98 MySQL是如何基于各种规则去优化执行计划的？（中）

今天我们来讲一下**子查询是如何执行的****，以及他的执行计划是如何优化的**。比如说类似于下面的SQL语句：

select * from t1 where x1 = (select x1 from t2 where id=xxx)

这就是一个典型的子查询

也就是说上面的SQL语句在执行的时候，其实会被拆分为两个步骤：第一个步骤先执行子查询，也就是：select x1 from t2 where id=xxx，直接根据主键定位出一条数据的x1字段的值。接着再执行select * from t1 where x1=子查询的结果值，这个SQL语句。

这个第二个SQL执行，其实也无非就是跟之前讲的单表查询的方式是一样的，其实大家看到最后会发现，这个SQL语句最核心的就是单表查询的几种执行方式，其他的多表关联，子查询，这些都是差不多这个意思。

最多就是在排序、分组聚合的时候，可能有的时候会直接用上索引，有的时候用不上索引就会基于内存或者临时磁盘文件执行。

另外还有一种子查询，就是：

select * from t1 where x1 = (select x1 from t2 where t1.x2=t2.x2)

这种时候，你会发现子查询里的where条件依赖于t1表的字段值，所以这种查询就会效率很低下，他需要遍历t1表里每一条数据，对每一条数据取出x2字段的值，放到子查询里去执行，找出t2表的某条数据的x1字段的值，再放到外层去判断，是否符合跟t1表的x1字段匹配。

其实大家只要理解透彻了前面的内容，现在看这些SQL语句的执行原理都是比较简单的，并没有什么新意，那么接着我们就重点来讲讲这个子查询执行的时候，执行计划上会有哪些优化的规则。

今天我们重点来讲一下IN语句结合子查询的一个优化手段，假设有如下的一个SQL语句：

select * from t1 where x1 in (select x2 from t2 where x3=xxx)

这个SQL语句就是典型的一个子查询运用，子查询查一波结果，然后判断t1表哪些数据的x1值在这个结果集里。

这个可能大家会想当然的认为先执行子查询，然后对t1表再进行全表扫描，判断每条数据是否在这个子查询的结果集里，但是这种方式其实效率是非常低下的。

所以其实对于上述的子查询，执行计划会被优化为，先执行子查询，也就是select x2 from t2 where x3=xxx这条SQL语句，把查出来的数据都写入一个临时表里，也可以叫做物化表，意思就是说，把这个中间结果集进行物化。

这个物化表可能会基于memory存储引擎来通过内存存放，如果结果集太大，则可能采用普通的b+树聚簇索引的方式放在磁盘里。但是无论如何，这个物化表都会建立索引，所以大家要清楚，这波中间结果数据写入物化表是有索引的。

接着大家可能会想，此时是不是全表扫描t1表，对每条数据的x1值都去物化表里根据索引快速查找一下是否在这个物化表里？如果是的话，那么就符合条件了。但是这里还有一个优化的点，那就是他可以反过来思考。

也就是说，假设t1表的数据量是10万条，而物化表的数据量只有500条，那么此时完全可以改成全表扫描物化表，对每个数据值都到t1表里根据x1这个字段的索引进行查找，查找物化表的这个值是否在t1表的x1索引树里，如果在的话，那么就符合条件了。

所以基于IN语句的子查询执行方式，实际上会在底层被优化成如上所述。

End