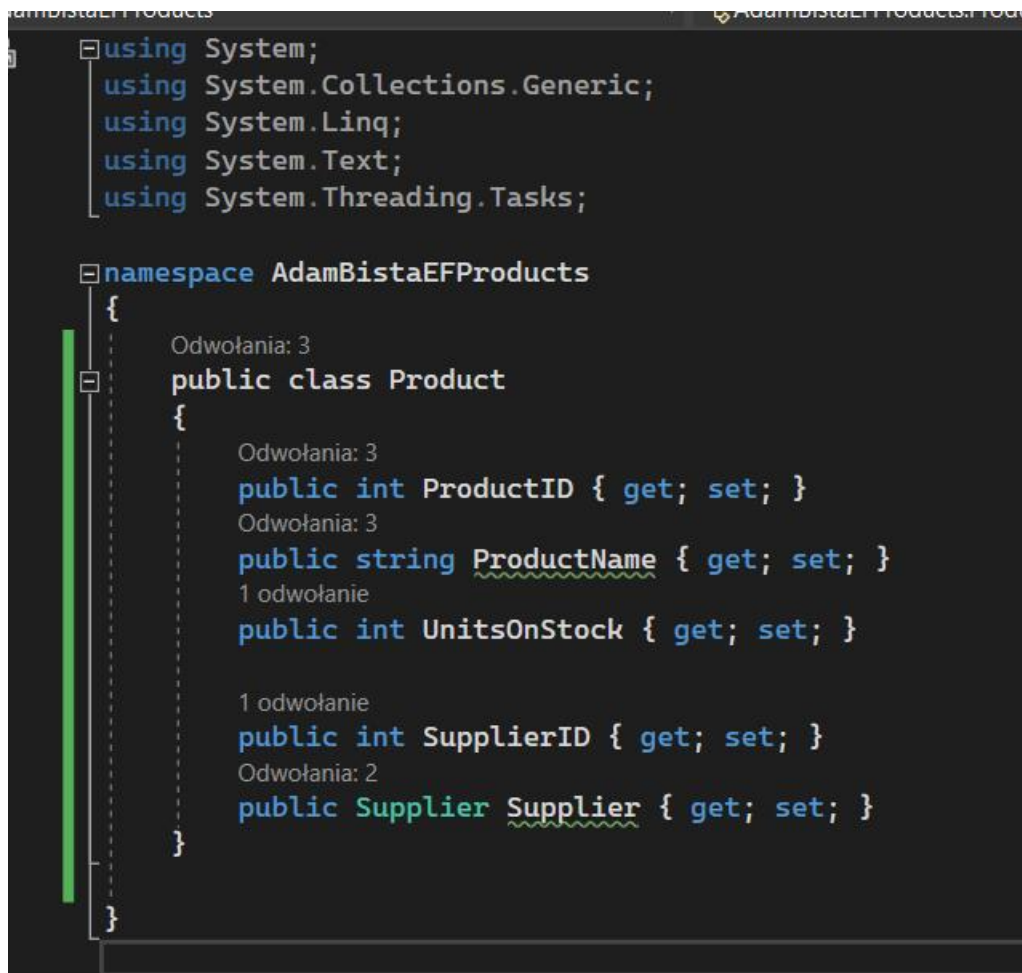


Adam Biśta

I. Struktura tabel

1. Product



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

namespace AdamBistaEFProducts
{
    Odwołania: 3
    public class Product
    {
        Odwołania: 3
        public int ProductID { get; set; }
        Odwołania: 3
        public string ProductName { get; set; }
        1 odwołanie
        public int UnitsOnStock { get; set; }

        1 odwołanie
        public int SupplierID { get; set; }
        Odwołania: 2
        public Supplier Supplier { get; set; }
    }
}
```

2. Supplier:

```
using System.Threading.Tasks;

namespace AdamBistaEFProducts
{
    Odwołania: 3
    public class Supplier
    {
        1 odwołanie
        public int SupplierID { get; set; }
        Odwołania: 3
        public string CompanyName { get; set; }
        Odwołania: 2
        public string Street { get; set; }
        Odwołania: 2
        public string City { get; set; }
    }
}
```

3. ProductContext:

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

namespace AdamBistaEFProducts
{
    Odwołania: 5
    public class ProductContext : DbContext
    {
        Odwołania: 2
        public DbSet<Product> Products { get; set; }
        Odwołania: 2
        public DbSet<Supplier> Suppliers { get; set; }

        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=ProductsDatabase");
        }
    }
}
```

Uwaga! W dalszej części postanowiłem pominąć w screenach namespace, tam się nic nie zmieniało. Skupiłem się na samych klasach i wywołaniach w programie.

Screeny / kody wysyłane z Program.cs będą znajdować się w w obszarze z komentarzem:

```
using Microsoft.EntityFrameworkCore;

namespace AdamBistaEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            //kod
        }
    }
}
```

Migracja:

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabase1
```

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update
```

4. Program:

Wprowadzenie dostawcy i wyświetlenie dostawców:

```
ProductContext productContext = new ProductContext();

// Dodaj nowego dostawcę
var newSupplier = new Supplier
{
    CompanyName = "Company X",
    Street = "123 Main St",
    City = "New York"
};

productContext.Suppliers.Add(newSupplier);
productContext.SaveChanges();

// Wyświetl dostawców
var suppliersQuery = from s in productContext.Suppliers
                    select s;

Console.WriteLine("Dostawcy:");
foreach (var supplier in suppliersQuery)
{
    Console.WriteLine($"ID: {supplier.SupplierID}, " +
        $"Nazwa: {supplier.CompanyName}, " +
        $"Ulica: {supplier.Street}, " +
        $"Miasto: {supplier.City}");
}
```

Z uwagi na fakt, że wyczyściłem bazę danych z produktów, dodam ten sam produkt i przypiszę do dostawcy.

```
//ostatnio wprowadzony produkt
// Dodaj nowy produkt z przypisanym dostawcą
var lastProduct = new Product
{
    ProductName = "Pomidor",
    Supplier = new Supplier
};

productContext.Products.Add(lastProduct);
productContext.SaveChanges();
if (lastProduct != null)
{
    string existingProductName = lastProduct.ProductName;
    Console.WriteLine($"Ostatnio wprowadzony produkt: {existingProductName}");

    // Pobierz informacje o produkcie z dostawcą
    var updatedProduct = productContext.Products
        .Include(p => p.Supplier)
        .FirstOrDefault(p => p.ProductID == lastProduct.ProductID);

    // Wyświetl informacje o produkcie z dostawcą
    Console.WriteLine("\nProdukt z aktualizowanym dostawcą:");
    Console.WriteLine($"ID: {updatedProduct.ProductID}, " +
        $"Nazwa: {updatedProduct.ProductName}, " +
        $"Ilość na stanie: {updatedProduct.UnitsOnStock}, " +
        $"Dostawca: {updatedProduct.Supplier.CompanyName}");
}
else
{
    Console.WriteLine("Brak produktów w bazie danych.");
}
```

konsola:

```
Dostawcy:
ID: 10, Nazwa: Company X, Ulica: 123 Main St, Miasto: New York
Ostatnio wprowadzony produkt: Pomidor

Produkt z aktualizowanym dostawcą:
ID: 6, Nazwa: Pomidor, Ilość na stanie: 0, Dostawca: Company X
```

II. Odwracanie relacji zgodnie ze schematem w zadaniu

Product:

```
Odwołania: 2
public class Product
{
    1 odwołanie
    public int ProductID { get; set; }
    Odwołania: 4
    public string ProductName { get; set; }
    Odwołania: 4
    public int UnitsOnStock { get; set; }
}
}
```

Supplier:

```
public class Supplier
{
    Odwołania: 2
    public int SupplierID { get; set; }
    Odwołania: 2
    public string CompanyName { get; set; }
    1 odwołanie
    public string Street { get; set; }
    1 odwołanie
    public string City { get; set; }
    Odwołania: 6
    public ICollection<Product> Products { get; set; }
}
```

ProductContext nie zmienił się.

Migracja:

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabase2
```

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update
```

Dodamy najpierw sprzedawcę, następnie stworzymy nowe produkty:

```
ProductContext productContext = new ProductContext();
// Stwórz nowego dostawcę
var newSupplier = new Supplier
{
    CompanyName = "Super firma",
    Street = "Sienkiewicza 10a",
    City = "Chrzanow"
};
productContext.Suppliers.Add(newSupplier);
productContext.SaveChanges();

// dodajemy nowe produkty
List<Product> newProducts = new List<Product>
{
    new Product { ProductName = "Pomidor", UnitsOnStock = 5},
    new Product { ProductName = "Orzech", UnitsOnStock = 10},
    new Product { ProductName = "Przenica", UnitsOnStock = 1500}
};
foreach (Product product in newProducts)
{
    productContext.Products.Add(product);
    productContext.SaveChanges();
}
```

Wiążemy produkty ze sprzedawcą:

```
newSupplier.Products = new List<Product>();

foreach (var product in newProducts)
{
    newSupplier.Products.Add(product);
}
productContext.SaveChanges();
```

Test:

```
//testujemy naszą bazę:
// Pobierz dostawcę z bazy danych wraz z przypisanymi produktami
Supplier existingSupplier = productContext.Suppliers
    .Include(s => s.Products)
    .SingleOrDefault(s => s.SupplierID == newSupplier.SupplierID);

// Sprawdź, czy dostawca istnieje
if (existingSupplier != null)
{
    // Sprawdź, czy dostawca ma przypisane produkty
    if (existingSupplier.Products != null && existingSupplier.Products.Any())
    {
        Console.WriteLine($"Produkty dostarczane przez {existingSupplier.CompanyName}:");
        foreach (var product in existingSupplier.Products)
        {
            Console.WriteLine($"ID: {product.ProductID}, " +
                $"Nazwa: {product.ProductName}, " +
                $"Ilość na stanie: {product.UnitsOnStock}");
        }
    }
    else
    {
        Console.WriteLine("Dostawca nie ma przypisanych żadnych produktów.");
    }
}
else
{
    Console.WriteLine("Nie można znaleźć dostawcy.");
}
```

Konsola:

```
Produkty dostarczane przez Super firma:
ID: 1, Nazwa: Pomidor, Ilość na stanie: 5
ID: 2, Nazwa: Orzech, Ilość na stanie: 10
ID: 3, Nazwa: Przenica, Ilość na stanie: 1500
```

III. Modelowanie relacji obustronnej

Supplier pozostanie taki sam z podpunktu II.

ProductContext również nie zmieni się.

Product:

```
Odwołania: 9
public class Product
{
    1 odwołanie
    public int ProductID { get; set; }
    Odwołania: 4
    public string ProductName { get; set; }
    Odwołania: 4
    public int UnitsOnStock { get; set; }

    Odwołania: 0
    public Supplier Supplier { get; set; }
}
```

Migracja:

```
Done.
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabase3
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update
```

Program:

```
ProductContext productContext = new ProductContext();
// a) Stwórz kilka produktów
List<Product> newProducts = new List<Product>
{
    new Product { ProductName = "Pomidor", UnitsOnStock = 5},
    new Product { ProductName = "Orzech", UnitsOnStock = 10},
    new Product { ProductName = "Przenica", UnitsOnStock = 1500}
};
```

```
// b) Dodaj je do produktów dostarczanych przez ostatnio dodanego dostawcę
// Pobierz ostatnio dodanego dostawcę
Supplier lastSupplier = productContext.Suppliers
    .OrderByDescending(s => s.SupplierID)
    .FirstOrDefault();

// Jeśli nie ma dostawcy, stwórz nowego
if (lastSupplier == null)
{
    lastSupplier = new Supplier
    {
        CompanyName = "Firma Super",
        Street = "Cezarego 32",
        City = "Młoszowa"
    };
    productContext.Suppliers.Add(lastSupplier);
    productContext.SaveChanges();
}
```

```
// Dodaj produkty do ostatnio dodanego dostawcy
if (lastSupplier.Products == null)
{
    lastSupplier.Products = new List<Product>();
}

foreach (var product in newProducts)
{
    product.Supplier = lastSupplier;
    productContext.Products.Add(product);
    productContext.SaveChanges();
}
```

Test programu:

```
var productsQuery = from p in productContext.Products
    select p;

// Wyświetl zawartość tabeli Product
Console.WriteLine("Tabela Product:");
foreach (var product in productsQuery)
{
    Console.WriteLine($"ID: {product.ProductID}, " +
        $"Nazwa: {product.ProductName}, " +
        $"Ilość na stanie: {product.UnitsOnStock}, " +
        $"Firma: {product.Supplier.CompanyName}");
}
```



```
// Zapytanie do tabeli Supplier
var suppliersQuery = from s in productContext.Suppliers
                    select s;

// Wyświetl zawartość tabeli Supplier
Console.WriteLine("\nTabela Supplier:");
foreach (var supplier in suppliersQuery)
{
    Console.WriteLine($"ID: {supplier.SupplierID}, " +
        $"Nazwa: {supplier.CompanyName}, " +
        $"Ulica: {supplier.Street}, " +
        $"Miasto: {supplier.City}, " +
        $"Produkty: ");

    // Jeśli dostawca ma przypisane produkty, wyświetl je
    if (supplier.Products != null && supplier.Products.Count > 0)
    {
        foreach (var product in supplier.Products)
        {
            Console.WriteLine($"  - ID: {product.ProductID}, " +
                $"Nazwa: {product.ProductName}, " +
                $"Ilość na stanie: {product.UnitsOnStock}");
        }
    }
    else
    {
        Console.WriteLine("  Brak produktów.");
    }
}
}
```

Konsola:

```
Tabela Product:
ID: 10, Nazwa: Pomidor, Ilość na stanie: 5,Firma: Firma Super
ID: 11, Nazwa: Orzech, Ilość na stanie: 10,Firma: Firma Super
ID: 12, Nazwa: Przenica, Ilość na stanie: 1500,Firma: Firma Super

Tabela Supplier:
ID: 3, Nazwa: Firma Super, Ulica: Cezarego 32, Miasto: Mloszowa, Produkty:
  - ID: 10, Nazwa: Pomidor, Ilość na stanie: 5
  - ID: 11, Nazwa: Orzech, Ilość na stanie: 10
  - ID: 12, Nazwa: Przenica, Ilość na stanie: 1500
```

IV. Modelowanie relacji wiele-do-wielu

Product:

```
public class Product
{
    public Product()
    {
        this.Invoices = new HashSet<Invoice>();
    }

    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public int UnitsOnStock { get; set; }
}
```

```

    public Supplier Supplier { get; set; }

    public virtual ICollection<Invoice> Invoices { get; set; }
}

```

Invoice:

```

public class Invoice
{
    public Invoice()
    {
        this.Products = new HashSet<Product>();
    }
    public int InvoiceID { get; set; }
    public int InvoiceNumber { get; set; }
    public int Quantity { get; set; }
    public virtual ICollection<Product> Products { get; set; }
}

```

ProductContext:

```

public class ProductContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    public DbSet<Invoice> Invoices { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("DataSource=ProductsDatabase4
    }
}

```

Supplier pozostaje taki sam jak w podpunkcie poprzednim.

Migracja:

```

PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabase4

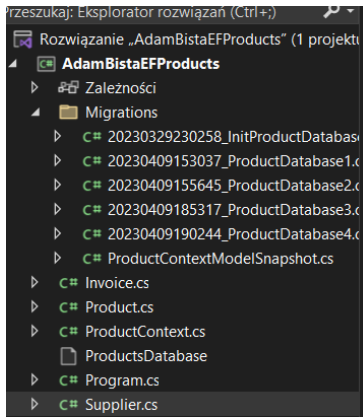
```

```

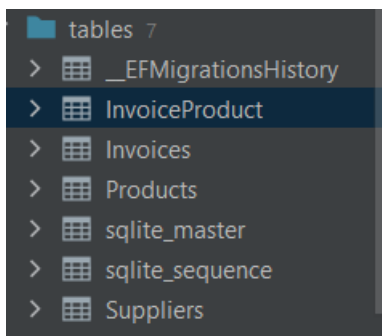
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update

```

Struktura:



DataGrip:



Dodajemy analogiczne dane jak w poprzednim podpunkcie:

Supplier:

	SupplierID	CompanyName	Street	City
1	1	Nowa firma	Kwiatowa 15	Warszawa

Product:

	ProductID	ProductName	SupplierID	UnitsOnStock
1	1	Pomidor	1	5
2	2	Orzech	1	10
3	3	Przenica	1	1500

a) Wpisujemy w Program.cs:

```
ProductContext productContext = new ProductContext();

Product product1 = productContext.Products.Find(1);
Product product2 = productContext.Products.Find(2);
Product product3 = productContext.Products.Find(3);

// nowe faktury ( czyli transakcje)
```

```

Invoice invoice1 = new Invoice
{
    InvoiceNumber = 101,
    Quantity = 5
};

Invoice invoice2 = new Invoice
{
    InvoiceNumber = 102,
    Quantity = 10
};

// Dodaj wybrane produkty do faktur
invoice1.Products.Add(product1);
invoice1.Products.Add(product2);

invoice2.Products.Add(product2);
invoice2.Products.Add(product3);

// Dodaj nowe faktury do bazy danych
productContext.Invoices.Add(invoice1);
productContext.Invoices.Add(invoice2);

// Zapisz zmiany w bazie danych
productContext.SaveChanges();

```

Wyniki Wywołań w datagripie:

InvoiceProduct:

	InvoicesInvoiceID	ProductsProductID
1	1	1
2	1	2
3	2	2
4	2	3

Invoice:

	InvoiceID	InvoiceNumber	Quantity
1	1	101	5
2	2	102	10

b) Program.cs:

```

ProductContext productContext = new ProductContext();

// Pobierz fakturę o numerze np 102
int selectedInvoiceNumber = 102;
Invoice selectedInvoice = productContext
    .Invoices
    .Include(i => i.Products)

```

```

        .SingleOrDefault(i => i.InvoiceNumber == selectedInvoiceNumber);

        if (selectedInvoice != null)
        {
            Console.WriteLine($"Numer faktury:
{selectedInvoice.InvoiceNumber}");
            Console.WriteLine("Sprzedane produkty:");

            foreach (var product in selectedInvoice.Products)
            {
                Console.WriteLine($"ID: {product.ProductID}, " +
                    $"Nazwa: {product.ProductName}, " +
                    $"Ilość na stanie: {product.UnitsOnStock}");
            }
        }
        else
        {
            Console.WriteLine("Faktura o podanym numerze nie istnieje.");
        }
    }
}

```

Wyniki w konsoli:

```

Numer faktury: 102
Sprzedane produkty:
ID: 2, Nazwa: Orzech, Ilość na stanie: 10
ID: 3, Nazwa: Przenica, Ilość na stanie: 1500

```

c) Program.cs:

```

ProductContext context = new ProductContext();
// Pobierz produkt o wybranym ID
int selectedProductId = 2;
Product selectedProduct = context.Products
    .Include(p => p.Invoices)
    .SingleOrDefault(p => p.ProductID == selectedProductId);

if (selectedProduct != null)
{
    Console.WriteLine($"Produkt: {selectedProduct.ProductName}");
    Console.WriteLine("Faktury, na których produkt został
sprzedany:");

    foreach (var invoice in selectedProduct.Invoices)
    {
        Console.WriteLine($"ID: {invoice.InvoiceID}, Numer faktury:
{invoice.InvoiceNumber}, Ilość: {invoice.Quantity}");
    }
}

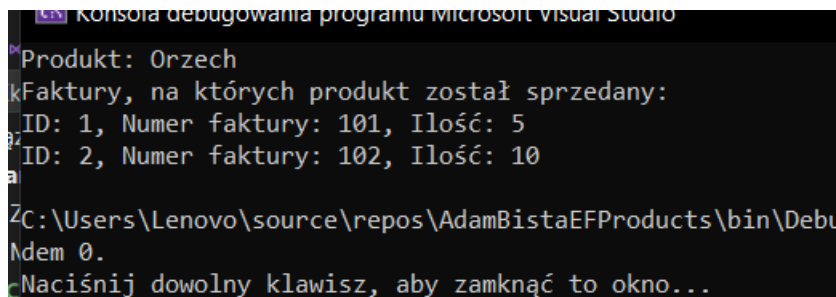
```

```

    }
    else
    {
        Console.WriteLine("Produkt o podanym ID nie istnieje.");
    }
}

```

Konsola:



```

Konsola debugowania programu Microsoft Visual Studio
Produkt: Orzech
Faktury, na których produkt został sprzedany:
ID: 1, Numer faktury: 101, Ilość: 5
ID: 2, Numer faktury: 102, Ilość: 10
ZC:\Users\Lenovo\source\repos\AdamBistaEFProducts\bin\Debug
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

V. Dziedziczenie

Table-Per-Hierarchy:

Company:

```

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; }
}

```

Supplier:

```

public class Supplier : Company
{
    public string BankAccountNumber { get; set; }
    public ICollection<Product> Products { get; set; }
}

```

Customer:

```

public class Customer : Company
{
    public double Discount { get; set; }
}

```

ProductContext:

```

public class ProductContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Company> Companies { get; set; } // Zamiast Suppliers

    public DbSet<Invoice> Invoices { get; set; }
}

```

```

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductsDatabase");
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Company>()
                .HasDiscriminator<string>("CompanyType")
                .HasValue<Supplier>("Supplier")
                .HasValue<Customer>("Customer");
        }
    }
}

```

Migracja:

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabaseTPH
```

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update
```

Wywołania w TPH:

Program.cs:

```

var context = new ProductContext();
context.Companies.AddRange(
    new Supplier
    {
        CompanyName = "Leszek",
        Street = "ul. Słoneczna 5",
        City = "Warszawa",
        ZipCode = "12-001",
        BankAccountNumber = "930390220"
    },
    new Supplier
    {
        CompanyName = "ZordonMonkey",
        Street = "ul. Wrocławska 23",
        City = "Kraków",
        ZipCode = "32-000",
        BankAccountNumber = "958654425"
    },
    new Customer
    {
        CompanyName = "AlfredDB",
        Street = "ul. Mickiewicza 10",
        City = "Gdańsk",
        ZipCode = "08-808",
        Discount = 0.7
    },
    new Customer
    {
        CompanyName = "AdamB",
        Street = "ul. Polna 15",
        City = "Wrocław",
        ZipCode = "77-808",
        Discount = 0.9
    }
);

context.SaveChanges();

```

DataGrip:

	CompanyID	BankAccountNumber	City	CompanyName	CompanyType	Street	ZipCode	Discount
1	5	<null>	Gdańsk	AlfredDB	Customer	ul. Mickiewicza 10	08-808	0.7
2	6	<null>	Wrocław	AdamB	Customer	ul. Polna 15	77-808	0.9
3	7	930390220	Warszawa	Leszek	Supplier	ul. Słoneczna 5	12-001	<null>
4	8	958654425	Kraków	ZordonMonkey	Supplier	ul. Wrocławska 23	32-000	<null>

Pobranie danych:

```
var context = new ProductContext();
var suppliers = context.Companies.OfType<Supplier>().ToList();
var customers = context.Companies.OfType<Customer>().ToList();
var companies = context.Companies.OfType<Company>().ToList();

Console.WriteLine("Table-Per-Hierarchy:");
Console.WriteLine("Suppliers:");
foreach (var supplier in suppliers)
{
    Console.WriteLine($"{supplier.CompanyName} {supplier.Street},
{supplier.City}, {supplier.ZipCode}, {supplier.BankAccountNumber}");
}
Console.WriteLine("Customers:");
foreach (var customer in customers)
{
    Console.WriteLine($"{customer.CompanyName} {customer.Street},
{customer.City}, {customer.ZipCode}, {customer.Discount}");
}
Console.WriteLine("Companies:");
foreach (var company in companies)
{
    Console.WriteLine($"{company.CompanyName} {company.Street},
{company.City}, {company.ZipCode}");
}
```

Konsola:

```
Table-Per-Hierarchy:
Suppliers:
Leszek ul. Słoneczna 5, Warszawa, 12-001, 930390220
ZordonMonkey ul. Wrocławska 23, Kraków, 32-000, 958654425
Customers:
AlfredDB ul. Mickiewicza 10, Gdańsk, 08-808, 0,7
AdamB ul. Polna 15, Wrocław, 77-808, 0,9
Companies:
AlfredDB ul. Mickiewicza 10, Gdańsk, 08-808
AdamB ul. Polna 15, Wrocław, 77-808
Leszek ul. Słoneczna 5, Warszawa, 12-001
ZordonMonkey ul. Wrocławska 23, Kraków, 32-000
```

VI. Table-Per-Type

Struktura tabel:

ProductContext:


```

public class ProductContext : DbContext
{
    //public DbSet<Product> Products { get; set; }
    public DbSet<Company> Companies { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    public DbSet<Customer> Customers { get; set; }

    //public DbSet<Invoice> Invoices { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("Datasource=ProductsDatabase");
    }
}

```

W Supplier oraz Customer dodałem na górze:

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

Supplier:

```

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

```

Customer:

```

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

```

Company pozostaje bez zmian jak w TPH

Migracja:

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef migrations add ProductDatabaseTPT
```

```
PS C:\Users\Lenovo\source\repos\AdamBistaEFProducts> dotnet ef database update
```

Program.cs:

```

ProductContext context = new ProductContext();
context.Suppliers.AddRange(
    new Supplier { SupplierID = 1, CompanyName = "Leszek", Street =
"Krakowska 15", City = "Poznań", ZipCode = "23-440", BankAccountNumber =
"154678966" },
    new Supplier { SupplierID = 2, CompanyName = "Wozny", Street =
"Miejscowa 3", City = "Gdańsk", ZipCode = "37-668", BankAccountNumber =
"989684999" }
);
context.Customers.AddRange(

```

```

new Customer { CustomerID = 1, CompanyName = "Adam", Street = "Zimowa
2", City = "Warszawa", ZipCode = "94-332", Discount = 0.8 },
new Customer { CustomerID = 2, CompanyName = "Michał", Street =
"Jesienna 20", City = "Jaworzno", ZipCode = "23-222", Discount = 0.7 }
);
context.SaveChanges();

```

DataGrip:

Company:

	CompanyID	CompanyName	Street	City	ZipCode
1	7	Adam	Zimowa 2	Warszawa	94-332
2	8	Michał	Jesienna 20	Jaworzno	23-222
3	9	Leszek	Krakowska 15	Poznań	23-440
4	10	Wozny	Miejscowa 3	Gdańs	37-668

Customer:

	CompanyID	CustomerID	Discount
1	7	1	0.8
2	8	2	0.7

Supplier:

	CompanyID	SupplierID	BankAccountNumber
1	9	1	154678966
2	10	2	989684999

Pobranie danych o Customers i Suppliers:

```
ProductContext context = new ProductContext();

// Pobieranie danych
var suppliers = context.Suppliers.ToList();
var customers = context.Customers.ToList();

// Wypisywanie danych
Console.WriteLine("Suppliers:");
foreach (var supplier in suppliers)
{
    Console.WriteLine($"SupplierID: {supplier.SupplierID},
CompanyName: {supplier.CompanyName}, Street: {supplier.Street}, City:
{supplier.City}, ZipCode: {supplier.ZipCode}, BankAccountNumber:
{supplier.BankAccountNumber}");
}

Console.WriteLine("\nCustomers:");
foreach (var customer in customers)
{
    Console.WriteLine($"CustomerID: {customer.CustomerID},
CompanyName: {customer.CompanyName}, Street: {customer.Street}, City:
{customer.City}, ZipCode: {customer.ZipCode}, Discount: {customer.Discount}");
}
```

Konsola:

```
Suppliers:
SupplierID: 1, CompanyName: Leszek, Street: Krakowska 15, City: Poznań, ZipCode: 23-440, BankAccountNumber: 154678966
SupplierID: 2, CompanyName: Wozny, Street: Miejsowa 3, City: Gdańs, ZipCode: 37-668, BankAccountNumber: 989684999

Customers:
CustomerID: 1, CompanyName: Adam, Street: Zimowa 2, City: Warszawa, ZipCode: 94-332, Discount: 0,8
CustomerID: 2, CompanyName: Michał, Street: Jesienna 20, City: Jaworzno, ZipCode: 23-222, Discount: 0,7

C:\Users\Lenovo\source\repos\AdamBistaEFProducts\bin\Debug\net7.0\AdamBistaEFProducts.exe (proces 14188) zakończono z ko
dem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

TPH a TPT:

W TPT trzeba tworzyć osobne tabele podczas dziedziczenia natomiast w TPH wystarczy jedna tabela.

W TPT trzeba tworzyć osobne klucze CustomerID oraz SupplierID

W TPH trzeba dodać linię w ProductContext:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Company>()
        .HasDiscriminator<string>("CompanyType")
        .HasValue<Supplier>("Supplier")
        .HasValue<Customer>("Customer");
}
```

