МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И**

**СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащийся группы ПО-455 М.Г.Верхов

Шифр 15

2023

**6 Опишите таблицы, создание, модификацию, удаление таблиц.**

Создание таблицы

Для создания таблиц используется команда CREATE TABLE. Эта команды применяет ряд операторов, которые определяют столбцы таблицы и их атрибуты. Общий формальный синтаксис команды CREATE TABLE:

CREATE TABLE название\_таблицы

(название\_столбца1 тип\_данных атрибуты\_столбца1,

 название\_столбца2 тип\_данных атрибуты\_столбца2,

 ................................................

 название\_столбцаN тип\_данных атрибуты\_столбцаN,

 атрибуты\_уровня\_таблицы

После команды CREATE TABLE идет название таблицы. Имя таблицы выполняет роль ее идентификатора в базе данных, поэтому оно должно быть уникальным. Затем в скобках перечисляются названия столбцов, их типы данных и атрибуты. В самом конце можно определить атрибуты для всей таблицы. Атрибуты столбцов, а также атрибуты таблицы указывать необязательно.

Создадим простейшую таблицу. Для этого выполним следующий скрипт:

CREATE DATABASE productsdb;

USE productsdb;

CREATE TABLE Customers

(

    Id INT,

    Age INT,

    FirstName VARCHAR(20),

    LastName VARCHAR(20)

);

Таблица не может создаваться сама по себе. Она всегда создается в определенной базе данных. Вначале здесь создается база данных productsdb. И затем, чтобы указать, что все дальнейшие операции, в том числе создание таблицы, будут производиться с этой базой данных, применяется команда USE.

Далее собственно идет создание таблицы, которая называется Customers. Она определяет четыре столбца: Id, Age, FirstName, LastName. Первые два столбца представляют идентификатор клиента и его возраст и имеют тип INT, то есть будут хранить числовые значения. Следующие столбцы представляют имя и фамилию клиента и имеют тип VARCHAR(20), то есть представляют строку длиной не более 20 символов. В данном случае для каждого столбца определены имя и тип данных, при этом атрибуты столбцов и таблицы в целом отсутствуют.

И в результате выполнения этой команды будет создана база данных productsdb, в которой будет создана таблица Customers.

**Переименование таблиц**

Если после создания таблицы мы захотим ее переименовать, то для этого нужно использовать команду RENAME TABLE, которая имеет следующий синтаксис:

|  |  |
| --- | --- |
|  | RENAME TABLE старое\_название TO новое\_название; |

Например, переименуем таблицу Customers в Clients:

|  |  |
| --- | --- |
|  | RENAME TABLE Customers TO Clients; |

**Полное удаление данных**

Для полного удаления данных, очистки таблицы применяется команда TRUNCATE TABLE.

Например, очистим таблицу Clients:

|  |  |
| --- | --- |
|  | TRUNCATE TABLE Clients; |

**Удаление таблиц**

Для удаления таблицы из БД применяется команда DROP TABLE, после которой указывается название удаляемой таблицы. Например, удалим таблицу Clients:

|  |
| --- |
| DROP TABLE Clients; |

**Изменение таблиц и столбцов**

Если таблица уже была ранее создана, и ее необходимо изменить, то для этого применяется команда ALTER TABLE. Ее сокращенный формальный синтаксис:

ALTER TABLE название\_таблицы

{ ADD название\_столбца тип\_данных\_столбца [атрибуты\_столбца] |

  DROP COLUMN название\_столбца |

  MODIFY COLUMN название\_столбца тип\_данных\_столбца [атрибуты\_столбца] |

  ALTER COLUMN название\_столбца SET DEFAULT значение\_по\_умолчанию |

  ADD [CONSTRAINT] определение\_ограничения |

  DROP [CONSTRAINT] имя\_ограничения}

Вообще данная команда поддерживает гораздо больше опций и возможностей. Все их можно посмотреть в документации. Рассмотрим лишь основные сценарии, с которыми мы можем столкнуться.

### Добавление нового столбца

Добавим в таблицу Customers новый столбец Address:

|  |
| --- |
| ALTER TABLE Customers  ADD Address VARCHAR(50) NULL; |

В данном случае столбец Address имеет тип VARCHAR и для него определен атрибут NULL.

### Удаление столбца

Удалим столбец Address из таблицы Customers:

ALTER TABLE Customers

DROP COLUMN Address;

### Изменение значения по умолчанию

Установим в таблице Customers для столбца Age значение по умолчанию 22:

|  |
| --- |
| ALTER TABLE Customers  ALTER COLUMN Age SET DEFAULT 22; |

### Изменение типа столбца

Изменим в таблице Customers тип данных у столбца FirstName на CHAR(100) и установим для него атрибут NULL:

|  |
| --- |
| ALTER TABLE Customers  MODIFY COLUMN FirstName CHAR(100) NULL; |

### Добавление и удаление внешнего ключа

Пусть изначально в базе данных будут добавлены две таблицы, никак не связанные:

CREATE TABLE Customers

(

    Id INT PRIMARY KEY AUTO\_INCREMENT,

    Age INT,

    FirstName VARCHAR(20) NOT NULL,

    LastName VARCHAR(20) NOT NULL

);

CREATE TABLE Orders

(

    Id INT PRIMARY KEY AUTO\_INCREMENT,

    CustomerId INT,

    CreatedAt Date

);

Добавим ограничение внешнего ключа к столбцу CustomerId таблицы Orders:

ALTER TABLE Orders

ADD FOREIGN KEY(CustomerId) REFERENCES Customers(Id);

При добавлении ограничений мы можем указать для них имя, используя оператор CONSTRAINT, после которого указывается имя ограничения:

ALTER TABLE Orders

ADD CONSTRAINT orders\_customers\_fk

FOREIGN KEY(CustomerId) REFERENCES Customers(Id);

В данном случае ограничение внешнего ключа называется orders\_customers\_fk. Затем по этому имени мы можем удалить ограничение:

ALTER TABLE Orders

DROP FOREIGN KEY orders\_customers\_fk;

### Добавление и удаление первичного ключа

Добавим в таблицу Products первичный ключ:

CREATE TABLE Products

(

    Id INT,

    Model VARCHAR(20)

);

ALTER TABLE Products

ADD PRIMARY KEY (Id);

Теперь удалим первичный ключ:

ALTER TABLE Products

DROP PRIMARY KEY;

**54 Охарактеризуйте общие особенности трехзвенной архитектуры.**

Трёху́ровневая архитекту́ра (*трёхзве́нная архитекту́ра*, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *three-tier*) — [архитектурная модель](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [программного комплекса](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), предполагающая наличие в нём трёх типов компонентов (уровней, звеньев): [клиентских приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (с которыми работают [пользователи](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C)), [серверов приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (с которыми работают клиентские приложения) и [серверов баз данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (с которыми работают серверы приложений)

|  |
| --- |
|  |
| Рисунок 1. Трехзвенная архитектура с сервером приложений |

Клиенты (рисунок. 1) содержат только слой логики представления прикладного ПО, а алгоритмы бизнес-логики и логики доступа к данным перемещены в среднее звено. В этом случае сервер приложений обеспечивает «общее хранилище» бизнес-правил и процедур. Клиенты соединяются с сервером приложений и предоставляют ему данные для обработки. Совместное использование алгоритмов бизнес-логики, общих для всего приложения, обладает важными достоинствами. Помимо «утоньшения» клиента, эта стратегия ведет к созданию системы, в которой будущие обновления прикладного ПО будут производиться, главным образом, на сервере приложений, что упрощает процесс модификаций.

Обычно сервер приложений поддерживает пул ограниченного числа открытых подключений к базам данных и вместо того чтобы делать каждое подключение к базе данных выделенным для определенного клиента (как в двухзвенной архитектуре), подключения многократно используются для выполнения запросов различных клиентов.

Есть и другие преимущества. Во-первых, так как вся «важная» часть прикладной логики теперь централизована в среднем звене, нет необходимости поддерживать сложные механизмы аутентификации на стороне клиентов.

Во-вторых, аппаратная платформа, на которой выполняется сервер приложений, может быть достаточно мощной; это дает дополнительную степень масштабируемости всей прикладной системы.

В-третьих, централизованный доступ к данным в серверах приложений делает всю прикладную систему менее зависящей от конкретной СУБД.

Наконец, сервер приложений обеспечивает эффективную стратегию для интеграции. Придерживаясь того же протокола коммуникации, что и клиент, другое «внешнее» приложение может легко взаимодействовать с «чужим» сервером приложений. Эта конфигурация допускает интеграцию приложений не только на уровне данных, но и на уровне правил бизнес-логики. Это чрезвычайно важно, потому что совместное использование данных разными прикладными программами может вести к логическим противоречиям в базе данных. Типичное решение состоит в том, чтобы копировать одни и те же правила и алгоритмы в несколько прикладных программ. Но тогда очень затрудняются их поддержка и обновление — любое изменение кода должно проводиться во всех прикладных программах, которые его используют. Если же бизнес-правила сосредоточены на сервере приложений и используются совместно, то такой проблемы нет.

Отличие «многозвенной» архитектуры от «двухзвенной». Довольно распространена модель работы, когда клиент обращается не непосредственно к серверу БД, а к промежуточной программе. Эта программа обычно называется сервером приложений. Такую архитектуру называют «трехзвенной», в отличие от «двухзвенной» архитектуры клиент-сервер.

**81 «Библиотека компакт-дисков»**

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server по № варианта (задание 61-90), содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.

**Решение**

Создадим форму и разместим на ней компонент TabControl в котором разместим 4 вкладки, по одной для каждой таблицы и итоговую для выполнения заданий. Внешний вид формы представлен на рисунке 2. На каждой из них разместим соответствующие элементы управления: поля ввода данных и кнопки для добавления, редактирования и удаления записи, а на итоговой вкладке кнопки для запуска заданий.

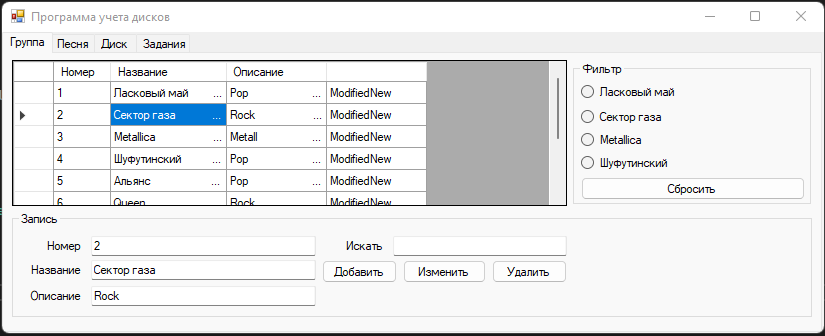


Рисунок 2 Внешний вид основной формы

Для примера добавления строки в таблицу внешней формой создадим вторую форму и разместим на ней элементы управления (рисунок 3)

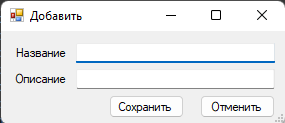


Рисунок 3 Внешний вид формы 2

Заполним таблицы данными, у нас это будут таблицы: Диск, Песня и Группа. Внешний вид приложения со вкладками Песня и Диск представлен на рисунках 4 и 5.

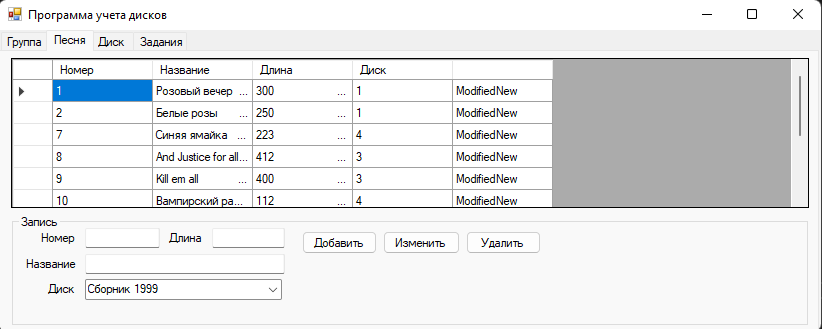


Рисунок 4 Приложение вкладка Песня

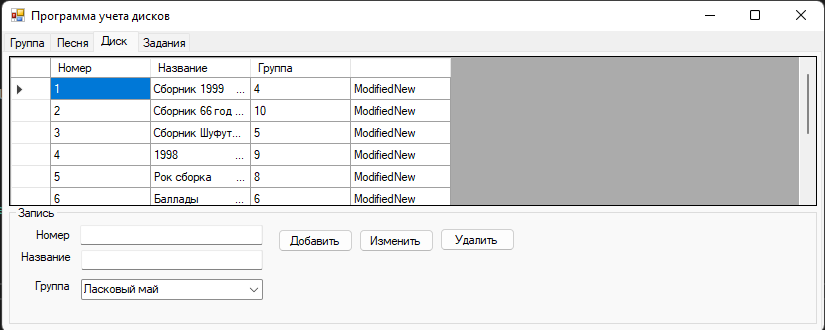


Рисунок 5 Приложение, вкладка Диск

На рисунке 6 представлен внешний вид приложения с открытой вкладкой для выполнения заданий. Выполнено задание вывода данных из двух таблиц.

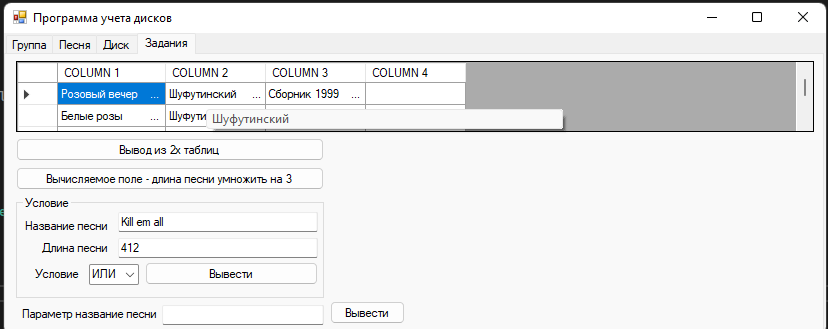


Рисунок 6 Приложение вкладка выполнения заданий

На рисунке 7 представлен результат выполнения действий над полями таблицы, длина песни умножается на 3 и выводится в третьем столбце.

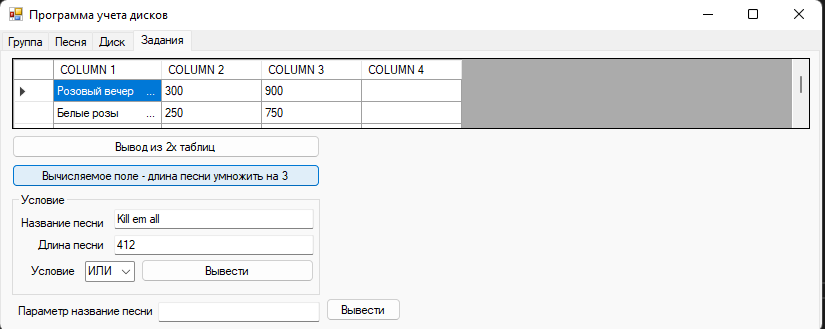


Рисунок 7 Результат выполнения задания вычисляемых полей

На рисунке 8 представлен вид приложения с выполненным заданием выбора песни по названию, либо по длине.

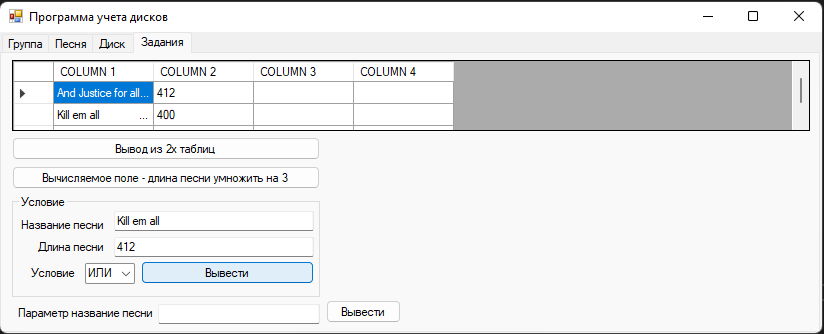


Рисунок 8 Результат выполнения задания Условие

На рисунке 9 представлен вывод задания с параметром, в качестве параметра название песни

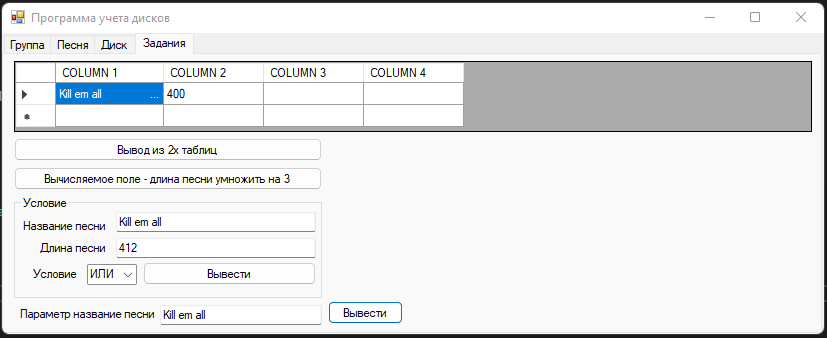
****

Рисунок 9 Результат выполнения задания Параметр

На рисунке 10 представлен результат выполнения задания Фильтр

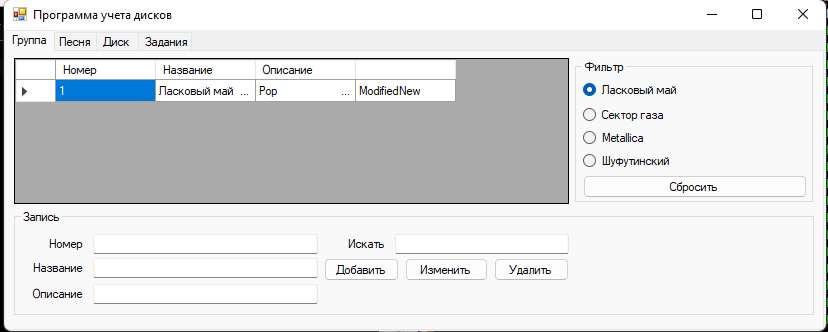
****

Рисунок 10результат выполнения задания Фильтр

На рисунке 11 представлен результат выполнения задания Поиск

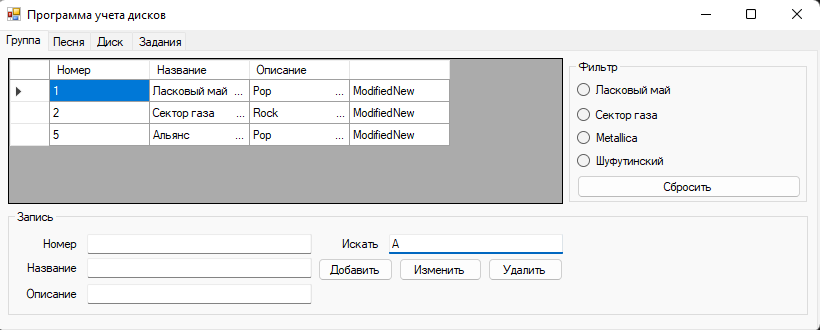


Рисунок 11результат выполнения задания Поиск

**Код программы, Форма 1**

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;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Disk

{

enum rowState

{

Existed,

New,

Modified,

ModifiedNew,

Deleted

}

public partial class Form1 : Form

{

DataBase database = new DataBase();

int selectedRow;

public Form1()

{

InitializeComponent();

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Disk]");

}

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

{

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Disk]");

}

private void CreateColumns()

{

dataGridView1.Columns.Add("ID", "Номер");

dataGridView1.Columns.Add("Name", "Название");

dataGridView1.Columns.Add("Descr", "Описание");

dataGridView1.Columns.Add("IsNew", String.Empty);

dataGridView2.Columns.Add("ID", "Номер");

dataGridView2.Columns.Add("Name", "Название");

dataGridView2.Columns.Add("Len", "Длина");

dataGridView2.Columns.Add("IDDisk", "Диск");

dataGridView2.Columns.Add("IsNew", String.Empty);

dataGridView3.Columns.Add("ID", "Номер");

dataGridView3.Columns.Add("Name", "Название");

dataGridView3.Columns.Add("IDGroup", "Группа");

dataGridView3.Columns.Add("IsNew", String.Empty);

dataGridView4.Columns.Add("Col1", "COLUMN 1");

dataGridView4.Columns.Add("Col2", "COLUMN 2");

dataGridView4.Columns.Add("Col3", "COLUMN 3");

dataGridView4.Columns.Add("Col4", "COLUMN 4");

}

private void ReadSinglRow(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), rowState.ModifiedNew);

}

private void ReadSinglRow2(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetInt32(3), rowState.ModifiedNew);

}

private void ReadSinglRow3(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetInt32(2), rowState.ModifiedNew);

}

private void ReadSinglRow4(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetString(0), record.GetString(1), record.GetString(2), record.GetString(3));

}

private void refreshDG(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Group]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dgw, reader);

}

reader.Close();

}

private void refreshDG2(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Song]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow2(dgw, reader);

}

reader.Close();

}

private void refreshDG3(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Disk]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow3(dgw, reader);

}

reader.Close();

}

private void refreshDG4(DataGridView dgw, string s)

{

dgw.Rows.Clear();

string querystr = s;// $"Select \* from [Disk]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dgw, reader);

}

reader.Close();

}

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

{

}

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

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

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

{

Form2 f\_add = new Form2();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where concat(Name,Descr) like '%" + textBox4.Text + "%'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

}

private void radioButton1\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Ласковый май'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

refreshDG(dataGridView1);

}

private void button3\_Click\_1(object sender, EventArgs e)

{

Form2 f\_add = new Form2();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void dataGridView1\_CellClick\_1(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void radioButton2\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Сектор газа'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton4\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Шуфутинский'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton3\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Group] Where Name = 'Metallica'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"delete from [Group] where ID={textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"update [Group] set Name='{textBox2.Text}' , Descr = '{textBox3.Text}' where ID = {textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "databaseDataSet1.Group". При необходимости она может быть перемещена или удалена.

this.groupTableAdapter.Fill(this.databaseDataSet1.Group);

// TODO: данная строка кода позволяет загрузить данные в таблицу "databaseDataSet1.Disk". При необходимости она может быть перемещена или удалена.

this.diskTableAdapter.Fill(this.databaseDataSet1.Disk);

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

textBox5.Text = row.Cells[0].Value.ToString();

textBox6.Text = row.Cells[1].Value.ToString();

textBox7.Text = row.Cells[2].Value.ToString();

comboBox1.SelectedIndex = Int32.Parse(row.Cells[3].Value.ToString())-1;

}

}

private void button5\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Song] (Name,Len, IdDisk) values ('{textBox6.Text}' , '{textBox7.Text}',{comboBox1.SelectedIndex+1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button7\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Song] where ID={textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

textBox8.Text = row.Cells[0].Value.ToString();

textBox9.Text = row.Cells[1].Value.ToString();

comboBox2.SelectedIndex = Int32.Parse(row.Cells[2].Value.ToString())-1;

}

}

private void button8\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Disk] (Name,IdGroup) values ('{textBox9.Text}' ,{comboBox2.SelectedIndex + 1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button10\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Disk] where ID={textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button6\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Song] set Name='{textBox6.Text}' , Len = '{textBox7.Text}', IdDisk = '{comboBox1.SelectedIndex+1}' where ID = {textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button9\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Disk] set Name='{textBox9.Text}' , IdGroup = '{comboBox2.SelectedIndex+1}' where ID = {textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button11\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select s.name as col1, g.name as col2, d.name as col3,'' as col4\r\nfrom [Disk] d, [Song] s, [Group] g\r\nwhere d.idgroup=g.id and s.iddisk=d.id";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button12\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, CAST(len\*3 AS varchar(5)) as col3, '' as col4 from [Song] s";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button13\_Click(object sender, EventArgs e)

{// условие

dataGridView4.Rows.Clear();

string s = "OR";

if (comboBox3.SelectedIndex == 0) {

s = "AND"; }

string n = textBox10.Text;

string l = textBox11.Text;

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, '' as col3, '' as col4 from [Song] s where Name='{n}' {s} Len='{l}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button14\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string n = textBox12.Text;

string querystr = $"select name as col1, CAST(len AS varchar(5)) as col2, '' as col3, '' as col4 from [Song] s where Name='{n}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

}

class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=Database;Integrated Security=True");

public void openConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()

{

return sqlConnection;

}

}

}

**Форма 2**

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;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Disk

{

public partial class Form2 : Form

{

DataBase dataBase = new DataBase();

public Form2()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

dataBase.openConnection();

var addQuery = $"insert into [Group] (Name,Descr) values ('{textBox1.Text}' , '{textBox2.Text}')";

var command = new SqlCommand(addQuery, dataBase.getConnection());

command.ExecuteNonQuery();

dataBase.closeConnection();

this.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

this.Close();

}

}

}

**Список использованных источников**

1. Дейт, К.Дж. Введение в системы баз данных/ К.Дж.Дейт. – М.: Вильямс, 2018
2. Дюбуа, Поль. MySQI / Поль Дюбуа. – 3-е изд. – М.: Издательский дом "Вильямс", 2007. - 816 с.: ил.
3. Колисниченко, Д.Н. Профессиональное программирование на PНP / Д.Н. Колисниченко. – Санкт-Петербург: БХВ-Петербург, 2007. –416 с.
4. Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас. – Минск:РИПО, 2016
5. Роб, П. Системы баз данных: проектирование, реализация и управление / П.Роб, К.Коронел. – 5-е изд., перераб. и доп.: пер. с англ. – Санкт-Петербург: БХВ-Петербург, 2004. - 1040 с.: ил
6. Фаронов, В.В. Программирование баз данных в Delphi / В.В.Фаронов. – 2-е изд. – СанктПетербург.: Питер, 2004. – 459 с.
7. Хернандес, М.Дж. SQL-запросы для простых смертных: практическое руководство по манипулированию данными в SQL / М.Дж.Хернандес, Дж.Л.Вьескас. – Москва: Лори, 2000. - 473 с.