以MySQL单表查询来举例，看看执行计划包含哪些内容（2）？

今天我们继续来说执行计划里包含的数据访问方式，上次说了const和ref，以及ref_or_null，想必大家都理解了，今天来说说其他的数据访问方式

先说说range这个东西，这个东西顾名思义，其实就是你SQL里有范围查询的时候就会走这个方式。

比如写一个SQL是select * from table where age>=x and age <=x，假设age就是一个普通索引，此时就必然利用索引来进行范围筛选，一旦利用索引做了范围筛选，那么这种方式就是range。

接着停下脚步做个总结，假设你在执行计划里看到了const、ref和range，他们是什么意思?

别担心，他们都是说基于索引在查询，总之都是走索引，所以一般问题不是太大，除非你通过索引查出来的数据量太多了，比如上面那个范围筛选，一下子查出来10万条数据，那不是想搞死MySQL么！是不是!

下面我们来讲一种比较特殊的数据访问方式，就是index，可能有的人看到这个index，天真的认为，这不就是通过索引来获取数据么，从索引根节点开始一通二分查找，不停的往下层索引跳转，就可以了，速度超快，感觉上跟ref或者range是一回事。

那你就大错特错了!

假设我们有一个表，里面完整的字段联合索引是KEY(x1,x2,x3)，好，现在我们写一个SQL语句是select x1,x2,x3 from table where x2=xxx，相信大多数同学看到这里，都会觉得，完蛋了，x2不是联合索引的最左侧的那个字段啊!

对的，这个SQL是没办法直接从联合索引的索引树的根节点开始二分查找，快速一层一层跳转的，那么他会怎么执行呢？不知道大家是否发现这个SQL里要查的几个字段，就是联合索引里的几个字段，巧了!

所以针对这种SQL，在实际查询的时候，就会直接遍历KEY(x1,x2,x3)这个联合索引的索引树的叶子节点，大家还记得聚簇索引和普通索引的叶子节点分别存放了什么吗?

聚簇索引的叶子节点放的是完整的数据页，里面包含完整的一行一行的数据，联合索引的叶子节点放的也是页，但是页里每一行就x1、x2、x3和主键的值!

所以此时针对这个SQL，会直接遍历KEY(x1,x2,x3)索引树的叶子节点的那些页，一个接一个的遍历，然后找到 x2=xxx 的那个数据，就把里面的x1，x2，x3三个字段的值直接提取出来就可以了！这个遍历二级索引的过程，要比遍历聚簇索引快多了，毕竟二级索引叶子节点就包含几个字段的值，比聚簇索引叶子节点小多了，所以速度也快！

也就是说，此时只要遍历一个KEY(x1,x2,x3)索引就可以了，不需要回源到聚簇索引去！**针对这种只要遍历二级索引就可以拿到你想要的数据，而不需要回源到聚簇索引的访问方式，就叫做index访问方式！**

是不是跟大家一开始理解的很不一样？没错，所以理解执行计划的前提，是对索引结构和使用索引的原理有一个透彻的理解，在这个基础之上，很容易就可以理解各种各样的执行计划里的访问方式了，脑子里甚至直接可以知道不同的访问方式在图里的执行路径。

现在我们停一下脚步，思考一下，之前说的const、ref和range，本质都是基于索引树的二分查找和多层跳转来查询，所以性能一般都是很高的，然后接下来到index这块，速度就比上面三种要差一些了，因为他是走遍历二级索引树的叶子节点的方式来执行了，那肯定比基于索引树的二分查找要慢多了，但是还是比全表扫描好一些的。

End