

验证结构更新方法、数据校验方法、设备、介质及产品

技术领域

[01]本说明书一个或多个实施例涉及数据库技术领域，尤其涉及一种验证结构更新方法、基于验证结构的数据校验方法、电子设备、计算机可读存储介质及计算机程序产品。

背景技术

[02]在相关技术中，数据库服务商可以基于其持有的数据库为不同用户提供各种数据库服务，如数据查询服务、数据更新服务等。为了可以让客户端有效验证数据库服务商的服务端返回的数据的正确性，服务端在响应客户端的数据查询请求时，会向客户端返回查询结果与查询结果证明，客户端根据查询结果证明以及提前获取的校验凭证，可以校验查询结果的正确性。

[03]其中，校验凭证可以基于服务端维护的 ADS（authenticated data structure，可验证数据结构）结构获得。服务端在基于客户端发送的数据更新请求对其管理的数据库中的数据更新时，也需要相应更新 ADS 结构；服务端基于更新后的 ADS 结构将更新后的校验凭证同步给所有客户端，客户端才能基于更新后的校验凭证对查询到的最新更新的数据进行校验。

[04]但是，相关技术中更新 ADS 结构所需消耗的资源较大，为了保证数据库中的数据更新性能和数据查询性能不受影响，通常采用旁路更新 ADS 结构的方式。但这种方式将导致没有办法确保校验凭证的实时性，也就没有办法对查询到的最新更新的数据做可信校验。

发明内容

[05]有鉴于此，本说明书一个或多个实施例提供一种验证结构更新方法、基于验证结构的数据校验方法、电子设备、计算机可读存储介质及计算机程序产品。

[06]为实现上述目的，本说明书一个或多个实施例提供技术方案如下。

[07]根据本说明书一个或多个实施例的第一方面，提出了一种验证结构更新方法，所述验证结构用于对数据库中的数据表进行数据校验；所述验证结构包括树和日志；所述树中的各个叶子节点用于记录所述数据表中的各个元组的哈希值，以及所述树中的非叶子

节点记录的哈希值由其孩子节点记录的哈希值聚合得到；其中，所述树的根节点记录的根哈希值和所述日志作为校验凭证被同步至对所述数据表有查询需求的第二客户端中；所述方法包括：

响应于第一客户端对所述数据表中的元组的更新指令，获取更新后的元组的哈希值，将更新后的元组的哈希值记录在所述日志中，并将所述日志同步至对所述数据表有查询需求的第二客户端中；

响应于所述日志满足更新条件，利用所述日志中记录的各个元组的哈希值更新所述树中与各个所述元组对应的叶子节点，并基于更新后的叶子节点记录的哈希值更新所述树中的非叶子节点，得到更新后的树以及将记录的哈希值更新到树中的更新后的日志，将所述更新后的树的根节点记录的根哈希值和所述更新后的日志同步至对所述数据表有查询需求的第二客户端中。

[08]根据本说明书一个或多个实施例的第二方面，提出了一种基于验证结构的数据校验方法，所述验证结构基于第一方面所述的验证结构更新方法得到；所述方法包括：

响应于第二客户端对数据库中的数据表的查询指令，查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值；

若是，将所述日志记录的所述目标元组的哈希值返回给所述第二客户端，以使得所述第二客户端基于所述目标元组的哈希值和存储在其本地的所述日志对所述目标元组进行数据校验；

若否，查询与所述数据表对应的树，确定所述目标元组对应的叶子节点在所述树中的证明路径并返回给所述第二客户端，以使得所述第二客户端基于所述证明路径和存储在本地根哈希值进行数据校验。

[09]根据本说明书实施例的第三方面，提供一种基于验证结构的数据校验方法，应用于第二客户端，所述第二客户端持有的校验凭证基于第一方面所述的验证结构更新方法得到，所述校验凭证包括与数据库中的数据表对应的根哈希值和日志，所述日志记录有所述数据表中的至少一个元组的哈希值；

所述方法包括：

向所述数据库发送针对所述数据表的查询指令；

若所述数据库返回所述查询指令指示的目标元组的哈希值，基于所述目标元组的哈

希值和存储在其本地的所述日志对所述目标元组进行数据校验；

若所述数据库返回所述查询指令指示的目标元组的证明路径，基于所述证明路径和存储在本地的所述根哈希值进行数据校验。

[10]根据本说明书实施例的第四方面，提供一种电子设备，包括：

处理器；

用于存储处理器可执行指令的存储器；

其中，所述处理器执行所述可执行指令时，用于实现第一方面所述的方法。

[11]根据本说明书实施例的第五方面，提供一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现上述任一项所述方法的步骤。

[12]根据本说明书实施例的第六方面，提供一种计算机程序产品，包括计算机程序，该计算机程序被处理器执行时实现上述任一项所述方法的步骤。

[13]本说明书的实施例提供的技术方案可以包括以下有益效果。

[14]本说明书实施例中，首先提供了一种验证结构，验证结构用于对数据库中的数据表进行数据校验；验证结构包括树和日志；树中的各个叶子节点用于记录数据表中的各个元组的哈希值，以及树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到；其中，树的根节点记录的根哈希值和日志作为校验凭证被同步至对数据表有查询需求的第二客户端中。

[15]本实施例提供的验证结构更新方法，通过将根哈希值和日志作为校验凭证，进而在验证结构更新过程中，在接收到更新指令后，可以即时更新验证结构中的日志，并同步给对数据表有查询需求的客户端，从而保证了数据校验的实时性；并且只有在满足更新条件时才会将日志中记录的所有哈希值一次性更新到树中，避免了实时更新树带来的资源损耗问题，在数据查询场景下能够提供更好的性能和响应速度，同时保持数据更新的高效性。

[16]本实施例提供的数据校验方法，在进行数据查询时可以先访问日志再访问树，如果查询指令指示的目标元组的哈希值记录在日志中可以直接返回给第二客户端，有利于提高查询效率；对于未记录在日志中的目标元组，可以利用验证结构中的树获得证明路径并返回给第二客户端进行数据校验；并且由于上述的验证结构的即时更新，使得数据查询场景下可以对最新更新的数据进行数据验证，在数据查询场景下能够提供更好的性能

和响应速度。

[17]应当理解的是，以上的一般描述和后文的细节描述仅是示例性和解释性的，并不能限制本公开。

附图说明

[18]图 1 是一示例性实施例提供的一种场景示意图。

[19]图 2 是一示例性实施例提供的一种验证结构的更新方法的流程图。

[20]图 3 是一示例性实施例提供的一种基于验证结构的数据校验方法的流程图。

[21]图 4 是一示例性实施例提供的一种验证结构更新前后的示意图。

[22]图 5 是一示例性实施例提供的一种基于 MVCC 的数据表的示意图。

[23]图 6 是一示例性实施例提供的一种默克尔 B 树的示意图。

[24]图 7 是一示例性实施例提供的另一种基于验证结构的数据校验方法的流程图。

[25]图 8 是一示例性实施例提供的一种电子设备的结构示意图。

具体实施方式

[26]这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施例中所描述的实施方式并不代表与本说明书一个或多个实施例相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本说明书一个或多个实施例的一些方面相一致的装置和方法的例子。

[27]需要说明的是：在其他实施例中并不一定按照本说明书示出和描述的顺序来执行相应方法的步骤。在一些其他实施例中，其方法所包括的步骤可以比本说明书所描述的更多或更少。此外，本说明书中所描述的单个步骤，在其他实施例中可能被分解为多个步骤进行描述；而本说明书中所描述的多个步骤，在其他实施例中也可能被合并为单个步骤进行描述。

[28]在相关技术中，数据库服务商可以基于其持有的数据库为不同用户提供各种数据库服务，如数据查询服务、数据更新服务等。为了可以让客户端有效验证数据库服务商的服务端返回的数据的正确性，服务端在响应客户端的数据查询请求时，会向客户端返回

查询结果与查询结果证明，客户端根据查询结果证明以及提前获取的校验凭证，可以校验查询结果的正确性。

[29]在一个例子中，数据所有者可以将数据库管理委托给外包数据库服务商，从而外包数据库服务商可以为不同用户提供各种数据库服务。又或者，数据库服务商可以基于自身建立的数据库为不同用户提供各种数据库服务。本实施例对此不做任何限制。

[30]其中，校验凭证可以基于服务端维护的 ADS (authenticated data structure, 可验证数据结构) 结构获得。服务端在基于客户端发送的数据更新请求对其管理的数据库中的数据更新时，也需要相应更新 ADS 结构；然后服务端基于更新后的 ADS 结构将更新后的校验凭证同步给所有客户端，客户端才能基于更新后的校验凭证对查询到的最新更新的数据进行校验。

[31]以默克尔树 (MerkleTree)、默克尔 B 树 (MerkleBTree) 等作为 ADS 结构进行举例说明。

[32]默克尔树用来记录数据库表当前数据状态。一般意义上来讲，它是哈希大量聚集数据单元的一种方式。每个叶子节点对应一个数据单元 (也即数据表中的一个元组) 的哈希，非叶子节点由其孩子节点哈希聚合，计算出该节点的哈希值。重复同样的过程，直至计算出根节点的哈希值。

[33]BTree(B 树)是数据库表索引最常用的索引结构，是一种自平衡且能够保持数据有序的树。这种数据结构能够让查找数据、顺序访问、插入数据及删除的动作，都在对数时间内完成。

[34]MBT(MerkleBTree, 默克尔 B 树)是 MerkleTree 结合 BTree 的数据结构，既可记录表示当前数据状态，又可作为查询索引，替换传统的 BTree 索引。

[35]默克尔树或默克尔 B 树中的各个叶子节点用于记录数据表中的各个元组的哈希值，以及树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到，重复同样的过程，直至计算出根节点的根哈希值。各个客户端获取到的校验凭证为根节点的根哈希值 RootHash。

[36]服务端在基于客户端发送的数据更新请求对其管理的数据库中的数据更新的情况下，比如更新了数据库表中的某个元组，需要重新计算该元组的哈希值，利用该元组的哈希值更新默克尔树中的与该元组对应的叶子节点，进而更新该叶子节点的父节点的哈希值，重复相同的过程，直到更新根节点的根哈希值，并将更新后的根节点的根哈希

值 RootHash 作为最新的校验凭证同步给各个对该数据表有查询需求的客户端。

[37]服务端在响应客户端的数据查询请求时,会向客户端返回查询结果与查询结果证明,比如查询结果为某个元组,查询结果证明为该元组在默克尔树中对应的叶子节点到根节点的证明路径 ProofPath。客户端在接收到查询结果和证明路径 ProofPath 之后,根据查询结果计算哈希值,再结合证明路径 ProofPath 进行哈希聚合反推得到最终的计算结果,若最终的计算结果与客户端持有的根节点的根哈希值 RootHash 相同,则说明数据可信未被修改;反之,则说明数据已被修改。

[38]由于数据库查询场景对于更新查询的时延要求较高,实时更新 ADS 结构(如上述的默克尔树、默克尔 B 树)所需消耗的资源较大,为了保证数据库中的数据更新性能和数据查询性能不受影响,通常采用旁路更新 ADS 结构的方式。但这种方式导致的问题是:没有办法确保基于 ADS 结构确定的校验凭证的实时性,也就没有办法对查询到的最新更新的数据做可信校验。显然,对于一些敏感重要数据场景(例如:金融资产数据,平台用户权限数据),无法做到实时校验是不满足要求的。

[39]基于此,本说明书实施例提供了一种验证结构更新方法以及一种基于验证结构的数据校验方法,旨在解决数据库查询场景中实时校验数据准确性的问题。

[40]首先提供了一种验证结构,验证结构用于对数据库中的数据表进行数据校验;验证结构包括树和日志;树中的各个叶子节点用于记录数据表中的各个元组的哈希值,以及树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到;其中,树的根节点记录的根哈希值和日志作为校验凭证被同步至对数据表有查询需求的第二客户端中。

[41]通过将根哈希值和日志作为校验凭证,进而在验证结构更新过程中,在接收到更新指令后,可以即时更新验证结构中的日志,并同步给对数据表有查询需求的客户端,从而保证了数据校验的实时性;并且只有在满足更新条件时才会将日志中记录的所有哈希值一次性更新到树中,避免了实时更新树带来的资源损耗问题,在数据查询场景下能够提供更好的性能和响应速度,同时保持数据更新的高效性。

[42]在进行数据查询时可以先访问日志再访问树,如果查询指令指示的目标元组的哈希值记录在日志中可以直接返回给第二客户端,有利于提高查询效率;对于未记录在日志中的目标元组,可以利用验证结构中的树获得证明路径并返回给第二客户端进行数据校验。并且由于上述的验证结构的即时更新,使得数据查询场景下可以对最新更新的数据进行数据验证,在数据查询场景下能够提供更好的性能和响应速度。

[43]本说明书实施例提供了一种验证结构更新方法以及一种基于验证结构的数据校验方法可以由服务端来执行。示例性的，该服务端承担了管理数据库的角色，负责处理客户端的请求并执行相应的操作；以及处理数据库的各种操作，如数据的插入、更新、删除和查询等。验证结构更新方法和基于验证结构的数据校验方法应用于服务端中，用于增强服务端对数据库的数据管理能力，可以确保数据库中的数据始终处于正确和完整的状态，并提供可信的数据校验服务。

[44]示例性的，请参阅图 1，服务端 10 用于管理数据库，并且服务端 10 与第一客户端 20 和第二客户端 30 通信连接，第一客户端 20 是指对数据有更新需求的客户端；第二客户端 30 是指对数据有查询需求的客户端。

[45]在一些可能的应用场景中，第一客户端 20 和第二客户端 30 为不同的客户端，比如第一客户端 20 为与数据提供方对应的客户端，第二客户端 30 为与数据使用方对应的客户端。在另一些可能的应用场景中，第一客户端 20 和第二客户端 30 为同一客户端。

[46]在一些实施例中，请参阅图 2，图 2 为本说明书实施例提供一种验证结构更新方法的流程示意图，该方法可以由负责管理数据库的服务端来执行，验证结构用于对数据库中的数据表进行数据校验；验证结构包括树和日志；树中的各个叶子节点用于记录数据表中的各个元组的哈希值，以及树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到；其中，树的根节点记录的根哈希值和日志作为校验凭证被同步至对数据表有查询需求的第二客户端中；该方法包括 S101 至 S102。

[47]在 S101 中，响应于第一客户端对数据表中的元组的更新指令，获取更新后的元组的哈希值，将更新后的元组的哈希值记录在日志中，并将日志同步至对数据表有查询需求的第二客户端中。

[48]示例性的，由于日志中记录了更新后的元组的哈希值，也即日志发生了变动，因此需要在将更新后的元组的哈希值记录在日志中之后，将日志同步至对数据表有查询需求的第二客户端中，以保证在数据查询阶段进行数据校验的准确性。

[49]在 S102 中，响应于日志满足更新条件，利用日志中记录的各个元组的哈希值更新树中与各个元组对应的叶子节点，并基于更新后的叶子节点记录的哈希值更新树中的非叶子节点，得到更新后的树以及将记录的哈希值更新到树中的更新后的日志，将更新后的树的根节点记录的根哈希值和更新后的日志同步至对数据表有查询需求的第二客户端中。

[50]示例性的，当日志满足更新条件时，服务端会将日志中记录的各个元组的哈希值批量更新到树中，然后释放日志对应的存储空间。也即树和日志均发生了变动，则服务端需要将更新后的树的根节点记录的根哈希值和更新后的日志同步至对数据表有查询需求的第二客户端中，以保证在数据查询阶段进行数据校验的准确性。

[51]本实施例中，通过将根哈希值和日志作为校验凭证，进而在接收到更新指令后，可以即时更新验证结构中的日志，并同步给对数据表有查询需求的客户端，从而保证了数据校验的实时性；并且只有在满足更新条件时才会将日志中记录的所有哈希值一次性更新到树中，避免了实时更新树带来的资源损耗问题，在数据查询场景下能够提供更好的性能和响应速度，同时保持数据更新的高效性。

[52]需要说明的是，数据库中的数据表均有与之一一对应的验证结构，该验证结构包括树和日志；树中的各个叶子节点用于记录该数据表中的各个元组的哈希值，树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到；日志用于记录待更新到树中的至少一个元组的哈希值。其中，元组是关系数据库中的基本概念，关系是一张表，表中的每行（即数据库中的每条记录）就是一个元组，每列就是一个属性。

[53]示例性的，验证结构中的树包括默克尔树。

[54]示例性的，上述的更新条件包括日志记录的哈希值的数量达到预设阈值；或者，日志的存储容量已满；或者，达到预设的更新时间点等。本实施例对此不做任何限制。

[55]在一些实施例中，请参阅图 3，本说明书实施例还提供了一种基于验证结构的数据校验方法，验证结构基于上述的验证结构更新方法得到；方法包括 S201 至 S203。

[56]在 S201 中，响应于第二客户端对数据库中的数据表的查询指令，查询与数据表对应的日志中是否记录查询指令指示的目标元组的哈希值。

[57]在 S202 中，若是，将日志记录的目标元组的哈希值返回给第二客户端，以使得第二客户端基于目标元组的哈希值和存储在其本地的日志对目标元组进行数据校验。

[58]在 S203 中，若否，查询与数据表对应的树，确定目标元组对应的叶子节点在树中的证明路径并返回给第二客户端，以使得第二客户端基于证明路径和存储在本地的根哈希值进行数据校验。

[59]本实施例中，在进行数据查询时可以先访问日志再访问树，如果查询指令指示的目标元组的哈希值记录在日志中可以直接返回给第二客户端，有利于提高查询效率；对于未记录在日志中的目标元组，可以利用验证结构中的树获得证明路径并返回给第二客户

端进行数据校验。并且由于上述的验证结构的即时更新，使得数据查询场景下可以对最新更新的数据进行数据验证，在数据查询场景下能够提供更好的性能和响应速度。

[60]示例性的，服务端在返回目标元组的哈希值或者目标元组对应的证明路径时，还一并返回从数据库中查询得到的目标元组。

[61]在服务端将目标元组和日志记录的目标元组的哈希值返回给第二客户端时，第二客户端可以查询存储在其本地的日志中记录的目标元组的哈希值与服务端返回的目标元组的哈希值是否一致，若是，则数据校验通过，服务端一并返回的目标元组可信未被修改；若否，则数据校验未通过，说明服务端一并返回的目标元组已被修改。

[62]在服务端将目标元组和目标元组对应的证明路径返回给第二客户端时，第二客户端可以基于目标元组计算哈希值，并基于计算得到的哈希值和证明路径进行哈希聚合反推得到最终的计算结果，若最终的计算结果与客户端持有的根哈希值相同，则数据校验通过，服务端一并返回的目标元组可信未被修改；反之，则数据校验未通过，说明服务端一并返回的目标元组已被修改。

[63]在一些实施例，为了在进行数据查询时能够加速遍历日志，日志中内置有布隆过滤器。BF（Bloom Filter，布隆过滤器）是一种概率型数据结构，用于判断一个元素是否在一个集合中，它的核心思想是利用多个哈希函数和一个二进制向量来表示集合中的元素。布隆过滤器的特点是，它可以告诉你某个元素“肯定不在过滤器中”或者“可能在过滤器中”，布隆过滤器的优点是空间效率和查询时间都比较高，特别适合于处理大规模数据的存在性检测问题。

[64]在进行验证结构的更新时，在将更新后的元组的哈希值记录在日志中之后，服务端可以利用更新后的元组的哈希值更新布隆过滤器。则在进行数据查询时，服务端可以利用日志中的布隆过滤器查询目标元组的哈希值是否记录在日志中；若布隆过滤器反馈目标元组的哈希值未记录在日志中，执行查询树的步骤；否则，遍历日志中记录的哈希值直到查询到目标元组的哈希值。本实施例通过利用布隆过滤器实现提高数据查询效率。

[65]示例性的，为了进一步实现加速遍历，日志包括至少两个存储单元，可以在每个存储单元中均内置一个布隆过滤器。在进行验证结构的更新时，在将更新后的元组的哈希值记录在日志中的其中一个存储单元之后，可以利用更新后的元组的哈希值更新该存储单元的布隆过滤器。则在进行数据查询时，服务端先遍历日志中的各个存储单元的布隆过滤器，直到其中一个存储单元的布隆过滤器反馈目标元组的哈希值可能存储在本存储

单元中、或者所有布隆过滤器均反馈目标元组的哈希值未记录在本存储单元中。

[66]在一些实施例中，对于图 2 中的 S102，当日志满足更新条件时，服务端会将日志中记录的各个元组的哈希值批量更新到树中，然后释放日志对应的存储空间，以便日志可以继续记录新的哈希值。

[67]示例性的，请参阅图 4，假设存在一数据表，该数据表包括由 v1、v2、v3、v4、v5、v6 和 v7 总计 7 行记录，也即 7 个元组。该数据表对应有验证结构，包括树和日志，树的叶子节点中记录有 v1~v7 分别对应的哈希值 h1~h7，第一客户端在数据更新过程中对元组 v2 和 v4 进行了更新，假设更新了元组 v2 和 v4 中的某个属性值；日志中记录有更新后的元组 v2 对应的哈希值 h2' 和更新后的元组 v4 对应的哈希值 h4'。当日志记录的哈希值的数量达到预设阈值（比如预设阈值为 2）时，日志中记录的更新后的元组 v2 对应的哈希值 h2' 和更新后的元组 v4 对应的哈希值 h4' 被批量更新到树中，并且需要相应地更新相关非叶子节点，从而得到更新后的验证结构，其中更新后的验证结构中的树中的灰色节点为需要更新哈希值的节点。

[68]示例性的，为了减少树的整体更新开销，在将日志中记录的哈希值更新到树的过程中，如果日志中记录有针对于同一个元组的多个哈希值，则服务端可以利用多个哈希值中的最新记录的哈希值更新树中与元组对应的叶子节点，也就是说，只需将最新记录的哈希值写回到树中的与该元组对应的叶子节点，减少了同一个元组的多个哈希值重复写入同一个叶子节点带来了更新开销，也有利于提高更新效率。

[69]在一些实施例中，数据库采用 MVCC 多版本并发控制机制来支持事务；每个事务对应唯一的事务标识，事务包括查询事务和更新事务，更新事务包括针对于数据表的更新指令；查询事务包括针对于数据表的查询指令。

[70]MVCC（Multi-Version Concurrency Control，多版本并发控制）是一种数据库并发控制机制，它通过在数据库中维护多个版本的数据来支持并发事务的执行。每个事务在读取数据时，可以看到一个可见的、一致的数据库状态，而不会受到其他事务的影响。

[71]MVCC 的核心思想是通过为每个事务创建一个独立的事务标识，并使用事务标识来判断事务的可见性。事务标识是一个递增的数字。具体来说，当一个事务开始时，它会被分配一个唯一的事务标识。在读取数据时，事务只能看到在其事务标识之前提交的数据。如果其他事务正在修改或删除某个数据项，而该事务的事务标识大于当前事务的事务标识，那么当前事务将无法看到这些修改或删除操作的结果。

[72]数据库为每个修改操作创建一个新的版本，并将旧版本的数据保留在历史记录中。每个版本都与执行该修改操作的事务标识相关联。也就是说，数据表中的各个元组对应至少有一个版本，各个版本均有用于表示该版本的元组所服务的事务标识范围，事务标识范围根据用于更新该版本的元组的更新事务和用于更新下一版本的元组的更新事务两者的事务标识共同确定，换句话说，各个版本所服务的最小事务标识为更新该版本的元组的更新事务的事务标识，以及所服务的最大事务标识小于用于更新下一版本的元组的更新事务的事务标识。

[73]针对于数据库中的数据表的每个操作都会有对应的事务标识，同一个元组的不同版本均会有 $xmin$ （所服务的最小事务标识）、 $xmax$ （所服务的最大事务标识）两项来标记该版本的元组所服务的事务标识范围。在一个例子中，请参阅图 5，一个数据表中，对 id 为 2 的元组进行更新，产生 id 为 2' 的更新后的元组。更新事务 ID 为 150，所以将旧数据行($id=2$)的 $xmax$ 置为 150，设置新数据行($id=2'$)的 $xmin$ 置为 150。事务标识在 $[100, 150)$ 的操作会对 $id=2$ 数据行访问；事务 ID 大于等于 150 的操作会对 $id=2'$ 数据行访问。

[74]为了使验证结构适应于 MVCC 多版本并发控制机制，数据库中的每个数据表对应至少有一个版本的验证结构，各个版本的验证结构所服务的事务标识范围不同。服务端在响应于日志满足更新条件，将日志中记录的各个元组的哈希值批量更新到树中并释放日志对应的存储空间之后，可以获取更新后的树以及将记录的哈希值更新到树中的更新后的日志，然后将更新后的树和更新后的日志确定为最新版本的验证结构；以及根据更新后的树中的各个叶子节点对应的元组所服务的事务标识范围，确定最新版本的验证结构所服务的事务标识范围。本实施例通过确定最新版本的验证结构所服务的事务标识范围，明确各个版本的验证结构所服务的事务标识范围，以便在后续查询过程中能够准确地利用相应版本的验证结果进行数据校验。

[75]请参阅图 4，例如图 4 所示的更新前的验证结构为第一版本的验证结构，在将图 4 所示的验证结构中的日志的哈希值更新到树中之后，可以将更新后的验证结构确定为第二版本的验证结构。两个版本的验证结构所服务的事务标识范围不同，结合图 5 中的 id 为 2 的元组举例，假设第一版本的验证结构用于对 id 为 2 的元组进行数据校验，所服务的事务标识范围为 $[100, 150)$ ，第二版本的验证结构用于对 id 为 2' 的元组进行数据校验，所服务的事务标识范围为 $[150, +\infty)$ 。

[76]则在进行数据查询时，响应于第二客户端对数据库中的数据表的查询指令，服务端首先获取查询指令携带的查询事务的事务标识；根据查询指令携带的查询事务的事务标

识和数据表对应的不同版本的验证结构所服务的事务标识范围，确定用于服务查询指令所属查询事务的目标验证结构。然后查询目标验证结构中的日志中是否记录查询指令指示的目标元组的哈希值；若是，将日志记录的目标元组的哈希值返回给第二客户端，以使得第二客户端基于目标元组的哈希值和存储在其本地的日志对目标元组进行数据校验。若否，查询目标验证结构中的树，确定目标元组对应的叶子节点在树中的证明路径并返回给第二客户端，以使得第二客户端基于证明路径和存储在本地的根哈希值进行数据校验。本实施例根据查询事务的事务标识确定目标验证结构，可以直接定位到服务该查询事务的验证结构，这样可以减少不必要的遍历和校验过程，提高查询的效率和响应速度。

[77]在一些实施例中，服务端可以实时或者周期性地确定数据库当前进行的事务中的最小事务标识，然后将最小事务标识与不同版本的验证结构所服务的事务标识范围分别进行比对，以确定不同版本的验证结构中满足删除条件的待删除验证结构并将其进行删除；其中，该删除条件包括待删除验证结构所服务的最大事务标识小于数据库当前进行的事务中的最小事务标识。本实施例通过删除不再使用的验证结构，实现对验证结构的有效回收，可以释放存储空间，提高数据库的整体性能和效率。

[78]在一些实施例中，为了进一步提高数据库的更新查询性能，实现将验证功能和索引功能相结合。上述的验证结构还用于进行数据索引；树中的各个叶子节点还用于记录数据表中的各个元组的索引值和存储地址；树中的各个非叶子节点还记录用于表示查找顺序的元组的索引值；日志还用于记录待更新到树中的至少一个元组的索引值和存储地址；其中，各个元组的索引值包括该元组的主键或者该元组中除了主键之外的属性值。本实施例使用验证结构进行数据索引，可以加快查询的速度，验证结构中的树可以记录元组的索引值和存储地址，通过索引值的查找，可以快速定位到目标元组的存储位置，减少了遍历整个数据表的开销，提高了查询的效率。并且通过将元组的索引值记录在非叶子节点中，可以实现对元组的有序访问，在查询或者范围查找操作中，可以根据索引值的顺序进行遍历，减少磁盘 IO 的次数，提高数据访问的效率。通过将验证结构用于数据索引，还可以保证索引的一致性和完整性，在更新操作时，先更新验证结构，再更新数据表，可以确保索引和数据的一致性，这样可以避免索引损坏或者数据异常的情况，提升数据库的可靠性。

[79]示例性的，验证结构中的树包括由默克尔树结构和 B 树结构结合得到的默克尔 B 树。默克尔 B 树将默克尔树和 B 树结构相结合，以实现索引和验证功能；在默克尔 B 树中，

每个叶子节点存储了数据表中的元组的索引值和存储地址，并计算了元组数据的哈希值。非叶子节点存储了用于表示查找顺序的元组的索引值，并计算了子节点的哈希值。通过默克尔树的哈希值机制，可以验证树中的数据完整性和一致性。默克尔 B 树使得在验证结构中进行索引和验证操作更加高效和可靠，它既利用了 B 树的平衡性和多路搜索的特点，减少了磁盘 IO 的次数，又借助了默克尔树的哈希值机制，保证了数据的完整性和一致性，这样可以提高验证结构的性能和效率，同时确保数据的安全和可靠性。

[80]在一个例子中，请参阅图 6，示出了用于数据验证以及索引的默克尔 B 树。假设存在一数据表，该数据表包括由 v1、v2、v3、v4、v5、v6 和 v7 总计 7 行记录，也即 7 个元组；默克尔 B 树中的各个叶子节点分别记录有 v1~v7 分别对应的<索引值，哈希值，元组的存储地址>，默克尔 B 树的非叶子节点用于记录用于表示查找顺序的元组的索引值和由其孩子节点的哈希值聚合得到的哈希值，比如根节点中的 v4 对应的索引值表示从左子树查找 v1~v4 对应的<索引值，哈希值，元组的存储地址>，根节点中的 v5 对应的索引值表示从右子树查找 v5~v7 对应的<索引值，哈希值，元组的存储地址>；根节点的孩子节点中的索引值表示的含义类似，此处不在赘述。

[81]则在进行数据查询时，服务端响应于第二客户端对数据库中的数据表的查询指令，根据查询指令携带的目标元组的索引值，查询与数据表对应的日志中是否记录有目标元组的存储地址和哈希值；若是，从日志记录的目标元组的存储地址读取目标元组，将目标元组和日志记录的目标元组的哈希值返回给第二客户端；若否，查询树以确定目标元组对应的叶子节点，从目标元组对应的叶子节点记录的存储地址读取目标元组，并基于目标元组对应的叶子节点确定证明路径，将目标元组和目标元组对应的叶子节点在树中的证明路径返回给第二客户端。本实施例中，通过在日志记录中查询目标元组的存储地址和哈希值，可以快速定位到目标元组并返回给第二客户端，从而提高了查询的速度和效率；并且通过将索引功能和验证功能相结合，在索引目标元组的存储地址时能够一并获取到进行数据校验的哈希值，有利于提高更新查询以及数据校验的性能。

[82]相应地，请参阅图 7，本说明书实施例还提供了一种基于验证结构的数据校验方法，应用于第二客户端，所述第二客户端持有的用于进行数据校验的校验凭证基于上述的验证结构更新方法得到，所述校验凭证包括与数据库中的数据表对应的根哈希值和日志，所述日志记录有所述数据表中的至少一个元组的哈希值。所述方法包括 S301 至 S303。

[83]在 S301 中，向所述数据库发送针对所述数据表的查询指令。

[84]在 S302 中，若所述数据库返回所述查询指令指示的目标元组的哈希值，基于所述目

标元组的哈希值和存储在其本地的所述日志对所述目标元组进行数据校验。

[85]在 S303 中，若所述数据库返回所述查询指令指示的目标元组的证明路径，基于所述证明路径和存储在本地地的所述根哈希值进行数据校验。

[86]本实施例中，第二客户端持有根哈希值和日志作为校验凭证，从而可以基于根哈希值和日志进行数据校验。

[87]示例性的，服务端在返回目标元组的哈希值或者目标元组对应的证明路径时，还一并返回从数据库中查询得到的目标元组。

[88]在服务端将目标元组和日志记录的目标元组的哈希值返回给第二客户端时，第二客户端可以查询存储在其本地的日志中记录的目标元组的哈希值与服务端返回的目标元组的哈希值是否一致，若是，则数据校验通过，服务端一并返回的目标元组可信未被修改；若否，则数据校验未通过，说明服务端一并返回的目标元组已被修改。

[89]在服务端将目标元组和目标元组对应的证明路径返回给第二客户端时，第二客户端可以基于目标元组计算哈希值，并基于计算得到的哈希值和证明路径进行哈希聚合反推得到最终的计算结果，若最终的计算结果与客户端持有的根哈希值相同，则数据校验通过，服务端一并返回的目标元组可信未被修改；反之，则数据校验未通过，说明服务端一并返回的目标元组已被修改。

[90]以上实施方式中的各种技术特征可以任意进行组合，只要特征之间的组合不存在冲突或矛盾，但是限于篇幅，未进行一一描述，因此上述实施方式中的各种技术特征的任意进行组合也属于本说明书公开的范围。

[91]在一些实施例中，本说明书实施例还提供了一种电子设备，包括：处理器；用于存储处理器可执行指令的存储器；其中，所述处理器通过运行所述可执行指令以实现上述中任一项所述的方法。

[92]示例性的，图 8 是一示例性实施例提供的一种电子设备的示意结构图。请参考图 8，在硬件层面，该设备包括处理器 802、内部总线 804、网络接口 806、内存 808 以及非易失性存储器 810，当然还可能包括其他场景所需要的硬件。本说明书一个或多个实施例可以基于软件方式来实现，比如由处理器 802 从非易失性存储器 810 中读取对应的计算机程序到内存 808 中然后运行。当然，除了软件实现方式之外，本说明书一个或多个实施例并不排除其他实现方式，比如逻辑器件抑或软硬件结合的方式等等，也就是说以下处理流程的执行主体并不限定于各个逻辑单元，也可以是硬件或逻辑器件。

[93]在一些实施例中，本说明书实施例还提供了一种计算机可读存储介质，其上存储有计算机指令，该指令被处理器执行时实现如上述任一项所述方法的步骤。

[94]在一些实施例中，本说明书实施例还提供了一种计算机程序产品，包括计算机程序，该计算机程序被处理器执行时实现上述任一项所述方法的步骤。

[95]需要说明的是，本说明书所涉及的用户信息（包括但不限于用户设备信息、用户个人信息等）和数据（包括但不限于用于分析的数据、存储的数据、展示的数据等），均为经用户授权或者经过各方充分授权的信息和数据，并且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准，并提供有相应的操作入口，供用户选择授权或者拒绝。

[96]上述实施例阐明的系统、装置、模块或单元，具体可以由计算机芯片或实体实现，或者由具有某种功能的产品来实现。一种典型的实现设备为计算机，计算机的具体形式可以是个人计算机、膝上型计算机、蜂窝电话、相机电话、智能电话、个人数字助理、媒体播放器、导航设备、电子邮件收发设备、游戏控制台、平板计算机、可穿戴设备或者这些设备中的任意几种设备的组合。

[97]在一个典型的配置中，计算机包括一个或多个处理器（CPU）、输入/输出接口、网络接口和内存。

[98]内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器（RAM）和/或非易失性内存等形式，如只读存储器（ROM）或闪存（flash RAM）。内存是计算机可读介质的示例。

[99]计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存（PRAM）、静态随机存取存储器（SRAM）、动态随机存取存储器（DRAM）、其他类型的随机存取存储器（RAM）、只读存储器（ROM）、电可擦除可编程只读存储器（EEPROM）、快闪记忆体或其他内存技术、只读光盘只读存储器（CD-ROM）、数字多功能光盘（DVD）或其他光学存储、磁盒式磁带、磁盘存储、量子存储器、基于石墨烯的存储介质或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体（transitory media），如调制的数据信号和载波。

[100]还需要说明的是，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性

的包含，从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[101]上述对本说明书特定实施例进行了描述。其它实施例在所附权利要求书的范围内。在一些情况下，在权利要求书中记载的动作或步骤可以按照不同于实施例中的顺序来执行并且仍然可以实现期望的结果。另外，在附图中描绘的过程不一定要求示出的特定顺序或者连续顺序才能实现期望的结果。在某些实施方式中，多任务处理和并行处理也是可以的或者可能是有利的。

[102]在本说明书一个或多个实施例使用的术语是仅仅出于描述特定实施例的目的，而非旨在限制本说明书一个或多个实施例。在本说明书一个或多个实施例和所附权利要求书中所使用的单数形式的“一种”、“所述”和“该”也旨在包括多数形式，除非上下文清楚地表示其他含义。还应当理解，本文中使用的术语“和/或”是指并包含一个或多个相关联的列出项目的任何或所有可能组合。

[103]应当理解，尽管在本说明书一个或多个实施例可能采用术语第一、第二、第三等来描述各种信息，但这些信息不应限于这些术语。这些术语仅用来将同一类型的信息彼此区分开。例如，在不脱离本说明书一个或多个实施例范围的情况下，第一信息也可以被称为第二信息，类似地，第二信息也可以被称为第一信息。取决于语境，如在此所使用的词语“如果”可以被解释成为“在……时”或“当……时”或“响应于确定”。

[104]以上所述仅为本说明书一个或多个实施例的较佳实施例而已，并不用以限制本说明书一个或多个实施例，凡在本说明书一个或多个实施例的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本说明书一个或多个实施例保护的范围之内。

权利要求书

1、一种验证结构更新方法，所述验证结构用于对数据库中的数据表进行数据校验；所述验证结构包括树和日志；所述树中的各个叶子节点用于记录所述数据表中的各个元组的哈希值，以及所述树中的非叶子节点记录的哈希值由其孩子节点记录的哈希值聚合得到；其中，所述树的根节点记录的根哈希值和所述日志作为校验凭证被同步至对所述数据表有查询需求的第二客户端中；所述方法包括：

响应于第一客户端对所述数据表中的元组的更新指令，获取更新后的元组的哈希值，将更新后的元组的哈希值记录在所述日志中，并将所述日志同步至对所述数据表有查询需求的第二客户端中；

响应于所述日志满足更新条件，利用所述日志中记录的各个元组的哈希值更新所述树中与各个所述元组对应的叶子节点，并基于更新后的叶子节点记录的哈希值更新所述树中的非叶子节点，得到更新后的树以及将记录的哈希值更新到树中的更新后的日志，将所述更新后的树的根节点记录的根哈希值和所述更新后的日志同步至对所述数据表有查询需求的第二客户端中。

2、根据权利要求1所述的方法，所述日志中内置有布隆过滤器；

在将更新后的元组的哈希值记录在所述日志中之后，还包括：利用所述更新后的元组的哈希值更新所述布隆过滤器；

和/或

所述更新条件包括：所述日志记录的哈希值的数量达到预设阈值。

3、根据权利要求1所述的方法，所述利用所述日志中记录的各个元组的哈希值更新所述树中与各个所述元组对应的叶子节点，包括：

若所述日志中记录有针对于同一个元组的多个哈希值，利用多个哈希值中的最新记录的哈希值更新所述树中与所述元组对应的叶子节点。

4、根据权利要求1所述的方法，所述数据库采用 MVCC 多版本并发控制机制来支持事务；每个事务对应唯一的事务标识，所述事务包括查询事务和更新事务，所述更新事务包括所述更新指令；

所述数据表中的各个元组对应至少一个版本，各个版本均有用于表示该版本的元组所服务的事务标识范围，所述事务标识范围根据用于更新该版本的元组的更新事务和用于更新下一版本的元组的更新事务两者的事务标识确定；

所述数据表对应至少一个版本的验证结构，各个版本的验证结构所服务的事务标识范围不同；

在所述将所述更新后的树的根节点记录的哈希值和所述更新后的日志同步至对所述数据表有查询需求的第二客户端之前，还包括：

将所述更新后的树和所述更新后的日志确定为最新版本的验证结构；以及

根据所述更新后的树中的各个叶子节点对应的元组所服务的事务标识范围，确定所述最新版本的验证结构所服务的事务标识范围。

5、根据权利要求4所述的方法，还包括：

确定所述数据库当前进行的事务中的最小事务标识；

将所述最小事务标识与不同版本的验证结构所服务的事务标识范围分别进行比对，以确定所述不同版本的验证结构中满足删除条件的待删除验证结构并将其进行删除；其中，所述删除条件包括待删除验证结构所服务的最大事务标识小于所述最小事务标识。

6、根据权利要求1所述的方法，所述验证结构还用于进行数据索引；所述树中的各个叶子节点还用于记录所述数据表中的各个元组的索引值和存储地址；所述树中的各个非叶子节点还记录用于表示查找顺序的元组的索引值；所述日志还用于记录待更新到树中的至少一个元组的索引值和存储地址；

其中，各个元组的索引值包括该元组的主键或者该元组中除了主键之外的属性值。

7、一种基于验证结构的数据校验方法，所述验证结构基于权利要求1至6任意一项所述的验证结构更新方法得到；所述方法包括：

响应于第二客户端对数据库中的数据表的查询指令，查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值；

若是，将所述日志记录的所述目标元组的哈希值返回给所述第二客户端，以使得所述第二客户端基于所述目标元组的哈希值和存储在其本地的所述日志对所述目标元组进行数据校验；

若否，查询与所述数据表对应的树，确定所述目标元组对应的叶子节点在所述树中的证明路径并返回给所述第二客户端，以使得所述第二客户端基于所述证明路径和存储在本地的根哈希值进行数据校验。

8、根据权利要求7所述的方法，所述日志中内置有布隆过滤器；

所述查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值，包括：

利用所述日志中的布隆过滤器查询所述目标元组的哈希值是否记录在所述日志中；

若所述布隆过滤器反馈所述目标元组的哈希值未记录在所述日志中，执行查询所述树的步骤；否则，遍历所述日志中记录的哈希值直到查询到所述目标元组的哈希值。

9、根据权利要求 7 所述的方法，所述数据库采用 MVCC 多版本并发控制机制来支持事务；每个事务对应唯一的事务标识，所述事务包括查询事务和更新事务，所述查询事务包括所述查询指令；

在所述查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值之前，还包括：

获取所述查询指令携带的查询事务的事务标识；

根据所述查询指令携带的查询事务的事务标识和所述数据表对应的不同版本的验证结构所服务的事务标识范围，确定用于服务所述查询指令所属查询事务的目标验证结构；

所述查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值，包括：

查询所述目标验证结构中的所述日志中是否记录所述查询指令指示的目标元组的哈希值；

所述查询与所述数据表对应树，包括：

查询所述目标验证结构中的所述树。

10、根据权利要求 7 所述的方法，所述验证结构还用于进行数据索引；

所述查询与所述数据表对应的日志中是否记录所述查询指令指示的目标元组的哈希值，包括：

根据所述查询指令携带的目标元组的索引值，查询与所述数据表对应的日志中是否记录有所述目标元组的存储地址和哈希值；

所述方法还包括：

若是，从所述日志记录的所述目标元组的存储地址读取所述目标元组，将所述目标元组和所述日志记录的所述目标元组的哈希值返回给所述第二客户端；

若否，查询所述树以确定所述目标元组对应的叶子节点；从所述目标元组对应的叶子节点记录的存储地址读取所述目标元组，并将所述目标元组和所述目标元组对应的叶子节点在所述树中的证明路径返回给所述第二客户端。

11、一种基于验证结构的数据校验方法，应用于第二客户端，所述第二客户端持有的校验凭证基于权利要求 1 至 6 任意一项所述的验证结构更新方法得到，所述校验凭证包括与数据库中的数据表对应的根哈希值和日志，所述日志记录有所述数据表中的至少一个元组的哈希值；

所述方法包括：

向所述数据库发送针对所述数据表的查询指令;

若所述数据库返回所述查询指令指示的目标元组的哈希值,基于所述目标元组的哈希值和存储在其本地的所述日志对所述目标元组进行数据校验;

若所述数据库返回所述查询指令指示的目标元组的证明路径,基于所述证明路径和存储在本地的所述根哈希值进行数据校验。

12、一种电子设备,包括:

处理器;

用于存储处理器可执行指令的存储器;

其中,所述处理器通过运行所述可执行指令以实现如权利要求 1 至 11 中任一项所述的方法。

13、一种计算机可读存储介质,其上存储有计算机指令,该指令被处理器执行时实现如权利要求 1 至 11 中任一项所述方法的步骤。

14、一种计算机程序产品,包括计算机程序,该计算机程序被处理器执行时实现权利要求 1 至 11 任意一项所述方法的步骤。