

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 工资管理系统 |
| 姓 名： | 刘峻屹 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1803 |
| 学 号： | U201814812 |
| 指导教师： | 瞿彬彬 |

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

2021年6月30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 1](#_Toc3131)

[2 软件功能学习 1](#_Toc26309)

[2.1 任务要求 1](#_Toc6852)

[2.2 完成过程 1](#_Toc5216)

[3 SQL练习 7](#_Toc1397)

[3.1 建表 7](#_Toc31852)

[3.2 数据更新 11](#_Toc19731)

[3.3 查询 13](#_Toc15855)

[3.4 DBMS函数 19](#_Toc30870)

[4 数据库应用系统设计 20](#_Toc16350)

[4.1系统设计目标 20](#_Toc26490)

[4.2 需求分析 21](#_Toc3774)

[4.3 总体设计 23](#_Toc15196)

[4.4 数据库设计 27](#_Toc6195)

[4.5 详细设计与实现 31](#_Toc15302)

[4.6 系统测试 54](#_Toc15415)

[4.7 系统设计与实现总结 71](#_Toc27927)

[5 实验总结 72](#_Toc10443)

# 1 课程任务概述

1. **软件功能学习部分**

完成下列1~2题，并在实践报告中叙述过程，可适当辅以插图（控制在A4三页篇幅以内）

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，要求要有通过数据库的软件功能进行的备份和通过文件形式的脱机备份。

2）练习在新增的数据库上增加用户并配置权限的操作。

1. **Sql练习部分**

假设在某个区域内的所有地点都存储在地点表中。该地区中的所有人员信息存储在人员表中。根据收集到的人员行程建立了疫情期间某个地区的人员行程表。

该地区的部分人员进行了核酸检测，检测结果保存在诊断表中。根据诊断表中的检测结果，对新冠确诊和无症状感染者的密切接触者进行隔离，密切接触者的信息存储到密切接触表中，隔离信息存储到隔离表中。隔离地点共设置了4个，并且每个隔离地点都有容量限制，隔离点的信息存储在隔离地点表中。

**2.1 建表**

**1）创建下列相关的关系，包括主码和外码的说明**

**人员表【人员编号，姓名，电话】**

person (id int, fullname char(20), telephone char(11))

主码为人员编号

**地点表【地点编号，地点名称】**

location (id int, location\_name char(20))

主码为地点编号

**行程表【行程编号，人员编号，所在地点编号，开始时间，结束时间】**

itinerary (id int, p\_id int, loc\_id int, s\_time datetime, e\_time datetime)

主码为行程编号，

**诊断表【诊断编号，人员编号，诊断日期，诊断结果】**

diagnose\_record (id int, p\_id int, datetime datetime, result int)

主码为诊断编号，诊断结果包括：1：新冠确诊，2：无症状感染者，3：正常

**密切接触者表【密切接触编号，被接触者编号，接触日期，接触地点，病例人员编号】**

close\_contact (id int, p\_id int, datetime datetime, loc\_id int, case\_p\_id int)

主码为密切接触编号

**隔离表【隔离编号，被隔离人员编号，开始隔离日期，结束隔离日期，隔离地点编号，隔离状态】**

isolation\_record (id int, p\_id int, s\_date datetime, e\_date datetime, isol\_loc\_id int, state int)

主码为隔离编号，隔离状态：1：正在隔离 2：隔离结束 3：转入医院

**隔离地点表【隔离地点编号，隔离地点名，房间容量】**

isolation\_location (id int, location\_name char(20), capacity int)

主码为隔离房间编号

**2）观察性实验**

验证在设置外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

**2.2 数据更新**

**1）分别用一条sql语句完成对乘车记录表基本的增、删、改的操作；**

**2）批处理操作**

将行程表中所有到达地点2的记录插入到新表location\_record\_2中。

**3）数据导入导出**

通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

**4）观察性实验**

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

**5）触发器实验**

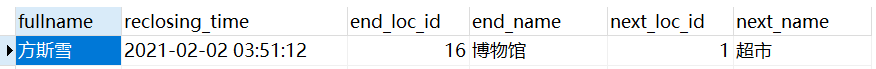
编写一个触发器，用于实现以下完整性控制规则：

当隔离表中的某位隔离人员在诊断表中的诊断结果为“1：新冠确诊”，将隔离状态从“1”改成“3”。

**2.3 查询**

请分别用一条SQL语句完成下列各个小题的查询需求：

1. 查询截至目前的人流量大于30的地点，并按照从高到低排序
2. 查询出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
3. 查询行程表中连续行程中的一对行程地点（例如，方斯雪，在2021-02-02 03:02:12至2021-02-02 03:51:12 在博物馆中，2021-02-02 03:51:12 至 2021-02-02 04:23:12出现超市中，输出【姓名，重合时间，起始地点id，起始地点，结束地点id，结束地点】，查询结果如下图所示）



1. 查询贾涵山的行程情况
2. 查询地名中带有‘店’字的地点名称
3. 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”，查询他接触到的人员的名字
4. 查询正在使用的隔离区名（使用DISTINCT关键字）
5. 查询有出行记录的人员（使用EXISTS关键字）
6. 查询没有达到地点“Today便利店”的人数（使用NOT EXISTS 关键字）
7. 查询去过所有地点的人员
8. 新建一个视图，里面有每个隔离点的id,名称，已用房间，容量
9. 从视图中查询还有空房间的隔离点的剩余房间数目
10. 查询靳宛儿传染的人有哪些
11. 查询每个地点的密切接触者的数量
12. 查询感染人数最多的用户的名字和感染人数
13. 查询2021-02-02 10:00:00到14:00:00，行程记录最频繁的五个人及行程记录数目

**2.4 了解系统的查询性能分析功能（选做）**

选择上述2.3任务中某些较为复杂的SQL语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

**2.5 DBMS函数及存储过程和事务（选做）**

1. 编写一个依据人员编号计算其达到所有地点的次数的自定义函数，并利用其查询至少到达过3个地点的人员。
2. 尝试编写DBMS的存储过程，建立每个隔离点的人数统计表，并通过存储过程更新该表。
3. **数据库应用系统设计**

自行选择所擅长的DBMS软件以及数据库应用系统（客户端程序或者网站）的程序开发工具，参考实验任务书中给的题目例子，拟定一个自己感兴趣的数据库应用系统题目，完成该小型数据库应用系统的设计与实现工作。主要内容包括：需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

通过此次系统设计的综合实践，我们希望达到以下目标：

* 熟悉大型数据库管理系统的结构与组成；
* 熟悉数据库应用系统的设计方法和开发过程；
* 掌握一种大型数据库管理系统(DM5、ORACLE或SQL SERVER)的应用技术和开发工具的使用；
* 熟悉数据库设计工具的使用；
* 熟悉数据库安全的相关知识和技术；

熟悉数据库系统的管理和维护。

# 软件功能学习

## 2.1 任务要求

完成下列1~2题，并在实践报告中叙述过程，可适当辅以插图

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，要求要有通过数据库的软件功能进行的备份和通过文件形式的脱机备份。

2）练习在新增的数据库上增加用户并配置权限的操作。

## 2.2 完成过程

实验环境：

1. 操作系统：ubuntu18.04
2. 数据库：MySql
3. 交互图形界面：Navicat Premium

### 2.2.1 练习两种备份方式

首先需要建立数据库covid9，再建立相关的表，使用老师提供的sql文件导入表的相关数据，导入成功后数据库部分截图如图1所示：

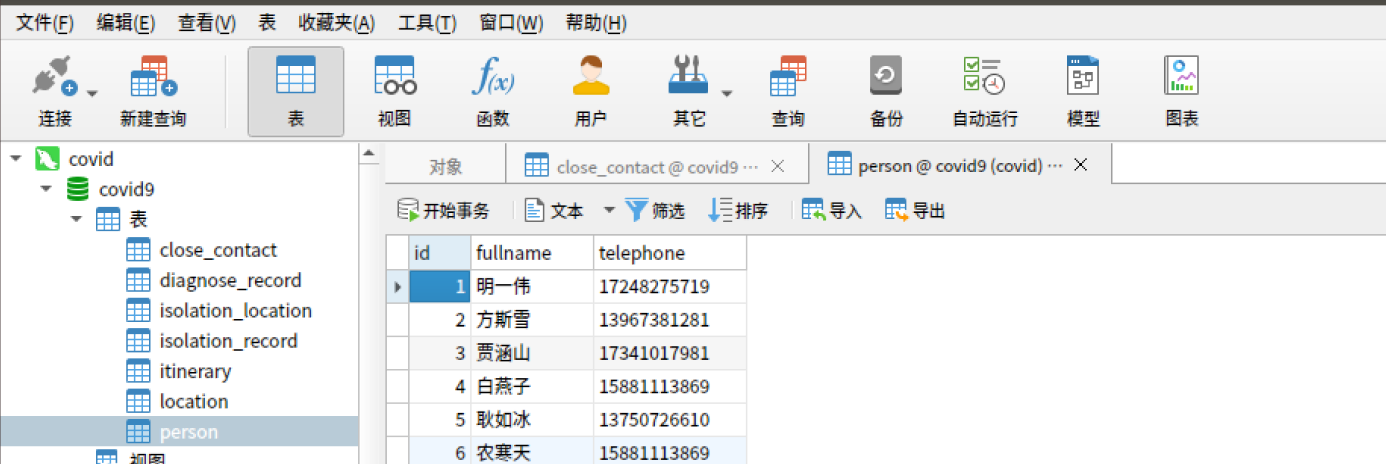


图2-1 导入数据

1. 使用数据库的软件功能进行备份

首先点击navicat premium中的备份按钮，然后点击新建备份按钮，如图2所示：

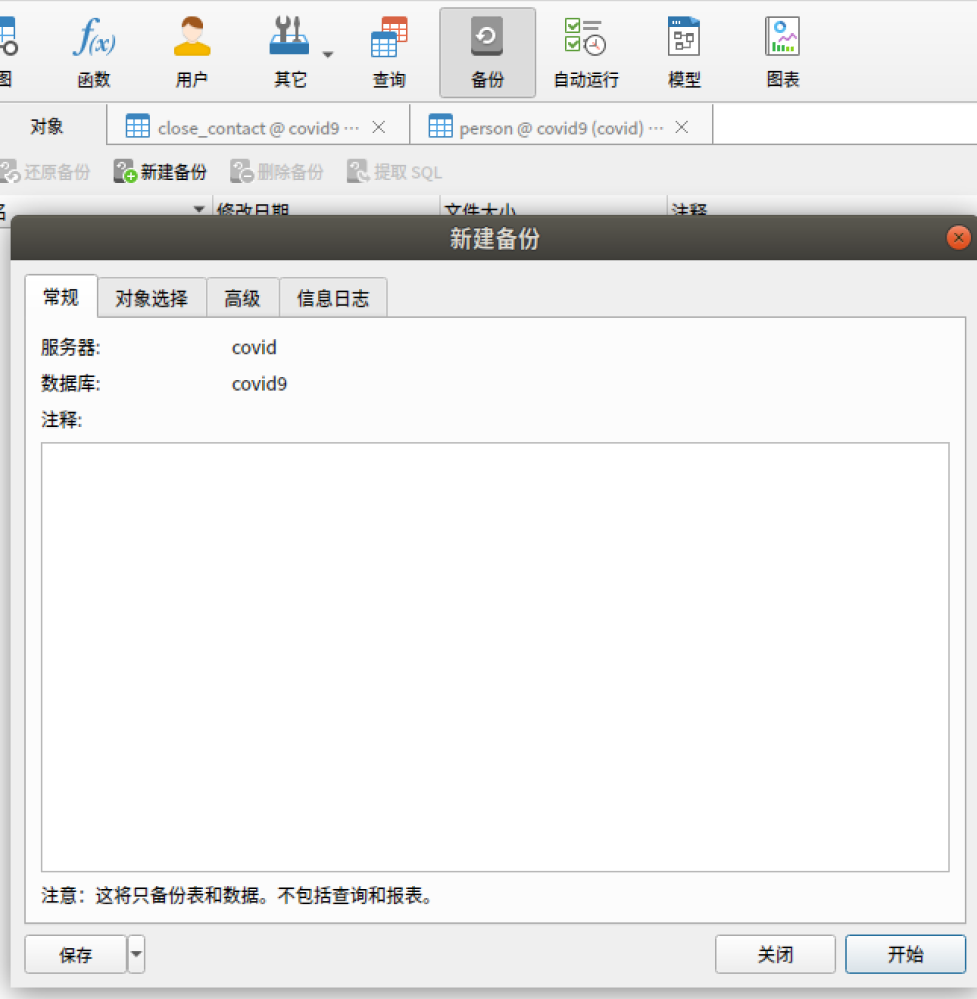


图2-2 备份操作截图

操作结果如图3所示：



图2-3 备份结果

备份好的数据如图4所示：

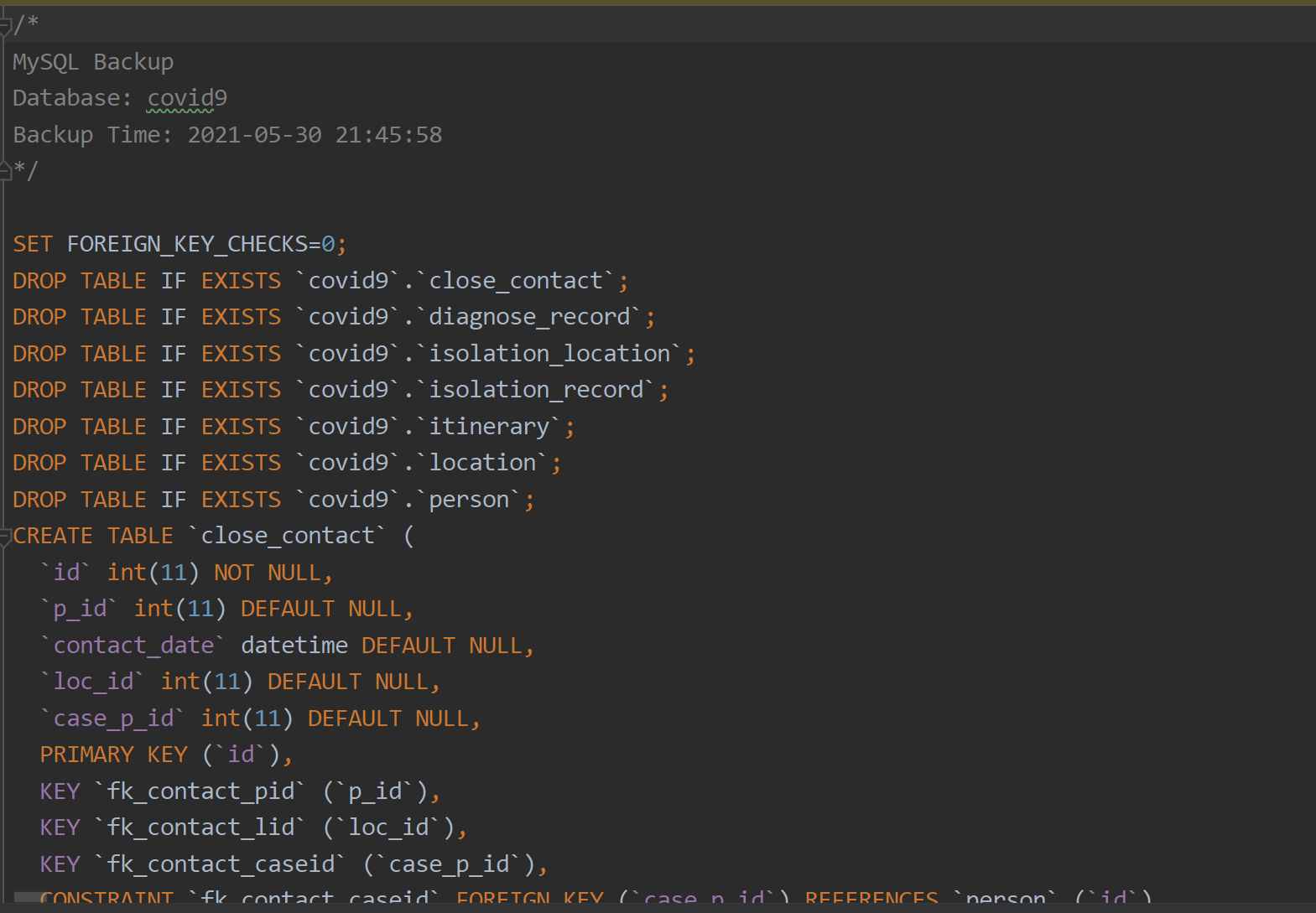


图2-4 备份内容截图

1. 命令行脱机备份

首先需要进入数据库并找到数据文件存放的目录，查找目录如图5所示：

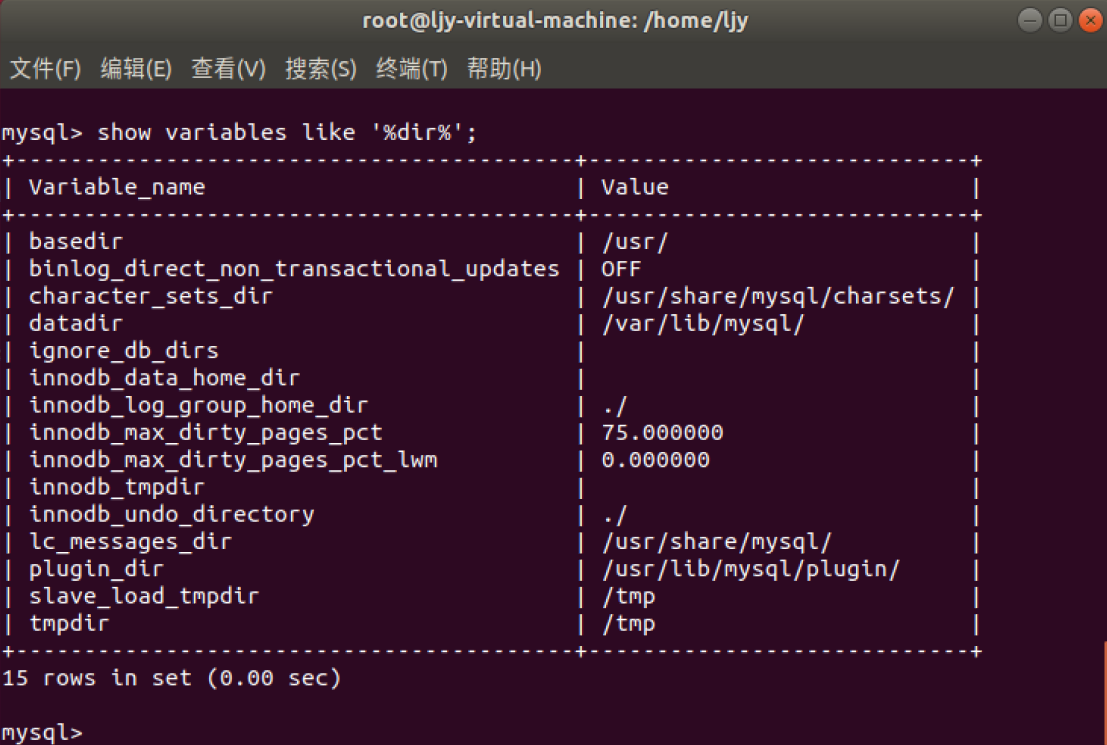


图2-5 查找数据文件存放目录

其中数据目录就是datadir的所在位置，即/var/lib/mysql/

执行以下命令

cd /var/lib/ #进入数据目录的上一级目录

service mysql stop #关闭mysql服务

tar jcvf ~/backup.tar.bz2 mysql/ #打包压缩该目录到根目录

命令行脱机备份成功，备份文件即为backup.tar.bz2

当需要恢复备份的时候，执行以下命令

cd ~/ #进入备份文件的保存目录

tar jxvf backup.tar.bz2 mysql/ #解压

gunzip -r mysql/ #递归解压

service mysql stop #关闭服务

rm -r /var/lib/mysql/ #删除原目录

mv ~/mysql/ /var/lib/ #把备份的文件移动到/var/lib/里面去替代原来的mysql

service mysql start #重启mysql服务

### 2.2.2练习增加用户并配置权限

使用navicat premium用户设置，用户信息如图6所示：

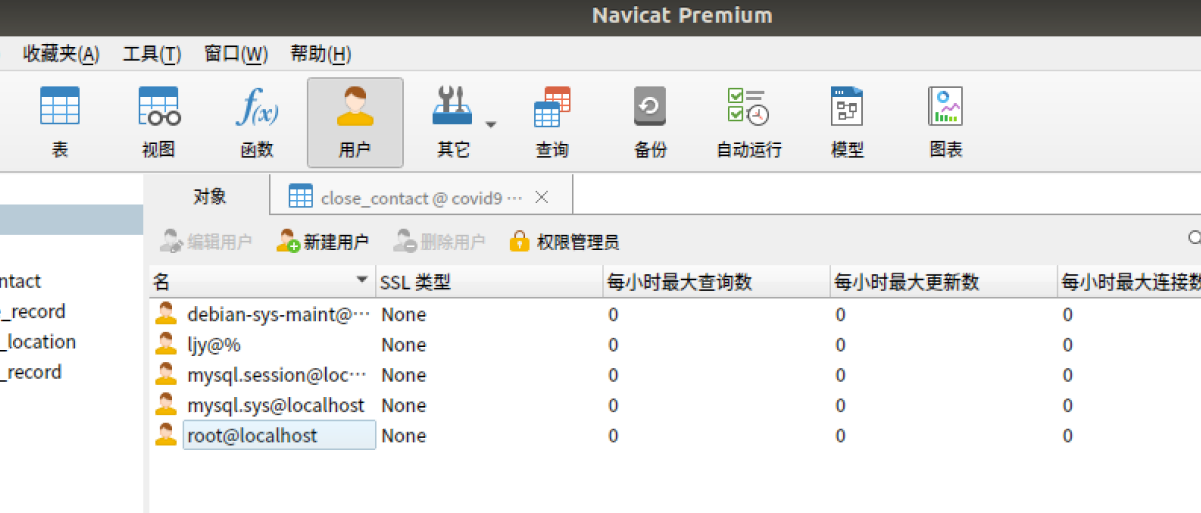


图2-6 用户信息

选择新建用户，具体设置如图7所示

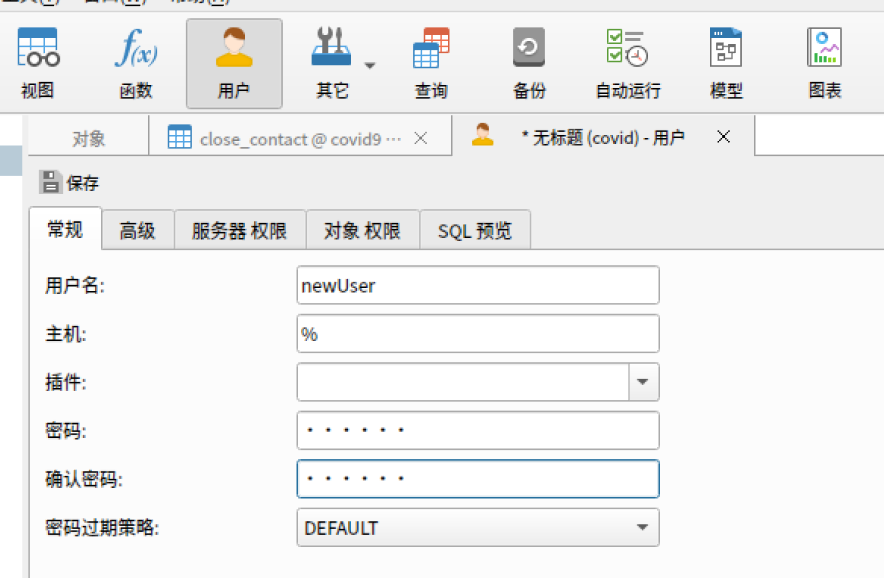


图2-7 新建用户设置

除了Insert权限不给予，其他权限全部给予newUser。

然后登录用户newUser，如图8所示：

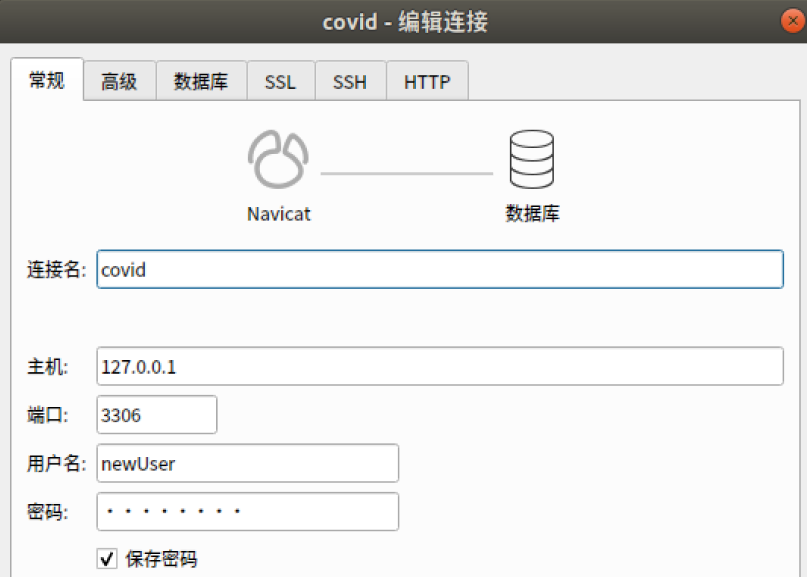


图2-8 登录新建用户newUser

因为没有insert权限，因此我们尝试insert一条数据进入表person，操作如图9所示：

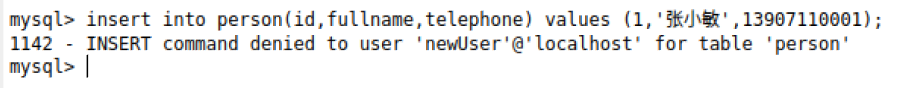


图2-9 新建用户进行插入操作

由图中可知，由于没有insert权限，因此插入失败

然后我们修改权限，使新用户newUser获得insert权限，再进行一次插入操作，如图9所示：

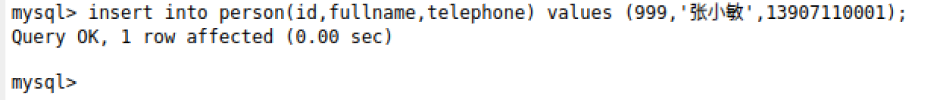


图2-10 获得权限后再进行插入操作

由图中可知，拥有了insert权限，插入成功。

# 3 SQL练习

## 3.1 建表

假设在某个区域内的所有地点都存储在地点表中。该地区中的所有人员信息存储在人员表中。根据收集到的人员行程建立了疫情期间某个地区的人员行程表。

该地区的部分人员进行了核酸检测，检测结果保存在诊断表中。根据诊断表中的检测结果，对新冠确诊和无症状感染者的密切接触者进行隔离，密切接触者的信息存储到密切接触表中，隔离信息存储到隔离表中。隔离地点共设置了4个，并且每个隔离地点都有容量限制，隔离点的信息存储在隔离地点表中。

### **根据以上背景，请在DBMS中创建这些关系，包括主码和外码的说明，并写出指定关系的建表SQL语句**

人员表【人员编号，姓名，电话】

person (id int, fullname char(20), telephone char(11))

主码为人员编号

create table person

(

id int,

fullname char(20) NOT NULL,

telephone char(11) NOT NULL ,

CONSTRAINT pk\_person PRIMARY KEY(id)

);

地点表【地点编号，地点名称】

location (id int, location\_name char(20))

主码为地点编号

create table location

(

id int,

location\_name char(20) NOT NULL ,

CONSTRAINT pk\_location PRIMARY KEY (id)

);

行程表【行程编号，人员编号，所在地点编号，开始时间，结束时间】

itinerary (id int, p\_id int, loc\_id int, s\_time datetime, e\_time datetime)

主码为行程编号；p\_id是外码，被参照表是person；loc\_id是外码，被参照表是location

create table itinerary

(

id int,

p\_id int,

loc\_id int,

s\_time datetime,

e\_time datetime,

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)

);

诊断表【诊断编号，人员编号，诊断日期，诊断结果】

diagnose\_record (id int, p\_id int, datetime datetime, result int)

主码为诊断编号，诊断结果包括：1：新冠确诊，2：无症状感染者，3：正常；p\_id是外码，被参照表是person；

create table diagnose\_record

(

id int,

p\_id int,

diagnose\_date datetime,

result int,

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

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

);

密切接触者表【密切接触编号，被接触者编号，接触日期，接触地点，病例人员编号】

close\_contact (id int, p\_id int, datetime datetime, loc\_id int, case\_p\_id int)

主码为密切接触编号；p\_id是外码，被参照表是person；loc\_id是外码，被参照表是location；case\_p\_id是外码，被参照表是person；

create table close\_contact

(

id int,

p\_id int,

contact\_date datetime,

loc\_id int,

case\_p\_id int,

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)

);

隔离表【隔离编号，被隔离人员编号，开始隔离日期，结束隔离日期，隔离地点编号，隔离状态】

isolation\_record (id int, p\_id int, s\_date datetime, e\_date datetime, isol\_loc\_id int, state int)

主码为隔离编号，隔离状态：1：正在隔离 2：隔离结束 3：转入医院；

create table isolation\_location

(

id int,

location\_name char(20),

capacity int,

CONSTRAINT pk\_isolation\_loc PRIMARY KEY (id)

);

隔离地点表【隔离地点编号，隔离地点名，房间容量】

isolation\_location (id int, location\_name char(20), capacity int)

主码为隔离房间编号；p\_id是外码，被参照表示person；isol\_loc\_id是外码，被参照表是isolation\_location;

create table isolation\_record

(

id int,

p\_id int,

s\_date datetime,

e\_date datetime,

isol\_loc\_id int,

state int,

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)

);

### **观察性实验**

验证在设置外码时是否一定要参考被参照关系的主码。

通过分析我们可以知道，外码一定是被参照表的主码，当外码不是主码是建表会出现错误，我通过新建2个表来测试以上结果。

1. 建立一个新表person1，其中id设为外码，被参照表为person，因为person的id是person表的主码，因此建表成功，如图10所示：

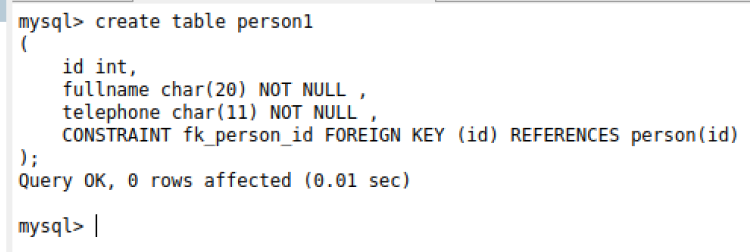


图3-1 建立person1表

1. 建立一个新表person2，其中fullname设为外码，被参照表为person，因为person的fullname不是person表的主码，因此建表失败，如图11所示：

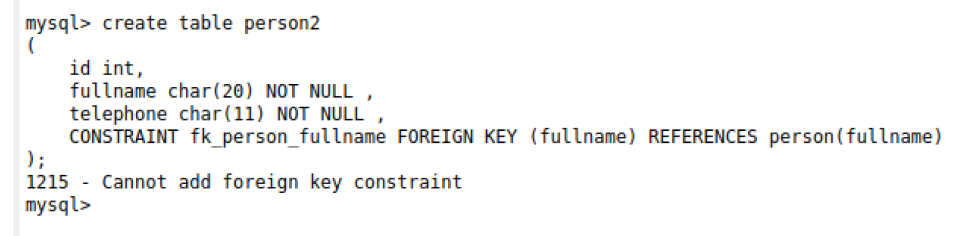


图3-2 建立person2表

### **数据准备**

在软件功能学习任务中已经完成插入实验数据的任务。

## 3.2 数据更新

1. **分别用一条sql语句完成对人员表基本的增、删、改的操作；**

insert into person(id,fullname,telephone) values (3,'孙二娘',13307100003);

delete from person where id=2 and fullname='李大锤';

update person set telephone=13607176668 where fullname='张小敏' and id=1;

1. **数据导入导出**

在软件功能学习任务中已经完成相关任务，在此不再重复说明。

1. **观察性实验**

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

建立一个与person表类似的person1表，但不设置主码，然后插入重复元组，再对已有数据进行删除、修改，如图12所示：

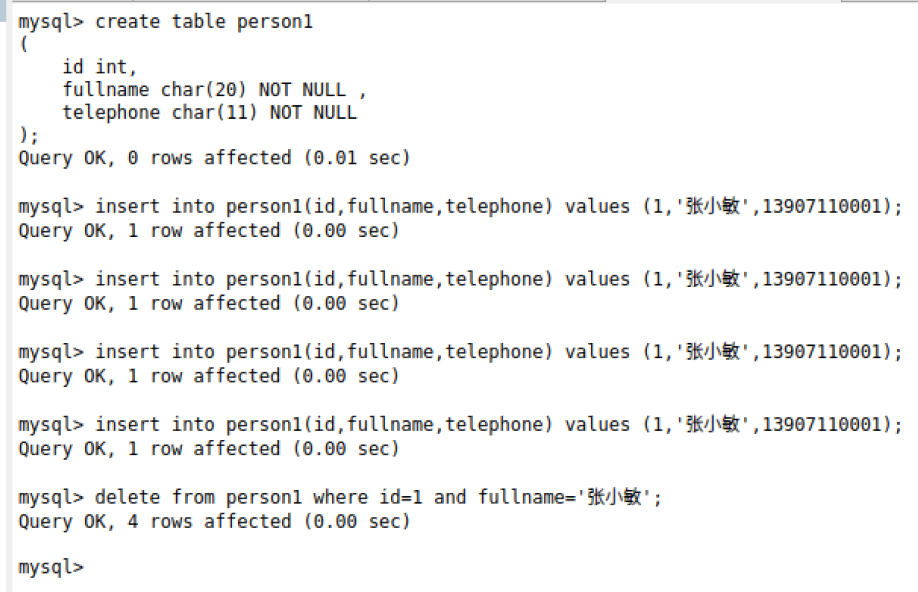


图3-3 删除测试

删除的结果是四个重复的元组都被删除，person1又变回了一个空表。

然后再重新插入四个重复元组，如图13所示：

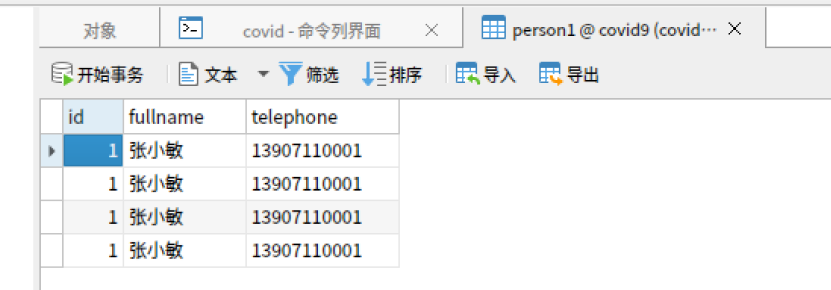
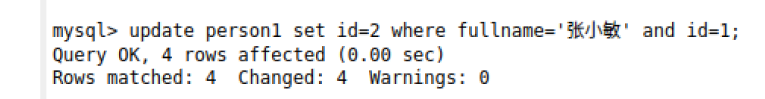


图3-4 插入重复元组的表

将id修改成2，结果如图14所示：



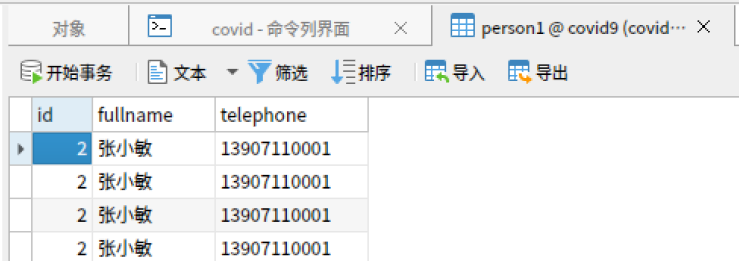


图3-5 修改重复元组后的表

所以当一个关系中没有主码时，插入重复元组，然后进行删除、修改操作的时候会将所有匹配的元组全部进行相关操作。

1. **触发器实验**

编写一个触发器，用于实现以下完整性控制规则：

当隔离表中的某位隔离人员在诊断表中的诊断结果为“1：新冠确诊”，将隔离状态从“1”改成“3”。

delimiter ;;

create trigger task

after update on diagnose\_record

for each row

begin

if NEW.result=1 then

update isolation\_record

set state=3

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

end if;

end

;;

create trigger task1

after insert on diagnose\_record

for each row

begin

if NEW.result=1 then

update isolation\_record

set state=3

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

end if;

end

;;

delimiter ;

## 3.3 查询

请分别用一条SQL语句完成下列各个小题的需求：

1. 查询截至目前的人流量大于30的地点，并按照从高到低排序

select location\_name,count(\*) as visitors

FROM location,itinerary

where location.id=itinerary.loc\_id

group by loc\_id

having visitors>30

order by visitors desc,location\_name;

1. 查询出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

select isolation\_location.location\_name,count(\*) as number

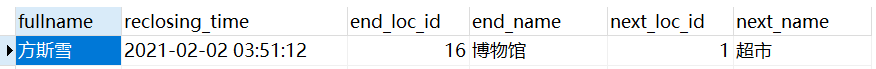
from isolation\_location,isolation\_record

where isolation\_record.isol\_loc\_id=isolation\_location.id and isolation\_record.state=1

group by isol\_loc\_id

order by number desc ,location\_name;

1. 查询行程表中连续行程中的一对行程地点（例如，方斯雪，在2021-02-02 03:02:12至2021-02-02 03:51:12 在博物馆中，2021-02-02 03:51:12 至 2021-02-02 04:23:12出现超市中，输出【姓名，重合时间，起始地点id，起始地点，结束地点id，结束地点】，查询结果如下图所示）



select person.id,fullname,telephone,i1.e\_time as reclosing\_time,i1.loc\_id as loc1,l1.location\_name as address1,i2.loc\_id as loc2,l2.location\_name as address2

from person,itinerary i1,itinerary i2,location l1,location l2

where person.id>30

and person.id=i1.p\_id

and person.id=i2.p\_id

and i1.e\_time=i2.s\_time

and l1.id=i1.loc\_id

and l2.id=i2.loc\_id

order by person.id,reclosing\_time

1. 查询充珉瑶和贾涵山的行程情况

select person.fullname,person.telephone,l.location\_name,itinerary.s\_time,itinerary.e\_time

from person left outer join itinerary on person.id = itinerary.p\_id left outer join location l on itinerary.loc\_id = l.id

where fullname='充珉瑶' or fullname='贾涵山'

order by person.id desc,s\_time ;

1. 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。

select location.id,location.location\_name

from location

where location\_name like '%店%'

order by location.id;

1. 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”，查询他接触到的人员的名字。

select distinct person.fullname,person.telephone

from itinerary,location,person

where (itinerary.s\_time between '2021-02-02 20:05:40' and '2021-02-02 21:25:40'

or itinerary.e\_time between '2021-02-02 20:05:40' and '2021-02-02 21:25:40')

and itinerary.loc\_id=(

select location.id

where location\_name='活动中心'

)

and itinerary.p\_id=person.id

order by fullname;

1. 查询仍正在使用的隔离点名称。

select isolation\_location.location\_name

from isolation\_location

where isolation\_location.id in (

select distinct isol\_loc\_id

from isolation\_record

where state=1

order by isol\_loc\_id

)

order by isolation\_location.id;

1. 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。

select fullname,telephone

from person

where exists(

select distinct \*

from itinerary,location

where p\_id=person.id and fullname !='充珉瑶'

)

order by person.id limit 0,30;

1. 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）。

select count(\*) as number

from person

where not exists(

select \*

from itinerary

where p\_id=person.id and loc\_id=(

select location.id

from location

where location\_name='Today便利店'

)

)

1. 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。

select fullname

from person

where not exists(

select \*

from location

where not exists(

select \*

from itinerary

where p\_id=person.id and loc\_id=location.id

)

)

order by fullname

1. 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。

create view isolation\_location\_status as

select il.id, location\_name, capacity,COUNT(if(state=1,1,null)) occupied

from isolation\_record right outer join isolation\_location il on isolation\_record.isol\_loc\_id = il.id

group by il.id;

1. 从视图isolation\_location\_status中查询各隔离点的剩余房间数。

select location\_name,capacity-occupied as available\_rooms

from isolation\_location\_status

order by id

1. 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。

select p2.fullname,p2.telephone

from person p1,itinerary i1,itinerary i2,person p2

where p1.fullname='靳宛儿'

and i1.p\_id=p1.id

and i2.loc\_id=i1.loc\_id

and (i1.s\_time between i2.s\_time and i2.e\_time

or i1.e\_time between i2.s\_time and i2.e\_time

or i2.s\_time between i1.s\_time and i1.e\_time

or i2.e\_time between i1.s\_time and i1.e\_time)

and i2.p\_id!=p1.id

and i2.p\_id=p2.id

order by p2.fullname

1. 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。

select l1.location\_name ,count(\*) as close\_contact\_number

from close\_contact,location l1

where close\_contact.loc\_id=l1.id

group by l1.id

order by close\_contact\_number desc,location\_name

1. 查询感染人数最多的人。

select \*

from (select case\_p\_id,fullname,count(\*) as infected\_number

from person,close\_contact

where person.id=case\_p\_id

group by person.id) b

where infected\_number=(

select max(infected\_number)

from (select case\_p\_id,fullname,count(\*) as infected\_number

from person,close\_contact

where person.id=case\_p\_id

group by person.id) a

)

1. 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。

select person.fullname,count(\*) as record\_number

from person,itinerary

where person.id=itinerary.p\_id

and

(

s\_time between '2021-02-02 10:00:00' and '2021-02-02 14:00:00'

or e\_time between '2021-02-02 10:00:00' and '2021-02-02 14:00:00'

or '2021-02-02 10:00:00' between s\_time and e\_time

or '2021-02-02 14:00:00' between s\_time and e\_time

)

group by person.fullname

order by record\_number desc ,person.fullname limit 0,3

1. 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。

select location\_name,capacity

from isolation\_location

where capacity=(

select max(capacity)

from isolation\_location

where capacity!=(

select max(capacity)

from isolation\_location

)

)

## 3.4 DBMS函数

编写一个依据人员编号计算其达到所有地点的次数的自定义函数，并利用其查询至少到达过3个地点的人员。

(1)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:

delimiter ;;

create function Count\_Records(uid INT)

returns int

begin

return if((select count(\*) as number

from person,itinerary,location

where person.id=p\_id

and location.id=itinerary.loc\_id

and person.id=uid

group by fullname) is null,0,(select count(\*) as number

from person,itinerary,location

where person.id=p\_id

and location.id=itinerary.loc\_id

and person.id=uid

group by fullname));

end;;

delimiter ;

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。

select person.id,fullname,telephone

from person,itinerary,location

where person.id=itinerary.p\_id

and location.id=itinerary.loc\_id

and Count\_Records(person.id)>=3

group by person.id

order by person.id

# 4 数据库应用系统设计

## 4.1系统设计目标

### 4.1.1 系统应用背景

在当今社会已经走上心里高速公路的大背景下，国家争议前所未有的速度向前发展，作为现代化企业管理，计算机的普及与计算机技术的应用已经是企业向前发展的一个重要手段。而每一个公司甚至公司内部各部门的实际情况也不尽相同，这就对企业管理提出了更高的要求。

随着经济的发展，企业正向着大型化、规模化发展，而对于大中型企业，员工、职称等跟工资管理有关的信息随之急剧增加。在这种情况下单靠人工来处理员工的工资不但显得力不从心，而且极容易出错。所以设计一套实用的管理信息系统也是必要的。本次实验主要设计了一个企业的工资管理信息系统。该系统的具体任务就是由计算机来代替一系列诸如增加新员工、删除就员工、工资修改、查询、统计等操作。这样就使办公人员可以轻松快捷地完成工资管理的任务。

### 4.1.2 总体目标

系统应达到以下目标

对于员工用户：

查询员工用户的个人信息

查询员工用户的考勤信息

查询员工用户的加班信息

查询员工用户的月工资详情

查询员工用户的工资汇总信息

对于管理员：

查询、修改、增加、删除所有员工的详细信息

查询、修改、增加、删除所有公司部门的详细信息

查询、修改、增加、删除所有公司职位的详细信息

查询、修改、增加、删除所有员工考勤情况

查询、修改、增加、删除所有员工加班情况

查询所有员工的月工资详情

查询所有员工的工资汇总情况

对于系统：

根据员工加班类型、加班时间、加班日期自动计算该次加班所得加班费

根据员工每月的总工资来计算年终奖金

如果删除部门或者职位信息时，自动将员工表中相关位置置为空

根据员工的底薪、津贴、加班津贴、缺勤扣除的情况自动计算每月工资

有便于使用的界面

## 4.2 需求分析

### 4.2.1 系统功能划分

图4-1 系统功能划分图

### 4.2.2 数据和数据流

数据库中有以下表：

1. 用户账号密码表
2. 管理员账号密码表
3. 员工基本信息表
4. 公司部门信息表
5. 公司职位信息表
6. 加班类型信息表
7. 员工加班津贴表
8. 员工考勤信息表
9. 员工月工资表
10. 员工工资汇总表

其中表1、2用于处理登录和注册事件。表3到表8用于存储员工的工资详情数据，表9、10用于存储根据工资详情数据计算出的工资总额信息。

数据流图如下图。



图4-2 顶层数据流图



图4-3 底层数据流图

## 4.3 总体设计

### 4.3.1 系统实现

系统前端使用JavaFx实现，系统后端使用mysql实现。总体设计架构图如下图所示。



图4-4 系统架构示意图

### 4.3.2 系统功能

系统主要功能模块流程图如下：

1. 登录模块



图4-5 登录选项流程图

1. 用户/管理员登录模块



图4-6 登录模块流程图

1. 用户操作模块



图4-7 用户操作流程图

1. 管理员操作模块

图4-8 管理员操作流程图

## 4.4 数据库设计

### 4.4.1 ER图设计

图4-9 系统ER图

### 4.4.2 数据库表设计

1.用户登录信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| id | Char(20) | 用户账号 | 主码 | Not null |
| Password | Char(20) | 密码 |  | Not null |
| p\_id | Int | 员工id | 外码 | Not null |

2.管理员登录信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| Id | Char(20) | 管理员账号 | 主码 | Not null |
| Password | Char(20) | 密码 |  | Not null |

3.员工基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| ID | Int auto\_increment | 员工id | 主码 | NOT NULL |
| NAME | CHAR(20) | 员工姓名 |  | NOT NULL |
| SEX | CHAR(2) | 员工性别 |  | NOT NULL |
| AGE | INT | 员工年龄 |  |  |
| ADDRESS | CHAR(40) | 员工住址 |  |  |
| TELEPHONE | CHAR(11) | 员工电话号码 |  |  |
| typeID | Int | 员工工种编号 | 外码 | On delete set null |
| deptID | Int | 员工所属部门编号 | 外码 | On delete set null |

4.公司部门表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| deptID | Int auto\_increment | 部门编号 | 主码 | NOT NULL |
| NAME | CHAR(20) | 部门名称 |  | NOT NULL |
| ALLOWANCE | DECIMAL(9,4) | 部门津贴 |  | 默认为0 |
| MANAGER | CHAR(20) | 部门经理 |  |  |
| TELEPHONE | CHAR(11) | 办公室电话 |  |  |

5.员工工种表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| typeID | Int auto\_increment | 员工工种编号 | 主码 | NOT NULL |
| NAME | CHAR(20) | 员工工种名称 |  | NOT NULL |
| deptID | Int | 员工所属部门编号 | 外码 | On delete set null |
| SALARY | DECIMAL(9,4) | 员工所属工种基本工资 |  |  |
| LEVEL | INT | 员工工种等级 |  | NOT NULL |

6.加班类型表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| ewtypeID | Int auto\_increment | 加班类型编号 | 主码 | NOT NULL |
| ewtypeNAME | CHAR(20) | 加班类型名称 |  | NOT NULL |
| ewtypeSALARY | DECIMAL(9,4) | 加班额外薪酬 |  |  |

7.员工加班津贴表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| ID | Int | 员工编号 | 外码 | NOT NULL |
| ewdate | DATE | 加班日期 |  | NOT NULL |
| ewtypeID | Int | 加班类型编号 | 外码 | NOT NULL |
| ewtime | Int | 加班时间（天） |  | NOT NULL |
| ewSALARY | DECIMAL(9,4) | 加班薪酬 |  |  |

8.员工考勤表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| Attenddate | DATE | 缺勤、迟到日期 |  | NOT NULL |
| ID | Int | 缺勤、迟到人员编号 | 外码 | NOT NULL |
| Deduction | DECIMAL(9,4) | 扣除工资 |  | NOT NULL |

9.员工月工资表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| ID | Int | 员工编号 | 外码 | NOT NULL |
| Jan | DECIMAL(9,4) | 员工1月份工资 |  |  |
| Feb | DECIMAL(9,4) | 员工2月份工资 |  |  |
| Mar | DECIMAL(9,4) | 员工3月份工资 |  |  |
| Apr | DECIMAL(9,4) | 员工4月份工资 |  |  |
| May | DECIMAL(9,4) | 员工5月份工资 |  |  |
| Jun | DECIMAL(9,4) | 员工6月份工资 |  |  |
| Jul | DECIMAL(9,4) | 员工7月份工资 |  |  |
| Aug | DECIMAL(9,4) | 员工8月份工资 |  |  |
| Sept | DECIMAL(9,4) | 员工9月份工资 |  |  |
| Oct | DECIMAL(9,4) | 员工10月份工资 |  |  |
| Nov | DECIMAL(9,4) | 员工11月份工资 |  |  |
| Dec | DECIMAL(9,4) | 员工12月份工资 |  |  |

10.员工总工资表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 完整性约束 | 空值情况 |
| ID | Int | 员工编号 | 外码 | NOT NULL |
| Bonus | DECIMAL(9,4) | 员工年终奖金 |  |  |
| Year\_salary | DECIMAL(9,4) | 员工本年工资 |  |  |

## 4.5 详细设计与实现

### 4.5.1 开发环境

操作系统：windows 10

Jdk：jdk11

开发语言：java+sql

数据库：mysql 5.7.34-0 ubuntu18.04.1

### 4.5.2 系统业务流程图

根据用户权限类型，分为普通用户业务流程图和管理员业务流程图：

1.普通员工用户业务流程图：

图4-10 用户业务流程图

2.管理员用户业务流程图：

图4-11 管理员业务流程图

### 4.5.3 关键技术和算法说明

1.数据库连接模块

算法设计：为方便连接数据库，设计Dbutil工具类用于连接数据库并返回连接，使用DriverManager连接到服务器，然后使用服务器ip以及端口连接到数据库，填入账号密码并获得连接，之后再其他类中直接使用连接conn即可。

具体代码如下：

**public class** Dbutil {  
 **static** String *UserName*;  
 **static** String *Password*;  
 **static** String *url*=**"jdbc:mysql://localhost:3306/Salary\_Management\_System"**;  
 **static** String *user\_name*=**"root"**;  
 **static** String *password*=**"123456"**;  
 **static** Connection *conn*=**null**;  
 **public static** Connection myConnection() {  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**); *conn*= DriverManager.*getConnection*(*url*,*user\_name*,*password*);  
 **return** *conn*;  
 } **catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 } **catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
}

2.用户登录模块

算法设计：从前端获取用户输入的账户名和密码，然后再后端连接服务器，判断用户输入的账户名是否在数据库中有记录，再判断密码是否匹配，如果不匹配则登录失败，匹配则登录成功，进入用户使用界面。

具体代码如下：

**public class** User\_Login\_Controller {  
  
 **public** User\_Login\_Controller(){}  
 **public static int** *p\_id*;  
 @FXML  
 **private** TextField **User\_name**;  
 @FXML  
 **private** PasswordField **User\_password**;  
 @FXML  
 **private** Button **User\_login\_button**;  
 @FXML  
 **public void** Click\_User\_login(ActionEvent event) {  
 Connection connection= Dbutil.*myConnection*();  
 **if**(connection==**null**){  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"连接数据库失败！"**);  
 alert.show();  
 }  
 **else**{  
 **if**(**User\_name**.getText().equals(**""**))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"账号不可为空"**);  
 alert.showAndWait();  
 **return** ;  
 }**else if**(**User\_password**.getText().equals(**""**))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"密码不可为空"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 }  
 **try**{  
 String sql=**"select** *\** **from User\_login\_info\_table where id=?"**;  
 PreparedStatement ptmt=connection.prepareStatement(sql);  
 ptmt.setString(1,**User\_name**.getText());  
 ResultSet rs=ptmt.executeQuery();  
 **if**(!(rs.next()))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"该用户名不存在"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 String realPassword=rs.getString(**"Password"**);  
 String inputPassword=**User\_password**.getText();  
 *p\_id*=rs.getInt(3);  
 **if**(!(realPassword.equals(inputPassword)))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"密码错误，请重新输入"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 **else** {  
 Stage currentStage=(Stage)**User\_login\_button**.getScene().getWindow();  
 currentStage.close();  
 Parent root= FXMLLoader.*load*(getClass().getResource(**"../User.fxml"**));  
 Stage newStage=**new** Stage();  
 newStage.setTitle(**"工资管理系统 当前用户："**+**User\_name**.getText()+**" 员工ID："**+*p\_id*);  
 newStage.setScene(**new** Scene(root));  
 newStage.show();  
 ptmt.close();  
 connection.close();  
 }  
  
 } **catch** (SQLException | IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

3.管理员登录模块

算法设计：从前端获取用户输入的账户名和密码，然后再后端连接服务器，判断用户输入的账户名是否在数据库中有记录，再判断密码是否匹配，如果不匹配则登录失败，匹配则登录成功，进入管理员使用界面。

**public class** Admin\_Login\_Controller {  
 @FXML  
 **private** TextField **Admin\_name**;  
 @FXML  
 **private** PasswordField **Admin\_password**;  
 @FXML  
 **private** Button **Admin\_login\_button**;  
  
 @FXML  
 **void** Click\_Admin\_login() {  
 Connection connection= Dbutil.*myConnection*();  
 **if**(connection==**null**){  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"连接数据库失败！"**);  
 alert.show();  
 }  
 **else**{  
 **if**(**Admin\_name**.getText().equals(**""**))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"账号不可为空"**);  
 alert.showAndWait();  
 **return** ;  
 }**else if**(**Admin\_password**.getText().equals(**""**))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"密码不可为空"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 }  
 **try**{  
 String sql=**"select** *\** **from Admin\_login\_info\_table where id=?"**;  
 PreparedStatement ptmt=connection.prepareStatement(sql);  
 ptmt.setString(1,**Admin\_name**.getText());  
 ResultSet rs=ptmt.executeQuery();  
 **if**(!(rs.next()))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"该用户名不存在"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 String realPassword=rs.getString(**"Password"**);  
 String inputPassword=**Admin\_password**.getText();  
 **if**(!(realPassword.equals(inputPassword)))  
 {  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"密码错误，请重新输入"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 **else** {  
 Stage currentStage=(Stage)**Admin\_login\_button**.getScene().getWindow();  
 currentStage.close();  
 Parent root= FXMLLoader.*load*(getClass().getResource(**"../Admin.fxml"**));  
 Stage newStage=**new** Stage();  
 newStage.setTitle(**"工资管理系统 当前用户："**+**Admin\_name**.getText());  
 newStage.setScene(**new** Scene(root));  
 newStage.show();  
 ptmt.close();  
 connection.close();  
 }  
  
 } **catch** (SQLException | IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

4.查询功能模块

由于查询功能的实现基本上相同，因此只举一个例子。

算法设计：首先先写出查询出我们想要的结果的sql语句，将查询结果放入一个ObservableList中，类型为查询出的结果的类，然后将查询结果赋予给这个新建的列表中，然后再讲查询得到的表显示到前端的TableView之中。查询员工基本信息的示例代码如下：

**public class** Search\_e\_info {  
  
  
 **public static void** Show\_select\_e\_info(TableView<Employee> table, TableColumn id, TableColumn name, TableColumn sex, TableColumn age, TableColumn addr, TableColumn tel, TableColumn dept, TableColumn type, String select\_name)  
 {  
 String sql\_search\_e=**"select id, Employee\_Basic\_Information.name, sex, age, address, Employee\_Basic\_Information.telephone, Department.NAME, Employee\_Type.NAME from Employee\_Basic\_Information,Employee\_Type,Department where Employee\_Type.typeID=Employee\_Basic\_Information.typeID and Department.deptID=Employee\_Basic\_Information.deptID and Employee\_Basic\_Information.NAME like \'%"** + select\_name + **"%\'"**;  
 ObservableList<Employee> Info\_list=FXCollections.*observableArrayList*();  
 id.setCellValueFactory(**new** PropertyValueFactory<>(**"id"**));  
 name.setCellValueFactory(**new** PropertyValueFactory<>(**"name"**));  
 sex.setCellValueFactory(**new** PropertyValueFactory<>(**"sex"**));  
 age.setCellValueFactory(**new** PropertyValueFactory<>(**"age"**));  
 addr.setCellValueFactory(**new** PropertyValueFactory<>(**"address"**));  
 tel.setCellValueFactory(**new** PropertyValueFactory<>(**"telephone"**));  
 dept.setCellValueFactory(**new** PropertyValueFactory<>(**"department"**));  
 type.setCellValueFactory(**new** PropertyValueFactory<>(**"type"**));  
 table.getItems().clear();  
 **try** {  
 Admin\_Controller.*ptmt*=Admin\_Controller.*conn*.prepareStatement(sql\_search\_e);  
*// ptmt.setString(1,select\_name);* Admin\_Controller.*rs*=Admin\_Controller.*ptmt*.executeQuery();  
 **while**(Admin\_Controller.*rs*.next())  
 {  
 Info\_list.add(**new** Employee(Admin\_Controller.*rs*.getInt(1),Admin\_Controller.*rs*.getString(2),Admin\_Controller.*rs*.getString(3),Admin\_Controller.*rs*.getInt(4),Admin\_Controller.*rs*.getString(5),Admin\_Controller.*rs*.getString(6),Admin\_Controller.*rs*.getString(7),Admin\_Controller.*rs*.getString(8)));  
 }  
  
 table.setItems(Info\_list);  
 }**catch** (Exception e)  
 {  
 e.printStackTrace();  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }**finally** {  
 **if**(Admin\_Controller.*ptmt*!=**null**&&Admin\_Controller.*stmt*!=**null**&&Admin\_Controller.*rs*!=**null**)  
 {  
 **try** {  
 Admin\_Controller.*ptmt*=**null**;  
 Admin\_Controller.*stmt*=**null**;  
 Admin\_Controller.*rs*=**null**;  
 }**catch** (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

5.添加数据功能模块

由于添加功能的实现基本相同，因此只举一个列子。

算法设计：首先点击添加按钮弹出添加模块的操作界面，然后选择需要添加的信息，后端从前端获取需要添加的信息，然后根据需要添加的信息写sql语句，执行即可将数据添加进入数据库。添加员工基本信息的代码示例如下：

**public class** Insert\_in\_e\_basic {  
 @FXML  
 **private** TextField **I\_e\_age**;  
  
 @FXML  
 **private** TextField **I\_e\_tel**;  
  
 @FXML  
 **private** TextField **I\_e\_name**;  
  
 @FXML  
 **private** TextField **I\_e\_addr**;  
  
 @FXML  
 **private** TextField **I\_e\_sex**;  
  
 @FXML  
 **private** ChoiceBox **I\_e\_type\_box**;  
  
 @FXML  
 **private** ChoiceBox **I\_e\_dept\_box**;  
  
  
 @FXML  
 **void** Insert\_e\_dept\_box\_setItems() {  
 e\_choice\_dept\_box\_setItems.*setItems*(**I\_e\_dept\_box**);  
 }  
  
 @FXML  
 **void** Insert\_e\_type\_box\_setItems() {  
 e\_choice\_type\_box\_setItems.*setItems*(**I\_e\_type\_box**);  
 }  
  
 @FXML  
 **void** Click\_insert\_e\_info() {  
 Connection connection= Dbutil.*myConnection*();  
 PreparedStatement ptmt=**null**;  
 ResultSet rs=**null**;  
 **int** dept\_id=0;  
 **int** type\_id=0;  
 String sql;  
 sql=**"select deptID from Department where NAME=\'"**+**I\_e\_dept\_box**.getValue()+**"\'"**;  
 **try**{  
 ptmt=connection.prepareStatement(sql);  
 rs=ptmt.executeQuery();  
 **while**(rs.next())  
 {  
 dept\_id=rs.getInt(1);  
 }  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 sql=**"select typeID from Employee\_Type where name=\'"**+**I\_e\_type\_box**.getValue()+**"\'"**;  
 **try**{  
 ptmt=connection.prepareStatement(sql);  
 rs=ptmt.executeQuery();  
 **while**(rs.next())  
 {  
 type\_id=rs.getInt(1);  
 }  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 sql=**"insert into Employee\_Basic\_Information(name, sex, age, address, telephone, typeid, deptid) values (\'"**+**I\_e\_name**.getText()+**"\',\'"**+**I\_e\_sex**.getText()+**"\',"**+**I\_e\_age**.getText()+**",\'"**+**I\_e\_addr**.getText()+**"\',\'"**+**I\_e\_tel**.getText()+**"\',"**+type\_id+**","**+dept\_id+**")"**;  
 **try** {  
 ptmt=connection.prepareStatement(sql);  
 ptmt.execute();  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }**finally** {  
 **if**(connection!=**null**&&ptmt!=**null**)  
 {  
 **try** {  
 connection.close();  
 ptmt.close();  
 }**catch** (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

6.修改数据功能模块

由于修改数据功能的实现基本相同，因此只举一个例子。

算法设计：点击修改数据按钮，弹出修改数据的界面，然后后端从该界面获取需要修改的信息，根据得到的对应信息获得对应sql语句，然后执行该sql语句即可将数据修改。修改员工基本信息代码示例如下：

@FXML  
**private** ChoiceBox **e\_basic\_info\_Choice\_box\_type**;  
@FXML  
**private** ChoiceBox **e\_basic\_info\_Choice\_box\_dept**;  
@FXML  
**void** Click\_update\_e\_basic\_info() {  
 Connection connection= Dbutil.*myConnection*();  
 PreparedStatement ptmt=**null**;  
 ResultSet rs=**null**;  
 **int** dept\_id=0;  
 **int** type\_id=0;  
 String sql;  
 String dept=(String)**e\_basic\_info\_Choice\_box\_dept**.getValue();  
 sql=**"select deptID from Department where NAME=\'"**+dept+**"\'"**;  
 **try**{  
 ptmt=connection.prepareStatement(sql);  
 rs=ptmt.executeQuery();  
 **while**(rs.next())  
 {  
 dept\_id=rs.getInt(1);  
 }  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();  
 Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 String type=(String)**e\_basic\_info\_Choice\_box\_type**.getValue();  
 sql=**"select typeID from Employee\_Type where name=\'"**+type+**"\'"**;  
 **try**{  
 ptmt=connection.prepareStatement(sql);  
 rs=ptmt.executeQuery();  
 **while**(rs.next())  
 {  
 type\_id=rs.getInt(1);  
 }  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }  
 sql=**"update Employee\_Basic\_Information set deptID="**+dept\_id+**",typeID="**+type\_id+**" where ID="**+ Admin\_Controller.*e\_basic\_p\_id*;  
 **try** {  
 ptmt=connection.prepareStatement(sql);  
 ptmt.execute();  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();Alert alert=**new** Alert(Alert.AlertType.***INFORMATION***);  
 alert.setHeaderText(**"操作失败"**);  
 alert.showAndWait();  
 **return** ;  
 }**finally** {  
 **if**(connection!=**null**&&ptmt!=**null**)  
 {  
 **try** {  
 connection.close();  
 ptmt.close();  
 }**catch** (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 }  
}

7.删除数据功能模块

由于删除数据功能的实现基本相同，因此只举一个例子。

算法设计：当选择一条数据，点击修改窗口的删除按钮，将该条记录的主码送给后端，后端根据该主码写出相应的删除sql语句，然后执行该sql语句，删除对应的数据记录。删除员工基本信息的代码如下：

**void** Click\_delete\_e\_basic\_info() {  
 Connection connection=Dbutil.*myConnection*();  
 PreparedStatement ptmt=**null**;  
 String sql=**"delete from Employee\_Basic\_Information where ID="**+Admin\_Controller.*e\_basic\_p\_id*;  
 **try** {  
 ptmt=connection.prepareStatement(sql);  
 ptmt.execute();  
 }**catch** (SQLException e)  
 {  
 e.printStackTrace();  
 }**finally** {  
 **if**(connection!=**null**&&ptmt!=**null**)  
 {  
 **try** {  
 connection.close();  
 ptmt.close();  
 }**catch** (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 }  
}

### 4.5.4 数据库触发器、存储过程定义及实现

1.删除员工触发器

当删除某个员工时，该员工的相关记录都应该删除，定义一个触发器，当发生删除员工事件时，同时删除与该员工相关的其他数据，代码如下所示：

**delimiter** ;;  
**create trigger** delete\_employee  
 **before delete on** Employee\_Basic\_Information  
 **for each row  
 begin  
 delete from** Employee\_Attend **where** Employee\_Attend.**ID**=**OLD**.**ID**;  
 **delete from** Employee\_month\_Salary **where** Employee\_month\_Salary.**ID**=**OLD**.**ID**;  
 **delete from** Extra\_work\_allowance **where** Extra\_work\_allowance.**ID**=**OLD**.**ID**;  
 **delete from** Sum\_Month\_Salary **where** Sum\_Month\_Salary.**ID**=**OLD**.**ID**;  
 **delete from** User\_login\_info\_table **where** User\_login\_info\_table.**p\_id**=**OLD**.**ID**;  
 **end** ;;  
**delimiter** ;

2.计算员工的每月工资

员工的月工资计算公式为 员工月工资=部门津贴+每月底薪+加班费-缺勤扣费。定义一个存储过程，遍历员工表，计算每个员工的每月工资详情，然后再将结果插入员工月工资表中，当需要计算月工资的时候执行call calculate\_Month\_salary();即可，具体代码如下：

**delimiter** ;;  
**create procedure** *calculate\_Month\_salary*()  
**begin  
 declare** v1 **int**;  
 **declare** month\_count **int**;  
 **declare** p\_id **int**;  
 **declare** sum **int**;  
 **set** v1=0;  
 **set** month\_count=0;  
 **set** sum=0;  
 **delete from** Employee\_month\_Salary;  
 **while** v1<(**select** *count*(*\**) **from** Employee\_Basic\_Information) **do  
 set** p\_id=(**select ID  
 from** Employee\_Basic\_Information  
 **limit** v1,1);  
 **insert into** Employee\_month\_Salary(**ID**, **Jan**, **Feb**, **Mar**, **Apr**, **May**, **Jun**, **Jul**, **Aug**, **Sept**, **Oct**, **Nov**, **`Dec`**)**VALUES** (p\_id,0,0,0,0,0,0,0,0,0,0,0,0);  
 **set** v1=v1+1;  
 **end while** ;  
  
 **while** month\_count<12 **do  
 set** v1=0;  
 **while** v1<(**select** *count*(*\**)  
 **from** Employee\_Basic\_Information) **do  
  
 set** p\_id=(**select ID  
 from** Employee\_Basic\_Information  
 **limit** v1,1);  
  
 **set** sum=0;  
 **set** sum=(**select ALLOWANCE from** Department **where deptID**=(**select deptID from** Employee\_Basic\_Information **where ID**=p\_id))  
 +(**select SALARY from** Employee\_Type **where typeID**=(  
 **select typeID  
 from** Employee\_Basic\_Information  
 **where ID**=p\_id  
 ))  
 +*ifnull*((**select ewSALARY  
 from** Extra\_work\_allowance  
 **where ID**=p\_id **and** *month*(**ewdate**)=**case** month\_count **when** 0 **then** 1 **when** 1 **then** 2 **when** 2 **then** 3 **when** 3 **then** 4 **when** 4 **then** 5 **when** 5 **then** 6 **when** 6 **then** 7 **when** 7 **then** 8 **when** 8 **then** 9 **when** 9 **then** 10 **when** 10 **then** 11 **when** 11 **then** 12 **end**),0)  
 -*ifnull*((**select** *sum*(**Deduction**)  
 **from** Employee\_Attend  
 **where ID**=p\_id **and** *month*(**notAttenddate**)=**case** month\_count **when** 0 **then** 1 **when** 1 **then** 2 **when** 2 **then** 3 **when** 3 **then** 4 **when** 4 **then** 5 **when** 5 **then** 6 **when** 6 **then** 7 **when** 7 **then** 8 **when** 8 **then** 9 **when** 9 **then** 10 **when** 10 **then** 11 **when** 11 **then** 12 **end**),0);  
 **case** month\_count **when** 0 **then update** Employee\_month\_Salary **set Jan**=sum **where ID**=p\_id;  
 **when** 1 **then update** Employee\_month\_Salary **set Feb**=sum **where ID**=p\_id;  
 **when** 2 **then update** Employee\_month\_Salary **set Mar**=sum **where ID**=p\_id;  
 **when** 3 **then update** Employee\_month\_Salary **set Apr**=sum **where ID**=p\_id;  
 **when** 4 **then update** Employee\_month\_Salary **set May**=sum **where ID**=p\_id;  
 **when** 5 **then update** Employee\_month\_Salary **set Jun**=sum **where ID**=p\_id;  
 **when** 6 **then update** Employee\_month\_Salary **set Jul**=sum **where ID**=p\_id;  
 **when** 7 **then update** Employee\_month\_Salary **set Aug**=sum **where ID**=p\_id;  
 **when** 8 **then update** Employee\_month\_Salary **set Sept**=sum **where ID**=p\_id;  
 **when** 9 **then update** Employee\_month\_Salary **set Oct**=sum **where ID**=p\_id;  
 **when** 10 **then update** Employee\_month\_Salary **set Nov**=sum **where ID**=p\_id;  
 **when** 11 **then update** Employee\_month\_Salary **set `Dec`**=sum **where ID**=p\_id;  
 **end case**;  
 **set** v1=v1+1;  
 **end while** ;  
 **set** month\_count=month\_count+1;  
 **end while** ;  
**end**;;  
**delimiter** ;

3.计算加班薪酬

员工加班薪酬=员工加班时间\*加班单位时间薪酬，定义一个存储过程，计算有加班记录的员工的所有加班薪酬，通过在加班类型表中获取加班的单位时间薪酬，再在员工加班记录表中得到加班时间，两者相乘即得到员工该次加班薪酬，需要计算加班薪酬时直接执行call calculate\_ewSalary();即可，具体代码如下：

**delimiter** ;;  
**create procedure** *calculate\_ewSalary*()  
**begin  
 declare** v1 **int**;  
 **declare** p\_id **int**;  
 **declare** ewtype\_ID **int**;  
 **declare** ew\_time **int**;  
 **declare** ew\_salary **int**;  
 **declare** Salary **int**;  
 **set** v1=0;  
 **set** Salary=0;  
 **while** v1<(**select** *count*(*\**) **from** Extra\_work\_allowance) **do  
 set** p\_id=(**select ID from** Extra\_work\_allowance **limit** v1,1);  
 **set** ewtype\_ID=(**select ewtypeID from** Extra\_work\_allowance **where ID**=p\_id);  
 **set** ew\_time=(**select ewtime from** Extra\_work\_allowance **where ID**=p\_id);  
 **set** ew\_salary=(**select ewtypeSALARY from** Extra\_work\_type **where ewtypeID**=ewtype\_ID);  
 **update** Extra\_work\_allowance **set ewSALARY**=ew\_time\*ew\_salary **where ID**=p\_id;  
 **set** v1=v1+1;  
 **end while** ;  
**end** ;;  
**delimiter** ;

4.计算年末总工资和年终奖金

需要计算每个员工的年终奖金，年终奖金公式为：员工的年终奖金＝（员工本年度的工资总和＋津贴的总和）/12，因此也需要计算本年度工资总和，因此直接从员工每月工资表中，将每月的工资加起来即可得到本年度工资总和，然后再计算年终奖金即可，需要计算时直接执行call calculate\_Year\_bonus();即可，代码如下：

**delimiter** ;;  
**create procedure** *calculate\_Year\_bonus*()  
**begin  
 declare** v1 **int**;  
 **declare** sum **float**;  
 **declare** yearsalary **int**;  
 **declare** p\_id **int**;  
 **declare** allowance **int**;  
 **delete from** Sum\_Month\_Salary;  
 **set** v1=0;  
 **while** v1<(**select** *count*(*\**) **from** Employee\_Basic\_Information) **do  
 set** p\_id=(**select ID  
 from** Employee\_Basic\_Information  
 **limit** v1,1);  
 **set** sum=0;  
 **set** yearsalary=0;  
 **set** allowance=0;  
 **set** yearsalary=(**select Jan from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Feb from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Mar from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Apr from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select May from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Jun from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Jul from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Aug from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Sept from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Oct from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select Nov from** Employee\_month\_Salary **where ID**=p\_id)+  
 (**select `Dec` from** Employee\_month\_Salary **where ID**=p\_id);  
 **set** allowance=*ifnull*((**select** *sum*(**ewSALARY**) **from** Extra\_work\_allowance **where id**=p\_id),0);  
 **set** sum=yearsalary/12;  
 **insert into** Sum\_Month\_Salary(**ID**,**Bonus**,**year\_salary**) **values** (p\_id,sum,yearsalary-allowance);  
 **set** v1=v1+1;  
 **end while** ;  
**end**;;  
  
**delimiter** ;

### 4.5.5 界面设计

1.登录选择界面



图4-12 登录选择界面

2.用户登录界面



图4-13 用户登录界面

3.用户操作界面



图4-14 用户操作界面

4.管理员登录界面



图4-15 管理员登录界面

5.管理员操作界面



图4-16 管理员操作界面

## 4.6 系统测试

### 4.6.1 测试环境

操作系统：Windows 10

虚拟机：ubuntu 18.04

数据库：mysql

开发工具：javaFX+SceneBuilder

开发平台：Intellij IDEA

### 4.6.2 功能测试

1.用户账户错误



图4-17 用户名错误

2.用户密码错误



图4-18 用户密码错误

3.用户查询员工信息

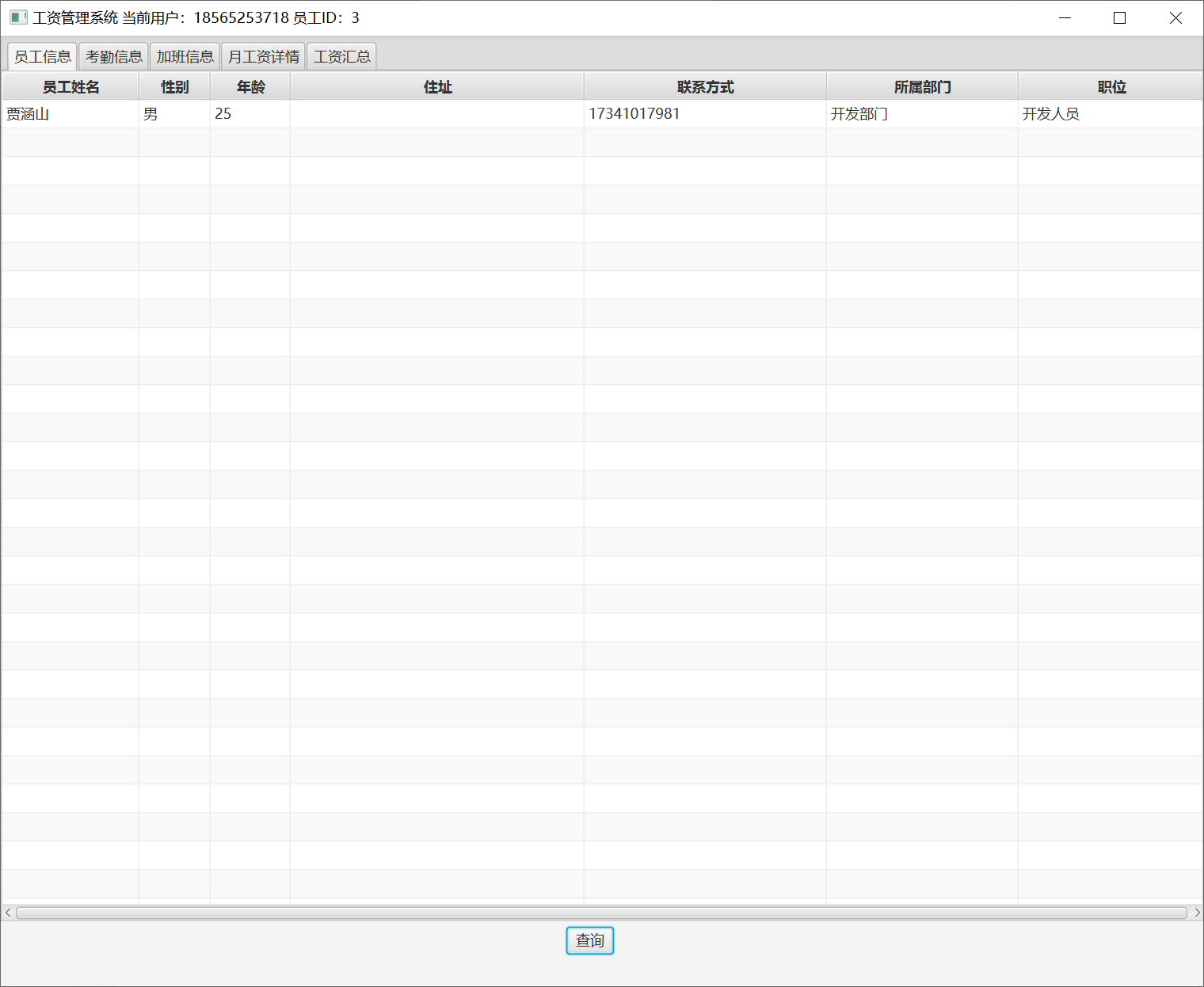


图4-19 用户账号查询员工信息

4.用户查询考勤信息

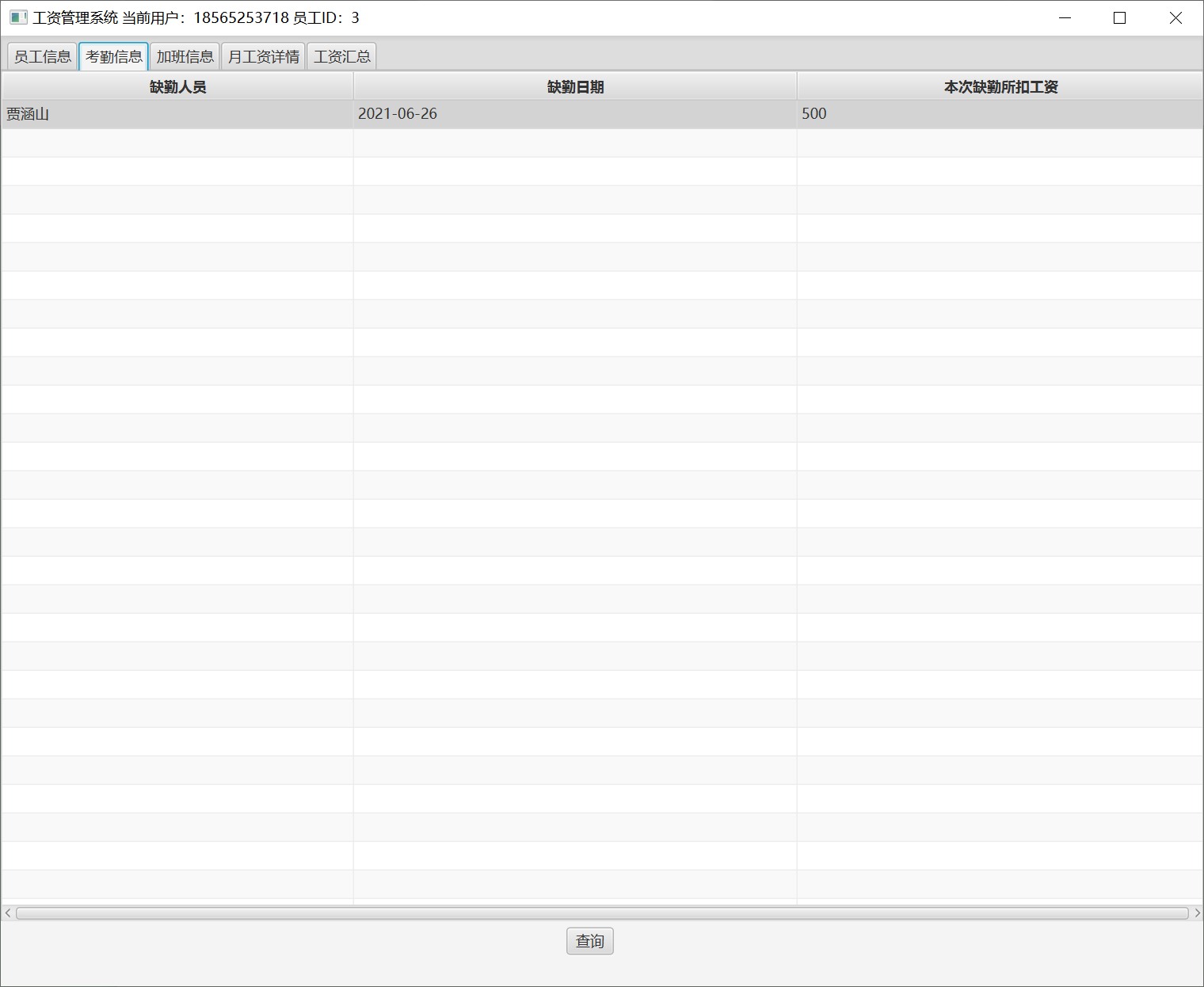


图4-20 用户账号查询考勤信息

5.用户查询加班信息

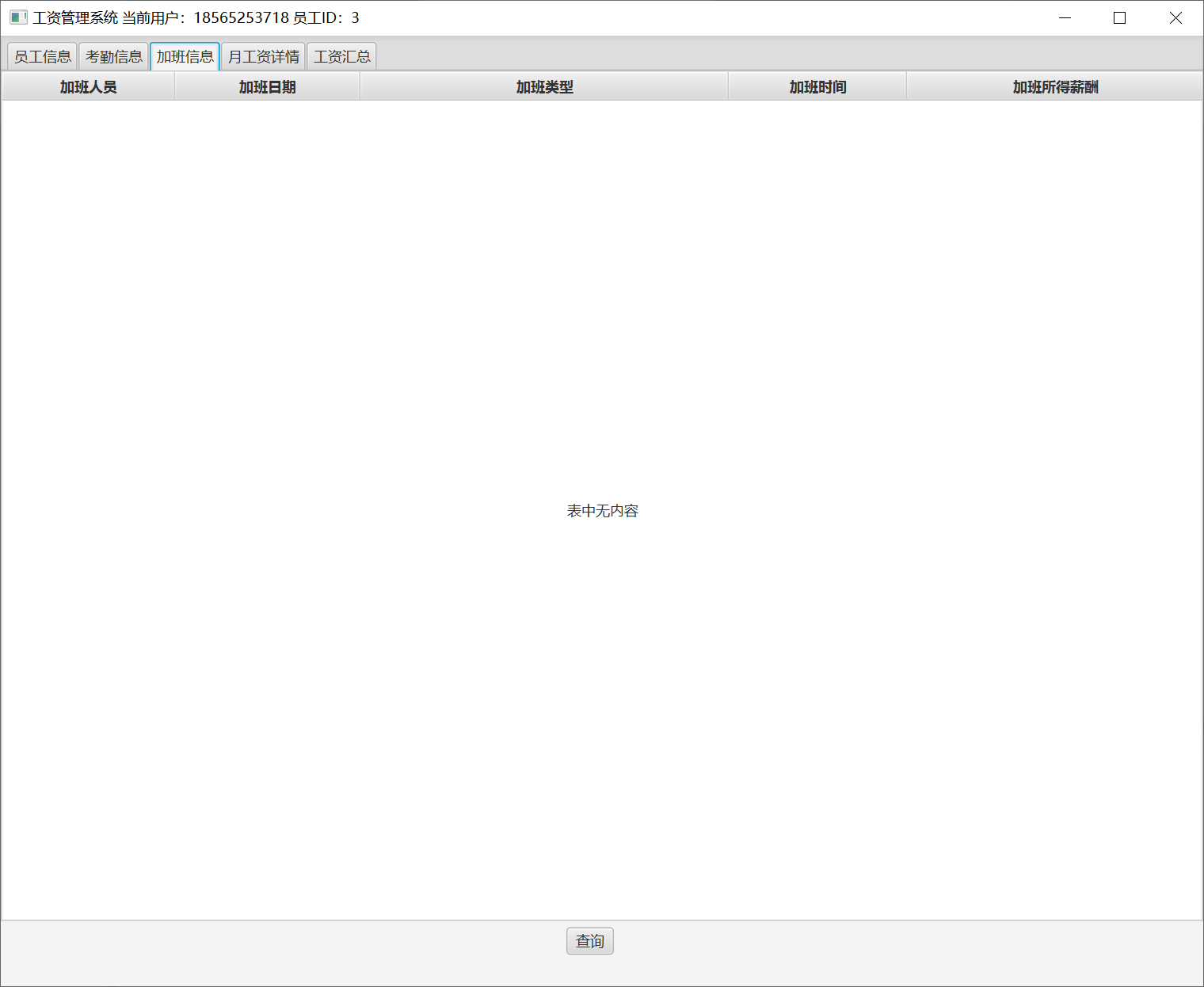


图4-21 用户查询加班信息（图中表格无内容是由于该员工无加班记录）

6.用户查询月工资详情

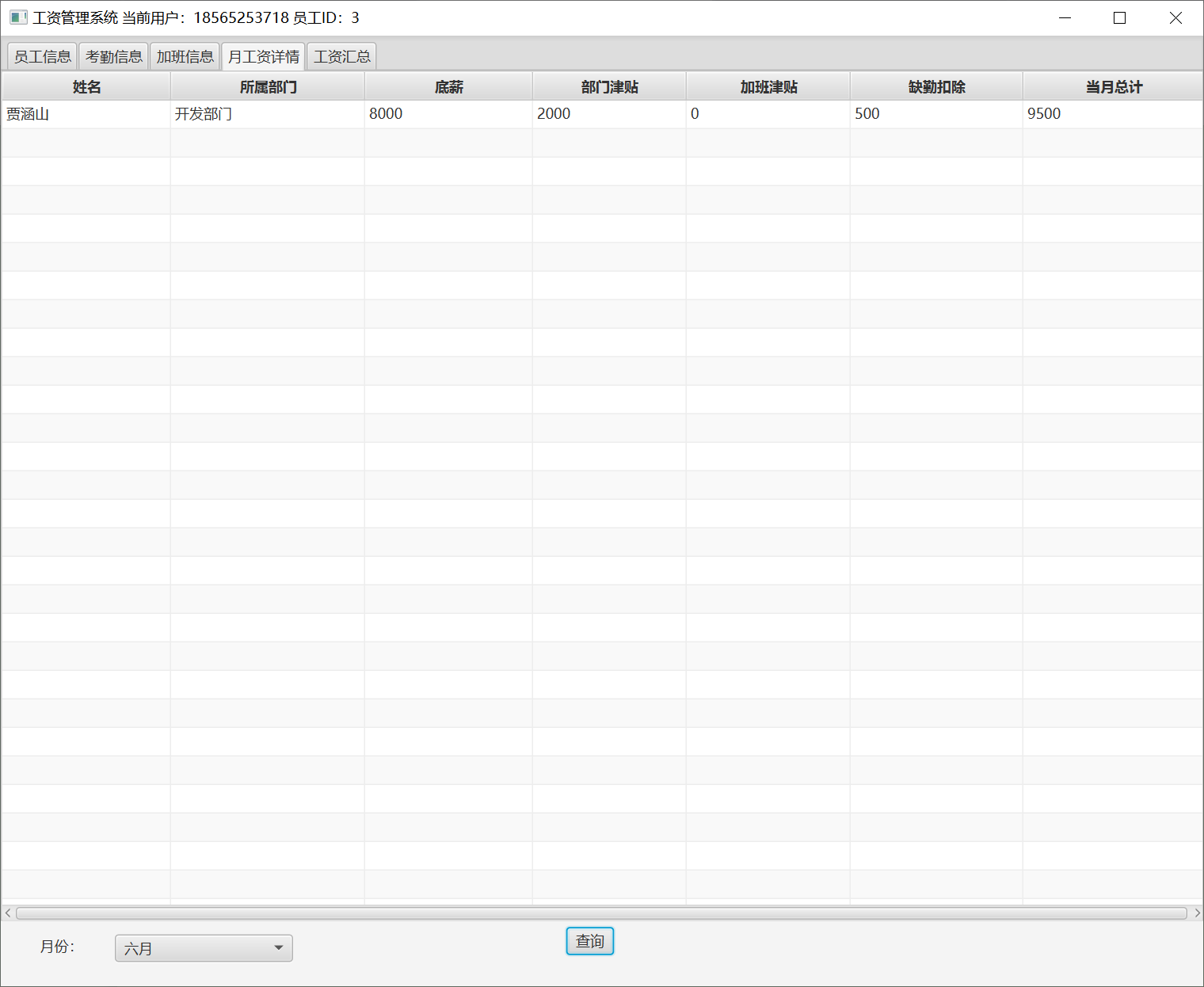


图4-22 用户查询某月的工资详情

7.用户查询工资汇总信息

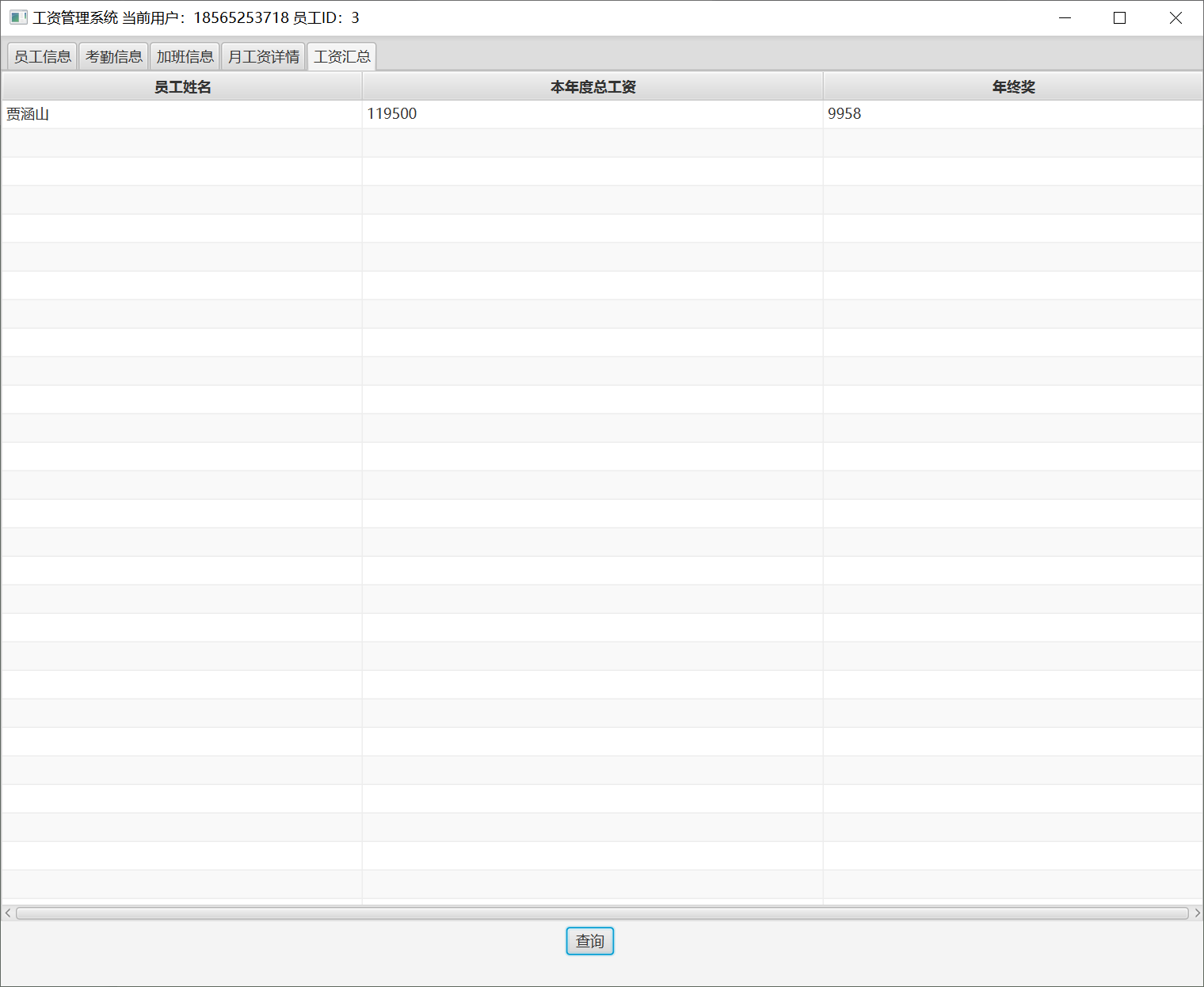


图4-23 用户查询工资汇总信息

8.管理员账户错误



图4-24 管理员账户错误

9.管理员密码错误



图4-25 管理员密码错误

10.管理员查询员工信息

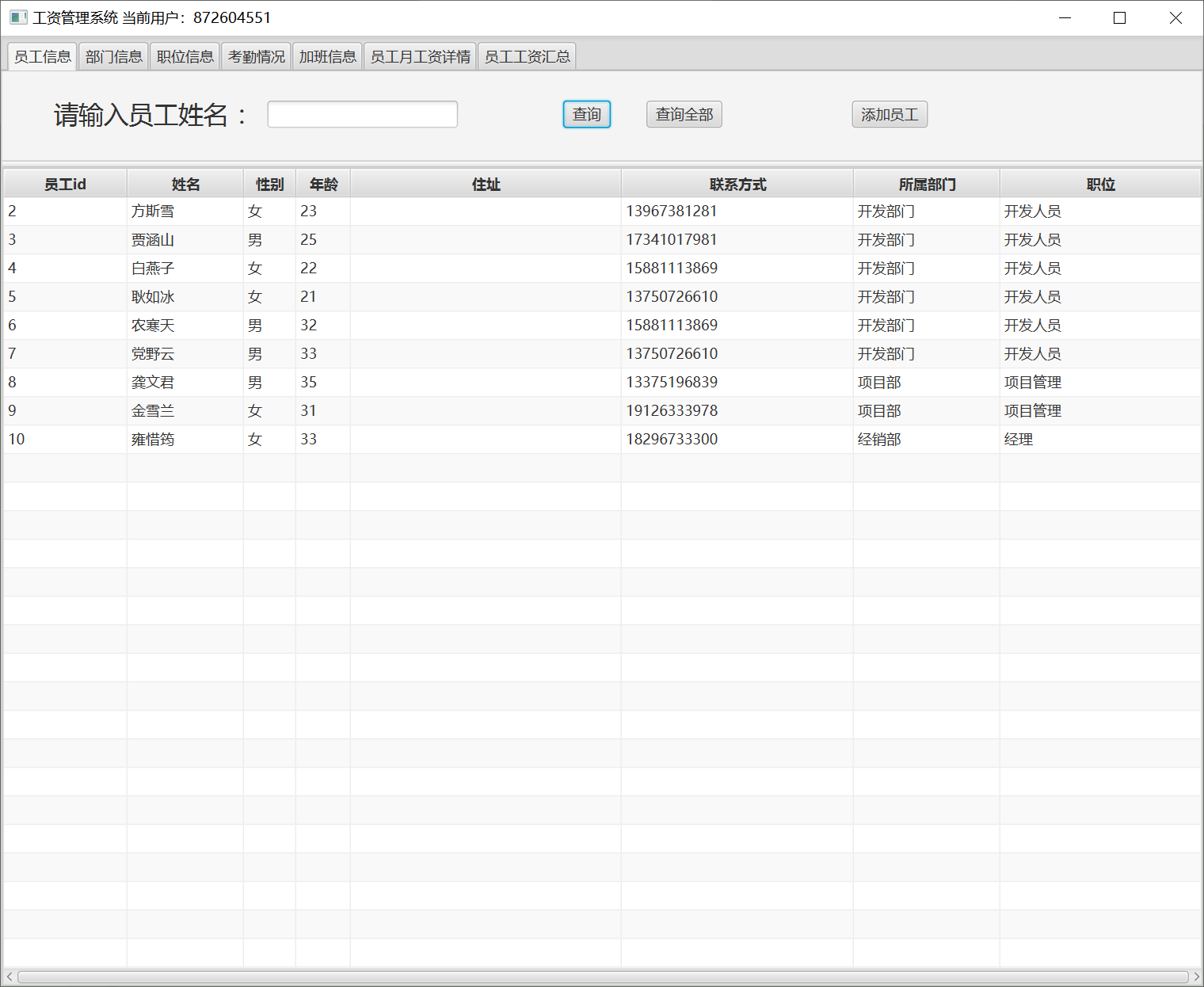


图4-26 管理员查询员工信息

11.管理员修改员工信息

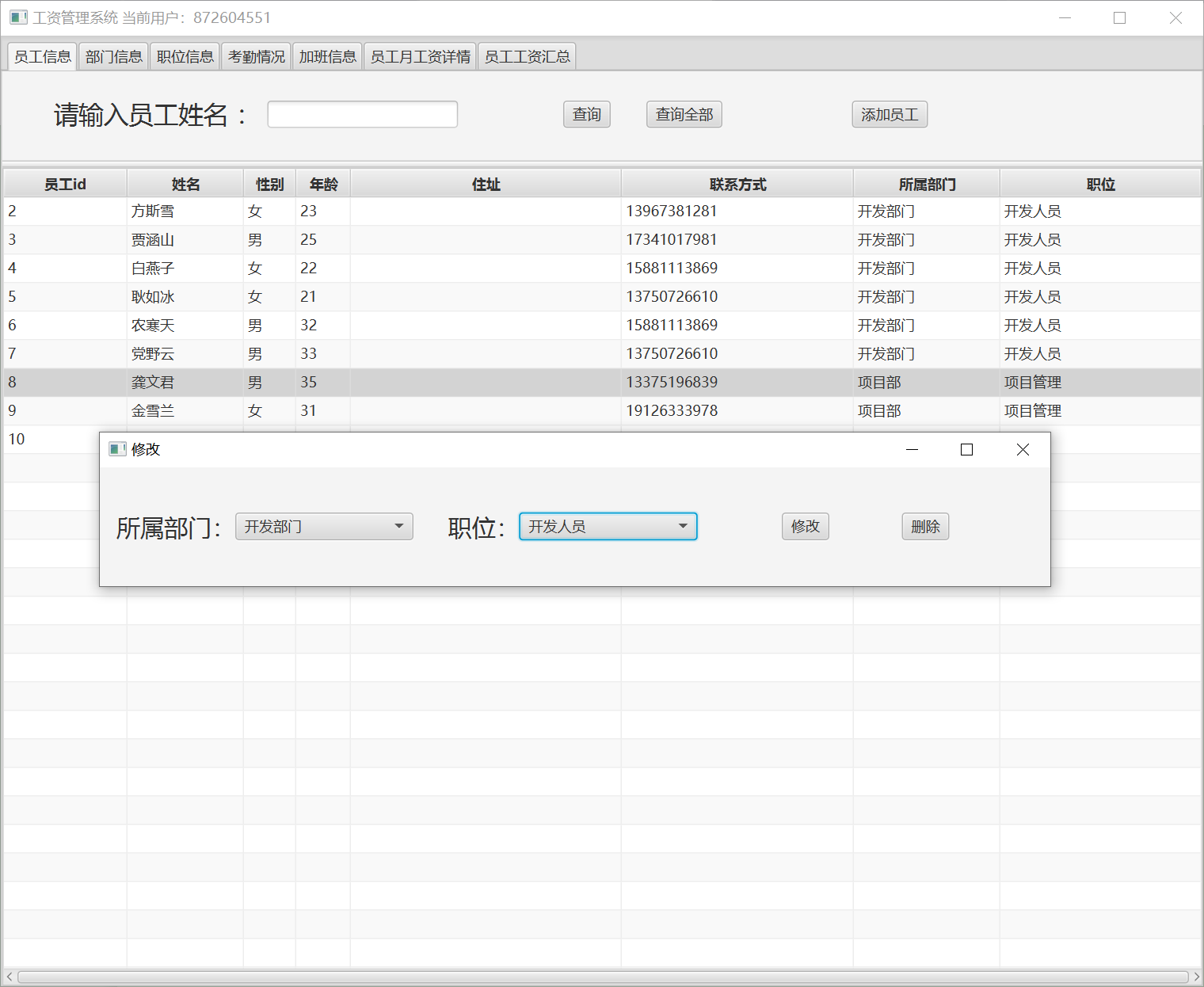


图4-27 修改的设定

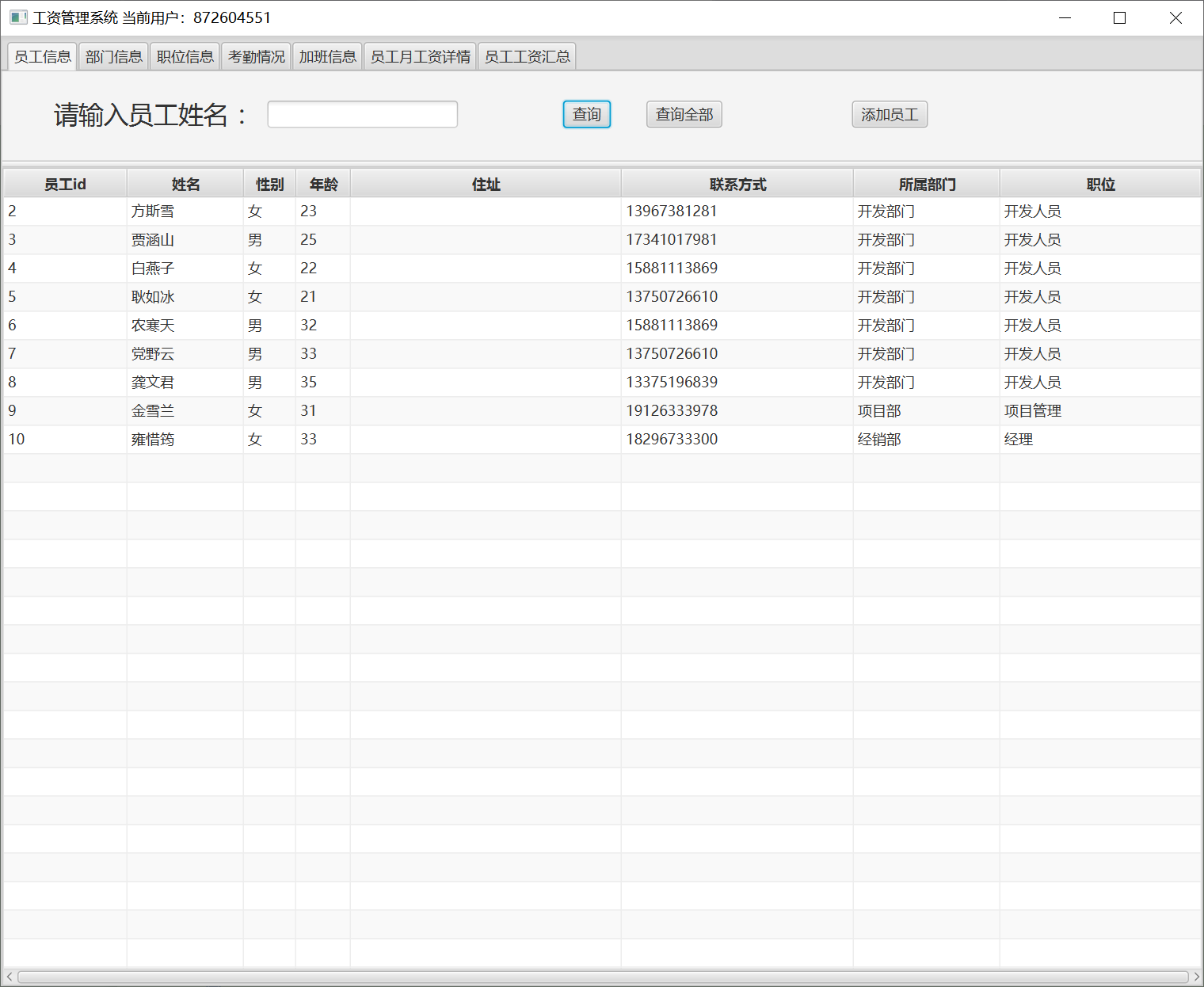


图4-28 修改后的表

12.管理员删除员工

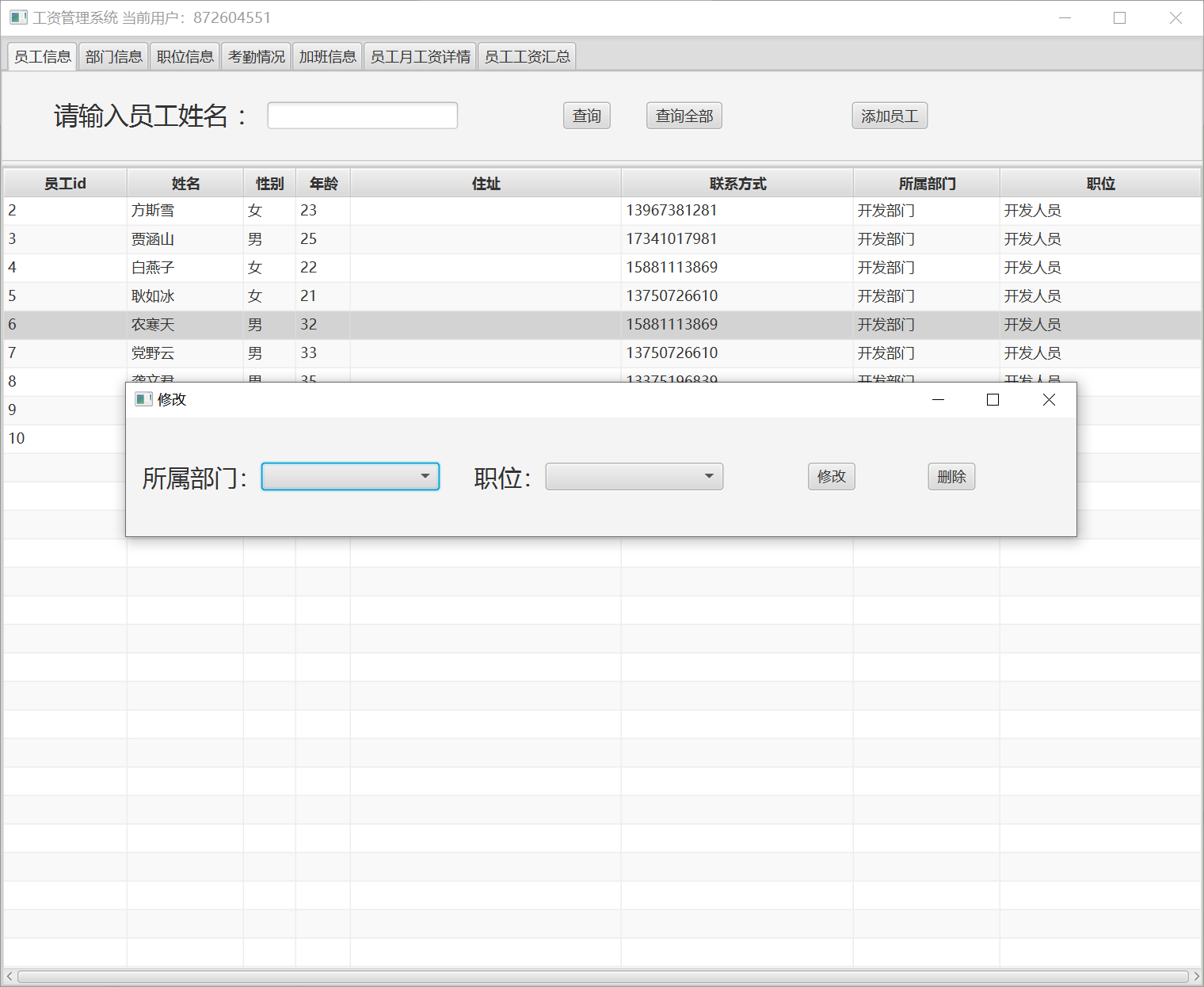


图4-29 删除的人员

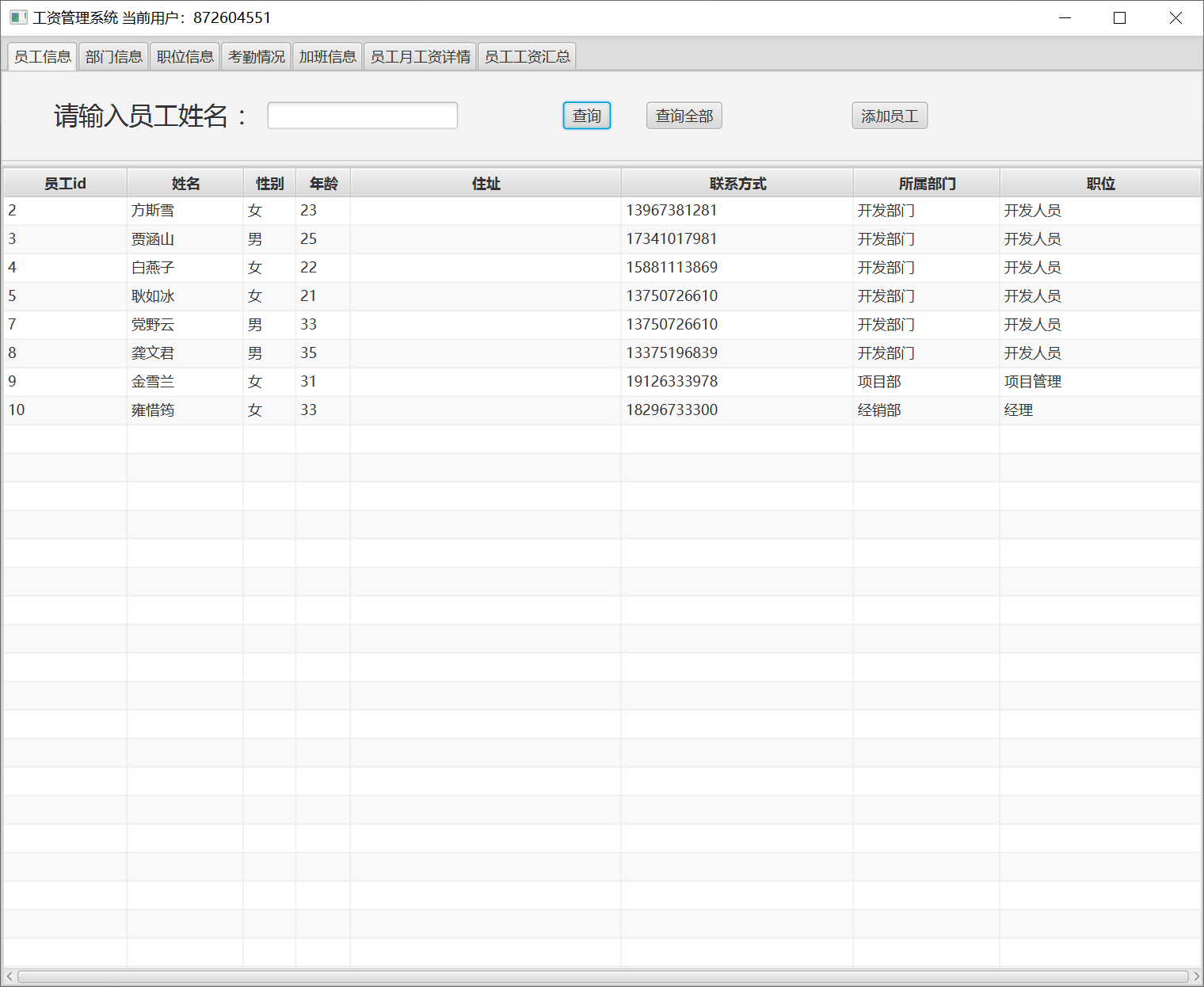


图4-30 删除后的表

13.管理员添加员工

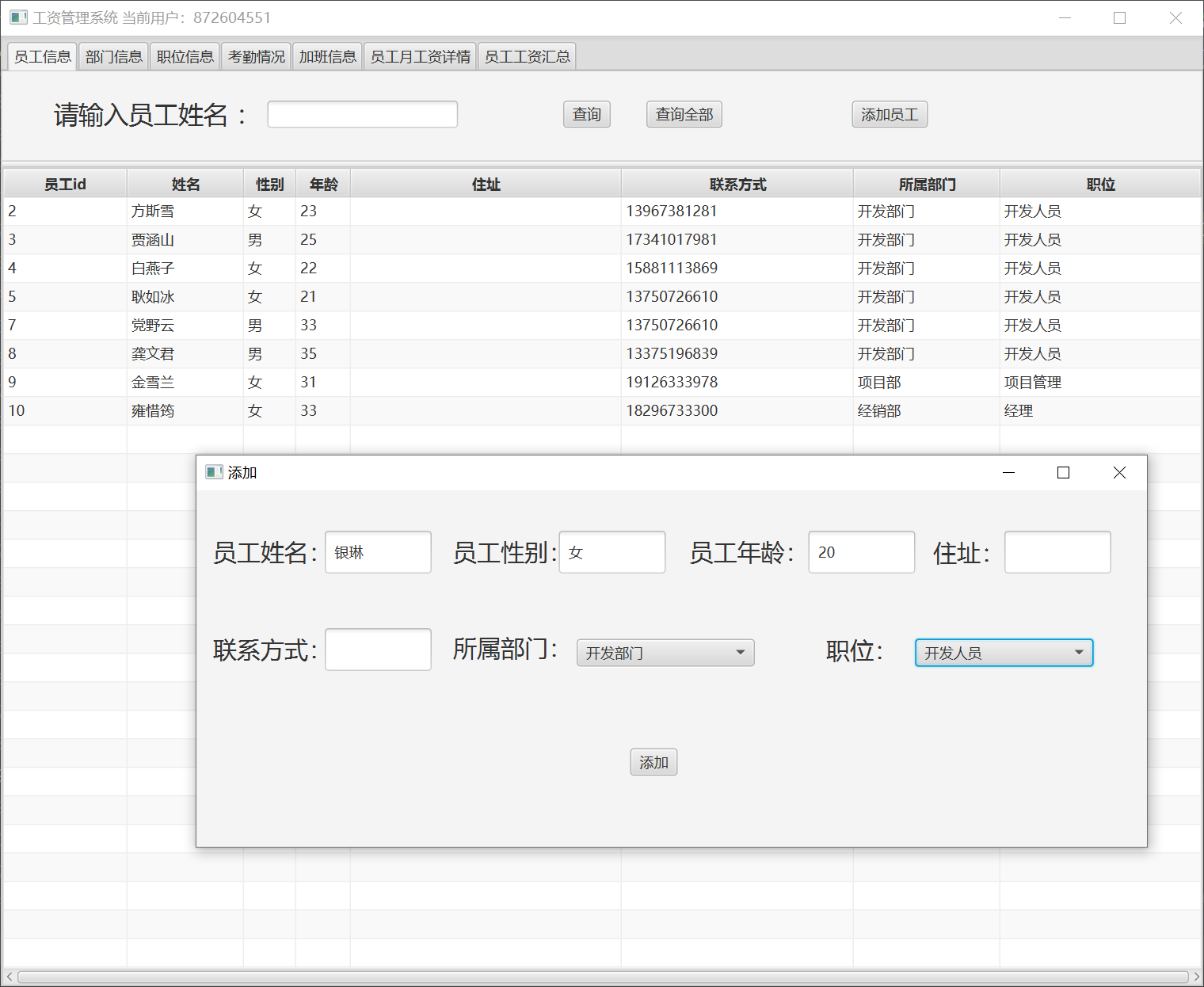


图4-31 添加的员工

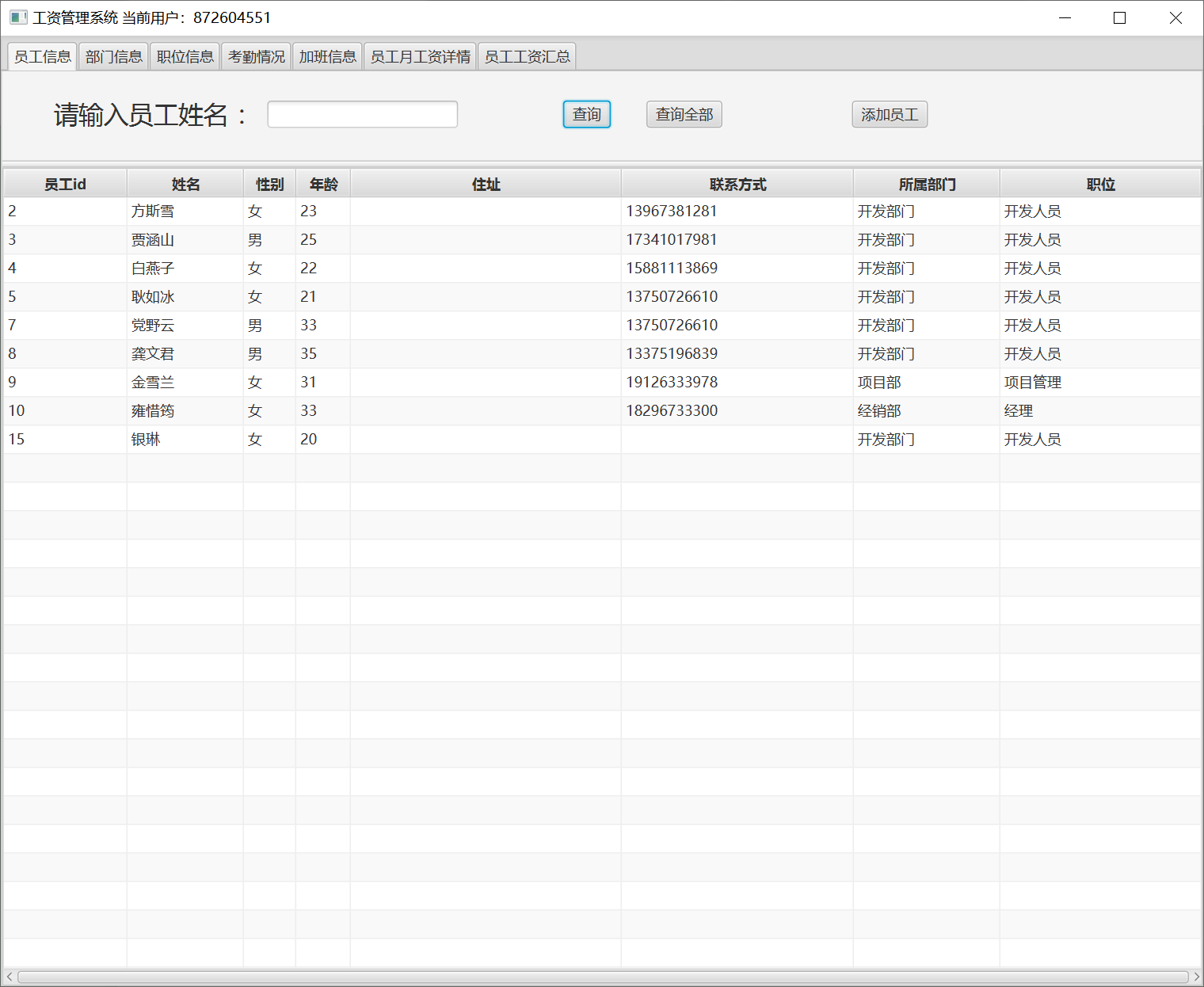


图4-32 添加后的员工表

14.管理员查询部门信息

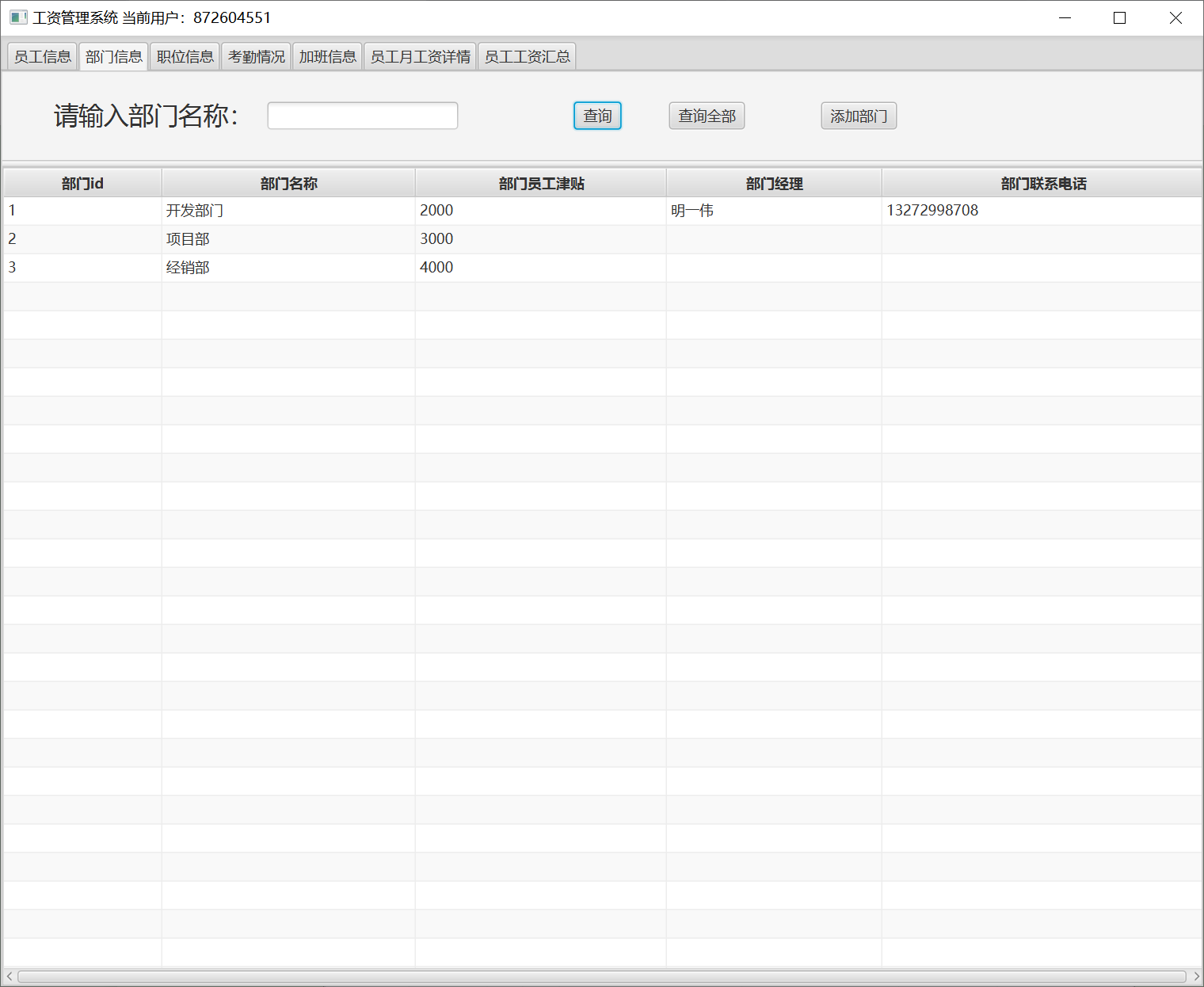


图4-33 管理员查询部门信息

15.管理员修改部门信息

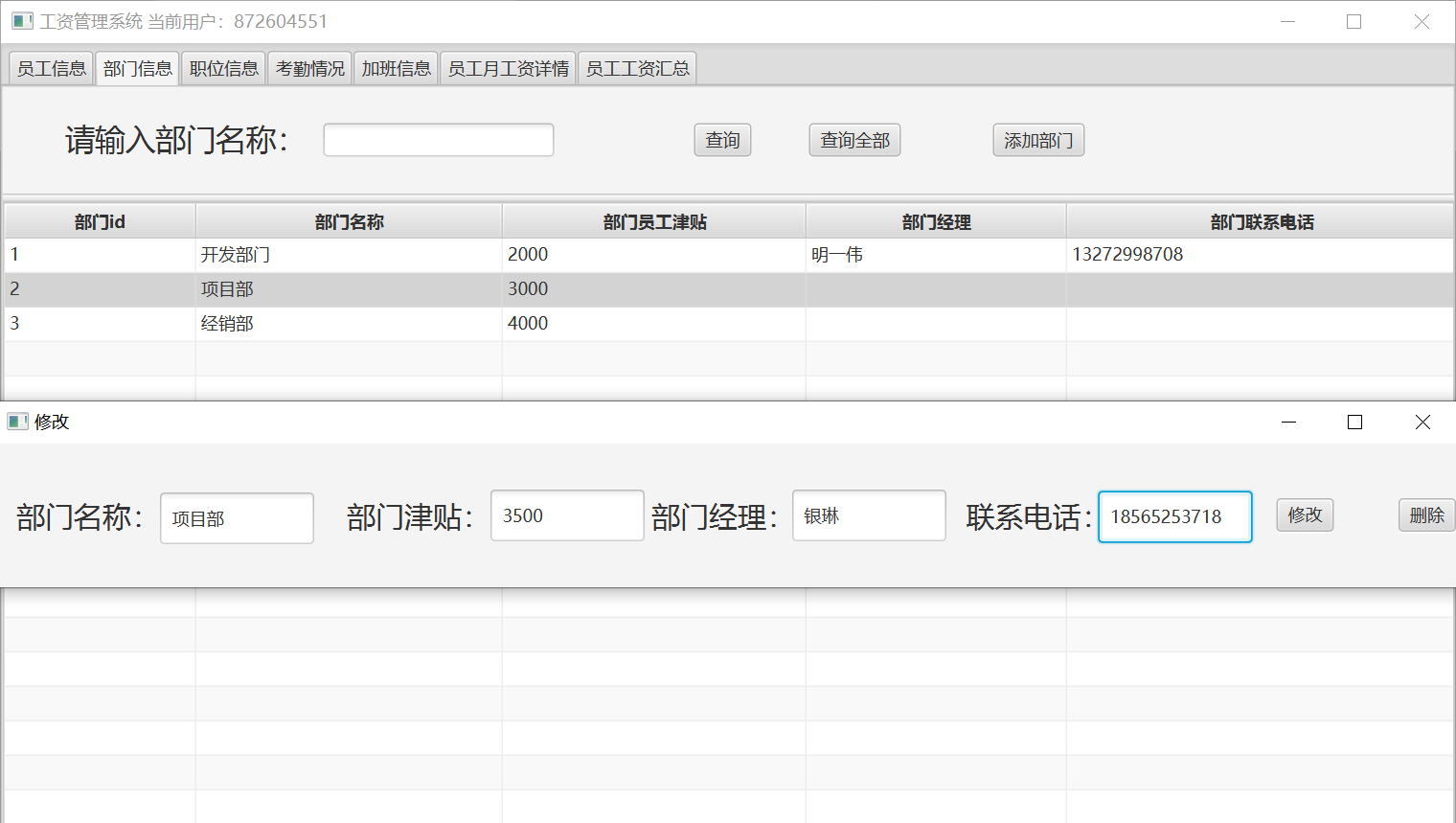


图4-34 管理员修改部门信息



图4-35 修改后的表

16.管理员添加部门信息



图4-36 添加的部门信息



图4-37 添加后的部门表

17.管理员删除部门信息



图4-38 删除摸鱼部前



图4-39 删除后的部门表

18.管理员查询职位信息



图4-40 管理员查询职位信息

19.管理员修改职位信息



图4-41 修改的信息



图4-42 修改后的职位表

20.管理员添加职位信息



图4-43 添加的职位信息



图4-44 添加后的职位表

21.管理员删除职位信息



图4-45 选中的要删除的信息



图4-46 删除后的职位表

22.管理员查看考勤情况



图4-47 管理员查看的考勤情况

23.管理员添加考勤记录



图4-48 添加考勤的记录



图4-49 添加后的考勤表

24.管理员修改考勤记录



图4-50 修改的考勤信息



图4-51 修改后的考勤表

25.管理员删除考勤记录



图4-52 删除的考勤信息



图4-53 删除后的考勤表

26.管理员查看加班信息



图4-54 管理员查看的加班表

27.管理员添加加班记录



图4-55 添加的加班记录



图4-56 添加后的加班记录表

28.管理员修改加班记录



图4-57 修改的加班记录



图4-58 修改后的加班记录表

29.管理员删除加班信息



图4-59 删除的加班记录



图4-60 删除后的加班记录表

30.管理员查看员工月工资详情



图4-61 管理员查看员工月工资详情

31.管理员查看员工工资汇总信息



图4-62 管理员查看员工工资汇总信息

## 4.7 系统设计与实现总结

1.数据库设计

根据给定的要求，设计数据库表，并画出ER图，数据流图等，然后将设计好的表在mysql中建立，建立好完整性约束。

2.系统架构设计

根据操作流程，规划好相关操作模块，根据相关的功能去完成模块代码的设计和编写。

3.界面设计

使用SceneBuilder，根据相关的功能设计前端界面，使得前端界面能够完成给定的相关功能操作，最终得到相关的fxml文件。

4.后端开发

在java中使用mysql相关api对数据库进行相关的增删改查操作，实现DBMS的核心操作，完善系统的功能。

5.测试以及debug、完善系统功能

将数据导入数据库中，并模仿用户使用的情节，找出系统的bug和不足之处，进行改善。

# 5 实验总结

这是我目前做过的工程量最大的一次实验，首先是学习数据库软件的使用方法，我选择的是navicat，由于是在虚拟机中建立的数据库，而navicat我也是放在虚拟机中，因此连接数据库没有任何的问题，但是之后在设计做数据库应用系统设计的时候，想在IDEA中连接数据库并进行相关操作，但是始终无法连接到虚拟机中的数据库，查了好多资料才知道是由于网络端口映射的时候，并没有开放3306端口，导致IDEA始终连接不上，最后开放了3306端口，从虚拟机外也能连接到数据库了。其次在Educoder上的相关建表、修改、删除、添加、查询等操作也让我熟悉了mysql语句的写法，为之后的数据库应用系统设计打下了基础。

这次的实验大头就是数据库应用系统设计，需要采取C\S模式或者B\S模式的方式进行验收，但是我之前没有接触过前端设计，因此只得当场学习前端的设计。由于这学期刚学过java，因此选择使用javaFX来进行前端设计，同时使用SceneBuilder来辅助设计界面，虽然界面不够美观，但是还算是能够完成相关的功能任务。先在数据库中建好了相关的表并插入了一些数据，然后根据要求的功能，设计前端界面，并在后端进行相关的数据处理。由于是第一次进行前端设计，也遇到了很多的问题，在网上查阅了许多的资料，这些问题得以解决。此次的实验从头到尾全部由我自己设计，没有参考其他人的相关模板，极大的锻炼了我相关的能力，当这个工程完成时，我自己也有很大的成就感。

在这次实验中，我学会了前端的设计，同时也学会了数据库的相关操作，让我受益匪浅，虽然付出的时间很多，但是我的收获也很多，这一切都是值得的。