

# Praca samodzielna

## 2.2 Wprowadzenie pojęcia dostawcy

```
using System;
namespace JuliaSmerdelEFProducts
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
    }
}
```

```
using System;
using Microsoft.EntityFrameworkCore;
namespace JuliaSmerdelEFProducts
{
    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=ProductsDatabase");
        }
    }
}
```

```
using System;
namespace JuliaSmerdelEFProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
        public Supplier Supplier { get; set; }
    }
}
```

```

namespace JuliaSmerdelEFPProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            bool goodDecision = false;
            ProductContext productContext = new ProductContext();

            Console.WriteLine("Tworzenie nowego produktu:");
            Product product = createNewProduct();

            do
            {
                Console.WriteLine("Czy stworzyć nowego dostawcę i przypisać go do tego produktu? (tak/nie)");
                string decision = Console.ReadLine();

                switch (decision)
                {
                    case "tak":
                        goodDecision = true;
                        Console.WriteLine("Tworzenie nowego dostawcy:");
                        Supplier supplier = createNewSupplier();
                        productContext.Suppliers.Add(supplier);
                        product.Supplier = supplier;
                        break;

                    case "nie":
                        goodDecision = true;
                        displaySuppliers(productContext);
                        supplier = findSupplier(productContext);
                        product.Supplier = supplier;
                        break;
                }
            } while (!goodDecision);

            productContext.Products.Add(product);
            Console.WriteLine("Dane zostały zaktualizowane");
            productContext.SaveChanges();
        }
    }
}

```

Ln 41,

```

private static Supplier findSupplier(ProductContext productContext)
{
    Console.WriteLine("Podaj id dostawcy, który ma być przypisany do nowego produktu.");
    int id = Int32.Parse(Console.ReadLine());

    var query = from supp in productContext.Suppliers where supp.SupplierID == id select supp;

    return query.FirstOrDefault();
}

private static void displaySuppliers(ProductContext productContext)
{
    Console.WriteLine("Oto wszyscy dostawcy:");
    foreach(Supplier supp in productContext.Suppliers)
    {
        Console.WriteLine($"{ supp.SupplierID} { supp.CompanyName}");
    }
}

```

```
private static Supplier createNewSupplier()
{
    Console.WriteLine("Podaj nazwę dostawcy");
    string suppName = Console.ReadLine();

    Console.WriteLine("Podaj miasto dostawcy");
    string suppCity = Console.ReadLine();

    Console.WriteLine("Podaj ulicę dostawcy");
    string suppStreet = Console.ReadLine();

    Supplier supplier = new Supplier
    {
        CompanyName = suppName,
        Street = suppStreet,
        City = suppCity
    };

    return supplier;
}

private static Product createNewProduct()
{
    Console.WriteLine("Podaj nazwę produktu");
    string prodName = Console.ReadLine();

    Console.WriteLine("Podaj liczbę sztuk produktu");
    int quantity = Int32.Parse(Console.ReadLine());

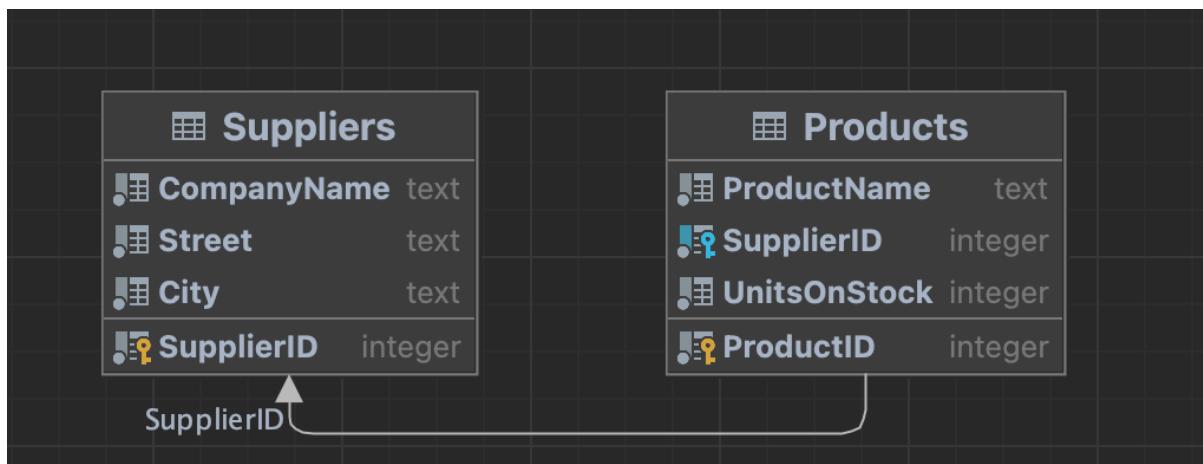
    Product product = new Product
    {
        ProductName = prodName,
        UnitsOnStock = quantity
    };

    return product;
}
```

Przykłady wywołania:

```
Tworzenie nowego produktu:  
Podaj nazwę produktu  
Jabłko  
Podaj liczbę sztuk produktu  
12  
Czy stworzyć nowego dostawcę i przypisać go do tego produktu? (tak/nie)  
tak  
Tworzenie nowego dostawcy:  
Podaj nazwę dostawcy  
Apple  
Podaj miasto dostawcy  
Szczecin  
Podaj ulicę dostawcy  
Nowa  
Dane zostały zaktualizowane
```

```
Tworzenie nowego produktu:  
Podaj nazwę produktu  
Laptop  
Podaj liczbę sztuk produktu  
2  
Czy stworzyć nowego dostawcę i przypisać go do tego produktu? (tak/nie)  
nie  
Oto wszyscy dostawcy:  
1 Apple  
2 PanŁódek  
3 Kaufland  
Podaj id dostawcy, który ma być przypisany do nowego produktu.  
1  
Dane zostały zaktualizowane
```



	ProductID	ProductName	SupplierID	UnitsOnStock
1	1	Jabłko	1	12
2	2	Łódka	2	3
3	3	Zeszyt	3	48
4	4	Laptop	1	2

	SupplierID	CompanyName	Street	City
1	1	Apple	Nowa	Szczecin
2	2	PanŁódek	Wodna	Łódź
3	3	Kaufland	Polska	Warszawa

## 2.3 Odwrócenie relacji

Kod klasy „Program” został zmieniony względem punktu 2.2 wyłącznie w klasie „Main” w dwóch miejscach.

```
static void Main(string[] args)
{
    bool goodDecision = false;
    ProductContext productContext = new ProductContext();

    Console.WriteLine("Tworzenie nowego produktu:");
    Product product = createNewProduct();

    do
    {
        Console.WriteLine("Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy? (tak/nie)");
        string decision = Console.ReadLine();

        switch (decision)
        {
            case "tak":
                goodDecision = true;
                Console.WriteLine("Tworzenie nowego dostawcy:");
                Supplier supplier = createNewSupplier();
                productContext.Suppliers.Add(supplier);
                Console.WriteLine("Przypisanie podanego produktu do podanego dostawcy");
                supplier.Products.Add(product);
                break;

            case "nie":
                goodDecision = true;
                displaySuppliers(productContext);
                supplier = findSupplier(productContext);
                Console.WriteLine("Przypisanie podanego produktu do podanego dostawcy");
                supplier.Products.Add(product);
                break;
        }
    } while (!goodDecision);

    productContext.Products.Add(product);
    productContext.SaveChanges();
}
```

```
using System;
namespace JuliaSmerdelEFProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
    }
}
```

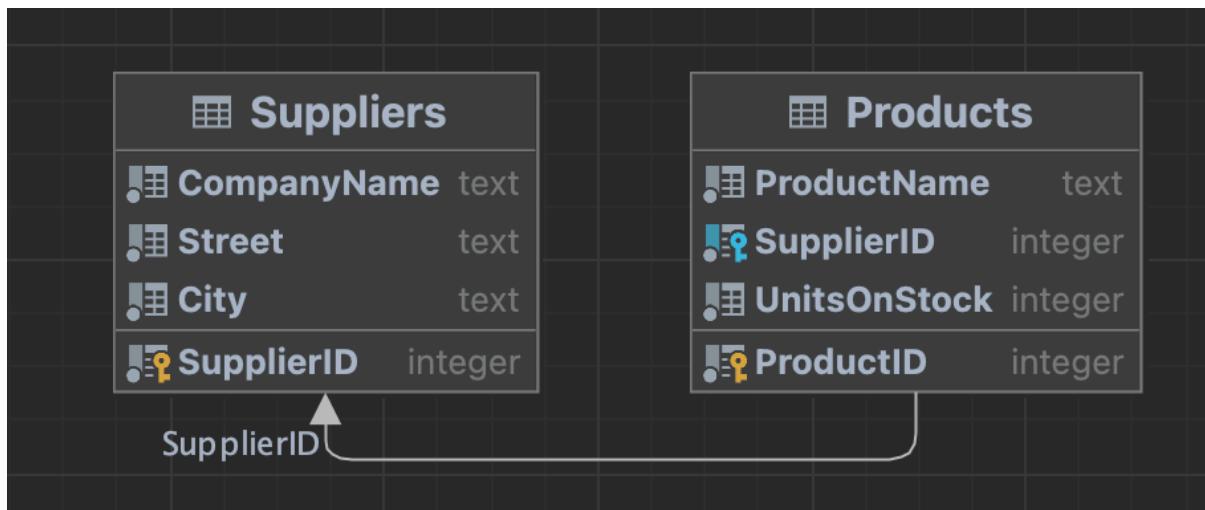
```
using System;
namespace JuliaSmerdelEFPProducts
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public ICollection<Product> Products { get; set; } = new List<Product>();
    }
}
```

Przykłady wywołania:

```
Tworzenie nowego produktu:
Podaj nazwę produktu
Telefon
Podaj liczbę sztuk produktu
3
Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy? (tak/nie)
tak
Tworzenie nowego dostawcy:
Podaj nazwę dostawcy
Hjułałej
Podaj miasto dostawcy
Hrubieszów
Podaj ulicę dostawcy
Herbaciana
Przypisanie podanego produktu do podanego dostawcy

Tworzenie nowego produktu:
Podaj nazwę produktu
Poduszka
Podaj liczbę sztuk produktu
43
Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy? (tak/nie)
nie
Oto wszyscy dostawcy:
1 Hjułałej
2 Kiszką
3 Azorki
Podaj id dostawcy, który ma być przypisany do nowego produktu.
2
Przypisanie podanego produktu do podanego dostawcy
```

Pomimo odwrócenia relacji, DataGrip relacja wciąż jest ustawiona w taki sam sposób jak w punkcie 2.2



	ProductID	ProductName	SupplierID	UnitsOnStock
1	1	Telefon	1	3
2	2	Myszka	2	12
3	3	Pistacje	3	874
4	4	Poduszka	2	43

	SupplierID	CompanyName	Street	City
1	1	Hjułałej	Herbaciana	Hrubieszów
2	2	Kiszka	Koronna	Kielce
3	3	Azorki	Reksiowa	Burki

## 2.4 Relacja dwustronna

W klasie „Program” zmiana nastąpiła wyłącznie w metodzie „Main”.

```
static void Main(string[] args)
{
    bool goodDecision = false;
    ProductContext productContext = new ProductContext();

    Console.WriteLine("Tworzenie nowego produktu:");
    Product product = createNewProduct();

    do
    {
        Console.WriteLine("Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy" +
            " oraz dostawcę do produktu? (tak/nie)");
        string decision = Console.ReadLine();

        switch (decision)
        {
            case "tak":
                goodDecision = true;
                Console.WriteLine("Tworzenie nowego dostawcy:");
                Supplier supplier = createNewSupplier();
                productContext.Suppliers.Add(supplier);
                Console.WriteLine("Przypisanie podanego produktu do podanego dostawcy oraz dostawcy do produktu");
                supplier.Products.Add(product);
                product.Supplier = supplier;
                break;

            case "nie":
                goodDecision = true;
                displaySuppliers(productContext);
                supplier = findSupplier(productContext);
                Console.WriteLine("Przypisanie podanego produktu do podanego dostawcy oraz dostawcy do produktu");
                supplier.Products.Add(product);
                product.Supplier = supplier;
                break;
        }
    } while (!goodDecision);

    productContext.Products.Add(product);
    productContext.SaveChanges();
```

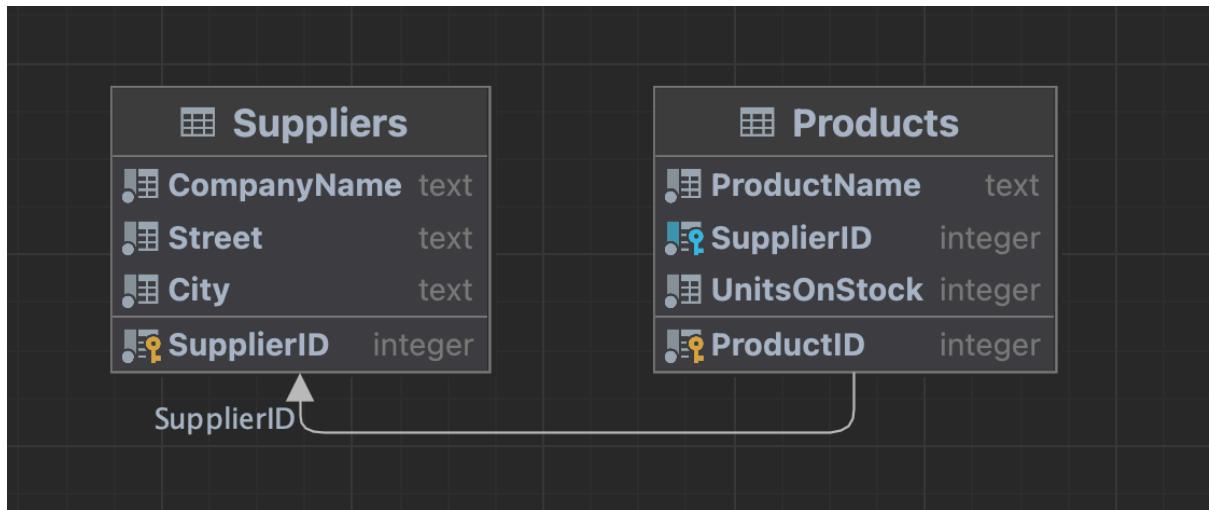
```
using System;
namespace JuliaSmerdelEFPProducts
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public ICollection<Product> Products { get; set; } = new List<Product>();
    }
}
```

```
using System;
namespace JuliaSmerdelEFPProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
        public Supplier Supplier { get; set; }
    }
}
```

## Przykładowe wywołania:

```
Tworzenie nowego produktu:
Podaj nazwę produktu
Baton
Podaj liczbę sztuk produktu
27
Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy oraz dostawcę do produktu? (tak/nie)
tak
Tworzenie nowego dostawcy:
Podaj nazwę dostawcy
Notab
Podaj miasto dostawcy
Bydgoszcz
Podaj ulicę dostawcy
Mars
Przypisanie podanego produktu do podanego dostawcy oraz dostawcy do produktu
```

```
Tworzenie nowego produktu:
Podaj nazwę produktu
Płetwy
Podaj liczbę sztuk produktu
22
Czy stworzyć nowego dostawcę i przypisać produkt do tego dostawcy oraz dostawcę do produktu? (tak/nie)
nie
Oto wszyscy dostawcy:
1 Notab
2 Oceania
3 AgataMeble
Podaj id dostawcy, który ma być przypisany do nowego produktu.
2
Przypisanie podanego produktu do podanego dostawcy oraz dostawcy do produktu
```



Ponownie diagram w DataGripie wygląda identycznie do tego w punkcie 2.2, mimo zmiany relacji w EntityFramework.

	ProductID	ProductName	SupplierID	UnitsOnStock
1	1	Baton	1	27
2	2	Maska	2	3
3	3	Krzesło	3	42
4	4	Płetwy	2	22

	SupplierID	CompanyName	Street	City
1	1	Notab	Mars	Bydgoszcz
2	2	Oceania	Morska	Łódź
3	3	AgataMeble	Meblowa	Poznań

## 2.5 Relacja wiele-do-wielu

Dla ułatwienia operowania po bazie danych w celu zdobycia informacji o danej fakturze, stworzona została pomocnicza klasa InvoiceInfo, która zawiera wszystkie niezbędne informacje na temat danej faktury.

```
using System;
using Microsoft.EntityFrameworkCore;
namespace JuliaSmerdelEFProducts
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Invoice> Invoices { get; set; }
        public DbSet<InvoiceInfo> InvoiceInfos { get; set; }

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

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceInfo>().HasKey(t => new { t.InvoiceNumber, t.ProductID });
        }
    }
}
```

```
using System;
namespace JuliaSmerdelEFProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
        public virtual ICollection<InvoiceInfo> InvoiceInfos { get; set; }
    }
}
```

```
using System;
using System.ComponentModel.DataAnnotations;

namespace JuliaSmerdelEFProducts
{
    public class Invoice
    {
        [Key]
        public int InvoiceNumber { get; set; }
        public virtual ICollection<InvoiceInfo> InvoiceInfos { get; set; }
    }
}
```

```

using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace JuliaSmerdelEFProducts
{
    public class InvoiceInfo
    {
        [Key, Column(Order = 0)]
        public int InvoiceNumber { 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 Quantity { get; set; }
    }
}

```

```

namespace JuliaSmerdelEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            bool stillStanding = true;
            ProductContext productContext = new ProductContext();

            while(stillStanding)
            {
                Console.WriteLine("add - dodanie nowego produktu, buy - kupowanie produktów, invoice - przeglądanie faktury, exit - wyjście ");
                string decision = Console.ReadLine();

                switch (decision)
                {
                    case "add":
                        Console.WriteLine("Tworzenie nowego produktu:");
                        addProduct(productContext);
                        break;

                    case "buy":
                        displayProducts(productContext);
                        invoiceFun(productContext);
                        break;

                    case "invoice":
                        displayInvoice(productContext);
                        break;

                    case "exit":
                        stillStanding = false;
                        break;
                    default:
                        Console.WriteLine("Podana została zła komenda");
                        break;
                }
            }
        }
    }
}

```

```

private static void displayInvoice(HandlerContext productContext)
{
    Console.WriteLine("Podaj ID faktury, którą chcesz zobaczyć.");
    int id = Int32.Parse(Console.ReadLine());
    Invoice invoice = productContext.Invoices.First(inv => inv.InvoiceNumber == id);

    foreach(InvoiceInfo stats in invoice.InvoiceInfos)
    {
        Console.WriteLine($"ID:{stats.ProductID} Liczba kupionych produktów:{stats.Quantity}");
    }
}

private static void invoiceFun(HandlerContext productContext)
{
    List<InvoiceInfo> invoiceInfo = new();
    Console.WriteLine("Aby zakończyć kupowanie produktów do faktury, kliknij Enter.");
    while (true)
    {
        Console.WriteLine("Podaj ID produktu, który chcesz kupić.");
        string input = Console.ReadLine();
        if (input == String.Empty)
        {
            Console.WriteLine("Zakończono kupowanie produktów");
            break;
        }
        int id = Int32.Parse(input);

        Console.WriteLine("Podaj, ile produktu chcesz kupić.");
        string input2 = Console.ReadLine();

        if (input2 == String.Empty)
        {
            Console.WriteLine("Zakończono kupowanie produktów");
            break;
        }
        int quantity = Int32.Parse(input2);
    }
}

```

```

    Product product = productContext.Products.First(prod => prod.ProductID == id);
    if (quantity > product.UnitsOnStock)
    {
        Console.WriteLine("Podana liczba jest większa niż liczba dostępnych sztuk.");
    }
    else
    {
        product.UnitsOnStock = product.UnitsOnStock - quantity;

        invoiceInfo.Add(new InvoiceInfo { ProductID = id, Quantity = quantity });
    }
}

Invoice invoice = new Invoice
{
    InvoiceInfos = invoiceInfo
};
productContext.Add(invoice);
productContext.SaveChanges();

}

```

```

private static void addProduct(ProductsContext productContext)
{
    Console.WriteLine("Podaj nazwę produktu");
    string prodName = Console.ReadLine();

    Console.WriteLine("Podaj liczbę sztuk produktu");
    int quantity = Int32.Parse(Console.ReadLine());

    Product product = new Product
    {
        ProductName = prodName,
        UnitsOnStock = quantity
    };

    productContext.Products.Add(product);
    productContext.SaveChanges();
}

private static void displayProducts(ProductsContext productContext)
{
    Console.WriteLine("Oto wszystkie produkty:");
    foreach (Product prod in productContext.Products)
    {
        Console.WriteLine($"ID:{prod.ProductID} Nazwa:{prod.ProductName} Dostępnych:{prod.UnitsOnStock} ");
    }
}

```

## Przykłady wywołania:

```

add - dodanie nowego produktu, buy - kupowanie produktów, invoice - przeglądanie faktury, exit - wyjście
add
Tworzenie nowego produktu:
Podaj nazwę produktu
Lampa
Podaj liczbę sztuk produktu
2
add - dodanie nowego produktu, buy - kupowanie produktów, invoice - przeglądanie faktury, exit - wyjście
add
Tworzenie nowego produktu:
Podaj nazwę produktu
Pomarańcza
Podaj liczbę sztuk produktu
0

```

```

add - dodanie nowego produktu, buy - kupowanie produktów, invoice - przeglądanie faktury, exit - wyjście
buy
Oto wszystkie produkty:
ID:1 Nazwa:Lampa Dostępnych:2
ID:2 Nazwa:Pomarańcza Dostępnych:0
ID:3 Nazwa:Laptop Dostępnych:7
ID:4 Nazwa:Poduszka Dostępnych:67
Podaj ID produktu, który chcesz kupić.
3
Podaj, ile produktu chcesz kupić.
2

```

```

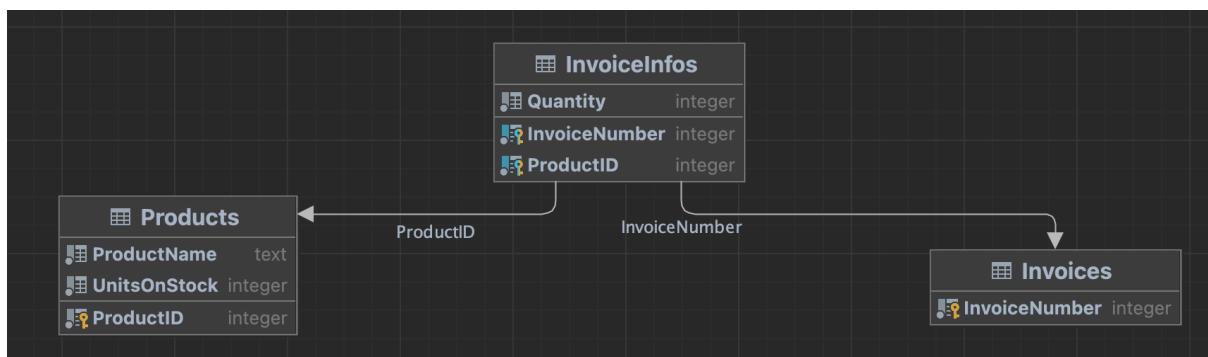
buy
Oto wszystkie produkty:
ID:1 Nazwa:Lampa Dostępnych:2
ID:2 Nazwa:Pomarańcza Dostępnych:0
ID:3 Nazwa:Laptop Dostępnych:5
ID:4 Nazwa:Poduszka Dostępnych:67
Podaj ID produktu, który chcesz kupić.
2
Podaj, ile produktu chcesz kupić.
2
Podana liczba jest większa niż liczba dostępnych sztuk.

```

```

add - dodanie nowego produktu, buy - kupowanie produktów, invoice - przeglądanie faktury, exit - wyjście
invoice
Podaj ID faktury, którą chcesz zobaczyć.
1
ID:3 Liczba kupionych produktów:2
ID:4 Liczba kupionych produktów:13

```



	ProductID	ProductName	UnitsOnStock
1	1	Lampa	2
2	2	Pomarańcza	0
3	3	Laptop	5
4	4	Poduszka	35

	InvoiceNumber
1	1
2	2

	InvoiceNumber	ProductID	Quantity
1	1	3	2
2	1	4	13
3	2	4	19

## 2.6 Dziedziczenie Table-Per-Hierarchy

```
using System;
namespace JuliaSmerdelEFCompanies
{
    public abstract class Company
    {
        public int CompanyID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public string ZipCode { get; set; }
    }
}
```

```
using System;
namespace JuliaSmerdelEFCompanies
{
    public class Customer : Company
    {
        public int CustomerID { get; set; }
        public int Discount { get; set; }
    }
}
```

```
using System;
namespace JuliaSmerdelEFCompanies
{
    public class Supplier : Company
    {
        public int SupplierID { get; set; }
        public string bankAccountNumber { get; set; }
    }
}
```

```
using System;
using Microsoft.EntityFrameworkCore;

namespace JuliaSmerdelEFCompanies
{
    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=CompaniesDatabase");
        }
    }
}
```

```
using JuliaSmerdelEFCompanies;

namespace JuliaSmerdelEFProducts
{
    class Program
    {
        static void Main(string[] args)
        {
            CompanyContext companyContext = new CompanyContext();

            bool stop = false;

            while (!stop)
            {
                Console.WriteLine("add - dodanie danych do bazy, show - pokazanie danych, exit - wyjście");
                string input = Console.ReadLine();
                switch (input)
                {
                    case "add":
                        addFun(companyContext);
                        break;
                    case "show":
                        show(companyContext);
                        break;
                    case "exit":
                        stop = true;
                        break;
                    default:
                        Console.WriteLine("Zła komenda.");
                        break;
                }
            }
        }
    }
}
```

```
private static void addFun(CompanyContext companyContext)
{
    Console.WriteLine("Aby wyjść, kliknij Enter");
    while (true)
    {
        Console.WriteLine("Podaj nazwę firmy");
        string comName = Console.ReadLine();

        if (comName == String.Empty)
        {
            Console.WriteLine("Zakończono dodawanie klienta");
            break;
        }

        Console.WriteLine("Podaj miasto firmy");
        string comCity = Console.ReadLine();

        if (comCity == String.Empty)
        {
            Console.WriteLine("Zakończono dodawanie klienta");
            break;
        }

        Console.WriteLine("Podaj ulicę firmy");
        string comStreet = Console.ReadLine();

        if (comStreet == String.Empty)
        {
            Console.WriteLine("Zakończono dodawanie klienta");
            break;
        }

        Console.WriteLine("Podaj kod pocztowy firmy");
        string comCode = Console.ReadLine();

        if (comCode == String.Empty)
        {
            Console.WriteLine("Zakończono dodawanie klienta");
            break;
        }
    }
}
```

```

Console.WriteLine("supplier - dodanie dostawcy, customer - dodanie klienta");
string input = Console.ReadLine();

if (input == String.Empty)
{
    Console.WriteLine("Zakończono dodawanie klienta");
    break;
}

switch (input)
{
    case "supplier":
        Supplier supp = addSupplier(comName, comCity, comStreet, comCode);
        companyContext.Suppliers.Add(supp);
        companyContext.SaveChanges();
        return;

    case "customer":
        Customer cus = addCustomer(comName, comCity, comStreet, comCode);
        companyContext.Customers.Add(cus);
        companyContext.SaveChanges();
        return;

    default:
        Console.WriteLine("Wrong command");
        return;
}

}

}

```

```

private static Supplier addSupplier(string comName, string comCity, string comStreet, string comCode)
{
    Console.WriteLine("Podaj numer konta bankowego:");
    string bank = Console.ReadLine();

    Supplier supplier = new Supplier
    {
        CompanyName = comName,
        Street = comStreet,
        City = comCity,
        ZipCode = comCode,
        bankAccountNumber = bank
    };

    return supplier;
}

private static Customer addCustomer(string comName, string comCity, string comStreet, string comCode)
{
    Console.WriteLine("Podaj wartość zniżki w %:");
    int discount = int.Parse(Console.ReadLine());

    Customer customer = new Customer
    {
        CompanyName = comName,
        Street = comStreet,
        City = comCity,
        ZipCode = comCode,
        Discount = discount
    };

    return customer;
}

```

```

private static void show(CompanyContext companyContext)
{
    while (true)
    {
        Console.WriteLine("supplier - zobaczenie dostawców, customer - zobaczenie klientów");
        string decision = Console.ReadLine();
        if (decision == String.Empty)
        {
            Console.WriteLine("Zakończono przeglądanie klientów");
            break;
        }

        switch (decision)
        {
            case "supplier":
                showSupp(companyContext);
                return;
            case "customer":
                showCus(companyContext);
                return;
            default:
                Console.WriteLine("Podano złą komendę.");
                return;
        }
    }
}

private static void showSupp(CompanyContext companyContext)
{
    foreach(Supplier supp in companyContext.Suppliers)
    {
        Console.WriteLine($" {supp.CompanyID} {supp.CompanyName} {supp.City} {supp.bankAccountNumber}");
    }
}

private static void showCus(CompanyContext companyContext)
{
    foreach (Customer cus in companyContext.Customers)
    {
        Console.WriteLine($" {cus.CompanyID} {cus.CompanyName} {cus.City} {cus.Discount}");
    }
}

```

Przykłady wywołania:

```

add - dodanie danych do bazy, show - pokazanie danych, exit - wyjście
add
Aby wyjść, kliknij Enter
Podaj nazwę firmy
Kubki
Podaj miasto firmy
Warszawa
Podaj ulicę firmy
Pilna
Podaj kod pocztowy firmy
34-567
supplier - dodanie dostawcy, customer - dodanie klienta
supplier
Podaj numer konta bankowego:
1234678987

```

```

add - dodanie danych do bazy, show - pokazanie danych, exit - wyjście
show
supplier - zobaczenie dostawców, customer - zobaczenie klientów
supplier
1 Frutki Wrocław 123456789
2 Szadur Chełm 88654321
3 Papa Papa 197383687
7 Kubki Warszawa 1234678987

```

```

add - dodanie danych do bazy, show - pokazanie danych, exit - wyjście
show
supplier - zobaczenie dostawców, customer - zobaczenie klientów
customer
4 Tik Tak 10
5 Lala Piła 2
6 Siklawa Walbrzych 10
add - dodanie danych do bazy, show - pokazanie danych, exit - wyjście

```

Companies	
CompanyID	CompanyName
	Street
	City
	ZipCode
	Discriminator
CustomerID	
Discount	
SupplierID	
bankAccountNumber	
CompanyID	

CompanyID	CompanyName	Street	City	ZipCode	Discriminator	CustomerID	Discount	SupplierID	bankAccountNumber
1	Frutki	Mita	Wrocław	12-345	Supplier	<null>	<null>	0	123456789
2	Szadur	Byka	Chełm	22-300	Supplier	<null>	<null>	0	88654321
3	Papa	APA	Papa	aka	Supplier	<null>	<null>	0	197383687
4	Tik	Nowa	Tak	12-345	Customer	0	10	<null>	<null>
5	Lala	Koła	Piła	12-212	Customer	0	2	<null>	<null>
6	Siklawa	Kubiecka	Walbrzych	98-789	Customer	0	10	<null>	<null>
7	Kubki	Pilna	Warszawa	34-567	Supplier	<null>	<null>	0	1234678987

## 2.7 Dziedziczenie Table-Per-Type

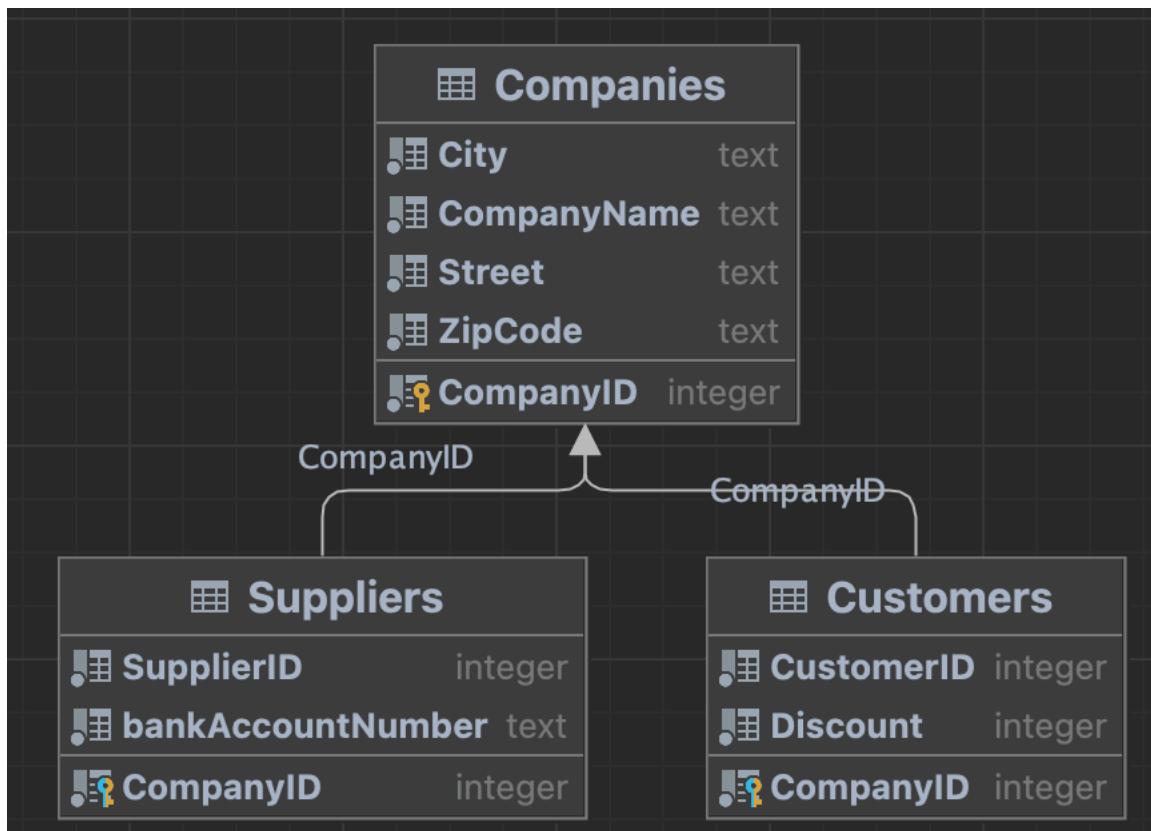
Zmiana nastąpiła tylko w dwóch klasach. Dodawanie oraz wyświetlanie danych wygląda identycznie jak w punkcie 2.6.

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace JuliaSmerdelEFCompanies
{
    [Table("Customers")]
    public class Customer : Company
    {
        public int CustomerID { get; set; }
        public int Discount { get; set; }
    }
}
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace JuliaSmerdelEFCompanies
{
    [Table("Suppliers")]
    public class Supplier : Company
    {
        public int SupplierID { get; set; }
        public string bankAccountNumber { get; set; }
    }
}
```



	CompanyID	City	CompanyName	Street	ZipCode
1	1	Wrocław	Frutki	Miła	12-345
2	2	Łódź	Łosoś	Rybna	33-333
3	3	Warszawa	Star	Polna	12-122
4	4	Bydgoszcz	Danonki	Mleczna	10-007

	CompanyID	CustomerID	Discount
1		3	0 10
2		4	0 2

	CompanyID	SupplierID	bankAccountNumber
1		1	0 123456789
2		2	0 987654321

## **Porównanie dziedziczenia Table-Per-Hierarchy i Table-Per-Type:**

W strategii Table-Per-Hierarchy tworzona jest jedna tabela, która zawiera dane klas dziedziczących. Gdy klasa dziedzicząca posiada atrybut, którego nie ma w klasie nadzędnej, dodawana jest nowa kolumna i dla pozostałych klas wpisywane są tam wartości null. Dzięki takiemu podejściu zmniejsza się liczbę wykonywanych operacji na tabelach. Jednak baza danych jest mniej przejrzysta, gdyż może posiadać wiele wartości null.

W strategii Table-Per-Type tworzonych jest kilka tabel, osobno dla każdej z klas. Tabele klas dziedziczących są łączone z tabelą nadzędzią relacją 1 do 1.

Dzięki temu podejściu baza danych jest bardziej przejrzysta i nie posiada wartości null. Jednak konieczne jest wykonywanie wielu operacji łączenia tabel (join), aby uzyskać potrzebne dane.