻，能够精准地判断出有效时间范围内包括指定时刻的各个目标元组，保障了元组可见性判断的准确度。

进一步地，当数据读取请求携带指定时间段时，根据指定时间段、产生各个元组的事务的全局提交时刻、修改各个元组的事务的全局提交时刻，能够精准地判断出有效时间范围与指定时间段存在交集的各个目标元组，保障了元组可见性判断的准确度。

进一步地，在获取全局事务状态时，首先在本地事务状态列表中进行查询，如果本地查询不到对应的全局事务状态，那么向其他节点设备发送查询请求，只要任一个节点设备返回了全局事务状态，即可进入下一步获取全局提交时刻的流程，这是上述实施例的在去中心化的全局事务状态的管理机制下，在获取全局事务状态的过程中，能够避免频繁地网络信令开销。而无需等待所有节点设备返回全局事务状态，是因为基于实施例的数据同步机制，一旦有任一个节点设备能够返回全局事务状态，即证明该事务已经完成全局提交并在该节点设备上进行了数据同步，也就不必等待后续其他节点设备返回全局事务状态，能够缩短数据读取过程的响应时长。

图 8 是本公开实施例提供的一种事务一致性点的示意图，参见图 8，假设 HTAC 系统中存在三个数据项，分别为 r1、r2、r3（三者可分布不同的节点设备上），下面对从 t0 时刻~t4 时刻所经历的一致性状态进行分析：

初始元组 $r1^0$、$r2^0$、$r3^0$ 处于一个一致性状态，图中用白色圆圈表示初始元组，用实线表示 t0 时刻的一致性状态；

当某一个全局写事务发生时，会致使元组的数据状态发生改变（从当前态转变为过渡态或历史态），如全局写事务 T1 在 t1 时刻全局提交，修改了元组 $r1^0$，生成一个对应的第一元组 $r1^1$，图中用黑色圆圈表示第一元组，用长划线表示 t1 时刻的一致性状态；

后来，全局写事务 T2 在 t2 时刻全局提交，修改了元组 $r2^0$ 和 $r3^0$，产生两个对应的第二元组 $r2^2$ 和 $r3^2$，在图中用斜线圆圈表示第二元组，用短划线表示 t2 时刻的一致性状态；

后来，全局写事务 T3 在 t3 时刻提交，修改了元组 $r1^1$ 和 $r3^2$，产生两个对应的第三元组 $r1^3$ 和 $r3^3$，在图中用网格圆圈表示第三元组，用点划线表示 t3 时刻的一致性状态；

后来，全局写事务 T4 在 t4 时刻提交，修改了元组 $r2^2$，产生对应的第四元组 $r2^4$，在图中用网点圆圈表示第四元组，用点线表示 t4 时刻的一致性状态。

经过 T1~T4 这一系列全局写事务的操作，在全时态数据的维度上进行观察，产生了图中所示的实线、长划线、短划线、点划线、点线共 5 个一致性状态，每一条曲线均可以代表一个一致性状态。那么，如果需要查询 t1.5、t3.5 等历史时刻的历史态元组，那么只需要应用本公开实施例提供的数据读取方法，就能够基于事务状态提取算法和全时态可见性判断算法的约束，找出图中曲线所示的符合一致性状态的历史态元组中，相对于本次全局读事务可见的目标元组。

下面针对 DRCC 异常来进行分析，以论证本公开中数据读取方法能够避免发生上述 DRCC 异常：

结合图 3，以涉及两节点设备的写事务为例，假设全局写事务 T 在节点设备 A 上对元组 x（n）执行写操作，并产生了新的元组 x（n+1），在节点设备 B 上对元组 y（n）执行写操作，并产生了新的元组 y（n+1），下面针对元组 x（n）和 y（n）是否可见进行分类讨论。

首先，当数据读取请求携带指定时刻时，对于元组 x（n）和 y（n）而言，假设产生元组的各个事务（不一定是同一个事务）分别满足可见性判断条件，也即是说产生元组 x（n）的事务的全局提交时刻小于指定时刻，且产生元组 y（n）的事务的全局提交时刻小于指定时刻，那么继续对修改元组 x（n）和 y（n）的事务进行判断，修改元组 x（n）和 y（n）的事务均为同一个全局写事务 T，因此根据判断全局写事务 T 的全局提交时刻的赋值情况，结合全时态可见性判断算法，从而验证输出的元组是否符合读操作的事务一致性。

1、如果指定时刻下，节点设备 A 和节点设备 B 均已经对全局写事务 T 完成提交，全局事务状态为 committed 状态。那么事务完成时刻一定小于指定时刻，全局提交时刻被置为事务完成时刻，那么全局提交时刻也小于指定时刻，结合全时态可见性判断算法，判断出 x（n）和 y（n）对全局读操作一定不可见。因此，节点设备 A 和节点设备 B 均会输出已提交的最新元组，也即是分别输出 x（n+1）和 y（n+1），此时满足全局读操作的事务一致性。

2、如果指定时刻下，节点设备 A 上已经对全局写事务 T 完成提交，但节点设备 B 上还未对全局写事务 T 完成提交（也即是传统算法中会产生 DRCC 异常的情况），全局事务状态为 committing 状态。由于节点设备 A 能提交，证明指定时刻一定是全局准备完成后，并且协调节点设备已经发出了提交指令，由于本公开中对 2PC 算法进行了改进，当全局准备完成后，协调节点设备会通知目标节点设备直接将全局事务状态置为 committed 状态，并且事务状态元组中会已经记录一个事务完成时刻，那么只需比较事务完成时刻与指定时刻之间的大小关

系，能够得到该全局写事务 T 的全局提交时刻，从而结合全时态可见性判断算法，即可得到元组 x（n）和 y（n）是否对数据读取请求可见。

此时，要么该全局提交时刻大于指定时刻，判断出 x（n）和 y（n）均可见，节点设备 A 和节点设备 B 分别输出 x（n）和 y（n）；要么该全局提交时刻小于指定时刻，判断出 x（n）和 y（n）均不可见，节点设备 A 和节点设备 B 分别输出 x（n+1）和 y（n+1），此时满足全局读操作的事务一致性。

3、如果指定时刻下，节点设备 A 和节点设备 B 均已经对全局写事务 T 完成回滚，全局事务状态为 aborted 状态，那么将全局提交时刻置为指定时刻与任一正数相加所得的数值，导致全局提交时刻大于指定时刻，结合全时态可见性判断算法，判断出 x（n）和 y（n）均可见。因此，节点设备 A 和节点设备 B 分别输出 x（n）和 y（n），此时满足全局读操作的事务一致性。

4、如果指定时刻下，节点设备 A 已经对全局写事务 T 完成回滚，但节点设备 B 还未对全局写事务 T 完成回滚，全局事务状态为 aborting 状态，那么将全局提交时刻置为指定时刻与任一正数相加所得的数值，导致全局提交时刻大于指定时刻，结合全时态可见性判断算法，判断出 x（n）和 y（n）均可见。因此，节点设备 A 和节点设备 B 分别输出 x（n）和 y（n），此时满足全局读操作的事务一致性。

5、如果指定时刻下，节点设备 A 和节点设备 B 均已经对全局写事务 T 准备完成，由于本公开提前将全局事务状态置为 committed 状态，因此事务状态元组中也会已经记录一个事务完成时刻，同理比较事务完成时刻与指定时刻之间的大小关系，能够得到该全局写事务 T 的全局提交时刻，从而结合全时态可见性判断算法，即可得到元组 x（n）和 y（n）是否对数据读取请求可见。

此时，要么该全局提交时刻大于指定时刻，判断出 x（n）和 y（n）均可见，节点设备 A 和节点设备 B 分别输出 x（n）和 y（n）；要么该全局提交时刻小于指定时刻，判断出 x（n）和 y（n）均不可见，节点设备 A 和节点设备 B 分别输出 x（n+1）和 y（n+1），此时满足全局读操作的事务一致性。

6、如果指定时刻下，节点设备 A 已经对全局写事务 T 准备完成，但节点设备 B 还未对全局写事务 T 准备完成，全局事务状态为 preparing 状态，将全局提交时刻置为指定时刻与任一正数相加所得的数值，导致全局提交时刻大于指定时刻，结合全时态可见性判断算法，判断出 x（n）和 y（n）均可见。因此，节点设备 A 和节点设备 B 分别输出 x（n）和 y（n），此时满足全局读操作的事务一致性。

7、如果指定时刻下，节点设备 A 和节点设备 B 均尚未对全局写事务 T 开始准备，全局事务状态为 running 状态，那么将全局提交时刻置为指定时刻与任一正数相加所得的数值，导致全局提交时刻大于指定时刻，结合全时态可见性判断算法，判断出 x（n）和 y（n）均可见。因此，一定无法读取到 x（n+1）和 y（n+1），节点设备 A 和节点设备 B 分别输出 x（n）和 y（n），此时满足全局读操作的事务一致性。

在上述过程中，对所有可能发生的情况进行了分类讨论，可以看出本公开的数据读取方法，能够满足所有情况下全局读操作的事务一致性，当然，上面是针对数据读取请求携带指定时刻来分析，然而由于数据读取请求携带指定时间段时，是分别将指定时间段的起始时刻和终止时刻视为“指定时刻”，分别执行全时态可见性判断算法，因此只要在指定时刻下满足事务一致性，在指定时间段也一定能够保证事务一致性，从而能够验证本公开中数据读取方

法的正确的、可靠性和稳定性。

示意性地，在一种应用于HTAC系统的场景中，TP集群通常存储当前态元组和过渡态元组，而AP集群则存储历史态元组，那么在TP集群和AP集群之前可以设置网关服务器作为HTAC系统的协调节点设备，网关服务器与数据读取请求所涉及的各个参与节点设备建立通信连接，分别向TP集群和AP集群内的各个参与节点设备下发数据读取请求。

在TP集群或AP集群中任一参与节点设备均可以应用本公开实施例提供数据读取方法，找到各个参与节点设备之间基于快照隔离机制的共同的一致性点，并向网关服务器输出对读事务可见的目标元组，网关服务器汇总各个目标元组后向发起读事务的提交流程，并向读取业务的请求方终端输出目标元组，以保证同时读取到TP集群和AP集群内全态数据的事务一致性，实现了跨集群、跨节点执行全局读操作在事务级别的一致性。

当然了，还可以在全局读事务开始时，调度器随机分配一个节点设备作为协调节点设备，实现去中心化地全局读操作，构造了轻量级地、去中心化的事务处理机制（或读或写的事务处理流程均有改进），以节约网络信令开销，优化读事务的性能，提升了数据读取的效率，避免了单点性能瓶颈。

在本公开实施例中，所提供的事务状态提取算法和全时态可见性判断算法可以构成去中心化全时态数据一致性读算法，能够保证分布式数据库系统中全时态数据的一致性读性能，扩充了MVCC机制下的可见性判断算法，使得在分布式数据库系统中，对任意时间点均具有符合一致性状态的数据读取能力。

图9是本公开实施例提供的一种数据读取装置的结构示意图，参见图9，该装置包括：

第一确定模块901，用于当接收到数据读取请求时，根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组；

第一获取模块902，用于获取该多个元组所对应的多个全局事务的全局事务状态；

第二获取模块903，用于根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻；

第二确定模块904，用于基于该多个全局事务的全局提交时刻，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。

本公开实施例提供的装置，通过当接收到数据读取请求时，根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组，由于数据库系统中维护了各个全局事务的全局事务状态，从而获取该多个元组所对应的多个全局事务的全局事务状态，这些获取到的全局事务状态都是对数据读取请求而言符合一致性条件的全局事务状态，从而根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻，也即在不同的全局事务状态下，对全局提交时刻进行不同的赋值，相当于对各个符合一致性条件的全局事务状态进行量化，从而能够基于该多个全局事务的全局提交时刻，对各个元组进行可见性判断，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。正是由于数据库系统中维护了全局事务状态，因此在找到符合一致性条件的全局事务状态后，为了能够方便地进行可见性判断，对不同的全局事务状态赋予不同的全局提交时刻，从而直接基于全局提交来判断元组的可见性，也就避免了发生DRCC异常，保证了数据读取过程中所读取的数据处于事务一致的状态，也即是保证了数据库系统中全局读操作的事务一致性。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该第二获取模块 903 用于：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为已提交状态或正在提交状态，当该任一全局事务的事务完成时刻大于该指定时刻时，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值；当该任一全局事务的事务完成时刻不大于该指定时刻时，将该任一全局事务的全局提交时刻置为该事务完成时刻。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该第二获取模块 903 用于：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值。

在一种可能实施方式中，当该数据读取请求中携带指定时间段时，该第二获取模块 903 用于：

基于该指定时间段的起始时刻，执行获取该多个全局事务在该起始时刻下的全局提交时刻的操作；

基于该指定时间段的终止时刻，执行获取该多个全局事务在该终止时刻下的全局提交时刻的操作。

在一种可能实施方式中，该第二确定模块 904 用于：

当该数据读取请求中携带指定时刻时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该第二确定模块 904 用于：

当该数据读取请求中携带指定时间段时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时间段的终止时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时间段的起始时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该第一获取模块 902 用于：

对该多个元组中的任一元组上记录的全局事务标识，从本地的事务状态列表中，查询是否存在与该全局事务标识所对应的全局事务状态；

当存在时，获取与该全局事务标识所对应的全局事务状态；

当不存在时，向该全局事务标识对应的多个节点设备发送查询请求，接收该多个节点设备中任一节点设备返回的全局事务状态。

在一种可能实施方式中，每个全局事务对应于一个事务状态元组，该事务状态元组包括该每个全局事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项。

在一种可能实施方式中，基于图 9 的装置组成，该装置还包括：

当对任一个全局写事务提交完成时，向该任一个全局写事务对应的目标节点设备发送数据同步请求，接收该目标节点设备返回的全局事务状态，该目标节点设备用于存储该任一个全局写事务的全局事务状态；或，

如果当前执行的事务涉及到该任一个全局写事务对应的该目标节点设备，与该目标节点

设备之间进行增量同步；或，

每间隔目标时长，与该目标节点设备之间进行增量同步。

需要说明的是：上述实施例提供的数据读取装置在读取数据时，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，上述实施例提供的数据读取装置与数据读取方法实施例属于同一构思，其具体实现过程详见数据读取方法实施例，这里不再赘述。

图 10 是本公开实施例提供的一种数据读取装置的结构示意图，参见图 10，该装置包括：

发送模块 1001，用于当获取到全局读事务时，向该全局读事务涉及的多个节点设备发送数据读取请求；

提交模块 1002，用于当接收到该多个节点设备中每个节点设备所返回的目标元组时，提交该全局读事务，该目标元组相对于该数据读取请求可见，该目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

该发送模块 1001，还用于当该全局读事务提交完成时，发送该每个节点设备所返回的目标元组。

本公开实施例提供的装置，通过在获取到全局读事务时，向多个节点设备发送数据读取请求，当接收各个节点设备返回的目标元组时，提交全局读事务，由于目标元组是由各个节点设备基于全局提交时刻而确定的，因此各个节点设备返回的目标元组是符合一致性读条件的可见元组，因此当全局读事务提交完成时，发送各个目标元组，从而能够避免 DRCC 异常，保证了数据读取过程中所读取的数据处于事务一致的状态，也即是保证了数据库系统中全局读操作的事务一致性。

在一种可能实施方式中，基于图 10 的装置组成，该装置还包括：

对任一全局写事务，向该任一全局写事务涉及的多个节点设备发送数据准备请求；

当接收到该多个节点设备中每个节点设备所返回的准备成功响应时，向该多个节点设备发送提交指令，通知目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已提交状态；

当接收到该多个节点设备中任一节点设备所返回的准备失败响应时，向该多个节点设备发送回滚指令，通知该目标节点设备将该任一全局写事务的全局事务状态置为正在回滚状态，当接收到该多个节点设备中每个节点设备所返回的回滚完成响应时，通知该目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已回滚状态。

在一种可能实施方式中，基于图 10 的装置组成，该装置还包括：

从该任一全局写事务涉及的多个节点设备中随机选取该目标节点设备；或，

将该任一全局写事务涉及的多个节点设备中第一个被访问的节点设备确定为该目标节点设备。

需要说明的是：上述实施例提供的数据读取装置在读取数据时，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，上述实施例提供的数据读取装置与数据读取方法实施例属于同一构思，其具体实现过

程详见数据读取方法实施例，这里不再赘述。

图 11 是本公开实施例提供的计算机设备的结构示意图，参见图 11，该计算机设备可以为上述各实施例中所涉及的协调节点设备或参与节点设备。该计算机设备 1100 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）1101 和一个或一个以上的存储器 1102，其中，该存储器 1102 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 1101 加载并执行以实现上述各个实施例提供的数据读取方法。当然，该计算机设备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，该计算机设备还可以包括其他用于实现设备功能的部件，在此不做赘述。

在一种可能实施方式中，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下操作：

当接收到数据读取请求时，根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组；

获取该多个元组所对应的多个全局事务的全局事务状态；

根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻；

基于该多个全局事务的全局提交时刻，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为已提交状态或正在提交状态，当该任一全局事务的事务完成时刻大于该指定时刻时，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值；当该任一全局事务的事务完成时刻不大于该指定时刻时，将该任一全局事务的全局提交时刻置为该事务完成时刻。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值。

在一种可能实施方式中，当该数据读取请求中携带指定时间段时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

基于该指定时间段的起始时刻，执行获取该多个全局事务在该起始时刻下的全局提交时刻的操作；

基于该指定时间段的终止时刻，执行获取该多个全局事务在该终止时刻下的全局提交时刻的操作。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

当该数据读取请求中携带指定时刻时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

当该数据读取请求中携带指定时间段时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时间段的终止时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时间段的起始时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中的任一元组上记录的全局事务标识，从本地的事务状态列表中，查询是否存在与该全局事务标识所对应的全局事务状态；

当存在时，获取与该全局事务标识所对应的全局事务状态；

当不存在时，向该全局事务标识对应的多个节点设备发送查询请求，接收该多个节点设备中任一节点设备返回的全局事务状态。

在一种可能实施方式中，每个全局事务对应于一个事务状态元组，该事务状态元组包括该每个全局事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

当任一个全局写事务提交完成时，向该任一个全局写事务对应的目标节点设备发送数据同步请求，接收该目标节点设备返回的全局事务状态，该目标节点设备用于存储该任一个全局写事务的全局事务状态；或，

如果当前执行的事务涉及到该任一个全局写事务对应的该目标节点设备，与该目标节点设备之间进行增量同步；或，

每间隔目标时长，与该目标节点设备之间进行增量同步。

在示例性实施例中，该计算机设备的一个或多个存储器中存储的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下操作：

当获取到全局读事务时，向该全局读事务涉及的多个节点设备发送数据读取请求；

当接收到该多个节点设备中每个节点设备所返回的目标元组时，提交该全局读事务，该目标元组相对于该数据读取请求可见，该目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当该全局读事务提交完成时，发送该每个节点设备所返回的目标元组。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

对任一全局写事务，向该任一全局写事务涉及的多个节点设备发送数据准备请求；

当接收到该多个节点设备中每个节点设备所返回的准备成功响应时，向该多个节点设备发送提交指令，通知目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已提交状态；

当接收到该多个节点设备中任一节点设备所返回的准备失败响应时，向该多个节点设备发送回滚指令，通知该目标节点设备将该任一全局写事务的全局事务状态置为正在回滚状态，当接收到该多个节点设备中每个节点设备所返回的回滚完成响应时，通知该目标节点设备记

录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已回滚状态。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

从该任一全局写事务涉及的多个节点设备中随机选取该目标节点设备；或，

将该任一全局写事务涉及的多个节点设备中第一个被访问的节点设备确定为该目标节点设备。

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计算机设备中的一个或多个处理器执行以完成上述实施例中的数据读取方法。例如，该计算机可读存储介质可以是ROM、随机存取存储器（RAM）、CD-ROM、磁带、软盘和光数据存储设备等。

在一种可能实施方式中，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下操作：

当接收到数据读取请求时，根据该数据读取请求携带的读取条件，确定符合该读取条件的多个元组；

获取该多个元组所对应的多个全局事务的全局事务状态；

根据该多个全局事务的全局事务状态，获取该多个全局事务的全局提交时刻；

基于该多个全局事务的全局提交时刻，从该多个元组中确定目标元组，该目标元组相对于该数据读取请求可见。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为已提交状态或正在提交状态，当该任一全局事务的事务完成时刻大于该指定时刻时，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值；当该任一全局事务的事务完成时刻不大于该指定时刻时，将该任一全局事务的全局提交时刻置为该事务完成时刻。

在一种可能实施方式中，当该数据读取请求中携带指定时刻时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中任一元组所对应的任一全局事务，如果该任一全局事务在该指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将该任一全局事务的全局提交时刻置为该指定时刻与任一正数相加所得到的数值。

在一种可能实施方式中，当该数据读取请求中携带指定时间段时，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

基于该指定时间段的起始时刻，执行获取该多个全局事务在该起始时刻下的全局提交时刻的操作；

基于该指定时间段的终止时刻，执行获取该多个全局事务在该终止时刻下的全局提交时刻的操作。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

当该数据读取请求中携带指定时刻时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

当该数据读取请求中携带指定时间段时，对该多个元组中任一元组，如果产生该任一元组的全局事务的全局提交时刻小于该指定时间段的终止时刻，且修改该任一元组的全局事务的全局提交时刻大于该指定时间段的起始时刻，将该任一元组确定为一个目标元组。

在一种可能实施方式中，该至少一条程序代码由该一个或多个处理器加载并执行如下操作：

对该多个元组中的任一元组上记录的全局事务标识，从本地的事务状态列表中，查询是否存在与该全局事务标识所对应的全局事务状态；

当存在时，获取与该全局事务标识所对应的全局事务状态；

当不存在时，向该全局事务标识对应的多个节点设备发送查询请求，接收该多个节点设备中任一节点设备返回的全局事务状态。

在一种可能实施方式中，每个全局事务对应于一个事务状态元组，该事务状态元组包括该每个全局事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

当任一个全局写事务提交完成时，向该任一个全局写事务对应的目标节点设备发送数据同步请求，接收该目标节点设备返回的全局事务状态，该目标节点设备用于存储该任一个全局写事务的全局事务状态；或，

如果当前执行的事务涉及到该任一个全局写事务对应的该目标节点设备，与该目标节点设备之间进行增量同步；或，

每间隔目标时长，与该目标节点设备之间进行增量同步。

在示例性实施例中，该计算机可读存储介质中存储的至少一条程序代码由计算机设备的一个或多个处理器加载并执行以实现如下操作：

当获取到全局读事务时，向该全局读事务涉及的多个节点设备发送数据读取请求；

当接收到该多个节点设备中每个节点设备所返回的目标元组时，提交该全局读事务，该目标元组相对于该数据读取请求可见，该目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当该全局读事务提交完成时，发送该每个节点设备所返回的目标元组。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

对任一全局写事务，向该任一全局写事务涉及的多个节点设备发送数据准备请求；

当接收到该多个节点设备中每个节点设备所返回的准备成功响应时，向该多个节点设备发送提交指令，通知目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已提交状态；

当接收到该多个节点设备中任一节点设备所返回的准备失败响应时，向该多个节点设备发送回滚指令，通知该目标节点设备将该任一全局写事务的全局事务状态置为正在回滚状态，当接收到该多个节点设备中每个节点设备所返回的回滚完成响应时，通知该目标节点设备记录该任一全局写事务的事务完成时刻，并通知该目标节点设备将该任一全局写事务的全局事务状态置为已回滚状态。

在一种可能实施方式中，该至少一条程序代码还由该一个或多个处理器加载并执行如下操作：

从该任一全局写事务涉及的多个节点设备中随机选取该目标节点设备；或，

将该任一全局写事务涉及的多个节点设备中第一个被访问的节点设备确定为该目标节点设备。

在一些实施例中，还提供一种包括至少一条程序代码的计算机程序或计算机程序产品，当其在计算机设备上运行时，使得计算机设备执行前述各个实施例所提供的数据读取方法中任一种可能实现方式，在此不作赘述。

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。

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

权 利 要 求 书

1、一种数据读取方法，其特征在于，所述方法包括：

当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，确定符合所述读取条件的多个元组；

获取所述多个元组所对应的多个全局事务的全局事务状态；

根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

2、根据权利要求 1 所述的方法，其特征在于，当所述数据读取请求中携带指定时刻时，所述根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻包括：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为已提交状态或正在提交状态，当所述任一全局事务的事务完成时刻大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值；当所述任一全局事务的事务完成时刻不大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述事务完成时刻。

3、根据权利要求 1 所述的方法，其特征在于，当所述数据读取请求中携带指定时刻时，所述根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻包括：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值。

4、根据权利要求 1 所述的方法，其特征在于，当所述数据读取请求中携带指定时间段时，所述根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻包括：

基于所述指定时间段的起始时刻，执行获取所述多个全局事务在所述起始时刻下的全局提交时刻的操作；

基于所述指定时间段的终止时刻，执行获取所述多个全局事务在所述终止时刻下的全局提交时刻的操作。

5、根据权利要求 1 所述的方法，其特征在于，所述基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组包括：

当所述数据读取请求中携带指定时刻时，对所述多个元组中任一元组，如果产生所述任一元组的全局事务的全局提交时刻小于所述指定时刻，且修改所述任一元组的全局事务的全局提交时刻大于所述指定时刻，将所述任一元组确定为一个目标元组。

6、根据权利要求 1 所述的方法，其特征在于，所述基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组包括：

当所述数据读取请求中携带指定时间段时，对所述多个元组中任一元组，如果产生所述任一元组的全局事务的全局提交时刻小于所述指定时间段的终止时刻，且修改所述任一元组的全局事务的全局提交时刻大于所述指定时间段的起始时刻，将所述任一元组确定为一个目标元组。

7、根据权利要求 1 所述的方法，其特征在于，所述获取所述多个元组所对应的多个全局事务的全局事务状态包括：

对所述多个元组中的任一元组上记录的全局事务标识，从本地的事务状态列表中，查询是否存在与所述全局事务标识所对应的全局事务状态；

当存在时，获取与所述全局事务标识所对应的全局事务状态；

当不存在时，向所述全局事务标识对应的多个节点设备发送查询请求，接收所述多个节点设备中任一节点设备返回的全局事务状态。

8、根据权利要求 1 所述的方法，其特征在于，每个全局事务对应于一个事务状态元组，所述事务状态元组包括所述每个全局事务的全局事务标识、事务开始时刻、事务完成时刻、全局事务状态或者节点设备标识中的至少一项。

9、根据权利要求 1 所述的方法，其特征在于，所述方法还包括：

当任一个全局写事务提交完成时，向所述任一个全局写事务对应的目标节点设备发送数据同步请求，接收所述目标节点设备返回的全局事务状态，所述目标节点设备用于存储所述任一个全局写事务的全局事务状态；或，

如果当前执行的事务涉及到所述任一个全局写事务对应的所述目标节点设备，与所述目标节点设备之间进行增量同步；或，

每间隔目标时长，与所述目标节点设备之间进行增量同步。

10、一种数据读取方法，其特征在于，所述方法包括：

当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

11、根据权利要求 10 所述的方法，其特征在于，所述方法还包括：

对任一全局写事务，向所述任一全局写事务涉及的多个节点设备发送数据准备请求；

当接收到所述多个节点设备中每个节点设备所返回的准备成功响应时，向所述多个节点设备发送提交指令，通知目标节点设备记录所述任一全局写事务的事务完成时刻，并通知所述目标节点设备将所述任一全局写事务的全局事务状态置为已提交状态；

当接收到所述多个节点设备中任一节点设备所返回的准备失败响应时，向所述多个节点设备发送回滚指令，通知所述目标节点设备将所述任一全局写事务的全局事务状态置为正在

回滚状态，当接收到所述多个节点设备中每个节点设备所返回的回滚完成响应时，通知所述目标节点设备记录所述任一全局写事务的事务完成时刻，并通知所述目标节点设备将所述任一全局写事务的全局事务状态置为已回滚状态。

12、根据权利要求 11 所述的方法，其特征在于，所述方法还包括：

从所述任一全局写事务涉及的多个节点设备中随机选取所述目标节点设备；或，

将所述任一全局写事务涉及的多个节点设备中第一个被访问的节点设备确定为所述目标节点设备。

13、一种数据读取装置，其特征在于，所述装置包括：

第一确定模块，用于当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，确定符合所述读取条件的多个元组；

第一获取模块，用于获取所述多个元组所对应的多个全局事务的全局事务状态；

第二获取模块，用于根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

第二确定模块，用于基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

14、一种数据读取装置，其特征在于，所述装置包括：

发送模块，用于当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

提交模块，用于当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

所述发送模块，还用于当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

15、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如下操作：

当接收到数据读取请求时，根据所述数据读取请求携带的读取条件，确定符合所述读取条件的多个元组；

获取所述多个元组所对应的多个全局事务的全局事务状态；

根据所述多个全局事务的全局事务状态，获取所述多个全局事务的全局提交时刻；

基于所述多个全局事务的全局提交时刻，从所述多个元组中确定目标元组，所述目标元组相对于所述数据读取请求可见。

16、根据权利要求 15 所述的计算机设备，其特征在于，当所述数据读取请求中携带指定时刻时，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如下操作：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为已提交状态或正在提交状态，当所述任一全局事务的事务完成时刻大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值；当所述任一全局事务的事务完成时刻不大于所述指定时刻时，将所述任一全局事务的全局提交时刻置为所述事务完成时刻。

17、根据权利要求 15 所述的计算机设备，其特征在于，当所述数据读取请求中携带指定时刻时，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如下操作：

对所述多个元组中任一元组所对应的任一全局事务，如果所述任一全局事务在所述指定时刻的全局事务状态为正在执行状态、正在准备状态、已回滚状态或者正在回滚状态中任一状态，将所述任一全局事务的全局提交时刻置为所述指定时刻与任一正数相加所得到的数值。

18、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如下操作：

当获取到全局读事务时，向所述全局读事务涉及的多个节点设备发送数据读取请求；

当接收到所述多个节点设备中每个节点设备所返回的目标元组时，提交所述全局读事务，所述目标元组相对于所述数据读取请求可见，所述目标元组由每个节点设备基于多个全局事务的全局提交时刻，从多个元组中确定所得；

当所述全局读事务提交完成时，发送所述每个节点设备所返回的目标元组。

19、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 9 中任一项所述的数据读取方法。

20、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 10 至权利要求 12 中任一项所述的数据读取方法。

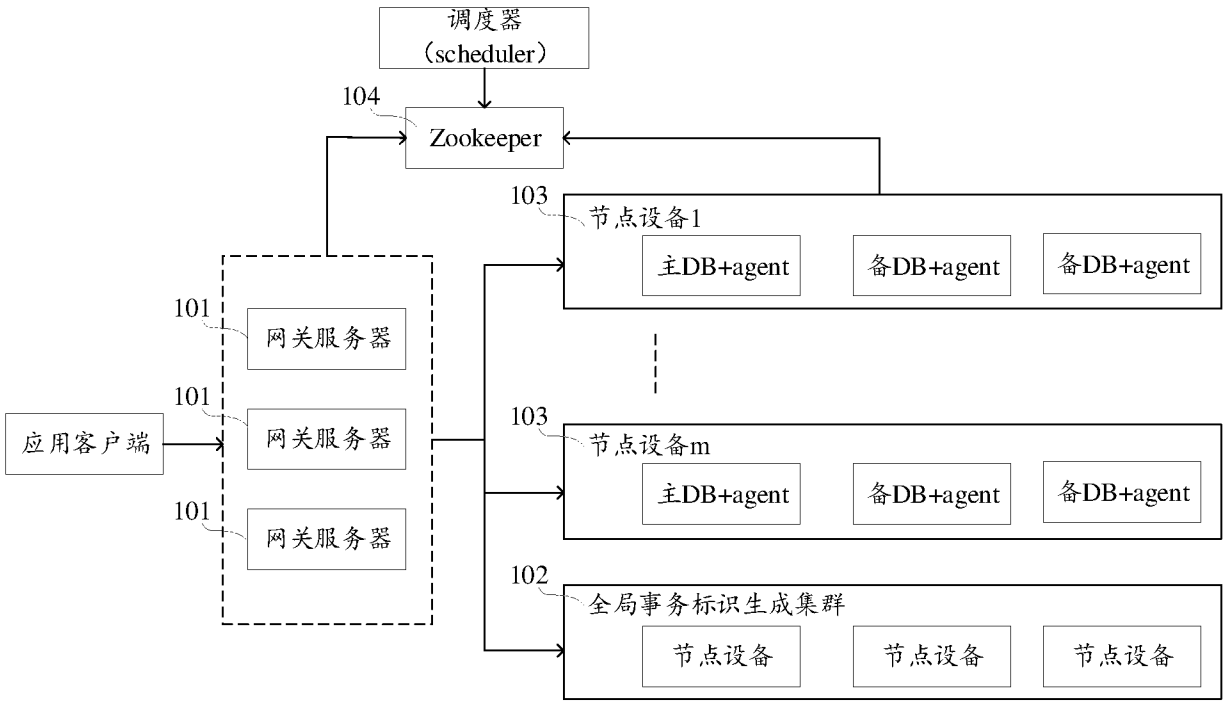
调度器
（scheduler）
104
Zookeeper
103
节点设备1
主DB+agent
备DB+agent
备DB+agent
101
网关服务器
101
网关服务器
应用客户端
101
网关服务器
103
节点设备m
主DB+agent
备DB+agent
备DB+agent
102
全局事务标识生成集群
节点设备
节点设备
节点设备

图 1

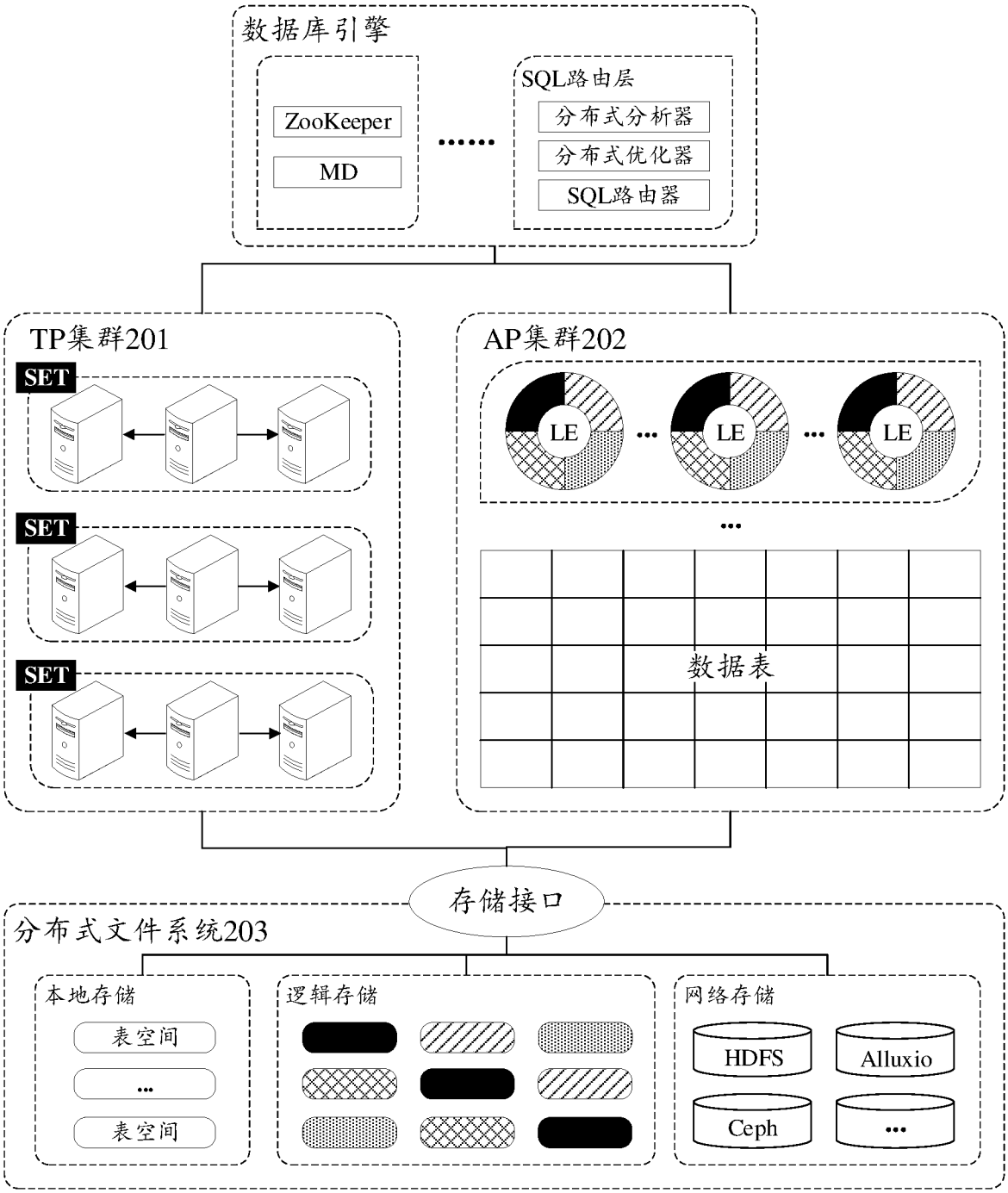
数据库引擎
ZooKeeper
MD
.....
SQL路由层
分布式分析器
分布式优化器
SQL路由器
TP集群201
SET
SET
SET
AP集群202
LE
LE
LE
数据表
存储接口
分布式文件系统203
本地存储
表空间
...
表空间
逻辑存储
网络存储
HDFS
Alluxio
Ceph
...

图 2

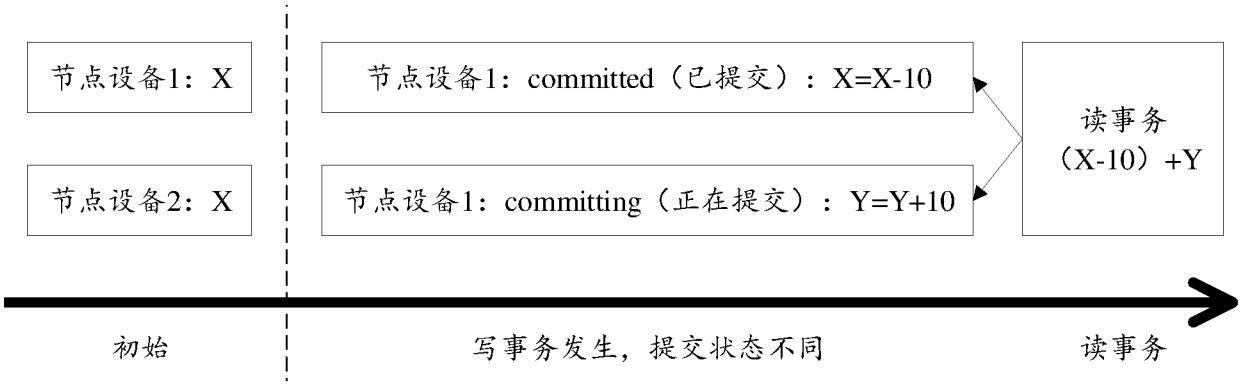
节点设备1：X
节点设备2：X
节点设备1：committed（已提交）：X=X-10
节点设备1：committing（正在提交）：Y=Y+10
读事务
（X-10）+Y
初始
写事务发生，提交状态不同
读事务

图 3

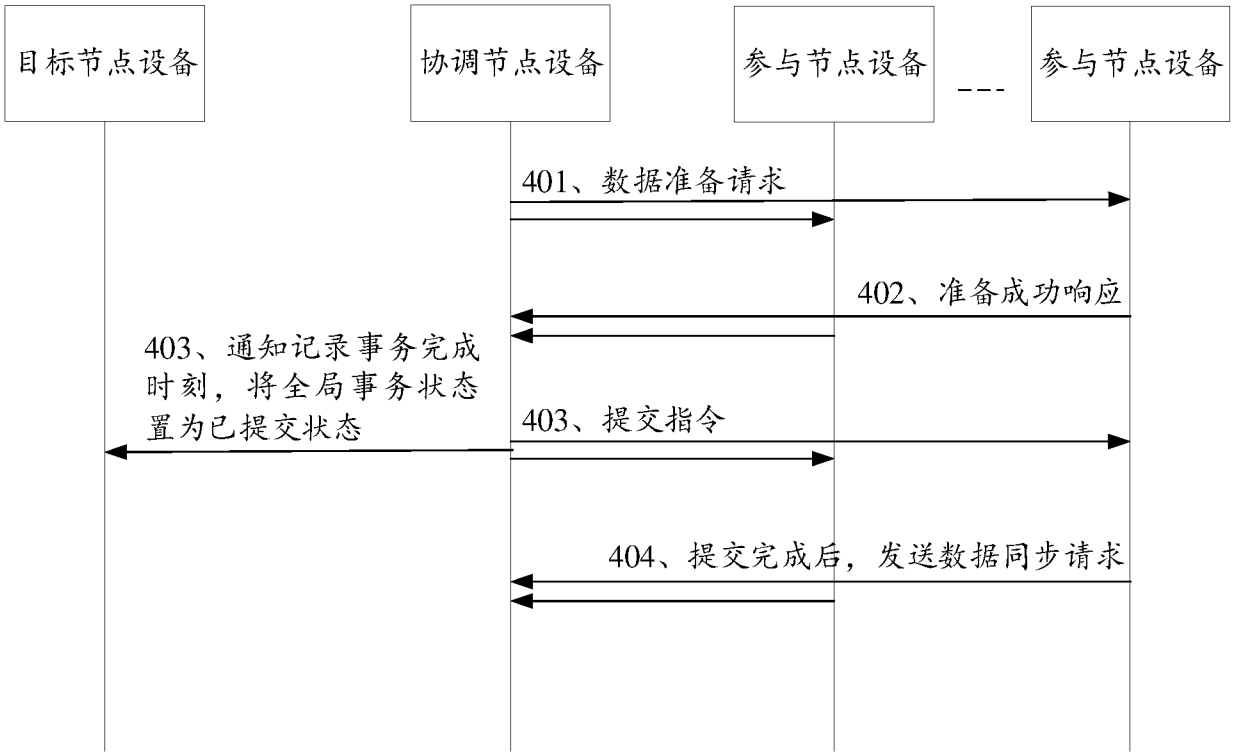
目标节点设备
协调节点设备
参与节点设备
参与节点设备
401、数据准备请求
402、准备成功响应
403、通知记录事务完成时刻，将全局事务状态置为已提交状态
403、提交指令
404、提交完成后，发送数据同步请求

图 4


记录头
主键
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 5

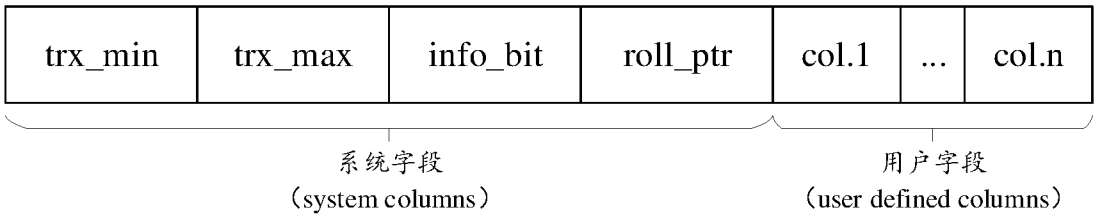
trx_min
trx_max
info_bit
roll_ptr
col.1
...
col.n
系统字段
（system columns）
用户字段
（user defined columns）

图 6

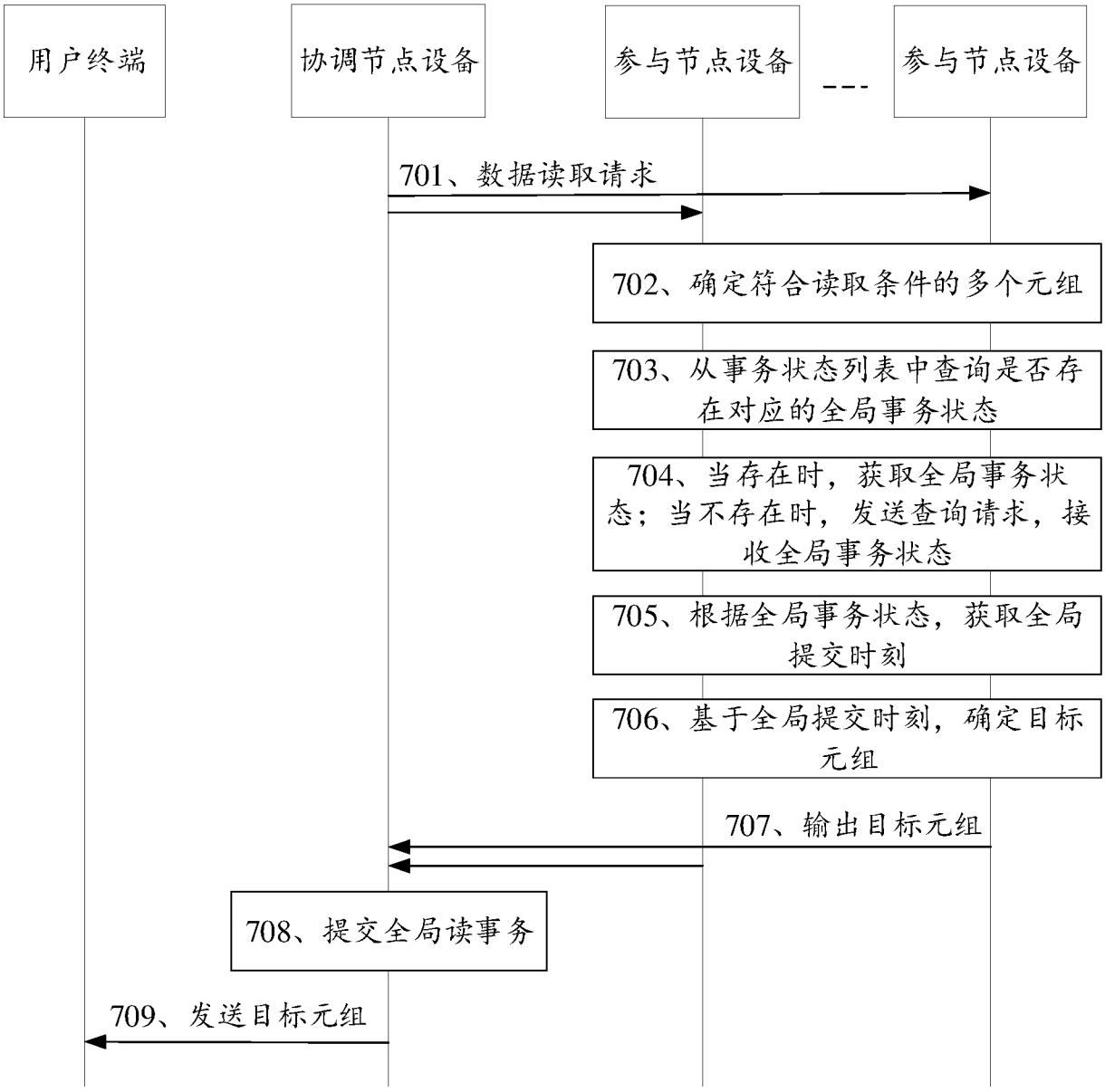
用户终端
协调节点设备
参与节点设备
参与节点设备
701、数据读取请求
702、确定符合读取条件的多个元组
703、从事务状态列表中查询是否存在对应的全局事务状态
704、当存在时，获取全局事务状态；当不存在时，发送查询请求，接收全局事务状态
705、根据全局事务状态，获取全局提交时刻
706、基于全局提交时刻，确定目标元组
707、输出目标元组
708、提交全局读事务
709、发送目标元组

图 7

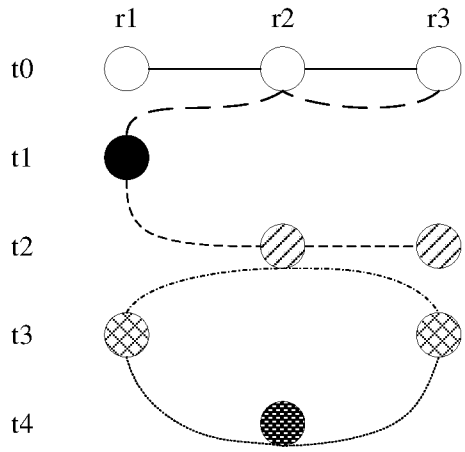
r1
r2
r3
t0
t1
t2
t3
t4

图 8

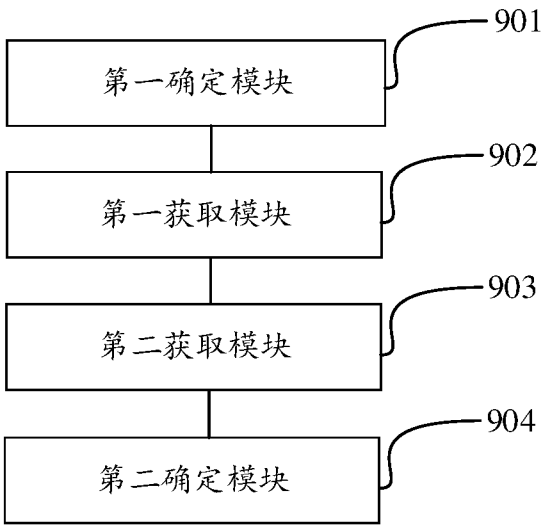
901
第一确定模块
902
第一获取模块
903
第二获取模块
904
第二确定模块

图 9

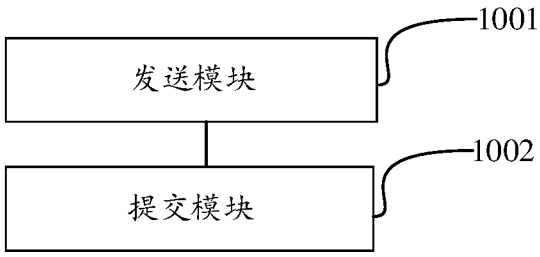
1001
发送模块
1002
提交模块

图 10

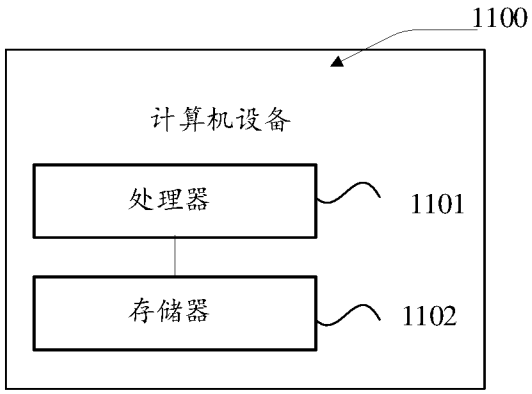

图 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/108100

A. CLASSIFICATION OF SUBJECT MATTER

G06F 16/2458(2019.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, WPI, EPODOC, CNKI, ISI: 分布, 节点, 数据, 元, 可见, 可读, 可获取, 完成, 提交, 结束, 时间, 时刻, 全局, 事务, 读半已提交异常, distribution, node, data, meta, visible, readable, available, submit, complete, time, global, transaction

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 111190935 A (RENMIN UNIVERSITY OF CHINA; et al.) 22 May 2020 (2020-05-22) claims 1-15, description paragraphs [0332]-[0347]	1-20
X	CN 109977171 A (RENMIN UNIVERSITY OF CHINA) 05 July 2019 (2019-07-05) description, paragraphs [0139]-[0239] and [0322]-[0349]	1, 7, 8, 10, 13-15, 18-20
A	CN 109739935 A (TENCENT TECHNOLOGY SHENZHEN CO., LTD.) 10 May 2019 (2019-05-10) entire document	1-20
A	CN 106598992 A (ZTE CORPORATION) 26 April 2017 (2017-04-26) entire document	1-20
A	US 2016253375 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 01 September 2016 (2016-09-01) entire document	1-20

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 October 2020	**30 October 2020**

Name and mailing address of the ISA/CN	Authorized officer
China National Intellectual Property Administration (ISA/CN) **No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088** **China**	
Facsimile No. **(86-10)62019451**	Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/108100

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 111190935 A	22 May 2020	None	
CN 109977171 A	05 July 2019	None	
CN 109739935 A	10 May 2019	None	
CN 106598992 A	26 April 2017	WO 2017063520 A1	20 April 2017
US 2016253375 A1	01 September 2016	JP 2013033345 A	14 February 2013
		US 9348640 B2	24 May 2016
		US 9646045 B2	09 May 2017
		US 2013036136 A1	07 February 2013

国际检索报告

国际申请号
PCT/CN2020/108100

A. 主题的分类

G06F 16/2458(2019.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CNPAT, WPI, EPODOC, CNKI, ISI: 分布，节点，数据，元，可见，可读，可获取，完成，提交，结束，时间，时刻，全局，事务，读半已提交异常，distribution, node, data, meta, visible，readable, available, submit, co-mplete，time，global, transaction

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
PX	CN 111190935 A (中国人民大学 等) 2020年 5月 22日 (2020-05-22) 权利要求1-15，说明书第[0332]-[0347]段	1-20
X	CN 109977171 A (中国人民大学) 2019年 7月 5日 (2019-07-05) 说明书第[0139]-[0239]，[0322]-[0349]段	1、7、8、10、13-15、18-20
A	CN 109739935 A (腾讯科技深圳有限公司) 2019年 5月 10日 (2019-05-10) 全文	1-20
A	CN 106598992 A (中兴通讯股份有限公司) 2017年 4月 26日 (2017-04-26) 全文	1-20
A	US 2016253375 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 2016年 9月 1日 (2016-09-01) 全文	1-20

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2020年 10月 16日	2020年 10月 30日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 李娜 电话号码 86-(10)-53961403

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2020/108100

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	111190935	A	2020年 5月 22日	无			
CN	109977171	A	2019年 7月 5日	无			
CN	109739935	A	2019年 5月 10日	无			
CN	106598992	A	2017年 4月 26日	WO	2017063520	A1	2017年 4月 20日
US	2016253375	A1	2016年 9月 1日	JP	2013033345	A	2013年 2月 14日
				US	9348640	B2	2016年 5月 24日
				US	9646045	B2	2017年 5月 9日
				US	2013036136	A1	2013年 2月 7日