МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине «Разработка приложений баз данных для информационных систем»

# на тему: «Использование *Entity* *Framework* и *LINQ* для работы с базами данных»

Выполнил: студент гр. ИТИ-31

*Степченко М.Р.*

Принял: преподаватель

Малиновский И. Л.

Гомель 2023

**Цель работы:** ознакомиться с возможностями *ENTITY* *FRAMEWORK* и получить навыки написания *LINQ* запросов к объектам, связанным с таблицами базы данных СУБД *MS* *SQL* сервер.

**Практическое задание**

**Задание.**

3.1. Создать с использованием .*NET* *Core* *Entity* *Framework* *Core* консольное приложение, содержащее набор классов, моделирующих предметную область соответствующей своему варианту и ранее созданную и заполненной тестовыми данными задания базой *MS* *SQL* *Server*. Для этого необходимо создать:

* Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту.
* Класс контекста данных.

3.2. Выполнить, используя объекты *Entity* *Framework* *Core* и *LINQ*:

1. Выборку всех данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один» – 1 шт.
2. Выборку 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.
3. Выборку 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.
4. Выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.
5. Выборку 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.
6. Вставку данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.
7. Вставку данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:
8. Удаление данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.
9. Удаление данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.
10. Обновление удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.

3.3. Разместить выполненный проект на *github*.

Предметная область базы данных «Система для тестирования»:

1. Пользователь (логин, пароль, *email*, имя, фамилия).

2. Тест (название, тематика, сложность, количество вопросов,

пользователь, отметка о прохождении).

3. Вопрос (название, текст вопроса, тест, количество баллов за

правильный ответ).

4. Ответ (текст ответа, вопрос).

5. Результат прохождения теста (оценка, тест, пользователь).

**Ход работы**

**Порядок выполнения задания.**

Для создания классов таблиц и представлений, а также класса контекста данных в командной строке была прописана следующая команда: *Scaffold-DbContext "Server = MSI; Database = MarriageAgency; Trusted\_Connection = True;".* Листинг созданных после выполнения команды классов приведён в приложении А.

После запуска приложения пользователь может выбрать один из 10 пунктов меню, каждый из которых представляет собой результат выполнения заданий 3.2.

Для выборки данных из таблицы *Employees,* стоящей на стороне «один»,в классе *Program* был разработан метод *FirstTask,* результат выполнения которого представлен на рисунке 1.

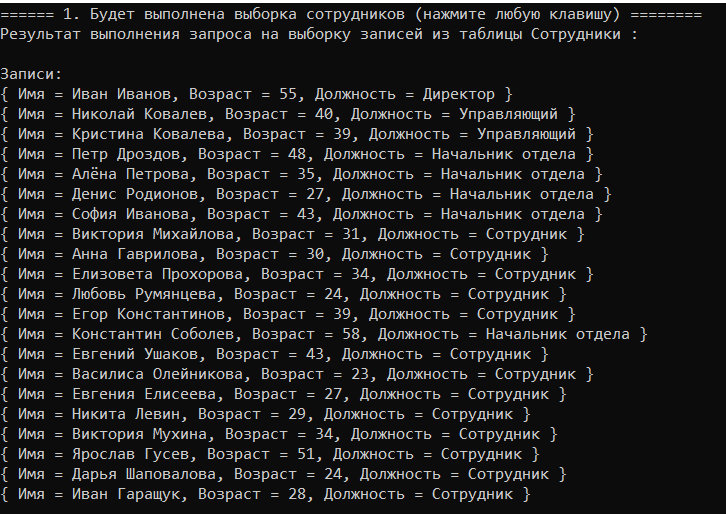


Рисунок 1 – Выборка данных из таблицы *Employees*

Длявыборки данных из таблицы *Employees,* стоящей на стороне «один», был разработан метод *SecondTask.* Вывод данных осуществляется только с записями, где *Bithday.Year* меньше 1983. Результат выполнения метода представлен на рисунке 2.

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

Рисунок 2 – Вывод данных из таблицы *Employees*, где *Bithday.Year* меньше 1983

Для выборки данных о количестве тестов каждой тематики был разработан метод *ThirdTask.* Для реализации данного метода была выполнена группировка по полю *ProvidedServices* из таблицы *Employees,* а также с помощью Count было подсчитано количество каждых записей. Результат выполнения метода представлен на рисунке 3.

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Рисунок 3 – Вывод данных о количестве выполненных услуг каждого сотрудника

Для выборки данных из таблиц *Employees* и *ProvidedServices* был разработан метод *FourthTask*, результат которого приведён на рисунке 4. Для удобства были выведены первые 10 записей.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Выборка данных из таблиц *Employees* и *ProvidedServices*

Для вывода данных из таблиц *ProvidedServices* и *Employees,* где цена больше 250, был разработан метод *FifthTask*, результат выполнения которого приведён на рисунке 5.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 – Выборка данных из таблиц *ProvidedServices* и *Employees* с фильтрацией

Для вставки данных в таблицу *Employees,* стоящей на стороне отношения «один», был разработан метод SixthTask, который предоставляет пользователю ввод нового сотрудника. После выполнения метода изменения сохраняются в базе данных. Результат выполнения метода показан на рисунке 6.

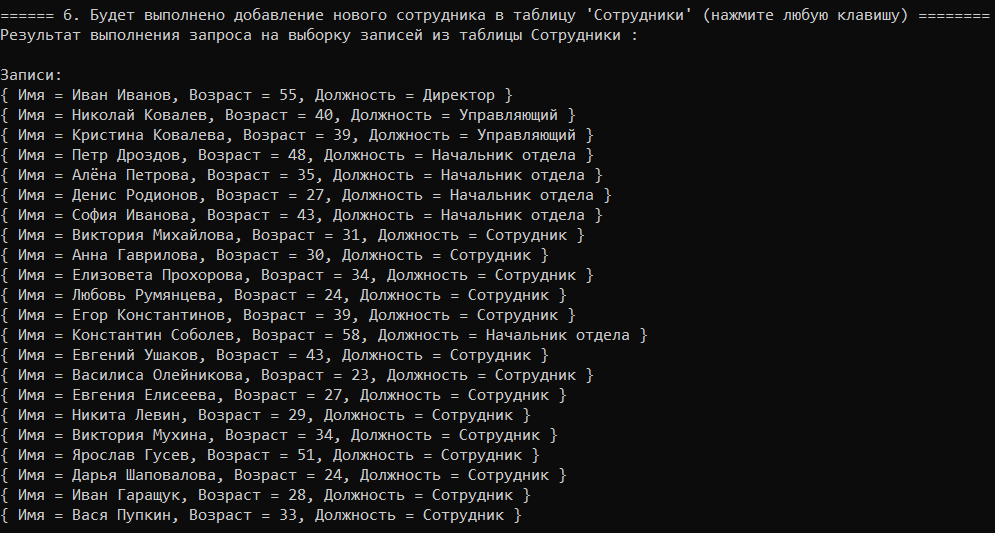


Рисунок 6 – Результат вставки данных в таблицу *Employees*

Для вставки данных в таблицу *ProvidedServices*, стоящей на стороне «Многие», был разработан метод *SeventhTask*. На рисунке 7 показан результат выполнения метода.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 7 – Вставка данных в таблицу *ProvidedServices*

Для удаления данных из таблицы *Employees,* стоящей на стороне «Один»,был разработан метод *EighthTask*. Результат выполнения метода показан на рисунке 8.

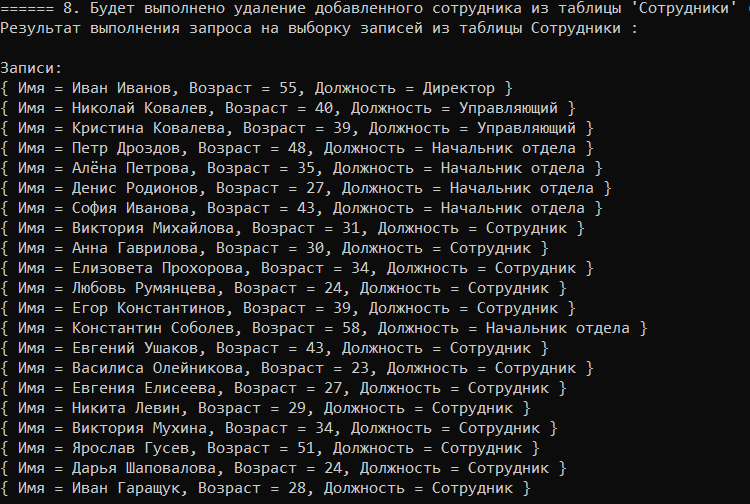


Рисунок 8 – Удаление данных из таблицы *Employees*

Для удаления данных из таблицы *ProvidedServices,* стоящей на стороне «многие», был разработан метод *NinthTask.* Пользователь вводит id теста для удаления. Результат выполнения метода показан на рисунке 9.

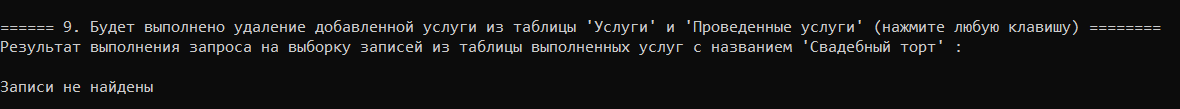


Рисунок 9 – Удаление данных из таблицы *ProvidedServices*

Для изменения ответов в таблицы *Employees* в определённом диапазоне *id* вопросов был разработан метод *TenthTask.* Пользователь вводит начальный *id* и конечный *id* вопросов*,* в пределах которого будет происходить изменение ответов: должность сотрудника поменялась на управляюещего. Результат выполнения метода показан на рисунке 10.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Обновление данных в таблице *Employees*

**Вывод:** в процессе работы ознакомились с разработкой программ с использованием *Entity* *Framework* и *LINQ*.

**Приложение А**

**(обязательное)**

**Листинг программы**

**Program.cs**

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json.Serialization;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Collections.Immutable;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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

namespace MarriageAgencyConsole

{

internal class Program

{

static void Main(string[] args)

{

using (MarriageAgencyContext db = new MarriageAgencyContext())

{

Console.WriteLine("====== 1. Будет выполнена выборка сотрудников (нажмите любую клавишу) ========");

Console.ReadKey();

SelectStaff(db);

Console.WriteLine("====== 2. Будет выполнена выборка клиентов старше 40 лет (нажмите любую клавишу) ========");

Console.ReadKey();

SelectClientsOlderThen(db, 40);

Console.WriteLine("====== 3. Будет выполнена выборка сотрудников, сгрупированная по количеству выполненных услуг (нажмите любую клавишу) ========");

Console.ReadKey();

SelectStaffWithNumberOfProvidedServices(db);

Console.ReadKey();

Console.WriteLine("====== 4. Будет выполнена выборка сотрудников и выполненных ими услуг (нажмите любую клавишу) ========");

Console.ReadKey();

SelectProvidedServicesWithServices(db);

Console.ReadKey();

Console.WriteLine("====== 5. Будет выполнена выборка сотрудников и выполненных ими услуг со стоимостью больше 250 (нажмите любую клавишу) ========");

Console.ReadKey();

SelectProvidedServicesWithServicesWhereCostBiggerThen(db, 250);

Console.ReadKey();

Console.WriteLine("====== 6. Будет выполнено добавление нового сотрудника в таблицу 'Сотрудники' (нажмите любую клавишу) ========");

Console.ReadKey();

InsertEmployee(db);

SelectStaff(db);

Console.ReadKey();

Console.WriteLine("====== 7. Будет выполнено добавление новой услуги в таблицу 'Услуги' и 'Проведенные услуги' (нажмите любую клавишу) ========");

Console.ReadKey();

InsertProvidedServices(db);

SelectProvidedServicesByServiceName(db, "Свадебный торт");

Console.ReadKey();

Console.WriteLine("====== 8. Будет выполнено удаление добавленного сотрудника из таблицы 'Сотрудники' (нажмите любую клавишу) ========");

Console.ReadKey();

DeleteEmployee(db);

SelectStaff(db);

Console.ReadKey();

Console.WriteLine("====== 9. Будет выполнено удаление добавленной услуги из таблицы 'Услуги' и 'Проведенные услуги' (нажмите любую клавишу) ========");

Console.ReadKey();

DeleteProvidedService(db);

SelectProvidedServicesByServiceName(db, "Свадебный торт");

Console.ReadKey();

Console.WriteLine("====== 10. Будет выполнено обновление должности сотрудника в таблице 'Сотрудники' (нажмите любую клавишу) ========");

Console.ReadKey();

UpdateEmployeePosition(db);

SelectStaff(db);

Console.ReadKey();

}

}

static void Print(string sqltext, IEnumerable items, int count)

{

Console.WriteLine(sqltext);

if (count == 0)

{

Console.WriteLine("Записи не найдены");

}

else

{

Console.WriteLine("Записи: ");

foreach (var item in items)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine();

}

Console.ReadKey();

}

static void SelectStaff(MarriageAgencyContext db)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from e in db.Employees

select new

{

Имя = e.Name,

Возраст = EF.Functions.DateDiffYear(e.Bithdate, currentDate),

Должность = e.Position

};

string comment = "Результат выполнения запроса на выборку записей из таблицы Сотрудники : \r\n";

Print(comment, queryLINQ1.ToList(), queryLINQ1.Count());

}

static void SelectClientsOlderThen(MarriageAgencyContext db, int age)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from c in db.Clients

where (EF.Functions.DateDiffYear(c.Bithdate, currentDate) >= age)

select new

{

Имя = c.Name,

Возраст = EF.Functions.DateDiffYear(c.Bithdate, currentDate),

Пол = c.Sex,

Количество\_детей = c.Children

};

string comment = "Результат выполнения запроса на выборку отсортированных записей из таблицы Клиенты : \r\n";

Print(comment, queryLINQ1.Take(5).ToList(), queryLINQ1.Count());

}

static void SelectStaffWithNumberOfProvidedServices(MarriageAgencyContext db)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from ps in db.ProvidedServices

group ps by ps.EmployeeId into gr

select new

{

EmployeeID = gr.Key,

Count = gr.Count()

};

var queryLINQ2 = from e in db.Employees

join q in queryLINQ1

on e.Id equals q.EmployeeID

select new

{

Имя = e.Name,

Возраст = EF.Functions.DateDiffYear(e.Bithdate, currentDate),

Должность = e.Position,

Выполненные\_услуги = q.Count

};

string comment = "Результат выполнения запроса на выборку записей из таблицы Сотрудники, сгруппированных по количеству выполненных услуг : \r\n";

Print(comment, queryLINQ2.Take(5).ToList(), queryLINQ2.Count());

}

static void SelectProvidedServicesWithServices(MarriageAgencyContext db)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from e in db.Employees

join ps in db.ProvidedServices

on e.Id equals ps.EmployeeId

join alls in db.AllServices

on ps.ServiceId equals alls.Id

select new

{

Имя = e.Name,

Возраст = EF.Functions.DateDiffYear(e.Bithdate, currentDate),

Должность = e.Position,

Услуга = alls.Name,

Дата = ps.Date.GetValueOrDefault().ToString("dd.MM.yyyy"),

Цена = alls.Cost

};

string comment = "Результат выполнения запроса на выборку записей из таблицы Сотрудники и выполненных ими услуг : \r\n";

Print(comment, queryLINQ1.Take(5).ToList(), queryLINQ1.Count());

}

static void SelectProvidedServicesWithServicesWhereCostBiggerThen(MarriageAgencyContext db, int cost)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from e in db.Employees

join ps in db.ProvidedServices

on e.Id equals ps.EmployeeId

join alls in db.AllServices

on ps.ServiceId equals alls.Id

where (alls.Cost >= cost)

select new

{

Имя = e.Name,

Возраст = EF.Functions.DateDiffYear(e.Bithdate, currentDate),

Должность = e.Position,

Услуга = alls.Name,

Дата = ps.Date.GetValueOrDefault().ToString("dd.MM.yyyy"),

Цена = alls.Cost

};

string comment = "Результат выполнения запроса на выборку записей из таблицы Сотрудники и выполненных ими услуг со стоимостью больше 250 : \r\n";

Print(comment, queryLINQ1.Take(5).ToList(), queryLINQ1.Count());

}

static void InsertEmployee(MarriageAgencyContext db)

{

Employee employee = new Employee

{

Name = "Вася Пупкин",

Position = "Сотрудник",

Bithdate = new DateTime(1990,10,17),

};

db.Employees.Add(employee);

db.SaveChanges();

}

static void InsertProvidedServices(MarriageAgencyContext db)

{

AllService service = new AllService

{

Name = "Свадебный торт",

Description = "Очень вкусный торт",

Cost = 15

};

db.AllServices.Add(service);

db.SaveChanges();

ProvidedService providedService = new ProvidedService

{

ClientId = 1,

Date = DateTime.Now,

ServiceId = service.Id,

EmployeeId = 1

};

db.ProvidedServices.Add(providedService);

db.SaveChanges();

}

static void SelectProvidedServicesByServiceName(MarriageAgencyContext db, string name)

{

DateTime currentDate = System.DateTime.Now;

var queryLINQ1 = from ps in db.ProvidedServices

join alls in db.AllServices

on ps.ServiceId equals alls.Id

join e in db.Employees

on ps.EmployeeId equals e.Id

join c in db.Clients

on ps.ClientId equals c.Id

where (alls.Name == name)

select new

{

Имя\_сотрудника = e.Name,

Возраст\_сотрудника = EF.Functions.DateDiffYear(e.Bithdate, currentDate),

Должность = e.Position,

Имя\_клиента = c.Name,

Возраст\_клиента = EF.Functions.DateDiffYear(c.Bithdate, currentDate),

Услуга = alls.Name,

Дата = ps.Date.GetValueOrDefault().ToString("dd.MM.yyyy"),

Цена = alls.Cost

};

string comment = "Результат выполнения запроса на выборку записей из таблицы выполненных услуг с названием 'Свадебный торт' : \r\n";

Print(comment, queryLINQ1.Take(5).ToList(), queryLINQ1.Count());

}

static void DeleteEmployee(MarriageAgencyContext db)

{

string name = "Вася Пупкин";

var employee = db.Employees.Where(e => e.Name == name);

db.Employees.RemoveRange(employee);

db.SaveChanges();

}

static void DeleteProvidedService(MarriageAgencyContext db)

{

string name = "Свадебный торт";

var service = db.AllServices.Where(alls => alls.Name == name);

var providedService = db.ProvidedServices

.Include("Service")

.Where(ps => (ps.Service.Name == name));

db.ProvidedServices.RemoveRange(providedService);

db.SaveChanges();

db.AllServices.RemoveRange(service);

db.SaveChanges();

}

static void UpdateEmployeePosition(MarriageAgencyContext db)

{

string name = "Константин Соболев";

var employee = db.Employees.Where(e => e.Name == name).FirstOrDefault();

if (employee != null)

{

employee.Position = "Сотрудник";

};

db.SaveChanges();

}

}

}