JDBC

1、JDBC 是什么？

Java Database Connection:java 数据库连接技术。

2、JDBC 作用？

使用 JDBC 实现让 Java 应用程序可以操作数据库系统。

3、连接方式：

四大连接方案：

1 桥连： JDBC-ODBC 连接。

2 直连： 直接通过 JDBC 驱动连接数据库。

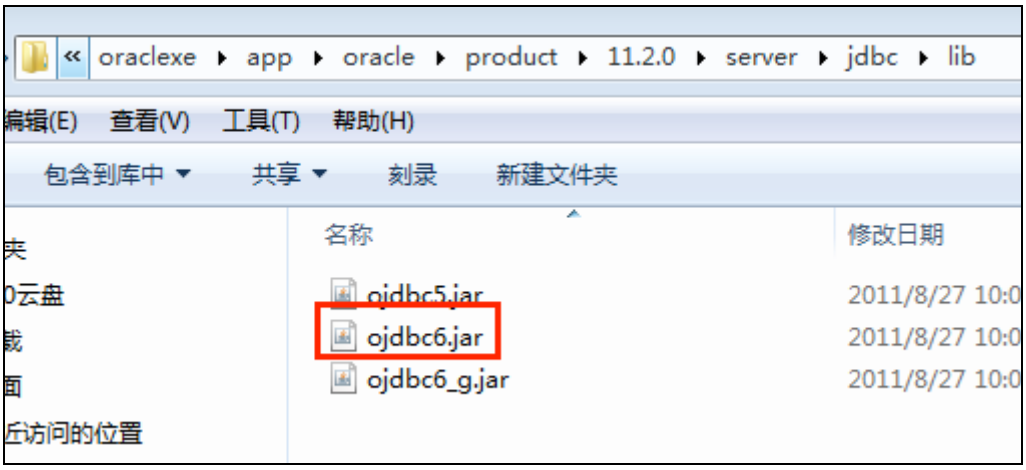
4、JDBC 本身是 SUN 公司提出一个标准

在 SUN，在 Java 程序中只有接口没有实现类。--提供接口

具体实现类是由每一个数据库厂商提供。--JDBC 驱动

针对每一种不同的数据库都有自己的驱动。

5、支持 Oracle 数据库的驱动



6、JDBC 中的核心接口与类

6.1、包：java.sql.* javax.sql.*

6.2、三个核心接口

6.3.1、Connection 接口：连接数据库接口。负责与数据库之间的连接。

6.3.2、Statement 接口：操作数据库接口，常用的子接口 PreparedStatement 接口，预处理 SQL 语句。负责执行 CRUD 命令。

6.3.3、ResultSet 接口：结果集接口。负责接收查询的结果。

6.3、驱动管理器类：DriverManager 类。负责加载数据库连接的驱动。

7、JDBC 操作的基本流程

7.1、加载 JDBC 的驱动-连接哪个数据库产品就加载哪个数据库的 JDBC 驱动

7.2、创建与数据库之间的连接

7.3、执行 CRUD 命令

7.4、如果是查询就处理结果集

7.5、关闭连接，释放资源

对数据库进行查询操作

```
package com.no4.test;
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;
public class Test {
    public static void main(String[] args) throws Exception {
        //连接数据库的四大参数项
        String username = "scott";
        String password = "tiger";
        String driver_class="oracle.jdbc.driver.OracleDriver";//驱动类名称
        String url = "jdbc:oracle:thin:@localhost:1521:XE";//连接字符串
//      7.1、加载 JDBC 的驱动-连接哪个数据库产品就加载哪个数据库的 JDBC 驱动
        Class.forName(driver_class);
//      7.2、创建与数据库之间的连接
        Connection conn = DriverManager.getConnection(url, username    , password);
        System.out.println(conn);
//      7.3、执行 CRUD 命令
        String sql = "select * from emp";
        Statement stat = conn.createStatement();
//      7.4、如果是查询就处理结果集
        ResultSet rs = stat.executeQuery(sql);
        //rs 是结果集对象，结果集就是一个二维表格。
        while(rs.next()){//下一行。如果返回 true 说明有记录。
            int empno = rs.getInt(1);
            String ename = rs.getString("ename");
            Date date = rs.getDate("hiredate");
            System.out.println(empno+":"+ename+":"+date);
        }
//      7.5、关闭连接，释放资源
        rs.close();
        stat.close();
        conn.close();
    }
}
```

对数据库进行增删改操作

```java
package com.no4.test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;
```

```
public class UpdateTest {

    public static void main(String[] args) throws Exception {
        //连接数据库的四大参数项
        String username = "scott";
        String password = "tiger";
        String driver_class="oracle.jdbc.driver.OracleDriver";//驱动类名称
        String url = "jdbc:oracle:thin:@localhost:1521:XE";//连接字符串
//      7.1、加载 JDBC 的驱动-连接哪个数据库产品就加载哪个数据库的 JDBC 驱动
        Class.forName(driver_class);
//      7.2、创建与数据库之间的连接
        Connection conn = DriverManager.getConnection(url, username    , password);
        System.out.println(conn);
//      7.3、执行 CRUD 命令
        String sql = "insert into dept(deptno,dname,loc) values(50 , '总部' ,'北京远洋
')";
        Statement stat = conn.createStatement();
        int count = stat.executeUpdate(sql);//增删改都调用这个方法
        if(count > 0) {
            System.out.println("操作成功！ ");
        }
//      7.5、关闭连接，释放资源
        stat.close();
        conn.close();
    }
}
```

8、Statement 接口：

8.0、获得 Statement 对象

```
Statement stat = conn.createStatement();
```

8.1、执行查询的方法：

```
ResultSet rs = stat.executeQuery(SQL);
```

8.2、执行增删改的方法：

```
int i = stat.executeUpdate(SQL);
```

9、ResultSet 接口：

结果集对象下移一条时调用方法 next();

9.1、两个值：BOF ，EOF

9.1.1、BOF：第一行的前面。rs 对象一开始指向的位置就是 BOF。

9.1.2、EOF：最后一行的后面。rs 对象的 next 方法是下移一行，如果移动之后指向了 EOF 返回 false，表示没有记录

9.2、从结果集对象中取值。

9.2.1、getXxxx(int):其中 Xxxx 表示的数据类型。int 表示的列的编号，从 1 开始

9.2.2、getXxxx(String)：String 表示列的名称。推荐使用

10、PreparedStatement 接口：Statement 子接口，预处理 SQL 语句。

10.1、创建 PreparedStatement 接口的实例

```
String sql = "insert into dept(deptno,dname,loc) values(?,?,?)";//可以使用?占位符
PreparedStatement pstat = conn.prepareStatement(sql);
```

其中：参数 sql 就是一个要预编译的 sql 命令

10.2、为占位符赋值

```
//为占位符赋值
pstat.setInt(1, deptno);
pstat.setString(2, dname);
pstat.setString(3, loc);
```

10.3、执行

```
pstat.executeUpdate();
```

```
package com.no4.test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;

public class PreparedStatementTest {

    public static void main(String[] args) throws Exception {
        // 连接数据库的四大参数项
        String username = "scott";
        String password = "tiger";
        String driver_class = "oracle.jdbc.driver.OracleDriver";// 驱动类名称
        String url = "jdbc:oracle:thin:@localhost:1521:XE";// 连接字符串
        // 7.1、加载 JDBC 的驱动-连接哪个数据库产品就加载哪个数据库的 JDBC 驱动
        Class.forName(driver_class);
        // 7.2、创建与数据库之间的连接
        Connection conn = DriverManager.getConnection(url, username, password);
        System.out.println(conn);
        // 7.3、执行 CRUD 命令
        int deptno = 80;
        String dname = "审核部";
        String loc = "上海";
        String sql = "insert into dept(deptno,dname,loc) values(?,?,?)";//可以使用?占
位符
        PreparedStatement pstat = conn.prepareStatement(sql);
        //为占位符赋值
        pstat.setInt(1, deptno);
```

```
            pstat.setString(2, dname);
            pstat.setString(3, loc);
            //执行
            int i = pstat.executeUpdate();
            if(i >0){
                System.out.println("操作成功！");
            }
            // 7.5、关闭连接，释放资源
            conn.close();
        }
}
```

11、针对 Connection 进行独立的封装

针对所有数据库操作，第一步都是区得数据库连接。
获得数据库连接也是最消耗资源的动作。

```
package com.no4.utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBUtils {
    private static Connection connection;

    public static Connection getConnection() throws ClassNotFoundException,
            SQLException {
        if (DBUtils.connection == null || DBUtils.connection.isClosed()) {
            // 连接数据库的四大参数项
            String username = "scott";
            String password = "tiger";
            String driver_class = "oracle.jdbc.driver.OracleDriver";// 驱动类名称
            String url = "jdbc:oracle:thin:@localhost:1521:XE";// 连接字符串
            Class.forName(driver_class);
            DBUtils.connection = DriverManager.getConnection(url, username,
                    password);
        }
        return DBUtils.connection;
    }

    public static void closeConnection() {
        try {
```

```
            if (DBUtils.connection != null && !DBUtils.connection.isClosed()) {
                DBUtils.connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

12、创建实体类来映射数据表

```
public class Dept {
    private int deptno;
    private String dname;
    private String loc;

    public Dept() {

    public Dept(int deptno, String dname, String loc) {

    public int getDeptno() {

    public void setDeptno(int deptno) {

    public String getDname() {

    public void setDname(String dname) {

    public String getLoc() {

    public void setLoc(String loc) {


}
```

13、针对 CRUD 操作的封装

DAO：封装针对数据库的 CRUD 的操作

DeptDAO.java：

```java
public class DeptDAO {

    public void save(Dept dept){

    public void update(Dept dept){

    public void delete(int deptno){

    public Dept findById(int deptno){

    public List<Dept> findAll(){
}
```

```java
    public void save(Dept dept) throws ClassNotFoundException, SQLException {
//      String sql = "insert into dept(deptno,dname,loc) values(S_DEPT_ID.nextval,?,?)";
        String sql = "insert into dept(deptno,dname,loc) values(?,?,?)";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setInt(1, dept.getDeptno());//手动设置主键
        pstat.setString(2, dept.getDname());
        pstat.setString(3, dept.getLoc());
        pstat.executeUpdate();
    }
```

```java
    public Dept findById(int deptno) throws ClassNotFoundException,
            SQLException {
        String sql = "select * from dept where deptno = ?";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setInt(1, deptno);
        ResultSet rs = pstat.executeQuery();
        if (rs.next()) {
            Dept dept = new Dept();
            dept.setDeptno(rs.getInt("deptno"));
            dept.setDname(rs.getString("dname"));
            dept.setLoc(rs.getString("loc"));
            return dept;
        } else {
            return null;
        }
    }
```

```java
    public List<Dept> findAll() throws ClassNotFoundException, SQLException {
        List<Dept> deptList = new ArrayList<Dept>();
        String sql = "select * from dept";
        Connection conn = DBUtils.getConnection();
```

```
        PreparedStatement pstat = conn.prepareStatement(sql);
        ResultSet rs = pstat.executeQuery();
        while (rs.next()) {
            Dept dept = new Dept();
            dept.setDeptno(rs.getInt("deptno"));
            dept.setDname(rs.getString("dname"));
            dept.setLoc(rs.getString("loc"));
            deptList.add(dept);
        }
        return deptList;
    }
```

14、连接现在应该在业务层关闭

BIZ：业务层。处理异常，关闭数据库连接

DeptBIZ.java:

```
    public boolean save(Dept dept){
        DeptDAO deptDAO = new DeptDAO();
        try {
            deptDAO.save(dept);
            return true;
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return false;
        } finally{
            DBUtils.closeConnection();//关闭连接
        }
    }
```

```
    public Dept findById(int deptno) {
        DeptDAO deptDAO = new DeptDAO();
        try {
            return deptDAO.findById(deptno);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBUtils.closeConnection();
        }
    }
```

```
    public List<Dept> findAll(){
        DeptDAO deptDAO = new DeptDAO();
```

```
        try {
            return deptDAO.findAll();
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBUtils.closeConnection();
        }
    }
```

测试类：SaveTest.java

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        System.out.println("输入部门编号：");
        int deptno = scanner.nextInt();
        scanner.nextLine();
        System.out.println("输入部门名称：");
        String dname = scanner.nextLine();
        System.out.println("输入部门地址：");
        String loc = scanner.nextLine();

        Dept dept = new Dept(deptno,dname,loc);

        DeptBIZ deptBIZ = new DeptBIZ();
        if(deptBIZ.save(dept)){
            System.out.println("保存成功！");
        }else{
            System.out.println("保存失败！");
        }
    }
```

测试类：FindByIdTest.java

```
public class FindByIdTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        System.out.println("输入部门编号：");
        int deptno = scanner.nextInt();
        scanner.nextLine();
```

```
        DeptBIZ deptBIZ = new DeptBIZ();

        Dept dept = deptBIZ.findById(deptno);

        if(dept==null){
            System.out.println("部门编号有错误，没有找到相应部门信息！");
        }else{
            System.out.println(dept.getDeptno());
            System.out.println(dept.getDname());
            System.out.println(dept.getLoc());
        }
    }

}
```

测试类：FindAllTest.java

```
package com.no4.test;
import java.util.List;
import com.no4.biz.DeptBIZ;
import com.no4.domain.Dept;
public class FindAllTest {
    public static void main(String[] args) {
        DeptBIZ deptBIZ = new DeptBIZ();
        List<Dept> deptList  = deptBIZ.findAll();
        if(deptList == null ){
            System.out.println("查询出错！");
        }else if(deptList.isEmpty()){
            System.out.println("查询无结果！");
        }else{
            for (Dept dept : deptList) {
                System.out.print(dept.getDeptno()+"|");
                System.out.print(dept.getDname()+"|");
                System.out.println(dept.getLoc());
                System.out.println("----------");
            }
        }
    }
}
```

15、有外键关系表的操作-emp 表

15.1、实体类

```
public class Emp {
    private BigDecimal empno;
    private String ename;
    private String job;
    private BigDecimal mgr;
    private Timestamp hiredate;
    private BigDecimal sal;
    private BigDecimal comm;
    private Dept dept;
    public Emp() {
        // TODO Auto-generated constructor
    }
```

int 与 Integer 之间的区别 ： Integer 比 int 多一个值 ：null

15.2、DAO 封装

15.2.1、findById 方法：

```
public Emp findById(int empno) throws ClassNotFoundException, SQLException {
    String sql = "select * from emp inner join dept on emp.deptno=dept.deptno where empno=?";
    Connection conn = DBUtils.getConnection();
    PreparedStatement pstat = conn.prepareStatement(sql);
    pstat.setInt(1, empno);
    ResultSet rs = pstat.executeQuery();
    if (rs.next()) {
        Emp emp = new Emp();
        emp.setEmpno(rs.getBigDecimal("empno"));
        emp.setEname(rs.getString("ename"));
        emp.setHiredate(rs.getTimestamp("hiredate"));
        emp.setSal(rs.getBigDecimal("sal"));
        emp.setComm(rs.getBigDecimal("comm"));
        emp.setJob(rs.getString("job"));
        emp.setMgr(rs.getBigDecimal("mgr"));
        Dept dept = new Dept();
        dept.setDeptno(rs.getInt("deptno"));
        dept.setDname(rs.getString("dname"));
        dept.setLoc(rs.getString("loc"));
        emp.setDept(dept);
        return emp;
    } else {
        return null;
    }
```

15.3、测试类：

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    EmpDAO empDAO = new EmpDAO();
    try {
        Emp emp = empDAO.findById(7839);
        System.out.println(emp);
        System.out.println("工号: "+emp.getEmpno());
        System.out.println("姓名: "+emp.getEname());
        System.out.println(emp.getComm());
        System.out.println(emp.getJob());
        System.out.println("上级领导编号："+emp.getMgr());
        System.out.println(emp.getSal());
        System.out.println(emp.getHiredate());
        System.out.println(emp.getDept().getDeptno());
        System.out.println(emp.getDept().getDname());
        System.out.println(emp.getDept().getLoc());
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtils.closeConnection();
    }

}
```

15.4、针对 Oracle 数据库类型的处理方案

1、所有数值类型使用 BigDecimal

2、varchar 使用 String

3、日期使用 Timestamp

16、数据库连接池技术

16.1、什么是连接池？

就是一个存放数据库连接的池子。
能帮我创建，维护，管理数据库连接对象。
可以通过缓存的方式将一个数据库的连接可以给多个与数据库的操作使用。
数据库开发时最消耗就是创建连接。
连接池就是一个出租车。10 万。

在连接池维护多个数据库之间的连接对象。当你需要时就给你一个数据库的连接，当你用完时就释放还给连接池。

1 天 1000 个数据库连接。老的方式：1000 次。连接池：10 个（同一时间最多只有 10 个用户）。

使用连接池可以有效的减少创建数据库的连接的次数。

16.2、数据源对象 javax.sql.DataSource

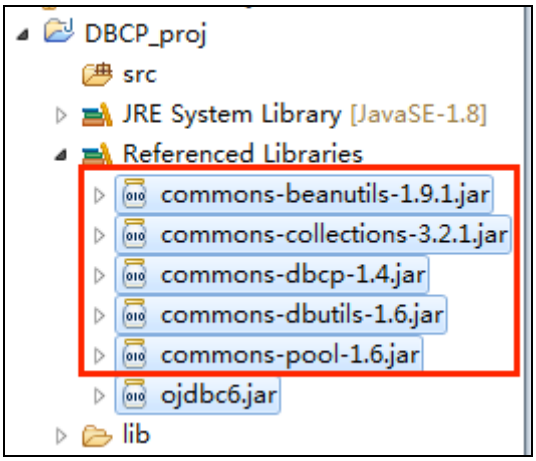
在 Java 程序中是使用数据源的对象实现的连接池的技术。

DataSource 对象中创建和维护一定数量的 Connection 对象。

16.3、世面上主流有:DBCP ，C3P0

16.4、使用 DBCP 创建数据库连接对象

16.4.1、加载驱动。加载 DBCP 的驱动



16.4.2、设置 DBCP 连接池参数

创建一个资源文件.properteis。

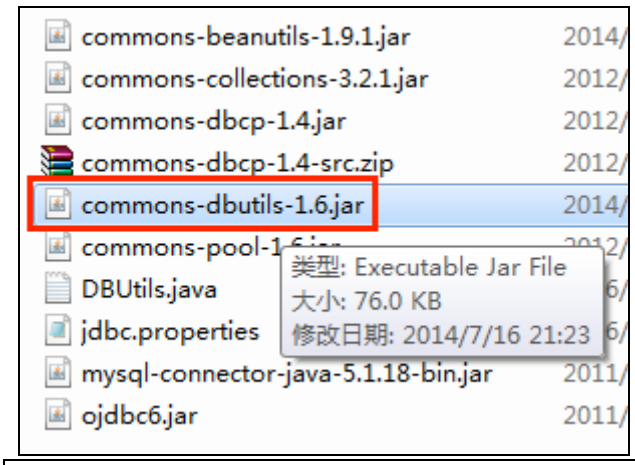
```
driverClassName=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:XE
username=scott
password=tiger

maxActive=50--最大活动数量

maxIdle=20  --最大空闲数量

maxWait=60000 --最大等待时间，毫秒。
```

16.4.3、修改 DBUtils.java 类。

```
package com.no4.utils;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.Properties;
```

```java
import javax.sql.DataSource;
import org.apache.commons.dbcp.BasicDataSourceFactory;

public class DBUtils {
    private static Connection conn;
    private static DataSource dataSource;
    static {
        Properties properties = new Properties();
        try {
            properties.load(DBUtils.class
                    .getResourceAsStream("jdbc.properties"));
            DBUtils.dataSource = BasicDataSourceFactory
                    .createDataSource(properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static DataSource getDataSource() {
        return dataSource;
    }

    public static Connection getConnection() throws SQLException {
        if (conn == null || conn.isClosed()) {
            conn = dataSource.getConnection();
        }
        return conn;
    }

    public static void closeConnection() {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

17、使用 DBUtils 工具进行 CRUD 操作




```
package com.no4.dao;

import java.sql.SQLException;
import java.util.List;

import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanHandler;
import org.apache.commons.dbutils.handlers.BeanListHandler;

import com.no4.domain.Dept;
import com.no4.utils.DBUtils;

public class DeptDAO {
    public List<Dept> findAll() throws SQLException {
        String sql = "select * from dept";
        QueryRunner queryRunner = new QueryRunner(DBUtils.getDataSource());
        return queryRunner.query(sql, new BeanListHandler<Dept>(Dept.class));
    }

    public Dept findById(int deptno) throws SQLException{
        String sql = "select * from dept where deptno = ? ";
        QueryRunner queryRunner = new QueryRunner(DBUtils.getDataSource());
        return queryRunner.query(sql, new BeanHandler<Dept>(Dept.class), deptno);
    }

    public int save(Dept dept) throws SQLException{
        String sql = "insert into dept(deptno,dname,loc) values(?,?,?)";
        QueryRunner queryRunner = new QueryRunner(DBUtils.getDataSource());
        int i = queryRunner.update(sql,
dept.getDeptno(),dept.getDname(),dept.getLoc());
```

```
        return i;
    }
}
```

18、Java 程序中使用事务

18.1、数据表


```
public class Card {
    private BigDecimal cardId;
    private String cardName;
    private BigDecimal cardMoney;
    public Card() {
    public Card(BigDecimal cardId, Strin
```

18.2、JDBC 操作-DAO

```
package com.no4.dao;

import java.sql.SQLException;

import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanHandler;

import com.no4.domain.Card;
import com.no4.utils.DBUtils;

public class CardDAO {
    public void update(Card card) throws Exception{
        String sql = "update newcard set card_money = ? where card_id = ?";
```

```
        QueryRunner queryRunner = new QueryRunner(DBUtils.getDataSource());
        queryRunner.update(sql, card.getCardMoney() , card.getCardId());
    }

    public Card findById(int cardId) throws SQLException{
        String sql = "select * from newcard where card_id = ? ";
        QueryRunner queryRunner = new QueryRunner(DBUtils.getDataSource());
        return queryRunner.query(sql, new BeanHandler<Card>(Card.class), cardId);
    }
}
```

18.3、业务层-BIZ

事务处理在 BIZ 层、

1、开启事务:Connection.setAutoCommit(false);

2、提交事务：Connection.commit();

3、回滚事务：Connection.rollback();

```
    /**
     * 存钱方法
     *
     * @param cardId
     *            卡号
     * @param money
     *            金额
     * @return true 操作成功
     */
    public boolean save(int cardId, double money) {
        CardDAO cardDAO = new CardDAO();
        try {
            Card card = cardDAO.findById(cardId);
            double temp = card.getCardMoney().doubleValue() + money;
            card.setCardMoney(new BigDecimal(temp));
            // 手动开启一个事务
            DBUtils.getConnection().setAutoCommit(false);// 设置自动提交事务为关闭
            cardDAO.update(card);
            DBUtils.getConnection().commit();// 手动提交事务
```

```
            System.out.println("操作成功！");
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            try {
                DBUtils.getConnection().rollback();// 手动回滚事务
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            System.out.println("操作失败！");
            return false;
        } finally {
            DBUtils.closeConnection();
        }

    }
```

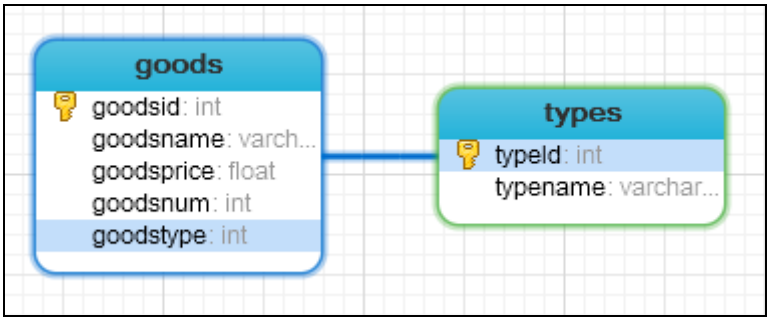
```
    /**
     * 转账业务方法
     *
     * @param loadCardId
     *            取款的账号
     * @param saveCardId
     *            存款的账号
     * @param money
     *            金额
     * @return
     */
    public boolean saveload(int loadCardId, int saveCardId, double money) {
        CardDAO cardDAO = new CardDAO();
        try {
            Card loadCard = cardDAO.findById(loadCardId);
            Card saveCard = cardDAO.findById(saveCardId);
            // 从 loadcard 取钱
            double loadtemp = loadCard.getCardMoney().doubleValue() - money;
            loadCard.setCardMoney(new BigDecimal(loadtemp));
            // 从 savecard 存钱
            double savetemp = saveCard.getCardMoney().doubleValue() + money;
            saveCard.setCardMoney(new BigDecimal(savetemp));

            //转账.事务
            DBUtils.getConnection().setAutoCommit(false);
            cardDAO.update(saveCard);
            System.out.println("save ok!");
            cardDAO.update(loadCard);
```

```
            System.out.println("load ok!");
            DBUtils.getConnection().commit();
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            try {
                DBUtils.getConnection().rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            return false;
        } finally{
            DBUtils.closeConnection();
        }
    }
```

19、操作 MySQL 数据库

19.1、MySQL 数据库模型

数据库名：goodsdb



19.2、实体类

```
public class Types {
    private Integer typeid;
    private String typename;
    public Types() {
```

```java
public class Goods {
    private Integer goodsid;
    private String goodsname;
    private Float goodsprice;
    private Integer goodsnum;
    private Integer goodstype;// 为了给QueryRunner

    private Types types;// 映射关系

    public Goods() {
        // TODO Auto-generated constructor stub
    }
```

19.3、DAO 层

```java
    public void save(Types types) throws SQLException{
        String sql = "insert into types(typename) values(?)";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, types.getTypename());
        pstat.executeUpdate();
    }
```

19.4、BIZ 层-与数据库产品无关。与之前内容一致。

20、DBUtils 与多线程

Connection 对象，连接对象，在有事务情况，在一个事务中保证是同一个连接对象。

所以我们不能每次调用 DBUtils 工具类的 getConnection 方法都获得一个新的连接对象。

处理方案： DBUtils 类中创建一个 static 的 connection。在 getConnection 方法中每次在获得连接时都判断，判断连接对象是否存在，如果存在就直接返回连接对象，如果不存在就创建一个连接。

在多线程环境下，使用 Static 的 Connection 对象，是不是只有一份。多个线程同时使用一个 Connection 对象。

操作时有一步骤是必须执行。关闭 Connection 对象。

保证在一个线程中的同一个事务操作是同一个连接对象。在每个不同线程对象中使用不同的连接对象。

20.1、ThreadLocal 类

本地线程对象。每个线程有一个。这个对象就是容器，只能存放一个内容。

ThreadLocal<Connection> threadLocal = new ThreadLocal<Connection>();

创建了一个存放 Connection 对象的本地线程对象。

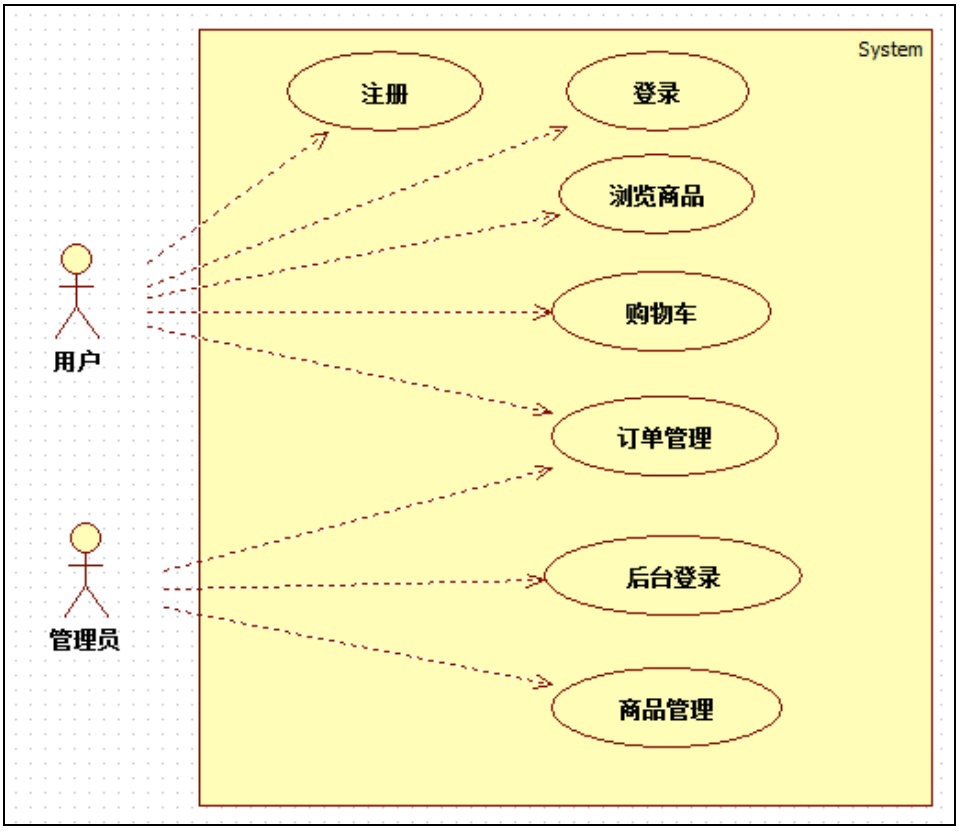
20.2、ThreadLocal 类的 set 和 get 方法

20.3、修改后的 DBUtils.java 类

```
package com.no4.utils;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.Properties;
import javax.sql.DataSource;
import org.apache.commons.dbcp.BasicDataSourceFactory;

public class DBUtils {
    private static DataSource dataSource;
    private static ThreadLocal<Connection> threadLocal = new
ThreadLocal<Connection>();
    static {
        Properties properties = new Properties();
        try {
            properties.load(DBUtils.class
                    .getResourceAsStream("jdbc.properties"));
            DBUtils.dataSource = BasicDataSourceFactory
                    .createDataSource(properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static DataSource getDataSource() {
        return dataSource;
```

```
    }
    public static Connection getConnection() throws SQLException {
        Connection conn = threadLocal.get();
        if (conn == null || conn.isClosed()) {
            conn = dataSource.getConnection();
            threadLocal.set(conn);
        }
        return conn;
    }
    public static void closeConnection() {
        try {
            Connection conn = threadLocal.get();
            if (conn != null && !conn.isClosed()) {
                conn.close();//关闭连接，释放连接给数据源
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally{
            threadLocal.set(null);
        }
    }
}
```

21、综合练习



数据库：Oracle 数据库，新建账号：jack21，jack21

序列对象:MYSEQ

主键使用 UUID。

工程使用 JDK8

分层：domain,dao,biz,view

事务在 biz 层处理。

密码 MD5 加密

21.1、环境搭建和工具类的使用

21.1.1、环境搭建

21.1.2、UUID

```
package com.no4.utils;

import java.util.UUID;

public class UUIDUtils {

    public static String getUUID(){
        UUID uuid = UUID.randomUUID();
        return uuid.toString().replace("-", "");
    }
}
```

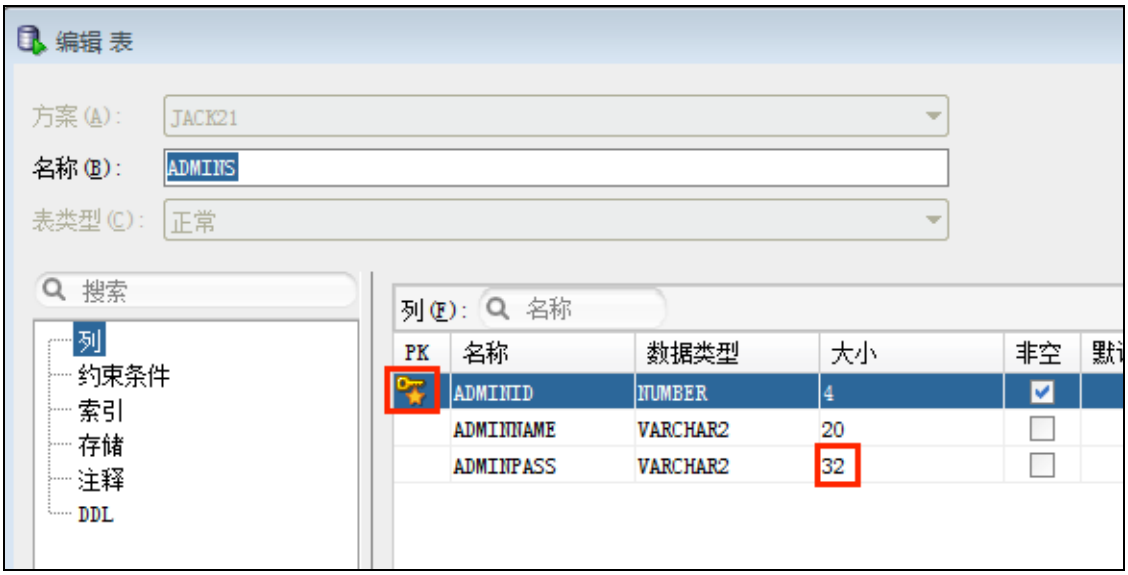
21.1.3、MD5

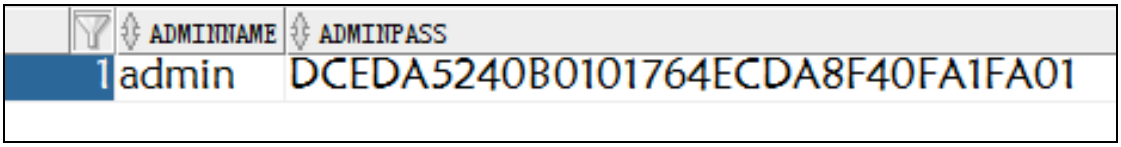
```
String pass = "123";
MD5Utils md5Utils = new MD5Utils();
String md5pass = md5Utils.getMD5ofStr(pass);
System.out.println(md5pass);
```

21.1.4、DBUtils

21.2、用例 1：后台管理登录功能

21.2.1、数据库模型





21.2.2、DOMAIN 实体域

```
public class Admins {
    private BigDecimal adminid;
    private String adminname;
    private String adminpass;
    public Admins() {
    public Admins(BigDecimal adminid, String adminname, String ad

    public Admins(String adminname, String adminpass) {

    public BigDecimal getAdminid() {
    public void setAdminid(BigDecimal adminid) {
    public String getAdminname() {
    public void setAdminname(String adminname) {
    public String getAdminpass() {
    public void setAdminpass(String adminpass) {

}
```

21.2.3、DAO 数据访问层

```
    public Admins findByName(String adminName) throws Exception{
        String sql = "select * from admins where adminname=?";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, adminName);
        ResultSet rs = pstat.executeQuery();
        if(rs.next()){
            Admins admins = new Admins();
            admins.setAdminid(rs.getBigDecimal("adminid"));
            admins.setAdminname(rs.getString("adminname"));
            admins.setAdminpass(rs.getString("adminpass"));
            return admins;
        }else{
            return null;
        }
    }
```

21.2.4、BIZ 业务层

```
public class AdminsBIZ {
    private AdminsDAO adminsDAO = new AdminsDAO();
```

```
    /**
     * 后台管理登录方法
     * @param adminname 管理员名称
     * @param adminpass 密码
     * @return 查询到管理员对象，返回 null 表示查询失败
     */
    public Admins isLogin(String adminname,String adminpass){
        try {
            Admins admins = adminsDAO.findByName(adminname);
            if(admins!=null){
                MD5Utils md5Utils = new MD5Utils();
                String temppass = md5Utils.getMD5ofStr(adminpass);
                if(admins.getAdminpass().equals(temppass)){
                    return admins;
                }else{
                    return null;
                }
            }else{
                return null;
            }
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } finally{
            DBUtils.closeConnection();
        }
    }
```

21.2.5、VIEW 视图层

```
    public static void adminLoginView(){
        System.out.println("请输入管理员账号：");
        String adminname = SCANNER.nextLine();
        System.out.println("请输入管理员密码：");
        String adminpass = SCANNER.nextLine();
        AdminsBIZ adminsBIZ = new AdminsBIZ();
        Admins admins = adminsBIZ.isLogin(adminname, adminpass);
        if(admins == null){
```

```
            System.out.println("登录失败！");
        }else{
            System.out.println("登录成功！进入后台管理界面！");
            RunStatusUtils.loginAdmins = admins;
        }
    }
```

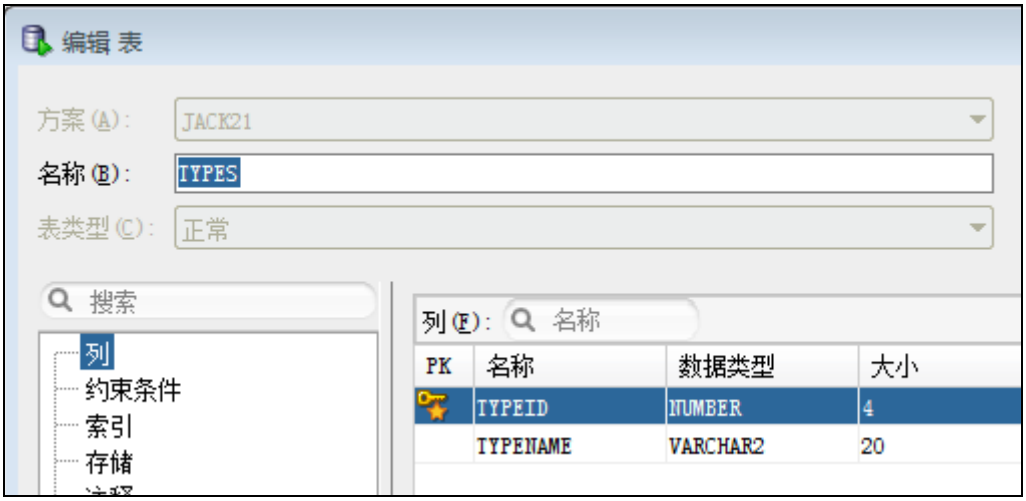
21.3、用例 2：保存商品功能

21.3.1、业务分析：

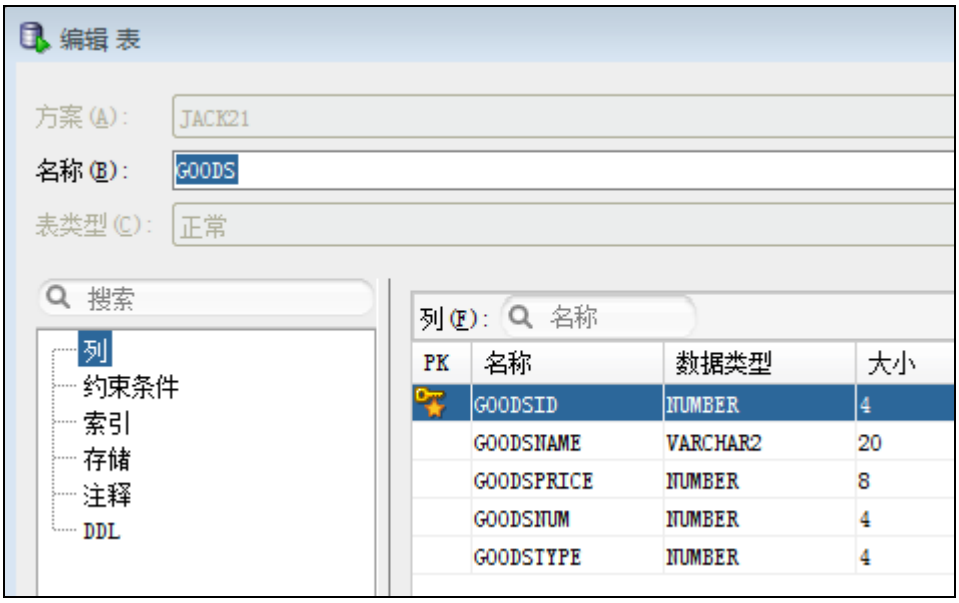
1、商品的类型应该是选择的方式

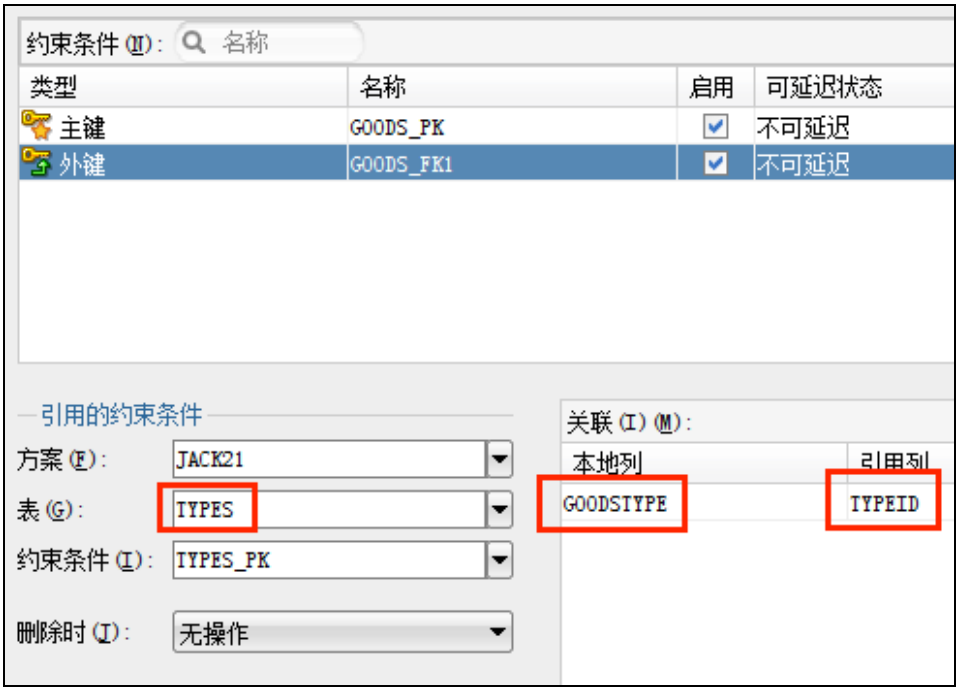
2、商品的名称不能重复

21.3.2、数据库模型



```
insert into types(typeid,typename) values(myseq.nextval,'数码产品');
commit;
insert into types(typeid,typename) values(myseq.nextval,'家用电器');
commit;
insert into types(typeid,typename) values(myseq.nextval,'服装服饰');
commit;
```





```
insert into goods(goodsid,goodsname,goodsprice,goodsnum,goodstype)
values(myseq.nextval,'SONY 单反照相机',10000,10,2);
commit;
```

21.3.3、DOMAIN 实体域

```
public class Types {
    private BigDecimal typeId;
    private String typeName;
    public Types() {
    public Types(BigDecimal typeId, String typeName) {
    public BigDecimal getTypeId() {
    public void setTypeId(BigDecimal typeId) {
    public String getTypeName() {
    public void setTypeName(String typeName) {
```

```
public class Goods {
    private BigDecimal goodsId;
    private String goodsName;
    private BigDecimal goodsPrice;
    private BigDecimal goodsNum;
    private Types goodsType;
    public Goods() {
```

21.3.4、DAO 数据访问层

1、TypesDAO.java：查询所有商品类型

```
    public List<Types> findAll() throws SQLException{
        List<Types> typesList = new ArrayList<Types>();
        String sql = "select * from types";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        ResultSet rs = pstat.executeQuery();
        while(rs.next()){
            Types types = new Types();
            types.setTypeId(rs.getBigDecimal("typeid"));
            types.setTypeName(rs.getString("typename"));
```

```
            typesList.add(types);
        }
        return typesList;
    }
```

2、GoodsDAO.java：按商品名称查询

```
    public Goods findByGoodsName(String goodsName) throws SQLException {
        String sql = "select * from goods left join types "
                + " on goods.goodstype=types.typeid " + " where goodsname = ?";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, goodsName);
        ResultSet rs = pstat.executeQuery();
        if (rs.next()) {
            Goods goods = new Goods();
            goods.setGoodsId(rs.getBigDecimal("goodsid"));
            goods.setGoodsName(rs.getString("goodsname"));
            goods.setGoodsPrice(rs.getBigDecimal("goodsprice"));
            goods.setGoodsNum(rs.getBigDecimal("goodsnum"));
            Types types = new Types();
            types.setTypeId(rs.getBigDecimal("typeid"));
            types.setTypeName(rs.getString("typename"));
            goods.setGoodsType(types);
            return goods;
        } else {
            return null;
        }
    }
```

3、GoodsDAO.java：保存商品

```
    public void save(Goods goods) throws SQLException {
        String sql = "insert into "
                + " goods(goodsid,goodsname,goodsprice,goodsnum,goodstype) "
                + " values(myseq.nextval,?,?,?,?)";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, goods.getGoodsName());
        pstat.setBigDecimal(2, goods.getGoodsPrice());
        pstat.setBigDecimal(3, goods.getGoodsNum());
        pstat.setBigDecimal(4, goods.getGoodsType().getTypeId());
        pstat.executeUpdate();
    }
```

21.3.5、BIZ 业务层

1、TypesBIZ.java：查询所有商品类型

```
public List<Types> findAll(){
    try {
        return typesDAO.findAll();
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    } finally{
        DBUtils.closeConnection();
    }
}
```

2、GoodsBIZ.java：判断商品名称是否存在

```
public Boolean existGoodsName(String goodsName) {
    try {
        Goods goods = goodsDAO.findByGoodsName(goodsName);
        if (goods == null) {
            return false;
        } else {
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

3、GoodsBIZ.java：保存商品，事务处理

```
public Boolean save(Goods goods) {
    try {
        if (existGoodsName(goods.getGoodsName())) {
            return false;
        } else {
            DBUtils.getConnection().setAutoCommit(false);
            goodsDAO.save(goods);
            DBUtils.getConnection().commit();
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
```

```
            try {
                DBUtils.getConnection().rollback();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            return null;
        } finally {
            DBUtils.closeConnection();
        }
    }
```

21.3.6、VIEW 视图层

1、显示所有商品类型借用户选择

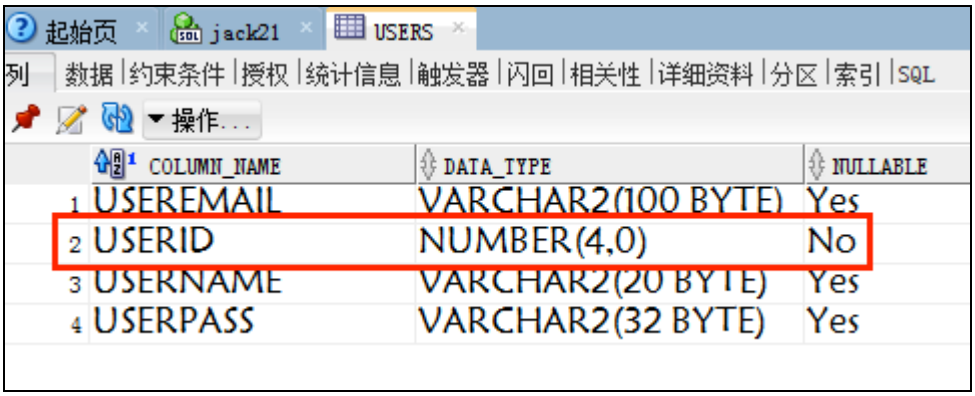
2、保存商品的视图

```
    public static void saveGoodsView(){
        System.out.println("商品名称：");
        String goodsName = SCANNER.nextLine();
        System.out.println("商品价格：");
        BigDecimal goodsPrice = SCANNER.nextBigDecimal();
        SCANNER.nextLine();
        System.out.println("商品数量：");
        BigDecimal goodsNum = SCANNER.nextBigDecimal();
        SCANNER.nextLine();
        System.out.println("商品类型的编号：");
        TypesBIZ typesBIZ = new TypesBIZ();
        List<Types> typesList = typesBIZ.findAll();
        for (Types types : typesList) {
            System.out.println(types.getTypeId()+":"+types.getTypeName());
        }
        BigDecimal typeId = SCANNER.nextBigDecimal();
        SCANNER.nextLine();

        Goods goods = new Goods();
        goods.setGoodsName(goodsName);
        goods.setGoodsPrice(goodsPrice);
        goods.setGoodsNum(goodsNum);
        Types types = new Types();
        types.setTypeId(typeId);
        goods.setGoodsType(types);
```

```
        GoodsBIZ goodsBIZ = new GoodsBIZ();
        if(goodsBIZ.save(goods)){
            System.out.println("保存成功！");
        }else{
            System.out.println("保存失败！");
        }
    }
```

21.4、用例 3：用户登录功能

21.4.1、数据库模型



21.4.2.DOMAIN

```
public class Users {
    private BigDecimal userId;
    private String userName;
    private String userPass;
    private String userEmail;
}
```

21.4.3、DAO

```
    public Users findByName(String userName) throws Exception{
        String sql = "select * from users where username=?";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, userName);
        ResultSet rs = pstat.executeQuery();
```

```
        if(rs.next()){
            Users users = new Users();
            users.setUserId(rs.getBigDecimal("userid"));
            users.setUserName(rs.getString("username"));
            users.setUserPass(rs.getString("userpass"));
            users.setUserEmail(rs.getString("useremail"));
            return users;
        }else{
            return null;
        }
    }
```

21.4.4、BIZ

```
    public Users isLogin(String username,String userpass){
        try {
            Users users = usersDAO.findByName(username);
            if(users==null){
                return null;
            }else{
                MD5Utils md5Utils = new MD5Utils();
                String temp = md5Utils.getMD5ofStr(userpass);
                if(temp.equals(users.getUserPass())){
                    return users;
                }else{
                    return null;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } finally{
            DBUtils.closeConnection();
        }
    }
```

21.4.5、VIEW

```
    public static void userLoginView(){
        System.out.println("请输入用户名：");
        String userName = SCANNER.nextLine();
        System.out.println("请输入密码：");
        String userPass = SCANNER.nextLine();
```

```
        UsersBIZ usersBIZ = new UsersBIZ();
        Users users = usersBIZ.isLogin(userName, userPass);
        if(users==null){
            System.out.println("登录失败！");
        }else{
            System.out.println("登录成功！");
            RunStatusUtils.setLoginUsers(users);
        }
    }
```

21.5、用例 4：显示所有商品信息

21.5.1、GoodsDAO.java：查询所有方法

```
    public List<Goods> findAll() throws SQLException{
        List<Goods> goodsList = new ArrayList<Goods>();
        String sql = "select * from goods left join types "
                + " on goods.goodstype=types.typeid ";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        ResultSet rs = pstat.executeQuery();
        while (rs.next()) {
            Goods goods = new Goods();
            goods.setGoodsId(rs.getBigDecimal("goodsid"));
            goods.setGoodsName(rs.getString("goodsname"));
            goods.setGoodsPrice(rs.getBigDecimal("goodsprice"));
            goods.setGoodsNum(rs.getBigDecimal("goodsnum"));
            Types types = new Types();
            types.setTypeId(rs.getBigDecimal("typeid"));
            types.setTypeName(rs.getString("typename"));
            goods.setGoodsType(types);
            goodsList.add(goods);
        }
        return goodsList;
    }
```

21.5.2、GoodsBIZ.java：查询所有方法

```
    public List<Goods> findAll() {
        try {
            return goodsDAO.findAll();
```

```
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBUtils.closeConnection();
        }
    }
```

21.5.3、View

```
    public static void shopView(){
        GoodsBIZ goodsBIZ = new GoodsBIZ();
        List<Goods> goodsList = goodsBIZ.findAll();
        if(goodsList !=null && !goodsList.isEmpty()){
            System.out.println("编号\t 名称\t\t 价格\t 数量\t 类型");
            for (Goods goods : goodsList) {
                System.out.print(goods.getGoodsId()+"\t");
                System.out.print(goods.getGoodsName()+"\t");
                System.out.print(goods.getGoodsPrice()+"\t");
                System.out.print(goods.getGoodsNum()+"\t");
                System.out.println(goods.getGoodsType().getTypeName());
            }
        }else{
            System.out.println("查询所有商品失败！");
        }
    }
```

21.6、用例 5：购物车操作

购物车操作只要在程序端就可以完成了，不一定保存到数据库。

21.6.1、用例 5.1：创建购物车对象

1、创建**购物项**的实体类：

```
 * 对应购物车中增加的内容
public class ShopItem {
    private Goods goods;//购买的商品
    private int num;//购买的数量
    public ShopItem() {
    public ShopItem(Goods goods, int num) {
    public Goods getGoods() {
    public void setGoods(Goods goods) {
    public int getNum() {
    public void setNum(int num) {

}
```

2、创建购物车对象：

购物车是一个容器，存放的购物的内容，购买的内容用购物项表示

购物车就是一个集合对象。集合的泛型是 ShopItem。

集合会两类：

1 List

List<ShopItem> shopcar = new ArrayList<>();

2 Map 推荐

Map<商品的编号，ShopItem> shopcar = new HashMap<>( );

购物车对象就是一个 Map 的集合。

因为购物车是每个用户有自己的。ThreadLocal

```
public class RunStatusUtils {
    public static Admins loginAdmins;// 保存当前登录的管理员
    private static ThreadLocal<Users> usersLocal = new ThreadLocal<Users>();
    private static ThreadLocal<Map<BigDecimal, ShopItem>> shopCarLocal = new ThreadLocal<>();

    public static Map<BigDecimal, ShopItem> getShopCar() {
        Map<BigDecimal, ShopItem> shopcar = shopCarLocal.get();
        if (shopcar == null) {
            shopcar = new HashMap<BigDecimal, ShopItem>();
            shopCarLocal.set(shopcar);
        }
        return shopcar;
    }
```

```
package com.no4.utils;

import java.math.BigDecimal;
import java.util.HashMap;
```

```
import java.util.Map;
import com.no4.domain.Admins;
import com.no4.domain.ShopItem;
import com.no4.domain.Users;

public class RunStatusUtils {
    public static Admins loginAdmins;// 保存当前登录的管理员
    private static ThreadLocal<Users> usersLocal = new ThreadLocal<Users>();
    private static ThreadLocal<Map<BigDecimal, ShopItem>> shopCarLocal = new
ThreadLocal<>();

    public static Map<BigDecimal, ShopItem> getShopCar() {
        Map<BigDecimal, ShopItem> shopcar = shopCarLocal.get();
        if (shopcar == null) {
            shopcar = new HashMap<BigDecimal, ShopItem>();
            shopCarLocal.set(shopcar);
        }
        return shopcar;
    }

    public static void setLoginUsers(Users users) {
        usersLocal.set(users);
    }

    public static Users getLoginUsers() {
        return usersLocal.get();
    }
}
```

21.6.2、用例 5.2：购物车的业务类

1、GoodsDAO.java：按主键查询商品的方法

```
    public Goods findByGoodsId(BigDecimal goodsId) throws SQLException {
        String sql = "select * from goods left join types "
                + " on goods.goodstype=types.typeid " + " where goodsid = ?";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setBigDecimal(1, goodsId);
        ResultSet rs = pstat.executeQuery();
        if (rs.next()) {
```

```
            Goods goods = new Goods();
            goods.setGoodsId(rs.getBigDecimal("goodsid"));
            goods.setGoodsName(rs.getString("goodsname"));
            goods.setGoodsPrice(rs.getBigDecimal("goodsprice"));
            goods.setGoodsNum(rs.getBigDecimal("goodsnum"));
            Types types = new Types();
            types.setTypeId(rs.getBigDecimal("typeid"));
            types.setTypeName(rs.getString("typename"));
            goods.setGoodsType(types);
            return goods;
        } else {
            return null;
        }
    }
```

2、ShopCarBIZ.java：购物车的业务类

```
package com.no4.biz;

import java.math.BigDecimal;
import java.sql.SQLException;
import java.util.Map;
import com.no4.dao.GoodsDAO;
import com.no4.domain.Goods;
import com.no4.domain.ShopItem;
import com.no4.utils.DBUtils;
import com.no4.utils.RunStatusUtils;

public class ShopCarBIZ {
    private GoodsDAO goodsDAO = new GoodsDAO();
    public boolean addGoods(BigDecimal goodsId,int num){
        //1、获得购物车
        Map<BigDecimal, ShopItem> shopcar = RunStatusUtils.getShopCar();
        //2、判断这个商品是不是已经在购物车中
        if(shopcar.containsKey(goodsId)){
            //如果商品已经存在，就修改购买数量
            ShopItem item = shopcar.get(goodsId);
            int temp = item.getNum() + num;
            item.setNum(temp);
            return true;
        }else{
            //如果商品不存在 ，就第一次加到购物车。
            try {
                Goods goods = goodsDAO.findByGoodsId(goodsId);
                ShopItem shopItem = new ShopItem(goods, num);
```

```
                shopcar.put(goodsId, shopItem);
                return true;
            } catch (SQLException e) {
                e.printStackTrace();
                return false;
            }finally{
                DBUtils.closeConnection();
            }
        }
    }

    public Map<BigDecimal, ShopItem> getShopCar(){
        return RunStatusUtils.getShopCar();
    }
}
```

21.6.3、用例 5.3、购物车的视图层

选择商品加到购物车，增加成功之后显示购物车中的信息

```
    public static void shopView() {
        GoodsBIZ goodsBIZ = new GoodsBIZ();
        List<Goods> goodsList = goodsBIZ.findAll();
        if (goodsList != null && !goodsList.isEmpty()) {
            System.out.println("编号\t 名称\t\t 价格\t 数量\t 类型");
            for (Goods goods : goodsList) {
                System.out.print(goods.getGoodsId() + "\t");
                System.out.print(goods.getGoodsName() + "\t");
                System.out.print(goods.getGoodsPrice() + "\t");
                System.out.print(goods.getGoodsNum() + "\t");
                System.out.println(goods.getGoodsType().getTypeName());
            }
        } else {
            System.out.println("查询所有商品失败！");
        }
        System.out.println("请选择你要购买的商品：");
        BigDecimal goodsId = SCANNER.nextBigDecimal();
        SCANNER.nextLine();
        System.out.println("请输入数量:");
        int num = SCANNER.nextInt();
        SCANNER.nextLine();

        ShopCarBIZ shopCarBIZ = new ShopCarBIZ();
```

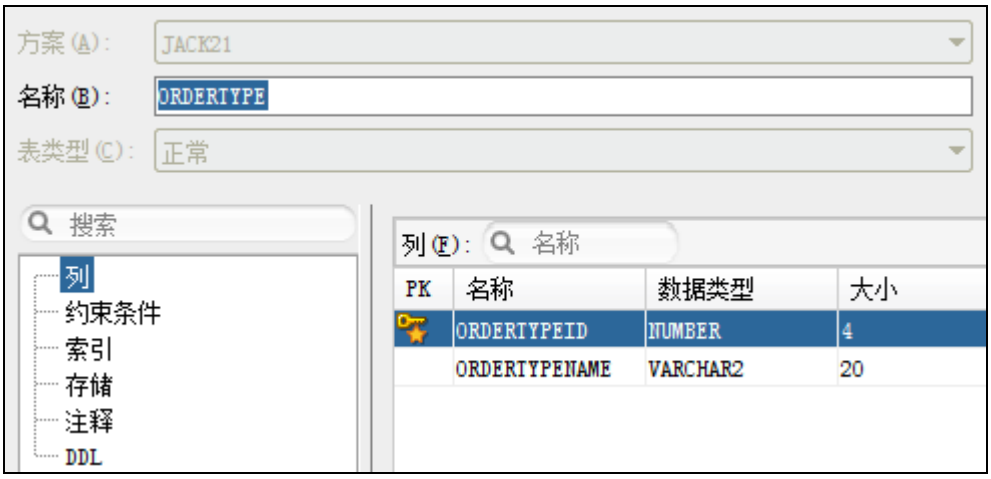
```
        if (shopCarBIZ.addGoods(goodsId, num)) {
            System.out.println("增加成功！");
            showShopCarView();
        } else {
            System.out.println("增加商品失败！");
        }
    }
```

```
    public static void showShopCarView() {
        System.out.println("显示购物车中的信息：");
        double sum = 0;
        Map<BigDecimal, ShopItem> shopcar = RunStatusUtils.getShopCar();
        Set<Entry<BigDecimal, ShopItem>> entrySet = shopcar.entrySet();
        System.out.println("编号\t 名称\t\t 价格\t 类型\t 购买数量\t 小计");
        for (Entry<BigDecimal, ShopItem> entry : entrySet) {
            Goods goods = entry.getValue().getGoods();
            System.out.print(goods.getGoodsId() + "\t");
            System.out.print(goods.getGoodsName() + "\t");
            System.out.print(goods.getGoodsPrice() + "\t");
            System.out.print(goods.getGoodsType().getTypeName() + "\t");
            System.out.print(entry.getValue().getNum() + "\t");
            System.out.println((goods.getGoodsPrice().doubleValue() * entry
                    .getValue().getNum()) );
            sum += goods.getGoodsPrice().doubleValue() * entry.getValue().getNum();
        }
        System.out.println("总计："+sum);

        System.out.println("1、继续购买");
        System.out.println("2、结算");

    }
```

21.7、用例 6：生成订单功能

21.7.1、数据库模型

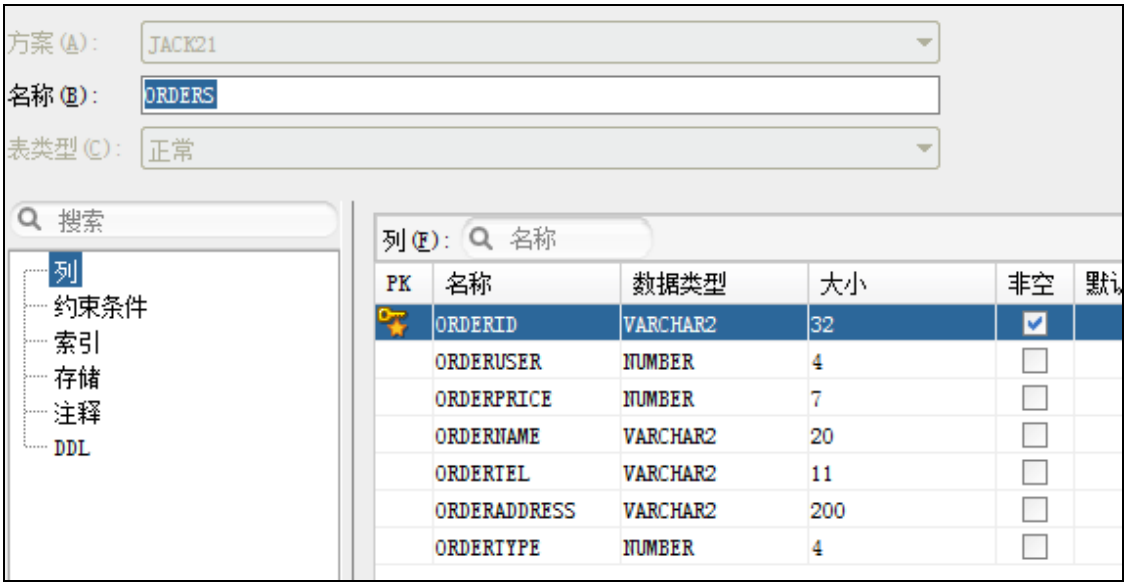
1、订单状态表

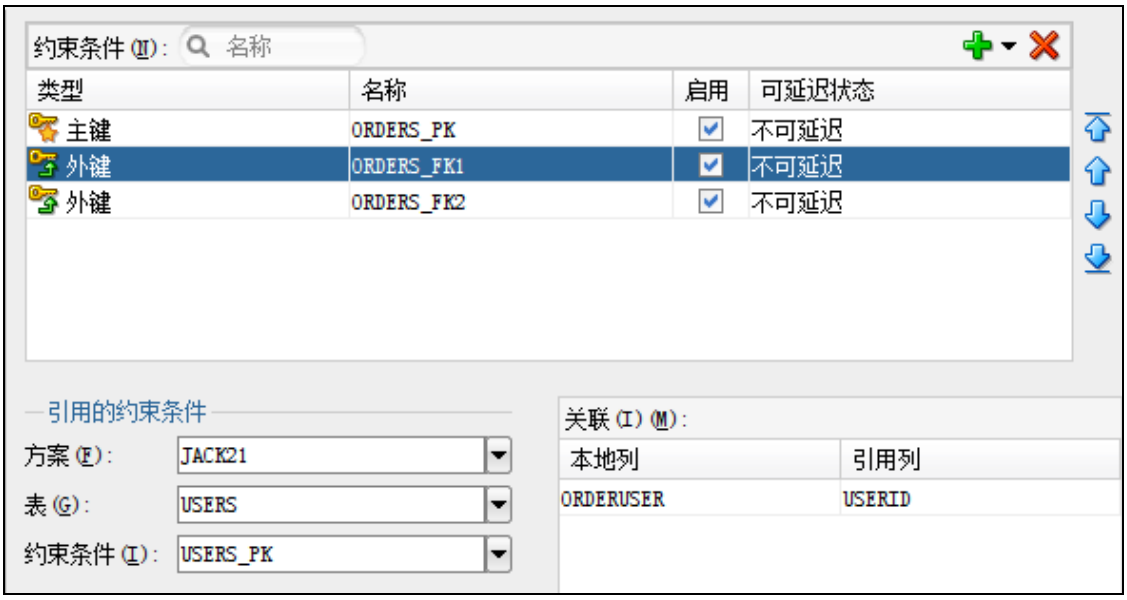


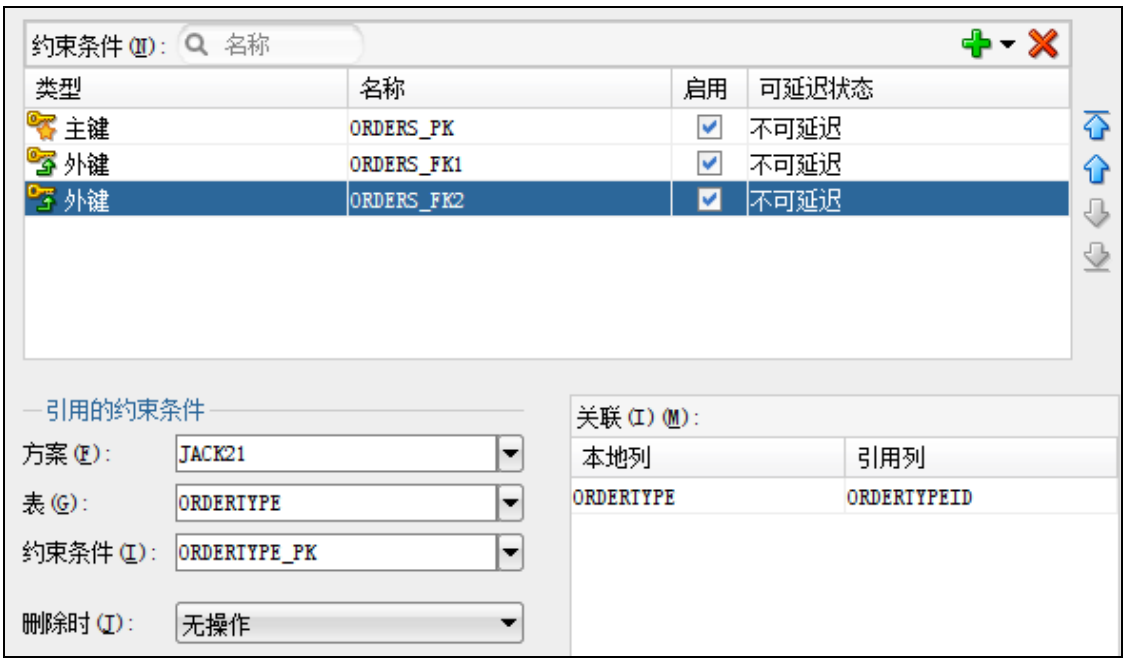

```
CREATE TABLE ORDERTYPE
(
  ORDERTYPEID NUMBER(4, 0) NOT NULL
, ORDERTYPENAME VARCHAR2(20 BYTE)
, CONSTRAINT ORDERTYPE_PK PRIMARY KEY ( ORDERTYPEID )
)
```

```
INSERT INTO "JACK21"."ORDERTYPE" (ORDERTYPEID, ORDERTYPENAME) VALUES ('1', '订单未付款')
INSERT INTO "JACK21"."ORDERTYPE" (ORDERTYPEID, ORDERTYPENAME) VALUES ('2', '订单已付款')
INSERT INTO "JACK21"."ORDERTYPE" (ORDERTYPEID, ORDERTYPENAME) VALUES ('3', '订单已发货')
```

2、订单表



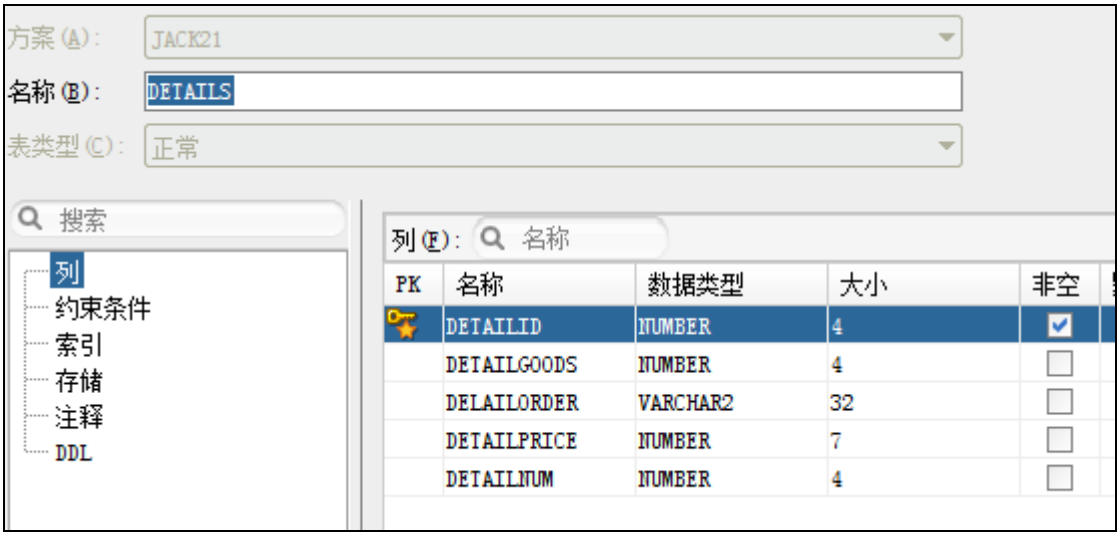


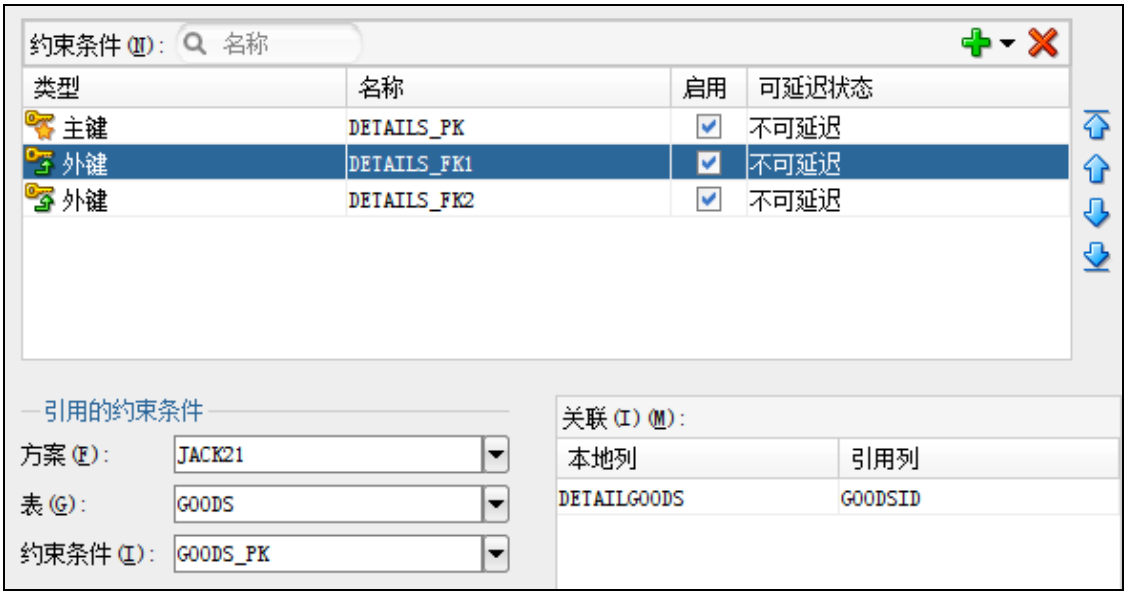


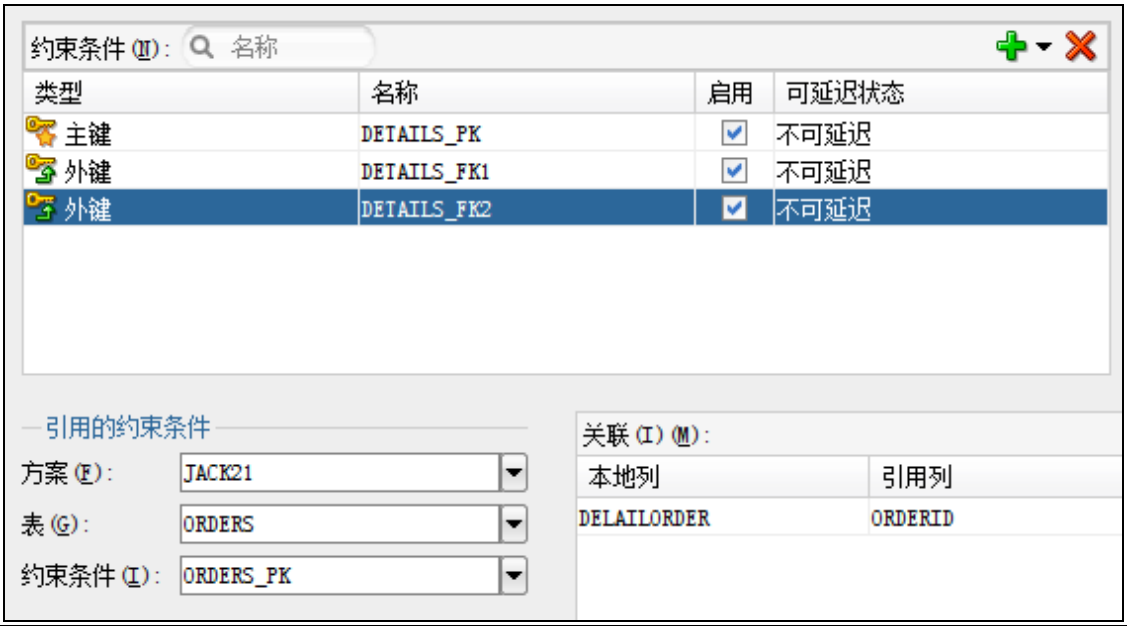

```
CREATE TABLE ORDERS
(
  ORDERID VARCHAR2(32 BYTE) NOT NULL
, ORDERUSER NUMBER(4, 0)
, ORDERPRICE NUMBER(7, 2)
, ORDERNAME VARCHAR2(20 BYTE)
, ORDERTEL VARCHAR2(11 BYTE)
, ORDERADDRESS VARCHAR2(200 BYTE)
, ORDERTYPE NUMBER(4, 0)
, CONSTRAINT ORDERS_PK PRIMARY KEY
  (
    ORDERID
  )
```

```
);

ALTER TABLE ORDERS
ADD CONSTRAINT ORDERS_FK1 FOREIGN KEY
(
  ORDERUSER
)
REFERENCES USERS
(
  USERID
);

ALTER TABLE ORDERS
ADD CONSTRAINT ORDERS_FK2 FOREIGN KEY
(
  ORDERTYPE
)
REFERENCES ORDERTYPE
(
  ORDERTYPEID
);
```

3、订单明细








```
CREATE TABLE DETAILS
(
  DETAILID NUMBER(4, 0) NOT NULL
, DETAILGOODS NUMBER(4, 0)
, DELAILORDER VARCHAR2(32 BYTE)
, DETAILPRICE NUMBER(7, 2)
, DETAILNUM NUMBER(4, 0)
, CONSTRAINT DETAILS_PK PRIMARY KEY
  (
    DETAILID
  )
);

ALTER TABLE DETAILS
```

```
ADD CONSTRAINT DETAILS_FK1 FOREIGN KEY
(
  DETAILGOODS
)
REFERENCES GOODS
(
  GOODSID
);

ALTER TABLE DETAILS
ADD CONSTRAINT DETAILS_FK2 FOREIGN KEY
(
  DELAILORDER
)
REFERENCES ORDERS
(
  ORDERID
);
```

21.7.2、生成订单的业务分析

生成订单业务时对数据库都进行哪些操作？

1、针对订单表做 insert into

2、针对订单明细表做 insert into

3、针对商品表做 update，减库存

使用哪个技术保存三个操作能一起完成。事务。

21.7.3、订单的业务编写 DAO 层

1、订单操作的 DAO：保存订单

```
public class OrderType {
    private BigDecimal orderTypeId;
    private String orderTypeName;
}
```

```
public class Orders {
    private String orderId;
    private String orderName;
    private BigDecimal orderPrice;
```

```
    private String orderTel;
    private String orderAddress;
    private OrderType orderType;
    private Users orderUser;
}
```

```
public class OrdersDAO {
    public void save(Orders orders) throws SQLException {
        String sql = "insert into "
                +
"orders(orderId,orderUser,orderPrice,ordername,ordertel,orderaddress,ordertype)"
                + "  values(?,?,?,?,?,?,?)";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setString(1, orders.getOrderId());
        pstat.setBigDecimal(2, orders.getOrderUser().getUserId());
        pstat.setBigDecimal(3, orders.getOrderPrice());
        pstat.setString(4, orders.getOrderName());
        pstat.setString(5, orders.getOrderTel());
        pstat.setString(6, orders.getOrderAddress());
        pstat.setBigDecimal(7, orders.getOrderType().getOrderTypeId());
        pstat.executeUpdate();
    }
}
```

2、订单明细操作的 DAO：保存明细

```
public class Details {
    private Orders detailOrder;
    private Goods detailGoods;
    private BigDecimal detailId;
    private BigDecimal detailNum;
    private BigDecimal detailPrice;
}
```

```
public class DetailsDAO {
    public void save(Details details) throws SQLException {
        String sql = "insert into
details(detailId,detailgoods,detailorder,detailNum,detailprice) "
                + " values(myseq.nextval,?,?,?,?)";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setBigDecimal(1, details.getDetailGoods().getGoodsId());
        pstat.setString(2, details.getDetailOrder().getOrderId());
        pstat.setBigDecimal(3, details.getDetailNum());
        pstat.setBigDecimal(4, details.getDetailPrice());
```

```
        pstat.executeUpdate();
    }
}
```

3、商品操作的 DAO-GoodsDAO.java：修改库存

```
    public void update(BigDecimal goodsId,int num) throws SQLException{
        String sql = "update goods set goodsnum = goodsnum-? where goodsid = ? ";
        Connection conn = DBUtils.getConnection();
        PreparedStatement pstat = conn.prepareStatement(sql);
        pstat.setInt(1, num);
        pstat.setBigDecimal(2, goodsId);
        pstat.executeUpdate();
    }
```

21.7.4、订单的业务方法的编写 BIZ 层

```
package com.no4.biz;

import java.math.BigDecimal;
import java.sql.SQLException;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import com.no4.dao.DetailsDAO;
import com.no4.dao.GoodsDAO;
import com.no4.dao.OrdersDAO;
import com.no4.domain.Details;
import com.no4.domain.Orders;
import com.no4.domain.ShopItem;
import com.no4.utils.DBUtils;
import com.no4.utils.RunStatusUtils;

public class OrdersBIZ {
    private OrdersDAO ordersDAO = new OrdersDAO();
    private DetailsDAO detailsDAO = new DetailsDAO();
    private GoodsDAO goodsDAO = new GoodsDAO();

    public boolean save(Orders orders) {
        try {
            DBUtils.getConnection().setAutoCommit(false);
            // 保存订单
            ordersDAO.save(orders);
            // 购物车
```

```
            Map<BigDecimal, ShopItem> shopcar = RunStatusUtils.getShopCar();
            Set<Entry<BigDecimal, ShopItem>> entrySet = shopcar.entrySet();
            for (Entry<BigDecimal, ShopItem> entry : entrySet) {
                Details details = new Details();// 明細
                details.setDetailGoods(entry.getValue().getGoods());
                details.setDetailOrder(orders);
                details.setDetailPrice(entry.getValue().getGoods()
                        .getGoodsPrice());
                details.setDetailNum(new BigDecimal(entry.getValue().getNum()));
                detailsDAO.save(details);// 保存明細
                goodsDAO.update(entry.getValue().getGoods().getGoodsId(), entry
                        .getValue().getNum());//修改库存数量
            }
            DBUtils.getConnection().commit();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            try {
                DBUtils.getConnection().rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            return false;
        } finally {
            DBUtils.closeConnection();
        }
    }
}
```

21.7.5、订单业务的视图 VIEW

```
        System.out.println("1、继续购买");
        System.out.println("2、结算");
        int choose = SCANNER.nextInt();SCANNER.nextLine();
        switch(choose){
        case 1:
            shopView();
            break;
        case 2:
            Orders orders = new Orders();
            orders.setOrderId(UUIDUtils.getUUID());
            System.out.println("请输入收件人姓名：");
```

```
            orders.setOrderName(SCANNER.nextLine());
            System.out.println("请输入收件人联系方式：");
            orders.setOrderTel(SCANNER.nextLine());
            System.out.println("请输入收件人地址：");
            orders.setOrderAddress(SCANNER.nextLine());
            orders.setOrderPrice(new BigDecimal(sum));
            OrderType orderType = new OrderType();
            orderType.setOrderTypeId(new BigDecimal(1));
            orders.setOrderType(orderType);
            orders.setOrderUser(RunStatusUtils.getLoginUsers());

            OrdersBIZ ordersBIZ = new OrdersBIZ();
            if(ordersBIZ.save(orders)){
                System.out.println("保存成功！");
                System.out.println("显示订单所有信息！");
            }else{
                System.out.println("保存失败！");
            }
            break;
        }
```

22、调用 Oracle 数据库带输入参数的存储过程

22.1、调用存储过程的语句格式有规定：

```
String sql = "{call 存储过程名（参数列表）}";
```

22.2、调用存储过程的对象使用 CallableStatement

```
CallableStatement cstat = conn.prepareCall(sql);
```

```
create or replace procedure proc_save_dept
(
  deptno in number
, dname in varchar2
, loc in varchar2
) as
begin
  insert into dept(deptno,dname,loc) values(deptno,dname,loc);
  commit;
```

```
end proc_save_dept;
```

```
    public void save(Dept dept) throws SQLException{
        String sql = "{call proc_save_dept(?,?,?)}";//调用存储过程
        Connection conn = DBUtils.getConnection();
        CallableStatement cstat = conn.prepareCall(sql);
        cstat.setBigDecimal(1, dept.getDeptno());
        cstat.setString(2, dept.getDname());
        cstat.setString(3, dept.getLoc());
        cstat.executeUpdate();
    }
```

```
        Dept dept = new Dept(new BigDecimal(4), "有关部门", "不知");
        DeptDAO deptDAO = new DeptDAO();
        try {
            DBUtils.getConnection().setAutoCommit(false);
            deptDAO.save(dept);
            DBUtils.getConnection().commit();
            System.out.println("成功！");
        } catch (SQLException e) {
            e.printStackTrace();
            try {
                DBUtils.getConnection().rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            System.out.println("失败！");
        } finally{
            DBUtils.closeConnection();
        }
```

23、调用 Oracle 数据库带输出参数的存储过程

```
create or replace procedure proc_getEnameByNo
(
  eno in number
, ename out varchar2
, sal out number
, comm out number
) as
begin
  select ename,sal,comm into ename,sal,comm from emp where emp.empno=eno;
end proc_getEnameByNo;
```

```
    public String findNameByNo(BigDecimal empno) throws SQLException{
```

```
        String sql ="{call proc_getEnameByNo(?,?,?,?)}";
        Connection conn = DBUtils.getConnection();
        CallableStatement cstat = conn.prepareCall(sql);
        cstat.setBigDecimal(1, empno);
        //针对 out 类型的参数，你第一步先注册
        cstat.registerOutParameter(2, Types.VARCHAR);
        cstat.registerOutParameter(3, OracleTypes.NUMBER);
        cstat.registerOutParameter(4, OracleTypes.NUMBER);
        cstat.executeUpdate();
        String ename = cstat.getString(2);
        return ename;
    }
```

```
        DeptDAO deptDAO = new DeptDAO();
        try {
            String ename = deptDAO.findNameByNo(new BigDecimal(7839));
            System.out.println(ename);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally{
            DBUtils.closeConnection();
        }
```

24、调用 Oracle 数据库带输入输出参数的存储过程

```
create or replace procedure p5
(
  sal in out number
) as
begin
  sal := sal * 2;
end p5;
```

```
        String sql ="{call p5(?)}";
        Connection conn = DBUtils.getConnection();
        CallableStatement cstat = conn.prepareCall(sql);
        cstat.setBigDecimal(1, new BigDecimal(100));
        cstat.registerOutParameter(1, OracleTypes.NUMBER);
        cstat.executeUpdate();
        BigDecimal sal = cstat.getBigDecimal(1);
        System.out.println(sal);
        DBUtils.closeConnection();
        System.out.println("end");
```

25、调用 Oracle 数据库的存储函数

```
create or replace function f1
(
  deptid in number
) return varchar2 as
  dname varchar2(14) ;
begin
  select dname into dname from dept where deptno = deptid;
  return dname;
end f1;
```

```
        String sql = "{? = call f1(?)}";
        Connection conn = DBUtils.getConnection();
        CallableStatement cstat = conn.prepareCall(sql);
        cstat.registerOutParameter(1, Types.VARCHAR);
        cstat.setBigDecimal(2, new BigDecimal(66));
        cstat.executeUpdate();
        String dname = cstat.getString(1);
        System.out.println(dname);
        DBUtils.closeConnection();
```

26、调用 Oracle 数据库的过程，返回多行查询结果，返回动态游标对象

```
create or replace package pack1 as
  type myrefcursortype is ref cursor;--动态游标类型
  procedure pack_proc(minsal in number,myref out myrefcursortype);
end pack1;
```

```
create or replace package body pack1 as
procedure pack_proc(minsal in number,myref out myrefcursortype) as
  begin
    open myref for  'select * from emp where sal >'|| to_char(minsal);
  end pack_proc;
end pack1;
```

```
String sql = "{call pack1.pack_proc(?,?)}";
Connection conn = DBUtils.getConnection();
CallableStatement cstat = conn.prepareCall(sql);
cstat.setFloat(1, 10000.0f);
cstat.registerOutParameter(2, OracleTypes.CURSOR);
cstat.executeUpdate();
```

```
ResultSet rs = (ResultSet) cstat.getObject(2);
while (rs.next()) {
    System.out.println(rs.getString("ename"));
}
DBUtils.closeConnection();
```