MySQL 基础语法

第1节 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

第2节 数据库的介绍

2.1 数据库概述

2.1.1 数据的存储方式

Java 中创建对象： Student s = new Student(1, "张三") 存在内存中

学习了 Java IO 流：把数据保存到文件中

存储位置	优点	缺点
内存	速度快	不能永久保存，数据是临时状态。
文件	数据可以永久保存	操作数据不方便，查询某个数据。
数据库	1) 数据可以永久保存 2) 查询速度快 3) 对数据的管理方便	占用资源，需要购买。

2.1.2 什么是数据库

1) 存储数据的仓库
2) 本质上是一个文件系统，还是以文件的方式存在服务器的电脑上的。
3) 所有的关系型数据库都可以使用通用的 SQL 语句进行管理 DBMS DataBase Management System

2.2 常见数据库排行榜

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

- **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。

- **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，如：Android。

2.2.1 为什么选择 MySQL

1) 免费
2) 功能强大

第3节 数据库的安装与卸载

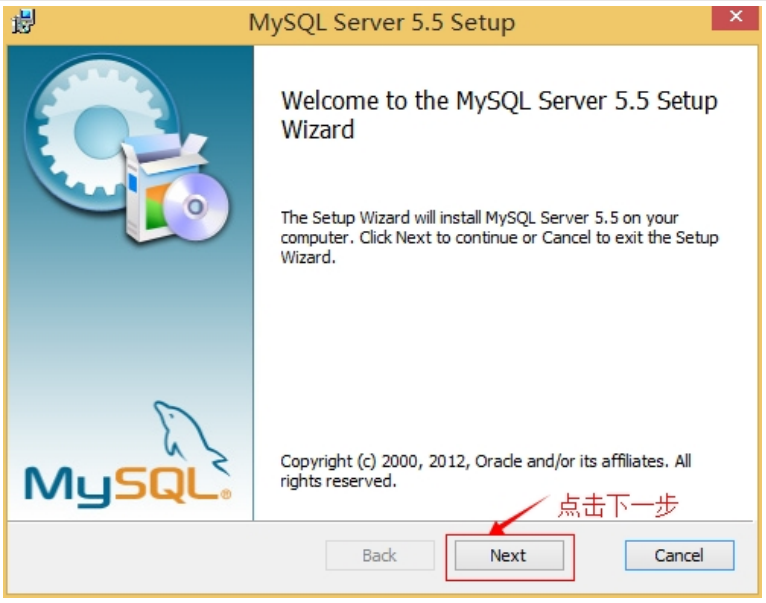
安装过程分成两个部分：

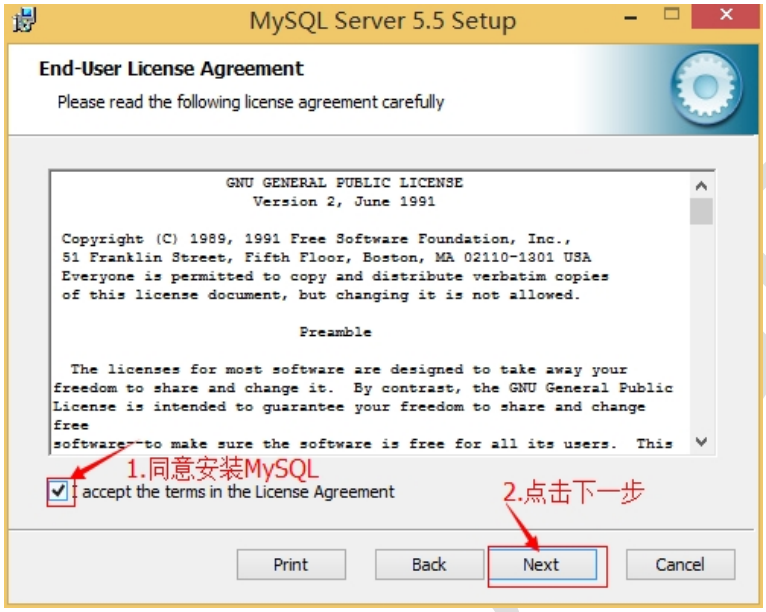
1) 文件解压和复制过程，默认的安装目录：C:\Program Files\MySQL\MySQL Server 5.5\
2) 安装好以后必须对 MySQL 服务器进行配置

在 mysql 中管理员的名字：root

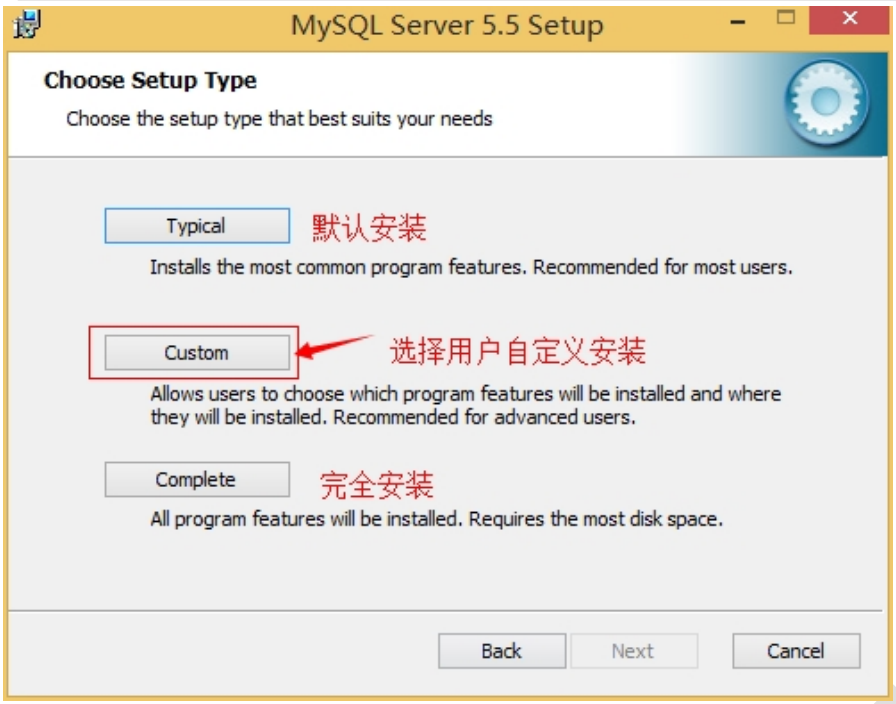
3.1 数据库的安装

1. 打开下载的 mysql 安装文件双击解压缩，运行“mysql-5.5.40-win32.msi”。

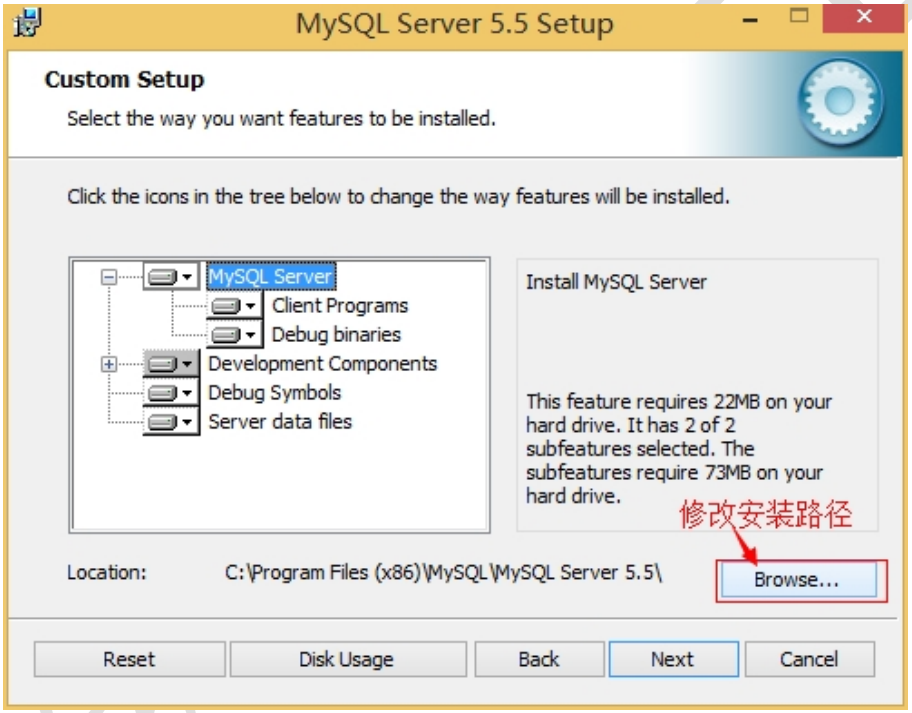




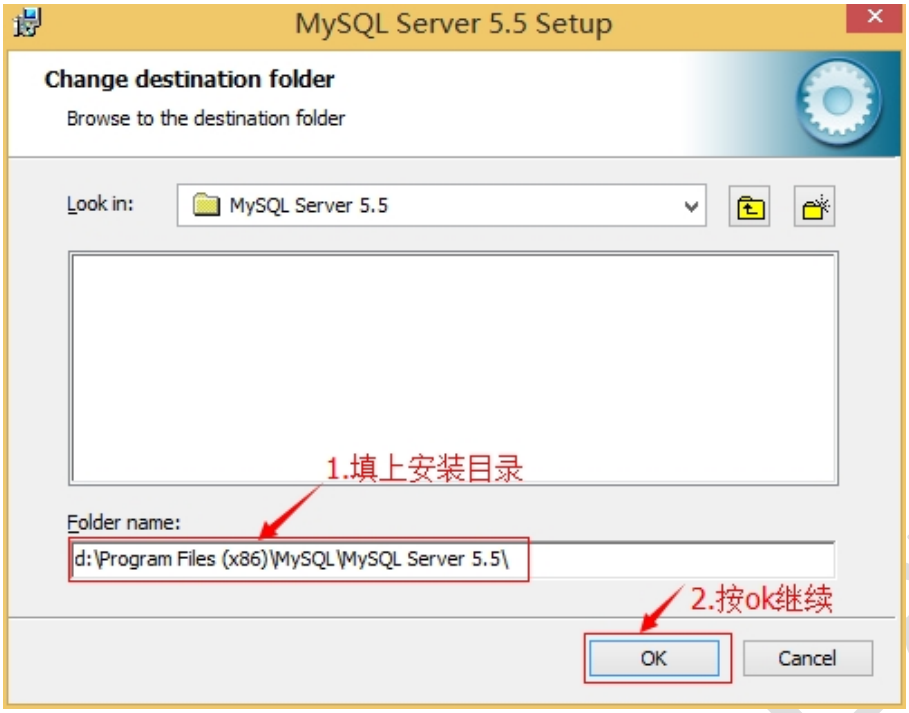
2. 选择安装类型，有Typical（默认）”、“Complete（完全）”、“Custom（用户自定义）”三个选项，选择Custom”，按“next”键继续。



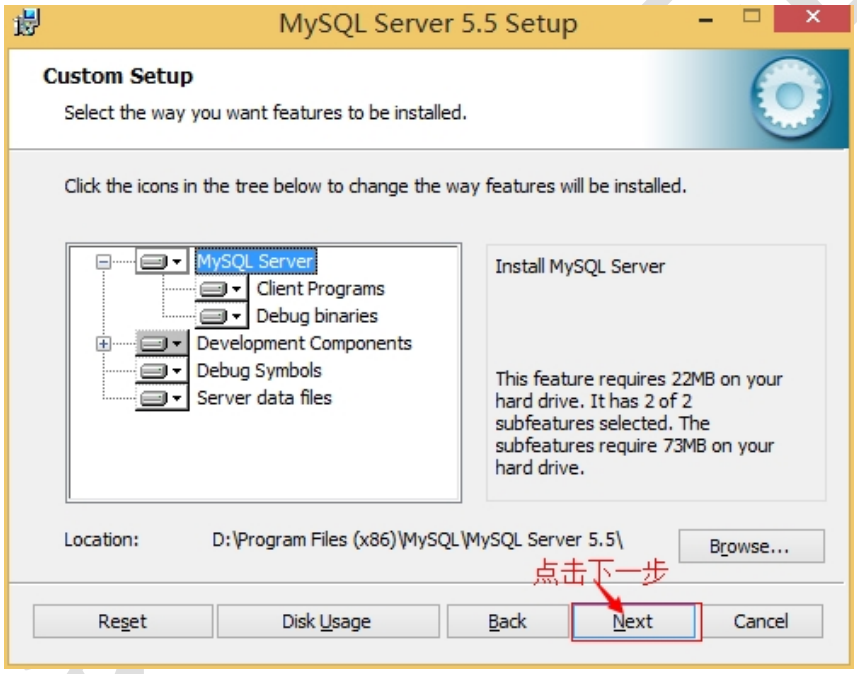
3. 点选“Browse”，手动指定安装目录。

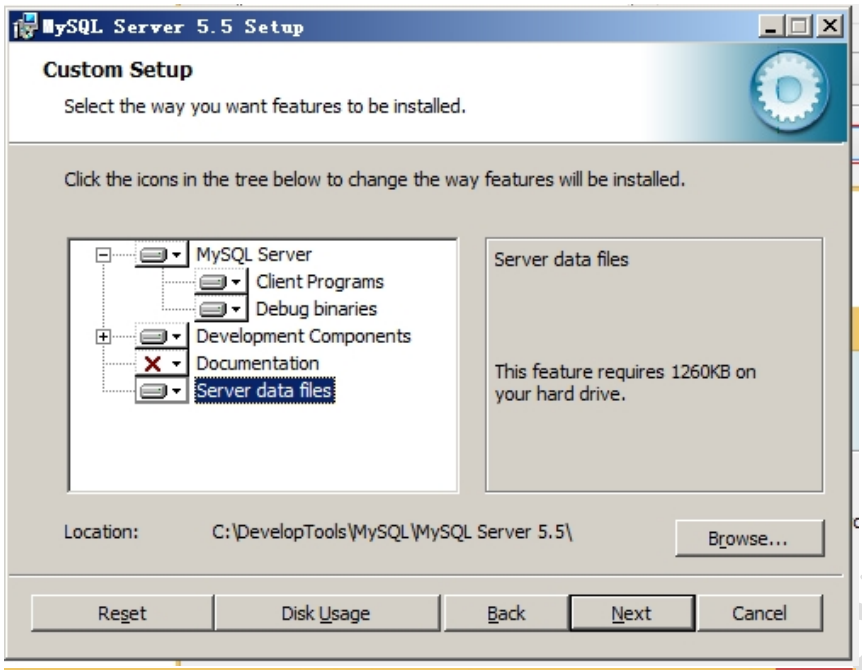


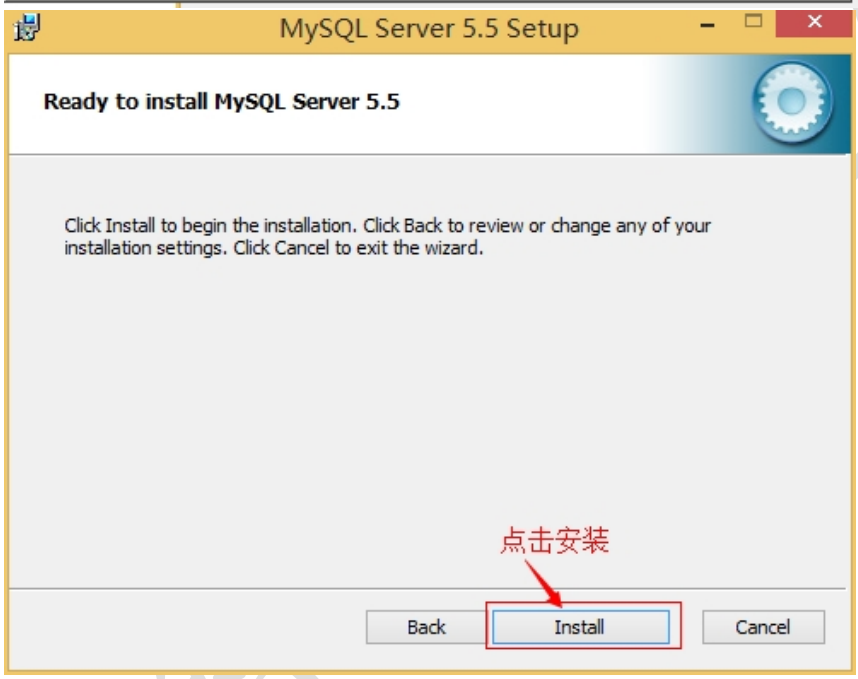
4. 填上安装目录，我的是“d:\Program Files (x86)\MySQL\MySQL Server 5.0”，按“OK”继续。



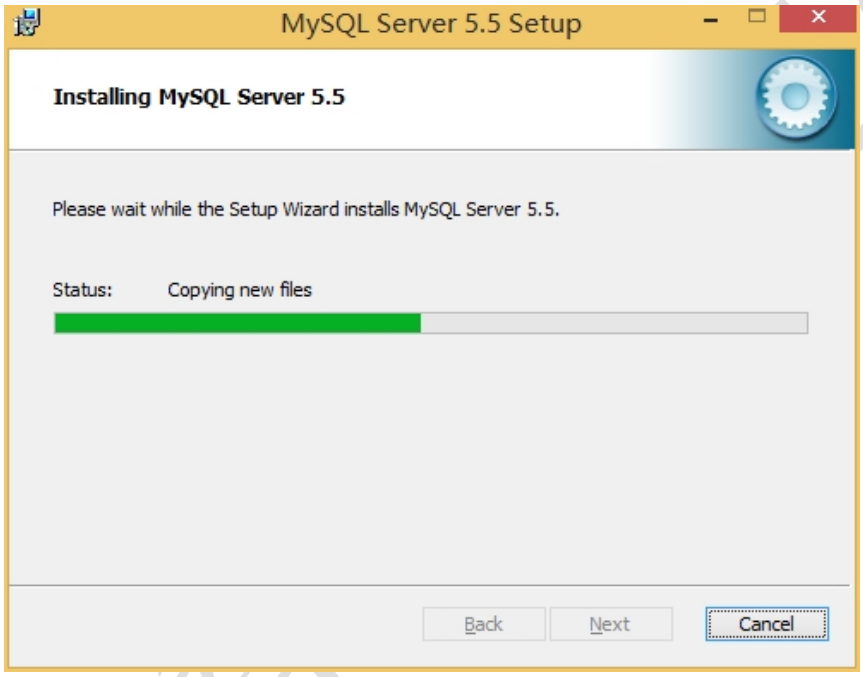
5. 确认一下先前的设置，如果有误，按“Back”返回重做。按“Install”开始安装。



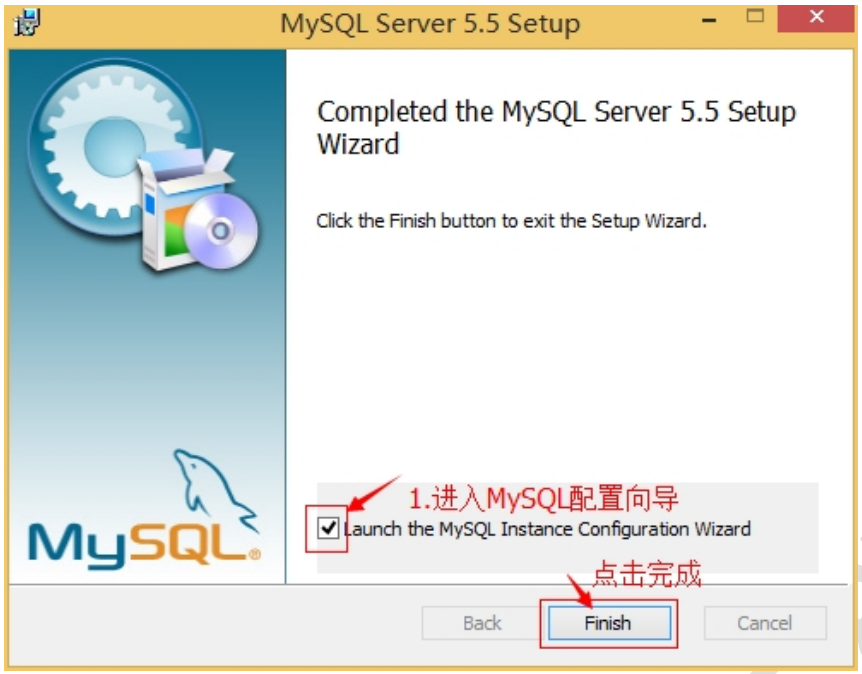

MySQL Server 5.5 Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
MySQL Server
Client Programs
Debug binaries
Development Components
Documentation
Server data files
Server data files
This feature requires 1260KB on your hard drive.
Location:
C:\DevelopTools\MySQL\MySQL Server 5.5\
Browse...
Reset
Disk Usage
Back
Next
Cancel


MySQL Server 5.5 Setup
Ready to install MySQL Server 5.5
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
点击安装
Back
Install
Cancel


MySQL Enterprise
A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services, and support to ensure your business achieves the highest levels of reliability, security, and uptime.
An Enterprise Subscription includes:
1. The MySQL Enterprise Server - The most reliable, secure, and up-to-date version of the world's most popular open source database.
2. MySQL Enterprise Monitor Service - An automated virtual database assistant.
3. MySQL Production Support - Technical and consultative support when you need it, along with service packs, hot-fixes, and more.
For more information click [More...] or visit www.mysql.com/enterprise
More ...
< Back
Next >
Cancel

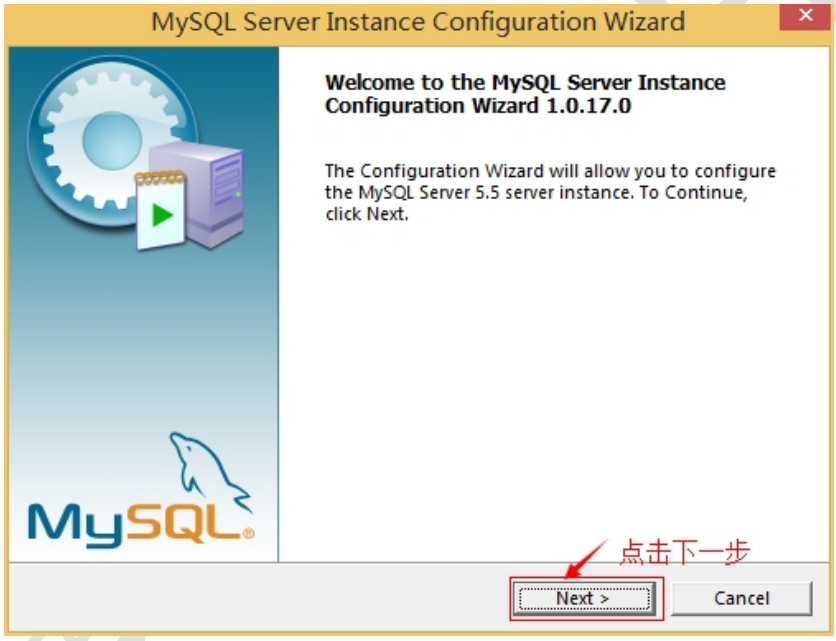

MySQL Server 5.5 Setup
Installing MySQL Server 5.5
Please wait while the Setup Wizard installs MySQL Server 5.5.
Status: Copying new files
Back
Next
Cancel

6. 正在安装中，请稍候，直到出现下面的界面，则完成 MYSQL 的安装

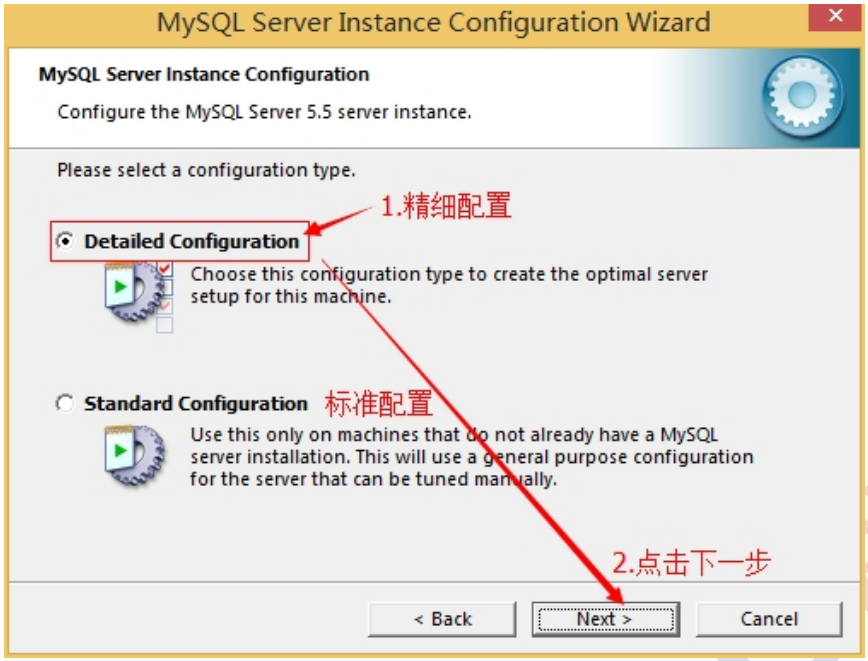


✧ 数据库安装好了还需要对数据库进行配置才能使用 MYSQL 的配置

7. 安装完成了，出现如下界面将进入 mysql 配置向导。



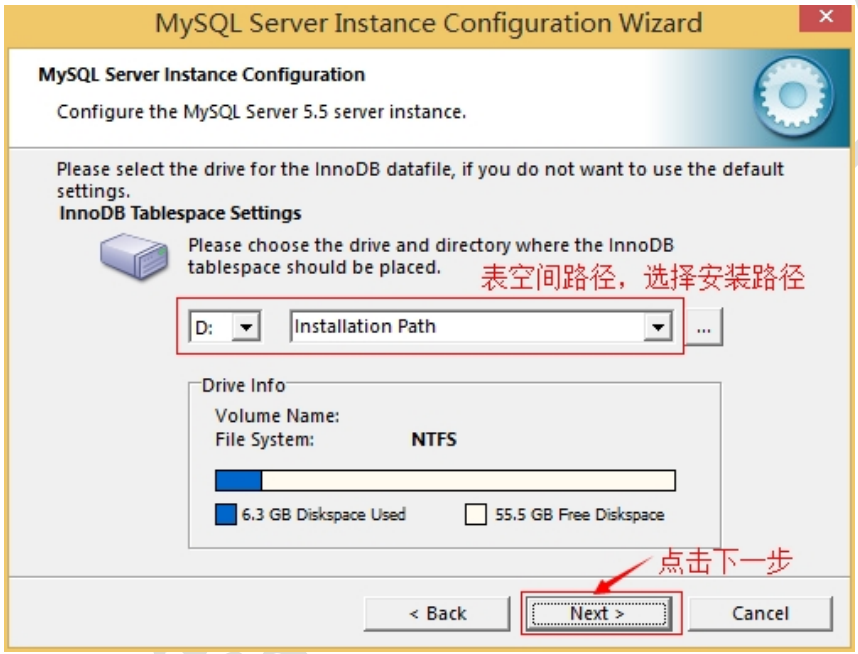
8. 选择配置方式，“Detailed Configuration（手动精确配置）”、“Standard Configuration（标准配置）”，我们选择“Detailed Configuration”，方便熟悉配置过程。



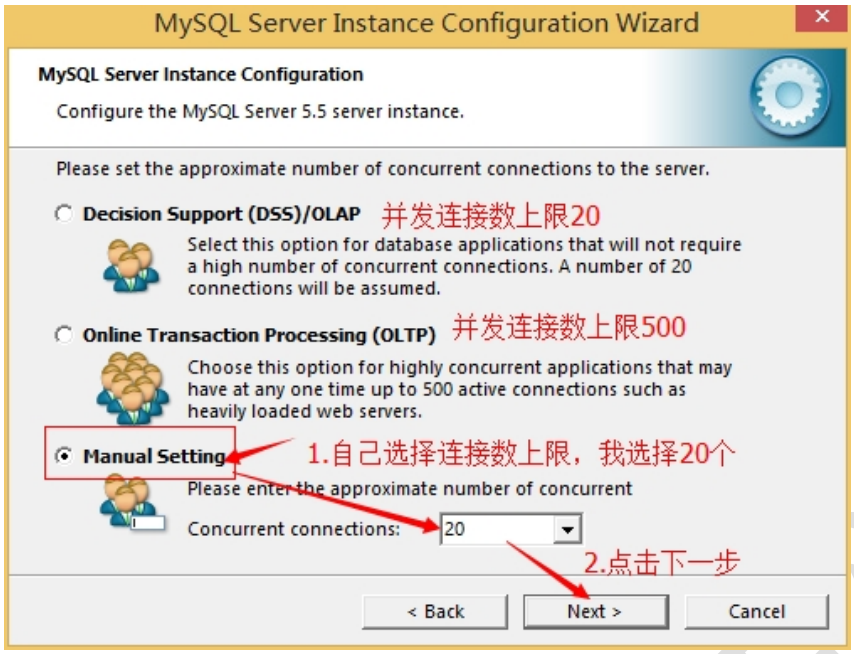
9. 选择服务器类型，“Developer Machine（开发测试类，mysql 占用很少资源）”、“Server Machine（服务器类型，mysql 占用较多资源）”、“Dedicated MySQL Server Machine（专门的数据库服务器，mysql 占用所有可用资源）”

10. 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。

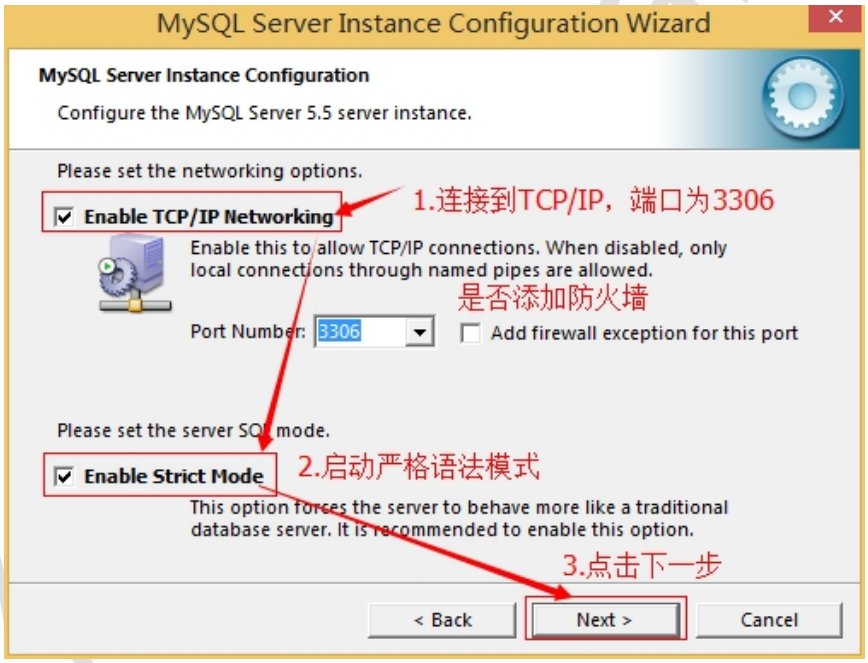

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the database usage.
Multifunctional Database
1.多用途数据库
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.
Transactional Database Only
事务数据库
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.
Non-Transactional Database Only
非事务数据库
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.
2.点击下一步
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the drive for the InnoDB datafile, if you do not want to use the default settings.
InnoDB Tablespace Settings
Please choose the drive and directory where the InnoDB tablespace should be placed.
表空间路径，选择安装路径
D:
Installation Path
...
Drive Info
Volume Name:
File System:
NTFS
6.3 GB Diskspace Used
55.5 GB Free Diskspace
点击下一步
< Back
Next >
Cancel

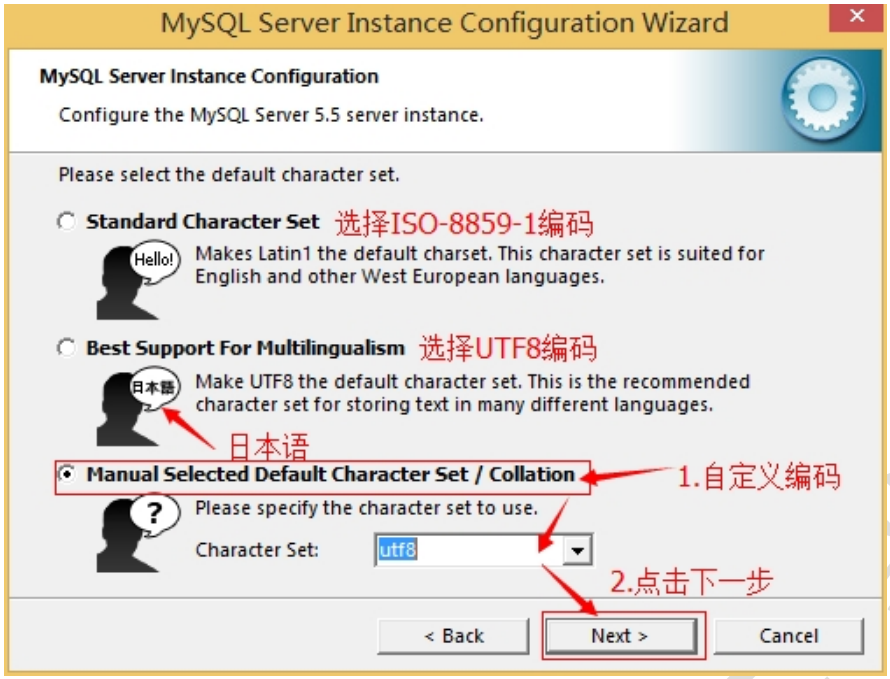
11. 选择网站并发连接数，同时连接的数目“Decision Support(DSS)/OLAP（20个左右）”、“Online Transaction Processing(OLTP)（500 个左右）”、“Manual Setting（手动设置，自己输一个数）”。



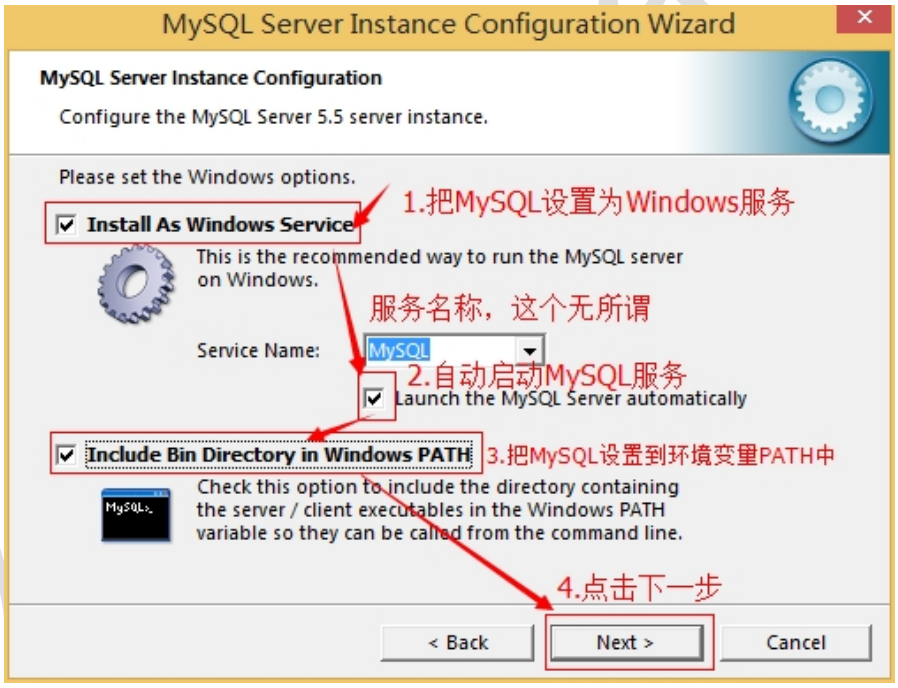
12. 是否启用 TCP/IP 连接，设定端口，如果不启用，就只能在自己的机器上访问 mysql 数据库了，在这个页面上，您还可以选择“启用标准模式”（Enable Strict Mode），这样 MySQL 就不会允许细小的语法错误。如果是新手，建议您取消标准模式以减少麻烦。但熟悉 MySQL 以后，尽量使用标准模式，因为它可以降低有害数据进入数据库的可能性。按“Next”继续



13. 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。

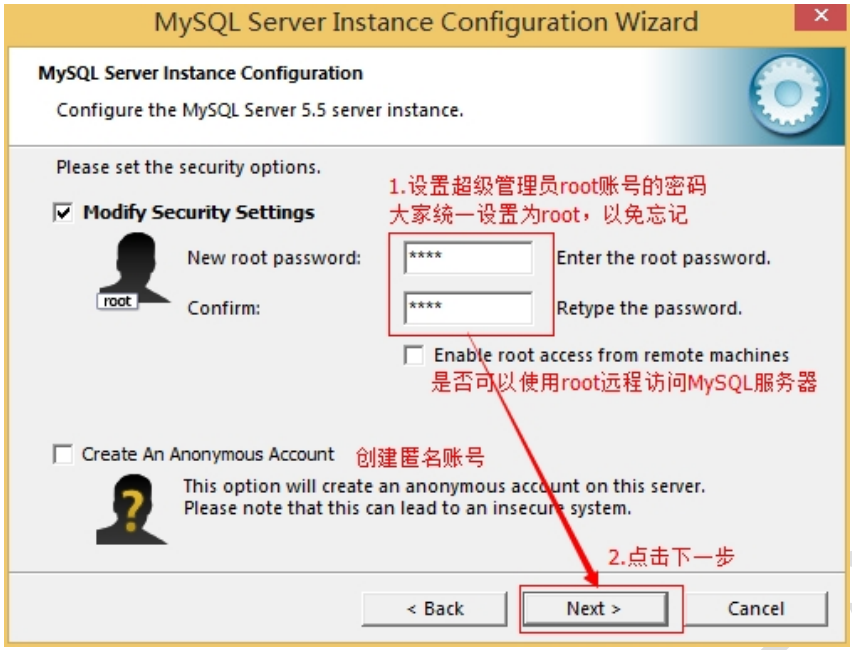


14. 选择是否将 mysql 安装为 windows 服务，还可以指定 Service Name（服务标识名称），是否将 mysql 的 bin 目录加入到 Windows PATH（加入后，就可以直接使用 bin 下的文件，而不用指出目录名，比如连接，“mysqlexe -uusername -ppassword;”就可以了，不用指出 mysql.exe 的完整地址，很方便），我这里全部打上了勾，Service Name 不变。按“Next”继续。

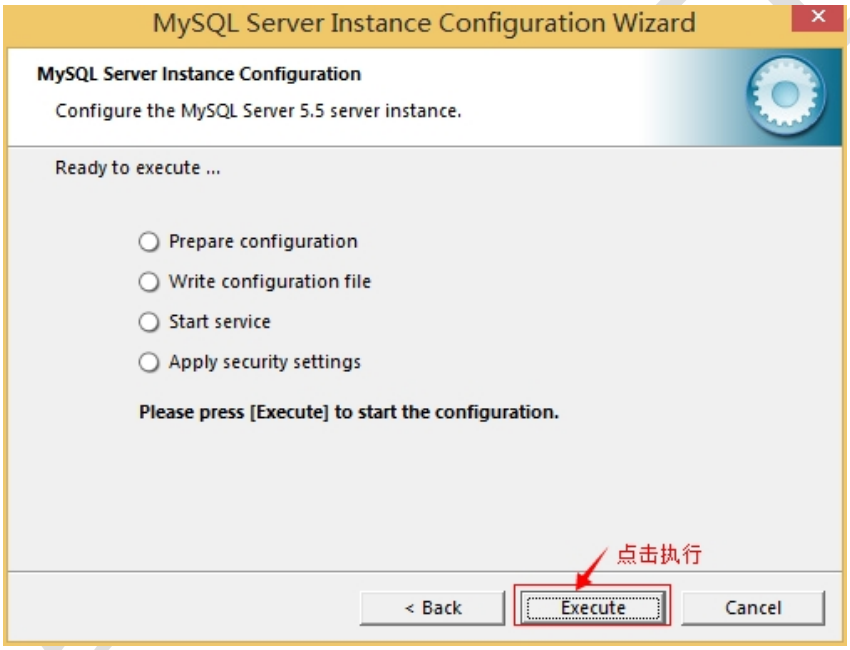


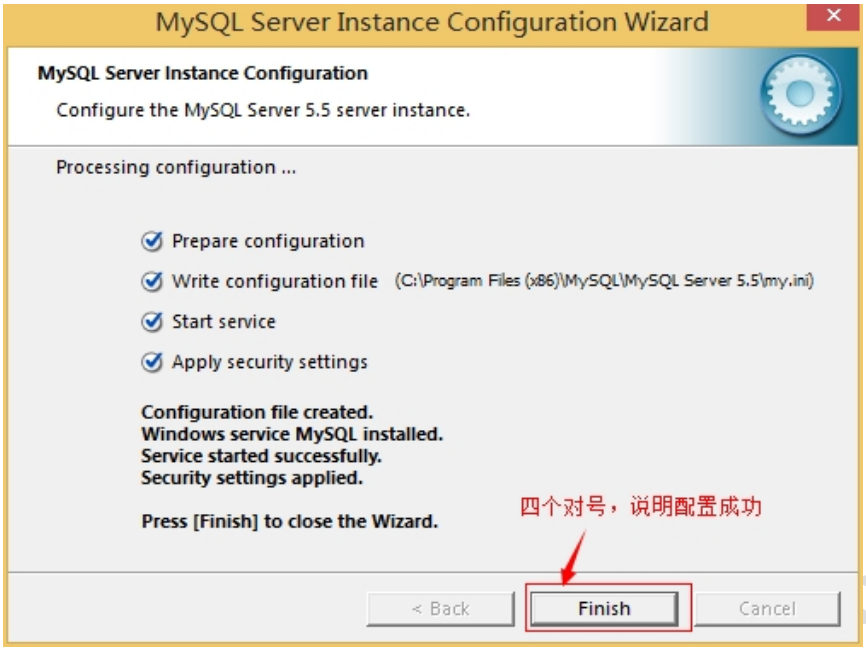
15. 询问是否要修改默认 root 用户（超级管理）的密码。“Enable root access from remote machines（是否允许 root 用户在其它的机器上登陆，如果要安全，就不要勾上，如果要方便，就勾上它）”。最后“Create An Anonymous Account（新建一个匿名用户，匿名用户可以连接数据库，不能操作数据，包括查询）”，

一般就不用勾了，设置完毕，按“Next”继续。

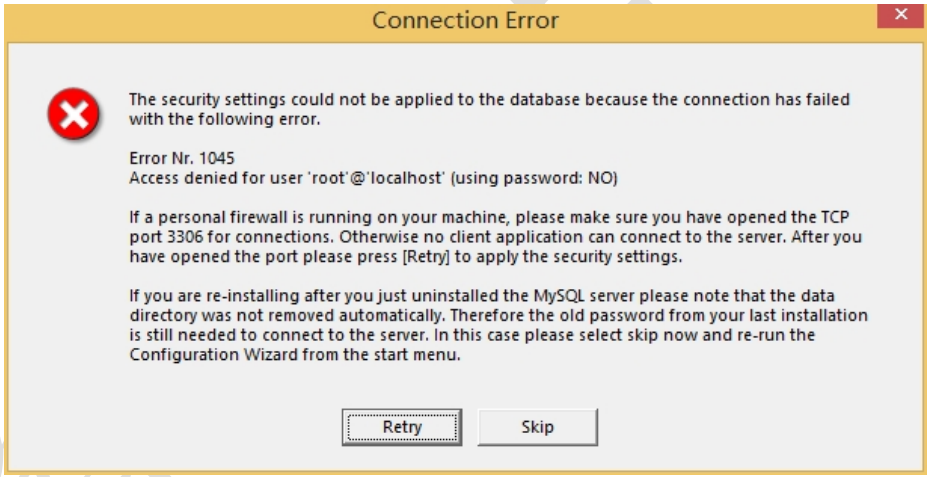


16. 确认设置无误，按“Execute”使设置生效，即完成 MYSQL 的安装和配置。





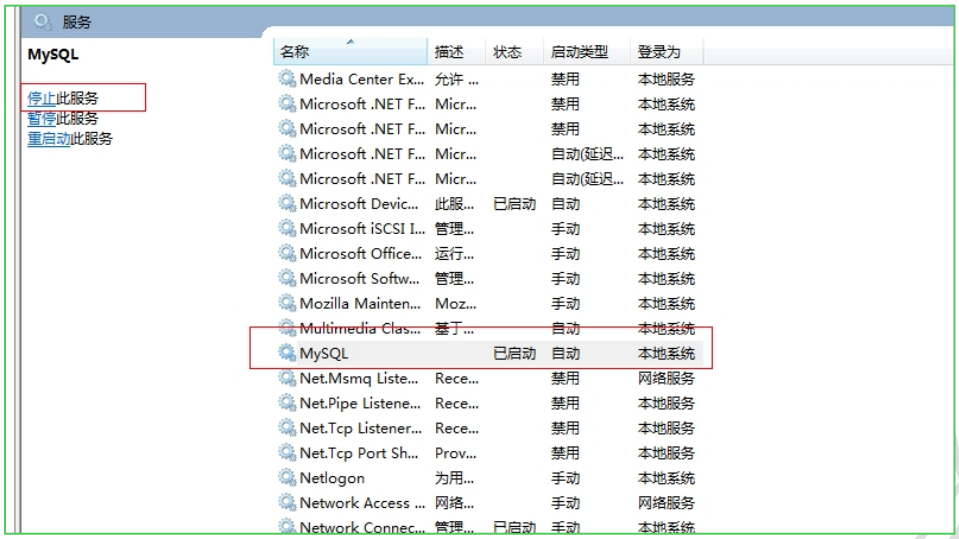
注意：设置完毕，按“Finish”后有一个比较常见的错误，就是不能“Start service”，一般出现在以前有安装 mysql 的服务器上，解决的办法，先保证以前安装的 mysql 服务器彻底卸载掉了；不行的话，检查是否按上面一步所说，之前的密码是否有修改，照上面的操作；如果依然不行，将 mysql 安装目录下的 data 文件夹备份，然后删除，在安装完成后，将安装生成的 data 文件夹删除，备份的 data 文件夹移回来，再重启 mysql 服务就可以了，这种情况下，可能需要将数据库检查一下，然后修复一次，防止数据出错。



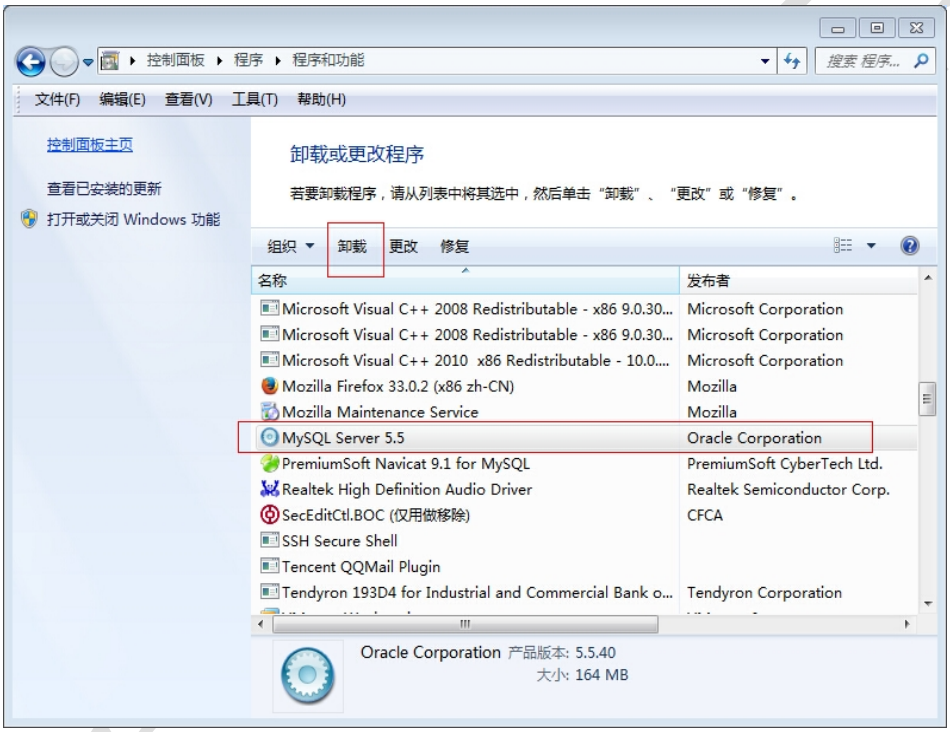
✧ 解决方法：卸载 MySQL,重装 MySQL

3.2 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。

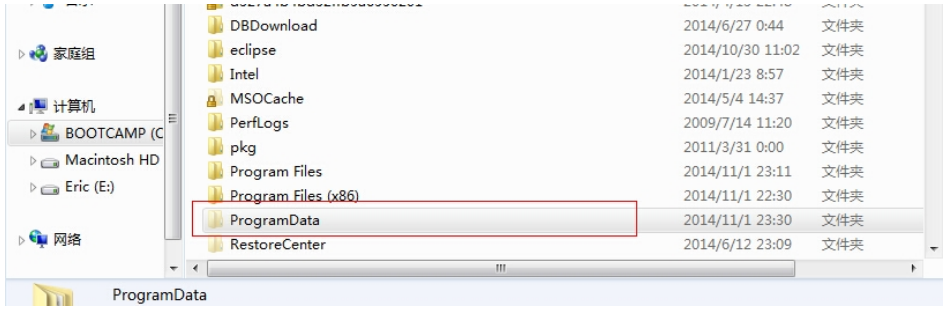


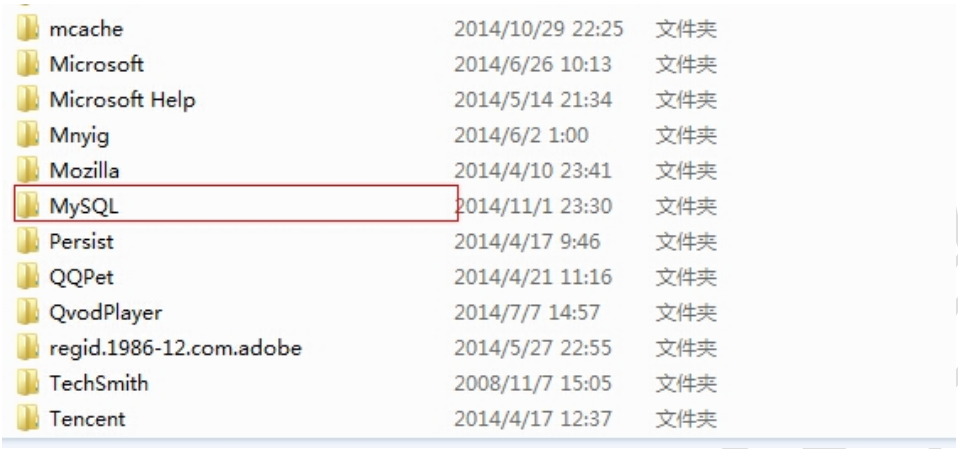
2. 卸载 MySQL 安装程序。找到“控制面板”-> "程序和功能"，卸载 MySQL 程序。



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)





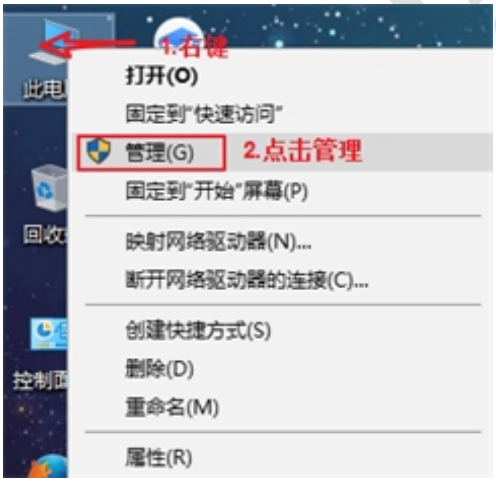
第4节　数据库服务的启动与登录

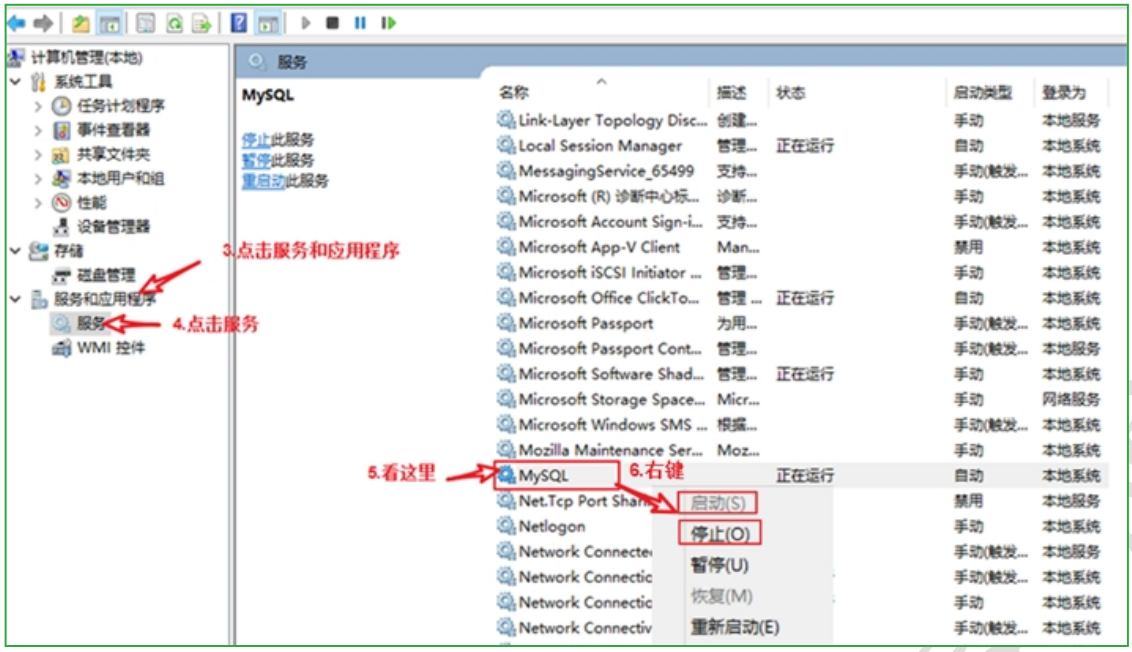
MySQL 服务器启动方式有两种：

1) 通过服务的方式自动启动
2) 手动启动的方式

4.1　Windows 服务方式启动

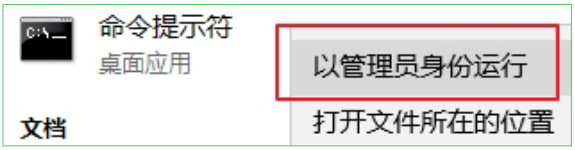
4.1.1　操作步骤：





4.2 DOS 命令方式启动

4.2.1 操作步骤：




```
C:\windows\system32>net start mysql
MySQL 服务正在启动 .
MySQL 服务已经启动成功。


C:\windows\system32>net stop mysql
MySQL 服务正在停止.
MySQL 服务已成功停止。
```

4.3 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

4.3.1 登录格式 1：u 和 p 后面没有空格

```
mysql -u 用户名 -p 密码
```

```
C:\windows\system32>mysql -uroot -proot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- 后输入密码方式：

```
C:\windows\system32>mysql -uroot -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.40 MySQL Community Server (GPL)
```

4.3.2 登录格式 2：

```
mysql -hip 地址 -u 用户名 -p 密码
```

- 127.0.0.1 代表本机的 IP 地址

```
C:\windows\system32>mysql -h127.0.0.1 -uroot -proot
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

4.3.3 登录格式 3：

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

```
C:\windows\system32>mysql --host=127.0.0.1 --user=root --password=root
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

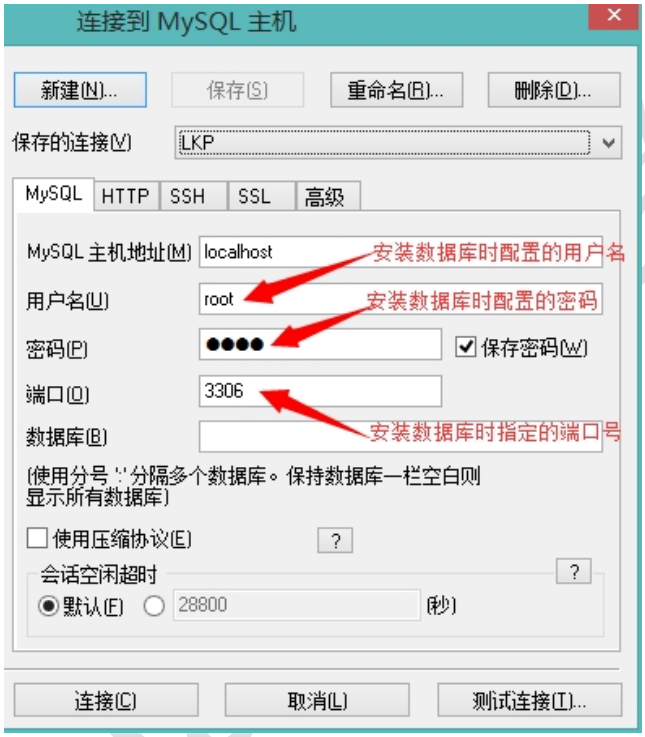
4.3.4 退出 MySQL：

```
quit 或 exit
```

4.4 SQLyog 图形化工具——客户端

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

4.4.1 使用 SQLyog 登录数据库



4.5 MySQL 目录结构

MySQL 的目录结构	描述
bin<目录>	所有 mysql 的可执行文件，如：mysql.exe
MySQLInstanceConfig.exe	 数据库的配置向导，在安装时出现的内容
data<目录>	系统必须的数据库所在的目录
my.ini 文件	mysql 的配置文件，一般不建议去修改。
c:\ProgramData\MySQL\MySQL Server 5.5\data	我们自己创建的数据库所在的文件夹

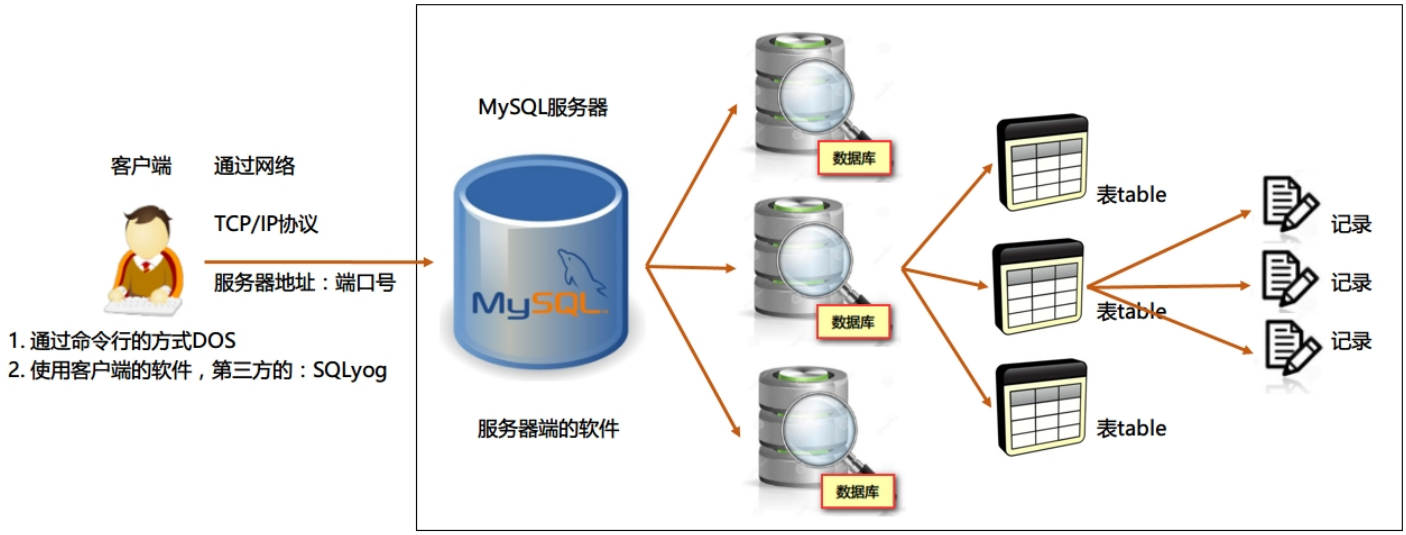
4.6 数据库管理系统

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表内的数据

4.7 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 User 的数据。

数据库管理系统、数据库和表的关系如图所示：



4.7.1 结论：

1) 一个数据库服务器包含多个库
2) 一个数据库包含多张表
3) 一张表包含多条记录

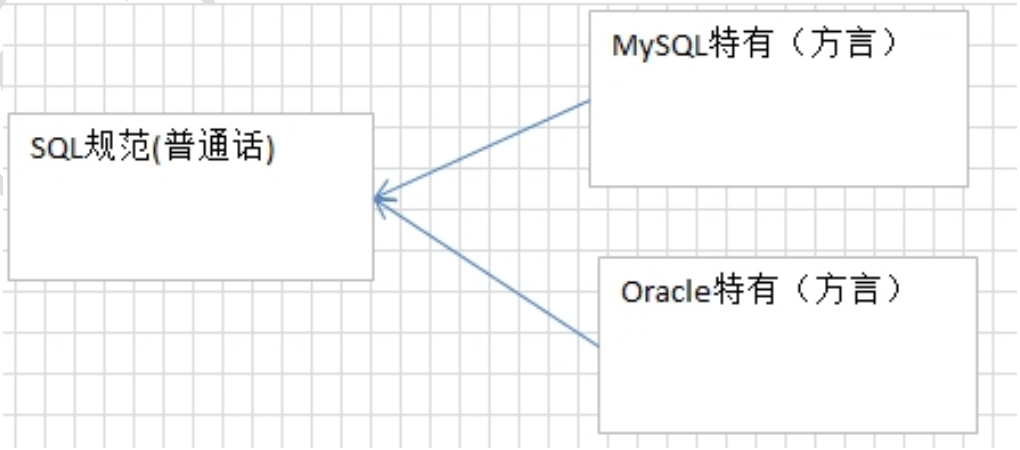
第5节 SQL 的概念

5.1 什么是 SQL

Structured Query Language 结构化查询语言

5.2 SQL 作用

1) 是一种所有关系型数据库的查询规范，不同的数据库都支持。
2) 通用的数据库操作语言，可以用在不同的数据库中。
3) 不同的数据库 SQL 语句有一些区别



5.3 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：对表中的记录操作增删改
3) Data Query Language(DQL 数据查询语言)，如：对表中的查询操作
4) Data Control Language(DCL 数据控制语言)，如：对用户权限的设置

5.4 MySQL 的语法

1) 每条语句以分号结尾，如果在 SQLyog 中不是必须加的。
2) SQL 中不区分大小写，关键字中认为大写和小写是一样的
3) 3 种注释：

注释的语法	说明
--空格	单行注释
/* */	多行注释
#	这是 mysql 特有的注释方式

第6节 DDL 操作数据库

6.1 创建数据库

6.1.1 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;
```

6.1.2 具体操作：

```
-- 直接创建数据库 db1
create database db1;

-- 判断是否存在，如果不存在则创建数据库 db2
create database if not exists db2;

-- 创建数据库并指定字符集为 gbk
create database db3 default character set gbk;
```

6.2 查看数据库

✧ 按 tab 键可以自动补全关键字

```
-- 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 db3;
show create database db1;
```

6.3 修改数据库

6.3.1 修改数据库默认的字符集

```
ALTER DATABASE 数据库名 DEFAULT CHARACTER SET 字符集;
```

6.3.2 具体操作：

- 将 db3 数据库的字符集改成 utf8

```
alter database db3 character set utf8;
```

6.4 删除数据库

6.4.1 删除数据库的语法

```
DROP DATABASE 数据库名;
```

6.4.2 具体操作：

✧ 每行 SQL 语句需要选中再执行，可以按 F9

- 删除 db2 数据库

```
drop database db2;
```

6.5 使用数据库

6.5.1 查看正在使用的数据库

```
SELECT DATABASE();  使用的一个 mysql 中的全局函数
```

6.5.2 使用/切换数据库

```
USE 数据库名;
```

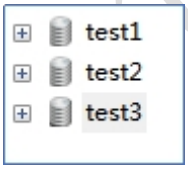
6.5.3 具体操作：

```
-- 查看正在使用的数据库
select database();

-- 改变要使用的数据库
use db4;
```

6.5.4 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？

- 这是一条错误的语句，如果要选中一个数据库，应用使用：use test2;

第7节 DDL 操作表结构

✧ 前提先使用某个数据库

7.1 创建表

7.1.1 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
);
```

7.1.2 关键字说明：

创建表的关键字	说明
CREATE	创建
TABLE	表

7.2 MySQL 数据类型

7.2.1 常使用的数据类型如下：

类型	描述
int	整型
double	浮点型
varchar	字符串型
date	日期类型，格式为 yyyy-MM-dd，只有年月日，没有时分秒；

7.2.2 详细的数据类型如下

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)
	smallint	小整型：小的整数(占 16 位二进制)
	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)
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
大二进制	tinyblob Big Large Object	允许长度 0~255 字节
	blob	允许长度 0~65535 字节
	mediumblob	允许长度 0~167772150 字节
	longblob	允许长度 0~4294967295 字节
大文本	tinytext	允许长度 0~255 字节

	text	允许长度 0~65535 字节
	mediumtext	允许长度 0~167772150 字节
	longtext	允许长度 0~4294967295 字节

7.2.3 具体操作:

- 创建 student 表包含 id,name,birthday 字段

```
create table student (
    id int,  -- 整数
    name varchar(20),  -- 字符串
    birthday date -- 生日，最后没有逗号
);
```

7.3 查看表

7.3.1 查看某个数据库中的所有表

```
SHOW TABLES;
```

7.3.2 查看表结构

```
DESC 表名;
```

7.3.3 查看创建表的 SQL 语句

```
SHOW CREATE TABLE 表名;
```

7.3.4 具体操作：

- 查看 day21 数据库中的所有表

```
use day21;
show tables;
```

- 查看 student 表的结构

```
desc student;
```

  - 执行结果：

Field	Type	Null	Key	Default	Extra
id	int(11) 7B	YES		(NULL) 0K	
name	varchar(20) 11B	YES		(NULL) 0K	
birthday	date 4B	YES		(NULL) 0K	

- 查看 student 的创建表 SQL 语句

```
show create table student;
```

  - 执行结果：

` 存在的目的是为了避免关键字的冲突

```
CREATE TABLE `student` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(20) DEFAULT NULL,
  `birthday` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

7.4 快速创建一个表结构相同的表

7.4.1 语法

```
CREATE TABLE 新表名 LIKE 旧表名;
```

7.4.2 具体操作：

- 创建 s1 表，s1 表结构和 student 表结构相同

```
-- 创建一个 s1 的表与 student 结构相同
create table s1 like student;

desc s1;
```

7.5 删除表

7.5.1 直接删除表

```
DROP TABLE 表名;
```

7.5.2 判断表是否存在，如果存在则删除表

```
DROP TABLE IF EXISTS 表名;
```

7.5.3 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 `create`;
```

- 与直接删除的区别

如果表不存在，不删除，存在则删除

7.6 修改表结构

7.6.1 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 varchar(20)

```
alter table student add remark varchar(20);
```

Field	Type		Null	Key	Default		Extra
id	int(11)	7B	YES		(NULL)	0K	
name	varchar(20)	11B	YES		(NULL)	0K	
birthday	date	4B	YES		(NULL)	0K	
remark	varchar(20)	11B	YES		(NULL)	0K	

7.6.2 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);
```

Field	Type		Null	Key	Default		Extra
id	int(11)	7B	YES		(NULL)	0K	
name	varchar(20)	11B	YES		(NULL)	0K	
birthday	date	4B	YES		(NULL)	0K	
remark	varchar(100)	12B	YES		(NULL)	0K	

7.6.3 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);
```

Field	Type		Null	Key	Default		Extra
id	int(11)	7B	YES		(NULL)	0K	
name	varchar(20)	11B	YES		(NULL)	0K	
birthday	date	4B	YES		(NULL)	0K	
intro	varchar(30)	11B	YES		(NULL)	0K	

7.6.4 删除列 DROP

```
ALTER TABLE 表名 DROP 列名;
```

- 删除 student 表中的字段 intro

```
alter table student change remark intro varchar(30);
```

Field	Type		Null	Key	Default		Extra
id	int(11)	7B	YES		(NULL)	0K	
name	varchar(20)	11B	YES		(NULL)	0K	
birthday	date	4B	YES		(NULL)	0K	

7.6.5 修改表名

```
RENAME TABLE 表名 TO 新表名;
```

- 将学生表 student 改名成 student2

```
rename table student to student2;
```

7.6.6 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

```
alter table student2 character set gbk;
```

第8节 DML 操作表中的数据

用于对表中的记录进行增删改操作

8.1 插入记录

INSERT [INTO] 表名 [字段名] VALUES (字段值)

INSERT INTO 表名：表示往哪张表中添加数据
(字段名 1，字段名 2, ...)：要给哪些字段设置值
VALUES (值 1，值 2, ...)：设置具体的值

8.1.1 插入全部字段

- 所有的字段名都写出来

```
INSERT INTO 表名 (字段名 1，字段名 2，字段名 3…) VALUES (值 1，值 2，值 3);
```

- 不写字段名

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

8.1.2 插入部分数据

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

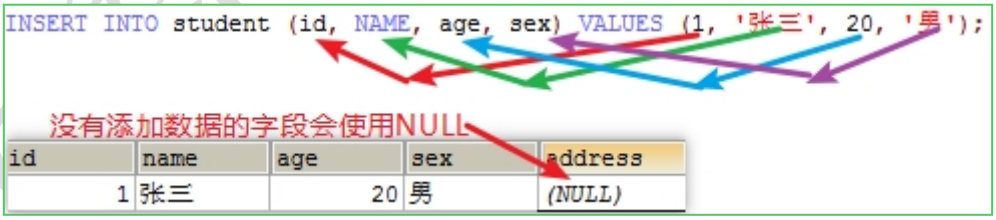
✧ 注：没有添加数据的字段会使用 NULL

8.1.3 具体操作:

- 插入所有的列，向学生表中

```
insert into student (id,name,age,sex) values (1, '孙悟空', 20, '男');
insert into student (id,name,age,sex) values (2, '孙悟天', 16, '男');
```

  - 执行效果：

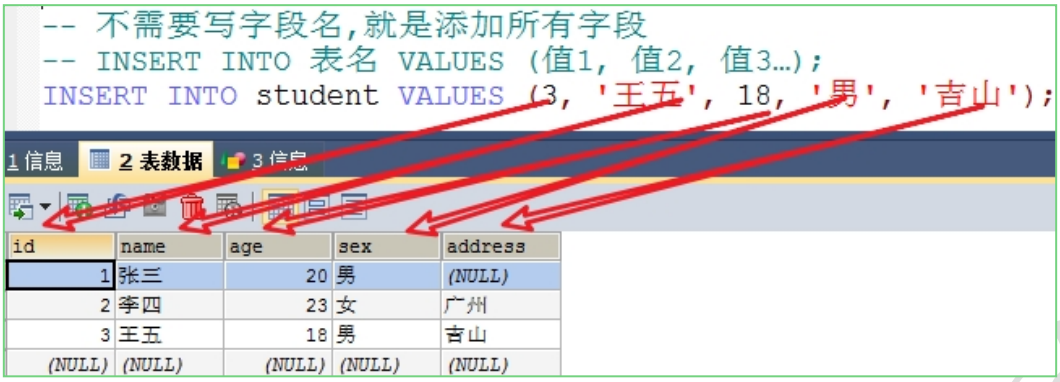


- 向表中插入所有字段

```
-- 插入所有列
insert into student values (3, '孙悟饭', 18, '男', '龟仙人洞中');
-- 如果只插入部分列，必须写列名
insert into student values (3, '孙悟饭', 18, '男');

select * from student;
```

- 执行效果



8.2 DOS 命令窗口操作数据乱码问题的解决

当我们使用 DOS 命令行进行 SQL 语句操作如有有中文会出现乱码

```
mysql> select * from student;
+------+-----------+------+------+-----------------+
| id   | name      | age  | sex  | address         |
+------+-----------+------+------+-----------------+
|    1 | 瀛欐偰绌?  |   20 | 鑾?  | NULL            |
|    2 | 瀛欐偰澶?  |   16 | 鑾?  | NULL            |
|    3 | 瀛欐偰椹?  |   18 | 鑾?  | 榫熶粈浜烘磨涓?  |
+------+-----------+------+------+-----------------+
3 rows in set (0.00 sec)
```

8.2.1 insert 的注意事项：

1) 插入的数据应与字段的数据类型相同

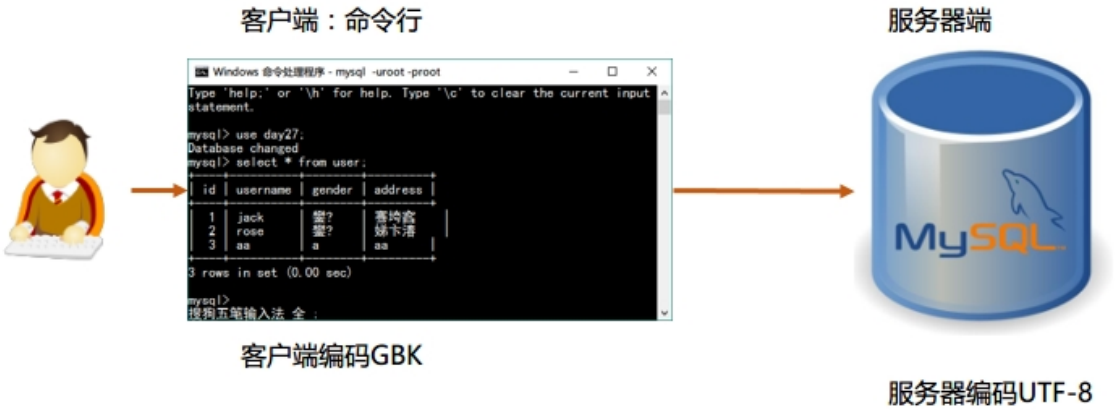
2) 数据的大小应在列的规定范围内，例如：不能将一个长度为 80 的字符串加入到长度为 40 的列中。

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。

8.2.2 乱码产生的原因



8.2.3 查看 MySQL 内部设置的编码

- 查看包含 character 开头的全局变量

```
show variables like 'character%';
```

  - 执行效果

```
Variable_name            | Value
-------------------------+----------------------------------------------------------
character_set_client     | utf8
character_set_connection | utf8
character_set_database   | utf8
character_set_filesystem | binary
character_set_results    | utf8
character_set_server     | utf8
character_set_system     | utf8
```

8.2.4 解决方案

修改 client、connection、results 的编码为 GBK，保证和 DOS 命令行编码保持一致

单独设置	说明
set character_set_client=gbk;	修改客户端的字符集为 GBK
set character_set_connection=gbk;	修改连接的字符集为 GBK
set character_set_results=gbk;	修改查询的结果字符集为 GBK

- 同时设置三项

```
set names gbk;
```

✧ 注意：退出 DOS 命令行就失效了，需要每次都配置

```
mysql> select * from student;
+------+--------+------+------+------------+
| id   | name   | age  | sex  | address    |
+------+--------+------+------+------------+
|    1 | 孙悟空 |   20 | 男   | NULL       |
|    2 | 孙悟天 |   16 | 男   | NULL       |
|    3 | 孙悟饭 |   18 | 男   | 龟仙人洞中 |
+------+--------+------+------+------------+
```

8.3 蠕虫复制

8.3.1 什么是蠕虫复制：

将一张已经存在的表中的数据复制到另一张表中。

8.3.2 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;
```

8.3.3 具体操作:

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

```
drop table student2;

create table student2 like student;

-- 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 name,age 字段数据到 student2 表中，两张表都写出相应的列名
insert into student2 (name,age) select name,age from student;

select * from student2;
```

id	name	age	sex	address
(NULL)	孙悟空	20	(NULL)	(NULL)
(NULL)	孙悟天	16	(NULL)	(NULL)
(NULL)	孙悟饭	18	(NULL)	(NULL)

8.4 更新表记录

UPDATE 表名 SET 列名=值 [WHERE 条件表达式]

UPDATE: 需要更新的表名
SET: 修改的列值
WHERE: 符合条件的记录才更新

你可以同时更新一个或多个字段。
你可以在 **WHERE** 子句中指定任何条件。

8.4.1 不带条件修改数据

UPDATE 表名 SET 字段名=值; -- 修改所有的行

8.4.2 带条件修改数据

UPDATE 表名 SET 字段名=值 WHERE 字段名=值;

8.4.3 具体操作：

```
-- 不带条件修改数据，将所有的性别改成女
update student set sex = '女';

-- 带条件修改数据，将 id 号为 2 的学生性别改成男
update student set sex='男' where id=2;

-- 一次修改多个列，把 id 为 3 的学生，年龄改成 26 岁，address 改成北京
update student set age=26, address='北京' where id=3;
```

8.5 删除表记录

DELETE FROM 表名 [WHERE 条件表达式]

如果没有指定 **WHERE** 子句，**MySQL** 表中的所有记录将被删除。
你可以在 **WHERE** 子句中指定任何条件

8.5.1 不带条件删除数据

DELETE FROM 表名;

8.5.2 带条件删除数据

```
DELETE FROM 表名 WHERE 字段名=值;
```

8.5.3 使用 truncate 删除表中所有记录

```
TRUNCATE TABLE 表名;
```

8.5.4 truncate 和 delete 的区别：

truncate 相当于删除表的结构，再创建一张表。

8.5.5 具体操作：

```
-- 带条件删除数据，删除 id 为 1 的记录
delete from student where id=1;

-- 不带条件删除数据,删除表中的所有数据
delete from student;
```

第9节 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

```
SELECT 列名 FROM 表名 [WHERE 条件表达式]
```

1) **SELECT 命令可以读取一行或者多行记录。**
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。**

9.1 简单查询

9.1.1 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;
```

9.1.2 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 name 和 age 列

```
select name,age from student;
```

9.2 指定列的别名进行查询

9.2.1 使用关键字

- 使用别名的好处： 显示的时候使用新的名字，并不修改表的结构。

9.2.2 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;
```

9.2.3 具体操作：

```
-- 使用别名
select name as 姓名,age as 年龄 from student;
-- 表使用别名
select st.name as 姓名,age as 年龄 from student as st
```

- 表使用别名的原因：用于多表查询操作

9.3 清除重复值

9.3.1 查询指定列并且结果不出现重复数据

```
SELECT DISTINCT 字段名 FROM 表名;
```

9.3.2 具体操作：

- 查询学生来至于哪些地方

```
-- 查询学生来至于哪些地方
select address from student;

-- 去掉重复的记录
select distinct address from student;
```

9.4 查询结果参与运算

9.4.1 某列数据和固定值运算

```
SELECT 列名1 + 固定值 FROM 表名;
```

9.4.2 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

9.4.3 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

9.4.4 实现：

```
select * from student;

-- 给所有的数学加 5 分
select math+5 from student;

-- 查询 math + english 的和
select * from student;

select *,(math+english) as 总成绩 from student;
-- as 可以省略
```

```
select *,(math+english) 总成绩 from student;
```

9.5 条件查询

9.5.1 为什么要条件查询

如果没有查询条件，则每次查询所有的行。实际应用中，一般要指定查询的条件。对记录进行过滤。

9.5.2 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);
```

- 运算符

比较运算符	说明
>、<、<=、>=、=、<>	<>在 SQL 中表示不等于，在 mysql 中也可以使用!= 没有==
BETWEEN...AND	在一个范围之内，如：between 100 and 200 相当于条件在 100 到 200 之间，包头又包尾
IN(集合)	集合表示多个值，使用逗号分隔
LIKE '张%'	模糊查询
IS NULL	查询某一列为 NULL 的值，注：不能写=NULL

- 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 &&	与，SQL 中建议使用前者，后者并不通用。
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
%	匹配任意多个字符串
_	匹配一个字符

```
-- 查询姓马的学生
```

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

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，且姓名有两个字的学生
select * from student3 where name like '马_';
```