医院门诊管理系统

1 医院门诊管理系统概述

1.1 背景

医院是以向人们提供医疗服务为主要目的的医疗机构，同时具有卫生事业管理和企业管理的职能。与传统企业相比，医院组织的业务和构成更为复杂。现代医院的综合管理是否先进，直接体现在信息化水平上，医院在经营管理改革过程中面临着管理目标不明确、缺乏先进的管理方法和工具、医院业务流程不清晰等不同方面。此外，医院还面临着日益激烈的市场竞争，要在竞争中取胜，还必须采用先进的管理方法和工具。因此，我们希望通过该系统解决一些常见问题，例如节省患者挂号时间以及更好地管理医院门诊各个部门。

1.2 系统开发目的和意义

系统开发的目的是制作一个网络原生的、集成的、多设施的和可扩展的医院门诊管理系统平台。它可以满足各个医院门诊的行政、运营和临床部门的所有信息和通信需求。此系统是一个现代化的解决方案，使医疗机构能够克服当今医疗服务中的最大挑战，它能够轻松地管理所有部门、病人和员工，改善病人体验，更好地管理药房库存，高效的计费及更多功能。本组认为医院门诊信息系统是现代医院管理的重要工具和手段，是医院深化改革、加强管理、提高效益、和谐发展的重要保障，对于提高医疗质量、促进资源共享、拓展信息服务、提高医院竞争力具有重要意义。总之，通过实施医院门诊信息系统，可以实现医院门诊内部管理一体化、员工工作高效化、科室协作关系简单化、患者费用清单化、诊疗信息电子化，使医疗服务过程更加高效、有序、规范，给医院和患者带来全新的诊疗环境和更加完善的医疗服务。

1.3 国内外现状及分析

近年来，中国政府对医院门诊信息化建设给予了充分的重视和支持，医院门诊信息系统建设发展迅速，但总体情况，特别是在应用系统软件和管理水平方面，与其他先进国家仍有较大差距。目前，县级以上医院才有自己基本的医院门诊信息系统，而其他的乡镇医院才刚开始建设自己的医院门诊信息系统。这些现象都充分说明了人民对医院门诊信息系统建设的认识达到了一个新的高度。医院门诊信息系统的建设为医院门诊带来了效率、效益和较高的管理水平，使我们进一步认识到医院门诊信息系统建设的重要性和必要性。医院门诊信息系统的发展趋势是将各种部门连接起来，更容易共享数据。总而言之，医院门诊信息系统的应用能够进一步地提高中国医院管理水平。

1.4 系统的主要功能

我们的医院门诊管理系统功能包括了门诊预约系统，挂号系统，财务管理系统，后勤管理系统以及药库管理系统。

2 系统分析

2.1 管理功能分析

1) 职员管理

职员管理中包括医生、药房医务人员，以及其他职员比如收费室职员以及挂号职员等。职员系统中包括职员编号、职员账号名以及职员账号密码。职员在输入其职员编号后，会进入到对应的工作页面，并查看及应用相应数据。

2) 患者管理

在患者管理系统中，主要记录患者的基本信息，比如患者姓名、性别、身份证号、住址、患者监护人等。系统中支持添加、修改、删除、查询患者信息。

3) 门诊挂号

门诊的挂号系统包括门诊挂号、挂号单查询以及门诊挂号结账等。如果挂号之前知道病号信息库中已存在该病号，则可以直接调出该病号进行挂号操作。挂号后患者挂号信息会传到医生系统和收费室，医生或收费室则可直接调用。

4) 科室管理

科室系统中主要记载各科室的科室编号、科室名以及联系方式。系统能随时更新科室的信息，方便工作人员参考。

5) 医生管理

医生管理中，维护医生及相关医护人员的基本信息有利于管理者了解医院人员的动态。与患者管理系统相比起来，医生管理系统较复杂但却是很关键的子系统。医生能在该系统记载患者的病历和处方，实现患者病历信息的数字化。医生必须从挂号系统中调用已挂号的病号信息，再输入提交病号处方并传递到收费室产生收费。

6) 诊断结果管理

诊断结果主要记录患者编号、医生编号、病号以及处方号。经过医生诊断，诊断结果生成并记录在系统里。其主要记录患者的个人病情情况并且以处方号为连接，查看处方信息。

7) 处方管理

处方管理系统包括处方单录入和处方单查询。处方管理系统也是医院的重要板块，是医疗工作的重要信息来源。其主要是用来记载医嘱的开立和实施以及相关辅助功能。

8) 挂号收费和处方收费

收费系统主要是根据患者的挂号费以及药费等同时收费的系统。该系统包括可查询患者挂号费用和药品费用。为了给患者以及医务人员带来便利，通过以一次性付款以及收费的方式，给所有相关人士带来行动以及时间上的便利。患者在缴清费用后，则可到药品发放部门取药。

9) 药品发放

药品发放系统具体包括了发药单查询、药品查询。药品发放系统是由药房医务人员来操作的版块，药房通过收款单来确认是否可以给患者发药。患者交款后可以直接到药房取药，节约了大量的人力和时间。发药的同时，药品存库量将同时减少。通过输入处方编号即可查询到要发的药品名称。

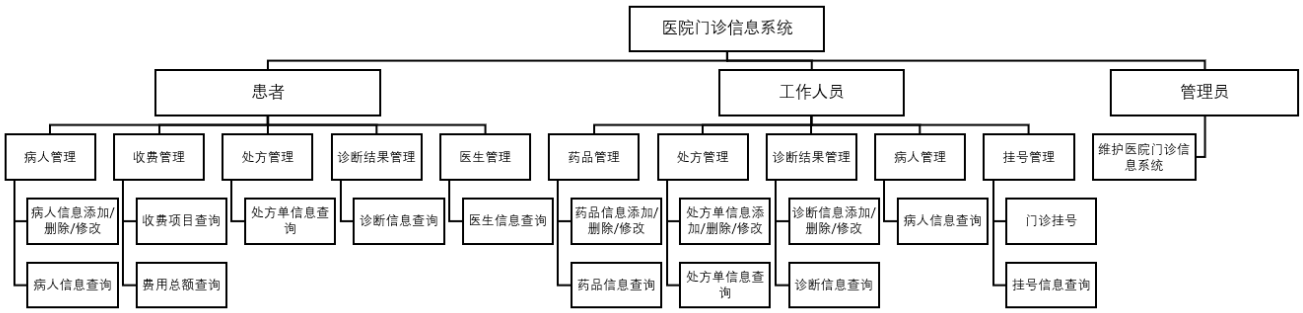

图 1 系统总体功能

2.2 数据处理流程分析

数据字典

数据字典是指描述数据的信息和集合，是对系统中使用的所有数据元素的定义的集合，数据字典在系统设计中占据了很重要的地位。数据字典一般包括数据项、数据结构、数据流、数据存储、处理逻辑等。

2.2.1 数据项

数据项是已经不可再分的数据单位，本系统一共包括 40 个数据项，其中列出了数据项说明、字段类型及长度、与其他数据项的关系以及其取值范围。

实体	数据项名	数据项说明	类型（长度）	关系	是否为空
患者	Pno	患者编号	varchar（20）	主键，外键	
	Pname	患者姓名	varchar（20）		NOT NULL
	Psex	患者性别	varchar（20）		NOT NULL
	Page	患者年龄	int		
	Ptel	患者电话	varchar（20）		NOT NULL
	Ppass	登录密码	varchar（20）		NOT NULL
挂号单	Rno	挂号编号	varchar（20）	主键，外键	NOT NULL
	Rway	挂号方式	varchar（20）		
	Rdate	挂号日期	date		NOT NULL
	Pno	患者编号	varchar（20）	外键	NOT NULL
	Dno	医生编号	Varchar（20）	外键	NOT NULL
	Bno	收费单号	varchar（20）		NOT NULL
科室	DPno	科室编号	varchar（20）	主键	

	DPname	科室名称	varchar（20）		NOT NULL
	DPtel	科室电话	varchar（20）		NOT NULL
	DPaddr	科室楼层	varchar（3）		NOT NULL
医生	Dno	医生编号	varchar（20）	主键	
	Dname	医生姓名	varchar（20）		NOT NULL
	Dsex	医生性别	varchar（20）		NOT NULL
	Dtitle	医生职称	varchar（20）		NOT NULL
	Dtel	医生电话	varchar（20）		NOT NULL
	DPno	科室编号	varchar（20）	外键	NOT NULL
诊断结果	Dno	医生编号	varchar（20）	外键	
	Pno	患者编号	varchar（20）	外键	
	Illness	疾病症状	varchar（20）		NOT NULL
	PRno	处方编号	varchar（20）	外键	
处方	PRno	处方编号	varchar（20）	主键	
	PRdate	开方日期	date		NOT NULL
	Mno	药品编号	varchar（20）	外键	
	Bno	收费单号	varchar（20）	外键	NOT NULL
收费单	Bno	收费单号	varchar（20）	主键，外键	
	Bdate	收费日期	date		
	Bmoney	收费金额	float		NOT NULL
	Brsn	收费原因	varchar（20）		
药品	Mno	药品编号	varchar（20）	主键	
	Mname	药品名字	varchar（20）		NOT NULL
	Mprice	药品价格	float		NOT NULL
	Mquantity	药品数量	int		NOT NULL
职员	Eid	职员工号	varchar（20）	主键	
	Ename	职员姓名	varchar（20）		NOT NULL
	Epass	职员登录密码	varchar（20）		NOT NULL

图 2 数据项

2.2.2 数据结构

数据结构反映数据之间的关系，一共九个实体也就是具有九个数字结构，其中说明了实体的结构名及其属性。

数据实体	数据结构名	属性
患者	Patient	Pno,Pname,Psex,Page,Ptel,Ppass
挂号单	Register	Rno,Rway,Rdate,Pno,Bno,Dno
科室	Department	DPno,DPname,DPtel,DPaddr
医生	Doctor	Dno,Dname,Dsex,Dtitle,Dtel,DPno
诊断结果	Diagnose	Dno,Pno,Illness,PRno
处方	Prescription	PRno,PRdate,Mno,Bno
收费单	Bill	Bno,Bdate,Bmoney,Brsn
药品	Medicine	Mno,Mname,Mprice,Mquantity
职员	Employee	Eid,Ename,Epass

图 3 数据结构

2.2.3 数据流

数据流是数据结构在系统内传输的途径，一共九条数据流。

数据流编号	数据流名称	简述	数据流来源	数据流去向	数据流组成	数据流量	高峰流量
F1	挂号请求	患者进行挂号申请。	患者	挂号处理	患者信息、分配医生	每日 1000 人	每日 5000 人
F2	挂号单	挂号单由患者信息组成，并生成挂号费用。挂号处针对患者的需求为其分配医生。	挂号处理	患者	挂号单编号，患者、患者信息、分配医生、科室、收费信息	每日 1000 次	每日 5000 次
F3	看病	患者去到医生诊室号进行就诊。	患者	诊断处理	患者信息、初诊信息、患者	每日 1000 人	每日 5000 人
F4	诊断信息	将诊断信息汇总形成患者病历。	诊断处理	确诊处理	患者、治疗信息	每日 1000 次	每日 5000 次
F5	处方	医生针对患者的病情开处方给患者同时生成处方费用。	确诊处理	患者	患者、处方信息、病历信息、收费信息	每日 1000 次	每日 5000 次
F6	缴费	患者根据各项费用进行结算。	患者	收费处理	收费信息	每日 1000 次	每日 5000 次
F7	收费凭证	患者缴费后获取收费凭证以备后续使用。	收费处理	患者	收费信息、详细记录	每日 1000 次	每日 5000 次
F8	取药	患者到药房领取药品。	患者	取药处理	患者、药品信息	每日 1000 次	每日 5000 次
F9	药物	药房人员依照处方将所列药品交给患者。	取药处理	患者	患者、药品信息	每日 1000 次	每日 5000 次

图 4 数据流

2.2.4 数据流程图

数据流程图主要展示的是数据的流向和去向，反映信息在系统中的流动、处理和存储情况。数据流程图主要分为顶层数据流程图、零层数据流程图和第一层数据流程图。

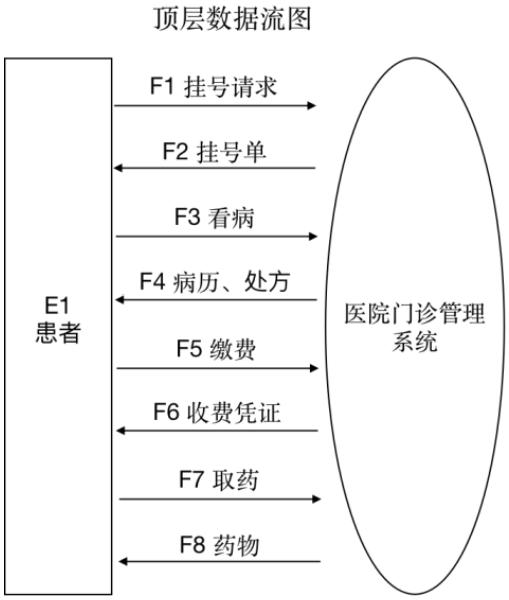

图 5 顶层数据流程图

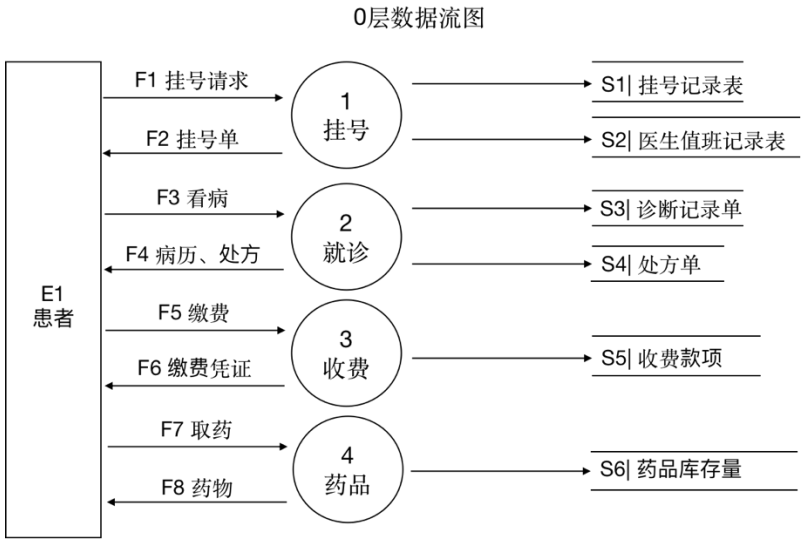

图 6 第零层数据流层图

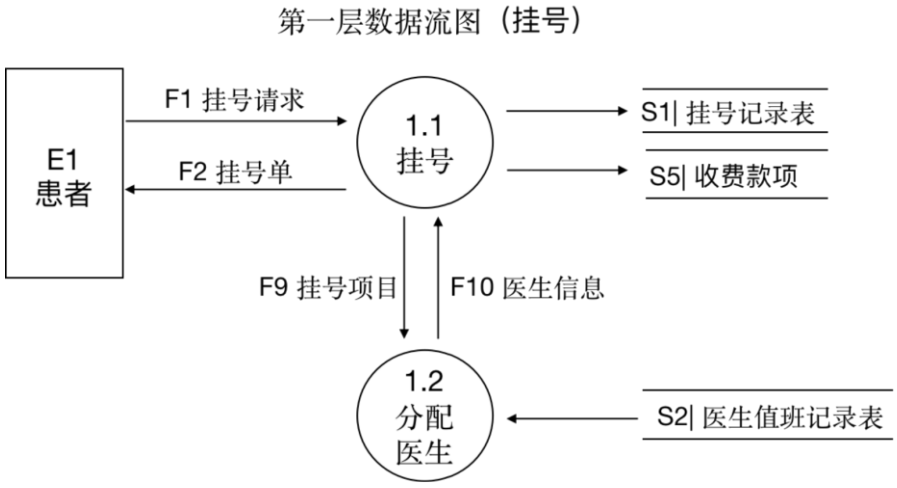

图 7 第一层数据流程图（挂号）

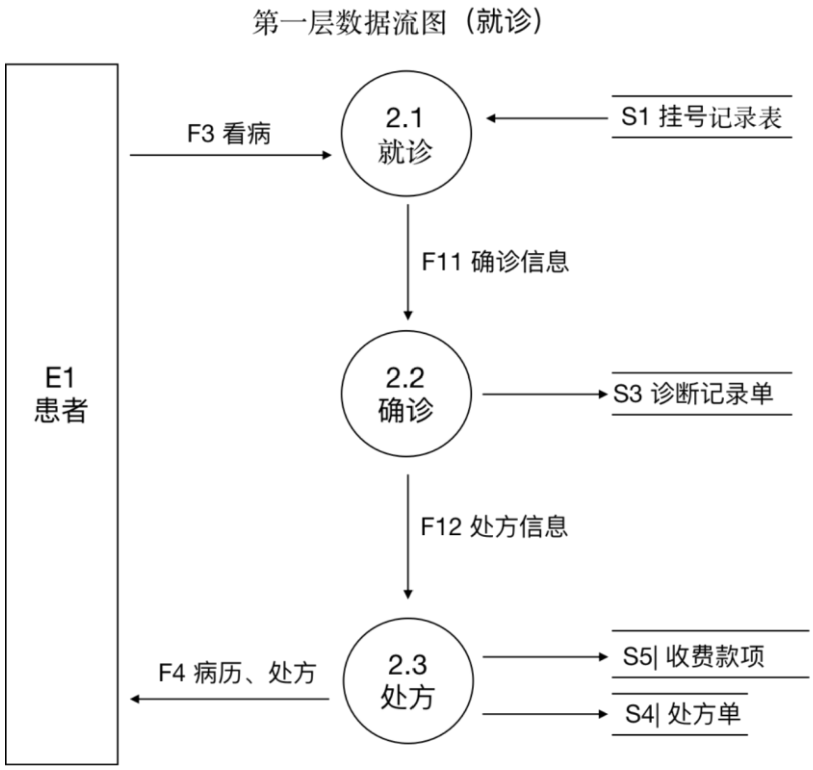

图 8 第一层数据流程图（就诊）

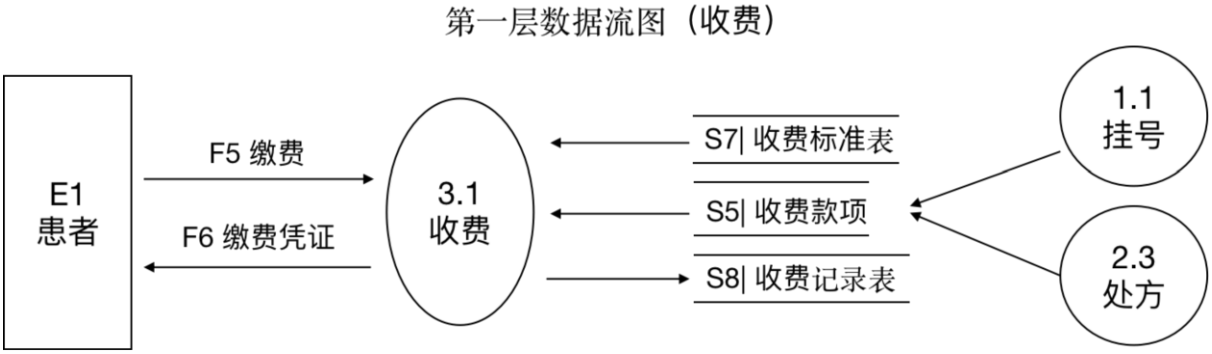

图 9 第一层数据流程图（收费）

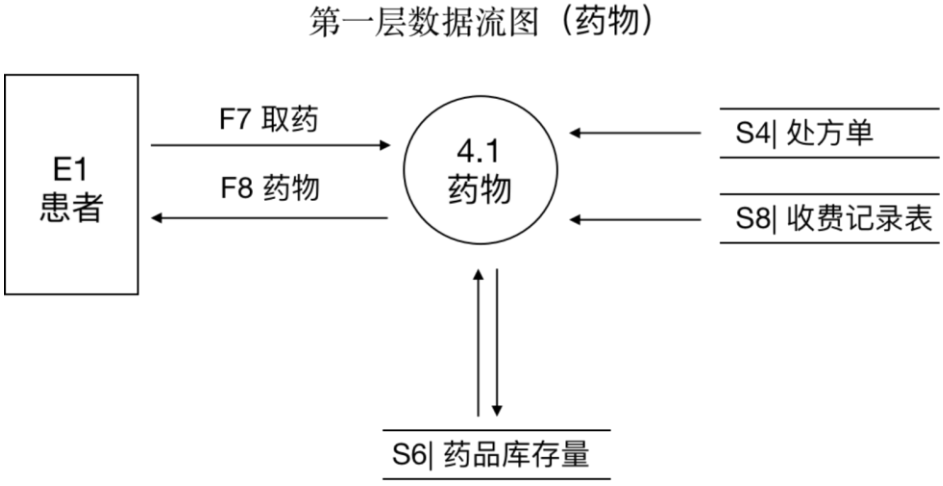

图 10 第一层数据流图（药物）

2.2.5 业务流程图

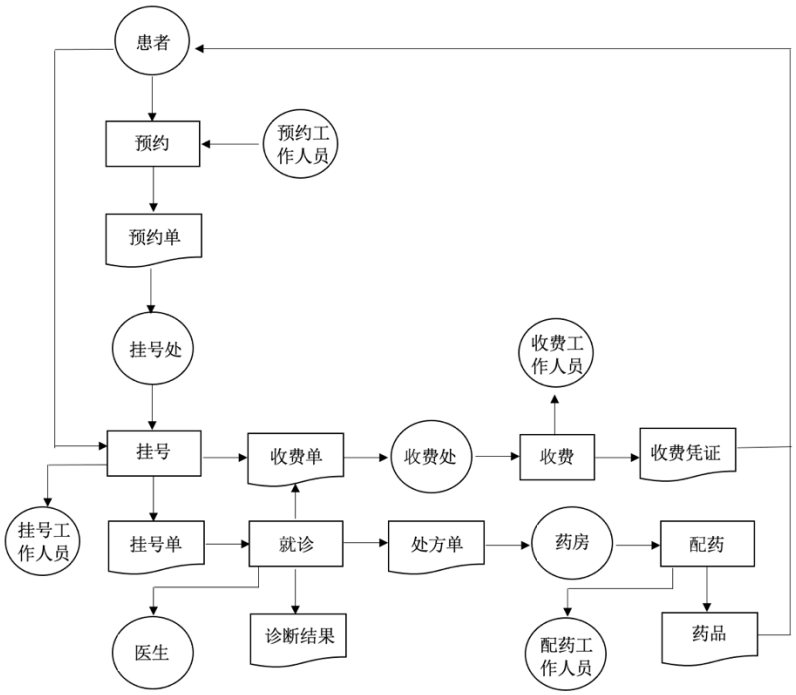

图 11 业务流程图

2.2.6 数据存储

数据存储是数据保存的地方也是数据流的来源和去向之一。其中说明了该数据存储的简述、数据村粗的组成及相关的处理。

数据存储编号	数据存储名称	简述	数据存储组成	相关的处理
S1	挂号记录表	记录患者的挂号信息	患者信息，医生	1.1，2.1
S2	医生值班记录表	记录医生的值班安排	医生值班安排	1.2
S3	诊断记录单	记录患者的病情及诊断过程	诊断过程	2.2
S4	处方单	记录医生给患者开的处方	处方信息	2.3，4.1
S5	收费款项	记录患者看诊的所有费用详情，如挂号费，处方费	收费信息	1.1，2.3， 3.1
S6	药品库存量	记录药物价格剩余量等	药品信息	4.1
S7	收费标准表	收费的标准	收费款项标准	3.1
S8	收费记录表	记录所有患者的收费情况	收费信息	3.1，4.1

图 12 数据存储

2.2.7 处理逻辑

本报告采用判定表来描述处理逻辑。主要包括处理逻辑的简述、输入输出的数据流以及该处理的事物。

处理逻辑	处理逻辑名称	简述	输入的数据流	处理	输出的数据流	处理频率
1.1	挂号	处理对患者挂号的请求	患者信息	安排挂号	挂号单	每人一次
1.2	分配医生	根据挂号请求分配医生	挂号项目	分配医生	医生信息	每人一次
2.1	就诊	医生对病情进行诊断	诊断请求	初步诊断	诊断信息	每人一次
2.2	确诊	完成诊断，宣布确诊	诊断信息	确诊	诊断结果	每人一次
2.3	处方	针对患者的病情开处方	诊断结果	开处方	处方单	每人一次
3.1	收费	费用内容和标准	收费标准	收取费用	收费记录	每人一次
4.1	药物	领取药物或退药	处方单	取药退药	药物信息	每人一次

图 13 处理逻辑

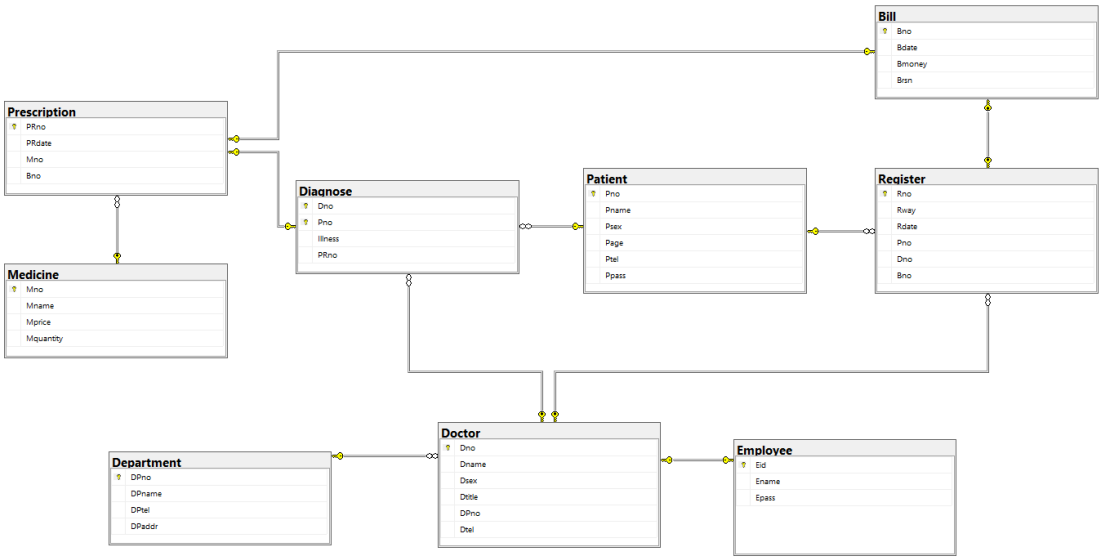

图 14 UML 建模方法

2.3 存储分析

ER 模型指的是实体关系模型，在概念结构设计阶段中用来描述信息的需求和存储在数据库中的信息类型。它充分地反映事物与事物之间的联系，并且在逻辑设计上，概念模型要映射到逻辑模型如关系模型上。我们将 ER 图分成总 ER 图以及属性 ER 图，并列出了所有实体与联系属性的属性 ER 图。

以用户视角来述说，患者登记挂号后拥有挂号单并同时生成挂号收费单，接着通过科室分配医生给患者就诊。医生以工作人员身份登入系统，在确定患者确诊后撰写诊断书并产生处方，患者同时收到该诊断结果以及处方单，同时间处方费用录入处方收费单。患者缴费后，以处方单为准领取药品。

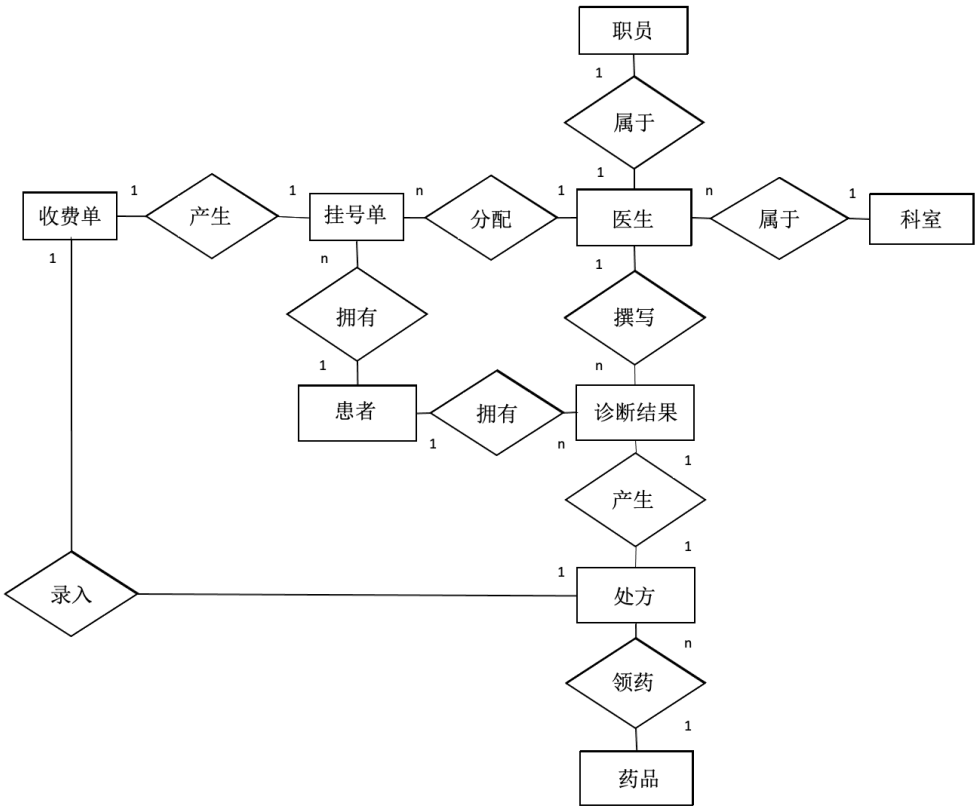

图 15 总 ER 图

业务规则：

1. 患者由唯一标识的患者编号构成，需存储患者的姓名、性别、年龄等基本信息。不允许同一个患者在同一个时段内就诊不同的部门。
2. 职员由职员编号编号唯一识别，医生属于职员。在医生表里为主键，用于记录医生姓名、联系电话、职位等信息。医生编号相等于职员编号。
3. 一个科室部门可以有多个医生，但是一个医生只属于一个科室。当就诊相同病症的多位医生属于同一科室部门时，具有不同的医生编号。
4. 挂号单由挂号编号唯一识别，需存储挂号时间等信息。
5. 患者挂号后，需记录患者的挂号信息。
6. 诊断结果编号构成诊断结果的唯一识别，需记录病名、处方、就诊医生等信息。一份诊断结果只属于一个患者。
7. 处方编号构成处方的唯一识别，需存储就诊日期等信息。
8. 医生给一位患者就诊，只写一张处方。
9. 药品由药品编号唯一识别，需存储药品名称、价格、库存等信息。
10. 每个患者拥有属于自己的挂号单、诊断结果和处方。

在逻辑结构设计中，医院门诊管理系统所涉及到的关系主要有：

1. 患者和医生为 n:1（多对一）的关系，将其之间的联系与 n 端实体合并。
2. 患者和挂号单的关系为 1:n（一对多），将其之间的联系与 n 端实体合并。
3. 科室和医生为 1:n（一对多）的关系，将其之间的联系与 n 端实体合并
4. 医生和诊断结果的关系为 1:n（一对多），将其之间的联系与任意一端实体合并。
5. 诊断结果和处方单的关系为 1:1（一对一），将其之间的联系与任意一端实体合并。
6. 处方单、收费单和药品之间的联系为三元的关系，将它们之间的联系转换为独立的关系模式。

具体转换为下列关系模式型（主键加下划线，外键为***斜体加粗***），确定函数依赖及属于第几范式：

1. 职员（职员工号、职员姓名、职员登录密码）
   职员工号→职员姓名，职员工号→职员登录密码
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
2. 患者（患者编号，患者姓名，患者性别，患者年龄，患者电话，登录密码）
   患者编号→患者姓名，患者编号→患者性别，患者编号→患者年龄，患者编号→患者电话，患者编号→登录密码
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
3. 医生（医生编号，医生姓名、医生性别、医生职称、医生电话、***科室编号***）
   医生编号→医生姓名，医生编号→医生性别，医生编号→医生职称，医生编号→电话，医生编号→科室编号
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
4. 科室（科室编号，科室名称，科室电话、科室楼层）
   科室编号→科室名称，科室编号→科室电话，科室编号→科室楼层
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
5. 挂号单（挂号编号，挂号方式，挂号日期，收费单号、***患者编号***，***医生编号***）
   挂号编号→挂号方式，挂号编号→挂号日期，挂号编号→收费单号, 挂号单号→患者编号、挂号编号→医生编号
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
6. 诊断结果（***医生编号***，***患者编号***，疾病症状，***处方编号***）
   （医生编号，患者编号）→处方编号，患者编号→疾病症状
   有传递依赖，没有部分依赖，因此符合第三范式要求。
7. 处方（处方编号，开方日期，***药品编号***，***收费单号***）
   处方编号→开方日期，处方编号→药品编号，处方编号→收费单号
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。

8. 药品（药品编号，药品名字，药品价格，药品数量）
   药品编号→药品名字，药品编号→药品价格，药品编号→药品数量
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。
9. 收费单（收费单号，收费日期，收费金额，收费原因）
   收费单号→收费日期，收费单号→收费金额，收费单号→收费原因
   上述没有传递依赖及部分依赖关系，因此符合第三范式要求。

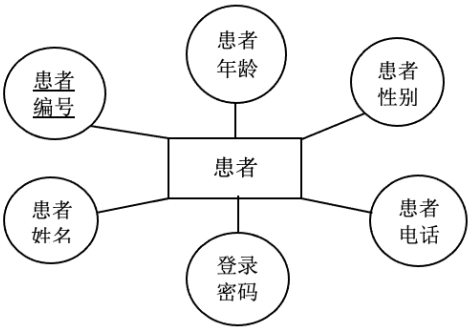

图 16 患者属性 ER 图

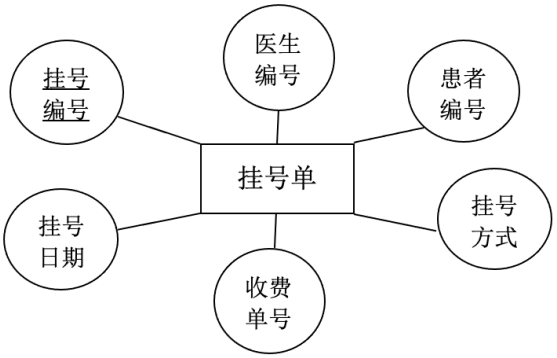

图 17 挂号单属性 ER 图

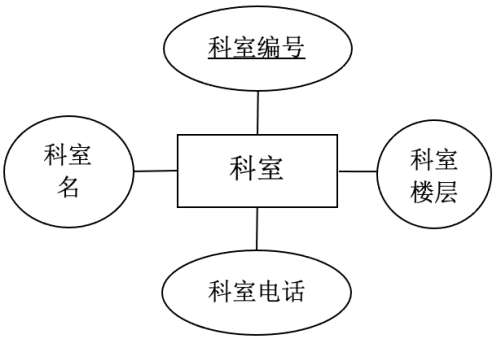

图 18 科室属性 ER 图

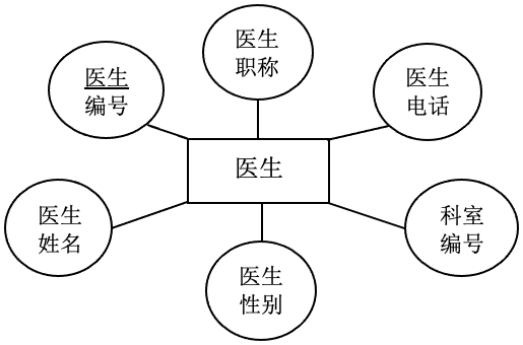

图 19 医生属性 ER 图

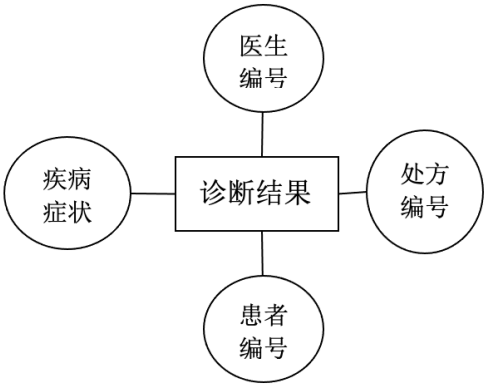

图 20 诊断结果属性 ER 图

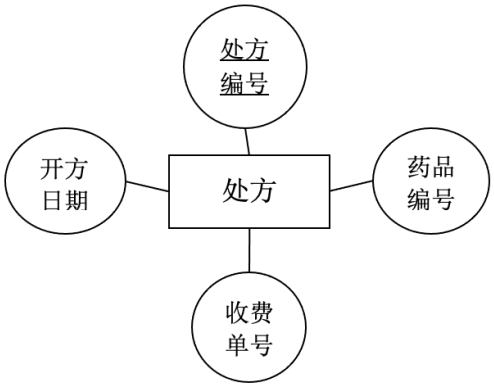

图 21 处方属性 ER 图

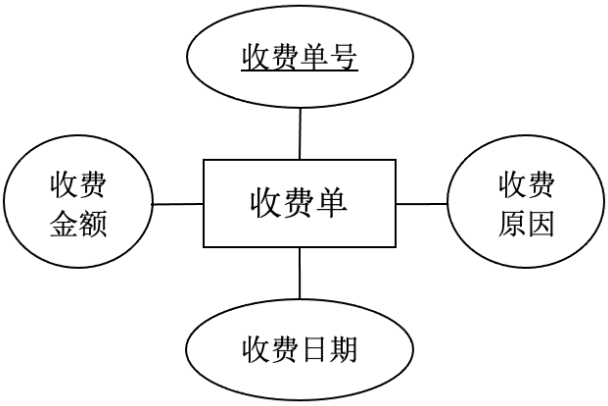

图 22 收费单属性 ER 图

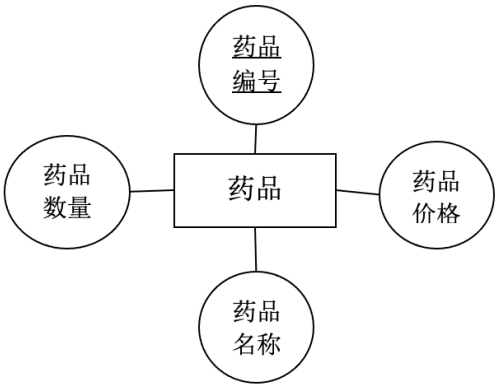

图 23 药品属性 ER 图

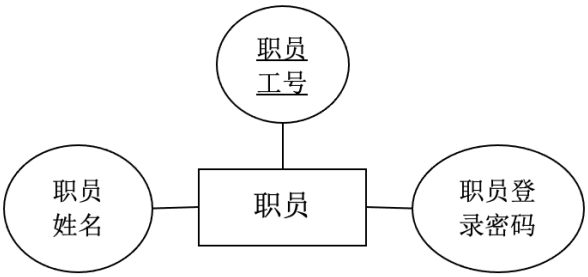

图 24 职员属性 ER 图

2.4 查询要求分析

数据库的物理结构主要包括数据库在物理设备上的存储结构与存取方式。物理结构中，数据的基本存取单位是存储记录。通过建立适合的索引，能加速对表的查询访问，当按条件查询的时候首先先查询索引，再通过索引查询相关数据。

```
create unique index unique_Mname on Medicine(Mname);
create unique index unique_Pname on Patient(Pname);
create index idx_bno on Bill(Bno asc);
```

图 25 索引

2.5 输出输入分析

系统实施后，不同用户端的界面实现的基本功能也不同。下面将对医院门诊管理系统中各界面进行介绍。

患者（用户）首先需登录信息录入界面，此界面主要用于录入患者的详细信息，以便门诊部门方便调用信息。接着，患者可通过在界面上获取诊断信息、取药单、缴费等举动，并获取挂号单、诊断信息、处方以及缴费凭证。

管理人员首先也需登录到界面，其基本操作主要是维持系统正常运行，实现各个方面信息，如患者信息、医生信息、科室信息、挂号记录、收费管理、处方管理、诊断结果管理、药品管理的添加和删除，并且输出相应信息。

工作人员（职员）填写相应信息登录到界面后，工作人员能添加患者诊断记录、患者药物处方或处理收费凭证以及药物管理等。随后，系统就会把相应信息输出到患者界面。

图 26 医院门诊管理系统 HIPO 图

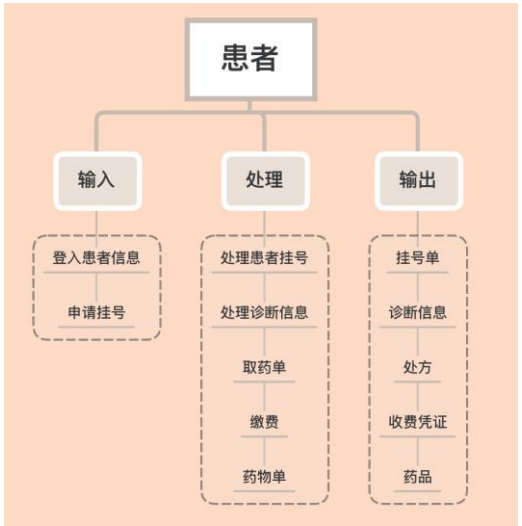

图 27 患者 HIPO 图

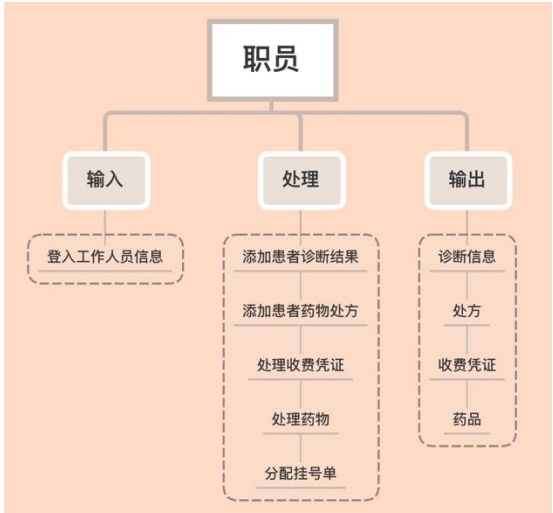

图 28 职员 HIPO 图

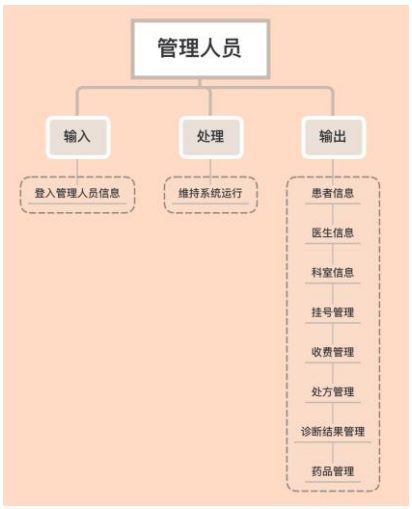

图 29 管理人员 HIPO 图

2.6 系统性能要求分析

本组设计的系统主要面向几个性能给各用户提供便利，也就是模块化设计、信息共享、准确及时交流信息、操作简单、实时特色以及查询功能齐全。

1. 模块化设计

   其具有良好的可扩充性，以适应医院不同阶段的发展需要。通过方便的系统剪裁功能，各子系统间能任意选择是否连接。

2. 信息共享、准确及时交流信息

   其主要是通过发挥网络功能，达到减少重复操作，提高工作效率的目的。通过充分利用计算机网络及关系型数据库的资源共享、数据共享等技术，彻底改变手工或单机管理对信息收集处理中的重复、混乱和容易出错的状况。当一个环节录入信息后，其它环节可以共享，确保数据的准确性和一致性。在基本信息录入时，系统采用拼音输入方式，鼠标操作，大大提高工作效率。

3. 操作简单，维护方便

   以人机界面友好，操作简单为基，用户不需要记忆任何计算机命令，并且每个系统都具有系统维护功能，对可变化的项目可自行维护，不需改变程序。

4. 实时特色

   实时特色主要体现在二十四小时不间断的高度安全性和可靠性、数据传输准确快速以及能很好地适应医院门诊业务流程需要。

5. 查询功能齐全

   该系统能够在任意时间内对每个子系统的业务情况、统计报表进行汇总及查询，同时能对几种情况任意组合查询、统计，这操作很大程度上减少了统计人员的工作强度。此外，院领导通过查询系统也能及时了解业务情况或财务情况。借助此性能帮助，院里的事后统计变为实时跟踪，静态管理则更为动态管理，提高了医院的管理水平运行环境。

3 系统分析

3.1 系统架构设计

系统架构设计的本质是清晰、有序地对系统进行构架，以让业务的发展快速扩展。架构是经过思考斟酌后在现有的资源约束下做出最合理的决策。由于门诊业务繁重，因此采用分布式架构作为此系统模型。分布式系统是一个硬件或软件系统分布在不同的网路计算机上，彼此之间仅仅通过消息的传递进行通信及协调。在分布式系统中，一组独立计算机展示给使用者的就算是一个整体，系统可以拥有多个通用的物理和逻辑架构，通过不同的物理和逻辑架构实现信息交换。通常，使用者都不会关心系统是属于单机还是属于分布式，用户关心的只是使用舒适度及可用性。因此，本系统采用统一的访问接口和方式，不会因为分布式系统内部的变动而改变访问的方式。

3.2 功能结构设计

1. 登录模块

功能： 其功能是验证登录者的用户名和密码是否存在以及是否相符。若验证成功则打开相应操作界面，否则将显示用户名密码错误提示。
输入： 用户名和密码
处理： 到用户数据库中验证是否存在及是否正确
输出： 用户名和密码错误提示，及返回相应操作页面

2. 挂号模块

功能： 录入患者信息生成挂号单，该挂号单贯穿于患者整个就诊过程，生成的挂号单暂存于挂号单数据库中，以供医生诊断以及取药付款调用。
输入： 患者编号、所挂科室编号、患者基本信息（患者编号、姓名、性别、年龄、住址电话）。
输出： 挂号单、包含患者编号及其基本信息。该基本信息写入患者信息库，该挂号单写入挂号单数据库。

3. 问诊模块

功能： 给医生提供患者基本信息，包含个人信息及病史，供诊断使用。在医生开药时提供一个药品检查功能，如果医生所开药物在药物数据库中查询不到，则返回开药失败。
输入： 患者编号、所开药物名称、诊断结果（文本格式）
输出： 诊断结果写入病史数据库、所开药物列表写入挂号单数据库、输出诊断后的挂号单以供取药付款使用。

3.3 系统流程图设计

系统流程图是一种描绘系统物理模型的图示，以黑盒子的形式用图形符号描述系统中每个部件的流动情况。有了系统流程图，可以使相关人员更了解系统业务的处理情况。由于登录此系统的使用者分成两种，也就是一般用户及职员，因此会有两种程序流程图。两张程序图的操作基本一样，用户及职员在登录界面输入登录信息后，由系统判断使用者是否输入正确的登录信息，若用户为首次使用则返回登录页面进行注册；若用户或职员输入错误的登录信息同样也将返回登录页面。若输入正确的登录信息，使用者将被带进系统页面中，以查询或编辑相应信息。在使用完毕后，则点击退出系统就是结束。

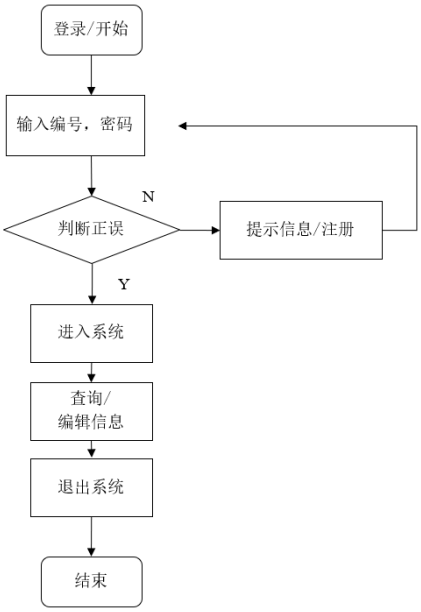

图 30 一般用户程序流程图

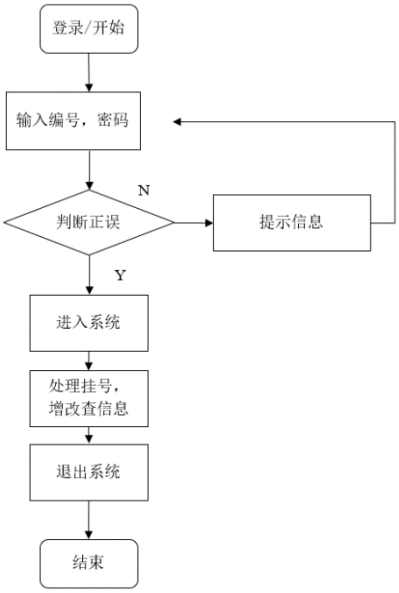

图 31 职员程序流程图

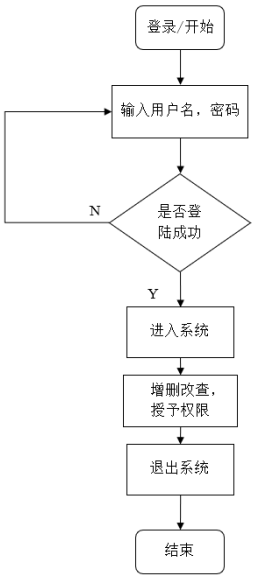

图 32 管理人员程序流程图

3.4 数据存储设计

随着医院门诊每日运行，为了保证各人士业务进行顺利，门诊各方面的数据存储起着很大的作用。门诊的运行主要依靠数据的存储确保所有业务无误。

当患者挂号时，Register 表就会新增一条挂号信息，同时在 Patient 表添加该患者记录以及在 Bill 表中具有一条挂号收费记录。随后患者确诊完毕后，Diagnose 表会新增一条诊断结果记录，同时在 Prescription 表拥有一条处方记录及在 Bill 表中会有相应的收费信息。通过数据的存储，具有权限的人可插入一条药品信息、修改某科室的电话、修改某药品剩余量、在 Dept Doctor 查询医院门诊部各个科室的医生人数、在 Doctor_Patient 查询某患者的主治医生信息、在 Doctor_Patient 查询某医生主治的全部患者信息、在 Patient_Diag 查询某患者的诊断结果、在 Patient_Med 查询某患者购买的药品、在 BillDetails 查询某患者购买的药品总额以及在 BillDetails 查询某患者的收费总额。

```
Create Procedure Add_Patient (
p_Rno varchar(20),
p_Rway varchar(20),
p_Rdate datetime(3),
p_Pno varchar(20),
p_Bno varchar(20),
p_Bdate datetime(3),
p_Pname varchar(20),
p_Psex varchar(20),
p_Page int,
p_Ppass varchar(20),
p_Dno varchar(20),
p_Bmoney double,
p_Ptel varchar(20))
begin
insert into Patient values(p_Pno,p_Pname,p_Psex,p_Page,p_Ptel,p_Ppass);
insert into Bill values(p_Bno,NOW(3),p_Bmoney,'挂号');
insert into Register values(p_Rno,p_Rway,NOW(3),p_Pno,p_Dno,p_Bno);
```

图 33 患者挂号数据存储

```
Create procedure addDiagnose (
p_Dno varchar(20),
p_Pno varchar(20),
p_Illness varchar(20),
p_PRno varchar(20),
p_PRdate datetime(3),
p_Mno varchar(20),
p_Bno varchar(20),
p_Bmoney double,
p_Bdate datetime(3))
begin
insert into Bill
values(p_Bno,NOW(3),p_Bmoney,',药品');
insert into Prescription
values(p_PRno,NOW(3),p_Mno,p_Bno);
insert into Diagnose
values(p_Dno,p_Pno,p_Illness,p_PRno);
```

图 34 患者就诊数据存储

```
Create procedure MedicineInsert (
p_Mno varchar(20),
p_Mname varchar(20),
p_Mprice double,
p_Mquantity int)
begin
insert into Medicine
values(p_Mno,p_Mname,p_Mprice,p_Mquantity);
```

图 35 插入药品信息

```
Create procedure change_tel (
p_DPno varchar(20),
p_DPtel varchar(20))
begin
update Department
set DPtel = p_DPtel
where DPno = p_DPno;
```

图 36 修改某一科室电话

```
Create procedure change_med (
p_Mno varchar(20),
p_Mquantity int)
begin
update Medicine
set Mquantity=p_Mquantity
where Mno=p_Mno;
```

图 37 修改药品剩余量

```
Create procedure Dept_Doc()
begin
select DPname,COUNT(Dno) as rs
from Dept_Doctor
group by DPname;
```

图 38 查询各个科室医生人数

```
Create procedure P_doctor (
p_Pname varchar(20))
begin
select *
from Doctor
where Dno =(select Dno from Doctor_Patient where Pname=p_Pname);
```

图 39 查询患者的主治医生信息

```
Create procedure P_Diag (
p_Pno varchar(20))
begin
select Illness
from Patient_Diag
where Pno=p_Pno;
```

图 40 查询患者的诊断结果

```
Delimiter //
Create procedure D_Patient (
p_Dno varchar(20))
begin
select Pno,Pname,Psex,Page
from Patient
where Pname in(select Pname from Doctor_Patient where Dno=p_Dno);
```

图 41 查询医生主治的全部患者信息

```
Create procedure P_Med (
p_Pno varchar(20))
begin
select Mname
from Patient_Med
where Pno=p_Pno;
```

图 42 查询患者购买的药品

```
Create procedure MedPay (
p_Pno varchar(20))
begin
select Brsn,Bmoney
from BillDetails
where Pno=p_Pno and Brsn='%,药品';
```

图 43 查询患者购买的药品总额

```
Create procedure Pay (
p_Pno varchar(20))
begin
select sum(Bmoney) as total
from BillDetails
where Pno=p_Pno
group by Pno;
```

图 44 查询患者的收费总额

下面展示的是各种视图。

```
Select * from BillDetails;
```

Pno	Bno	Bdate	Bmoney	Brsn
P004	B004	2021-10-03	66	挂号，诊查，取药
P003	B003	2021-10-03	18	挂号，诊查
P002	B002	2021-10-01	50.55	挂号，诊查，取药
P001	B001	2021-10-01	10	挂号，诊查

图

图 45 收费视图

```
Select * from Dept_Doctor;
```

Dno	Dname	DPname
D001	李小玉	内科
D002	简隋英	外科
D003	顾飞飞	妇科
D004	蒋丞	儿科
D005	江添	眼科
D006	盛晓望	皮肤科
D007	吕冰冰	口腔科
D008	程博衍	耳鼻喉科
D009	许静	中医科
D010	陈泊桥	神经科
D011	杨萱	体检科
D012	方琪文	男科

图 46 医生科室视图

```
Select * from E_ID_Pass_Name;
```

Eid	Ename	Epass
D001	Andy123	andy123
D002	Nicole	nicole123
D003	Cindy	cindy123
D004	Danny	danny123
D005	Eddie	eddie123
D006	Frankie	frankie123
D007	Giselle	giselle123
D008	Henry	henry123
D009	Irene	irene123
D010	Jason	jason123
D011	Kelly	kelly123
D012	Lucas	lucas123

图 47 患者登录信息视图

```
Select * from Patient_Diag;
```

Pno	Pname	Illness
P001	姜楚	肺炎，呼吸不顺
P004	温隋	感冒
P003	陈娜娜	胃炎
P002	简言	失眠，食欲不振

图 48 患者诊断结果

```
Select * from Patient_Med;
```

Pno	Mname
P001	板蓝根颗粒
P002	B族维生素片
P003	安胃片
P004	感冒清热颗粒

图 49 患者药品视

```
Select * from Patient_Register;
```

Pno	Pname	Ptel	Rno	Rway	Rdate	Dno
P001	姜楚	13263691111	R001	电话预约	2021-10-01 00:00:00.000	D001
P002	简言	13263692222	R002	窗口预约	2021-10-01 00:00:00.000	D010
P003	陈娜娜	13263693333	R003	窗口预约	2021-10-03 00:00:00.000	D004
P004	温隋	13263694444	R004	电话预约	2021-10-03 00:00:00.000	D002

图 50 患者挂号视图

```
Select * from Med_Price;
```

Mname	Mprice
板蓝根颗粒	10
B族维生素片	15
安胃片	5
感冒清热颗粒	5
红霉素眼膏	10

图 51 药品价格视图

```
Select * from Doctor_Patient;
```

Dno	Pno	Dname	Pname	Ptel
D001	P001	李小玉	姜楚	13263691111
D010	P002	陈泊桥	简言	13263692222
D004	P003	蒋丞	陈娜娜	13263693333
D002	P004	简陪英	温陪	13263694444

图 52 医生患者视图

3.5 界面设计

本系统的使用者为患者、医务人员、管理人，都属于非计算机专业人员，因此我们采用简单明了的界面设计，以让系统使用者能够更容易上手。在正式进入系统前，管理人员、职员以及用户需分别选择登录页面，随后输入其登录信息，其中需确保输入的正确性以及系统要求性。信息成功验证后，使用者则会被带到相应的主菜单界面。以职员为例，职员在登录成功后在功能选择界面可选择查询相关信息，如科室管理、医生资料、患者资料、挂号资料、处方资料、诊断结果、药品资料、收费资料以及职员资料。

图 53 系统界面

图 54 管理员登录界面

图 55 职员登录界面

图 56 用户登录界面

图 57 游客操作界面

图 58 操作选择界面

图 59 科室操作界面

图 60 医生操作界面

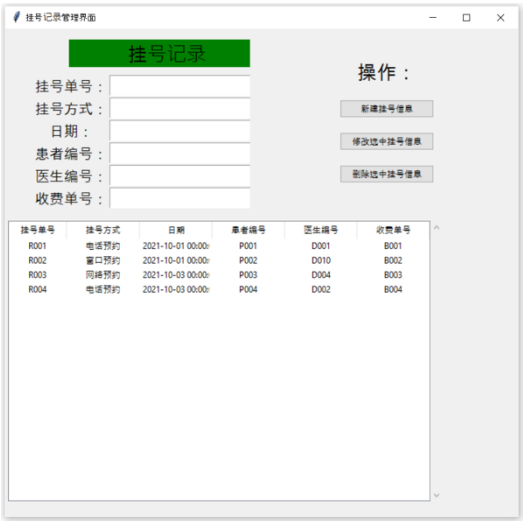

图 61 挂号操作界面

图 62 处方操作界面

图 63 药品操作界面

图 64 收费操作界面

图 65 患者操作界面

图 66 职员操作界面

3.6 程序设计

我们以 MSSQL 数据库系统为基础，再将数据代码转变为 MYSQL 语句，创建门诊数据库以及各种基本表，并且在这基础上，可以增加插入表里信息。此外，通过创建视图以方便不同用户及系统使用者查看数据库中的数据。视图不仅可以简化用户的理解，也可以确保信息的安全性，用户只能查看被授予权限的信息。

```
Create Database Hospital;
Use Hospital;
```

图 67 Hospital 数据库

创建基本表：

```
Create Table Employee
(Eid varchar(20) Primary Key,
 Ename varchar(20) UNIQUE NOT NULL,
 Epass varchar(20) NOT NULL check(char_length(rtrim(Epass))>6),
 CONSTRAINT Check_Ename check (Ename like '%[a-z]%' OR Ename like '%[0-9]%' OR Ename like '%[A-Z]%'),
 CONSTRAINT Check_Epass check (Epass like '%[a-z]%' OR Epass like '%[0-9]%' OR Epass like '%[A-Z]%')
);
```

图 68 职员基本表

```
Create Table Department
    (DPno varchar(20) Primary Key,
    DPname varchar(20) UNIQUE NOT NULL,
    DPtel varchar(11)UNIQUE NOT NULL,
    DPaddr varchar(3) NOT NULL,
    CONSTRAINT Check_DPno CHECK(DPno like 'DP%'),
    CONSTRAINT Check_DPaddr CHECK(DPaddr like 'F%'),
    CONSTRAINT Check_DPtel CHECK(DPtel not like '%[^0-9]%')
    );
```

图 69 科室基本表

```
Create Table Doctor
    (Dno varchar(20) Primary Key references Employee(Eid),
    Dname varchar(20) NOT NULL,
    Dsex varchar(2) CHECK(Dsex='男' or Dsex='女') NOT NULL,
    Dtitle varchar(20) NOT NULL,
    DPno varchar(20) NOT NULL references Department(DPno),
    Dtel varchar(20) UNIQUE NOT NULL,
    CONSTRAINT Check_Dtel CHECK(Dtel not like '%[^0-9]%')
    );
```

图 70 医生基本表

```
Create Table Patient
    (Pno varchar(20) Primary Key,
    Pname varchar(20) NOT NULL,
    Psex varchar(20) DEFAULT('男') CHECK(Psex='男' or Psex='女') NOT NULL,
    Page int CHECK(Page>=1 and Page<=150),
    Ptel varchar(20) NOT NULL,
    Ppass varchar(20) NOT NULL,
    CONSTRAINT Check_Pno CHECK(Pno like 'P%'),
    CONSTRAINT Check_Ptel CHECK(Ptel not like '%[^0-9]%'),
    CONSTRAINT Check_Ppass check (Ppass like '%[a-z]%' OR Ppass like '%[0-9]%' OR Ppass like '%[A-Z]%')
    );
```

图 71 患者基本表

```
Create Table Register
    (Rno varchar(20) Primary Key NOT NULL,
    Rway varchar(20),
    Rdate datetime(3) DEFAULT(now(3)) NOT NULL,
    Pno varchar(20) NOT NULL references Patient(Pno),
    Dno varchar(20) NOT NULL references Doctor(Dno),
    Bno varchar(20) UNIQUE NOT NULL,
    CONSTRAINT Check_Rno CHECK(Rno like 'R%'),
    CONSTRAINT Check_Bno CHECK(Bno like 'B%')
    );
```

图 72 挂号基本表

```
Create Table Diagnose
    (Dno varchar(20) references Doctor(Dno),
    Pno varchar(20) references Patient(Pno),
    Illness text NOT NULL,
    PRno varchar(20) UNIQUE,
    Primary Key(Dno,Pno),
    CONSTRAINT Check_PRno CHECK(PRno like 'PR%')
    );
```

图 73 诊断结果基本表

```
Create Table Prescription
    (PRno varchar(20) Primary Key references Diagnose(PRno),
     PRdate date DEFAULT(now(3))NOT NULL,
     Mno varchar(20) references Medicine(Mno),
     Bno varchar(20) NOT NULL references Bill(Bno)
     );
```

图 74 处方基本表

```
Create Table Medicine
     (Mno varchar(20) Primary Key,
     Mname varchar(20) UNIQUE NOT NULL,
     Mprice double NOT NULL CHECK(Mprice>0),
     Mquantity int NOT NULL CHECK(Mquantity>=1),
     CONSTRAINT Check_Mno CHECK(Mno like 'M%'),
     CONSTRAINT Check_Mprice CHECK(Mprice not like '%[^0-9]%')
     );
```

图 75 药品基本表

```
Create Table Bill
     (Bno varchar(20) Primary Key references Register(Bno),
     Bdate date DEFAULT(now(3)),
     Bmoney float DEFAULT('10') CHECK(Bmoney>0)NOT NULL,
     Brsn varchar(20) DEFAULT('挂号,诊查'),
     CONSTRAINT Check_Bmoney CHECK(Bmoney not like '%[^0-9]%_%')
     );
```

图 76 收费基本表

```
Insert into Employee values
     ('D001','Andy123','andy123'),
     ('D002','Nicole','nicole123'),
     ('D003','Cindy','cindy123'),
     ('D004','Danny','danny123'),
     ('D005','Eddie','eddie123'),
     ('D006','Frankie','frankie123'),
     ('D007','Giselle','giselle123'),
     ('D008','Henry','henry123'),
     ('D009','Irene','irene123'),
     ('D010','Jason','jason123'),
     ('D011','Kelly','kelly123'),
     ('D012','Lucas','lucas123'),
     ('PH001','JordanWang','0308wangxiaobo'),
     ('PH002','NickYang','yangjiaqiang0508'),
     ('S001','TiffanyLiu','liuyuqing3330'),
     ('S002','DonaldJian','yanhaojie9901');
```

图 77 职员表插入数据

```
Insert into Department values
     ('DP001','内科','04512340987','F1'),
     ('DP002','外科','04514521234','F1'),
     ('DP003','妇科','04510987654','F2'),
     ('DP004','儿科','04511234567','F2'),
     ('DP005','眼科','04518798729','F3'),
     ('DP006','皮肤科','04512946842','F3'),
     ('DP007','口腔科','04512049537','F4'),
     ('DP008','耳鼻喉科','04518478883','F4'),
     ('DP009','中医科','04519473395','F5'),
     ('DP010','神经科','04518469083','F5'),
     ('DP011','体检科','04514378605','F6'),
     ('DP012','男科','04519074093','F6');
```

图 78 科室表插入数据

```
Insert into Doctor values
    ('D001','李小玉','男','中级医师','DP001','83051201'),
    ('D002','简隋英','女','初级医师','DP002','83051202'),
    ('D003','顾飞飞','女','正高级医师','DP003','83051203'),
    ('D004','蒋丞','男','副高级医师','DP004','83051204'),
    ('D005','江添','男','中级医师','DP005','83051205'),
    ('D006','盛晓望','男','初级医师','DP006','83051206'),
    ('D007','吕冰冰','女','正高级医师','DP007','83051207'),
    ('D008','程博衍','男','中级医师','DP008','83051208'),
    ('D009','许静','女','初级医师','DP009','83051209'),
    ('D010','陈泊桥','男','副高级医师','DP010','83051210'),
    ('D011','杨萱','女','中级医师','DP011','83051211'),
    ('D012','方琪文','男','初级医师','DP012','83051212');
```

图 79 医生表插入数据

```
Insert into Patient values
    ('P001','姜楚','女','50','13263691111','Anny123'),
    ('P002','简言','男','26','13263692222','bobby123'),
    ('P003','陈娜娜','女','30','13263693333','nana0529'),
    ('P004','温隋','女','10','13263694444','11Daisyyy');
```

图 80 患者表插入数据

```
Insert into Medicine values
    ('M001','板蓝根颗粒','10','200'),
    ('M002','B族维生素片','15','400'),
    ('M003','安胃片','5','500'),
    ('M004','感冒清热颗粒','5','500'),
    ('M005','红霉素眼膏','10','300');
```

图 81 药品表插入数据

```
Insert into Diagnose values
    ('D001','P001','肺炎,呼吸不顺','PR001'),
    ('D010','P002','失眠,食欲不振','PR002'),
    ('D004','P003','胃炎','PR003'),
    ('D002','P004','感冒','PR004'),
    ('D005','P005','沙眼','PR005');
```

图 82 诊断结果表插入数据

```
Insert into Prescription values
    ('PR001','2021-10-1','M001','B001'),
    ('PR002','2021-10-1','M002','B002'),
    ('PR003','2021-10-3','M003','B003'),
    ('PR004','2021-10-3','M004','B004');
```

图 83 处方表插入数据

创建视图：

```
Create View BillDetails
      As Select Distinct Diagnose.Pno,Bill.Bno,Bdate,Bmoney,Brsn
         From Prescription,Bill,Diagnose,Register
         Where Register.Pno=Diagnose.Pno
               And (Diagnose.PRno = Prescription.PRno
                    And Prescription.Bno = Bill.Bno
                    Or Register.Bno=Bill.Bno);
```

图 84 收费细则视图

```
Create View Dept_Doctor
      As Select Dno,Dname,DPname
         From Doctor,Department
         Where Department.DPno = Doctor.DPno
```

图 85 科室医生视图

```
Create View Doctor_Patient
      As Select Register.Dno,Register.Pno,Dname,Pname,Ptel
         From Patient,Register,Doctor
         Where Patient.Pno = Register.Pno
              And(Doctor.Dno=Register.Dno);
```

图 86 医生患者视图

```
Create View Patient_Register
      As Select Patient.Pno,Pname,Ptel,Rno,Rway,Rdate,Register.Dno
         From Patient,Register
         Where Register.Pno = Patient.Pno
```

图 87 患者挂号视图

```
Create View Patient_Diag
        As Select Patient.Pno,Pname,Illness
           From Patient,Diagnose
           Where Diagnose.Pno = Patient.pno ;
```

图 88 诊断结果视图

```
Create View Patient_Med
       As Select Pno,Mname
          From Medicine,Prescription,Diagnose
          Where Medicine.Mno=Prescription.Mno
               And Diagnose.PRno = Prescription.PRno ;
```

图 89 药品视图

```
Create View Med_Price
        As Select Mname,Mprice
           From Medicine
```

图 90 药品费用视图

```
Create View E_ID_Pass_Name
       As Select Eid,Ename,Epass
          From Employee,Doctor
          Where Employee.Eid=Doctor.Dno
```

图 91 医生查看个人信息视图

此外，我们通过触发器来确保数据的完整性以及一致性。当向处方表(Prescription)中添加元组时,同步对药品表(Medicine)更新，即药品数量要减少相应数目。

```
Create Trigger T1 on Prescription
after insert,update
as
declare @PRno varchar(20),@Mno varchar(20)
declare c2 cursor for select PRno,Mno from inserted
open c2
fetch next from c2 into @PRno,@Mno
while(Not_found=0)
do
    update Medicine
    set Mquantity=Mquantity-1
    where Mno=@Mno
    fetch next from c2 into @PRno,@Mno
end while
close c2
```

图 92 T1 触发器

接着，我们将给予游客、患者、职员及管理人员一些相应的权限。通过给游客授予权限后，游客仅能查看医生所在科室、药品价格以及科室信息。患者的权限则是仅能更新患者信息、查看费用情况、医生所在科室、医生患者信息、诊断结果、药品信息、挂号信息、药品价格及科室信息。

由于部分操作都由职员来完成，因此职员的权限也相对应多。相应部门的职员可以查看、输入、更新挂号信息；查看、更新、插入患者信息、收费信息、药品信息、诊断结果、处方；查看、更新科室信息；查看患者挂号信息、收费信息、医生所在科室、医生患者信息、患者诊断结果、患者药品；操作医生所在科室信息、增加患者、增加诊断结果、输入药品、修改电话、修改药品、操作患者医生信息、操作患者诊断结果、操作患者处方、操作患者缴费等。管理员权限主要是对各部门信息的增删改查。

```
CREATE Role Guests

GRANT Select On Dept_Doctor To Guests;
GRANT Select ON Med_Price TO Guests;
GRANT Select ON Department TO Guests;
```

图 93 游客权限

```
CREATE Role Patients;

GRANT Select(Pno,Pname,Psex,Page,Ptel,Ppass),UPDATE(Pname,Psex,Page,Ptel,Ppass) On Patient To Patients;
GRANT Select ON BillDetails To Patients;
GRANT Select ON Dept_Doctor To Patients;
GRANT Select ON Doctor_Patient To Patients;
GRANT Select ON Patient_Diag To Patients;
GRANT Select ON Patient_Med To Patients;
GRANT Select ON Patient_Register To Patients;
GRANT Select ON Department To Patients;
GRANT Select ON Med_Price To Patients;
```

图 94 患者权限

```
CREATE Role Employees;

GRANT Select,Insert,Update ON Register To Employees;
GRANT Select,Update,Insert ON Patient To Employees;
GRANT Select,Update,Insert ON Bill To Employees;
GRANT Select,Update,Insert ON Medicine To Employees;
GRANT Select,Update,Insert ON Diagnose To Employees;
GRANT Select,Update ON Department To Employees;
GRANT Select,Update ON Doctor To Employees;
GRANT Select,Update,Insert ON Prescription To Employees;
GRANT Select ON Patient_Register To Employees;
GRANT Select ON BillDetails To Employees;
GRANT Select ON Dept_Doctor To Employees;
GRANT Select ON Doctor_Patient To Employees;
GRANT Select ON Patient_Diag To Employees;
GRANT Select ON Patient_Med To Employees;
GRANT EXECUTE ON Dept_Doc To Employees;
GRANT EXECUTE ON Add_patient To Employees;
GRANT EXECUTE ON addDiagnose To Employees;
GRANT EXECUTE ON MedicineInsert To Employees;
GRANT EXECUTE ON change_tel To Employees;
GRANT EXECUTE ON change_med To Employees;
GRANT EXECUTE ON Dept_Doc To Employees;
GRANT EXECUTE ON P_doctor To Employees;
GRANT EXECUTE ON D_Patient To Employees;
GRANT EXECUTE ON P_Diag To Employees;
GRANT EXECUTE ON P_Med To Employees;
GRANT EXECUTE ON MedPay To Employees;
GRANT EXECUTE ON Pay To Employees;
```

图 95 职员权限

```
CREATE Role Admin;

GRANT Select,Update,Insert,Delete on Doctor to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Employee to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Department to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Medicine to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Register to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Bill to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Prescription to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Diagnose to Admin WITH GRANT OPTION;
GRANT Select,Update,Insert,Delete on Patient to Admin WITH GRANT OPTION;
```

图 96 管理人员权限

数据控制部分采用 Python 连接 MySQL 数据库来操纵数据，其主要步骤为：

1. 增：
   a. 增加科室
      Sql = "INSERT INTO Department (DPno, DPname, DPtel,DPaddr) VALUES ('%s','%s','%s','%s')"
      输入科室编号，科室名称，科室联系方式，科室楼层进行匹配 SQL 语句。
   b. 增加医生
      Sql = "INSERT INTO Doctor (Dno, Dname,Dsex,Dtitle,DPno,Dtel) VALUES ('%s','%s','%s','%s','%s','%s')"
      输入医生编号，姓名，性别，职称，科室编号和电话进行匹配 SQL 语句。
   c. 增加患者
      Sql = "INSERT INTO Patient(Pno,Pname,Psex,Page,Ptel,Ppass) VALUES ('%s','%s','%s','%s','%s','%s')"

输入患者的编号，姓名，性别，年龄，联系方式和患者登录密码进行匹配 SQL 语句。

d. 增加药品
Sql = "INSERT INTO Medicine (Mno,Mname,Mprice,Mquantity) VALUES ('%s','%s','%s','%s')"
输入药品的编号，名称，价格和库存进行匹配 SQL 语句。

e. 增加处方
Sql = "INSERT INTO Prescription (PRno,PRdate,Mno,Bno) VALUES ('%s','%s', '%s', '%s')"
输入处方的编号，就诊日期，药品编号和收费单进行匹配 SQL 语句。

f. 增加收费单
Sql = "INSERT INTO Bill (Bno,Bdate,Bmoney,Brsn) VALUES ('%s','%s','%s','%s')"
输入收费单号，日期，收费总额和收费方式进行匹配 SQL 语句。

g. 增加挂号单
Sql = "INSERT INTO Register (Rno,Rway,Rdate,Pno,Bno) VALUES ('%s','%s','%s','%s','%s')"
输入挂号单号，挂号方式，挂号日期，患者编号和收费单号来进行匹配 SQL 语句。

2. 删：
a. 删除患者：输入患者编号，删除其挂号记录，诊断结果记录和患者记录。
I. 删除挂号记录
Sql ="DELETE FROM Register WHERE Pno = '%s'"
II. 删除诊断结果记录
Sql ="DELETE FROM Diagnose WHERE Pno = '%s' "
III. 删除患者记录
Sql ="DELETE FROM Patient Where Pno = '%s'"

3. 改：
a. 修改药品库存：输入药品编号来修改药品库存
Sql = "UPDATE Medicine SET Mquantity = %d WHERE Mno = '%s' "
b. 修改药品售价：输入药品编号来修改药品售价。
Sql = "UPDATE Medicine SET Mprice = %d WHERE Mno = '%s' "

接着，通过 Python tkinter 实现系统的界面设计，安装 Python 的 pymysql 包并利用它将数据库写入 MySQL。

```
import pymysql
from tkinter import ttk
import tkinter as tk
import tkinter.font as tkFont
from tkinter import *
import tkinter.messagebox as messagebox

class Startpage:
    def __init__(self, parent_window):
        parent_window.destroy()
        self.window = tk.Tk()
        self.window.title('医院门诊管理系统')
        self.window.geometry('600x700')
        photo = tk.PhotoImage(file="img11.png")
        label =tk.Label(self.window,
                        text="医院门诊管理系统",
                        justify=tk.LEFT,
                        image=photo,
                        compound=tk.CENTER,
                        font=("SimHei", 30),
                        fg="black")
        label=Label(self.window,text="医院门诊管理系统", font=("SimHei", 30),image=photo)  #
        label.pack(pady=10)
        Button(self.window, text="管理员登录", font=tkFont.Font(size=16), command=lambda: Adminpage(self.window), width=30,
               height=2,fg='black', bg='gray', activebackground='gray', activeforeground='black').pack()
        Button(self.window, text="职员登录", font=tkFont.Font(size=16), command=lambda: Staffpage(self.window), width=30,
               height=2, fg='black', bg='gray', activebackground='gray', activeforeground='black').pack()
        Button(self.window, text="用户登录", font=tkFont.Font(size=16), command=lambda: userspage(self.window), width=30,
               height=2,fg='black', bg='gray', activebackground='gray', activeforeground='black').pack()
        Button(self.window, text="游客登录", font=tkFont.Font(size=16), command=lambda: Youkepage(self.window), width=30,
               height=2,fg='black', bg='gray', activebackground='gray', activeforeground='black').pack()
        Button(self.window, text='退出系统', height=2, font=tkFont.Font(size=16), width=30, command=self.window.destroy,
               fg='black', bg='gray', activebackground='gray', activeforeground='black').pack()
        self.window.mainloop()
```

图 97 导入过程

```
class Adminpage:
    def __init__(self, parent_window):
        parent_window.destroy()
        self.window = tk.Tk()
        self.window.title('管理员登录页面')
        self.window.geometry('300x450')
        label = tk.Label(self.window, text='管理员登录', bg='Green', font=('Simsun', 20), width=30, height=2)
        label.pack()
        Label(self.window, text='管理员账号: ', font=tkFont.Font(size=14)).pack(pady=25)
        self.admin_username = tk.Entry(self.window, width=30, font=tkFont.Font(size=14), bg='Ivory')
        self.admin_username.pack()
        Label(self.window, text='管理员密码: ', font=tkFont.Font(size=14)).pack(pady=25)
        self.admin_pass = tk.Entry(self.window, width=30, font=tkFont.Font(size=14), bg='Ivory', show='*')
        self.admin_pass.pack()
        Button(self.window, text="登录", width=8, font=tkFont.Font(size=12), command=self.login,fg='white', bg='gray', activebackground='black', activeforeground='white'
        Button(self.window, text="返回首页", width=8, font=tkFont.Font(size=12), command=self.back,fg='white', bg='gray', activebackground='black', activeforeground='whit
        self.window.protocol("WM_DELETE_WINDOW", self.back)
        self.window.mainloop()
    def login(self):
        print(str(self.admin_username.get()))
        print(str(self.admin_pass.get()))
        admin_pass = None
```

图 98 管理员登录页面的初始化

```
class Gongnengjiemian:
    def __init__(self, parent_window):
        parent_window.destroy()
        self.window= tk.Tk()
        self.window.title('功能选择界面')
        self.window.geometry('300x560')
        Button(self.window, text="科室管理", width=18, font=tkFont.Font(size=12), command=lambda: AdminManage(self.window),fg='white', bg='green', activebackground='black
        Button(self.window, text="医生资料管理", width=18, font=tkFont.Font(size=12), command=lambda: AdminManage1(self.window),fg='white', bg='green', activebackground='
        Button(self.window, text="患者资料管理", width=18, font=tkFont.Font(size=12),command=lambda: AdminManage2(self.window),fg='white', bg='green', activebackground='b
        Button(self.window, text="挂号资料管理", width=18, font=tkFont.Font(size=12),command=lambda: AdminManage3(self.window),fg='white', bg='green', activebackground='b
        Button(self.window, text="处方资料管理", width=18, font=tkFont.Font(size=12),command=lambda: AdminManage4(self.window),fg='white', bg='green', activebackground='b
        Button(self.window, text="诊断结果资料管理", width=18, font=tkFont.Font(size=12), command=lambda: AdminManage5(self.window),fg='white', bg='green', activebackgrour
        Button(self.window, text="药品资料管理", width=18, font=tkFont.Font(size=12),  command=lambda: AdminManage6(self.window),fg='white', bg='green', activebackground=
        Button(self.window, text="收费资料管理", width=18, font=tkFont.Font(size=12),  command=lambda: AdminManage7(self.window),fg='white', bg='green', activebackground=
        Button(self.window, text="职员资料管理", width=18, font=tkFont.Font(size=12),  command=lambda: AdminManage8(self.window),fg='white', bg='green', activebackground=
        Button(self.window, text="返回首页", width=8, font=tkFont.Font(size=12), command=self.back,fg='white', bg='gray', activebackground='black', activeforeground='whit
        self.window.protocol("WM_DELETE_WINDOW", self.back)
        self.window.mainloop()
    def back(self):
        Startpage(self.window)
```

图 99 功能界面的初始化

```
class Staffpage:
    def __init__(self, parent_window):
        parent_window.destroy()
        self.window = tk.Tk()
        self.window.title('职员登录')
        self.window.geometry('300x450')
        label = tk.Label(self.window, text='职员登录', bg='green', font=('Simsun', 20), width=30, height=2)
        label.pack()
        Label(self.window, text='职员账号: ', font=tkFont.Font(size=14)).pack(pady=25)
        self.student_id = tk.Entry(self.window, width=30, font=tkFont.Font(size=14), bg='Ivory')
        self.student_id.pack()
        Label(self.window, text='职员密码: ', font=tkFont.Font(size=14)).pack(pady=25)
        self.student_pass = tk.Entry(self.window, width=30, font=tkFont.Font(size=14), bg='Ivory', show='*')
        self.student_pass.pack()
        Button(self.window, text="登录", width=8, font=tkFont.Font(size=12), command=self.login,fg='white', bg='gray', activebackground='black', activeforeground='white'
        Button(self.window, text="返回首页", width=8, font=tkFont.Font(size=12), command=self.back,fg='white', bg='gray', activebackground='black', activeforeground='whit
        self.window.protocol("WM_DELETE_WINDOW", self.back)
        self.window.mainloop()
    def login(self):
        print(str(self.student_id.get()))
        print(str(self.student_pass.get()))
        stu_pass = None
```

图 100 职员登录页面的初始化

本系统管理员操作界面里共有 9 个功能页面，在此只展示科室部分的代码，即科室资料管理操作页面、增加科室信息、修改科室信息、删除科室信息。此外还有游客进入页面初始化、插入 SQL 语句，删除、增加及使用数据库及插入 SQL 语句创建基本表和增添数据。

```
class AdminManage:
    def __init__(self, parent_window):
        parent_window.destroy()
        self.window = Tk()
        self.window.title('科室资料操作界面')
        self.frame_left_top = tk.Frame(width=300, height=200)
        self.frame_right_top = tk.Frame(width=300, height=200)
        self.frame_center = tk.Frame(width=600, height=400)
        self.frame_bottom = tk.Frame(width=750, height=50)

        self.columns = ("科室编号", "科室名称", "联系方式","科室楼层")
        self.tree = ttk.Treeview(self.frame_center, show="headings", height=18, columns=self.columns)
        self.vbar = ttk.Scrollbar(self.frame_center, orient=VERTICAL, command=self.tree.yview)

        self.tree.configure(yscrollcommand=self.vbar.set)

        self.tree.column("科室编号", width=100, anchor='center')
        self.tree.column("科室名称", width=100, anchor='center')
        self.tree.column("联系方式", width=100, anchor='center')
        self.tree.column("科室楼层", width=100, anchor='center')

        self.tree.grid(row=0, column=0, sticky=NSEW)
        self.vbar.grid(row=0, column=1, sticky=NS)
        self.dpno = []
        self.dpname = []
        self.dptel = []
        self.dpaddr =[]

        db = pymysql.Connect(
            host='127.0.0.1',
            port=3306,
            user='Boon',
            passwd='123456',
            db='hospital',
            charset='utf8'
            )
        cursor = db.cursor()  # 使用cursor()方法获取操作游标
```

图 101 科室资料管理操作页面

```
    sql = "INSERT INTO department(dpno, dpname, dptel,dpaddr) VALUES('%s', '%s', '%s', '%s') " % (self.var_dpno.get(), self.var_dpname.get(), self.var_dptel.get

    try:
        cursor.execute(sql)
        db.commit()
    except:
        db.rollback()
        messagebox.showinfo('警告！', '数据库连接失败2！')
    db.close()
    self.dpno.append(self.var_dpno.get())
    self.dpname.append(self.var_dpname.get())
    self.dptel.append(self.var_dptel.get())
    self.dpaddr.append(self.var_dpaddr.get())
    self.tree.insert('', len(self.dpno) - 1, values=(self.dpno[len(self.dpno) - 1], self.dpname[len(self.dpno) - 1], self.dptel[len(self.dpno) - 1], self.dpaddr
    self.tree.update()
    messagebox.showinfo('提示！', '插入成功！')
else:
    messagebox.showinfo('警告！', '请填写科室数据')
```

图 102 增加科室信息

```
    sql = "UPDATE department SET dpname = '%s',dptel = '%s', dpaddr = '%s' WHERE dpno= '%s'" % (self.var_dpname.get(), self.var_dptel.get(), self.var_dpno.get()
    try:
        cursor.execute(sql)
        db.commit()
        messagebox.showinfo('提示！', '更新成功！')
    except:
        db.rollback()
        messagebox.showinfo('警告！', '更新失败，数据库连接失败3！')
    db.close()
    dpno_index = self.dpno.index(self.row_info[0])
    self.dpname[dpno_index] = self.var_dpname.get()
    self.dptel[dpno_index] = self.var_dptel.get()
    self.dpaddr[dpno_index] = self.var_dpaddr.get()
    self.tree.item(self.tree.selection()[0], values=(self.var_dpno.get(), self.var_dpname.get(), self.var_dptel.get(), self.var_dpaddr.get()))
else:
    messagebox.showinfo('警告！', '不能修改科室姓名！')
```

图 103 修改科室信息

```
sql = "DELETE FROM department WHERE dpno = '%s'" % (self.row_info[0])
try:
    cursor.execute(sql)
    db.commit()
    messagebox.showinfo('提示！', '删除成功！')
except:
    db.rollback()
    messagebox.showinfo('警告！', '删除失败，数据库连接失败4！')
db.close()
dpno_index = self.dpno.index(self.row_info[0])
print(dpno_index)
del self.dpno[dpno_index]
del self.dpname[dpno_index]
del self.dptel[dpno_index]
del self.dpaddr[dpno_index]
print(self.dpno)
self.tree.delete(self.tree.selection()[0])  # 删除所选行
print(self.tree.get_children())
```

图 104 删除科室信息

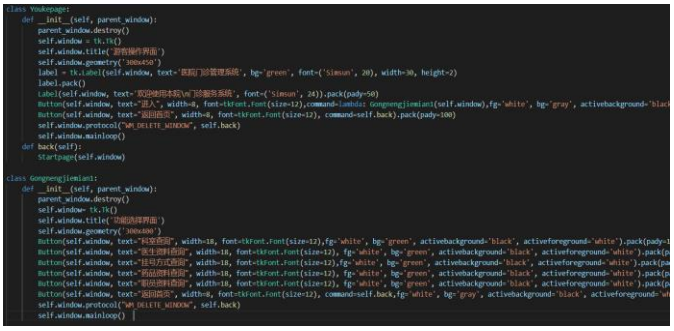

图 105 游客进入页面初始化

```
sql = """
DROP Database IF EXISTS hospital
 """
cursor.execute(sql)

sql = """
CREATE Database IF NOT EXISTS hospital
 """
cursor.execute(sql)

sql = """
Use hospital
 """
cursor.execute(sql)
```

图 106 删除、增加及使用数据库

```
sql = """
CREATE TABLE IF NOT EXISTS department(
    dpno varchar(20) Primary Key,
    dpname varchar(20) UNIQUE NOT NULL,
    dptel varchar(11)UNIQUE NOT NULL,
    dpaddr varchar(3) NOT NULL,
    CONSTRAINT Check_dpno CHECK(dpno like 'DP%'),
    CONSTRAINT Check_dpaddr CHECK(dpaddr like 'F%'),
    CONSTRAINT Check_dptel CHECK(dptel not like '%[^0-9]%')
    )ENGINE = InnoDB
    DEFAULT    CHARSET = utf8;
    """
cursor.execute(sql)

sql = """
 Insert into department (dpno,dpname,dptel,dpaddr)
 values
    ('DP001','内科','04512340987','F1'),
    ('DP002','外科','04514521234','F1'),
    ('DP003','妇科','04510987654','F2'),
    ('DP004','儿科','04511234567','F2'),
    ('DP005','眼科','04518798729','F3'),
    ('DP006','皮肤科','04512946842','F3'),
    ('DP007','口腔科','04512049537','F4'),
    ('DP008','耳鼻喉科','04518478883','F4'),
    ('DP009','中医科','04519473395','F5'),
    ('DP010','神经科','04518469083','F5'),
    ('DP011','体检科','04514378605','F6'),
    ('DP012','男科','04519074093','F6');

"""
cursor.execute(sql)
```

图 107 创建基本表和增添数据

4 系统实施

4.1 系统实施环境

医院门诊管理系统是一个集合了患者以及医生资料数据的庞大数据库，能够利用数据实现效用最大化，提高工作效率和减轻工作人员工作量。数据的采集和应用需要建立在一个良好的环境才能更好地实现。医院门诊信息系统在以下环境中能够更好地实现：

1. 完整的设备及良好的网络环境

医院门诊信息系统的基本需求就是依据系统设计中信息系统的硬件结构和软件结构购置相应的硬件设备和系统软件包括操作系统、数据库系统和开发工具。

2. 人员培训

医院门诊信息系统的运行和管理还需要人的管理、控制及维护。程序设计员需要进行专门的系统软件培训。而在管理信息系统投入使用之前，还需要对一大批未来系统的使用人员进行培训，包括系统操作员、系统维护人员等。

3. 系统能够切换并交付使用

系统的切换包括进行基本数据的准备、数据的编码、系统的参数设置、初始数据的录入等多项工作。在系统正式交付使用之前，必须进行一段时间的试运行，以进一步发现及更正系统存在的问题。在过程中，需要大量人员及涉及到多个业务部门。

4.2 数据获取

本次医院门诊管理系统项目的数据是从网上搜集的，本小组并没有从单个数据集里获取医院门诊的数据，我们确定了各个表的属性并在创建基本表之后，在网上搜查资料并自行填写进基本表里。专业名词的数据，我们皆从网上搜集，比如：科室名称、病名称、药品名称等。而编号、联系方式、姓名等数据皆自行编写进基本表里。

4.3 数据预处理

数据的质量直接决定了模型的预测和泛化能力的好坏。它涉及很多因素，包括准确性、完整性、一致性、时效性、可信性和解释性。而在真实数据中，我们拿到的数据可能包含了大量的缺失值，也可能因为人工录入错误导致有异常点存在。数据清洗的结果是对各种脏数据进行对应方式的处理，得到标准的、干净的、连续的数据，提供给数据统计、数据挖掘等使用。医院门诊信息系统数据预处理的主要步骤分为：数据清理、数据集成及数据匿名化。

1. 数据清理

数据清理的意思是删除重复信息、纠正存在的错误，处理无效值和缺失值并提供数据一致性。需要进行清理的数据大多数都是残缺数据、错误数据及重复数据。残缺数据在医院门诊信息系统是很常见的，这一类数据主要是一些应该有的信息缺失，如患者地址、联系方式等。残缺数据可以经过计算机过滤出来，并要求患者重新提交，再写入数据库以补全信息数据。

错误数据产生的原因是业务系统不够健全，在接收输入后没有进行判断直接写入后台数据库造成的。在医院门诊信息系统也很常见，例如日期格式不正确、日期越界、数值数据输入成全角数字字符等。这一类错误需要进行修正再抽取写入，否则可能导致系统运行失败。第三个是重复数据。在医院门诊信息系统中，维表时常会出现重复数据，例如患者重复填写了注册表格等。这一类错误数据需要确认更改再整理就好了。数据清洗是一个反复的过程，只有不断的发现问题，再解决问题。

2. 数据集成

在医院门诊信息系统中，由于开发时间或开发部门的不同，数据往往有多个异构且运行在不同的软硬件平台上的信息系统同时运行。这些系统的数据源彼此独立、相互封闭，使得数据难以在系统之间交流、共享和融合，从而形成了"信息孤岛"。因此，共享信息在医院门诊信息系统是非常重要的。表结构，表间关系，编码的含义都必须明确公开，程序员才可以把医院门诊信息系统里的所有有关联的数据进行集成来达成信息共享，信息最有效化的目的。

3. 数据匿名化

数据匿名化是为了更好地处理医院及患者信息的隐私问题，通过对这部分数据进行匿名处理，来保证数据的隐秘性与安全性，以防数据的泄露为医院带来经营风险，或是给患者的治疗造成安全隐患。

数据预处理在这类拥有庞大数据信息的系统扮演着重要的角色。有效的数据预处理可以为医院带来了新的发展方向。医院门诊信息系统本身拥有各类数据中的无价值数据、恶意数据及失效数据的存在，因此，强化数据的有效管理与处理，并重视相应技术的开发与应用是必要的。

4.4 系统实施效果（截图说明）

本数据库使用 MYSQL，操作系统使用 Windows10。系统的面向对象主要是非计算机专业人员，因此以简便易操作为核心理念来设计。管理员通过增、删、改来确保信息资料的准确性，实现了职员和患者查询的高效性，很好地达到信息的流通共享。通过这一系列实施，门诊管理水平方面显著大程度提高，不单能减少医务人员的工作量，提高医务人员的工作质量并且提高患者的就诊便利。

图 108 增加信息

图 109 删除信息

图 110 修改信息

5 总结与展望

随着数据时代及医院门诊数量逐渐增加的发展，本组认为医院门诊信息系统的发展趋势是必不可少的。可想而知,医院门诊信息系统可以为医护人员以及患者带来便利，例如提高管理水平、节省人力等，也为未来数字信息发展有极大的帮助。本组希望医院门诊信息系统能在未来被各个大小医院门诊使用，也希望在数字化时代里，每个人都能享受数字化所来带的便利。

本组通过这次报告学习了很多关于 SQL Server 的知识，学习从零开始建立数据库，加深了对于 SQL 的理解，深感不易。本小组在此次的报告中建立了最基本的数据库，竭尽所能运用自己所学知识去完善一个数据库信息管理系统。此报告需要运用高级语言完成界面实现，本小组选择了采用 Python 的 Tkinter 来实现界面。遗憾的是，本小组成员在这方面知识尚浅、没有经验，需要在过程中慢慢摸索，导致系统功能的完整性和完备性不足，无法很好的实现界面。