MySQL 索引 & B+ 树

作者：someone33881
原文链接：https://ld246.com/article/1547799336047
来源网站：链滴

一、背景

    在程序员面试的世界
，凡是涉及到数据库mysql，基本都会问索引，而问到索引更深入一点就都会涉及到B+树，因此本文定对B+树这样一种数据结构进行较为详细的学习！

二、MySQL索引

1、索引类型

- 主键索引|primary：不允许为null
- 普通索引|normal：普通非唯一索引
- 唯一索引|unique：表示唯一的，不允许重复的索引，可以为null
- 全文索引|fulltext：全文搜索的索引，用于搜索很长一篇文章时效果最好
- 空间索引|spatial：

2、索引结构

- 哈希索引：底层数据结构即是哈希表，对于绝大数需求为单条记录查询的性能最快
- BTree索引：mysql中BTree索引使用的是B树中的B+Tree，只是在两种主要的存储引擎中的实现方不同而已(在适用中存在差别)

3、MyISAM B+Tree VS InnoDB B+Tree

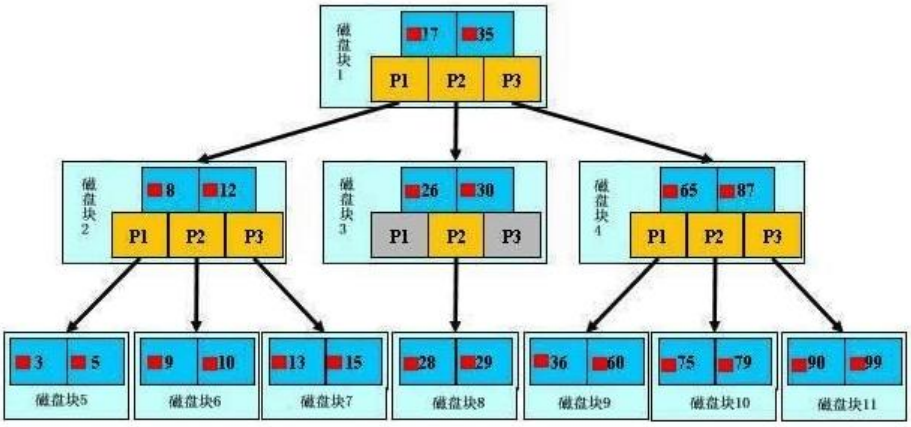
- MyISAM在B+Tree的叶节点存储关键字外，还存储对应数据的存放地址；主索引和辅助索引没什区别，只是主索引中的关键字一定是唯一的 =》称之为非聚簇索引

- InnoDB在B+Tree的叶节点不仅存储关键字，同时也将对应数据存放在那儿（数据物理存放顺序与引顺序一致） =》称之为聚簇索引
- =>对于大数据量的排序、全表扫描、count之类操作，MyISAM由于索引所占空间小，且这些操作是需要在内存中完成的，因此MyISAM更具有优势
- 一言以蔽之， **“MyISAM中B+Tree叶节点的data域存放的是数据记录的地址，索引文件和数据件是分离的”，而“InnoDB数据文件本身就是索引文件，其B+Tree叶节点data域保存的是完整的据记录”**

三、B+Tree

1、数据结构

- B+Tree是由一个个磁盘块组成的树形结构，每个磁盘块均由数据项和指针组成；
- 所有的数据均是存放在叶子节点的，非叶子节点不存放数据



2、查找

- 或者从最小关键字起顺序查找
- 或者从根结点开始，进行随机查找

在查找时，若在非叶子节点上的关键值等于给定值，并不会终止，而是会继续向下直到叶子节点！也，在B+树中，无论查找成功与否，每次查找均是一条从根到叶子节点的路径！

四、BTree

BTree即是B树（不要读成B-树而丢人现眼啊！！！）

数据库之所以使用树结构进行存储，出发点当然是因为的树的查询效率到且可以保持有序，但是为什不是使用二叉查找树呢？（二叉查找树的时间复杂度是O(logN),从算法逻辑上讲，二叉查找树的查找度和比较次数都是最小的 => 但是，在数据库存储的查找时不得不考虑的一个因素是“磁盘IO”

=> 数据库索引是存储在磁盘中的，在数据量比较大的时候，索引的大小可能会达到几个G甚至更大

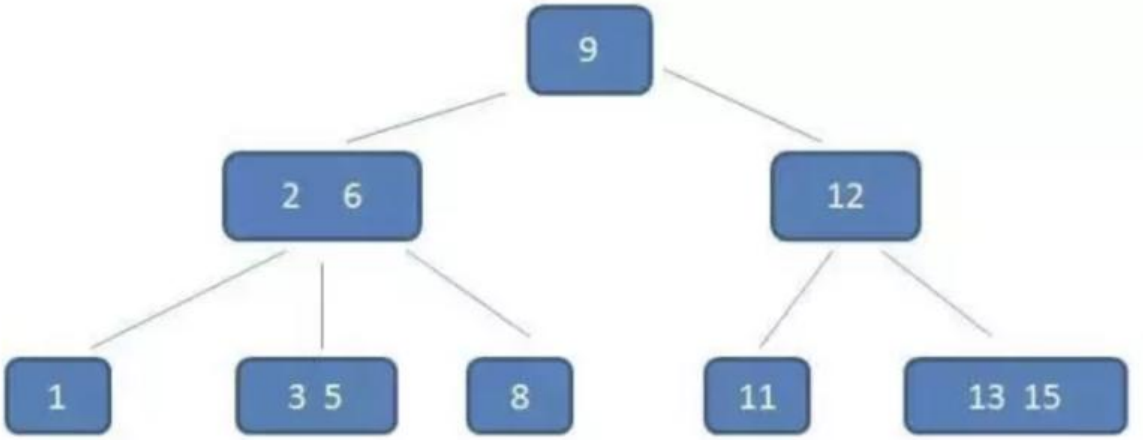
=> 因此，是不可能把整个索引都加载到内存中的，只能是通过逐一加载磁盘页（也即对应索引树中节点）

=> 因此，磁盘IO的次数，在最坏的情况下是等于树的高度的 ==》于是，为了减少磁盘IO的次数，要降低树的高度，也即将“瘦高”的树结构变成“矮胖”，这也是B树的特征之一）

B树是一种多路平衡查找树，每一个节点最多包含k个孩子，k称之为B树的阶；k的大小取决于磁盘页大小

1、数据结构 - m阶B树

- 每个节点最多有m-1个关键字
- 根节点至少有1个关键字，非根节点至少有Math.ceil(m/2) - 1个关键字
- 每个节点中的关键字都是按照从小到大排列，每个关键字的左子树中的所有关键字都小于它，而右树中的所有关键字都大于它
- 所有叶子节点均位于同一层，或者说根节点到每个叶子节点的长度都相同



=>>B树在查询过程中的比较次数，并不比二叉查找树少，但是由于单一节点中的元素不止一个元素其是当单一节点中的元素很多时，即多个元素在同一个磁盘页中时却只需要一次磁盘IO（仅是在内存做多次比较而已） =>相比较于磁盘IO的速度，内存中的比较耗时几乎可以忽略 =>因此，只要树高度足够低，IO次数足够少，查询性能就会提升

==》因此，相比较之下，即使同一个节点内元素多一些也没多大影响，仅仅是多了几次内存交互，要是不超过磁盘页的大小即可！

2、B树的插入和删除

五、总结

- B树：有序数组 + 多路平衡查找树
- B+树：有序数组链表（叶字节点构成链表） + 多路平衡查找树
- B*树：一棵丰满的B+树