

# 【MySQL】SQL性能分析 (七)

🚗 MySQL学习·第七站~

📌 本文已收录至专栏：[MySQL通关路](#)

❤️ 文末附全文思维导图，感谢各位点赞收藏支持~

★ 学习汇总贴，超详细思维导图：[【MySQL】学习汇总\(完整思维导图\)](#)

假如我们需要对SQL进行优化，我们就必须对他足够的了解，比如 对哪一类SQL进行优化（增删改查）？每一条SQL的性能怎样（执行耗时）？接下来我们来学习一下常见的几种SQL性能分析手段~

## 一.SQL执行频率

我们可以在使用 `use 数据库名` 命令切换到指定数据库之后，通过 `show [session|global] status` 命令可以查看服务器状态信息。

```
1  -- 切换到指定数据库
2  use my_db;
3  -- |查看当前数据库状态
4  show GLOBAL STATUS;
```

信息	结果1	概况	状态
	Variable_name	Value	
▶	Aborted_clients	9	
	Aborted_connects	12	
	Acl_cache_items_count	0	
	Binlog_cache_disk_use	0	
	Binlog_cache_use	141	
	Binlog_stmt_cache_disk_u	0	
	Binlog_stmt_cache_use	0	
	Bytes_received	27138609	
	Bytes sent	202276678	

CSDN @观止study

或者直接使用如下指令，模糊匹配查询当前数据库的 `INSERT、UPDATE、DELETE、SELECT` 的访问频次：

```
-- session 是查看当前会话
-- global 是查询全局数据
SHOW GLOBAL STATUS LIKE 'Com_____';
```

```
6 -- 查看本次MySQL服务开启
7 -- | (或重置) 到现在总请求数
8 SHOW GLOBAL STATUS LIKE 'Com_____';
```

信息	结果1	概况	状态
Variable_name	Value		
Com_binlog	0		
Com_commit	947		
Com_delete	70	←	删除次数
Com_import	0		
Com_insert	3570	←	插入次数
Com_repair	0		
Com_revoke	0		
Com_select	41213	←	查询次数
Com_signal	0		
Com_update	107	←	更新次数
Com_xa_end	0		

CSDN @观止study

通过上述指令，我们可以查看到当前数据库到底是以查询为主，还是以增删改为主，从而为数据库优化提供参考依据。 **如果是查询为主，那么就要考虑对数据库的索引进行优化了。**如果是增删改为主，我们可以考虑使用其他手段对其进行优化。

假设我们知道了数据库以查询为主，**我们又该如何定位针对于哪些查询语句进行优化呢？**对此我们可以借助于慢查询日志。

## 二.慢查询日志

慢查询日志记录了**执行时间超过指定参数 (long\_query\_time, 单位: 秒, 默认10秒) 的所有 SQL语句的日志**。MySQL的慢查询日志**默认没有开启**，需要我们手动开启，我们可以查看一下系统变量 `slow_query_log`。

```
mysql> show variables like 'slow_query_log';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | OFF   |
+-----+-----+
1 row in set (0.01 sec)
```

关闭

CSDN @观止study

如果要开启慢查询日志，需要在MySQL的配置文件 (`/etc/my.cnf`) 中配置如下信息：

```
-- 1.开启MySQL慢日志查询开关
slow_query_log = 1

-- 2.设置慢日志的时间为2秒，SQL语句执行时间超过2秒，就会视为慢查询，记录慢查询日志
long_query_time = 2

-- 3.配置完毕之后，重新启动MySQL服务器进行测试，查看慢日志文件中记录的信息
systemctl restart mysqld

-- 4.随后我们可以在/var/lib/mysql/localhost-slow.log中
-- 查看慢日志文件中记录的信息
cat /var/lib/mysql/localhost-slow.log
```

```
# innodb_buffer_pool_size = 128M
#
# Remove the leading "# " to disable binary logging
# Binary logging captures changes between backups and is enabled by
# default. It's default setting is log_bin=binlog
# disable_log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
#
# Remove leading # to revert to previous value for default_authentication_plugin,
# this will increase compatibility with older clients. For background, see:
# https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_authentication_plugin
# default_authentication_plugin=mysql_native_password

datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

#慢查询日志
slow_query_log=1

long_query_time=2

:x
```

**追加配置**

**按 ESC 输入 :x 保存退出**

CSDN @观止study

- 我们可以执行一条比较耗时的SQL语句（耗时超过指定的2s），然后看慢查询日志是否记录了相关信息。

```
mysql> select count(*) from tb_sku;
+-----+
| count(*) |
+-----+
| 10000000 |
+-----+
1 row in set (13.35 sec)
```

耗时超过指定时间

```
/usr/sbin/mysqld, Version: 8.0.26 (MySQL Community Server - GPL). started with:
Tcp port: 3306 Unix socket: /var/lib/mysql/mysql.sock
Time          Id Command      Argument
```

查看慢查询日志发现  
记录了相关SQL执行信息

```
# Time: 2021-10-28T15:45:39.688679Z
# User@Host: root[root] @ localhost [] Id:      8
# Query time: 13.350650  Lock_time: 0.000358 Rows_sent: 1  Rows_examined: 0
use itcast;
SET timestamp=1635435926;
select count(*) from tb_sku;
```

CSDN\_@观止study

如此，通过慢查询日志，我们就可以具体的定位出执行效率比较低的SQL，从而有针对性的进行优化。

### 三.profile详情

`show profiles` 能够帮助我们在做SQL优化时了解到时间都耗费到哪里去了。相对于慢查询日志只可以查看超过指定时间的SQL，它可以帮助我们查看任意时间耗费的SQL执行情况。

不过，在使用之前，我们需要通过`have_profiling` 参数，查看到当前MySQL是否支持profile操作。如果是支持 profile操作的，我们可能还需要手动打开该操作。

```
-- 1.查看当前MySQL是否支持profile操作
SELECT @@have_profiling ;

-- 2.开启profile操作
-- session 当前会话
-- global 全局数据
-- 0 - 关闭, 1 - 开启
SET [ session | global ] profiling = 1;
```

```
mysql> select @@have_profiling;
+-----+
| @@have_profiling |
+-----+
| YES              |
+-----+
1 row in set, 1 warning (0.00 sec)

mysql>
mysql> select @@profiling;
+-----+
| @@profiling |
+-----+
|          0 |
+-----+
1 row in set, 1 warning (0.00 sec)

mysql> set profiling = 1;
Query OK, 0 rows affected, 1 warning
```

支持profile

未开启

设置开启

CSDN @观止study

- 打开开关后，我们所执行的SQL语句，都会被记录执行时间耗费。我们直接执行如下的SQL语句进行测试：

```
select * from tb_user;
select * from tb_user where id = 1;
select * from tb_user where name = '白起';
select count(*) from tb_sku;
```

执行一系列的SQL的操作，然后通过如下指令查看指令的执行耗时：

- 查看**每一条SQL的耗时基本情况**

```
show profiles;
```



```
mysql> show profiles;
```

Query_ID	Duration	Query
id	耗时	
2	0.00041925	SELECT DATABASE()
3	0.01222200	show databases
4	0.00726975	show tables
5	0.00162125	show tables
6	0.00062300	select * from tb_user
7	0.00097975	select count(*) from tb_user
8	0.00053200	select * from tb_user
9	9.94508625	select count(*) from tb_sku
10	0.00366750	select @@have_profiling
11	0.00035200	select @@profiling
12	0.00015650	select @@profiling
13	0.00053075	select * from tb_user
14	0.00062125	select * from tb_user where id = 1
15	0.00741925	select * from tb_user where name = '白起'
16	9.53712800	select count(*) from tb_sku

15 rows in set, 1 warning (0.00 sec)

刚在执行每条SQL耗时

- 查看指定query\_id的SQL语句各个阶段的耗时情况

```
show profile for query query_id;
```

```
mysql> show profile for query 16;
```

Status	Duration
starting	0.000137
Executing hook on transaction	0.000009
starting	0.000028
checking permissions	0.000027
Opening tables	0.000033
init	0.000006
System lock	0.000010
optimizing	0.000022
statistics	0.000015
preparing	0.000013
executing	9.536456
end	0.000025
query end	0.000007
waiting for handler commit	0.000013
closing tables	0.000014
freeing items	0.000048
logging slow query	0.000244
cleaning up	0.000024

18 rows in set, 1 warning (0.01 sec)

根据show profiles;查看指定SQL的query\_id  
随之可以根据query\_id查看指定语句具体情况

- 查看指定query\_id的SQL语句CPU的使用情况

```
show profile cpu for query query_id;
```

CSDN @观止study

CSDN @观止study

```
mysql> show profile cpu for query 16;
```

查看cpu耗时情况

Status	Duration	CPU_user	CPU_system
starting	0.000137	0.000029	0.000105
Executing hook on transaction	0.000009	0.000002	0.000005
starting	0.000028	0.000005	0.000022
checking permissions	0.000027	0.000006	0.000021
Opening tables	0.000033	0.000007	0.000026
init	0.000006	0.000002	0.000005
System lock	0.000010	0.000005	0.000020
optimizing	0.000022	0.000001	0.000004
statistics	0.000015	0.000004	0.000012
preparing	0.000013	0.000002	0.000010
executing	9.536456	5.610130	21.677326
end	0.000025	0.000004	0.000013
query end	0.000007	0.000001	0.000005
waiting for handler commit	0.000013	0.000003	0.000011
closing tables	0.000014	0.000003	0.000011
freeing items	0.000048	0.000011	0.000039
logging slow query	0.000244	0.000034	0.000127
cleaning up	0.000024	0.000005	0.000016

CSDN @观止study

## 四.explain执行计划

通过上述手段我们只能获悉SQL语句的执行耗时情况，它对于SQL的性能只能进行粗略的判断。我们还可以通过 **EXPLAIN** 或者 **DESC** 命令获取 MySQL **如何执行 SELECT 语句的信息**，包括在 SELECT 语句执行过程中表如何连接和连接的顺序，据此更加准确的判断SQL语句的性能。

- 使用语法

-- 直接在select语句之前加上关键字 **explain** 或 **desc**

**EXPLAIN SELECT** 字段列表 **FROM** 表名 **WHERE** 条件...;

```
mysql> explain select * from tb_user where id = 1;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	tb_user	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL

1 row in set, 1 warning (0.00 sec)

CSDN @观止study

Explain 执行计划中**各个字段的含义**:

字段	含义
id	select查询的序列号，表示 <b>查询中执行select子句或者是操作表的顺序</b> (id相同，执行顺序从上到下；id不同，值越大，越先执行)。
select_type	表示 SELECT 的类型，常见的取值有 SIMPLE（简单表，即不使用表连接 或者子查询）、PRIMARY（主查询，即外层的查询）、UNION（UNION 中的第二个或者后面的查询语句）、SUBQUERY（SELECT/WHERE之后包含了子查询）等
type	表示连接类型，性能 <b>由好到差</b> 的连接类型为NULL、system、const、eq_ref、ref、range、index、all。
possible_key	在这张表上 <b>可能会使用到</b> 的索引，一个或多个。
key	<b>实际使用的</b> 索引，如果为NULL，则没有使用索引。
key_len	表示索引中使用的字节数，该值为索引字段最大可能长度，并非实际使用长度，在不损失精确性的前提下，长度越短越好。
rows	MySQL认为必须要执行查询的行数，在innodb引擎的表中，是一个估计值，可能并不总是准确的。
filtered	表示返回结果的行数占需读取行数的百分比，filtered 的值越大越好。

- 对于 type 字段值补充说明：
  - NULL：一般不太可能优化到NULL,除非在查询的时候**不访问任何表**，比如 `Select 'A'`
  - system：一般出现在**访问系统表**时
  - const：一般出现在**使用主键或者唯一索引访问**时
  - ref：一般出现在**使用非唯一性索引访问**时
  - range：一般出现在**使用了非唯一索引,但是范围匹配,比如age > 18**

## 五.全文概览





