



下载APP



24 | 查询有点慢，语句该如何写？

2021-05-06 朱晓峰

MySQL 必知必会

[进入课程 >](#)

讲述：朱晓峰

时长 09:26 大小 8.65M



你好，我是朱晓峰。这节课，我想和你聊一聊怎么对查询语句进行调优。

你肯定遇到过这样的情况：你写的 SQL 语句执行起来特别慢，要等好久才出结果，或者是干脆就“死”在那里，一点反应也没有。一旦遇到这种问题，你就要考虑进行优化了。


如果你开发过数据库应用，肯定会有这样的体会：**让应用运行起来不难，但是要运行得又快又好，就没那么容易了。**这很考验我们的内功。

而要想提高应用的运行效率，你就必须掌握优化查询的方法。今天，我就给你讲一下 MySQL 的查询分析语句和 2 种优化查询的方法。



查询分析语句

虽然 MySQL 的查询分析语句并不能直接优化查询，但是却可以帮助你了解 SQL 语句的执行计划，有助于你分析查询效率低下的原因，进而有针对性地进行优化。查询分析语句的语法结构是：

 复制代码


```
1 { EXPLAIN | DESCRIBE | DESC }查询语句;
```

下面我借助一个小例子，给你详细地讲解一下，怎么使用查询分析语句，来分析一个查询的执行计划。

假设有一个销售流水表（demo.trans），里面有 400 万条数据，如下所示：

itemnumber (商品编号)	quantity (销售数量)	price (价格)	actualvalue (销售金额)	cashiernumber (收款机号)	branchnumber (门店编号)	transdate (交易时间)

现在，我要查询一下商品编号是 1 的商品，在 2020 年 6 月 18 日上午 9 点到 12 点之间的销售明细。代码如下所示：

 复制代码

```
1 mysql> SELECT itemnumber,quantity,price,transdate
2 -> FROM demo.trans
3 -> WHERE itemnumber=1
4 -> AND transdate>'2020-06-18 09:00:00'
5 -> AND transdate<'2020-06-18 12:00:00';
6 +-----+-----+-----+-----+
7 | itemnumber | quantity | price | transdate |
8 +-----+-----+-----+-----+
9 | 1 | 0.276 | 70.00 | 2020-06-18 11:04:00 |
10 | 1 | 1.404 | 70.00 | 2020-06-18 11:10:57 |
11 | 1 | 0.554 | 70.00 | 2020-06-18 11:18:12 |
12 | 1 | 0.431 | 70.00 | 2020-06-18 11:27:39 |
13 | 1 | 0.446 | 70.00 | 2020-06-18 11:42:08 |
14 | 1 | 0.510 | 70.00 | 2020-06-18 11:56:43 |
15 +-----+-----+-----+-----+
16 6 rows in set (6.54 sec)
```

结果显示，有 6 条记录符合条件。这个简单的查询一共花去了 6.54 秒，这个速度显然太慢了。

为了找到查询运行慢的原因，咱们来分析一下它，看看为什么会用这么多时间，有没有办法优化。现在，我们用下面的语句分析一下这个查询的具体细节：

[复制代码](#)

```
1 mysql> EXPLAIN SELECT itemnumber,quantity,price,transdate -- 分析查询执行情况
2 -> FROM demo.trans
3 -> WHERE itemnumber=1 -- 通过商品编号筛选
4 -> AND transdate>'2020-06-18 09:00:00' -- 通过交易时间筛选
5 -> AND transdate<'2020-06-18 12:00:00';
6 +-----+-----+-----+-----+-----+-----+-----+-----+-----+
7 | id | select_type | table | partitions | type | possible_keys | key | key_len
8 +-----+-----+-----+-----+-----+-----+-----+-----+-----+
9 | 1 | SIMPLE | trans | NULL | ALL | NULL | NULL | NULL | NULL | NULL | 4157166 | 1.11
10 +-----+-----+-----+-----+-----+-----+-----+-----+-----+
11 1 row in set, 1 warning (0.00 sec)
```

这个结果集中包含了很多个字段，理解这些字段的意思，是我们优化查询的关键。

字段比较多，我先说说简单的字段，你只要知道概念就可以了。另外一类比较复杂的字段，一会儿我再具体讲。

1. id：是一个查询序列号。
2. table：表示与查询结果相关的表的名称。
3. partition：表示查询访问的分区。
4. key：表示优化器最终决定使用的索引是什么。
5. key_len：表示优化器选择的索引字段按字节计算的长度。如果没有使用索引，这个值就是空。
6. ref：表示哪个字段或者常量被用来与索引字段比对，以读取表中的记录。如果这个值是“func”，就表示用函数的值与索引字段进行比对。
7. rows：表示为了得到查询结果，必须扫描多少行记录。
8. filtered：表示查询筛选出的记录占全部表记录数的百分比。

9. possible_key: 表示 MySQL 可以通过哪些索引找到查询的结果记录。如果这里的值是空，就说明没有合适的索引可用。你可以通过查看 WHERE 条件语句中使用的字段，来决定是否可以通过创建索引提高查询的效率
10. Extra: 表示 MySQL 执行查询中的附加信息。你可以点击这个 [🔗 链接](#) 查询详细信息。
11. type: 表示表是如何连接的。至于具体的内容，你可以参考下 [🔗 查询分析语句输出内容说明](#)。

除了刚刚这些字段，还有 1 个比较重要，那就是 select_type。它表示查询的类型，主要有 4 种取值。

SIMPLE: 表示简单查询，不包含子查询和联合查询。


PRIMARY: 表示是最外层的查询。

UNION: 表示联合查询中的第二个或者之后的查询。

DEPENDENTUNION: 表示联合查询中的第二个或者之后的查询，而且这个查询受外查询的影响。

关于这个 DEPENDENTUNION 取值，不是很好理解，我举个小例子。

假设我们有下面的查询语句：

 复制代码

```
1 mysql> SELECT *
2 -> FROM demo.goodsmaster a
3 -> WHERE itemnumber in
4 -> (
5 -> SELECT b.itemnumber
6 -> FROM demo.goodsmaster b
7 -> WHERE b.goodsname = '书'
8 -> UNION
9 -> SELECT c.itemnumber
10 -> FROM demo.goodsmaster c
11 -> WHERE c.goodsname = '笔'
12 -> );
13 +-----+-----+-----+-----+-----+-----+-----+
14 | itemnumber | barcode | goodsname | specification | unit | salesprice | avgim
15 +-----+-----+-----+-----+-----+-----+-----+
16 | 1 | 0001 | 书 | 16开 | 本 | 89.00 | 31.00 |
17 | 2 | 0002 | 笔 | NULL | 包 | 5.00 | 2.87 |
18 +-----+-----+-----+-----+-----+-----+-----+
```

```
19 2 rows in set (0.00 sec)
```

对这个语句的执行进行分析，得到如下的结果：

[复制代码](#)


```
1 mysql> EXPLAIN SELECT *
2 -> FROM demo.goodsmaster a
3 -> WHERE a.itemnumber in
4 -> (
5 -> SELECT b.itemnumber
6 -> FROM demo.goodsmaster b
7 -> WHERE b.goodsname = '书'
8 -> UNION
9 -> SELECT c.itemnumber
10 -> FROM demo.goodsmaster c
11 -> WHERE c.goodsname = '笔'
12 -> );
13 +-----+-----+-----+-----+-----+-----+-----+
14 | id | select_type | table | partitions | type | possible_keys | key | key_len
15 +-----+-----+-----+-----+-----+-----+-----+
16 | 1 | PRIMARY | a | NULL | ALL | NULL | NULL | NULL | 2 | 100.00 | Usin
17 | 2 | DEPENDENT SUBQUERY | b | NULL | eq_ref | PRIMARY | PRIMARY | 4 | func |
18 | 3 | DEPENDENT UNION | c | NULL | eq_ref | PRIMARY | PRIMARY | 4 | func | 1 |
19 | NULL | UNION RESULT | <union2,3> | NULL | ALL | NULL | NULL | NULL | NULL |
20 +-----+-----+-----+-----+-----+-----+-----+
21 4 rows in set, 1 warning (0.00 sec)
```

MySQL 在执行的时候，会把这个语句进行优化，重新写成下面的语句：

[复制代码](#)

```
1 SELECT *
2 FROM demo.goodsmaster a
3 WHERE EXISTS
4 (
5 SELECT b.id
6 FROM demo.goodsmaster b
7 WHERE b.goodsname = '书' AND a.itemnumber=b.itemnumber
8 UNION
9 SELECT c.id
10 FROM demo.goodsmaster c
11 WHERE c.goodsname = '笔' AND a.itemnumber=c.itemnumber
12 );
```

在这里，子查询中的联合查询是：

 复制代码

```
1 SELECT c.id
2 FROM demo.goodsmaster c
3 WHERE c.goodsname = '笔' AND a.itemnumber=c.itemnumber
```

这个查询就用到了与外部查询相关的条件 `a.itemnumber=c.itemnumber`，因此，查询类别就变成了“UNION DEPENDENT”。

好了，现在，我们已经知道了查询分析语句的结果集中各个字段的含义。现在来分析一下刚刚的查询语句。

我们发现，这个查询是一个简单查询，涉及的表是 `demo.trans`，没有分区，连接类型是扫描全表，没有索引，一共要扫描的记录数是 4157166。因此，查询速度慢的主要原因是没有索引，导致必须要对全表进行扫描才能完成查询。所以，针对这个问题，可以通过创建索引的办法，来提高查询的速度。

下面，我们用条件语句中的筛选字段 `itemnumber` 和 `transdate` 分别创建索引：

 复制代码

```
1 mysql> CREATE INDEX itemnumber_trans ON demo.trans(itemnumber);
2 Query OK, 0 rows affected (59.86 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
4
5 mysql> CREATE INDEX transdate_trans ON demo.trans(transdate);
6 Query OK, 0 rows affected (56.75 sec)
7 Records: 0 Duplicates: 0 Warnings: 0
```

然后我们再次运行刚才的查询，看看优化有没有起作用：

 复制代码

```
1 mysql> SELECT itemnumber,quantity,price,transdate
2     -> FROM demo.trans
3     -> WHERE itemnumber=1
4     -> AND transdate>'2020-06-18 09:00:00'
5     -> AND transdate<'2020-06-18 12:00:00';
6 +-----+-----+-----+-----+
```

```

7 | itemnumber | quantity | price | transdate |
8 +-----+-----+-----+-----+
9 |          1 | 0.276    | 70.00 | 2020-06-18 11:04:00 |
10 |          1 | 1.404    | 70.00 | 2020-06-18 11:10:57 |
11 |          1 | 0.554    | 70.00 | 2020-06-18 11:18:12 |
12 |          1 | 0.431    | 70.00 | 2020-06-18 11:27:39 |
13 |          1 | 0.446    | 70.00 | 2020-06-18 11:42:08 |
14 |          1 | 0.510    | 70.00 | 2020-06-18 11:56:43 |
15 +-----+-----+-----+-----+
16 6 rows in set (0.09 sec)

```

结果显示，查询只运行了 0.09 秒，跟之前的 6.54 秒相比，快了很多。这说明我们的优化措施起了作用。下面我们再次运行查询分析语句，查看一下现在的查询执行计划。

[复制代码](#)

```

1 mysql> EXPLAIN SELECT itemnumber,quantity,price,transdate
2     -> FROM demo.trans
3     -> WHERE itemnumber=1                -- 按商品编号筛选
4     -> AND transdate>'2020-06-18 09:00:00'-- 按照交易时间筛选
5     -> AND transdate<'2020-06-18 12:00:00';
6 +----+-----+-----+-----+-----+-----+
7 | id | select_type | table | partitions | type | possible_keys |
8 +----+-----+-----+-----+-----+-----+
9 | 1 | SIMPLE      | trans | NULL        | range | itemnumber_trans,transdate_t
10 +----+-----+-----+-----+-----+-----+
11 1 row in set, 1 warning (0.01 sec)

```

结果显示，这一次的查询执行计划有了改变。系统发现，有 2 个索引 `itemnumber_trans` 和 `transdate_trans` 可以使用，并且最终选择了使用交易时间字段创建的索引 `transdate_trans` 来执行查询。扫描的方式也不再是全表扫描，而是改为了区域扫描，实际的扫描记录数也减少到了 552 个。这样一来，查询更加精准，查询的速度自然也就大幅提高了。

至此，我给你介绍了查询分析语句，并且演示了如何通过使用查询分析语句对慢查询的执行计划进行分析，并且利用分析的结果对慢查询进行优化。接下来，我再给你介绍 2 种优化查询的方法。

2 种查询优化的方法

优化查询最有效的方法就是创建索引。关于如何创建索引，我已经在 [🔗 第 11 讲](#) 中介绍过了，这里就不多说了。下面我来讲讲怎么在包含关键字 “LIKE” 和 “OR” 的条件语句中，利用索引提高查询效率。

使用关键字“LIKE”


“LIKE” 经常被用在查询的限定条件中，通过通配符 “%” 来筛选符合条件的记录。比如，

1. WHERE 字段 LIKE ‘%aa’，表示筛选出所有以字段以 “aa” 结尾的记录；
2. WHERE 字段 LIKE ‘aa%’，表示筛选出所有以 “aa” 开始的记录；
3. WHERE 字段 LIKE ‘%aa%’，表示所有字段中包含 “aa” 的记录。

这里你要注意的，通配符在前面的筛选条件是不能用索引的。也就是说，WHERE 字段 LIKE ‘%aa’ 和 WHERE 字段 LIKE ‘%aa%’ 都不能使用索引，但是通配符在后面的筛选条件，就可以使用索引。


下面，我举个小例子，通过查询分析语句来验证一下索引的使用情况。

假设我用商品流水表的字段商品条码 “barcode” 创建了一个索引：

 复制代码

```
1 mysql> CREATE INDEX trans_barcode ON demo.trans(barcode);
2 Query OK, 0 rows affected (1 min 20.78 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

我们来看看如果把通配符放在前面，能不能用到索引：

 复制代码

```
1 mysql> EXPLAIN SELECT * FROM demo.trans
2 -> WHERE barcode LIKE '%182505';
3 +----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 | id | select_type | table | partitions | type | possible_keys | key | key_len
5 +----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 | 1 | SIMPLE | trans | NULL | ALL | NULL | NULL | NULL | NULL | NULL | 4144028 | 11.1
7 +----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
8 1 row in set, 1 warning (0.00 sec)
```


查询分析的结果显示，type 的值是 “ALL” ， key 是空，表示需要进行全表扫描，没有索引可用，rows 的值是 4144028，基本上要检索全部记录，效率非常低。

再看看通配符在后面的情况：

[复制代码](#)

```
1 mysql> EXPLAIN SELECT * FROM demo.trans
2 -> WHERE barcode LIKE '6953150%';
3 +-----+-----+-----+-----+-----+-----+-----+-----+
4 | id | select_type | table | partitions | type | possible_keys | key | key_len
5 +-----+-----+-----+-----+-----+-----+-----+-----+
6 | 1 | SIMPLE | trans | NULL | range | trans_barcode | trans_barcode | 803 | NU
7 +-----+-----+-----+-----+-----+-----+-----+-----+
8 1 row in set, 1 warning (0.00 sec)
```

type 的值是 “range” ，意思是使用索引检索一个给定范围的记录。rows 的值是 563，也就是只需要扫描 563 条记录就行了，这样效率就高多了。

使用关键字“OR”

关键字 “OR” 表示 “或” 的关系，“WHERE 表达式 1 OR 表达式 2”，就表示表达式 1 或者表达式 2 中只要有一个成立，整个 WHERE 条件就是成立的。

需要注意的是，只有当条件语句中只有关键字 “OR” ，并且 “OR” 前后的表达式中的字段都建有索引的时候，查询才能用到索引。

同样，为了方便你理解，我还是举个小例子，通过查询分析语句来实际验证一下。

我刚才已经用字段条码给商品流水表创建了一个索引，现在我再用商品编号 “itemnumber” 创建一个索引：

[复制代码](#)

```
1 mysql> CREATE INDEX trans_itemnumber ON demo.trans(itemnumber);
2 Query OK, 0 rows affected (20.24 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

我们先看一下关键字 “OR” 前后的表达式中的字段都创建了索引的情况：

[复制代码](#)

```
1 mysql> EXPLAIN SELECT * FROM demo.trans
2 -> WHERE barcode LIKE '6953150%'
3 -> OR itemnumber = 1;
4 +----+-----+-----+-----+-----+-----+-----+-----+
5 | id | select_type | table | partitions | type | possible_keys | key | key_len
6 +----+-----+-----+-----+-----+-----+-----+-----+
7 | 1 | SIMPLE | trans | NULL | index_merge | trans_barcode,trans_itemnumber | t
8 +----+-----+-----+-----+-----+-----+-----+-----+
9 1 row in set, 1 warning (0.01 sec)
```

查询分析结果显示，有 2 个索引可以使用，分别是 “trans_barcode” 和 “trans_itemnumber”。key=index_merge，就说明优化器选择了合并索引的方式。因此，这个关键字 “OR” 前后的表达式中的字段都创建了索引的查询，是可以用到索引的。

在下面的例子中，表达式 goodsname LIKE '%海鲜菇%' 中的字段 goodsname 没有创建索引，我们来验证一下查询是否能够利用索引：

[复制代码](#)

```
1 mysql> EXPLAIN SELECT * FROM demo.trans
2 -> WHERE barcode LIKE '6953150%'
3 -> OR goodsname LIKE '%海鲜菇%';
4 +----+-----+-----+-----+-----+-----+-----+-----+
5 | id | select_type | table | partitions | type | possible_keys | key | key_len
6 +----+-----+-----+-----+-----+-----+-----+-----+
7 | 1 | SIMPLE | trans | NULL | ALL | trans_barcode | NULL | NULL | NULL | 41440
8 +----+-----+-----+-----+-----+-----+-----+-----+
9 1 row in set, 1 warning (0.00 sec)
```

查询分析的结果显示，type=ALL，是全表扫描，不能用索引。

总结

今天，我们学习了优化查询的方法，包括用来分析查询执行情况的查询分析语句，语法结构是：

```
1 { EXPLAIN | DESCRIBE | DESC }查询语句；
```

同时，我们还学习了在使用关键字“LIKE”和“OR”的情况下，用索引来提高查询效率的方法。

特别需要提醒你注意的是，在使用“LIKE”关键字的条件语句中，通配符“%”在前面的筛选条件不能使用索引，通配符“%”在后面的筛选条件可以使用索引。在使用“OR”关键字的条件语句中，只有关键字“OR”前后的表达式中的字段都创建了索引，条件语句才能使用索引。

关于优化查询，还有一个值得关注的点，就是子查询。这是 MySQL 的一项重要功能，可以帮助我们通过一个 SQL 语句实现比较复杂的查询。但是，**子查询的执行效率不高**。因为 MySQL 会用临时表把子查询的结果保存起来，然后再使用临时表的内容完成查询。这样一来，查询就多了一个创建临时表的过程，执行效率没有连接查询高。针对这种情况，建议你子查询转换成连接查询，这样可以进一步提高查询的效率。

思考题

假设现在有一个这样的查询，请你把这个包含子查询的查询改成 2 个表的连接查询。

```
1 mysql> EXPLAIN SELECT * FROM demo.trans
2 -> WHERE itemnumber IN
3 -> (
4 -> SELECT itemnumber FROM demo.goodsmaster
5 -> WHERE goodsname LIKE '%书%'
6 -> );
7 +----+-----+-----+-----+-----+-----+-----+-----+
8 | id | select_type | table | partitions | type | possible_keys | key | key_len
9 +----+-----+-----+-----+-----+-----+-----+-----+
10 | 1 | SIMPLE | goodsmaster | NULL | ALL | PRIMARY | NULL | NULL | NULL | 2 | 5
11 | 1 | SIMPLE | trans | NULL | ref | trans_itemnumber | trans_itemnumber | 5 |
12 +----+-----+-----+-----+-----+-----+-----+-----+
13 2 rows in set, 1 warning (0.00 sec)
```

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，也欢迎你把它分享给你的朋友或同事，我们下节课见。

[提建议](#)

更多学习推荐

 极客大学

175 道 Go 工程师 大厂常考面试题

限量免费领取 

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 23 | ER模型：如何理清数据库设计思路？

下一篇 25 | 表太大了，如何设计才能提高性能？

精选留言 (4)

[写留言](#)**朱晓峰** 置顶

2021-05-20

你好，我是朱晓峰，下面我就来公布一下上节课思考题的答案：

上节课，我们学习了ER模型。下面是思考题的答案：

实体：...

展开 

**Harry**

2021-05-08

以前查看执行计划只知道使用 explain 今次知道了 describe 语句也能达到同样效果！

此外，还学习了在使用关键字 LIKE 和 OR 的情况下，用索引来提高查询效率的具体方法。...

展开 ∨

作者回复: 子查询可以解决一些复杂的查询问题。遇到特别复杂的查询，需要用临时表通过多个步骤最终得到结果的场景，可以通过分析每个步骤的查询语句，还有临时表的数据量，来决定优化策略



1

**沈康**

2021-05-22

```
select a.*  
from demo.trans a,demo.goodsmaster b  
where  
a.itemnumber=b.itemnumber  
and ...
```

展开 ∨

**floating**

2021-05-13

课后思考题：

```
SELECT  
*
```

```
FROM...
```

展开 ∨

作者回复: 请参考思考题答案



