原 数据库索引（B树，B+树，哈希）

2018年05月05日 13:39:49 李子树呢 阅读数：1655

CSDN 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qq_32483145/article/details/80191323

> 数据库索引是存储引擎用于快速找到记录的一种数据结构。 《高性能MySQL》

一. 什么是索引?

索引的目的就是便于快速查找。一本书的索引就是目录，可以让我们快速定位到要查找的内容；数据库的数据是以记录的方式存在的，所以索引的目的就是便于查找某一些记录。

索引类型(常见的数据库书籍中的关于索引类别的一些称呼):

①唯一索引：不允许其中任何两行具有相同值的索引

使用主键和候选键建立的索引就是唯一索引，因为主键和候选键都可以确定唯一一个元组，即一张表中不存在相同的主键和候选键。在MySQL中，当你建立一个主键和候选键之后，MySQ立索引。毕竟要想满足唯一性，依然会在更新数据的时候检验是否已经存在该主键或者候选键，而索引无疑是快速检验的标配。

②.主键索引：可以认为是特殊的唯一索引，仅利用主键建立的索引

③.单一索引：任何一个单一数据项建立的索引

假如有下表【country，city，personNumber】，如果我们想查询某个国家的人数，我们就应当以国家建立索引，这样单一数据项建立的索引就是单一索引。

④.复合索引：多个数据项建立的索引

假如有下表【country，city，personNumber】，如果我们想查询某个城市的人数，我们就应当以【country，city】建立索引，多个数据项建立索引的时候，我们应当指定其排序的先后顺优先排序，城市次之。

⑤.聚簇索引：利用主键建立的索引，其物理存放顺序与主键顺序一致。因为数据只有一个物理存放顺序，所以一个表只有一个聚簇索引。

在MySQL中，选定主键之后将会自动为主键创建索引。该索引可以维护主键的唯一性。非叶子节点包含了主键值，而叶子节点则指向了一条完整的记录

⑥.非聚簇索引(二级索引，辅助索引)：除了聚簇索引之外，其余所有的索引都是非聚簇索引

非聚簇索引为什么是二级索引呢？重点在于一个二字。可以料想如果WHERE条件不是根据主键进行索引，那么我们就需要基于该非主键建立的索引进行检索，这样建立的索引叶子节点中包再使用主键在聚簇索引中找寻到完整的记录。可以说进行了两次B+ Tree查找而不是一次

⑦.覆盖索引：一个索引包含(覆盖)所要查询的字段的值，注意覆盖索引与具体的查询有关

假如有下表【country，city，personNumber】，如果我们想查询某个城市的人数，覆盖索引指的是可以将你想要查询的列建立在一个索引中，如（国家，城市，人数）作为一个复合索引国家的所有城市利用索引就可以完成，实际上这也相当于完成了聚簇索引的功能

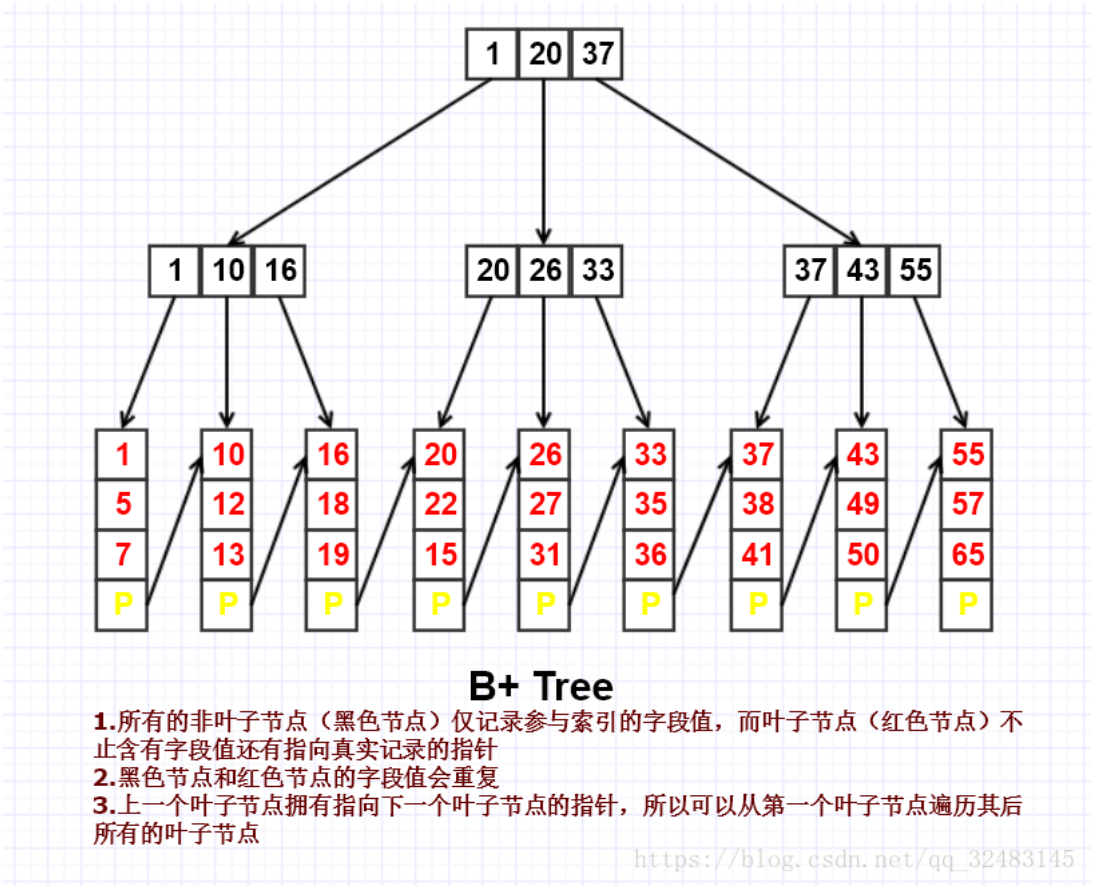
二. 如何快速找到记录?

说到查找算法，最朴实的就是无序排列的遍历了。在数据量较小的时候（PS：如果数据量较小我想数据库也不会有这么快的发展），此方法也不失为一种好的方法，毕竟没有相应数据结构的维护的时候就可以再快一点。在学习算法的书籍中，数据结构总也是扮演着重要的角色。无序排列的数据简直就是一种佛系的数据结构，不管不顾它就是那样。一个好的算法基于一个合适的数据结构的例子基于堆这种数据结构；二叉搜索基于一棵构建好的二叉搜索树，哈希查找基于哈希表……所以数据库中为了快速查找到需要的记录也需要选择合适的数据结构。

三. 什么样的数据结构适合作为索引?

下面就介绍几种数据库中快速查找记录的数据结构：

Ⅰ. B+ Tree索引(MySQL，SQL Server，Oracle)



以上为一个3阶的B+ Tree，其上的数字我们可以认为使用ID建立起来的单一索引。如果需要使用如下SQL语句进行查询：

```
SELECT * FROM STUDENTS WHERE ID=1
```

这个查询语句只需要三次查找就可以找到ID为1的叶子节点，找到存放该条记录（存放了ID为1的学生的所有属性）的物理地址，进而找到该条数

B+ Tree索引优点

①.全值匹配：指的是和索引中所有列进行匹配。假设以(姓，名，出生日期)三个数据项建立复合索引，那么可以查找姓名为张三，出生日期在2000-12-12的人

②.匹配最左前缀：假设以(姓，名，出生日期)三个数据项建立复合索引，可以查找所有姓张的人

③.匹配列前缀：假设有姓为司徒，司马的人，我们也可以查找第一列的前缀部分，如查找所有以司开头的姓的人

④.匹配范围值：可以查找所有在李和张之间的姓的人，注意范围查询只在复合索引的优先排序的第一列。（假设姓名按照拼音排序）

⑤.精确匹配前面列并范围匹配后一列：可以查找姓李并出生日期在2000-12-12之后的人或姓名为张三并出生日期在2000-12-12之后的人，注意范围第一个范围查询后面的列无法

⑥.只访问索引的查询：即查询只需访问索引，而无需访问数据行。（此时应想到索引中的覆盖索引）

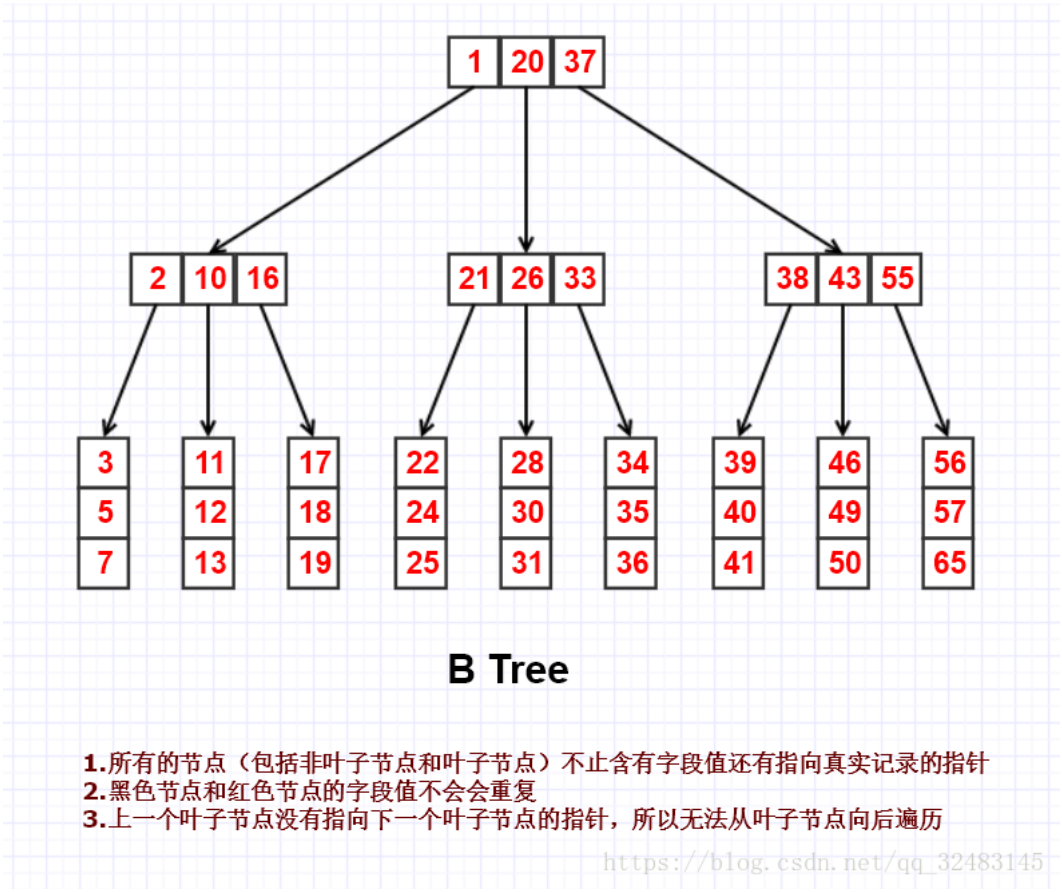
B+ Tree索引缺点

①.如果不是按照索引的最左列开始查找，则无法使用索引。如无法查找名为龙的人，也无法查找在2000-12-12之后出生的人，当然也无法查找姓中以龙结尾的人（注意为和含有的

②.不能跳过索引中的列：无法查找姓李并在2000-12-12之后出生的人

③.如果查询中包括某个列的范围查询，则其右边所有列都无法使用索引优化查询

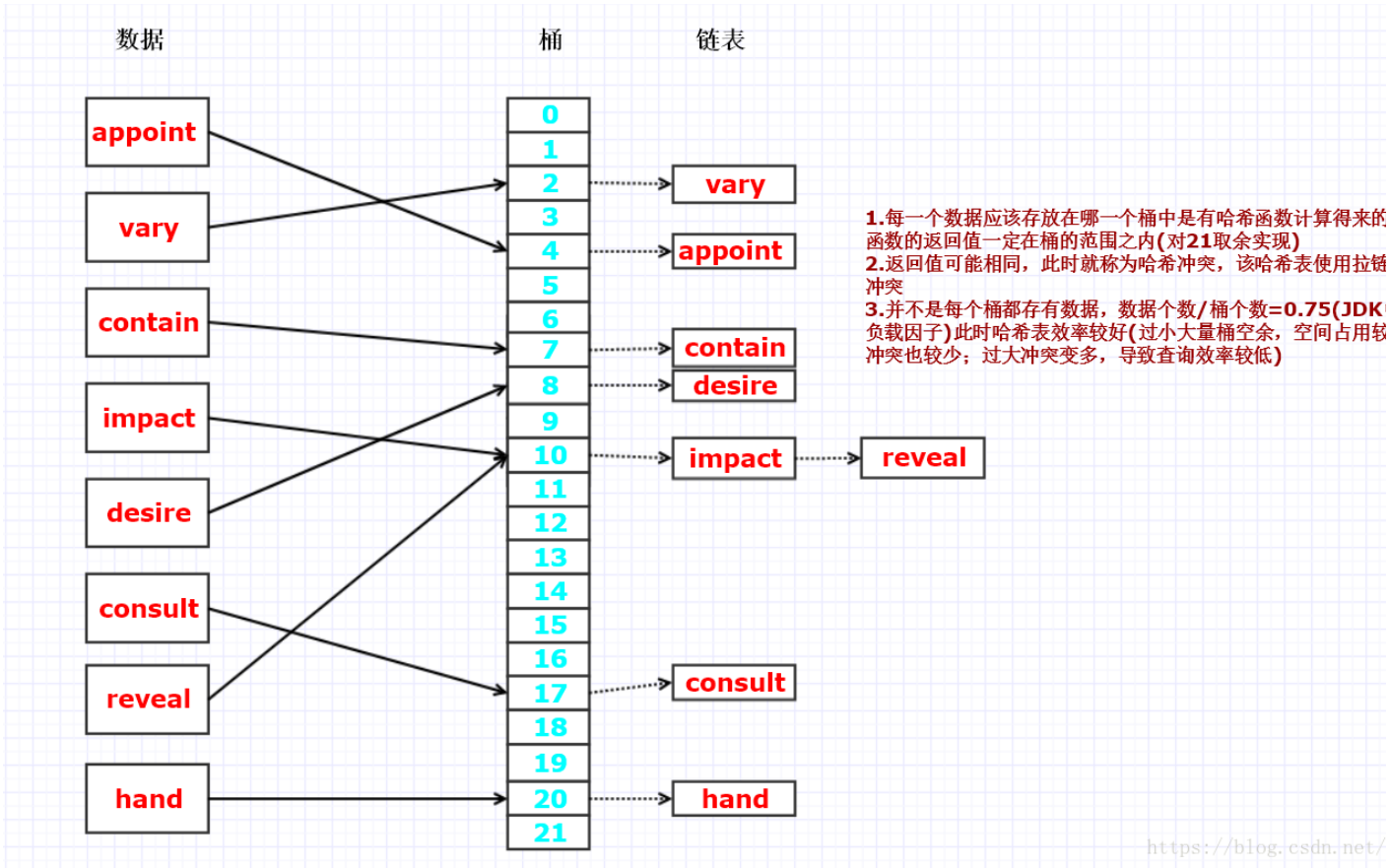
Ⅱ. B Tree索引



1.所有的节点（包括非叶子节点和叶子节点）不止含有字段值还有指向真实记录的指针
2.黑色节点和红色节点的字段值不会会重复
3.上一个叶子节点没有指向下一个叶子节点的指针，所以无法从叶子节点向后遍历

https://blog.csdn.net/qq_32483145

Ⅲ.哈希索引(MySQL，Oracle)



1.每一个数据应该存放在哪一个桶中是有哈希函数计算得来的
函数的返回值一定在桶的范围之内(对21取余实现)
2.返回值可能相同，此时就称为哈希冲突，该哈希表使用拉链
冲突
3.并不是每个桶都存有数据，数据个数/桶个数=0.75(JDK
负载因子)此时哈希表效率较好(过小大量桶空余，空间占用较
冲突也较少；过大冲突变多，导致查询效率较低)

https://blog.csdn.net/

哈希索引优点

①.快速查询：参与索引的字段只要进行Hash运算之后就可以快速定位到该记录，时间复杂度约为1

哈希索引缺点

①.哈希索引只包含哈希值和行指针，所以不能用索引中的值来避免读取行

②.哈希索引数据并不是按照索引值顺序存储的，所以也就无法用于排序和范围查询

③.哈希索引也不支持部分索引列查询，因为哈希索引始终是使用索引列的全部数据进行哈希计算的。

④.哈希索引只支持等值比较查询，如=，IN()，<=>操作

⑤.如果哈希冲突较多，一些索引的维护操作的代价也会更高

参考资料：

《高性能MySQL 第三版》 Baron Schwartz / Peter Zaitsev / Vadim Tkachenko著

0 0条评论 收藏 分享 下一篇