104 透彻研究通过explain命令得到的SQL执行计划（5）

上回我们通过一些复杂的SQL语句给大家讲解了执行计划里的select_type一般都会有哪些取值，这次我们再来看看执行计划里的type有哪些取值，其实select_type并不是很关键，因为他主要是代表了大SQL里的不同的SELECT代表了一个什么角色，比如有的SELECT是PRIMARY查询，有的是UNION，有的是SUBQUERY。

但是这个type就非常关键了，因为他直接决定了对某个表是如何从里面查询数据的，关于这个查询方式我们之前早就讲过了，包括了const、ref、range、index、all这几种方式，分别是根据主键/唯一索引查询，根据二级索引查询，对二级索引进行全索引扫描，对聚簇索引进行全表扫描。

那今天我们就重点来通过几个SQL语句来看看在什么情况下会有什么样的type取值。

首先，假设是类似于select * fromt1 where id=110这样的SQL，直接根据主键进行等值匹配查询，那执行计划里的type就会是const，意思就是极为快速，性能几乎是线性的。

事实也确实是极为快速的，因为主键值是不会重复的，这个唯一值匹配，在一个索引树里跳转查询，基本上几次磁盘IO就可以定位到了。

接着我们来看一个SQL语句：

EXPLAIN SELECT * FROM t1 INNER JOIN t2 ON t1.id = t2.id

这里是通过两个表的id进行关联查询的，此时他的执行计划如下：

```
+----+-------------+-------+------------+--------+---------------+---------+---------+-----------------+------+----------+-------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref       | rows | filtered | Extra |
+----+-------------+-------+------------+--------+---------------+---------+---------+-----------------+------+----------+-------+
|  1 | SIMPLE      | t1    | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL      | 3467 |   100.00 | NULL  |
|  1 | SIMPLE      | t2    | NULL       | eq_ref | PRIMARY       | PRIMARY | 10      | test_db.t1.id |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+--------+---------------+---------+---------+-----------------+------+----------+-------+
```

在这个执行计划里，我们会发现针对t1表是一个全表扫描，这个是必然的，因为关联的时候会先查询一个驱动表，这里就是t1，他没什么where筛选条件，自然只能是全表扫描查出来所有的数据了。

接着针对t2表的查询type是eq_ref，而且使用了PRIMARY主键。这个意思就是说，针对t1表全表扫描获取到的每条数据，都会去t2表里基于主键进行等值匹配，此时会在t2表的聚簇索引里根据主键值进行快速查找，所以在连接查询时，针对被驱动表如果基于主键进行等值匹配，那么他的查询方式就是eq_ref了。

而如果要是正常基于某个二级索引进行等值匹配的时候，type就会是ref，而如果基于二级索引查询的时候允许值为null，那么查询方式就会是ref_or_null

另外之前讲过，有一些特殊场景下针对单表查询可能会基于多个索引提取数据后进行合并，此时查询方式会是index_merge这种。

而查询方式是range的话就是基于二级索引进行范围查询，查询方式是index的时候是直接扫描二级索引的叶子节点，也就是扫描二级索引里的每条数据，最后如果是all的话就是全表扫描，也就是对聚簇索引的叶子节点扫描每条数据。

基本上执行计划里的type就这么几种取值了，其实之前都讲过，这里主要是带着大家来复习一遍。今天我们就讲到这里，明天我们接着讲解执行计划。

End