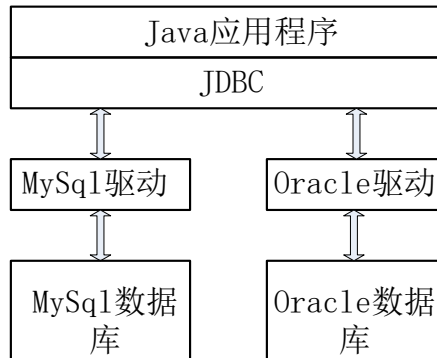


# Java 与数据库开发

章铁飞 浙江工商大学计算机学院  
tfzhang@mail.zjgsu.edu.cn

Java 基于如下的方式访问数据库：



开发人员需要负责的是 Java 应用程序与 JDBC 部分的开发；JDBC 的全称是 Java Database Connectivity，Java 数据库连接。JDBC 提供的是一组与平台无关，可用于连接数据库及执行 SQL 语句的 API。

先来看看什么是 SQL 语句。SQL，全称是 Structured Query Language，结构化查询语言，是关系数据库的标准语言。既然是语言，就离不开语句，常见的 SQL 的语句和例句如下：

	例句	含义
创建数据库	Create Database test;	创建一个名为 test 的数据库；
创建数据表	Create table student ( id varchar(10) not null, name varchar(16) not null, score int not null )	创建一个名为 stdudet 的数据表，其中包含 id, name 和 score 三个值；
插入数据	Insert into student (id, name, score) values ('12345', 'zhang3', 87);	向数据表 student 中插入一个叫 zhang3 的学生信息；
查询数据	Select id, name, score from student where id=12345	从数据表 student 中查询 id 为 12345 的学生的学号，姓名与分数；
删除数据	Delete from student where id=12345	从数据表 student 中删除 id 为 12345 的学生信息；
更新数据	Update student set score=90 where id=12345	更新数据表中 id 为 12345 的学生的分数；

sql 还有很多其他语句，主要的操作包括增删改查：

C 增加(Create)

R 读取查询(Retrieve)

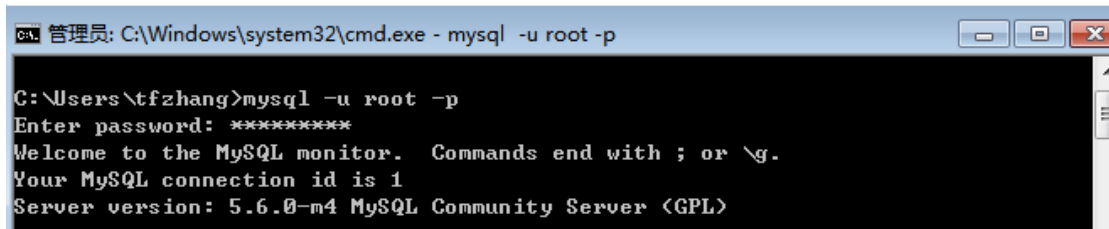
U 更新(Update)

D 删除>Delete)

后续介绍 mysql 时，还会以上述例句为例说明。

MySQL 是常用的一种免费的数据库，本文使用的 mysql-5.6.0 版本，安装文件见附录；具体的安装过程，请读者见文献[1]。安装完后，可能在 configuration 遇到的问题，不用担心，可借助网络解决。

使用时，在命令行输入 mysql，启动客户端，界面如下：




```
管理员: C:\Windows\system32\cmd.exe - mysql -u root -p

C:\Users\tfzhang>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.0-m4 MySQL Community Server (GPL)
```

mysql 后面的参数-u，用户名；-p，提示密码输入。

输入正确的密码后，则正确登陆。登陆后，一般会出于安全的考虑创建一个新的用户，因为 root 用户的权限太高。

创建新用户之前，先通过命令 show database 查看当前已有的数据库：

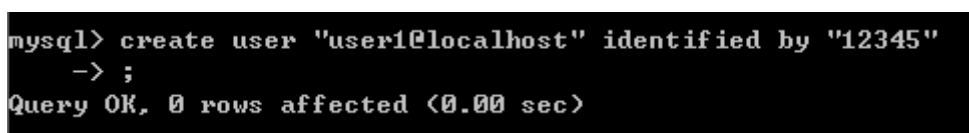


```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
4 rows in set (0.00 sec)
```

可以发现已经存在 test 数据库。

小目标 1：创建新用户 user1，只对 test 数据库享有所有权限。

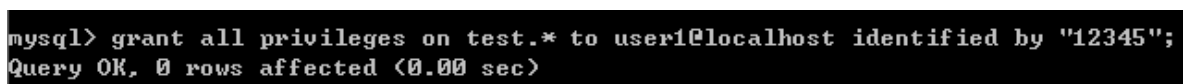
创建用户 user1 并且指定密码为 12345 的命令：



```
mysql> create user "user1@localhost" identified by "12345"
-> ;
Query OK, 0 rows affected (0.00 sec)
```

注意每条 sql 语句须以分号结尾，如果以回车结尾，系统默认为命令还没结束，会以符号"->"换行。

赋予 user1 对 test 数据库的所有权限的命令：



```
mysql> grant all privileges on test.* to user1@localhost identified by "12345";
Query OK, 0 rows affected (0.00 sec)
```

退出当前的 root 用户，再以 user1 登陆，运行 show databases 后，果然可以看到其中有 test；

小目标 2: 删除 test 数据库后再重新 create, 创建 student 表, 插入 zhang3 和 li4, 查询 zhang3 的信息, 再删除 li4。

查看数据库 test 中的内容可以使用命令:

show tables;

删除 test 数据库和再建 test 的命令如下:

```
mysql> drop database test;
Query OK, 0 rows affected (1.18 sec)

mysql> show databases;
+-----+
! Database      !
+-----+
! information_schema !
+-----+
1 row in set (0.00 sec)

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

test 数据库中创建 student 表的命令如下:

```
mysql> create table student(
-> id varchar(10) not null,
-> name varchar(16) not null,
-> phone varchar(16) not null);
Query OK, 0 rows affected (0.34 sec)
```

向数据表中插入 zhang3 和 li4:

```
mysql> insert into student(id, name, phone) values('1', 'zhang3', '1234567');
Query OK, 1 row affected (0.09 sec)

mysql> insert into student(id, name, phone) values(
-> '2', 'li4', '2345');
Query OK, 1 row affected (0.04 sec)
```

现在要查询 zhang3 的信息:

```
mysql> select * from student where name="zhang3";
+----+-----+-----+
! id ! name   ! phone   !
+----+-----+-----+
! 1  ! zhang3 ! 1234567 !
+----+-----+-----+
1 row in set (0.00 sec)
```

删除 zhang3 的信息:

```
mysql> select * from student;
+----+-----+-----+
| id | name  | phone |
+----+-----+-----+
| 1  | zhang3 | 1234567 |
| 2  | li4   | 2345   |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql> delete from student where name="zhang3";
Query OK, 1 row affected (0.11 sec)

mysql> select * from student;
+----+-----+-----+
| id | name  | phone |
+----+-----+-----+
| 2  | li4   | 2345   |
+----+-----+-----+
1 row in set (0.00 sec)
```

以上在 mysql 客户端展现如何操作数据库，如何直接在 Java 应用程序中操作数据库呢？此时就需要利用 JDBC。JDBC 提供一系列接口，使得直接在 java 应用程序中，就能调用执行 sql 语句。

#### 任务 1:

使用 jdbc 连接 mysql 数据库，并且以 user1 用户名登录到 mysql;  
访问 test 数据库，并且读出其中所有的数据，打印到终端;  
最后关闭数据库连接。

使用 JDBC 的主要步骤如下：

##### 1. 加载 JDBC 驱动;

Java 语句：Class.forName("com.mysql.jdbc.Driver");

##### 2. 建立与数据库的连接;

Java 语句：Connection con = DriverManager.getConnection(url, userName, password);

其中，url 指定要连接的数据库名，userName 表示数据库用户名，password 与当前用户相对应的密码;

##### 3. 使用 SQL 语句进行数据库操作;

对数据库的操作主要分为数据更新和数据查询;

##### 4. 释放资源;

数据库操作完毕后，关闭数据库连接以及释放相关资源;  
conn.close();

一般我们不会直接使用 root 用户，而是新建一个用户，并赋予该数据库操作数据库的权限，jdbc 后续使用该数据库登录操作。

具体代码见 case1.

任务 2: 创建一个 `Student` 类, 其中包含 `id`, `name` 和 `score` 三个属性; 创建数据库操作的 DAO 接口, 按照实现 DAO 接口的方式重写任务 1 中的类; 测试代码如下所述:

生成三个具体的 `Student` 实例, `zhang3`, `li4`, `wang5`:

1. 将这三个对象的数据写到 `test` 数据库中;
2. 将数据库中 `wang5` 的条目删除;
3. 将 `zhang3` 的分数更新为 90;
4. 找到数据库中 `id=1` 的条目, 并且返回一个对应的 `Student` 对象;

小目标 1: 通过 JDBC 为用户 `user1` 创建名为 `test` 的数据库, 并在其中创建 `student` 数据表, 再插入 `zhang3` 和 `li4` 两位学生的信息, 然后顺序查询两个学生的信息, 最后删除学生 `zhang3` 的条目。

具体使用的 `java.sql.*` 的有关语句, 可以先查询官方 API 文档中有关 `java.sql` 部分内容或者参考有关书籍。本文使用的 `jdbbc` 版本为 `jdbbc-5.1.22`, 安装文件见附录, 与 JUnit 的安装配置过程一样。

请读者先自行完成小目标 1, 再对照 `database` 文件中的 `DataTest.java` 示例代码。下面对示例代码中部分内容予以说明:

```
public static final String user = "user1";
public static final String userpass = "12345";
```

此处的 `user1` 及对应的 `password` 是以 `root` 用户登录 `mysql` 后创建的。

`PreparedStatement` 的下列用法, 有助于批量化的插入数据, 只要将 `set` 有关语句放在循环中执行。

```
String cmdStr = "insert into student (id, name, phone) values (?, ?, ?)";
try{
    cmd = con.prepareStatement(cmdStr);
    cmd.setString(1, "1");
    cmd.setString(2, "zhang3");
    cmd.setString(3, "1234");
    cmd.executeUpdate();
}....
```

下面的语句表示访问本地数据库, 如果要访问远程, 就要填写相应的 `url` 地址。

```
public static final String dburl = "jdbc:mysql://localhost:3306";
```

数据的事务操作:

一组数据库操作包含在一个事务中, 即意味着这组事务要么都运行成功, 要么都不运行成功; 不可能存在部分运行成功, 而其他运行不成功的情况。

典型的应用场景:

比如 A 的银行账户转 200 元到 B 账户，就设计到两个数据库更新操作：

数据库操作 1: A 的账户-200;

数据库操作 2: B 的账户+200;

这两个数据库操作就应该囊括在一个事务中，保证两个操作要么都执行，要么保证都不能顺利得到执行。否则，现实代码执行过程中，可能会出现某种意外，一个操作的代码正常执行，而另一个操作的代码无法正常执行，即出现单方账户变动的情况；事务这一机制可以保证在发生上述情况下，回滚已经执行的单方操作，维持资金总额的平衡。

具体的代码举例如下：

```
String sql1 = "update customer set account = account -200 where id = A";  
s.execute(sql1);
```

```
//不小心写错写成了 updata(而非 update)
```

```
String sql2 = "updata customer set account = account +200 where id = B";  
s.execute(sql2);
```

比如数据操作 2 中的 update 写成 updata，导致不能顺利执行，就会出现 A 被扣钱而 B 没有加钱的资金总额不平衡情况。所以，要将上述的两个数据库操作，放在一个事务中，即放在语句 c.setAutoCommit(false)（c 是 connection 引用）和 c.commit()之间：

```
c.setAutoCommit(false);  
String sql1 = "update customer set account = account -200 where id = A";  
s.execute(sql1);  
  
//不小心写错写成了 updata(而非 update)  
String sql2 = "updata customer set account = account +200 where id = B";  
s.execute(sql2);  
c.commit();
```

一旦数据操作 2 因为上述原因不能顺利执行，由于两个操作在事务中，所以操作步骤 1 虽然成功了，但还是被回滚，即撤销操作，保证资金总额的平衡。

要注意的一点：在 Mysql 中，只有当表的类型是 INNODB 的时候，才支持事务，所以需要把表的类型设置为 INNODB,否则无法观察到事务。

什么是 ORM？

Object Relationship Database Mapping，对象和关系数据库的映射关系，简单而言，一个对象对应数据库里的一条记录。

比如我们存在数据库里的学生的条目：(id, name, score);

同时 java 中定义一个类：Student;

```
public class Student{  
    public int id;  
    public String name;  
    public int score;  
}
```

根据上述的 ORM 设计，往数据库中增加一个学生的记录：

```
public static void add(Student st1);
```

什么是 DAO？

数据库访问对象，即基于 ORM 将数据库的相关操作封装在一个类里，其他地方看不到 JDBC 操作。

DAO 接口：

```
public interface DAO{
    public void add(Student st);
    public void update(Student st);
    public void delete(int id);
    public Student get(int id);
}
```

```
public class StudentDAO implements DAO{
```

```
    public StudentDAO() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

```
    public Connection getConnection() throws SQLException {
        return DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8", "root",
            "admin");
    }
```

```
    public void add(Student st){
        ...
    }
```

```
    public void add(Student st) {
```

```
        String sql = "insert into student values(null,?,?,?)";
```

```
        try (Connection c = getConnection(); PreparedStatement ps = c.prepareStatement(sql);) {
            ps.setString(1, st.name);
            ps.setFloat(2, st.hp);
            ps.setInt(3, st.damage);
            ps.execute();
        }
```

```

        ResultSet rs = ps.getGeneratedKeys();
        if (rs.next()) {
            int id = rs.getInt(1);
            hero.id = id;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public void update(Student st){
    ...
}

public void delete(int id){
    ...
}

public Student get(int id){
    ...
}
}

```

数据库连接池：

与多线程技术相结合；

初始化创建多个 connection 对象；供使用；

JDBC 的使用基本参照手册即可，多是约定的操作，不是太难，所以，本章也没有太多内容。剩下的主要依靠读者的实践，尤其是针对特定业务的数据库设计，这才是难点所在。

参考文献：

1. Java 编程手记-从实践中学习 Java。欧二强等编著。 清华大学出版社。

作业 1：

创建一个名称为 bank 的数据库，再创建一个 customer 的 table，table 中的条目是：

id	amount
A	500
B	700

存在两个用户 A 和 B，他们账户的金额分别为 500 元和 700 元，现在用户 A 要转账给用户 B200 元，请故意在代码中引入错误，从而比较没有事务处理和有事处理的不同，书写代码并且对输出进行截图。



作业 2:

原有 DAO 接口的基础上，再增加两个方法:

`public List<Student> list();` //返回表中所有的条目信息，并且存放在 `ArrayList` 数据结构中返回;

`public List<Student> list(int start, int count);` //返回表中序号从 `start` 开始的 `count` 个条目的信息，并且存放在 `ArrayList` 数据结构中返回;

测试代码:

创建 20 个 `Student` 对象，并且都 `add` 到 `test` 数据库中，然后采用 `list(10, 5)`，查询数据库第 10 到第 14 个数据条目。再使用 `list()` 方法查询所有的数据条目。

请在课堂代码的基础上完成上述的修改，并且截图。

**作业提交注意:**

提交可运行的源代码文件和必要的 `word` 说明文档；压缩打包。

其他注意事项以以往作业相同；