B2L3

Entity Framework Core

Lärandemål EF Core (ORM)

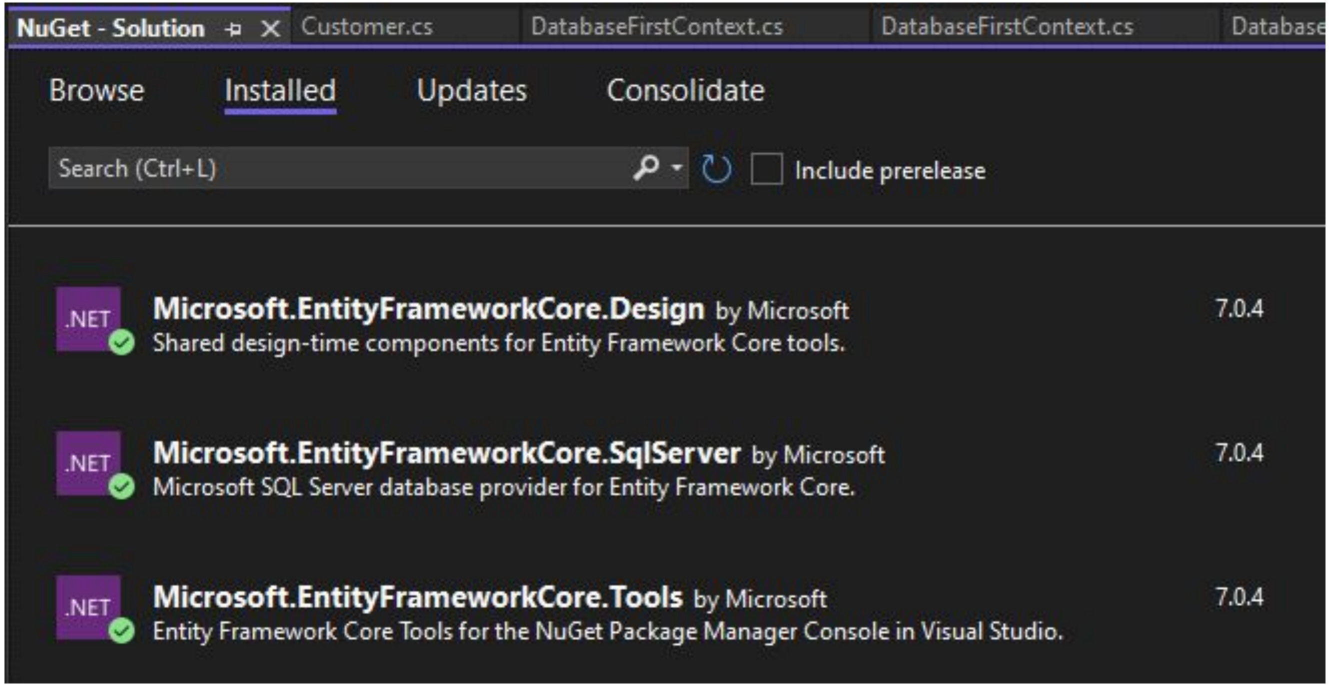
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with many databases, including SQL Database (on-premises and Azure), SQLite, MySQL, PostgreSQL, and Azure Cosmos DB.

- Grundläggande förståelse för EF Core arkitekturen
- Kunna “reverse engineer” en befintlig databas (database first)
- Skapa Migration filer i PMC med PowerShell
- Skapa och anropa enklare stored procedures
- Skriva SQL queries direkt i koden samt kännedom om risker med det!
- Förståelse för hur Data Annotationer och SQL Constraints förhåller sig till varandra
- Kunna Skapa en databas (code first)

NuGet - Solution

För att använda EF Core, utföra CRUD operationer mot databasen, Reverse Engineer databasen etc behöver vi installera en rad paket i vårt projekt.

Vi ska nu övergripande kika på vad de gör för att bättre hänga med i koden…



Microsoft.EntityFrameworkCore.Design by Microsoft 7.0.4
Shared design-time components for Entity Framework Core tools.

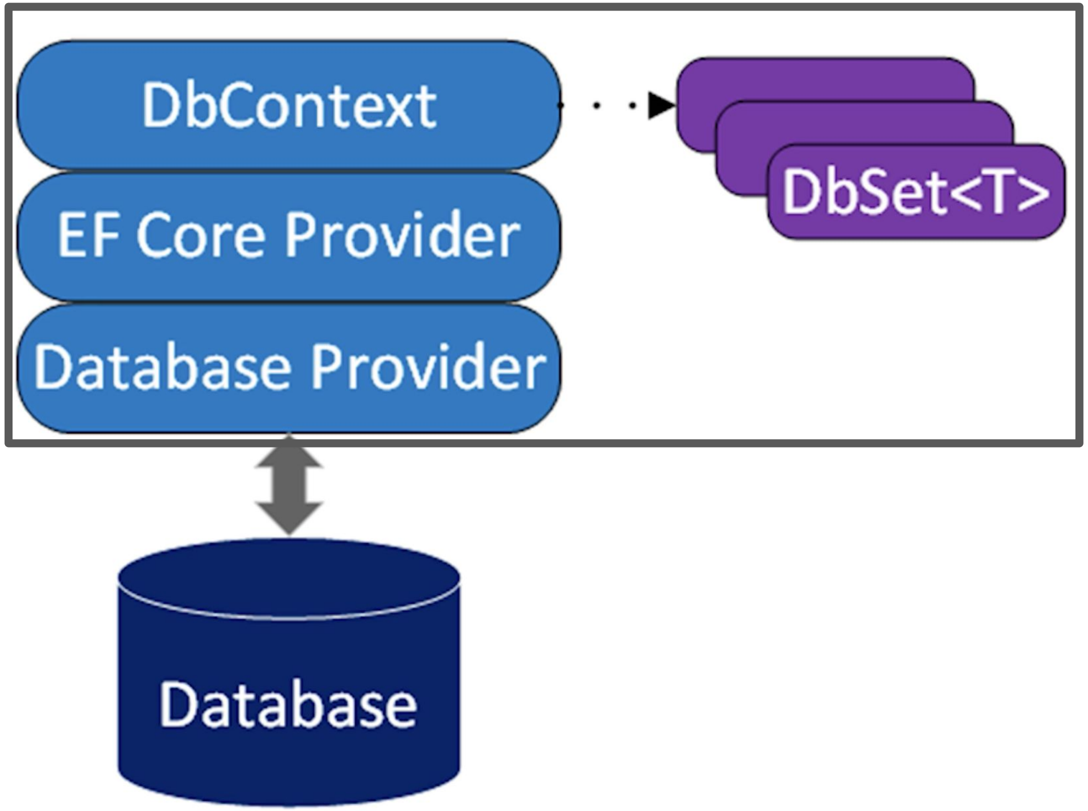
Microsoft.EntityFrameworkCore.SqlServer by Microsoft 7.0.4
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by Microsoft 7.0.4
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Database + Application

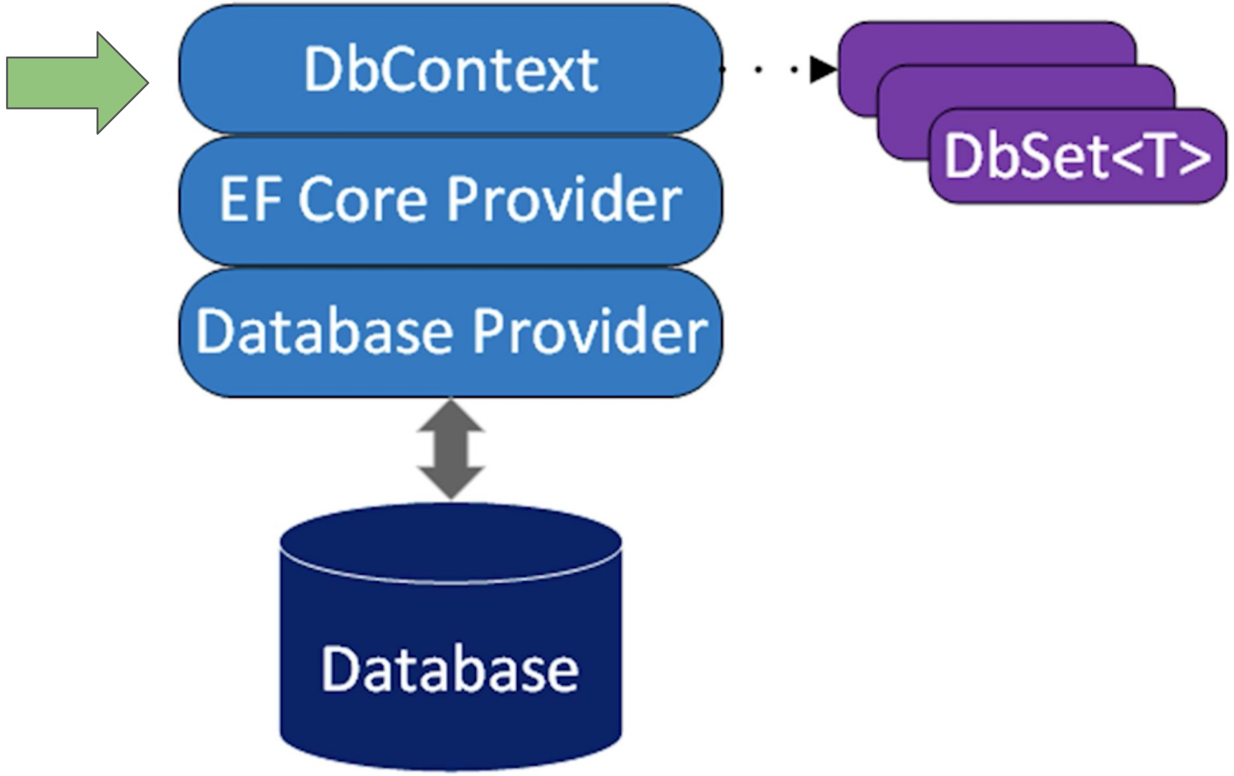
- **MicrosoftEntityFramworkCore.Design**
  - “Design-time” (utveckligsfasen) logic för scaffolding/generera kod
- ***Microsoft.EntityFrameworkCore.SQLServer***
  - *Database Provider (se kommande slides)*
- ***Microsoft.EntityFrameworkCore.Tools***
  - *Möjliggör för kommandon som Scaffold-DbContext, Add-Migration, Update-Database*

EF Core Application



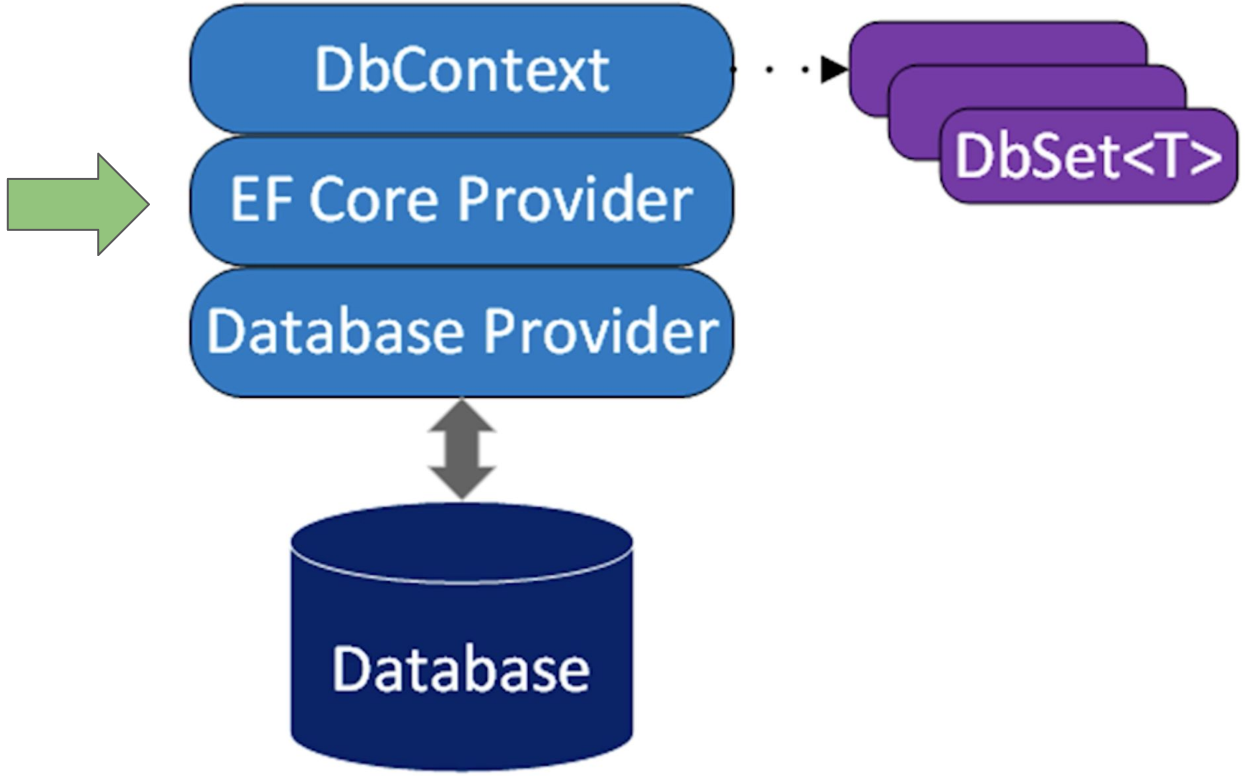
DbContext

- En instans av DbContext representerar en session med databasen
- Lifetime knuten till unit-of-work (en “CRUD operation”)
- har en ChangeTracker som bevakar “state changes” för dina entity instanser (*tänk git snapshot*)
- DbSet<T> representerar tabellerna i databasen



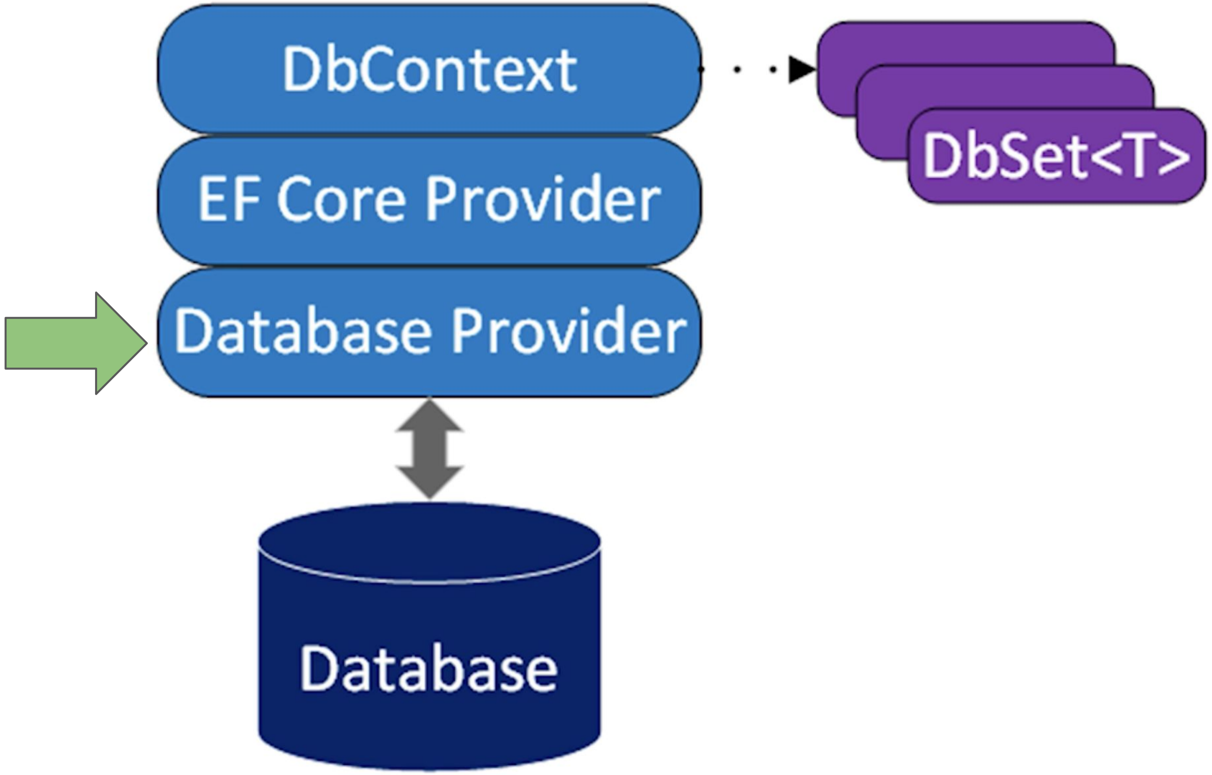
EF Core Provider

- EF Core Provider översätter “object graph changes” till SQL som vår database provider sedan kan köra.



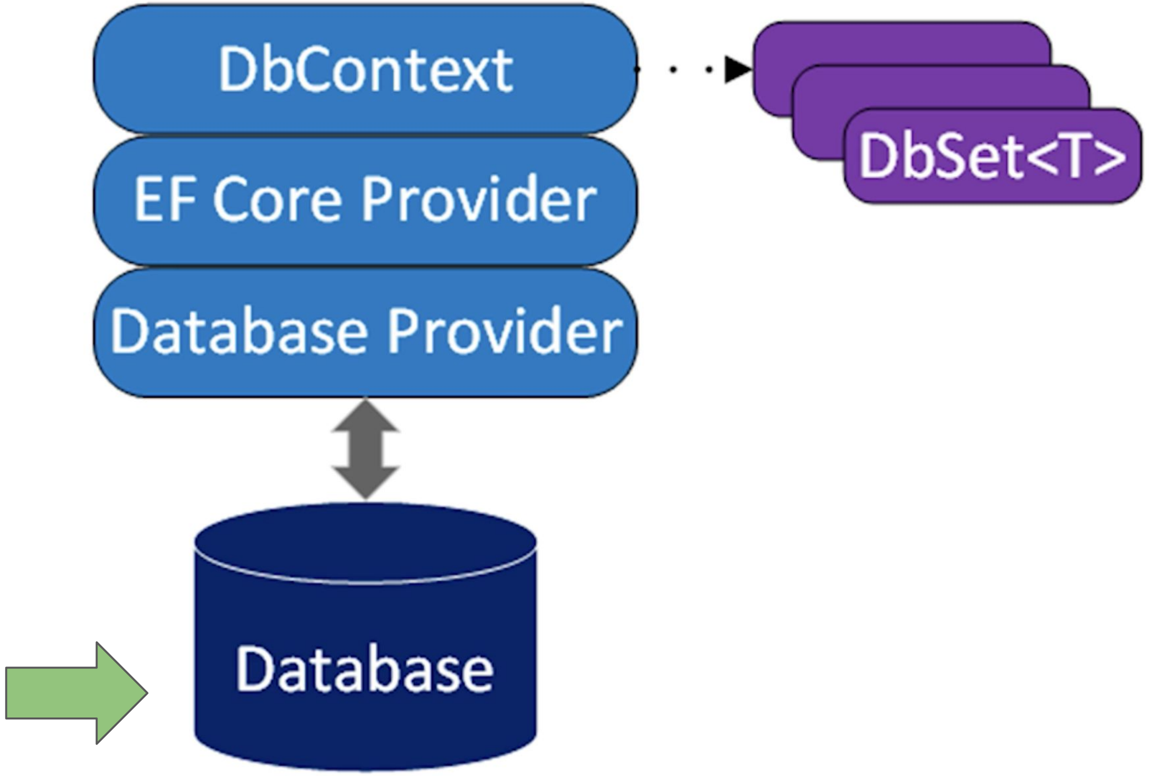
Database Provider

- Brygga mellan app och DB
- Ansvarar för kommunikation mellan EF Core och DB
- Utför SQL (DML *tänk CRUD*) operationer baserat på ChangeTracker state när vi anropar SaveChanges().
- Finns många DB providers utöver SQL Server, ex: *PostgreSQL, MySQL etc…*
- Förse projektet med en DB provider genom NuGet - Solution Eller Package Manager Console: *Install-Package <provider_package_name>*

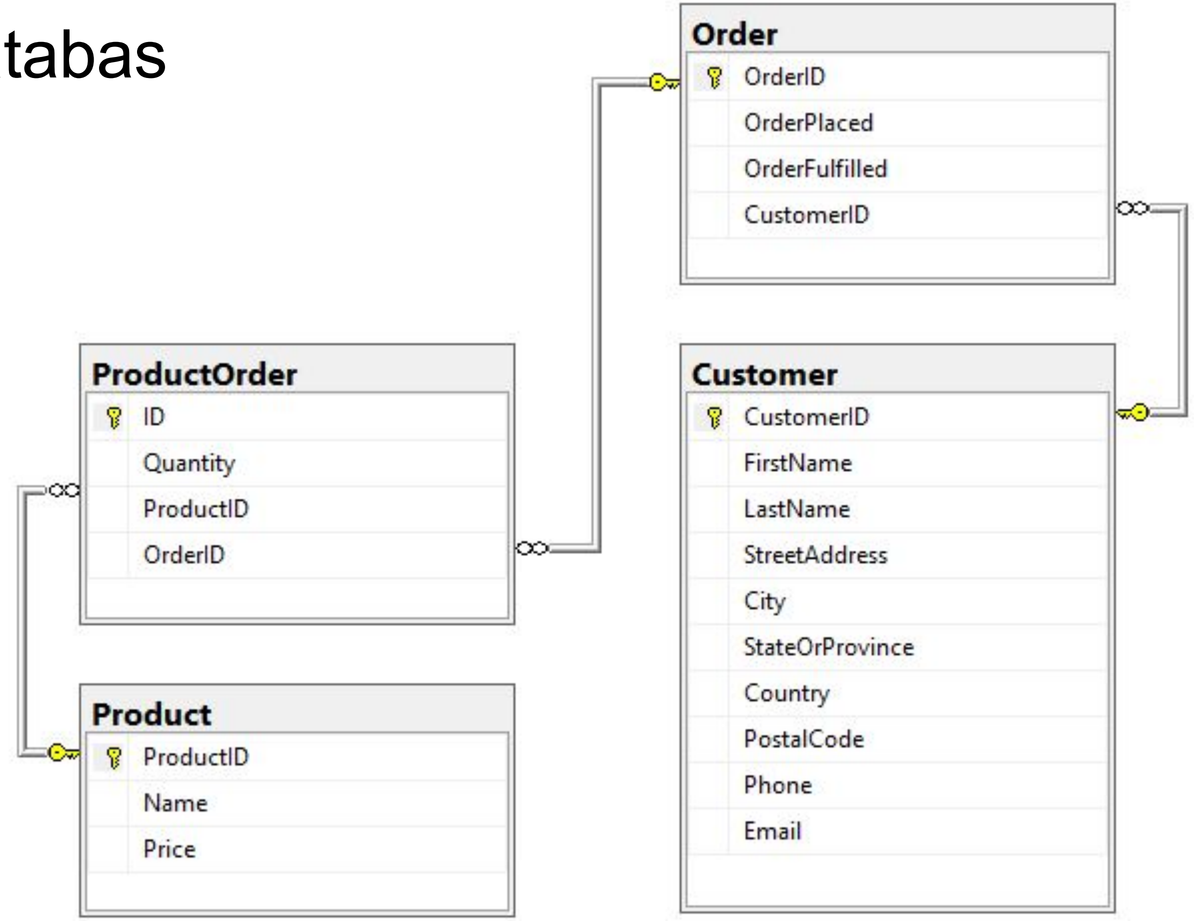


Database

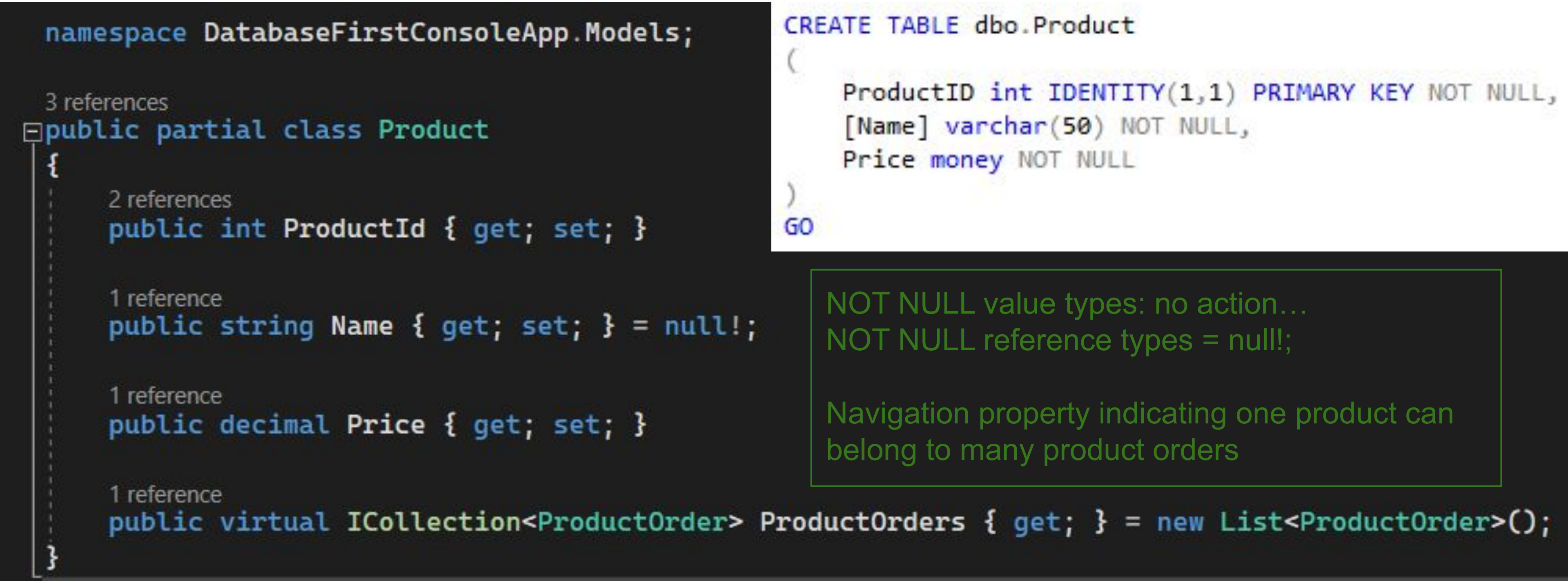
- Befintlig (Database First) eller databas vi ämnar skapa (Code First)
- Val av (DB) avgör direkt val av DB Provider
- Database Driver (tänk adapter/api) möjliggör för kommunikation mellan **Database Provider** och RDBMS/database
- anslutning upprättas via connection string)



Vår exempeldatabas



NOT NULL (Required) value type vs reference type



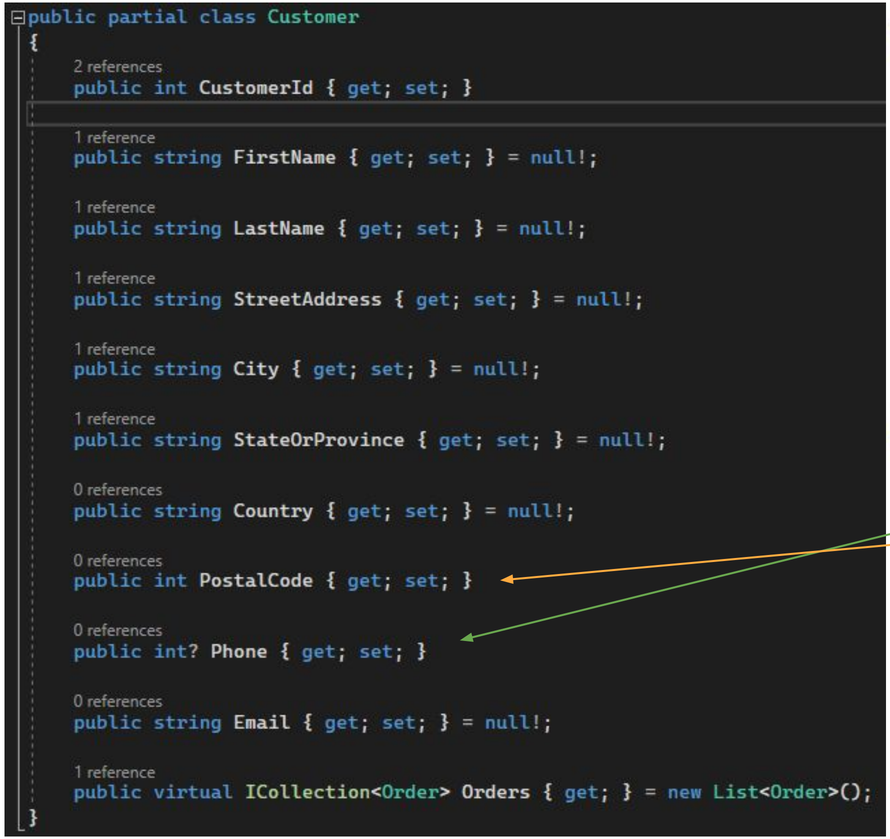
```
namespace DatabaseFirstConsoleApp.Models;

3 references
public partial class Product
{
    2 references
    public int ProductId { get; set; }

    1 reference
    public string Name { get; set; } = null!;

    1 reference
    public decimal Price { get; set; }

    1 reference
    public virtual ICollection<ProductOrder> ProductOrders { get; } = new List<ProductOrder>();
}
```

```
CREATE TABLE dbo.Product
(
    ProductID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    [Name] varchar(50) NOT NULL,
    Price money NOT NULL
)
GO
```

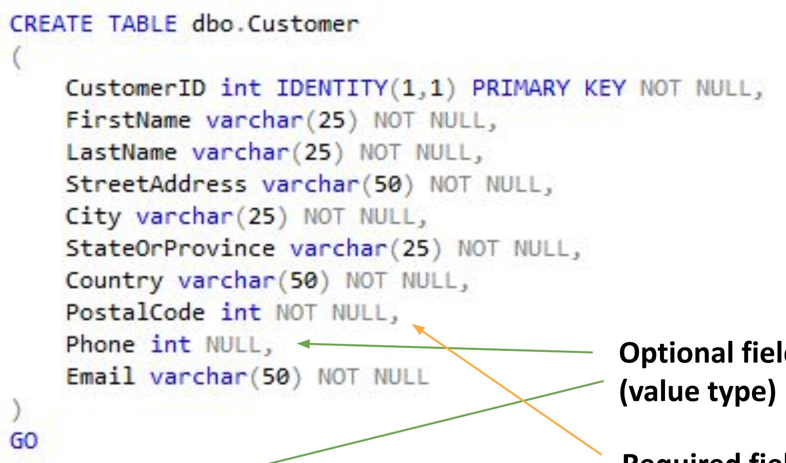
NOT NULL value types: no action…
NOT NULL reference types = null!;

Navigation property indicating one product can belong to many product orders

Optional vs Required value type fields



```
public partial class Customer
{
    2 references
    public int CustomerId { get; set; }

    1 reference
    public string FirstName { get; set; } = null!;

    1 reference
    public string LastName { get; set; } = null!;

    1 reference
    public string StreetAddress { get; set; } = null!;

    1 reference
    public string City { get; set; } = null!;

    1 reference
    public string StateOrProvince { get; set; } = null!;

    0 references
    public string Country { get; set; } = null!;

    0 references
    public int PostalCode { get; set; }

    0 references
    public int? Phone { get; set; }

    0 references
    public string Email { get; set; } = null!;

    1 reference
    public virtual ICollection<Order> Orders { get; } = new List<Order>();
}
```

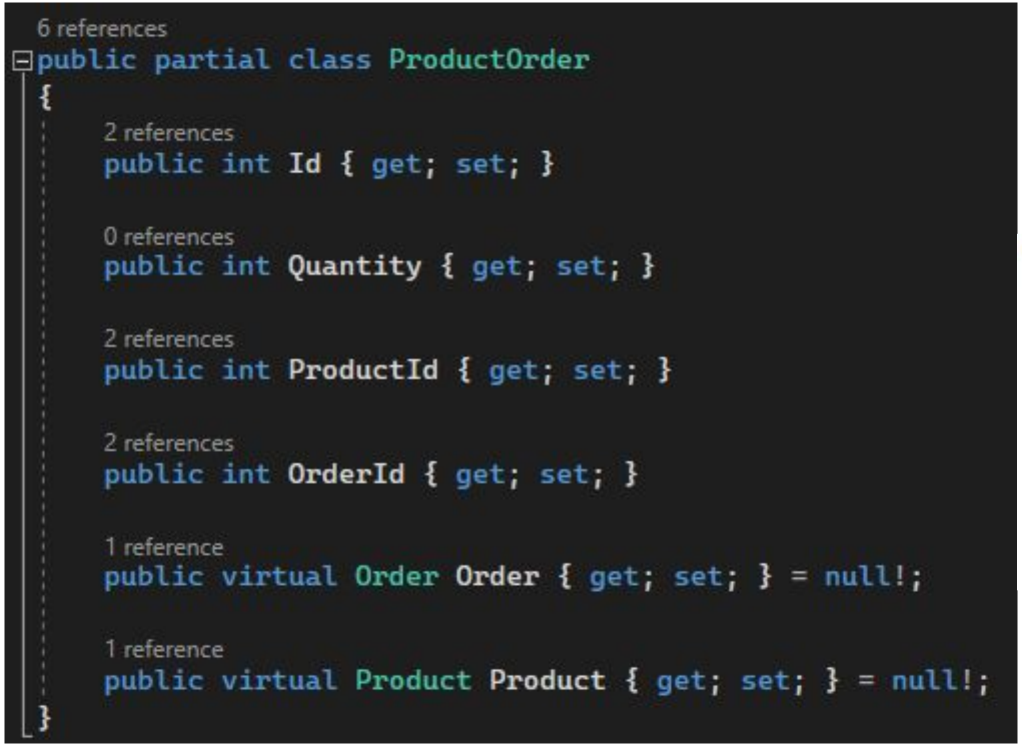


```
CREATE TABLE dbo.Customer
(
    CustomerID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    FirstName varchar(25) NOT NULL,
    LastName varchar(25) NOT NULL,
    StreetAddress varchar(50) NOT NULL,
    City varchar(25) NOT NULL,
    StateOrProvince varchar(25) NOT NULL,
    Country varchar(50) NOT NULL,
    PostalCode int NOT NULL,
    Phone int NULL,
    Email varchar(50) NOT NULL
)
GO
```

Optional field (value type)

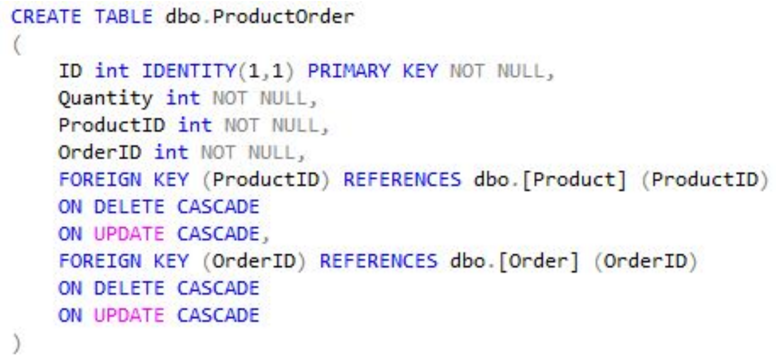
Required field (value type)

Collections (ref type) are initialised by default to avoid NullReferenceException when adding an Order instance…



```
6 references
public partial class ProductOrder
{
    2 references
    public int Id { get; set; }

    0 references
    public int Quantity { get; set; }

    2 references
    public int ProductId { get; set; }

    2 references
    public int OrderId { get; set; }

    1 reference
    public virtual Order Order { get; set; } = null!;

    1 reference
    public virtual Product Product { get; set; } = null!;
}
```

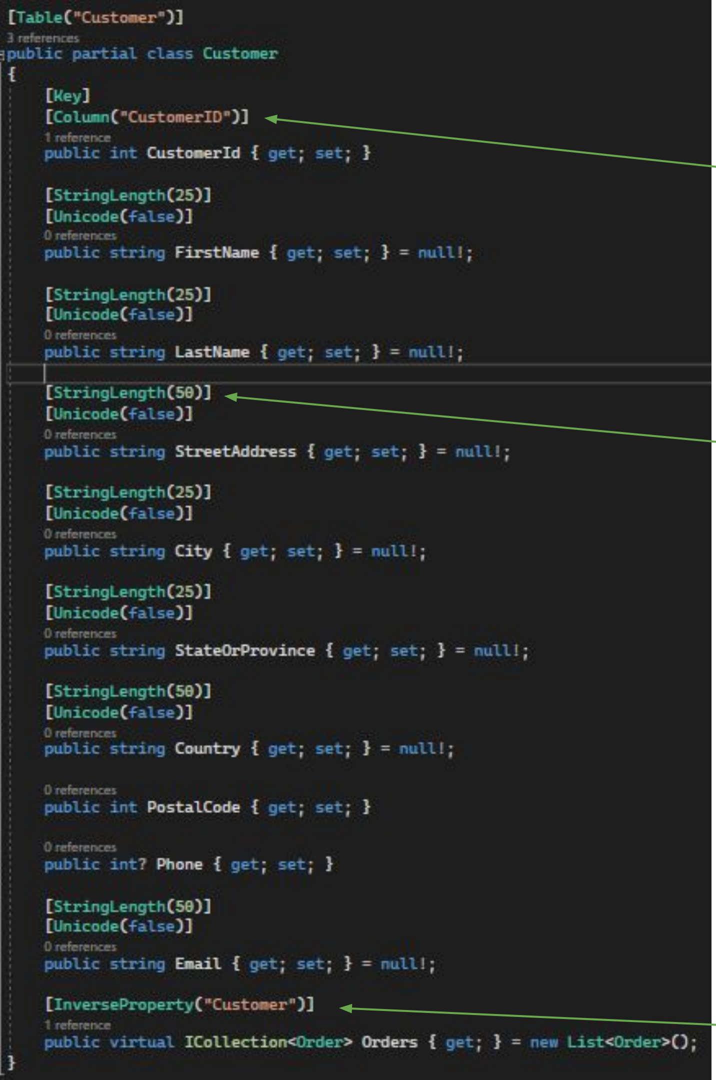
Kan ni gissa vilka fält som är obligatoriska/NOT NULL/Required?



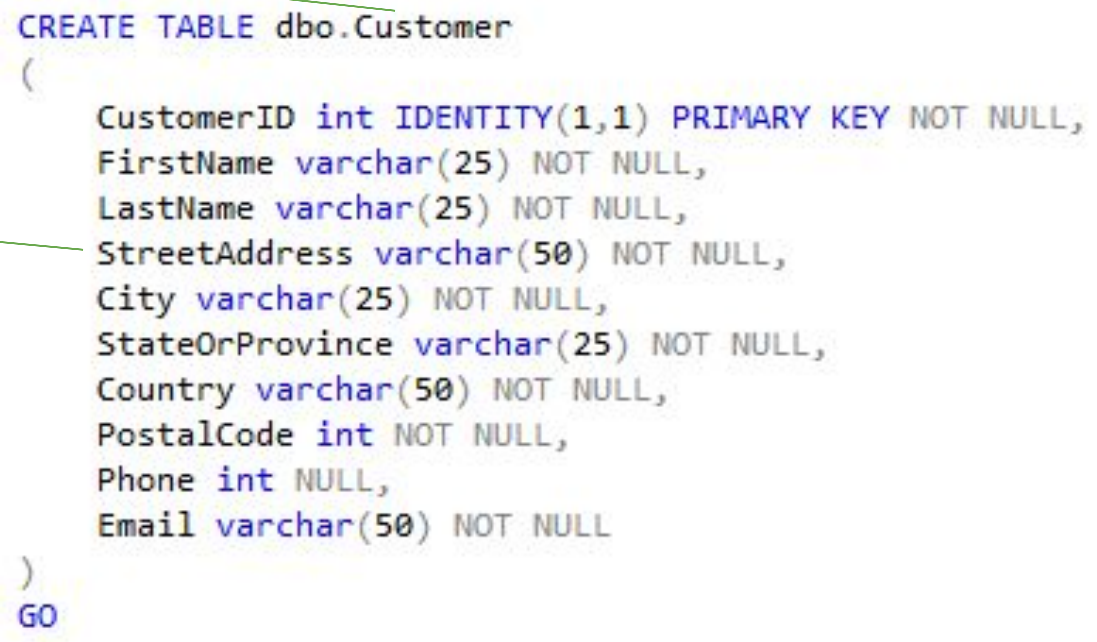
```
CREATE TABLE dbo.ProductOrder
(
    ID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    Quantity int NOT NULL,
    ProductID int NOT NULL,
    OrderID int NOT NULL,
    FOREIGN KEY (ProductID) REFERENCES dbo.[Product] (ProductID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (OrderID) REFERENCES dbo.[Order] (OrderID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
```

Data Annotations

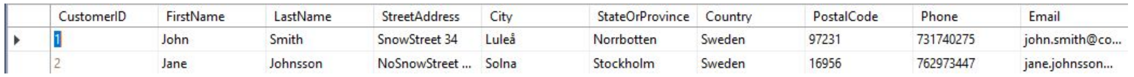
C# Attributes



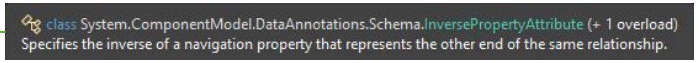
```
[Table("Customer")]
3 references
public partial class Customer
{
    [Key]
    [Column("CustomerID")]
    1 reference
    public int CustomerId { get; set; }

    [StringLength(25)]
    [Unicode(false)]
    0 references
    public string FirstName { get; set; } = null!;

    [StringLength(25)]
    [Unicode(false)]
    0 references
    public string LastName { get; set; } = null!;

    [StringLength(50)]
    [Unicode(false)]
    0 references
    public string StreetAddress { get; set; } = null!;

    [StringLength(25)]
    [Unicode(false)]
    0 references
    public string City { get; set; } = null!;

    [StringLength(25)]
    [Unicode(false)]
    0 references
    public string StateOrProvince { get; set; } = null!;

    [StringLength(50)]
    [Unicode(false)]
    0 references
    public string Country { get; set; } = null!;

    0 references
    public int PostalCode { get; set; }

    0 references
    public int? Phone { get; set; }

    [StringLength(50)]
    [Unicode(false)]
    0 references
    public string Email { get; set; } = null!;

    [InverseProperty("Customer")]
    1 reference
    public virtual ICollection<Order> Orders { get; } = new List<Order>();
}
```



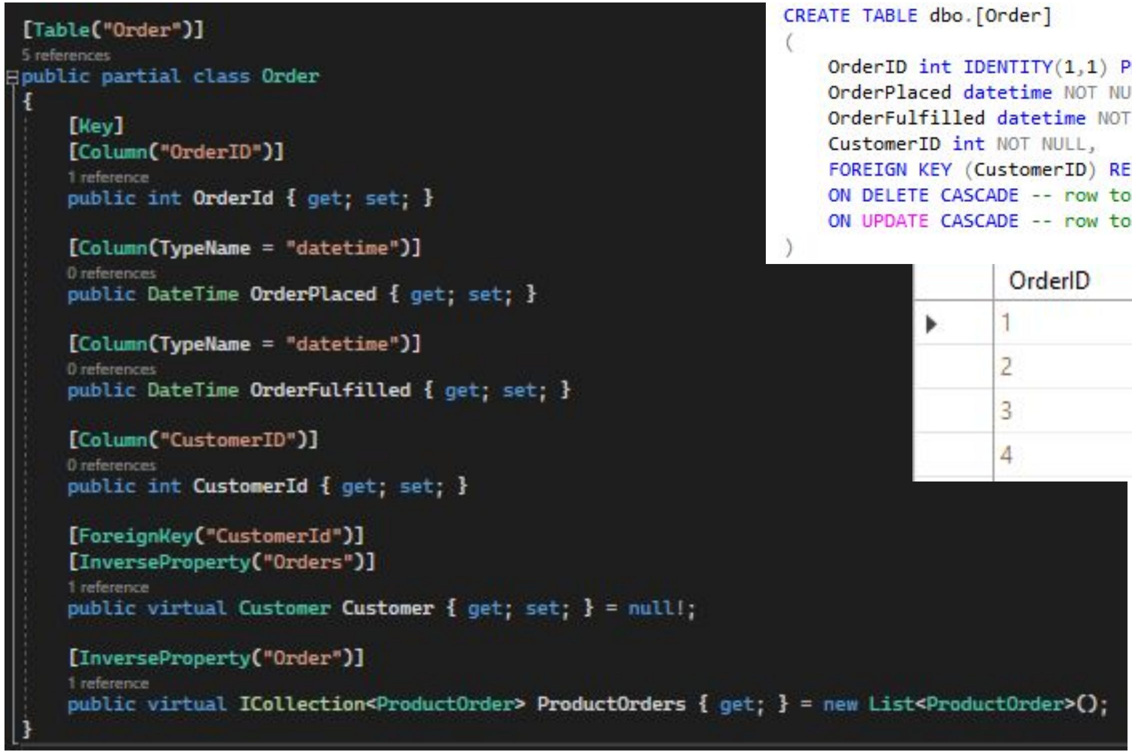
```
CREATE TABLE dbo.Customer
(
    CustomerID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    FirstName varchar(25) NOT NULL,
    LastName varchar(25) NOT NULL,
    StreetAddress varchar(50) NOT NULL,
    City varchar(25) NOT NULL,
    StateOrProvince varchar(25) NOT NULL,
    Country varchar(50) NOT NULL,
    PostalCode int NOT NULL,
    Phone int NULL,
    Email varchar(50) NOT NULL
)
GO
```



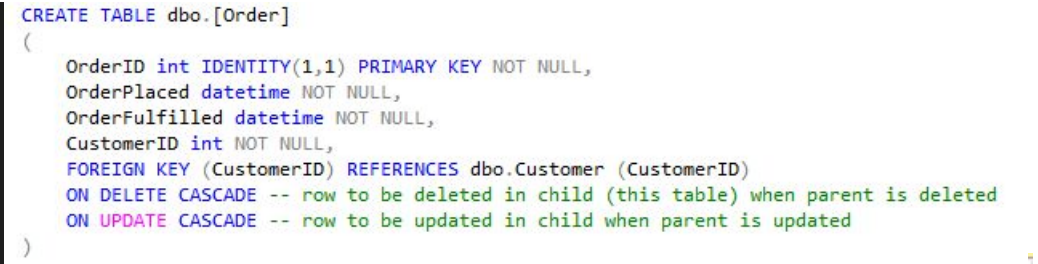
CustomerID	FirstName	LastName	StreetAddress	City	StateOrProvince	Country	PostalCode	Phone	Email
1	John	Smith	SnowStreet 34	Luleå	Norrbotten	Sweden	97231	731740275	john.smith@co...
2	Jane	Johnsson	NoSnowStreet ...	Solna	Stockholm	Sweden	16956	762973447	jane.johnsson...



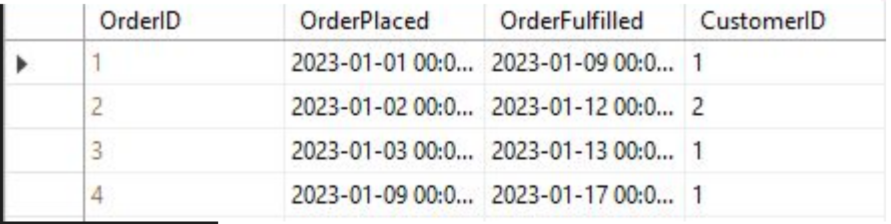
class System.ComponentModel.DataAnnotations.Schema.InversePropertyAttribute (+ 1 overload)
Specifies the inverse of a navigation property that represents the other end of the same relationship.



```
[Table("Order")]
5 references
public partial class Order
{
    [Key]
    [Column("OrderID")]
    1 reference
    public int OrderId { get; set; }

    [Column(TypeName = "datetime")]
    0 references
    public DateTime OrderPlaced { get; set; }

    [Column(TypeName = "datetime")]
    0 references
    public DateTime OrderFulfilled { get; set; }

    [Column("CustomerID")]
    0 references
    public int CustomerId { get; set; }

    [ForeignKey("CustomerId")]
    [InverseProperty("Orders")]
    1 reference
    public virtual Customer Customer { get; set; } = null!;

    [InverseProperty("Order")]
    1 reference
    public virtual ICollection<ProductOrder> ProductOrders { get; } = new List<ProductOrder>();
}
```



```
CREATE TABLE dbo.[Order]
(
    OrderID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    OrderPlaced datetime NOT NULL,
    OrderFulfilled datetime NOT NULL,
    CustomerID int NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES dbo.Customer (CustomerID)
    ON DELETE CASCADE -- row to be deleted in child (this table) when parent is deleted
    ON UPDATE CASCADE -- row to be updated in child when parent is updated
)
```



	OrderID	OrderPlaced	OrderFulfilled	CustomerID
▶	1	2023-01-01 00:0...	2023-01-09 00:0...	1
	2	2023-01-02 00:0...	2023-01-12 00:0...	2
	3	2023-01-03 00:0...	2023-01-13 00:0...	1
	4	2023-01-09 00:0...	2023-01-17 00:0...	1

Tips för vidareläsning

- Entity Framework documentation | Microsoft Learn
- Persist and retrieve relational data with Entity Framework Core - Training | Microsoft Learn
- Database Providers - EF Core | Microsoft Learn
- Entity Framework CORE | 101
- Migrations Overview - EF Core | Microsoft Learn
- Saving Data - EF Core | Microsoft Learn
- SQL Injection
- Get started with Transact-SQL programming - Training | Microsoft Learn