数据库第十章——第十三章

第十章：存储和文件结构

RAID磁盘阵列

RAID级别

RAID0:块级拆分但没有任何冗余。4块。

RAID1:使用块级拆分的磁盘镜像。8块（复制了4块）。

RAID2:内存风格的纠错码组织结构，奇偶校验。（7块）。

RIAID3:位交叉的奇偶校验组织结构。（5块）。

RAID4:块交叉的奇偶校验组织结构。（5块）。

RAID5:块交叉的分布奇偶校验。（5块）。

RAID6:P+Q冗余。

第十一章：索引和散列

索引是index_field(s) + pointer，提高查询效率，缺点是维护索引有额外开销，需要建立合适的索引。

总体上可以分为顺序索引、散列索引和位图索引。

顺序索引，常用B+树实现

（1）聚集索引和非聚集索引：聚集索引（主索引clustering index\primary index）只有一个，主文件按某个搜索码排序，该搜索码对应的索引是聚集索引，搜索码不一定是主键（即聚集索引的物理顺序与索引顺序相同）。

建表时的主键必须是索引，默认是聚集索引，根据聚集索引插入数据。

非聚集索引（辅助索引）的物理顺序与索引顺序不同。

（2）稠密索引（dense index）和稀疏索引(sparse index)：稠密索引是文件中的每个搜索码

值都有一个索引项；稀疏索引是只为某些搜索码值建立索引项。

只有文件是聚集索引（排好序）排好顺序时才能使用稀疏索引。

聚集索引可以是稀疏索引；非聚集索引一定是稠密索引。

（3）多级索引：如B+树索引

（4）倒排索引：便于查找一个关键字出现在了哪些文件中。

散列索引，不能进行范围查找

（1）静态散列索引：就是一张散列表，每个entry包含索引项+指针，用链表等方法解决冲突。缺点是数据变多后需要rehash

（2）动态散列索引：取散列值的前i位做索引，有指向对应桶的指针，桶有多个指向文件的指针

位图索引

可以用位运算实现高效查询。例如gender的位图m=10010，f=01101表示0,3记录为男性，1,2,4为女性

B-树

一个m阶的B树具有如下几个特征：

1.根结点至少有两个子女。

2.每个中间节点都包含k-1个元素和k个孩子，其中 m/2 <= k <= m

3.每一个叶子节点都包含k-1个元素，其中 m/2 <= k <= m

4.所有的叶子结点都位于同一层。

5.每个节点中的元素从小到大排列，节点当中k-1个元素正好是k个孩子包含的元素的值域分划。

一个m阶的B+树具有如下几个特征：

1.有k个子树的中间节点包含有k个元素（B树中是k-1个元素），每个元素不保存数据，只用来索引，所有数据都保存在叶子节点。

2.所有的叶子结点中包含了全部元素的信息，及指向含这些元素记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。

3.所有的中间节点元素都同时存在于子节点，在子节点元素中是最大（或最小）元素。

B+树的优势：

1.单一节点存储更多的元素，使得查询的IO次数更少。

2.所有查询都要查找到叶子节点，查询性能稳定。

3.所有叶子节点形成有序链表，便于范围查询。

第十二章——查询处理

参考：数据库系统概念－查询处理 - zeroArn - 博客园

传送磁盘块数和搜索磁盘次数

一个磁盘子系统传输一个块的数据时间为tT

磁盘块平均访问时间（磁盘搜索时间加上旋转延迟）为tS

一次传输b个块以及执行S次磁盘搜索的操作将消耗b*tT+
S*tS

选择运算

- A1：线性搜索，平均代价Br/2，最坏情况Br
- A1: 二分搜索，属性有序，代价[logBr]

索引选择

- A2: (主索引，码属性等值比较)可以检索到唯一一条满足条件的记录，代价：B+树树高加上读取一条记录I/O代价

- A3: (主索引，非码属性等值比较)主索引可以检索到多条满足条件的记录，且多条记录顺序存储，代价：B+树树高加上具有搜索码值的盘块数

- A4: (辅助索引，等值比较)索引字段为码属性直接得到一条记录 · 索引字段为非码属性，得到多条记录

比较选择

- A5: (主索引，比较)B+树有序主索引

- A6: (辅助索引，比较)有序辅助索引，小值从小段开始，大值从　大端开始

复杂选择的实现

- A7: （利用一个索引的合取选择）先选择满足一个条件的记录，加入缓冲区，在缓冲区中验证其他条件。通过组合不同的条件和算法使消耗最小。

- A8: (使用组合索引所用合取) 如果组合索引和要合取的内容相同，直接利用合适的组合索引查询，使用A2、A3、A4中的

- A9: (通过标识符的交实现合取选择)每个条件遍历标记一边，取所有被标记的交集。 代价：扫描各个单独索引代价的总和加上获取检索到的指针列表的交集中的记录的代价。因此应该把指向同一个磁盘块的指针归并到一起，这样只需要一次I/O即可。

- A10: （通过标识符的并实现析取选择）逐一扫描索引获取满足单个条件的元祖指针，将所有指针集做并集。

!查询处理时间消耗

排序

外部排序归并算法

对不能全部存放在内存中的关系的排序叫外排序，最常用的是外排序归并算法。M表示内存缓冲区可以用于排序的块数，即内存的缓冲区能容纳的磁盘块数。

1.建立多个排好序的归并段，每一段都是排序过的

2.对归并段进行归并，对N个归并段归并，叫做N路归并。

注释每一趟归并有最多有M-1个归并段参与

外部排序归并的代价

Br代表包含关系r中记录的磁盘块数目。第一阶段读入关系的每一块数据并写出，共需2br次磁盘块运输，初始归并段数是【br*M*】，由于每一趟归并让归并段数目减少为原来的【*1*M-1】，因此总共需要的归并趟数为【logm-1（br*M*）】。对于每一趟归并，关系的每一数据块读写各一次，最后一次可以不写入磁盘，关系外排序的磁盘块传输的总数为【*br*（*2[logm-1*（*br*M）]+1）】。

此外，算上磁盘搜索的代价，在归并阶段，如果每次从一个归并段读取bb块数据，则每一趟归并需要【br*bb*】次磁盘搜索以读取数据，读写间隔可能头移动到别处，加上*2*【*br*bb】次磁盘搜索，除了最后一趟以外。假设输出阶段也分配了bb个块，每一趟可以归并【M/bb】-1个归并段，则搜索的总次数为：
2【br*M*】+【*br*bb】（2[logm-1（br/M）]-1）

连接运算

- 嵌套循环nested loop join（重名属性会出现）别从两个表读一行数据进行两两对比，复杂度是n^2

- 块嵌套循环block nested loop join，每次在块内循环嵌套检查元组匹配，有效减少比较次数。分别从两个表读很多行数据，然后进行两两对比，复杂度也是n^2，只是少了些函数调用等overhead

- 索引嵌套循环连接index nested loop join：嵌套循环连接的内层如果有索引，使用索引代替循环；从第一个表读一行，然后在第二个表的索引中查找这个数据，索引是B+树索引，复杂度可以近似认为是nlogn，比上面两个好很多，这就是要保证关联字段有索引的原因

  散列连接hash join： 用第一个表（小表）建hash table，第二个表在hash table中查找匹配的项，复杂度是n。缺点是hash table占的内存可能会比较大，不过也有基于磁盘的hash join，实现起来比较复杂

- 归并连接：用于自然链接和等值连接
  - 双有序直接有序归并
  - 单有序，归并后，对索引项进行按地址排序，可实现有序

- 消除重复: 代价高，明确声明是否去重
  - 归并/散列可直接在过程中消除重复

- 集合
  - 并集：Hr建立散列索引，将Hs中元组加入到上诉散列索引中，条件是该元组不在散列索引中，散列索引最终即结果

  - 交集：Hr建立散列索引，对Hs中元组探查散列索引，出现在散列索引的记录放入结果

  - 差集：Hr建立散列索引，对Hs中元组探查散列索引，出现在散列索引的记录从索引中删除

外连接

计算连接法：

左外连接：计算链接，将左集未参与链接集合扩展放入结果

右外连接：计算链接，将右集未参与链接集合扩展放入结果

全外连接：计算链接，将左右集未参与链接集合扩展放入结果

嵌套修改法

左外连接：左边外循环，右表内循环，内外匹配加入结果，内部均不匹配的外记录也加入结果

右外连接：右边外循环，左表内循环，内外匹配加入结果，内部均不匹配的外记录也加入结果

扩展归并连接获得自然链接和等值全外联。 归并完成，将不与另一关系任一记录匹配的记录加入结果

表达式计算

- 实体化：中间结果实体化，供下一层运算，磁盘代价高，双缓冲降低磁盘代价

- 流水线：生产者驱动流水线，需求驱动流水线