

同濟大學

TONGJI UNIVERSITY

《数据库系统原理》

大作业报告

作业名称

城市医疗信息管理系统

姓 名

吴达鹏 1751118

学院（系）

电子与信息工程学院

专 业

计算机科学与技术

任课教师

李文根

日 期

2023 年 6 月 4 日

目录

《数据库系统原理》	1
大作业报告	1
一、 需求分析	4
(一) 功能需求	5
1、 病人信息查询	5
2、 病人预约医生	5
3、 病例信息提供	5
(二) 平台需求	6
(三) 性能需求	6
1、 实时性	6
2、 安全性	6
3、 准确性	6
(四) 维护需求	6
二、 可行性分析:	6
(一) 技术可行性:	6
(二) 应用可行性:	7
三、 概念设计	8
(三) 模块分割	8
(四) 设计 E-R 图	8
(五) 关系说明	10
1、 用户模块	10
2、 病历模块	10
3、 治疗模块	10
4、 机构模块	10
四、 逻辑设计	11
(一) 关系模式	11
(二) 关系表	11
(三) 完整性约束、表之间的关系、模块间的关系	11
(四) 数据字典	13
(五) 关系模式的优化及规范化设计:	18

五、 物理设计	19
(一) 选择合适的 DBMS	19
(二) 定义数据库, 表及字段的命名规范	19
(三) 字段类型的选择	20
(四) 建立合适的索引	20
(五) 采用缓存机制	22
(六) 安全性和权限控制	22
六、 项目管理	23
(一) 框架选择	23
(二) 开发平台	24
七、 系统实现	25
(一) 系统架构搭建	25
(二) 系统的功能实现逻辑原理	26
(三) 系统编写过程	26
(四) 系统功能测试	26
八、 总结	48

摘要

本项目基于上学期设计的《看病就诊及医疗信息管理系统》。立项时正值疫情期间，疫情防控占用了大量的医疗资源，许多医院在接收病人或病人问诊时，要求 48 小时内的核酸报告，当天做的核酸又不能马上得到结果。百姓有病难医的问题日益凸显，为了方便患者有病能够及早得到治疗，方便医疗机构更好的实现医疗就诊信息管理，从而实现病人预约——医生看病——结果反馈整个就诊流程所产生信息存储、查询、修改等，设计该数据库。但很快全国迎来了全面解封，很多模块存在的意义大幅下降，于是在上学期的基础上进行了简化。仍想做医疗主题的原因是，虽然疫情结束了，但是医疗同爱情、人文一样，永远是时代发展的主题，任何年代都有他存在的价值。故将项目改名为《城市医疗信息管理系统》，去除了之前预约、看病、核酸检测等模块，重点放在了城市甚至全国的医疗信息管理。比如，提供全国的医院、药房的药品、医生、患者的查询，以及各种病例的信息查询功能，小至生活中常见的感冒、发烧等引起的不适，大致各种疑难杂症，信息保存在该数据库中，供有需要的用户查询使用。通过这种方式，可以减少去医院看病的人数，可以为医生提供一些病症的参考，可以在一定程度上减轻医院的负载，将医院有限的医疗资源提供给更需要的人。

技术上，本项目后端采用 springboot+mybatis-plus 架构，前端使用 html、css、vue.js，数据库实现使用 mysql，实现了 redis 缓存、简单的事务控制以及主从库分离，可在一定程度上保证操作的正确性以及较高的查询的速度和效率。正文部分，主要是项目及数据库表设计的内容。

关键字：实体、联系、ER 图、数据字典等。

一、需求分析

全国各大大小小的医院病人、病房经常是人满为患，普通病患就医也需要花很长的时间去排队挂号、问诊、拿药、付费，并且，大多数患者求医都相对盲目，没有提前了解医院/医生的信息就乱投医，导致“不对症下药”，治疗达不到理想的效果。为了解决以上存在的问题，一方面，可以优化医院的看病服务，减少排队时间，一方面，可以减少患者不必要的挂号问诊，减少排队人数。为了实现第二方面，系统需要为患者提供真实、可靠、可信的城市医疗信息，包括但不限于城市医院信息、医院的医生信息、医院的药品信息及登记在册的病例信息，供有需要的患者查询参考，对症投医或暂不就诊。同时，系统存储的病例信息、医院、医生信息也可供医生作参考、了解、学习，丰富自己的视野、提升自己的医疗水准、与其他医生交流专业知识等等。总之，系统存储的信息，能够为医生、患者双方都提供有用的服务。

（一）功能需求

1、病人信息查询

生活中，如果病人发现自身有异样，不清楚原因的情况下，可以先在该数据库中搜索是否存在相似病例，根据病例的描述预估该病的严重程度，如果不严重（吃药可以解决），可以根据病例的处方，到药房、小门诊、或家中备药，找到相应的药品，减少医院排队问诊的人数。如果情况比较严重，也可在系统中搜索该方向较为有名的、或评价较好的医生，定点投医。

2、病人预约医生

病人可根据医生查询上班时间表、打电话等方式，在线上填写预约表单，完成预约，医生根据预约信息坐诊，如果提供线上缴费功能，可减少到医院大厅挂号、结账排队的等待时间。

3、病例信息提供

该数据库旨在存储有价值的病例的发现、诊断、治疗方式/处方、治疗结果，主治医生等信息，供其他患者或医生或有需求的人士查询。对于一些已经被治愈的疑难杂症，也可将病历信息公开（对医生、患者信息作保密处理，需要时可申请联系、沟通），供其他医生学习、了解、参考。这样，当医生再遇到类似的病例时，怎么治疗？需要用到什么医疗设备？可以做一个参考借鉴，也可以及时找到解决过该病例的主治医生交流治疗方式。

（二）平台需求

支持 Windows 10 平台即可。

（三）性能需求

1、实时性

对任意细节的修改影响属性时，要求属性能够瞬时响应（保守估计响应时间需小于 0.1s），因为各城市同时在线的病人、医生可能很多，要求查询速度快，页面响应及时，在数据表设计时应纳入考虑。

2、安全性

（1）对任意具有约束的数据项，需要保证高级程序与底层数据库的二重保障，即在高级程序中需要对错误的数据进行判断并给出提示；假设存在错误数据传递给数据库，当数据库拒绝接受时，高级程序也应捕获错误并回传提示。

（2）对于病人或医生的具体部分个人信息展示（如电话、住址、病历等），需得到其本人的同意，否则，当其他用户查询这部分信息时，需要将其隐藏，确保用户信息不泄露。

3、准确性

对于每一种病历信息的存储，应尽可能保证它的准确性，才具有一定的参考价值。如果大部分数据和信息都不准确，将使平台失去信服力，无法发挥作用。这一点需要从医生角度来保证，一方面，要核实从医资格证，一方面，可通过设置填写病历、病历核查等步骤来实现，只有通过了核查的病历及治疗方式的记录，才会出现在系统中。

（四）维护需求

- 1、设置管理员账户可对系统内容进行动态调整；
- 2、系统的设计要为后续升级系统考虑；
- 3、代码可读性强、模块清晰明确。

二、可行性分析：

（一）技术可行性：

1、技术基础设施：经初步评估，技术基础设施包括充足的硬件设备、高速网络和可靠的数据库系统，能够支持城市医疗信息管理系统的部署和运行。

2、系统集成：在与医院信息系统和其他关键系统的集成方面，存在一定的复杂性，但与相关供应商的沟通和技术支持下，集成是可行的。

3、数据安全性：对于医疗信息管理系统，数据安全性至关重要。将采取合适的隐私保护措施、数据加密技术和访问控制，确保医疗信息的安全性和合规性。

4、可扩展性：考虑到未来系统的发展和扩展需求，将采用可扩展的架构和技术方案，以满足不断增长的用户和数据量，保证系统的可持续运营和扩展能力。

5、技术支持：后端开发使用 `springboot` 框架，前端使用 `html`、`css`、`vue.js`，数据库使用 `mysql + redis`，如果需要配置主从库，可以考虑使用虚拟机。

（二）应用可行性：

1、用户需求：通过现实经验及与周围朋友、亲戚的沟通，对目标用户的需求有了初步了解，并将根据其反馈进行系统定制和功能开发，以提供更好的医疗信息管理体验。

2、组织适应性：在系统部署过程中，将与医疗机构密切合作，提供培训和变革管理支持，确保组织能够适应系统的引入和调整相关业务流程。

3、可接受性：经过初步调研，发现医疗专业人员对于引入城市医疗信息管理系统持有积极态度，患者也对其接受度较高。此外，系统的设计将遵循相关法规和政策，以确保合规性。

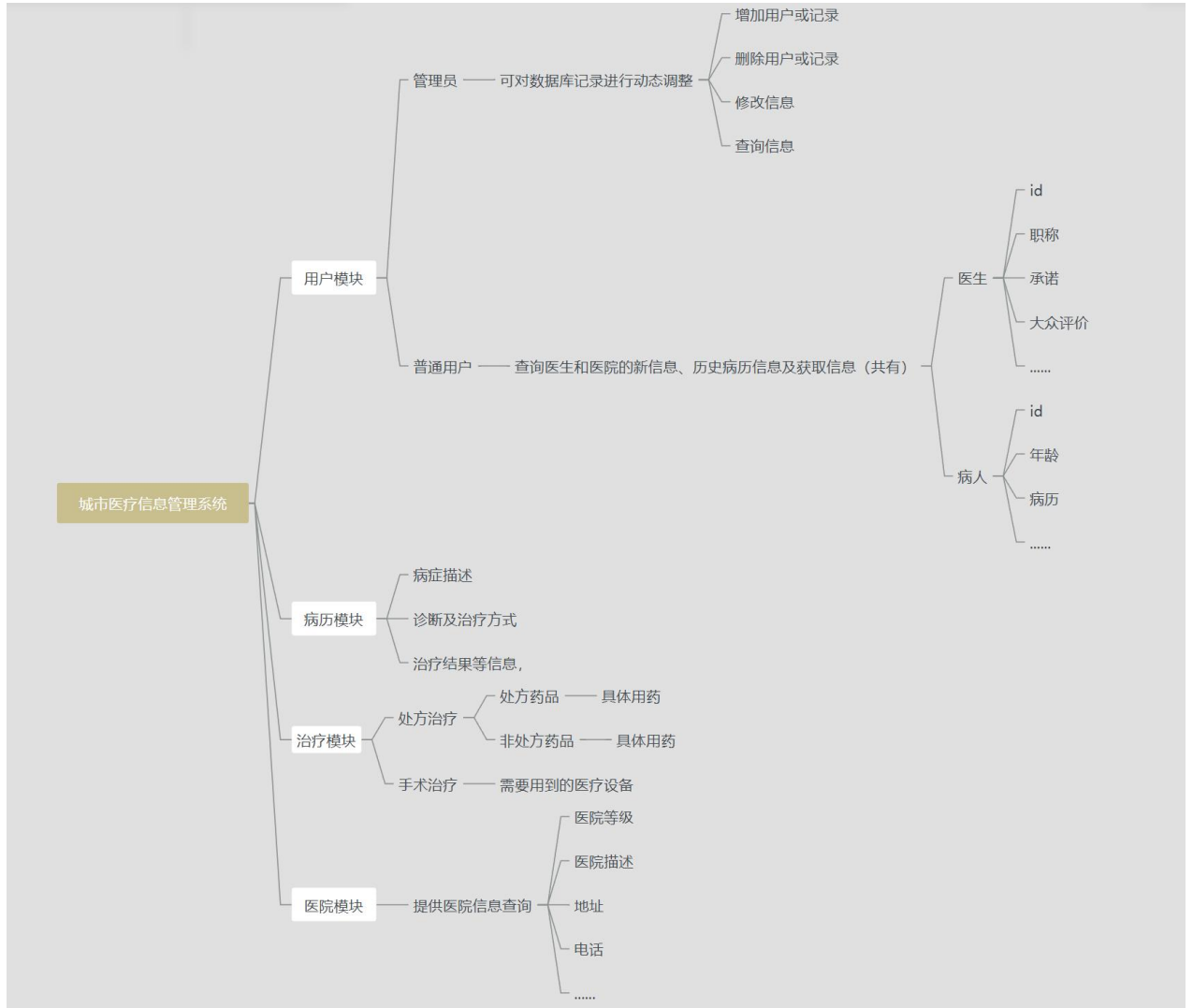
4、经济可行性：根据经济效益分析，我们预测城市医疗信息管理系统将带来明显的投资回报和成本效益。在长期运营成本方面，我们将优化系统的资源利用和维护成本，确保在经济上可行。

5、法律合规性：将严格遵守医疗行业的法律、法规和标准要求，特别关注数据隐私保护、电子健康记录管理等方面的合规性，确保系统在法律层面的合法性和合规性。

三、概念设计

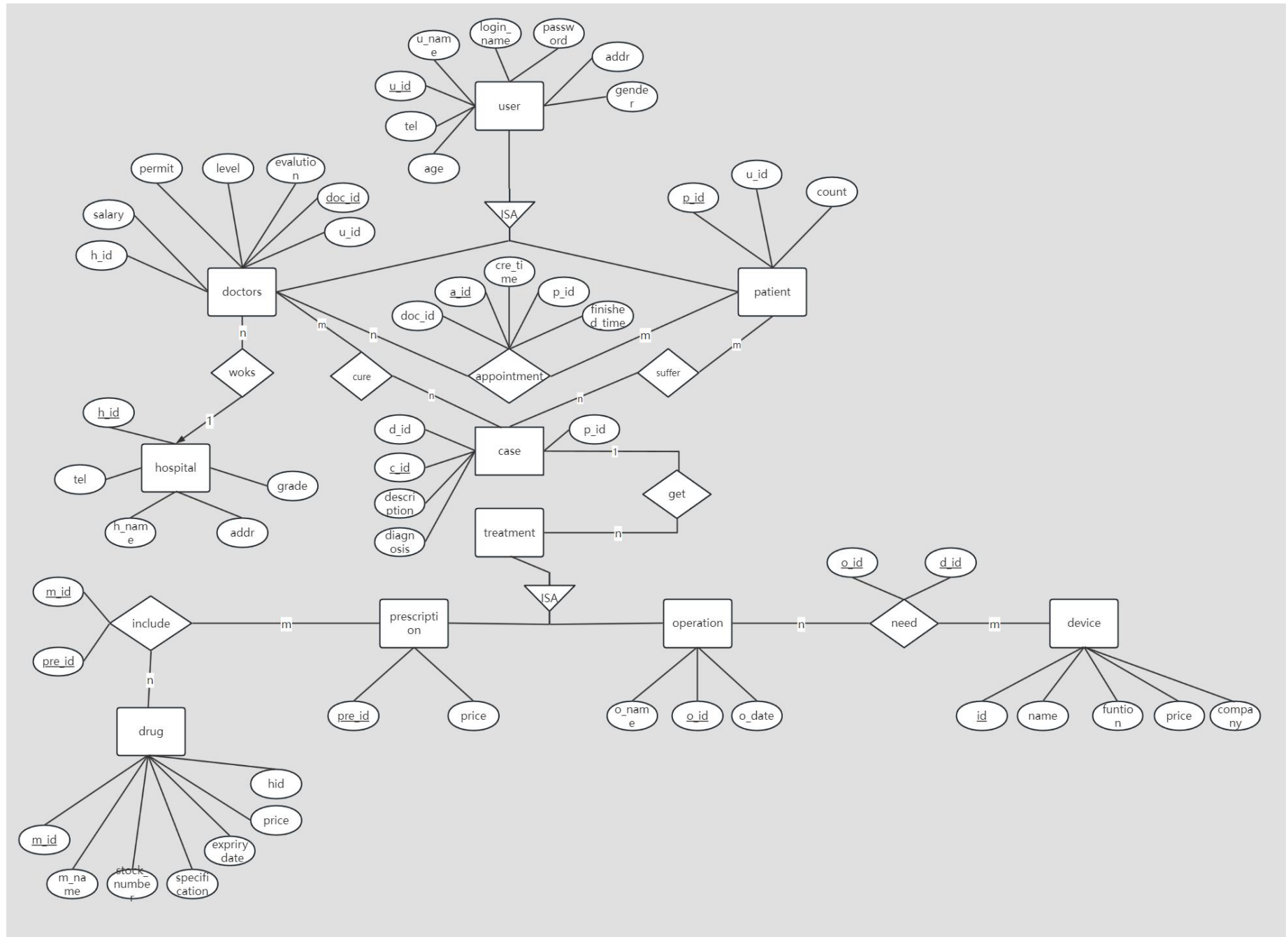
（三）模块分割

大致将系统分为用户、病历、医院、预约四个模块，具体分割如下：



（四）设计 E-R 图

根据需求分析、模块分割，设计 E-R 图如下：



（五）关系说明

1、用户模块

User 表统一管理用户，将用户类型分为医生和病人（由于管理员权限等级区别与病人和医生，管理员单独一个表），实体间的关系为预约（appointment），主要用于病人和医生间的预约就诊，一位医生可以接收多位病人的预约，一位病人也可预约多位医生，属于多对多的关系。

2、病历模块

提供病历信息，一个病人可能存在多次病历，一种病历也可能来自多个病人，对于每一种病历，记录了对应的病人 id、医生 id，病例的症状和医生诊断结果，病人和医生间通过一次病例、病例的诊疗相互关联，属于多对多关系。

3、治疗模块

一次病例有其主要的治疗方式，分为处方治疗和手术治疗，如果是处方治疗，需要有处方信息（药物及用量），如果是手术，可提供手术难度、使用的主要医疗设备、结果等信息。

4、机构模块

主要提供医院的咨询服务，医生依赖于医院而存在，具体信息详见 E-R 图。

四、逻辑设计

（一）关系模式

将 E-R 图转化为关系模式。具体步骤如下：

1、将各个实体的名字转换为各个关系模式的名字

E-R 图中所有实体包括 user、doctor、patient、hospital、cases、drugs、prescription、operation、device，实体间多对多的关系包括 appointment、include、need 共 12 个将其全部转化为关系模式。

2、实体属性就是关系的属性，实体的码就是关系的码

```
Admin=(id, username, password)
User=(u_id, u_name, login_name, password, age, tel, addr, gender, user_type)
Doctor=(d_id, salary, permit, level, evaluation, h_id, uid)
Patient=(p_id, name, address, create_time, uid)
Hospital=(h_id, h_name, grade, addr, tel)
Cases=(c_id, p_id, d_id, diagnosis)
Prescription=(pre_id, price)
Drug=(m_id, m_name, stock_number, specification, expiry_date, price)
Operation=(o_id, o_name, o_date, result)
Device(d_id, d_name, function, pur_price, company)
```

属性和码均已在 E-R 中展示，体现到关系模式中。

3、实体间联系的转换

1 对 1 联系：在任意一方加入对方的主码并设为其外码，并加入联系本身的属性。

1 对 n 联系：将 1 方的主码加入 n 方作为外码，同时将联系的属性加入 n 方。

n 对 m 联系：将联系本身转换为一个关系模式，将联系双方的主码加入其中设为码，并将联系的属性也加入其中。

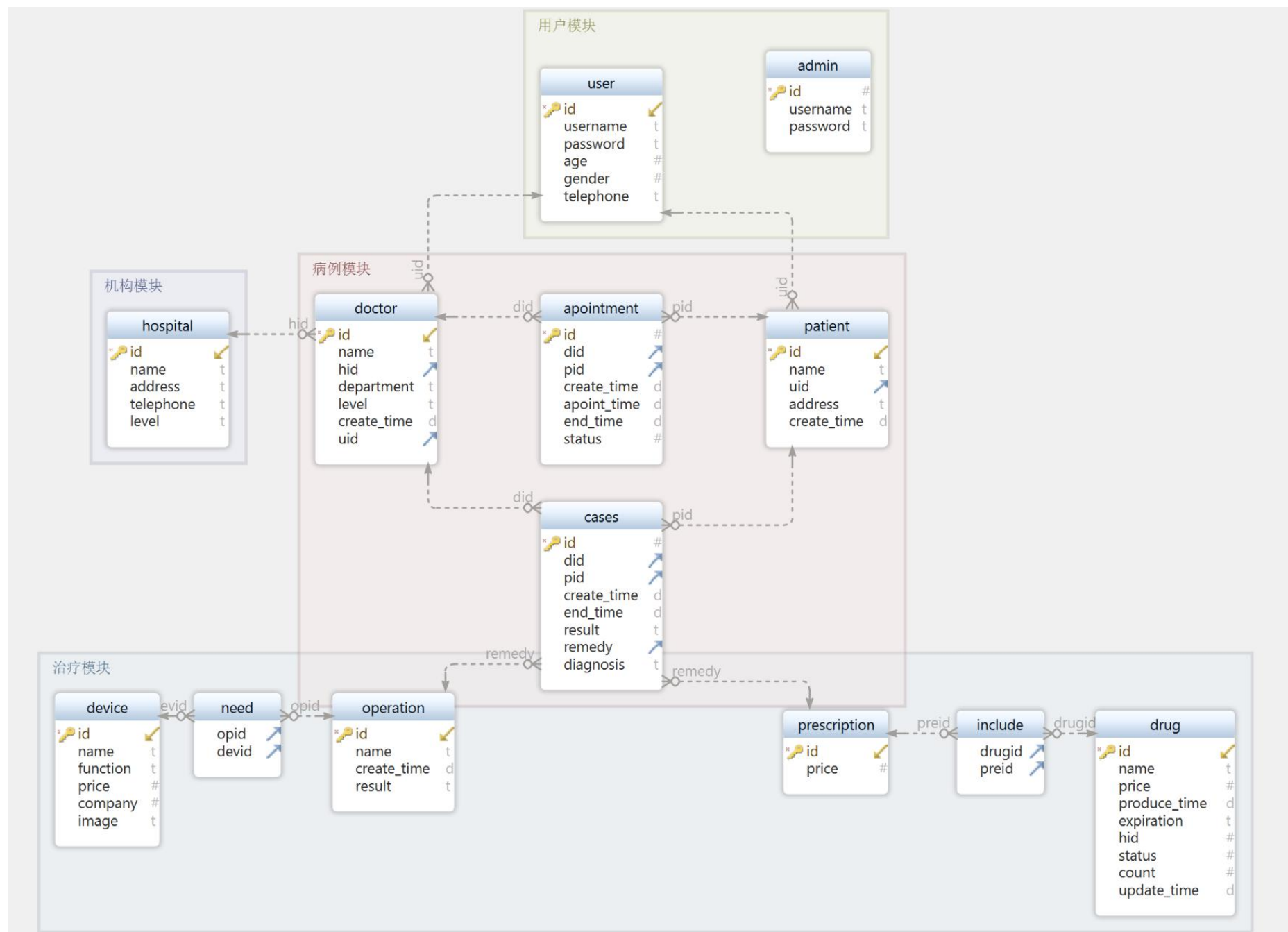
前两步在设计 E-R 图时，已将相关的外码属性添加，现只需将 n 对 m 的联系单独转换为关系模式，包括 appointment、include、need 共 3 个，设计如下：

（二）联系表

```
Apointment=(id, create_time, appoint_time, end_time, status, did, pid)
Include=(m_id, pre_id)
Need(o_id, d_id)
```

（三）完整性约束、表之间的关系、模块间的关系

用 DBSchema 将关系模式转化为关系表，添加外键、非空等完整性约束，并将各表模块化展示：



（四）数据字典

实体+关系共 12 个关系模式。表结构设计和完整性约束如下：

Admin:

属性名	类型	约束	备注
id	int	primary key	管理员的唯一标识
username	varchar(20)	unique	登录用户名
password	varchar(20)		密码

User:

属性名	类型	约束	备注
id	int	primary key	用户的唯一标识
username	varchar(20)	unique	登录用户名
password	varchar(20)		密码
gender	char(1)	check '男' or '女'	性别
age	int	not null	年龄
telephone	varchar(20)		电话

Doctor:

属性名	类型	约束	备注
id	int	primary key;auto increment	医生的唯一标识
u_id	int	foreign key references user.id	用户 id
h_id	int	foreign key references hospital.h_id	医院 id
department	varchar(100)		部门

属性名	类型	约束	备注
level	varchar(10)	not null	职称等级
Create_time	datetime		注册时间

Patient:

属性名	类型	约束	备注
id	int	primary key ; auto increment	病人的唯一标识
u_id	int	foreign key references user.id	用户 id
name	Varchar		姓名
Address	Varchar		地址
Create_time	Datetime		注册时间

Appointment:

属性名	类型	约束	备注
Id	int	primary key;auto increment	一次预约的唯一标识
d_id	int	foreign key references doctor.doc_id	医生 id
p_id	int	foreign key references patient.p_id	病人 id
create_time	datetime		创建时间
Apoint_time	Datetime		预约时间
End_time	Datetime		结束时间
Status	Int	Check '1 ' or '0'	预约状态

Hospital:

属性名	类型	约束	备注
id	int	primary key; auto increment	医院的唯一标识
name	varchar(20)	not null	医院名字
Address	varchar(20)		地址
telephone	varchar(20)		电话
level	varchar(10)		医院等级

cases（病例）：

属性名	类型	约束	备注
id	int	primary key; auto increment	一个病例的唯一标识
Did	Int	foreign key references doctor.id	医生 id
Pid	Int	foreign key references patient.id	病人 id
description	varchar(100)	not null	病例描述
diagnosis	varchar(50)	not null	医生诊断
Create_time	Datetime		创建时间
End_time	Datetime		结束时间
Result	Varchar		结果
Remedy	Bigint	foreign key references prescription.id or operation.id	治疗方式

prescription（处方）：

属性名	类型	约束	备注
id	int	primary key; auto increment	唯一标识
price	Double		处方总额

include（某个处方下的所有药品）：

属性名	类型	约束	备注
drugid	int	primary key;foreign key references drug.id	药品 id，联合主键
preid	int	primary key;foreign key references prescription.pre_id	处方 id，联合主键
number	int	not null,check >0	数量

drug（总药品）：

属性名	类型	约束	备注
Id	int	primary key; auto increment	药品的唯一标识
name	varchar(20)	unique not null	药品的名字
price	double		价格
Produce_time	Date		生产日期
Expiration	Varchar(10)		保质期
Status	int	Check 1 or 0	售卖状态
Hid	Int	foreign key references hospital.id	所属医院
Count	Int	Check count > 0	剩余数量
Update_time	Int		最后更新时间

operation（手术）：

属性名	类型名	约束	备注
id	int	primary key ; auto increment	一次手术的唯一标识
name	varchar(20)	not null	手术名称
Create_time	date		手术日期
Result	int	Check 1 or 0	手术结果

need（某一台手术需要的设备）：

属性名	类型名	约束	备注
opid	int	primary key;foreign key references operation.o_id	手术 id，联合主键
devid	int	primary key;foreign key references device.d_id	设备 id，联合主键

device（设备）：

属性名	类型	约束	备注
id	int	primary key ; auto increment	设备 id
name	varchar(20)	not null	设备名字
funtion	varchar(50)		功能
price	int		购置价格
Company	Varchar(50)		出厂公司
Image	Varchar(100)		图片路径

(五) 关系模式的优化及规范化设计:

关于第 3NF, 在设计关系模式时已经考虑:

(1) 不包含多值属性, 不含有部分依赖;

(2) 不包含传递依赖;

故本系统的设计是满足第 3NF 的。

五、物理设计

一般地，数据库的物理设计是关于如何在计算机存储设备上组织和存储数据的过程。它涉及以下内容：

存储结构：确定如何将数据存储在物理存储介质上，例如硬盘或固态驱动器。这包括选择适当的存储格式、文件组织方式和存储引擎。

表和索引设计：确定如何组织表和索引以提高查询性能。这包括选择适当的数据类型、字段长度和约束，以及定义主键、外键和索引。

数据分区和分片：根据性能和可用性需求，将数据分割成多个分区或分片。这可以提高查询和维护操作的效率，并支持数据的水平扩展。

数据复制和备份策略：确定数据复制和备份的方式，以确保数据的可靠性和容灾能力。这包括选择适当的复制策略和备份频率，以及定义恢复和故障转移策略。

缓存和缓冲区管理：决定如何使用缓存和缓冲区来优化数据库访问。这包括设置适当的缓存大小、缓冲区管理算法和数据预读取策略。

安全性和权限控制：定义数据库的安全性措施，包括访问控制、用户权限和数据加密等。这确保只有授权的用户能够访问和修改数据。

性能调优：通过优化查询、索引和存储结构等方面，提高数据库的性能和响应时间。这包括监控和调整数据库参数、优化查询语句和重建索引等操作。

容量规划：估计数据库的存储需求，并根据数据的增长预测进行容量规划。这包括确定存储设备的容量和扩展计划，以满足未来的需求。

数据库的物理设计需要考虑多个因素，包括性能、可用性、安全性和可维护性等。它需要综合考虑数据库管理系统的功能和特性，以及特定应用的需求和约束、索引，以确保数据库在物理层面上的高效和可靠运行。鉴于以上物理设计的特点，选择部分作以下设计：

（一）选择合适的 DBMS

由于本项目未经过更加专业考察、调研、以及数据库结构设计，仅为本人第一次数据库设计的尝试，选用本人熟悉的、互联网级别的数据库管理系统 mysql，存储引擎选用默认的 InnoDB。

（二）定义数据库，表及字段的命名规范

主要根据两个原则：

- 1、可读性原则。通过下划线格式化名字。如 `cust_address` 而不是 `custaddress`。
- 2、表意性原则。见名知意，如表的过程应该能体现存储的数据内容。

（三）字段类型的选择

数据类型一方面影响数据存储空间的开销，另一方面也会影响数据查询性能。

当一个列可以选择多种数据类型时，应该优先选择数字类型，其次是日期或二进制类型，最后是字符类型。

对于相同级别的数据类型，应该优先选择占用空间小的数据类型。

以上选择原则主要是从以下两个角度考虑：

- 1、在对数据进行比较（查询条件，JOIN 条件及排序）操作时，同样的数据，字符处理往往比数字处理慢。
- 2、在数据库中，数据处理以页为单位，列的长度越小，利于性能提升。

（四）建立合适的索引

索引一定要建立在查询更快、占用空间更小的基础上建立。通过上网查询资料，总结了索引设计原则：

- 1、针对于数据量较大，且查询频繁的表建立索引。
- 2、针对于常作为查询条件 where、排序 order by、分组 group by 操作的字段建立索引。
- 3、尽量选择区分度高的列作为索引，尽量建立唯一索引，区分度越高，索引的效率越高。
- 4、如果是字符串类型的字段，字段的长度较长，可以针对于字段的特点，建立前缀索引。
- 5、尽量使用联合索引，减少单列索引，查询时，联合索引很多时间可以覆盖索引，节省存储空间，避免回表，提高查询效率。
- 6、要控制索引的数量，索引并不是多多益善，索引越多，维护索引结构的代价也就越大，会影响增删改查的效率。
- 7、如果索引列不能存储 NULL 值，在创建表时使用 NOT NULL 进行约束。当优化器知道每列是否包含 NULL 值时，它可以更好地确定哪个索引最有效地用于查询。

基于以上原则，选择建立索引如下：

- 1、各表的主键均建立索引，单一主键建立索引，联合主键建立联合索引，为聚集索引；

2、其他需要特殊考虑的索引如下表：

表名	属性	索引类型	选择原因
Doctor	u_id	唯一索引	作为外键查询 User 的信息
Doctor	level	常规索引	可能会经常要通过医生职称等级来筛选医生信息
Patient	u_id	唯一索引	作为外键查询 User 的信息
Hospital	grade	常规索引	可能会经常要通过医院级别来筛选医院信息
Hospital	Address	常规索引	病人可能经常筛选附近的医院
Operation	Create_time	唯一索引	可能会经常通过手术时间来查询某一台手术
Appointment	doc_id,p_id	联合索引	作为外键查询医生、病人信息
Appointment	Status	常规索引	医生可能会经常查看自己未完成的预约
Cases	Diagnosis	常规索引	病人可能经常筛选和自己诊断情况相同的病例信息进行查看
Cases	Create_time End_time	联合索引	可能经常一个时间段内的病例查看信息
Drug	Status	普通索引	病人可能经常筛选在售卖状态的药品查看信息

（五）采用缓存机制

这里使用 nosql 型数据库 redis 作为缓存。

Redis（Remote Dictionary Server）是一种内存中的数据存储系统，它可以作为缓存层和 MySQL 数据库之间的中间件，用于提高数据库的效率。下面是 Redis 帮助 MySQL 作为缓存提高数据库效率的几种方式：

1、减轻数据库负载：通过将常用的查询结果和数据存储在 Redis 的内存中，可以减轻 MySQL 数据库的负载。当应用程序需要获取数据时，可以首先从 Redis 中查询，如果缓存中存在数据，则无需访问 MySQL 数据库，从而降低数据库的压力。

2、加快数据访问速度：由于 Redis 将数据存储在内存中，相比于 MySQL 的磁盘存储，Redis 具有更快的读取速度。通过将热门数据、频繁查询的结果或复杂计算的中间结果存储在 Redis 中，可以显著加快数据访问速度，提高应用程序的响应性能。

3、缓解数据库瓶颈：对于一些需要频繁查询或计算的场景，MySQL 可能成为性能瓶颈。通过将这些热点数据存储在 Redis 中，可以将部分查询或计算从 MySQL 转移到 Redis 中完成，从而减少对 MySQL 的压力，提高系统的并发性能。

4、实时数据处理：对于需要实时数据处理和分析的场景，Redis 可以作为数据的缓冲区，快速接收和存储数据，然后批量写入 MySQL 进行持久化。这种方式可以有效降低对 MySQL 的实时写入压力，保证系统的实时性能。

需要注意的是，Redis 作为缓存层时需要确保缓存的数据与数据库中的数据保持一致性，通常使用合适的缓存策略和缓存失效机制来管理数据的更新和过期。本系统使用 springboot 作为后端服务器，可以很方便地整合 redis。

（六）安全性和权限控制

对用户的密码使用 MD5 算法加密，以便保护数据库数据，通过直接访问数据库是无法获得正确的登录密码的。

六、项目管理

（一）框架选择

后端：springboot2

前端：html、css、vue.js

数据库：mysql + redis

持久层：mybatis-plus

选择原因：

1、快速开发：Spring Boot 提供了一种快速构建应用程序的方式，通过自动配置和约定大于配置的原则，可以快速搭建项目的基础结构，减少了开发人员的工作量，提高了开发效率。

2、简化配置：Spring Boot 通过自动配置功能，减少了繁琐的配置工作。它可以根据项目中使用的依赖和组件自动配置相关的功能，简化了开发人员的配置过程，使得项目的配置更加简洁和易于管理。

3、组件丰富：Spring Boot 集成了众多的开箱即用的组件，如 Spring MVC、Spring Data JPA、Spring Security、redis 等，这些组件可以快速集成到项目中，提供常用的功能支持，减少了开发人员的重复劳动。

4、高度可扩展：Spring Boot 采用模块化的设计，组件之间松耦合，使得系统更加可扩展。开发人员可以根据项目需求选择需要的组件进行集成，也可以自定义开发组件进行扩展，从而满足不同项目的需求。

5、社区支持：Spring Boot 拥有庞大的开发者社区支持，社区中有大量的文档、教程和解决方案可供参考。开发人员可以通过社区的资源获取帮助和支持，解决问题和学习新的技术。

因为项目仅由个人进行开发，为方便调试运行和修改代码，并没有做成前后端分离的架构，而选用 html、css、vue.js。

选择 MySQL 和 Redis 作为数据库的原因如下：

MySQL：

1、成熟稳定：MySQL 是一款经过长期发展和广泛应用的关系型数据库管理系统，具有稳定性高、可靠性强的特点。它已经在各种规模的应用中得到验证，并且拥有庞大的用户社区和活跃的开发者的支持。

2、数据一致性和完整性：MySQL 作为关系型数据库，支持 ACID 事务特性，能够确保数据的一致性和完整性。它提供了强大的数据操作功能，包括事务管理、数据约束、外键关联等，可以满足大多数应用场景的数据管理需求。

3、强大的查询功能：MySQL 具有丰富的查询语言和优化器，可以对数据进行高效的查询和分析。它支持复杂的查询操作，包括联表查询、聚合函数、子查询等，能够满足各种复杂的数据查询需求。

4 可扩展性：MySQL 支持水平和垂直的扩展方式。通过主从复制和分片技术，可以将数据分布到多个服务器上，提高系统的并发处理能力和数据存储容量。

Redis:

1、高性能：Redis 是一款基于内存的键值存储数据库，具有出色的读写性能。由于数据存储在内存中，使得 Redis 能够提供低延迟的数据访问，适用于对响应速度要求较高的场景。

2、缓存功能：Redis 的主要应用场景之一是缓存。通过将热门数据存储在 Redis 中，可以大大减少对后端数据库的访问，提高系统的响应速度和并发能力。Redis 支持灵活的缓存策略和过期设置，能够满足不同业务需求。

3、数据结构丰富：Redis 支持丰富的数据结构，包括字符串、哈希表、列表、集合、有序集合等。这些数据结构的支持使得 Redis 不仅仅是一个简单的键值存储，还可以进行复杂的数据操作和计算，满足更多的业务需求。

4、持久化支持：Redis 支持数据的持久化，可以将数据存储到磁盘中，确保数据的持久性。它提供了两种持久化方式，即快照和日志，可以根据需求选择适合的方式来保证数据的安全性。

(二) 开发平台

前期全部在 windows 操作系统，使用 IDEA 开发，后期因为配置了主从库，需要多台设备，故将数据库部署到两台 linux-centos7 操作系统上。

七、系统实现

（一）系统架构搭建

一般来说，使用 `springboot` 作为后端开发框架，架构主要分为控制层（`controller`）、业务层（`service`）、数据访问层（也即持久层，`DAO/mapper`），用于实现代码的模块化和职责分离。它们各自的作用如下

1、控制层（Controller）：

（1）负责接收用户的请求，处理请求参数，调用业务层进行业务处理，并将处理结果返回给客户端。

（2）通常包含处理请求的方法（如 `GET`、`POST`、`PUT`、`DELETE` 等），并通过路由将请求映射到相应的处理方法。

（3）还可以进行请求验证、异常处理、日志记录等操作。

2、业务层（Service）：

（1）负责实现应用的核心业务逻辑，它是控制层和数据访问层之间的桥梁。

（2）处理业务逻辑，包括对数据的处理、算法的应用、业务规则的验证等。

（3）通常包含一组服务类，每个服务类负责一个或多个相关的业务功能。

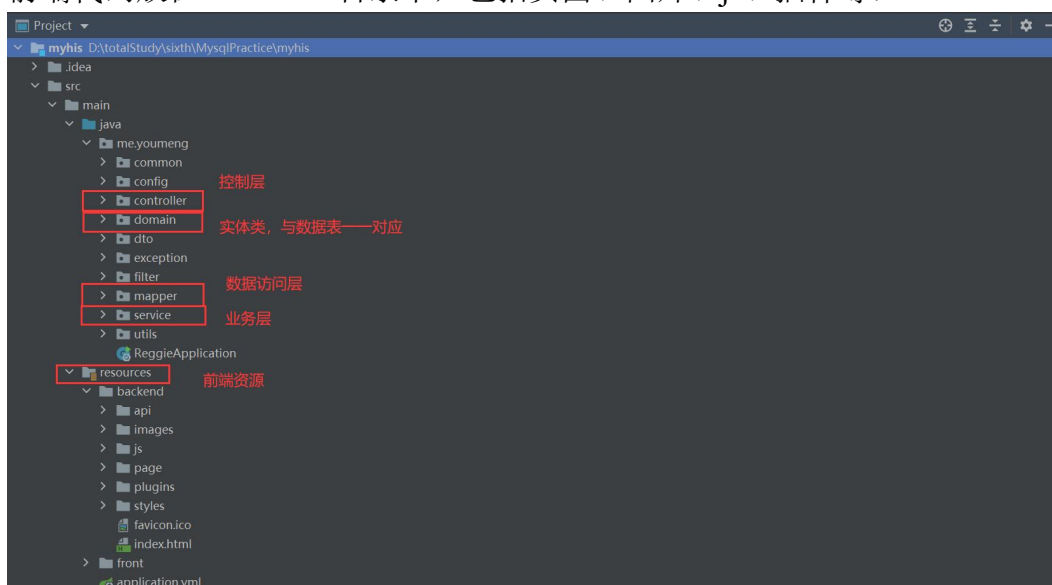
3、数据访问层（DAO）：

（1）负责与数据库或其他数据存储进行交互，包括数据的读取、写入、更新和删除等操作。

（2）封装了对数据的具体访问细节，包括 `SQL` 查询、`ORM` 映射、事务管理等。

（3）通常包含一组数据访问对象（`DAO`），每个 `DAO` 负责与一个或多个数据表或实体进行交互。

前端代码放在 `resources` 目录下，包括页面、图片、`js`、插件等：



（二）系统的功能实现逻辑原理

整个项目的入口是 `ReggieApplication` 启动类，项目启动后，可以通过 IP:端口//资源路径在网页中请求项目资源。当用户在视图表现层（即浏览器）点击按钮发送请求后，会根据按钮绑定的事件，使用 `axios` 向后端发送资源请求，后端根据对应的请求路径，到控制层（`controller`）中找到对应的请求接口，进行参数处理后，调用业务层（`service`）接口处理业务逻辑，当需要对数据进行持久化及查询操作时，会调用数据访问层接口（`mapper`），由数据访问层对数据库进行访问存储操作。得到的查询、增删改的结果原路返回，最终由控制层交由视图表现层，再由视图表现层渲染到页面上展示给用户。

（三）系统编写过程

首先，搭建整个项目的框架，使得在视图表现层的操作能够正确向后端发送请求，并且后端能接收到请求并做出响应，对数据库简单配置到本机下，确保系统逻辑能跑通即可。

而后，根据已经设计好的数据库表，建立与之一一对应的实体类，再根据用户需求，为每一个表添加增删改查功能。每一个实体类有自己的 `controller`、`mapper` 和 `service`，但均按照相同的框架来处理，仅具体的业务逻辑不同。

每添加一个请求路径（即 `controller` 中的一个函数），对该功能进行测试，是否能正确访问并返回期待的结果。

将基本功能实现后，考虑对系统的优化。一是 `springboot` 整合 `redis`，使得我们可以使用 `redis` 作为数据缓存，当大量的访问请求的资源相同时，从第二次开始可以直接从缓存中去取数据，而不必每次都查询数据库，缓解数据库压力。二是配置主从库分离，使用两台虚拟机，主库的每一次修改操作，会生成对应的日志记录，从库会根据日志记录，做同样的修改操作。主库主要负责与表修改相关的操作，如表结构更改、数据的增删改等，从库主要负责查询。并且，我们可以为主库配置多个从库，多个从库采用轮询机制担负查询任务，进一步分解查询压力，提高系统的效率。

（四）系统功能测试

1、正确性测试

主要对各模块的增删改查、模糊搜索的正确性进行测试，下面进行测试展示：

（1）病人管理

分页查询：

localhost:8080/backend/index.html

算法刷题 CSDN博客 论文检索 bilibili 求职 百度 考研 编程 北京市公共数据开... Mushroom Classif... 学生云服务平台, 学生... 蓝桥杯官网 Lab: Xv6 and Unix... 中间代码生成报告... chargeit体验

患者管理

管理员

Health+

患者管理

- 医生管理
- 医院管理
- 病例管理
- 处方管理
- 药品管理
- 手术管理
- 设备管理
- 预约管理

请输入患者姓名

+ 添加患者

患者姓名	账号	性别	年龄	手机号	住址	注册时间	操作
关羽	guanyu	男	33	18018594179	同济大学	2023-06-05 15:28:02	删除 重置
张飞	zhangfei	男	22	18018594179	四川成都	2023-06-06 14:12:02	删除 重置
刘备	liubei	男	44	18018594179	荆州	2023-06-06 14:12:40	删除 重置
大乔	dajiao	女	20	13314442555	江苏徐州	2023-06-06 14:13:11	删除 重置
小乔	xiaoqiao1	女	18	18015852563	江苏徐州	2023-06-06 14:13:38	删除 重置

共 60 条 5条/页 < 1 2 3 4 5 6 ... 12 > 前往 1 页

localhost:8080/backend/index.html

算法刷题 CSDN博客 论文检索 bilibili 求职 百度 考研 编程 北京市公共数据开... Mushroom Classif... 学生云服务平台, 学生... 蓝桥杯官网 Lab: Xv6 and Unix... 中间代码生成报告... chargeit体验

患者管理

管理员

Health+

患者管理

- 医生管理
- 医院管理
- 病例管理
- 处方管理
- 药品管理
- 手术管理
- 设备管理
- 预约管理

请输入患者姓名

+ 添加患者

患者姓名	账号	性别	年龄	手机号	住址	注册时间	操作
关羽	guanyu	男	33	18018594179	同济大学	2023-06-05 15:28:02	删除 重置
张飞	zhangfei	男	22	18018594179	四川成都	2023-06-06 14:12:02	删除 重置
刘备	liubei	男	44	18018594179	荆州	2023-06-06 14:12:40	删除 重置
大乔	dajiao	女	20	13314442555	江苏徐州	2023-06-06 14:13:11	删除 重置
小乔	xiaoqiao1	女	18	18015852563	江苏徐州	2023-06-06 14:13:38	删除 重置
毛融	Melissa Wilson	^	7	18150719689	北京 北京市	2023-06-07 14:59:26	删除 重置
韩杰	Kimberly Johnson	T	44	18605866437	吉林省 白城市	2023-06-07 14:59:26	删除 重置
邱秀兰	James Anderson	8	75	19838746775	吉林省 辽源市	2023-06-07 14:59:26	删除 重置
卢露	Jeffrey Anderson	4	98	19872015244	广东省 深圳市	2023-06-07 14:59:28	删除 重置
韩明	Lisa Hernandez	n	15	18123149366	河北省 秦皇岛市	2023-06-07 14:59:28	删除 重置

共 60 条 10条/页 < 1 2 3 4 5 6 > 前往 1 页

localhost:8080/backend/index.html

算法刷题 CSDN博客 论文检索 bilibili 求职 百度 考研 编程 北京市公共数据开... Mushroom Classif... 学生云服务平台, 学生... 蓝桥杯官网 Lab: Xv6 and Unix... 中间代码生成报告... chargeit体验

患者管理

管理员

Health+

患者管理

- 医生管理
- 医院管理
- 病例管理
- 处方管理
- 药品管理
- 手术管理
- 设备管理
- 预约管理

请输入患者姓名

+ 添加患者

患者姓名	账号	性别	年龄	手机号	住址	注册时间	操作
薛亮英	Donald Johnson	T	112	18168274866	澳门特别行政区 离岛	2023-06-07 14:59:34	删除 重置
任娜	Elizabeth Moore	@	80	18142350143	广西壮族自治区 柳州市	2023-06-07 14:59:34	删除 重置
张涛	Frank Johnson	D	86	18157828615	内蒙古自治区 乌兰察布市	2023-06-07 14:59:34	删除 重置
史刚	Joseph Brown	Z	65	18646760763	吉林省 辽源市	2023-06-07 14:59:35	删除 重置
江萍	Nancy Harris	Z	29	18174458955	青海省 海东市	2023-06-07 14:59:35	删除 重置
黎娟	Susan Rodriguez	M	83	19814977560	湖南省 娄底市	2023-06-07 14:59:35	删除 重置
史勇	Jose Harris	V	59	18655038044	湖南省 株洲市	2023-06-07 14:59:37	删除 重置
于秀兰	Kevin Miller	F	60	18154397706	河南省 开封市	2023-06-07 14:59:37	删除 重置
段军	Donna Rodriguez	O	40	13516957634	河北省 保定市	2023-06-07 14:59:37	删除 重置
郑超	Kenneth Thomas	w	97	18132111669	内蒙古自治区 赤峰市	2023-06-07 14:59:40	删除 重置

共 60 条 10条/页 < 1 2 3 4 5 6 > 前往 3 页

登录管理系统后，初始界面为“患者管理”，会发送请求查询 patient 表和 user 表，将两表数据做一个拼接后渲染到页面上，可以看到各项信息都正常，并且分页功能也正常（由于使用 apifox 测试时，测试数据没有限制好，导致性别一栏是各种字符，而不仅仅是限制于‘男’、‘女’）。

添加病人：

我们添加一个病人名为“吕奉先”

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

返回添加患者

管理员

姓名*

吕奉先

用户名*

lfengxian

密码*

lfengxian

手机号*

18018594179

家庭地址*

同济大学

年龄

30

性别

☒男☐女

取消

保存

患者管理

管理员

请输入患者姓名

添加患者

患者姓名	账号	性别	年龄	手机号	住址	注册时间	操作
吕奉先	lfengxian	男	30	18018594179	同济大学	2023-06-07 15:38:35	<div>新增删除</div>

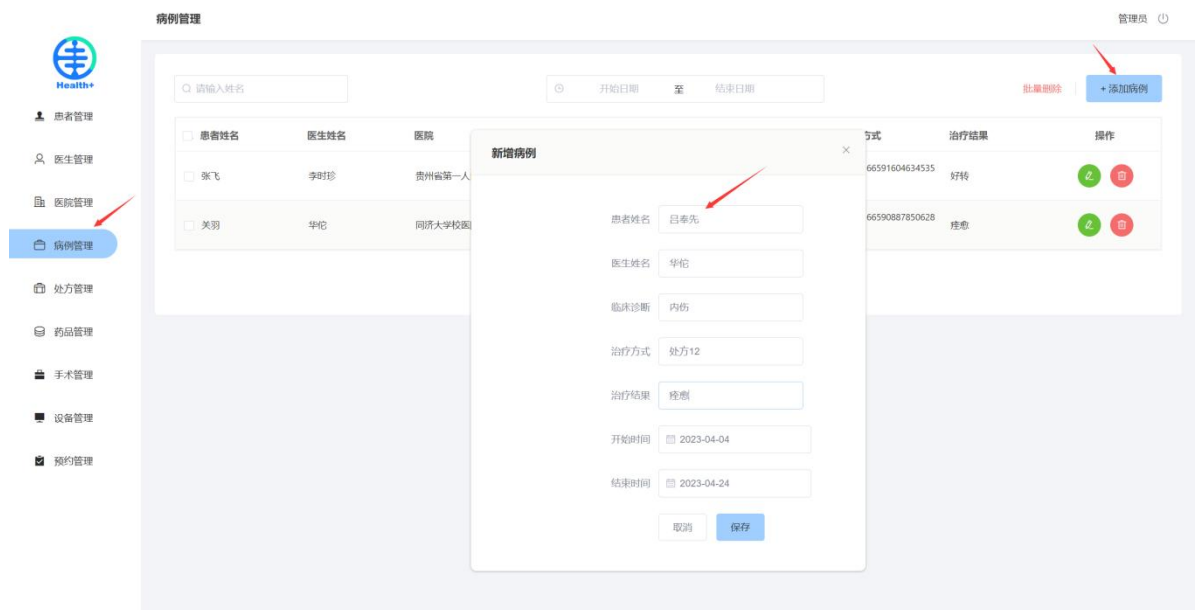
共 61 条

5条/页

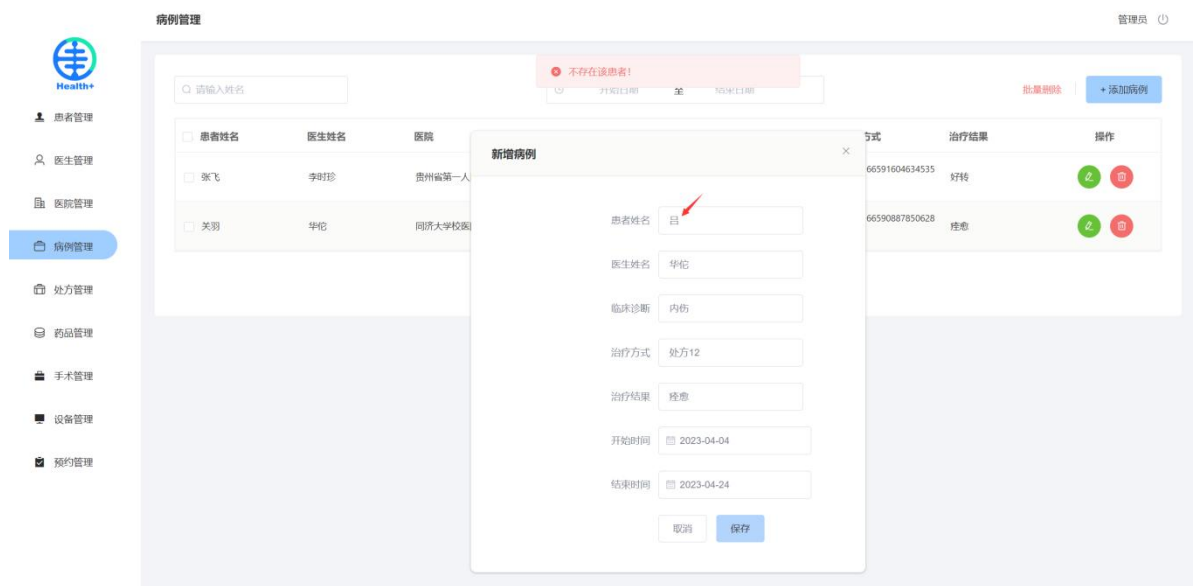
< 1 ... 8 9 10 11 12 13 >

前往 13 页

在最后一页可以看到我们刚才添加的病人信息。为了方便演示多表联查、连删，我们再添加一个关于“吕奉先”的病例和预约，分别位于“病例管理”和“预约管理”：



这里必须确保患者姓名和医生姓名都必须在患者表、医生表中能查到，否则会提示该患者/医生不存在，如下：



检查添加的病例：



同样的方式添加“吕奉先”的预约表：

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

预约管理

管理员

请输入姓名

开始日期至结束日期

批量删除

+ 添加预约

患者姓名	医生姓名	医院	预约完成状态	操作
<input type="checkbox"/> 关羽	华佗	同济大	已完成	<div>🗑️</div>
<input type="checkbox"/> 张飞	华佗	同济大	待完成	<div>🗑️</div>
<input type="checkbox"/> 大乔	李时珍	贵州	待完成	<div>🗑️</div>

新增预约

患者姓名

吕奉先

医生姓名

华佗

开始时间

2023-04-04 00:00:00

预约时间

2023-04-24 15:00:00

结束时间

选择日期

完成状态

☐ 已完成 ☒ 待完成

取消

保存

预约管理

管理员

请输入姓名

开始日期至结束日期

批量删除

+ 添加预约

🟢 预约添加成功!

患者姓名	医生姓名	医院	开始时间	预约时间	完成时间	预约完成状态	操作
<input type="checkbox"/> 关羽	华佗	同济大	2023-04-04 00:00:00	2023-06-06 15:00:00	2023-06-06 15:46:43	已完成	<div>🗑️</div>
<input type="checkbox"/> 张飞	华佗	同济大	2023-04-04 00:00:00	2023-06-06 15:00:00		待完成	<div>🗑️</div>
<input type="checkbox"/> 吕奉先	华佗	同济大	2023-04-04 00:00:00	2023-04-24 15:00:00		待完成	<div>🗑️</div>
<input type="checkbox"/> 大乔	李时珍	贵州	2023-04-04 00:00:00	2023-06-06 15:00:00		待完成	<div>🗑️</div>

共 4 条 10条/页 < 1 > 前往 1 页

删除病人：

我们删除刚才添加的“吕奉先”这个病人：

The screenshot shows the 'Patient Management' (患者管理) interface. On the left is a sidebar with navigation options: 患者管理 (selected), 医生管理, 医院管理, 病例管理, 处方管理, 药品管理, 手术管理, 设备管理, and 预约管理. The main area displays a table of patients. The first row is for '吕奉先' (Lv Fengxian) with ID 'lvfengxian', male, age 30, phone '18018594179', address '同济大学', and registration time '2023-06-07 15:38:35'. A red arrow points to the '删除' (Delete) button in the '操作' (Actions) column. A confirmation dialog box is centered on the screen with the text: '提示 此操作将永久删除该信息，是否确认？' (Warning: This operation will permanently delete this information. Are you sure?). The dialog has '取消' (Cancel) and '确定' (Confirm) buttons. A red arrow points to the '确定' button. At the top right, there is a '+ 添加患者' (Add Patient) button. The bottom of the table shows pagination: '共 61 条' (Total 61 items), '5条/页' (5 items per page), and page numbers 1 through 13.

可以到患者管理、病例管理、预约管理三张表中查看刚才添加的有关“吕奉先”的记录都已经被删除：

The screenshot shows the 'Patient Management' (患者管理) interface after the deletion. The sidebar is the same. The main area displays a table of patients. The first row is for '杨明' (Yang Ming) with ID 'Nancy Williams', female, age 59, phone '18166560644', address '贵州省黔东南苗族侗族自治州', and registration time '2023-06-07 15:26:18'. The table continues with '郑芳' (Zheng Fang), '苏霞' (Su Xia), '傅艳' (Fu Yan), and '魏杰' (Wei Jie). The '操作' (Actions) column for each row contains '删除' (Delete) and '恢复' (Restore) buttons. At the bottom of the table, the pagination shows '共 60 条' (Total 60 items), '5条/页' (5 items per page), and page numbers 1 through 12. The '吕奉先' patient is no longer present in the list.

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

病例管理

管理员

请输入姓名

开始日期至结束日期

批量删除

+ 添加病例

患者姓名	医生姓名	医院	开始时间	结束时间	临床诊断	治疗方式	治疗结果	操作
张飞	李时珍	贵州省第一人民医院	2023-04-04 00:00:00	2023-04-10 00:00:00	冠状病毒	处方166591604634535936	好转	<div>编辑</div> <div>删除</div>
关羽	华佗	同济大学医院	2020-07-30 00:00:00	2020-07-31 00:00:00	中毒	手术166590887850628710	痊愈	<div>编辑</div> <div>删除</div>

共 2 条10条/页<1>前往1页

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

预约管理

管理员

请输入姓名

开始日期至结束日期

批量删除

+ 添加预约

患者姓名	医生姓名	医院	开始时间	预约时间	完成时间	预约完成状态	操作
关羽	华佗	同济大学医院	2023-04-04 00:00:00	2023-06-06 15:00:00	2023-06-06 15:46:43	已完成	<div>编辑</div> <div>删除</div>
张飞	华佗	同济大学医院	2023-04-04 00:00:00	2023-06-06 15:00:00		待完成	<div>编辑</div> <div>删除</div>
大乔	李时珍	贵州省第一人民医院	2023-04-04 00:00:00	2023-06-06 15:00:00		待完成	<div>编辑</div> <div>删除</div>

共 3 条10条/页<1>前往1页

修改病人信息：

我们回到患者管理页面，修改“关羽”的信息

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

患者管理

管理员

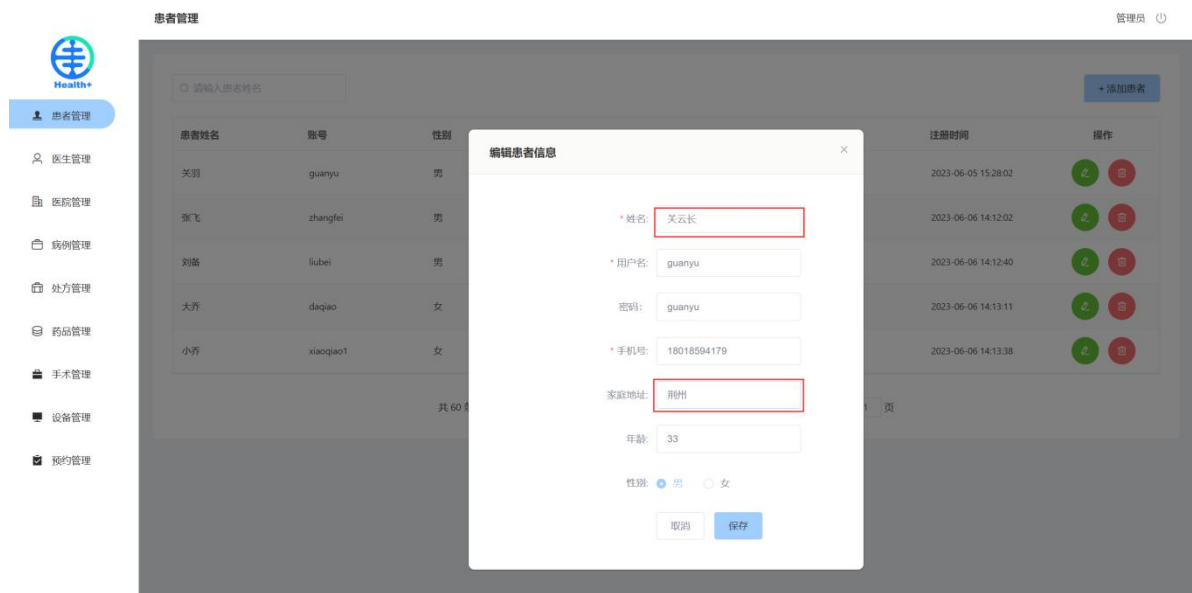
请输入患者姓名

+ 添加患者

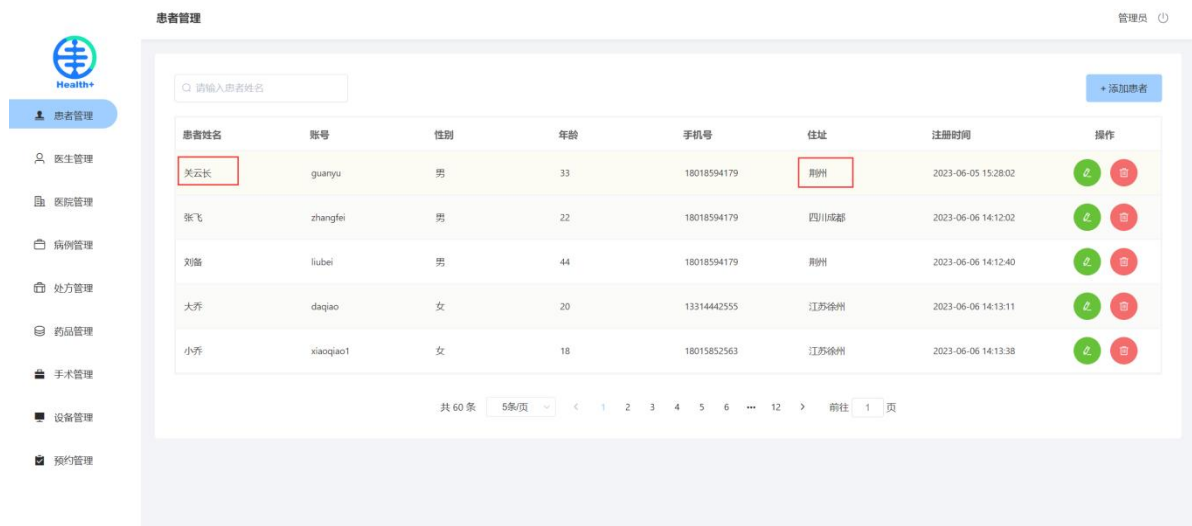
患者姓名	账号	性别	年龄	手机号	住址	注册时间	操作
关羽	guanyu	男	33	18018594179	同济大学	2023-06-05 15:28:02	<div>编辑</div> <div>删除</div>
张飞	zhangfei	男	22	18018594179	四川成都	2023-06-06 14:12:02	<div>编辑</div> <div>删除</div>
刘备	liubei	男	44	18018594179	荆州	2023-06-06 14:12:40	<div>编辑</div> <div>删除</div>
大乔	daqiao	女	20	13314442555	江苏徐州	2023-06-06 14:13:11	<div>编辑</div> <div>删除</div>
小乔	xiaoqiao1	女	18	18015852563	江苏徐州	2023-06-06 14:13:38	<div>编辑</div> <div>删除</div>

共 60 条5条/页<123456...12>前往1页

将姓名改为关云长，住址改为“荆州”：



可以看到信息也被成功修改：



注意：这里不能用户的账号不能重复，因为其设置了 **unique** 属性，如果修改或添加时发现存在相同的账号，会提示错误。

模糊查询：

提供了模糊查询功能，我们在输入框中输入病人姓名关键字“乔”，则会查询名字中所有带有“乔”字的病人：

(4) 病例管理

基本同患者管理模块，这里增加了“批量删除”和按日期检索功能，下面进行测试：

病例管理

管理员

请输入姓名

开始日期 至 结束日期

批量删除 + 添加病例

患者姓名	医生姓名	医院	开始时间	结束时间	临床诊断	治疗方式	治疗结果	操作
张飞	华佗	同济大学医院	2023-04-04 00:00:00	2023-04-29 00:00:00	发烧	处方12	痊愈	
张飞	李时珍	贵州第一人民医院	2023-04-04 00:00:00	2023-04-10 00:00:00	冠状病毒	处方166591604634535936	好转	
大乔	华佗	同济大学医院	2023-04-01 00:00:00	2023-04-29 00:00:00	感冒	处方11	痊愈	
小乔	扁鹊	北京市协和医院	2023-03-15 00:00:00	2023-04-29 00:00:00	骨折	手术12	好转	
关云长	华佗	同济大学医院	2020-07-30 00:00:00	2020-07-31 00:00:00	中毒	手术166590887850628710	痊愈	

共 5 条 10条/页 < 1 > 前往 1 页

表格共有 5 条数据，我们按红框中的时间段来筛选，筛选 2023 年 3 月 1 日至 4 月 1 日的记录：

病例管理

管理员

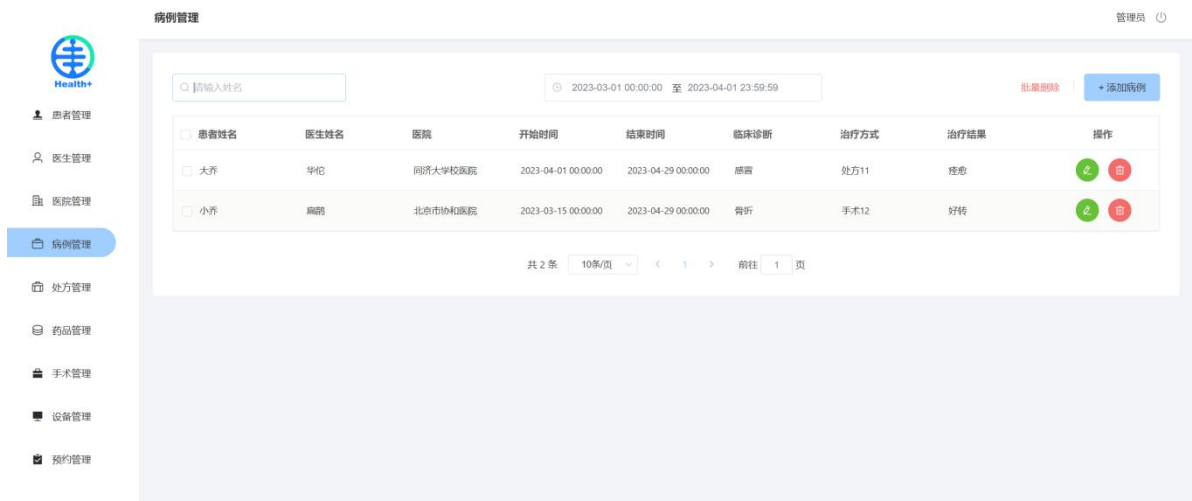
请输入姓名

开始日期 至 结束日期

批量删除 + 添加病例

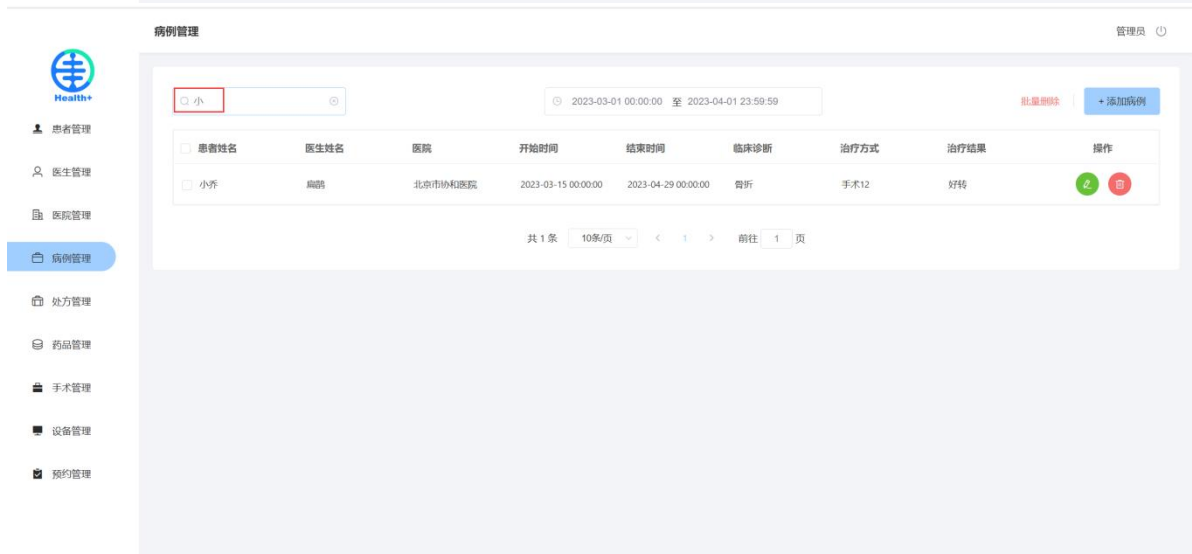
患者姓名	医生姓名	医院	开始时间	结束时间	临床诊断	治疗方式	治疗结果	操作
张飞	华佗	同济大学医院	2023-					
张飞	李时珍	贵州第一人民医院	2023-					
大乔	华佗	同济大学医院	2023-					
小乔	扁鹊	北京市协和医院	2023-					
关云长	华佗	同济大学医院	2020-					

共 5 条 10条/页 < 1 > 前往 1 页



可以看到有两条符合条件的记录。

这里还支持双条件查询，比如在姓名输入框中，输入医生或病人的姓名，可进一步筛选记录：



批量删除：

我们选中 1、3、4 调记录，点击批量删除：

The screenshot shows the '病例管理' (Case Management) interface. A confirmation dialog titled '确定删除' (Confirm Deletion) is displayed, asking '确定删除该病例，是否继续?' (Confirm deleting this case, continue?). The dialog has '取消' (Cancel) and '确定' (Confirm) buttons. A red arrow points to the '确定' button. The background table shows a list of cases with columns: 患者姓名 (Patient Name), 医生姓名 (Doctor Name), 医院 (Hospital), 开始时间 (Start Time), 结束时间 (End Time), 临床诊断 (Clinical Diagnosis), 治疗方式 (Treatment Method), 治疗结果 (Treatment Result), and 操作 (Action). The table is filtered to show 5 records. The '操作' column has '删除' (Delete) and '恢复' (Restore) icons. The '批量删除' (Batch Delete) button is visible in the top right corner.

患者姓名	医生姓名	医院	开始时间	结束时间	临床诊断	治疗方式	治疗结果	操作
张飞	华佗	同济大学医院	2023-04-04 00:00:00	2023-04-29 00:00:00	发烧	处方12	痊愈	删除 恢复
张飞	李时珍	贵州省第一人民医院	2023-04-04 00:00:00	2023-04-10 00:00:00	冠状病毒	处方166591604634535936	好转	删除 恢复
大乔	华佗	同济大学医院	2023-04-01 00:00:00	2023-04-29 00:00:00	感冒	处方11	痊愈	删除 恢复
小乔	扁鹊	北京市协和医院				手术12	好转	删除 恢复
关云长	华佗	同济大学医院				手术166590867850628710	痊愈	删除 恢复

共 5 条 10条/页 1 2 3 4 5 前往 1 页

可以看到记录被成功删除。

(5) 处方管理

功能基本同患者管理，但此处药品一栏中用“,”分割的每一项药品，都必须在药品表中查得到，否则该处方无效，添加失败。

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

处方管理

管理员

请输入处方id

批量删除

+ 添加处方

处方id	药品	价格	操作
<input type="checkbox"/> 1665916046345359362	板蓝根, 红霉素眼药膏	¥ 11.2	<div>批</div> <div>售</div>
<input type="checkbox"/> 1665971030147485698	人参, 枸杞, 黑枸杞	¥ 108	<div>批</div> <div>售</div>

共 2 条 10条/页 < 1 > 前往 1 页

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

处方管理

管理员

请输入处方id

批量删除

+ 添加处方

新增处方

所包含药物 葡萄糖, 枸杞

处方价格 9.15

取消 保存

因为药品表里并没有葡萄糖：

(6) 药品管理

Health+

患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

药品管理

管理员

请输入药品名称

请输入医院名称

批量删除

批量启售

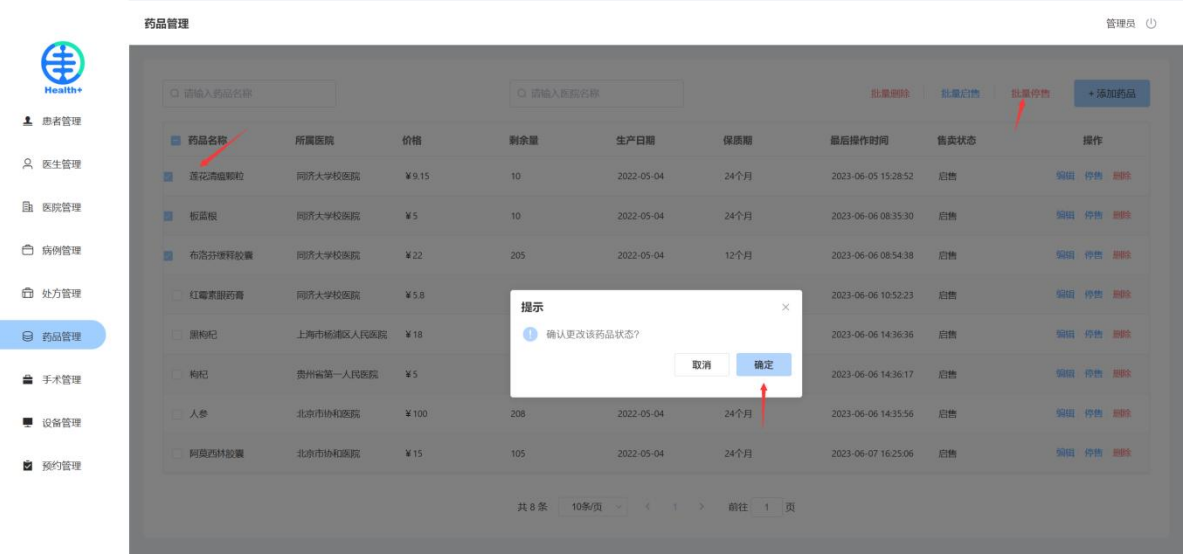
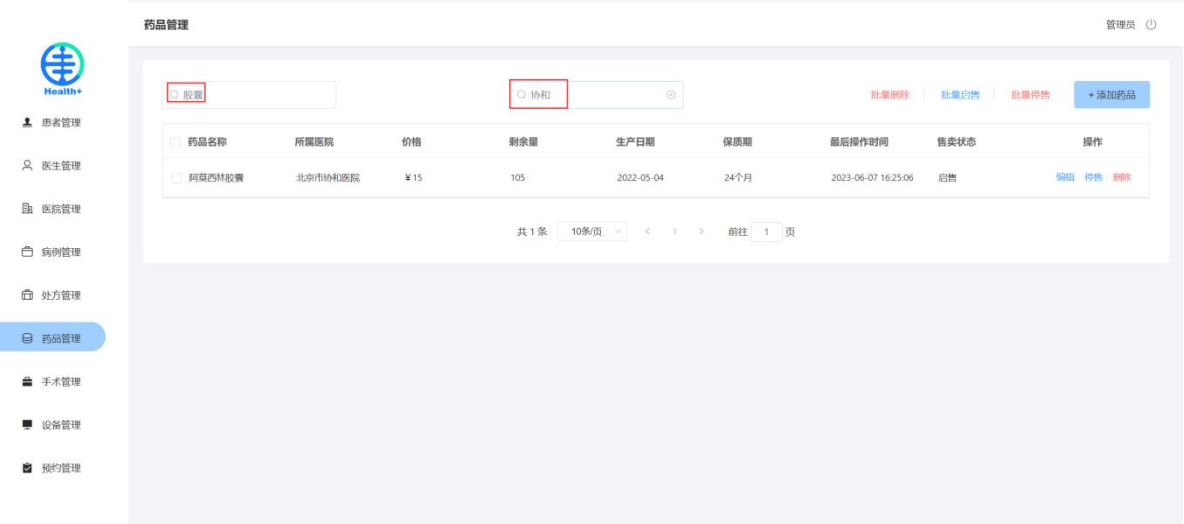
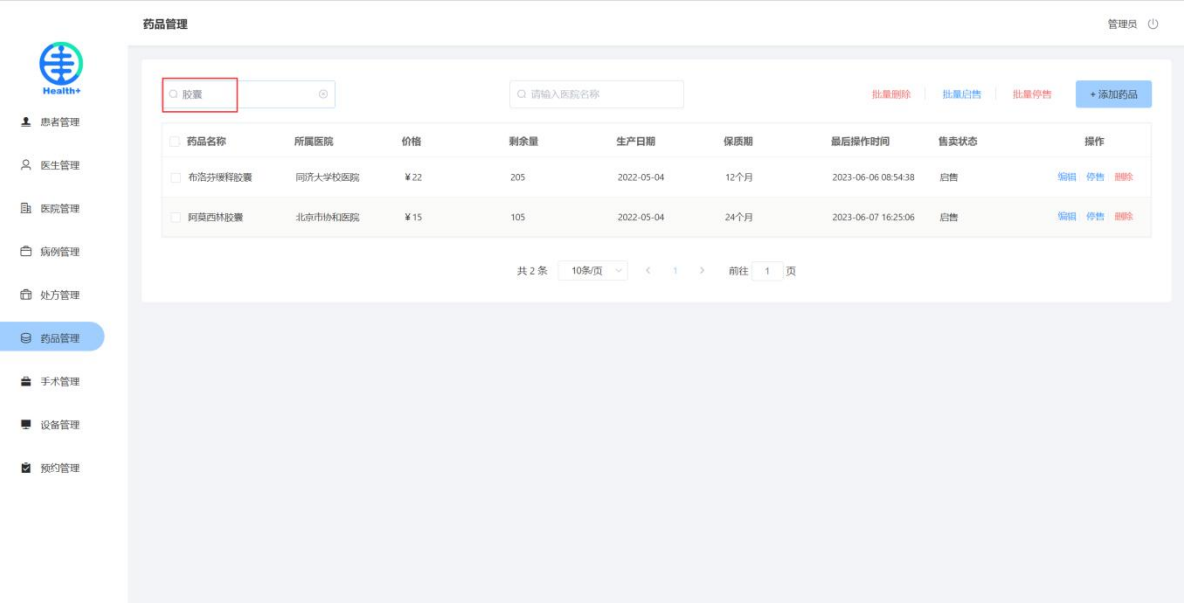
批量停售


+ 添加药品

药品名称	所属医院	价格	剩余量	生产日期	保质期	最后操作时间	售卖状态	操作
<input type="checkbox"/> 莲花清瘟颗粒	同济大学附属医院	¥ 9.15	10	2022-05-04	24个月	2023-06-05 15:28:52	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 板蓝根	同济大学附属医院	¥ 5	10	2022-05-04	24个月	2023-06-06 08:35:30	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 布洛芬缓释胶囊	同济大学附属医院	¥ 22	205	2022-05-04	12个月	2023-06-06 08:54:38	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 红霉素眼药膏	同济大学附属医院	¥ 5.8	100	2022-05-04	24个月	2023-06-06 10:52:23	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 黑枸杞	上海市杨浦区人民医院	¥ 18	192	2022-05-04	24个月	2023-06-06 14:36:36	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 枸杞	贵州省第一人民医院	¥ 5	100	2022-05-04	24个月	2023-06-06 14:36:17	启售	<div>编辑</div> <div>停售</div> <div>删除</div>
<input type="checkbox"/> 人参	北京市协和医院	¥ 100	208	2022-05-04	24个月	2023-06-06 14:35:56	启售	<div>编辑</div> <div>停售</div> <div>删除</div>

共 7 条 10条/页 < 1 > 前往 1 页

功能基本同患者管理，提供药品、医院双名称模糊查询，并增加了批量起售和批量停售功能：





患者管理

医生管理

医院管理

病例管理

处方管理

药品管理

手术管理

设备管理

预约管理

药品管理

管理员

药品状态已经更改成功!

请输入药品名称

批量删除 批量启售 批量停售 + 添加药品

药品名称	所属医院	价格	剩余量	生产日期	保质期	最后操作时间	售卖状态	操作
莲花清瘟颗粒	同济大学附属医院	¥9.15	10	2022-05-04	24个月	2023-06-05 15:28:52	停售	编辑 启售 删除
板蓝根	同济大学附属医院	¥5	10	2022-05-04	24个月	2023-06-06 08:35:30	停售	编辑 启售 删除
布洛芬缓释胶囊	同济大学附属医院	¥22	205	2022-05-04	12个月	2023-06-06 08:54:38	停售	编辑 启售 删除
红霉素眼药膏	同济大学附属医院	¥5.8	100	2022-05-04	24个月	2023-06-06 10:52:23	启售	编辑 停售 删除
黑枸杞	上海市杨浦区人民医院	¥18	192	2022-05-04	24个月	2023-06-06 14:36:36	启售	编辑 停售 删除
枸杞	贵州省第一人民医院	¥5	100	2022-05-04	24个月	2023-06-06 14:36:17	启售	编辑 停售 删除
人参	北京市协和医院	¥100	208	2022-05-04	24个月	2023-06-06 14:35:56	启售	编辑 停售 删除
阿莫西林胶囊	北京市协和医院	¥15	105	2022-05-04	24个月	2023-06-07 16:25:06	启售	编辑 停售 删除

共 8 条 10条/页 < 1 > 前往 1 页

药品管理

管理员

请输入药品名称

请输入医院名称

批量删除 批量启售 批量停售 + 添加药品

药品名称	所属医院	价格	剩余量	生产日期	保质期	最后操作时间	售卖状态	操作
莲花清瘟颗粒	同济大学附属医院	¥9.15	10	2022-05-04	24个月	2023-06-05 15:28:52	停售	编辑 启售 删除
板蓝根	同济大学附属医院	¥5	10	2022-05-04	24个月	2023-06-06 08:35:30	停售	编辑 启售 删除
布洛芬缓释胶囊	同济大学附属医院	¥22	205	2022-05-04	12个月	2023-06-06 08:54:38	停售	编辑 启售 删除
红霉素眼药膏	同济大学附属医院	¥5.8	100	2022-05-04	24个月	2023-06-06 10:52:23	启售	编辑 停售 删除
黑枸杞	上海市杨浦区人民医院	¥18	192	2022-05-04	24个月	2023-06-06 14:36:36	启售	编辑 停售 删除
枸杞	贵州省第一人民医院	¥5	100	2022-05-04	24个月	2023-06-06 14:36:17	启售	编辑 停售 删除
人参	北京市协和医院	¥100	208	2022-05-04	24个月	2023-06-06 14:35:56	启售	编辑 停售 删除
阿莫西林胶囊	北京市协和医院	¥15	105	2022-05-04	24个月	2023-06-07 16:25:06	启售	编辑 停售 删除

共 8 条 10条/页 < 1 > 前往 1 页

提示

确认更改该药品状态?

取消 确定

药品管理

管理员

药品状态已经更改成功!

请输入药品名称

批量删除 批量启售 批量停售 + 添加药品

药品名称	所属医院	价格	剩余量	生产日期	保质期	最后操作时间	售卖状态	操作
莲花清瘟颗粒	同济大学附属医院	¥9.15	10	2022-05-04	24个月	2023-06-05 15:28:52	启售	编辑 停售 删除
板蓝根	同济大学附属医院	¥5	10	2022-05-04	24个月	2023-06-06 08:35:30	启售	编辑 停售 删除
布洛芬缓释胶囊	同济大学附属医院	¥22	205	2022-05-04	12个月	2023-06-06 08:54:38	启售	编辑 停售 删除
红霉素眼药膏	同济大学附属医院	¥5.8	100	2022-05-04	24个月	2023-06-06 10:52:23	启售	编辑 停售 删除
黑枸杞	上海市杨浦区人民医院	¥18	192	2022-05-04	24个月	2023-06-06 14:36:36	启售	编辑 停售 删除
枸杞	贵州省第一人民医院	¥5	100	2022-05-04	24个月	2023-06-06 14:36:17	启售	编辑 停售 删除
人参	北京市协和医院	¥100	208	2022-05-04	24个月	2023-06-06 14:35:56	启售	编辑 停售 删除
阿莫西林胶囊	北京市协和医院	¥15	105	2022-05-04	24个月	2023-06-07 16:25:06	启售	编辑 停售 删除

共 8 条 10条/页 < 1 > 前往 1 页

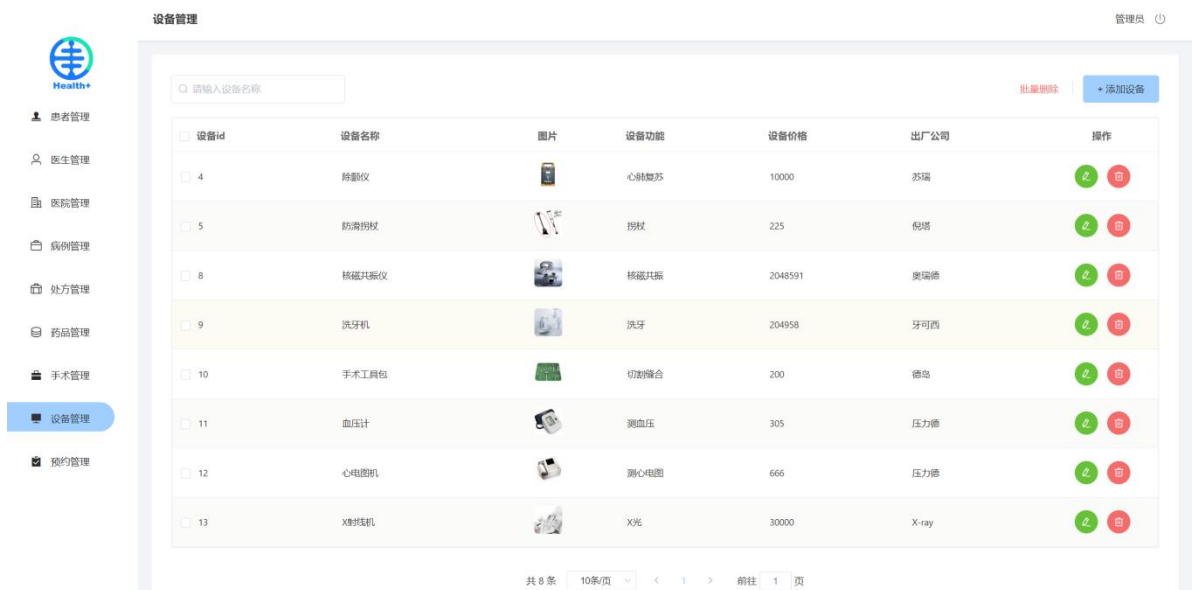
(7) 手术管理

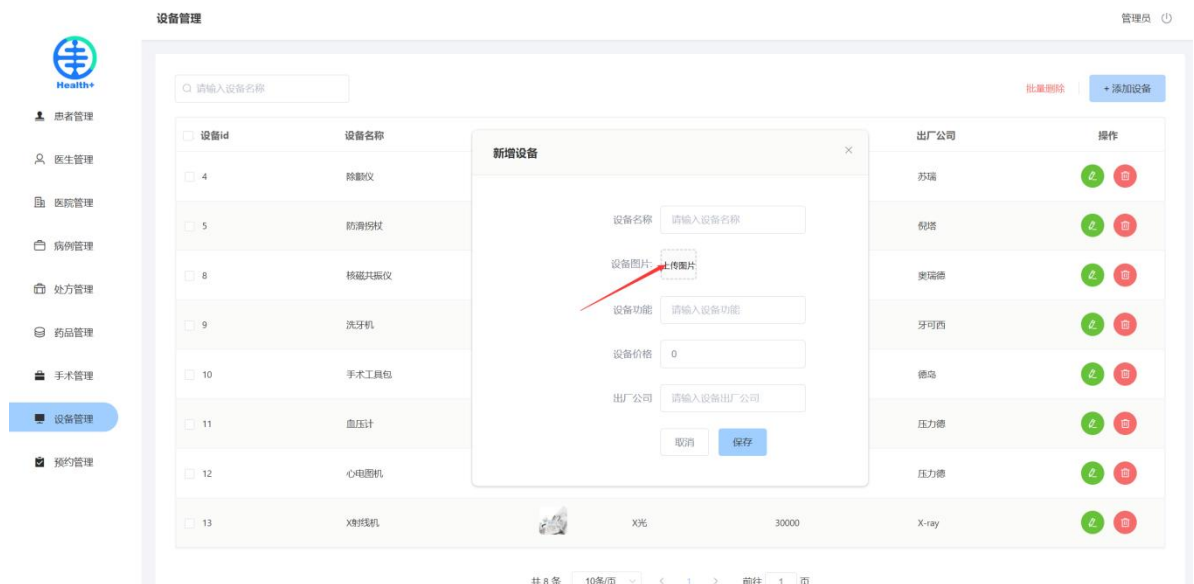
功能基本同处方管理，其中“仪器设备”一栏中用逗号分割的每个设备也必须存在于设备表中，否则添加失败：



(8) 设备管理

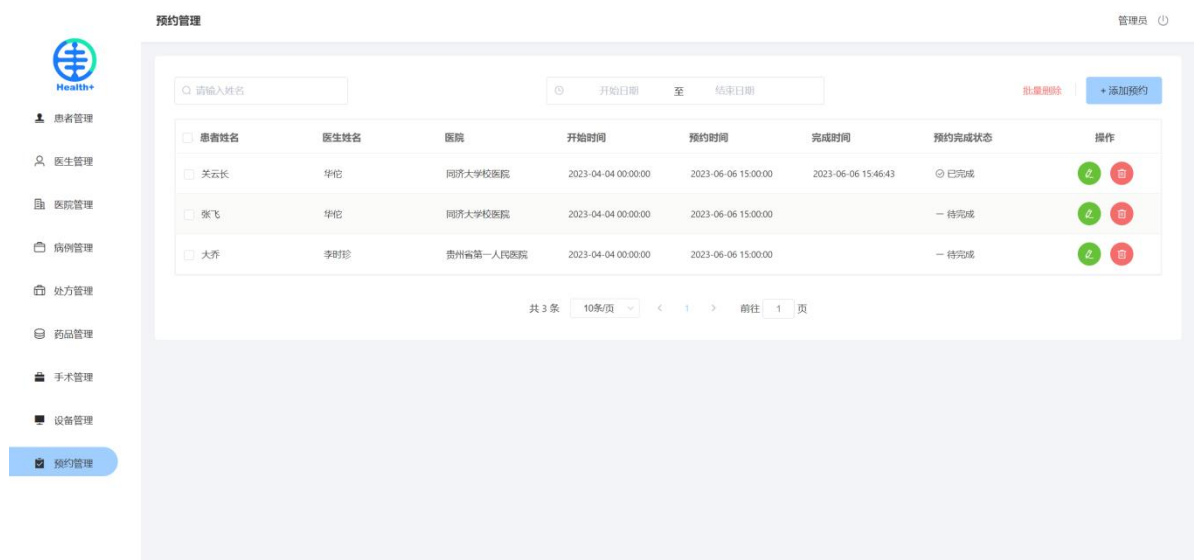
基本同药品管理，这里添加了设备的图片展示，添加设备时也需要上传图片。





(9) 预约管理

基本同病例管理。



2、redis 效率测试

当查询操作比较频繁时，使用 redis 的作用是比较明显的。比如，使用 apifox 模拟 3 个线程的 10 次查询，重复 5 次，共 150 次查询请求，有 redis 缓存和无 redis 缓存的查询时间对比如下：



全部 成功 失败

线程 1

第 1 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 2 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 3 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 4 轮



全部 成功 失败

线程 1

第 1 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 2 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 3 轮

1 病人模块频繁查询

● 已完成

通过率: 100.00%

[详情 >](#)

第 4 轮

可以看到, 有 redis 的情况下, 平均接口请求时间少了 10 倍, 优化效果极为显著。对于查询和增删改交替的情况, 也能实现优化, 但表改动时一般会清除 redis 的某些缓存, 下次查询又得查 mysql, 故优化效果没有单查询操作效率高:

增删改查交替操作 共90次请求, 有redis

已完成
90

● 通过

98.89%89

● 失败

1.11%1

● 未测

0.00%0

总耗时

27.653 秒

平均接口请求耗时

22 毫秒

循环数

执行:30

断言数

执行:0失败:0

导出报告

全部	成功	失败
线程 1		
第 1 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
第 2 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
第 3 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
← 病人模块的增删改查		
增删改查交替进行, 共90次测试, 无redis		

增删改查交替进行, 共90次测试, 无redis

已完成
90

● 通过

98.89%89

● 失败

1.11%1

● 未测

0.00%0

总耗时

32.974 秒

平均接口请求耗时

105 毫秒

循环数

执行:30

断言数

执行:0失败:0

导出报告

全部	成功	失败
线程 1		
第 1 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
第 2 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
第 3 轮		
1 病人模块的增删改查	<div><div>● 已完成</div></div>	通过率: 100.00% 详情 >
第 4 轮		

3、主从库测试

清空控制台，在患者管理中添加一名病人，查看控制台打印的日志：

```
2023-06-07 16:43:28.168 INFO 23496 --- [nio-8080-exec-6] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 16:43:28.168 INFO 23496 --- [nio-8080-exec-6] ShardingSphere-SQL : SQL: INSERT INTO patient ( id,
name,
uid,
address,
create_time ) VALUES ( ?,
?,
?,
?,
? )
:: DataSources: master
```

可以看到插入语句执行时，使用的是 master 数据源，执行完插入语句后的查询语句，使用的是 slave 数据源：

```
2023-06-07 16:43:28.737 INFO 23496 --- [nio-8080-exec-7] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 16:43:28.737 INFO 23496 --- [nio-8080-exec-7] ShardingSphere-SQL : SQL: SELECT COUNT(*) FROM patient ::: DataSources: slave
==> Parameters:
<== Columns: COUNT(*)
<== Row: 63
<== Total: 1
```

这里主从库是配置在两台虚拟机上的：

```
#主从库配置，主库负责增删改，从库负责查询，缓解数据库压力
shardingsphere:
  datasource:
    names:
      master,slave
    master:
      type: com.alibaba.druid.pool.DruidDataSource
      driver-class-name: com.mysql.cj.jdbc.Driver
      url: jdbc:mysql://192.168.179.139:3306/hospital_info_system?serverTimezone=Asia/Shanghai&useUnicode=true&characterEncoding=utf-8&zeroDateTimeB
      username: root
      password: 123456
    slave:
      type: com.alibaba.druid.pool.DruidDataSource
      driver-class-name: com.mysql.cj.jdbc.Driver
      url: jdbc:mysql://192.168.179.140:3306/hospital_info_system?serverTimezone=Asia/Shanghai&useUnicode=true&characterEncoding=utf-8&zeroDateTimeB
      username: root
      password: 123456
  masterslave:
    #读写分离机制
    load-balance-algorithm-type: round_robin #轮询
    #最终的数据源名称
    name: dataSource
    master-data-source-name: master
    slave-data-source-names: slave
```

这样，就认证了主从库的工作机制。

4、事务控制测试

使用删除医院功能来测试事务控制。删除医院，意味着要删除该医院下的所有药品、所有医生及与该医生有关预约和病例，一旦一张表的删除操作执行不成功，对其他表的修改将不会被提交。假设我们点击删除某医院，执行下段代码：

```
HospitalController.java
126 @Transactional
127 @DeleteMapping("/{delete}")
128 public R<String> delete(Long id){
129
130     //将该医院所有相关的药品、医生、病例都删除
131     LambdaQueryWrapper<Drug> drugLambdaQueryWrapper = new LambdaQueryWrapper<>();
132     drugLambdaQueryWrapper.eq(Drug::getHid, id);
133     drugService.remove(drugLambdaQueryWrapper);
134
135     //删除医生
136     LambdaQueryWrapper<Doctor> doctorLambdaQueryWrapper = new LambdaQueryWrapper<>();
137     doctorLambdaQueryWrapper.eq(Doctor::getHid, id);
138     List<Doctor> doctors = doctorService.list(doctorLambdaQueryWrapper);
139     List<Long> ids = doctors.stream().map(doctor -> {
140         Long did = doctor.getId();
141         return did;
142     }).collect(Collectors.toList());
143     doctorService.remove(doctorLambdaQueryWrapper);
144
145     //与医生有关的预约都删除
146     LambdaQueryWrapper<Appointment> appointmentLambdaQueryWrapper = new LambdaQueryWrapper<>();
147     appointmentLambdaQueryWrapper.in(Appointment::getDid, ids);
148     appointmentService.remove(appointmentLambdaQueryWrapper);
149
150     //医生的账户也删除
151     LambdaQueryWrapper<User> userLambdaQueryWrapper = new LambdaQueryWrapper<>();
152     List<Long> uids = doctors.stream().map(doctor -> {
153         Long uid = doctor.getUid();
154         return uid;
155     }).collect(Collectors.toList());
156     userLambdaQueryWrapper.in(User::getId, uids);
157     userService.remove(userLambdaQueryWrapper);
158
159     //与医生有关的病例都删除
160     LambdaQueryWrapper<Cases> casesLambdaQueryWrapper = new LambdaQueryWrapper<>();
161     casesLambdaQueryWrapper.in(Cases::getDid, ids);
162     casesService.remove(casesLambdaQueryWrapper);
163
164     hospitalService.removeById(id);
165
166     clearRedisCache();
167     return R.success("医院删除成功！");
168 }
169 public void clearRedisCache(){
```

从日志报告中可以看到，函数调用时创建了一个 `sqlSession`，并注册了一个事务。而后对 `drug`、`doctor`、`appointment`、`user`、`cases`、`hospital` 表分别执行相应的删除操作，均操作成功后事务才被提交，并取消注册，关闭会话。

```
Console Endpoints
Creating a new SqlSession
Registering transaction synchronization for SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
JDBC Connection [org.apache.shardingsphere.shardingjdbc.jdbc.core.connection.MasterSlaveConnection@3ed08902] will be managed by Spring
==> Preparing: DELETE FROM drug WHERE (hid = ?)
2023-06-07 19:57:47.749 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.749 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM drug

WHERE (hid = ?) ::: DataSources: master
==> Parameters: 1666412541624569858(Long)
<== Updates: 0
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: SELECT id,name,hid,department,level,create_time,uid FROM doctor WHERE (hid = ?)
2023-06-07 19:57:47.755 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.755 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: SELECT id,name,hid,department,level,create_time,uid FROM doctor

WHERE (hid = ?) ::: DataSources: master
==> Parameters: 1666412541624569858(Long)
<== Columns: id, name, hid, department, level, create_time, uid
Row: 1666414084163706881, 吴关鹏, 1666412541624569858, 内科, 医师, 2023-06-07 19:57:33, 1666414084054654978
<== Total: 1
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: DELETE FROM doctor WHERE (hid = ?)
2023-06-07 19:57:47.759 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.759 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM doctor

WHERE (hid = ?) ::: DataSources: master
==> Parameters: 1666412541624569858(Long)
<== Updates: 1
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: DELETE FROM appointment WHERE (did IN (??))
2023-06-07 19:57:47.762 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.762 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM appointment

WHERE (did IN (??)) ::: DataSources: master
==> Parameters: 1666414084163706881(Long)
<== Updates: 0
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: DELETE FROM user WHERE (id IN (??))
2023-06-07 19:57:47.764 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.765 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM user

WHERE (id IN (??)) ::: DataSources: master
==> Parameters: 1666414084054654978(Long)
<== Updates: 1
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: DELETE FROM cases WHERE (did IN (??))
2023-06-07 19:57:47.768 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.768 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM cases

WHERE (did IN (??)) ::: DataSources: master
==> Parameters: 1666414084163706881(Long)
<== Updates: 0
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28] from current transaction
==> Preparing: DELETE FROM hospital WHERE id=?
2023-06-07 19:57:47.782 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : Rule Type: master-slave
2023-06-07 19:57:47.782 INFO 3672 --- [nio-8080-exec-2] ShardingSphere-SQL : SQL: DELETE FROM hospital WHERE id=? ::: DataSources: master
==> Parameters: 1666412541624569858(Long)
<== Updates: 1
Releasing transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Transaction synchronization committing SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Transaction synchronization deregistering SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Transaction synchronization closing SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5ed64e28]
Creating a new SqlSession
SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@612aa308] was not registered for synchronization because synchronization is not active
JDBC Connection [org.apache.shardingsphere.shardingjdbc.jdbc.core.connection.MasterSlaveConnection@655edfe8] will not be managed by Spring
```

（这里因为删除操作不容易触发异常，导致不好模拟事务的处理过程，所以只好通过查看日志的方式来确定事务的开启和关闭。但开发过程中会报出很多莫名其妙的异常，证明该事务控制确实是有效的）

八、总结

项目从 0 开始，历经两个学期，最终完成。上学期主要完成了数据库表的设计，由于知识体系的不完善，考虑不周，表结构多次的确定几经波折，经历了 n 多次返工，造成了较大的时间开销。有些表的改动，完全是因为在开发过程中发现设计不合理，导致代码编写不方便，需要重构表结构，在整合后端技术时也有可能需要更改表和字段的名称等，如果从一开始就熟悉和掌握这种设计模式，开发效率就会大大提升。

同时因为现实环境的变化，自己没有坚持最初第一版的设计，删除了许多感觉不必要的内容，尽可能地简单、又不失内容地完成项目，简化后端、前端开发，将工作重心放在数据库表的设计、功能体现上，达到《数据库设计原理》这门课程的目的要求。当然，系统的设计并不是十分完善的，存在一些不合理的地方，比如 1、处方中包含的药品数量，应该是大于等于 0，而不是固定为 1，处方的总价格，应该是由处方中药品单价*数量之和来确定的，而为了简化开发，处方的总价格是在添加时手动输入的；2、对于病例表中的治疗方式，应该引用处方 id 或手术 id 作为外键，将病例管理和处方管理、手术管理关联起来，但这里也是简单处理，使用字符串手动输入治疗方式.....

总的来说，尽管系统不是很合理、完善，但设计开发的过程还是收获颇丰，熟悉了数据库设计的整个流程，深入体会了事务控制带来的好处，简单了解了 nosql 型数据库 redis，以及如何设计索引结构、配置主从库等提升查询效率或减少数据库压力的方式等等，对自己以后的学习的发展是极有帮助的。