(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104899297 B
(45)授权公告日 2019.02.26

(21)申请号 201510310349.X

(22)申请日 2015.06.08

(65)同一申请的已公布的文献号
申请公布号 CN 104899297 A

(43)申请公布日 2015.09.09

(73)专利权人 南京航空航天大学
地址 210016 江苏省南京市秦淮区御道街29号

(72)发明人 秦小麟 王胜 史文浩 王潇逸 李博涵

(74)专利代理机构 南京经纬专利商标代理有限公司 32200
代理人 熊玉玮

(51)Int.Cl.
G06F 16/22(2019.01)

(56)对比文件
CN 103823865 A,2014.05.28,
CN 101763415 A,2010.06.30,
US 2004/0193632 A1,2004.09.30,
Yinan Li.Tree Indexing on Flash Disks.《IEEE》.2009,1303-1306页.
王胜 等.可持久化CSB+-树索引技术研究.《CNKI》.2014,182-192页.
Yinan Li等.Tree Indexing on Solid State Drives.《36th International Conference on Very Large Data Bases》.2010,第1195-1206页.

审查员 冯雅

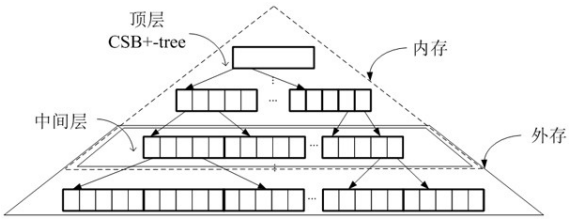
权利要求书1页 说明书7页 附图5页

(54)发明名称

创建具有存储感知的混合索引的方法

(57)摘要

本发明公开了具有存储感知的混合索引结构，属于数据库查询的技术领域。混合索引结构采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构，顶层、中间层、底层数据在内存充足时均存储在内存中，中间层数据在内存不足时局部或全部存储在外存中。本发明还公开了混合索引结构的创建方法、读取方法、范围查询方法、重构方法，通过内存映射长度在内存不足时将部分索引数据存储在外存中，有效地利用内存、外存等存储资源，能够根据系统情况或用户效率需求合理决策索引结构内外存占用比例，提高查询效率。



CN 104899297 B

1.创建具有存储感知的混合索引的方法,其特征在于:

具有存储感知的混合索引,采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构,顶层、中间层、底层数据在内存充足时均存储在内存中,中间层数据在内存不足时局部或全部存储在外存中,

索引头文件创建在映射区域起始段,所述索引头文件包括:指示索引树含有常规键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、记录用于调整内存中存储数据的内存映射长度,索引树每一层节点描述信息包括:起始偏移量、结束偏移量、区域长度、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记;

方法具体如下:

计算内存可用连续空间大小以及索引结构占用空间并初始化分层地址表,

创建位于外存中的初始索引文件以形成索引结构底层,

采用内存映射技术并考虑内存可用连续空间,按照分层地址表将初始索引文件部分映射入内存以形成若干有序索引项节点序列构成的中间层,

在映射区域起始段创建索引头并初始化分层地址表,

在主存中建立索引树并根据分层地址表从上层叶子节点开始逐层向下层叶子节点插入键值以形成顶层,在上层叶子节点溢出时将其与下层叶子节点合并,取出下层叶子节点的第一项作为上层叶子节点的指针项后构造上层叶子节点,迭代此过程直至完成索引树的构造,在顶层占用空间超出阈值时将溢出的节点与中间层进行合并,

同步内外存的索引结构数据。

创建具有存储感知的混合索引的方法

技术领域

[0001] 本发明公开了创建具有存储感知的混合索引的方法，属于数据库查询的技术领域。

背景技术

[0002] 现有数据库索引按照存储方式可分为两种，其一是采用完全基于硬盘的频繁访问方式来进行查询等操作，而未充分利用主存和缓存等处理速度更快的设备，性能无法突破设备瓶颈。另一种则是主存索引技术，即将索引完全建立在主存中，作为一种高效的索引方式，往往利用缓存技术，在现代计算机框架下性能非常高效，但当数据量大到一定程度时，如目前采用异构的计算资源的进行海量数据处理的云计算场景下，内存容量差异较大，如在内存容量一般的主机上，将无法容纳空间占用较大的索引结构。限于内存空间和存储等问题，主存索引不具有传统硬盘索引技术的可重用性和空间不受限等优点；而限于硬盘读写效率问题，传统硬盘索引不具有主存索引的高效性。

[0003] 主存容量相对于硬盘等外存设备往往差距较大，因此当索引结构占用空间较大时，将整个索引结构置于主存中将不实际，根据主存占用情况进行自适应性的将底层调整至硬盘中，结合两种索引技术，动态利用外存、内存，以保证索引结构同时具有空间不受限、高效、持久化等特点。创建于主存中的索引结构的存储问题决定着数据完整性和系统实时性，若未在外存中存储相应的数据结构，系统重启时需重新创建将浪费大量的计算资源，因此将主存中的索引结构存储到硬盘中对于索引的持久性和重用性十分重要。

[0004] FD-tree是一种采用分散层叠技术和对数技术的高效索引结构，最小化随机读写的次数和限制随机读写的区域，有限提高了索引结构更新的效率。但是该索引结构仅将位于索引树顶层的B+-树导入到主存中，在内存能够容纳索引结构时却无法将整个结构导入内存中而未充分利用主存和缓存，因此效率仍有待提高。

发明内容

[0005] 本发明所要解决的技术问题是针对上述背景技术的不足，提供了创建具有存储感知的混合索引的方法，基于FD-tree索引结构进行改进，基于内存空间决定索引树的内外存分布，将存储在硬盘上的索引结构局部或全部导入到主存中，提高查询和更新效率，同时不破坏系统的持久性特征，解决了仅将位于索引树顶层的B+-树导入到主存中的索引结构存在索引效率低的技术问题。

[0006] 本发明为实现上述发明目的采用如下技术方案：

[0007] 创建具有存储感知的混合索引的方法，具有存储感知的混合索引，采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构，顶层、中间层、底层数据在内存充足时均存储在内存中，中间层数据在内存不足时局部或全部存储在外存中，

[0008] 索引头文件创建在映射区域起始段，所述索引头文件包括：指示索引树含有常规

键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、记录用于调整内存中存储数据的内存映射长度，索引树每一层节点描述信息包括：起始偏移量、结束偏移量、区域长度、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记；

[0009] 方法具体如下：

[0010] 计算内存可用连续空间大小以及索引结构占用空间并初始化分层地址表，

[0011] 创建位于外存中的初始索引文件以形成索引结构底层，

[0012] 采用内存映射技术并考虑内存可用连续空间，按照分层地址表将初始索引文件部分映射入内存以形成若干有序索引项节点序列构成的中间层，

[0013] 在映射区域起始段创建索引头并初始化分层地址表，

[0014] 在主存中建立索引树并根据分层地址表从上层叶子节点开始逐层向下层叶子节点插入键值以形成顶层，在上层叶子节点溢出时将其与下层叶子节点合并，取出下层叶子节点的第一项作为上层叶子节点的指针项后构造上层叶子节点，迭代此过程直至完成索引树的构造，在顶层占用空间超出阈值时将溢出的节点与中间层进行合并，

[0015] 同步内外存的索引结构数据。

[0016] 读取具有存储感知的混合索引的方法，

[0017] 具有存储感知的混合索引，采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构，顶层、中间层、底层数据在内存充足时均存储在内存中，中间层数据在内存不足时局部或全部存储在外存中，

[0018] 索引头文件创建在映射区域起始段，所述索引头文件包括：指示索引树含有常规键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、记录用于调整内存中存储数据的内存映射长度，索引树每一层节点描述信息包括：起始偏移量、结束偏移量、区域长度、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记；

[0019] 方法通过溢出检测实现索引树的查询和更新，具体如下：获取索引头中的内存映射长度，在节点偏移量大于内存映射长度溢出时采用文件读写方式读出整个节点块，在节点偏移量小于内存映射长度时通过节点偏移量计算内存地址并按照指针方式直接读写。

[0020] 具有存储感知的混合索引的范围查询方法，

[0021] 具有存储感知的混合索引，采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构，顶层、中间层、底层数据在内存充足时均存储在内存中，中间层数据在内存不足时局部或全部存储在外存中，

[0022] 索引头文件创建在映射区域起始段，所述索引头文件包括：指示索引树含有常规键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、记录用于调整内存中存储数据的内存映射长度，索引树每一层节点描述信息包括：起始偏移量、结束偏移量、区域长度、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记；

[0023] 方法具体如下：

[0024] 在中间层局部脏标记均为非脏时跳至底层进行范围查询，

[0025] 在中间层有局部脏标记为脏时从顶层开始依次检查索引树每一层的局部脏标记：该层所有局部脏标记非脏时跳至下一层，该层有局部脏标记为脏时将其中的常规键值筛选加入到查询结果集中。

[0026] 重构具有存储感知的混合索引的方法，

[0027] 具有存储感知的混合索引，采用由上至下划分为位于主存中的顶层、位于内存中的中间层、位于外存中的底层这三部分的树状索引结构，顶层、中间层、底层数据在内存充足时均存储在内存中，中间层数据在内存不足时局部或全部存储在外存中，

[0028] 索引头文件创建在映射区域起始段，所述索引头文件包括：指示索引树含有常规键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、记录用于调整内存中存储数据的内存映射长度，索引树每一层节点描述信息包括：起始偏移量、结束偏移量、区域长度、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记；

[0029] 方法具体如下：获取缓存索引树全局脏标记以及每一层的局部脏标记，仅在全局脏标记为脏时依次将标记为脏状态的层和标记为脏状态的最近下层合并，直至将所有上层常规键值合并到叶子层为止，根据当前运行环境设置参数并基于叶子层构造上层。

[0030] 本发明采用上述技术方案，具有以下有益效果：有效地利用内存、外存等存储资源，能够根据系统情况或用户效率需求合理决策索引结构内外存占用比例，特别在针对不同的数据负载情况时，能有效解决主存索引的空间占用问题和硬盘索引的效率折中问题，实现索引结构在效率和空间占用上的有效均衡与双赢，对拥有异构的计算资源并进行海量数据处理的云计算场景，具有较好的实用性和通用性。

附图说明

[0031] 图1是本发明的原理示意图。

[0032] 图2是索引头示意图。

[0033] 图3是索引创建操作示意图。

[0034] 图4是范围查询操作示意图。

[0035] 图5是索引合并操作示意图。

[0036] 图6是索引重构操作示意图。

具体实施方式

[0037] 下面详细描述本发明的实施方式，下面通过参考附图描述的实施方式是示例性的，仅用于解释本发明，而不能解释为对本发明的限制。

[0038] 本领域的技术人员可以理解，除非另外定义，这里使用的所有术语(包括技术术语和科学术语)具有本发明所属技术领域中的普通技术人员的一般理解相同的意义。还应该理解的是，诸如通用字典中定义的那些术语应该被理解为具有与现有技术的上下文中的意义一致的意义，并且除非像这里一样定义，不会用理想化或过于正式的含义来解释。

[0039] 本发明涉及的是一种具有存储感知的自调整混合索引，为实现索引树的持久化特征，采用内存映射技术将索引结构高效地保存到外存中。如图1所示采用树状索引结构，上下划分为三部分，分别为位于主存的缓存敏感树(CSB+-树)，称为顶层，同样位于内存的中间层和位于外存中的底层，中间层和底层均由若干层有序索引项节点序列组成。当内存充足时三部分全部置于内存中，内存不足时中间层可局部位于外存中，主要通过对内存映射长度的设置进行适应性变化。索引项分为四种类型，分为常规项(键值+类型+行标识符)、待删项(键值+类型)、内部指针项(键值+类型+本层指针)、外部指针项(键值+类型+下层指针)，采用统一的数据结构存储，基于类型进行区分。

[0040] 索引头文件创建在映射区域(索引树内存中的区域)起始段,索引头文件如图2所示包括:指示索引树含有常规键值与否的全局脏标记、记录索引树每一层节点描述信息的分层地址表、调整内存中存储数据的内存映射长度,索引树每一层节点描述信息包括:起始偏移量、结束偏移量、区域长度(结束偏移量与起始偏移量之差)、头节点偏移量、导入比例、指示该层节点含有常规键值与否的局部脏标记。

[0041] 进行索引创建前,先获取系统当前可用连续内存空间,同时根据键值数计算索引空间占用、节点大小、相邻层次比。然后创建索引文件并根据分层地址表将其映射入主存中,初始化分层地址表并开始创建索引树,可最大程度地利用内存空间同时保证最高的查询效率。

[0042] 为解决主存索引空间占用问题,特别应对海量数据索引场景时,宝贵的内存空间将不足以容纳索引树,采用基于存储感知的适应性分层方法对索引树进行创建,将能够根据内存空间占用,动态地将部分索引结构置于外存中,利用基于溢出检测的节点读写方法,保证了索引树的普适性同时最大化效率,为对索引树进行高效管理,利用分层地址表对索引树的每一层的信息进行高效存储。

[0043] 对于不同的设备环境,相同的索引结构将无法满足最优性能,为能够对索引进行自调整以满足多样性环境需求,本发明采用基于跨层合并的索引重构算法,实现将键值全部集中到叶子层中并对节点大小进行调整。

[0044] 针对范围查询需从顶层开始扫描至叶子层浪费大量资源的问题,本发明应用一种基于标记的范围查询算法,能有效降低读写开销。

[0045] 实施例一:索引创建及重构过程,具体步骤流程如图3、图6所示,包括:

[0046] 1、索引创建

[0047] A.获取内存可占用的连续空间大小,根据键值数计算索引结构空间占用,初始化分层地址表:

[0048] 首先计算完全采用顶层树的空间占用,若超过阈值(根据可内存占用计算*50%),则计算B+-树的最适宜层数的占用空间,其余层采用有序索引项序列层,分别计算层数和分层地址表;

[0049] B.创建相应大小的初始索引文件:

[0050] 索引文件即存储整个索引结构的文件,位于外存中,索引文件的目的在于当系统断电后,内存数据丢失,重启后能够基于索引文件快速启动进行查询;

[0051] C.采用内存映射技术按照分层地址表将索引文件部分映射入内存;

[0052] D.在映射区域起始段创建索引头并初始化;

[0053] E.将键值逐步插入位于内存中的上层B+-树,若溢出则将其与下层合并:

[0054] 该过程将全部在主存中进行,当顶层占用空间超出阈值,则将其与下层进行合并,合并过程是对两个有序链表进行合并,合并过程中下层节点全部预留30%空间以供后期合并以减少不必要的写入开销;

[0055] F.合并完成后重构上层直至顶层:

[0056] 重构过程中,取出下层有序链表的起始键值即代表键,初始化为指向该节点的指针项,并按序构造上层序列,迭代此过程直至完成B+-树的构造;

[0057] G.进行内外存索引结构数据同步:

[0058] 数据同步主要采用内存映射提供刷新方法，即对映射区域进行刷新，相对于文件读写的方法，该方法能在短时间内完成对一个较大文件的刷新，效率非常高。

[0059] 2、索引重构

[0060] A.读取索引树的脏标记，若为否，直接跳至F；否则跳至B；

[0061] 脏标记位于索引头中，指示着该层中除了指针项是否存在常规键值，若显示为脏状态，则说明除叶子层外某一层中存在常规键值。否则只有叶子层中存在常规键，可直接重构；

[0062] B.获取顶层的根节点并到达叶子层；

[0063] C.获取索引头中当前层的脏标记；

[0064] D.若为真，则与标记为脏状态的最近下层进行合并，跳至该层并跳至F：

[0065] 该步骤的目的在于将上层的常规键值合并至同样存在常规键值的层中，最终将索引常规键合并至叶子层，优点是避免扫描只含有指针项的层，节约大量时间；

[0066] E.若为否，跳至下一层；

[0067] F.若未到达叶子层则跳至C；否则跳至G；

[0068] G.基于运行环境设置新的节点大小和层比例，基于叶子层构建上层：

[0069] 运行环境主要包括查询负载和内外存环境，通过设置不同的节点大小和层间比进行适应性自调整。

[0070] 实施例二：对索引的插入、删除、点查询、范围查询和修改等五项操作，具体如下所示：

[0071] 1.点查询操作

[0072] A.在顶层中进行查找，找到相应叶节点；

[0073] B.在该节点内进行键值比较；

[0074] C.若找到相应键值，判断是否为索引项，若是则返回相应的行标识符；

[0075] D.若为待删项，则该键值不存在，查询失败，返回；否则获取相应的子节点偏移量；

[0076] E.若未找到相应键值，则找到小于其的最大指针项并获取子节点偏移量；

[0077] F.根据基于溢出检测的节点读写方法访问该子节点；

[0078] G.重复B-E步骤，直至到达底层节点；

[0079] H.若节点中存在该键值，则返回对应的行标识符；

[0080] I.否则查询失败。

[0081] 2.范围查询如图4所示

[0082] A.获取顶层的根节点并到达叶子层；

[0083] B.获得该层对应的起始节点和结束节点地址：

[0084] 范围查询拥有给定的范围，根据范围分别进行点查询找到该层中对应的子节点；

[0085] C.从索引头中获取该层的脏标记；

[0086] D.若标记为真，则在起始节点和结束节点之间搜索键值并加入到结果集中：

[0087] 加入到结果集时，若出现冲突，即表示为待删项，将丢弃该键值并删除结果集中原有的键值；

[0088] E.否则更新起始节点和结束节点；

[0089] F.若起始节点和结束节点均不为空则跳至B步骤；

[0090] G.否则停止并返回结果集：

[0091] 当起始节点和结束节点均为空时，表示已遍历完叶子层，无子节点，便将其赋为空以预示查询结束。

[0092] 3.插入操作

[0093] A.将键值初始化为常规索引项插入至B+-树中；

[0094] B.若B+-树溢出则执行合并操作；

[0095] C.否则插入成功；

[0096] 4.合并操作如图5所示

[0097] A.根据分层地址表分别获取待合并上下层头节点地址；

[0098] 待合并层分别为Li层和Li+1层，分别读取索引头中的分层地址表，获取头节点偏移量，即为头节点位于索引文件中的相对位置。

[0099] B.从头开始选取上层的节点，并获取其在下层的子节点：

[0100] 遍历节点内所有键值，判断类型为外部索引项，获取其键值部分，即为指向子节点的偏移量；

[0101] C.将上层中的常规项或待删项与下层的子节点进行合并：

[0102] 若在子节点中出现待删键，则在上下层中直接删除该项；若为常规项，则按序合并，由于节点中通常保留部分空余空间，所以将不会出现新的节点空间申请，当节点溢出时将申请新节点，重新分配子节点间的键值数以达到平衡；

[0103] D.重新构造Li层及以上层：

[0104] 待合并完成后即不再出现溢出时，选择下层节点的第一项作为新的上层指针项，不断迭代直至完成B+-树的构造。

[0105] 5.删除操作

[0106] A.在顶层B+-树中先实施删除操作：

[0107] 由于B+-树中可能存在后期插入常规项键值，所以先在顶层进行查询，若未检索到相关项，则查询下层节点；若查询到常规项，而未合并到下层中，则采用传统的B+-树删除操作即可；

[0108] B.若待删键位于下层，则插入标记为待删项的键值至B+-树中：

[0109] 当该键处于下层时，则采用懒惰删除方法，并非真正删除，而是插入类型为待删键到B+-树中，进行查询时，根据类型能较快判断为待删键值，而不用遍历到更下层中查找；

[0110] C.等待合并操作以将待删项完全清除：

[0111] 为减少I/O开销，因此在合并过程中统一将待删项去除。具体方法是合并过程中若出现两个相同的键值，其中一项为常规项，另一项为待删项，则直接将两项删除，此方法避免频繁的节点变化，而将更新操作集中到一个小的区域。

[0112] 综上所述，本发明根据内外存信息进行适应性调整索引结构内外存占用比例，进一步利用主存索引技术提高查询效率，同时将整个索引结构持久化到硬盘中，当内存无空间可容纳索引结构时，仍能基于硬盘上的索引文件进行查询，使其具有可伸缩性和拓展性；当内存空间充足时，直接将整索引树导入内存中，作为主存索引，具有高效的查询效率，当可用连续内存空间不足时，局部导入内存进行查询。根据系统占用情况，本发明不仅可以作为主存索引方案，同时也可以作为传统的基于硬盘的索引方案，调整导入内外存的分布比

例,以解决内存占用和性能的均衡问题。

[0113] 本领域普通技术人员可以理解:附图只是一个实施例的示意图,附图中的模块或流程并不一定是实施本发明所必须的。

[0114] 通过以上的实施方式的描述可知,本领域的技术人员可以清楚地了解到本发明可借助软件加必需的通用硬件平台的方式来实现。基于这样的理解,本发明的技术方案实质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来,该计算机软件产品可以存储在存储介质中,如ROM/RAM、磁碟、光盘等,包括若干指令用以使得一台计算机设备(可以是个人计算机、服务器,或者网络设备等)执行本发明的实施例或实施例的某些部分所述的方法。

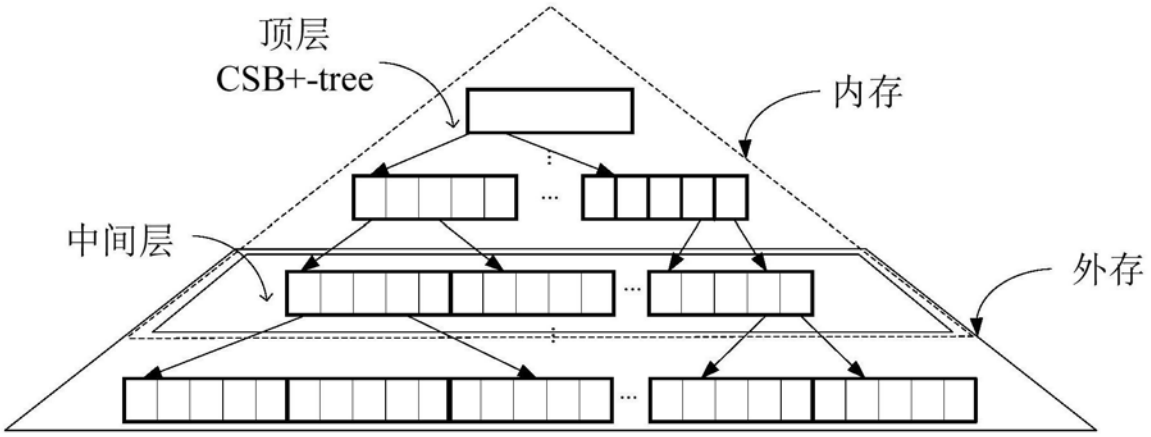

顶层
CSB+-tree
内存
中间层
外存

图1

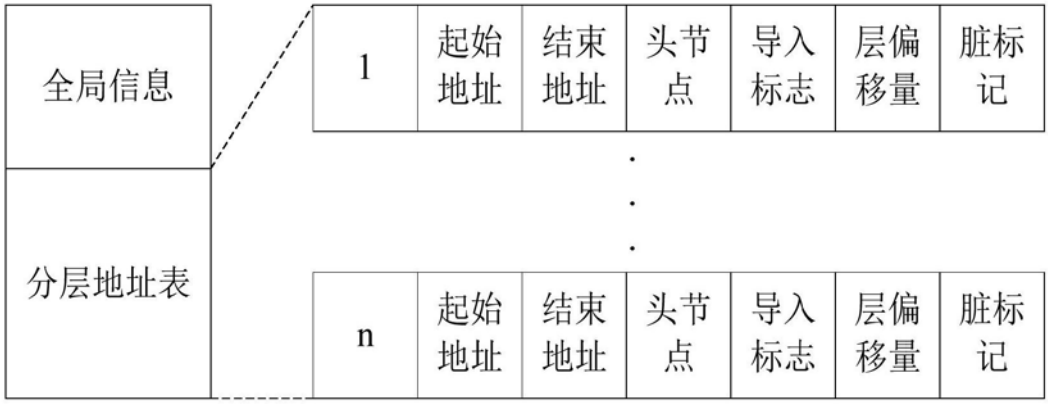

全局信息
分层地址表
1
起始地址
结束地址
头节点
导入标志
层偏移量
脏标记
n
起始地址
结束地址
头节点
导入标志
层偏移量
脏标记

图2

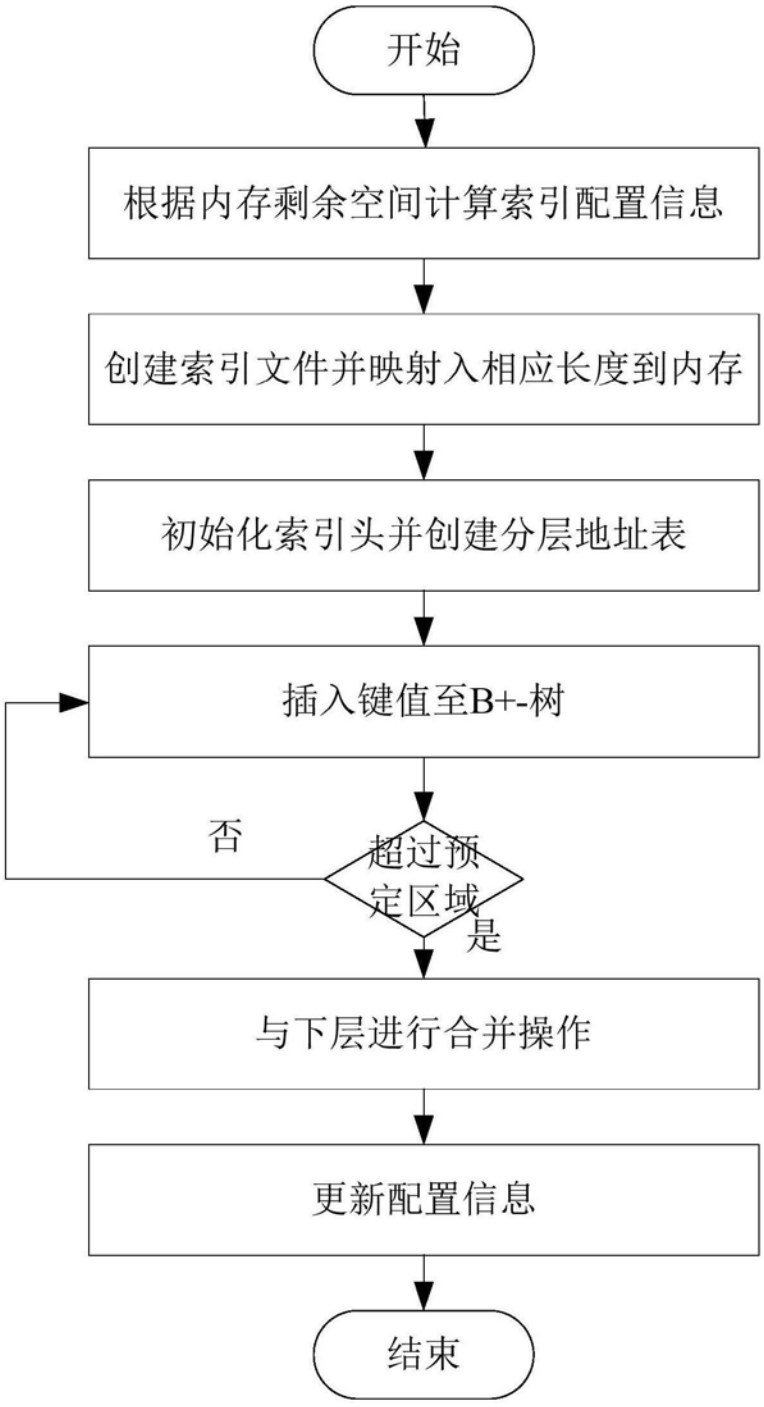

开始
根据内存剩余空间计算索引配置信息
创建索引文件并映射入相应长度到内存
初始化索引头并创建分层地址表
插入键值至B+-树
否
超过预定区域
是
与下层进行合并操作
更新配置信息
结束

图3

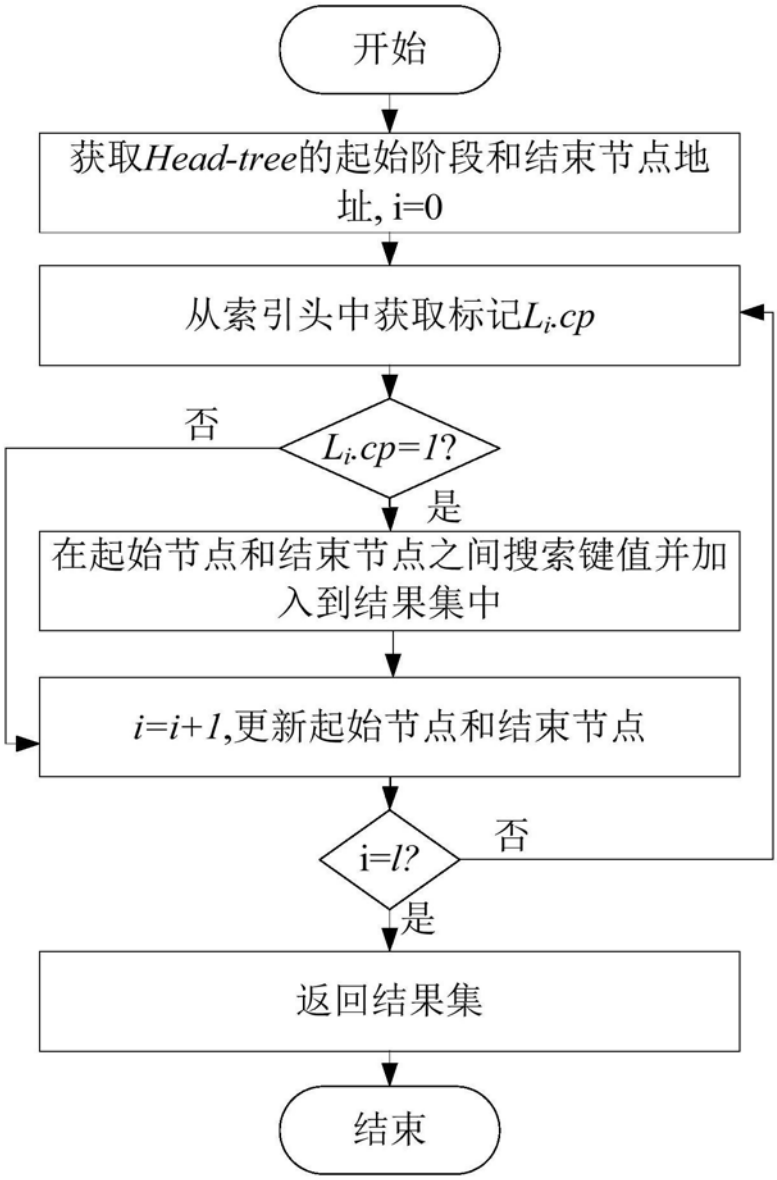

开始
获取Head-tree的起始阶段和结束节点地址, i=0
从索引头中获取标记$L_i.cp$
否
$L_i.cp=1$?
是
在起始节点和结束节点之间搜索键值并加入到结果集中
$i=i+1$,更新起始节点和结束节点
i=l?
否
是
返回结果集
结束

图4

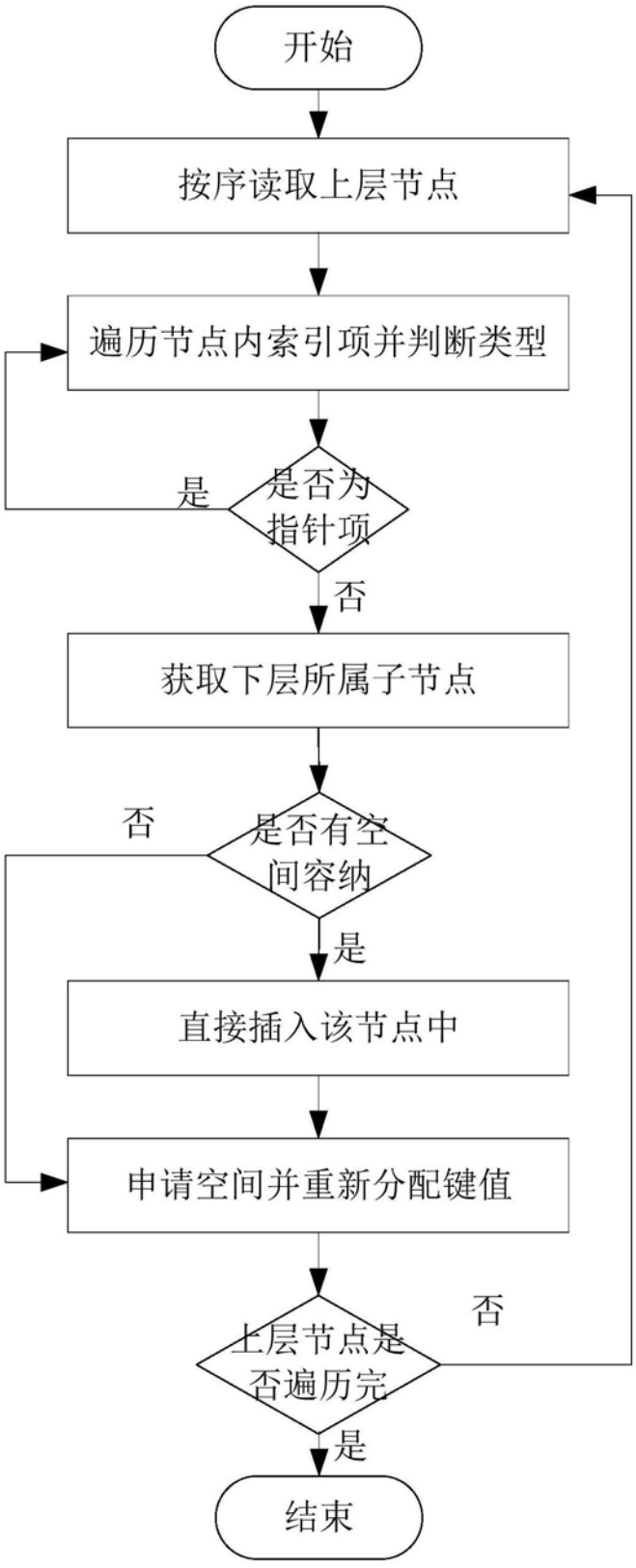

开始
按序读取上层节点
遍历节点内索引项并判断类型
是否为指针项
是
否
获取下层所属子节点
是否有空间容纳
否
是
直接插入该节点中
申请空间并重新分配键值
上层节点是否遍历完
否
是
结束

图5

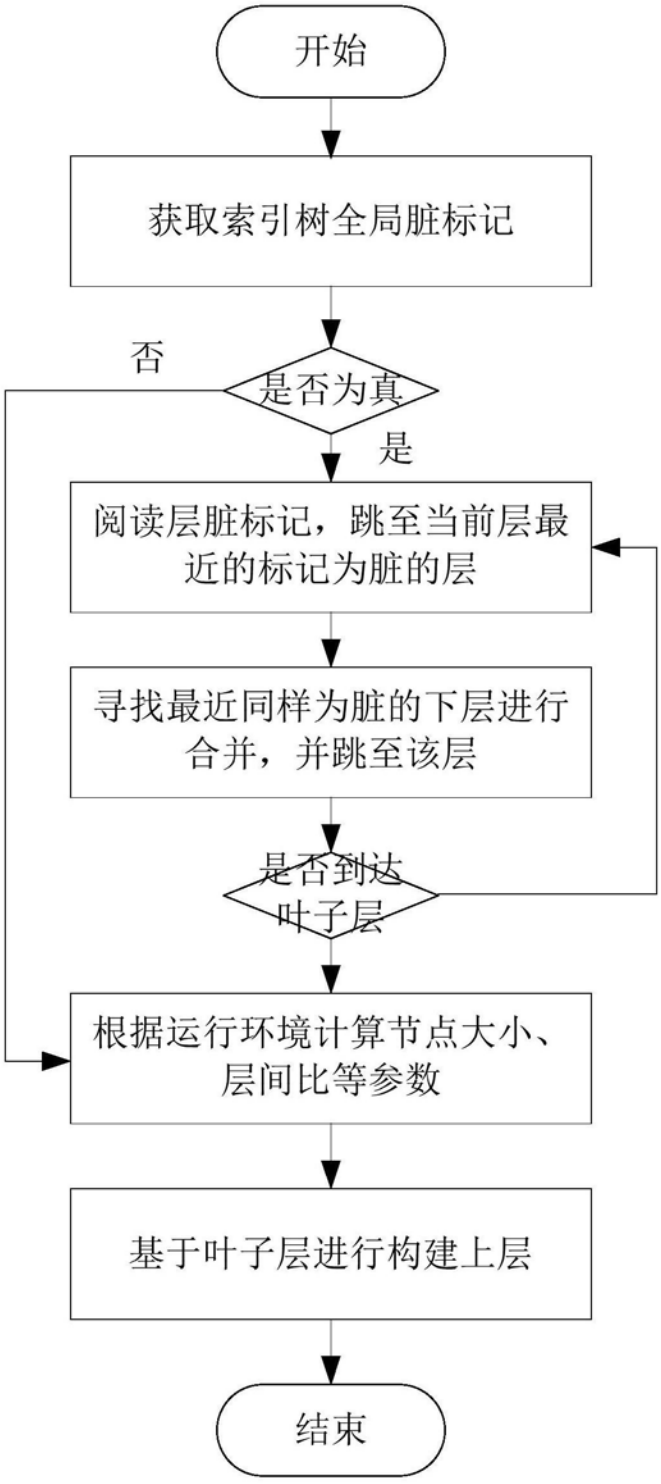

开始
获取索引树全局脏标记
否
是否为真
是
阅读层脏标记，跳至当前层最近的标记为脏的层
寻找最近同样为脏的下层进行合并，并跳至该层
是否到达叶子层
根据运行环境计算节点大小、层间比等参数
基于叶子层进行构建上层
结束

图6