**南昌航空大学**

**20学年—21学年第 2 学期 医疗软件技术基础 大作业**

专业名称： 生物医学工程 实验学时： 2

学号： 19084127 姓名： 周亚诺

实验题目： QT数据库编程

实验环境： QT6.2，Mysql

实验目的：

1．掌握 QT数据库操作组件；

2．掌握Mysql数据库软件操作；

3. 掌握基本的SQL语句。

实验内容：

（1）通过Mysql设计数据库；

（2）分别建立病人信息查询、修改、删除和开处方操作界面；

（3）实现数据的添加、删除、修改、查询操作；

实验代码：

QT += quick qml sql \

widgets

CONFIG += c++11

# The following define makes your compiler emit warnings if you use

# any feature of Qt which as been marked deprecated (the exact warnings

# depend on your compiler). Please consult the documentation of the

# deprecated API in order to know how to port your code away from it.

DEFINES += QT\_DEPRECATED\_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.

# In order to do so, uncomment the following line.

# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += main.cpp \

cdataclass.cpp \

cworker.cpp \

mycommon.cpp

RESOURCES += qml.qrc

# Additional import path used to resolve QML modules in Qt Creator's code model

QML\_IMPORT\_PATH =

# Additional import path used to resolve QML modules just for Qt Quick Designer

QML\_DESIGNER\_IMPORT\_PATH =

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

DISTFILES += main.qml \

Rx.qml \

Add.qml \

Edit.qml

HEADERS += \

cdataclass.h \

cworker.h \

mycommon.h

#include <QGuiApplication>

#include <QQmlApplicationEngine>

int main(int argc, char \*argv[])

{

QGuiApplication app(argc, argv);

qmlRegisterType<CDataClass>("dataclass", 1, 0, "CDataClass"); // 注册类，Q\_INVOKABLE方式调用c++里面函数

qmlRegisterType<UserTableModel>("UserTableModel", 0, 1, "UserTableModel");

qmlRegisterType<LogTableModel>("LogTableModel", 0, 1, "LogTableModel");

qmlRegisterType<MyTreeModel>("MyTreeModel", 0, 1, "MyTreeModel");

MyCommon::instance(); //初始化单例类

QQmlApplicationEngine engine;

CDataClass dataClass;

engine.rootContext()->setContextProperty("datacls", &dataClass);

const QUrl url(u"qrc:/main.qml"\_qs);

QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,

&app, [url](QObject \*obj, const QUrl &objUrl) {

if (!obj && url == objUrl)

QCoreApplication::exit(-1);

}, Qt::QueuedConnection);

engine.load(url);

return app.exec();

}

#ifndef MYCOMMON\_H

#define MYCOMMON\_H

#include <QWidget>

//定义数据处理结果状态码

enum RET\_CODE {

RET\_OK = 0, //成功

RET\_DBERR\_OPEN, //数据库查询打开失败

RET\_DBERR\_RUN, //SQL执行失败

RET\_PARAMERR, //参数错误

RET\_NOFUNC, //方法不存在

RET\_NOWORKTYPE //处理类型不存在

};

//定义数据处理结果状态码对应的信息

extern QStringList RET\_MSG;

//定义命令参数数据结构

typedef struct \_CmdData {

QString func; //处理的函数名称

QMap<QString, QString> params; //参数列表

} CmdData;

//定义命令处理结果数据结构

typedef struct \_RstData {

int retCode; //结果状态码

QString func; //处理的函数名称

QString msg; //结果信息

QVector< QVector<QString> > result; //处理结果数据，二位数组

} RstData;

// MYSQL数据库信息

typedef struct \_MysqlInfo

{

int port;

QString host, name, usr, pwd;

} MysqlInfo;

class MyCommon : public QWidget

{

Q\_OBJECT

public:

explicit MyCommon(QWidget \*parent = nullptr);

~MyCommon();

static MyCommon \*instance(); //定义单例类

static void InitDataBase(const MysqlInfo &dbInfo); //初始化数据库

static QSqlDatabase GetNewDatabase() //获取数据库实例

{

QSqlDatabase newDb;

if (QSqlDatabase::contains("mysql\_1"))

{

int n = QSqlDatabase::connectionNames().size();

newDb = QSqlDatabase::cloneDatabase(mDatabase, QString("mysql\_%1").arg(n));

}

else

{

newDb = QSqlDatabase::cloneDatabase(mDatabase, "mysql\_1");

}

return newDb;

}

signals:

public slots:

private:

static float xScal, yScal;

static QRect mScreenRect;

static MyCommon \*self;//单例模式

static QSqlDatabase mDatabase;

static MysqlInfo mDbInfo;

};

#endif // MYCOMMON\_H

#include "mycommon.h"

MyCommon \*MyCommon::self = nullptr;

QSqlDatabase MyCommon::mDatabase;

MysqlInfo MyCommon::mDbInfo;

QStringList RET\_MSG = QStringList() << "成功" << "数据库查询打开失败" << "SQL执行失败" << "参数错误" << "方法不存在" << "处理类型不存在";

MyCommon::MyCommon(QObject \*parent) : QObject(parent)

{

//注册元类型:主要是在定义信号槽的时候，传递的参数类型不一定是QT所识别的

qRegisterMetaType<CmdData>("CmdData");

qRegisterMetaType<RstData>("RstData");

mDbInfo.port = 3306;

mDbInfo.host = "127.0.0.1";

mDbInfo.name = "test\_db";

mDbInfo.usr = "root";

mDbInfo.pwd = "123456";

InitDataBase(mDbInfo);

}

MyCommon::~MyCommon()

{

if (self != nullptr)

{

delete self;

}

}

void MyCommon::InitDataBase(const MysqlInfo &dbInfo)

{

mDatabase = QSqlDatabase::addDatabase("QMYSQL");

mDatabase.setHostName(dbInfo.host);//设置主机地址

mDatabase.setPort(dbInfo.port); //设置端口

mDatabase.setDatabaseName(dbInfo.name); //设置数据库名称

mDatabase.setUserName(dbInfo.usr); //设置用户名

mDatabase.setPassword(dbInfo.pwd); //设置密码

}

#ifndef CDATACLASS\_H

#define CDATACLASS\_H

#include <QObject>

class CDataClass;

typedef void (CDataClass::\*PTRFUN)(const CmdData &argcs); //函数指针，用于分发命令

class CDataClass : public QObject

{

Q\_OBJECT

public:

explicit CDataClass(QObject \*parent = nullptr);

~CDataClass()

{

mWorkerThread.quit();

mWorkerThread.wait();

}

Q\_INVOKABLE void handleCmdDataQML(const QString &func, const QStringList &keys,

const QStringList &values); //供页面调用的命令函数，分发到具体的处理函数

Q\_INVOKABLE void checkUserPwd(const CmdData &argcs); //验证输入的用户名和密码

Q\_INVOKABLE void getUsersData(const CmdData &argcs); //查询用户信息

Q\_INVOKABLE void addUsersData(const CmdData &argcs); //增加用户信息

Q\_INVOKABLE void editUsersData(const CmdData &argcs); //编辑用户信息

Q\_INVOKABLE void getLogsData(const CmdData &argcs); //查询日志信息

signals:

//把页面接受的命令，发送到线程里面的槽函数

void operate(const int type, const QString &func, const QString &cmd);

//把线程里面的处理结果返回给页面

void operateResult(const RstData &rstData);

//登录结果信号

void signalLoginResult(const bool &result);

//消息显示信号

void signalMeaasge(const QString &msg);

public slots:

void handleResults(const RstData &rstData); //接受线程里面处理结果

private:

QThread mWorkerThread; //定义处理线程

QMap<QString, PTRFUN> mFuncMap; //定义命令处理函数映射关系

};

#endif // CDATACLASS\_H

#include "cdataclass.h"

CDataClass::CDataClass(QObject \*parent) : QObject(parent)

{

Worker \*worker = new Worker; //定义数据处理类

worker->moveToThread(&mWorkerThread); //把数据处理类移到线程

connect(&mWorkerThread, &QThread::finished, worker, &QObject::deleteLater);

//定义信号槽，把命令发送到线程里面的槽函数

connect(this, &CDataClass::operate, worker, &Worker::doWork);

//定义信号槽，接收线程里面发送的结果

connect(worker, &Worker::resultReady, this, &CDataClass::handleResults);

mWorkerThread.start(); //开启线程

//初始化命令处理函数映射关系

mFuncMap["checkUserPwd"] = &CDataClass::checkUserPwd;

mFuncMap["getUsersData"] = &CDataClass::getUsersData;

mFuncMap["addUsersData"] = &CDataClass::addUsersData;

mFuncMap["editUsersData"] = &CDataClass::editUsersData;

mFuncMap["getLogsData"] = &CDataClass::getLogsData;

}

void CDataClass::handleResults(const RstData &rstData)

{

if (rstData.func == "checkUserPwd")

{

if (rstData.result.size() > 0)

{

emit signalLoginResult(true);

}

else

{

emit signalLoginResult(false);

}

}

else

{

emit operateResult(rstData);

}

}

void CDataClass::handleCmdDataQML(const QString &func, const QStringList &keys,

const QStringList &values)

{

qDebug() << "[CDataClass::handleCmdDataQML]" << func << keys << values;

CmdData argcs;

argcs.func = func;

if (keys.size() != values.size())

{

return;

}

for (int var = 0; var < keys.size(); ++var)

{

argcs.params[keys[var]] = values[var];

}

handleCmdData(argcs);

}

void CDataClass::handleCmdData(const CmdData &argcs)

{

RstData rstData;

if (!mFuncMap.contains(argcs.func))

{

rstData.retCode = RET\_NOFUNC;

rstData.msg = RET\_MSG[rstData.retCode];

rstData.func = argcs.func;

emit signalMeaasge(rstData.msg);

return;

}

(this->\*mFuncMap[argcs.func])(argcs);

}

#ifndef CWORKER\_H

#define CWORKER\_H

#include <QObject>

enum WORK\_TYPE {

WORK\_DB\_QUERY = 0, //数据库查询

WORK\_DB\_RUN //数据库更新（增、删、改）

};

class Worker : public QObject

{

Q\_OBJECT

public:

explicit Worker(QObject \*parent = nullptr);

void testAddData1(); //测试函数

signals:

void resultReady(const RstData &rstData); //返回处理结果信号

public slots:

void doWork(const int type, const QString &func, const QString &cmd); //接收页面命令槽函数

private:

QSqlDatabase mDatabase; //数据库操作对象

int RunSql(const QString &sqlStr); //执行sql语句，写入接口

int RunSql(const QString &prepare, const QMap<QString, QVariant> &values);

int RunSqlColRow(const QString &sqlStr, QVector< QVector<QString> > &result); //执行sql语句，查询接口, 返回二维数组[列][行]

int RunSqlRowCol(const QString &sqlStr, QVector< QVector<QString> > &result); //执行sql语句，查询接口, 返回二维数组[行][列]

};

#endif // CWORKER\_H

import QtQuick 2.0

import QtQuick.Controls 2.12

Window {

visible: true

width: 640

height: 600

title: *qsTr*("MedicalSoftware")

property int i :0

onCompleted: {

odbc.connect('pass', mysql, '192.168.0.239')

}

StackView {

id: *stack*

initialItem: *mainView*

anchors.fill: *parent*

}

function *showMsg*(title, text) {

messageDialog.title = *title*;

messageDialog.text = *text*;

messageDialog.visible = true;

}

function *handleStartProgressBar*() {

progressBar.visible = true;

progressBar.start();

}

function *handleSignalLoginResult*(result)

{

progressBar.setProgressValue(100);

if (*result*)

{

loginpage.visible = false;

mainwld.windowTitle = 'QML演示系统';

}

else

{

*showMsg*('提示', '账号或密码不正确！');

}

}

function *handleSignalMeaasge*(msg)

{

progressBar.setProgressValue(100);

*showMsg*('提示', *msg*);

}

Connections

{

target: *datacls*

onSignalLoginResult: *handleSignalLoginResult*(result)

onSignalMeaasge: *handleSignalMeaasge*(msg)

}

Timer {

id: *timeUpdate*

interval: 1000; running: true; repeat: true

onTriggered: {

//console.log(Qt.formatDateTime(new Date(), "dddd hh:mm:ss"), new Date().getDay());

labelTime.text = Qt.formatDateTime(new *Date*(), "dddd hh:mm:ss");

}

}

Page {

id: *mainView*

Button {

id: *search\_button*

x: 295

y: 31

width: 75

height: 25

text: *qsTr*("搜索")

topPadding: 2

}

TextField {

id: *textField*

x: 134

y: 31

height: 25

placeholderText: *qsTr*("请输入关键词")

onEditingFinished: {

if (textInput.focus) {

*tablemodelid*.getData(['search'], [textInput.text]);

progressBar.visible = true;

progressBar.start();

}

textInput.focus = false;

}

}

ComboBox {

id: *comboBox*

x: 28

y: 31

width: 75

height: 25

model: ListModel{

id: *combo*

ListElement {

text: "姓名"

}

ListElement {

text: "性别"

}

ListElement {

text: "出生日期"

}

ListElement {

text: "居住地址"

}

ListElement {

text: "联系电话"

}

}

}

Button {

id: *add\_button*

x: 549

y: 31

width: 75

height: 25

text: *qsTr*("添加记录")

topPadding: 2

onClicked: {

pass(id.selected());

*page\_add*.visible =true;

*page\_add*.stack = *stack*;

*stack*.push(*page\_add*)

}

}

Button {

id: *edit\_button*

x: 549

y: 68

width: 75

height: 25

text: *qsTr*("修改记录")

topPadding: 2

onClicked: {

pass(id.selected());

*page\_edit*.visible =true;

*page\_edit*.stack = *stack*;

*stack*.push(*page\_edit*)

}

}

Button {

id: *rx\_button*

x: 549

y: 105

width: 75

height: 25

text: *qsTr*("处方")

topPadding: 2

onClicked: {

pass(id.selected());

*page\_rx*.visible =true;

*page\_rx*.stack = *stack*;

*stack*.push(*page\_rx*)

}

}

Button {

id: *delete\_button*

x: 549

y: 142

width: 75

height: 25

text: *qsTr*("删除记录")

onClicked: {

obdc.pop(id.selected())

}

}

Button {

id: *undo\_button*

x: 549

y: 179

width: 75

height: 25

text: *qsTr*("撤销操作")

onClicked: {

obdc.reverse()

}

}

Rectangle {

x: 28

y: 68

width: 492; height: 475

Component {

id: *contactDelegate*

Item {

width: 492; height: 21

Row {

spacing: 1

Rectangle {

width: 20

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: num ;font.pointSize: 12 }

}

Rectangle{

width: 80

height: 20

color: "#00ffffff"

border.width: 1

Text { text: name ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: birth ;font.pointSize: 12 }

}

Rectangle{

width: 120

height: 20

color: "#00ffffff"

border.width: 1

Text { text: address ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: number ;font.pointSize: 12 }

}

}

}

}

Row {

anchors.fill: *parent*

anchors.bottomMargin: 450

spacing: 1

Rectangle {

width: 20

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "#" ;font.pointSize: 12 }

}

Rectangle{

width: 80

height: 20

color: "#00ffffff"

border.width: 0

Text { text: "姓名" ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "出生日期" ;font.pointSize: 12 }

}

Rectangle{

width: 120

height: 20

color: "#00ffffff"

border.width: 0

Text { text: "居住地址" ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "联系电话" ; font.pointSize: 12 }

}

}

ListView {

anchors.fill: *parent*

anchors.topMargin: 26

delegate: *contactDelegate*

highlight: Rectangle { color: "lightsteelblue"; radius: 5 }

focus: true

model: UserTableModel {

id: *tablemodelid*

onSignalRecvhandleResult: {

progressBar.setProgressValue(100);

}

}

Component.onCompleted: {

*tablemodelid*.getData(all);

}

}

}

}

Add{

id:*page\_add*

visible: false

}

Edit{

id:*page\_edit*

visible: false

}

Rx{

id:*page\_rx*

visible: false

}

}

import QtQuick 2.0

import QtQuick.Controls 2.15

Page {

id: *addWindow*

visible: true

property StackView stack: null

Button {

height: 25

width: 75

text: "<-"

anchors.left: *parent*.left

anchors.top: *parent*.top

flat: false

onClicked: *stack*.pop()

}

Text {

id: *text1*

x: 230

y: 37

text: *qsTr*("你将添加以下记录：")

font.pixelSize: 20

}

Text {

id: *text2*

x: 190

y: 147

text: *qsTr*("姓名：")

font.pixelSize: 20

}

Text {

id: *text3*

x: 190

y: 187

text: *qsTr*("性别：")

font.pixelSize: 20

}

Text {

id: *text4*

x: 150

y: 227

text: *qsTr*("出生日期：")

font.pixelSize: 20

}

Text {

id: *text5*

x: 150

y: 267

text: *qsTr*("居住地址：")

font.pixelSize: 20

}

Text {

id: *text6*

x: 150

y: 307

text: *qsTr*("联系电话：")

font.pixelSize: 20

}

Button {

id: *button*

x: 276

y: 420

text: *qsTr*("确认添加")

font.pixelSize: 20

onClicked: {

tablemodelid.append(['append'], ['name', 'gender', 'birth', 'address', 'phone'], [*name\_field*, *gender\_field*, *birth\_field*, *address\_field*, *phone\_field*])

*stack*.pop()

}

}

TextField {

id: *name\_field*

x: 256

y: 147

width: 200

placeholderText: *qsTr*("请输入姓名")

font.pixelSize: 20

}

TextField {

id: *gender\_field*

x: 256

y: 187

width: 200

placeholderText: *qsTr*("请输入性别")

font.pixelSize: 20

}

TextField {

id: *birth\_field*

x: 256

y: 227

width: 200

placeholderText: *qsTr*("请输入出生日期")

font.pixelSize: 20

}

TextField {

id: *address\_field*

x: 256

y: 267

width: 200

placeholderText: *qsTr*("请输入居住地址")

font.pixelSize: 20

}

TextField {

id: *phone\_field*

x: 256

y: 307

width: 200

placeholderText: *qsTr*("请输入联系电话")

font.pixelSize: 20

}

}

/\*##^##

Designer {

D{i:0;autoSize:true;height:480;width:640}

}

##^##\*/

import QtQuick 2.0

import QtQuick.Controls 2.15

Page {

id: *editWindow*

visible: true

property StackView stack: null

Button {

height: 25

width: 75

text: "<-"

anchors.left: *parent*.left

anchors.top: *parent*.top

flat: false

onClicked: *stack*.pop()

}

Text {

id: *text1*

x: 230

y: 37

text: *qsTr*("你将修改以下记录：")

font.pixelSize: 20

}

Text {

id: *text2*

x: 190

y: 147

text: *qsTr*("姓名：")

font.pixelSize: 20

}

Text {

id: *text3*

x: 190

y: 187

text: *qsTr*("性别：")

font.pixelSize: 20

}

Text {

id: *text4*

x: 150

y: 227

text: *qsTr*("出生日期：")

font.pixelSize: 20

}

Text {

id: *text5*

x: 150

y: 267

text: *qsTr*("居住地址：")

font.pixelSize: 20

}

Text {

id: *text6*

x: 150

y: 307

text: *qsTr*("联系电话：")

font.pixelSize: 20

}

Button {

id: *button*

x: 276

y: 420

text: *qsTr*("确认修改")

font.pixelSize: 20

onClicked: {

*tablemodelid*.drop(passed.get\_id())

*tablemodelid*.append(['append'], ['name', 'gender', 'birth', 'address', 'phone'], [*name\_field*, *gender\_field*, *birth\_field*, *address\_field*, *phone\_field*])

*stack*.pop()

}

}

TextField {

id: *name\_field*

x: 256

y: 147

width: 200

placeholderText: passed.get\_field('name')

font.pixelSize: 20

}

TextField {

id: *gender\_field*

x: 256

y: 187

width: 200

placeholderText: passed.get\_field('gender')

font.pixelSize: 20

}

TextField {

id: *birth\_field*

x: 256

y: 227

width: 200

placeholderText: passed.get\_field('birth')

font.pixelSize: 20

}

TextField {

id: *address\_field*

x: 256

y: 267

width: 200

placeholderText: passed.get\_field('address')

font.pixelSize: 20

}

TextField {

id: *phone\_field*

x: 256

y: 307

width: 200

placeholderText: passed.get\_field('phone')

font.pixelSize: 20

}

}

/\*##^##

Designer {

D{i:0;autoSize:true;height:480;width:640}

}

##^##\*/

import QtQuick 2.0

import QtQuick.Controls 2.15

Page {

id: *rxWindow*

visible: true

property StackView stack: null

Button {

height: 25

width: 75

text: "<-"

anchors.left: *parent*.left

anchors.top: *parent*.top

flat: false

onClicked: *stack*.pop()

}

Rectangle {

x: 24

y: 39

width: 492; height: 50

Component {

id: *contactDelegate*

Item {

width: 492; height: 21

Row {

spacing: 1

Rectangle {

width: 20

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: num ;font.pointSize: 12 }

}

Rectangle{

width: 80

height: 20

color: "#00ffffff"

border.width: 1

Text { text: name ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: birth ;font.pointSize: 12 }

}

Rectangle{

width: 120

height: 20

color: "#00ffffff"

border.width: 1

Text { text: address ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 1

Text { text: number ;font.pointSize: 12 }

}

}

}

}

Row {

anchors.fill: *parent*

anchors.bottomMargin: 450

spacing: 1

Rectangle {

width: 20

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "#" ;font.pointSize: 12 }

}

Rectangle{

width: 80

height: 20

color: "#00ffffff"

border.width: 0

Text { text: "姓名" ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "出生日期" ;font.pointSize: 12 }

}

Rectangle{

width: 120

height: 20

color: "#00ffffff"

border.width: 0

Text { text: "居住地址" ;font.pointSize: 12 }

}

Rectangle{

width: 100

height: 20

color: "lightsteelblue"

border.width: 0

Text { text: "联系电话" ; font.pointSize: 12 }

}

}

ListView {

anchors.fill: *parent*

anchors.topMargin: 26

delegate: *contactDelegate*

highlight: Rectangle { color: "lightsteelblue"; radius: 5 }

focus: true

model:UserTableModel {

id: *tablemodelid*

onSignalRecvhandleResult: {

progressBar.setProgressValue(100);

}

}

Component.onCompleted: {

*tablemodelid*.getData(passed.get\_id());

}

}

}

Rectangle {

x: 24

y: 95

width: 492

height: 183

ListView {

id: *listView*

anchors.fill: *parent*

delegate: Item {

x: 5

width: 80

height: 40

Text {

text: name

anchors.verticalCenter: *parent*.verticalCenter

font.pointSize: 12

}

}

model:UserTableModel {

id: *med*

onSignalRecvhandleResult: {

progressBar.setProgressValue(100);

}

}

Component.onCompleted: {

*med*.getData(obdc='medic', passed.get\_id());

}

}

}

TextArea {

id: *textArea*

x: 24

y: 293

width: 492

height: 157

placeholderText: *qsTr*("在此处添加处方")

}

Button {

id: *edit\_rx\_button*

x: 534

y: 95

width: 75

height: 25

text: *qsTr*("修改处方")

onClicked: {

*med*.drop(id.selected())

*med*.append(['rx'], [*textArea*.getFormattedText()])

}

}

Button {

id: *del\_rx\_button*

x: 534

y: 132

width: 75

height: 25

text: *qsTr*("删除处方")

onClicked: *med*.drop(id.selected())

}

Button {

id: *add\_rx\_button*

x: 534

y: 293

width: 75

height: 25

text: *qsTr*("添加处方")

onClicked: *med*.append(['rx'], [*textArea*.getFormattedText()])

}

}

/\*##^##

Designer {

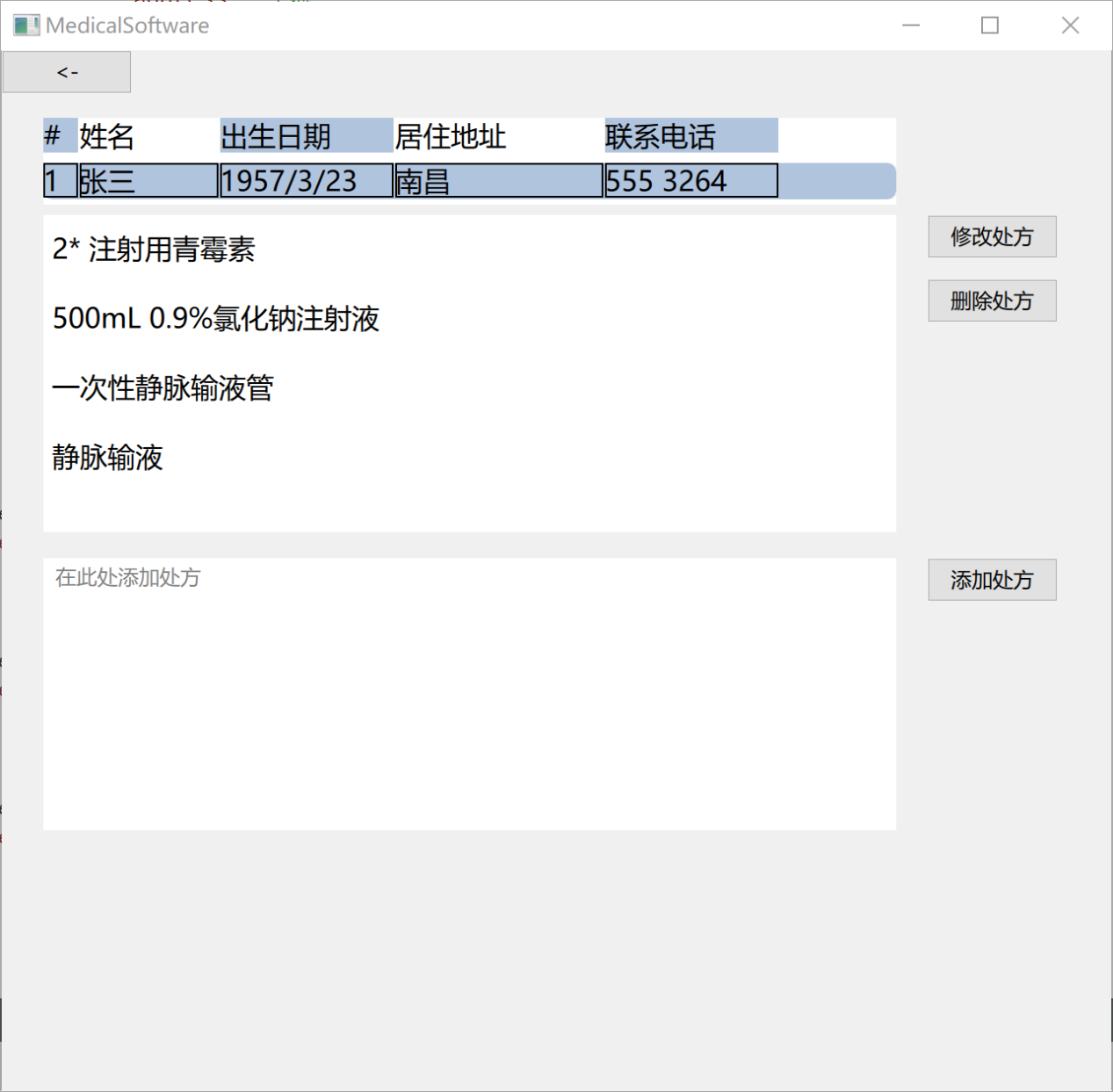
D{i:0;autoSize:true;height:480;width:640}D{i:42}D{i:43}

}

##^##\*/









实验小结：

掌握了QT数据库操作组件，掌握了MySQL数据库软件操作，掌握了基本的SQL语句。