数据库课程设计 B

实习报告

姓名：常文瀚

学号：20181001095

班级：191181

指导老师：罗晖

中国地质大学（武汉）

2020 年 6 月 15 日

目　录

一. 需求分析

1. 所选题目：

户口管理系统

2. 功能需求：

（1）对人口信息的添加、删除、修改、查询四种功能；

（2）在人口信息添加后，从人口信息中挑选某人进行立户，使其成为户主；

（3）可以对户主的特定信息进行添加、删除、修改、查询四种操作；

（4）在立户成功后，可以在之后添加的人员中与某一个户主建立人口关系；

（5）在关系建立成功后，可以对人口关系表中的数据进行增删改查四种操作；

（6）可以对管理员信息进行添加、删除、修改、查询四种功能；

3. 数据需求：

（1）信息添加时，性别应只能限制在“男”或“女”；

（2）在 SQL Server 中设置的日期信息均为 datetime 类型，所以日期信息应当严格遵守 xxxx-x-x 的格式；

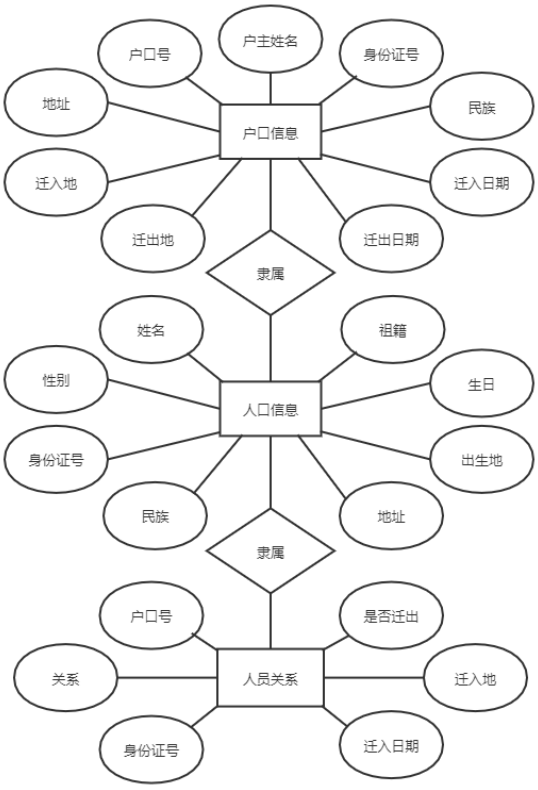
（3）人员关系应带严格遵守“兄弟”、“父子”、“母子”等该类型信息格式；

（4）身份证号、民族、户籍号等重要信息，一经录入，不得修改

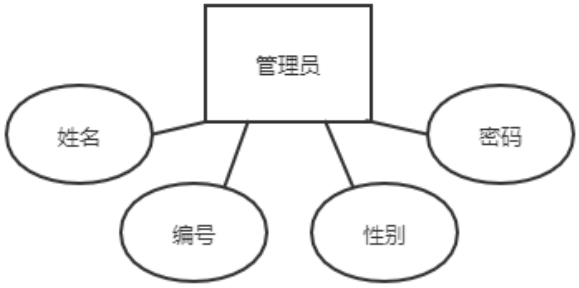
（5）非日期、非号码类型信息，必须使用中文信息录入，否则无效；

二. 概念结构设计

1. 户口信息关系 E-R 图



2.管理员 E-R 图



三.逻辑结构设计

1. 数据表逻辑结构

（1）GL（管理员信息表）

管理员信息表用于保存管理员的信息，如表 1 所示。

列名	数据类型	长度	说明	备注
Gname	char	20	管理员账户名	非空
Number	char	20	管理员密码	主码
Sex	char	2	管理员性别	
Paswd	varchar	20	管理员密码	非空

表 1 管理员信息表

（2）HK（户口信息表）

户口信息表用于保存户主信息，如表 2 所示

列名	数据类型	长度	说明	备注
Hno	char	20	户籍号	主码
Hname	char	20	户主姓名	非空
ID	char	20	身份证号	非空
Nation	char	20	民族	非空
Ads	varchar	20	住址	非空
Indate	Datetime		迁入日期	非空
wherein	varchar	20	迁入地	非空
Outdata	Datatime		迁出日期	
Whereout	varchar	20	迁出地	

表 2 户口信息表

（3）RK（人口信息表）

人口信息表用于保存人口信息，如表 3 所示

列名	数据类型	长度	说明	备注
Rname	char	20	姓名	非空
Rsex	char	2	性别	非空
ID	char	20	身份证号	主码
Nation	char	20	民族	非空
Province	char	20	祖籍	非空
Birdate	Datetime		生日	非空
Birpace	varchar	20	出生地	
Ads	varchar	20	现居地	

表 3 人口信息表

（4）RIN（人口关系信息表）

人口关系信息表用于保存人口关系信息，如表 4 所示

列名	数据类型	长度	说明	备注
Hno	char	20	姓名	外码
Relation	char	20	关系	非空
ID	char	20	身份证号	外码
Indate	Datetime		迁入日期	非空
wherein	varchar	20	迁入地	非空
bollout	char	20	是否迁出	非空

表 4 人口信息表

2.样例数据

（1）GL 表样例数据

LAPTOP-NJPGO1M...ording - dbo.GL

	Gname	Number	Sex	Paswd
▶	dmin	001 ...	男	admin
	admin3 ...	003 ...	女	admin
*	NULL	NULL	NULL	NULL

（2）HK 表样例数据

LAPTOP-NJPGO1M...rding - dbo.HK

	Hno	Hname	ID	Nation	Ads	Indate	wherein	Outdata	Whereout
▶	109001	常文瀚 ...	12010900...	汉族 ...	天津市	2000-05-3...	天津	NULL	NULL
	13265498...	高天翔 ...	12010900...	汉族 ...	上海	2001-07-0...	河南	1900-01-0...	null
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

（3）RK 表样例数据

LAPTOP-NJPGO1M...ording - dbo.RK

	Rname	Rsex	ID	Nation	Province	Birdate	Birpace	Ads
▶	张亮	男	12010544...	汉族	广州	2000-09-0...	广州	天津
	张伟	女	12010876...	大和	东京都	1945-08-1...	东京都	京都
	常文瀚	男	12010900...	汉族	天津	2000-05-3...	天津	滨海新区
	高天翔	男	12010900...	汉族	焦作	2000-03-0...	焦作	焦作
	田冰	男	14785236...	汉族	广东	2000-03-0...	广州	天津
	老八	女	8888	汉族	北京	2000-03-0...	北京	东京
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

（4）RIN 表样例数据

LAPTOP-NJPGO1M...rding - dbo.RIN　LAPTOP-NJPGO1M...ording - dbo.RK

	Hno	Relation	ID	Indate	wherein	bollout
▶	109001	兄弟	12010900...	2016-08-0...	北京	是
	12010900...	兄弟	14785236...	2003-03-0...	广州	否
	12010900...	兄弟	8888	2000-03-0...	北京	否
*	NULL	NULL	NULL	NULL	NULL	NULL

四. 系统设计

1. 环境设置

1.1 开发平台软硬件环境

（1）开发语言：C++、SQL；

（2）数据库：SQL Server；

（3）开发平台：Qt(MinGW)、SSMS 18；

（4）数据库与平台连接方式：ODBC；

（5）开发系统：Windows 10；

1.2 ODBC 数据配置

运行本实例需要配置用户 DSN,在 ODBC 数据源管理器中配置用户 DSN 连接指定的 SQL Server 数据库。关键操作步骤如下:

(1)单击“开始”按钮，选择“程序”→“管理工具”→“数据源(ODBC)”命令，打开“ODBC 数据源管理器”对话框，打开“用户 DSN”选项卡。

(2)单击“添加”按钮，打开“创建新数据源”对话框,选择安装数据源的驱动程序，这里选择“SQL Server”。

(3)单击“完成”按钮，打开“创建到 SQL Server 的新数据源”对话框，在“名称”文本框中设置数据源名称为“recording" ,在“描述”文本框设置数据源描述为“配置 SQL Server 数据库 DSN" ,在“服务器”下拉列表框中选择连接的数据库所在服务器为“LAPTOP-NJPGO1MV”。

(4)单击“下一步”按钮，选择“使用用户输入登录 ID 和密码的 SQL Server 验证”单选框，在“登录 ID”文本框中输入 SQL Server 用户登录 D,这里为“Sa”,在“密码”输入 cwh745115。

(5)单击“下一步”按钮，选择“更改默认的数据库为”复选框，并在其下拉列表框中选择连接的数据库名称,这里选择“recording" (与自己上交的数据库名-致) ,单击“下一步”按钮。

(6)单击“完成”按钮，打开“ODBC Microsoft SQL Server 安装”对话框，显示新创建的 ODBC 数据源配置信息。

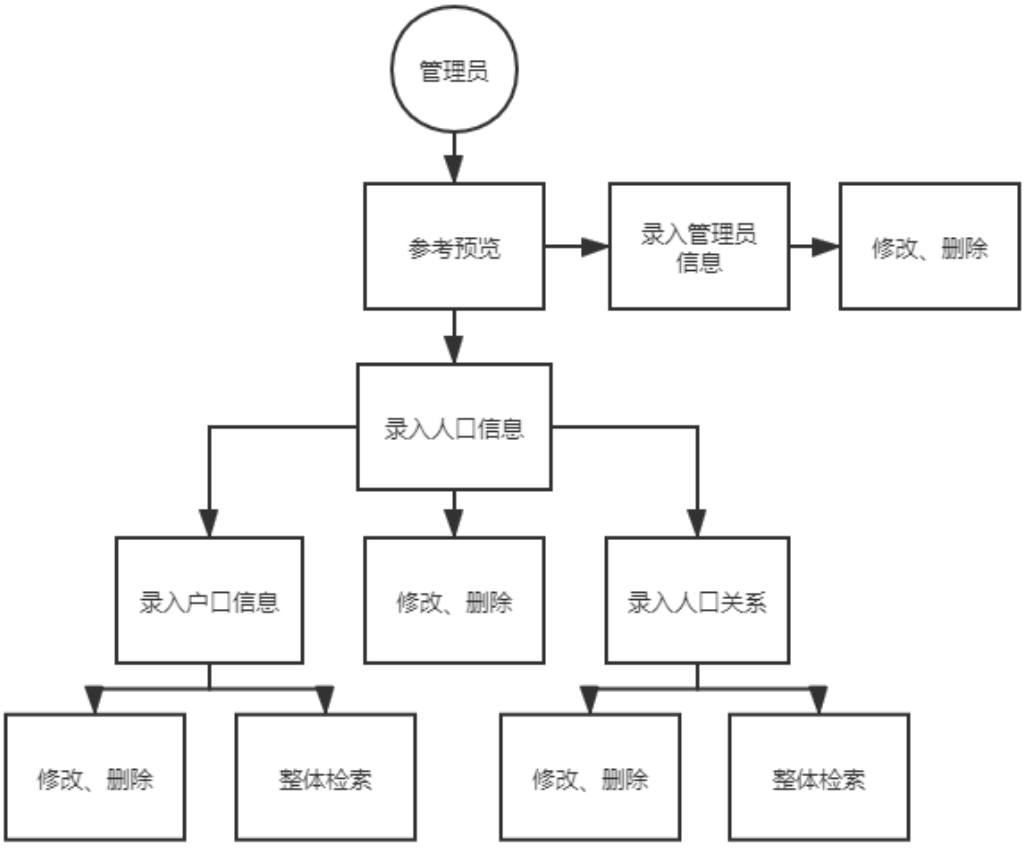
(7)单击“测试数据源”按钮测试数据库连接是否成功，如果测试成功，单击“确定”按钮，完成数据源配置。上述红色字需要与自己代码中(如代码截图)一致。注意大小写的区别。

```
bool OpenDatabase()
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QODBC");    //数据库驱动类型为SQL Server
    qDebug()<<"ODBC driver?"<<db.isValid();
    QString dsn = QString::fromLocal8Bit("QTDSN");       //数据源名称
    db.setHostName("localhost");                         //选择本地主机，127.0.1.1
    db.setDatabaseName(dsn);                             //设置数据源名称
    db.setUserName("sa");                                //登录用户
    db.setPassword("cwh745115");                         //密码
    if(!db.open())                                       //打开数据库
    {
        qDebug()<<db.lastError().text();
        QMessageBox::critical(0, QObject::tr("Database error"), db.lastError().text());
        return false;                                    //打开失败
    }
    else
    {
        qDebug()<<"database open success!";
        QSqlQuery query(db); //查询Card表并输出，测试能否正常操作数据库
    }return true;
}
```

2. 系统功能

2.1 系统模块图

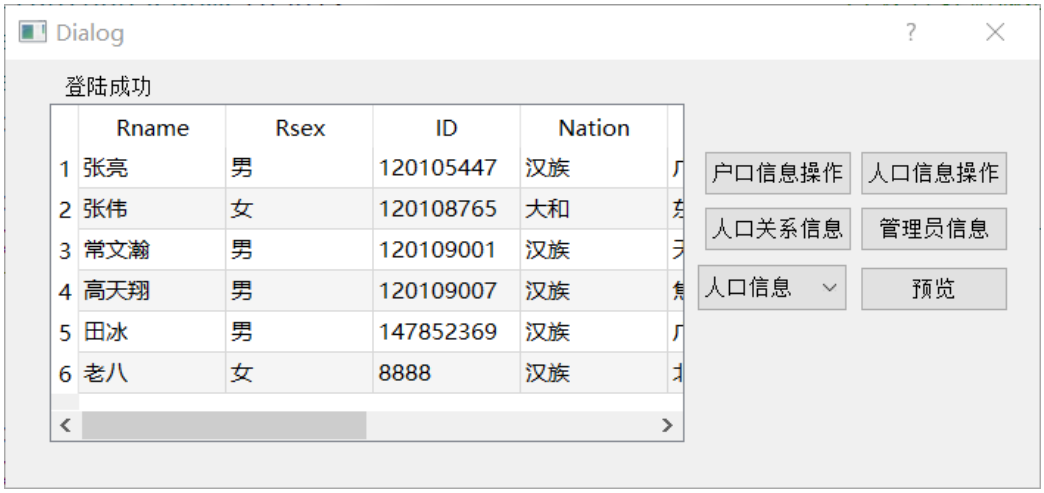
2.2 业务流程



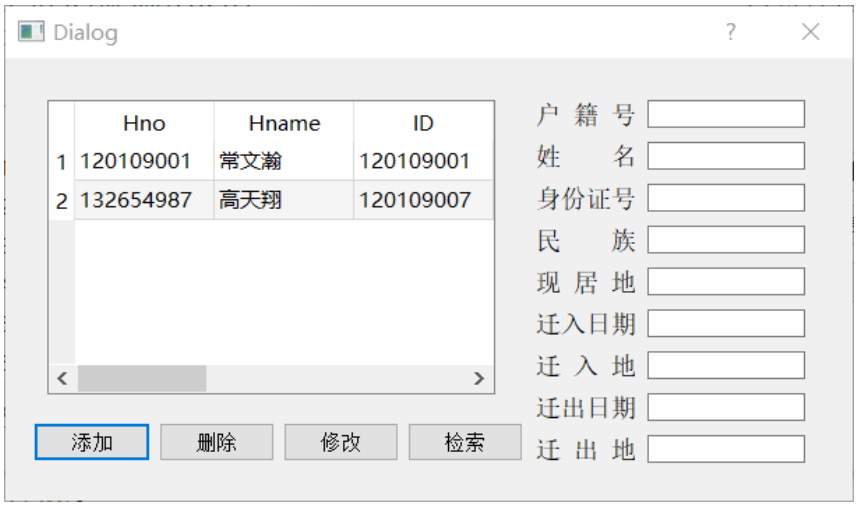
2.3 系统模块功能简介

(1)登录模块：管理员可以登录进行操作

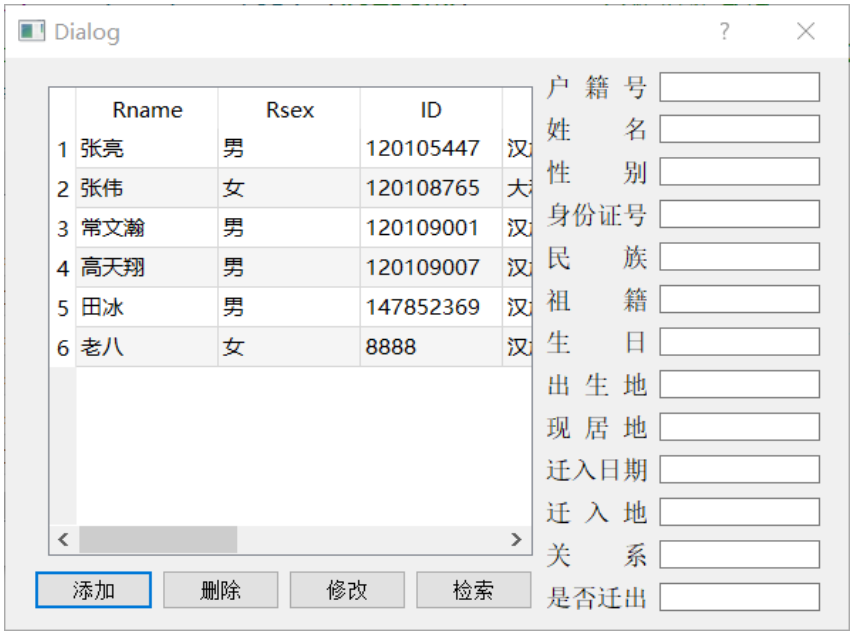
（2）预览界面：可以进行信息预览，但是不能对其操作



（3）户口信息操作模块：对户口信息进行增删改查操作，可以预览

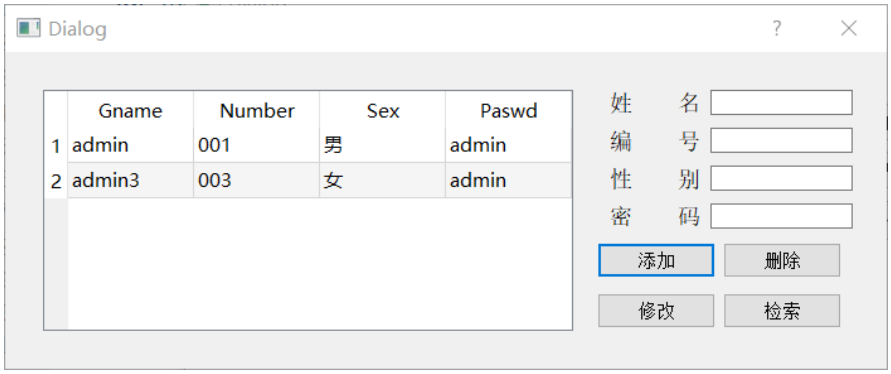


（4）人口信息操作模块：对人口信息进行增删改查操作，可以预览



（5）人口关系操作模块：对人口关系信息进行增删改查操作，可以预览

（6）管理员信息操作模块：对管理员信息进行增删改查的操作



3. 模块设计

3.1 公共类设计

（1）建立 MainWindow 类，在 MainWindow 类中定义 mainfunc 对象以实现界面跳转，代码如下：

```
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void login();
private slots:
```

```
    void on_login_clicked();
    void on_exit_clicked();
private:
    Ui::MainWindow *ui;
    MainFunction mainfunc;
};
```

（2）建立 hkctrl 类，以实现对户口信息的操作，代码如下：

```
class HKctrl : public QDialog
{
    Q_OBJECT
public:
    explicit HKctrl(QWidget *parent = nullptr);
    ~HKctrl();
    void search();      //检索显示所有信息
    void clearData();   //处理 SQL 自动修改空日期为 1900 年的情况
    void addInfo();     //添加人员
    void judged();      //判断删除
    void remove();      //删除函数
    void change();      //修改功能
private slots:
    void on_search_clicked();
    void on_add_clicked();
    void on_delete_2_clicked();
    void on_change_clicked();
private:
    Ui::HKctrl *ui;
};
```

（3）建立 rkctrl 类，实现对人口信息的操作，代码如下：

```
class RKctrl : public QDialog
{
    Q_OBJECT
public:
    explicit RKctrl(QWidget *parent = nullptr);
    ~RKctrl();
```

```
    void addInfo(); //添加信息
    void search();  //检索遍历
    void judged();  //判断 ID 是否重复后添加信息
    void remove();  //删除信息
    void change();  //修改信息
    void addInToRin(); //向户籍信息中同步添加信息
private slots:
    void on_add_clicked();
    void on_change_clicked();
    void on_search_clicked();
    void on_delete_2_clicked();
private:
    Ui::RKctrl *ui;
};
```

（4）建立 rinctrl 类，实现对人口信息的操作，代码如下：

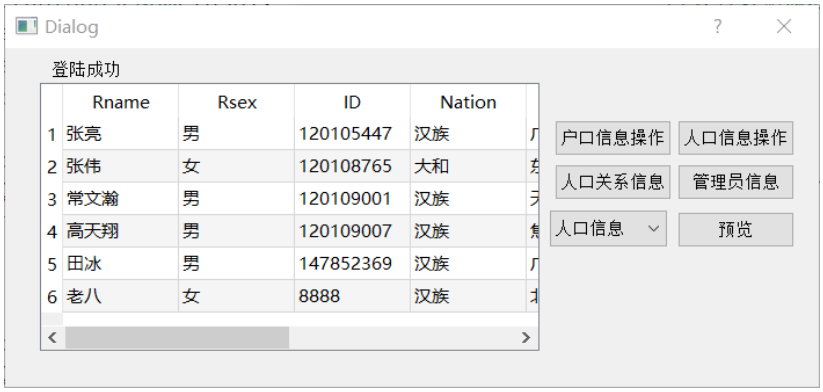
```
class RINctrl : public QDialog
{
    Q_OBJECT
public:
    explicit RINctrl(QWidget *parent = nullptr);
    ~RINctrl();
    void addInfo();
    void search();
    void change();
    void deleteInfo();
    void judge();
    bool judgeFK();
private slots:
    void on_add_clicked();
    void on_delete_2_clicked();
    void on_change_clicked();
    void on_search_clicked();
private:
    Ui::RINctrl *ui;
};
```

（5）建立 glctrl 类，实现对管理员信息的操作，代码如下：

```
class GLctrl : public QDialog
{
    Q_OBJECT
public:
    explicit GLctrl(QWidget *parent = nullptr);
    ~GLctrl();
    void search();      //检索显示所有信息
    void addInfo();     //添加信息
    void judge();       //判断是否加入
    void deleteInfo();  //删除信息
    void change();      //修改信息
private slots:
    void on_add_clicked();
    void on_delete_2_clicked();
    void on_change_clicked();
    void on_search_clicked();
private:
    Ui::GLctrl *ui;
};
```

3.2 主窗体设计

（1）主窗体截图：



（2）窗口及菜单设计过程：

首先，主菜单窗口需要有让管理员有预览功能，这样如果只是检索信息可以更加方便快捷。其次，我将主要菜单需要的按钮放在右侧，将可选区域缩小，方便操作者选择选项。最后，我选择在表格中加入隔行变色的效果，使操作者可以更好的区分每条信息，并且在查询信息拖动滑块时不会串行。需要强调的是，在这里的检索功能中，我选择的是 comboBox 控件，这样无论是空间上，还是对选项的约束条件都起到了很好地兼容效果。

3.3 登录模块设计

（1）登录模块技术分析

本系统使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 MainWindow 类连接 ODBC 数据源中 recording 的数据表 GL，以方便操作数据表。

（2）登录模块实现过程

本模块使用的数据表为 GL。

表 1　登录模块控件属性设置

控件 ID	控件属性	对应变量
LineEdit	登录账号	LineEdit
LineEdit	登录密码	LineEdit_2
Push Button	登录按钮	login
Push Button	退出按钮	exit

① 选择 Qt 默认的 MainWindow 所属的 ui 界面

② 选择默认的 MainWindow 类

③ 在 MainWindow 中添加 login 函数，为了管理员登录，代码如下：

```
void MainWindow::login()
{
    //获取账户和密码
    QString account = this->ui->lineEdit->text();
    QString password = this->ui->lineEdit_2->text();
    qDebug()<<"名字="<<account;
    qDebug()<<"密码="<<password;
    QSqlTableModel model;   //实例化 model
    model.setTable("GL");   //设置所需要操作的表格
    model.setFilter(tr("Gname = '%1' and paswd =
'%2'").arg(account).arg(password));
    model.select();
    if(model.rowCount()==1){
        mainfunc.exec();   //如果存在则跳转至操作界面
        this->hide();
    }
    else{
        QMessageBox::warning(this, tr("warn"), tr("登陆失败"));
    }
}
```

④ 右键点击控件 login 添加槽函数，为了完成登录功能。代码如下：

```
void MainWindow::on_login_clicked()
{
    login();
}
```

⑤ 右键点击控件 exit 添加槽函数，为了完成退出功能。代码如下：

```
void MainWindow::on_exit_clicked()
{
    this->close();
}
```

3.4 主菜单模块设计

（1）主菜单模块技术分析

本模块使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 mainfunction 类连接 ODBC 数据源中 recording 的数据表 GL,HK,RK,RIN，以方便操作数据表，实现检索预览四个表中所有数据的功能。

（2）主菜单模块实现过程

本模块使用的数据表为 GL,HK,RK,RIN。

表 2　主菜单模块控件属性设置

控件 ID	控件属性	对应变量
tableView	显示信息	tableView
comboBox	表的选项	tableName
Push Button	户口操作按钮	HK
Push Button	人口操作按钮	RK
Push Button	人口信息操作按钮	RIN
Push Button	管理员操作按钮	GL
Push Button	预览信息按钮	search

① 选择 Qt 默认的 mainfunction 所属的 ui 界面

② 选择默认的 mainfunction 类

③ 在 mainfunction 中添加 search 函数，为了预览数据库信息，代码如下：

```
void MainFunction::search(){
    QSqlQueryModel *model = new QSqlQueryModel;
    if(this->ui->tableName->currentIndex()==0){
        model->setQuery("select * from HK");//这里直接设置 SQL 语句，忽略最后一个参数
        ui->tableView->setModel(model);
```

```
    }
    else if (this->ui->tableName->currentIndex()==1) {
        model->setQuery("select * from RK");
        ui->tableView->setModel(model);
    }
    else if (this->ui->tableName->currentIndex()==2) {
        model->setQuery("select * from GL");
        ui->tableView->setModel(model);
    }
    else if (this->ui->tableName->currentIndex()==3) {
        model->setQuery("select * from RIN");
        ui->tableView->setModel(model);
    }
    //以下是视觉方面的效果，不加也没影响
    //隔行变色
    ui->tableView->setAlternatingRowColors(true);
    int row_count = model->rowCount();
    for(int i =0; i < row_count; i++)
    {
        ui->tableView->setRowHeight(i, 20);
    }
}
```

④ 右键点击控件 HK 添加槽函数，为了完成户口编辑页面跳转功能。代码如下：

```
void MainFunction::on_HK_clicked()
{
    hkctrl.exec();
}
```

⑤ 右键点击控件 RK 添加槽函数，为了完成人员信息编辑页面跳转功能。代码如下：

```
void MainFunction::on_RK_clicked()
{
    rkctrl.exec();
}
```

⑥ 右键点击控件 RIN 添加槽函数，为了完成人员关系编辑页面跳转功能。代码如下：

```
void MainFunction::on_RIN_clicked()
{
```

```
    rinctrl.exec();
}
```

⑦ 右键点击控件 GL 添加槽函数，为了完成管理员信息编辑页面跳转功能。代码如下：

```
void MainFunction::on_GL_clicked()
{
    glctrl.exec();
}
```

3.5 户口信息模块设计

（1）户口信息模块技术分析

本模块使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 hkctrl 类连接 ODBC 数据源中 recording 的数据表 HK,以方便操作数据表，实现对 HK 表的增删改查四种功能。

（2）户口信息模块实现过程

本模块使用的数据表为 HK。

表 3　户口信息模块控件属性设置

控件 ID	控件属性	对应变量
Push Button	添加按钮	Add
Push Button	删除按钮	Delete_2
Push Button	修改按钮	Change
Push Button	检索按钮	search
tableView	显示信息	tableView
LineEdit	获取户籍号	number
LineEdit	获取姓名	name
LineEdit	获取身份证号	ID
LineEdit	获取民族	nation
LineEdit	获取住址	ads
LineEdit	获取迁入日期	indate
LineEdit	获取迁入地	wherein
LineEdit	获取迁出日期	outdate
LineEdit	获取迁出地	whereout

① 选择 HK 所属的 ui 界面，添加控件

② 编写添加信息函数 addInfo 与 judged，实现添加信息功能，代码如下：

```
void HKctrl::addInfo()
{
    QSqlQuery q;
```

```
    q.prepare("insert into HK values (?,?,?,?,?,?,?,?,?)");
    q.addBindValue(this->ui->number->text());
    q.addBindValue(this->ui->name->text());
    q.addBindValue(this->ui->id->text());
    q.addBindValue(this->ui->nation->text());
    q.addBindValue(this->ui->ads->text());
    q.addBindValue(this->ui->indate->text());
    q.addBindValue(this->ui->wherein->text());
    q.addBindValue(this->ui->outdate->text());
    q.addBindValue(this->ui->whereout->text());
    q.exec();
}

void HKctrl::judged()
{
    QSqlTableModel model;   //实例化 model
    model.setTable("RK");   //设置所需要操作的表格
    model.setFilter(tr("ID = '%1'").arg(ui->id->text()));
    model.select();
    if(model.rowCount()==1){
        if(ui->outdate->text().length()>0){
            addInfo();
        }
        else if (ui->outdate->text().length()==0) {
            addInfo();
            clearData();
        }
    }
    else{
        QMessageBox::warning(this,tr("警告"),tr("当前 ID 使用者不存在人口信息表，故不许立户"),QMessageBox::Yes);   //如果存在则跳转至操作界面
    }
    search();
}
```

③ 编写删除信息的函数 remove()，代码如下

```
void HKctrl::remove()
{
    int curRow = ui->tableView->currentIndex().row();   //设置行号，准备数据
    QModelIndex index = ui->tableView->currentIndex();
    QString id = index.sibling(curRow,2).data().toString();    //删除 data
    QSqlQuery q;
    int ok = QMessageBox::warning(this,tr("删除该户主信息!"),tr("你确定删除当前行吗？"),QMessageBox::Yes,QMessageBox::No);
    if(ok == QMessageBox::Yes){
        q.prepare("delete from HK where id = (?)");
        q.addBindValue(id);
        q.exec();
    }
    search();
}
```

④编写修改信息函数 change()，代码如下

```
void HKctrl::change()
{
    QSqlQuery q;
    QModelIndex index = ui->tableView->currentIndex();
    int curRow = ui->tableView->currentIndex().row();
    int curColume = ui->tableView->currentIndex().column();
    qDebug()<<curRow<<curColume<<endl;
    if(curColume==0){
        QMessageBox::warning(this,tr("警告"),tr("户籍号禁止修改"),QMessageBox::Yes);
    }
    else if (curColume==1) {
        q.prepare("UPDATE HK SET Hname = (?) where Hname = (?) and ID = ?");
        q.addBindValue(ui->name->text());
        q.addBindValue(index.sibling(curRow,1).data().toString());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
```

```
    }
    else if (curColume==4) {
        q.prepare("UPDATE HK SET Ads = ? where ID = ?");
        q.addBindValue(ui->ads->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==5) {
        q.prepare("UPDATE HK SET Indate = (?) where ID = ?");
        q.addBindValue(ui->indate->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==6) {
        q.prepare("UPDATE HK SET wherein = ? where ID = ?");
        q.addBindValue(ui->wherein->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==7) {
        q.prepare("UPDATE HK SET Outdata = ? where ID = ?");
        q.addBindValue(ui->outdate->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==8) {
        q.prepare("UPDATE HK SET Whereout = ? where ID = ?");
        q.addBindValue(ui->whereout->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==2) {
        QMessageBox::warning(this,tr("警告"),tr("身份证号禁止修改
"),QMessageBox::Yes);
    }
    else if (curColume==3) {
```

```
        QMessageBox::warning(this,tr("警告"),tr("民族禁止修改
"),QMessageBox::Yes);
    }
    search();
}
```

⑤编写检索函数，实现对所有信息的检索

```
void HKctrl::search()
{
    QSqlQueryModel *model = new QSqlQueryModel;
    model->setQuery("select * from HK");
    ui->tableView->setModel(model);
    //设置效果
    ui->tableView->setAlternatingRowColors(true);
    int row_count = model->rowCount();
    for(int i =0; i < row_count; i++)
    {
        ui->tableView->setRowHeight(i, 20);
    }
}
```

⑥ 为所有按钮控件添加槽函数，以实现各种功能，代码如下：

```
void HKctrl::on_search_clicked()
{
    search();
}
void HKctrl::on_add_clicked()
{
    judged();
}
void HKctrl::on_delete_2_clicked()
{
    remove();
}
void HKctrl::on_change_clicked()
{
    change();
}
```

3.6 人口信息模块设计

（1）人口信息模块技术分析

本模块使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 rkctrl 类连接 ODBC 数据源中 recording 的数据表 RK,以方便操作数据表，实现对 RK 表的增删改查四种功能。

（2）人口信息模块实现过程

本模块使用的数据表为 RK。

表 4　人口信息管理模块控件属性设置

控件 ID	控件属性	对应变量
Push Button	添加按钮	Add
Push Button	删除按钮	Delete_2
Push Button	修改按钮	Change
Push Button	检索按钮	search
tableView	显示信息	tableView
LineEdit	获取户籍号	number
LineEdit	获取姓名	name
LineEdit	获取身份证号	ID
LineEdit	获取民族	nation
LineEdit	获取住址	ads
LineEdit	获取迁入日期	indate
LineEdit	获取迁入地	wherein
LineEdit	获取迁出日期	outdate
LineEdit	获取迁出地	whereout
LineEdit	获取祖籍	Province
LineEdit	获取生日	Birdate
LineEdit	获取出生地	Birplace
LineEdit	获取关系	relation

① 选择 HK 所属的 ui 界面，添加控件

② 编写添加信息函数 addInfo 与 judged，实现添加信息功能，代码如下：

```
void RKctrl::addInfo()  //添加人口信息
{
    QSqlQuery q;
    q.prepare("insert into RK values (?,?,?,?,?,?,?,?)");
    q.addBindValue(this->ui->name->text());
```

```
    q.addBindValue(this->ui->sex->text());
    q.addBindValue(this->ui->id->text());
    q.addBindValue(this->ui->nation->text());
    q.addBindValue(this->ui->province->text());
    q.addBindValue(this->ui->birdate->text());
    q.addBindValue(this->ui->birplace->text());
    q.addBindValue(this->ui->ads->text());
    q.exec();
    search();
}
void RKctrl::judged()   //完整化，先判断再添加
{
    QSqlTableModel model;   //实例化 model
    model.setTable("RK");   //设置所需要操作的表格
    model.setFilter(tr("ID = '%1'").arg(ui->id->text()));
    model.select();
    if(model.rowCount()==1){
        QMessageBox::warning(this,tr("警告"),tr("当前 ID 使用者已存在
"),QMessageBox::Yes);   //如果存在则跳转至操作界面
    }
    else{
        addInfo();
        addInToRin();
    }
}
```

③编写删除信息函数，实现删除某人员信息的功能

```
void RKctrl::remove()
{
    int curRow = ui->tableView->currentIndex().row();   //设置行号，准备数
据
    QModelIndex index = ui->tableView->currentIndex();
    QString id = index.sibling(curRow,2).data().toString();    //删除 data
    QSqlQuery q;
    int ok = QMessageBox::warning(this,tr("删除该人信息!"),tr("你确定删除当
前行吗？"),QMessageBox::Yes,QMessageBox::No);
    if(ok == QMessageBox::Yes){
```

```
        q.prepare("delete from RIN where id = (?)");
        q.addBindValue(id);
        q.exec();
        q.prepare("delete from RK where id = (?)");
        q.addBindValue(id);
        q.exec();
    }
    search();
}
```

④ 编写修改信息函数，实现对信息的修改功能

```
void RKctrl::change()
{
    QSqlQuery q;
    QModelIndex index = ui->tableView->currentIndex();
    int curRow = ui->tableView->currentIndex().row();
    int curColume = ui->tableView->currentIndex().column();
    if(curColume==0){
        q.prepare("UPDATE RK SET Rname = (?) where Rname = (?) and ID
= ?");
        q.addBindValue(ui->name->text());
        q.addBindValue(index.sibling(curRow,0).data().toString());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==1) {
        q.prepare("UPDATE RK SET Rsex = (?) where ID = ?");
        q.addBindValue(ui->sex->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==4) {
        q.prepare("UPDATE RK SET Province = (?) where ID = ?");
        q.addBindValue(ui->province->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
```

```
    else if (curColume==5) {
        q.prepare("UPDATE RK SET Birdate = (?) where ID = ?");
        q.addBindValue(ui->birdate->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==6) {
        q.prepare("UPDATE RK SET Birpace = ? where ID = ?");
        q.addBindValue(ui->birplace->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==7) {
        q.prepare("UPDATE RK SET Ads = ? where ID = ?");
        q.addBindValue(ui->ads->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==2) {
        QMessageBox::warning(this,tr("警告"),tr("身份证号禁止修改
"),QMessageBox::Yes);
    }
    else if (curColume==3) {
        QMessageBox::warning(this,tr("警告"),tr("民族禁止修改
"),QMessageBox::Yes);
    }
    search();
    QMessageBox::warning(this,tr("警告"),tr("请确认当前信息是否需要在迁入人
员数据库中修改！"),QMessageBox::Yes);
}
```

⑤ 编写检索函数 search，实现对所有人员信息的检索功能

```
void RKctrl::search()
{
    QSqlQueryModel *model = new QSqlQueryModel;
    model->setQuery("select * from RK");
    ui->tableView->setModel(model);
```

```
    //设置效果
    ui->tableView->setAlternatingRowColors(true);
    int row_count = model->rowCount();
    for(int i =0; i < row_count; i++){
        ui->tableView->setRowHeight(i, 20);
    }
}
```

⑥ 为所有按钮控件添加槽函数，达到实现功能的效果

```
void RKctrl::on_add_clicked()
{
    judged();
}
void RKctrl::on_delete_2_clicked()
{
    remove();
}
void RKctrl::on_change_clicked()
{
    change();
}
void RKctrl::on_search_clicked()
{
    search();
}
```

3.7 人口关系模块设计

（1）人口关系模块技术分析

本模块使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 RINctrl 类连接 ODBC 数据源中 recording 的数据表 RIN,以方便操作数据表，实现对 RIN 表的增删改查四种功能。

（2）人口关系模块实现过程

本模块使用的数据表为 RIN。

表 5　人口信息管理模块控件属性设置

控件 ID	控件属性	对应变量
Push Button	添加按钮	Add
Push Button	删除按钮	Delete_2
Push Button	修改按钮	Change

Push Button	检索按钮	search
tableView	显示信息	tableView
LineEdit	获取户籍号	number
LineEdit	获取关系	relation
LineEdit	获取身份证号	ID
LineEdit	获取迁入日期	indate
LineEdit	获取迁入地	wherein
LineEdit	确认是否迁出	boolout

① 选择 RIN 所属的 ui 界面，添加控件

② 编写添加信息函数 judgeFK,addInfo 与 judged，实现外码的判断与添加信息功能，代码如下：

```
//判断是否符合外码条件
bool RINctrl::judgeFK(){
    QSqlTableModel HKmodel,RKmodel;   //实例化 model
    HKmodel.setTable("HK");   //设置所需要操作的表格
    RKmodel.setTable("RK");
    HKmodel.setFilter(tr("Hno = '%1'").arg(ui->number->text()));
    RKmodel.setFilter(tr("ID = '%1'").arg(ui->id->text()));
    HKmodel.select();
    RKmodel.select();
    if(HKmodel.rowCount()==1 && RKmodel.rowCount()==1){
        return true;
    }
    else {
        return false;
    }
}


void RINctrl::judge(){
    if(judgeFK()==true){
        addInfo();
    }
    else {
        QMessageBox::warning(this,tr("警告"),tr("户籍号或身份证号不存在，经返回预览页查询"),QMessageBox::Yes);   //如果存在则跳转至操作界面
```

```
    }
}


void RINctrl::addInfo(){
    QSqlQuery q;
    q.prepare("insert into RIN values (?,?,?,?,?,?)");
    q.addBindValue(this->ui->number->text());
    q.addBindValue(this->ui->relation->text());
    q.addBindValue(this->ui->id->text());
    q.addBindValue(this->ui->indate->text());
    q.addBindValue(this->ui->wherein->text());
    q.addBindValue(this->ui->boolout->text());
    q.exec();
    search();
}
```

③ 编写 deleteInfo 函数，以实现信息的删除功能，代码如下：

```
void RINctrl::deleteInfo(){
        int curRow = ui->tableView->currentIndex().row();   //设置行号，准备数据
        QModelIndex index = ui->tableView->currentIndex();
        QString id = index.sibling(curRow,2).data().toString();    //删除data
        QSqlQuery q;
        int ok = QMessageBox::warning(this,tr("删除该人信息!"),tr("你确定除当前行吗？"),QMessageBox::Yes,QMessageBox::No);
        if(ok == QMessageBox::Yes){
            q.prepare("delete from RIN where id = (?)");
            q.addBindValue(id);
            q.exec();
        }
        search();
}
```

④ 编写 change 函数，实现对已存在的信息的修改

```
void RINctrl::change(){
    QSqlQuery q;
    QModelIndex index = ui->tableView->currentIndex();
```

```
    int curRow = ui->tableView->currentIndex().row();
    int curColume = ui->tableView->currentIndex().column();
    if(curColume==0){
        QMessageBox::warning(this,tr("警告"),tr("户籍号禁止修改
"),QMessageBox::Yes);
    }
    else if (curColume==1) {
        q.prepare("UPDATE RIN SET Relation = (?) where ID = ?");
        q.addBindValue(ui->relation->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==3) {
        q.prepare("UPDATE RIN SET Indate = (?) where ID = ?");
        q.addBindValue(ui->indate->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==4) {
        q.prepare("UPDATE RIN SET wherein = (?) where ID = ?");
        q.addBindValue(ui->wherein->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==5) {
        q.prepare("UPDATE RIN SET bollout = (?) where ID = ?");
        q.addBindValue(ui->boolout->text());
        q.addBindValue(index.sibling(curRow,2).data().toString());
        q.exec();
    }
    else if (curColume==2) {
        QMessageBox::warning(this,tr("警告"),tr("身份证号禁止修改
"),QMessageBox::Yes);
    }
    search();
}
```

⑤ 为控件添加槽函数，使其能够出发指定的函数以实现功能，代码如下：

```
void RINctrl::on_add_clicked()
{
    judge();
}
void RINctrl::on_delete_2_clicked()
{
    deleteInfo();
}
void RINctrl::on_change_clicked()
{
    change();
}
void RINctrl::on_search_clicked()
{
    search();
}
```

3.8 管理员信息模块设计

（1）管理员信息模块技术分析

本模块使用 ODBC 连接数据源，在进行对数据表的操作时使用基于 GLnctrl 类连接 ODBC 数据源中 recording 的数据表 GL,以方便操作数据表，实现对 GL 表的增删改查四种功能。

（2）管理员信息模块实现过程

本模块使用的数据表为 GL。

表 6　管理员信息管理模块控件属性设置

控件 ID	控件属性	对应变量
Push Button	添加按钮	add
Push Button	删除按钮	Delete_2
Push Button	修改按钮	change
Push Button	检索按钮	search
tableView	显示信息	tableView
LineEdit	获取管理员帐号	name
LineEdit	获取管理员密码	password
LineEdit	获取身份证号	id
LineEdit	获取管理员性别	sex

①编写 addInfo 函数与 judge 函数，达到添加信息的效果

```
void GLctrl::addInfo(){
    QSqlQuery q;
    q.prepare("insert into GL values (?,?,?,?)");
    q.addBindValue(this->ui->name->text());
    q.addBindValue(this->ui->id->text());
    q.addBindValue(this->ui->sex->text());
    q.addBindValue(this->ui->password->text());
    q.exec();
    search();
}


void GLctrl::judge(){
    //先判断是否输入了姓名与编号
    if(ui->id->text().length()>0 && ui->name->text().length()>0){
        QSqlTableModel model;   //实例化 model
        model.setTable("GL");   //设置所需要操作的表格
        model.setFilter(tr("Number= '%1'").arg(ui->id->text()));
        model.select();
        //判断是否重复添加编号
        if(model.rowCount()==1){
            QMessageBox::warning(this,tr("警告"),tr("当前管理员编号重复，请
重新设置"),QMessageBox::Yes);   //如果存在则跳转至操作界面
        }
        else{
            addInfo();
        }
    }
    else{
        QMessageBox::warning(this,tr("警告"),tr("姓名或管理员编号为空，请检
查后重新添加"),QMessageBox::Yes);
    }
    search();
}
```

②编写 deleteInfo 函数，达到删除信息的效果

```
void GLctrl::deleteInfo(){
    int curRow = ui->tableView->currentIndex().row();   //设置行号，准备数据
    QModelIndex index = ui->tableView->currentIndex();
    QString id = index.sibling(curRow,1).data().toString();    //删除 data
    QSqlQuery q;
    int ok = QMessageBox::warning(this,tr("删除该户主信息!"),tr("你确定删除当前行吗？"),QMessageBox::Yes,QMessageBox::No);
    if(ok == QMessageBox::Yes){
        q.prepare("delete from GL where Number = (?)");
        q.addBindValue(id);
        q.exec();
    }
    search();
}
```

③编写 change 函数，达到修改信息的效果

```
void GLctrl::change(){
    QSqlQuery q;
    QModelIndex index = ui->tableView->currentIndex();
    int curRow = ui->tableView->currentIndex().row();
    int curColume = ui->tableView->currentIndex().column();
    if(curColume==0){
        q.prepare("UPDATE GL SET Gname = (?) where Gname = (?) and Number = ?");
        q.addBindValue(ui->name->text());
        q.addBindValue(index.sibling(curRow,0).data().toString());
        q.addBindValue(index.sibling(curRow,1).data().toString());
        q.exec();
    }
    else if(curColume==1){
        QMessageBox::warning(this,tr("警告"),tr("管理员号禁止修改"),QMessageBox::Yes);
    }
    else if(curColume==2){
        q.prepare("UPDATE GL SET Sex = (?) where Number = ?");
        q.addBindValue(ui->sex->text());
```

```
        q.addBindValue(index.sibling(curRow,1).data().toString());
        q.exec();
    }
    else if(curColume==3){
        q.prepare("UPADTE GL SET Paswd = ? where Number = ?");
        q.addBindValue(ui->password->text());
        q.addBindValue(index.sibling(curRow,1).data().toString());
        q.exec();
    }

    search();
}
```

④编写 search 函数，达到检索信息的效果

```
void GLctrl::search(){

    QSqlQueryModel *model = new QSqlQueryModel;
    model->setQuery("select * from GL");
    ui->tableView->setModel(model);
    //设置效果
    ui->tableView->setAlternatingRowColors(true);
    int row_count = model->rowCount();
    for(int i =0; i < row_count; i++)
    {
        ui->tableView->setRowHeight(i, 20);
    }
}
```

⑤为按钮控件编写槽函数，使其可以触发所需要的功能

```
void GLctrl::on_add_clicked()
{
    judge();
}
void GLctrl::on_delete_2_clicked()
{
    deleteInfo();
}
```

```
void GLctrl::on_change_clicked()
{
    change();
}
void GLctrl::on_search_clicked()
{
    search();
}
```

五.实习体会

通过这次数据库课程设计，我掌握了 Qt、SSMS 软件的使用，也进一步的了解了数据库的意义，将它与现实中的问题联系在一起，能够使问题简单化，更容易操作。我这次建立的是户口管理系统，户口管理系统是-功能十分强大的管理系统，它集各种功用于一身，可以完成户籍管理中的各种操作，如对人员信息的成绩的添加或删除等。在制作的过程中，程序编写也十分复杂，工作量很大，编写一定要认真，一个小小的失误都可能使程序出现大的漏洞。

在初次接触 Qt 与 SQL Server 的连接使用时，我遇到了如何使用既存的 API 的问题，在查阅大量网络资料，并且查看 Qt 帮助手册后我成功实现了从 Qt 平台对数据库进行操作。并且快速掌握了可视化程序中对数据库的增、删、改、查四种操作，在这之后，我对编程中需要考虑到的一些约束进行了考虑，为初步实现的 demo 添加了很多信息约束条件，如录入信息的格式规范、录入信息的添加顺序、删除信息的删除信息等。

在数据库的约束这一部分，我为人口关系表添加了外码，人口关系表中户籍号需要参照户口信息表中的户籍号，身份证号需要参照人口信息表中的身份证号，也就是如果需要添加人口之间的信息，那么就必须有户籍存在且有关系的两个人也存在，这样在添加人口信息时就需要先向人口信息表添加信息，再根据既存信息添加人口关系，如下图。

```
else{
    addInfo();
    addInToRin();
}
```

图一 添加人口信息的顺序

除了添加信息时的参照约束条件，还需要注意在删除信息时的参照约束条件，简单的来说，在删除信息时，要先删除人口关系表中的信息，再删除人口信息表中的信息，因为如果人口信息已经被删除导致不存在，那么人口关系表中对应的信息是不可能存在的，所以删除时一定要注意先删除关系，再删除具体的个人信息，这样才能时刻保持人员信息表与人员关系表的同步处理。

```
q.prepare("delete from RIN where id = (?)");
q.addBindValue(id);
q.exec();
q.prepare("delete from RK where id = (?)");
q.addBindValue(id);
q.exec();
```

图二 删除人口信息的顺序