

Sprawozdanie – Entity Framework

Autor: Adrian Żerebiec

Spis treści

1. Efekt pracy na laboratorium	3
1.1 Klasa Program	3
1.2 Klasa ProductContext	3
1.3 Klasa Product	4
1.4 Pliki	4
1.5 Wynik działania programu	4
2. Zadanie domowe	5
2.1 Zadanie 1	5
2.1.1 Klasa Supplier	5
2.1.2 Klasa Product	5
2.1.3 Klasa ProductContext	6
2.1.4 Klasa Program	6
2.1.5 Diagram bazy danych	8
2.1.6 Przykład działania	8
2.1.7 Tabele	9
2.2 Zadanie 2	10
2.2.1 Klasa Supplier	10
2.2.2 Klasa Product	10
2.2.3 Klasa ProductContext	10
2.2.4 Klasa Program	11
2.2.5 Diagram bazy danych	13
2.2.6 Przykład działania	13
2.2.7 Tabele	14
2.3 Zadanie 3	14
2.3.1 Klasa Supplier	14
2.3.2 Klasa Product	15
2.3.3 Klasa ProductContext	15
2.3.4 Klasa Program	15
2.3.5 Diagram bazy danych	17
2.3.6 Przykład działania	17
2.3.7 Tabele	18

2.4 Zadanie 4	18
2.4.1 Klasa Product	18
2.4.2 Klasa Invoice	18
2.4.3 Klasa InvoiceHelper	19
2.4.4 Klasa ProductContext	19
2.4.5 Klasa Program	20
2.4.6 Diagram bazy danych.....	26
2.4.7 Przykład działania	26
2.4.8 Tabele	28
2.5 Zadanie 5	30
2.5.1 Klasa Customer	30
2.5.2 Klasa Supplier	30
2.5.3 Klasa Company	31
2.5.4 Klasa CompanyType.....	31
2.5.5 Klasa CompanyContext.....	31
2.5.6 Klasa Program	32
2.5.7 Diagram bazy danych.....	35
2.5.8 Przykład działania	35
2.5.9 Tabele	36
2.6 Zadanie 6	36
2.6.1 Klasa Customer	36
2.6.2 Klasa Supplier	36
2.6.3 Pozostałe klasy	36
2.6.4 Diagram bazy danych.....	37
2.6.5 Przykład działania	37
2.6.6 Tabele	37
2.6.7 Porównanie dziedziczenia Table-Per-Hierarchy i Table-Per-Type	38
2.7 Końcowe pliki.....	38

1.Efekt pracy na laboratorium

1.1Klasa Program

```
using System;
using System.Collections.Generic;
using System.Linq;
namespace AdrianZerebiecEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Podaj nazwę produktu");
            string prodName = Console.ReadLine();
            Console.WriteLine("Poniżej lista produktów zarejestrowanych w naszej bazie danych");

            ProductContext productContext = new ProductContext();
            Product product = new Product { ProductName = prodName };
            productContext.Products.Add(product);
            productContext.SaveChanges();

            var query = from prod in productContext.Products
                        select prod.ProductName;
            foreach (var pName in query)
            {
                Console.WriteLine(pName);
            }
        }
    }
}
```

1.2Klasa ProductContext

```
using System;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;

namespace AdrianZerebiecEFProducts
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }

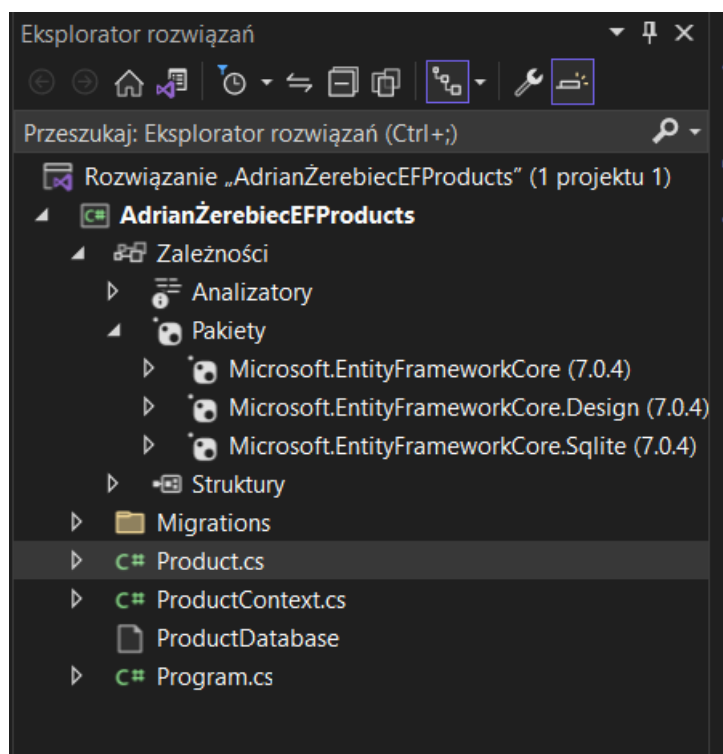
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=ProductDatabase");
        }
    }
}
```

1.3 Klasa Product

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

namespace AdrianŻerebiecEFProducts
{
    Odwołania: 3
    public class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 2
        public string ProductName { get; set; }
        Odwołania: 0
        public int UnitOnStock { get; set; }
    }
}
```

1.4 Pliki



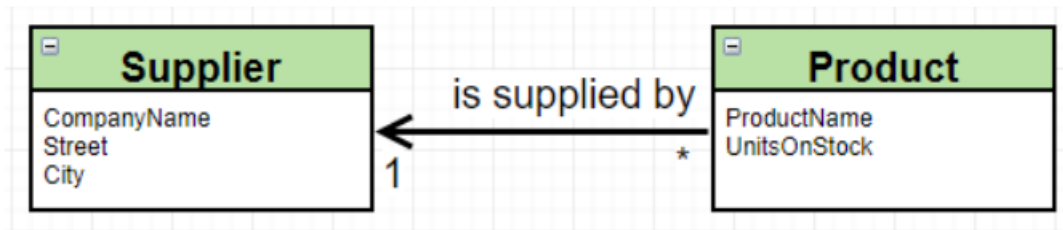
1.5 Wynik działania programu

```
Konsola debugowania programu Microsoft Visual Studio
Podaj nazwę produktu
Flamaster
Poniżej lista produktów zarejestrowanych w naszej bazie danych
Flamaster
Flamaster
Flamaster
Flamaster
```

2. Zadanie domowe

2.1 Zadanie 1

Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



2.1.1 Klasa Supplier

```
namespace AdrianZerebiecEFProducts
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public override string ToString()
        {
            return CompanyName;
        }
    }
}
```

2.1.2 Klasa Product

```
namespace AdrianZerebiecEFProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitOnStock { get; set; }
        public Supplier? Supplier { get; set; } = null;
        public override string ToString()
        {
            return $"{ProductName} ({UnitOnStock} szt.)";
        }
    }
}
```

2.1.3 Klasa ProductContext

```
namespace AdrianZerebiecEFProducts
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductDatabase");
        }
    }
}
```

2.1.4 Klasa Program

```
namespace AdrianZerebiecEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();
            Product product = createNewProduct();
            Supplier supplier = null;

            bool createdSupplier = false;
            bool isValidChoice = false;
            while (!isValidChoice)
            {
                Console.WriteLine("Czy chcesz dodać nowego dostawcę?");
                string choice = Console.ReadLine();
                if (choice == "tak")
                {
                    isValidChoice = true;
                    supplier = createNewSupplier();
                    createdSupplier = true;
                    break;
                }
                else if (choice == "nie")
                {
                    isValidChoice = true;
                    Console.WriteLine("Lista wszystkich dostawców");
                    foreach (Supplier suppliers in context.Suppliers)
                    {
                        Console.WriteLine($"[{suppliers.SupplierID}]
{suppliers}");
                    }
                    supplier = chooseSupplier(context);
                    break;
                }
                else
                {
                    Console.WriteLine("Niepoprawna odpowiedź");
                    Console.WriteLine("Spróbuj ponownie");
                }
            }
            product.Supplier = supplier;
            Console.WriteLine("Dodano dostawcę do produktu");
            if (createdSupplier)
```

```

        {
            context.Suppliers.Add(supplier);
        }
        if (supplier != null) //wcześniej zawsze dodawałem stąd przesunięcie
w indeksach gdyż usuwałem ręcznie złe dane
        {
            context.Products.Add(product);
            context.SaveChanges();
        }
    }

    private static Product createNewProduct()
    {
        Console.WriteLine("Podaj nazwę produktu: ");
        string productName = Console.ReadLine();
        Console.WriteLine("Podaj ile sztuk produktu jest dostępnych: ");
        int units = Int32.Parse(Console.ReadLine());
        Console.WriteLine("Trwa tworzenie produktu");
        Product product = new Product
        {
            ProductName = productName,
            UnitOnStock = units
        };
        Console.WriteLine($"Stworzono produkt: {product}");
        return product;
    }

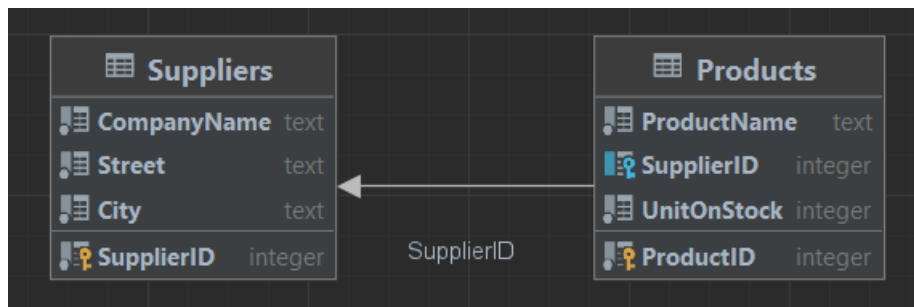
    private static Supplier createNewSupplier()
    {
        Console.WriteLine("Podaj nazwę dostawcy:");
        string companyName = Console.ReadLine();
        Console.WriteLine("Podaj miasto:");
        string city = Console.ReadLine();
        Console.WriteLine("Podaj ulicę:");
        string street = Console.ReadLine();
        Console.WriteLine("Trwa tworzenie dostawcy");
        Supplier supplier = new Supplier
        {
            CompanyName = companyName,
            City = city,
            Street = street
        };
        Console.WriteLine($"Stworzono dostawcę: {supplier}");
        return supplier;
    }

    private static Supplier chooseSupplier(ProductContext productContext)
    {
        Console.WriteLine("Podaj ID wybranego dostawcy: ");
        if (int.TryParse(Console.ReadLine(), out int choice))
        {
            return productContext.Suppliers.FirstOrDefault(s => s.SupplierID ==
choice);
        }

        Console.WriteLine("Niepoprawny wybór");
        return null;
    }
}

```

2.1.5 Diagram bazy danych



2.1.6 Przykład działania

```
Podaj nazwę produktu:
Ołówek
Podaj ile sztuk produktu jest dostępnych:
76
Trwa tworzenie produktu
Stworzono produkt: Ołówek (76 szt.)
Czy chcesz dodać nowego dostawcę?
tak
Podaj nazwę dostawcy: Ołówkowe Imperium
Podaj miasto: Warszawa
Podaj ulicę: Grunwaldzka
Trwa tworzenie dostawcy
Stworzono dostawcę: Ołówkowe Imperium
Dodano dostawcę do produktu
```

```
Podaj nazwę produktu:
Lipton
Podaj ile sztuk produktu jest dostępnych:
96
Trwa tworzenie produktu
Stworzono produkt: Lipton (96 szt.)
Czy chcesz dodać nowego dostawcę?
nie
Lista wszystkich dostawców
[1] Rowerex
[2] Rowerex
[3] Rowers
[4] Castrol
[5] Kredex
[6] Napojex
[7] Media RTV
[8] Hurtownia kredek
[9] Ołówkowe Imperium
Podaj ID wybranego dostawcy: 6
Dodano dostawcę do produktu
```


2.1.7 Tabele

Tabela Suppliers

	SupplierID	CompanyName	Street	City
1	1	Rowerex	Zarzeczna	Łódź
2	2	Rowerex	Pomorska	Łódź
3	3	Rowers	Nocna	Szczecin
4	4	Castrol	Mickiewicza	Kraków
5	5	Kredex	Polna	Łódź
6	6	Napojex	Grunwaldzka	Lublin
7	7	Media RTV	Fordon	Bydgoszcz
8	8	Hurtownia kredok	Poznańska	Toruń
9	9	Ołówkowe Imperium	Grunwaldzka	Warszawa

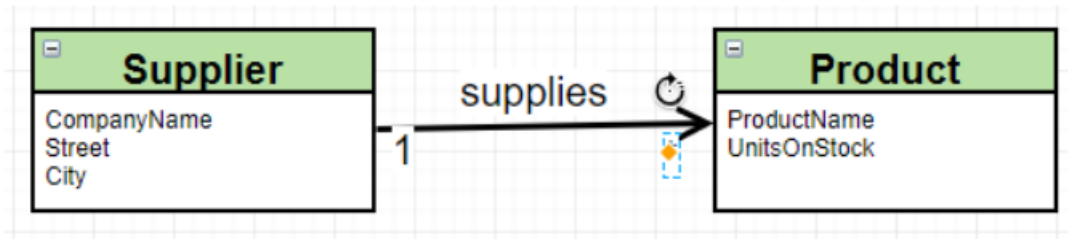
Tabele Products

	ProductID	ProductName	SupplierID	UnitOnStock
2	2	Rower górski	1	12
3	3	Rower górski	2	44
4	4	Rower	3	5
5	5	rower	4	44
6	6	Kredka	5	4
7	7	Coca-Cola	6	54
8	8	Laptop Lenovo	7	4
9	9	Kredka	8	41
10	10	Pióro	8	77
11	11	Rower Kross	1	47
12	15	Pepsi	6	45
13	16	Ołówek	9	76
14	17	Lipton	6	96

Przesunięcie w indeksach wyniku z tego iż ręcznie usunąłem kilka złych danych, dodanych przed poprawą kodu.

2.2 Zadanie 2

Odwróć relacje zgodnie z poniższym schematem



2.2.1 Klasa Supplier

namespace AdrianŻerebiecEFProducts

```
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public ICollection<Product> Products { get; set; } = new List<Product>();
        public override string ToString()
        {
            return CompanyName;
        }
    }
}
```

2.2.2 Klasa Product

namespace AdrianŻerebiecEFProducts

```
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitOnStock { get; set; }
        public override string ToString()
        {
            return $"{ProductName} ({UnitOnStock} szt.)";
        }
    }
}
```

2.2.3 Klasa ProductContext

Bez zmian w porównaniu do Zadania

2.2.4 Klasa Program

```
namespace AdrianZerebiecEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();
            Product product = createNewProduct();
            Supplier supplier = null;

            bool createdSupplier = false;
            bool isValidChoice = false;
            while (!isValidChoice)
            {
                Console.WriteLine("Czy chcesz dodać nowego dostawcę?");
                string choice = Console.ReadLine();
                if (choice == "tak")
                {
                    isValidChoice = true;
                    supplier = createNewSupplier();
                    createdSupplier = true;
                    break;
                }
                else if (choice == "nie")
                {
                    isValidChoice = true;
                    Console.WriteLine("Lista wszystkich dostawców");
                    foreach (Supplier suppliers in context.Suppliers)
                    {
                        Console.WriteLine($"{suppliers.SupplierID}
{suppliers}");
                    }
                    supplier = chooseSupplier(context);
                    break;
                }
                else
                {
                    Console.WriteLine("Niepoprawna odpowiedź");
                    Console.WriteLine("Spróbuj ponownie");
                }
            }
            if (supplier != null)
            {
                supplier.Products.Add(product);
                Console.WriteLine("Dodano produkt do dostawcy");
                if (createdSupplier)
                {
                    context.Suppliers.Add(supplier);
                }
                context.Products.Add(product);
                context.SaveChanges();
            }
            else
            {
                Console.WriteLine("Źle podane dane");
            }
        }

        private static Product createNewProduct()
        {
            Console.WriteLine("Podaj nazwę produktu: ");
            string productName = Console.ReadLine();
        }
    }
}
```

```

        Console.WriteLine("Podaj ile sztuk produktu jest dostępnych: ");
        int units = Int32.Parse(Console.ReadLine());
        Console.WriteLine("Trwa tworzenie produktu");
        Product product = new Product
        {
            ProductName = productName,
            UnitOnStock = units
        };
        Console.WriteLine($"Stworzono produkt: {product}");
        return product;
    }

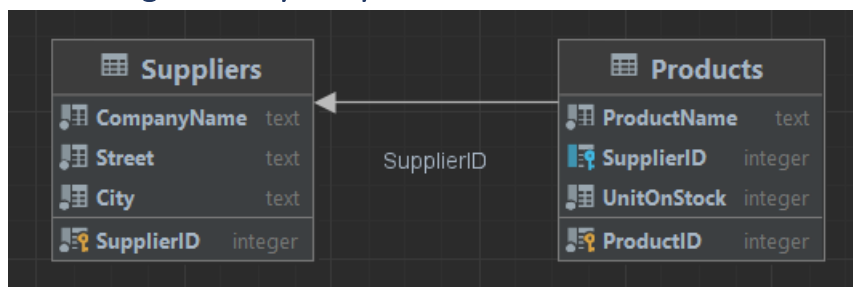
    private static Supplier createNewSupplier()
    {
        Console.Write("Podaj nazwę dostawcy:");
        string companyName = Console.ReadLine();
        Console.Write("Podaj miasto:");
        string city = Console.ReadLine();
        Console.Write("Podaj ulicę:");
        string street = Console.ReadLine();
        Console.WriteLine("Trwa tworzenie dostawcy");
        Supplier supplier = new Supplier
        {
            CompanyName = companyName,
            City = city,
            Street = street
        };
        Console.WriteLine($"Stworzono dostawcę: {supplier}");
        return supplier;
    }

    private static Supplier chooseSupplier(ProductContext productContext)
    {
        Console.Write("Podaj ID wybranego dostawcy: ");
        if (int.TryParse(Console.ReadLine(), out int choice))
        {
            return productContext.Suppliers.FirstOrDefault(s => s.SupplierID ==
choice);
        }

        Console.WriteLine("Niepoprawny wybór");
        return null;
    }
}

```

2.2.5 Diagram bazy danych



Jak widać mimo odwrócenia relacji w bazie danych relacja wygląda tak samo.

Najprawdopodobniej Entity Framework dokonał optymalizacji, a co za tym idzie nie musimy w tabeli Suppliers trzymać powielonych dostawców różniących się SupplierID oraz kluczem obcym. Byłoby to problematyczne w rozróżnianiu dostawców.

2.2.6 Przykład działania

```
Podaj nazwę produktu:
Kawa
Podaj ile sztuk produktu jest dostępnych:
712
Trwa tworzenie produktu
Stworzono produkt: Kawa (712 szt.)
Czy chcesz dodać nowego dostawcę?
nie
Lista wszystkich dostawców
[1] Rowerex
[2] Rowerex
[3] Rowers
[4] Castrol
[5] Kredex
[6] Napojex
[7] Media RTV
[8] Hurtownia kredex
[9] Ołówkowe Imperium
[10] Hurtowania Produktów
Podaj ID wybranego dostawcy: 10
Dodano produkt do dostawcy
```

```
Podaj nazwę produktu:
Woda
Podaj ile sztuk produktu jest dostępnych:
1000
Trwa tworzenie produktu
Stworzono produkt: Woda (1000 szt.)
Czy chcesz dodać nowego dostawcę?
tak
Podaj nazwę dostawcy: Kropla Wody
Podaj miasto: Żywiec
Podaj ulicę: Morska
Trwa tworzenie dostawcy
Stworzono dostawcę: Kropla Wody
Dodano produkt do dostawcy
```

2.2.7 Tabele

Tabela Suppliers

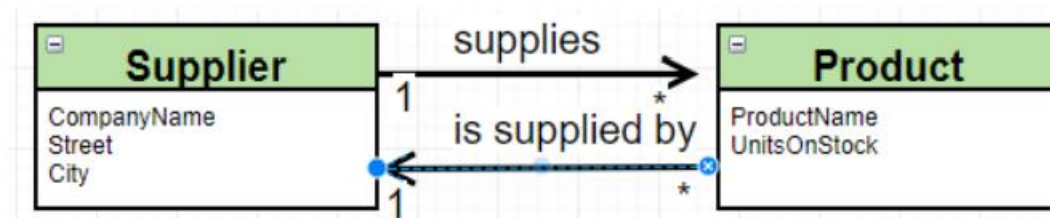
10		10	Hurtowania Produktów	Zagłoby	Wrocław
11		11	Kropla Wody	Morska	Żywiec

Tabele Products

15		18	Jajka	10	45
16		19	Kawa	10	712
17		20	Mąka	10	56
18		21	Musztarda	10	11
19		22	Woda	11	1000

2.3 Zadanie 3

Zamodeluj relację dwustronną jak poniżej:



2.3.1 Klasa Supplier

```
namespace AdrianZerebiecEFProducts
```

```
{
```

```
    public class Supplier
```

```
    {
```

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

```
        public string CompanyName { get; set; }
```

```
        public string Street { get; set; }
```

```
        public string City { get; set; }
```

```
        public ICollection<Product> Products { get; set; } = new List<Product>();
```

```
        public override string ToString()
```

```
        {
```

```
            return CompanyName;
```

```
        }
```

```
    }
```

```
}
```

2.3.2 Klasa Product

```
namespace AdrianZerebiecEFProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitOnStock { get; set; }
        public Supplier? Supplier { get; set; } = null;
        public override string ToString()
        {
            return $"{ProductName} ({UnitOnStock} szt.)";
        }
    }
}
```

2.3.3 Klasa ProductContext

```
namespace AdrianZerebiecEFProducts
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductDatabase");
        }
    }
}
```

2.3.4 Klasa Program

```
namespace AdrianZerebiecEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();
            Product product = createNewProduct();
            Supplier supplier = null;

            bool createdSupplier = false;
            bool isValidChoice = false;
            while (!isValidChoice)
            {
                Console.WriteLine("Czy chcesz dodać nowego dostawcę?");
                string choice = Console.ReadLine();
                if (choice == "tak")
                {
                    isValidChoice = true;
                    supplier = createNewSupplier();
                    createdSupplier = true;
                    break;
                }
                else if (choice == "nie")
            }
        }
    }
}
```

```

        {
            isValidChoice = true;
            Console.WriteLine("Lista wszystkich dostawców");
            foreach (Supplier suppliers in context.Suppliers)
            {
                Console.WriteLine($"[{suppliers.SupplierID}]
{suppliers}");
            }
            supplier = chooseSupplier(context);
            break;
        }
        else
        {
            Console.WriteLine("Niepoprawna odpowiedź");
            Console.WriteLine("Spróbuj ponownie");
        }
    }
    if (supplier != null)
    {
        product.Supplier = supplier;
        supplier.Products.Add(product);
        Console.WriteLine("Dodano produkt do dostawcy");
        if (createdSupplier)
        {
            context.Suppliers.Add(supplier);
        }
        context.Products.Add(product);
        context.SaveChanges();
    }
    else
    {
        Console.WriteLine("Źle podane dane");
    }
}

private static Product createNewProduct()
{
    Console.WriteLine("Podaj nazwę produktu: ");
    string productName = Console.ReadLine();
    Console.WriteLine("Podaj ile sztuk produktu jest dostępnych: ");
    int units = Int32.Parse(Console.ReadLine());
    Console.WriteLine("Trwa tworzenie produktu");
    Product product = new Product
    {
        ProductName = productName,
        UnitOnStock = units
    };
    Console.WriteLine($"Stworzono produkt: {product}");
    return product;
}

private static Supplier createNewSupplier()
{
    Console.WriteLine("Podaj nazwę dostawcy:");
    string companyName = Console.ReadLine();
    Console.WriteLine("Podaj miasto:");
    string city = Console.ReadLine();
    Console.WriteLine("Podaj ulicę:");
    string street = Console.ReadLine();
    Console.WriteLine("Trwa tworzenie dostawcy");
    Supplier supplier = new Supplier
    {
        CompanyName = companyName,

```



```

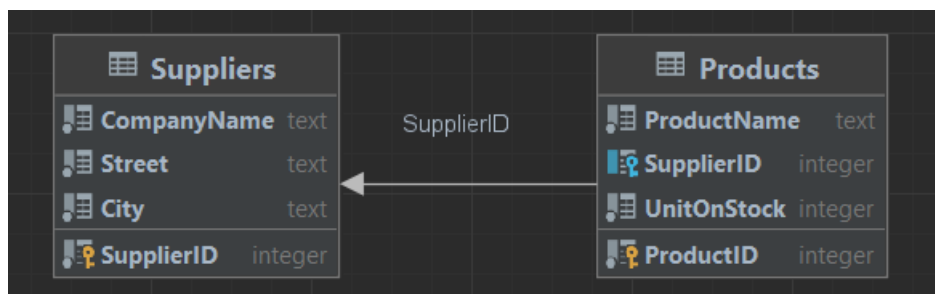
        City = city,
        Street = street
    };
    Console.WriteLine($"Stworzono dostawcę: {supplier}");
    return supplier;
}

private static Supplier chooseSupplier(ProductContext productContext)
{
    Console.Write("Podaj ID wybranego dostawcy: ");
    if (int.TryParse(Console.ReadLine(), out int choice))
    {
        return productContext.Suppliers.FirstOrDefault(s => s.SupplierID ==
choice);
    }

    Console.WriteLine("Niepoprawny wybór");
    return null;
}
}
}

```

2.3.5 Diagram bazy danych



Entity Framework pozwala nam na zapisanie relacji dwustronnej. Jednak zapisane przez nas relacje są przekształcane do takich, które da zapisać się w bazie, stąd kolejny raz obserwujemy analogiczny diagram.

2.3.6 Przykład działania

```

Podaj nazwę produktu:
Makaron
Podaj ile sztuk produktu jest dostępnych:
45
Trwa tworzenie produktu
Stworzono produkt: Makaron (45 szt.)
Czy chcesz dodać nowego dostawcę?
tak
Podaj nazwę dostawcy: MakaronX
Podaj miasto: Białystok
Podaj ulicę: Urzędnicza
Trwa tworzenie dostawcy
Stworzono dostawcę: MakaronX
Dodano produkt do dostawcy

```

```

Podaj nazwę produktu:
Świderki
Podaj ile sztuk produktu jest dostępnych:
11
Trwa tworzenie produktu
Stworzono produkt: Świderki (11 szt.)
Czy chcesz dodać nowego dostawcę?
nie
Lista wszystkich dostawców
[1] Rowerex
[2] Rowerex
[3] Rowers
[4] Castrol
[5] Kredex
[6] Napojex
[7] Media RTV
[8] Hurtownia kredex
[9] Ołówkowe Imperium
[10] Hurtowania Produktów
[11] Kropla Wody
[12] MakaronX
Podaj ID wybranego dostawcy: 12
Dodano produkt do dostawcy

```

2.3.7 Tabele

Tabela Supplier

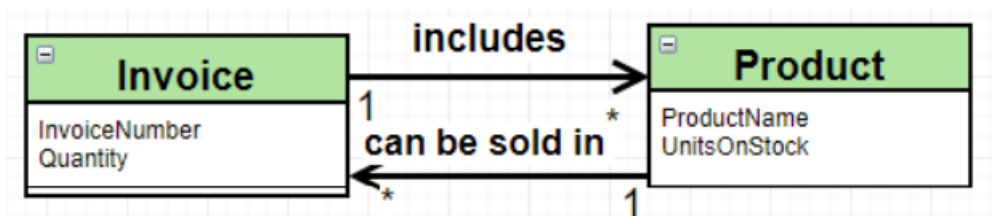
12	12	MakaronX	Urzędnicza	Białystok
13	13	Tanie Ubrania	Rzeczna	Gdańsk

Tabela Product

20	23	Makaron	12	45
21	24	Świderki	12	11
22	25	Koszulka	13	44

2.4 Zadanie 4

Zamodeluj relacje wiele-do-wielu, jak poniżej:



2.4.1 Klasa Product

namespace AdrianZerebiecEFProducts

```
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitOnStock { get; set; }
        public virtual ICollection<InvoiceHelper> InvoiceHelpers { get; set; }
        public override string ToString()
        {
            return $"{ProductName} (Pozostało {UnitOnStock} szt.)";
        }
    }
}
```

2.4.2 Klasa Invoice

namespace AdrianZerebiecEFProducts

```
{
    public class Invoice
    {
        [Key]public int InvoiceID { get; set; }
        public virtual ICollection<InvoiceHelper> InvoiceHelpers { get; set; }
        public override string ToString()
        {
            StringBuilder sb = new($"Faktura o ID {InvoiceID}:");
            foreach (InvoiceHelper item in InvoiceHelpers)
            {

```

```

        sb.Append($"{item}");
    }
    return sb.ToString();
}
}
}

```

2.4.3 Klasa InvoiceHelper

```

namespace AdrianZerebiecEFProducts
{
    public class InvoiceHelper
    {
        [Key, Column(Order = 0)]
        public int InvoiceID { get; set; }

        [Key, Column(Order = 1)]
        public int ProductID { get; set; }
        public virtual Invoice Invoice { get; set; }
        public virtual Product Product { get; set; }
        public int units { get; set; }
        public override string ToString()
        {
            return $"{Product} (Kupiono {units} szt.)";
        }
    }
}

```

2.4.4 Klasa ProductContext

```

namespace AdrianZerebiecEFProducts
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Invoice> Invoices { get; set; }
        public DbSet<InvoiceHelper> InvoiceHelpers { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductDatabase");
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceHelper>()
                .HasKey(x => new { x.InvoiceID, x.ProductID });
        }
    }
}

```

2.4.5 Klasa Program

```
namespace AdrianZerebiecEFProducts
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            using ProductContext context = new ProductContext();
            bool exit = false;
            Console.WriteLine("Co chciałbyś zrobić?");
            Console.WriteLine("Wpisz jedną z komend: dodaj, usuń, sprzedaj, pokaż  
wszystko, pokaż dostępne, zamknij");
            Console.WriteLine("dodaj - dodawanie produktu");
            Console.WriteLine("usuń - usuwanie produktu");
            Console.WriteLine("sprzedaj - sprzedawanie produktów");
            Console.WriteLine("pokaż wszystko - wyświetla wszystkie produkty");
            Console.WriteLine("pokaż dostępne - wyświetla dostępne produkty");
            Console.WriteLine("pokaż fakture - pokazuje fakture o wybranym ID");
            Console.WriteLine("zamknij - kończy działanie");

            while (!exit)
            {
                string command = Console.ReadLine();
                if (command == "dodaj")
                {
                    addProduct(context);
                }
                else if (command == "usuń")
                {
                    removeProduct(context);
                }
                else if (command == "sprzedaj")
                {
                    sellProduct(context);
                }
                else if (command == "pokaż wszystko")
                {
                    showAllProduct(context);
                }
                else if (command == "pokaż dostępne")
                {
                    showAvailableProduct(context);
                }
                else if (command == "pokaż fakture")
                {
                    Console.WriteLine("Podaj ID faktury którą chcesz zobaczyć");
                    var invoiceIds = context.Invoices.Select(i => i.InvoiceID);
                    Console.WriteLine("Dostępne Id faktur:");
                    foreach (var ids in invoiceIds)
                    {
                        Console.WriteLine($"{ids}");
                    }
                    int id = Int32.Parse(Console.ReadLine());
                    showInvoice(id);
                }
                else if (command == "zamknij")
                {
                    exit = true;
                    Console.WriteLine("Dziękuję za skorzystanie");
                    break;
                }
                else
            }
        }
    }
}
```

```

        {
            Console.WriteLine("Nieznane polecenie");
        }
    }

    public static void addProduct(ProductContext context)
    {
        Console.WriteLine("Podaj nazwę produktu: ");
        string name = Console.ReadLine();

        int units;
        while (true)
        {
            Console.WriteLine("Podaj liczbę dostępnych produktów (liczba musi być większa od 0): ");
            if (int.TryParse(Console.ReadLine(), out units) && units > 0)
            {
                break;
            }
            Console.WriteLine("Nieprawidłowa liczba");
        }

        Console.WriteLine("Tworzenie produktu");
        Product product = new Product()
        {
            ProductName = name,
            UnitOnStock = units,
        };
        Console.WriteLine($"Stworzono nowy produkt: {product}");
        Console.WriteLine("Zapisywanie produktu");
        context.Products.Add(product);
        context.SaveChanges();
    }

    public static void showAllProduct(ProductContext context)
    {
        Console.WriteLine("Lista wszystkich produktów:");
        var query = context.Products.ToList();

        if (!query.Any())
        {
            Console.WriteLine("Brak produktów w bazie");
        }
        else
        {
            foreach (var product in query)
            {
                Console.WriteLine($"[{product.ProductID}] {product.ProductName} (Pozostało: {product.UnitOnStock})");
            }
        }
    }

    public static void removeProduct(ProductContext context)
    {
        showAllProduct(context);
        Console.WriteLine("Podaj ID produktu do usunięcia: ");
        int id;
        while (true)
        {
            if (int.TryParse(Console.ReadLine(), out id))

```

```

        {
            var query = context.Products.FirstOrDefault(p => p.ProductID
== id);
            if (query != null)
            {
                context.Remove(query);
                context.SaveChanges();
                Console.WriteLine($"Produkt o ID {id} usunięty
pomyślnie");
                break;
            }
        }
        Console.WriteLine("Nieprawidłowe ID produktu");
    }
}

public static void showAvailableProduct(ProductContext context)
{
    Console.WriteLine("Lista dostępnych produktów");
    var query = from product in context.Products
                where product.UnitOnStock > 0
                select product;

    if (query.Count() == 0)
    {
        Console.WriteLine("Brak produktów w bazie");
    }
    foreach (var product in query)
    {
        Console.WriteLine($"[{product.ProductID}]
{product.ProductName} {Pozostało: {product.UnitOnStock}}");
    }
}

public static void showInvoice(int invoiceId)
{
    using (var context = new ProductContext())
    {
        var invoice = context.Invoices.Include(x => x.InvoiceHelpers)
            .ThenInclude(x => x.Product)
            .SingleOrDefault(x => x.InvoiceID == invoiceId);

        if (invoice == null)
        {
            Console.WriteLine($"Faktura z ID {invoiceId} nie istnieje");
            return;
        }

        Console.WriteLine(invoice);
    }
}

public static List<InvoiceHelper> chooseInvoiceItem(ProductContext
context)
{
    Console.WriteLine("Wybierz produkty, które mają zostać dodane do
faktury:");
    Console.WriteLine("Wprowadź ID produktu, po spacji liczbę
sprzedawanych sztuk, a następnie wciśnij Enter.");
    Console.WriteLine("Aby zakończyć dodawanie produktów, wpisz
'koniec'");

    showAvailableProduct(context);

    Dictionary<Product, int> addedItems = new();

```

```

while (true)
{
    string input = Console.ReadLine();

    if (input == "koniec")
    {
        Console.WriteLine("Zakończono dodawanie produktów do
faktury.");
        break;
    }

    string[] splitted = input.Split();

    if (splitted.Length != 2 || !int.TryParse(splitted[0], out int
productId) || !int.TryParse(splitted[1], out int units))
    {
        Console.WriteLine("Nieprawidłowe dane. Wprowadź ID produktu,
po spacji liczbę sprzedawanych sztuk.");
        continue;
    }

    Product product = context.Products.FirstOrDefault(p =>
p.ProductID == productId);

    if (product == null)
    {
        Console.WriteLine("Nie znaleziono produktu o podanym ID.");
        continue;
    }

    int newUnits = units + (addedItems.ContainsKey(product) ?
addedItems[product] : 0);

    if (newUnits > product.UnitOnStock)
    {
        Console.WriteLine($"Dostępnych jest tylko
{product.UnitOnStock} szt.");
        continue;
    }

    Console.WriteLine($"Dodano {units} szt. {product.ProductName} do
faktury. Razem na fakturze jest {newUnits} szt.");

    if (addedItems.ContainsKey(product))
    {
        addedItems[product] = newUnits;
    }
    else
    {
        addedItems.Add(product, newUnits);
    }
}

List<InvoiceHelper> invoiceHelpers = new();

foreach (KeyValuePair<Product, int> item in addedItems)
{
    Console.WriteLine($"{item.Key.ProductName}, {item.Value} szt.");
    invoiceHelpers.Add(new InvoiceHelper
    {
        Product = item.Key,
        units = item.Value
    });
}

```

```

        });
    }

    return invoiceHelpers;
}

public static void sellProduct(ProductContext context)
{
    Console.WriteLine("Wybierz produkty, które mają zostać dodane do faktury:");
    Console.WriteLine("Wprowadź ID produktu, po spacji liczbę sprzedawanych sztuk, a następnie wciśnij Enter.");
    Console.WriteLine("Aby zakończyć dodawanie produktów, wpisz 'koniec'");

    showAvailableProduct(context);

    Dictionary<Product, int> addedItems = new();

    while (true)
    {
        string input = Console.ReadLine();

        if (input == "koniec")
        {
            Console.WriteLine("Zakończono dodawanie produktów do faktury.");
            break;
        }

        string[] splitted = input.Split();

        if (splitted.Length != 2 || !int.TryParse(splitted[0], out int productId) || !int.TryParse(splitted[1], out int units))
        {
            Console.WriteLine("Nieprawidłowe dane. Wprowadź ID produktu, po spacji liczbę sprzedawanych sztuk.");
            continue;
        }

        Product product = context.Products.FirstOrDefault(p => p.ProductID == productId);

        if (product == null)
        {
            Console.WriteLine("Nie znaleziono produktu o podanym ID.");
            continue;
        }

        int newUnits = units + (addedItems.ContainsKey(product) ? addedItems[product] : 0);

        if (newUnits > product.UnitOnStock)
        {
            Console.WriteLine($"Dostępnych jest tylko {product.UnitOnStock} szt.");
            continue;
        }

        Console.WriteLine($"Dodano {units} szt. {product.ProductName} do faktury. Razem na fakturze jest {newUnits} szt.");

        if (addedItems.ContainsKey(product))

```



```

        {
            addedItems[product] = newUnits;
        }
        else
        {
            addedItems.Add(product, newUnits);
        }
    }

    List<InvoiceHelper> invoiceHelpers = new();

    foreach (KeyValuePair<Product, int> item in addedItems)
    {
        Console.WriteLine($"{item.Key.ProductName}, {item.Value} szt.");
        invoiceHelpers.Add(new InvoiceHelper
        {
            Product = item.Key,
            units = item.Value
        });
    }

    if (invoiceHelpers.Count == 0)
    {
        Console.WriteLine("Nie dokonano zakupów, nie można utworzyć faktury.");
        return;
    }

    Console.WriteLine("Aktualizacja bazy");

    foreach (InvoiceHelper inv in invoiceHelpers)
    {
        inv.Product.UnitOnStock -= inv.units;
        Console.WriteLine($"Pozostało {inv.Product.UnitOnStock} szt. {inv.Product.ProductName}");
    }

    Invoice invoice = new Invoice
    {
        InvoiceHelpers = invoiceHelpers
    };

    context.Invoices.Add(invoice);
    context.SaveChanges();
    Console.WriteLine($"Utworzono nową fakturę o ID: {invoice.InvoiceID}");
}

public static void showInvoicesForProduct(ProductContext context, int ID)
{
    var invoices = context.Invoices
        .Where(i => i.InvoiceHelpers.Any(h => h.Product.ProductID == ID))
        .ToList();

    if (invoices.Count == 0)
    {
        Console.WriteLine($"Nie znaleziono faktur dla produktu o ID: {ID}.");
        return;
    }

    Console.WriteLine($"Faktury dla produktu o ID: {ID}:");
    foreach (var invoice in invoices)
    {

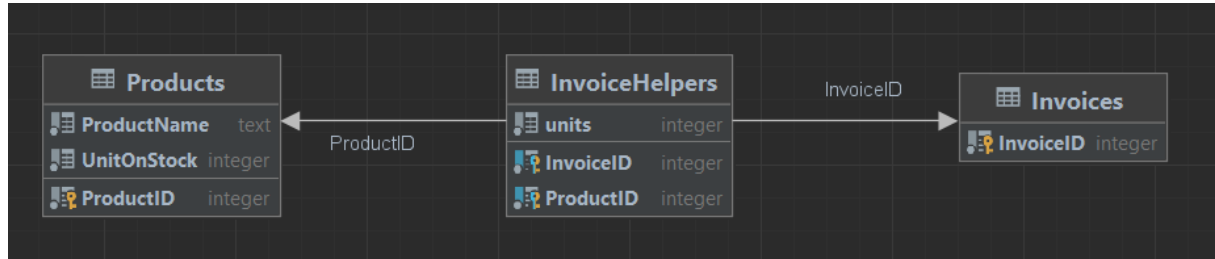
```

```

        Console.WriteLine($"ID faktury: {invoice.InvoiceID}");
    }
}
}

```

2.4.6 Diagram bazy danych



2.4.7 Przykład działania

```

Co chciałbyś zrobić?
Wpisz jedną z komend: dodaj, usuń, sprzedaj, pokaż wszystko, pokaż dostępne, zamknij
dodaj - dodawanie produktu
usuń - usuwanie produktu
sprzedaj - sprzedawanie produktów
pokaż wszystko - wyświetla wszystkie produkty
pokaż dostępne - wyświetla dostępne produkty
pokaż fakture - pokazuje fakturę o wybranym ID
zamknij - kończy działanie
pokaż fakturę
Podaj ID faktury którą chcesz zobaczyć
Dostępne Id faktur:
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
12
Faktura o ID 12:
Marcherwka (Pozostało 60 szt.) (Kupiono 10 szt.)
pokaż wszystko
Lista wszystkich produktów:
[1] Rower górski (Pozostało: 2)
[2] Koła do roweru (Pozostało: 10)
[3] Coca-Cola (Pozostało: 0)
[4] Marcherwka (Pozostało: 60)
pokaż dostępne
Lista dostępnych produktów
[1] Rower górski(Pozostało: 2)
[2] Koła do roweru(Pozostało: 10)
[4] Marcherwka(Pozostało: 60)
zamknij
Dziękuję za skorzystanie

```

```

Co chciałbyś zrobić?
Wpisz jedną z komend: dodaj, usuń, sprzedaj, pokaż wszystko, pokaż dostępne, zamknij
dodaj - dodawanie produktu
usuń - usuwanie produktu
sprzedaj - sprzedawanie produktów
pokaż wszystko - wyświetla wszystkie produkty
pokaż dostępne - wyświetla dostępne produkty
pokaż faktury - pokazuje faktury o wybranym ID
zamknij - kończy działanie
usuń
Lista wszystkich produktów:
[1] Rower górski (Pozostało: 2)
[2] Koła do roweru (Pozostało: 10)
[3] Coca-Cola (Pozostało: 0)
[4] Marchewka (Pozostało: 60)
[5] Kapusta (Pozostało: 62)
Podaj ID produktu do usunięcia:
3
Produkt o ID 3 usunięty pomyślnie
pokaż wszystko
Lista wszystkich produktów:
[1] Rower górski (Pozostało: 2)
[2] Koła do roweru (Pozostało: 10)
[4] Marchewka (Pozostało: 60)
[5] Kapusta (Pozostało: 62)
pokaż dostępne
Lista dostępnych produktów
[1] Rower górski(Pozostało: 2)
[2] Koła do roweru(Pozostało: 10)
[4] Marchewka(Pozostało: 60)
[5] Kapusta(Pozostało: 62)
zamknij
Dziękuję za skorzystanie

```

```

Co chciałbyś zrobić?
Wpisz jedną z komend: dodaj, usuń, sprzedaj, pokaż wszystko, pokaż dostępne, zamknij
dodaj - dodawanie produktu
usuń - usuwanie produktu
sprzedaj - sprzedawanie produktów
pokaż wszystko - wyświetla wszystkie produkty
pokaż dostępne - wyświetla dostępne produkty
pokaż faktury - pokazuje faktury o wybranym ID
zamknij - kończy działanie
dodaj
Podaj nazwę produktu:
Kapusta
Podaj liczbę dostępnych produktów (liczba musi być większa od 0):
111
Tworzenie produktu
Stworzono nowy produkt: Kapusta (Pozostało 111 szt.)
Zapisywanie produktu
pokaż wszystko
Lista wszystkich produktów:
[1] Rower górski (Pozostało: 2)
[2] Koła do roweru (Pozostało: 10)
[3] Coca-Cola (Pozostało: 0)
[4] Marchewka (Pozostało: 60)
[5] Kapusta (Pozostało: 111)
sprzedaj
Wybierz produkty, które mają zostać dodane do faktury:
Wprowadź ID produktu, po spacji liczbę sprzedawanych sztuk, a następnie wciśnij Enter.
Aby zakończyć dodawanie produktów, wpisz 'koniec'
Lista dostępnych produktów
[1] Rower górski(Pozostało: 2)
[2] Koła do roweru(Pozostało: 10)
[4] Marchewka(Pozostało: 60)
[5] Kapusta(Pozostało: 111)
5 49
Dodano 49 szt. Kapusta do faktury. Razem na fakturze jest 49 szt.

Nieprawidłowe dane. Wprowadź ID produktu, po spacji liczbę sprzedawanych sztuk.
koniec
Zakończono dodawanie produktów do faktury.
Kapusta, 49 szt.
Aktualizacja bazy
Pozostało 62 szt. Kapusta
Utworzono nową fakturę o ID: 13
pokaż faktury
Podaj ID faktury którą chcesz zobaczyć
Dostępne Id faktur:
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
13
Faktura o ID 13:
Kapusta (Pozostało 62 szt.) (Kupiono 49 szt.)
zamknij
Dziękuję za skorzystanie

```

```

Co chciałbyś zrobić?
Wpisz jedną z komend: dodaj, usuń, sprzedaj, pokaż wszystko, pokaż dostępne, zamknij
dodaj - dodawanie produktu
usuń - usuwanie produktu
sprzedaj - sprzedawanie produktów
pokaż wszystko - wyświetla wszystkie produkty
pokaż dostępne - wyświetla dostępne produkty
pokaż faktury - pokazuje faktury o wybranym ID
faktura dla produktu - pokazuje faktury, na których jest dany produkt
zamknij - kończy działanie
faktury dla produktu
Nieznane polecenie
faktura dla produktu
Lista wszystkich produktów:
[1] Rower górski (Pozostało: 2)
[2] Koła do roweru (Pozostało: 10)
[4] Marcherwka (Pozostało: 60)
[5] Kapusta (Pozostało: 62)
Podaj id produktu:
1
Faktury dla produktu o ID: 1:
ID faktury: 1
ID faktury: 2
ID faktury: 3
ID faktury: 4
ID faktury: 9
pokaż faktury
Podaj ID faktury którą chcesz zobaczyć
Dostępne Id faktur:
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
9
Faktura o ID 9:
Rower górski (Pozostało 2 szt.) (Kupiono 1 szt.)
zamknij
Dziękuję za skorzystanie

```

2.4.8 Tabele

Tabela Products




	 ProductID	 ProductName	 UnitOnStock
1	1	Rower górski	2
2	2	Koła do roweru	10
3	4	Marcherwka	60
4	5	Kapusta	62

Tabela Invoices





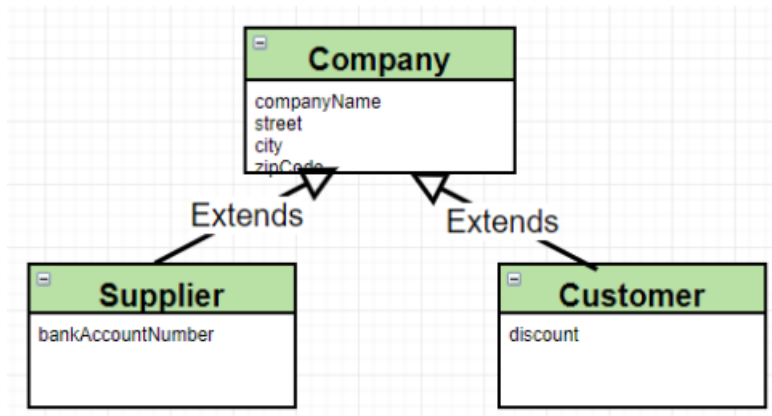
	 InvoiceID ↕
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

Tabela InvoiceHelpers

	 InvoiceID ↕	 ProductID ↕	 units ↕
1	1	1	2
2	2	1	1
3	3	1	2
4	3	2	2
5	4	1	2
6	4	2	4
7	7	2	5
8	8	2	1
9	9	1	1
10	10	4	20
11	11	4	10
12	12	4	10
13	13	5	49

2.5 Zadanie 5

Dziedziczenie:



2.5.1 Klasa Customer

```
namespace AdrianZerebiecEFProducts
{
    public class Customer : Company
    {
        public int CustomerID { get; set; }
        public int Discount { get; set; }
        public override string ToString()
        {
            return $"{base.ToString()} (klient)";
        }
    }
}
```

2.5.2 Klasa Supplier

```
namespace AdrianZerebiecEFProducts
{
    public class Supplier: Company
    {
        public int SupplierID { get; set; }
        public string BankAccountNumber { get; set; } = string.Empty;
        public override string ToString()
        {
            return $"{base.ToString()} (dostawca)";
        }
    }
}
```

2.5.3 Klasa Company

```
namespace AdrianZerebiecEFProducts
{
    public abstract class Company
    {
        public int CompanyID { get; set; }
        public string CompanyName { get; set; } = string.Empty;
        public string Street { get; set; } = string.Empty;
        public string City { get; set; } = string.Empty;
        public string ZipCode { get; set; } = string.Empty;

        public override string ToString()
        {
            return $"[{CompanyID}] {CompanyName}";
        }
    }
}
```

2.5.4 Klasa CompanyType

```
namespace AdrianZerebiecEFProducts
{
    public static class CompanyType
    {
        public const string CUSTOMER = "klient";
        public const string SUPPLIER = "dostawca";
        public static List<string> ALL_TYPES = new()
        {
            CUSTOMER,
            SUPPLIER
        };
    }
}
```

2.5.5 Klasa CompanyContext

```
namespace AdrianZerebiecEFProducts
{
    public class CompanyContext : DbContext
    {
        public DbSet<Company>? Companies { get; set; }
        public DbSet<Supplier>? Suppliers { get; set; }
        public DbSet<Customer>? Customers { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductDatabase");
        }
    }
}
```

2.5.6 Klasa Program

```
class Program
{
    static void Main(string[] args)
    {
        using CompanyContext context = new CompanyContext();
        Console.WriteLine("Co chcesz zrobić?");
        Console.WriteLine("Lista komend:");
        Console.WriteLine("dodaj - umożliwia dodanie firmy");
        Console.WriteLine("pokaż - pokazuje wybrane firmy");
        Console.WriteLine("koniec - kończy działanie");

        bool exit = false;
        while (!exit)
        {
            Console.WriteLine("Podaj komendę: ");
            string command = Console.ReadLine();
            if (command == "dodaj")
            {
                AddCompany(context);
            }
            else if (command == "pokaż")
            {
                DisplayCompanies(context);
            }
            else if (command == "koniec")
            {
                exit = true;
                break;
            }
            else
            {
                Console.WriteLine("Nieznana komenda");
            }
        }
    }

    private static void AddCompany(CompanyContext context)
    {
        while (true)
        {
            string type = "";
            List<string> choices = CompanyType.ALL_TYPES;
            Console.WriteLine("Możliwe wybory:");
            foreach (string choice in choices)
            Console.WriteLine($"{choice}");
            Console.WriteLine("Wpisz 'koniec' aby zakończyć");
            bool exit = false;
            while (!exit)
            {
                string? choice = Console.ReadLine()?.Trim();
                if (choice == null) break;
                foreach (string possible in choices)
                {
                    if (choice.Equals(possible))
                    {
                        type = choice;
                        exit = true;
                    }
                }
            }
        }
    }
}
```



```

    }
    if (exit) break;
    else if (choice.Equals("koniec"))
    {
        Console.WriteLine("Dziękuję za skorzystanie");
        type = String.Empty;
        exit = true;
    }
    else
    {
        Console.WriteLine("Błędna komenda");
    }
}

if (type == "") return;
Console.WriteLine("Podaj nazwę firmy: ");
string companyName = Console.ReadLine().Trim();
Console.WriteLine("Podaj ulicę: ");
string street = Console.ReadLine().Trim();
Console.WriteLine("Podaj miasto: ");
string city = Console.ReadLine().Trim();
Console.WriteLine("Podaj kod pocztowy: ");
string postalCode = Console.ReadLine().Trim();

if (type == CompanyType.CUSTOMER)
{
    Console.WriteLine("Podaj wartość zniżki w procentach:");
    int discount = Int32.Parse(Console.ReadLine().Trim());
    var customer = new Customer
    {
        CompanyName = companyName,
        Street = street,
        City = city,
        ZipCode = postalCode,
        Discount = discount
    };
    context.Companies.Add(customer);
}
else if (type == CompanyType.SUPPLIER)
{
    Console.WriteLine("Podaj numer konta bankowego: ");
    string bankAccount = Console.ReadLine().Trim();
    var supplier = new Supplier
    {
        CompanyName = companyName,
        Street = street,
        City = city,
        ZipCode = postalCode,
        BankAccountNumber = bankAccount
    };
    context.Companies.Add(supplier);
}
context.SaveChanges();
}

private static void DisplayCompanies(CompanyContext context)
{
    List<string> types = new();
    types.AddRange(CompanyType.ALL_TYPES);
    types.Add("wszytsko");
    Console.WriteLine("Wprowadź typ firm, które chcesz wypisać: ");

```

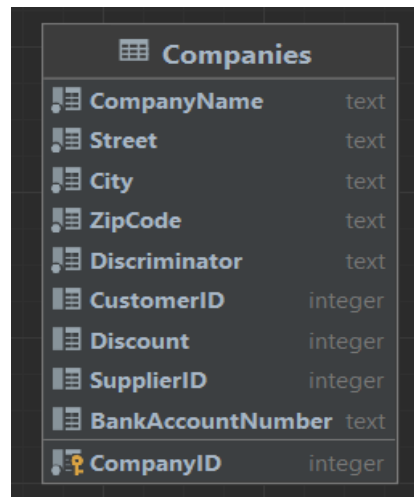
```

string type = "";
List<string> choices = types; ;
Console.WriteLine("Możliwe wybory:");
foreach (string choice in choices) Console.WriteLine($"{choice}");
Console.WriteLine("Wpisz 'koniec' aby zakończyć");
bool exit = false;
while (!exit)
{
    string? choice = Console.ReadLine()?.Trim();
    if (choice == null) type = "";
    foreach (string possible in choices)
    {
        if (choice.Equals(possible))
        {
            type = choice;
            exit = true;
        }
    }
    if (exit) break;
    else if (choice.Equals("koniec"))
    {
        Console.WriteLine("Dziękuję za skorzystanie");
        type = String.Empty;
        exit = true;
    }
    else
    {
        Console.WriteLine("Błędna komenda");
    }
}

if (type == "")
{
    Console.WriteLine("Nie wprowadzono typu firmy.");
    return;
}
if (type == "wszytsko")
{
    Console.WriteLine("Lista wszystkich firm: ");
    foreach (Company company in context.Companies)
    {
        Console.WriteLine(company);
    }
}
else if (type == CompanyType.SUPPLIER)
{
    Console.WriteLine("Lista wszystkich dostawców: ");
    foreach (Supplier supplier in context.Suppliers)
    {
        Console.WriteLine(supplier);
    }
}
else if (type == CompanyType.CUSTOMER)
{
    Console.WriteLine("Lista wszystkich klientów: ");
    foreach (Customer customer in context.Customers)
    {
        Console.WriteLine(customer);
    }
}
}
}
}
}

```

2.5.7 Diagram bazy danych



2.5.8 Przykład działania

```
Co chcesz zrobić?
Lista komend:
dodaj - umożliwia dodanie firmy
pokaż - pokazuje wybrane firmy
koniec - kończy działanie
Podaj komendę:
dodaj
Możliwe wybory:
klient
dostawca
Wpisz 'koniec' aby zakończyć
klient
Podaj nazwę firmy:
Pudzianex
Podaj ulicę:
Morska
Podaj miasto:
Gdańsk
Podaj kod pocztowy:
82-291
Podaj wartość zniżki w procentach:
10
Możliwe wybory:
klient
dostawca
Wpisz 'koniec' aby zakończyć
dostawca
Podaj nazwę firmy:
Unixer
Podaj ulicę:
Typowa
Podaj miasto:
Kraków
Podaj kod pocztowy:
98-912
Podaj numer konta bankowego:
797123731290832
Możliwe wybory:
klient
dostawca
Wpisz 'koniec' aby zakończyć
koniec
Dziękuję za skorzystanie
Podaj komendę:
koniec
```

```
Co chcesz zrobić?
Lista komend:
dodaj - umożliwia dodanie firmy
pokaż - pokazuje wybrane firmy
koniec - kończy działanie
Podaj komendę:
pokaż
Wprowadź typ firm, które chcesz wypisać:
Możliwe wybory:
klient
dostawca
wszystko
Wpisz 'koniec' aby zakończyć
wszystko
Lista wszystkich firm:
[6] Napojex (klient)
[7] Rowerex (dostawca)
[8] Drutex (klient)
[9] Dostartex (dostawca)
[10] Hurtownia warzyw (klient)
[11] Hurtownia ołówków (dostawca)
[12] Pudzianex (klient)
[13] Unixer (dostawca)
Podaj komendę:
pokaż
Wprowadź typ firm, które chcesz wypisać:
Możliwe wybory:
klient
dostawca
wszystko
Wpisz 'koniec' aby zakończyć
klient
Lista wszystkich klientów:
[6] Napojex (klient)
[8] Drutex (klient)
[10] Hurtownia warzyw (klient)
[12] Pudzianex (klient)
Podaj komendę:
koniec
```

2.5.9 Tabele

	CompanyID	CompanyName	Street	City	ZipCode	Discriminator	CustomerID	Discount	SupplierID	BankAccountNumber
1	6	Napojex	Szwedzka	Warszawa	22-122	Customer	0	5	<null>	<null>
2	7	Rowerex	Lubelska	Kraków	22-982	Supplier	<null>	<null>	0	45648126516816841
3	8	Drutex	Kryształowa	Bydgoszcz	82-982	Customer	0	2	<null>	<null>
4	9	Dostartex	Dostawcza	Łódź	87-092	Supplier	<null>	<null>	0	798213793092316899081
5	10	Hurtownia warzyw	Chełmicza	Chełm	98-082	Customer	0	10	<null>	<null>
6	11	Hurtownia ołówków	Pomorska	Toruń	87-212	Supplier	<null>	<null>	0	78123632189241094217
7	12	Pudzianex	Morska	Gdańsk	82-291	Customer	0	10	<null>	<null>
8	13	Unixer	Typowa	Kraków	98-912	Supplier	<null>	<null>	0	797123731290832

2.6 Zadanie 6

Zamodeluj tę samą hierarchię dziedziczenia, ale tym razem użyj strategii Table-PerType

2.6.1 Klasa Customer

```
namespace AdrianZerebiecEFProducts
{
    [Table("Customers")]
    public class Customer : Company
    {
        public int CompanyID { get; set; }
        public int Discount { get; set; }
        public override string ToString()
        {
            return $"{base.ToString()} (klient)";
        }
    }
}
```

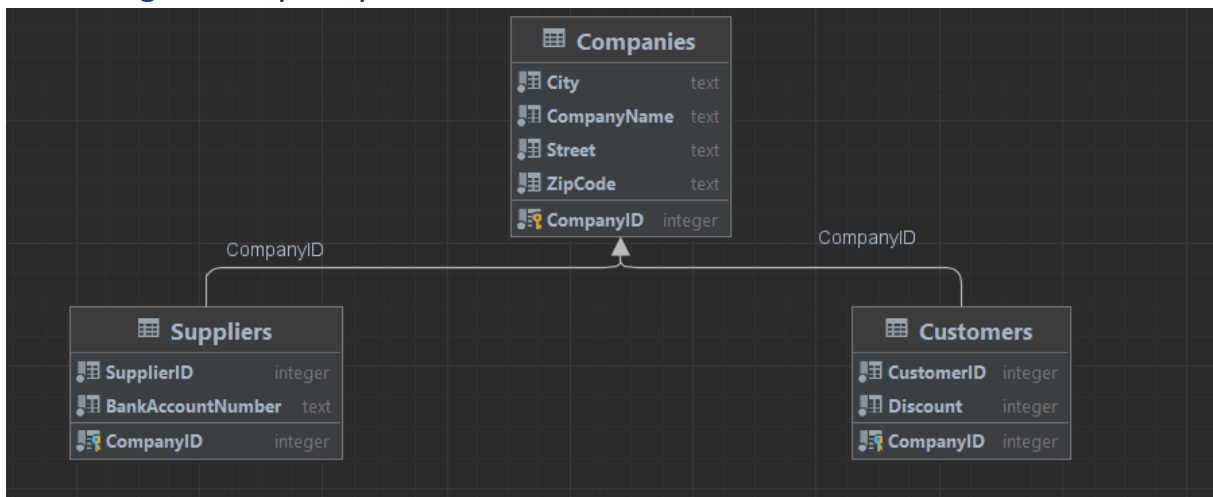
2.6.2 Klasa Supplier

```
namespace AdrianZerebiecEFProducts
{
    [Table("Suppliers")]
    public class Supplier : Company
    {
        public int CompanyID { get; set; }
        public string BankAccountNumber { get; set; } = string.Empty;
        public override string ToString()
        {
            return $"{base.ToString()} (dostawca)";
        }
    }
}
```

2.6.3 Pozostałe klasy

Pozostałe klasy nie zostały zmienione

2.6.4 Diagram bazy danych



2.6.5 Przykład działania

Działanie analogiczne jak w punkcie 2.5.8

2.6.6 Tabele

Suppliers

	CompanyID	BankAccountNumber
1	2	874398014893013

Customers

	CompanyID	Discount
1	1	10

Companies

	CompanyID	City	CompanyName	Street	ZipCode
1	1	Łódź	Drutex	Typowa	09-921
2	2	Kraków	Loki-Koki	Rzeczna	09-212

2.6.7 Porównanie dziedziczenia Table-Per-Hierarchy i Table-Per-Type

2.6.7.1 Table-Per-Hierarchy

Tworzona jest jedna tabela, zawierająca dane klas dziedziczących oraz dane charakterystyczne dla każdej z klas dziedziczących. Jeśli klasa posiada atrybut, którego nie ma w klasie to dostaje tam wartość null oraz odpowiednie wartości parametrów. Dzięki takiemu podejściu nie musimy wykonywać dużo operacji join, gdyż wszystko jest w jednej tabeli. Problem w tym, że przy dużej ilości null'i tracimy na przejrzystości i na miejscu.

2.6.7.2 Table-Per-Type

Tworzone jest kilka tabel, osobne dla każdej z klas. Jednak są one połączone ze sobą relacją typu 1 do 1. Nie musimy dzięki temu trzymać pustych komórek z null'em oraz schemat jest bardziej czytelny. Problem w tym, że trzeba będzie wykonywać zdecydowanie więcej operacji typu join.

2.7 Końcowe pliki

