昨日回顾

数据库

创建一个数据库: create database 数据库名;

修改数据库: alter database 数据库名

删除数据库: drop database 数据库名;

显示数据库: show databases;

显示创建数据库的语句: show create database 数据库;

数据表

创建一个表: create table 表名 (字段 字段类型 ,…);

删除一个表: drop table 表名;

显示所有表: show tables;

显示创建表: show create table 表名;

修改表: 表选项,表名,表字段

重命名表名:rename table 旧表名 to 新表名;

表字段: 增加表字段 alter table 表名 add 字段名 类型;

删除字段: alter table 表名 drop 字段名;

重命名字段: alter table 表名 change 旧名字 新名字 字段类型

修改字段类型: alter table 表名 modify 字段名 类型

表选项: alter table 表名 charset | engine

数据

插入数据: insert into 表名(字段列表) values (值列表),(值列表);

查询数据: select \*|字段列表 from 表名 [where 条件判断];

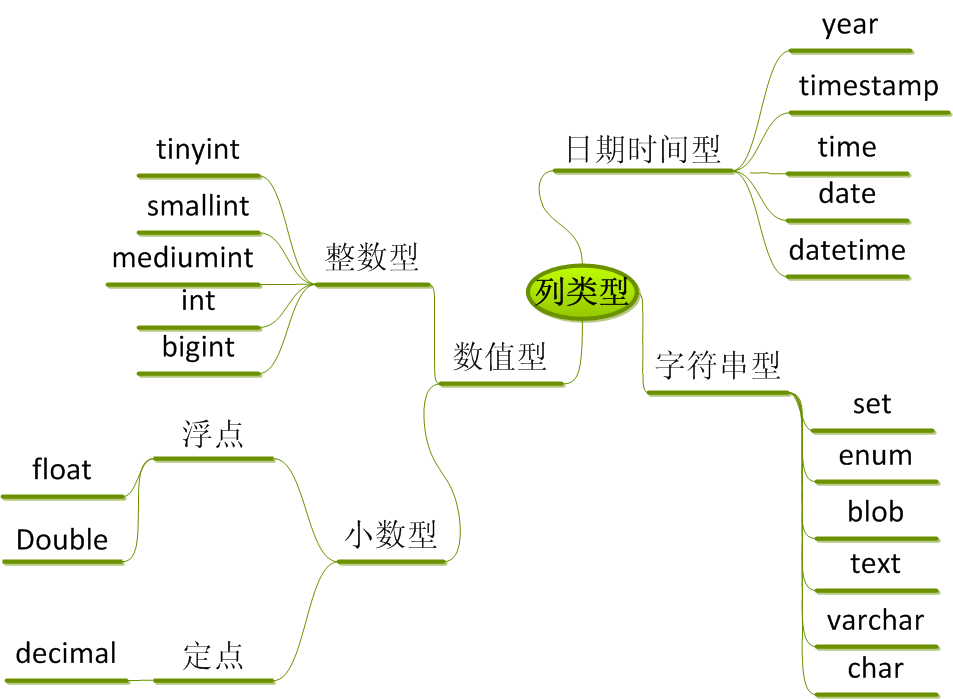
删除数据: delete from 表名 [where 条件判断]

更新数据: update 表名 set 字段名 = 值 [where 条件判断]

# 第9讲 数值类型

## 数据类型介绍

数值型（整型及浮点型）、字符串类型、日期时间型



## 整数



### 类型

tiny微小

small 小

medium 中型

这么多数据类型,我们以够用就可以!!! 数据类型越小,运行的效率越高

### M

表示最小显示宽度。超出显示宽度，不超出类型的表示范围，可以存储。不足显示宽度，左侧使用空格进行填充。

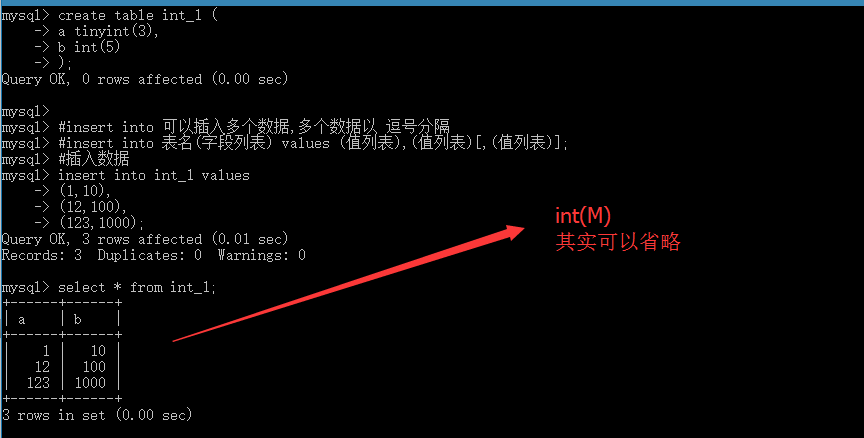
int(M)

如果我设置最小显示宽度为 3

1

12

123



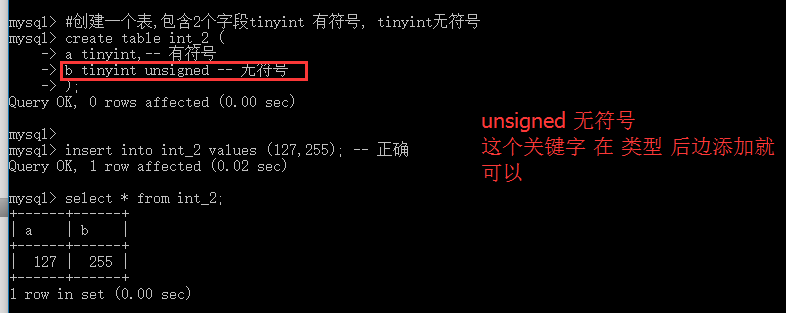
### unsigned无符号属性

整形默认情况下是区分正负.(最高位就是符号位)

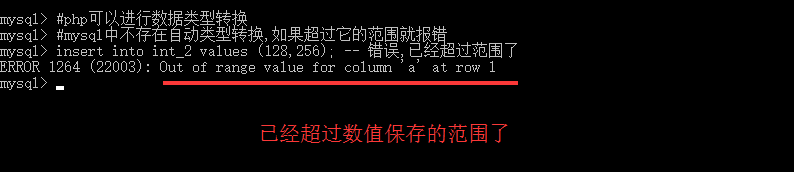
无符号都是正数,也就说 原来最高位为1的时候表示 负数,

现在这个最高位不表示符号位了表示值,这个值都是正数

* 有符号与无符号的区别在于：最高位是否为符号位。
* 整型添加unsigned属性后，只能存储无符号数。
* 无符号表示范围是同类型有符号数正数部分的二倍



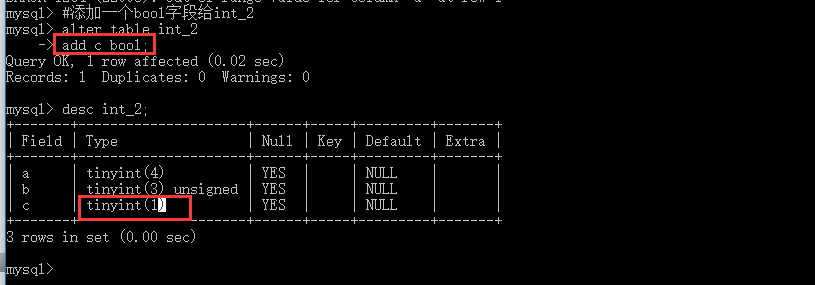
这5中数据类型,大家一定要注意 范围,超过范围就插入失败



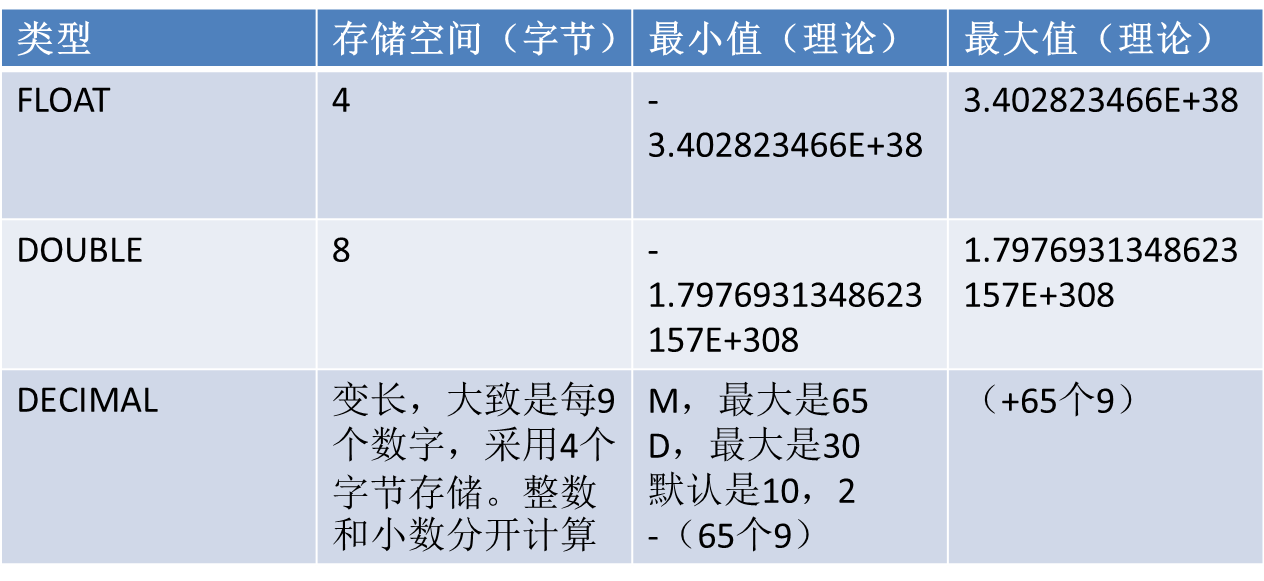
### 注意事项

①究竟选择哪种类型？原则是：够用就行了！因为占用的空间越小，效率就越高

②在数据库中没有发现bool型,其实我们也可以用tinyint 表示bool型;

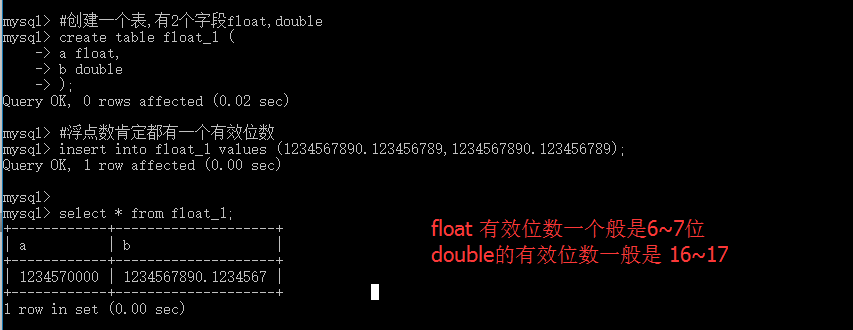


## 小数

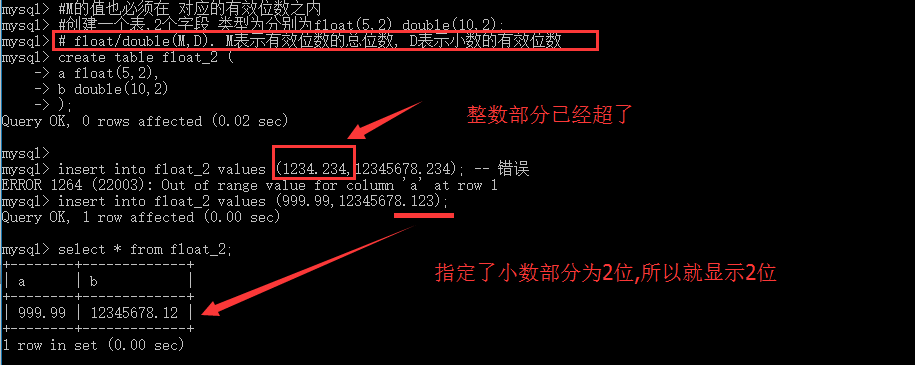


### 浮点数

float 和double占的字节数不一样,我们根据具体的需求来选择;



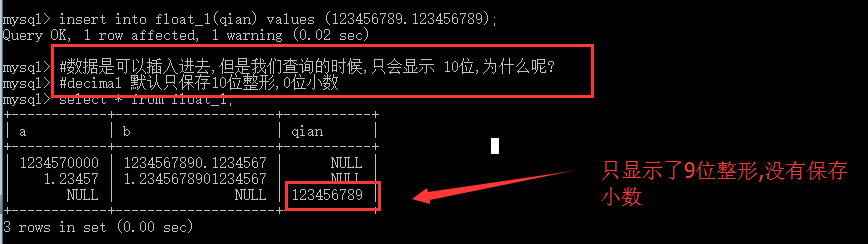
float/double 也存在(M,D)的情况



### 定点数

decimal (也叫做货币类型)

定点数在存储是时候,会把整数部分和小数部分 分开存在.



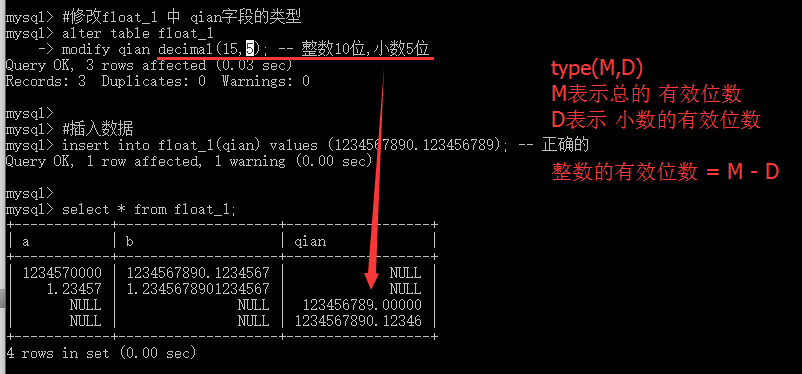
### M,D

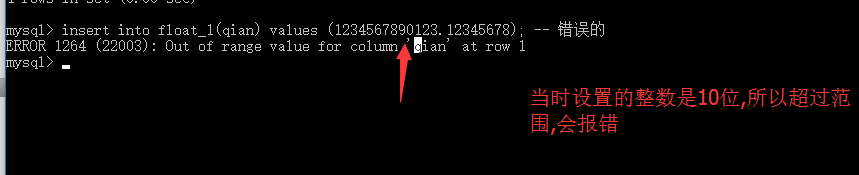
float,double,decimal后面的M表示总共的宽度的位数。D表示小数的位数。

是宽度,我之前说错了,我错了,我错了!!

type(M,D);

type(15,5);





* decimal最多可以精确存储30位小数。
* 精确存储的原理：整数部分与小数部分分开存储，每9位连续数字占据4个字节空间，而且存储每一位数字的值。通常用来存储货币。所以又称货币型

第一个概念: 数值范围. 就是浮点数的最小值和最大值.表示他的值很大

第二个概念: 有效位数.

1.234567890123456789 肯定在数值范围内,但是 小数部分只能最多转换成二进制存64bit ,所以会造成小数部分位数的丢失. 有效位数是1.123456789012345 最后这个5后边的数值就丢失了!!!

第三个概念type(M,D): 首先要确保 这个值在 有效范围 和 有效位数之内.

举例来说明:

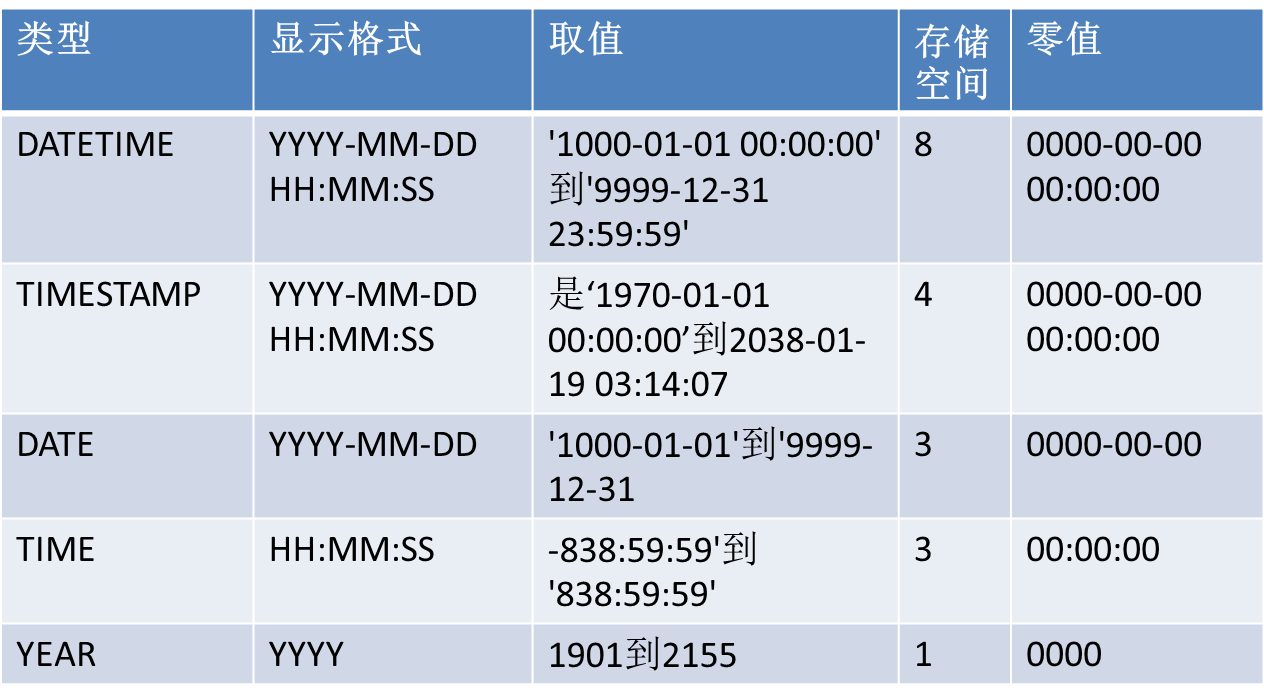
float: 范围是3.4e38

有效位数是 6

float(5,2): 整数部分占3位(5-2),小数部分占2位 .它的数值范围是-999.99 ,999.99

这个位表示 数值的宽度

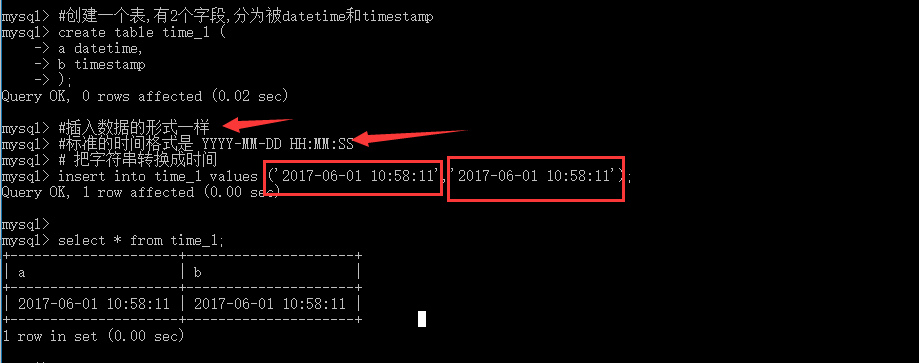
# 第10讲 日期时间类型



## Datetime和timestamp

datetime和timestamp的显示形式和插入形式基本上是一样的！

datetime和timestamp的存储方式和日期范围不一样！在timestamp存储是整型，但是显示和插入的形式和datetime是一样的，都是年月日时分秒！





注意事项:

一般在实际开发过程中,我们表中只创建一个时间戳的字段,这个字段如果不插入数据,会自动的填充这条数据插入的时候的时间

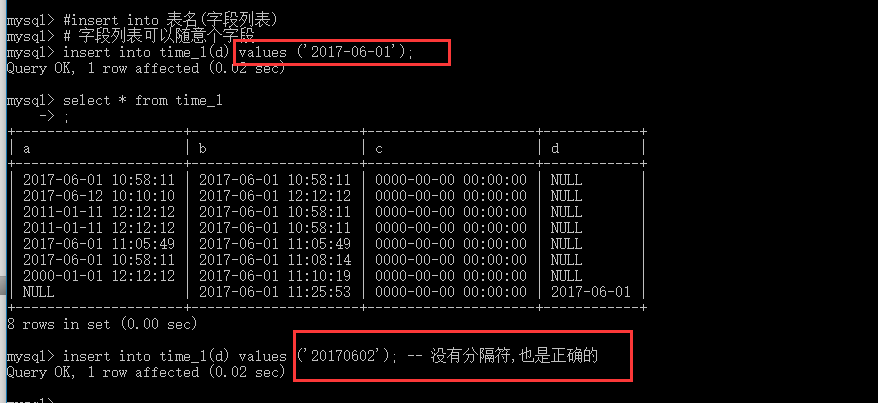
如果创建多个timestamp 字段,肯定会有一个字段来记录时间戳.其他的字段如果不插入数据会使用默认值0000-00-00 00:00:00



## date

date可以理解成为 datetime 的date部分

date就有年月日



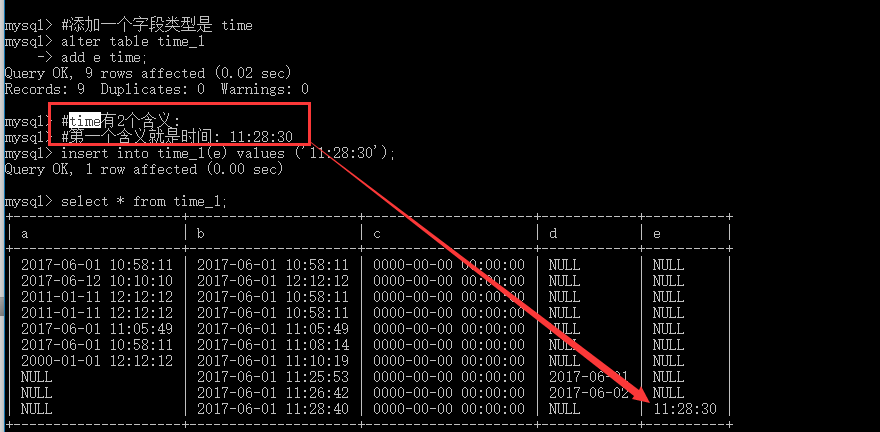
## time

time有2层含义.一层含义表示 当前的时间 小时:分钟:秒数

另外一层含义可以表示 时间间隔 11:22:22上的课,现在时间是 11:33:22

时间间隔是: 00:11:00

* 一般表示时间，时分秒；



* 可以表示时间间隔

插入间隔的时候可以采用 “D HH:MM:SS” D表示天数,这个天数也是有一个范围的

注意: 时间必须采用标准的时间



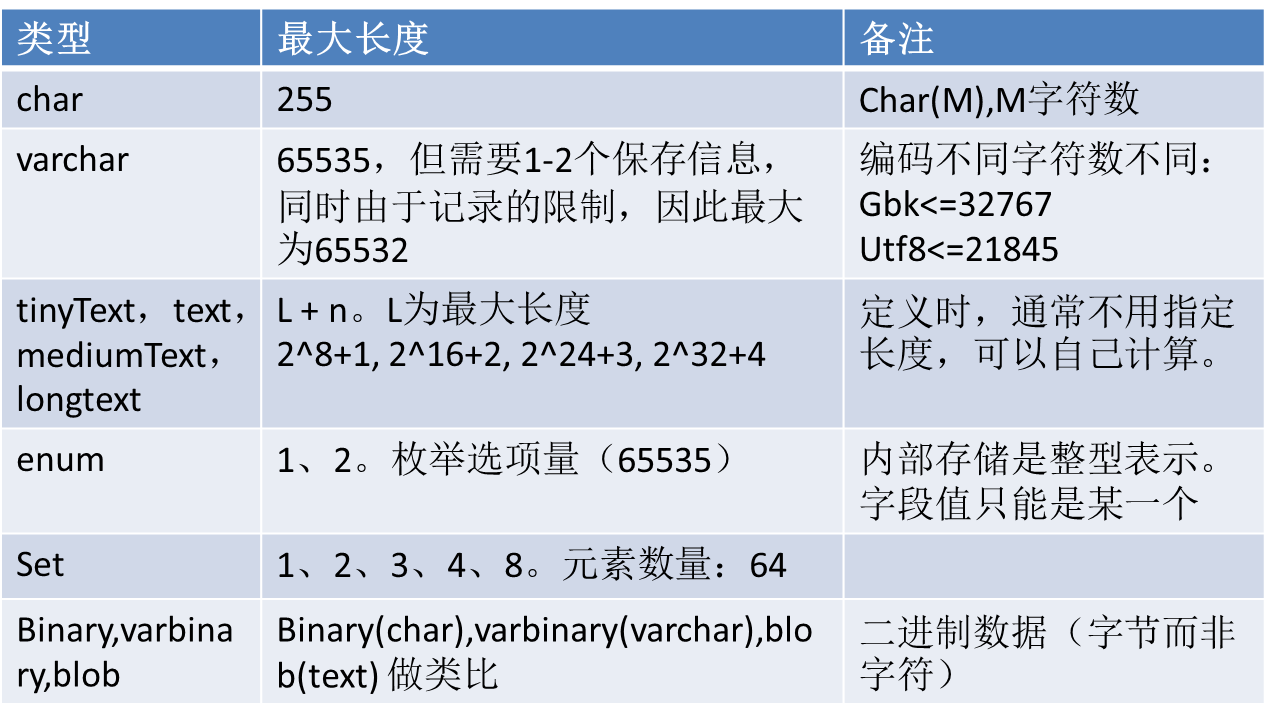
## year

有两位格式及4位格式。

两位（00-69,2000--2069）（70-99 1970--1999） 表示范围1970---2069

四位表示范围：1901-2155

# 第11讲 字符串类型



## char和varchar

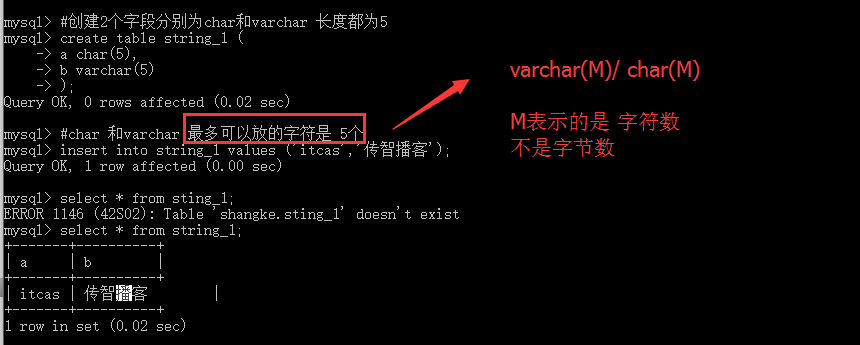
共同点: 在定义的时候都需要定义字符串的长度：char(M),varchar(M)

不同点:

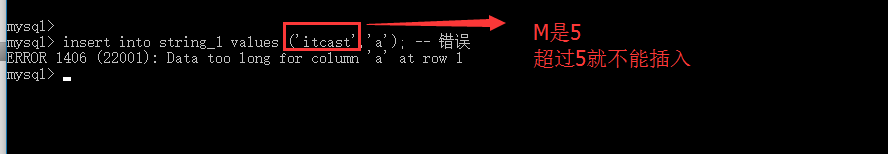
其中M表示允许的字符串最大长度

但是M在varchar中表示的是允许的最大范围，而varchar的实际长度是可变的，不超过M就行了！

M表示字符数



插入的字符数不能超过M



char和varchar的选择问题. 如果我们定义的一些字段是固定长度的我们就采用char.

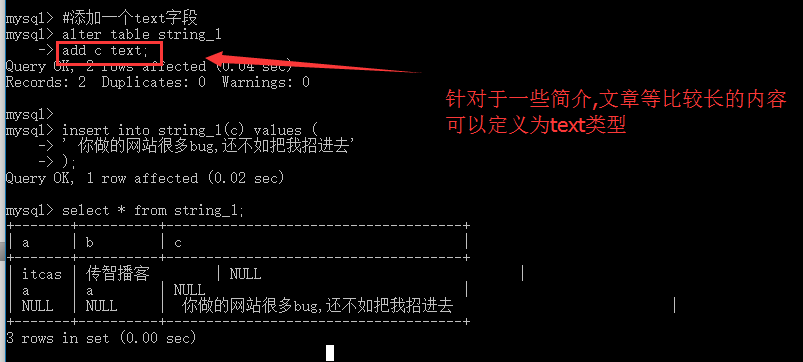
例如: 电话号码, 11位. 身份证号码. 密码通过md5之后的值是一个32个字符串

md5的算法是 任意长度的字符通过md5之后,都会变成一个32位的字符串.

md5不可以逆.

## text

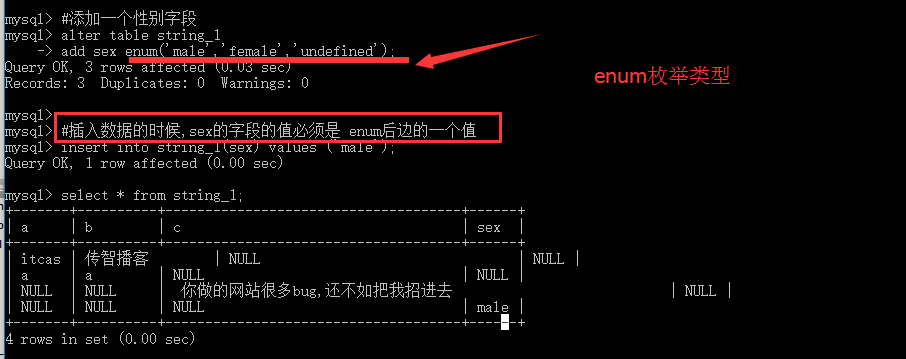
可以存储较长的字符串。不能设置默认值



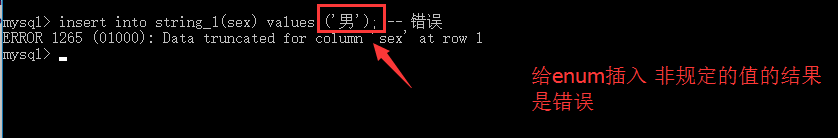
## enum枚举类型

枚举类型就是单选类型.

以 字段名 enum(值1,值2,值3,…) 的形式进行声明,在插入数据的时候,只能选择enum后边的一个值!!!!



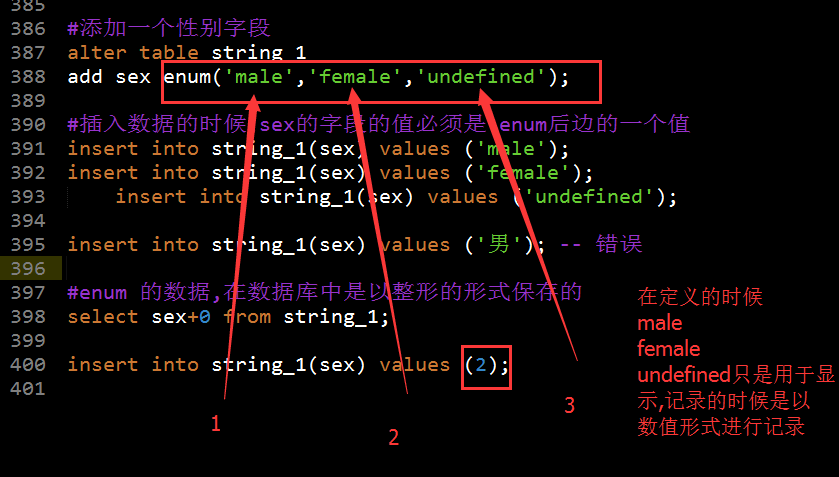
如果插入的数据不是enum后边的会报错



其实enum插入的或者保存的是 一个 数值.

enum在声明的时候,第一个元素的值对应的整形是 1

第二个是 2,第三个是3,顺次下去. 最大值是65535.



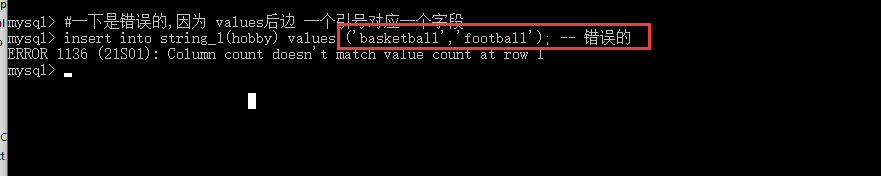
## set集合

类似于 多选.

有一些爱好,爱好不止一个,可以选择多个.

set集合类型以 字段名 set(值1,值2,值3,…) 的形式 声明,我们在选择字段值的时候,也必须选择 set后边的值.这个值可以是多个.

  
  
错误插入形式:



其实set集合的数据也是以整形的形式保存的,但是和enum有一些区别.

其中set的第一个值为 1, 0000 0001

set的第二个值为2 0000 0010

set的第三个值为4 0000 0100

set的第四个值为8 0000 1000

set的第五个值为16 0001 0000

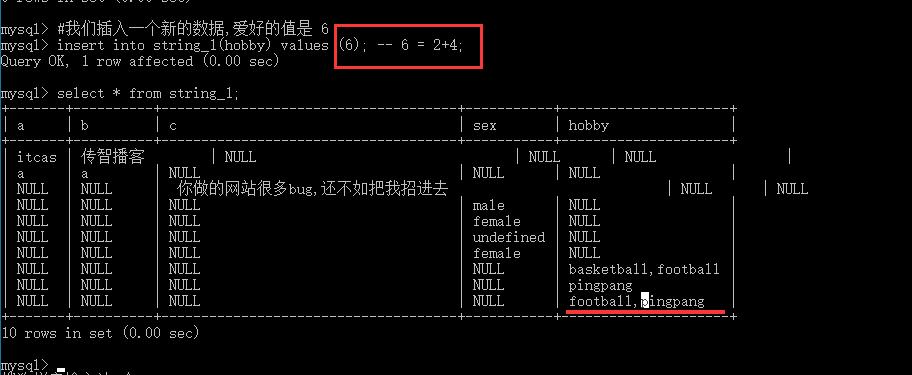
…



set最多 是64中情况

64/8 = 8 8个字节

11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111



创建一个学生表

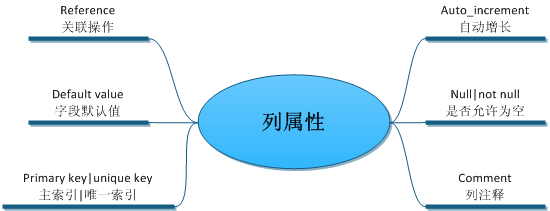
# 学生id,学生姓名,学生性别,学生的入学时间,学生的电话号码,

# 学生的爱好,学生的自我简介



# 第12讲 列属性(列约束)

列属性(列约束)就是为了更好的记录我们的数据

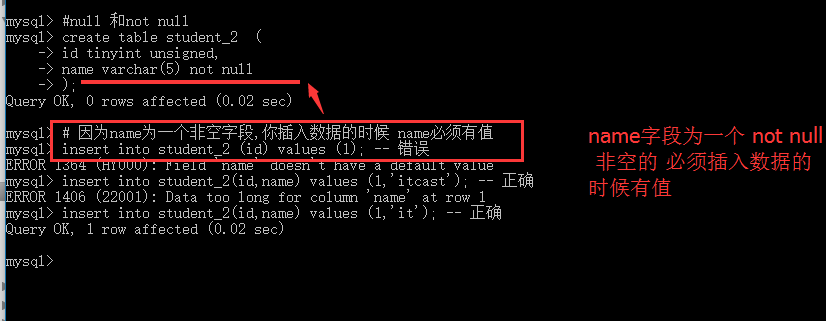


## null和not null

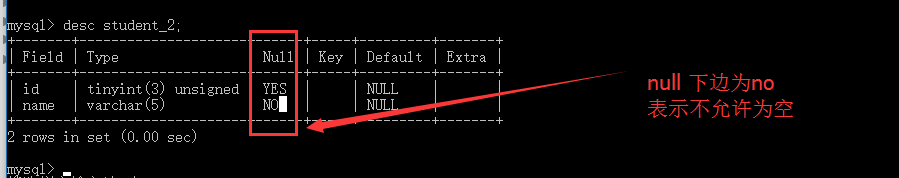
null 是默认的列约束,允许字段为空

not null 表示不允许字段为空

null 和 not null我们直接加入在 字段类型的后边就可以



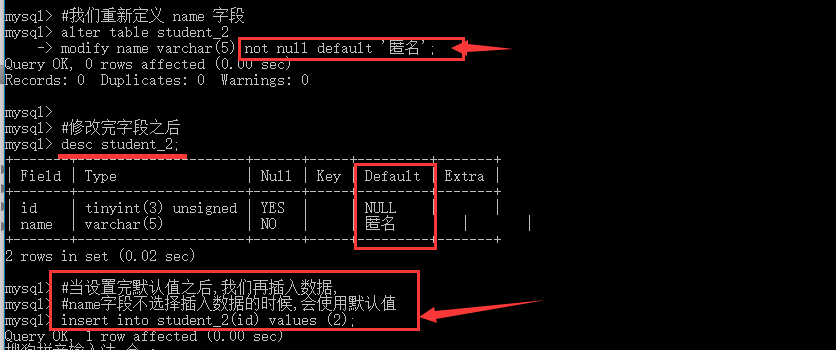
我们可以通过 desc 表名 来查看表的列约束



## default

default 中文含义是默认

一般和not null 配合使用 default 值; 给 not null 字段一个默认值

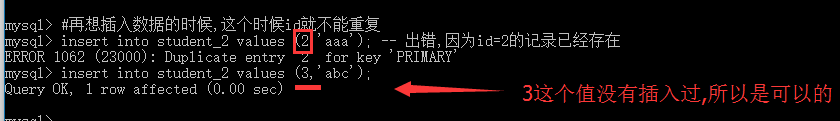


## primary key

### 概念

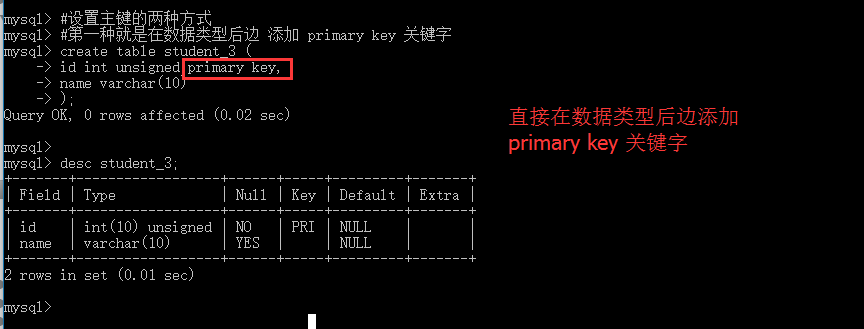
primary key 是 主键约束,简称 pk. 主键约束就是为了保证数据记录唯一.



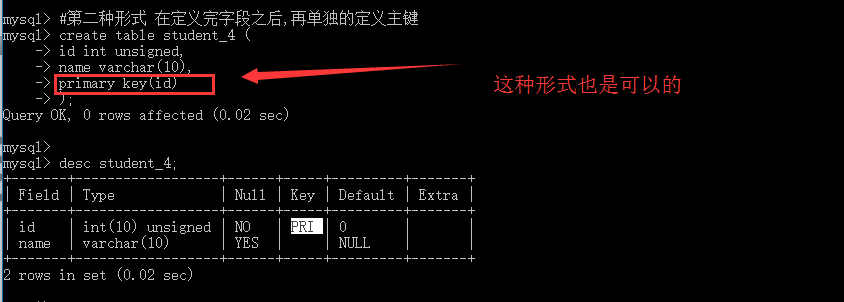


### 设置主键的两种方式

* 直接在定义字段的时候在后面进行设置



* 定义完字段后载定义主键. 定义主键的形式是 primary key (字段列表)



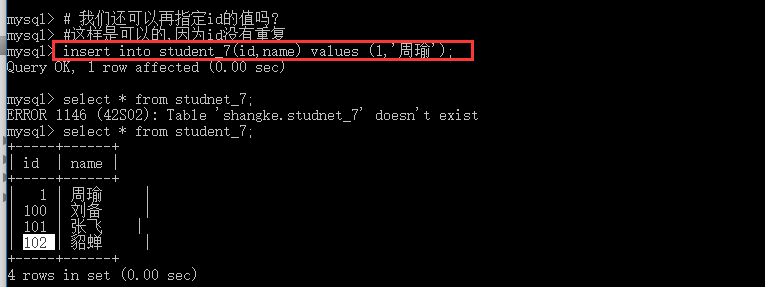
第一种形式一般是 主键就是一个单独的字段.

第二种形式 一般是 定义多个字段为主键



我们单独定义了一个学生id这个字段来确保数据的唯一性.

name + 身高 + 体重 + 发型



## unique key

### 概念

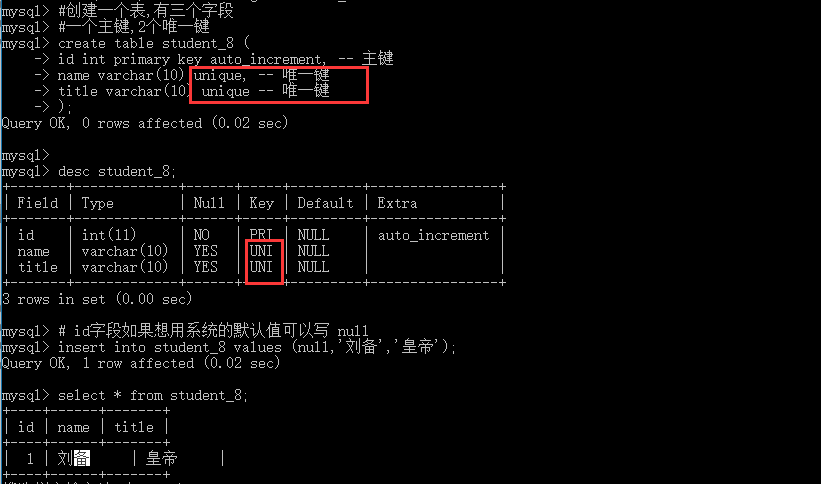
unique 是唯一性约束,就是保证字段里的值是一个唯一的.

字段有很多字段

学生表: 学号,姓名,性别,年龄,家庭住址,别名

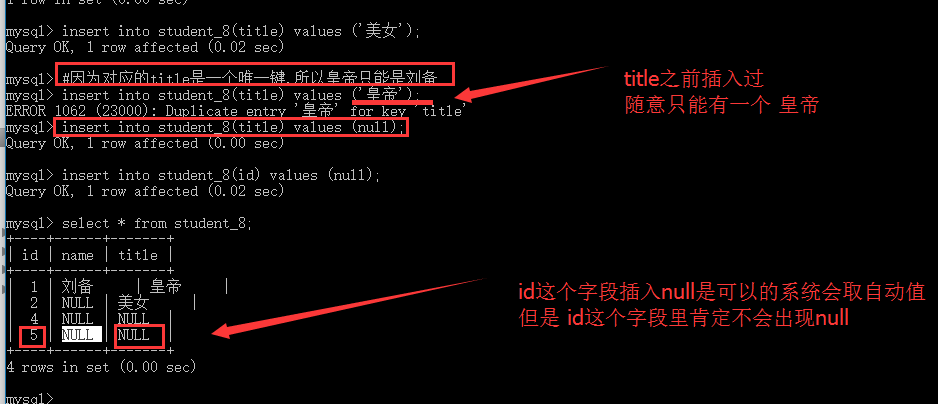
主键只能有一个, 学号

唯一键可以有多个, 姓名可以是唯一键,别名也可以是一个唯一键,



### 与主键的区别

* 主键只能有一个，但是唯一键可以有很多个
* 主键的值不能为null，但是唯一键可以为null

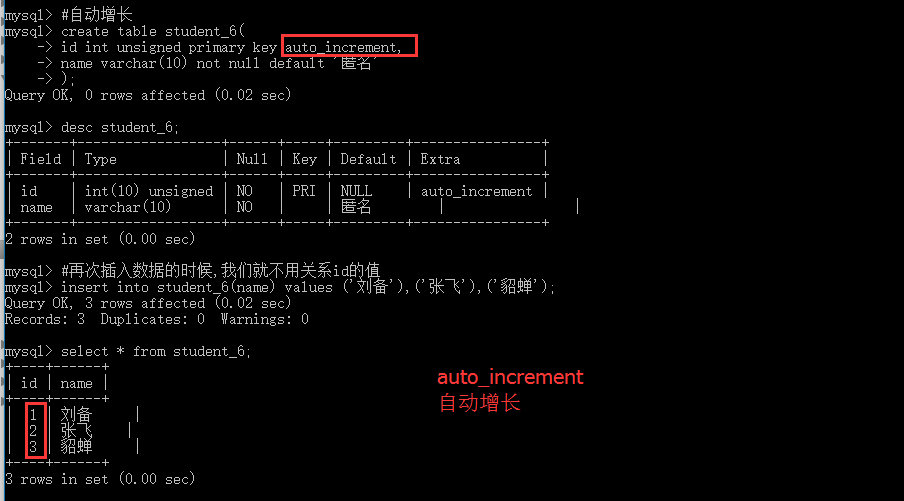


## auto\_increment

### 作用

就是在插入数据的时候,自动的增加1

auto\_increment一般和 主键配合使用



### 注意事项

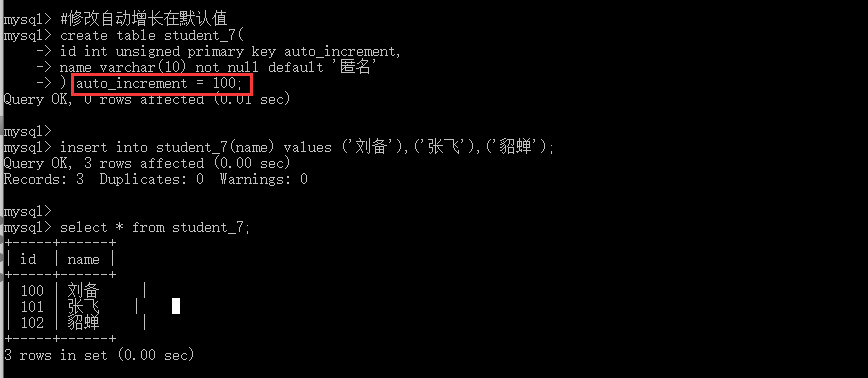
* 字段必须是整型，必须为索引（key）.
* 不指定字段值时，自动增长1.默认从1开始。

我们也可以通过 修改auto\_increment 来修改初始值

create table 表名 (

id int primary key auto\_increment

)auto\_increment=100;

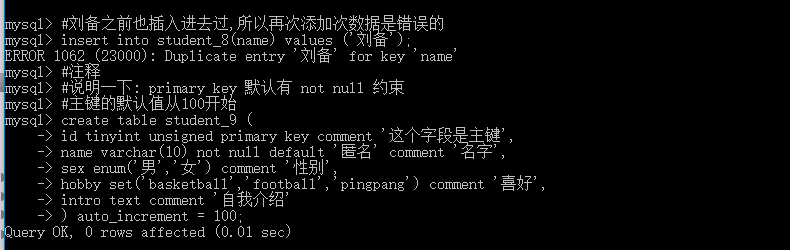


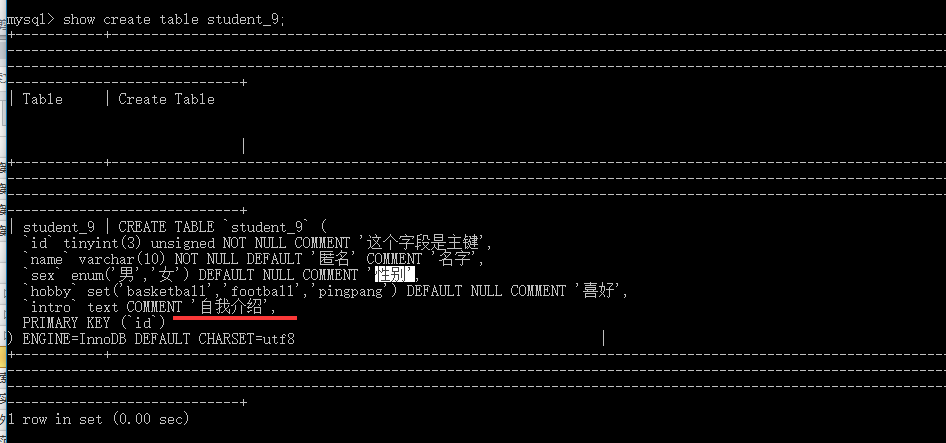
* 自增长一般与主键组合使用

## comment

注释,用于记录字段的说明,

comment 一般都是加在 字段类型的最后面.





# 第13讲 索引

## 概念

索引就是一些’内置表’,该表的数据是对某个真实表的某个或某些字段的数据进行了排序之后的存储形式

## 作用

极大的提高查询数据的速度

## 索引分类

### 普通索引

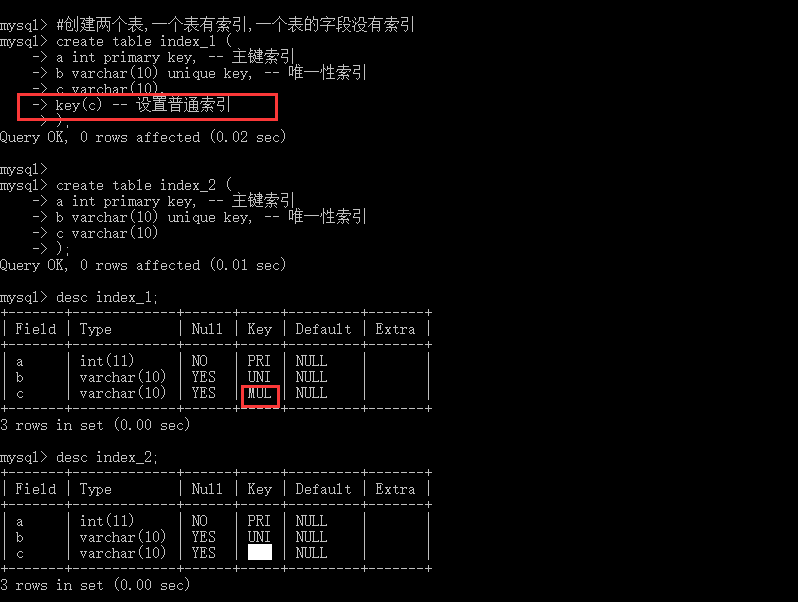
普通索引用 key(字段列表);

### 唯一索引

设置了唯一键,自动就有唯一索引

### 主键索引

设置了主键,自动就有主键索引



# 第14讲 实体与实体关系

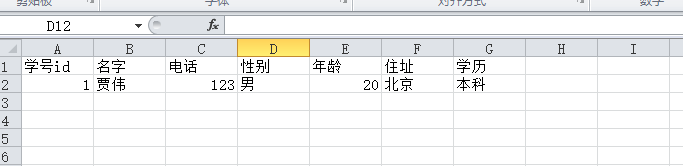
学生表: 学生id, 学生姓名,性别,爱好

老师表: 老师id,老师名字,授课班级

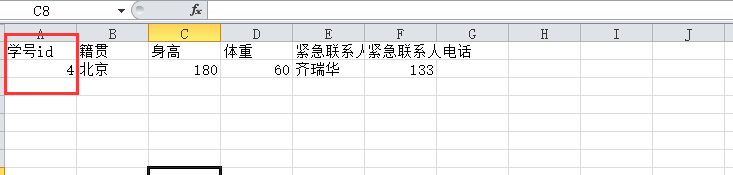
现实生活中，实体与实体之间肯定是有关系的，所以，我们在设计数据表的时候，就应该体现出表与表之间的关系，而我们把这种关系总结成三种: 一对一,一对多(多对一),多对多.

## 一对一

常用的信息:



不常用的信息:



常用信息的id和 不常用信息的id是 有一个对应关系: 这个对应关系是 1:1

1:1的关系一般都是 为了拆分表,让我们的表相对来说小(独立一点)

## 一对多(多对一)

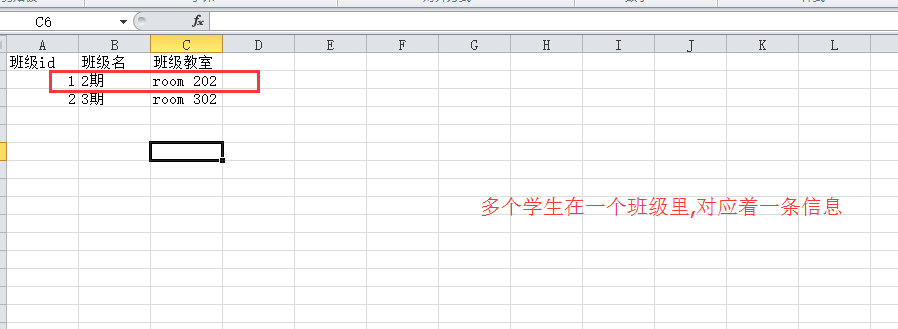
1:n (n:1)

多表示 多个学生

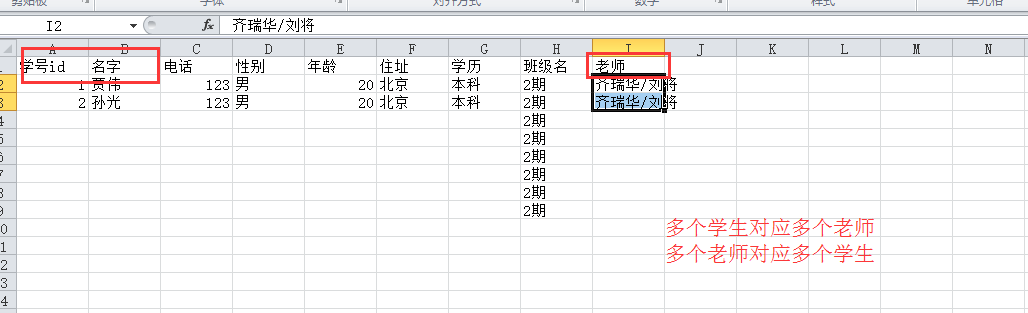


节省了2期每个都要写的,修改班级时更方便

1 表示 1个班级



## 多对多



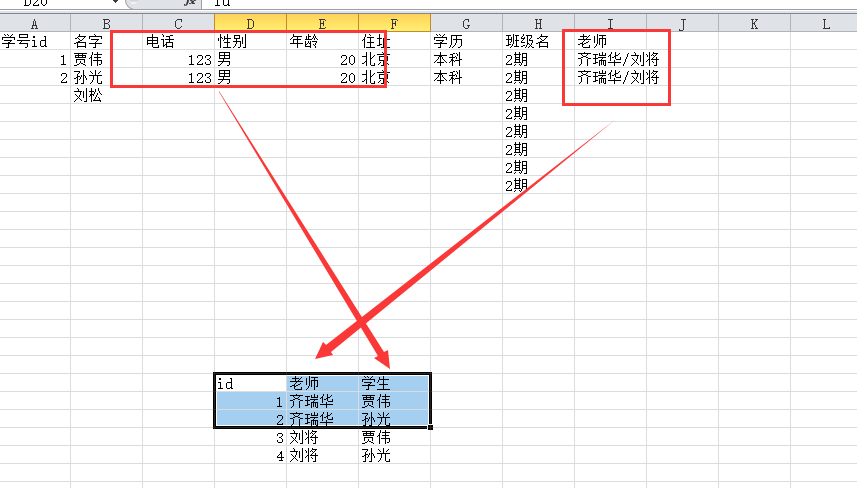
问题:

学生的老师字段,有多个数据.

我们需要对多对多的关系进行一个拆分!!!

把多对多的关系拆分成 多对一 或者说 一对多

我们需要再次建立一个表格,来记录所需要的关系.



# 外键

## 概念

如果一个实体A的某个字段，刚好指向或者是引用另一个实体B的主键，那么实体A的这个字段就叫作外键

A的主键是 a\_id

在a的表里,我需要 b的主键,这个时候, b的主键就不叫主键了,叫a的外键

B的主键是 b\_id

A理解为 学生表:

学生id, 学生姓名,性别,班级

B理解为 班级

班级名,班级教室,班级课桌数



## 语法

foreign key(本表字段) references外表名(外表字段) [选项];

## 删除外键和添加外键

### 删除外键

### 添加外键

## 级联操作

### 主表更新

### 主表删除

### 级联操作

级联操作一共有三种形式：

* **cascade：**同步操作，或者串联操作，也就是当父表的主键字段更新或者删除的时候，子表的外键字段也进行相应的更新或者删除！
* **set null：**设置为null，也就是当父表的主键字段更新或者删除的时候，子表的外键字段就设置为null，当然，前提是子表中的外键字段没有非空约束！
* **restrict：**就是拒绝主表更新或者删除！

## 注意事项

* 外键约束只有InnoDB存储引擎才支持

# 范式

## 概念

范式就是我们设计表的基本规范！Normal Format

### 作用

通过合理的数据存储，从而使得数据的冗余度最小化以及运行效率的最大化！

### 分层

根据不同的需求标准，一层一层的严格递进，一层比一层严格，理论上来说，范式一共有6层！

* 比如：第一范式，第二范式……
* 要想满足第二范式，首先要满足第一范式，依次类推！但是，后面的范式实在是太严格了，很难达到，所以在数据库中，只引入三层范式！
* 满足三层范式的数据库就是设计比较合理的数据库！

## 第一范式

### 概念

是最容易满足的，要求把各种数据设计成一个一个的单独的字段，不能再分割！

第一范式也叫满足“原子性”

例如: 性别和年龄设计在一个字段就不满足第一范式

## 第二范式

### 概念

就是在满足第一范式的基础上再满足以下的两个条件：

* 表中的每一行都具有唯一可区分的特性（就是不存在完全相同的记录）
* 不能有部分依赖

### 什么叫依赖？

如果确定表中的某个字段A，就一定能确定该表的另一个字段B，那么我们就说B依赖于A

例如: 所有的非主键字段都依赖于主键

### 什么叫做部分依赖？

假如一个表的组合主键是（A，B），其他字段都应该依赖于（A，B），但是如果此时有一个字段C，它只依赖于A，也就是说，只要A确定了，C也就确定了，这种情况就叫作部分依赖

例如: 想要知道学生的选修分数,必须先知道选修的课程. 选修的学分就部分依赖于课程和学生

## 第三范式

### 概念

在满足第二范式的基础之上消除传递依赖

### 什么是传递依赖？

如果一个表中有某个字段不直接依赖于主键，而是依赖于其他的字段，就叫作传递依赖！

* 假如主键是A，B依赖于A，而C又依赖于B！
* 注意：并不是说C字段不依赖于主键A，而是C只能等B确定后才依赖于A！

例如: 想要了解一个学生他的院系电话和院系地址,这个时候需要先确定院系