长沙理工大学

《Java 程序设计实训》报告

仓库信息管理系统设计

学　　院	计算机与通信工程	专　　业	数据科学与大数据技术
班　　级	大数据 1901 班	学　　号	201908170113
学生姓名	陈 翊	指导教师	尹　波
课程成绩		完成日期	2021 年 1 月 4 日

Java 程序设计实训成绩评定标准

毕业要求	考核与评价方式及成绩比例（%）			成绩比例（%）
	系统演示	项目答辩	实训报告	
实训目标 1：**基础理论知识**	/	/	30	30
实训目标 2：**创新精神、设计能力**	20	/	/	20
实训目标 3：**问题分析和应用**	/	30	/	30
实训目标 4：**综合素质、项目适应力**	/	/	10	10
实训目标 5：**团队合作、社会责任感**	/	/	10	10
合计	20	30	50	100

Java 程序设计实训成绩评定表

毕业要求	考核与评价方式及成绩比例（%）			成绩比例（%）
	系统演示	项目答辩	实训报告	
实训目标 1：**基础理论知识**	/	/		
实训目标 2：**创新精神、设计能力**		/	/	
实训目标 3：**问题分析和应用**	/		/	
实训目标 4：**综合素质、项目适应力**	/	/		
实训目标 5：**团队合作、社会责任感**	/	/		
合计				

指导教师对 Java 程序设计实训的评定意见

综合成绩 _____指导教师签字_____ 年 月 日

仓库信息管理系统设计

学生姓名：陈 翊　　指导老师：尹 波

摘　要 仓库由贮存物品的库房、运输传送设施、出入库房的输送管道和设备以及消防设施、管理用房等组成。是保管、储存物品的建筑物和场所的总称。仓库中往往存储着同类别或是不同类别的货物，货物的数量，货物的负责人和入库的时间等信息对货物的管理有着十分重要的作用。在传统的管理模式下，仓库货物的信息往往由管理员在单页上记录，这样不仅效率低下，在对数据记录的增删改查时也可能造成数据的丢失或不一致。本系统在这样的需求下应运而生，使用 JavaFX 开发简单易用的图形界面，结合 SQL 数据库以实现对仓库货物信息的统一管理和简单分析。在功能上，系统可以对仓库数据进行增删改查的操作，在此基础上还增加了批量操作和可视化的操作，并额外提供用户管理的功能。思路上采用 Java 面向对象技术，利用了 JavaFX 的封装特性，并辅佐以 Java 众多强大的 API，实现了系统的高效快速开发。

关键词　JavaFX；Java；数据库；GUI 程序设计；信息管理系统；面向对象程序设计

Design of Warehouse Information Management System

Student name: Yi Chen Advisor：Bo Yin

Abstract The warehouse is composed of the warehouse for storing goods, transportation facilities, transportation pipelines and equipment entering and leaving the warehouse, fire-fighting facilities, management room, etc. It is the general name of the buildings and places where goods are kept and stored. Warehouse often stores the same or different types of goods, the quantity of goods, the person in charge of goods and the time of warehousing and other information plays a very important role in the management of goods. In the traditional management mode, the warehouse cargo information is often recorded by the administrator on a single page, which is not only inefficient, but also may cause data loss or inconsistency when the data records are added, deleted or modified. This system arises at the historic moment under such demand, uses JavaFX to develop simple and easy-to-use graphical interface, combines with SQL database to realize the unified management and simple analysis of warehouse cargo information. In terms of function, the system can add, delete, modify and query the warehouse data. On this basis, it also adds batch operation and visual operation, and provides additional user management functions. Java object-oriented technology, JavaFX encapsulation features and many powerful Java APIs are used to realize the efficient and rapid development of the system.

Keywords JavaFx; Java; Information management system; design of GUI program; object-oriented programming

目　录

1. 引言

1.1 项目背景

仓库在现实生活中的用途十分广泛，各种商城、超市要利用仓库存放物资，药房、医院等要利用仓库存放药品，企业、工厂等要存放原材料、生产成品，因此仓库的管理成了一项十分重要的工作。

随着计算机的应用普及，目前大多数企业的仓库管理数据资料已开始采用计算机数据系统管理，但数据还是采用先纸张记录、再手工输入计算机的方式进行采集和统计整理。这不仅造成大量的人力资源浪费，而且由于人为的因素，数据录入速度慢、准确率低。随着企业规模的不断发展，仓库管理的物资种类机数量在不断增加、出入库频率剧增，仓库管理作业也已十分复杂和多样化，传统的人工仓库作业模式和数据采集方式已难以满足仓库管理的快速、准确要求，严重影响了企业的运行工作效率，成为制约企业发展的一大障碍。

1.2 项目意义

人工管理仓库既费时又费力，而且容易造成混乱，严重时会影响商城、企业的正常运作，造成恶劣的后果。随着信息技术的发展，办公自动化的普及，如何快速、高效、便捷地管理仓库收到了较高的关注。

利用计算机桌面程序，实现对仓库数据的操作与分析可以大大提高传统管理模式的工作效率，减小出错的可能性。这样的一套管理系统将改变传统单页记录的模式，让自动化办公成为现实。

1.3 系统主要任务

从普通用户的角度，实现数据的查询和检索。

从管理用户的角度，实现数据的批量处理与检索。

从系统用户的角度，实现以上功能的同时可以对其他用户进行管理。

系统应该可以适配多种版本的数据库连接，在数据库配置变动和可以快速设置新连接。对于存放在数据库的数据，实现基本的增删改查功能后，对数据经行简单的分析和

筛查，对繁琐的信息简单化、可视化。对货物的信息数据化，通过算法智能查找空位，将重量大的货物优先排放在货架的底层，在货架信息发生变动时可以快速方便的对整体数据经行移动和整合，对所有仓库货物信息，按条件（编号、名称、负责人、入库信息、描述）查询浏览。

设计简单实用的用户界面，对不同级别的用户做出的不同操作执行后反馈。拥有较好的交互性，使不同专业背景的用户可以快速上手。

2. 系统方案设计

2.1 需求分析

总体设计目标：数据的处理准确，程序的逻辑通顺，交互界面易操作，不存在重大的漏洞。使用者在没有接触过系统的前提下可以快速上手，操作简单，逻辑清晰。

本系统要求能对所有仓库货物信息，按条件（编号、名称、负责人、入库信息、描述）查询浏览，对数据经行简单的分析处理以及实现与数据库相连的增删改操作。

具体功能描述如下：

① 添加新的货物信息；

② 修改已有货物信息；

③ 删除已有货物信息；

④ 查找已有货物信息；

⑤ 查看所有货物信息；

⑥ 按指定信息过滤；

⑦ 连接不同类型的数据库；

⑧ 为不同的使用者安排不同的权限；

⑨ 批量导入 xls 文件的数据；

⑩ 实现数据分析与可视化；

⑪ 实现对用户的管理操作；

2.2 系统流程图

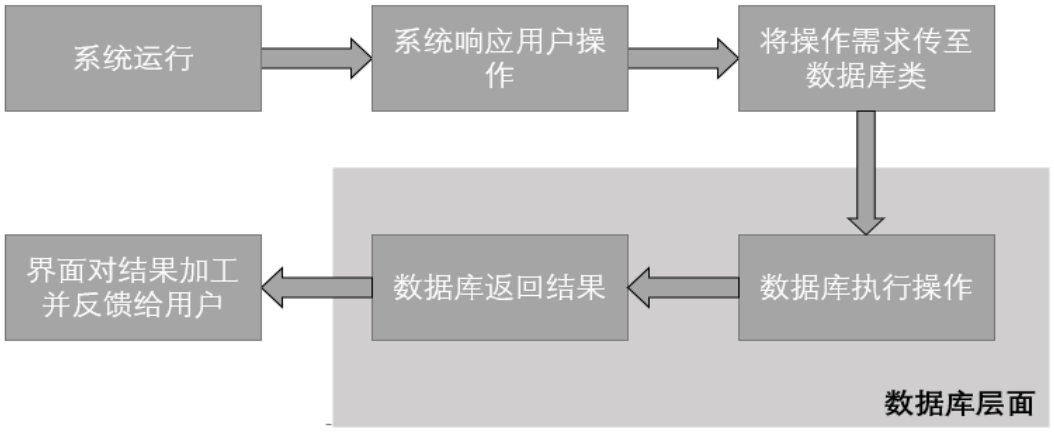

图 2.1 主流程图

2.3 功能结构分析

① 读取数据库模块：根据用户自定义的连接 URL 和账号密码，系统将实现对数据库的连接，为之后的数据操作做准备。

② 添加模块：创建对话框，指引用户输入相应的数据以添加货物信息。信息无误则存入数据库，同时在界面。

③ 删除模块：根据用户点击的行数获得用户想要删除的信息，删除操作应当需要高级权限且需要再次确认，确认后将操作数据库将记录删除，并刷新界面数据。

④ 查找模块：根据用户选择的关键字和输入的字段，将所有数据库中符合的信息显示在屏幕上。

⑤ 修改模块：根据用户点击的行数获得用户想要修改的信息，当用户在相应的单元格中输入修改后的信息后，将数据直接提交至数据库，完成一次修改操作

⑥ 导入 xls 文件模块：打开并读取用户选择的 xls 类型文件，将数据经过合法性校验后筛选出可以导入数据库的信息，经过用户确认后批量导入数据至数据库。

⑦ 用户模块：用户可以登录至不同权限的账户实现不同的操作，用户信息也要实现增删改查。

⑧ 交互模块：不同的控件点击后要和用户实现交互，如操作是否成功，下一步该如何等。

3. 系统软件开发环境

本系统使用eclipse作为程序开发工具，辅佐以JavaFX Scene Builder，以实现GUI程序设计。数据库方面选用Microsoft SQL，配合SQL Server Management Studio经行数据库管理。使用exe4j Wizard和inno setup编译器完成系统的发打包并生成可在64为系统运行的可执行文件。

图 3.1 eclipse logo

图 3.2 SQL logo

4. 系统实现

4.1 主要数据结构

4.1.1 Cargo 类

作为仓库管理系统，最主要的就是仓库中的货物实体类。一个货物应当有以下的属性：

```
private final SimpleStringProperty uid;          //uid（主键）
private final SimpleStringProperty block;        //所在仓库块号
private final SimpleStringProperty shelf;        //所在仓库区的货架号
private final SimpleStringProperty layer;        //所在仓库货架的层号
private final SimpleStringProperty cargoName;    //货物的名称
private final SimpleStringProperty manage;       //货物的负责人
private final SimpleStringProperty weight;       //重量
private final SimpleStringProperty inTime;       //入库时间
private final SimpleStringProperty description;  //附加描述
```

由于要将货物信息在 JavaFX 的 TableView 中显示，而 TableView 的存储规范为 SimpleStringProperty 类，所以这里的货物属性都使用 SimpleStringProperty 类定义。

之后是 Cargo 类的有参构造方法 toString 方法和 getter 与 setter，这里不再赘述。

4.1.2 User 类

每个对系统进行操作的用户都是一个实体，在操作前需要登录系统。一个用户具有以下属性：

```
private final SimpleStringProperty uid;          //uid（主键）
private final SimpleStringProperty userName;     //用户名
private final SimpleStringProperty passwd;       //登录密码
private final SimpleStringProperty permission;   //系统权限
private final SimpleStringProperty mailAddr;     //邮箱地址
private final SimpleStringProperty status;       //状态
```

和 Cargo 一样，用户信息在 JavaFX 的 TableView 中显示，而 TableView 的存储规范为 SimpleStringProperty 类，所以这里的用户属性都使用 SimpleStringProperty 类定义。

之后是 User 类的有参构造方法 toString 方法和 getter 与 setter，这里不再赘述。

4.2 数据库（SQL）类

由于本管理系统使用数据库来存储信息，许多模块都涵盖了与数据库之间的数据交换，所以，专门设置一个类用于处理数据库的事务是很有必要的，这样不仅可以增加代码的复用性，还可以方便数据的集中处理，对于程序的整体结构也是有优化作用的。

连接数据库用到了 java.jdbc 包，在导入了 jre 后，就可以使用如下方式连接数据库：

```
DriverManager.getConnection(DB_URL, user, passwd);
```

其中 DB_URL 为数据库连接地址，其中包含了端口，服务器地址等信息。user 和 passwd 分别是连接数据库时的帐号和密码。为了符合面向对象的规范，SQL 类中包含了连接的相关属性：

```
/*默认数据库的配置*/
private static String DB_URL = "jdbc:sqlserver://localhost;DatabaseName=master";
private static String serverURL = "localhost";
private static String databaseName = "master";
private static String user = "chenyi";
private static String passwd = "xxxxx";
/*数据库连接的属性*/
private Connection con;
private Statement state;
private ResultSet rs;
```

现在以 isResultSetEmpty 方法解释 SQL 类的处理逻辑。

```
public boolean isResultSetEmpty(String sql) throws SQLException{
    //返回 sql 语句 select 结果是否为空的逻辑值 如果空则返回 true
    boolean flag = true;
    this.getConnection();
    rs = state.executeQuery(sql);
    while (rs.next()) {
        flag = false;
    }
    this.close();
    return flag;
}
```

可以看到，方法的参数传入一个 String 类型的数据库语句，在调用数据库 executeQuery

方法前，需要调用 getConnection 方法连接至数据库，保存连接的 Connection 类属性。之后，用 rs（result set）保存数据库语句执行后返回的结果集。访问结果集内容的方法类似于 hashmap，使用 next 对集合进行遍历。在此方法中，如果结果集不为空，则返回 false，否则返回 true。由于在调用 getConnection 时可能会失败，所以在要在申明处加上异常。方法返回结束前，调用了 close 方法，关闭了连接，清空了结果集，单次执行结束。

SQL 类的其他方法都可大致遵循这样的模式：

(1)获得数据库的连接；

(2)执行相关的 SQL 语句；

(3)如果有返回值则遍历结果集；

SQL 作为工具类，大多数时间被其他模块调用，处理了和数据相关的事务，由于复用性高，代码的封装也应该留有较多的灵活的参数。

4.3 用户界面介绍

JavaFx 的界面四个部分组成：.fxml .css 和两个.java 文件。它们之间的关系是：.fxml 文件负责界面的外观版面等基本信息，是用户一眼所看到的内容。.css 负责整体渲染界面的样式和效果，合理地使用可以简化界面的美化。两个 java 文件一个用于界面的生成，即存放了.fxml .css 文件的地址以及界面内的内容、标题、图标等；另外一个是界面的控制器文件，它内含了界面的事件处理方法，使界面拥有了交互性。

.fxml .css 文件在界面生成时被访问，用于界面的生成，在.fxml 中定义了控件的识别 id 和控件被触发后需要响应的方法名，这样当用户在界面出发事件时（鼠标点击、键盘输入、拖动等），控制器文件就会调用相应的函数去相应（如下图 4.1）。

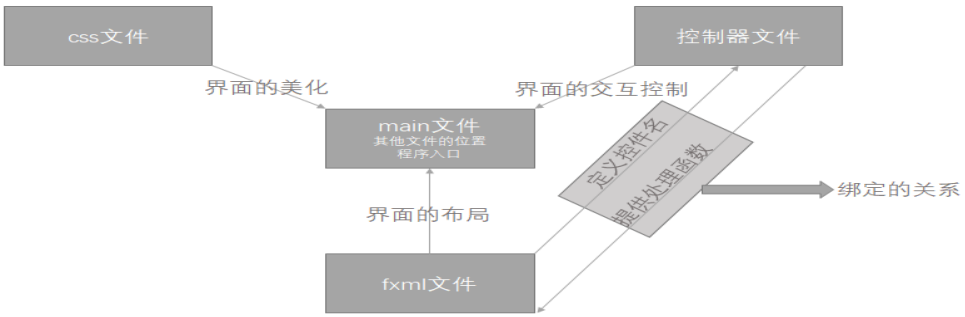

图 4.1 文件结构

4.3.1 主界面 MyScene.fxml

本管理系统的界面设计采用了工具栏+操作面板的模式。所有的控制面板都会出现在右侧同一区域，固定了视觉区域。左侧为按钮工具栏，用来实现不同功能面板之间的转换。右侧的不同功能的面板处于一种“叠加”的状态，同一时刻只有一个面板处于可见状态，举例来说，就好比每个功能面板是一个演员，只有当需要他表演时才将聚光灯打向他，而其他的演员处于不可见的状态。

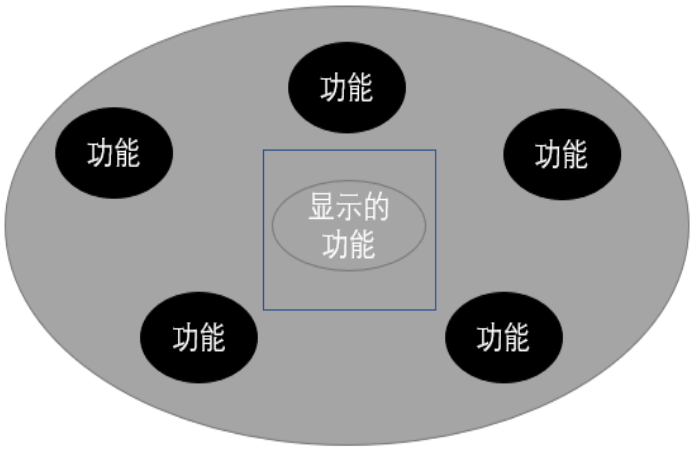

图 4.2 舞台的显示

图 4.3 系统开始运行

由图 4.3 可见，系统在刚刚启动时关闭了除“连接设置”和“帮助”外的所以按钮，即用户在连接至数据库后才可以进行下一步操作。

图 4.4

同样的，只有在用户登录成功后才可以对数据进行访问和修改。

图 4.5 用户登录模式

图 4.6 数据操作模式

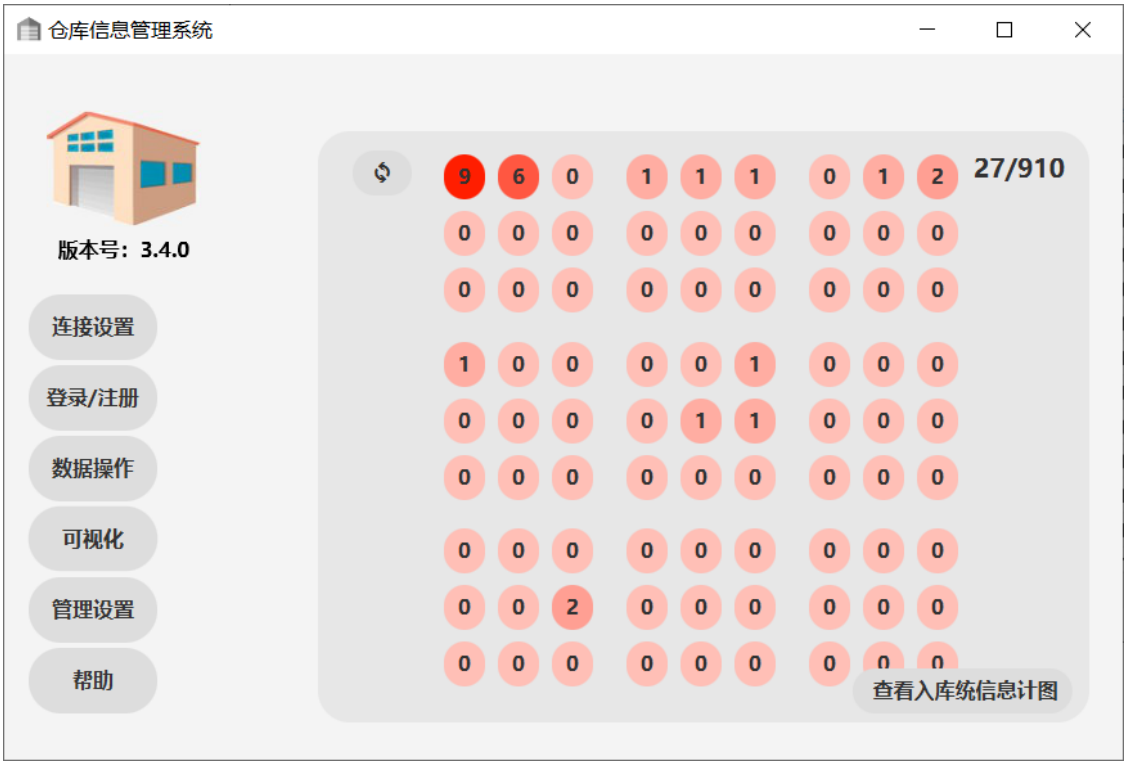

图 4.7 可视化模式

图 4.8 管理模式

图 4.9 表的行相应双击事件

表 4.1 各个操作面板的主要形态

操作功能	**访问所需权限**	**主要功能**
连接设置	无	保存和测试数据库的连接设置
登录/注册	已经连接至数据库	登录至已有用户或注册新用户
数据操作	用户已经成功登录	对数据经行增删改查（其中增删改）需要管理权限
可视化	用户已经成功登录	在数据面板上（鼠标悬停在圆形上）查看数据，或是查看统计图表
管理设置	用户已经成功登录且为管理员或以上身份	查看用户的信息和状态并进行更改
帮助	无	回到提示界面或打开帮助文档

其他特点：数据表和用户信息表的每一行，都可以通过双击（需要有管理员权限）来实现更多的功能，如货物的删除、移动，用户的删除、提升/降级权限等。将鼠标悬停在控件上（不做点击）可以看到提示。

4.3.2 添加货物界面 AddCargoScene.fxml

添加货物界面通过单击添加按钮以打开（如图 4.10）。该界面在收集用户的输入后会要求用户测试信息的合法性，在用户输入的信息合法无误后方可提交至数据库。为了防止用户在测试通过后修改信息再提交，该界面无论测试是否通过，在将货物信息提交至数据库前都会都数据进行合法性校验。

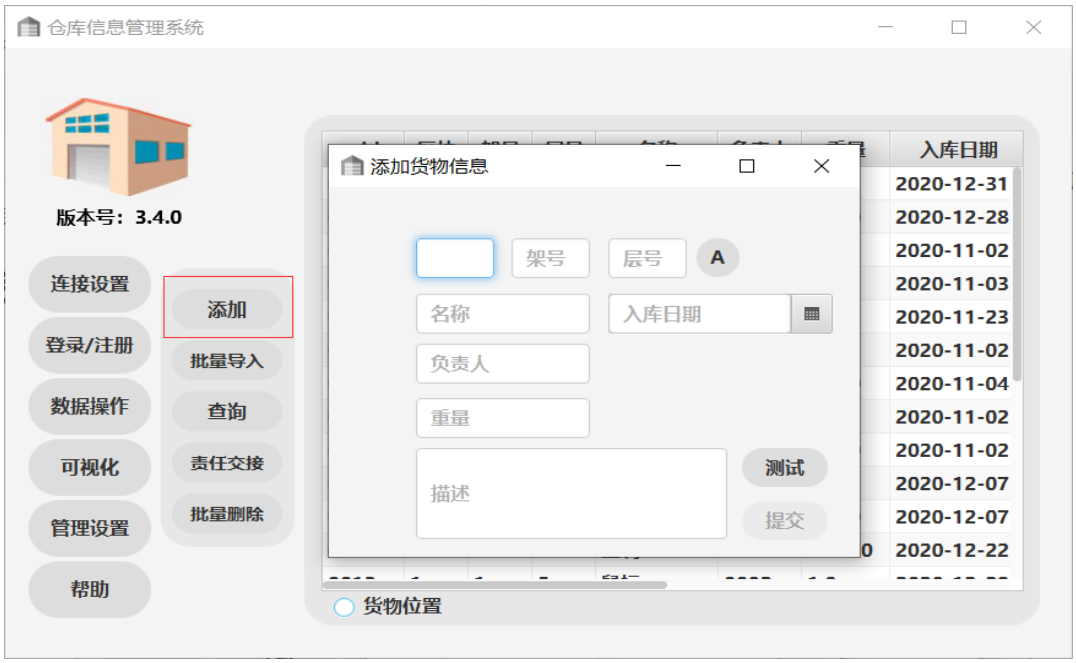

图 4.10 添加货物信息界面

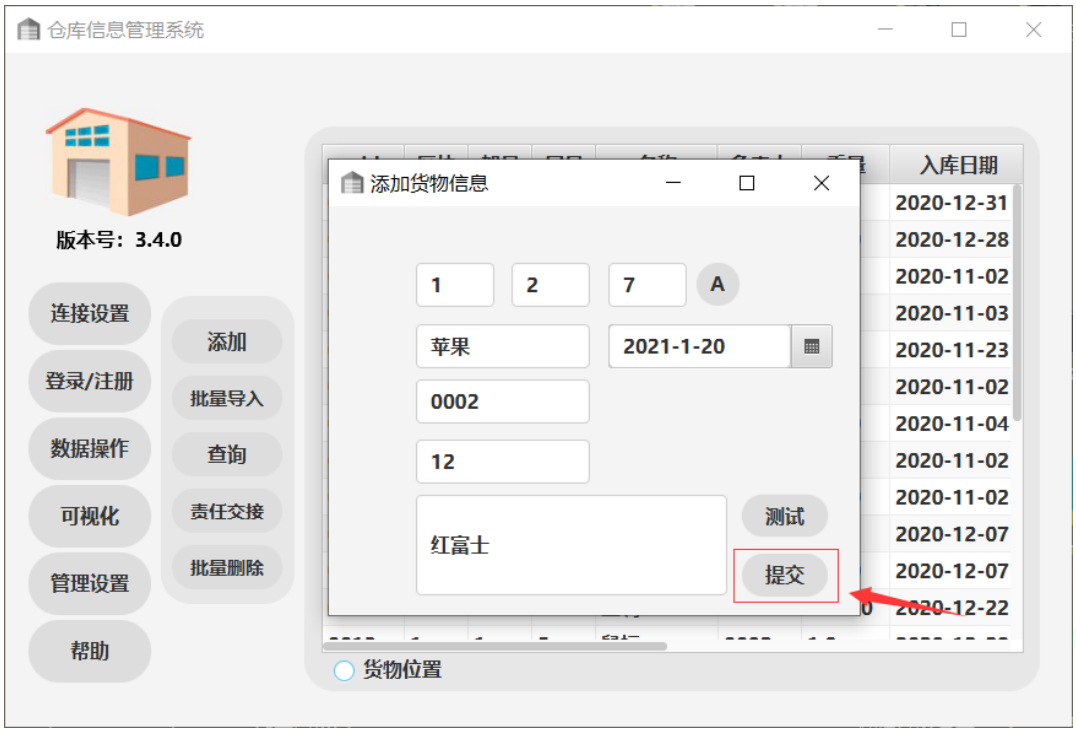

图 4.11 信息无误允许提交

值得一提的是，负责人属性和用户的 uid 构成参照关系，所以在添加货物信息前要确保负责人已经存在在数据库中，否则无法通过测试。

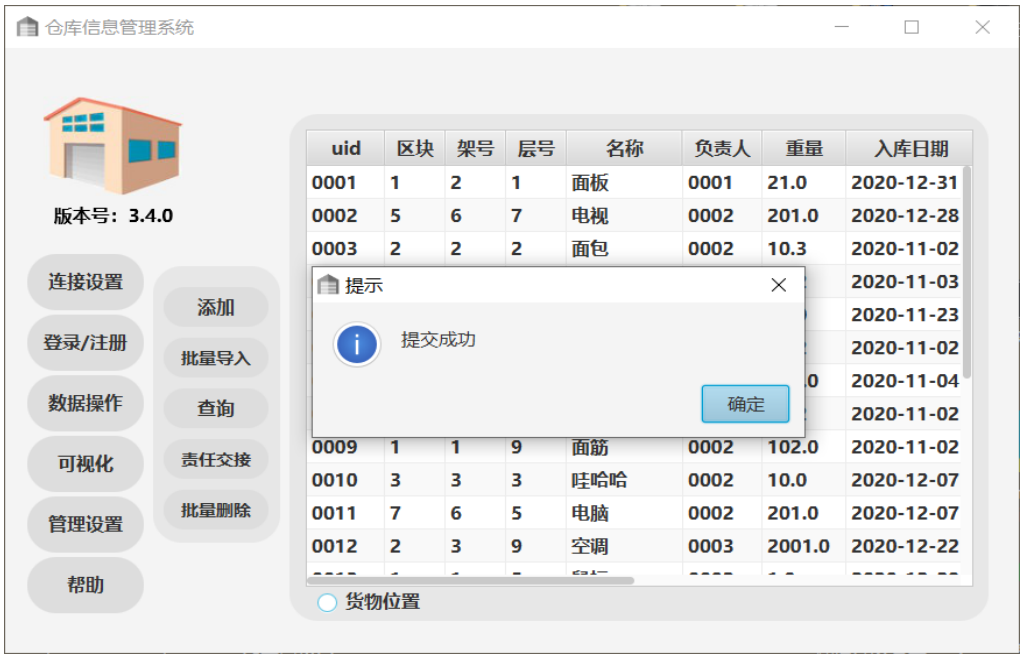

图 4.12 提交成功

4.3.3 入库信息统计图界面 CargoChart.fxml

此界面功能唯一，就是将单日入库的个数以条形图的形式显示，通过点击可视化面板中的“查看入库信息统计图”按钮打开。

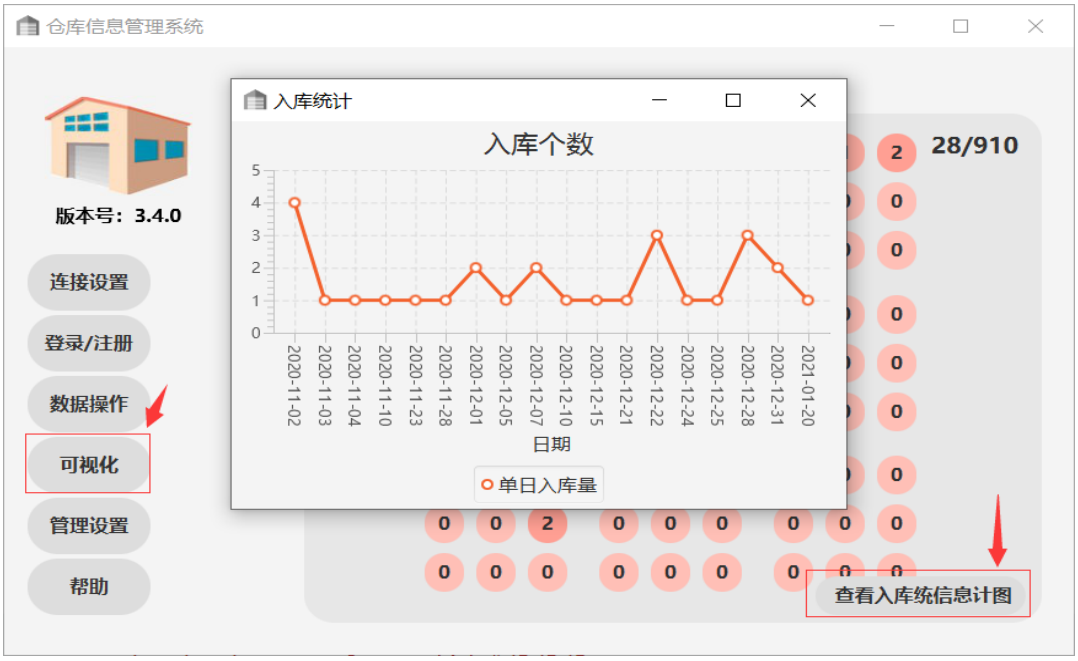

图 4.13 入库信息统计图界面

4.4 主要业务的实现思路与方法

表 4.2 部分方法与控件的命名规则

功能与作用	命名
按钮控件	Btn 结束
输入框控件	Field 结束
界面控件	Pane 结束
鼠标单击事件	onclick 开头
初始化方法	init 开头
打开界面操作	open 开头

4.4.1 连接设置

主要思路：系统要同时适配 SQL 和 mysql，并且在未来服务器的地址发生变化时也要做到不修改代码的情况下可以快速连接。用户在开启系统后，需要在连接设置的面板选择数据库的类型，随后，系统开放相应的所需的输入框，用户在输入完后单击“保存”以在 SQL 类中保存已经填写的属性值。

```
/*保存当前连接数据的响应事件*/
public void onclickSaveConnectionInfoBtn(ActionEvent event) {
    /*属性字段*/
    if (chooseServiceTypeBox.getValue() == null) {
        MyAlert alert = new MyAlert(AlertType.WARNING, "请选择服务类型");
        return;
    }

    //调用SQL.setConnection 静态方法设置连接信息
    if (chooseServiceTypeBox.getValue().equals("mysql")) {
        /*获得输入信息*/
        SQL.setConnection("mysql", null, null, serverUserName, serverPasswd);
    }
```

```
    if (chooseServiceTypeBox.getValue().equals("SQL server")) {
        /*获得输入信息*/
        SQL.setConnection("SQL", serverURL, serverName, serverUserName, serverPasswd);
    }
}
```

需要注意的是，SQL 的语法要求在数据库名和表名之间需要".."来连接，而 mysql 则需要用"."来连接，所以在保存配置时需要为数据处理事务类的静态变量赋值，以便在之后生成合法的 sql 语句。

4.4.2 用户登录

在本系统中，MyController.java 文件中有一个私有变量，它用来存储当前登录的用户，当无人登陆时为 null 值。

```
private User curUser = null;        //用于存储当前登录的用户的信息
```

主要思路：用户登录的主要流程可以分解为如图:

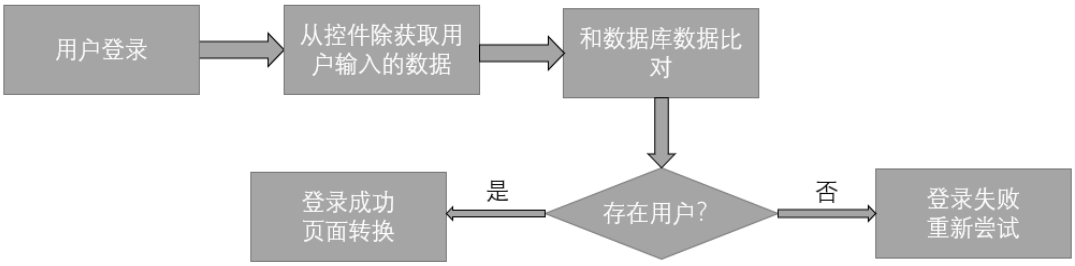

图 4.14 用户登录流程图

```
/*登录按钮的具体实现*/
public void login() throws SQLException {
    MyAlert alert;
    String userName = idField.getText();
    String passwd = passwdField.getText();

    /*通过 SQL 类登录 并将登录成功的对象传给 curUser*/
    SQL sql = new SQL();
    curUser = sql.login(userName, passwd);
```

```
        if (curUser == null) {
            alert = new MyAlert(AlertType.ERROR, "登录失败");
            return;
        }

        //处理被冻结的用户
        if (curUser.getStatus().equals("0")) {
            alert = new MyAlert(AlertType.WARNING, "您已被冻结");
            frozeMode(true);
        }

        // 转变页面
         /*登录用户信息的陈列*/
        //删除原来填写的内容
        idField.clear();
        passwdField.clear();
    }
```

当用户登录完毕后，就将用户的信息显示在窗口中，由于同时只有一个用户登录，所以将信息窗覆盖在登录窗口之上是合理的（如图 4.15），用户在此界面可以退出登录或是修改自己的登录密码。

图 4.15 用户登信息显示面板

4.4.3 数据操作

数据的操作主要分为增删改查，主要的事务逻辑可以分解为如图：

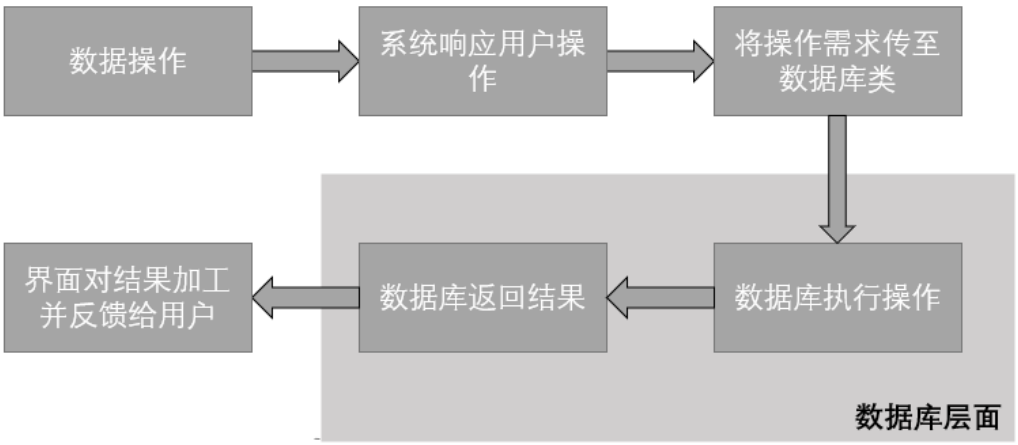

图 4.16 数据操作流程图

下面以查询功能为例，介绍思路与方法。

首先，主控制器收到了用户点击查询按钮的事件，随后进入事件的处理。程序初始化了查询对象的选择范围，随后调用封装好的模块（后文会进行介绍）生成一个对话框。在对话框中，系统收集了用户想要查找的对象和字段（图 4.17），并生成了 sql 查询语句“likestr”，随后，将其交由 SQL 类处理，并将返回的值显示在 TableView 上。

可以看到，系统与用户的交互与反馈和数据的处理与查询实现了分离，常用功能大多进行了封装。

```
/*响应点击搜索按钮的事件*/
public void onclickSearchCargoBtn(ActionEvent event) throws SQLException {
    Map<String, String> references = new HashMap<String, String>();
    references.put("uid", "uid");
    references.put("货物名", "cargoName");
    references.put("负责人", "manaInCharge");
    references.put("入库日期", "inTime");
    references.put("描述", "description");

    //调用封装好的模块
    Pair<String, String> getStr = initSearchDialog();
    if (getStr.getKey() == null) {
        return;
    }
```

```
String range = getStr.getKey();
String key = getStr.getValue();

String likeStr = "SELECT * FROM warehouseManageSysDB" + DOT + "cargoInfo" + " WHERE " + references.get(range) + " LIKE '%" + key + "%'";

//将结果显示在 TableView 中
SQL sql = new SQL();
cargoTableView.setItems(sql.getCargoData(likeStr));

}
```

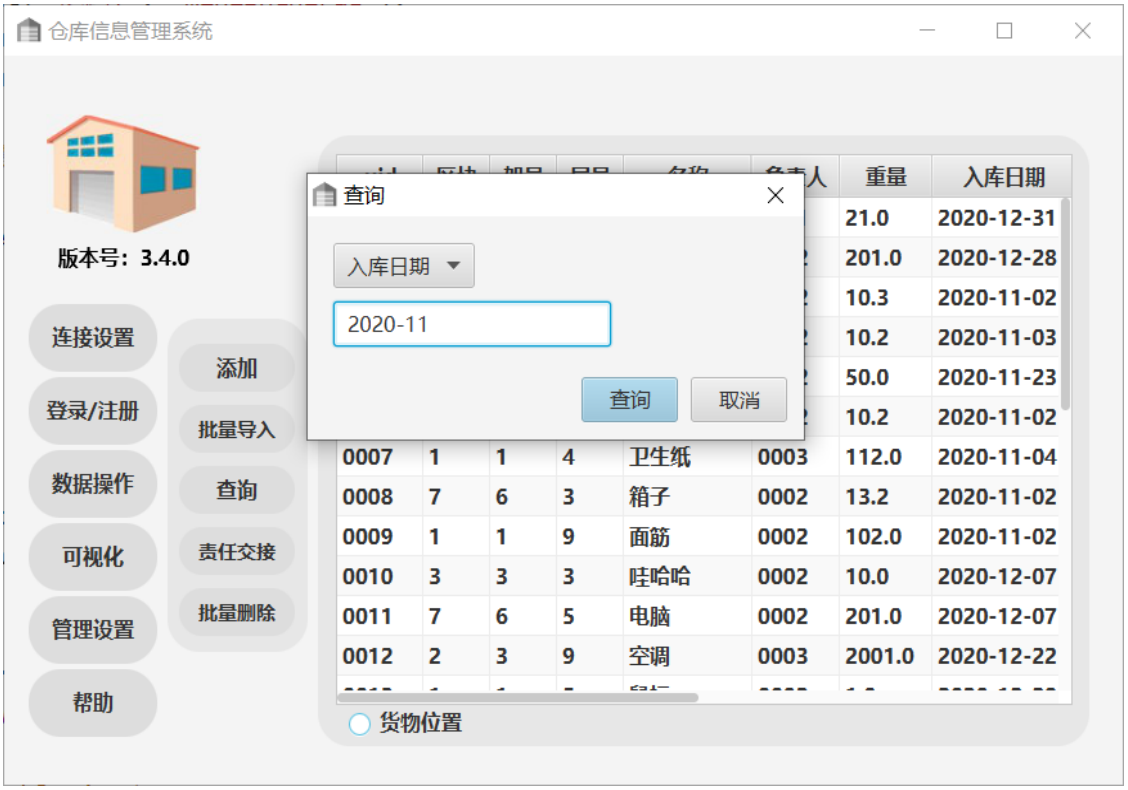

图 4.17 查询货物信息对话框

4.4.4 可视化

可视化的面板上是仓库的抽象模拟，每个货架有九层，每个货区有九个货架，整个仓库有九个货区。面板上的每个圆形代表着一个货架，他们按照 3X3 排列成货区，再按 3X3 排列成整个仓库（如图 4.18）。

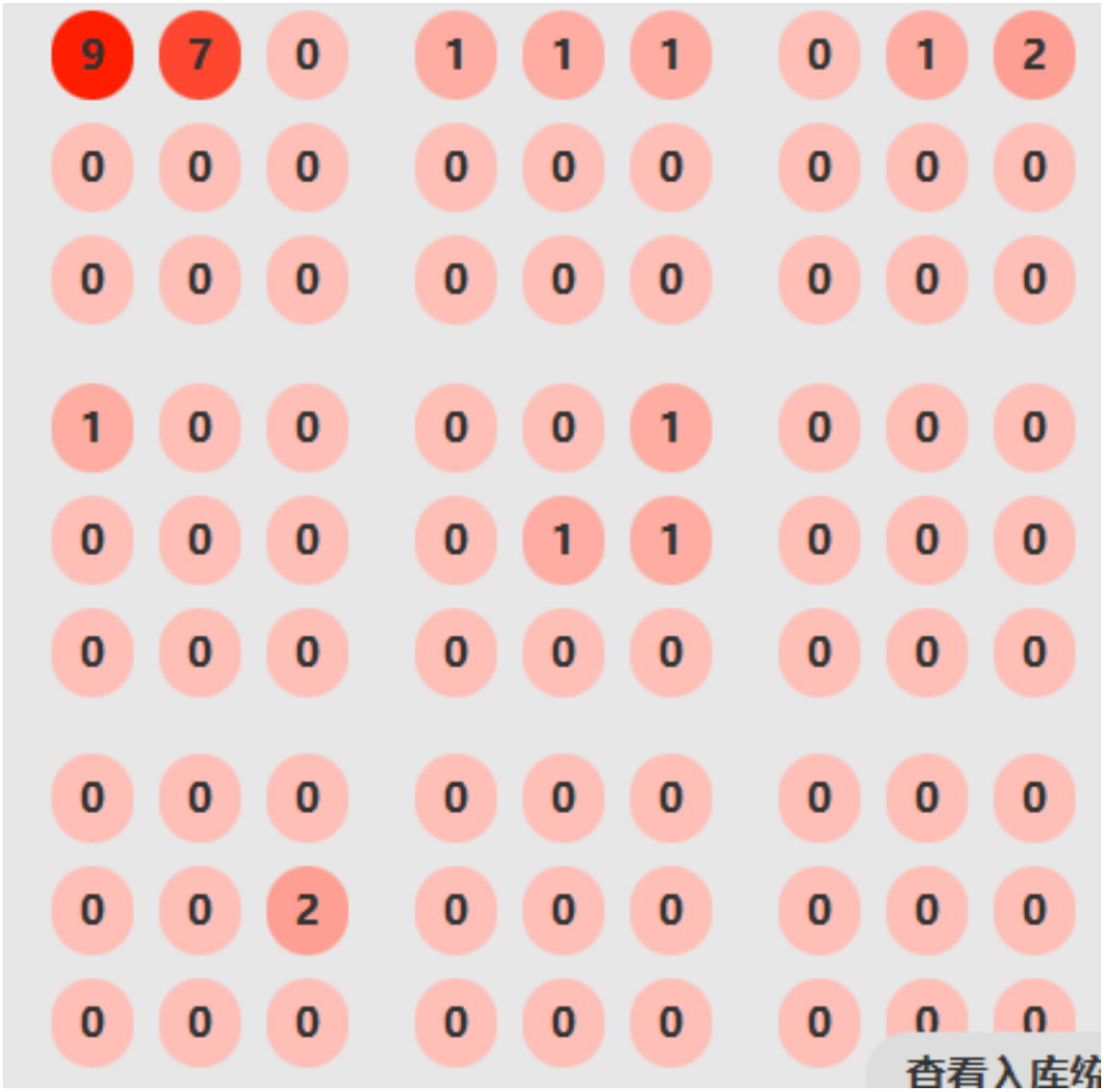

图 4.18 可视化界面

每个圆形上有一个数字，它代表着该货架上的货物个数，将鼠标悬停在其之上，可以看到该位置的货区号、货架号和存放货物总重以及存放货物的简略信息，货架的颜色越深，其存放的货物的个数就越多，用户可以直观地感受仓库的货物分布情况。右下角附有“入库信息统计图表”按钮，单击以打开图表界面。

```
/*响应打开可视化界面的事件*/
public void onclickVisualizationBtn(ActionEvent event) throws SQLException {
    hideAllPane();
    visualizationPane.setVisible(true);
    initVisualizationPane();
}
```

单击打开可视化界面按钮的事件除了将页面显示，并没有执行别的操作，是因为初始化整个界面需要较长的事件，如果等待用户想要访问后再初始化，则会让用户等待较长的时间，所以初始化的部分工作在用户成功连接数据库之后就由一个叫 MyThread 的线程完成了。由于 JavaFX 规定所有和界面相关的操作都只可由 ApplicationThread 来完成，所以 MyThread 完成的工作也比较有限，它负责收集数据库中的货物位置，并将其打包为 Map 类，以便之后被别的模块调取。

```
public class MyTread extends Thread {
    String name;
    public MyTread(String name){
        this.name = name ;        // 通过构造方法配置 name 属性
    }
    public void run(){   // 覆写 run()方法，作为线程的操作主体
        SQL sql = new SQL();
        try {
            //为 static 的 cargoPosMap 赋值
            MyController.cargoPosMap = sql.getCargoNum();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.println("THREAD over");
    }
}
```

有了 MyThread 线程的预备操作，初始化可视化面板的时间被大幅缩减了，只需要根据 Map 中存放的数据对圆形经行初始化就可以，减少了等待时间。

```
public void initVisualizationPane() throws SQLException {
        //着色板 初始化赋值略
        Map<Integer, String> colorMap = new HashMap<Integer, String>();

        /*设置 GridPane*/
        GridPane grid = visualizationGridPane;

        /*初始化模拟点*/
        for (int row = 0; row < 3; row++) {
            for (int col = 0; col < 3; col++) {
                GridPane mGrid = new GridPane();

                for (int i = 0; i < 3; i++) { // 行
                    for (int j = 0; j < 3; j++) { // 列
```

```
                    Button btn = new Button();
                    //注意行列和 block shelf 的关系
                    int b = row * 3 + col + 1;
                    int s = i * 3 + j + 1;
                    ArrayList<Integer> key=new ArrayList<Integer>();
                    key.add(b); key.add(s);
                     //没有货物的货架的属性
                    btn.setText("0");
                    btn.setStyle("-fx-background-color: " + colorMap.get(0) + ";");
                    //为有货物的货架着色并生成信息
                    if (cargoPosMap.containsKey(key)) {
                        btn.setText(""+cargoPosMap.get(key));
                        btn.setStyle("-fx-background-color: " + colorMap.get(cargoPosMap.g
et(key)) + ";");
                        //添加悬浮信息框
                        Tooltip tip = new Tooltip(sql.getBlockShelfCargoInfo(b, s));
                        btn.setTooltip(tip);
                    }
                    //大 Pane 套小 Pane
                    mGrid.add(btn, j, i);
                }
            }
            grid.add(mGrid, col, row);
        }
    }
}
```

图 4.19 悬浮信息框

4.4.5 用户管理

整个系统的功能会产生不同的后果，所以为用户设置权限等级可以使管理受到限制，系统的用户权限分为三级：

表 4.3 用户权限等级

用户类型	guest	manager	owner
权限	查看数据、查找数据	对数据的增删改查	所有（包括用户的管理权限）

在用户访问某个功能时，如果该功能需要管理员以上权限，则系统会判断用户的权限等级合法具有权限的用户才能进行进一步操作：

```
/*判断用户当前的状态与权限*/
public boolean checkPermision() {
    // 如果用户是 guest 或未登录或处于冻结状态，则返回 false
    if (curUser == null || curUser.getPermission().trim().equals("guest") ||
                                                curUser.getStatus().equals("0")) {
        return false;
    }
    return true;
}
```

对于一个用户，管理员可以对其实施“删除”、“重置密码”、“冻结/解冻”、“升级/降级权限”的操作，对于保留用户“admin”则不得做出任何操作。对于用户数据表，设置一个监听方法，用于捕获用户的双击双击操作，简单来说，就是用户在规定时间内双击了某一行，那么就会弹出针对改行实行操作的对话框，当用户选择相应操作后，进入相应的模块执行操作。

```
private Date lastClickTime;    //记录上一次的点击时间
private User uertTemp;         //记录上一次点击的行

@FXML
private void userSelect() throws SQLException {
    // 为 userTableView 创建一个点击事件反馈方法 返回值为行信息
    User row = userTableView.getSelectionModel().getSelectedItem();
    if (row == null)
        return;
    if (row != uertTemp) {
```

```
            uertTemp = row;
            lastClickTime = new Date();
        } else if (row == uertTemp) {
            Date now = new Date();
            long diff = now.getTime() - lastClickTime.getTime();
            // 设置双击间隔
            if (diff < 300) {
                /*身份认证 代码略*/

                List<String> choices = new ArrayList<>();
                choices.add("删除");
                choices.add("重置密码");
                choices.add("冻结/解冻");
                choices.add("升级/降级权限"); //仅限 owner 操作
                //得到用户选择
                String res = dialogChoose("您似乎想要进一步修改数据", choices);

                /*注：对选择的相关操作 代码略*/
                //刷新 TableView
                initUserTableView();
            } else {
                //保存上一次的点击时间
                lastClickTime = new Date();
            }
        }
}
```

针对不同的操作，执行的过程统一为：生成 sql 语句，在 SQL 类中执行语句，刷新页面。

4.4.6 其他工具类

4.4.6.1 MyAlert

MyAlert 是为了满足界面弹出大量对话框需求而封装的工具类，其主要功能类似一个工厂，程序只需指定对话框的类型（警告、信息、确认等）和对话框中的提示信息，就可以生成一个弹出对话框，而无需在每次生成弹窗时产生大量重复的代码。MyAlert 类的类声明如下：

```
public class MyAlert {
    Alert alert;        //弹窗的类型
```

```
    Optional<ButtonType> opt;      //用户的选择

    public MyAlert(AlertType A, String str){
        alert = new Alert(A);
        alert.setContentText(str);
        //设置 alert 的图标 代码略
        opt = alert.showAndWait();
    }

    //返回 Optional
    public Optional<ButtonType> getOption(){
        return opt;
    }
}
```

图 4.20 一个信息弹窗

4.4.6.2 ReadExcel

ReadExcel 的主要工作是将用户指定的 xls 类型文件读取，并对其中的数据经行分析，将表中符合要求的货物数据展示在屏幕上，是批量导入功能的重要一环。

cargoName	manaInCharge	cargoWeight	inTime	description
茶叶	0003	asd	2020-12-10	立顿
电脑包	0007	3.2	2020-12-1	沙滩款
西裤	0001	11	2020-131-16	黑色
课桌	0001	121	2020-11-10	成套 附送椅子
投影仪	0001	0.2	2020-12-22	720p
		2.1	2020-11-28	光明
公园石桌	0003	11	2020-12-5	1.5米
手机壳	0001	12.1	2020-12-7	iPhone 12 黄色

图 4.21 xls 文件中的内容

可以看到（图 4.21），在将要进行测试导入的文件中存在着一些错误的记录（用红色标出），例如重量不为数字，日期格式有误，负责人不存在等错误。在执行导入操作后，系统

会自动找出合法的信息（图 4.22）。

图 4.22 导入信息罗列

只有在经过合法校验且用户许可后，货物信息才会被录入到数据库中，这大大简化了数据的筛查过程，使导入的效率有显著的提升。

5. 工作总结与展望

通过这次实训，我又通过学习完成了一整个图形界面的开发。在学习了数据库的相关知识后，我将项目的数据存在了数据库，在后期，还结合了云数据库，实现了系统的远程访问和跨平台连接。在下定决心的那一刻起，我就只管突破问题，解决疑惑。互联网真是最好的老师，我在论坛上学到了一整套开发流程，还最终实现了自己的想法。活在当下“只要找对了工具，就不要怕实现不了想法”。在编程地过程中，我养成了好的代码习惯，并认识到了很多之前都没有发生的问题，这一定会为我将来的程序开发打下基础。而实践真正更多的是下定了我“多多实践，多多练习”的决心。

虽然程序还有许多不足，或许也有没有找到的漏洞，但是我却从中收获了许多：

① 动手写代码前要对整体有一个大的规划，而不是想到哪写到哪，混乱的逻辑会给你带来更多的麻烦；

②多多调试，只有站在使用者的角度才能发现更多的问题；

③不要怕犯错，代码的世界最不怕的就是犯错，因为你有改正的机会；

④多尝试未知，总是写着几行“祖传代码”是走不远的，要敢于尝试新东西。

程序的世界是如此精彩，写程序的人是如此乐在其中。看着自己的程序“高楼万丈平地起”，最后一步步按着预期运行，这种感觉也许就是快乐吧！希望在日后可以多多锻炼自己的动手能力和思考能力，不断完善自己的编程技术。

6. 参考文献

[1] https://baike.baidu.com/item/JavaFX/4231502?fr=aladdin 百度百科，Javafx 词条
[2]《Java面向对象程序设计》机械工业出版社，2018
[3] https://www.w3cschool.cn/java/java-inheritance.html w3cschool，Java 继承
[4] https://cloud.tencent.com/developer/ask/49427 腾讯云，JavaFX捕获键盘的Enter键入
[5] https://www.yiibai.com/javafx/javafx-tutorial-for-beginners.html 易百教程，JavaFX快速入门
[6] https://blog.csdn.net/dan1374219106/article/details/98035682 CSDN，连接数据库并获取数据库数据
[7] https://zhuanlan.zhihu.com/p/25578170 知乎，Java语法清单
[8] https://blog.csdn.net/chaiwenjun000/article/details/51967864 CSDN，Eclipse打包带mysql的java程序
[9] https://www.cnblogs.com/liuling/archive/2013/04/10/openexe.html 博客园，Java打开本地应用程序
[10] https://www.runoob.com/sql/sql-tutorial.html 菜鸟教程，SQL 教程

7. 附录

核心代码：

```
/**
 * SQL 类 处理所有数据库相关的事务
 * @author user yi
 *
 */
@SuppressWarnings("unused")
public class SQL {

    static String serverType = null;
    static String DOT = "..";
    // 针对不同登录方法的驱动配置
    static final String MYSQL_JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String SQL_JDBC_DRIVER = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    /*默认数据库的配置*/
    private static String DB_URL = "jdbc:sqlserver://localhost;DatabaseName=master";
    private static String serverURL = "localhost";
    private static String databaseName = "master";
    private static String user = "chenyi";
    private static String passwd = "xxxxx";

    /*数据库连接的属性*/
    private Connection con;
    private Statement state;
    private ResultSet rs;

        /*添加用户*/
    public void addUser(String _userName, String _passwd, String _mailAddr) throws SQLExce
ption {
        getConnection();
        //先找到合适的 id
        String uid = findValidId();

        String s = "INSERT INTO warehouseManageSysDB"+DOT+"userInfo VALUES('" + uid
+ "','" + _userName + "','" + _passwd
                + "','" + _mailAddr + "','guest',1" + ")";
        update(s);     //执行 update 的语句的方法，适用于 insert delete 等
        this.close();
    }

    /*删除用户信息*/
```

```
    public void delUser(String Uid){
        //进入次页面的用户无需再次验证身份
        String s = "DELETE FROM warehouseManageSysDB"+DOT+"userInfo WHERE Uid = '
"+Uid+"'";
        System.out.println(s);
        update(s);
        this.close();
    }

    /*找到有效的用户 id 采用遍历法*/
    public String findValidId() throws SQLException{
        int i = 1;
        String sql = "select Uid from warehouseManageSysDB"+DOT+"userInfo order by uid";
        this.getConnection();
        rs = state.executeQuery(sql);


        while (rs.next()) {
            if (int2Id(i).equalsIgnoreCase(rs.getString("Uid").trim())){
                i++;
                System.out.println(i);
            }else{

                System.out.println(int2Id(i));
                return int2Id(i);
            }
        }
        this.close();
        return int2Id(i);
    }

    /*返回货架货架的货物数量*/
    public Map<ArrayList<Integer>,Integer> getCargoNum() throws SQLException {
        Map<ArrayList<Integer>,Integer> map = new HashMap<ArrayList<Integer>,Integer>();
        this.getConnection();

        rs = state.executeQuery("SELECT * FROM warehouseManageSysDB"+DOT+"cargoInfo");

        while (rs.next()) {
            ArrayList<Integer> key = new ArrayList<Integer>();
            key.add(Integer.parseInt(rs.getString("blockno")));
            key.add(Integer.parseInt(rs.getString("shelveno")));

            if (map.containsKey(key)){
```

```
                map.put(key, map.get(key)+1);
            }else map.put(key, 1);
        }
        this.close();
        return map;
    }

    /*返回所有的货物的信息*/
    public ObservableList<Cargo> getCargoData(String str) throws SQLException {
        String sql = str;
        this.getConnection();
        if (str.equals("all")){
            sql = "select * from warehouseManageSysDB"+DOT+"cargoInfo";
        }
        ObservableList<Cargo> cargoList = FXCollections.observableArrayList();

        rs = state.executeQuery(sql);// 结果存储在 rs 结果集中，可进行后续的操作

        /* TIPS: 需要使用 SQL 语句进行查询操作，需给登录的用户（这里为 chenyi）授予 sysadmin 角色，否则请求会被拒绝 */
        while (rs.next()) {
            Cargo c = new Cargo(rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),
                    rs.getString(5),rs.getString(6),rs.getString(7),rs.getString(8),rs.getString(9));
            cargoList.add(c);
        }
        this.close();
        return cargoList;
    }

    /*返回每日的入库货物数量 用 Map 存储*/
    public Map<String, Integer> getCargoPerDay() throws SQLException{
        this.getConnection();
        Map<String, Integer> map = new LinkedHashMap<String, Integer>();
        String s = "SELECT * FROM warehouseManageSysDB"+DOT+"cargoInfo ORDER BY inTime";
        rs = state.executeQuery(s);
        while (rs.next()) {
            String date = rs.getString("inTime");
            if (map.containsKey(date)){
                map.put(date,map.get(date)+1);
            }else{
                map.put(date, 1);
            }
        }
```

```
        this.close();
        return map;
    }

    /*返回指定货架的货物存放数量*/
    public int getOccupiedLayerNumberInShelf(int block, int shelf) throws SQLException {
        this.getConnection();
        int count = 0;
        String sql = "select * from warehouseManageSysDB"+DOT+"cargoInfo "
                    + "WHERE blockno ="+block+" AND shelveno = "+shelf;
        System.out.println(sql);
        rs = state.executeQuery(sql);

        while (rs.next()) {
            count++;
        }
        this.close();
        return count;
    }

    /*获取单个货架上的货物信息汇总串*/
    public String getBlockShelfCargoInfo(int b, int s) throws SQLException{
        this.getConnection();
        String res = "区号:"+b+"  "+"货架号:"+s+"  总重:";
        String sum = "select SUM(cargoWeight) s from warehouseManageSysDB"+DOT+"cargoI
nfo WHERE blockno=" + b + " AND shelveno= "+s;
        System.out.println(sum);
        rs = state.executeQuery(sum);
        //获得总重
        while (rs.next()) {

            res += rs.getString("s")+"\n";
        }
        rs.close();


        String str = "select inTime,manaInCharge,cargoName from warehouseManageSysDB"+DO
T+"cargoInfo WHERE blockno=" + b + " AND shelveno= "+s;
        rs = state.executeQuery(str);

        while (rs.next()) {
            res += rs.getString("inTime") +"  "+ rs.getString("manaInCharge") +"  "+ rs.getStrin
g("cargoName") + "\n";
        }
```

```
        this.close();
        return res;

    }

    /*判断给定语句的结果是否为空*/
    public boolean isResultSetEmpty(String sql) throws SQLException{
        //返回 sql 语句 select 结果是否为空的逻辑值 如果空则返回 true
        boolean flag = true;
        this.getConnection();
        rs = state.executeQuery(sql);
        while (rs.next()) {
            flag = false;
        }
        this.close();
        return flag;
    }

    /*封装数据的修改（增、删、改）方法*/
    public int update(String sql) {
        int row = -1;
        this.getConnection();
        try {
            row = state.executeUpdate(sql);
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("错误: SQL 尝试失败");
        } finally {
            this.close();
        }
        this.close();
        return row;
    }

    /*将单个货物提交至数据库 需检验完合法性再调用*/
    public boolean updateCargo(Cargo cargo) {
        try {
            String uid = findValidCargoId();
            String cargoName = cargo.getCargoName().trim();
            String manager = cargo.getManage().trim();
            double weight = Double.parseDouble(cargo.getWeight());
            String inTime = cargo.getInTime().toString().trim();
            String description = cargo.getDescription().trim();
```

```
            String update = "INSERT INTO warehouseManageSysDB"+DOT+"cargoInfo"
                    + " VALUES('" +uid+ "'," + cargo.getBlock() + "," +cargo.getShelf()+","

                    + cargo.getLayer() + ",'"+cargoName + "','" + manager + "',"
                    + weight + ",'" + inTime + "','" + description + "')";

            System.out.println(update);
            update(update);
            this.close();
            return true;
        } catch (NumberFormatException e) {
            e.printStackTrace();
            return false;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally{
            this.close();
        }

    }


/**
 * 程序的主要控制器
 * 负责 1.初始化主界面的内容
 *      2.响应主界面的用户操作
 * @author user yi
 *
 */
@SuppressWarnings("unused")
public class MyController implements Initializable {
    public static String DOT = "..";    //用于适配SQL 与 mysql 的语法区别
    public static Map<ArrayList<Integer>, Integer> cargoPosMap; //当前货物单位位置量的临时存放对象
    private User curUser = null;        //用于存储当前登录的用户的信息

    /*主界面控件略*/

    /** 方法部分 **/
    /*Initializable 接口的方法实现*/
    @Override
    public void initialize(URL arg0, ResourceBundle arg1) {
```

```
        //打开软件时显示初始模式
        initMode();
        //为 passwdField 添加键盘实践监听 达到回车确认的效果
        passwdField.setOnKeyPressed(new EventHandler<KeyEvent>() {
            @Override
            public void handle(KeyEvent ke) {
                if (ke.getCode().equals(KeyCode.ENTER)) {
                    try {
                        //按下回车后执行登录操作
                        login();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        /*添加控件的悬浮提示*/
    }

    /*登录按钮的具体实现*/
    public void login() throws SQLException {
        MyAlert alert;
        String userName = idField.getText();
        String passwd = passwdField.getText();

        /*通过 SQL 类登录 并将登录成功的对象传给 curUser*/
        SQL sql = new SQL();
        curUser = sql.login(userName, passwd);
        if (curUser == null) {
            alert = new MyAlert(AlertType.ERROR, "登录失败");
            return;
        }

        System.out.println(curUser.toString());

        if (curUser.getStatus().equals("0")) {
            alert = new MyAlert(AlertType.WARNING, "您已被冻结");
            frozeMode(true);
        }

        // 转变页面

        /*登录用户信息的陈列*/
        //删除原来填写的内容
```

```
        idField.clear();
        passwdField.clear();
    }

    /*登出按钮的响应*/
    public void logout(ActionEvent event) {
        curUser = null; // 设置当前帐号为空
        hideAllPane();
        MyAlert alert = new MyAlert(AlertType.INFORMATION, "退出登录");
        frozeMode(false); //涉及到冻结模式的退出
        nonUserMode(true);
    }

    /*注册按钮点击的响应*/
    public void regit2SQL(ActionEvent event) throws SQLException {
        if (isRegistValid()) {
            MyAlert alert = new MyAlert(AlertType.INFORMATION, "注册成功");
            // 界面转换
            hideAllPane();
            loginPane.setVisible(true);

        }
    }
        /*判断用户当前的状态与权限*/
    public boolean checkPermision() {
        // 如果用户是 guest 或未登录或处于冻结状态，则返回 false
        if (curUser == null || curUser.getPermission().trim().equals("guest") || curUser.getStatus().e
quals("0")) {
            return false;
        }
        return true;
    }

    /*封装好的选择弹窗 返回用户的选择的String*/
    public String dialogChoose(String description, List<String> choices) {
        ChoiceDialog<String> dialog = new ChoiceDialog<String>("功能", choices);

        dialog.setTitle(description);
        dialog.setHeaderText(null);
        dialog.setContentText("选择您需要的功能:");
        Stage stage = (Stage) dialog.getDialogPane().getScene().getWindow();
        stage.getIcons().add(new Image("/application/icon.png"));
        Optional<String> result = dialog.showAndWait();
        if (result.isPresent()) {
```

```
            System.out.println("Your choice: " + result.get());
            return result.get();
        }
        return "cancel";
    }

    /*初始化搜索的对话框 与 initResponsibilitySwitchDialog 形似*/
    public Pair<String, String> initSearchDialog() {
        Dialog<Pair<String, String>> dialog = new Dialog<>();
        dialog.setTitle("查询");

        ButtonType confirmButtonType = new ButtonType("查询", ButtonData.OK_DONE);
        dialog.getDialogPane().getButtonTypes().addAll(confirmButtonType, ButtonType.CANCEL);

        Stage stage = (Stage) dialog.getDialogPane().getScene().getWindow();
        stage.getIcons().add(new Image("/application/icon.png"));

        GridPane grid = new GridPane();
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(20, 150, 10, 10));

        List<String> choices = new ArrayList<>();

        choices.add("货物名");
        choices.add("负责人");
        choices.add("入库日期");
        choices.add("描述");

        ChoiceBox<String> fuc = new ChoiceBox<String>();

        fuc.getItems().add("uid");
        fuc.getItems().add("货物名");
        fuc.getItems().add("负责人");
        fuc.getItems().add("入库日期");
        fuc.getItems().add("描述");

        TextField sLable = new TextField();
        sLable.setPromptText("匹配串（支持模糊查找）");

        grid.add(fuc, 1, 0);
        grid.add(sLable, 1, 1);
        dialog.getDialogPane().setContent(grid);
```

```
        dialog.setResultConverter(dialogButton -> {
            if (dialogButton == confirmButtonType) {
                return new Pair<String, String>(fuc.getValue(), sLable.getText());
            }
            return null;
        });

        Optional<Pair<String, String>> result = dialog.showAndWait();

        if (result.isPresent()) {
            return new Pair<>(fuc.getValue(), sLable.getText());
        } else {
            return null;
        }

    }

    /*初始化货物信息显示表*/
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public void initCargoTableView() throws SQLException {
        cargoTableView.setEditable(false);
        cargoTableView.getColumns().clear();

        if (checkPermision()) {
            // 该行需要在列编辑事件创立前执行，否则无法修改
            cargoTableView.setEditable(true);
        }

        TableColumn uidCol = new TableColumn("uid");
        uidCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("uid"));

        TableColumn blockCol = new TableColumn("区块");
        blockCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("block"));

        TableColumn shelfCol = new TableColumn("架号");
        shelfCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("shelf"));

        TableColumn layerCol = new TableColumn("层号");
        layerCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("layer"));

        TableColumn cargoNameCol = new TableColumn("名称");
        cargoNameCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("cargoName
"));
        cargoNameCol.setCellFactory(TextFieldTableCell.forTableColumn());
```

```
        cargoNameCol.setOnEditCommit(new EventHandler<CellEditEvent<Cargo, String>>() {
            @Override
            public void handle(CellEditEvent<Cargo, String> t) {
                //获取 uid 以便于数据库操作
                String uid = ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow
()).getUid().trim();
                String newName = t.getNewValue();

                MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "确认更改？");
                if (alert.getOption().get() != ButtonType.OK) {
                    cargoTableView.refresh();
                    return;
                }
                //更改 TableView 中的数据
                ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow()).setCargoNa
me(newName);

                String str = "UPDATE warehouseManageSysDB" + DOT + "cargoInfo" + " SE
T cargoName = '" + newName
                        + "' WHERE uid ='" + uid + "'";
                //将更新同步到数据库
                SQL sql = new SQL();
                sql.update(str);
                alert = new MyAlert(AlertType.INFORMATION, "更改成功");
                //刷新 TableView
                cargoTableView.refresh();
            }
        });

        TableColumn manageCol = new TableColumn("负责人");
        manageCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("manage"));
        manageCol.setCellFactory(TextFieldTableCell.forTableColumn());
        manageCol.setOnEditCommit(new EventHandler<CellEditEvent<Cargo, String>>() {
            @Override
            public void handle(CellEditEvent<Cargo, String> t) {

                String uid = ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow
()).getUid().trim();
                String newMana = t.getNewValue();

                MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "确认更改？");
                if (alert.getOption().get() != ButtonType.OK) {
                    cargoTableView.refresh();
                    return;
```

```
                }
                String str = "SELECT * FROM warehouseManageSysDB" + DOT + "userInfo WHERE uid ='" + newMana + "'";
                SQL sql = new SQL();
                //合法性校验
                try {
                    if (sql.isResultSetEmpty(str)) {
                        alert = new MyAlert(AlertType.ERROR, "无此负责人");
                        cargoTableView.refresh();
                        cargoTableView.refresh();
                        return;
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }

                ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow())).setCargoName(newMana);

                str = "UPDATE warehouseManageSysDB" + DOT + "cargoInfo" + " SET manaInCharge = '" + newMana
                        + "' WHERE uid ='" + uid + "'";
                sql.update(str);
                alert = new MyAlert(AlertType.INFORMATION, "更改成功");
                cargoTableView.refresh();
            }
        });

        TableColumn weightCol = new TableColumn("重量");
        weightCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("weight"));
        weightCol.setCellFactory(TextFieldTableCell.forTableColumn());
        weightCol.setOnEditCommit(new EventHandler<CellEditEvent<Cargo, String>>() {
            @Override
            public void handle(CellEditEvent<Cargo, String> t) {

                String uid = ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow())).getUid().trim();
                String newWeight = t.getNewValue();

                MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "确认更改？");
                if (alert.getOption().get() != ButtonType.OK) {
                    return;
                }
                //合法性的校验
```

```
                try {
                    if (Double.parseDouble(newWeight) < 0) {
                        alert = new MyAlert(AlertType.ERROR, "输入小于 0");
                        cargoTableView.refresh();
                        return;
                    }
                } catch (NumberFormatException e) {
                    alert = new MyAlert(AlertType.ERROR, "输入有误");
                    cargoTableView.refresh();
                    return;
                }

                ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow())).setCargoName(newWeight);
                String str = "UPDATE warehouseManageSysDB" + DOT + "cargoInfo" + " SET cargoWeight = '" + newWeight
                        + "' WHERE uid ='" + uid + "'";
                SQL sql = new SQL();
                sql.update(str);
                alert = new MyAlert(AlertType.INFORMATION, "更改成功");
                cargoTableView.refresh();
            }
        });

        //此项不得更改
        TableColumn inTimeCol = new TableColumn("入库日期");
        inTimeCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("inTime"));

        TableColumn descriptionCol = new TableColumn("描述");
        descriptionCol.setCellValueFactory(new PropertyValueFactory<Cargo, String>("description"));
        descriptionCol.setCellFactory(TextFieldTableCell.forTableColumn());
        descriptionCol.setOnEditCommit(new EventHandler<CellEditEvent<Cargo, String>>() {
            @Override
            public void handle(CellEditEvent<Cargo, String> t) {

                String uid = ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow())).getUid().trim();
                String newDescription = t.getNewValue();

                MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "确认更改？");
                if (alert.getOption().get() != ButtonType.OK) {
                    cargoTableView.refresh();
                    return;
```

```
                }
                ((Cargo) t.getTableView().getItems().get(t.getTablePosition().getRow())).setCargoName(newDescription);

                String str = "UPDATE warehouseManageSysDB" + DOT + "cargoInfo" + " SET description = '" + newDescription
                        + "' WHERE uid ='" + uid + "'";
                SQL sql = new SQL();
                sql.update(str);
                alert = new MyAlert(AlertType.INFORMATION, "更改成功");
                cargoTableView.refresh();
            }
        });

        SQL sql = new SQL();
        //从数据库读取数据 并将列信息放入
        cargoTableView.setItems(sql.getCargoData("all"));
        cargoTableView.getColumns().addAll(uidCol, blockCol, shelfCol, layerCol, cargoNameCol, manageCol, weightCol,
                inTimeCol, descriptionCol);

        sql.close();
    }

    /*初始化可视化界面*/
    public void initVisualizationPane() throws SQLException {
        SQL sql = new SQL();
        //着色版
        Map<Integer, String> colorMap = new HashMap<Integer, String>();
        colorMap.put(9, "#ff1e00");
        colorMap.put(8, "#ff3419");
        colorMap.put(7, "#ff472f");
        colorMap.put(6, "#ff5741");
        colorMap.put(5, "#ff6854");
        colorMap.put(4, "#ff7c6a");
        colorMap.put(3, "#ff8c7d");
        colorMap.put(2, "#ff9f93");
        colorMap.put(1, "#ffada2");
        colorMap.put(0, "#ffbfb6");

        /*设置 GridPane*/
        GridPane grid = visualizationGridPane;

        grid.setHgap(36);
```

```
        grid.setVgap(36);
        grid.setPadding(new Insets(30, 100, 30, 100));


        Set<Integer> block = new HashSet<Integer>();
        Set<Integer> shelf = new HashSet<Integer>();
        /*计算每个货架的货物数量*/
        int sum = 0;
        for (Map.Entry<ArrayList<Integer>, Integer> entry : cargoPosMap.entrySet()) {
            System.out.println(entry.getKey() + ":" + entry.getValue());
            sum += entry.getValue();
        }

        sumLabel.setText(sum+"/910"); //显示在界面上

        /*初始化模拟点*/
        for (int row = 0; row < 3; row++) {
            for (int col = 0; col < 3; col++) {
                GridPane mGrid = new GridPane();
                mGrid.setHgap(10);
                mGrid.setVgap(10);

                for (int i = 0; i < 3; i++) { // 行
                    for (int j = 0; j < 3; j++) { // 列

                        Button btn = new Button();
                        btn.setMaxHeight(15);
                        btn.setMaxWidth(18);

                        //注意行列和block shelf 的关系
                        int b = row * 3 + col + 1;
                        int s = i * 3 + j + 1;
                        ArrayList<Integer> key = new ArrayList<Integer>();
                        key.add(b);
                        key.add(s);
                        btn.setText("0");
                        btn.setStyle("-fx-background-color: " + colorMap.get(0) + ";");

                        //为有货物的货架着色并生成信息
                        if (cargoPosMap.containsKey(key)) {
                            btn.setText(""+cargoPosMap.get(key));
                            btn.setStyle("-fx-background-color: " + colorMap.get(cargoPosMap.g
et(key)) + ";");
```

```
                    //添加悬浮信息框
                    Tooltip tip = new Tooltip(sql.getBlockShelfCargoInfo(b, s));
                    btn.setTooltip(tip);
                }

                //大 Pane 套小 Pane
                mGrid.add(btn, j, i);
            }
        }
        grid.add(mGrid, col, row);
    }
}
System.out.println("init over");
}

/*响应检查连接的事件*/
public void onclickCheckConnectionBtn(ActionEvent event) {
    SQL sql = new SQL();
    if (sql.getConnection() == null) {
        connectionIndicator.setStyle("-fx-fill:RED;");
    } else {
        MyAlert alert = new MyAlert(AlertType.INFORMATION, "连接正常");
        connectionIndicator.setStyle("-fx-fill:GREEN;");
        loginBtn.setDisable(false);

        /*
         * 创建一个线程用于获取初始化 visualizationPane 所需的数据
         * 可以大幅减少用户的等待时间 提升实验体验
         */

        MyTread t1 = new MyTread("t1");
        t1.start();

    }
}



/*响应冻结所有 guest 用户按钮点击的事件*/
public void onclickFrozeAllGuestBtn(ActionEvent event) {
    String s = "UPDATE warehouseManageSysDB" + DOT + "userInfo SET status = '0' W
HERE permission = 'guest' ";
    SQL sql = new SQL();
    sql.update(s);
```

```
    }

    /*刷新 VisualizationPane*/
    public void onclickRefreshVisualizationGridBtn(ActionEvent event) throws SQLException {
        // warning: 刷新需要一段时间
        SQL sql = new SQL();
        cargoPosMap = sql.getCargoNum();
        initVisualizationPane();
    }

    /*响应点击批量导入按钮的事件*/
    public void onclickImportExcel(ActionEvent event) throws Exception {
        if (!checkPermision()) {
            MyAlert alert = new MyAlert(AlertType.ERROR, "您无权导入");
            return;
        }
        SQL sql = new SQL();
        String str = "";
        //显示文件选择器
        FileChooserScene fileChooserScene = new FileChooserScene();
        fileChooserScene.start(new Stage());

        //将选择结果传入 ReadExcel 类的静态方法经行读取处理
        ArrayList<Cargo> cargoList = ReadExcel.readXLS(FileChooserScene.path);

        //显示有效的信息 获取用户同意
        for(Cargo cargo:cargoList) {
            if (SQL.isCargoInfoValid(cargo, true, false)){
                str += cargo.getInTime()+"  "+cargo.getManage()+"  "+cargo.getCargoName() +
"\n";
            }
        }

        MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "你确定要导入以下货物?\n"
+str);
        if (alert.getOption().get() != ButtonType.OK) return;

        for(Cargo cargo:cargoList) {
            if (SQL.isCargoInfoValid(cargo, true, false)){
                sql.updateCargo(cargo);
            }
        }
        alert = new MyAlert(AlertType.INFORMATION, "导入完成");
        initCargoTableView();
```

```
    }


    //监听的事件间隔以及临时存放类 短时间内双击同一个单元格以开启事件
    private Date lastClickTime;
    Cargo cargoTemp;
    @FXML
    private void CargoSelect() throws SQLException {
        // 为 cargoTableView 创建一个点击事件反馈方法 与 userSelect 形似
        Cargo row = cargoTableView.getSelectionModel().getSelectedItem();
        if (row == null)
            return;
        if (row != cargoTemp) {
            cargoTemp = row;
            lastClickTime = new Date();
        } else if (row == cargoTemp) {
            Date now = new Date();
            long diff = now.getTime() - lastClickTime.getTime();
            if (diff < 300) {
                if (!checkPermision()) {
                    MyAlert alert = new MyAlert(AlertType.ERROR, "您无权访问");
                    return;
                }

                /*进行相关操作*/
                List<String> choices = new ArrayList<>();
                choices.add("删除");
                choices.add("移动");
                String res = dialogChoose("您似乎想要进一步修改数据", choices);

                if (res.equals("删除")) {
                    MyAlert alert = new MyAlert(AlertType.CONFIRMATION, "你确定要删
除？\n" + row.toString());
                    if (alert.getOption().get() != ButtonType.OK)
                        return;
                    String s = "DELETE FROM warehouseManageSysDB" + DOT + "cargoInf
o WHERE uid = '" + row.getUid()
                            + "'";
                    SQL sql = new SQL();
                    sql.update(s);
                    alert = new MyAlert(AlertType.INFORMATION, "删除成功");
                    initCargoTableView();
                }
```

```
                if (res.equals("移动")) {
                    ArrayList<Integer> newPos = initChangePosDialog();
                    if (newPos == null)
                        return;
                    SQL sql = new SQL();
                    if (!sql.isResultSetEmpty("SELECT * FROM warehouseManageSysDB" + D
OT + "cargoInfo WHERE blockno="
                            + newPos.get(0) + " AND shelveno=" + newPos.get(1) + " AND
layer=" + newPos.get(2))) {
                        MyAlert alert = new MyAlert(AlertType.ERROR, "位置被占用");
                        return;
                    }
                    String str = "UPDATE warehouseManageSysDB" + DOT + "cargoInfo SE
T blockno=" + newPos.get(0)
                            + ",shelveno =" + newPos.get(1) + ",layer=" + newPos.get(2) + "
WHERE blockno="
                            + row.getBlock() + " AND shelveno =" + row.getShelf() + " AN
D layer=" + row.getLayer();
                    System.out.println(str);

                    sql.update(str);

                    MyAlert alert = new MyAlert(AlertType.INFORMATION, "修改成功");
                    initCargoTableView();
                }

                System.out.println(row.toString());
            } else {
                lastClickTime = new Date();
            }
        }
    }

}
```

系统设计