

数据库操作方法、装置、设备及存储介质

5 本申请要求于 2022 年 03 月 04 日提交中国专利局、申请号为 202210208474.X、申请名称为“数据库操作方法、装置、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申请中。

技术领域

本申请涉及数据库技术领域，尤其涉及一种数据库操作方法、装置、设备及存储介质。

背景技术

10 目前，可以采用多叉树存储数据库中的表，多叉树例如可以为平衡多路查找树（B-Tree）、B+Tree 等。

通常，如果一写线程需要修改某个多叉树的结构，则该写线程需要先给该多叉树整体加读写锁（SX latch），通过对整棵树加读写锁可以避免在该写线程修改该多叉树的结构的过程中，其他写线程也修改该多叉树的结构所带来的冲突。如果有另一写线程也需要修改该多叉树的结构，则需要等待该写线程释放对该多叉树整体加的读写锁之后才能进行。

因此，存在由于针对同一多叉树的结构修改操作（Structure Modification Operation, SMO）无法并发，导致处理效率较低的问题。

发明内容

20 本申请实施例提供一种数据库操作方法、装置、设备及存储介质，用以解决现有技术中针对同一多叉树的结构修改操作无法并发，导致处理效率较低的问题。

第一方面，本申请实施例提供一种数据库操作方法，所述数据库使用多叉树作为索引，所述多叉树的节点与内存页面一一对应，所述方法包括：

写线程从所述多叉树的根节点开始遍历，查找与写操作的搜索关键字匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

25 如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；

30 以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

第二方面，本申请实施例提供一种数据库操作装置，所述数据库使用多叉树作为索引，所述多叉树的节点与内存页面一一对应，所述装置包括：

35 查找模块，用于写线程从所述多叉树的根节点开始遍历，查找与写操作的搜索关键字

匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

修改模块，用于如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；

- 5 所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

- 10 第三方面，本申请实施例提供一种计算机设备，包括：存储器、处理器；其中，所述存储器用于存储一条或多条计算机指令，其中，所述一条或多条计算机指令被所述处理器执行时实现如第一方面中任一项所述的方法。

第四方面，本申请实施例提供一种计算机可读存储介质，其上存储有计算机程序，当所述计算机程序被执行时，实现如第一方面中任一项所述的方法。

- 15 本申请实施例还提供一种计算机程序，当所述计算机程序被计算机执行时，用于实现如第一方面任一项所述的方法。

- 在本申请实施例中，写线程不是申请为某个多叉树整体加读写锁，而是按照加锁规则申请为目标节点及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为目标节点和其他节点加排它锁后，对目标节点和其他节点进行相应修改，目标节点是指查找到的与搜索关键字匹配的索引项所在的节点，实现了单个写线程可以通过仅对多叉树中其需要修改的节点对应的内存页面加排它锁来进行多叉树结构的修改，使得多个写线程之间的冲突是在页面级别，从而使得针对同一多叉树的结构修改操作能够并发，从而能够提高处理效率。另外，通过加锁规则包括如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有任意节点对应内存页面的排它锁的基础上，申请为关联节点对应的内存页面加排它锁之前，需要释放该任意节点对应内存页面的排他锁，并为该任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置预设标记，则需要释放自身持有的锁，使得按照加锁规则申请加锁，能够避免出现线程间死锁的问题。

#### 附图说明

- 30 为了更清楚地说明本申请实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作一简单地介绍，显而易见地，下面描述中的附图是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图1为本申请实施例提供的数据库访问系统的结构示意图；

图2为本申请实施例提供的设置预设标记的示意图；

- 35 图3为本申请实施例提供的读写锁SX、共享锁S以及排它锁X之间兼容关系的示意图；

图4为本申请实施例提供的数据库操作方法的流程示意图；

图5A-图5D为本申请一实施例提供的插入操作的处理过程的示意图；

- 图 6A-图 6D 为本申请另一实施例提供的插入操作的处理过程的示意图；  
图 7A-图 7D 为本申请一实施例提供的删除操作的处理过程的示意图；  
图 8A-图 8D 为本申请另一实施例提供的删除操作的处理过程的示意图；  
图 9A-图 9D 为本申请又一实施例提供的删除操作的处理过程的示意图；  
5 图 10 为本申请一实施例提供的数据库操作装置的结构示意图；  
图 11 为本申请一实施例提供的计算机设备的结构示意图。

具体实施方式

为使本申请实施例的目的、技术方案和优点更加清楚，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，显然，所描述的实施例是本  
10 申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员  
在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

在本申请实施例使用的术语是仅仅出于描述特定实施例的目的，而非旨在限制本申  
请。在本申请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，“多种”一般包含至少两种，但  
15 是不排除包含至少一种的情况。

应当理解，本文中使用的术语“和/或”仅仅是一种描述关联对象的关联关系，表示  
可以存在三种关系，例如，A 和/或 B，可以表示：单独存在 A，同时存在 A 和 B，单独存  
在 B 这三种情况。另外，本文中字符“/”，一般表示前后关联对象是一种“或”的关系。

取决于语境，如在此所使用的词语“如果”、“若”可以被解释成为“在……时”或  
20 “当……时”或“响应于确定”或“响应于检测”。类似地，取决于语境，短语“如果确  
定”或“如果检测（陈述的条件或事件）”可以被解释成为“当确定时”或“响应于确定”  
或“当检测（陈述的条件或事件）时”或“响应于检测（陈述的条件或事件）”。

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商品或者系统不仅包括那些要素，而且还包括没有明确  
25 列出的其他要素，或者是还包括为这种商品或者系统所固有的要素。在没有更多限制情况  
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商品或者系统中还  
存在另外的相同要素。

另外，下述各方法实施例中的步骤时序仅为一种举例，而非严格限定。

下面通过一个示例性的应用场景具体说明本申请各个实施例提供的数据库操作方法。  
30 图 1 为本申请一实施例提供的数据库访问系统的结构示意图。如图 1 所示，该数据库  
访问系统可以包括数据库引擎 11 以及数据库（Database）12。

数据库 12 是建立在计算机存储设备上，支持以行式存储架构来组织、存储和管理数  
据的仓库。在本申请实施例中，并不限定承载数据库 12 的计算机存储设备的实现形式。  
数据库 12 可以是但不限于行式数据库，还可以是行列混合的数据库。

35 数据库引擎 11 是用于存储、处理和保护数据的核心服务，其任务包括处理与数据库  
12 相关的事务，例如设计并创建数据库 12，保存数据库 12 所需的各种列表和文档等，针  
对数据库 12 提供日常管理支持以优化数据库 12 的性能。以数据库 12 为 MySQL 数据库为  
例，数据库引擎 11 例如可以为 InnoDB。

为了便于处于与数据库 12 相关的事务，数据库引擎 11 与数据库 12 之间可以建立通信连接，该通信连接可以是有线或无线网络连接。可选的，在部署实现上，数据库引擎 11 与数据库 12 可以部署在同一物理设备上实现，也可以部署在不同物理设备上实现。当数据库引擎 11 与数据库 12 部署在不同物理设备上实现时，两者可以部署同一局域网内，也可以部署在不同局域网内。

数据库引擎 11 使用索引组织表，每个表的数据都放在一个对应的索引中，该索引称为聚集索引 (clustered index)，使用索引组织表的目的是：动态地组织磁盘文件结构，维护数据记录有序；借助索引快速定位记录。除了聚集索引，一个表中的其它索引称为二级索引 (secondary indexes)。二级索引的每个索引项 (record) 除了包含本身的 columns，还包含其对应的数据行的主关键字 (primary key)，数据库引擎 11 可以利用主关键字去主索引找到完整的行 (row)。

数据库引擎 11 可以使用多叉树作为索引的数据结构，多叉树的节点与内存页面 (page) 一一对应，节点可以存储在对应的内存页面中，多叉树例如可以包括 B-Tree、B+Tree 等，以下主要以 B-Tree 为例进行具体说明。B-Tree 的结构可以有如下几点特性：1) 实际数据全部存储在叶子 (leaf) 层；2) 非叶子 (non-leaf) 层只存储索引项 (key, page no)，每个索引项指向唯一一个孩子 (child) 节点；3) 一个索引项的 key 为 P，它的孩子节点只能存 key  $\geq$  P 并且  $< P_1$  的索引项，其中  $P_1$  是下一个索引项的 key；4) 每层节点通过双向链表串起来。

如图 1 所示，该数据库访问系统还可以包括客户端 13。客户端 13 与数据库引擎 11 之间通信连接，该通信连接可以是有线或无线网络连接。可选的，客户端 13 可以与数据库引擎 11 处于同一局域网内，也可以处于不同局域网内。

客户端 13 可以看作是数据库 12 面向用户提供的交互接口，允许用户通过客户端 13 访问数据库 12。在需要访问数据库 12 时，客户端 13 可以向数据库引擎 11 发送数据库访问请求，数据库引擎 11 还可以响应客户端 13 的数据库访问申请为数据库 12 进行相应操作并向客户端 13 返回相应操作结果。

基于客户端 13 发送的数据库访问请求，数据库引擎 11 可以确定操作类型以及搜索关键字。操作类型可以包括增加、删除和查询，插入操作和删除操作可以统称为写操作，查询操作可以称为读操作。需要说明的是，关于数据库引擎 11 确定操作类型以及搜索关键字的实现方式，可以参考相关技术中的具体描述，在此不再赘述。

其中，对于读操作，数据库引擎 11 可以通过创建对应的读线程对数据库 12 进行具体操作。对于写操作，数据库引擎 11 可以通过创建对应的写线程对数据库 12 进行具体操作。在实际应用中，会存在写线程需要修改多叉树的结构的情况，修改多叉树的结构可以包括向多叉树中增加节点，或者从多叉树中删除节点。需要说明的是，如果一个写线程需要对多叉树的结构进行修改，表示该写线程对应的写操作需要触发结构修改操作。

假设把多叉树存储的索引看成一个黑盒，不关心内部具体的数据结构，外面看来只有有序排列的索引项，所有的读取、插入、删除等操作都能原子地一步完成。这时多线程并发操作不会感知到索引结构，只需要考虑事务层面的约束，例如数据库引擎 11 中一个事务对索引项加逻辑锁 (lock) 阻止其它事务访问。但是实际的多叉树操作不是原子的，例如，一个写线程在做结构修改操作时会涉及多个内存页面的改动。

通常，如果一写线程需要修改某个多叉树的结构，则该写线程需要先给该多叉树整体加读写锁，如果有另一写线程也需要修改该多叉树的结构，则需要等待该写线程释放对该多叉树整体加的读写锁之后才能进行，因此存在针对同一多叉树的结构修改操作无法并发，导致处理效率较低的问题。

- 5 为了解决针对同一多叉树的结构修改操作无法并发，导致处理效率较低的技术问题，在本申请实施例中，写线程不是申请为某个多叉树整体加读写锁，而是按照加锁规则申请为目标节点及需要关联修改的其他节点对应的内存页面加排它锁（X latch），以在成功为目标节点和其他节点加排它锁后，对目标节点和其他节点进行相应修改，目标节点是指查找到的与搜索关键字匹配的索引项所在的节点，实现了单个写线程可以通过仅对多叉树
- 10 中其需要修改的节点对应的内存页面加排它锁来进行多叉树结构的修改，使得多个写线程之间的冲突是在页面级别，从而使得针对同一多叉树的结构修改操作能够并发，从而能够提高处理效率。另外，通过加锁规则包括如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有任意节点对应内存页面的排它锁的基础上，申请为关联节点对应的内存
- 15 页面加排它锁之前，需要释放该任意节点对应内存页面的排他锁，并为该任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置预设标记，则需要释放自身持有的锁，使得按照加锁规则申请加锁，能够避免出现线程间死锁的问题。

本申请实施例中，多个对应的写操作会触发结构修改操作的线程会同时进入多叉树，每个线程要拿多个内存页面的排它锁，为了避免线程间死锁设计了如下3点加锁规则。

- 20 规则1：规定加锁方向，加锁方向可以根据需要灵活设计。示例性的，加锁方向包括沿着多叉树自上而下、从左到右的方向，以下主要以加锁方向为沿着多叉树自上而下、从左到右的方向为例进行具体说明。

- 规则2：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有该任意节点对应内存页面的排它锁的基础上，申请为该关联节点对应的内存页面加排它锁之前，需要释放该任意节点对应内存页面的排他锁，并为该任意节点对应的内存页面设置预设标记。
- 25

示例性的，任一写线程在做结构修改操作的过程中，申请父节点或左侧的邻居节点对应的内存页面的排他锁之前，需要对当前持有排它锁的内存页面设置预设标记，并释放当前持有的排他锁。

- 30 由于结构修改操作涉及到从下到上、从右到左的修改节点，与自上而下、从左到右的加锁方向相反，不能采用持有排它锁再去反向加排它锁的方式，因此本申请实施例提出了采用预设标记来解决这个问题，线程反向加锁前将持有排它锁的内存页面设置预设标记。

- 其中，某个内存页面有预设标记可以表示该内存页面处于结构修改操作的中间状态，即针对该内存页面进行修改的结构修改操作还未完成，该内存页面可以记为处于SMO中间状态的内存页面，相应的，该内存页面对应的节点可以记为处于SMO中间状态的节点。应
- 35 理解，一旦任一结构修改操作完成，与该结构修改操作对应的预设标记应该被清除。

一个实施例中，可以使用读写锁作为预设标记。需要说明的是，释放任一节点对应内存页面的排它锁与为该任一节点对应的内存页面加读写锁可以具有原子性，即可以在释放任一节点对应内存页面的排它锁的同时为该任一节点对应的内存页面加读写锁，也即可以同时完成排它锁的释放和预设标记的设置。以下主要以使用读写锁作为预设标记为例进行

具体说明。

以目标节点为节点 1，目标节点右侧的邻居节点为节点 2，目标节点的父节点为节点 3 为例，如图 2 所示，可以先将节点 1 和节点 2 的排他锁 X 降级为读写锁 SX，再去申请节点 3 的排它锁 X。

- 5 除了排它锁和读写锁，还有一种锁是共享锁（S latch），读写锁 SX、共享锁 S 以及排它锁 X 三者之间的兼容关系可以如图 3 所示，其中，0 表示可以兼容，X 表示不可以兼容。图 3 中的第一行表示：在一线程已对某一内存页面加共享锁的基础上，其他线程还可以再对该内存页面加共享锁；在一线程已对某一内存页面加读写锁的基础上，其他线程还可以再对该内存页面加共享锁；在一线程已对某一内存页面加排它锁的基础上，其他线程不可以再对该内存页面加共享锁。图 3 中的第二行表示：在一线程已对某一内存页面加共享锁的基础上，其他线程还可以再对该内存页面加读写锁；在一线程已对某一内存页面加读写锁的基础上，其他线程不可以再对该内存页面加读写锁；在一线程已对某一内存页面加排它锁的基础上，其他线程不可以再对该内存页面加读写锁。图 3 中的第三行表示：在一线程已对某一内存页面加共享锁的基础上，其他线程不可以再对该内存页面加排它锁；在一线程已对某一内存页面加读写锁的基础上，其他线程不可以再对该内存页面加排它锁；在一线程已对某一内存页面加排它锁的基础上，其他线程不可以再对该内存页面加排它锁。

规则 3：如果申请加排它锁的任一内存页面已被其他写线程设置预设标记，则需要释放自身持有的锁。

- 20 如果写线程不释放已持有的锁，会与进行结构修改操作的其他写线程之间互相占有并等待，造成死锁。应理解，写线程不应该修改被其他写线程设置有预设标记的内存页面，因为之前的结构修改操作还没提交，写线程遍历到有预设标记的节点，可以等待之前的结构修改操作完成再修改该节点。

- 25 在实际应用中，为了实现“如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁”这一加锁规则，可以在申请为节点加排它锁的已有模式的基础上，提供一种申请为节点加排它锁的新模式。其中，已有模式可以记为常规模式，新模式可以记为非常规模式。

- 30 其中，针对于常规模式，在某一写线程采用常规模式申请为某一内存页面加排他锁时，如果该内存页面未被其他线程加锁，则该写线程可以直接获得加锁成功的结果；如果该内存页面已被其他线程加锁，则该写线程会阻塞直到获得加锁成功的结果。可以看出，任一写线程采用常规模式申请为内存页面加排它锁的结果只有加锁成功。应理解，某个写线程申请为某个内存页面加排它锁的结果为加锁成功，表示该写线程成功为该内存页面加上了排它锁，即该写线程持有了该内存页面的排它锁。

- 35 针对于非常规模式，在某一写线程采用非常规模式申请为某一内存页面加排他锁时，如果该内存页面已被其他线程设置有预设标记，则该写线程可以直接获得加锁失败的结果，而不是继续等待；如果该内存页面未被设置有预设标记且未被其他线程加锁，则该写线程可以直接获得加锁成功的结果；如果该内存页面未被设置有预设标记但被其他线程加锁，则该写线程会阻塞直到获得加锁成功或加锁失败的结果。可以看出，任一写线程采用非常规模式申请为内存页面加排它锁的结果可能是加锁成功，也可能是加锁失败。应理解，某个写线程申请为某个内存页面加排它锁的结果为加锁失败，表示该写线程为该内存页面加

排它锁未成功, 即该写线程未持有该内存页面的排它锁。

需要说明的是, 在使用读写锁作为预设标记时, 由于其他写线程是在释放一内存页面的排它锁的同时为该内存页面加读写锁, 因此, 在其他写线程持有该内存页面的排它锁的情况下, 该写线程再申请该对应内存页面的排它锁, 可以出现该写线程先阻塞一段时间, 后获得加锁失败的结果的情况。

在该写线程采用非常规模式申请为任一内存页面加排它锁的结果为加锁失败时, 可以表征该任一内存页面已被其他写线程设置预设标记, 从而该写线程可以释放自身持有的锁。

在实际应用中, 该写线程可以通过调用加锁函数的方式来申请对内存页面加排它锁, 线程可以将内存页面的页面编号 (page no) 作为加锁函数的输入。一个实施例中, 常规模式和非常规模式对应的加锁函数可以相同, 例如均为 `rw_lock_x_lock_func` 函数。其中, 以预设标记为读写锁为例, 写线程可以在 `rw_lock_x_lock_func` 函数中尝试加排它锁, 失败后判断内存页面是否被其他线程加上了读写锁, 如果是则立即返回加锁失败。

在返回加锁失败后, 该写线程要释放自己持有的锁, 但可以不立即从根节点重试 (retry from root), 否则可能会再次遇到同一个设置有预设标记的内存页面, 不停重试会消耗处理器的资源。这种情况下, 该写线程可以等待内存页面的结构修改操作整体完成再从根节点重试, 因此在获得针对某个内存页面的加锁失败的结果后, 该写线程可以不采用非常规模式申请为该内存页面加排它锁, 而是采用常规模式申请为该内存页面加排它锁, 这样该写线程可以阻塞直到其他写线程的结构修改操作完成并释放排它锁。

下面结合附图, 对本申请的一些实施方式作详细说明。在不冲突情况下, 下述的实施例及实施例中的特征可以相互组合。

图4为本申请一实施例提供的数据库操作方法的流程示意图, 本实施例的方法可以应用于部署有数据库引擎11的计算机设备, 该计算机设备上运行有与写操作对应的写线程。如图4所示, 本实施例的方法可以包括:

步骤41, 写线程从所述多叉树的根节点开始遍历, 查找与写操作的搜索关键字匹配的目标索引项, 以查找到所述目标索引项所在的目标节点;

步骤42, 如果所述写操作需要触发结构修改操作, 所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁, 以在成功为所述目标节点和所述其他节点加排它锁后, 对所述目标节点和所述其他节点进行相应修改; 所述加锁规则包括: 如果任意节点到关联节点的方向不满足加锁方向的要求, 则在持有所述任意节点对应内存页面的排它锁的基础上, 申请为所述关联节点对应的内存页面加排它锁之前, 需要释放所述任意节点对应内存页面的排他锁, 并为所述任意节点对应的内存页面设置预设标记; 以及, 如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记, 则需要释放自身持有的锁。

为了避免死锁, 乐观读写操作通常要持有遍历路径上所有非叶子节点对应的内存页面 (简称为非叶子内存页面 non-leaf page) 的共享锁, 单一操作的加锁范围比较大, 并发操作越多越会加剧锁竞争, 在某些关键节点 (例如, 根节点 root、非叶子节点 non-leaf node) 的竞争会更明显。

为了减少加锁范围, 在遍历过程中可以尽量减小加锁范围, 具体的, 在遍历过程中最多同时持有2个内存页面的共享锁, 因为沿着一个指针从一个节点到另一个节点时, 不能

有其他线程去改变这个指针，因此要先拿到后一个节点对应内存页面的共享锁，再放开前一个节点对应内存页面的共享锁，从而有利于降低并发操作的竞争。

基于此，一个实施例中，本申请实施例提供的方法还可以包括：在需要由任一节点遍历到所述任一节点的后一节点时，所述写线程申请为所述后一节点对应的内存页面加共享锁，并在成功为所述后一节点对应的内存页面加共享锁后，释放所述任一节点对应内存页面的共享锁。

写线程从多叉树的根节点开始遍历是为了查找与写操作的搜索关键字匹配的目标索引项（以下记为目标索引项），目标索引项所在的节点可以记为目标节点。在实际应用中，写线程可以通过调用定位函数的方式，从多叉树的根节点开始遍历，查找与写操作的搜索关键字匹配的目标索引项。

在此情况下，前述对于遍历过程的改进可以实现在定位函数中，前述加锁函数可以在定位函数中被调用，其中，定位函数例如可以为 `btr_cur_search_to_nth_level` 函数。定位函数的输入可以包括搜索关键字、锁模式（`latch_mode`）和层（`level`）。

示例性的，锁模式主要可以包括如下几种：`BTR_SEARCH_LEAF`、`BTR_MODIFY_LEAF`、`BTR_MODIFY_TREE` 和 `BTR_CONT_MODIFY_TREE`。其中，`BTR_SEARCH_LEAF`：定位到叶子内存页面（`leaf page`）上的索引项，并对内存页面加 `S latch`，可以用于对叶子节点进行的读操作。`BTR_MODIFY_LEAF`：定位到叶子内存页面上的索引项，并对内存页面加 `X latch`，准备修改索引项，可以用于对叶子节点进行乐观写操作。`BTR_MODIFY_TREE`：定位到指定层，对目标索引项所在内存页面和其左右邻居内存页面加 `X latch`，准备修改目标索引项并可能发生结构修改操作，可以用于叶子节点和非叶子节点进行悲观写操作。`BTR_CONT_MODIFY_TREE`：定位到指定层的索引项，只对目标索引项所处内存页面加 `X latch`，可以用于非叶子节点进行乐观写操作。

其中，乐观写操作是指不需要修改多叉树的结构的操作，即不需要触发结构修改操作的写操作，悲观写操作是指需要修改多叉树的结构的操作，即需要触发结构修改操作的写操作。根据写操作类型的不同，悲观写操作可以包括悲观插入操作和悲观删除操作。

示例性的，在写线程对应的写操作为插入操作时，写线程可以先使用 `BTR_MODIFY_LEAF` 模式定位到目标节点，并持有目标节点对应内存页面的排它锁，如果发现目标节点对应内存页面的容量不够容纳插入操作需要插入的索引项，可以触发悲观插入。触发悲观插入后，写线程可以释放目标节点对应内存页面的排它锁，重新以 `BTR_MODIFY_TREE` 模式定位，并持有目标节点及其邻居节点对应内存页面的排它锁，然后进入悲观插入流程。

示例性的，在写线程对应的写操作为删除操作时，写线程可以先使用 `BTR_MODIFY_LEAF` 模式定位到目标节点，并持有目标节点对应内存页面的排它锁，如果发现删除目标节点对应内存页面中删除操作需要删除的索引项后内存页面中剩余空间太大，可以触发悲观删除。触发悲观删除后，写线程可以释放目标节点对应内存页面的排它锁，重新以

`BTR_MODIFY_TREE` 模式定位，并持有目标节点及其邻居节点对应内存页面的排它锁，然后进入悲观删除流程。

应理解，在进入悲观插入流程和悲观删除流程之前，写线程是采用非常规模式申请为目标节点及其邻居对应的内存页面加排它锁的方式，来持有目标节点及其邻居节点对应内存页面的排它锁。

本申请实施例中，如果写操作需要触发结构修改操作，写线程可以按照加锁规则申请为目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为目标节点和其他节点加排它锁后，对目标节点和其他节点进行相应修改。其中，需要关联修改的其他节点可以包括目标节点右侧的邻居节点，目标节点左侧的邻居节点，目标节点的父节点中的一个或多个。关于加锁规则的具体内容可以参见前述描述，在此不再赘述。

示例性的，对于悲观插入操作，写线程可以先通过调用分裂函数的方式，确定分裂点（split point）和分裂顺序，分裂函数的输入可以包括页面编号，分裂函数的输出可以包括分裂点和分裂顺序。分裂点可以用于描述目标节点中索引项分裂的位置，对应了目标节点中的一个索引项；分裂顺序可以用于描述是在目标节点的右侧增加节点，还是在目标节点的左侧增加节点，前者可以称为右分裂（right split），后者可以称为左分裂（left split）；分裂函数例如可以为 `btr_page_split_and_insert` 函数。需要说明的是，对于确定分裂点和分裂顺序的实现方式，可以参考相关技术中的具体描述，在此不再赘述。

一个实施例中，在分裂顺序为右分裂的情况下，步骤 42 具体可以包括如下步骤 A1-步骤 A3。

步骤 A1，所述写线程申请为所述目标节点及其右侧的邻居节点对应的内存页面加排它锁，并申请新内存页面作为新节点，所述写线程持有所述新内存页面的排它锁。

其中，如果目标节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。示例性的，写线程可以释放其持有的全部锁。进一步的，本实施例提供的方法还可以包括：如果目标节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。示例性的，写线程可以采用常规模式申请为目标节点对应的内存页面加排它锁，并在成功为目标节点加排它锁后，重新从多叉树的根节点开始遍历，查找目标索引项。例如可以重新使用 `BTR_MODIFY_TREE` 模式调用 `btr_cur_search_to_nth_level` 函数。

如果右侧的邻居节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点右侧的邻居节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该邻居节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

写线程在成功为目标节点及其右侧的邻居节点对应的内存页面加排它锁后，可以执行步骤 A2。

步骤 A2，所述写线程将所述目标节点中的部分索引项移动到所述新节点中，向所述新节点中插入所述插入操作需要插入的索引项，并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项，将所述新节点插入到所述目标节点与所述邻居节点之间。

其中，目标节点中移动到新节点中的部分索引项是根据分裂点确定，部分索引项可以包括分裂点对应的索引项，或者，也可以不包括分裂点对应的索引项。

步骤 A3，所述写线程先为所述目标节点、所述新节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点、所述新节点和所述邻居节点对应内存页面的排它锁，再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁，并

在成功为所述父节点对应的内存页面加排它锁后，所述写线程通过在所述父节点中增加与  
所述新节点对应的索引项，将所述新节点插入到所述父节点下。

其中，如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记，写线程  
释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的父节点对  
5 应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等  
待该父节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引  
项。

示例性的，首先，如图 5A 所示，写线程可以持有目标节点 51 及其右侧的邻居节点 52  
对应内存页面的排它锁 X；然后，如图 5B 所示，写线程可以申请新内存页面（new page）  
10 作为新节点 53，持有新内存页面的排它锁 X，将目标节点 51 中的部分索引项移动到新节  
点 53 中，并将插入操作需要插入的索引项插入到新节点 53 中；之后，如图 5C 所示，写  
线程可以通过修改目标节点 51、新节点 53 和邻居节点 52 中的索引项（每层节点通过双向  
链表串起来的情况下），将新节点 53 插入到目标节点 51 与邻居节点 52 之间；之后，如  
图 5D 所示，写线程可以先在释放目标节点 51、邻居节点 52 和新节点 53 对应内存页面的  
15 排它锁的同时为目标节点 51、邻居节点 52 和新节点 53 对应内存页面加读写锁 SX，再采  
用非常模式申请为目标节点的父节点 54 对应的内存页面加排它锁 X，并在成功为父节点  
54 对应的内存页面加排它锁后，通过在父节点 54 中增加与新节点 53 对应的索引项，将新  
节点 53 插入到父节点 54 下。

需要说明的是，图 5A 至图 5D 中，一个大矩形框表示一个节点，一个小矩形框表示节  
20 点中的一个索引项，且深灰色的小矩形框表示增加的索引项，并且为了便于理解，一节点  
对应内存页面的锁类型是标记在该节点附近。

另一个实施例中，在分裂顺序为左分裂的情况下，步骤 42 具体可以包括如下步骤 B1-  
步骤 B5。

步骤 B1，所述写线程申请为所述目标节点及其左侧的邻居节点对应的内存页面加排它  
25 锁，并申请新内存页面作为新节点，所述写线程持有所述新内存页面的排它锁。

其中，如果目标节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持  
有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点对应的内存页面已被  
其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点对应的  
内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

30 如果左侧的邻居节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持  
有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点左侧的邻居节点对应  
的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等  
待该邻居节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引  
项。

35 写线程在成功为目标节点及其左侧的邻居节点对应的内存页面加排它锁后，可以执行  
步骤 B2。

步骤 B2，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设  
标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再采用所述非常模式  
申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的

内存页面加排它锁后，通过将所述父节点中原本指向所述目标节点的索引项修改为指向所述新节点，将所述新节点插入到所述父节点下。

其中，如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该父节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

步骤 B3，所述写线程采用所述非常规模式申请为所述目标节点和所述邻居节点对应的内存页面加排他锁，并成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，将所述目标节点中的部分索引项移动到所述新节点中，向所述新节点中插入所述插入操作需要插入的索引项，并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项，将所述新节点插入到所述目标节点与所述邻居节点之间。

其中，如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点（或邻居节点）对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

目标节点中移动到新节点中的部分索引项是根据分裂点确定，部分索引项可以包括分裂点对应的索引项，或者，也可以不包括分裂点对应的索引项。

步骤 B4，所述写线程通过在所述父节点中增加与所述目标节点对应的索引项，将所述目标节点插入到所述父节点下。

可选的，虽然父节点到其子节点（即目标节点和邻居节点）的顺序（即从上至下）满足加锁方向（即自上而下）的要求，但是为了能够尽早的释放父节点的排它锁，提高并发性能，所述写线程采用所述非常规模式申请为所述目标节点和所述邻居节点对应的内存页面加排他锁之前还可以包括：所述写线程为所述父节点对应的内存页面设置所述预设标记，并释放所述父节点对应内存页面的排它锁。

相应的，步骤 B4 之前还可以包括：所述写线程为所述目标节点、所述新节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点、所述新节点和所述邻居节点对应内存页面的排它锁；所述写线程采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁。其中，在成功为所述父节点对应的内存页面加排它锁后，所述写线程采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁。

示例性的，首先，如图 6A 所示，写线程可以持有目标节点 61 及其左侧的邻居节点 62 对应内存页面的排它锁 X；然后，如图 6B 所示，写线程可以申请新内存页面作为新节点 63，持有新内存页面的排它锁 X，写线程还可以先在释放目标节点 61 和邻居节点 62 对应内存页面的排它锁的同时为目标节点 61 和邻居节点 62 对应的内存页面加读写锁 SX，再采用非常规模式申请为目标节点的父节点 64 对应的内存页面加排它锁 X，并在成功为父节点 64 对应的内存页面加排它锁后，通过将父节点 64 中原本指向目标节点 61 的索引项修改为指向新节点 63，将新节点 63 插入到父节点 64 下；之后，如图 6C 所示，写线程可以先在释放父节点 64 对应内存页面的排它锁 X 的同时为父节点 64 对应的内存页面加读写锁 SX，

再采用非常规模式请求为目标节点 61 和邻居节点 62 对应的内存页面加排它锁，并在成功为目标节点 61 和邻居节点 62 对应的内存页面加排它锁后，将目标节点 61 中的部分索引项移动到新节点 63 中，将插入操作需要插入的索引项插入到新节点 63 中，并通过修改目标节点 61、新节点 63 和邻居节点 62 中的索引项(每层节点通过双向链表串起来的情况下)，

5 将新节点 63 插入到目标节点 61 与邻居节点 62 之间；之后，如图 6D 所示，所述写线程可以通过在父节点 64 中增加与目标节点 61 对应的索引项，将目标节点 61 插入到父节点 64 下。

需要说明的是，图 6A 至图 6D 中，一个大矩形框表示一个节点，一个小矩形框表示节点中的一个索引项，且深灰色的小矩形框表示增加的索引项，并且为了便于理解，一节点

10 对应内存页面的锁类型是标记在该节点附近。

需要说明的是，在进行分裂的节点（即目标节点）为根节点时，可以先将根节点中的索引项拷贝到一个新节点中，并将新节点作为根节点的子节点进行分裂。

示例性的，对于悲观删除操作，悲观删除操作触发的结构修改操作可以分为丢弃（discard）和压缩（compress）两类。其中，discard 是从多叉树中删除一个空节点；

15 compress 是节点中还有索引项，与其他节点（邻居节点或父节点）合并。

对于 discard，如果删除之前目标节点中仅有一条索引项，发生删除操作时会直接删掉目标节点。一个实施例中，在需要删除目标节点的情况下，步骤 42 具体可以包括如下步骤 C1-步骤 C3。

步骤 C1，所述写线程申请对所述目标节点及其邻居节点对应的内存页面加排它锁。

20 其中，如果目标节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

如果邻居节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。

25 进一步的，本实施例提供的方法还可以包括：如果目标节点的邻居节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该邻居节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

写线程在成功为目标节点及其邻居节点对应的内存页面加排它锁后，可以执行步骤 C2。

步骤 C2，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点对应的索引项，将所述目标节点从所述父节点下删除。

30

其中，如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该父节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

35

步骤 C3，所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，在

成功为所述邻居节点对应的内存页面加排它锁后，通过修改所述邻居节点中的索引项将所述目标节点从所述多叉树中删除，并在成功为所述目标节点加排它锁后，释放所述目标节点对应的内存页面。

其中，如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点（或邻居节点）对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

可选的，虽然父节点到其子节点（即目标节点和邻居节点）的顺序（即从上至下）满足加锁方向（即自上而下）的要求，但是为了能够尽早的释放父节点的排它锁，提高并发性能，所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁之前还可以包括：所述写线程为所述父节点对应的内存页面设置所述预设标记，并释放所述父节点对应内存页面的排它锁。

示例性的，首先，如图 7A 所示，写线程可以持有目标节点 71 及其左侧的邻居节点 72 和右侧的邻居节点 73 对应内存页面的排它锁 X；然后，如图 7B 所示，写线程可以先在释放目标节点 71、邻居节点 72 和邻居节点 73 对应内存页面的排它锁的同时为目标节点 71、邻居节点 72 和邻居节点 73 对应的内存页面加读写锁 SX，再采用非常规模式申请为目标节点的父节点 74 对应的内存页面加排它锁 X，并在成功为父节点 74 对应的内存页面加排它锁后，通过删除父节点 74 中与目标节点 71 对应的索引项，将目标节点 71 从父节点 74 下删除；之后，如图 7C 所示，写线程可以先在释放父节点 74 对应内存页面的排它锁的同时为父节点 74 对应的内存页面加读写锁 SX，再采用非常规模式申请为目标节点 71、邻居节点 72 和邻居节点 73 对应的内存页面加排它锁 X，并在成功为邻居节点 72 和邻居节点 73 对应的内存页面加排它锁后，通过修改邻居节点 72 和邻居节点 73 中的索引项（每层节点通过双向链表串起来的情况下），将目标节点 71 多叉树中删除；之后，如图 7D 所示，写线程可以在成功为目标节点 71 加排它锁后，释放目标节点 71 对应的内存页面。

需要说明的是，图 7A 至图 7D 中，一个大矩形框表示一个节点，一个小矩形框表示节点中的一个索引项，且深灰色的小矩形框表示删除操作需要删除的索引项，并且为了便于理解，一节点对应内存页面的锁类型是标记在该节点附近。

对于 compress，如果删除之后目标节点中还剩余索引项，需要与其他节点合并，可以分为与父节点合并还是与邻居节点合并。其中，与父节点合并可以称为 lift up，与邻居节点合并可以称为 merge。

示例性的，对于 lift up，如果发现目标节点所在层上只有一个节点，就需要与父节点合并。需要说明的是，如果目标节点的父节点所在的层上只有一个节点，且目标节点的父节点中也只有一条索引项，则还可以将目标节点的父节点与目标节点的父节点的父节点合并，依照这样的合并方式最终可以合并到多叉树的根节点。示例性的，假设目标节点为节点 a，目标节点的父节点为节点 b，目标节点的父节点的父节点为节点 c，则在申请为节点 b 对应的内存页面加排它锁之前，可以先为节点 a 设置预设标记，并释放节点 a 对应内存页面的排它锁；在申请为节点 c 对应的内存页面加排它锁之前，可以先为节点 b 设置预设标记，并释放节点 b 对应内存页面的排它锁。

示例性的，对于 merge，可以检查邻居节点是否能容纳目标节点中剩余的索引项，可以先尝试与左侧的邻居节点合并，如果失败，再尝试与右侧的邻居节点合并。其中，与右侧的邻居节点合并可以称为右合并（right merge），与左侧的邻居节点合并可以称为左合并（left merge）。

- 5 一个实施例中，需要将目标节点中剩余的索引项合并至目标节点左侧的邻居节点的情况下，步骤 42 具体可以包括如下步骤 D1-步骤 D4。

步骤 D1，所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁。

- 10 其中，如果目标节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

- 15 如果邻居节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的邻居节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该邻居节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

写线程在成功为目标节点对应的内存页面加排它锁后，可以执行步骤 D2。

写线程在成功为目标节点及其邻居节点对应的内存页面加排它锁后，可以执行步骤 D3。

步骤 D2，在成功为所述目标节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的所述目标索引项删除。

- 20 步骤 D3，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点对应的索引项，将所述目标节点从所述父节点下删除。

- 25 其中，如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该父节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

- 30 步骤 D4，所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点左侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

- 35 其中，如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点（或邻居节点）对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

可选的，虽然父节点到其子节点（即目标节点和邻居节点）的顺序（即从上至下）满

足加锁方向（即自上而下）的要求，但是为了能够尽早的释放父节点的排它锁，提高并发性能，所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁之前还可以包括：所述写线程为所述父节点对应的内存页面设置所述预设标记，并释放所述父节点对应内存页面的排它锁。

- 5 示例性的，首先，如图 8A 所示，写线程可以持有目标节点 81 及其左侧的邻居节点 82 和右侧的邻居节点 83 对应内存页面的排它锁 X；然后，如图 8B 所示，写线程可以先在释放目标节点 81、邻居节点 82 和邻居节点 83 对应内存页面的排它锁的同时为目标节点 81、邻居节点 82 和邻居节点 83 对应的内存页面加读写锁 SX，再采用非常规模式申请为目标节点的父节点 84 对应的内存页面加排它锁 X，并在成功为父节点 84 对应的内存页面加排它锁后，通过删除父节点 84 中与目标节点 81 对应的索引项，将目标节点 81 从父节点 84 下删除；之后，如图 8C 所示，写线程可以先在释放父节点 84 对应内存页面的排它锁的同时为父节点 84 对应的内存页面加读写锁 SX，再采用非常规模式申请为目标节点 81、邻居节点 82 和邻居节点 83 对应的内存页面加排它锁 X，并在成功为目标节点 81、邻居节点 82 和邻居节点 83 对应的内存页面加排它锁后，将目标节点 81 中剩余的索引项移动至邻居节点 82 中，并通过修改邻居节点 82 和邻居节点 83 中的索引项（每层节点通过双向链表串起来的情况下），将目标节点 81 从二叉树中删除；之后，如图 8D 所示，写线程可以释放目标节点 81 对应的内存页面。

- 20 需要说明的是，图 8A 至图 8D 中，一个大矩形框表示一个节点，一个小矩形框表示节点中的一个索引项，并且为了便于理解，一节点对应内存页面的锁类型是标记在该节点附近。

另一个实施例中，需要将目标节点中剩余的索引项合并至目标节点右侧的邻居节点的情况下，步骤 42 具体可以包括如下步骤 E1-步骤 E4。

步骤 E1，所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁。

- 25 其中，如果目标节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

- 30 如果邻居节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的邻居节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该邻居节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

写线程在成功为目标节点对应的内存页面加排它锁后，可以执行步骤 E2。

写线程在成功为目标节点及其邻居节点对应的内存页面加排它锁后，可以执行步骤 E3。

步骤 E2，所述写线程将所述目标节点中的所述目标索引项删除。

- 35 步骤 E3，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点右侧的邻居节点对应的索引项，将所述目标节点右侧的邻居节点从所述父节点下删除，并通过将所述父节点中原本指向所述目标节点的索引

项修改为指向所述目标节点右侧的邻居节点，将所述目标节点右侧的邻居节点插入到所述父节点之下。

其中，如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点的父节点对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待该父节点对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

步骤E4，所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点右侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

其中，如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记，写线程释放其持有的锁。进一步的，本实施例提供的方法还可以包括：如果目标节点（或邻居节点）对应的内存页面已被其他写线程设置预设标记触发写线程释放其持有的锁，则写线程可以等待目标节点（或邻居节点）对应的内存页面的预设标记被清除后，重新从根节点开始遍历，查找目标索引项。

可选的，虽然父节点到其子节点（即目标节点和邻居节点）的顺序（即从上至下）满足加锁方向（即自上而下）的要求，但是为了能够尽早的释放父节点的排它锁，提高并发性能，所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁之前还可以包括：所述写线程为所述父节点对应的内存页面设置所述预设标记，并释放所述父节点对应内存页面的排它锁。

示例性的，首先，如图9A所示，写线程可以持有目标节点91及其左侧的邻居节点92和右侧的邻居节点93对应内存页面的排它锁X；然后，如图9B所示，写线程可以先在释放目标节点91、邻居节点92和邻居节点93对应内存页面的排它锁的同时为目标节点91、邻居节点92和邻居节点93对应的内存页面加读写锁SX，再采用非常规模式申请为目标节点的父节点94对应的内存页面加排它锁X，并在成功为父节点94对应的内存页面加排它锁后，通过删除父节点94中与邻居节点93对应的索引项，将所述邻居节点93从父节点94下删除；之后，如图9C所示，写线程可以通过将父节点94中原本指向目标节点91的索引项修改为指向邻居节点93，将邻居节点93插入到父节点94之下；之后，如图9D所示，写线程可以先在释放父节点94对应内存页面的排它锁的同时为父节点94对应的内存页面加读写锁SX，再采用非常规模式申请为目标节点91、邻居节点92和邻居节点93对应的内存页面加排它锁X，并在成功为目标节点91、邻居节点92和邻居节点93对应的内存页面加排它锁后，将目标节点91中剩余的索引项移动至邻居节点93中，通过修改邻居节点92和邻居节点93中的索引项（每层节点通过双向链表串起来的情况下）将目标节点91从多叉树中删除，并释放目标节点91对应的内存页面。

需要说明的是，图9A至9D中，一个大矩形框表示一个节点，一个小矩形框表示节点中的一个索引项，且深灰色的小矩形框表示被修改的索引项，并且为了便于理解，一节点对应内存页面的锁类型是标记在该节点附近。

本申请实施例中，读操作可以兼容预设标记，即内存页面上的预设标记可以与其它线

程获取共享锁不冲突，让读操作能继续执行。可选的，对于处于 SMO 中间状态的节点，写线程可以为其设置一个指向其的新邻居节点的指针，从而其它读操作访问到处于 SMO 中间状态的节点，可以在其新邻居节点中继续查找，解决了结构修改操作完成前多叉树的结构不完整的问题。例如，右分裂过程中，旧节点 (old node) 中可以设置有指向新节点 (new node) 的指针，相当于成为新节点的 foster parent，读操作可以由旧节点遍历到新节点。在每层节点是通过双向链表 (level list) 串起来的情况下，可以复用节点的左右指针。

基于此，一个实施例中，本实施例提供的方法还可以包括：针对被自身设置所述预设标记的任一节点，如果所述任一节点的邻居节点需要发生变化，在所述任一节点中设置指向所述任一节点的新邻居节点的指针。在此情况下，示例性的，如果读操作进入到处于 SMO 中间状态的内存页面后，可以根据搜索关键字与该内存页面索引项的最大 key 比较的结果，决定是否要继续查找其右侧的邻居节点。

与写线程类似，对于读线程，在需要由任一节点遍历到所述任一节点的后一节点时，读线程可以申请为所述后一节点对应的内存页面加共享锁，并在成功为所述后一节点对应的内存页面加共享锁后，释放所述任一节点对应内存页面的共享锁。

在本申请实施例中，写线程不是申请为某个多叉树整体加读写锁，而是按照加锁规则申请为目标节点及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为目标节点和其他节点加排它锁后，对目标节点和其他节点进行相应修改，目标节点是指查找到的与搜索关键字匹配的索引项所在的节点，实现了单个写线程可以通过仅对多叉树中其需要修改的节点对应的内存页面加排它锁来进行多叉树结构的修改，使得多个写线程之间的冲突是在页面级别，从而使得针对同一多叉树的结构修改操作能够并发，另外，通过加锁规则包括如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有任意节点对应内存页面的排它锁的基础上，申请为关联节点对应的内存页面加排它锁之前，需要释放该任意节点对应内存页面的排他锁，并为该任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置预设标记，则需要释放自身持有的锁，使得按照加锁规则申请加锁，能够避免出现线程间死锁的问题。

图 10 为本申请一实施例提供的数据库操作装置的结构示意图；参考附图 10 所示，本实施例提供了一种数据库操作装置，该装置可以执行上述方法实施例提供的数据库操作方法，具体的，该装置可以包括：

查找模块 101，用于所述写线程从所述多叉树的根节点开始遍历，查找与所述写操作的搜索关键字匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

修改模块 102，用于如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

可选的，所述加锁方向包括沿着所述多叉树自上而下、从左到右的方向。

可选的, 所述需要释放所述任意节点对应内存页面的排他锁并为所述任意节点对应的内存页面设置所述预设标记, 包括: 需要在释放所述任意节点对应内存页面的排它锁的同时为所述任意节点对应的内存页面加读写锁。

可选的, 所述写操作为插入操作, 且需要在所述目标节点的右侧增加节点; 修改模块 5 102 具体用于:

所述写线程申请为所述目标节点所述目标节点及其右侧的邻居节点对应的内存页面加排它锁, 并申请新内存页面作为新节点, 所述写线程持有所述新内存页面的排它锁;

在成功为所述目标节点以及所述邻居节点对应的内存页面加排它锁后, 所述写线程将所述目标节点中的部分索引项移动到所述新节点中, 向所述新节点中插入所述插入操作需要插入的索引项, 并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项, 将所述新节点插入到所述目标节点与所述邻居节点之间;

所述写线程先为所述目标节点、所述新节点和所述邻居节点对应的内存页面设置所述预设标记, 并释放所述目标节点、所述新节点和所述邻居节点对应内存页面的排它锁, 再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁, 并在成功为所述父节点对应的内存页面加排它锁后, 所述写线程通过在所述父节点中增加与所述新节点对应的索引项, 将所述新节点插入到所述父节点下。

可选的, 所述写操作为插入操作, 且需要在所述目标节点的左侧增加节点; 修改模块 102 具体用于:

所述写线程申请为所述目标节点及其左侧的邻居节点对应的内存页面加排它锁, 并申请新内存页面作为新节点, 所述写线程持有所述新内存页面的排它锁;

在成功为所述目标节点以及所述邻居节点对应的内存页面加排它锁后, 所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记, 并释放所述目标节点和所述邻居节点对应内存页面的排它锁, 再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁, 并在成功为所述父节点对应的内存页面加排它锁后, 通过 25 将所述父节点中原本指向所述目标节点的索引项修改为指向所述新节点, 将所述新节点插入到所述父节点下;

所述写线程采用所述非常规模式申请为所述目标节点和所述邻居节点对应的内存页面加排他锁, 并成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后, 将所述目标节点中的部分索引项移动到所述新节点中, 向所述新节点中插入所述插入操作需要插入的索引项, 并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项, 将所述新节点插入到所述目标节点与所述邻居节点之间;

所述写线程通过在所述父节点中增加与所述目标节点对应的索引项, 将所述目标节点插入到所述父节点下。

可选的, 修改模块 102 还用于所述写线程采用所述非常规模式申请为所述目标节点和所述邻居节点对应的内存页面加排他锁之前, 所述写线程为所述父节点对应的内存页面设置所述预设标记, 并释放所述父节点对应内存页面的排它锁;

修改模块 102 用于所述写线程通过在所述父节点中增加与所述目标节点对应的索引项, 将所述目标节点插入到所述父节点下之前, 还用于: 所述写线程为所述目标节点、所述新节点和所述邻居节点对应的内存页面设置所述预设标记, 并释放所述目标节点、所述新节

点和所述邻居节点对应内存页面的排它锁；所述写线程采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁。

可选的，所述写操作为删除操作，且需要删除所述目标节点；修改模块 102 具体用于：所述写线程申请对所述目标节点及其邻居节点对应的内存页面加排它锁；

5 在成功为所述目标节点以及所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点对应的索引项，将所述目标节点从所述父节点下删除；

10 所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁，在成功为所述邻居节点对应的内存页面加排它锁后，通过修改所述邻居节点中的索引项将所述目标节点从所述多叉树中删除，并在成功为所述目标节点加排它锁后，释放所述目标节点对应的内存页面。

15 可选的，所述写操作为删除操作，且需要将所述目标节点中剩余的索引项合并至所述目标节点左侧的邻居节点；修改模块 102 具体用于：

所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁；

在成功为所述目标节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的所述目标索引项删除；

20 在成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再采用所述非常规模式申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点对应的索引项，将所述目标节点从所述父节点下删除；

25 所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点左侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

30 可选的，所述写操作为删除操作，且需要将所述目标节点中剩余的索引项合并至所述目标节点右侧的邻居节点；修改模块 102 具体用于：

所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁；

在成功为所述目标节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的所述目标索引项删除；

35 在成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点右侧的邻居节点对应的索引项，将所述目标节点右侧的邻居节点从所述父节点下删除，并通过将所述父节点中原本指向所述目标节点的索引项修改为指向所述目标节点

右侧的邻居节点，将所述目标节点右侧的邻居节点插入到所述父节点之下；

所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点右侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

进一步可选的，修改模块 102 用于所述写线程采用所述非常规模式申请为所述目标节点及其邻居节点对应的内存页面加排它锁之前，还用于所述写线程为所述父节点对应的内存页面设置所述预设标记，并释放所述父节点对应内存页面的排它锁。

可选的，查找模块 101 还用于在需要由任一节点遍历到所述任一节点的后一节点时，所述写线程申请为所述后一节点对应的内存页面加共享锁，并在成功为所述后一节点对应的内存页面加共享锁后，释放所述任一节点对应内存页面的共享锁。

可选的，修改模块 102 还用于针对被自身设置所述预设标记的任一节点，如果所述任一节点的邻居节点需要发生变化，在所述任一节点中设置指向所述任一节点的新邻居节点的指针。

可选的，查找模块 101，还用于如果某一内存页面已被其他写线程设置所述预设标记触发所述写线程释放其持有的锁，则所述写线程等待所述某一内存页面的所述预设标记被清除后，重新从所述根节点开始遍历，查找所述目标索引项。

图 10 所示装置可以执行图 4 所示实施例的方法，本实施例未详细描述的部分，可参考对图 4 所示实施例的相关说明。该技术方案在执行过程和技术效果参见图 4 所示实施例中的描述，在此不再赘述。

在一个可能的实现中，图 10 所示装置的结构可实现为一计算机设备。如图 11 所示，该计算机设备可以包括：处理器 111 和存储器 112。其中，存储器 112 用于存储支持计算机设备执行上述图 4 所示实施例中提供的方法的程序，处理器 111 被配置为用于执行存储器 112 中存储的程序。

程序包括一条或多条计算机指令，其中，一条或多条计算机指令被处理器 111 执行时能够实现如下步骤：

写线程从所述多叉树的根节点开始遍历，查找与写操作的搜索关键字匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

可选的，处理器 111 还用于执行前述图 4 所示实施例中的全部或部分步骤。

其中，计算机设备的结构中还可以包括通信接口 113，用于计算机设备与其他设备或

通信网络通信。

另外，本申请实施例提供了一种计算机可读存储介质，其上存储有计算机程序，当所述计算机程序被执行时，实现如图4所示实施例所述的方法。

5 以上所描述的装置实施例仅仅是示意性的，其中所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。本领域普通技术人员在不付出创造性的劳动情况下，即可以理解并实施。

10 通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到各实施方式可借助加必需的通用硬件平台的方式来实现，当然也可以通过硬件和软件结合的方式来实现。基于这样的理解，上述技术方案本质上或者说对现有技术做出贡献的部分可以以计算机产品的形式体现出来，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器、CD-ROM、光学存储器等）上实施的计算机程序产品的形式。

15 本申请是参照根据本申请实施例的方法、设备（系统）、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程设备的处理器以产生一个机器，使得通过计算机或其他可编程设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

20 这些计算机程序指令也可存储在能引导计算机或其他可编程设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

25 这些计算机程序指令也可装载到计算机或其他可编程设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

30 在一个典型的配置中，计算设备包括一个或多个处理器（CPU）、输入/输出接口、网络接口和内存。

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。

35 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、链表、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，磁带磁磁盘存储或其

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。

- 最后应说明的是：以上各实施例仅用以说明本申请的技术方案，而非对其限制；尽管
- 5 参照前述各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应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分或者全部技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本申请各实施例技术方案的范围。

权 利 要 求 书

1、一种数据库操作方法，其特征在于，所述数据库使用多叉树作为索引，所述多叉树的节点与内存页面一一对应，所述方法包括：

5 写线程从所述多叉树的根节点开始遍历，查找与写操作的搜索关键字匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

10 如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

15 2、根据权利要求1所述的方法，其特征在于，所述加锁方向包括沿着所述多叉树自上而下、从左到右的方向。

3、根据权利要求1所述的方法，其特征在于，所述需要释放所述任意节点对应内存页面的排他锁并为所述任意节点对应的内存页面设置所述预设标记，包括：需要在释放所述任意节点对应内存页面的排它锁的同时为所述任意节点对应的内存页面加读写锁。

20 4、根据权利要求1-3任一项所述的方法，其特征在于，所述写操作为插入操作，且需要在所述目标节点的右侧增加节点；所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改，包括：

25 所述写线程申请为所述目标节点及其右侧的邻居节点对应的内存页面加排它锁，并申请新内存页面作为新节点，所述写线程持有所述新内存页面的排它锁；

30 在成功为所述目标节点以及所述邻居节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的部分索引项移动到所述新节点中，向所述新节点中插入所述插入操作需要插入的索引项，并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项，将所述新节点插入到所述目标节点与所述邻居节点之间；

30 所述写线程先为所述目标节点、所述新节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点、所述新节点和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，所述写线程通过在所述父节点中增加与所述新节点对应的索引项，将所述新节点插入到所述父节点下。

35 5、根据权利要求1-3任一项所述的方法，其特征在于，所述写操作为插入操作，且需要在所述目标节点的左侧增加节点；所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改，包括：

所述写线程申请为所述目标节点及其左侧的邻居节点对应的内存页面加排它锁，并申请新内存页面作为新节点，所述写线程持有所述新内存页面的排它锁；

在成功为所述目标节点以及邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过将所述父节点中原本指向  
5 所述目标节点的索引项修改为指向所述新节点，将所述新节点插入到所述父节点下；

所述写线程申请为所述目标节点和所述邻居节点对应的内存页面加排他锁，并成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，将所述目标节点中的部分索引项移动到所述新节点中，向所述新节点中插入所述插入操作需要插入的索引项，并通过修改所述目标节点、所述新节点和所述邻居节点中的索引项，将所述新节点插入到所述目标  
10 节点与所述邻居节点之间；

所述写线程通过在所述父节点中增加与所述目标节点对应的索引项，将所述目标节点插入到所述父节点下。

6、根据权利要求 1-3 任一项所述的方法，其特征在于，所述写操作为删除操作，且需要删除所述目标节点；所述写线程按照加锁规则申请为所述目标节点以及需要关联修改  
15 的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改，包括：

所述写线程申请对所述目标节点及其邻居节点对应的内存页面加排它锁；

在成功为所述目标节点以及所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点  
20 和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与  
所述目标节点对应的索引项，将所述目标节点从所述父节点下删除；

所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，在成功为所述邻居节点对应的内存页面加排它锁后，通过修改所述邻居节点中的索引项将所述目标  
25 节点从所述多叉树中删除，并在成功为所述目标节点加排它锁后，释放所述目标节点对应的内存页面。

7、根据权利要求 1-3 任一项所述的方法，其特征在于，所述写操作为删除操作，且需要将所述目标节点中剩余的索引项合并至所述目标节点左侧的邻居节点；所述写线程按照加锁规则申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，  
30 以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改，包括：

所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁；

在成功为所述目标节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的所述目标索引项删除；

35 在成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和  
所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与  
所述目标节点对应的索引项，将所述目标节点从所述父节点下删除；

所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点左侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

- 5 8、根据权利要求 1-3 中任一项所述的方法，其特征在于，所述写操作为删除操作，且需要将所述目标节点中剩余的索引项合并至所述目标节点右侧的邻居节点；所述写线程按照预设加锁策略申请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改，包括：

- 10 所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁；

在成功为所述目标节点对应的内存页面加排它锁后，所述写线程将所述目标节点中的所述目标索引项删除；

- 15 在成功为所述目标节点和所述邻居节点对应的内存页面加排它锁后，所述写线程先为所述目标节点和所述邻居节点对应的内存页面设置所述预设标记，并释放所述目标节点和所述邻居节点对应内存页面的排它锁，再申请为所述目标节点的父节点对应的内存页面加排它锁，并在成功为所述父节点对应的内存页面加排它锁后，通过删除所述父节点中与所述目标节点右侧的邻居节点对应的索引项，将所述目标节点右侧的邻居节点从所述父节点下删除，并通过将所述父节点中原本指向所述目标节点的索引项修改为指向所述目标节点右侧的邻居节点，将所述目标节点右侧的邻居节点插入到所述父节点之下；

- 20 所述写线程申请为所述目标节点及其邻居节点对应的内存页面加排它锁，并在成功为所述目标节点及其邻居节点对应的内存页面加排它锁后，将所述目标节点中剩余的索引项移动至所述目标节点右侧的邻居节点中，通过修改所述目标节点的邻居节点中的索引项，将所述目标节点从所述多叉树中删除，并释放所述目标节点对应的内存页面。

- 25 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：如果某一内存页面已被其他写线程设置所述预设标记触发所述写线程释放其持有的锁，则所述写线程等待所述某一内存页面的所述预设标记被清除后，重新从所述根节点开始遍历，查找所述目标索引项。

- 30 10、根据权利要求 1 所述的方法，其特征在于，所述方法还包括：在需要由任一节点遍历到所述任一节点的后一节点时，所述写线程申请为所述后一节点对应的内存页面加共享锁，并在成功为所述后一节点对应的内存页面加共享锁后，释放所述任一节点对应内存页面的共享锁。

- 11、根据权利要求 1 所述的方法，其特征在于，所述方法还包括：针对被自身设置所述预设标记的任一节点，如果所述任一节点的邻居节点需要发生变化，在所述任一节点中设置指向所述任一节点的新邻居节点的指针。

- 35 12、一种数据库操作装置，其特征在于，所述数据库使用多叉树作为索引，所述多叉树的节点与页面一一对应，所述装置包括：

查找模块，用于写线程从所述多叉树的根节点开始遍历，查找与写操作的搜索关键字匹配的目标索引项，以查找到所述目标索引项所在的目标节点；

修改模块，用于如果所述写操作需要触发结构修改操作，所述写线程按照加锁规则申

5 请为所述目标节点以及需要关联修改的其他节点对应的内存页面加排它锁，以在成功为所述目标节点和所述其他节点加排它锁后，对所述目标节点和所述其他节点进行相应修改；所述加锁规则包括：如果任意节点到关联节点的方向不满足加锁方向的要求，则在持有所述任意节点对应内存页面的排它锁的基础上，申请为所述关联节点对应的内存页面加排它锁之前，需要释放所述任意节点对应内存页面的排他锁，并为所述任意节点对应的内存页面设置预设标记；以及，如果申请加排它锁的任一内存页面已被其他写线程设置所述预设标记，则需要释放自身持有的锁。

10 13、一种计算机设备，其特征在于，包括：存储器、处理器；其中，所述存储器用于存储一条或多条计算机指令，其中，所述一条或多条计算机指令被所述处理器执行时实现如权利要求 1 至 11 中任一项所述的方法。

14、一种计算机可读存储介质，其特征在于，其上存储有计算机程序，当所述计算机程序被执行时，实现如权利要求 1 至 11 任一项所述的方法。