Федеральное государственное бюджетное образовательное учреждение

высшего образования «Омский государственный технический университет»

Институт военно-технического образования

РАСЧЁТНО – ГРАФИЧЕСКАЯ РАБОТА

По дисциплине «Конструирование программного обеспечения»

Вариант №1

Выполнил: студент

группы ПЭ-171

Алтынпара А.В.

Проверил: старший

преподаватель АСОиУ

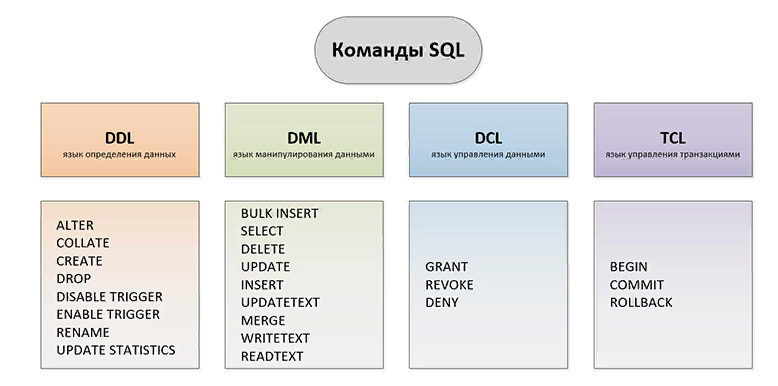
Убалехт И.П.

Омск 2020

# Введение

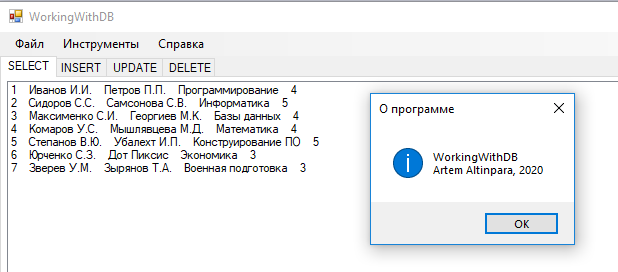
Задание: Предметная область «Учёт оценок за экзамен». Есть множество студентов, множество преподавателей, множество предметов. Формируется экзаменационная ведомость – оценки студентам за экзамены. Система должна «уметь» добавлять студентов, преподавателей, предметы, и выводить ведомость с оценками.

Каждый сайт в Интернете, любой проект, обрабатывающий значительный объем информации, вынужден хранить эту информацию в тех или иных базах данных (БД). Подавляющее большинство проектов информацию сохраняют в БД реляционного типа, делая записи в различных подобиях таблиц. Как внесение новых записей, так и обращение к имеющимся, осуществляется с благодаря использованию запросов, составляемых конструкциями SQL (structured query language) – непроцедурного декларативного языка структурированных запросов. В нашем случае это подразумевает, что, используя конструкции SQL, мы будем обращаться к БД, сообщая что нужно сделать с данными, но не указывая способ, как именно это нужно сделать (рис.1).



**Реализация**

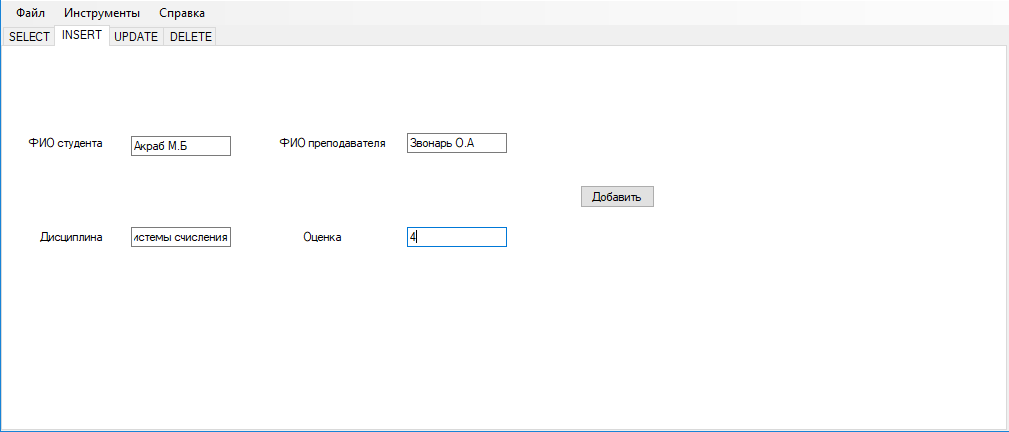
При запуске программы появляется окно listbox1, в котором будут отображаться результаты поиска студентов, преподавателей, дисциплин и оценок. (рис.2)



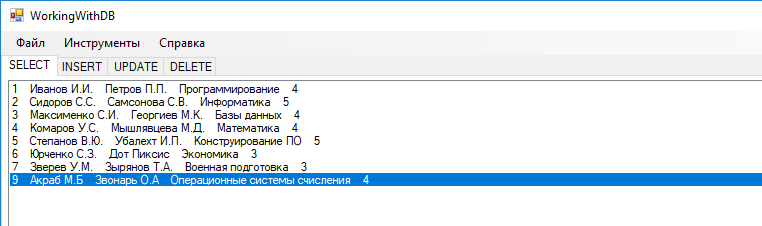
Добавляет данные сам пользователь. Программа при нажатии на вкладку «Инструменты» «Обновить» добавляет, изменяет существующую или удаляет запись таблицы. (рис.3)



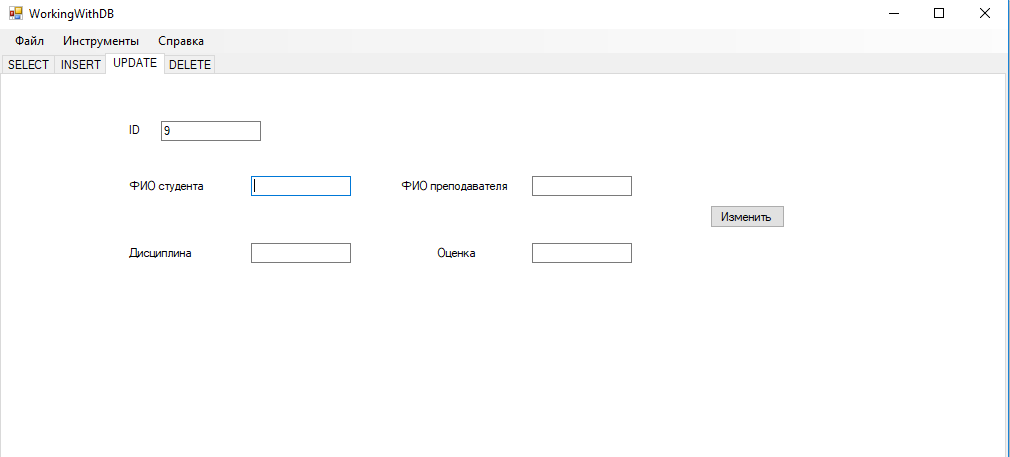
Вкладка INSERT представлена четырьмя элементами textbox и пятью элементами button. (рис.4)



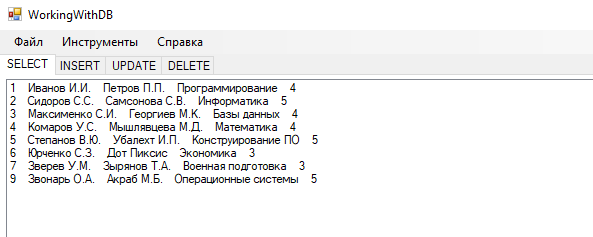
Пользователь может вводить данные в соответствующие окна и при нажатии кнопки «Добавить» и последующей комбинации «Инструменты» «Обновить» появляется новая запись. (рис.5)



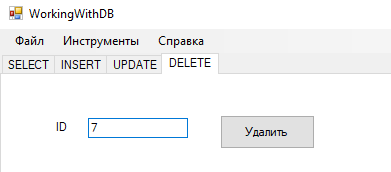
Запрос на изменение представлен теми же элементами, что и запрос INSERT. Добавляется лишь поле ID. (рис.6)



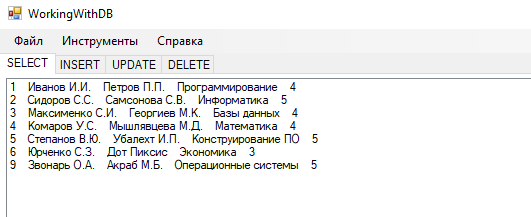
Программа понимает, какую запись в таблице нужно изменить по введённому пользователем идентификатору. Он вводит изменённые данные нужного идентификатора и нажимает кнопку «Изменить». «Инструменты» «Обновить» (рис.7)



Запрос на удаление записи из таблицы представлен ниже (рис.8)



Пользователь вводит нужный номер записи и нажимает комбинацию кнопок «Удалить». «Инструменты» «Обновить». Результат операции представлен ниже (рис.9).



Запись под номером 7 успешно удалена.

**Заключение**

Данная программа, написанная в Visual Studio c# 2013, прекрасно отражает возможности работы с БД и полностью выполняет задачи, поставленные выше.

**Список использованных источников**

Сайт ITVDN.com – [электронный ресурс] https://itvdn.com/ru/blog/article/m-sql

Сайт habr.com – [электронный ресурс] <https://habr.com/ru/post/480838/>

Сайт tradebenefit.ru – [электронный ресурс] <http://tradebenefit.ru/primery-mysql-zaprosov>

**Приложение А – код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WorkingWithDB

{

public partial class Form1 : Form

{

SqlConnection sqlConnection;//чтобы законектиться к БД создаём объект

public Form1()

{

InitializeComponent();

}

private async void Form1\_Load(object sender, EventArgs e)

{

string connectionString = @"Data Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\Артём Вячеславович\documents\visual studio 2013\Projects\WorkingWithDB\WorkingWithDB\Database.mdf;Integrated Security=True";

sqlConnection = new SqlConnection(connectionString);//создаём экземпляр класса sqlConnection

await sqlConnection.OpenAsync();//параллельное открытие соединения с БД

SqlDataReader sqlReader = null;//нужно получить содержимое БД

SqlCommand command = new SqlCommand("SELECT \* FROM [Accounting]", sqlConnection);//пишем первый запрос

try

{

sqlReader = await command.ExecuteReaderAsync();//метод считывания таблицы

while (await sqlReader.ReadAsync())//метод перемещения к следующей записи таблицы

{

listBox1.Items.Add(Convert.ToString(sqlReader["Id"]) + " " + Convert.ToString(sqlReader["Student"]) + " " + Convert.ToString(sqlReader["Teacher"]) + " " + Convert.ToString(sqlReader["Discipline"]) + " " + Convert.ToString(sqlReader["Evaluation"]));//заполняем листбокс

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

if (sqlReader != null)//чтобы код в обработчике событий выполнился/невыполнился,закрываем наш sqlReader

sqlReader.Close();

}

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)//чтобы не было утечки данных, закрываем соединение

{

if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)//если соединение не пусто и не закрыто уже

Application.Exit();

}

private async void button1\_Click(object sender, EventArgs e)

{

if (label7.Visible)//обрабатываем исключение

label7.Visible = false;

if (!String.IsNullOrEmpty(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox1.Text) &&

!String.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrWhiteSpace(textBox2.Text)&&

!String.IsNullOrEmpty(textBox8.Text) && !string.IsNullOrWhiteSpace(textBox8.Text)&&

!String.IsNullOrEmpty(textBox7.Text) && !string.IsNullOrWhiteSpace(textBox7.Text))

{

SqlCommand command = new SqlCommand("INSERT INTO [Accounting] (Student, Teacher, Discipline, Evaluation)VALUES(@Student, @Teacher, @Discipline, @Evaluation)", sqlConnection);

command.Parameters.AddWithValue("Student", textBox1.Text);//добавляем значения Student, Teacher, Discipline, Evaluation,чтобы БД их видела

command.Parameters.AddWithValue("Teacher", textBox2.Text);

command.Parameters.AddWithValue("Discipline", textBox8.Text);

command.Parameters.AddWithValue("Evaluation", textBox7.Text);

await command.ExecuteNonQueryAsync();//т.к. это INSERT И нам не нужно ничего возвращать

}

else

{

label7.Visible = true;

label7.Text = "Поля 'Студент', 'Преподаватель','Дисциплина' и 'Оценка' должны быть заполнены!";

}

}

private async void обновитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

listBox1.Items.Clear();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Accounting]", sqlConnection);

try

{

sqlReader = await command.ExecuteReaderAsync();//считывание таблицы

while (await sqlReader.ReadAsync())

{

listBox1.Items.Add(Convert.ToString(sqlReader["Id"]) + " " + Convert.ToString(sqlReader["Student"]) + " " + Convert.ToString(sqlReader["Teacher"]) + " " + Convert.ToString(sqlReader["Discipline"]) + " " + Convert.ToString(sqlReader["Evaluation"]));

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

if (sqlReader != null)

sqlReader.Close();

}

}

private async void button2\_Click(object sender, EventArgs e)

{

if (label8.Visible)//обрабатываем исключение

label8.Visible = false;

if (!String.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text) &&

!String.IsNullOrEmpty(textBox4.Text) && !string.IsNullOrWhiteSpace(textBox4.Text)&&

!String.IsNullOrEmpty(textBox3.Text) && !string.IsNullOrWhiteSpace(textBox3.Text)&&

!String.IsNullOrEmpty(textBox9.Text) && !string.IsNullOrWhiteSpace(textBox9.Text)&&

!String.IsNullOrEmpty(textBox10.Text) && !string.IsNullOrWhiteSpace(textBox10.Text))

{

SqlCommand command = new SqlCommand("UPDATE [Accounting] SET [Student]=@Student, [Teacher]=@Teacher, [Discipline]=@Discipline, [Evaluation]=@Evaluation WHERE [Id]=@Id", sqlConnection);

command.Parameters.AddWithValue("Id",textBox5.Text);

command.Parameters.AddWithValue("Student", textBox4.Text);

command.Parameters.AddWithValue("Teacher", textBox3.Text);

command.Parameters.AddWithValue("Discipline", textBox9.Text);

command.Parameters.AddWithValue("Evaluation", textBox10.Text);

await command.ExecuteNonQueryAsync();

}

else if (!String.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text))

{

label8.Visible = true;

label8.Text = "ID должен быть заполнен!";

}

else

{

label8.Visible = true;

label8.Text = "Поля 'Id', 'Студент', 'Преподаватель','Дисциплина' и 'Оценка' должны быть заполнены!";

}

}

private async void button3\_Click(object sender, EventArgs e)

{

if (label9.Visible)//обрабатываем исключение

label9.Visible = false;

if (!String.IsNullOrEmpty(textBox6.Text) && !string.IsNullOrWhiteSpace(textBox6.Text))

{

SqlCommand command = new SqlCommand("DELETE FROM [Accounting] WHERE [Id]=@Id", sqlConnection);

command.Parameters.AddWithValue("Id", textBox6.Text);

await command.ExecuteNonQueryAsync();

}

else

{

label9.Visible = true;

label9.Text = "Поле 'ID' должно быть заполнено!";

}

}

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("WorkingWithDB\nArtem Altinpara, 2020", "О программе", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}