

数据库系统原理实验期末作业实验报告

——Community_Hospital社区医院管理系统

一、实验目的

- 综合能力培养：掌握综合运用数据库原理、方法和技术进行数据库应用系统分析、设计和开发的能力。
- 全流程实践：理解并实践从需求分析到系统演示的完整软件开发生命周期（SDLC）。

二、实验环境

- 数据库：MySQL 8.0+
- 开发语言：C++
- 数据库连接库：MySQL Connector C++ 8.3.0
- 开发环境：Visual Studio/GCC/Clang（支持C++11及以上标准）

三、实验场景

本实验为综合型实验项目，并非单一SQL语句练习。

项目背景

为实现社区医院门诊全流程数字化管理，覆盖患者预约挂号、现场就诊登记、缴费结算、病历管理等核心环节，同时通过数据分析掌握患者就诊规律与科室负荷情况，为医生排班、药品储备和运营决策提供数据支撑，提升门诊服务效率与管理精细化水平，特开发本社区医院门诊管理系统。

团队构成

- 24347104徐智勇，需求分析，数据库设计，程序设计，程序测试，程序Debug, 实验报告撰写。
- 24347012陈泽鸿，需求分析，数据库设计，程序测试，程序Debug, PPT制作。

四、实验内容与步骤

数据库设计和应用开发主要包括以下核心步骤：

1. 需求分析

（一）、核心用户角色及需求

（1）患者（客户）

- 网上预约：需填写姓名、就诊科室、联系电话及预计到达时间，完成门诊预约申请。
- 到院登记就诊：需提供姓名、就诊科室、联系电话、性别、身份证号码、诊室号等信息，完成现场就诊登记。
- 预约验证与就诊分配：已网上预约的患者，到院后可查询预约信息、验证身份，系统自动分配至相应诊室就诊。
- 缴费结算：就诊结束后，完成费用结算（支持医保、自费等支付方式），结算完成后离院。

(2) 前台（挂号 / 收费人员）

- 就诊信息录入：患者到院登记时，将其就诊信息准确存入就诊信息表。
- 预约挂号处理：记录患者预约信息至预约表；患者到院后核验预约信息，将其转入就诊信息表，并将预约表中对应记录标记为“已完成”或删除。
- 就诊状态更新与费用记录：患者缴费结算后，将其就诊状态从“就诊进行中”转为“已离院”，同步记录本次就诊总费用、医保报销金额、自费金额。
- 费用收取与收入记录：根据系统生成的收费报表收取患者费用，并实时记录医院门诊收入。
- 患者简略信息查询：可查看患者姓名、就诊科室、诊室号、就诊状态等基础信息，辅助服务开展。

(3) 管理人员（医务科 / 行政主管）

- 排班信息管理：可增添、修改诊室及医生排班信息，包括诊室编号、所属科室、可接诊时间段、医生姓名及职称。
- 账单统计查询：支持按日期、科室、医生等维度，统计门诊收入金额与就诊人次，生成相关账单报表。
- 患者详细信息查询：可通过姓名、电话、身份证号、诊室号或就诊时间等多条件，查询患者完整就诊信息。
- 员工信息查询：可查询医生、护士、行政人员等员工的工号、岗位、所属科室、工作状态等信息。
- 员工信息修改：可更新员工联系方式、排班权限、职称等相关信息。

(二)、系统核心功能需求

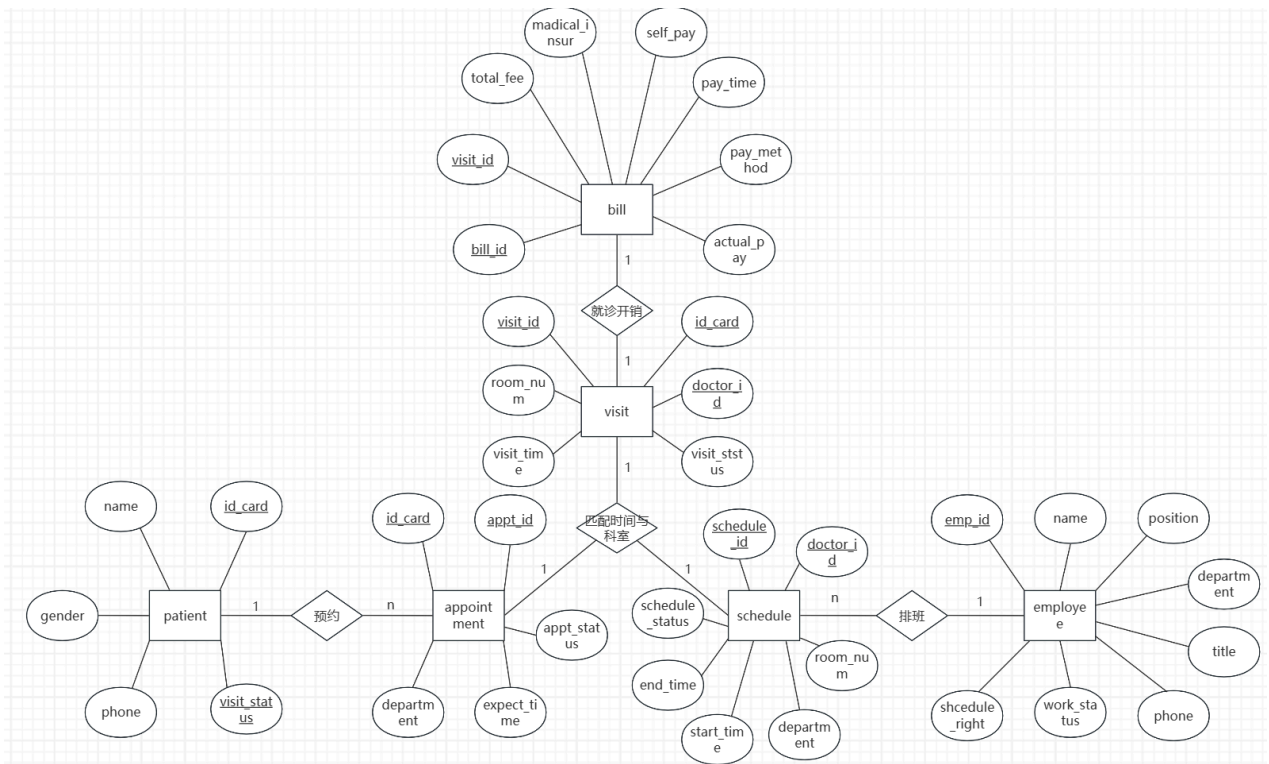
- 预约与登记功能：覆盖患者网上预约、现场登记、预约信息核验与就诊分配全流程，确保患者信息准确流转。
- 缴费结算功能：支持多支付方式对接，实现就诊费用计算、医保 / 自费拆分、费用收取记录与就诊状态同步更新。
- 排班管理功能：支持诊室与医生排班信息的新增、修改，保障门诊接诊资源合理配置。
- 查询统计功能：包括患者信息查询、员工信息查询、门诊收入与就诊人次统计，满足不同角色的数据使用需求。
- 数据管理功能：确保患者信息、就诊记录、费用数据、员工信息等数据的存储、更新、查询过程中，保持一致性与完整性，满足各类完整性约束。

(三)、非功能需求

- 数据一致性与完整性：系统需满足各类数据完整性约束，确保患者信息、费用记录、排班信息等关键数据无错误、无冗余、无冲突。
- 可操作性：功能流程清晰，界面简洁易用，适配患者、前台人员、管理人员等不同用户的操作习惯，降低使用门槛。
- 可追溯性：所有关键操作（如预约提交、登记录入、费用结算、信息修改）需留下记录，支持后续查询与核对。
- 可视化需求：系统相关的 E-R 图、表结构、业务流程图、统计报表等需清晰呈现，便于用户理解与使用（适配 PPT 演示与系统操作场景）。

2. 数据库设计

- 概念结构设计：绘制E-R图（实体-关系图）。



- 逻辑结构设计：将E-R图转换为关系模式，进行规范化处理（3NF），设计表结构(pk代表主码；fk代表外码)。

```
patient(id_card(pk), name, gender, phone, visit_status)
```

```
employee(emp_id(pk), name, position, department, title, phone, work_status, schedule_right)
```

```
appointment(appt_id(pk), id_card(fk), department, expect_time, appt_status)
```

```
schedule(schedule_id(pk), doctor_id(fk), room_num, department, start_time, end_time, schedule_status)
```

```
visit(visit_id(pk), id_card(fk), room_num, doctor_id(fk), visit_time, visit_status)
```

```
bill(bill_id(pk), visit_id(fk), total_fee, medical_insur, self_pay, pay_time, pay_method, actual_pay)
```

- 物理结构设计：确定存储结构、分区策略等。

3. 数据库创建和数据加载

- 编写SQL脚本创建数据库和表，编写存储过程、触发器。

```
CREATE DATABASE IF NOT EXISTS community_hospital CHARACTER SET utf8mb4;
```

```
USE community_hospital;
```

```
CREATE TABLE IF NOT EXISTS patient (
```

```
id_card VARCHAR(18) PRIMARY KEY COMMENT '身份证号（主键）',
```

```
name VARCHAR(50) NOT NULL COMMENT '姓名',
```

```
gender VARCHAR(10) COMMENT '性别',
```

```
phone VARCHAR(20) COMMENT '联系电话',
```

```

visit_status VARCHAR(20) COMMENT '就诊状态（未就诊/就诊中/已离院）'
) COMMENT '患者信息表';

CREATE TABLE employee (
    emp_id INT PRIMARY KEY COMMENT '工号（主键）',
    name VARCHAR(50) NOT NULL COMMENT '姓名',
    position VARCHAR(50) COMMENT '岗位（医生/护士/行政）',
    department VARCHAR(50) COMMENT '所属科室',
    title VARCHAR(50) COMMENT '职称（主治医师/护士等）',
    phone VARCHAR(20) COMMENT '联系方式',
    work_status VARCHAR(20) COMMENT '工作状态（在职/离职）',
    schedule_right TINYINT(1) COMMENT '排班权限（1=有, 0=无）'
) COMMENT '员工信息表';

CREATE TABLE appointment (
    appt_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '预约ID（主键）',
    id_card VARCHAR(18) COMMENT '患者身份证号（外键）',
    department VARCHAR(50) COMMENT '就诊科室',
    expect_time DATETIME COMMENT '预计到达时间',
    appt_status VARCHAR(20) COMMENT '预约状态（未核验/已完成/已取消）',
    -- 外键关联患者表
    FOREIGN KEY (id_card) REFERENCES patient(id_card)
) COMMENT '预约信息表';

CREATE TABLE schedule (
    schedule_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '排班ID（主键）',
    room_num INT COMMENT '诊室号',
    department VARCHAR(50) COMMENT '所属科室',
    doctor_id INT COMMENT '医生工号（外键）',
    start_time DATETIME COMMENT '可接诊起始时间',
    end_time DATETIME COMMENT '可接诊截止时间',
    schedule_status VARCHAR(20) COMMENT '排班状态（有效/无效）',
    -- 外键关联员工表（医生）
    FOREIGN KEY (doctor_id) REFERENCES employee(emp_id)
) COMMENT '排班信息表';

CREATE TABLE visit (
    visit_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '就诊ID（主键）',
    id_card VARCHAR(18) COMMENT '患者身份证号（外键）',
    room_num INT COMMENT '诊室号',
    doctor_id INT COMMENT '医生工号（外键）',
    visit_time DATETIME COMMENT '就诊时间',
    visit_status VARCHAR(20) COMMENT '就诊状态（未就诊/就诊中/已离院）',
    -- 外键关联患者表、员工表
    FOREIGN KEY (id_card) REFERENCES patient(id_card),
    FOREIGN KEY (doctor_id) REFERENCES employee(emp_id)
) COMMENT '就诊信息表';

CREATE TABLE bill (
    bill_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '账单ID（主键）',
    visit_id INT COMMENT '就诊ID（外键）',

```

```

total_fee DECIMAL(10,2) COMMENT '费用总额',
medical_insur DECIMAL(10,2) COMMENT '医保金额',
self_pay DECIMAL(10,2) COMMENT '自费金额',
pay_time DATETIME COMMENT '收费时间',
pay_method VARCHAR(20) COMMENT '支付方式（银行卡支付/微信支付/支付宝支付/现金支付）',
actual_pay DECIMAL(10,2) COMMENT '实际收取金额',
-- 外键关联就诊表
FOREIGN KEY (visit_id) REFERENCES visit(visit_id)
) COMMENT '账单信息表';

-- 创建用户
CREATE USER IF NOT EXISTS 'patient'@'localhost'
IDENTIFIED BY '123456';

-- 对目标表逐个赋予所有权限
GRANT ALL PRIVILEGES ON community_hospital.patient TO 'patient'@'localhost';
GRANT ALL PRIVILEGES ON community_hospital.appointment TO 'patient'@'localhost';
GRANT ALL PRIVILEGES ON community_hospital.schedule TO 'patient'@'localhost';
GRANT ALL PRIVILEGES ON community_hospital.visit TO 'patient'@'localhost';
GRANT ALL PRIVILEGES ON community_hospital.bill TO 'patient'@'localhost';

-- 刷新权限
FLUSH PRIVILEGES;

```

- 准备测试数据并批量导入（测试数据较多，不再赘述，已上传本项目[GitHub](#)仓库中的 data.sql 文件）。

4. 数据库应用软件的功能设计和开发

- 编写控制台应用，通过实现应用程序与数据库的交互，开发过程截图如下，完整代码已上传[GitHub](#)，不再赘述。

```

// 科室+起止时间确定范围统计收入
int cnt = 0;
long double total_income = 0;
string department, start_time, end_time;
cout << "请输入统计科室: " << endl; cin >> department;
cout << "请输入查询起始时间 (格式: XXXX-XX-XX XX:XX): " << endl; cin >> start_time;
cout << "请输入查询结束时间 (格式: XXXX-XX-XX XX:XX): " << endl; cin >> end_time;
ostringstream case3;
case3 << "SELECT * FROM bill WHERE pay_time >= '"
    << start_time << "' AND pay_time <= '"
    << end_time << "' AND visit_id IN (SELECT visit_id FROM visit V, schedule S WHERE V.room_num = S.room_num AND V.doctor_id = S.doct
    << department << "')";
sql = case3.str();
res = stmt->executeQuery(sql.c_str());
while (res->next()) {
    cnt++;
    cout << "bill_id: " << res->getInt("bill_id") << endl;
    cout << "total fee: " << res->getDouble("total_fee") << endl;
    cout << "medical_insur: " << res->getDouble("medical_insur") << endl;
    cout << "self_pay: " << res->getDouble("self_pay") << endl;
    cout << "total_income += res->getDouble("self_pay");
    total_income += res->getDouble("self_pay");

    cout << "该时间段内, 科室: " << department << "接诊记录中共有 " << cnt << "条收入记录, " << "共收入 " << total_income << "元" << endl;
    break;
}

default:
    return false;
break;
}

```

输出

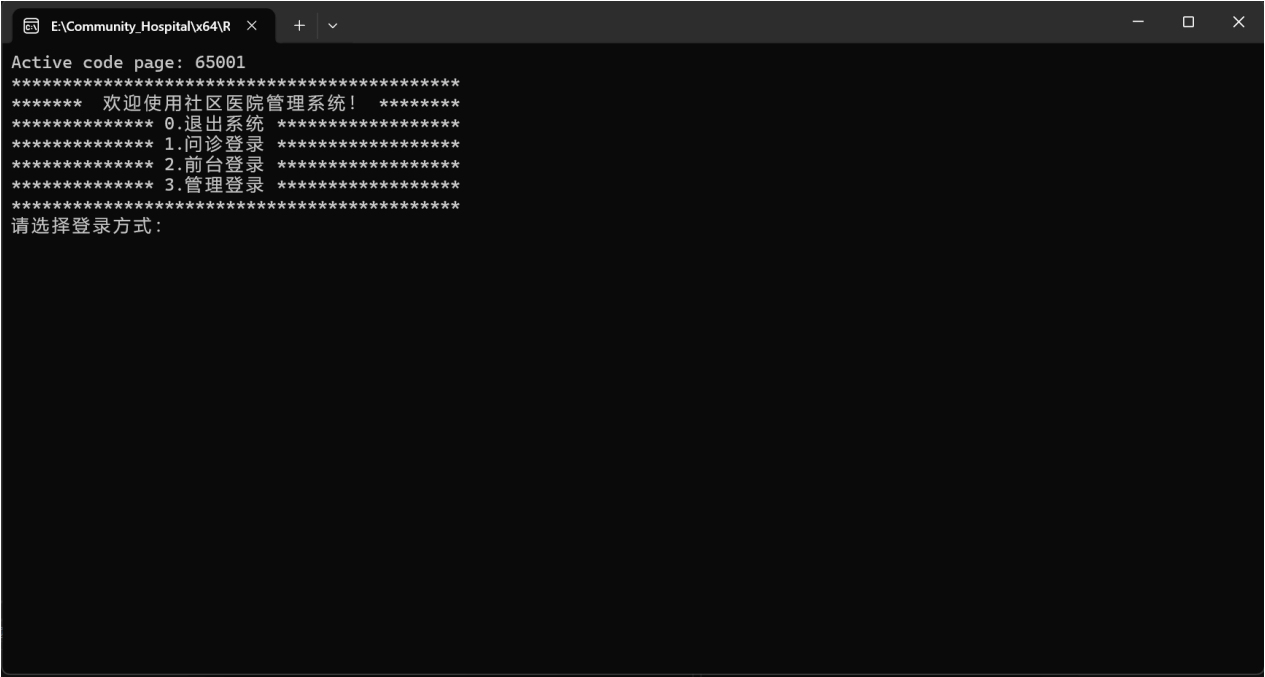
```

显示输出来源(S): 调试
"Community_Hospital.exe" (Win32): 已加载 "C:\Windows\System32\TPHLPAPI.DLL", 包含/排除设置已禁用符号加载。
"Community_Hospital.exe" (Win32): 已加载 "C:\Windows\System32\msi.dll", 包含/排除设置已禁用符号加载。
线程 1972 已退出, 返回值为 3221225786 (0xc000013a)。
线程 3720 已退出, 返回值为 3221225786 (0xc000013a)。
线程 3302 已退出, 返回值为 3221225786 (0xc000013a)。
程序 "[36596] Community_Hospital.exe" 已退出, 返回值为 3221225786 (0xc000013a)。

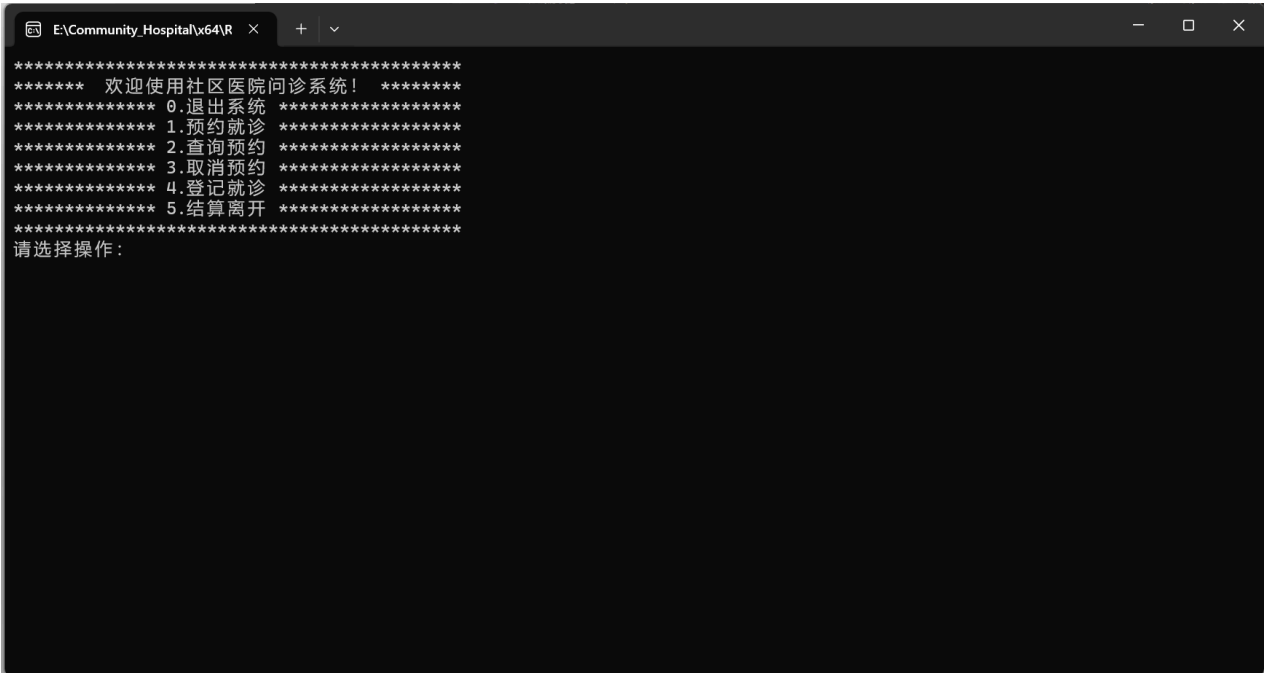
```

5. 数据库应用系统测试

- 进行单元测试、集成测试和系统测试，确保功能正确且无重大Bug，以下仅展示部分主要测试截图，具体测试见PPT与演示视频。
- 系统主入口



- 问诊登录主页面



- 问诊登录查询预约功能

```
E:\Community_Hospital\x64\R  x + v

***** 欢迎使用社区医院问诊系统! *****
***** 0.退出系统 *****
***** 1.预约就诊 *****
***** 2.查询预约 *****
***** 3.取消预约 *****
***** 4.登记就诊 *****
***** 5.结算离开 *****
*****
请选择操作:
2
预约号: 1
身份证号: 110101198001010011
科室: 内科
预约时间: 2025-12-05 09:15:30
预约状态: 未核验
*****
***** 欢迎使用社区医院问诊系统! *****
***** 0.退出系统 *****
***** 1.预约就诊 *****
***** 2.查询预约 *****
***** 3.取消预约 *****
***** 4.登记就诊 *****
***** 5.结算离开 *****
*****
请选择操作:
```

• 问诊登录结算离开功能

```
E:\Community_Hospital\x64\R  x + v

请选择操作:
2
预约号: 1
身份证号: 110101198001010011
科室: 内科
预约时间: 2025-12-05 09:15:30
预约状态: 未核验
*****
***** 欢迎使用社区医院问诊系统! *****
***** 0.退出系统 *****
***** 1.预约就诊 *****
***** 2.查询预约 *****
***** 3.取消预约 *****
***** 4.登记就诊 *****
***** 5.结算离开 *****
*****
请选择操作:
5
医疗总费用: 268.5元; 其中医保承担: 188元; 您需支付: 80.5元。
*****
***** 欢迎使用社区医院问诊系统! *****
***** 0.退出系统 *****
***** 1.预约就诊 *****
***** 2.查询预约 *****
***** 3.取消预约 *****
***** 4.登记就诊 *****
***** 5.结算离开 *****
*****
请选择操作:
```

• 分诊台登录过程

```
E:\Community_Hospital\x64\R  ×  +  ▾

***** 欢迎使用社区医院管理系统! *****
***** 0.退出系统 *****
***** 1.问诊登录 *****
***** 2.前台登录 *****
***** 3.管理登录 *****
*****
请选择登录方式:
2
请输入你的工号:
1
您没有分诊台登录权限! 请选择其他登录方式。
***** 欢迎使用社区医院管理系统! *****
***** 0.退出系统 *****
***** 1.问诊登录 *****
***** 2.前台登录 *****
***** 3.管理登录 *****
*****
请选择登录方式:
2
请输入你的工号:
2
请输入分诊台统一登录密码:
reception123|
```

• 分诊台主页面

```
E:\Community_Hospital\x64\R  ×  +  ▾

***** 欢迎使用社区医院分诊系统! *****
***** 0.退出系统 *****
***** 1.患者就诊 *****
***** 2.预约管理 *****
***** 3.就诊状态管理与账单结算 *****
***** 4.收费收入记录与人次记录 *****
***** 5.查询患者简要信息 *****
*****
请选择操作:
```

• 分诊台查询患者简要信息功能


```
E:\Community_Hospital\x64\R  × + ▾

***** 欢迎使用社区医院分诊系统! *****
***** 0.退出系统 *****
***** 1.患者就诊 *****
***** 2.预约管理 *****
***** 3.就诊状态管理与账单结算 *****
***** 4.收费收入记录与人次记录 *****
***** 5.查询患者简要信息 *****
*****
请选择操作:
5
请输入需要查询的患者身份证号:
110101198001010011
姓名: 张伟
就诊科室: 内科
诊室号: 1001
就诊状态: 已离院
***** 欢迎使用社区医院分诊系统! *****
***** 0.退出系统 *****
***** 1.患者就诊 *****
***** 2.预约管理 *****
***** 3.就诊状态管理与账单结算 *****
***** 4.收费收入记录与人次记录 *****
***** 5.查询患者简要信息 *****
*****
请选择操作:
```

• 行政管理登录过程

```
E:\Community_Hospital\x64\R  × + ▾

***** 欢迎使用社区医院管理系统! *****
***** 0.退出系统 *****
***** 1.问诊登录 *****
***** 2.前台登录 *****
***** 3.管理登录 *****
*****
请选择登录方式:
3
请输入你的工号:
1
您没有行政管理登录权限! 请选择其他登录方式。
***** 欢迎使用社区医院管理系统! *****
***** 0.退出系统 *****
***** 1.问诊登录 *****
***** 2.前台登录 *****
***** 3.管理登录 *****
*****
请选择登录方式:
3
请输入你的工号:
3
请输入行政管理统一登录密码:
manage123
```

• 行政管理主页面

```
E:\Community_Hospital\x64\R  × + v
***** 欢迎使用社区医院行政系统! *****
***** 0.退出系统 *****
***** 1.排班管理 *****
***** 2.账单查询 *****
***** 3.查询患者信息 *****
***** 4.查询员工信息 *****
***** 5.修改员工信息 *****
*****
请选择操作:
```

• 行政管理账单查询功能查询过程

```
E:\Community_Hospital\x64\R  × + v
***** 欢迎使用社区医院行政系统! *****
***** 0.退出系统 *****
***** 1.排班管理 *****
***** 2.账单查询 *****
***** 3.查询患者信息 *****
***** 4.查询员工信息 *****
***** 5.修改员工信息 *****
*****
请选择操作:
2
请选择操作方式: 1.时间段内收入; 2.医生+时间段内收入; 3.科室+时间段内收入
1
请输入查询起始时间 (格式: XXXX-XX-XX_XX:XX:XX) :
2025-12-01_00:00:00
请输入查询结束时间 (格式: XXXX-XX-XX_XX:XX:XX) :
2025-12-31_23:59:59
账单号: 51
总开销: 328.5
医保承担: 229.95
自费部分: 98.55
-----
账单号: 52
总开销: 186
医保承担: 130.2
自费部分: 55.8
-----
账单号: 53
总开销: 920
医保承担: 644
```

• 行政管理账单查询功能统计结果

```
E:\Community_Hospital\x64\R  × + ▾
医保承担：203
自费部分：87
-----
账单号：92
总开销：820.6
医保承担：574.42
自费部分：246.18
-----
账单号：93
总开销：570.8
医保承担：399.56
自费部分：171.24
-----
账单号：94
总开销：1450
医保承担：1015
自费部分：435
-----
该时间段内共有30条收入记录，共收入6138.93元
***** 欢迎使用社区医院行政系统！ *****
***** 0.退出系统 *****
***** 1.排班管理 *****
***** 2.账单查询 *****
***** 3.查询患者信息 *****
***** 4.查询员工信息 *****
***** 5.修改员工信息 *****
*****
请选择操作：
```

6. 应用软件、数据库和文档提交

- 打包所有源码、数据库脚本和文档进行归档，详见(https://github.com/Nitroglycerine-X/Community_Hospital)。

7. 应用系统演示和汇报

- 详见[测试录屏](#)与PPT内容。

五、实验思考题

1. 功能与数据的关系：数据库应用系统的功能设计对数据库设计有何影响？

- 数据查询功能对数据库设计的影响。
索引设计：频繁的查询功能是索引设计的核心依据。例如，用户登录的“账号密码验证”功能，会要求在用户表的账号字段上建立唯一索引，以提升查询效率。反之，若某字段仅用于数据录入而极少被查询，则无需建立索引，避免增加写入开销。
视图与物化视图设计：复杂的统计查询功能会促使设计人员创建视图，将多表关联、聚合计算的逻辑封装在视图中，简化应用程序的查询代码；对于超大规模数据的统计查询，还可能设计物化视图（如 MySQL 的汇总表），提前缓存计算结果以提升查询性能。
分表分库策略：系统存在“按时间范围查询不同医生/科室收入”的功能，且订单数据量极大，会采用按时间分表（如按年、按月分表）的设计；若存在“按地区查询用户”的功能，可能采用按地区分库的设计，降低单表/单库的数据量，提升查询速度。
- 业务逻辑复杂度对数据库结构的影响。
表结构设计：复杂的业务逻辑会直接改变表的结构和关系。例如，电商系统的“订单支付”功能若支持多种支付方式（微信、支付宝、银行卡），则需设计支付方式表与订单表关联，而非在订单表中直接用字段存储支付方式；若业务要求记录订单的“创建、支付、发货、完成”等状态流转，需在订单表中增加状态字段，并可能设计订单状态日志表记录状态变更历史。
冗余字段设计：为简化复杂业务逻辑的实现，可能会在表中增加冗余字段。例如，社交系统的“朋友圈动态”功能，若频繁需要显示发布者的昵称和头像，会在动态表中冗余用户昵称和头像URL字段，而非每次查询都关联用户表，以减少联表查询的复杂度。

存储引擎选择：若业务逻辑涉及事务（如转账、订单支付），需选择支持事务的 InnoDB 存储引擎；若仅为日志记录等无需事务的功能，可选择 MyISAM 存储引擎以提升写入性能。

- 数据操作功能对数据库设计的影响。

写入 / 更新频率：高频写入的功能（如系统日志记录）会影响数据库的存储策略，例如采用批量插入、分区表等方式降低写入压力；高频更新的功能（如商品库存修改）会要求表结构尽量精简，避免大字段（如 TEXT、BLOB）存在于频繁更新的表中，以减少 IO 开销。

数据删除策略：若功能要求“软删除”（如用户注销但保留数据），则需在表中增加删除标记字段（如 is_deleted），而非物理删除数据；若功能要求“定期清理过期数据”，则需设计定时任务，并结合分表策略简化删除操作。

- 权限与安全功能对数据库设计的影响。

若系统有“按角色分配操作权限”的功能，需设计用户表、角色表、权限表及多对多的关联表（用户-角色表、角色-权限表），以实现细粒度的权限控制；若功能要求记录用户的操作日志，需设计操作日志表，记录操作人、操作时间、操作内容等信息。

2. 协调设计：如何协调数据库应用系统的功能设计与数据库设计？

- 前期：需求驱动的整体规划，奠定协同基础。

统一需求分析：先通过需求调研明确系统的核心功能模块（如用户管理、订单管理、商品管理）和核心数据实体（用户、订单、商品），绘制用例图和实体关系草图（E-R 草图），让功能设计和数据库设计的人员对系统有统一的认知。

核心实体优先设计：针对核心功能对应的核心数据实体，先完成概念结构设计（E-R 图）和初步的逻辑结构设计（关系模式），确定核心表的主键、主要字段和关联关系，为功能模块的开发提供基础数据模型。

技术选型协同：功能设计的技术栈（如 Java Spring Boot、Python Django）和数据库选型（如 MySQL、PostgreSQL）需同步确定，避免因技术栈差异导致后续数据库设计调整（例如 Django ORM 对 MySQL 数据类型的适配要求）。

- 中期：迭代式开发，动态协调调整。

在敏捷开发模式下，功能设计和数据库设计通常以迭代的方式推进，而非“一次性定版”，具体流程为：

小步快跑：将项目划分为多个迭代周期（如 2 周一个迭代），每个迭代完成一个小的功能模块（如“用户注册登录”）。

同步设计与开发：针对当前迭代的功能模块，开发人员先明确功能的输入输出、业务逻辑，DBA 同步设计对应的数据库表结构、索引等；开发人员基于初步的数据库设计编写代码，若在开发中发现数据模型无法满足功能需求（如缺少字段、关联关系不合理），及时与 DBA 沟通调整。

原型验证：针对核心功能，可先搭建原型系统，通过实际的功能测试验证数据库设计的合理性（如查询效率、表结构是否满足业务逻辑），并快速迭代优化。

这种“先定核心数据模型，再迭代优化”的方式，既避免了“先定表结构再写代码”的僵化（可能导致表结构无法适配功能变化），也避免了“先写代码再设计表结构”的混乱（可能导致数据模型不规范、冗余严重）。

- 后期：团队协作与文档同步，保障一致性。

建立沟通机制：设立 DBA 与开发人员的日常沟通渠道（如专门的沟通群），及时解决功能设计与数据库设计的冲突；对于重大的设计变更，需组织评审会，确保所有相关人员达成一致。

文档同步更新：功能设计文档（如功能规格说明书）和数据库设计文档（如数据字典、E-R 图）需同步更新，记录每次变更的原因和内容，避免因文档不一致导致后续开发混乱。

数据库重构：在系统开发过程中，若功能需求发生重大变化，可通过数据库重构（如增加字段、修改索引、调整表关联关系）适配功能需求，但需注意通过数据迁移脚本保证现有数据的兼容性。

- 特殊场景：分阶段侧重不同设计。

小型项目：可采用“先设计核心表结构，再开发功能”的方式，简化流程，提高效率。

大型项目：需成立专门的设计团队，功能设计和数据库设计同步推进，通过领域驱动设计（DDD）将业务领域模型转化为数据模型和功能模块，实现二者的深度协同。

3. 类型映射：数据库应用开发中使用的程序设计语言(C++)的数据类型与数据库 (MySQL) 的数据类型如何互操作？

- MySQL 官方提供的 C++ 封装库，封装性好，易用性高，自动处理部分类型转换。
面向对象接口： `Connection`、`Statement`、`ResultSet` 等类
自动类型转换： `ResultSet::getInt()`、`getString()`、`getDouble()` 等方法直接返回 C++ 类型
空值处理： `ResultSet::isNull()` 判断字段是否为 `NULL`
- 在使用上述Connector/C++库时，处理数据库中的 `DATE`/`TIME`/`DATETIME`/`TIMESTAMP` 等类型数据需要使用 `getString()`。
例如，本系统中设计 排班时间 就诊时间 支付时间 时用到数据库中的 `DATETIME` 类型，使用C++中的 `String` 类型传入时间需要按照 `XXXX-XX-XX_XX:XX:XX` 的格式。

六、自由探索任务

1. 版本控制 (Git)

- 在GitHub上建立仓库，使用Git进行代码管理。

2. Mock数据生成

- 使用大语言模型生成体现测试功能的多条数据。

实验总结

1. 总结

本次实验是对本学期数据库课程内容的全面验收。综合应用数据库语句与程序开发语言，从数据 (Data) 和功能 (Function) 两个维度协调设计一个完整的系统。

2. 改进方向

本系统在权限控制与管理方面，仍主要依赖于程序设计过程中的接口限制，对于数据库自主用户权限控制方面还可以继续细化，做到权限精准分配，从源头上杜绝越权操作。