

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 医院管理系统设计 |
| 姓 名： | 周大伟 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 校交1803 |
| 学 号： | U201815553 |
| 指导教师： | 江胜 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2021年 6月 19日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

**目 录**

[1.课程任务概述 4](#_Toc29010)

[1.1 软件功能学习 4](#_Toc23754)

[1.2 SQL练习 4](#_Toc26203)

[2.软件功能学习 11](#_Toc20527)

[2.1 备份 11](#_Toc32713)

[2.2 用户权限 13](#_Toc26537)

[3.SQL练习 15](#_Toc21045)

[3.1 创建数据库和表 15](#_Toc22906)

[3.2 表的更新 17](#_Toc9885)

[3.3 数据查询 17](#_Toc12111)

[3.4 创建触发器 24](#_Toc14872)

[3.5 创建并使用自己创建的函数 25](#_Toc2965)

[4.应用系统设计 26](#_Toc24645)

[4.1系统设计目标 26](#_Toc27308)

[4.2 需求分析 26](#_Toc17986)

[4.3 总体设计 31](#_Toc18065)

[4.4 数据库设计 35](#_Toc16928)

[4.5 详细设计与实现 40](#_Toc30961)

[4.6 系统测试 52](#_Toc15153)

[4.7 系统设计与实现总结 56](#_Toc22216)

[5.课程总结 57](#_Toc22956)

# 1.课程任务概述

## 1.1 软件功能学习

1. 练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式：数据和日志文件的脱机备份、系统的备份功能。
2. 练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 1.2 SQL练习

### 1.2.1 创建数据库和表

**第1关 创建用于新冠疫情常态防控的数据库**

1. 任务描述：创建数据库。
2. 数据库信息说明：创建用于管理新冠疫情防控的数据库:covid19mon。

**第2关 创建表和主外码约束**

1. 任务描述：建立新冠常态防控信息系统所用到的表，为表设置主码、外码，并定义主外码之间的参照完整性。
2. 表信息说明：

表 1-1 人员表(person)



表 1-2 地点表(location)



表 1-3 行程表(itinerary)



表 1-4 诊断表(diagnose\_record)



表 1-5 密切接触者表(close\_contact)



表 1-6 隔离表(isolation\_record)



表 1-7 隔离地点表(isolation\_location)



### 1.2.2 表的更新

**第1关 数据插入**

1. 任务描述：练习insert语句,向人员表person插入数据。
2. 插入信息：

表 1-8 插入人员数据



**第2关 数据删除**

任务描述：删除前一关中插入的人员编号为2，姓名为’李大锤’的这条数据。

**第3关 修改数据**

任务描述：将姓名为“张小敏”,人员编号为1的电话号码改为13607176668。

### 1.2.3 数据查询

**第1关 人流量大于30的地点**

1. 任务描述：查询累计人流量大于30的地点名称和累计人流量。
2. 编程要求：查询累计人流量大于30的地点名称和累计人流量，请用visitors作累积人流量的标题名称。查询结果按照人流量从高到低排序,人流量相同时，依地点名称顺序排序。

**第2关 每个隔离点正在进行隔离的人数**

1. 任务描述：查询每个隔离地中正在进行隔离的人数，并按数量由多到少排序。
2. 编程要求：查询每个隔离地及该地正在进行隔离的人数，以number为隔离人数的标题.查询结果依隔离人数由多到少排序，隔离人数相同时，再依隔离点名称顺序排序。

**第3关 接续行程**

1. 任务描述：查询行程表中同一个人接续行程的时间和地点信息。所谓接续行程指同一个人第一段行程的结束时间与第二段行程的开始时间重合。
2. 编程要求：查询人员编号大于30的接续行程，输出信息包括： 人员编号,姓名,重合时间,起始地点id,起始地点,结束地点id,结束地点。查询结果依人员编号排序。

**第4关 充珉瑶和贾涵山的行程情况**

1. 任务描述：查询充珉瑶和贾涵山的行程情况。
2. 编程要求：查询充珉瑶和贾涵山的行程情况。查询结果包括：姓名、电话、到过什么地方（地名），何时到达，何时离开。列名原样列出即可，不必用别名。查询结果依人员编号降序排序。同一人员行程依行程开始时间顺序排列。没有任何出行记录的，在出行有关的栏目内均填写NULL即可。

**第5关 地名中带有‘店’字的地点名称**

1. 任务描述：查询地名中带有‘店’字的地点编号和名称。查询结果按地点编号排序。
2. 编程要求：查询地名中带有‘店’字的地点编号和名称。查询结果按地点编号排序。

**第6关 确诊者的接触者**

1. 任务描述：查询确诊者的接触者。
2. 编程要求：新发现一位确诊者，已知他在2021.2.2日20:05:40到21:25:40之间在“活动中心”逗留，凡在此间在同一地点逗留过的，视为接触者，请查询接触者的姓名和电话。查询结果按姓名排序。

**第7关 仍在使用的隔离点**

1. 任务描述：查询仍正在使用的隔离点名称。
2. 编程要求：查询仍在使用的隔离点名称。注意，隔离记录里如果只有隔离结束或确诊转院的记录，表明该隔离点已暂时停用，只要还有一个人在此处隔离，表明该隔离点仍在使用。查询结果按隔离点编号排序。

**第8关 查询有出行记录的人员**

1. 任务描述：用带EXISTS关键字的子查询,查询有有出行记录的人员及其联系电话。
2. 编程要求：查询前30位有出行记录的人员姓名和电话。查询结果按照人员编号排序。

**第9关 没有去过“Today便利店“的人数**

1. 任务描述：统计查询人员表中没有去过地点“Today便利店”的人数（使用NOT EXISTS 关键字）。
2. 编程要求：查询人员表中没有去过地点“Today便利店”的人数。请给统计出的人数命名为number。

**第10关 去过所有地点的人员**

1. 任务描述：查询人员表去过所有地点的人员姓名。查询结果依人员姓名的字典顺序排序。
2. 编程要求：查询人员表去过所有地点的人员姓名。查询结果依人员姓名顺序排序。

**第11关 隔离点的现状视图**

1. 任务描述：建立隔离点现状的视图，视图命名为isolation\_location\_status，内容包括：地点编号，隔离地点名，房间容量，已占用量。请保持原列名不变，已占用量由统计函数计算得出，该计算列命名为occupied。
2. 编程要求：

视图名称：isolation\_location\_status

内容包括：隔离地点编号，隔离地点名，房间容量，已占用量。

请保持原列名不变，已占用量由统计函数计算得出，该计算列命名为occupied。只有正在该隔离点隔离的人才占用隔离点的位置。隔离结束或已转院的人表明已经腾出了原有位置，不再占用资源。

**第12关 各隔离点的剩余房间数**

1. 任务描述：从视图isolation\_location\_status中查询各隔离点的剩余房间数。
2. 编程要求：从视图isolation\_location\_status中查询各隔离点的剩余空房间的数目。需要列出的数据项为：隔离点名称，剩余房间数。其中剩余房间数为计算得出，请给该列命名为available\_rooms。查询结果依隔离点编号排序。

**第13关 与无症状感染者靳宛儿有过接触的人**

1. 任务描述：筛查发现，靳宛儿为无症状感染者。现需查询其接触 者姓名和电话，以便通知并安排隔离。凡行程表中，在同一地点逗留时间与靳宛儿有交集的，均视为接触者。
2. 编程要求：查询靳宛儿接触者的姓名和电话。与靳宛儿在同一地点逗留时间有交集的均为其接触者。查询结果按照人员姓名排序。

**第14关 每个地点发生的密切接触者人数**

1. 任务描述：查询每个地点的密切接触者的数量，列出内容包括：地点名称，密接者人数。
2. 编程要求：依据密切接触表的内容查询每个地点的密切接触者的数量，列出内容包括：地点名称，密接者人数。人数由统计获得，列名命名为close\_contact\_number.查询结果依密接者人数降序排列。密接者人数相同时，依地点名称排序。

**第15关 感染人数最多的人**

1. 任务描述：查询感染人数最多的人。
2. 编程要求：查询感染人数最多的人员编号，姓名，和被其感染的人数。感染人数由统计所得，命名为infected\_number。

**第16关 行程记录最频繁的3个人**

1. 任务描述：查询2021-02-02 10:00:00到14:00:00期间，行程记录最频繁的三个人及行程记录的条数。
2. 编程要求：查询2021-02-02 10:00:00到14:00:00期间，行程记录最频繁的三个人的姓名及行程记录条数。记录条数命名为record\_number。如果出现记录数并列情况，再按姓名排序。

**第17关 房间数第2多的隔离点**

1. 任务描述：查询隔离点中，房间数(capacity)居第二多的隔离点名称及其房间数。
2. 编程要求：从隔离点中，查询房间数(capacity)居第二多的隔离点名称及其房间数。
3. 注意：如果房间数从多到少依次有100,100,80,80,70,…，那么，两个80都是第2多。而两个100都是第1多。

### 1.2.4 创建触发器

**第1关 隔离点的人员“确诊新冠”后，自动转院**

1. 任务描述：

隔离点的人员一旦确诊“新冠”后，将被转入医院，请编写一个触发器，用于实现以下完整性控制规则：

当隔离表（isolation\_record）中的某位隔离人员在诊断表（diagnose\_record）中的诊断结果（result）为1(新冠确诊)”时，自动将隔离表中的隔离状态（state)改成3（转入医院）。

1. 编程要求：

用create trigger语句创建符合任务要求的触发器（触发器名称自已命名）：

当隔离表（isolation\_record）中的某位隔离人员在诊断表（diagnose\_record）中的诊断结果（result）为1(新冠确诊)”时，自动将隔离表中的隔离状态（state)改成3（转入医院）。

### 1.2.5 创建并使用自己创建的函数

**第1关 创建函数并在语句中使用它**

1. 任务描述：
2. 编写一个依据人员编号计算其到达所有地点的次数(即行程表中的记录数)的自定义函数，同一人员到达同一地点多次，去几次算几次。
3. 并利用其查询至少有3条行程记录的人员。
4. 编程要求：
5. 用create function语句创建符合以下要求的函数：

·依据人员编号计算其到达所有地点的次数(即行程表中的记录数)。

·函数名为：Count\_Records。函数的参数名可以自己命名

1. 利用创建的函数，仅用一条SQL语句查询在行程表中至少有3条行程记录的人员信息，查询结果依人员编号排序。

# 2.软件功能学习

**实验环境**

操作系统：Windows 10 家庭中文版

数据库：MySQL

交互图形界面：Navicat for MySQL 15.0.23

## 2.1 备份

### 2.1.1 数据导入

按照实验要求建立相关的数据库和表。

下载群文件夹中的 MySQLl实训数据批量插入.sql文件，添加到Navicat的查询列表中，执行。

得到如图 2-1 插入数据结果：

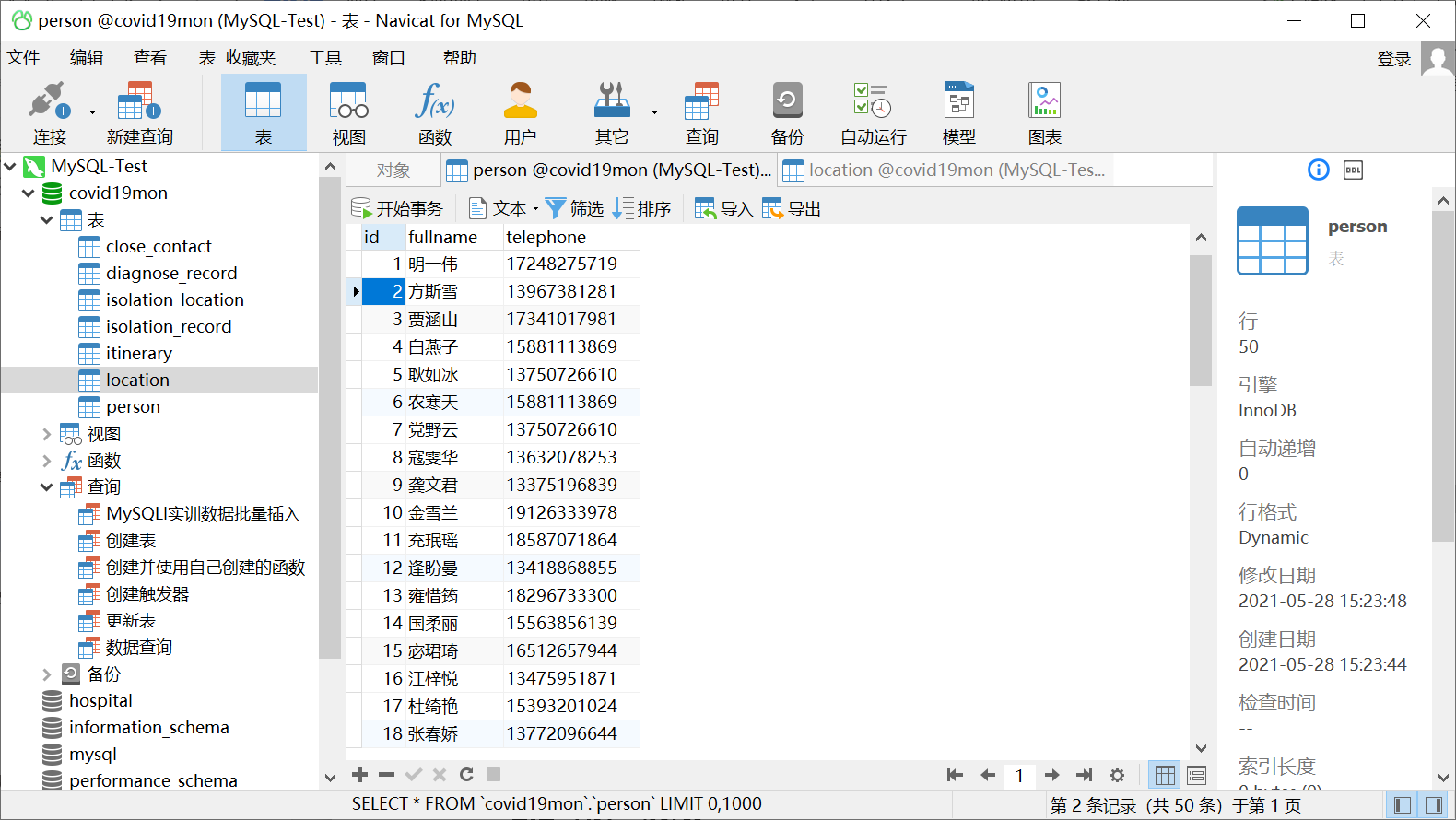


图 2-1 插入数据结果

### 2.1.2 热备份

在Navicat界面中新建备份，并选中需要备份的表、视图、函数和事件，我选择了备份所有的文件。然后会得到一个.nb3文件。

备份后得到如图 2-2 热备份结果：

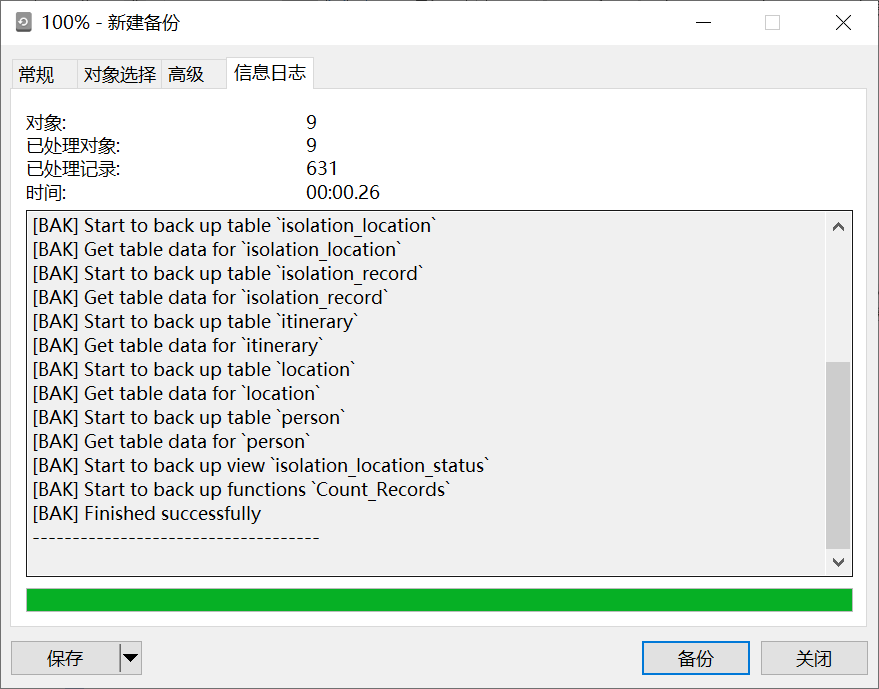


图 2-2 热备份结果

### 2.1.3 冷备份

直接从MySQL目录下复制所要备份的文件，这种备份方式不需要登陆数据库。

图 2-3 MySQL数据文件路径：

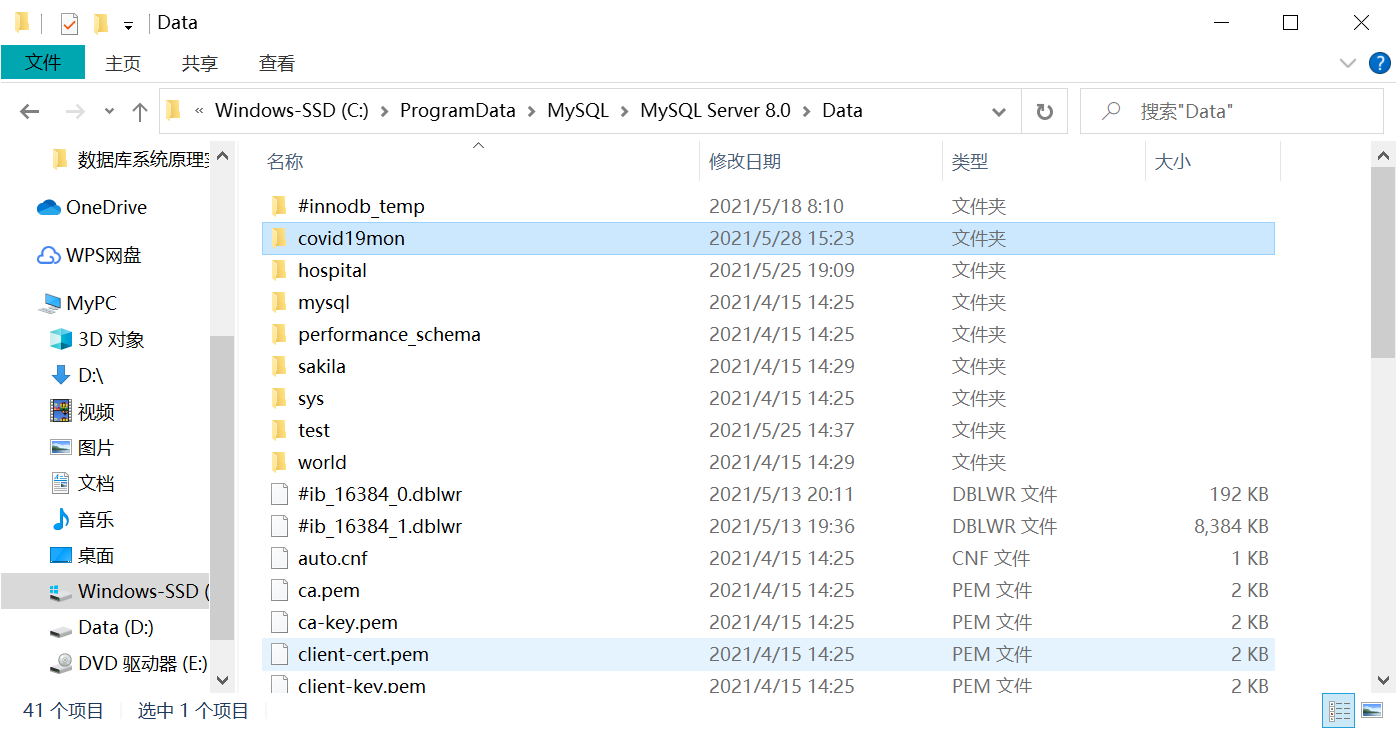


图 2-3 MySQL数据文件路径

## 2.2 用户权限

在Navicat界面中新建用户，并选择用户的操作权限。

得到如图 2-4 新建用户信息和图 2-5 新建用户权限。

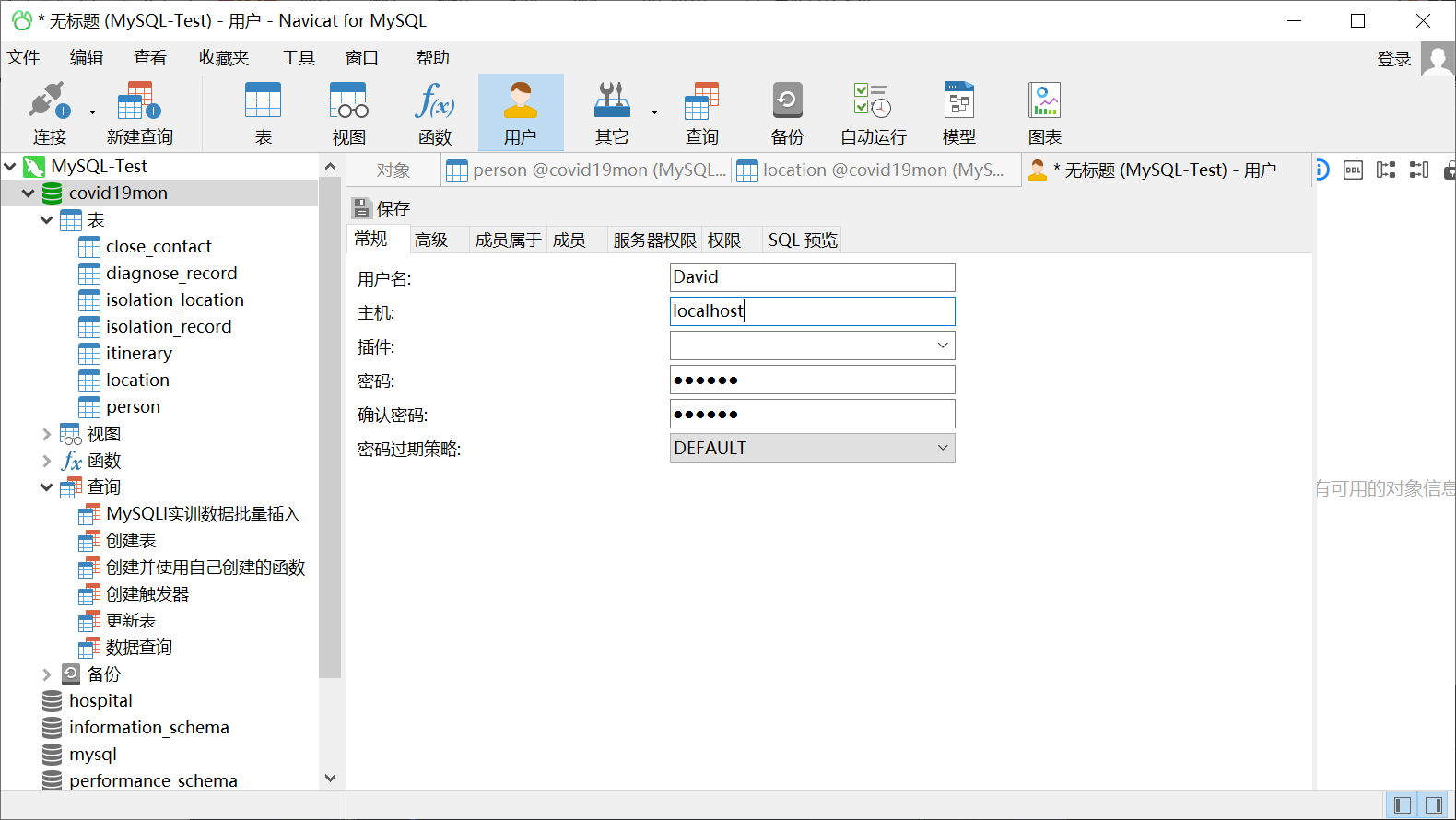


图 2-4 新建用户信息

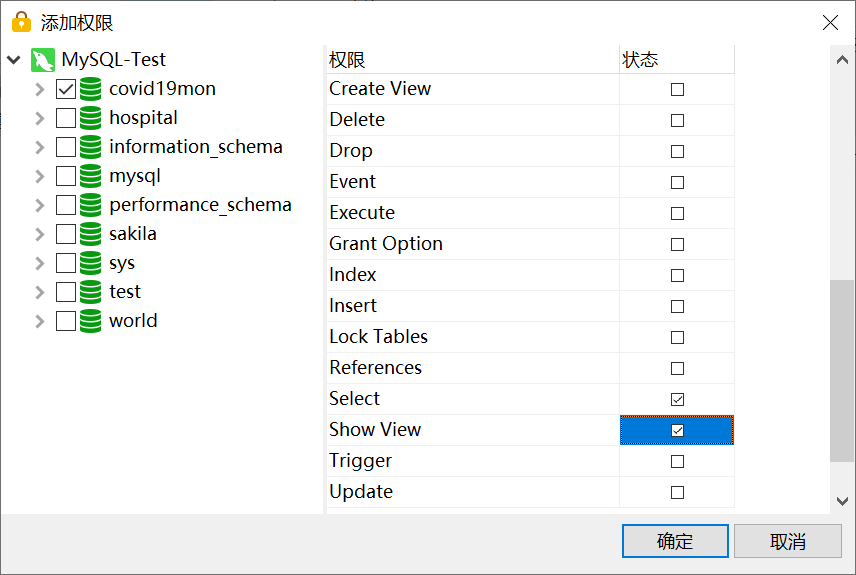


图 2-5 新建用户权限

由以上可以看出，所新建用户的权限只限于查询操作，通过用查询、插入、删除指令，验证权限的正确性。

如图 2-6 删除表操作所示，用户David没有删除表的权限：

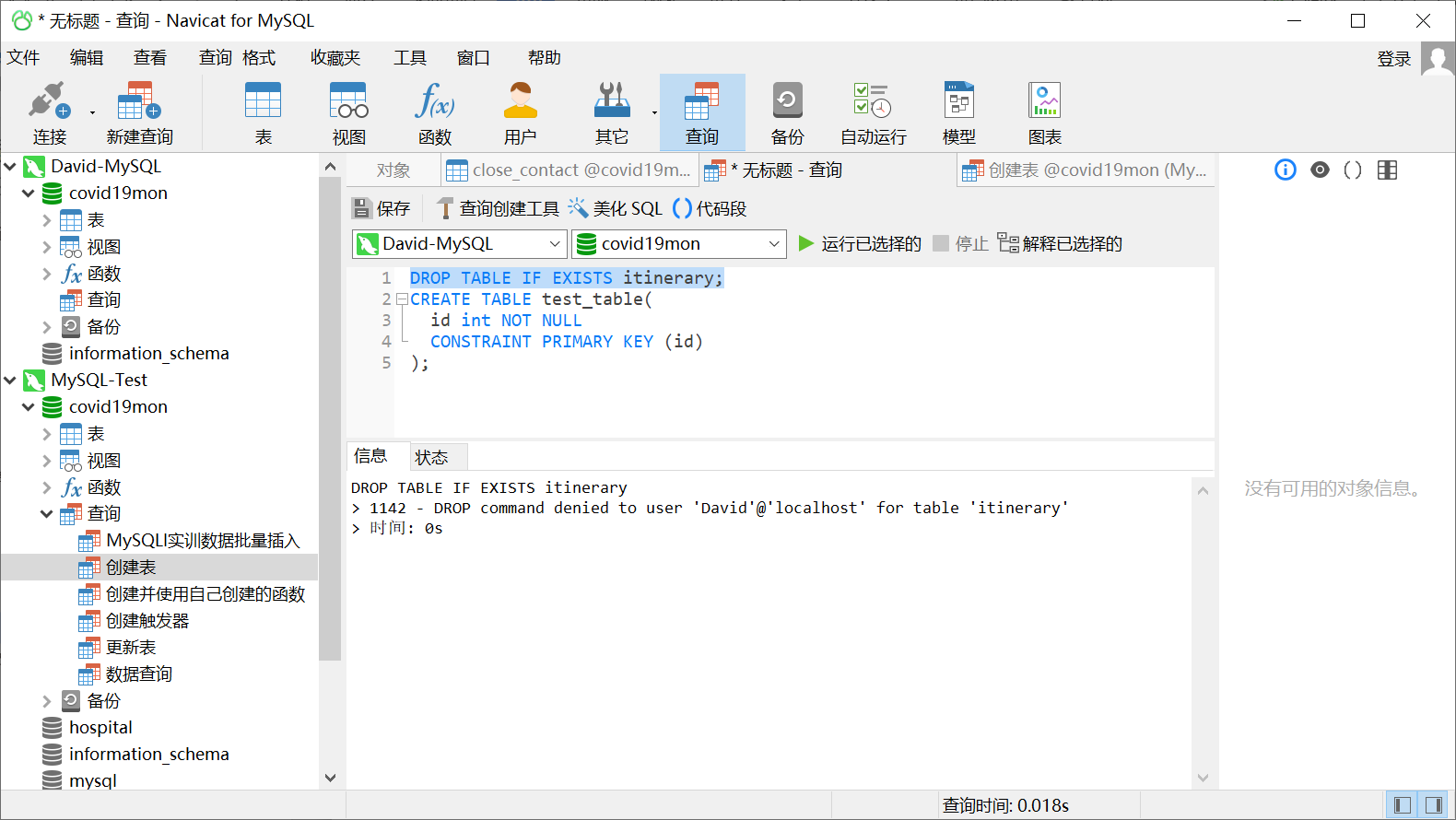


图 2-6 删除表操作

如图 2-7 新建表操作所示：用户David没有新建表的权限：

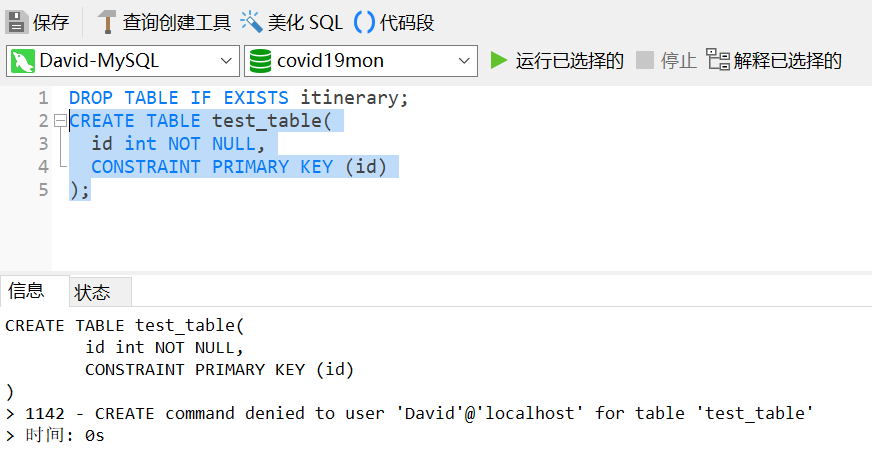


图 2-7 新建表操作

如图 2-8 查询操作所示，用户David可以对表进行查询操作：

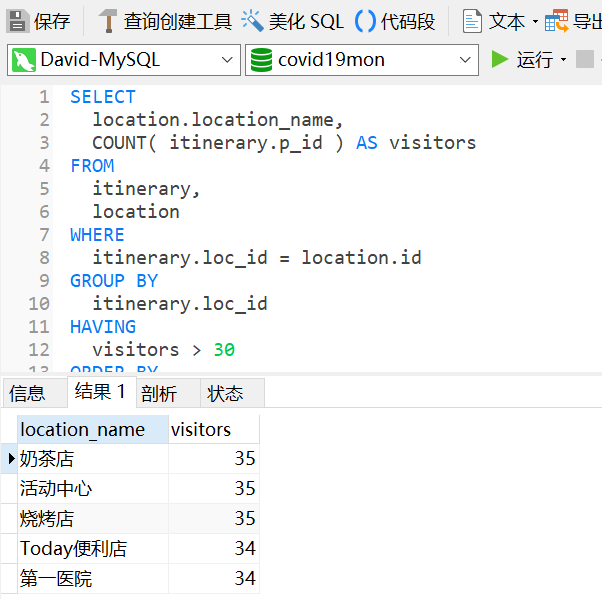


图 2-8 查询操作

# 3.SQL练习

## 3.1 创建数据库和表

### **第1关 创建用于新冠疫情常态防控的数据库**

create database covid19mon;

### 第2关 创建表和主外码约束

-- 表1 人员表(person)

CREATE TABLE person(

  id int NOT NULL,

  fullname char(20) NOT NULL,

  telephone char(11) NOT NULL,

  CONSTRAINT pk\_person PRIMARY KEY (id)

);

-- 表2 地点表(location)

CREATE TABLE location(

    id int NOT NULL,

    location\_name char(20) NOT NULL,

    CONSTRAINT pk\_location PRIMARY KEY (id)

);

-- 表3 行程表（itinerary）

CREATE TABLE itinerary(

    id int NOT NULL,

    p\_id int NULL,

    loc\_id int NULL,

    s\_time datetime NULL,

    e\_time datetime NULL,

    CONSTRAINT pk\_itinerary PRIMARY KEY (id),

    CONSTRAINT fk\_itinerary\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_itinerary\_lid FOREIGN KEY (loc\_id) REFERENCES location (id)

);

-- 表4 诊断表（diagnose\_record）

CREATE TABLE diagnose\_record(

    id int NOT NULL,

    p\_id int NULL,

    diagnose\_date datetime NULL,

    result int NULL,

    CONSTRAINT pk\_diagnose\_record PRIMARY KEY (id),

    CONSTRAINT fk\_diagnose\_pid FOREIGN KEY (p\_id) REFERENCES person (id)

);

-- 表5 密切接触者表（close\_contact）

CREATE TABLE close\_contact(

    id int NOT NULL,

    p\_id int NULL,

    contact\_date datetime NULL,

    loc\_id int NULL,

    case\_p\_id int NULL,

    CONSTRAINT pk\_close\_contact PRIMARY KEY (id),

    CONSTRAINT fk\_contact\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_contact\_lid FOREIGN KEY (loc\_id) REFERENCES location (id),

    CONSTRAINT fk\_contact\_caseid FOREIGN KEY (case\_p\_id) REFERENCES person (id)

);

-- 表6 隔离地点表（isolation\_location）

CREATE TABLE isolation\_location(

    id int NOT NULL,

    location\_name char(20) NULL,

    capacity int NULL,

    CONSTRAINT pk\_isolation\_loc PRIMARY KEY (id)

);

-- 表7 隔离表（isolation\_record）

CREATE TABLE isolation\_record(

    id int NOT NULL,

    p\_id int NULL,

    s\_date datetime NULL,

    e\_date datetime NULL,

    isol\_loc\_id int NULL,

    state int NULL,

    CONSTRAINT pk\_isolation PRIMARY KEY (id),

    CONSTRAINT fk\_isolation\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_isolation\_lid FOREIGN KEY (isol\_loc\_id) REFERENCES isolation\_location (id)

);

## 3.2 表的更新

### 第1关 数据插入

INSERT INTO person (id, fullname, telephone) VALUES (1, '张小敏', '13907110001');

INSERT INTO person (id, fullname, telephone) VALUES (2, '李大锤', '18907110002');

INSERT INTO person (id, fullname, telephone) VALUES (3, '孙二娘', '13307100003');

### 第2关 数据删除

DELETE FROM person WHERE id=2;

### 第3关 修改数据

UPDATE person SET telephone='13607176668' WHERE id=1;

## 3.3 数据查询

### 第1关 人流量大于30的地点

SELECT

    location.location\_name,

    COUNT( itinerary.p\_id ) AS visitors

FROM

    itinerary,

    location

WHERE

    itinerary.loc\_id = location.id

GROUP BY

    itinerary.loc\_id

HAVING

    visitors > 30

ORDER BY

    visitors DESC,

    location.location\_name ASC;

### 第2关 每个隔离点正在进行隔离的人数

SELECT

    isolation\_location.location\_name,

    COUNT( isolation\_record.p\_id ) AS number

FROM

    isolation\_record,

    isolation\_location

WHERE

    isolation\_record.isol\_loc\_id = isolation\_location.id

    AND isolation\_record.state = 1

GROUP BY

    isolation\_location.id

ORDER BY

    number DESC,

    isolation\_location.location\_name ASC;

### 第3关 接续行程

SELECT

    person.id,

    person.fullname,

    person.telephone,

    start\_itinerary.e\_time AS reclosing\_time,

    start\_location.id AS loc1,

    start\_location.location\_name AS address1,

    end\_location.id AS loc2,

    end\_location.location\_name AS address2

FROM

    person,

    itinerary AS start\_itinerary,

    itinerary AS end\_itinerary,

    location AS start\_location,

    location AS end\_location

WHERE

    start\_itinerary.p\_id > 30

    AND start\_itinerary.p\_id = person.id

    AND end\_itinerary.p\_id = person.id

    AND start\_itinerary.loc\_id = start\_location.id

    AND end\_itinerary.loc\_id = end\_location.id

    AND start\_itinerary.e\_time = end\_itinerary.s\_time

ORDER BY

    person.id,

    reclosing\_time;

### 第4关 充珉瑶和贾涵山的行程情况

SELECT

    person.fullname,

    person.telephone,

    location.location\_name,

    itinerary.s\_time,

    itinerary.e\_time

FROM

    person

    LEFT JOIN itinerary ON itinerary.p\_id = person.id

    LEFT JOIN location ON itinerary.loc\_id = location.id

WHERE

    person.fullname = '充珉瑶'

    OR person.fullname = '贾涵山'

ORDER BY

    person.id DESC,

    itinerary.s\_time;

### 第5关 地名中带有‘店’字的地点名称

SELECT

    location.id,

    location.location\_name

FROM

    location

WHERE

    location.location\_name LIKE '%店%';

### 第6关 确诊者的接触者

SELECT

    person.fullname,

    person.telephone

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id

    JOIN location ON itinerary.loc\_id = location.id

WHERE

    location.location\_name = '活动中心'

    AND itinerary.e\_time > '2021-02-02 20:05:40'

    AND itinerary.s\_time < '2021-02-02 21:25:40'

ORDER BY

    person.fullname;

### 第7关 仍在使用的隔离点

SELECT DISTINCT

    isolation\_location.location\_name

FROM

    isolation\_location

WHERE

    isolation\_location.id IN ( SELECT isolation\_record.isol\_loc\_id FROM isolation\_record WHERE isolation\_record.state = 1 );

### 第8关 查询有出行记录的人员

SELECT

    person.fullname,

    person.telephone

FROM

    person

WHERE

    EXISTS ( SELECT DISTINCT itinerary.p\_id FROM itinerary WHERE person.id = itinerary.p\_id ORDER BY itinerary.p\_id )

    LIMIT 0,

    30;

### 第9关 没有去过“Today便利店“的人数

SELECT

    COUNT( person.id ) AS number

FROM

    person

WHERE

    NOT EXISTS (

    SELECT DISTINCT

        itinerary.p\_id

    FROM

        itinerary

        JOIN location ON itinerary.loc\_id = location.id

    WHERE

        itinerary.p\_id = person.id

        AND location.location\_name = 'Today便利店'

    );

### 第10关 去过所有地点的人员

SELECT

    person.fullname

FROM

    person

WHERE

    NOT EXISTS (

    SELECT

        \*

    FROM

        location

    WHERE

    NOT EXISTS ( SELECT \* FROM itinerary WHERE itinerary.p\_id = person.id AND itinerary.loc\_id = location.id ))

ORDER BY

    person.fullname;

### 第11关 隔离点的现状视图

CREATE VIEW isolation\_location\_status AS SELECT

isolation\_location.id,

isolation\_location.location\_name,

isolation\_location.capacity,

COUNT( isolation\_record.state = 1 OR NULL ) AS occupied

FROM

    isolation\_location

    INNER JOIN isolation\_record ON isolation\_location.id = isolation\_record.isol\_loc\_id

GROUP BY

    isolation\_location.id;

### 第12关 各隔离点的剩余房间数

SELECT

    isolation\_location\_status.location\_name,

    ( isolation\_location\_status.capacity - isolation\_location\_status.occupied ) AS available\_rooms

FROM

    isolation\_location\_status

ORDER BY

    isolation\_location\_status.id;

### 第13关 与无症状感染者靳宛儿有过接触的人

SELECT

    person.fullname,

    person.telephone

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id,

    (

    SELECT

        itinerary.loc\_id,

        itinerary.s\_time,

        itinerary.e\_time

    FROM

        person

        JOIN itinerary ON person.id = itinerary.p\_id

    WHERE

        person.fullname = '靳宛儿'

    ) AS temp\_table

WHERE

    person.fullname != '靳宛儿'

    AND itinerary.loc\_id = temp\_table.loc\_id

    AND itinerary.s\_time <= temp\_table.e\_time AND itinerary.e\_time >= temp\_table.s\_time

ORDER BY

    person.fullname;

### 第14关 每个地点发生的密切接触者人数

SELECT

    location.location\_name,

    COUNT( close\_contact.p\_id ) AS close\_contact\_number

FROM

    close\_contact

    JOIN location ON close\_contact.loc\_id = location.id

GROUP BY

    close\_contact.loc\_id

ORDER BY

    close\_contact\_number DESC,

    location.location\_name;

### 第15关 感染人数最多的人

SELECT

    close\_contact.case\_p\_id,

    person.fullname,

    COUNT( close\_contact.p\_id ) AS infected\_number

FROM

    close\_contact

    JOIN person ON close\_contact.case\_p\_id = person.id

GROUP BY

    close\_contact.case\_p\_id

ORDER BY

    infected\_number DESC

    LIMIT 0,

    1;

### 第16关 行程记录最频繁的3个人

SELECT

    person.fullname,

    COUNT( itinerary.id ) AS record\_number

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id

WHERE

    itinerary.e\_time >= '2021-02-02 10:00:00'

    AND itinerary.s\_time <= '2021-02-02 14:00:00'

GROUP BY

    person.id

ORDER BY

    record\_number DESC,

    person.fullname

    LIMIT 0,

    3;

### 第17关 房间数第2多的隔离点

SELECT

    isolation\_location.location\_name,

    isolation\_location.capacity

FROM

    isolation\_location

WHERE

    isolation\_location.capacity <( SELECT MAX( isolation\_location.capacity ) FROM isolation\_location )

LIMIT 0,1;

## 3.4 创建触发器

### 第1关 隔离点的人员“确诊新冠”后，自动转院

DROP TRIGGER IF EXISTS state\_change1;

DROP TRIGGER IF EXISTS state\_change2;

DELIMITER ||

CREATE TRIGGER state\_change1 AFTER UPDATE ON diagnose\_record FOR EACH ROW

BEGIN

    IF

        NEW.result = 1

        THEN

            UPDATE isolation\_record

            SET isolation\_record.state = 3

            WHERE

                isolation\_record.p\_id = NEW.p\_id;

    END IF;

END||

CREATE TRIGGER state\_change2 AFTER INSERT ON diagnose\_record FOR EACH ROW

BEGIN

    IF

        NEW.result = 1

        THEN

            UPDATE isolation\_record

            SET isolation\_record.state = 3

            WHERE

                isolation\_record.p\_id = NEW.p\_id;

    END IF;

END||

DELIMITER;

## 3.5 创建并使用自己创建的函数

### 第1关 创建函数并在语句中使用它

set global log\_bin\_trust\_function\_creators=1;

DELIMITER ||

DROP FUNCTION IF EXISTS Count\_Records;

CREATE FUNCTION Count\_Records(person\_id int)

    RETURNS int

BEGIN

    #Routine body goes here...

    RETURN(

        SELECT

            COUNT(itinerary.id)

        FROM

            itinerary

        WHERE

            itinerary.p\_id=person\_id);

END||

DELIMITER ;

SELECT

    \*

FROM

    person

WHERE

    Count\_Records ( person.id )>= 3

ORDER BY

    person.id;

# 4.应用系统设计

## 4.1系统设计目标

采用C/S模式实现一个医院管理系统。完成药品、诊疗、医师、病人、病房、科室等信息的管理。

设计目标：

1. 有面向患者的挂号服务。
2. 有面向医师的诊断服务。
3. 有面向管理人员的各类表项管理服务
4. 病人可以查看医师的相关信息、药品相关信息以及病房的部分信息。
5. 医师能够为病人办理开药、入院和出院手续。
6. 医院系统需要统计各类收费的财务报表。
7. 系统需包含事务（包含commit，rollback），存储过程/触发器，视图，函数。
8. 在程序中体现MySQL和JAVA语言的结合。

## 4.2 需求分析

### 4.2.1 功能需求

1. 患者不需要注册或登录账号即可拥有相关服务。在挂号服务中需要患者输入正确的证件号(18位)、手机号(11位)以及所挂号的医师等信息；在医师查询、药品查询、病房查询中可以通过搜索框检索对应的信息，并且部分信息需要对患者进行掩盖。

2. 医师需要通过自己的账号来登录以获取服务。在医师个人信息界面需显示医生的所有信息，且医生可以对自己的信息做需要的修改；在诊断界面需要显示当前正在处理的病人部分信息（姓名、年龄、性别），并且有开药、住院等办理手续；在出院办理界面，医师可以通过病人姓名或者证件号进行对应病房查询，并且进行出院办理。

3. 管理人员需要通过管理员账号来登录以获取服务。管理人员拥有查看和修改所有表项的权力，但财务表只允许查看；管理人员可以删除或添加相关数据。

4. 系统需要有信息输入的检错能力，禁止错误信息录入，保证输入到数据库的信息都是正确的。

### 4.2.2 性能需求

病人、医生、管理员所见到的数据应该是实时的，即任何数据发生修改后，各个人员所见到的数据都是修改后的。

### 4.2.3 数据完整性需求

1. [实体完整性](https://baike.baidu.com/item/%E5%AE%9E%E4%BD%93%E5%AE%8C%E6%95%B4%E6%80%A7" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7/_blank)

每个数据表都必须有主键，而作为主键的所有字段，其属性必须是独一及非空值，用id表示。

1. [域完整性](https://baike.baidu.com/item/%E5%9F%9F%E5%AE%8C%E6%95%B4%E6%80%A7" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7/_blank)

所有表中出现的性别栏只允许输入“男”或“女”；

所有的状态(state)项只有0或1；

费用类型为double，但不允许出现负数；

病房容量和入住人数需要大于等于0；

证件号需要为18位char类型；

电话号码需要为11位char类型。

1. [参照完整性](https://baike.baidu.com/item/%E5%8F%82%E7%85%A7%E5%AE%8C%E6%95%B4%E6%80%A7" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7/_blank)

所有表中引用的外键，要么为空，要么是在对应外表中存在。

### 4.2.4 数据流图

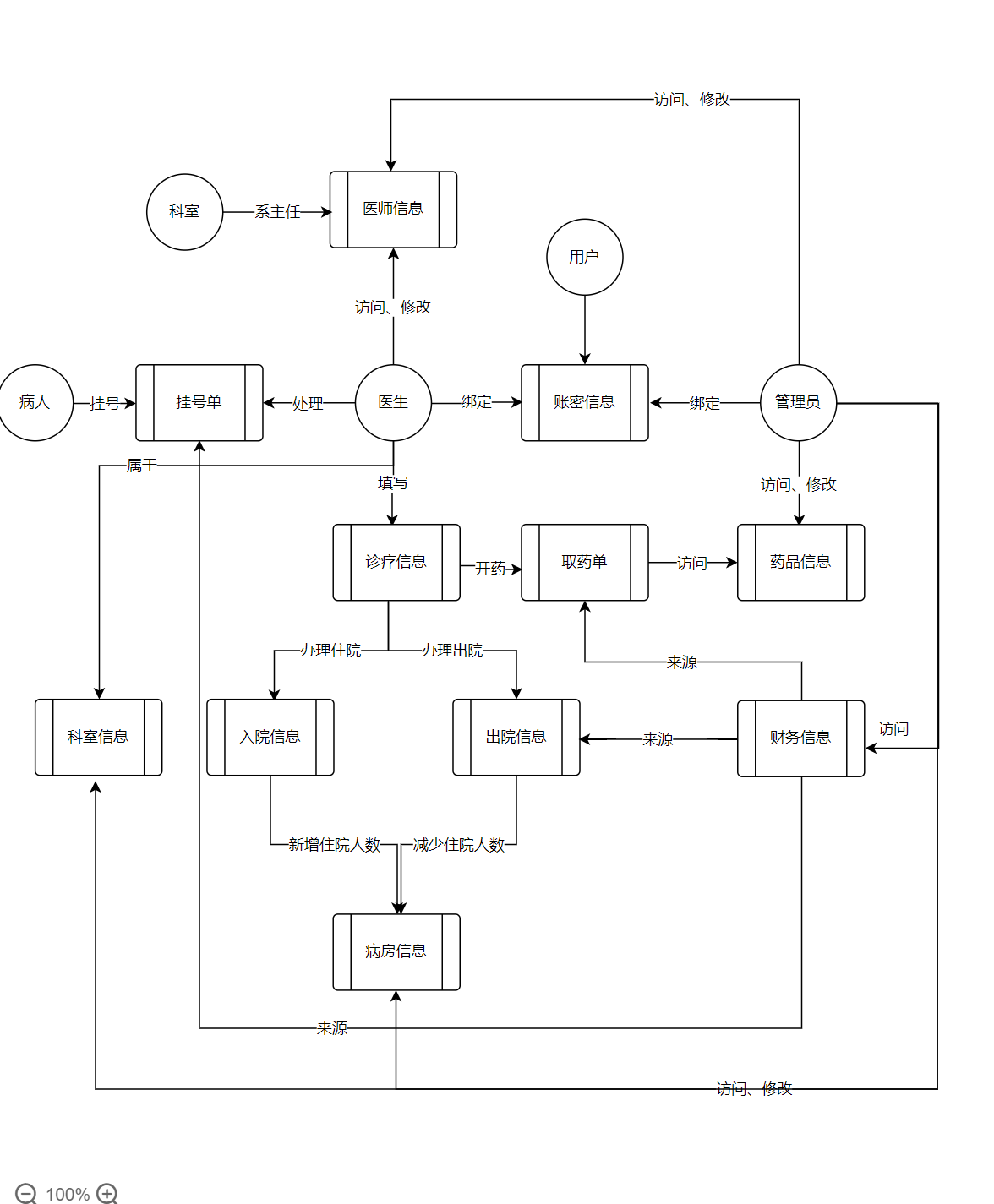


图 4-1 数据流图

### 4.2.5 数据字典

表 4-1 医院管理系统数据字典

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段名 | 数据类型 | 默认值 | 是否允许为空 |
| 入院信息 | id | int | NULL | 否 |
| 入院信息 | 入住时间 | datetime | NULL | 否 |
| 入院信息 | 病人 | int | NULL | 是 |
| 入院信息 | 办理人员 | int | NULL | 否 |
| 入院信息 | 病房 | int | NULL | 否 |
| 入院信息 | state | int | NULL | 是 |
| 出院信息 | id | int | NULL | 否 |
| 出院信息 | 病人 | int | NULL | 否 |
| 出院信息 | 办理人员 | int | NULL | 否 |
| 出院信息 | 出院时间 | datetime | NULL | 否 |
| 出院信息 | 费用 | int | NULL | 是 |
| 医生信息 | id | int | NULL | 否 |
| 医生信息 | 姓名 | char(30) | NULL | 否 |
| 医生信息 | 性别 | char(10) | NULL | 是 |
| 医生信息 | 出生日期 | date | NULL | 是 |
| 医生信息 | 入职日期 | date | NULL | 是 |
| 医生信息 | 所属科室 | int | NULL | 是 |
| 医生信息 | 职务 | char(30) | NULL | 是 |
| 医生信息 | 是否为专家 | int | NULL | 是 |
| 医生信息 | 电话号码 | char(30) | NULL | 是 |
| 医生信息 | 电子邮箱 | char(30) | NULL | 是 |
| 医生信息 | 挂号费 | int | NULL | 是 |
| 取药单 | id | int | NULL | 否 |
| 取药单 | 取药时间 | datetime | NULL | 是 |
| 取药单 | 费用 | double | NULL | 是 |
| 挂号信息 | id | int | NULL | 否 |
| 挂号信息 | 就诊病人 | int | NULL | 否 |
| 挂号信息 | 医生 | int | NULL | 否 |
| 挂号信息 | 挂号时间 | datetime | NULL | 否 |
| 挂号信息 | 挂号费 | int | NULL | 否 |
| 挂号信息 | 是否为专家号 | tinyint(1) | NULL | 是 |
| 挂号信息 | 状态 | int | NULL | 是 |
| 用户 | id | int | NULL | 否 |
| 用户 | 用户名 | char(30) | NULL | 否 |
| 用户 | 密码 | char(30) | NULL | 否 |
| 用户 | 权限等级 | int | NULL | 是 |
| 用户 | 医师 | int | NULL | 是 |
| 病人信息 | id | int | NULL | 否 |
| 病人信息 | 证件号 | char(20) | NULL | 是 |
| 病人信息 | 姓名 | char(30) | NULL | 否 |
| 病人信息 | 性别 | char(20) | NULL | 是 |
| 病人信息 | 出生日期 | date | NULL | 是 |
| 病人信息 | 联系方式 | char(20) | NULL | 是 |
| 病房信息 | id | int | NULL | 否 |
| 病房信息 | 病房号 | char(30) | NULL | 否 |
| 病房信息 | 病房容量 | int | NULL | 是 |
| 病房信息 | 房间类型 | char(20) | NULL | 是 |
| 病房信息 | 入住人数 | int | NULL | 是 |
| 病房信息 | 备注 | char(30) | NULL | 是 |
| 科室信息 | id | int | NULL | 否 |
| 科室信息 | 科室名称 | char(30) | NULL | 否 |
| 科室信息 | 系主任 | int | NULL | 是 |
| 药品信息 | id | int | NULL | 否 |
| 药品信息 | 名称 | char(30) | NULL | 否 |
| 药品信息 | 剂型 | char(30) | NULL | 是 |
| 药品信息 | 规格 | char(30) | NULL | 是 |
| 药品信息 | 使用说明 | char(255) | NULL | 是 |
| 药品信息 | 参考价格 | double | NULL | 是 |
| 药品信息 | 类型 | char(30) | NULL | 是 |
| 药物清单 | id | int | NULL | 否 |
| 药物清单 | 单号 | int | NULL | 否 |
| 药物清单 | 药品 | int | NULL | 否 |
| 药物清单 | 费用 | double | NULL | 否 |
| 诊断信息 | id | int | NULL | 否 |
| 诊断信息 | 病人 | int | NULL | 否 |
| 诊断信息 | 医生 | int | NULL | 否 |
| 诊断信息 | 诊断使时间 | datetime | NULL | 是 |
| 诊断信息 | 开药单号 | int | NULL | 是 |
| 诊断信息 | 入院单号 | int | NULL | 是 |

## 4.3 总体设计

### 4.3.1 控制器（controller）

**1. 医师登录界面控制器（[DoctorLoginController](http://localhost:63342/%E5%8C%BB%E9%99%A2%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/HospitalManagementSystem/JavaDoc/HospitalManagementSystem/controller/DoctorLoginController.html" \o "HospitalManagementSystem.controller中的类)）**

void onClickLogin() ：医师登录

void onClickRegister() ：注册新账户

void onClickReturn() ：返回上一级

void setLoginApp (Main loginApp) ：设置登陆界面APP

**2. 医师服务界面控制器（DoctorServiceController）**

void onClickAddMedicine() ：添加药品至取药单

void onClickDeleteMedicine() ：从取药单中移除所选药品

void onClickDiagnosis() ：提交诊断信息

void onClickDischargedSearch() ：病人所在病房信息查询

void onClickDischargedSubmit() ：提交出院

void onClickReturn() ：返回上一级

void onClickSubmit() ：信息修改提交

void setDoctorServiceApp (Main doctorServiceApp, int doctorID) ：设置医师服务界面

**3. 主界面控制器（MainController）**

void onClickLogin() ：点击医师登陆后进行的操作

void onClickService() ：点击看病服务后进行的操作

void setMyApp (Main myApp) ：设置注解面APP

**4. 管理人员界面控制器（ManagerController）**

void setManagerApp (Main managerApp) ：设置管理人员界面

void onClickDoctorSearch() ：医师搜索

void onClickMedicineSearch() ：药品信息查询

void onClickWarnSearch() ：病房信息查询

void onClickDepartmentSearch() ：科室信息查询

void onClickFinanceSearch() ：财务查询

void setDoctorData (ObservableList<Doctor> doctorData) ：医师数据导入

void setDepartmentData (ObservableList<Department> departmentData) ：科室数据导入

void onClickReturn() ：返回上一级

void setFinanceData (ObservableList<Finance> financeData) ：科室数据导入

void onEditDepartment(TableColumn.CellEditEvent<Department,String> departmentStringCellEditEvent)：修改药品信息表

void onEditWarn(TableColumn.CellEditEvent<Ward,String> wardStringCellEditEvent) ：修改病房信息表

void onEditDepartment(TableColumn.CellEditEvent<Department,String> departmentStringCellEditEvent) ：修改科室信息表

void onEditDoctor(TableColumn.CellEditEvent<Doctor,String> doctorStringCellEditEvent) ：修改医师信息表

void onClickMedicineAdd() ：新增药品数据

void onClickMedicineDelete() ：删除药品数据

void onClickWardAdd() ：新增病房数据

void onClickWardDelete() ：删除病房数据

void onClickDoctorAdd() ：新增医师数据

void onClickDoctorDelete() ：删除医师数据

void onClickDepartmentAdd() ：新增科室数据

void onClickDepartmentDelete() ：删除科室数据

**5. 挂号信息界面控制器（RegisterInfoController）**

void setRegisterInfoApp (Main registerInfoApp,int registerNum) ：设置挂号信息界面

void onClickReturn() ：返回上一级

**6. 病人服务界面控制器（ServiceController）**

void setServiceApp​(Main serviceApp) ：设置服务界面APP

void onPressedSex() ：选择性别栏

void onPressedDepartment() ：选择科室栏

void onPressedDoctor() ：选择医师栏

void onClickReturn() ：返回上一级

void onClickSubmit() ：挂号

void onClickDoctorSearch() ：医生信息查询

void onClickMedicineSearch() ：药品信息查询

void onClickWarnSearch() ：病房信息查询

### 4.3.2 函数（function）

**Func（医院系统函数）**

static void ConnectToDataBase() ：连接数据库

static void CloseConnection() ：关闭连接

static int getYear (java.util.Date date) ：计算所给日期之间至今过了多少年

static int Check (java.lang.String account,java.lang.String password) ：检测账户密码的正确性

### 4.2.3 模块（model）

**1. 科室（Department）**

id - :科室id

departmentName - :名称

deanName - :系主任

**2. 医师（Doctor）**

id - :人员id

name - :姓名

sex - :性别

birthday - :出生日期

workingDay - :工作日期

department - :科室

job - :职务

isExpert - :是否为专家

phoneNumber - :电话号码

email - :电子邮箱

registerFee - :挂号费

**3. 财务（Finance）**

id - :编号

incomeSource - :来源

incomeTime - :时间

fee - :费用

**4. 主类（Main）**

void gotoMain() ：跳转至主界面

void gotoDoctorLogin() ：跳转至医师登陆界面

void gotoRegister (int registerNum) ：跳转至挂号信息界面

static void main (java.lang.String[] args) ：主函数

**5. 药品（Medicine）**

id - :药品ID

name - :名称

dosageForm - :剂型

specifications - :规格

introduction - :使用说明

price - :参考价格

type - :类型

**6. 病人（Patient）**

id - :病人id

name - :姓名

sex - :性别

birthday - :出生日期

phone - :电话号码

**7. 病房（Ward）**

id - :房间ID

number - :房间号

capacity - :容量

type - :类型

used - :已住人数

remarks - :备注

## 4.4 数据库设计

### 4.4.1 E-R图模型

通过Navicat逆向工程导出的医院管理系统E-R图模型如图 4-1 医院管理系统E-R图所示：

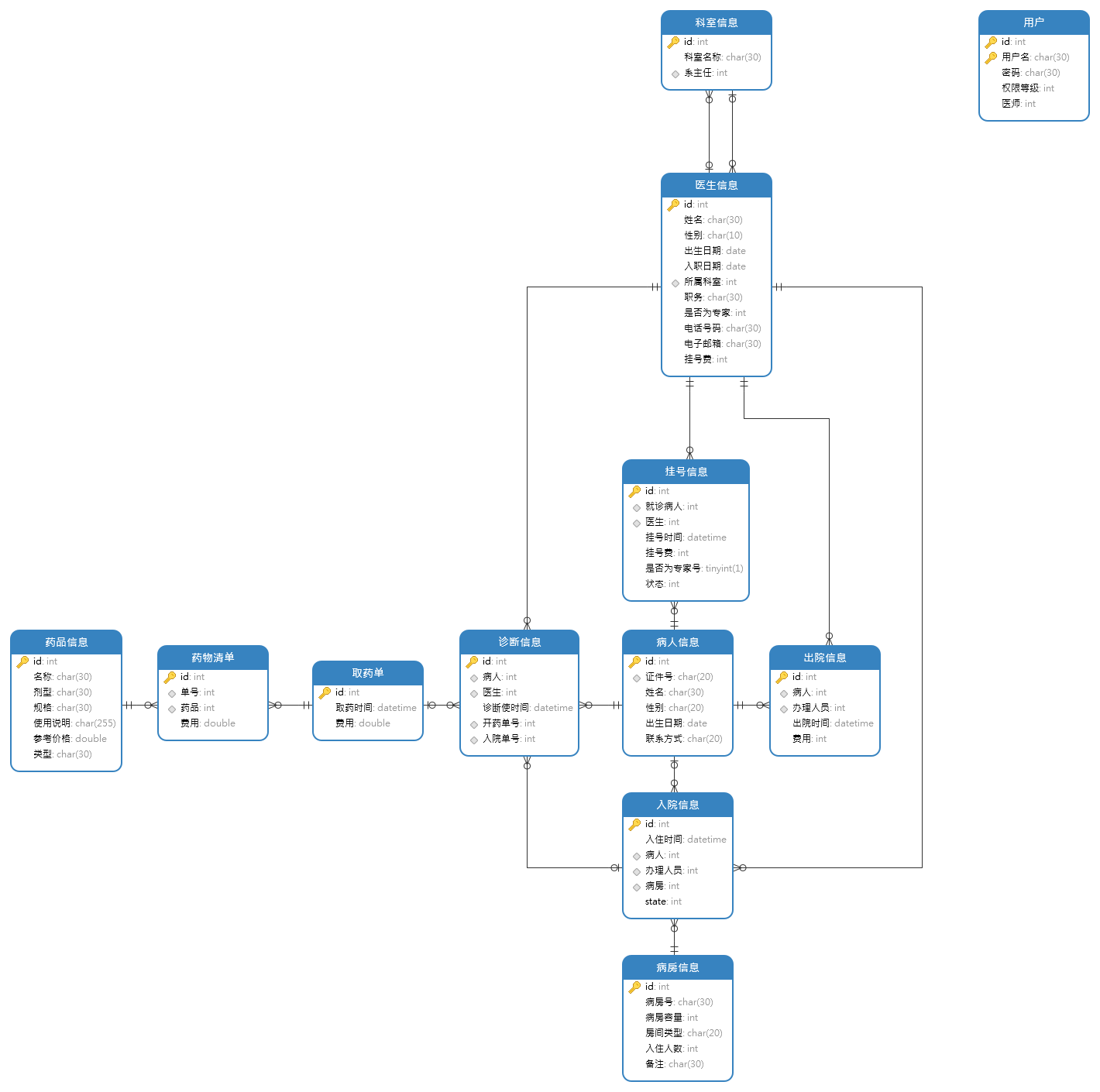


图 4-2 医院管理系统E-R图

### 4.4.2 数据库逻辑结构设计

1. 入院信息表逻辑结构如图 4-2所示：



图 4-3 入院信息表

2. 出院信息表逻辑结构如图 4-3所示：

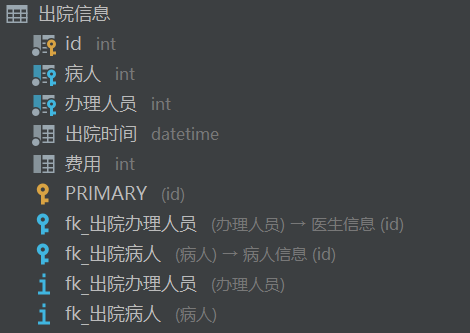


图 4-4 出院信息表

3. 医生信息表逻辑结构如图 4-4所示：



图 4-5 医生信息表

4. 取药单表逻辑结构如图 4-5所示：

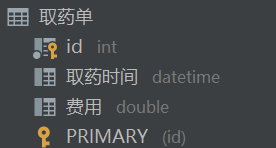


图 4-6 取药单

5. 挂号信息表逻辑结构如图 4-6所示：



图 4-7 挂哈信息

6. 用户信息表逻辑结构如图 4-7所示：



图 4-8 用户

7. 病人信息表逻辑结构如图 4-8所示：



图 4-9 病人信息

8. 病房信息表逻辑结构如图 4-9所示：



图 4-10 病房信息

9. 科室信息表逻辑结构如图 4-10所示：

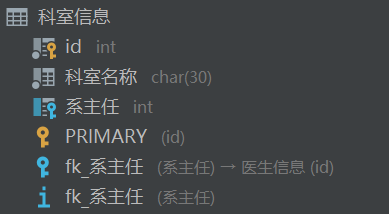


图 4-11 科室信息

10. 药品信息表逻辑结构如图 4-11所示：



图 4-12 药品信息

11. 药物清单表逻辑结构如图 4-12所示：

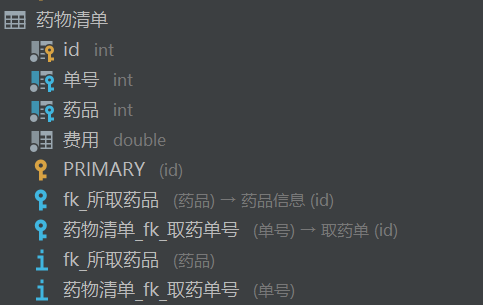


图 4-13 药物清单

12. 诊断信息表逻辑结构如图 4-13所示：



图 4-14 诊断信息

## 4.5 详细设计与实现

### 4.5.1 控制器（controller）

**1. 医师登录界面控制器（[DoctorLoginController](http://localhost:63342/%E5%8C%BB%E9%99%A2%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/HospitalManagementSystem/JavaDoc/HospitalManagementSystem/controller/DoctorLoginController.html" \o "HospitalManagementSystem.controller中的类)）**

（1）登录：通过绑定点击登录按钮的触发事件，监听是否请求登录。

首先利用Func函数中的Check函数，检查输入的用户名和密码的正确性，如果输入正确，Check函数会返回用户信息的id。然后用

"SELECT 医师 FROM 用户 WHERE id="+id;

进行对应的医师查找，并得到医师的doctorId，如果是管理员则doctorId=0。

根据相应的doctorId跳转到医师服务界面或者是管理员界面。

如果输入错误，则弹出错误信息。

（2）注册：通过绑定点击注册按钮的触发事件，监听是否请求注册。

根据用户名和密码输入框中的信息添加新的用户。首先判断用户名是否在用户表中已经注册，即

"SELECT 用户名 FROM 用户";

然后依次比较输入的用户名和表中是否有重复，重复则显示错误信息，若表中不存在该用户，则进行用户添加。添加用户时，需要确定添加的id，利用

"SELECT COUNT(\*) AS NUM FROM 用户";

得到当前有多少用户记录，然后添加的用户id为用户记录的数量+1。再获取从文本框输入的账户和密码信息，通过INSERT语句新插入一条用户，初始绑定为一个空医师，权限等级为0。

（3）返回：返回时进行界面切换，直接切换到主界面即可。

**2. 医师服务界面控制器（DoctorServiceController）**

（1）初始化：在初始化界面导入相应需要的数据。如插入性别选项、导入科室数据、导入个人信息、导入病人信息等。

性别选项只设置“男”和“女”，即

cbxSex.getItems().addAll("男","女");

个人信息界面设有科室修改选项，只允许修改为选项栏中的科室，所以选项栏要导入科室名称。因此先用

"SELECT 科室信息.id, 科室名称, 姓名 FROM 科室信息 JOIN 医生信息 医 on 医.id = 科室信息.系主任";

将科室的id、科室名称和科室系主任信息导入到Department类中，然后再获取科室的名称依次加入到选项栏中。

在个人信息中导入数据时，直接将对应医师id的信息查询

"SELECT 医.id, 姓名, 性别, 出生日期, 入职日期, 科室名称, 职务, 是否为专家, 电话号码, 电子邮箱,挂号费 FROM 医生信息 医 JOIN 科室信息 科 on 医.所属科室 = 科.id";

然后将所得到的数据依次提取到相应文本框中即可。

在导入病人数据时，要先根据挂号表中，此挂在此医师下的病人以及未处理的挂号单的信息进行查询，即

"SELECT id,就诊病人 FROM 挂号信息 WHERE 医生="+doctorID+" AND 状态=0 ORDER BY id";

由此可以得到待处理的病人数目，然后显示排在最前面的病人信息。

导入药品选项和导入病房选项同理，查询对应数据表中的信息后按照需要的数据进行导入。

1. 返回：返回时进行界面切换，直接切换到主界面即可。
2. 信息修改提交：个人信息修改后，提交已修改的信息。

先根据科室的名称从科室信息表中找到科室的id，然后再由修改的各项信息对医生信息表中的对应医生进行修改，即：

"UPDATE 医生信息 " +  
 "SET 姓名='"+fieldName.getText()+  
 "',性别='"+cbxSex.getValue()+  
 "',出生日期='"+

dateBirthday.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"))+  
 "',入职日期='"+

dateWorkingDay.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"))+  
 "',所属科室="+rs.getInt(1)+  
 ",职务='"+fieldJob.getText()+  
 "',是否为专家="+expert+  
 ",电话号码='"+fieldPhone.getText()+  
 "',电子邮箱='"+fieldEmail.getText()+  
 "',挂号费="+fieldRegisterFee.getText()+  
 " WHERE id="+doctorID;

1. 添加药品至取药单：在开药选项栏中选择了药物后，点击添加按钮时，自动将此药品加入到药品清单列表。

首先判断选项栏内容是否为空，不为空的话，添加药物名称至列表，即

if(!cbxMedicine.getValue().equals("")){  
 listMedicine.getItems().add(cbxMedicine.getValue());  
}

1. 从取药单中移除所选药品：在列表中选中对应药品所在的行，点击删除按钮时，在列表中删除该行。

获取在列表中所选的行信息，然后删除该行，即

String remove=listMedicine.getSelectionModel().getSelectedItem();  
listMedicine.getItems().remove(remove);

1. 提交诊断信息：点击提交按钮时，提交对病人的诊断信息，包括开药单和入院单。

首先需要关闭连接的自动提交，以此来进行可回滚的操作，即

Func.connection.setAutoCommit(false);

然后判断是否有开药，如果有开药则需要添加取药单。添加取药单时同样需要获取已有数据的数量，由此确定取药单的单号medicineListId，然后新插入一个取药单数据，此时暂时把费用置为空，即

"INSERT INTO 取药单 VALUES("+medicineListId+  
 ",'"+df.format(new Date())+"',null)";

每当开药的列表中有一个药品，则需要新建一条药物清单的数据，此药物清单的取药单单号均绑定为medicineListId。根据列表中的药物名称，找到对应的药品信息，插入到药物清单中，即

sql="SELECT id,参考价格 FROM 药品信息 WHERE 名称='"+medicineName+"'";  
rs=Func.statement.executeQuery(sql);  
rs.next();  
totalPrice+=rs.getDouble(2);  
sql="INSERT INTO 药物清单 VALUES("+subMedicineListId+  
 ","+medicineListId+","+rs.getInt(1)+  
 ","+rs.getDouble(2)+")";

将每个药物清单数据中的单价加起来，即可得到取药单中的总价格。

当有选择病房时，也就是需要进行入院办理时，添加入院信息时要先根据病房号查找到病房的编号wardId，然后插入住院信息，如下：

"INSERT INTO 入院信息 VALUES("+admitId+  
 ",'"+df.format(new Date())+"',"+patientId+","+doctorID+","+wardId+",0)";

入院信息中的state表示目前的状态，0表示仍在住院，1表示已经出院。

并且住院后需要将病房信息更新，其入住人数需要加1，即

"UPDATE 病房信息 SET 入住人数=入住人数+1 WHERE id="+wardId;

最后如果其中有某一条语句出错或者抛出异常，则需要进行回滚操作，以保证操作的原子性；正确执行每一条语句后提交修改。

Func.connection.rollback();

Func.connectione.commit();

每当处理完一位病人后，自动切换到下一位病人，即需要更新显示的病人信息；若已经是最后一位病人，则无需切换。

1. 出院办理：首先需要依据输入病人的姓名或者证件号进行病人住院信息查询，然后填写住院费用后，即可出院。

对于住院病人的查询，需要查询来自入院信息表中的数据，如果此病人在入院信息中，且state=0，即仍在住院，则显示其相关信息，然后由医师输入住院费后，进行出院办理。

出院时需要更新入院信息表中此病人的state，即设置为1，表示已经出院；同时还需要将病房中入住人数减1；最后需要在出院信息表中插入新的数据，表示该病人由该医师办理出院。

**3. 主界面控制器（MainController）**

（1）医师登录：跳转至医师登录界面。

（2）病人服务：跳转至病人服务界面。

**4. 管理人员界面控制器（ManagerController）**

（1）初始化：管理人员界面需要导入所有表的信息。

初始化时，查询所有表的信息，并建立对应的类型实体，用于存储相关信息，然后在TableView中显示出来，例如：

sql = "SELECT 科室信息.id, 科室名称, 姓名 FROM 科室信息 LEFT JOIN 医生信息 医 on 医.id = 科室信息.系主任 "+  
 "ORDER BY 科室信息.id";  
ResultSet rs = Func.statement.executeQuery(sql);  
while (rs.next()) {  
 Department department = new Department(  
 rs.getInt(1), rs.getString(2), rs.getString(3)  
 );  
 departmentData.add(department);  
}  
setDepartmentData(departmentData);

按照类似的操作依次导入科室信息、医师信息、药品信息、病房信息、财务信息。

其中财务信息需要通过查询挂号单中的挂号费、出院信息中的住院费以及取药单中的取药费来得到。

（2）搜索功能：通过界面中的搜索框，输入对应的搜索信息，点击搜索按钮后显示需要的信息。

搜索框限制了搜索的内容，如医师搜索只限于搜索医师的名字、病房搜索只限于病房号、药品搜索只限于药品名称等。首先通过搜索栏中的内容，如果内容为空，则显示表中的全部数据；不为空，则在表中显示对应的信息，例如查询医师姓名时，选择对应的医师信息进行展示，不存在则展示空，即

ObservableList<Doctor> doctorMach = FXCollections.observableArrayList();  
//查找到对应的医生  
for (Doctor doctor : doctorData) {  
 if (doctor.getName().equals(fieldDoctorSearch.getText())) {  
 doctorMach.add(doctor);  
 }  
}  
setDoctorData(doctorMach);

1. 修改功能：选择表项中的某个单元格时，可进行编辑操作，编辑完成后会自动提交修改到数据库，完成对数据的修改。

不允许对财务表的修改。

对其他表进行修改时，首先需要确定所编辑的表格的位置，然后找到数据库中对应的数据，检查修改的合法性，修改合法则提交到数据库，否则驳回，如修改药品信息，首先修改TableView表中的内容

//按照修改的行和列，修改对应数据  
switch (column){  
 case 1:  
 medicineData.get(row).setName(newValue);  
 break;  
 case 2:  
 medicineData.get(row).setDosageForm(newValue);  
 break;  
 case 3:  
 medicineData.get(row).setSpecifications(newValue);  
 case 4:  
 medicineData.get(row).setIntroduction(newValue);  
 case 5:  
 try {  
 int price=Integer.parseInt(newValue);  
 medicineData.get(row).setPrice(price);  
 } catch (NumberFormatException e) {  
 e.printStackTrace();  
 new Alert(Alert.AlertType.INFORMATION, "格式有误，修改失败").showAndWait();  
 return;  
 }  
 break;  
 case 6:  
 medicineData.get(row).setType(newValue);  
 break;  
 default:  
 break;  
}

确定修改的合法性后，将修改提交到数据库

String sql="UPDATE 药品信息 " +  
 "SET 名称='"+medicine.getName()+  
 "',剂型='"+medicine.getDosageForm()+  
 "',规格='"+medicine.getSpecifications()+  
 "',使用说明='"+medicine.getIntroduction()+  
 "',参考价格="+medicine.getPrice()+  
 ",类型='"+medicine.getType()+  
 "' WHERE id="+medicine.getId();  
int len=Func.statement.executeUpdate(sql);  
if(len>0){  
 new Alert(Alert.AlertType.INFORMATION, "修改成功").showAndWait();  
}  
else {  
 new Alert(Alert.AlertType.INFORMATION, "修改失败").showAndWait();  
}

（4）删除功能：在表格中选择某一行数据后，点击删除按钮对数据进行删除，并提交至数据库中。

和修改类似，首先需要确定所选中的行，然后先删除TableView中的内容，在确定删除合法后，提交到数据库。

其中会存在科室删除的合法性判断，如果科室中仍有医师数据存在时，不允许删除，也就是说，只要还有一位医师是属于该科室的，则不允许删除该科室，只有当该科室中全部的医师都被删除后，该科室才会被删除。

（5）添加功能：添加某一个表中的数据时，首先需要填写添加的信息，然后确定数据合法后，在数据库对应的表中添加该数据项。

例如，添加药品信息

int id=medicineData.get(medicineData.size()-1).getId()+1;  
String sql="INSERT INTO 药品信息 VALUES("+id+  
 ",'"+fieldAddMedicineName.getText()+  
 "','"+fieldAddMedicineDosage.getText()+  
 "','"+fieldAddMedicineSpecifications.getText()+  
 "','"+fieldAddMedicineIntroduction.getText()+  
 "',"+fieldAddMedicinePrice.getText()+  
 ",'"+fieldAddMedicineType.getText()+"')";  
Medicine medicine=new Medicine(id,

fieldAddMedicineName.getText(),

fieldAddMedicineDosage.getText(),  
 fieldAddMedicineSpecifications.getText(),

fieldAddMedicineIntroduction.getText(),  
 Integer.parseInt(fieldAddMedicinePrice.getText()),

fieldAddMedicineType.getText());  
int len=Func.statement.executeUpdate(sql);

**5. 挂号信息界面控制器（RegisterInfoController）**

（1）初始化：挂号信息界面要显示挂号的病人信息、医师信息以及挂号的时间、费用等。

首先根据挂号的病人id查询到病人的信息，在对应的显示框中进行显示；然后再根据医师id查询到医师的信息，同样显示在对应的显示框中。

（2）返回上一级：切换到主界面。

**6. 病人服务界面控制器（ServiceController）**

（1）初始化：初始化界面中，有挂号提交，因此病人需要选择所要挂号的医师，故要导入医师的姓名信息。同时，病人可以查看部分药品、病房、医师等信息，故也需要有这些表信息的导入。

导入表信息的部分同以上的各个操作，先查询表中所需要的信息，然后建立对应的类型实体，再在TableView中显示出来即可。

在挂号界面，病人可以首先选择需要挂号的科室，然后选择科室中的医师，故需要对医师进行筛选功能，筛选所选择科室的医师，如果没有选择科室，则需要显示全部的医师，如下

//如果已经选择了科室，则只添加对应科室的医师  
if(department!=null&&!department.equals("全部")){  
 //当列表中已有数据，则说明不需要进行添加  
 if(cbxDoctor.getItems().size()>0){  
 return;  
 }  
  
 for(Doctor doctor:doctorData){  
 if(doctor.getDepartment().equals(department)){  
 cbxDoctor.getItems().add(doctor.getName());  
 }  
 }  
}  
//没有选择科室，则加入全部医师  
else {  
 //只当没有显示全部医师时，进行加入  
 if(cbxDoctor.getItems().size()<doctorData.size()){  
 for(Doctor doctor:doctorData){  
 cbxDoctor.getItems().add(doctor.getName());  
 }  
 }  
}

1. 挂号提交：病人输入个人信息以及选择挂号的医师后，点击挂号按钮则可进行挂号。

提交挂号时，需要新建立一个挂号单的数据，首先要获得病人的信息，如果病人信息表中不存在该病人，则需要在病人信息表中加入该病人，然后获取病人id，在挂号单中进行添加。

在此前提为每个病人的证件号码是唯一的，病人输入个人信息时会被要求输入证件号码，提交时，先在病人信息表中查询该证件号码是否已经有病人信息存在，若存在则可直接进行挂号单的生成，否则还需要先在病人信息表中创建新的病人信息，在进行挂号单的创建。

挂号完成后，跳转至挂号信息显示界面。

### 4.5.2 函数（Function）

（1）数据库连接函数：连接到对应的数据库。

通过JDBC驱动，进行数据库的绑定，建立数据库连接后数据相关信息，如下：

// 注册 JDBC 驱动  
Class.forName(JDBC\_DRIVER);  
// 打开链接  
System.out.println("连接数据库...");  
connection = DriverManager.getConnection(DB\_URL,USER,PASS);  
statement = connection.createStatement();

（2）关闭数据库连接：在程序结束时需要关闭数据库连接。

先判断是否有连接资源，若发现有数据库连接，则进行关闭，如下：

if (statement != null) {  
 statement.close();  
}  
if (connection != null) {  
 connection.close();  
}

（3）计算年限：计算所给日期至今过了多少年。

先获取当前日期，和所给参数日期进行比较。先将年份做差得到预估年限，再比较月份，如果所给日期的月在当前月之后，则预估年限减1得到正确年限；如果月相同，则比较天，如果所给日期的天在当前天之后，则预估年限减1得到正确年限。即：

int differenceYear = yearNow - yearBirth; //计算整岁数  
if (monthNow <= monthBirth) {  
 if (monthNow == monthBirth) {  
 if (dayOfMonthNow < dayOfMonthBirth) {  
 differenceYear--;//当前日期在生日之前，年龄减一  
 }  
 }  
 else{  
 differenceYear--;//当前月份在生日之前，年龄减一  
 }  
}

（3）检验登录：检测账户密码的正确性。

通过对数据库中用户表的数据查询，得到所有的账户以及对应的密码，再和所输入的账密做比较，如果正确则返回此账户密码所对应的用户id编号，否则返回0。

### 4.5.3 模块（model）

模块中部分信息已在总体设计中说明，以下只详细说明以下主类。

1. start函数：JavaFX界面启动，即显示主界面。
2. 各类界面跳转函数：跳转至对应界面。
3. 界面替换函数：实现界面替换的主要部分。

对传入的参数，即所需要跳转的界面进行设置，获取界面资源后，将其设置为当前显示页，如下

FXMLLoader loader = new FXMLLoader();  
InputStream in = Main.class.getResourceAsStream(fxml);  
loader.setBuilderFactory(new JavaFXBuilderFactory());  
loader.setLocation(Main.class.getResource(fxml));  
  
Pane pane;  
pane = loader.load(in);  
assert in != null;  
in.close();  
Scene scene = new Scene(pane,MINIMUM\_WINDOW\_WIDTH,

MINIMUM\_WINDOW\_HEIGHT);  
stage.setScene(scene);  
return loader.getController();

（4）主函数：程序运行入口，包含连接数据库、启动界面、关闭数据库。

### 4.5.4 界面资源（resources）

通过JavaFX界面设计，和Scene Builder可视化界面设计，对各个界面进行设计和按钮、表格、文本框等器件进行函数绑定。

所设计的界面fxml资源如下

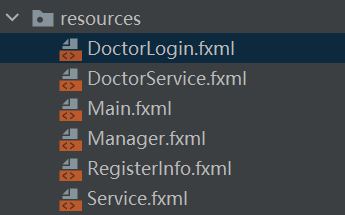


图 4-15 界面资源

## 4.6 系统测试

### 4.6.1 主菜单展示

主菜单界面如图 4-15所示：



图 4-16 主菜单

### 4.6.2 病人服务

（1）病人服务界面如图 4-16 病人服务所示：





图 4-17 病人服务

病人可以进行医师、药品、病房的查询；当然最终要的，可以进行挂号操作，输入姓名、证件号、联系方式后，选择相应的医师即可进行挂号。点击提交按钮后会自动跳转到挂号信息界面，如图 4-17所示。

（2）挂号信息界面如图 4-17所示：



图 4-18 挂号信息

### 4.6.3 医师服务

（1）医师登录界面如图 4-18所示：



图 4-19 登录界面

输入正确的账户密码后即可登录到对应医师界面，测试中使用[001][123]账密登录，信息为周大伟医师。

（2）医师服务如所示：



图 4-20 医师服务界面

个人信息展示页可以修改个人信息，点击保存修改按钮后即可进行修改。

诊断页会显示正在处理的病人，医师可以在药品选项栏中选择需要的药品进行添加或者在药品表中选择对应的行进行删除，也可以选择病房进行入院办理，点击提交按钮即表示该病人以及处理完成，自动跳转到下一位病人。

出院办理界面可以通过病人姓名或者证件号查询正在住院的病人，点击确认出院后即可进行出院办理。

### 4.6.4 管理人员服务

（1）管理人员登录界面如图 4-18所示。

管理人员测试时登录[100][123]账密进程测试。

（2）管理人员服务界面如图 4-20所示：



图 4-21 管理人员服务

以上只显示了药品管理界面，其他的界面大同小异，不做多余的展示。

在界面的表格中可以双击单元格进行信息修改，修改完成后会自动提交到数据库中，也可以选择某一行进行删除。同样所有的管理表项中都加入了搜索功能，检索对应的信息即可得到相应的数据。

在添加数据的界面中，需要正确输入所添加数据的所有信息，然后点击提交按钮则会在数据库中添加对应的数据。

（3）财务管理如所示：



图 4-22 财务管理

财务管理同其他表略有区别，众所周知财务是不能进行删除修改以及随意添加的，所以财务表只能阅读，不能修改。

## 4.7 系统设计与实现总结

整个医院管理系统耗时约一周，几乎是从零开始一路完成。虽然前期遇到了很多奇怪的问题，比如需要配环境、安装各种插件等，但根据来自各个网站的教程、指导等相关文献，慢慢的能自己进行开发。

对于数据库部分，由于医院管理这一选题的独特性，也就是相对来说比较复杂，一次性建立起一个完美的数据库是几乎不太现实的，所以最开始建立的一些表后面会经历各种修改，比如主键、外键、数据类型等等。其次就是关于药品的多项选择，需要通过父子表项来进行实现，也是费了一点时间才想到。

而JAVA代码方面，之前没有过对界面设计的学习和相关的实践操作，几乎都是当场现学现卖，从网站上找一些例子进行适当的参考，然后再做出自己的界面。考虑到每个界面需要完成对系统的一些相关操作，故界面设计方面也是改了又改，最后才到最终提交的版本。

总体上来说，虽没有实现很完善的医院功能，但也实现了基础了数据增删改查、用户和数据库的交互、医师病人管理人员角色的功能，还是比较有收获的。

# 5.课程总结

1. 数据库的基本操作。即增、删、改、查四大类操作，均在Educoder上进行了训练，尤其是查询操作，无论是简单还是复杂的查询，现在都能熟练进行。

2. 数据库的设计。在医院管理系统设计中，进行了数据库设计的实际操作，完成了自己的医院系统，在看到了数据库实际应用方面的同时，也感受到了设计一个好的数据库系统的复杂性，和真是案例中数据库系统的复杂性。

3. 客户端开发。虽然并没有真正意义上完成一个好的客户端，但在实践中也掌握了一些JAVA前端开发的技巧和能力，同样也了解了一些数据库和前端相互交互的过程，对与客户端-服务器模式也有了更加深刻的理解。

4. 部分建议。相较于其他的实验选题，医院管理系统难度系数感觉上要高不少，要考虑的因素也是很多的，不仅仅是数据表的增删改查这么简单，而是要考虑到不同角色所需要的不同服务，因此希望老师简化一下医院管理系统的选题，或者将其他选题上升到同等复杂度。