# mysql

出自：兄弟连IT教育

转载请注明出处注意:\*\* 代表重点

# 第一章mysql表设计\*\*

## E-R图\*\*

* + 1. E-R图组成
       1. 长方形 实体 -》数据表
       2. 椭圆形 属性 -》字段
       3. 菱形 关系 -》实体和另外一个实体存在的关系(1)一对一 hasOne 主外键通过某一个id 关联

用户表-》用户详情表商品表-》商品详情表

* + - 1. 一对多 hasMany主外键通过某一个id 关联

用户表-》多个地址

1把锁-》多把钥匙

一个英雄-》多个皮肤一个栏目-》多个列表一个用户-》多个帖子

* + - * 1. 多对多ManyToMany 一般情况下新建一个关系表

帖子收藏

一个用户可以收藏多个帖子一个帖子可以被多个用户收藏

排片,选课

* + - 1. 属于belongTo主外键通过某一个id 关联

多个皮肤属于英雄 多个列表属于当前栏

## 表的结构要满足三范式标准\*\* 纯理论 了解

* + 1. 原子性: 数据表字段达到不可再分状态,例如商品详情图,城市级联 省市县
    2. 唯一性 : 数据表中只有唯一的一个记录，在一个表里,不要出现相同的记录，添加主键
    3. 无冗余性 :如果一个字段可以从其他字段计算出来,这个字段不需要设计,例如商品总价...

不能完全应用有所有实战 能遵守的时候 就遵守

## 选择合适的引擎(myisam和innodb的区别)\*\*

* + 1. 数据查询的速度myisam快innodb慢
    2. 事务

myisam不支持innodb支持

* + 1. 全文索引

myisam支持innodb不支持

* + 1. 文件存储的区别

1. myisam有三个文件

.frm 存储表结构 字段

.myd 存储数据

.myi 存储索引

1. innodb有两个文件

.frm 存储表结构

.ibd 表数据和索引

* + 1. 总结

频繁查询的表可以设置成myisam, 对数据安全性比较高的设置成innodb

## 表字段的类型要设置合理\*\*

* + 1. 字段选择顺序 数字 -> 时间和枚举 -> 字符串 ,数字时间枚举要比字符串检索效率快
    2. 尽量不要使用null类型

null也会被当做检索数据源，增加检索时间

# 第二章mysql备份\*\*

## 普通备份\*\*

* + 1. 使用mysqldump备份单个数据库中的所有表

mysqldump -u root -h localhost -p 数据库 >c:/xxx.sql

* + 1. 使用mysqldump备份数据库中的某个表

mysqldump -u root -h localhost -p ku stu >c:/xxx.sql

* + 1. 使用mysqldump备份多个数据库

mysqldump -u root -h localhost -p --databases 数据库1 数据库2 >c:/xxx.sql

注意:使用--databases参数之后，必须指定至少一个数据库的名称，多个数据库名称之间用空格隔开

* + 1. 使用--all-databases参数备份系统中所有的数据库

mysqldump -u root -h localhost -p --all-databases >c:/xxx.sql

# 第三章mysql数据库优化\*\*

## 定位慢语句\*\*

* + 1. 作用定位慢语句
    2. 修改mysql 配置文件

my.ini

* + 1. 在[mysqld]里加入如下代码

slow\_query\_log = ON 是否开启慢查询日志

slow\_query\_log\_file = /slow.log 存储路径 可以不设置long\_query\_time = 1 慢查寻阀值单位秒，超过阀值记录日志

* + 1. 重启mysql

## 增删改优化

* + 1. 查看电脑的cpu使用和内存使用
       1. window打开任务管理器 ctrl+alt+delete
       2. linux top命令
       3. 如果cpu使用率高或者内存占有率大的话,及时释放内存
    2. 查看表的内容是不是太大了

select count(\*) from test

如果太大,可以考虑使用缓存

* + 1. 检查表的索引是不是很多

desc 表名

如果表索引很多,及时减少索引,因为索引的创建也会消耗资源开销

* + 1. mysql 主从服务器 读写分离

## 查询 优化\*\*

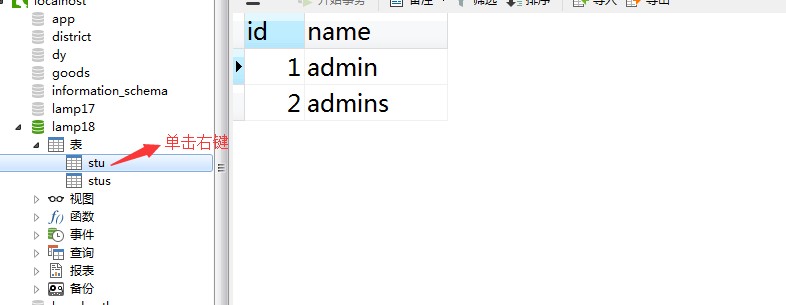
### 缓存优化\*\*

原则:数据读的多写的少,类似于微博等，读的时候，先读缓存，缓存没有的话，那么就读数据库，然后取出数据后 放入缓存，同时返回响应，可以使用redis,或者memcache 前两天学的那玩意

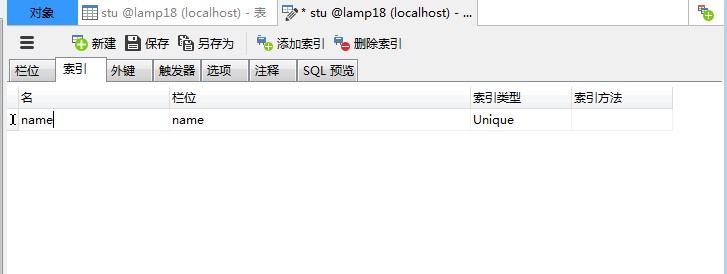
### 索引优化\*\*

* + - 1. 索引的定义

索引就类似于字典目录,使用索引可快速访问数据库表中的特定信息

* + - 1. 索引类型
         1. 主键索引 它 是一种特殊的[唯一索引](https://www.baidu.com/s?wd=%E5%94%AF%E4%B8%80%E7%B4%A2%E5%BC%95&amp;tn=SE_PcZhidaonwhc_ngpagmjz&amp;rsv_dl=gh_pc_zhidao)，不允许有空值
         2. 唯一索引 与"普通索引"类似，不同的就是：索引列的值必须唯一，但允许有空值
         3. 普通索引 最基本的索引，没有任何限制
      2. 索引创建
         1. 通过数据库管理工具创建,例如:Navicat

单击设计表



* + - * 1. 命令创建

<1 主键索引:

建表时:create table test (id int primary key auto\_increment, name char(30));

建表后 :alter table test add primary key (id)

<2 唯一索引

建表时:create table test (id int, name char(30), unique name(name))

建表后:alter table test add unique name (name)

<3 普通索引

建表时:create table test (id int, name char(30), index name(name))

建表后:alter table test add index name (name)

* + - 1. 索引删除
         1. Navicat 数据库管理工具删除
         2. 命令删除

<1 主键索引

alter table test modify id int(10) 或者 alter table test drop primary key

<2 非主键索引

alter table test drop index name

* + - 1. 索引查看

show index from 表名或者 desc 表名

* + - 1. 注意点

索引并非是绝对的好, 索引的创建也是有资源开销的,比如磁盘空间的占用, 并且索引还会影响增删改语句的执行效率. 索引的建立一定要合理 哪些字段 经常被查询

* + - 1. explain语句分析 分析sql的性能
      2. (1)示例

explain select \* from user\G

(2)根据返回参数来判定是否需要加索引

<1 row 查询次数

<2 type(连接类型)

《1 好坏顺序

system > const > eq\_ref > ref > fulltext > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL

《2 const

表中满足条件的记录最多一条,通常会出现在主键和unique索引中 ,例如: explain select \* from test where id = 1\G;

《3 eq\_ref

某一列等于带索引的列 ,例如:

explain select a.*,b.* from goods as a,cate as b where a.cate\_id = b.id and a.id > 5000\G;

《4 ref

使用普通索引进行查询

select \* from test where nickname = 'xiaohigh';

《5 ref\_or\_null

通过普通索引检索,并且会检索null值

explain select \* from test\_2 where nickname = 'xiaohigh' or nickname = null\G

《6 range

explain select \* from test where id > 3;

《7 index

跟all一样,不过只会扫描索引. explain select id from test

《8 all

全表扫描

explain select \* from test

《9 注意

查询时一定不能出现all类型 (全表扫描,速度是最慢的) ,如果在一个数据很多的数据表里分析sql

语句的时候,返回类型出现index或者all,那这个时候必须加索引,索引个数尽量小于表字段的三分之一

### mysql主从服务器搭建,读写分离优化\*\*

* + - 1. 目的

主服务器写数据,从服务器读取数据

* + - 1. 主从服务器搭建
         1. 在两台服务器环境下安装mysql数据库 iptables -F(linux关闭防火墙)

(2)主服务器

<1 修改主服务器配置文件

《 1 vi /etc/my.cnf windows上面的配置文件 my.ini

《2 [mysqld]

log-bin=mysql-bin (51行 第一个) //[必须]启用二进制日志

《3 server-id=1(59行) (一般来说默认开启) 1 主服务器标识

《4 重启mysql服务

mysql 关闭

mysqladmin -u root -p shutdown mysql启动

mysqld\_safe -u mysql &

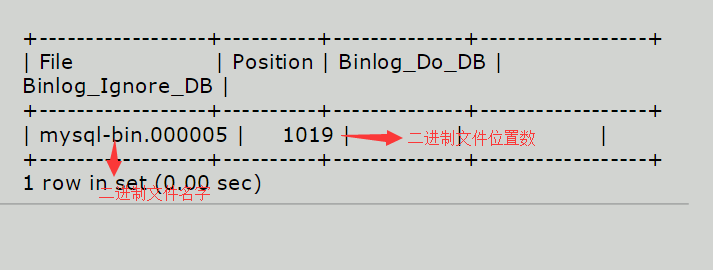
Windows 直接重启wamp

<2 主服务器创建mysql用户

1 mysql> GRANT REPLICATION SLAVE ON \*.\* to 'slaver'@'%' identified by '123456'

<3 查看当前主服务器信息

show master status

(3)从服务器 换电脑啦

<1修改主服务器的mysql配置文件vi /etc/my.cnf

<2 修改配置文件信息

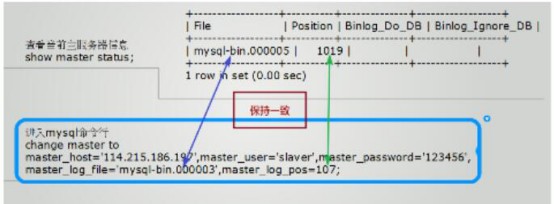
[mysqld] 开启server-id=2(96行)(默认没有开启) 重启mysql服务

<3 进入从服务器 mysql命令行 ,执行如下命令

change master to master\_host='192.168.14.254',master\_user='slaver',master\_password='123456',master\_log\_file='mysql-bin.000010',master\_log\_pos=11246376;

注意:master\_host 主服务器ip master\_user 主服务器创建的用户名 master\_password 密码

master\_log\_ﬁle 值必须和主服务器相同 master\_log\_pos 必须和主服务器相同

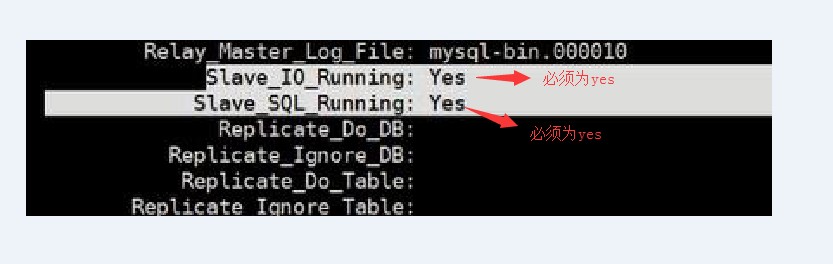


<4 开启

start slave; (从服务器上写)

Ok 代表成功

<5 查看状态

show slave status\G(从服务器上写)

返回的两个参数全部为yes才代表,主从服务器搭建成功

* + - 1. 代码演示

在localhost本地连接 主从服务器,做读写分离

<1 zhu.php 主服务器代码 做数据写入

1. <?php
2. //主服务器 数据的插入
3. $pdo=new PDO("mysql:host=主服务器ip;dbname=aaaaa","root","123456");
4. //设置字符集
5. $pdo->exec("set names utf8");
6. $pdo->exec("insert into stu(name)value('admins')"); 7 ?>

<2 cong.php 从服务器代码 做数据的读取

1. <?php
2. //从服务器做数据的读取
3. $pdo=new PDO("mysql:host=从服务器ip;dbname=aaaaaa","root","123456");
4. $pdo->exec("set names utf8");
5. //获取数据
6. $list=$pdo->query("select \* from stu");
7. //获取结果集
8. $data=$list->fetchAll(PDO::FETCH\_ASSOC);
9. echo "<pre>";
10. var\_dump($data); 11 ?>

<3 在浏览器里执行zhu.php

直接报错-》mysql Access denied for user 'root'@'localhost'

原因:localhost 远程连接linux下mysql root没有权限解决方法:登录主服务器mysql 执行如下命令:

1. GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '你当前数据库的密码' WITH GRANT OPTION;

修改配置文件 my.ini

Bind-address=0.0.0.0

创建一个远程连接的用户

grant all on \*.\* to xiaoyu@'%' identified by '123456' with grant option

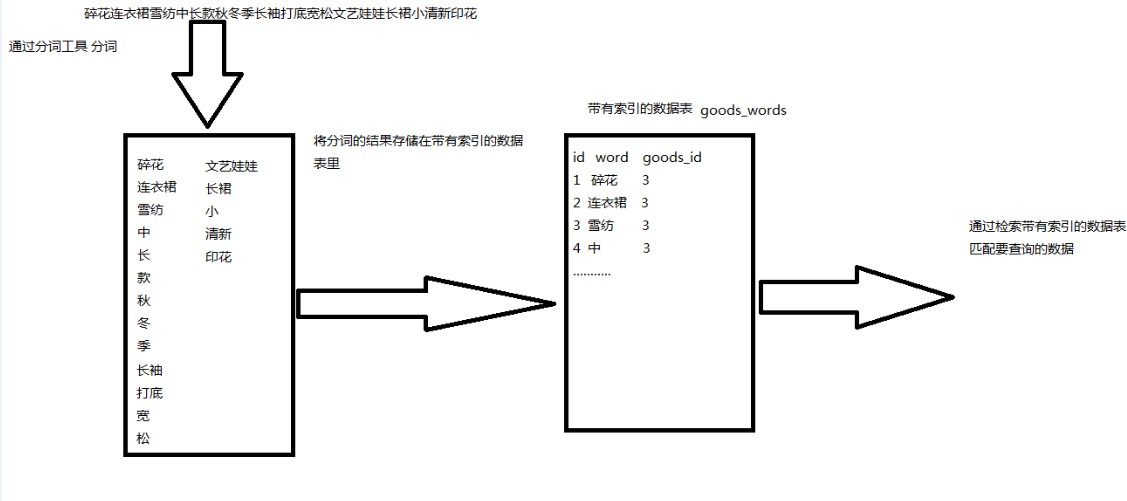
执行完命令,重新执行zhu.php ,这下不报错了,数据做了写入

<4 在浏览器里执行cong.php 读取主服务器写入的数据

### 中文分词优化\*\*

* + - 1. 使用中文分词背景 sql语句 用到了like 里面的索引不生效了 like的查询效率非常的低

普通的模糊搜索，检索数据的时候,用不到索引，而是全文检索,全文检索速率很慢,尤其是在千万级的数据里, 这个时候我们考虑中文分词优化查询,其核心使其用到索引,提高数据的检索效率

* + - 1. 中文分词优化原理
      2. 分词工具使用 scws4

//声明字符串 分词的数据源

$text = <<<EOF

怎么会爱上了他，还不是因为眼瞎，放弃了一片森林，却被一颗破草拌倒了。

EOF;

//导入导入中文分词工具包分词类

require './pscws4/pscws4.class.php';

// 实例化

$cws = new PSCWS4();

//设置字符集

$cws->set\_charset('utf8');

//设置中华词典（分词的规则）

$cws->set\_dict('./pscws4/etc/dict.utf8.xdb');

//设置utf8规则

$cws->set\_rule('./pscws4/etc/rules.utf8.ini');

//忽略标点符号

$cws->set\_ignore(true);

//传递字符串

$cws->send\_text($text);

//获取权重最高的前十个词

$ret = $cws->get\_tops(5);

//获取全部的分词结果

// $data=$cws->get\_result();

//打印

echo "<pre>"; var\_dump($ret);

//关闭

$cws->close();

?>

<?php

//test.php

//

// Usage on command-line: php test.php <file|textstring>

// Usage on web:

header('content-type:text/html;charset=utf-8');

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

* + - 1. 中文分词优化代码

1. <?php
2. //导入导入中文分词工具包分词类
3. require './pscws4/pscws4.class.php';
4. class Chinese{
5. //成员属性
6. //成员方法 $str 分词的数据源
7. public static function fenci($str){
8. // 实例化
9. $cws = new PSCWS4();
10. //设置字符集
11. $cws->set\_charset('utf8');
12. //设置中华词典（分词的规则）
13. $cws->set\_dict('./pscws4/etc/dict.utf8.xdb');
14. //设置utf8规则

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31 }

32 }

$cws->set\_rule('./pscws4/etc/rules.utf8.ini');

//忽略标点符号

$cws->set\_ignore(true);

//传递字符串

$cws->send\_text($str);

//获取权重最高的前十个词

// $ret = $cws->get\_tops(5);

//获取全部的分词结果

$data=$cws->get\_result();

//打印

// echo "<pre>";

// var\_dump($ret);

//关闭

$cws->close();

//返回结果return $data;

1. //连接数据库
2. $pdo=new PDO("mysql:host=localhost;dbname=lamp","root","123456");
3. //设置字符集
4. $pdo->exec("set names utf8");
5. //准备sql
6. $sql="select title,id from goods";
7. //执行sql
8. $list=$pdo->query($sql);
9. //获取结果集
10. $data1=$list->fetchAll(PDO::FETCH\_ASSOC);
11. // echo "<pre>";
12. // var\_dump($data1);
13. //遍历
14. foreach($data1 as $key=>$value){
15. //分词数据源 $value['title']
16. $data2=Chinese::fenci($value['title']);
17. // echo "<pre>";
18. // var\_dump($data2);666
19. //遍历
20. foreach($data2 as $k=>$v){
21. //把分词的结果 id存储在goods\_words表
22. $pdo->exec("insert into goods\_words (word,goods\_id)values('".$v['word']."','".$value['id']."')");

55 }

56 }

57

58

59

60 ?>

* + - 1. 中文分词优化mysql搜索实例

1. fenci.php

1 <?php

* 1. //判断是否具有搜索关键词
  2. if(empty($\_GET['title'])){
  3. //接数据库
  4. $pdo=new PDO("mysql:host=localhost;dbname=lamp","root","123456");
  5. $pdo->exec("set names utf8");
  6. //；获取goods表数据
  7. $sql="select id,title from goods";
  8. //执行sql
  9. $list=$pdo->query($sql);
  10. //获取结果集
  11. $data1=$list->fetchAll(PDO::FETCH\_ASSOC);
  12. }else{
  13. //包含form.php
  14. require("./form.php"); 16 }

17

18 ?>

1. <!doctype html>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. <title>前台首页</title>
6. </head>
7. <body>
8. <center>
9. <form action="fenci.php" method="get">
10. 搜索:<input type="text" name="title">
11. <input type="submit" value="搜索">
12. </form>

31

1. <table border="1px" width="400px">
2. <tr>
3. <th>ID</th>
4. <th>TITLE</th>

36

37 </tr>

1. <?php foreach($data1 as $row){ ?>
2. <tr>
3. <td><?php echo $row['id'] ?></td>
4. <td><?php echo $row['title'] ?></td> 42 </tr>

43 <?php } ?>

1. </table>
2. </center>
3. </body>
4. </html>
5. form.php
6. <?php
7. //获取搜索的关键词
8. $title=isset($\_GET['title'])?$\_GET['title']:'';
9. $pdo=new PDO("mysql:host=localhost;dbname=lamp","root","123456");
10. $pdo->exec("set names utf8");
11. //从goods\_words 表查询数据
12. $sql1="select word,goods\_id from goods\_words where word='{$title}'"; 8 //执行
13. $list1=$pdo->query($sql1);
14. //获取结果集
15. $data2=$list1->fetchAll(PDO::FETCH\_ASSOC);
16. // echo "<pre>";
17. // var\_dump($data2);
18. //存储goods\_id
19. $arr=array();
20. //遍历数据
21. foreach($data2 as $key=>$value){
22. $arr[$key]=$value['goods\_id']; 19 }
23. //把数组转换为字符串
24. $id=implode(",",$arr);
25. // echo "<pre>";
26. // var\_dump($arr);
27. //通过id匹配数据
28. //准备sql
29. $sql2="select id,title from goods where id in({$id})";
30. //执行sql
31. $list3=$pdo->query($sql2);
32. //获取结果集
33. $data1=$list3->fetchAll(PDO::FETCH\_ASSOC);
34. // echo "<pre>";
35. // var\_dump($data1);
36. return $data1; 34 ?>

### 其他优化小技巧\*\*

* + - 1. 尽量使用一条语句插入, 避免循环插入.
      2. 不要将图片存入到数据库中,用路径代替
      3. 尽量避免使用select \* ,效率较低 用谁查谁
      4. 分组时添加group by id order by null 及时释放资源
      5. 使用or语句要注意, 两侧的语句都有索引才会使用索引.
      6. 针对myisam的表要定期执行命令 optimize table test 合并表空间产生碎片

# 第四章mysql 权限操作\*\*

## 用户管理\*\*

* + 1. 必须在root下创建用户
    2. 创建用户

insert into mysql.user(Host,User,Password)values(“localhost”,用户名,password(密码));

创建完毕后:ﬂush privileges刷新权限

* + 1. 查看用户列表

select User from mysql.user group by User;

## 权限分配\*\*

* + 1. 必须在root下做权限分配
    2. 分配权限

1 grant 权限 on \*.\* to xx@localhost identified by '用户密码';

* + 1. 授权完毕之后要刷新权限 ﬂush privileges;
    2. 权限列表

create 建库建表权限drop 删库删表权限insert 数据插入权限delete 数据删除权限alter 数据修改权限select 数据读取权限index 索引操作权限

* + 1. \*. \* 代表数据库 数据表
    2. xx 用户名
    3. 查看权限
       1. 登录当前用户查看当前用户（自己）权限

show grants;

* + - 1. 登录root查看其他 MySQL 用户权限

show grants for 用户@localhost;

## 权限夺回\*\*

* + 1. 必须在root下做权限夺回
    2. 权限夺回命令

revoke 权限 on 数据库.数据表 from 用户名@localhost;

* + 1. 授权完毕之后要刷新权限 ﬂush privileges;

## 注意\*\*

* + 1. Test information\_schema数据库和test前缀的数据库不受权限控制
    2. 授权完毕之后要刷新权限 ﬂush privileges;

# 第五章mysql日志操作

## 日志介绍

Mysql 日志操作可以快速的记录mysql的操作信息,

## 日志操作

修改mysql 配置文件

Linux 配置文件 vim /etc/my.cnf

Windows 配置文件 my.ini

## 错误日志操作

* + 1. 在启动或者关闭数据库信息的时候,出现错误,会记录一些日志信息
    2. 设置
       1. 修改mysql 配置文件

vim /etc/my.cnf

* + - 1. 在[mysqld]里加入如下代码

log-error="/tmp/mysql-error.log"

* + - 1. 重启mysql
         1. mysql 关闭

mysqladmin -u root -p shutdown

* + - * 1. mysql 启动

mysqld\_safe -u mysql &

* + - 1. 查看/tmp/mysql-error.log 文件

vim /tmp/mysql-error.log

## 查询日志 操作

* + 1. 在执行增删改查的时候,日志信息记录
    2. 修改mysql 配置文件

vim /etc/my.cnf

* + 1. 在[mysqld]里加入如下代码

log="/tmp/mysql-query.log"

* + 1. 重启mysql
       1. mysql 关闭

mysqladmin -u root -p shutdown

* + - 1. mysql 启动

mysqld\_safe -u mysql &

* + 1. 查看/tmp/mysql-query.log 文件

vim /tmp/mysql-query.log

## 慢查询日志 操作

* + 1. 作用定位慢语句
    2. 修改mysql 配置文件

vim /etc/my.cnf

* + 1. 在[mysqld]里加入如下代码

log-slow-queries="/tmp/mysql-slow.log" long\_query\_time=1 （单位为秒,超过1秒代表慢语句）

* + 1. 重启mysql
       1. mysql 关闭

mysqladmin -u root -p shutdown

* + - 1. mysql 启动

mysqld\_safe -u mysql &

* + 1. 查看/tmp/mysql-slow.log vim /tmp/mysql-slow.log

## 二进制日志 操作

二进制日志 默认开启