

MYSQL基础语法

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1章 数据库的介绍

1.1 数据库概述

1.1.1 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。

1.1.2 数据的存储方式

1. 数据保存在内存

```
int[] arr = new int[]{1, 2, 3, 4};  
ArrayList<Integer> list = new ArrayList<Integer>();  
list.add(1);
```

new出来的对象存储在堆中。堆是内存中的一小块空间

优点：内存速度快 缺点：断电/程序退出，数据就清除了。内存价格贵

2. **数据保存在普通文件** 优点：永久保存 缺点：查找、增加、修改、删除数据比较麻烦，效率低
3. **数据保存在数据库** 优点：永久保存，通过SQL语句比较方便的操作数据库

1.2 数据库的优点

数据库是按照特定的格式将数据存储文件中，通过SQL语句可以方便的对大量数据进行增、删、改、查操作，数据库是对大量的信息进行管理的高效的解决方案。

1.3 常见数据库

Rank			DBMS	Database Model	Score		
Nov 2016	Oct 2016	Nov 2015			Nov 2016	Oct 2016	Nov 2015
1.	1.	1.	Oracle +	Relational DBMS	1413.01	-4.09	-67.94
2.	2.	2.	MySQL +	Relational DBMS	1373.56	+10.91	+86.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	-0.38	+91.48
4.	↑ 5.	↑ 5.	PostgreSQL	Relational DBMS	325.82	+7.12	+40.13
5.	↓ 4.	↓ 4.	MongoDB +	Document store	325.48	+6.67	+20.87
6.	6.	6.	DB2	Relational DBMS	181.46	+0.90	-21.07
7.	7.	↑ 8.	Cassandra +	Wide column store	133.97	-1.09	+1.05
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	125.97	+1.30	-14.99
9.	9.	↑ 10.	Redis	Key-value store	115.54	+6.00	+13.13
10.	10.	↓ 9.	SQLite	Relational DBMS	112.00	+3.43	+8.55

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。**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。**SQLServer**：MicroSoft 公司收费的中型的数据库。C#、.net等语言常使用。**SyBase**：已经淡出历史舞台。提供了一个非常专业数据建模的工具PowerDesigner。**SQLite**：嵌入式的小型数据库，应用在手机端。

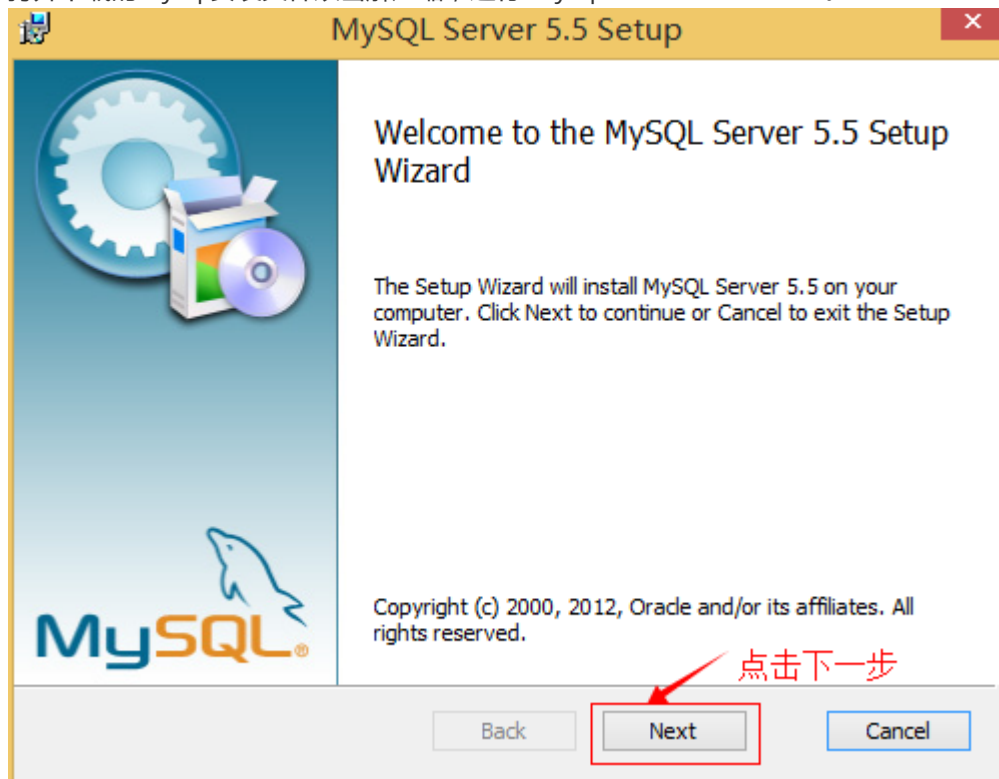
常用数据库：**MYSQL**，**Oracle** 在web应用中，使用的最多的就是MySQL数据库，原因如下：

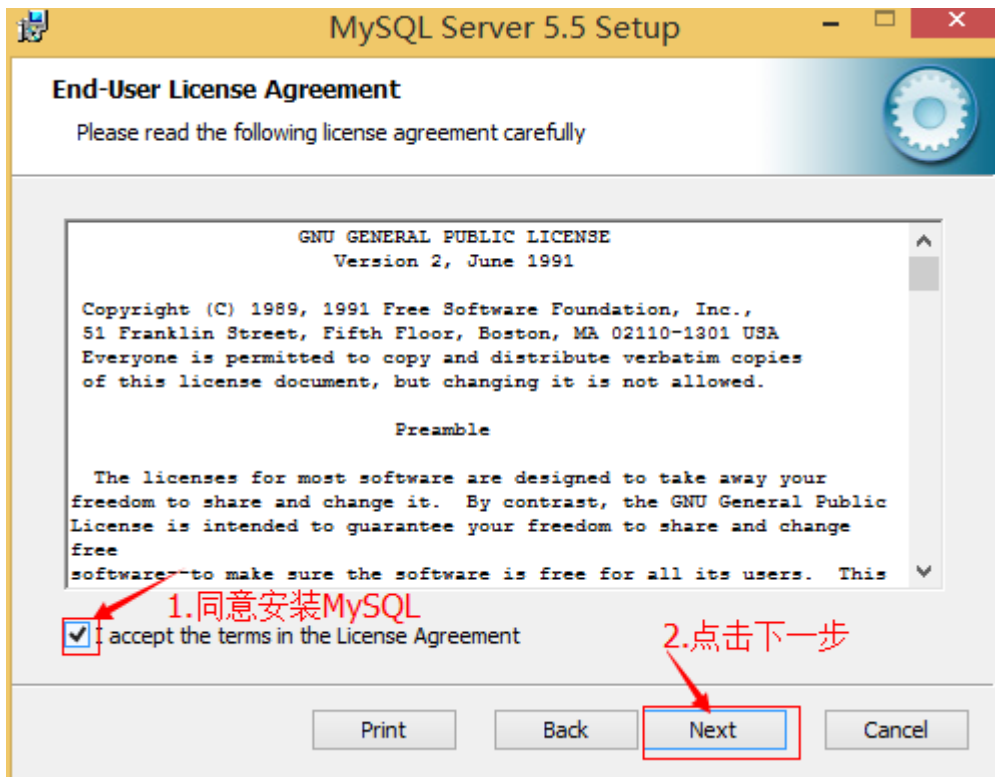
1. 开源、免费
2. 功能足够强大，足以应付web应用开发（最高支持千万级别的并发访问）

第2章 数据库的安装与使用

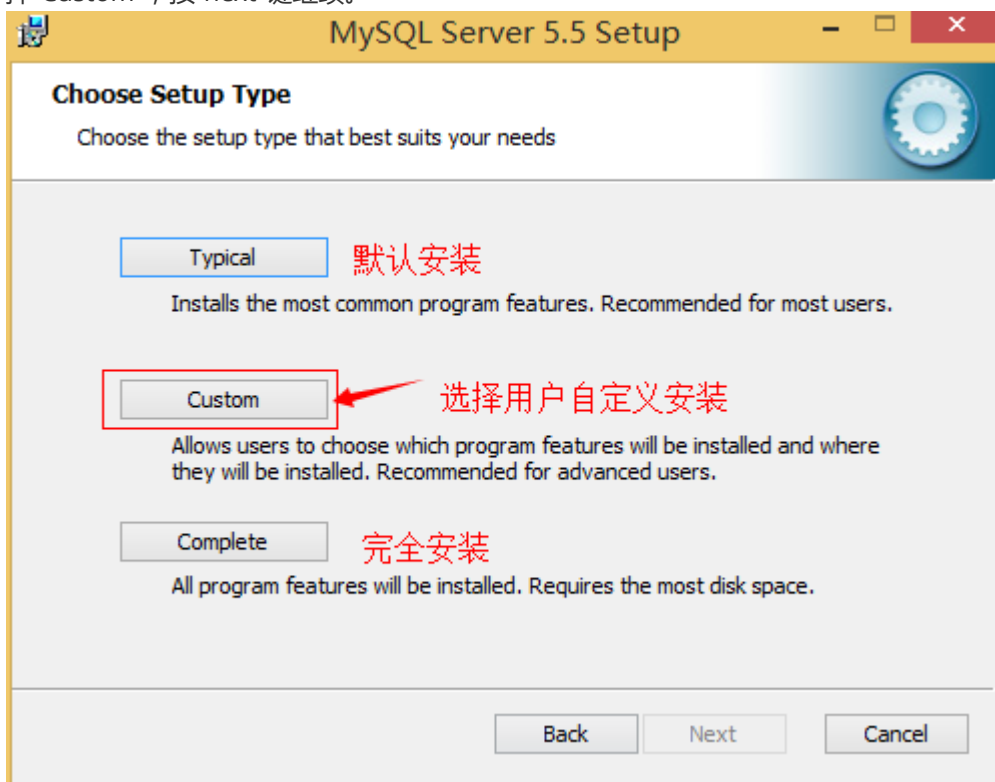
2.1 数据库的安装

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

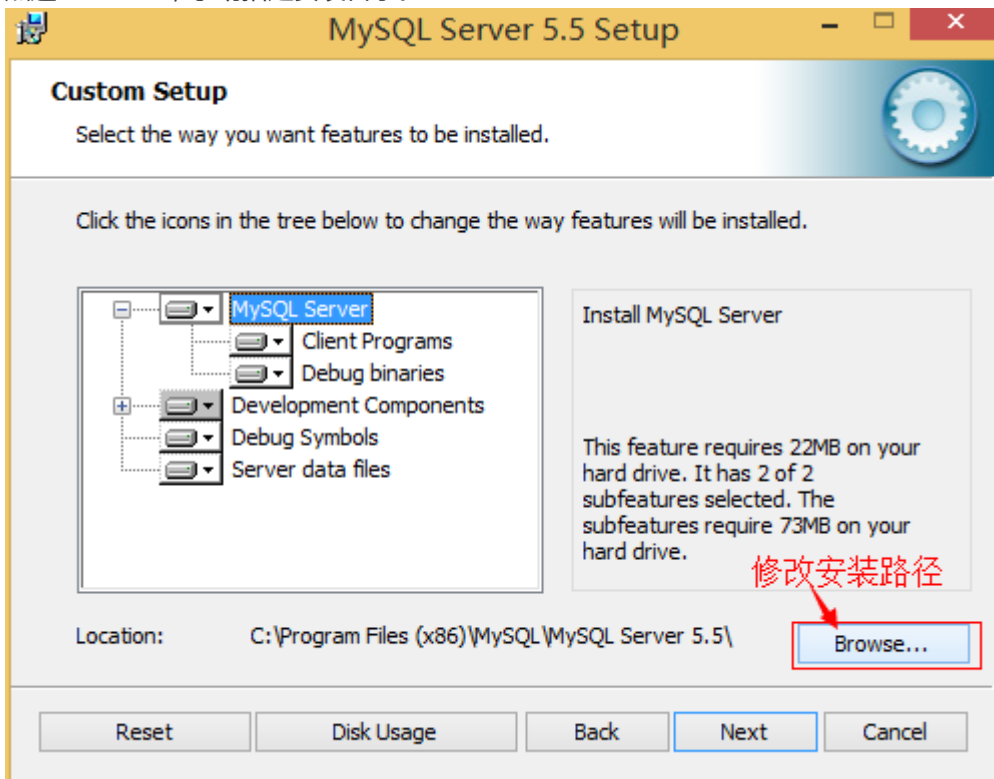




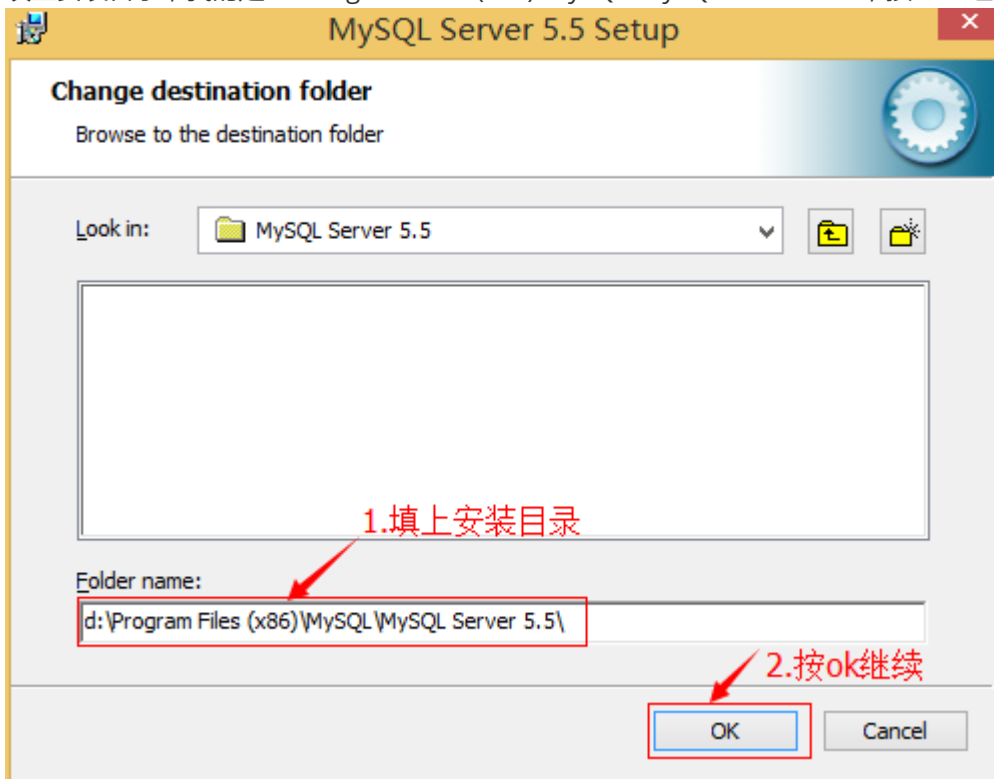
2. 选择安装类型，有“Typical（默认）”、“Complete（完全）”、“Custom（用户自定义）”三个选项，选择“Custom”，按“next”键继续。



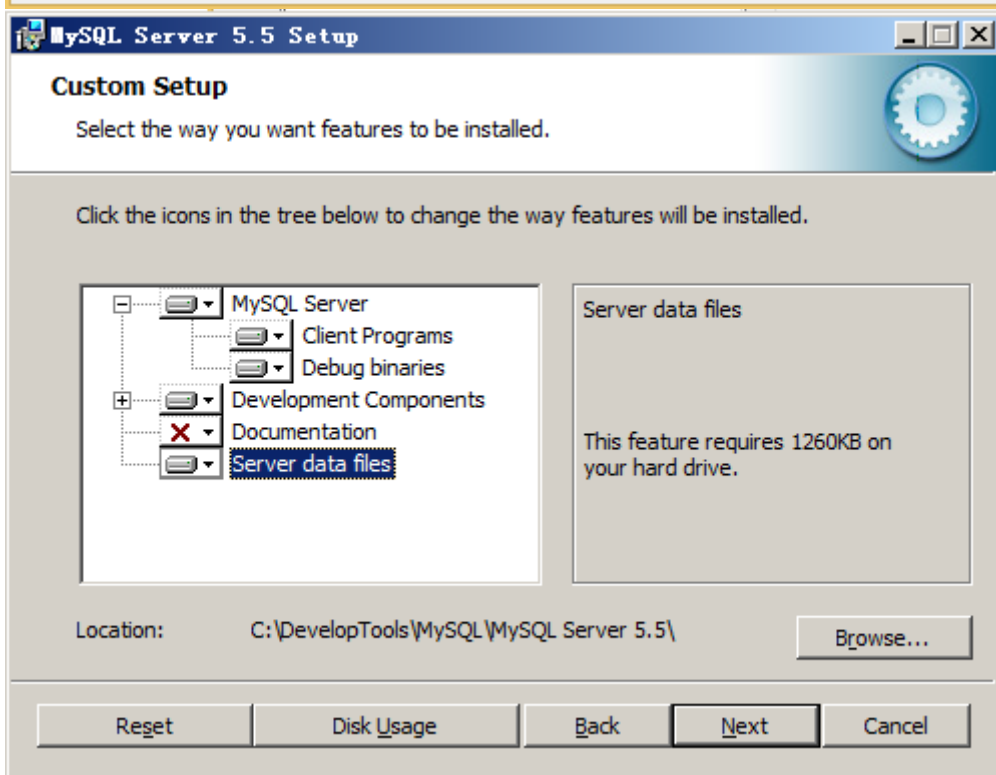
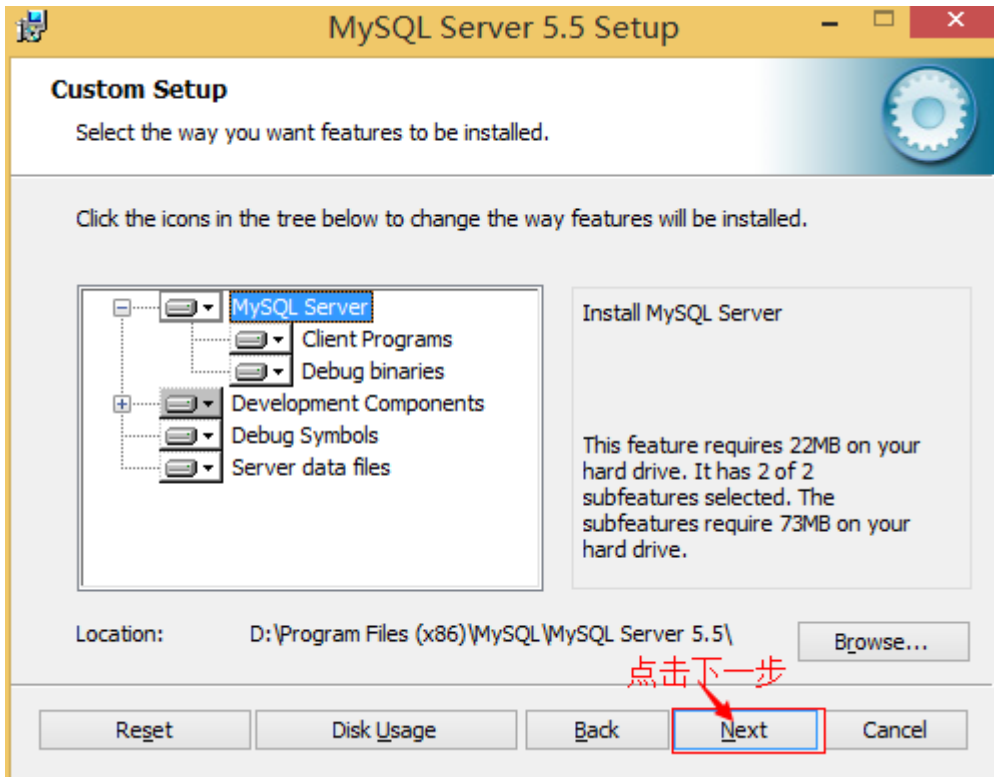
3. 点选“Browse”，手动指定安装目录。



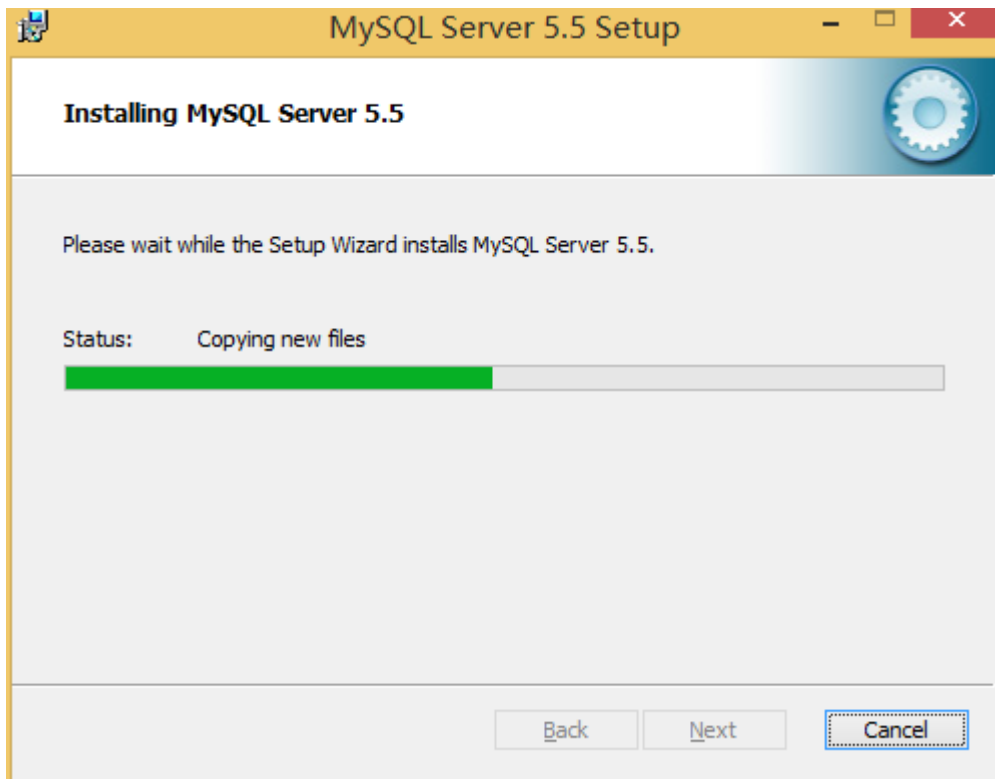
4. 填上安装目录，我的是“d:\Program Files (x86)\MySQL\MySQL Server 5.0”，按“OK”继续。



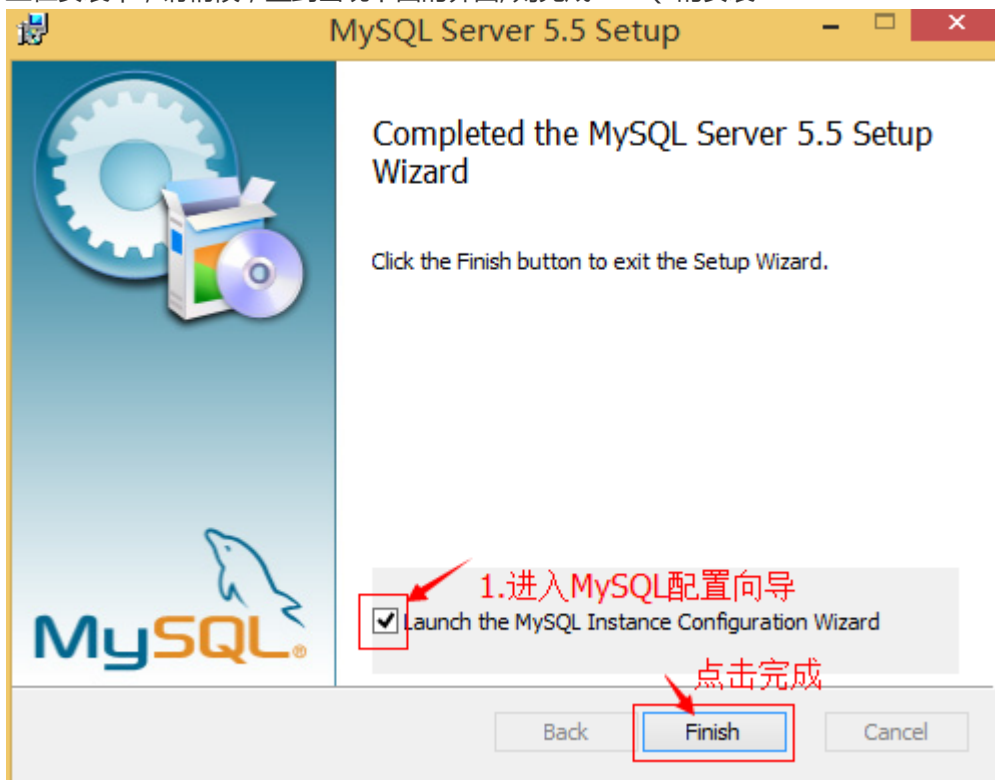
5. 确认一下先前的设置，如果有误，按“Back”返回重做。按“Install”开始安装。





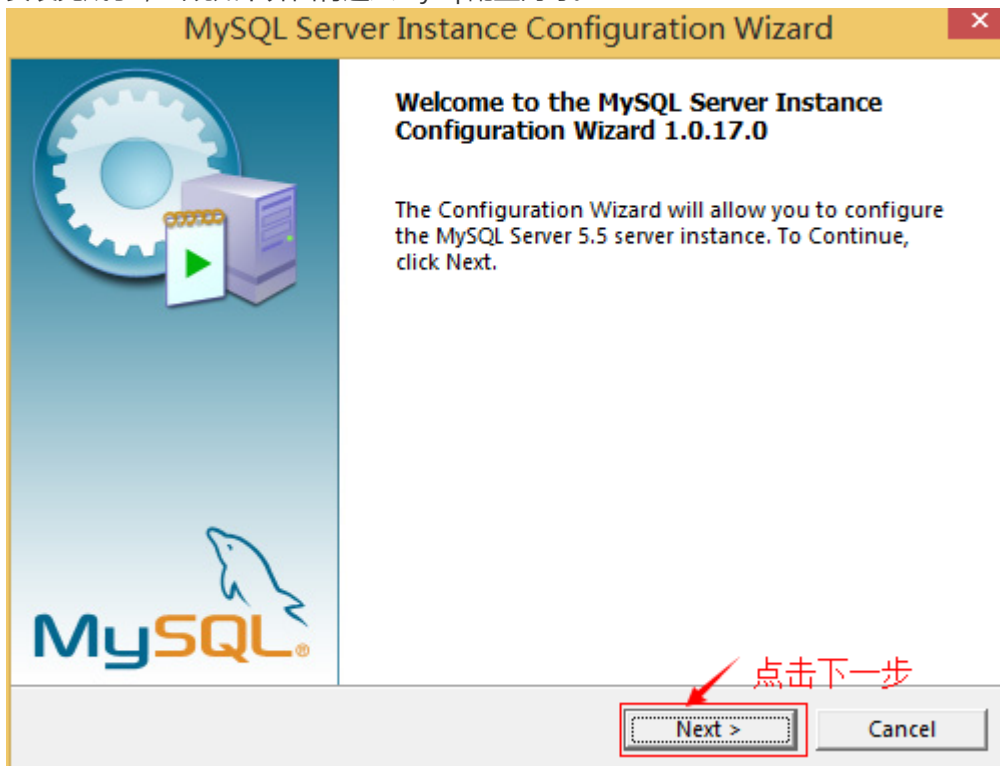


6. 正在安装中，请稍候，直到出现下面的界面，则完成MYSQL的安装

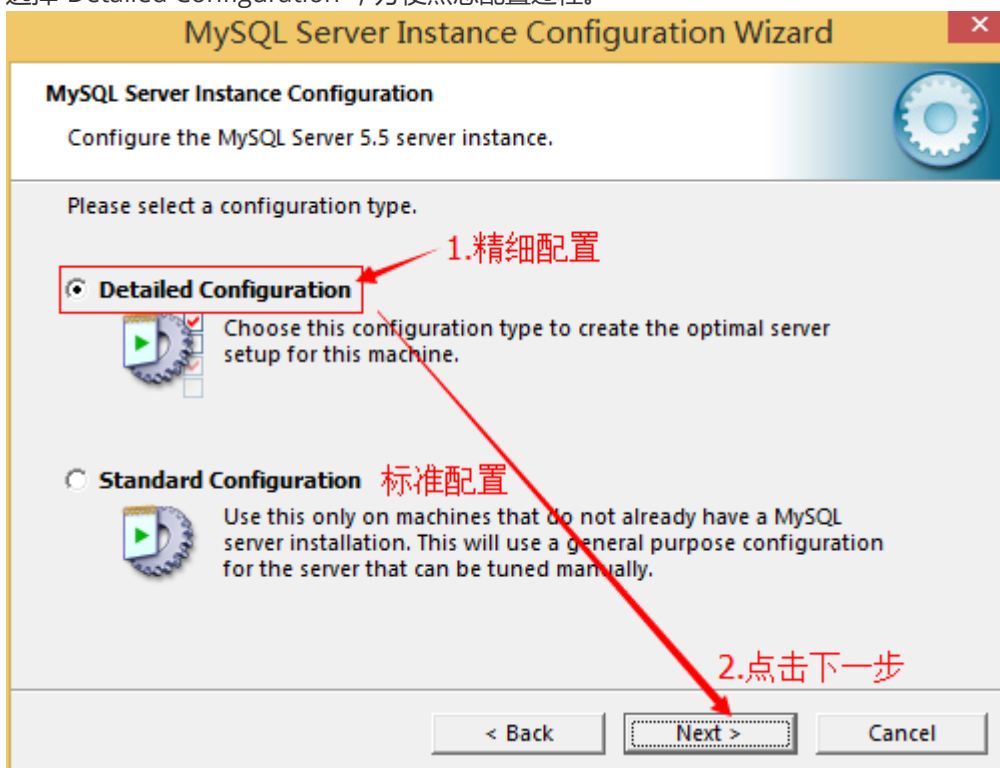


数据库安装好了还需要对数据库进行配置才能使用 MySQL的配置

1. 安装完成了，出现如下界面将进入mysql配置向导。

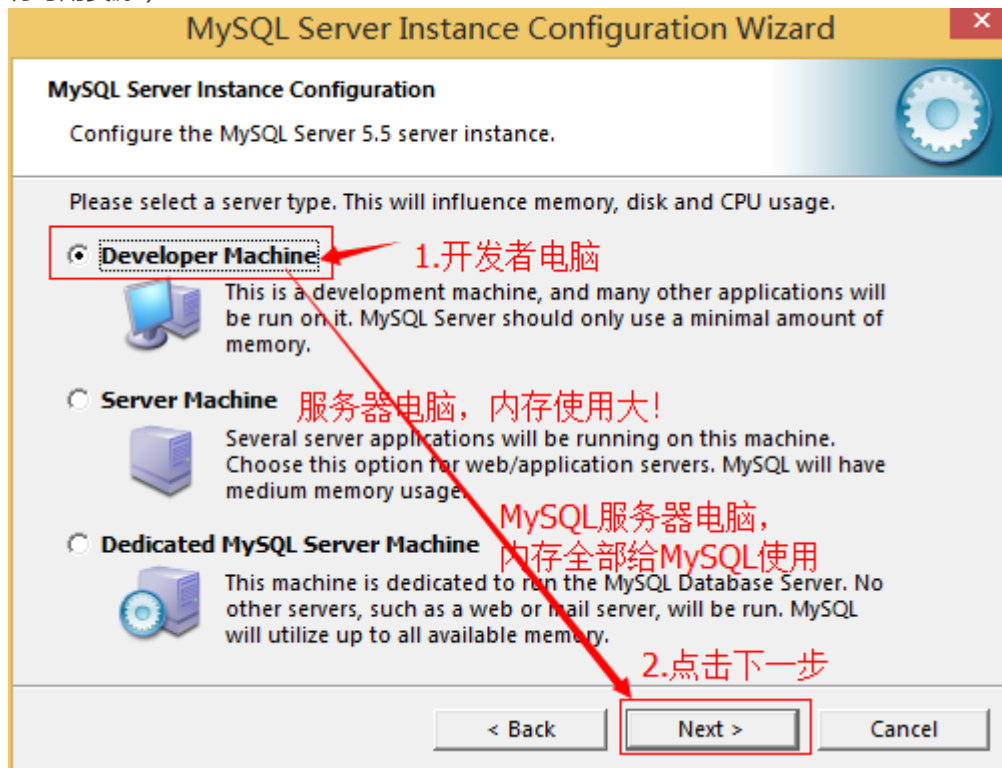


2. 选择配置方式，“Detailed Configuration (手动精确配置)”、“Standard Configuration (标准配置)”，我们选择“Detailed Configuration”，方便熟悉配置过程。

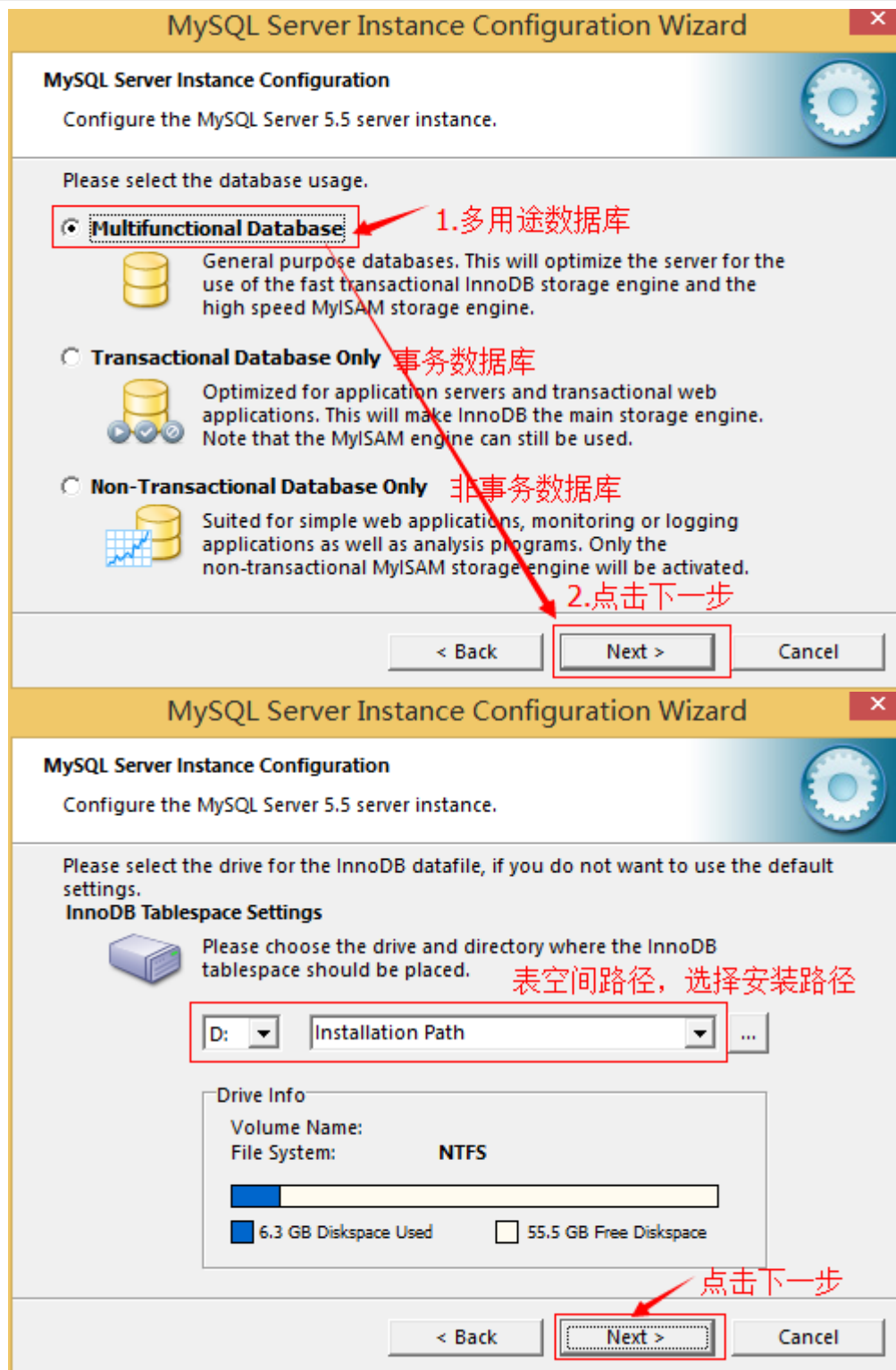


3. 选择服务器类型，“Developer Machine (开发测试类，mysql占用很少资源)”、“Server Machine (服务器类型，mysql占用较多资源)”、“Dedicated MySQL Server Machine (专门的数据库服务器，mysql占用所

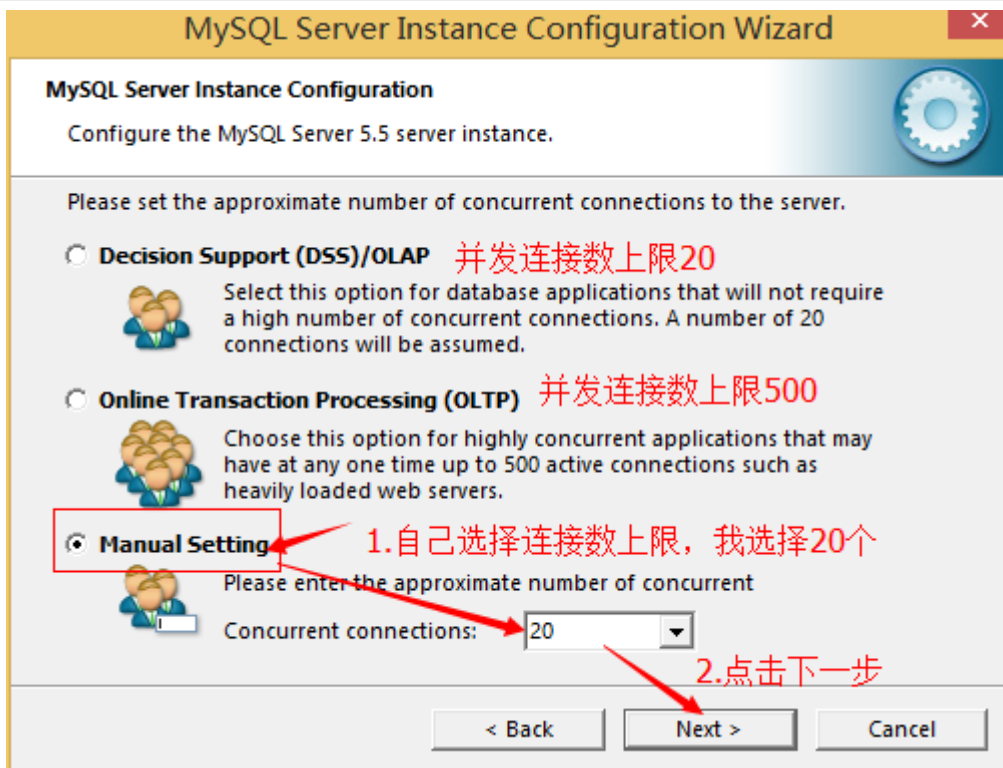
有可用资源) ”



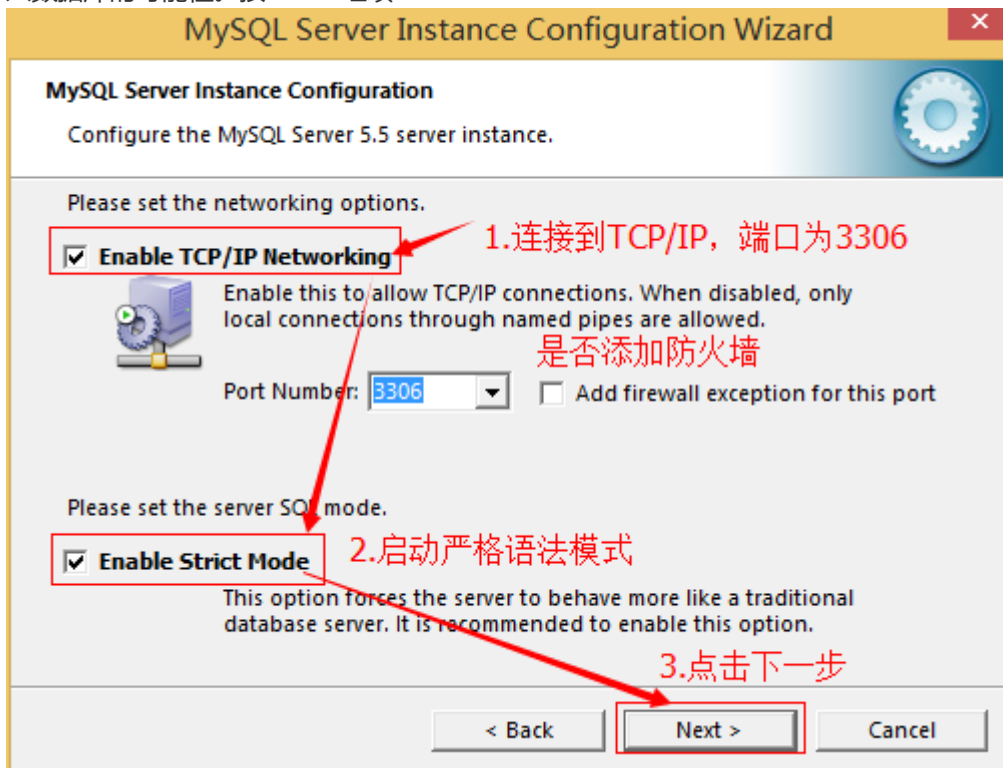
4. 选择mysql数据库的大致用途，“Multifunctional Database（通用多功能型，好）”、“Transactional Database Only（服务器类型，专注于事务处理，一般）”、“Non-Transactional Database Only（非事务处理型，较简单，主要做一些监控、记数用，对MyISAM数据类型的支持仅限于non-transactional），按“Next”继续。



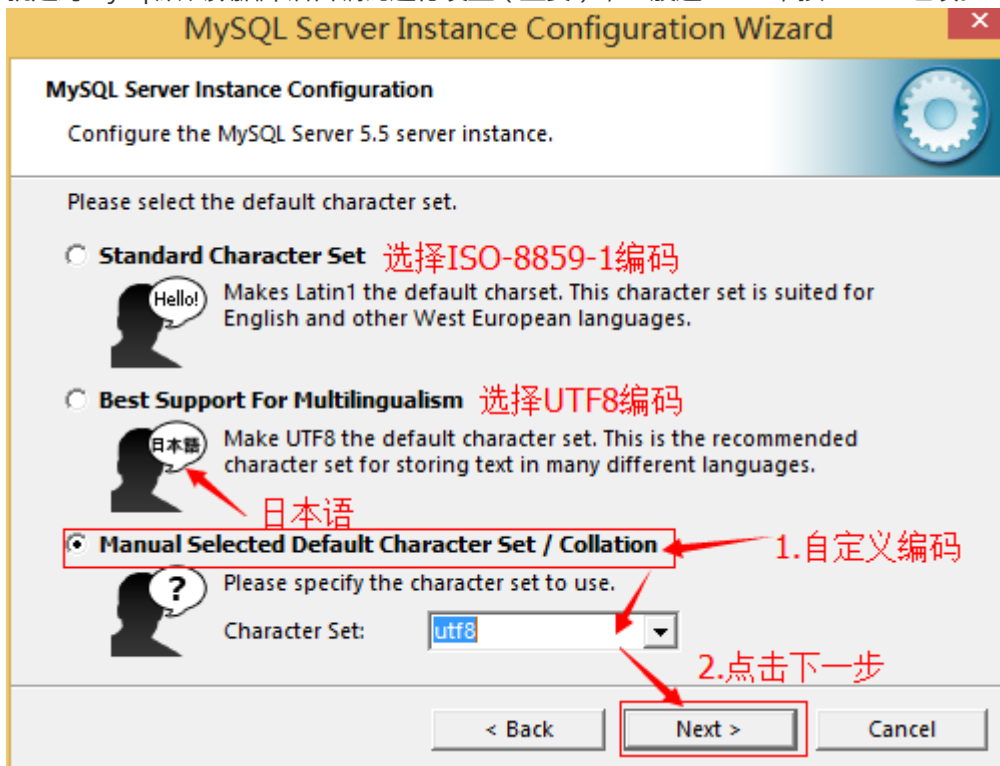
5. 选择网站并发连接数，同时连接的数目，“Decision Support(DSS)/OLAP（20个左右）”、“Online Transaction Processing(OLTP)（500个左右）”、“Manual Setting（手动设置，自己输一个数）”。



6. 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



7. 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。

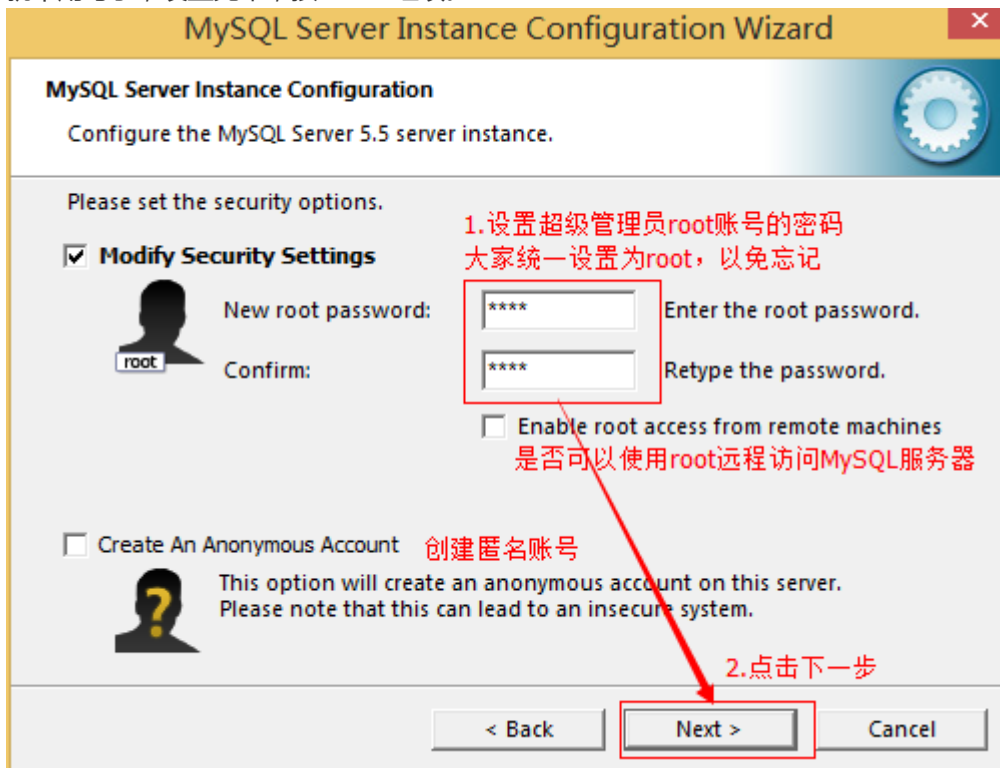


8. 选择是否将mysql安装为windows服务，还可以指定Service Name（服务标识名称），是否将mysql的bin目录加入到Windows PATH（加入后，就可以直接使用bin下的文件，而不用指出目录名，比如连接，“mysql.exe -uusername -ppassword;”就可以了，不用指出mysql.exe的完整地址，很方便），我这里全部打上了勾，Service Name不变。按“Next”继续。

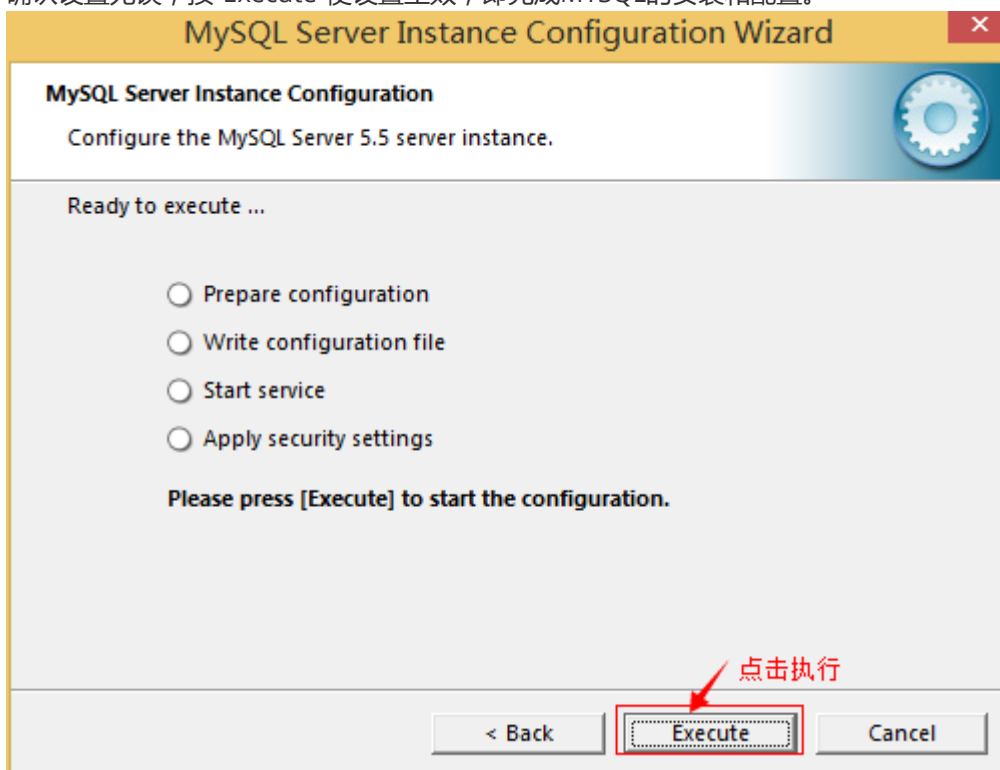


9. 询问是否要修改默认root用户（超级管理）的密码。“Enable root access from remote machines（是否允许root用户在其它的机器上登陆，如果要安全，就不要勾上，如果要方便，就勾上它）”。最后“Create An Anonymous Account（新建一个匿名用户，匿名用户可以连接数据库，不能操作数据，包括查询）”，一般

就不用勾了，设置完毕，按“Next”继续。

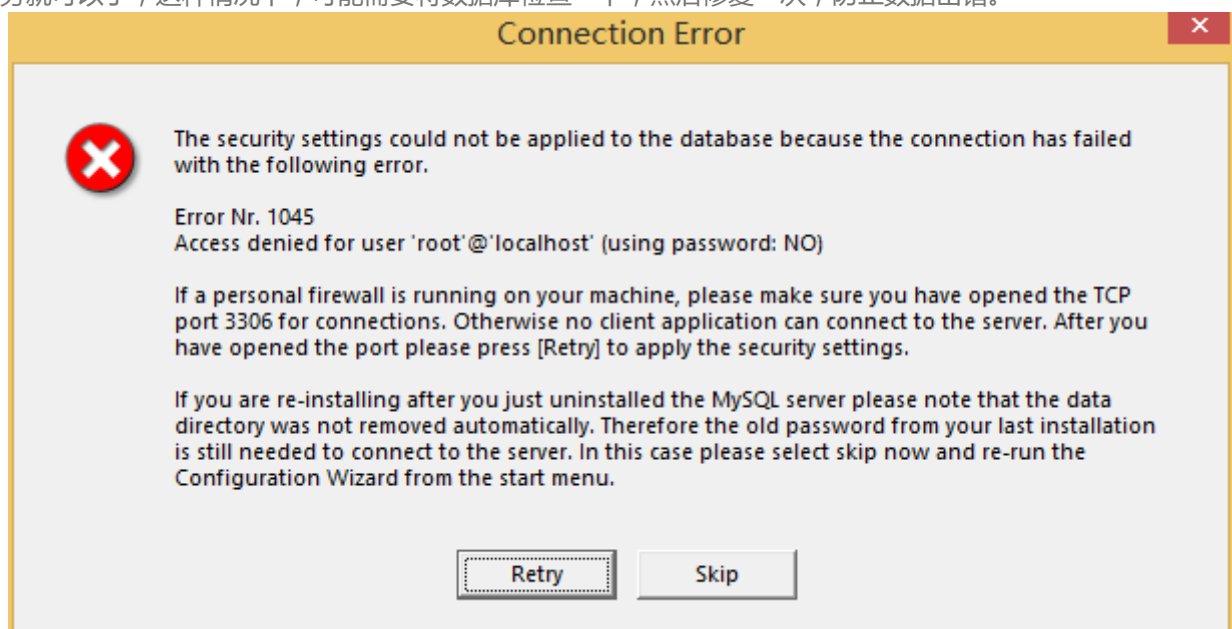


10. 确认设置无误，按“Execute”使设置生效，即完成MYSQL的安装和配置。





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。

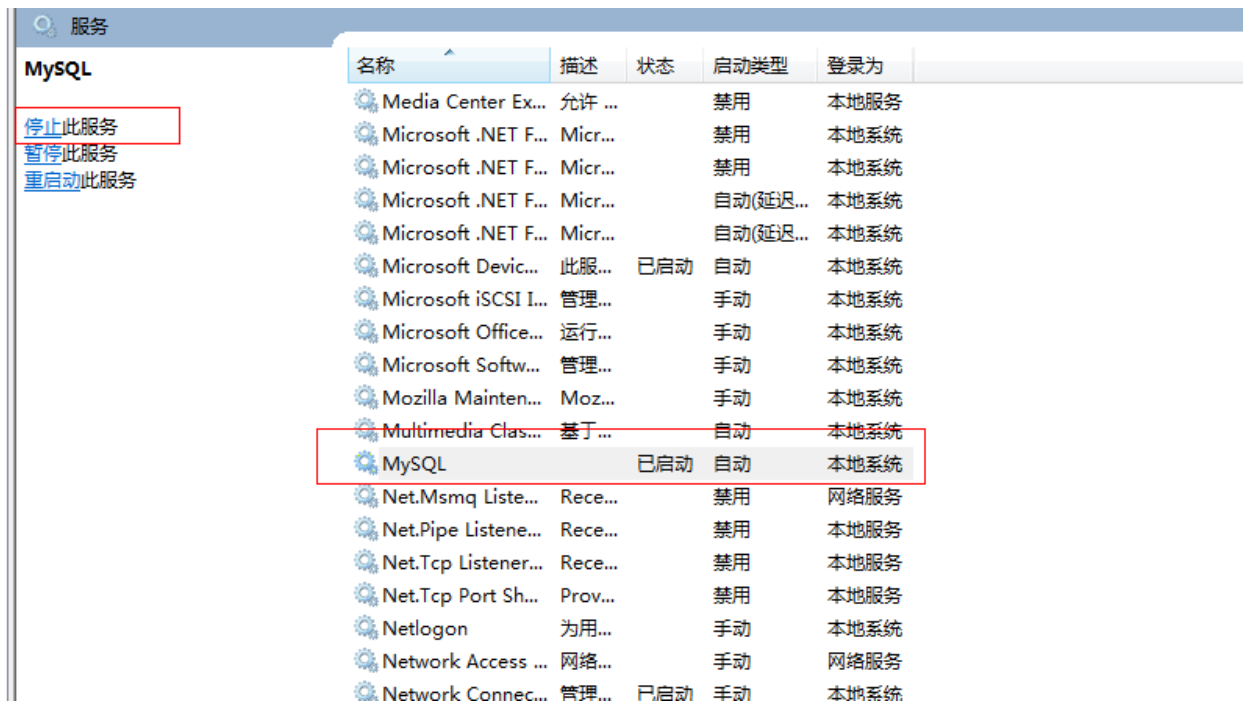


解决方法：卸载MySQL,重装MySQL

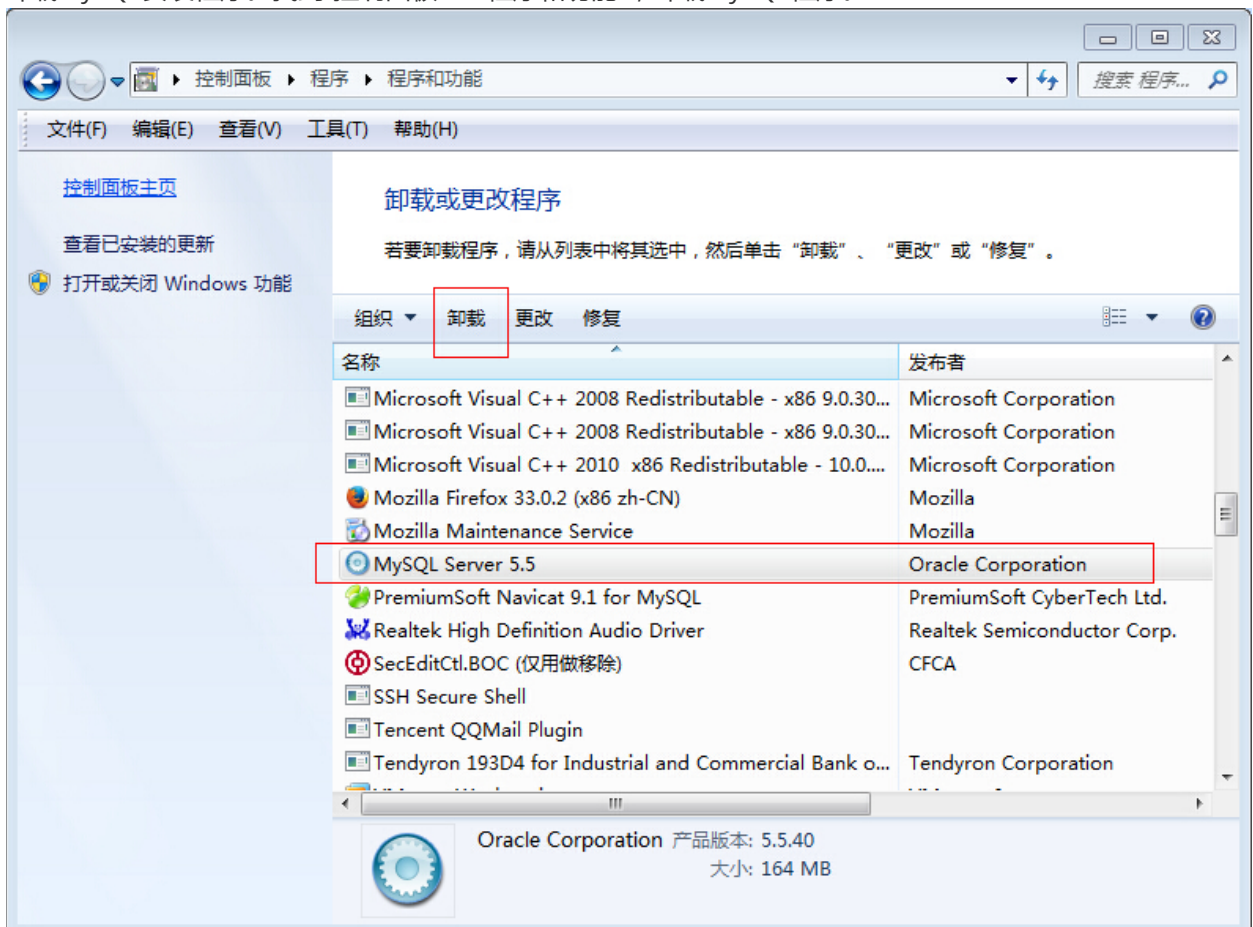
2.3 数据库的卸载

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。

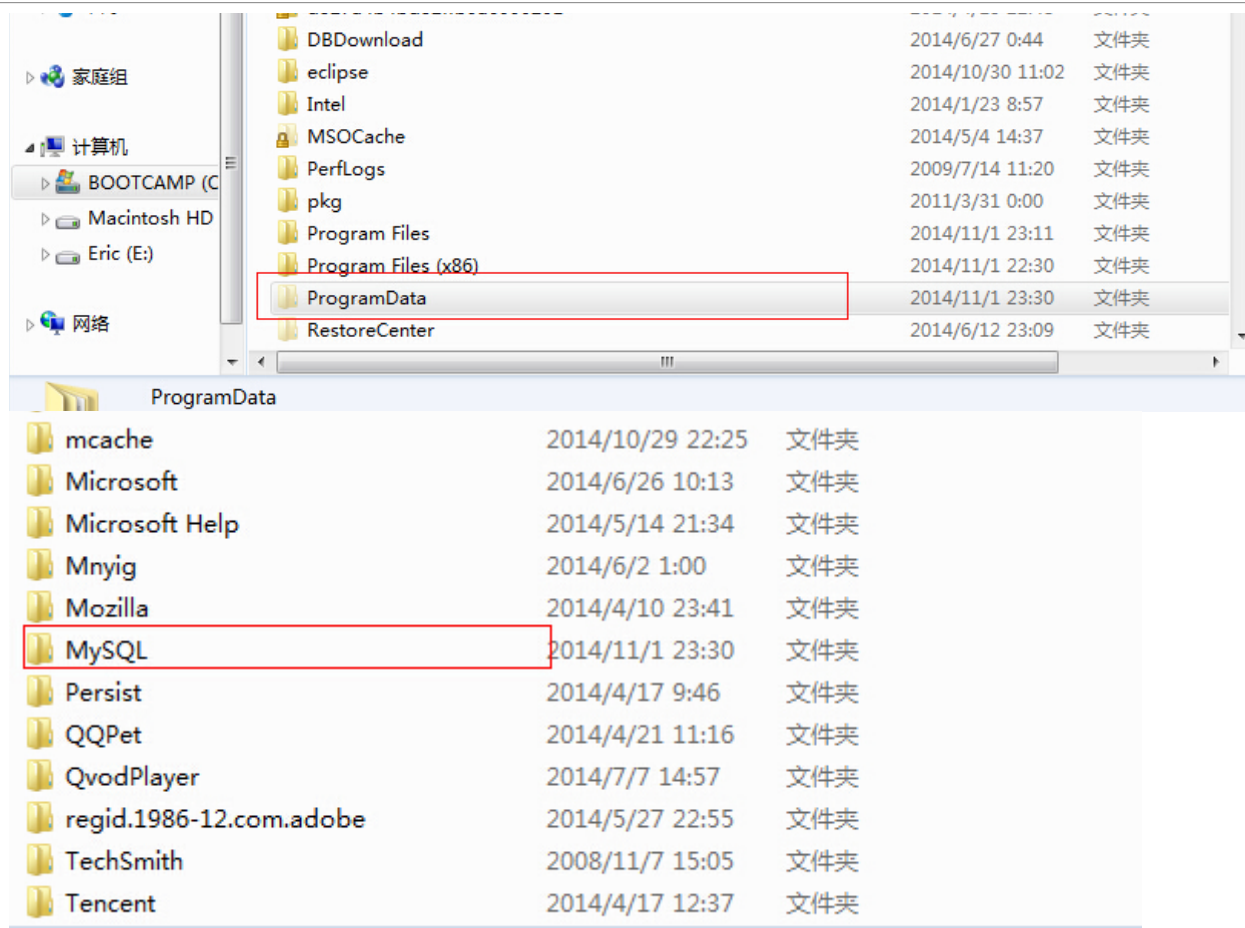


2. 卸载MySQL安装程序。找到“控制面板”->“程序和功能”，卸载MySQL程序。



3. 删除MySQL安装目录下的所有文件。

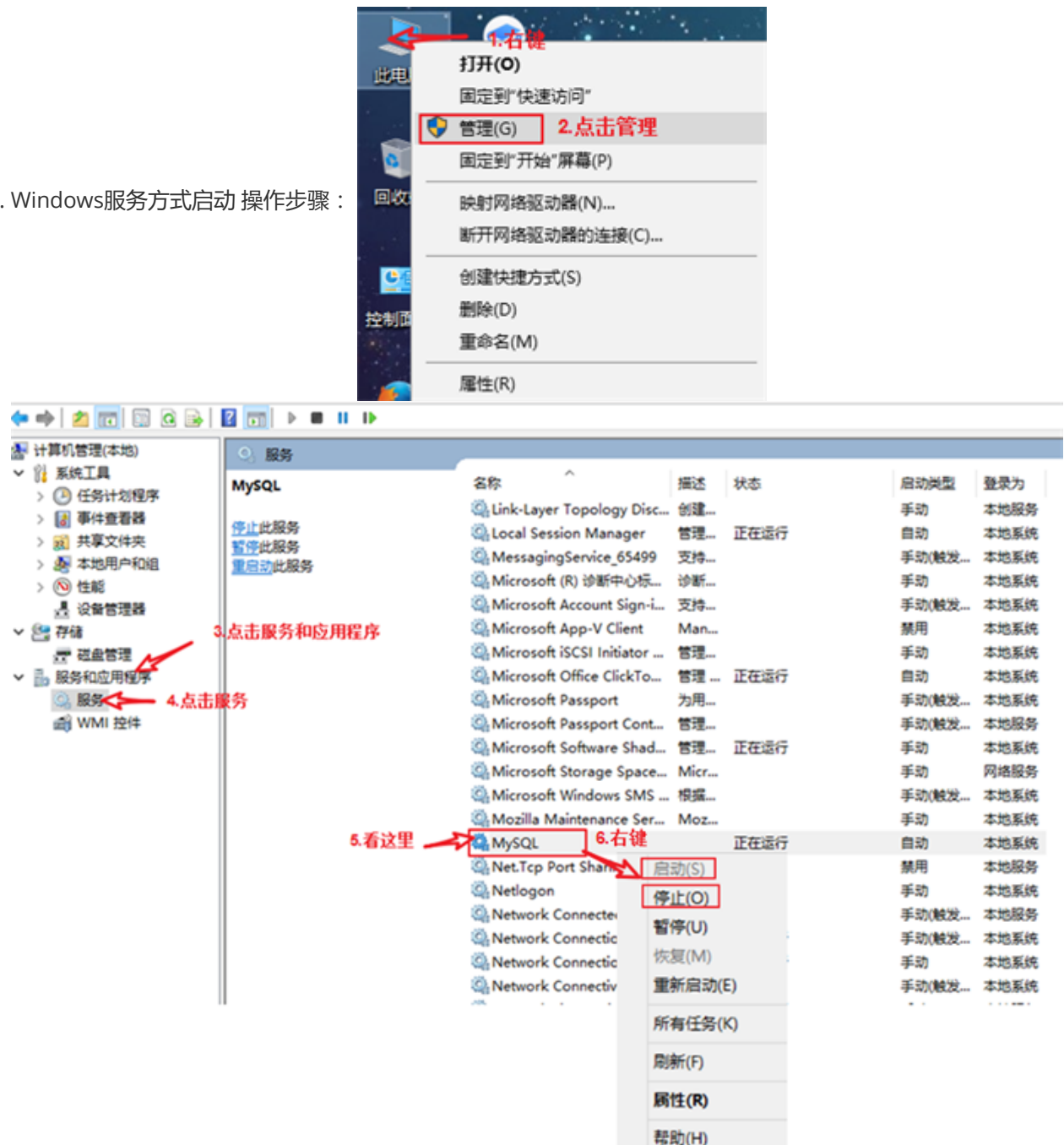
4. 删除c盘ProgramData目录中关于MySQL的目录。路径为：C:\ProgramData\MySQL(是隐藏文件,需要显示出来)



2.4 数据库的启动

MySQL启动方式和普通的windows程序双击启动方式不同，分为以下2种：

1. Windows服务方式启动 操作步骤：



2. DOS命令方式启动 操作步骤：
- 命令提示符 桌面应用
 - 以管理员身份运行
 - 打开文件所在的位置
 - 固定到“开始”屏幕
 - 固定到任务栏

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

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

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

启动MySQL: `net start mysql` 停止MySQL: `net stop mysql`

2.5 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

1. 登录格式1: `mysql -u用户名 -p密码` 例如: `mysql -uroot -proot`

```
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：

`mysql -u用户名 -p回车`

密码

```
C:\Windows\system32>mysql -uroot -p 先输入账号,后密码
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.40 MySQL Community Server (GPL)
```



2. 登录格式2: `mysql -hip地址 -u用户名 -p密码` 例如: `mysql -h127.0.0.1 -uroot -proot`

```
C:\Windows\system32>mysql -h127.0.0.1 -uroot -proot
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)

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>
```

3. 登录格式3: `mysql --host=ip地址 --user=用户名 --password=密码` 例如: `mysql --host=localhost --user=root --password=root`

```
C:\Windows\system32>mysql --host=127.0.0.1 --user=root --password=root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
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>
```

4. 退出MySQL: `exit`

```
C:\Users\zhangping>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> exit
Bye


C:\Users\zhangping>
```

exit表示退出mysql


2.6 SQLyog图形化工具安装

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



1. 双击  SQLyog-11.3.3-0.x64.exe
2. 一直下一步,直到出现下面对话框



3. 双击  SQLyog-11.3.3-0.Regged.reg 进行注册
4. 重启SQLyog即可

5. 使用SQLyog登录数据库



2.7 MySQL目录结构

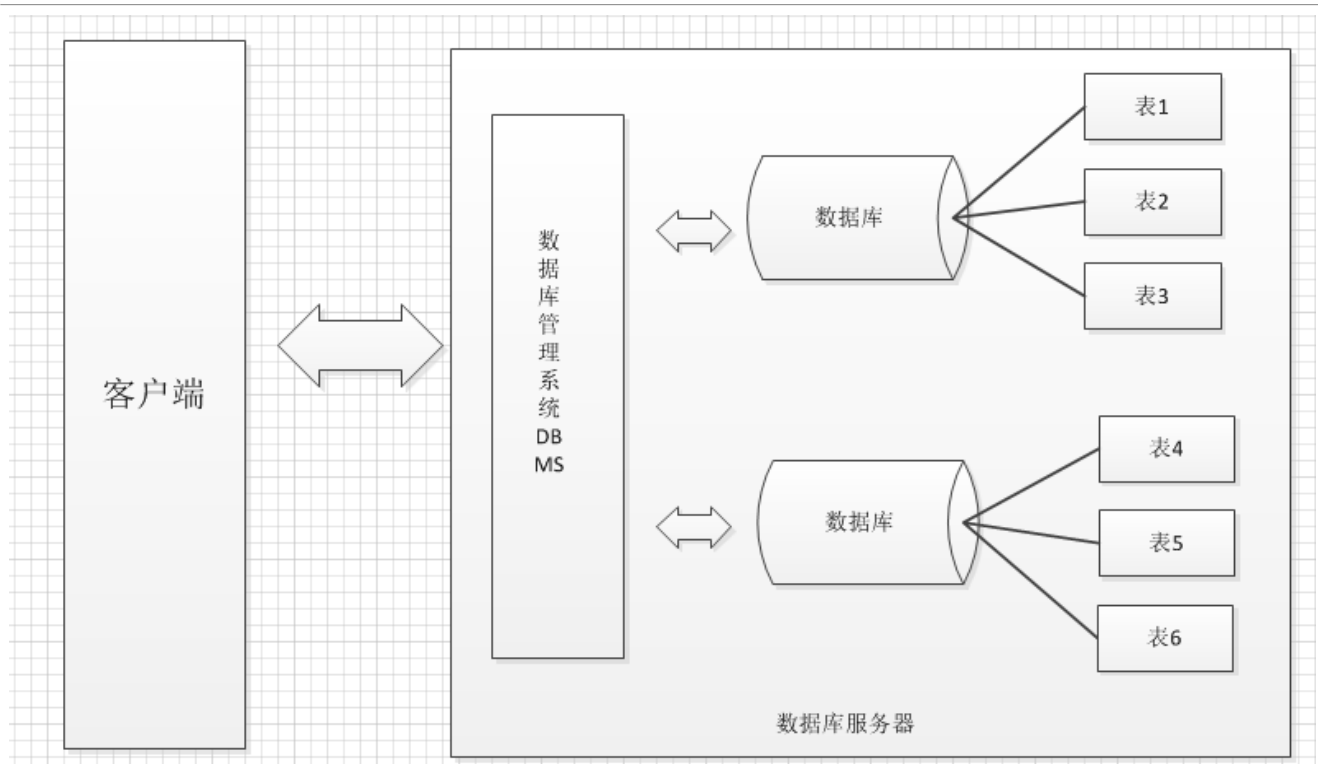
```
-- bin: mysql相关的可执行文件*.exe
|-- MySQLInstanceConfig.exe mysql的配置程序
|-- data: mysql自带的数据库文件
|-- include: c语言的头文件(不用关注)
|-- lib: 存放mysql使用到的dll动态库(相当于jar包, 不用关注)
|-- my.ini mysql的配置文件, 配置了mysql的相关信息
```

2.8 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

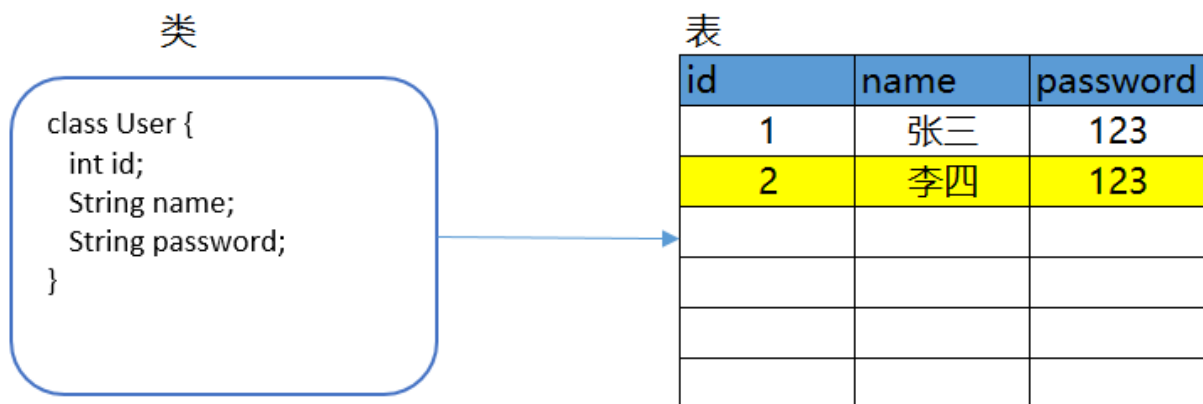
2.9 数据库管理系统、数据库和表的关系

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:



先有数据库 → 再有表 → 再有数据 一个库包含多个表

2.10 实体类与表的对应关系



`new User(1,"张三","123")`
`new User(2,"李四","123")`

一行称为一条记录

Java中的类对应数据库中表
Java中的一个对象对应表中的一条记录

第3章 SQL语句

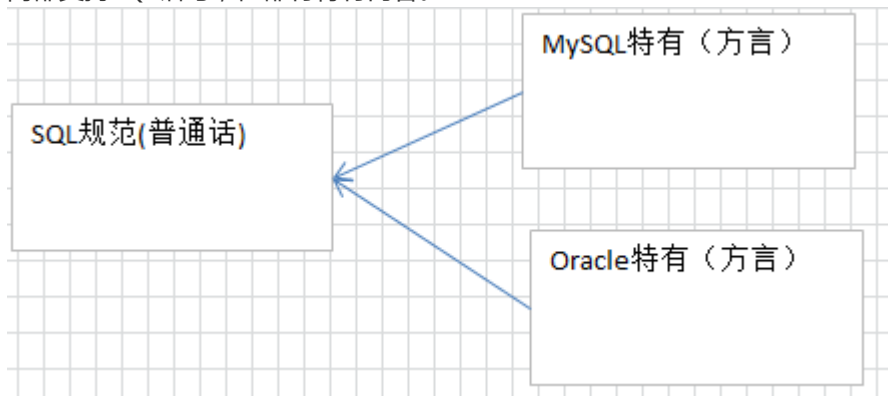
3.1 SQL的概念

3.1.1 什么是SQL

结构化查询语言(Structured Query Language)简称SQL,SQL语句就是对数据库进行操作的一种语言。

3.1.2 SQL作用

通过SQL语句我们可以方便的操作数据库中的数据、表、数据库。SQL是数据库管理系统都需要遵循的规范。不同的数据库生产厂商都支持SQL语句，但都有特有内容。



3.1.2 SQL语句分类

1. DDL(Data Definition Language)数据定义语言
用来定义数据库对象：数据库，表，列等。关键字：create, drop, alter等
2. DML(Data Manipulation Language)数据操作语言
用来对数据库中表的数据进行增删改。关键字：insert, delete, update等
3. DQL(Data Query Language)数据查询语言
用来查询数据库中表的记录(数据)。关键字：select, where等
4. DCL(Data Control Language)数据控制语言(了解)
用来定义数据库的访问权限和安全级别，及创建用户。关键字：GRANT，REVOKE等

3.2 SQL通用语法

1. SQL语句可以单行或多行书写，以分号结尾。
2. 可使用空格和缩进来增强语句的可读性。
3. MySQL数据库的SQL语句不区分大小写，关键字建议使用大写。

```
SELECT * FROM student;
```

4. 3种注释 单行注释: -- 注释内容 或 # 注释内容(mysql特有) 多行注释: /* 注释 */

3.3 DDL语句

3.3.1 DDL操作数据库

3.3.1.1 创建数据库

1. 直接创建数据库

```
CREATE DATABASE 数据库名;
```

2. 判断是否存在并创建数据库

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

3. 创建数据库并指定字符集(编码表)

```
CREATE DATABASE 数据库名 CHARACTER SET 字符集;
```

4. 具体操作：

- 直接创建数据库db1

```
CREATE DATABASE db1;
```

```
mysql> create database db1;  
Query OK, 1 row affected (0.00 sec)
```

- 判断是否存在并创建数据库db2

```
CREATE DATABASE IF NOT EXISTS db2;
```

```
mysql> CREATE DATABASE IF NOT EXISTS db2;  
Query OK, 1 row affected (0.00 sec)
```

- 创建数据库并指定字符集为gbk

```
CREATE DATABASE db2 CHARACTER SET gbk;
```

```
mysql> CREATE DATABASE db3 CHARACTER SET gbk;  
Query OK, 1 row affected (0.01 sec)
```

3.3.1.2 查看数据库

1. 查看所有的数据库

```
SHOW databases;
```

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| test |  
+-----+  
4 rows in set (0.00 sec)
```

2. 查看某个数据库的定义信息

```
SHOW CREATE DATABASE 数据库名;
```

```
mysql> show create database test;  
+-----+-----+  
| Database | Create Database |  
+-----+-----+  
| test | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */ |  
+-----+-----+  
1 row in set (0.00 sec)
```

3.3.1.3 修改数据库

修改数据库字符集格式

ALTER DATABASE 数据库名 DEFAULT CHARACTER SET 字符集;

具体操作：

- 将db3数据库的字符集改成utf8

```
ALTER DATABASE db3 DEFAULT CHARACTER SET utf8;
```

```
mysql> SHOW CREATE DATABASE db3;
+-----+-----+
| Database | Create Database |
+-----+-----+
| db3      | CREATE DATABASE `db3` /*!40100 DEFAULT CHARACTER SET gbk */ |
+-----+-----+
1 row in set (0.00 sec)

mysql> ALTER DATABASE db3 DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> ALTER DATABASE db3 DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW CREATE DATABASE db3;
+-----+-----+
| Database | Create Database |
+-----+-----+
| db3      | CREATE DATABASE `db3` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+-----+-----+
1 row in set (0.00 sec)
```

3.3.1.4 删除数据库

DROP DATABASE 数据库名;

具体操作：

- 删除db2数据库

```
DROP DATABASE db2;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| day24 |
| db2 |
| db3 |
| mysql |
| performance_schema |
| test |
+-----+
7 rows in set (0.00 sec)

mysql> DROP DATABASE db2;
Query OK, 0 rows affected (0.02 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| day24 |
| db3 |
| mysql |
| performance_schema |
| test |
+-----+
6 rows in set (0.00 sec)

mysql>
```

db2数据库已经被删除了

3.3.1.5 使用数据库

1. 查看正在使用的数据库

```
SELECT DATABASE();
```

2. 使用/切换数据库

```
USE 数据库名;
```

具体操作：

- 查看正在使用的数据库

```
SELECT DATABASE();
```

```
mysql> select database();
+-----+
| database() |
+-----+
| NULL |
+-----+
1 row in set (0.00 sec)
```

- 使用db1数据库

```
USE db1;
```



```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)

mysql> USE db1;
Database changed
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| db1       |
+-----+
1 row in set (0.00 sec)

mysql>
```

3.3.2 DDL操作表

前提先使用某个数据库

3.3.2.1 创建表

```
CREATE TABLE 表名 (字段名1 字段类型1, 字段名2 字段类型2...);
```

建议写成如下格式:

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

关键字说明：

```
CREATE -- 表示创建
TABLE  -- 表示创建一张表
```

MySQL数据类型 MySQL中的我们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；↕

详细的数据类型如下(不建议详细阅读！)

分类	类型名称	说明
整数类型	tinyInt	很小的整数
	smallint	小的整数
	mediumint	中等大小的整数
	int(integer)	普通大小的整数
小数类型	float	单精度浮点数
	double	双精度浮点数
	decimal (m,d)	压缩严格的定点数
日期类型	year	YYYY 1901~2155
	time	HH:MM:SS -838:59:59~838:59:59
	date	YYYY-MM-DD 1000-01-01~9999-12-31
	datetime	YYYY-MM-DD HH:MM:SS 1000-01-01 00:00:00~ 9999-12-31 23:59:59
	timestamp	YYYY-MM-DD HH:MM:SS 1970~01~01 00:00:01 UTC~2038-01-19 03:14:07UTC
文本、二进制类型	CHAR(M)	M为0~255之间的整数
	VARCHAR(M)	M为0~65535之间的整数
	TINYBLOB	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
	LOB	允许长度0~4294967295字节
	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
	VARBINARY(M)	允许长度0~M个字节的变长字节字符串
	BINARY(M)	允许长度0~M个字节的定长字节字符串

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  
);
```

3.3.2.2 查看表

1. 查看某个数据库中的所有表

```
SHOW TABLES;
```

2. 查看表结构

```
DESC 表名;
```

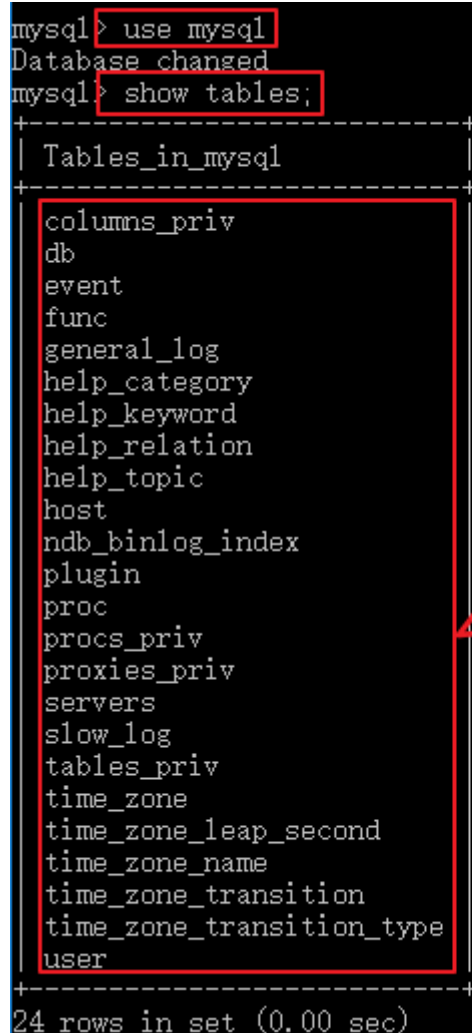
3. 查看创建表的SQL语句

```
SHOW CREATE TABLE 表名;
```

具体操作：

- 查看mysql数据库中的所有表

```
SHOW TABLES;
```



```
mysql> use mysql  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_mysql |  
+-----+  
| columns_priv  
| db  
| event  
| func  
| general_log  
| help_category  
| help_keyword  
| help_relation  
| help_topic  
| host  
| ndb_binlog_index  
| plugin  
| proc  
| procs_priv  
| proxies_priv  
| servers  
| slow_log  
| tables_priv  
| time_zone  
| time_zone_leap_second  
| time_zone_name  
| time_zone_transition  
| time_zone_transition_type  
| user  
+-----+  
24 rows in set (0.00 sec)
```


- 查看student表的结构

```
DESC student;
```

```
mysql> desc student;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11) | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| birthday | date   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 查看student的创建表SQL语句

```
SHOW CREATE TABLE student;
```

```
mysql> show create table student;
+-----+-----+
| Table | Create Table |
+-----+-----+
| student | CREATE TABLE `student` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(20) DEFAULT NULL,
  `birthday` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+-----+
1 row in set (0.00 sec)
```

3.3.2.3 快速创建一个表结构相同的表

```
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;
```

```
mysql> create table s1 like student;
Query OK, 0 rows affected (0.02 sec)

mysql> SHOW CREATE TABLE s1;
+-----+-----+
| Table | Create Table |
+-----+-----+
| s1    | CREATE TABLE `s1` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(20) DEFAULT NULL,
  `birthday` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+-----+
1 row in set (0.00 sec)
```

3.3.2.4 删除表

1. 直接删除表

DROP TABLE 表名;

2. 判断表是否存在并删除表

DROP TABLE IF EXISTS 表名;

具体操作：

- 直接删除表s1表

DROP TABLE s1;

```
mysql> show tables;
+-----+
| Tables_in_db1 |
+-----+
| s1             |
| s2             |
| s3             |
| student        |
+-----+
4 rows in set (0.00 sec)

mysql> DROP TABLE s1;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+-----+
| Tables_in_db1 |
+-----+
| s2             |
| s3             |
| student        |
+-----+
3 rows in set (0.00 sec)
```

s1表被删除了

- 判断表是否存在并删除s1表

```
DROP TABLE IF EXISTS s1;
```

```
mysql> show tables;
+-----+
| Tables_in_db1 |
+-----+
| s2             |
| s3             |
| student        |
+-----+
3 rows in set (0.00 sec)

mysql> DROP TABLE s1;
ERROR 1051 (42S02): Unknown table 's1'
mysql> DROP TABLE IF EXISTS s1;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

s1表不存在，直接删除会报错

增加了判断后表不存在不会报错

3.3.2.5 修改表结构

修改表结构使用不是很频繁，只需要了解，等需要使用的时候再回来查即可

1. 添加表列 `ALTER TABLE 表名 ADD 列名 类型;`

具体操作：

- 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);
```

```
mysql> desc s1;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | YES  |     | NULL    |       |
| name       | varchar(20)   | YES  |     | NULL    |       |
| birthday   | date          | YES  |     | NULL    |       |
+-----+
3 rows in set (0.01 sec)

mysql> alter table s1 add remark varchar(20);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc s1;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | YES  |     | NULL    |       |
| name       | varchar(20)   | YES  |     | NULL    |       |
| birthday   | date          | YES  |     | NULL    |       |
| remark     | varchar(20)   | YES  |     | NULL    |       |
+-----+
4 rows in set (0.00 sec)
```

2. 修改列类型 `ALTER TABLE 表名 MODIFY列名 新的类型;` 具体操作：

- 将student表中的remark字段的改成varchar(100)

```
ALTER TABLE student MODIFY remark VARCHAR(100);
```

```
mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| birthday | date     | YES  |     | NULL    |       |
| remark | varchar(20) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> alter table s1 modify remark varchar(100);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| birthday | date     | YES  |     | NULL    |       |
| remark | varchar(100) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

3. 修改列名 ALTER TABLE 表名 CHANGE 旧列名 新列名 类型; 具体操作：

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

```
ALTER TABLE student CHANGE remark intro varchar(30);
```

```
mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| birthday | date     | YES  |     | NULL    |       |
| remark | varchar(100) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> alter table s1 change remark intro varchar(30);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| birthday | date     | YES  |     | NULL    |       |
| intro | varchar(30) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

4. 删除列 ALTER TABLE 表名 DROP 列名; 具体操作：

- 删除student表中的字段intro

```
ALTER TABLE student DROP intro;
```

```
mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | YES  |     | NULL    |       |
| name  | varchar(20)   | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
| intro | varchar(30)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> alter table s1 drop intro;
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc s1;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | YES  |     | NULL    |       |
| name  | varchar(20)   | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

5. 修改表名 RENAME TABLE 表名 TO 新表名; 具体操作：

- 将学生表student改名为student2

```
RENAME TABLE student TO student2;
```

```
mysql> rename table student to student2;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables
+-----+
| Tables_in_day21 |
+-----+
| employee        |
| s1               |
| s3               |
| student2        |
+-----+
4 rows in set (0.00 sec)
```

6. 修改字符集 ALTER TABLE 表名 character set 字符集; 具体操作：

- 将sutden2表的编码修改成gbk

```
ALTER TABLE student2 character set gbk;
```

```
mysql> show create table student2;
+-----+-----+
| Table | 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 |
+-----+-----+
1 row in set (0.00 sec)

mysql> alter table student2 character set gbk;
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show create table student2;
+-----+-----+
| Table | Create Table |
+-----+-----+
| student2 | 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 |
+-----+-----+
1 row in set (0.00 sec)
```

3.4 DML语句

3.4.1 插入记录

3.4.1.1 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...);
```

3.4.1.2 插入部分数据

`INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...);` 没有添加数据的字段会使用NULL

1. 关键字说明

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

2. 注意

- 值与字段必须对应，个数相同，类型相同
- 值的数据大小必须在字段的长度范围内
- 除了数值类型外，其它的字段类型的值必须使用引号引起。（建议单引号）
- 如果要插入空值，可以不写字段，或者插入null

3. 具体操作:

- 插入部分数据，往学生表中添加 id, name, age, sex数据

```
INSERT INTO student (id, NAME, age, sex) VALUES (1, '张三', 20, '男');
```

```
INSERT INTO student (id, NAME, age, sex) VALUES (1, '张三', 20, '男');
```

没有添加数据的字段会使用NULL

id	name	age	sex	address
1	张三	20	男	(NULL)

- 向表中插入所有字段

- 所有的字段名都写出来

```
INSERT INTO student (NAME, id, age, sex, address) VALUES ('李四', 2, 23, '女', '广州');
```

```
INSERT INTO student (NAME, id, age, sex, address) VALUES ('李四', 2, 23, '女', '广州');
```

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州

所有字段都添加数据

- 不写字段名

```
INSERT INTO student VALUES (3, '王五', 18, '男', '北京');
```

```
-- 不需要写字段名,就是添加所有字段  
-- INSERT INTO 表名 VALUES (值1, 值2, 值3...);  
INSERT INTO student VALUES (3, '王五', 18, '男', '吉山');
```

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州
3	王五	18	男	吉山
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

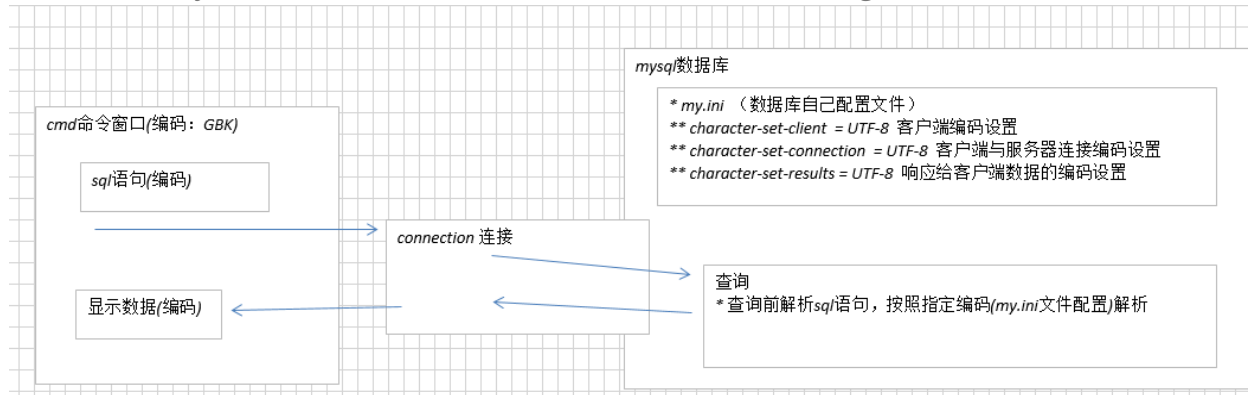
3.4.1.3 DOS命令窗口操作数据乱码问题的解决

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，导致SQL执行失败

```
mysql> INSERT INTO student VALUES (3, '王五', 18, '男', '北京');
ERROR 1366 (HY000): Incorrect string value: '\xCD\xF5\xCE\xE5' for column 'name' at row 1
mysql>
```

中文出现乱码了

错误原因:因为MySQL的客户端设置编码是utf8,而系统的DOS命令行编码是gbk，编码不一致导致的乱码



查看 MySQL 内部设置的编码 `show variables like 'character%';`

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

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

1. 单独设置

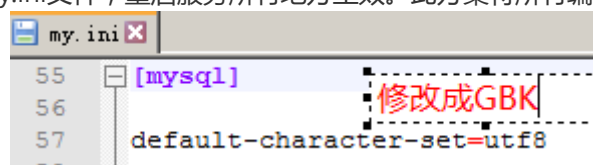
```
set character_set_client=gbk;
set character_set_connection=gbk;
set character_set_results=gbk;
```

2. 快捷设置

```
set names gbk;
```

注意：以上2种方式为临时方案，退出DOS命令行就失效了，需要每次都配置

3. 修改MySQL安装目录下的my.ini文件，重启服务所有地方生效。此方案将所有编码都修改了[不建议]



3.4.1.4 蠕虫复制

什么是蠕虫复制：在已有的数据基础之上，将原来的数据进行复制，插入到对应的表中 语法格式：`INSERT INTO 表名1 SELECT * FROM 表名2;` 作用:将 表名2 中的数据复制到 表名1 中

具体操作:

- 创建student2表，student2结构和student表结构一样

```
CREATE TABLE student LIKE student2;
```

- 将student表中的数据添加到student2表中

```
INSERT INTO student SELECT * FROM student2;
```

注意：如果只想复制student表中name,age字段数据到student2表中使用如下格式 `INSERT INTO student2(NAME, age) SELECT NAME, age FROM student;`

`INSERT INTO student2(NAME, age) SELECT NAME, age FROM student;`

放到目标表的哪些字段 复制原表的哪些字段

id	name	age	sex	address	math	english
1	张三	20	男	(NULL)	67	96
2	李四	23	女	广州	88	92
3	王五	18	男	北京	56	65
4	赵六	28	男	南京	99	98

id	name	age	sex	address	math	english
1	张三	20	男	(NULL)	67	96
2	李四	23	女	广州	88	92
3	王五	18	男	北京	56	65
4	赵六	28	男	南京	99	98
(NULL)	张三	20	(NULL)	(NULL)	(NULL)	(NULL)
(NULL)	李四	23	(NULL)	(NULL)	(NULL)	(NULL)
(NULL)	王五	18	(NULL)	(NULL)	(NULL)	(NULL)
(NULL)	赵六	28	(NULL)	(NULL)	(NULL)	(NULL)

只复制了指定字段的数据，没有指定字段的数据没有复制

3.4.2 更新表记录

- 不带条件修改数据 `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;`
- 关键字说明

UPDATE: 修改数据
SET: 修改哪些字段
WHERE: 指定条件

4. 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';
```

修改前

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州
3	王五	18	男	北京
4	赵六	28	男	南京

修改后

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	女	广州
3	王五	18	女	北京
4	赵六	28	女	南京

- 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;
```

修改前

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	女	广州
3	王五	18	女	北京
4	赵六	28	女	南京

修改后

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
3	王五	18	女	北京
4	赵六	28	女	南京

- 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;
```

修改前

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
3	王五	18	女	北京
4	赵六	28	女	南京

修改后 一次性修改2个字段

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
3	王五	26	女	北京
4	赵六	28	女	南京

3.4.3 删除表记录

- 不带条件删除数据 `DELETE FROM 表名;`
- 带条件删除数据 `DELETE FROM 表名 WHERE 字段名=值;`
- truncate删除表记录 `TRUNCATE TABLE 表名;`

truncate和delete的区别：

- delete是将表中的数据一条一条删除
- truncate是将整个表摧毁，重新创建一个新的表，新的表结构和原来表结构一模一样

```
mysql> select * from student;
+----+-----+-----+-----+-----+-----+-----+
| id | name | age | sex | address | math | english |
+----+-----+-----+-----+-----+-----+-----+
| 1  | 张三 | 20  | 女  | NULL    | 67   | 96      |
| 2  | 李四 | 23  | 男  | 广州    | 88   | 92      |
| 3  | 王五 | 18  | 男  | 北京    | 56   | 65      |
| 4  | 赵六 | 28  | 男  | 南京    | 99   | 98      |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> TRUNCATE TABLE student;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from student;
Empty set (0.00 sec)

mysql>
```

truncate 也将数据库中的数据全部删除了
truncate和delete的区别：
1.delete是将表中的数据一条一条删除
2.truncate是将整个表摧毁，重新创建一个新的表

4. 具体操作：

- 带条件删除数据，删除id为3的记录

```
DELETE FROM student WHERE id=3;
```

删除前

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
3	王五	26	女	北京
4	赵六	28	女	南京

删除后

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
4	赵六	28	女	南京

满足条件的记录被删除了

- 不带条件删除数据，删除表中的所有数据

```
DELETE FROM student;
```

删除前

id	name	age	sex	address
1	张三	20	女	(NULL)
2	李四	23	男	广州
4	赵六	28	女	南京

删除后

id	name	age	sex	address
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

所有的记录都被删除

3.5 DQL

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

3.5.1 简单查询

3.5.1.1 查询表所有数据

1. 使用*表示所有列 `SELECT * FROM 表名;` 具体操作：

```
SELECT * FROM student;
```

所有列的数据都被查询出来

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州
3	王五	18	男	北京
4	赵六	28	男	南京

2. 写出查询每列的名称 `SELECT 字段名1, 字段名2, 字段名3, ... FROM 表名;` 具体操作：

```
SELECT id, NAME ,age, sex, address FROM student;
```

所有列的数据都被查询出来

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州
3	王五	18	男	北京
4	赵六	28	男	南京

3.5.1.2 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... FROM 表名;`

具体操作：查询student表中的name 和 age 列

```
SELECT NAME, age FROM student;
```

NAME	age
张三	20
李四	23
王五	18
赵六	28

只显示查询的字段数据，没有查询的不显示

3.5.1.3 别名查询

1. 查询时给列、表指定别名需要使用AS关键字

2. 使用别名的好处是方便观看和处理查询到的数据 `SELECT 字段名1 AS 别名, 字段名2 AS 别名... FROM 表名;`

`SELECT 字段名1 AS 别名, 字段名2 AS 别名... FROM 表名 AS 表别名;` 注意:

查询给表取别名目前还看不到效果，需要到多表查询的时候才能体现出好处 AS关键字可以省略

3. 具体操作：

- 查询student表中name 和 age 列，name列的别名为“姓名”，age列的别名为“年龄”

```
SELECT NAME AS 姓名, age AS 年龄 FROM student;
```

NAME	age
张三	20
李四	23
王五	18
赵六	28

没有别名

姓名	年龄
张三	20
李四	23
王五	18
赵六	28

有别名

使用别名的好处是方便观看和处理查询到的数据

- 查询student表中name和age列，student表别名为s

```
SELECT NAME, age FROM student AS s;
```

NAME	age
张三	20
李四	23
王五	18
赵六	28

查询给表取别名目前还看不到效果，需要到多表查询的时候才能体现出好处

3.5.1.4 清除重复值

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;`

2. 具体操作：

- 查询name，age列并且结果不出现重复name

```
SELECT DISTINCT NAME, age FROM student;
```

id	name	age	sex	address
1	张三	20	男	(NULL)
2	李四	23	女	广州
3	王五	18	男	北京
4	赵六	28	男	南京
5	赵六	15	男	南京
6	王五	22	男	北京
7	赵六	38	男	南京

NAME	age
张三	20
李四	23
王五	22
赵六	15

重复的姓名只剩下一条数据

3.5.1.5 查询结果参与运算

1. 某列数据和固定值运算 `SELECT 列名1 + 固定值 FROM 表名;`
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

3. 需求：

- 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

4. 实现：

- 修改student表结构,添加数学和英语成绩列

```
ALTER TABLE student ADD math INT;
ALTER TABLE student ADD english INT;
```

- 给每条记录添加对应的数学和英语成绩

id	name	age	sex	address	math	english
1	张三	20	男	(NULL)	67	96
2	李四	23	女	广州	88	92
3	王五	22	男	北京	56	65
4	赵六	15	男	南京	99	98

- 查询math + english的和

```
SELECT math + english FROM student;
```

math + english
163
180
121
197

结果确实将每条记录的math和english相加，但是效果不好看

- 查询math + english的和使用别名“总成绩”

```
SELECT math + english 总成绩 FROM student;
```

总成绩
163
180
121
197

- 查询所有列与math + english的和并使用别名“总成绩”

```
SELECT *, math + english 总成绩 FROM student;
```

id	name	age	sex	address	math	english	总成绩
1	张三	20	男	(NULL)	67	96	163
2	李四	23	女	广州	88	92	180
3	王五	18	男	北京	56	65	121
4	赵六	28	男	南京	99	98	197

所有列数据

参与运算的数据

- 查询姓名、年龄，将每个人的年龄增加10岁

```
SELECT name, age + 10 FROM student;
```

查询前

id	name	age	sex	address	math	english
1	张三	20	男	(NULL)	67	96
2	李四	23	女	广州	88	92
3	王五	18	男	北京	56	65
4	赵六	28	男	南京	99	98

查询后年龄增加10岁

name	age + 10
张三	30
李四	33
王五	28
赵六	38