108 透彻研究通过explain命令得到的SQL执行计划（9）

今天是我们学习SQL执行计划的最后一讲，下周就要开始进入SQL调优实战案例环节了，我们会讲解大量的SQL调优实战案例，所以大家务必要把SQL执行计划都给掌握的扎实一些。

今天我们来看看执行计划里平时常见的最后两种，一个是Using filesort，一个是Using temprory。

先来看看**Using filesort**是什么意思，首先大家要知道，有的时候我们在SQL语句里进行排序的时候，如果排序字段是有索引的，那么其实是直接可以从索引里按照排序顺序去查找数据的，比如这个SQL：

EXPLAIN SELECT * FROM t1 ORDER BY x1 LIMIT 10

这就是典型的一个排序后再分页的语句，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       | index | NULL          | index_x1 | 458     | NULL |   10 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+----------+---------+------+------+----------+-------+
```

大家可以看到，这个SQL语句，他是用了index方式访问的，意思就是说直接扫描了二级索引，而且实际使用的索引也是index_x1，本质上来说，他就是在 index_x1索引里，按照顺序找你LIMIT 10要求的10条数据罢了。

所以大家看到返回的数据是10条，也没别的过滤条件了，所以filtered是100%，也就是10条数据都返回了。

但是如果我们排序的时候是没法用到索引的，此时就会基于内存或者磁盘文件来排序，大部分时候得都基于磁盘文件来排序，比如说这个SQL：

EXPLAIN SELECT * FROM t1 ORDER BY x2 LIMIT 10

x2字段是没有索引的，此时执行计划如下

```
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra          |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------+
```

| 1 | SIMPLE | t1 | NULL | ALL | NULL | NULL | NULL | NULL | 4578 | 100.00 | Using filesort |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------+

这个SQL很明确了，他基于x2字段来排序，是没法直接根据有序的索引去找数据的，只能把所有数据写入一个临时的磁盘文件，基于排序算法在磁盘文件里按照x2字段的值完成排序，然后再按照LIMIT 10的要求取出来头10条数据。

所以大家以后要注意一下，这种把表全数据放磁盘文件排序的做法真的是相当的糟糕，性能其实会极差的。

最后给大家讲一下，如果我们用group by、union、distinct之类的语法的时候，万一你要是没法直接利用索引来进行分组聚合，那么他会直接基于临时表来完成，也会有大量的磁盘操作，性能其实也是极低的。

比如这个SQL：

EXPLAIN SELECT x2, COUNT(*) AS amount FROM t1 GROUP BY x2

这里的x2是没有索引的，所以此时的执行计划如下

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-----------------+

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-----------------+

| 1 | SIMPLE | t1 | NULL | ALL | NULL | NULL | NULL | NULL | 5788 | 100.00 | Using temporary |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-----------------+

这个SQL里只能对全表数据放到临时表里做大量的磁盘文件操作，然后才能完成对x2字段的不同的值去分组，分组完了以后对不同x2值的分组去做聚合操作，这个过程也是相当的耗时的，性能是极低的。

所以大家最后记住，其实未来在SQL调优的时候，核心就是分析执行计划里哪些地方出现了全表扫描，或者扫描数据过大，尽可能通过合理优化索引保证执行计划每个步骤都可以基于索引执行，避免扫描过多的数据。

End