**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базы данных»

Отчет по лабораторной работе №4

"Создание оконного приложения для работы с базой данных"

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-42Б |  | преподаватель каф. ИУ5 |
| Афонин И.И. |  | Силантьева Е. Ю. |

2025 г.

## Цель лабораторной работы

Изучить основы работы с Qt, получить навыки создания взаимодействий между приложением и базой данных PostgreSQL.

## Полученное задание

Спроектировать интерфейс приложения, содержащий две взаимосвязанных формы с данными из базы данных с возможностью чтения, добавления, обновления, удаления данных, кнопки для перехода между формами.

Требования к интерфейсу:

Форма 1 должна содержать:

• две связанные между собой таблицы из базы данных;

• подписи названий таблиц/столбцов;

• навигацию;

• компоненты типа Textbox для изменения данных;

• кнопку для обновления информации в таблицах, кнопку сохранения;

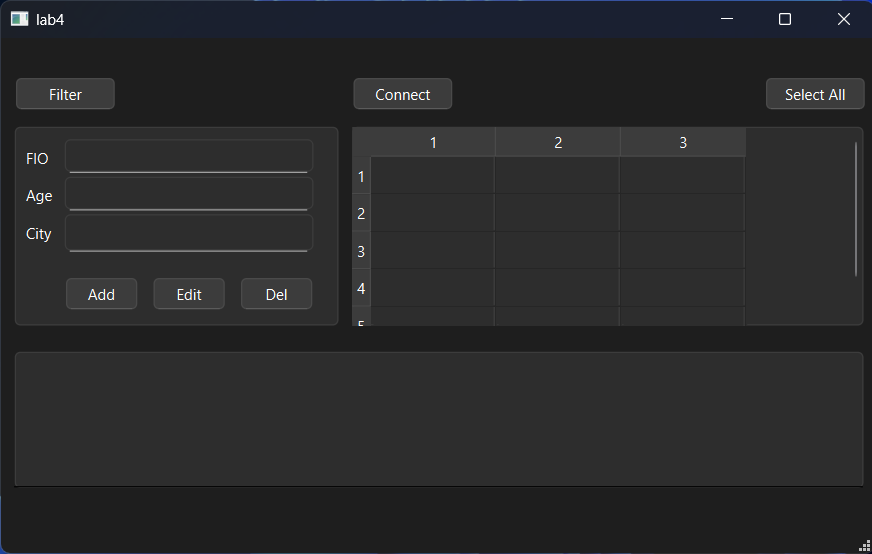
• кнопку перехода к другой форме (форма 2).

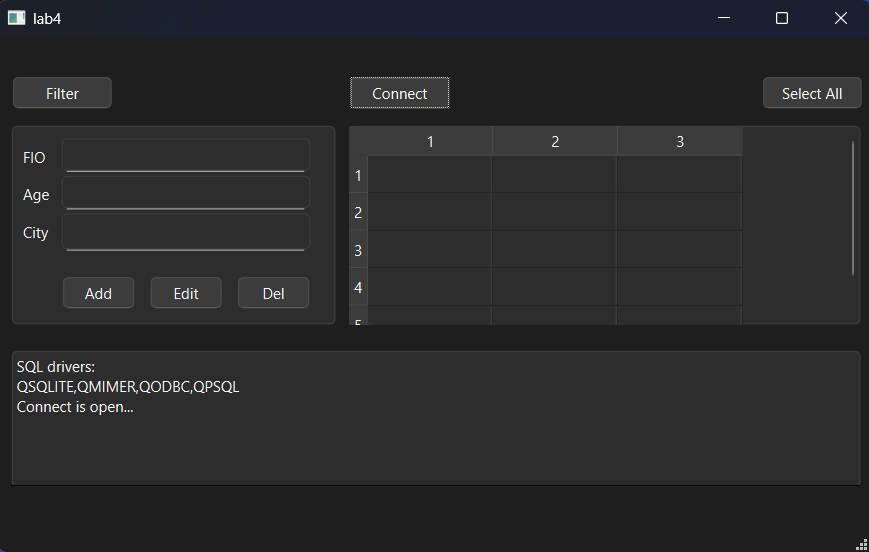
Форма 2 должно содержать:

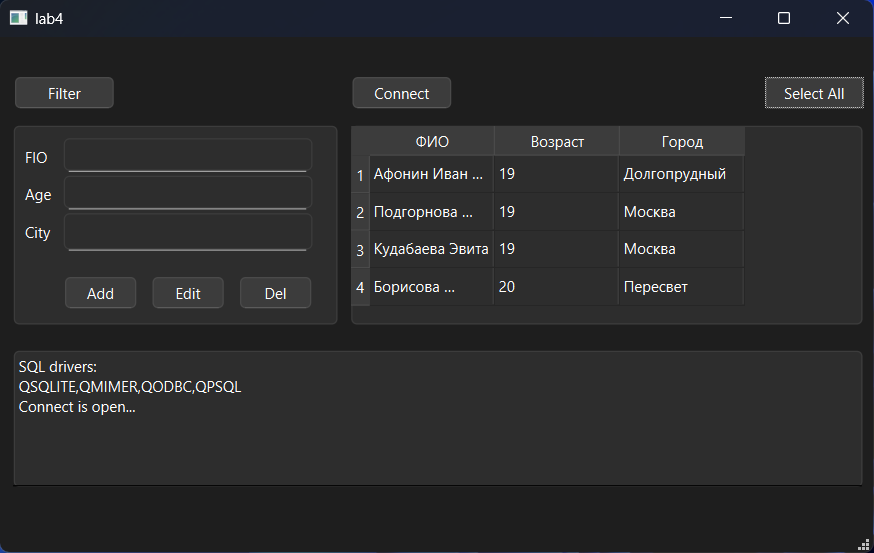
• вывод результата запроса с фильтром по текущей записи Формы 1.

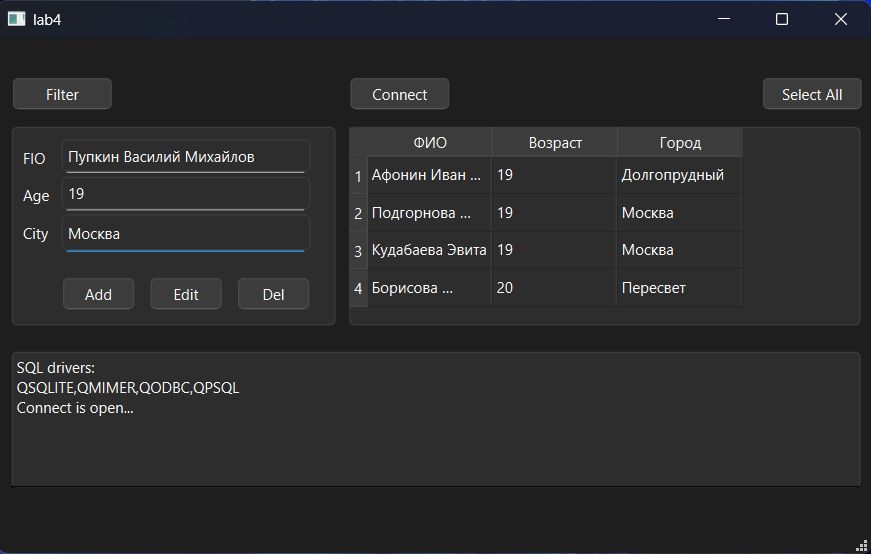
## Выполнение лабораторной работы

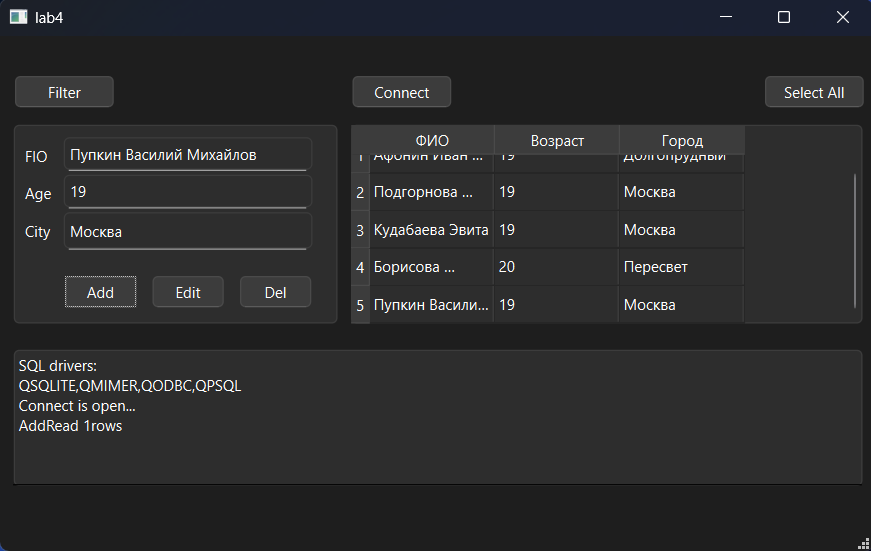
Начальная форма:

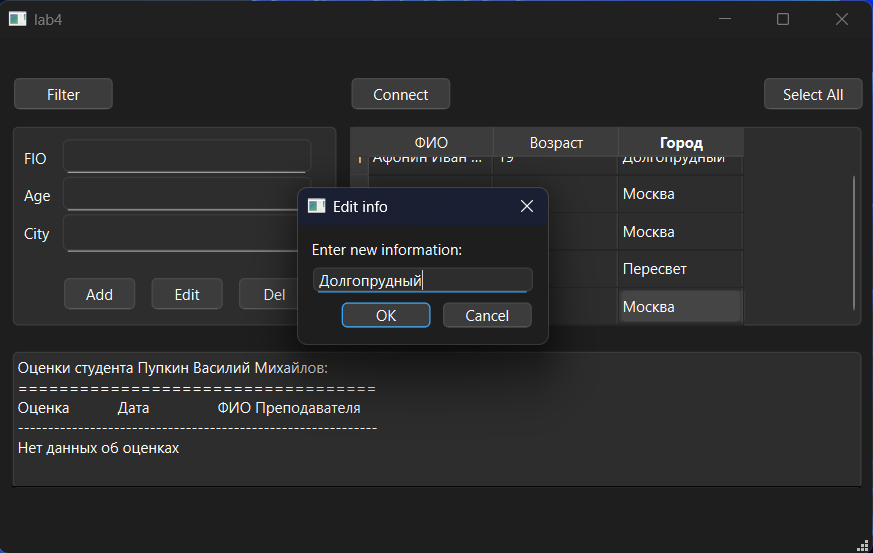


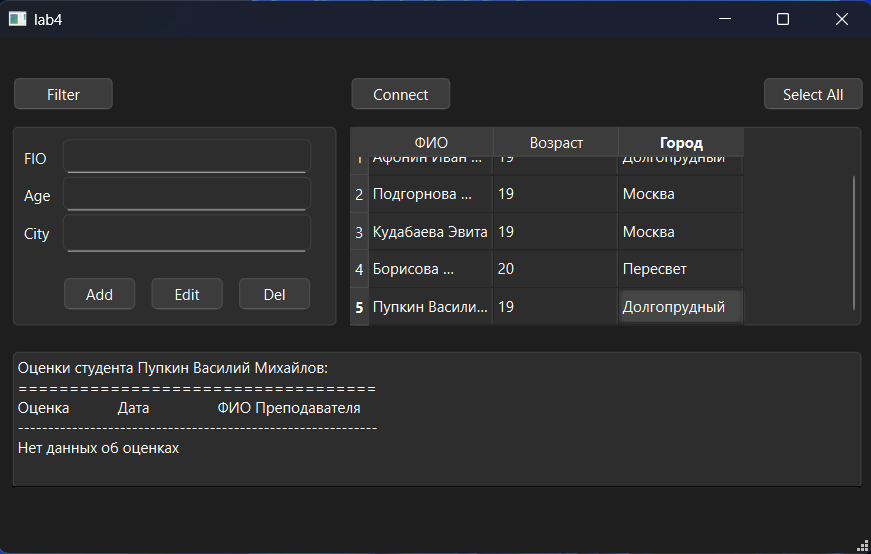
Подключимся к базе данных, нажав на кнопку Connect: 

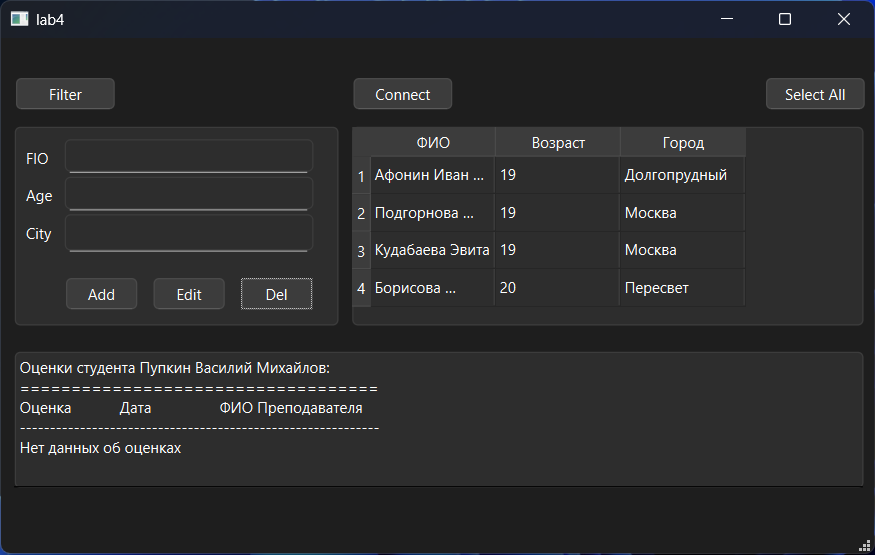
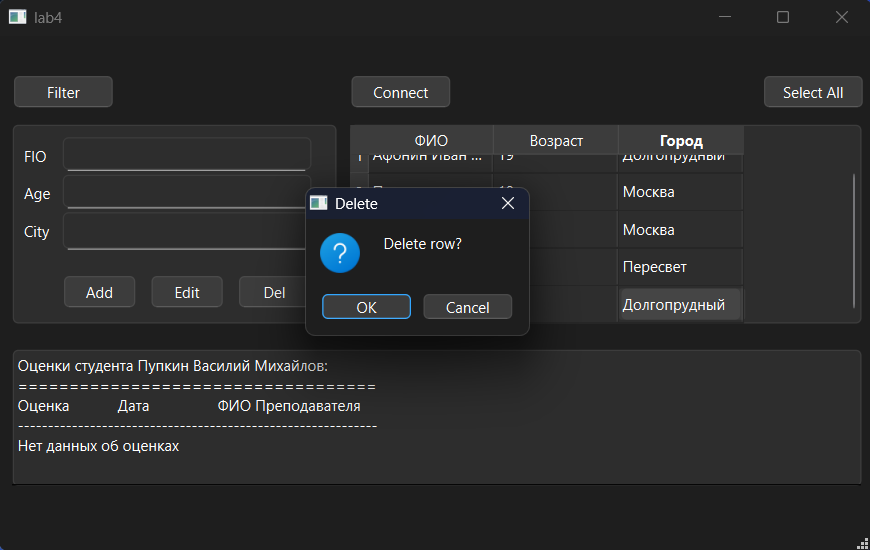
Получим данные из базы данных, нажав на Select All: 

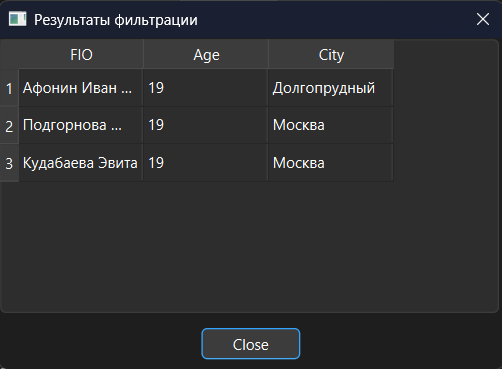
Добавим новую запись: 



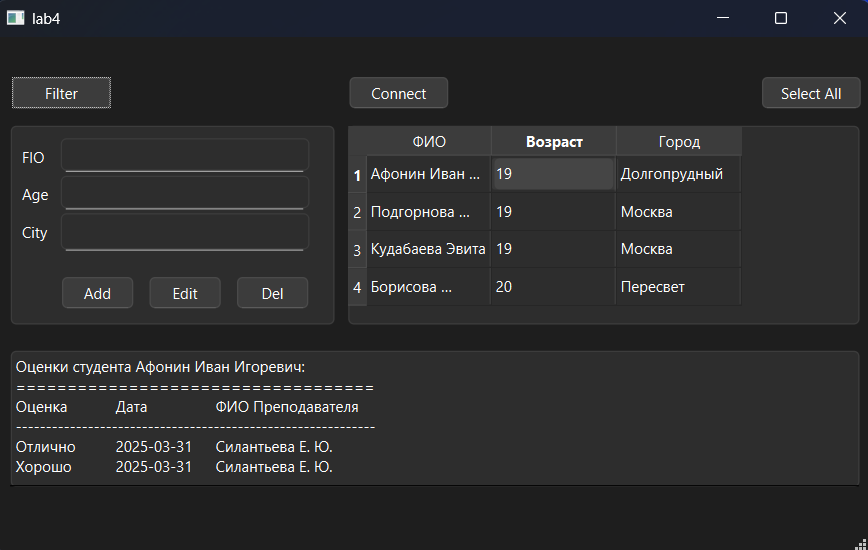
Попробуем отредактировать данные: 



А теперь удалим ее: 

Перейдем к другой форме, выбрав ячейку с числом 19 в столбце Возраст в таблице и нажав кнопку Filter: 

Выберем любого пользователя, нажав на любой столбец в строке с его именем, для вывода информации из связанной таблицы.



Исходный код главной формы:

//mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QSqlDatabase>

#include <QMessageBox>

#include <QSqlQuery>

#include <QDebug>

#include <QInputDialog>

#include "filterwindow.h"

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;

FilterWindow \*filterWindow;

public slots:

void dbconnect();

void selectAll();

void add();

void del();

void edit();

void openFilterWindow();

void showStudentMarks();

};

#endif // MAINWINDOW\_H

//mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QSqlError>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

, filterWindow(nullptr)

{

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

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

connect(ui->btnFilter,SIGNAL(clicked(bool)),this, SLOT(openFilterWindow()));

connect(ui->twOrg, &QTableWidget::itemSelectionChanged, this, &MainWindow::showStudentMarks);

}

void MainWindow::dbconnect()

{

if(!dbconn.isOpen())

{

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

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

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

dbconn.setDatabaseName("dbtest1");

dbconn.setHostName("localhost");

dbconn.setUserName("postgres");

dbconn.setPassword("admin");

if( dbconn.open() )

ui->teResult->append("Connect is open...");

else

{

ui->teResult->append("Error of connect:");

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

}

}

else

ui->teResult->append("Connect is already open...");

}

void MainWindow::selectAll()

{

ui->twOrg->clearContents();

if (!dbconn.isOpen()) {

dbconnect();

if (!dbconn.isOpen()) {

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

return;

}

}

QSqlQuery query(dbconn);

QString sqlstr = "SELECT \* FROM students";

if (!query.exec(sqlstr)) {

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

return;

}

ui->twOrg->setColumnCount(3);

QStringList headers;

headers << "ФИО" << "Возраст" << "Город";

ui->twOrg->setHorizontalHeaderLabels(headers);

if (query.isActive()) {

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

} else {

ui->twOrg->setRowCount(0);

}

int i = 0;

while (query.next()) {

ui->twOrg->setItem(i, 0, new QTableWidgetItem(query.value("fio").toString()));

ui->twOrg->setItem(i, 1, new QTableWidgetItem(query.value("age").toString()));

ui->twOrg->setItem(i, 2, new QTableWidgetItem(query.value("city").toString()));

i++;

}

}

void MainWindow::add()

{

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

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

return;

}

}

QSqlQuery query(dbconn);

QString sqlstr = "insert into students(fio, age, city) values(?,?,?)";

query.prepare(sqlstr);

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

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

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

if( !query.exec() )

{

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

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

return;

}

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

selectAll();

}

void MainWindow::del()

{

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

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

return;

}

}

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

if( currow < 0 )

{

QMessageBox::critical(this,"Error","Not selected row!");

return;

}

if( QMessageBox::question(this,"Delete","Delete row?",

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

return;

QSqlQuery query(dbconn);

QString sqlstr = "delete from students where fio = '"

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

if( !query.exec(sqlstr) )

{

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

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

return;

}

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

selectAll();

}

void MainWindow::edit()

{

if( !dbconn.isOpen() )

{

dbconnect();

if( !dbconn.isOpen() )

{

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

return;

}

}

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

int curcol = ui->twOrg->currentColumn();

if( currow < 0 || curcol < 0)

{

QMessageBox::critical(this,"Error","Not selected row or column!");

return;

}

QString fio = ui->twOrg->item(currow, 0)->text();

QString currentValue = ui->twOrg->item(currow, curcol)->text();

bool ok;

QString newValue = QInputDialog::getText(this, "Edit info",

"Enter new information:",

QLineEdit::Normal,

currentValue, &ok);

if (!ok || newValue.isEmpty() || newValue == currentValue)

return;

QSqlQuery query(dbconn);

QString columnName;

switch(curcol) {

case 0: columnName = "fio"; break;

case 1: columnName = "age"; break;

case 2: columnName = "city"; break;

default: return;

}

QString sqlstr;

if (columnName == "age") {

sqlstr = QString("update students set %1 = %2 where fio = ?").arg(columnName).arg(newValue.toInt());

} else {

sqlstr = QString("update students set %1 = ? where fio = ?").arg(columnName);

}

query.prepare(sqlstr);

if (columnName == "age") {

query.bindValue(0, fio);

} else {

query.bindValue(0, newValue);

query.bindValue(1, fio);

}

if( !query.exec() )

{

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

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

return;

}

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

selectAll();

}

void MainWindow::openFilterWindow()

{

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

if(currow < 0)

{

QMessageBox::critical(this,"Error","No row selected!");

return;

}

int curcol = ui->twOrg->currentColumn();

if(curcol < 0)

{

QMessageBox::critical(this,"Error","No column selected!");

return;

}

QString filterValue = ui->twOrg->item(currow, curcol)->text();

QString columnName;

switch(curcol) {

case 0: columnName = "fio"; break;

case 1: columnName = "age"; break;

case 2: columnName = "city"; break;

default: return;

}

if(!filterWindow)

{

filterWindow = new FilterWindow(this, dbconn);

}

filterWindow->filterData(filterValue, columnName);

filterWindow->show();

}

void MainWindow::showStudentMarks()

{

if (!dbconn.isOpen()) {

return;

}

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

if (currentRow < 0) {

return;

}

QString studentFio = ui->twOrg->item(currentRow, 0)->text();

ui->teResult->clear();

QSqlQuery query(dbconn);

QString sqlstr = "SELECT m.mark\_value, m.mark\_date, m.teacher\_fio "

"FROM marks m "

"JOIN stud\_marks sm ON m.mark\_id = sm.mark\_id "

"JOIN students s ON s.student\_id = sm.student\_id "

"WHERE s.fio = ?";

query.prepare(sqlstr);

query.bindValue(0, studentFio);

if (!query.exec()) {

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

return;

}

ui->teResult->append(QString("Оценки студента %1:").arg(studentFio));

ui->teResult->append("===================================");

ui->teResult->append("Оценка\tДата\tФИО Преподавателя");

ui->teResult->append("------------------------------------------------------------");

while (query.next()) {

QString value = query.value("mark\_value").toString();

QString date = query.value("mark\_date").toString();

QString fio = query.value("teacher\_fio").toString();

ui->teResult->append(QString("%1\t%2\t%3").arg(value, date, fio));

}

if (query.size() == 0) {

ui->teResult->append("Нет данных об оценках");

}

}

MainWindow::~MainWindow()

{

if( dbconn.isOpen())

dbconn.close();

if(filterWindow)

delete filterWindow;

delete ui;

}

//main.cpp

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

a.addLibraryPath(a.applicationDirPath()+"/plugins");

MainWindow w;

w.show();

return a.exec();

}

//mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>698</width>

<height>412</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Minimum" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="windowTitle">

<string>lab4</string>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QPushButton" name="btnSelectAll">

<property name="geometry">

<rect>

<x>610</x>

<y>30</y>

<width>83</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Select All</string>

</property>

</widget>

<widget class="QFrame" name="frame">

<property name="geometry">

<rect>

<x>10</x>

<y>70</y>

<width>261</width>

<height>161</height>

</rect>

</property>

<property name="frameShape">

<enum>QFrame::Shape::StyledPanel</enum>

</property>

<widget class="QPushButton" name="btnEdit">

<property name="geometry">

<rect>

<x>110</x>

<y>120</y>

<width>61</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Edit</string>

</property>

</widget>

<widget class="QPushButton" name="btnAdd">

<property name="geometry">

<rect>

<x>40</x>

<y>120</y>

<width>61</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Add</string>

</property>

</widget>

<widget class="QPushButton" name="btnDel">

<property name="geometry">

<rect>

<x>180</x>

<y>120</y>

<width>61</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Del</string>

</property>

</widget>

<widget class="QLineEdit" name="leFIO">

<property name="geometry">

<rect>

<x>40</x>

<y>10</y>

<width>201</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLineEdit" name="leAge">

<property name="geometry">

<rect>

<x>40</x>

<y>40</y>

<width>201</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLineEdit" name="leCity">

<property name="geometry">

<rect>

<x>40</x>

<y>70</y>

<width>201</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="lbFIO">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>41</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>FIO</string>

</property>

</widget>

<widget class="QLabel" name="lbAge">

<property name="geometry">

<rect>

<x>10</x>

<y>40</y>

<width>41</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Age</string>

</property>

</widget>

<widget class="QLabel" name="lbCity">

<property name="geometry">

<rect>

<x>10</x>

<y>70</y>

<width>41</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>City</string>

</property>

</widget>

</widget>

<widget class="QPushButton" name="btnConnect">

<property name="geometry">

<rect>

<x>280</x>

<y>30</y>

<width>83</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Connect</string>

</property>

</widget>

<widget class="QTableWidget" name="twOrg">

<property name="geometry">

<rect>

<x>280</x>

<y>70</y>

<width>411</width>

<height>161</height>

</rect>

</property>

<property name="rowCount">

<number>5</number>

</property>

<property name="columnCount">

<number>3</number>

</property>

<row/>

<row/>

<row/>

<row/>

<row/>

<column/>

<column/>

<column/>

</widget>

<widget class="QTextEdit" name="teResult">

<property name="geometry">

<rect>

<x>10</x>

<y>250</y>

<width>681</width>

<height>111</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="btnFilter">

<property name="geometry">

<rect>

<x>10</x>

<y>30</y>

<width>83</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Filter</string>

</property>

</widget>

<zorder>frame</zorder>

<zorder>btnSelectAll</zorder>

<zorder>btnConnect</zorder>

<zorder>twOrg</zorder>

<zorder>teResult</zorder>

<zorder>btnFilter</zorder>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>698</width>

<height>25</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

Исходный код второй формы:

//filterwindow.h

#ifndef FILTERWINDOW\_H

#define FILTERWINDOW\_H

#include <QDialog>

#include <QSqlDatabase>

namespace Ui {

class FilterWindow;

}

class FilterWindow : public QDialog

{

Q\_OBJECT

public:

explicit FilterWindow(QWidget \*parent = nullptr, QSqlDatabase db = QSqlDatabase());

~FilterWindow();

void filterData(const QString &filterValue, const QString &column);

private slots:

void onCloseClicked();

private:

Ui::FilterWindow \*ui;

QSqlDatabase dbconn;

};

#endif // FILTERWINDOW\_H

//filterwindow.cpp

#include "filterwindow.h"

#include "ui\_filterwindow.h"

#include <QSqlQuery>

#include <QSqlError>

#include <QMessageBox>

FilterWindow::FilterWindow(QWidget \*parent, QSqlDatabase db) :

QDialog(parent),

ui(new Ui::FilterWindow),

dbconn(db)

{

ui->setupUi(this);

setWindowTitle("Результаты фильтрации");

ui->twFiltered->setHorizontalHeaderLabels({"FIO", "Age", "City"});

connect(ui->btnClose,SIGNAL(clicked(bool)),this, SLOT(onCloseClicked()));

}

void FilterWindow::filterData(const QString &filterValue, const QString &column)

{

ui->twFiltered->clearContents();

if(!dbconn.isOpen())

{

QMessageBox::critical(this,"Error","Database connection is not open");

return;

}

QSqlQuery query(dbconn);

QString sqlstr = QString("SELECT \* FROM students WHERE %1 = ?").arg(column);

query.prepare(sqlstr);

query.bindValue(0, filterValue);

if(!query.exec())

{

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

return;

}

if(query.isActive())

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

else

ui->twFiltered->setRowCount(0);

int i = 0;

while(query.next())

{

ui->twFiltered->setItem(i,0,new QTableWidgetItem(query.value("fio").toString()));

ui->twFiltered->setItem(i,1,new QTableWidgetItem(query.value("age").toString()));

ui->twFiltered->setItem(i,2,new QTableWidgetItem(query.value("city").toString()));

i++;

}

}

void FilterWindow::onCloseClicked()

{

this->close();

}

FilterWindow::~FilterWindow()

{

delete ui;

}

//filterwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>FilterWindow</class>

<widget class="QDialog" name="FilterWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>404</width>

<height>265</height>

</rect>

</property>

<property name="windowTitle">

<string>Dialog</string>

</property>

<widget class="QPushButton" name="btnClose">

<property name="geometry">

<rect>

<x>160</x>

<y>230</y>

<width>83</width>

<height>29</height>

</rect>

</property>

<property name="text">

<string>Close</string>

</property>

</widget>

<widget class="QTableWidget" name="twFiltered">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>401</width>

<height>221</height>

</rect>

</property>

<property name="rowCount">

<number>5</number>

</property>

<property name="columnCount">

<number>3</number>

</property>

<row/>

<row/>

<row/>

<row/>

<row/>

<column/>

<column/>

<column/>

</widget>

</widget>

<resources/>

<connections/>

</ui>

**Выводы по лабораторной работе**

Были получены теоретические и практические навыки создания представлений данных в СУБД PostgreSQL.