Jakub Szaredko

Zadanie domowe

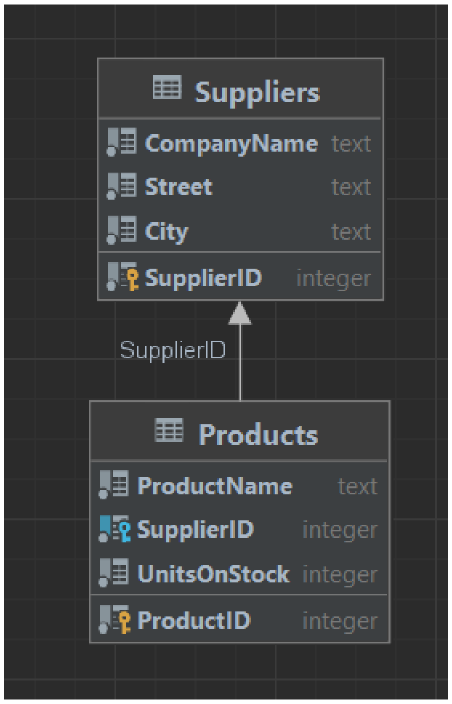
1.

a i b)

```
sqlite> SELECT * FROM Suppliers;

1|Krakow Speed|Nawojki|Krakow

sqlite> SELECT * FROM Products;

1|Strawberries|1|0
```

Dodałem dostawcę, po czym produkt, ponieważ wcześniej kopiowałem zawsze bazę danych, co za czym idzie dane nie były przechowywane.

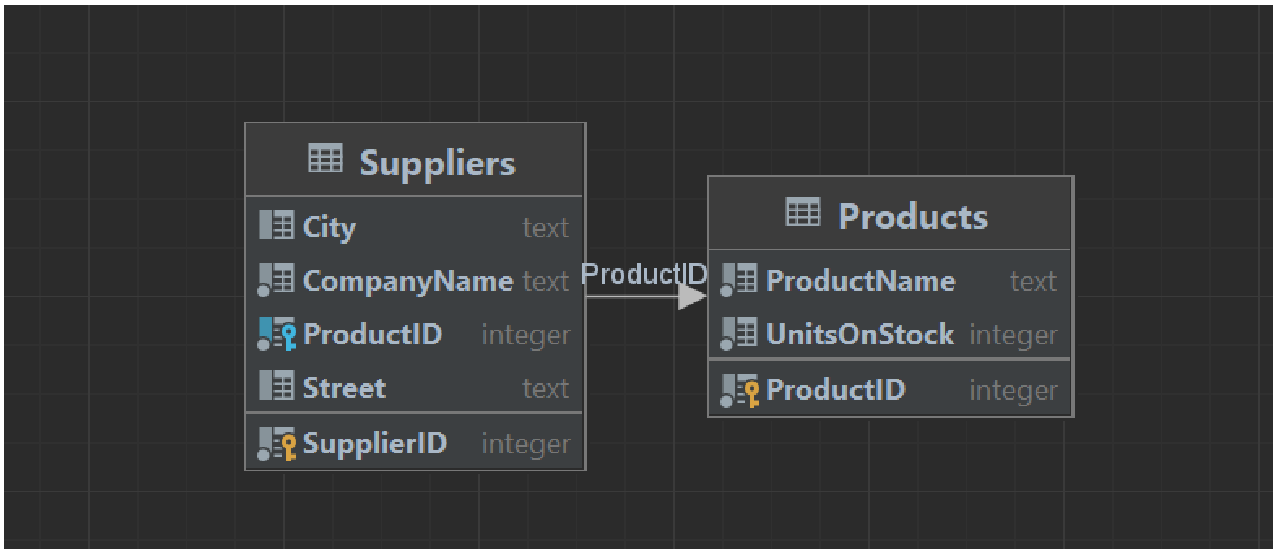
```
class Program
    {
        public static void Main(string[] args)
        {
            ProductContext productContext = new ProductContext();

            Console.WriteLine("Enter a new supplier [company name;street;city]");
            string[] supplierData = Console.ReadLine().Split(';');

            Supplier supplier = new Supplier {
                CompanyName = supplierData[0], Street = supplierData[1], City =
supplierData[2]
            };

            Console.WriteLine("Enter a new product name");
            string productName = Console.ReadLine();

            Product product = new Product { ProductName = productName };
            productContext.Products.Add(product);

            supplier.Products.Add(product);
            productContext.Suppliers.Add(supplier);

            productContext.SaveChanges();

            Console.WriteLine("\nList of all products stored in the database:");

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

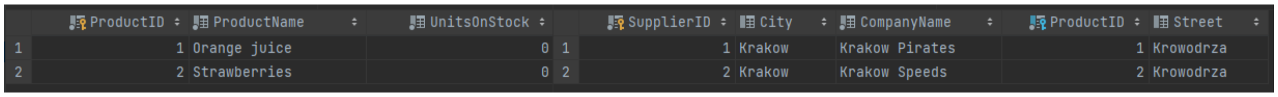

Suppliers
CompanyName text
Street text
City text
SupplierID integer
SupplierID
Products
ProductName text
SupplierID integer
UnitsOnStock integer
ProductID integer

2.

```
internal class Supplier
{
  public int SupplierID { get; set; }
  public string CompanyName { get; set; }
  public string? Street { get; set; }
  public string? City { get; set; }

  public Product? Product { get; set; }

  public Supplier()
  {

  }
    }
```

```
internal class Product
{
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public int UnitsOnStock { get; set; }
    public ICollection<Supplier> Suppliers { get; set; }

    public Product()
    {
        Suppliers = new List<Supplier>();
    }
      }

class Program
{
    public static void Main(string[] args)
    {
        ProductContext productContext = new ProductContext();

        Console.WriteLine("Enter a new supplier [company name;street;city]");
        string[] supplierData = Console.ReadLine().Split(';');

        Supplier supplier = new Supplier
        {
            CompanyName = supplierData[0],
            Street = supplierData[1],
            City = supplierData[2]
        };

        Console.WriteLine("Enter a new product name");
        string productName = Console.ReadLine();

        Product product = new Product { ProductName = productName };
        product.Suppliers.Add(supplier);
        productContext.Products.Add(product);

        supplier.Product = product;
        productContext.Suppliers.Add(supplier);

        productContext.SaveChanges();
    }
  }
```





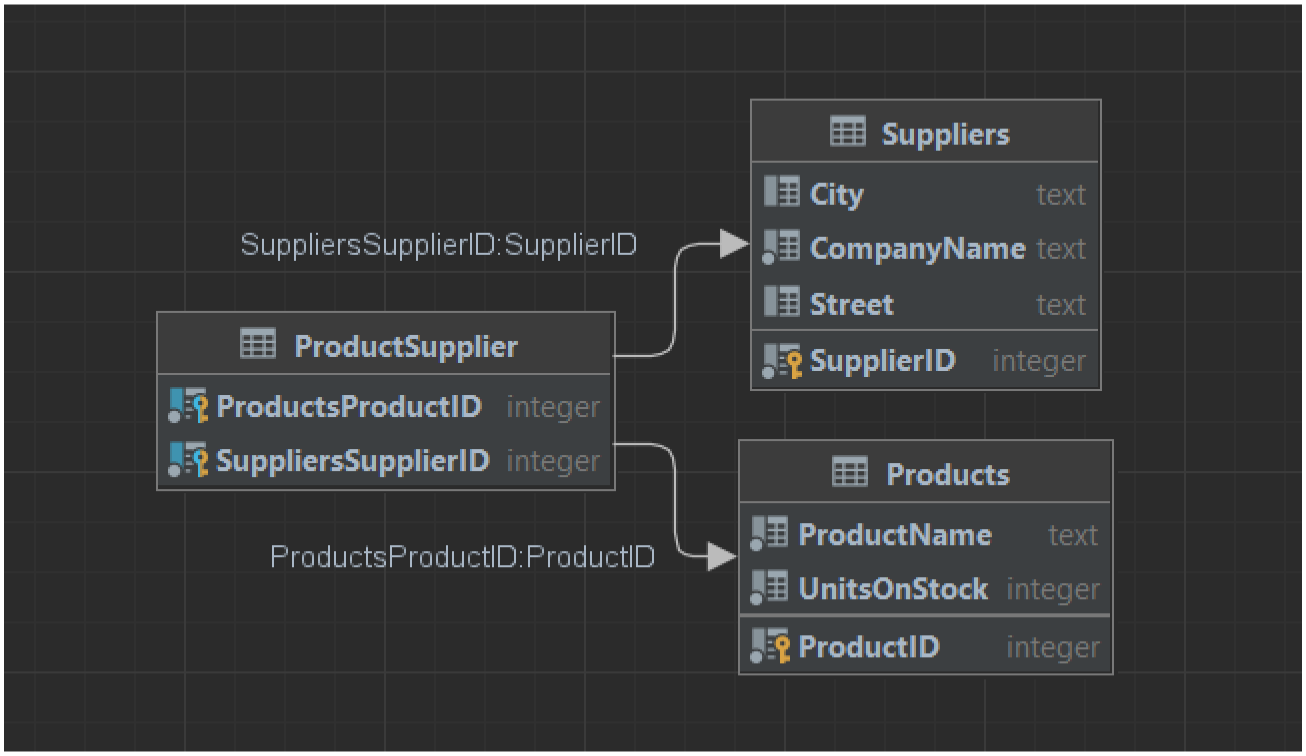
	ProductID	ProductName	UnitsOnStock
1	1	Orange juice	0
2	2	Strawberries	0

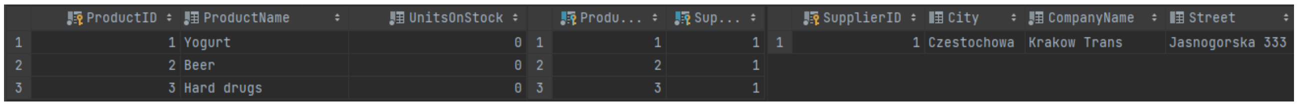
	SupplierID	City	CompanyName	ProductID	Street
1	1	Krakow	Krakow Pirates	1	Krowodrza
2	2	Krakow	Krakow Speeds	2	Krowodrza

3.

```
internal class Supplier
{
    public int SupplierID { get; set; }
    public string CompanyName { get; set; }
    public string? Street { get; set; }
    public string? City { get; set; }

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

internal class Product
{
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public int UnitsOnStock { get; set; }
    public List<Supplier> Suppliers { get; } = new();

    public Product()
    {
        ProductName = string.Empty;
    }

    public Product(string productName)
    {
        ProductName = productName;
    }
}

class Program
```

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

            List<Product> products = new();
            products.Add(new Product("Yogurt"));
            products.Add(new Product("Beer"));
            products.Add(new Product("Hard drugs"));

            Supplier supplier = new Supplier() {
                CompanyName = "Krakow Trans", City = "Czestochowa", Street = "Jasnogorska 333"
            };

            foreach (Product product in products)
            {
                supplier.Products.Add(product);
                product.Suppliers.Add(supplier);

                productContext.Products.Add(product);
            }
            productContext.Suppliers.Add(supplier);

            productContext.SaveChanges();
        }
    }
}
```





	ProductID	ProductName	UnitsOnStock
1	1	Yogurt	0
2	2	Beer	0
3	3	Hard drugs	0

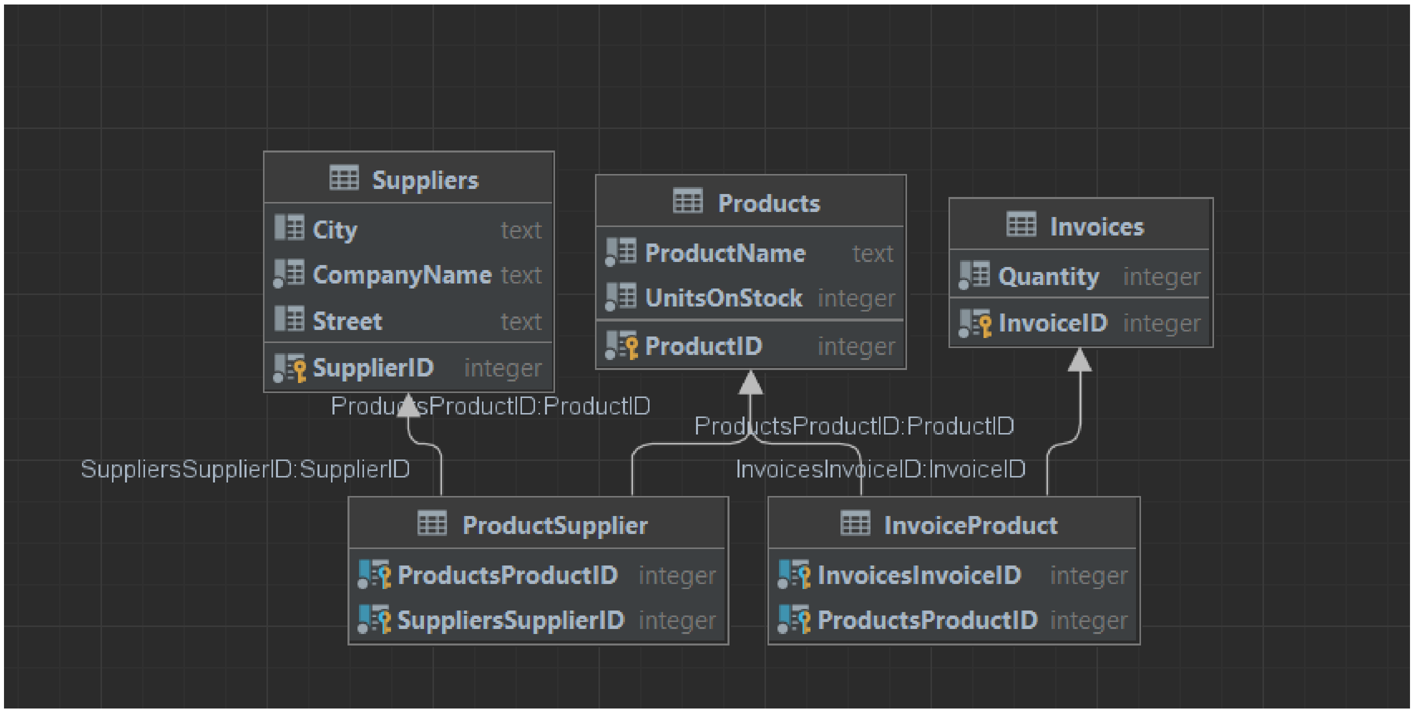
	Produ...	Sup...
1	1	1
2	2	1
3	3	1

	SupplierID	City	CompanyName	Street
1	1	Czestochowa	Krakow Trans	Jasnogorska 333

4.

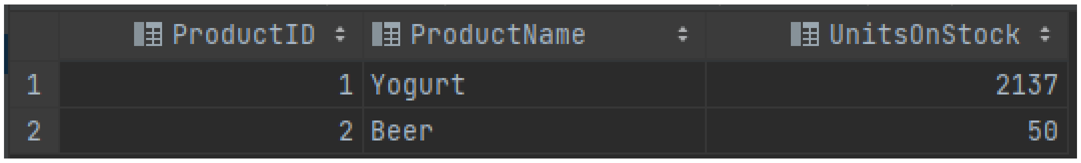
```
internal class Product
{
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public int UnitsOnStock { get; set; }
    public List<Supplier> Suppliers { get; } = new();
    public List<Invoice> Invoices { get; } = new();

    public Product()
    {
        ProductName = string.Empty;
    }

    public Product(string productName, int unitsOnStock)
    {
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
    }
}

internal class Invoice
{
    public int InvoiceID { get; set; }
    public int Quantity { get; set; }

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

    public Invoice(int quantity)
    {
        this.Quantity = quantity;
    }
}
```

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

            List<Product> products = new();
            products.Add(new Product("Yogurt", 2137));
            products.Add(new Product("Beer", 50));
            products.Add(new Product("Hard drugs", 3));

            List<Invoice> invoices = new();
            invoices.Add(new Invoice(1));
            invoices.Add(new Invoice(4));

            Supplier supplier = new Supplier() {
                CompanyName = "Krakow Trans", City = "Czestochowa", Street = "Jasnogorska 333"
            };

            products[0].Invoices.Add(invoices[0]);
            products[1].Invoices.Add(invoices[0]);
            products[1].Invoices.Add(invoices[1]);
            products[2].Invoices.Add(invoices[1]);

            invoices[0].Products.Add(products[0]);
            invoices[0].Products.Add(products[1]);
            invoices[1].Products.Add(products[1]);
            invoices[1].Products.Add(products[2]);

            foreach (Product product in products)
            {
                supplier.Products.Add(product);
                product.Suppliers.Add(supplier);

                productContext.Products.Add(product);
            }
            foreach (Invoice invoice in invoices)
            {
                productContext.Invoices.Add(invoice);
            }
            productContext.Suppliers.Add(supplier);

            productContext.SaveChanges();
        }
    }
```


Suppliers
City text
CompanyName text
Street text
SupplierID integer
Products
ProductName text
UnitsOnStock integer
ProductID integer
Invoices
Quantity integer
InvoiceID integer
ProductsProductID:ProductID
ProductsProductID:ProductID
SuppliersSupplierID:SupplierID
InvoicesInvoiceID:InvoiceID
ProductSupplier
ProductsProductID integer
SuppliersSupplierID integer
InvoiceProduct
InvoicesInvoiceID integer
ProductsProductID integer

Produkty, które są zawarte w fakturze o ID 1

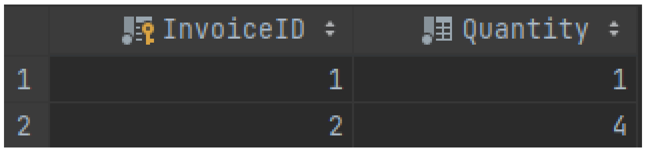
```
SELECT P.* FROM Products P
JOIN InvoiceProduct IP on P.ProductID = IP.ProductsProductID
JOIN Invoices I on IP.InvoicesInvoiceID = I.InvoiceID
WHERE I.InvoiceID = 1;
```



	ProductID	ProductName	UnitsOnStock
1	1	Yogurt	2137
2	2	Beer	50

Faktury, które zawierają produkty o ID 2

```
SELECT I.* FROM Invoices I
JOIN InvoiceProduct IP on I.InvoiceID = IP.InvoicesInvoiceID
JOIN Products P on P.ProductID = IP.ProductsProductID
WHERE P.ProductID = 2;
```

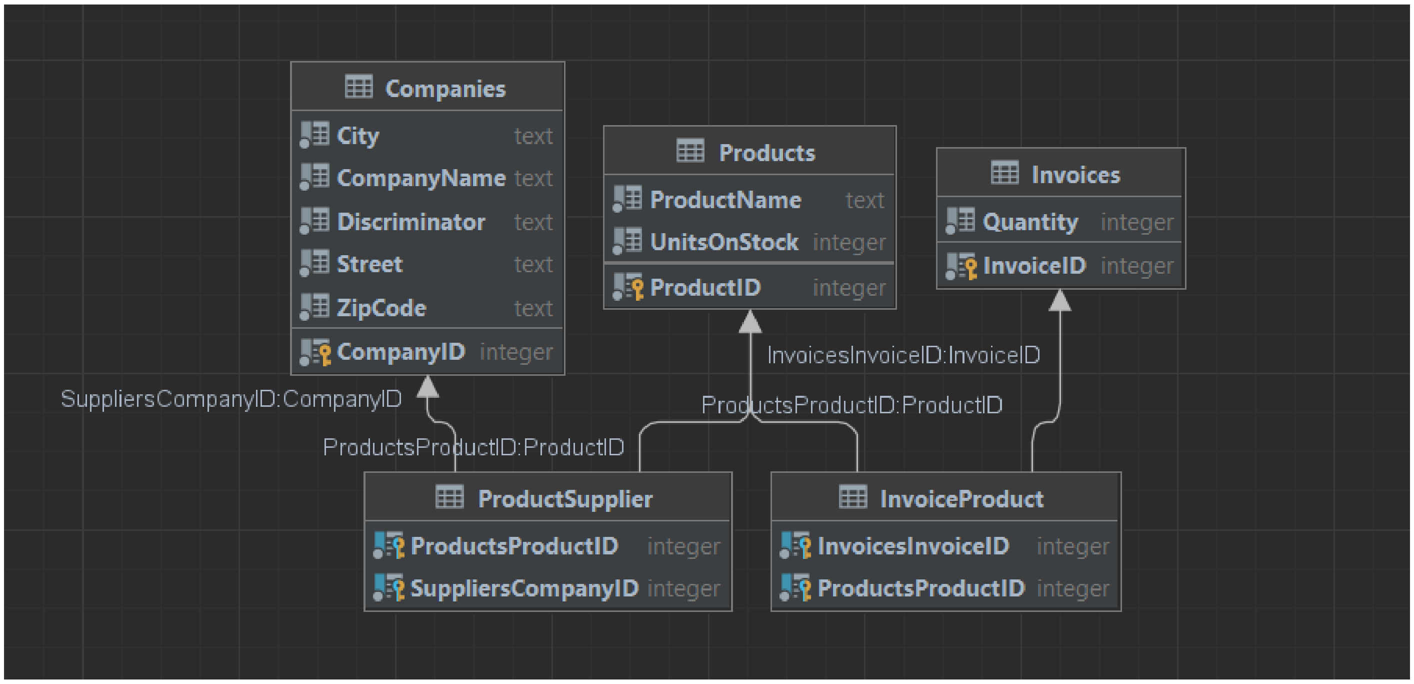


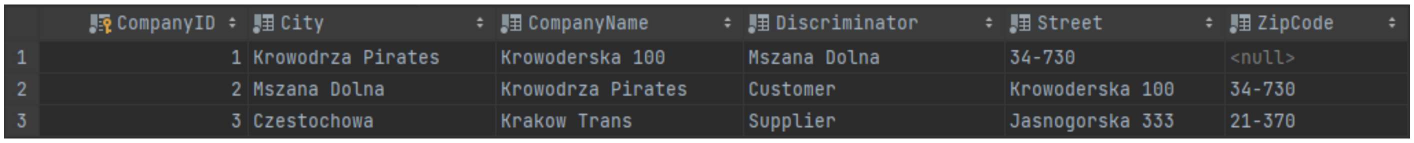
	InvoiceID	Quantity
1	1	1
2	2	4

5.

```
    internal class Company
    {
        [Key]
        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; }
    }

    internal class Customer : Company
    {
        public float Discount;
    }

    internal class Supplier : Company
    {
        public string BankAccountNumber;
        public List<Product> Products { get; } = new();
    }
```

```
class Program
{
    public static void Main(string[] args)
    {
        ProductContext productContext = new ProductContext();

        List<Product> products = new();
        products.Add(new Product("Yogurt", 2137));
        products.Add(new Product("Beer", 50));
        products.Add(new Product("Hard drugs", 3));

        List<Invoice> invoices = new();
        invoices.Add(new Invoice(1));
        invoices.Add(new Invoice(4));

        Supplier supplier = new Supplier()
        {
            CompanyName = "Krakow Trans",
            Street = "Jasnogorska 333",
            City = "Czestochowa",
            ZipCode = "21-370",
            BankAccountNumber = "000000"
        };

        Customer customer = new Customer()
        {
            CompanyName = "Krowodrza Pirates",
            Street = "Krowoderska 100",
            City = "Mszana Dolna",
            ZipCode = "34-730",
            Discount = .2f
        };

        products[0].Invoices.Add(invoices[0]);
        products[1].Invoices.Add(invoices[0]);
        products[1].Invoices.Add(invoices[1]);
        products[2].Invoices.Add(invoices[1]);

        invoices[0].Products.Add(products[0]);
        invoices[0].Products.Add(products[1]);
        invoices[1].Products.Add(products[1]);
        invoices[1].Products.Add(products[2]);

        foreach (Product product in products)
        {
            supplier.Products.Add(product);
            product.Suppliers.Add(supplier);

            productContext.Products.Add(product);
        }
        foreach (Invoice invoice in invoices)
        {
            productContext.Invoices.Add(invoice);
        }
        productContext.Suppliers.Add(supplier);
        productContext.Customers.Add(customer);

        productContext.SaveChanges();
    }
}
```



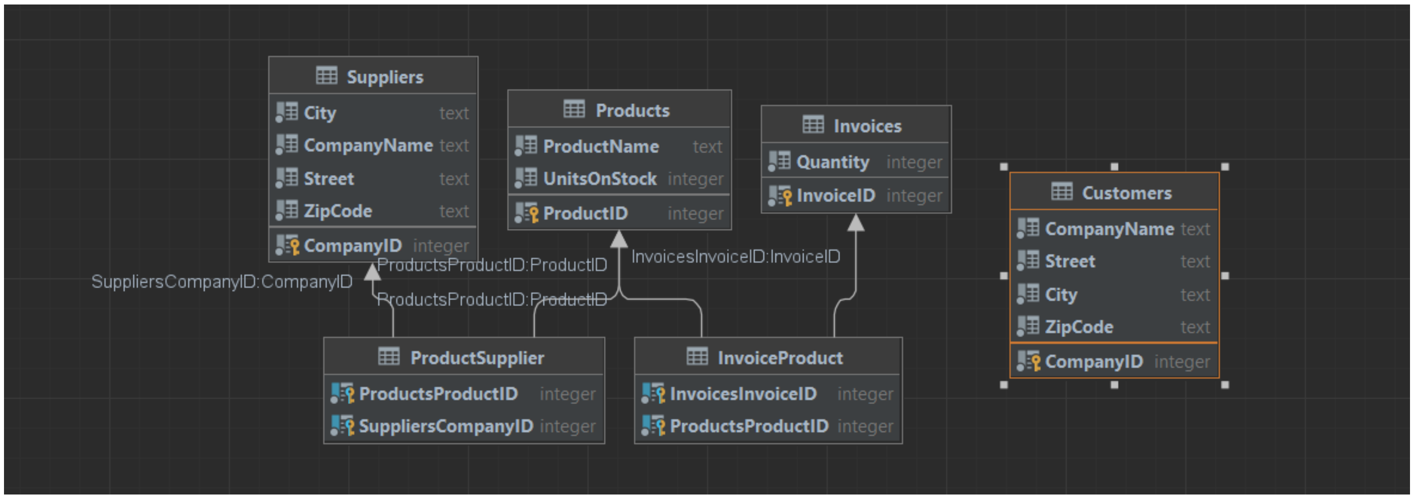


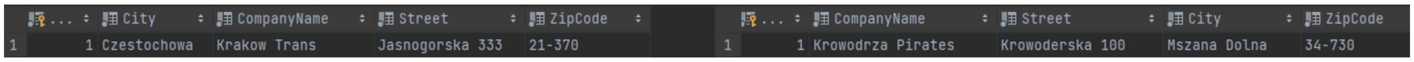
	CompanyID	City	CompanyName	Discriminator	Street	ZipCode
1	1	Krowodrza Pirates	Krowoderska 100	Mszana Dolna	34-730	<null>
2	2	Mszana Dolna	Krowodrza Pirates	Customer	Krowoderska 100	34-730
3	3	Czestochowa	Krakow Trans	Supplier	Jasnogorska 333	21-370

6.

Projekt nie zmienił się oprócz zmiany ProductsContext

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

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<Company>().UseTpcMappingStrategy();
        }

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





	...	City	CompanyName	Street	ZipCode
1	1	Czestochowa	Krakow Trans	Jasnogorska 333	21-370

	...	CompanyName	Street	City	ZipCode
1	1	Krowodrza Pirates	Krowoderska 100	Mszana Dolna	34-730

Różnica między Table per Hierarchy a Table per Type jest taka, że przy pierwszej opcji tworzona jest jedna wspólna tabela dla wszystkich klas dziedziczącej po klasie bazowej.