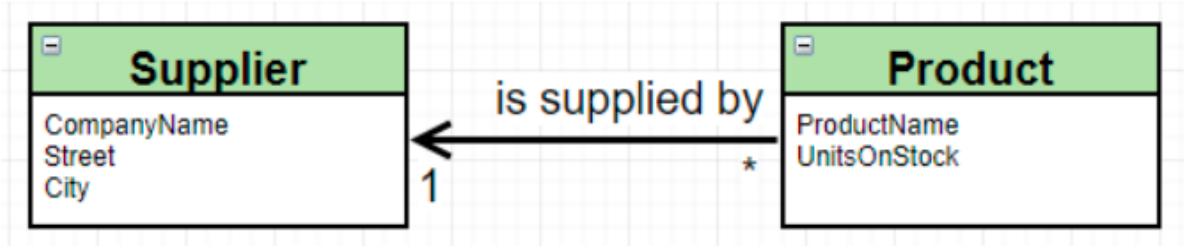
Laboratorium 2: Entity Framework

Autor: Krzysztof Solecki

1. Modyfikacja modelu i dodanie obiektu Supplier:



- Klasa Supplier:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 4
    public class Supplier
    {
        Odwołania: 0
        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: 0
        public override string ToString()
        {
            return CompanyName;
        }
    }
}
```

- Klasa Product:

```
8   namespace KrzysztofSoleckiEFLab
9
    Odwołania: 6
10  internal class Product
11  {
        Odwołania: 0
12      public int ProductID {  get; set; }
        Odwołania: 2
13      public string ProductName { get; set; }
        Odwołania: 2
14      public int UnitsOnStock { get; set; }
15
16      [ForeignKey("Supplier")]
        Odwołania: 0
17      public int SupplierID { get; set; }
        1 odwołanie
18      public Supplier? Supplier { get; set; }
19
        Odwołania: 0
20      public override string ToString()
21      {
22          return $"{ProductName}, units on stock: {UnitsOnStock}";
23      }
24
25  }
26
27
```

- Klasa ProductContext:

```
8   namespace KrzysztofSoleckiEFLab
9   {
        Odwołania: 6
10      internal class ProductContext :DbContext
11      {
            1 odwołanie
12          public DbSet<Product> Products { get; set;}
            Odwołania: 0
13          public DbSet<Supplier> Suppliers { get; set;}
14
            Odwołania: 0
15          protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
16          {
17              base.OnConfiguring(optionsBuilder);
18              optionsBuilder.UseSqlite("Datasource=ProductDatabase");
19          }
20      }
21  }
22
```

Migracja i update bazy danych:

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add SupplierOn
eToManyMigration
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the l
atest features and bug fixes. See https://aka.ms/AAc1fbw for more information.
Done. To undo this action, use 'ef migrations remove'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the l
atest features and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418202700_SupplierOneToManyMigration'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>
```

A. Stwórz nowego dostawcę:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 0
    class Program
    {    Liczba odwołań dla metoda: 0. (Alt+2)
        Odwołania: 0
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();
            Supplier supplier = createNewSupplier();
            productContext.Suppliers.Add(supplier);
            productContext.SaveChanges();
        }

        1 odwołanie
        private static Supplier createNewSupplier() {
            Console.WriteLine("Type Company Name: ");
            string? companyName = Console.ReadLine();
            Console.WriteLine("Type Street: ");
            string? street = Console.ReadLine();
            Console.WriteLine("Type City: ");
            string? city = Console.ReadLine();
            Console.WriteLine("Supplier sucessfully added to database!");

            return new Supplier
            {
                CompanyName = companyName,
                City = city,
                Street = street
            };
        }

    }
}
```

```
C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab\bin\Debug\net7.0\KrzysztofSoleckiEFLab.exe
Type Company Name:
FirstCompany
Type Street:
Krakowska 65
Type City:
Kraków
Supplier sucessfully added to database!
```

Suppliers

1 row | Tx Auto | DDL | CSV | Suppliers

WHERE | ORDER BY

	SupplierID	City	CompanyName	Street
1	1	Kraków	FirstCompany	Krakowska 65

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

KrzysztofSoleckiEFLab | KrzysztofSoleckiEFLab.Program | Main(string[] args)

```
namespace KrzysztofSoleckiEFLab
{
    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();
            }
        }
    }
}
```

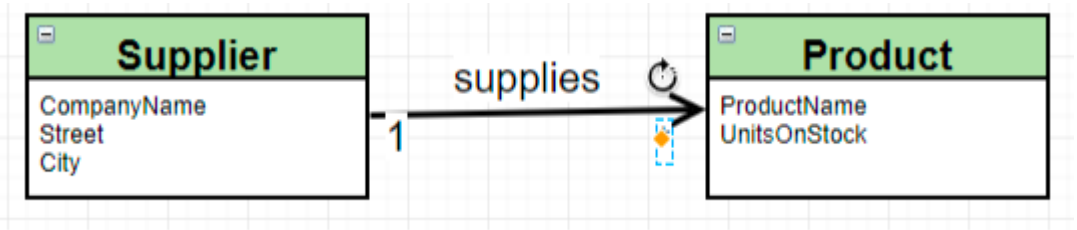
Products

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. Odwrócenie relacji:



Klasa ProductContext nie zmieniła się względem poprzedniego podpunktu.

- Klasa Product:

```
8   namespace KrzysztofSoleckiEFLab
9   {
        Odwołania: 9
10      public class Product
11      {
            Odwołania: 0
12          public int ProductID {  get; set; }
            Odwołania: 6
13          public string ProductName { get; set; }
            Odwołania: 6
14          public int UnitsOnStock { get; set; }
15
16          [ForeignKey("Supplier")]
            Odwołania: 0
17          public int SupplierID { get; set; }
18
19
            Odwołania: 0
20          public override string ToString()
21          {
22              return $"{ProductName}, units on stock: {UnitsOnStock}";
23          }
24
25      }
26  }
27
```

- Klasa Supplier:

```
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace KrzysztofSoleckiEFLab
8   {
        Odwołania: 2
9       public class Supplier
10      {
            Odwołania: 0
11          public int SupplierID { get; set; }
            1 odwołanie
12          public string CompanyName { get; set; }
            Odwołania: 0
13          public string Street { get; set; }
            Odwołania: 0
14          public string City { get; set; }
15
            Odwołania: 0
16          public ICollection<Product> Products { get; set; }
17
            Odwołania: 0
18          public override string ToString()
19          {
20              return CompanyName;
21          }
22      }
23  }
24
```

Migracja i update bazy danych:

```
C:\WINDOWS\system32\cmd.exe
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.5.3
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************
Nie można odnaleźć określonego pliku.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab>cd KrzysztofSoleckiEFLab

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add ProductOneToManySu
ppliersMigration
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Done. To undo this action, use 'ef migrations remove'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418215510_ProductOneToManySuppliersMigration'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>
```

A. Dodanie kilku produktów:
B. Dodanie ich do produktów dostarczanych przez nowo utworzonego dostawcę:

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

            string companyName = "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= companyName,
                Street = street,
                City = city,
                Products = products
            };

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

Products [ProductsDatabase] | Suppliers [ProductsDatabase]

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

WHERE | ORDER BY

	SupplierID	City	CompanyName	Street
1	1	Gdańsk	NewCompany	Zamorska 160

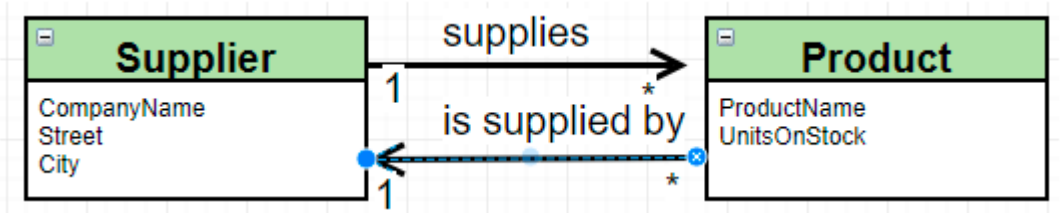
Products [ProductsDatabase] | Suppliers [ProductsDatabase]

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. Modelowanie relacji obustronnej:



- Klasa Product:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 9
    public class Product
    {
        Odwołania: 0
        public int ProductID {  get; set; }
        Odwołania: 6
        public string ProductName { get; set; }
        Odwołania: 6
        public int UnitsOnStock { get; set; }

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

        Odwołania: 0
        public override string ToString()
        {
            return $"{ProductName}, units on stock: {UnitsOnStock}";
        }

    }
}
```

- Klasa Supplier:

```
7   namespace KrzysztofSoleckiEFLab
8   {
        Odwołania: 4
9       public class Supplier
10      {
            Odwołania: 0
11          public int SupplierID { get; set; }
            Odwołania: 2
12          public string CompanyName { get; set; }
            1 odwołanie
13          public string Street { get; set; }
            1 odwołanie
14          public string City { get; set; }
15
            Odwołania: 0
16          public ICollection<Product> Products { get; set; }
17
            Odwołania: 0
18          public override string ToString()
19          {
20              return CompanyName;
21          }
22      }
23  }
24
```

Migracja i update bazy danych:

```
C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add OneToNandtoOne
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Done. To undo this action, use 'ef migrations remove'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418221106_OneToNandtoOne'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>_
```

A. Dodanie kilku produktów:
B. Dodanie ich do produktów dostarczanych przez nowo utworzonego dostawcę:

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

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

            Supplier supplier = new Supplier
            {
                CompanyName = companyName,
                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();
        }
    }
}
```

Suppliers [ProductsDatabase]

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

WHERE | ORDER BY

	SupplierID	City	CompanyName	Street
1	1	Warszawa	NewCompany	Śródmieście 160

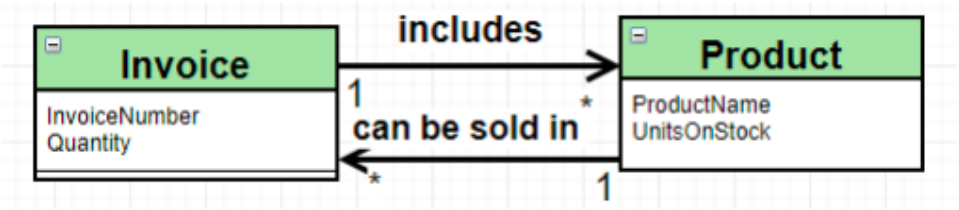
Products [ProductsDatabase]

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. Modelowanie relacji wiele do wielu:



- Klasa Product:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 11
    public class Product
    {
        Odwołania: 5
        public Product()
        {
            this.InvoiceProducts = new HashSet<InvoiceProduct>();
        }
        Odwołania: 0
        public int ProductID {  get; set; }
        Odwołania: 6
        public string ProductName { get; set; }
        Odwołania: 6
        public int UnitsOnStock { get; set; }

        1 odwołanie
        public virtual ICollection<InvoiceProduct> InvoiceProducts { get; set; }
        Odwołania: 0
        public override string ToString()
        {
            return $"{ProductName}, units on stock: {UnitsOnStock}";
        }

    }
}
```

- Klasa Invoice:

```
namespace KrzysztofSoleckiEFLab
{
    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 InvoiceProduct:

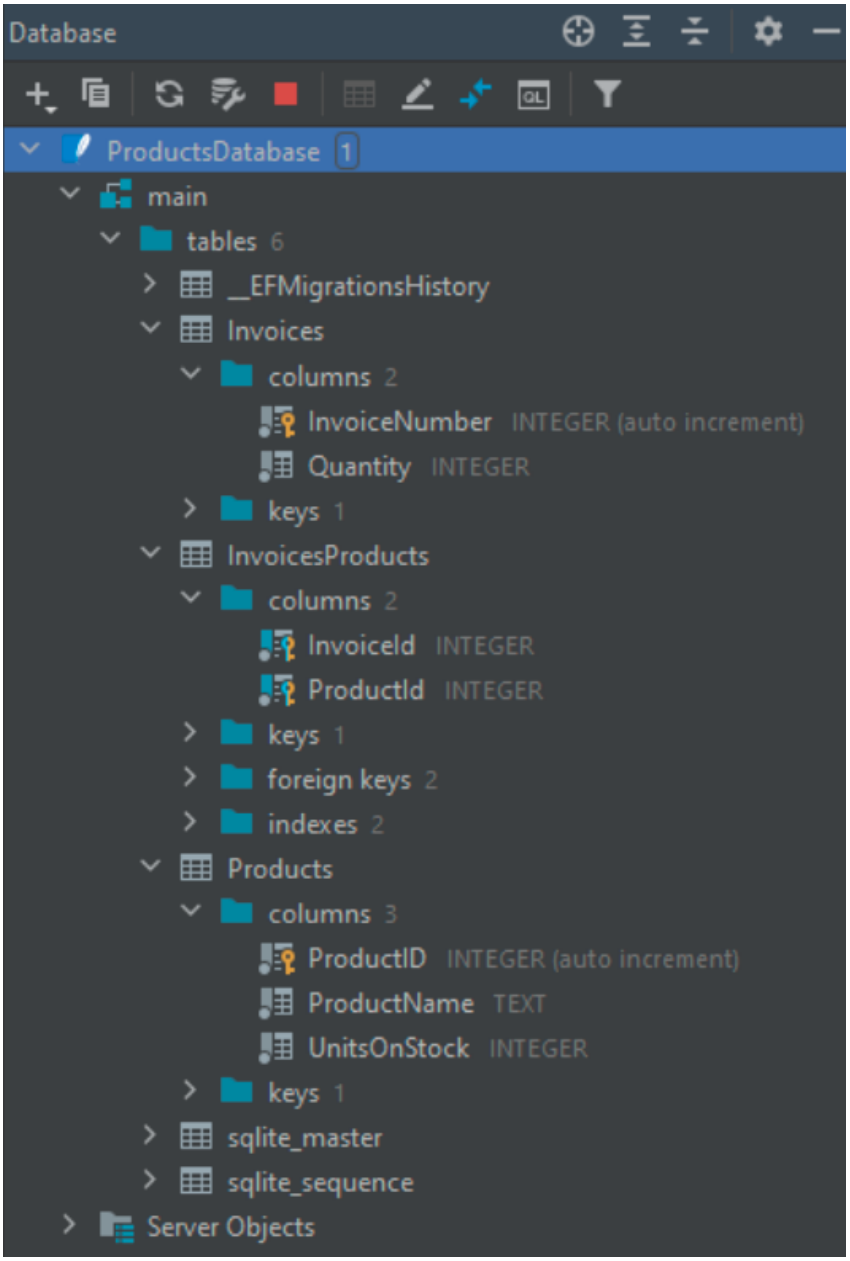
```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 6
    public class InvoiceProduct
    {
        1 odwołanie
        public int InvoiceID { get; set; }
        Odwołania: 0
        public Invoice Invoice { get; set; }
        1 odwołanie
        public int ProductId { get; set; }
        Odwołania: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

    }
}
```

- Klasa ProductContext:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 7
    public class ProductContext :DbContext
    {
        1 odwołanie
        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=ProductDatabase");
        }

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

Migracja i update bazy danych:

```
C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add ManyToMany
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
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'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418223410_ManyToMany'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>
```



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

```
1   namespace KrzysztofSoleckiEFLab
2   {
        Odwołania: 0
3       class Program
4       {
            Odwołania: 0
5           static void Main(string[] args)
6           {
7               ProductContext productContext = new ProductContext();
8
9               List<Product> products = new List<Product>();
10              products.Add(new Product { ProductName = "Myszka", UnitsOnStock = 10 });
11              products.Add(new Product { ProductName = "Monitor", UnitsOnStock = 20 });
12              products.Add(new Product { ProductName = "Słuchawki", UnitsOnStock = 30 });
13              products.Add(new Product { ProductName = "Pendrive", UnitsOnStock = 40 });
14              products.Add(new Product { ProductName = "Laptop", UnitsOnStock = 50 });
15
16              productContext.Products.AddRange(products);
17              productContext.SaveChanges();
18          }
19      }
20  }
```

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

```
namespace KrzysztofSoleckiEFLab
{
    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();
        }
    }
}
```

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

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceNumber	Quantity
1	1	5

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

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	InvoiceId	ProductId
1	1	1

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

Wykonanie transakcji każdego produktu po jednej sztuce:

```
namespace KrzysztofSoleckiEFLab
{
    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();
        }
    }
}
```

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

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

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. Produkty sprzedane w ramach faktury 2:

```
namespace KrzysztofSoleckiEFLab
{
    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 item in query )
            {
                Console.WriteLine( item.productName );
            }
        }
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
Myszka
Monitor
Słuchawki
Pendrive
Laptop
```

C. Pokaż faktury w ramach których był sprzedany wybrany produkt:

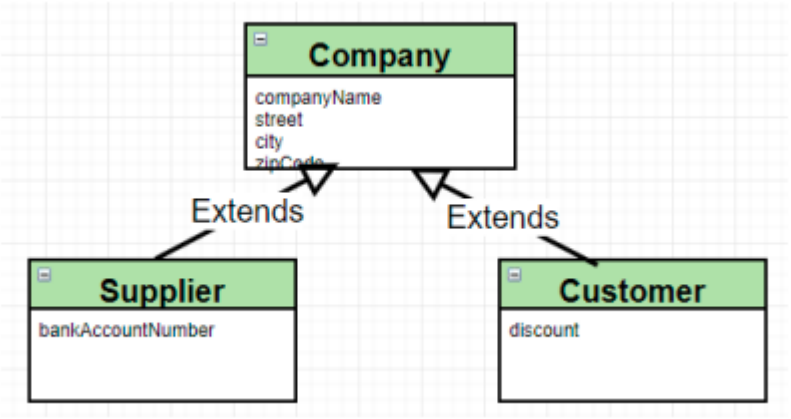
```
namespace KrzysztofSoleckiEFLab
{
    — odwołania
    class Program
    {
        — odwołania
        static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            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 item in query )
            {
                Console.WriteLine( item.invoiceId );
            }
        }
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
1
2
```

5. Dziedziczenie:

A. Wprowadź do modelu poniższą hierarchie dziedziczenia używając startegii Table-PerHierarchy:



Klasa Customer:

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

Klasa Company:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 2
    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 Supplier:

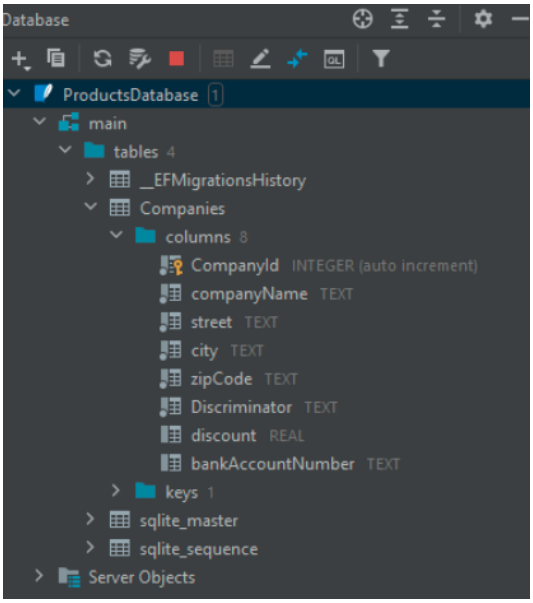
```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 0
    public class Supplier:Company
    {
        Odwołania: 0
        public string bankAccountNumber {  get; set; }
    }
}
```

Klasa ProductContext:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 7
    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=ProductDatabase");
        }

    }
}
```

Migracja i update bazy danych:

```
C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add TablePerHierarchyM
igration
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
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'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418232324_TablePerHierarchyMigration'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>
```



B. Dodaj i pobierz z bazy kilka firm obu rodzajów

```
namespace KrzysztofSoleckiEFLab
{
    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

```
namespace KrzysztofSoleckiEFLab
{
    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

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>

```
namespace KrzysztofSoleckiEFLab
{
    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
```

```
namespace KrzysztofSoleckiEFLab
{
    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
```

6. Zamodeluj te samą hierarchie dziedziczenia, ale tym razem użyj strategii Table-PerType:

Klasa Customer:

```
namespace KrzysztofSoleckiEFLab
{
    [Table("Customers")]
    Odwołania: 2
    public class Customer :Company
    {
        1 odwołanie
        public float discount { get; set; }
    }
}
```

Klasa Supplier:

```
namespace KrzysztofSoleckiEFLab
{
    [Table("Suppliers")]
    Odwołania: 2
    public class Supplier:Company
    {
        1 odwołanie
        public string bankAccountNumber {  get; set; }
    }
}
```

Klasa Company:

```
namespace KrzysztofSoleckiEFLab
{
    [Table("Companies")]
    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;}
    }
}
```

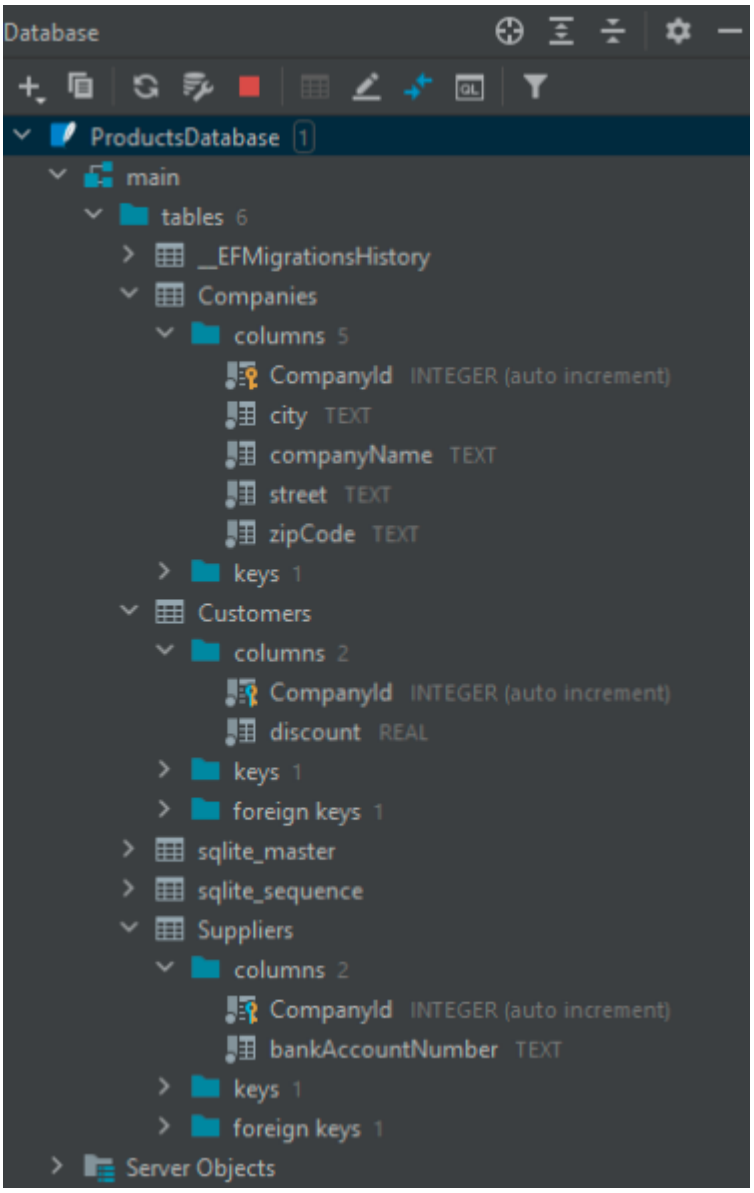
Klasa ProductContext:

```
namespace KrzysztofSoleckiEFLab
{
    Odwołania: 9
    public class ProductContext :DbContext
    {
        1 odwołanie
        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=ProductDatabase");
        }

        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 danych i update bazy danych:

```
C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef migrations add TablePerTypeMigration
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest feature
s and bug fixes. See https://aka.ms/AAc1fbw for more information.
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'

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>dotnet ef database update
Build started...
Build succeeded.
The Entity Framework tools version '7.0.4' is older than that of the runtime '7.0.5'. Update the tools for the latest feature
s and bug fixes. See https://aka.ms/AAc1fbw for more information.
Applying migration '20230418234457_TablePerTypeMigration'.
Done.

C:\Users\Lenovo\source\repos\new\KrzysztofSoleckiEFLab\KrzysztofSoleckiEFLab>
```

Struktura bazy danych:



A. Dodaj i pobierz z bazy kilka firm obu rodzajów:

```
namespace KrzysztofSoleckiEFLab
{
    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]] | 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

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

```
namespace KrzysztofSoleckiEFLab
{
    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

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

Customers [ProductsDatabase [2]]

2 rows

WHERE

	CompanyId	discount
1	7	0.05
2	8	0.1

```
namespace KrzysztofSoleckiEFLab
{
    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
```

```
namespace KrzysztofSoleckiEFLab
{
    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.Suppliers
                        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. Porównaj obie strategie modelowania dziedziczenia:

Table-Per-Hierarchy

- szybkie wykonywanie operacji CRUD, ponieważ wszystkie dane znajdują się w jednej tabeli.
- minimalizacja liczby tabel
- redundancja danych

- złożoność modyfikacji klas. Usuwanie/dodawanie atrybutów wymaga dodania/usunięcia kolumny z każdej z klas, gdyż wszystkie znajdują się w jednej tabeli.

Table-Per-Type

- 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.
- wadą jest szybkość wykonywania operacji CRUD, gdyż nasze zapytania bardzo często wymagają join'ów. Kolejną wadą jest duża liczba tabel w bazie danych.