# MySQL基础

**今日目标:**

> - 完成MySQL的安装及登陆基本操作
> - 能通过SQL对数据库进行CRUD
> - 能通过SQL对表进行CRUD
> - 能通过SQL对数据进行CRUD

# 1，数据库相关概念

以前我们做系统，数据持久化的存储采用的是文件存储。存储到文件中可以达到系统关闭数据不会丢失的效果，当然文件存储也有它的弊端。

假设在文件中存储以下的数据：

```
姓名   年龄   性别   住址
张三   23   男     北京西三旗
李四   24   女     北京西二旗
王五   25   男     西安软件新城
```

现要修改李四这条数据的性别数据改为男，我们现学习的IO技术可以通过将所有的数据读取到内存中，然后进行修改再存到该文件中。通过这种方式操作存在很大问题，现在只有三条数据，如果文件中存储1T的数据，那么就会发现内存根本就存储不了。

现需要既能持久化存储数据，也要能避免上述问题的技术使用在我们的系统中。数据库就是这样的一门技术。

## 1.1 数据库

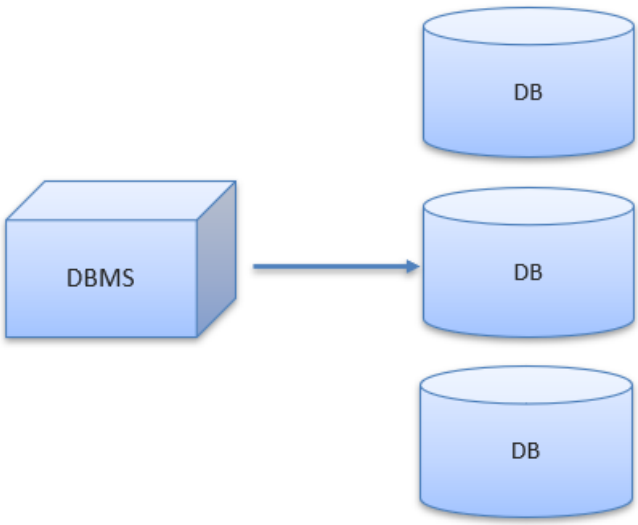
- ==存储和管理数据的仓库，数据是有组织的进行存储。==
- 数据库英文名是 DataBase，简称DB。

数据库就是将数据存储在硬盘上，可以达到持久化存储的效果。那又是如何解决上述问题的？使用数据库管理系统。

## 1.2 数据库管理系统

- ==管理数据库的大型软件==
- 英文：DataBase Management System，简称 DBMS

在电脑上安装了数据库管理系统后，就可以通过数据库管理系统创建数据库来存储数据，也可以通过该系统对数据库中的数据进行数据的增删改查相关的操作。我们平时说的MySQL数据库其实是MySQL数据库管理系统。




通过上面的描述，大家应该已经知道了 `数据库管理系统` 和 `数据库` 的关系。那么有有哪些常见的数据库管理系统呢？

## 1.3 常见的数据库管理系统

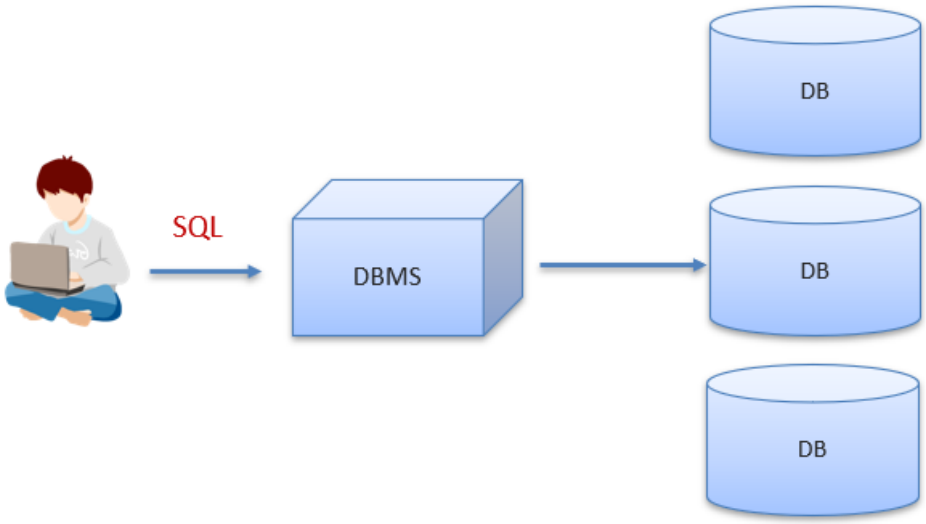
| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Apr 2018 | Mar 2018 | Apr 2017 | | | Apr 2018 | Mar 2018 | Apr 2017 |
| 1. | 1. | 1. | Oracle | Relational DBMS | 1289.79 | +0.18 | -112.21 |
| 2. | 2. | 2. | MySQL | Relational DBMS | 1226.40 | -2.46 | -138.22 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational DBMS | 1095.51 | -9.28 | -109.26 |
| 4. | 4. | 4. | PostgreSQL | Relational DBMS | 395.47 | -3.88 | +33.69 |
| 5. | 5. | 5. | MongoDB | Document store | 341.41 | +0.89 | +15.98 |
| 6. | 6. | 6. | DB2 | Relational DBMS | 188.95 | +2.28 | +2.29 |
| 7. | 7. | 7. | Microsoft Access | Relational DBMS | 132.22 | +0.27 | +4.04 |
| 8. | ↑ 9. | ↑ 11. | Elasticsearch | Search engine | 131.36 | +2.81 | +25.69 |
| 9. | ↓ 8. | 9. | Redis | Key-value store | 130.11 | -1.12 | +15.75 |
| 10. | 10. | ↓ 8. | Cassandra | Wide column store | 119.09 | -4.40 | -7.10 |
| 11. | 11. | ↓ 10. | SQLite | Relational DBMS | 115.99 | +1.17 | +2.19 |

接下来对上面列举的数据库管理系统进行简单的介绍：

- Oracle：收费的大型数据库，Oracle 公司的产品
- ==MySQL==： 开源免费的中小型数据库。后来 Sun公司收购了 MySQL，而 Sun 公司又被 Oracle 收购

- SQL Server：MicroSoft 公司收费的中型的数据库。C#、.net 等语言常使用
- PostgreSQL：开源免费中小型的数据库
- DB2：IBM 公司的大型收费数据库产品
- SQLite：嵌入式的微型数据库。如：作为 Android 内置数据库
- MariaDB：开源免费中小型的数据库

我们课程上学习的是MySQL数据库管理系统，PostgreSQL在一些公司也有使用，此时大家肯定会想以后在公司中如果使用我们没有学习过程的PostgreSQL数据库管理系统怎么办？这点大家大可不必担心，如下图所示：




我们可以通过数据库管理系统操作数据库，对数据库中的数据进行增删改查操作，而怎么样让用户跟数据库管理系统打交道呢？就可以通过一门编程语言（SQL）来实现。

## 1.4 SQL

- 英文：Structured Query Language，简称 SQL，结构化查询语言
- 操作关系型数据库的编程语言
- 定义操作所有关系型数据库的统一标准，可以使用SQL操作所有的关系型数据库管理系统，以后工作中如果使用到了其他的数据库管理系统，也同样的使用SQL来操作。

# 2，MySQL

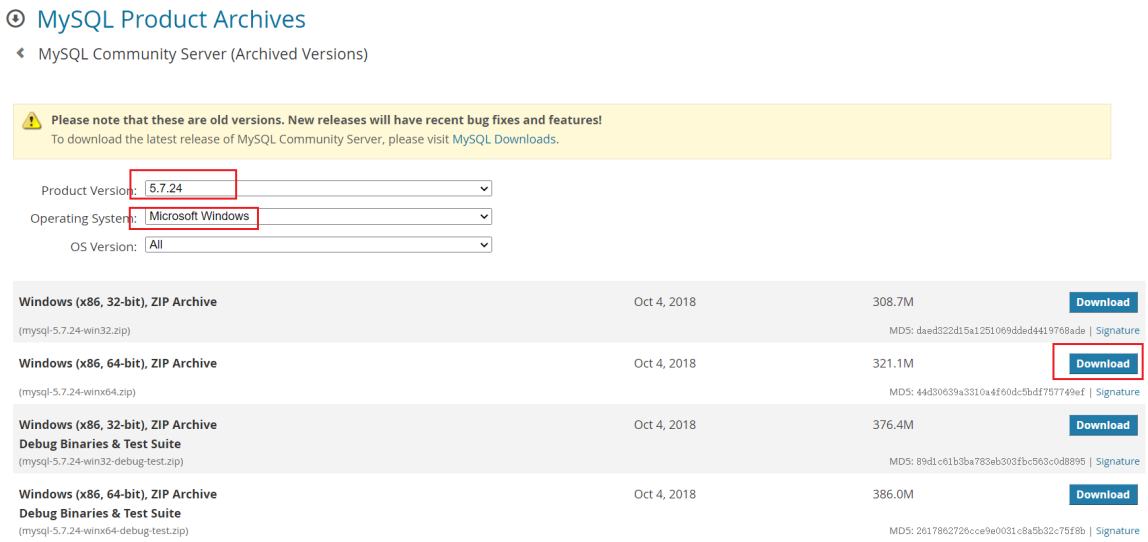
# 2.1 MySQL安装

> **安装环境:Win10 64位**
> **软件版本:MySQL 5.7.24 解压版**

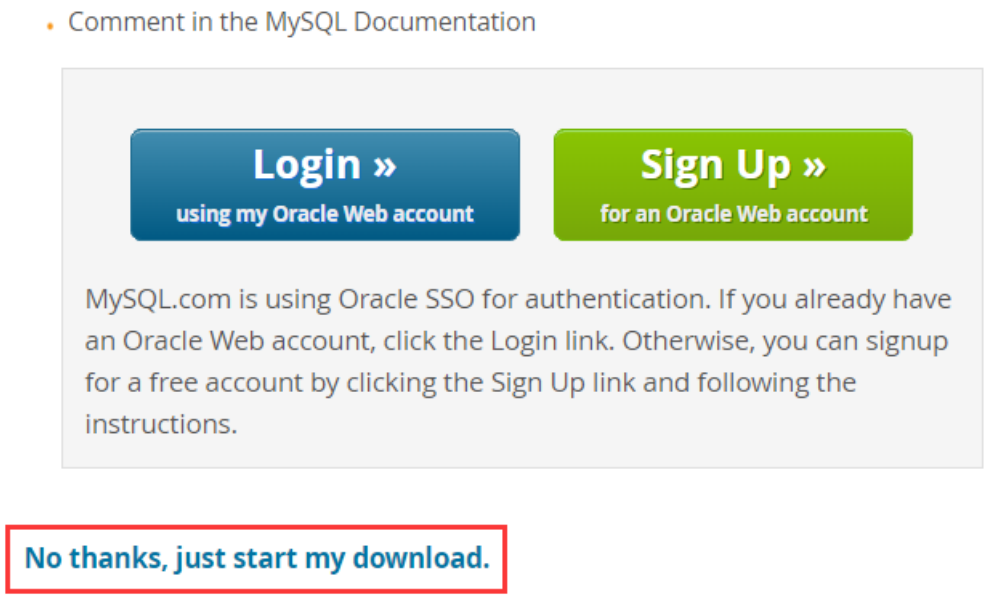
## 2.1.1 下载

https://downloads.mysql.com/archives/community/

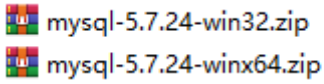
点开上面的链接就能看到如下界面:




选择选择和自己**系统位数**相对应的版本点击右边的`Download`，此时会进到另一个页面，同样在接近页面底部的地方找到如下图所示的位置:



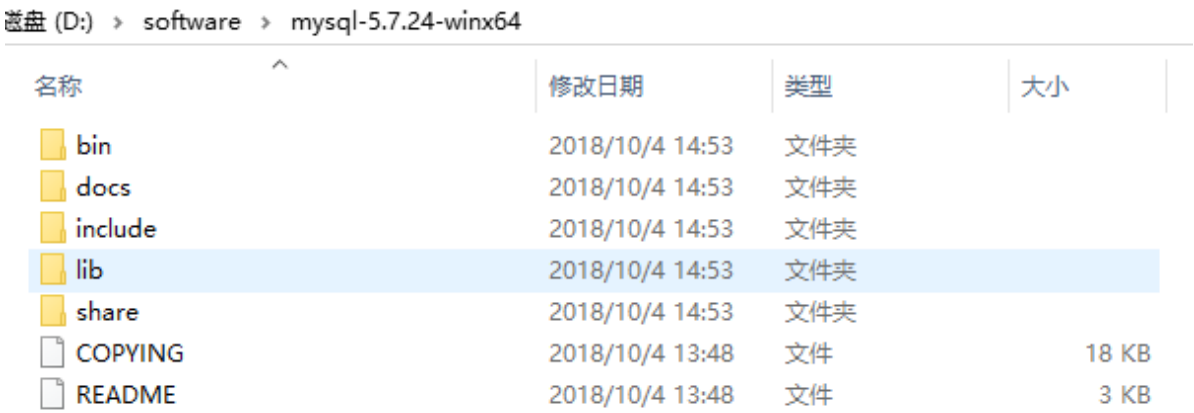

不用理会上面的登录和注册按钮，直接点击`No thanks, just start my download.`就可以下载。




### 2.1.2 安装(解压)

下载完成后我们得到的是一个压缩包，将其解压，我们就可以得到MySQL 5.7.24的软件本体了(就是一个文件夹)，我们可以把它放在你想安装的位置。

---



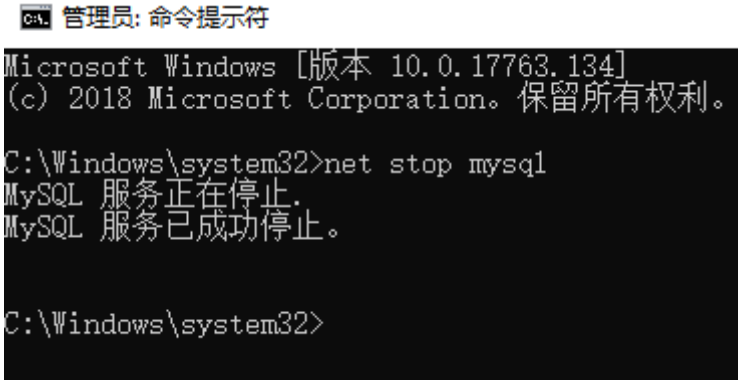

## 2.2 MySQL卸载

如果你想卸载MySQL，也很简单。

右键开始菜单，选择`命令提示符(管理员)`，打开黑框。

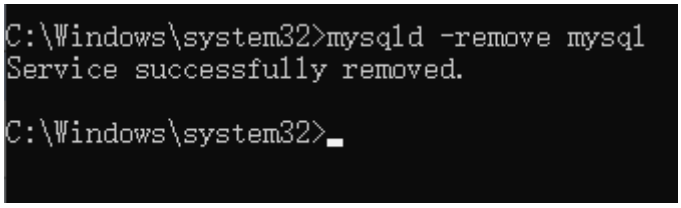
1. 敲入`net stop mysql`，回车。

```
net stop mysql
```




2. 再敲入`mysqld -remove mysql`，回车。

```
mysqld -remove mysql
```




3. 最后删除MySQL目录及相关的环境变量。

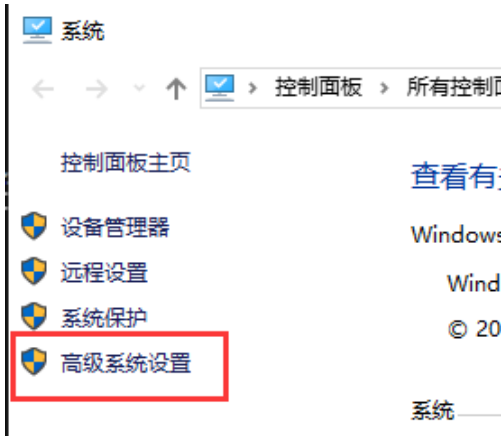
**至此，MySQL卸载完成！**

# 2.3 MySQL配置

## 2.3.1 添加环境变量

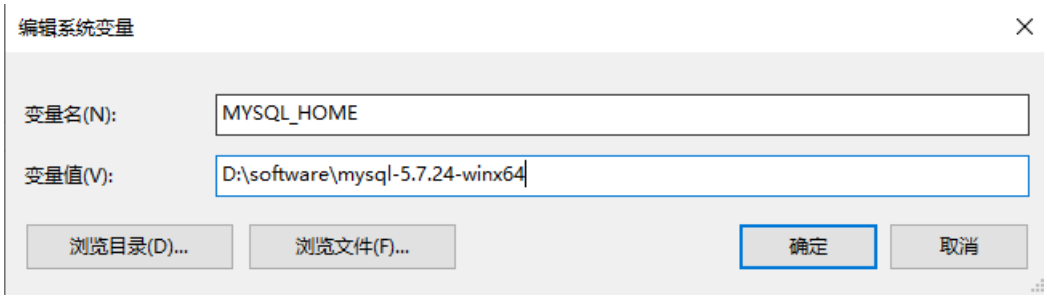
> 环境变量里面有很多选项，这里我们只用到`Path`这个参数。为什么在初始化的开始要添加环境变量呢？
> 在黑框(即CMD)中输入一个可执行程序的名字，Windows会先在环境变量中的`Path`所指的路径中寻找一遍，如果找到了就直接执行，没找到就在当前工作目录找，如果还没找到，就报错。我们添加环境变量的目的就是能够在任意一个黑框直接调用MySQL中的相关程序而不用总是修改工作目录，大大简化了操作。

右键`此电脑`→`属性`，点击`高级系统设置`

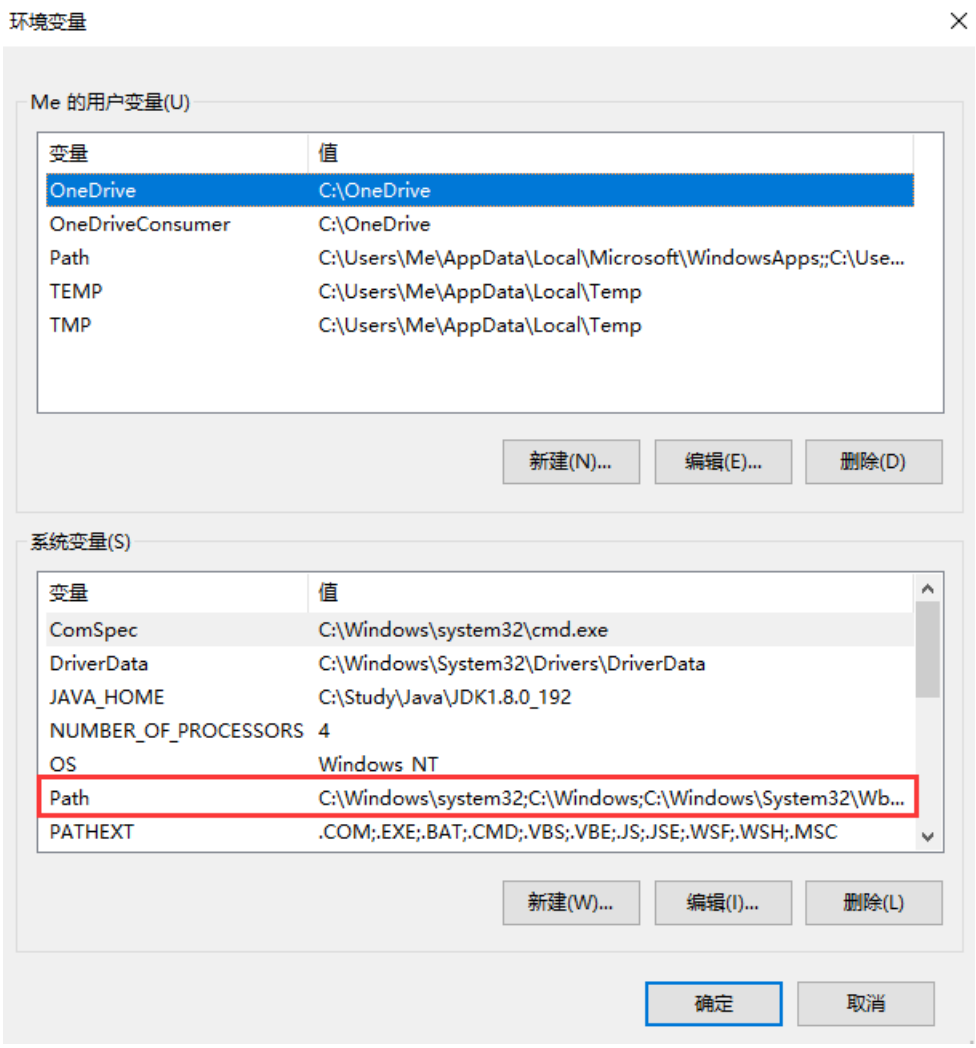



点击`环境变量`

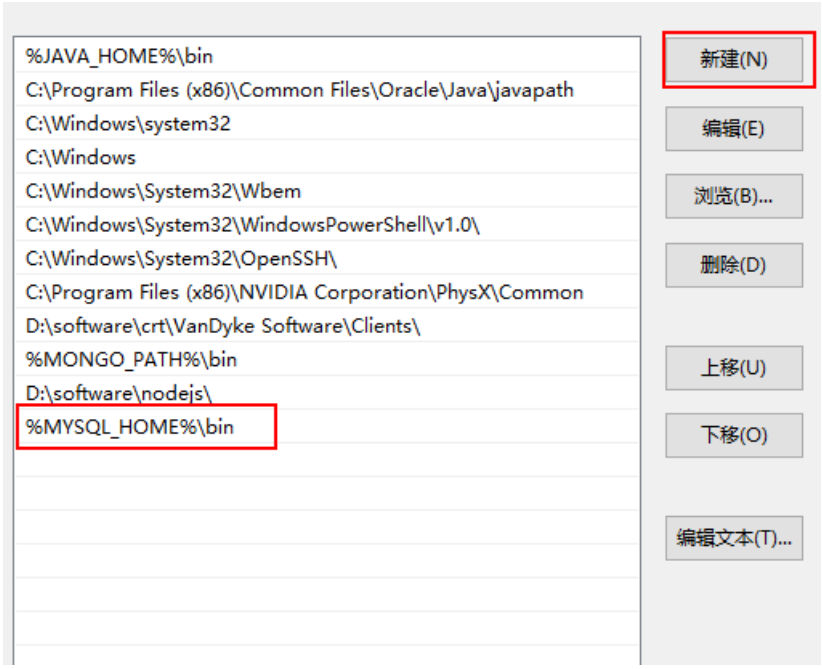

在`系统变量`中新建MYSQL_HOME




在`系统变量`中找到并**双击**`Path`




点击 新建




最后点击确定。

**如何验证是否添加成功?**

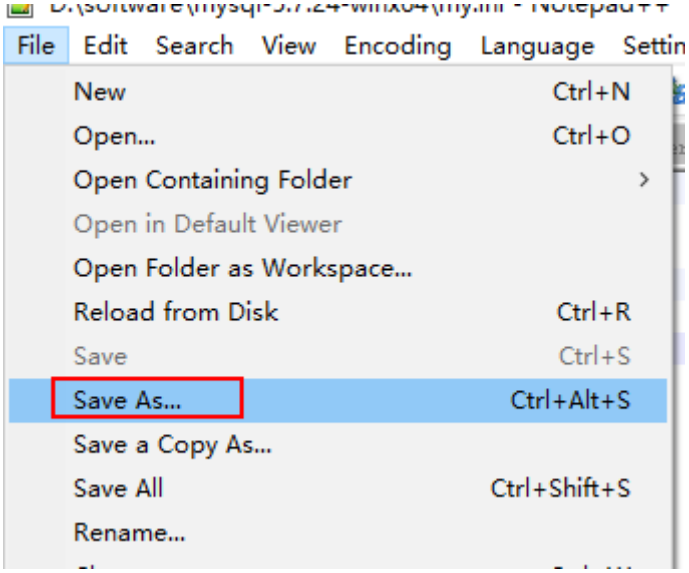
右键开始菜单(就是屏幕左下角)，选择`命令提示符(管理员)`，打开黑框，敲入`mysql`，回车。
如果提示`Can't connect to MySQL server on 'localhost'`则证明添加成功；
如果提示`mysql不是内部或外部命令，也不是可运行的程序或批处理文件`则表示添加添加失败，请重新检查步骤并重试。

### 2.3.2 新建配置文件

新建一个文本文件，内容如下：

```
[mysql]
default-character-set=utf8

[mysqld]
character-set-server=utf8
default-storage-engine=INNODB
sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_S
UBSTITUTION
```

把上面的文本文件另存为，在保存类型里选`所有文件 (*.*)`，文件名叫`my.ini`，存放的路径为MySQL的`根目录`(例如我的是`D:\software\mysql-5.7.24-winx64`,根据自己的MySQL目录位置修改)。



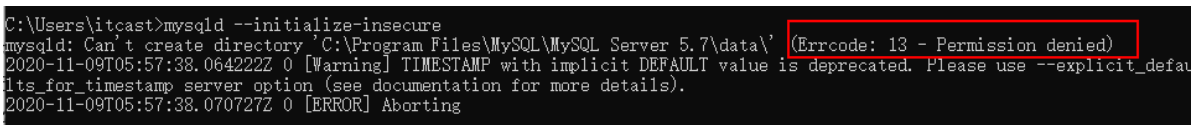

上面代码意思就是配置数据库的默认编码集为utf-8和默认存储引擎为INNODB。

### 2.3.3 初始化MySQL

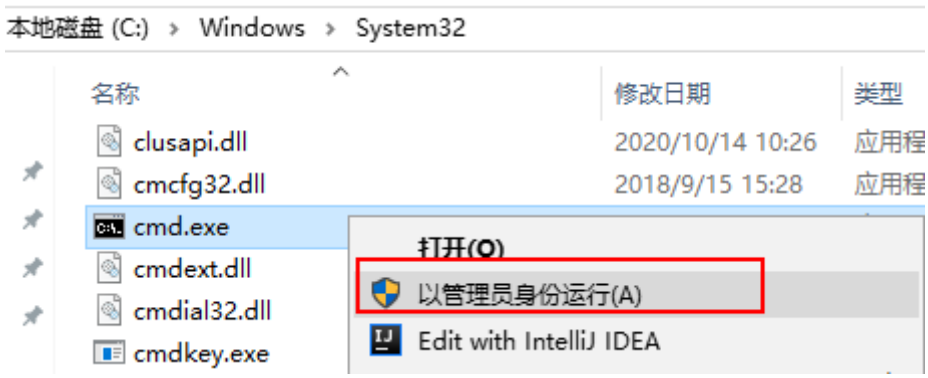
在刚才的黑框中敲入`mysqld --initialize-insecure`，回车，稍微等待一会，如果出现没有出现报错信息(如下图)则证明data目录初始化没有问题，此时再查看MySQL目录下已经有data目录生成。

```
mysqld --initialize-insecure
```


tips：如果出现如下错误



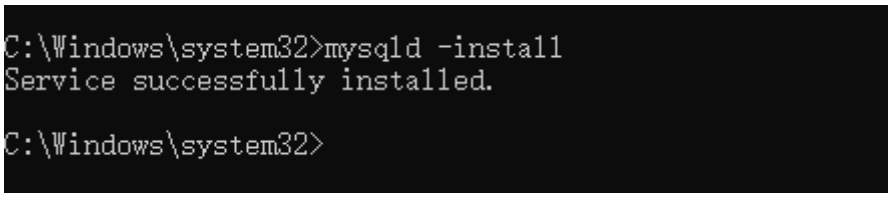

是由于权限不足导致的，去`C:\Windows\System32`下以管理员方式运行cmd.exe




## 2.3.4 注册MySQL服务

在黑框里敲入`mysqld -install`，回车。

```
mysqld -install
```



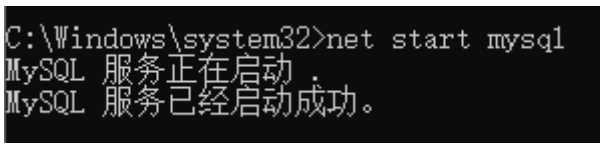

现在你的计算机上已经安装好了MySQL服务了。

MySQL服务器

## 2.3.5 启动MySQL服务

在黑框里敲入`net start mysql`，回车。

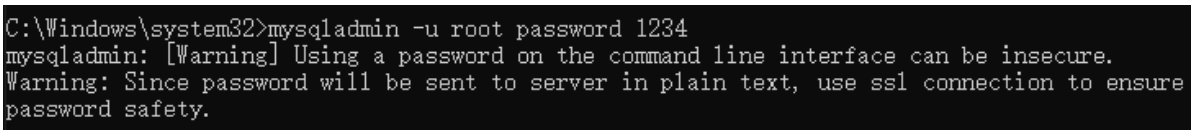
```
net start mysql  // 启动mysql服务

net stop mysql  // 停止mysql服务
```



### 2.3.6 修改默认账户密码

在黑框里敲入`mysqladmin -u root password 1234`，这里的`1234`就是指默认管理员(即root账户)的密码，可以自行修改成你喜欢的。

```
mysqladmin -u root password 1234
```




**至此，MySQL 5.7 解压版安装完毕!**

## 2.4 MySQL登陆和退出

### 2.4.1 登陆

右键开始菜单，选择`命令提示符`，打开黑框。
在黑框中输入，`mysql -uroot -p1234`，回车，出现下图且左下角为`mysql>`，则登录成功。

```
mysql -uroot -p1234
```


**到这里你就可以开始你的MySQL之旅了!**

登陆参数:

```
mysql -u用户名 -p密码 -h要连接的mysql服务器的ip地址(默认
127.0.0.1) -P端口号(默认3306)
```

### 2.4.2 退出

退出mysql:

```
exit
quit
```

## 2.5 MySQL数据模型

**关系型数据库:**

> 关系型数据库是建立在关系模型基础上的数据库，简单说，关系型数据库是由多张能互相连接的 二维表 组成的数据库

如下图，`订单信息表` 和 `客户信息表` 都是有行有列二维表我们将这样的称为关系型数据库。

订单信息表

| 订单编号 | 订单项目 | 负责人 | 业务员 | 订单数量 | 客户编号 |
|---|---|---|---|---|---|
| 001 | 挖掘机 | 刘明 | 李东明 | 1台 | 1 |
| 002 | 冲击钻 | 李刚 | 霍新峰 | 8个 | 2 |
| 003 | 铲车 | 郭新一 | 艾美丽 | 2辆 | 1 |

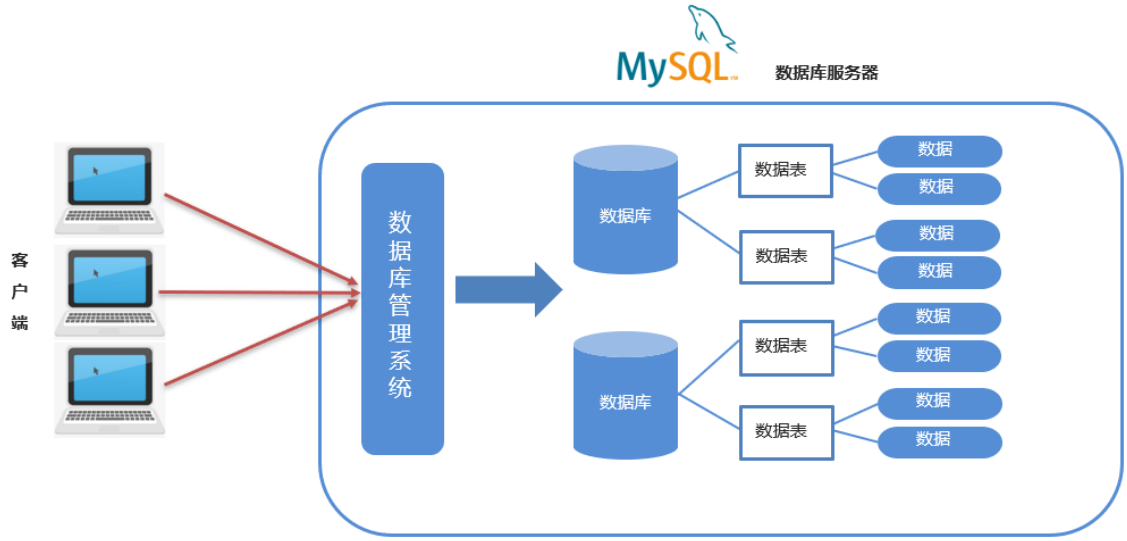
客户信息表

| 客户编号 | 客户名称 | 所属公司 | 联系方式 |
|---|---|---|---|
| 1 | 李聪 | 五一建设 | 13253661015 |
| 2 | 刘新明 | 个体经营 | 13285746958 |

接下来看关系型数据库的优点:

- 都是使用表结构，格式一致，易于维护。
- 使用通用的 SQL 语言操作，使用方便，可用于复杂查询。
  - 关系型数据库都可以通过SQL进行操作，所以使用方便。
  - 复杂查询。现在需要查询001号订单数据，我们可以看到该订单是1号客户的订单，而1号订单是李聪这个客户。以后也可以在一张表中进行统计分析等操作。
- 数据存储在磁盘中，安全。

**数据模型:**

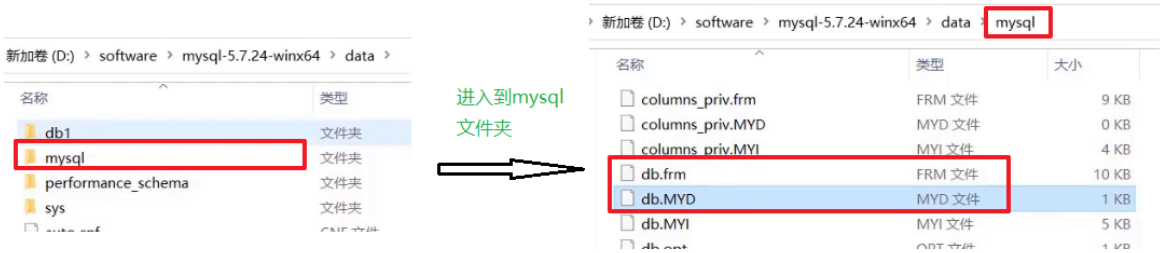



如上图，我们通过客户端可以通过数据库管理系统创建数据库，在数据库中创建表，在表中添加数据。创建的每一个数据库对应到磁盘上都是一个文件夹。比如可以通过SQL语句创建一个数据库（数据库名称为db1），语句如下。该语句咱们后面会学习。

```
mysql> create database db1;
```

我们可以在数据库安装目录下的data目录下看到多了一个 `db1` 的文件夹。所以，**在MySQL中一个数据库对应到磁盘上的一个文件夹。**

而一个数据库下可以创建多张表，我们到MySQL中自带的mysql数据库的文件夹目录下：




而上图中右边的 `db.frm` 是表文件，`db.MYD` 是数据文件，通过这两个文件就可以查询到数据展示成二维表的效果。

**小结：**

- MySQL中可以创建多个数据库，每个数据库对应到磁盘上的一个文件夹
- 在每个数据库中可以创建多个表，每张都对应到磁盘上一个 frm 文件
- 每张表可以存储多条数据，数据会被存储到磁盘中 MYD 文件中

# 3，SQL概述

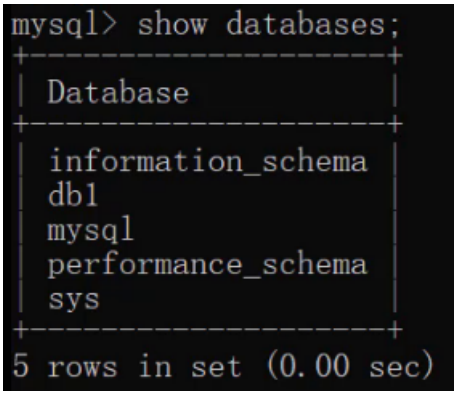
了解了数据模型后，接下来我们就学习SQL语句，通过SQL语句对数据库、表、数据进行增删改查操作。

## 3.1 SQL简介

- 英文：Structured Query Language，简称 SQL
- 结构化查询语言，一门操作关系型数据库的编程语言
- 定义操作所有关系型数据库的统一标准
- 对于同一个需求，每一种数据库操作的方式可能会存在一些不一样的地方，我们称为“方言”

## 3.2 通用语法

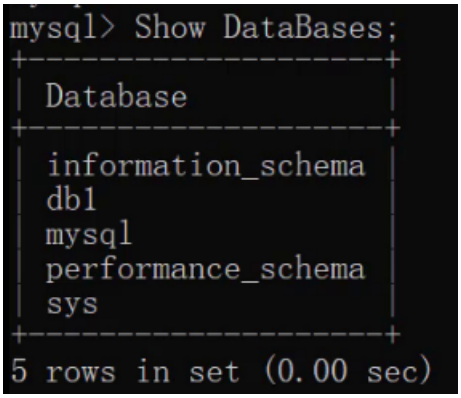
- SQL 语句可以单行或多行书写，以分号结尾。

  如上，以分号结尾才是一个完整的sql语句。

- MySQL 数据库的 SQL 语句不区分大小写，关键字建议使用大写。

  同样的一条sql语句写成下图的样子，一样可以运行处结果。

- 注释

- 单行注释: -- 注释内容 或 #注释内容(MySQL 特有)

```
mysql> Show DataBases;-- 查询所有数据库名称
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
mysql> show databases; #查询所有数据库名称
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

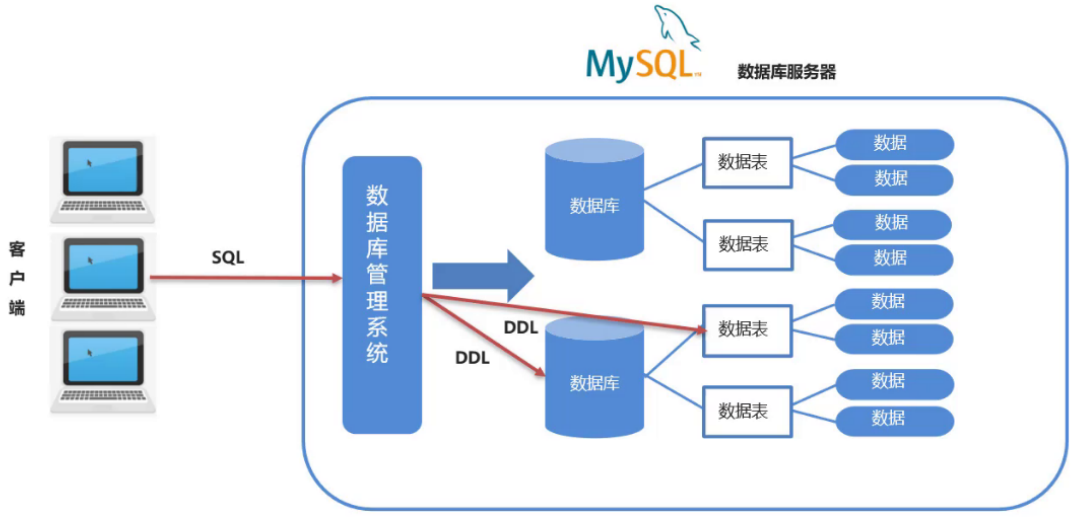
> 注意：使用-- 添加单行注释时，--后面一定要加空格，而#没有要求。

- 多行注释: /* 注释 */

## 3.3 SQL分类

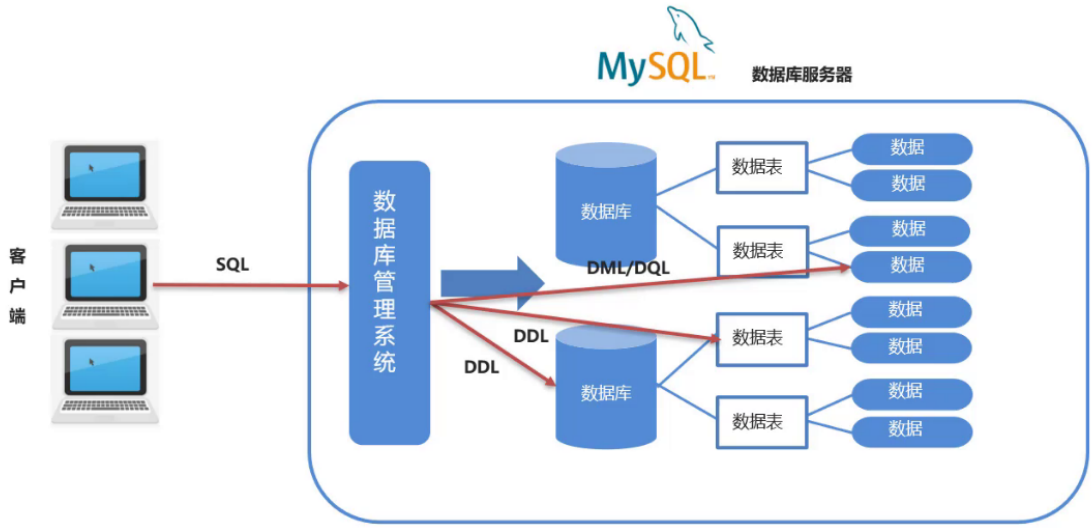
- DDL(Data Definition Language)： 数据定义语言，用来定义数据库对象：数据库，表，列等

  DDL简单理解就是用来操作数据库，表等



- DML(Data Manipulation Language) 数据操作语言，用来对数据库中表的数据进行增删改

  DML简单理解就对表中数据进行增删改




- DQL(Data Query Language) 数据查询语言，用来查询数据库中表的记录(数据)

  DQL简单理解就是对数据进行查询操作。从数据库表中查询到我们想要的数据。

- DCL(Data Control Language) 数据控制语言，用来定义数据库的访问权限和安全级别，及创建用户

  DML简单理解就是对数据库进行权限控制。比如我让某一个数据库表只能让某一个用户进行操作等。

> 注意： 以后我们最常操作的是 `DML` 和 `DQL` ，因为我们开发中最常操作的就是数据。

# 4，DDL:操作数据库

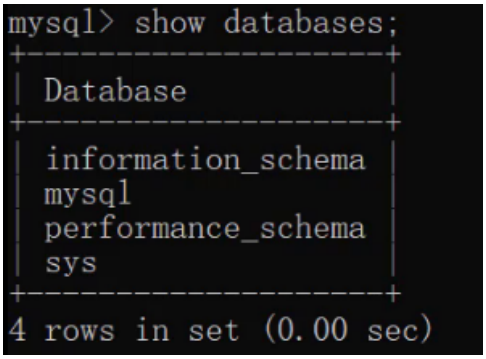
我们先来学习DDL来操作数据库。而操作数据库主要就是对数据库的增删查操作。

## 4.1 查询

查询所有的数据库

```
SHOW DATABASES;
```

运行上面语句效果如下：



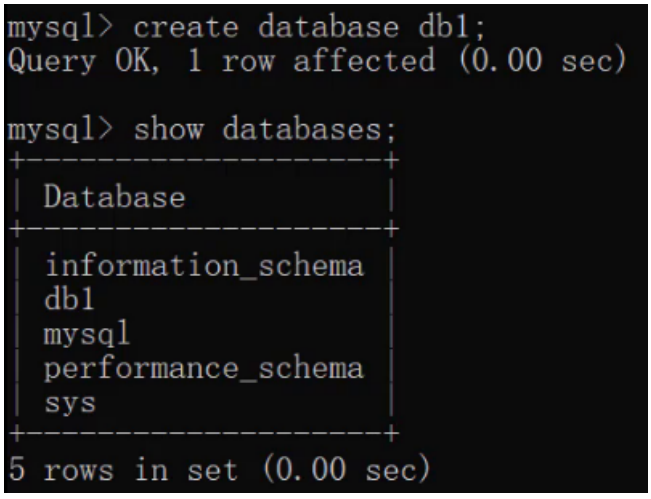

上述查询到的是的这些数据库是mysql安装好自带的数据库，我们以后不要操作这些数据库。

## 4.2 创建数据库

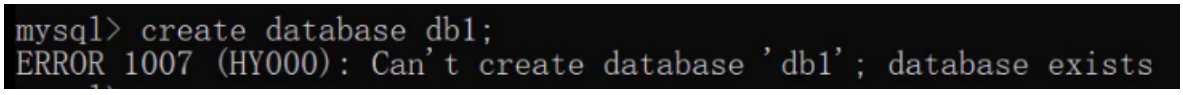
- **创建数据库**:

```
CREATE DATABASE 数据库名称;
```

运行语句效果如下:




而在创建数据库的时候，我并不知道db1数据库有没有创建，直接再次创建名为db1的数据库就会出现错误。

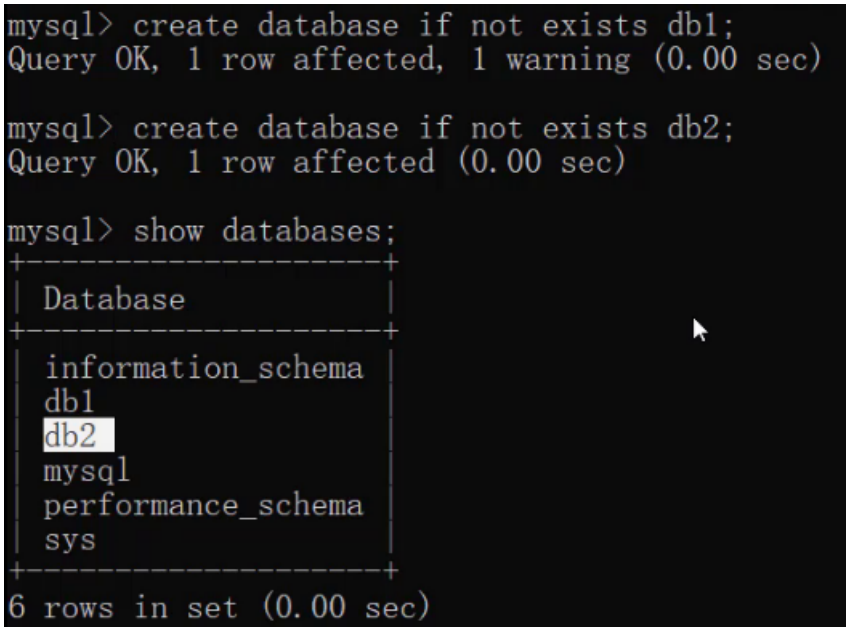



为了避免上面的错误，在创建数据库的时候先做判断，如果不存在再创建。

- **创建数据库(判断，如果不存在则创建)**

```
CREATE DATABASE IF NOT EXISTS 数据库名称;
```

运行语句效果如下：




从上面的效果可以看到虽然db1数据库已经存在，再创建db1也没有报错，而创建db2数据库则创建成功。

## 4.3 删除数据库

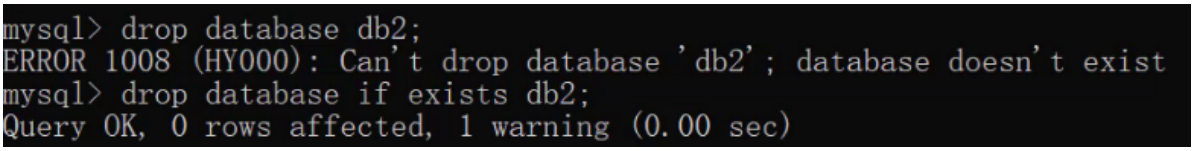
- **删除数据库**

```
DROP DATABASE 数据库名称;
```

- **删除数据库(判断，如果存在则删除)**

```
DROP DATABASE IF EXISTS 数据库名称;
```

运行语句效果如下：




## 4.4 使用数据库

数据库创建好了，要在数据库中创建表，得先明确在哪儿个数据库中操作，此时就需要使用数据库。

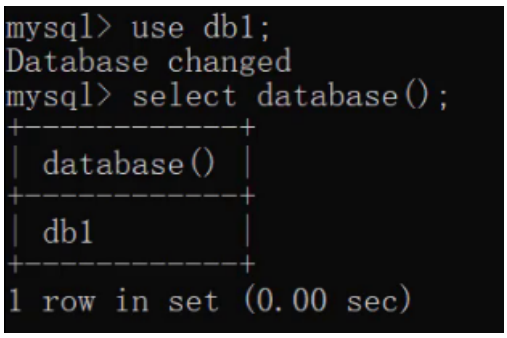
- **使用数据库**

```
USE 数据库名称;
```

- **查看当前使用的数据库**

```
SELECT DATABASE();
```

运行语句效果如下：




# 5，DDL:操作表

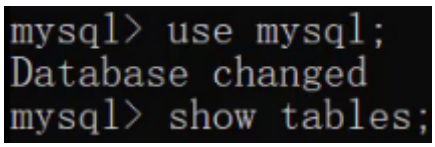
操作表也就是对表进行增（Create）删（Retrieve）改（Update）查（Delete）。

## 5.1 查询表

- **查询当前数据库下所有表名称**

```
SHOW TABLES;
```

我们创建的数据库中没有任何表，因此我们进入mysql自带的mysql数据库，执行上述语句查看




- **查询表结构**

```
DESC 表名称;
```

查看mysql数据库中func表的结构，运行语句如下：

```
mysql> desc func;
+-------+---------------------------------+------+-----+---------+-------+
| Field | Type                            | Null | Key | Default | Extra |
+-------+---------------------------------+------+-----+---------+-------+
| name  | char(64)                        | NO   | PRI |         |       |
| ret   | tinyint(1)                      | NO   |     | 0       |       |
| dl    | char(128)                       | NO   |     |         |       |
| type  | enum('function','aggregate')    | NO   |     | NULL    |       |
+-------+---------------------------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

## 5.2 创建表

- **创建表**

```
CREATE TABLE 表名 (
    字段名1  数据类型1,
    字段名2  数据类型2,
    …
    字段名n  数据类型n
);
```

> 注意：最后一行末尾，不能加逗号

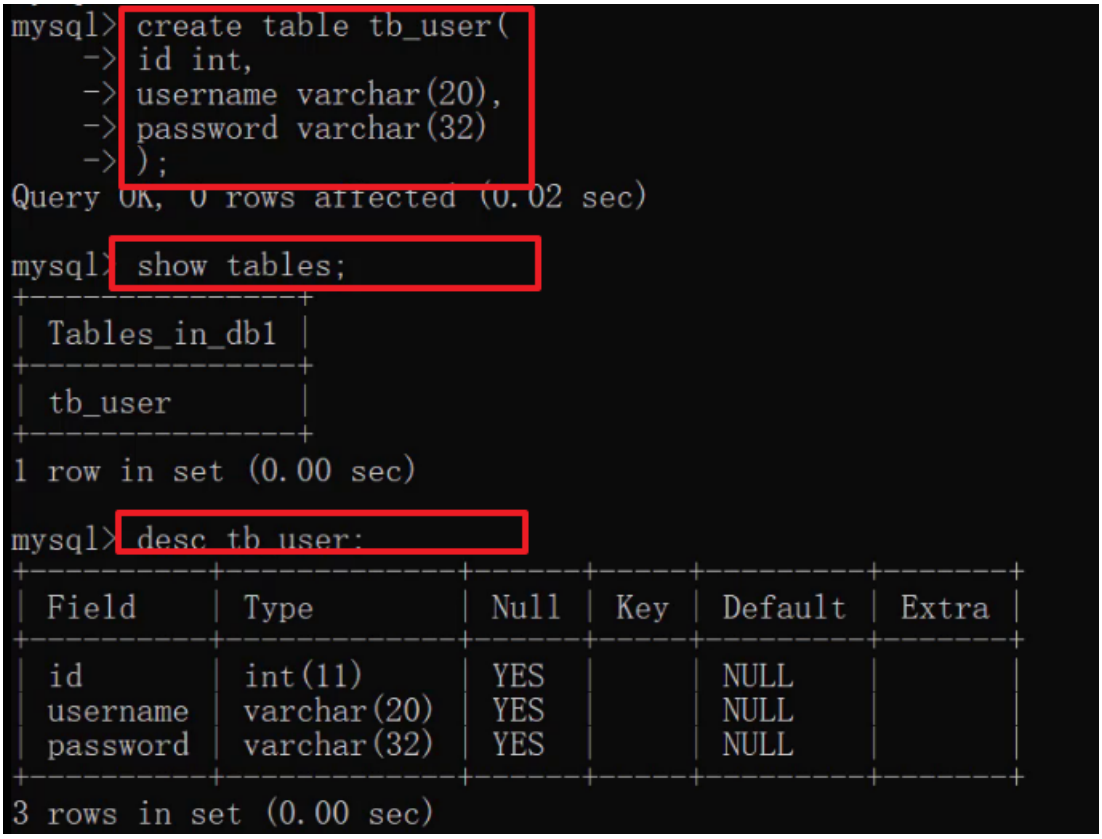
知道了创建表的语句，那么我们创建创建如下结构的表

tb_user

| id | username | password |
|---|---|---|
| | | |
| | | |
| | | |

```
create table tb_user (
    id int,
    username varchar(20),
    password varchar(32)
);
```

运行语句如下：




## 5.3 数据类型

MySQL 支持多种类型，可以分为三类：

- 数值

```
tinyint : 小整数型，占一个字节
int : 大整数类型，占四个字节
    eg : age int
double : 浮点类型
    使用格式: 字段名 double(总长度,小数点后保留的位数)
    eg : score double(5,2)
```

- 日期

```
date : 日期值。只包含年月日
    eg : birthday date :
datetime : 混合日期和时间值。包含年月日时分秒
```

- 字符串

```
char ：  定长字符串。
    优点：存储性能高
    缺点：浪费空间
    eg ：  name char(10)  如果存储的数据字符个数不足10个，也会占10个的空间
varchar ：  变长字符串。
    优点：节约空间
    缺点：存储性能底
    eg ：  name varchar(10) 如果存储的数据字符个数不足10个，那就数据字符个数是几就占几个的空间
```

> 注意：其他类型参考资料中的《MySQL数据类型].xlsx》

**案例：**

```
需求：设计一张学生表，请注重数据类型、长度的合理性
    1. 编号
    2. 姓名，姓名最长不超过10个汉字
    3. 性别，因为取值只有两种可能，因此最多一个汉字
    4. 生日，取值为年月日
    5. 入学成绩，小数点后保留两位
    6. 邮件地址，最大长度不超过 64
    7. 家庭联系电话，不一定是手机号码，可能会出现 - 等字符
    8. 学生状态（用数字表示，正常、休学、毕业...）
```

语句设计如下：

```sql
create table student (
    id int,
    name varchar(10),
    gender char(1),
    birthday date,
    score double(5,2),
    email varchar(15),
    tel varchar(15),
    status tinyint
);
```

## 5.4 删除表

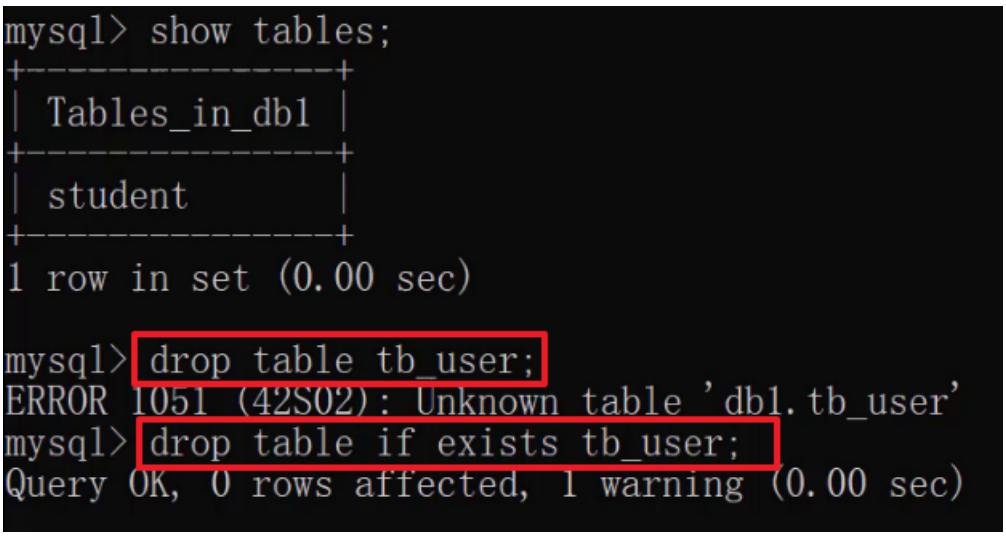
- **删除表**

```
DROP TABLE 表名;
```

- **删除表时判断表是否存在**

```
DROP TABLE IF EXISTS 表名;
```

运行语句效果如下：




## 5.5 修改表

- **修改表名**

```
ALTER TABLE 表名 RENAME TO 新的表名;

-- 将表名student修改为stu
alter table student rename to stu;
```

- **添加一列**

```
ALTER TABLE 表名 ADD 列名 数据类型;

-- 给stu表添加一列address，该字段类型是varchar(50)
alter table stu add address varchar(50);
```

- **修改数据类型**

```
ALTER TABLE 表名 MODIFY 列名 新数据类型;

-- 将stu表中的address字段的类型改为 char(50)
alter table stu modify address char(50);
```

- **修改列名和数据类型**

```
ALTER TABLE 表名 CHANGE 列名 新列名 新数据类型;

-- 将stu表中的address字段名改为 addr，类型改为varchar(50)
alter table stu change address addr varchar(50);
```

- **删除列**

```
ALTER TABLE 表名 DROP 列名;

-- 将stu表中的addr字段 删除
alter table stu drop addr;
```

# 6，navicat使用

通过上面的学习，我们发现在命令行中写sql语句特别不方便，尤其是编写创建表的语句，我们只能在记事本上写好后直接复制到命令行进行执行。那么有没有刚好的工具提供给我们进行使用呢？ 有。

## 6.1 navicat概述

- Navicat for MySQL 是管理和开发 MySQL 或 MariaDB 的理想解决方案。
- 这套全面的前端工具为数据库管理、开发和维护提供了一款直观而强大的图形界面。
- 官网： http://www.navicat.com.cn

# 6.2 navicat安装

参考：资料\navicat安装包\navicat_mysql_x86\navicat安装步骤.md

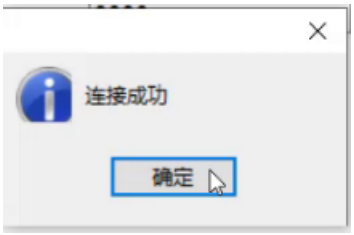
# 6.3 navicat使用

## 6.3.1 建立和mysql服务的连接

第一步： 点击连接，选择MySQL


第二步：填写连接数据库必要的信息


以上操作没有问题就会出现如下图所示界面：

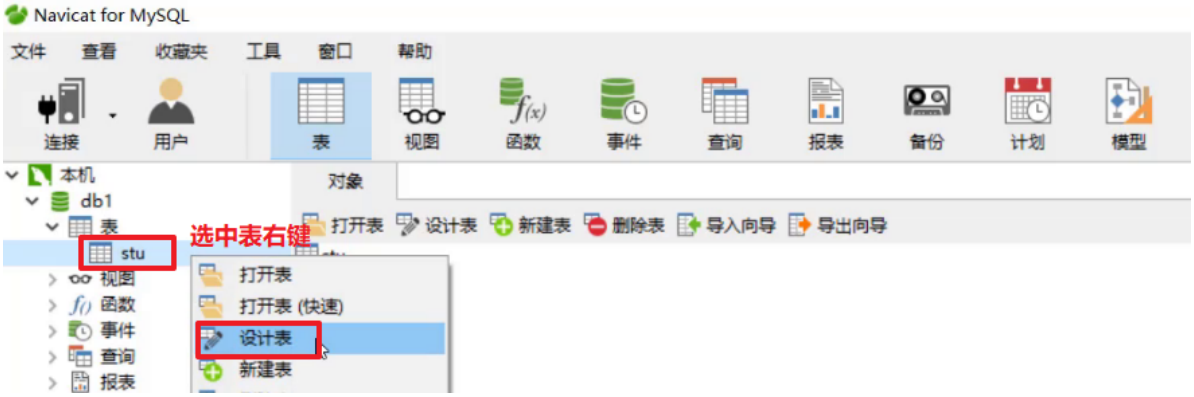



## 6.3.2 操作

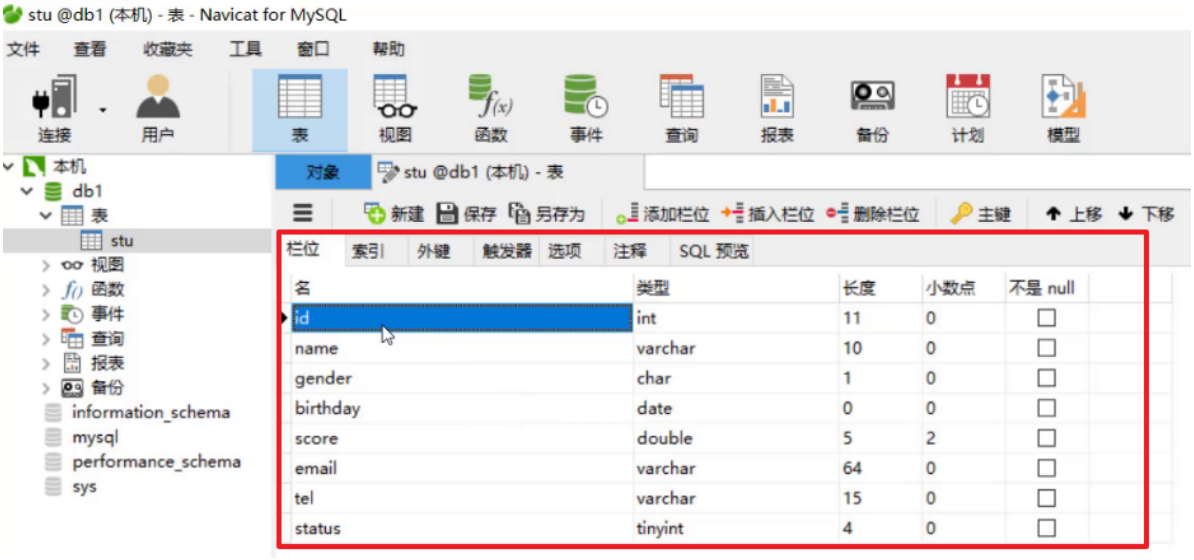
连接成功后就能看到如下图界面：


- **修改表结构**

通过下图操作修改表结构：

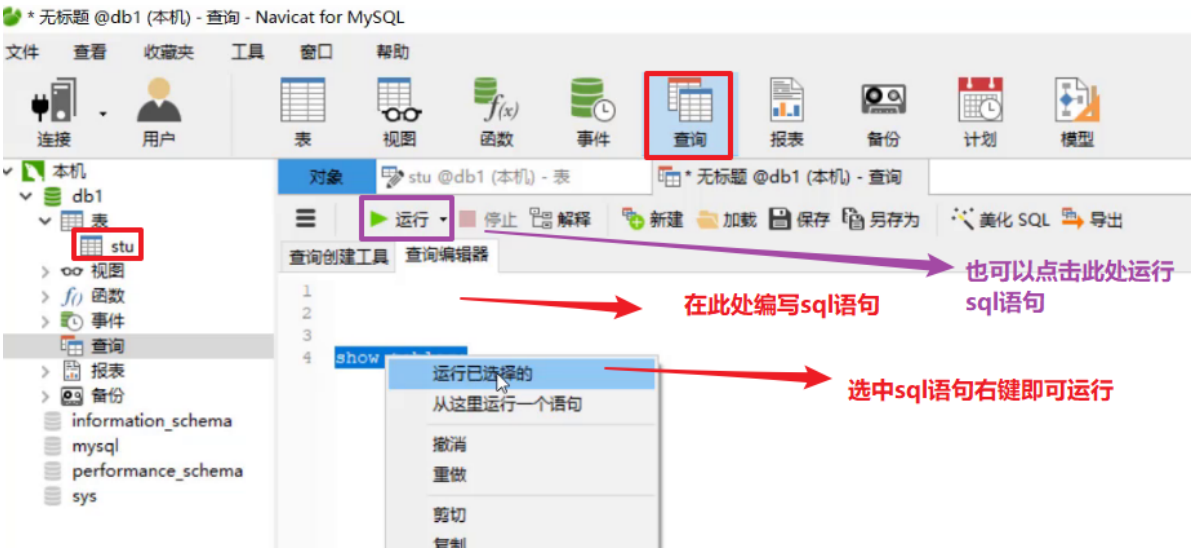



点击了设计表后即出现如下图所示界面，在图中红框中直接修改字段名，类型等信息：




- **编写SQL语句并执行**

按照如下图所示进行操作即可书写SQL语句并执行sql语句。




# 7，DML

DML主要是对数据进行增（insert）删（delete）改（update）操作。

## 7.1 添加数据

- **给指定列添加数据**

```
INSERT INTO 表名(列名1,列名2,…) VALUES(值1,值2,…);
```

- **给全部列添加数据**

```
INSERT INTO 表名 VALUES(值1,值2,…);
```

- **批量添加数据**

```
INSERT INTO 表名(列名1,列名2,…) VALUES(值1,值2,…),(值1,值2,…),
(值1,值2,…)…;
INSERT INTO 表名 VALUES(值1,值2,…),(值1,值2,…),(值1,值2,…)…;
```

- **练习**

为了演示以下的增删改操作是否操作成功，故先将查询所有数据的语句介绍给大家：

```
select * from stu;
```

```
-- 给指定列添加数据
INSERT INTO stu (id, NAME) VALUES (1, '张三');
-- 给所有列添加数据，列名的列表可以省略的
INSERT INTO stu
(id,NAME,sex,birthday,score,email,tel,STATUS) VALUES
(2,'李四','男','1999-11-
11',88.88,'lisi@itcast.cn','13888888888',1);

INSERT INTO stu VALUES (2,'李四','男','1999-11-
11',88.88,'lisi@itcast.cn','13888888888',1);

-- 批量添加数据
INSERT INTO stu VALUES
    (2,'李四','男','1999-11-
11',88.88,'lisi@itcast.cn','13888888888',1),
    (2,'李四','男','1999-11-
11',88.88,'lisi@itcast.cn','13888888888',1),
    (2,'李四','男','1999-11-
11',88.88,'lisi@itcast.cn','13888888888',1);
```

# 7.2 修改数据

- **修改表数据**

```
UPDATE 表名 SET 列名1=值1,列名2=值2,… [WHERE 条件] ;
```

> 注意：
>
> 1. 修改语句中如果不加条件，则将所有数据都修改！
> 2. 像上面的语句中的中括号，表示在写sql语句中可以省略这部分

- **练习**
  - 将张三的性别改为女

    ```
    update stu set sex = '女' where name = '张三';
    ```

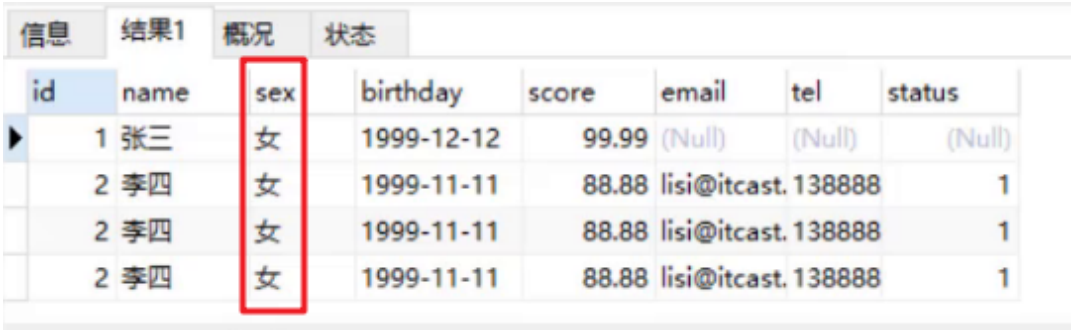
  - 将张三的生日改为 1999-12-12 分数改为99.99

    ```
    update stu set birthday = '1999-12-12', score =
    99.99 where name = '张三';
    ```

  - 注意：如果update语句没有加where条件，则会将表中所有数据全部修改！

    ```
    update stu set sex = '女';
    ```

    上面语句的执行完后查询到的结果是：




| id | name | sex | birthday | score | email | tel | status |
|---|---|---|---|---|---|---|---|
| 1 | 张三 | 女 | 1999-12-12 | 99.99 | (Null) | (Null) | (Null) |
| 2 | 李四 | 女 | 1999-11-11 | 88.88 | lisi@itcast. | 138888 | 1 |
| 2 | 李四 | 女 | 1999-11-11 | 88.88 | lisi@itcast. | 138888 | 1 |
| 2 | 李四 | 女 | 1999-11-11 | 88.88 | lisi@itcast. | 138888 | 1 |

## 7.3 删除数据

- **删除数据**

```
DELETE FROM 表名 [WHERE 条件] ;
```

- **练习**

```
-- 删除张三记录
delete from stu where name = '张三';

-- 删除stu表中所有的数据
delete from stu;
```

# 8，DQL

当然上图中的难度字段当我们点击也可以实现排序查询操作。从这个例子我们就可以看出，对于数据库的查询时灵活多变的，需要根据具体的需求来实现，而数据库查询操作也是最重要的操作，所以此部分需要大家重点掌握。

接下来我们先介绍查询的完整语法：

```
SELECT
    字段列表
FROM
    表名列表
WHERE
    条件列表
GROUP BY
    分组字段
HAVING
    分组后条件
ORDER BY
    排序字段
LIMIT
    分页限定
```

为了给大家演示查询的语句，我们需要先准备表及一些数据：

```
-- 删除stu表
drop table if exists stu;


-- 创建stu表
CREATE TABLE stu (
 id int, -- 编号
 name varchar(20), -- 姓名
 age int, -- 年龄
 sex varchar(5), -- 性别
 address varchar(100), -- 地址
 math double(5,2), -- 数学成绩
 english double(5,2), -- 英语成绩
 hire_date date -- 入学时间
);

-- 添加数据
INSERT INTO
stu(id,NAME,age,sex,address,math,english,hire_date)
VALUES
(1,'马运',55,'男','杭州',66,78,'1995-09-01'),
(2,'马花疼',45,'女','深圳',98,87,'1998-09-01'),
(3,'马斯克',55,'男','香港',56,77,'1999-09-02'),
(4,'柳白',20,'女','湖南',76,65,'1997-09-05'),
(5,'柳青',20,'男','湖南',86,NULL,'1998-09-01'),
(6,'刘德花',57,'男','香港',99,99,'1998-09-01'),
(7,'张学右',22,'女','香港',99,99,'1998-09-01'),
(8,'德玛西亚',18,'男','南京',56,65,'1994-09-02');
```

接下来咱们从最基本的查询语句开始学起。

# 8.1 基础查询

## 8.1.1 语法

- **查询多个字段**

```
SELECT 字段列表 FROM 表名;
SELECT * FROM 表名; -- 查询所有数据
```

- **去除重复记录**

```
SELECT DISTINCT 字段列表 FROM 表名;
```

- **起别名**

```
AS: AS 也可以省略
```

## 8.1.2 练习

- 查询name、age两列

  ```
  select name,age from stu;
  ```

- 查询所有列的数据，列名的列表可以使用*替代

  ```
  select * from stu;
  ```

  上面语句中的*不建议大家使用，因为在这写*不方便我们阅读sql语句。我们写字段列表的话，可以添加注释对每一个字段进行说明

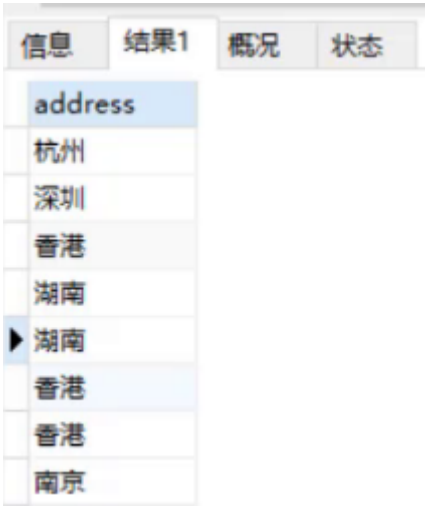
  ```
  SELECT
    NAME, -- 姓名
    age -- 年龄
  FROM
    stu;
  ```

  而在上课期间为了简约课程的时间，老师很多地方都会写*。

- 查询地址信息

  ```
  select address from stu;
  ```

  执行上面语句结果如下：




从上面的结果我们可以看到有重复的数据，我们也可以使用 `distinct` 关键字去重重复数据。

- 去除重复记录

```
select distinct address from stu;
```

- 查询姓名、数学成绩、英语成绩。并通过as给math和english起别名（as关键字可以省略）

```
select name,math as 数学成绩,english as 英文成绩 from stu;
select name,math 数学成绩,english 英文成绩 from stu;
```

# 8.2 条件查询

## 8.2.1 语法

```
SELECT 字段列表 FROM 表名 WHERE 条件列表;
```

- **条件**

条件列表可以使用以下运算符

| 符号 | 功能 |
| --- | --- |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| = | 等于 |
| <> 或 != | 不等于 |
| BETWEEN ... AND ... | 在某个范围之内(都包含) |
| IN(...) | 多选一 |
| LIKE 占位符 | 模糊查询 _单个任意字符 %多个任意字符 |
| IS NULL | 是NULL |
| IS NOT NULL | 不是NULL |
| AND 或 && | 并且 |
| OR 或 \|\| | 或者 |
| NOT 或 ! | 非，不是 |

## 8.2.2 条件查询练习

- 查询年龄大于20岁的学员信息

```
select * from stu where age > 20;
```

- 查询年龄大于等于20岁的学员信息

```
select * from stu where age >= 20;
```

- 查询年龄大于等于20岁 并且 年龄 小于等于 30岁 的学员信息

```
select * from stu where age >= 20 &&  age <= 30;
select * from stu where age >= 20 and  age <= 30;
```

> 上面语句中 && 和 and 都表示并且的意思。建议使用 and 。
>
> 也可以使用 between ... and 来实现上面需求

```
select * from stu where age BETWEEN 20 and 30;
```

- 查询入学日期在'1998-09-01' 到 '1999-09-01' 之间的学员信息

```
select * from stu where hire_date BETWEEN '1998-09-01'
and '1999-09-01';
```

- 查询年龄等于18岁的学员信息

```
select * from stu where age = 18;
```

- 查询年龄不等于18岁的学员信息

```
select * from stu where age != 18;
select * from stu where age <> 18;
```

- 查询年龄等于18岁 或者 年龄等于20岁 或者 年龄等于22岁的学员信息

```
select * from stu where age = 18 or age = 20 or age =
22;
select * from stu where age in (18,20 ,22);
```

- 查询英语成绩为 null的学员信息

  null值的比较不能使用 = 或者 != 。需要使用 is 或者 is not

```
select * from stu where english = null; -- 这个语句是不行
的
select * from stu where english is null;
select * from stu where english is not null;
```

### 8.2.3 模糊查询练习

> 模糊查询使用like关键字，可以使用通配符进行占位:
>
> (1) _ : 代表单个任意字符
>
> (2) % : 代表任意个数字符

- 查询姓'马'的学员信息

```
select * from stu where name like '马%';
```

- 查询第二个字是'花'的学员信息

```
select * from stu where name like '_花%';
```

- 查询名字中包含 '德' 的学员信息

```
select * from stu where name like '%德%';
```

## 8.3 排序查询

### 8.3.1 语法

```
SELECT 字段列表 FROM 表名 ORDER BY 排序字段名1 [排序方式1],排序字段名2 [排序方式2] …;
```

上述语句中的排序方式有两种，分别是：

- ASC： 升序排列 **(默认值)**
- DESC： 降序排列

> 注意：如果有多个排序条件，当前边的条件值一样时，才会根据第二条件进行排序

### 8.3.2 练习

- 查询学生信息，按照年龄升序排列

```
select * from stu order by age ;
```

- 查询学生信息，按照数学成绩降序排列

```
select * from stu order by math desc ;
```

- 查询学生信息，按照数学成绩降序排列，如果数学成绩一样，再按照英语成绩升序排列

```
select * from stu order by math desc , english asc ;
```

# 8.4 聚合函数

## 8.4.1 概念

==将一列数据作为一个整体，进行纵向计算。==

如何理解呢？假设有如下表

| id | name | age | sex | address | math | english | hire_date |
|---|---|---|---|---|---|---|---|
| 1 | 马运 | 55 | 男 | 杭州 | 66 | 78 | 1995-09-01 |
| 2 | 马花疼 | 45 | 女 | 深圳 | 98 | 87 | 1998-09-01 |
| 3 | 马斯克 | 55 | 男 | 香港 | 56 | 77 | 1999-09-02 |
| 4 | 柳白 | 20 | 女 | 湖南 | 76 | 65 | 1997-09-05 |
| 5 | 柳青 | 20 | 男 | 湖南 | 86 | (Null) | 1998-09-01 |
| 6 | 刘德花 | 57 | 男 | 香港 | 99 | 99 | 1998-09-01 |
| 7 | 张学右 | 22 | 女 | 香港 | 99 | 99 | 1998-09-01 |
| 8 | 德玛西亚 | 18 | 男 | 南京 | 56 | 65 | 1994-09-02 |

现有一需求让我们求表中所有数据的数学成绩的总和。这就是对math字段进行纵向求和。

## 8.4.2 聚合函数分类

| 函数名 | 功能 |
|---|---|
| count(列名) | 统计数量（一般选用不为null的列） |
| max(列名) | 最大值 |
| min(列名) | 最小值 |
| sum(列名) | 求和 |
| avg(列名) | 平均值 |

## 8.4.3 聚合函数语法

```
SELECT 聚合函数名(列名) FROM 表;
```

> 注意：null 值不参与所有聚合函数运算

### 8.4.4 练习

- 统计班级一共有多少个学生

```
select count(id) from stu;
select count(english) from stu;
```

上面语句根据某个字段进行统计，如果该字段某一行的值为null的话，将不会被统计。所以可以在count(*) 来实现。* 表示所有字段数据，一行中也不可能所有的数据都为null，所以建议使用 count(*)

```
select count(*) from stu;
```

- 查询数学成绩的最高分

```
select max(math) from stu;
```

- 查询数学成绩的最低分

```
select min(math) from stu;
```

- 查询数学成绩的总分

```
select sum(math) from stu;
```

- 查询数学成绩的平均分

```
select avg(math) from stu;
```

- 查询英语成绩的最低分

```
select min(english) from stu;
```

## 8.5 分组查询

### 8.5.1 语法

```
SELECT 字段列表 FROM 表名 [WHERE 分组前条件限定] GROUP BY 分组字段名 [HAVING 分组后条件过滤];
```

> 注意：分组之后，查询的字段为聚合函数和分组字段，查询其他字段无任何意义

### 8.5.2 练习

- 查询男同学和女同学各自的数学平均分

```
select sex, avg(math) from stu group by sex;
```

> 注意：分组之后，查询的字段为聚合函数和分组字段，查询其他字段无任何意义

```
select name, sex, avg(math) from stu group by sex;  -- 这里查询name字段就没有任何意义
```

- 查询男同学和女同学各自的数学平均分，以及各自人数

```
select sex, avg(math),count(*) from stu group by sex;
```

- 查询男同学和女同学各自的数学平均分，以及各自人数，要求：分数低于70分的不参与分组

```
select sex, avg(math),count(*) from stu where math > 70 group by sex;
```

- 查询男同学和女同学各自的数学平均分，以及各自人数，要求：分数低于70分的不参与分组，分组之后人数大于2个的

```
select sex, avg(math),count(*) from stu where math > 70 group by sex having count(*)  > 2;
```

**where 和 having 区别：**

- 执行时机不一样：where 是分组之前进行限定，不满足where条件，则不参与分组，而having是分组之后对结果进行过滤。
- 可判断的条件不一样：where 不能对聚合函数进行判断，having 可以。

## 8.6 分页查询

如下图所示，大家在很多网站都见过类似的效果，如京东、百度、淘宝等。分页查询是将数据一页一页的展示给用户看，用户也可以通过点击查看下一页的数据。


接下来我们先说分页查询的语法。

### 8.6.1 语法

```
SELECT 字段列表 FROM 表名 LIMIT  起始索引 , 查询条目数;
```

> 注意： 上述语句中的起始索引是从0开始

### 8.6.2 练习

- 从0开始查询，查询3条数据

```
select * from stu limit 0 , 3;
```

- 每页显示3条数据，查询第1页数据

```
select * from stu limit 0 , 3;
```

- 每页显示3条数据，查询第2页数据

```
select * from stu limit 3 , 3;
```

- 每页显示3条数据，查询第3页数据

```
select * from stu limit 6 , 3;
```

从上面的练习推导出起始索引计算公式：

```
起始索引 = (当前页码 - 1) * 每页显示的条数
```