[第一章、数据基础 4](#_Toc39566133)

[第二章、Mysql的安装与配置 4](#_Toc39566134)

[2.1 Mysql的下载和安装 4](#_Toc39566135)

[2.1.1下载Mysql 4](#_Toc39566136)

[2.2.2 安装Mysql 6](#_Toc39566137)

[2.2 mysql的卸载 12](#_Toc39566138)

[2.3 服务列表中启动停止Mysql服务 13](#_Toc39566139)

[2.4 Mysql图形化管理 14](#_Toc39566140)

[2.4.1下载 14](#_Toc39566141)

[2.4.2安装 14](#_Toc39566142)

[2.4.3.激活 17](#_Toc39566143)

[2.4.4使用 18](#_Toc39566144)

[第三章、数据库的基本操作 20](#_Toc39566145)

[3.1创建数据库 20](#_Toc39566146)

[3.1.1 基本语法： 20](#_Toc39566147)

[3.1.2.参数说明： 20](#_Toc39566148)

[3.2查看数据库 22](#_Toc39566149)

[3.2.1.基础语法： 22](#_Toc39566150)

[3.3选择数据库 23](#_Toc39566151)

[3.3.1.基础语法： 23](#_Toc39566152)

[3.4删除数据库 23](#_Toc39566153)

[3.4.1基础语法 24](#_Toc39566154)

[3.4.2 参数说明 24](#_Toc39566155)

[3.5 数据库存储引擎 25](#_Toc39566156)

[3.5.1 Mysql存储引擎 25](#_Toc39566157)

[3.5.2 InnoDb存储引擎 25](#_Toc39566158)

[3.5.3 MyISM存储引擎 25](#_Toc39566159)

[3.5.4 MEMORY存储引擎 25](#_Toc39566160)

[第四章、数据表的基本操作 25](#_Toc39566161)

[4.1数据类型 25](#_Toc39566162)

[4.1.1数字 26](#_Toc39566163)

[4.1.2.日期和时间 26](#_Toc39566164)

[4.1.3.字符串 26](#_Toc39566165)

[4.2创建数据表 26](#_Toc39566166)

[4.2.1.基础语法 26](#_Toc39566167)

[4.3查看表结构 28](#_Toc39566168)

[4.3.1查看基本结构语句 28](#_Toc39566169)

[4.3.2查看详细结构语句 28](#_Toc39566170)

[4.4修改数据表 28](#_Toc39566171)

[4.4.1修改表名 28](#_Toc39566172)

[4.4.2修改字段的数据类型 29](#_Toc39566173)

[4.4.3修改字段名 30](#_Toc39566174)

[4.4.4添加字段 31](#_Toc39566175)

[4.4.5修改字段的排列位置 32](#_Toc39566176)

[4.4.6删除字段 33](#_Toc39566177)

[4.5数据完整性约束 34](#_Toc39566178)

[4.5.1主键约束 35](#_Toc39566179)

[4.5.2唯一性约束 37](#_Toc39566180)

[4.5.3非空约束 38](#_Toc39566181)

[4.5.4默认约束 39](#_Toc39566182)

[4.5.5字段值自动增加 39](#_Toc39566183)

[4.5.6外键约束 40](#_Toc39566184)

[第五章、表数据的增删改操作 42](#_Toc39566185)

[5.1 插入数据 42](#_Toc39566186)

[5.1.1插入完整数据 42](#_Toc39566187)

[2.插入部分数据 42](#_Toc39566188)

[5.1.3插入多记录 43](#_Toc39566189)

[5.2修改数据 43](#_Toc39566190)

[5.3删除数据 44](#_Toc39566191)

[5.3.1.通过delete语句删除数据 44](#_Toc39566192)

[5.3.2通过truncate table语句删除数据 45](#_Toc39566193)

[第六章、数据的查询 45](#_Toc39566194)

[第七章、视图 53](#_Toc39566195)

[第八章、索引 56](#_Toc39566196)

[第九章、存储过程和触发器 58](#_Toc39566197)

[第十章、事务 62](#_Toc39566198)

[第十一章、数据安全 63](#_Toc39566199)

# 第一章、数据基础

1. 数据概述
2. 数据模型

# 第二章、Mysql的安装与配置

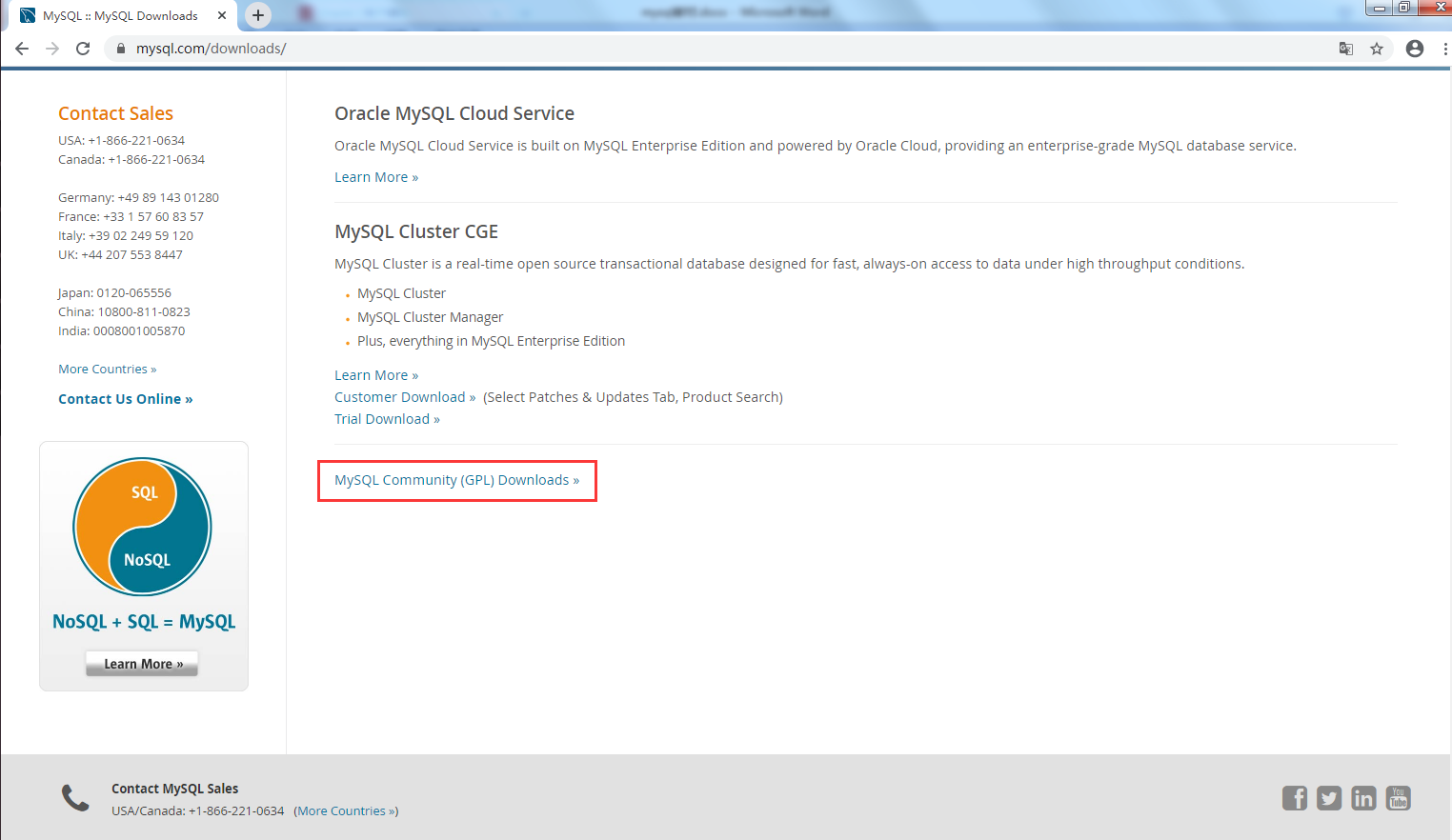
## 2.1 Mysql的下载和安装

### 2.1.1下载Mysql

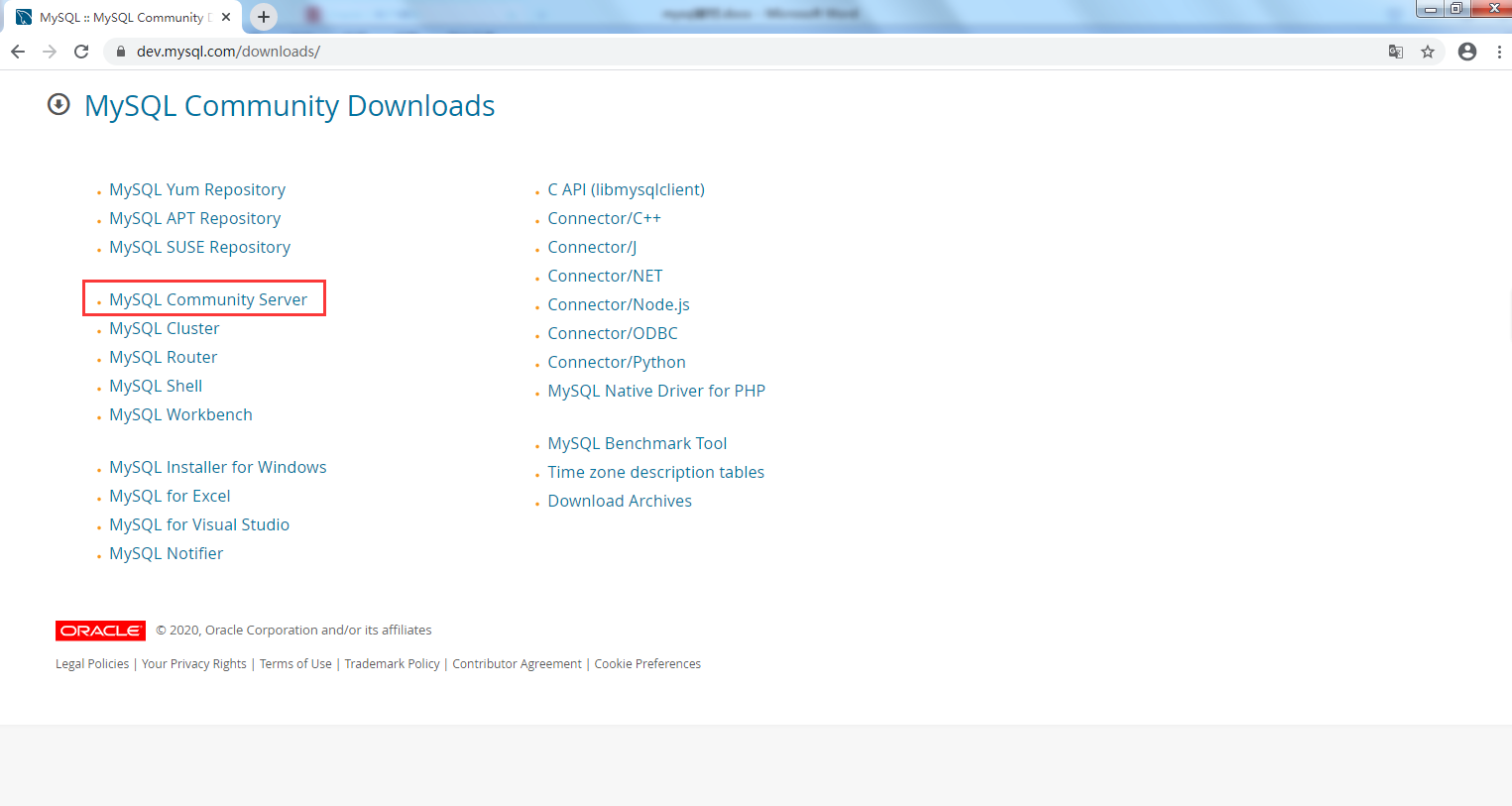
官网地址：<https://www.mysql.com/>

Mysql针对个人用户和商业用户提供不同版本的产品。Mysql Community Edition（社区版）提供个人用户免费下载，而对于商业用户，有标准版，企业版，集成版等多个版本可供选择，以满足特殊的商业和技术需求。

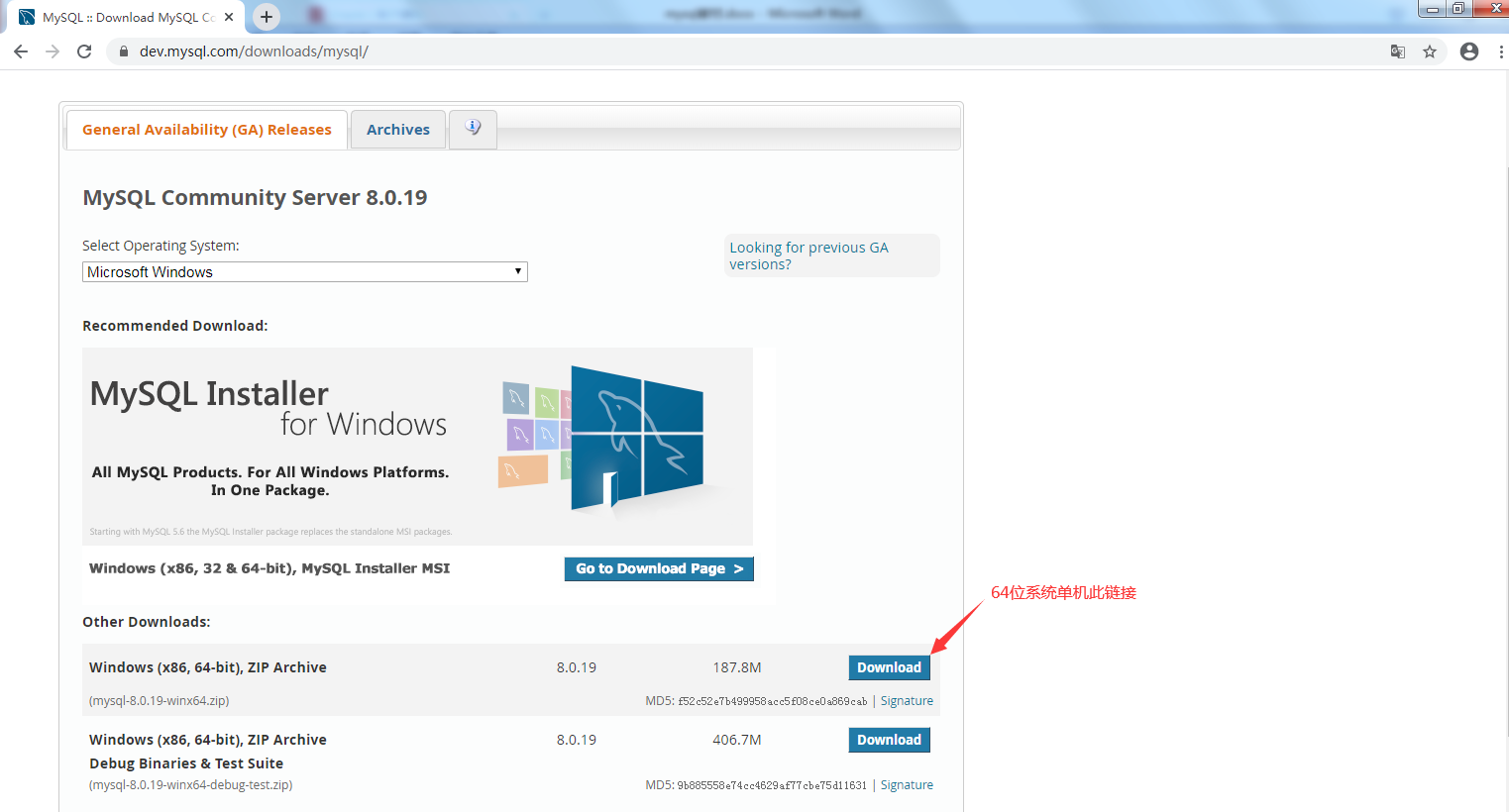
Mysql是开源软件，个人用户可以登录其官网地址下载相应的版本，登录Mysql （如果没有Mysql官网的账号密码，则请先注册），点击Downloads导航栏，将页面滚动至底部，如图所示：



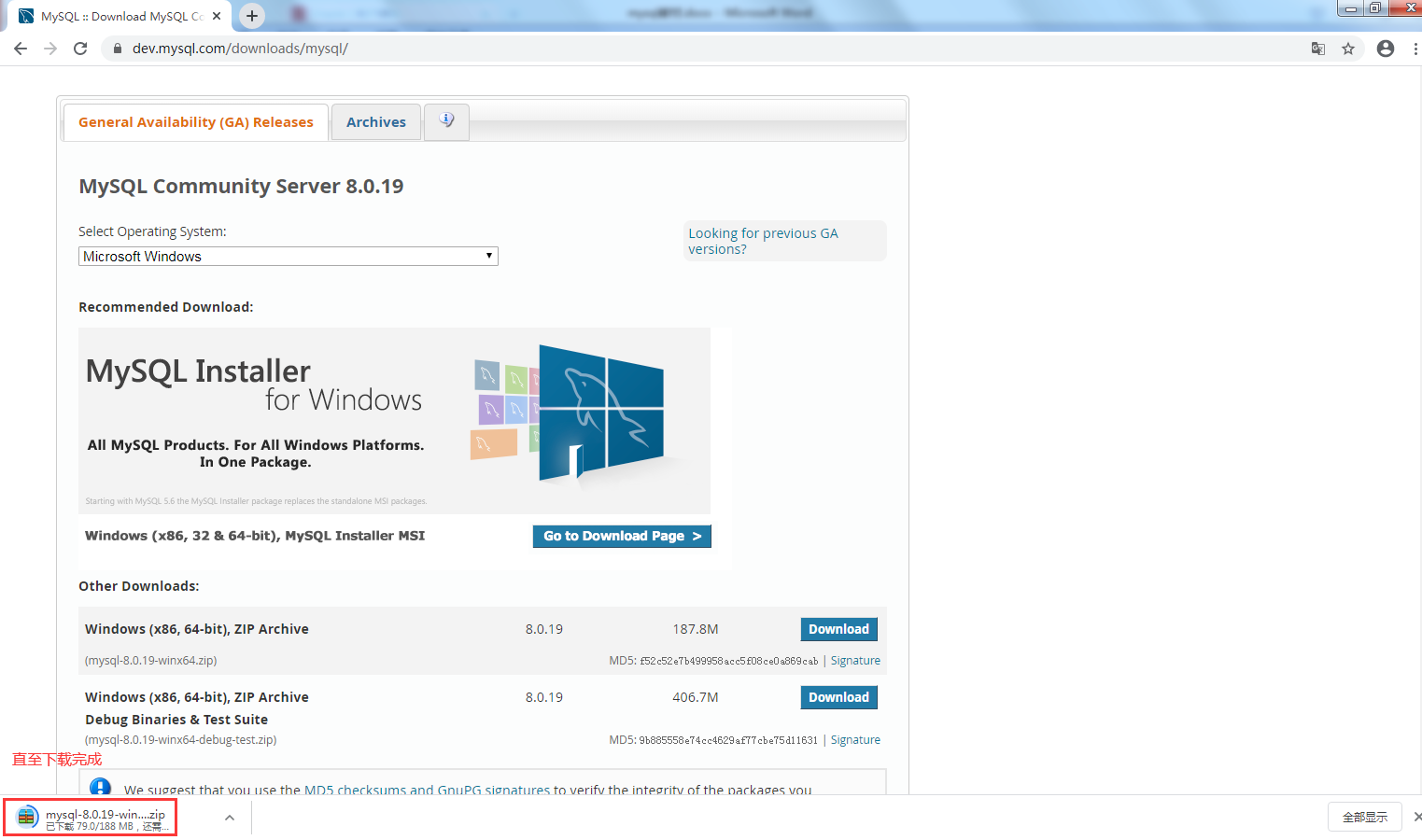
单击以上的超链接，进入社区版界面，如图所示：



单击以上的超链接，滚动至以下界面，如图所示：



单击Download按钮，开始下载，直至下载完成。

****

### 2.2.2 安装Mysql

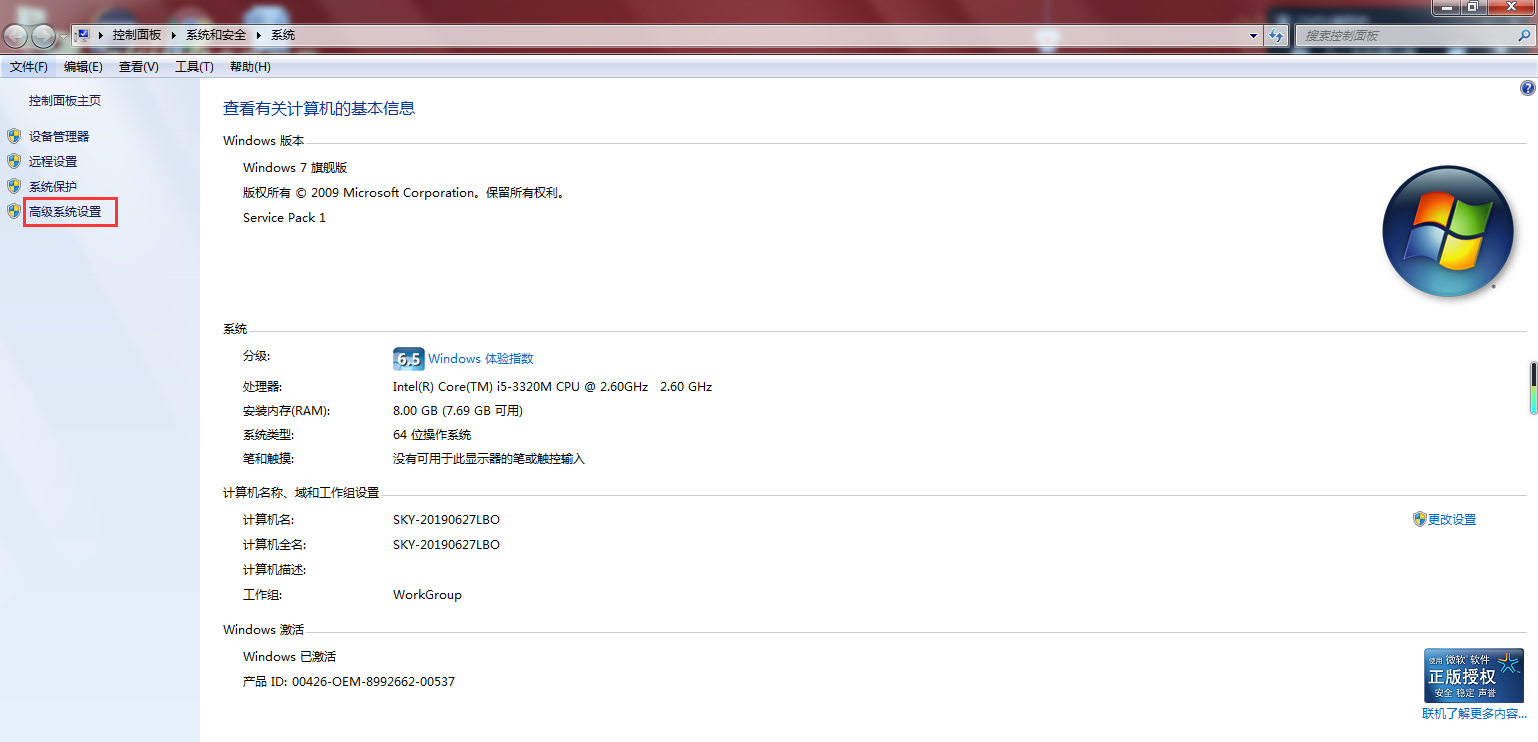
1.解压

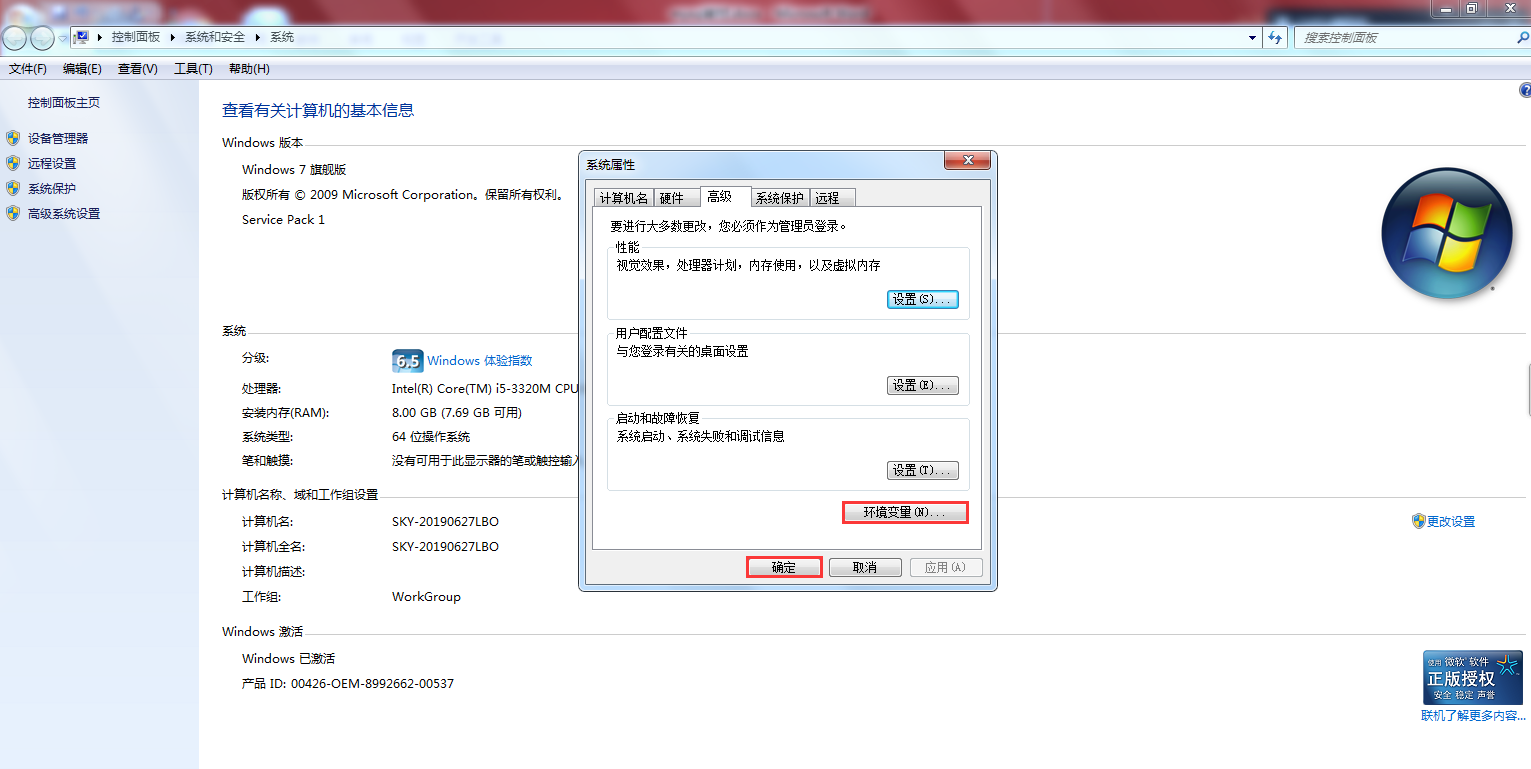
将解压下载得到的mysql-8.0.19-winx64.zip 安装文件到自己习惯的软件安装目录，比如D:\soft，解压完成后，在D:\soft目录下可以看到多了一个mysql-8.0.19-winx64的文件夹即安装成功，但是还需要做初始化配置

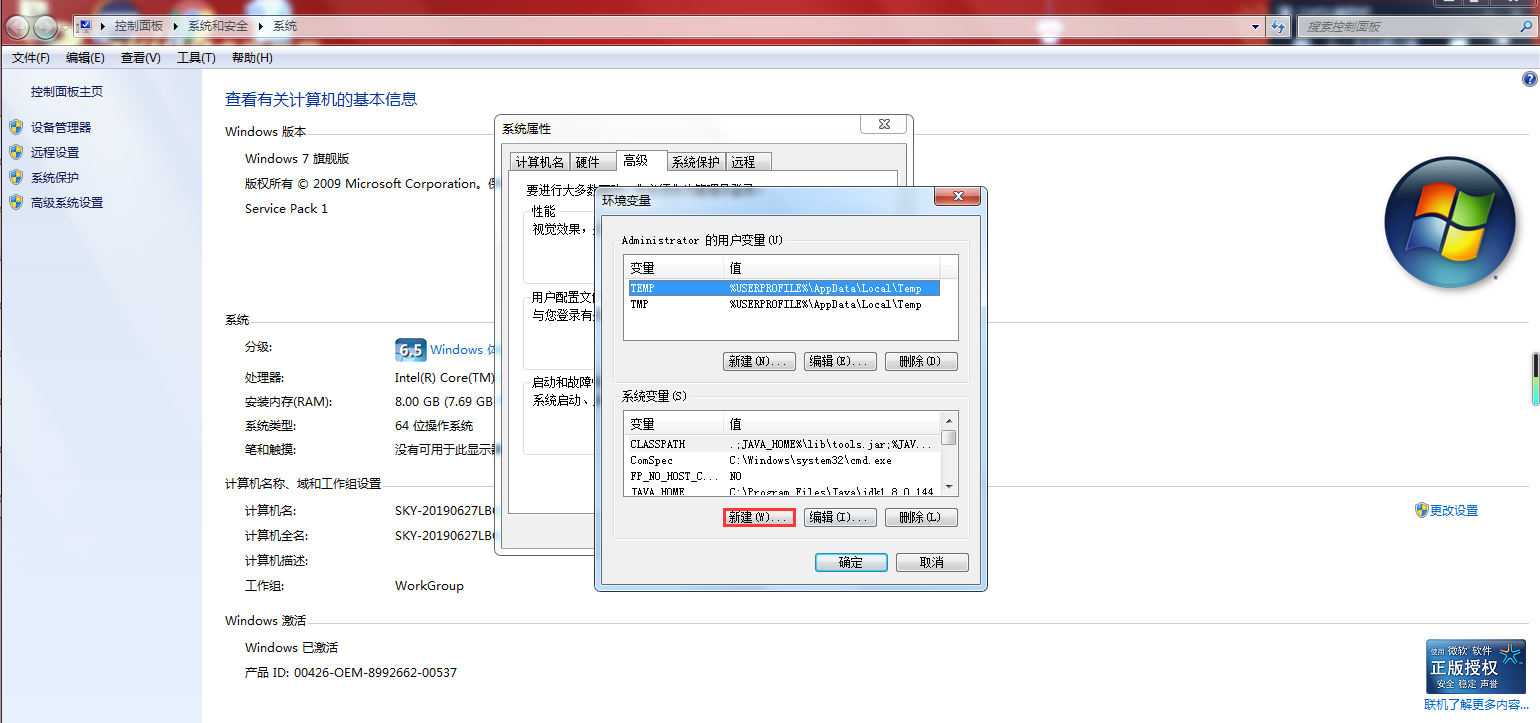
2.配置环境变量

计算机----》右键，属性----》高级系统设置----》坏境变量----》系统变量

1) 新建变量MYSQL\_HOME





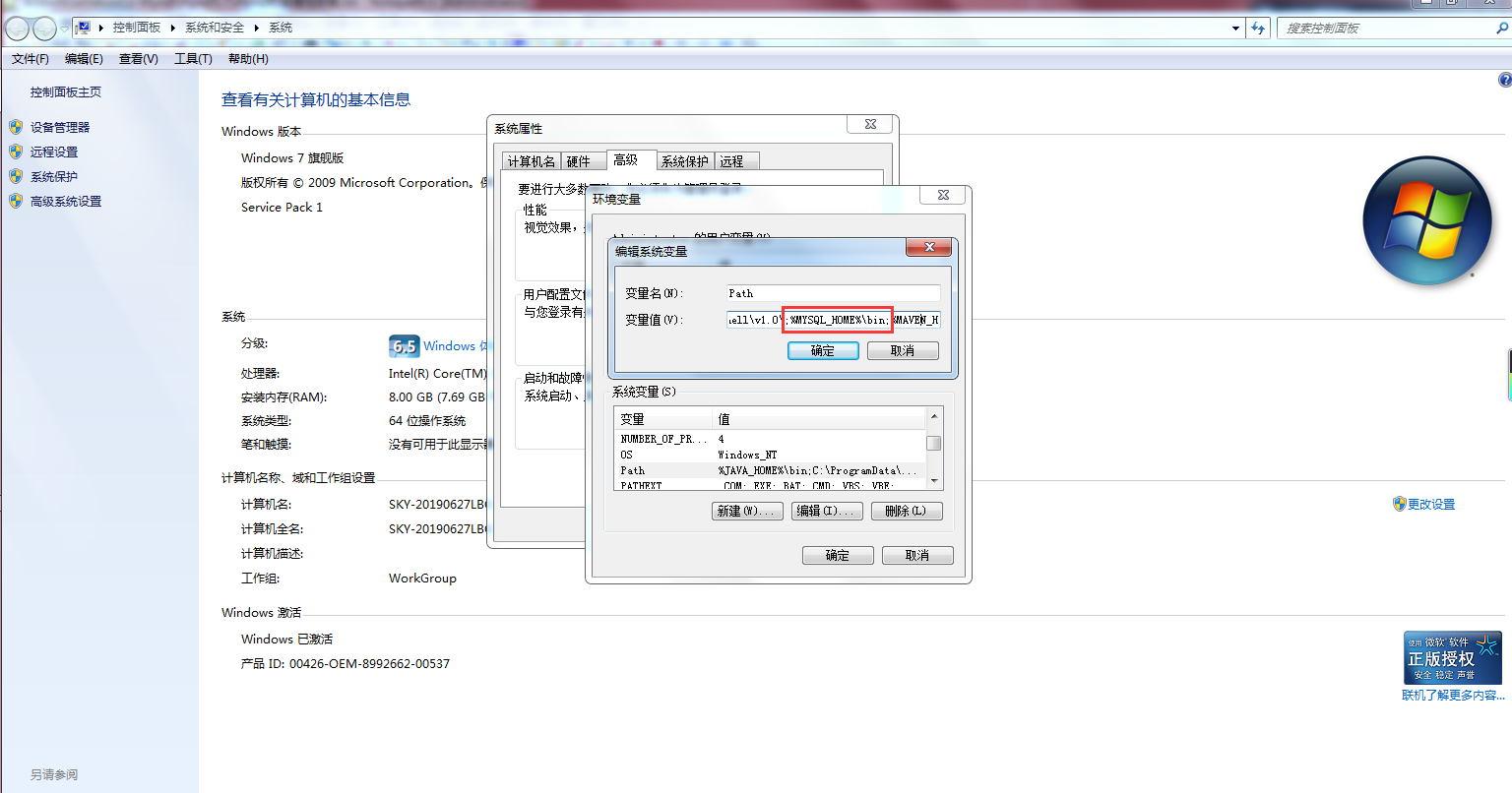




2）编辑变量Path

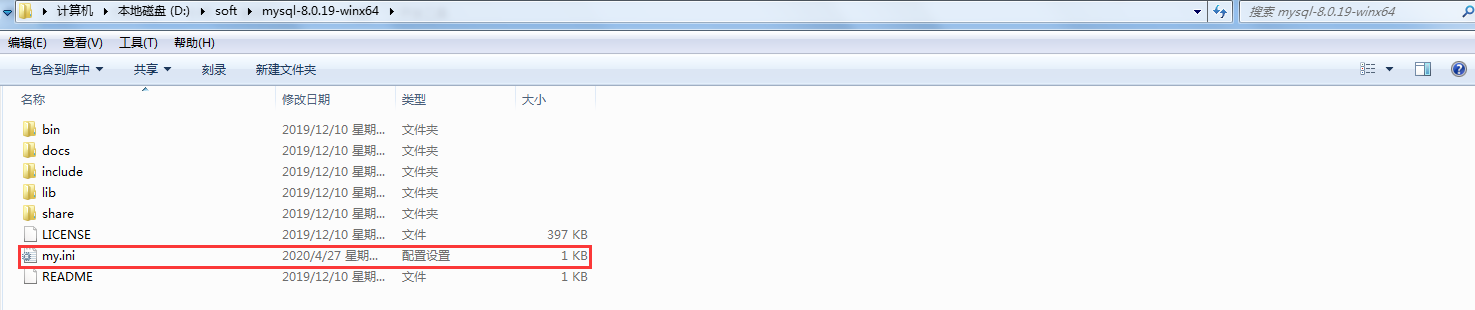
在系统变量中找到Path变量，在变量值的输入框中，输入“%MYSQL\_HOME%\bin;”

注意：如果前面有值，则记得在“%MYSQL\_HOME%\bin;”前添加分号”;”

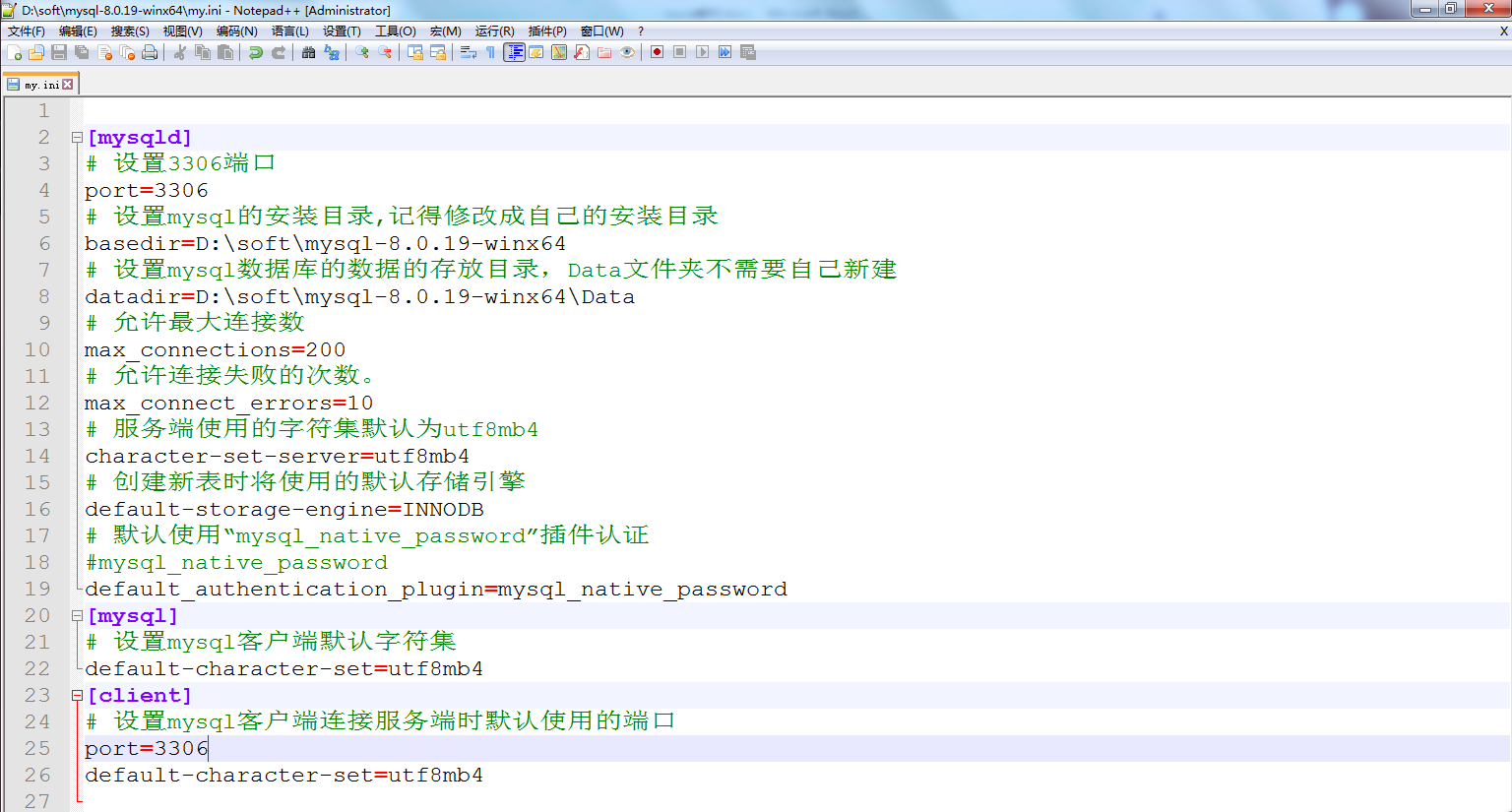


3. 新建my.ini文件

在Mysql的安装目录下新建my.ini文件：

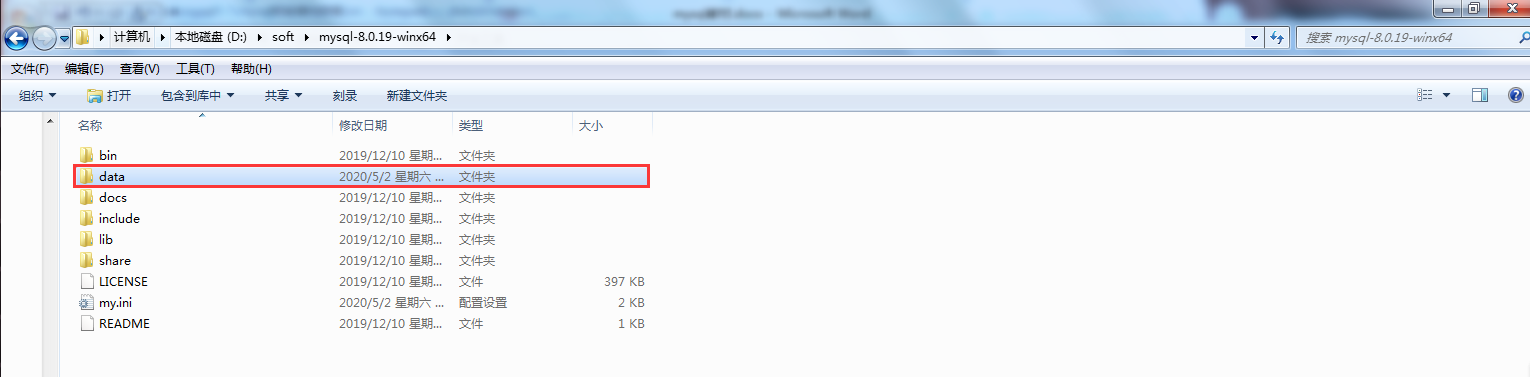


内容如下：



4.初始化

Windows搜索框中搜索”cmd”以管理员身份打开黑窗口，并进入Mysql的安装目录下的bin目录，输入命令：mysqld --initialize –console，在控制台输出了随机密码，在mysql的安装目录中也出现了data文件夹，证明初始化成功



5.安装Mysql

命令：mysqld –install

6.启动MySQL服务

命令：net start mysql

7.设置密码

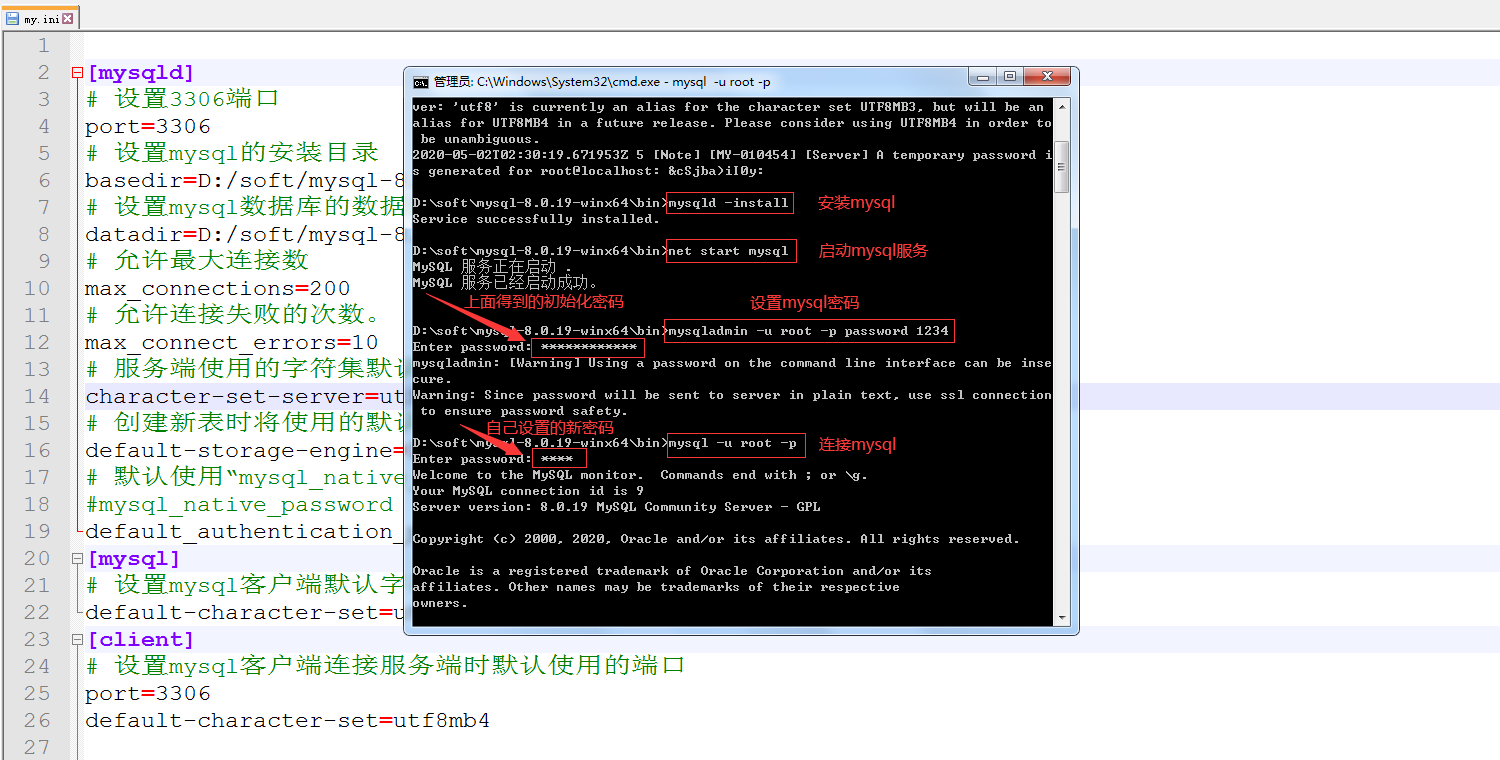
命令：mysql admin -u root -p password 新密码

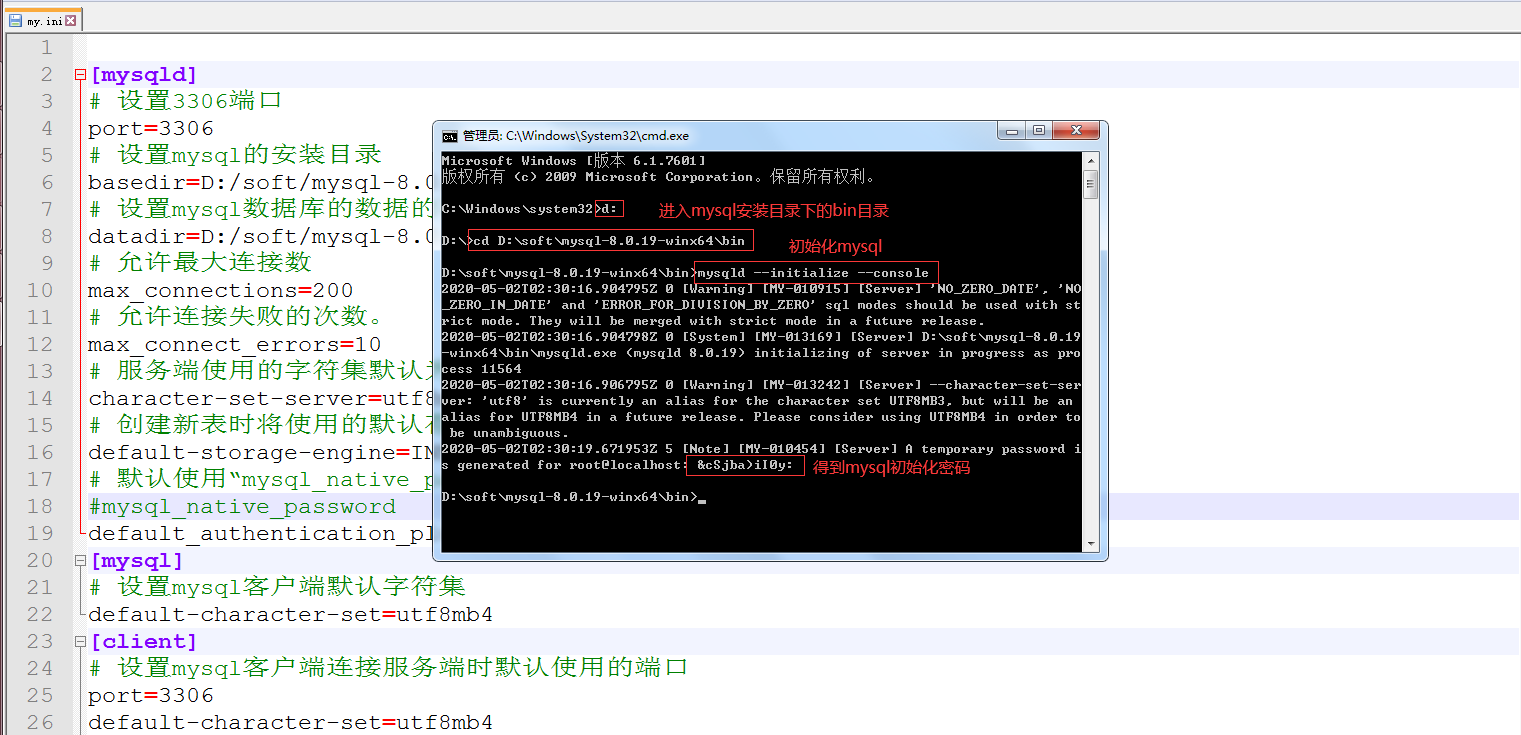
注意：需要输入旧密码，即：初始化mysql时，控制台输出的随机字符串密码

8. 连接 mysql

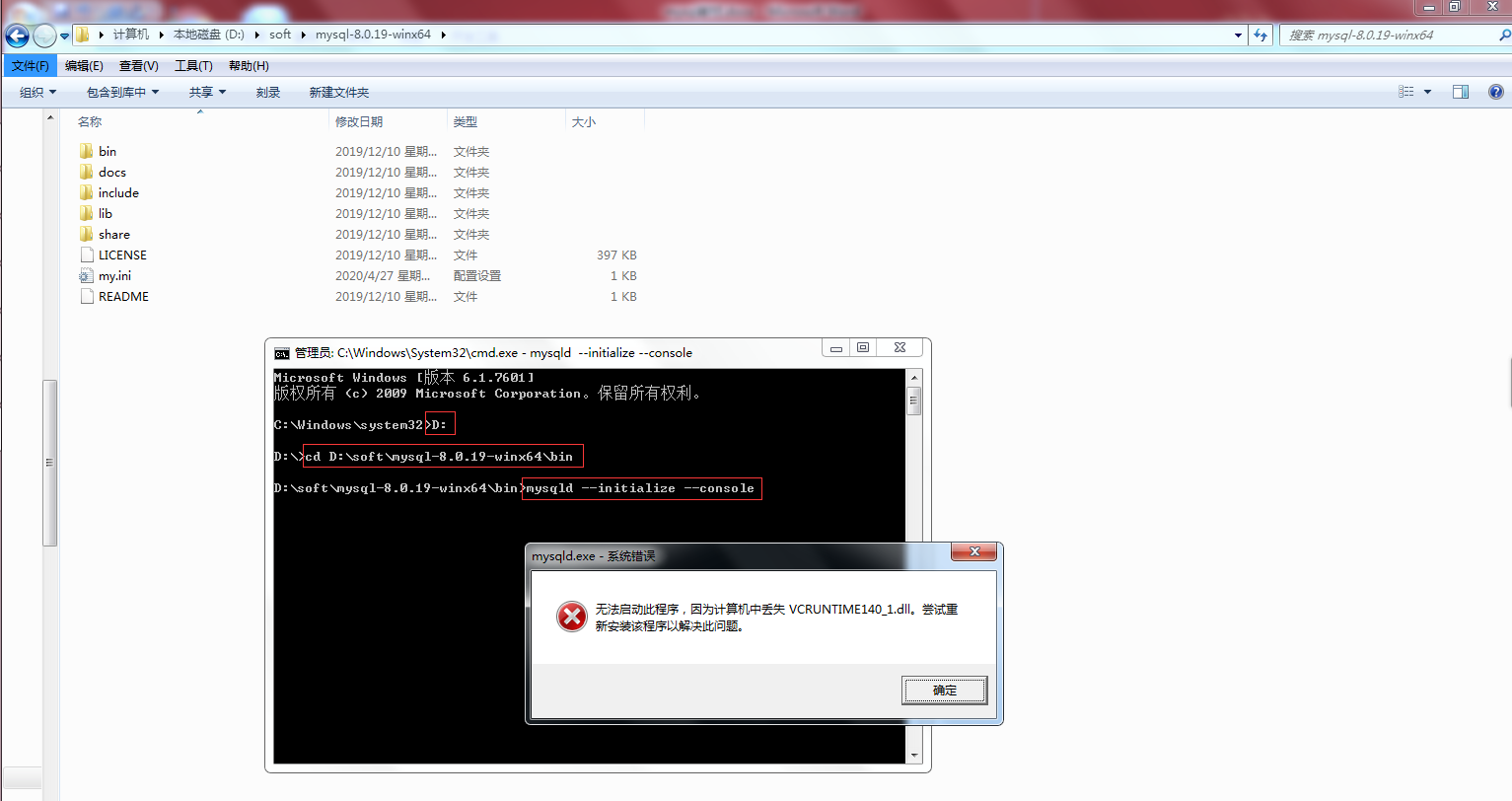
命令：mysql -u root –p

安装mysql操作演示如下图：





安装可能报错：



解决方案：

下载安装VC\_redist.x64.exe即可（根据自己的系统找64位或者32位）

9.断开mysql连接

1）quit

2）exit

10.命令操作mysql

1）显示数据库 show databases;

2）新建数据库 create database mydatabase;

3）使用数据库 use mydatabase;

4）新建表

5）显示表 show tables；

6）CRUD操作

## 2.2 mysql的卸载

1.停止mysql服务

命令：net stop mysql

2.删除mysql

但是服务只是禁用状态，服务还是存在。

命令：sc delete mysql

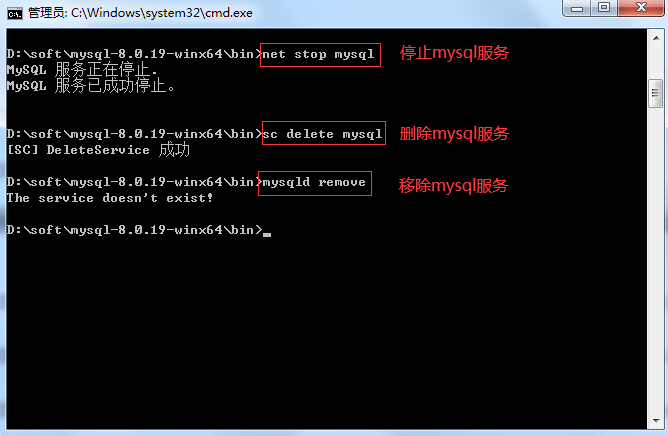
3.移除mysql

在服务列表中已经找不到mysql服务了

命令：mysqld remove

注意：需要进入bin目录中操作

停止、删除、移除mysql服务操作如下图：



4.删除注册表信息

在windos搜索框中搜索” regedit”，打开注册表。

根据如下节点找到一级级找到mysql服务，然后右键删除。

HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\Eventlog\Application\MySQL

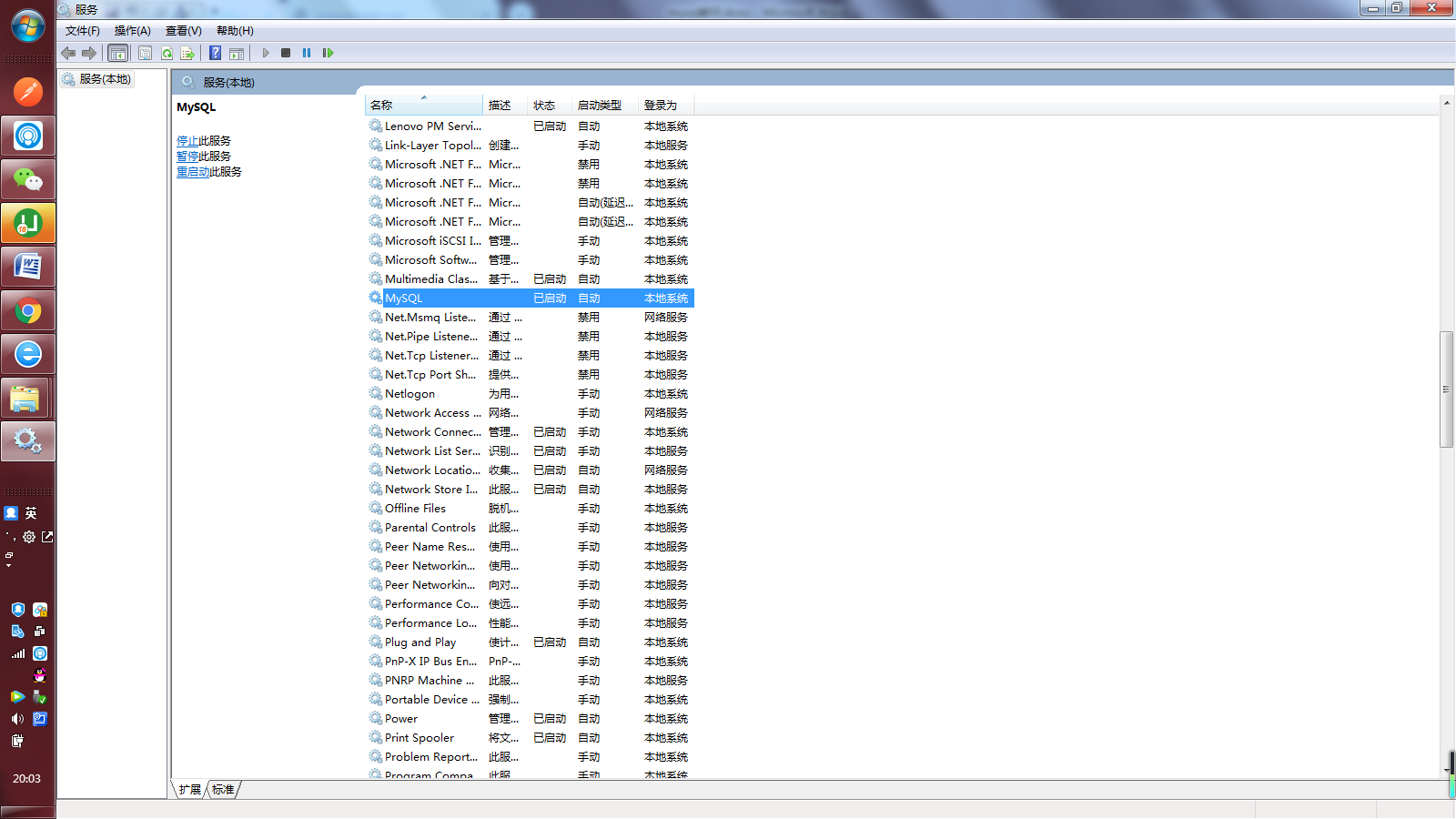
HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services\Eventlog\Application\MySQL

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application\MySQL

## 2.3 服务列表中启动停止Mysql服务

1. 启动停止Mysql服务

Window系统的搜索框中搜索“服务”，打开如图所示的界面，滚动至找到Mysql服务



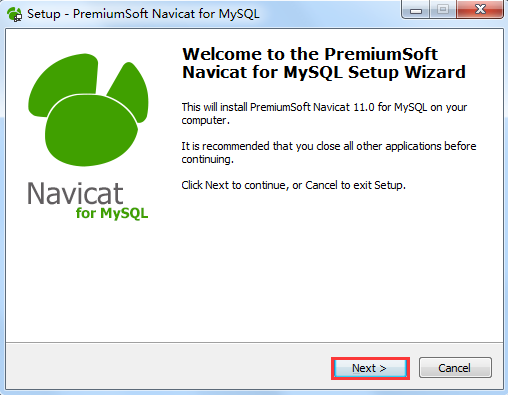
## 2.4 Mysql图形化管理

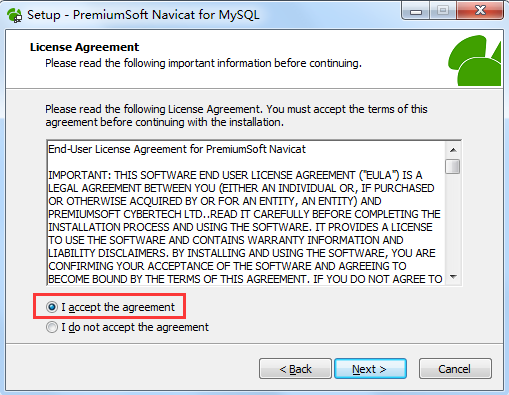
### 2.4.1下载

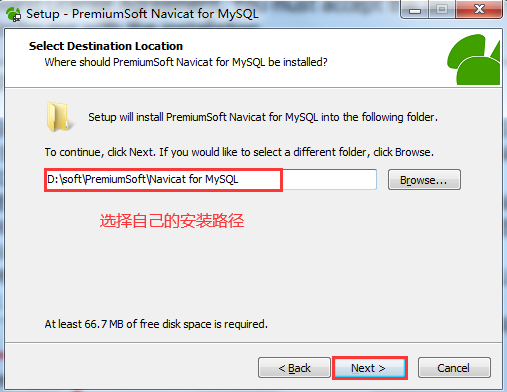
<https://navicatformysql.en.softonic.com/download>

### 2.4.2安装

将下载得到的navicat110\_mysql\_en\_x64.exe文件双击按提示安装

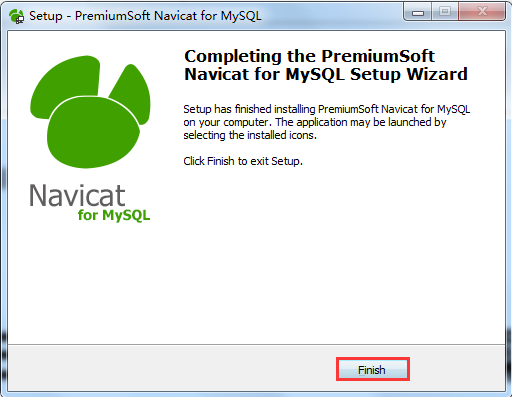






一直Next，直到install安装即可。

最后：



在桌面上即可看到绿色的navicat for mysql 的图标

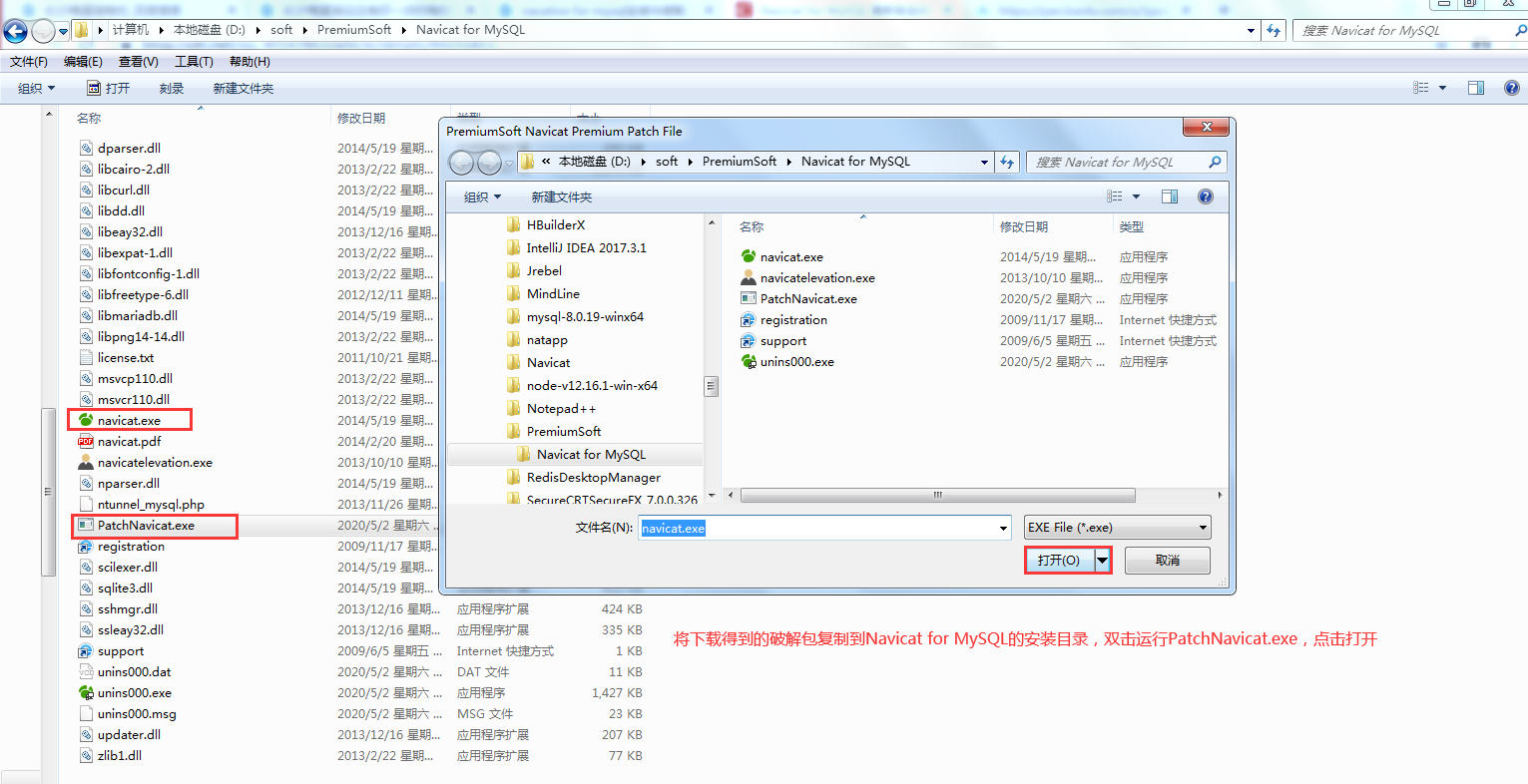


双击打开Navicat for MySQL

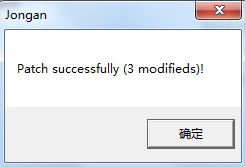
### 2.4.3.激活

第一次双击Navicat，提示只有30天的试用期，所以需要激活或者破解

百度下载破解文件PatchNavicat.exe，双击破解文件PatchNavicat.exe，点击“打开”



点击“确定”即可



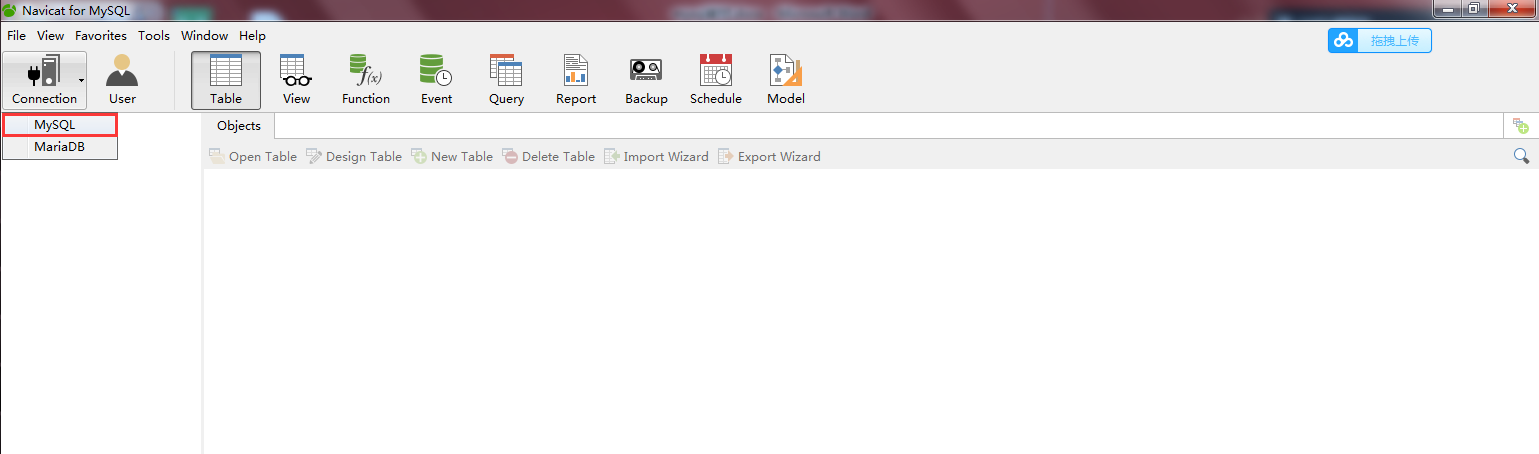
再次进入navicat则没有试用30天的提示了。

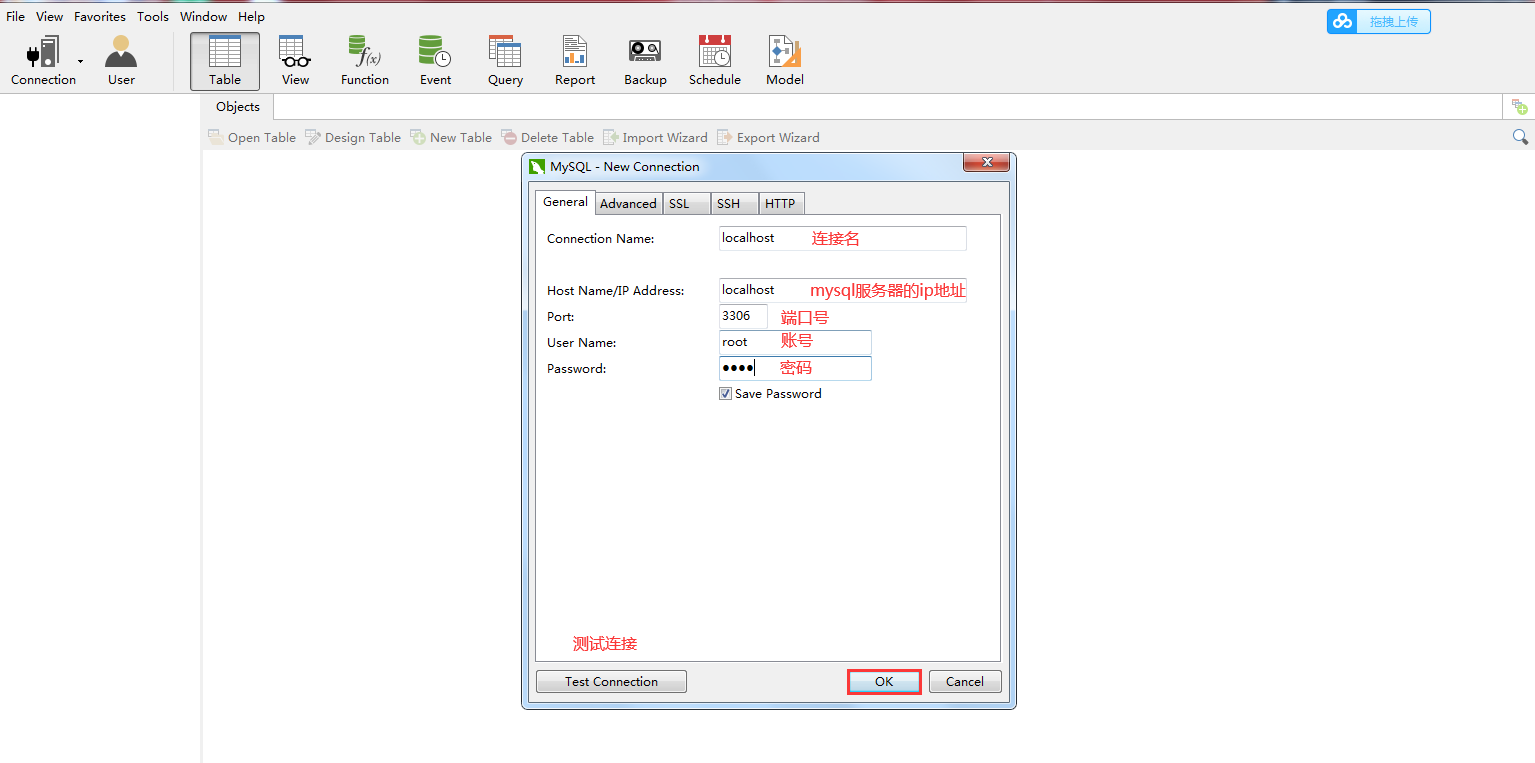
### 2.4.4使用

1.创建连接

Connect----》MySQL

操作演示如图：

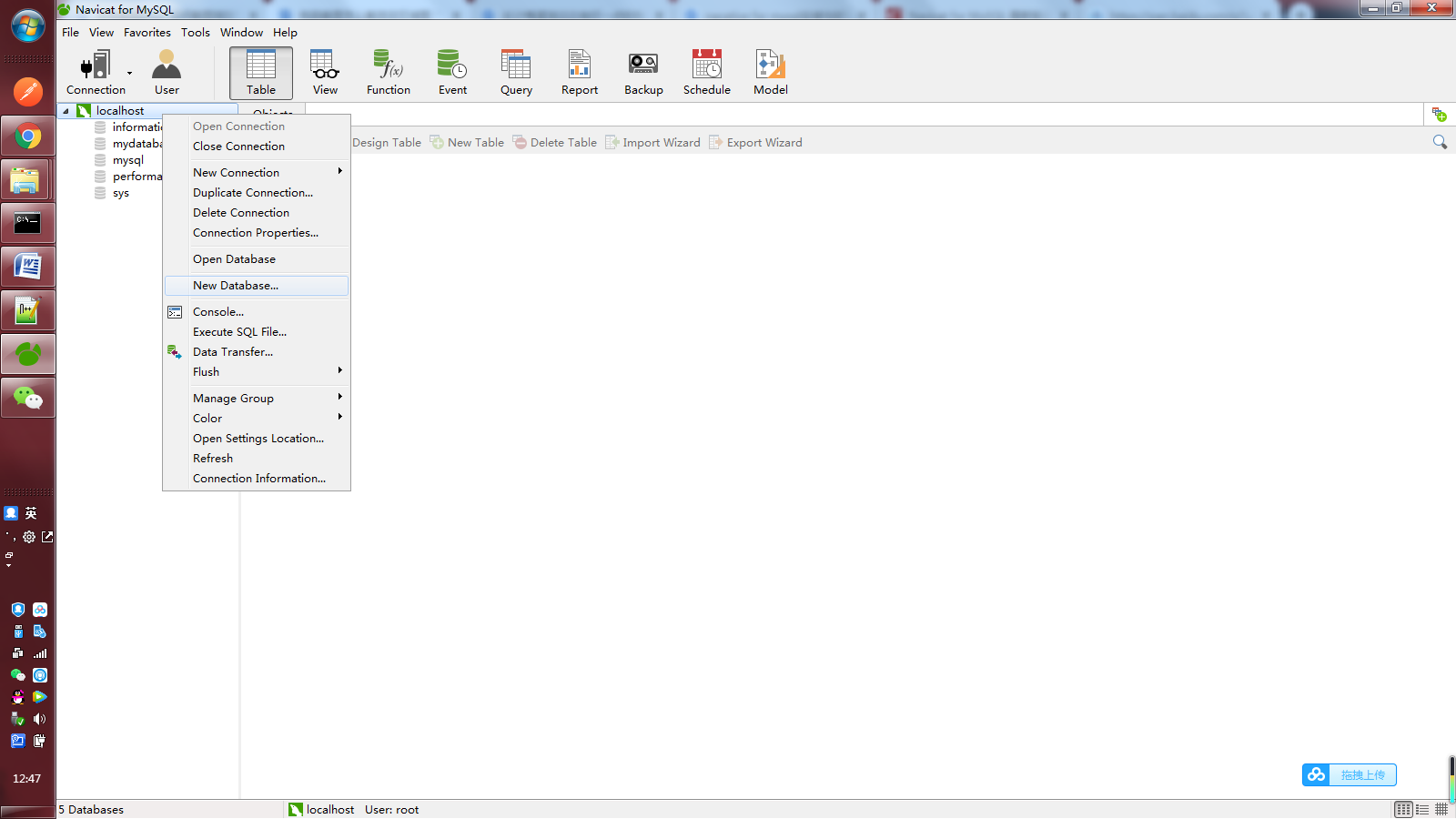


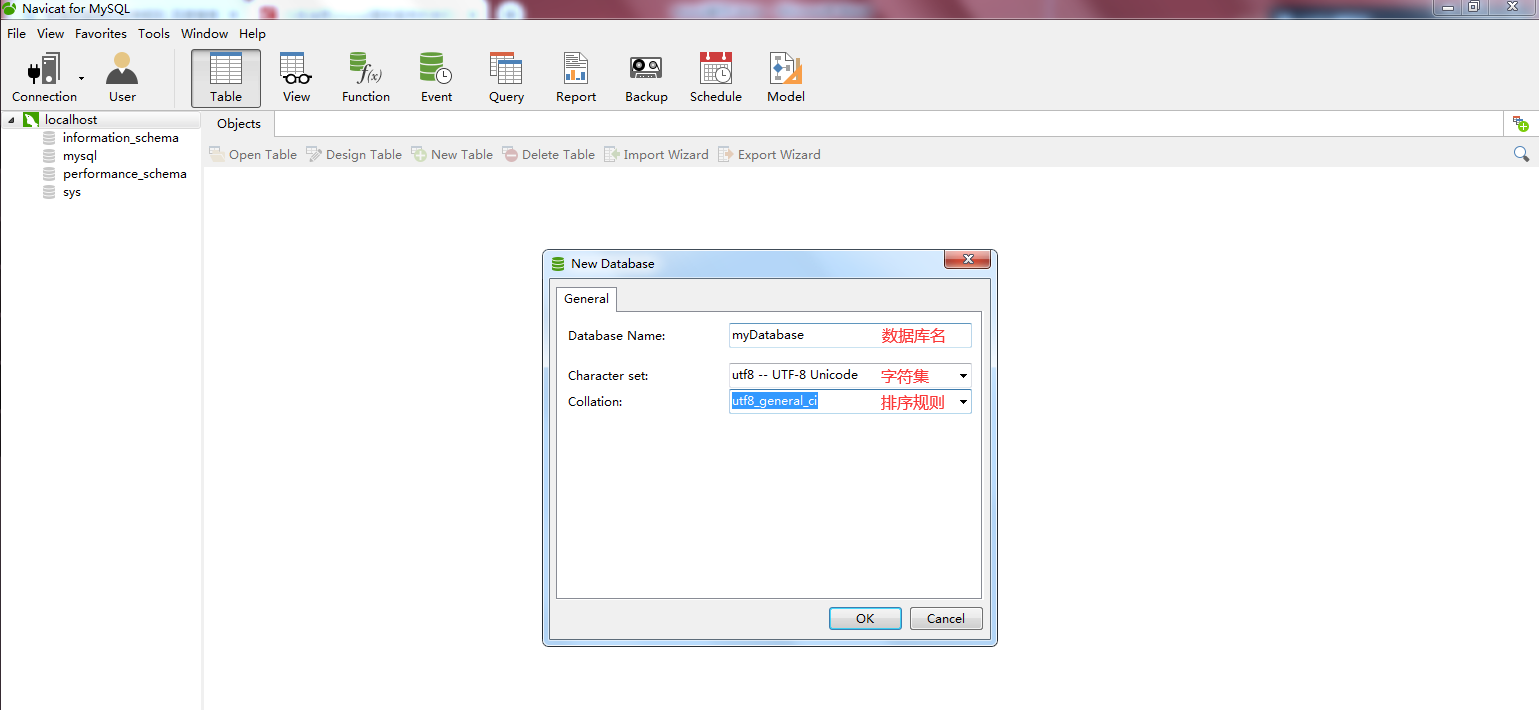


2.创建数据库

选中连接----》右键----》New Database

操作演示如图：

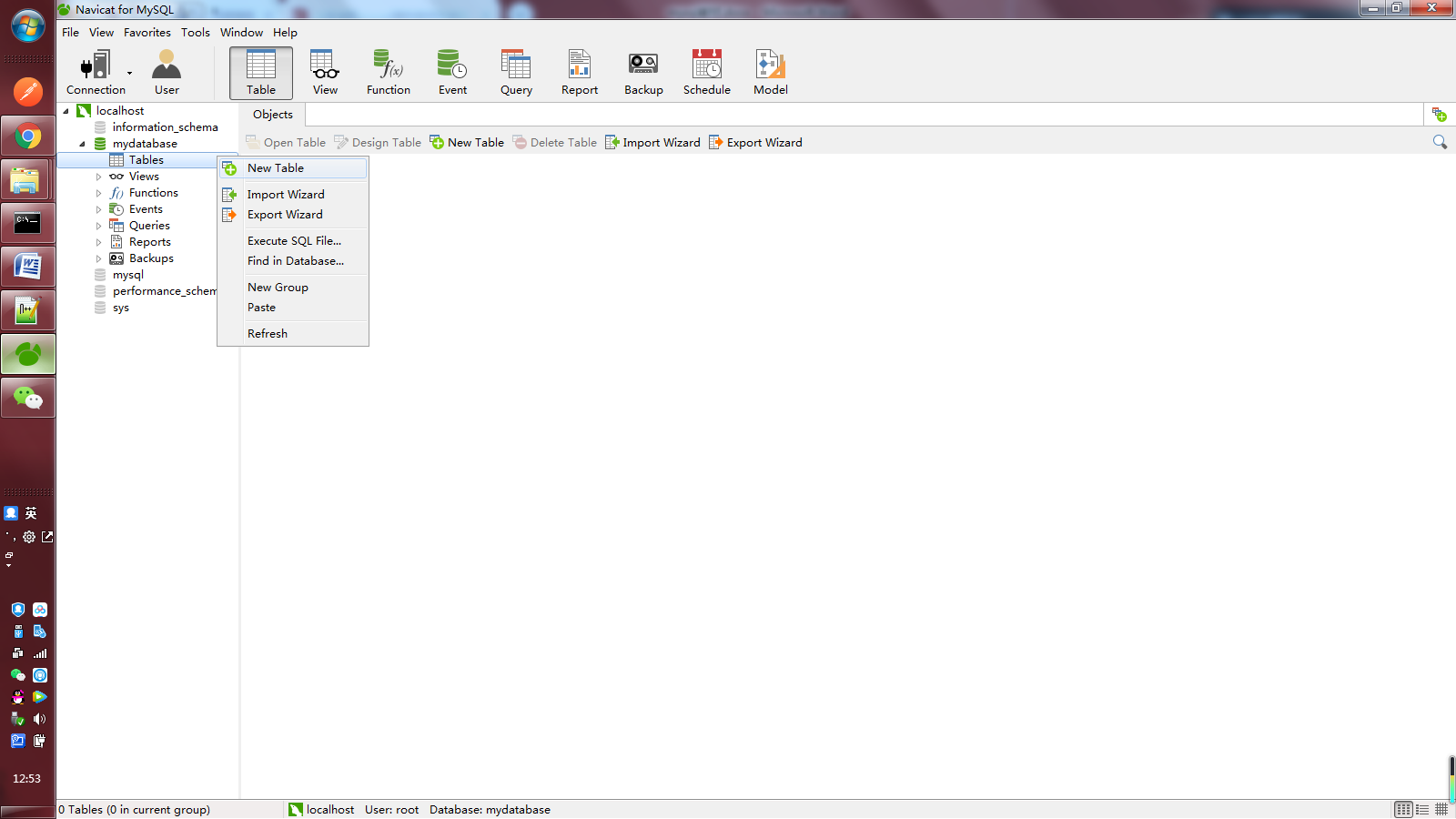




3.新建表

选中tables----》右键----》New Table

操作演示如图：



# 第三章、数据库的基本操作

## 3.1创建数据库

### 3.1.1 基本语法：

CREATE DATABASE [IF NOT EXISTS] 数据库名；

### 3.1.2.参数说明：

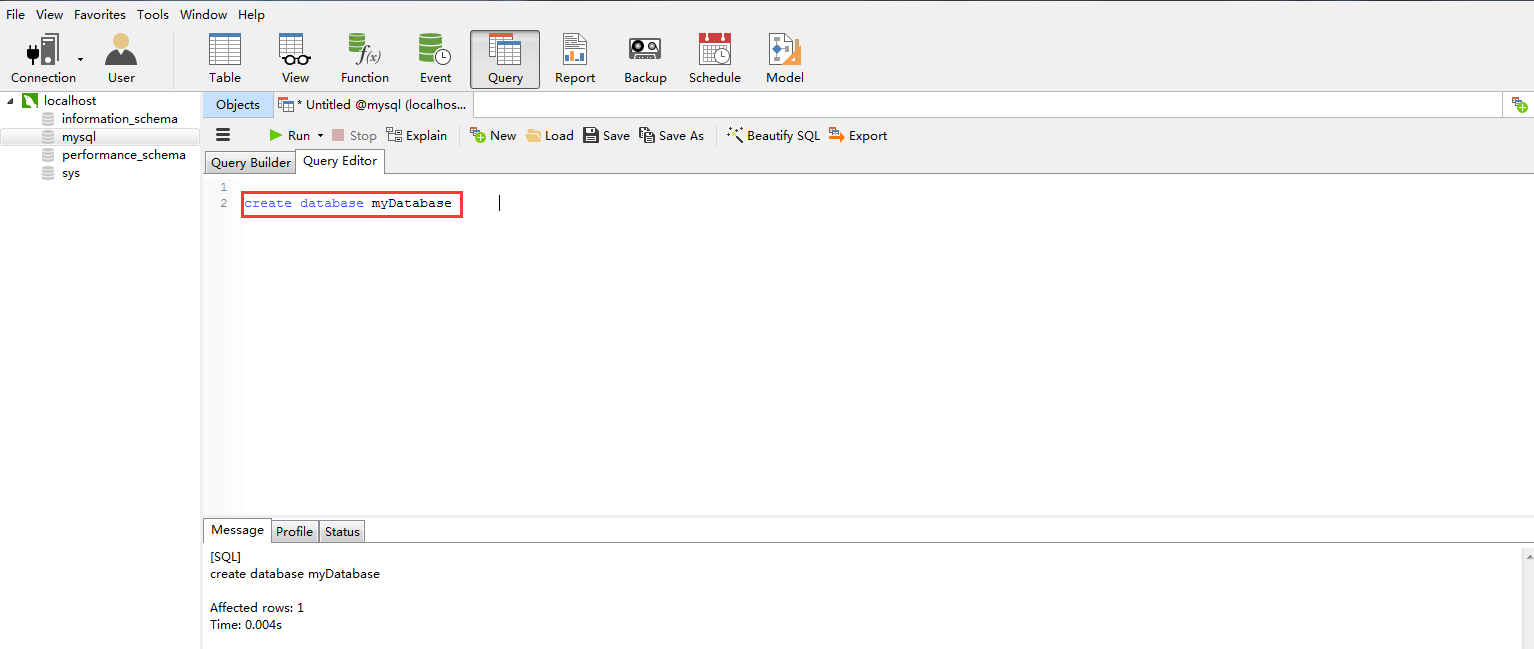
1、[IF NOT EXISTS]：

可选子句，意思是创建的数据库在Mysql的服务器中不存在则创建，否则不创建，这样可以防止重复创建名字相同的数据库

案例1：创建数据库myDatabase

create database myDatabase

操作演示如图：



注意事项：

1.重复运行会报错，因为已经有同名的数据库了，报错信息如下：

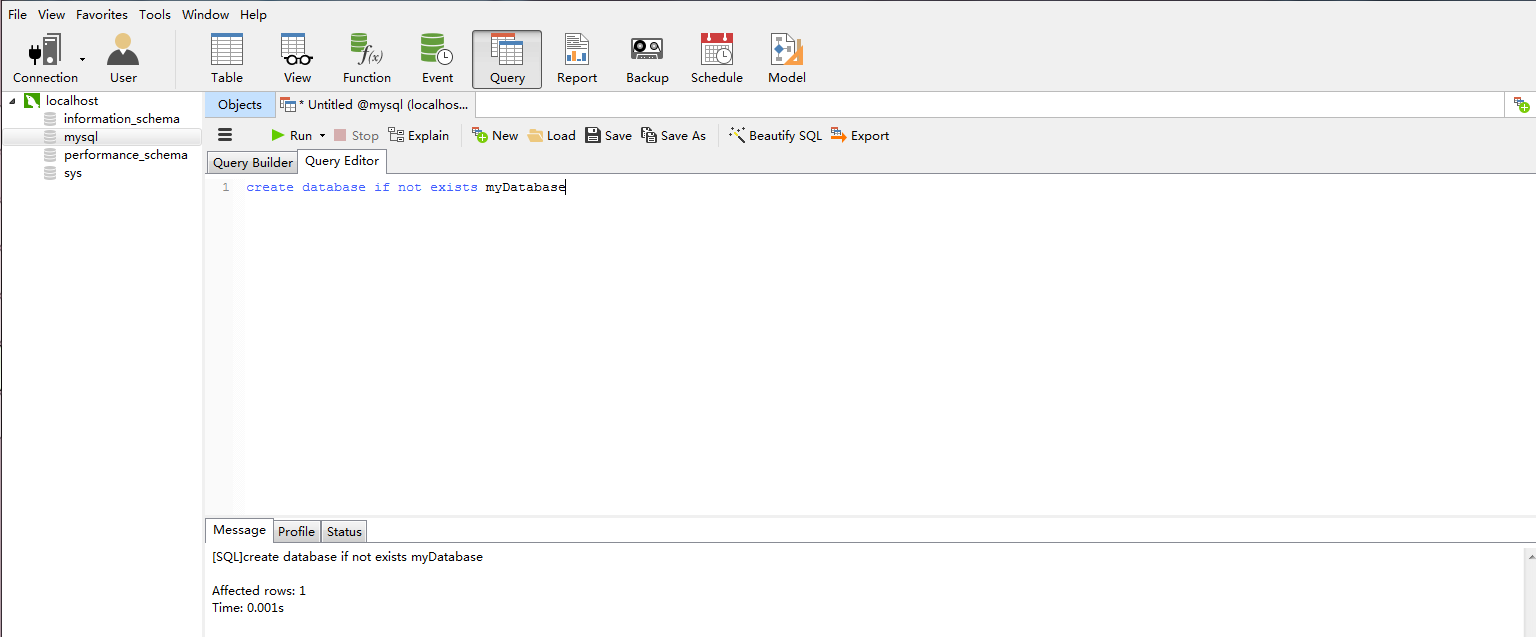
[Err] 1007 - Can't create database ' myDatabase '; database exists

2.数据库名后面不要写分号”’;”，虽然不会报错，但是会把分号认为是数据库名的一部分

案例2：如果不存在数据库myDatabase则创建数据库myDatabase

create database if not exists myDatabase

操作演示如图：



注意事项：

1. 重复运行不会报错，因为如果库已经存在，则不会再去创建。
2. 数据库名后面不要写分号”’;”，虽然不会报错，但是会把分号认为是数据库名的一部分

创建数据库时，数据库的命名规则如下：

1）不能与其他数据库同名，否则将发生错误

[Err] 1007 - Can't create database 'abc'; database exists

2)名称可以由任意字母，阿拉伯数字，下划线和$组成，可以使用上述任意字符开头，但是不能使用单独的数字，否则会造成它与数值相混

3）名称最长为64个字符

4）不能使用MYSQL关键字作为数据库名

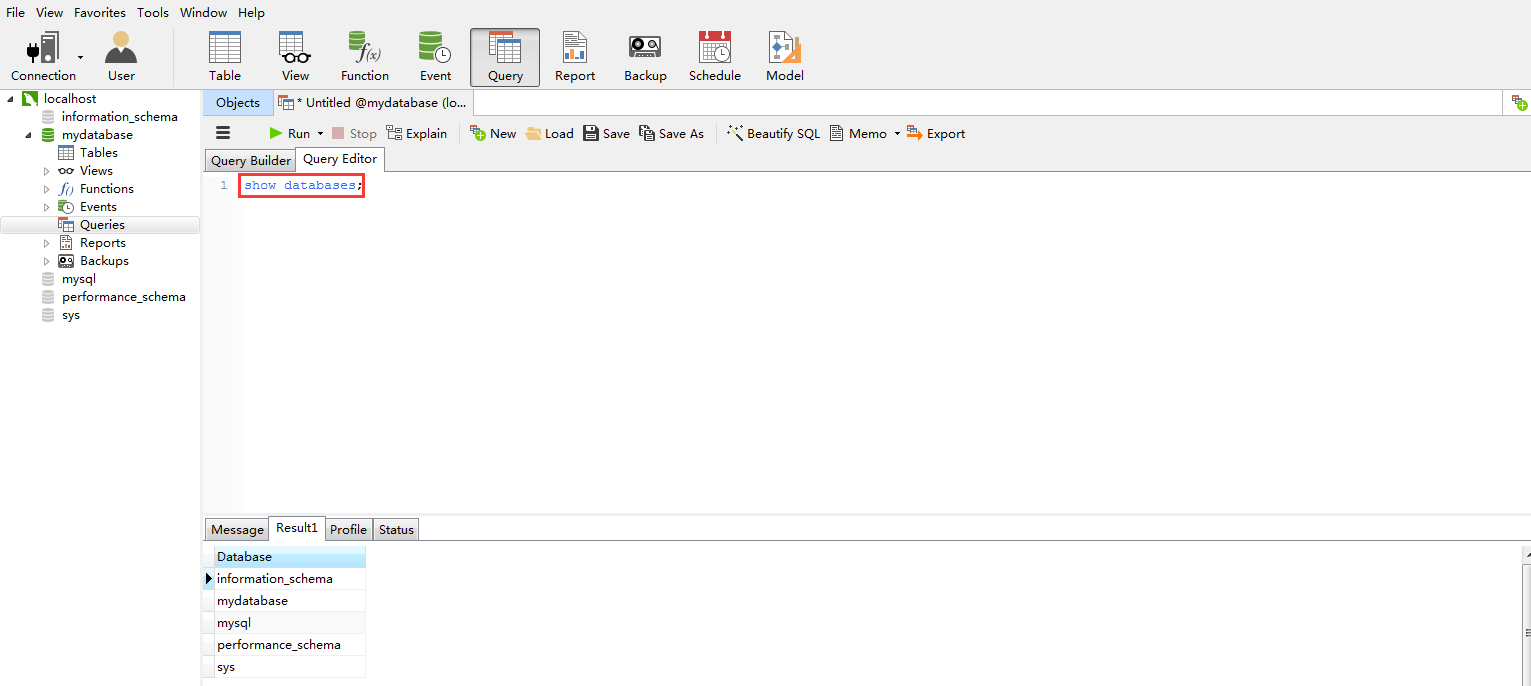
5）默认情况下Windows下对数据库名的大小写不敏感，而在Linux下对对数据库名的大小写敏感，为了能在不同的平台间进行移植，建议采用小写的数据库名

## 3.2查看数据库

### 3.2.1.基础语法：

show databases;

操作演示如图：



注意事项：

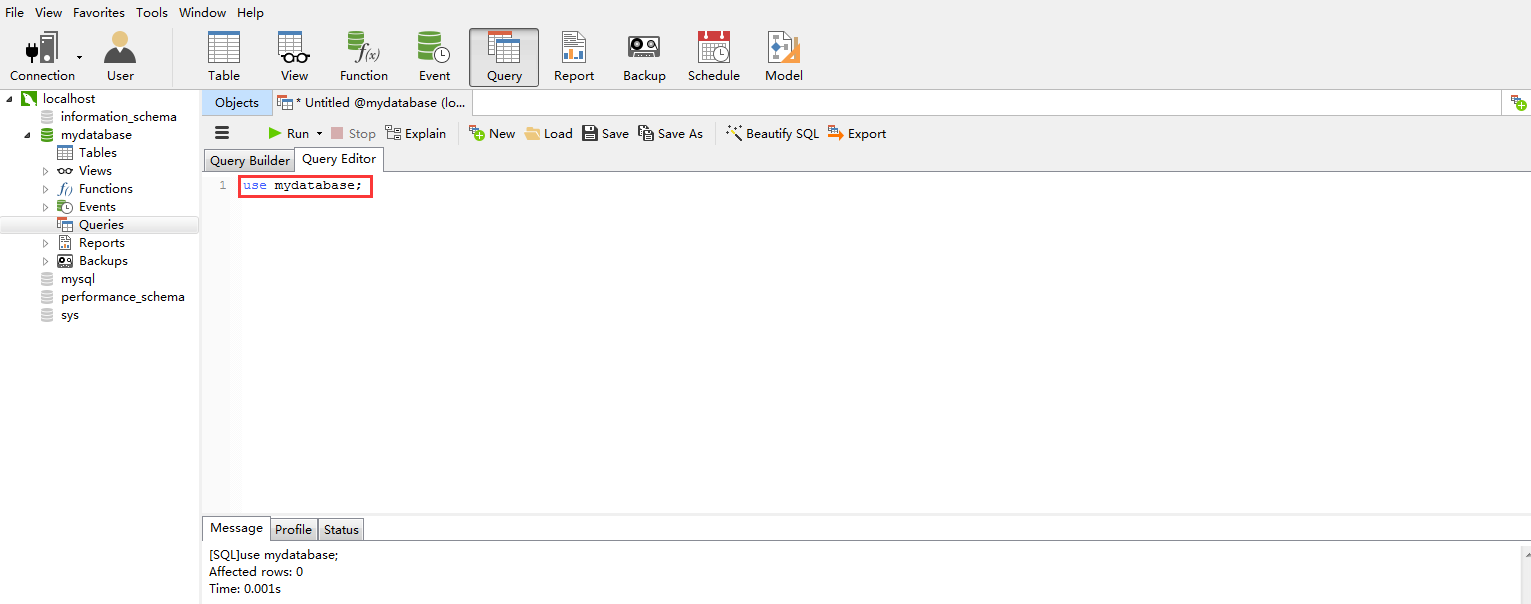
1.databaeses后面的s，容易粗心写成show databaese;然后一直报语法错误：You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DATABASE' at line

## 3.3选择数据库

### 3.3.1.基础语法：

use 数据库名；

操作演示如图：



## 3.4删除数据库

### 3.4.1基础语法

drop database [if exists] 数据库名；

### 3.4.2 参数说明

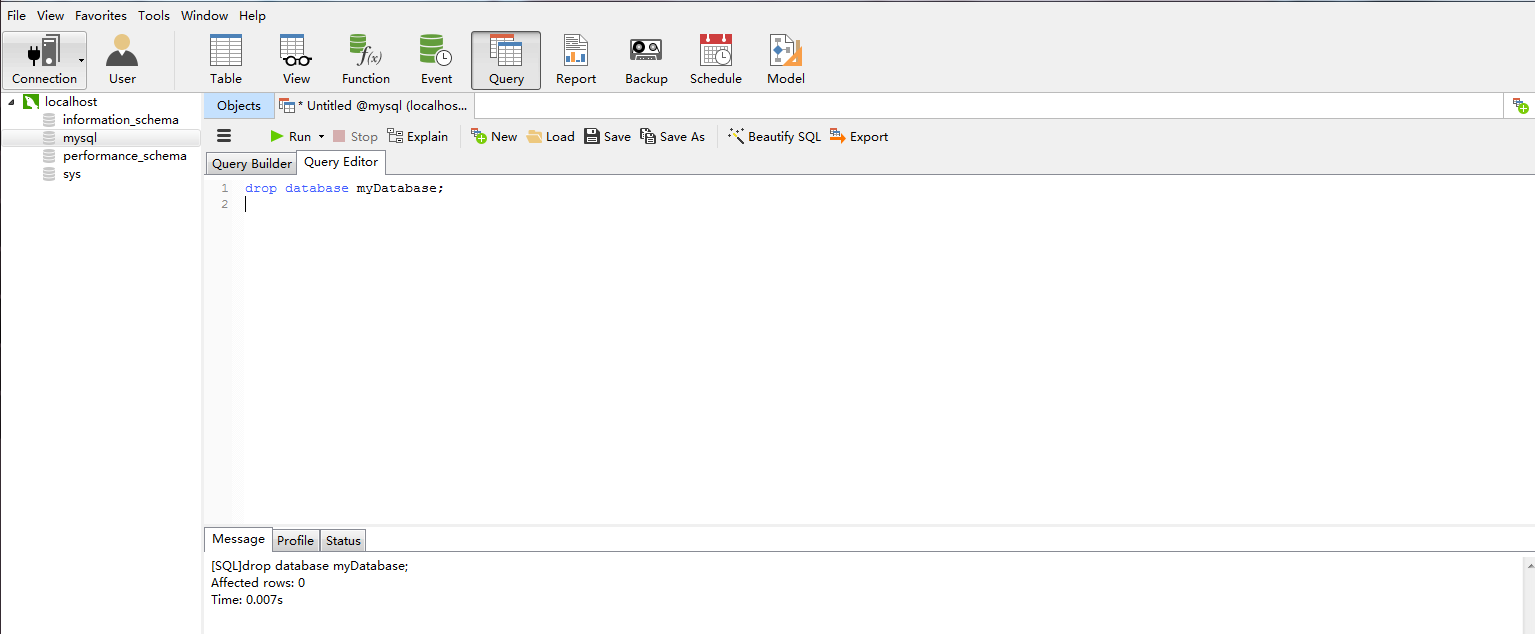
1.[ if exist]：

可选参数，释义：如果存在

案例1：删除myDatabase数据库

drop database myDatabase;

操作演示如图：



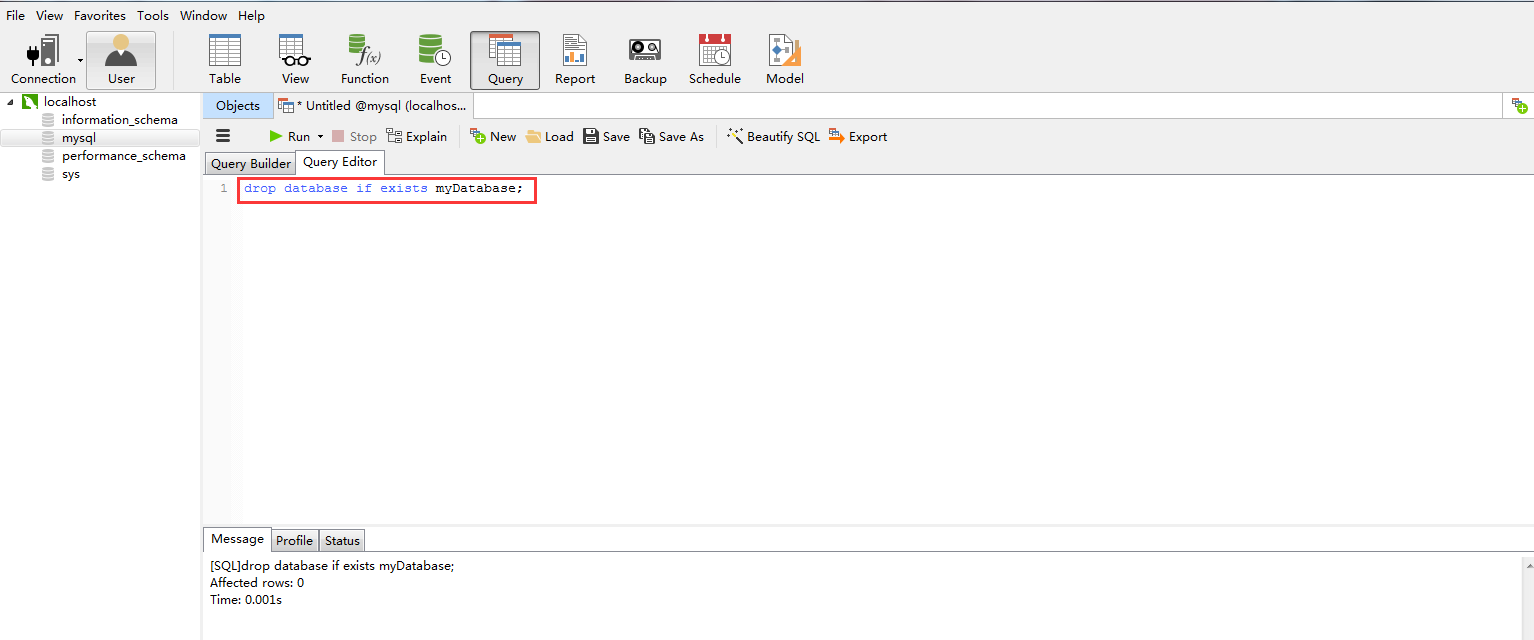
注意事项：

1.重复运行报错，原因同创建数据库

案例2：如果存在数据库myDatabase，删除myDatabase数据库

drop database if exists myDatabase;

操作演示如图：



## 3.5 数据库存储引擎

### 3.5.1 Mysql存储引擎

### 3.5.2 InnoDb存储引擎

### 3.5.3 MyISM存储引擎

### 3.5.4 MEMORY存储引擎

# 第四章、数据表的基本操作

## 4.1数据类型

### 4.1.1数字

### 4.1.2.日期和时间

### 4.1.3.字符串

## 4.2创建数据表

### 4.2.1.基础语法

create table 数据库表名 （

字段名1 数据类型 [列约束条件] [默认值]，

字段名2 数据类型 [列约束条件] [默认值]，

……

[表级约束条件]

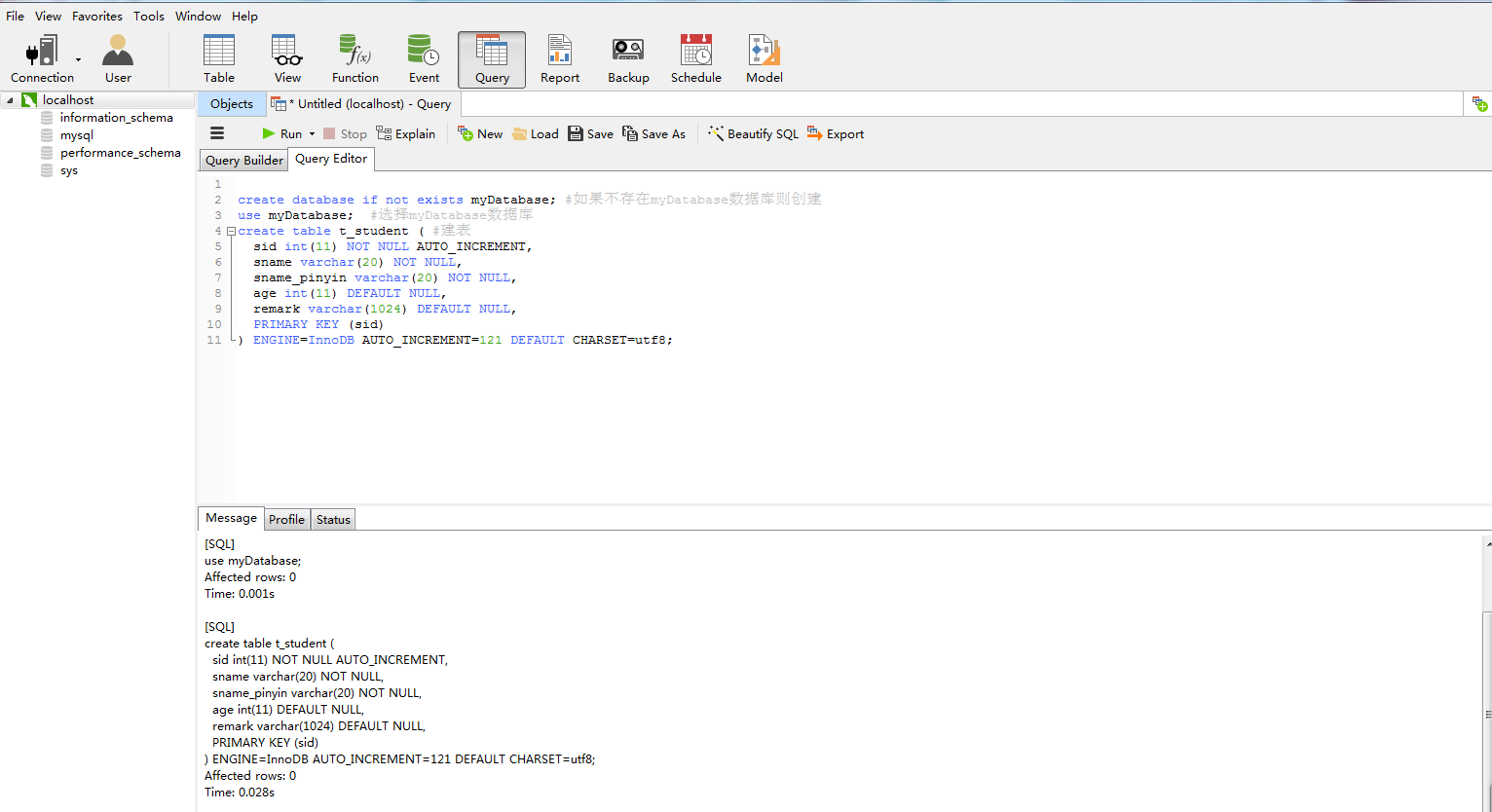
）;

案例1: 在myDatabase库中创建学生表t\_students

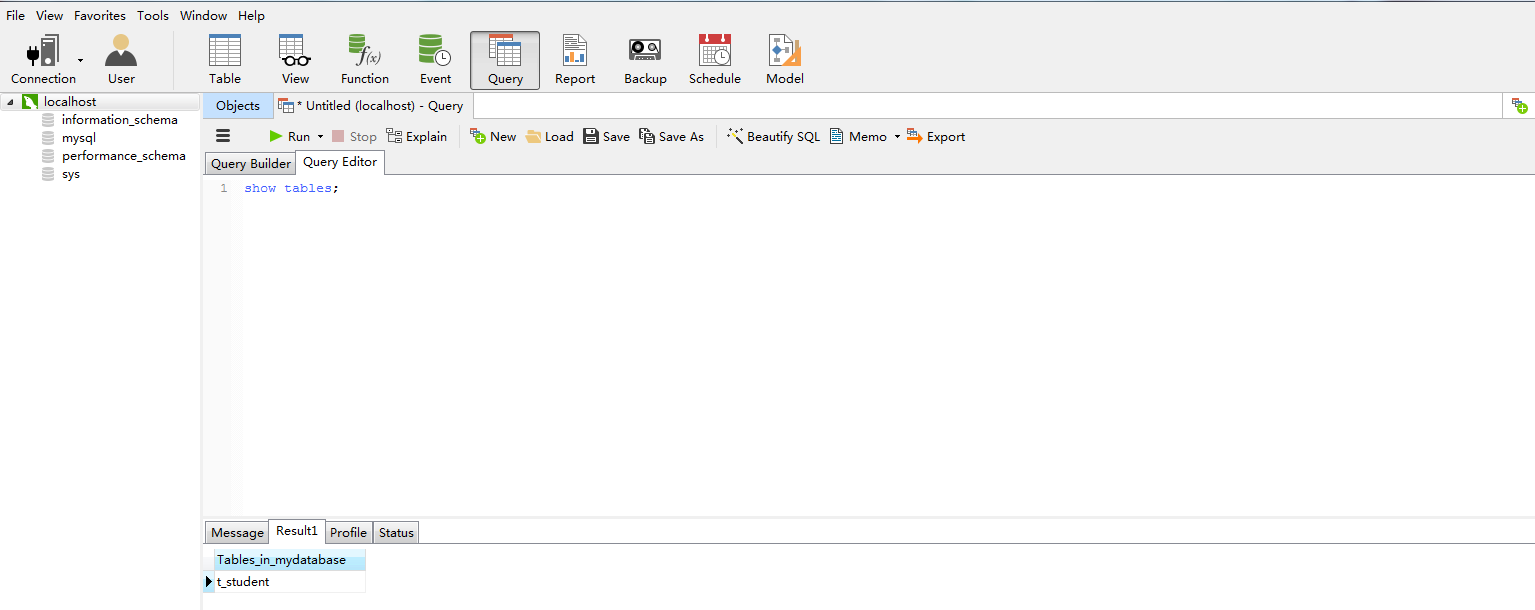
表结构如图：

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
|  |  | 学生id |
|  |  | 学生名字 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

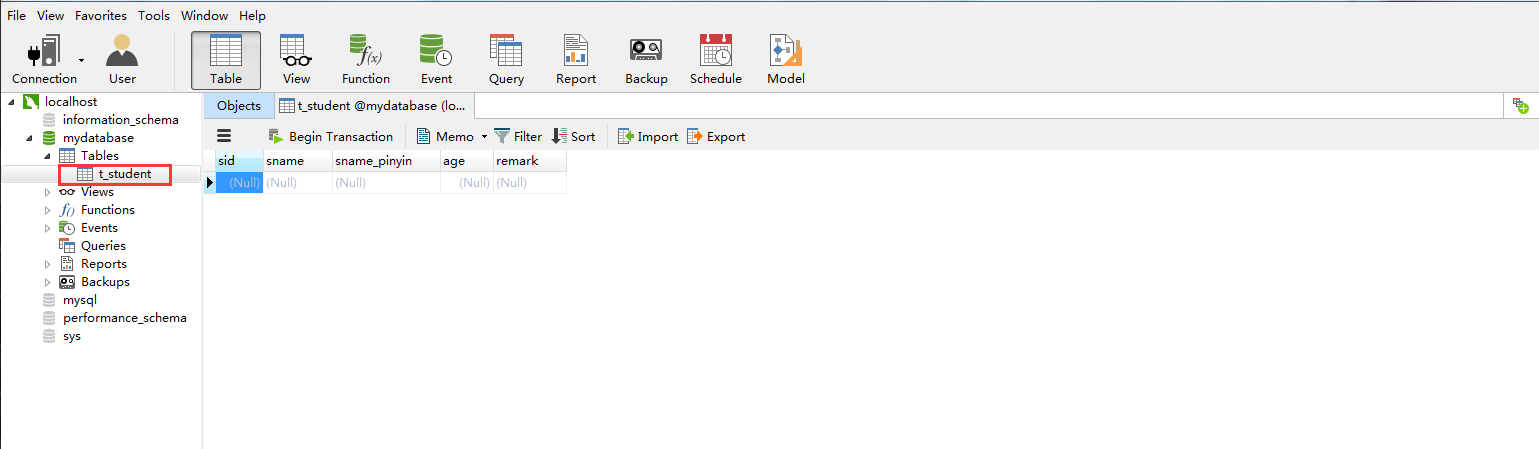
操作演示如图：



执行语句成功后，就创建了一个名为t\_students的数据表，使用show table；语句查看数据表是否创建成功



找到数据库，点开也可以看到t\_students表：



## 4.3查看表结构

### 4.3.1查看基本结构语句

### 4.3.2查看详细结构语句

## 4.4修改数据表

### 4.4.1修改表名

1.基础语法：

alter table原表名rename [to] 新表名;

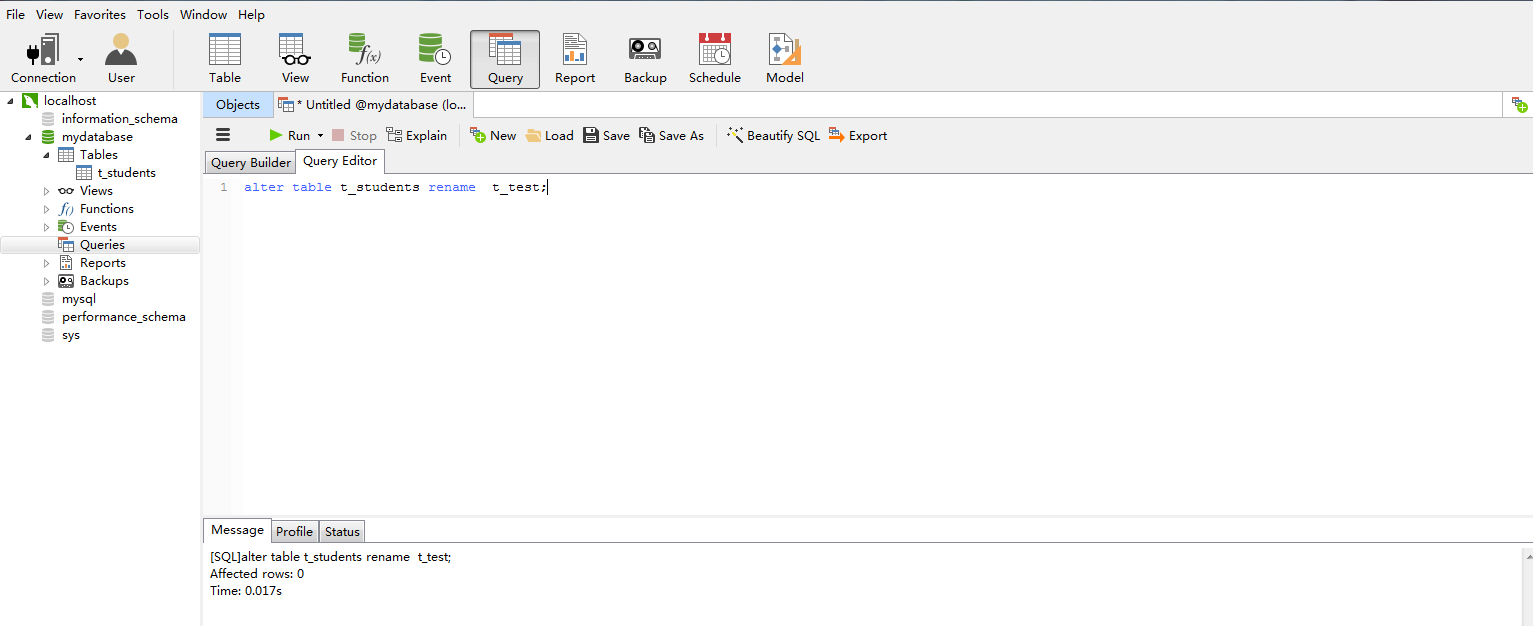
2.参数说明：

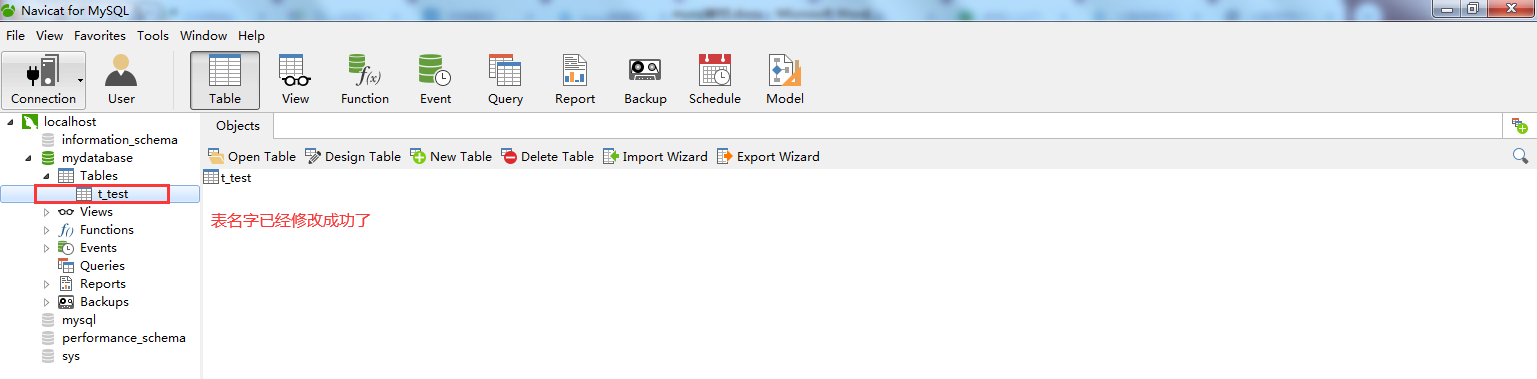
[TO] :可选参数，不管是否使用都不影响运行结果

案例1：把t\_students表重命名为t\_test

alter table t\_student rename t\_students;

操作演示如图：





### 4.4.2修改字段的数据类型

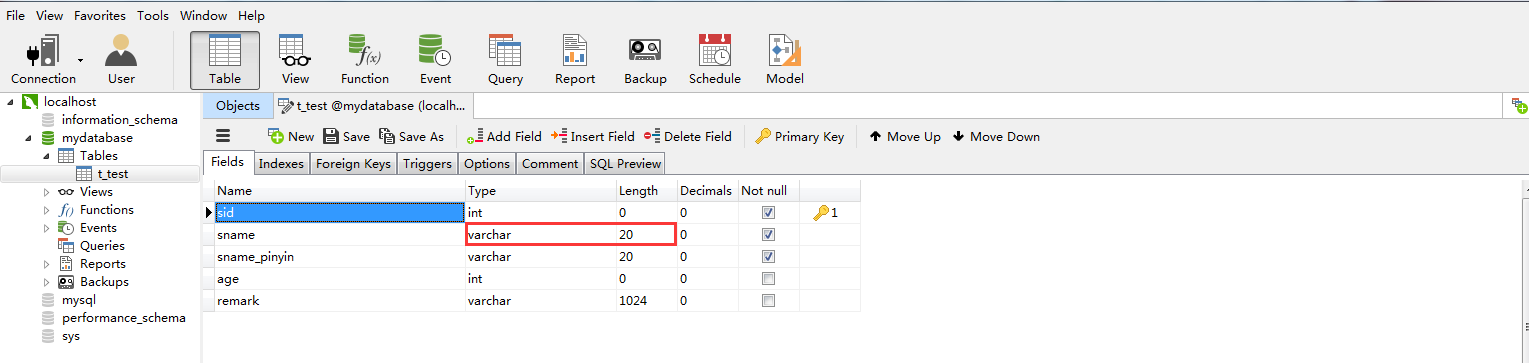
1.基础语法：

alter table 表名 modify | change 字段名 数据类型;

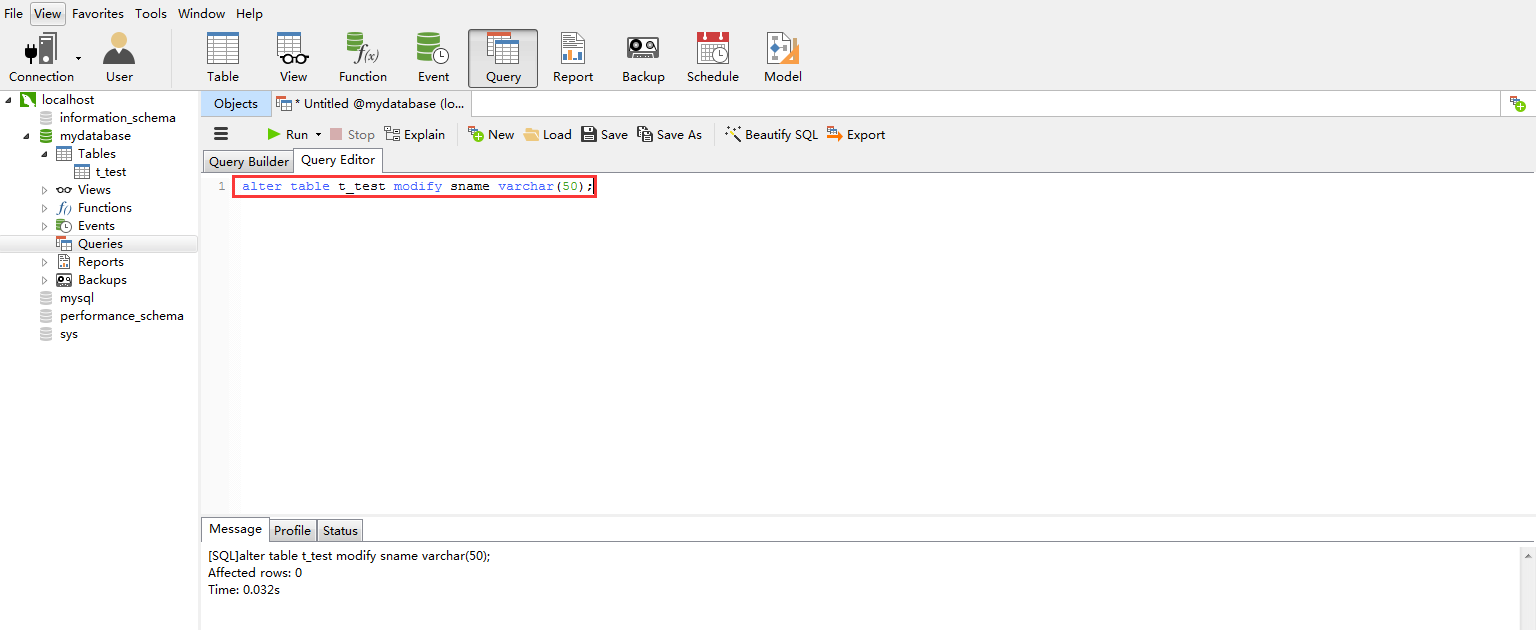
案例1：修改学生姓名的数据库类型varchar(20)为varchar(50)

alter table t\_test modify sname varchar(50);

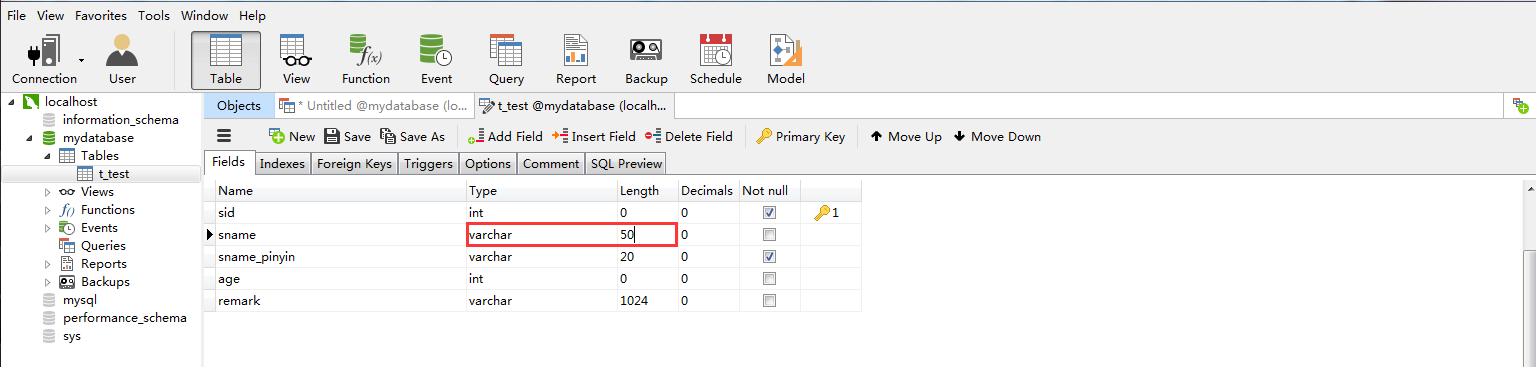
修改前：选中表----》右键----》Design Table可查看到表结构



操作演示如图：



修改后：



注意事项：change和modifiy的区别，change可以修改字段名称，modify不行

### 4.4.3修改字段名

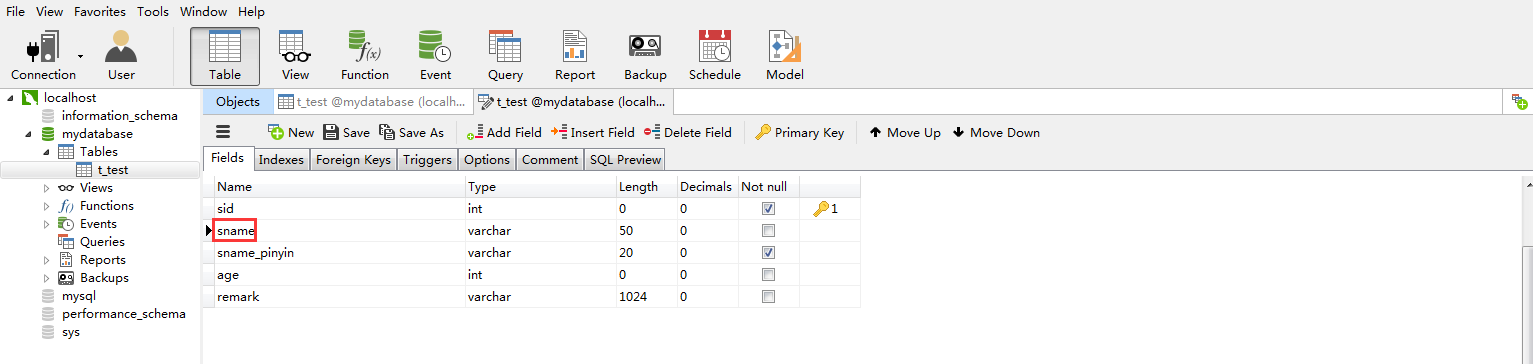
1.基础语法

alter table 表名 change原字段名 新字段名 新数据类型;

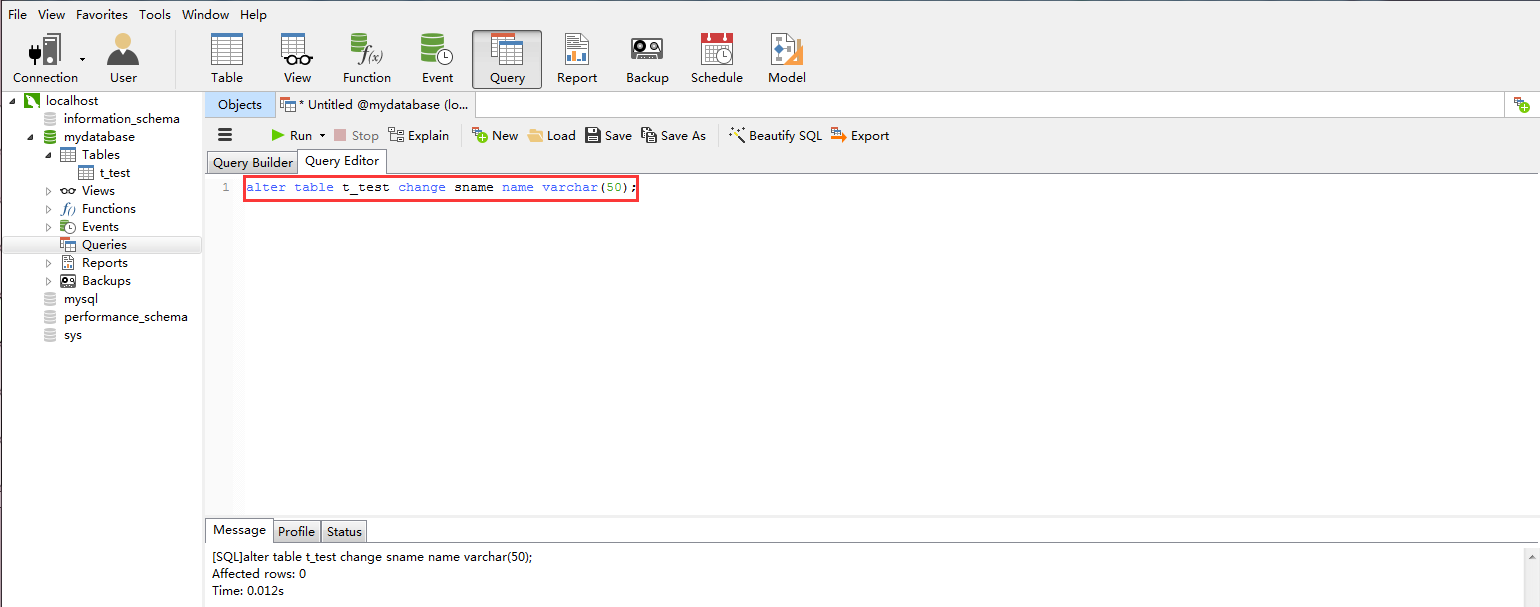
案例1：修改t\_test表中的sname的字段名为name

alter table t\_test change sname name varchar(50);

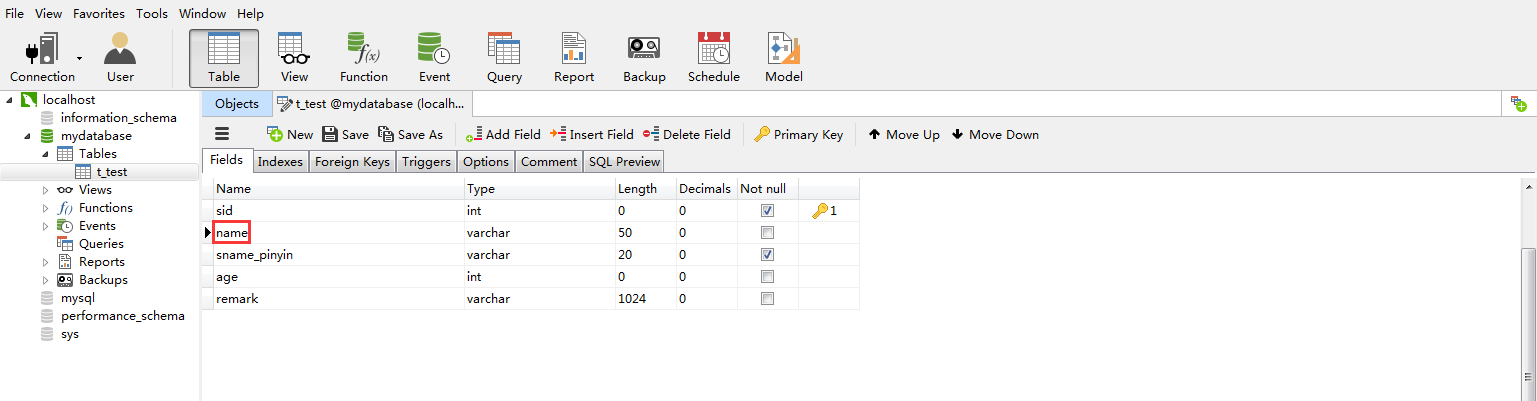
修改前：



操作演示如图：



修改后：



### 4.4.4添加字段

1.基础语法

alter table表名 add 新字段名 数据类型

[约束条件] [FIREST | AFTER 已存在的字段名]；

2.参数说明

1)[约束条件]

2)first和after已存在的字段名

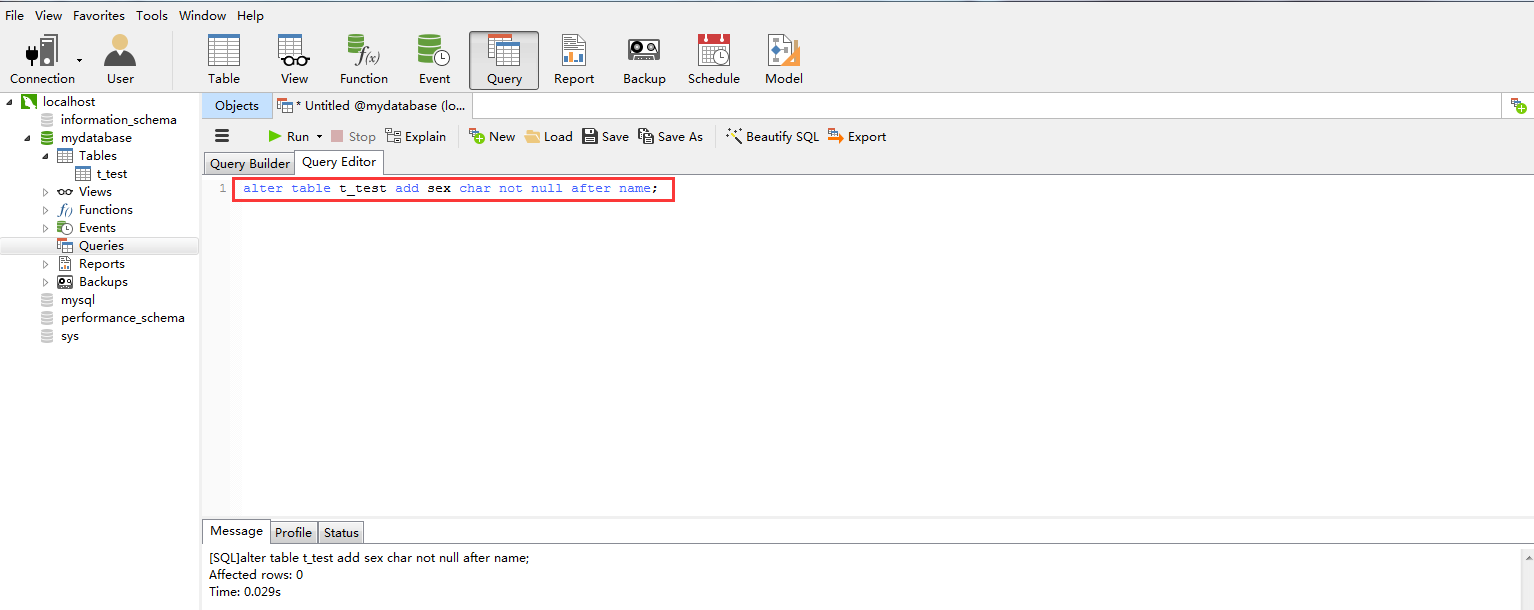
都表示添加的字段排序的位置，first表示放到最前面，after 已存在的字段名表示放到某字段后

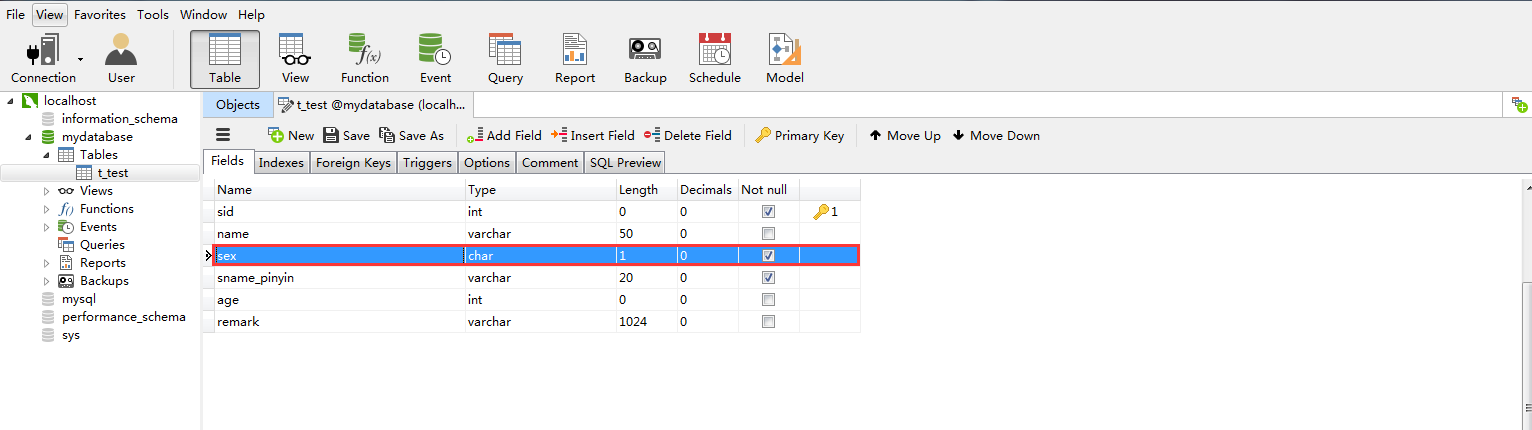
注意事项：只能在某个已有字段的后面添加新字段，不能在它的前面添加新字段。

案例1：添加一个char类型的sex字段并且放到name字段后，默认不为空

alter table t\_test add sex char not null after name;

操作演示如图：

运行后效果如图：



### 4.4.5修改字段的排列位置

1. 基础语法

alter table 表名 modify 字段1 数据类型 first | after 字段2;

1. 参数说明

字段1：要修改排列顺序的字段

first | after 字段2 :

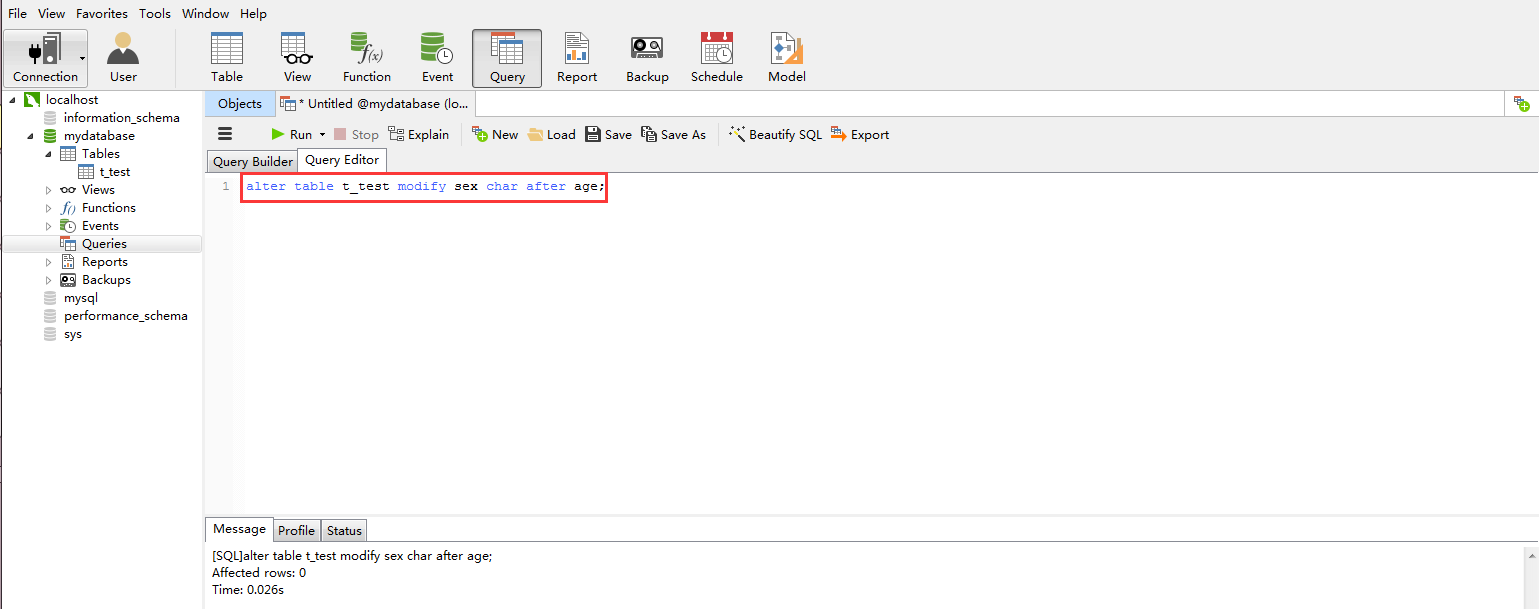
first 更改字段1的顺序为表的第一个字段

after 字段2 更改字段1的顺序为字段2的后面

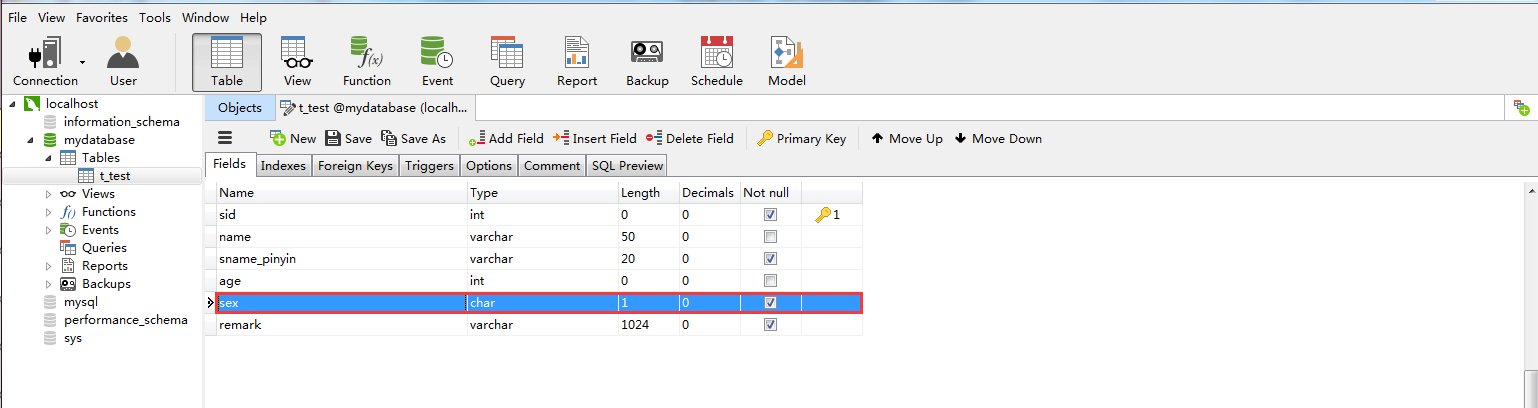
案例1：更改sex字段的顺序为age字段后面

alter table t\_test modify sex char after age;

操作演示如图：



运行后效果如图：



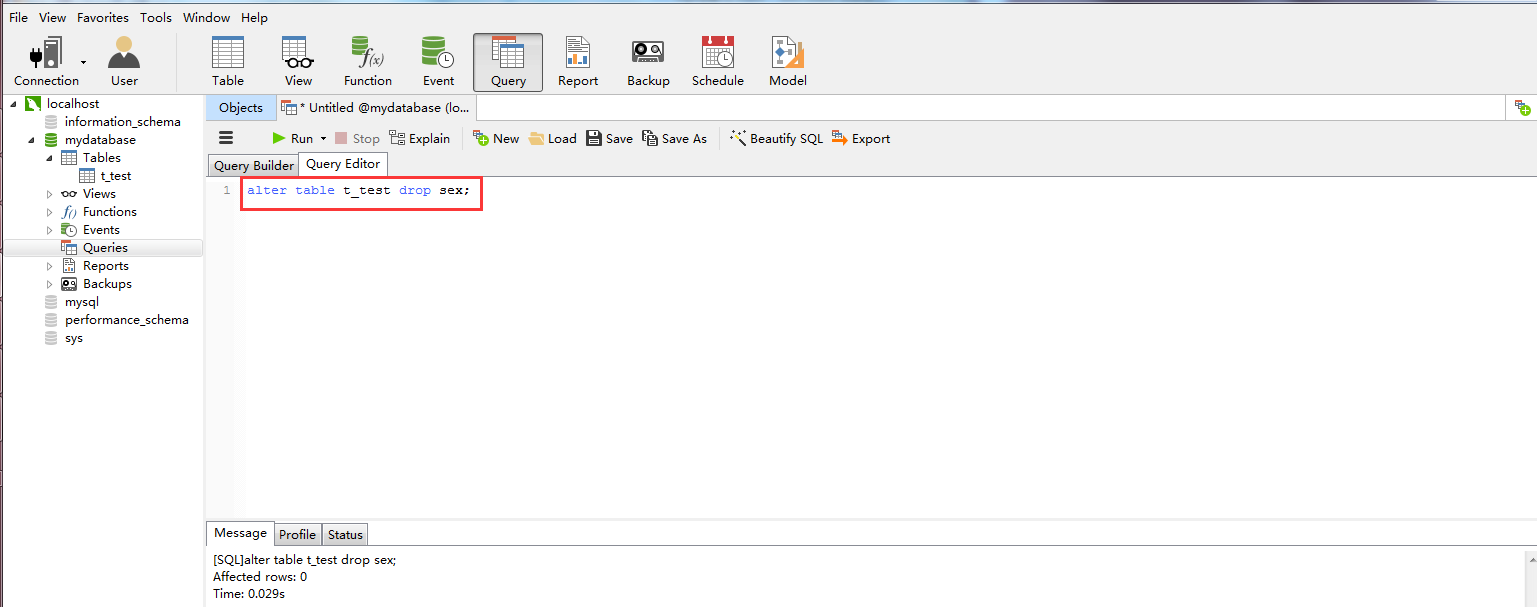
### 4.4.6删除字段

1.基础语法：

alter table 表名 drop 字段名;

案例1：删除t\_test表中的sex字段

alter table t\_test drop sex;



## 4.5数据完整性约束

约束是一种限制，它通过对表的行或列的数据做出限制，来确保表的数据的完整性、唯一性。

MYSQL中，常用的几种约束：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 约束类型 | 主键 | 外键 | 唯一 | 非空 | 自增 | 默认 |
| 关键字 | primary key | foreign key | unique | not null | auto\_increment | default |

### 4.5.1主键约束

主键约束相当于唯一约束 + 非空约束的组合，主键约束列不允许重复，也不允许出现空值。

当创建主键的约束时，系统默认会在所在的列和列组合上建立对应的唯一索引。

1. 单字段主键

1)在定义字段的同时指定主键

基础语法：

字段名 数据类型 PRIMARY KEY [默认值]；

案例1：创建一个账号表，在定义字段userid的同时指定主键

create table t\_account1 (

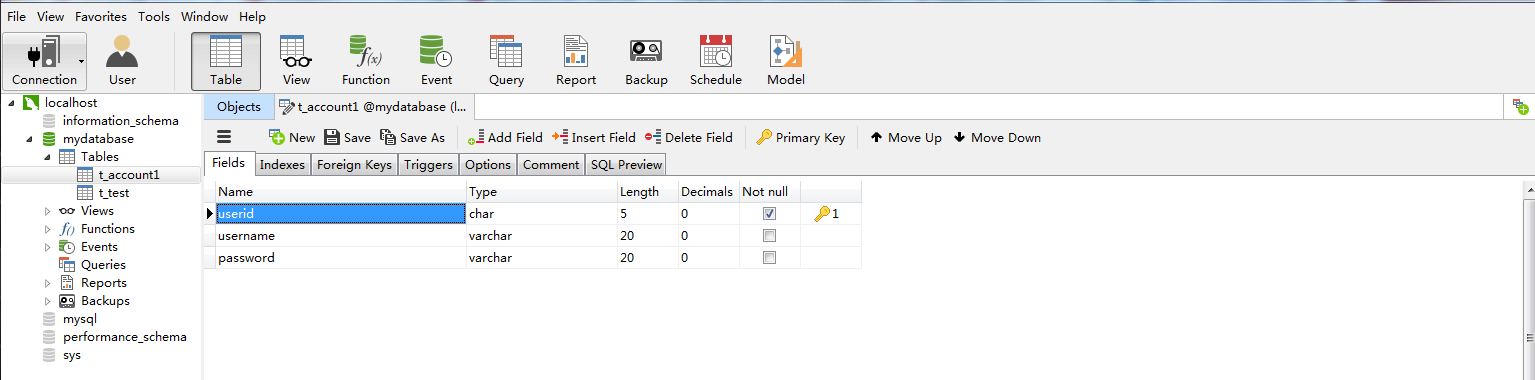
userid char(5) primary key,

username varchar(20),

password varchar(20)

);

运行效果如图：



2)在定义字段的后指定主键

基础语法：

[constraint 约束名] primary key (字段名)

案例：1创建一个账号表，在定义所有的字段后指定主键

create table t\_account2 (

userid char(5),

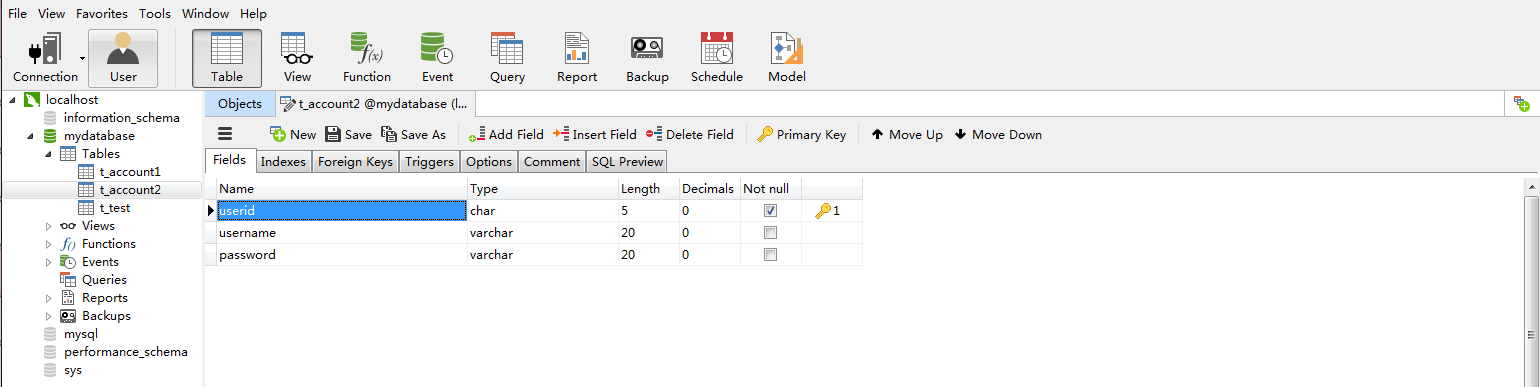
username varchar(20),

password varchar(20),

primary key(userid)

);

运行效果如图：



1. 多字段联合主键

基础语法：

PRIMARY KEY (字段1，字段2，…,字段n);

案列1：假设账户表中没有userid，则可以用用户名和密码联合起来作为主键

create table t\_account3 (

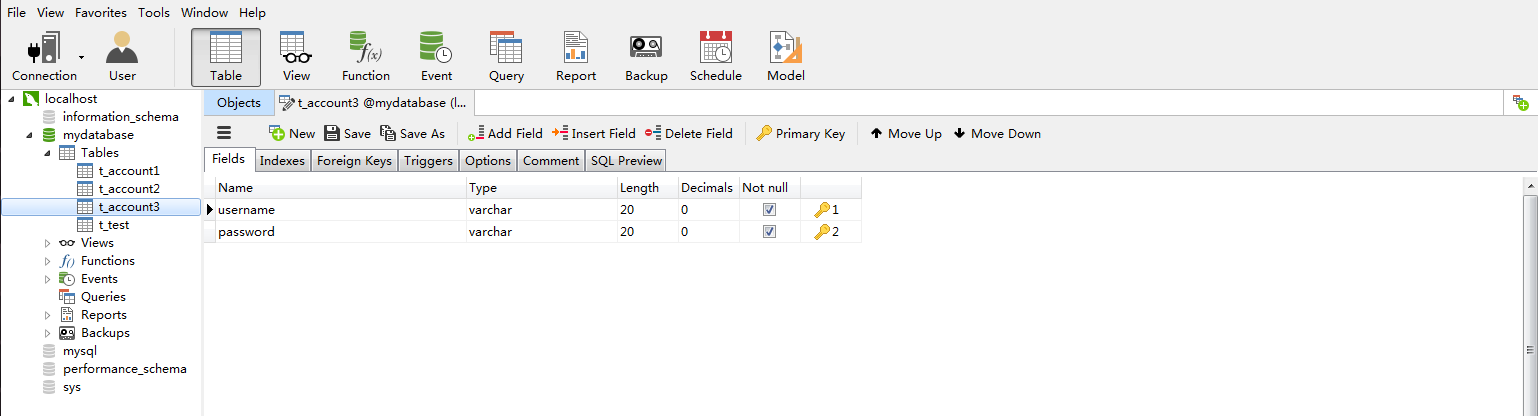
username varchar(20),

password varchar(20),

primary key(username,password)

);

运行效果如图：



### 4.5.2唯一性约束

1）在定义字段的同时指定唯一性约束

基础语法：

字段名 数据类型 unique;

案例1：创建账号表，在定义列的同时指定书本的名字唯一

create table t\_account4 (

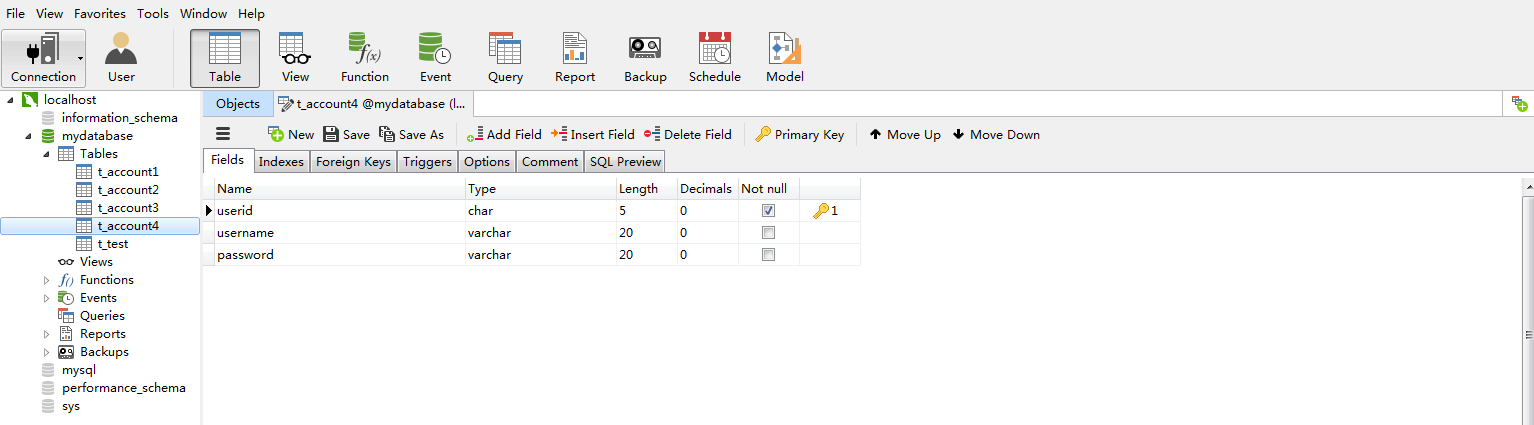
userid char(5) primary key,

username varchar(20) unique,

password varchar(20)

);

运行效果如图：



2）在定义所有字段后，指定唯一性约束

基础语法：

[constraint 约束名] unique (字段名);

案例1：创建账号表，在定义所有字段后，指定唯一性约束

create table t\_account5 (

userid char(5) primary key,

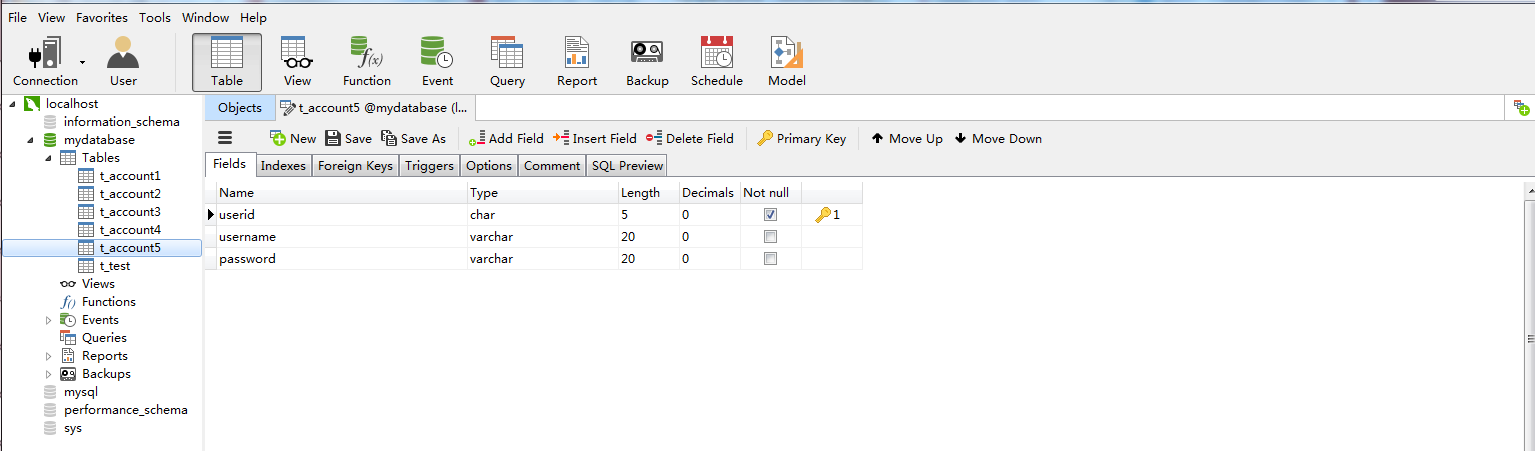
username varchar(20),

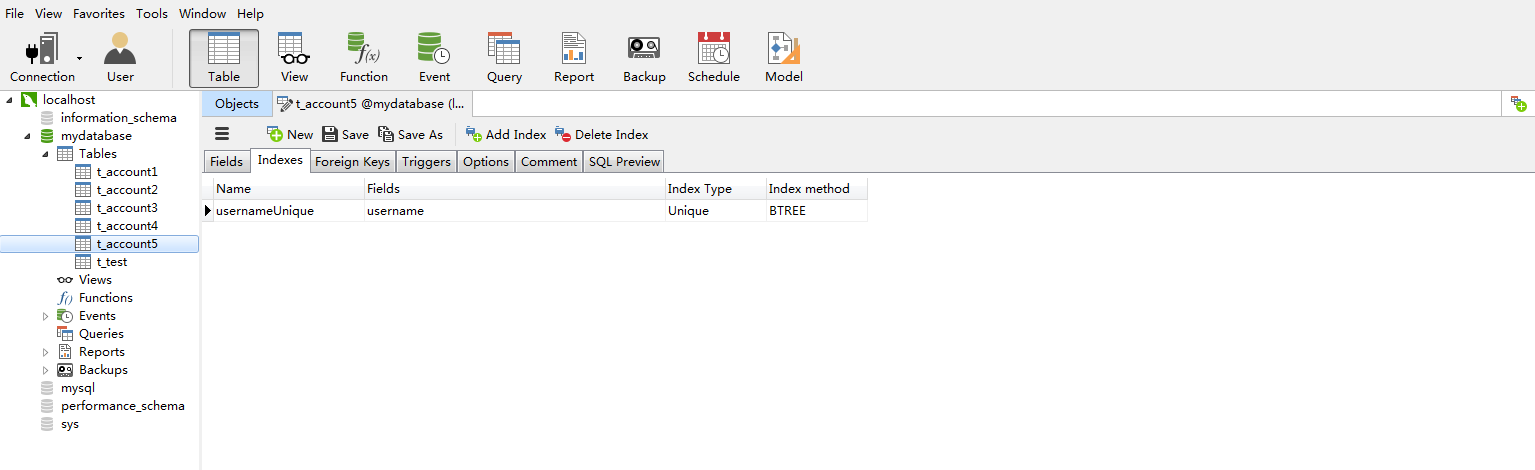
password varchar(20),

constraint usernameUnique unique (username)

);

运行效果如图：





### 4.5.3非空约束

基础语法：

字段名 数据类型 not null；

案例1：创建账户表，指定用户名不为空

create table t\_account6 (

userid char(5) primary key,

username varchar(20) not null,

password varchar(20)

);

### 4.5.4默认约束

基础语法：

字段名 数据类型 not null；

案例1：创建账户表，指定密码默认为1234

create table t\_account7 (

userid char(5) primary key,

username varchar(20),

password varchar(20) default 12345

);

### 4.5.5字段值自动增加

基础语法：

字段名 数据类型aoto\_increment;

案例1：创建用户表，使userid列的值在添加记录的时候自动增加

create table t\_account8 (

userid int(11) primary key auto\_increment,

username varchar(20),

password varchar(20)

);

### 4.5.6外键约束

基础语法：

alter table 表名 add constraint 外键约束名 foreign key(外键字段) references 关联表名(关联字段)

案例1： 订单表与订单详细表的主外键关系

alter table t\_order\_line add constraint FK\_order\_line\_id

foreign key(lin\_ord\_id) references t\_orders(ord\_id);

表一：订单表t\_orders

create table t\_orders(

ord\_id int primary key auto\_increment, -- 订单编号，唯一 customer\_id int not null,

customer\_name varchar(100) not null,

addr varchar(255) not null,

ord\_date date not null,

status char(1) not null

);

表二：订单详细表order\_line 表结构

create table sale\_order\_line (

lin\_id int primary key auto\_increment, -- 订单条目编号，唯一

lin\_order\_id int not null, -- sale\_order表的外键，对应的订单编号

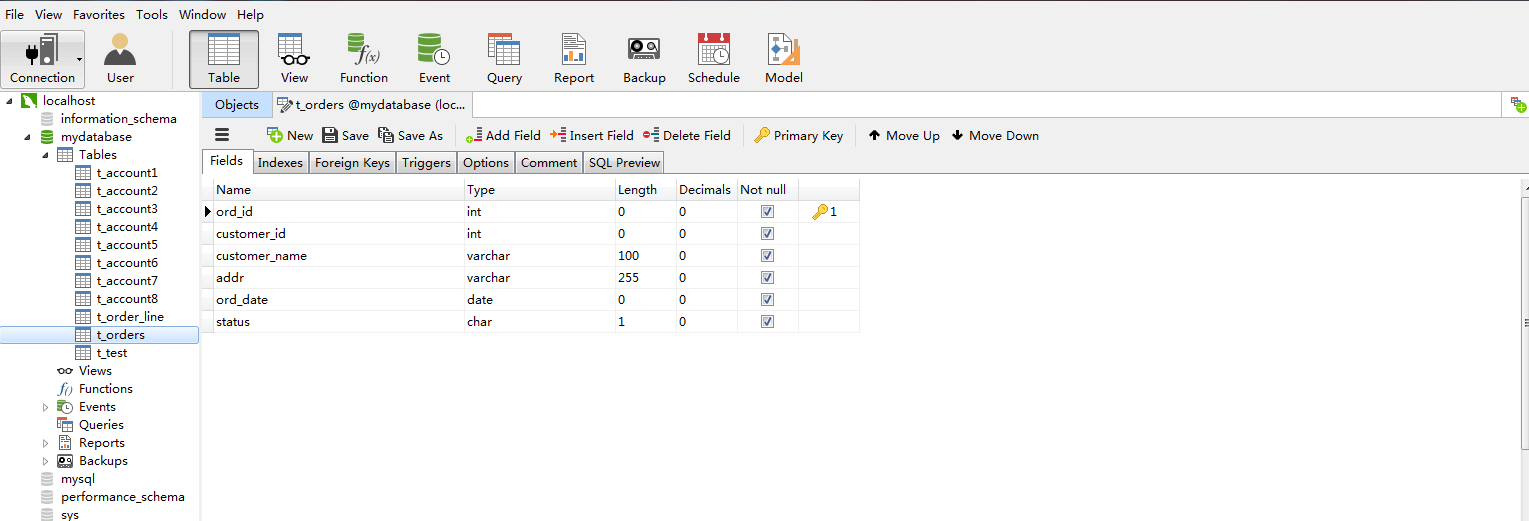
product\_name varchar(50),

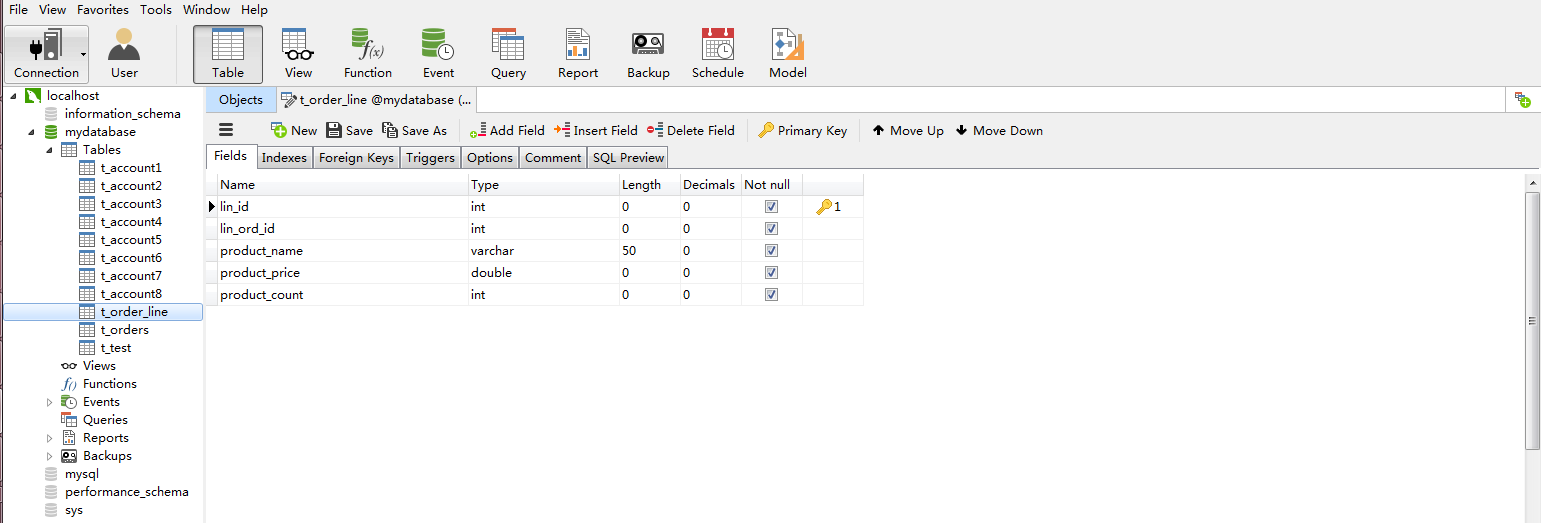
product\_price double not null,

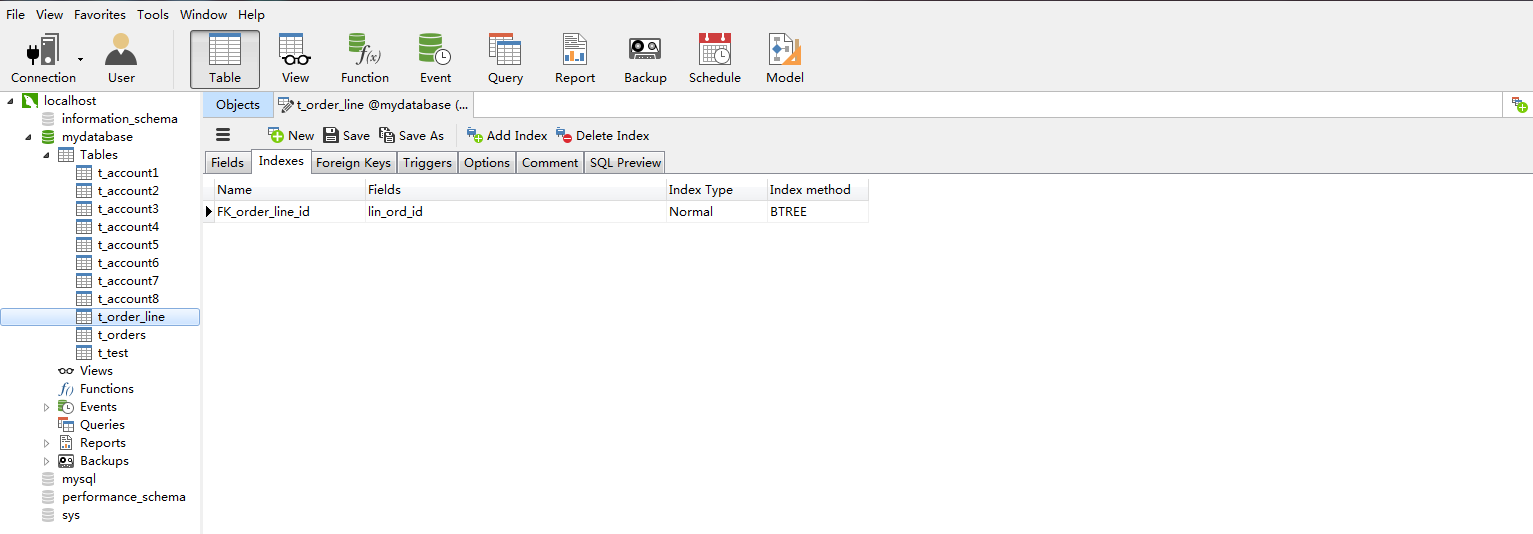
product\_count int not null

);

运行效果如图：







# 第五章、表数据的增删改操作

## 5.1 插入数据

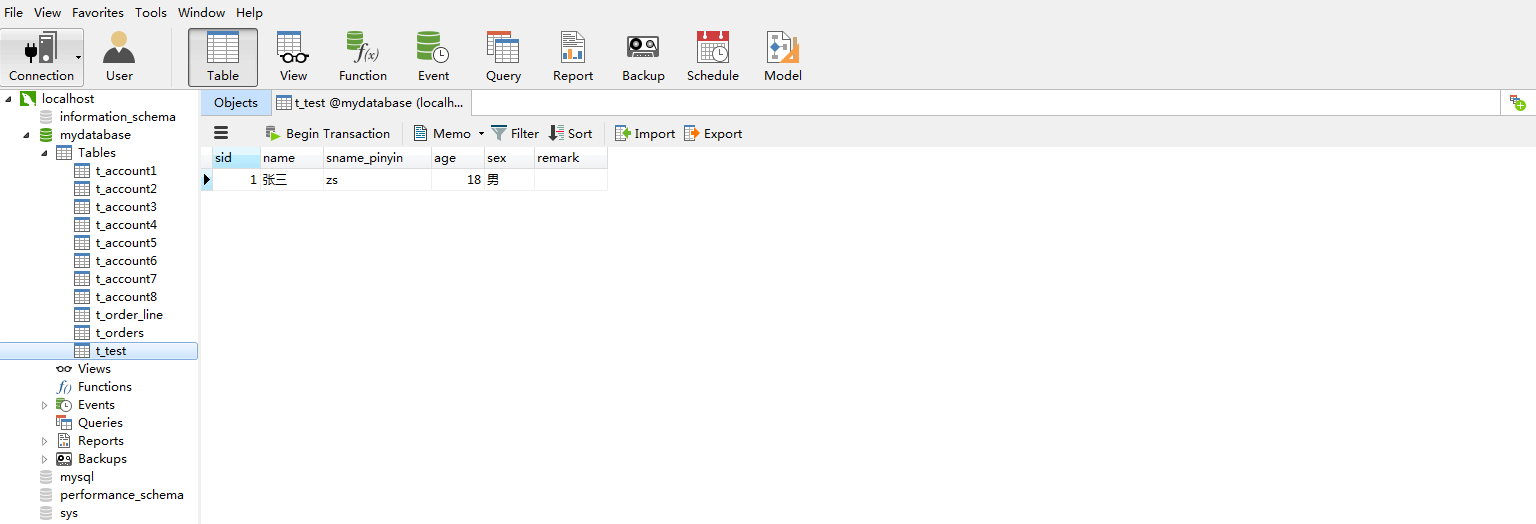
基础语法：

参数说明：

### 5.1.1插入完整数据

insert into t\_test values('1', '张三', 'zs', '18', '男', '');

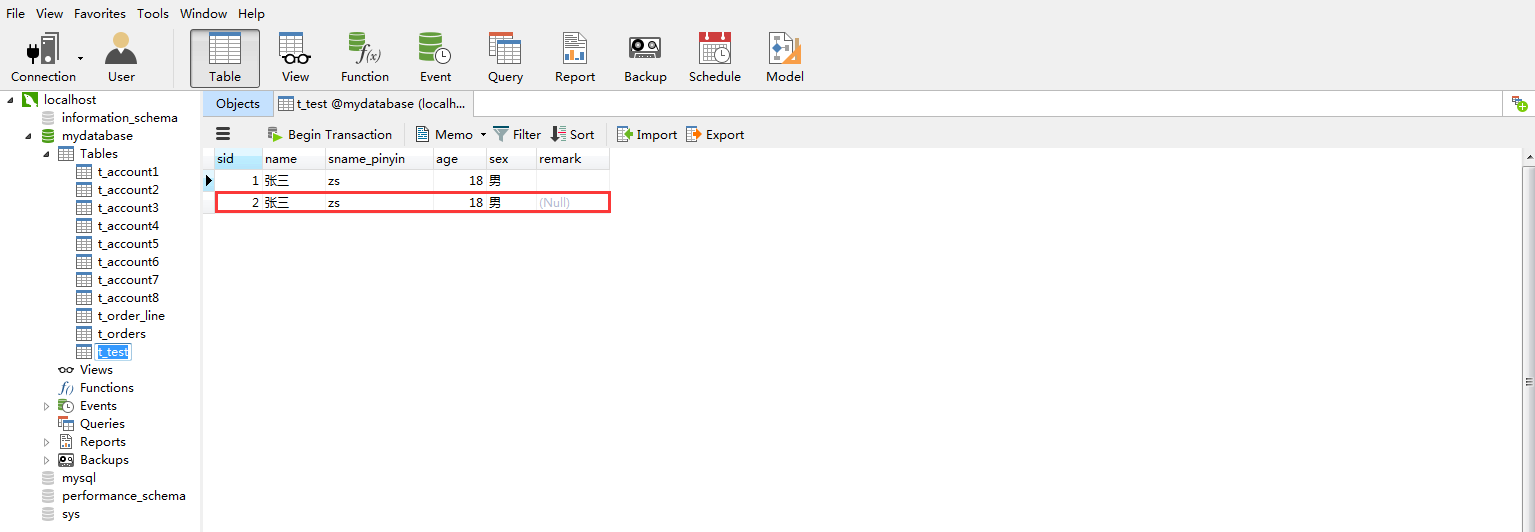
运行后效果如图：



### 2.插入部分数据

insert into t\_test (sid, name, sname\_pinyin, age, sex) values('2', '张三', 'zs', '18', '男');

运行后效果如图：



### 5.1.3插入多记录

insert into t\_test (sid, name, sname\_pinyin, age, sex, remark)

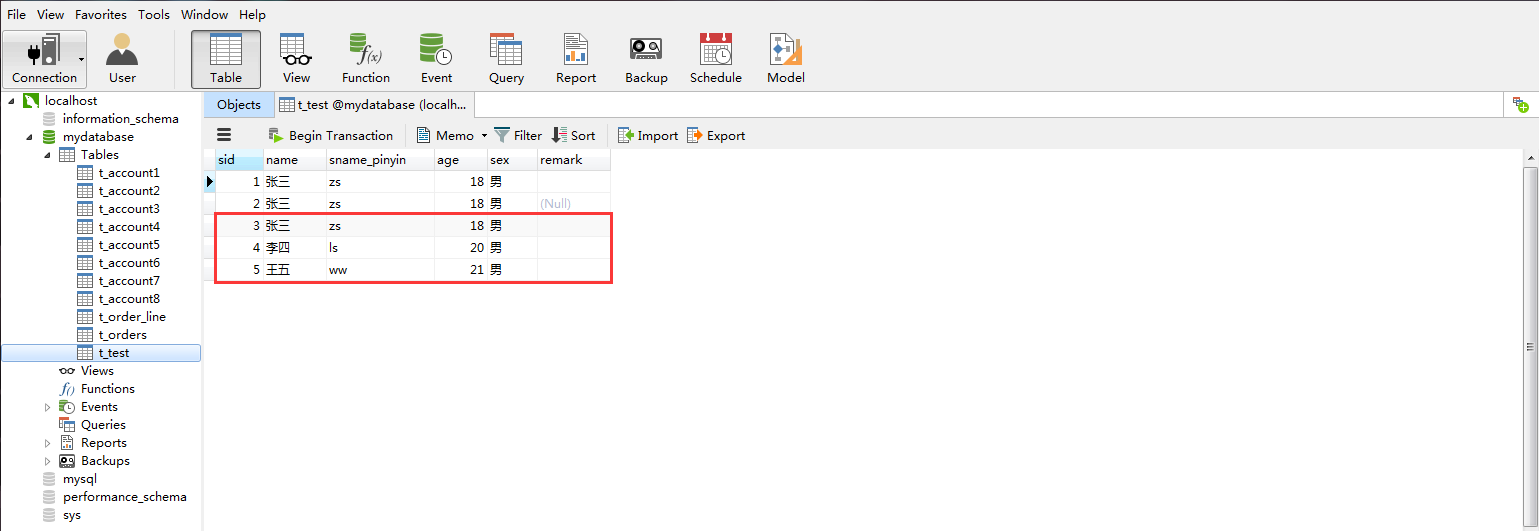
values

('3', '张三', 'zs', '18', '男', ''),

('4', '李四', 'ls', '20', '男', ''),

('5', '王五', 'ww', '21', '男', '');

运行后效果如图：



## 5.2修改数据

基础语法：

UPDATE [LOW\_PROORITY] [IGNORE] 数据表名 SET 字段1=值1 [,字段2=值2…]

[where 条件表达式]

[order by …]

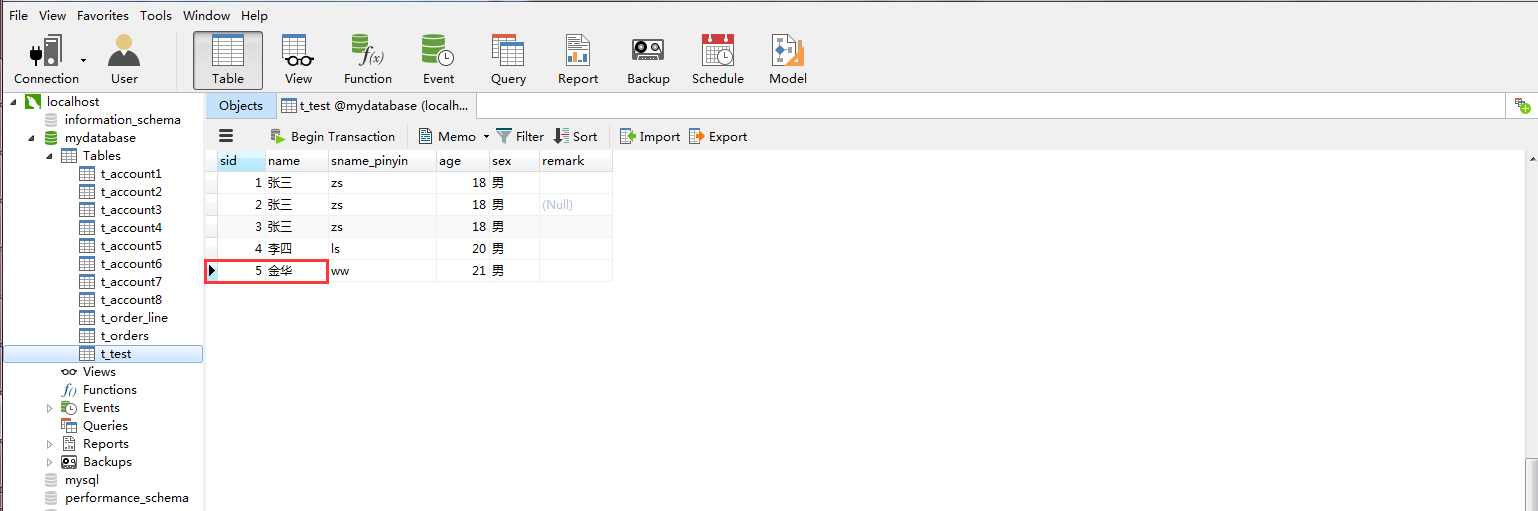
[limit 行数]

参数说明：

案例1：更新id为5的记录的姓名为“金华”

update t\_test SET name='金华' where sid='5';

运行效果如图：



## 5.3删除数据

### 5.3.1.通过delete语句删除数据

基础语法：

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE] FROM tbl\_name [PARTITION (partition\_name [, partition\_name] ...)] [WHERE where\_condition] [ORDER BY ...] [LIMIT row\_count]

参数说明：同上（修改数据）

案例1：删除t\_test表中sid=5的记录

delete from t\_test where sid=5;

### 5.3.2通过truncate table语句删除数据

基础语法：

truncate [table] 数据库表名

参数说明：

[table]：可选字句，写不写都是一样的效果

案例1：删除t\_test表中的数据

truncate t\_test;

运行完成之后的效果就是：整个表的数据都没有了，而且如图重新再该表中添加数据的时候，sid还是会从1开始

# 数据的查询

## 6.1基本查询语句

select [distinct] <字段列表>

from <数据表>

[<连接类型> join <数据表> on <连接条件>]

[where <查询条件>]

[group by <字段列表>]

[having <条件表达式>]

[limit [offset,] <限制行数>]

参数说明：

。。。

## 6.2表单查询

### 6.2.1简单数据记录查询

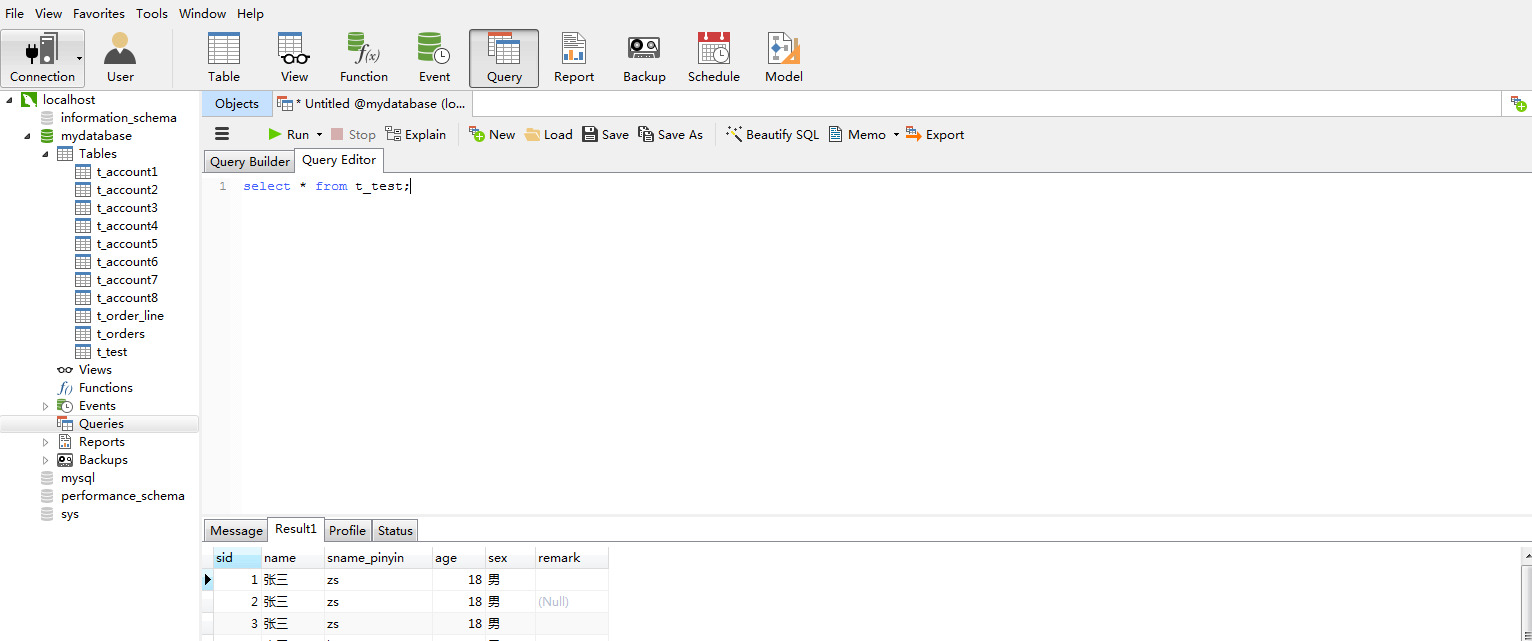
基础语法：

select \* | <字段列表> from 数据表；

案例1：\* 查询所有字段

select \* from t\_test;

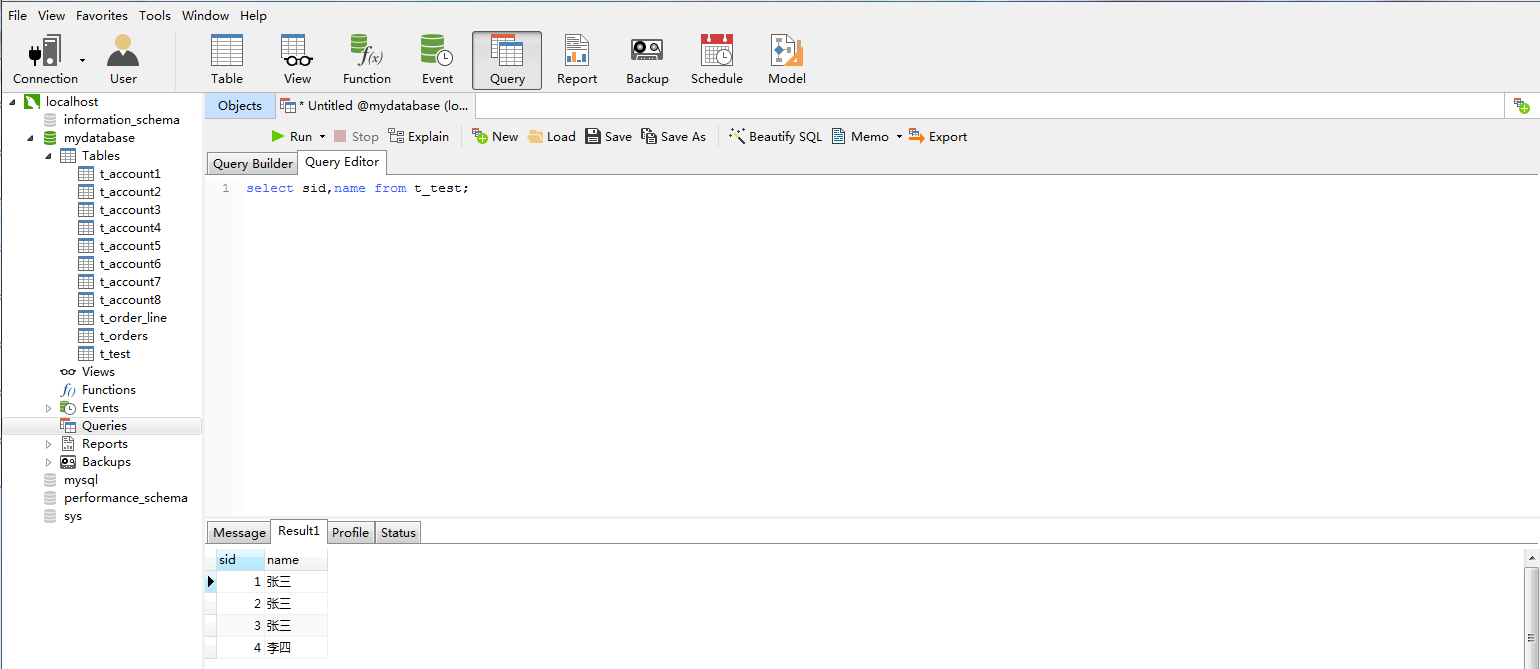
查询结果如图：



案例2：指定需要查询的字段

select sid，sname from students；

查询结果如图：



1. 去除重复查询结果

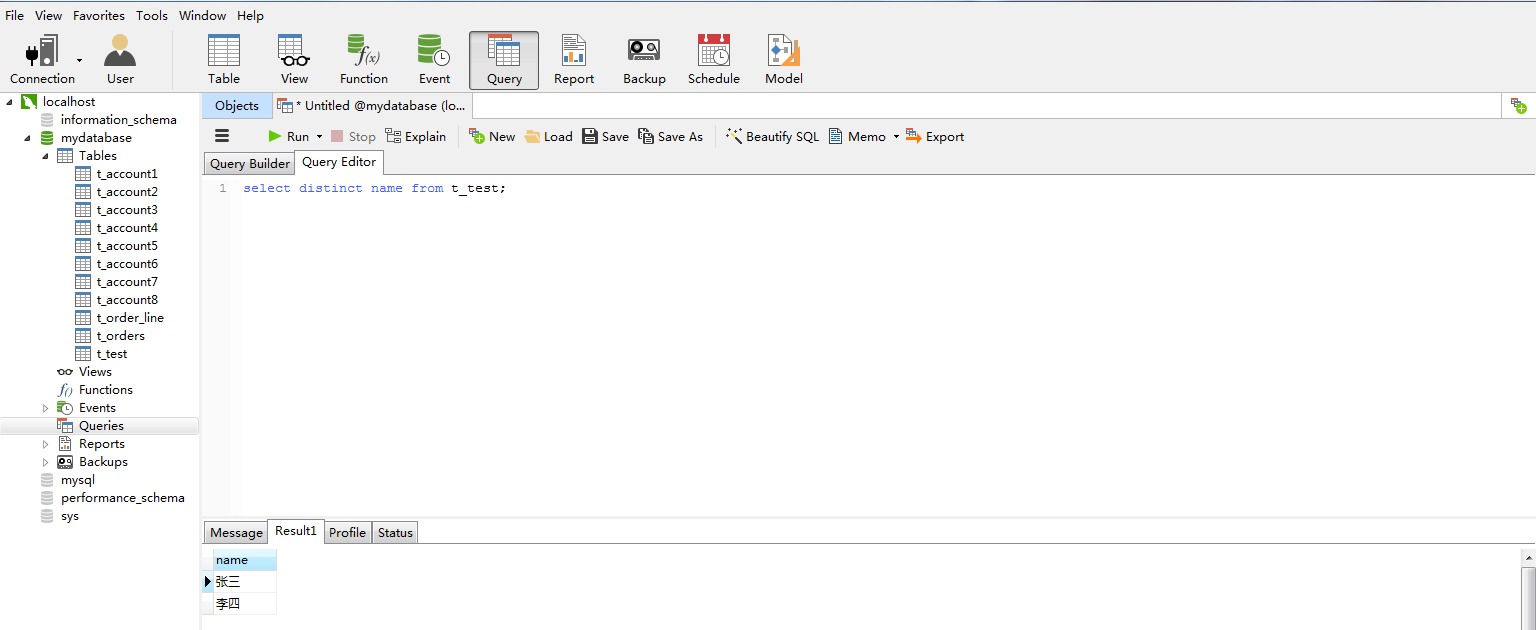
基础语法：

select distinct 字段名 from 表名；

案例1：

select distinct name from t\_test;

查询效果如图：



注意：关键字distinct一旦使用，将会应用于所有指定的字段，而不仅是某一个字段，也就是说素有字段的组合值重复时才会被消除

### 6.2.3限制查询结果数量

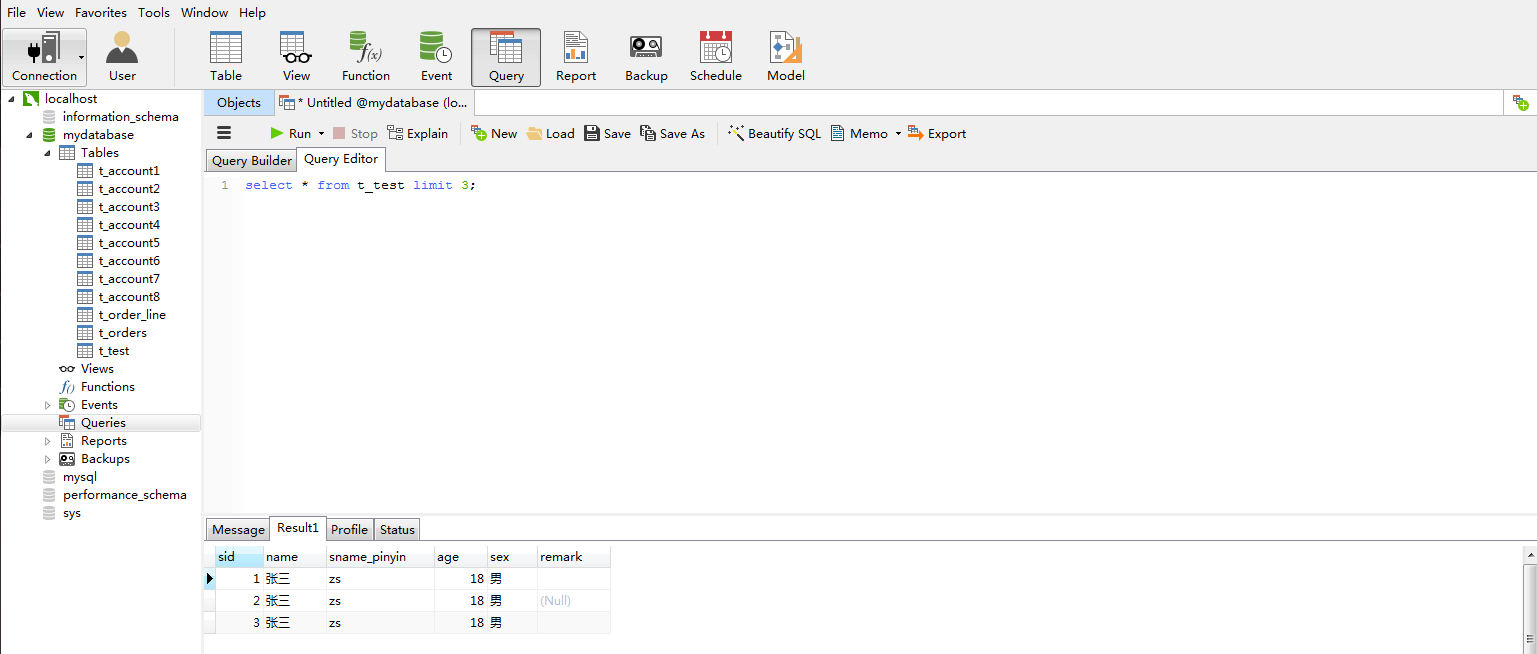
基础语法：

select \* | <字段列表> from 表名 limit [位置便宜量，] 行数；

案例1：查询t\_test表中的前3条数据

select \* from t\_test limit 3;

运行效果如图：



扩展：oracle中限制结果数量的关键字是top

### 6.2.4条件查询语句

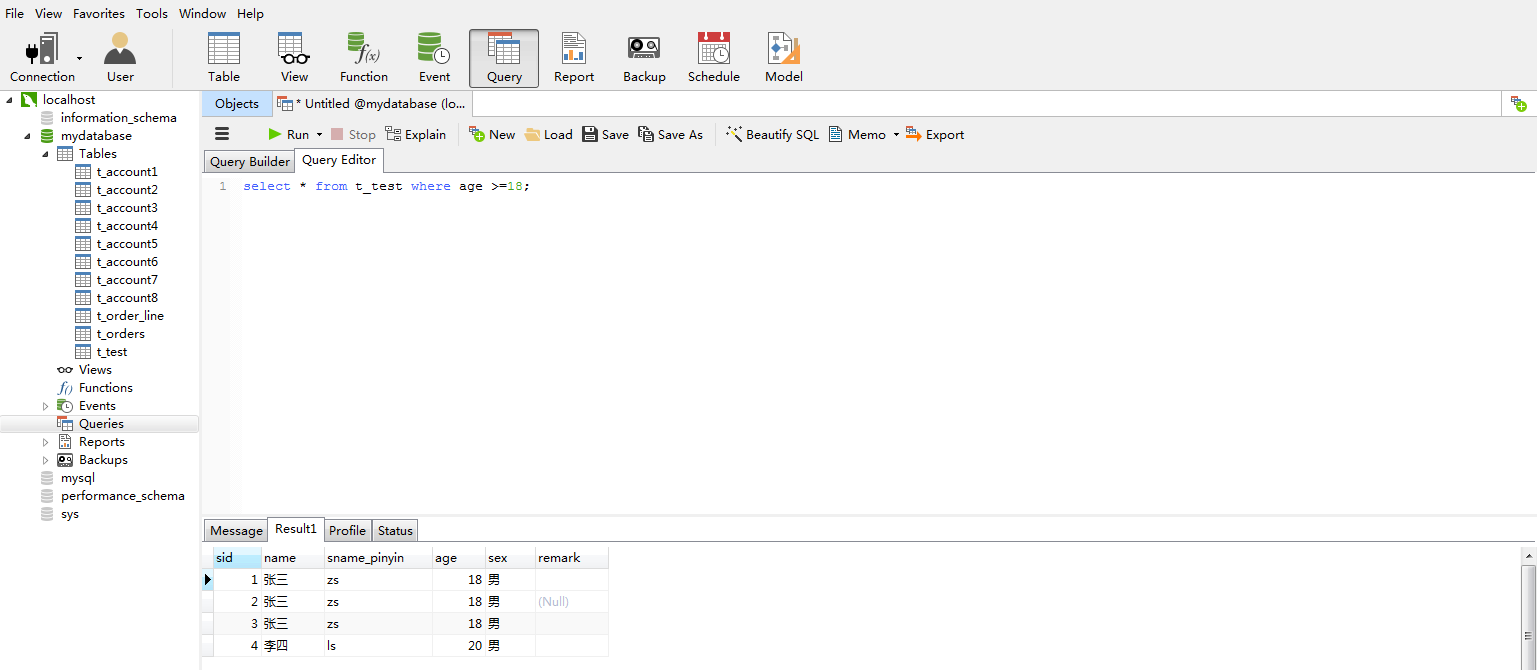
基础语法：

select \*|<字段列表> from 表名 where 查询条件

1. 关系运算条件查询

|  |  |
| --- | --- |
| 运算符 | 描述 |
| > | 大于 |
| < | 小于 |
| = | 等于 |
| ！= | 不等于 |
| >= | 大于或等于 |
| <= | 小于或等于 |

案例1：查询学生年龄在大于等于18的

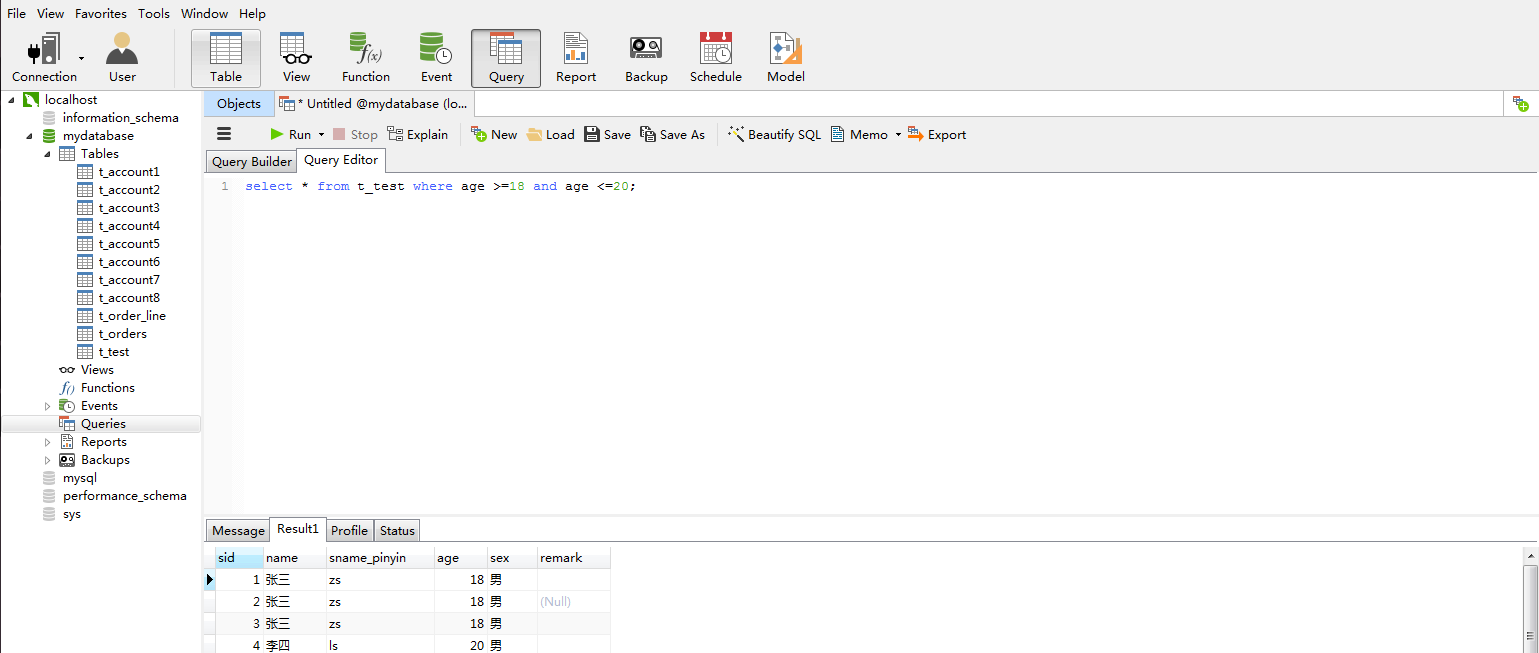


1. 逻辑运算条件查询

|  |  |
| --- | --- |
| 运算符 | 描述 |
| and (&&) | 逻辑与 |
| or (||) | 逻辑或 |
| xor | 逻辑异或 |
| not (!) | 逻辑非 |

案例1：查询年龄大于等于18并且小于等于19的学生

运行效果图：



1. 带关键字between and查询

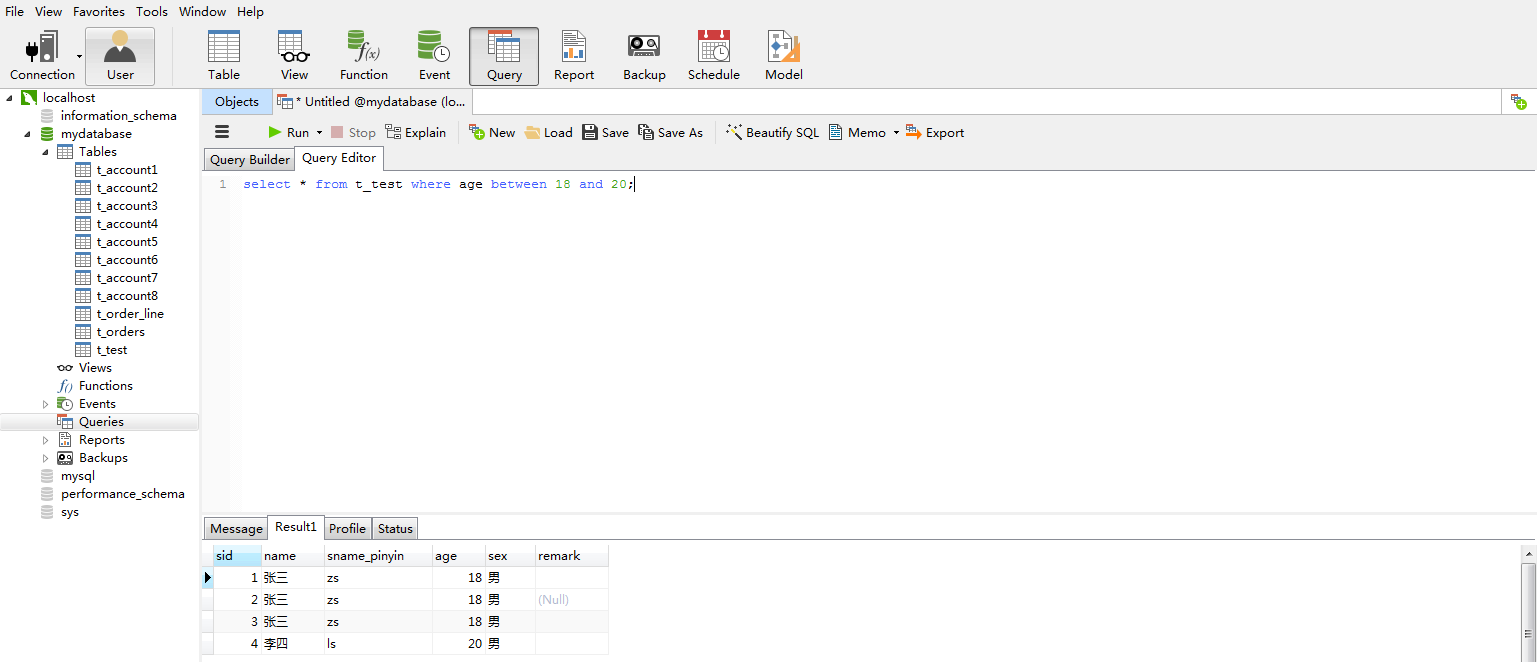
基础语法：

select 字段1，字段2…  
from 表名

where 字段1 between 值1 and 值2；

案例1：查询年龄在18到20之间的学生

运行结果如图：



1. 带关键字like查询

关键字like支持的通配符有%和\_

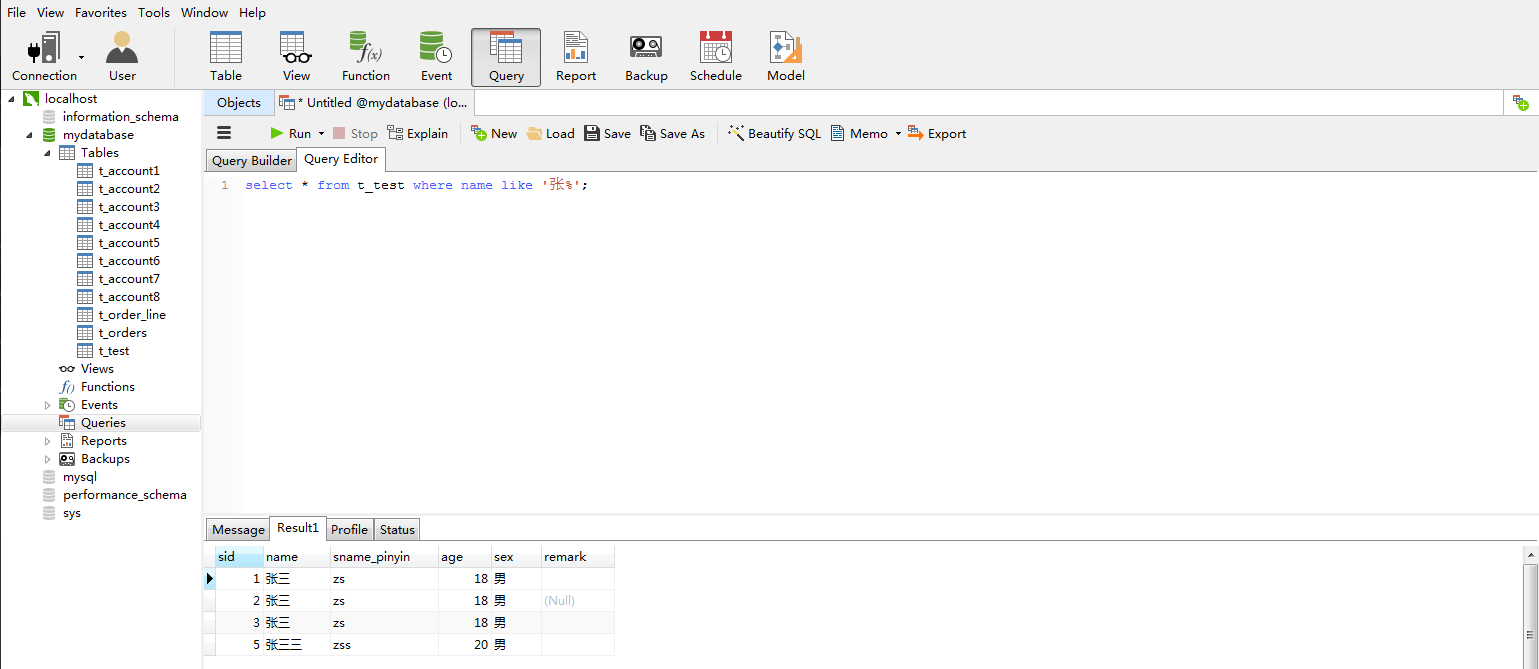
%:可以匹配任意长度的字符

\_:只能匹配单个字符

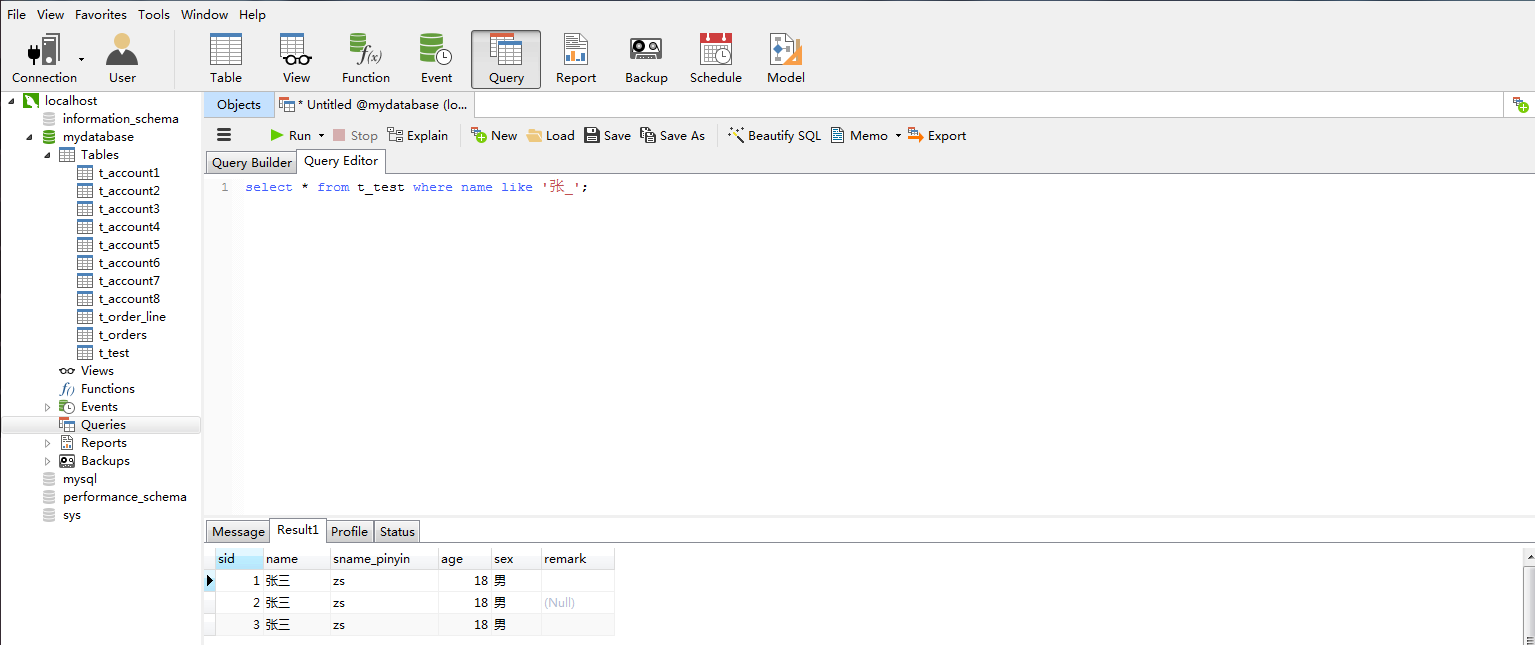
基础语法：

案例1：查询名字以张开头的学生

select \* from t\_test where name like '张%';



案例2：查询名字以张开头且名字长度为2的学生



注意：like运算符要对字段数据进行逐一扫描匹配，实际执行的效率比较差

1. 带关键字is null的空值查询

基础语法：

select 字段1，字段2…

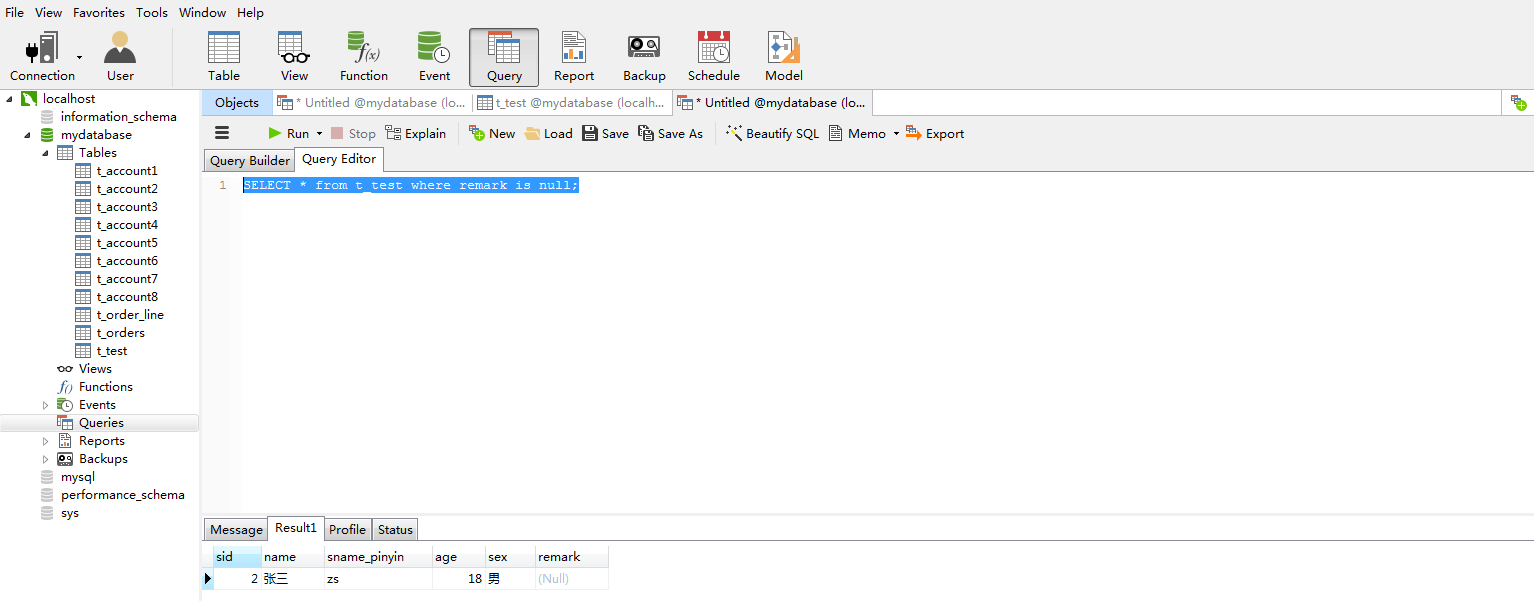
from 表名

where 字段1 is null;

案例1：查询备注is null 的学生

SELECT \* from t\_test where remark is null;

运行结果如图：



1. Order by对查询结果排序

基础语法：

select 字段1，字段2…

from 表名

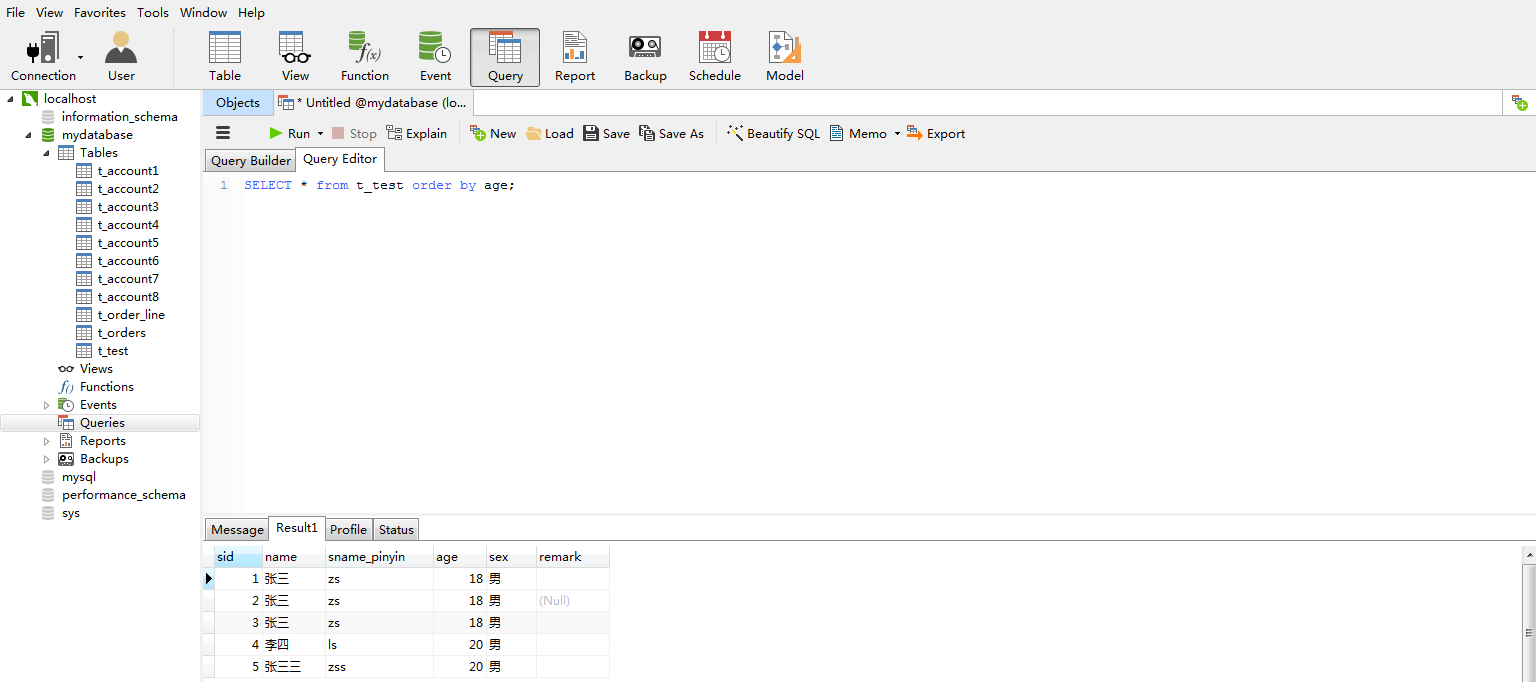
[where condition]

order by 字段1 [asc | desc] [,字段2 [asc | desc]];

案例1：根据年龄字段排序

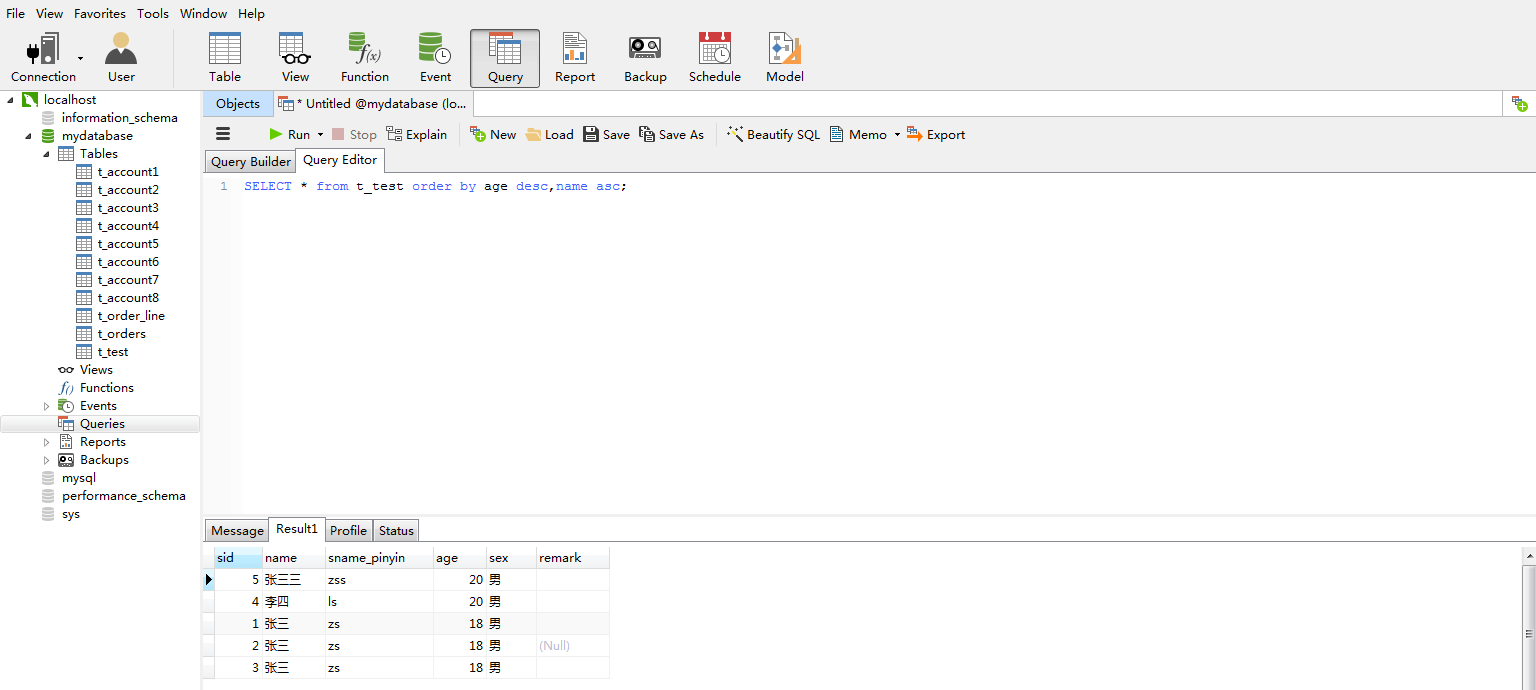
select \* from t\_test order by age;

运行结果：



案例2：SELECT \* from t\_test order by age desc,name asc;

运行结果如图：



## 6.3使用统计语句查询

|  |  |
| --- | --- |
| 函数 | 作用 |
| count | 统计表中记录的行数 |
| sum | 计算某一字段值的总和 |
| avg | 计算某一字段值的平均值 |
| max | 计算某一字段值的最大值 |
| min | 计算某一字段值的最小值 |

1. count()函数

案例1：统计性别为男的学生的人数

select count(\*) from t\_student where sex ='男';

运行效果图：

案例2：用count(borthday)统计性别为男的学生的人数

运行效果图：

count(\*)和count(具体某个字段)的区别：

count(\*)会统计所有满足where条件的记录

count(borthday)统计所有满足where条件的记录的基础上统计borthday不为空的记录数

1. sum() 函数

案例1：统计张三的各课课程的总分数

运行效果图：

1. avg() 函数

案例1：统计张三的各科课程的平均分

1. max() 函数

案例1：统计张三的各科课程的最高分

1. min() 函数

案例1：统计张三的各科课程的最高分

## 6.4分组数据查询

基础语法：

select 字段1，字段2，…

from 表名

group by 字段1，字段2，…

[having 条件表达式]

### 6.4.1单字段分组查询

案例1:根据班级进行分组

select \* from t\_student group by classes ;

运行效果如图：

运行结果只显示了每个班级的第一条记录，如果希望得到每个分组中的某个字段的所有取值，可以通过group\_concat()实现。

select group\_concat(sname) as names,classes from t\_student group by classes ;

运行效果如图：

单独使用group by实现分组意义并不大，因为它只会显示每个分组中的第一条记录，但是如果与统计函数结合使用，可以统计处每个分组的各类数据

案例1：统计每个班级的人数

select count(sname),classes from t\_student group by classes ;

运行效果如图：

注意：

(1) 出现在select后面的字段 要么是是聚合函数中的,要么就是group by 中的.

(2) 要筛选结果 可以先使用where 再用group by 或者先用group by 再用having

### 6.4.2多字段分组查询

案例2：根据班级和性别进行分组

select \* from t\_student group by classes,sex;

select classes,sex,count(\*) from t\_student group by classes,sex;

有问题???????? 也是只是会显示每个分组的第一条记录呀

### 6.4.3 having子句限定分组查询

having关键字和where关键字都是用于设置条件表达式的，两者的区别在于，1.having关键字后可以有统计函数，而where关键字则不能

2.where关键字的作用是在于对查询结果进行分组前将不符合where条件的记录去掉，即在分组之前过滤数据，having子句的作用是筛选满足条件的组，即在分组过滤数据

案例1：根据班级进行分组

select group\_concat(sid) as sids,classes from t\_student group by classes;

案例2：根据班级进行分组，使用having子句过滤出学生的数量大于2个的班级

select group\_concat(sid) as sids,classes from t\_student group by classes having count(sid) >2;

## 6.5连接查询

### 6.5.1 笛卡尔查询

基础语法：

案例1：

select \* from t\_book b,t\_category c;

运行效果如图：

### 6.5.2 有效的笛卡尔查询

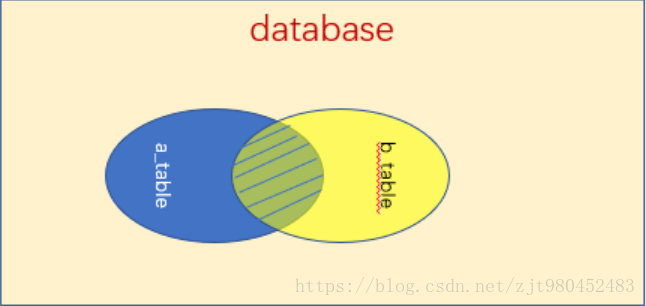
案例1：

select \* from t\_book b,t\_book\_category bc where b.book\_id =bc.bid;

### 6.5.2内连接

案例1：

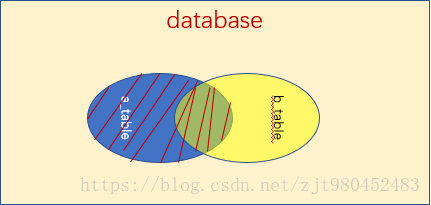
select \* from t\_book b inner join t\_book\_category bc on b.book\_id =bc.bid;



### 6.5.3左外连接

案例2：

select \* from t\_book b left join t\_book\_category bc on b.book\_id =bc.bid;



注意：内连接和外连接的区别

1. 内连接其实就是满足条件的笛卡尔查询
2. 外连接主表中的记录在从表中找不到匹配条件时，用null匹配

### 6.5.3多表联查

案例1：

select \* from t\_book b,t\_book\_category bc,t\_category c where b.book\_id =bc.bid and c.category\_id=bc.cid;

运行效果演示：

## 6.6子查询

1. 带in的子查询
2. 带exists关键字的子查询
3. 带any关键字的子查询
4. 带all关键字的子查询
5. 带比较运算符的子查询
6. 合并查询结果

合并查询结果将多条select语句的查询结果合并到一起组成单个结果集。进行合并操作时，两个结果集对应的列数和数据类型必须相同。每个select语句之间使用union或者union all关键字分割，union关键字会去除合并结果集中重复的数据记录，而union all关键字不会。

基础语法：

select 语句1 union |union all

select 语句2 union |union all

select 语句n；

案例1：

。。。

# 第七章、视图

## 7.1视图概述

1. 视图是从一个或多个基本表中导出的虚拟表。视图与基本表不同，视图不对数据进行实际存储，数据库中只存储视图的定义，用户对视图数据进行操作时，系统会跟进视图定义去操作相关联的基本表，Mysql从5.0开始可以使用视图

## 7.2视图创建

基础语法：

create [or replace] view 视图名 [(字段名，…)]

as select 语句

[with [cascaded | local] check option ]

参数说明：

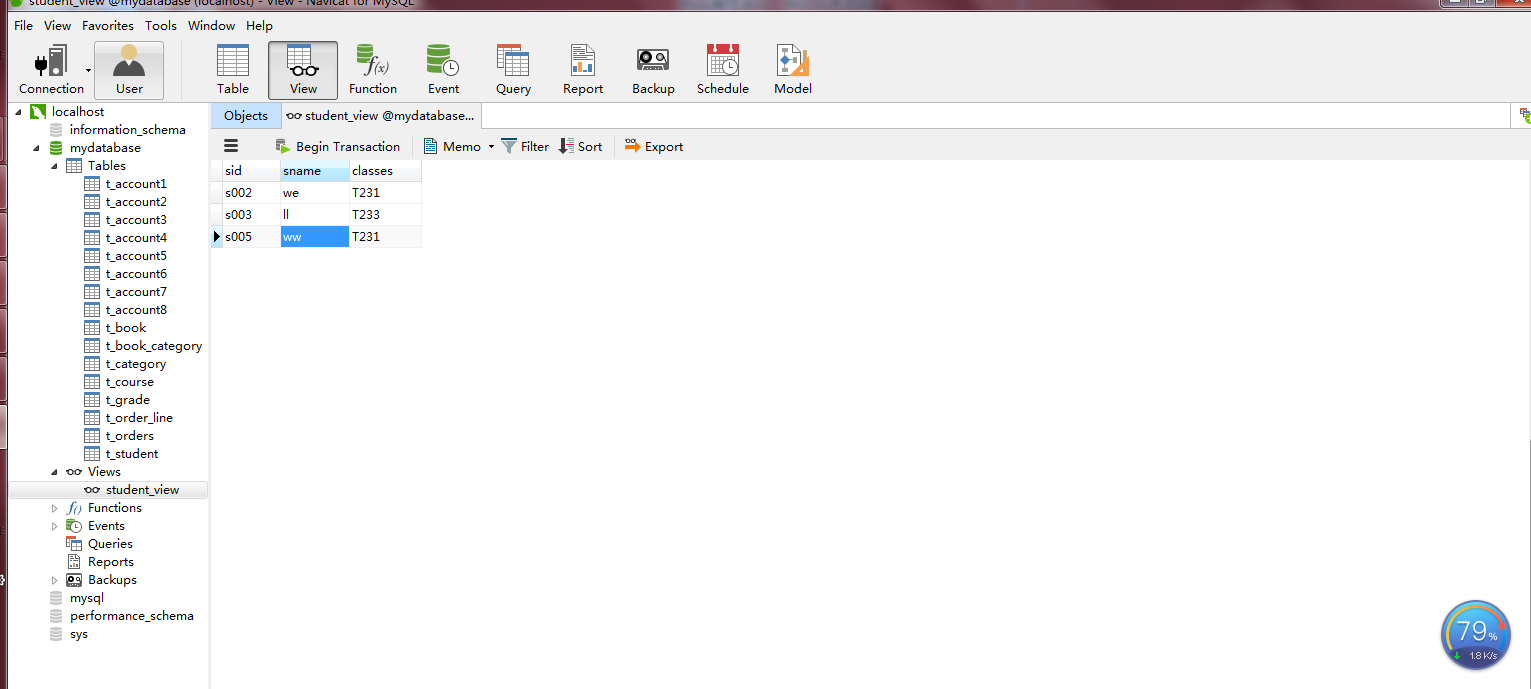
。。。

案例1：创建student\_view视图，包括student表的sid，sname，classes字段

create view student\_view

as select sid,sname,classes from t\_student;

运行效果如图：



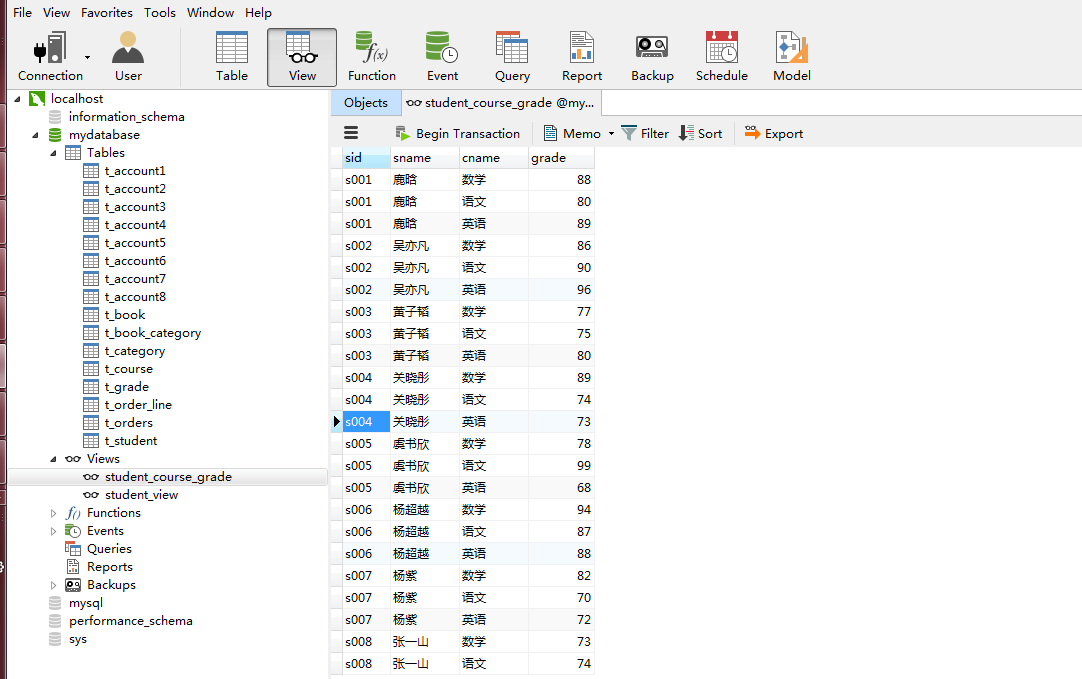
案例2：创建视图student\_course\_grade,包含sid，sname，cname，grade

因为涉及到的字段来自t\_student，t\_course,t\_grade，所以需要进行多表联查

create view student\_course\_grade

as select s.sid,s.sname,c.cname,g.grade from t\_student s, t\_grade g ,t\_course c where s.sid=g.sid and c.cid=g.cid;

运行效果如图：

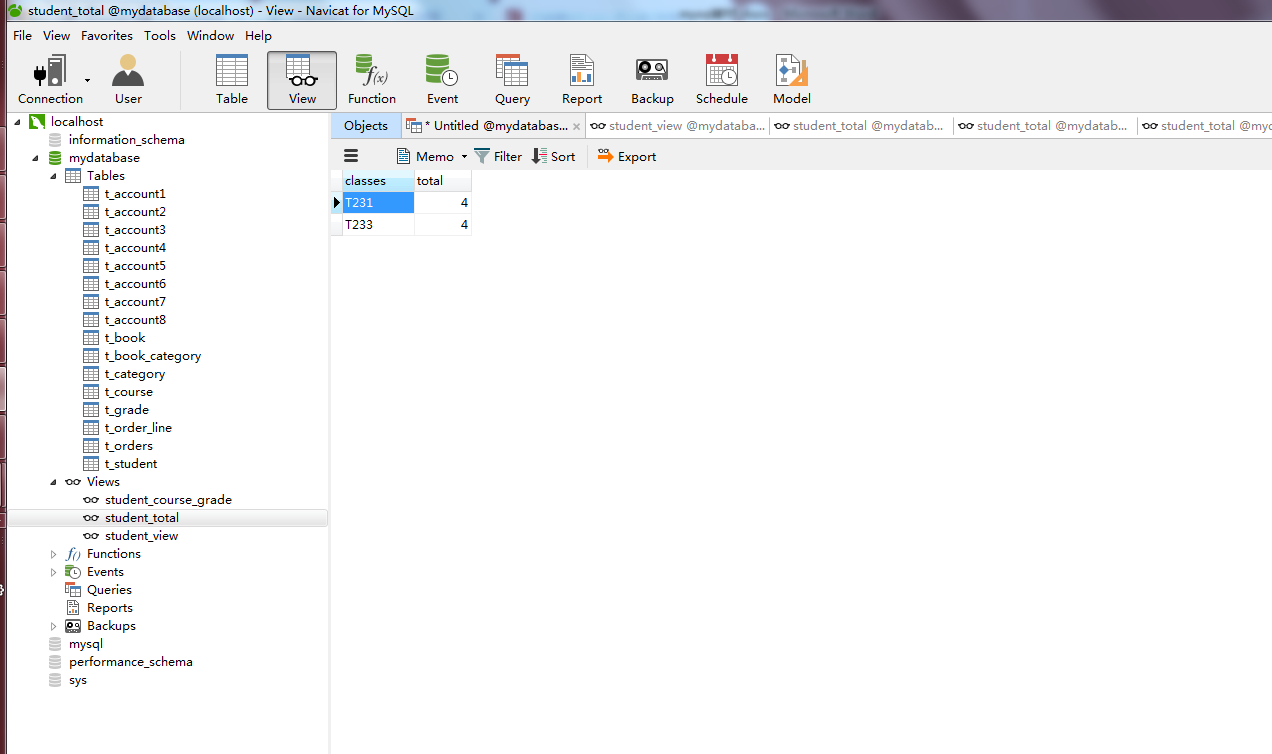


案例3：创建一个视图，包含classes字段，以及每个班的人数

create view student\_total(classes,total)

as select classes, count(sname) from t\_student group by classes;

运行效果如图：



## 7.3视图操作

### 7.3.1查看视图

1.使用describe语句查看视图的基本信息

基础语法：

describe 视图名；

案例1: describe student\_view;



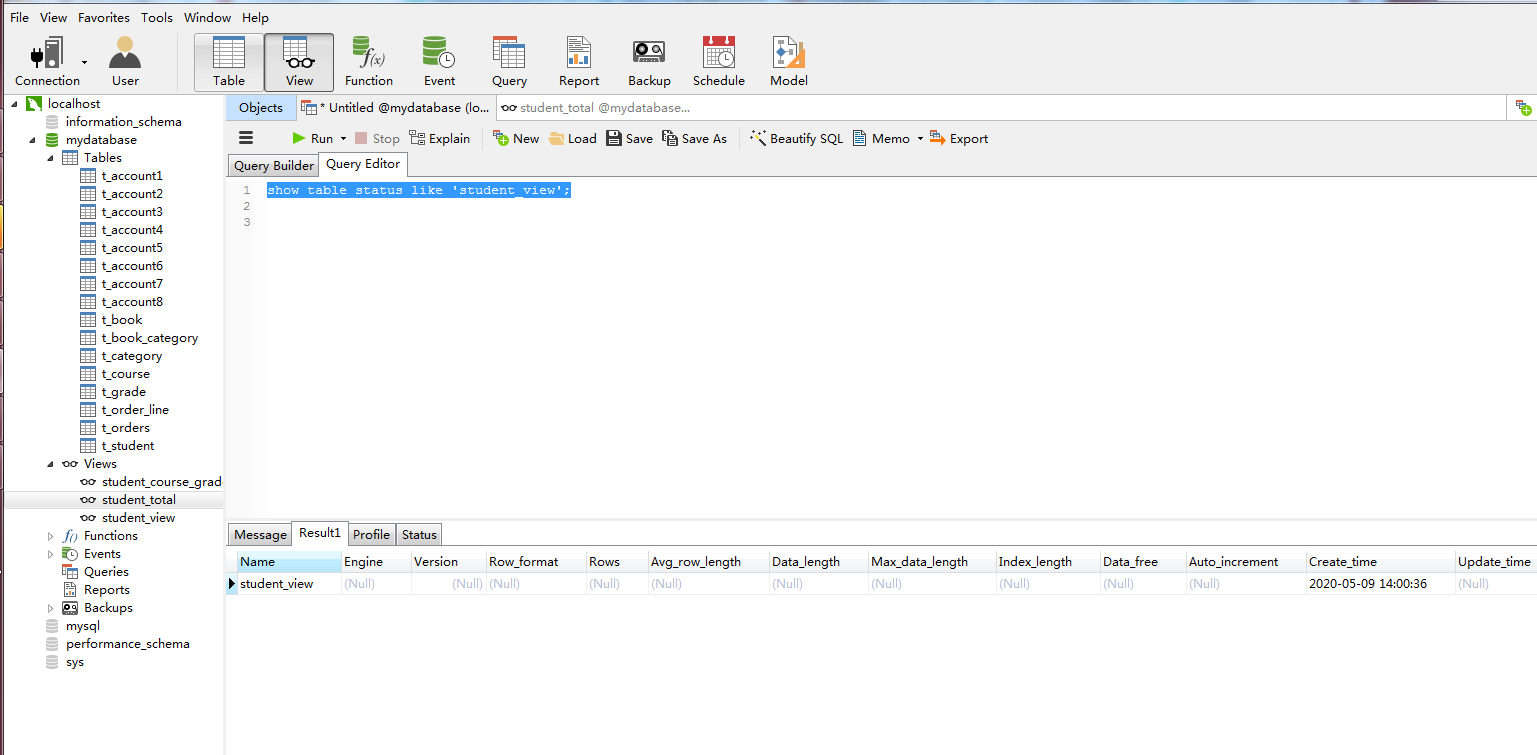
2.使用show table status语句查看视图的基本信息

基础语法：

show table status like '视图名' ；

案例1:

show table status like 'student\_view';



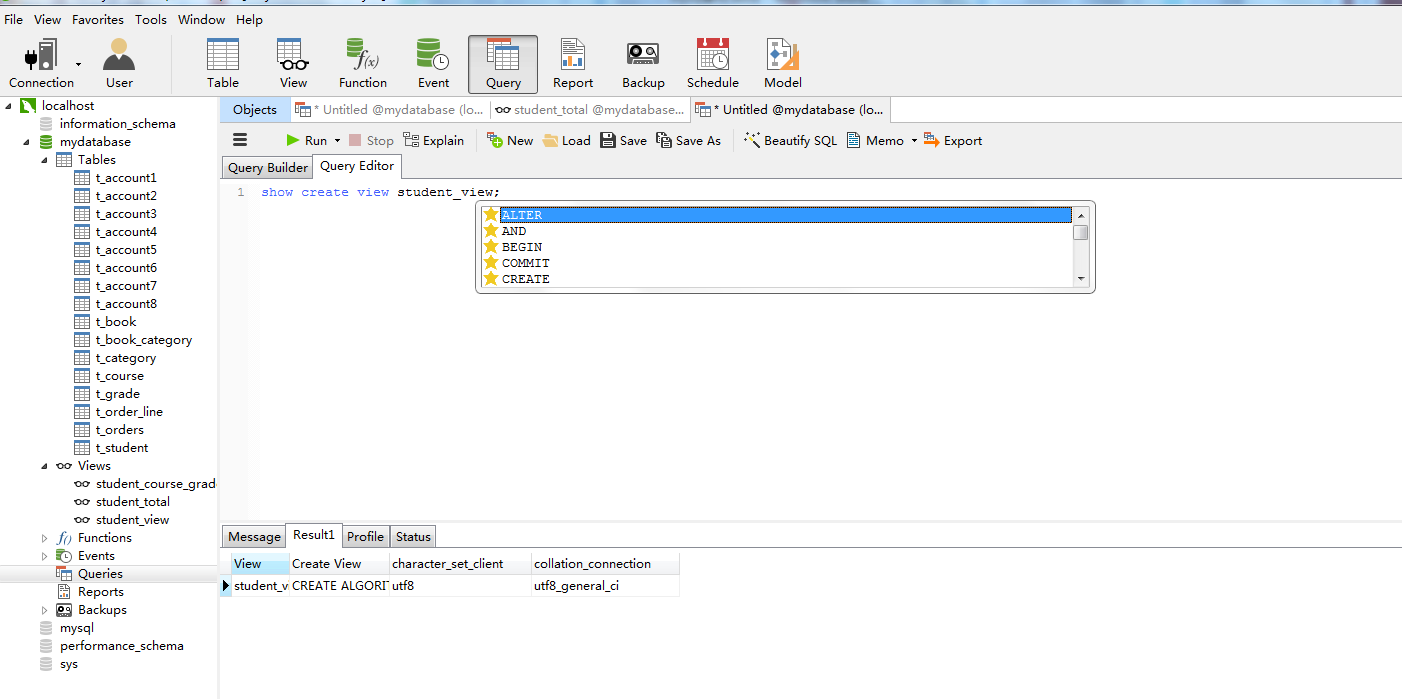
3.使用show create view语句查看视图的基本信息

基础语法：

show create view视图名；

案例1:

show create view student\_view;



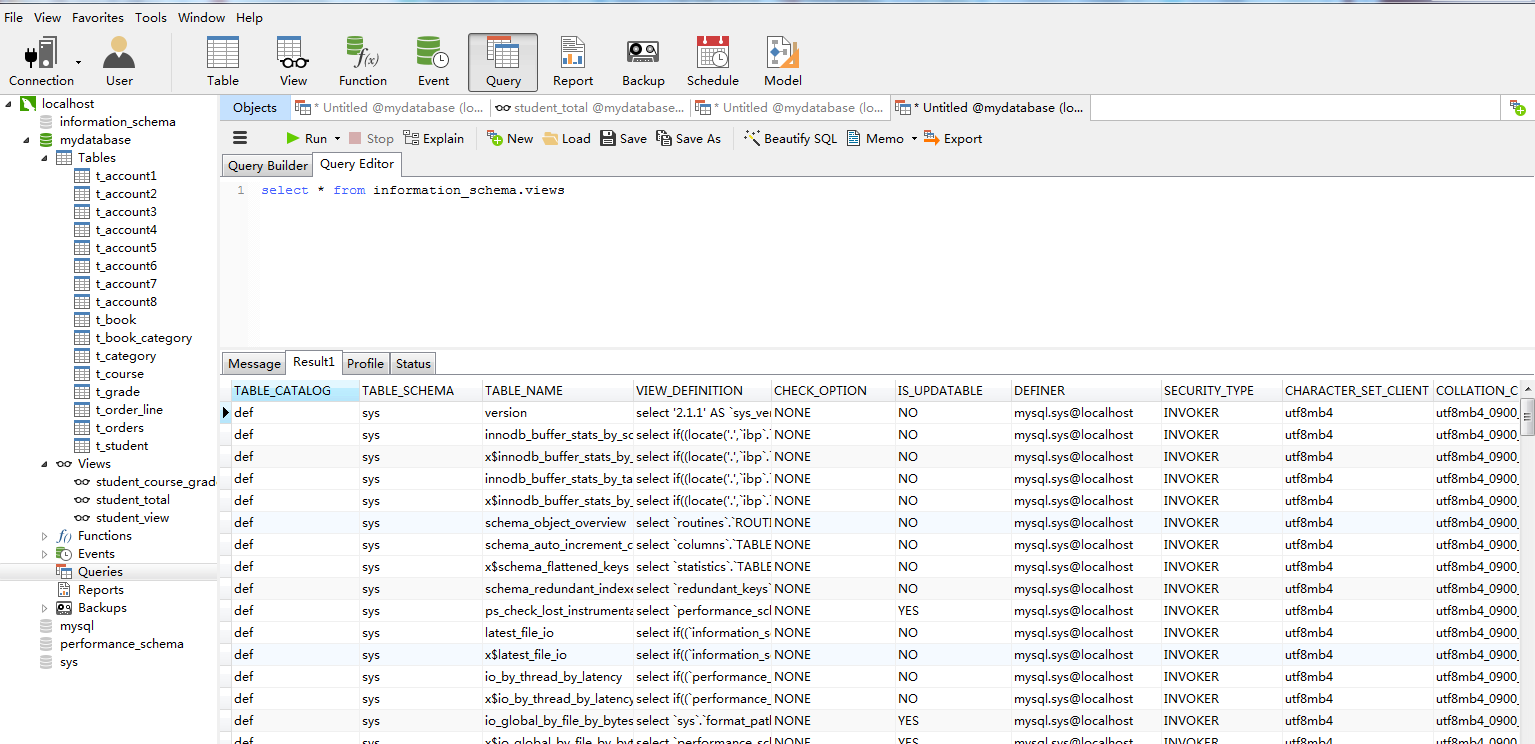
4.在view表中查看视图的详细信息

基础语法：

select \* from information\_scheme.views；

案例1:

show create view student\_view;



### 7.3.2修改视图

1.使用create or replace view修改视图

基础语法：

create or replace view 视图名 [(字段名，…)]

as select 语句

[with [cascaded | local] check option ]

案例1：

create or replace view student\_view

as select sid,sname,classes,borthday from t\_student;

使用alterview语句修改视图

基础语法：

alter view 视图名 [(列名，…)]

as select 语句

[with [cascaded | local] check option ]

案例1：

。。。

1. 更新视图

更新视图是指通过视图插入，更新和删除表中的数据，因为视图是一个虚拟表，所以更新视图就更新其关联表中的数据，要通过视图更新基本表数据，必须保证视图是可更新视图，即可以在insert update delete等语句中使用他们，对于可更新视图，视图中的行和基本表中的行必须具有一对一的关系，如果视图中包含下述表结构中的任意一种，那么它就是不可更新视图。

1. 聚合函数
2. distinct关键字
3. group by子句
4. order by子句
5. having子句
6. union运算符
7. 位于选择列表中的子查询
8. from子句中包含多个表
9. select 语句中引用了不可更新视图
10. where子句中的子查询，引用from子句中的表
11. 删除视图

基础语法：

drop view [if exists] 视图名，…

参数说明：

案例1：

。。。

# 第八章、索引

1. 索引概述

数据库中的索引类似于书中的目录，表中的数据类似于书中的内容，读者可以通过书的目录找到某些内容所在的具体位置，同理数据库的索引有助于快速检索数据，，在关系型数据库中，索引是一种可以加快数据检索的数据结构，主要用于提高性能，因为检索可以从大量数据中迅速找到所需要的数据，不再需要检索整个数据库

1. 索引含义和特点
2. 索引分类

普通索引：仅加速查询

唯一索引：加速查询 + 列值唯一（可以有null）

主键索引：加速查询 + 列值唯一（不可以有null）+ 表中只有一个

组合索引：多列值组成一个索引，专门用于组合搜索，其效率大于索引合并

全文索引：对文本的内容进行分词，进行搜索

ps.索引合并，使用多个单列索引组合搜索

*覆盖索引，select的数据列只用从索引中就能够取得，不必读取数据行，换句话说查询列要被所建的索引覆盖*

1. 创建索引

创建表时创建索引

基础语法：

CREATE TABLE  表名( 属性名 数据类型[完整性约束条件],   
属性名 数据类型[完整性约束条件],   
......   
属性名 数据类型    
[ UNIQUE | FULLTEXT | SPATIAL ]  INDEX | KEY   
[ 别名]  ( 属性名1  [(长度)]  [ ASC | DESC] )   
);

参数说明：

案例1：

。。。

在已存在表中创建索引

基础语法：

alter table 表名

add primary key (字段名，… [asc|desc])

|add index [索引名] (字段名，… [asc|desc])

|add [unique|fulltext|spatial] [index|key] [索引名] (字段名，… [asc|desc])

参数说明：

案例1：

。。。

1. 删除索引

使用alter table语句删除索引

基础语法：

alter table 表名

|drop primary key

|drop index 索引名

案例1：

。。。

使用drop index语句删除索引

基础语法：

drop index 索引名 on 表名

案例1：

。。。

# 第九章、存储过程和触发器

1. sql编程基础
2. 存储过程和函数
3. 创建和调用存储过程

创建存储过程

基础语法：

create procedure 存储过程名 （[参数1，…]） [特性]

参数说明：

案例1：

。。。

调研存储过程

基础语法：

call 存储过程名（[参数1，…]）

参数说明：

案例1：

。。。

1. 创建和调用存储函数

创建存储函数

基础语法：

create fuction 存储函数名 （[参数1，…]） [特性]

returns 类型

存储数体

参数说明：

案例1：

。。。

调研存储函数

基础语法：

参数说明：

案例1：

。。。

1. 查看存储过程和函数

使用show status语句来查看存储过程和函数的状态

基础语法：

show procedure|function status [like ‘字符串’]

参数说明：

案例1：

。。。

使用show create语句来查看存储过程和函数的定义

基础语法：

show create procedure 存储过程名

参数说明：

案例1：

。。。

1. 修改存储过程和函数

基础语法：

alter procedure function存储过程或函数名 [特性]

参数说明：

案例1：

。。。

1. 删除存储过程和函数

基础语法：

drop procedure | function [if exists] [数据库名.] 存储过程函数或过程名

案例1：

。。。

1. 设置触发器
2. 创建触发器

基础语法：

create trigger 触发器名 触发事件 触发事件

on 表名for each row 触发器动作

参数说明:

案例1:

。。。

1. 查看触发器

使用show trigger 语句来查看当前数据库中的所有触发器

基础语法：

show triggers；

运行结果：

使用triggers 表中查看触发器

基础语法：

select \* from informatinn\_schema.triggger [where 查询条件]；

运行结果：

1. 删除触发器

基础语法：

drop trigger [if exists] [数据库名.] [触发器名]

案例1：

。。。

# 第十章、事务

1. 事物概述
2. 事务的创建与存在周期
3. 初始化事务

基础语法：

start transction;

1. 创建事务
2. 提交事务

在用户没有提交事务前，当其他的用户连接mysql服务器时，使用select语句查询结果，不会显示没有提交的事务

1. 撤销事务

撤销事务又被称为事务回滚

rollback;

1. Mysql事务行为
2. 自动提交
3. 事务的孤立级
4. 修改事务的孤立级
5. 事务的性能
6. 应用小事务
7. 选择合适的孤立级
8. 死锁的概念与避免

死锁即当两个或者多个处于不同序列的

# 第十一章、数据安全

、