Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Кафедра 402
«Радиосистемы управления и передачи информации»

Расчетно-графическая работа по дисциплине “Методы защищенного доступа к распределенным информационным ресурсам”

Выполнил: студент группы 04-421 _____ В.Ю. Прокашев

Проверил: старший преподаватель кафедры 402 _____ О. С. Красильникова

Москва 2012 г.

Содержание

Задание

Создание реляционной базы данных в MS SQL Server 2008 содержащей не менее 3 таблиц, связанных отношением «один ко многим» и «многие-ко-многим», все таблицы должны быть заполнены реальными, осмысленными данными.

Далее следует создать программу - интерфейс к этой базе данных в виде Windows-приложения на платформе .Net в среде программирования Visual Studio 2008. Приложение должно предоставлять возможность выводить на экран все данные из одной или нескольких таблиц или нескольких столбцов по определённым критериям, сохранять и загружать выведенные таблицы из базы данных. Кроме того позволять модифицировать данные в БД (добавление, изменение, удаление).

Текстовое описание базы данных и её схема

Созданная в процессе выполнения расчётной работы база данных PharmacyBase состоит из 5 таблиц:

1. Medicaments – таблица лекарственных средств с названием лекарственного средства - *title*, номером вида – *id_kind*, номером фармгруппы – *id_group*, объемом партии - *volume*, средней ценой *average_price* и порядковым номером лекарственного средства – *id_medicament*. Содержит 4415 реальных лекарственных средств;
2. Pharmacies – таблица аптек с названием аптеки - *name*, адресом - *address*, телефоном – *phone* и порядковым номером аптеки – *id_pharmacy*. Содержит 839 записей о реальных аптеках;
3. Pharmgroups – таблица сопоставления номера фарм группы - *id_group* и её названия – *name_of_group*. Всего 12 видов фармгрупп препаратов;
4. Pharmkinds - таблица сопоставления номера вида лекарства - *id_kind* и названия вида - *name_of_kind*. Всего 10 видов препаратов;
5. Indexes – таблица сопоставления наличия лекарств в аптеках в виде столбцов *id_pharmacy* и *id_medicament*. Содержит 61810 записей.

Таблицы Medicaments и Pharmgroups, Medicaments и Pharmkinds связаны отношением один ко многим. Одна фармгруппа/вид может содержать несколько лекарственных средств, а одно лекарственное средство принадлежит одной фармгруппе. Кроме того, реализовано каскадное связывание фармгрупп/видов и лекарственных средств.

Таблицы Medicaments, Pharmacies связаны отношением «многие-ко-многим». Один препарат может содержаться в нескольких аптеках, а одна аптека может реализовывать много препаратов. Кроме того, реализовано каскадное связывание таблиц Pharmacies, Indexes и Medicaments, Indexes.

Схема базы данных PharmacyBase показана на рисунке 1.

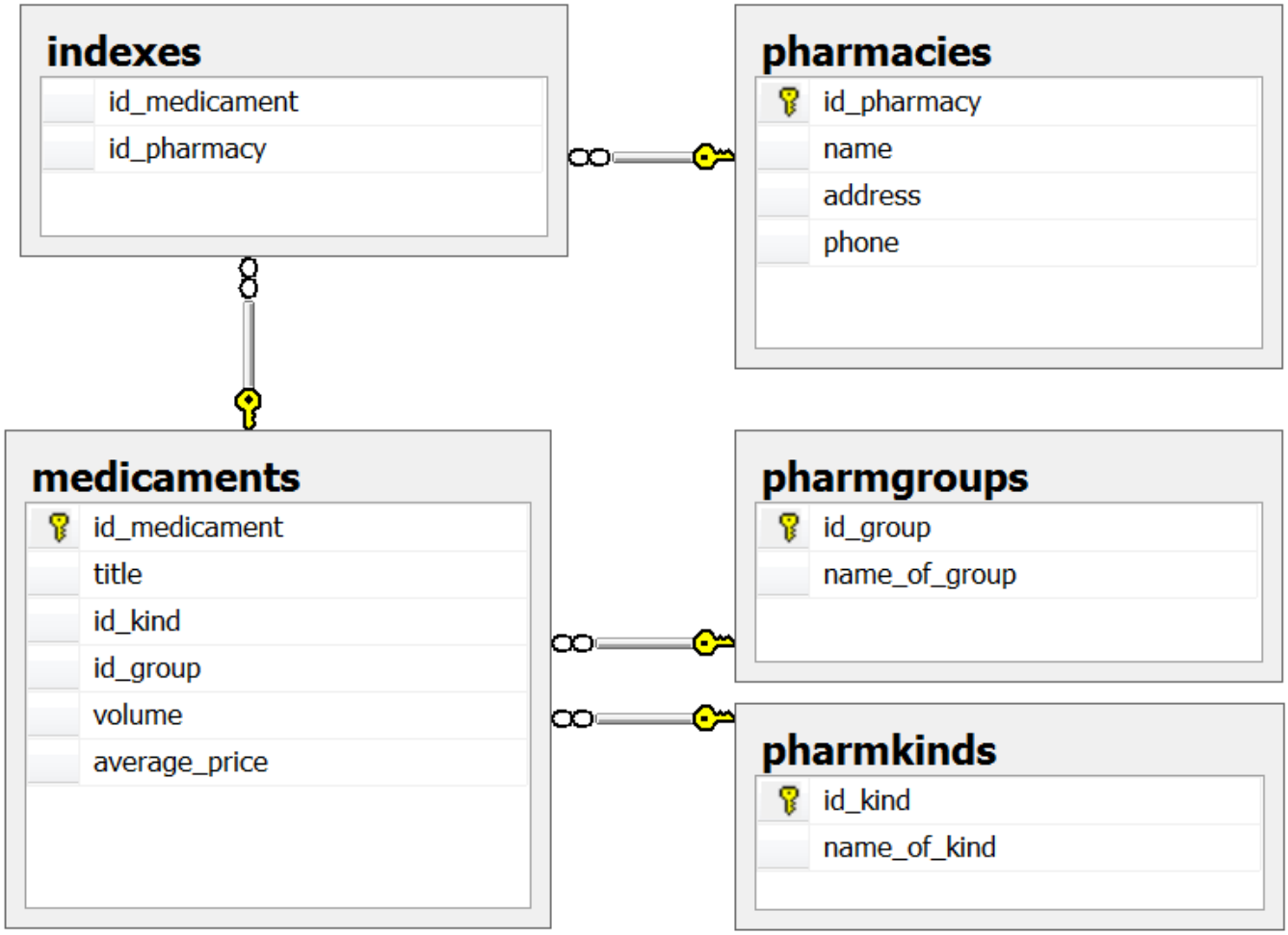
indexes
id_medicament
id_pharmacy
pharmacies
id_pharmacy
name
address
phone
medicaments
id_medicament
title
id_kind
id_group
volume
average_price
pharmgroups
id_group
name_of_group
pharmkinds
id_kind
name_of_kind

Рисунок 1.

Краткое текстовое описание программы

Приложение имеет две вкладки: вкладка отображения запросов и вкладка редактирования таблиц.

Во вкладке отображения запросов пользователь может выбрать интересующие столбцы и условия, накладываемые на них. Условия содержат операции: больше, меньше, равно, найти. Отсутствие выбранного условия трактуется программой как отсутствие условия для значения столбца.

Если пользователь выбирает столбцы, связанные с препаратом и с аптекой, то запрос к базе данных формируется с условием наличия препарата в данной аптеке.

Режим редактирования работает для каждой таблицы по отдельности при выборе отдельной таблицы. Изменения переносятся в базу данных при нажатии кнопки «Сохранить изменения» и осуществляются с помощью метода *DataTable.GetChanges*().

Сохранение и загрузка таблиц в текстовый XML документ осуществляется нажатиями кнопок «Сохранить таблицу в XML файл» и «Загрузить таблицу из XML файла». После нажатия на эти кнопки появляется диалог (*saveFileDialog*/*openFileDialog*)для выбора места и названия файла.

Размеры отображений таблиц изменяются динамически и связаны с изменением размера главного окна программы. Данное свойство элементов реализовано с помощью параметра *Dock*.

Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data.Sql;
using System.Collections;

namespace PharmacyBaseClient
{
    public partial class Form1 : Form
    {
        Form2 loginForm = new Form2();
        public Form1()
        {
            InitializeComponent();
        }

        DataSet sendRequest(string request) //Запрос к базе данных
        {
            DataSet buffer = new DataSet();
            if (loginForm.Text[0] != 'П')
            {
                SqlConnection connection = new SqlConnection(loginForm.Text);
                try
                {
                    if (request.Length != 0)
                    {
                        connection.Open();
                        SqlDataAdapter adapter = new SqlDataAdapter(request, connection);
                        SqlCommandBuilder command = new SqlCommandBuilder(adapter);
                        adapter.Fill(buffer);
                    }
                    else
                    {
                        MessageBox.Show("Нет запроса");
                        buffer = null;
                    }
                }
                catch (SqlException ex)
                {
                    MessageBox.Show(ex.ToString());
                }
                finally
                {
                    connection.Close();
                }
            }
            else
            {
                MessageBox.Show("Плохие параметры подключения");
                buffer = null;
            }
            return buffer;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            loginForm.ShowDialog();//Вызов настроек соединения
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            loginForm.ShowDialog();//Вызов настроек соединения
        }
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            loginForm.ShowDialog();//Вызов настроек соединения
        }
        private void button2_Click(object sender, EventArgs e)
        {
            //Запрос столбцов из таблицы по списку
            List<string> tablesList = new List<string>();
            List<string> columsList = new List<string>();
            List<string> conditionList = new List<string>();
            string colums = "";
            string tables = "";
            string condition = "";
            for (int i = 0; i < dataGridView3.Rows.Count; i++)
            {
                // Разбор списка запрашиваемых столбцов и таблиц
                if (dataGridView3.Rows[i].Cells[0].Value != null)
                {
                    if
(!tablesList.Contains((dataGridView3.Rows[i].Cells[0].Value.ToString().Split('.'))[0]))
                    {

tablesList.Add((dataGridView3.Rows[i].Cells[0].Value.ToString().Split('.'))[0]);
                    }
                    if
(!columsList.Contains(dataGridView3.Rows[i].Cells[0].Value.ToString()))
                    {
                        columsList.Add(dataGridView3.Rows[i].Cells[0].Value.ToString());
                    }
                    if (dataGridView3.Rows[i].Cells[1].Value != null &&
dataGridView3.Rows[i].Cells[2].Value != null)
                    {
                        if (dataGridView3.Rows[i].Cells[1].Value.ToString() == "Найти")
                        {

conditionList.Add(dataGridView3.Rows[i].Cells[0].Value.ToString() + " LIKE '%" +
dataGridView3.Rows[i].Cells[2].Value.ToString() + "%'");
                        }
                        else
                        {
                            int a;
                            if
(int.TryParse(dataGridView3.Rows[i].Cells[2].Value.ToString(), out a))
                            {

conditionList.Add(dataGridView3.Rows[i].Cells[0].Value.ToString() +
dataGridView3.Rows[i].Cells[1].Value.ToString() +
dataGridView3.Rows[i].Cells[2].Value.ToString());
                            }
                            else
                            {
                                MessageBox.Show("Значение не число");
                            }
                        }
                    }
                }
            }
            //Формирование требуемые условий для таблиц
            if (columsList.Contains("pharmgroups.name_of_group"))
            {
                if (!tablesList.Contains("medicaments"))
                {
                    tablesList.Add("medicaments");
                }
                conditionList.Add("medicaments.id_group = pharmgroups.id_group");
            }
            if (columsList.Contains("pharmkinds.name_of_kind"))
            {
                if (!tablesList.Contains("medicaments"))
                {
```

```
                tablesList.Add("medicaments");
            }
            conditionList.Add("medicaments.id_kind = pharmkinds.id_kind");
        }
        if (tablesList.Contains("medicaments") && tablesList.Contains("pharmacies"))
        {
            tablesList.Add("indexes");
            conditionList.Add("medicaments.id_medicament=indexes.id_medicament AND
indexes.id_pharmacy = pharmacies.id_pharmacy");
        }
        //Формирование текста запроса
        if (columsList.Count != 0)
        {
            colums = "SELECT";
            tables = " FROM";
            for (int i = 0; i < columsList.Count; i++)
            {
                colums = colums + " " + columsList[i];
                if (i != (columsList.Count - 1))
                {
                    colums = colums + ", ";
                }
            }
            for (int i = 0; i < tablesList.Count; i++)
            {
                tables = tables + " " + tablesList[i];
                if (i != (tablesList.Count - 1))
                {
                    tables = tables + ", ";
                }
            }
        }
        if (conditionList.Count != 0)
        {
            condition = " WHERE ";
            for (int i = 0; i < conditionList.Count; i++)
            {
                condition = condition + conditionList[i];
                if (i != (conditionList.Count - 1))
                {
                    condition = condition + " AND ";
                }
            }
        }
        string request = colums + tables + condition;
        //Вывод таблицы главное окно
        DataSet buffer = sendRequest(request);
        if (buffer != null)
        {
            dataGridView1.DataSource = buffer.Tables[0];
        }
    }
    private void button11_Click(object sender, EventArgs e)
    {
        //Добавление столбца для запроса на отображение
        dataGridView3.Rows.Add(listBox1.SelectedItem);
    }
    private void button12_Click(object sender, EventArgs e)
    {
        //Удаление столбца из запроса на отображение
        if (dataGridView3.CurrentCell != null)
        {
            dataGridView3.Rows.RemoveAt(dataGridView3.CurrentCell.RowIndex);
        }
    }
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            //Сохранение таблицы в файл
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                if (dataGridView1.DataSource != null)
                {
              ((DataTable)dataGridView1.DataSource).WriteXml(saveFileDialog1.FileName);
                }
                else
                {
                    MessageBox.Show("Нет данных для сохранения");
                }
            }
        }
        private void button7_Click(object sender, EventArgs e)
        {
            //Загрузка таблицы из файла
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                DataSet buffer = new DataSet();
                try
                {
                    buffer.ReadXml(openFileDialog1.FileName);
                    dataGridView1.DataSource = buffer.Tables[0];
                }
                catch
                {
                    MessageBox.Show("Ошибка чтения");
                }
            }
        }
        private void button5_Click(object sender, EventArgs e)
        {
            //Запрос вывода редактируемой таблицы
            if (comboBox1.SelectedItem != null)
            {
                button6.Enabled = true;
                DataSet buffer = new DataSet();

                buffer = sendRequest("SELECT * FROM " +
comboBox1.SelectedItem.ToString());

                dataGridView2.DataSource = buffer.Tables[0];
            }
        }
        private void button6_Click(object sender, EventArgs e)
        {
            //Сохранение изменений
            DataTable table = (DataTable)dataGridView2.DataSource;
            DataTable delta = table.GetChanges();
            if (delta!=null)
            {
                SqlConnection connection = new SqlConnection(loginForm.Text);
                connection.Open();

                SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM " +
comboBox1.SelectedItem.ToString(), connection);

                SqlCommandBuilder command = new SqlCommandBuilder(adapter);
                adapter.Update(delta);
                DataSet buffer = new DataSet();
                //Обновление отображения таблицы

                            buffer = sendRequest("SELECT * FROM " +
             comboBox1.SelectedItem.ToString());

                dataGridView2.DataSource = buffer.Tables[0];
            }
        }
```

```
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            //Блокировка изменения при изменении редактируемой таблицы
            button6.Enabled = false;
        }
        private void Form1_Resize(object sender, EventArgs e)
        {
            //Изменение размеров отображений таблиц
            dataGridView1.Height = this.Size.Height - 286;
            dataGridView2.Width = this.Size.Width - 130;
        }
    }
}
```

Результат выполнения

Основное окно программы представлено на рисунке 2.

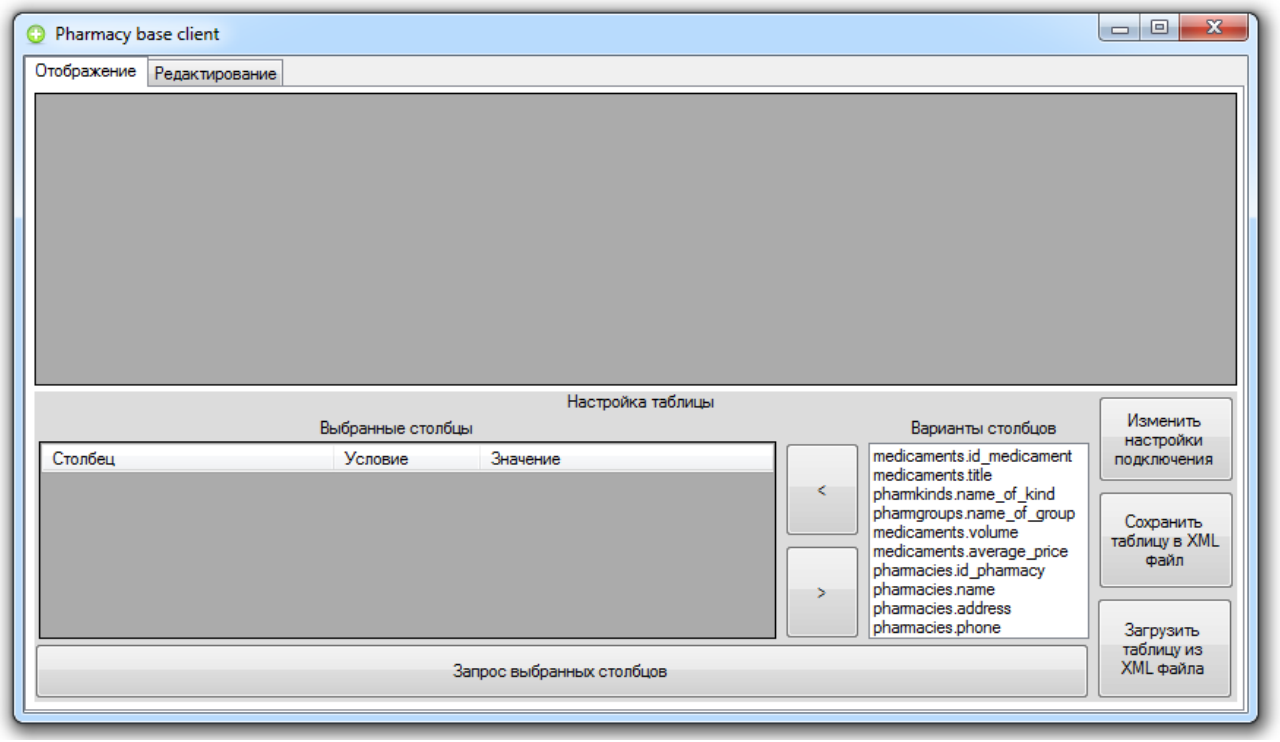

Рисунок 2.

Основное окно программы после выполнения запроса и изменения размера окна показано на рисунке 3.

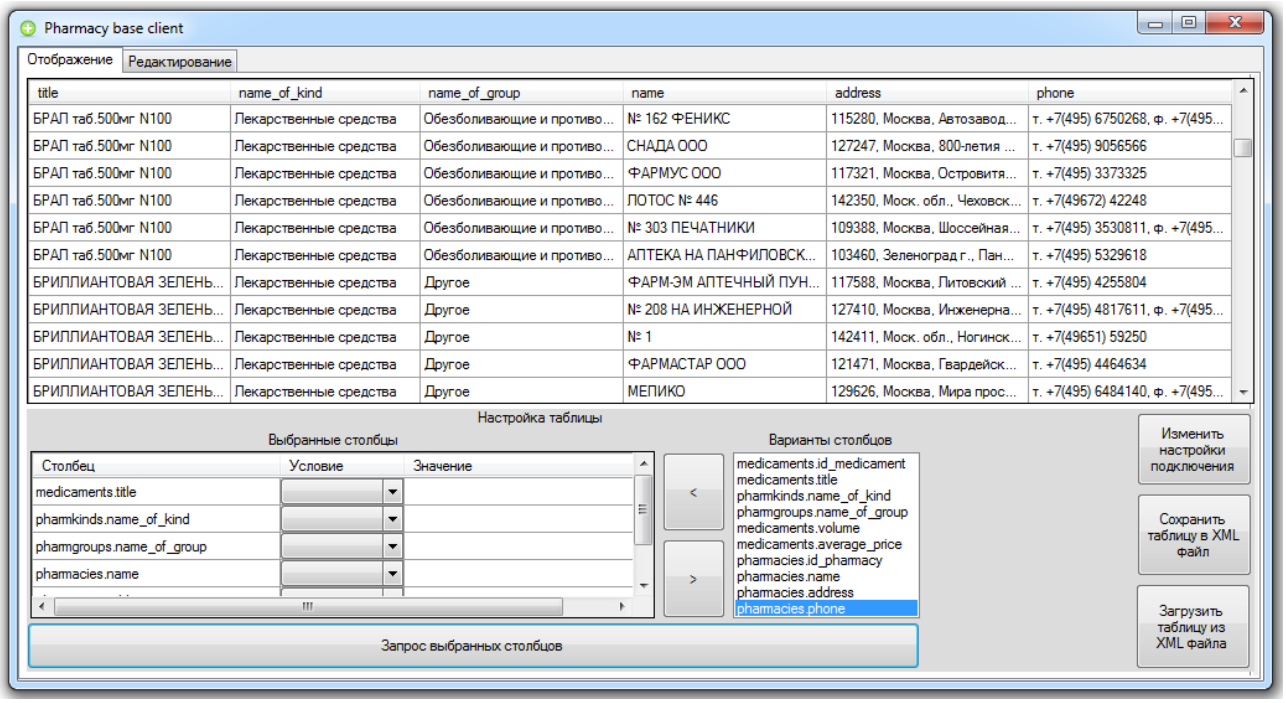

Рисунок 3.

Основное окно программы при поиске боярышника в названии лекарственного средства показано на рисунке 4.

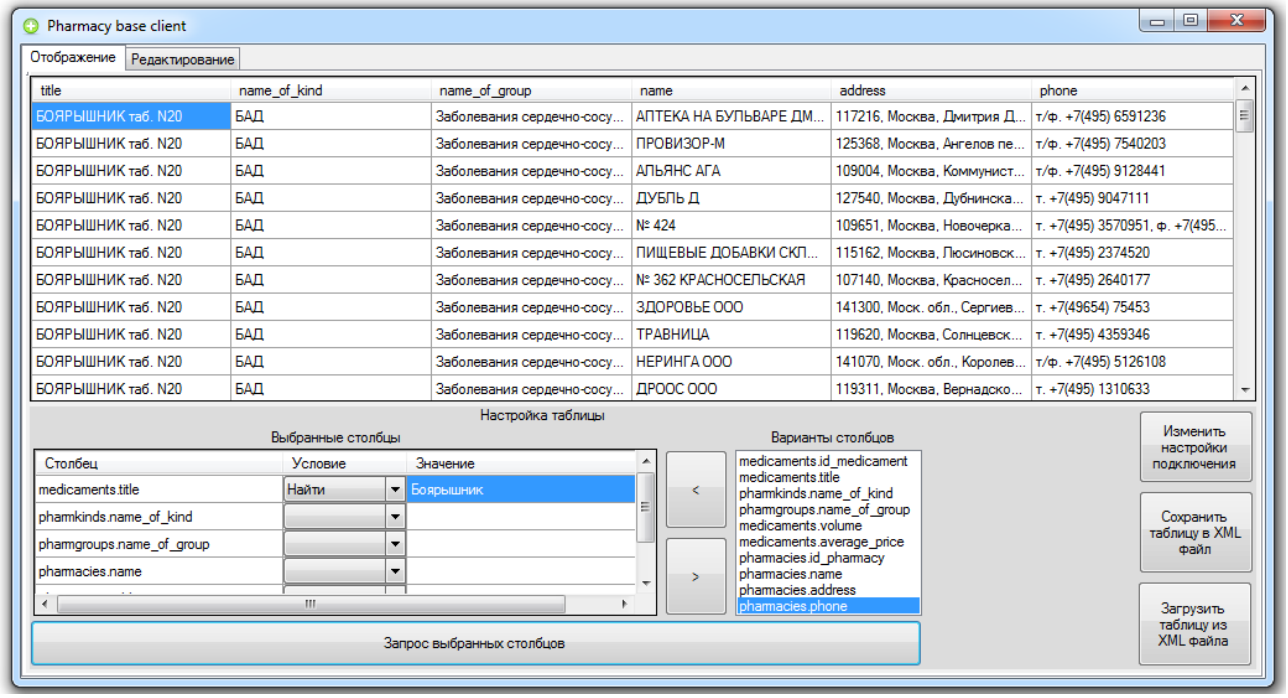

Рисунок 4.

Основное окно программы при запросе лекарственных средств, ценой до 470 рублей показано на рисунке 5.

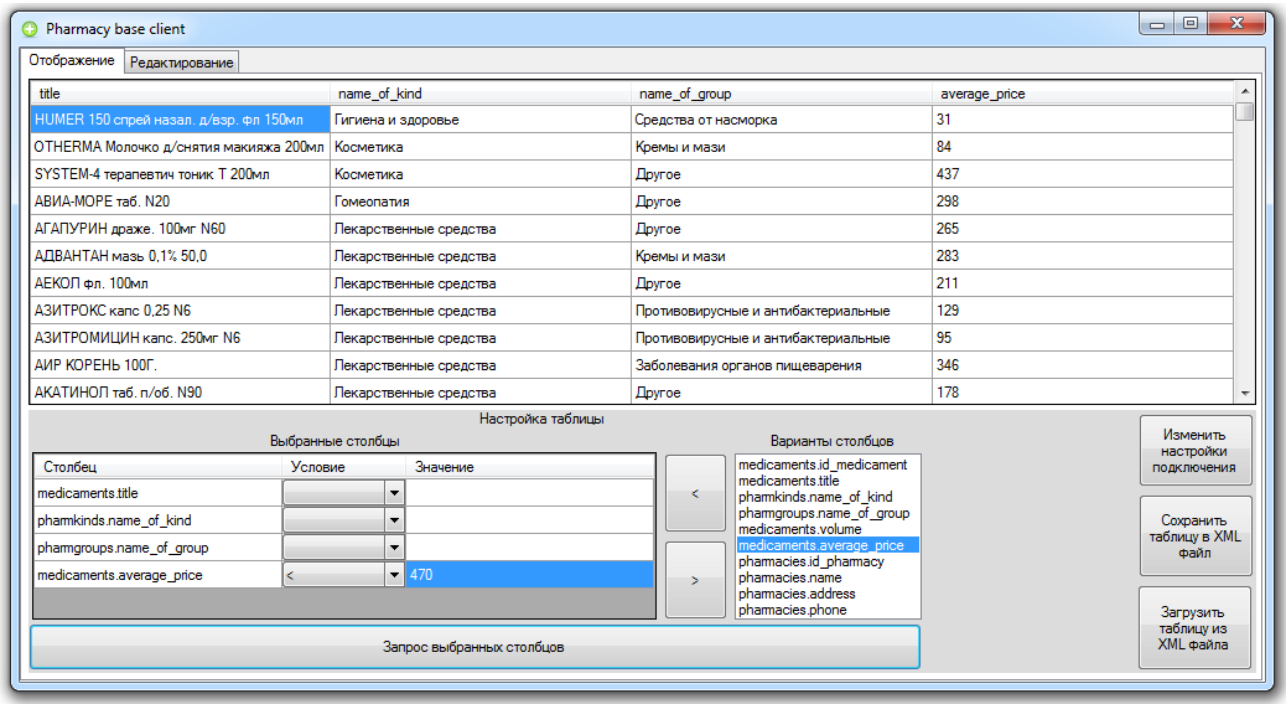

Рисунок 5.

Программа в режиме редактирования при выборе таблицы с информацией об аптеках показана на рисунке 6.

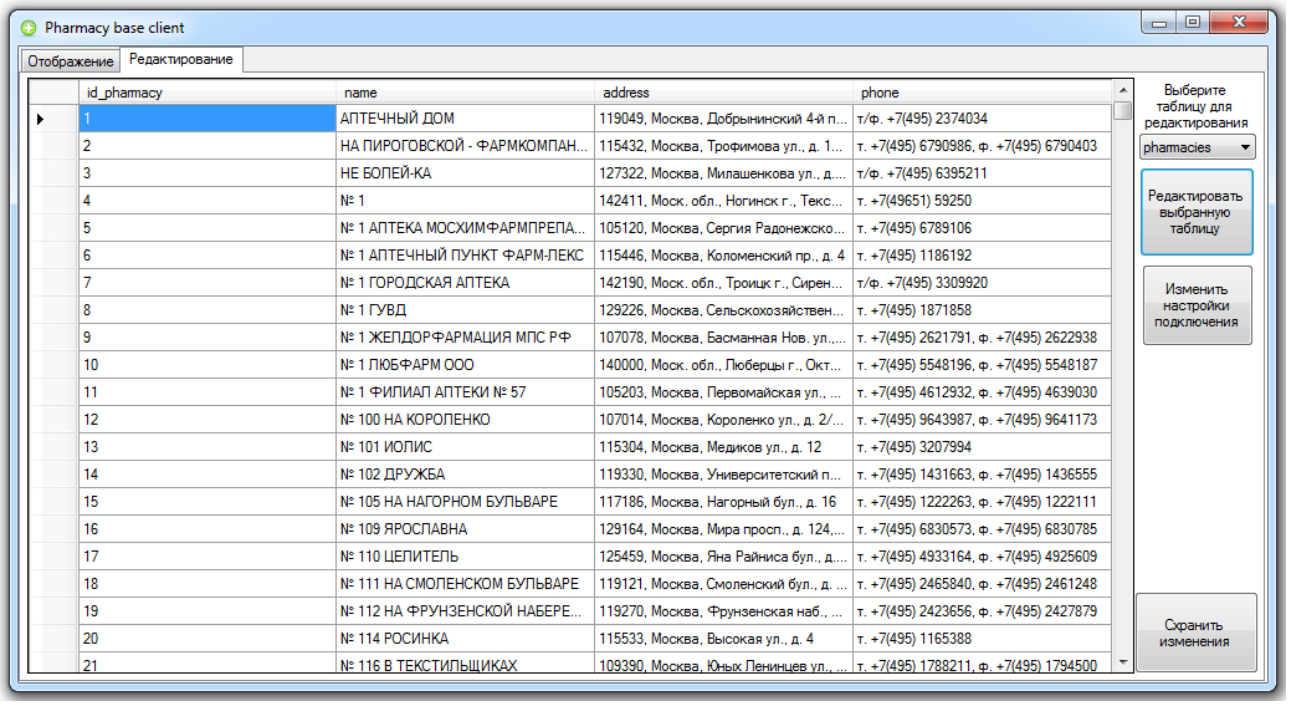

Рисунок 6.

Выводы

При выполнении данной расчётной работы я научился создавать SQL-запросы и отображать таблицы на платформе .Net для СУБД MS SQL Server 2008.

Научился создавать динамически изменяющийся интерфейс в зависимости от размеров окна приложения и работать с классами dataGridView и SqlDataAdapter, DataSet.

Список используемых источников

1. Просиз. Д. «Программирование для Microsoft» .Net /Пер. с англ. – М.: «Русская редакция», 2003. – 704 с.

2. Грабер М. «SQL» М.: «Лори», 2003 -664 с.

3. http://msdn.microsoft.com/ru-ru/