**Министерство науки и высшего образования** **Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления» Кафедра «Системы обработки информации и управления»**

Лабораторная работа №4

Курс «Базы данных»

«Создание оконного приложения   
для работы с базой данных»

Выполнил:

студент группы № ИУ5-45Б

Расулов А.Н.

Проверил:

Маслеников К.Ю.

2024 г.

**Цель лабораторной работы:**

Создать в среде Qt Creator на языке Qt С++ оконное приложение для работы с базой

данных.

mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QSqlDatabase>

#include <QTableWidgetItem>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui;

QSqlDatabase dbconn;

public slots:

void dbconnect();

void selectAll();

void add();

void edit();

void del();

void findSecondary();

void addSecondary();

void delSecondary();

void editSecondary();

private slots:

void on\_twOrg\_2\_itemClicked(QTableWidgetItem \*item);

void on\_twOrg\_itemClicked(QTableWidgetItem \*item);

};

#endif // MAINWINDOW\_H

main.cpp:

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QSqlError>

#include <QMessageBox>

#include <QSqlQuery>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(ui->btnConnect, SIGNAL(clicked(bool)),this, SLOT(dbconnect()));

connect(ui->btnSelectAll, SIGNAL(clicked(bool)),this, SLOT(selectAll()));

connect(ui->btnAdd, SIGNAL(clicked(bool)),this, SLOT(add()));

connect(ui->btnEdit, SIGNAL(clicked(bool)),this, SLOT(edit()));

connect(ui->btnDel, SIGNAL(clicked(bool)),this, SLOT(del()));

connect(ui->btnAdd\_2, SIGNAL(clicked(bool)),this, SLOT(addSecondary()));

connect(ui->btnEdit\_2, SIGNAL(clicked(bool)),this, SLOT(editSecondary()));

connect(ui->btnDel\_2, SIGNAL(clicked(bool)),this, SLOT(delSecondary()));

// Количество столбцов

ui->twOrg->setColumnCount(5);

// Возможность прокрутки

ui->twOrg->setAutoScroll(true);

// Режим выделения ячеек - только одна строка

ui->twOrg->setSelectionMode(QAbstractItemView::SingleSelection);

ui->twOrg->setSelectionBehavior(QAbstractItemView::SelectRows);

// Заголовки таблицы

ui->twOrg->setHorizontalHeaderItem(0, new QTableWidgetItem("ID"));

ui->twOrg->setHorizontalHeaderItem(1, new QTableWidgetItem("ФИО"));

ui->twOrg->setHorizontalHeaderItem(2, new QTableWidgetItem("Телефон"));

ui->twOrg->setHorizontalHeaderItem(3, new QTableWidgetItem("Адрес"));

ui->twOrg->setHorizontalHeaderItem(4, new QTableWidgetItem("Почта"));

ui->twOrg->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

// Разрешаем сортировку пользователю

ui->twOrg->setSortingEnabled(true);

// Запрет на изменение ячеек таблицы при отображении

ui->twOrg->setEditTriggers(QAbstractItemView::NoEditTriggers);

// Вторая таблица

// Количество столбцов

ui->twOrg\_2->setColumnCount(5);

// Возможность прокрутки

ui->twOrg\_2->setAutoScroll(true);

// Режим выделения ячеек - только одна строка

ui->twOrg\_2->setSelectionMode(QAbstractItemView::SingleSelection);

ui->twOrg\_2->setSelectionBehavior(QAbstractItemView::SelectRows);

// Заголовки таблицы

ui->twOrg\_2->setHorizontalHeaderItem(0, new QTableWidgetItem("ID"));

ui->twOrg\_2->setHorizontalHeaderItem(1, new QTableWidgetItem("Цена"));

ui->twOrg\_2->setHorizontalHeaderItem(2, new QTableWidgetItem("Дедлайн"));

ui->twOrg\_2->setHorizontalHeaderItem(3, new QTableWidgetItem("Дата создания"));

ui->twOrg\_2->setHorizontalHeaderItem(4, new QTableWidgetItem("Выполнен?"));

// Последний столбец растягивается при изменении размера формы

ui->twOrg\_2->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

// Разрешаем сортировку пользователю

ui->twOrg\_2->setSortingEnabled(true);

// Запрет на изменение ячеек таблицы при отображении

ui->twOrg\_2->setEditTriggers(QAbstractItemView::NoEditTriggers);

}

void MainWindow::dbconnect() {

if(!dbconn.isOpen()) {

// Если соединение не открыто, то вывести список доступных драйверов БД

// (вывод в поле teResult, метод append добавляет строки).

ui->teResult->append("SQL drivers:");

ui->teResult->append(QSqlDatabase::drivers().join(","));

dbconn=QSqlDatabase::addDatabase("QPSQL");

dbconn.setDatabaseName("laba4");

dbconn.setHostName("127.0.0.1");

dbconn.setUserName("topg");

dbconn.setPassword("1234");

// Открыть соединениe и результат вывести в окно вывода

if( dbconn.open() )

ui->teResult->append("Соединение открыто...");

else

{

ui->teResult->append("Ошибка соединения:");

ui->teResult->append(dbconn.lastError().text());

}

}

else

// Если соединение уже открыто, то сообщить об этом

ui->teResult->append("Соединение уже открыто...");

}

void MainWindow::selectAll() {

// Очистить содержимое компонента

ui->twOrg->clearContents();

// Если соединение не открыто, то вызвать нашу функцию для открытия

// если подключиться не удалось, то вывести сообщение об ошибке и

// выйти из функции

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this, "Ошибка", dbconn.lastError().text());

return;

}

}

// Создать объект запроса с привязкой к установленному соединению

QSqlQuery query(dbconn);

// Создать строку запроса на выборку данных

QString sqlstr = "SELECT \* FROM clients ORDER BY id ASC";

// Выполнить запрос и поверить его успешность

if( !query.exec(sqlstr) )

{

QMessageBox::critical(this,"Ошибка", query.lastError().text());

return;

}

// Если запрос активен (успешно завершен),

// то вывести сообщение о прочитанном количестве строк в окно вывода

// и установить количество строк для компонента таблицы

if( query.isActive())

ui->twOrg->setRowCount(query.size());

else

ui->twOrg->setRowCount( 0);

ui->teResult->append( QString("Считано строк: %1").arg(query.size()));

// Прочитать в цикле все строки результата (курсора)

// и вывести их в компонент таблицы

int i=0;

while(query.next())

{

ui->twOrg->setItem(i,0,new

QTableWidgetItem(query.value("id").toString()));

ui->twOrg->setItem(i,1,new

QTableWidgetItem(query.value("fullname").toString()));

ui->twOrg->setItem(i,2,new

QTableWidgetItem(query.value("telephone").toString()));

ui->twOrg->setItem(i,3,new

QTableWidgetItem(query.value("address").toString()));

ui->twOrg->setItem(i,4,new

QTableWidgetItem(query.value("email").toString()));

i++;

}

}

void MainWindow::add() {

// Подключиться к БД

if( !dbconn.isOpen() ) {

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

}

QSqlQuery query(dbconn);

// Создать строку запроса

QString sqlstr = "insert into clients(fullname, telephone, address, email) values(?,?,?,?)";

// Подготовить запрос

query.prepare(sqlstr);

// Передать параметры из полей ввода в запрос

query.bindValue(0,ui->leName->text());

query.bindValue(1,ui->leTelephone->text());

query.bindValue(2, ui->leAddress->text());

query.bindValue(3, ui->leEmail->text());

// Выполнить запрос

if( !query.exec() )

{

ui->teResult->append(query.lastQuery());

QMessageBox::critical(this,"Ошибка",query.lastError().text());

return;

}

// Если запрос выполнен, то вывести сообщение одобавлении строки

ui->teResult->append( QString("Добавлен %1ряд").arg(query.numRowsAffected()) );

// и обновить записи в компоненте таблицы

selectAll();

}

void MainWindow::edit() {

if(!dbconn.isOpen())

dbconnect();

if(!dbconn.isOpen()) {

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

// Получить номер выбранной строки в компоненте таблицы

int cur\_row = ui->twOrg->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if( cur\_row < 0 )

{

QMessageBox::critical(this,"Ошибка","Строка не выбрана!");

return;

}

QSqlQuery query(dbconn);

// Создать строку запроса

QString sqlstr = "update clients set fullname=:fn, telephone=:tel, address=:address, email=:em WHERE id=:id";

// Подготовить запрос

query.prepare(sqlstr);

// Передать параметры из полей ввода в запрос

auto fullname = ui->leName->text();

auto telephone = ui->leTelephone->text();

auto email = ui->leEmail->text();

auto address = ui->leAddress->text();

query.bindValue(":fn", fullname);

query.bindValue(":tel", telephone);

query.bindValue(":address", address);

query.bindValue(":em", email);

query.bindValue(":id", ui->twOrg->item(cur\_row, 0)->text().toInt());

// Выполнить запрос

if( !query.exec() ) {

ui->teResult->append( query.lastQuery());

QMessageBox::critical(this,"Ошибка",query.lastError().text());

return;

}

qDebug() << query.executedQuery();

// Если запрос выполнен, то вывести сообщение об обновлении строки

ui->teResult->append( QString("Edited 1 row"));

// и обновить записи в компоненте таблицы

selectAll();

}

void MainWindow::del() {

// Подключение к БД

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

}

// Получить номер выбранной строки в компоненте таблицы

int currow = ui->twOrg->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if( currow < 0 )

{

QMessageBox::critical(this,"Ошибка","Строка не выбрана!");

return;

}

// Спросить у пользователя подтверждение удаления записи

// Используется статический метод QMessageBox::question

// для задания вопроса, который возвращает код нажатой кнопки

if( QMessageBox::question(this,"Удаление","Удалить строку?",

QMessageBox::Cancel,QMessageBox::Ok)==QMessageBox::Cancel)

return;

// Создать объект запроса

QSqlQuery query(dbconn);

// Создать строку запроса.

// Вместо подготовки запроса и передачи параметров значение параметра

// конкатенируется со строкой запроса

// Обратите,что строковое значение помещается в одинарные кавычки

// Значение выбирается из компонента таблицы методом item(row,col)

QString sqlstr = "delete from clients where fullname = '"

+ ui->twOrg->item(currow,0)->text() + "'";

// Выполнить строку запроса и проверить его успешность

if( !query.exec(sqlstr) )

{

ui->teResult->append( query.lastQuery());

QMessageBox::critical(this,"Ошибка",query.lastError().text());

return;

}

// Вывести сообщение об удалении строки

ui->teResult->append( QString("Удалено строк: %1").arg(query.numRowsAffected()) );

// Обновить содержимое компонента таблицы

selectAll();

}

void MainWindow::findSecondary() {

// Очистить содержимое компонента

ui->twOrg\_2->clearContents();

// Если соединение не открыто, то вызвать нашу функцию для открытия

// если подключиться не удалось, то вывести сообщение об ошибке и

// выйти из функции

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

}

int currow = ui->twOrg->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if( currow < 0 )

{

QMessageBox::critical(this, "Ошибка", "Строка не выбрана!");

return;

}

// Создать объект запроса с привязкой к установленному соединению

QSqlQuery query(dbconn);

// Создать строку запроса на выборку данных

QString sqlstr = "select id, price, deadline, date\_of\_creation, done from orders where client\_id = :id ORDER BY id ASC";

query.prepare(sqlstr);

query.bindValue(":id",ui->twOrg->item(currow, 0)->text().toInt());

// Выполнить запрос и поверить его успешность

if( !query.exec() ) {

QMessageBox::critical(this, "Ошибка", query.lastError().text());

return;

}

// Если запрос активен (успешно завершен),

// то вывести сообщение о прочитанном количестве строк в окно вывода

// и установить количество строк для компонента таблицы

if( query.isActive())

ui->twOrg\_2->setRowCount(query.size());

else

ui->twOrg\_2->setRowCount(0);

ui->teResult->append( QString("Считано строк: %1").arg(query.size()));

// Прочитать в цикле все строки результата (курсора)

// и вывести их в компонент таблицы

int i=0;

while(query.next()) {

ui->twOrg\_2->setItem(i,0,new

QTableWidgetItem(query.value("id").toString()));

ui->twOrg\_2->setItem(i,1,new

QTableWidgetItem(query.value("price").toString()));

ui->twOrg\_2->setItem(i,2,new

QTableWidgetItem(query.value("deadline").toString()));

ui->twOrg\_2->setItem(i,3,new

QTableWidgetItem(query.value("date\_of\_creation").toString()));

ui->twOrg\_2->setItem(i,4,new

QTableWidgetItem(query.value("done").toString()));

i++;

}

}

void MainWindow::addSecondary() {

// Подключиться к БД

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

}

int currow = ui->twOrg->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if( currow < 0 ) {

QMessageBox::critical(this,"Ошибка","Строка не выбрана!");

return;

}

QSqlQuery query(dbconn);

// Создать строку запроса

QString sqlstr = "insert into orders(client\_id, price, deadline, date\_of\_creation, done) "

"values (:client\_id, :price, :deadline, :doc, :done)";

// Подготовить запрос

query.prepare(sqlstr);

// Передать параметры из полей ввода в запрос

query.bindValue(":client\_id",ui->twOrg->item(currow, 0)->text());

query.bindValue(":price", ui->lePrice->text());

query.bindValue(":deadline",ui->deDeadline->text());

query.bindValue(":doc",ui->deCreate->text());

query.bindValue(":done",ui->checkBox->isChecked());

// Выполнить запрос

if( !query.exec() ) {

ui->teResult->append( query.lastQuery());

QMessageBox::critical(this,"Ошибка",query.lastError().text());

return;

}

// Если запрос выполнен, то вывести сообщение одобавлении строки

ui->teResult->append( QString("Добавлен %1 ряд").arg(query.numRowsAffected()) );

// и обновить записи в компоненте таблицы

findSecondary();

}

void MainWindow::editSecondary() {

if(!dbconn.isOpen())

{

dbconnect();

}

if(!dbconn.isOpen())

{

QMessageBox::critical(this,"Error",dbconn.lastError().text());

return;

}

// Получить номер выбранной строки в компоненте таблицы

int currow = ui->twOrg\_2->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if(currow < 0) {

QMessageBox::critical(this,"Ошибка","Строка не выбрана!");

return;

}

QSqlQuery query(dbconn);

// Создать строку запроса

QString sqlstr = "update orders set price=:pr, deadline=:d, date\_of\_creation=:doc, done=:done where id=:id";

query.prepare(sqlstr);

const auto price = ui->lePrice->text();

const auto deadline = ui->deDeadline->text();

const auto creation\_date = ui->deCreate->text();

const auto done = ui->checkBox->isChecked();

query.bindValue(":pr", price);

query.bindValue(":d", deadline);

query.bindValue(":doc", creation\_date);

query.bindValue(":done", done);

query.bindValue(":id",ui->twOrg\_2->item(currow, 0)->text().toInt());

// Выполнить запрос

if( !query.exec() )

{

ui->teResult->append( query.lastQuery());

QMessageBox::critical(this,"Error",query.lastError().text());

return;

}

// Если запрос выполнен, то вывести сообщение об обновлении строки

ui->teResult->append( QString("Edited 1 row"));

// и обновить записи в компоненте таблицы

findSecondary();

}

void MainWindow::delSecondary() {

// Подключиться к БД

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

QMessageBox::critical(this,"Ошибка",dbconn.lastError().text());

return;

}

}

int currow = ui->twOrg\_2->currentRow();

// Если он меньше 0 (строка не выбрана), то

// сообщение об ошибке и выход из функции

if( currow < 0 )

{

QMessageBox::critical(this,"Ошибка","Строка не выбрана!");

return;

}

if( QMessageBox::question(this,"Удаление","Удалить строку?",

QMessageBox::Cancel,QMessageBox::Ok)==QMessageBox::Cancel)

return;

// Создать объект запроса

QSqlQuery query(dbconn);

QString sqlstr = "delete from orders where id=" + ui->twOrg\_2->item(currow, 0)->text();

// Выполнить строку запроса и проверить его успешность

if( !query.exec(sqlstr) ) {

ui->teResult->append( query.lastQuery());

QMessageBox::critical(this,"Ошибка",query.lastError().text());

return;

}

ui->teResult->append( QString("Удалено строк: %1").arg(query.numRowsAffected()) );

// Обновить содержимое компонента таблицы

findSecondary();

}

MainWindow::~MainWindow() {

if( dbconn.isOpen())

dbconn.close();

delete ui;

}

void MainWindow::on\_twOrg\_2\_itemClicked(QTableWidgetItem \*item) {

int currow = item->row();

const auto price = ui->twOrg\_2->item(currow, 1)->text();

const auto deadline = ui->twOrg\_2->item(currow, 2)->text();

const auto creation\_date = ui->twOrg\_2->item(currow, 3)->text();

const auto done = ui->twOrg\_2->item(currow, 4)->text() == "true";

ui->lePrice->setText(price);

ui->deDeadline->setDate(QDate::fromString(deadline, "yyyy-MM-dd"));

ui->deCreate->setDate(QDate::fromString(creation\_date, "yyyy-MM-dd"));

ui->checkBox->setChecked(done);

}

void MainWindow::on\_twOrg\_itemClicked(QTableWidgetItem \*item) {

int currow = item->row();

const auto fullname = ui->twOrg->item(currow, 1)->text();

const auto tel = ui->twOrg->item(currow, 2)->text();

const auto address = ui->twOrg->item(currow, 3)->text();

const auto email = ui->twOrg->item(currow, 4)->text();

ui->leName->setText(fullname);

ui->leTelephone->setText(tel);

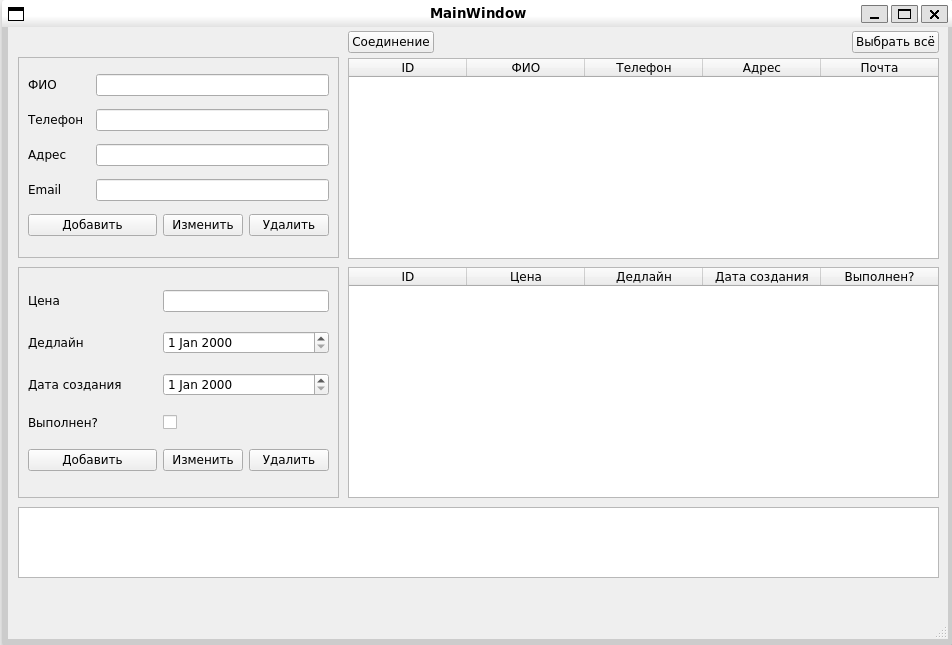
ui->leAddress->setText(address);

ui->leEmail->setText(email);

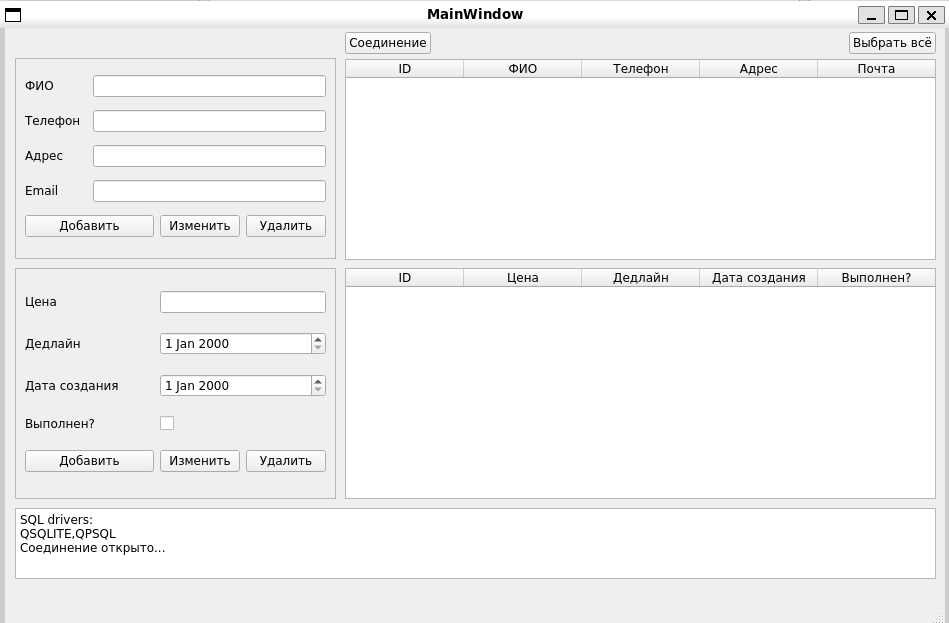
findSecondary();

}

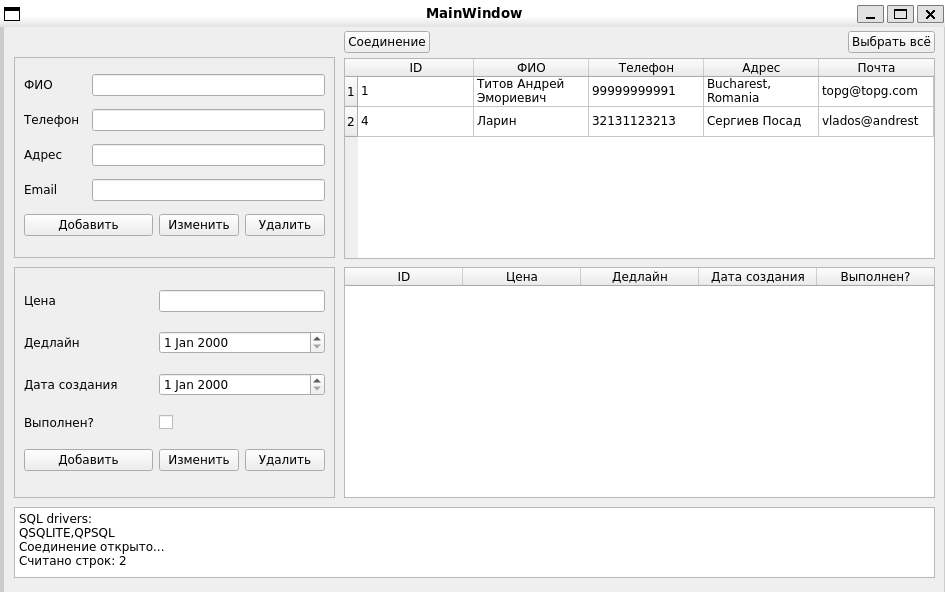
Главное окно:



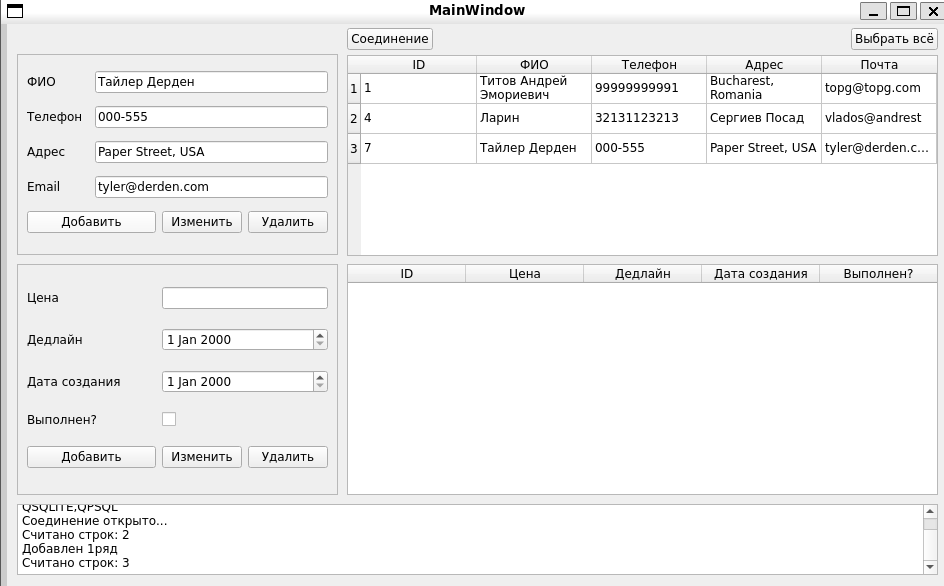
Проверка установленных драйверов для работы с БД и установка соединения с БД:



Отображение записей из таблицы:

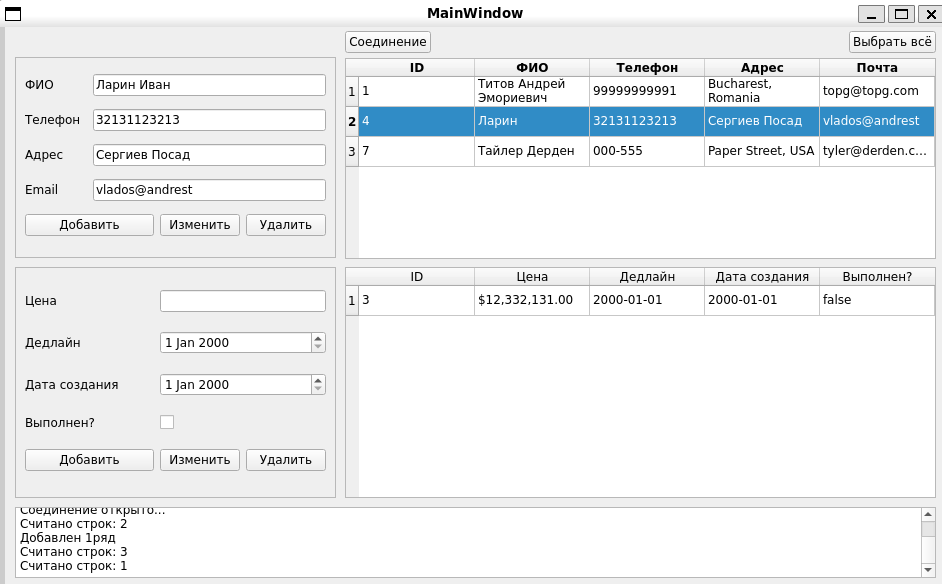


Проверка добавления записей:

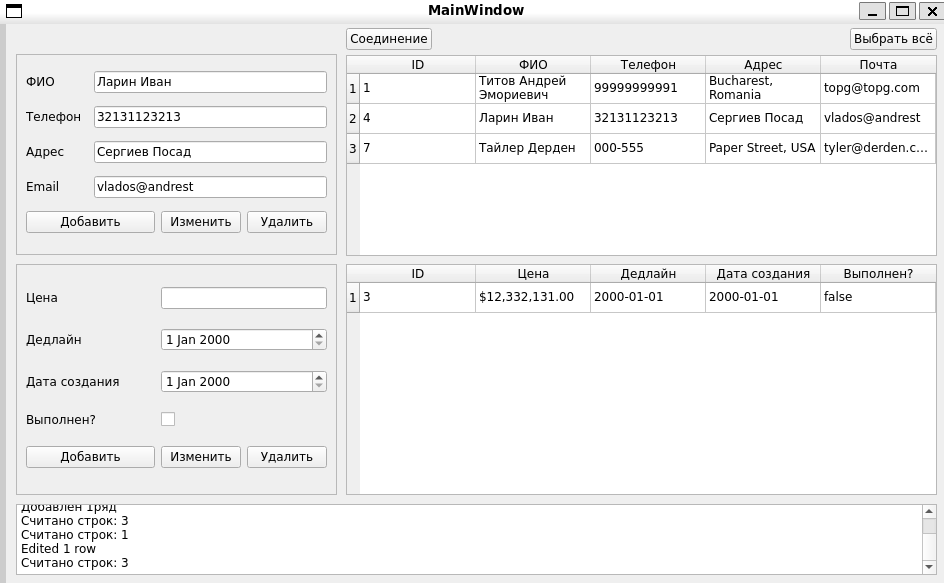


Проверка редактирования записей:

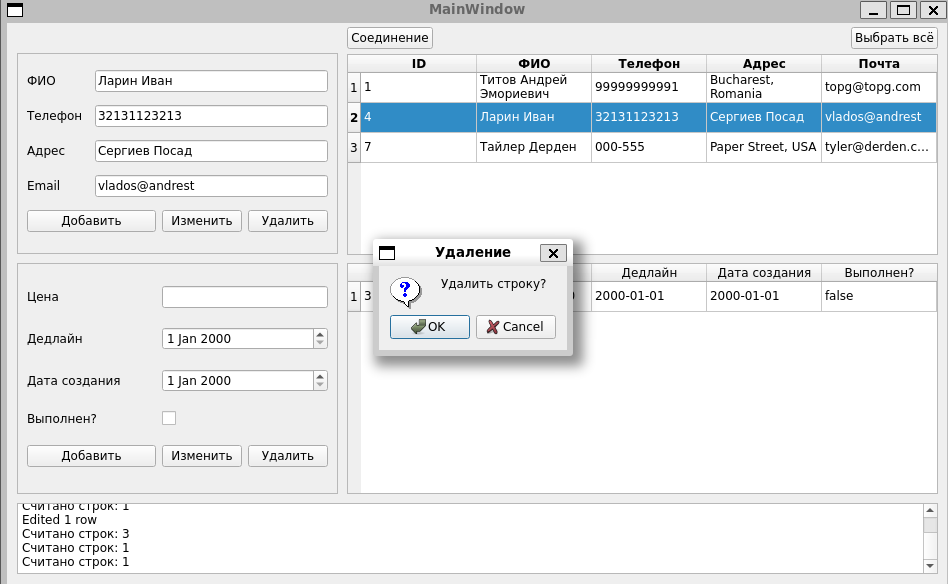
*До*

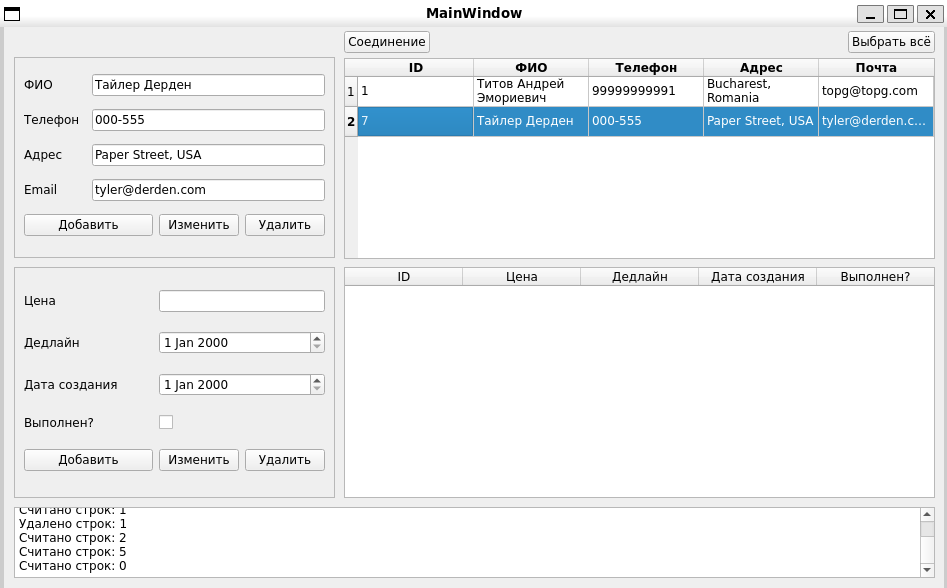


*После*

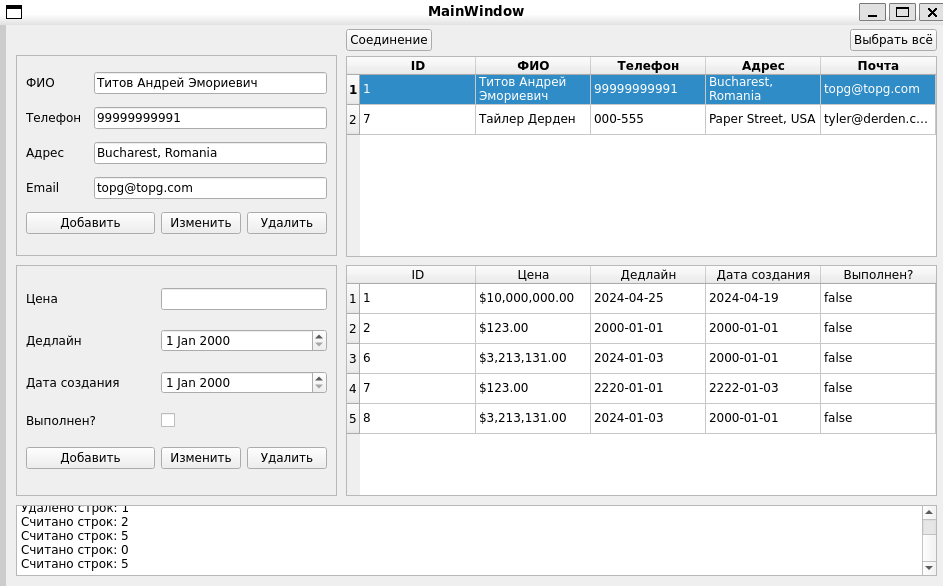


Проверка удаления записей:

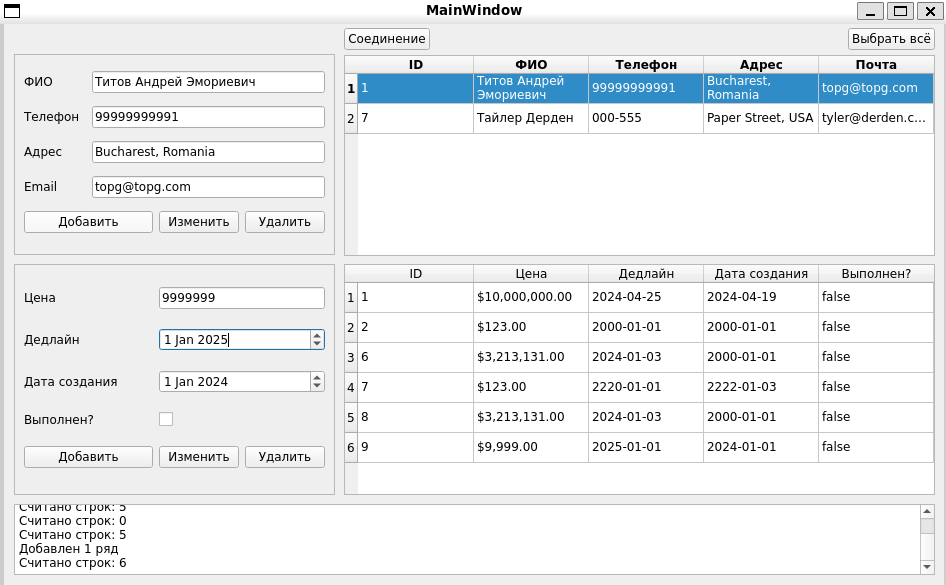


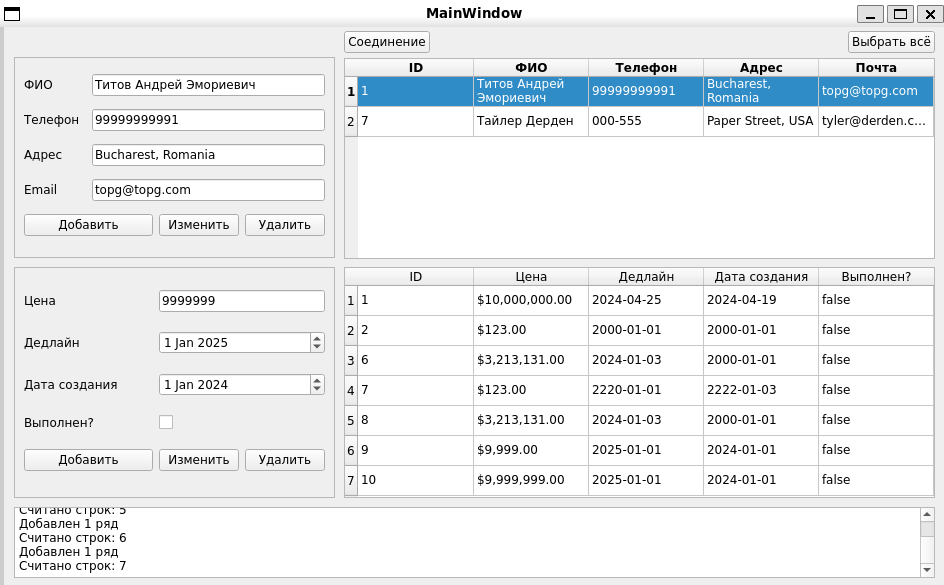


Результат выполнения запроса к двум таблицам:



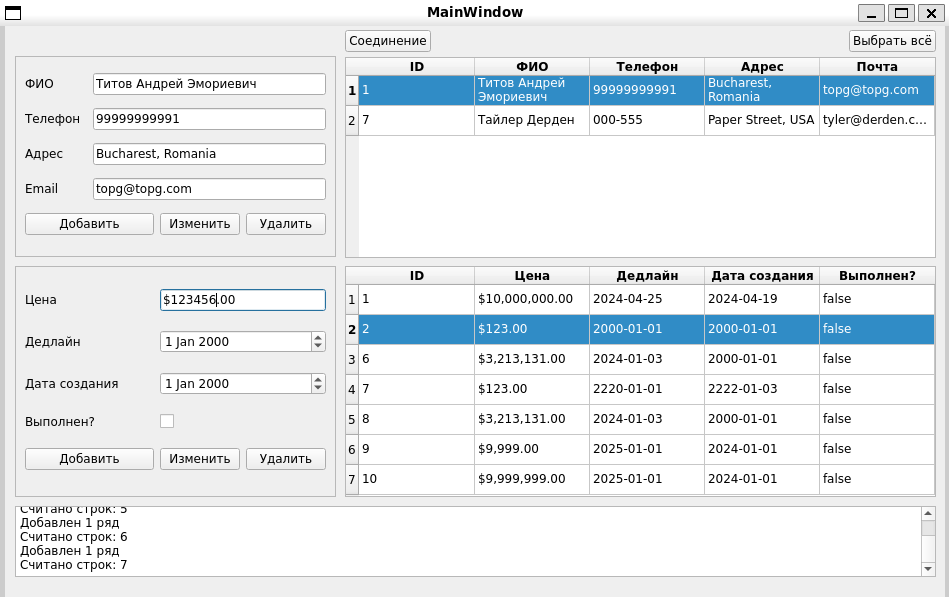
Проверка добавления записей:

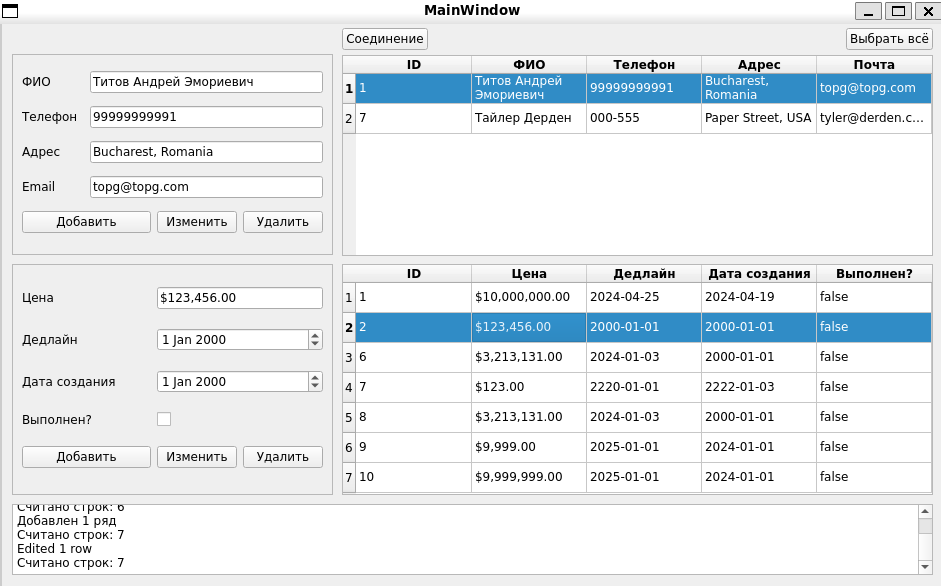




Проверка изменения записей:

*До:*



*После:*

Проверка удаления записей:

