1. 课程介绍

* 1.介绍什么是mysql优化;（了解）
* 2.mysql优化方法；（掌握）
* 3.Mysql索引的使用;（掌握）
* 4.分表技术;（掌握）

1. mysql优化概述

概述: 前面我们学习了页面静态化和redis,它们是通过不操作mysql数据库达到提速目的。但是某些功能是一定要操作数据库的，这就要求我们必须对mysql本身进行优化。

**mysql数据库优化的常见方法:**

* + - 1. 表的设计要合理(满足3NF) 3范式
      2. 创建适当索引[主键索引|唯一索引|普通索引|全文索引|空间索引]
      3. 对SQL语句优化---->定位慢查询(explain)
      4. 使用分表技术(重点【水平分表，垂直分表】), 分区技术(了解)
      5. 读写分离(配置)
      6. 创建适当存储过程，函数，触发器
      7. 对my.ini优化，优化配置
      8. 软件硬件升级

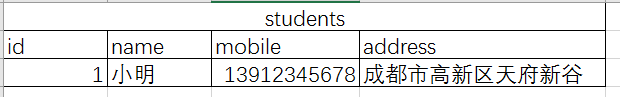
1. 表的设计满足3NF

概述: 目前我们的表的设计，最高级别的范式是"6NF",对PHP程序员而言，我们的表满足3NF即可。

* 1. 1NF

所谓1NF,就是

1. 指表的属性(列)具有原子性， 即表的列的不能再分了。



1. 不能有重复的列

特殊

1. 只要是关系型数据库，就天然的满足1NF
2. 常见数据库

关系型数据库(mysql, oracle, sql server,informix, db2 , postgres)

非关系型数据(Nosql类型的数据库由Redis, MongoDB)

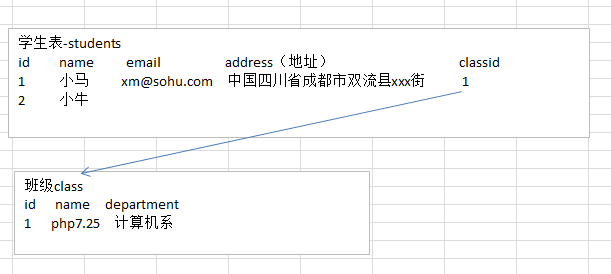
* 1. 2NF

所谓2NF,就是指我们的表中不能有完全重复的一条记录（行）.一般情况下通过设置一个主键来搞定，而且该主键是自增的。

* 1. 3NF(外键)

所谓3NF就是指，**如果列的内容可以被推导(显式推导，隐式推导)出，那么我们就不要单独的用一列存放。**

举例:下面是满足3NF



* 1. 反3NF

在通常情况下，我们的表的设计要严格的遵守3NF,但也有例外。有时为了提高查询的效率，我们需要违反3NF。

1. 构建海量表,定位慢查询

为了讲解这个优化，我们需要构建一个海量表(8000000),而且每条数据不一样。

* 1. 构建海量表步骤

1. 创建一个测试数据库

create database bigdb;

1. 创建表

CREATE TABLE dept( /\*部门表\*/

deptno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0,

dname VARCHAR(20) NOT NULL DEFAULT "",

loc VARCHAR(13) NOT NULL DEFAULT ""

) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

加入数据: dept.sql

#创建表EMP雇员

CREATE TABLE emp

(empno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0, /\*编号\*/

ename VARCHAR(20) NOT NULL DEFAULT "", /\*名字\*/

job VARCHAR(9) NOT NULL DEFAULT "",/\*工作\*/

mgr MEDIUMINT UNSIGNED NOT NULL DEFAULT 0,/\*上级编号\*/

hiredate DATE NOT NULL,/\*入职时间\*/

sal DECIMAL(7,2) NOT NULL,/\*薪水\*/

comm DECIMAL(7,2) NOT NULL,/\*红利\*/

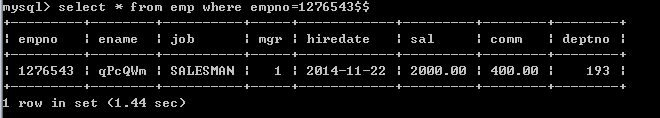
deptno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0 /\*部门编号\*/

)ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

加入数据:emp.sql

* 1. 海量表带来的问题

看一个案例



* 1. 如何定位慢查询(slow query)

介绍: 在默认情况下，mysql 是不会记录慢查询的，所以我们在测试时，可以指定mysql记录慢查询.

开启慢查询的两种方法:

* 启动时，这样启动

cmd>bin/mysqld.exe --safe-mode --slow-query-log

或者是

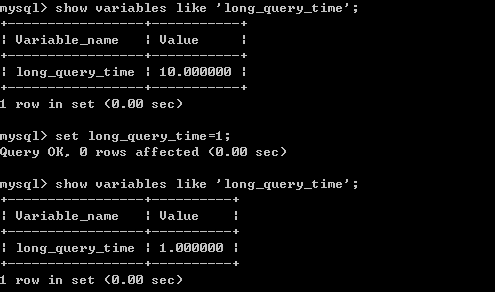
在my.ini的[mysqld]下添加一下代码并且重启

log-slow-queries = D:/server/mysql/mysqlslowquery.log(注意斜杠)

注：mysql5.6版本slow-query-log-file

long\_query\_time = 1 指定超过1秒算慢查询

* 为了测试，我们修改 long\_query\_time



* 记录下慢查询

|  |
| --- |
| # Time: 141122 10:39:45  # User@Host: root[root] @ localhost [127.0.0.1]  # Query\_time: 1.625093 Lock\_time: 0.001000 Rows\_sent: 0 Rows\_examined: 8000000  use testdb;  SET timestamp=1416623985;  select \* from emp where ename='IUYTOPUYQWE'; |

说明: Query\_time是查询的时间

Lock\_time：等待时间

* 1. 开启慢查询牺牲sql的执行效率

如何使用慢查询?

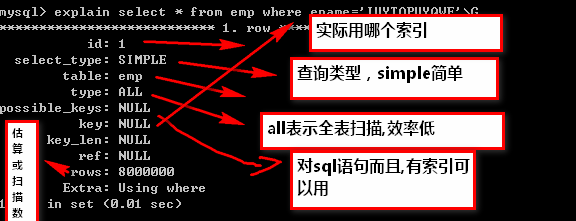
1. 系统上线之后,将慢查询开启一个星期.

2. 当你认为系统所由于数据增多导致系统执行缓慢.再开启慢查询找到执行慢的sql语句,然后在优化它.

* 1. 如何分析一个sql语句的问题-explain 工具
     + 1. 基本用法:

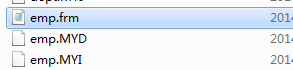
explain sql\G

* + - 1. 案例:



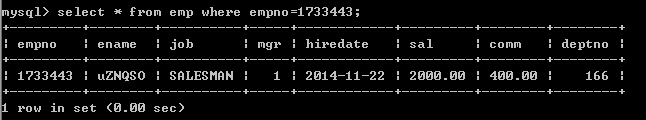
* 1. 先使用索引来优化sql查询
     + 1. 给empno段添加主键索引

alter table emp add primary key (empno);



一个表(存储引擎是MyISAM),对应三个文件 xx.frm 表结构 xx.MYD 数据文件 xx.MYI 索引文件

* + - 1. 通过测试看效果



1. 索引的详解（重点）
   1. 索引创建
      1. 主键索引的创建

主键索引的创建有两种形式, 1.在创建表的时候，直接指定某列或者某几列为主键,这时就有主键索引, 2. 添加表后，再指定主键索引

* 直接创建主键索引

注意:如果是自增, 该主键不能够删除



* 先创建表，再指定主键



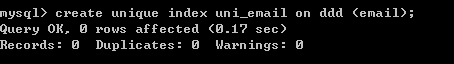
增加主键

ALTER TABLE 表名 ADD PRIMARY KEY (列1, 列名2..)

* 主键索引的特点
  + - 1. 一个表最多只能有一个主键
      2. 一个主键可以指向多列(复合主键)
      3. 主键索引的效率是最高，因此我们应该给id,一般id是自增.
      4. 主键索引列是不能重复，也不能为null
    1. 唯一索引的创建
* 直接在创建表的时候，指定某列或某几列为唯一索引



* 把表创建好后，再指定某列或者某几列为唯一索引



说明: 使用 create unique index 指令，必须指定索引名。



说明: 使用alter table 指令，可以指定索引名，也可以不指定。

* 唯一索引的特点

1. 一张表可以有多个唯一索引
2. 唯一索引不能重复，但是如果你没有指定not null ，唯一索引列可以为null,而且可以有多个.
3. 什么时候使用唯一索引，当某列数据不会重复，才能使用
4. 唯一索引效率也很高，可以考虑优先使用
   * 1. 普通索引的创建

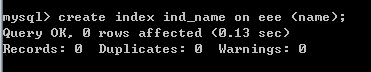
* 在创建表时指定索引,通过key或者index

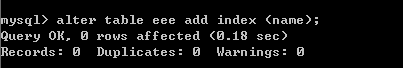


* 把表创建好后，再指定某列或者某几列为索引



* 添加普通索引(2种方式)





* 特点

1. 一张表中可以有多个普通索引，一个普通索引页可以指向多列
2. 普通索引列的数据可以重复
3. 效率相对而言低.
   1. 索引的查询

* desc 表名
* show keys from 表名\G
* show index from 表名\G
* show indexes from 表名\G
  1. 索引的修改

先删除，再添加。

* 1. 索引的删除

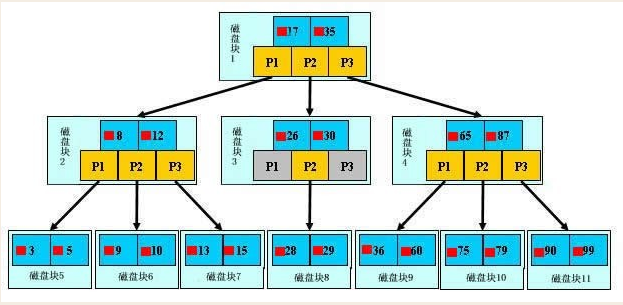
DROP INDEX 索引名 ON 表;

ALTER TABLE 表名 DROP INDEX 索引名；

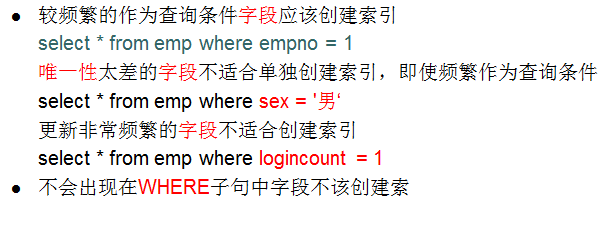
* 1. 索引的原理

添加索引后为什么会快?

B+树结构



* 1. 索引的注意事项



**优点:**

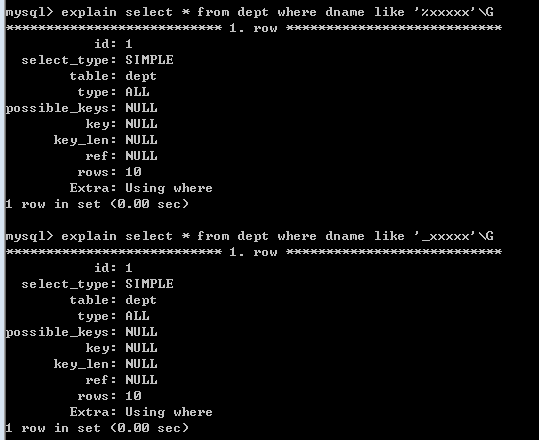
查询速度快…

**索引的缺点:**

索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。

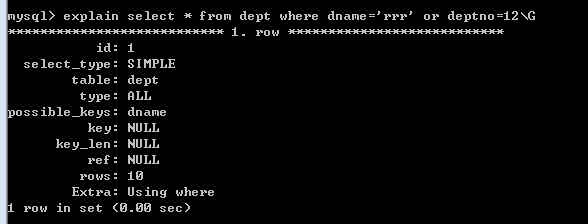
建立索引一定要根据自己的需求来…

1. sql优化方案
   1. sql语句的优化和正确使用索引
      * 1. 任何地方都不要使用 **select \* from t** ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。
        2. 对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。
        3. 应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：   
                **select id from t where num is null**   
                可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：   
                **select id from t where num=0**
        4. 应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如：   
                select id from t where num=10 or num=20   
                可以这样查询：   
                select id from t where num=10   
                union all   
                select id from t where num=20
        5. 并不是所有索引对查询都有效，SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。或者查询结果占全表的20%以上时，也不会利用索引。
        6. 对于使用like的查询，查询如果是‘%aaa’‘\_aa' 不会使用到索引‘aaa%’会使用到索引

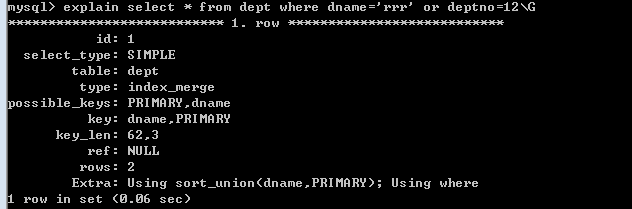


说明: 在like语句中，如果 '' 中最前有 \_ 或者 %就使用不到索引，如果在中间或者最后有 \_ 或者 %可以使用到索引。

* + - 1. 如果条件中有or，则要求or的所有字段都必须有索引，否则不会使用索引



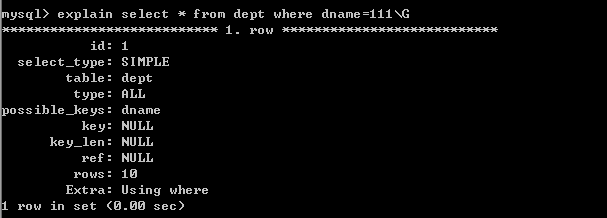
说明：因为 deptno 没有索引，所以整个sql语句就没有使用到索引。

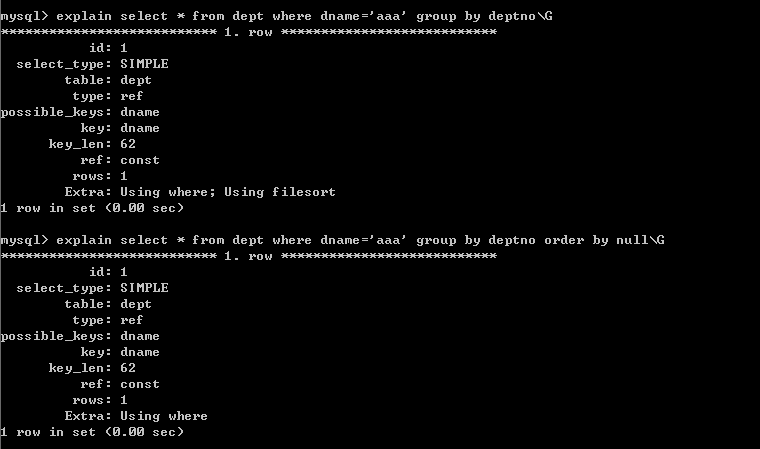


如果在 deptno上也创建索引，就可以使用到索引了.

如果mysql认为全表扫描效率更高，就不会使用索引，而会全表扫描

* + - 1. 如果列类型是字符串，那一定要在条件中将数据使用引号引用起来。否则不使用索引





* + - 1. 有些情况下，可以使用连接来替代子查询。因为使用join，MySQL不需要在内存中创建临时表

子查询：select \* from emp where deptno in (select deptno from dept)

连接：select \* from emp left join dept on emp.deptno=dept.deptno where emp.deptno=dept.deptno

* 1. 管理员在导入大量数据，可以这样提高速度

大批量插入数据(MySql管理员) ，先禁用索引，再导入数据，再启用索引

对于MyISAM：

alter table table\_name disable keys;

执行insert语句导入

alter table table\_name enable keys;

对于Innodb：

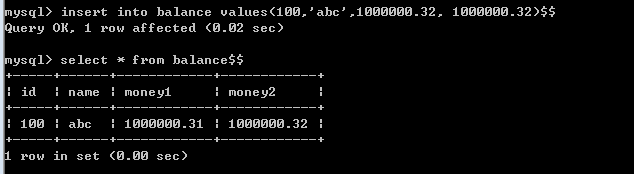
1. 将要导入的数据按照主键排序
2. set unique\_checks=0,关闭唯一性校验。
3. set autocommit=0,关闭自动提交。
   1. 如何选择存储引擎



* 如何选择的原则

1. MyISAM:默认的MySQL存储引擎。如果应用是以读操作和插入操作为主，只有很少的更新和删除操作，并且对事务的完整性要求不是很高。其优势是访问的速度快。(尤其适合论坛的帖子/信息表/新闻/商品表表)
2. InnoDB：提供了具有提交、回滚和崩溃恢复能力的事务安全。但是对比MyISAM，写的处理效率差一些并且会占用更多的磁盘空间(如果对安全要求高，则使用innodb)。[账户，积分,余额]
   1. 如何选择正确的数据类型
      1. 在满足需求的情况下尽量选择小的类型.
      2. 在精度要求高的应用中，建议使用定点数来存储数值，以保证结果的准确性。decimal 不要用float.

举例:

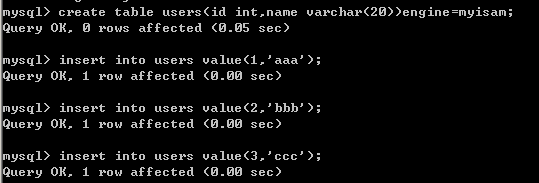


说明: 这里我们看的 float(10,2) , 和 decimal(10,2) decimal 更精准。所以我们对精度高的列，要使用decimal 类型。

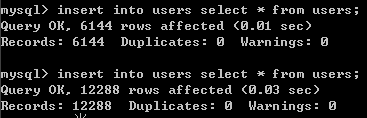
* + 1. 对存储引擎是MyISAM的表，要定时碎片整理

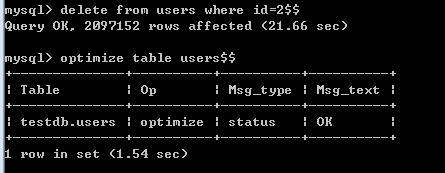
举例说明：当我们在users表中有大量数据时，我们delete 数据后，我们发现磁盘空间没有回收，因此我们需要定时的进行碎片整理.如下:

创建表:



复制大量数据到同一个表中:





optimize: 该命令可以使表中的数据彻底从数据文件中删除.

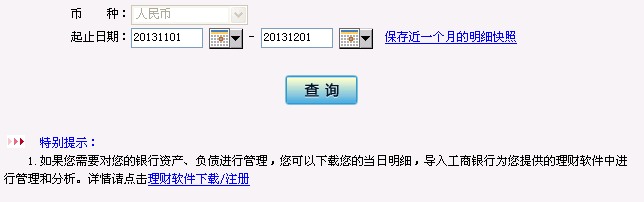
1. 分表技术和分区技术

当一个表很大很大时，我们可以考虑添加索引，当索引都解决不了这个问题时，我们可以使用分表技术或者分区技术搞定。

* 1. 水平分割表
     + 1. 按时间（按年、月分表），例如：订单表
       2. 按区间范围，例如：图书表
  2. 垂直分表

所谓垂直分割，就是把表的某个大字段(而且很少查询)，单独的取出，放入到另外一个表，并通过id关联.例如用户表和用户详情表。

* 1. 分区技术(了解)
     1. 实际案例



当某个表海量，而且我们的查询条件是按照日期来检索.

|  |
| --- |
| --创建两张表  --分区表  CREATE TABLE part\_balance  (  id int default NULL,  name varchar(30) default NULL,  savetime date default NULL  ) engine=myisam  PARTITION BY RANGE (year(savetime))  (  PARTITION p0 VALUES LESS THAN (1995),  PARTITION p1 VALUES LESS THAN (1996) ,  PARTITION p2 VALUES LESS THAN (1997) ,  PARTITION p3 VALUES LESS THAN (1998) ,  PARTITION p4 VALUES LESS THAN (1999) ,  PARTITION p5 VALUES LESS THAN (2000) ,  PARTITION p6 VALUES LESS THAN (2001) ,  PARTITION p7 VALUES LESS THAN (2002) ,  PARTITION p8 VALUES LESS THAN (2003) ,  PARTITION p9 VALUES LESS THAN (2004) ,  PARTITION p10 VALUES LESS THAN (2010),  PARTITION p11 VALUES LESS THAN MAXVALUE );  --普通表  create table no\_part\_balance  (  id int default NULL,  name varchar(30) default NULL,  savetime date default NULL  ) engine=myisam;  --将普通表中的数据复制到分区表中  insert into part\_balance select \* from no\_part\_balance;  --执行看看时间如何,结论分区表的时间要短很多.  select count(\*) from no\_part\_balance where savetime > '1995-01-01' and savetime < '1995-12-31';  select count(\*) from part\_balance where savetime > '1995-01-01' and savetime < '1995-12-31';  --使用explain 看看mysql是如何执行的  explain partitions select count(\*) from no\_part\_balance where savetime > '1995-01-01' and savetime < '1995-12-31'\G  explain partitions select count(\*) from part\_balance where savetime > '1995-01-01' and savetime < '1995-12-31'\G  --增加未索引字段查询  select count(\*) from part\_balance where savetime > '1995-01-01' and savetime < '1996-12-31' and name='hello';  select count(\*) from no\_part\_balance where savetime > '1995-01-01' and savetime < '1996-12-31' and name='hello';  --查询数据本身,得出结论也是分区表快  explain select \* from part\_balance where savetime = '2000-08-18'\G;  explain select \* from no\_part\_balance where savetime = '2000-08-18'\G; |

* 限制
  + - 1. 只能对数据表的整型列进行分区，或者数据列可以通过分区函数转化成整型列
      2. 最大分区数目不能超过1024
      3. 不支持外键
      4. 不支持全文索引（fulltext）
      5. 按日期进行分区非常适合，因为很多日期函数可以用。但是对于字符串来说合适的分区函数不太多

1. 数据库my.ini配置
   1. INNODB

数据库配置最重要的参数就是内存，如果主要用的是innodb引擎，需要把下面两个参数调大

　innodb\_additional\_mem\_pool\_size = 64M

　innodb\_buffer\_pool\_size =1G

参数说明：

innodb\_additional\_mem\_pool\_size这个参数主要缓存innodb表的索引，数据，插入数据时的缓冲。为Innodb加速优化首要参数。

该参数分配内存的原则：这个参数默认分配只有8M，可以说是非常小的一个值。如果是一个专用ＤＢ服务器，那么他可以占到内存的70%-80%。这个参数不能动态更改，所以分配需多考虑。分配过大，会使Swap占用过多，致使Mysql的查询特慢。如果你的数据比较小，那么可分配是你的数据大小＋１０％左右做为这个参数的值。

innodb\_buffer\_pool\_size用于缓存 索引 和 数据的内存大小, 这个当然是越多越好, 数据读写在内存中非常快, 减少了对磁盘的读写。 当数据提交或满足检查点条件后才一次性将内存数据刷新到磁盘中。然而内存还有操作系统或数据库其他进程使用, 一般设置 buffer pool 大小为总内存的 3/4 至 4/5。 若设置不当, 内存使用可能浪费或者使用过多。

* 1. MYISAM

对于myisam，需要调整key\_buffer\_size

key\_buffer\_size是myisam索引缓冲，对myisam很重要，缓存myisam表索引数据

若主要使用myisam，推荐设置可用RAM的20%-50%，更多的留给OS，缓存从\*.MYD读取的

如果设置太大浪费内存空间，缓存太小缓存命中率低

在my.ini中适当调整最大连接数和查询缓存

max\_connetions 1000

query\_cache\_size 100m

1. 定时完成的任务说明

项目实际需求，请完成定时备份某个数据库，或者定时备份数据库的某些表的操作

windows 下每隔1小时，备份一次数据newsdb

windows 每天晚上2:00 备份 testdb 下 dept某一张表

* 1. Window下定时备份

1. 使用mysqldump 手动备份表



恢复数据库或者表



Mysql –uroot –padmin xxx < xxx.sql

1. 可以直接写一个mytask.bat 文件，双击即可备份。在该文件中写入指令

@echo off

set "Ymd=%date:~,4%%date:~5,2%%date:~8,2%"

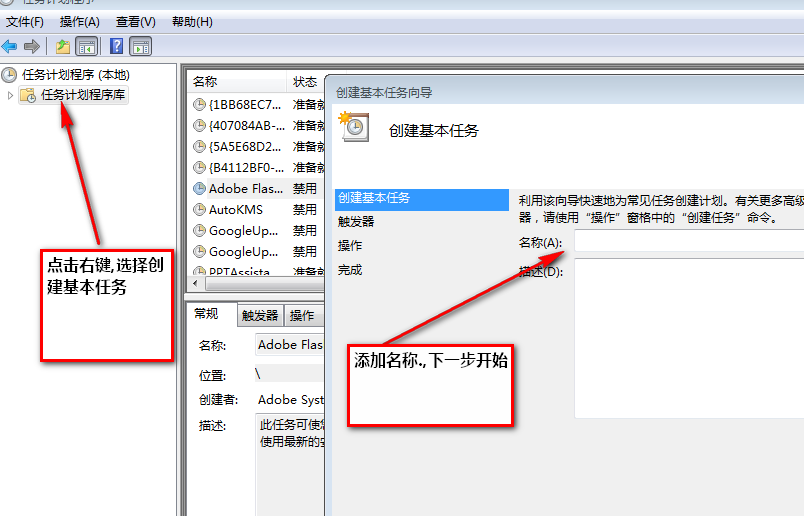
mysqldump -uroot -padmin ajaxdemo >d:/ajaxdemo\_%Ymd%.sql

@echo on

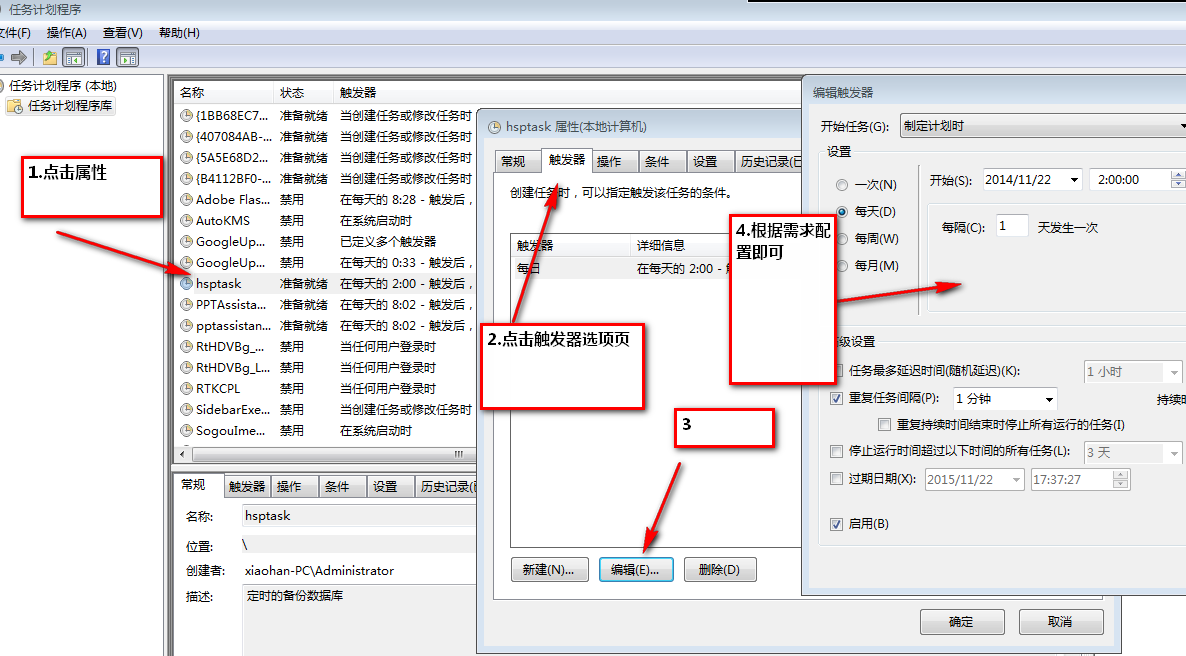
1. 下面我们可以使用系统计划任务/crontab来自动完成.

win7

开始->输入“计划任务”-》点击进入



通过配置后



* 1. Linux下的定时维护
     + 1. 直接执行mysqldump命令，每天凌晨三点备份

# crontab -e

00 3 \* \* \* mysqldump -uroot -proot dbname > /root/dbname.sql

1. 中午练习题

手写sql语句实现，如何查询一个数据表是否存在。例如：查询user表是否已经存在？

1. 作业
   * + 1. 垂直分表（user表），用户注册，用户列表，用户信息查看，修改用户信息
       2. 练习按时间分表（订单表）
       3. 实现按区间范围水平分表，根据图书ISBN号水平分表。完成图书信息增删改查（根据ISBN号搜索图书）。
2. 课程总结
   1. 重点
      * 1. 如何优化mysql
   2. 难点
      * 1. 水平分表
   3. 如何掌握？
      * 1. 练习+记忆...
3. 面试题
   * + 1. 如何优化mysql？