6 1

首页推荐

一个不可思议的SQL优化过程及扩展几个需掌握的几个知识点

原创 chengang 2024-12-18 722

1、问题复现

昨天在微信群里有人发了一个问题，觉得不可思议，认为是MySQL的Bug，但随后题主贴出执行计划，一看执行计划就知晓问题点了，但我觉得此问题可以扩展几个重要的知识点，我用我的本地数据做了复现。问题是下面两个SQL

```
-- SQL1
select profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid

-- SQL2
select profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid
```

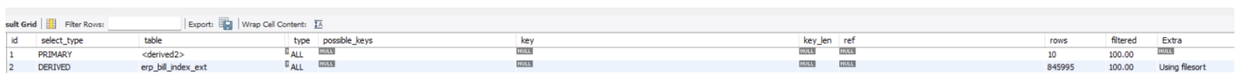
在我本地，第一个SQL执行需要0.328S，第二个SQL需要5.125S

问题是为什么加了limit 10的语句反而还要慢15倍左右，这个问题第一眼看，是觉得有点违反常理。

2、问题分析

第一个反映就是需要看执行计划

select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
SIMPLE	erp_bill_index_ext	index	NULL	idx_profileid_sorcebillid	14	NULL	845995	100.00	Using index; Using filesort



id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	10	100.00	NULL
2	DERIVED	erp_bill_index_ext	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort

通用对比执行计划，可以获得两个重要的信息
1、第一个SQL 优化器消除了派生表
2、第一个SQL 走的是索引扫描，且是索引覆盖

看到计划，估计大家都已知晓问题，并有了解决方案

3、扩展知识点

我们来扩展一些其它知识

a.如何看优化后的语句

执行explain后，执行show warnings;
我们来看SQL1优化后的样子

SQL1优化后的语句变成了

```
select profileid,sourcebillid from erp_bill_index_ext order by  sourcebillid desc
```

在这个例子中，派生表被优化掉了，性能有较大提升，但也有的SQL派生表被优化掉了，性能下降的很厉害，那我们如何控制派生表能不能被干掉呢？

b.如何控制派生表要不要和外层合并

派生表优化官方文档
有一段文章

> It is possible to disable merging by using in the subquery any constructs that prevent merging, although these are not as explicit in their effect on materialization. Constructs that prevent merging are the same for derived tables, common table expressions, and view references:
> Aggregate functions or window functions (SUM(), MIN(), MAX(), COUNT(), and so forth)
> DISTINCT
> GROUP BY
> HAVING
> LIMIT
> UNION or UNION ALL
> Subqueries in the select list
> Assignments to user variables
> References only to literal values (in this case, there is no underlying table)

chengang
LV.5
关注
97 文章 162 粉丝 109K+ 浏览量
获得了 888 次点赞
内容获得 228 次评论
获得了 288 次收藏

TA的专栏
MySQL生产实战优化 收录 15 篇内容
有意思的SQL 收录 13 篇内容
MySQL事务 收录 2 篇内容

热门文章
开源SQL审核工具Yearning搭建及使用 2022-03-21 6092浏览
深入理解MySQL锁 （S、X、IS、IX）RC模式下解析 2022-01-06 5789浏览
一文详细讲清楚 MySQL为什么是小表驱动大表？ 2023-03-10 5389浏览
很难的sql面试题解 记录一个超复杂的sql实现 2021-09-16 4813浏览
分布式 MySQL XA 详解。 2022-05-10 4415浏览

在线实训环境入口
MySQL在线实训环境 查看详情 »

最新文章
python+cline+deepseek+(deepseek-reasoner/deepseek-chat大模型)自动化编程 2025-02-07 126浏览
MySQL 利用JSON可以创建表值函数 2025-01-09 46浏览
MySQL derived_merge优化的bug 导致查询结果不正确 【官方已确认Bug】 2025-01-08 150浏览
MySQL RC模式下奇怪的行锁探讨 2024-10-16 259浏览
小心这种写法真会删库跑路，你的工资怕不够赔！！！(二) 2024-10-14 372浏览

目录
- 1、问题复现
- 2、问题分析
- 3、扩展知识点
  - a.如何看优化后的语句
  - b.如何控制派生表要不要和外层合并
  - c.全面禁止开发使用*
  - d.利用索引扫描与索引覆盖最后一张底牌
- 4、解决方案

1、当子查询包含上述语句的时候，就不能被合并，如本例中的SQL2因为子查询有了limit所以不能合并优化
所以当你想被合并或不想合并都可以通过增加或减少上述子句来控制
比如本例中被合并的SQL1 只需要稍动一点手脚即可阻止合并,增加一个用户变量在语句中即可

```
select profileid,sourcebillid from(select *,@i:=0 from erp_bill_index_ext order by  sourceb
```

```
explain
select profileid,sourcebillid from(select *,@i:=0 from erp_bill_index_ext order by  sourcebillid desc)  as t order by profileid;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort
2	DERIVED	erp_bill_index_ext	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort

2、通过hint来控制
no_merge

```
select /*+no_merge(t)*/
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)
```

```
explain
select /*+no_merge(t)*/
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)  as t ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	845995	100.00	NULL
2	DERIVED	erp_bill_index_ext	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort

derived_merge

```
SET @@optimizer_switch='derived_merge =off';
explain
select
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)
```

```
SET @@optimizer_switch='derived_merge =off';
explain
select
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)  as t ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	845995	100.00	NULL
2	DERIVED	erp_bill_index_ext	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort

但我推荐使用set_var模式，让影响变更小，只影响当前SQL

```
explain
select /*+set_var(optimizer_switch='derived_merge =off')*/
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)
```

```
explain
select /*+set_var(optimizer_switch='derived_merge =off')*/
 profileid,sourcebillid from(select * from erp_bill_index_ext order by  sourcebillid desc)  as t ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	845995	100.00	NULL
2	DERIVED	erp_bill_index_ext	ALL	NULL	NULL	NULL	NULL	845995	100.00	Using filesort

c.全面禁止开发使用*

这个例子中有一个特别有意思的点：即使这位研发人员不会任何SQL优化，只要遵守不使用* 也能得到最佳结果
我们来看看刚才慢15倍的SQL2,是因为将limit 放到派生表中的原因吗？ 不是，是因为子查询中用到了* 我们直接去掉*看看结果

```
select
profileid,sourcebillid from(select profileid,sourcebillid from erp_bill_index_ext order by
```

```
select
profileid,sourcebillid from(select profileid,sourcebillid from erp_bill_index_ext order by  sourcebillid desc limit 10)  as t
```

profileid	sourcebillid
200012229	1247419397
200012229	1247418456
200012229	1247400393
200012229	1247385786
200012229	1247382147
200012229	1247379321
200012229	1247374309
200012229	1247372977
200012229	1247369808

#	Time	Action	Message	Duration / Fetch
1	11:03:30	select profileid,sourcebillid from(select profileid,sourcebillid from erp_bill_index_ext order by sourcebillid desc limit 10) as t LIMIT 0, 400	10 row(s) returned	0.282 sec / 0.000 s

执行结果0.282S 比刚才的SQL1还要快

d.利用索引扫描与索引覆盖最后一张底牌

我们在优化SQL过程中，常常因为某些SB需求不能减少扫描行数，不能消除排序，比如产品就是我就是要查所有日期的交易记录，不限定查询时间，且还要支持按金额大小排序，还要有支付金额合计
那我们写的SQL 大概是这样的

```
select userid,paytotal,paydate from paylist where paydate >=xx and paydate <=xx
order by paytotal desc
```

上述SQL需求不变那么将很难减少行数，取消排序
如果paylist由于设计原因，又是一个较宽的表 那么就只有建一个paydate，userid,paytotal的组合索引，即使用户选的时间跨度很大最差也能让整个SQL走索引扫描并索引覆盖

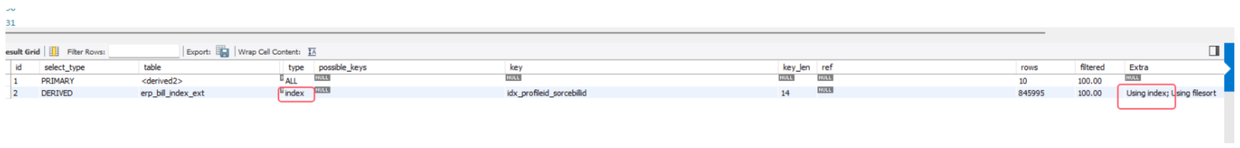
如何建立一个好的组合索引请看我的另一篇文章MySQL联合索引最佳实践

4、解决方案

通过看执行计划，此SQL1能得到很好的性能提升，并不是合并了派生表，而是因为合并了派生表，刚好按语义消除了全表扫描。走了索引扫描且索引覆盖

那我们子查询中也做到索引扫描且索引覆盖性能就能达到最优

```
select profileid,sourcebillid from(select profileid,sourcebillid from erp_bill_index_ext or
```



id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	10	100.00	NULL
2	DERIVED	erp_bill_index_ext	index	NULL	idx_profileid_sorcebillid	14	NULL	845995	100.00	Using index; Using filesort

墨力计划 mysql mysql优化

最后修改时间：2025-01-16 19:26:18

「喜欢这篇文章，您的关注和赞赏是给作者最好的鼓励」

关注作者 赞赏

【版权声明】本文为墨天轮用户原创内容，转载时必须标注文章的来源（墨天轮），文章链接，文章作者等基本信息，否则作者和墨天轮有权追究责任。如果您发现墨天轮中有涉嫌抄袭或者侵权的内容，欢迎发送邮件至：contact@modb.pro进行举报，并提供相关证据，一经查实，墨天轮将立刻删除相关内容。

文章被以下合辑收录

MySQL生产实战优化（共15篇）
生产环境遇到的慢SQL进行优化。
收藏合辑

评论

分享你的看法，一起交流吧～

星星之火可以燎原 LV.6
通过看执行计划，此SQL1能得到很好的性能提升，并不是合并了派生表，而是因为合并了派生表，刚好按语义消除了全表扫描。走了索引扫描且索引覆盖
1月前 点赞 评论

相关阅读

【干货】2024年下半年墨天轮最受欢迎的50篇技术文章+文档
墨天轮编辑部 1559次阅读 2025-02-13 10:42:44

MySQL性能分析的“秘密武器”，深度剖析SQL问题
szrsu 680次阅读 2025-01-23 09:59:26

2025年1月“墨力原创作者计划”获奖名单公布
墨天轮编辑部 348次阅读 2025-02-13 15:07:02

MySQL 主从节点切换指导
CuiHulong 302次阅读 2025-01-23 11:50:29

[MYSQL] 忘记root密码时, 不需要重启也能强制修改了!
大大刺猬 286次阅读 2025-02-06 11:12:15

mysql 内存使用率高问题排查
蔡璐 266次阅读 2025-02-06 10:02:23

MySQL 9.2.0 中的更新（2025-01-21，创新版本）
通讯员 151次阅读 2025-01-22 09:54:21

MySQL基础高频面试题－划重点、敲难点
锁钥 146次阅读 2025-02-03 07:52:28

MySQL 底层数据&日志刷新策略解读

CuiHulong 133次阅读 2025-02-11 10:56:13

[MYSQL] mysql主从延迟案例(有索引但无主键)

大大刺猬 107次阅读 2025-01-21 13:53:21