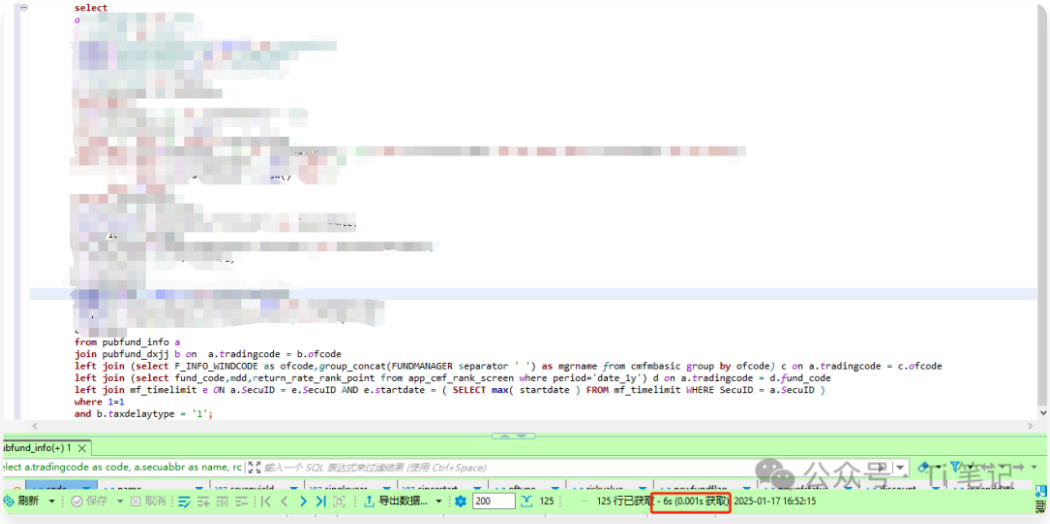
SQL优化——我是如何将SQL执行性能提升10倍的

原创 Triagen Ti 笔记 2025年01月17日 19:54 广东

本文通过记录一条SQL语句的性能优化过程，介绍MySQL中SQL语句优化的一般思路。

一、优化前

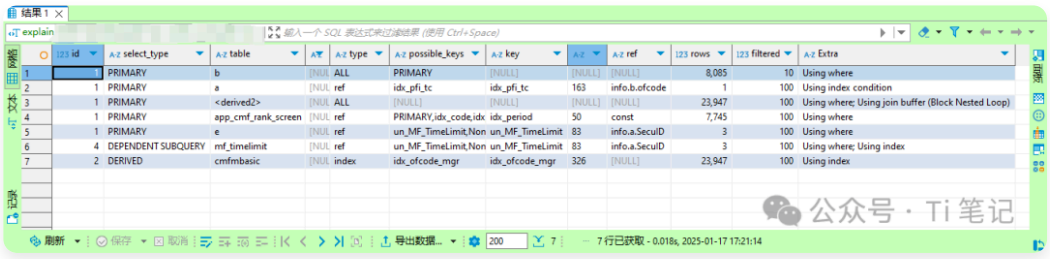
1. SQL语句及其执行时长



问题SQL语句查询需要6s

二、优化思路

1. explain 查看执行计划



执行计划

执行计划解读(逐行)

- 存储引擎全量读取了表 b (**type=ALL**)，预估会读取8085条数据(**rows=8085**，依赖统计信息，非精确值)，MySQL服务器对读取结果进行了过滤(**Extra=Using where**， where 条件 b.taxdelaytype = '1')，预计过滤后还剩10%(**filtered=10**)。

- 存储引擎通过索引读取了表 a ，对于每个 where 条件值，可能找到多条符合条件的记录(**type=ref**)，使用的索引为 idx_pfi_tc (**key=idx_pfi_tc**，**possible_keys**为候选的索引列表，**key**为通过统计信息计算后选出的索引)，索引过滤相关的条件列为 info.b.ofcode (**ref=info.b.ofcode**)，并且使用了索引下推[1] (**Extra=Using index condition**)。
- 存储引擎全量读取了表 <derived2> ，这里的 <derived2> 并非真实表，其代表的是执行计划最后一行查询出来的临时表(**select_type=DERIVED,id=2**)，MySQL服务器在执行 left join 的过程中，使用了Block Nested-Loop Join[2](**Extra=Using join buffer (Block Nested Loop)**)，并对连接结果进行了过滤(**Extra=Using where**， on 条件 a.tradingcode = c.ofcode)。
- 存储引擎通过索引 idx_period 读取表 app_cmf_rank_screen (**type=ref**)，MySQL服务器在执行 left join 的过程中，使用了Index Nested-Loop Join[3](**Extra**没有对连接进行说明，就是默认的**Index Nested-Loop Join**)，并对连接结果进行了过滤(**Extra=Using where**， on 条件 a.tradingcode = d.fund_code)。
- 存储引擎通过索引 un_MF_TimeLimit 读取表 e (**type=ref**)，MySQL服务器在连接后对结果进行了过滤(**Extra=Using where**， on 条件 a.SecuID = e.SecuID AND e.startdate = ...)。
- 存储引擎通过索引 un_MF_TimeLimit 读取表 mf_timelimit (**type=ref**)，并且使用了覆盖索引[4] (**Extra=Using index**，因为索引 un_MF_TimeLimit 中包含 startdate 和 SecuID)。
- 存储引擎通过索引 idx_ofcode_mgr 读取表 cmfmbasic (**type=ref**)，并且同样使用了**覆盖索引**。

执行计划总结

从执行计划来看，初步怀疑是因为第三行的 Using join buffer (Block Nested Loop) 导致查询效率低下，于是尝试通过调整 join_buffer_size 的大小进行优化，然而并没有效果，优化有点陷入僵局了。

2. show warnings 查看告警信息

show warnings 查看告警信息

告警信息解读

在执行了 explain 查看执行计划命令之后，可以通过 show warnings 查看相关的告警信息。 show warnings 结果的第二和第三行显示，有索引没有被用上，通过告警里面的字段可知是表 app_cmf_rank_screen ，而与其相关的表是 a(pubfund_info) ，两者通过条件 a.tradingcode = d.fund_code 相关联.

3. 告警信息确认

```
select table_name,column_name,column_type,character_set_name,collation_name
from information_schema.columns
where table_schema ='info'
and table_name in ('app_cmf_rank_screen','pubfund_info')
and column_name in ('fund_code','tradingcode');
```


查看表的列属性

结果显示，我们两张表关联字段的排序规则确实是不一样的，表 app_cmf_rank_screen 是按 utf8mb4_general_ci 排序，表 pubfund_info 是按 utf8mb4_bin 排序，所以导致有索引无法被使用。

三、优化后

两张表的排序规则统一为 utf8mb4_bin

语句性能优化到0.7s

优化后的执行计划

相比于优化前，SQL语句执行执行时长从6s下降到0.7s，性能提升了10倍。从执行计划来看，表 app_cmf_rank_screen 使用上了性能更好的主键索引，大大减少了存储引擎读取表 app_cmf_rank_screen 的数据量。

四、总结

1. 表创建的时候，不要自己指定表或者字段的字符集和排序规则，使用数据库默认的全局规则就好。
2. 做SQL优化的时候，不用急于分析执行计划，可以 explain 后先 show warnings 查看告警信息，告警信息里一般会有优化的思路。

引用链接

[1] 索引下推: *https://learn.lianglianglee.com/%e4%b8%93%e6%a0%8f/MySQL%e5%ae%9e%e6%88%9845%e8%ae%b2/05%20%20%e6%b7%b1%e5%85%a5%e6%b5%85%e5%87%ba%e7%b4%a2%e5%bc%95%ef%bc%88%e4%b8%8b%ef%bc%89.md*

[2] Block Nested-Loop Join: *https://learn.lianglianglee.com/%e4%b8%93%e6%a0%8f/MySQL%e5%ae%9e%e6%88%9845%e8%ae%b2/34%20%20%e5%88%b0%e5%ba%95%e5%8f%af%e4%b8%8d%e5%8f%af%e4%bb%a5%e4%bd%bf%e7%94%a8join%ef%bc%9f.md*

[3] Index Nested-Loop Join: *https://learn.liangliangle.com/%e4%b8%93%e6%a0%8f/MySQL%e5%ae%9e%e6%88%9845%e8%ae%b2/34%20%20%e5%88%b0%e5%ba%95%e5%8f%af%e4%b8%8d%e5%8f%af%e4%bb%a5%e4%bd%bf%e7%94%a8join%ef%bc%9f.md*

[4] 覆盖索引: *https://learn.liangliangle.com/%e4%b8%93%e6%a0%8f/MySQL%e5%ae%9e%e6%88%9845%e8%ae%b2/05%20%20%e6%b7%b1%e5%85%a5%e6%b5%85%e5%87%ba%e7%b4%a2%e5%bc%95%ef%bc%88%e4%b8%8b%ef%bc%89.md*

Triagen

喜欢作者

SQL性能优化 1 MySQL 17