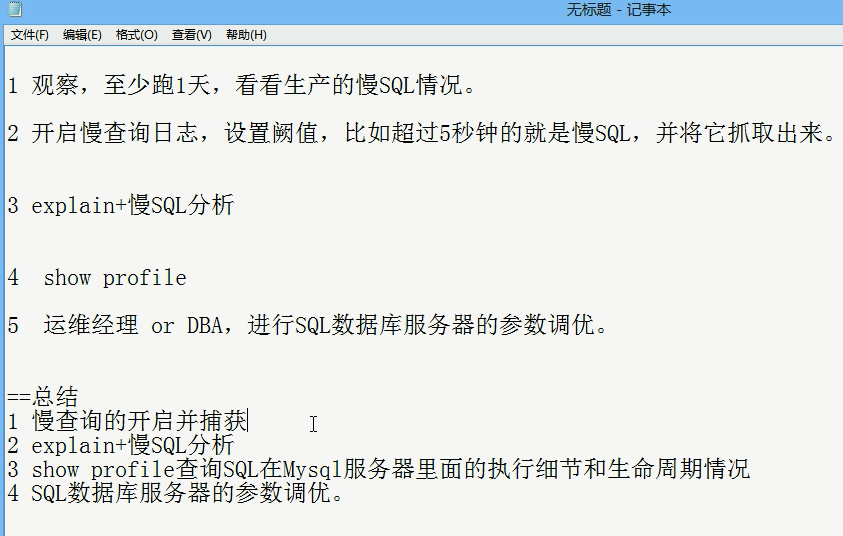
# .总述



# 查询优化

## \*永远小表驱动大表

类似嵌套循环Nested Loop

### In 和exists的区别

1. Select \* from A where id in (select id from B)

如果子查询结果集小于父查询结果集，in比exists效率高

1. Select \* from A where exists (select 1 from B where A.id=B.id)

如果子查询结果集大于父查询结果集，exists比in效率高

## Order by优化

ORDER BY子句 ，尽量使用Index方式排序，避免使用FileSort方式排序

### MySQL支持两种方式排序

FileSort和Index，Index效率高

Order By满足两种情况会使用Index方式排序

1. Order By语句使用索引最左前列
2. 使用Where子句与Order By子句条件列满足索引最左前列

尽可能在索引列上完成排序操作，一定要遵循最佳左前缀原则。

### Filesort

#### 双路排序

MySQL4.1之前使用双路排序，字面意思是，扫描两次磁盘，最终得到数据，读取行指针个order by 列，对他们进行排序，然后扫描已经排序好的列表，按照列表中的值重新从列表中读取对应的数据输出。

从磁盘取出排序字段后，在buffer进行排序，在从磁盘取其他字段。

#### 单路排序

从磁盘读取查询需要的所有列，按照order by列在buffer对它们进行排序，然后扫描排序后的列表进行输出。它的效率会更快一些，避免了第二次读取数据。并且把随机IO变成了顺序IO，但是它会使用更多的空间，因为它把每一行都保存在内存中了。

#### 算法结论

由于单路是后出的算法，总体而言要好于多路，但是单路有问题。

在sort\_buffer中，方法B比方法A要多占用很多空间，因为方法B就是把所有字段都取出，所以有可能取出的数据的总大小超出了sort\_buffer的容量，导致每次只能取sort\_buffer容量大小的数据，进行排序（创建tmp文件，多路合并），排完再取sort\_buffer容量大小，再排……从而对此IO

#### 优化策略

1. 增大sort\_buffer\_size参数的设置
2. 增大max\_length\_for\_sort\_data参数的设置

#### 提高Order By的速度

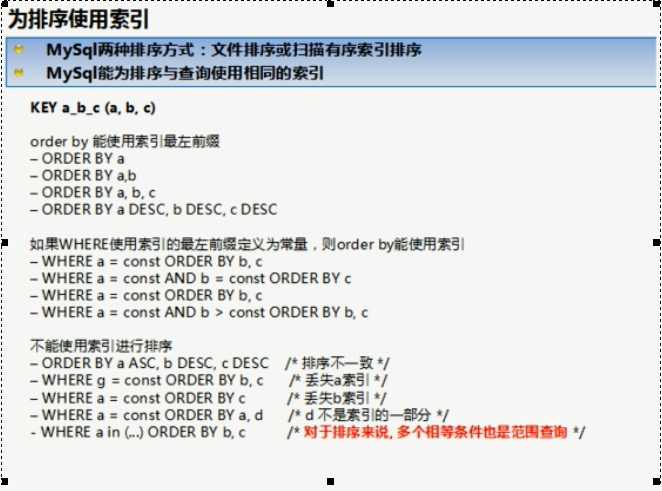
1. Order by时select \*是一个大忌，只需查询需要的字段，这点非常重要，这里的影响是：
   1. 当查询的字段大小总和小于Max\_length\_for\_sort\_data而且排序字段不是Text|Bolb类型时，会用改进后的算法——单路排序，否则用老算法——多路排序。
   2. 两种算法的数据都有可能超出sort\_buffer的容量，超出之后，会创建tmp文件进行合并排序，导致多次I/O，但是用单路排序算法的风险会更大一些，所以要提高sort\_buffer\_size。
2. 尝试提高sort\_buffer\_size

不管哪种算法，提高这个参数都会提高效率，当然，要根据系统的能力去提高，因为这个参数是针对每个进程的。

1. 尝试提高max\_length\_for\_sort\_data

提高这个参数，会增加用改进算法的概率。但是如果设置的太高，数据总容量超出sort\_buffer的概率就增大，明显症状是高的磁盘I/O活动和低的处理器使用率。

### 为排序使用索引



## Group by优化

类似于order by

1. Group by 实质是先排序后进行分组，遵照索引建的最佳左前缀
2. 当无法使用索引列，增大max\_length\_for\_sort\_data参数设置+增大sort\_buffer\_size参数的设置
3. Wehre高于having，能写在where限定的条件就不要去having限定了。

# .慢查询日志

## 详解

MySQL的慢查询日志是MySQL提供的一种日志记录，它用来记录在MySQL中响应时间超过阙值的语句，具体指运行时间超过long\_query\_time值的SQL，则会被记录到慢查询日志中。

具体值运行时间超过long\_query\_time值的SQL，则会被记录到慢查询日志中。Long\_query\_time的默认值为10，意思是运行10秒以上的语句。

由他来查看哪些SQL超出了我们的最大忍耐时间值，比如一条SQL执行超过5秒钟，我们就算慢SQL，希望能搜集超过5秒的SQL，结合之间explain进行全面分析。

## 用法

### 说明

默认情况下，MySQL数据库没有开启慢查询日志，需要我们手动来设置这个参数。

当然，如果不是调优需要的话，一般不建议启动该参数，因为开启慢查询日志会或多或少带来一定的性能影响。慢查询日志支持将日志记录写入文件。

### 命令行操作

#### 查看开启状态

SHOW VARIABLES LIKE '%slow\_query\_log%';

#### 开启

set global slow\_query\_log=1;

#### 查看慢阙值时间

SHOW VARIABLES LIKE '%slow\_query\_time%';

#### 设置慢阙值时间

set global long\_query\_time=3;

#### 查询系统中有多少条慢查询

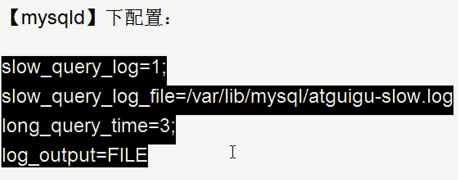
show global status like ‘%Slow\_queries%’;

#### 补充

重新连接或新开一个会话才能看到修改

修改后只对当前数据库有效，重启MySQL后失效，长久有效需要修改配置文件，但是一般不长久开启。

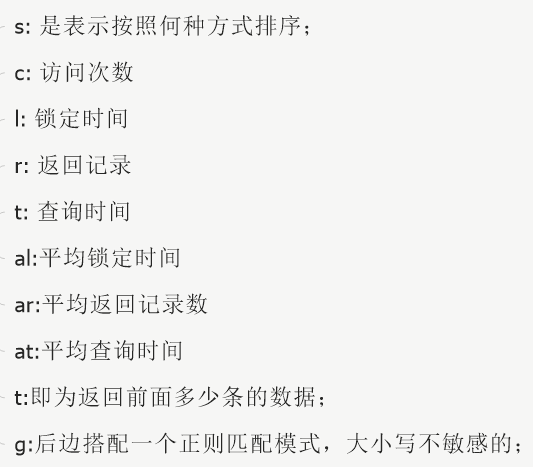
### my.cnf配置



## \*日志分析工具mysqldumpslow

再生产中，如果手工分析日志，查找、分析SQL显然是个体力活，MySQL提供了日志分析工具mysqldumpslow。

### 指令详情



### 实例



# .批量数据脚本

# .Show Profile

是mysql提供可以用来分析当前会话中语句执行的资源消耗情况。可以用于SQL的调优的测量。比如说，一条SQL执行的慢，可以分析出该条慢SQL每一步执行了多少秒。

## 开启Show profile

### 查看状态

Show variables like ‘profiling’;

默认关闭。

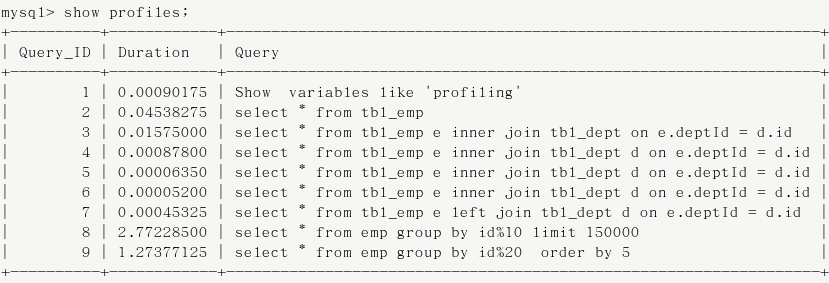
### 设置开启

Set profiling=on;

## 使用show profile

### 查看sql执行记录

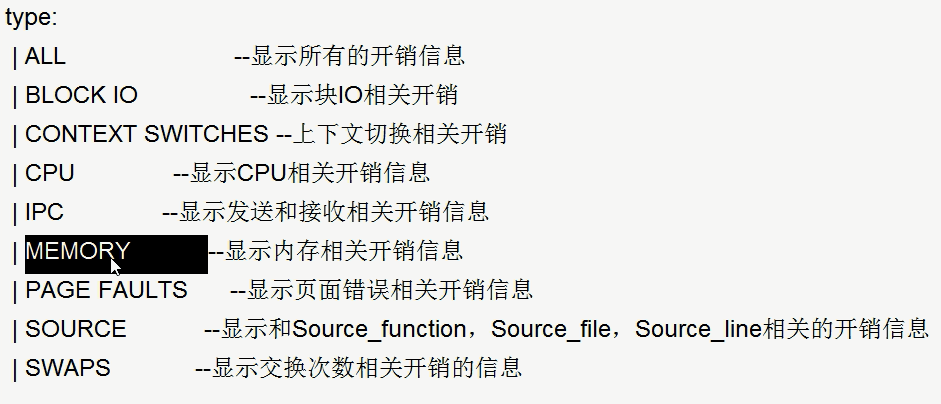
输入指令show profiles;



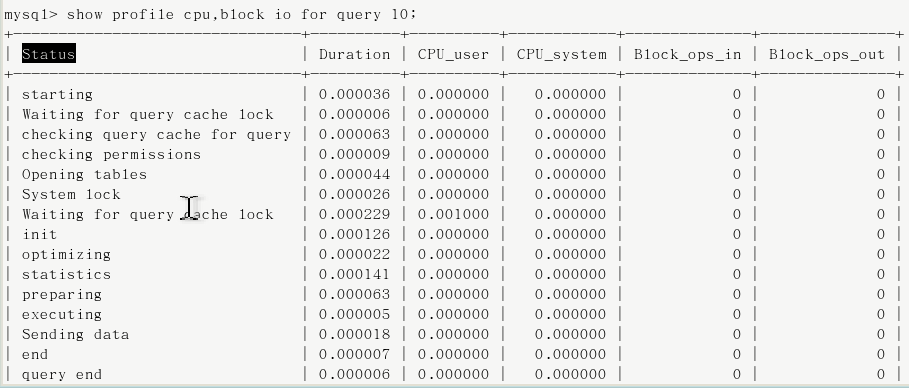
### 诊断SQL

Show profile cpu,block io for query 3;

查看ID为3的SQL的cpu和io使用情况，可以选择



会出现以下表格



这只是一部分，SQL不一样，显示的status也会不太一样，显示部分项的时候需要注意了。

### 惹事的status

#### Converting HEAP toMyISAM

查询结果太大，内存不够用了，往磁盘上搬了。

#### Creating tmp table

创建临时表，拷贝数据到临时表，用完再删除

#### Copying to tmp table on disk

把内存中了临时表复制到磁盘，危险。

#### Locked

# .全局查询日志

**永远不要再生产环境开启该功能！**测试环境下使用，把所有的SQL抓出来看看。推荐使用Show profile方式。

## 配置启用



## 编码启用

