四川大學

SICHUAN UNIVERSITY

医院门诊挂号系统

数 据 库 原 理 实 验 期 末 报 告

任课教师 段磊

学　　院 计算机学院

专　　业 计算机科学与技术

组　　别 第四组

组　　长 郑仕博

成　　员 吴正博，高俊翔

2025 年 12 月 21 日

目录

1 引言

1.1 目的

本报告面向课程《数据库原理》实验的期末验收，系统性描述“医院门诊挂号系统”的需求、数据库设计与实现方案，说明系统**做什么**、**怎么做**以及关键业务规则如何落地到关系模式与事务逻辑中。

1.2 项目范围

系统覆盖门诊挂号的核心业务链：账号与角色、科室/医生信息、号源（排班）管理、在线挂号、余额支付与取消退款、就诊病历、站内消息与权限控制。系统包含四类角色：患者、医生、科室管理员、系统管理员。

1.3 文档概览

第 1 章说明背景与范围；第 2 章给出系统概览与典型流程；第 3 章描述整体架构与关键实现策略；第 4 章给出数据库 E-R 图、关系模式与约束；第 5 章说明各业务模块的实现；第 6 章概述前端页面；第 7 章为需求矩阵；第 8 章附录提供部署与测试方式。

1.4 参考资料

- 项目文档：`README.md`、`SYSTEM_README.md`、`requirements/需求.md`、`requirements/数据表设计.md`、`requirements/模块接口.md`。
- 数据库脚本：`sql/schema.sql`、`sql/sample_data.sql`。
- PostgreSQL 官方文档（SQL 语法、约束与事务语义）。

1.5 术语与缩略语

E-R 图（实体联系图）、PK（主键）、FK（外键）、ACID（事务特性）、REST（接口风格）、Slot（号源/时间段）、Reg（挂号记录）、RBAC（基于角色的访问控制，本文实现为简化版）。

2 系统概览

2.1 系统目标

以“数据库为中心”完成一个可运行的 Web 项目，能够体现数据库原理课程的核心点：关系模式设计、主外键约束、典型增删改查、关键业务规则的一致性维

护（容量、金额、状态流转）以及面向多角色的权限控制。

2.2 角色与权限

- 患者（patient）：注册/登录；查询科室、医生与可预约号源；创建挂号、支付/取消、充值；查看挂号与就诊记录；在满足规则下发送站内消息。
- 医生（doctor）：查看某日挂号患者列表；为已挂号记录录入病历；向患者/科室管理员/系统管理员发送消息（受“已支付挂号 +10 天有效期”约束）。
- 科室管理员（dept_manager）：维护本科室医生归属；为医生创建号源（日期、起止时间、容量、费用、备注）；查看本科室号源与挂号记录；按规则向本科室医生/患者发送消息。
- 系统管理员（admin）：创建医生/科室管理员/系统管理员账号；冻结/解冻账号；通过消息机制发送系统通知。

2.3 典型业务流程

1. 管理员初始化：执行 `sql/schema.sql` 建表，执行 `sql/sample_data.sql` 插入样例数据；使用系统管理员账号登录进行账号维护，使用科室管理员账号登录进行号源维护。
2. 患者挂号：选择科室与日期查询医生号源（含起止时间、余量与费用）→ 选择 `slotId` 创建挂号 → 余额支付或取消（已支付按 90% 退款）。
3. 医生就诊：医生端按日期查看挂号患者列表 → 对某挂号记录录入病历（诊断、治疗方案、医嘱）→ 患者在个人中心查看就诊记录。
4. 消息沟通：基于“已支付挂号关系 + 双方最近消息 10 天内活跃 + 角色限制”校验发送权限；支持会话列表与会话明细查询。

3 系统架构

3.1 总体架构

系统采用前后端分离三层结构：

- 前端：Vue2，实现登录/注册、挂号页、消息页、个人中心与各角色工作台；通过代理将 `/api` 请求转发到后端。
- 后端：Spring Boot + MyBatis，提供 REST 接口（统一前缀 `/api`），业务逻辑集中在 Service 层，DAO 使用 Mapper 访问 PostgreSQL。

- 数据库：PostgreSQL，采用“手动执行脚本建库建表”的方式，保证验收时数据库模式清晰可复现。

3.2 鉴权与会话策略（实验简化）

为了聚焦数据库与业务一致性，本项目未引入完整 JWT/Spring Security。登录成功后返回的 `token` 简化为 `userId`，前端在请求头携带 `X-User-Id`，后端据此识别当前用户并做角色校验。

3.3 关键一致性策略

- 号源容量一致性：创建/取消挂号在同一事务内同时写入 `registration` 并更新 `doctor_time_slot.booked_count`，避免出现“挂号成功但容量未扣减”等不一致。
- 事务边界：挂号创建、支付与取消均以事务方式执行，保证“状态更新 + 金额/余量变更”原子提交或回滚。
- 金额统一以“分”存储：`patient.money`、`doctor_time_slot.fee`、`registration.reg_fee` 统一使用整数分，避免浮点误差；前端录入“元”，后端转换为“分”落库。
- 业务约束校验：不可预约过去日期；号源不足禁止挂号；同一患者同一号源仅允许一条未取消记录；号源时间段不可重叠；取消已支付挂号按 90% 退款；消息需满足“付费挂号关系 + 10 天有效期”。

4 数据设计

4.1 关系模式图

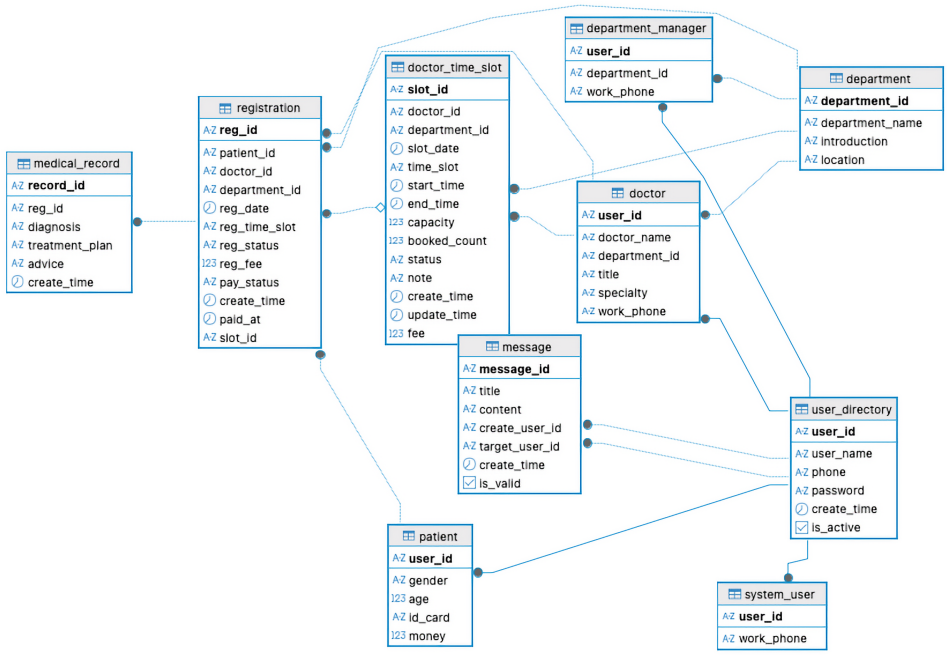

图 1: 数据库核心实体关系图

4.2 关系模式与数据字典

表结构与字段含义以 `sql/schema.sql` 和 `requirements/数据表设计.md` 为准，核心表如下：

表名	核心字段（节选）	说明
`user_directory`	`user_id`, `user_name`, `phone`, `password`, `create_time`, `is_active`	统一用户表（四类角色共享）
`patient`	`user_id`, `gender`, `age`, `id_card`, `money`	患者扩展信息，`money` 单位分
`department`	`department_id`, `department_name`, `introduction`, `location`	科室主数据
`doctor`	`user_id`, `doctor_name`, `department_id`, `title`, `specialty`	医生信息，关联科室
`department_manager`	`user_id`, `department_id`, `work_phone`	科室管理员，关联科室
`"system_user"`	`user_id`, `work_phone`	系统管理员

表名	核心字段（节选）	说明
doctor_time_slot	slot_id, doctor_id, slot_date, start_time, end_time, capacity, booked_count, fee	医生号源（排班）
registration	reg_id, patient_id, doctor_id, reg_date, reg_time_slot, reg_status, reg_fee, pay_status, slot_id	挂号记录与状态流转
medical_record	record_id, reg_id, diagnosis, treatment_plan, advice	病历，一次挂号对应一条记录
message	message_id, title, content, create_user_id, target_user_id, create_time, is_valid	站内消息
audit_log	log_id, user_id, action, target, time	操作审计

4.3 关键约束与规则落地

- 主外键约束：在 sql/schema.sql 中通过 FK 约束保障实体引用完整性（如 registration.patient_id 引用 patient.user_id）。
- 手机号唯一：user_directory.phone 设置唯一约束，避免重复注册。
- 号源时间段不重叠：科室管理员创建号源时，后端在同一医生同一日期内检查 start_time/end_time 区间是否与已有号源重叠。
- 容量控制：创建挂号前校验 capacity > booked_count；成功后在事务内将 booked_count + 1，取消时 booked_count - 1。
- 同号源重复挂号限制：创建挂号前按 (patient_id, slot_id) 查询是否存在未取消记录，避免重复占号。
- 取消退款：已支付挂号取消按 90% 退款并回写余额；未支付直接取消。
- 消息权限与时效：发送消息需满足“已支付挂号关系”且双方最近消息在 10 天内；超期仅可查看历史，不允许继续发送。

4.4 状态字段约定

- 挂号状态 registration.reg_status：Booked（已预约）、Canceled（已取消）、Finished（已完成，就诊流程可扩展）。
- 支付状态 registration.pay_status：Unpaid（未支付）、Paid（已支付）、Refunded（已退款/作废）。

4.5 脚本与示例数据

项目不自动建表。验收或演示时，建议使用数据库管理工具连接 PostgreSQL 并执行：`sql/schema.sql` 创建表结构，`sql/sample_data.sql` 插入示例数据（含科室、医生、管理员、号源与挂号样例）。

5 组件设计

5.1 账号与用户管理

- 患者注册：写入 `user_directory` 与 `patient`（统一用户 + 角色扩展信息）。
- 登录：支持使用 `userId` 或手机号登录；返回 `role` 与 `token=userId`。
- 冻结/解冻：系统管理员通过更新 `user_directory.is_active` 控制账号可用性。

5.2 科室、医生与号源

- 科室列表：提供科室名称与地点，供挂号页筛选。
- 医生列表：按科室与日期返回医生信息与可预约号源（含起止时间、余量与费用）。
- 号源创建：科室管理员为本科室医生创建号源；时间段校验 `start_time < end_time` 且不重叠。

5.2.1 号源创建（科室管理员）实现步骤

1. 权限校验：当前用户必须是科室管理员，且只能为本科室医生创建号源。
2. 参数校验：日期、起止时间、容量、费用必填，且满足 `start_time < end_time`、`capacity > 0`。
3. 重叠检查：查询该医生该日期的已有号源，若存在区间重叠（非 `end <= existing.start` 且非 `start >= existing.end`）则拒绝创建。
4. 落库：生成 `slot_id`；将展示字段 `time_slot` 固化为 “HH:mm-HH:mm”；费用由 “元” 换算为 “分”；初始化 `booked_count=0` 与 `status=OPEN`。

5.3 挂号、支付与取消

- 创建挂号：必须选择 `slotId`；校验日期合法、医生科室匹配、号源余量；写入 `registration` 并扣减余量。

- 支付：从 patient.money 扣减 registration.reg_fee，写入支付时间并更新状态。
- 取消：更新挂号状态；已支付按 90% 退款；释放号源余量。

5.3.1 创建挂号实现步骤

1. 校验 regDate 不能早于当天。
2. 校验医生属于所选科室，防止“科室/医生不匹配”造成脏数据。
3. 通过 slotId 定位号源，并校验号源的医生/科室/日期与请求一致。
4. 校验余量：capacity > booked_count 才允许创建。
5. 校验重复挂号：同一患者同一 slotId 若存在未取消记录则拒绝创建。
6. 写入 registration：复制费用到 reg_fee；将 reg_time_slot 统一保存为“HH:mm-HH:mm”；初始化 reg_status=Booked、pay_status=Unpaid。
7. 更新号源：booked_count + 1（与挂号写入处于同一事务内）。

5.3.2 支付与取消实现步骤

1. 支付：校验余额充足后扣减 patient.money，并将 pay_status 更新为 Paid，记录 paid_at。
2. 取消：仅允许取消 Booked 状态；已支付按 90% 退款并更新余额；将挂号状态置为 Canceled/Refunded 并释放号源（booked_count - 1）。

5.4 病历与就诊记录

医生对挂号记录录入病历（诊断、治疗方案、医嘱），患者与医生端分别提供查询接口，保证一次挂号对应一份可追溯的就诊记录。

5.5 站内消息

消息模块支持收件箱/发件箱、会话列表（最近一条）与会话明细。发送消息时进行角色与关系校验，并结合“10 天活跃期”限制控制继续对话的权限。

5.5.1 消息权限与 10 天有效期

1. 角色限制：按发送方/接收方角色确定可通信范围（患者仅能向就诊医生或对应科室管理员发送；医生可向就诊患者/本科室管理员/系统管理员发送；科室管理员可向本科室医生与本科室就诊患者发送；系统管理员可向任意用户发送）。

2. 关系校验：对“患者 ↔ 医生/科室管理员”等关系，要求存在近 10 天内的**已支付挂号**记录。

3. 会话活跃校验：若双方最近一条消息时间超过 10 天，禁止继续发送（历史可查看）。

4. 撤回/作废：通过 `message.is_valid` 进行逻辑失效，不物理删除，便于追溯。

6 人机界面设计

前端按角色展示不同导航与工作台：

- 登录/注册：患者注册与四类用户登录入口。
- 挂号页（患者）：选择科室与日期，展示医生与可约号源，完成挂号创建。
- 消息页（全角色）：会话列表 + 聊天明细 + 发送消息。
- 个人中心（全角色）：查看个人信息、修改密码；患者可充值与管理挂号记录；医生可查看挂号与病历。
- 医生端：按日期查看患者挂号并录入病历。
- 科室端：维护医生归属、创建号源、查看本科室挂号记录。
- 系统管理端：创建账号、冻结/解冻、发送系统通知。

7 需求矩阵

需求	实现（主要接口/数据表/页面）
登录/注册与修改密码	/api/auth/*；user_directory、patient；登录注册页、个人中心
科室与医生号源查询	/api/departments、/api/doctors；department、doctor、doctor_time_slot；挂号页
创建挂号、支付与取消	/api/registrations、/api/registrations/{id}/pay、/api/registrations/{id}/cancel；registration、doctor_time_slot、patient；个人中心
号源维护（科室管理员）	/api/dept/slots、/api/dept/doctors；doctor_time_slot；科室端
病历与就诊记录	/api/medical-records；medical_record、registration；医生端、个人中心
站内消息与权限时效	/api/messages*；message、registration；消息页
账号创建与冻结/解冻	/api/users/*；user_directory 与角色子表；系统管理端

表 2: 需求—实现矩阵（节选）

8 APPENDICES

8.1 部署与运行

1. 准备 PostgreSQL（本地 localhost:5432），连接后执行 sql/schema.sql（可选执行 sql/sample_data.sql）。

2. 后端配置运行 mvn compile 和 mvn package 进行编译后，用 java -jar 运行生成的 jar 包。

3. 前端安装依赖 npm install , 编译静态文件 npm run build ， 配置服务器能够访问静态文件。

4. 使用 nginx 将前端 /api/ 请求代理到后端使静态文件（即前端）可以向后端发送请求。

5. 可以通过 http://database.zhengshibo.top:2000/访问前端页面。

8.2 演示账号与建议流程

若执行了 sql/sample_data.sql，可使用系统管理员账号 AD0001 / admin123 登录进行演示；患者可通过前端注册创建。建议演示顺序：患者挂号（创建/支付/取消）→ 医生查看挂号并写病历 → 双方消息沟通 → 管理端创建账号与冻结解冻。