# 6.MySQL

## 6.1.数据库

**什么是数据库**

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，简而言之就是存储数据的仓库。

**数据库的分类：**

层次式数据库, 网络式数据库, 关系型数据库

数据库可以按照一定的数据结构存储管理大量的数据及数据与数据之间的关系，它本质上是一种信息管理系统。数据库根据存储采用的数据结构的不同可以分为许多种，其中常见的有层次式数据库、网络式数据库、关系型数据库。其中关系型数据库占据着市场的主流

**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。**这种定义听起来十分抽象，这里我们不深入讨论什么叫做“关系模型”--大学计算机专业专门有一门课叫“离散数学”专门讨论过关系模型 --，只是简单的表述为 **利用表来存储数据，用表和表之间的关系保存数据之间的关系的数据库称为关系型数据库**，这个定义不太严谨，但是更好理解。

**常见的关系型数据库:**

SqlServer

Oracle

MySql

DB2

...

**MySql介绍**

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是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL的特点是体积小、速度快、总体拥有成本低，并且开源。

**MySql数据库的安装与配置**

mysql下载地址：<https://www.mysql.com/downloads/>， 安装步骤：

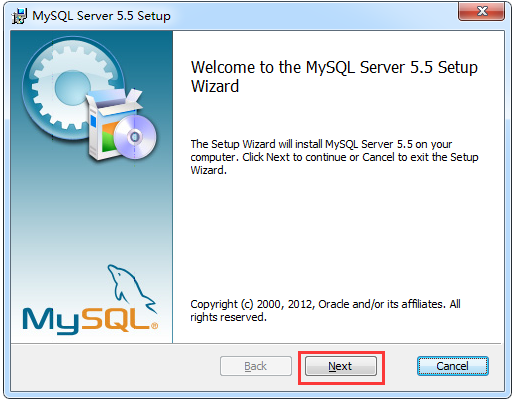
！！！mysql安装的路径中不要有中文和空格

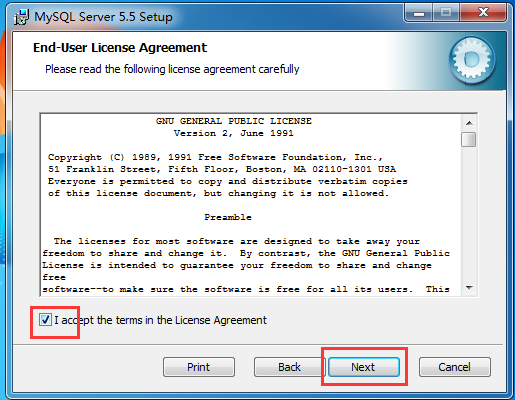
默认的端口3306不要修改, 保持默认即可

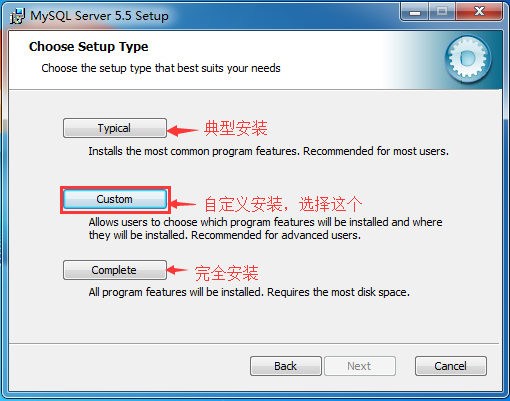
**安装文件存放路径：不能有中文和空格！**

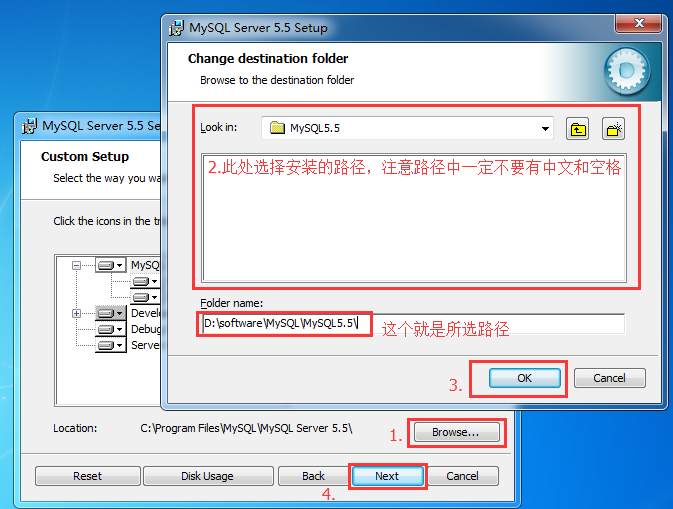
打开安装程序

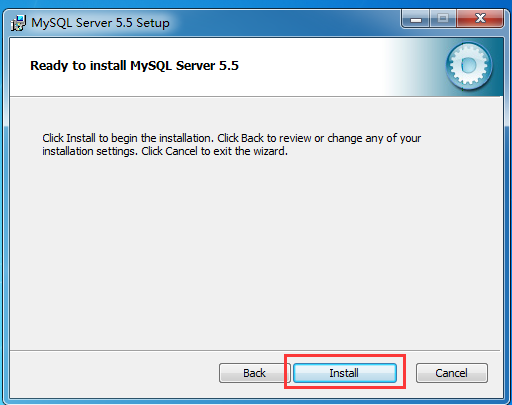


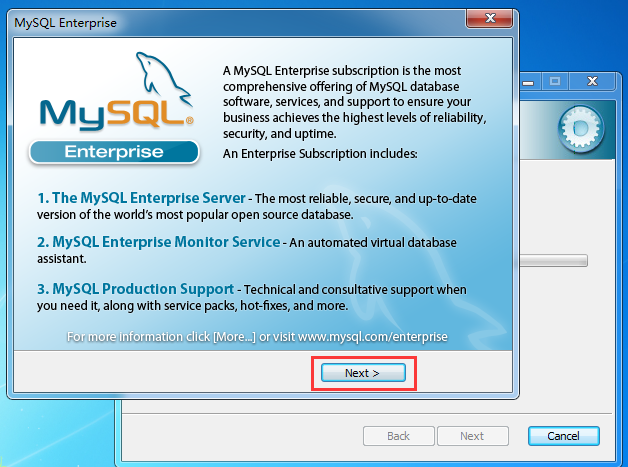


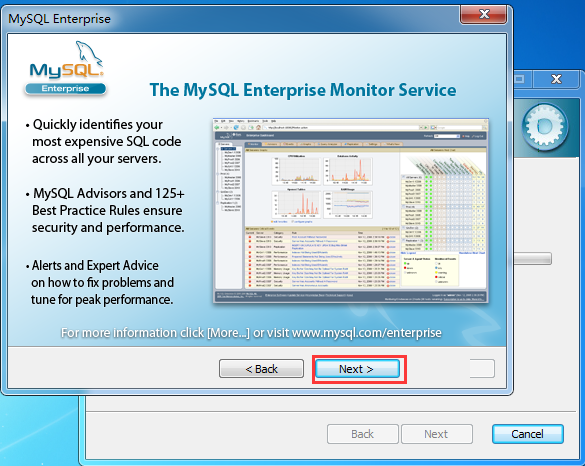


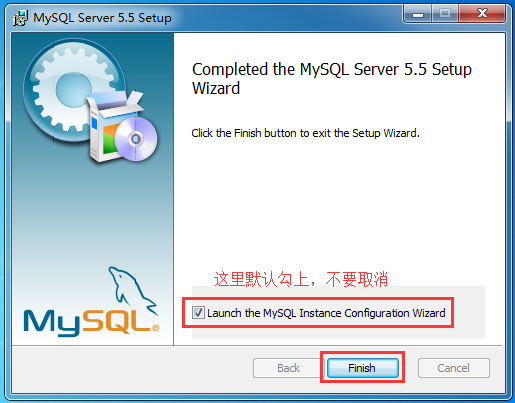


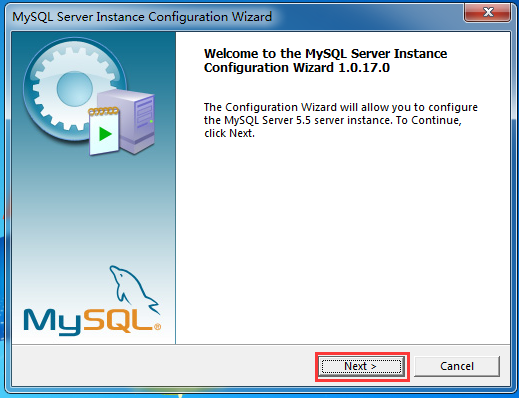


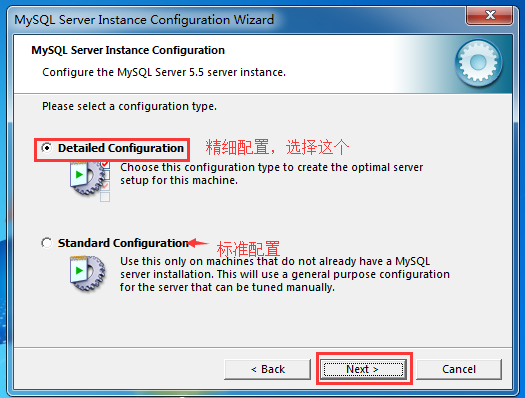


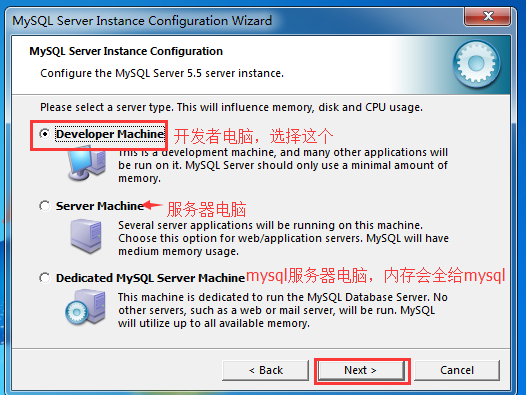


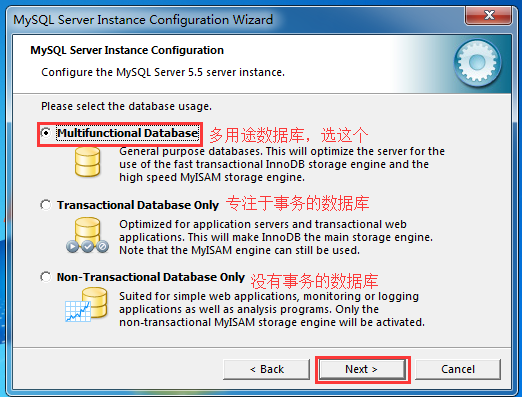


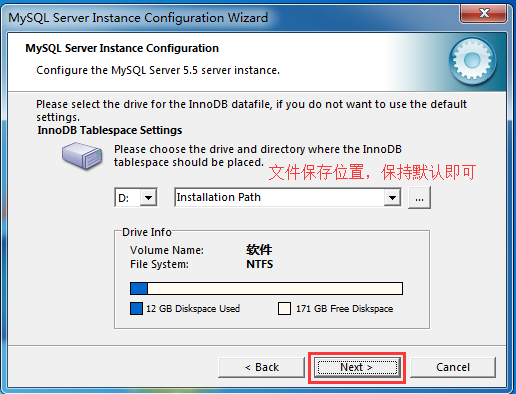


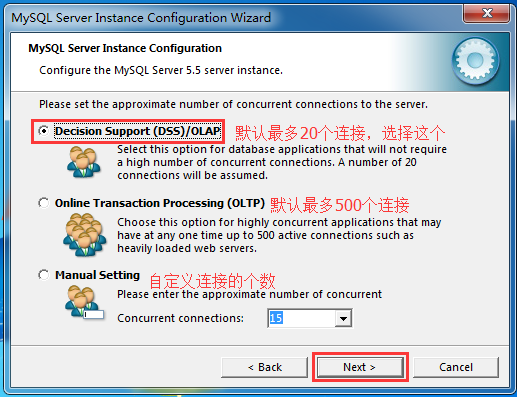


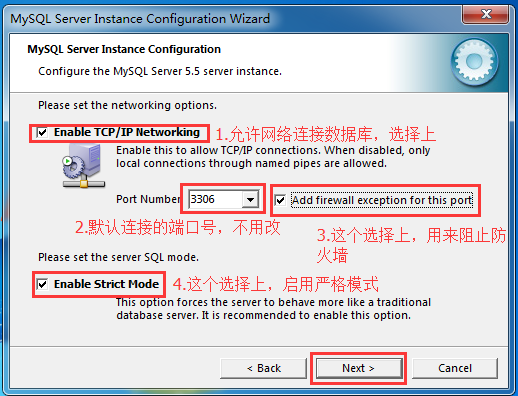


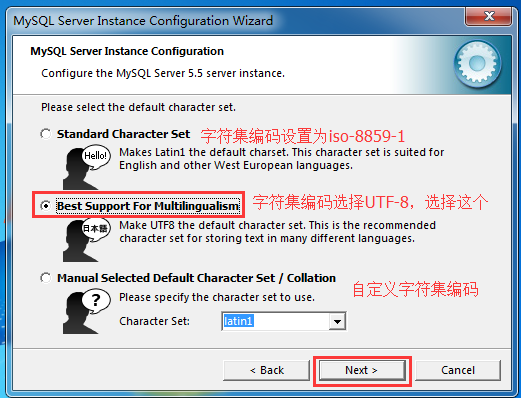


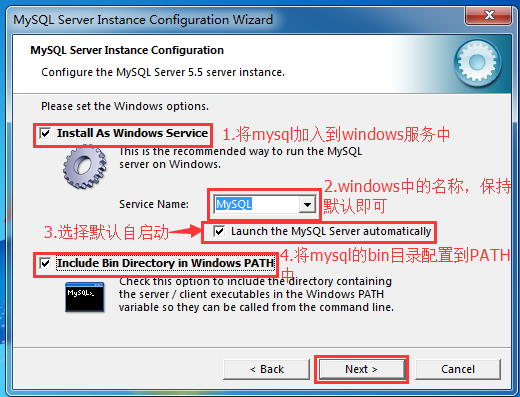


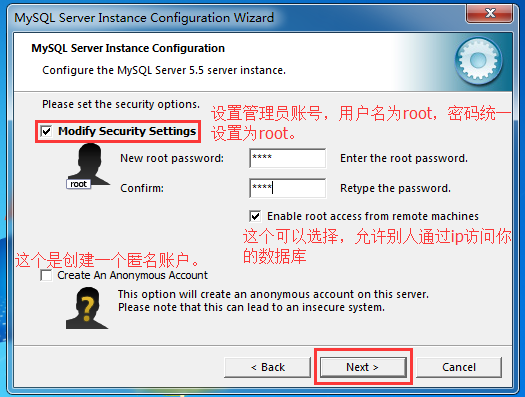


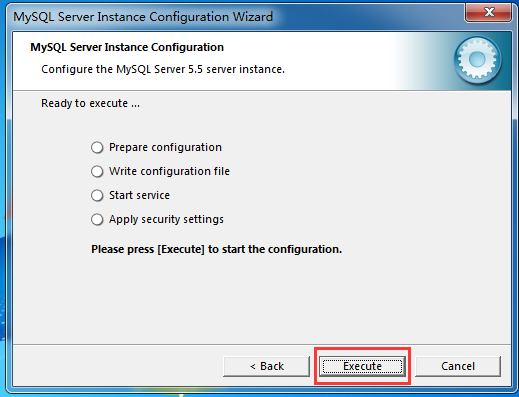


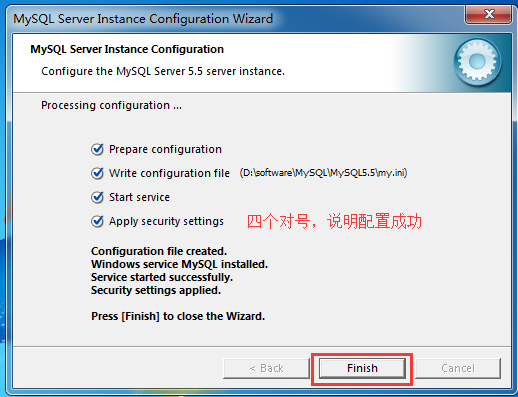










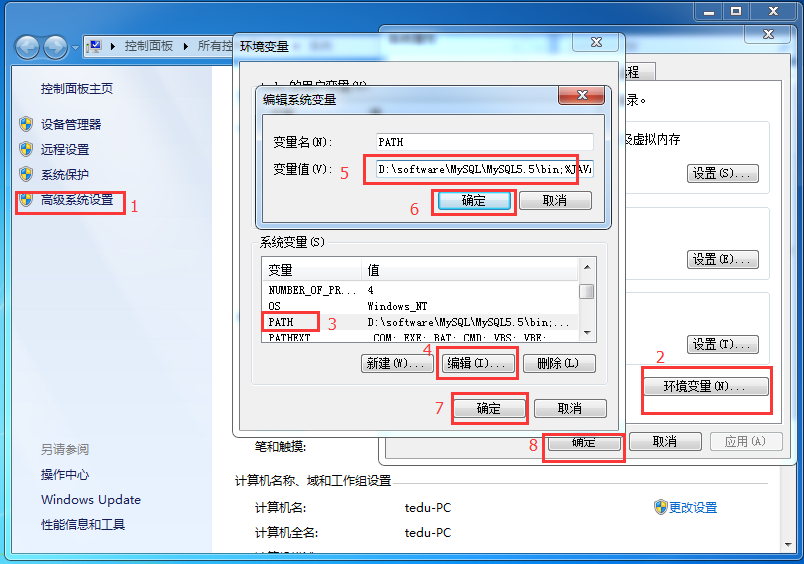


配置环境变量PATH:

找到mysql的安装目录下的bin目录，比如D:\software\MySQL\MySQL5.5\bin

然后：

计算机右键->属性->高级系统设置->环境变量->系统变量中找PATH->编辑->变量值->在最前面加上路径，并以“;”分号结尾，与后面的变量分隔开，注意是在英文状态下输入分号。



校验是否安装成功：

打开cmd窗口，输入mysql -uroot -proot 回车



如果有如上欢迎进入的提示，则证明安装成功

卸载：先停止服务：计算机->管理->服务->mysql->停止

然后卸载：控制面板->程序和功能->mysql->卸载

最后清除目录：C:\ProgramData\MySQL 注意，此目录是隐藏的，先显示隐藏目录

**MySQL相关命令**

**登录mysql客户端命令：mysql -uroot -p123 -h localhost；**

-u：后面的root是用户名，这里使用的是超级管理员root；

-p：后面的123是密码，这是在安装MySQL时就已经指定的密码；

-h：后面给出的localhost是服务器主机名，它是可以省略的，例如：mysql -u root -p 123；

**退出客户端命令：quit或exit；**

扩展: 通过命令启动、停止、删除服务(在C:\Windows\System32下以管理员运行cmd.exe)

启动：net start 服务名称

关闭：net stop 服务名称

删除：sc delete 服务名称

(如果服务名中间有空格，就需要前后加引号)

问题:



没有配置mysql的bin目录到path环境变量中!!!!

**MySql常见的概念**

数据库服务器： 安装的mysql软件就是mysql服务器。

数据库： 是数据库表的集合，数据库服务器中可以创建多个数据库。

表： 类似于一个excel表格，用于存储数据。

表记录： 即表中的一行数据。

## 6.2.SQL语言

**SQL概述**

SQL，指结构化查询语言，全称是 Structured Query Language。

SQL是操作关系型数据库通用的语言

SQL是非过程性语言

SQL是一个标准，各大数据库厂商都根据该标准提供了实现。

数据库厂商为了增强数据库的功能，添加了一些非标准的SQL，称之为数据库的“方言”。

**SQL能干什么？**

SQL 可以创建、修改、删除数据库，包括查询服务器中所有数据库。

SQL 可以创建、修改、删除数据库表，包括查询数据库中所有的表

SQL 可以在数据库中插入新的记录，或修改、删除已有的记录

SQL可以查询数据库中的数据

……

## 6.3.利用sql语句来操作数据库

### 6.3.1.创建数据库

**语法：**

CREATE DATABASE [IF NOT EXISTS] db\_name [CHARACTER SET charset\_name] [COLLATE collation\_name]

其中charset\_name是为数据库指定的默认字符集

Collate是为数据库指定的默认校对规则

（校对规则是在字符集内用于比较字符的一套规则,可以控制select查询时where条件大小写是否敏感的规则。)

**练习：**

~创建一个名称为mydb1的数据库。

create database mydb1;

~创建一个使用gbk字符集的mydb2数据库。

create database mydb2 character set gbk;

### 6.3.2.查看数据库

**语法：**

显示所有数据库：

SHOW DATABASES

显示数据库创建语句：

SHOW CREATE DATABASE db\_name

**练习：**

~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 mydb2;

### 6.3.3.修改数据库

**注意: 数据库一旦创建成功, 其名字无法修改!**

**语法：**

ALTER DATABASE db\_name  [CHARACTER SET charset\_name] [COLLATE collation\_name]

**练习：**

~查看服务器中的数据库，并把mydb2库的字符集修改为utf8

alter database mydb2 character set utf8;

### 6.3.4.删除数据库

**语法：** DROP DATABASE [IF EXISTS] db\_name

**练习：**

~删除前面创建的mydb2数据库

DROP DATABASE mydb2;

### 6.3.5.选择数据库

语法：

选择数据库: use db\_name;

查询当前选择的数据: select database();

( 没有退出数据库的命令, 如果想退出当前数据库进入另一个数据,直接use切换到另一个数据库就可以了 )

## 6.4.利用sql语句来操作数据库表

### 6.4.1.MySQL中常见的数据类型

**字符串类型**

**char(n)** 定长字符串

n的最大值为255, 表示存储255个字符

例如：stuid char(10)

一般用于长度固定的字符串，比如手机号、身份证号等，效率高

**varchar(n)** 不定长字符串

n的最大值为 65535个字节(5.0以后的版本)

例如：username varchar(50)

一般用于长度不固定的字符串，比如用户名、昵称等，节省空间

**数值类型**

**TINYINT:**占用1个字节，相对于java中的byte

**SMALLINT:**占用2个字节，相对于java中的short

**INT：**占用4个字节，相对于java中的int

**BIGINT：**占用8个字节，相对于java中的long

**FLOAT：**4字节单精度浮点类型，相对于java中的float

**DOUBLE：**8字节双精度浮点类型，相对于java中的double

**大数据类型**

**BLOB：**

大二进制类型，可以存入二进制类型的数据，通过这个字段，可以将图片、音频、视频等数据以二进制的形式存入数据库。最大为4GB。

**TEXT：**

大文本，被声明为这种类型的字段，可以保存大量的字符数据，最大为4GB。

注意：text属于mysql的方言，在其他数据库中为clob类型

**日期类型**

DATE：日期 2017-11-05

TIME：时间 格式 'HH:MM:SS' 19:19:19

DATETIME：日期时间 2017-11-05 19:19:19 年份范围：1000~9999

TIMESTAMP：时间戳 2017-11-05 19:19:19 年份范围：1970~2037

**逻辑型**

BIT型字段只能取两个值：0或1。

### 6.4.2.字段的约束

主键约束：保证所约束列中的值必须唯一且不能为空。

添加主键约束(创建表时)： col\_name datatype primary key

或(表已存在)： Alter table tabname add primary key(col\_name)

设置主键自动增长：col\_name datatype primary key auto\_increment

唯一约束： unique 保证所约束列的值必须唯一，即不能重复。

非空约束： not null 保证所约束列的值不能为空

外键约束：(放到后面单讲)

### 6.4.3.新增表

**语法：**

CREATE TABLE table\_name (

field1 datatype,

field2 datatype,

field3 datatype

)[character set 字符集] [collate 校对规则]

field：指定列名　datatype：指定列类型

**注意：**

创建表时，要根据需保存的数据创建相应的列，并根据数据的类型定义相应的列类型。

创建表时，一般不需要指定字符集编码和校对规则，和数据库保持一致即可。

**练习:**

~创建employee表，表中字段信息如下：



建表语句1：

create table employee(

id int,

name varchar(50),

gender char(1),

birthday date,

entry\_date date,

job varchar(50),

salary double,

resume text

);

建表语句2：

create table employee(

id int primary key auto\_increment,

name varchar(50) unique,

gender bit not null,

birthday date,

entry\_date date,

job varchar(50),

salary double,

resume text

);

### 6.4.4.查看表

**语法：**

查看表结构：

desc tabName

查看当前所有表：

show tables

查看当前数据库表建表语句

show create table tabName;

### 6.4.5.修改表

**语法：**

增加列：

ALTER TABLE tabname ADD col\_name datatype [DEFAULT expr][,ADD col\_name datatype...];

修改列：

ALTER TABLE tabname MODIFY (col\_name datatype [DEFAULT expr][,MODIFY col\_name datatype]...);

删除列：

ALTER TABLE tabname DROP [COLUMN] col\_name;

修改表名：

ALTER TABLE old\_tabname RENAME TO new\_tabname;

或

RENAME TABLE old\_tabname TO new\_tabname;

修改列名称：

ALTER TABLE tabname CHANGE [COLUMN] old\_col\_name new\_col\_name datatype;

修改列的顺序：

ALTER TABLE tabname MODIFY col\_name1 datatype AFTER col\_name2;

修改表的字符集：

ALTER TABLE tabname CHARACTER SET character\_name;

**练习：**

~在上面员工表的基础上增加一个image列。

alter table employee add image blob;

~修改job列，使其长度为60。

alter table employee modify job varchar(60);

~删除gender列。

alter table employee drop gender;

~表名改为user。

alter table employee rename to user;

或

rename table user to employee;

~列名name修改为username

alter table employee change name username varchar(100);

~将image插入到id列的后面

alter table employee modify image blob after id;

~修改表的字符集为GBK

alter table employee character set gbk;

### 6.4.6.删除表

**语法：** DROP TABLE tabname;

**练习：** ~删除employee表

drop table employee;

## 6.5.利用sql语句来操作数据库表记录

### 6.5.1.INSERT

**语法：**

INSERT INTO tabname [(column [, column...])] VALUES (value [, value...]);

**注意：**

\* 插入的数据应与字段的数据类型相同

例如: name列对应的值为字符串, 插入的数据类型也应该是字符串类型!

\* 数据的大小应在列的规定范围内

例如: name列字符长度为10, 那么插入的数据长度不能超过10!

\* 在values中列出的数据位置必须与被对应列的排列位置相对应

\* 字符串和日期格式的数据要用单引号引起来

例如: '张飞', '2017-6-1'

\* 如果要插入所有字段可以省写列列表，直接按表中字段声明的顺序写值列表

**练习：**

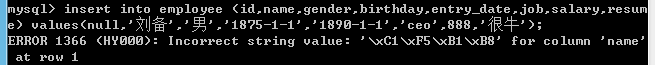
~向员工表中插入三条数据

insert into employee (id, name, gender, birthday, entry\_date, job, salary, resume) values(null, '刘备', '男', '1876-1-1', '1902-1-1', 'CEO', 100000, '很牛!!!');

insert into employee values(null, '张飞', '男' , '1879-1-1', '1905-1-1', '大数据工程师', 20000, '还行吧!');

insert into employee values(null, '小乔', '女' , '1882-1-1', '1906-1-1', '程序员鼓励师', 40000, '美女!');

**\*\* 在MySQL中插入中文数据，或查询中文数据时的乱码问题:**

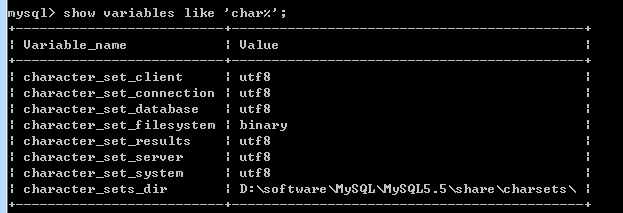


\*\*乱码问题：客户端发送的数据是GBK的，而服务器使用的utf-8的编码来处理客户端发来的数据，两端使用的编码不一致导致乱码。

\*\*解决方法1：在客户端可以通过 set names gbk; 通知服务器使用指定码表来处理客户端发送的数据。这种方式只对当前cmd命令窗口有效，每次新开的窗口都要设置一次。

\*\*解决方式2：可以通过修改MySQL安装目录下的my.ini文件(57行)中的配置，来指定服务器端使用的码表。这种方式一劳永逸。(注意: 配置完成后需要重启服务器!!);

查询数据库中编码: show variables like 'char%';



### 6.5.2.UPDATE

**语法：**

UPDATE tab\_name SET col\_name1=expr1 [, col\_name2=expr2 ...] [WHERE where\_definition]

UPDATE语法可以用新值更新原有表行中的各列。

SET子句指示要修改哪些列和要给予哪些值。

WHERE子句指定应更新哪些行。如没有WHERE子句，则更新所有的行。

**练习：**

~将所有员工薪水修改为30000元。

update employee set salary=30000;

~将姓名为’貂蝉’的员工薪水修改为11000元。

update employee set salary=11000 where name='貂蝉';

~将张飞的薪水在原有基础上增加1000元。

update employee set salary=salary+1000 where name=’张飞’;

注意：mysql不支持+=符号

### 6.5.3.DELETE

**语法：**

DELETE FROM tab\_name [WHERE where\_definition]

where用来筛选要删除的记录，如果不使用where子句，将删除表中所有数据。

delete语句不能删除某一列的值

delete语句仅删除记录，不删除表本身。如要删除表，使用drop table语句。

**练习：**

~删除表中名称为’张飞’的记录。

delete from employee where name='张飞';

~删除表中所有记录。

delete from employee;

### 6.5.4.SELECT

**基本的查询**

**语法：**

SELECT [DISTINCT] \* | {column1, column2. column3..} FROM table\_name;

select 指定查询哪些列的数据。

column指定列名。

\* 号代表查询所有列。

from指定查询哪张表。

DISTINCT可选，指显示结果时，是否剔除重复数据

**练习：**

执行下面的SQL，创建exam表并插入数据

create table exam(

id int primary key auto\_increment,

name varchar(20) not null,

chinese double,

math double,

english double

);

insert into exam values(null,'关羽',85,76,70);

insert into exam values(null,'张飞',70,75,70);

insert into exam values(null,'赵云',90,65,95);

insert into exam values(null,'张三丰',85,79,null);

~查询表中所有学生的信息。

select \* from exam;

~查询表中所有学生的姓名和对应的语文成绩。

select name, chinese from exam;

~过滤表中重复数据。

select distinct chinese from exam;

~在所有学生分数上加10分特长分显示。

select name,chinese+10,math+10,english+10 from exam;

或者

select name,chinese+10 as 语文, math+10 as 数学, english+10 英语 from exam;###通过as关键词指定别名, as可以省略不写!

或者

select name,chinese+10 '语文', math+10 数学, english+10 英语 from exam;###通过as关键词指定别名, as可以省略不写!

~统计每个学生的总分。(ifnull(colname, 0))

select name, chinese+math+english 总分 from exam;

select name, chinese+math+ifnull(english, 0) 总分 from exam;

**使用where子句的查询**

**语法：**

SELECT \* | 列名 FROM tablename [WHERE where\_definition]

**练习：**

~查询姓名为关羽的学生成绩

select \* from exam where name='关羽';

~查询英语成绩大于90分的同学

select \* from exam where english>90;

~查询总分大于230分的所有同学

select name, chinese+math+ifnull(english, 0) 总分 from exam where chinese+math+ifnull(english, 0)>230;###在where子句中不能使用列别名!!!

~查询语文分数在 80~100之间的同学。

select \* from exam where chinese>80 and chinese <100;

或者

select \* from exam where chinese between 80 and 100;#包括80和100

~查询数学分数为75,76,77的同学。再查询分数不在这个范围内的同学

select \* from exam where math=75 or math=76 or math=77;

select \* from exam where !(math=75 or math=76 or math=77);

select \* from exam where not(math=75 or math=76 or math=77);

或者

select \* from exam where math in(75, 76, 77);

select \* from exam where math not in(75, 76, 77);

~查询数学分>70，语文分>80的同学。

select \* from exam where math>70 and chinese>80;

~查询所有姓张的学生成绩。

like: 模糊查询

百分号（%）: 匹配0个或者多个字符。 下划线（\_）:匹配一个字符



select \* from exam where name like '张%';

select \* from exam where name like '张\_';

select \* from exam where name like '%三%';

**排序查询**

**语法：**

SELECT column1, column2, column3.. FROM tablename order by column asc|desc;

Order by 指定排序的列，排序的列即可是表中的列名，也可以是select 语句后指定的列名。

Asc 升序(默认)、Desc 降序

ORDER BY 子句应位于SELECT语句的结尾。

**练习：**

~对数学成绩排序后输出。

select math from exam order by math;

~对总分排序按从高到低的顺序输出

select name, chinese+math+ifnull(english, 0) 总分 from exam order by 总分 desc;

~对姓张的学生总分排序输出

select name, chinese+math+ifnull(english, 0) 总分 from exam where name like '张%' order by 总分;

**聚合函数**

**语法：**

求符合条件的记录中指定列的记录数

select count(列名)… from tablename [WHERE where\_definition]

求符合条件的记录中指定列的和值

select sum(列名)… from tablename [WHERE where\_definition]

求符合条件的记录中指定列的平均值

select avg(列名)… from tablename [WHERE where\_definition]

求符合条件的记录中指定列的最大值

select max(列名)… from tablename [WHERE where\_definition]

求符合条件的记录中指定列的最小值

select min(列名)… from tablename [WHERE where\_definition]

**练习：**

~统计一个班级共有多少学生？

select count(\*) from exam;

~统计数学成绩大于75的学生有多少个？

select count(\*) from exam where math>75;

~统计总分大于230的人数有多少？

select count(\*) from exam where chinese+math+ifnull(english, 0)>230;

~统计一个班级数学总成绩？

select sum(math) 数学总成绩 from exam;

~统计一个班级语文、英语、数学三科成绩的总和

select sum(math)+sum(chinese)+sum(ifnull(english, 0)) 总成绩 from exam;

或者

select sum(chinese+math+ifnull(english, 0)) from exam;

~统计一个班级英语成绩平均分

select avg(english) from exam;

~求一个班级总分平均分？

select avg(chinese+math+ifnull(english, 0)) from exam;

~求班级总分最高分和最低分

select max(chinese+math+ifnull(english, 0)) from exam;

select min(chinese+math+ifnull(english, 0)) from exam;

**分组查询**

**语法：**

SELECT column1, column2. column3.. FROM tablename group by column having ...

**练习：**

执行下面的SQL，创建orders表并插入数据

create table orders(

id int,

product varchar(20),

price float

);

insert into orders(id,product,price) values(1,'小米手机',900);

insert into orders(id,product,price) values(2,'奥妙洗衣液',60);

insert into orders(id,product,price) values(3,'乐视TV',90);

insert into orders(id,product,price) values(4,'联想键盘',80);

insert into orders(id,product,price) values(5,'奥妙洗衣液',60);

insert into orders(id,product,price) values(6,'小米手机',900);

~对订单表中商品归类后，显示每一类商品的总价

select \* from orders group by product;

select count(\*) from orders group by product;

select sum(price) from orders group by product;

~查询购买了几类商品，并且每类总价大于100的商品

select product,sum(price) 总价 from orders group by product having sum(price)>100;

~查询单价小于100而总价大于100的商品的名称.

select \* from orders where price<100 group by product having sum(price) > 100;

**\*\*where和having的区别？**

\* where子句和having子句都可以进行过滤, 但是使用场景有所不同:

\* where子句对分组之前的数据进行过滤,不能使用聚合函数和列别名

\* having子句对分组之后的数据进行过滤,可以使用聚合函数和别名

\* 使用where子句的地方一般都可以用having替代，但是使用having的地方一般不能用where替代

## 6.6.数据库的备份与恢复

**备份数据库, 在cmd窗口中:**

备份命令: mysqldump -u用户名 -p 数据库名字 > 数据文件的位置

例如: mysqldump -uroot -p mydb1 > e:/1.sql

**恢复数据到数据库**

方式一: 在cmd中:

命令: mysql -u用户名 -p 数据库名字 < 数据文件的位置

例如: mysql -uroot -p mydb3 < e:/1.sql

方式二: 在mysql客户端中

命令: SOURCE 数据文件的位置

例如: source e:/1.sql

## 6.7.外键约束

外键：唯一标识其他表中的一条记录，用来通知数据库两张表字段之间的对应关系, 并让数据库帮我们维护这样关系的键就叫做外键

外键作用: 确保数据库数据的完整性和一致性

添加外键: 例如:foreign key(dept\_id) references dept(id)

**案例:**

create table dept(

id int primary key auto\_increment,

name varchar(20)

);

insert into dept values(null, '财务部');

insert into dept values(null, '人事部');

insert into dept values(null, '科技部');

insert into dept values(null, '销售部');

create table emp(

id int primary key auto\_increment,

name varchar(20),

dept\_id int,

foreign key(dept\_id) references dept(id)

);

insert into emp values(null, '张三', 1);

insert into emp values(null, '李四', 2);

insert into emp values(null, '老王', 3);

insert into emp values(null, '赵四', 4);

insert into emp values(null, '刘能', 4);

**要求：**

（1）执行上面的SQL语句，创建两张表（dept和emp），在创建时先不指定dept\_id为外键，尝试删除部门表中的某一个部门。

（2）将dept和emp表删除重建，在创建时指定dept\_id为外键，再次尝试删除部门表中的某一个部门。

## 6.8.多表设计

（1）1 - \*（一对多）：在多的一方添加列保存一的一方的主键来作为外键, 来保存两张表之间的关系

（2）1 - 1（一对一）：在任意一方添加列保存另一方的主键作为外键, 来保存两张表之间的关系

（3）\* - \*（多对多）：在一张第三方的表中分别保存两张表的主键作为外键, 来保存两张表之间的关系，可以把多对多的关系拆分成两个一对多的关系来理解

## 6.9.多表查询

**案例：**

create table dept(

id int primary key auto\_increment,

name varchar(20)

);

insert into dept values(null, '财务部');

insert into dept values(null, '人事部');

insert into dept values(null, '科技部');

insert into dept values(null, '销售部');

create table emp(

id int primary key auto\_increment,

name varchar(20),

dept\_id int

);

insert into emp values(null, '张三', 1);

insert into emp values(null, '李四', 2);

insert into emp values(null, '老王', 3);

insert into emp values(null, '刘能', 5);

**需求1：查询出部门表和员工表，同时列出部门信息和员工信息。**

select \* from dept,emp;（笛卡尔积查询）

select \* from dept, emp where emp.dept\_id=dept.id;

或者

select \* from dept inner join emp on emp.dept\_id=dept.id;

**需求2：查询出部门信息和部门所对应的员工信息，同时列出那些没有员工的部门**

select \* from dept left join emp on emp.dept\_id=dept.id;

**需求3：查询出部门信息和部门所对应的员工信息，同时列出那些没有部门的员工**

select \* from dept right join emp on emp.dept\_id=dept.id;

**需求4：查询出部门信息和部门所对应的员工信息, 同时列出没有员工的部门和那些没有部门的员工**

select \* from dept full join emp on emp.dept\_id=dept.id;###mysql不支持

**笛卡尔积查询：**

两张表相乘得出来的结果。如果左边表有m条记录，右边有n条记录，则查询出来的结果就是m\*n条。这些查询结果中包含大量错误的结果，通常不会使用这种查询。

**内连接查询：查询出左边表(dept)有且右边表(emp)也有的记录。**

select \* from emp inner join dept on dept.id=dept\_id;

**左外连接查询：在内连接查询的基础上，加上左边表有而右边表没有的记录**

\*\*查询出部门所对应的员工信息, 同时列出那些没有员工的部门

select \* from dept left join emp on dept.id=dept\_id;

**右外连接查询：在内连接查询的基础上，加上右边表有而左边表没有的记录。**

\*\*查询出部门所对应的员工信息, 同时列出那些没有部门的员工

select \* from dept right join emp on dept.id=dept\_id;

**全外连接查询：在内连接查询的基础上，加上左边表有而右边表没有的记录 和 右边表有而左边表没有的记录。**

\*\*查询出部门所对应的员工信息,同时列出那些没有员工的部门及些没有部门的员工

select \* from dept full join emp on dept.id=dept\_id;###mysql不支持全外连接查询

**使用union模拟全外连接查询：**

select \* from dept left join emp on dept.id=dept\_id

union

select \* from dept right join emp on dept.id=dept\_id;

## 6.10.作业练习

1. select from where group by having order by 这条语句中关键词执行的优先顺序

答：查询中用到的关键词主要包含六个，其中select和from是必须的，其他关键词是可选的，这六个关键词的执行顺序与sql语句的书写顺序并不是一样的，而是按照下面的顺序来执行：

from--where--group by--having--select--order by,

from:需要从哪个数据表检索数据

where:过滤表中数据的条件

group by:如何将上面过滤出的数据分组

having:对上面已经分组的数据进行过滤的条件

select:查看结果集中的哪个列，或列的计算结果

order by :按照什么样的顺序来查看返回的数据

1. 两台电脑，我的装了个MySQL。给你用户名和密码，你怎么登陆我的？

答：通过命令：mysql -u用户名 -p密码 -h ip 即可登陆

3.删除一张表中所有数据的方式？

答：delete from 表名;

4.给出成绩全部合格的学生信息(包含姓名、课程、分数)，注：分数在60以上评为合格？

表名：StudentScore

|  |  |  |
| --- | --- | --- |
| name | course | score |
| 张三 | 语文 | 81 |
| 张三 | 数学 | 75 |
| 李四 | 语文 | 56 |
| 李四 | 数学 | 90 |
| 王五 | 语文 | 81 |
| 王五 | 数学 | 100 |
| 王五 | 英语 | 49 |

答：

SELECT \* from studentscore WHERE name = (SELECT name FROM studentscore  GROUP BY name HAVING min(score)>=60 );

1. 给出平均进价在2元以下的商品名称？

表名：Product



答：SELECT name from product GROUP BY name HAVING avg(price)<2;

1. 给出高考总分在600以上的学生准考证号？

表名：ScoreInfo



答：SELECT no from scoreinfo  GROUP BY no HAVING sum(score)>600;

1. (1)查询出该俱乐部里男性会员和女性会员的总数？

(2)按照性别区分并按照年龄排序？

表名：club

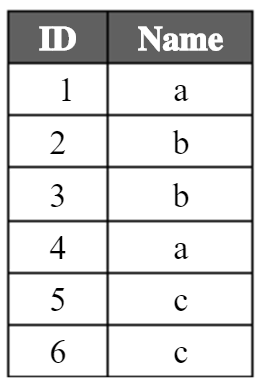


答：（1）Select count (\*) from club group by gender;

（2）select \* from club order by gender,age;

1. 执行一个删除语句，当Name列上有相同时,只保留ID这列上值小的

表名：team



答：delete from team where id not in(select min(id) from team group by name);

或者：delete from team where id not in (select i from (select min(id) i from team group by name) t);

1. 查询出“张”姓学生中平均成绩大于75分的学生信息？

表名：student



答：

select \* from student where name in(select name from student where name like '张%' group by name having avg(score)>75);

10.有以下两张表：

Table\_Student有三个字段Code、Name、Age，其中Code为主键；

Table\_Result有三个字段Code、Class、Score，其中Code为主键。两表如下：

|  |  |  |
| --- | --- | --- |
| Code | Name | Age |
| 97001 | 张三 | 22 |
| 97002 | 赵四 | 21 |
| 97003 | 张飞 | 20 |
| 97004 | 李五 | 22 |

|  |  |  |
| --- | --- | --- |
| Code | Class | Score |
| 97001 | 数学 | 80 |
| 97002 | 计算机 | 59 |
| 97003 | 计算机 | 60 |
| 97004 | 数学 | 55 |

请按以下要求写出相应SQL语句

1. 找出所有姓张的学生，并按年龄从小到大排列

select \* from Table\_Student where name like ‘张%’ order by age;

1. 取出计算机科考成绩不及格的学生

select \* from Table\_Student a,Table\_Result b where a.code=b.code and b.class=’计算机’ and b.score<60;

1. 通过等值连接，取出Name、Class、Score

select a.name,b.class,b.score from Table\_Student a,Table\_Result b where a.code=b.code

或者

select a.name,b.class,b.score from Table\_Student a inner join Table\_Result b on a.code=b.code;

1. 在Table\_Student表中增加一条学生记录（学号：97005 姓名:赵六 年龄：20）

insert into Table\_Student(code,name,age) values(‘97005’,’赵六’,20);

1. 李五的年龄记录错了，应该是21，根据主键进行更新

update Table\_Student set age=21 where name=’李五’;

1. 删除Table\_Student中没有考试成绩的学生记录，请使用not in条件

delete from Table\_Student where code not in(select a.code from Table\_Student a,Table\_Result b where a.code=b.code);

11.where和having的区别

答： where子句对分组之前的数据进行过滤,不能使用聚合函数和列别名

having子句对分组之后的数据进行过滤,可以使用聚合函数和别名

12.数据表定义如下：

学生表（student）（学号，姓名，年龄，所在系）

课程表（course）（课程编号，课程名，学分）

关系表（sc）(学号、课程编号、成绩)

1. 查询出选修了“数据库”的学生姓名。

select 姓名 from student s,course c,sc where s.学号=sc.学号 and sc.课程编号=c.课程编号 and c.课程名=’数据库’;

1. 查询平均成绩大于85分的学生姓名和平均成绩。

select 姓名,avg(成绩) from student s,sc where s.学号=sc.学号 group by sc.no having avg(成绩)>85;

1. 添加一个课程信息（“java”，“1009”，2）

insert into course values(‘1009’,’java’,2);

1. 检索考试成绩有不及格的学生姓名及对应课程和成绩

select 姓名,课程名,成绩 from student s,course c,sc where s.学号=sc.学号 and sc.课程编号=c.课程编号 and 成绩<60;

（5）检索选修了“大数据”课程的学生的姓名及其成绩,查询结果按分数的降序排序.

select 姓名,成绩 from student s,course c,sc where s.学号=sc.学号 and sc.课程编号=c.课程编号 and cname='大数据' order by 成绩 desc;

（6）查询1001课程比1002课程成绩高的所有学生的学号

select s1.学号 from sc s1,sc s2 where s1.学号=s2.学号 and s1.课程编号=1001 and s2.课程编号=1002 and s1.成绩>s2.成绩;