第一章 数据库简介

1.1 简介

数据库（DataBase，DB）：指长期保存在计算机的存储设备上，按照一定规则组织起来，可以被各种用户或应用共享的数据集合。

数据库管理系统（DataBase Management System，DBMS）：指一种操作和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中的数据。

数据库软件应该为**数据库管理系统**，数据库是通过数据库管理系统创建和操作的。

数据库：存储、维护和管理数据的集合。

1.2 常见数据库管理系统

- Oracle：Oracle数据库被认为是业界目前比较成功的关系型数据库管理系统。Oracle数据库可以运行在UNIX、Windows等主流操作系统平台，完全支持所有的工业标准，并获得最高级别的ISO标准安全性认证。

- MySQL：MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

- DB2：DB2是IBM公司的产品，DB2数据库系统采用多进程多线索体系结构，其功能足以满足大中公司的需要，并可灵活地服务于中小型电子商务解决方案。

- Microsoft SQL Server：SQL Server 是Microsoft 公司推出的关系型数据库管理系统。具有使用方便可伸缩性好与相关软件集成程度高等优点。

1.3 三大范式（规范）

什么是三大范式：

第一范式：**无重复的列。**当关系模式R的所有属性都不能在分解为更基本的数据单位时，称R是满足第一范式的，简记为1NF。满足第一范式是关系模式规范化的最低要求，否则，将有很多基本操作在这样的关系模式中实现不了。

第二范式：**属性完全依赖于主键 [消除部分子函数依赖]。**如果关系模式R满足第一范式，并且R得所有非主属性都完全依赖于R的每一个候选关键属性，称R满足第二范式，简记为2NF。第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。**第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。**为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。这个唯一属性列被称为主关键字或主键、主码。

第三范式：**属性不依赖于其它非主属性 [消除传递依赖]。**设R是一个满足第一范式条件的关系模式，X是R的任意属性集，如果X非传递依赖于R的任意一个候选关键字，称R满足第三范式，简记为3NF. 满足第三范式（3NF）必须先满足第二范式（2NF）。**第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。**

注：关系实质上是一张二维表，其中每一行是一个元组，每一列是一个属性

第二范式（2NF）和第三范式（3NF）的概念很容易混淆，区分它们的关键点在于，2NF：非主键列是否完全依赖于主键，还是依赖于主键的一部分；3NF：非主键列是直接依赖于主键，还是直接依赖于非主键列。

1.4 MySQL安装和卸载

- 安装和配置步骤(详见下发的资料)

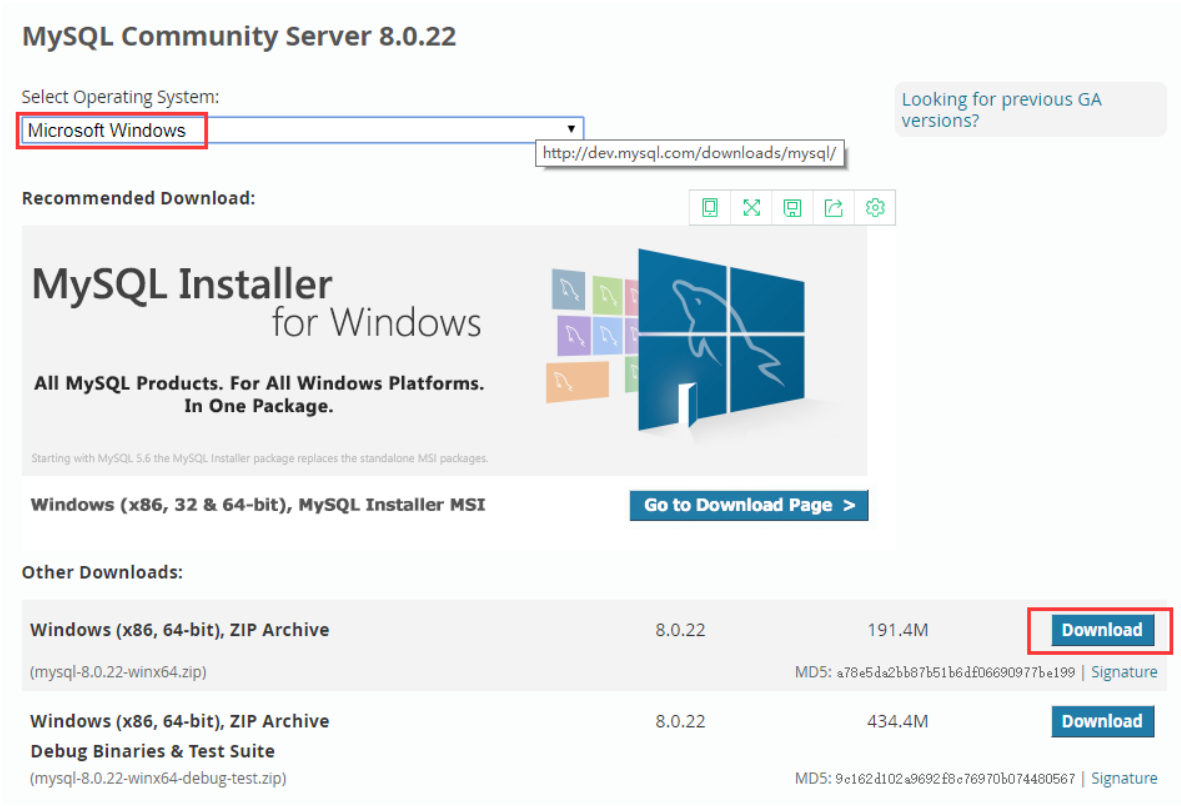
  使用版本:MYSQL8.0.22

1.4.1 安装

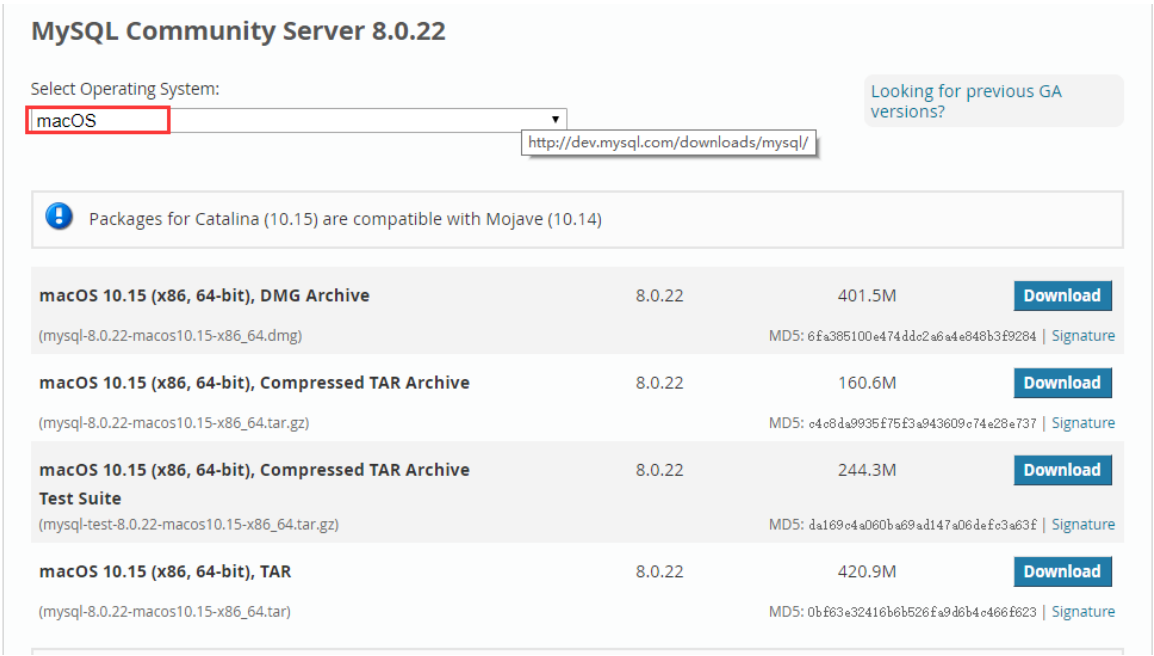
步骤1:访问地址:https://dev.mysql.com/downloads/mysql/

步骤2:下载压缩包

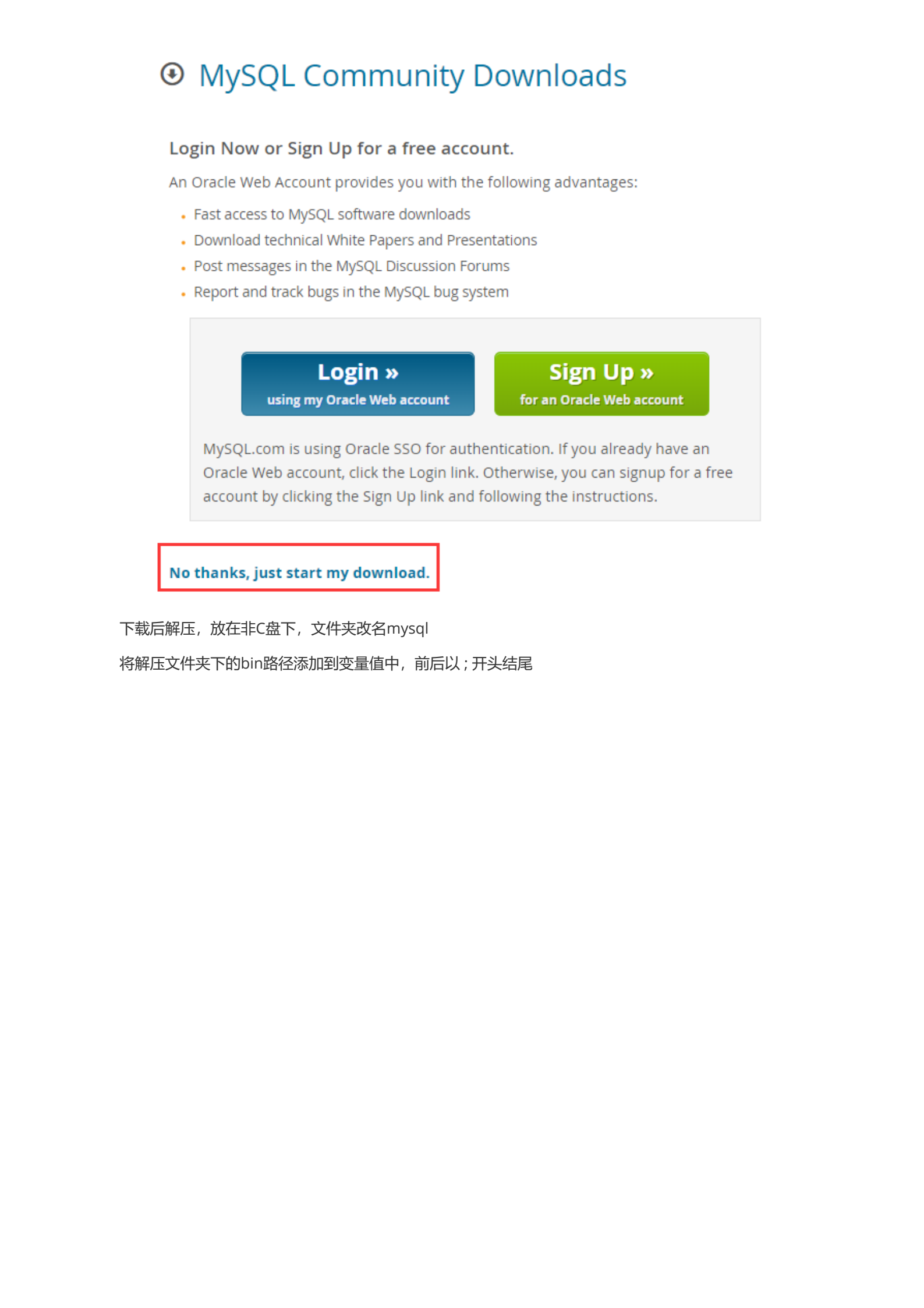
for window



for mac

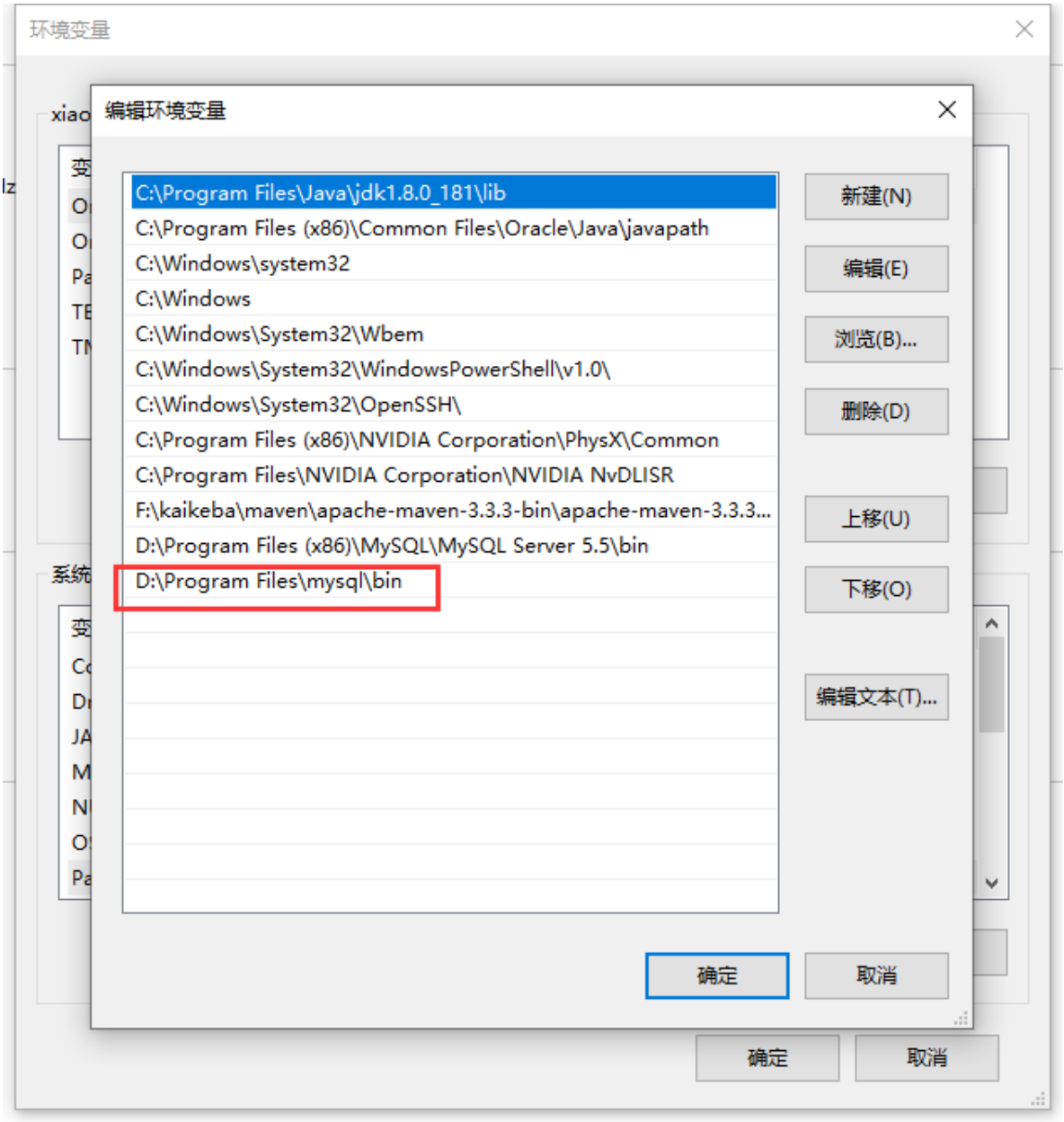


没有账户的点击左下方:No thanks。

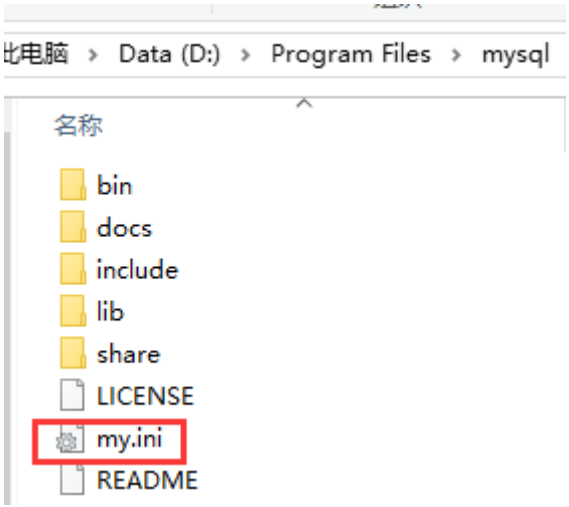


下载后解压，放在非C盘下，文件夹改名mysql

将解压文件夹下的bin路径添加到变量值中，前后以；开头结尾

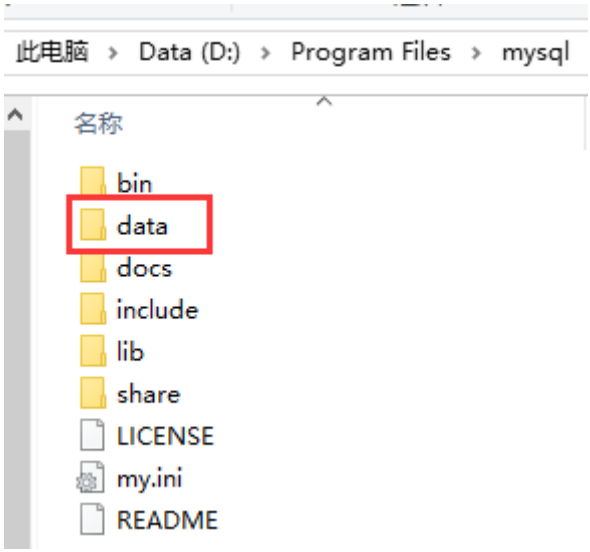


步骤3:在mysql文件夹下找到my.ini或my-default.ini，如果没有.ini结尾的文件,直接创建该文件。新增内容为如下，注意basedir和datadir是我自己的路径位置，自定义。记得新增一个文件Data文件夹




```
[mysqld]
# 设置3306端口
port=3306
# 设置mysql的安装目录
basedir=D:\Program Files\mysql
```

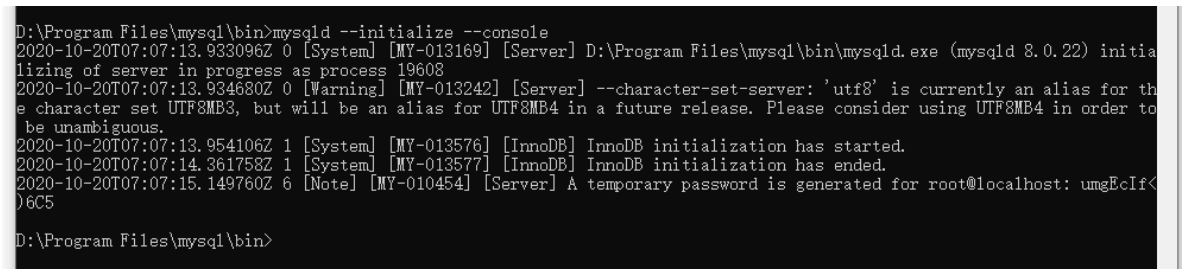
```
# 设置mysql数据库的数据的存放目录
datadir=D:\Program Files\mysql\data
# 允许最大连接数
max_connections=200
# 允许连接失败的次数。这是为了防止有人从该主机试图攻击数据库系统
max_connect_errors=10
# 服务端使用的字符集默认为UTF8
character-set-server=utf8
# 创建新表时将使用的默认存储引擎
default-storage-engine=INNODB
# 默认使用“mysql_native_password”插件认证
default_authentication_plugin=mysql_native_password
[mysql]
# 设置mysql客户端默认字符集
default-character-set=utf8
[client]
# 设置mysql客户端连接服务端时默认使用的端口
port=3306

default-character-set=utf8
```



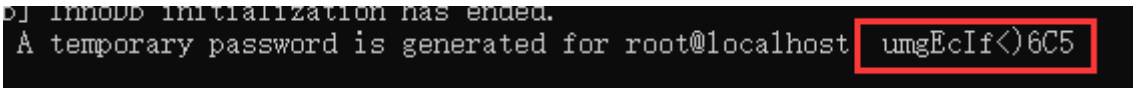
步骤4:安装mysql

在mysql的安装目录中，打开bin文件夹，运行cmd.执行初始化数据库的指令：

```
mysqld --initialize --console
```



root用户的初始化密码:



要是你不小心关掉cmd，或者没记住，那也没事，删掉初始化的 datadir 目录，再执行一遍初始化命令，又会重新生成的。

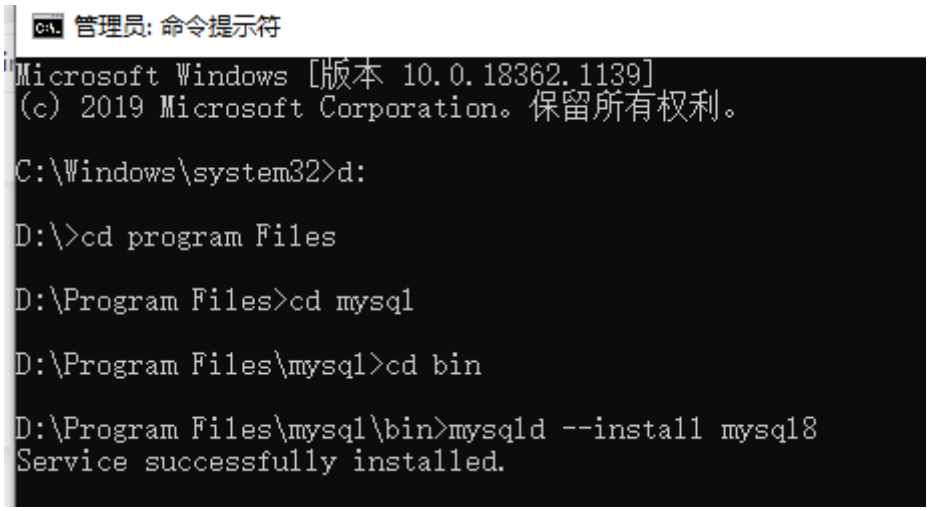
步骤5:安装服务

在MySQL安装目录的 bin 目录下执行命令:

mysqld --install [服务名]　　这里的服务名默认是mysql，可以自定义

```
D:\Program Files\mysql\bin>mysqld --install mysql8
Install/Remove of the Service Denied!
```

如果提示上述错误，需要关闭cmd,重新打开，使用管理员身份执行



安装完成之后

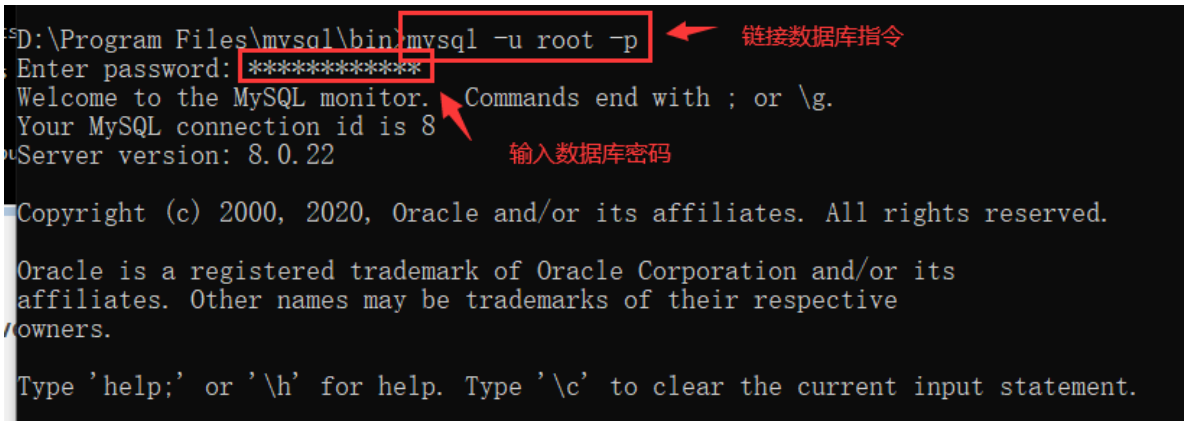
通过命令**net start mysql8**启动MySQL的服务了。

通过命令**net stop mysql8**停止服务。

```
D:\Program Files\mysql\bin>net start mysql8
mysql8 服务正在启动 .
mysql8 服务已经启动成功。


D:\Program Files\mysql\bin>net stop mysql8
mysql8 服务正在停止.
mysql8 服务已成功停止。
```

注意:安装时，卸载其他版本的mysql数据库

步骤6:链接数据库



修改账户密码:

```
alter user 'root'@'localhost' identified with mysql_native_password BY '新密码';
```

示例:

```
alter user 'root'@'localhost' identified with mysql_native_password BY '123456';
```

修改密码，注意命令尾的分号一定要有，这是mysql的语法

```
mysql> alter user 'root'@'localhost' identified with mysql_native_password BY '123456';
Query OK, 0 rows affected (0.01 sec)
```

退出数据库:

```
mysql> quit
Bye
```

1.4.2 卸载

步骤1:使用管理员身份运行cmd,关闭mysql服务

```
C:\Windows\system32>net stop mysql8
mysql8 服务正在停止.
mysql8 服务已成功停止。
```

步骤2:删除mysql服务

命令:sc delete mysql8 或者 mysqld remove mysql8

```
C:\Windows\system32>sc delete mysql8
[SC] DeleteService 成功
```

步骤3:删除mysqlDB目录文件（安装mysql时my.ini指定的目录）

第二章 SQL语言

2.1 概述

SQL：Structure Query Language（结构化查询语言），SQL被美国国家标准局（ANSI）确定为关系型数据库语言的美国标准，后来被国际化标准组织（ISO）采纳为关系数据库语言的国际标准。

各数据库厂商都支持ISO的SQL标准，**普通话**

各数据库厂商在标准的基础上做了自己的扩展，**方言**

SQL 是一种标准化的语言，它允许你在数据库上执行操作，如创建项目，查询内容，更新内容，并删除条目等操作。

Create, Read, Update, and Delete 通常称为CRUD操作。

2.2 SQL语句分类

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

- DML（Data Manipulation Language）：数据操作语言，用来定义数据库记录（数据）增删改。

- DCL（Data Control Language）：数据控制语言，用来定义访问权限和安全级别。

- DQL（Data Query Language）：数据查询语言，用来查询记录（数据）查询。

注意：sql语句以;结尾

mysql中的关键字不区分大小写

2.3 DDL操作数据库

1 创建

CREATE DATABASE语句用于创建新的数据库：

编码方式:gb2312,utf-8,gbk,iso-8859-1

```
//create database 数据库名
CREATE DATABASE mydb1;
//create database 数据库名 character set 编码方式
CREATE DATABASE mydb2 character SET GBK;
 //create database 数据库名 set 编码方式 collate 排序规则
CREATE DATABASE mydb3 character SET GBK COLLATE gbk_chinese_ci;
```

2 查看数据库

查看当前数据库服务器中的所有数据库

```
show databases;
```

查看前面创建的mydb2数据库的定义信息

```
//show create database 数据库名;
Show CREATE DATABASE mydb2;
```

3 修改数据库

```
   alter  database  数据库名  character set 编码方式
```

查看服务器中的数据库，并把mydb2的字符集修改为utf8;

```
ALTER DATABASE mydb2 character SET utf8;
```

4 删除数据库

drop database 数据库名

```
DROP DATABASE mydb3;
```

5 其他语句

查看当前使用的数据库

```
Select database();
```

切换数据库: use 数据库名

```
USE mydb2;
```

2.4 DDL操作表

CREATE TABLE语句用于创建新表。

语法:

```
CREATE TABLE 表名(
列名1 数据类型 [约束],
列名2 数据类型 [约束],
列名n 数据类型 [约束]
);
```

说明:表名，列名是自定义，多列之间使用逗号间隔，最后一列的逗号不能写

[约束] 表示可有可无

示例:

```
CREATE TABLE Employees(
   id INT ,
   age INT ,
   first VARCHAR(255),
   last VARCHAR(255)
);
```

常用数据类型:

int：整型

double：浮点型，例如double(5,2)表示最多5位，其中必须有2位小数，即最大值为999.99；默认支持四舍五入

char：固定长度字符串类型； char(10) 'aaa ' 占10位

varchar：可变长度字符串类型； varchar(10) 'aaa' 占3位

text：字符串类型，比如小说信息；

blob：字节类型，保存文件信息(视频，音频，图片)；

date：日期类型，格式为：yyyy-MM-dd；

time：时间类型，格式为：hh:mm:ss

timestamp：时间戳类型 yyyy-MM-dd hh:mm:ss 会自动赋值

datetime:日期时间类型 yyyy-MM-dd hh:mm:ss

2.4.1 其他表操作

drop table 表名;

```
DROP TABLE table_name;
```

当前数据库中的所有表

show tables;

```
SHOW TABLES;
```

查看表的字段信息

desc 表名;

```
DESC employee;
```

增加列:在上面员工表的基本上增加一个image列。

alter table 表名 add 新列名 新的数据类型

```
ALTER TABLE employee ADD image blob;
```

修改job列，使其长度为60。

alter table 表名 change 旧列名 新列名 新的数据类型

```
ALTER TABLE employee MODIFY job varchar(60);
ALTER TABLE employee  change  job job varchar(60);
```

列名name修改为username

```
ALTER TABLE user CHANGE name username varchar(100);
```

删除image列,一次只能删一列。

alter table 表名 drop 列名

```
ALTER TABLE employee DROP image;
```

修改表名,表名改为user。

alter table 旧表名 rename 新表名;

```
alter table user rename  users;
```

查看表格的创建细节

show create table 表名;

```
SHOW CREATE TABLE user;
```

修改表的字符集为gbk

alter table 表名 character set 编码方式

```
ALTER TABLE user CHARACTER SET gbk;
```

练习：

表名 card(会员卡表)

列名 数据类型

cardid int

cardnum varchar(20)

regDate date

需求：

(1)创建该表

(2)将card表名修改为CardInfo

(3)添加delDate(注销时间) 列到表中

(4)将cardnum改为varchar(30)

(5)删除regDate列

(6)删除cardInfo表

2.5 DML操作

DML是对表中的数据进行增、删、改的操作。不要与DDL混淆了。

主要包括：INSERT 、UPDATE、 DELETE

小知识：

在mysql中，字符串类型和日期类型都要用单引号括起来。

空值：null

（1）插入操作：INSERT:

insert into 表名(列名) values(数据值);

```
insert into student(stuname,stuage,stusex,birthday) values('张三1',18,'a','2000-1-1');
```

-- 注意:1多列和多个列值之间使用逗号隔开 2.列名要和列值一一对应

-- 非数值的列值两侧需要加单引号

常见错误: Data too long for column 'stusex' at row 1

-- 添加数据的时候可以将列名省略->当给所有列添加数据的时候

-- 此时列值的顺序按照数据表中列的顺序执行

insert into student values('李四',12,'1111',189.98,'2000-1-1','男','2007-1-1');

-- 同时添加多行

insert into 表名(列名) values(第一行数据),(第二行数据),(),();

```
insert into student(stuname,stuage,stusex,birthday)
values('张三3',18,'a','2000-1-1'),
('张三4',18,'a','2000-1-1'),
('张三5',18,'a','2000-1-1'),
('张三6',18,'a','2000-1-1'),
('张三7',18,'a','2000-1-1'),
('张三8',18,'a','2000-1-1');
```

注意：列名与列值的类型、个数、顺序要一一对应。

参数值不要超出列定义的长度。

如果插入空值，请使用null

插入的日期和字符一样，都使用引号括起来。

练习准备：

```
create table emp(
    id int primary key,
    name varchar(100) not null,
    gender varchar(10) not null,
    birthday date,
    salary float(10,2),
    entry_date date,
    resume text
);
```

```
INSERT INTO emp(id,name,gender,birthday,salary,entry_date,resume)
VALUES(1,'zhangsan','female','1990-5-10',10000,'2015-5-5-','goodgirl');

INSERT INTO emp(id,name,gender,birthday,salary,entry_date,resume)
VALUES(2,'lisi','male','1995-5-10',10000,'2015-5-5','good boy');

INSERT INTO emp(id,name,gender,birthday,salary,entry_date,resume)
VALUES(3,'你好','male','1995-5-10',10000,'2015-5-5','good boy');
```

sql中的运算符:

（1）算术运算符:+，-，*，/(除法),求余(%)

示例：

5/2

5%2

2/5

2%5

（2）赋值运算符:=

注：赋值方向：从右往左赋值

示例： name='张三'

(3) 逻辑运算符：

and(并且),or(或者),not（取非）

作用:用于连接多个条件时使用

(4) 关系运算符：

>,<,>=,<=,!=(不等于),=(等于),<>(不等于)

补充:查询所有数据:select * from 表名

（2）修改（更新）操作：UPDATE:

语法：UPDATE 表名 SET 列名1=列值1,列名2=列值2 ... WHERE 列名=值

练习：

```
将所有员工薪水修改为5000元。
将姓名为'zs'的员工薪水修改为3000元。
将姓名为'aaa'的员工薪水修改为4000元,resume改为ccc。
将wu的薪水在原有基础上增加1000元。
```

（3）删除操作：DELETE:

语法： DELETE from 表名 【WHERE 列名=值】

练习：

```
删除表中名称为'zs'的记录。
删除表中所有记录。
使用truncate删除表中记录。
TRUNCATE TABLE emp;
```

- DELETE 删除表中的数据，表结构还在;删除后的数据可以找回

- TRUNCATE 删除是把表直接DROP掉，然后再创建一个同样的新表。

- 删除的数据不能找回。执行速度比DELETE快。

练习题:

Manager(管理员表)：

mid 编号 int （主键）

mname 名字 varchar(20)

age 年龄 int

sex 性别 char(2)

password 密码 varchar(20)

address 地址 varchar(20)

phone 电话 varchar(20)

数据：

1 王子 18 男 123 北京 110

2 公主 20 女 456 上海 220

3 太子 23 男 789 南京 330

需求：

(1)创建表

(2)将数据插入到表中

(3)将王子的年龄修改为24

(4)将地址是南京的管理员改为天津

(5)将性别是女,并且年龄大于30的用户密码改为888888

(6)将所有用户的密码恢复最初设置111111

(7)将员工的电话中不是110的电话号码改为7654321

(8)将王子的年龄改为18，地址改为承德，性别改为女

(9) 删除王子的信息

(10)删除地址在南京并且年龄大于60的员工信息

(11)删除不在北京的员工信息

(12)删除地址在北京或上海的员工信息

(13)删除电话号码是空的员工信息

小结:

为空的条件：列名 is null or 列名=''

注:两个单引号表示空字符串

日期类型值的区别:

date：yyyy-MM-dd （年月日）

time：hh:mm:ss (时分秒)

datetime:yyyy-MM-dd hh:mm:ss (年月日时分秒)

获取当前系统时间:now()

```
select now();
```

2.6 DCL

1、创建用户：

create user 用户名@指定ip identified by 密码;

```
create user test123@localhost IDENTIFIED by 'test123'
```

create user 用户名@客户端ip identified by 密码; 指定IP才能登陆

```
create user test456@10.4.10.18 IDENTIFIED by 'test456'
```

create user 用户名@‘% ’ identified by 密码 任意IP均可登陆

```
create user test7@'%'  IDENTIFIED by 'test7'
```

2、用户授权：

grant 权限1,权限2,……,权限n on

数据库名.* to 用户名@IP; 给指定用户授予指定指定数据库指定权限

```
grant select,insert,update,delete,create on
chaoshi.* to 'test456'@'127.0.0.1';
```

grant all on . to 用户名@IP 给指定用户授予所有数据库所有权限

```
grant all on *.* to 'test456'@'127.0.0.1'
```

3、用户权限查询：

show grants for 用户名@IP;

```
show grants for 'root'@'%';
```

4、撤销用户权限：

revoke 权限1,权限2,……,权限n on 数据库名.* from 用户名@IP;

```
REVOKE SELECT ON *.* FROM 'root'@'%' ;
```

5、删除用户：

drop user 用户名@IP;

```
drop user test123@localhost;
```

第三章 DQL数据查询

DQL数据查询语言（重要）

数据库执行DQL语句不会对数据进行改变，而是让数据库发送结果集给客户端。

查询返回的结果集是一张虚拟表。

查询关键字：SELECT

语法： SELECT 列名 FROM 表名 【WHERE --> BROUP BY-->HAVING--> ORDER BY】

* 表示所有列

```
SELECT    要查询的列名称
FROM     表名称
WHERE 限定条件 /*行条件*/
GROUP BY grouping_columns /*对结果分组*/
HAVING condition /*分组后的行条件*/
ORDER BY sorting_columns /*对结果分组*/
LIMIT offset_start, row_count /*结果限定*/
```

示例操作：

1>创建学生表并添加数据

```
#创建表stu
CREATE TABLE stu (
    sid CHAR(6),
    sname  VARCHAR(50),
    age INT,
    gender VARCHAR(50)
);
#添加数据
INSERT INTO stu VALUES('S_1001', 'liuYi', 35, 'male');
INSERT INTO stu VALUES('S_1002', 'chenEr', 15, 'female');
INSERT INTO stu VALUES('S_1003', 'zhangSan', 95, 'male');
INSERT INTO stu VALUES('S_1004', 'liSi', 65, 'female');
INSERT INTO stu VALUES('S_1005', 'wangWu', 55, 'male');
INSERT INTO stu VALUES('S_1006', 'zhaoLiu', 75, 'female');
INSERT INTO stu VALUES('S_1007', 'sunQi', 25, 'male');
INSERT INTO stu VALUES('S_1008', 'zhouBa', 45, 'female');
INSERT INTO stu VALUES('S_1009', 'wuJiu', 85, 'male');
INSERT INTO stu VALUES('S_1010', 'zhengShi', 5, 'female');
INSERT INTO stu VALUES('S_1011', 'xxx', NULL, NULL);
```

2>创建雇员表并添加数据

```
#创建雇员表
CREATE TABLE emp2(
    empno INT,
    ename VARCHAR(50),
    job VARCHAR(50),
    mgr INT,
    hiredate DATE,
    sal DECIMAL(7,2),
    comm decimal(7,2),
    deptno INT
) ;
#添加数据
INSERT INTO emp2 values(7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);
INSERT INTO emp2 values(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);
INSERT INTO emp2 values(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);
INSERT INTO emp2 values(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);
INSERT INTO emp2 values(7654,'MARTIN','SALESMAN',7698,'1981-09-
28',1250,1400,30);
INSERT INTO emp2 values(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);
INSERT INTO emp2 values(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);
INSERT INTO emp2 values(7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20);
INSERT INTO emp2 values(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);
```

```
INSERT INTO emp2 values(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);
INSERT INTO emp2 values(7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20);
```

3>创建部门表并添加数据

#创建部门表

```
CREATE TABLE dept(
    deptno INT,
    dname varchar(14),
    loc varchar(13)
);
#添加数据
INSERT INTO dept values(10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO dept values(20, 'RESEARCH', 'DALLAS');
INSERT INTO dept values(30, 'SALES', 'CHICAGO');
INSERT INTO dept values(40, 'OPERATIONS', 'BOSTON');
```

3.1 简单查询

查询所有列

```
SELECT * FROM stu;
```

查询指定列

```
SELECT sid, sname, age FROM stu;
```

3.2 条件查询

条件查询就是在查询时给出WHERE子句，在WHERE子句中可以使用如下运算符及关键字：

=、!=、<>、<、<=、>、>=; BETWEEN...AND； IN(set)； IS NULL； AND；OR； NOT；

(1)查询性别为女，并且年龄50以内的记录

```
SELECT * FROM stu WHERE gender='female' AND age<50;
```

(2)查询学号为S_1001，或者姓名为liSi的记录

```
SELECT * FROM stu WHERE sid ='S_1001' OR sname='liSi';
```

(3)查询学号为S_1001，S_1002，S_1003的记录

列名 in (列值1,列值2)

```
SELECT * FROM stu  WHERE sid IN ('S_1001','S_1002','S_1003');
```

(4)查询学号不是S_1001，S_1002，S_1003的记录

```
SELECT * FROM tab_student  WHERE sid NOT IN('S1001','S1002','S_1003');
```

(5)查询年龄为null的记录

```
SELECT * FROM stu WHERE age IS NULL;
```

(6)　查询年龄在20到40之间的学生记录

```
SELECT * FROM stu  WHERE age>=20 AND age<=40;
```

或者:列名 between 开始值 and 结束值；//注意:1.开始值<结束值 2.包含临界值的

```
SELECT * FROM stu WHERE age BETWEEN 20 AND 40;
```

(7)　查询性别非男的学生记录

```
SELECT * FROM stu WHERE gender!='male';
```

或者

```
SELECT * FROM stu WHERE gender<>'male';
```

或者

```
SELECT * FROM stu  WHERE NOT gender='male';
```

(8) 查询姓名不为null的学生记录

```
SELECT * FROM stu  WHERE NOT sname IS NULL;
```

或者

```
SELECT * FROM stu  WHERE sname IS NOT NULL;
```

3.3 模糊查询

当想查询姓名中包含a字母的学生时就需要使用模糊查询了。模糊查询需要使用关键字LIKE。

语法: 列名　like '表达式'　//表达式必须是字符串

通配符:

_(下划线):　任意一个字符

%：任意0~n个字符,'张%'

(1)查询姓名由3个字构成的学生记录

```
SELECT * FROM stu  WHERE sname LIKE '___';
```

模糊查询必须使用LIKE关键字。其中“”*匹配任意一个字，3个*“”表示3个任意字。

(2)查询姓名由5个字母构成，并且第5个字母为“i”的学生记录

```
SELECT * FROM stu   WHERE sname LIKE '____i';
```

(3)查询姓名以“z”开头的学生记录

```
SELECT * FROM stu  WHERE sname LIKE 'z%';
```

其中“%”匹配0~n个任何字母。

(4)查询姓名中第2个字母为“i”的学生记录

```
SELECT * FROM stu WHERE sname LIKE '_i%';
```

(5)查询姓名中包含“a”字母的学生记录

```
SELECT * FROM stu WHERE sname LIKE '%a%';
```

3.4 字段控制查询

(1)去除重复记录

去除重复记录（两行或两行以上记录中系列的上的数据都相同），例如emp表中sal字段就存在相同的记录。当只查询emp表的sal字段时，那么会出现重复记录，那么想去除重复记录，需要使用DISTINCT：

```
SELECT DISTINCT sal FROM emp;
```

(2)查看雇员的月薪与佣金之和

因为sal和comm两列的类型都是数值类型，所以可以做加运算。如果sal或comm中有一个字段不是数值类型，那么会出错。

```
SELECT *,sal+comm FROM emp;
```

comm列有很多记录的值为NULL，因为任何东西与NULL相加结果还是NULL，所以结算结果可能会出现NULL。下面使用了把NULL转换成数值0的函数IFNULL：

```
SELECT *,sal+IFNULL(comm,0) FROM emp;
```

(3)给列名添加别名

在上面查询中出现列名为sal+IFNULL(comm,0)，这很不美观，现在我们给这一列给出一个别名，为total：

```
SELECT *, sal+IFNULL(comm,0) AS  total FROM emp;
```

给列起别名时，是可以省略AS关键字的：

```
SELECT *,sal+IFNULL(comm,0)  total FROM emp;
```

3.5 排序

语法： order by 列名 asc/desc

//asc 升序 desc 降序 默认不写的话是升序

(1) 查询所有学生记录，按年龄升序排序

```
SELECT *
FROM stu
ORDER BY age ASC;
```

或者

```
SELECT *
FROM stu
ORDER BY age;
```

(2) 查询所有学生记录，按年龄降序排序

```
SELECT *
FROM stu
ORDER BY age DESC;
```

(3) 查询所有雇员，按月薪降序排序，如果月薪相同时，按编号升序排序

多列排序：当前面的列的值相同的时候，才会按照后面的列值进行排序

```
SELECT * FROM emp
ORDER BY sal DESC,empno ASC;
```

3.6 聚合函数

聚合函数是用来做纵向运算的函数：

COUNT(列名)：统计指定列不为NULL的记录行数；

MAX(列名)：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；

MIN(列名)：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；

SUM(列名)：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；

AVG(列名)：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；

(1) COUNT

当需要纵向统计时可以使用COUNT()。

查询emp表中记录数：

```
SELECT COUNT(*) AS cnt FROM emp;
```

查询emp表中有佣金的人数：

```
SELECT COUNT(comm) cnt FROM emp;
```

注意，因为count()函数中给出的是comm列，那么只统计comm列非NULL的行数。

l 查询emp表中月薪大于2500的人数：

```
SELECT COUNT(*) FROM emp WHERE sal > 2500;
```

l 统计月薪与佣金之和大于2500元的人数：

```
SELECT COUNT(*) AS cnt FROM empWHERE sal+IFNULL(comm,0) > 2500;
```

l 查询有佣金的人数，以及有领导的人数：

```
SELECT COUNT(comm), COUNT(mgr)FROM emp;
```

(2) SUM和AVG

当需要纵向求和时使用sum()函数。

l 查询所有雇员月薪和：

```
SELECT SUM(sal) FROM emp;
```

l 查询所有雇员月薪和，以及所有雇员佣金和：

```
SELECT SUM(sal), SUM(comm) FROM emp;
```

l 查询所有雇员月薪+佣金和：

```
SELECT SUM(sal+IFNULL(comm,0))FROM emp;
```

l 统计所有员工平均工资：

```
SELECT AVG(sal) FROM emp;
```

(3) MAX和MIN

l 查询最高工资和最低工资：

```
SELECT MAX(sal), MIN(sal) FROM  emp;
```

3.7 分组查询

当需要分组查询时需要使用GROUP BY子句，例如查询每个部门的工资和，这说明要使用部分来分组。

注意:如果查询语句中有分组操作，则select后面能添加的只能是聚合函数和被分组的列名

3.7.1 分组查询

l 查询每个部门的部门编号和每个部门的工资和：

```
SELECT deptno, SUM(sal) FROM emp GROUP BY deptno;
```

查询每个部门的部门编号以及每个部门的人数：

```
SELECT deptno,COUNT(*) FROM emp GROUP BY deptno;
```

l 查询每个部门的部门编号以及每个部门工资大于1500的人数：

```
SELECT deptno,COUNT(*) FROM emp WHERE sal>1500 GROUP BY deptno;
```

3.7.2 HAVING子句

l 查询工资总和大于9000的部门编号以及工资和：

```
SELECT deptno, SUM(sal)
FROM emp
GROUP BY deptno
HAVING SUM(sal) > 9000;
```

注：having与where的区别:

1.having是在分组后对数据进行过滤,where是在分组前对数据进行过滤

2.having后面可以使用分组函数(统计函数)

where后面不可以使用分组函数。

WHERE是对分组前记录的条件，如果某行记录没有满足WHERE子句的条件，那么这行记录不会参加分组；而HAVING是对分组后数据的约束。

补充: 多列分组

-- 统计出stu表中每个班级的男女生各多少人

select gradename,gender ,count(*) from stu

group by gradename,gender

3.8 LIMIT

LIMIT用来限定查询结果的起始行，以及总行数。

limit 开始下标,显示条数；//开始下标从0开始

limit 显示条数；//表示默认从0开始获取数据

1.查询5行记录，起始行从0开始

```
SELECT * FROM emp LIMIT 0, 5;
```

注意，起始行从0开始，即第一行开始！

2.查询10行记录，起始行从3开始

```
SELECT* FROM emp LIMIT 3, 10;
```

3.8.1 分页查询

如果一页记录为10条，希望查看第3页记录应该怎么查呢?

l 第一页记录起始行为0，一共查询10行； limit 0,10

l 第二页记录起始行为10，一共查询10行；limit 10,10

l 第三页记录起始行为20，一共查询10行； limit 20,10

pageIndex 页码值　pageSize 每页显示条数

```
limit   (pageindex-1)*pagesize,pagesize;
```

查询语句书写顺序：select – from- where- groupby- having- order by-limit

查询语句执行顺序：from - where -group by -having - select - order by-limit

第四章 使用开发工具实现数据库操作

4.1 基于Navicat实现数据库操作

4.2 基于SQLyog实现数据库操作

练习1:

Manager(管理员表):

mid 编号　int （主键）

mname 名字　varchar(20)

age 年龄　int

sex 性别　char(2)

password 密码 varchar(20)

address 地址 varchar(20)

phone　电话 varchar(20)

数据:

1 王子 18 男 123 北京 110

2 公主 20 女 456 上海　220

3 太子 23 男 789 南京　330

(14)查询公主的所有信息

(15)查询年龄在18-30之间的管理员姓名

(16)查询表中所有的用户名和电话

(17)查询性别是男，名字是王子的个人信息

(18)查询出地址在北京和上海的员工信息

练习2:

scores

stuid int 学生id

java int java成绩

mysql int mysql成绩

stuname varchar(20) 学生姓名

数据:

1 67 78 张三

2 87 55 李四

3 66 90 王五

4 98 78 赵六

5 80 88 田七

需求：

(1)对java成绩进行降序排序

(2)得到mysql成绩前三名

(3)得到java学生中最后一名的学生信息

(4)查询出两门成绩都优秀(>=80)的学生姓名

(5)查询出成绩在90分以上(>=90)的学生信息

(6)查询出每名学员的java,mysql,总成绩

(7)显示出每名学生的总分以及姓名

练习3：

测试数据：

郭敬明 1371234567 北京 java S1101 89 1979-04-05

张三丰 1372839201 上海 数据库 S1102 67 1967-09-07

赵敏 1387839201 山东 mysql S1103 99 1987-09-07

Student2

stuname 姓名 varchar(20)

telphone 电话 varchar(20)

address 住址 varchar(20)

subject 科目 varchar(20)

stuNo 学号 varchar(20)

score 成绩 int

birthday 出生日期 date

//1.要查询列 2.条件

a.查询住址为“山东”的学生姓名、电话、住址

b.查询名称中含有“数据库”字样科目信息

c.查询电话中以“1387”开头的学生信息

d.查询姓姜的，三个字的学生信息

e.查询学号为S1101的指定java,mysql科目考试成绩

f.查询出80后学员信息

g.查询出家庭住址在北上广的学生名字

h.显示成绩在第5-10名的学生名字和电话

i.查询分数在80-90之间并且在北京的学生

练习4:聚合函数练习

表：scores2

年级 grade varchar(10)

学号 stuno varchar(20)

考试时间 examDate date

科目 subject varchar(20)

成绩 score int

学期 xueqi int

数据:

S1 S1101 2015-02-03 C 89 1

S2 S1103 2015-03-03 JAVA 90 2

S3 S1102 2015-07-03 C 100 1

1.查询学生总人数

2.学号为S1101的学生第一学期考试总成绩,平均分

3.查询2013年3月22日科目“C”的最高分、最低分、平均分

4.查询2013年3月22日科目“C”及格学生的平均分

5.查询所有参加“C”科目考试的平均分

6.查看考java的人数

练习5:分组练习

表名：student

年级(grade) varchar(10)

学生姓名(name) varchar(10)

学时(xueshi) int --每人单个学时

参加考试(isexam) char(1) 是/否、

课程(subject) varchar(10)

分数(score) int

数据:

1 张三 10 是 java 99

1 李四 10 否 java 0

2 王五 20 是 mysql 88

2 赵六 20 是 mysql 77

2 王五 20 是 java 99

2 赵六 20 否 java 0

1 张三 10 是 mysql 88

练习：

a:查询每个年级的总学时数，并按照升序排列

b:查询每个参加考试的学员的平均分

c:查询每门课程的平均分，并按照降序排列

练习6:综合练习

Student

科目名称 subjectName varchar(20)

学生姓名 stuname varchar(20)

学生地址 address varchar(20)

学生性别 sex char(2)

电子邮件 email varchar(30)

年级 grade varchar(10)

出生日期 birthday date

考试日期 examDate date

成绩 scores int

数据：

JAVA 张三 北京 男 123@qq.com S1 1990-03-04 2013-5-6 89

html 李四 上海 男 S2 1993-08-04 2014-5-6 87

html 王五 北京 男 123@qq.com S2 1990-03-04 2015-4-6 90

1.查询S2的科目名称

2.查询S2男同学的姓名和住址

3.查询无电子邮件的学生姓名和年级信息

4.查询出生日期在1993年之后的S2的学生姓名和年级信息

5.查询参加了日期为2013年2月15日的“HTML” 科目考试的成绩信息