链滴

# explain 查看 SQL 的执行计划

作者：Hefery

原文链接：https://ld246.com/article/1641490453083

来源网站：链滴

# 为什么要查看SQL的执行计划

- 通过执行计划对SQL使用索引的情况分析
- 通过慢查询日志获取有性能问题的SQL

# 如何使用explain命令

在查询语句前加上 explain 即可，如explain select * from t_test

- id：执行SELECT语句的顺序。id值相同时，执行顺序由上至下；id值越大优先级越高，越先被执行

在有子查询是用得到，因为 MySQL 优化器可以对 SQL 的执行顺序进行编排。单查询往往是1

- select_type：

SIMPLE：不包含子查询或是UNION操作的查询

PRIMARY：查询中如果包含任何子查询，那么最外层的查询则被标记为PRIMARY

SUBQUERY：SELECT列表中的子查询

DEPENDENT SUBQUERY：子查询中的第一个SELECT，依赖外部结果的子查询

UNION：Union操作的第二个或是之后的查询的值为union

DEPENDENT UNION：当UNION做为子查询时，第二或是第二个后的查询的select_type值

UNION RESULT：UNION产生的结果集

DERIVED：出现在FROM子句中的子查询

- table：输出数据行所在的表的名称

<unionM,N> 由ID为M，N查询union产生的结果集

<derivedN>/<subqueryN> 由ID为N的查询产生的结果

- partitions：匹配的分区

对于分区表，显示查询的分区ID

对于非分区表，显示为NULL

- type：表的连接类型

ALL：Full Table Scan全表扫描，遍历全表以找到匹配的行，这是效率最差的联接方式

index：Full Index Scan全索引扫描，index与ALL区别为index类型只遍历索引树

range：索引范围扫描，常见于between、>、< 的查询条件，只检索给定范围的行，使用一个索引择行

index_merge：使用了索引合并优化方法

ref_of_null：类似于ref类型的查询，但是附加了对NULL值列的查询

ref：非唯一索引查找，上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

eq_ref：唯一索或主键引查找，对于每个索引键，表中只有一条记录与之匹配

const：表中有且只有一个匹配的行时使用，如对主键或是唯一索引的查询，这是效率最高的联接方式

system：这是const联接类型的一个特例，当查询的表只有一行时使用

NULL：在优化过程中分解语句，执行时甚至不用访问表或索引

- extra：MySQL解决查询的详细信息

原文链接：explain 查看 SQL 的执行计划

Distinct：优化distinct操作，在找到第一匹配的元组后即停止找同样值的动作

Not exists：使用not exists来优化查询

Using filesort：使用额外操作进行排序，通常会出现在order by或group by查询中

Using index：使用了覆盖索引进行查询

Using temporary ： MySQL需要使用临时表来处理查询，常见于排序、子查询和分组查询

Using where：需要在MySQL服务器层使用WHERE条件来过滤数据

select tables optimized away：直接通过索引来获得数据，不用访问表

- possible_keys：MySQL能使用那些索引来优化查询

查询列所涉及到的列上的索引都会被列出，但不一定会被使用

- key：查询优化器优化查询实际所使用的索引

如果没有可用的索引，则显示为NULL，如查询使用了覆盖索引，则该索引仅出现在Key列中

- key_len：索引中使用的字节数，可通过该列计算查询中使用的索引的长度

key_len显示的值为索引字段的最大可能长度，并非实际使用长度，即key_len是根据表定义计算而得不是通过表内检索出的

- ref：列与索引的比较，表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值
- rows：MySQL通过索引统计信息，估算所需读取的行数，rows值大小是个统计抽样结果，并不十准确
- filtered：返回结果的行数占需读取行数的百分比

Filtered列的值越大越好

Filtered列的值依赖说统计信息

# 执行计划的限制

无法展示存储过程，触发器，UDF对查询的影响

无法使用 explain 对存储过程进行分析

早期版本的 MySQL 只支持对 SELECT 语句进行分析

# 捕获有问题的SQL

- 启动慢查询日志

```
set global slow_query_log_file = /sql_log/slow_log.log;
set global log_queries_not_using_indexes=on;  # 未使用索引的SQL记录日志
set global long_query time=0.001;         # 抓取执行超过多少时间的SQL，单位：秒
set global low_query_log=on;
```

- 分析慢查询日志
  - mysqldumpslow

```
# 查看帮助信息
$ mysqldumpslow.pl --help
Usage: mysqldumpslow [ OPTS... ] [ LOGS... ]
```

```
Parse and summarize the MySQL slow query log. Options are

  --verbose    verbose
  --debug      debug
  --help       write this text to standard output

  -v           verbose
  -d           debug
  -s ORDER     what to sort by (al, at, ar, c, l, r, t), 'at' is default
               al: average lock time
               ar: average rows sent
               at: average query time
                c: count
                l: lock time
                r: rows sent
                t: query time
  -r           reverse the sort order (largest last instead of first)
  -t NUM       just show the top n queries
  -a           don't abstract all numbers to N and strings to 'S'
  -n NUM       abstract numbers with at least n digits within names
  -g PATTERN   grep: only consider stmts that include this string
  -h HOSTNAME  hostname of db server for *-slow.log filename (can be wildcard),
               default is '*', i.e. match all
  -i NAME      name of server instance (if using mysql.server startup script)
  -l           don't subtract lock time from total time
```

- pt-query-digest

```
# 查看帮助信息
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，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，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、jsn-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小时前开始计
--until 截止时间，配合—since可以分析一段时间内的慢查询

# 案例解读
# 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
# 343ms user time, 78ms system time, 0 rss, 0 vsz
# 工具执行时间
# Current date: Thu Mar 29 15:51:38 2018
```

```
# 运行分析工具的主机名
# Hostname: NB2015041602
# 被分析的文件名
# Files: /d/xampp/mysql/data/NB2015041602-slow.log
# 语句总数量，唯一的语句数量，QPS，并发数
# Overall: 5 total, 3 unique, 0.00 QPS, 0.05x concurrency _________________
# 日志记录的时间范围
# Time range: 2018-03-28 14:02:06 to 14:22:10
# 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           60s     10s     17s     12s     17s      3s     11s
# 锁占用时间
# Lock time           1ms       0   500us   200us   490us   240us       0
# 发送到客户端的行数
# Rows sent            50      10      10      10      10       0      10
# select语句扫描行数
# Rows examine     629.99k  45.43k 146.14k 126.00k 143.37k  39.57k 143.37k
# 查询的字符数
# Query size         2.81k     235   1.36k  575.40   1.33k  445.36  234.30
# String:
# Databases    database_base
# Hosts        localhost
# Users        root
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms
# 100ms
#    1s
#  10s+  ###############################################################

# Tables
#    SHOW TABLE STATUS FROM `database_base` LIKE 'table_list1'\G
#    SHOW CREATE TABLE `database_base`.`table_list1`\G
#    SHOW TABLE STATUS FROM `database_base` LIKE 'user_list'\G
#    SHOW CREATE TABLE `database_base`.`user_list`\G
# EXPLAIN /*!50100 PARTITIONS*/
select SQL_CALC_FOUND_ROWS al.*, ul.Alias as userName
       FROM table_list1 al
       LEFT JOIN user_list ul ON ul.ID = al.UserId
        WHERE TRUE  AND (al.SupportCountrys LIKE '%%')

        limit 80, 10\G


直接分析慢查询文件
pt-query-digest  slow.log > slow_report.log

分析最近12小时内的查询
```

```
pt-query-digest  --since=12h  slow.log > slow_report2.log

分析指定时间范围内的查询
pt-query-digest slow.log --since '2017-01-07 09:30:00' --until '2017-01-07 10:00:00'> > slow_
eport3.log

分析含有select语句的慢查询
pt-query-digest --filter '$event->{fingerprint} =~ m/^select/i' slow.log> slow_report4.log

针对某个用户的慢查询
pt-query-digest --filter '($event->{user} || "") =~ m/^root/i' slow.log> slow_report5.log

查询所有全表扫描或full join的慢查询
pt-query-digest --filter '(($event->{Full_scan} || "") eq "yes") ||(($event->{Full_join} || "") eq "ye
")' slow.log> slow_report6.log

把查询保存到query_review表
pt-query-digest --user=root –password=abc123 --review  h=localhost,D=test,t=query_review
-create-review-table  slow.log

把查询保存到query_history表
pt-query-digest  --user=root –password=abc123 --review  h=localhost,D=test,t=query_histor
--create-review-table  slow.log_0001
pt-query-digest  --user=root –password=abc123 --review  h=localhost,D=test,t=query_histor
--create-review-table  slow.log_0002

通过tcpdump抓取的tcp协议数据，然后分析
tcpdump -s 65535 -x -nn -q -tttt -i any -c 1000 port 3306 > mysql.tcp.txt
pt-query-digest --type tcpdump mysql.tcp.txt> slow_report9.log

分析biglog
mysqlbinlog mysql-bin.000093 > mysql-bin000093.sql
pt-query-digest  --type=binlog  mysql-bin000093.sql > slow_report10.log

分析general log
pt-query-digest  --type=genlog  localhost.log > slow_report11.log
```