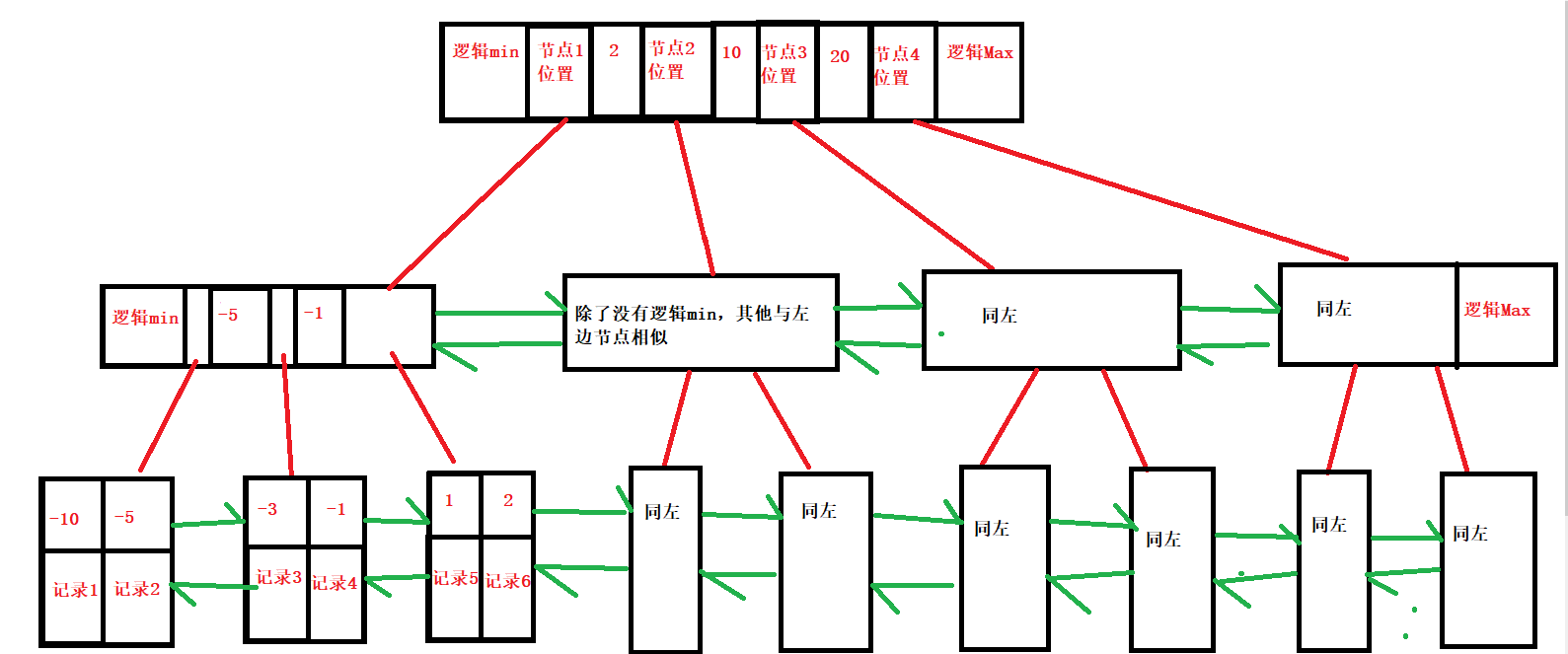
首先先介绍一下索引的B+树结构：



最上面一层为B+树的根节点

第二层是子节点

第三层是叶节点（记录1、记录2等表示“行数据”）

值得注意的是：不管是叶节点还是同一层的子节点，都会构成一个双向链表；叶节点是为了完成高效率的“范围查询”，而子节点是为了完成“子节点分裂”的需求

此外，由于MylSUM的索引文件和数据文件是分开的，因此记录1、记录2等表示“行数据在数据文件中的位置”，属于“非聚集索引”

接下来介绍几个索引失效的场景：

**1.联合索引没有遵守最左前缀匹配**：

假设用a，b，c三个字段创建了联合索引，使用select \* from table where b=1 and c=3 的情况是不会走联合索引的

只有a相同的情况，b和c的比较才有意义，因为只要a1大于a2，那么不管b2大于b1多少，都会认为a1对应的行数据大于a2的

**2.使用like “%xx”模糊查询**：

将字符串索引看成一个由**多个单字符字段组成的联合索引**（因为比较字符串大小时，是一个个字符进行比较的）

比如将一个字符串“黄飞虎”看成abc字段组成的联合索引，那么’黄’对应字段a的值，’飞’对应字段b的值，’虎’对应字段c的值

那么 like “黄%” 就相当于 select \* from table where a=’黄’，明显是会走联合索引的；而

like “%飞虎” 就相当于 select \* from table where b=’飞’ and c=’虎’ ，因为不能保证a的值一样（因为只要a1大于a2，那么不管b2大于b1多少，都会认为a1对应的行数据大于a2的

），索引没法走联合索引，最终就导致索引失效

**3.对索引字段进行修改：**

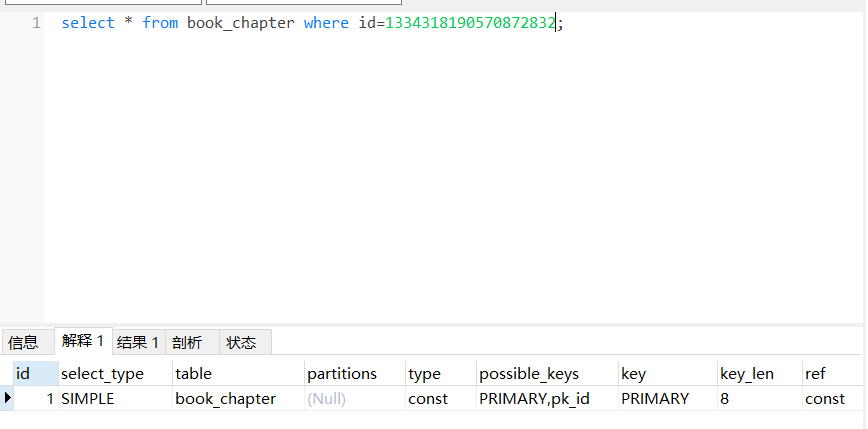
比如select \* from table where a+1=10 ，即使你对a字段建立了索引，也不会走索引字段，因为这个**等价于找“a+1字段”**，而这个字段在索引中不存在

想要达成目的，你可以将其改成select \* from table where a=10-1 ，这样就会走a字段的索引

**4.使用or条件查询：**

比如select \* from table where a=1 or b=2 ，只要a或者b有一个不是索引字段，那么就不会走索引

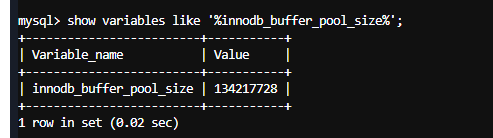
**5.使用 select \* 并不会导致索引失效**：



通过explain命令可以看见，该语句是走了主键索引的

不过，虽然select \*会走索引，但是可能会导致获取不需要的数据，一定程度上还是会影响效率

**6.数据量较少**：

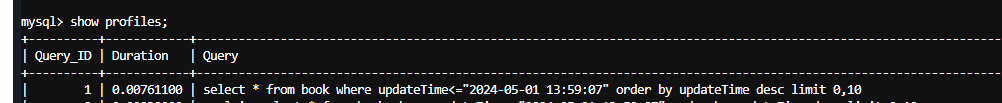
 大概为128MB

可以通过该SQL语句，看到MySQL的buffer pool的大小，128MB对于表数据量少的情况，就算将整个表装进buffer pool中都是可能的，基本就相当于直接对内存进行操作了；这种情况，会导致其对**“精准找到需要的硬盘中的数据页，将其加载进buffer pool”**的优化效果不明显

但即使如此，建立索引还是能**优化排序效果**：



执行后，能调用 show profiles 语句查看SQL语句的执行时间



这是没有对updateTime建立索引的情况，耗时为7ms（这种情况，已经是**提前预热过buffer pool的情况**）



而这个是对updateTime建立索引的情况，耗时为4ms，优化了3ms