**MySQL慢查日志的开启方式和储存格式**

**怎样MySQL慢查日志对有效率问题的SQL进行监控**

show variables like 'slow\_query\_log';

查询是否开启慢查询日志

show variables like 'log\_queries\_not\_using\_indexes';

查询是否把没有使用索引的查询记录慢查询日志

show variables like 'slow%' ;

查看慢查询日志文件路径

show variables like 'long\_query\_time';

查询超过多长时间记录慢查询日志

set global slow\_query\_log=on;

开启记录慢查询日志

set global slow\_query\_log\_file = '/home/mysql/sql\_log/mysql-slow.log';

设置日志文件路径

set global log\_queries\_not\_using\_indexes=on;

设置记录未使用索引的查询

set global long\_query\_time=1;

设置大于1s的查询记录起来

**日志文件内容格式**

**格式：**

＃User@Host:root[root] @localhost

@Query\_time:0.000024 Lock\_time:0.000000 Rows\_sent:0 Rows\_examined:0

SET timestamp=1402389328;

select ＊ from sys

说明：

执行SQL的主机信息 哪个用户在哪个主机执行

SQL的执行信息 包括查询时间 锁定时间 发送行数 扫描行数

SQL执行时间 执行时间戳

SQL的内容

**慢查询日志分析工具**

**mysqldumpslow (mysqldumpslow MySQL官方的分析工具)**

格式：mysqldumpslow [options] [logfile]

>mysqldumpslow -h 列出指令参数列表

－s 排序

－t num   前num条

exp:

>mysqldumpslow -t 10 -s asc /home/..../slow.log

指令分析的报表内容如下：

Count:1 Time=0.20s (0s)   Lock=0.00s Rows=0.0, root[root]@[127.0.0.1]

select.......from   table

执行的次数  花费的时间  锁定时间  行数  执行用户及主机  及具体SQL内容

**pt-query-digest(慢查日志的分析工具)**

输出到文件

>pt-query-digest slow-log > slow\_log.report

输出到数据库表

>pt-query-digest slow.log -review \

h=127.0.0.1,D=test,p=root,P=3306,u=root,t=query\_review \

--create-reviewtable \

--review-history t=hostname\_slow

>pt-query-digest -h  参数列表

--limit  限定多少条 百分数或数字

－－review 将结果保存到物理表

e.g:

>pt-query-digest -t /home/..../slow.log

**如何通过慢查询日志发现有问题的SQL**

**在分析出来的慢查询日志中主要关注下面几个方面**

1.查询次数多且每次查询占用时间长的SQL即平均每次查询时间较长的

  通常为pt-query-digest分析的前几个查询

2.IO大的SQL即扫描的行数，行数越多

  注意：pt-query-digest分析中的Rows examine项

3.未命中索引的SQL即Rows examine远远大于Row send说明索引命中率低用的全表扫描

  注意：pt-query-digest分析中的Rows examine和Row send 的对比

**通过explain查询和分析SQL的执行计划**

**e.g:**

mysql>explain select customer\_id,first\_name,last\_name from customer;

|id|select\_type|table |type|possible\_keys|key|key\_len|ref|rows|Extra|

+....+...........+.......+.......+.........+.......+...+....+....+..+

|1 |SIMPLE     |customer|ALL|NULL        |NULL...........|671|.......

**explain返回各列的含义**

**table :**显示这一行的数据是哪张表的

**type:**这是重要的列， 显示连接使用了何种类型。从最好到最差的连接类型为

     const  常数查找 唯一索引或唯一主键查找

     eq\_reg 范围查找 主键范围查找等

     ref   一个表是基于某一个表的查找

     range  基于索引的范围查找

     index  基于索引的扫描

     ALL    全表扫描

**possible\_keys:**显示可能应用在这张表中的索引。如果为空，没有可能的索引

**key:**实际使用的索引。如果为NULL，则没有使用索引

**key\_len:**使用索引的长度。在不损失精确性的情况下，长度越短越好

**ref:**显示索引的哪一列被使用了，如果可能的话，是一个常数

**rows:**MySQL认为必须检查的用来请求数据的的行数

**Extra:**扩展列 需要注意的返回值

    Using filesort:看到这个的时候，查询就需要优化了。MySQL需要进行

      额外的步骤来发现如何对返回的行排序。它根据连接类型以及存储排序键

      值和匹配条件的全部行的指针来排序全部行

   Using temporary:看到这个的时候，查询需要优化了。这里MySQL需要创建

      临时表来存储结果，这通常发生在对不同的列集进行order by 上，而不是

      group by 上。

**count()和max()的优化方法**

**优化max（）**

创建索引

**优化count()**

作用：1.统计某一列非空(not null)值得数量,即统计某列有值得结果数,使用count(col)。

2.统计结果集的行数，此时不用管某列是否为null值。即使用count(\*).

**count优化总结：**

1.如果表没有主键，那么count（1）比count（\*）快。

2.如果有主键，那么count（主键，联合主键）比count（\*）快。

3.如果表只有一个字段，count（\*）最快。

4.当统计某一列等于多少的值得时候可以使用下面两种方法。

#统计出表中id为23的值的数量的两种方法

SELECT SUM(IF(id = 23,1,0)) FROM table

SELECT COUNT(id = 23 OR NULL) FROM table

**对于MyISAM表来说**：

5.任何不是统计某一列的情况下SELECT COUNT(\*) FROM tablename是最优选择；

6.尽量减少SELECT COUNT(\*) FROM table WHERE COL = ‘value’ 这种查询，

7.杜绝SELECT COUNT(COL) FROM table WHERE COL2 = ‘value’ 的出现。因为MySQL会自动将count(col)转为count(\*),这样也同样耗费了些时间，如果col没有指定为NOT NULL 的话，那么效率就更低了，MySQL就必须要判断每一行的值是否为空

**如何选择合适的列建立索引**

1.在where从句，group by 从句，order by从句,on从句中出现的列

2.索引字段越小越好 因为数据存储以页为单位，一页数据越多，IO数据量大，速度越慢，所以越小越好

3.离散度大的列放到联合索引的前面

**select \* from payment where staff\_id=2 and customer\_id=584;**

是index(staff\_id,customer\_id)好，还是index(customer\_id,staff\_id)好？

**由于customer\_id的离散度更大**，所以应该是用index(customer\_id,staff\_id)

这里有个离散度的概念，怎么理解这个术语呢？

如果一个字段的唯一值出现越多，离散度越大，用sql来查询比较就是

select count(distinct customer\_id),count(distinct staff\_id) from payment;

如果一个索引包括了一个查询中的所有列，则称之为**覆盖索引**

**索引的维护及优化－－－重复及冗余索引**

**重复索引是指相同的列以相同的顺序建立同类型的索引，**

如下表中的primarykey和id上的索引就是重复索引。

create table test(

    id int not null primary key,

    name varchar(10) not null,

    title varchar(50) not null,

    unique(id),

) engine=innodb;

**冗余索引是指多个索引的前缀列相同，或是在联合索引中包含了主键的索引，**

下面这个列子中的key(name,id)就是一个冗余索引

create table test(

   id int not null primary key,

   name varchar(10) not null,

   title varchar(50) not null,

   key(name,id)

) engine=innodb;

注意：innodb类型的表中每一个索引后面都会跟个主键，所以上面是冗余的。

**怎么查找重复冗余索引**

use schema\_information;

select a.table\_schema as '数据名',

       a.table\_name as '表名',

       a.index\_name as '索引1',

       b.index\_name as ' 索引2',

       a.column\_name as  '重复列名'

from statistics a join statistics b on a.table\_schema=b.table\_schema

and a.table\_name=b.table\_name and a.seq\_in\_index=b.seq\_in\_index and

a.column\_name=b.column\_name where a.seq\_in\_index=1 and a.index\_name<>b.index\_name;

**显示表结构语句：**

show create table employees.debt\_emp;

**可以使用pt-duplicate-key-checker工具检查重复及冗余索引,还能给出索引维护建议**

参数为数据库用户名  密码  IP

pt-duplicate-key-checker \

-u root \

-p '6b\_w1!mfc&0Q' \

-h 127.0.0.1

**索引的维护及优化－－－删除不用的索引**

目前MySQL中还没有记录索引的使用情况，但是在PerconMySQL和MariaDB中

可以通过INDEX\_STATISTICS表来查看哪些索引未使用，但在MySQL中目前智能通过慢

查日志配合pt-index-usage工具来进行索引使用情况的分析。

pt-index-usage \

  -uroot -p '' \

  mysql-slow.log

**数据库结构优化－选择合适的数据类型**

**数据类型的选择，重点在于合适二字**

1.使用可以存下你的数据的最小的数据类型

2.使用简单的数据类型，int要比varchar类型在mysql处理上简单。

3.尽可能的使用not null定义字段

4.尽量少用text等大数据类型，非用不可时最好考虑分表

**使用int存储日起时间，利用from\_unixtime()，unix\_timestamp()两个函数来进行转换**

from\_unixtime() 将int转换为时间格式  unix\_timestamp()反之。

**使用bigint来存储IP地址，利用inet\_aton(),inet\_ntoa()两个函数转换。**

inet\_aton()将IP地址转为bigint类型，inet\_ntoa()反之

**以下两条可不重点关注，主要看对业务的分析需要进行把重复率较高的字段分出来建立多张表还是把经常使用的字段为了优化查询在表中多加字段以空间换取时间的操作**

**× 数据库结构优化－表的范式化和反范式化**

**范式化是指数据库设计的规范，目前说道范式化一般是指第三设计范式，也就是要求数据库表中不存在非关键字段对任意候选关键字段的传递函数依赖则符合第三范式**

e.g.:

商品名称    价格   重量   有效期   分类

－－－－－－－－－－－－－－－－－－－－－－－－－－－－

可乐        3     250ml  201406  饮料    酸性饮料

存在以下传递函数依赖关系

商品名称->分类->分类描述

也就是说存在非关键字段 分类描述对关键字段商品名称的传递函数的依赖

不符合第三范式存在下面问题：

1.数据冗余

2.数据插入异常

3.数据更新异常

4.数据删除异常

**× 数据库结构优化－表的反范式化优化**

**反范式化是指为了查询效率的考虑把原本符合第三范式的表适当的增加冗余，以达到优化查询效率的目的，反范式化是一种以空间来换取时间的操作。**

**数据库结构优化－表的垂直划分**

**表的垂直拆分，就是把原来一个有很多列的表拆分成多张表，这解决了表的宽度问题。**

通常垂直拆分可以按以下原则来进行：

1.把不常用的字段单独放到一个表

2.把大字段独立存放到一个表

3.把经常在一起使用的字段放到一起

**数据库结构优化－表的水平拆分（分表的概念）**

**表的水平拆分是为了解决单表数据量过大的问题,水平拆分的表每一个表的结构都一样。**

常用的水平拆分方法为：

1.对主键进行hash计算，如果要拆分5个表则使用mod(id,5)取出0-4个值

2.针对不同的hashID把数据存到不同的表中。

**操作系统配置优化**

**网络方面的配置，要修改/etc/sysctl.conf文件**

#增加tcp支持的队列数

net.ipv4.tcp\_max\_syn\_backlog=65535

#减少断开连接时，资源的回收

net.ipv4.tcp\_max\_tw\_buckets=8000

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle=1

net.ipv4.tcp\_fin\_timeout=10

**打开文件数的限制，可以使用ulimit -a 查看目录的各位闲置，可以修改**

**/etc/security/limits.conf文件，增加以下内容以修改打开文件数量的限制**

\*soft nofile 65535

\*hard nofile 65535

注意：MySQL服务器上最好关闭iptables,selinux等防火墙软件,因为多少会带来网络原因的系统损耗，特别是访问量大的网站，这种损耗还是比较大，可以使用硬件防火墙代替，还要注意文件类型

**MySQL配置优化**

**MySQL可以通过启动时指定配置参数和使用配置文件两种方法进行配置，大多数情况下配置文件位于/etc/my.cnf或/etc/mysql/my.cnf；在windows系统配置文件可以位于**

**C：/windows/my.ini文件**

MySQL查找配置的顺序可以通过以下方法获得：

**$:/usr/sbin/mysqld --verbose --help | grep -A 1 'Default options'**

注意：如果存在多个位置存在配置文件，则后面的会覆盖前面的

**常用参数说明**

**innodb\_buffer\_pool\_size**

非常重要的参数，用于配置innodb的缓冲池如果数据库中只有innodb表，则推荐配置

量为总内存的75%

下面是检测语句

select engine,round(sum(data\_ength + index\_length)/1024/1024,1) as 'Total MB',

from information\_schema.tables where table\_schema not in ('information\_schema',

'performance\_schema') group by engine;

innodb\_buffer\_pool\_size >= Total MB

-----

innodb\_buffer\_pool\_instances

**innodb\_log\_buffer\_size**

MySQL5.5中新增参数，可以控制缓冲池的个数，默认情况下只有一个

innodb log缓冲的大小，由于日志最长每秒钟就会刷新所以一般不用太大

innodb\_flush\_log\_at\_trx\_commit

关键参数，对innodb的IO效率影响比较大，默认值为1，可以取0，1，2三个值，一般

建议设为2，但如果数据安全性要求比较高则使用默认值1.

innodb\_read\_io\_threads

innodb\_write\_io\_threads

以上两个参数决定了innodb读写IO进程数，默认4

**innodb\_file\_per\_table**

关键参数，控制innodb中每一个表使用独立的表空间，默认为OFF，也就是所有表都

会建立在共享表空间。

innodb\_stats\_on\_metadata

决定了MySQL在什么情况下会刷新innodb表的统计信息

**mysql第三方配置工具：tools.percona.com/wizard**

**硬件优化**

选择CPU

mysql有些工作只使用单核CPU,建议单核，CPU:单核频率高，mysql5.5不要超过32核

DISK IO优化

raid0 条带，多个磁盘连接成一个硬盘使用，IO最好
raid1 镜像，至少两个磁盘，魅族磁盘存储数据相同
raid5 把>=3个磁盘当成一个逻辑盘使用，读写时建立奇偶校验信息并存储到不同磁盘，当有一个盘发生损坏时，利用剩下的数据和奇偶校验信息去恢复被损坏的数据
raid1+0 1和0的结合，同时具备raid0和raid1的优缺点，一般建议使用这个级别

磁盘阵列：（SNA和NAT）

1. 常用于高可用

2. 某些存储阵列顺序读写效率高，但是随机读写不如人意，但随机读写几率较大