**Цель работы**: Научится создавать прикладные программы, работающие с базой данных.

**Задание :**

- Создать полнофункциональное клиентское приложение с GUI для базы данных курсовой работы на одном из языков программирования: Qt С++ , Java или С#.

- Создать клиентские программы для базы данных курсовой работы на двух оставшихся языках программирования. В программах должно быть реализовано:

* просмотр содержимого какой-либо таблицы базы данных;
* Добавление, удаление и редактирование содержимого таблицы базы данных.
* выполнение SQL-запроса с параметрами, хранимой процедуры и триггера.

**Текст программы (C++) :**

***// mainwindow.h***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPushButton>

#include <QLabel>

#include <QLineEdit>

#include <QtSql/QSqlRelationalTableModel>

#include <QtSql/QSqlTableModel>

#include <QTableView>

class MainWindow : public QMainWindow

{

Q\_OBJECT

private:

QSqlDatabase db;

QSqlRelationalTableModel \*model;

QSqlQueryModel \*qmodel;

QTableView \*tableView;

QTableView \*tableView1;

QPushButton \*Button1;

QLabel\* Label1;

QLineEdit\* Edit1;

QMenu \*menuTable, \*menuEdit,\*menuQuery;

private slots:

void slot\_1();

void slot\_2();

void dataEdit1();

void dataEdit2();

void dataEdit3();

void slot\_query1();

void slot\_query2();

void on\_Button1\_clicked();

void on\_Button2\_clicked();

public:

MainWindow(QWidget \*parent = 0);

~*MainWindow*();

};

#endif // MAINWINDOW\_H

***// mainwindow.cpp***

#include "mainwindow.h"

#include <QtSql/QSqlDatabase>

#include <QtSql/QSqlError>

#include <QtSql/QSqlQuery>

#include <QMessageBox>

#include <QVBoxLayout>

#include <QWidget>

#include <QMenu>

#include <QMenuBar>

#include <QHeaderView>

#include <QtSql/QSqlRelationalDelegate>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

{

db=QSqlDatabase::addDatabase("QODBC3"); //("MySQL");

db.setHostName("localhost");

db.setPort(3306);

db.setDatabaseName("mysqlnew"); //("kursovaya\_bd");

db.setUserName("root");

db.setPassword("root");

if (!db.open())

QMessageBox::critical(parent,QObject::tr("Database Error"),db.lastError().text());

else

{ menuTable = new QMenu("Таблицы");

menuTable->addAction("Музыкальная группа", this, SLOT(slot\_1()));

menuTable->addAction("Песня", this, SLOT(slot\_2()));

menuEdit = new QMenu("Правка");

menuEdit->addAction("Добавить", this, SLOT(dataEdit1()));

menuEdit->addAction("Удалить", this, SLOT(dataEdit2()));

menuEdit->addAction("Сохранить изменения", this, SLOT(dataEdit3()));

menuEdit->setEnabled(false);

menuQuery = new QMenu("Запросы");

menuQuery->addAction("Поиск информации о песне с данным названием",

this, SLOT(slot\_query1()));

menuQuery->addAction("Поиск информации об исполнителях по группе",

this, SLOT(slot\_query2()));

menuBar()->addMenu(menuTable);

menuBar()->addMenu(menuEdit);

menuBar()->addMenu(menuQuery);

}

}

MainWindow::~*MainWindow*()

{

}

void MainWindow::slot\_1()

{ model = new QSqlRelationalTableModel();

model->*setTable*("musikalnaya\_gruppa");

model->*setHeaderData*(0, Qt::Horizontal,"Название группы");

model->*setHeaderData*(1, Qt::Horizontal,"Год образования");

model->*setHeaderData*(2, Qt::Horizontal,"Страна");

model->*select*();

model->*setEditStrategy*(QSqlTableModel::OnManualSubmit);

tableView = new QTableView(this);

tableView->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

tableView->*setModel*(model);

QVBoxLayout \*layout = new QVBoxLayout();

layout->addWidget(tableView);

tableView->show();

QWidget \*box = new QWidget();

box->setLayout(layout);

setCentralWidget(box);

menuEdit->setEnabled(true);

}

void MainWindow::slot\_2()

{ model = new QSqlRelationalTableModel();

model->*setTable*("pesnya");

model->*setHeaderData*(0, Qt::Horizontal,"Название песни");

model->*setHeaderData*(1, Qt::Horizontal,"Автор текста");

model->*setHeaderData*(2, Qt::Horizontal,"Композитор");

model->*setHeaderData*(3, Qt::Horizontal,"Дата создания");

model->*setHeaderData*(4, Qt::Horizontal,"Язык песни");

model->*setHeaderData*(5, Qt::Horizontal,"Название группы");

model->*select*();

model->*setEditStrategy*(QSqlTableModel::OnManualSubmit);

tableView = new QTableView(this);

tableView->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

tableView->*setModel*(model);

tableView->setItemDelegate(new QSqlRelationalDelegate(tableView));

QVBoxLayout \*layout = new QVBoxLayout();

layout->addWidget(tableView);

tableView->show();

model->*setEditStrategy*(QSqlTableModel::OnManualSubmit);

QWidget \*box = new QWidget();

box->setLayout(layout);

setCentralWidget(box);

menuEdit->setEnabled(true);

// menuQuery->setEnabled(true);

}

void MainWindow::dataEdit1() //добавить

{ model->insertRow(model->*rowCount*());

QMessageBox::information(0,"CООБЩЕНИЕ","После добавления,необходимо сохранить изменения!!!");

}

void MainWindow::dataEdit2() //удалить

{ if (tableView->currentIndex().isValid())

{ model->removeRow(tableView->currentIndex().row());

model->submitAll();

} else QMessageBox::information(0,"CООБЩЕНИЕ","Выберите удалямую строку!!!");

}

void MainWindow::dataEdit3() //сохранить изменения

{ if (tableView->currentIndex().isValid())

{ if (model->submitAll()==false)

{ QMessageBox::critical(0,"CООБЩЕНИЕ" , "Некорректное действие. \n Данные могут быть потеряны " +

model->lastError().text());

model->revertAll();

}

else QMessageBox::information(0,"CООБЩЕНИЕ","Изменения сохранены");

}

}

void MainWindow::slot\_query1() // Поиск информации о песне с данным названием

{ menuEdit->setEnabled(false);

QVBoxLayout \*layout2 = new QVBoxLayout();

QWidget \*box1 = new QWidget();

Button1 = new QPushButton("Поиск");

Label1 = new QLabel(" Название песни : ");

Edit1 = new QLineEdit();

QHBoxLayout \*Layout1 = new QHBoxLayout();

Layout1->addWidget(Label1);

Layout1->addWidget(Edit1);

Layout1->addWidget(Button1);

layout2->addLayout(Layout1);

tableView1 = new QTableView(this);

tableView1->resizeColumnsToContents();

layout2->addWidget(tableView1);

box1->setLayout(layout2);

setCentralWidget(box1);

connect(Button1, SIGNAL(clicked()), this, SLOT(on\_Button1\_clicked()));

}

void MainWindow::on\_Button1\_clicked()

{ QString name = Edit1->text();

qmodel = new QSqlQueryModel();

qmodel->setQuery("CALL proc1(\'" +name+"\')");

qmodel->setQuery("select Nazvanie\_pesni, Avtor\_teksta, Kompositor,Data\_sozdania, Nazvanie\_gruppi\

from pesnya\

where Nazvanie\_pesni=\'"+name+"\'");

qmodel->*setHeaderData*(0, Qt::Horizontal,"Название песни");

qmodel->*setHeaderData*(1, Qt::Horizontal,"Автор текста");

qmodel->*setHeaderData*(2, Qt::Horizontal,"Композитор");

qmodel->*setHeaderData*(3, Qt::Horizontal,"Дата создания");

qmodel->*setHeaderData*(4, Qt::Horizontal,"Название группы");

tableView1->*setModel*(qmodel);

tableView1->show();

tableView1->resizeColumnsToContents();

}

void MainWindow::slot\_query2()

{ menuEdit->setEnabled(false);

QVBoxLayout \*layout2 = new QVBoxLayout();

QWidget \*box1 = new QWidget();

Button1 = new QPushButton("Поиск");

Label1 = new QLabel("Группа : ");

Edit1 = new QLineEdit();

QHBoxLayout \*Layout1 = new QHBoxLayout();

Layout1->addWidget(Label1);

Layout1->addWidget(Edit1);

Layout1->addWidget(Button1);

layout2->addLayout(Layout1);

tableView1 = new QTableView(this);

layout2->addWidget(tableView1);

box1->setLayout(layout2);

setCentralWidget(box1);

connect(Button1, SIGNAL(clicked()), this, SLOT(on\_Button2\_clicked()));

}

void MainWindow::on\_Button2\_clicked()

{ QString group = Edit1->text();

qmodel = new QSqlQueryModel();

qmodel->setQuery("CALL ispolniteli(\'"+ group +"\ ')");

qmodel->setQuery("select musikalnaya\_gruppa.Nazvanie\_gruppi, ispolnitel.Psevdonim\_Familia, ispolnitel.Vozrast, ispolnitel.Amplua\

from (ispolnitel join musical\_group\_ispolnitel on musical\_group\_ispolnitel.Identifikazionnij\_nomer\_ispolnitelya = ispolnitel.Identifikazionnij\_nomer\_ispolnitelya) join musikalnaya\_gruppa on musical\_group\_ispolnitel.Nazvanie\_gruppi = musikalnaya\_gruppa.Nazvanie\_gruppi\

where musikalnaya\_gruppa.Nazvanie\_gruppi = \'"+ group +"\ '");

qmodel->*setHeaderData*(0, Qt::Horizontal,"Название группы");

qmodel->*setHeaderData*(1, Qt::Horizontal,"Псевдоним");

qmodel->*setHeaderData*(2, Qt::Horizontal,"Возраст");

qmodel->*setHeaderData*(3, Qt::Horizontal,"Амплуа");

tableView1->*setModel*(qmodel);

tableView1->show();

tableView1->resizeColumnsToContents();

}

***// main.cpp***

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

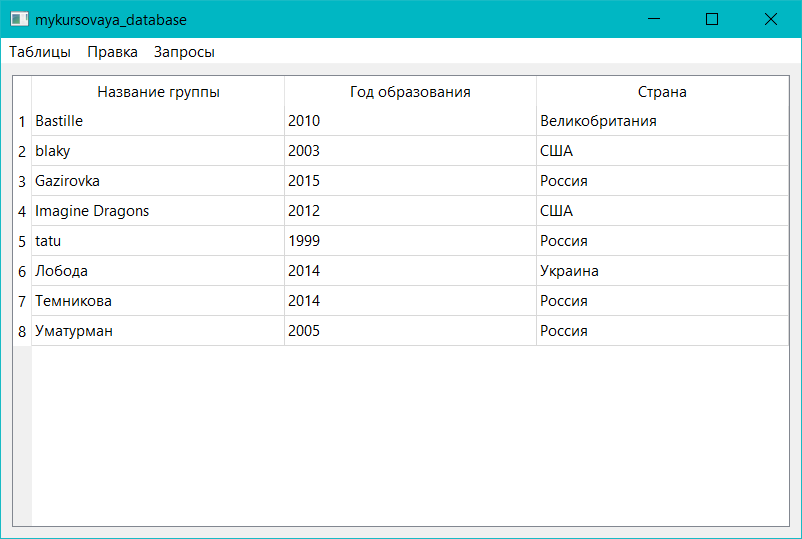
w.resize(800, 500);

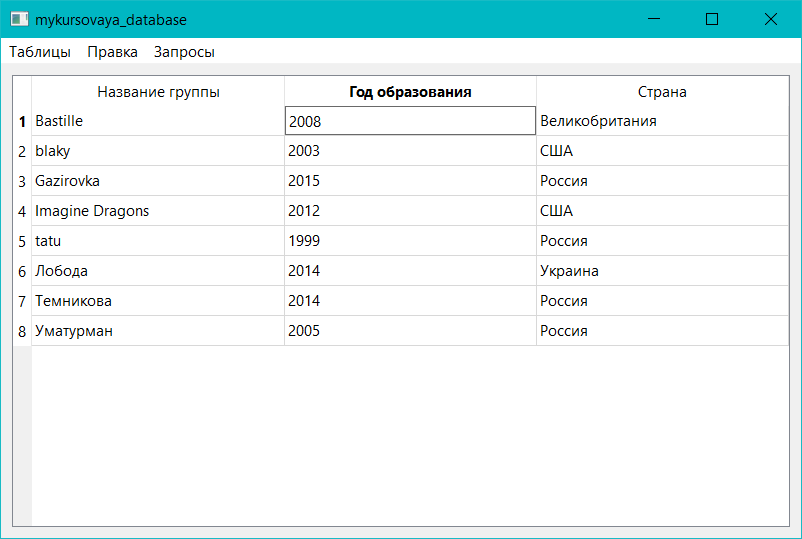
w.show();

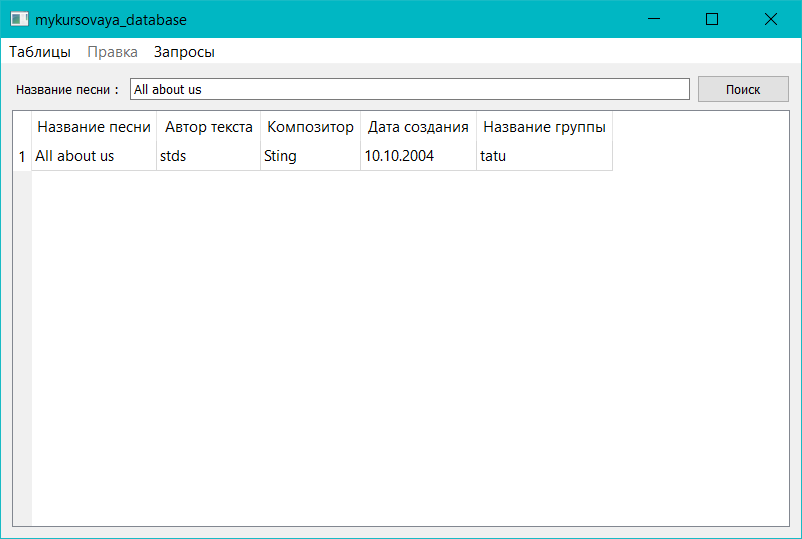
return a.exec();

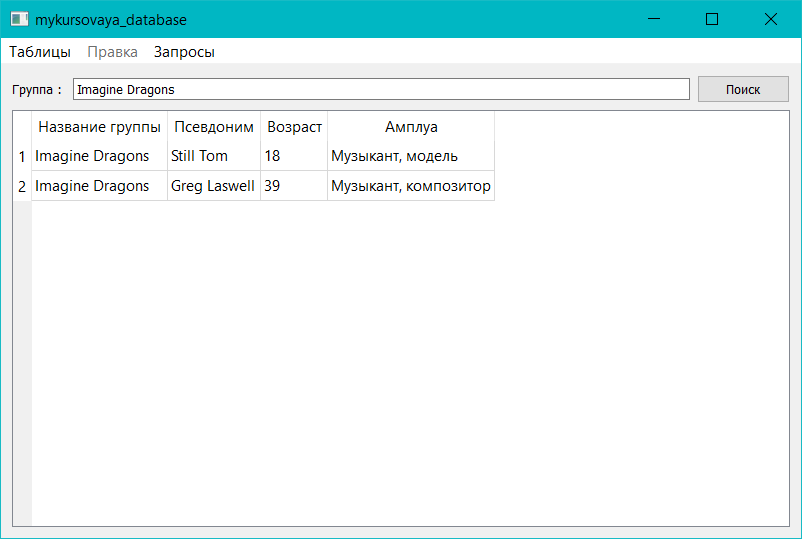
}

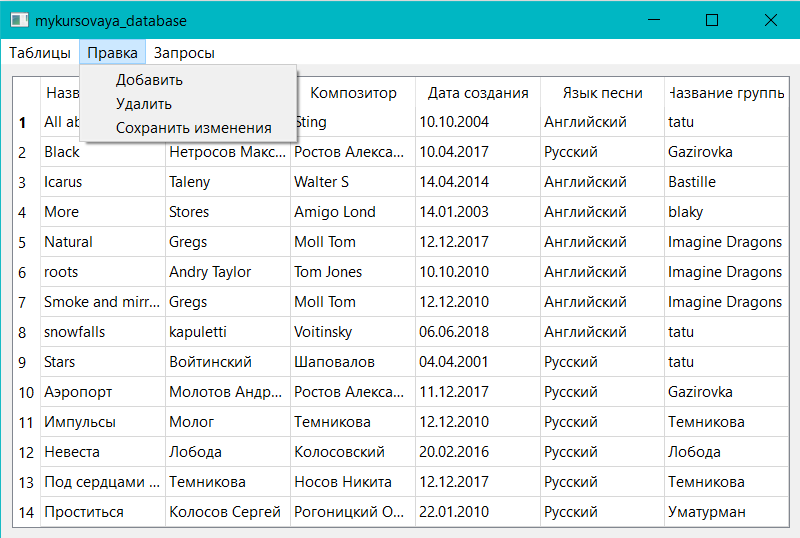
**Тесты (QT) :**

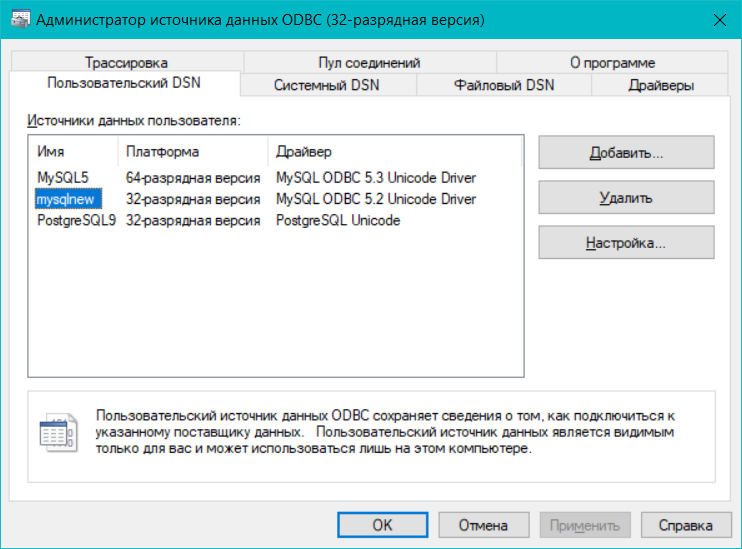












**Текст программы (Java) :**

package empl;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.sql.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Vector;

public class Emplw implements ActionListener

{

JFrame frame;

JScrollPane scrollTable, paneQuery;

JPanel panel;

JTable table;

JButton buttonQ;

ActionListener bListener;

JLabel labelQ;

JTextField textQ;

DefaultTableModel model;

boolean isConnect, isAdded;

String driverName = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/kursovaya\_bd";

String userName = "root";

String password = "root";

Connection con;

PreparedStatement ps;

Statement stm;

ResultSet rs;

//Menu

JMenuBar menu1 = new JMenuBar();

JMenu mTables = new JMenu("Таблицы");

JMenu mOperations = new JMenu("Правка");

JMenu mQuerys = new JMenu("Запросы");

JMenuItem item1\_1 = new JMenuItem("Музыкальные группы");

JMenuItem item1\_2 = new JMenuItem("Песня");

JMenuItem item2\_1 = new JMenuItem("Добавить");

JMenuItem item2\_2 = new JMenuItem("Удалить");

JMenuItem item2\_3 = new JMenuItem("Сохранить изменения");

JMenuItem item3\_1 = new JMenuItem("Поиск информации о песне с данным названием");

JMenuItem item3\_2 = new JMenuItem("Состав исполнителей группы с заданным названием, их возраст и амплуа");

public void createUI()

{

//menu

menu1.add(mTables);

menu1.add(mOperations);

menu1.add(mQuerys);

mTables.add(item1\_1);

mTables.add(item1\_2);

mOperations.add(item2\_1);

mOperations.add(item2\_2);

mOperations.add(item2\_3);

mOperations.setEnabled(false);

mQuerys.add(item3\_1);

mQuerys.add(item3\_2);

item1\_1.addActionListener(this);

item1\_2.addActionListener(this);

item2\_1.addActionListener(this);

item2\_2.addActionListener(this);

item2\_3.addActionListener(this);

item3\_1.addActionListener(this);

item3\_2.addActionListener(this);

//frame

frame = new JFrame("Database empl");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(null);

frame.setVisible(true);

frame.setSize(550, 350);

frame.setJMenuBar(menu1);

//base

table = new JTable();

table.setCellSelectionEnabled(true);

scrollTable = new JScrollPane(table);

scrollTable.setBounds(0, 0, 535, 290);

frame.add(scrollTable);

paneQuery = new JScrollPane();

paneQuery.setBounds(0, 0, 535, 50);

paneQuery.hide();

frame.add(paneQuery);

try

{

con = DriverManager.getConnection(url, userName, password);

stm = con.createStatement(

ResultSet.TYPE\_FORWARD\_ONLY,

ResultSet.CONCUR\_UPDATABLE);

isConnect = true;

}

catch(Exception ex)

{

isConnect = false;

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

//query

buttonQ = new JButton("Поиск");

buttonQ.setBounds(430,15,90,25);

paneQuery.add(buttonQ);

labelQ = new JLabel("");

labelQ.setBounds(20,15,150,25);

paneQuery.add(labelQ);

textQ = new JTextField();

textQ.setBounds(180,15,240,26);

paneQuery.add(textQ);

}

public void actionPerformed(ActionEvent ae)

{

if(isConnect)

{

scrollTable.setBounds(0, 0, 535, 290);

paneQuery.hide();

if(ae.getSource() == item1\_1) showTable1();

if(ae.getSource() == item1\_2) showTable2();

if(ae.getSource() == item2\_1) addRow();

if(ae.getSource() == item2\_2) delRow();

if(ae.getSource() == item2\_3) saveTable();

if(ae.getSource() == item3\_1) query1();

if(ae.getSource() == item3\_2) query2();

}

}

public void showTable1()

{

isAdded = false;

String[] columnNames = {"Название группы", "Год образования", "Страна"};

model = new DefaultTableModel();

model.setColumnIdentifiers(columnNames);

table.setModel(model);

try

{

String sql = "select \* from musikalnaya\_gruppa";

rs = stm.executeQuery(sql);

int i =0;

while(rs.next())

{

model.addRow(new Object[]{

rs.getString("Nazvanie\_gruppi"),

rs.getInt("God\_obrazovaniya"),

rs.getString("Strana"),

});

i++;

}

if(i <1)

JOptionPane.showMessageDialog(null, "Записей не найдено","Error", JOptionPane.ERROR\_MESSAGE);

mOperations.setEnabled(true);

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

public void showTable2()

{

isAdded = false;

String[] columnNames = {"Название песни", "Автор текста", "Композитор", "Дата создания", "Язщык песни","Название группы"};

model = new DefaultTableModel();

model.setColumnIdentifiers(columnNames);

table.setModel(model);

try

{

String sql = "select \* from pesnya";

rs = stm.executeQuery(sql);

int i =0;

while(rs.next())

{

model.addRow(new Object[]{

rs.getString("Nazvanie\_pesni"),

rs.getString("Avtor\_teksta"),

rs.getString("Kompositor"),

rs.getDate("Data\_sozdania"),

rs.getString("Yazik\_pesni"),

rs.getString("Nazvanie\_gruppi"),});

i++;

}

if(i <1)

JOptionPane.showMessageDialog(null, "Записей не найдено","Error", JOptionPane.ERROR\_MESSAGE);

mOperations.setEnabled(true);

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

public void addRow()

{

if (isAdded == false)

{

isAdded = true;

model.addRow(new Vector());

JOptionPane.showMessageDialog(null, "После добавления необходимо сохранить изменения!","Внимание", JOptionPane.INFORMATION\_MESSAGE);

}

}

public void delRow()

{

try

{

int actRow = table.getSelectedRow();

String sql = "SELECT \* from " + rs.getMetaData().getTableName(1);

rs.absolute(actRow+1);

rs.deleteRow();

rs.close();

model.removeRow(actRow);

rs = stm.executeQuery(sql);

JOptionPane.showMessageDialog(null, "Удалено!","", JOptionPane.INFORMATION\_MESSAGE);

}

catch (SQLException ex)

{

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

public void saveTable()

{

try

{

int i;

String sql = "SELECT \* from " + rs.getMetaData().getTableName(1);

int rowCnt = model.getRowCount();

for(i=0; i<rowCnt; i++)

{

if(isAdded && (i==rowCnt-1)) rs.moveToInsertRow();

else rs.absolute(i+1);

for(int j=0; j<model.getColumnCount(); j++) {

if( model.getValueAt(i, j)== null)

rs.updateNull(j+1);

else

{ int type = rs.getMetaData().getColumnType(j+1);

if (type == Types.INTEGER)

rs.updateInt(j+1, Integer.parseInt(model.getValueAt(i, j).toString()));

else if (type == Types.CHAR || type == Types.VARCHAR)

rs.updateString(j+1, model.getValueAt(i, j).toString());

else if (type == Types.FLOAT || type == Types.DOUBLE)

rs.updateDouble(j+1, Double.parseDouble(model.getValueAt(i, j).toString()));

else if (type == Types.BOOLEAN)

rs.updateBoolean(j+1, Boolean.parseBoolean(model.getValueAt(i, j).toString()));

else if (type == Types.DATE)

rs.updateDate(j+1, Date.valueOf(model.getValueAt(i, j).toString()));

}

}

if(isAdded && (i==rowCnt-1)) rs.insertRow();

else rs.updateRow();

}

rs.close();

rs = stm.executeQuery(sql);

JOptionPane.showMessageDialog(null, "Сохранено!","", JOptionPane.INFORMATION\_MESSAGE);

isAdded = false;

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(null,

ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

public void query1()

{

labelQ.setText("Название песни :");

scrollTable.setBounds(0, 50, 535, 240);

paneQuery.show();

mOperations.setEnabled(false);

model = new DefaultTableModel();

table.setModel(model);

buttonQ.removeActionListener(bListener);

bListener = new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

String[] columnNames = {"Название песни", "Автор текста", "Композитор", "Дата создания", "Название группы"};

model = new DefaultTableModel();

model.setColumnIdentifiers(columnNames);

table.setModel(model);

try {

String sql = "SELECT Nazvanie\_pesni, Avtor\_teksta, Kompositor, Data\_sozdania, Nazvanie\_gruppi FROM pesnya "

+ "WHERE Nazvanie\_pesni = \'" + textQ.getText() + "\' ";

rs = stm.executeQuery(sql);

int i =0;

while(rs.next())

{

model.addRow(new Object[]{

rs.getString("Nazvanie\_pesni"),

rs.getString("Avtor\_teksta"),

rs.getString("Kompositor"),

rs.getDate("Data\_sozdania"),

rs.getString("Nazvanie\_gruppi"),});

i++;

}

if(i <1)

JOptionPane.showMessageDialog(null, "Записей не найдено","Error", JOptionPane.ERROR\_MESSAGE);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

};

buttonQ.addActionListener(bListener);

}

public void query2()

{

labelQ.setText("Группа :");

scrollTable.setBounds(0, 50, 535, 240);

paneQuery.show();

mOperations.setEnabled(false);

model = new DefaultTableModel();

table.setModel(model);

buttonQ.removeActionListener(bListener);

bListener = new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

String[] columnNames = {"Название группы", "Псевдоним", "Возраст", "Амплуа"};

model = new DefaultTableModel();

model.setColumnIdentifiers(columnNames);

table.setModel(model);

try {

String sql = (" CALL ispolniteli (\'"+textQ.getText()+"\')");

rs = stm.executeQuery(sql);

int i =0;

while(rs.next())

{

model.addRow(new Object[]{

rs.getString("Nazvanie\_gruppi"),

rs.getString("Psevdonim\_Familia"),

rs.getString("Vozrast"),

rs.getString("Amplua"),

});

i++;

}

if(i <1)

JOptionPane.showMessageDialog(null, "Записей не найдено","Error", JOptionPane.ERROR\_MESSAGE);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage(),"Error", JOptionPane.ERROR\_MESSAGE);

}

}

};

buttonQ.addActionListener(bListener);

}

public static void main(String[] args) {

Emplw sr = new Emplw();

sr.createUI();

}

}

**Тесты (NetBeans) :**

