网站优化-mysql优化第二天

目录

[一、查询缓存 1](#_Toc531966736)

[1、具体使用 1](#_Toc531966737)

[2、无缓存 3](#_Toc531966738)

[（1）缓存失效 3](#_Toc531966739)

[（2）不使用缓存 4](#_Toc531966740)

[（3）生成多个缓存 4](#_Toc531966741)

[（4）禁用缓存 5](#_Toc531966742)

[3、查看缓存空间使用情况 5](#_Toc531966743)

[二．分区技术 6](#_Toc531966744)

[1、分区介绍 6](#_Toc531966745)

[2、分区算法 7](#_Toc531966746)

[（1）list分区 7](#_Toc531966747)

[（2）Range（范围） 9](#_Toc531966748)

[（3）Hash（哈希） 11](#_Toc531966749)

[（4）Key（键值） 11](#_Toc531966750)

[3、 分区管理 12](#_Toc531966751)

[4、特别注意； 14](#_Toc531966752)

[三、分表技术 16](#_Toc531966753)

[1. 分表设计 16](#_Toc531966754)

[2. 垂直分表（比较常用） 17](#_Toc531966755)

[四、数据碎片与维护 18](#_Toc531966756)

[五、范式讲解 19](#_Toc531966757)

[第一范式： 19](#_Toc531966758)

[第二范式 19](#_Toc531966759)

[第三范式 19](#_Toc531966760)

[反三范式（逆范式） 20](#_Toc531966761)

[六、视图 20](#_Toc531966762)

[1、视图的定义 20](#_Toc531966763)

[2、视图的作用 21](#_Toc531966764)

[3、查询视图 25](#_Toc531966765)

[4、修改视图 25](#_Toc531966766)

[5、删除视图 25](#_Toc531966767)

[6、查看视图结构 26](#_Toc531966768)

[7、查看所有视图 26](#_Toc531966769)

[8、视图与表的关系 27](#_Toc531966770)

[七、SQL 编程 28](#_Toc531966771)

[1、变量声明 28](#_Toc531966772)

[（1）会话变量 28](#_Toc531966773)

[（2）普通变量 29](#_Toc531966774)

[（3）变量赋值形式 29](#_Toc531966775)

[2、运算符 30](#_Toc531966776)

[3、语句块包含符 30](#_Toc531966777)

[4、if判断 31](#_Toc531966778)

[5、case判断 33](#_Toc531966779)

[6、循环 34](#_Toc531966780)

[八、存储过程 36](#_Toc531966781)

[1、概念 36](#_Toc531966782)

[2、存储过程的优点 37](#_Toc531966783)

[3、创建存储过程 37](#_Toc531966784)

[4、调用存储过程 39](#_Toc531966785)

# 一、查询缓存

## 1、具体使用

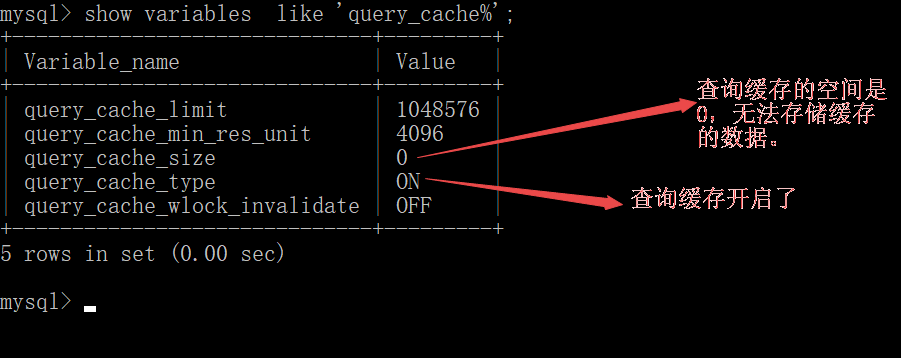
什么是查询缓存？

mysql服务器提供的，用于缓存select语句结果的一种内部内存缓存系统。

如果开启了查询缓存，将所有的查询结果，都缓存起来，使用同样的select语句，再次查询时，直接返回缓存的结果即可

查看缓存设置情况，并给缓存空间设置大小：

> show variables like ‘query\_cache%’; //查看缓存使用情况

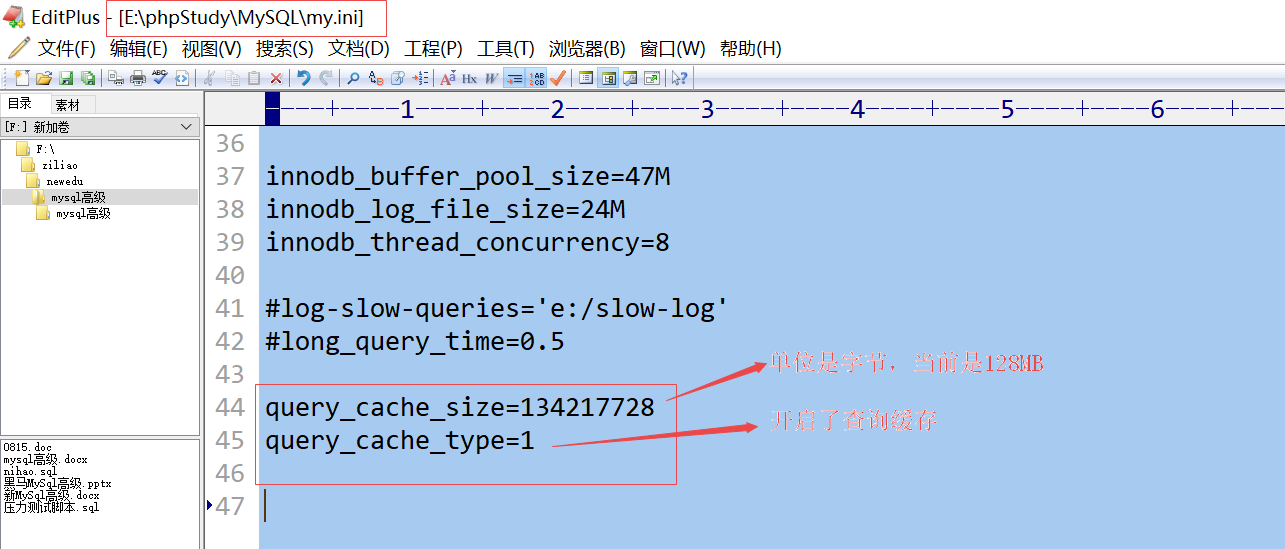


query\_cache\_size:缓存空间大小

query\_cache\_type:是否有开启缓存

如何开启查询缓存，并设置缓存空间大小？

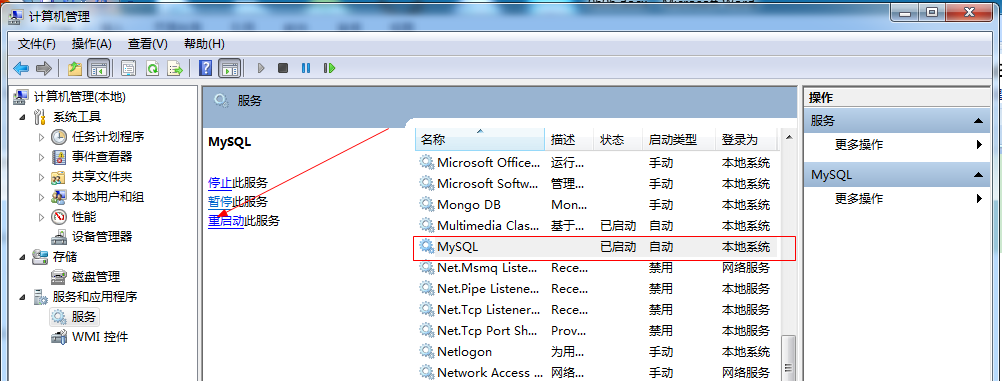
在my.ini中对上边的两个变量进行配置：



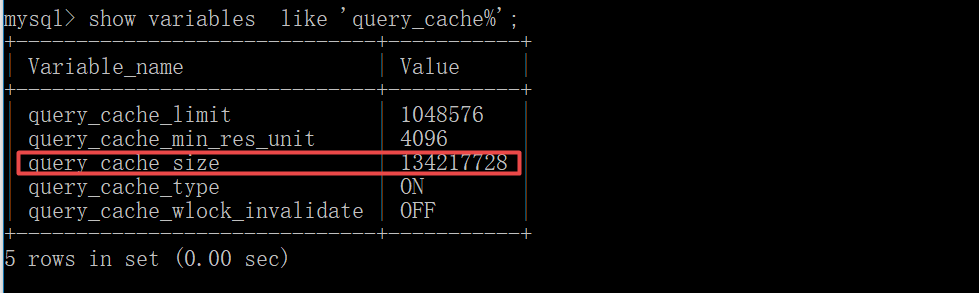
query\_cache\_size=134217728

query\_cache\_type=1

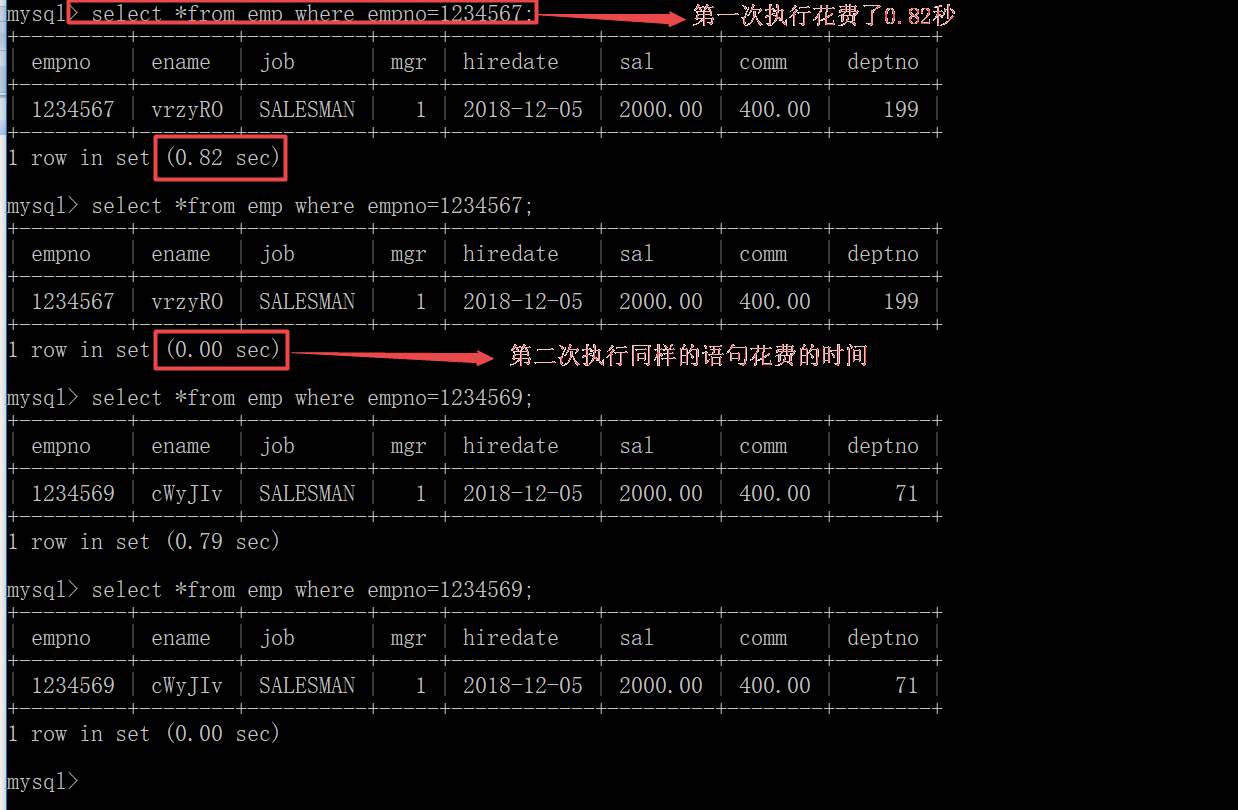
配置完成，之后需要重启mysql，



查看缓存开启成功：show variables like ‘query\_cache%’;



sql语句第一次执行没有缓存，之后就有缓存了：

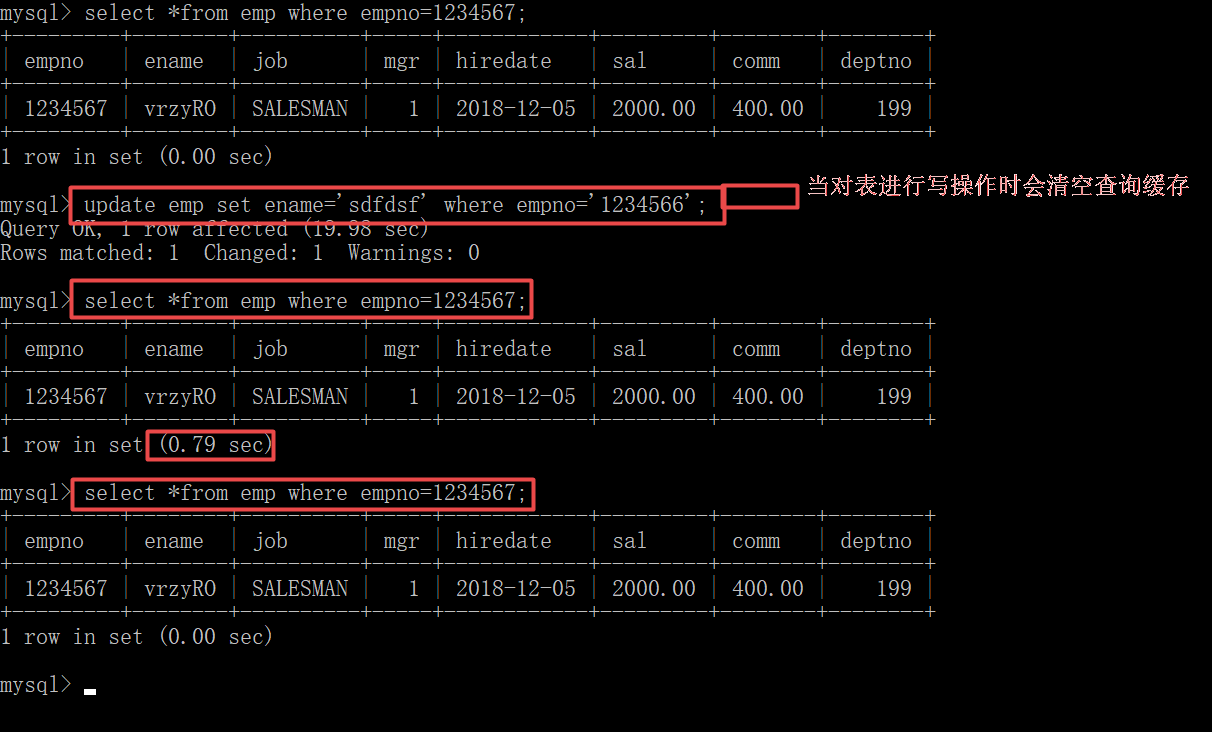


## 2、无缓存

### （1）缓存失效

数据表的数据(数据有修改)有变化 或者 数据表结构(字段的增、减)有变化，则会清空全部的缓存数据，即缓存失效。

update emp set job=’123456’ where empno=123456;



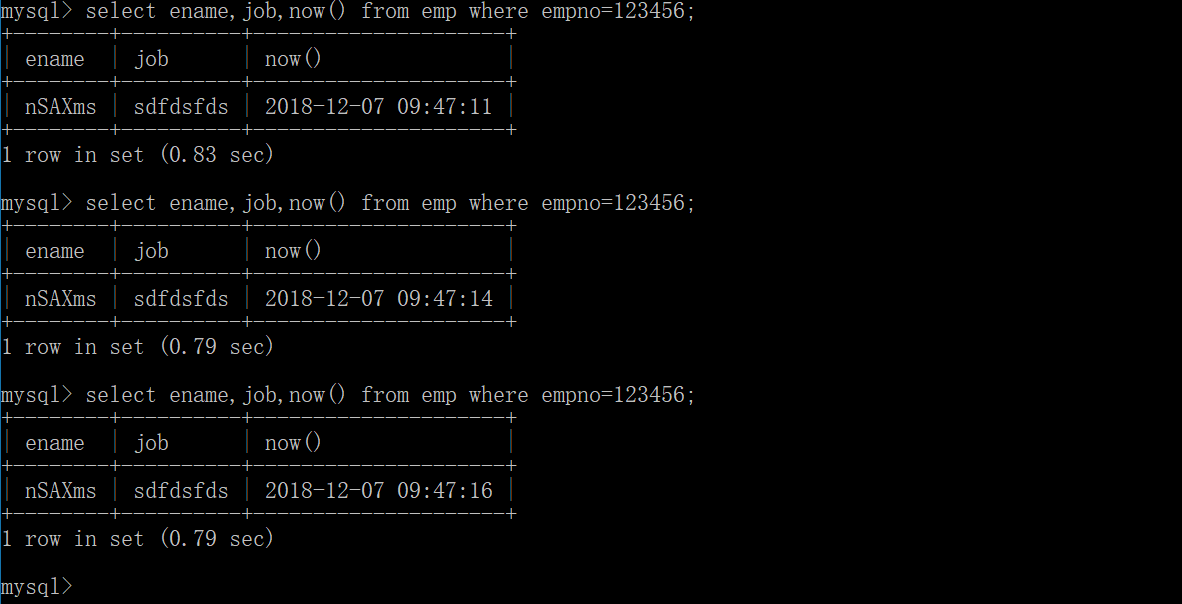
上图，执行了一个update语句，导致之前存在缓存(empno=1234567)被清空了

### （2）不使用缓存

sql语句有变化表达式，则不会生成/使用缓存。

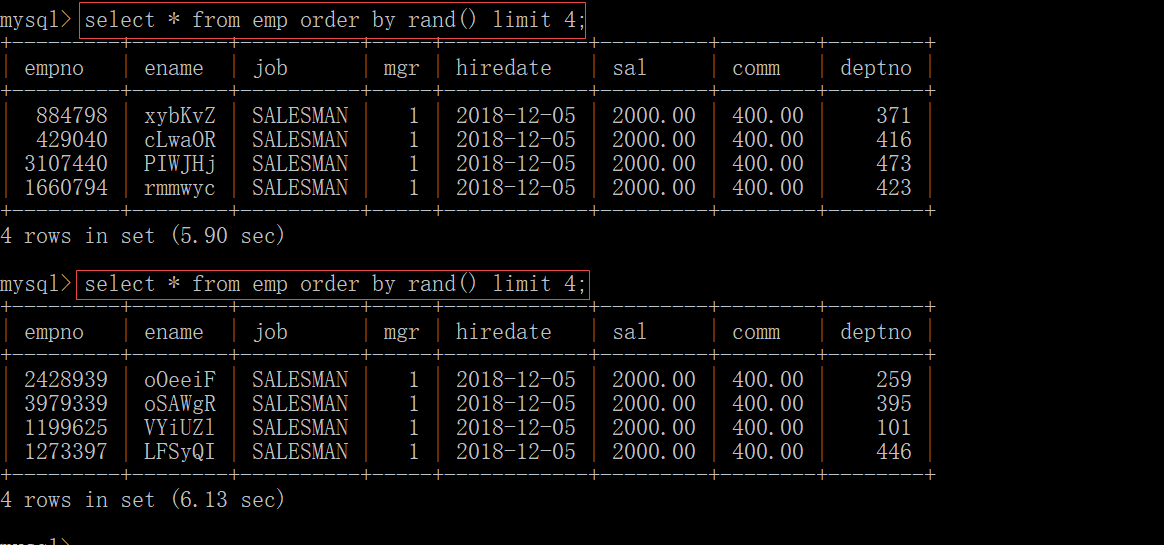
例如有 时间信息、随机数等

select ename,job,now() from emp where empno=123456;



上图，在sql语句中有“时间”变化的表达式，则不使用缓存

select \* from emp order by rand() limit 4;

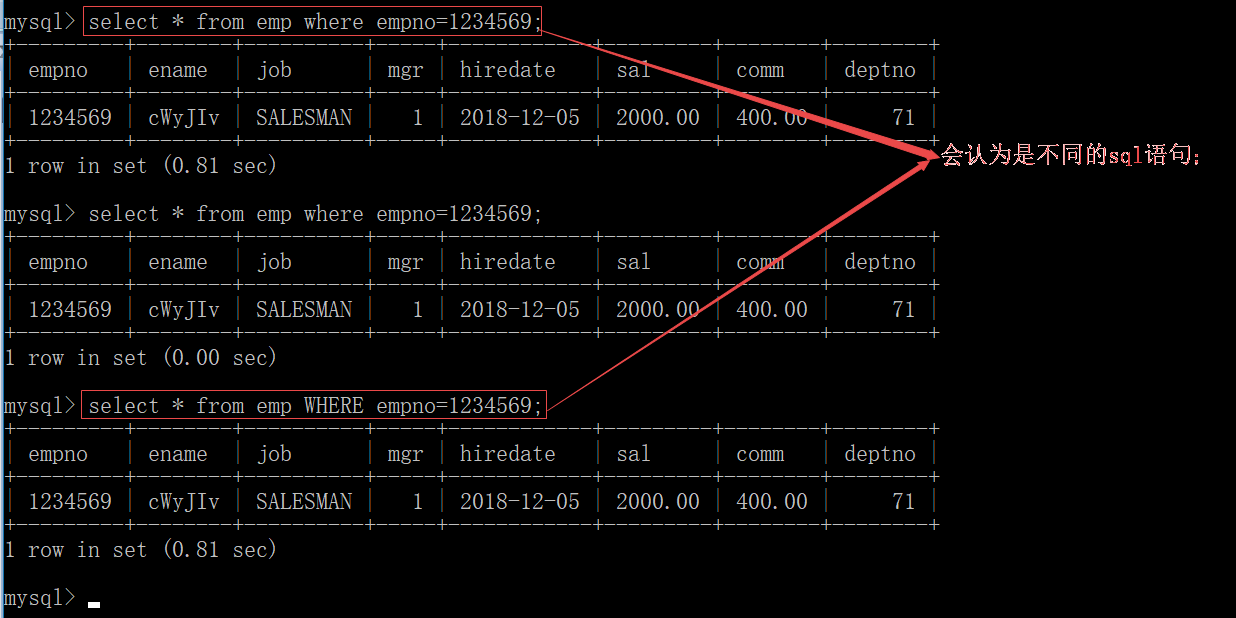


上图，sql语句中有“随机数”的表达式，不给使用缓存

### （3）生成多个缓存

生成缓存的sql语句对“空格”、“大小写”比较敏感

相同结果的sql语句，由于空格、大小写问题就会分别生成多个缓存。



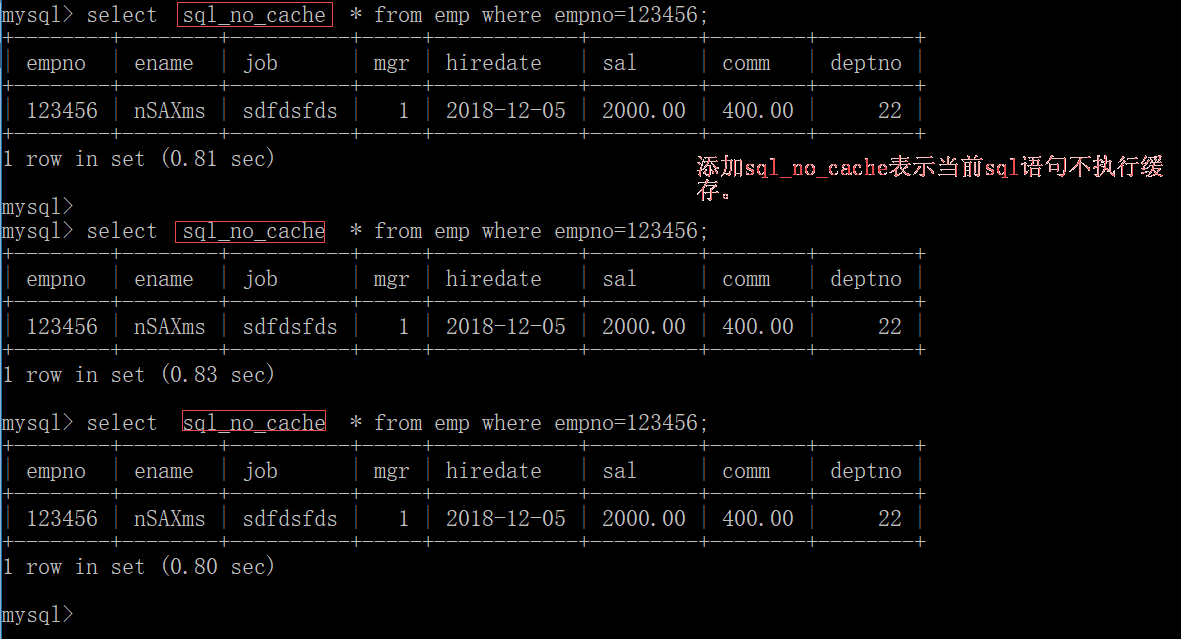
注意：相同结果的sql语句，由于大小写问题会分别生成缓存：

### （4）禁用缓存

sql\_no\_cache 不进行缓存

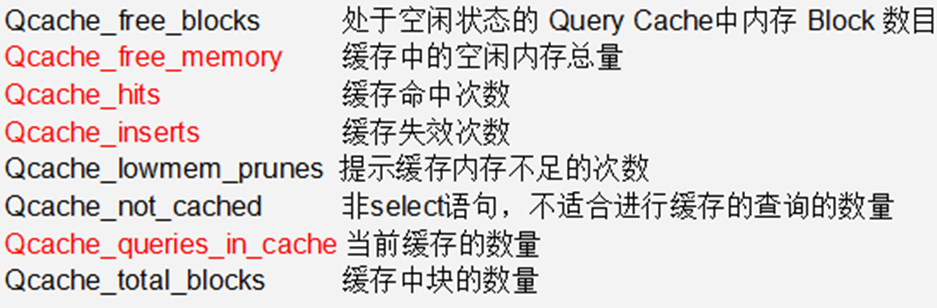
select sql\_no\_cache \* from emp where empno=123456;

意思是当前查询结果不使用查询缓存；

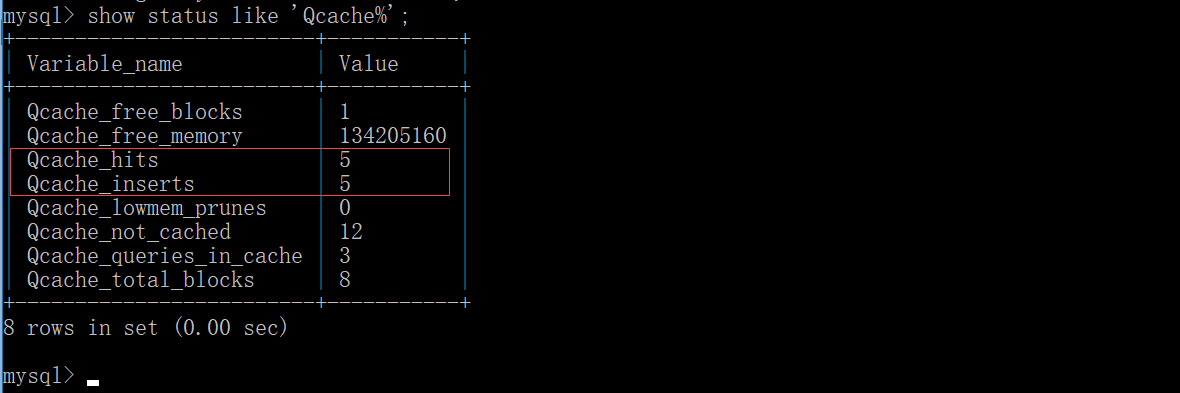


## 3、查看缓存空间使用情况

> show status like ‘Qcache%’; //查看缓存使用情况



如下图，再次使用一个缓存，并读取一次，发现缓存相关参数有变化：



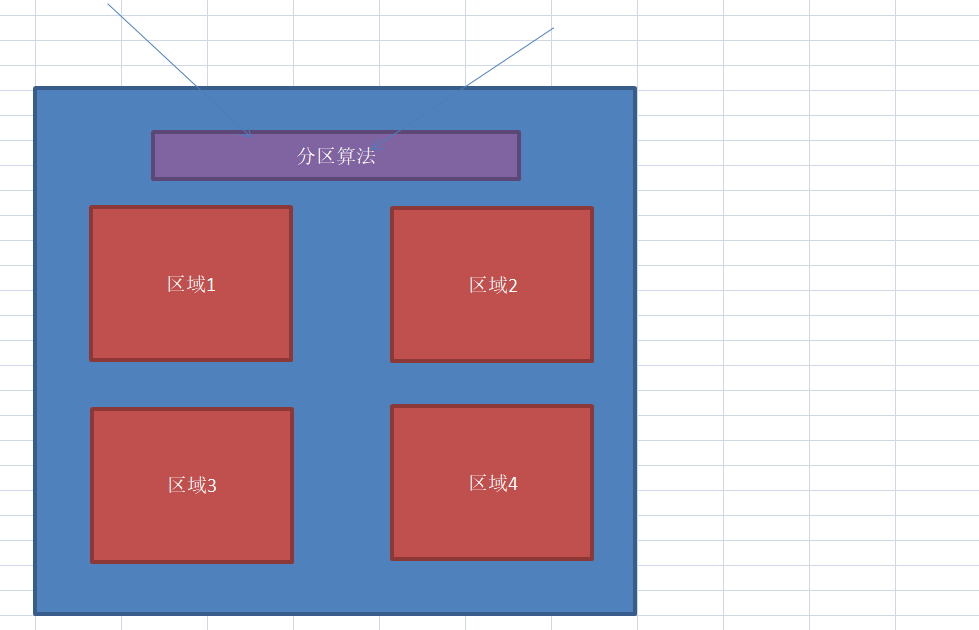
# 二．分区技术

## 1、分区介绍

基本概念，把一个表，从逻辑上分成多个区域，便于存储数据。

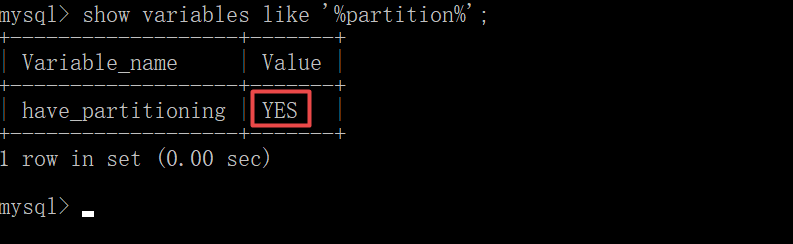
采用分区的前提，数据量非常大。

如果数据表的记录非常多，比如达到上亿条，数据表的活性就大大降低，数据表的运行速度就比较慢、效率低下，影响mysql数据库的整体性能，就可以采用分区解决，分区是mysql本身就支持的技术。



查看当前mysql软件是否支持分区；

show variables like '%partition%';



以上的结构，在创建（修改）表时，可以指定表，可以被分成几个区域。

利用表选项：partition 完成。

create table table\_name(

字段信息，

索引，

)engine myisam charser utf8

partition by 分区算法（分区字段）（

分区选项

）；

分区算法：

条件分区：list (列表) range(范围) 取模轮询（hash,key）

## 2、分区算法

### （1）list分区

**list :条件值为一个数据列表。**

通过预定义的列表的值来对数据进行分割

例子：假如你创建一个如下的一个表，该表保存有全国20家分公司的职员记录，这20家分公司的编号从1到20.而这20家分公司分布在全国4个区域，如下表所示：

职员表：emp

id name store\_id(分公司的id)

12 小宝 1

14 二宝 6

北部 1,4,5,6,17,18

南部 2,7,9,10,11,13

东部 3,12,19,20

西部 8,14,15,16

insert into emp values(12,’xiaobao’,14)

insert into emp values(15,’二bao’,17)

create table p\_list(

id int,

name varchar(32),

store\_id int

)engine myisam charset utf8

partition by list (store\_id)(

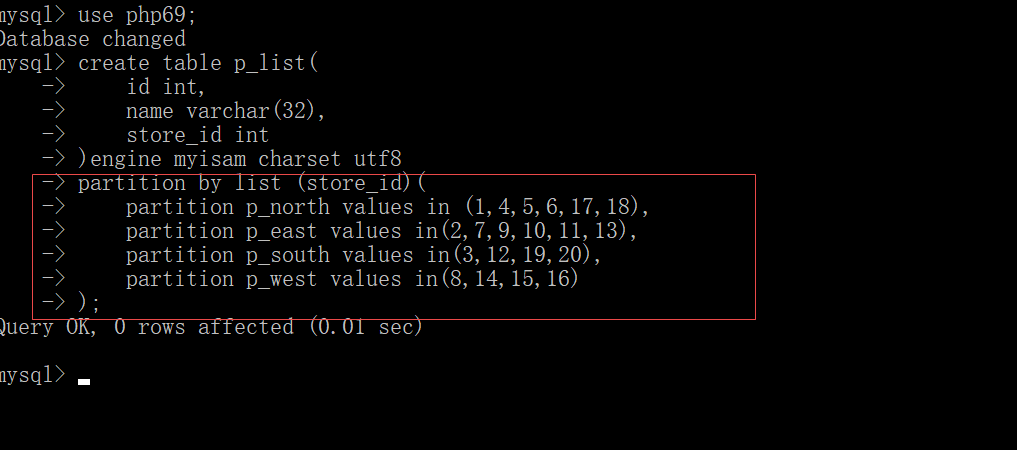
partition p\_north values in (1,4,5,6,17,18),

partition p\_east values in(2,7,9,10,11,13),

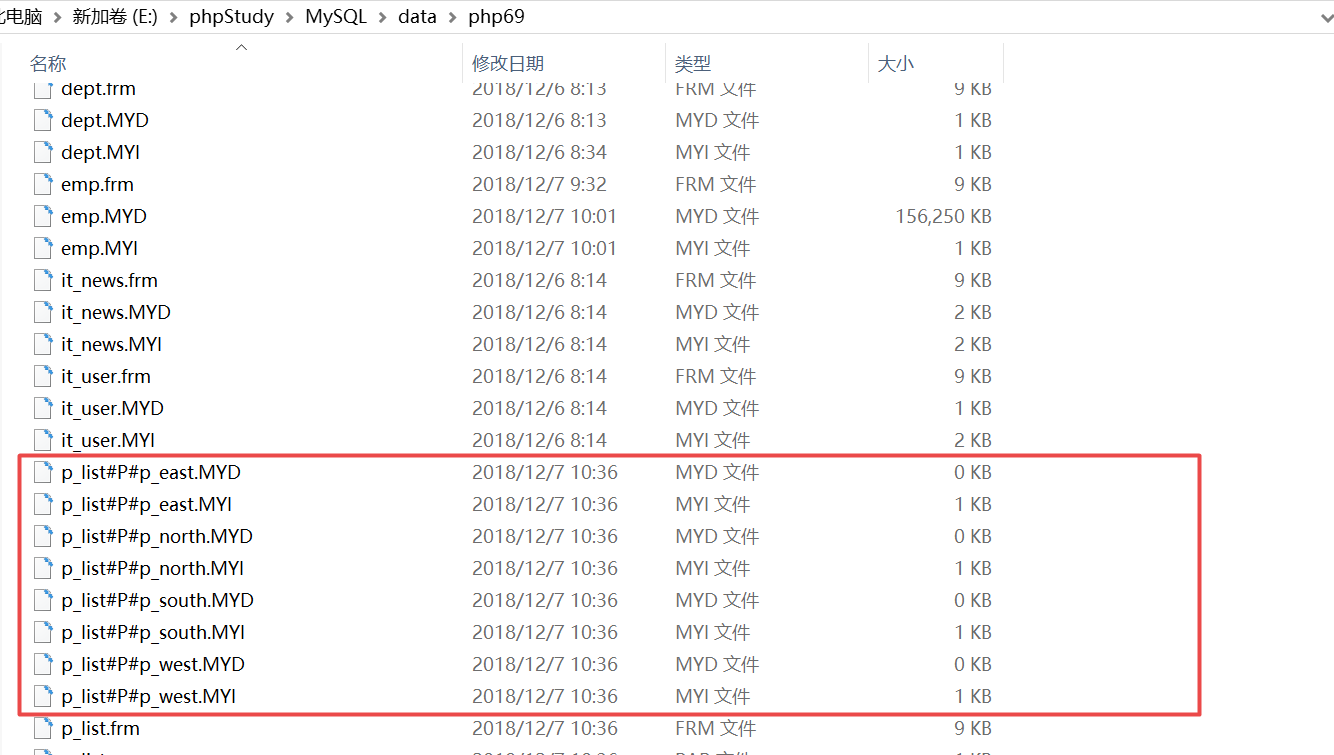
partition p\_south values in(3,12,19,20),

partition p\_west values in(8,14,15,16)

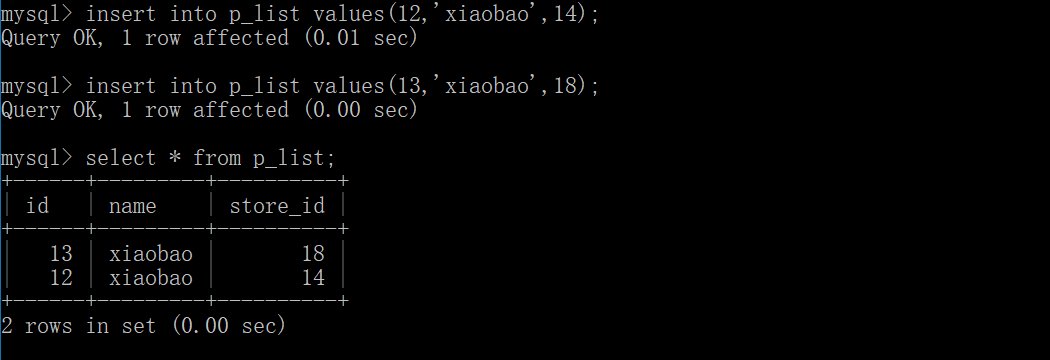
);



创建分区表后查看文件，

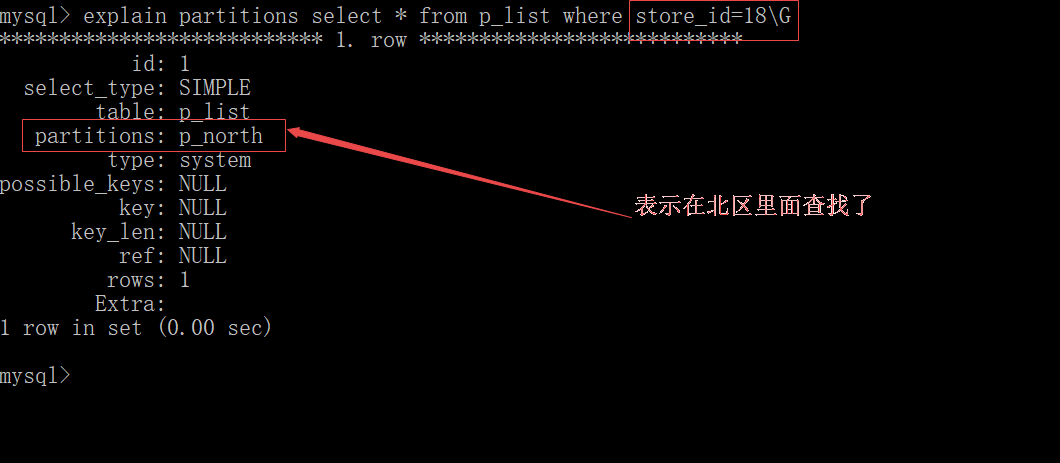


添加几条数据，测试是否用到了分区：

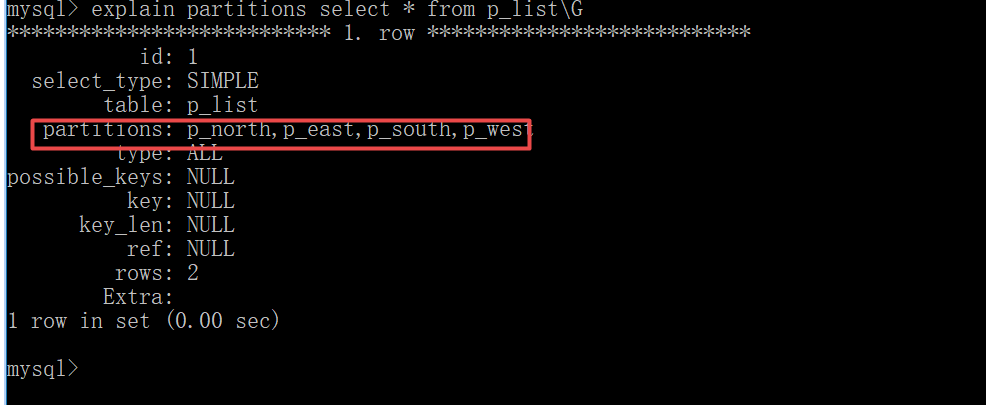


explain partitions select \* from p\_list where store\_id=20\G

**注意：在使用分区时，where后面的字段必须是分区字段，才能使用到分区。**



如下查询，没有分区条件，则会到所有的分区里面去查找，即便如此，查询效率也要比单表查询高。



### （2）Range（范围）

这种模式允许将数据划分不同范围。例如可以将一个表通过月份划分成若干个分区

create table p\_range(

id int,

name varchar(32),

birthday date

)engine myisam charset utf8

partition by range (month(birthday))(

partition p\_1 values less than (4),

partition p\_2 values less than(7),

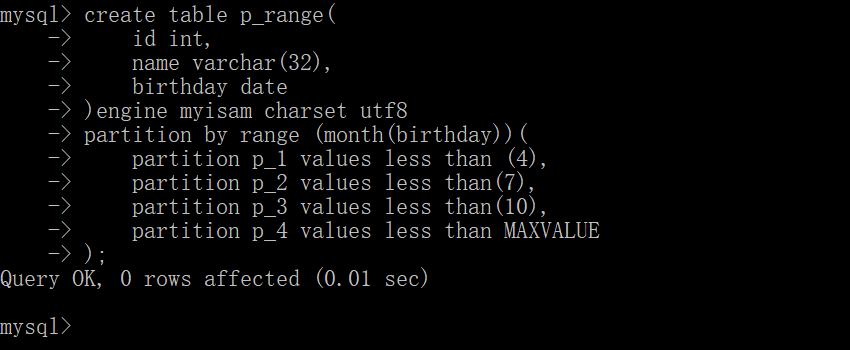
partition p\_3 values less than(10),

partition p\_4 values less than MAXVALUE

);

less than 小于;

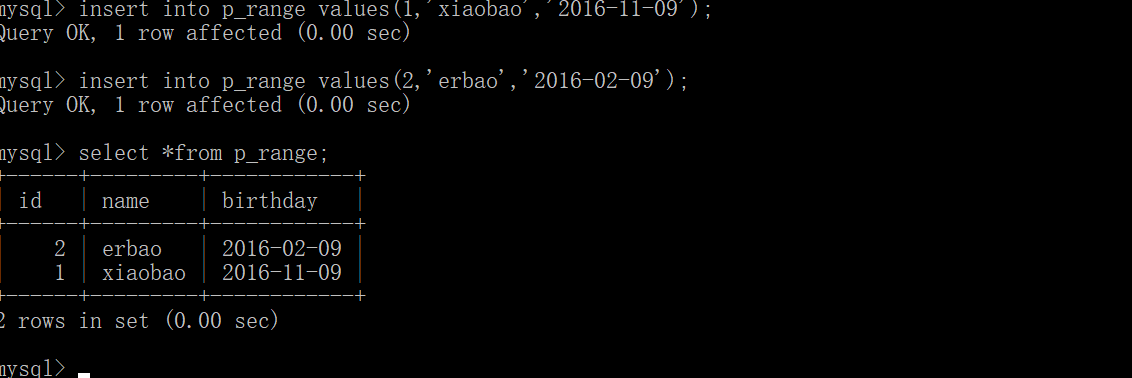
MAXVALUE 可能的最大值



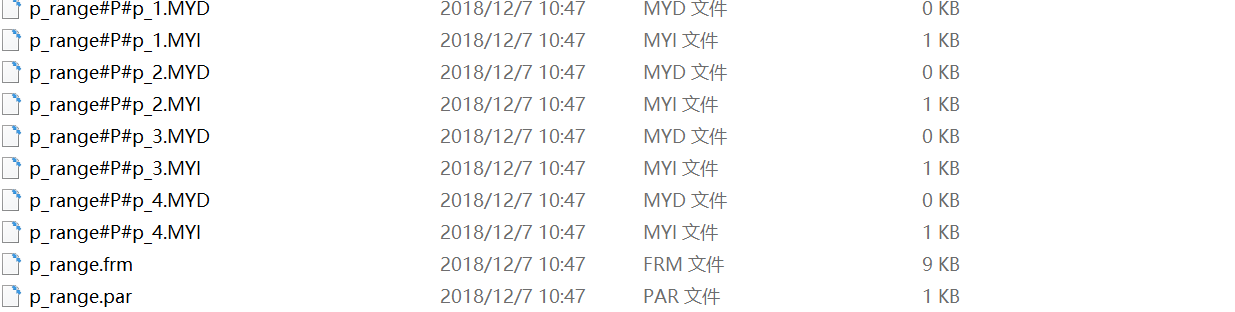
insert into p\_range values(1,’xiaobao’,’2016-09-09’);

insert into p\_range values(1,’xiaobao’,’2016-11-09’);

插入的数据如下；



分区的效果如下；



### （3）Hash（哈希）

这种模式允许通过对表的一个或多个列的Hash Key进行计算，最后通过这个Hash码不同数值对应的数据区域进行分区。例如可以建立一个对表主键进行分区的表。

create table p\_hash(

id int,

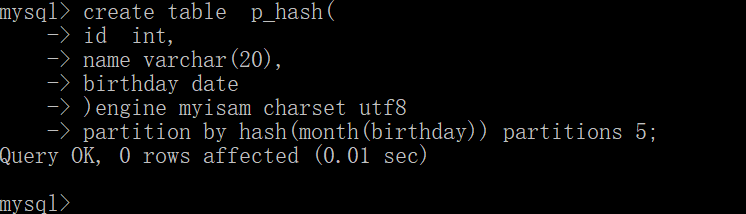
name varchar(20),

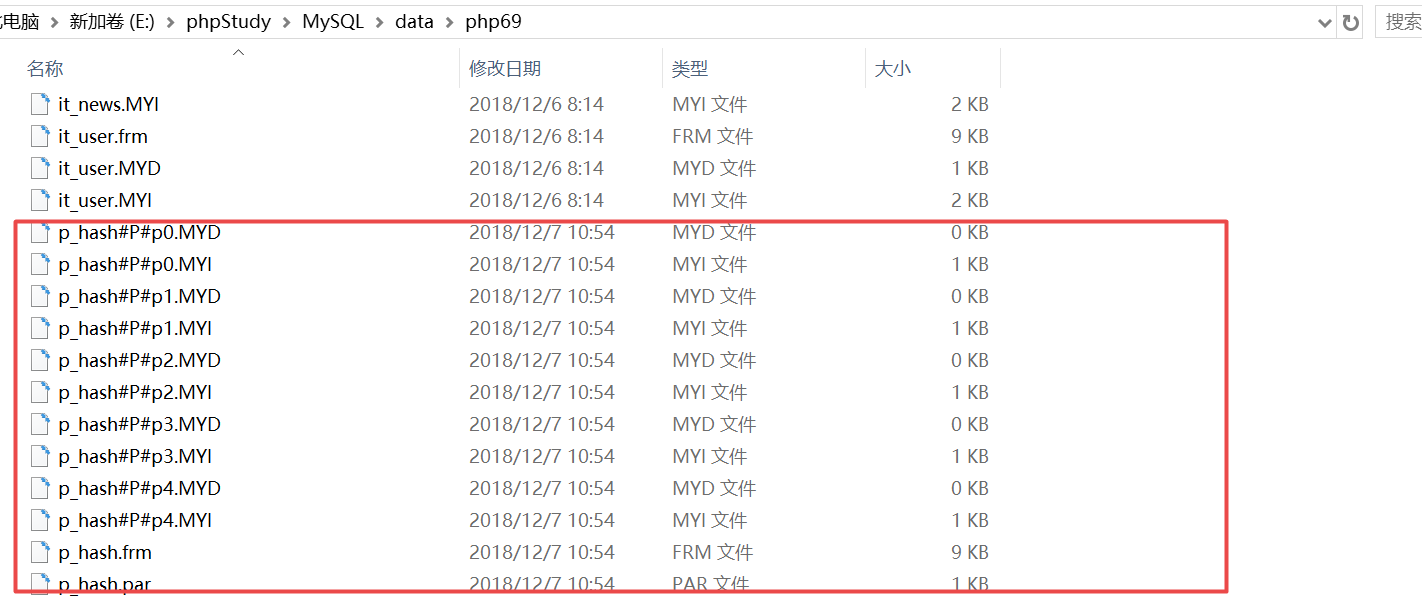
birthday date

)engine myisam charset utf8

partition by hash(month(birthday)) partitions 5;

分区效果如下；





### （4）Key（键值）

上面Hash模式的一种延伸，这里的Hash Key是MySQL系统产生的。

create table p\_key(

id int,

name varchar(32),

birthday date

)engine myisam charset utf8

partition by key (id) partitions 5;

## 3、 分区管理

具体就是对已经存在的分区进行增加、减少操作。

#### （1）删除分区

删除分区：

① 在key/hash领域不会造成数据丢失(删除分区后数据会重新整合到剩余的分区去)

② 在range/list领域会造成数据丢失

求余方式(key/hash):

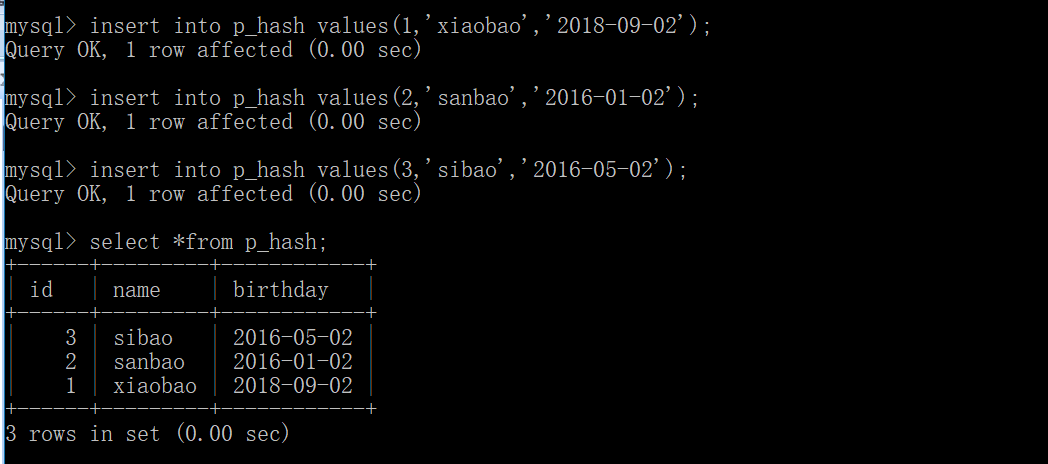
>alter table 表名 coalesce partition 数量;

范围方式(range/list):

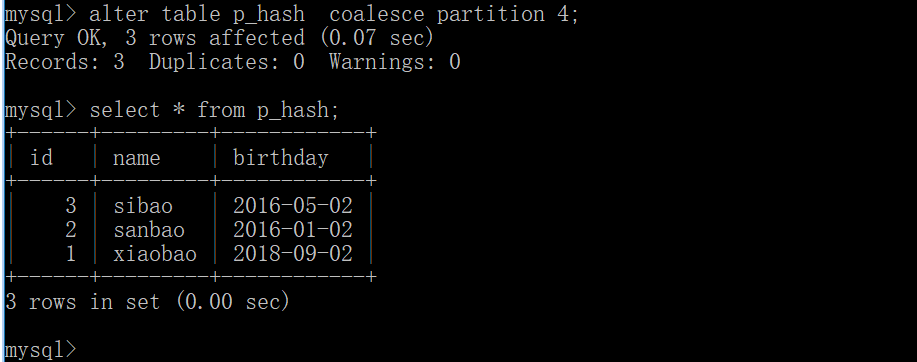
>alter table 表名 drop partition 分区名称;

1）删除hash类型分区

删除分区之前，数据如下

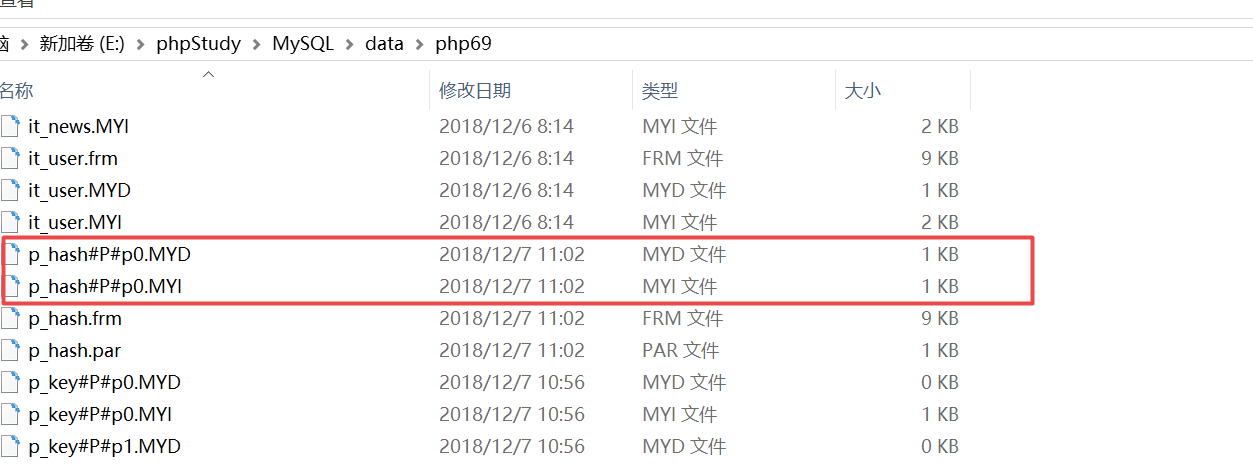


执行删除分区的操作：alter table p\_hash coalesce partition 4



上图，把5个分表中的4个都删除，只剩下一个

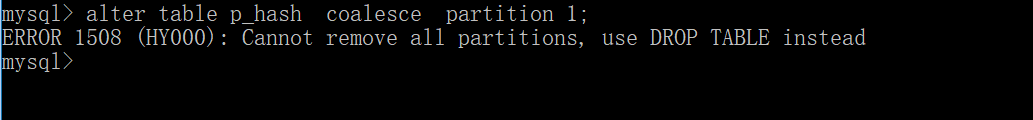
剩余一个分表效果：



并且，数据没有减少：

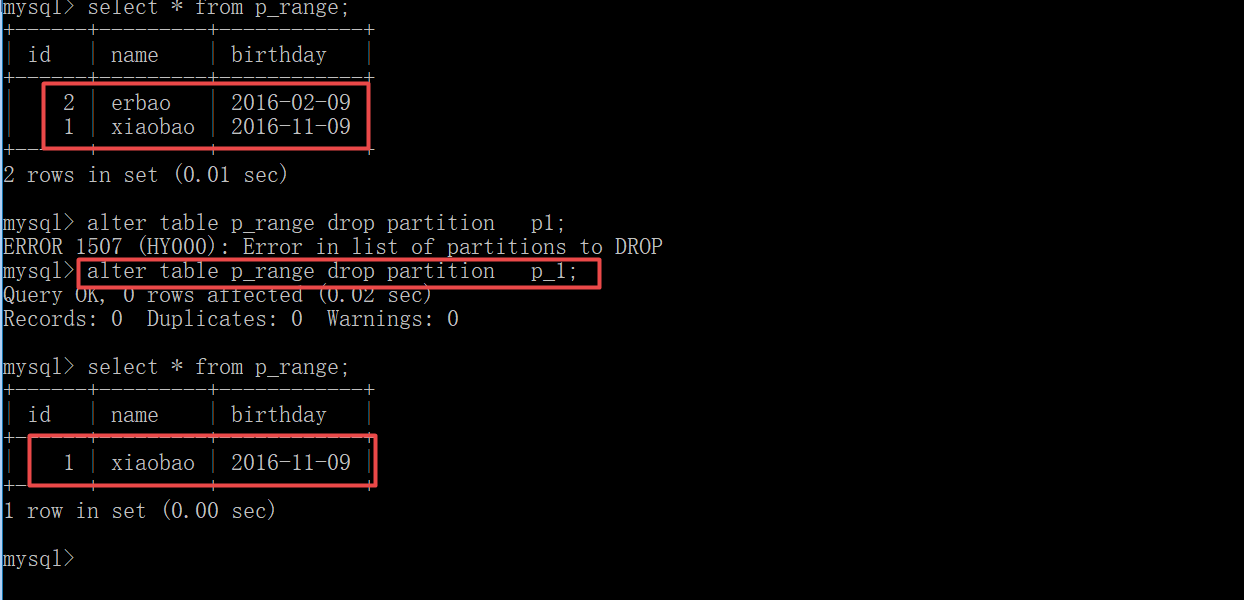
剩余唯一一个分区的时候，就禁止删除了，但是可以drop掉整个数据表，如下图：

alter table p\_hash coalesce partition 1;



2）删除list类型分表(数据有对应丢失)

alter table p\_list drop partition p\_north;



#### （2）增加分区

求余方式： key/hash

> alter table 表名 add partition partitions 数量;

范围方式： range/list

> alter table 表名 add partition(

partition 名称 values less than (常量)

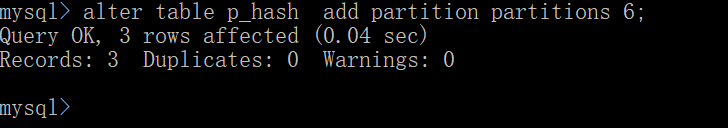
或

partition 名称 values in (n,n,n)

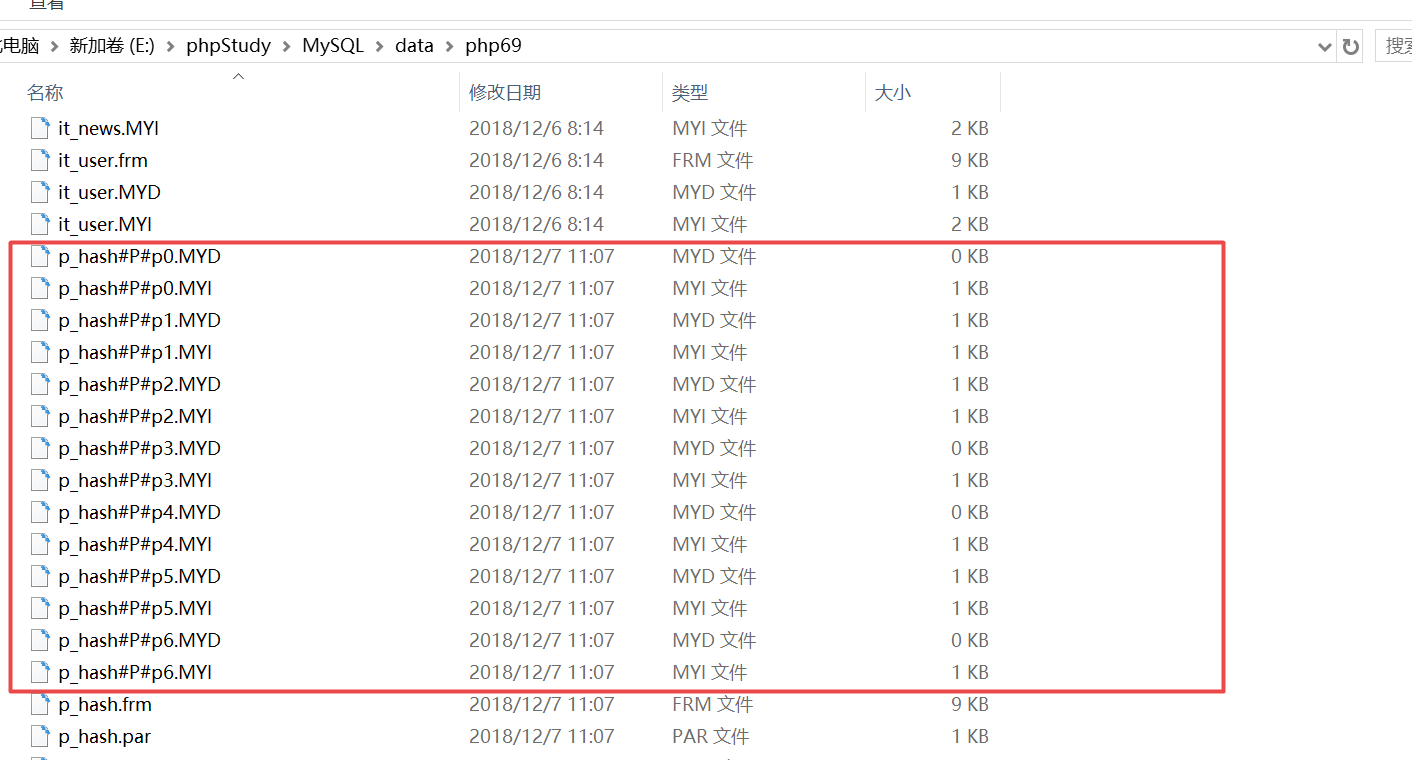
);

1) 给p\_hash 增加hash分表

alter table p\_hash add partition partitions 6;



增加后，一共有7个分表体现：



上图，分表增加好后，又把数据平均地分配给各个分表存储。

## 4、特别注意；

create table p\_range2(

id int primary key auto\_increment,

name varchar(32),

birthday date

)engine myisam charset utf8

partition by range (month(birthday))(

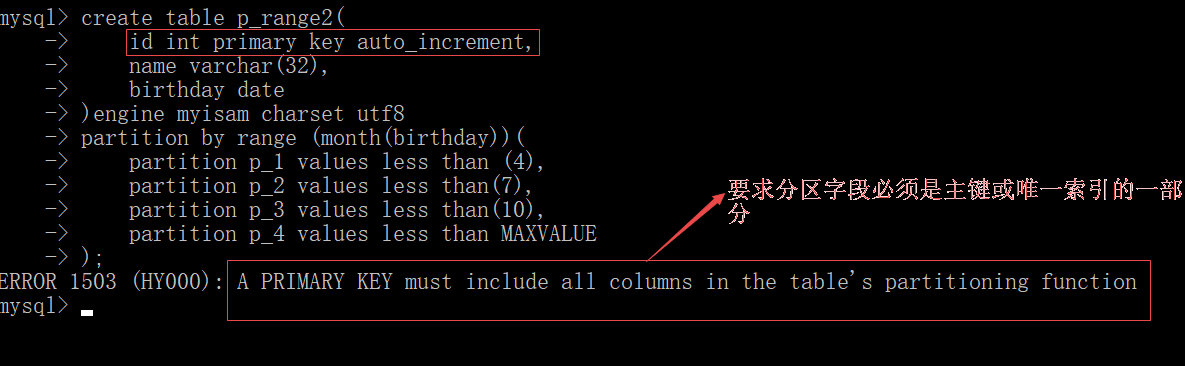
partition p\_1 values less than (4),

partition p\_2 values less than(7),

partition p\_3 values less than(10),

partition p\_4 values less than MAXVALUE

);

****

**注意：创建分区的字段必须是主键或唯一索引的一部分**

primary key(id,birthday)

不等价于如下量行代码；

primary key(id)

primary key(birthday)

create table p\_range2(

id int auto\_increment,

name varchar(32),

birthday date,

primary key(id,birthday)

)engine myisam charset utf8

partition by range (month(birthday))(

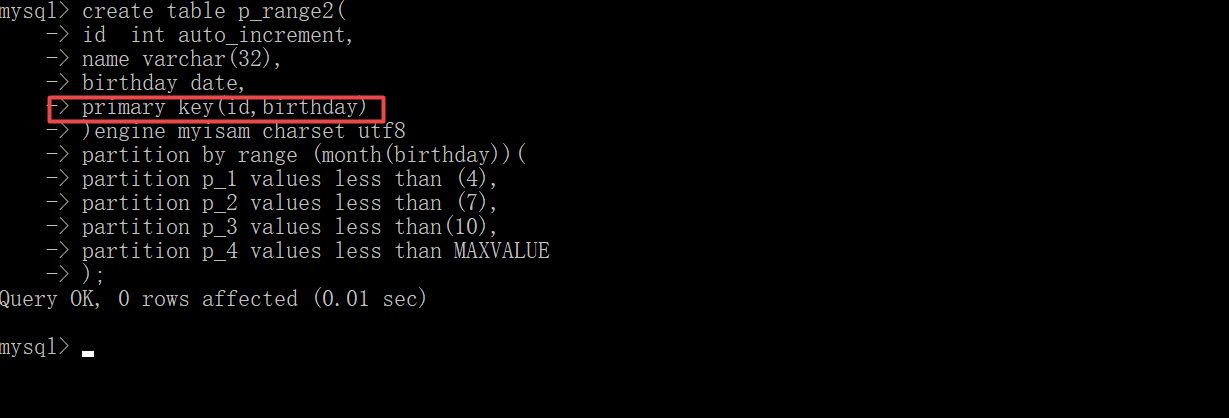
partition p\_1 values less than (4),

partition p\_2 values less than (7),

partition p\_3 values less than(10),

partition p\_4 values less than MAXVALUE

);

****

create table p\_range3(

id int auto\_increment,

name varchar(32),

birthday date,

unique key(id,birthday)

)engine myisam charset utf8

partition by range (month(birthday))(

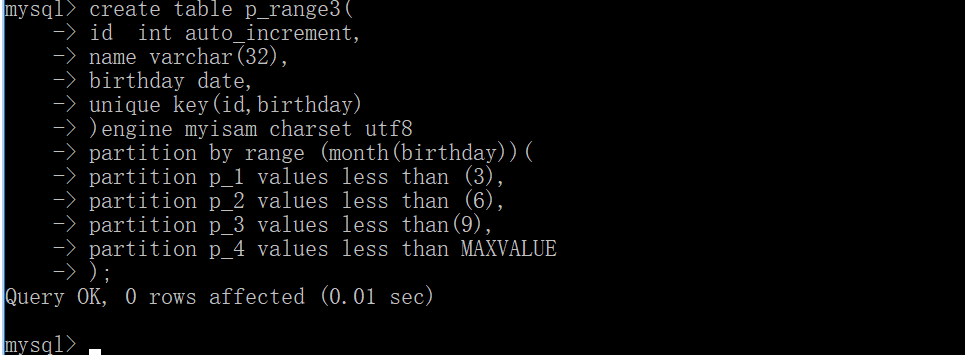
partition p\_1 values less than (3),

partition p\_2 values less than (6),

partition p\_3 values less than(9),

partition p\_4 values less than MAXVALUE

);



# 三、分表技术

## 1. 分表设计

物理方式分表设计

自己手动创建多个数据表出来

php程序需要考虑分表算法：数据往哪个表写，从哪个表读

分表1

PHP程序

分表2

分表算法

分表3

QQ的登录表。假设QQ的用户有10亿，如果只有一张表，每个用户登录的时候数据库都要从这10亿中查找，会很慢很慢。如果将这一张表分成100份，每张表有1000万条，就小了很多，比如qq0,qq1,qq1...qq99表。

用户登录的时候，可以将用户的id%100，那么会得到0-99的数，查询表的时候，将表名qq跟取模的数连接起来，就构建了表名。比如123456789用户，取模的89，那么就到qq89表查询，查询的时间将会大大缩短。

注册时，如何存储到多张表里面？

登录时，如何知道查询那张表？

注册时

$user\_id = $redis->incr(‘user\_id’);

表单提交过来的内容；

$username = ‘大宝’;

假如我们要分四张表来存储；

$user\_id%4 = 获取余数

假如$user\_id=8了 余数是0，那我们就存储到user\_0表里面了，

user\_0表里面的字段 id $user\_id ‘大宝’

$redis->set($username\_register\_name,$user\_id)

登录时，我们用名称来登录；

$username = ‘大宝’,如何知道该名称在那张表里面呢？

$username->user\_id->通过user\_id算出存储的表；

## 2. 垂直分表（比较常用）

水平分表：是把一个表的全部记录信息分别存储到不同的分表之中。

垂直分表：是把一个表的全部字段分别存储到不同的表里边。

有的时候，一个数据表设计好了，里边有许多字段，但是这些字段有的是经常使用的，有的是不常用的。在进行正常数据表操作的时候，不常用的字段也会占据一定的资源，对整体操作的性能造成一定的干扰、影响。

为了减少资源的开销、提升运行效率，就可以把不常用的字段给创建到一个专门的辅表中去。

同一个业务表的不同字段分别存储到不同数据表的过程就是“垂直分表”。

例如：

会员数据表有如下字段：

会员表： user\_id 登录名 密码 邮箱 手机号码 身高 体重 性别 家庭地址 身份证号码

以上表，红色是常用的，蓝色的是不常用的

为了使得常用字段运行速度更快、效率更高，把常用字段给调出来,因此数据表做以下垂直分表设计：

会员表(主)user字段：user\_id 登录名 密码 邮箱 手机号码

会员表(辅)user\_fu字段：user\_id 身高 体重 性别 家庭地址 身份证号码

以上把会员表根据字段是否常用给分为两个表的过程就是垂直分表。

存储文章

经常查询的数据 title（标题） author（作者）

# 四、数据碎片与维护

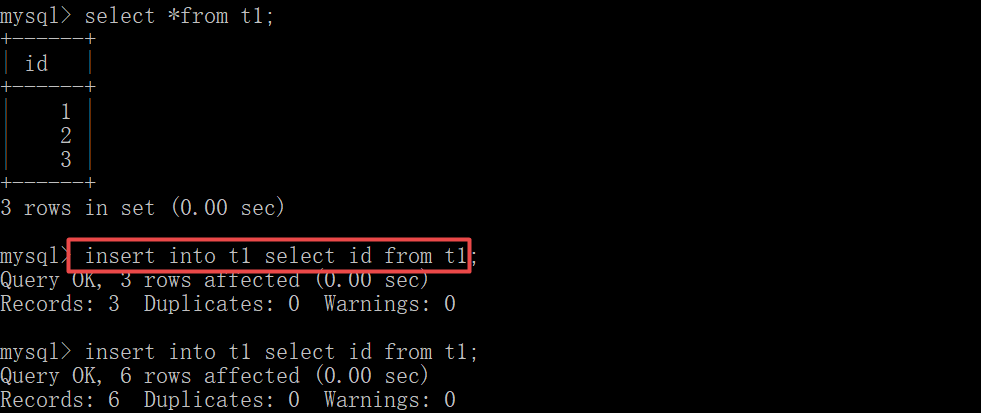
在长期的数据更改过程中，索引文件和数据文件，都将产生**空洞，形成碎片**，我们可以通过一个操作（不产生对数据实质影响的操作）来修改表，

建表语句：

create table t1(id int)engine myisam;

insert into t1 values(1),(2),(3)

insert into t1 select \* from t1;



表的原始大小：



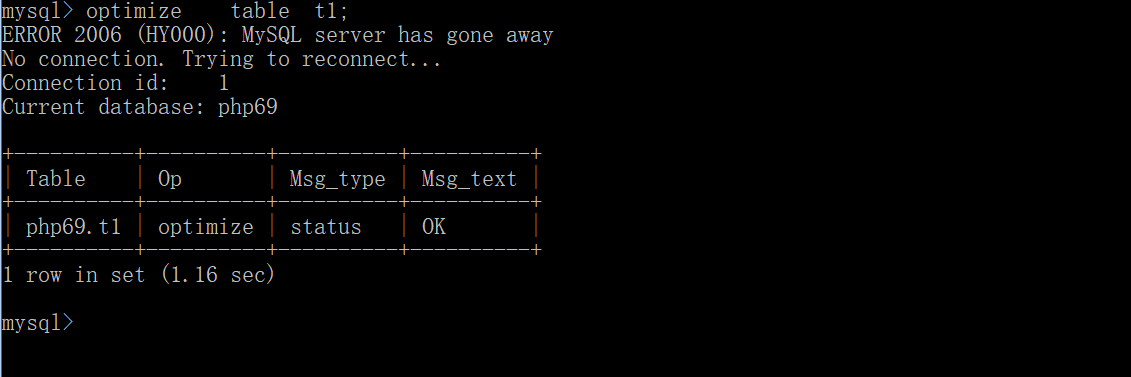
删除了一部分数据，应该表的容量会减少一部分，但是没有减掉，





开始整理：

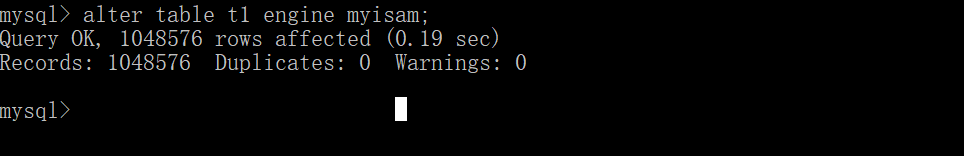
optimize table 表名;



整理后的结果，容量减少了一部分。



比如：表的引擎为innodb,可以**alter table xxx engine innodb**



**optimize table 表名，也可以修复。**

注意：修复表的数据及索引碎片，就会把所有的数据文件重新整理一遍，使之对齐，这个过程，如果表的行数比较大，也是比较耗费资源的操作，所以，不能频繁的修复。

如果表的update，delete操作很频繁，可以按周月来修复。

# 五、范式讲解

## 第一范式：

我们的表要满足两个条件：

（1）表的属性（列）要具有原子性（不可分割）

（2）表不能有重复的列，

只要是关系型数据库，就天然的满足第一范式。

关系型数据库:有行和列的概念，即为二维表格，常见的有mysql , sql server, oracle , informix ， db2, postgresql

非关系型数据库：面向对象和集合，没有行和列的概念。

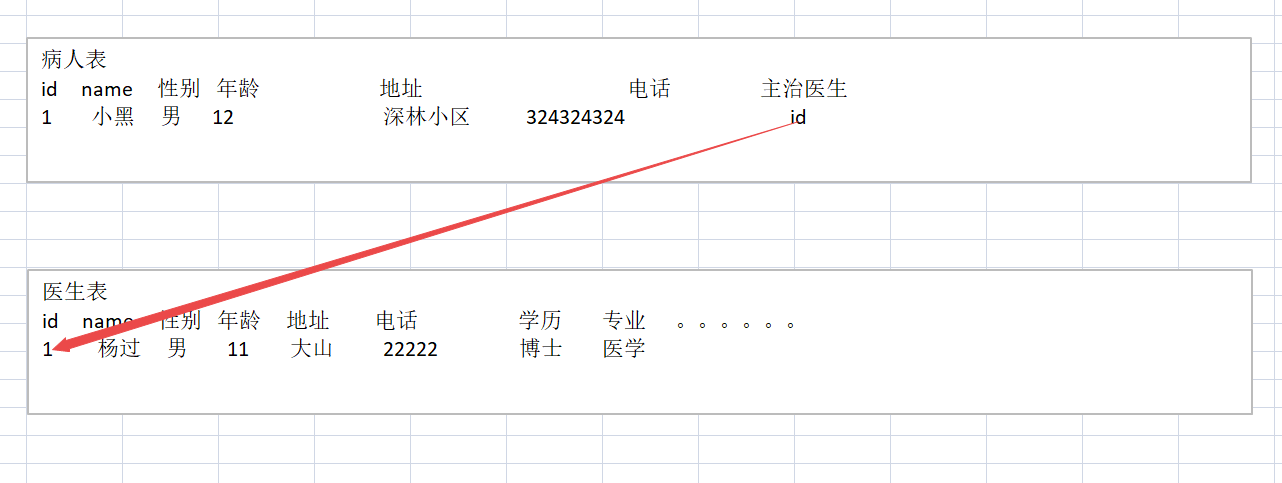
## 第二范式

表要满足：不能存在完全相同的两条记录，通常是通过设置一个主键来实现，主键一般是非业务逻辑主键。

## 第三范式

表中不能存在冗余数据，表中列的值，如果可以通过推导出来，则就不应该设置该列。

病人信息表应该为：

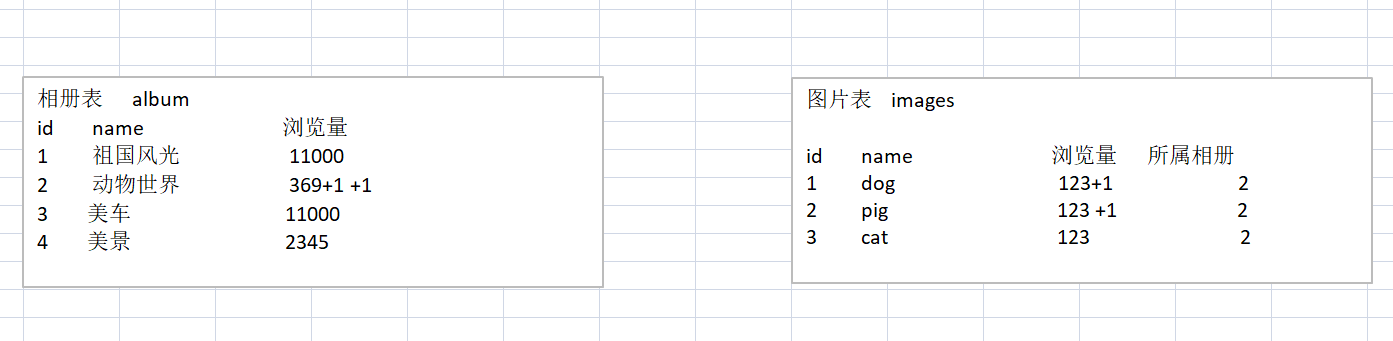


## 反三范式（逆范式）

有的时候基于性能考虑，需要有意违反 三范式，适度的冗余，以达到提高查询效率的目的。

相册浏览次数设计案例：





# 六、视图

## 1、视图的定义

视图的定义：

视图是由**查询结果**形成的一张**虚拟表**，是表通过某种运算得到的一个投影。

**创建视图的语法：**

create view view\_name as select 语句

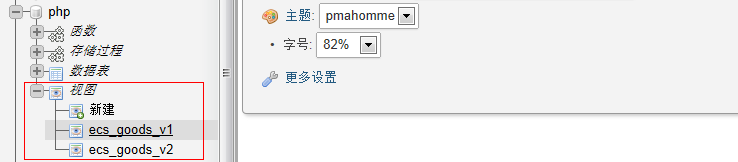
说明：

（1）**视图名跟表名是一个级别的名字，隶属于数据库；**

（2）该语句的含义可以理解为：就是将该select命名为该名字（视图名）；

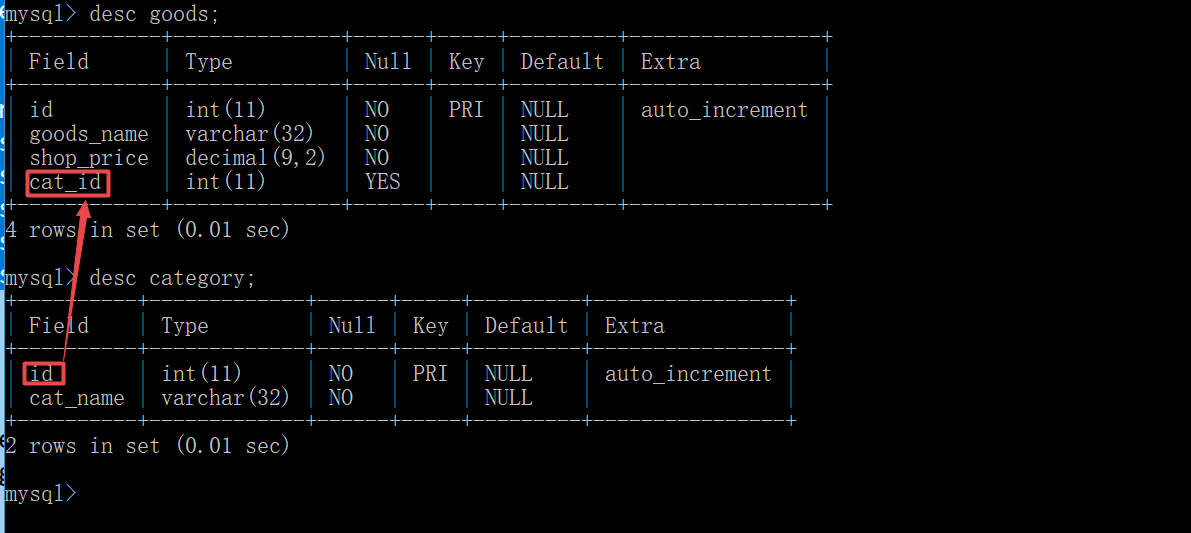
（3）视图也可以设定自己的字段名，而不是select语句本身的字段名——通常不设置。

（4）视图的使用，几乎跟表一样！



## 2、视图的作用

准备测试数据；goods表和category表；

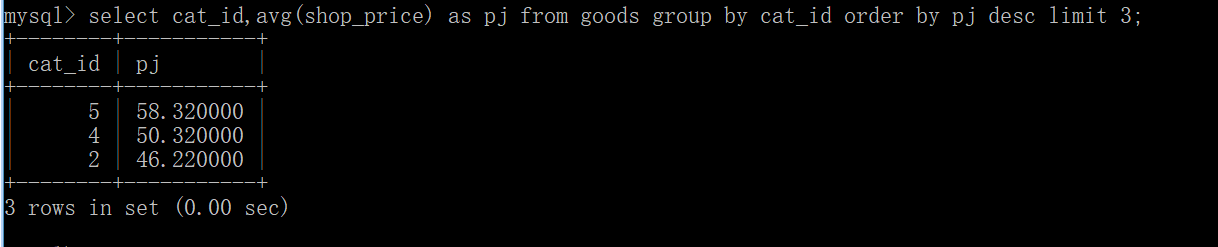


**（1）可以简化查询。**

**案例1：查询平均价格前3高的栏目。**

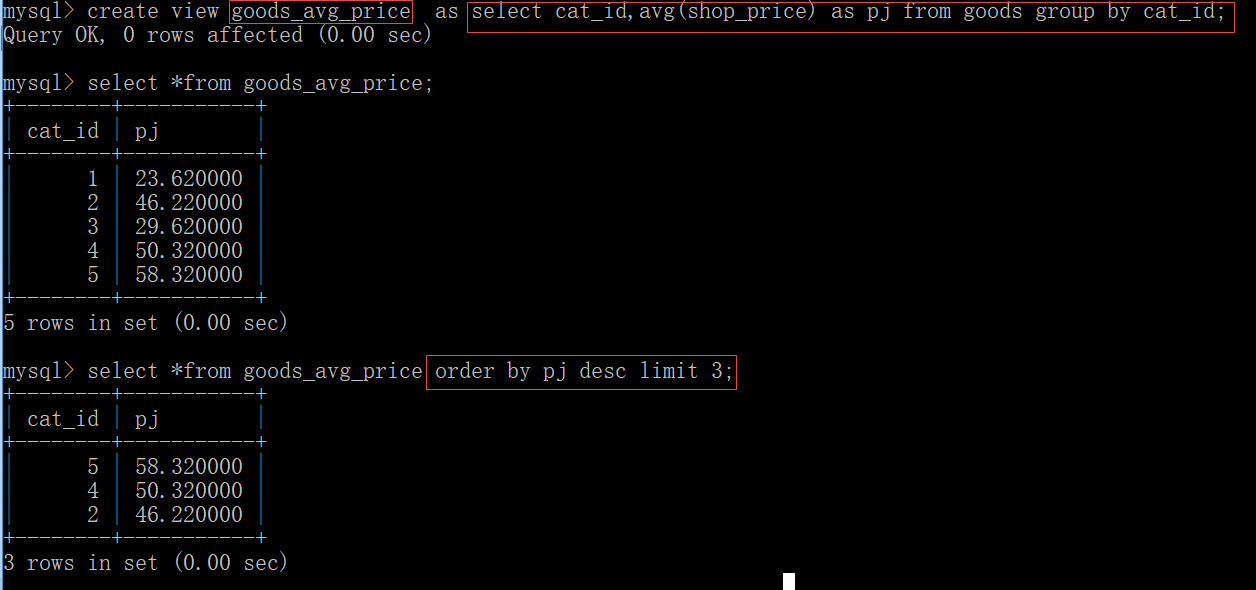
传统的sql语句写法

select cat\_id,avg(shop\_price) pj from goods group by cat\_id order by pj desc limit 3;



创建一个视图

create view ecs\_goods\_v1 as select cat\_id,avg(shop\_price) pj from ecs\_goods group by cat\_id;



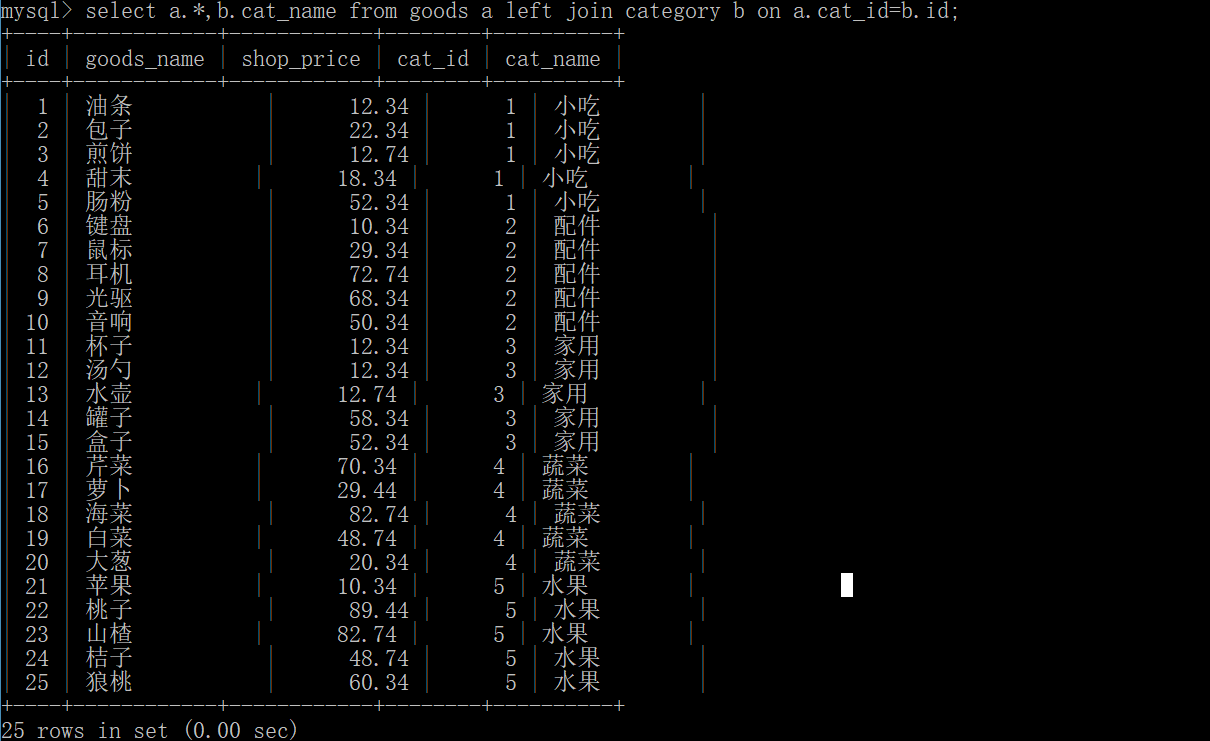
创建好了视图，**再次查询平均价格前3高的栏目时，我们就可以直接查询视图**

**select \* from ecs\_goods\_v1 order by pj desc limit 3;**

案例2:查询出商品表，以及所在的栏目名称；

传统的写法

select goods\_id,goods\_name,b.cat\_name from ecs\_goods a left join ecs\_category b on a.cat\_id=b.cat\_id;

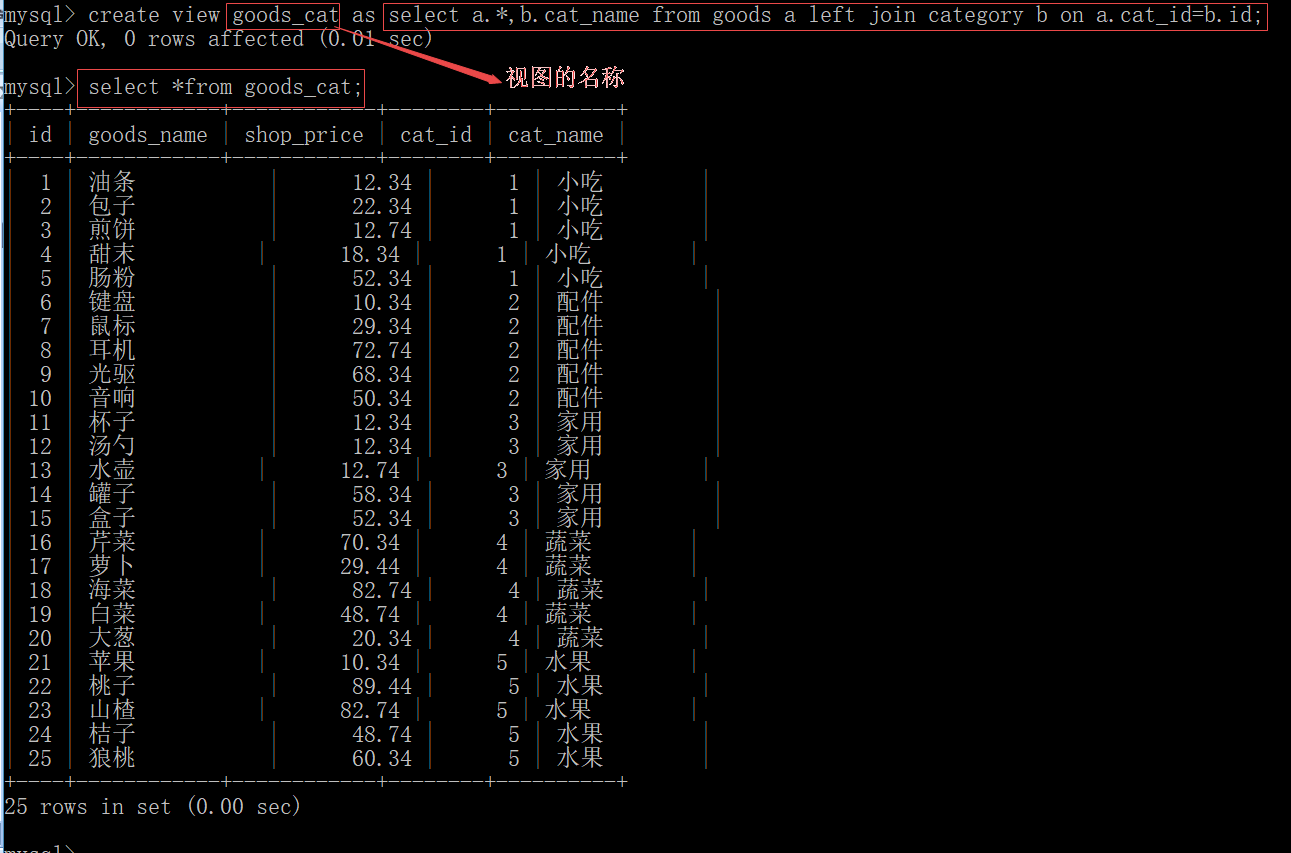


创建一个视图

create view ecs\_goods\_v2 as select goods\_id,goods\_name,b.cat\_name from ecs\_goods a left join ecs\_category b on a.cat\_id=b.cat\_id;

查询视图；

select \* from ecs\_goods\_v2;



（2）可以进行权限控制，

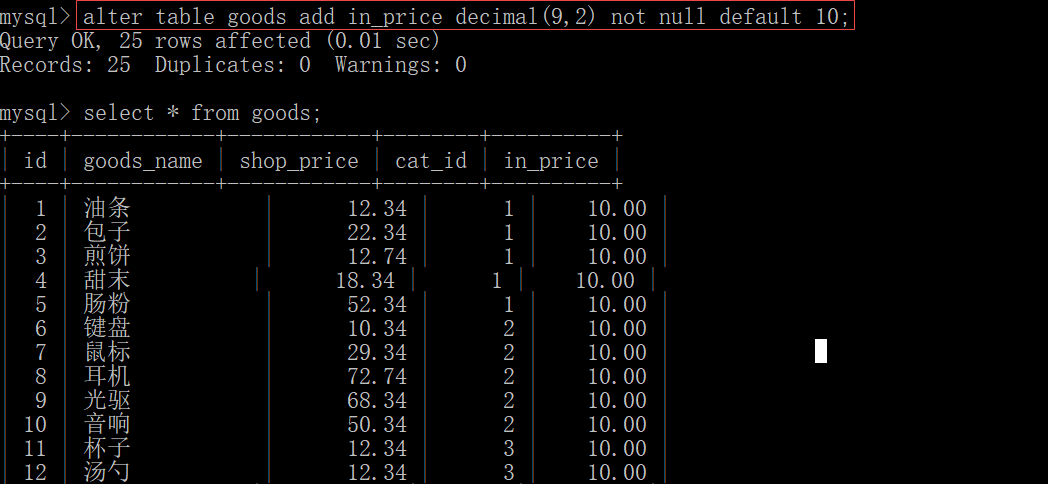
把表的权限封闭，但是开放相应的视图权限，视图里只开放部分数据，比如某张表，用户表为例，2个网站搞合作，可以查询对方网站的用户，需要向对方开放用户表的权限，但是呢，又不想开放用户表中的密码字段。

再比如一个goods表，两个网站搞合作，可以相互查询对方的商品表，比如进货价格字段不能让对方查看。

案例：

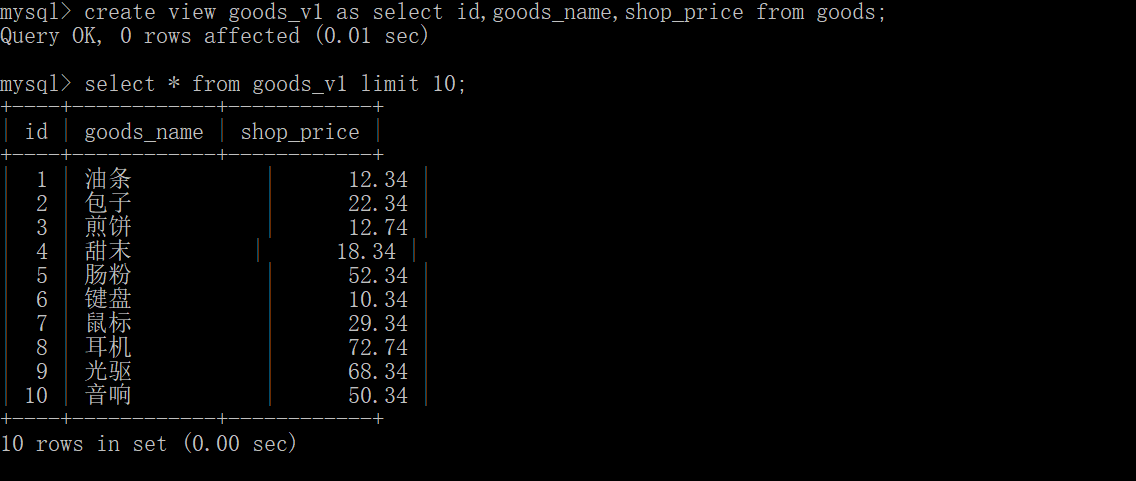
（1）创建一个goods表，添加几条数据

给测试的goods表添加一个in\_price(进货价格)字段；



（2）创建一个视图

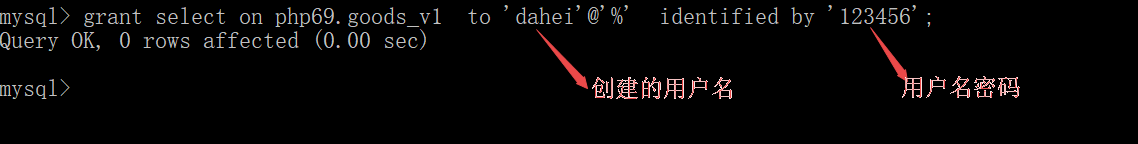
create view goods\_v1 as select id,goods\_name,shop\_price from goods;



（3）授权一个账号

grant 权限 on 数据库名称.视图名或表名 to ‘用户名称’@’%’ identified by ‘密码’’

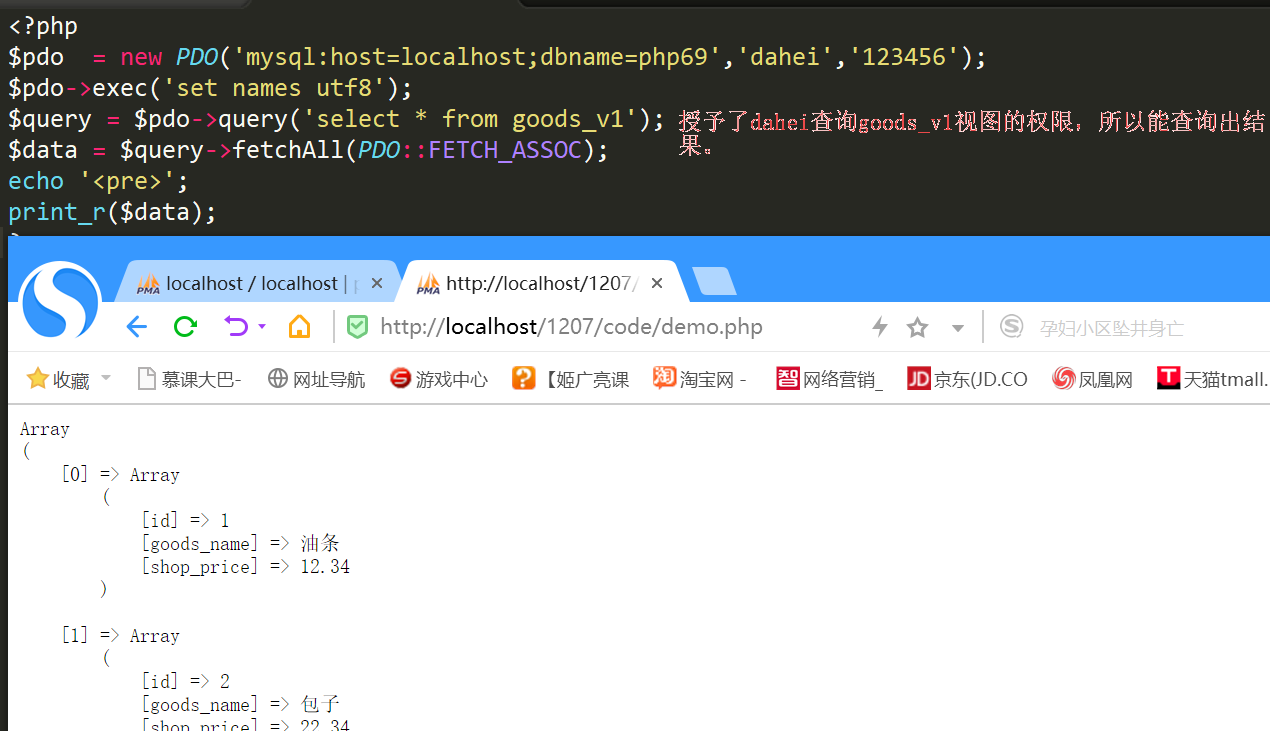
grant select on php.goods\_v1 to ‘xiaolei’@’%’ identified by ‘1234’’



以上语句，表示创建了一个 dahei的用户，密码是123456，权限时再php69库下面的goods\_v1视图具有查询的权限；

（4）案例测试；

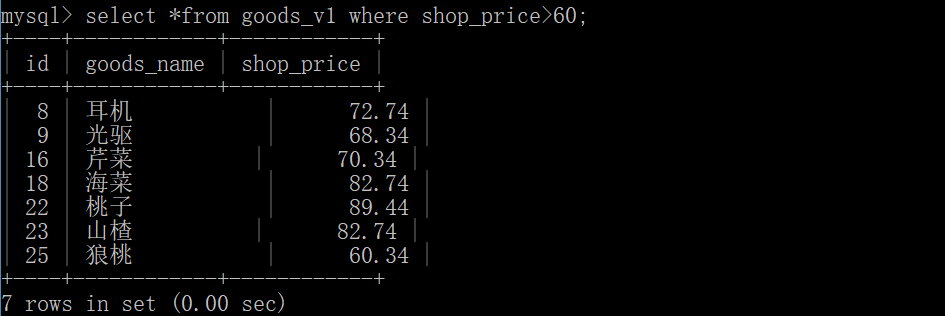




## 3、查询视图

语法：select \* from 视图名 [where 条件]

视图和表一样，可以添加where 条件



## 4、修改视图

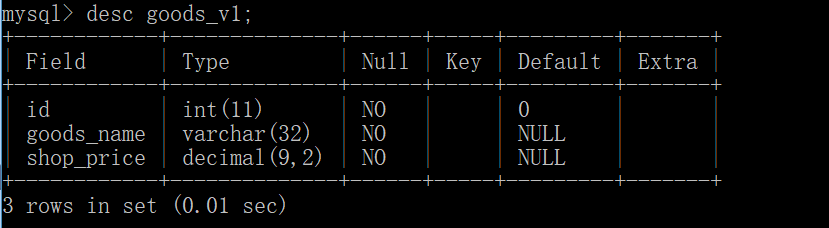
alter view view\_name as select XXXX

## 5、删除视图

drop view 视图名称

## 6、查看视图结构

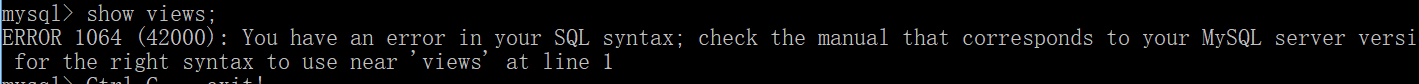
和表一样的，语法，desc 视图名称

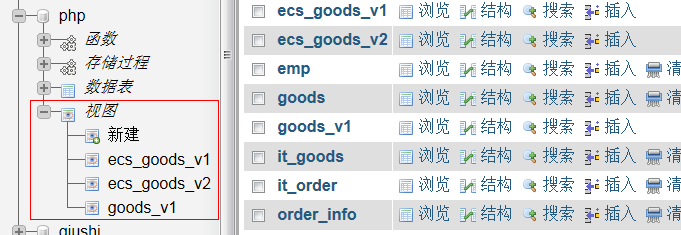


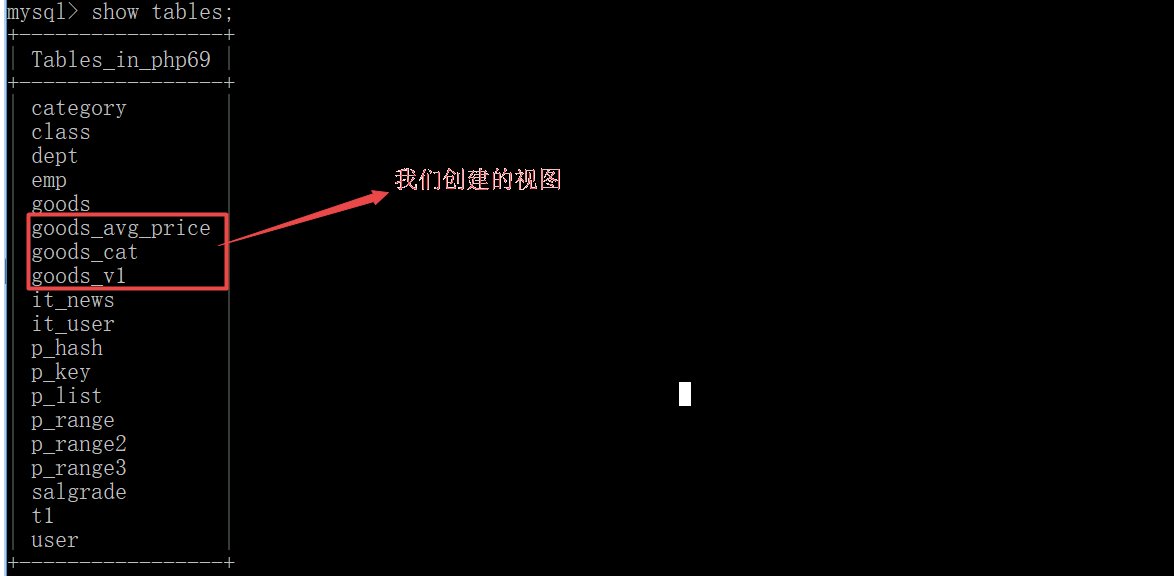
## 7、查看所有视图

和表一样，语法：show tables;

注意：没有show views语句；

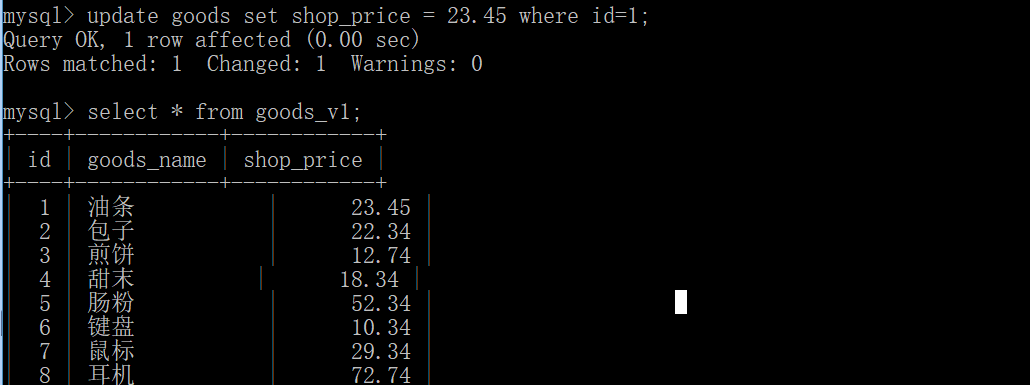




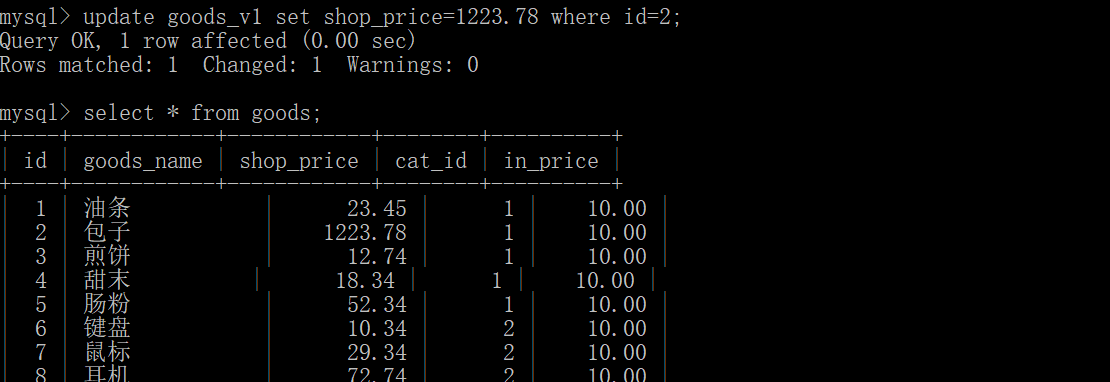


## 8、视图与表的关系

视图是表的查询结果，自然表的数据改变了，影响视图的结果。



（1）视图的数据与表的数据一一对应时，可以修改。

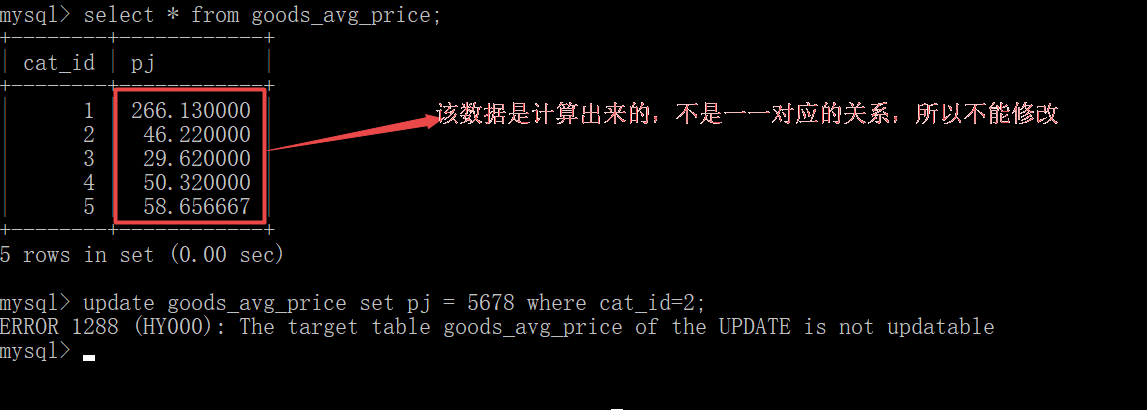


（2）视图增删该也会影响表，但是视图并不是总是能增删该的。

create view lmj as select cat\_id,max(shop\_price) as lmj from goods group by cat\_id;

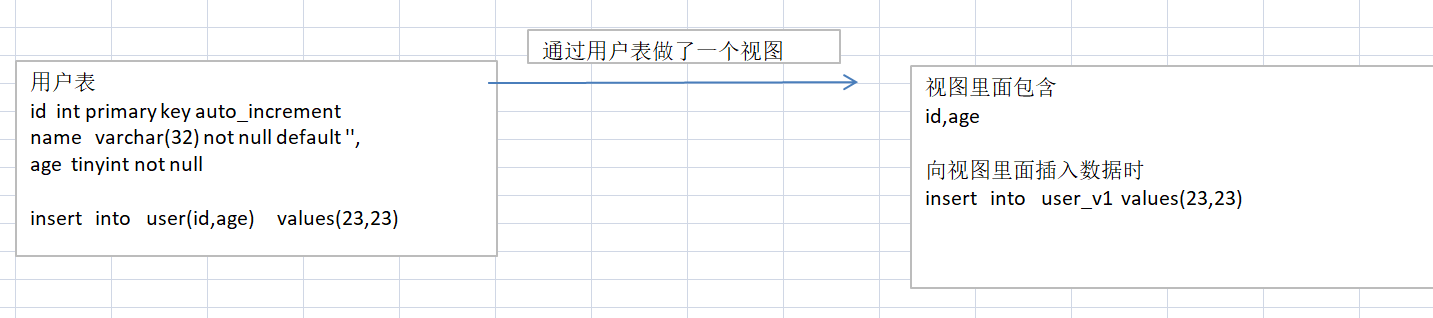
mysql> update lmj set lmj=1000 where cat\_id=4;

ERROR 1288 (HY000): The target table lmj of the UPDATE is not updatable



（3）对于视图insert还应注意，视图必须包含表中没有默认值的列。





注意：向视图里面插入数据时，视图必须包含表中没有默认值的列，才能插入成功，否则就插入失败。

**注意：在实际的开发中，不要对视图进行增删改。**

# 七、SQL 编程

## 1、变量声明

### （1）会话变量

定义形式：

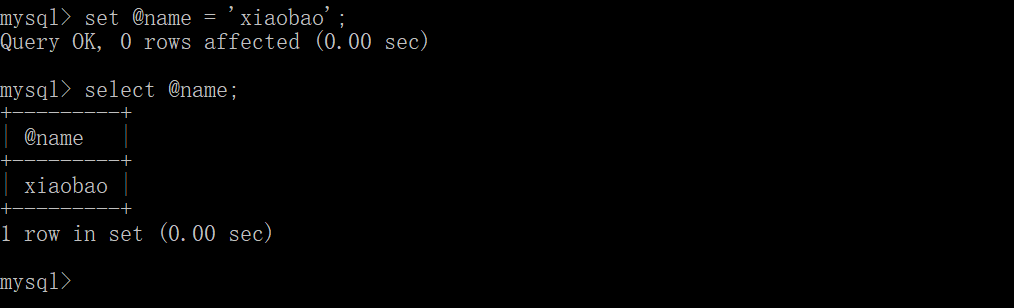
set @变量名 = 值；

说明：

1，跟php类似，第一次给其赋值，就算定义了

2，它可以在编程环境和非编程环境中使用！

3，使用的任何场合也都带该“@”符号。



### （2）普通变量

定义形式：

declare 变量名 类型 【default 默认值】；

说明：

1、它必须先声明（即定义），此时也可以赋值；

2、赋值跟会话变量一样： set 变量名 = 值；

3、它只能在编程环境中使用！！！

说明：什么是编程环境？

**编程环境是指 （1）存储过程 （2）函数 （3）触发器**。

### （3）变量赋值形式

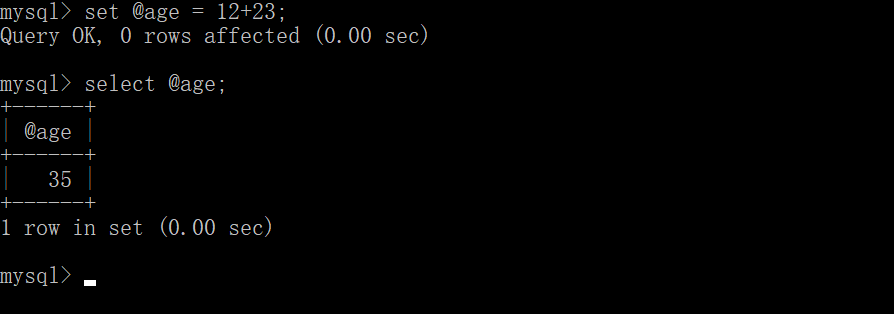
语法1：

set 变量名 = 表达式;#此语法中的变量必须先使用declare声明，在编程环境中使用

语法2：

set @变量名=表达式；

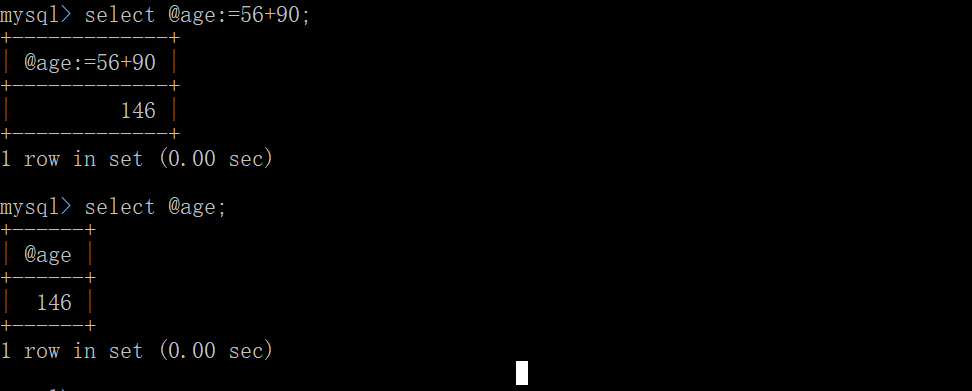
#此方式可以无需declare语法声明，而是直接赋值，类似php定义变量并赋值。



语法3：

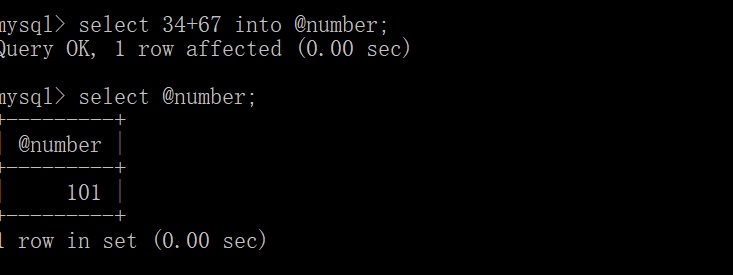
select @变量名:=表达式；

#此语句会给该变量赋值，同时还会作为一个select语句输出‘结果集’。



语法4：

select 表达式 into @变量名;#此语句虽然看起来是select语句，但其实并不输出‘结果集’，而是给变量赋值。



## 2、运算符

（1）算术运算符

+、-、\*、/、%

**注意：mysql没有++和—运算符**

（2）关系运算符

>、>=、<、<=、=（等于）、<>（不等于） !=（不等于）

（3）逻辑运算符

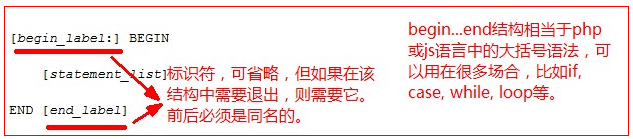
and（与）、or（或）、not（非）

## 3、语句块包含符

所谓语句块包含符，在js或php中，以及绝大部分的其他语言中，都是大括号：{}

它用在很多场合：if， switch, for, function

而mysql编程中的语句块包含符是begin end结构。



## 4、if判断

MySQL支持两种判断，第一个是if判断，第二个 case判断

**if语法**

单分支

if 条件 then

//代码

end if;

双分支

if 条件 then

代码1

else

代码2

end if;

多分支

if 条件 then

代码1

elseif 条件 then

代码2

else

代码3

end if;

通过存储过程 来体验 if语句的结构 ，

创建存储过程语法：

create procedure 存储过程名(参数1,参数2,…)

begin

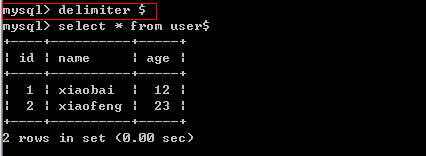
//代码;

end

案例：接收4个数字，

如果输入1则输出春天，2=》夏天 3=》秋天 4 =》冬天 其他数字=》出错

**注意：通常情况下，“；“表示SQL语句结束，同时向服务器提交并执行。但是存储过程中有很多SQL语句，每一句都要以分号隔开，这时候我们就需要使用其他符号来代替向服务器提交的命令。通过delimiter命令更改语句结束符。**



create procedure p1(num int)

begin

if num=1 then

select '春天' as '季节';

elseif num=2 then

select '夏天' as '季节';

elseif num=3 then

select '秋天' as '季节';

elseif num=4 then

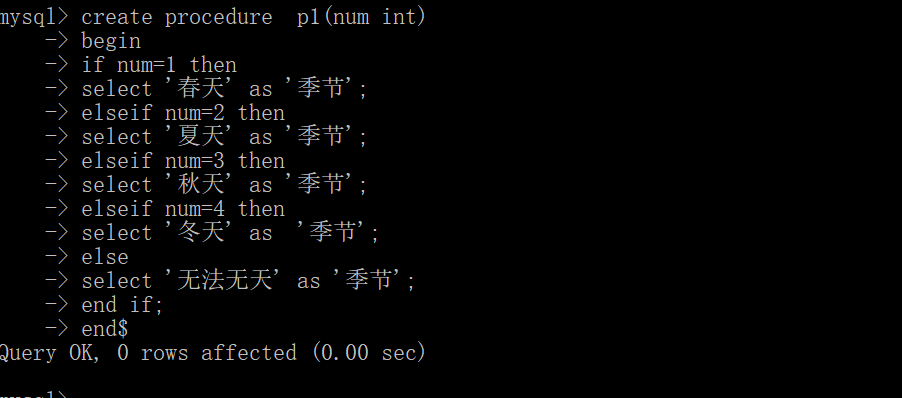
select '冬天' as '季节';

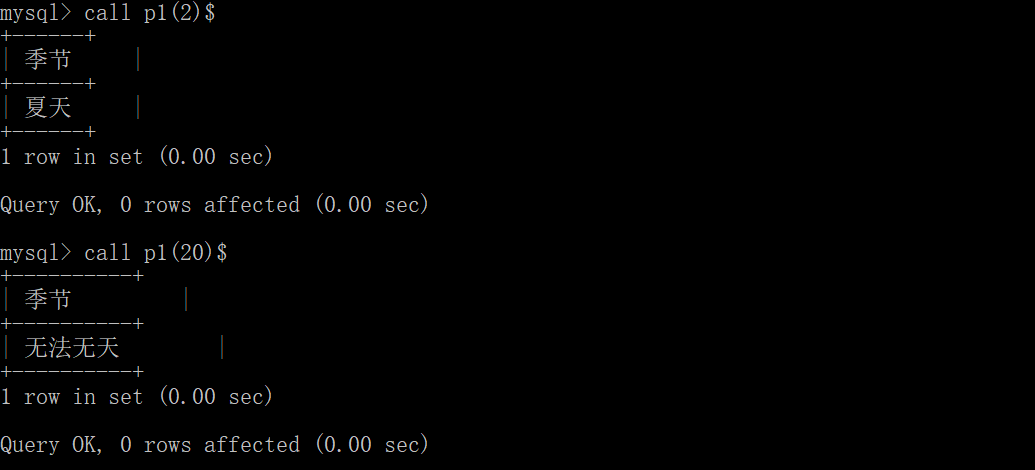
else

select '无法无天' as '季节';

end if;

end$





## 5、case判断

语法：

case 变量

when 值 then 语句;

when 值 then 语句;

else 语句;

end case ;

案例：接收4个数字，

如果输入1则输出春天，2=》夏天 3=》秋天 4 =》冬天 其他数字=》出错

create procedure p2(num int)

begin

case num

when 1 then select '春天' as '季节';

when 2 then select '夏天' as '季节';

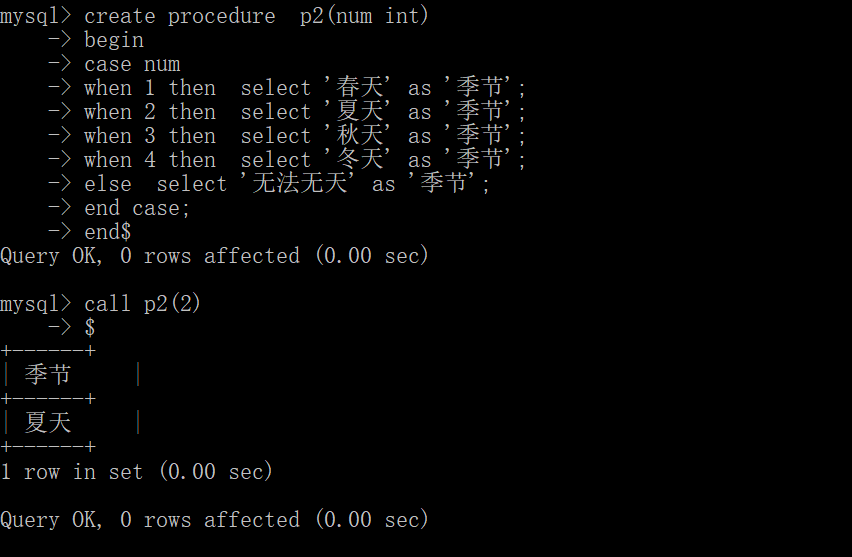
when 3 then select '秋天' as '季节';

when 4 then select '冬天' as '季节';

else select '无法无天' as '季节';

end case;

end$



## 6、循环

MySQL支持的循环有loop、while、repeat循环

（1）loop循环

语法：

标签名:loop

leave 标签名 --退出循环

end loop;

#案例：创建一个存储过程，完成计算1到n的和。

create procedure p3(n int)

begin

declare i int default 1;

declare s int default 0;

aa:loop

set s=s+i;

set i=i+1;

**if i>n then**

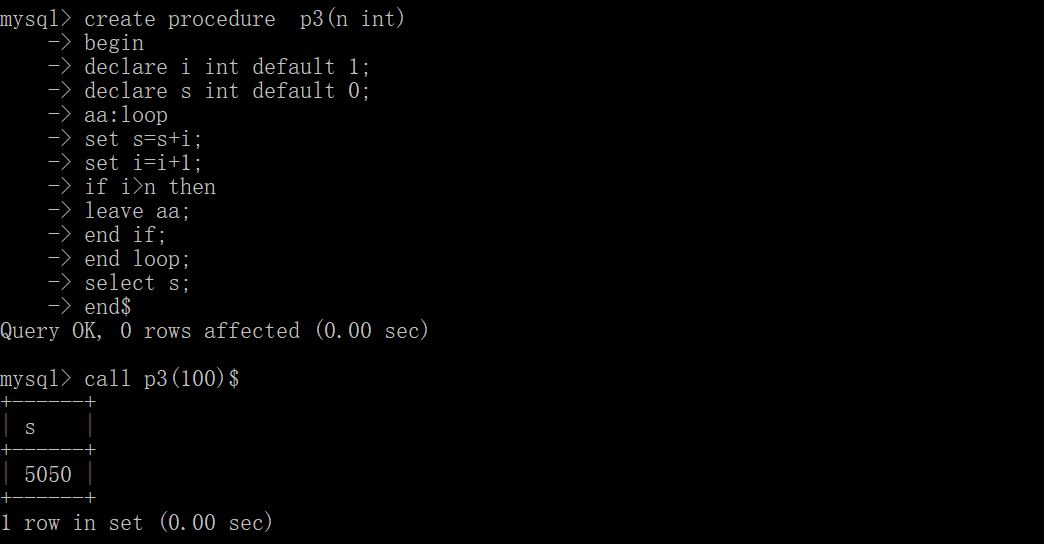
**leave aa;**

**end if;**

end loop;

select s;

end$



（2）while循环

[标签:]while 条件 do

//代码

end while;

#案例：创建一个存储过程，完成计算1到n的和。

create procedure p4(n int)

begin

declare i int default 1;

declare s int default 0;

while i<=n do

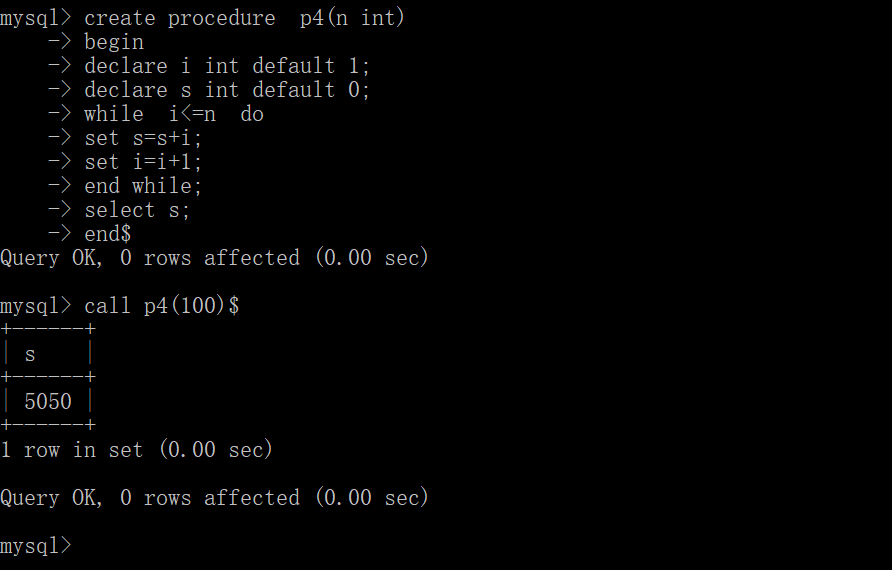
set s=s+i;

set i=i+1;

end while;

select s;

end$



# 八、存储过程

## 1、概念

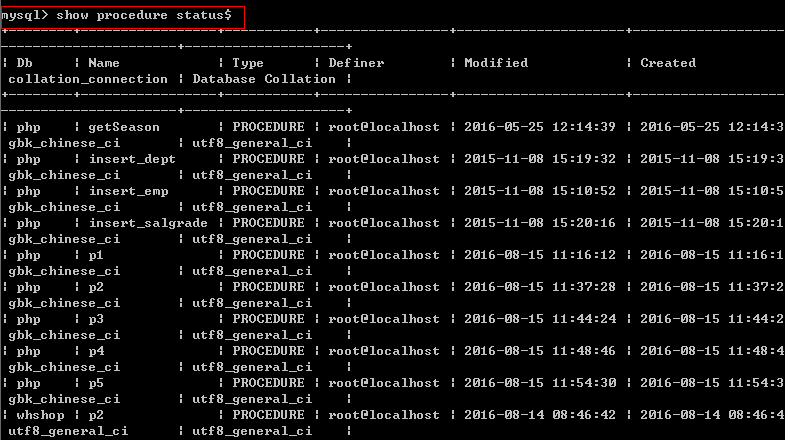
存储过程(procedure)

概念类似于函数，就是把一段代码封装起来，当要执行这一段代码的时候，可以通过调用该存储过程来实现。在封装的语句体里面，可以同if/else ,case,while等控制结构。

可以进行sql编程。

查看现有的存储过程。

show procedure status



## 2、存储过程的优点

存储过程（Stored Procedure）是在大型[数据库系统](http://baike.baidu.com/view/7809.htm)中，一组为了完成特定功能的SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象，任何一个设计良好的数据库应用程序都应该用到存储过程。

（1）存储过程只在创造时进行编译，以后每次执行存储过程都不需再重新编译，而一般SQL语句每执行一次就编译一次,所以使用存储过程可**提高数据库执行速度**。

（2）当对数据库进行复杂操作时(**如对多个表进行Update,Insert,Query,Delete时**)，可将此复杂操作用存储过程**封装起来**与数据库提供的事务处理结合一起使用。

（3）存储过程可以**重复使用**,可减少数据库开发人员的工作量

（4）安全性高,可设定只有某些用户才具有对指定存储过程的使用权

## 3、创建存储过程

语法：

create procedure 存储过程名(参数1,参数2,…)

begin

//代码

end

参数的类型：

**in（输入参数）： 表示该形参只能接受实参的数据——这是默认值，不写就是in；**

out（输出参数）：表示该形参其实是用于将内部的数据“传出”到外部给实参；

inout（输入输出参数）：具有上述2个功能。

案例1：查询一个表里面某些语句

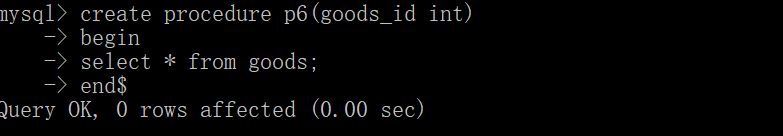
create procedure p6(goods\_id int)

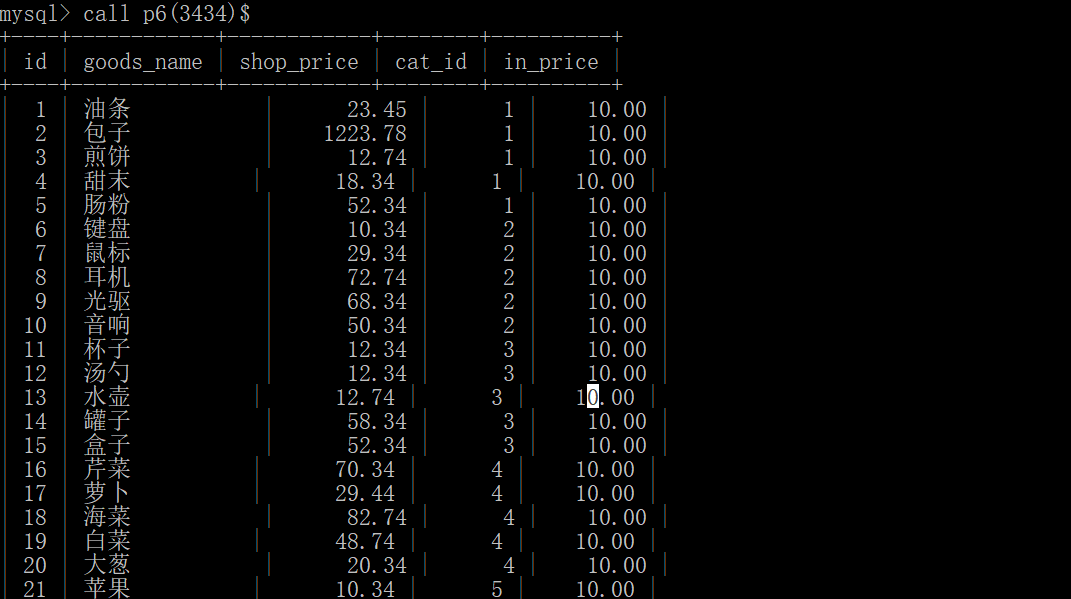
begin

select \* from goods;

end$

call p6()





案例2：第二个存储过程体会参数，使用参数

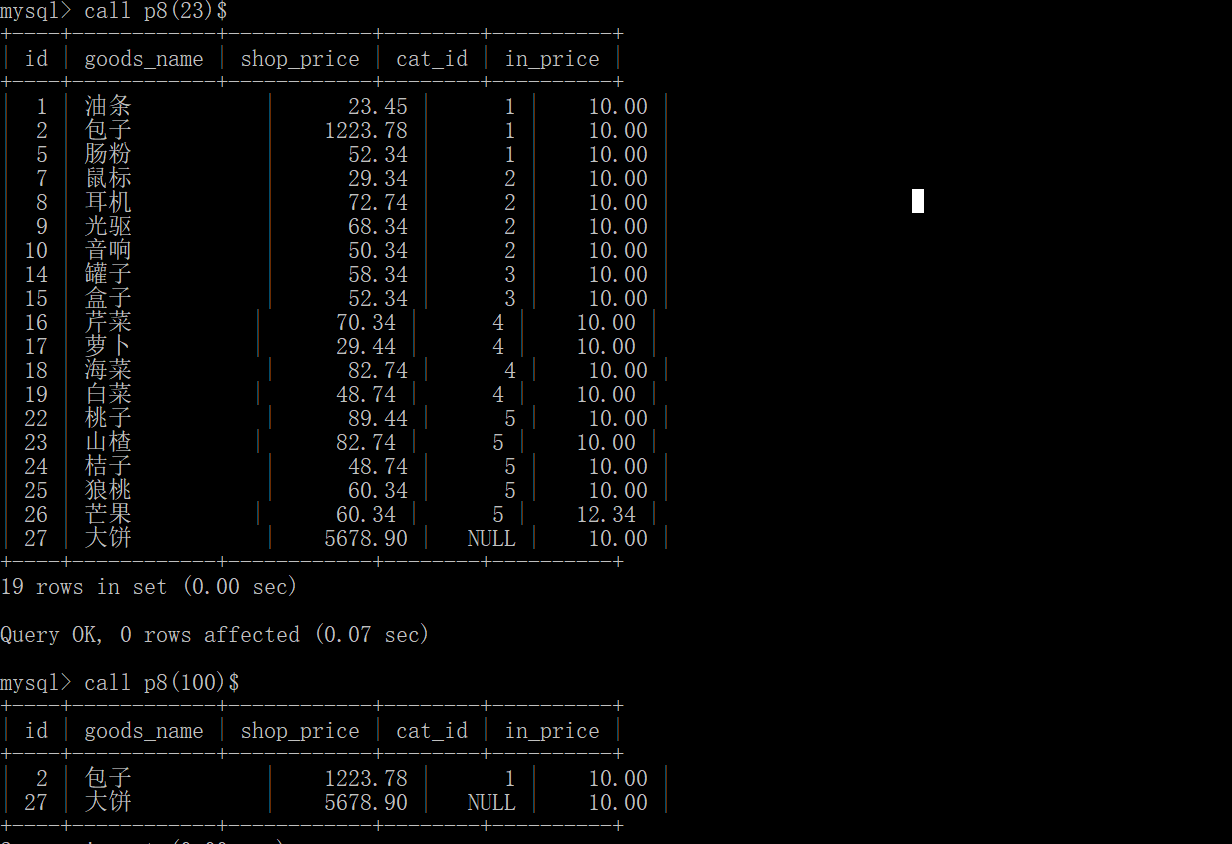
比如我们取出某个id的数据

create procedure p8(price float)

begin

select \* from goods where shop\_price>price;

end$



说明：

（1）存储过程中，可有各种编程元素：变量，流程控制，函数调用；

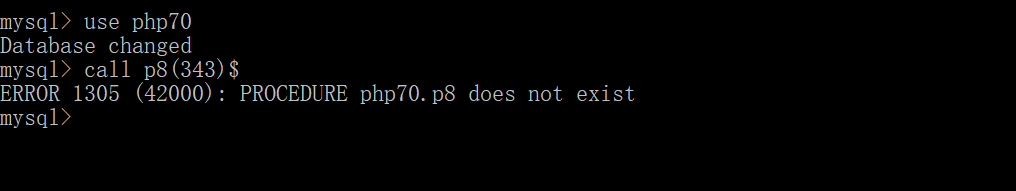
（2）还可以有：增删改查等各种mysql语句；

（3）其中select（或show，或desc）会作为存储过程执行后的“结果集”返回；

（4）形参可以设定数据的“进出方向”：

**（5）存储过程是属于数据库，在哪个数据库里面定义的，就在哪个数据库里面调用。**

**如下图，在别的数据库里面调用其他数据库里面定义的存储过程时，会报如下提示。**



## 4、调用存储过程

语法：

call 存储过程名称(参数)

在php里面如何调用，

mysql\_query(‘call p7(5)’);