МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

дисциплина «Избранные главы информатики»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«**Разработка модели предметной области с использованием ENTITY FRAMEWORK**»

Вариант № 16

Выполнил:

студент группы ИП-31, Самусев Р.А.

Принял:

доцент Асенчик О.Д.

Гомель 2018

**1. Цель работы:** Ознакомиться с возможностями ENTITY FRAMEWORK для моделирования предметной области, выборки, вставки, удаления и обновления данных объектов, связанных с таблицами базы данных заданной предметной области.

**2. Ход работы и результаты.**

**Листинг класса модели Рекламы**

using System;

using System.Collections.Generic;

using System.Text;

namespace advertisingAgency

{

public class Advedirsments

{

public int AdvedirsmentsId { get; set; }

public string AdType { get; set; }

public string AdDescription { get; set; }

public virtual ICollection<Orders> Orders { get; set; }

public override string ToString()

{

return "Тип: " + AdType + " Описание: " + AdDescription;

}

}

}

**Листинг класса модели Заказы**

using System;

using System.Collections.Generic;

namespace advertisingAgency

{

public class Orders

{

public int OrdersId { get; set; }

public DateTime DateOrder { get; set; }

public DateTime DateBegin { get; set; }

public DateTime DateEnd { get; set; }

public string Location { get; set; }

public int OrderCost { get; set; }

public bool PaymentStatus { get; set; }

public int EmployeeId { get; set; }

public int? CustomersId { get; set; }

public int? AdvedirsmentsId { get; set; }

public virtual Advedirsments Advedirsments { get; set; }

public virtual Customers Customers { get; set; }

public override string ToString()

{

return " Id Рекламы: " + AdvedirsmentsId + " Id Заказчика: " + CustomersId + " Цена Услуги: " +

OrderCost + " Id Сотрудника: " + EmployeeId + " Локация рекламы: " + Location

+ " Дата договора: " + DateOrder + " Дата начала: " + DateBegin + " Дата окончания: " + DateEnd +

" Статус оплаты: " + PaymentStatus;

}}}

**Листинг класса модели Заказчики**

using System;

using System.Collections.Generic;

using System.Text;

namespace advertisingAgency

{

public class Customers

{

public int CustomersId { get; set; }

public string CustomerName { get; set; }

public string CustomerAddress { get; set; }

public string CustomerTelephone { get; set; }

public virtual ICollection<Orders> Orders { get; set; }

public override string ToString()

{

return "Имя заказчика: " + CustomerName + " адрес: " + CustomerAddress + "телефон: " + CustomerTelephone;

}

}

}

**Листинг класса контекста данных**

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using System.IO;

namespace advertisingAgency

{

public class adAgencyContext:DbContext

{

public DbSet<Advedirsments> Advedirsments { get; set; }

public DbSet<Customers> Customers { get; set; }

public DbSet<Orders> Orders { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

var builder = new ConfigurationBuilder();

// установка пути к текущему каталогу

builder.SetBasePath(Directory.GetCurrentDirectory());

// получаем конфигурацию из файла appsettings.json

builder.AddJsonFile("appsettings.json");

// создаем конфигурацию

var config = builder.Build();

// получаем строку подключения

string connectionString = config.GetConnectionString("SQLConnection");

var options = optionsBuilder

.UseSqlServer(connectionString)

.Options;

}

}

}

**Листинг класса для инициализации базы данных**

using System;

using System.Linq;

namespace advertisingAgency

{

public static class DbInitializator

{

public static void Initialize(adAgencyContext db)

{

db.Database.EnsureCreated();

if (db.Advedirsments.Any())

{

return; // База данных инициализирована

}

int advedirsmentsNumber = 5;

int customersNumber = 100;

int ordersNumber = 100;

string adType;

string adDescription;

string customerName;

string customerAddress;

string customerTelephone;

Random randObj = new Random(1);

//словарь типов рекламы

string[] adTypes = { "Телевизионная", "Интернет-реклама", "Печатная", "Радио", "Уличная" };

//словарь описание типов рекламы

string[] adDescrip = { "Видеоролик, бегущая строка", "реклама в Интернете, блогах, SMM",

"реклама в газетах, наклейки, визтки", "реклама в радиовещании", "Баннеры" };

//заполнение таблицы рекламы

for (int advedirsmentId = 1; advedirsmentId <= advedirsmentsNumber; advedirsmentId++)

{

adType = adTypes[advedirsmentId - 1];

adDescription = adDescrip[advedirsmentId - 1];

db.Advedirsments.Add(new Advedirsments { AdType = adType, AdDescription = adDescription, AdvedirsmentsId = advedirsmentId });

}

//сохранение изменений в базу данных, связанную с объектом контекста

db.SaveChanges();

//словарь название заказчиков

string[] custNames = { "LowAgency", "FedEx", "BSBank", "Samsung", "Unicredit", "MTS", "Bordeux" };

//словарь адресов заказчиков

string[] custAddress = { "ул. Пушкина 2", "пр.Речицкий 80", "ул. Гагарина 38",

"ул. Центральная 7", "ул. Киевская 7", "ул. Давыдовская 14",

"ул. Б.Хмельницкого 80" };

//словарь телефонов заказчиков

string[] custTelehone = { "577-19-05", "303-23-68", "158-38-64", "190-65-87", "715-71-77" };

//заполнение таблицы заказчики

for (int customerId = 1; customerId <= customersNumber; customerId++)

{

customerName = custNames[randObj.Next(custNames.Length)];

customerAddress = custAddress[randObj.Next(custAddress.Length)];

customerTelephone = custTelehone[randObj.Next(custTelehone.Length)];

db.Customers.Add(new Customers { CustomerName = customerName, CustomerAddress = customerAddress, CustomerTelephone = customerTelephone });

}

//сохранение изменений в базу данных, связанную с объектом контекста

db.SaveChanges();

//массив id сотрудника

int[] employesId = { 1, 2, 3, 4, 5 };

//словарь рапсоложений реклам заказчика

string[] locations = { "ул. Богданова", "пр. Речицкий", "ул. Барыкина", "ул. Машерова",

"ул. Спартака", "пр. Космонавтов", "ул. Ветеранов", "пл. Соборная" };

for(int orderId = 1; orderId <= ordersNumber; orderId++)

{

int advedirsmentId = randObj.Next(1, advedirsmentsNumber - 1);

int customerId = randObj.Next(1, customersNumber - 1);

int orderCost = randObj.Next(1000, 10000);

int employeerId = employesId[randObj.Next(employesId.Length)];

string location = locations[randObj.Next(locations.Length)];

DateTime today = DateTime.Now.Date;

DateTime dateOrder = today.AddDays(-orderId);

DateTime dateBegin = dateOrder.AddDays(randObj.Next(7));

DateTime dateEnd = dateBegin.AddDays(randObj.Next(7, 30));

bool paymentStatus;

int isPay = randObj.Next(0, 1);

if (isPay == 1)

paymentStatus = true;

else

paymentStatus = false;

db.Orders.Add(new Orders {DateOrder = dateOrder, DateBegin = dateBegin, DateEnd = dateEnd,

Location = location, OrderCost = orderCost, PaymentStatus = paymentStatus, EmployeeId = employeerId,

CustomersId = customerId, AdvedirsmentsId = advedirsmentId });

}

//сохранение изменений в базу данных, связанную с объектом контекста

db.SaveChanges();

}

}

}

**Строка подключения к базе данных из конфигурационного файла *appsettings.json***

{

"ConnectionStrings": {

"SQLConnection": "Server=(localdb)\\mssqllocaldb;Database=adAgency;Trusted\_Connection=True;"

}

}

**Листинг класса Program, в котором выполняются LINQ запросы**

using System;

using System.Collections;

using System.Linq;

namespace advertisingAgency

{

class Program

{

static void Main(string[] args)

{

using (adAgencyContext db = new adAgencyContext())

{

DbInitializator.Initialize(db);

Console.WriteLine("Первое задание");

Print("Выборка записей таблицы Рекламы", db.Advedirsments.ToArray());

Print("Выборка записей таблицы Заказчики", db.Customers.ToArray());

Console.WriteLine("\nВторое задание\n");

Print("Выборка с ограничением из таблицы Рекламы", db.Advedirsments.Where(a => a.AdvedirsmentsId == db.Advedirsments.First().AdvedirsmentsId).ToArray());

Console.WriteLine("\nТретье задание\n");

var query1 = from o in db.Orders

group o.OrderCost by o.CustomersId into gr

select new

{

customersId = gr.Key,

Сумма = gr.Sum()

};

Print("Выборка с суммой стоимости заказов по клиентам:", query1.ToArray());

Console.WriteLine("\nЧетвёртое задание\n");

var query2 = from o in db.Orders

join c in db.Customers

on o.CustomersId equals c.CustomersId

orderby c.CustomerName descending

select new

{

Заказчик = c.CustomerName,

Дата\_соглашения = o.DateOrder,

Стоимость\_заказа = o.OrderCost

};

Print("Выборка по полям двух таблиц связи один-ко-многим:", query2.ToArray());

Console.WriteLine("\nПятое задание\n");

DateTime day = new DateTime(1, 1, 2018);

var query3 = from o in db.Orders

join c in db.Customers

on o.CustomersId equals c.CustomersId

where(o.DateBegin >= day)

orderby c.CustomerName descending

select new

{

Заказчик = c.CustomerName,

Дата\_соглашения = o.DateOrder,

Стоимость\_заказа = o.OrderCost

};

Print("Выборка с ограничением по полям двух таблиц связи один-ко-многим:", query3.ToArray());

Console.WriteLine("\nШестое задание\n");

// Создать новую рекламу

Advedirsments ad = new Advedirsments

{

AdType = "SMS-реклама",

AdDescription = "Реклама распространяется клиентам через SMS"

};

// Создать нового заказчика

Customers customer = new Customers

{

CustomerName = "БургерКинг",

CustomerAddress = "ул. Северная 7",

CustomerTelephone = "380-65-87"

};

db.Advedirsments.Add(ad);

db.Customers.Add(customer);

db.SaveChanges();

Print("Таблица рекламы после вставки: ", db.Advedirsments.ToArray());

Print("\nТаблица заказчики после вставки: ", db.Advedirsments.ToArray());

Console.WriteLine("\nСедьмое задание\n");

DateTime today = DateTime.Now.Date;

//создать новый заказ

Orders order = new Orders

{

AdvedirsmentsId = 1,

CustomersId = 3,

OrderCost = 1000,

EmployeeId = 5,

Location = "ул. Набережная",

DateOrder = today.AddDays(-15),

DateBegin = today.AddDays(10),

DateEnd = today.AddDays(30),

PaymentStatus = true

};

db.Orders.Add(order);

db.SaveChanges();

Print("Таблица заказы после вставки: ", db.Orders.ToArray());

Console.WriteLine("\nВосьмое задание\n");

var adDel = db.Advedirsments.Where(a => a.AdvedirsmentsId == 6);

db.Advedirsments.RemoveRange(adDel);

db.SaveChanges();

Print("Таблица рекламы после удаления записи: ", db.Advedirsments.ToArray());

Console.WriteLine("\nДевятое задание\n");

db.Orders.Remove(db.Orders.ToArray()[10]);

db.SaveChanges();

Print("Таблица заказы после удаления записи: ", db.Advedirsments.ToArray());

db.Orders.SingleOrDefault(o => o.OrdersId == db.Orders.First().OrdersId).OrderCost += 3000;

db.SaveChanges();

Print("Обновлённая таблица заказы", db.Orders.ToArray());

Console.ReadKey();

}

}

static void Print(string sqltext, IEnumerable items)

{

Console.WriteLine(sqltext);

Console.WriteLine("Записи: ");

foreach (var item in items)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine();

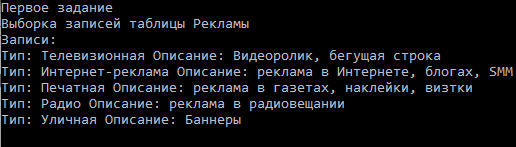
Console.ReadKey();

}

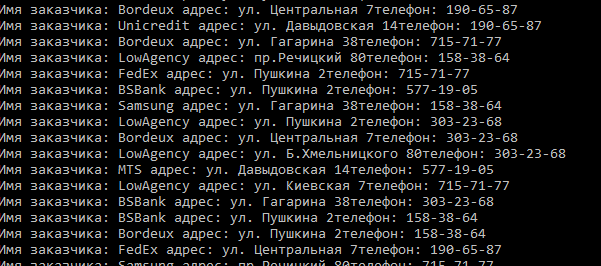
}

}

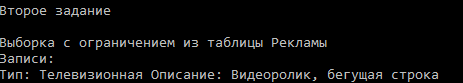
1. Результат LINQ запроса выборки всех данных из таблицы стоящей в схеме базы данных нас стороне отношения «один» – 1 шт.



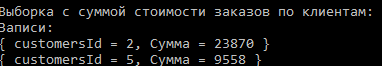
1. Результат LINQ запроса выборки данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей – 1 шт.



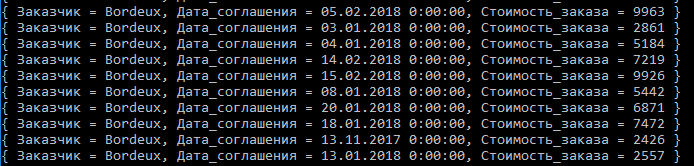
1. Результат LINQ запроса выборки данных, сгруппированных по любому из полей данных с выводом какого-либо итогового результата (min, max, avg, сount или др.) по выбранному полю из таблицы, стоящей в схеме базы данных нас стороне отношения «многие» – 1 шт.



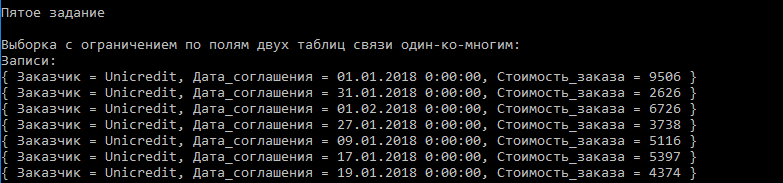
1. Результат LINQ запроса выборки данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.



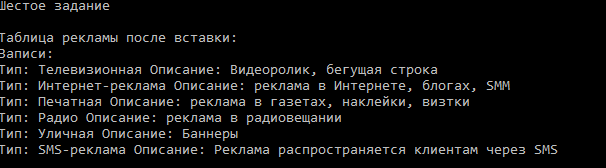
1. Результат LINQ запроса выборки данных из двух таблиц, связанных между собой отношением «один-ко-многим» и отфильтрованным по некоторому условию, налагающему ограничения на значения одного или нескольких полей – 1 шт.



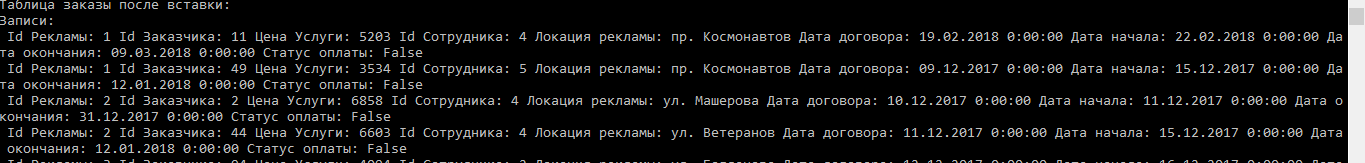
1. Результат LINQ запроса вставки данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.



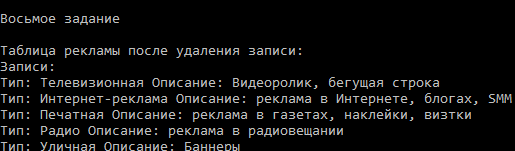
1. Результат LINQ запроса вставки данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:



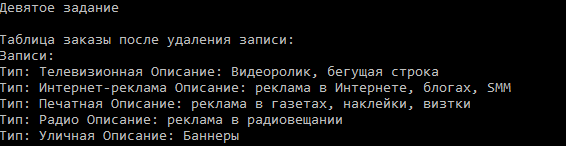
1. Результат LINQ запроса удаления данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.



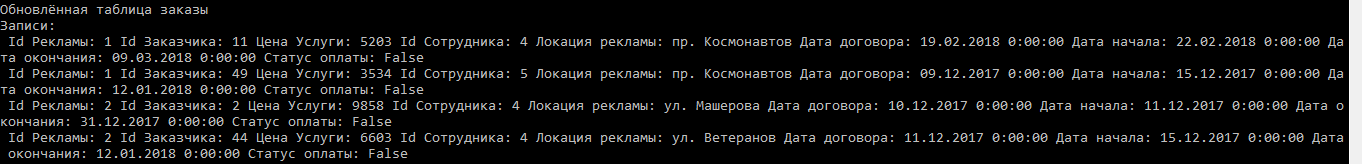
1. Результат LINQ запроса удаления данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.



1. Результат LINQ запроса обновления удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.



Результат LINQ



**3. Вывод:** в ходе выполнения данной лабораторной работы были получены навыки в работе с ENTITY FRAMEWORK для моделирования предметной области, выборки, вставки, удаления и обновления данных объектов, связанных с таблицами базы данных заданной предметной области. Задания выполнены полностью