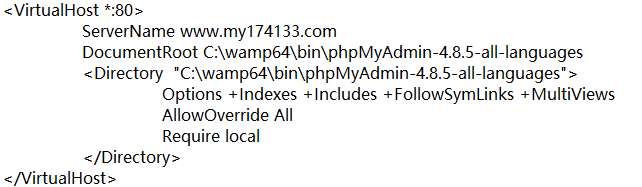
# 搭建PHPmyadmin数据库管理

1. 将phpmyadmin的“站点文件”放置在一个确定位置
2. 打开apache的虚拟主机配置文件：httpd-vhost.conf，并添加站点



<VirtualHost \*:80>

ServerName www.my174133.com

DocumentRoot C:\wamp64\bin\phpMyAdmin-4.8.5-all-languages

<Directory "C:\wamp64\bin\phpMyAdmin-4.8.5-all-languages">

Options +Indexes +Includes +FollowSymLinks +MultiViews

AllowOverride All

Require local

</Directory>

</VirtualHost>

1. 再在host文件添加域名



Extension\_php\_mbstring.dll

**字段类型**

**字符类型**

**MySQL中的字符串，应该使用“单引号”**

**Varchar类型：**

变长度字符串，使用时我们必须设定其长度，最大长度“理论值”**65535个**实际其实UI大只能是6533个，但考虑到存储的字符编码不同也会有进一步减少，比如，

如果存储中文gbk，则最多是65533/2个

如果存储中文utf8，则最多是65533/3个

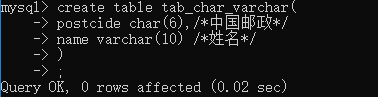
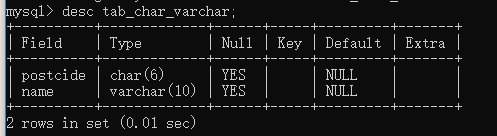
此类型的实际长度是存储内容决定，而设定值只是表示最多可存储的字符个数。

注意：这里的存储限制，其实都来源于一个表格的“一行”的数据存储最大容量限制：65535。

**Char类型**

定长字符串，使用时通常需要设定其长度，不设定 默认为1，最多255个

定长字符串使用于存储数据都是可见的明确的固定长度字符，比如手机号，邮政编码。如果少于设定长度。也能存，但都会补空格填满

**Enum类型**

可以理解为单选项字符串数据类型，他非常适合于存储表单界面中的“单选项值”

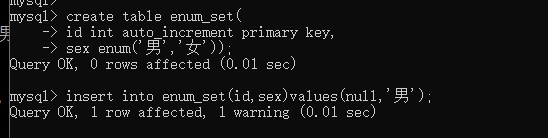
他设定的时候，是需要给定“固定的几个选项”，然后存储的时候，就只存储一个值；

形式如下：

Enum（“选项1”，“选项2”，“选项3”…）

实际内部：

这些字符串选项对应的是如下数字值：1，2，3，4….最多65535个



**Set类型**

可以理解为多选项字符串数据类型，他非常适合于存储表单界面中的“多选项值”

他设定的时候，是需要给定“固定的几个选项”，然后存储的时候，就只存储其中若干个值；

形式如下：

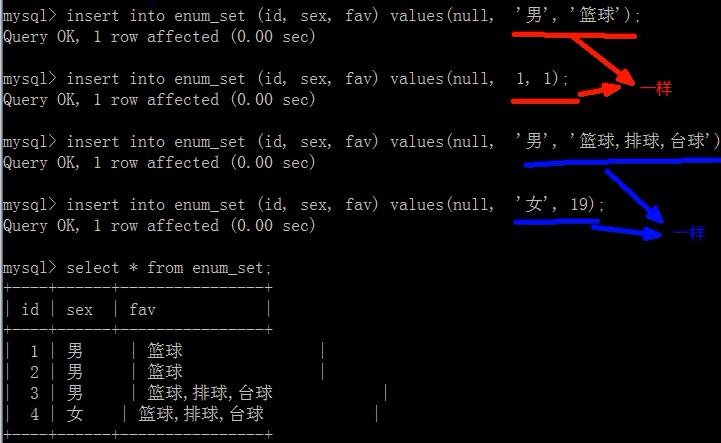
set（“选项1”，“选项2”，“选项3”…）

实际内部：

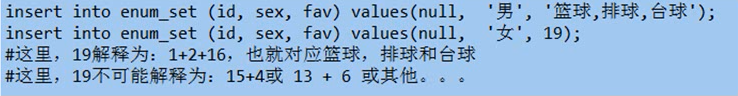
这些字符串选项对应的是如下数字值：1，2，4，8，16….最多64个

写入数据形式：

可以用该选项字符串并用逗号分开本身，也可以用对应的数字的和



解释：



**Text类型**

他成为“长文本”字符类型，通常，其中存储的数据不占据表格的数据容量限制，其本身最长可存储65535个字符。

其他同类字符类型：smalltext，tinyint，longtext

**其他（了解）**

Binary类型：定长二进制字符串类型

Varbinary类型：变长二进制字符串类型

Blob类型：二进制数据类型，通常存的仍然是二进制值，但其适用于存储“图片”，“其他文件”等，但极少用！

**时间类型**

Datetime类型：

时间日期类型

Date类型

日期类型

Time类型

时间类型

Year类型

年份类型

Timestamp类型

时间戳类型：就是指一个时间的“数据值”，类似js中的gettime或php中time，他们都得到的是一个“整数数组”

它的一个重要作用是：他会自动获得时间戳的数据值——相当于now（）；

**在应用中，时间日期类型，在我们自己给定1的数据情形下，需要适用“单引号”，跟字符串一样！**

# 表定义语句

**基本语法形式：**

* **Create table [if not exists] 表名（字段列表【索引或约束列表】）【表选项列表】**
* **或者 and**
* **Create table [if not exists] 表名（字段1，字段2【索引1，索引2】）【表选项1，表选项2】**

**字段设定形式：**

**字段名 类型 【字段属性1 字段属性2】**

**说明：**

1. **字段名可以直接取**
2. **类型就是所学的int tinyint float double varchar ….**
3. **字段属性可以有多个（根据具体需要），相互之间直接空格隔开，主要有：**

**Auto\_increment** 只用于整数类型，让该字段的值自动获得一个增长值。通常用于做一个表的第一个字段的设定，并且通常还当做主键（primary key）

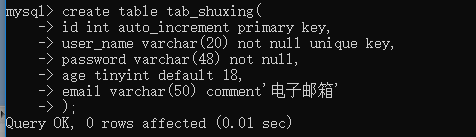
**Primary key**：设定该字段为主键，此时该字段的值就可以“唯一确定”一行数据；

**Unique Key:**该字段唯一的

**Not null**：用于设定该字段不能为空null

**Default xx值**、用于设定该字段的默认值，没有给值就用xx

**Comment‘**字段说明文字’;

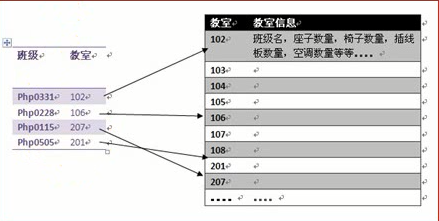




# 索引

什么是索引？

索引是系统内部自动维护的一个‘数据表’，他的作用是：可以极大地加快数据的查找速度！这个隐藏的数据表，其中的数据是自动排好序的，其查找速度就是建立在这个基础上。



通常，所谓建立索引，其实是指定一个表的某个或某些字段作为“索引数据字段”就可以了，形式：

索引类型（要建立索引的字段名）

索引类型有如下几个

普通索引形式：**Key（字段名）**

含义：就是一个索引而已，没有其他作用，只能加快查找速度

唯一索引形式：**unique key（字段名）**

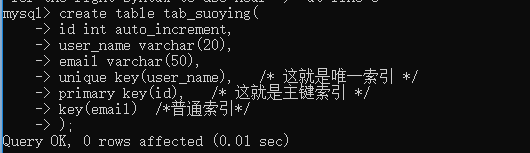
含义：是一个索引而且还可以设定其字段的值不能重复，没有其他作用，加快查找速度

主键索引形式：**primary key （字段名）**

含义：是一个索引而且还具有区分该表中的任何一行数据的作用（其实也是唯一性），他其实比唯一索引多一点功能，唯一性可以为空null，而主键不能为空，只能加快查找速度

全文索引形式：**fulltext（字段名）**

外键索引形式：**foreign key（字段名）references 其他表（对应其他表中的字段名）**



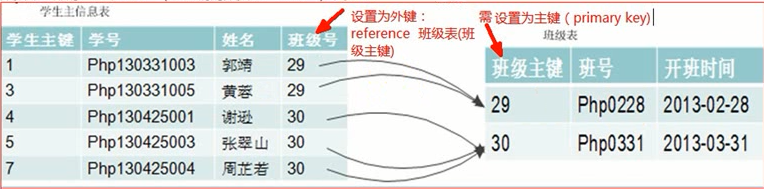
此时，该表中如果以id，user\_name，或email做条件进行查找，就会“很快”，而以age做条件就会“很慢”。

**外键索引**

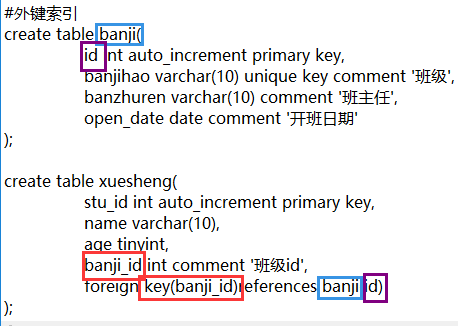
**什么叫外键？**

**就是指，设定某个表中某个字段，他的数据的值，必须在另一个表中的某一个字段中存在！**

外键索引形式：**foreign key（字段名）references 其他表（对应其他表中的字段名）**

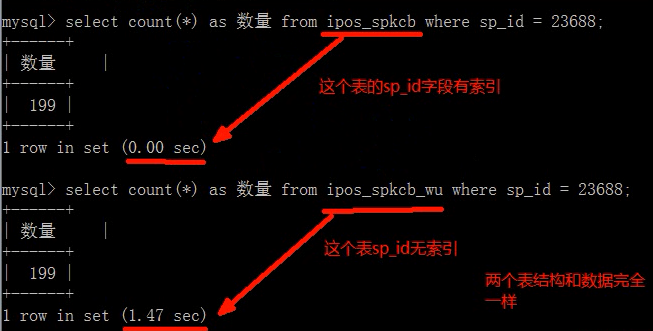


**示例！！！**



**此时，插入xuesheng表中的数据时，banji\_id字段的值，就不可以随便写入了，而必须是banji表中的id字段所已经有的数据值，才可以插入**

# 百万级数据有索引和无索引的查找速度



# 约束

## 什么叫约束？

就是要求数据需要满足什么条件的一种“规定”。

主要有如下几种约束：

主键约束:

形式：primary key（字段名）

含义：使该设定字段的值可以用于“唯一确定一行数据”，其实就是“主键的意思”

唯一约束

形式：unique key（字段名）

含义：使该设定字段的值具有“唯一性”（自然也是可区分）

外键约束

形式：foreign key（字段名）reference其他表名（对应其他表中的字段名）

含义：使该设定字段的值，必须是其谁定的对应表中的对应字段中已经有该值了。

非空约束：

形式：not null

含义：其实就是设定一个字段时写的那个“not null”属性

默认约束：这个约束只能写在字段属性上

形式：default xx值

含义：其实就是设定一个字段时写的那个default默认值属性；这个约束只能写在字段属性上

检查约束：

形式：check（某种判断语句）比如

create table tab1（

age tinyint，

check （age>=0 and age<100）/\* 这就是检查约束 \*/

）

不然就能-255-127

其实，主键约束，唯一约束，外键约束，只是“同一件事情的2个不同说法”，他们同时也称为“主键索引”，“唯一索引”，“外键索引”

# 表选项列表

表选项就是，创建一个表的时候，对该表的整体设定，主要有：

Charset =要使用的字符编码 默认数据库级别

Engine=要使用的存储引擎（表类型） 默认为innoDB

Auto\_increment=设定当前表的自增长字段的初始值，默认1；

Comment=该表的一些说明文字

说明：

1. 设定的字符编码是为了跟数据库设定的不一样，如果一样，就不需要设定了，因为其会自动使用数据库级别的设定；
2. Engine（存储引擎）在代码层面，就是几个名词：innoDB，myisam，BDB，archive，memory；默认是innoDB

什么是存储引擎？

存储引擎是将数据存储到硬盘的“机制”；其实，也就几种机制（如上名字所述）：

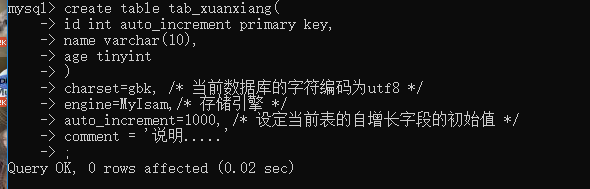
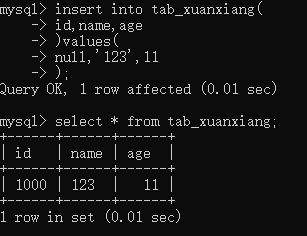
不同的存储引擎，其实主要是从2个大的层面来设计存储机制；

1. 尽可能快的速度
2. 尽可能多的功能

选择不同的存储引擎，就是上述性能和功能的“权衡”

大体如下：



. 

# 修改表：

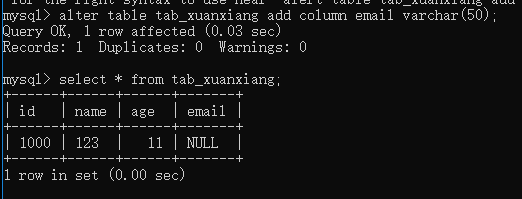
几点说明：

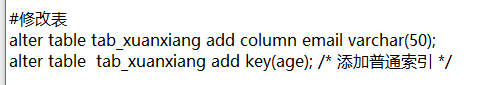
1. 修改表，是指修改表的结构——正如创建表也是设定表结构.
2. 创建表能做的事，修改表几乎都能做——但很不推荐去修改表，而是应该在创建表的时候就基本确认表的结构
3. 大体来说有：
   1. 可以对字段进行：添加，删除，修改
   2. 可以对索引进行：添加，删除
4. 表的选项，通常“都是修改”，即不写任何表选项，他们都有其默认值；

常见几个：

* 添加字段：alter table 表名 add [column] **新字段名 字段类型 [字段属性列表]**；
* 修改字段（并可改名）：alter table 表名 change [column] 旧字段名 **新字段名 新字段类型 [新字段属性列表]**；
* 删除字段：alter table 表名 drop [column] 字段名；
* 添加普通索引：alter table 表名 add key [索引名] (字段名1[，字段名2,...])；
* 添加唯一索引(约束)：alter table 表名 add unique key (字段名1[，字段名2,...])；
* 添加主键索引(约束)：alter table 表名 add primary key (字段名1[，字段名2,...])；
* 修改表名：alter table 旧表名 rename [to] 新表名；

演示：



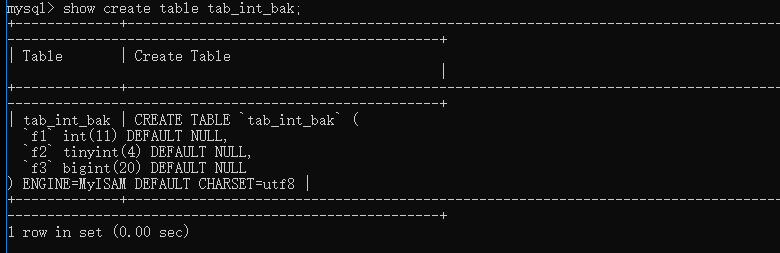


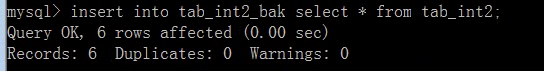
# 删除表

Drop table 【if exitss】 表名

## 其他表的相关语句：

* 显示当前数据库中的所有表: show tables；
* 显示某表的结构: desc 表名； 或：describe 表名；
* 显示某表的创建语句：show create table 表名；
* 重命名表：rename table 旧表名 to 新表名；
* 从已有表复制表结构：create table [if not exists] 新表名 like 原表名;



表的完整复制

# 视图定义语句.( view)

什么叫视图？

类比：什么叫做函数？

就是一段代码。我把他封装起来，并给一个名字，以后，要使用该段代码，就方便了，使用函数名就可以了

视图：

就是一个select语句（通常比较复杂），我们给其一个名字（视图名），以后要使用（执行）该select语句，就方便了，使用视图名

视图的创建语法：

语法形式：

create view 视图名 【（字段名1，字段名2，字段名3，....）】 as select语句；

举例：

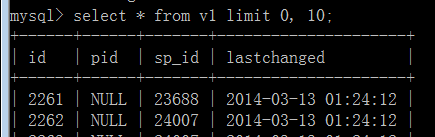
**create view v1 as**

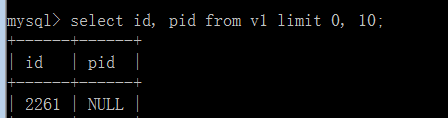
**select id, f1, name, age, email, p\_id, f3 from 表1 where id > 7 and id < 100 or f1 < 1000 and age > 10**

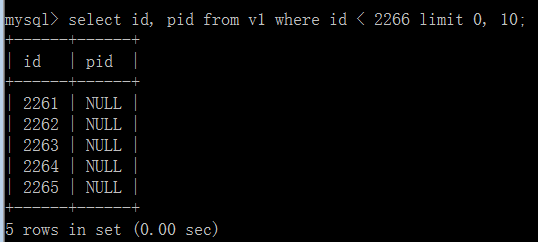
## 使用视图：

基本上，当做一个表用就了！

比如：







## 删除视图：

drop view 【if exists】 视图名；