**MySQL基础语法**

# 回顾

## 四种方法引用类型

|  |  |
| --- | --- |
| 方法引用类型 | 语法 |
| 静态方法引用 | 类名::静态方法名 |
| 对象方法引用 | 对象名::成员方法 |
| 类构造器引用 | 类名::new |
| 数组构造器引用 | 类名[]::new |

## 方法引用写法

|  |  |  |  |
| --- | --- | --- | --- |
| 接口中的抽象方法 | 引用的方法 | Lambda表达式 | 方法引用 |
| int calc(int num) | **int Math.abs(int x)** | **num -> Math.abs(num)** | **Math::abs** |
| void accept(String s) | **void println(String s)** | **s -> System.out.println(s)** | **System.out::println** |
| Car makeCar(String name) | **new Car(String name)** | **name -> new Car(name)** | **Car::new** |
| int[] buildArray(int length) | **new int[length]** | **length -> new int[length]** | **int[]::new** |

## 得到流对象

|  |  |
| --- | --- |
| java.util.Collection接口两个默认方法 | 说明 |
| Stream stream() | 得到集合的串行流 |
| Stream parallelStream() | 得到集合的并行流 |

## Stream流中方法的作用

|  |  |  |
| --- | --- | --- |
| 方法名 | 是否终结方法 | 方法作用 |
| forEach | 是 | 对流中的元素进行遍历 |
| count | 是 | 统计流中元素的个数 |
| filter | 否 | 过滤，保留符合条件的元素 |
| limit | 否 | 取得前面的n个元素 |
| skip | 否 | 跳过前面的n个元素 |
| map | 否 | 将流中每个元素都转成另一种元素 |
| concat | 否 | 拼接流，返回一个新的流 |

# 学习目标

1. 能够理解数据库的概念
2. 能够安装MySQL数据库
3. 能够启动,关闭及登录MySQL
4. 能够使用SQL语句操作数据库
5. 能够使用SQL语句操作表结构
6. 能够使用SQL语句进行数据的添加修改和删除的操作
7. 能够使用SQL语句简单查询数据
8. 能够使用SQL语句进行条件查询

# 数据库的介绍

## 数据库概述

### 数据的存储方式

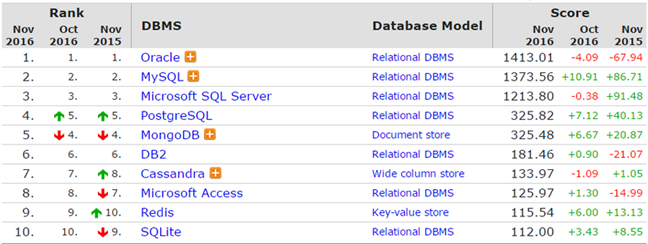
|  |  |  |
| --- | --- | --- |
| 存储位置 | 优点 | 缺点 |
| 内存 | 运行速度快 如：new Student() | 关机数据就丢失 |
| 文件 | IO流，可以永久保存数据 | 读取和写入速度慢  数据管理和维护不方便 |
| 数据库 | 可以永久保存大量的数据  数据管理和维护方便，统一使用SQL语句 | 要钱  占用资源，再安装数据库 |

### 什么是数据库

存储数据的仓库. 其本质是一个文件系统，数据库按照特定的格式将数据存储起来，用户可以对数据库中的数

据进行增加，修改，删除及查询操作。 增删改查(CRUD Create Retrieve Update Delete)

## 常见数据库排行榜



* **MySQL**：开源免费的数据库，小型的数据库，已经被Oracle收购了。MySQL6.x版本也开始收费。后来Sun公司收购了MySQL，而Sun公司又被Oracle收购



* **Oracle**：收费的大型数据库，Oracle公司的产品。Oracle收购SUN公司，收购MySQL。



* **DB2** ：IBM公司的数据库产品,收费的。常应用在银行系统中。
* **SQL Server**：MicroSoft 公司收费的中型的数据库。C#、.net等语言常使用。



* **SQLite**: 嵌入式的小型数据库，应用在手机端。

### 为什么选择MySQL

1. 开源、免费
2. 功能足够强大，足以应付web应用开发（最高支持千万级别的并发访问）

# 数据库的安装与卸载

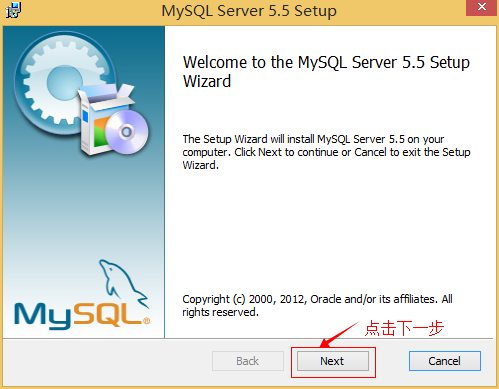
MySQL是一个运行在服务器上的软件

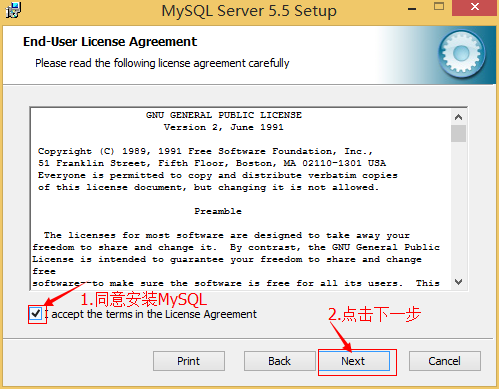
安装两个过程：

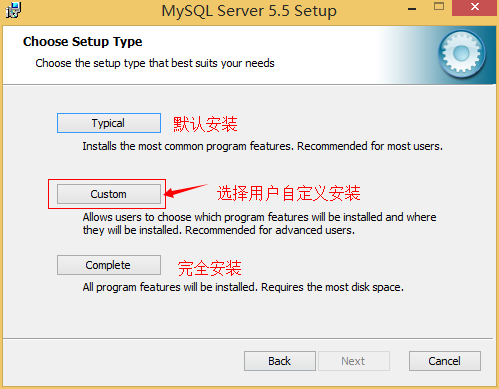
1. 文件的解压和复制，过程很快
2. 服务器的配置(重点)
3. 超级用户管理员root，密码root

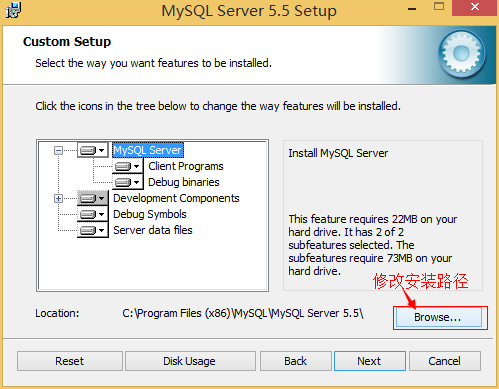
## 数据库的安装

* 1. 打开下载的mysql安装文件双击解压缩，运行“mysql-5.5.40-win64.msi”。

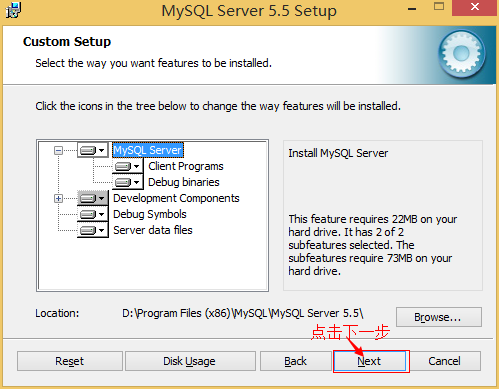
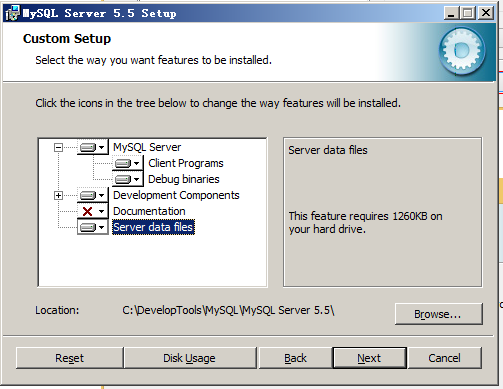
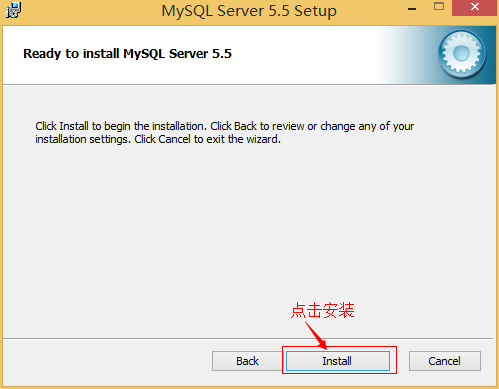
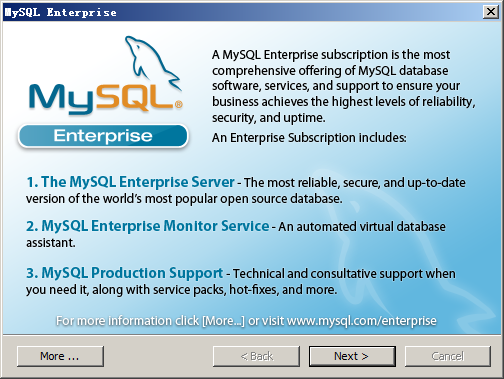
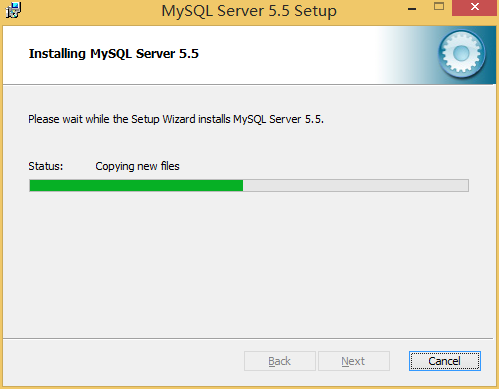
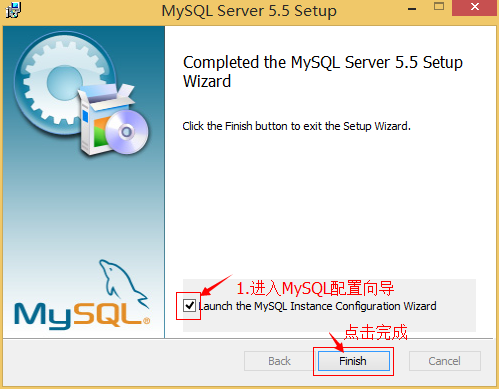
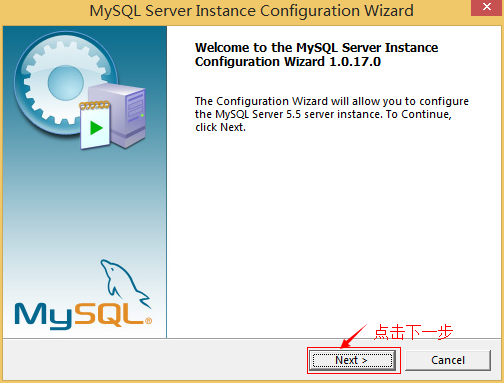
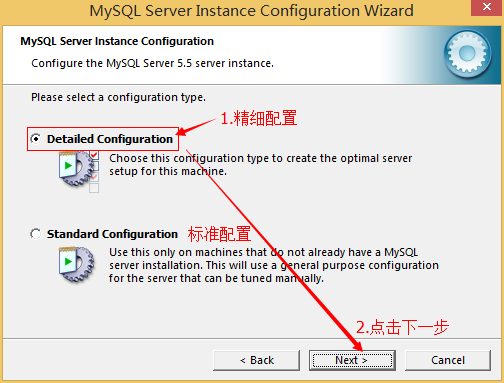
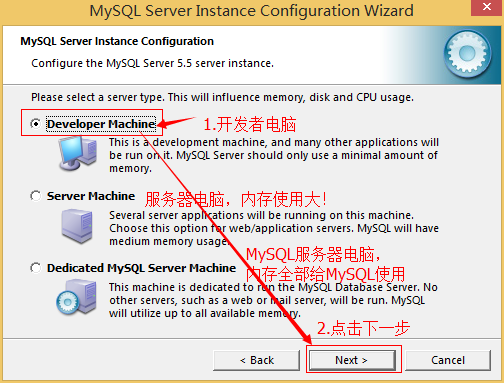
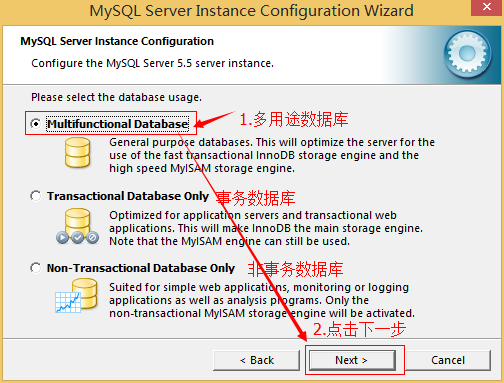
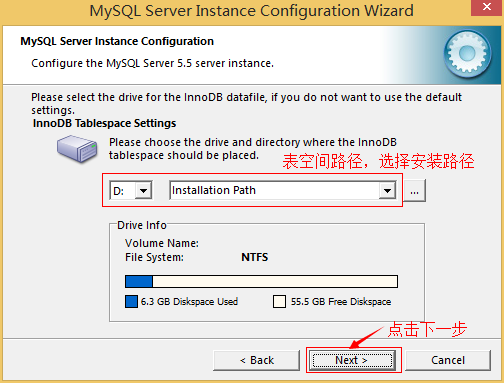
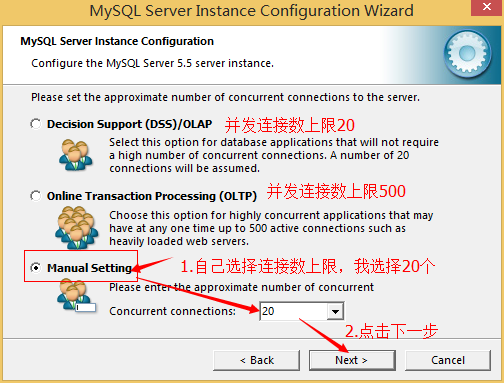
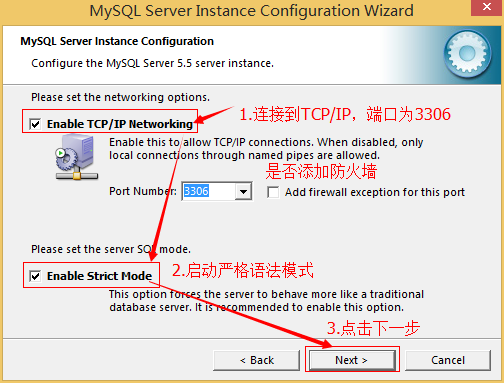
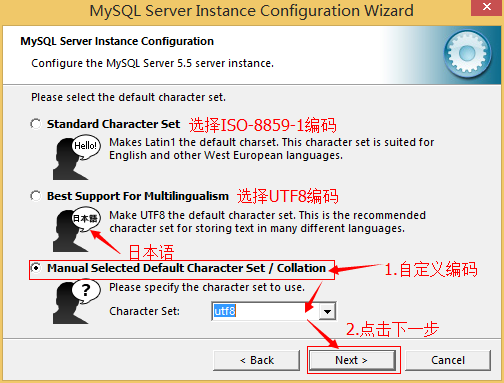
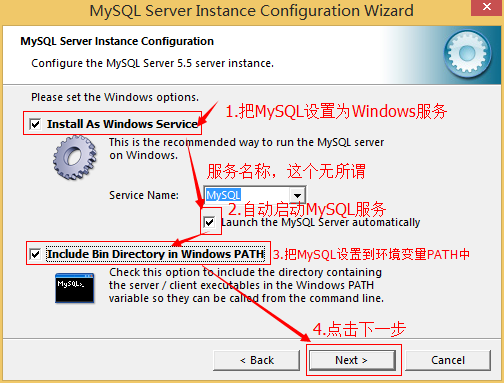
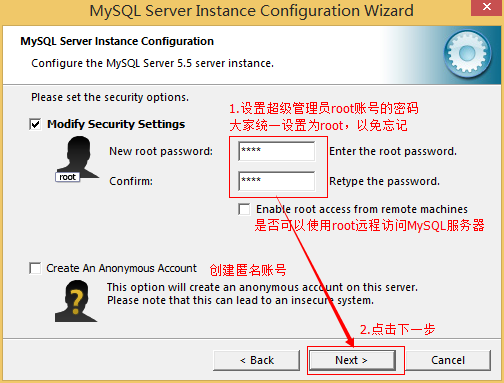


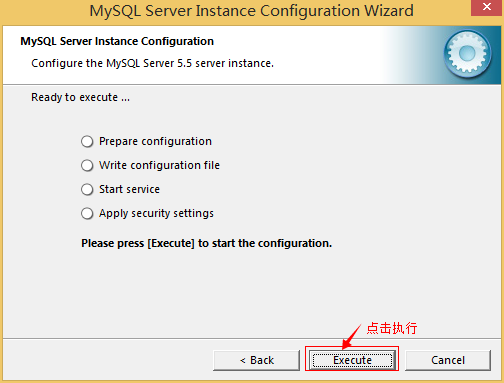


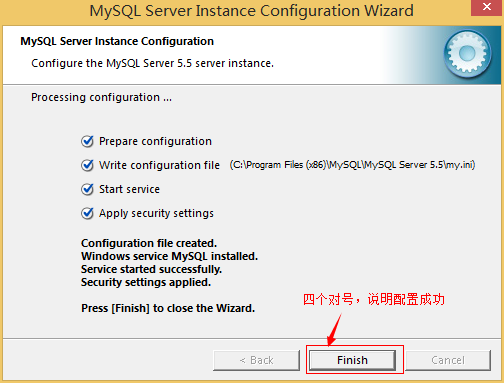
* 1. 选择安装类型，有“Typical（默认）”、“Complete（完全）”、“Custom（用户自定义）”三个选项，选择“Custom”，按“next”键继续。   
     

3. 点选“Browse”，手动指定安装目录。 

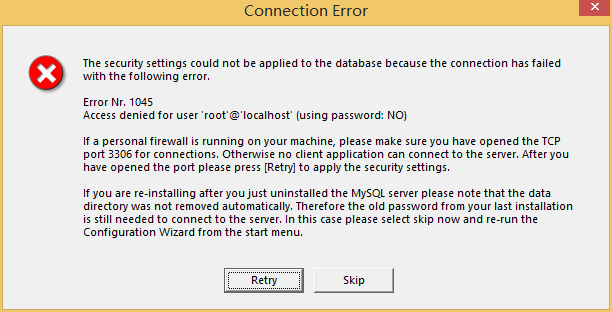
4. 填上安装目录，我的是“d:\Program Files (x86)\MySQL\MySQL Server 5.0”，按“OK”继续。 

1. 确认一下先前的设置，如果有误，按“Back”返回重做。按“Install”开始安装。     
2. 正在安装中，请稍候，直到出现下面的界面, 则完成MYSQL的安装 
   * 数据库安装好了还需要对数据库进行配置才能使用 MYSQL的配置
3. 安装完成了，出现如下界面将进入mysql配置向导。 
4. 选择配置方式，“Detailed Configuration（手动精确配置）”、“Standard Configuration（标准配置）”，我们选择“Detailed Configuration”，方便熟悉配置过程。 
5. 选择服务器类型，“Developer Machine（开发测试类，mysql占用很少资源）”、“Server Machine（服务器类型，mysql占用较多资源）”、“Dedicated MySQL Server Machine（专门的数据库服务器，mysql占用所有可用资源）”  
    
6. 选择mysql数据库的大致用途，“Multifunctional Database（通用多功能型，好）”、“Transactional Database Only（服务器类型，专注于事务处理，一般）”、“Non-Transactional Database Only（非事务处理型，较简单，主要做一些监控、记数用，对MyISAM数据类型的支持仅限于non-transactional），按“Next”继续。  
     
7. 选择网站并发连接数，同时连接的数目，“Decision Support(DSS)/OLAP（20个左右）”、“Online Transaction Processing(OLTP)（500个左右）”、“Manual Setting（手动设置，自己输一个数）”。 
8. 是否启用TCP/IP连接，设定端口，如果不启用，就只能在自己的机器上访问mysql数据库了，在这个页面上，您还可以选择“启用标准模式”（Enable Strict Mode），这样MySQL就不会允许细小的语法错误。如果是新手，建议您取消标准模式以减少麻烦。但熟悉MySQL以后，尽量使用标准模式，因为它可以降低有害数据进入数据库的可能性。按“Next”继续 
9. 就是对mysql默认数据库语言编码进行设置（重要），一般选UTF-8，按 “Next”继续。 
10. 选择是否将mysql安装为windows服务，还可以指定Service Name（服务标识名称），是否将mysql的bin目录加入到Windows PATH（加入后，就可以直接使用bin下的文件，而不用指出目录名，比如连接，“mysql.exe -uusername -ppassword;”就可以了，不用指出mysql.exe的完整地址，很方便），我这里全部打上了勾，Service Name不变。按“Next”继续。 
11. 询问是否要修改默认root用户（超级管理）的密码。“Enable root access from remote machines（是否允许root用户在其它的机器上登陆，如果要安全，就不要勾上，如果要方便，就勾上它）”。最后“Create An Anonymous Account（新建一个匿名用户，匿名用户可以连接数据库，不能操作数据，包括查询）”，一般就不用勾了，设置完毕，按“Next”继续。 
12. 确认设置无误，按“Execute”使设置生效，即完成MYSQL的安装和配置。



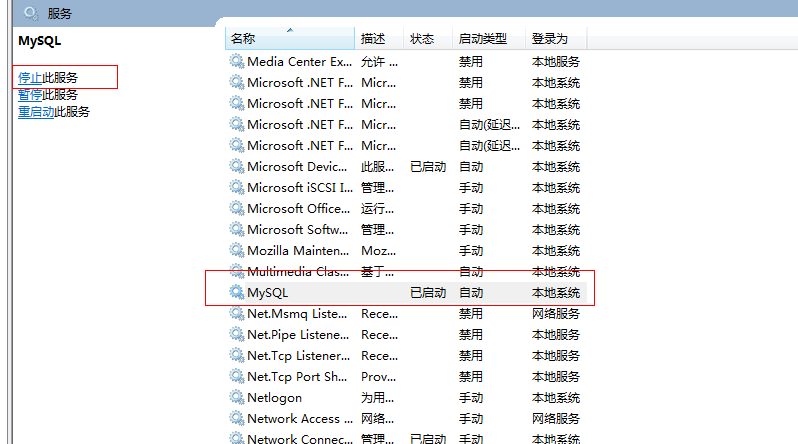
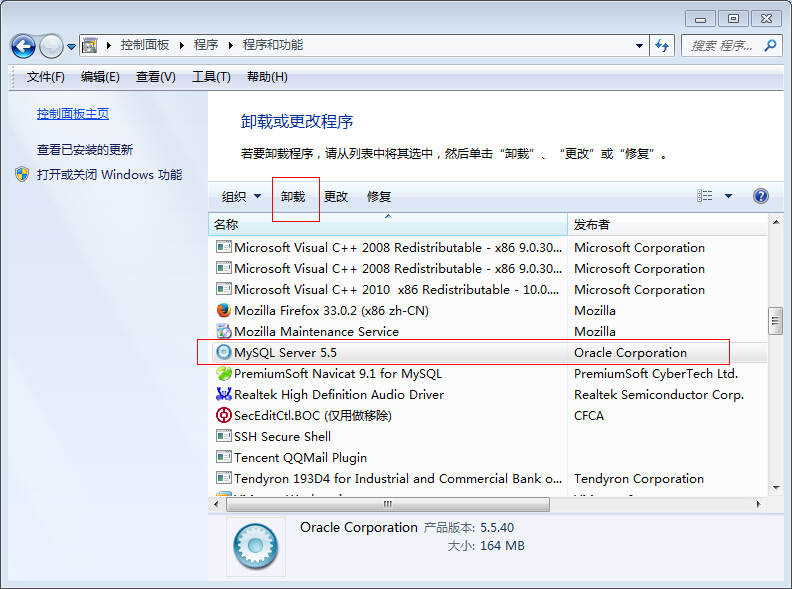
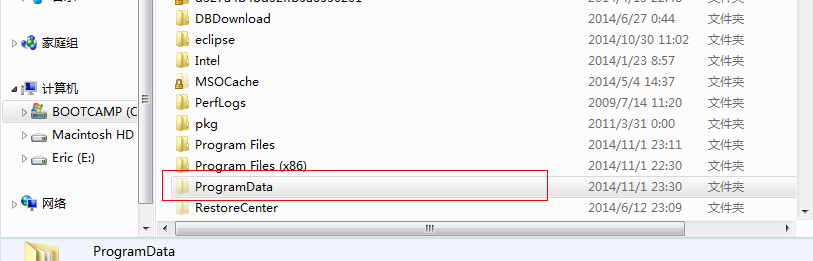
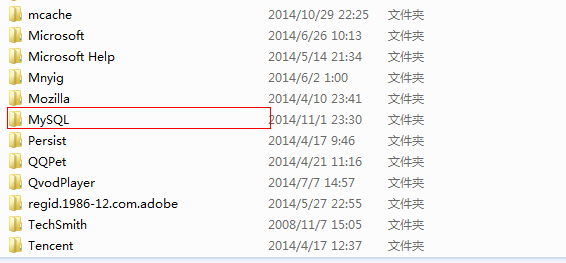


注意：设置完毕，按“Finish”后有一个比较常见的错误，就是不能“Start service”，一般出现在以前有安装mysql的服务器上，解决的办法，先保证以前安装的mysql服务器彻底卸载掉了；不行的话，检查是否按上面一步所说，之前的密码是否有修改，照上面的操作；如果依然不行，将mysql安装目录下的data文件夹备份，然后删除，在安装完成后，将安装生成的 data文件夹删除，备份的data文件夹移回来，再重启mysql服务就可以了，这种情况下，可能需要将数据库检查一下，然后修复一次，防止数据出错。



* + 解决方法：卸载MySQL,重装MySQL

## 数据库的卸载

1. 停止window的MySQL服务。 找到“控制面板”-> “管理工具”-> “服务”，停止MySQL后台服务。 
2. 卸载MySQL安装程序。找到“控制面板”-> "程序和功能"，卸载MySQL程序。 
3. 删除MySQL安装目录下的所有文件。
4. 删除c盘ProgramDate目录中关于MySQL的目录。路径为：C:\ProgramData\MySQL(是隐藏文件,需要显示出来)  
     

## 安装好的MySQL目录结构

|  |  |
| --- | --- |
| MySQL的目录结构 | 描述 |
| bin<目录> | 要使用到的所有的exe可执行文件 |
| MySQLInstanceConfig.exe | 数据库配置向导文件，如果在配置过程中出了问题或者需要重新配置，可以再次运行这个程序 |
| data<目录> | 自带的核心的数据库，基础数据库。 |
| my.ini | 配置文件 |
| c:\ProgramData\MySQL\MySQL Server 5.5\data\ | 我们创建的数据库所在的目录 |

# 数据库服务的启动与登录

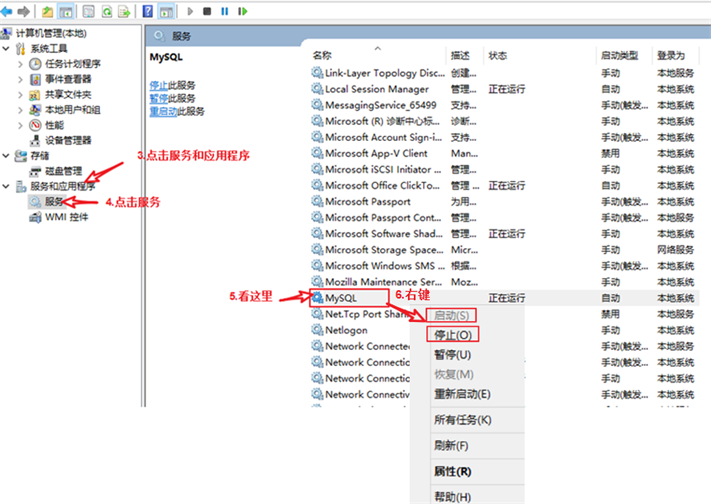
MySQL服务器启动方式有两种：

1. 通过DOS命令行的方式启动
2. 在服务器中通过服务界面启动

## Windows服务方式启动

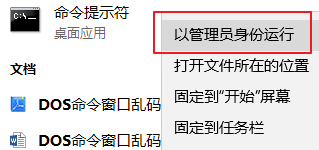
### 操作步骤：





## DOS命令方式启动

### 操作步骤：



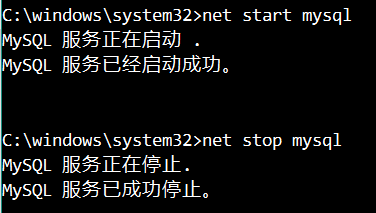
* 启动MYSQL:

|  |
| --- |
| net start mysql |

* 停止MYSQL:

|  |
| --- |
| net stop mysql |

### 执行效果



## 控制台连接数据库

MySQL是一个需要账户名密码登录的数据库，登陆后使用，它提供了一个默认的root账号，使用安装时设置的密码即可登录

### 登录格式1：

|  |
| --- |
| mysql -u用户名 -p密码 |

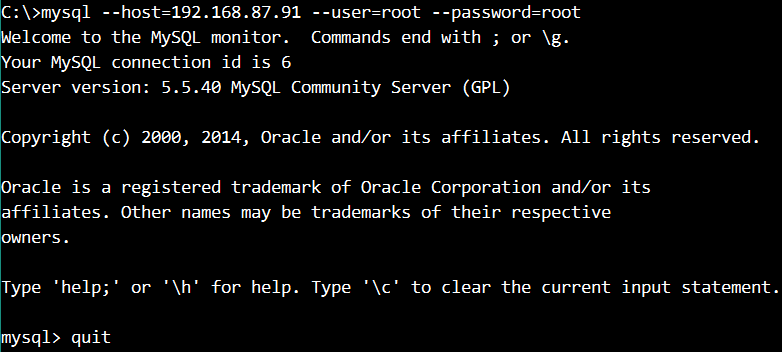
### 登录格式2：

|  |
| --- |
| mysql -hip地址 -u用户名 -p密码 |

### 登录格式3：

|  |
| --- |
| mysql --host=ip地址 --user=用户名 --password=密码 |

### 分别使用三种方式登录



### 退出MySQL：

|  |
| --- |
| quit或exit |

## SQLyog图形化工具

SQLyog是业界著名的Webyog公司出品的一款简洁高效、功能强大的图形化MySQL数据库管理工具。使用SQLyog可以快速直观地让您从世界的任何角落通过网络来维护远端的MySQL数据库



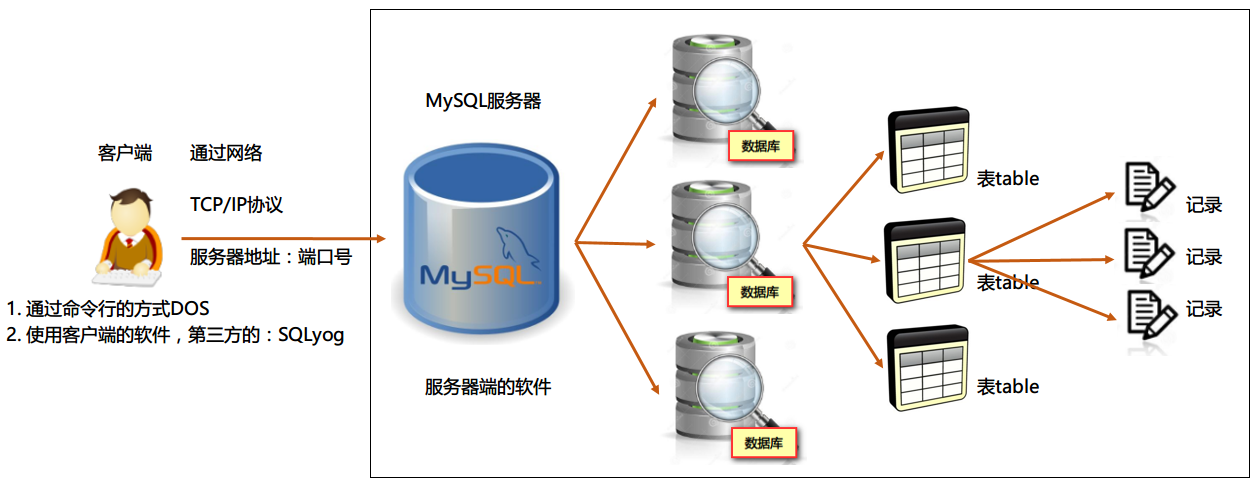
### 使用SQLyog登录数据库



## 数据库管理系统、数据库和表的关系

数据库管理程序(DBMS)可以管理多个数据库，一般开发人员会针对每一个应用创建一个数据库。为保存应用中实体的数据，一般会在数据库创建多个表，以保存程序中实体的数据。

数据库管理系统、数据库和表的关系如图所示：



### 结论：

1. 一个服务器中包含多个数据库
2. 一个数据库中包含多张表
3. 一张表中包含多条记录

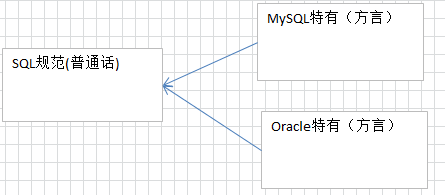
# SQL的概念

## 什么是SQL

SQL: Structured Query Language 结构化查询语言

## SQL作用

1. 是一种数据库管理语言的规范，所有的数据库都支持这种语言。如：MySQL或Oracle
2. 用来管理和维护数据库的语言，所有的数据库SQL都大同小异。也有一些区别。



## SQL语句分类

1. Data Definition Language - DDL 数据定义语言，如：建库，建表
2. Data Manipulation Language - DML 数据操纵语言，如：对记录增删改的操作
3. Data Query Language - DQL 数据查询语言，如：对记录进行查询
4. Data Control Language - DCL 数据控制语言，如：对用户进行管理

## MySQL的语法

1. 所有的SQL语句以分号结尾
2. SQL是不区分大小写，关键字一般使用大写。
3. 3种注释：

|  |  |
| --- | --- |
| 注释的语法 | 说明 |
| -- 空格 | 单行注释 |
| /\* \*/ | 多行注释 |
| # | 单行注释，不推荐，只在mysql中支持 |

# DDL操作数据库

## 创建数据库

### 创建数据库的几种方式

* 创建数据库

|  |
| --- |
| CREATE DATABASE 数据库名; |

* 判断数据库是否已经存在，不存在则创建数据库

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS 数据库名; |

* 创建数据库并指定字符集

|  |
| --- |
| CREATE DATABASE 数据库名 CHARACTER SET 字符集; |

### 具体操作：

-- 直接创建数据库db1

create database db1;

-- 显示创建的sql语句

show create database db1;

-- 按Tab可以进行关键字的补全

-- 判断是否存在，如果不存在则创建数据库db2

create database if not exists db1;

create database if not exists db2;

-- 创建数据库并指定字符集为gbk

create database db3 character set gbk;

show create database db3;

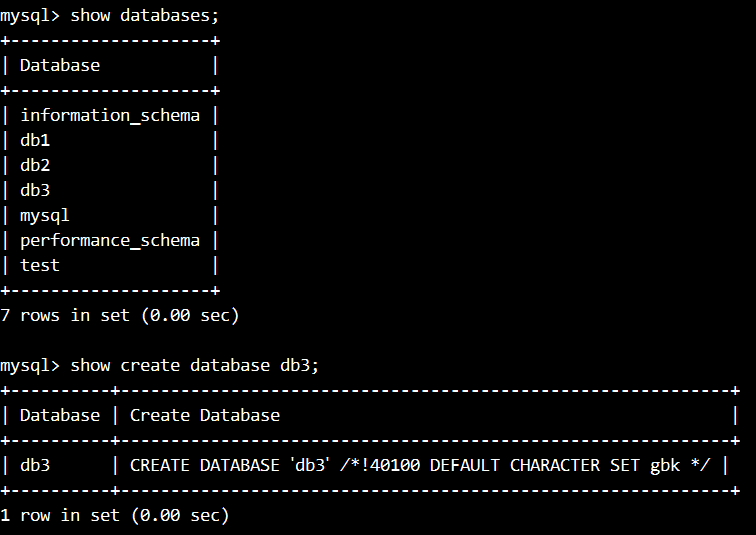
## 查看数据库

-- 查看所有的数据库

show databases;

-- 查看某个数据库的定义信息

show create database db1;



## 修改数据库

### 修改数据库默认的字符集

|  |
| --- |
| ALTER DATABASE 数据库名 CHARACTER SET 字符集; |

### 具体操作：

-- 将db3数据库的字符集改成utf8

alter database db3 character set utf8;

## 删除数据库

### 删除数据库的语法

|  |
| --- |
| DROP DATABASE 数据库名; |

### 具体操作：

-- 删除db2数据库

drop database db2;

## 使用数据库

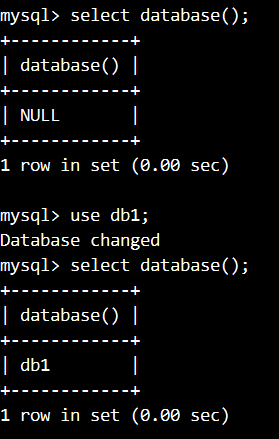
### 查看正在使用的数据库

|  |
| --- |
| SELECT DATABASE(); -- 调用一个方法 |

### 使用/切换数据库

|  |
| --- |
| USE 数据库名; |

### 具体操作：



### 面试题：

在MySQL数据库软件中，有如下三个数据库：

 登录数据库之后，输入语句：select database test2; 运行结果是什么?

答：这条语句错误，如果要使用某个数据库：use test2;

# DDL操作表结构

前提先使用某个数据库: use 数据库名

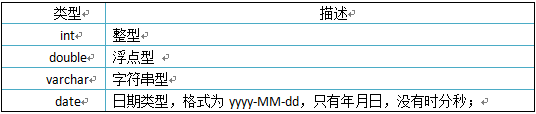
## 创建表

### 创建表的格式

|  |
| --- |
| CREATE TABLE 表名 ( 字段名1 字段类型1,  字段名2 字段类型2  ); |

## MySQL数据类型

### 常使用的数据类型如下：



### 常用数据类型说明

|  |  |  |
| --- | --- | --- |
| 分类 | 类型名称 | 类型说明 |
| 整数 | tinyInt | 微整型：很小的整数(占8位二进制) 255 |
| smallint | 小整型：小的整数(占16位二进制) 16K |
| mediumint | 中整型：中等长度的整数(占24位二进制) |
| int(integer) | 整型：整数类型(占32位二进制) |
| bigint | 大整型：占64位二进制 |
| 小数 | float | 单精度浮点数，占4个字节 |
| double | 双精度浮点数，占8个字节 |
| 日期 | time | 表示时间类型 |
| date | 表示日期类型 |
| datetime | 同时可以表示日期和时间类型 |
| 字符串 | char(m) | 固定长度的字符串，无论使用几个字符都占满全部，M为0~255之间的整数 |
| varchar(m) | 可变长度的字符串，使用几个字符就占用几个，M为0~65535之间的整数 |

### 具体操作:

-- 创建student表包含id,name,birthday字段

create table student (

id int,

name varchar(20),

birthday date

);

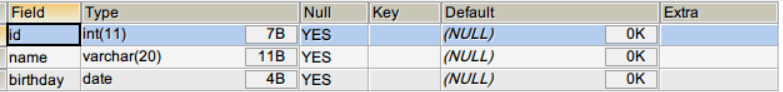
## 查看表

### 查看某个数据库中的所有表

|  |
| --- |
| SHOW TABLES; |

### 查看表结构

|  |
| --- |
| DESC 表名; |



### 查看创建表的SQL语句

|  |
| --- |
| SHOW CREATE TABLE 表名; |

### 具体操作：

-- 查看mysql数据库中的所有表

show tables;

-- 查看student表的结构

desc student;

-- 查看student的创建表SQL语句

show create table student;

-- 生成

CREATE TABLE `student` (

`id` int(11) DEFAULT NULL,

`name` varchar(20) DEFAULT NULL,

`birthday` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8

-- ` 引用起来的目的是为了避免关键字的冲突

-- 创建一个表，表名叫create

create table a (a int);

create table `create`(a int);

desc a;

desc `create`;

## 快速创建一个表结构相同的表

### 语法

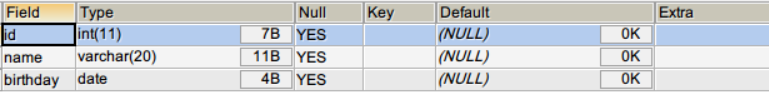
|  |
| --- |
| CREATE TABLE 新表名 LIKE 旧表名; |

### 具体操作：

-- 创建s1表，s1表结构和student表结构相同

create table s1 like student;

desc s1;



## 删除表

### 直接删除表

|  |
| --- |
| DROP TABLE 表名; |

### 判断表是否存在，如果存在则删除表

|  |
| --- |
| DROP TABLE IF EXISTS 表名; |

### 具体操作：

-- 直接删除表s1表

drop table s1;

-- 判断表是否存在并删除s1表

drop table if exists s1;

## 修改表结构

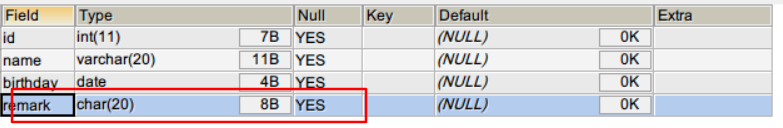
### 添加表列ADD

|  |
| --- |
| ALTER TABLE 表名 ADD 列名 类型; |

-- 为学生表添加一个新的字段remark,类型为char(20)

alter table student add remark char(20);

desc student;



### 修改列类型MODIFY

|  |
| --- |
| ALTER TABLE 表名 MODIFY列名 新的类型; |

-- 将student表中的remark字段的改成varchar(100)

alter table student modify remark varchar(100);

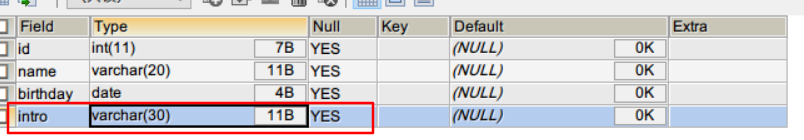
### 修改列名 CHANGE

|  |
| --- |
| ALTER TABLE 表名 CHANGE 旧列名 新列名 类型; |

-- 将student表中的remark字段名改成intro，类型varchar(30)

alter table student change remark intro varchar(30);

desc student;



### 删除列 DROP

|  |
| --- |
| ALTER TABLE 表名 DROP 列名; |

-- 删除student表中的字段intro

alter table student drop intro;

### 修改表名RENAME

|  |
| --- |
| RENAME TABLE 表名 TO 新表名; |

-- 将学生表student改名成student2

rename table student to student2;

### 修改字符集character set

|  |
| --- |
| ALTER TABLE 表名 character set 字符集; |

-- 将student2表的编码修改成gbk

show create table student2;

CREATE TABLE `student2` (

`id` int(11) DEFAULT NULL,

`name` varchar(20) DEFAULT NULL,

`birthday` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8

alter table student2 character set gbk;

CREATE TABLE `student2` (

`id` int(11) DEFAULT NULL,

`name` varchar(20) CHARACTER SET utf8 DEFAULT NULL,

`birthday` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=gbk

# DML操作表中的数据

## 插入记录

### 插入全部字段

|  |
| --- |
| INSERT INTO 表名 VALUES (值1, 值2, 值3…); |

### 插入部分数据

|  |
| --- |
| INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...); |

* + 注：没有添加数据的字段会使用NULL
* 具体操作:

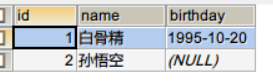
-- 向表中插入所有字段，不写字段名

-- 在SQL中字符串使用单引号，mysql中可以使用双引号，日期也使用单引号

insert into student values(1, '白骨精', '1995-10-20');

-- 插入部分数据，往学生表中添加 id, name数据

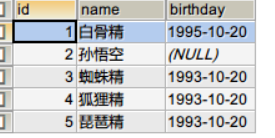
insert into student (id,name) values (2,'孙悟空');



### 插入多条记录MySQL

|  |
| --- |
| INSERT INTO 表名 values (值1,值2),(值1,值2),(值1,值2); |

insert into student values(3, '蜘蛛精', '1993-10-20'),(4, '狐狸精', '1993-10-20'),(5, '琵琶精', '1993-10-20');



### 插入记录小结：

|  |
| --- |
| 1. 插入的数据应与字段的数据类型相同 2. 数据的大小应在列的规定范围内，例如：不能将一个长度为80的字符串加入到长度为40的列中。 3. 在values中列出的数据位置必须与被加入的列的排列位置相对应。在mysql中可以使用value，但不建议使用，功能与values相同。 4. 字符和日期型数据应包含在单引号中。MySQL中也可以使用双引号做为分隔符。 5. 不指定列或使用null，表示插入空值。 |

## 蠕虫复制

### 什么是蠕虫复制：

从一张已经存在的表中复制记录到当前表中

### 语法格式：

* 将表名2中的所有的列复制到表名1中

|  |
| --- |
| INSERT INTO 表名1 SELECT \* FROM 表名2; |

* 只复制部分列

|  |
| --- |
| INSERT INTO 表名1(列1, 列2) SELECT 列1, 列2 FROM student; |

### 具体操作:

-- 创建student2表，student2结构和student表结构一样

-- 将student表中的数据添加到student2表中

insert into student2 select \* from student;

-- 如果只想复制student表中id,name字段数据到student2表中，两张表都写出相应的列名

insert into student2(id,name) select id,name from student;



## 更新表记录

### 不带条件修改数据

如果不指定条件，则是修改所有的记录

|  |
| --- |
| UPDATE 表名 SET 字段名=值; |

### 带条件修改数据

|  |
| --- |
| UPDATE 表名 SET 字段名=值 WHERE 字段名=值; |

### 具体操作：

-- 不带条件修改数据，将所有的性别改成女

-- 添加性别和地址

alter table student add gender char(1), add address varchar(30);

select \* from student;

update student set gender = '女';

-- 带条件修改数据，将id号为2的学生性别改成男

update student set gender = '男' where id = 2;

-- 一次修改多个列，把id为3的学生，修改生日：2000-11-11，address改成北京

update student set birthday = '2000-11-11', address = '北京' where id = 3;

## 删除表记录

### 不带条件删除数据

不带条件删除表中所有的记录

|  |
| --- |
| DELETE FROM 表名; |

### 带条件删除数据

|  |
| --- |
| DELETE FROM 表名 WHERE 字段名=值; |

### 使用truncate删除表中所有记录

|  |
| --- |
| TRUNCATE TABLE 表名; |

### truncate和delete的区别：

1. delete是删除表的记录，不会影响表的结构
2. truncate是删除结构和记录，再创建一个与之前的表结构相同的空表

### 具体操作：

-- 不带条件删除数据,删除表中的所有数据

delete from student;

insert into student (id,name,birthday) select id,name,birthday from student2;

truncate student;

-- 删除了表结构

drop table student;

# DQL查询表中的数据

查询不会对数据库中的数据进行修改.只是一种显示数据的方式

用户在客户端编写SQL语句 🡪 MySQL服务器 🡪 在服务器上执行SQL 🡪 将执行结果发给客户端

|  |
| --- |
| SELECT 列名 FROM 表名 [WHERE 条件表达式] |
| 1. SELECT 命令可以读取一条或者多条记录。 2. 你可以使用星号（\*）来代替其他字段，SELECT语句会返回表的所有字段数据 3. 你可以使用 WHERE 语句来包含任何条件。 |

## 简单查询

### 查询表所有行和列的数据

* 使用\*表示所有列

|  |
| --- |
| SELECT \* FROM 表名; |

-- 查询所有的学生：

select \* from student;

### 查询指定列

* 查询指定列的数据,多个列之间以逗号分隔

|  |
| --- |
| SELECT 字段名1, 字段名2, 字段名3, ... FROM 表名; |

-- 查询student表中的id 和 name列

select id,name from student;

## 指定列的别名进行查询

### 使用关键字

* 使用别名的好处： 可以查询结果指定其它的列名

### 语法：

* 对列指定别名

|  |
| --- |
| SELECT 字段名1 AS 别名, 字段名2 AS 别名... FROM 表名; |

* 对列和表同时指定别名

|  |
| --- |
| SELECT 字段名1 AS 别名, 字段名2 AS 别名... FROM 表名 AS 表别名; |

### 具体操作：

-- 查询student表中id 和 name 列，id为编号，name列的别名为”姓名”

select id as 编号, name as 姓名 From student;

-- 查询student表中id 和 name 列，id为编号，name列的别名为”姓名”，表别名s

select id as 编号, name as 姓名 From student as s;

## 清除重复值

### 查询指定列并且结果不出现重复数据

|  |
| --- |
| SELECT DISTINCT 字段名 FROM 表名; |

### 具体操作：

-- 查询学生来至哪些城市，去掉重复

select distinct address from student;



## 查询结果参与运算

### 某列数据和固定值运算

|  |
| --- |
| SELECT 列名1 + 固定值 FROM 表名; |

### 某列数据和其他列数据参与运算

|  |
| --- |
| SELECT 列名1 + 列名2 FROM 表名; |

* + 注意: 参与运算的必须是数值类型

### 需求：

* 准备数据：添加数学，英语成绩列,给每条记录添加对应的数学和英语成绩，查询的时候将数学和英语的成绩相加

### 实现：

-- 修改student表结构,添加数学和英语成绩列

select \* from student;

alter table student add math int, add english int;

-- 查询姓名、数学，将数学每个减1分

select name,math from student;

select name,math-10 from student;

-- 查询所有列与math + english的和并使用别名”总成绩”

select \*,math+english as 总成绩 from student;

-- as可以省略

select \*,math+english 总成绩 from student;



# DQL条件查询

## 条件查询

### 为什么要条件查询

如果不指定查询条件，查询的是所有的记录，如果数据库中记录很多，速度会很慢。

### 条件查询的语法

|  |
| --- |
| SELECT 字段名 FROM 表名 WHERE 条件; |
| 流程：取出表中的每条数据，满足条件的记录就返回，不满足条件的记录不返回 |

## 准备数据

创建一个学生表，包含如下列：

CREATE TABLE student3 (  
 id int, -- 编号  
 name varchar(20), -- 姓名  
 age int, -- 年龄  
 sex varchar(5), -- 性别  
 address varchar(100), -- 地址  
 math int, -- 数学  
 english int -- 英语  
);  
  
INSERT INTO student3(id,NAME,age,sex,address,math,english) VALUES (1,'马云',55,'男','杭州',66,78),(2,'马化腾',45,'女','深圳',98,87),(3,'马景涛',55,'男','香港',56,77),(4,'柳岩',20,'女','湖南',76,65),(5,'柳青',20,'男','湖南',86,NULL),(6,'刘德华',57,'男','香港',99,99),(7,'马德',22,'女','香港',99,99),(8,'德玛西亚',18,'男','南京',56,65);

## 比较运算符

|  |  |
| --- | --- |
| 比较运算符 | 说明 |
| >、<、<=、>=、=、<> | 在mysql中可以使用!=表示不等于，标准的SQL中使用<> |

### 具体操作：

-- 查询math分数大于80分的学生

select \* from student3 where math > 80;

-- 查询english分数小于或等于80分的学生

select \* from student3 where english <=80;

-- 查询age等于20岁的学生

select \* from student3 where age =20;

-- 查询age不等于20岁的学生，注：不等于有两种写法

select \* from student3 where age !=20;

select \* from student3 where age <>20;

## 逻辑运算符

|  |  |
| --- | --- |
| 逻辑运算符 | 说明 |
| and 或 && | 标准的sql是使用前者，mysql支持两种，与 |
| or 或 || | 或 |
| not 或 ! | 非 |

### 具体操作：

-- 查询age大于35且性别为男的学生(两个条件同时满足)

select \* from student3 where age>35 and sex='男';

-- 查询age大于35或性别为男的学生(两个条件其中一个满足)

select \* from student3 where age>35 or sex='男';

-- 查询id是1或3或5的学生

select \* from student3 where id=1 or id=3 or id=5;

## in关键字

|  |
| --- |
| SELECT 字段名 FROM 表名 WHERE 字段 in (数据1, 数据2...); |
| in里面的每个数据都会作为一次条件，只要满足条件的就会显示 |

### 具体操作：

-- 查询id是1或3或5的学生

select \* from student3 where id in (1,3,5);

-- 查询id不是1或3或5的学生

select \* from student3 where id not in (1,3,5);

## 范围查询

|  |
| --- |
| BETWEEN 值1 AND 值2 |
| 表示从值1到值2范围，包头又包尾 |
| 比如：age BETWEEN 80 AND 100 相当于： age>=80 && age<=100 |

-- 查询english成绩大于等于75，且小于等于90的学生

select \* from student3 where english between 75 and 90;

## like关键字

|  |
| --- |
| LIKE表示模糊查询 |
| SELECT \* FROM 表名 WHERE 字段名 LIKE '通配符字符串'; |

### MySQL通配符

|  |  |
| --- | --- |
| 通配符 | 说明 |
| % | 匹配任意多个字符 |
| \_ | 匹配一个字符 |

-- 查询姓马的学生

select \* from student3 where name like '马%';

-- 查询姓名中包含'德'字的学生

select \* from student3 where name like '%德%';

-- 查询姓马，且姓名有三个字的学生, 两个下划线

select \* from student3 where name like '马\_\_';

-- 查询英语为null

select \* from student3 where english = null;

select \* from student3 where english is null;

## 条件查询小结

|  |  |
| --- | --- |
| 比较运算符 | 说明 |
| >、<、<=、>=、=、<> | 判断大小比较 |
| BETWEEN...AND | 范围查询，包头又包尾 |
| IN(集合) | 数据列在集合中存在 |
| LIKE '张%' | 模糊查询 |
| IS NULL | 判断某一列是否为null |