# 慢查询

## 什么是慢查询

慢查询日志，顾名思义，就是查询慢的日志，是指mysql记录所有执行超过long\_query\_time参数设定的时间阈值的SQL语句的日志。该日志能为SQL语句的优化带来很好的帮助。默认情况下，慢查询日志是关闭的，要使用慢查询日志功能，首先要开启慢查询日志功能。

## 慢查询配置

### 慢查询基本配置

* slow\_query\_log 启动停止技术慢查询日志
* slow\_query\_log\_file 指定慢查询日志得存储路径及文件（默认和数据文件放一起）
* long\_query\_time 指定记录慢查询日志SQL执行时间得伐值（单位：秒，默认10秒）
* log\_queries\_not\_using\_indexes 是否记录未使用索引的SQL
* log\_output 日志存放的地方【TABLE】【FILE】【FILE,TABLE】

配置了慢查询后，它会记录符合条件的SQL

包括：

* 查询语句
* 数据修改语句
* 已经回滚得SQL

**实操：**

**通过下面命令查看下上面的配置：**

show VARIABLES like '%slow\_query\_log%'

show VARIABLES like '%slow\_query\_log\_file%'

show VARIABLES like '%long\_query\_time%'

show VARIABLES like '%log\_queries\_not\_using\_indexes%'

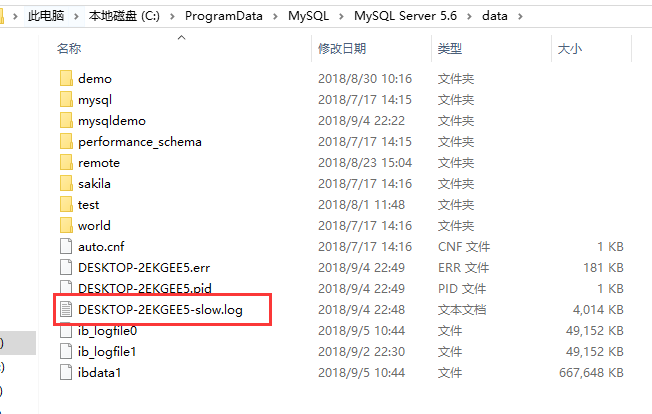
show VARIABLES like 'log\_output'

set global long\_query\_time=0; ---默认10秒，这里为了演示方便设置为0

set GLOBAL slow\_query\_log = 1; --开启慢查询日志

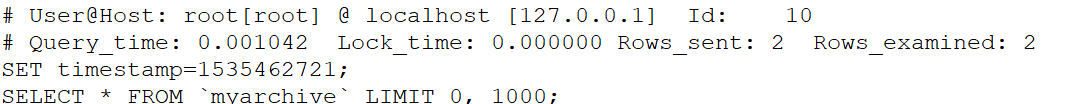
set global log\_output='FILE,TABLE' --项目开发中日志只能记录在日志文件中，不能记表中

设置完成后，查询一些列表可以发现慢查询的日志文件里面有数据了。

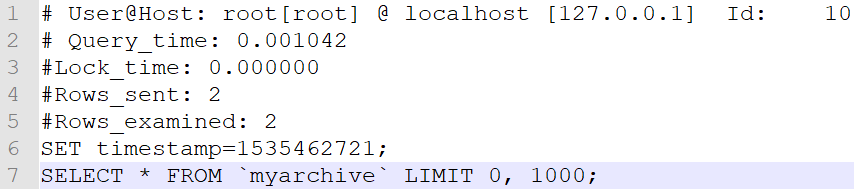


### 慢查询解读

从慢查询日志里面摘选一条慢查询日志，数据组成如下



把为解读放吧，慢查询格式显示



第一行：用户名 、用户的IP信息、线程ID号

第二行：执行花费的时间【单位：毫秒】

第三行：执行获得锁的时间

第四行：获得的结果行数

第五行：扫描的数据行数

第六行：这SQL执行的具体时间

第七行：具体的SQL语句

## 慢查询分析

慢查询的日志记录非常多，要从里面找寻一条查询慢的日志并不是很容易的事情，一般来说都需要一些工具辅助才能快速定位到需要优化的SQL语句，下面介绍两个慢查询辅助工具

### Mysqldumpslow

常用的慢查询日志分析工具，汇总除查询条件外其他完全相同的SQL，并将分析结果按照参数中所指定的顺序输出。

**语法：**

mysqldumpslow -s r -t 10 slow-mysql.log

-s order (c,t,l,r,at,al,ar)

c:总次数

t:总时间

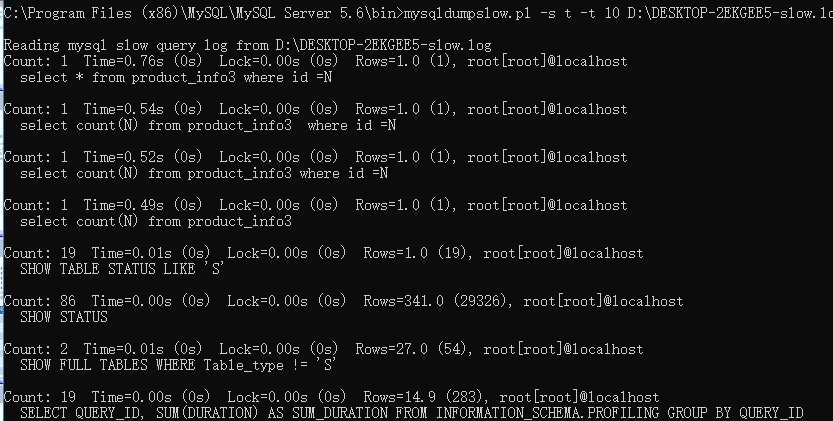
l:锁的时间

r:总数据行

at,al,ar :t,l,r平均数 【例如：at = 总时间/总次数】

-t top 指定取前面几天作为结果输出

mysqldumpslow.pl -s t -t 10 D:\DESKTOP-2EKGEE5-slow.log



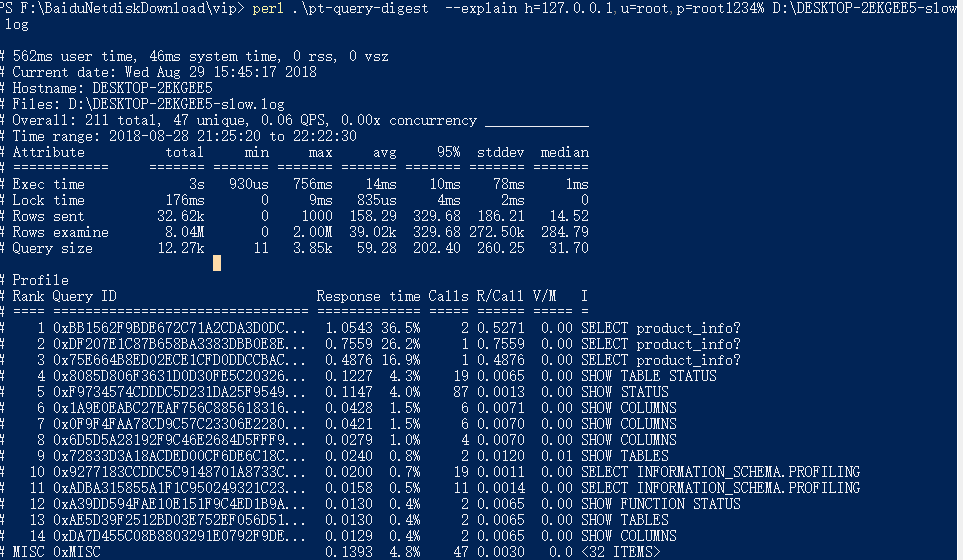
### pt\_query\_digest

是用于分析mysql慢查询的一个工具，与mysqldumpshow工具相比，py-query\_digest 工具的分析结果更具体，更完善。

有时因为某些原因如权限不足等，无法在服务器上记录查询。这样的限制我们也常常碰到。

首先来看下一个命令

perl .\pt-query-digest --explain h=127.0.0.1,u=root,p=root1234% D:\DESKTOP-2EKGEE5-slow.log



汇总的信息【总的查询时间】、【总的锁定时间】、【总的获取数据量】、【扫描的数据量】、【查询大小】

Response: 总的响应时间。

time: 该查询在本次分析中总的时间占比。

calls: 执行次数，即本次分析总共有多少条这种类型的查询语句。

R/Call: 平均每次执行的响应时间。

Item : 查询对象

#### 扩展阅读：

****pt-query-digest语法及重要选项****

pt-query-digest [OPTIONS] [FILES] [DSN]

* --create-review-table 当使用--review参数把分析结果输出到表中时，如果没有表就自动创建。
* --create-history-table 当使用--history参数把分析结果输出到表中时，如果没有表就自动创建。
* --filter 对输入的慢查询按指定的[字符串](http://www.php.cn/wiki/57.html" \t "http://www.php.cn/_blank)进行匹配过滤后再进行分析
* --limit 限制输出结果百分比或数量，默认值是20,即将最慢的20条语句输出，如果是50%则按总响应时间占比从大到小排序，输出到总和达到50%位置截止。
* --host mysql服务器地址
* --user mysql用户名
* --password mysql用户密码
* --history 将分析结果保存到表中，分析结果比较详细，下次再使用--history时，如果存在相同的语句，且查询所在的时间区间和历史表中的不同，则会记录到数据表中，可以通过查询同一CHECKSUM来比较某类型查询的历史变化。
* --review 将分析结果保存到表中，这个分析只是对查询条件进行参数化，一个类型的查询一条记录，比较简单。当下次使用--review时，如果存在相同的语句分析，就不会记录到数据表中。
* --output 分析结果输出类型，值可以是report(标准分析报告)、slowlog(Mysql slow log)、[json](http://www.php.cn/wiki/1488.html" \t "http://www.php.cn/_blank)、json-anon，一般使用report，以便于阅读。
* --since 从什么时间开始分析，值为字符串，可以是指定的某个”yyyy-mm-dd [hh:mm:ss]”格式的时间点，也可以是简单的一个时间值：s(秒)、h(小时)、m(分钟)、d(天)，如12h就表示从12小时前开始统计。
* --until 截止时间，配合—since可以分析一段时间内的慢查询。

****分析pt-query-digest输出结果****

第一部分：总体统计结果

* Overall：总共有多少条查询
* Time range：查询执行的时间范围
* unique：唯一查询数量，即对查询条件进行参数化以后，总共有多少个不同的查询
* total：总计 min：最小 max：最大 avg：平均
* 95%：把所有值从小到大排列，位置位于95%的那个数，这个数一般最具有参考价值
* median：中位数，把所有值从小到大排列，位置位于中间那个数

# 该工具执行日志分析的用户时间，系统时间，物理内存占用大小，虚拟内存占用大小

# 340ms user time, 140ms system time, 23.99M rss, 203.11M vsz

# 工具执行时间

# Current date: Fri Nov 25 02:37:18 2016

# 运行分析工具的主机名

# Hostname: localhost.localdomain

# 被分析的文件名

# Files: slow.log

# 语句总数量，唯一的语句数量，QPS，并发数

# Overall: 2 total, 2 unique, 0.01 QPS, 0.01x concurrency

# 日志记录的时间范围

# Time range: 2016-11-22 06:06:18 to 06:11:40

# 属性 总计 最小 最大 平均 95% 标准 中等

# Attribute total min max avg 95% stddev median

# ============ ======= ======= ======= ======= ======= ======= =======

# 语句执行时间

# Exec time 3s 640ms 2s 1s 2s 999ms 1s

# 锁占用时间

# Lock time 1ms 0 1ms 723us 1ms 1ms 723us

# 发送到客户端的行数

# Rows sent 5 1 4 2.50 4 2.12 2.50

# select语句扫描行数

# Rows examine 186.17k 0 186.17k 93.09k 186.17k 131.64k 93.09k

# 查询的字符数

# Query size 455 15 440 227.50 440 300.52 227.50

****第二部分：查询分组统计结果****

* Rank：所有语句的排名，默认按查询时间降序排列，通过--order-by指定
* Query ID：语句的ID，（去掉多余空格和文本字符，计算[hash](http://www.php.cn/wiki/762.html" \t "http://www.php.cn/_blank)值）
* Response：总的响应时间
* time：该查询在本次分析中总的时间占比
* calls：执行次数，即本次分析总共有多少条这种类型的查询语句
* R/Call：平均每次执行的响应时间
* V/M：响应时间Variance-to-mean的比率
* Item：查询[对象](http://www.php.cn/wiki/60.html" \t "http://www.php.cn/_blank)

# Profile

# Rank Query ID Response time Calls R/Call V/M Item

# ==== ================== ============= ===== ====== ===== ===============

# 1 0xF9A57DD5A41825CA 2.0529 76.2% 1 2.0529 0.00 SELECT

# 2 0x4194D8F83F4F9365 0.6401 23.8% 1 0.6401 0.00 SELECT wx\_member\_base

第三部分：每一种查询的详细统计结果

由下面查询的详细统计结果，最上面的[表格](http://www.php.cn/code/5947.html" \t "http://www.php.cn/_blank)列出了执行次数、最大、最小、平均、95%等各项目的统计。

* ID：查询的ID号，和上图的Query ID对应
* Databases：数据库名
* Users：各个用户执行的次数（占比）
* Query\_time distribution ：查询时间分布, 长短体现区间占比，本例中1s-10s之间查询数量是10s以上的两倍。
* Tables：查询中涉及到的表
* Explain：SQL语句

# Query 1: 0 QPS, 0x concurrency, ID 0xF9A57DD5A41825CA at byte 802

# This item is included in the report because it matches --limit.

# Scores: V/M = 0.00

# Time range: all events occurred at 2016-11-22 06:11:40

# Attribute pct total min max avg 95% stddev median

# ============ === ======= ======= ======= ======= ======= ======= =======

# Count 50 1

# Exec time 76 2s 2s 2s 2s 2s 0 2s

# Lock time 0 0 0 0 0 0 0 0

# Rows sent 20 1 1 1 1 1 0 1

# Rows examine 0 0 0 0 0 0 0 0

# Query size 3 15 15 15 15 15 0 15

# String:

# Databases test

# Hosts 192.168.8.1

# Users mysql

# Query\_time distribution

# 1us

# 10us

# 100us

# 1ms

# 10ms

# 100ms

# 1s ################################################################

# 10s+

# EXPLAIN /\*!50100 PARTITIONS\*/

select sleep(2)\G