MySQL数据库SQL优化案例(走错索引)

首页推荐

原创 陈举超 2025-07-17

数据库版本:

MySQL 8.0.36

问题现象:

慢SQL:执行耗时11分钟

```
SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyyyyyy t2 WHERE t1.KEYNAME='c
```

结果集:631

快SQL:0.17秒
CHECKTIME='20240826' 条件执行慢，20240826 的前一天和后一天执行都很快，而且结果集相同

```
SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyyyyyy t2 WHERE t1.KEYNAME='c
SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyyyyyy t2 WHERE t1.KEYNAME='c
```

结果集:631

问题分析

检查数据量:

```
---16242914
select count(*) from xxxxxxxxxxxx;
---1049
select count(*) from yyyyyyyy;
---71018
select count(*) from xxxxxxxxxxxx where CHECKTIME='20240826';
---38462
select count(*) from xxxxxxxxxxxx where CHECKTIME='20240825';
---38362
select count(*) from xxxxxxxxxxxx where CHECKTIME='20240827';
```

对比SQL执行计划:
慢:

```
mysql> explain format=tree SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyy
+----------------------------------------------------------------------------------------------
| EXPLAIN
+----------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=72929 rows=31.7)
    -> Filter: (t2.inspection_flag = 0)  (cost=107 rows=105)
        -> Table scan on t2  (cost=107 rows=1049)
    -> Filter: ((t1.checktime = TIMESTAMP'2024-08-26 00:00:00') and (t1.keyname = 'cpuLoad5
        -> Index lookup on t1 using idx_xxxxxxxxxxxx_hostid (hostid=t2.hostid)  (cost=606
 |
+----------------------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

执行过程:
1.全表扫描t2表（yyyyyyyy）：
执行Table scan on t2（全表扫描）
扫描全部1049行数据
成本：107

陈举超
MVP LV.5
关注
408 文章
388 粉丝
579K+ 浏览量
获得了 966 次点赞
内容获得 259 次评论
获得了 1781 次收藏

TA的专栏
SQLServer数据库
收录 9 篇内容

热门文章
Oracle RAC百问百答-01 RAC篇 2021-08-01 12841浏览
DBA常用命令之东北大乱炖 2021-09-06 9877浏览
Oracle 架构汇总 2022-01-03 9366浏览
记一次惨败的Oracle DBA面试经历 2021-04-14 8756浏览
Oracle DG同步失败问题处理(一) 2021-04-14 7622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
Oracle数据文件被误删除，无任何备份，如何导出剩余的数据？ 2天前 154浏览
Oracle 12.1.2.0 的这个BUG会导致频繁宕机，赶快处理！ 2025-08-03 152浏览
Oracle 23ai Datatype Limits有哪些？ 2025-07-30 72浏览
Oracle AWR夺命33问，你能过几关？ 2025-07-26 356浏览
面试:已知100多个数据库CVE漏洞编号，如何快速查询这些漏洞影响的数据库版... 2025-07-24 235浏览

目录
- 数据库版本:
- 问题现象:
- 问题分析
- 优化方案:

2.过滤t2表：
应用条件t2.xxx_flag = 0
过滤后保留约105行（1049 × 10%）
成本：107（主要消耗在扫描）

3.嵌套循环连接（对t2的每行）：
对t2的105行中的每一行：
a. 使用idx_xxxxxxxxxxxxx_hostid索引查找t1表
b. 每次查找返回约880行（总 105×880=92400行）
c. 对每批880行应用过滤条件：
t1.checktime='20240826'
t1.keyname='xxxLoad5'
d. 过滤后保留约0.302行（最终105*0.302=31.7行）

快：

```
mysql> explain format=tree SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyy
+------------------------------------------------------------------------------------------
| EXPLAIN
+------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=49699 rows=285)
    -> Filter: ((t1.keyname = 'cpuLoad5') and (t1.hostid is not null))  (cost=48701 rows=28
        -> Index lookup on t1 using i_xxxxxxxxxxxxx_checktime (checktime=TIMESTAMP'2024-08-
    -> Filter: (t2.inspection_flag = 0)  (cost=0.25 rows=0.1)
        -> Single-row index lookup on t2 using PRIMARY (hostid=t1.hostid)  (cost=0.25 rows=
 |
+------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

执行顺序与过程：
1.索引扫描t1表：
使用i_xxxxxxxxxxxxx_checktime索引定位checktime='20240825'
返回70288行（某日全量数据）
成本：48701

2.过滤t1表：
应用keyname='xxxLoad5'条件
保留约2851行
成本：48701（主要消耗在索引扫描）

3.嵌套循环连接（对t1的每行）：
对t1的2851行中的每一行：
a. 使用主键索引查找t2表
b. 精确匹配（hostid=t1.hostid）
c. 单行查找（成本仅0.25）
d. 应用xxx_flag=0过滤
e. 保留约0.1行（最终2851*0.1=285行）

对比执行快的执行计划，表关联顺序和选择的索引不一样，hint强制关联顺序和强制走索引速度都很快

```
SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 force index(i_xxxxxxxxxxxxx_chec
631 rows in set (0.23 sec)
```

```
+------------------------------------------------------------------------------------------
| EXPLAIN
+------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=96269 rows=1311)
    -> Filter: ((t1.keyname = 'cpuLoad5') and (t1.hostid is not null))  (cost=91683 rows=13
        -> Index lookup on t1 using i_xxxxxxxxxxxxx_checktime (checktime=TIMESTAMP'2024-08-
    -> Filter: (t2.inspection_flag = 0)  (cost=0.25 rows=0.1)
        -> Single-row index lookup on t2 using PRIMARY (hostid=t1.hostid)  (cost=0.25 rows=
 |
+------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

```
SELECT /*+ JOIN_ORDER(t1,t2)*/ t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1,yyyyyyyy
631 rows in set (0.21 sec)
```

```
+-----------------------------------------------------------------------------------------------
| EXPLAIN
+-----------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=92764 rows=532)
    -> Filter: ((t1.keyname = 'cpuLoad5') and (t1.hostid is not null))  (cost=90904 rows=53
        -> Index lookup on t1 using i_xxxxxxxxxxxxx_checktime (checktime=TIMESTAMP'2024-08-
    -> Filter: (t2.inspection_flag = 0)  (cost=0.25 rows=0.1)
        -> Single-row index lookup on t2 using PRIMARY (hostid=t1.hostid)  (cost=0.25 rows=
 |
+-----------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

慢SQL现象：

1.索引选择不当：

(1)使用hostid索引而非checktime索引;

2.过滤顺序错误：

(1)先关联再过滤（最耗资源步骤）

实际需要数据：checktime='20240826'（单日数据）

3.回表代价高昂：

每次索引查找后需回表验证checktime和keyname

4.嵌套循环放大问题：

小表驱动大表时合理，但此处驱动方式错误

应先用时间条件过滤大表，再关联小表

问题分析:

慢SQL未走高效索引的原因与优化方案

为什么CHECKTIME索引未被使用？

1. 优化器成本估算错误

   MySQL优化器基于统计信息计算成本，但统计信息可能过时或不准确

查看统计信息

```
SHOW INDEX FROM xxxxxxxxxxxxx;
```

```
+---------------+------------+---------------------------+--------------+-------------+----
| Table         | Non_unique | Key_name                  | Seq_in_index | Column_name | Col
+---------------+------------+---------------------------+--------------+-------------+----
| xxxxxxxxxxxxx |          0 | PRIMARY                   |            1 | valuesid    | A
| xxxxxxxxxxxxx |          1 | i_xxxxxxxxxxxxx_keyname   |            1 | keyname     | A
| xxxxxxxxxxxxx |          1 | i_xxxxxxxxxxxxx_checktime |            1 | checktime   | A
| xxxxxxxxxxxxx |          1 | idx_xxxxxxxxxxxxx_hostid  |            1 | hostid      | A
+---------------+------------+---------------------------+--------------+-------------+----
4 rows in set (0.13 sec)
```

检查行数估算

预估CHECKTIME 索引成本比实际偏高

统计信息预估 rows=131052 实际 rows=71018

```
EXPLAIN SELECT COUNT(*) FROM xxxxxxxxxxxxx WHERE CHECKTIME='20240826';
+----+-------------+---------------+------------+------+---------------------------+-------
| id | select_type | table         | partitions | type | possible_keys             | key
+----+-------------+---------------+------------+------+---------------------------+-------
|  1 | SIMPLE      | xxxxxxxxxxxxx | NULL       | ref  | i_xxxxxxxxxxxxx_checktime | i_xxxx
+----+-------------+---------------+------------+------+---------------------------+-------
1 row in set, 1 warning (0.02 sec)
```

统计信息预估 rows=70288 实际 rows=38462

```
EXPLAIN SELECT COUNT(*) FROM xxxxxxxxxxxxx WHERE CHECKTIME='20240825';
+----+-------------+---------------+------------+------+---------------------------+-------
| id | select_type | table         | partitions | type | possible_keys             | key
+----+-------------+---------------+------------+------+---------------------------+-------
|  1 | SIMPLE      | xxxxxxxxxxxxx | NULL       | ref  | i_xxxxxxxxxxxxx_checktime | i_xxxx
+----+-------------+---------------+------------+------+---------------------------+-------
1 row in set, 1 warning (0.00 sec)
```

可能是统计信息不准或表碎片引起的

查询表的碎片率不是很高。

预估表关联成本偏低
统计信息预估: rows=923172 实际 rows=16242099

```
EXPLAIN format=tree SELECT COUNT(*) FROM xxxxxxxxxxxx INNER JOIN yyyyyyyy ON xxxxxxxxxxxx
+-----------------------------------------------------------------------------------------
| EXPLAIN
+-----------------------------------------------------------------------------------------
| -> Aggregate: count(0)  (cost=185369 rows=1)
    -> Nested loop inner join  (cost=93052 rows=923172)
        -> Index scan on yyyyyyyy using i_yyyyyyyy_ip  (cost=107 rows=1049)
        -> Covering index lookup on xxxxxxxxxxxx using idx_xxxxxxxxxxxx_hostid (hostid=yy
 |
+-----------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

所以当达到某个CHECKTIME数据量阈值时，优化器认为 idx_xxxxxxxxxxxx_hostid 索引比 i_xxxxxxxx xxxxx_checktime 索引成本低。

检查统计信息
表统计信息

```
mysql> show table status like 'xxxxxxxxxxxx';
+---------------+--------+---------+------------+----------+----------------+-------------+
| Name          | Engine | Version | Row_format | Rows     | Avg_row_length | Data_length |
+---------------+--------+---------+------------+----------+----------------+-------------+
| xxxxxxxxxxxx | InnoDB |      10 | Dynamic    | 15469519 |             92 |  1438646272 |
+---------------+--------+---------+------------+----------+----------------+-------------+
1 row in set (0.06 sec)

mysql> select * from information_schema.tables where table_schema='cjc' and table_name='xxx
+---------------+-------------+---------------+------------+--------+---------+-----------
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME    | TABLE_TYPE | ENGINE | VERSION | ROW_FORMAT
+---------------+-------------+---------------+------------+--------+---------+-----------
| def           | cjc          | xxxxxxxxxxxx | BASE TABLE | InnoDB |      10 | Dynamic
+---------------+-------------+---------------+------------+--------+---------+-----------
1 row in set (0.00 sec)
```

索引统计信息

```
show index from xxxxxxxxxxxx;
+---------------+------------+---------------------------+--------------+-------------+----
| Table         | Non_unique | Key_name                  | Seq_in_index | Column_name | Col
+---------------+------------+---------------------------+--------------+-------------+----
| xxxxxxxxxxxx |          0 | PRIMARY                   |            1 | valuesid    | A
| xxxxxxxxxxxx |          1 | i_xxxxxxxxxxxx_keyname   |            1 | keyname     | A
| xxxxxxxxxxxx |          1 | i_xxxxxxxxxxxx_checktime |            1 | checktime   | A
| xxxxxxxxxxxx |          1 | idx_xxxxxxxxxxxx_hostid  |            1 | hostid      | A
+---------------+------------+---------------------------+--------------+-------------+----
4 rows in set (0.02 sec)

select * from information_schema.statistics where table_schema='cjc' and table_name='xxxxxx
+---------------+-------------+---------------+------------+-------------+---------------
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME    | NON_UNIQUE | INDEX_SCHEMA | INDEX_NAME
+---------------+-------------+---------------+------------+-------------+---------------
| def           | cjc          | xxxxxxxxxxxx |          1 | cjc          | i_xxxxxxxxxxx
| def           | cjc          | xxxxxxxxxxxx |          1 | cjc          | i_xxxxxxxxxxx
| def           | cjc          | xxxxxxxxxxxx |          1 | cjc          | idx_xxxxxxxxx
| def           | cjc          | xxxxxxxxxxxx |          0 | cjc          | PRIMARY
+---------------+-------------+---------------+------------+-------------+---------------
4 rows in set (0.01 sec)
```

列直方图

```
mysql> SELECT * FROM information_schema.column_statistics where SCHEMA_NAME='cjc' and table
Empty set (0.00 sec)
```

优化方案:

1.改写SQL
把on条件提前，然后再where条件。或者把表关联改成子查询，想先过滤再关联。但是执行计划都没有变化，SQL还是执行很慢。

2.统计信息、直方图
重新收集了表统计信息，checktime 和 hostid列直方图(列数据有倾斜)，执行计划都没有变化，SQL还是执行很慢。

```
show variables like 'innodb_stats_persistent';
set global innodb_stats_persistent_sample_pages=100;
ANALYZE TABLE xxxxxxxxxxxx;
ANALYZE TABLE xxxxxxxxxxxx UPDATE HISTOGRAM ON checktime;
ANALYZE TABLE xxxxxxxxxxxx UPDATE HISTOGRAM ON hostid;
SELECT * FROM information_schema.column_statistics where SCHEMA_NAME='cjc' and table_name='
```

3.复合索引
覆盖where列和查询列，checktime列在索引最左测
没测试，应该会有效果

4.重建表
优化器预估的checktime列索引偏高，怀疑和表碎片有关，尝试重建表。

```
mysql> alter table xxxxxxxxxxxx engine=InnoDB;
Query OK, 0 rows affected (4 min 10.61 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

重建表后，执行计划发生变化，SQL执行速度由11分钟，降到0.19秒。

新执行计划如下:

```
mysql> explain format=tree SELECT t1.HOSTID,t1.KEYATTR,t1.VALUE FROM xxxxxxxxxxxx t1 ,yyyy
+------------------------------------------------------------------------------------------
| EXPLAIN
+------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=111499 rows=635)
    -> Filter: ((t1.checktime = TIMESTAMP'2024-08-26 00:00:00') and (t1.keyname = 'cpuLoad5
        -> Intersect rows sorted by row ID  (cost=109275 rows=6356)
            -> Index range scan on t1 using i_xxxxxxxxxxxxx_checktime over (checktime = '20
            -> Index range scan on t1 using i_xxxxxxxxxxxxx_keyname over (keyname = 'cpuLoc
    -> Filter: (t2.inspection_flag = 0)  (cost=0.25 rows=0.1)
        -> Single-row index lookup on t2 using PRIMARY (hostid=t1.hostid)  (cost=0.25 rows=
 |
+------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

5.分区表
改成分区表，按月分区
没执行，应该也会有效果

###chenjuchao 20250717###
欢迎关注我的公众号《**IT小Chen**》

墨力计划 mysql

评论

分享你的看法，一起交流吧～

老钢炮 LV.2
收集下表的统计信息是不是也可以的？不用重建表呢
6天前 点赞 2

陈举超 MVP LV.5
@老钢炮: 只收集统计信息不行，执行计划没变化
6天前 点赞 回复

老钢炮 LV.2
@陈举超 嗯行，之前一般收集统计信息好像可以解决，下次碰到类似场景我也试试看
3天前 点赞 回复

相关阅读

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/25期）
墨天轮小助手　469次阅读　2025-07-25 15:54:18

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/17期）
墨天轮小助手　436次阅读　2025-07-17 15:31:18

墨天轮「实操看我的」数据库主题征文活动启动
墨天轮编辑部　379次阅读　2025-07-22 16:11:27

深度解析MySQL的半连接转换
听见风的声音　205次阅读　2025-07-14 10:23:00

MySQL 9.4.0 正式发布，支持 RHEL 10 和 Oracle Linux 10
严少安　199次阅读　2025-07-23 01:21:32

索引条件下推和分区--一条SQL语句执行计划的分析
听见风的声音　196次阅读　2025-07-23 09:22:58

null和子查询--not in和not exists怎么选择？
听见风的声音　182次阅读　2025-07-21 08:54:19

使用 MySQL Clone 插件为MGR集群添加节点
黄山谷　163次阅读　2025-07-23 22:04:19

MySQL 8.0.40：字符集革命、窗口函数效能与DDL原子性实践
shunwah　141次阅读　2025-07-15 15:27:19

MySQL 数字处理函数ROUND和TRUNCATE
CuiHulong　135次阅读　2025-07-31 09:59:58