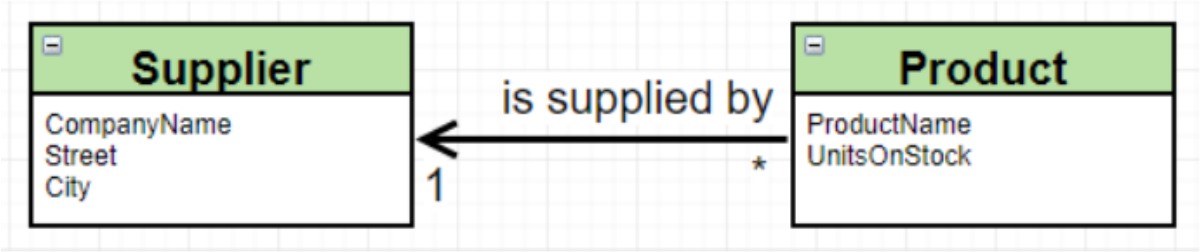
Skowron Mateusz

1. Modyfikacja modelu wprowadzając pojęcie dostawcy



Klasa ***Supplier***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 2
    public class Supplier
    {
        Odwołania: 0
        public int SupplierID { get; set; }
        Odwołania: 0
        public string CompanyName { get; set; }
        Odwołania: 0
        public string Street { get; set; }
        Odwołania: 0
        public string City { get; set; }
    }
}
```

Klasa ***Product***

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    1 odwołanie
    public class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 0
        public string ProductName { get; set; }
        Odwołania: 0
        public int UnitsOnStock { get; set; }
        [ForeignKey("Supplier")]
        Odwołania: 0
        public int SupplierId { get; set; }
        Odwołania: 0
        public Supplier Supplier { get; set; }
    }
}
```

Modyfikacja klasy ***DbContext***

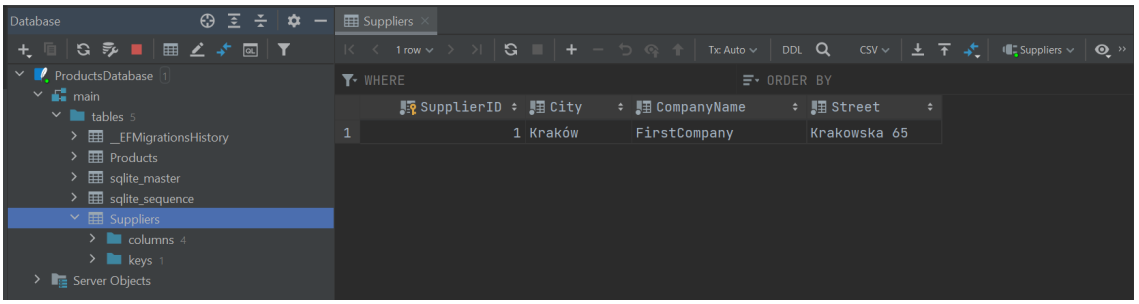
```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    Odwołania: 6
    public class ProductContext: DbContext
    {
        Odwołania: 2
        public DbSet<Product> Products { get; set; }
        Odwołania: 0
        public DbSet<Supplier> Suppliers { get; set; }

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

}
```

Migracja i update bazy danych

```
Windows PowerShell

PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migra
tions add SupplierOneToManyMigration
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef datab
ase update
Build started...
Build succeeded.
Applying migration '20220424205837_SupplierOneToManyMigration'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

Stwórz nowego dostawcę.

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            string compName = "FirstCompany";
            string street = "Krakowska 65";
            string city = "Kraków";

            ProductContext productContext = new ProductContext();
            Supplier supplier = new Supplier { CompanyName = compName, Street = street, City = city };

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

        }
    }
}
```



Wprowadziłem dane do tabel.

Tabela ***Products*** :

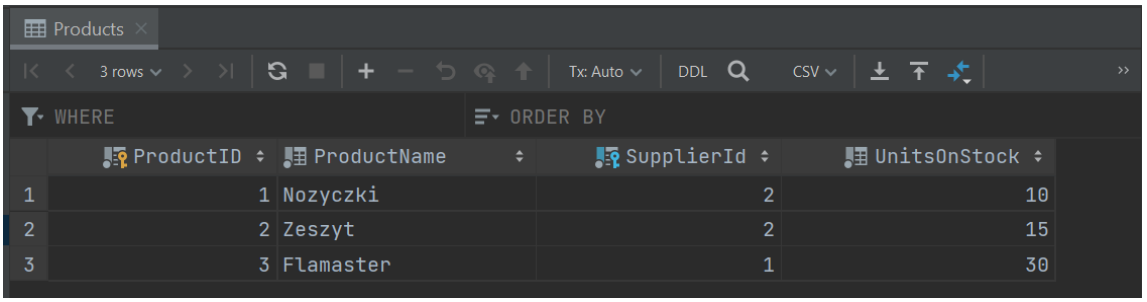
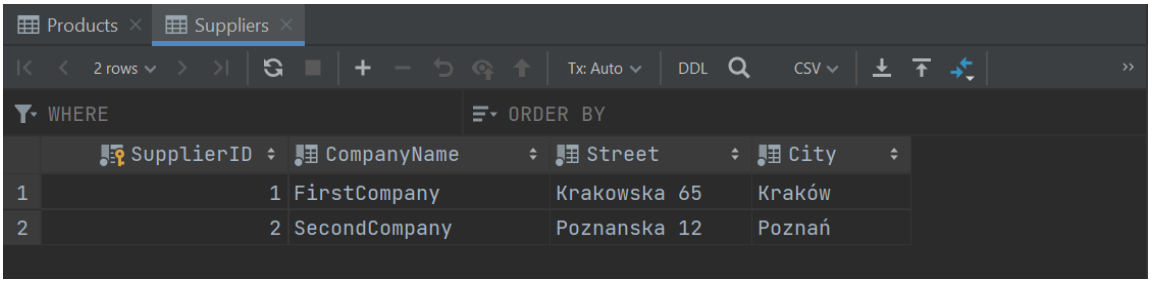


Table ***Suppliers***:



a. Znajdź poprzednio wprowadzony produkt i ustaw jego dostawcę na właśnie dodanego.

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            var product = productContext.Products.SingleOrDefault(p => p.ProductName == "Nozyczki");
            if(product != null)
            {
                product.SupplierId = 1;
                productContext.SaveChanges();
            }
        }
    }
}
```

Tabela ***Products*** po zmianie:

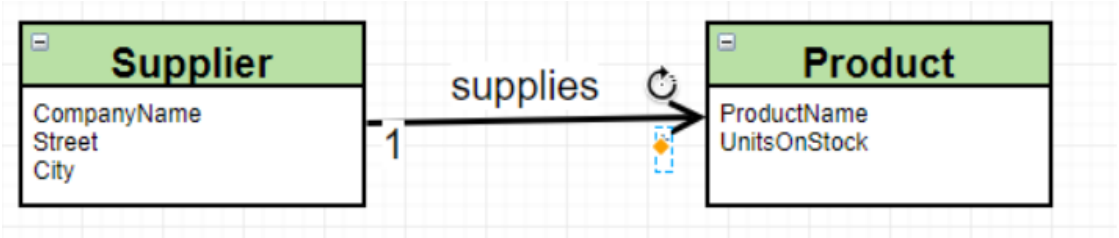
Products

3 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	ProductID	ProductName	SupplierId	UnitsOnStock
1	1	Nozyczki	1	10
2	2	Zeszyt	2	15
3	3	Flamaster	1	30

2. Odwróć relację



W klasie ***ProductContext*** nie wprowadziłem żadnych zmian względem poprzedniego podpunktu.

Klasa ***Product***:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    Odwołania: 9
    public class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 5
        public string ProductName { get; set; }
        Odwołania: 5
        public int UnitsOnStock { get; set; }

        [ForeignKey("Supplier")]
        Odwołania: 0
        public int SupplierId { get; set; }

    }
}
```

Klasa ***Supplier***

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

namespace MateuszSkowronEFProducts
{
    1 odwołanie
    public class Supplier
    {
        Odwołania: 0
        public int SupplierID { get; set; }
        Odwołania: 0
        public string CompanyName { get; set; }
        Odwołania: 0
        public string Street { get; set; }
        Odwołania: 0
        public string City { get; set; }

        Odwołania: 0
        public ICollection<Product> Products { get; set; }
    }
}
```

Migracja oraz update bazy danych

```
Windows PowerShell

PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add ProductOneToManySuppliersMigration
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20220424233929_ProductOneToManySuppliersMigration'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

a. Stwórz kilka produktów
b. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę

```
namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            string compName = "NewCompany";
            string street = "Zamorska 160";
            string city = "Gdańsk";

            List<Product> products = new List<Product>();
            products.Add(new Product { ProductName = "Flamaster", UnitsOnStock = 10 });
            products.Add(new Product { ProductName = "Nozyczki", UnitsOnStock = 20 });
            products.Add(new Product { ProductName = "Linijka", UnitsOnStock = 30 });
            products.Add(new Product { ProductName = "Dlugopis", UnitsOnStock = 40 });
            products.Add(new Product { ProductName = "Zszywacz", UnitsOnStock = 50 });

            Supplier supplier = new Supplier { CompanyName = compName, Street = street, City = city , Products = products };
            productContext.Suppliers.Add(supplier);
            productContext.SaveChanges();
        }
    }
}
```

Tabela ***Suppliers***

Products [ProductsDatabase] | Suppliers [ProductsDatabase]

1 row | Tx: Auto | DDL | CSV | Suppliers

WHERE | ORDER BY

	SupplierID	City	CompanyName	Street
1	1	Gdańsk	NewCompany	Zamorska 160

Tabela ***Products***

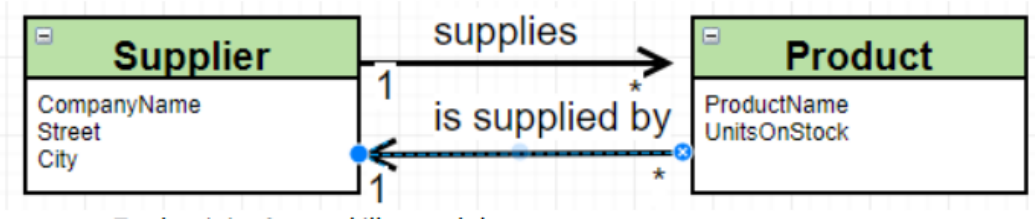
Products [ProductsDatabase] | Suppliers [ProductsDatabase]

5 rows | Tx: Auto | DDL | CSV | Products

WHERE | ORDER BY

	ProductID	ProductName	SupplierId	UnitsOnStock
1	1	Flamaster	1	10
2	2	Nozyczki	1	20
3	3	Linijka	1	30
4	4	Dlugopis	1	40
5	5	Zszywacz	1	50

3. Zamodeluj relację dwustronną



Klasa ***Product***

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    Odwołania: 9
    public class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 5
        public string ProductName { get; set; }
        Odwołania: 5
        public int UnitsOnStock { get; set; }

        [ForeignKey("Supplier")]
        Odwołania: 0
        public int SupplierId { get; set; }
        Odwołania: 0
        public Supplier Supplier { get; set; }

    }
}
```

Klasa ***Supplier***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 4
    public class Supplier
    {
        Odwołania: 0
        public int SupplierID { get; set; }
        1 odwołanie
        public string CompanyName { get; set; }
        1 odwołanie
        public string Street { get; set; }
        1 odwołanie
        public string City { get; set; }

        1 odwołanie
        public ICollection<Product> Products { get; set; }
    }
}
```

Migracja oraz update bazy danych

```
Windows PowerShell
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add OneToNandNtoOne
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20220425090747_OneToNandNtoOne'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

a. Stwórz kilka produktów
b. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            string compName = "NewCompany";
            string street = "Śródmieście 160";
            string city = "Warszawa";

            Supplier supplier = new Supplier { CompanyName = compName, Street = street, City = city };
            productContext.Suppliers.Add(supplier);

            List<Product> products = new List<Product>();
            products.Add(new Product { ProductName = "Myszka", UnitsOnStock = 10, Supplier = supplier });
            products.Add(new Product { ProductName = "Monitor", UnitsOnStock = 20, Supplier = supplier });
            products.Add(new Product { ProductName = "Słuchawki", UnitsOnStock = 30, Supplier = supplier });
            products.Add(new Product { ProductName = "Pendrive", UnitsOnStock = 40, Supplier = supplier });
            products.Add(new Product { ProductName = "Laptop", UnitsOnStock = 50, Supplier = supplier });

            productContext.Products.AddRange(products);
            productContext.SaveChanges();
        }
    }
}
```

Tabela ***Suppliers***

Suppliers [ProductsDatabase]

1 row | Tx: Auto | DDL | CSV | Suppliers

WHERE | ORDER BY

	SupplierID	City	CompanyName	Street
1	1	Warszawa	NewCompany	Śródmieście 160

Tabela ***Products***

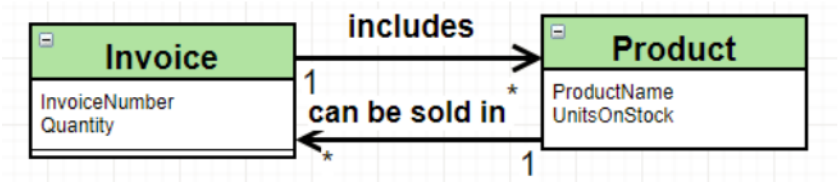
Products [ProductsDatabase]

5 rows | Tx: Auto | DDL | CSV | Products

WHERE | ORDER BY

	ProductID	ProductName	SupplierId	UnitsOnStock
1	1	Myszka	1	10
2	2	Monitor	1	20
3	3	Słuchawki	1	30
4	4	Pendrive	1	40
5	5	Laptop	1	50

4. Zamodeluj relację wiele-do-wielu



Klasa ***InvoiceProduct***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 6
    public class InvoiceProduct
    {
        1 odwołanie
        public int InvoiceId { get; set; }
        Odwołania: 0
        public Invoice Invoice { get; set; }
        1 odwołanie
        public int ProductId { get; set; }
        Odwołania: 0
        public Product Product { get; set; }
    }
}
```

Klasa ***Product***

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    Odwołania: 3
    public class Product
    {
        Odwołania: 0
        public Product()
        {
            this.InvoiceProducts = new HashSet<InvoiceProduct>();
        }
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 0
        public string ProductName { get; set; }
        Odwołania: 0
        public int UnitsOnStock { get; set; }
        1 odwołanie
        public virtual ICollection<InvoiceProduct> InvoiceProducts { get; set; }
    }
}
```

Klasa ***Invoice***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 3
    public class Invoice
    {
        Odwołania: 0
        public Invoice()
        {
            this.InvoiceProducts = new HashSet<InvoiceProduct>();
        }
        [Key]
        Odwołania: 0
        public int InvoiceNumber { get; set; }
        Odwołania: 0
        public int Quantity { get; set; }

        1 odwołanie
        public virtual ICollection<InvoiceProduct> InvoiceProducts { get; set; }
    }
}
```

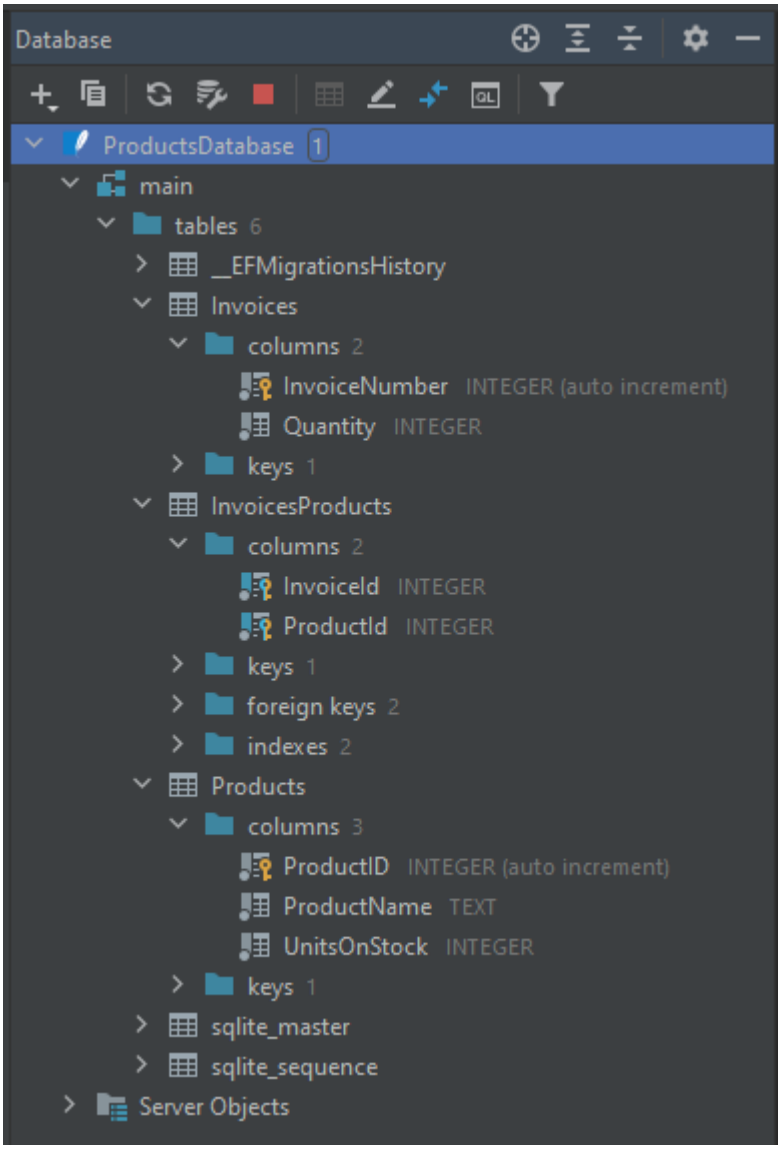
Klasa ***kontekstowa***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 33
    public class ProductContext: DbContext
    {
        Odwołania: 0
        public DbSet<Product> Products { get; set; }
        Odwołania: 0
        public DbSet<Invoice> Invoices { get; set; }
        Odwołania: 0
        public DbSet<InvoiceProduct> InvoicesProducts { get; set; }
        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductsDatabase");
        }

        Odwołania: 0
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceProduct>().HasKey(sc => new { sc.InvoiceId, sc.ProductId });
        }
    }
}
```

Migracja i update bazy danych

```
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add ManyToMany
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add ManyToManyApply
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20220425092753_ManyToMany'.
Applying migration '20220425092805_ManyToManyApply'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

Struktura bazy prezentuje się następująco



a. Stwórz kilka produktów i “sprzedaj” je na kilku transakcjach

Dodajmy kilka produktów

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            List<Product> products = new List<Product>();
            products.Add(new Product { ProductName = "Myszka", UnitsOnStock = 10 });
            products.Add(new Product { ProductName = "Monitor", UnitsOnStock = 20});
            products.Add(new Product { ProductName = "Słuchawki", UnitsOnStock = 30});
            products.Add(new Product { ProductName = "Pendrive", UnitsOnStock = 40});
            products.Add(new Product { ProductName = "Laptop", UnitsOnStock = 50 });

            productContext.Products.AddRange(products);
            productContext.SaveChanges();
        }
    }
}
```

Products [ProductsDatabase]

5 rows Tx: Auto DDL

WHERE ORDER BY

	ProductID	ProductName	UnitsOnStock
1	1	Myszka	10
2	2	Monitor	20
3	3	Słuchawki	30
4	4	Pendrive	40
5	5	Laptop	50

Sprzedajmy myszkę w ilości 5

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            var mouse = productContext.Products.SingleOrDefault(p => p.ProductName == "Myszka"); ;

            if(mouse != null && mouse.UnitsOnStock >= 5)
            {
                Invoice invoiceOne = new Invoice{ Quantity = 5};
                InvoiceProduct invoiceProduct = new InvoiceProduct { Invoice = invoiceOne, Product = mouse };

                productContext.InvoicesProducts.Add(invoiceProduct);

                mouse.UnitsOnStock -= 5;
            }
            productContext.SaveChanges();
        }
    }
}
```

Tabela ***Invoices***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceNumber	Quantity
1	1	5

Tabela ***InvoicesProducts***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceId	ProductId
1	1	1

Tabela ***Products***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

5 rows | Tx: Auto | DDL

WHERE | ORDER BY

	ProductID	ProductName	UnitsOnStock
1	1	Myszka	5
2	2	Monitor	20
3	3	Słuchawki	30
4	4	Pendrive	40
5	5	Laptop	50

Sprzedajmy każdy produkt w ilości 1

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            var mouse = productContext.Products.SingleOrDefault(p => p.ProductName == "Myszka"); ;
            var monitor = productContext.Products.SingleOrDefault(p => p.ProductName == "Monitor");
            var headPhones = productContext.Products.SingleOrDefault(p => p.ProductName == "Słuchawki");
            var pendrive = productContext.Products.SingleOrDefault(p => p.ProductName == "Pendrive");
            var laptop = productContext.Products.SingleOrDefault(p => p.ProductName == "Laptop");

            if(mouse != null && monitor != null && headPhones != null && pendrive != null && laptop != null
                && mouse.UnitsOnStock >= 1 && monitor.UnitsOnStock >= 1 && headPhones.UnitsOnStock >= 1
                && pendrive.UnitsOnStock >= 1 && laptop.UnitsOnStock >= 1)
            {
                Invoice invoiceOne = new Invoice{ Quantity = 5};
                List<InvoiceProduct> invoiceProducts = new List<InvoiceProduct>();

                InvoiceProduct invoiceProductMouse = new InvoiceProduct { Invoice = invoiceOne, Product = mouse };
                InvoiceProduct invoiceProductMonitor = new InvoiceProduct { Invoice = invoiceOne, Product = monitor };
                InvoiceProduct invoiceProductHeadPhones = new InvoiceProduct { Invoice = invoiceOne, Product = headPhones };
                InvoiceProduct invoiceProductPendrive = new InvoiceProduct { Invoice = invoiceOne, Product = pendrive };
                InvoiceProduct invoiceProductLaptop = new InvoiceProduct { Invoice = invoiceOne, Product = laptop };

                invoiceProducts.Add(invoiceProductMouse);
                invoiceProducts.Add(invoiceProductMonitor);
                invoiceProducts.Add(invoiceProductHeadPhones);
                invoiceProducts.Add(invoiceProductPendrive);
                invoiceProducts.Add(invoiceProductLaptop);

                productContext.InvoicesProducts.AddRange(invoiceProducts);

                mouse.UnitsOnStock -= 1;
                monitor.UnitsOnStock -= 1;
                headPhones.UnitsOnStock -= 1;
                pendrive.UnitsOnStock -= 1;
                laptop.UnitsOnStock -= 1;
            }
            productContext.SaveChanges();
        }
    }
}
```

Tabela ***Invoices***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

2 rows | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceNumber	Quantity
1	1	5
2	2	5

Tabela ***InvoiceProduct***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

6 rows | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceId	ProductId
1	1	1
2	2	1
3	2	2
4	2	3
5	2	4
6	2	5

Tabela ***Products***

Invoices [ProductsDatabase [2]] | InvoicesProducts [ProductsDatabase [2]] | Products [ProductsDatabase [2]]

5 rows | Tx: Auto | DDL

WHERE | ORDER BY

	ProductID	ProductName	UnitsOnStock
1	1	Myszka	4
2	2	Monitor	19
3	3	Słuchawki	29
4	4	Pendrive	39
5	5	Laptop	49

b. Pokaż produktu sprzedane w ramach wybranej faktury

Produkty sprzedane w ramach faktury 2

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            var query = from invoiceproduct in productContext.InvoicesProducts
                        join product in productContext.Products
                        on invoiceproduct.ProductId equals product.ProductID
                        where invoiceproduct.InvoiceId == 2
                        select new
                        {
                            productName = product.ProductName
                        };
            foreach (var product in query)
            {
                Console.WriteLine(product.productName);
            }
        }
    }
}
```

```
Konsola debugowania programu Microsoft Visual Studio
Myszka
Monitor
Słuchawki
Pendrive
Laptop

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronEFProducts\bin\Debug\net6.0\MateuszSkowronEF
Products.exe (proces 24404) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje
-> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

c. Pokaż faktury w ramach których był sprzedany wybrany produkt

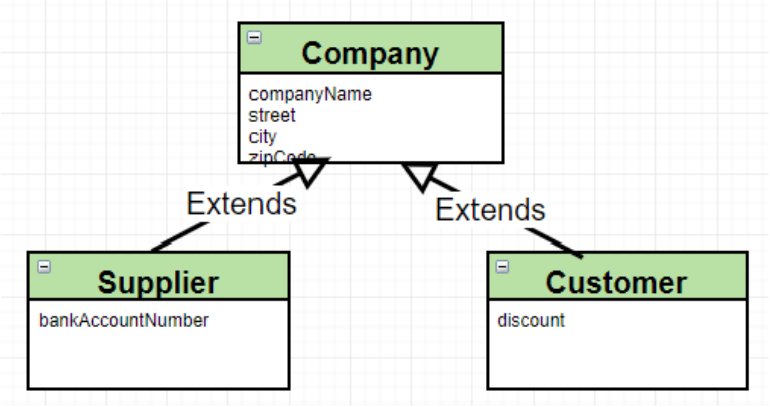
Faktury w ramach których był sprzedany produkt o Id równym 1

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            var query = from invoiceproduct in productContext.InvoicesProducts
                        join product in productContext.Products
                        on invoiceproduct.ProductId equals product.ProductID
                        where invoiceproduct.ProductId == 1
                        select new
                        {
                            invoiceID = invoiceproduct.InvoiceId
                        };
            foreach (var invoice in query)
            {
                Console.WriteLine(invoice.invoiceID);
            }
        }
    }
}
```

```
Konsola debugowania programu Microsoft Visual Studio
1
2

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronEFProducts\bin\Debug\net6.0\MateuszSkowronEFProducts
.exe (proces 24468) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debug
owanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

5. Dziedziczenie

a. Wprowadź do modelu poniższą hierarchię dziedziczenia używając strategii *Table-Per-Hierarchy.*



Klasa bazowa ***Company***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 3
    public abstract class Company
    {
        Odwołania: 0
        public int CompanyId { get; set; }
        Odwołania: 0
        public string companyName { get; set; }
        Odwołania: 0
        public string street { get; set; }
        Odwołania: 0
        public string city { get; set; }
        Odwołania: 0
        public string zipCode { get; set; }
        Odwołania: 0
        public string Discriminator { get; private set; }

    }
}
```

Klasa ***Customer*** dziedzicząca po *Company*

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

namespace MateuszSkowronEFProducts
{
    1 odwołanie
    public class Customer : Company
    {
        Odwołania: 0
        public float discount { get; set; }
    }
}
```

Klasa ***Supplier*** dziedzicząca po *Company*

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

namespace MateuszSkowronEFProducts
{
    1 odwołanie
    public class Supplier : Company
    {
        Odwołania: 0
        public string bankAccountNumber { get; set; }
    }
}
```

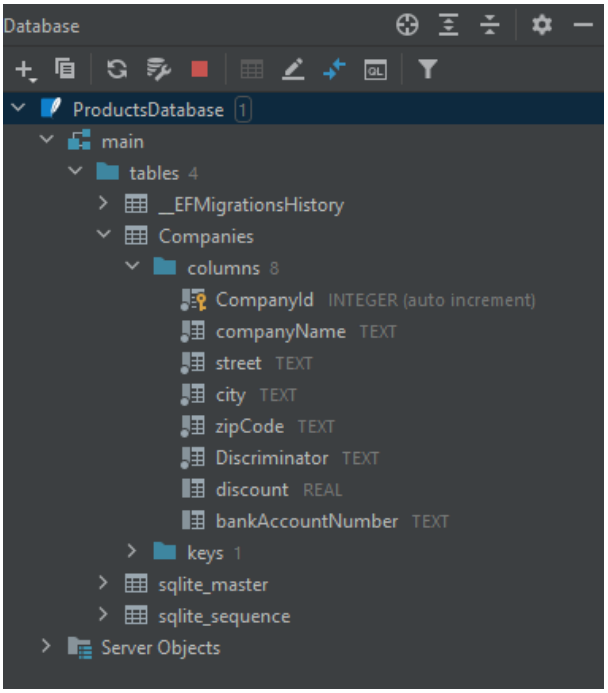
Klasa ***kontekstowa***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 33
    public class ProductContext: DbContext
    {
        Odwołania: 0
        public DbSet<Company> Companies { get; set; }
        Odwołania: 0
        public DbSet<Customer> Customers { get; set; }
        Odwołania: 0
        public DbSet<Supplier> Suppliers { get; set; }
        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductsDatabase");
        }

    }
}
```

Migracja i update bazy danych

```
Windows PowerShell
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add TablePerHierarchyMigration
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add TablePerHierarchyMigrationApply
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20220425140915_TablePerHierarchyMigration'.
Applying migration '20220425140925_TablePerHierarchyMigrationApply'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

Struktura bazy danych



b. Dodaj i pobierz z bazy kilka firm obu rodzajów

Dodanie dwóch dostawców

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            List<Company> companies = new List<Company>();
            companies.Add(new Supplier
            {
                companyName = "KrakówSuppliers",
                street = "Krakowska 65",
                city = "Kraków",
                zipCode = "38-203",
                bankAccountNumber = "1050 0099 7603 1234 5678 9123"
            });
            companies.Add(new Supplier
            {
                companyName = "PoznańSuppliers",
                street = "Poznańska 12",
                city = "Poznań",
                zipCode = "12-213",
                bankAccountNumber = "1231 1223 7603 1234 6343 2343"
            });

            context.Companies.AddRange(companies);
            context.SaveChanges();

        }
    }
}
```

Companies [ProductsDatabase [2]]

2 rows | Tx: Auto | DDL | CSV | Companies

WHERE | ORDER BY

	CompanyId	companyName	street	city	zipCode	Discriminator	discount	bankAccountNumber
1	1	KrakówSuppliers	Krakowska 65	Kraków	38-203	Supplier	<null>	1050 0099 7603 1234 5678 9123
2	2	PoznańSuppliers	Poznańska 12	Poznań	12-213	Supplier	<null>	1231 1223 7603 1234 6343 2343

Dodanie dwóch klientów

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            List<Company> companies = new List<Company>();
            companies.Add(new Customer
            {
                companyName = "KrakówCustomers",
                street = "Rynek 15",
                city = "Kraków",
                zipCode = "04-252",
                discount = 0.05F
            });
            companies.Add(new Customer
            {
                companyName = "PoznańCustomers",
                street = "Śródmieście 421",
                city = "Poznań",
                zipCode = "83-121",
                discount = 0.10F
            });

            context.Companies.AddRange(companies);
            context.SaveChanges();

        }
    }
}
```

Companies [ProductsDatabase [2]]

4 rows | Tx: Auto | DDL | CSV | Companies

WHERE | ORDER BY

	CompanyId	companyName	street	city	zipCode	Discriminator	discount	bankAccountNumber
1	1	KrakówSuppliers	Krakowska 65	Kraków	38-203	Supplier	<null>	1050 0099 7603 1234 5678 9123
2	2	PoznańSuppliers	Poznańska 12	Poznań	12-213	Supplier	<null>	1231 1223 7603 1234 6343 2343
3	3	KrakówCustomers	Rynek 15	Kraków	04-252	Customer	0.05	<null>
4	4	PoznańCustomers	Śródmieście 421	Poznań	83-121	Customer	0.1	<null>

Pobieranie dostawców

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            var query = from company in context.Companies
                        where company.Discriminator == "Supplier"
                        select new
                        {
                            companyName = company.companyName
                        };

            foreach (var company in query)
            {
                Console.WriteLine(company.companyName);
            }

        }
    }
}
```

```
Konsola debugowania programu Microsoft Visual Studio
KrakówSuppliers
PoznańSuppliers

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronEFProducts\b
zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania,
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Pobieranie klientów

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            var query = from company in context.Companies
                        where company.Discriminator == "Customer"
                        select new
                        {
                            companyName = company.companyName
                        };

            foreach (var company in query)
            {
                Console.WriteLine(company.companyName);
            }

        }
    }
}
```

```
Konsola debugowania programu Microsoft Visual Studio
KrakówCustomers
PoznańCustomers

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronEFProducts\bin\Debug\net6
 zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

6. **Zamodeluj tę samą hierarchię dziedziczenia, ale tym razem użyj strategii *Table-Per-Type***

Klasa bazowa ***Company***

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    [Table("Companies")]
    Odwołania: 3
    public class Company
    {
        Odwołania: 0
        public int CompanyId { get; set; }
        Odwołania: 0
        public string companyName { get; set; }
        Odwołania: 0
        public string street { get; set; }
        Odwołania: 0
        public string city { get; set; }
        Odwołania: 0
        public string zipCode { get; set; }
    }
}
```

Klasa ***Customer*** dziedzicząca po *Company*

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    [Table("Customers")]
    1 odwołanie
    public class Customer : Company
    {
        Odwołania: 0
        public float discount { get; set; }
    }
}
```

Klasa ***Supplier*** dziedzicząca po *Company*

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MateuszSkowronEFProducts
{
    [Table("Suppliers")]
    1 odwołanie
    public class Supplier : Company
    {
        Odwołania: 0
        public string bankAccountNumber { get; set; }
    }
}
```

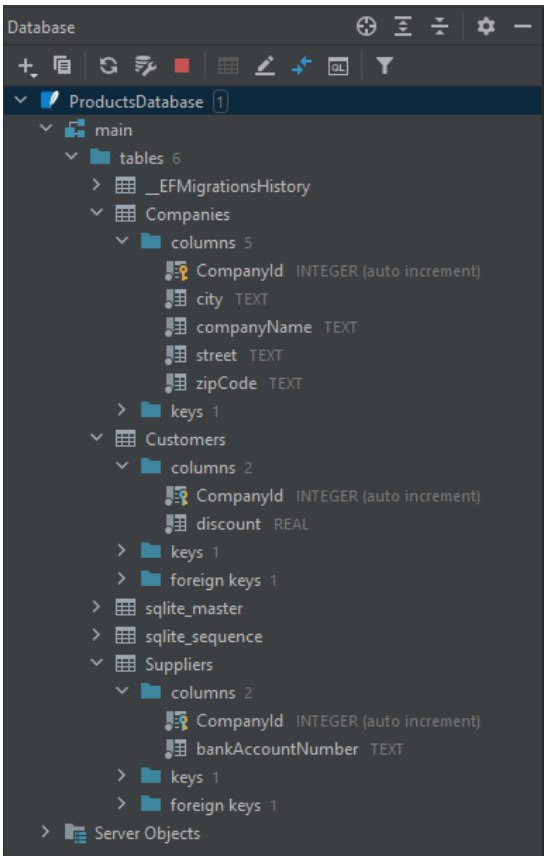
Klasa ***kontekstowa***

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

namespace MateuszSkowronEFProducts
{
    Odwołania: 38
    public class ProductContext: DbContext
    {
        Odwołania: 0
        public DbSet<Company> Companies { get; set; }
        Odwołania: 0
        public DbSet<Customer> Customers { get; set; }
        Odwołania: 0
        public DbSet<Supplier> Suppliers { get; set; }
        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("DataSource=ProductsDatabase");
        }

        Odwołania: 0
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Customer>().ToTable("Customers")
                .Property(c => c.discount)
                .IsRequired();
            modelBuilder.Entity<Supplier>().ToTable("Suppliers")
                .Property(s => s.bankAccountNumber)
                .IsRequired();
        }
    }
}
```

Migracja i update bazy danych

```
Windows PowerShell
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add TablePerTypeMigration
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef migrations add TablePerTypeMigrationApply
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20220425144612_TablePerTypeMigration'.
Applying migration '20220425144621_TablePerTypeMigrationApply'.
Done.
PS C:\Users\HP\Desktop\C#.NET etc\lab\MateuszSkowronEFProducts>
```

Struktura bazy danych



a. Dodaj i pobierz z bazy kilka firm obu rodzajów

Dodanie dostawców

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            List<Company> companies = new List<Company>();
            companies.Add(new Supplier
            {
                companyName = "KrakówSuppliers",
                street = "Krakowska 65",
                city = "Kraków",
                zipCode = "38-203",
                bankAccountNumber = "1050 0099 7603 1234 5678 9123"
            });
            companies.Add(new Supplier
            {
                companyName = "PoznańSuppliers",
                street = "Poznańska 12",
                city = "Poznań",
                zipCode = "12-213",
                bankAccountNumber = "1231 1223 7603 1234 6343 2343"
            });

            context.Companies.AddRange(companies);
            context.SaveChanges();

        }
    }
}
```

Tabela ***Companies***

Companies [ProductsDatabase [2]] | Suppliers [ProductsDatabase [2]]

2 rows | Tx: Auto | DDL

WHERE | ORDER BY

	CompanyId	city	companyName	street	zipCode
1	5	Kraków	KrakówSuppliers	Krakowska 65	38-203
2	6	Poznań	PoznańSuppliers	Poznańska 12	12-213

Tabela ***Suppliers***

Companies [ProductsDatabase [2]] | Suppliers [ProductsDatabase [2]]

2 rows | Tx: Auto | DDL

WHERE | ORDER BY

	CompanyId	bankAccountNumber
1	5	1050 0099 7603 1234 5678 9123
2	6	1231 1223 7603 1234 6343 2343

Dodanie klientów

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            List<Company> companies = new List<Company>();
            companies.Add(new Customer
            {
                companyName = "KrakówCustomers",
                street = "Rynek 15",
                city = "Kraków",
                zipCode = "04-252",
                discount = 0.05F
            });
            companies.Add(new Customer
            {
                companyName = "PoznańCustomers",
                street = "Śródmieście 421",
                city = "Poznań",
                zipCode = "83-121",
                discount = 0.10F
            });

            context.Companies.AddRange(companies);
            context.SaveChanges();

        }
    }
}
```

Tabela ***Companies***

Companies [ProductsDatabase [2]]

4 rows | Tx: Auto | DDL

WHERE | ORDER BY

	CompanyId	city	companyName	street	zipCode
1	5	Kraków	KrakówSuppliers	Krakowska 65	38-203
2	6	Poznań	PoznańSuppliers	Poznańska 12	12-213
3	7	Kraków	KrakówCustomers	Rynek 15	04-252
4	8	Poznań	PoznańCustomers	Śródmieście 421	83-121

Tabela ***Customers***

Customers [ProductsDatabase [2]]

2 rows

WHERE | OR

	CompanyId	discount
1	7	0.05
2	8	0.1

Pobieranie dostawców

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            var query = from company in context.Companies
                        join supplier in context.Suppliers
                        on company.CompanyId equals supplier.CompanyId
                        select new
                        {
                            companyName = company.companyName
                        };

            foreach(var company in query)
            {
                Console.WriteLine(company.companyName);
            }
        }
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
KrakówSuppliers
PoznańSuppliers

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronEFProduc
 zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowa
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Pobieranie klientów

```
using System;
using System.Threading;

namespace MateuszSkowronEFProducts
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext context = new ProductContext();

            var query = from company in context.Companies
                        join customer in context.Customers
                        on company.CompanyId equals customer.CompanyId
                        select new
                        {
                            companyName = company.companyName
                        };

            foreach(var company in query)
            {
                Console.WriteLine(company.companyName);
            }
        }
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
KrakówCustomers
PoznańCustomers

C:\Users\HP\Desktop\C#.NET etc\Lab\MateuszSkowronE
zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu d
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

b. Porównaj obie strategie modelowania dziedziczenia

Porównując oba rozwiązania myślę, że najlepiej będzie wskazać zarówno wady i zalety każdego z osobna.

Table-Per-Hierarchy
Do zalet tego podejścia na pewno należy szybkość wykonywania operacji CRUD, ponieważ wszystkie dane znajdują się w jednej tabeli. Kolejną zaletą jest minimalizacja liczby tabel, gdyż tak jak w przykładzie powyżej trzy klasy zmieściliśmy w jednej tabeli.
Wadą tego podejścia jest zdecydowanie redundancja danych. Niektóre kolumny są NULL'ami, a może być ich bardzo dużo. Złożoność modyfikacji klas. Usuwanie/dodawanie atrybutów wymaga dodania/usunięcia kolumny z każdej z klas, gdyż wszystkie znajdują się w jednej tabeli.

Table-Per-Type
Do zalet tego podejścia należy na pewno spójność danych. Wszystkie tabele odpowiadają trzeciej postaci normalnej. Nie występuje redundancja danych. Mamy dużą elastyczność jeśli chodzi o modyfikacje obiektów. Możemy bez problemu dodawać/usuwać kolejne atrybuty klas, gdyż nie mają one wpływu na pozostałe.
Widoczną wadą jest szybkość wykonywania operacji CRUD, gdyż nasze zapytania bardzo często wymagają join'ów. Kolejną wadą jest duża liczba tabel w bazie danych.

Mi osobiście wygodniej pracuje się z drugim podejściem, gdyż jest ono bardziej intuicyjne.