# mysql数据库

**第一天**

永久性保存数据的仓库。

php的变量，在php脚本执行周期，临时性保存变量的概念！

## 什么是关系型

所谓关系型数据库，基于关系模型建立的数据！

关系模型：

mysql数据是关系型数据库！

何谓关系型（关系模型），利用关系(二维表)，去描述实体信息，与实体之间的联系的数据库架构就是关系型数据！

所谓关系：二维表！

学生信息

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 名字 | 年龄 | 所属班级 |
| Itcast\_007 | 王翦 | 66 | Java1011 |
| Itcast\_010 | 李白 | 44 | Php1016 |
| Itcast\_001 | 杜十娘 | 33 | Ios1021 |
| Itcast\_123 | 喜羊羊 | 11 | Java1011 |

班级信息

|  |  |  |
| --- | --- | --- |
| 班级名称 | 教室号 |  |
| Php1016 | 102 |  |
| Java1011 | 201 |  |
| .net0918 | 108 |  |
| Ios1021 | 218 |  |

### 其他的类型的数据介绍

nosql：not only sql，非关系型数据库

sql：一门语言，结构化查询语言，操作关系型数据的语言！

典型的是 key / value型，键值对型

‘abcedefakasdnfakjsdi’=>{‘itcast\_007’, 王翦, 66, java1011}

‘asdfkndikciuehaalcidk’=>{‘itcast\_001’, 杜十娘, 33, ios1021, ‘female’, {‘珠宝’, ‘百宝箱’,’投河’}}

### 关系型数据的基本概念

关系，二维表

行，记录，一行就是一条记录。

列，字段，一列就是一个字段。

SQL：结构化查询语言，操作关系型数据库的语言！

## 操作MYSQL基本流程

什么数据库系统，mysql 就是数据库系统！

数据库系统最基本应该由：

数据系统 = 数据库（数据主体部分） + 数据管理系统（操作数据的工具）

DBS（DataBase System） = DB(Database) + DBMS(Database Managenemt System)



mysql的操作，是基于 C/S 的！

Client / Server，客户端/服务器.

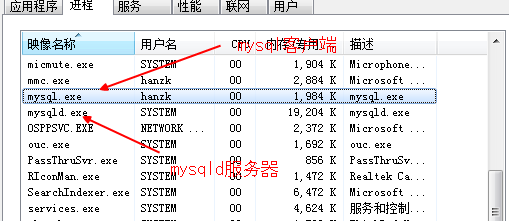
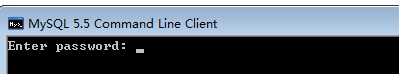
c/s指的是不同的服务器，提供的是不同的终端访问方式！

操作mysql，就一定：通过操作mysql客户端，向mysql服务器发出指令，从而完成操作！



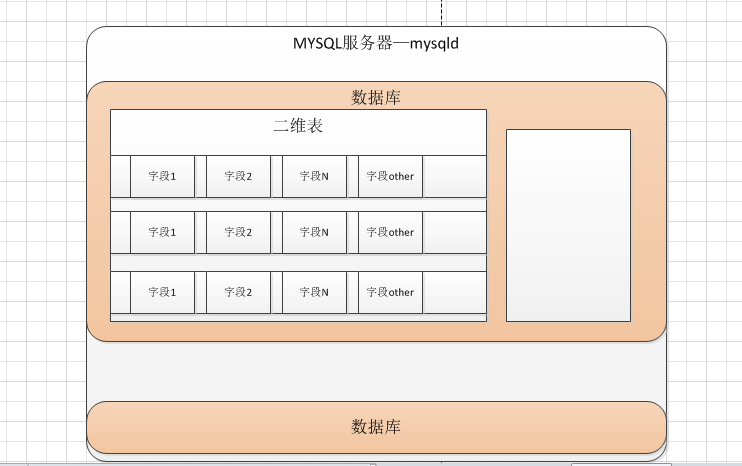
安装mysql时，自动携带一个命令行的客户端！mysql。

利用该客户端，向服务器发送指令，然后等待执行结果即可！



任何操作mysql服务器行为的行为，都是mysql客户端发出的！

## msyql服务器的基本结构体系



数据库服务器 ->库 ->表 ->字段（数据）

完成数据的操作，先建立数据的结构（由库到表再到字段）

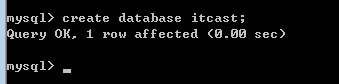
SQL：大体分成典型：DML（数据管理语句，数据操作），DDL（数据定义语句，数据结构的控制语句，表操作和库操作）

# 库操作，DDL

## 创建，create

（create，几乎所有的结构都是用该语法完成）

create database 库名 [库选项]



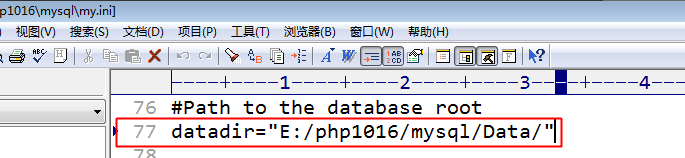
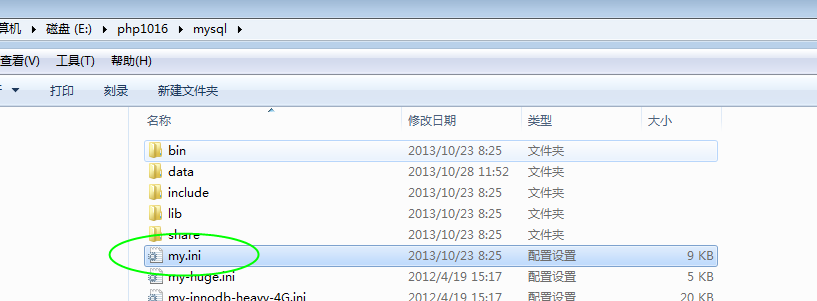
注意的问题：

库选项，只有字符集，校对集的概念！

每个库，会对应一个数据目录

存放在当前mysql的总的数据库目录内

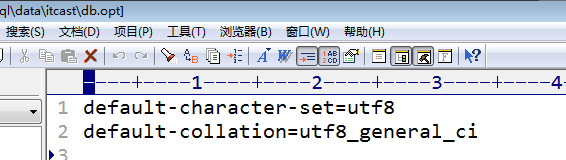
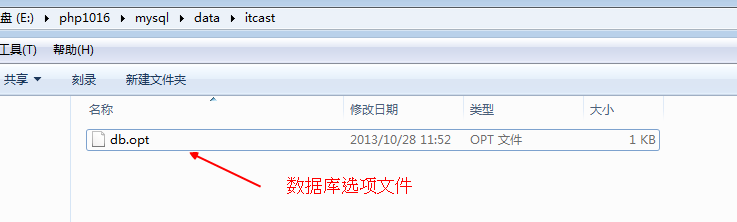
参考 mysql的配置文件得到该目录：



看到刚刚的数据库目录：



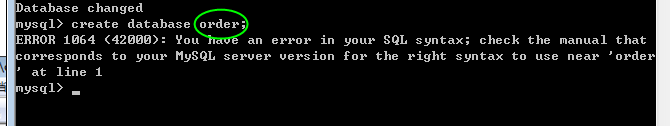
看看目录内，没有表，没有其他文件，除了一个数据库选项文件：



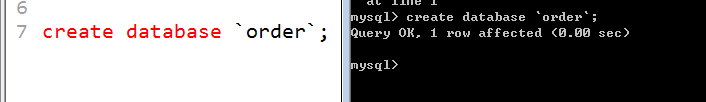
数据库名的问题

大小写问题，取决于mysql服务器，所在的操作系统！（建议是，认为区分）

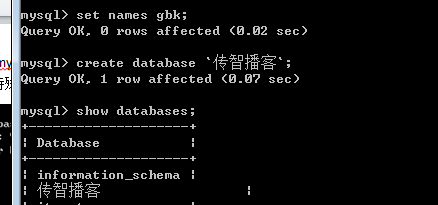
特殊名称，关键字，特殊字符等！默认是不可以的！



但是，可以使用反引号将名称包裹起来，告知服务器，此处一个名字，而不是特殊操作！



中文等都可以作为标识符（库名），需要同样反引号！（多字节字符，还需要注意字符集的问题）



## 查询（看看有哪些库，库的基本结构），show

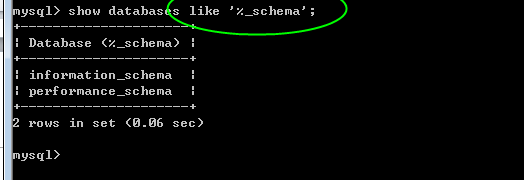
### 查询已有的数据库名字

show databases;



show databases likes;

like 关键字用于过滤多个数据库！



可以使用通配符（通用匹配符，可以匹配多个字符）

% 匹配任意字符的任意次数（包括0次）的组合！

\_ 匹配任意字符的一次！

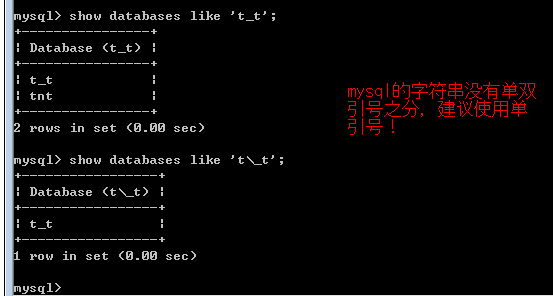
like ‘x\_y’;

x1y xby xxy（可以）

xy(不可以)

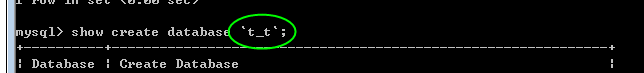
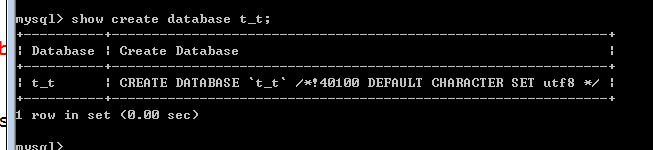
通配符是与 like 关键字一起使用！

注意如需要匹配特定的通配符，则需要对通配符转义，使用反斜杠\完成转义！



### 查看某个数据库的定义

show create database 库名；



## 修改，alter

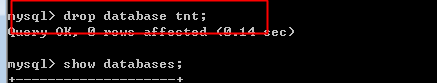
alter database 数据库名

只能修改数据库选项



## 删除，drop

drop database 名字



## if not exists, if exists

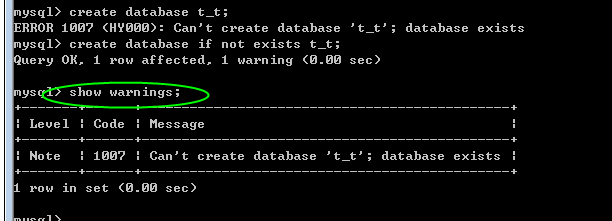
在 create 与 drop 时，创建和删除时，有两个额外的操作：

create database if not exists 库名

如果不存在则创建

drop database if exists 库名

如果存在，则删除



# 表操作，DDL

表本身，与表结构的操作！

## 创建，create table

create table 表名 (

字段的定义

) [表选项];

其中表名，一定先要确定数据库！因此一个典型的表名是由两部分组成：

所在库.表名

test.itcast test库内itcast表

itcast.stu itcast库内的stu表

但是我们可以设置默认数据库，如果不指定则使用默认数据库（当前数据库）

use 数据库名。选择默认数据库！

在使用表名但是没有指明其所在数据库时，默认数据库才会起作用！



在itcast库内创建：

use itcast ; create table stu;

或者

create table itcast.stu

其中字段的部分

字段才是最终的数据的载体（与变量的概念是类似的，都是基本保存数据的），mysql的是强类型，字段的类型是固定的，提前定义好的！

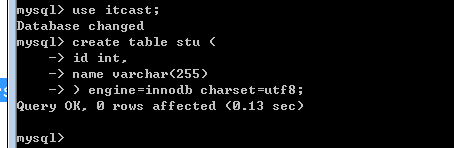
因此，在定义字段时，至少要字段名和字段类型！

两种最基本的mysql数据类型（int， varchar,varchar必须指定最大长度字符为单位）

表选项部分

典型的常用的表选项有：

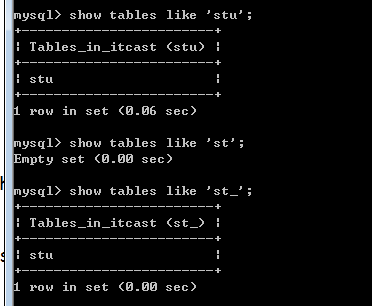
字符集（校对集），表引擎。



## 查看，show，desc

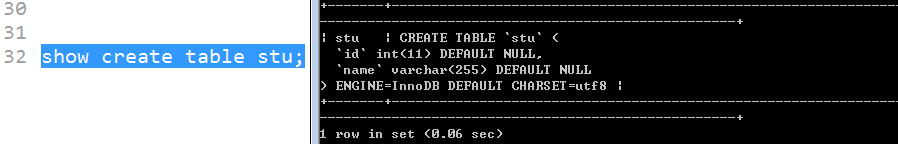
### 查看所有的表有哪些

show tables like ‘模式’



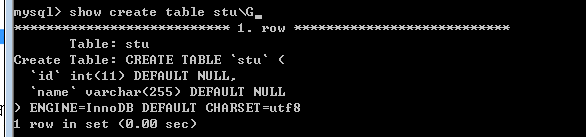
### 查看具体的建表语句

show create table table\_name



在mysql的命令行客户端，如果数据过多，不容易展示！

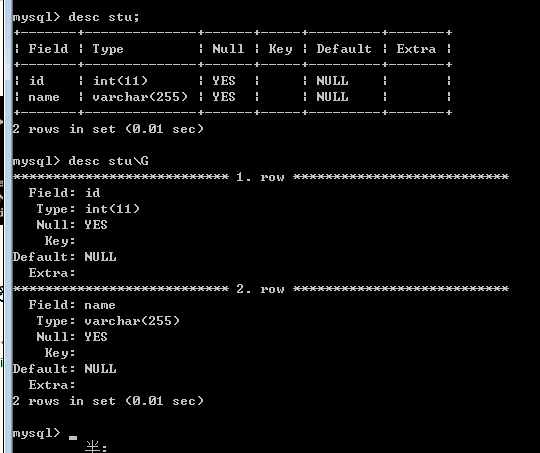
可以使用 \G 作为语句结束符！



### 查询表结构

desc 表名

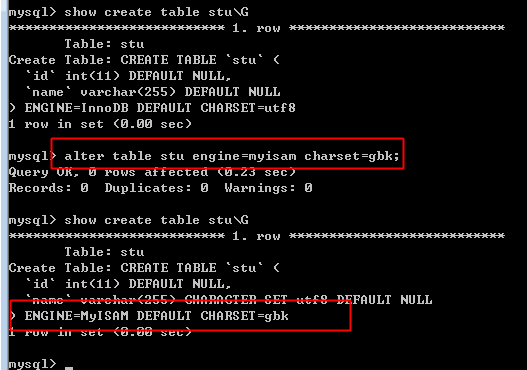
desc describe的简写



## 修改，alter table

### 修改表选项

alter table table\_name [新选项]



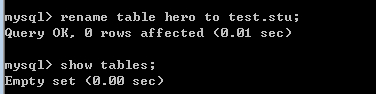
### 改表名

rename table原表名 to 新表名。



注意，表名可以由库名.表名形式的！

因此，可以跨库修改表名：只要在表名前增加库名即可

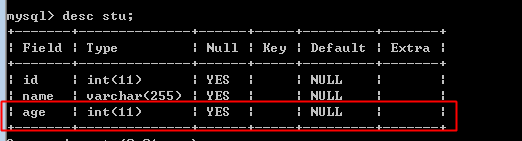


### 修改表结构，表内字段

#### 增加字段

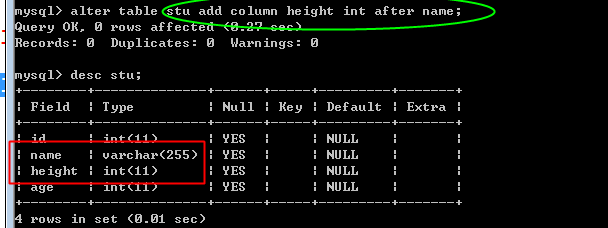
alter table table\_name add column 字段定义 [字段位置]

增加一个 age字段：



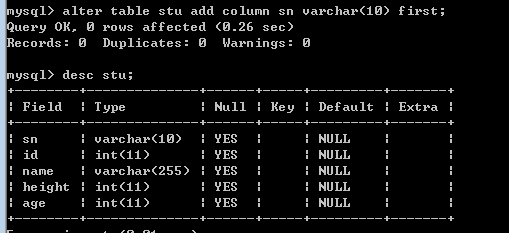
增加一个 height 在 name之后：

使用关键字 after some\_column\_name;



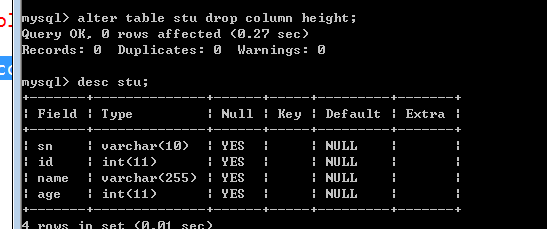
在最开始增加sn字段

使用关键字，first



#### 删除字段

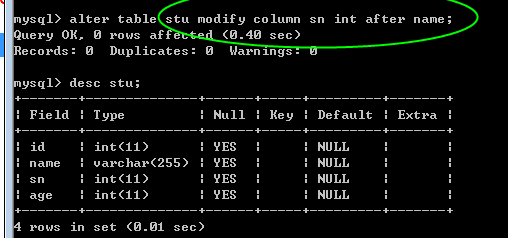
alter table table\_name drop column column\_name;



#### 修改已有字段（修改字段定义）

alter table table\_name modify column column\_name 新的定义！

可以修改位置



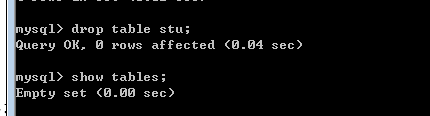
#### 字段改名

alter table table\_name change column 原字段名新字段名新字段定义！

注意，不是纯粹的改名，而是需要在修改定义的同时改名！



## 删除，drop table



drop table if exists

create table if not exists!

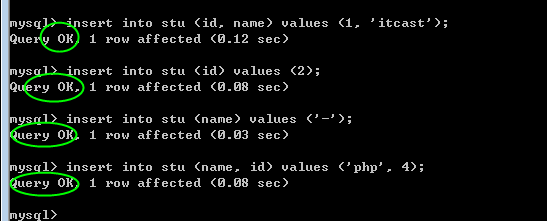
# 数据操作，DML

基本的操作

## 增加，创建，插入，insert

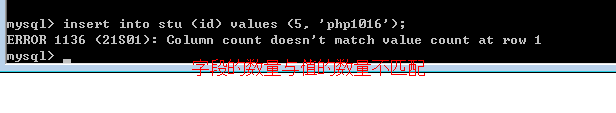
insert into 表名 (字段列表) values (与字段相对的值列表)

不一定要一次性插入所有字段，或者按照原始的字段顺序插入：



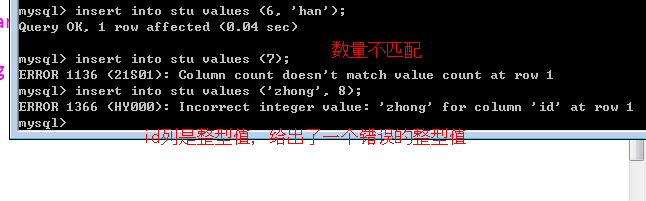
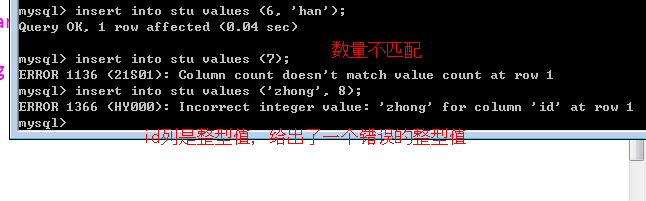
但是：

字段与值的数量一定要匹配：



特别的：

如果所有的值都按照字段的出现顺序都插入的话，可以省略字段列表部分！

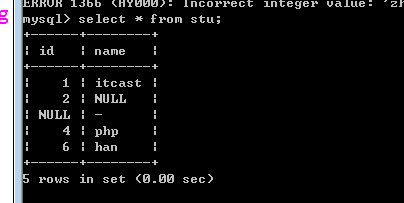


注意：数值类型，不需要增加引号！而字符串类型都需要出现引号内！（但是数值型，可以出现引号内）

## 查询，检索，选择，select

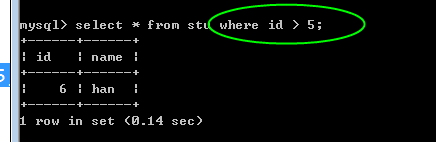
select 字段列表 from 表名 [where 条件表达式]

其中字段列表可以使用 \* 表示所有字段！



关于条件表达式，默认是没有，表示永远为真！

但是，很少出现没有条件的情况！

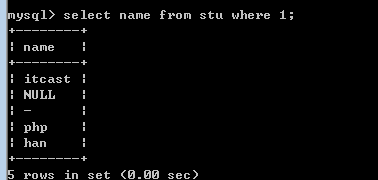


为了突出，应该所有的语句都有查询条件！即使没有条件，我也强制增加一个 where 1;



关于字段列表：

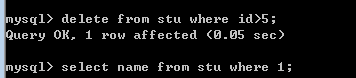
也应该够用就可以！



## 删除，delete

delete from 表名 where 条件;

关于条件，可以省略。表示永远为真。

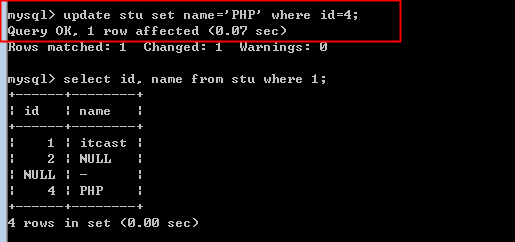


注意，删除是不可逆的。要避免没有条件的删除！

## 修改，更新，update

update 表名 set 字段=新值, 字段n=新值n where 条件

关于条件，可以省略。表示永远为真。



create，show，alter，drop database/talbe

add，modify，change，drop column

insert , select, update, delete table\_name

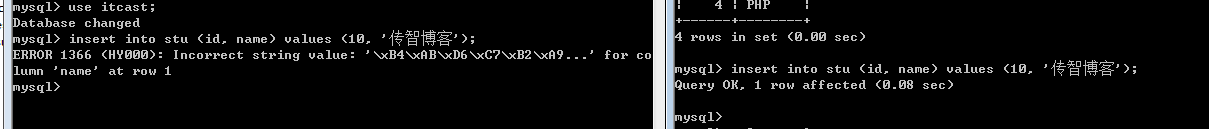
# 字符集

字符集，描述可见到的图形，在存储和传输时使用的编码称之为字符集！

指的是：

图形与编码之间的对应关系！

字 => 11100101 11100001 00010100



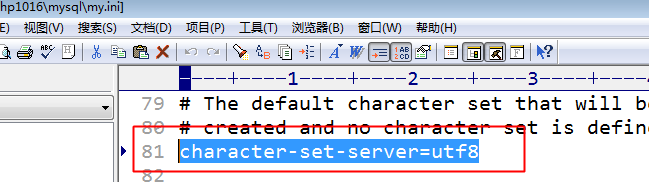
## 数据存储的编码

数据是存储在服务器端的！究竟是以哪种编码保存的，怎么决定？

数据是最终要映射（保存）到字段的层面上！因此决定数据的编码，也是最终由字段来决定！

确定数据的存储编码是由以下的方案完成的：

1. 先看字段的编码！如果存在则确定完毕！
2. 如果字段没有设置编码，查看表的编码，如果有，则确定数据编码！
3. 如果表没有编码，查看库的编码，如果有，则确定数据编码！
4. 如果库也没有编码，则有服务器程序确定！



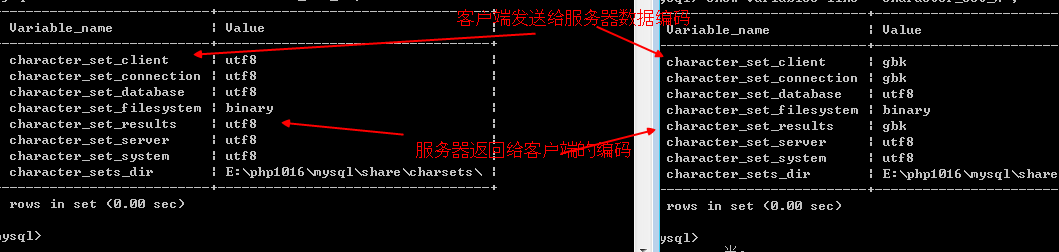
典型的编码：gbk，utf8!

## 客户端操作，查询的编码

在客户端与服务器端交互时，存在两个重要的编码：

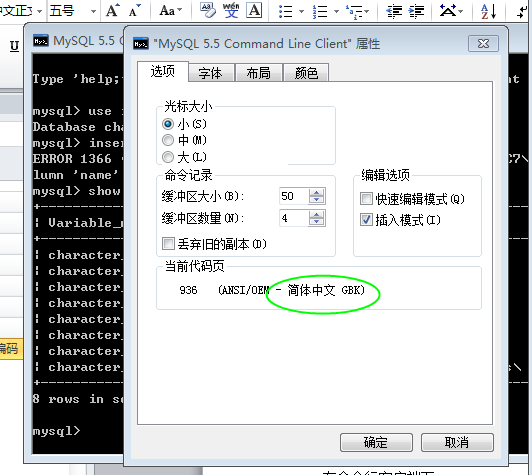
1. 客户端发送数据的编码
2. 客户端接收到服务器数据编码！

可以通过 show variables like ‘character\_set\_%’展示以character\_set开头变量，其中就有上面的两个值：

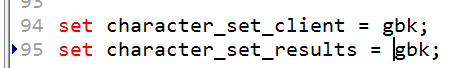


在命令行客户端下：

该客户端，只能是gbk编码！



使用 set 变量名 = 变量值的形式更改变量值！



set names gbk，就是一个快捷操作，将上面两个配置同时更改成目标编码！

set names gbk|utf8 取决于，客户端所能接受的编码！

整体流程：



set names gbk|utf8。

# PHP操作mysql服务器

php作为mysql服务器的客户端！

连接认证

发送sql

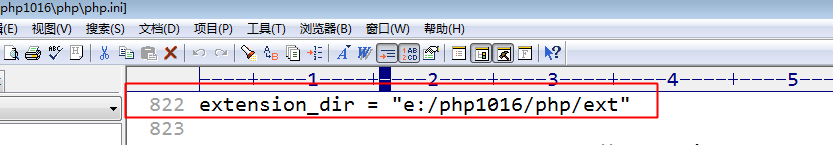
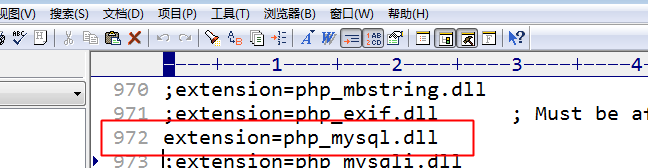
执行sql，生成结果（mysql-server）

处理结果

关闭连接

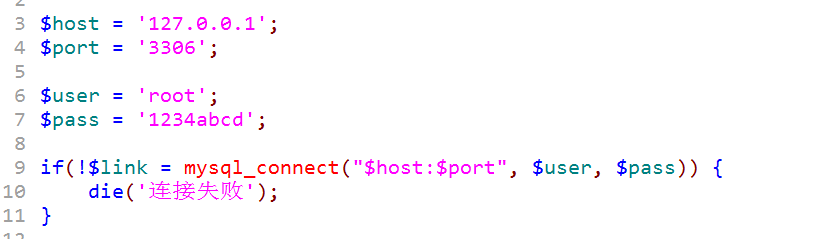
## 加载必要的扩展，使可以作为客户端使用

mysql\_connect();



## 连接认证

mysql\_connect()



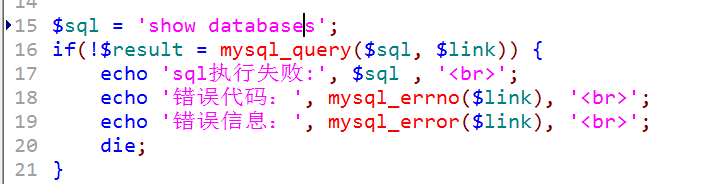
成功返回连接资源，失败 false！

## 向mysql服务器发送sql

mysql\_query(sql, 连接资源);

失败返回false，成功返回资源或者true！

可以使用 mysql \_error(连接) mysql\_errno(连接)获得错误信息和标识



## 处理返回数据

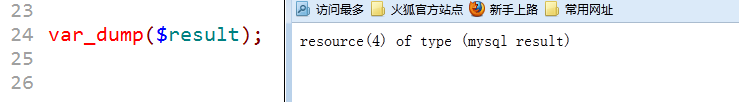
执行成功后：返回数据可以是资源也可以true。执行失败一定是false！

依据所执行的 sql，是否有返回数据！

返回资源：有返回数据：select，show，desc。

返回true：没有返回数据的： use，set，insert，update，delete，DDL

### 处理返回资源



称之为结果集（result set）类型资源！

结果集：结果的集合！

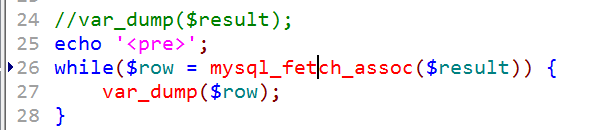
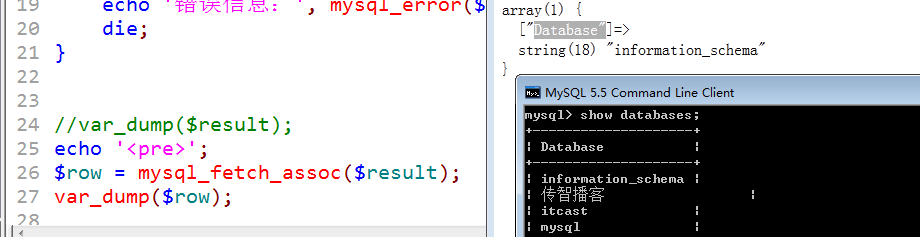
将数据，从结果集中取出来！称之为 fetch！

使用函数：

mysql\_fetch\_assoc|row|array。功能完全一致，只是返回的数据格式不同！

在结果集中，取得一条记录。结果集内也存在结果集记录指针的概念！

fetch一次，只能取得当前记录，但是可以向后移动记录指针！配合上循环结构可以将所有的记录从结果集中取出！



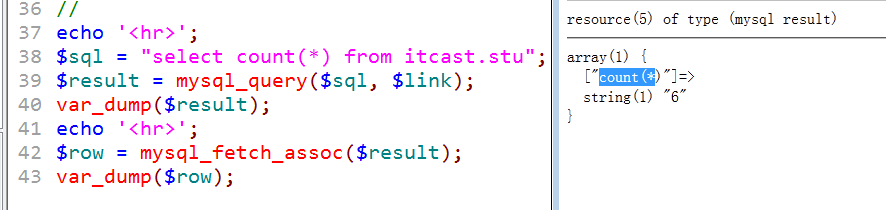
特别注意：

任何有结果的sql操作，返回的都是结果集！

结果集，就是一个二维表的结构！是一行行的记录组成！

即使，结果集中只有一条记录

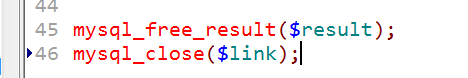
甚至，我们只需要返回一条数据！



## 释放资源

mysql\_free\_result(结果集)

mysql\_close(连接资源);



# 预习

校对集

列类型（数据类型）

列选项（列属性，列约束）

设计模式（范式，关联）

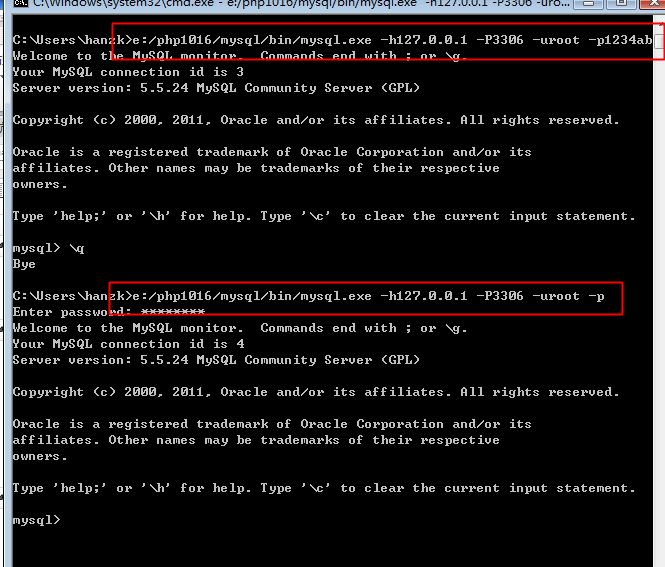
# 作业

1，模拟查找所有的数据表结构

2，编码问题！

**第二天**

连接的必要四个参数：



客户端发送一条sql：

客户端编码（character\_set\_client）->连接层编码（characger\_set\_connection）->服务器内部编码（server\_internal）

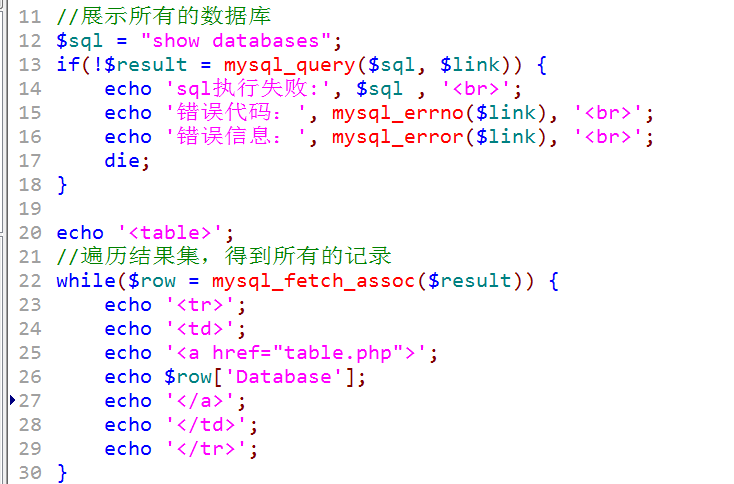
服务器端发送结果：

服务器内部编码（server\_internal）->连接层编码（characger\_set\_connection）->客户端接收的结果编码（character\_set\_results）

总体的编码问题：



先获得所有的数据库：



为其增加链接，请求到table.php展示所有的表，应该以 GET方式（在url上传递）形式将库名，传递到table.php

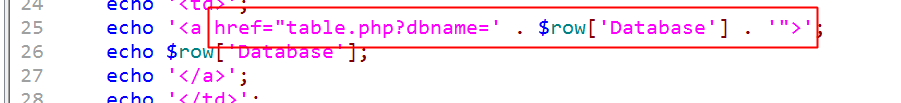
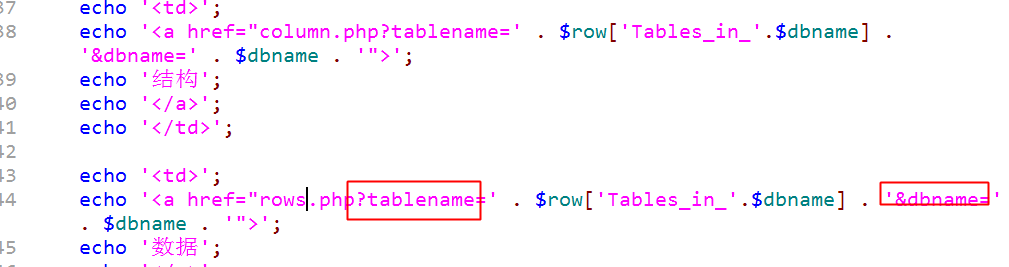


table.php

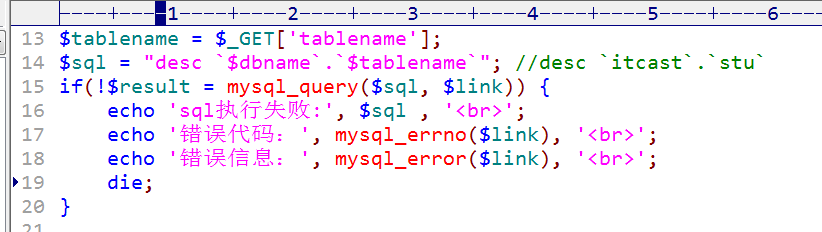
先获得表名列表，再为其增加指向结构和数据的连接！



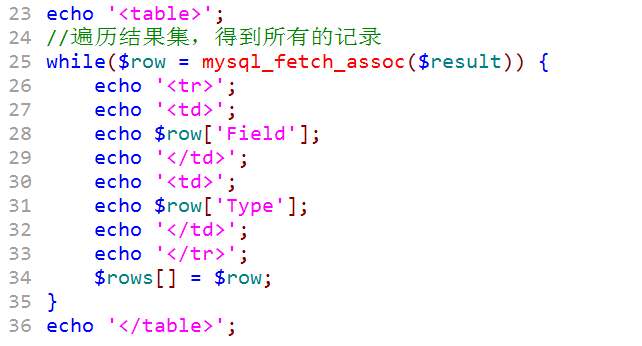
注意，在获得表结构与数据时，至少需要库名和表名两个参数！



column.php获得结构



展示：



rows.php

展示列表时，先展示字段名，再展示数据！

mysql\_num\_rows($result)获得结果集中的记录数量

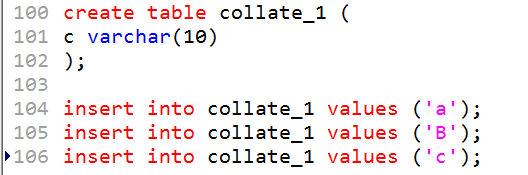
# 校对集，collate

指的是字符之间的比较关系！

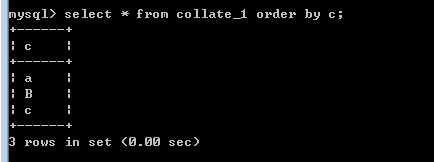
a B c

or？

B a c



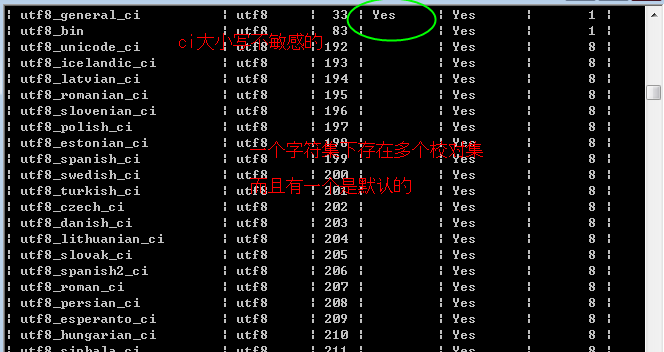
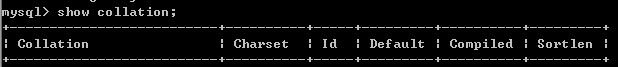
此时，使用 order by对结果排序，看结果：



顺序为 a-B-c 忽略了大小写！

可以被校对集改变：

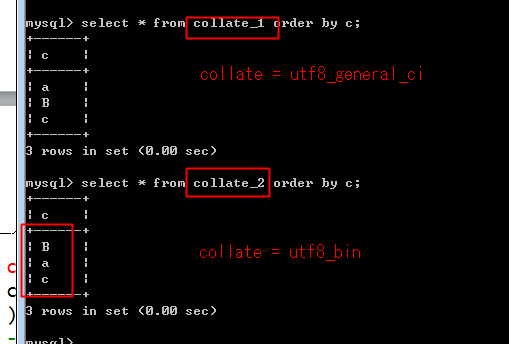
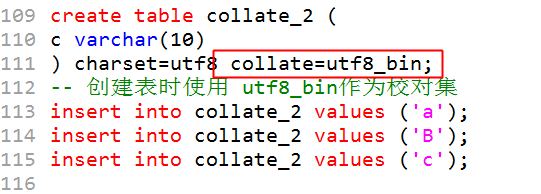
利用 show collation; 查看到所有的校对集！



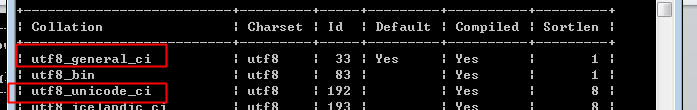
校对集，依赖于字符集！

校对集，指的是，在某个字符集下，字符的排序关系应该是什么，称之为校对集！

再创建一个 utt8\_bin的校对集表，在排序：



我们典型的选择：



典型的后缀：

\_bin 二进制编码层面直接比较：

\_ci 忽略大小写（大小写不敏感）比较

\_cs 大小写敏感比较

# mysql的数据类型，列类型，字段类型

三大类：

数值，字符串，日期时间

## 数值

### 整数，int

int 4个字节。

还有占用其他空间的整型：

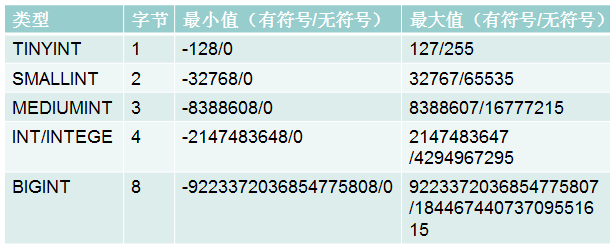
tinyint 1

smallint 2

mediumint 3

int 4

bigint 8



如何选择：

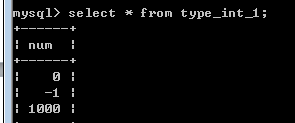
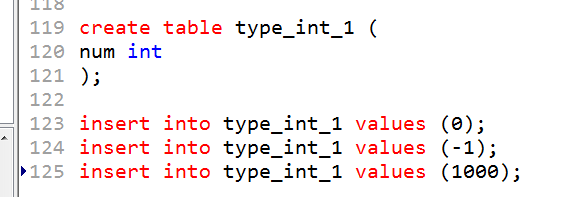
通过业务逻辑判断！常见的是 tinyint，int！

mysql的整型，有php整型不具备的概念：

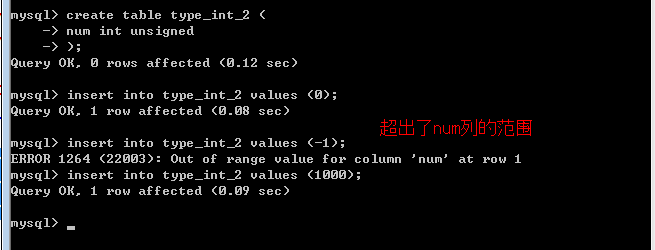
one，无符号

只能表示整数或0。那么最大的整数会很大！

默认是有符号！可通过整型的 unsigned选项，int unsigned 设置整型无符号

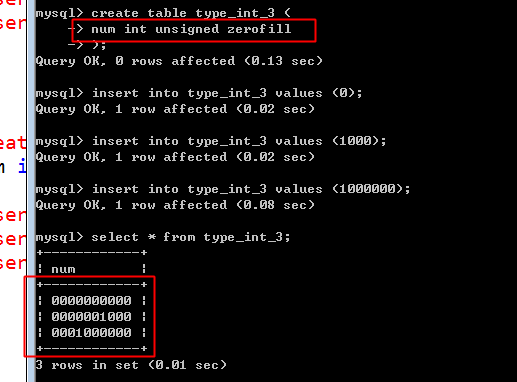


无符号的：



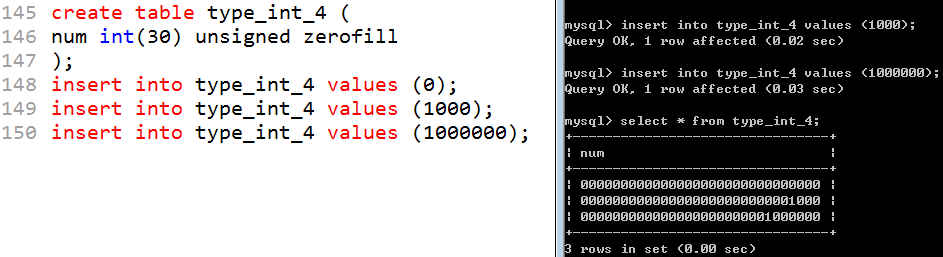
two，显示宽度

显示宽度，不决定整型的范围。而决定在显示出该数之后，如果宽度不够，则采用前导零不齐！此时需要额外的属性 zerofill 来设置！



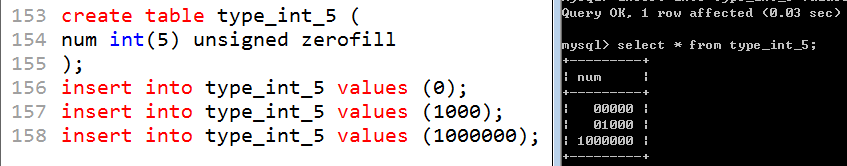
可见无符号，默认的显示宽度是 10，因此全都使用0来补齐。

可以设置整型的显示宽度！在 int(M)即可！



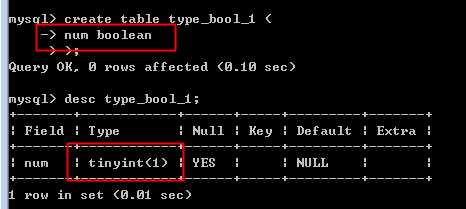
255个宽度就可以了！

注意，如果超出了宽度范围，则直接显示！



额外的，mysql中，没有布尔型！

但是存在Boolean这个关键字，表示 tinyint(1)

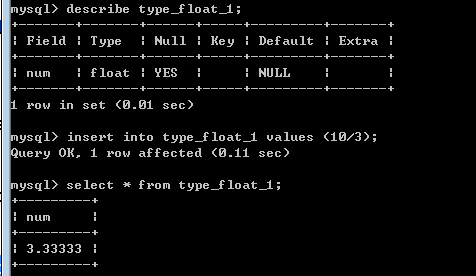


典型的，0表示假，1表示真！

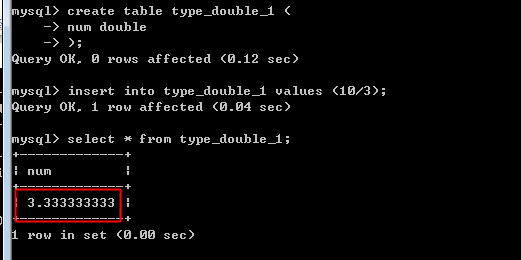
### 浮点数，float，double

单精度，float，4个字节

双精度，double，8个字节



双精度：



在定义浮点数，典型的需要指明其有效位数，和小数位数

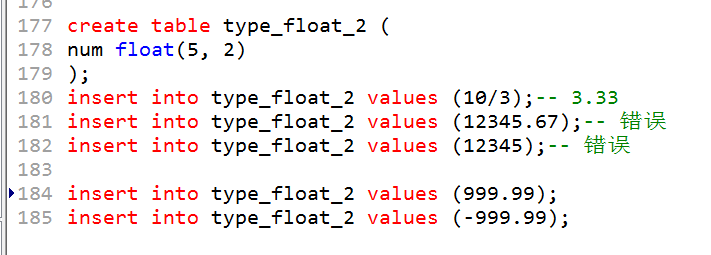
float(M,D)

double(M,D)

M：所有的有效位数

D：其中的小数位数

以上两个值，决定一个浮点数的有效范围！

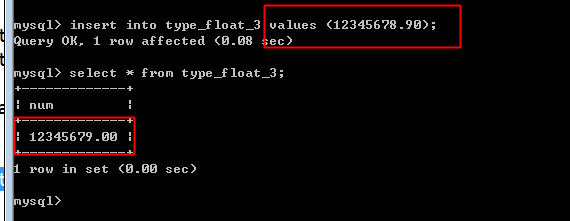


典型的浮点数

float(10, 2);

99999999.99

注意，浮点数，近似值，不是精确值！如果一个数，很大，接近最大值，可能出错！



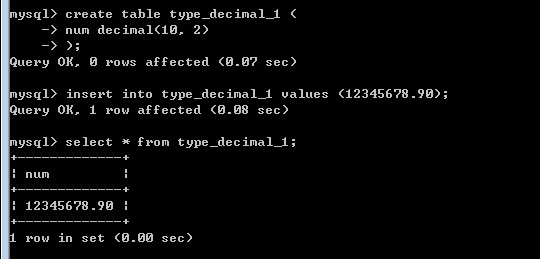
一旦出现精确数据（小数）需要保存可以使用，下面的定点数

### 定点数，decimal

小数点是固定的！

decmal，与定义浮点数一致，也有有效位数与小数位数的概念：

decimal(M,D)

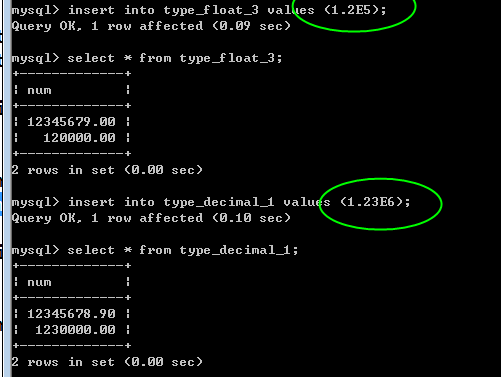


不是数的形式存储，类似于字符串的形式！



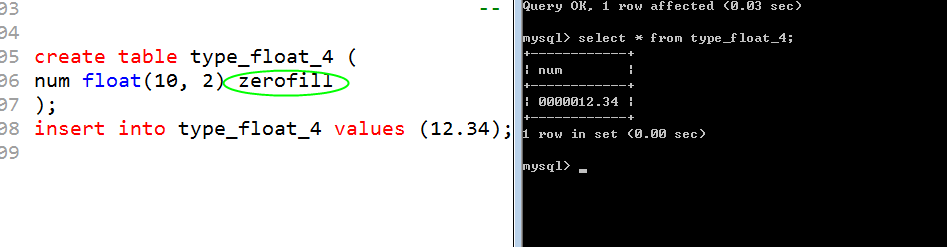
注意，关于小数：

1,支持科学计数法



2，同样支持 unsigned，无符号！

3，同样支持 zerofill，



TIP：小数的M，即表示有效范围，也表示显示宽度！（而整数的M只表示显示宽度）

# 字符串类型

## char

最基本（定长）的字符串类型！

用于保存，长度固定的内容！

速度快，但是保存变长数据，会浪费空间！

## varchar

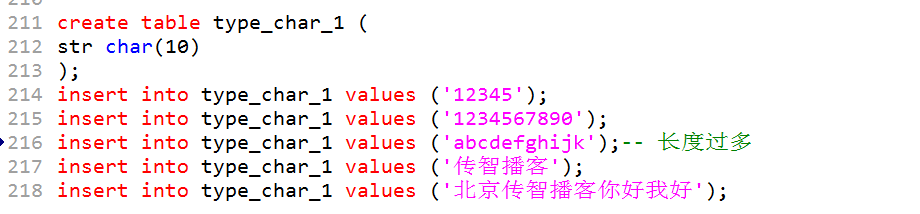
可变长度的字符串！

用于保存长度可变的数据！

保存变长数据时，节省空间，处理起来麻烦些！

char(L)，varchar(L)

L,表示每个数据的最大长度！单位是字符数（不是字节数）



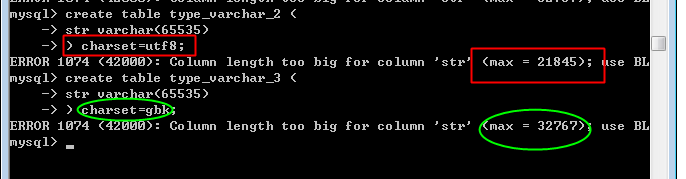
L 表示最大长度！

L的值不是多大都可以！

有最大值：

1，mysql一条记录最大不能超过65535个字节！（字段的长度加起来，不能超过这个值）

2，长度单位是字符数，与编码是相关的！

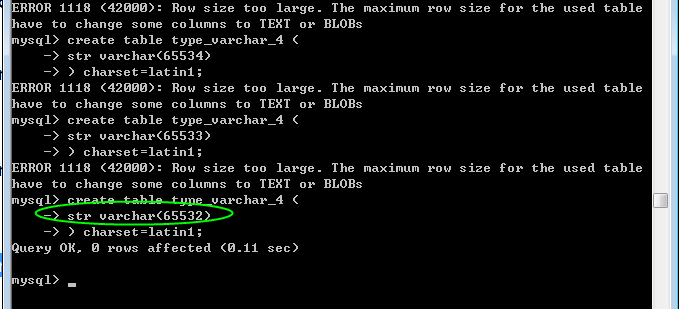


采用单字节编码测试下：

Latin1



测试发现只有一个字段，而且是单字节字符集，还是不能是65535，原因是？



为什么是 65532呢，少了三个字节：

3，由于varchar是变长，需要记录下真实的数据到底有多长！每个varchar类型的数据，还需要额外的1个或2个字节保存真实数据的长度！（取决于真实数据的长度）

4，整条记录，还需要一个字节来保存那些字段为null

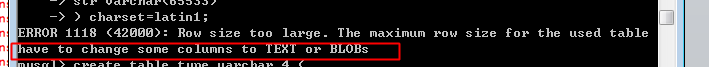


但是，在使用时，varchar 超过255就选择 text来保存！

## text

文本，不限长度的字符串！

该字段，不需要指定长度，而且也不会算入到记录的总长度内！



选择

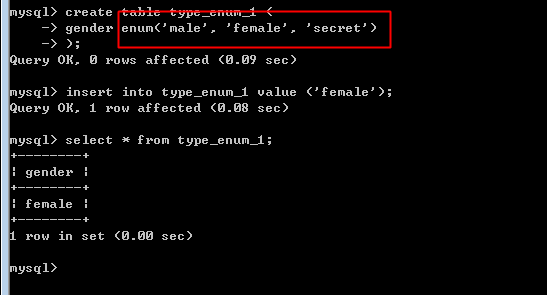
定长，char

变长较短（255）个字符之内，varchar

变长，较长，text

## 枚举，enum，单选

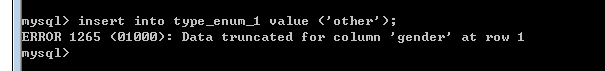
需要在定义枚举类型时，列出哪些是可能的！



在处理时，类似字符串型进行操作！

意义在于：

1. 限定值的可能性！



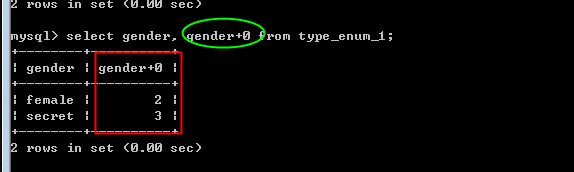
1. 速度快，比普通的字符串速度快！

原因是枚举型是利用整数进行管理的，能够2个字节进行管理！



每个值，都是一个整数标识，从第一个选项开始为1，逐一递增！

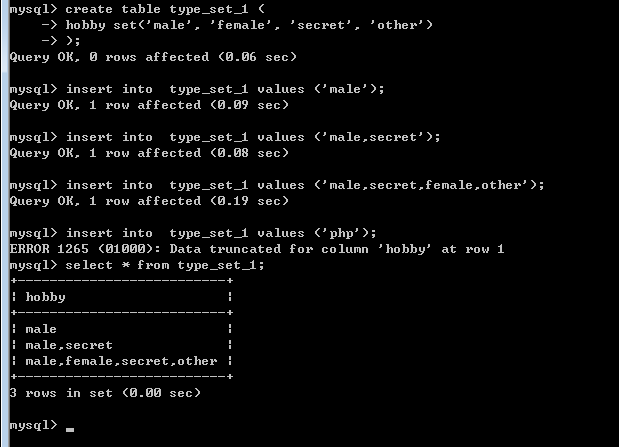
管理时整数的形式，速度比字符串快！



2 个字节，0-65535，因此可以有 65535个选项可以使用！

## 集合，set，复选

类似于 enum枚举，在定义时，也需要指定其已有值！



与字符串相比，优势是：

1. 也是采用整数进行管理的！采用位运算，从第一位开始为1,逐一x2！



1. 每个集合类型8个字节，64位，因此可以表示64个元素！

注意：站在 mysql的角度，尽量多用枚举和集合！

但是站在php操作mysql的角度，尽量少用！（兼容性差）

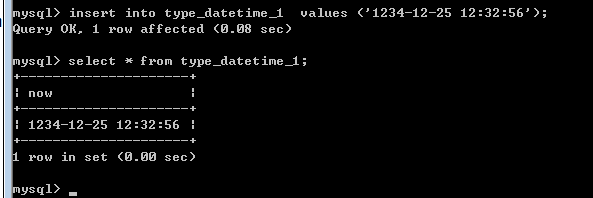
# 日期时间

## datetime

，日期时间，用于保存大范围的日期时间！

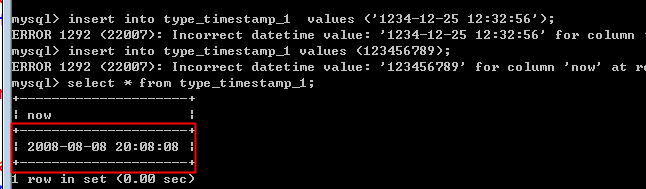
'1000-01-01 00:00:00'到'9999-12-31 23:59:59'

在处理时是使用字符串的形式进行管理！



## timestamp

时间戳，管理常用的典型时间，从1970-1-1开始。空间少，而且是以整型的形式管理，但是一个字符串的形式展示的！



## date

日期，如果只记录日期，不记录时间采用 date！

## year

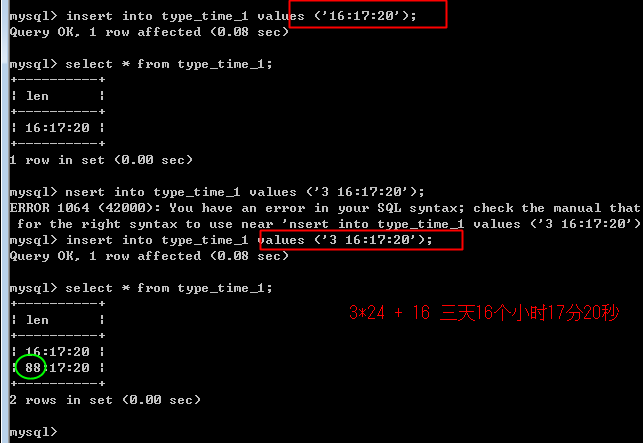
在保存年份的时候。

采用一个字节保存！因此只能表示 1901——2155年！

## time

时间，时间时刻！

还可以表示时间跨度！时间段的概念！



总结：

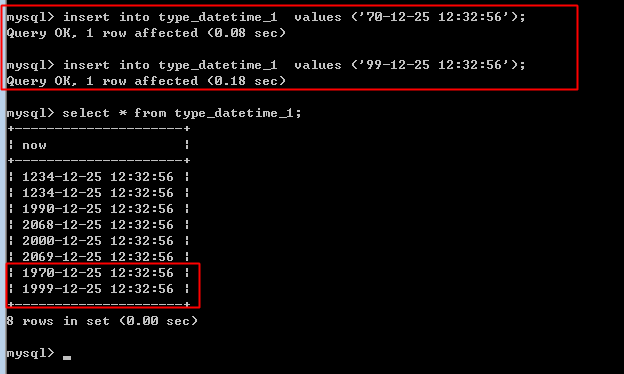
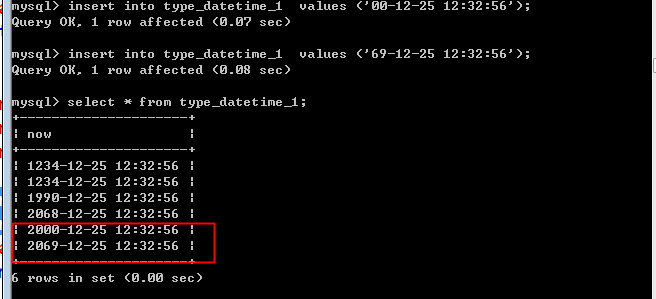
one：除了time，表示都是时间点的概念！time还表示时间跨度！

two：年份的表示

可以采用2位年份：

0-69 表示 20 XX

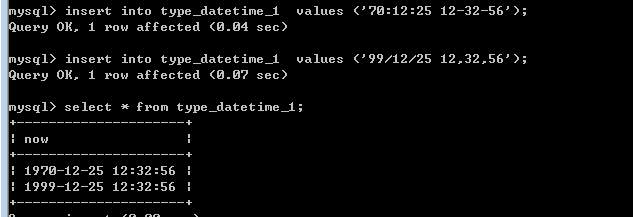
而 70-99 表示 19XX



1990

2090

1. 除了-还有：可以作为日期和时间分隔符之外，很多其他字符都可以！



放在php程序中：

小范围的日期：使用整型

大范围的日期：字符串！

# 列属性，字段约束

## NULL，not null属性空约束

mysql的NULL不是数据，也不是类型！只是标识属性！

用于说明某个字段，是否可以为null（是否可以什么都不存）

NULL采用关键NULL表示！（不是字符串）

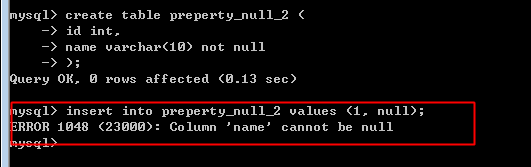
是：NULL，而不是：‘NULL’

属性：

null 表示可以为空



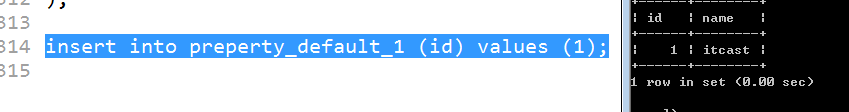
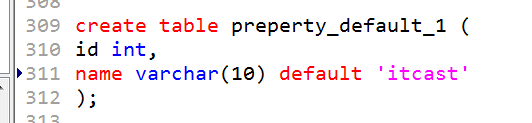
not null，表示不能为空



如果，在添加数据时，没有指定值，也可能会是NULL！

## default属性，默认值约束

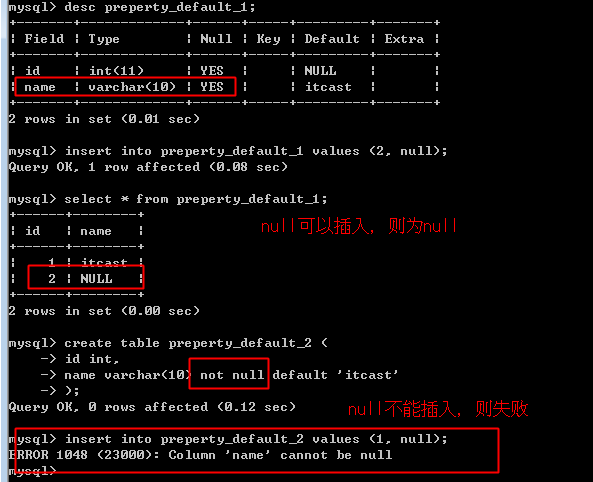
采用 default 关键字，来限定一个字段的默认值，在没有指定字段数据时，采用默认值！



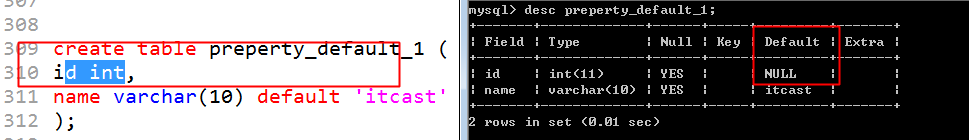
default 与 null 的处理关系！

如果此时，该字段被指定了一个null：

不能使用默认值，允许为null则为null，不允许则插入失败！

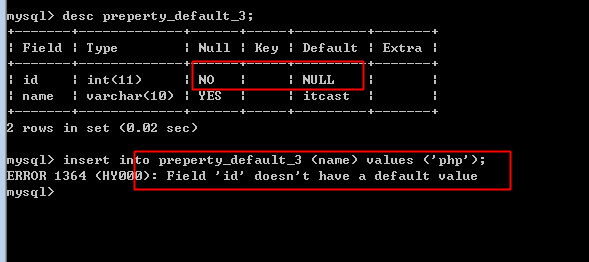


如果一个字段没有指定默认值，那么默认为NULL



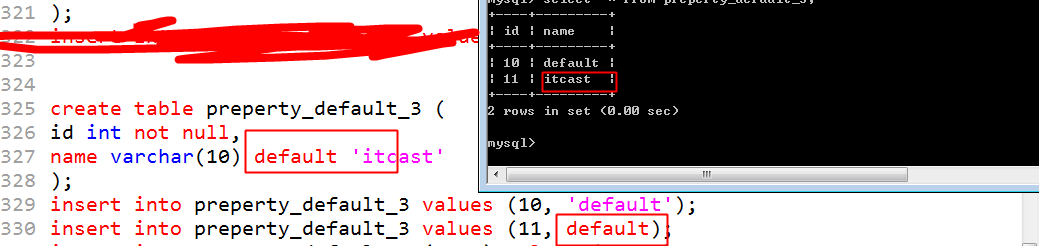
此时：如果在定义该字段时，不允许为NULL，则插入时，必须保证该字段有值才可以！

（另外一个选择是增加默认值）

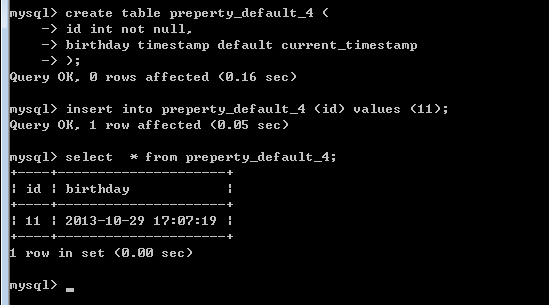


默认值，存在一些特殊的标记关键字：

default，用在值中的关键字！



current\_timestamp,用在第一个时间戳类型的字段上，表示当前的时间！



典型在很多表上增加一个 update\_time 将其默认值设置成 current\_timestamp。就可以记录下当前记录的最后操作时间！

## 主键约束/唯一约束，primary unique

站在约束的角度，限制的，该字段，值要唯一！

但是主键与唯一不是一个概念：

都是索引的一种！

主键：

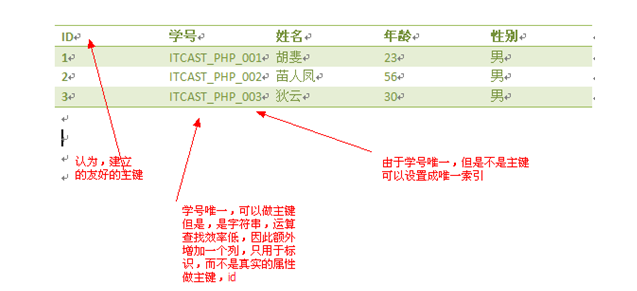
可以唯一标识记录的字段，称之为主键！

唯一：

保证在某个字段上的数据是唯一的，可以设置成唯一约束！

但是，一个表，只能有一个主键！

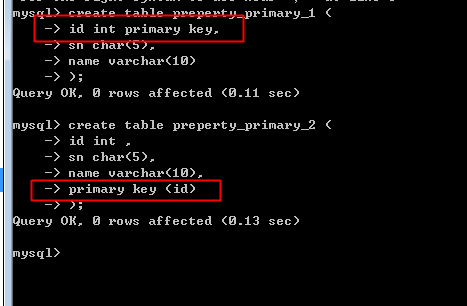
典型的，在创建表时，主动增加一个非实体的自然属性，充当主键，采用整型，运算速度快！



其他的唯一字段，建立唯一约束！

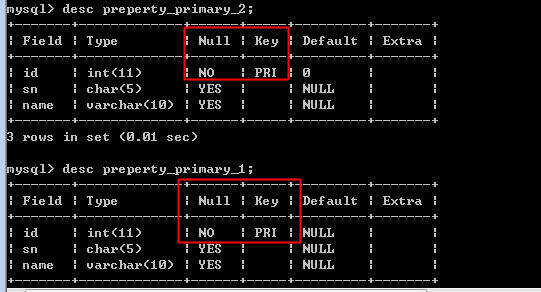
建立：

两种方案：

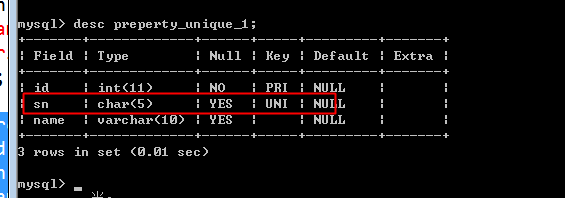
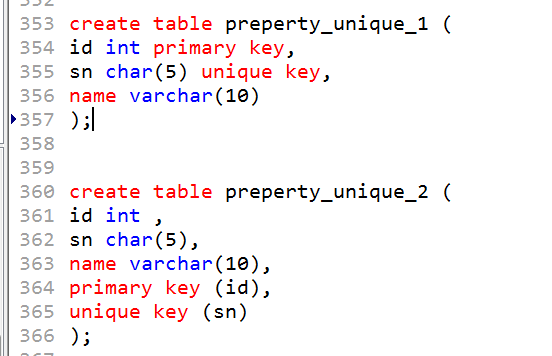


一旦创建了主键：

默认就是不能为空：

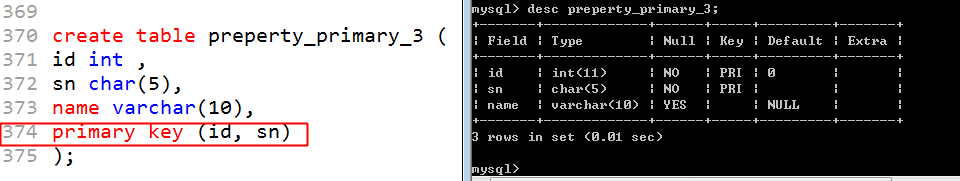


唯一，使用 unique key 来创建！



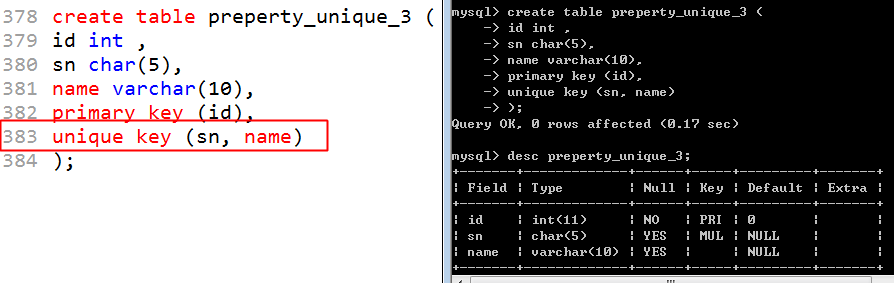
一个表可以有多个唯一，但是只能有一个主键！

注意：主键或者是唯一，都可以由多个字段组成！



因此，主键与字段的概念：字段来充当主键，不是字段就是主键！（称之为复合主键）

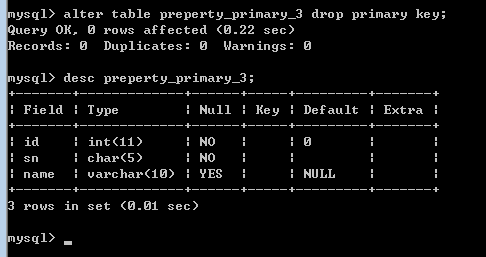
尽量采用 id 一个来作为主键！



### 如果管理的主键/唯一呢

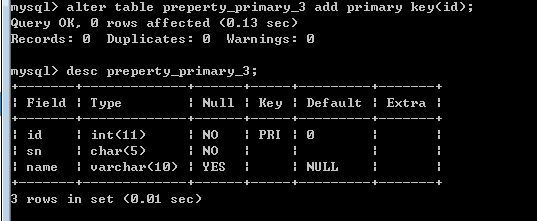
删除主键：

alter table表名 drop primary key;



添加主键

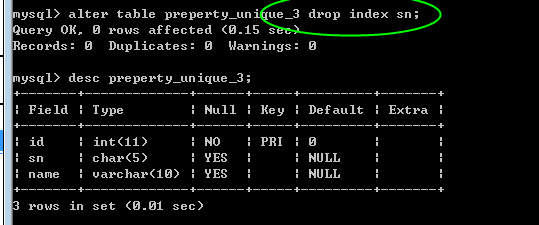
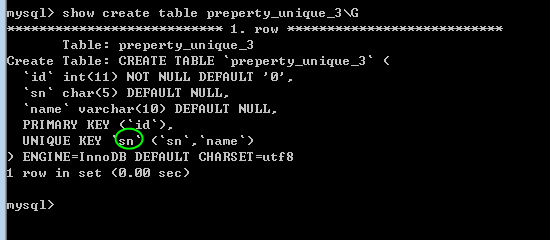
alter table 表名 add primary key(‘字段列表’);



删除唯一

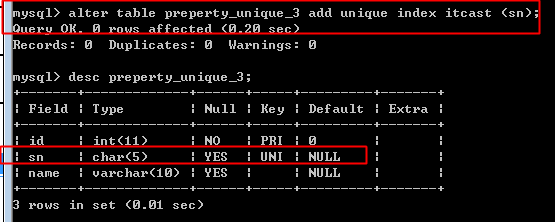
alter table 表名 drop index index\_name；

索引的名字，可以通过 show create table 看到！



增加唯一

alter table 表名 add unique key 索引名字 (字段列表)

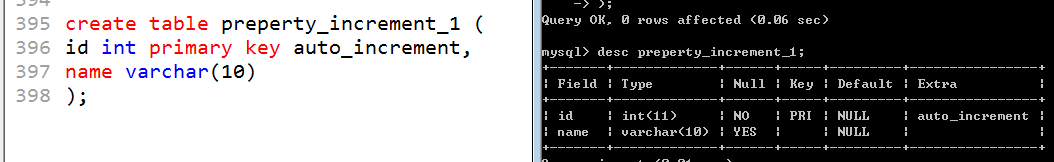


## 自动增长，auto\_increment

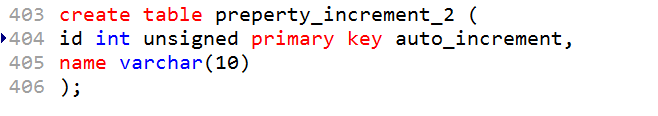
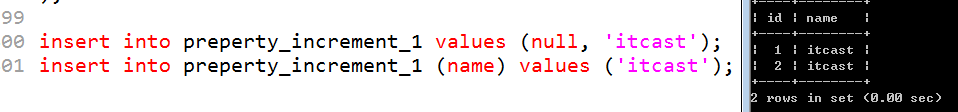
用于主键，只有一个字段主键，才能使用auto\_increment!

可以，从1开始，逐一递增的数值！

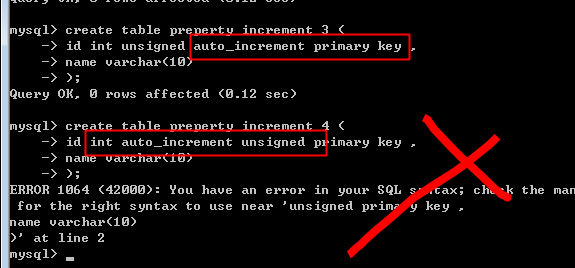
目的是保证唯一，计算方面！



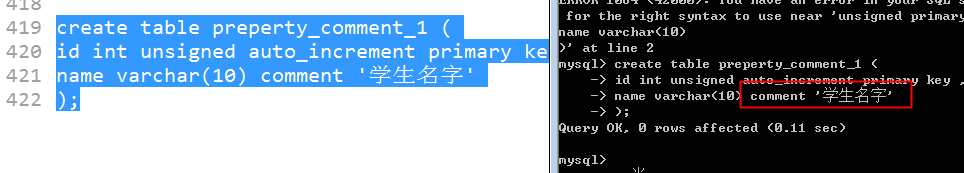
典型的，从1 开始，没有负数，可以采用无符号 unsigned整型！



注意： unsigned，不是列属性，是类型的一部分！（包括zerofill）因此位置上与类型在一起！



## comment，注释



外键！

# 预习

1. 范式
2. 表实体之间的关系
3. 外键约束

4，语句（查询语句）

# 作业

1 说说那些地方可以设置编码

2 增加对库的管理（删，增加）

增加对表的管理（增加，改名，删除）

增加对字段的管理（增加，删除）

增加对数据的管理（增加，删除，编辑）

位，bit，比特，计算机可以处理的最小单元

字节，Bytes 存储的最基本单位。KB 千字节， MB，兆字节，GB，1024\*MB，TB，1024\*GB。一个字节 8 个位。 1Byte = 8 bit.

字符，char，一个显示逻辑上的单位，一个图形就是一个字符。

字，双字节

双字，4字节。

1. 编程，定义一个函数

参数为 table\_name 表名，要求大家返回一个数组，内容是当前表的结构信息！只需要字段名,如果有主键，则在返回其主键(不考虑符合主键)

f1(stu)

array(‘id’, ‘name’，‘pk’=>’id’)

1. 建立一个学生信息表！尽可能利用多种类型！

**第三天**

编码：

建表，建库，建字段设置（数据库中的数据的编码）

PHP作为mysql服务器的客户端，设置的客户端编码和连接编码（set names）

设置php返回给浏览器数据的编码，（Content-Type,header(),<meta>）

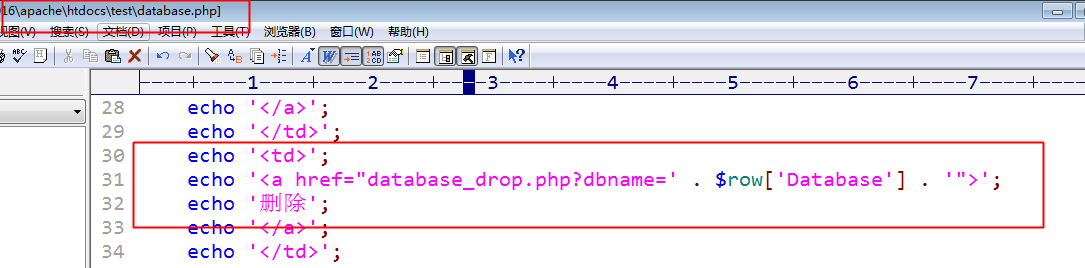
PHP文件本身保存的编码（文件编码，通过文本编辑器设置）

[浏览器查看时，可以强制指定编码]

## 删除库

增加删除的链接，传递所操作的库名！

在数据库列表页：



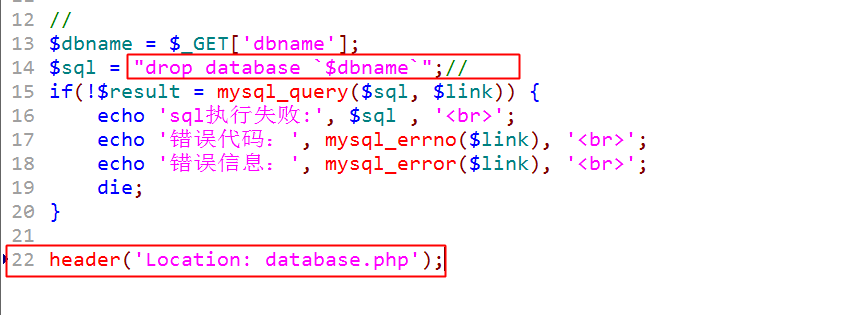
增加一个处理删除的功能页面：

在形成sql时，只要出现了标识符（库名，表名，字段名，索引名），都使用反引号！

请求跳转：

header(‘Location: url’);//告知浏览器，对新的url发出请求！

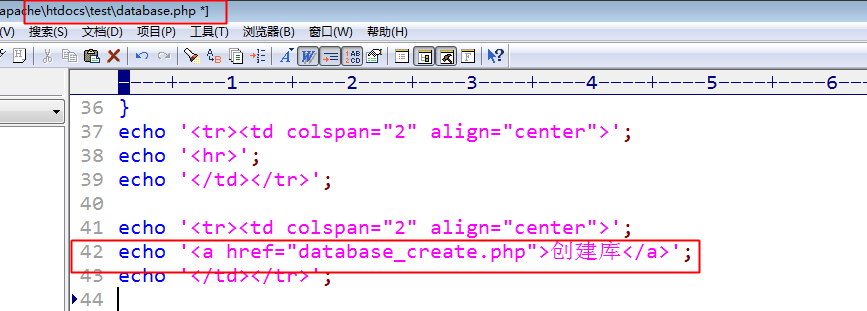
database\_drop.php



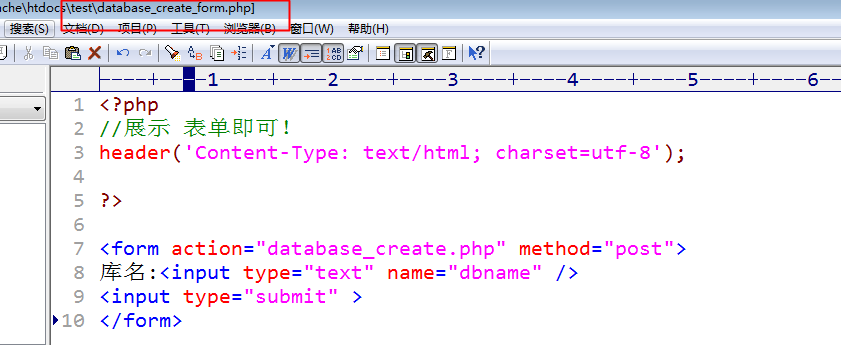
## 新建库

执行 create database

列表页，增加一个链接，请求到创建页：



增加一个create\_form表单页面



在增加一个脚本处理该数据即可

database\_create.php

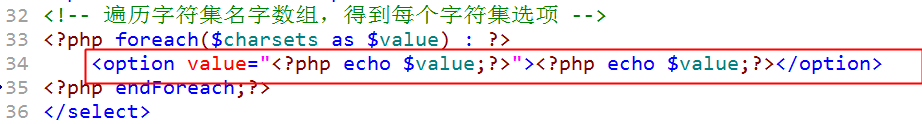
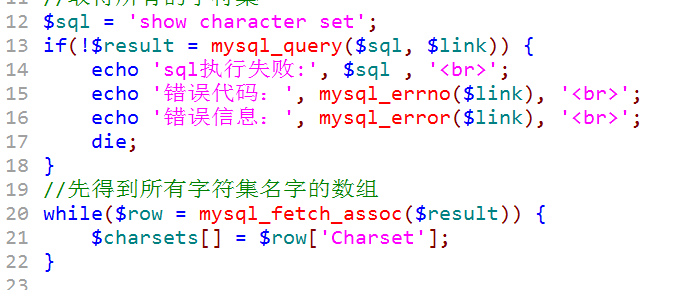


如果存在多个选项，因该如何处理？

例如字符集！

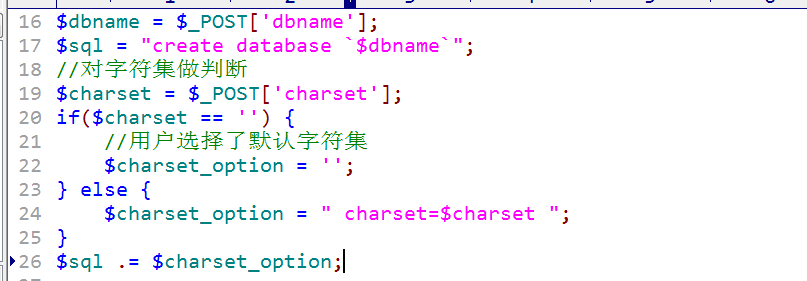
应该，取得所有的字符集。

利用 show character set;可以展示所有的字符集！



在创建库时，需要先对哪个字符集做一次判断！（是否是默认）

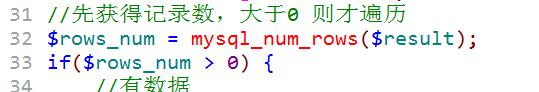
database\_create.php



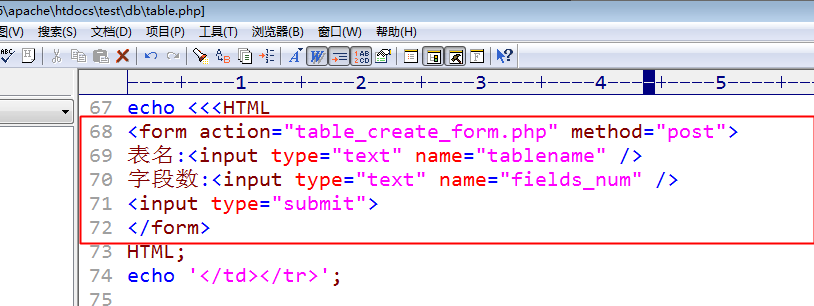
## 查询表：

table.php

利用mysql\_num\_rows($result)可以获得结果集中的记录数：

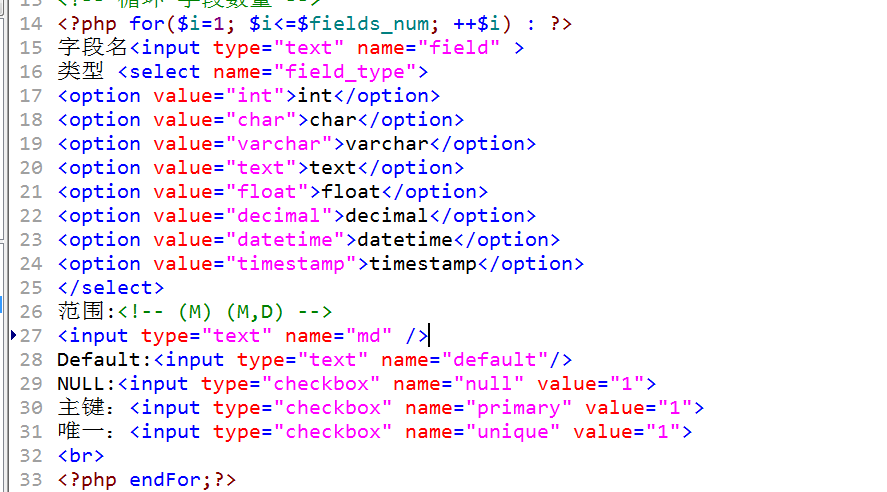


展示一个增加表名与记录数的小表单：



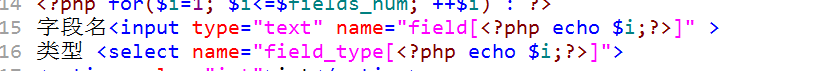
## 显示创建table的表单

table\_create\_form.php



## 处理提交数据：

应该为每组数据形成一个唯一的标识！采用数组下标的形式！



循环对数据做判断，然后依次拼凑sql语句：

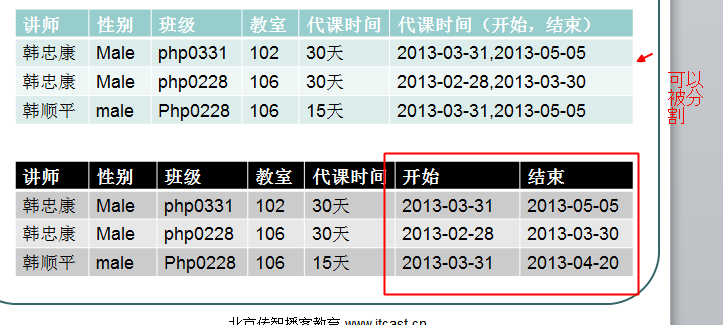
# 关系（二维表）的设计规范，范式

范式，NF，normal format，就是指对表的结构的要求！

目的：1,规范结构！2，减少数据冗余！

## 第一范式，1NF，字段原子性

要求字段不能再分，要求字段的原子性



## 第二范式，2NF，非部分依赖

增加唯一主键即可！ID

范式的要求，是逐渐递增！

在满足了第一范式的前提下，不能出现部分依赖！

部分依赖指的是：普通字段对主键是完全依赖的，而不应该是依赖主键的一部分！

依赖：可以通过那个字段去决定另一个字段

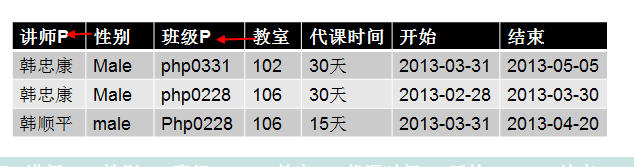
因此，出现主键部分依赖的前提是，出现复合主键！



其中代课时间，开始和结束时间，没有部分依赖！称之为完全依赖于主键：

但是，性别，依赖于讲师字段即可！

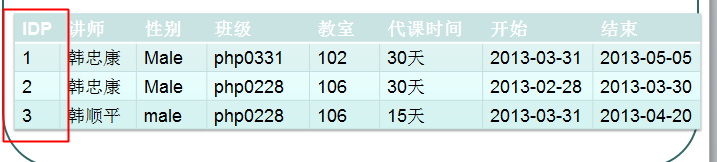
讲师字段是主键的一部分！因此称之为性别部分依赖于主键



因此，该表不符合第二范式！

怎么做？

消灭复合主键即可！增加一个唯一字段的主键即可。增加一个与业务逻辑毫无关系的，唯一的ID主键，int unsigned primary key auto\_increment



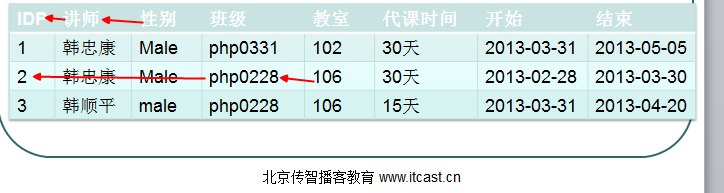
## 第三范式，3NF，非依赖传递

在满足第二范式的前提下，取消传递依赖，就是第三范式！

传递依赖：如果字段B对字段A有依赖，而字段C对字段B存在依赖。则出现了传递依赖！

讲师依赖于ID，而性别依赖于讲师。

班级依赖于ID，而教室依赖于班级。

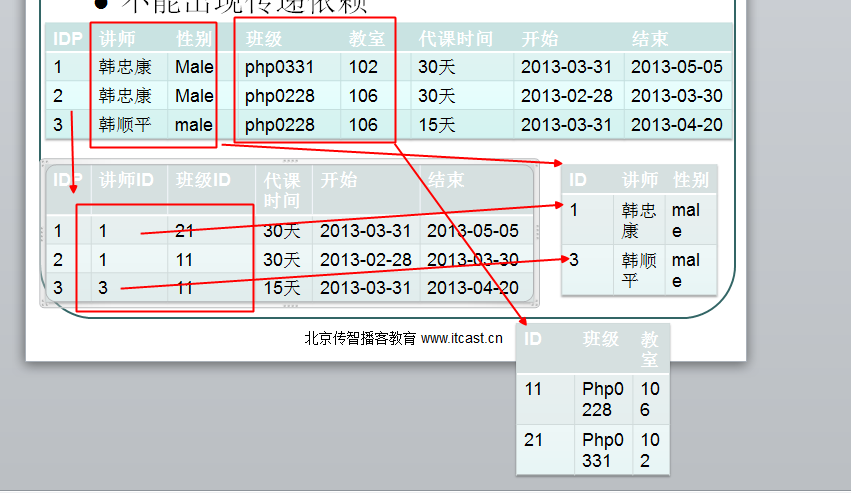


称之为传递依赖！

解决，要保证所有的字段都完全依赖于主键，而不依赖于其他字段！

将独立的实体信息，使用独立的关系（二维表）进行保存！

分别增加讲师，班级表，将代课信息内的讲师和班级信息拆分出：



总结：

每个实体建立一个表，为每个表增加一个主键ID即可！

一个实体表应该如何设计

# 实体之间的关系的设计

多个是体表应该如何设计！

## 实体之间存在哪些关系？

班级，学生两类实体！

一对多，多对一，1:N, N:1

班级，讲师两类实体！

多对多，M：N

学生常用信息，学生不常用信息

一对一，1：1

## 如何设计？

### 多对一，一对多

在多的那端（那个表内），增加一个字段，用于保存于当前记录相关联的一端记录的主键！



### 多对多

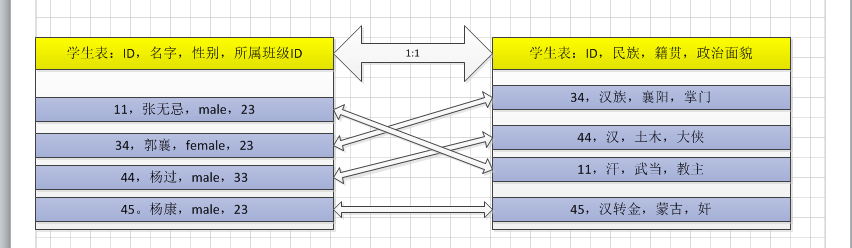


增加一个专门管理关联的表，使班级与讲师都与关连表存在联系。从而是两个实体间有多对多的关系！



因此，一个多对多，会拆分成两个多对一！

### 一对一

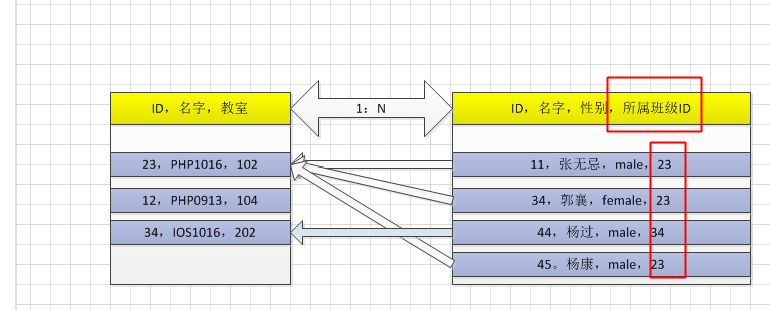


可见，两个表之间存在相同的主键ID即可！

# 外键约束

约束的作用，是用于保证数据的完整性或者合理性的工具!

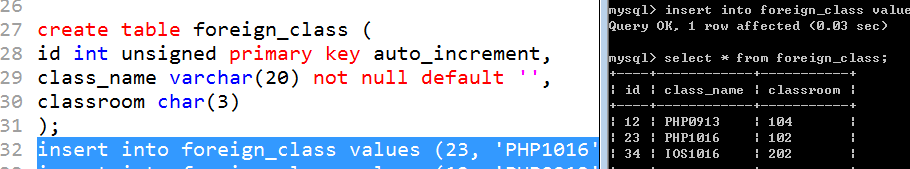
外键：foreign key，当前表内，指向其他表的主键的字段，称之为外键！



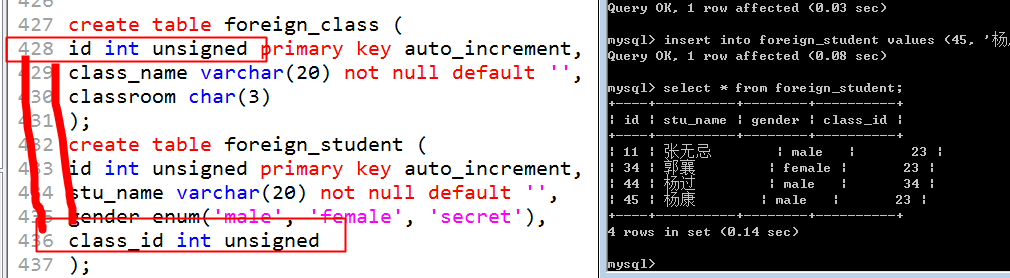
外键约束：用于限制相关联的记录在逻辑上保证合理性的约束称之为外键约束！

约束，不是字段。

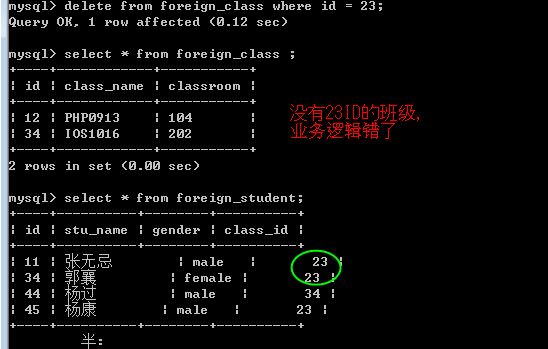
建立班级表



再创建学生表



看看删除班级的情况：



出现了不合理数据：

此时，可以通过增加外键约束的方式，来限制以上的操作！

## 增加外键

alter table 表名 add constraint 约束的名字 foreign key 外键索引名字 (外键字段名) references 关联表名 (关联字段) [操作]



再删除个试试：

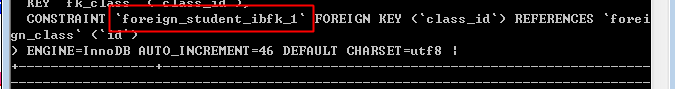


注意：如果当前的数据，已经不符合所见约束关联，则创建失败！

## 删除外键

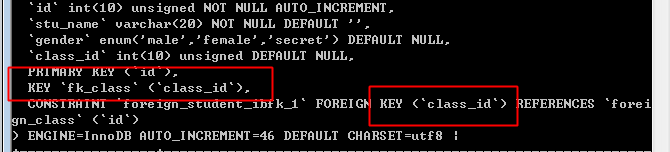
alter table table\_name drop foreign\_key外键名字！

可以通过 show create table 查看约束的名字：

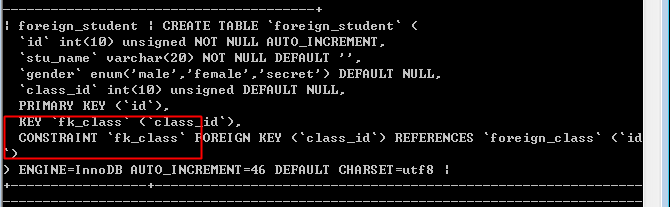
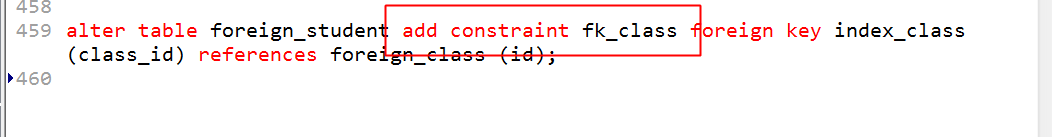


注意，外键约束与索引的关系：

如果需要在某个字段上，增加外键约束，那么需要该字段也同样有索引！如果该字段上，没有索引，此时，mysql会自动在该字段上增加一个普通索引！



可以选择指定外键约束的名字：



注意上面的外键约束自动建立的索引的名字，与外键的名字相同！

总结：在创建时：

1. 外键与相应关联表的主键类型
2. 已有数据，必须满足约束条件才可以！
3. 可以使用 constraint 关键字，为外键约束起名字！

## 约束操作

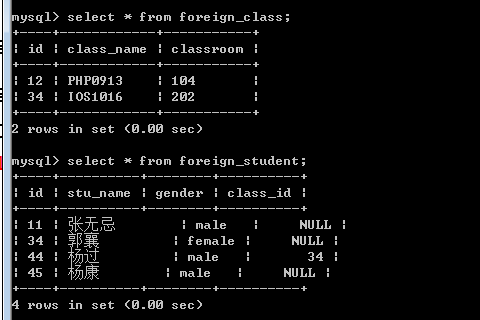
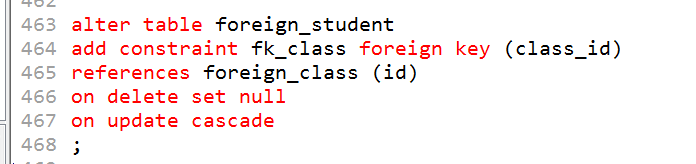
在对父表（被关联的表）做操作时，有三种行为：

1. 严格限制，拒绝操作。restrict
2. 置 null，set null
3. 级联操作，cascade

以上三个行为操作，会在主表记录被删除或者更新时被使用！

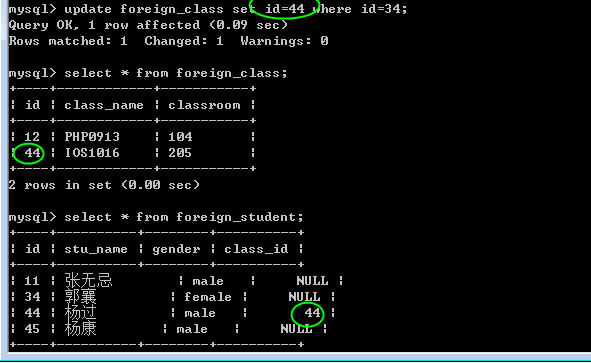
on delete set null

on update cascade



更新时的级联操作：

只有在关联表的主键发生变化时，才会影响到从表的关联字段的变化！

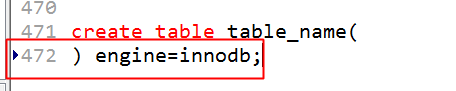


主表：被关联的

从表：发出关联的！

应该注意的问题：

关于，外键约束，只能在当前的mysql的的innodb表类型（引擎）下才会生效！

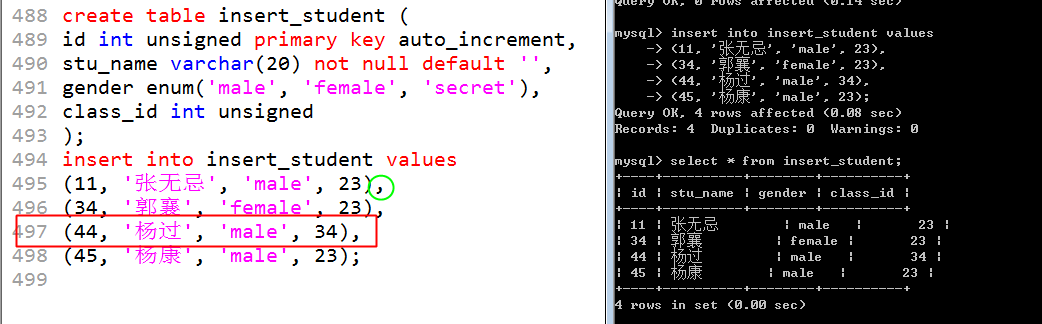


外键，站在php程序的角度，用到的不多！

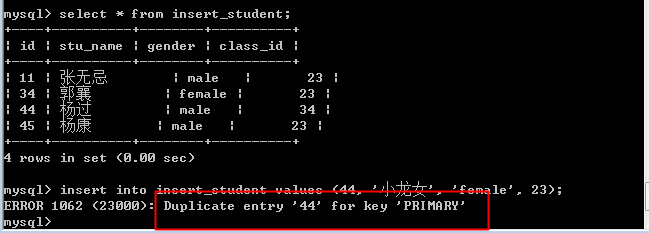
数据操作，DML，DATA Management Language

# 增加操作insert into

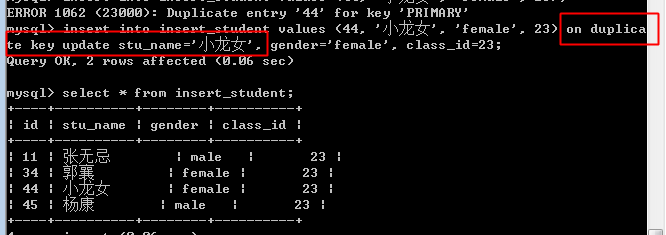
## 一条insert into可以插入多条记录



## insert into 支持判断主键（唯一）是否冲突，从而执行更新

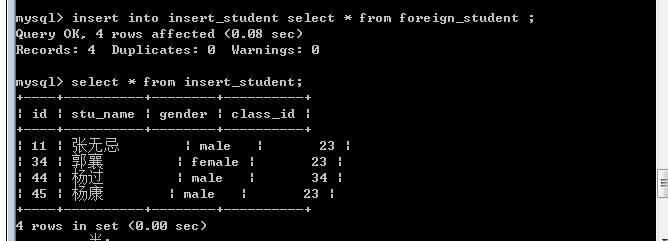


insert into 表名 (字段列表) values (值列表) on duplicate key update 字段=值,字段=值



注意，update后没有 set

## insert into 表名 select 语句;



注意，并不是一定要字段数一致，才可以完成操作，只要是字段数量与字段类型一致，就可以完成插入！

# 查询 select 的用法

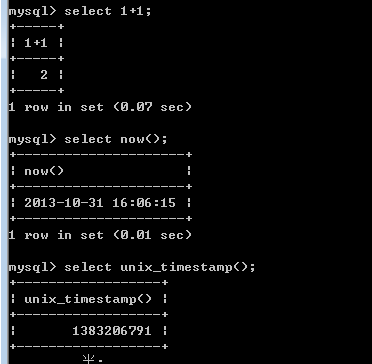
## 基本的查询语句的结构

select [字段表达式列表] [from子句] [where 子句] [group by 子句] [having 子句] [order by子句] [limit 子句]

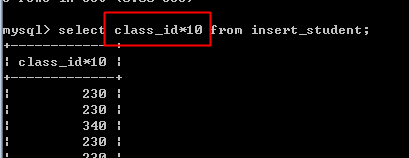
## 字段表达式子句

注意，表达式与别名的概念

1+1，$v1+10, abs(-10)

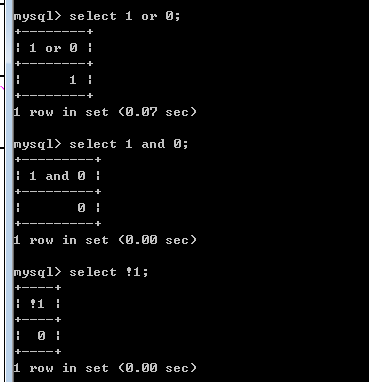


如果为字段名，那么字段名就是一个变量的概念，可以参与运算！

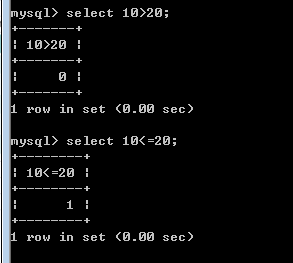


因此，可以利用各种运算符，来形成SQL中的表达式！

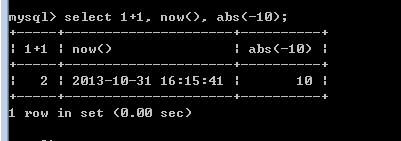
例如：逻辑运算符：



关系运算符：

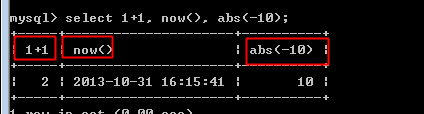


可以是一个列表



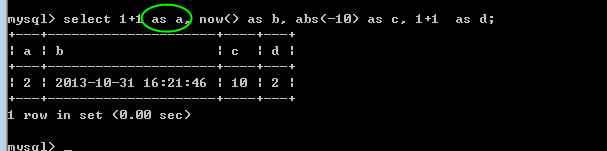
每个表达式，可以有一个别名：

select结果内，以表达式本身来命名的！

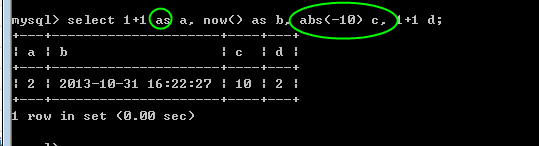


但是，有两个典型的问题：

1. 表达式直接做名字不容易识别！
2. 如果名字冲突，不能区分！



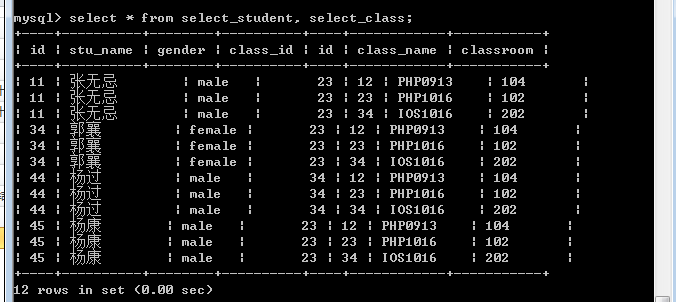
使用 AS 关键字可以使用别名，可以省略as关键字，但是不建议省略！



## from子句

表示查询的来源，就是表！

可以写表名列表，使用逗号分割

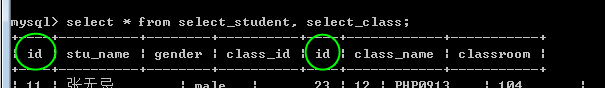


如果此时没有条件，相当于形成了一个笛卡尔积！

A集合的每个元素，都与B集合的每个元素之间有个关联！

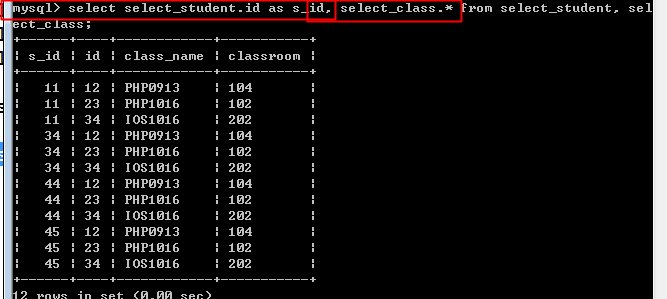
A表的所有记录，都与B表的所有记录之间存在关联！

此时结果中会将所有的字段都列出来，包括重名的！



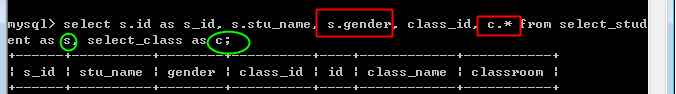
可以为相应的字段名起别名：

此时访问到某个字段需要使用表名.字段名的形式！



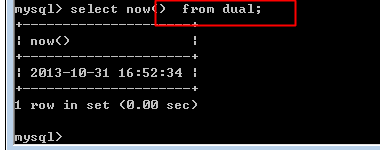
表的别名

如果多次出现表名.字段名的情况，可以为表名起别名！



dual 的问题

虚拟表名问题



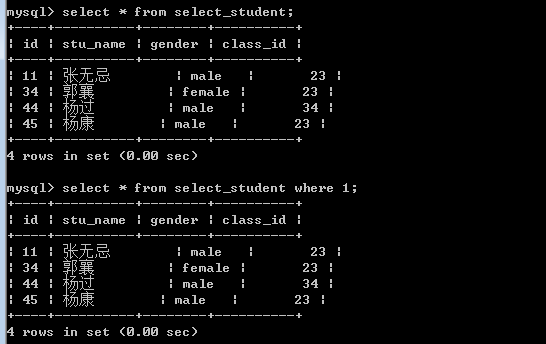
是语法更加规范而已！

## where子句，查询条件子句

where 条件表达式

运算符，变量（字段）

省略where子句，相当于永远为真~



where，找到每条记录，并依次执行条件表达式，根据条件结果返回数据！

形成条件表达式的基本要素：

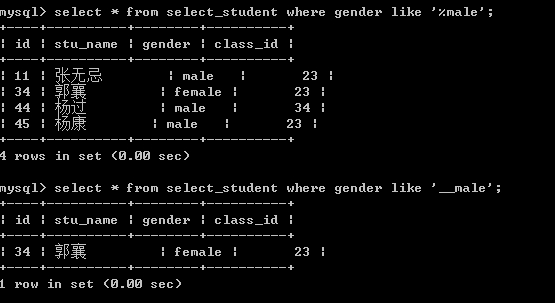
数据，（变量），运算符，函数调用()

典型的运算符：

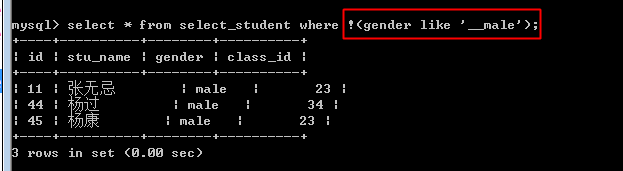
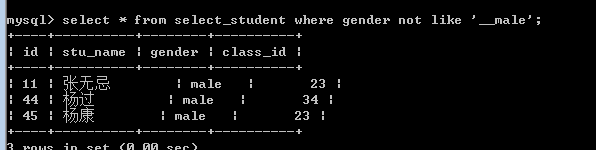
关系： > < >= <= != =

like模糊查询。数据 like ‘模式匹配符’

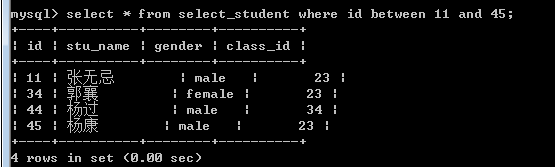
可以使用 % 和 \_ 作为通配符



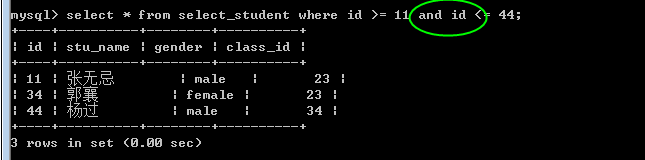
not like,不像like取反！



between 3 and 5，在某个区间，闭区间。



包括边界值：

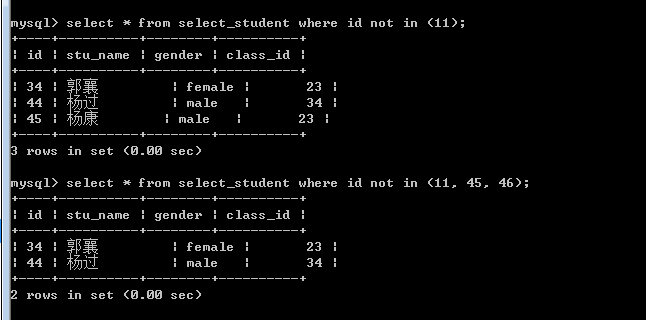


in，在某个集合之内

in (元素列表)

not in，不在某个集合内

not in (元素列表)



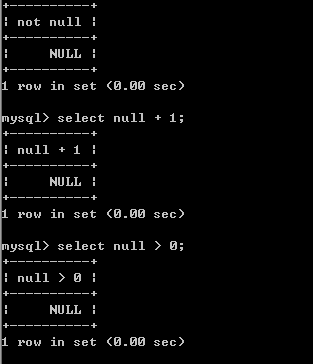
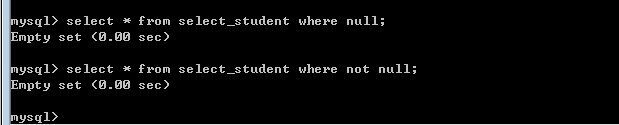
留心：

再做批量操作时，in notin的出镜率很高！

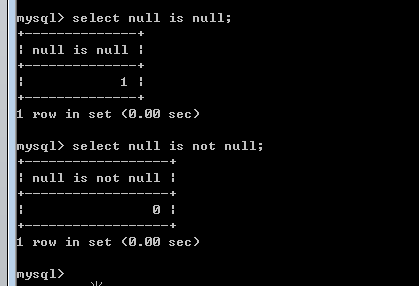
集合（3,4,5）区间（3,4）

null值的判断

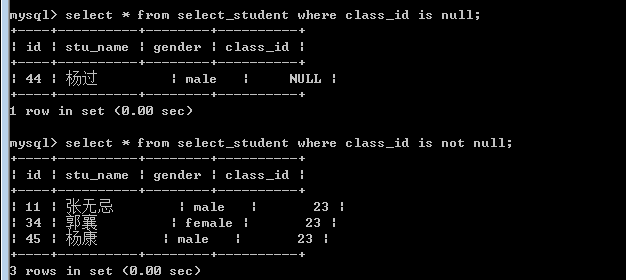
不能使用普通的运算符，因为运算的结果都是null，而且不能作为查询条件！



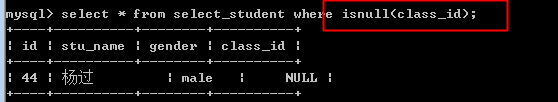
应该使用 is null 或者 is not null 来判断！



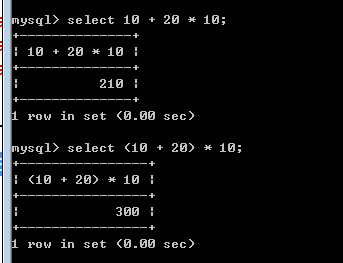
应该使用上面的来完成判断：



函数（isnull()）也可以完成类似的判断：



注意：运算符也有的优先级的概念，注意可以使用 () 来修改优先级！



**第四天**

# select 语句

## group by 子句，分组查询

对查询结果（已经通过where子句过滤之后的数据），按照某个字段，进行分组！

group by 字段！

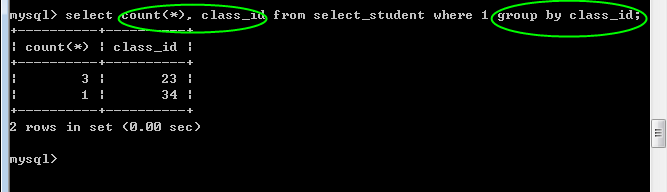
在分组的结果中，只会显示组内的头一条记录！因此，通常，分组之后的数据，除了分组的字段外，其他字段的逻辑含义很轻！

分组的作用，不在查询每个组内的具体数据。而其作用主要是在分组统计上：

### 合计函数

此时需要使用统计函数（合计函数）加以配合！

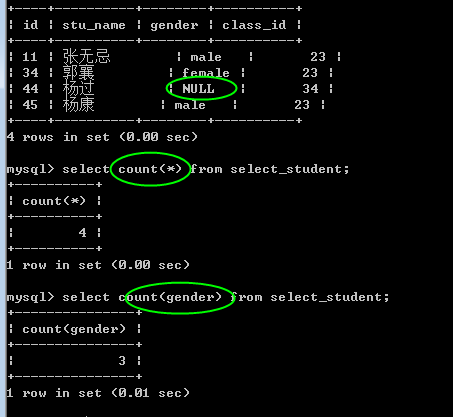
合计函数例如：count() 可以统计结果中的记录数，但是一旦使用了分组查询，则只会统计组内的数据！



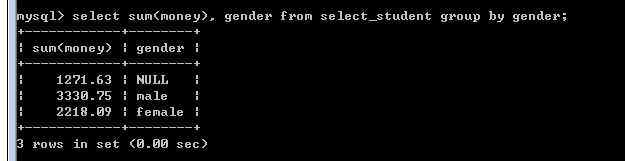
count()，统计记录数。典型的使用是 count(\*)，但是除了\*之外，是可以使用字段名的！

其中，只要记录存在，则count(\*)就会统计到数据，而如果相应的字段为null，则count(字段)不会统计上数据：

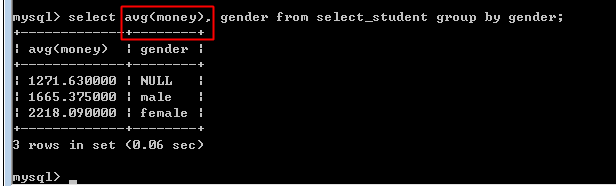
看下面对 gender 的统计结果：



sum(字段表达式)，统计和，对某个字段求和！



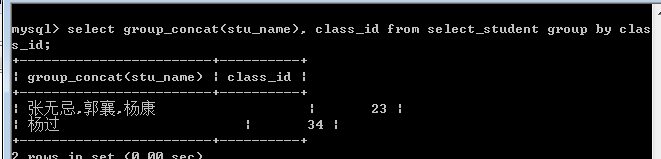
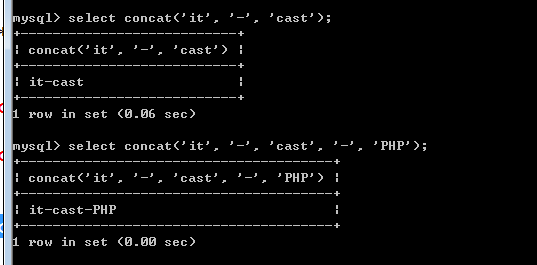
avg()，平均值



max(字段表达式)，最大值

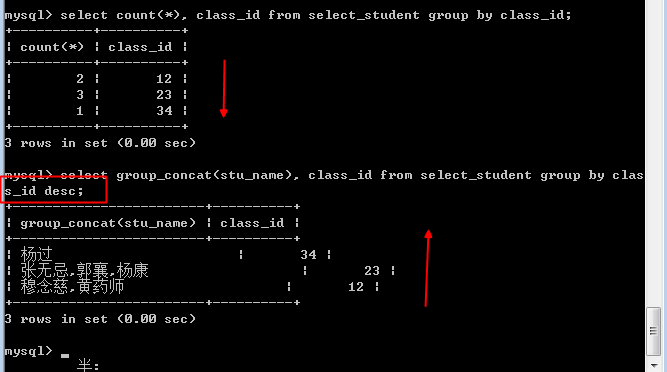
min(字段表达式)，最小值

group\_concat(字段表达式)，组内连接字符串



### 分组排序

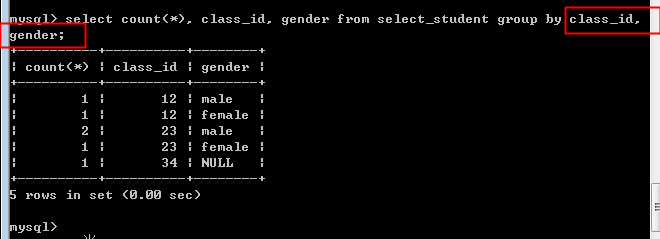
默认的分组后会按照分组字段对结果进行排序。可以group by子句指定排序的方式（升序ASC和降序DESC）



### 多字段分组

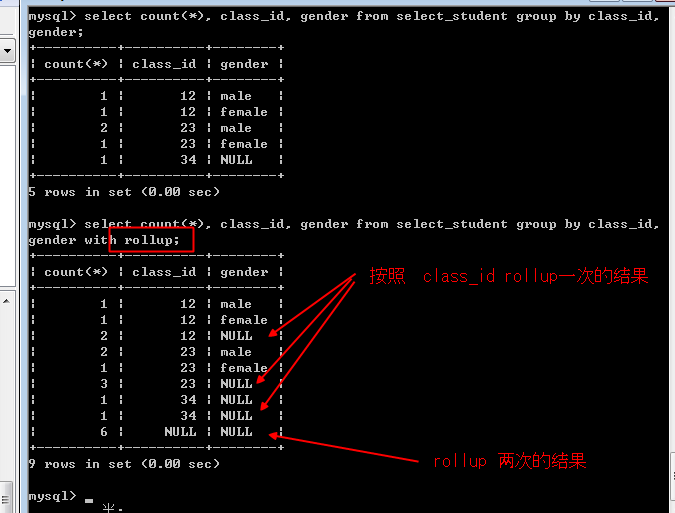
使用逗号分隔开多个分组字段即可！统计时，会按照多个字段的组合分组生成结果！

例如：统计每个班级内的男生和女生的数量！



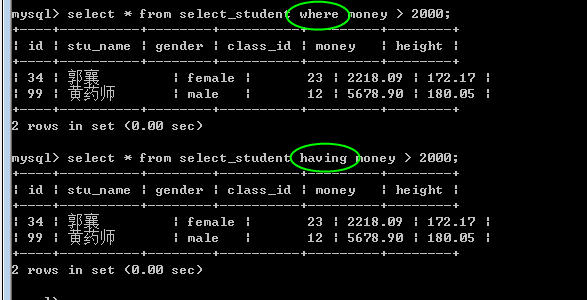
如果是多字段分组，需要查看每个分组的的详细情况：

可以使用关键字with rollup关键字来回滚统计：

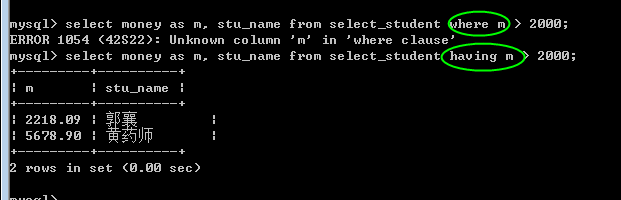


## having子句，条件子句

功能上与where类似，都是条件子句！



主要的区别，在于执行时机：

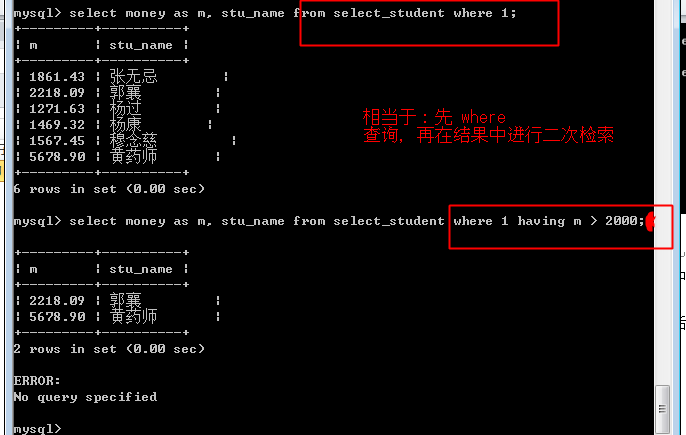


执行时机：

where，是开始时，从数据源中检索数据的条件。

而 having，是在筛选，分组之后，在得到的结果中，再次进行筛选的语法！

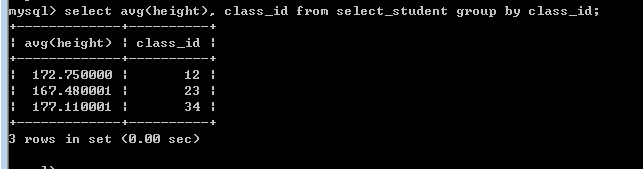
因此 having的结果一定是 where 已经过滤之后的结果!



having的作用在于，对结果进行二次处理！

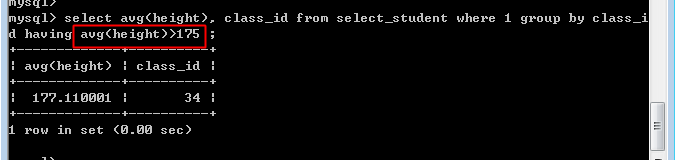
例如：找到平均身高高于 175cm的班级：

每个班级的平均身高：



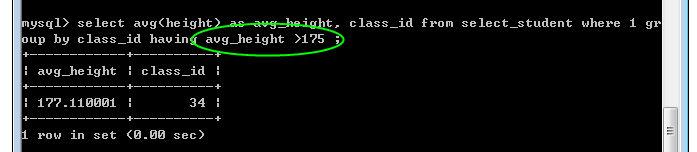
查询条件是 avg(height)之后的结果：

此时，where和groupby已经执行结束！可以使用 having 进行二次过滤：



为啥要有 having：where没有办法与合计函数一起使用！原因在于执行顺序问题！

典型的应该使用别的形式为 having完成条件表达式：

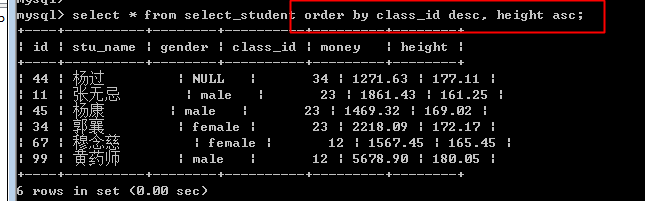


## order by 排序子句

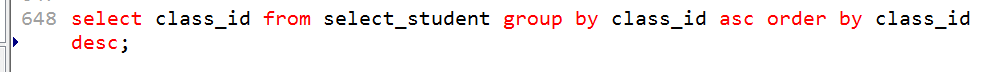
对结果进行排序的语句！

order by 字段名 [asc|desc], [字段名 [asc|desc],]

可见可按照多个字段进行排序



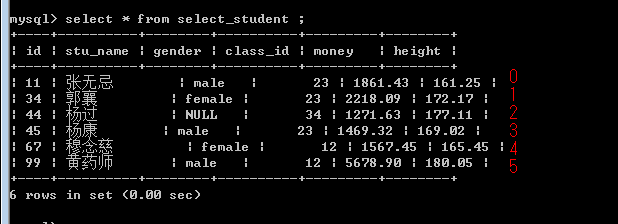
原则是，先按照第一个字段进行排序，如果字段值相同，则采用第二个，以此类推！



## limit 子句

限制结果记录数的子句！

从所有的结果中，选择部分结果的子句！



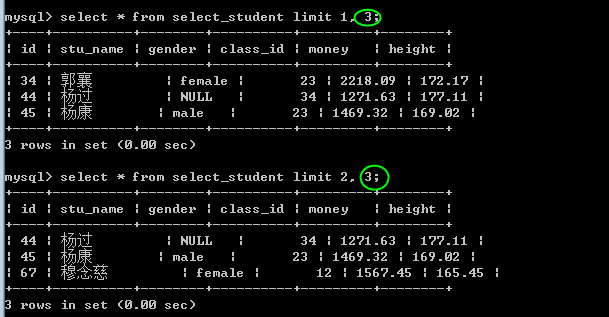
上面的是记录的位置：

可以从某个位置开始，取得多少条！

limit start, size;

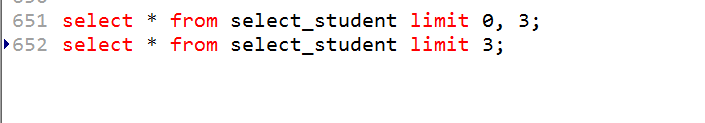
start:起始位置

size，取得的记录数



注意，第二个参数是，长度，而不是终止位置！

还有一个简写，省略start起始位置，表示从第一条记录开始：



分页！

## select的子句的总结

select子句的全部子句

### 书写顺序

字段表达式，from子句，where子句，group by子句，having子句，order by子句，limit子句。

书写顺序，与执行顺序！几乎是一样的！

### 执行顺序

from

where

group by

字段表达式，合计函数表达式

having

order by

limit

书写顺序不能错，但是子句几乎都可以省略！省略表示不发生操作！

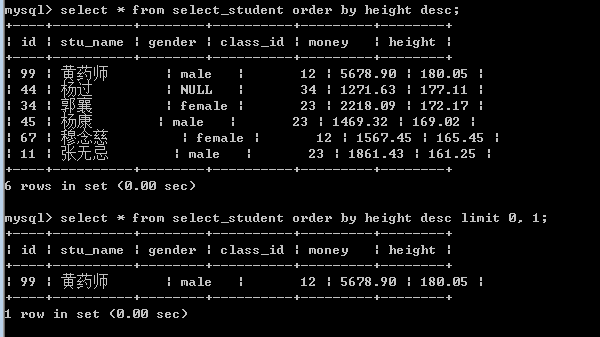
# 子查询，sub-query

出现在其他语句内部的查询语句，称之为子查询

场景：

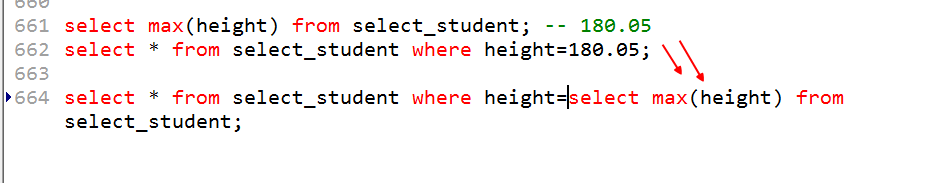
最高的学生

按照身高，降序，取得第一个！



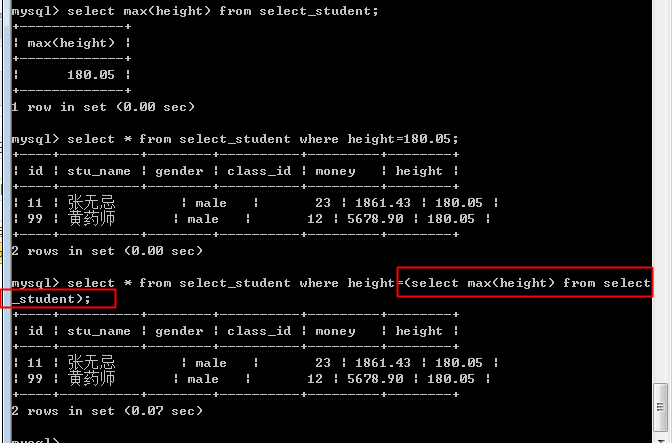
问题是出现等高的学生，问题不能处理!

应该，找到最高的身高，然后找到所有符合最高身高的学生！



**此时，**select max()出现在另外的语句内部，称之为子查询语句！

注意：子查询，应该始终出现在括号内！



## 子查询的分类

分类的依据！

两种分类依据：

1. 依据子查询出现的位置！

where型子查询，出现在where子句内！

from 型子查询，出现在from子句内！

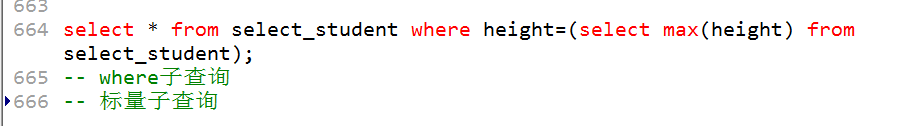
1. 依据子查询的返回数据的格式！

标量子查询，返回值是一个数据，称之为标量子查询！

列子查询，返回一个列，

行子查询，返回一个行，

表子查询，返回的是一个二维表



## from型

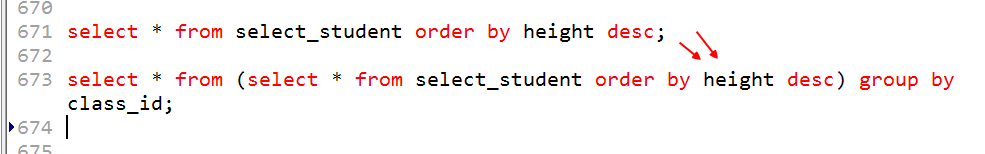
场景：

查询，每个班级之内，最高的学生

应该先将每个班最高的放置在班内的第一个位置，再按照班级进行分组！

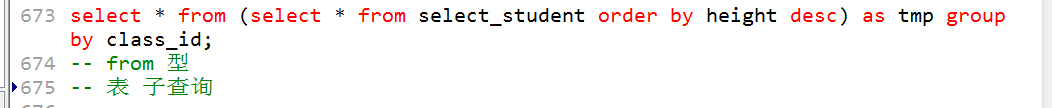
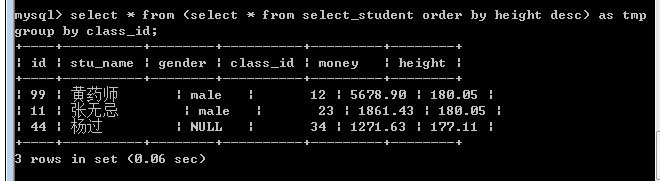
不能使用 order by 再 group by。

而需要，先用一个查询，得到身高排序结果，再将该结果分组！



留意：from 需要一个数据还是一个表，需要将子查询返回的数据，导出成表即可！

为子查询起个别名即可！



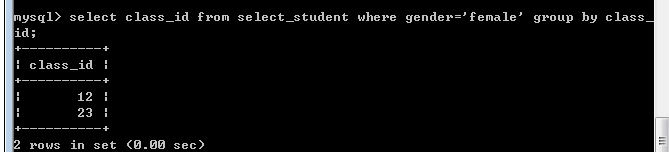
## 列子查询

返回值应该是一列！

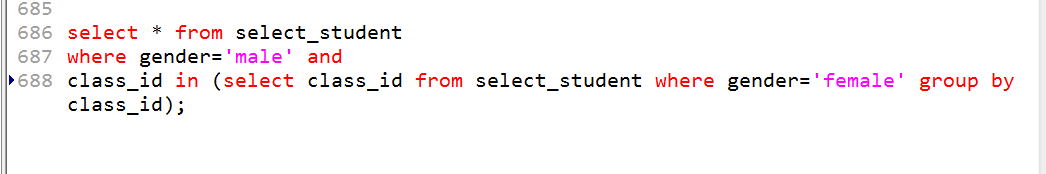
由于返回的是一列，是一类数据，看成是一个集合！

找到班级内有女同学的男学生信息：

条件，先确定哪些班级有女生！



再在该班级之内，找到所有的男生！



典型的列子查询使用 in， not in作为子查询的条件！

列子查询，还可以使用

= some

!= all

或者其他的运算符配合 some() 和 all（）语法完成！

some(), 表示集合中的一部分！

all(), 集中的全部

测试：

=some() 相当于in么？

!=some()相当于什么？不相当于not in！

哪个相当于not in ，与集合内的任何一个值都不相等！

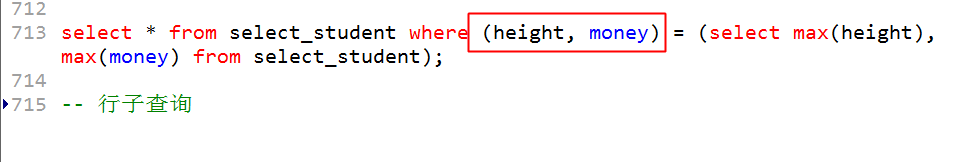
!= all() 是not in!

any就是some，一个功能！

## 行子查询

场景：找到，高富，最高并且最有钱！

使用行子查询可以，一次性查出来一个行（多个行）使用行进行匹配：



上面使用了 ()，构建了一行！与子查询的行作比较！

## exists型子查询

判断依据不是根据子查询所返回的数据！只是根据子查询是否存在返回数据来看；

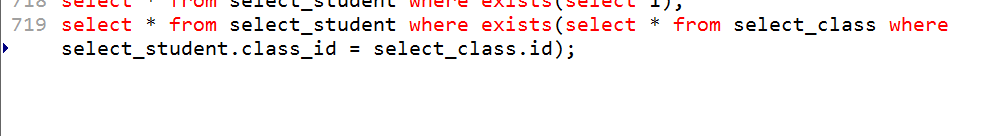
exists (子查询)

如果子查询存在返回数据，则exists返回真。反之，返回假！

出现在where条件内：

场景：

班级已经不存在的学生！



not exists！

# 连接查询，join

连接，多个表记录之间的连接！

场景：

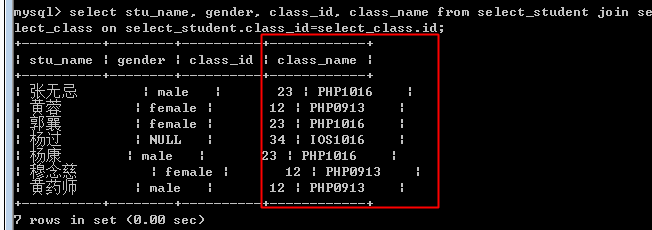
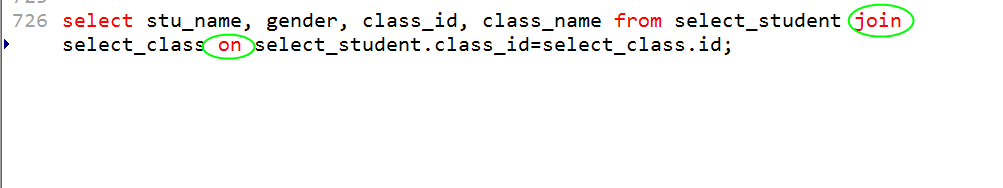
需要得到一个学生列表，要求是，展示：

学生，性别，班级名字

此时需要不单从学生表获取数据，还需要从班级表获得数据！

语法：

from 表名1 join 表名2 on 连接条件



过程是，先执行 from子句，需要连接join。

两个特殊的地方：

join ，连接

on ，连接条件

## join 连接语法

除了默认的连接之外，有其他形式的连接方式

内连接

外连接，左外连接，右外连接，[全外连接，也是外连接，但是不是mysql所支持的]

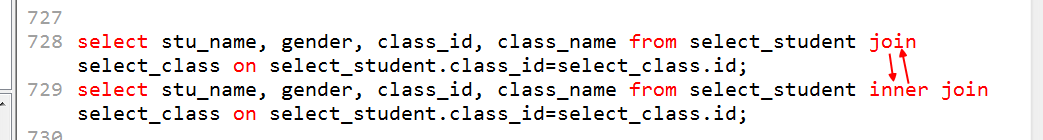
交叉连接

自然连接

### 内连接，inner join

记录与真实的记录连接，称之为内连接！（两条真实存在的记录的连接）

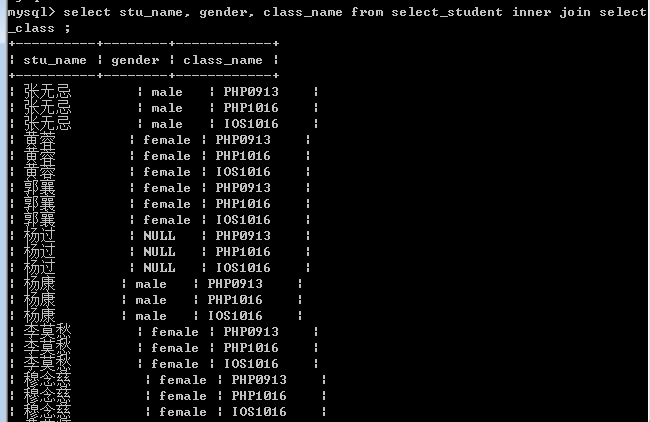
mysql默认的连接就是 inner join



内连接，可以连接省略条件！

on可以省略：相当于连接条件永远成立！

返回值是一个笛卡尔积！

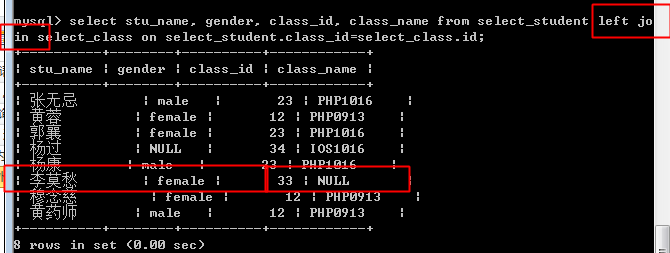


### 外连接，left join，right join

分成：左外连接left join，右外连接right join！

连接的记录，可能是一方不存在的！（两条记录中，可能某条不存在）

总结：内连接，外连接差别不大，只是外连接会将没有连接成功的记录，也出现最终的连接的结果内，而内连接，连接的结果只有连接成功的（两条记录都存在的）



注意好左外与右外的区别：

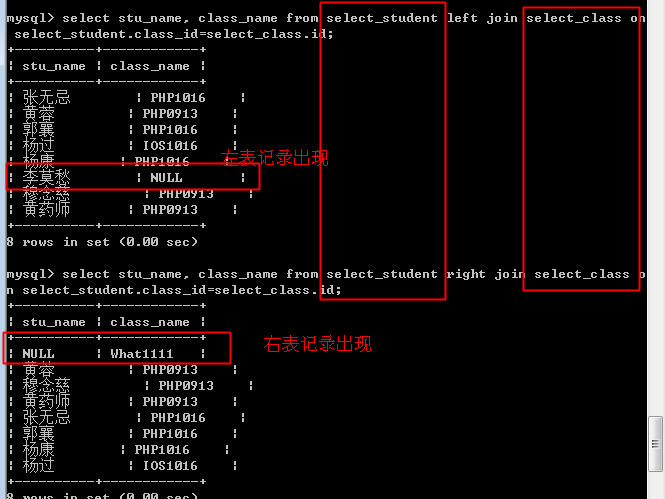
区别在于，那个表的记录（指的是连接失败的记录），会最终出现在连接结果内？

什么是左表和右表？

join关键字前面的（左边的）左表，join关键字后边的（右边的）右表！

左外：如果出现左表记录连接不上右表记录的，左表记录会出现正在最终的连接结果内！而右表记录相应设置成NULL。

右外：如果出现右表记录连接不上左表记录的，右表记录会出现正在最终的连接结果内！而左表记录相应设置成NULL。

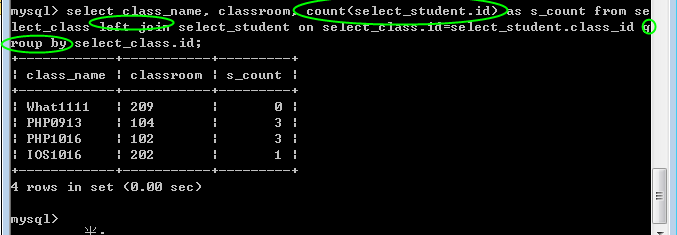
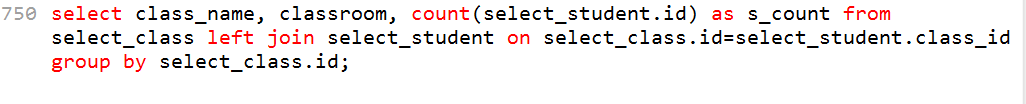


因此，可以交换表的位置，达到使用left与right join 混用的的目的！

问题：

统计每个班级内，学生的数量！，在班级列表内：

班级名，教室，学生数量



注意，外连接应该有条件!

### 交叉连接，cross join

结果与内连接一致！

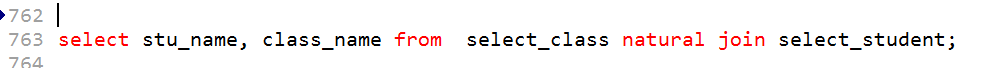
有时，在获得笛卡尔积时，显式的使用交叉连接！

交叉连接相当于是没有条件的内连接！

### 自然连接，natural join

mysql，自动判断连接条件，帮助我们完成连接！

典型的条件就是，表中的同名字段！

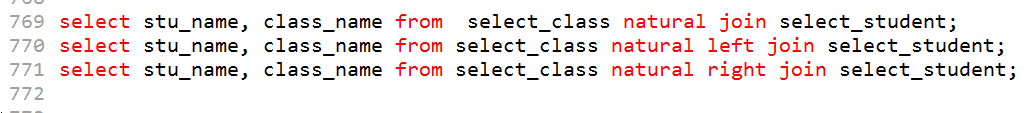


而自然连接也分内连接与外连接！

自然内连接：natural join

自然左外：natural left join

自然右外：natual right join



总结：

最终的效果只有：内，左外，右外！

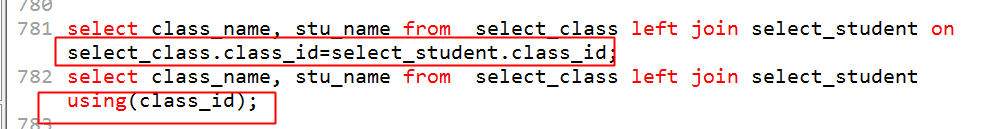
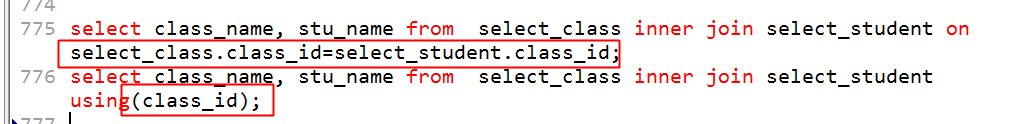
交叉，特殊的内!

自然，相当于自动判断连接条件，完成内，左外，右外！

## 连接条件，on，using

on，后面使用一个连接条件表达式！

using(连接字段)，要求使用同名字段进行连接！



using 的特别地方：

会对字段列表做一次整理！将连接字段作为一次显示！

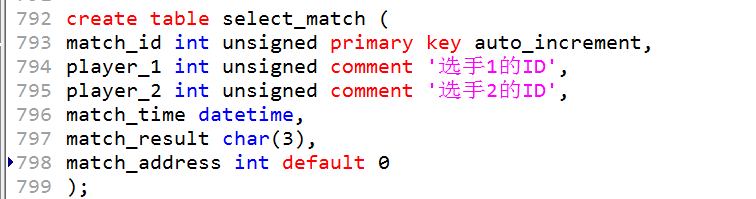


练习

设计一个系统，保存乒乓球联赛的比赛信息！

队员就是学生

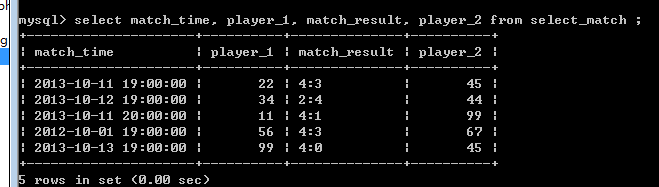
记录比赛信息：比赛表



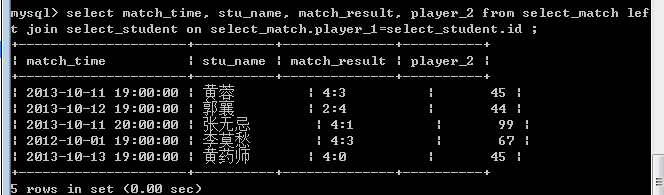
做一个比赛信息公告板，要求格式如下：

比赛时间 选手1名字 比赛结果 选手2名字

比赛时间 选手1ID 比赛结果 选手2ID



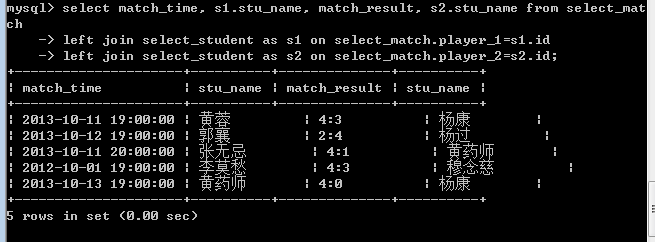
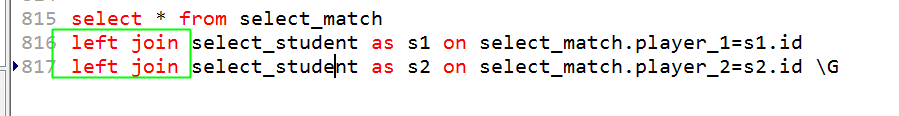
比赛时间 选手1名字 比赛结果 选手2ID



比赛时间 选手1名字 比赛结果 选手2名字

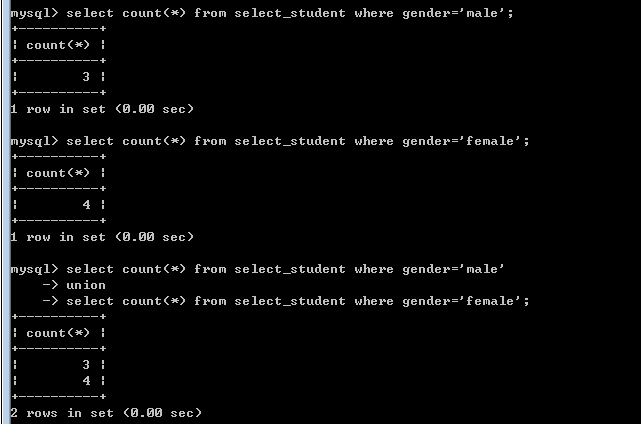
此时，可以对学生表再次连接！

出现一个表在一次查询时，被多次使用！注意，保证使用时没有歧义！为表起别名！



# union查询，联合查询

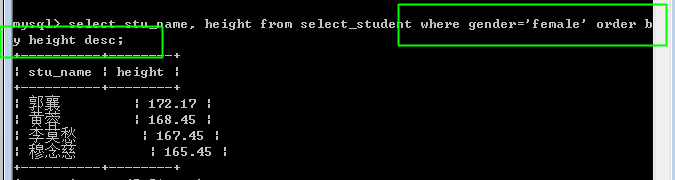
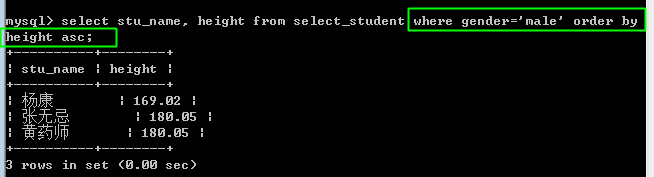
将多个查询的结果，并列到一个结果集合内！



此时，获得所有男生，按照身高升序排序。

获得所有的女生，按照身高降序排序。

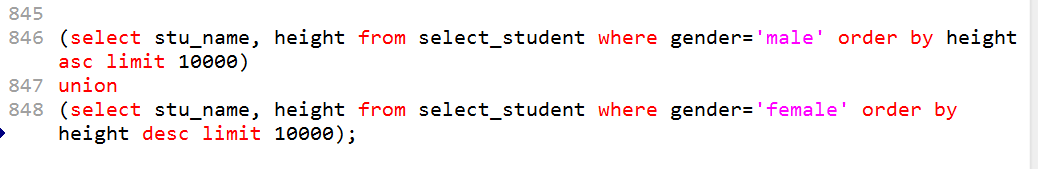
此时：



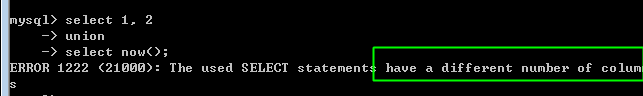
此时，将两个结果联合起来：

注意，在union是，如果子句中出现了 order by，则需要子句出现在小括号内！

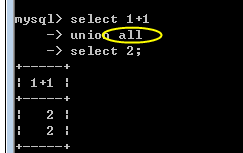
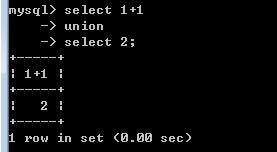
此时，子句的order by 也会在union的时候，会忽视掉！需要子句配合limit一起使用order by



union 的连接的两个子句，不要求实同表，只要求，列的数量相同！

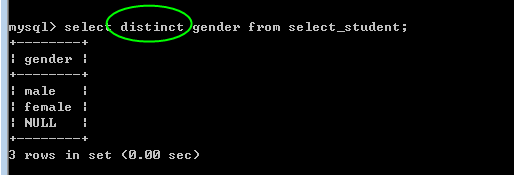
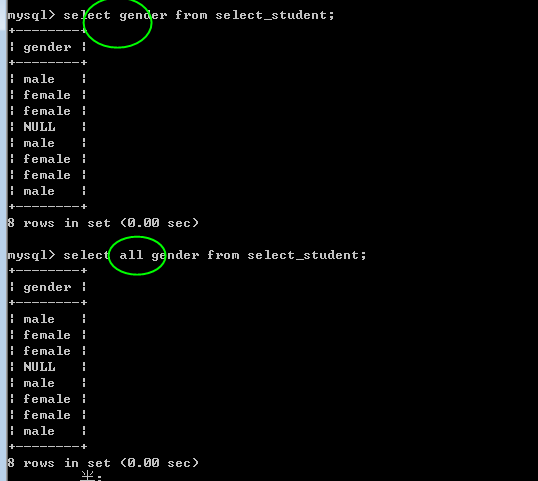


union会在联合时：主动去掉相同的记录：此时，可以使用 all关键字加以修正：



# select语句的选项

distinct，取消相同的记录

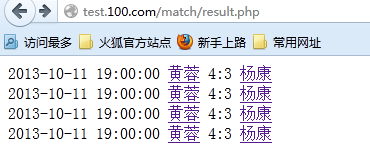


默认是 all，可以不写，表示所有的记录都出现！

# 作业

1连接查询练习：

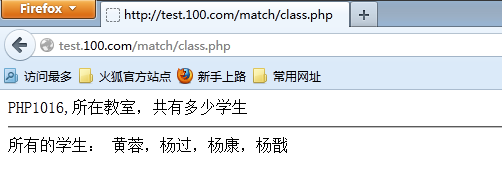
做乒乓球比赛公告页面：



点击进入，选手详细页：



接下来，进入班级信息页



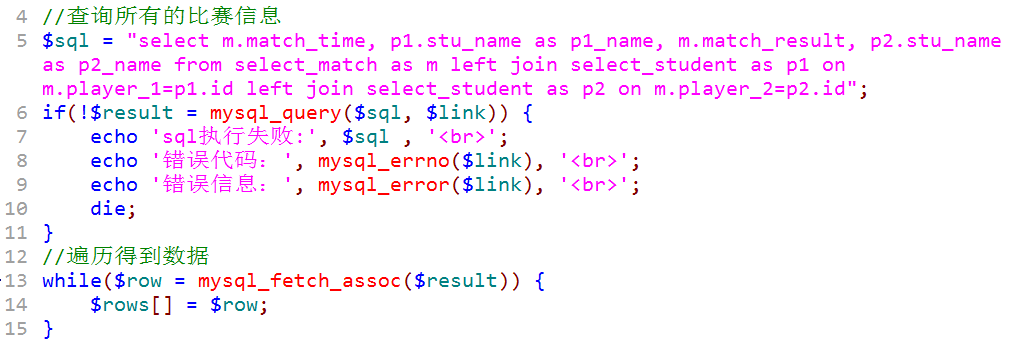
1. 设计，如果当前是篮球比赛呢？

设计，球队，球员，比赛之间的关系！

**第五天**

## 比赛列表

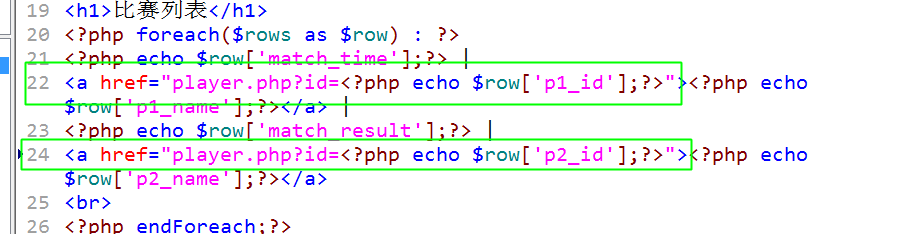
result.php



需要点击时，传递，参数，能够标识当前运动员的参数!

选择传递 ID 比较好！

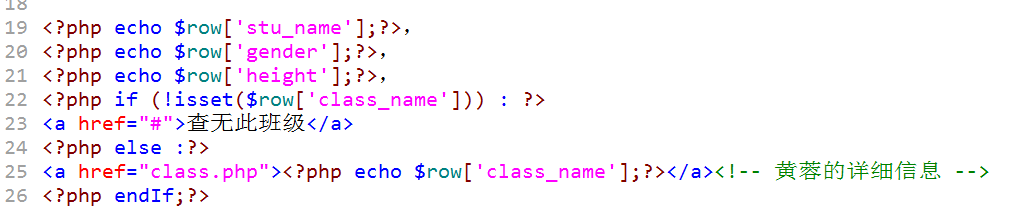
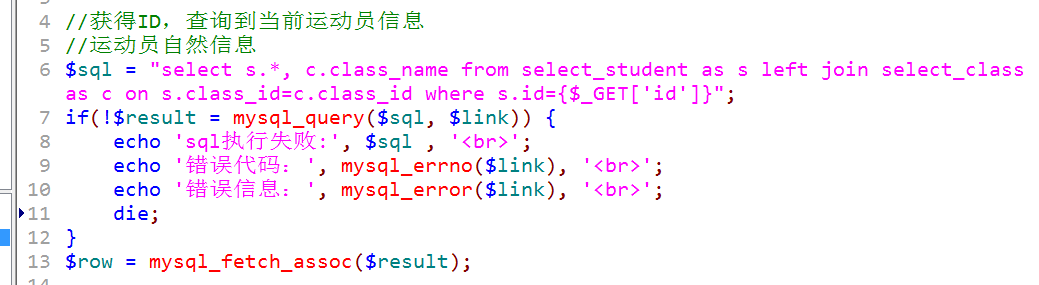
查询时，应该将两位运动员的ID查询出来！



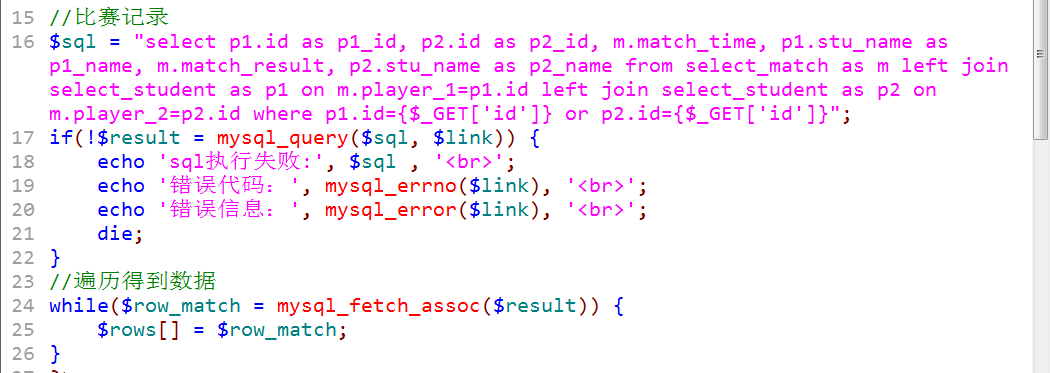
## 运动员信息简介

player.php

自然信息：



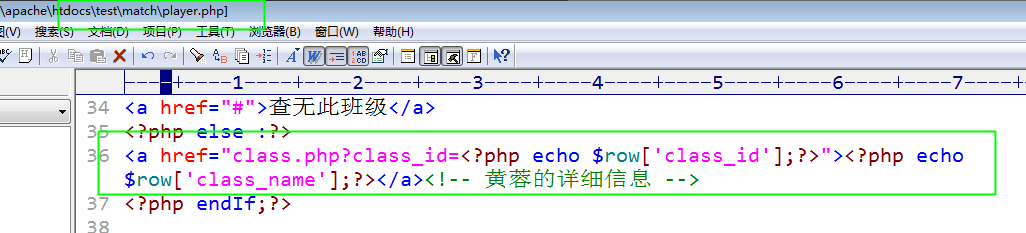
比赛信息



## 班级信息

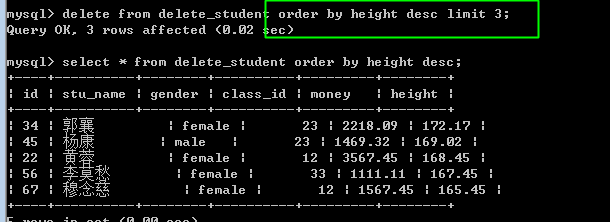
class.php

在链接上增加班级id参数：



# 删除数据，delete，truncate

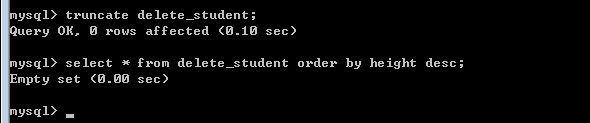
delete时，支持 order by 和 limit 来限制删除的数据记录!



delete from table\_name;

此时没有where子句，表示所有记录都删除！

如果清空表，此时可以独立使用 truncate 语句，完成清空表



注意与 delete from table的区别：

truncate 是删除表后重建表！

droptable,create table;

delete 是，得到每条记录，逐一删除！

导致的结果，效率上有差别，truncate效率高些！

# 更新，update

类似于删除，也可以使用 order by 和 limit 确定更新的记录！

# 视图，view

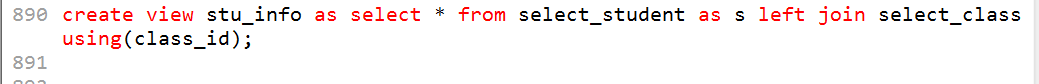
视图，是一张表，但是虚拟表！

是通过一条查询语句得到一个张虚拟表！

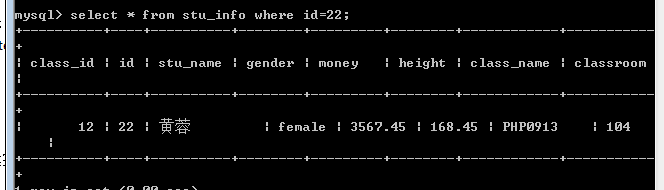
因此，认为视图，就是 select语句的结果！

语法：

create view 视图名字 AS 查询语句



创建完成后，就可以在视图内，完成查询工作了：



## 作用

可见，使用视图的功能之一，就是简化查询的业务逻辑！

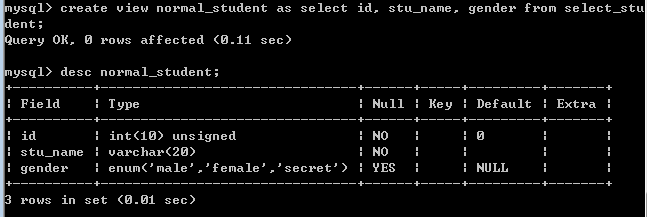
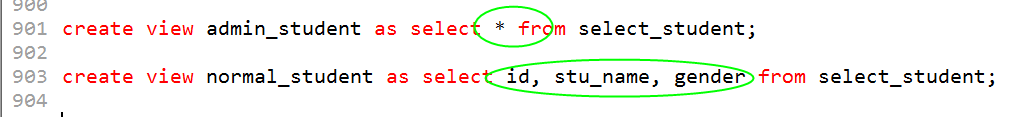
注意：视图内是不保存真实数据的！视图内，只有一条形成视图的 select 语句而已！

因此，每次从视图查询时，都需要利用视图，再在真实的表内将数据查询到！

类似一个 from型子查询！



功能之二：隐藏真实的表结构！



从而取得更大的兼容性

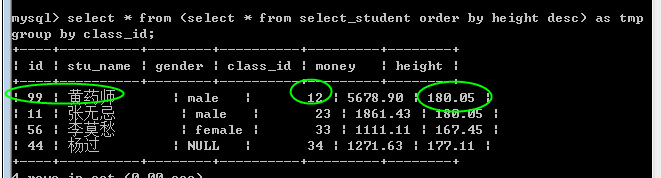
总结作用：1,简化逻辑，2,因此真实的结构（兼容性，限制用户的处理）

## 视图算法，执行方式

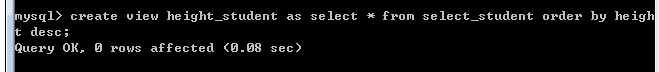
场景

取得每个班级最高的学生信息

使用 from型子查询，可以，但是使用同样逻辑的视图不行！



视图：



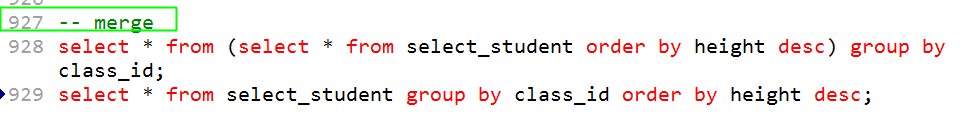
视图其实一共有三种执行方式：

merge，合并

temptable，临时表

undefined，未定义，就是默认的，mysql自己决定算法（从 merge，和temptable内选择）！

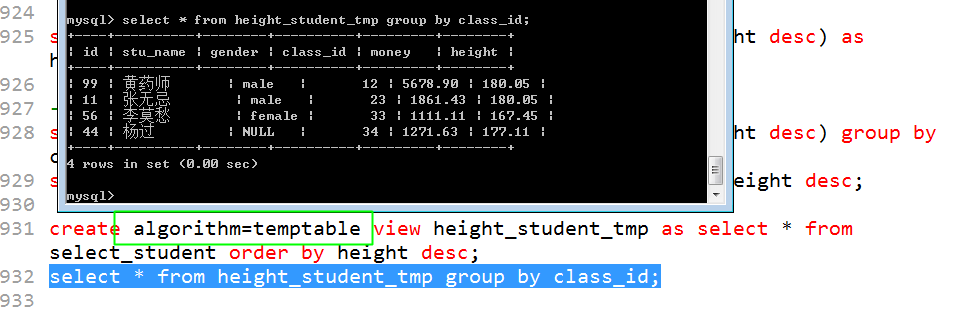
典型的都是选择的merge的：因此上面的视图查询：



如何解决？

可以选择决定视图的算法！在创建视图时，指定。

ALGORITHM=temptable即可



algorithm

用到比较少！

# 事务，transaction

一组 SQL 的集合，要不集体都执行成功，要不集体都失败，指的是，应该生成的影响退回到改组sql执行之前！

往往一个业务逻辑，是由多条语句组合完成！

## 语法

开启事务：start transaction可以简写成[begin]

记录下来，之后所执行的sql！（操作与结果）

提交：commit

如果所有的sql都执行成功，则提交。将sql的执行结果持久化到数据表内！

回滚：rollback

如果存在失败的sql，则需要回滚。将sql的执行结果，退回到事务开始之前

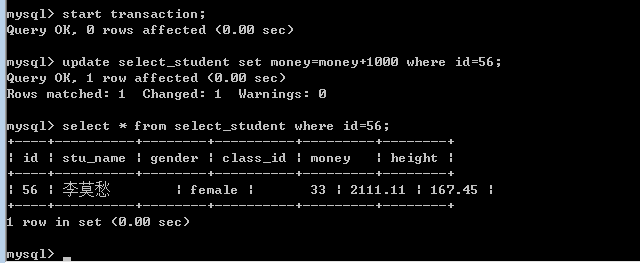
无论回滚还是提交，都会关闭该事务！（需要再次开启，才能使用）

事务，只针对当前的连接生效！

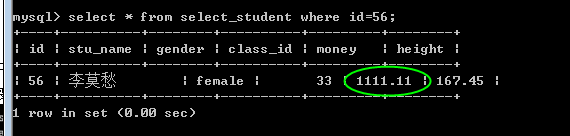
测试：

使用第一个链接A，开启事务后，执行一条update语句：

结果成功，数据已经变成修改之后！



此时没有提交通过其他连接B,查看，数据，是没有更改！

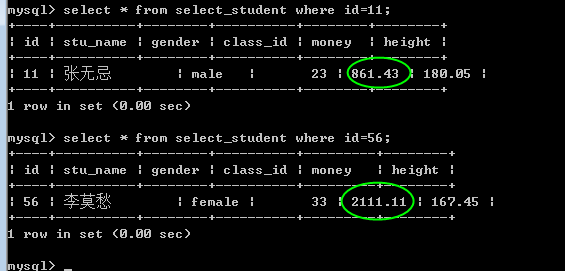


如果此时选择连接A 提交：

commit

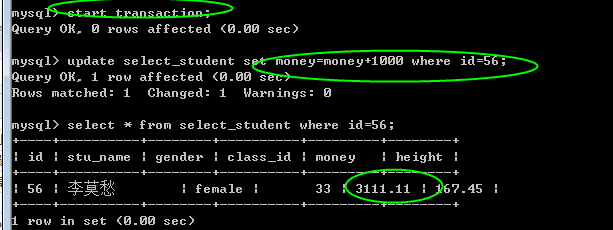


在其他的连接B，查看时，数据也发生了变化：

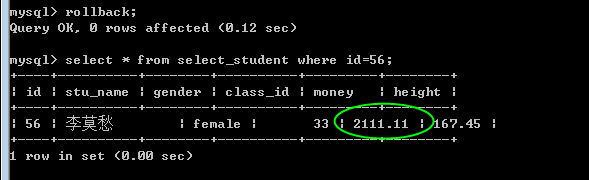


回滚：

一旦回滚，则事务内的所有的sql影响都会被撤销！



rollback



## 事务的基本原理，autocommit

提交，就会将结果持久化，不提交不会！

如果不开启事务，执行一条sql，马上会持久化数据！

可见，普通的执行，是立即提交！

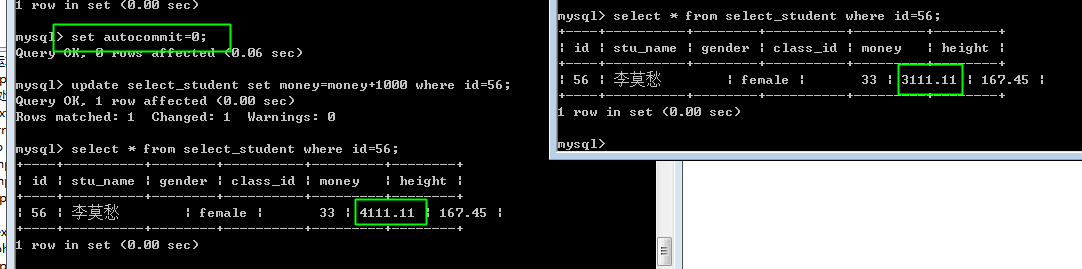
因为，默认的mysql对sql语句的执行是自动提交的！

开启事务其实就是，关闭了自动提交的功能！改成了commit执行手动提交！

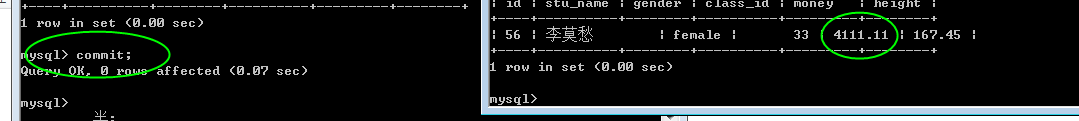
因此，可以通过简单的对是否自动提交加以设置，可以完成开启事务的目的！

自动提交的特征是保存在服务的一个叫做autocommit的一个变量内的！

使用 set 变量名=变量值的形式就可以完成修改：



此时提交：



此时的回滚或者提交，不会关闭事务！需要手动设置为开启！

注意：事务类似于外键约束，只被innodb引擎支持！

## 事务的特点，ACID

原子性，一致性，隔离型，持久性

原子性：事务是不可分割的。

一致性：保证数据在事务的执行周期内，是一致的！

隔离型：多个事务之间的干扰关系！隔离级别！

持久性：事务一旦被提交，就不可能在被回滚！

# 数据的备份

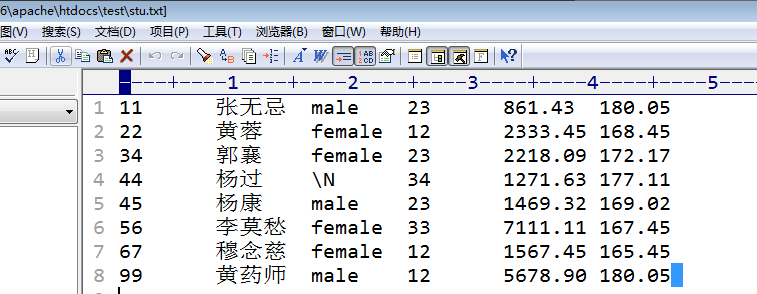
## 简单的数据备份（单表备份）

### 备份,select into outfile‘’

通过 select 语句将查询到的数据，以文本文件的形式，存储起来！

select into file

select 字段列表into outfile文件地址 from 表名 where 其他的select子句！



此时，相当于，将原本应该显示在命客户端的数据，写入到文件中！

对目标文件的要求，是目标文件不能存在！而且目标文件只要是普通的文本文件即可！

此时，对数据的生成格式，存在可以被设置的：

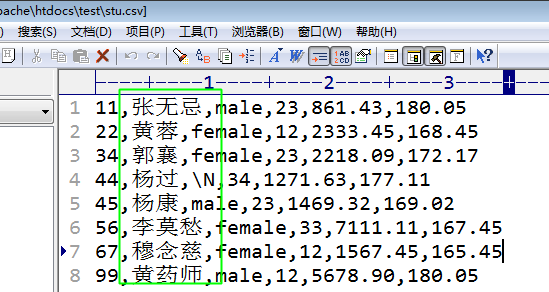
默认的字段之间的分隔符，使用制表符，而记录之间的分隔符，使用换行符！

但是，在数据存储时是可以被设置的：



### 还原load data infile‘’ into table table\_name

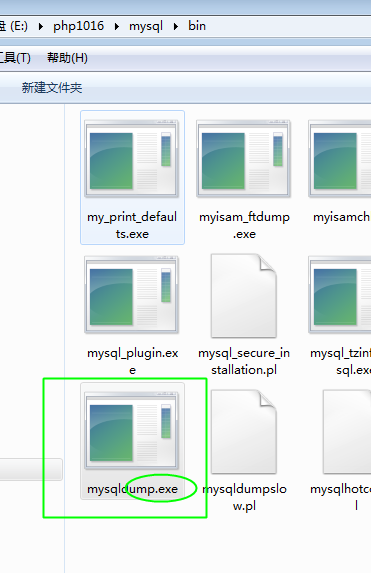
Load data infile filename into table\_name;



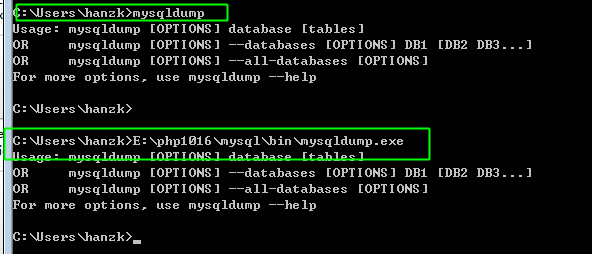
## 数据的导出工具，mysqldump

mysqldump，不是一个sql语句！

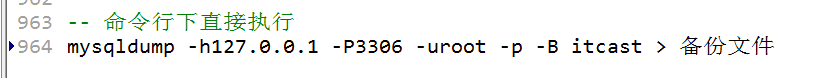
类似于mysql的一个mysql所带的一个工具！



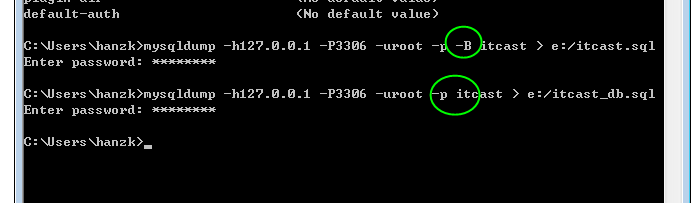
应该在命令行下直接执行，而不是在mysql登陆之后！



### 备份一个库



可以省略–B作为选项，表示不创建库，只备份库内的所有的表！

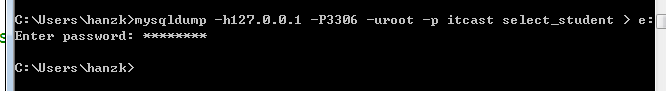


注意，上面>的语法，指的是输出重定向！

### 备份表

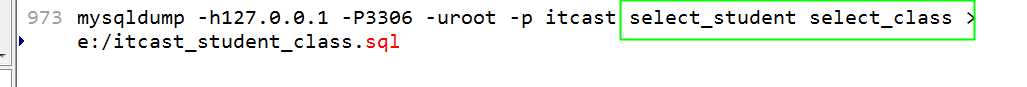
与备份库相比，多出了一个表名的值：

库名表名



可以一次备份多个表：

在表名部分写表名列表：



典型的备份都是：指的是将整个库或者整个表备份即可！

## 直接复制完成备份，额外的办法

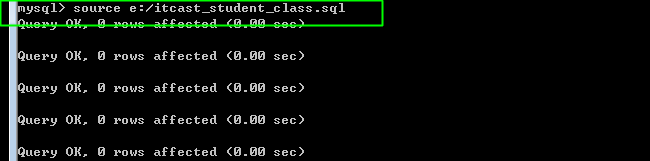
有时，在备份数据时，甚至可以直接选择将data目录，或者data目录中的相应子目录直接复制！

该行为不是每次都生效！

复制文件的方案，典型的是针对myisam格式的表发生的作用！

## 还原，执行外部文件内sql语句

可以在mysql客户单登陆后，使用source 指令，来强制执行一个文件内的sql语句！



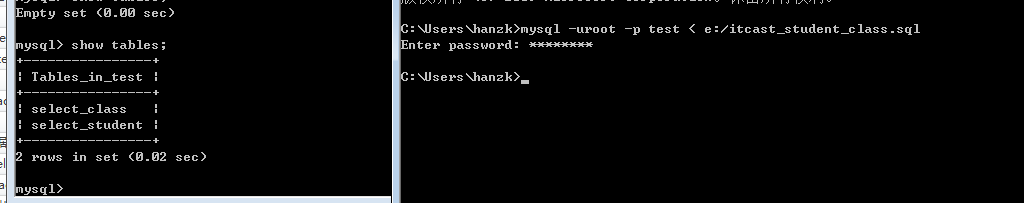
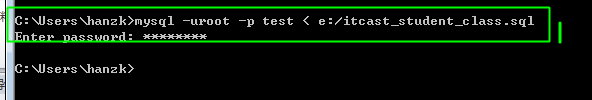
如果没有登陆可以选择采下面的形式：

mysql -h127.0.0.1 -P3306 -uroot -p 库名

登陆后直接选择数据库

mysql -h127.0.0.1 -P3306 -uroot -p 库名< e:/itcast\_student\_class.sql

表示，登陆后，选择数据库，并执行sql文件内的语句。



注意< ,称之输入重定向！

# 触发器,trigger

场景：

日志系统，记录对学生表有哪些操作！

解决的问题：

1. 得到每条学生记录被修改的时机，才能发出记录日志的动作！
2. 执行某段操作，需要得到当前处理的记录的信息！

触发器：一种编程设计！类似js的基于事件编程的程序设计的理念！可以在某个表的每条记录上，设置一个事件，从而对该表上的某些操作，加以监听！一旦所监听的行为出现，则会执行相应的代码。

记录 =button

（修改，删除，增加） =click

执行操作 =alert(‘Hello’);

以上的所有行为，都是采用sql完成的：

## 语法

create trigger 触发器名字触发条件，监听的内容，触发后执行的操作

CREATE TRIGGER *trigger\_nametrigger\_timetrigger\_event* ON *tbl\_name* FOR EACH ROW *trigger\_stmt*

其中，触发条件，事件。是由事件的时机，与事件的内容组成

时机：之前before，和之后after！

内容：增加insert，删除delete，修改update

因此，一共只有六种事件：

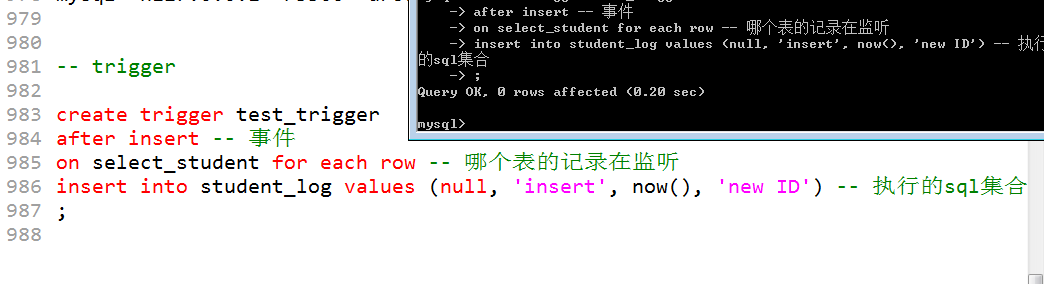
before insert before delete before update

after insert after delete after update

监听的主体是由表中的记录发出的

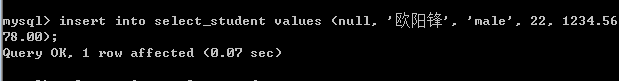
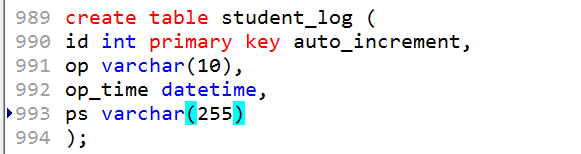
ontable\_name for each row

执行的操作，就是一段sql的集合！

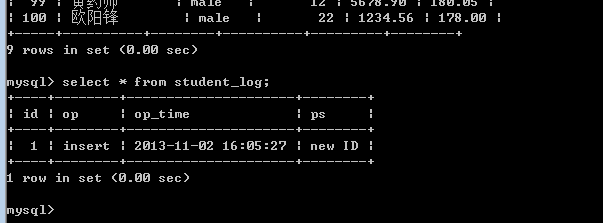


建立日志表：

执行插入：



日志表内的记录自动增加：



可见 insert into student\_log的触发程序，执行成功！

删除触发器

drop trigger 触发器名字

## 在触发程序中得到当前触发的记录信息

有，两个！

new（新的），old（旧的）

new，和old，都表示触发程序的记录！

new：新的记录。old：旧的记录！

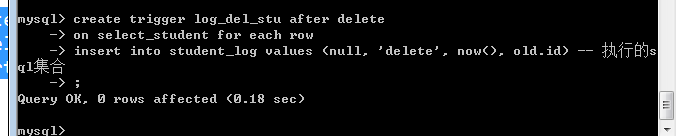
取决于当前操作（intser，update，delete）去使用其中某个：

insert，增加记录，没有旧记录，只有new关键字可以使用

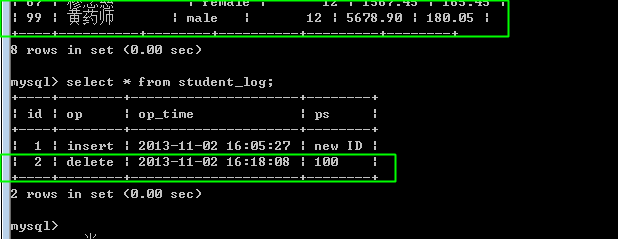
delete，删除记录，没有新纪录，只有old可用！

update，更新，既有新纪录，也有旧记录，更新前是旧记录，而更新后是新纪录！因此可以 new和 old

记录，当前学生被删除之后，记录日志，要求记录学生的id。



测试，删除记录：



此时，留意一下触发器，与具体的语法的执行时机：

当：insert into table操作！

判断，是否有before insert 触发器！有则执行触发程序！

真正执行 insert into

判断，是否有after insert 触发器！有则执行触发程序！

更新日志：

记录更新日志，要求是，只对某部分同学完成更新日志！

只记录，身高超过175学生的更新记录！记录学生id和修改前的身高与之后的身高

需要额外的增加条件判断！

逻辑分支语句：

if 条件 then

语句体

else if 条件 then

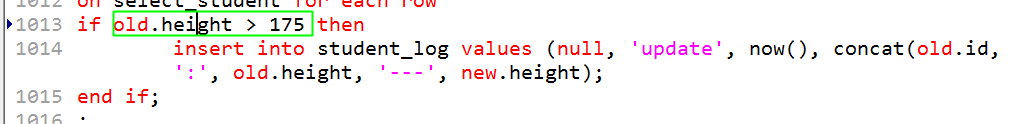
语句体

….

else

语句体

end if;



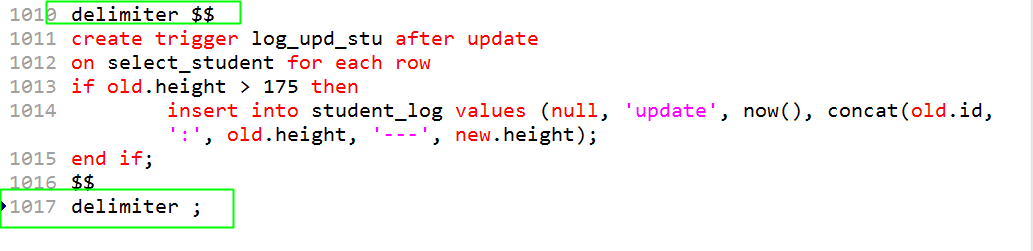
sql语句的结束符问题

可以修改最外层的语句结束符达到目的！

delimiter $$

将语句结束修改成 $$

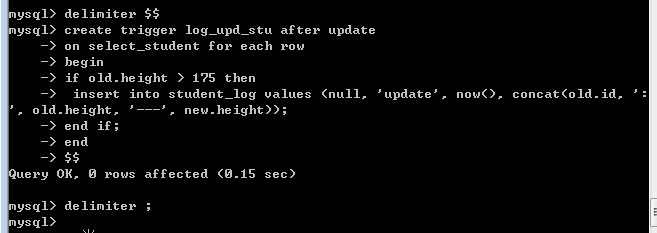
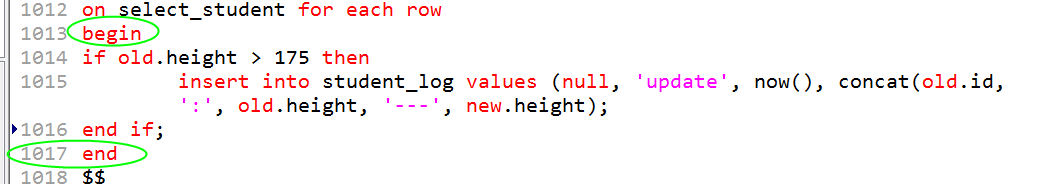
记住用完后要再修改回来！



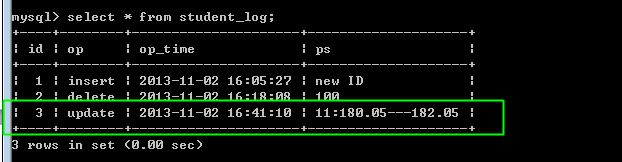
如果触发程序由多条语句组成块。此时就需要使用

begin

end 将语句块包裹！



测试：



注意，关于触发器：

1. 一个表上的一个事件只能有一个触发器，如果需要，只能将原始的去掉，新增！
2. 只要事件发生，触发程序就可能执行！一条语句可能触发多个触发程序！

例如：

insert into on duplicate key update

before insert trigger, insert 操作失败before update trigger, update操作, after update

before insert trigger, insert 操作成功after insert trigger

php项目中用的不多

# SQL编程介绍

一：SQL也是一门程序设计语言！可以用其编程！

二：基本常规的编程要素

变量，运算符，表达式，流程控制，函数

## 变量

典型的，字段名就是变量名，字段就是变量！

特殊的变量，例如系统内置变量（character\_set\_xxx, autocommit）

变量的典型操作：

赋值，取得值！

### 用户自定义变量

用户自定义变量需要使用 @作为变量名的前缀，用于区分是否是系统内置变量！

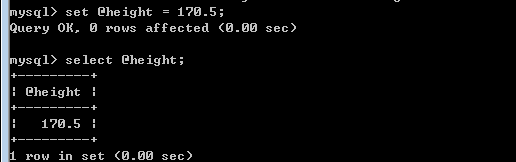
set 语句可以完成对变量的设置！

set 变量名=变量值！



怎么取得？

采用select 语句即可！



## 编程的的实现

一共有三种编程方式：

触发器，存储函数（自定义函数），存储过程

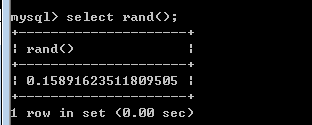
## 函数

### 内置函数

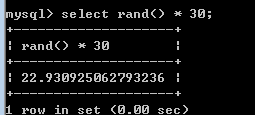
对于mysql来讲，有内置函数！



rand(),生成随机数的函数，得到0-1之间的随机数



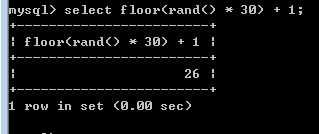
典型的为了得到某个范围内的随机数，需要\* N倍：



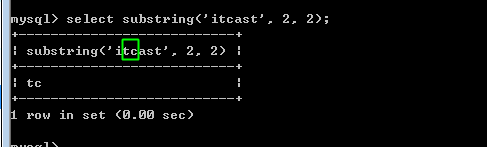
需要配合上取整使用，达到取得随机整数的目的：

floor()向下取整：

一到三十之间的随机数如下：

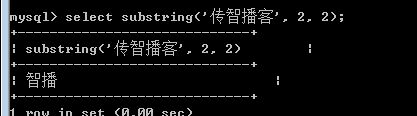


substring(字符串，位置，长度),截取字符串函数



注意，从1开始的下标!

而且是以为字符为单位：

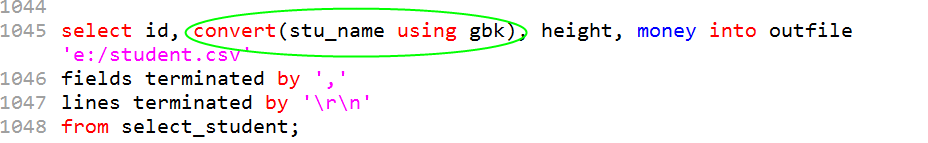


char\_length()字符数量

length()字节长度



convert(字符串 using 字符集)字符转换到相应的目标字符集上



### 自定义函数

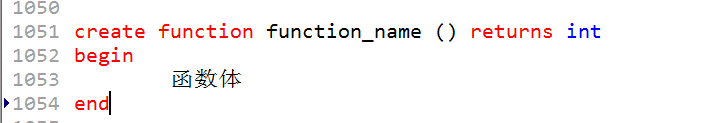
函数的要素：

函数名，函数体（返回值），函数参数

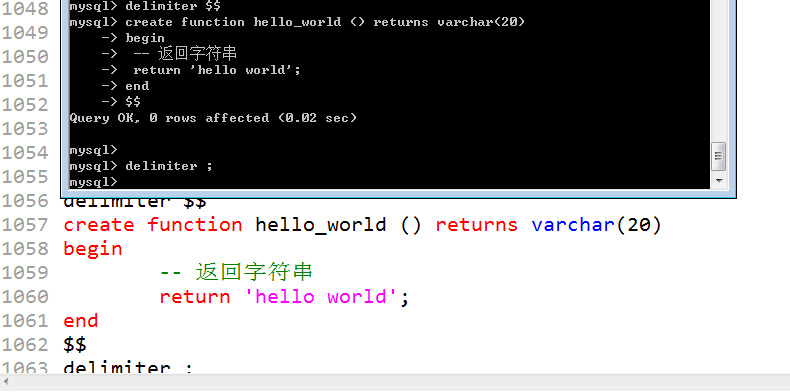
函数的基本使用：

声明，调用

#### 声明

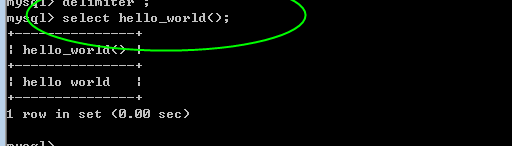


注意多条语句，修改结束符



#### 调用

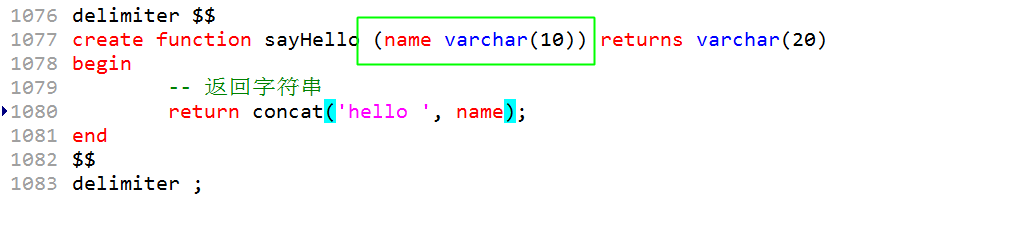
类似于系统函数即可！



注意，函数是存储在某个数据库内，因此与库是相关的！



#### 参数



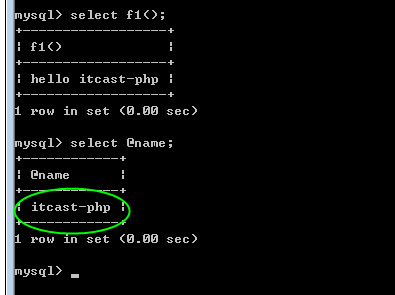
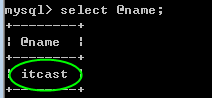
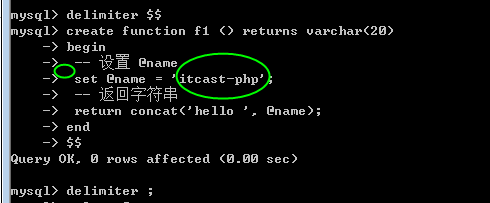
注意，参数也是需要有类型之分

而且，[不用使用@。原因是](mailto:不用使用@。原因是)函数内的变量是不用区分的！参数是一个局部变量！



#### 函数局部变量与函数外全局变量的作用域问题

作用域是重叠的！在函数内，可以访问到函数外所定义的全局变量！

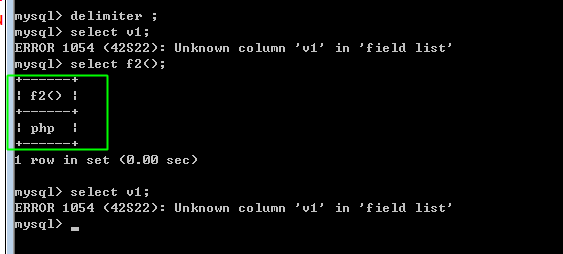
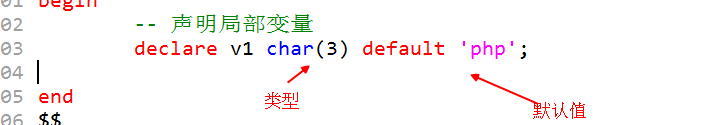


#### 局部变量

在函数内的变量！

参数是！

此时在函数内，使用关键字 declare 来声明局部变量（js中类似var的作用）！



#### 流程控制

分支

if 条件 then 语句

else if 条件 then 语句

else 语句

end if;

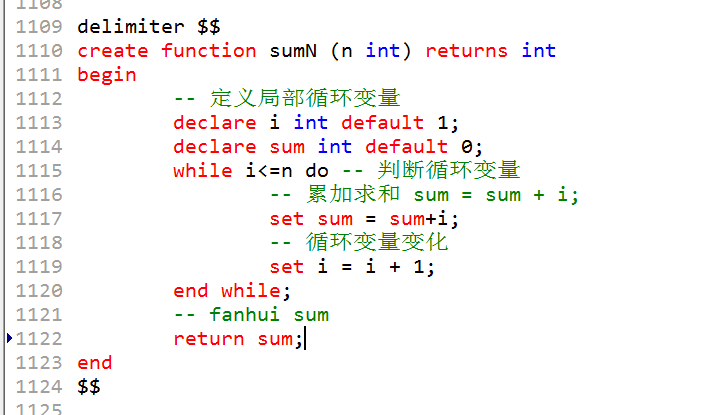
循环

while 条件 do

循环体

end while;

示例：求1-N之和：



练习：拼凑随机的名字！

getName(2|3|4)

参数为名字的长度，而返回值为名字！

业务逻辑：

取得姓，在某个集合内获取的！

再取的名字（根据字数，取得的数量不同）。

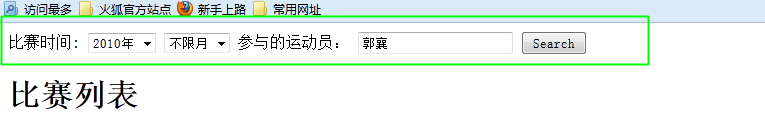
php层面不常用

# 作业

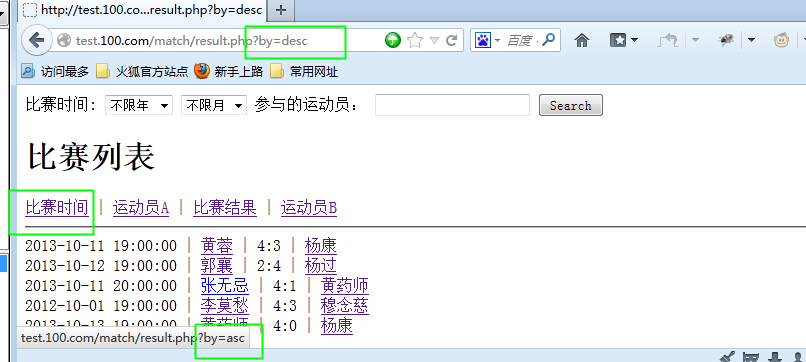
完成以下功能：

1，数据的筛选，搜索功能

可以主动增加条件



2，列表字段排序



注意，不用考虑太多关联的问题！

3，大家试着总结，视图的创建，删除，修改的语法！

4，完成一个转账系统，练习事务

5，要求大家完成一个mysql自定义函数，可以求N的阶乘！

N! = 1\*2\*3\*4\*N；（N! = N\*(N-1)!,）

（要求，判断N不能超过30，返回结果）

（测试，是否可以用递归,先测试一个下php的递归！）

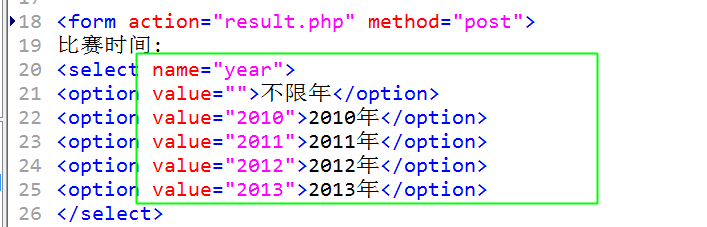
# 作业说明

## 搜索功能

result.php

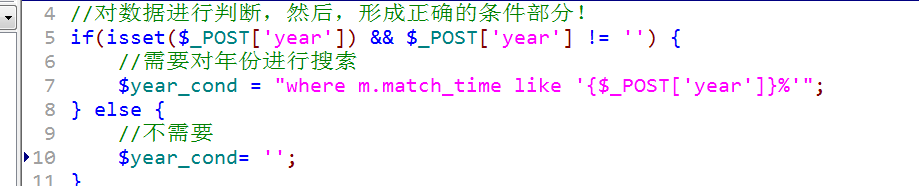
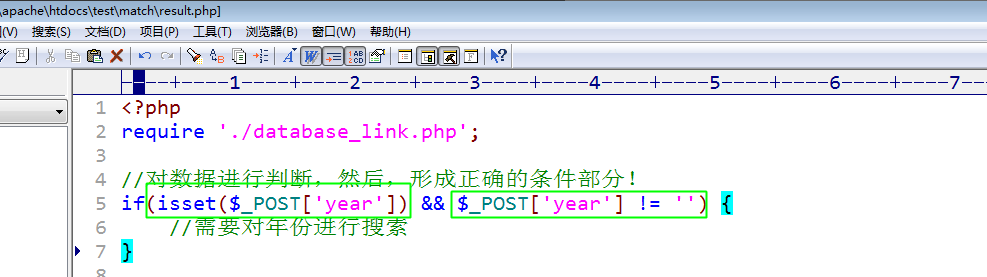
形成年份列表

下拉 select

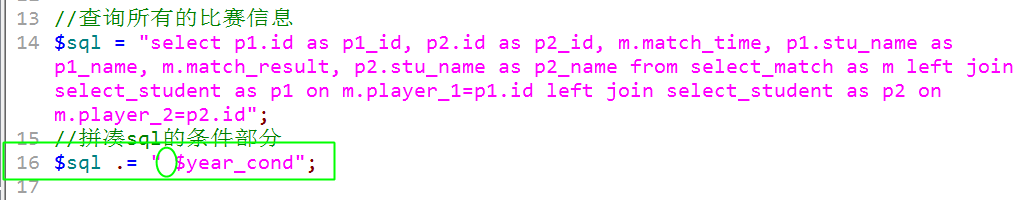


点击提交后，获得用户所填写的年份，然后对条件进行判断！





拼凑到sql语句中！

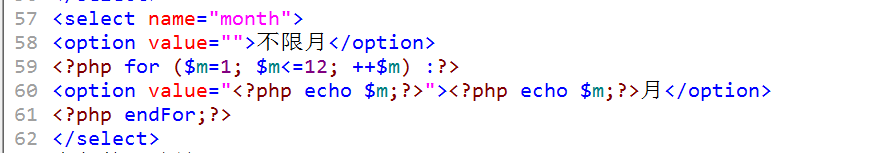


再对月份进行判断：

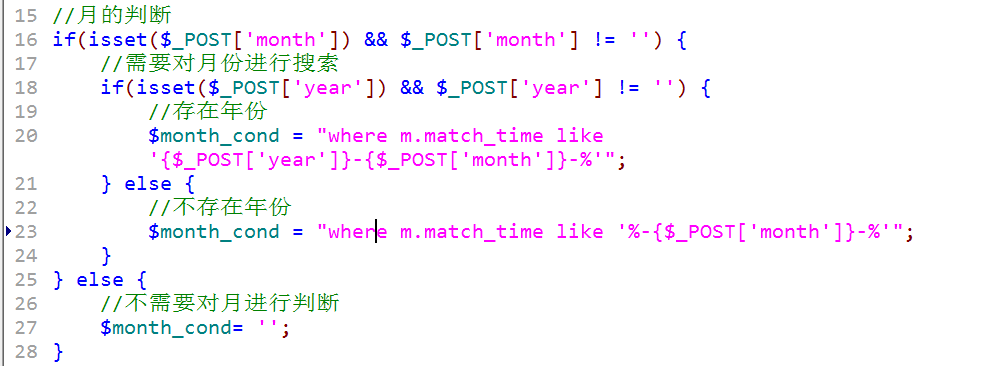
考虑，是否选择了月份

选择了月份，还要考虑是否选择了年份

形成，月份列表：



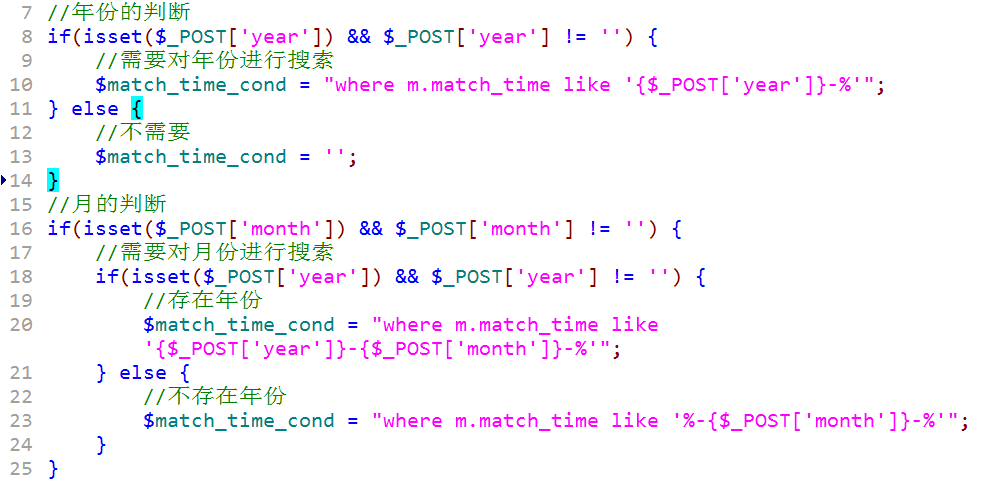
同理，需要对月份进行判断：



此时条件由多部分组成，考虑条件之间的关联问题！

典型：

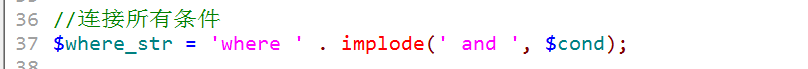
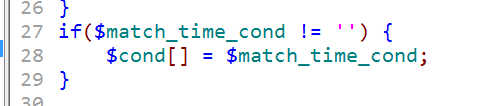
如果是一个字段上的条件，应该整理到一起！



增加一个名字的条件：

不是match\_time字段

典型的，如果是多个字段作为条件，会先将所有的条件，整理到一个数组内，再将数组内的元素连接起来即可！



# SQL编程

## 存储过程，procedure

与函数类型，都是一个功能模块的即可代码！

相对于函数，函数，倾向于某个功能点。

而过程，倾向于某个业务逻辑的整体实现！

功能点：自由得到用户名，函数

业务逻辑：向select\_student表内，插入1W条测试数据！，过程

### 语法

函数：create function

过程：createprocedure

创建

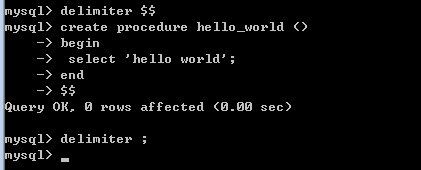
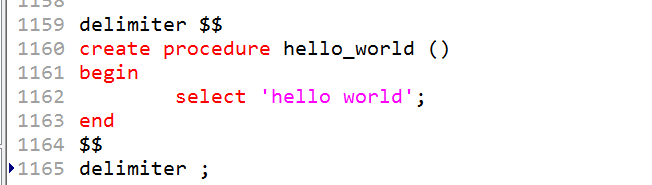
create procedure 过程名 (参数列表)

begin

过程体，执行代码的集合

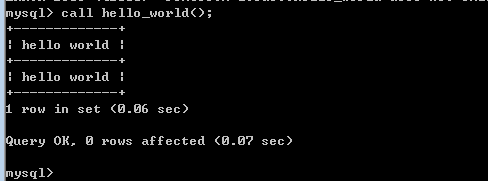
end

注意没有返回值



调用

不能直接调用！需要使用 call 关键字调用存储过程：



过程，是一个独立的业务逻辑，不能出现在表达式内。

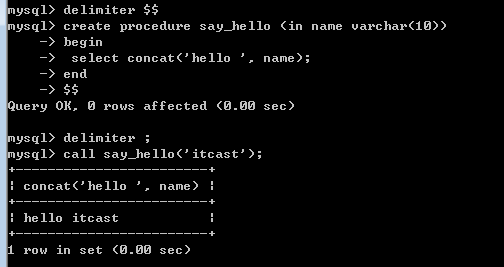
## 过程的参数

功能类似函数的参数，也是在运行时传递数据。

但是，参数是分成三种类型：

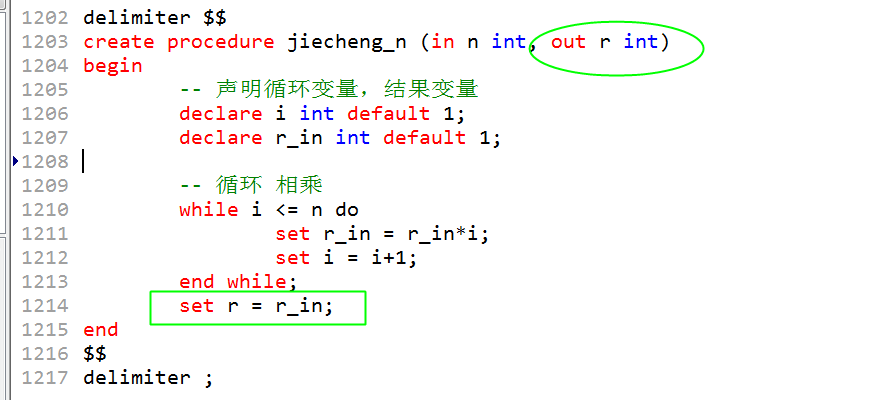
### 输入，in

调用过程时，给过程传递数据，就是输入参数

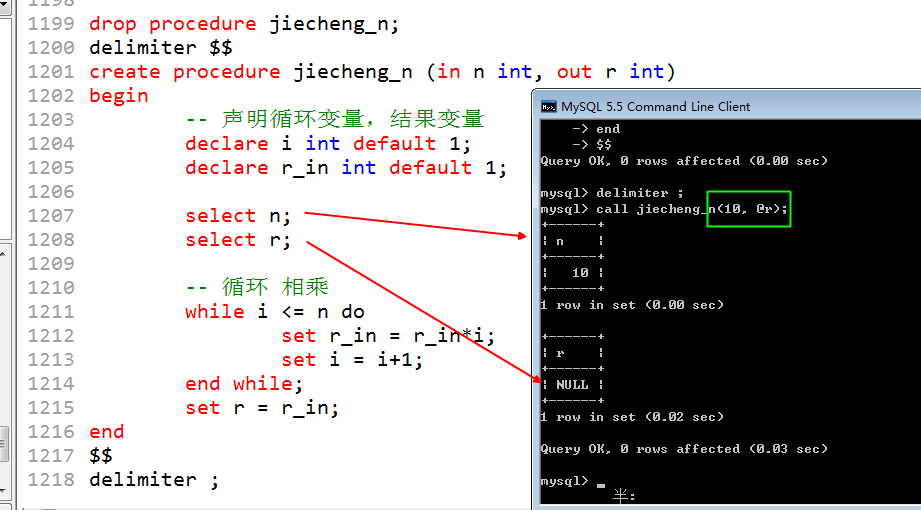


### 输出，out

过程之后后，可以利用该参数，将数据传递出来！



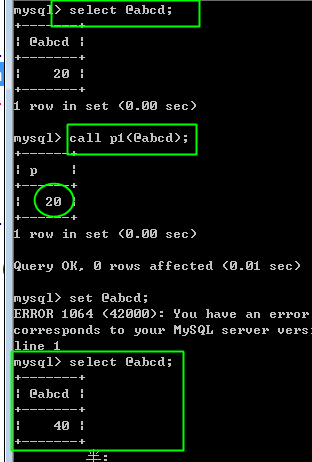
参数 r变量的值，要输出给调用的变量。调用时，要保证，相应位置的实参，一定要是一个变量才可以！



### 输入输出，inout

同时完成输入和输出功能。既可以输入数据，也可以输出数据！

利用：in，out，inout来声明参数，最像php中引用传递！



典型，可以利用过程，增加测试数据：

1W条学生数据！

## 过程与函数的区别：

函数有返回值，而过程没有

函数应该表达式内，而过程应该独立调用（不能出现在表达式内）！

参数上有区别，函数只有一类参数，只负责输入是参数数据！而过程，in，out，inout之分！

本质上，过程一整个业务逻辑。函数是一个特定功能点！

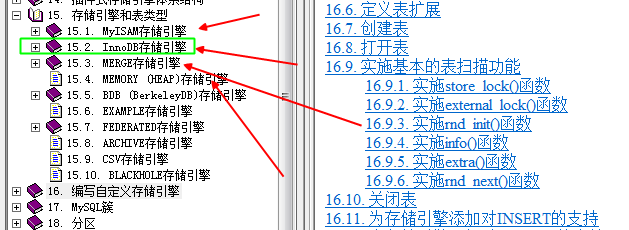
# 表的存储引擎

innobd，是一种mysql支持的存储引擎！

什么是存储引擎？

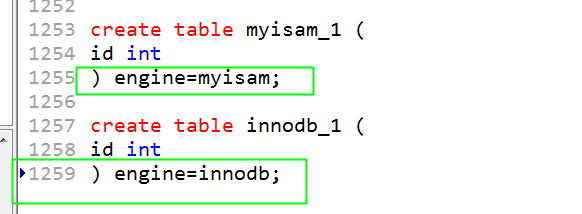
指的是，数据在服务器上的存储格式！

典型的mysql支持多种引擎：



不同的存储引擎意味着，存储方式的不同：

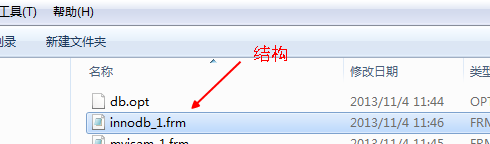
例如：innodb，与myisam：



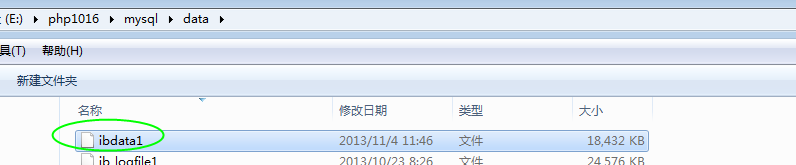
myisam三个文件



而innodb只有一个



那数据和索引在哪里？所有innodb的表的数据和索引都是统一管理的！称之为innodb数据空间：



对于mysql，常用的，innodb和myisam！

区别：

1. 文件的分配不一致
2. 功能上将：innodb，事务安全型存储引擎，支持外键，事务等！
3. 功能上：myisam，对查询有一定的额外优化！

选择：

1. 先参考功能。
2. 在参考性能（看增删改查哪种操作多）。增加和查询较多使用myisam。更新和删除较多使用innodb。

# root 密码忘了咋办

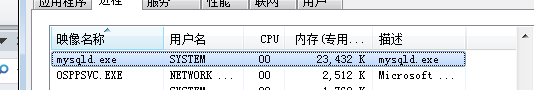
将其他的 user表的三个文件，拷贝到你的mysql目录下！

更新 root 用户 Password字段即可！

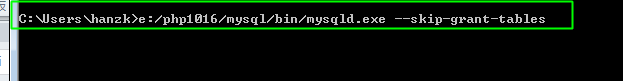
面临的问题是，没密码，登陆不上，没法更新

mysqld服务器程序，有一个选项，跳过权限认证选项！客户端登陆时随便！

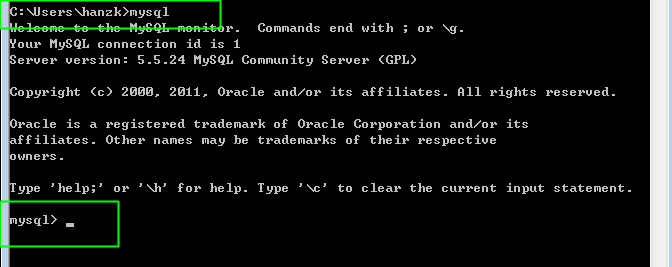
1. 重启开启mysqld服务。以跳过权限认证的模式！



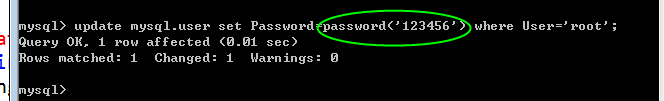
命令行执行mysqld命令开启



1. 直接登陆不用用户名密码



1. 更新mysql.user表root用户的password字段！



1. 重新开启mysqld

