西南大学

人工智能学院智能科学大类

《程序设计综合课程设计》报告

题　　目： QQ好友管理系统软件开发设计

级、专业： 2020级智能科学与技术专业6班

学生学号： 222020335220177

学生姓名： 严中圣

指导教师：

提交日期： 2021年7月10日

目录

1 实践课程设计任务描述

1.1 实践课程设计目的

本次程序设计实践课程设计要求运用所学编程技能独立开发一个小型应用软件，目的在于锻炼自身的编程技能以及提高对于软件架构设计的理解、应用；其次能够简单掌握数据库的设计及软件界面设计等操作，熟悉软件开发从设计到架构到封装打包的过程，对于计算机学科专业学生来说无疑是一次极佳的实践机会。

1.2 实践课程设计要求

本次选题为模拟 QQ 实现好友管理系统，具体设计要求如下：

(1) 能够实现 QQ 登录系统并具有独立的登录界面；

(2) 能够实现用户通过口令登录，且密码采用 MD5 加密算法封装验证；

(3) 能够实现 QQ 好友管理系统并具有独立的系统界面；

(4) 能够实现通过鼠标触发事件进行软件操作；

(5) 能够自主设计数据库作为数据存储环境；

(6) 能够增加好友（默认对方已经同意请求）、删除好友、移动好友分组、查看好友详细信息等；

(7) 能够实现好友分组以及分组的增删、修改等；

(8) 能够修改好友列表中已有好友信息（如备注等）；

(9) 能够按照首字母或分组查找好友；

(10) 能够将好友加入黑名单，加入黑名单的好友不能再被查找，并设置黑名单人数上限。

1.3 实践课程设计背景

在互联网极为发达的今天，网络通信已经成为人们生活中不可或缺的部分，而每个成熟的聊天软件都需要一个完善的好友管理系统作为支撑。本次实践通过模拟开发 QQ 聊天软件，旨在对常见操作背后原理进行熟悉并实践开发，为以后的工程设计实践奠定实践经验。

1.4 实践课程设计作品用途

本次设计开发的模拟 QQ 软件，现有基础可支持聊天软件的基础操作，后续在配合上 TCP 网络传输协议与聊天室界面设计以后即可实现正常的网络通信，如再配合服务器的开发，即可实现多人聊天室的信息传输以及多用户的聊天平台搭建。

2 功能需求分析

2.1 后台数据库搭建环境配置

对于本次设计的好友管理系统，必须依赖后台的 MySQL 数据库进行数据存储，否则则无法形成一个完整的管理系统，故在正式进入开发前先要对项目工程进行数据库设计及配置。本次项目开发基于 QT Creator 开发环境，我们首先对 QT 进行 MySQL 环境配置。

(1) 首先在项目配置文件 QQ_model.pro 中添加“QT+=SQL”配置项；同时在 QT 中安装 MySQL 驱动 QMySQL；

(2) 其次在头文件 mainwindow.h 中建立方法 createMysqlConn()，设置好待连接数据库的数据库管理系统、主机名、接口、用户名、密码以及需要需要调用的数据库名，调用 Qsqldatabase 类函数实现数据库连接并设置好提示消息“数据库已连接”/“数据库连接失败”;

(3) 在主程序 main.cpp 中导入头文件 mainwindow.h 最先进行方法 createMysqlConn() 调用并在一次连接失败后尝试二次连接。至此数据库连接配置已完成。

2.2 用户验证登录系统实现

用户验证登录系统需要包括数据库用户信息调出验证、用户选择“登录”或“退出”按钮、登录成功或失败提示指令，登录功能实现在 logindialog.h 头文件及 logindialog.cpp 源文件中。

(1) 向 logindialog.h 中导入 MySQL 相关库，包括 QDialog、QsqlQuery、QCryptographicHash 分别用于数据库的连接、查询及 MD5 加密算法，其次声明变量与方法;

(2) 在 logindialog.cpp 中设置登录界面大小及窗口位置，其次设置数据库调用指令将存储的用户密码与用户输入密码进行验证比对，并设置提示指令弹窗;

(3) 向 main.cpp 中导入头文件“logindialog.h”，初始启动程序 main.cpp 后，数据库首先成功连接之后初始化登录界面，调用 logindialog.cpp 中的 LoginDialog() 方法，验证通过后即显示初始化好友管理界面。

2.3 好友管理系统实现

好友管理系统主体程序包括 main.cpp、mainwindow.h 和 mainwindow.cpp.

(1) main.app 它是整个系统的主启动文件，在连接数据库与用户登录验证通过后执行初始化好友管理系统界面，即显示 Mainwindow 主窗口。

(2) mainwindow.h 它是程序头文件，包含程序中用到的各个全局变量的定义、方法声明，同时在此文件中设置了数据库连接的静态方法。

(3) mainwindow.app 它是本程序的主体源文件，其中包含各方法功能的具体实现代码，其中框架如下：

initMainWindow(): 执行系统主窗体初始化

onTableSelectChange(int row): 在用户选择不同好友信息时进行表单更新

showFriendPhoto(): 显示 QQ 头像

loadPreFriends(): 用来在对应分组下拉框加载好友列表

onPreNameComboBoxChange(): 在改选好友时联动进行好友列表信息更新

FriendsTableView_3_clicked(const QModelIndex &index): 联动显示好友信息

preGroupComboBox_currentIndexChanged(int index): 根据分组对应加载好友列表

preNameComboBox_currentIndexChanged(int index): 选择好友时联动显示好友信息

on_DeletePushButton_clicked(): 进行删除好友操作

on_newUploadPushButton_clicked(): 进行上传头像操作

on_NewPutinPushButton_clicked(): 进行添加好友操作

on_ChangePushButton_clicked(): 进行修改好友信息操作

on_SearchPushButton_clicked(): 进行好友搜索操作

从以上代码框架可看到整个程序的运作机制，一目了然。

2.4 功能组成结构图

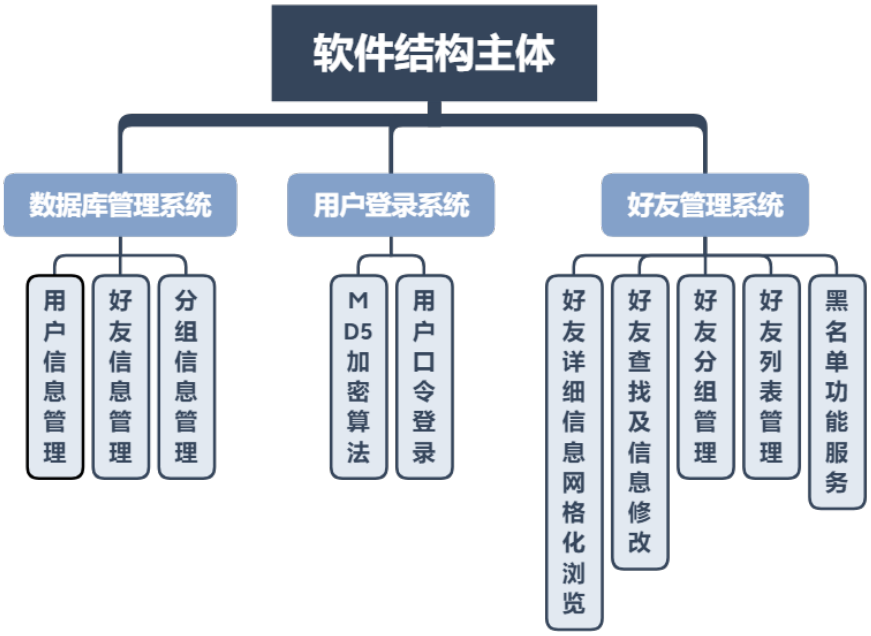

图 1: MySQL friends_inf 视图创建过程图

3 软件总体设计

3.1 软件开发环境

(1) Qt Creator 4.11.0，Based on Qt 5.14.0 (GCC 5.3.1 20160406 (Red Hat 5.3.1-6), 64 bit)

(2) Mysql 8.0.25-0ubuntu0.20.04.01

(3) OS:Ubuntu20.04LTS

(4) Opencv4.5.2

3.2 数据库设计

3.2.1 创建数据库

在 Mysql 中创建数据库，名称为 QQ_model, 其中建立三个表，分别为 Group(好友分组表)、friendsList(好友信息表)，Member(用户信息表)。

3.2.2 设计表

(1) 表结构设计

Group 表设计见下表

名	类型	长度	小数点	不是 null	虚拟	键	注释
User_ID	int			☑	☐	🔑1	YY账号
Group_ID	int			☑	☐		分组编号
Name	varchar	32		☑	☐		用户名
manual	char	2		☑	☐		性别
signature	varchar	32		☐	☐		个性签名
reference	varchar	32		☐	☐		备注
Picture	blob			☐	☐		QQ头像

图 2: Group 表设计图示

friendsList 表设计见下表

名	类型	长度	小数点	不是 null	虚拟	键	注释
Group_Name	char	16		☑	☐		分组名
Group_ID	int			☑	☐	🔑1	分组编号

图 3: friendsList 表设计图示

Member 表设计见下表

名	类型	长度	小数点	不是 null	虚拟	键	注释
MemberID	char	16		☑	☐	🔑1	用户登录账号
PassWord	char	50		☑	☐		登录口令

图 4: Member 表设计图示

(2) 外键关联

设计好表结构之后，为表之间建立外键连接。本项目在 friendsList 上建立一个外键关联，运用 Navicat for MySQL 数据库可视化工具进行设计，具体设计见下表:

名	字段	被引用的数据库	被引用的表（父）	被引用的字段	删除时	更新时
Group_ID	Group_ID	friendsList	Group	Group_ID	RESTRICT	RESTRICT

图 5: friendsList 表外键关联信息表

(3) 数据录入

设计好表及其关联之后，向各表中预先录入一些数据记录以供后面测试运行程序之用，具体见下列表:

Group_Name	Group_ID
家人	1
同学	2
朋友	3
陌生人	4
黑名单	5

图 6: Group 表预先录入信息

User_ID	Group_ID	Name	manual	signature	reference	Picture
10001	1	Yan	男	I love AI	myself	(BLOB) 23.13 KB
10002	1	Zhong	女	I love math	my family	(BLOB) 38.86 KB
10003	3	Xu	男	I love computer	my friend	(BLOB) 38.86 KB
10004	2	Huang	男	I love making	my classmate	(BLOB) 13.76 KB
10005	2	Wang	男	I love physics	my classmate	(BLOB) 9.32 KB
10007	4	Zhang	男	I love eating	strangers	(BLOB) 22.19 KB
10008	5	Liu	女	I love drinking	who?	(BLOB) 9.69 KB
10009	4	Zhou	男	I love cooking	啊啊啊	(BLOB) 39.61 KB
10010	3	He	女	I love flying	我的好朋友	(BLOB) 7.97 KB
10078	1	ZHou	男	hahha	嘿嘿	(BLOB) 39.61 KB
10086	3	Bai	男	i love football	heiheihei	(Null)
10098	2	Tian	男	我很帅哈哈哈	哈哈哈	(BLOB) 39.61 KB
10099	1	Yu	女	哈哈哈	哈哈哈	(BLOB) 13.76 KB

图 7: friendsList 表预先录入信息

MemberID	PassWord
yzs219	79bc9e8eb9798f415fee27eeeba···
yzs222	20020619yzs

图 8: Member 表预先录入信息

此外，对于图片的录入，我们通过编写代码将其通过二进制编码录入，具体代码如下：

```
QString photoPath="/home/god/QQ_model/QQ_photos/10.jpeg"; //储存路径
QFile photoFile(photoPath);
if (photoFile.exists())
{   //存入数据库
    QByteArray picdata;
    photoFile.open(QIODevice::ReadOnly);  //只读模式加载图片
    picdata=photoFile.readAll();
    photoFile.close();
    QVariant var(picdata);
    QString sqlstr = "update FriendsList set Picture =? where User_ID
   =10010";  //编写数据库更新命令
    query.prepare(sqlstr);
    query.addBindValue(var);
}
if (!query.exec())
{
    QMessageBox::information(0,QObject::tr("提示"),"照片写入失败！");
}
else
{
    QMessageBox::information(0,QObject::tr("提示"),"照片上传成功！");
}
```

3.2.3 创建视图

根据应用需要，本项目需要建立一个视图 friends_inf，用于显示好友的基本信息，创建如下图

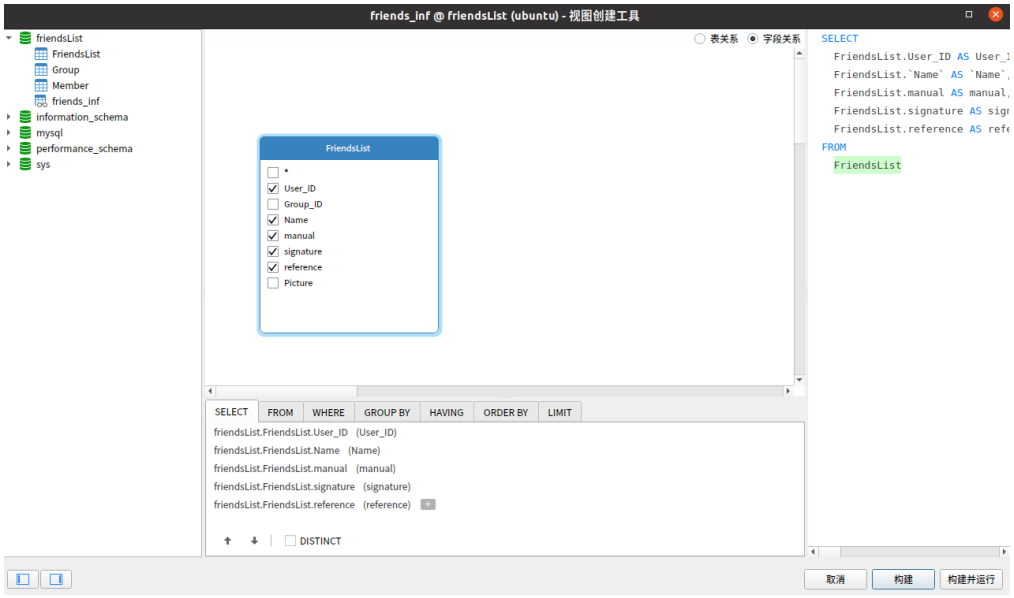

图 9: MySQL friends_inf 视图创建过程图

有了这个视图，就可以在程序中通过模型载入好友的基本信息显示于界面上，而屏蔽掉无关的信息项，非常方便。这样，系统运行所依赖的后台数据库就全部建立起来。

3.3 界面设计

3.3.1 总体设计

我们利用 QT Creator 的界面设计控件 QT designer 进行 UI 设计，为了使好友管理系统界面功能更加集中统一，我们将 QT 的 stacked Widget 控件与 Tool Box 控件结合使用，在一个统一的“好友管理”框中沿纵向布置“添加好友”、“删除好友”和“好友信息”页，分别对应系统功能需求中这三大模块，设计效果图见下图：

图 10: 初始化主窗体页面效果图

3.3.2 ”添加好友”页

“添加好友”页界面设计效果如下图所示，界面各控件属性设置见下表

图 11: “添加好友”页面效果图

表 1: “添加好友”页界面各控件属性设置

名称	类型	属性设置
newGroupComboBox	QComboBox	默认
newIDLineEdit	QLineEdit	默认
newNameLineEdit	QLineEdit	默认
newmanualLineEdit	QLineEdit	默认
newSignatureLineEDit	QLineEdit	默认
newReferenceLineEdit	QLineEdit	默认
newPictureLabel	QLabel	frameshape:Box frameShadow:Sunken text: 空 scaledContents: 勾选
newUploadPushButton	QPushButton	text: 上传...
NewPutinPushButton	QPushButton	text: 添加好友
FriendsTableView	QTableView	horizontalHeaderVisibe: 勾选 horizontalHeaderDefaultSectionSize:120 horizontalHeaderMinimumSectionSize:25 horizionHeaderStretchLastSection: 勾选 verticalHeaderVisible: 取消勾选

3.3.3 ”删除好友”页

“删除好友”页界面设计效果如下图所示，界面各控件属性设置见表

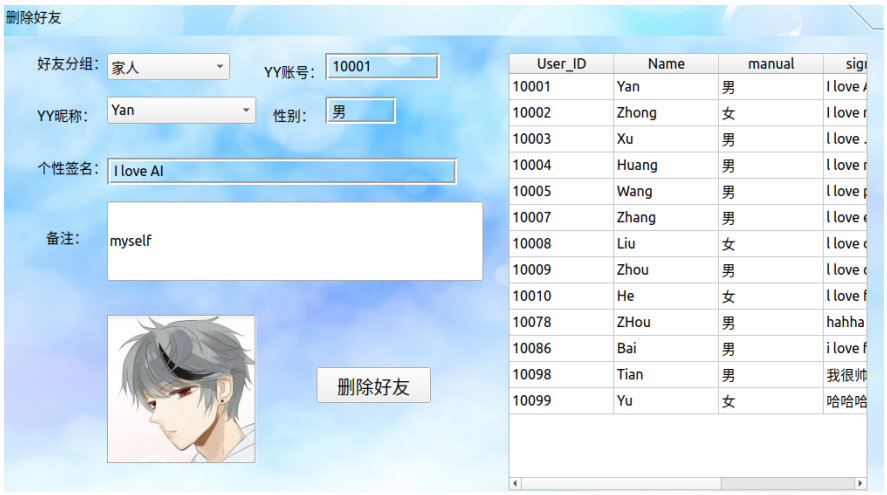

图 12: “删除好友”页面效果图

表 2: “删除好友”页界面各控件属性设置

名称	类型	属性设置
preGroupComboBox	QComboBox	默认
preIDLineEdit	QLineEdit	默认
preNameLineEdit	QLineEdit	默认
premanualLineEdit	QLineEdit	默认
preSignatureLineEDit	QLineEdit	默认
preReferenceLineEdit	QLineEdit	默认
prePictureLabel	QLabel	frameshape:Box frameShadow:Sunken text: 空 scaledContents: 勾选
DeletePushButton	QPushButton	text: 删除好友
FriendsTableView_2	QTableView	horizontalHeaderVisibe: 勾选 horizontalHeaderDefaultSectionSize:120 horizontalHeaderMinimumSectionSize:25 horizionHeaderStretchLastSection: 勾选 verticalHeaderVisible: 取消勾选

3.3.4 ”好友信息”页

“好友信息”页界面设计效果如下图所示，界面各控件属性设置见表

图 13: “好友信息”页面效果图

表 3: “好友信息”页界面各控件属性设置

名称	类型	属性设置
fixedGroupComboBox	QComboBox	默认
fixedIDLineEdit	QLineEdit	默认
fixedNameLineEdit	QLineEdit	默认
fixedmanualLineEdit	QLineEdit	默认
fixedSignatureLineEDit	QLineEdit	默认
changableReferenceLineEdit	QLineEdit	默认
SearchLineEdit	QLineEdit	text: 请输入用户账号
fixedPictureLabel	QLabel	frameshape:Box frameShadow:Sunken text: 空 scaledContents: 勾选
ChangePushButton	QPushButton	text: 确认修改
SearchPushButton	QPushButton	text: 查找
FriendsTableView_3	QTableView	horizontalHeaderVisibe: 勾选 horizontalHeaderDefaultSectionSize:120 horizontalHeaderMinimumSectionSize:25 horizionHeaderStretchLastSection: 勾选 verticalHeaderVisible: 取消勾选

3.3.5 登录窗口

登录窗口可以像程序主窗体一样在可视化设计器中进行设计，最终效果如下图所示，各控件属性设置见表

图 14: 登录窗口页面效果图

表 4: 登录窗口页界面各控件属性设置

名称	类型	属性设置
adminLineEdit	QComboBox	默认
pwdLineEdit	QLineEdit	echoMode:Password
loginPushButton	QPushButton	text: 登录
exitPushButton	QPushButton	text: 退出

3.4 MD5 加密算法设计说明

MD5 以 512 位分组来处理输入的信息，且每一分组又被划分为 16 个 32 位子分组，经过了一系列的处理后，算法的输出由四个 32 位分组组成，将这四个 32 位分组级联后将生成一个 128 位散列值。

在 MD5 算法中，首先需要对信息进行填充，使其字节长度对 512 求余数的结果等于 448。因此，信息的字节长度被扩展至 N*512+448，即 N*64+56 个字节，N 为一个正整数。填充的方法如下，在信息的后面填充一个 1 和无数个 0，直到满足上面的条件时才停止用 0 对信息的填充。然后再在这个结果后面附加一个以 64 位二进制表示的填充前的信息长度。经过这两步的处理，现在的信息字节长度=N*512+448+64=(N+1)*512，即长度恰好是 512 的整数倍数。这样做的原因是为满足后面处理中对信息长度的要求。MD5 中有四个 32 位被称作链接变量的整数参数，他们分别为：A=0x01234567，B=0x89abcdef，C=0xfedcba98，D=0x76543210。当设置好这四个链接变量后，就开始进入算法的四轮循环运算，循环的次数是信息中 512 位信息分组的数目。

将上面四个链接变量复制到另外四个变量中：A 到 a，B 到 b，C 到 c，D 到 d。主循环有四轮（MD4 只有三轮），每轮循环都很相似。第一轮进行 16 次操作。每次操作对 a、b、c 和 d 中的其中三个作一次非线性函数运算，然后将所得结果加上第四个变量（文本中的一个子分组和一个常数）。

再将所得结果向右环移一个不定的数，并加上 a、b、c 或 d 中之一。最后用该结果取代 a、b、c 或 d 中之一。以一下是每次操作中用到的四个非线性函数（每轮一个）。

$$F(X,Y,Z) = (X \wedge Y) \vee ((\sim X) \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\sim Z))$$

$$H(X,Y,Z) = X?Y?Z$$

$$\Gamma(X,Y,Z) = Y?(X \vee (\sim Z))$$

其中，?是异或，∧是与，∨是或，～是反符号。

如果 X、Y 和 Z 的对应位是独立和均匀的，那么结果的每一位也应是独立和均匀的。F 是一个逐位运算的函数。即，如果 X，那么 Y，否则 Z。函数 H 是逐位奇偶操作符。所有这些完成之后，将 A，B，C，D 分别加上 a，b，c，d。然后用下一分组数据继续运行算法，最后的输出是 A，B，C 和 D 的级联。最后得到的 A，B，C，D 就是输出结果，A 是低位，D 为高位，DCBA 组成 128 位输出结果。

具体实现编码如下：

```
QString LoginDialog::strToMd5(QString str)
{
    QString strMd5;
    QByteArray qba;
    qba=QCryptographicHash::hash(str.toLatin1(),QCryptographicHash::Md5);
   //调用QCryptographicHash类中生成密码散列的方法，生成二进制或文本数据
   的加密散列值
    strMd5.append(qba.toHex());
    return strMd5;
}
```

4 软件详细设计与编码

4.1 用户登录系统程序设计

登录功能实现在 logindialog.h 头文件和 logindialog.cpp 源文件中，首先在 logindialog.h 中声明变量和方法，完整代码如下：

```
#ifndef LOGINDIALOG_H
#define LOGINDIALOG_H

#include <QDialog>
#include <QSqlQuery>  //查询MYSQL的库
#include <QMessageBox> //弹窗信息提示库
#include <QCryptographicHash>  //包含MD5算法库
namespace Ui
{
    class LoginDialog;
}
class LoginDialog : public QDialog
{
    Q_OBJECT
    public:
    explicit LoginDialog(QWidget *parent = 0);
    ~LoginDialog();
```

```
    QString strToMd5(QString str);
    private slots:
    void on_loginPushButton_clicked();  //登录按钮单击事件槽
    void on_exitPushButton_clicked();   //退出按钮单击事件槽
    private:
    Ui::LoginDialog *ui;
};

#endif // LOGINDIALOG_H
```
然后，在logindialog.cpp源文件中实现登录验证功能，完整代码如下：
```
#include "logindialog.h"
#include "ui_logindialog.h"
LoginDialog::LoginDialog(QWidget *parent) :
QDialog(parent),
ui(new Ui::LoginDialog)
{
    ui->setupUi(this);
    setFixedSize(481,341);  //登录框固定大小
    ui->pwdLineEdit->setFocus();  //口令框置于焦点位置
}
LoginDialog::~LoginDialog()
{
    delete ui;
}
void LoginDialog::on_loginPushButton_clicked()
{
    if (!ui->pwdLineEdit->text().isEmpty())  //判断输入是否为空
    {
        QSqlQuery query;
        query.exec("select PassWord from Member where MemberID='"+  ui->
   adminLineEdit->text() + "'");  //从数据库查询口令密码
        query.next();
        QString pwdMd5=strToMd5(ui->pwdLineEdit->text());//将用户输入的口
   令字符串转为MD5加密串
        if (query.value(0).toString() == pwdMd5)
        {
            QDialog::accept();  //验证通过
        }
        else
        {
            QMessageBox::warning(this,tr("密码错误"),tr("请重新输入密码!
   "),QMessageBox::Ok);
            ui->pwdLineEdit->clear();
            ui->pwdLineEdit->setFocus();
        }
    }
    else
```

```
    {
        ui->pwdLineEdit->setFocus();
    }
}
void LoginDialog::on_exitPushButton_clicked()
{
    QDialog::reject();//退出登录界面
}
QString LoginDialog::strToMd5(QString str)  //MD5加密方法
{
    QString strMd5;
    QByteArray qba;
    qba=QCryptographicHash::hash(str.toLatin1(),QCryptographicHash::Md5);
    strMd5.append(qba.toHex());
    return strMd5;
}
```

4.2 主体程序框架

(1) main.cpp

它是整个系统的主启动文件，代码如下：

```
#include "mainwindow.h"
#include "logindialog.h"
#include <QProcess>  //Qt进程模块
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    if (!createMysqlConn()) //判断数据库连接情况
    {
        QProcess process;
        process.start("/usr/sbin/mysqld.exe");
        if (!createMysqlConn()) return 1;
    }
    LoginDialog logindlg; //登录对话框
    if (logindlg.exec()==QDialog::Accepted)
    {
        MainWindow w;
        w.show();   //启动主窗体
        return a.exec();
    }
    else
    {
        return 0;
    }
}
```

(2) mainwindow.h

它是程序头文件，包含程序中用到的各个全局变量的定义、方法声明，完整代码如下：

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QMessageBox>
#include <QFileDialog>
#include <QBuffer>
#include "opencv2/opencv.hpp"  //Opencv文件包含
#include <QSqlDatabase>        //MySQL数据库类
#include <QSqlTableModel>      //MySQL表模型类
#include <QSqlQuery>           //MySQL查询库类
#include <QTime>
#include <QPixmap>             //图像处理类库
using namespace cv;  //opencv命名空间
namespace Ui { class MainWindow; }
class MainWindow : public QMainWindow
{
    Q_OBJECT
    public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void initMainWindow();  //界面初始化方法
    void onTableSelectChange(int row);  //用户在好友列表网格选取好友
    时进行好友信息表单更新
    void showFriendPhoto();  //显示好友头像
    void loadPreFriends();  //“删除好友”页：对应分组加载好友列表
    void onPreNameComboBoxChange();  //“删除好友”页：改选好友时联动
    显示好友信息
    private slots:
    void on_FriendsTableView_3_clicked(const QModelIndex &index); //
    “好友信息”页：联动显示好友信息
    void on_preGroupComboBox_currentIndexChanged(int index); //“删除
    好友”页：根据类别对应加载对应好友列表
    void on_preNameComboBox_currentIndexChanged(int index);//“删除好
    友”页：选择好友时联动显示好友信息
    void on_DeletePushButton_clicked();  //删除好友操作
    void on_newUploadPushButton_clicked();  //上传头像
    void on_NewPutinPushButton_clicked(); //添加好友
    void on_ChangePushButton_clicked();//修改备注
    void on_SearchPushButton_clicked();//搜索好友
    private:
    Ui::MainWindow *ui;
    QImage myPicImg;  //保存好友头像
    QSqlTableModel *friends_model;  //访问数据库好友信息视图的模型
};
```

```
static bool createMysqlConn()  //访问MySQL数据库的静态方法
{
    QSqlDatabase sqldb=QSqlDatabase::addDatabase("QMYSQL");
    sqldb.setHostName("127.0.0.1");  //主机名
    sqldb.setDatabaseName("friendsList");  //数据库名称
    sqldb.setPort(3306); //接口名
    sqldb.setUserName("root"); //用户名
    sqldb.setPassword("20020619");//登录密码
    if (!sqldb.open())
    {
        QMessageBox::critical(0,QObject::tr("后台数据库连接失败"),"无
    法创建连接！请检查故障后重新连接！",QMessageBox::Cancel);
        return false;
    }
    return true;
}
#endif // MAINWINDOW_H
```

(3) mainwindow.cpp

它是本程序的主体源文件，其中包含各方法功能的具体实现代码，完整代码见附录 1.

5 总结分析与感悟

5.1 主要完成工作说明

本次实践，我主要负责了模拟 QQ 的第三版开发工作，运用 SQL 作为后台依赖，基于 QT Creator 进行软件开发，独立设立了软件 UI，运用 C++ 编写了相关功能方法代码，实现了相关接口连接，完成了软件大体设计。

5.2 软件缺陷及改进

- 鼠标右键的触发事件由于时间原因尚未完成总体设计，MySQL 中 Group 表还可以进行优化以实现分组的添加删除等功能；好友查找功能对于输入首字母弹出对于好友列表尚未实现，可再利用一个 ComBoBox 完成此项设计。
- 目前只是单用户管理员身份运行，未来可增加用户注册与服务器运行功能，实现多用户交互式平台与多用户聊天室窗口。
- UI 界面不够集成化，可将好友列表利用 Tree View 进行展示，显得更加有层次性；其次，可单独将 Table View 置于新页面，单独作为好友信息展示区，则界面看起来更为轻量化。
- 由于时间原因未来得及设计好友交互页面，可再设计一个 UI 界面作为聊天框并通过 socket 网络编程运用 TCP 协议进行信息传输实现聊天交互，同时在数据库中设计新表以存储聊天信息与时间节点。

5.3 软件设计未来规划

- 完善软件基础功能，优化 UI 界面

- 实现多用户服务器交互式聊天平台，具备独立聊天窗口与群聊界面
- 实现网络编程，用户间可通过 TCP 网络协议实现信息传输
- 实现聊天信息存储，增加历史聊天记录查看及查找功能

附录

A 附录 1 – mainwindow.cpp 程序代码

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
    /*setwindowicon*/
    //setWindowIcon(QIcon(QStringLiteral(":/background/mainlogo")));
    ui->setupUi(this);
    initMainWindow(); //执行初始化方法
}
MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::initMainWindow()  //对系统主窗体进行初始化
{
    //Qt::WindowFlags flags = this->windowFlags();
    //this->setWindowFlags(flags|Qt::WindowStaysOnTopHint);
    ui->stackedWidget->setCurrentIndex(1);  //置于好友管理页面
    ui->Friend_Operation->setCurrentIndex(0);     //“添加好友”页置于前
    面
    QSqlQueryModel *GroupModel = new QSqlQueryModel(this);  //好友类别模
    型数据
    GroupModel->setQuery("select Group_Name from `Group`");
    //QStringList strings;
    //QSqlQuery query("SELECT Group_Name FROM Group");
    //while (query.next()) {
        //        QString group_name = query.value(0).toString();
        //      strings.append(group_name);
        //}
    //ui->newGroupComboBox->addItems(strings);
    friends_model=new QSqlTableModel(this);  //好友信息视图
    friends_model->setTable("friends_inf");
    friends_model->select();
    ui->FriendsTableView->setModel(friends_model);   //初始化好友列表
    ui->FriendsTableView_2->setModel(friends_model);
    ui->FriendsTableView_3->setModel(friends_model);
    ui->newGroupComboBox->setModel(GroupModel);  //初始化下拉框
    ui->preGroupComboBox->setModel(GroupModel);
```

```
    ui->fixedGroupComboBox->setModel(GroupModel);
    loadPreFriends();
}
void MainWindow::onTableSelectChange(int row)  //“好友信息”页：用户选择
    信息时进行表单更新
{
    int r=1;
    if (row!=0) r=ui->FriendsTableView_3->currentIndex().row();
    QModelIndex index;
    index=friends_model->index(r,0);  //ID
    ui->fixedIDLabel->setText(friends_model->data(index).toString());
    index=friends_model->index(r,1);  //name
    ui->fixedNameLabel->setText(friends_model->data(index).toString());
    index=friends_model->index(r,2);  //manual
    ui->fixedmanualLabel->setText(friends_model->data(index).toString());
    index=friends_model->index(r,3);  //signature
    ui->fixedsignatureLabel->setText(friends_model->data(index).toString
    ());
    index=friends_model->index(r,4);  //reference
    ui->changableReferenceLineEdit->setText(friends_model->data(index).
    toString());
    showFriendPhoto();
    QSqlQuery query;
    query.exec(QString("select Group_Name from `Group` where Group_ID=(
    select Group_ID from FriendsList where Name='%1')").arg(ui->
    fixedNameLabel->text()));
    query.next();
    ui->fixedGroupComboBox->setCurrentText(query.value(0).toString());
    //实现分组信息联动
}
void MainWindow::showFriendPhoto()  //显示头像
{
    QPixmap photo;
    QModelIndex index;
    QSqlQueryModel *pictureModel =new QSqlQueryModel(this);
    QString name=ui->fixedNameLabel->text();
    pictureModel->setQuery("select Picture from FriendsList where Name='"
    +name+"'");
    index=pictureModel->index(0,0);
    photo.loadFromData(pictureModel->data(index).toByteArray(),"JPG");
    ui->fixedpictureLabel->setPixmap(photo);
}
void MainWindow::loadPreFriends()  //“删除好友”页：对应分组加载好友列表
{
    QSqlQueryModel *friends_name_model=new QSqlQueryModel(this);  //好友
    名称模型数据
    friends_name_model->setQuery(QString("select Name from FriendsList
```

```
    where Group_ID=(select Group_ID from `Group` where Group_Name='%1')")
    \
     .arg(ui->preGroupComboBox->currentText()));//加载好友列表
     ui->preNameComboBox->setModel(friends_name_model);
     onPreNameComboBoxChange();
}
void MainWindow::onPreNameComboBoxChange()  //“删除好友”页：改选好友时
    联动显示好友信息
{
     QSqlQueryModel *PreFriendsModel=new QSqlQueryModel(this);
     QString name=ui->preNameComboBox->currentText();
     PreFriendsModel->setQuery("select User_ID,manual,signature,reference,
    Picture from FriendsList where Name='"+name+"'");
     QModelIndex index;
     index=PreFriendsModel->index(0,0);  //账号
     ui->preIDLabel->setText(PreFriendsModel->data(index).toString());
     index=PreFriendsModel->index(0,1);  //性别
     ui->preManualLabel->setText(PreFriendsModel->data(index).toString());
     index=PreFriendsModel->index(0,2);  //个性签名
     ui->preSignatureLabel->setText(PreFriendsModel->data(index).toString
    ());
     index=PreFriendsModel->index(0,3);  //备注
     ui->preReferenceLineEdit->setText(PreFriendsModel->data(index).
    toString());
     //获取头像
     QPixmap photo;
     index=PreFriendsModel->index(0,4);
     photo.loadFromData(PreFriendsModel->data(index).toByteArray(),"JPG");
     ui->prePictureLabel->setPixmap(photo);
}
void MainWindow::on_FriendsTableView_3_clicked(const QModelIndex &index)
     //“好友信息”页：联动显示好友信息
{
     onTableSelectChange(1);
}
void MainWindow::on_preGroupComboBox_currentIndexChanged(int index)   //
     “删除好友”页：根据类别对应加载对应好友列表
{
     loadPreFriends();
}
void MainWindow::on_preNameComboBox_currentIndexChanged(int index)  //
     “删除好友”页：选择好友时联动显示好友信息
{
     onPreNameComboBoxChange();
}
void MainWindow::on_DeletePushButton_clicked()  //“删除好友”页：删除好
    友操作
```

```
{
    QSqlQuery query;
    query.exec(QString("delete from FriendsList where Name='%1'").arg(ui
   ->preNameComboBox->currentText()));//删除好友记录
    //刷新界面
    ui->preManualLabel->setText("");
    ui->preSignatureLabel->setText("");
    ui->preIDLabel->setText("");
    ui->preReferenceLineEdit->setText("");
    ui->prePictureLabel->clear();
    friends_model->setTable("friends_inf");
    friends_model->select();
    //刷新数据网格
    ui->FriendsTableView->setModel(friends_model);
    ui->FriendsTableView_2->setModel(friends_model);
    ui->FriendsTableView_3->setModel(friends_model);
}
void MainWindow::on_newUploadPushButton_clicked()  //上传头像
{
    QString picturename=QFileDialog::getOpenFileName(this,"选择头像",".",
   "Image File(*.png *.jpg *.jpeg *.bmp)");
    if (picturename.isEmpty()) return;
    myPicImg.load(picturename);
    ui->NewPictureLabel->setPixmap(QPixmap::fromImage(myPicImg));
}
void MainWindow::on_NewPutinPushButton_clicked() //添加好友
{
    QSqlQuery query;
    query.exec(QString("select Group_ID from `Group` where Group_Name
   ='%1'").arg(ui->newGroupComboBox->currentText()));
    query.next();
    int Group_ID=query.value(0).toInt();
    QString name =ui->newNameLineEdit->text();
    int User_ID=ui->newIDLineEdit->text().toInt();
    QString maunal=ui->newmanualLineEdit->text();
    QString signature=ui->newSignatureLineEdit->text();
    QString reference=ui->newRefenrenceLineEdit->text();
    query.exec(QString("insert into FriendsList(User_ID,Group_ID,Name,
   manual,signature,reference,Picture)values(%1,%2,'%3','%4','%5','%6',
   NULL)")\
    .arg(User_ID).arg(Group_ID).arg(name).arg(maunal).arg(signature).arg(
   reference));//添加好友信息
    //插入照片
    QByteArray picdata;
    QBuffer buffer(&picdata);
    buffer.open(QIODevice::WriteOnly);
    myPicImg.save(&buffer,"JPG");
```

```
    QVariant var(picdata);
    QString sqlstr="update FriendsList set Picture=? where Name='"+name+"
    '";
    query.prepare(sqlstr);
    query.addBindValue(var);
    if (!query.exec())
    {
        QMessageBox::information(0,QObject::tr("提示"),"添加失败");
    }
    else
    {
        QMessageBox::information(0,QObject::tr("提示"),"添加成功！");
    }
    ui->newIDLineEdit->setText("");
    ui->newNameLineEdit->setText("");
    ui->newmanualLineEdit->setText("");
    ui->newSignatureLineEdit->setText("");
    ui->newRefenrenceLineEdit->setText("");
    ui->NewPictureLabel->clear();
    friends_model->setTable("friends_inf");   //刷新网格信息
    friends_model->select();
    ui->FriendsTableView->setModel(friends_model);
    ui->FriendsTableView_2->setModel(friends_model);
    ui->FriendsTableView_3->setModel(friends_model);
}
void MainWindow::on_ChangePushButton_clicked()  //“好友信息”页：修改好
    友信息
{
    QSqlQuery query;
    QString new_reference=ui->changableReferenceLineEdit->text();
    query.exec(QString("update FriendsList set reference='%1' where Name
    ='%2'").arg(new_reference).arg(ui->fixedNameLabel->text()));
    if (!query.exec())
    {
        QMessageBox::information(0,QObject::tr("提示"),"修改失败");
    }
    else
    {
        QMessageBox::information(0,QObject::tr("提示"),"修改成功！");
    }
    friends_model->setTable("friends_inf");   //刷新网格信息
    friends_model->select();
    ui->FriendsTableView->setModel(friends_model);
    ui->FriendsTableView_2->setModel(friends_model);
    ui->FriendsTableView_3->setModel(friends_model);
}
void MainWindow::on_SearchPushButton_clicked()  //搜索好友
```

```
{
    QSqlQuery query;
    query.exec(QString("select User_ID,Group_ID,Name,manual,signature,
   reference,Picture from FriendsList where User_ID='%1'").\
    arg(ui->SearchLineEdit->text().toInt()));
    if (query.exec())
    {
        query.next();
        ui->fixedIDLabel->setText(query.value(0).toString());
        ui->fixedNameLabel->setText(query.value(2).toString());
        ui->fixedmanualLabel->setText(query.value(3).toString());
        ui->fixedsignatureLabel->setText(query.value(4).toString());
        ui->changableReferenceLineEdit->setText(query.value(5).toString()
   );
        showFriendPhoto();
        query.exec(QString("select Group_Name from `Group` where Group_ID
   =(select Group_ID from FriendsList where Name='%1')").\
        arg(ui->fixedNameLabel->text()));
        query.next();
        ui->fixedGroupComboBox->setCurrentText(query.value(0).toString())
   ;
    }
    else
    {
        QMessageBox::information(0,QObject::tr("提示"),"未检测到此好友");
    }
}
```

参考文献

[1] 陆文周.Qt5 开发及实例（第 4 版）[M]. 电子工业出版社: 北京,2019:403-434.

[2] 汤研. 基于 TCP 的简易聊天室设计与实现 [J]. 电脑编程技巧与维护,2020(12):27-29.