**数据库课程设计**

题 目 医院信息管理系统

目录

[一、 概述 2](#_Toc106109008)

[二、 需求分析 2](#_Toc106109009)

[（一） 数据结构 2](#_Toc106109010)

[（二） 数据流 8](#_Toc106109011)

[（三） 数据存储 9](#_Toc106109012)

[（四） 处理过程 9](#_Toc106109013)

[三、 概念结构设计 11](#_Toc106109020)

[（一） E-R图 11](#_Toc106109021)

[（二） 系统说明书 15](#_Toc106109025)

[四、 逻辑结构设计 19](#_Toc106109028)

[（一） 关系模式 19](#_Toc106109029)

[（二） 数据表的详细结构信息 19](#_Toc106109030)

[（三） 系统结构图（模块结构） 22](#_Toc106109031)

[五、 物理设计 22](#_Toc106109032)

[（一） 存储安排 22](#_Toc106109033)

[（二） 模块设计（IPO表） 23](#_Toc106109036)

[六、 数据库实施 23](#_Toc106109037)

[（一） 创建关系模式 23](#_Toc106109038)

[（二） 程序代码 27](#_Toc106109039)

[（三） 测试 33](#_Toc106109040)

[七、 系统设计相关代码 42](#_Toc106109041)

[（一） 存储过程的代码 42](#_Toc106109042)

[（二） 触发器的代码 42](#_Toc106109043)

[（三） 视图的代码 43](#_Toc106109044)

[八、 感想及心得体会 44](#_Toc106109045)

# 概述

随着社会的迅速发展，人类生活水平也在不断提高，人类也越来越重视医疗服务。做好医院各项事务的管理，像在职医生各项信息的管理，就诊病人各项信息的管理等等，是十分重要的。最初的医院管理由人力实现，而随着现代医院规模的增大，就诊人数的增多，光靠人力记录往往会出现错误。所以一个自动化，规范化的医院信息管理系统可以很好地保障医院各项事务的正常运行，维护好医疗系统的稳定。

# 需求分析

## 数据结构

**Department（科室）数据结构：**

**Department** =科室名+科室编号+医师数量+科室介绍

* 数据项：科室编号

含义说明：唯一标识每个科室

类型：字符型

长度：20

* 数据项：科室名

含义说明：科室的具体名字

类型：字符型

长度：20

* 数据项：医师数量

含义说明：科室的在职医师数量

类型：整型

长度：10

* 数据项：科室介绍

含义说明：科室的具体介绍

类型：字符型

长度：20

**Medicine（药品）数据结构：**

**Medicine** = 药品编号+药品名+药品售价+厂商+药品类型+药品进价+药品库存

* 数据项：药品编号

含义说明：唯一标识每个药品

类型：字符型

长度：20

* 数据项：药品名

含义说明：药品的具体名字

类型：字符型

长度：20

* 数据项：药品售价

含义说明：药品的销售价格

类型：浮点数型

长度：4

* 数据项：厂商

含义说明：药品的生产厂商

类型：字符型

长度：20

* 数据项：药品类型

含义说明：药品的所属类型

类型：字符型

长度：20

* 数据项：药品库存

含义说明：药品的安全等级

类型：整数型

长度：10

* 数据项：药品进价

含义说明：药品的进货价格

类型：浮点数型

长度：4

**Warehouse（仓库）数据结构：**

**warehous**e=仓库编号+位置+负责人

* 数据项：仓库编号

含义说明：唯一标识每个仓库

类型：字符型

长度：20

* 数据项：位置

含义说明：仓库的位置

类型：字符型

长度：10

* 数据项：负责人

含义说明：仓库的负责人

类型：字符型

长度：20

**Patient（病人）数据结构：**

**Patient**= 病人编号+姓名+性别+年龄+账户余额+籍贯+备注

* 数据项：病人编号

含义说明：唯一标识每个病人

类型：字符型

长度：20

* 数据项：姓名

含义说明：病人的姓名

类型：字符型

长度：20

* 数据项：性别

含义说明：病人的性别

类型：字符型

长度：4

* 数据项：年龄

含义说明：病人的年龄

类型：整数型

长度：3

* 数据项：籍贯

含义说明：病人的籍贯信息

类型：字符型

长度：20

* 数据项：账户余额

含义说明：病人的账户余额

类型：浮点数型

长度：10

* 数据项：备注

含义说明：医院对病人的特殊标记

类型：字符型

长度：10

**Doctor（医生）数据结构：**

**Doctor** = 医生编号+姓名+性别+年龄+职称 +电话号码

* 数据项：医生编号

含义说明：唯一标识每个医生

类型：字符型

长度：20

* 数据项：姓名

含义说明：医生的姓名

类型：字符型

长度：20

* 数据项：性别

含义说明：医生的性别

类型：字符型

长度：4

* 数据项：年龄

含义说明：医生的年龄

类型：整数型

长度：3

* 数据项：职称

含义说明：医生的职称

类型：字符型

长度：20

* 数据项：电话号码

含义说明：医生的电话号码

类型：字符型

长度：20

**RX（处方）数据结构：**

**RX** =处方编号+医生编号

* 数据项：处方编号

含义说明：唯一标识每个处方

类型：字符型

长度：20

* 数据项：医生编号

含义说明：开具处方的医师编号

类型：字符型

长度：20

取值范围：20位字符

**Booking form（挂号单）数据结构：**

**Booking form** =挂号单编号+挂号时间。

* 数据项：挂号单编号

含义说明：唯一标识每个挂号信息

类型：字符型

长度：20

* 数据项：挂号时间

含义说明：病人办理挂号的时间

类型：日期型

长度：8

**Payment（缴费单）数据结构：**

**Payment** =缴费单编号+处方编号+诊疗费+药物费用+总费用+缴费状态。

* 数据项：缴费单编号

含义说明：唯一标识每个缴费信息

类型：字符型

长度：20

* 数据项：处方编号

含义说明：所需缴费的处方单编号

类型：字符型

长度：20

* 数据项：诊疗费

含义说明：根据接诊医师职称而定的诊疗费用

类型：浮点型

长度：7

* 数据项：药物费用

含义说明：购买处方单上药物的费用

类型：浮点型

长度：7

* 数据项：总费用

含义说明：病人所缴纳的总费用

类型：字符型

长度：8

* 数据项：缴费状态

含义说明：病人提交缴费信息管理员未扣费前为“未缴费”;

类型：字符型

长度：10

**Store（入库记录）数据结构：**

**Store** =入库编号+仓库编号+数量。

* 数据项：入库编号

含义说明：唯一标识每个入库信息

类型：整型

长度：10

* 数据项：仓库编号

含义说明：要入库的药品存放的仓库编号。

类型：字符型

长度：20

* 数据项：数量

含义说明：药品入库数量

类型：整型

长度：10

**Deliver（出库记录）数据结构：**

**Deliver** =出库编号+仓库编号+数量。

* 数据项：出库编号

含义说明：唯一标识每个出库信息

类型：字符型

长度：20

* 数据项：仓库编号

含义说明：要出库药品存放的仓库编号。

类型：字符型

长度：20

* 数据项：数量

含义说明：药品出库数量

类型：整型

长度：10

## 数据流

表2.1 数据流表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据流名 | 数据流来源 | 数据流去向 | 组成 |
| 提交挂号请求 | 病人 | 挂号信息 | Booking form，Patient |
| 挂号信息 | 提交挂号请求 | 医生获取挂号单 | Booking form |
| 医生获取挂号单 | 病人提交挂号请求 | 医生查看病人信息 | Booking form |
| 医生查看病人信息 | 医生获取挂号单 | 医生诊断病人 | Rx |
| 医生诊断病人 | 医生查看病人信息 | 医生填写诊断信息 | Patient |
| 医生填写诊断信息 | 医生诊断病人 | 诊断信息 | Record |
| 诊断信息 | 医生填写诊断信息 | 病人获取诊断信息 | Record |
| 病人获取诊断信息 | 医生填写诊断信息 | 医生开具处方单 | Record |
| 医生开具处方单 | 医生填写诊断信息 | 处方信息 | Rx |
| 处方信息 | 医生开具处方单 | 病人获取处方 | Rx |
| 病人获取处方 | 医生开具处方 | 病人提交缴费信息 | Rx |
| 病人提交缴费信息 | 病人获取处方 | 管理员获取缴费信息 | Payment |
| 管理员获取缴费信息 | 病人提交缴费信息 | 管理员查看处方信息 | Payment |
| 管理员查看处方信息 | 管理员获取缴费信息 | 管理员查看药品信息 | Rx |
| 管理员查看药品信息 | 管理员查看处方信息 | 管理员出库药品 | Medicine |
| 管理员出库药品 | 管理员查看处方信息 | 管理员完成扣费 | Deliver，Medicine |
| 管理员完成扣费 | 管理员出库药品 | 病人信息 | Payment,Patient |

## 数据存储

表2.2 数据存储表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据存储名 | 输入的数据流 | 输出的数据流 | 组成 |
| 挂号信息 | 病人提交挂号请求 | 医生获取挂号单 | Doctor，Patient |
| 诊断信息 | 医生诊断病人 | 病人获取诊断信息 | Doctor，Patient |
| 处方信息 | 医师开具处方 | 病人获取处方 | Medicine，Rx |
| 药品信息 | 查看药品信息 | 管理员出库药品 | Medicine |
| 缴费信息 | 病人提交缴费信息 | 管理员获取缴费信息 | Medicine，Rx |

## 处理过程

## 病人提交挂号请求

处理过程： 病人提交挂号请求

说明：病人查看医师信息，根据自身需求选择医师挂号

输入：查看医师信息

输出：挂号信息

处理：病人在系统内选择科室和医师，填写相关信息提交生成挂号单。挂号单将分配给相应医师。一个病人可以挂号多个医师，一个医生可以接受多个病人的挂号。

## 医师诊断

处理过程： 医师诊治病人

说明： 医师获取病人信息后，将安排时间接诊病人

输入： 医师获取病人信息

输出： 诊断信息

处理： 医师线下诊治病人后，在系统中填写病人的确诊疾病信息，对于医师可以诊断多个病人，一个病人可以被多个医师诊断。填写好病人也可以登录系统查看这次的诊断信息。

## 开具处方

处理过程： 开具处方

说明： 医师确定病人所患疾病后，根据药品信息开具处方单

输入： 医师填写诊断信息

输出： 处方信息

处理： 医师根据现有药品和病人疾病信息开具处方单，一个医生可以开具多个处方，一个处方只由一个医师开具。一个病人可以去领取多个处方，一个处方只能需要一个病人领取。

## 病人缴费

处理过程： 病人缴费

说明： 病人拿到处方单后，在系统中提交缴费申请

输入： 病人获取处方信息

输出： 缴费信息

处理： 病人在系统中填写自己的编号和需交费的处方编号后，提交生成缴费申请，该申请将同步发送给管理员。一个病人可以提交生成多个缴费信息，但一个缴费信息只能属于一个病人。

## 管理员取药

处理过程：管理员取药(药品出库)

说明： 管理员根据处方单上的药品将药品出库

输入： 查看药品信息

输出： 管理员缴费

处理： 管理根据患者提供的药单向仓库将药品出库，仓库根据药品实际库存完成药品出库。

## 收费员缴费

办理过程： 收费员缴费

说明： 管理员取药后，对病人扣费

输入： 药品出库

输出： 病人信息

处理： 管理员根据病人提交的缴费申请的总费用，从病人账户上扣费。扣费后病人所提交申请的缴费状态由“未缴费”到“已缴费”。

# 概念结构设计

## E-R图

## 局部E-R图

1. 科室E-R图

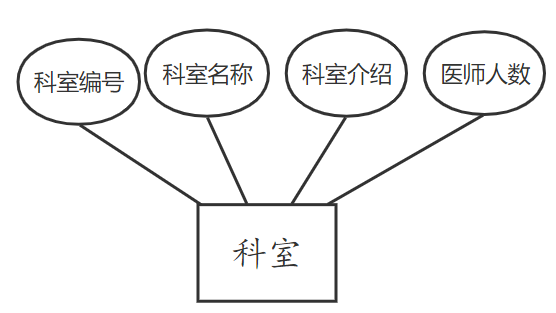


图3.1 科室E-R图

1. 医生E-R图

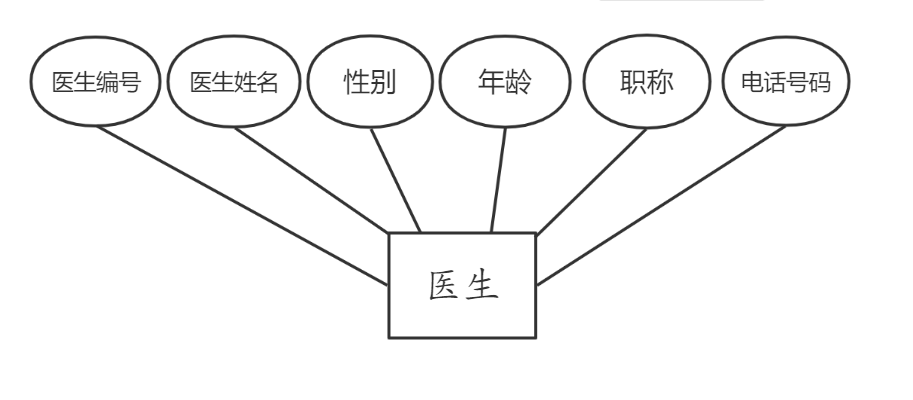


图3.2 医生局部E-R图

1. 病人E-R图

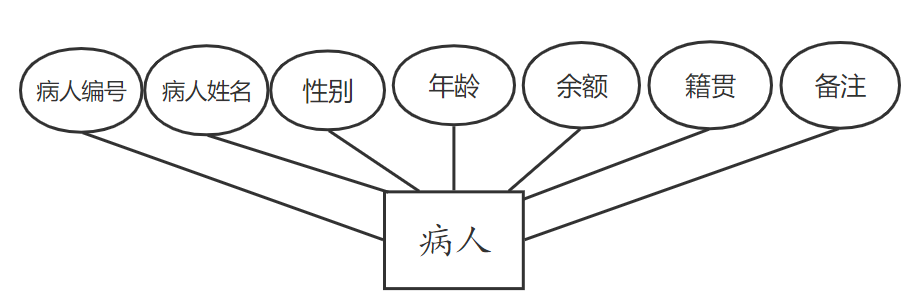


图3.3 病人局部E-R图

1. 药品E-R图

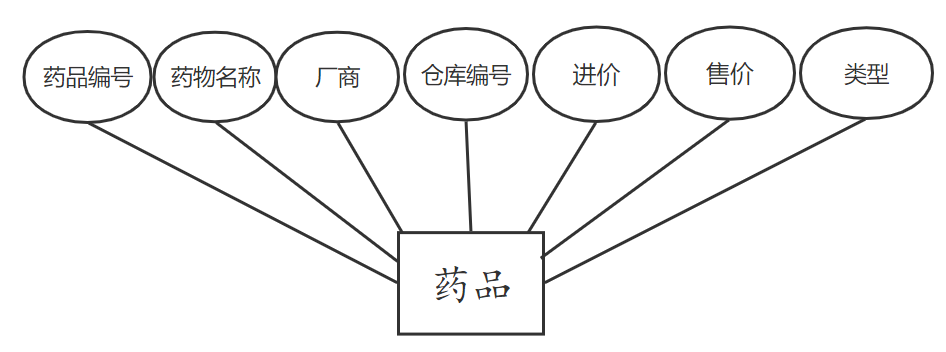


图3.4 药品局部E-R图

1. 挂号单E-R图

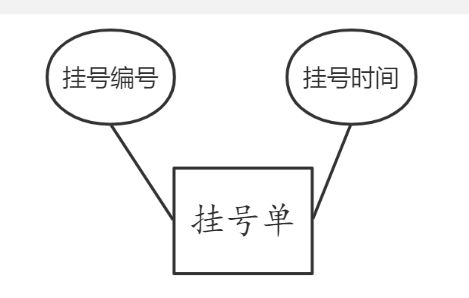


图3.5 挂号单局部E-R图

1. 处方 E-R图

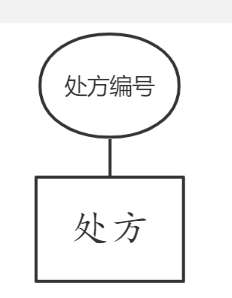


图3.6 处方局部E-R图

1. 仓库E-R图

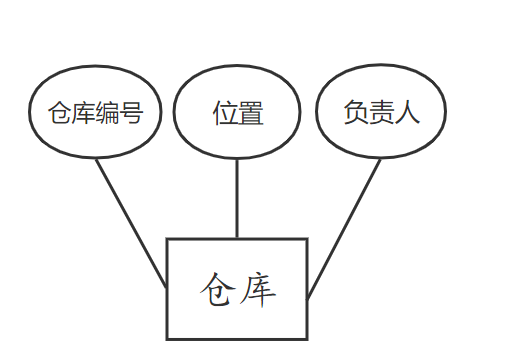


图3.7 仓库局部E-R图

1. 缴费单E-R图

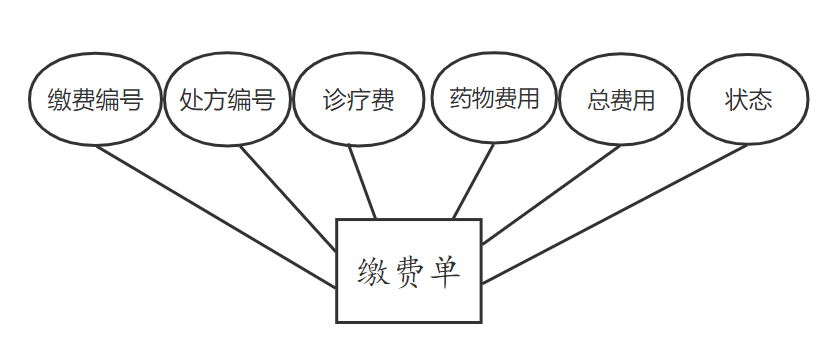


图3.8 缴费单局部E-R图

1. 入库记录E-R图

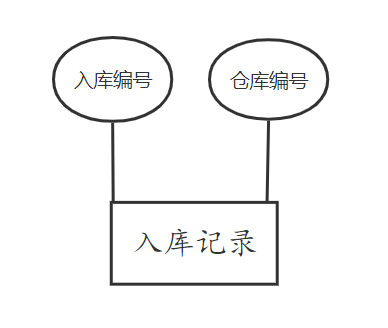


图3.9 入库记录局部E-R图

1. 出库记录E-R图

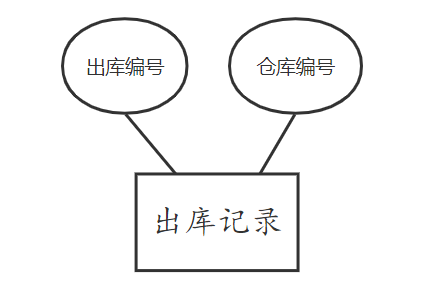


图3.10 出库记录局部E-R图

## 子系统E-R图

1. 就诊子系统

病人选择医师挂号，生成挂号单，医生获取对应挂号单；医生接诊病人后，填写诊断信息；医师开具处方，病人可以获取。

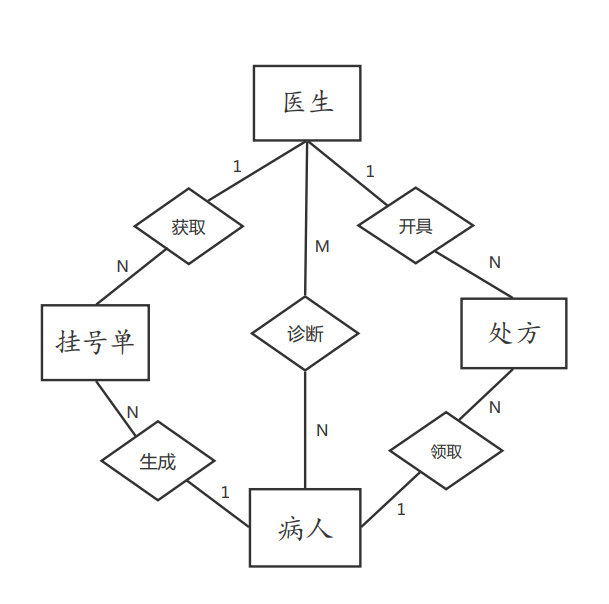


图3.11 就诊子系统

1. 缴费子系统

病人拿到处方单后，提交缴费信息，系统会根据该处方的药物信息计算费用

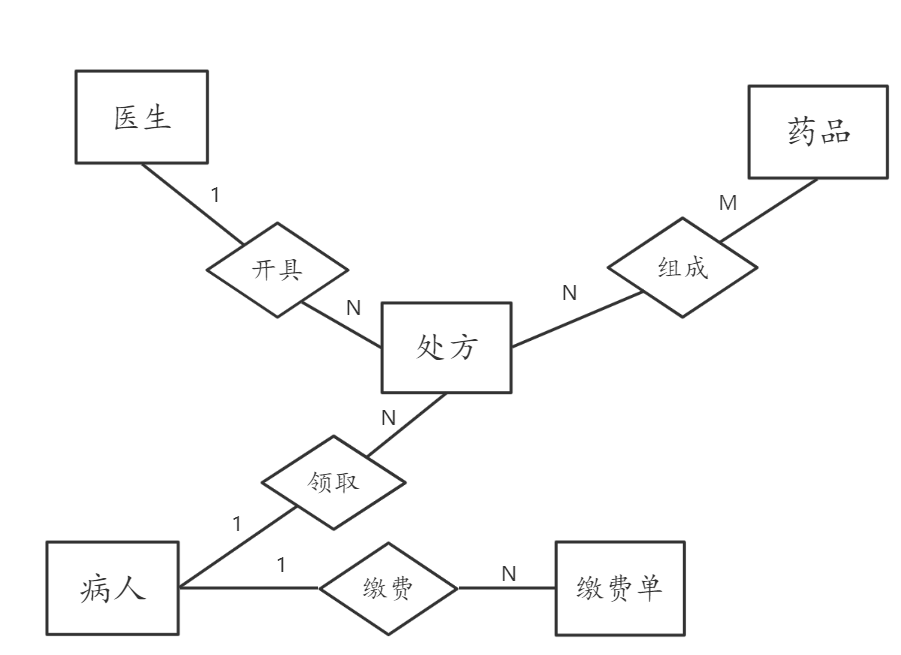


图3.12 缴费子系统

## 全局E-R图

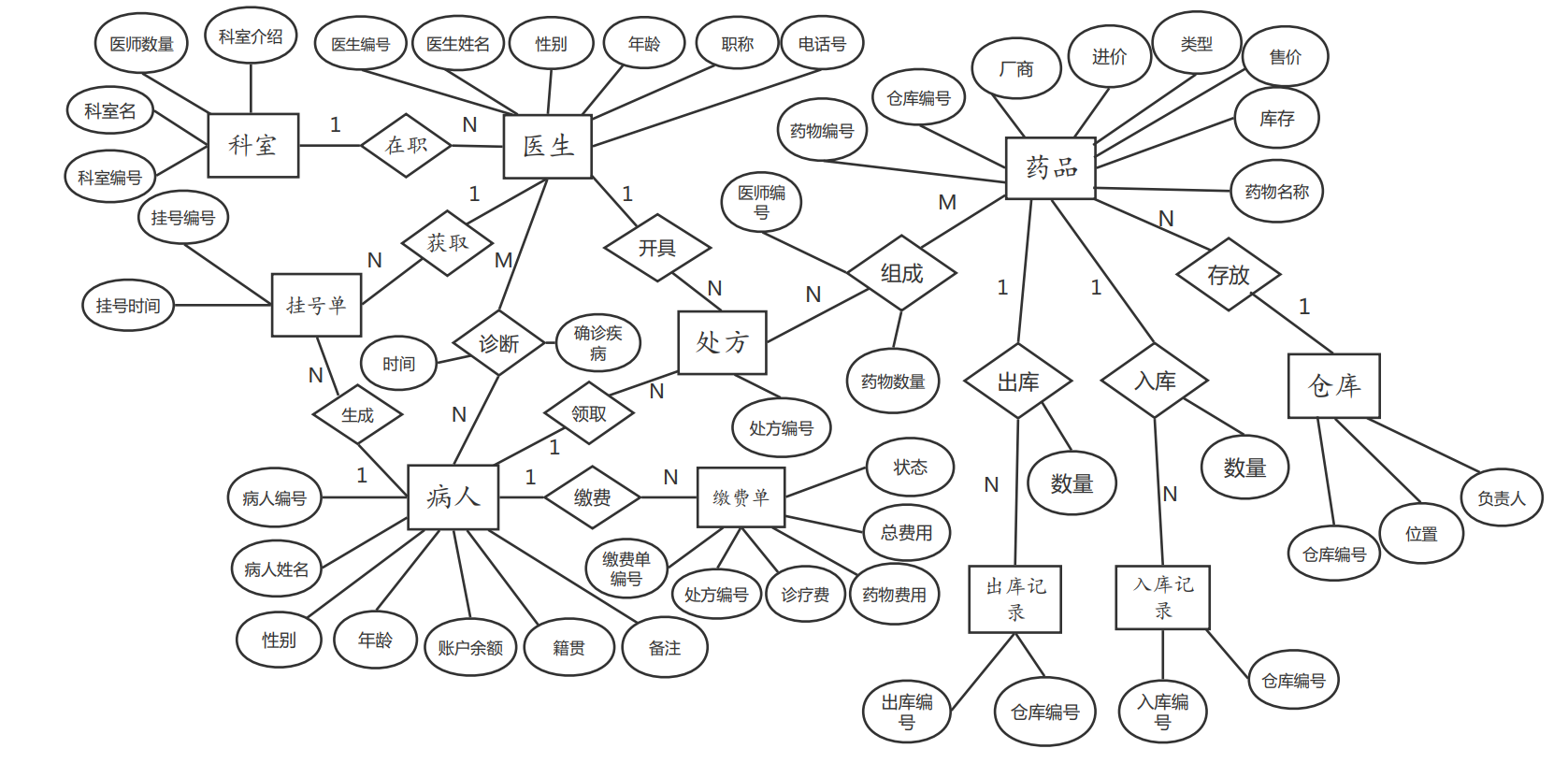


图3.13 全局E-R图

## 系统说明书

## 1、系统要求

医院信息管理系统是传统使用C/S架构的系统，本系统运行稳定，功能齐全。本系统的用户有：病人，可以查看修改个人信息，提交查看挂号申请，实现诊断管理，提交缴费申请;医生，可以查看修改个人信息，获取挂号单，填写诊断信息；管理员，实现对科室，病人，医生，药品，处方信息的管理，实现药品的出入库管理，实现对病人进行缴费。

系统的主要功能包括：

对于病人：

1. 个人信息管理：查看和编辑个人信息
2. 挂号管理：查看医师信息并选择医师提交挂号申请，查看自己的挂号记录
3. 就诊管理：查看自己的诊断信息记录，查看医师开具的处方信息
4. 缴费管理：查看自己的缴费记录，提交新的缴费信息

对于医生：

1. 个人信息管理：查看，编辑个人信息
2. 病人管理：查看自己接诊的病人信息
3. 就诊管理：查看自己诊断病人的诊断信息，填写诊断信息
4. 处方管理：查看自己开具的处方信息，添加新的处方并登记处方信息

对于管理员：

1. 医师管理：查看医院各科室医师的各项信息，添加新的医师，删除离职医师的信息
2. 病人管理：查看来医院就诊的所有病人的信息，编辑病人信息添加新的病人
3. 科室管理：查看每个科室的基本信息，查看每个科室某段时间内的就诊情况。
4. 药品管理：查看医院内每种药品的各项信息，编辑药品信息，添加新的药品，填写入库出库申请完成药品出入库。
5. 缴费管理：查看所有病人的缴费信息，对病人新提交的缴费申请予以收费。

## 数据流图

1. **顶层数据流图**

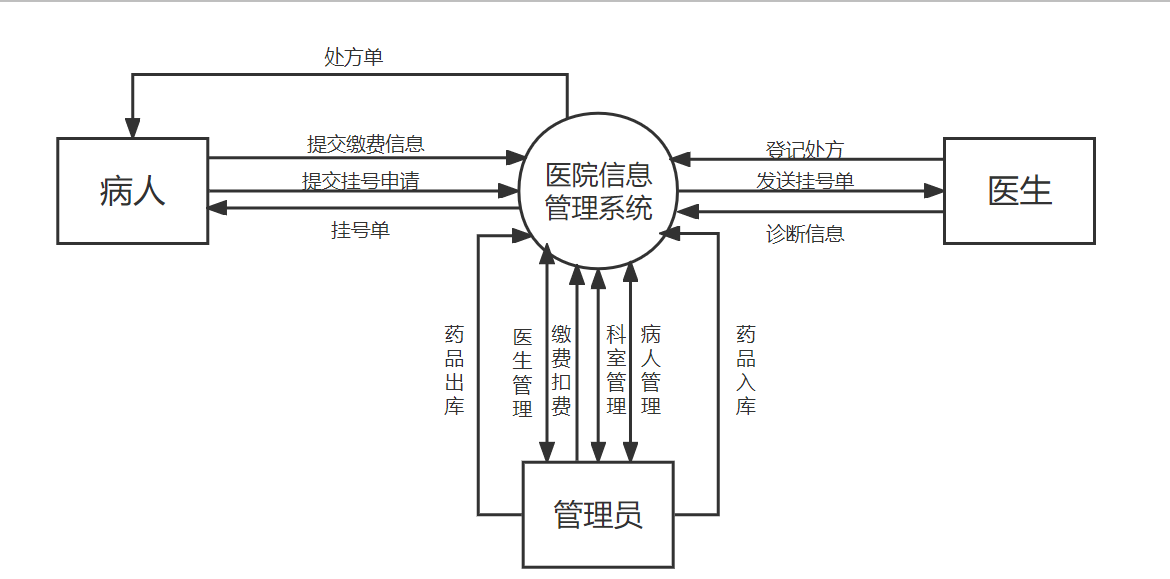


图3.14 顶层数据流图

1. **病人端数据流图**

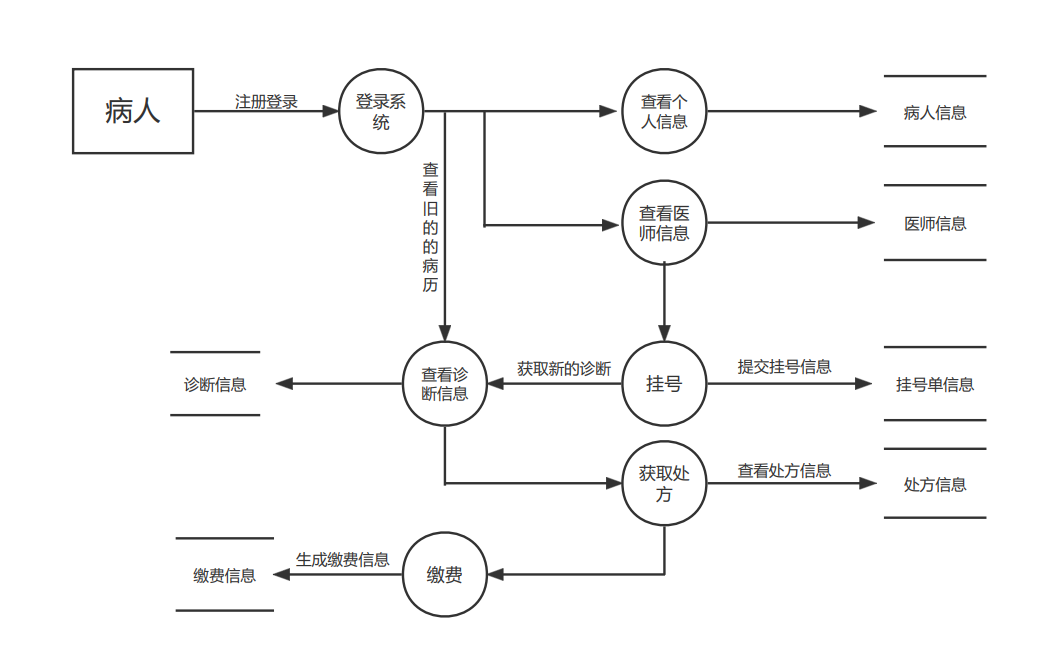
****

图3.15 病人端数据流图

1. **医生端数据流图**

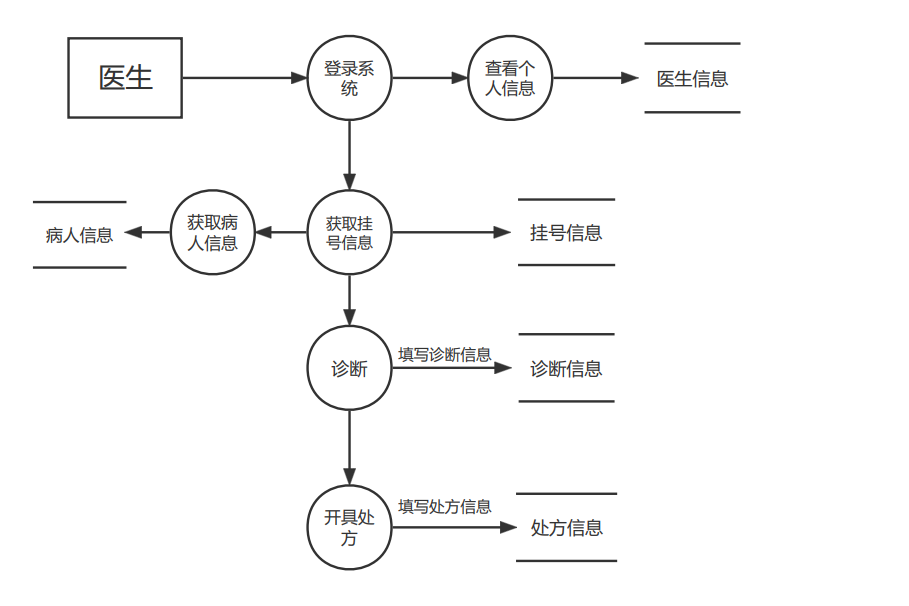
****

图3.16 医生端数据流图

1. **管理员端数据流图**

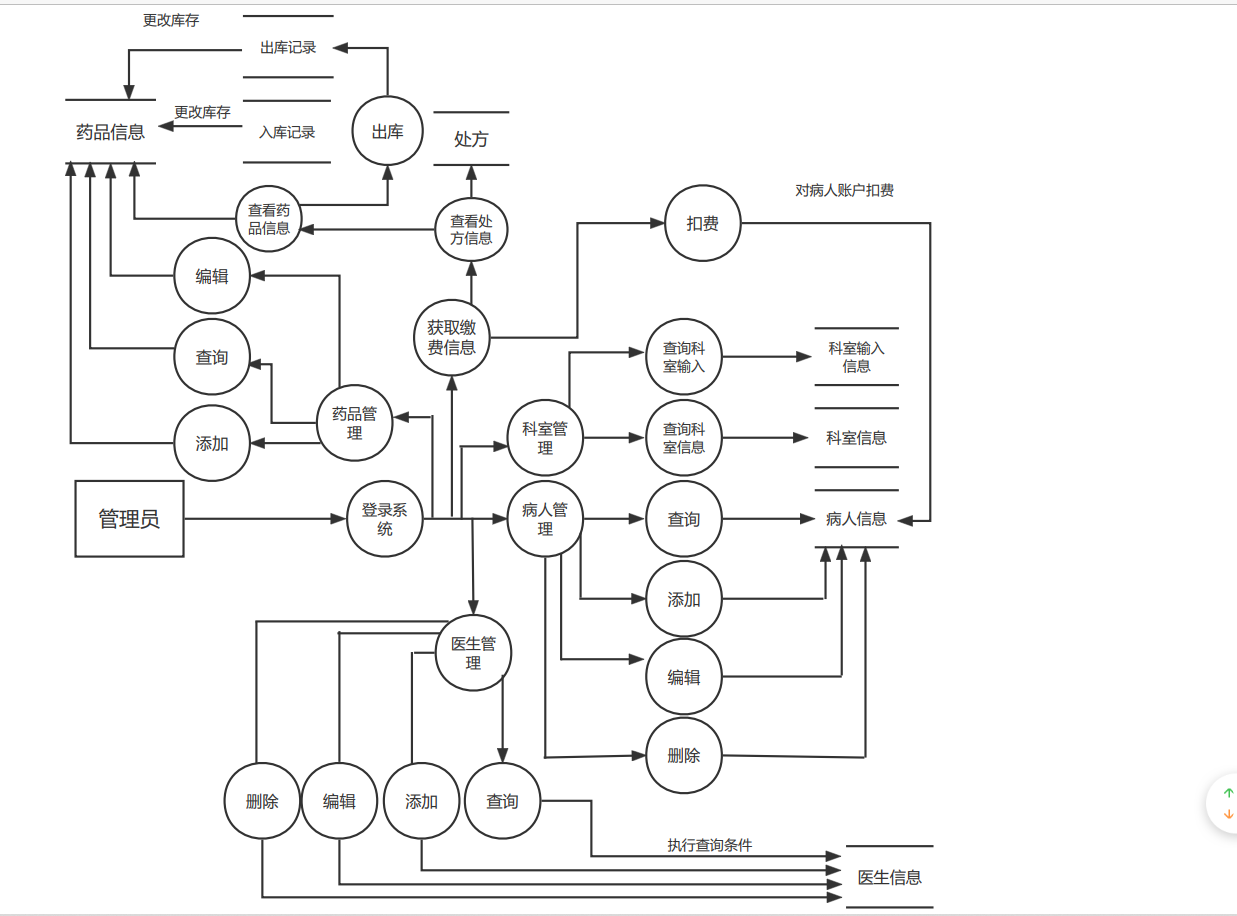
****

图3.17 管理员端数据流图

# 逻辑结构设计

## 关系模式

根据E-R图向关系模型的转换原则，医院信息管理系统的E-R图可以转换为下列关系模式：

(画横线的属性是关系的主码，画波浪线的属性是该关系的外码，画双横线的既是主码又是外码)

**1．**药品(药品编号，药品名，药品售价，厂商，药品类型，仓库编号，进价，库存数量)

**2．**仓库(仓库编号，位置，负责人）

**3．**科室(科室编号，科室名，医师数量，科室介绍）

**4．**病人(病人编号，姓名，出生年月，性别，年龄，账户余额，籍贯，备注)

**5．**医生(医生编号，医生姓名，性别，年龄，职称，电话号码，科室编号)

**7．**挂号单(挂号编号，挂号时间，医生编号，病人编号)

**8．**处方(处方编号，病人编号，医师编号)

**9．**组成(药品编号，处方编号，医生编号，药品数量

**10．**诊断(医生编号，病人编号，确诊疾病，诊断时间)

**11.** 缴费单(缴费单编号，处方编号，住院费，药品费，总费用，缴费状态，病人编号)

**12．**入库(入库编号，仓库编号，药品编号，库存数量)

**13．**出库(出库编号，仓库编号，药品编号，出库数量)

## 数据表的详细结构信息

表4.1. Department表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 科室编号 | Deid | char | 20 | 主码 |
| 科室名 | Dename | char | 20 |  |
| 医师数量 | Dsum | int | 10 |  |
| 科室介绍 | Dtext | char | 20 |  |

表4.2 Doctor表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 医生编号 | Doid | char | 20 | 主码 |
| 医生名 | Doname | char | 20 |  |
| 性别 | Dosex | char | 4 |  |
| 年龄 | Doage | int | 3 |  |
| 职称 | Dlevel | char | 20 |  |
| 电话号码 | Tel | char | 20 |  |
| 科室编号 | Deid | char | 20 | 外码 |

表4.3 Patient表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 病人编号 | Pid | char | 20 | 主码 |
| 病人名 | Pname | char | 20 |  |
| 性别 | PSex | char | 4 |  |
| 年龄 | Page | int | 3 |  |
| 账户余额 | Account | Float | 10 |  |
| 籍贯 | Place | char | 20 |  |
| 备注 | Wid | char | 10 |  |

表4.4 Medicine表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 药品编号 | Meid | char | 20 | 主码 |
| 药品名 | Mename | char | 20 |  |
| 售价 | Price | float | 4 |  |
| 产商 | [Firm](http://www.baidu.com/link?url=NT49QjTgRxtUvO9C4utrgwmBLOrHTCdR6lvDgJeLGOrS7Ve-xzvCW9KgUIuVfZb3f9sTo7SW1Og5C_2LM-o_1Xajtx7RFYf9Shp0aQ3Dfb3) | char | 20 |  |
| 类型 | Type | char | 20 |  |
| 仓库编号 | Waid | char | 20 | 外码 |
| 进价 | Price1 | float | 4 |  |
| 库存数量 | Number | int | 10 |  |

表4.5 Booking\_form表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 挂号编号 | Bookid | char | 20 | 主码 |
| 挂号时间 | Booktime | date | 8 |  |
| 医生编号 | Doid | char | 20 | 外码 |
| 病人编号 | Pid | char | 20 | 外码 |

表4.6 Warehouse表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 仓库编号 | Waid | char | 20 | 主码 |
| 位置 | Place | char | 20 |  |
| 负责人 | Principal | char | 20 |  |

表4.7 Rx表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 处方编号 | RXid | char | 20 | 主码 |
| 病人编号 | Pid | char | 20 | 外码 |
| 医生编号 | Doid | char | 20 | 外码 |

表4.8 Opener表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 处方编号 | RXid | char | 20 | 主码，外码 |
| 药品编号 | Meid | char | 20 | 主码，外码 |
| 医生编号 | Doid | char | 20 | 外码 |
| 药品数量 | Number | Int | 10 |  |

表4.9 Payment表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 缴费单编号 | Payid | char | 20 | 主码 |
| 病人编号 | Pid | char | 20 | 外码 |
| 住院费 | Cost1 | float | 7 |  |
| 药品费 | Cost2 | float | 7 |  |
| 总费用 | Cost | float | 8 |  |
| 处方编号 | RXid | char | 20 | 外码 |
| 缴费状态 | State | char | 10 |  |

表4.10 Record表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 医生编号 | Doid | char | 20 | 主码 ，外码 |
| 病人编号 | Pid | char | 20 | 主码 ，外码 |
| 确诊疾病 | Disease | char | 20 |  |
| 诊断时间 | Date | date | 8 |  |

表4.11 Store表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 入库编号 | Store\_id | int | 10 | 主码 |
| 仓库编号 | Waid | char | 20 | 外码 |
| 药品编号 | Meid | char | 20 | 外码 |
| 入库数量 | Snumber | int | 10 |  |

表4.12 Deliver表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性中文名 | 列名 | 数据类型 | 长度 | 完整性约束 |
| 出库编号 | Deliver\_id | int | 10 | 主码 |
| 仓库编号 | Waid | char | 20 | 外码 |
| 药品编号 | Meid | char | 20 | 外码 |
| 出库数量 | Dnumber | int | 10 |  |

## 系统结构图

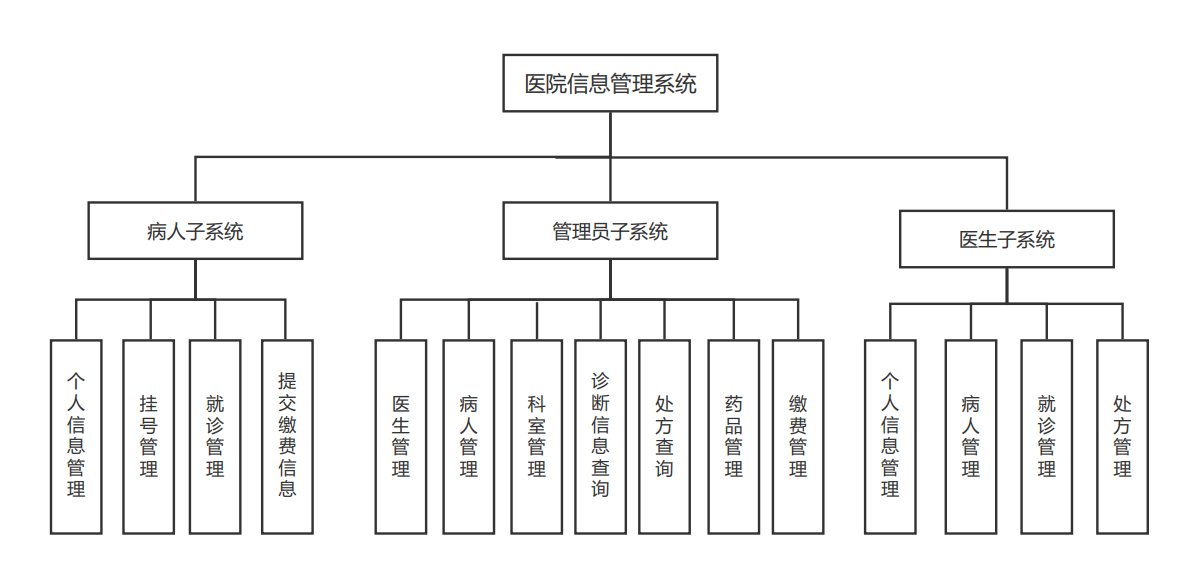


图4.1 系统结构图

# 物理设计

## 存储安排

确定数据库物理结构的内容要确定数据的存放位置和存储结构（关系、索引、聚簇、日志、备份）和确定系统配置。影响数据存放位置和存储结构的因素既包括硬件环境也包括应用需求（存取时间、存储空间利用率、维护代价）。由于此系统规模较小，所以可以将此系统放于同一磁盘上进行使用。

## 1、存取方法选择

DBMS常用存取方法有，索引方法，目前主要是B+树索引方法、聚簇（Cluster）方法和HASH方法。本系统对数据库物理设计的要求不高，采用

MYSQL 8.0 默认分配的物理空间即可满足实践要求，本系统对物理设计并未采取特别设计。

## 2、存取路径建立

数据库数据备份、日志文件备份等由于只在故障恢复时才使用，而且数据量很大，可以考虑与数据库对象（表、索引等）放在不同的磁盘以改进系统的性能。

## 模块设计（IPO表）

|  |
| --- |
| IPO表：挂号 |
| 输入：Pid，Doid  输出：Booking\_form  处理：病人登录系统后选择医师进行挂号，需要填写医师编号等信息，系统会分配给相应的医师以挂号信息 |

|  |
| --- |
| IPO表：缴费信息生成 |
| 输入：RXid |
| 输出：Payment |
| 处理：病人填写处方编号等信息，系统自动计算本次所需缴纳的费用，并在管理员端生成相应的缴费信息。 |

# 数据库实施

## 创建关系模式

（1）科室表

CREATE TABLE Department

(

Deid char(20) PRIMARY KEY,

Dename char(20)，

Dsum int(10) ,

Dtext char(20),

)

（2） 药品表

CREATE TABLE Medicine

(

Meid char(20) PRIMARY KEY,

Mename char(20) ,

Price float(4),

Frim char(20) ,

Type char(20),

Waid char(20),

Price1 float(4),

Snumber int(10),

FOREIGN KEY (Waid) REFERENCES Warehouse(Waid)

)

（3） 医生表

CREATE TABLE Doctor

(

Doid char(20) PRIMARY KEY,

Doname char(20),

Dosex char(4),

Doage Int(3),

Dlevel char(20),

Dcall char(20),

Deid char(20),

FOREIGN KEY Deid REFERENCES Department(Deid)

)

（4）病人表

CREATE TABLE Patient

(

Pid char(20) PRIMARY KEY,

Pname char(20),

Psex char(4),

Page int(3),

Account float(10),

Place char(20),

State char(10)

)

（5） 挂号单表

CREATE TABLE Booking\_form

(

Bookid char(20) PRIMARY KEY,

Booktime datetime,

Doid char(20),

Pid char(20)，

FOREIGN KEY (Doid) REFERENCES Doctor(Doid)，

FOREIGN KEY (Pid) REFERENCES Patient(Pid)

)

（6）仓库表

CREATE TABLE Warehouse

(

Waid char(20) PRIMARY KEY,

Place char(10),

Principal char(20)

)

（7）处方表

CREATE TABLE Rx

(

RXid char(20) PRIMARY KEY,

Pid char(20),

Doid char(20),

FOREIGN KEY (Doid) REFERENCES Doctor(Doid),

FOREIGN KEY (Pid) REFERENCES Patient(Pid)

)

（8）组成

CREATE TABLE Opener

(

Doid char(20),

Meid char(20),

RXid char(20),

Onumber Int(10),

PRIMARY KEY (Meid,RXid),

FOREIGN KEY (Meid) REFERENCES Medicine(Meid),

FOREIGN KEY (Doid) REFERENCES Doctor(Doid),

FOREIGN KEY (RXid) REFERENCES Rx(RXid)

)

（9）缴费单

CREATE TABLE Payment

(

Payid char(20),

Pid char(20),

Cost1 float(7),

Cost2 float(7)

Cost float(8),

RXid char(20),

State char(10),

PRIMARY KEY(Payid),

FOREIGN KEY (Pid) REFERENCES Patient(Pid),

FOREIGN KEY (Waid) REFERENCES Rx(RXid)

)

（10）出库

CREATE TABLE Deliver

(

deliver\_id int(10),

Waid char(20),

Meid char(20),

Dnumber int(10),

PRIMARY KEY(Deliver\_id),

FOREIGN KEY (Waid) REFERENCES Warehouse(Waid),

FOREIGN KEY (Meid) REFERENCES Medicine(Meid)

)

1. 入库

CREATE TABLE Store

(

store\_id int(10),

Waid char(20),

Meid char(20),

Dnumber int(10),

PRIMARY KEY(store\_id),

FOREIGN KEY (Waid) REFERENCES Warehouse(Waid),

FOREIGN KEY (Meid) REFERENCES Medicine(Meid)

)

1. 诊断

CREATE TABLE Record

(

Doid char(20),

Pid char(20),

Disease char(20),

Date datetime,

PRIMARY KEY(Doid,Pid),

FOREIGN KEY (Doid) REFERENCES Doctor(Doid),

FOREIGN KEY (Pid) REFERENCES Patient(Pid)

)

## 程序代码

**（1） 用户登录代码**

**public** **class** loginframe **extends** JFrame {

**private** JPanel contentPane;

**private** JTextField adminName;

**private** JTextField adminPd;

**private** JComboBox adminType;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

loginframe frame = **new** loginframe();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

**public** loginframe() {

setTitle("\u767B\u5F55\u9875\u9762");

setIconImage(Toolkit.*getDefaultToolkit*().getImage(loginframe.**class**.getResource("/image/\u516C\u53F8 (1).png")));

setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

setBounds(100, 100, 552, 422);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

// Image img=new ImageIcon("D:\\eclipse1\\医院信息管理系统\\src\\image\\首页.jpeg").getImage();

JLabel lblNewLabel = **new** JLabel("\u533B\u9662\u4FE1\u606F\u7BA1\u7406\u7CFB\u7EDF");

lblNewLabel.setFont(**new** Font("华文楷体", Font.***PLAIN***, 42));

lblNewLabel.setBounds(94, 27, 377, 75);

contentPane.add(lblNewLabel);

JLabel lblNewLabel\_1 = **new** JLabel("\u7528\u6237\u540D");

lblNewLabel\_1.setIcon(**new** ImageIcon(loginframe.**class**.getResource("/image/\u7528\u6237\u540D.png")));

lblNewLabel\_1.setFont(**new** Font("华文楷体", Font.***PLAIN***, 21));

lblNewLabel\_1.setBounds(70, 115, 92, 39);

contentPane.add(lblNewLabel\_1);

adminName = **new** JTextField();

adminName.setBounds(202, 122, 234, 31);

contentPane.add(adminName);

adminName.setColumns(10);

JLabel lblNewLabel\_2 = **new** JLabel("\u5BC6 \u7801");

lblNewLabel\_2.setIcon(**new** ImageIcon(loginframe.**class**.getResource("/image/\u5BC6\u7801.png")));

lblNewLabel\_2.setFont(**new** Font("华文楷体", Font.***PLAIN***, 21));

lblNewLabel\_2.setBounds(70, 186, 92, 28);

contentPane.add(lblNewLabel\_2);

adminPd = **new** JTextField();

adminPd.setBounds(202, 183, 234, 31);

contentPane.add(adminPd);

adminPd.setColumns(10);

JLabel lblNewLabel\_3 = **new** JLabel("\u7528\u6237\u7C7B\u578B");

lblNewLabel\_3.setIcon(**new** ImageIcon(loginframe.**class**.getResource("/image/\u6392\u8BFE.png")));

lblNewLabel\_3.setFont(**new** Font("华文楷体", Font.***PLAIN***, 19));

lblNewLabel\_3.setBounds(70, 240, 106, 36);

contentPane.add(lblNewLabel\_3);

adminType = **new** JComboBox();

adminType.setFont(**new** Font("华文楷体", Font.***PLAIN***, 17));

adminType.setModel(**new** DefaultComboBoxModel(**new** UserType[] {UserType.***ADMIN***,UserType.***DOCTOR***,UserType.***PATIENT***}));

adminType.setBackground(SystemColor.***controlHighlight***);

adminType.setBounds(202, 246, 151, 31);

contentPane.add(adminType);

JButton btnNewButton\_1 = **new** JButton("\u91CD\u7F6E");

btnNewButton\_1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent ae)

{

resetButton(ae);

}

});

btnNewButton\_1.setForeground(Color.***BLACK***);

btnNewButton\_1.setFont(**new** Font("华文楷体", Font.***PLAIN***, 20));

btnNewButton\_1.setBackground(SystemColor.***controlHighlight***);

btnNewButton\_1.setBounds(333, 340, 92, 23);

contentPane.add(btnNewButton\_1);

//登录

JButton btnNewButton\_1\_1 = **new** JButton("\u767B\u5F55");

btnNewButton\_1\_1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent ae)

{

confirmButton(ae);

}

});

btnNewButton\_1\_1.setForeground(Color.***BLACK***);

btnNewButton\_1\_1.setFont(**new** Font("华文楷体", Font.***PLAIN***, 20));

btnNewButton\_1\_1.setBackground(SystemColor.***controlHighlight***);

btnNewButton\_1\_1.setBounds(136, 340, 92, 23);

contentPane.add(btnNewButton\_1\_1);

btnNewButton\_1.setFocusable(**false**);

btnNewButton\_1\_1.setFocusable(**false**);

adminType.setFocusable(**false**);

setLocationRelativeTo(**null**);

}

**protected** **void** resetButton(ActionEvent ae) {

// **TODO** Auto-generated method stub

**this**.adminName.setText("");

**this**.adminPd.setText("");

}

**protected** **void** confirmButton(ActionEvent ae)

{

// **TODO** Auto-generated method stub

String name1=**this**.adminName.getText();

String password=**this**.adminPd.getText();

UserType usertype=(UserType)**this**.adminType.getSelectedItem();

**if**("管理员".equals(usertype.getName()))

{

AdminDao admindao=**new** AdminDao();

Admin admin=(Admin) admindao.selectAdmin(name1, password,0);

**if**(admin==**null**)

{

JOptionPane.*showMessageDialog*(**this**,"用户名或密码错误");

**return**;

}

**else**

{

JOptionPane.*showMessageDialog*(**this**,"登录成功");

}

Indexframe indexframe =**new** Indexframe(usertype,admin);

indexframe.setVisible(**true**);

**this**.dispose();

}

**else** **if**("医生".equals(usertype.getName()))

{

AdminDao admindao=**new** AdminDao();

Admin admin=(Admin) admindao.selectAdmin(name1, password,1);

**if**(admin==**null**)

{

JOptionPane.*showMessageDialog*(**this**,"用户名或密码错误");

**return**;

}

**else**

{

JOptionPane.*showMessageDialog*(**this**,"登录成功");

}

Indexframe\_d indexframe\_d =**new** Indexframe\_d(usertype,admin);

indexframe\_d.setVisible(**true**);

**this**.dispose();

}

**else**

{

AdminDao admindao=**new** AdminDao();

Admin admin=(Admin) admindao.selectAdmin(name1, password,2);

**if**(admin==**null**)

{

JOptionPane.*showMessageDialog*(**this**,"用户名或密码错误");

**return**;

}

**else**

{

JOptionPane.*showMessageDialog*(**this**,"登录成功");

}

Indexframe\_p indexframe\_p =**new** Indexframe\_p(usertype,admin);

indexframe\_p.setVisible(**true**);

**this**.dispose();

}

}

}

**(2) 缴费信息的生成**

**public** String addPayment(String Pid,String Doid,String Rxid)

{

String resultstr="添加失败";

String sql1="select Dlevel from doctor where Doid = '"+Doid+"'";

String sql2="select count(\*) from payment";

String sql3="select sum from pay where Rxid='"+Rxid+"'";

String sql="insert payment values(?,?,?,?,?,?,?)";

**int** sum1=0;

**double** cost1=0,cost2=0;

**try**

{

**this**.pSatement=**this**.con.prepareStatement(sql1);

ResultSet executeQuery=**this**.pSatement.executeQuery();

**while**(executeQuery.next())

{

String level=executeQuery.getString(1);

**if**("主任医师".equals(level))

{

cost1=300;

}

**else** **if**("副主任医师".equals(level))

{

cost1=200;

}

**else**

{

cost1=100;

}

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**try**

{

**this**.pSatement=**this**.con.prepareStatement(sql2);

ResultSet executeQuery=**this**.pSatement.executeQuery();

**while**(executeQuery.next())

{

sum1=executeQuery.getInt(1);

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**try**

{

**this**.pSatement=**this**.con.prepareStatement(sql3);

ResultSet executeQuery=**this**.pSatement.executeQuery();

**while**(executeQuery.next())

{

cost2=executeQuery.getInt(1);

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**try**

{

**this**.pSatement=**this**.con.prepareStatement(sql);

**this**.pSatement.setString(1, Pid);

**this**.pSatement.setString(2,String.*valueOf*(sum1+1));

**this**.pSatement.setFloat(3, (**float**) cost1);

**this**.pSatement.setFloat(4, (**float**) cost2);

**this**.pSatement.setFloat(5, (**float**) ((**float**) cost2+cost1));

**this**.pSatement.setString(6, Rxid);

**this**.pSatement.setString(7, "未缴费");

// System.out.println(this.pSatement);

**if**(**this**.pSatement.executeUpdate()>0)

{

resultstr="添加成功";

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** resultstr;

}

## 测试

1. **登录界面**

用户首先进入登录页面，输入账号，密码和身份后进入系统，通过系统验证后即可成功进入相应主页面，并弹出提示框“登录成功”，如果不能通过验证会显示“登录失败”。

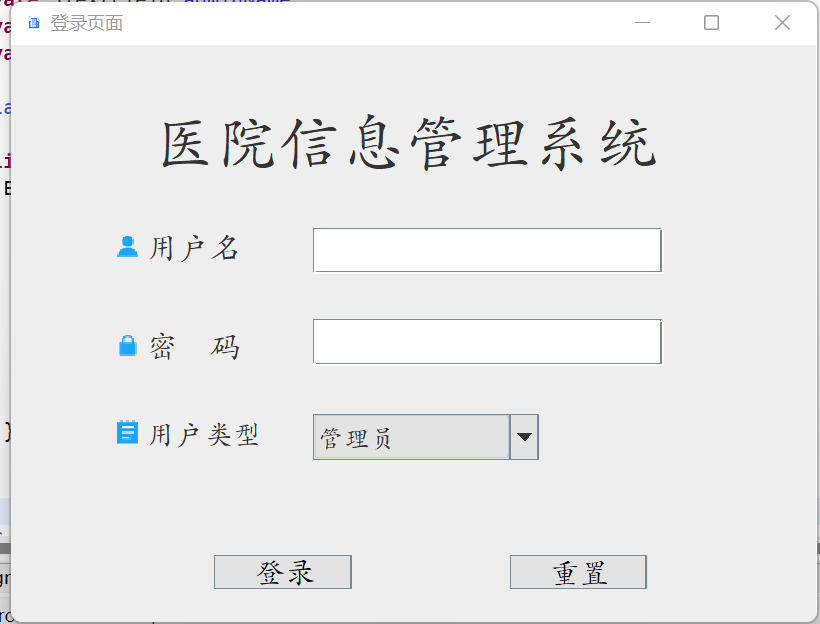


图5.1 登录主页面

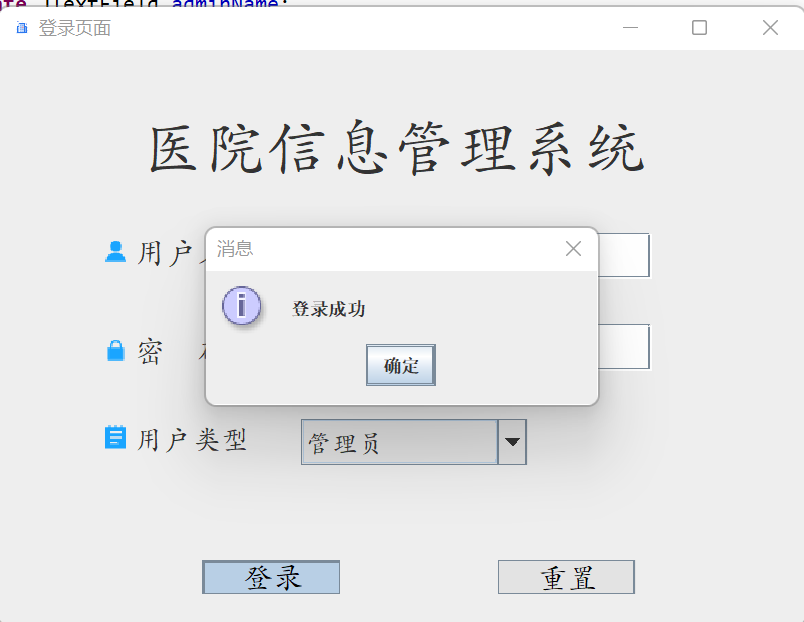


图5.2 登录成功页面

1. **主页面**

不同身份的用户进入系统后有不同的主页面，主页面也囊括了该身份用户所能使用的功能。

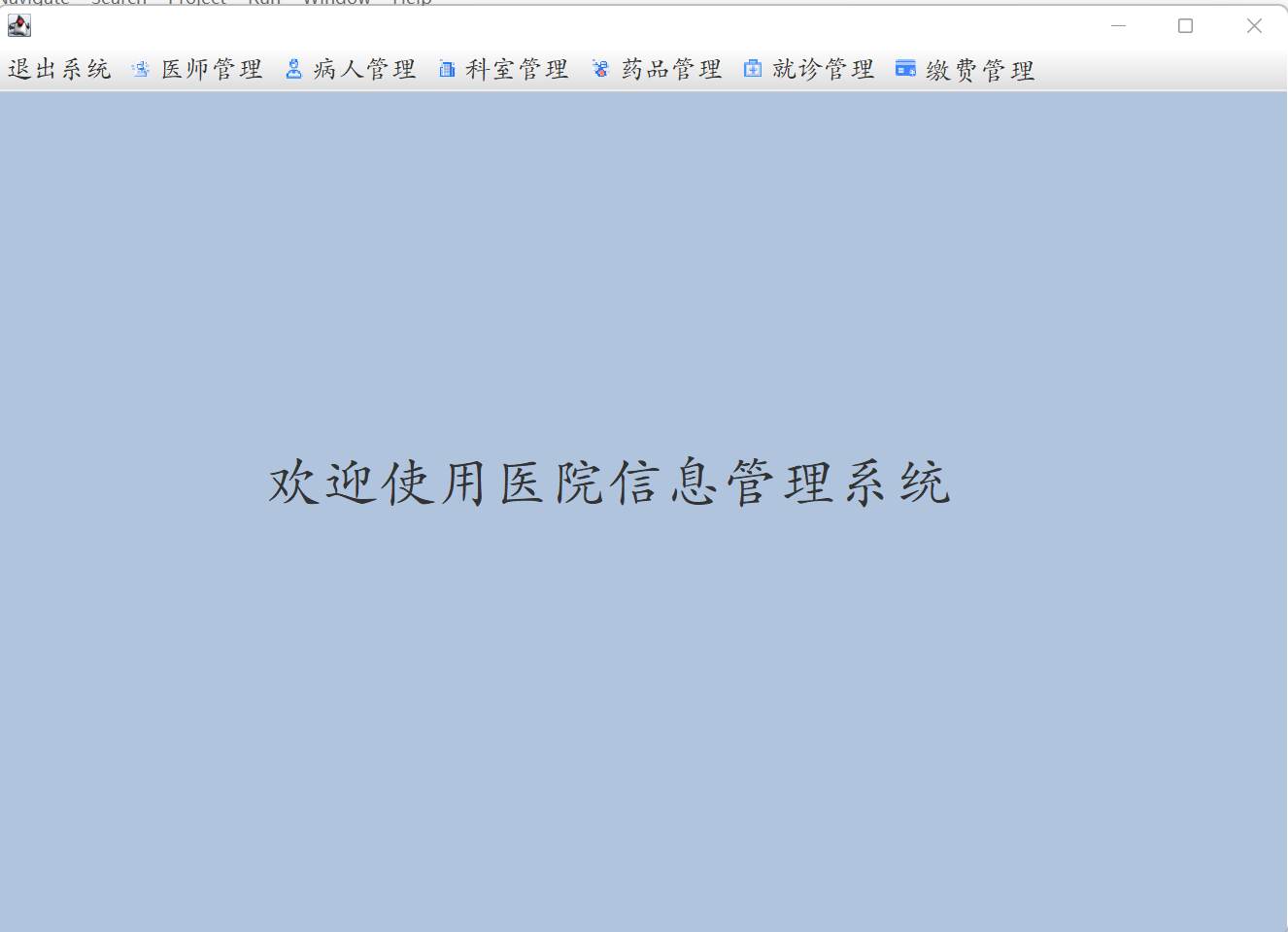


图5.3 管理员主页面

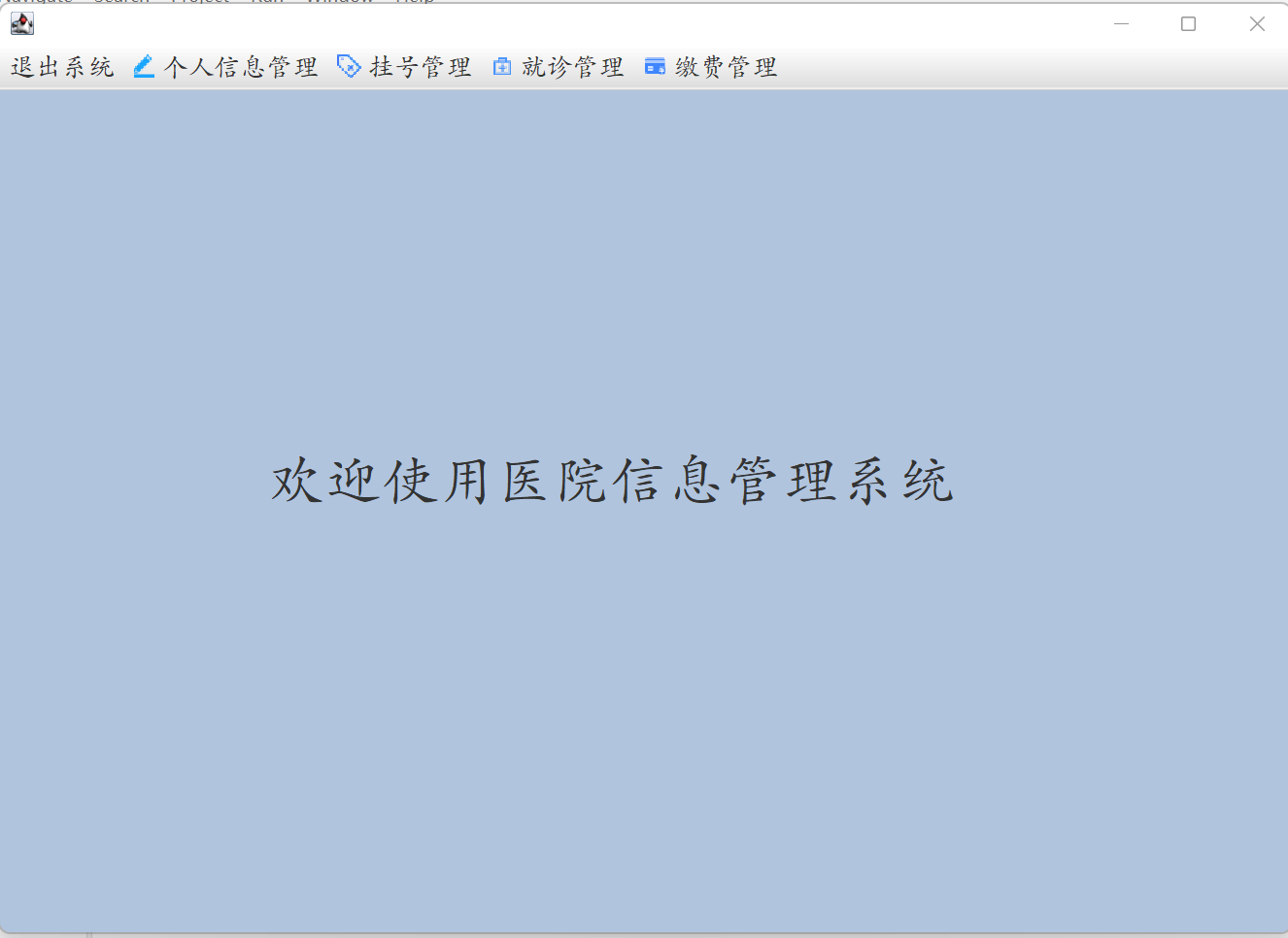


图5.4 病人主页面



图5.5 医生主页面

1. **挂号诊断**

**病人挂号**

病人点击菜单栏上的“挂号管理”，选择办理挂号，进入挂号页面，如图5.6所示



图5.6 挂号页面

病人点击挂号，填写医生编号和挂号时间后，点击确认，即可完成挂号。



图5.7 填写信息子页面

**医生诊断**

医生在收到挂号信息后，线下会安排时间对病人进行诊断，诊断后将填写诊断信息到对应诊断单，如图5.8，图5.9所示。

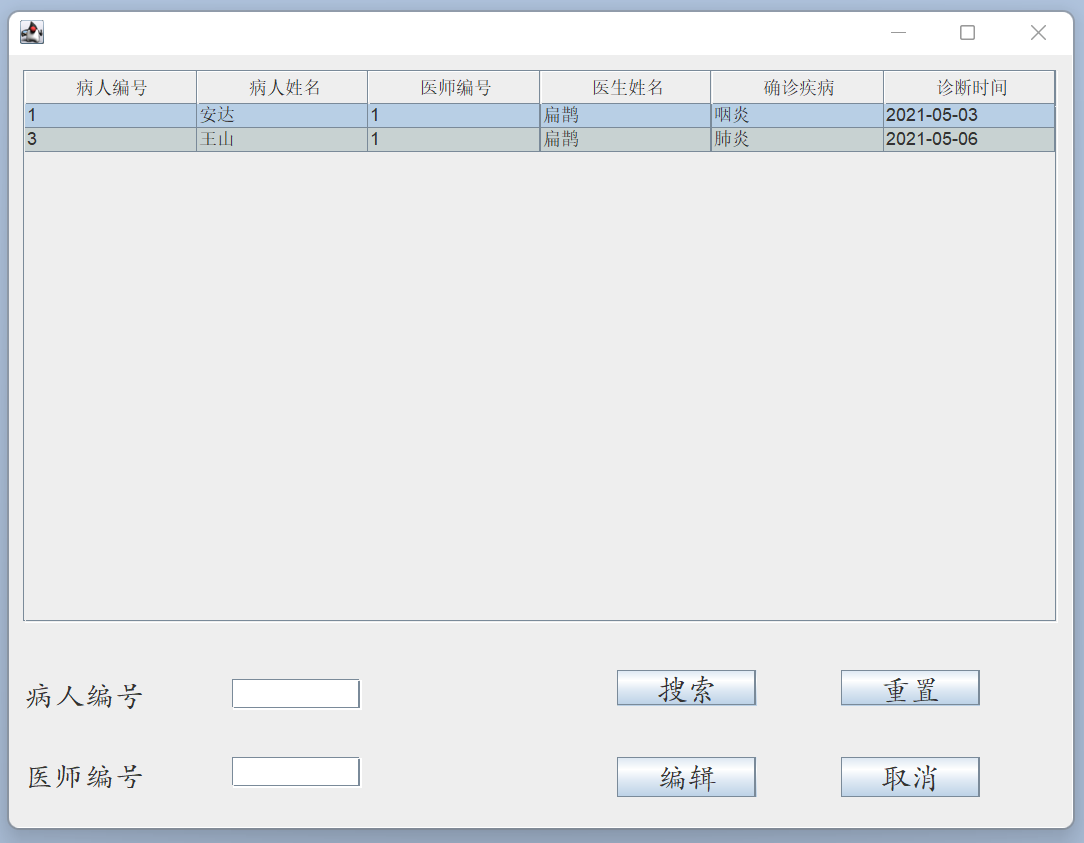


图5.8 诊断单页面

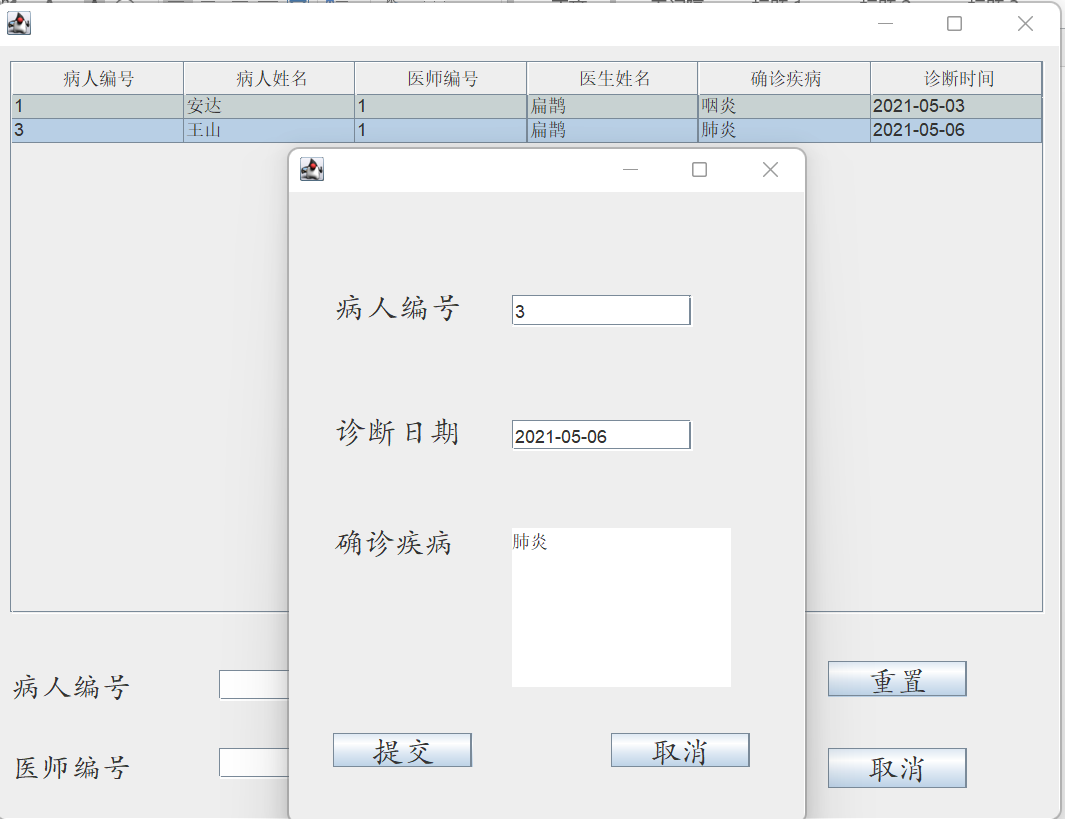


图5.9 编辑诊断信息子页面

1. **开具处方**

医生接诊病人后根据病人的确诊信息和医院现有药物开具处方，点击医生主页面的处方管理，选择登记处方，即可进入图5.10所示页面。



图5.10 处方登记主页面

点击添加处方，会弹出填写新的处方信息的页面，如图5.11所示

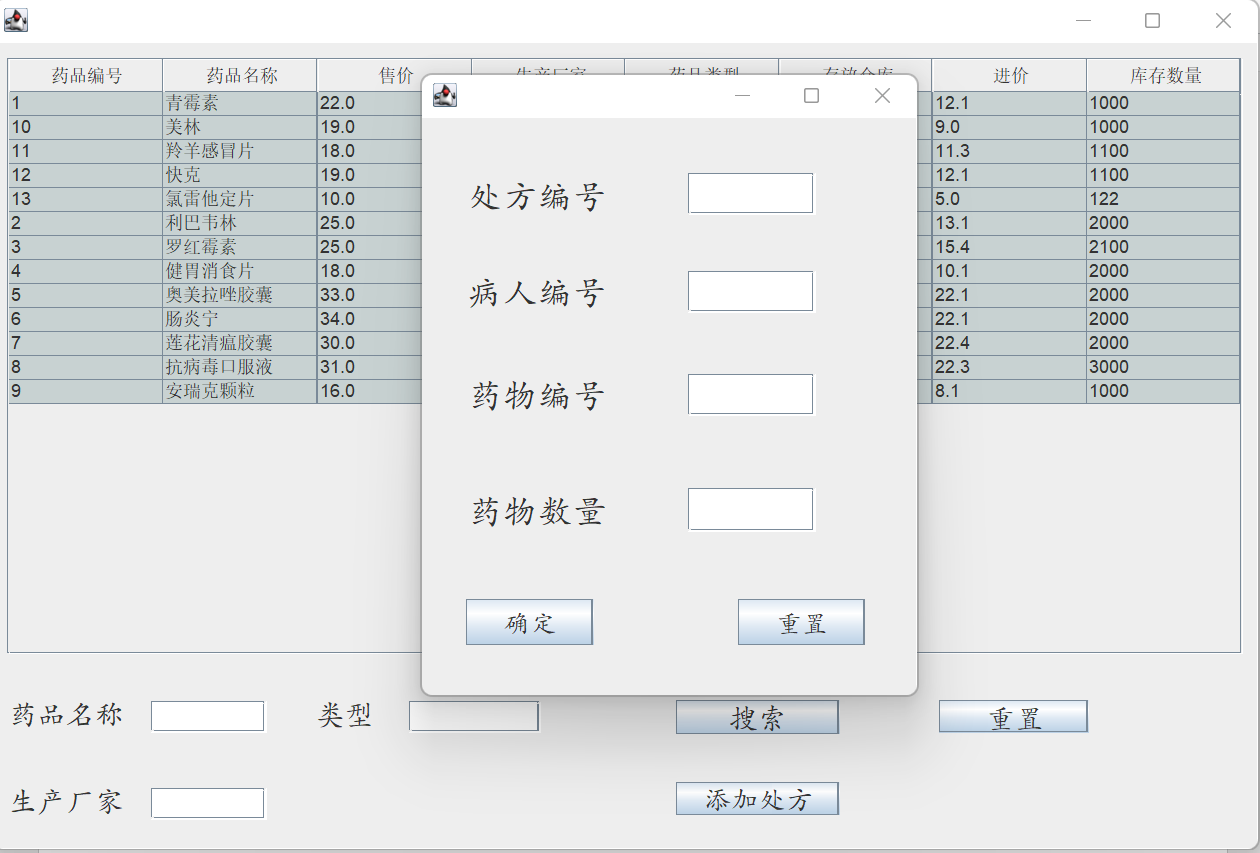


图5.11 登记子页面

1. **缴费**

**病人生成缴费信息**

点击病人主页面的缴费管理，选择办理缴费，进入缴费页面，填写相关信息，点击确认即生成新的缴费信息。

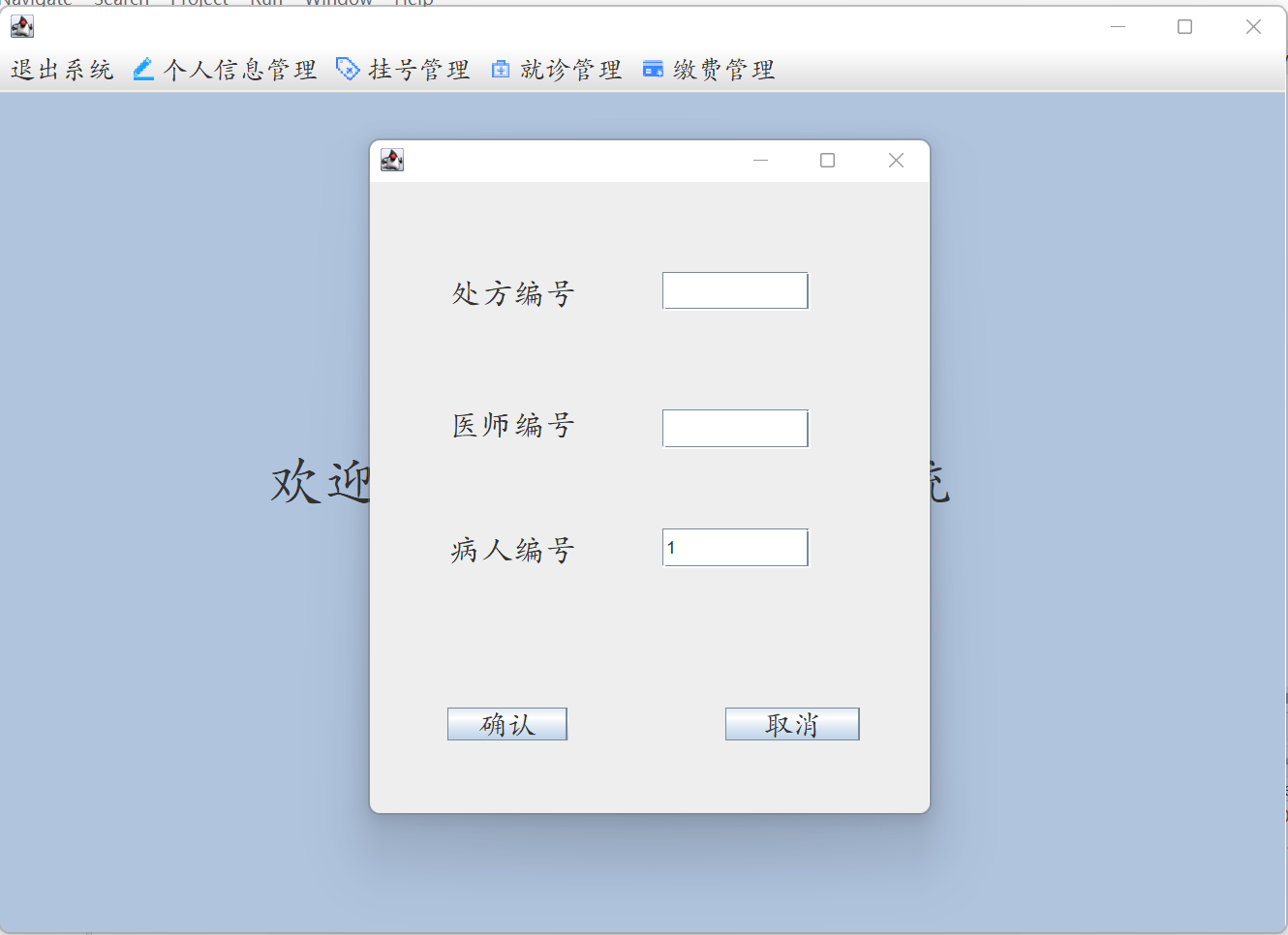


图5.12 缴费页面

**管理员收费**

管理员点击缴费管理，选择缴费信息，即可查看所有的缴费信息，选中缴费状态为“未收费”的一条信息，点击收费，如图5.13，图5.14所示。

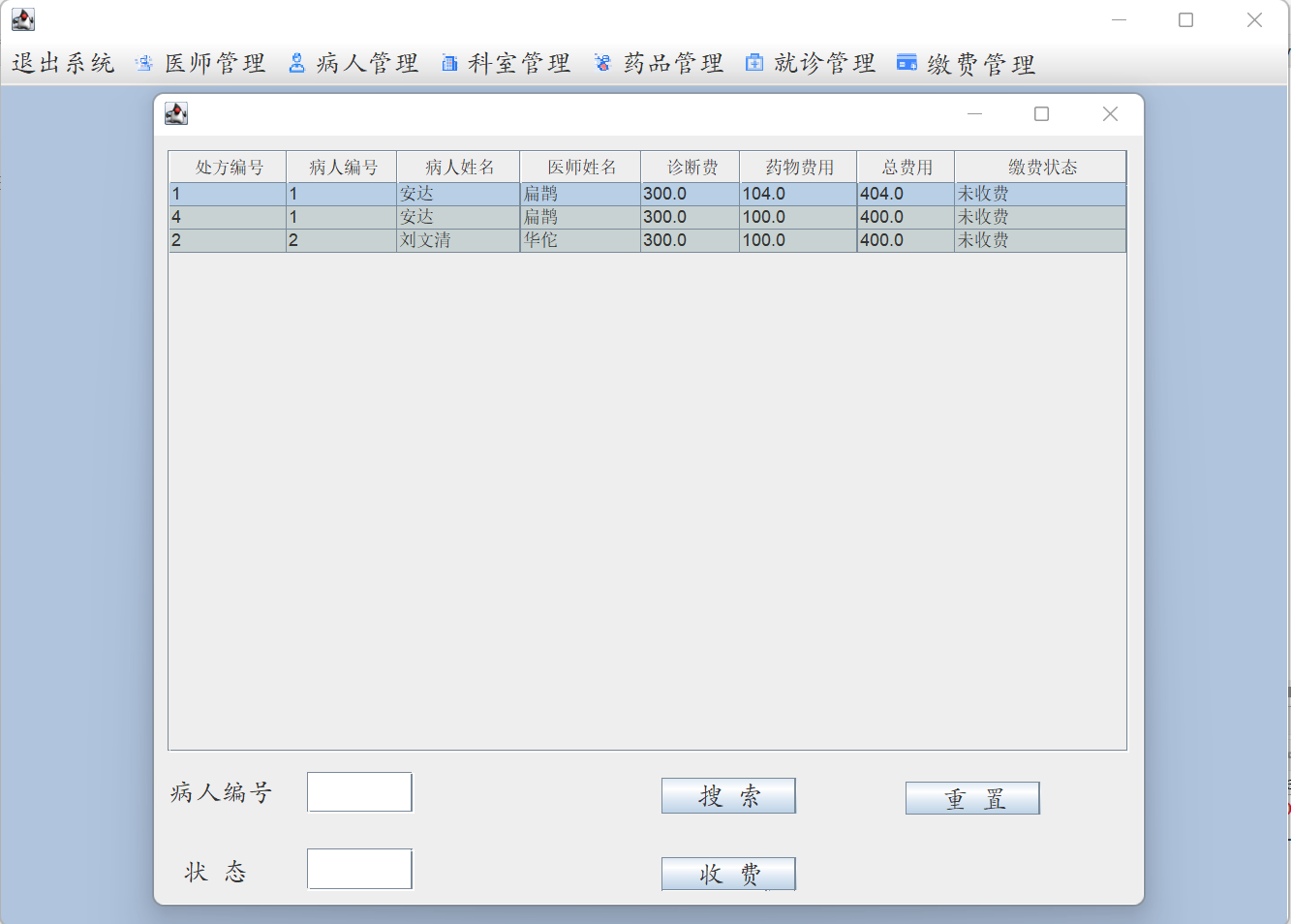
****

图5.13 收费页面

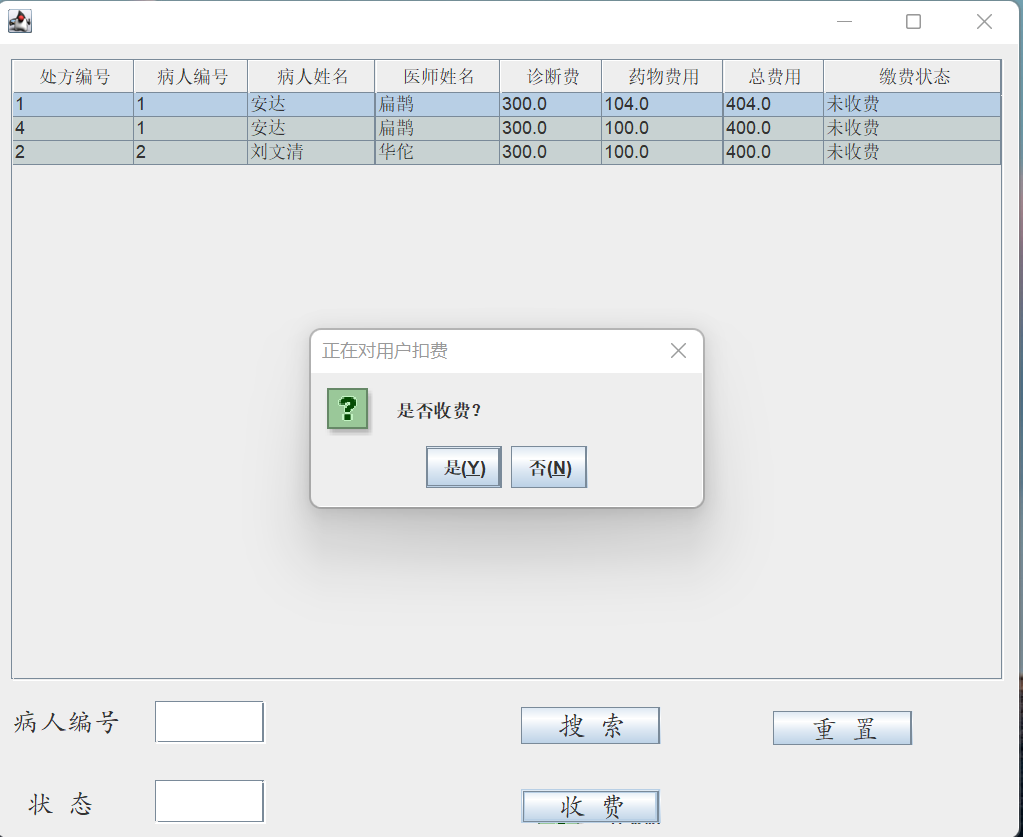


图5.14 提示框

点击收费后，系统会自动对病人的账户进行扣费，如果病人账户的余额能够支付本次费用，会弹出提示框，显示“收费成功”，反之，会显示“余额不足”

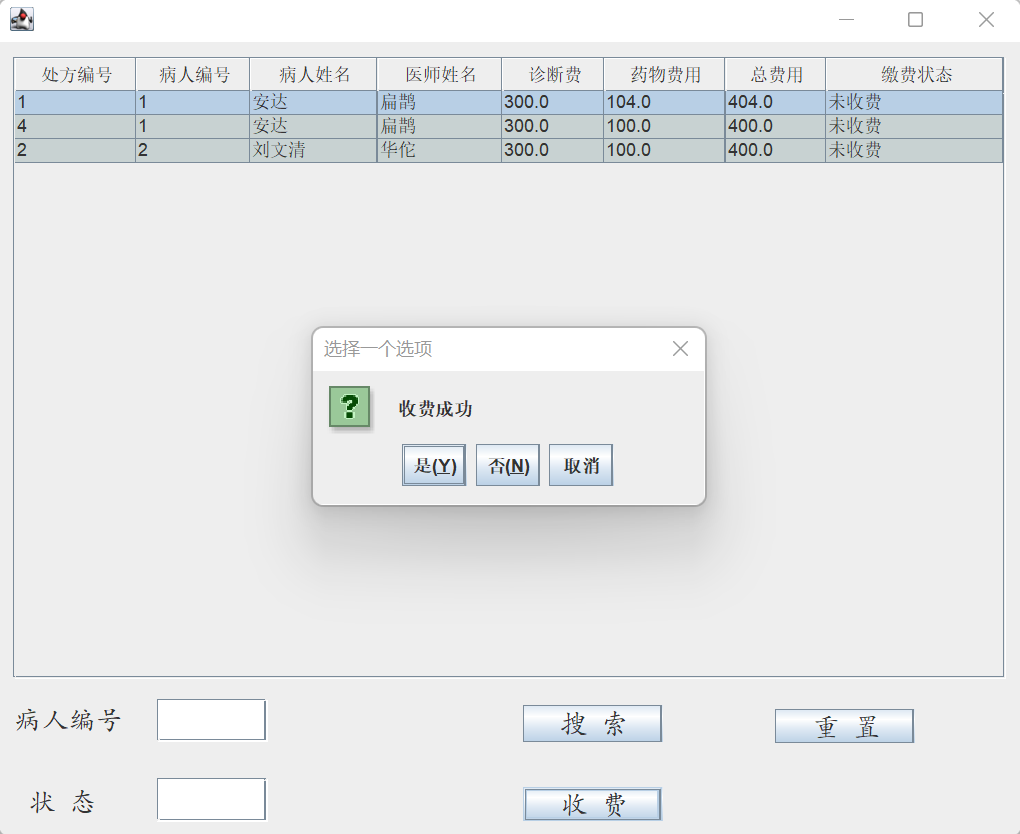


图5.15 收费成功提示框

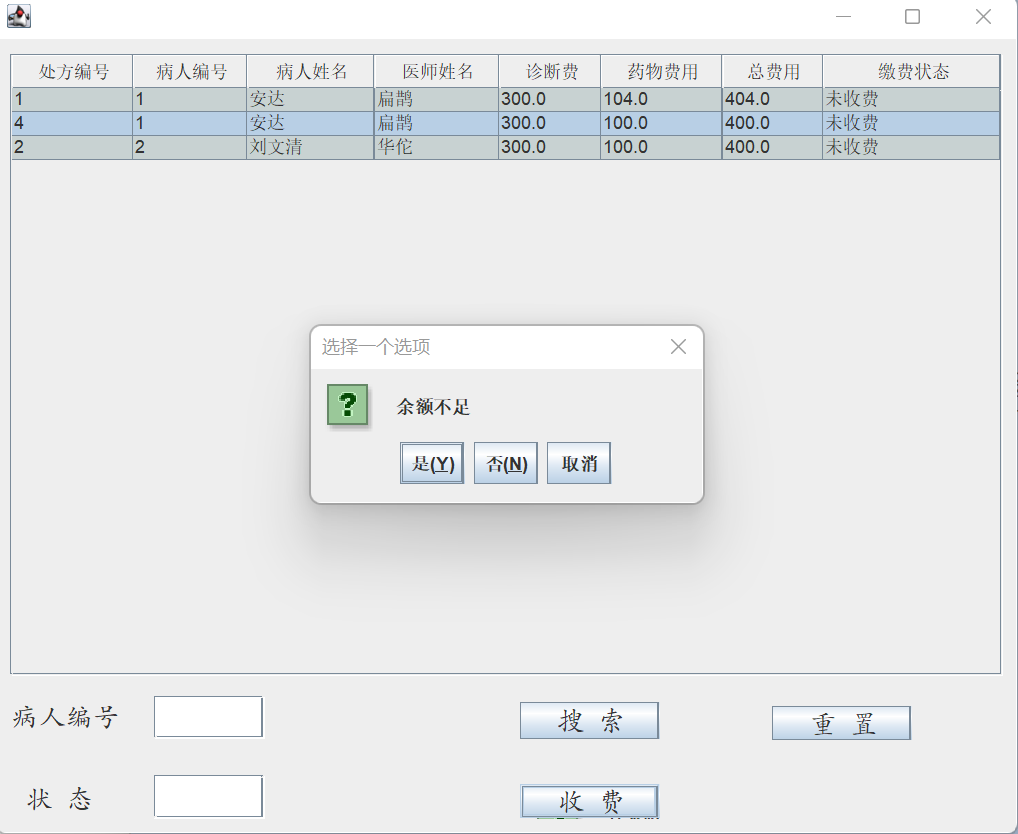


图5.16 余额不足提示框

# 系统设计相关代码

## 存储过程的代码

1. **统计各科室某段时间输入人数**

CREATE DEFINER=`upcmeta9`@`%` PROCEDURE `DEST1`(IN `date1` longtext,IN `date2` longtext)

BEGIN

SELECT Dename,COUNT(\*)

FROM department,booking\_form,doctor

WHERE doctor.Deid=department.Deid and booking\_form.Doid=doctor.Doid and Booktime BETWEEN date1 AND date2

GROUP BY department.Deid;

END

1. **统计各科室某段时间的输入情况**

CREATE DEFINER=`upcmeta9`@`%` PROCEDURE `DEST2`(IN `date1` longtext,IN `date2` longtext)

BEGIN

SELECT Dename,Booktime,booking\_form.Pid,doctor.Doname

FROM department,booking\_form,doctor

WHERE doctor.Deid=department.Deid and booking\_form.Doid=doctor.Doid and Booktime BETWEEN date1 AND date2;

END

## 触发器的代码

**（1）入库触发器（当药品入库时自动更改药品库存）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER t\_i\_s1 AFTER INSERT ON store

FOR EACH ROW

BEGIN

UPDATE medicine SET Snumber=Snumber+new.Snumber WHERE Meid=new.Meid;

END

**（2）出库触发器（当药品出库时自动更改药品库存）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER t\_i\_s AFTER INSERT ON deliver

FOR EACH ROW

BEGIN

UPDATE medicine SET Snumber=Snumber-new.Dnumber WHERE Meid=new.Meid;

END

**（3）添加医生触发器（当新医生信息被添加后，更改所属科室的医师人数）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER t\_i\_s2 AFTER INSERT ON doctor

FOR EACH ROW

BEGIN

UPDATE department SET Dsum=Dsum+1 WHERE Deid = new.Deid;

END

**（4）删除医生触发器（当某个医生离职，医师信息被删除后，更改所属科室的医师人数）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER t\_i\_s4 AFTER DELETE ON doctor

FOR EACH ROW

BEGIN

UPDATE department SET Dsum=Dsum+1 WHERE Deid = new.Deid;

END

**（5）挂号触发器（当挂号信息生成后，生成一条对应的确诊疾病空白的诊断信息）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER re AFTER INSERT ON booking\_form

FOR EACH ROW

BEGIN

INSERT INTO record VALUES(new.Doid,new.Pid," ",new.Booktime);

END

**（6）扣费触发器（缴费单表缴费状态变为“已收费”时，自动对病人账户进行扣费）**

CREATE DEFINER=`upcmeta9`@`%` TRIGGER pay AFTER UPDATE ON booking\_form

FOR EACH ROW

BEGIN

UPDATE patient SET Account=Account-new.Cost WHERE new.State='已收费' AND patient.Pid=new.Pid;

END

## 视图的代码

1. **药品库存视图**

CREATE VIEW mest

AS

SELECT Mename,Snumber

FROM medicine

1. **计算处方费用视图（根据每个处方的药物信息，计算费用）**

CREATE VIEW pay(RXid,sum)

AS

SELECT opener.RXid,SUM(opener.Onumber\*medicine.Price)

FROM opener,medicine

WHERE medicine.Meid=opener.Meid

GROUP BY opener.Rxid

1. **就诊情况视图**

CREATE VIEW dest

AS

SELECT Booktime,Dename,booking\_form.Pid,patient.Pname,doctor.Doname

FROM department,booking\_form,doctor,patient

WHERE doctor.Deid=department.Deid and patient.Pid=booking\_form.Pid AND booking\_form.Doid=doctor.Doid

# 感想及心得体会

本次数据库课设由于工程量不小且是因为第一次做这类项目，中间也遇到许多困难，好在都一一克服了。我的界面使Windowsbuilder插件辅助完成，该插件可以通过拖拽控件的方式来完成页面的设计，大大减少了我的工作量。关于数据库的设计开始由于没理清关系，三番五次的修改，这也给了我警醒，以后先不要着急编写程序，磨刀不误砍柴工。尽管可能页面稍显粗糙，但是这个课设也基本达到了我最初的构想，开发的过程算是一段难忘的记忆。