МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №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. ИТП-31
Бондарев Е.Ю.
Принял: ректор
Асенчик О.Д.

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.

Задание:

1.1. 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.

1.2. 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, count* 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.

1.3. Разместить выполненный проект на *github*.

Ход работы

В ходе выполнения лабораторной работы было создано консольное приложение на версии *.NET* 7.0. Далее для работы с базой данных при помощи *ENTITY FRAMEWORK,* который представляет собой объектно-

ориентированную, легковесную и расширяемую технологию от компании *Microsoft* для доступа к данным.

После установки всех необходимых пакетов приложение готово для работы с базой данных через *ENTITY FRAMEWORK*. В основе *ENTITY FRAMEWORK* лежит технология *ORM* (*object-relational mapping* – отображения данных на реальные объекты) которая позволят связать таблицы баз данных с *C#* объектами. Для связи объектов по определенным правилам существует три подхода.

Поскольку в предыдущей лабораторной работе уже была создана база данных, был использован подход *Database-first*. Для этого была применена команда *Scaffold-DbContext*, которая позволяет автоматически создать модели данных и контекст для работы с базой данных. Эта команда требует указания строки подключения к базе данных и провайдера (например, *SQL Server*) в качестве параметров. После выполнения этой команды были сгенерированы соответствующие классы моделей и класс контекста.

Для работы с базой данных был создан класс DbManager содержащий методы *SelectPublicationTypes, SelectPublicationTypesByType, SelectTotalSubscriptions, SelectEmployee, SelectEmployeeByOffice, InsertPublicationType, InsertEmployee, DeletePublicationType, DeleteEmployees, UpdatePublicationTypes*. Листинг данного класса указан в приложении А.

Первый метод *SelectPublicationTypes*, выполняет выборку данных из таблицы *PublicationTypes* которая находится на стороне отношения один. Пример работы этого метода указан на рисунке 1.

```
1. Выборка данных из таблицы на стороне отношения 'один'
-------------------------------------------------
Id: 1, Type: газета
Id: 2, Type: журнал
```

Рисунок 1 – Пример работы метода для выборки данных из таблицы на стороне отношения один

Далее был реализован метод *SelectPublicationTypesByType* для выборки данных из таблицы *PublicationType* которая находится на стороне отношения один с последующей фильтрацией данных по полю *Type*. Пример работы этого запроса указан на рисунке 2.

```
2. Выборка данных из таблицы на стороне отношения 'один' с фильтрацией данных
-------------------------------------------------
Id: 1, Type: газета
```

Рисунок 2 – Пример выполнения метода для выборки данных из таблицы на стороне отношения один с фильтрацией данных

Метод *SelectTotalSubscriptions* предназначен для подсчета количества подписок с равной продолжительностью из таблицы *Subscriptions* стоящей на

стороне отношения многие. Пример работы этого запроса указан на рисунке 3.

```
3. Выборка данных из таблицы на стороне отношения 'многие' с агрегацией данных
------------------------------------------------
Duration: 1, Total Subscriptions: 10
Duration: 2, Total Subscriptions: 10
Duration: 3, Total Subscriptions: 1
Duration: 4, Total Subscriptions: 7
Duration: 5, Total Subscriptions: 12
Duration: 6, Total Subscriptions: 9
Duration: 7, Total Subscriptions: 11
Duration: 8, Total Subscriptions: 7
Duration: 9, Total Subscriptions: 12
Duration: 10, Total Subscriptions: 6
Duration: 11, Total Subscriptions: 9
Duration: 12, Total Subscriptions: 6
```

Рисунок 3 – Пример запроса на выборку данных из таблицы на стороне отношения многие с последующей агрегацией данных

Далее был разработан метод *SelectEmployee* который выводит данные о работнике из таблицы *Employee* и его место работы из таблицы *Office*. Пример работы этого запроса указан на рисунке 4.

```
4. Выборка данных из таблиц на стороне отношения 'один-многие' с join
------------------------------------------------
OfficeName: Гагарина улица, EmployeeName: Иванов Иван Иванович
OfficeName: Гагарина улица, EmployeeName: Иванов Иван Иванович
OfficeName: Гагарина улица, EmployeeName: Иванов Иван Иванович
OfficeName: Гагарина улица, EmployeeName: Иванов Иван Иванович
OfficeName: Гагарина улица, EmployeeName: Иванов Иван Иванович
OfficeName: Севастопольская улица, EmployeeName: Петров Арсений Иванович
OfficeName: Севастопольская улица, EmployeeName: Петров Петр Андреевич
OfficeName: Молодежная улица, EmployeeName: Смирнов Кирил Сергеевна
OfficeName: Космонавтов улица, EmployeeName: Антонова Кирил Андреевич
OfficeName: Парковая улица, EmployeeName: Смирнов Арсений Петрович
OfficeName: Карла Маркса улица, EmployeeName: Смирнов Александр Андреевич
OfficeName: Карла Маркса улица, EmployeeName: Иванов Александр Петрович
OfficeName: Садовая улица, EmployeeName: Сидоров Евгений Сергеевна
OfficeName: Космическая улица, EmployeeName: Петров Иван Петрович
OfficeName: Космическая улица, EmployeeName: Антонова Евгений Петрович
OfficeName: Космическая улица, EmployeeName: Иванов Кирил Андреевич
OfficeName: Космическая улица, EmployeeName: Иванов Кирил Андреевич
OfficeName: Полярная улица, EmployeeName: Антонова Петр Сергеевна
OfficeName: Восточная улица, EmployeeName: Антонова Петр Иванович
OfficeName: Восточная улица, EmployeeName: Сидоров Иван Сергеевна
OfficeName: Фрунзе улица, EmployeeName: Антонова Иван Сергеевна
OfficeName: Фрунзе улица, EmployeeName: Антонова Евгений Петрович
OfficeName: Петровская улица, EmployeeName: Сидоров Евгений Александрович
OfficeName: Горная улица, EmployeeName: Антонова Иван Александрович
OfficeName: Горная улица, EmployeeName: Сидоров Петр Андреевич
OfficeName: Красноармейская улица, EmployeeName: Иванов Петр Петрович
OfficeName: Красноармейская улица, EmployeeName: Смирнов Арсений Сергеевна
OfficeName: Красноармейская улица, EmployeeName: Сидоров Петр Александрович
```

Рисунок 4 – Пример запроса на выборку данных из таблиц на стороне отношения один-многие

Метод *SelectEmployeeByOffice* предназначен для выборки данных из таблиц *Employee* и *Office*, стоящих на стороне отношения один-многие с последующей фильтрацией данных по названию офиса. Пример работы этого запроса указан на рисунке 5.

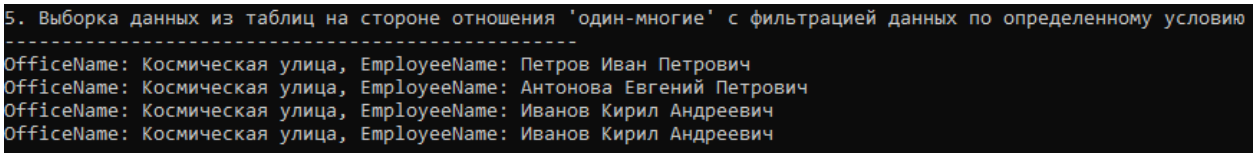

Рисунок 5 – Пример запроса на выборку данных из таблиц на стороне отношения один-многие с последующей фильтрацией данных

Далее был разработан метод *InsertPublicationType* на добавления данных в таблицу *PublicationType*, стоящую на стороне отношения один. Пример работы этого запроса указан на рисунке 6.

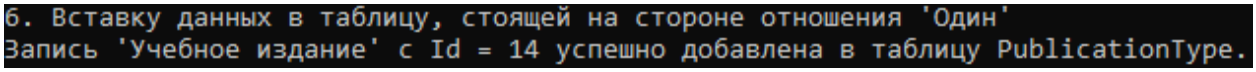

Рисунок 6 – Пример запроса на вставку данных в таблицу на стороне отношения один

Далее был разработан метод *InsertEmployee* для добавления данных в таблицу *Employee*, стоящую на стороне отношения многие. Пример работы этого запроса указан на рисунке 7.

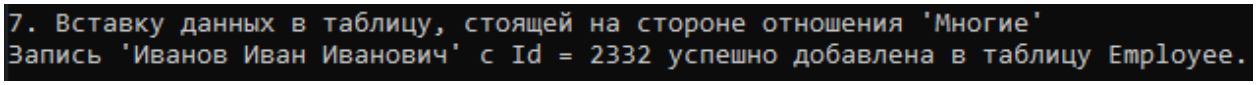

Рисунок 7 – Пример запроса на вставку данных в таблицу на стороне отношения многие

Далее был реализован метод *DeletePublicationType* для удаления данных из таблицы *PublicationType*, стоящей на стороне отношения многие. Пример работы этого запроса указан на рисунке 7.

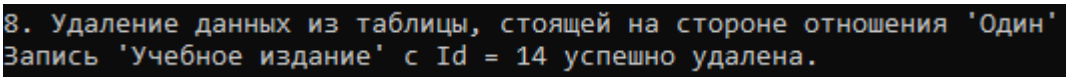

Рисунок 7 – Пример удаления данных на стороне отношения один

Далее был реализован метод *DeleteEmployees* для удаления данных из таблицы *Employees*, стоящей на стороне отношения многие. Пример работы этого запроса указан на рисунке 8.

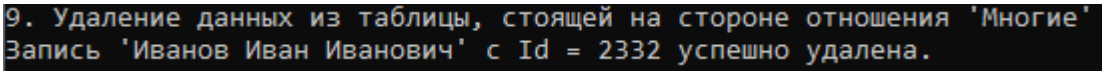

Рисунок 8 – Пример удаления данных на стороне отношения многие

Далее был разработан метод *UpdatePublicationTypes* для обновления данных в таблице *PublicationTypes*. Пример работы этого запроса указан на рисунке 9.

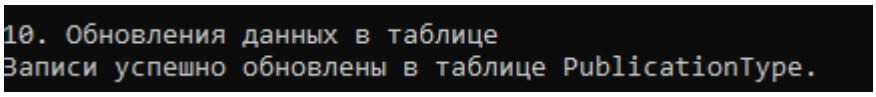

Рисунок 9 – Пример обновления данных в базе данных

Лабораторная работа размещена на *GitHub* по адресу - https://github.com/EvgeniBondarev/DDBAISE

Вывод: в результате выполнения данной лабораторной работы мы приобрели знания о технологии ENTITY FRAMEWORK, которая обеспечивает интеграцию базы данных с языком программирования C#. Кроме того, мы ознакомились с технологией LINQ, предоставляющей удобные средства для работы с данными в контексте объектно-реляционного отображения (ORM).

ПРИЛОЖНИЕ А

Листинг класса *Employee*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class Employee
{
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string Middlename { get; set; } = null!;

    public string Surname { get; set; } = null!;

    public int PositionId { get; set; }

    public int OfficeId { get; set; }

    public virtual Office Office { get; set; } = null!;

    public virtual EmployeePosition Position { get; set; } = null!;
}
```

Листинг класса *EmployeePosition*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class EmployeePosition
{
    public int Id { get; set; }

    public string? Position { get; set; }

    public virtual ICollection<Employee> Employees { get; set; } = new
List<Employee>();
}
```

Листинг класса *Office*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class Office
{
    public int Id { get; set; }

    public string OwnerName { get; set; } = null!;

    public string OwnerMiddlename { get; set; } = null!;

    public string OnwnerSurname { get; set; } = null!;

    public string StreetName { get; set; } = null!;

    public string MobilePhone { get; set; } = null!;
```

```
    public string Email { get; set; } = null!;

    public virtual ICollection<Employee> Employees { get; set; } = new
List<Employee>();

    public virtual ICollection<Subscription> Subscriptions { get; set; } = new
List<Subscription>();
}
```

Листинг класса *Publication*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class Publication
{
    public int Id { get; set; }

    public int TypeId { get; set; }

    public string Name { get; set; } = null!;

    public decimal Price { get; set; }

    public virtual ICollection<Subscription> Subscriptions { get; set; } = new
List<Subscription>();

    public virtual PublicationType Type { get; set; } = null!;
}
```

Листинг класса *PublicationType*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class PublicationType
{
    public int Id { get; set; }

    public string Type { get; set; } = null!;

    public virtual ICollection<Publication> Publications { get; set; } = new
List<Publication>();
}
```

Листинг класса *Recipient*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class Recipient
{
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string Middlename { get; set; } = null!;

    public string Surname { get; set; } = null!;
```

```
    public int AddressId { get; set; }

    public string MobilePhone { get; set; } = null!;

    public string Email { get; set; } = null!;

    public virtual RecipientAddress Address { get; set; } = null!;

    public virtual ICollection<Subscription> Subscriptions { get; set; } = new
List<Subscription>();
}
```

Листинг класса *RecipientAddress*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class RecipientAddress
{
    public int Id { get; set; }

    public string? Street { get; set; }

    public int? House { get; set; }

    public int? Apartment { get; set; }

    public virtual ICollection<Recipient> Recipients { get; set; } = new
List<Recipient>();
}
```

Листинг класса *Subscription*

```
using System;
using System.Collections.Generic;

namespace Lab2;

public partial class Subscription
{
    public int Id { get; set; }

    public int RecipientId { get; set; }

    public int PublicationId { get; set; }

    public int Duration { get; set; }

    public int OfficeId { get; set; }

    public string SubscriptionStartDate { get; set; } = null!;

    public virtual Office Office { get; set; } = null!;

    public virtual Publication Publication { get; set; } = null!;

    public virtual Recipient Recipient { get; set; } = null!;
}
```

Листинг класса *SubsCityContext*

```
using System;
```

```
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

namespace Lab2;

public partial class SubsCityContext : DbContext
{
    public SubsCityContext()
    {
    }

    public SubsCityContext(DbContextOptions<SubsCityContext> options)
        : base(options)
    {
    }

    public virtual DbSet<Employee> Employees { get; set; }

    public virtual DbSet<EmployeePosition> EmployeePositions { get; set; }

    public virtual DbSet<Office> Offices { get; set; }

    public virtual DbSet<OfficeView> OfficeViews { get; set; }

    public virtual DbSet<Publication> Publications { get; set; }

    public virtual DbSet<PublicationType> PublicationTypes { get; set; }

    public virtual DbSet<PublicationView> PublicationViews { get; set; }

    public virtual DbSet<Recipient> Recipients { get; set; }

    public virtual DbSet<RecipientAddress> RecipientAddresses { get; set; }

    public virtual DbSet<RecipientView> RecipientViews { get; set; }

    public virtual DbSet<Subscription> Subscriptions { get; set; }

    public virtual DbSet<SubscriptionView> SubscriptionViews { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
#warning To protect potentially sensitive information in your connection string, you
should move it out of source code. You can avoid scaffolding the connection string by
using the Name= syntax to read it from configuration - see
https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection
strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
        => optionsBuilder.UseSqlServer("Server=DESKTOP-
QAU182Q\\SQLEXPRESS;Database=SubsCity;Trusted_Connection=True;
TrustServerCertificate=True;");

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Employee>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Employee__3213E83FBA716D56");

            entity.ToTable("Employee");

            entity.Property(e => e.Id).HasColumnName("id");
            entity.Property(e => e.Middlename)
```

```
                .HasMaxLength(20)
                .HasColumnName("middlename");
            entity.Property(e => e.Name)
                .HasMaxLength(20)
                .HasColumnName("name");
            entity.Property(e => e.OfficeId).HasColumnName("office_id");
            entity.Property(e => e.PositionId).HasColumnName("position_id");
            entity.Property(e => e.Surname)
                .HasMaxLength(20)
                .HasColumnName("surname");

            entity.HasOne(d => d.Office).WithMany(p => p.Employees)
                .HasForeignKey(d => d.OfficeId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Employee__office__49C3F6B7");

            entity.HasOne(d => d.Position).WithMany(p => p.Employees)
                .HasForeignKey(d => d.PositionId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Employee__positi__4AB81AF0");
        });

        modelBuilder.Entity<EmployeePosition>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Employee__3213E83F167E0439");

            entity.ToTable("EmployeePosition");

            entity.HasIndex(e => e.Position,
"UQ__Employee__75FE9D9930B8A4EF").IsUnique();

            entity.Property(e => e.Id).HasColumnName("id");
            entity.Property(e => e.Position)
                .HasMaxLength(50)
                .HasColumnName("position");
        });

        modelBuilder.Entity<Office>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Office__3213E83F15657E01");

            entity.ToTable("Office");

            entity.HasIndex(e => e.MobilePhone,
"UQ__Office__3867605B3F8D03F9").IsUnique();

            entity.HasIndex(e => e.Email, "UQ__Office__AB6E6164EC42B120").IsUnique();

            entity.Property(e => e.Id).HasColumnName("id");
            entity.Property(e => e.Email)
                .HasMaxLength(255)
                .HasColumnName("email");
            entity.Property(e => e.MobilePhone)
                .HasMaxLength(20)
                .HasColumnName("mobile_phone");
            entity.Property(e => e.OnwnerSurname)
                .HasMaxLength(20)
                .HasColumnName("onwner_surname");
            entity.Property(e => e.OwnerMiddlename)
                .HasMaxLength(20)
```

```
            .HasColumnName("owner_middlename");
        entity.Property(e => e.OwnerName)
            .HasMaxLength(20)
            .HasColumnName("owner_name");
        entity.Property(e => e.StreetName)
            .HasMaxLength(50)
            .HasColumnName("street_name");
    });

    modelBuilder.Entity<OfficeView>(entity =>
    {
        entity
            .HasNoKey()
            .ToView("OfficeView");

        entity.Property(e => e.Email).HasMaxLength(255);
        entity.Property(e => e.MobilePhone).HasMaxLength(20);
        entity.Property(e => e.OfficeId)
            .ValueGeneratedOnAdd()
            .HasColumnName("OfficeID");
        entity.Property(e => e.OwnerMiddlename).HasMaxLength(20);
        entity.Property(e => e.OwnerName).HasMaxLength(20);
        entity.Property(e => e.OwnerSurname).HasMaxLength(20);
        entity.Property(e => e.StreetName).HasMaxLength(50);
    });

    modelBuilder.Entity<Publication>(entity =>
    {
        entity.HasKey(e => e.Id).HasName("PK__Publicat__3213E83FBFFED7F5");

        entity.ToTable("Publication");

        entity.Property(e => e.Id).HasColumnName("id");
        entity.Property(e => e.Name)
            .HasMaxLength(70)
            .HasColumnName("name");
        entity.Property(e => e.Price)
            .HasColumnType("decimal(10, 2)")
            .HasColumnName("price");
        entity.Property(e => e.TypeId).HasColumnName("type_id");

        entity.HasOne(d => d.Type).WithMany(p => p.Publications)
            .HasForeignKey(d => d.TypeId)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("FK__Publicati__type___3E52440B");
    });

    modelBuilder.Entity<PublicationType>(entity =>
    {
        entity.HasKey(e => e.Id).HasName("PK__Publicat__3213E83FF4660137");

        entity.ToTable("PublicationType");

        entity.Property(e => e.Id).HasColumnName("id");
        entity.Property(e => e.Type)
            .HasMaxLength(20)
            .HasColumnName("type");
    });

    modelBuilder.Entity<PublicationView>(entity =>
```

```
        {
            entity
                .HasNoKey()
                .ToView("PublicationView");

            entity.Property(e => e.PublicationId).HasColumnName("PublicationID");
            entity.Property(e => e.PublicationName).HasMaxLength(70);
            entity.Property(e => e.PublicationPrice).HasColumnType("decimal(10, 2)");
            entity.Property(e => e.PublicationType).HasMaxLength(20);
        });

        modelBuilder.Entity<Recipient>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Recipien__3213E83F8B16D66F");

            entity.ToTable("Recipient");

            entity.HasIndex(e => e.MobilePhone,
"UQ__Recipien__3867605B8CDDE220").IsUnique();

            entity.HasIndex(e => e.Email, "UQ__Recipien__AB6E6164F25EBCDE").IsUnique();

            entity.Property(e => e.Id).HasColumnName("id");
            entity.Property(e => e.AddressId).HasColumnName("address_id");
            entity.Property(e => e.Email)
                .HasMaxLength(255)
                .HasColumnName("email");
            entity.Property(e => e.Middlename)
                .HasMaxLength(20)
                .HasColumnName("middlename");
            entity.Property(e => e.MobilePhone)
                .HasMaxLength(20)
                .HasColumnName("mobile_phone");
            entity.Property(e => e.Name)
                .HasMaxLength(20)
                .HasColumnName("name");
            entity.Property(e => e.Surname)
                .HasMaxLength(20)
                .HasColumnName("surname");

            entity.HasOne(d => d.Address).WithMany(p => p.Recipients)
                .HasForeignKey(d => d.AddressId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Recipient__addre__4316F928");
        });

        modelBuilder.Entity<RecipientAddress>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Recipien__3213E83FEFE7DA13");

            entity.ToTable("RecipientAddress");

            entity.Property(e => e.Id).HasColumnName("id");
            entity.Property(e => e.Apartment).HasColumnName("apartment");
            entity.Property(e => e.House).HasColumnName("house");
            entity.Property(e => e.Street)
                .HasMaxLength(50)
                .HasColumnName("street");
        });
```

```
modelBuilder.Entity<RecipientView>(entity =>
{
    entity
        .HasNoKey()
        .ToView("RecipientView");

    entity.Property(e => e.RecipientEmail).HasMaxLength(255);
    entity.Property(e => e.RecipientId).HasColumnName("RecipientID");
    entity.Property(e => e.RecipientMiddlename).HasMaxLength(20);
    entity.Property(e => e.RecipientMobilePhone).HasMaxLength(20);
    entity.Property(e => e.RecipientName).HasMaxLength(20);
    entity.Property(e => e.RecipientStreet).HasMaxLength(50);
    entity.Property(e => e.RecipientSurname).HasMaxLength(20);
});

modelBuilder.Entity<Subscription>(entity =>
{
    entity.HasKey(e => e.Id).HasName("PK__Subscrip__3213E83FEB906352");

    entity.ToTable("Subscription");

    entity.Property(e => e.Id).HasColumnName("id");
    entity.Property(e => e.Duration).HasColumnName("duration");
    entity.Property(e => e.OfficeId).HasColumnName("office_id");
    entity.Property(e => e.PublicationId).HasColumnName("publication_id");
    entity.Property(e => e.RecipientId).HasColumnName("recipient_id");
    entity.Property(e => e.SubscriptionStartDate)
        .HasMaxLength(7)
        .HasColumnName("subscription_start_date");

    entity.HasOne(d => d.Office).WithMany(p => p.Subscriptions)
        .HasForeignKey(d => d.OfficeId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK__Subscript__offic__4D94879B");

    entity.HasOne(d => d.Publication).WithMany(p => p.Subscriptions)
        .HasForeignKey(d => d.PublicationId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK__Subscript__publi__4E88ABD4");

    entity.HasOne(d => d.Recipient).WithMany(p => p.Subscriptions)
        .HasForeignKey(d => d.RecipientId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK__Subscript__recip__4F7CD00D");
});

modelBuilder.Entity<SubscriptionView>(entity =>
{
    entity
        .HasNoKey()
        .ToView("SubscriptionView");

    entity.Property(e => e.OfficeOwnerName).HasMaxLength(20);
    entity.Property(e => e.PublicationName).HasMaxLength(70);
    entity.Property(e => e.RecipientName).HasMaxLength(20);
    entity.Property(e => e.SubscriptionId).HasColumnName("SubscriptionID");
    entity.Property(e => e.SubscriptionStartDate).HasMaxLength(7);
});

OnModelCreatingPartial(modelBuilder);
```

```
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

Листинг класса *DbManager*

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab2
{
    internal class DbManager
    {
        public void SelectPublicationTypes()
        {
            Console.WriteLine("1. Выборка данных из таблицы на стороне отношения
'один'");
            Console.WriteLine(new string('-', 50));
            using (var dbContext = new SubsCityContext())
            {
                var publicationTypes = dbContext.PublicationTypes.ToList();

                foreach (var publicationType in publicationTypes)
                {
                    Console.WriteLine($"Id: {publicationType.Id}, Type:
{publicationType.Type}");
                }
            }
            Console.WriteLine("\n\n");
        }
        public void SelectPublicationTypesByType(string targetType)
        {
            Console.WriteLine("2. Выборка данных из таблицы на стороне отношения
'один' с фильтрацией данных");
            Console.WriteLine(new string('-', 50));

            using (var dbContext = new SubsCityContext())
            {
                var filteredPublicationTypes = dbContext.PublicationTypes
                    .Where(pt => pt.Type == targetType)
                    .ToList();

                foreach (var publicationType in filteredPublicationTypes)
                {
                    Console.WriteLine($"Id: {publicationType.Id}, Type:
{publicationType.Type}");
                }
            }
            Console.WriteLine("\n\n");
        }
        public void SelectTotalSubscriptions()
        {
            Console.WriteLine("3. Выборка данных из таблицы на стороне отношения
'многие' с агрегацией данных");
            Console.WriteLine(new string('-', 50));
            using (var dbContext = new SubsCityContext())
            {

                var groupedSubscriptions = dbContext.Subscriptions
                    .GroupBy(s => s.Duration)
                    .Select(g => new
                    {
```

```
                    Duration = g.Key,
                    TotalSubscriptions = g.Count()
                })
                .ToList();

            foreach (var group in groupedSubscriptions)
            {
                Console.WriteLine($"Duration: {group.Duration}, Total
Subscriptions: {group.TotalSubscriptions}");
            }
        }
        Console.WriteLine("\n\n");
    }
    public void SelectEmployee()
    {
        Console.WriteLine("4. Выборка данных из таблиц на стороне отношения
'один-многие' с join");
        Console.WriteLine(new string('-', 50));

        using (var dbContext = new SubsCityContext())
        {
            var query = dbContext.Offices
                .Join(dbContext.Employees,
                    office => office.Id,
                    employee => employee.OfficeId,
                    (office, employee) => new
                    {
                        OfficeName = office.StreetName,
                        EmployeeName = $"{employee.Surname} {employee.Name}
{employee.Middlename}"
                    });

            foreach (var result in query)
            {
                Console.WriteLine($"OfficeName: {result.OfficeName},
EmployeeName: {result.EmployeeName}");
            }
        }
        Console.WriteLine("\n\n");
    }

    public void SelectEmployeeByOffice(string targetOfficeName)
    {
        Console.WriteLine("5. Выборка данных из таблиц на стороне отношения
'один-многие' с фильтрацией данных по определенному условию");
        Console.WriteLine(new string('-', 50));

        using (var dbContext = new SubsCityContext())
        {
            var query = dbContext.Offices
                .Join(dbContext.Employees,
                    office => office.Id,
                    employee => employee.OfficeId,
                    (office, employee) => new { Office = office, Employee =
employee })
                .Where(joinResult => joinResult.Office.StreetName ==
targetOfficeName)
                .Select(joinResult => new
                {
                    EmployeeName = $"{joinResult.Employee.Surname}
{joinResult.Employee.Name} {joinResult.Employee.Middlename}",
                    OfficeName = joinResult.Office.StreetName
                });

            foreach (var result in query)
```

```
                {
                    Console.WriteLine($"OfficeName: {result.OfficeName},
EmployeeName: {result.EmployeeName}");
                }
            }
            Console.WriteLine("\n\n");
        }

        public void InsertPublicationType(PublicationType newPublicationType)
        {
            Console.WriteLine("6. Вставку данных в таблицу, стоящей на стороне
отношения 'Один'");
            using (var dbContext = new SubsCityContext())
            {



                dbContext.PublicationTypes.Add(newPublicationType);

                dbContext.SaveChanges();

                Console.WriteLine($"Запись '{newPublicationType.Type}' с Id =
{newPublicationType.Id} успешно добавлена в таблицу PublicationType.");
            }
            Console.WriteLine("\n\n");
        }
        public void InsertEmployee(Employee newEmployee)
        {
            Console.WriteLine("7. Вставку данных в таблицу, стоящей на стороне
отношения 'Многие'");
            using (var dbContext = new SubsCityContext())
            {

                dbContext.Employees.Add(newEmployee);

                dbContext.SaveChanges();

                Console.WriteLine($"Запись '{newEmployee.Surname} {newEmployee.Name}
{newEmployee.Middlename}' с Id = {newEmployee.Id} успешно добавлена в таблицу
Employee.");
            }
            Console.WriteLine("\n\n");
        }
        public void DeletePublicationType(PublicationType publicationTypeToDelete)
        {
            Console.WriteLine("8. Удаление данных из таблицы, стоящей на стороне
отношения 'Один'");
            using (var dbContext = new SubsCityContext())
            {

                var existingPublicationType = dbContext.PublicationTypes
            .FirstOrDefault(pt => pt.Id == publicationTypeToDelete.Id && pt.Type ==
publicationTypeToDelete.Type);

                if (existingPublicationType != null)
                {

                    dbContext.PublicationTypes.Remove(existingPublicationType);
                    dbContext.SaveChanges();

                    Console.WriteLine($"Запись '{publicationTypeToDelete.Type}' с Id
= {publicationTypeToDelete.Id} успешно удалена.");
                }
            }
            Console.WriteLine("\n\n");
```

```
        }
        public void DeleteEmployees(Employee employeeToDelete)
        {
            Console.WriteLine("9. Удаление данных из таблицы, стоящей на стороне
отношения 'Многие'");
            using (var dbContext = new SubsCityContext())
            {
                var existingEmployee = dbContext.Employees
                    .FirstOrDefault(e => e.Id == employeeToDelete.Id &&
                                         e.Name == employeeToDelete.Name &&
                                         e.Middlename ==
employeeToDelete.Middlename);

                        if (existingEmployee != null)
                        {

                            dbContext.Employees.Remove(existingEmployee);
                            dbContext.SaveChanges();

                            Console.WriteLine($"Запись '{existingEmployee.Surname}
{existingEmployee.Name} {existingEmployee.Middlename}' с Id = {existingEmployee.Id}
успешно удалена.");
                        }
            }
            Console.WriteLine("\n\n");
        }
        public void UpdatePublicationTypes(string condition, string newType)
        {
            Console.WriteLine("10. Обновления данных в таблице");
            using (var dbContext = new SubsCityContext())
            {
                var publicationTypesToUpdate = dbContext.PublicationTypes
                            .Where(pt => pt.Type == condition);

                foreach (var publicationType in publicationTypesToUpdate)
                {
                    publicationType.Type = newType;
                }

                dbContext.SaveChanges();

                Console.WriteLine("Записи успешно обновлены в таблице
PublicationType.");
            }

            Console.WriteLine("\n\n");
        }
    }
}
```

Листинг класса *Program*

```
namespace Lab2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            DbManager dbManager = new DbManager();

            dbManager.SelectPublicationTypes();
            dbManager.SelectPublicationTypesByType("Газета");
            dbManager.SelectTotalSubscriptions();
            dbManager.SelectEmployee();
            dbManager.SelectEmployeeByOffice("Космическая улица");
```

```
            PublicationType publication = new PublicationType()
            {
                Type = "Учебное издание"
            };
            dbManager.InsertPublicationType(publication);

            Employee employee = new Employee()
            {
                Name = "Иван",
                Middlename = "Иванович",
                Surname = "Иванов",
                OfficeId = 1,
                PositionId = 1
            };
            dbManager.InsertEmployee(employee);
            dbManager.DeletePublicationType(publication);
            dbManager.DeleteEmployees(employee);
            dbManager.UpdatePublicationTypes("Учебное издание", "Новое издание");
        }
    }
}
```