## 题 目: 作业二 数据库编程

姓名：向杰 学号：14202401788 班级：计科20-2BJ

1. 初步需求分析

编写图形界面程序，要求通过按钮对Goods表中的记录的查询、插入、删除及修改功能。

1. 方案

（1）初步分析需求，基于JavaFx窗体框架，选用相应控件

（2）初步设计窗体草图，细化编程流程，采用DAO模式进行数据库操作

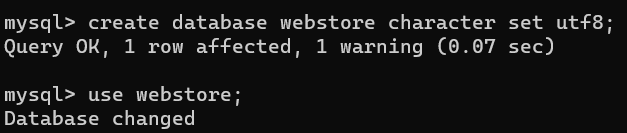
（3）创建数据库，创建数据表

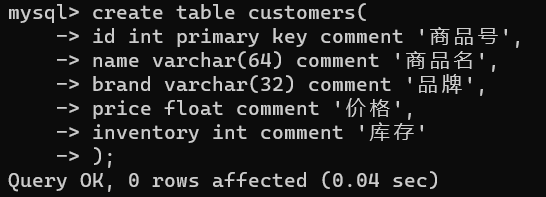
（4）使用vscode编写项目代码

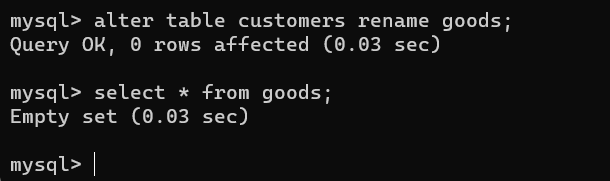
（5）绑定相关按钮事件

（6）进行测试

1. 具体实现









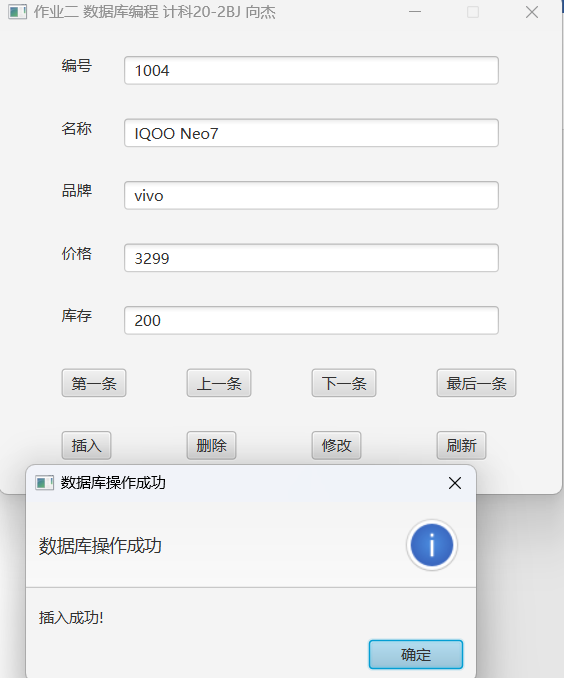




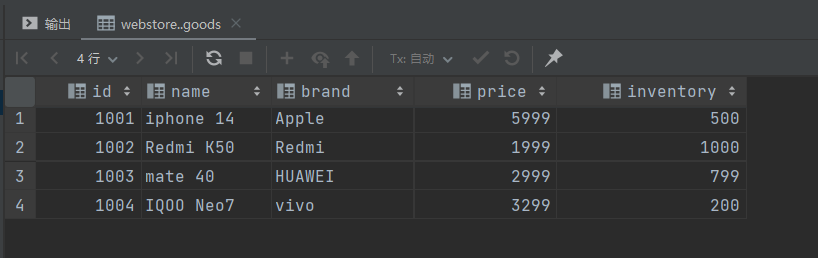




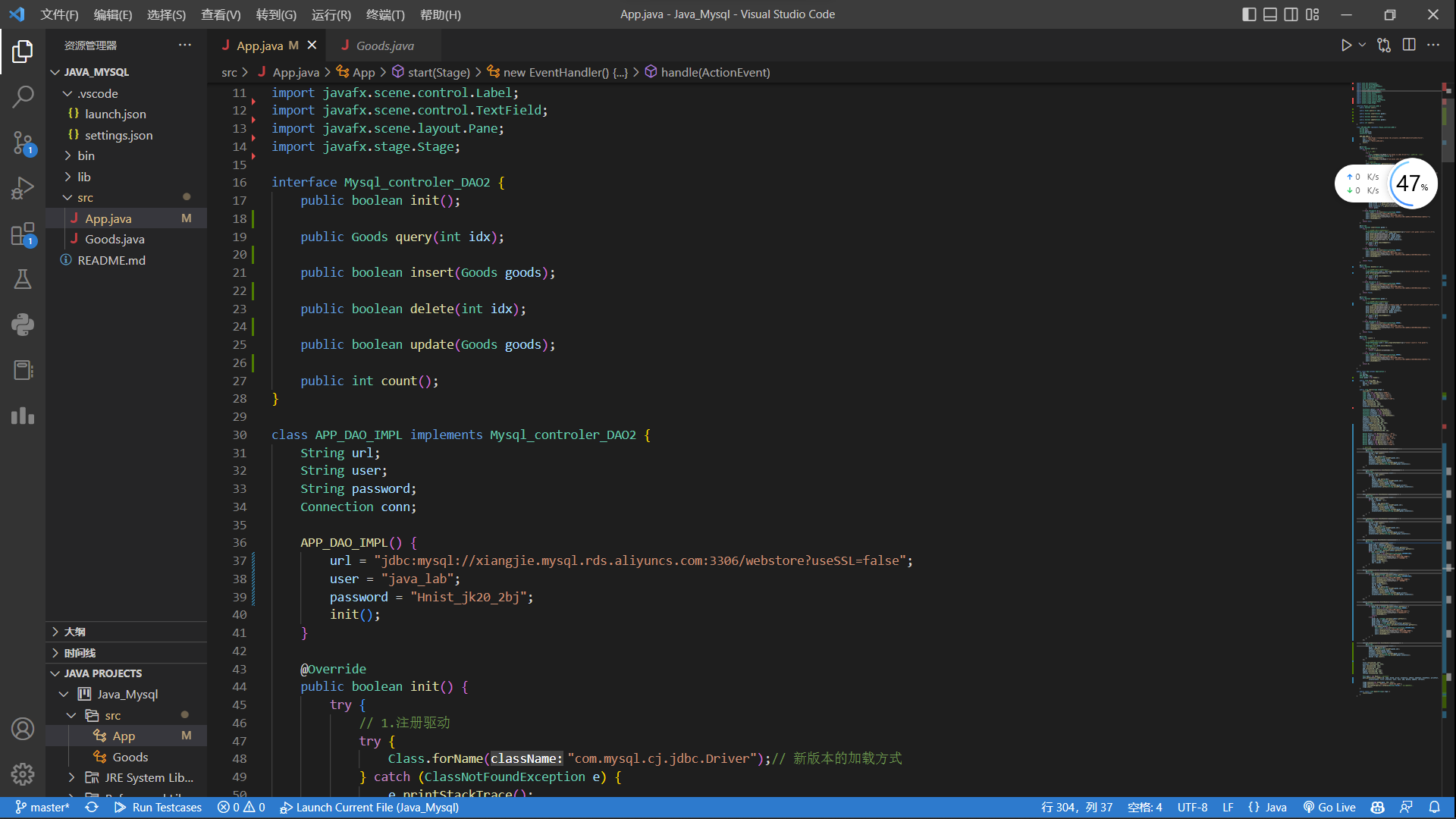




在后台数据库查看:



项目截图:



//App.java

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

interface Mysql\_controler\_DAO2 {

    public boolean init();

    public Goods query(int idx);

    public boolean insert(Goods goods);

    public boolean delete(int idx);

    public boolean update(Goods goods);

    public int count();

}

class APP\_DAO\_IMPL implements Mysql\_controler\_DAO2 {

    String url;

    String user;

    String password;

    Connection conn;

    APP\_DAO\_IMPL() {

        url = "jdbc:mysql://xiangjie.mysql.rds.aliyuncs.com:3306/webstore?useSSL=false";

        user = "java\_lab";

        password = "Hnist\_jk20\_2bj";

        init();

    }

    @Override

    public boolean init() {

        try {

            // 1.注册驱动

            try {

                Class.forName("com.mysql.cj.jdbc.Driver");// 新版本的加载方式

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

                Class.forName("com.mysql.jdbc.Driver");// 旧版本的加载方式

            }

            // 2.获取连接

            conn = DriverManager.getConnection(url, user, password);

        } catch (Exception ee) {

            ee.printStackTrace();

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库连接失败");

            alert.setHeaderText("数据库连接失败");

            alert.setContentText("本数据库使用的是远端数据库，请检查你的互联网连接是否成功!");

            alert.showAndWait();

            return false;

        }

        return true;

    }

    @Override

    public Goods query(int idx) {

        try {

            // 3.获取操作数据库的预处理对象

            PreparedStatement pstm = conn.prepareStatement("select \* from goods Limit ?,1");

            pstm.setInt(1, idx);

            // 4.执行SQL语句

            ResultSet rs = pstm.executeQuery();

            // 5.遍历结果集

            if (rs.next()) {

                Goods goods = new Goods();

                goods.id = rs.getInt("id");

                goods.name = rs.getString("name");

                goods.brand = rs.getString("brand");

                goods.price = rs.getFloat("price");

                goods.inventory = rs.getInt("inventory");

                return goods;

            }

        } catch (Exception e) {

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库操作失败");

            alert.setHeaderText("数据库操作失败");

            alert.setContentText("请检查互联网连接是否正常，或者数据输入是否合法!");

            alert.showAndWait();

        }

        return null;

    }

    @Override

    public boolean insert(Goods goods) {

        try {

            // 3.获取操作数据库的预处理对象

            PreparedStatement pstm = conn.prepareStatement("insert into goods values(?,?,?,?,?)");

            pstm.setInt(1, goods.id);

            pstm.setString(2, goods.name);

            pstm.setString(3, goods.brand);

            pstm.setFloat(4, goods.price);

            pstm.setInt(5, goods.inventory);

            // 4.执行SQL语句

            int count = pstm.executeUpdate();

            if (count > 0) {

                return true;

            }

        } catch (Exception e) {

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库操作失败");

            alert.setHeaderText("数据库操作失败");

            alert.setContentText("请检查互联网连接是否正常，或者数据输入是否合法!");

            alert.showAndWait();

        }

        return false;

    }

    @Override

    public boolean delete(int id) {

        try {

            // 3.获取操作数据库的预处理对象

            PreparedStatement pstm = conn.prepareStatement("delete from goods where id=?");

            pstm.setInt(1, id);

            // 4.执行SQL语句

            int count = pstm.executeUpdate();

            if (count > 0) {

                return true;

            }

        } catch (Exception e) {

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库操作失败");

            alert.setHeaderText("数据库操作失败");

            alert.setContentText("请检查互联网连接是否正常，或者数据输入是否合法!");

            alert.showAndWait();

        }

        return false;

    }

    @Override

    public boolean update(Goods goods) {

        try {

            // 3.获取操作数据库的预处理对象

            PreparedStatement pstm = conn

                    .prepareStatement("update goods set name=?,brand=?,price=?,inventory=? where id=?");

            pstm.setString(1, goods.name);

            pstm.setString(2, goods.brand);

            pstm.setFloat(3, goods.price);

            pstm.setInt(4, goods.inventory);

            pstm.setInt(5, goods.id);

            // 4.执行SQL语句

            int count = pstm.executeUpdate();

            if (count > 0) {

                return true;

            }

        } catch (Exception e) {

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库操作失败");

            alert.setHeaderText("数据库操作失败");

            alert.setContentText("请检查互联网连接是否正常，或者数据输入是否合法!");

            alert.showAndWait();

        }

        return false;

    }

    @Override

    public int count() {

        try {

            // 3.获取操作数据库的预处理对象

            PreparedStatement pstm = conn.prepareStatement("select count(\*) from goods");

            // 4.执行SQL语句

            ResultSet rs = pstm.executeQuery();

            // 5.遍历结果集

            if (rs.next()) {

                return rs.getInt(1);

            }

        } catch (Exception e) {

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("数据库操作失败");

            alert.setHeaderText("数据库操作失败");

            alert.setContentText("请检查互联网连接是否正常，或者数据输入是否合法!");

            alert.showAndWait();

        }

        return 0;

    }

}

public class App extends Application {

    int idx;

    int maxidx;

    APP\_DAO\_IMPL dao;

    Goods goods = new Goods();

    public void init\_APP() {

        dao = new APP\_DAO\_IMPL();

        maxidx = dao.count();

        idx = 0;

    }

    public void start(Stage stage) {

        init\_APP();

        Label id = new Label("编号");

        Label name = new Label("名称");

        Label brand = new Label("品牌");

        Label price = new Label("价格");

        Label inventory = new Label("库存");

        id.relocate(50, 20);

        name.relocate(50, 70);

        brand.relocate(50, 120);

        price.relocate(50, 170);

        inventory.relocate(50, 220);

        TextField idText = new TextField();

        TextField nameText = new TextField();

        TextField brandText = new TextField();

        TextField priceText = new TextField();

        TextField inventoryText = new TextField();

        idText.relocate(100, 20);

        nameText.relocate(100, 70);

        brandText.relocate(100, 120);

        priceText.relocate(100, 170);

        inventoryText.relocate(100, 220);

        idText.setPrefSize(300, 20);

        nameText.setPrefSize(300, 20);

        brandText.setPrefSize(300, 20);

        priceText.setPrefSize(300, 20);

        inventoryText.setPrefSize(300, 20);

        Button first = new Button("第一条");

        Button previous = new Button("上一条");

        Button next = new Button("下一条");

        Button last = new Button("最后一条");

        Button add = new Button("插入");

        Button delete = new Button("删除");

        Button update = new Button("修改");

        Button refresh = new Button("刷新");

        // 绑定事件

        first.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                maxidx = dao.count();

                idx = 0;

                goods = dao.query(idx);

                idText.setText(String.valueOf(goods.id));

                nameText.setText(goods.name);

                brandText.setText(goods.brand);

                priceText.setText(String.valueOf(goods.price));

                inventoryText.setText(String.valueOf(goods.inventory));

            }

        });

        previous.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                maxidx = dao.count();

                if (idx > 0) {

                    idx--;

                    goods = dao.query(idx);

                    idText.setText(String.valueOf(goods.id));

                    nameText.setText(goods.name);

                    brandText.setText(goods.brand);

                    priceText.setText(String.valueOf(goods.price));

                    inventoryText.setText(String.valueOf(goods.inventory));

                }

            }

        });

        next.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                maxidx = dao.count();

                if (idx < maxidx - 1) {

                    idx++;

                    goods = dao.query(idx);

                    idText.setText(String.valueOf(goods.id));

                    nameText.setText(goods.name);

                    brandText.setText(goods.brand);

                    priceText.setText(String.valueOf(goods.price));

                    inventoryText.setText(String.valueOf(goods.inventory));

                }

            }

        });

        last.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                maxidx = dao.count();

                idx = maxidx - 1;

                goods = dao.query(idx);

                idText.setText(String.valueOf(goods.id));

                nameText.setText(goods.name);

                brandText.setText(goods.brand);

                priceText.setText(String.valueOf(goods.price));

                inventoryText.setText(String.valueOf(goods.inventory));

            }

        });

        add.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                goods.name = nameText.getText();

                goods.brand = brandText.getText();

                goods.price = Float.parseFloat(priceText.getText());

                goods.inventory = Integer.parseInt(inventoryText.getText());

                goods.id = Integer.parseInt(idText.getText());

                if (dao.insert(goods)) {

                    Alert alert = new Alert(Alert.AlertType.INFORMATION);

                    alert.setTitle("数据库操作成功");

                    alert.setHeaderText("数据库操作成功");

                    alert.setContentText("插入成功!");

                    alert.showAndWait();

                    maxidx = dao.count();

                    idx = maxidx - 1;

                }

            }

        });

        delete.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                if (dao.delete(Integer.parseInt(idText.getText()))) {

                    Alert alert = new Alert(Alert.AlertType.INFORMATION);

                    alert.setTitle("数据库操作成功");

                    alert.setHeaderText("数据库操作成功");

                    alert.setContentText("删除成功!");

                    alert.showAndWait();

                    maxidx = dao.count();

                    idx %= maxidx;

                    goods = dao.query(idx);

                    idText.setText(String.valueOf(goods.id));

                    nameText.setText(goods.name);

                    brandText.setText(goods.brand);

                    priceText.setText(String.valueOf(goods.price));

                    inventoryText.setText(String.valueOf(goods.inventory));

                }

            }

        });

        update.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                if (goods.id != Integer.parseInt(idText.getText())) {

                    Alert alert = new Alert(Alert.AlertType.ERROR);

                    alert.setTitle("数据库操作失败");

                    alert.setHeaderText("数据库操作失败");

                    alert.setContentText("ID不可修改!");

                    alert.showAndWait();

                    return;

                } else {

                    goods.id = Integer.parseInt(idText.getText());

                    goods.name = nameText.getText();

                    goods.brand = brandText.getText();

                    goods.price = Float.parseFloat(priceText.getText());

                    goods.inventory = Integer.parseInt(inventoryText.getText());

                    if (dao.update(goods)) {

                        Alert alert = new Alert(Alert.AlertType.INFORMATION);

                        alert.setTitle("数据库操作成功");

                        alert.setHeaderText("数据库操作成功");

                        alert.setContentText("修改成功!");

                        alert.showAndWait();

                    }

                }

            }

        });

        refresh.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                goods = dao.query(idx);

                idText.setText(String.valueOf(goods.id));

                nameText.setText(goods.name);

                brandText.setText(goods.brand);

                priceText.setText(String.valueOf(goods.price));

                inventoryText.setText(String.valueOf(goods.inventory));

                maxidx = dao.count();

            }

        });

        first.relocate(50, 270);

        previous.relocate(150, 270);

        next.relocate(250, 270);

        last.relocate(350, 270);

        add.relocate(50, 320);

        delete.relocate(150, 320);

        update.relocate(250, 320);

        refresh.relocate(350, 320);

        Pane pane = new Pane();// 新建pane

        pane.getChildren().addAll(id, name, brand, price, inventory, idText, nameText, brandText, priceText,

                inventoryText, first, previous, next, last, add, delete, update, refresh);

        stage.setScene(new Scene(pane, 450, 370));

        stage.setTitle("作业二 数据库编程 计科20-2BJ 向杰");

        stage.resizableProperty().setValue(Boolean.FALSE);// 禁用最大化按钮

        stage.show();

    }

    public static void main(String[] args) {

        launch(args);

    }

}

//Goods.java

public class Goods {

    int id;

    String name;

    String brand;

    float price;

    int inventory;

    public Goods(int id, String name, String brand, float price, int inventory) {

        this.id = id;

        this.name = name;

        this.brand = brand;

        this.price = price;

        this.inventory = inventory;

    }

    public Goods(){};

}

四、总结

在本次实验中，我收获颇丰，让我学会了使用Javafx进行窗体设计和数据库编程，掌握了基本的窗体设计知识和数据库相关知识，丰富了我使用Javafx设计窗体和使用MySQL进行数据库操作的经验，在以后做项目时我会更加成熟。