Entity Framework

LID VAN ASSOCIATIE KU LEUVEN

Entity Framework: waarom?

The Entity Framework is a set of technologies in ADO.NET that support the development of data-oriented software applications. Architects and developers of data-oriented applications have struggled with the need to achieve two very different objectives. They must model the entities, relationships, and logic of the business problems they are solving, and they must also work with the data engines used to store and retrieve the data. The data may span multiple storage systems, each with its own protocols; even applications that work with a single storage system must balance the requirements of the storage system against the requirements of writing efficient and maintainable application code.

Entity Framework: waarom?

The Entity Framework enables developers to work with data in the form of domain-specific objects and properties, such as customers and customer addresses, without having to concern themselves with the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code than in traditional applications. Because the Entity Framework is a component of the .NET Framework, Entity Framework applications can run on any computer on which the .NET Framework is installed.

Zie: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview

Entity relationship model vs relationeel model

Entity Relationship vs Relationeel Model

- Peter Chen vs Edgar Codd

Relationeel model:

- Gaat over datanormalisatie
  - Vereenvoudigt het opslaan en onderhoud van data door duplicatie van gegevens te verminderen om zo dataconsistentie te verhogen (remember Relational Databases)

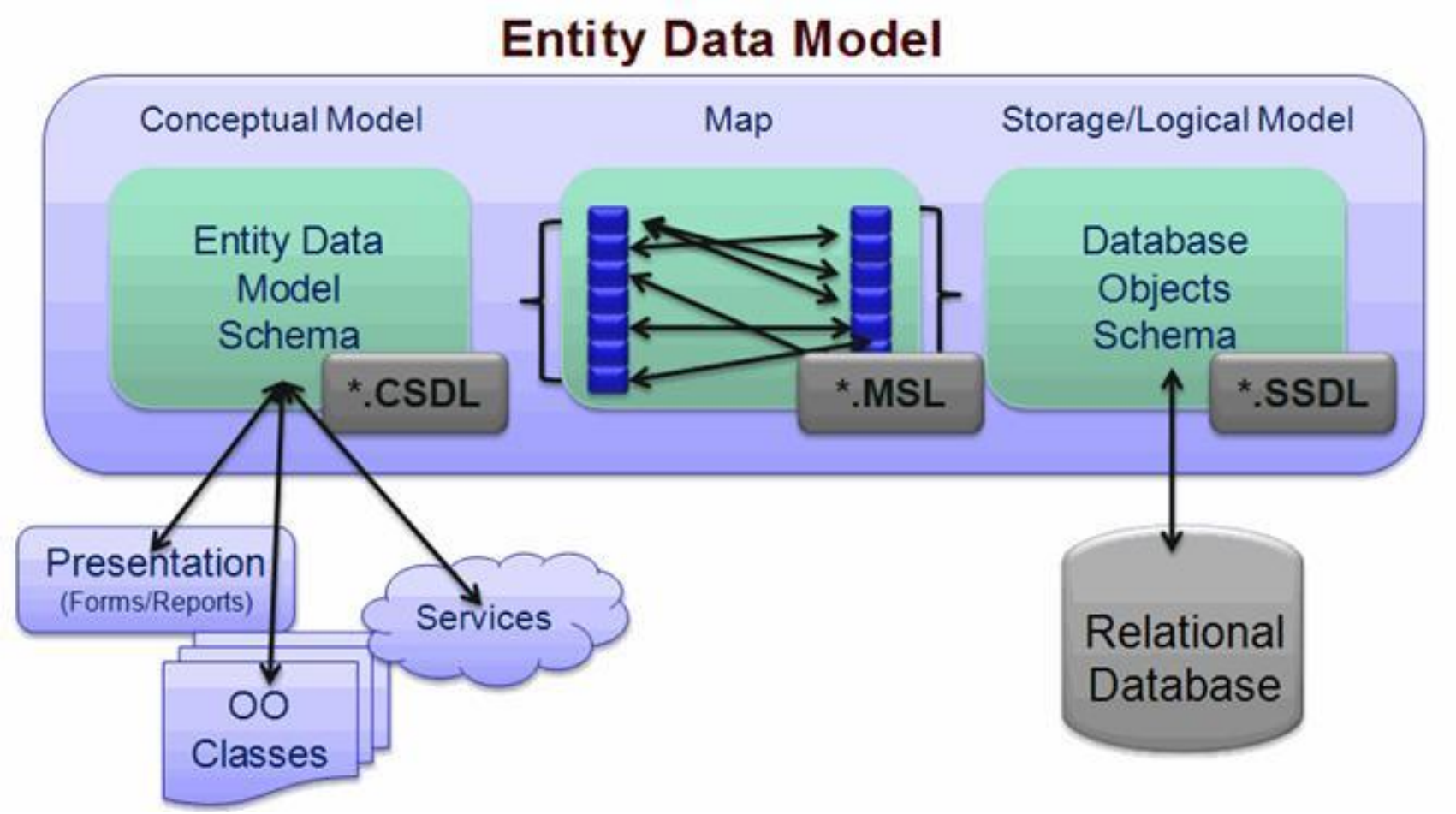
ER model

- Modeleert real-world concepten door complexe data op te delen in dingen (entiteiten) en associaties tussen die dingen (relaties)

Entity Framework

Entity Framework is Microsoft's implementatie van ER model die relationele databank schema's mapt op ENTITY DATA MODEL

Entity Data Model

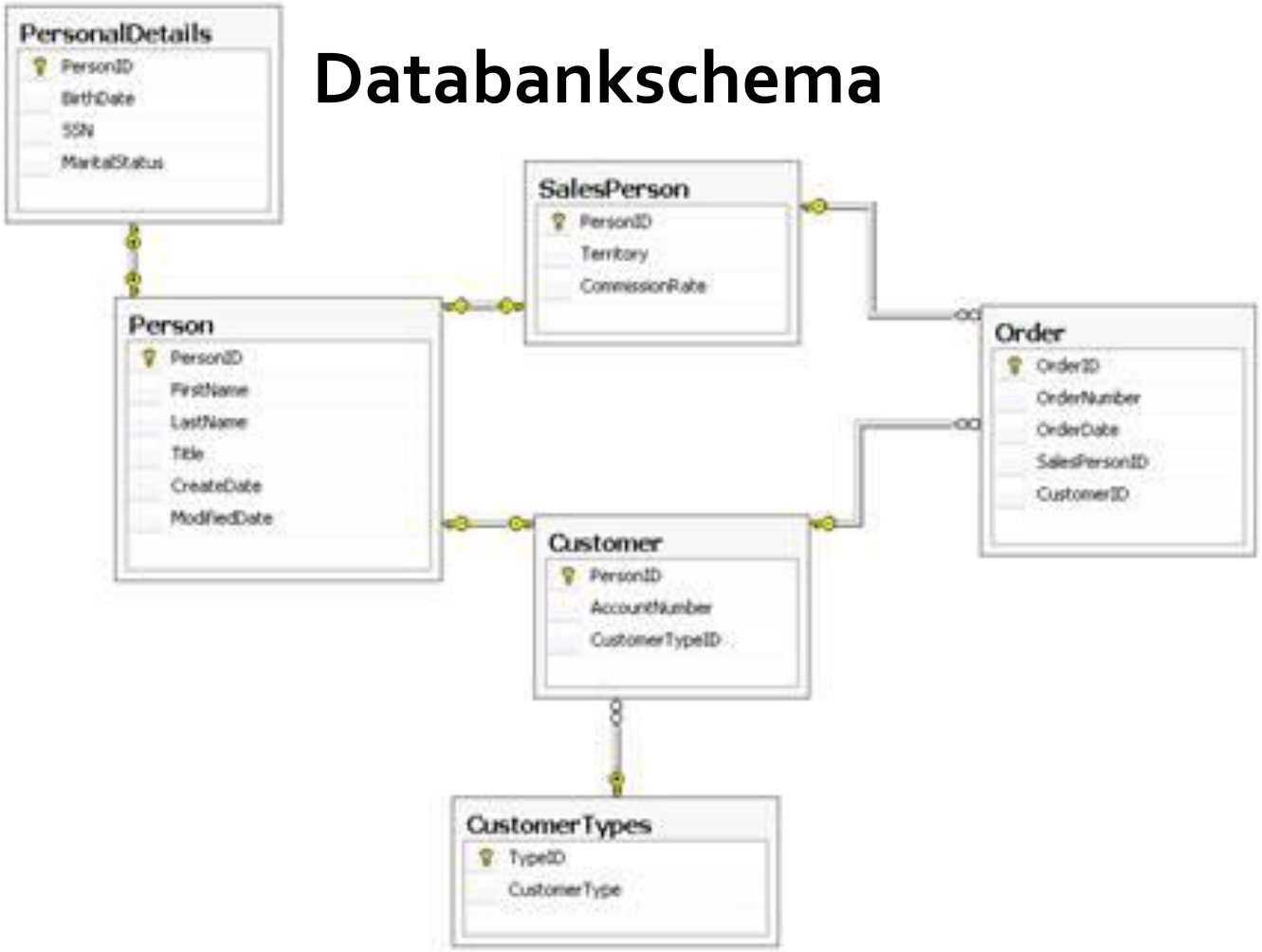


Entity Data Model in Entity Framework

Entity Data Model wizard

- Laat toe om vlug een model gebaseerd op een bestaande databank te creëren
- Voorbeeld

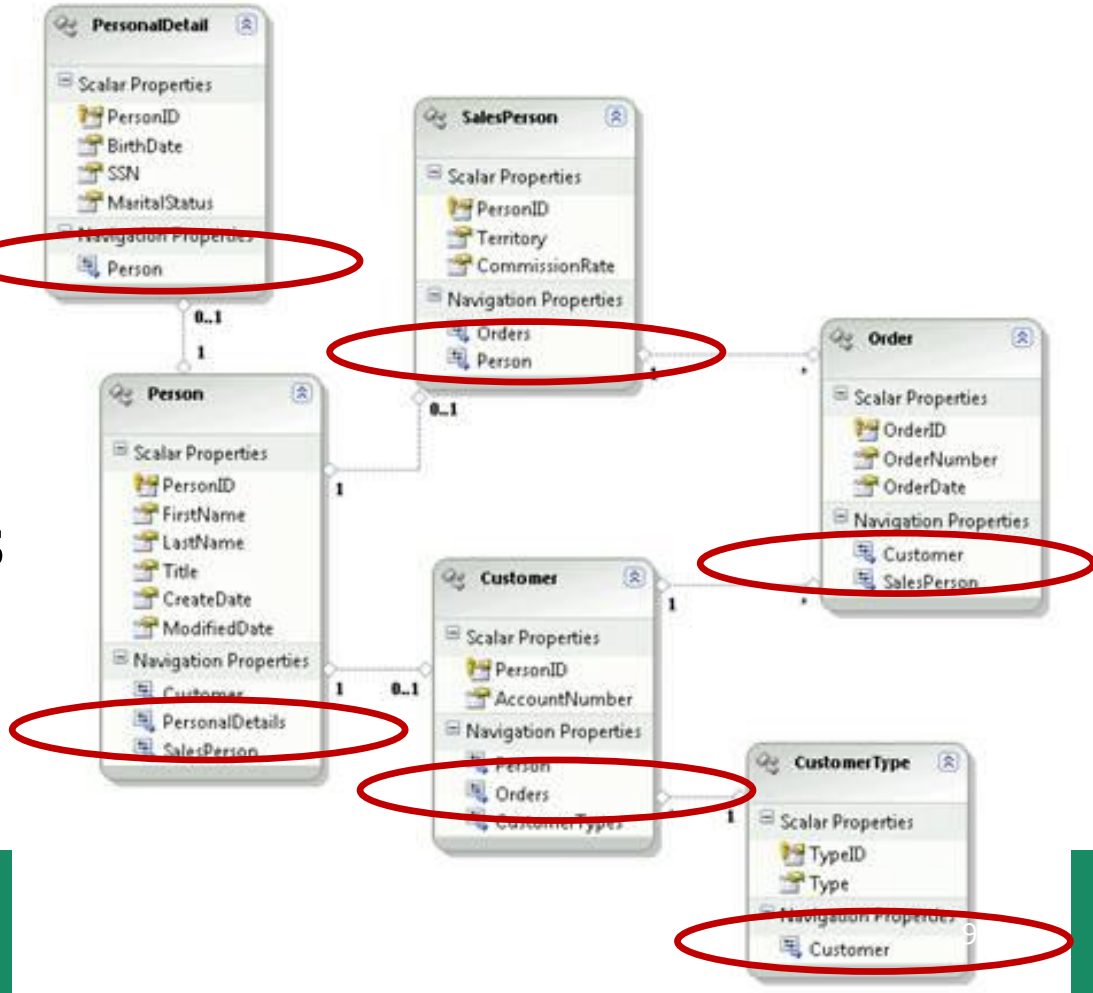
Entity Data Model in Entity Framework



Entity Data Model in Entity Framework

Corresponderende Model (gemaakt in EF Designer)

Relaties tussen 2 entities: niet via FK, maar door navigatieproperties



Entity Data Model in Entity Framework

Betekenis:

- Geen zorgen maken over JOINS om entities met elkaar te connecteren
- Entities zijn op een natuurlijke manier met elkaar verbonden

Van data model naar klassen

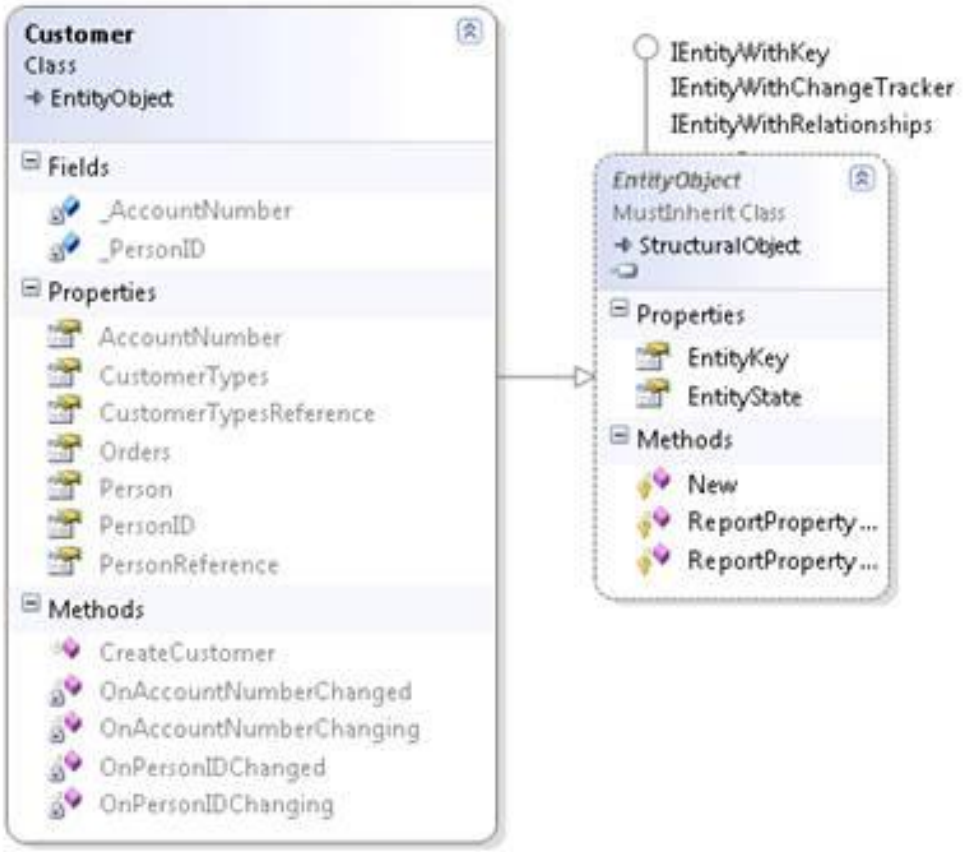
Doel van Entity Framework Designer

- Klassen genereren uit entities in model
- Elke entiteit wordt een klasse die overerft van EF EntityObject klasse
- EntityObject klasse voorziet de entiteiten met de mechanismes die nodig zijn om in het raamwerk ingeplugd te kunnen worden

Van data model naar klassen

Voorbeeld: Customer klasse

- Erft over van EntityObject



Entity Framework vs LINQ to SQL

Categorie	LINQ to SQL	Entity Framework
Complexiteit	redelijk	Complex
Model	Domein model	Conceptueel data model
Database Server	SQL Server	Verschillende RDBMS
Bestandstypes	DBML	EDMX, CDSL, MSL, SSDL
Query mogelijkheden	LINQ to SQL	LINQ to Entities, ESQL, Object services, Entity Client
Genereert databank uit model	Nee	Ja (Code First vs Database First)

Giving life to models

Entity Framework: laat ontwikkelaars toe om entities en relaties te queryen in het domein model (=conceptueel model in EF) terwijl er gesteund wordt op EF om deze operaties te vertalen naar databronafhankelijke commando's.

Dus geen hard-coded afhankelijkheden meer

Hoe? 3 –lagen-model!

Giving life to models

Conceptueel model, storage model (= logische model) en mapping tussen de twee modellen worden uitgedrukt in XML Schema's en worden gedefinieerd in

- Conceptual schema definition language (CSDL)
- Store schema definition language (SSDL)
- Mapping specification language (MSL)

Storage model en mappings kunnen veranderen zonder dat conceptueel model, data klassen of application code hoeft te veranderen

EF gebruikt model en mapping bestanden om entities en relaties te creëeren, lezen, updaten en deleten

3-TIER

Lagen in Entity Framework

Elk model: 3 lagen

- Conceptueel model
- Storage model
- Mapping

- Demo in Visual Studio

Conceptueel model

Laag definieert Entity Data Model (EDM)

Gedefinieerd mbv CSDL (Conceptual Schema Definition Language)

Storage model

- Identificeert onderliggende databank-elementen die het conceptueel model ondersteunen
- Wordt ook soms het logische model genoemd
- Storage model bevat entity en associatie definities: weerspiegelen logische voorstelling van de databank
- Bevat evenwel ook queries en stored procedures die gebruikt zullen worden door ADO.NET connecties en command objecten
- Model wordt gedefinieerd mbv SSDL (Store Schema Definition Language)

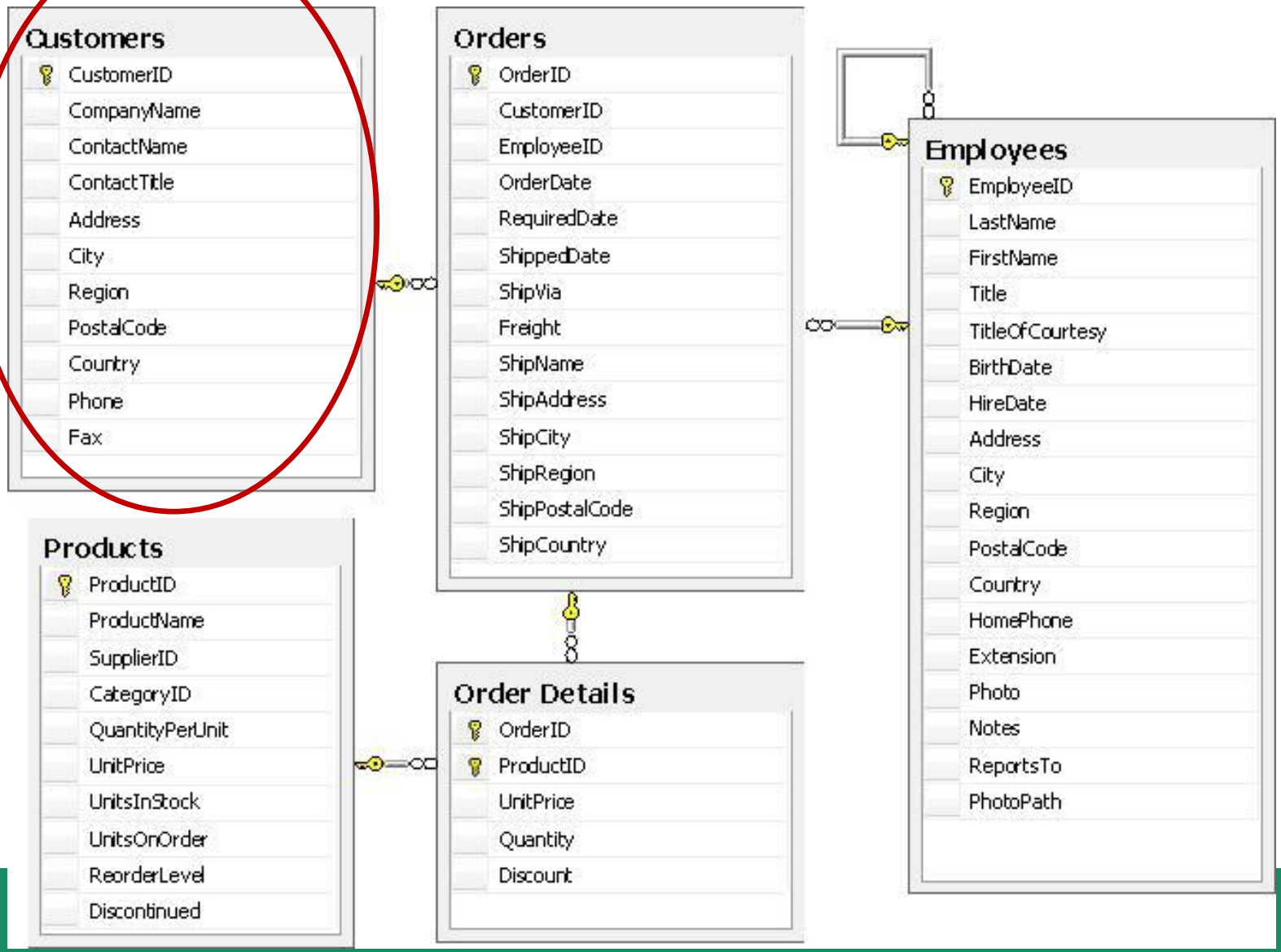
Model Mappings

Lijm tussen conceptueel en storage model

Taal: MSL (Mapping Specification Language)

- Mapping zegt hoe entities, properties, associaties in conceptuele model gekoppeld kunnen worden met specifieke items in storage model, die op hun beurt het pad naar databank voor elk stuk data definiëren

Voorbeeld


Customers
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath
Products
ProductID
ProductName
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued
Order Details
OrderID
ProductID
UnitPrice
Quantity
Discount

Entity?

Entiteit: instantie van EntityType (een “ding”)

EntityType: beschrijft eigenschappen (“Properties”) die structuur van entiteit beschrijven (bijv: verjaardag, gewicht, naam,…)

Moet een verzameling van sleuteleigenschappen bezitten die uniek de instantie identificeren (onderscheid maken tussen instanties van hetzelfde EntityType in een EntitySet)

- Komt overeen met PK in databanken

Relaties?

Definieert associaties (relaties) tussen entiteiten

Worden beschreven door

- AssociationType: definieert de types van entiteiten die deel uit maken van associaties
  - Bijv:
    - ManagerEmployee bestaat uit 2 Employee EntityTypes
    - Rollen: Manager en Werknemer
    - Kardinaliteit: elke werknemer heeft hoogstens 1 baas, terwijl baas verschillende werknemers kan hebben
- Relaties: 1-1, 1-n, of n-m

Containers?

EntitySets:

- Instanties van entiteiten leven in een EntitySet
- Één enkele instantie kan behoren tot slechts één EntitySet
- Komt overeen met een relationele tabel

RelationshipSets:

- Instanties van relaties leven in een RelationshipSet
- Komen overeen met join tabellen in relationele databanken

EntityContainers

- EntitySets en RelationshipSets zijn gedefinieerd in een EntityContainer

Corresponderende SSDL voor Customers tabel

```
<Schema Namespace="NorthwindModel.Store" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm/ssdl">
  <EntityContainer Name="dbo">
    <EntitySet Name="Customers“ EntityType="NorthwindModel.Store.Customer" />
  </EntityContainer>
  <EntityType Name="Customer">
    <Key>
      <PropertyRef Name="CustomerID" />
    </Key>
    <Property Name="CustomerID" Type="nchar“ Nullable="false" MaxLength="5" />
    <Property Name="CompanyName" Type="nvarchar“ Nullable="false" MaxLength="40" />
    <Property Name="ContactName" Type="nvarchar" MaxLength="30" />
    <Property Name="ContactTitle" Type="nvarchar" MaxLength="30" />
    <Property Name="Address" Type="nvarchar" MaxLength="60" />
    <Property Name="City" Type="nvarchar" MaxLength="15" />
    <Property Name="Region" Type="nvarchar" MaxLength="15" />
    <Property Name="PostalCode" Type="nvarchar" MaxLength="10" />
```

Corresponderende SSDL voor Customers tabel

```
    <Property Name="Country" Type="nvarchar" MaxLength="15" />
    <Property Name="Phone" Type="nvarchar" MaxLength="24" />
    <Property Name="Fax" Type="nvarchar" MaxLength="24" />
  </EntityType>

</Schema>
```

Corresponderende CSDL bestand voor Customers EntitySet

```
<Schema Namespace="NorthwindModel" Alias="Self“ xmlns="http://schemas.microsoft.com/ado/2006/04/edm">

  <EntityContainer Name="NorthwindEntities">
    <EntitySet Name="Customers"
               EntityType="NorthwindModel.Customer" />
  </EntityContainer>

  <EntityType Name="Customer">
    <Key>
      <PropertyRef Name="CustomerID" />
    </Key>
    <Property Name="CustomerID" Type="String“ Nullable="false" MaxLength="5" FixedLength="true" />
    <Property Name="CompanyName" Type="String“ Nullable="false" MaxLength="40" />
```

Corresponderende CSDL bestand voor Customers EntitySet

```
    <Property Name="ContactName" Type="String" MaxLength="30" />
    <Property Name="Title" Type="String" MaxLength="30" />
    <Property Name="Address" Type="String" MaxLength="60" />
    <Property Name="City" Type="String" MaxLength="15" />
    <Property Name="Region" Type="String" MaxLength="15" />
    <Property Name="PostalCode" Type="String" MaxLength="10" />
    <Property Name="Country" Type="String" MaxLength="15" />
    <Property Name="Phone" Type="String" MaxLength="24" />
    <Property Name="Fax" Type="String" MaxLength="24" />
  </EntityType>

</Schema>
```

Corresponderende MSL bestand voor mapping tussen Customers tabel en Customers EntitySet

```
<Mapping Space="C-S“ xmlns="urn:schemas-microsoft-
com:windows:storage:mapping:CS">

<EntityContainerMapping StorageEntityContainer="dbo“
                        CdmEntityContainer="NorthwindEntities">
  <EntitySetMapping Name="Customers" StoreEntitySet="Customers"
                    TypeName="NorthwindModel.Customer">
    <ScalarProperty Name="CustomerID" ColumnName="CustomerID" />
    <ScalarProperty Name="CompanyName" ColumnName="CompanyName" />
    <ScalarProperty Name="ContactName" ColumnName="ContactName" />
    <ScalarProperty Name="Title" ColumnName="ContactTitle" />
    <ScalarProperty Name="Address" ColumnName="Address" />
    <ScalarProperty Name="City" ColumnName="City" />
    <ScalarProperty Name="Region" ColumnName="Region" />
    <ScalarProperty Name="PostalCode" ColumnName="PostalCode" />
```

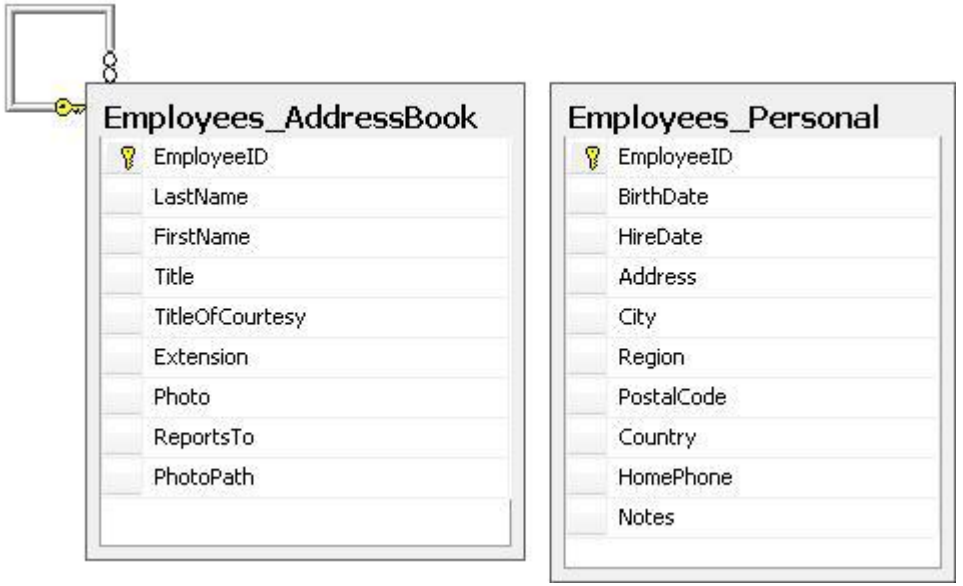
Corresponderende MSL bestand voor mapping tussen Customers tabel en Customers EntitySet

```
    <ScalarProperty Name="Country" ColumnName="Country" />
    <ScalarProperty Name="Phone" ColumnName="Phone" />
    <ScalarProperty Name="Fax" ColumnName="Fax" />
  </EntitySetMapping>

</EntityContainerMapping>

</Mapping>
```

Mapping van één enkele entity naar meerdere tabellen

Stel:

- Informatie gedefinieerd in EntityType is verspreid over meerdere tabellen in de databank
- Bijv: informatie van Employees entiteit is opgesplitst in 2 tabellen
- In dit geval: CSDL blijft hetzelfde
- MSL + SSDL: veranderen



MSL mapping tussen Employee EntityType en 2 relationele tabellen

```
<EntitySetMapping Name="Employees“ TypeName="NorthwindModel.Employee">

    <MappingFragment StoreEntitySet="Employees_AddressBook">
      <ScalarProperty Name="EmployeeID" ColumnName="EmployeeID" />
      <ScalarProperty Name="LastName" ColumnName="LastName" />
      <ScalarProperty Name="FirstName" ColumnName="FirstName" />
      <ScalarProperty Name="Title" ColumnName="Title" />
      <ScalarProperty Name="TitleOfCourtesy"
                      ColumnName="TitleOfCourtesy" />
      <ScalarProperty Name="Extension" ColumnName="Extension" />
      <ScalarProperty Name="Photo" ColumnName="Photo" />
      <ScalarProperty Name="PhotoPath" ColumnName="PhotoPath" />
    </MappingFragment>
```

MSL mapping tussen Employee EntityType en 2 relationele tabellen

```
<MappingFragment StoreEntitySet="Employees_Personal" >
  <ScalarProperty Name="EmployeeID" ColumnName="EmployeeID" />
  <ScalarProperty Name="BirthDate" ColumnName="BirthDate" />
  <ScalarProperty Name="HireDate" ColumnName="HireDate" />
  <ScalarProperty Name="Address" ColumnName="Address" />
  <ScalarProperty Name="City" ColumnName="City" />
  <ScalarProperty Name="Region" ColumnName="Region" />
  <ScalarProperty Name="PostalCode" ColumnName="PostalCode" />
  <ScalarProperty Name="Country" ColumnName="Country" />
  <ScalarProperty Name="HomePhone" ColumnName="HomePhone" />
  <ScalarProperty Name="Notes" ColumnName="Notes" />
</MappingFragment>

</EntitySetMapping>
```

Mapping relationships

Relationele databanken gebruiken foreign keys om referentiële integriteit tussen tabellen op te leggen

EF laat toe om deze foreign keys te mappen naar relaties door *Associations*

Voorbeeld van een associatie tussen Customers en Orders (SSDL):

```
<Association Name="FK_Orders_Customers">
    <End Role="Customers“ Type="NorthwindModel.Store.Customers"
         Multiplicity="0..1" />
    <End Role="Orders“ Type="NorthwindModel.Store.Orders“
         Multiplicity="*" />
</Association>
```

Mapping relationships

In CSDL wordt ons voorbeeld dan

```
<Association Name="FK_Orders_Customers">
    <End Role="Customers“ Type="NorthwindModel.Customers"
         Multiplicity="0..1" />
    <End Role="Orders“ Type="NorthwindModel.Orders“
         Multiplicity="*" />
</Association>
```

3 soorten overerving

OVERERVING

Overerving

Overerving is meestal een probleem in databanken

- In OO-programmeren komt dit vaak voor
- Maar niet evident om dit door te trekken naar data in databank
- Relationele databank kent het concept overerving niet

In ORM wordt dit op 3 manieren opgelost:

- Table per Class Hierarchy (TPH)
- Table per Type (TPT)
- Table per Concrete Class (TPC)

Table per Class Hierarchy (TPH)

Meest eenvoudige manier om overerving in databank te realiseren

Ook gekend als Single Table Inheritance

Alle concrete types in de overervingshiërarchie worden opgeslagen in 1 tabel

Extra kolom definieren die onderscheid maakt tussen subklassen

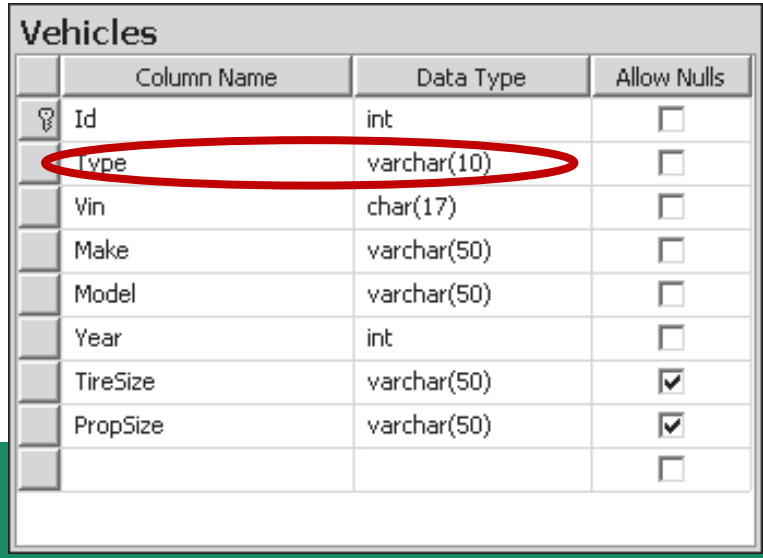
TPH: voorbeeld

Stel: abstracte klasse Vehicle (Id, Vin, Make, Model, Year)

- Car (TireSize) en Boat (PropellerSize) zijn subklassen (erven over van Vehicle)

Wordt opgeslagen in 1 enkele tabel

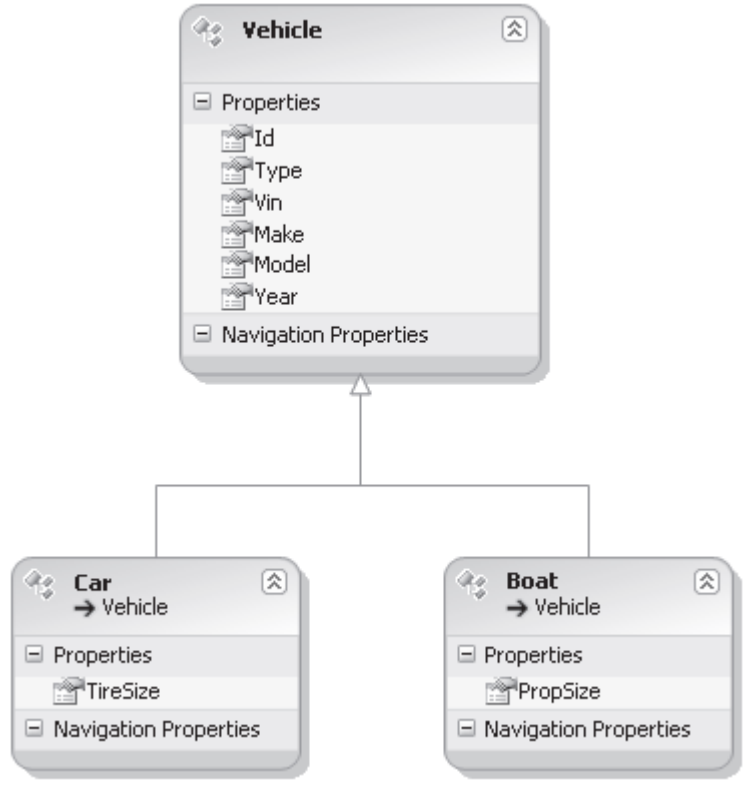
- Merk op:
  - TireSize en PropSize laten Null toe!



TPH: voorbeeld

Via wizard (Entity Data Model > Generate from database kiezen)

- Resultaat (na zelf Entities Cars en Boat bij te maken die overerven van Vehicle)



TPH: voorbeeld

Vul tabel in databank op met data:

ID	Type	Vin	Make	Model	Year	TireSize	PropSize
1	Car	ABC123	BMW	Z-4	2009	225/45R17	Null
2	Boat	DEF234	SeaRay	SunDeck	2005	Null	14.75x21
3	Car	GHI345	VW	Beatle	2007	205/55R16	Null
4	Boat	JKL456	Bayliner	3288	1993	Null	14.5x18

Code (alle Cars tonen in DataGridView)

```
private void tPHDisplayCarsToolStripMenuItem_Click(object sender, EventArgs e) {
        var db = new TablePerHierarchy.TablePerHierarchyEntities();
        gv.DataSource = (from c in db.Vehicles.OfType<TablePerHierarchy.Car>()
                         select c).ToList();
}
```

TPH: voorbeeld

Code (alle Boats tonen in DataGridView)

```
private void tPHDisplayBoatsToolStripMenuItem_Click(object sender, EventArgs e)
{
    var db = new TablePerHierarchy.TablePerHierarchyEntities();
    gv.DataSource = (from b in b.Vehicles.OfType<TablePerHierarchy.Boat>()
                     select b).ToList();
}
```

Zie ook http://msdn.microsoft.com/en-us/data/jj618292

Table per Type (TPT)

TPT: meest efficiënte methode

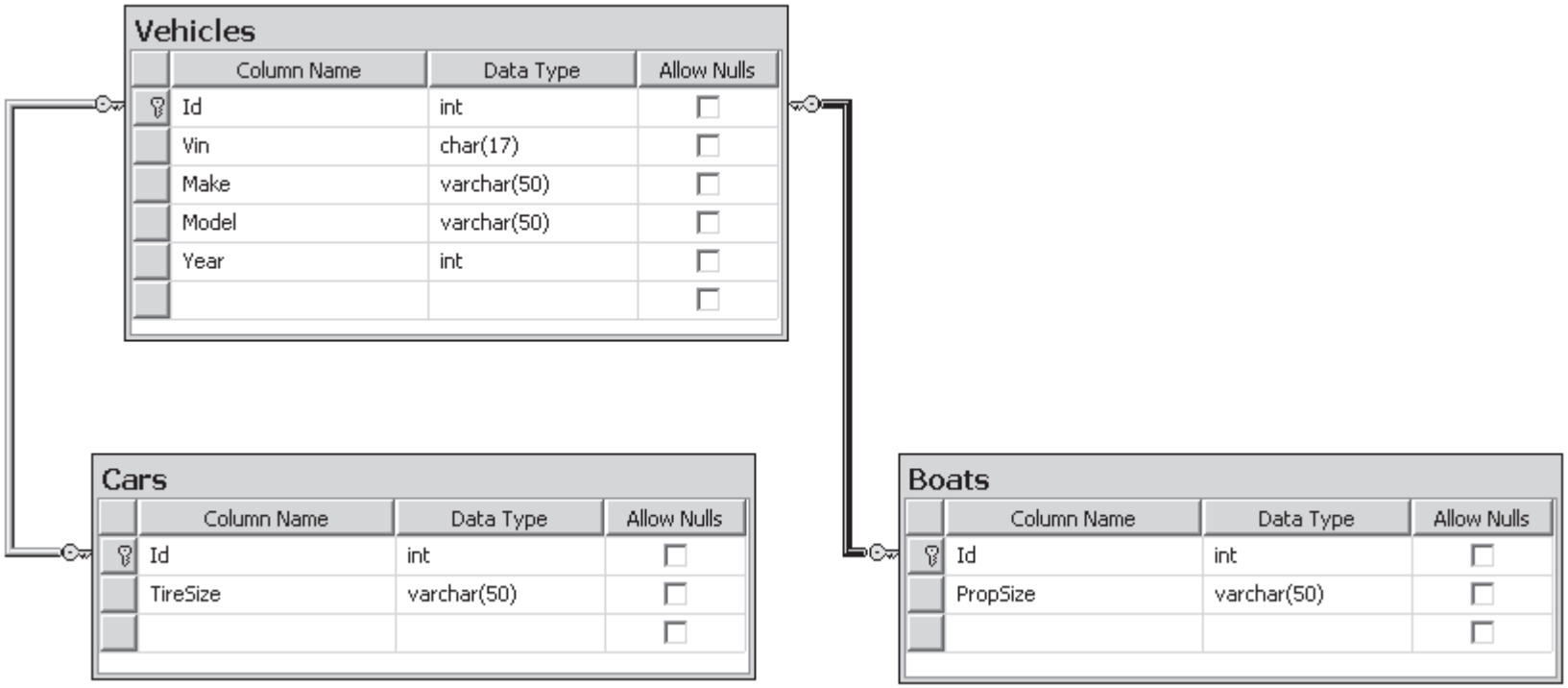
Elk type in de overervingshiërarchie wordt in zijn eigen tabel opgeslagen

Één-op-één mapping tussen type en tabel: meest logische van de 3 methodes

- Toename in het aantal joins

TPT: voorbeeld

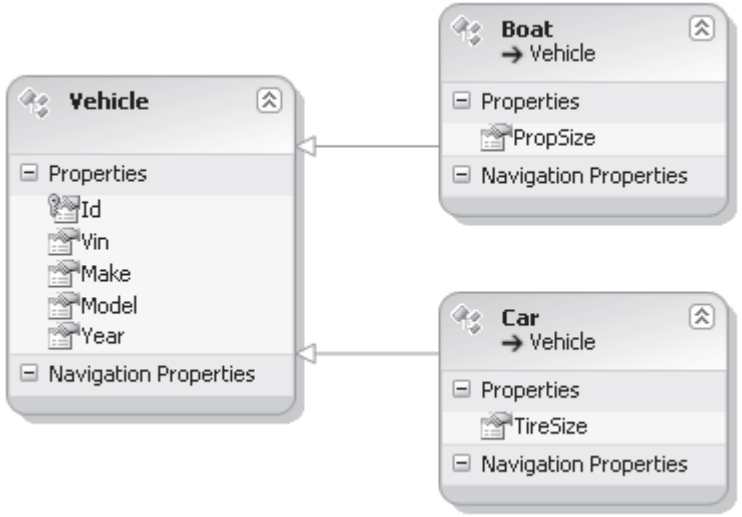
Voorstelling in databank:



TPT: voorbeeld

Entity Data Model

- Merk op: overerving moet je zelf aanpassen in de wizard



Zie ook http://msdn.microsoft.com/en-us/data/jj618293

Table per Concrete Class (TPC)

In deze laatste overervingsmethode

- Tabel voorzien voor elke CONCRETE klasse
- GEEN tabel voorzien voor ABSTRACTE klassen

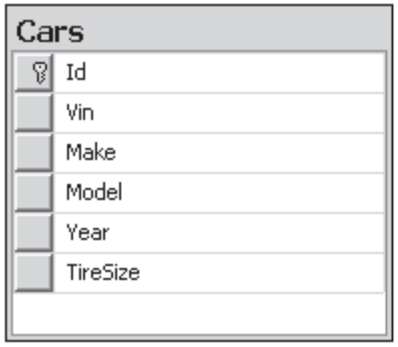
Nadeel:

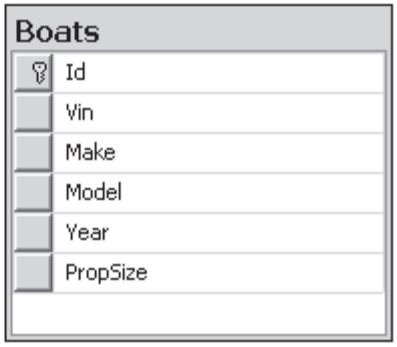
- Dubbele data neemt toe!

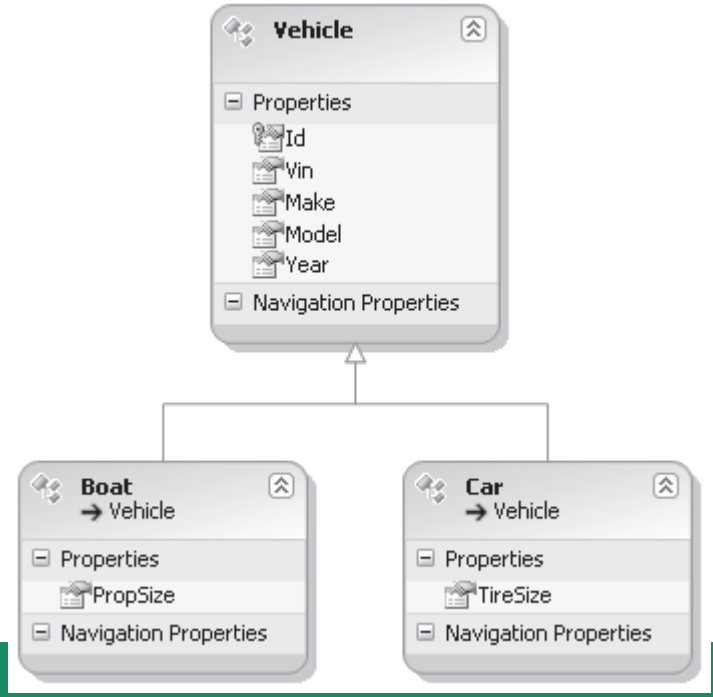
Niet standaard voorzien in EF

- Manueel XML bestanden aanpassen

TPC: voorbeeld







Entity Framework 6

DEMO

Model First vs Database First vs Code First Model

Model First

- Conceptueel model maken vóór het maken van databank
- Je kan databank genereren uit conceptueel model

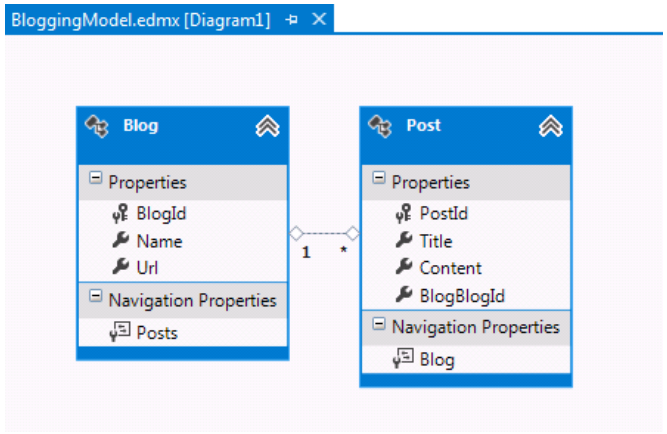
Database First

- Omgekeerd
- Conceptueel model laten genereren uit bestaande databank schema

Code First

- Klassen maken in C#
- Daaruit databank laten genereren

EF Development Workflows




```
public class Blog{
    public int BlogId { get; set; }
    public string Name { get; set; }
    public string Url { get; set; }

    public virtual List<Post> Posts { get;
                            set; }
}
public class Post{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }
    public int BlogId { get; set; }

    public virtual Blog Blog { get; set; }
}
```

New Database

Model First
- Maak model in designer
- Model => databank
- Model => klassen

Code First
- Klassen & mapping maken in code
- Model => databank
- Gebruik Migrations om databank aan te passen

Existing Database

Database First
- Reverse engineer model in designer
- Model => klassen

Code First
- Klassen & mapping maken in code
- Gebruik reverse engineer tools

EF Database First

Demo

Zie ook: https://msdn.microsoft.com/en-us/library/jj206878(v=vs.113).aspx

Entity Framework Code First to an Existing Database

Demo

Zie ook: https://msdn.microsoft.com/en-us/library/jj200620(v=vs.113).aspx

EF Code First to a New Database

Demo

Zie ook: https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx

EF Model First (new database)

Zie: https://msdn.microsoft.com/en-us/library/jj205424(v=vs.113).aspx

DBCONTEXT

DbContext

Zorgt voor interactie van data (uit databank) als objecten

DbContext beheert Entity-objecten tijdens run time, wat inhoudt het populeren van objecten van data uit databank, veranderingen bijhouden en data terugschrijven naar de databank

Een afgeleide klasse van DBContext maken

Beste manier is om klasse te maken die een subklasse is van DbContext en die DbSet properties die collecties van de bewuste entiteiten voorstellen teruggeeft

```
public class BloggingContext : DbContext {
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}
```

DbContext: levensduur

Lifetime van een context begint wanneer instantie wordt gemaakt en eindigt wanneer de instantie wordt verwijderd

Gebruik hiervoor using

```
using (var context = new BloggingContext()){
...
}
```

Indien je werkt met Windows Forms => gebruik een context instantie per formulier

DbContext

```
using (var db = new BloggingContext()){
            Console.Write("Enter a name for a new Blog: ");
            var name = Console.ReadLine();

            var blog = new Blog { Name = name };
            db.Blogs.Add(blog);
            db.SaveChanges();

            // Display all Blogs from the database
            var query = from b in db.Blogs
                        orderby b.Name
                        select b;

            Console.WriteLine("All blogs in the database:");
            foreach (var item in query)
            {
                Console.WriteLine(item.Name);
            }

            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
}
```

DBContext

http://msdn.microsoft.com/en-us/data/jj729737

Bronnen

MSDN Website over ADO.NET Entity Framework: https://msdn.microsoft.com/en-US/data/ef

Accessing Data with Microsoft .NET Framework 4: https://ptgmedia.pearsoncmg.com/images/9780735627390/samplepages/9780735627390.pdf

Introducing ADO.NET Entity Framework: http://www.code-magazine.com/article.aspx?quickid=0711051&page=1

Code First voorbeelden

http://www.code-magazine.com/Article.aspx?quickid=1108051