# 列类型（字段类型）

## 整数类型

### Tinyint

迷你整形，系统采用一个字节来保存的整形：一个字节 = 8位，最大能表示的数值是0-255

### Smallint

小整形，系统采用两个字节来保存的整形：能表示0-65535之间

### Mediumint

中整形，采用三个字节来保存数据。

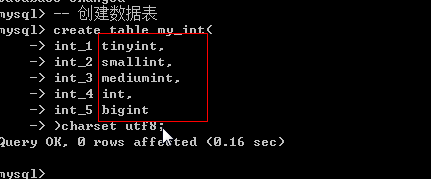
### Int

整形（标准整形），采用四个字节来保存数据。

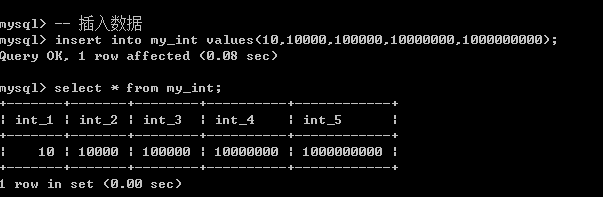
### Bigint

大整形，采用八个字节来保存数据。

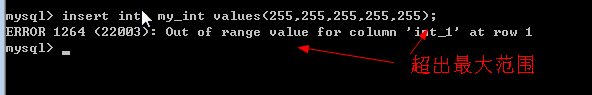
1、 创建数据表



2、 插入合理数据

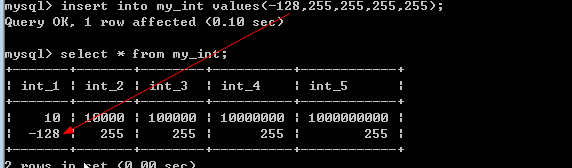


3、 插入错误数据（超出对应的数据范围）



4、 错误原因：并不是说tinyint没有这么大的空间，而是因为mysql默认的为整形增加负数。

实际表示的区间为-128,127

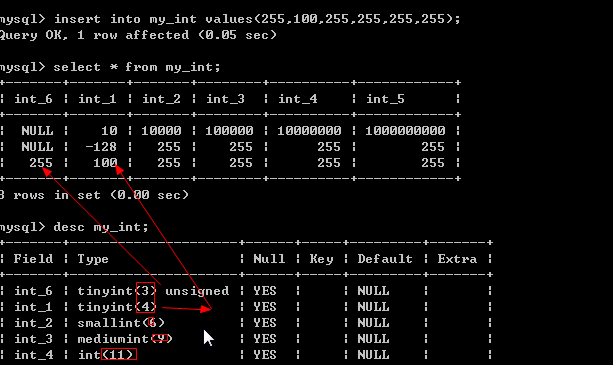


实际应用中，应该根据对应的数据的范围来选定对应的整形类型：通常使用的比较多的TINYINT和int。

### 无符号标识设定

无符号：表示存储的数据在当前字段中，没有负数（只有正数，区间为0-255）

基本语法：在类型之后加上一个 unsigned



## 显示长度

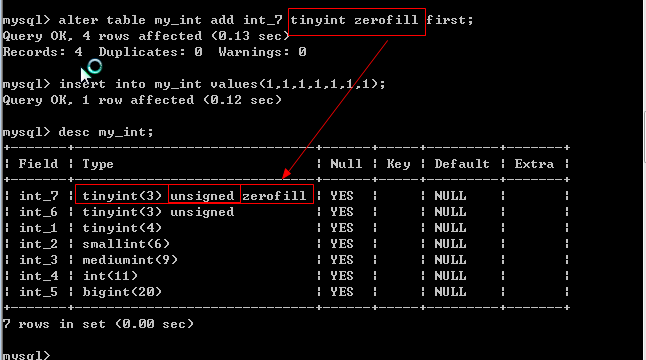
显示长度：指数据（整型）在数据显示的时候，到底可以显示多长位。

Tinyint(3)： 表示最长可以显示3位，unsigned说明只能是正数，0-255永远不会超过三个长度

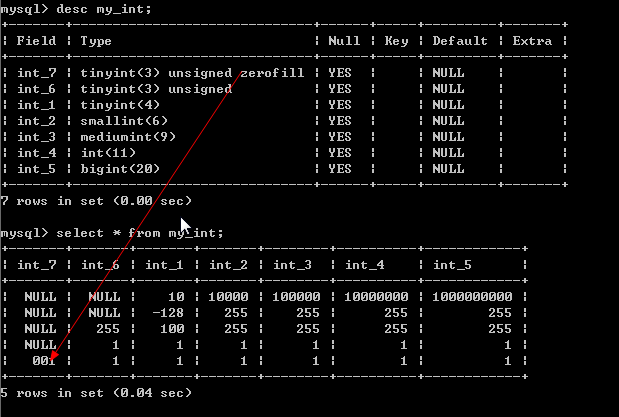
Tinyint(4)：表示最长可以显示4位，-128~127

显示长度只是代表了数据是否可以达到指定的长度，但是不会自动满足到指定长度：如果想要数据显示的时候，保持最高位（显示长度），那么还需要给字段增加一个zerofill属性才可以。

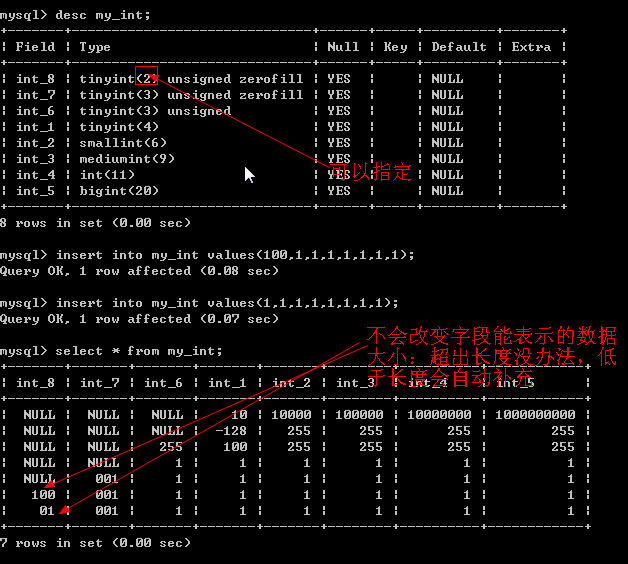
Zerofill：从左侧开始填充0（左侧不会改变数值大小），所以负数的时候就不能使用zerofill，一旦使用zerofill就相当于确定该字段为unsigned



数据显示的时候，zerofill会在左侧填充0到指定位：如果不足3位，那么填充到3位，如果本身已经够了或者超出，那么就不在填充。



显示长度可以自己设定：超出长度（但是不超出范围）不会影响，只会对不够长度的进行补充（显示长度）



# 小数类型

专门用来存储小数的

在Mysql中将小数类型又分为两类：浮点型和定点型

## 浮点型

浮点型又称之为精度类型：是一种有可能丢失精度的数据类型，数据有可能不那么准确（由其是在超出范围的时候）

浮点型之所以能够存储较大的数值（不精确），原因就是利用存储数据的位来存储指数

整型：所有位都为1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

计算结果：

浮点型：有部分用于存储数据，有部分用于存指数

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

前三位转换成十进制之后用作10的指数： 10^7 \* 数据值

### Float

Float又称之为单精度类型：系统提供4个字节用来存储数据，但是能表示的数据范围比整型大的多，大概是10^38；只能保证大概7个左右的精度（如果数据在7位数以内，那么基本是准确的，但是如果超过7位数，那么就是不准确的）

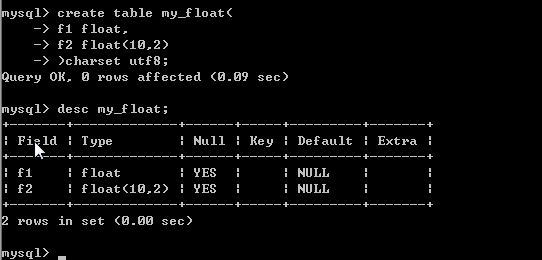
基本语法

Float：表示不指定小数位的浮点数

Float(M,D)：表示一共存储M个有效数字，其中小数部分占D位

Float(10,2)：整数部分为8位，小数部分为2位

1、 创建一个数据表保存浮点数据

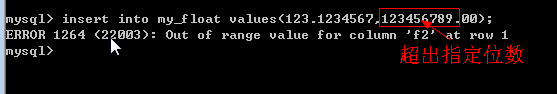


2、 存入数据：合法数据

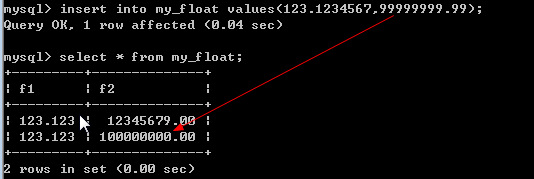


注意：如果数据精度丢失，那么浮点型是按照四舍五入的方式进行计算

3、 插入数据，超出大小

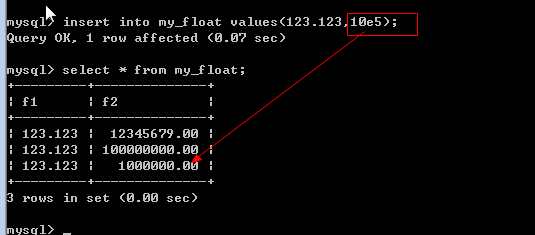


4、 数据长度刚好满足条件，但是会超出精度



说明：用户不能插入数据直接超过指定的整数部分长度，但是如果是系统自动进位导致，系统可以承担。

5、浮点数可以采用科学计数法来存储数据



浮点数的应用：通常是用来保存一些数量特别大，大到可以不用那么精确的数据。

### Double

Double又称之为双精度：系统用8个字节来存储数据，表示的范围更大，10^308次方，但是精度也只有15位左右。

## 定点数

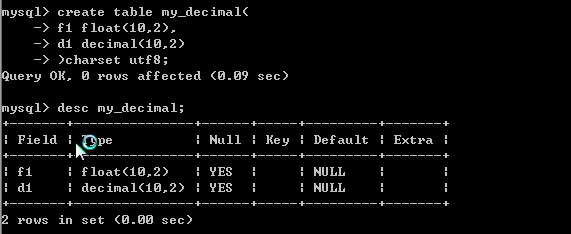
定点数：能够保证数据精确的小数（小数部分可能不精确，超出长度会四舍五入），整数部分一定精确

### Decimal

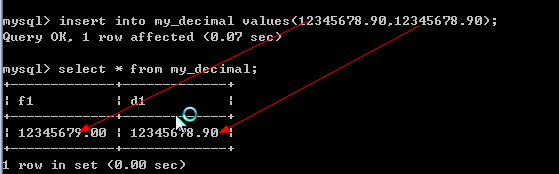
Decimal定点数：系统自动根据存储的数据来分配存储空间，每大概9个数就会分配四个字节来进行存储，同时小数和整数部分是分开的。

Decimal(M,D)：M表示总长度，最大值不能超过65，D代表小数部分长度，最长不能超过30。

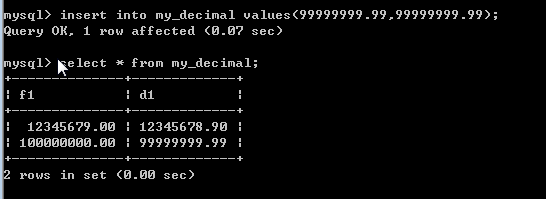
1、 创建表：与浮点数对比



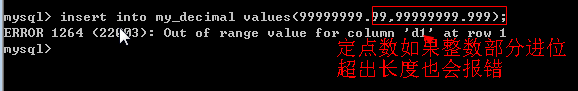
2、 插入正常数据



3、 插入最大数据



4、 尝试定点数进行四舍五入



定点数的应用：如果涉及到钱的时候有可能使用定点数

# 时间日期类型

## Date

日期类型：系统使用三个字节来存储数据，对应的格式为：YYYY-mm-dd，能表示的范围是从1000-01-01 到9999-12-12，初始值为0000-00-00

## Time

时间类型：能够表示某个指定的时间，但是系统同样是提供3个字节来存储，对应的格式为：HH:ii:ss，但是mysql中的time类型能够表示时间范围要大的多，能表示从-838:59:59~838:59:59，在mysql中具体的用处是用来描述时间段。

## Datetime

日期时间类型：就是将前面的date和time合并起来，表示的时间，使用8个字节存储数据，格式为YYYY-mm-dd HH:ii:ss，能表示的区间1000-01-01 00:00:00 到9999-12-12 23:59:59，其可以为0值：0000-00-00 00:00:00

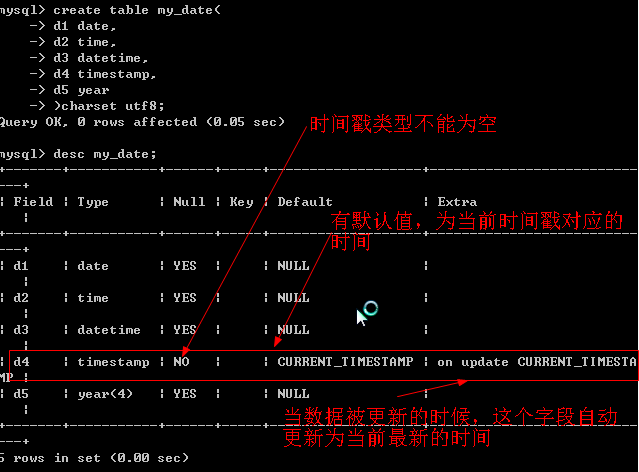
## Timestamp

时间戳类型：mysql中的时间戳只是表示从格林威治时间开始，但是其格式依然是：YYYY-mm-dd HH:ii:ss

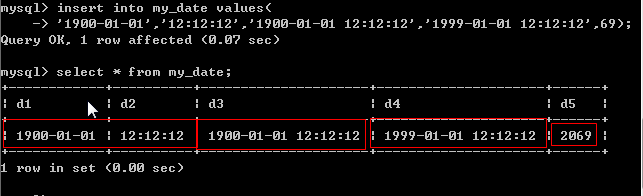
## Year

年类型：占用一个字节来保存，能表示1900~2155年，但是year有两种数据插入方式：0~99和四位数的具体年

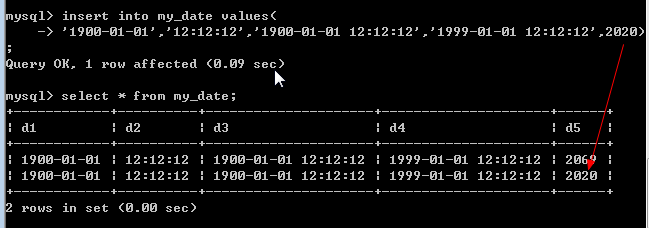
1、 创建对应的时间日期类型的数据表



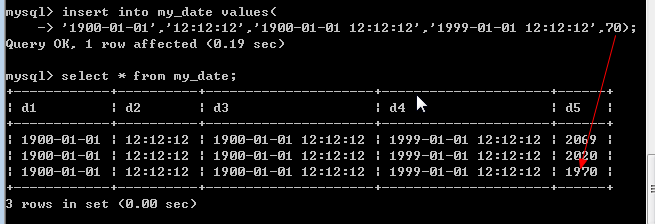
2、 插入数据：正常数据



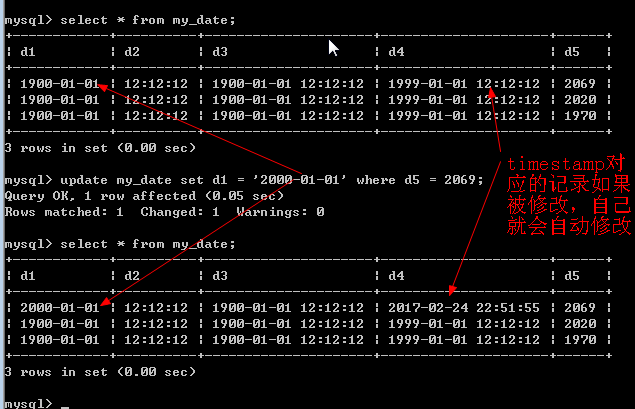
3、 year的特殊性：可以采用两位数的数据插入，也可以采用四位数的年份插入



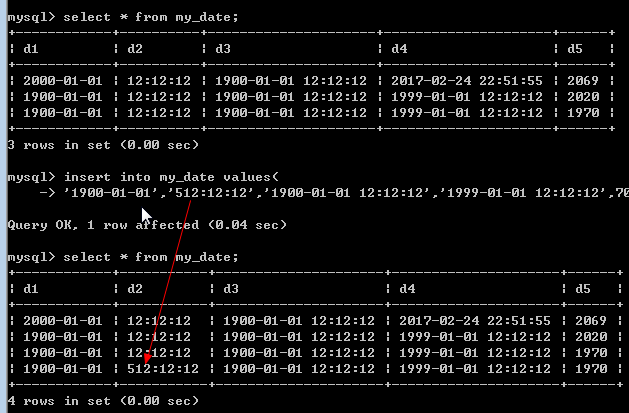
4、 year进行两位数插入的时候，有一个区间划分，零界点为69和70：当输入69以下，那么系统时间为20+数字，如果是70以上，那配系统时间为19+数字



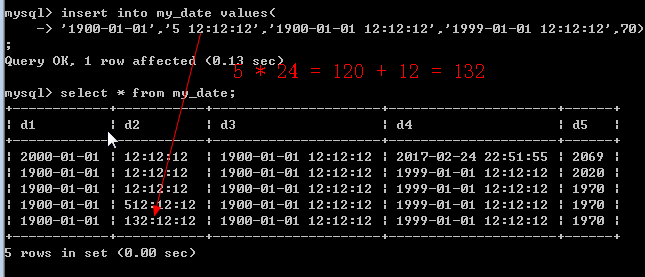
5、 timestamp当对应的数据被修改的时候，会自动更新（这个被修改的数据不是自己）



6、 time类型特殊性：本质是用来表示时间区间（当前时间之后的多少个小时），能表示的范围比较大



7、 在进行时间类型录入的时候（time）还可以使用一个简单的日期代替时间，在时间格式之前加一个空格，然后指定一个数字（可以是负数）：系统会自动将该数字转换成天数 \* 24小时，再加上后面的时间。



PHP中有着非常强大的时间日期转换函数：date将时间戳转换成想要的格式，strtotime又可以将很多格式转换成对应的时间戳。PHP通常不需要数据库来帮助处理这么复杂的时间日期，所以通常配合PHP的时候，时间的保存通常使用时间戳（真正），从而用整型来保存。字符串型

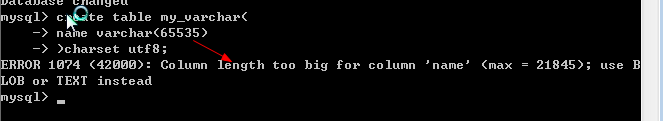
# Mysql记录长度

在mysql中，有一项规定：mysql的记录长度（record == 行row）总长度不能超过65535个字节。

Varchar能够存储的理论值为65535个字符：字符在不同的字符集下可能占用多个字节。

1、 创建表：证明varchar在mysql中能够达到的理论值（utf8和GBK）

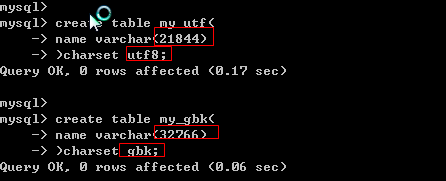
Varchar除了存储的数据本身要占用空间：还需要额外的空间来保存记录长度



2、 计算在utf8和GBK下对应的varchar能够存储的长度

Utf8 最多只能存储21844个字符

GBK最多只能存储32766个字符



# 字符串型

## Char

定长字符：指定长度之后，系统一定会分配指定的空间用于存储数据

基本语法：char(L)，L代表字符数（中文与英文字母一样），L长度为0到255

## Varchar

变长字符：指定长度之后，系统会根据实际存储的数据来计算长度，分配合适的长度（数据没有超出长度）

基本语法：Varchar(L)，L代表字符数，L的长度理论值位0到65535

因为varchar要记录数据长度（系统根据数据长度自动分配空间），所以每个varchar数据产生后，系统都会在数据后面增加1-2个字节的额外开销：是用来保存数据所占用的空间长度

如果数据本身小于127个字符：额外开销一个字节；如果大于127个，就开销两个字节

Char和varchar数据存储对比（utf8，一个字符都会占用3个字节）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 存储数据 | Char(2) | Varchar(2) | Char所占字节 | Varchar所占字节 |
| A | A | A | 2 \* 3 = 6 | 1 \* 3 + 1 = 4 |
| AB | AB | AB | 2 \* 3 = 6 | 2 \* 3 + 1 = 7 |

Char和varchar的区别

1、 char一定会使用指定的空间，varchar是根据数据来定空间

2、 char的数据查询效率比varchar高：varchar是需要通过后面的记录数来计算

如果确定数据一定是占指定长度，那么使用char类型；

如果不确定数据到底有多少，那么使用varchar类型；

如果数据长度超过255个字符，不论是否固定长度，都会使用text，不再使用char和varchar

## Text

文本类型：本质上mysql提供了两种文本类型

Text：存储普通的字符文本

Blob：存储二进制文本（图片，文件），一般都不会使用blob来存储文件本身，通常是使用一个链接来指向对应的文件本身。

Text：系统中提供的四种text

Tinytext：系统使用一个字节来保存，实际能够存储的数据为：2 ^ 8 + 1

Text：使用两个字节保存，实际存储为：2 ^ 16 + 2

Mediumtext：使用三个字节保存，实际存储为：2 ^ 24 + 3

Longtext：使用四个字节保存，实际存储为：2 ^ 32 + 4

注意：

1、 在选择对应的存储文本的时候，不用刻意去选择text类型，系统会自动根据存储的数据长度来选择合适的文本类型。

2、 在选择字符存储的时候，如果数据超过255个字符，那么一定选择text存储

## Enum

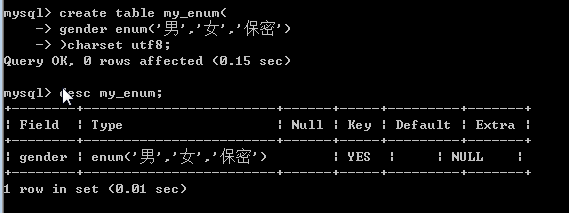
枚举类型：在数据插入之前，先设定几个项，这几个项就是可能最终出现的数据结果。

如果确定某个字段的数据只有那么几个值：如性别，男、女、保密，系统就可以在设定字段的时候规定当前字段只能存放固定的几个值：使用枚举

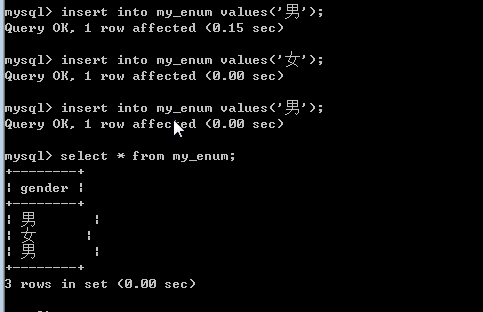
基本语法：enum(数据值1,数据值2…)

系统提供了1到2个字节来存储枚举数据：通过计算enum列举的具体值来选择实际的存储空间：如果数据值列表在255个以内，那么一个字节就够，如果超过255但是小于65535，那么系统采用两个字节保存。

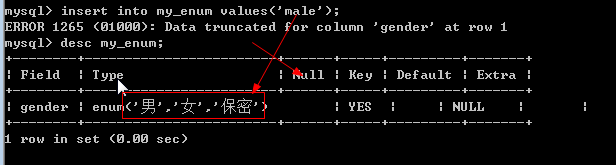
1、 创建表



2、 插入数据：合法数据，字段对应的值必须是设定表的时候所确定的值



3、 错误数据：enum有规范数据的功能，能够保证插入的数据必须是设定的范围，其他类型都不可以

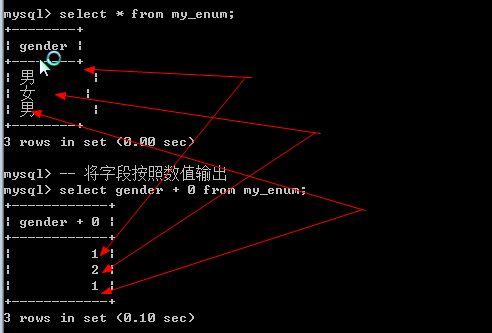


4、 枚举enum的存储原理：实际上字段上所存储的值并不是真正的字符串，而是字符串对应的下标：当系统设定枚举类型的时候，会给枚举中每个元素定义一个下标，这个下标规则从1开始

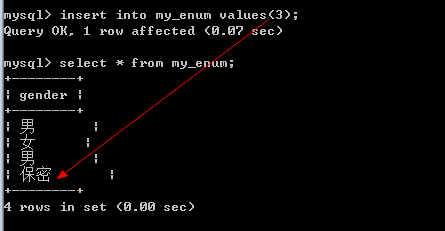
Enum(1=>‘男’,2=>’女’,3=>’保密’)

特性：在mysql中系统是自动进行类型转换的：如果数据碰到“+、-、\*、/”系统就会自动将数据转换成数值：而普通字符串转换成数值为0

Select 字段名 + 0 from 表名;



5、 既然实际enum字段存储的结果是数值：那么在进行数据插入的时候，就可以使用对应的数值来进行。



枚举的意义：

1、 规范数据本身，限定只能插入规定的数据项

2、 节省存储空间

## Set

集合：是一种将多个数据选项可以同时保存的数据类型，本质是将指定的项按照对应的二进制位来进行控制：1表示该选项被选中，0表示该选项没有被选中。

基本语法：set(‘值1’,’值2’,’值3’…)

系统为set提供了多个字节进行保存，但是系统会自动计算来选择具体的存储单元

1个字节：set只能有8个选项

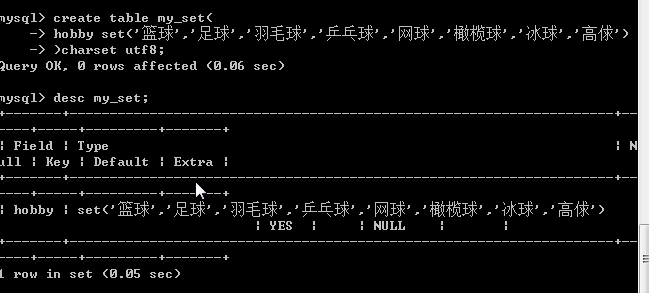
2个字节：set只能有16个选项

3个字节：set只能表示24个选项

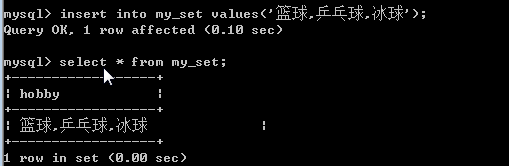
8个字节：set可以表示64个选项

Set和enum一样，最终存储到数据字段中的依然是数字而不是真实的字符串

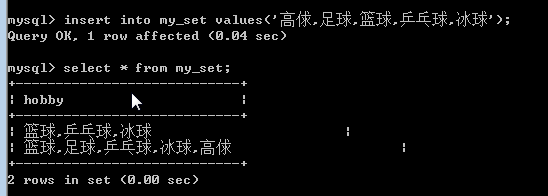
1、 创建表



2、 插入数据：可以插入多个数据，就是在数据插入的字符串中，使用对应的逗号“,”将选项进行隔开

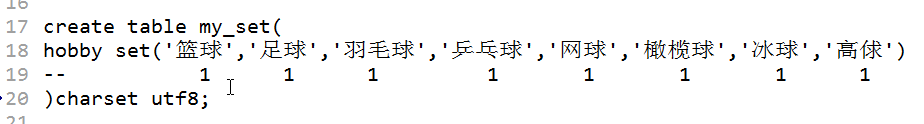


3、 数据选项所在的数据与数据插入的顺序无关：最终都会变成选项对应的顺序

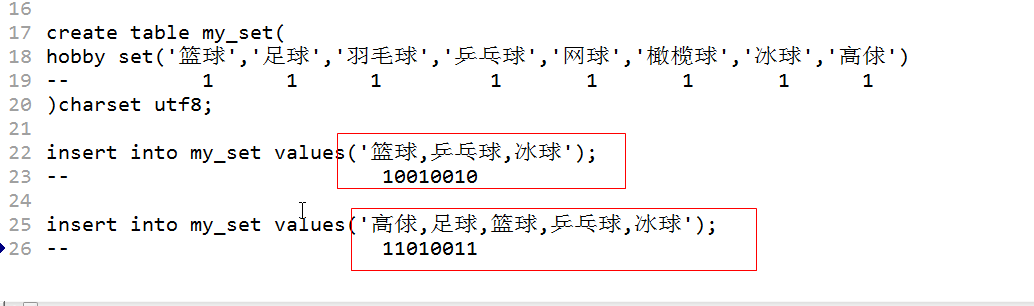


4、 分析数据存储的方式

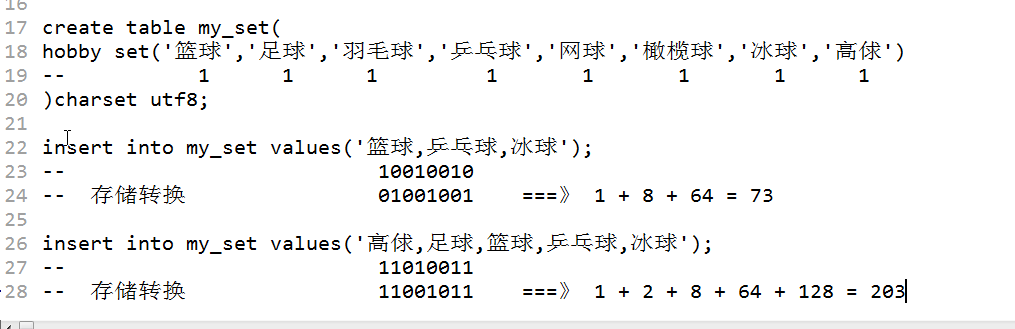
4.1 系统将对应的数据选项（设计）按照顺序进行编排：从第一个开始进行占位，每一个都对应一个二进制位。



4.2 数据在存储的时候，如果被选中，那么对应的位的值就为1，否则为0



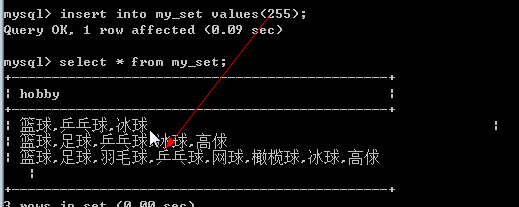
4.3 系统在进行存储的时候会自动将得到的最终的二进制颠倒过来，然后再进行转换成十进制存储



5、 查看数据：按照自动转换成数值来查看



6、 既然是数值，那么就可以插入数值来代替实际插入数据



注意：数字插入的前提是对应的二进制位上都有对应的数据项

Set集合的意义：

1、 规范数据

2、 节省存储空间

Enum：单选框

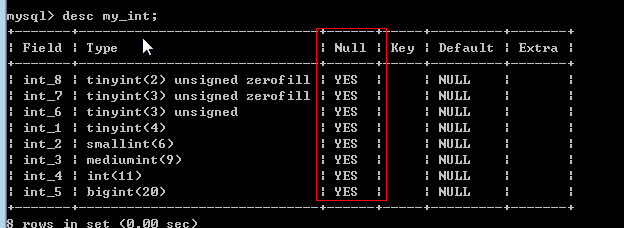
Set：复选框

# 列属性

列属性又称之为字段属性，在mysql中一共有6个属性：null，默认值，列描述，主键，唯一键和自动增长

## Null属性

NULL属性：代表字段为空



如果对应的值为YES表示该字段可以为NULL

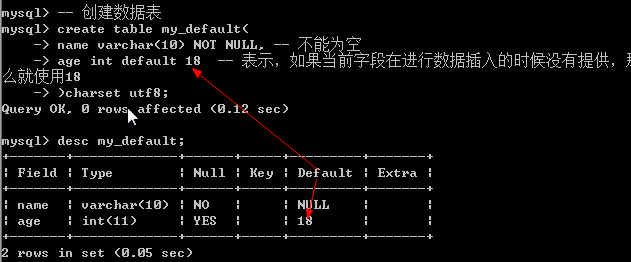
注意：

1、 在设计表的时候，尽量不要让数据为空

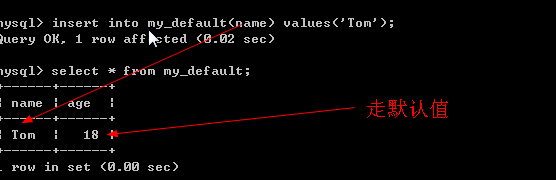
2、 Mysql的记录长度为65535个字节，如果一个表中有字段允许为NULL，那么系统就会设计保留一个字节来存储NULL，最终有效存储长度为65534个字节。

## 默认值

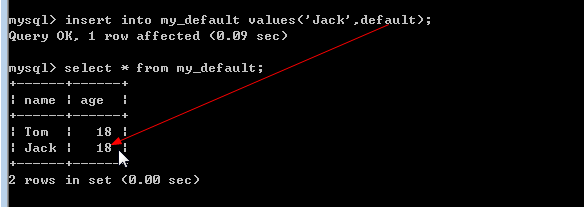
Default：默认值，当字段被设计的时候，如果允许默认条件下，用户不进行数据的插入，那么就可以使用事先准备好的数据来填充：通常填充的是NULL



测试：不给当前字段提供数据值



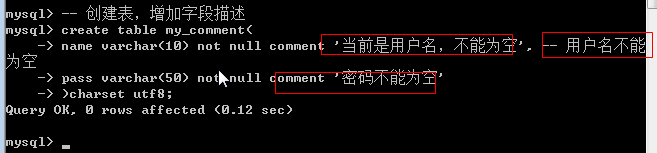
Default关键字的另外一层使用：显示的告知字段使用默认值：在进行数据插入的时候，对字段值直接使用default



## 列描述

列描述：comment，是专门用于给开发人员进行维护的一个注释说明

基本语法：comment ‘字段描述’;



查看Comment：必须通过查看表创建语句



# 主键

顾名思义：主要的键，primary key，在一张表中，有且只有一个字段，里面的值具有唯一性

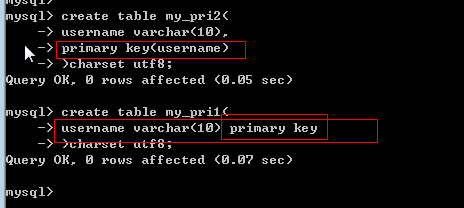
## 创建主键

### 随表创建

系统提供了两种增加主键的方式

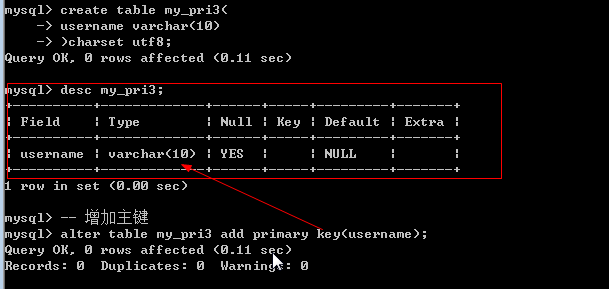
1、 方案1：直接在需要当做主键的字段之后，增加primary key属性来确定主键

2、 方案2：在所有字段之后增加primary key选项：primary key(字段信息)



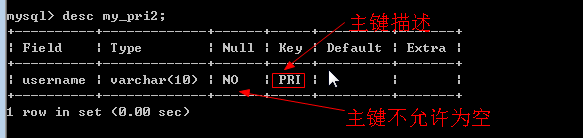
### 表后增加

基本语法：alter table 表名 add primary key(字段);

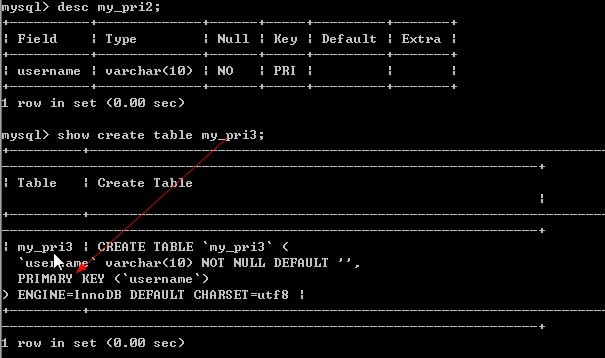


## 查看主键

方案1：查看表结构

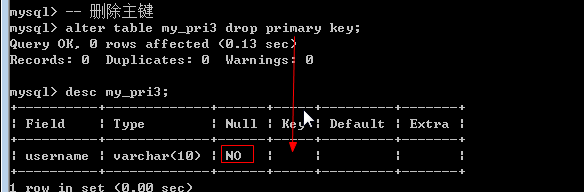


方案2：查看表的创建语句



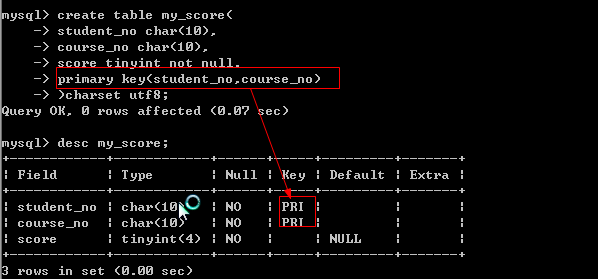
## 删除主键

基本语法：alter table 表名 drop primary key;



## 复合主键

案例：有一张学生选修课表：一个学生可以选修多个选修课，一个选修课也可以由多个学生来选：但是一个学生在一个选修课中只有一个成绩。

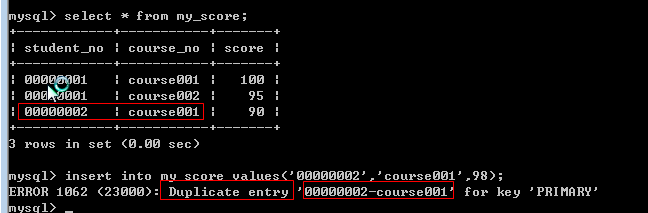


## 主键约束

主键一旦增加，那么对对应的字段有数据要求

1、 当前字段对应的数据不能为空；

2、 当前字段对应的数据不能有任何重复



## 主键分类

主键分类采用的是主键所对应的字段的业务意义分类

业务主键：主键所在的字段，具有业务意义（学生ID，课程ID）

逻辑主键：自然增长的整型（应用广泛）

# 自动增长

自动增长：auto\_increment，当给定某个字段该属性之后，该列的数据在没有提供确定数据的时候，系统会根据之前已经存在的数据进行自动增加后，填充数据。

通常自动增长用于逻辑主键。

## 原理

自动增长的原理：

1、 在系统中有维护一组数据，用来保存当前使用了自动增长属性的字段，记住当前对应的数据值，再给定一个指定的步长

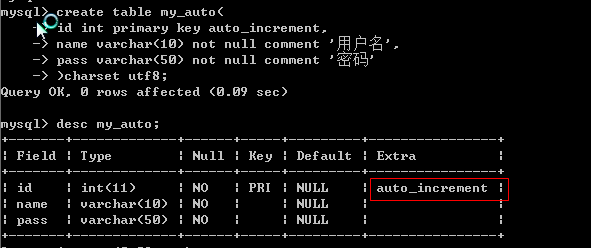
2、 当用户进行数据插入的时候，如果没有给定值，系统在原始值上再加上步长变成新的数据

3、 自动增长的触发：给定属性的字段没有提供值

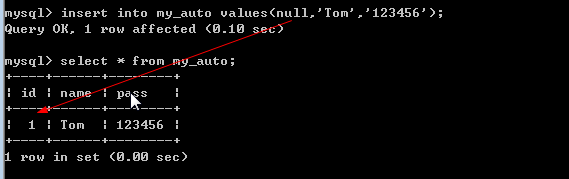
4、 自动增长只适用于数值

## 使用自动增长

基本语法：在字段之后增加一个属性auto\_increment

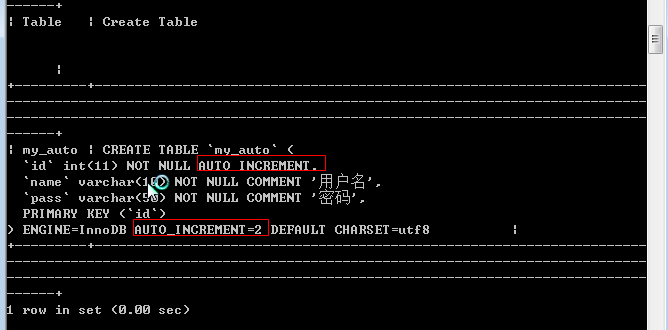


插入数据：触发自动增长，不能给定具体值



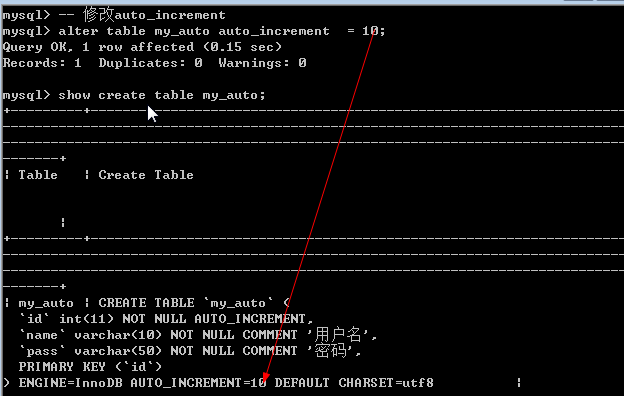
## 修改自动增长

1、 查看自增长：自增长一旦触发使用之后，会自动的在表选项中增加一个选项（一张表最多只能拥有一个自增长）



2、 表选项可以通过修改表结构来实现

Alter table 表名 auto\_increment = 值;



## 删除自动增长

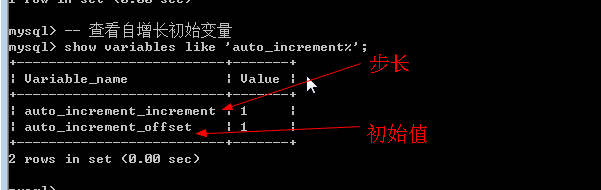
删除自增长：就是在字段属性之后不再保留auto\_increment，当用户修改自增长所在字段时，如果没有看到auto\_increment属性，系统会自动清除该自增长



## 初始设置

在系统中，有一组变量用来维护自增长的初始值和步长

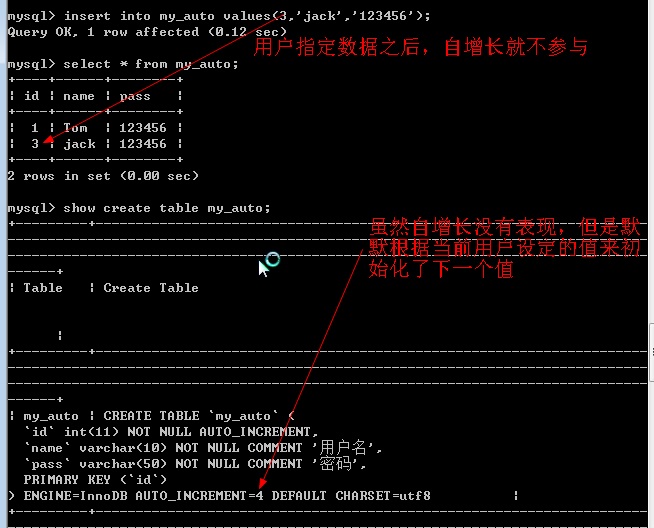
Show variables like ‘auto\_increment%’;



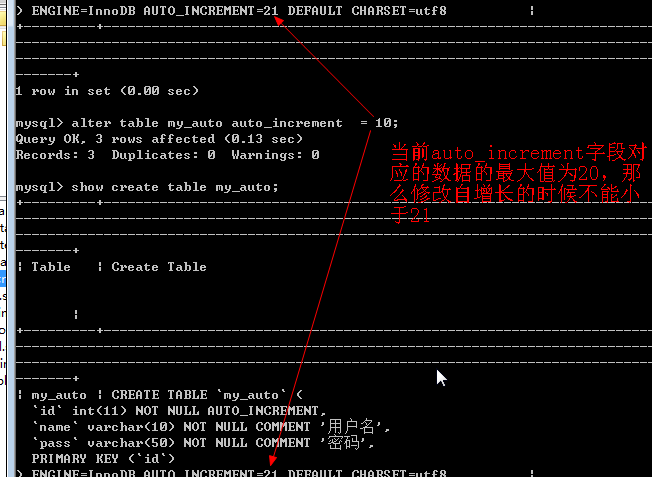
## 细节问题

1、 一张表只有一个自增长：自增长会上升到表选项中

2、 如果数据插入中没有触发自增长（给定了数据），那么自增长不会表现



3、 自增长修改的时候，值可以较大，但是不能比当前已有的自增长字段的值小



# 唯一键

唯一键：unique key，用来保证对应的字段中的数据唯一的。

主键也可以用来保证字段数据唯一性，但是一张表只有一个主键。

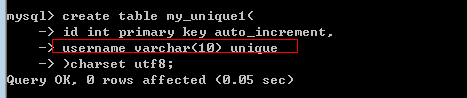
1、 唯一键在一张表中可以有多个。

2、 唯一键允许字段数据为NULL，NULL可以有多个（NULL不参与比较）

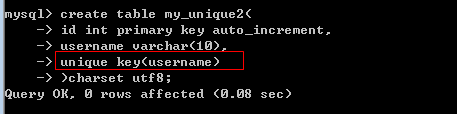
## 创建唯一键

创建唯一键与创建主键非常类似

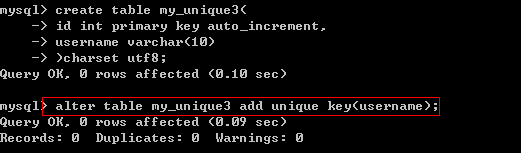
1、 直接在表字段之后增加唯一键标识符：unique[ key]



2、 在所有的字段之后使用unique key(字段列表);

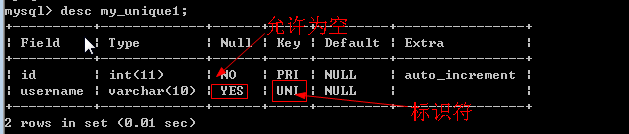


3、 在创建完表之后也可以增加唯一键  
alter table 表名 add unique key(字段列表);

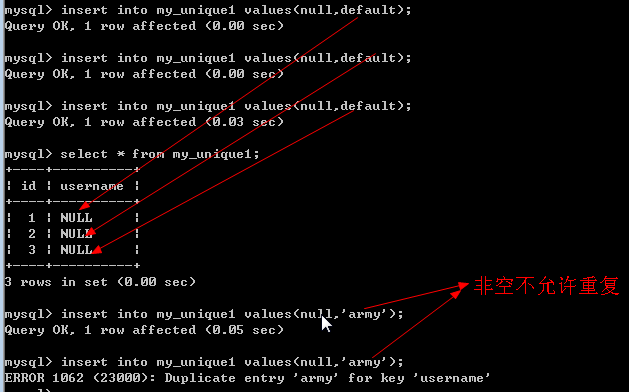


## 查看唯一键

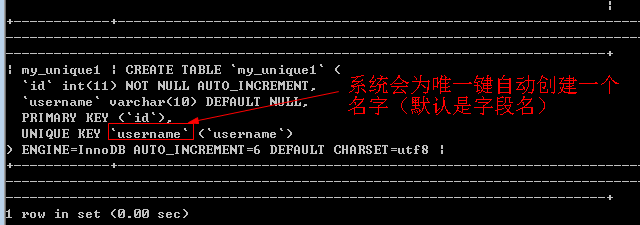
唯一键是属性，可以通过查看表结构来实现



唯一键效果：在不为空的情况下，不允许重复



在查看表创建语句的时候，会看到与主键不同的一点：多出一个“名字”

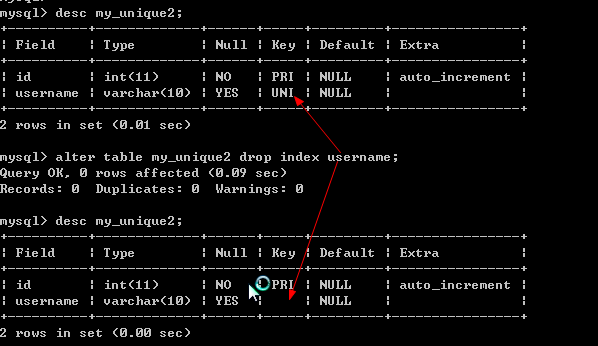


## 删除唯一键

一个表中允许存在多个唯一键：假设命令为主键一样：alter table 表名 drop unique key;//错误的

Index关键字：索引，唯一键是索引一种（提升查询效率）

删除的基本语法：alter table 表名 drop index 唯一键名字;



修改唯一键：先删除后增加

## 复合唯一键

唯一键与主键一样可以使用多个字段来共同保证唯一性；

一般主键都是单一字段（逻辑主键），而其他需要唯一性的内容都是由唯一键来处理。

# 表关系

表关系：表与表之间（实体）有什么样的关系，每种关系应该如何设计表结构。

## 一对一

一对一：一张表中的一条记录与另外一张表中最多有一条明确的关系：通常，此设计方案保证两张表中使用同样的主键即可

学生表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学生ID（PRI） | 姓名 | 年龄 | 性别 | 籍贯 | 婚否 | 住址 |
|  |  |  |  |  |  |  |

表的使用过程中：常用的信息会经常去查询，而不常用的信息会偶尔才会用到。

解决方案：将两张表拆分，常见的放一张表，不常见的放一张表

常用表

|  |  |  |  |
| --- | --- | --- | --- |
| 学生ID（PRI） | 姓名 | 年龄 | 性别 |
|  |  |  |  |

不常用表

|  |  |  |  |
| --- | --- | --- | --- |
| 学生ID（PRI） | 籍贯 | 婚否 | 住址 |
|  |  |  |  |

## 一对多

一对多，通常也叫作多对一的关系。通常一对多的关系设计的方案，在“多”关系的表中去维护一个字段，这个字段是“一”关系的主键。

母亲表

|  |  |  |  |
| --- | --- | --- | --- |
| 母亲ID | 姓名 | 年龄 | 身高 |
| M1 |  |  |  |
| M2 |  |  |  |

孩子表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 孩子ID | 姓名 | 年龄 | 身高 | 母亲ID |
| K1 |  |  |  | M1 |
| K2 |  |  |  | M1 |

## 多对多

多对多：一张表中的一条记录在另外一张表中可以匹配到多条记录，反过来也一样。

多对多的关系如果按照多对一的关系维护：就会出现一个字段中有多个其他表的主键，在访问的时候就会带来不便。

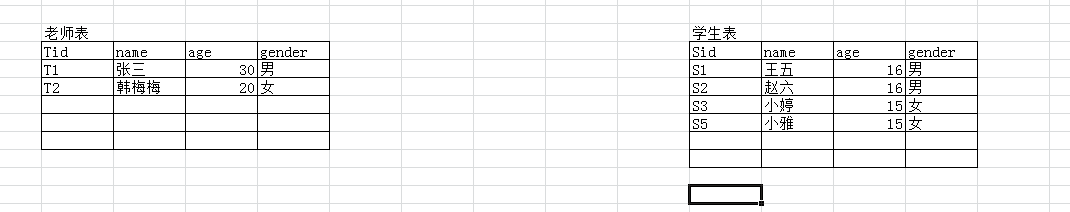
既然通过两张表自己增加字段解决不了问题，那么就通过第三张表来解决。

师生关系

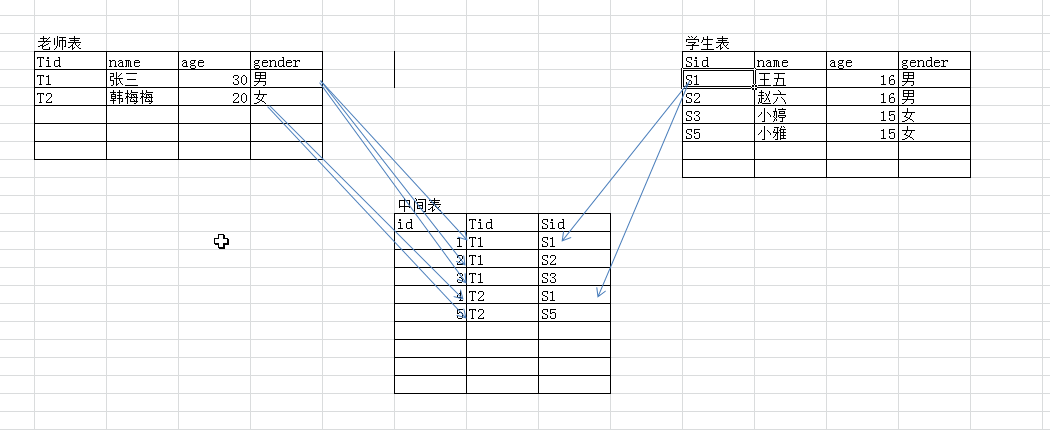
1、 一个老师教过多个班级的学生；

2、 一个学生听过多个老师讲的课；

首先得有两个实体：老师表和学生表



从中间设计一张表：维护两张表对应的联系：每一种联系都包含



多对多解决方案；增加一个中间表，让中间表与对应的其他表形成两个多对一的关系：多对一的解决方案是在“多”表中增加“一”表对应的主键字段。