(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 105359142 B
(45)授权公告日 2019.04.05

(21)申请号 201480037464.8
(22)申请日 2014.05.23
(65)同一申请的已公布的文献号
申请公布号 CN 105359142 A
(43)申请公布日 2016.02.24
(85)PCT国际申请进入国家阶段日
2016.01.04
(86)PCT国际申请的申请数据
PCT/CN2014/078304 2014.05.23
(87)PCT国际申请的公布数据
WO2015/176315 ZH 2015.11.26
(73)专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
(72)发明人 桑永嘉 李俊 施会华
(74)专利代理机构 深圳市深佳知识产权代理事务所(普通合伙) 44285
代理人 王仲凯
(51)Int.Cl.
G06F 16/00(2019.01)
(56)对比文件
CN 102508924 A,2012.06.20,
CN 101593202 A,2009.12.02,
US 2008162410 A1,2008.07.03,
US 2013173589 A1,2013.07.04,
审查员 李楠

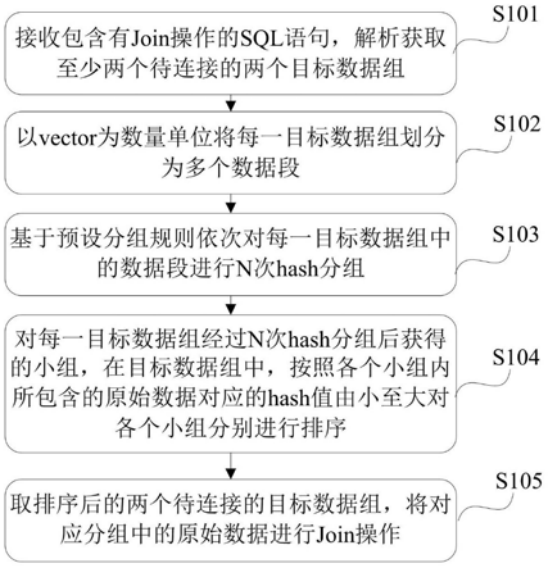
权利要求书6页 说明书19页 附图6页

(54)发明名称

哈希连接方法和装置

(57)摘要

本发明公开了一种哈希连接方法和装置。该方法在进行数据库查询时在目标数据组进行分组过程中,以vector为数量单位划分并计算数据段内原始数据的哈希值,并以比特位表示;再基于预设分组规则在哈希分组中,将位于指定位上取值相同的哈希值所对应的原始数据划分在同一小组内,在后续分组中利用前一次哈希分组中未被指定比特位继续执行哈希分组,同时,在进行分组的过程中,按照所述原始数据在目标数据组中的位置,对划分在同一小组中的原始数据进行排序,最后,再对进行分组以及排序后的待连接的目标数据组中对应分组中的原始数据进行连接操作。从而实现降低后续对各个分组进行排序的复杂度的目的。



CN 105359142 B

1.一种哈希连接方法,其特征在于,应用于数据库,包括:

接收包含有连接Join操作的结构化查询语言SQL语句,解析获取至少两个待连接的目标数据组;

以矢量vector为数量单位将每一目标数据组划分为多个数据段;

基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash分组,其中,在每次hash分组时,基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值,将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存,N取大于或等于1的正整数;

对每一目标数据组经过N次hash分组后获得的小组,在所述目标数据组中,按照各个小组中所包含的原始数据对应的hash值由小至大的对各个小组进行排序;

按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作。

2.根据权利要求1所述的方法,其特征在于,所述基于预设分组规则依次对每一目标数据组中的数据段进行N次hash分组中的第1次hash分组包括:

计算当前所述数据段内包含的原始数据的hash值,并用bit位表示计算所得hash值;

将位于指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存;

将每一个原始数据对应的hash值中未被指定的bit位与该原始数据进行关联,并保存;

所述基于预设分组规则依次对每一目标数据组中的数据段进行N次hash分组中的第2次至第n次hash分组包括:

对上一次hash分组后得到的任意一小组中的原始数据进行hash分组,n包含于N,取大于2的正整数包括:

基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位,将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存;

将每一个原始数据关联的剩余的未被指定的bit位再次保存。

3.根据权利要求1或2所述的方法,其特征在于,所述预设分组规则包括:预设hash分组次数N,或者预设分组总数S,或者预设hash分组次数N和预设分组总数S;

当所述预设分组规则是预设hash分组次数N时,依次对每一所述目标数据中的数据段进行hash分组,直至完成N次hash分组;

当所述预设分组规则是预设分组总数S时,依次对每一所述目标数据组中的数据段进行hash分组,直至每一所述目标数据组的分组数等于预设分组数;

当所述预设分组规则是预设hash分组次数N和预设分组总数S,且预设hash分组次数N的优先级高于预设分组总数S时,依次对每一所述目标数据组中的数据段进行hash分组,直至完成N次hash分组;

当所述预设分组规则是预设hash分组次数N和预设分组总数S,且预设分组总数S的优

先级高于预设hash分组次数N时，依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组总数S；

当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设hash分组次数N的优先级和预设分组总数S的优先级一致，依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组且每一所述目标数据组的分组数等于预设分组总数S；

其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数，N包含n；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数；

所述预设hash分组次数N与预设分组总数S的优先级由TLB的存储大小和cache的大小决定。

4.根据权利要求1或2所述的方法，其特征在于，所述预设分组规则包括：预设hash分组次数N，预设的每一次hash分组的分组数m和预设分组总数S；其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数，m小于N；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数；

所述依次对每一所述目标数据中的数据段进行hash分组时，按照预设的每一次hash分组的分组数进行分组，使得最后的分组次数等于预设hash分组次数，所分的小组的总数等于预设分组总数。

5.根据权利要求1或2所述的方法，其特征在于，所述以矢量vector为数量单位将每一目标数据组划分为多个数据段包括：

以矢量vector为数量单位，一个vector对应一个数据段，顺序将每一目标数据组划分为M个数据段，M的取值由所述目标数据组内的原始数据的个数，及数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

其中，第1至第M-1个数据段中所包含的原始数据的个数相同，第M个数据段中所包含的原始数据的个数小于或等于第1至M-1个数据段中所包含的原始数据的个数。

6.根据权利要求2所述的方法，其特征在于，将位于指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存包括：

获取当前所述数据段内包含的各个所述原始数据用bit位表示的hash值；

查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内，其中，依据数据库缓存cache的大小和页表缓冲TLB的存储大小指定当前hash分组所需用到的bit位；

遍历划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

按照各个下标的大小，从小至大排列各个下标对应的原始数据；

依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

7.根据权利要求2所述的方法，其特征在于，基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存包括：

调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组

中未被指定的bit位；

从调用的所述未被指定的bit位中确定当前hash分组过程中所需用到的bit位，其中，当前hash分组过程中所需用到的bit位依据数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内；

遍历划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

按照各个下标的大小，从小至大排列各个下标对应的原始数据；

依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

8.根据权利要求1或2所述的方法，其特征在于，所述按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作包括：

按顺序分别获取所述待连接的两个目标数据组进行N次hash分组后的各个小组；

两两小组为一对进行原始数据Join操作的方式，对两个目标数据组的各个小组中原始数据进行Join操作；

所述两两小组为一对进行原始数据Join操作的方式包括：

由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组；

若遍历到相同小组时，将所述小组中的原始数据，顺序与所述相同小组内的原始数据进行Join操作，其中，所述相同小组是指该小组内存储的原始数据的hash值与用于遍历的小组内存储的原始数据的hash值相同；

当所述小组中的原始数据都已进行执行Join操作后，移动至下一小组返回执行顺序遍历另一目标数据组中的各个小组这一步骤；

若未遍历到相同小组时，则移动至下一小组返回执行顺序遍历另一目标数据中的各个小组这一步骤；

直至所述目标数据组中的所有小组对另一目标数据组中的各个小组都执行遍历操作。

9.一种哈希连接装置，其特征在于，应用于数据库，包括：

接收单元，用于接收包含有连接Join操作的结构化查询语言SQL语句，解析获取至少两个待连接的目标数据组；

划分单元，用于以矢量vector为数量单位将每一目标数据组划分为多个数据段；

分组单元，用于基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash分组，其中，在每次hash分组时，基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值，将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，N取大于或等于1的正整数；

排序单元，用于对每一目标数据组经过N次hash分组后获得的小组，在所述目标数据组中，按照各个小组中所包含的原始数据对应的hash值由小至大对各个小组进行排序；

连接单元，用于按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作。

10.根据权利要求9所述的装置，其特征在于，所述分组单元包括：每一目标数据组中的

数据段进行第1次hash分组的一次hash分组模块;以及,对上一次hash分组后得到的任意一小组中的原始数据进行第2次至第n次hash分组的多次hash分组模块,n包含于N,取大于2的正整数;

所述一次hash分组模块,用于计算当前所述数据段内包含的原始数据的hash值,并用bit位表示计算所得hash值;将位于指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存;将每一个原始数据对应的hash值中未被指定的bit位与该原始数据进行关联并保存;

所述多次hash分组模块,用于基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位,将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存;将每一个原始数据关联的剩余的未被指定的bit位再次保存。

11.根据权利要求9或10所述的装置,其特征在于,包括:

当所述预设分组规则是预设hash分组次数N时,所述分组单元,用于依次对每一所述目标数据组中的数据段进行hash分组,直至完成N次hash分组;

当所述预设分组规则是预设分组总数S时,所述分组单元,用于依次对每一所述目标数据组中的数据段进行hash分组,直至每一所述目标数据组的分组数等于预设分组数;

当所述预设分组规则是预设hash分组次数N和预设分组总数S,且预设hash分组次数N的优先级高于预设分组总数S时,所述分组单元,用于依次对每一所述目标数据组中的数据段进行hash分组,直至完成N次hash分组;

当所述预设分组规则是预设hash分组次数N和预设分组总数S,且预设分组总数S的优先级高于预设hash分组次数N时,所述分组单元,用于依次对每一所述目标数据组中的数据段进行hash分组,直至每一所述目标数据组的分组数等于预设分组总数S;

当所述预设分组规则是预设hash分组次数N和预设分组总数S,且预设hash分组次数N的优先级和预设分组总数S的优先级一致,所述分组单元,用于依次对每一所述目标数据组中的数据段进行hash分组,直至完成N次hash分组且每一所述目标数据组的分组数等于预设分组总数S;

其中,N的取值由页表缓冲TLB的存储大小决定,为大于等于1的正整数,N包含n;S的取值由数据库缓存cache的大小决定,为大于等于2的正整数;所述预设hash分组次数N与预设分组总数S的优先级由TLB的存储大小和cache的大小决定。

12.根据权利要求9或10所述的装置,其特征在于,包括:

当所述预设分组规则包括预设hash分组次数N,预设的每一次hash分组的分组数m和预设分组总数S时,所述分组单元,用于按照预设的每一次hash分组的分组数进行分组,使得最后的分组次数等于预设hash分组次数,所分的小组的总数等于预设分组总数;

其中,N的取值由页表缓冲TLB的存储大小决定,为大于等于1的正整数,m小于N;S的取值由数据库缓存cache的大小决定,为大于等于2的正整数。

13.根据权利要求9或10所述的装置,其特征在于,所述划分单元包括:

第一划分模块,用于以矢量vector为数量单位,一个vector对应一个数据段,顺序将每

一目标数据组划分为M个数据段，M的取值由所述目标数据组内的原始数据的个数，及数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

其中，第1至第M-1个数据段中所包含的原始数据的个数相同，第M个数据段中所包含的原始数据的个数小于或等于第1至M-1个数据段中所包含的原始数据的个数。

14.根据权利要求10所述的装置，其特征在于，所述用于将位于指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存的所述一次hash分组模块包括：

获取子模块，用于获取当前所述数据段内包含的各个所述原始数据用bit位表示的hash值；

第一查找子模块，用于查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内，其中，依据数据库缓存cache的大小和页表缓冲TLB的存储大小指定当前hash分组所需用到的bit位；

第一遍历子模块，用于遍历划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

第一排列子模块，用于按照各个下标的大小，从小至大排列各个下标对应的原始数据；

第一排序子模块，用于依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

15.根据权利要求10所述的装置，其特征在于，所述基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存的所述多次hash分组模块包括：

调用子模块，用于调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组中未被指定的bit位；

确定子模块，用于从调用的所述未被指定的bit位中确定当前hash分组过程中所需用到的bit位，其中，当前hash分组过程中所需用到的bit位依据数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

第二查找子模块，用于查找当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内；

第二遍历子模块，用于遍历划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

第二排列子模块，用于按照各个下标的大小，从小至大排列各个下标对应的原始数据；

第二排序子模块，用于依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

16.根据权利要求9或10所述的装置，其特征在于，所述连接单元包括：

获取模块，用于按顺序分别获取所述待连接的两个目标数据组进行N次hash分组后的各个小组；

Join模块，用于两两小组为一对进行原始数据Join操作的方式，对两个目标数据组的

各个小组中原始数据进行Join操作；

其中，所述Join模块包括：

第三遍历子模块，用于由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组；若遍历到相同小组时，执行第一Join子模块；若未遍历到相同小组时，则移动至下一小组返回执行顺序遍历另一目标数据中的各个小组这一步骤；直至所述目标数据组中的所有小组对另一目标数据组中的各个小组都执行遍历操作；

所述第一Join子模块，用于将进行遍历的所述小组中的原始数据，顺序与所述相同小组内的原始数据进行Join操作，其中，所述相同小组是指该小组内存储的原始数据的hash值与用于遍历的小组内存储的原始数据的hash值相同；当所述小组中的原始数据都已进行执行Join操作后，移动至下一小组返回所述第三遍历子模块。

哈希连接方法和装置

技术领域

[0001] 本发明涉及数据库技术领域，更具体的说，是涉及一种哈希连接方法和装置。

背景技术

[0002] 随着数据库技术的发展和应用，数据库存储的数据量已从兆字节(M)及千兆字节(G)过渡到现在的兆兆字节(T)和千兆兆字节(P)。基于当前数据库所能存储的数据量，用户在查询数据库的过程中，所需要面对的则是G级、T级甚至P级的数据量。在查询如此大的数据量的情况下，需要满足查询的快速响应，则对数据库处理性能提出了很大的挑战，而对数据库性能产生至关重要的则是在查询过程中数据库对查询中包含的Join操作(连接操作)的处理响应时间。

[0003] 在数据库中实现Join操作的基本方法主要有Hash Join(哈希连接)，Merge Join以及针对Grace Join做改进后的Radix Join(聚集连接)算法。其中，在查询的过程中主要包括分组和Join两部分，为避免分组过程中，当分组数大于 CPU的TLB entry项(TLB，Translation Lookaside Buffer，页表缓冲，TLB entry 指在LTB中缓存的页表条目)时所导致的严重TLB miss(指TLB中没有所需的表页)问题，现有的查询在分组阶段多半采用多路分组的方法减少TLB miss。目前最常见的查询过程为：首先，采用多路分组的方式进行分组，且在每一次分组过程中对原始数据进行hash计算，然后，在获得多路分组后，对各个分组以及各个分组中的原始数据进行排序，最后，再对已知顺序的分组进行Join 操作。

[0004] 由上述可知，现有的进行数据库查询过程中，面临分组阶段所采用的多路分组需要多次计算hash值可能会产生大量cache miss(缓存缺失，指所请求的数据不在要访问的存储器层)，以及浪费计算资源的问题。

发明内容

[0005] 有鉴于此，本发明实施例的目的在于提供一种哈希连接方法和装置，以克服现有进行数据库查询过程中，所面临的浪费计算资源的问题。

[0006] 为实现上述目的，本发明实施例提供如下技术方案：

[0007] 本发明实施例的第一方面提供了一种哈希连接方法，应用于数据库，包括：

[0008] 接收包含有连接Join操作的结构化查询语言SQL语句，解析获取至少两个待连接的目标数据组；

[0009] 以矢量vector为数量单位将每一目标数据组划分为多个数据段；

[0010] 基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash 分组，其中，在每次hash分组时，基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值，将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，N取大于或等于1的正整数；

[0011] 对每一目标数据组经过N次hash分组后获得的小组，在所述目标数据组中，按照各

个小组中所包含的原始数据对应的hash值由小至大的对各个小组进行排序；

[0012] 按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作。

[0013] 本发明实施例的第一方面的第一种实现方式中，所述基于预设分组规则依次对每一目标数据组中的数据段进行N次hash分组中的第1次hash分组包括：

[0014] 计算当前所述数据段内包含的原始数据的hash值，并用bit位表示计算所得 hash值；

[0015] 将位于指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存；

[0016] 将每一个原始数据对应的hash值中未被指定的bit位与该原始数据进行关联，并保存；

[0017] 所述基于预设分组规则依次对每一目标数据组中的数据段进行N次hash 分组中的第2次至第n次hash分组包括：

[0018] 对上一次hash分组后得到的任意一小组中的原始数据进行hash分组，n包含于N，取大于2的正整数包括：

[0019] 基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存；

[0020] 将每一个原始数据关联的剩余的未被指定的bit位再次保存。

[0021] 本发明实施例的第一方面中所涉及的第一种所述预设分组规则包括：预设hash分组次数N，或者预设分组总数S，或者预设hash分组次数N和预设分组总数S；

[0022] 当所述预设分组规则是预设hash分组次数N时，依次对每一所述目标数据中的数据段进行hash分组，直至完成N次hash分组；

[0023] 当所述预设分组规则是预设分组总数S时，依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组数；

[0024] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设hash分组次数N的优先级高于预设分组总数S时，依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组；

[0025] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设分组总数S的优先级高于预设hash分组次数N时，依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组总数 S；

[0026] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设hash分组次数N的优先级和预设分组总数S的优先级一致，依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组且每一所述目标数据组的分组数等于预设分组总数S；

[0027] 其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数， N包含n；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数；

[0028] 所述预设hash分组次数N与预设分组总数S的优先级由TLB的存储大小和cache的

大小决定。

[0029] 本发明实施例第一方面中涉及到的第二种所述预设分组规则包括:预设 hash分组次数N,预设的每一次hash分组的分组数m和预设分组总数S;其中, N的取值由页表缓冲TLB的存储大小决定,为大于等于1的正整数,m小于N; S的取值由数据库缓存cache的大小决定,为大于等于2的正整数;

[0030] 所述依次对每一所述目标数据中的数据段进行hash分组时,按照预设的每一次hash分组的分组数进行分组,使得最后的分组次数等于预设hash分组次数,所分的小组的总数等于预设分组总数。

[0031] 本发明实施例的第一方面的第二种实现方式中,所述以矢量vector为数量单位将每一目标数据组划分为多个数据段包括:

[0032] 以矢量vector为数量单位,一个vector对应一个数据段,顺序将每一目标数据组划分为M个数据段,M的取值由所述目标数据组内的原始数据的个数,及数据库缓存cache的大小和页表缓冲TLB的存储大小决定;

[0033] 其中,第1至第M-1个数据段中所包含的原始数据的个数相同,第M个数据段中所包含的原始数据的个数小于或等于第1至M-1个数据段中所包含的原始数据的个数。

[0034] 本发明实施例第一方面的第三种实现方式中,将位于指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存包括:

[0035] 获取当前所述数据段内包含的各个所述原始数据用bit位表示的hash值;

[0036] 查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据,将各个原始数据划分在同一小组内,其中,依据数据库缓存cache 的大小和页表缓冲TLB的存储大小指定当前hash分组所需用到的bit位;

[0037] 遍历划分在同一小组内的各个原始数据的下标,所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置;

[0038] 按照各个下标的大小,从小至大排列各个下标对应的原始数据;

[0039] 依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0040] 本发明实施例第一种实现方式中,基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位,将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存包括:

[0041] 调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组中未被指定的bit位;

[0042] 从调用的所述未被指定的bit位中确定当前hash分组过程中所需用到的bit 位,其中,当前hash分组过程中所需用到的bit位依据数据库缓存cache的大小和页表缓冲TLB的存储大小决定;

[0043] 查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据,将各个原始数据划分在同一小组内;

[0044] 遍历划分在同一小组内的各个原始数据的下标,所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置;

[0045] 按照各个下标的大小,从小至大排列各个下标对应的原始数据;

[0046] 依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0047] 本发明实施例第一种实现方式中,所述按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作包括:

[0048] 按顺序分别获取所述待连接的两个目标数据组进行N次hash分组后的各个小组;

[0049] 两两小组为一对进行原始数据Join操作的方式,对两个目标数据组的各个小组中原始数据进行Join操作;

[0050] 所述两两小组为一对进行原始数据Join操作的方式包括:

[0051] 由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组;

[0052] 若遍历到相同小组时,将所述小组中的原始数据,顺序与所述相同小组内的原始数据进行Join操作,其中,所述相同小组是指该小组内存储的原始数据的hash值与用于遍历的小组内存储的原始数据的hash值相同;

[0053] 当所述小组中的原始数据都已进行执行Join操作后,移动至下一小组返回执行顺序遍历另一目标数据组中的各个小组这一步骤;

[0054] 若未遍历到相同小组时,则移动至下一小组返回执行顺序遍历另一目标数据中的各个小组这一步骤;

[0055] 直至所述目标数据组中的所有小组对另一目标数据组中的各个小组都执行遍历操作。

[0056] 本发明实施例的第二方面提供了一种哈希连接装置,应用于数据库,包括:

[0057] 接收单元,用于接收包含有连接Join操作的结构化查询语言SQL语句,解析获取至少两个待连接的目标数据组;

[0058] 划分单元,用于以矢量vector为数量单位将每一目标数据组划分为多个数据段;

[0059] 分组单元,用于基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash分组,其中,在每次hash分组时,基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值,将当前hash分组过程中指定 bit位上取值相同的hash值所对应的原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存,N取大于或等于1的正整数;

[0060] 排序单元,用于对每一目标数据组经过N次hash分组后获得的小组,在所述目标数据组中,按照各个小组中所包含的原始数据对应的hash值由小至大对各个小组进行排序;

[0061] 连接单元,用于按照排序依次取所述两个待连接的目标数据组中经由N次 hash分组后获得的各个小组中的原始数据进行Join操作。

[0062] 经由上述的技术方案可知,与现有技术相比,本发明实施例公开了一种哈希连接方法和装置。该方法在进行数据库查询时,在确定待连接的目标数据组之后,对目标数据组进行分组过程中,首先,以矢量vector为数量单位将待连接的目标数据组划分为多个数据段,然后,计算数据段内包含的原始数据的hash值,并用比特bit位表示hash值;然后,基于预设分组规则,在第1次 hash分组时计算所得的各个原始数据用bit位表示的hash值,在进行hash分组的过程中,将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内,并对划分在同一小组内的各个原始数据,按照各个原始数

据在所述目标数据组中的位置在同一小组内进行排序并保存。

[0063] 本发明实施例通过以vector为数量单位以及在hash分组过程中利用指定bit 位执行hash分组，能够实现同时对若干原始数据进行hash分组处理，且在多次hash分组的过程中不需要重复计算原始数据的hash值，即减少了cache miss缓存缺失，也省去了重复计算hash值避免了计算资源的浪费。

[0064] 并且每次分组划分至每个小组中的原始数据有序，这样完成多个数据段分组后得到的每个小组中的原始数据局部有序，在对局部有序的原始数据进行join时，其排序复杂度低于随机分配的原始数据进行join时的排序复杂度。

附图说明

[0065] 为了更清楚地说明本发明实施例的技术方案，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发明的实施例，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，还可以根据提供的附图获得其他的附图。

[0066] 图1为本发明实施例一公开的一种哈希连接方法的流程图；

[0067] 图2为本发明实施例三公开的示例四公开的3次hash分组的示意图；

[0068] 图3为本发明实施例四公开的每个数据段中包含相同原始数据的示意图；

[0069] 图4为本发明实施例四公开的第1次hash分组过程中划分小组的流程图；

[0070] 图5为本发明实施例四公开的对一段数据段中的原始数据进行hash分组的示意图；

[0071] 图6为本发明实施例四公开的第2次至第N次hash分组过程中划分小组的流程图；

[0072] 图7为本发明实施例四公开的对两个待连接的目标数据中各个小组中的原始数据进行Join操作的流程图；

[0073] 图8为本发明实施例五公开的一种哈希连接装置的结构示意图；

[0074] 图9为本发明实施例五公开的一种数据库管理系统的结构示意图。

具体实施方式

[0075] 为了引用和清楚起见，下文中使用的技术名词的说明、简写或缩写总结如下：

[0076] TLB，Translation Look aside Buffer，页表缓冲，TLB entry指在LTB中缓存的页表条目；

[0077] Radix Join，聚集连接；

[0078] cache miss，缓存缺失，指所请求的数据不在要访问的存储器层。

[0079] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0080] 由背景技术可知，在当前常用的查询过程中，分组阶段所采用的多路分组，在每一次分组过程中都需要采用一个一个处理原始数据的方式，需要多次计算原始数据的hash值从而面临浪费计算资源的问题。因此，本发明实施例提供了一种哈希连接方法和装置，通过

利用矢量vector为数量单位以及在后续分组过程中利用当前hash分组过程中被指定bit位执行hash分组，能够实现同时对若干原始数据进行hash分组处理，且在多次hash分组的过程中不需要重复计算原始数据的hash值，即减少了cache miss缓存缺失，也省去了重复计算hash 值，避免了计算资源的浪费。同时，每次hash分组划分至每个小组中的原始数据有序，这样完成多个数据段分组后得到的每个小组中的原始数据局部有序，在对局部有序的原始数据进行join时，其排序复杂度低于随机分配的原始数据进行join时的排序复杂度。具体过程通过以下本发明实施例进行详细说明。

[0081] 实施例一

[0082] 本发明实施例一公开了一种哈希连接方法，该方法应用于数据库，其流程如图1中的步骤S101至步骤S105所示，具体过程包括：

[0083] 步骤S101，接收包含有连接Join操作的结构化查询语言SQL语句，解析获取至少两个待连接的目标数据组；

[0084] 在执行数据库查询的过程中，执行步骤S101，由数据库对接收到的包含有Join操作的SQL查询语句进行解析，从中至少获取两个待连接的目标数据组。也就是说，以两个待连接的目标数据组为一对，在解析的过程中至少会出现两个待连接的目标数据组，也就是说待连接的目标数据组是成对解析的。

[0085] 步骤S102，以矢量vector为数量单位将每一目标数据组划分为确定数据的多个数据段；

[0086] 在步骤S102中，针对解析出的成对的两个待连接的目标数据组执行相同的操作，在划分数据段的过程中以一个目标数据组为例。

[0087] 以矢量vector为数量单位划分当前的目标数据组。具体的，以该vector为数量单位是指以一个vector内包含多少个原始数据为固定单位。并利用该 vector数量单位将目标数据组划分为多个数据段，也就是说一个数据段对应一个vector。

[0088] 需要说明的是，在通常情况下，以一个数据段中可包含的最多原始数据个数为一个vector单位，将所述目标数据组划分为多个数据段，划分后的各个数据段中包含的原始数据个数通常为相同的。当然也存在，按照预设分组规则以及该目标数据组中的原始数据的总个数对一个数量单元vector中所包含的原始数据的个数进行限定，并非以其所能包含的最多的原始数据的个数对该数量单元vector进行限定。

[0089] 上述两种方式，都不排除，最后一个数据段中所包含的原始数据的个数小于其他数据段中包含的原始数据的个数的情况。

[0090] 基于上述方式，执行步骤S102之后，可将每一个待连接的目标数据组都划分成多个数据段。本发明实施例采用矢量化的方法，在后续进行hash分组的过程中，以一个vector为数量单位，针对该vector内的原始数据同时计算hash 值，然后将同一个分组中的若干个原始数据一次性写入对应的分组中，从而减少了cache miss，能够提升join性能。

[0091] 步骤S103，基于预设分组规则依次对每一目标数据组中的数据段进行N 次hash分组，其中，在每次hash分组时，基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值，将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，N取大于或等于1的正整数；

[0092] 在执行上述步骤S103的过程中，基于预设分组规则依次对每个目标数据组中的数据段进行N次hash分组。其中，在第1次hash分组过程中，以一个目标数据组为例，由上至下从该目标数据的第一个数据段开始hash分组至最后一个数据段结束。以一个数据段为例，在进行第1次hash分组时，对该数据段内所包含的全部原始数据同时计算hash值，并将各个原始数据的hash值用bit位表示，该bit位与安装该数据库的计算机本身的位数相关，是由当前是计算机的 CPU最大寻址数决定的。

[0093] 例如，当前安装该数据库的计算机为32位，则在进行第1次hash分组过程中计算的原始数据所对应的hash值用32位的bit位表示。若当前安装该数据库的计算机为64位，则在进行第1次hash分组过程中计算的原始数据所对应的hash 值用64位的bit位表示。

[0094] 然后，根据当前第1次hash分组所需要利用到的bit位数，也就是指定bit位，在各个用bit位表示的hash值的指定bit位上进行比对，或遍历，或查找在指定 bit位上取值相同的hash值，并将该hash值所对应的原始数据划分在同一小组内。例如，第1次hash分组所需要的bit位数为2位，则此时从各个用bit位表示的hash值的最高位开始，向后取两位bit位进行比对，或遍历，或查找，将这两位bit位上数值相同的hash值所对应的原始数据划分在同一小组内。

[0095] 最后，针对划分在同一小组内的各个原始数据，按照各个原始数据在目标数据组中的位置在该小组内进行排序，该位置也可以认为是各个原始数据在数据段中的位置。例如，原始数据A、B、C划分在同一小组内，若A排在目标数据组的第3位，B排在目标数据组的第1位，C排在目标数据组的第6位，经过排序后，在该小组内A、B、C的实际存储顺序为：B、A、C。

[0096] 需要说明的是，由上至下对每一个数据段进行第1次hash分组的过程相同，从开始第1次hash分组开始依次从未被指定的最高位bit位开始指定bit位。在执行N次hash值分组过程中，除第1次hash分组时需要计算原始数据的hash值之后，后续hash分组过程中，仅利用各个原始数据对应的hash值中未被指定的bit 位进行hash分组，将当前hash分组过程中所需利用的bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并采用与第1次hash分组相同的方式，对划分在同一小组内的各个原始数据，按照各个原始数据在目标数据组或者数据段中的位置对各个原始数据在本小组内进行排序。

[0097] 在步骤S103中所提到的预设分组规则是指，预设hash分组次数N，或者预设分组总数S，或者预设hash分组次数N和预设分组总数S；以及，预设hash分组次数N，预设的每一次hash分组的分组数m和预设分组总数S。其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数，m小于N；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数。

[0098] 步骤S104，对每一目标数据组经过N次hash分组后获得的小组，在所述目标数据组中，按照各个小组内所包含的原始数据对应的hash值由小至大对各个小组分别进行排序；

[0099] 在步骤S104中，对按照预设分组规则进行N次hash分组后得到的目标数据组中的各个小组进行再次排序。方式为：按照该小组内所包含的原始数据的 hash值的大小，对各个小组进行排序。如：在对目标数据组进行分组后获得小组1、小组2和小组3；其中，小组1中包含的原始数据的hash值为3，小组2中包含的原始数据的hash值为5，小组3中包含的原始数据的hash值为0，在进行排序后，此时该目标数据组中的小组的顺序为：小组3、小组1和小组2。

[0100] 需要说明的是，按照预设分组规则进行N次hash分组后得到的各个小组，最后划分在同一小组内的原始数据通常对应相同的hash值。

[0101] 步骤S105，按照排序依次取所述两个待连接的目标数据组中经由N次hash 分组后获得的各个小组中的原始数据进行Join操作。

[0102] 针对通过上述执行步骤S102至步骤S104的hash分组过程中对所划分的同一小组内原始数据进行排序后的两个待连接的目标数据组，执行步骤S105，针对每个待连接的目标数据组中有序的小组，按照顺序将一个待连接的目标数据组中一个小组与另一个待连接的目标数据组中的一下小组进行Join操作，即将各个小组中有序的原始数据执行Join操作。从而实现当次的数据库查询的任务。

[0103] 针对现有技术中由于硬件的TLB entry项大于cache way的数目，采用一个个计算的hash值进行分组容易导致大量的cache thrashing，从而产生大量的 cache miss，影响原本join的性能的问题。通过上述本发明实施例一公开的哈希连接方法，以一个vector为数量单位按组计算hash值，然后将同一个分组中所包含的若干个原始数据对应的hash值一次性写入对应的小组中。以vector的形式进行hash分组则能够避免产生不必要的cache thrashing，从而减少了cache miss，实现提升Join性能的目的。并且，仅在第1次分组过程中计算各个原始数据的hash值，而将后续使用到的若干bit位记录到对应的各个原始数据的关联位置处以备后续分组过程中直接使用，从而省去重复计算hash值的代价，避免了资源浪费。

[0104] 同时，在本发明实施例一公开的哈希连接进行hash分组的过程中，在每一次hash分组之后，将原始数据写入各个对应的小组内之前，对各个小组内的原始数据进行排序，使得在最后分组完成之后，对各个小组做最后的排序时，由于在本发明实施例公开的多路分组的过程中已经对原始数据进行了一定程度上的局部排序，各个小组内的原始数据在局部上是有序的，因此仅需要对各个小组进行排序即可。通过该种方式，能够大大的降低现有技术中在分组完成之后，再对各个小组内的原始数据以及各个小组进行排序的复杂度，减少因排序消耗的时间。并且在这种局部有序的原始数据进行join时，其排序复杂度低于随机分配的原始数据进行join时的排序复杂度。

[0105] 实施例二

[0106] 基于本发明实施例一公开的哈希连接方法，在本发明实施例二中主要针对图1示出的步骤S103中提及的N次hash分组进行详细说明。

[0107] 基于预设分组规则依次对每一目标数据组中的数据段进行N次hash分组中的第1次hash分组的过程包括：

[0108] 步骤S1031，计算当前所述数据段内包含的原始数据的hash值，并用bit位表示计算所得hash值；

[0109] 基于执行步骤S102以一个vector为数量单位划分目标数据组，以所述目标数据组中的任意一个数据段为例，在执行步骤S1031时，同时计算同一数据段内所包含的各个原始数据的hash值，并利用比特bit位表示计算每个原始数据所得的hash值。如本申请实施例一中所述该比特bit位与安装该数据库的计算机本身的位数相关，是由当前是计算机的CPU最大寻址数决定的。

[0110] 步骤S1032，将位于指定bit位上取值相同的hash值所对应的各个原始数据划分在

同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存；

[0111] 在执行步骤S1032的过程中，在当前进行第1次hash次分组的过程中，依据数据缓存cache的大小和页表缓冲TLB的存储大小，确定当前hash分组所需要指定的bit位，针对该数据段中的各个原始数据对应的利用bit位表示的hash值，在划分小组的过程中，将指定bit位上取值相同的hash值所对应的原始数据划分在同一个小组内。

[0112] 如，在当前分组过程中需要两位bit，则针对当前用bit位表示的hash值从最高位起向最低位方向指定两位，在划分小组时，将指定的前两位相同的hash 值对应的原始数据划分在同一个小组内。

[0113] 同时，在依据指定的bit位相同，获知那些原始数据可以划分在同一个小组内时，利用该原始数据在目标数据组中的位置，在其当前所在的小组内进行排序。如，在同一小组内包含原始数据：A、B、C，其中A的位置在目标数据组的第6位，B的位置在目标数据组的第1位，C的位置在目标数据组的第4 位，则执行步骤S1033后获取到的保存后的小组内的原始数据的位置为：B、C、 A，这样使得每次划分得到的每个小组中的原始数据有序。

[0114] 步骤S1033，将每一个原始数据对应的hash值中未被指定的bit位与该原始数据进行关联，并保存于各个hash值对应的原始数据的关联位置处；

[0115] 基于步骤S1032，执行步骤S1033在划分小组后，将该原始数据对应的hash 值在该次hash分组过程中未被使用的，或者未被指定的bit位保存于该原始数据的关联位置处。其中，该关联位置可以为与该原始数据相邻的存储空间，也可以是其他与该原始数据建立关联的存储空间。

[0116] 针对目标数据组中的各个数据段执行完上述第1次hash分组之后，若满足预设分组规则，则停止进行再次分组。若不满足预设分组规则，则继续对当前第1次hash分组后的各个小组内的原始数据进行再次分组。

[0117] 所述第2次至第n次hash分组中对上一次hash分组后得到的任意一小组中的原始数据进行hash分组，n取大于2的正整数且包含于N中。上述基于预设分组规则依次对每一目标数据组中的数据段进行N次分组中的第2次甚至n次hash分组的过程包括：

[0118] 步骤S1034，基于当前小组内的原始数据关联位置处所保存的上一次hash 分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash 值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置对同一小组内的各个原始数据进行排序并保存；

[0119] 在执行步骤S1034的过程中，依据在原始数据关联位置处保存的bit位中指定的，当前进行hash分组所需要用到的bit位，划分位于指定bit位上取值相同的hash值所对应的原始数据为同一小组，同时，在依据指定的bit位相同，获知那些原始数据可以划分在同一个小组内时，利用该原始数据在目标数据组中的位置，在其当前所在的小组内进行排序。

[0120] 步骤S1035，将每一个原始数据关联的剩余的未被指定的bit位再次保存在所述原始数据的关联位置处；

[0121] 在步骤S1035中，将剩余的未被指定的bit位再次保存在所述原始数据的关联位置处，以备后续分组中使用。结合步骤S1032中的示例，当前在原始数据关联位置处保存的bit位为执行步骤S1032后剩余的未使用的bit。若当前进行 hash分组所需要用到的bit位仍为

两位，同样的，则指定的两位bit位则为从当前剩余的bit位的最高位开始向最低位方向处所取的两位。

[0122] 在执行完步骤S1034与步骤S1035之后，若当前的分组情况不满足预设分组规则，则返回循环执行步骤S1034和步骤S1035，直至满足预设分组规则时停止对当前目标数据组进行分组。

[0123] 通过执行步骤S1031至步骤S1035，对目标数据组进行满足预设分组规则的分组，并在每一次分组的过程中对划分在同一小组内的原始数据进行排序，使得每一次进行hash分组过程中得到分组结果虽然在整体上是无序的，但是在获得的每个小组内则是有序的，在这种局部有序的原始数据进行join时，其排序复杂度低于随机分配的原始数据进行join时的排序复杂度。

[0124] 通过上述本发明实施例二具体公开的仅在第1次分组过程中计算各个原始数据的hash值，而将后续使用到的若干bit位记录到对应的各个原始数据的关联位置处以备后续分组过程中直接使用，从而省去重复计算hash值的代价，避免了资源浪费。同时，在每一次hash分组之后，将原始数据写入各个对应的小组内之前，对各个小组内的原始数据进行排序，使得在最后hash分组完成之后，各个小组内的原始数据在局部上是有序的，因此仅需要对目标数据组hash分组后得到的各个小组进行排序即可。通过该种方式，能够大大的降低现有技术中在分组完成之后，再对各个小组内的原始数据以及各个小组进行排序的复杂度，减少因排序消耗的时间。

[0125] 实施例三

[0126] 基于本发明实施例一和实施例二公开的哈希连接方法，在本发明实施例二中主要针对图1示出的步骤S103中提及的预设分组规则进行详细说明。

[0127] 当所述预设分组规则是预设hash分组次数N时，在依次对每一所述目标数据组中的数据段进行hash分组的过程中，直至完成N次hash分组后停止对该目标数据组进行分组。其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数。

[0128] 示例一，由页面缓冲TLB的存储确定当前进行hash分组的目标数据组需要分4次，即N取值为4。在进行完第1次分组之后，基于本发明实施例一中公开的所述第2次至第n次hash分组中对上一次hash分组后得到的任意一小组中的原始数据进行hash分组的过程，在执行至第4次分组之后，停止对该目标数据组进行hash分组。此时，得到的小组数即为该目标数据组的分组数。

[0129] 当所述预设分组规则是预设分组总数S时，依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组总数等于预设分组总数S，停止对该目标数据组进行分组。该S的取值由数据库缓存cache的大小决定，为大于等于2的正整数。

[0130] 示例二，由数据库缓存cache的大小决定的当前目标数据组可分的预设分组总数为10时，针对当前目标数据组进行第1次hash分组，当该第1次hash分组完成后，得到的分组数小于10，则继续进行hash分组，直至当前目标数据组的分组数达到10之后停止hash分组。

[0131] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设hash分组次数N的优先级高于预设分组总数S时，依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组；

[0132] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设分组总数S

的优先级高于预设hash分组次数N时，依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组总数 S；

[0133] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设 hash分组次数N的优先级和预设分组总数S的优先级一致，依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组且每一所述目标数据组的分组数等于预设分组总数S；

[0134] 其中，所述预设hash分组次数N与预设分组总数S的优先级由TLB的存储大小和cache的大小决定。

[0135] 示例三，由页表缓冲TLB的存储大小决定的预设分组次数为3，由数据库缓存cache的大小决定的预设分组总数为16。当预设hash分组次数N的优先级和预设分组总数S的优先级一致时，则基于该预设分组次数对目标数据组进行3 次分组之后得到的分组总数正好为16；当预设hash分组次数N的优先级高于预设分组总数S时，此时基于该预设分组次数对目标数据组进行3次分组之后，可能存在的情况是，得到的分组总数小于16，或者等于16，或者大于16；当预设分组总数S的优先级高于预设hash分组次数N时，此时在分组的过程中，可能存在的情况是，在得到分组总数为16时，针对该目标数据组的分组次数大于3次，或者小于3次，或者等于3次。

[0136] 当所述预设分组规则包括：预设hash分组次数N，预设的每一次hash分组的分组数m和预设分组总数S；其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数，m小于N；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数；在依次对每一所述目标数据中的数据段进行hash分组时，按照预设的每一次hash分组的分组数m进行分组，使得最后的分组次数等于预设hash分组次数N，所分的小组的总数等于预设分组总数S。

[0137] 示例四，如图2所示，由页表缓冲TLB的存储大小决定的预设分组次数为3，每一次hash分组的分组数为2，由数据库缓存cache的大小决定的预设分组总数为16。在以vector为数量单位划分为2个数据段的该目标数据组中，在第1次hash 分组过程中分别将每个数据段再划分为2个小组，并分别写入对应的小组内；然后在第2次hash分组过程中将前一次分组后的每个小组再次划分为2个数据段并分别写入对应的小组内，依次类推直至对该目标数据组执行完3次hash分组并得到16个小组。

[0138] 在本发明实施例二中主要对图1示出的步骤S103中提及的在进行hash分组过程中基于的预设分组规则进行说明。该预设分组规则主要基于按照该数据库的计算机中的页表缓冲TLB的存储大小，以及数据库缓存cache的大小决定，基于该预设分组规则能够避免在分组的过程中出现cache miss的情况，进而提高后续Join的性能。

[0139] 实施例四

[0140] 基于本发明实施例一至实施例三公开的一种哈希连接方法，其中，针对图1中示出的步骤S102，所述以矢量vector为数量单位将每一目标数据组划分为多个数据段数，其具体过程包括：

[0141] 以矢量vector为数量单位，一个vector对应一个数据段，顺序将每一目标数据组划分为M个数据段，M的取值由所述目标数据组内的原始数据的个数，及数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

[0142] 其中，第1至第M-1个数据段中所包含的原始数据的个数相同，第M个数据段中所包含的原始数据的个数小于或等于第1至M-1个数据段中所包含的原始数据的个数。

[0143] 假设需要进行hash分组的目标数据组中总共包含有25个原始数据，以 vector为数量单元，该vector数量单位中包含5个原始数据，使5个原始数据构成一个数据段。以该vector数量单位划分包含有25个原始数据的目标数据组，可划分为5个数据段。第1个至第5个数据段中所包含的原始数据相同，如图3 所示给出的为每个数据段中所包含的原始数据个数相同的情况。

[0144] 假设需要进行hash分组的目标数据组中总共包含有28个原始数据，以 vector为数量单元，该vector数量单位中包含5个原始数据，使5个原始数据构成一个数据段。以该vector数量单位划分包含有28个原始数据的目标数据组，可划分为6个数据段。第1个至第5个数据段中所包含的原始数据相同，第6个数据段中包含3个原始数据，小于第1个值第5个数据段中包含的原始数据个数。

[0145] 基于本发明实施例二公开的一种哈希连接方法，其中，针对上述公开的步骤S1032中的，将位于指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置对同一小组内的各个原始数据进行排序并保存，其具体过程如图4所示，包括：

[0146] 步骤S201，获取当前所述数据段内包含的各个所述原始数据用bit位表示的hash值；

[0147] 步骤S202，查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内；

[0148] 基于步骤S201中获取的当前进行hash分组的数据段中各个原始数据的 hash值，该hash值用bit位表示。在步骤S202中，查找指定bit位上的hash值。该指定bit位可以是在进行本次分组之前依据数据库缓存cache的大小和页表缓冲 TLB的存储大小指定的；也可以在接收到需要进行hash分组时，依据数据库缓存cache的大小和页表缓冲TLB的存储大小对后续进行分组过程中所需要使用到的bit位进行指定，当在进行本次分组时，则不需要再重新指定，直接在本次hash分组所需要使用的bit位上查找即可。

[0149] 步骤S203，遍历将划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

[0150] 步骤S204，按照各个下标的大小，从小至大排列各个下标对应的原始数据；

[0151] 步骤S205，依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0152] 执行上述步骤S203至步骤S205在分组的过程中对将划分在同一小组内的原始数据进行排序并写入同一小组内保存，使得在对该目标数据组分组的过程中局部有序。具体过程举例说明，在进行hash分组时，以vector为数量单位的一段数据(如图5中的虚线框所示)，对该数据段中的原始数据一起计算hash 值。如图5所示，value为参与join的真实值，图5中position代表各个原始数据在整个数据段中的位置，position-1代表经过整理后分在同一组的各个原始数据的下标，hashvalue代表对应原始数据的hash值。

[0153] 在分组的过程中，遍历位于指定bit位上取值相同的hash值将其下标保存到position-1对应的小组中，然后，依次遍历position-1中保存的下标，并将该下标对应的原始数据写入到对应的小组中。

[0154] 通过执行上述步骤S203至步骤S205，在分组的过程中，在原始数据写入当前小组的同时，对本次需要写入当前小组的原始数据进行排序。在该vector 单位执行完上述分组

之后，对下一个相邻的vector进行如上操作，直到该目标数据组中的所有vector都执行完本次hash分组。进而得到该目标数据组的第1 次hash分组后各个在局部有序的小组，从而分担最终对各个小组进行排序时还要对其内部的原始数据进行排序的负担，实现了降低分组复杂度的目的。

[0155] 针对当前进行分组的目标数据组中的各个数据段都按照上述方式执行完第1次hash分组之后，若当前的分组满足预设分组规则，则停止进行再次分组。若不满足预设分组规则，则继续对当前第1次hash分组后的各个小组内的原始数据进行再次分组。基于本发明实施例二公开的一种哈希连接方法，其中，针对上述公开的步骤S1034，基于当前小组内的原始数据关联位置处所保存的上一次hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，其具体过程如图6所示，包括：

[0156] 步骤S301，调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组中未被指定的bit位；

[0157] 在执行步骤S301的过程中，当前小组为在上一次hash分组之后获得的各个小组中的任意一个小组，调用当前小组内所述原始数据关联位置处所保存的上一次hash分组中未被指定的bit位，是为了进一步的当前小组进行再次hash 分组。

[0158] 步骤S302，从调用的所述未被指定的bit位中确定当前hash分组过程中所需用到的bit位，其中，当前hash分组过程中所需用到的bit位依据数据库缓存 cache的大小和页表缓冲TLB的存储大小决定；

[0159] 步骤S303，查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内。

[0160] 步骤S304，遍历将划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

[0161] 步骤S305，按照各个下标的大小，从小至大排列各个下标对应的各个原始数据；

[0162] 步骤S306，依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0163] 上述步骤S304至步骤S306中对将划分在同一小组内的原始数据的排序过程与上述附图4中的步骤S203至步骤S205相同，具体说明可参照，这里不再进行赘述。

[0164] 针对该目标数据组前一次hash分组得到的各个小组执行上述步骤S301至步骤S306，从而获得再次hash分组后的内部原始数据有序的新小组，同样的，在每次hash分组完之后，若当前的hash分组满足预设分组规则，则停止hash分组。若不满足预设分组规则，则执行步骤S301至步骤S303再次对前一次hash 分组得到的各个小组进行分组，直至满足预设分组规则。

[0165] 基于上述本发明实施例一至本发明实施例三中公开的一种哈希连接方法，其中，针对上述公开的步骤S105，所述按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作，具体过程包括：

[0166] 步骤S501，按顺序分别获取所述待连接的两个目标数据组进行N次hash分组后的各个小组；

[0167] 在至少两个待连接的目标数据组都按照上述步骤S102至步骤S104进行hash分组

之后，执行步骤S501，获取待连接的两个目标数据组中的各个小组。

[0168] 步骤S502，两两小组为一对进行原始数据Join操作的方式，对两个目标数据组的各个小组中原始数据进行Join操作；

[0169] 针对进行hash分组之后的两个待连接的目标数据组，按照两两小组为一对进行原始数据Join操作，对两个待连接的目标数据组中各个小组中的原始数据进行Join操作的方式，如图7所示包括：

[0170] 步骤S503，由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组；

[0171] 步骤S504，判断当前小组是否在另一目标数据组中遍历到相同的小组，若是，则执行步骤S505，若否，则执行步骤S507；

[0172] 步骤S505，若遍历到相同小组时，将所述小组中的原始数据，顺序与所述相同小组内的原始数据进行Join操作，其中，所述相同小组是指该小组内存储的原始数据的hash值与用于遍历的小组内存储的原始数据的hash值相同；

[0173] 步骤S506，判断当前进行Join操作的两个小组中，任意一方中的原始数据是否都已经执行Join操作，若是，则执行步骤S507，若否，则继续执行两个小组内的原始数据的Join操作，并返回执行步骤S506；

[0174] 步骤S507，移动至下一小组返回执行步骤S503；

[0175] 循环执行上述步骤S503至步骤S507，直至所述目标数据组中的所有小组对另一目标数据组中的各个小组都执行遍历操作。

[0176] 在本发明实施例中hash连接进行分组以及Join过程中所需执行的过程。以vector为数量单位，仅在第1次分组过程中对每个vector单元内的原始数据同时计算hash值，然后将同一个分组中所包含的若干个原始数据对应的hash值一次性写入对应的分组中。而将后续使用到的若干bit位记录到对应的各个原始数据的关联位置处以备后续分组过程中直接使用，从而省去重复计算hash值的代价，避免了资源浪费。

[0177] 同时，在本发明实施例中，在每一次hash分组之后，将原始数据写入各个对应的小组内之前，对各个小组内的原始数据进行排序，以及针对本次hash 分组完成后对各个小组进行排序，使得在最后分组完成之后，对各个小组做最后的排序时能够实现降低对小组以及小组内部数据进行排序的负担，减少因排序消耗的时间的目的。

[0178] 实施例五

[0179] 针对上述本发明实施例一至本发明实施例四公开且详细描述的哈希连接方法，本发明实施例五还公开了对应执行上述方法的哈希连接装置以及数据库管理系统，下面给出具体的实施例进行详细说明。

[0180] 如图8所示，该哈希连接装置，应用于数据库，主要包括：接收单元101，划分单元102，分组单元103，排序单元104和连接单元105。

[0181] 接收单元101，用于接收包含有连接Join操作的结构化查询语言SQL语句，解析获取至少两个待连接的目标数据组；

[0182] 在执行接收单元101之后，针对解析获取到的每个目标数据组，进行后续的划分单元102，分组单元103和排序单元104经历划分，分组以及排序之后，进入连接单元105，使分组后的待连接的两个目标数据组执行Join操作。

[0183] 划分单元102，用于以矢量vector为数量单位将每一目标数据组划分为多个数据

段；

[0184] 分组单元103，用于基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash分组，其中，在每次hash分组时，基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值，将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，N取大于或等于1的正整数；

[0185] 排序单元104，用于对每一目标数据组经过N次hash分组后获得的小组，在所述目标数据组中，按照各个小组中所包含的原始数据对应的hash值由小至大对各个小组进行排序；

[0186] 连接单元105，用于按照排序依次取所述两个待连接的目标数据组中经过 N次hash分组后获得的各个小组中的原始数据进行Join操作。

[0187] 其中，所述分组单元103包括：由上至下对所述目标数据组中的数据段进行第1次hash分组一次hash分组模块1031；以及，对上一次hash分组后得到的任意一小组中的原始数据进行第2次至第n次hash分组的多次hash分组模块 1032，n取大于2的正整数；

[0188] 所述一次hash分组模块1031，用于计算当前所述数据段内包含的原始数据的hash值，并用比特bit位表示计算所得hash值；将位于指定bit位上取值相同的 hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存；将每一个原始数据对应的hash值中未被指定的bit位与该原始数据进行关联并保存；

[0189] 所述多次hash分组模块1032，用于基于当前小组内的原始数据所关联并保存的上一次hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存；将每一个原始数据关联的剩余的未被指定的bit位再次保存。

[0190] 上述具体过程以及执行的原理可参见上述本发明实施例一和本发明实施例二公开的内容，这里不再进行赘述。需要说明的是，分组单元103基于不同的预设分组规则其所执行的内容也有所不同。

[0191] 当所述预设分组规则是预设hash分组次数N时，所述分组单元，用于依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组；

[0192] 当所述预设分组规则是预设分组总数S时，所述分组单元，用于依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组数；

[0193] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设 hash分组次数N的优先级高于预设分组总数S时，所述分组单元，用于依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组；

[0194] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设分组总数S的优先级高于预设hash分组次数N时，所述分组单元，用于依次对每一所述目标数据组中的数据段进行hash分组，直至每一所述目标数据组的分组数等于预设分组总数S；

[0195] 当所述预设分组规则是预设hash分组次数N和预设分组总数S，且预设 hash分组

次数N的优先级和预设分组总数S的优先级一致，所述分组单元，用于依次对每一所述目标数据组中的数据段进行hash分组，直至完成N次hash分组且每一所述目标数据组的分组数等于预设分组总数S；

[0196] 当所述预设分组规则包括预设hash分组次数N，预设的每一次hash分组的分组数m和预设分组总数S时，所述分组单元，用于按照预设的每一次hash分组的分组数进行分组，使得最后的分组次数等于预设hash分组次数，所分的小组的总数等于预设分组总数；

[0197] 其中，N的取值由页表缓冲TLB的存储大小决定，为大于等于1的正整数，N包含n，m小于N；S的取值由数据库缓存cache的大小决定，为大于等于2的正整数；所述预设hash分组次数N与预设分组总数S的优先级由TLB的存储大小和cache的大小决定。

[0198] 上述分组单元103所对应的不同预设分组规则的示例可参见本发明实施例三中给出的示例，这里不再进行赘述。

[0199] 需要说明的是，上述图8中示出的所述划分单元102，其执行过程以及原理与上述本发明实施例四中对应公开的“以矢量vector为数量单位将每一所述目标数据组划分为多个数据段”说明部分相同，这里不再进行赘述，其主要包括：

[0200] 第一划分模块，用于以矢量vector为数量单位，一个vector对应一个数据段，顺序将每一目标数据组划分为M个数据段，M的取值由所述目标数据组内的原始数据的个数，及数据库缓存cache的大小和页表缓冲TLB的存储大小决定；

[0201] 其中，第1至第M-1个数据段中所包含的原始数据的个数相同，第M个数据段中所包含的原始数据的个数小于或等于第1至M-1个数据段中所包含的原始数据的个数。

[0202] 需要说明的是，所述用于将位于指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置，在同一小组内进行排序和保存的所述一次hash分组模块1031，其具体执行过程以及原理可参见上述本发明实施例三中公开的第1次hash详细说明部分，这里不再进行赘述，其主要包括：

[0203] 获取子模块，用于获取当前所述数据段内包含的各个所述原始数据用bit 位表示的hash值；

[0204] 第一查找子模块，用于查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内，其中，依据数据库缓存cache的大小和页表缓冲TLB的存储大小指定当前hash分组所需用到的bit位；

[0205] 第一遍历子模块，用于遍历划分在同一小组内的各个原始数据的下标，所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置；

[0206] 第一排列子模块，用于按照各个下标的大小，从小至大排列各个下标对应的原始数据；

[0207] 第一排序子模块，用于依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0208] 需要说明的是，所述基于当前小组内的原始数据所关联并保存的上一次 hash分组中未被指定的bit位，将当前hash分组过程中指定bit位上取值相同的 hash值所对应的各个原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序和保存的所述多次hash分组模块

1032,其具体执行过程以及原理可参见上述本发明实施例一至实施例四中公开的多次hash分组详细说明部分,这里不再进行赘述,其主要包括:

[0209] 调用子模块,用于调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组中未被指定的bit位;

[0210] 确定子模块,用于从调用的所述未被指定的bit位中确定当前hash分组过程中所需用到的bit位,其中,当前hash分组过程中所需用到的bit位依据数据库缓存cache的大小和页表缓冲TLB的存储大小决定;

[0211] 第二查找子模块,用于查找当前hash分组过程中指定bit位上取值相同的 hash值对应的各个原始数据,将各个原始数据划分在同一小组内;

[0212] 第二遍历子模块,用于遍历划分在同一小组内的各个原始数据的下标,所述各个原始数据的下标用于标识各个原始数据在所述目标数据组中的位置;

[0213] 第二排列子模块,用于按照各个下标的大小,从小至大排列各个下标对应的各个原始数据;

[0214] 第二排序子模块,用于依据所述从小至大的顺序将各个原始数据写入同一小组内并保存。

[0215] 需要说明的是,所述连接单元105,其具体执行过程以及原理可参见上述本发明实施例四中公开Join操作的详细说明部分,这里不再进行赘述,其主要包括:

[0216] 获取模块,用于按顺序分别获取所述待连接的两个目标数据组进行N次 hash分组后的各个小组;

[0217] Join模块,用于两两小组为一对进行原始数据Join操作的方式,对两个目标数据组的各个小组中原始数据进行Join操作;

[0218] 其中,所述Join模块包括:

[0219] 第三遍历子模块,用于由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组;若遍历到相同小组时,执行第一Join子模块;若未遍历到相同小组时,则移动至下一小组返回所述第二遍历子模块;直至所述目标数据组中的所有小组对另一目标数据组中的各个小组都执行遍历操作;

[0220] 所述第一Join子模块,用于将进行遍历的所述小组中的原始数据,顺序与所述相同小组内的原始数据进行Join操作,其中,所述相同小组是指该小组内存储的原始数据的hash值与用于遍历的小组内存储的原始数据的hash值相同;当所述小组中的原始数据都已进行执行Join操作后,移动至下一小组返回所述第三遍历子模块。

[0221] 本发明实施例五公开对应执行上述哈希连接方法的哈希连接装置,基于上述公开的各个单元以及模块,在对目标数据组执行hash分组的过程中,以一个vector为数量单位按组计算hash值,然后将同一个分组中所包含的若干个原始数据对应的hash值一次性写入对应的分组中。以vector的形式进行分组则能够避免产生不必要的cache thrashing,从而实现减少了cache miss,提升Join性能的目的。并且,仅在第1次分组过程中计算各个原始数据的hash值,而将后续使用到的若干bit位记录到对应的各个原始数据的关联位置处以备后续分组过程中直接使用,从而省去重复计算hash值的代价,避免了资源浪费。

[0222] 同时,在进行hash分组的过程中,在每一次hash分组之后,将原始数据写入各个对应的小组内之前,对各个小组内的原始数据进行排序,最终在各个小组做最后的排序时,仅

需要对各个小组进行排序即可。通过该种方式，能够实现大大的降低现有技术中在分组完成之后，再对各个小组内的原始数据以及各个小组进行排序的复杂度，减少因排序消耗的时间的目的。

[0223] 结合本发明公开的实施例描述的哈希连接方法，在数据管理系统中可以直接用硬件、处理器执行的存储器，或者二者的结合来实施。因此，本发明还对应上述本发明实施例公开的方法和装置公开了一种数据管理系统。下面给出具体的实施例进行详细说明。

[0224] 如图9所示，该数据管理系统1包括存储器11和通过总线12与存储器11连接的处理器13。

[0225] 该存储器11具有存储介质，该存储介质中存储有进行数据库查询时的程序。

[0226] 存储器11可能包含高速RAM存储器，也可能还包括非易失性存储器，例如至少一个磁盘存储器。

[0227] 该处理器13通过总线13与存储器11连接，当执行数据库查询时，该处理器13调用存储器11中存储的数据库查询程序。上述数据库查询程序可以包括程序代码，所述程序代码包括一系列按照一定顺排列的操作指令。处理器13 可能是一个中央处理器CPU，或者是特定集成电路，或者是被配置成实施本发明实施例的一个或多个集成电路。

[0228] 处理器13调用的进行数据调度的程序具体可以包括：

[0229] 接收包含有连接Join操作的结构化查询语言SQL语句，解析获取至少两个待连接的目标数据组；

[0230] 以矢量vector为数量单位将每一目标数据组划分为多个数据段；

[0231] 基于预设分组规则依次对每一目标数据组中的数据段进行N次哈希hash 分组，其中，在每次hash分组时，基于第1次hash分组计算所述数据段中的原始数据所得的用bit位表示的hash值，将当前hash分组过程中指定bit位上取值相同的hash值所对应的原始数据划分在同一小组内，并对划分在同一小组内的各个原始数据，按照各个原始数据在所述目标数据组中的位置在同一小组内进行排序并保存，N取大于或等于1的正整数；

[0232] 对每一目标数据组经过N次hash分组后获得的小组，在所述目标数据组中，按照各个小组中所包含的原始数据对应的hash值由小至大的对各个小组进行排序；

[0233] 按照排序依次取所述两个待连接的目标数据组中经由N次hash分组后获得的各个小组中的原始数据进行Join操作。

[0234] 综上所述：

[0235] 本发明实施例公开通过以vector为数量单位以及在后续分组过程中利用前一次hash分组过程中未被指定bit位继续执行hash分组，能够实现同时对若干原始数据进行hash分组处理，且在多次hash分组的过程中不需要重复计算原始数据的hash值，即减少了cache miss缓存缺失，也省去了重复计算hash值避免了计算资源的浪费。同时，在进行分组的过程中，按照所述原始数据在目标数据组中的位置，对划分在同一小组中的原始数据进行排序，实现降低后续对各个分组进行排序的复杂度的目的。并且每次分组划分至每个小组中的原始数据有序，这样完成多个数据段分组后得到的每个小组中的原始数据局部有序，在对局部有序的原始数据进行join时，其排序复杂度低于随机分配的原始数据进行join时的排序复杂度。

[0236] 本说明书中各个实施例采用递进的方式描述，每个实施例重点说明的都是与其他

实施例的不同之处，各个实施例之间相同相似部分互相参见即可。对于实施例公开的装置而言，由于其与实施例公开的方法相对应，所以描述的比较简单，相关之处参见方法部分说明即可。结合本文中所公开的实施例描述的方法或算法的步骤可以直接用硬件、处理器执行的软件模块，或者二者的结合来实施。软件模块可以置于随机存储器(RAM)、内存、只读存储器(ROM)、电可编程ROM、电可擦除可编程ROM、寄存器、硬盘、可移动磁盘、CD-ROM、或技术领域内所公知的任意其它形式的存储介质中。

[0237] 对所公开的实施例的上述说明，使本领域专业技术人员能够实现或使用本发明。对这些实施例的多种修改对本领域的专业技术人员来说将是显而易见的。

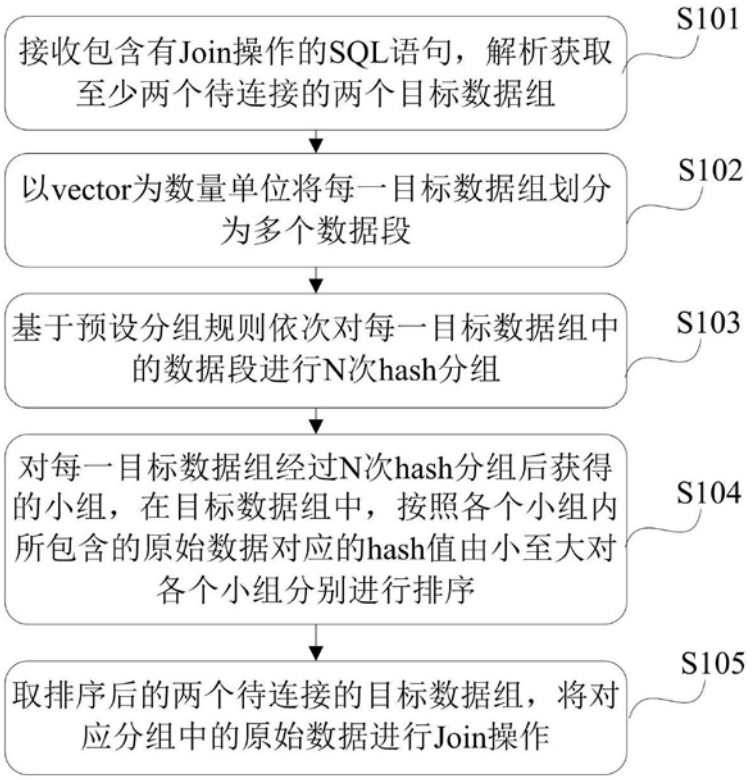

S101
接收包含有Join操作的SQL语句，解析获取至少两个待连接的两个目标数据组
S102
以vector为数量单位将每一目标数据组划分为多个数据段
S103
基于预设分组规则依次对每一目标数据组中的数据段进行N次hash分组
S104
对每一目标数据组经过N次hash分组后获得的小组，在目标数据组中，按照各个小组内所包含的原始数据对应的hash值由小至大对各个小组分别进行排序
S105
取排序后的两个待连接的目标数据组，将对应分组中的原始数据进行Join操作

图1

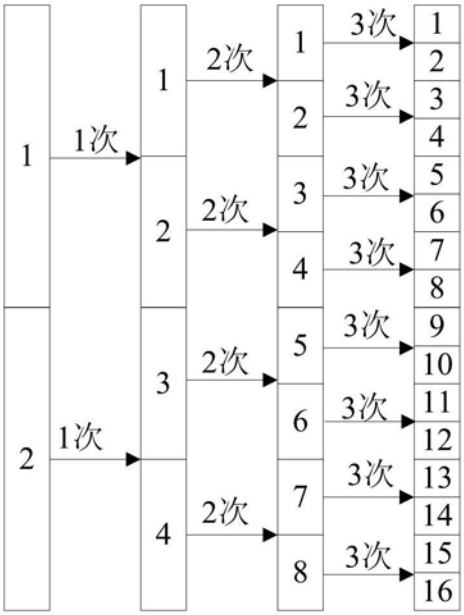

1
2
1次
1次
1
2
3
4
2次
2次
2次
2次
1
2
3
4
5
6
7
8
3次
3次
3次
3次
3次
3次
3次
3次
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

图2


一个vector
1
2
3
4
5
……
1
2
3
4
5
第1个数据段
第2至第4个数据段
第5个数据段

图3

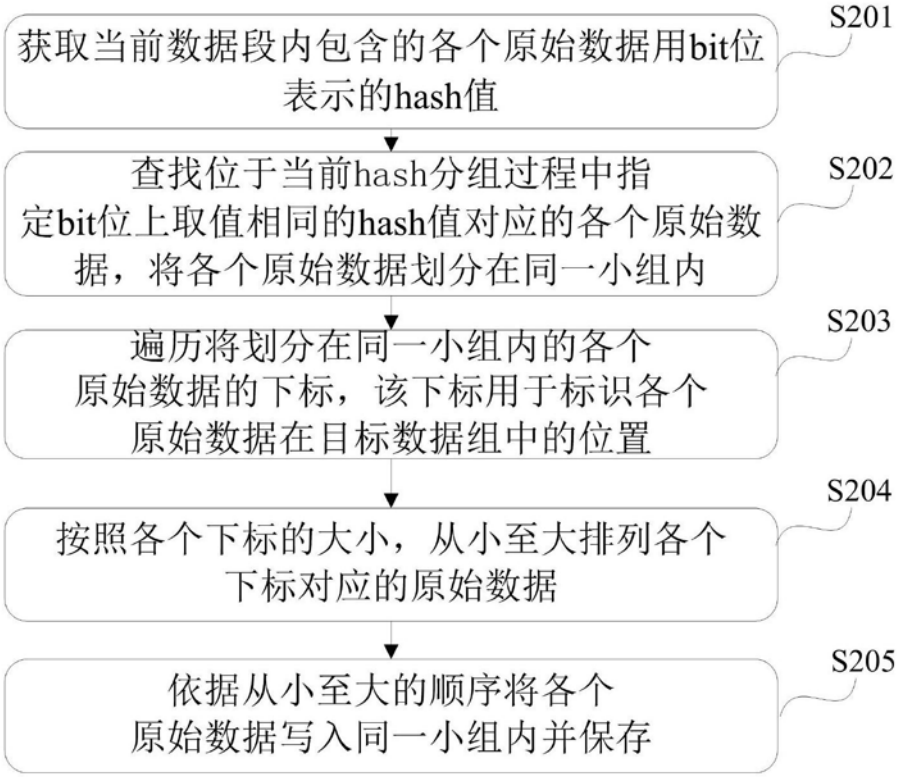

S201
获取当前数据段内包含的各个原始数据用bit位表示的hash值
S202
查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的各个原始数据，将各个原始数据划分在同一小组内
S203
遍历将划分在同一小组内的各个原始数据的下标，该下标用于标识各个原始数据在目标数据组中的位置
S204
按照各个下标的大小，从小至大排列各个下标对应的原始数据
S205
依据从小至大的顺序将各个原始数据写入同一小组内并保存

图4

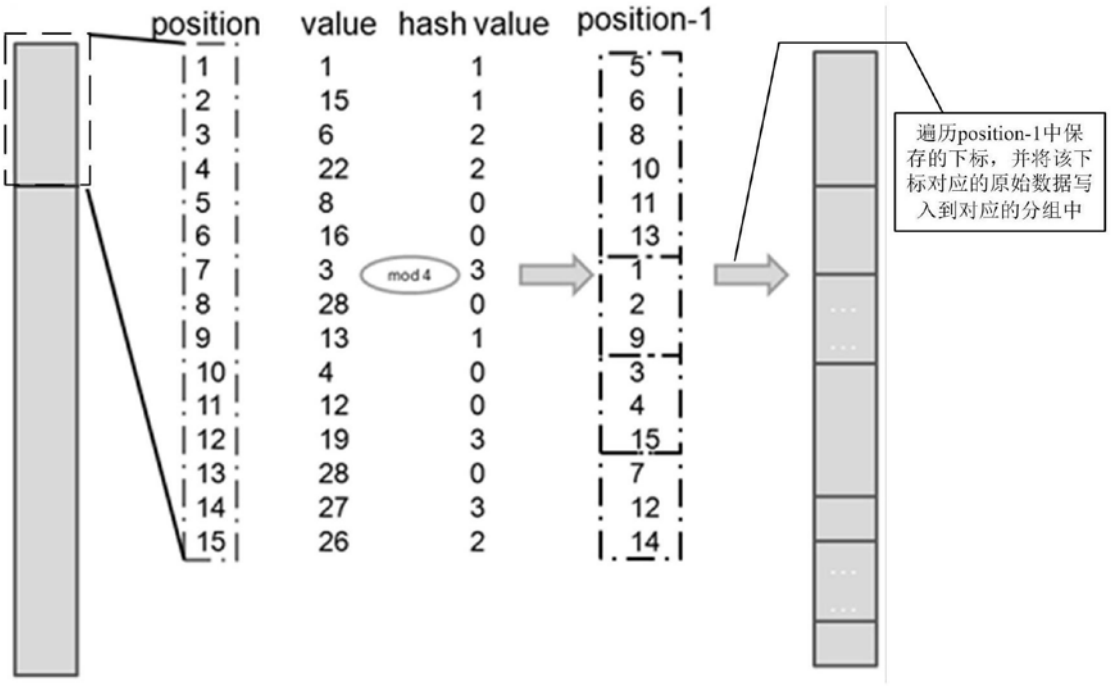

position
value
hash value
position-1
1 1 1 5
2 15 1 6
3 6 2 8
4 22 2 10
5 8 0 11
6 16 0 13
7 3 3 1
8 28 0 2
9 13 1 9
10 4 0 3
11 12 0 4
12 19 3 15
13 28 0 7
14 27 3 12
15 26 2 14
mod 4
遍历position-1中保存的下标，并将该下标对应的原始数据写入到对应的分组中

图5

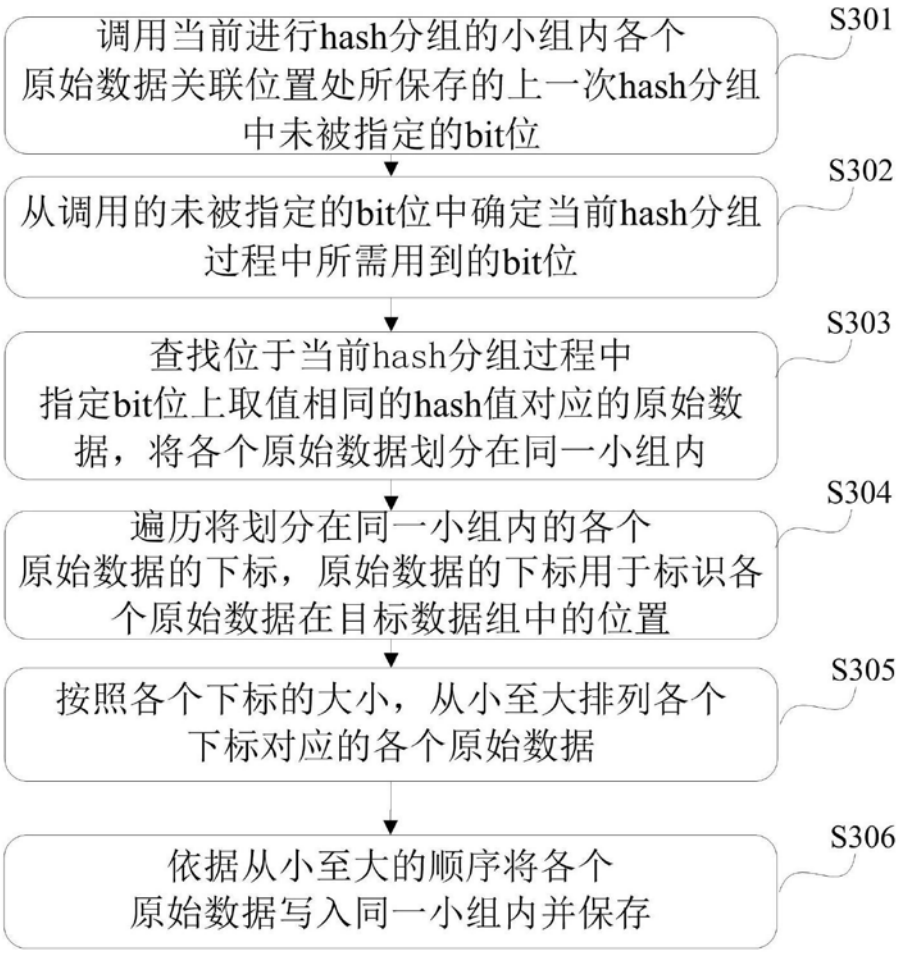

S301
调用当前进行hash分组的小组内各个原始数据关联位置处所保存的上一次hash分组中未被指定的bit位
S302
从调用的未被指定的bit位中确定当前hash分组过程中所需用到的bit位
S303
查找位于当前hash分组过程中指定bit位上取值相同的hash值对应的原始数据，将各个原始数据划分在同一小组内
S304
遍历将划分在同一小组内的各个原始数据的下标，原始数据的下标用于标识各个原始数据在目标数据组中的位置
S305
按照各个下标的大小，从小至大排列各个下标对应的各个原始数据
S306
依据从小至大的顺序将各个原始数据写入同一小组内并保存

图6

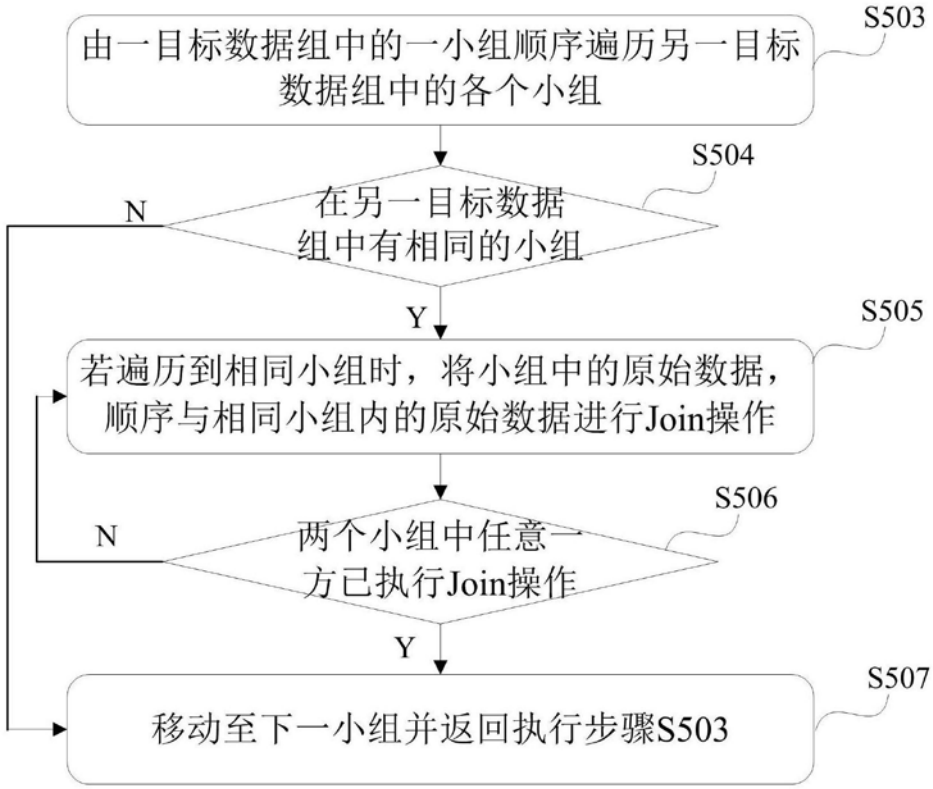

由一目标数据组中的一小组顺序遍历另一目标数据组中的各个小组
S503
在另一目标数据组中有相同的小组
S504
N
Y
若遍历到相同小组时，将小组中的原始数据，顺序与相同小组内的原始数据进行Join操作
S505
两个小组中任意一方已执行Join操作
S506
N
Y
移动至下一小组并返回执行步骤S503
S507

图7

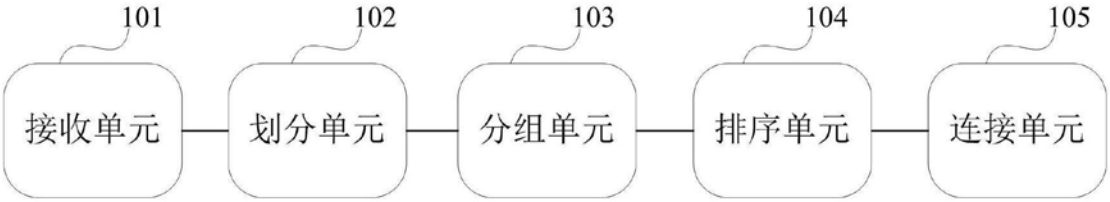

101
接收单元
102
划分单元
103
分组单元
104
排序单元
105
连接单元

图8

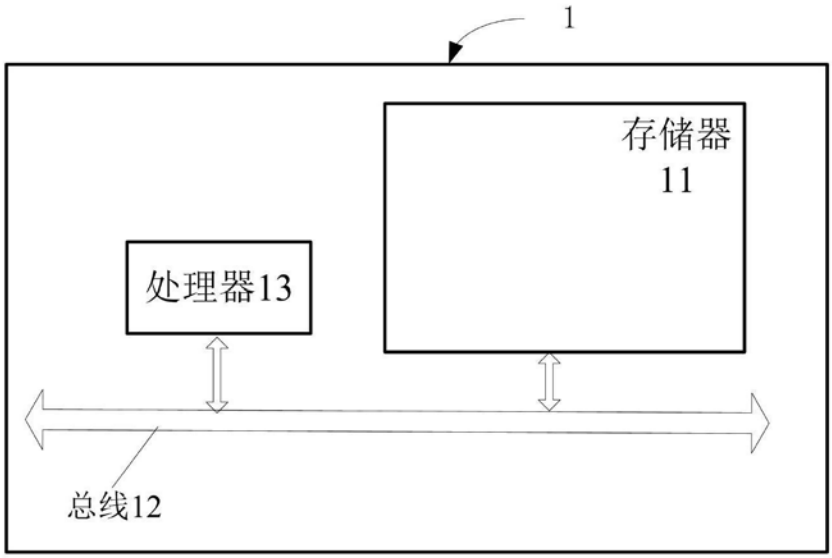


图9