



**C# 2018 - FRAMEWORK
ADO.NET – ENTITY FRAMEWORK**

Versie: 01-mei-2018

INHOUDSOPGAVE

1	INLEIDING	9
1.1	DOELSTELLING	9
1.2	VEREISTE VOORKENNIS.....	9
1.3	NODIGE SOFTWARE	9
2	DE OBJECT-RELATIONAL MISMATCH	10
2.1	ALGEMEEN.....	10
2.2	GRANULARITY.....	10
2.2.1	ALGEMEEN	10
2.2.2	VOORBEELD	10
2.3	INHERITANCE.....	11
2.3.1	TABLE PER CONCRETE CLASS	12
2.3.2	TABLE PER CLASS HIERARCHY.....	12
2.3.3	TABLE PER SUBCLASS.....	13
2.4	ASSOCIATIES.....	14
2.4.1	ÉÉN-OP-VEEL ASSOCIATIES	14
2.4.2	VEEL-OP-VEEL ASSOCIATIES	14
2.5	NAVIGEREN DOOR ASSOCIATIES	15
2.6	ORM (OBJECT-RELATIONAL MAPPING)	16
2.7	EDMX BESTAND VERSUS CODE FIRST	17
3	DE DATABASE	18
4	HET ENTITY DATA MODEL (EDM)	24
4.1	ALGEMEEN.....	24
4.2	JE EERSTE EDM	24
4.3	HET EDM ALS XML-BESTAND.....	32
4.3.1	ALGEMEEN	32
4.3.2	STORAGE MODELS.....	34
4.3.3	CONCEPTUAL MODELS.....	35
4.3.4	MAPPINGS.....	36
4.4	DE MODEL BROWSER	36

4.5	EXTRA TECHNIEKEN IN DE GRAFISCHE DESIGNER	38
4.6	DE DBCONTEXT CLASS	39
4.7	ENTITY CLASSES UITBREIDEN MET PROPERTIES EN METHODS	40
4.8	TAAK 01 : EFBANK MAKEN	41
5	QUERIES.....	42
5.1	ALGEMEEN.....	42
5.1.1	EEN FOREACH ITERATIE.....	42
5.1.2	EEN LINQ-QUERY.....	43
5.1.3	QUERY-METHODS	44
5.1.4	LINQ-QUERIES EN QUERIES MET METHODS VERGELEKEN.....	45
5.2	EEN ENTITY ZOEKEN OP ZIJN PRIMAIRE-KEY-WAARDE	48
5.3	GEDEELTELijke OBJECTEN OPHALEN.....	49
5.4	GROEPEREN IN QUERIES	50
5.5	LAZY LOADING	52
5.6	EAGER LOADING LOST HET PERFORMANTIEPROBLEEM OP	53
5.7	DE METHOD TOLIST VAN EEN QUERY	56
5.8	TAAK 02 : KLANTEN EN HUN REKENINGEN	58
6	ENTITIES TOEVOEGEN	59
6.1	ÉÉN ENTITY TOEVOEGEN	59
6.2	MEERDERE ENTITIES TOEVOEGEN	60
6.3	ENTITIES MET NIEUWE GEASSOCIEERDE ENTITIES TOEVOEGEN	60
6.4	EEN ENTITY TOEVOEGEN MET EEN ASSOCIATIE NAAR EEN BESTAANDE ENTITY	62
6.4.1	EEN ENTITY TOEVOEGEN EN DE ASSOCIATIE DEFINIËREN VANUIT DE VEEL-KANT	62
6.4.2	EEN ENTITY TOEVOEGEN EN DE ASSOCIATIE DEFINIËREN VANUIT DE ÉÉN-KANT	64
6.5	TAAK 03 : ZICHTREKENING TOEVOEGEN	65
7	ENTITIES WIJZIGEN	66
7.1	ÉÉN ENTITY WIJZIGEN	66
7.2	MEERDERE ENTITIES LEZEN EN SLECHTS ENKELE DAARVAN WIJZIGEN.....	67
7.3	ENTITIES WIJZIGEN DIE JE INDIRECT GELEZEN HEBT MET ASSOCIATIES	67

7.4	EEN ASSOCIATIE VAN EEN ENTITY WIJZIGEN	69
7.4.1	DE ASSOCIATIE WIJZIGEN VANUIT DE VEEL-KANT	69
7.4.2	DE ASSOCIATIE WIJZIGEN VANUIT DE ÉÉN-KANT	71
7.5	TAAK 04 : STORTEN	72
8	ENTITIES VERWIJDEREN.....	73
8.1	TAAK 05 : KLANT VERWIJDEREN.....	74
9	TRANSACTIES.....	75
9.1	ALGEMEEN.....	75
9.2	ISOLATION LEVEL.....	76
9.3	DE METHOD SAVECHANGES	77
9.4	EIGEN TRANSACTIEBEHEER MET TRANSACTIONSCOPE	77
9.4.1	ALGEMEEN	77
9.4.2	VOORBEELD	79
9.5	TAAK 06 : OVERSCHRIJVEN	84
10	OPTIMISTIC RECORD LOCKING	85
10.1	ALGEMEEN.....	85
10.2	TABLE ZONDER TIMESTAMP-KOLOM	87
10.3	TABLE MET TIMESTAMP-KOLOM	90
10.4	TAAK 07 : KLANT WIJZIGEN	92
11	ASSOCIATIES.....	94
11.1	ALGEMEEN.....	94
11.2	VEEL-OP-VEEL-ASSOCIATIES ZONDER EXTRA ASSOCIATIEINFORMATIE	94
11.2.1	DE TABLES BOEKEN, CURSUSSSEN EN BOEKENCURSUSSSEN TOEVOEGEN AAN DE DATABASE	94
11.2.2	DE ENTITIES DEFINIËREN DIE BIJ DE TABLES BOEKEN EN CURSUSSSEN HOREN	94
11.2.3	DE ENTITIES GEBRUIKEN VANUIT JE CODE	95
11.3	VEEL-OP-VEEL-ASSOCIATIES MET EXTRA ASSOCIATIEINFORMATIE	97
11.3.1	DE TABLES BOEKEN2, CURSUSSSEN2 EN BOEKENCURSUSSSEN2 TOEVOEGEN AAN DE DATABASE	97
11.3.2	DE ENTITIES DEFINIËREN DIE BIJ DE TABLES BOEKEN2, CURSUSSSEN2 EN BOEKENCURSUSSSEN2 HOREN	98
11.3.3	DE ENTITIES GEBRUIKEN VANUIT JE CODE	99
11.4	EEN ASSOCIATIE VAN EEN ENTITY-CLASS NAAR ZICHZELF	102
11.4.1	ALGEMEEN	102

11.4.2	DE TABLE CURSISTEN TOEVOEGEN AAN DE DATABASE	102
11.4.3	DE ENTITY DEFINIËREN DIE BIJ DE TABLE CURSISTEN HOORT.....	102
11.4.4	DE ENTITIES GEBRUIKEN VANUIT JE CODE	103
11.5	TAAK 08 : PERSONEEL.....	107
12	INHERITANCE	109
12.1	ALGEMEEN.....	109
12.2	TABLE PER CONCRETE CLASS (TPC).....	109
12.2.1	ALGEMEEN	109
12.2.2	DE TABLES TPCKLASSIKALECURSUSSEN EN TPCZELFSTUDIECURSUSSEN TOEVOEGEN AAN DE DATABASE.....	109
12.2.3	DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TABLES TPCKLASSIKALECURSUSSEN, EN TPCZELFSTUDIECURSUSSEN ..	111
12.2.4	DE ENTITY CURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN	113
12.2.5	DE ENTITIES GEBRUIKEN VANUIT JE CODE	116
12.3	TABLE PER HIERARCHY (TPH)	119
12.3.1	DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TPHCURSUSSEN	119
12.3.2	DE ENTITY KLOSSIKALECURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN	120
12.3.3	DE ENTITY ZELFSTUDIECURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN.....	122
12.3.4	DE ENTITIES GEBRUIKEN VANUIT JE CODE	124
12.4	TABLE PER TYPE (TPT)	126
12.4.1	DE TABLES TPTCURSUSSEN, TPTKLASSIKALECURSUSSEN EN TPTZELFSTUDIECURSUSSEN TOEVOEGEN AAN DE DATABASE	126
12.4.2	DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TABLES TPTCURSUSSEN, TPTKLASSIKALECURSUSSEN EN TPTZELFSTUDIECURSUSSEN	127
12.4.3	INHERITANCE DEFINIËREN	128
12.4.4	DE ENTITIES GEBRUIKEN VANUIT JE CODE	129
12.5	TAAK 09 : ZICHTREKENINGEN – SPAARREKENINGEN	131
13	COMPLEX TYPES	132
13.1	ALGEMEEN.....	132
13.2	EEN COMPLEX TYPE MAKEN OP BASIS VAN EEN BESTAANDE ENTITY.....	133
13.3	EEN COMPLEX TYPE HERBRUIKEN IN EEN ANDERE ENTITY	135
13.4	COMPLEX TYPE ALS PARTIAL CLASS	137
13.5	VOORBEELDGEbruik.....	137
14	ENUMS	139
14.1	ALGEMEEN.....	139
14.2	VOORDELEN VAN EEN ENUM	139

14.2.1	DE COMPILER CONTROLEERT DE inhoud VAN EEN ENUM-VARIAbELE	139
14.2.2	VISUAL STUDIO HELPT BIJ HET INVULLEN VAN EEN ENUM-VARIAbELE	139
14.3	ENUMS EN EF	139

15	VIEWS	144
-----------	--------------------	------------

15.1	ALGEMEEN.....	144
15.2	EEN VIEW AANMAKEN	144
15.3	DE DATA VAN EEN VIEW LEZEN VANUIT SQL	144
15.4	DE VOORBEELDVIEW	145
15.5	DE ENTITIES DEFINIËREN DIE HOREN BIJ DE VIEW	146
15.6	DE ENTITIES AANSPREKEN VANUIT CODE.....	148
15.7	TAk 10 : TOTALE SALDO PER KLANT	149

16	STORED PROCEDURES	150
-----------	--------------------------------	------------

16.1	ALGEMEEN.....	150
16.2	EEN STORED PROCEDURE AANMAKEN	150
16.3	EEN STORED PROCEDURE OPROEPEN VANUIT SQL	150
16.4	STORED PROCEDURES OPROEPEN MET EF	151
16.4.1	EEN STORED PROCEDURE DIE DATA TERUG GEEFT IN DE VORM VAN ENTITIES	151
16.4.2	EEN STORED PROCEDURE DIE DATA TERUG GEEFT IN EEN VORM DIE NIET OVEREENSTEMT MET DE STRUCTUUR VAN EEN ENTITY	
	155	
16.4.3	EEN STORED PROCEDURE DIE GEEN DATA TERUG GEEFT	158
16.4.4	EEN STORED PROCEDURE DIE DATA LEEST ALS EEN SCALAR VALUE	160
16.5	TAk 11 : ADMINISTRATIEVE KOST	161

17	CODE FIRST	162
-----------	-------------------------	------------

17.1	ALGEMEEN.....	162
17.2	DE ENTITY CLASSES VOOR DE NIEUWE DATABASE	162
17.3	DE DbContext CLASS	165
17.4	DE CONNECTIONSTRING	166
17.5	DE DbContext GEBRUIKEN	166
17.6	DE DATABASE OPNIEUW MAKEN	167

17.7	DE AANGEMAakte TABLE STRUCTUUR VERFIJNEN	169
17.7.1	EXPLICET DE TABLE-NAAM INSTELLEN	169
17.7.2	EXPLICET DE KOLOMNAAM INSTELLEN.....	170
17.7.3	EEN KOLOM INSTELLEN ALS VERPLICHT IN TE VULLEN.....	170
17.7.4	EEN KOLOM INSTELLEN ALS NIET VERPLICHT IN TE VULLEN	171
17.7.5	HET MAXIMUM AANTAL TEKENS IN EEN VARCHAR-KOLOM INSTELLEN.....	171
17.7.6	HET KOLOMTYPE INSTELLEN.....	171
17.7.7	DE PROPERTY INSTELLEN DIE BIJ DE PRIMARY KEY HOORT	172
17.7.8	EEN PRIMARY KEY DIE GEEN INT MET AUTONUMBER IS	173
17.8	COMPLEX TYPE	175
17.9	INHERITANCE.....	179
17.9.1	TABLE PER HIERARCHY (TPH)	179
17.9.2	TABLE PER TYPE (TPT)	182
17.9.3	TABLE PER CONCRETE CLASS (TPC)	186
17.10	ASSOCIATIES TUSSEN ENTITIES	189
17.10.1	ÉÉN-OP-VEEL-ASSOCIATIES.....	189
17.10.2	VEEL-OP-VEEL-ASSOCIATIES.....	195
17.10.3	EEN ASSOCIATIE NAAR DEZELFDE TABLE.....	200
17.11	TAAK 12 : CODE FIRST	204
18	WPF	205
18.1	DE ENTITY CLASSES VOOR DE NIEUWE DATABASE	205
18.2	DE DbCONTEXT CLASS	205
18.3	DE CONNECTIONSTRING	206
18.4	EEN INSTANTIE VAN DE DbCONTEXT CLASS	206
18.5	EEN LISTBOX TONEN MET DATA UIT DE DATABASE.....	207
18.6	EEN LISTBOX MET GERELATEERDE DATA TONEN.....	210
18.7	EEN DATAGRID TONEN MET DATA UIT DE DATABASE	211
18.8	DATA WIJZIGEN.....	212
19	VOORBEELDOPLOSSINGEN	215
19.1	TAAK 01 : BANK MAKEN.....	215
19.2	TAAK 02 : KLANTEN EN HUN REKENINGEN	215
19.3	TAAK 03 : ZICHTREKENING TOEVOEGEN	216
19.4	TAAK 04 : STORTEN.....	217

19.5	TAAK 05 : KLANT VERWIJDEREN.....	218
19.6	TAAK 06 : OVERSCHRIJVEN	219
19.7	TAAK 07 : KLANT WIJZIGEN	221
19.8	TAAK 08 : PERSONEEL.....	222
19.9	TAAK 09 : ZICHTREKENINGEN – SPAARREKENINGEN	223
19.10	TAAK 10 : TOTALE SALDO PER KLANT	224
19.11	TAAK 11: ADMINISTRATIEVE KOST	225
19.12	TAAK 12 : CODE FIRST	226
19.12.1	INSTALLEREN EF	226
19.12.2	ARTIKELGROEP	226
19.12.3	ARTIKEL	226
19.12.4	LEVERANCIER	227
19.12.5	FOODARTIKEL	227
19.12.6	NONFOODARTIKEL.....	227
19.12.7	CONTEXT CLASS	227
19.12.8	APP.CONFIG	227
19.12.9	PROGRAM	228
19.12.10	RESULTAAT IN SQL SERVER MANAGEMENT STUDIO.....	228
20	COLOFON.....	230

1 INLEIDING

1.1 DOELSTELLING

Je leert in deze module het **ADO.NET Entity Framework** gebruiken. Dit helpt je om objecten (in het interne geheugen) in verband te brengen met records (in een relationele database).

We gebruiken in deze cursus **EF** als afkorting voor **Entity Framework**.

1.2 VEREISTE VOORKENNIS

- C# PF
- SQL
- WPF
- ADO

1.3 NODIGE SOFTWARE

- Visual Studio 2017

<https://www.visualstudio.com/downloads/>

- SQL Server/ SQL Server Express.

<https://www.microsoft.com/nl-be/sql-server/sql-server-downloads>

EF werkt samen met veel merken databases (SQL Server, Oracle, DB2, MySQL, ...)

Je gebruikt in deze cursus SQL Server. Je kan een volwaardige SQL Server gebruiken of SQL Server Express (die met Visual Studio is meegeleverd).

- SQL Server Management Studio

<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms>

2 DE OBJECT-RELATIONAL MISMATCH

2.1 ALGEMEEN

Je stelt data in het interne geheugen voor als **objecten**. Dit zijn instanties van classes.

Je stelt dezelfde data in een database voor als **records** in een table.

De manier waarop je data voorstelt als objecten, stemt niet helemaal overeen met de manier waarop je die data voorstelt als records. Dit heet '**the object-relational mismatch**'.

Dit verschil heeft meerdere aspecten, hieronder uitgelegd.

2.2 GRANULARITY

2.2.1 ALGEMEEN

Granularity ('korreligheid') geeft aan in welke mate je data kan opsplitsen in onderdelen.

Een database heeft slechts twee granularity-niveaus: **tabellen en kolommen**.

- Tabellen bevatten kolommen
- Een kolom bevat een enkelvoudige waarde (getallen, datums, tekst) die je niet verder kan opsplitsen

C# heeft op het eerste zicht maar twee granularity-niveaus: **classes en properties**.

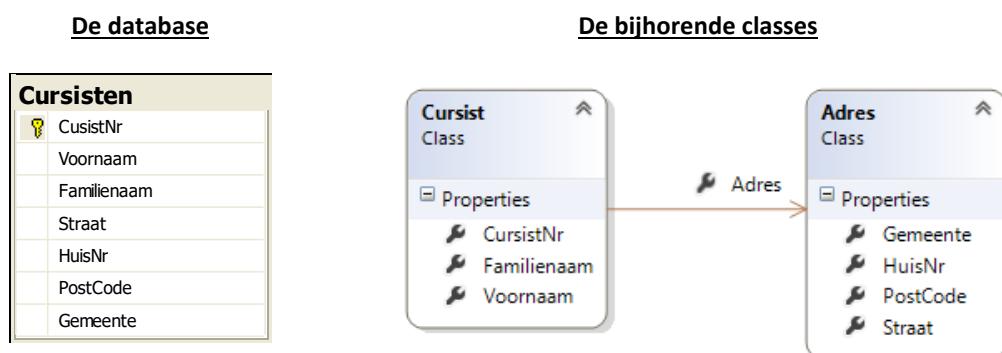
- Classes bevatten properties
- Een property kan een enkelvoudige waarde bevatten (int, decimal, ...). Een property kan echter ook een reference zijn naar een object dat ook properties bevat ...

Op die manier is de granularity van classes oneindig.

2.2.2 VOORBEELD

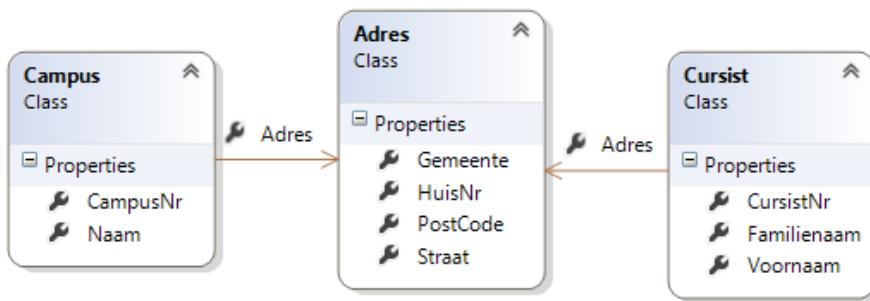
Het gegeven Cursist.

- In de database bevat één table alle cursisten eigenschappen
- Deze eigenschappen zijn in C# verdeeld over twee classes.



- De class **Cursist** bevat een reference naar de class **Adres**.
- De class **Adres** bevat de properties **Gemeente**, **HuisNr**, **Postcode** en **Straat**

Een aparte class **Adres** is handig: je kan die ook gebruiken vanuit andere classes:



Een class kan zelfs meerdere keren verwijzen naar één andere class.

Een class **Klant** verwijst bijvoorbeeld twee keer naar de class **Adres**

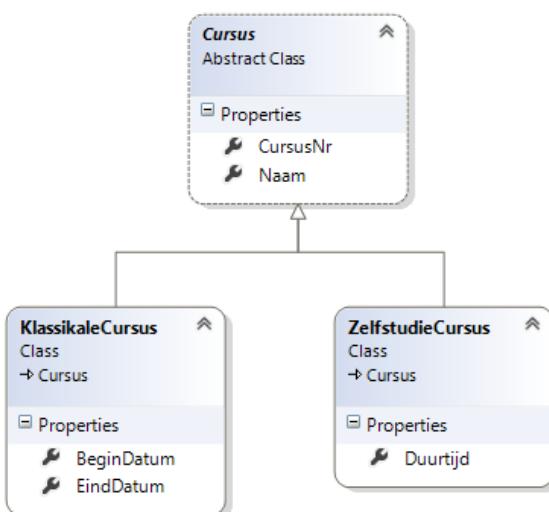
- Een eerste keer voor het **facturatieadres**
- Een tweede keer voor het **leveringsadres**



2.3 INHERITANCE

Inheritance is een essentieel onderdeel van C#.

Voorbeeld: de classes **KlassikaleCursus** en **ZelfstudieCursus** erven van de class **Cursus**:



Inheritance bestaat niet in een database. Je kan er inheritance enkel nabootsen, op drie manieren:

- Table per concrete class

- Table per class hierarchy
- Table per subclass

2.3.1 TABLE PER CONCRETE CLASS

De database bevat één table per niet-abstracte class.

De table bevat kolommen voor alle properties van de class, inclusief de geërfdde properties:

KlassikaleCursussen		ZelfstudieCursussen	
	CursusNr		CursusNr
	Naam		Naam
	BeginDatum		DuurTijd
	EindDatum		

Voorbeelddata:

De table **KlassikaleCursussen**:

CursusNr	Naam	BeginDatum	EindDatum
1	Frans voor beginners	01-09-2007	11-09-2007
2	Frans voor gevorderden	12-09-2007	22-09-2007

De table **ZelfstudieCursussen**:

CursusNr	Naam	DuurTijd
1	Franse correspondentie	5
2	Engelse correspondentie	5

Nadeel van table per concrete class:

Als je een property toevoegt aan de base class, moet je in de database aan meerdere tables een kolom toevoegen.

Als bijvoorbeeld de property **Prijs** toevoegt aan de class **Cursus**, moet je een kolom **Prijs** toevoegen aan de table **KlassikaleCursussen** én aan de table **ZelfstudieCursussen**.

2.3.2 TABLE PER CLASS HIERARCHY

De database bevat één table voor de complete class-inheritance-hiërarchie.

Deze table bevat kolommen voor alle properties van alle classes van de hiërarchie:

Cursussen	
	CursusNr
	Naam
	BeginDatum
	EindDatum
	DuurTijd
	Soort

Je neemt in de table ook een kolom op die aangeeft bij welke subclass een record hoort.

In het voorbeeld is dit de kolom **Soort**. De kolom bevat **K** als het record een **KlassikaleCursus** voorstelt en bevat **Z** als het record een **ZelfstudieCursus** voorstelt.

Voorbeelddata:De table **Cursussen**:

CursusNr	Naam	BeginDatum	EindDatum	DuurTijd	Soort
1	Frans voor beginners	01/09/2007	11/09/2007		K
2	Frans voor gevorderden	12/09/2007	22/09/2007		K
3	Franse correspondentie			5	Z
4	Engels correspondentie			5	Z

Nadeel van table per class hierarchy:

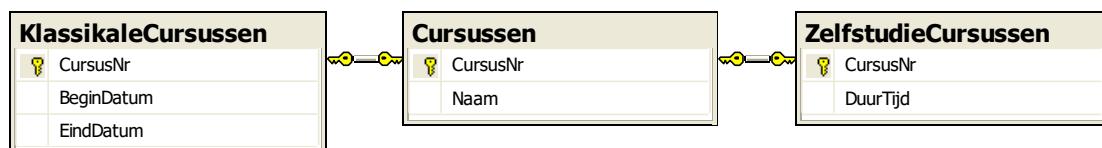
Je kan op de kolommen die horen bij properties van de subclasses geen constraints toepassen, want een constraint geldt voor álle records van de table.

Je kan bijvoorbeeld geen not-null-constraint op de kolom **DuurTijd** plaatsen, want deze kolom vul je enkel in bij een **ZelfstudieCursus**.

2.3.3 TABLE PER SUBCLASS

De database bevat één table per class uit de class inheritance.

- Iedere table bevat enkel kolommen voor de properties die de bijbehorende class niet erft.
- Een table die hoort bij een subclass bevat een primary key die ook een foreign key is naar de primary key van de table die hoort bij de base class:



De kolom **CursusNr** is in de table **Cursussen** een autonumber-kolom, maar niet in de tables **KlassikaleCursussen** en **ZelfstudieCursussen**.

Voorbeelddata:De table **Cursussen**:

CursusNr	Naam
1	Frans voor beginners
2	Frans voor gevorderden
3	Engels voor beginners
4	Engels voor gevorderden
5	Franse correspondentie
6	Engelse correspondentie

De table **KlassikaleCursussen**:

CursusNr	BeginDatum	EindDatum
1	01-09-2007	11-09-2007
2	12-09-2007	22-09-2007
3	01-09-2007	11-09-2007
4	12-09-2007	22-09-2007

De table **ZelfstudieCursussen**:

CursusNr	DuurTijd
5	5
6	5

Nadeel van table per subclass:

Je moet twee tables joinen om de gegevens van **KlassikaleCursussen** of **ZelfstudieCursussen** op te halen. Dit benadeelt de performantie.

Je ziet dat geen enkele van de drie inheritance-nabootsing perfect is.

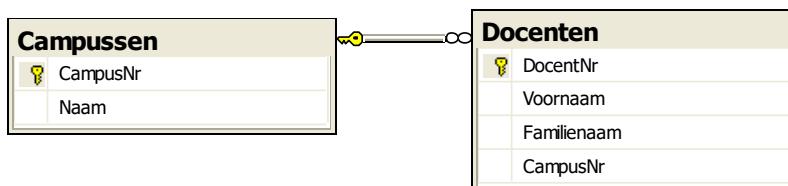
2.4 ASSOCIATIES

2.4.1 ÉÉN-OP-VEEL ASSOCIATIES

Je stelt een één-op-veel-associatie in de database voor met twee tables.

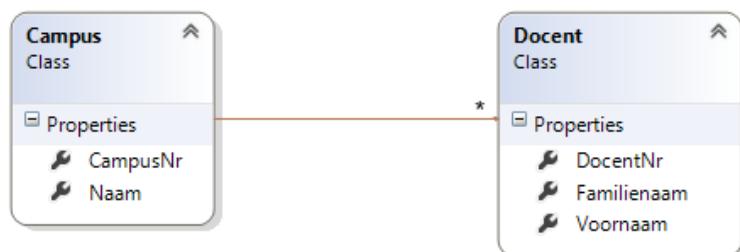
De table aan de veel-zijde van de associatie bevat een foreign-key-kolom, die verwijst naar de primary-key-kolom van de table aan de één-zijde van de associatie.

Voorbeeld:



Je stelt dezelfde één-op-veel-associatie in C# voor met twee classes, die je verbindt met een associatie.

Associaties zijn geen getallen, maar references!



De class aan de veel-zijde van de associatie (**Docent**) bevat één reference met als type de class aan de één-zijde van de associatie (**Campus**). **Docent** bevat dus een reference-variabele van het type **Campus**, waarmee je bijhoudt welke **campus** bij de **docent** hoort:

```
private Campus campusValue;
```

De class aan de één-zijde van de associatie bevat een verzameling references met als type de class aan de veel-zijde van de associatie. **Campus** bevat dus een verzameling reference-variabelen van het type **Docent**, waarmee je bijhoudt welke **docenten** bij de **campus** horen:

```
private List<Docent> docentenValue;
```

2.4.2 VEEL-OP-VEEL ASSOCIATIES

- Je hebt in een relationele database enkel één-op-veel relaties en één-op-één relaties.
- Je hebt een tussentable nodig om een veel-op-veel relatie tussen twee tables voor te stellen.
- De oorspronkelijke tables hebben dan een één-op-veel relatie met deze tussentable.

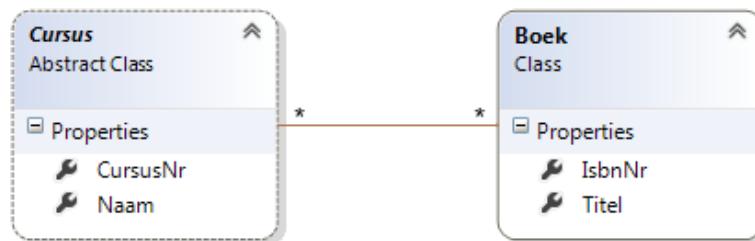
Voorbeeld:

- in een **cursus** worden meerdere **boeken** gebruikt.
- Één **boek** wordt soms gebruikt in meerdere **cursussen**.

De database: Met tussentable **CursussenBoeken**.



Je kan in C# die veel-op-veel associatie tussen **Cursus** en **Boek** uitdrukken zonder tussenclass:



Beide classes bevatten een verzameling references met als type de class aan de andere zijde van de associatie.

- **Cursus** bevat dus een verzameling reference-variabelen van het type **Boek**, waarmee je bijhoudt welke **boeken** bij de **cursus** horen:

```
private List<Boek> boeken;
```

- **Boek** bevat dus een verzameling reference-variabelen van het type **Cursus**, waarmee je bijhoudt welke **cursussen** van het **boek** gebruik maken:

```
private List<Cursus> cursussen;
```

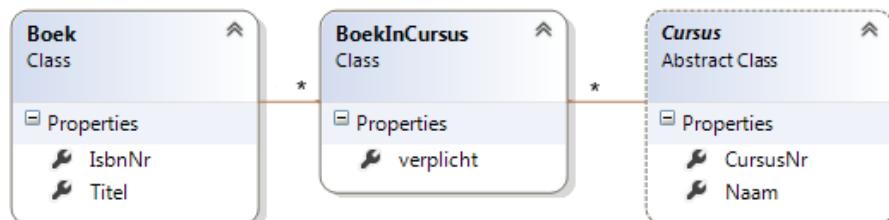
Opmerking:

Van zodra je over de associatie zelf properties bijhoudt, heb je ook een tussenclass nodig (zoals je in de database een tussentable hebt).

Voorbeeld:

de relatie tussen **Boek** en **Cursus**. Een **boek** kan een verplicht te lezen boek of een optioneel te lezen boek zijn in een **cursus**. De table **CursussenBoeken** bevat dan een kolom **verplicht**.

De aangepaste classes:



2.5 NAVIGEREN DOOR ASSOCIATIES

Je navigeert in C# van een object naar een geassocieerd object door de reference-variabele te volgen die de associatie definieert.

Je toont de **titels** van de **boeken** die bij een **cursus** horen als:

```
public void ToonCursusEnBijbehorendeBoeken(Cursus cursus)
{
    Console.WriteLine(cursus.Naam);

    foreach (var boek in cursus.Boeken)
    {
        Console.WriteLine(boek.IsbnNr);
        Console.WriteLine(boek.Titel);
    }
}
```

Je gebruikt in een database een SQL-join om gegevens uit gerelateerde tables op te halen.

Je gebruikt volgend SQL-statement om de gegevens van één **cursus** (bvb. nr. 7), samen met de gebruikte **boeken** binnen die **cursus** op te halen:

```
select      Naam, Boeken.IsbnNr, Titel
from        Cursussen
left join   CursussenBoeken on Cursussen.Cursusnr          = CursussenBoeken.CursusNr
join        Boeken       on CursussenBoeken.IsbnNr          = Boeken.IsbnNr
where       Cursussen.CursusNr = 7
```

Opmerking:

De left outer join tussen de tables **Cursussen** en **CursussenBoeken** geeft je ook informatie over **cursussen** die géén gerelateerde boeken hebben.

2.6 ORM (OBJECT-RELATIONAL MAPPING)

- Je stelt gegevens in C# dus op een andere manier voor dan in de database.
- Je zal ergens een vertaalslag moeten doen tussen deze verschillende visies.
- Deze vertaalslag zelf uitschrijven is niet gemakkelijk en vergt veel tijd.

Object-relational mapping is juist het converteren van de object-georiënteerde visie naar de database-visie.

Een ORM-library helpt je deze conversie te doen.

Entity Framework (EF) is een ORM-library van Microsoft.

Een ORM-library biedt volgende meerwaarden:

- Productiviteit

Zelf de code schrijven die de vertaalslag doet tussen de twee verschillende visies op gegevens vraagt veel code (en dus ook tijd).

- Onderhoudbaarheid

Het databaseschema wijzigt in de tijd. Het class diagram wijzigt in de tijd. Deze visies continu op elkaar afstemmen gaat gemakkelijker met een ORM-library dan zonder ORM-library.

- Databasemerk onafhankelijkheid

Een ORM-library neemt de verschillen tussen databases voor zijn rekening. Een voorbeeld is het automatisch nummeren van de primary-key-kolom bij nieuwe records. Je doet dit bij sommige databases (bvb SQL Server) met autonumber-kolommen, bij andere databases (bvb. Oracle) met sequences.

2.7 EDMX BESTAND VERSUS CODE FIRST

EF heeft twee manieren die verbanden leggen tussen de database tables en de classes:

- in een XML-bestand met de extensie EDMX.
- Met attributen die je tikt in je classes. Dit heet Code First.

Je leert eerst te werken met een EDMX-bestand, daarna met Code First.

Momenteel is Entity Framework 6 de aanbevolen versie. Deze versie zal blijven bestaan naast de nieuwste versie: Entity Framework Core. Deze versie werd oorspronkelijk Entity Framework 7 genoemd. Het is volledig Open Source. EF Core heeft echter (nog) niet alle mogelijkheden van EF 6. Momenteel is het dus aangeraden om EF Core enkel te gebruiken als je een nieuwe applicaties wil maken die gebruik maakt van .NET Core zoals Universal Windows Platform en ASP.NET Core.

In EF Core verdwijnt de Visual Design Tool, EF Designer, en verdwijnt ook de EDMX-aanpak. In nieuwe applicaties gebruik je dus best Code First.

Migrations, een nieuwe feature die het mogelijk maakt om je data-classes te wijzigen en daarna de database te updaten zonder data te verliezen, is ook enkel beschikbaar bij Code First.

Spijtig genoeg laat momenteel alleen het oudere Model First het omgekeerde toe: je data-classes updaten vanuit wijzigingen in de database. Er zal iets gelijkaardigs komen in Code First maar het blijft delicaat aangezien de developer misschien code of commentaar toegevoegd heeft die dan door een update overschreven zou worden.

Omdat je als developer ook met legacy code in aanraking zal komen, is het goed om EDMX te leren kennen. Maar als je nieuwe toepassingen maakt, maak dan gebruik van Code First.

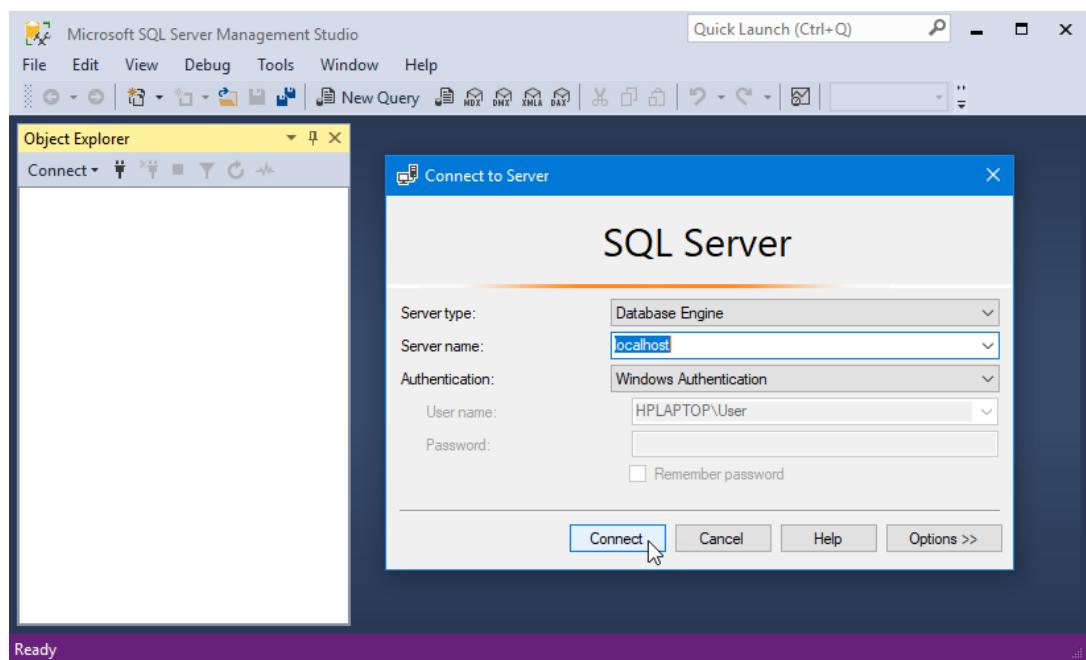
3 DE DATABASE

Je werkt in deze cursus met de database **EF0pleidingen**.

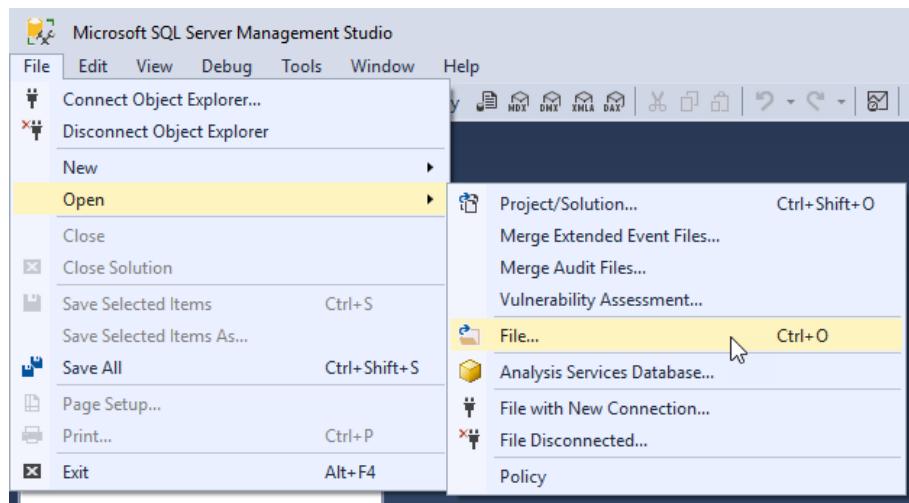
Het script **CreateOpleidingen.sql** uit het takenmateriaal maakt deze database.

Je voert dit script uit:

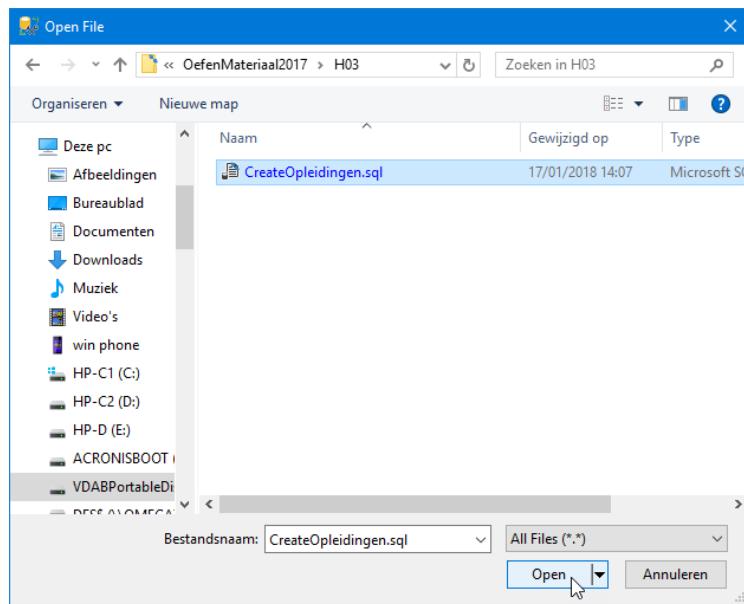
- Je start **SQL Server Management Studio**
- Je tikt in het venster **Connect to Server** `.\sqlexpress`, als je SQL Express gebruikt, of je kiest je volwaardige SQL Server (vb. `localhost (SQLServerEnterprise)` of `.\SQLEXPRESS`) uit de lijst.
- Je laat de keuze bij Authentication op **Windows Authentication**.
- Je kiest **Connect**.



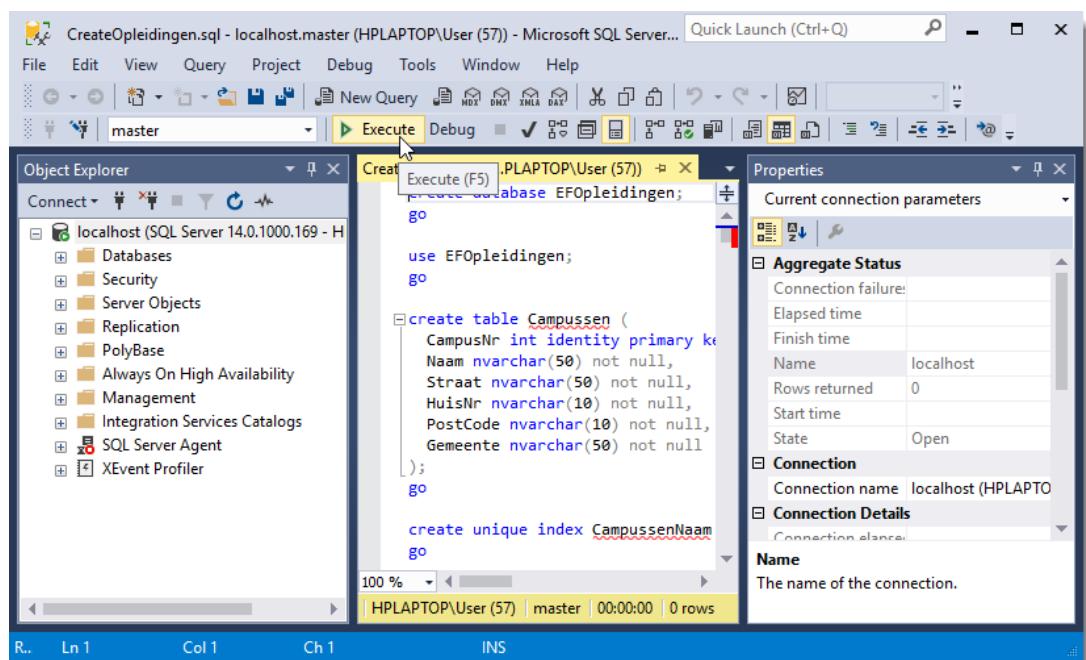
- Je kiest in het menu **File** de opdracht **Open** en de vervolgopdracht **File...** .



- Je selecteert **CreateOpleidingen.sql** en je kiest **Open**.



- Je kiest in de knoppenbalk de opdracht **Execute**.



- Je klikt in het linkerdeel met de rechtermuisknop op de map **Databases** en je kiest **Refresh**.
- Je klapst de map **Databases** open en je ziet de database **EFopleidingen**.

The screenshot shows the SSMS interface. The Object Explorer on the left lists databases, including 'localhost (SQL Server 14.0.1000.169 - HPLAPTOP)' which contains several system databases and user databases like 'BankEF', 'bieren', 'DWConfiguration', 'DW.Diagnostics', 'DWQueue', and 'EFOpleidingen'. The 'EFOpleidingen' database is selected. The 'Tables' node under it shows two tables: 'dbo.Campussen' and 'dbo.Docenten'. The right pane displays a script named 'CreateOpleidingen.sql' containing T-SQL code for creating the database and its objects.

Deze database bevat de tables **Campussen** en **Docenten**:

Campussen		
Column Name	Data Type	Allow Nulls
CampusNr	int	<input type="checkbox"/>
Naam	nvarchar(50)	<input type="checkbox"/>
Straat	nvarchar(50)	<input type="checkbox"/>
HuisNr	nvarchar(10)	<input type="checkbox"/>
PostCode	nvarchar(10)	<input type="checkbox"/>
Gemeente	nvarchar(50)	<input type="checkbox"/>

Docenten		
Column Name	Data Type	Allow Nulls
DocentNr	int	<input type="checkbox"/>
Voornaam	nvarchar(50)	<input type="checkbox"/>
Familienaam	nvarchar(50)	<input type="checkbox"/>
Wedde	decimal(10, 2)	<input type="checkbox"/>
CampusNr	int	<input type="checkbox"/>

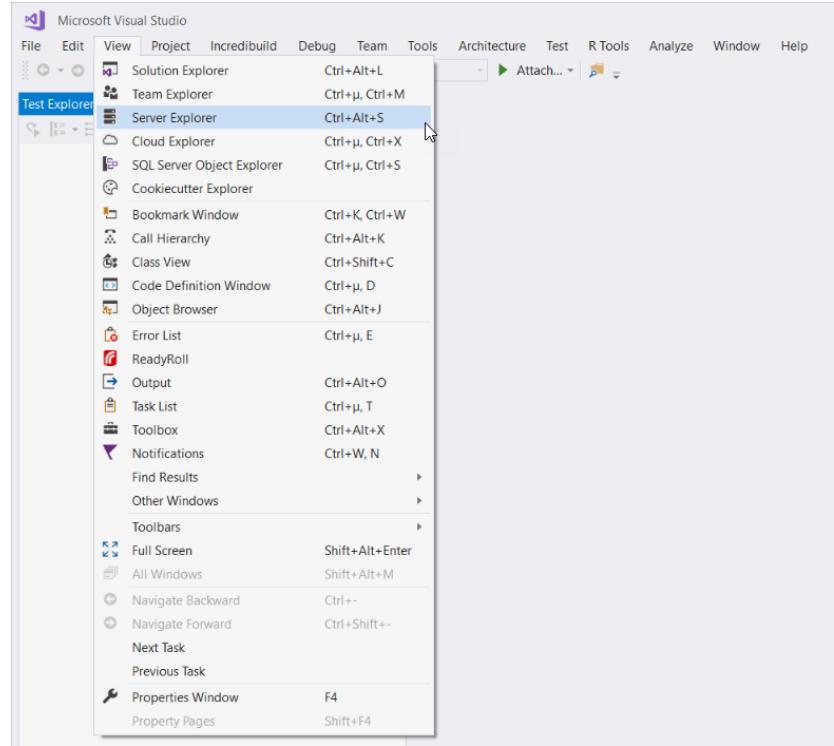
Opmerking:

de primary-key-kolommen van deze tables zijn autonumber-kolommen.

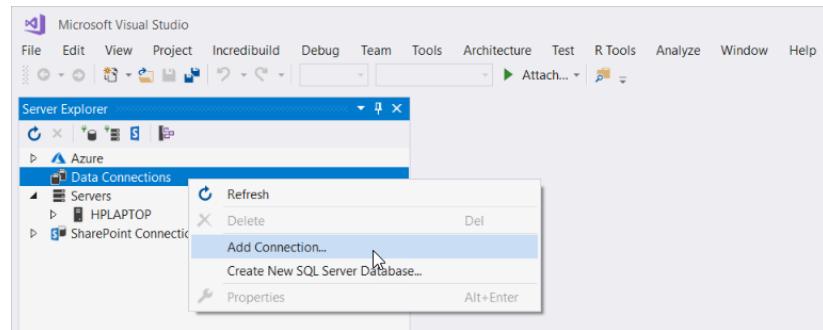
Je zorgt er voor dat je de database waarmee je applicatie zal samenwerken ook ziet vanuit **Visual Studio**.

Dit is belangrijk voor de volgende stappen:

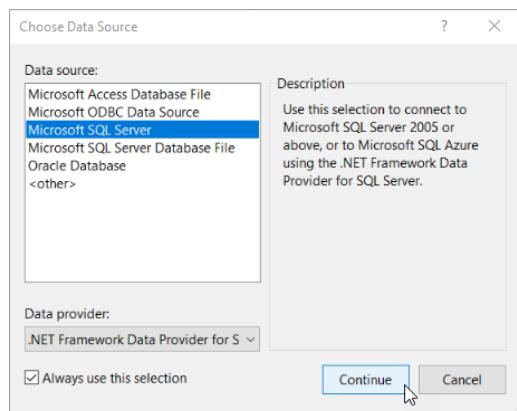
- Je kiest in **Visual Studio** in het menu **view** de opdracht **Server Explorer**.



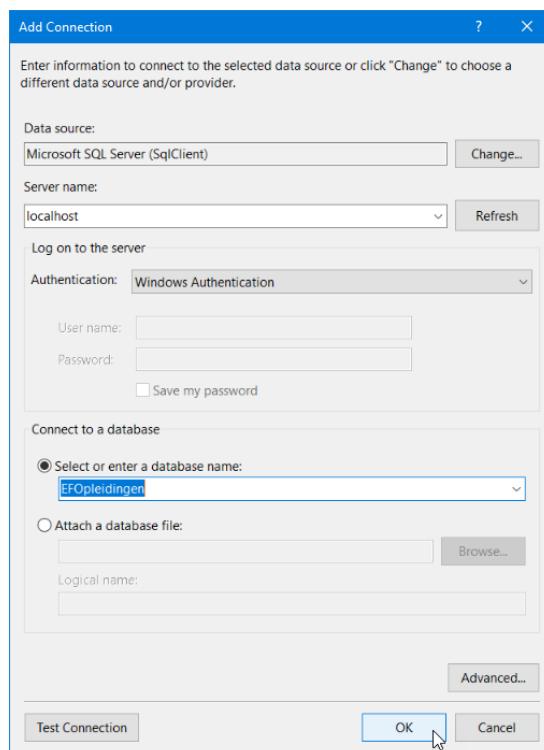
- Je klikt in de **Server Explorer** met de rechtermuisknop op **Data Connections** en je kiest **Add Connection**



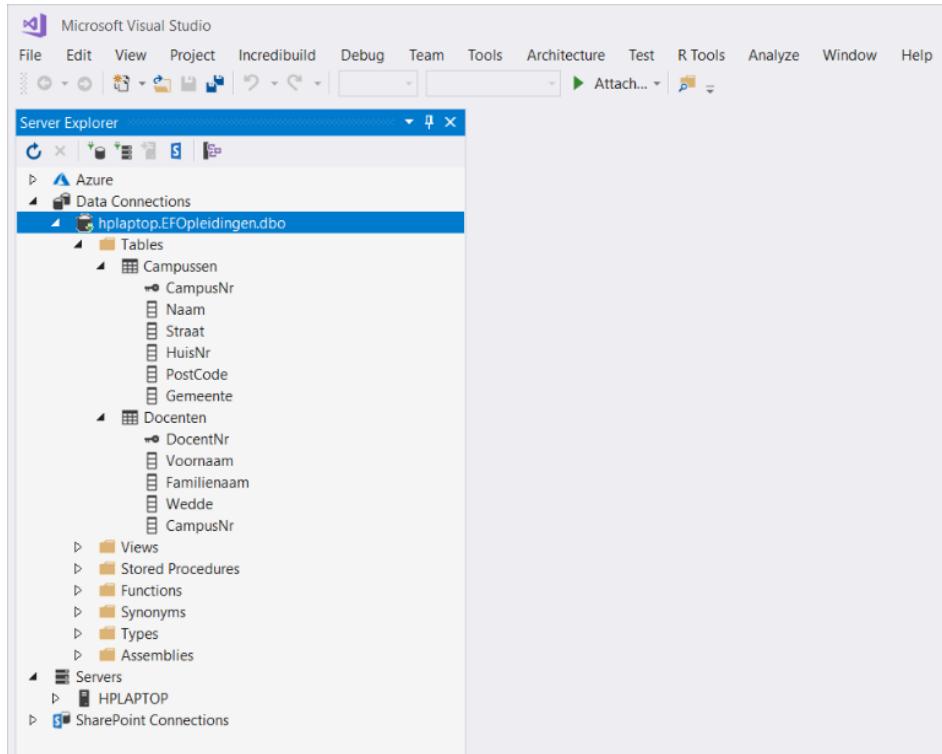
- Als de titel van het venster **Choose Data Source** is, kies je **Microsoft SQL Server** bij **Data Source** en je kiest **Continue**



- Je tikt bij Server name: .\sqlexpress, als je SQL Express gebruikt, of je kiest je volwaardige SQL Server (vb. localhost).
- Je kiest bij Select or enter a database name: EFopleidingen en je kiest **OK**.



- Je hebt nu toegang tot de database **EFopleidingen** in Visual Studio



4 HET ENTITY DATA MODEL (EDM)

4.1 ALGEMEEN

Een entity class is een class die een gegeven uit de werkelijkheid voorstelt.

De classes **Docent** en **Campus** zijn bijvoorbeeld entity classes.

Objecten van entity classes heten entities.

Het **Entity Data Model** beschrijft:

- De entity classes van je applicatie en de verbanden (inheritance, associatie) tussen deze entity classes.
- De structuur van de database die bij je applicatie hoort.
- Het verband tussen de entity classes en de database:
 - Welke entity class hoort bij welke table ?
 - Welke property van een entity class hoort bij welke kolom ?

Het **Entity Data Model** is een **XML**-bestand met de extensie **edmx**.

Je bewerkt dit XML-bestand met een grafische designer van Visual Studio.

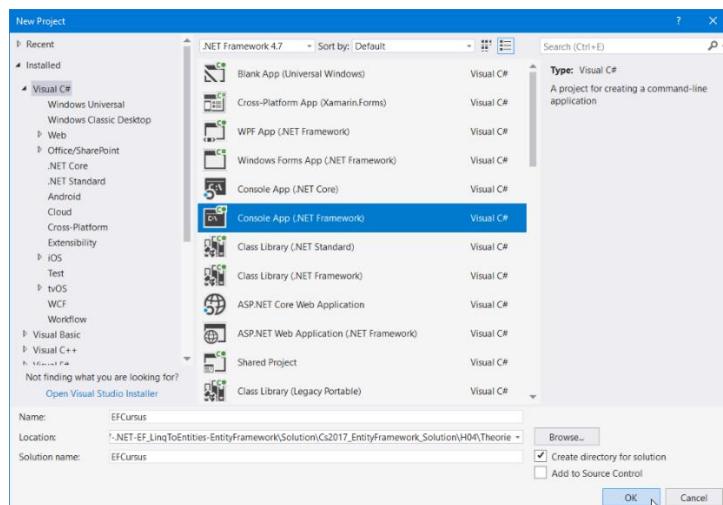
Visual Studio maakt daarna, gebaseerd op dit Entity Data Model:

- De entity classes als C# classes die je vanuit je applicatie aanspreekt.
- Een “**DbContext class**” waarmee je van je applicatie:
 - Records leest uit de database als entity.
 - Records toevoegt gebaseerd op entity.
 - Records wijzigt gebaseerd op gewijzigde entities.
 - Records verwijderd die horen bij entities.

Je leert in dit hoofdstuk het Entity Data Model en de bijbehorende designer kennen.

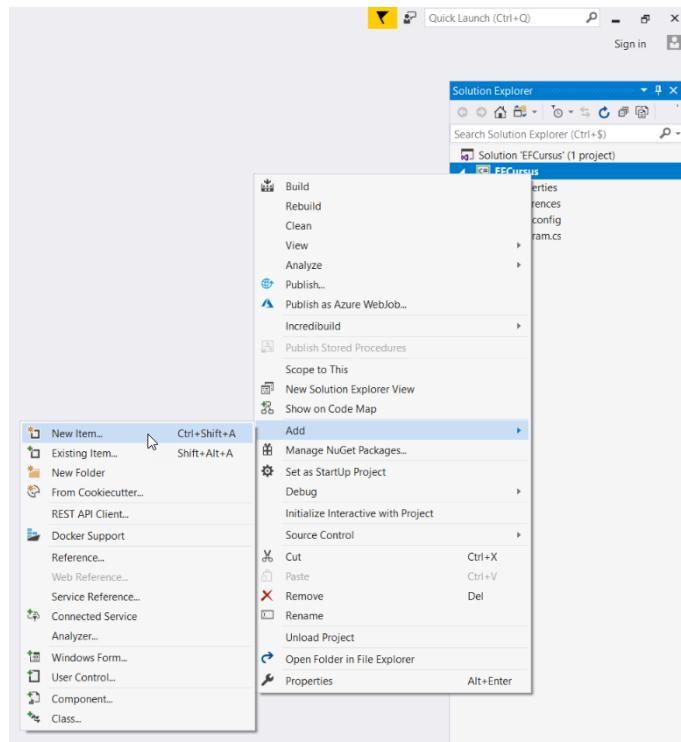
4.2 JE EERSTE EDM

Je maakt in Visual Studio een **Console Application** project met de naam **EFCursus**

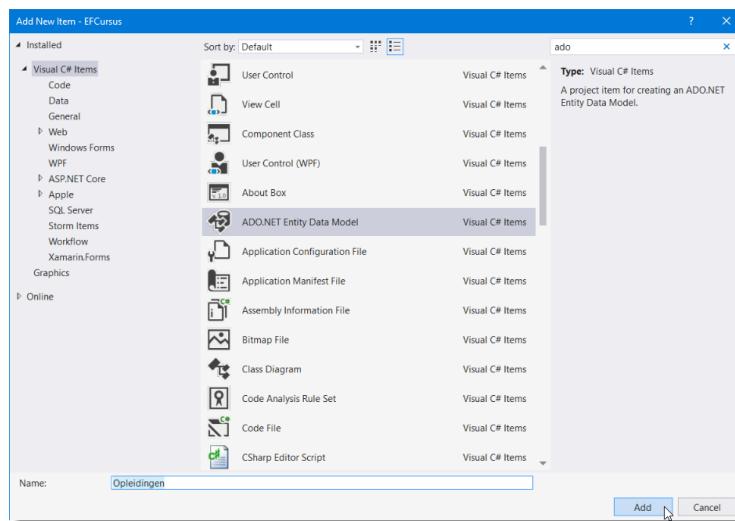


Je voegt aan het project een **Entity Data Model** toe:

- Je klikt in de **Solution Explorer** met de rechtermuisknop op je project (**EFCursus**), je kiest **Add** en daarna **New Item**.



- Je kiest in het middendeel **ADO.NET Entity Data Model**.
- Je tikt bij "Name" **Opleidingen** en je kiest **Add**.



- Je krijgt een vraag: **What should the model contain?**

- **EF Designer from database**

Visual Studio maakt hierbij entity classes die lijken op de tabelstructuren van de bijbehorende database. Zo win je tijd.

Je kan deze entity classes daarna uitbreiden (properties, methods, ...).

- **Empty EF Designer model**

Je definieert bij deze keuze de entity classes vanaf nul. Dit vraagt tijd.

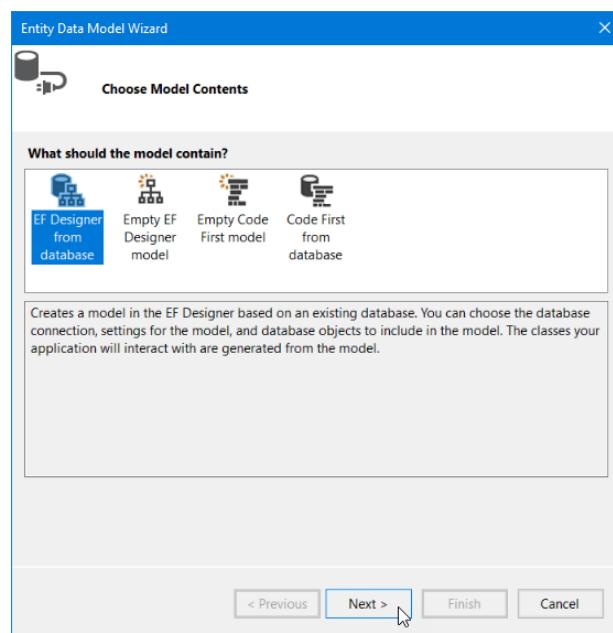
- **Empty Code First Model**

Code First wordt later in de cursus uitgelegd.

- **Code First from database**

Code First wordt later in de cursus uitgelegd

Je kiest **EF Designer from database** en je kiest **Next**.



- Je kiest bij **Which data connection should your application use to connect to the database?** één van de databaseconnecties die gedefinieerd zijn in de Server Explorer.

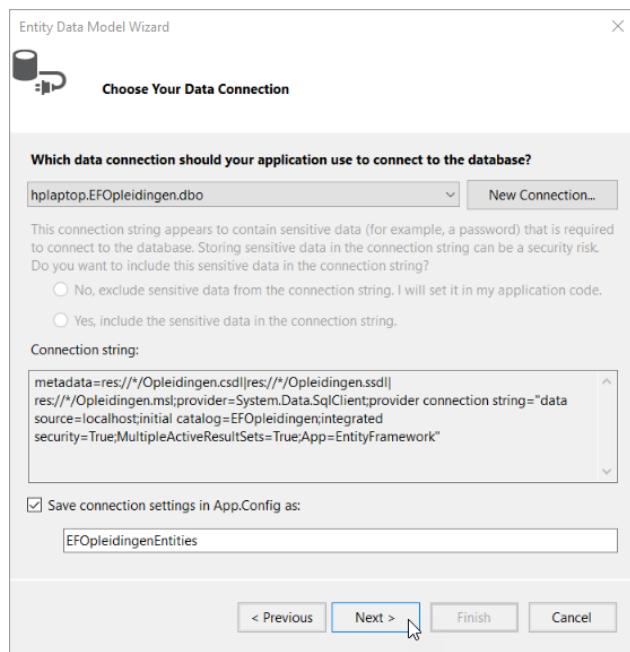
Je kiest in deze cursus de connectie naar de database **EFOpleidingen**.

- Als de vraag **Do you want to include this sensitive data in the connection string?** beschikbaar is, antwoord je **Yes**.
- Je laat het vinkje bij **Save connection settings in App.Config as** staan.

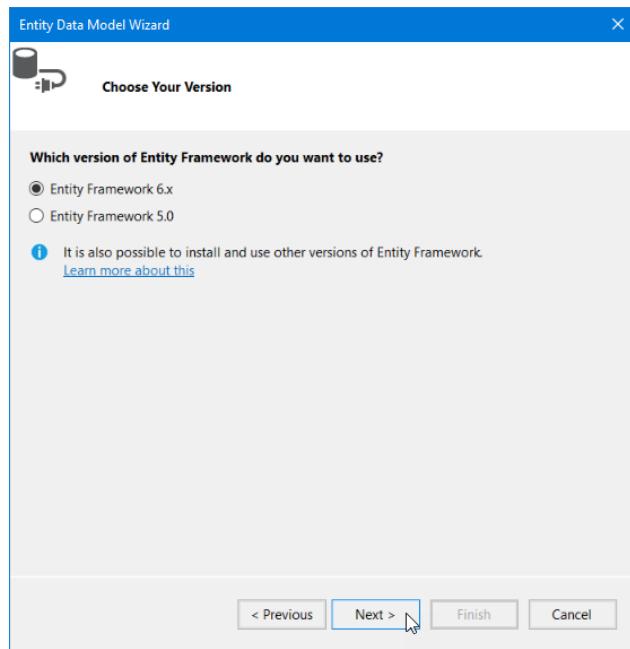
Visual Studio maakt in dit configuratiebestand van je applicatie een onderdeel **EFOpleidingenEntities** waarin de databaseconnectie gedefinieerd is.

Je past dit bestand aan als de databaseconnectie moet wijzigen.

- Je kiest **Next**.



- Je kiest **Entity Framework 6.x** en je kiest **Next**



- Je klapt bij **Which database objects do you want to include in your model?** het onderdeel **Tables** open. Je klapt daarbinnen **dbo** open.

Je plaatst een vinkje bij **Campussen** en **Docenten**.

- Je plaatst geen vinkje bij **Pluralize or singularize generated object names**. Dit is enkel nuttig als de tables in de database Engelse namen hebben. Je kan in dat geval wel een vinkje plaatsen. Visual Studio maakt dan bij een table met een naam in meervoudsvorm (bvb. Clients) een bijbehorende class met een naam in enkelvoudsvorm (bvb. Client).
- Je laat het vinkje staan bij **Include foreign key columns in the model**. Visual Studio stelt dan een foreign-key-kolom op twee manieren voor in de bijbehorende class:

- één keer als een property met als type de class die hoort bij de table waar de foreign key naar verwijst.

(Visual Studio stelt de foreign-key-kolom **CampusNr** in de table **Docenten** voor in de class **Docenten** als een property met als type de class **Campussen**).

Je gebruikt deze property om van een **docent** campusinformatie op te halen (zoals de naam van de **campus**).

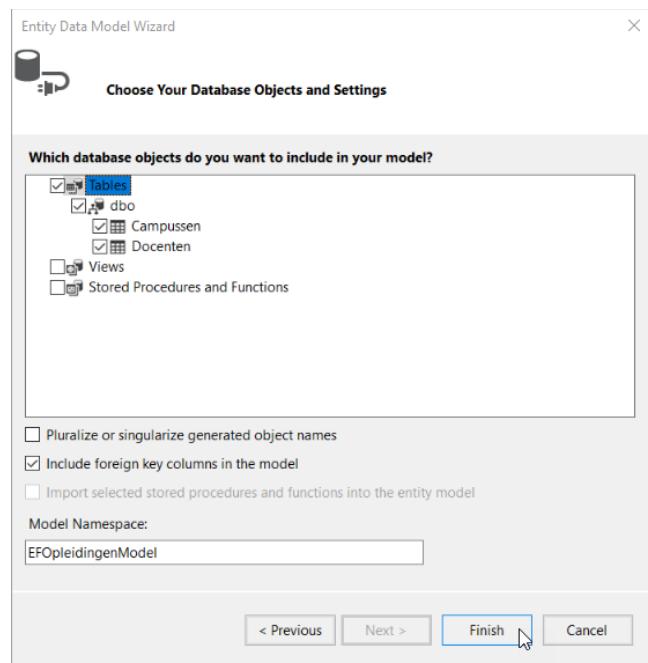
- één keer als een property met als type het .NET-type dat hoort bij het type van de foreign-key-kolom.

(Visual Studio stelt de foreign-key-kolom **CampusNr** in de table **Docenten** in de class **Docenten** ook voor als een property met als type int).

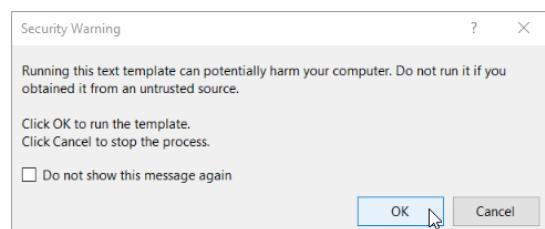
Je gebruikt in je code deze property als je van de **campus** die bij een **docent** hoort enkel het **campusnummer** wil ophalen. Dit ophalen gaat zeer snel, omdat dit **campusnummer** zich op dat moment al in het interne geheugen bevindt, en niet uit het gerelateerde record van de table **Campussen** moet gelezen worden.

Indien je het vinkje wegdoet, maakt Visual Studio enkel de eerste voorstelling van de foreign key (de reference naar de class **Campussen**).

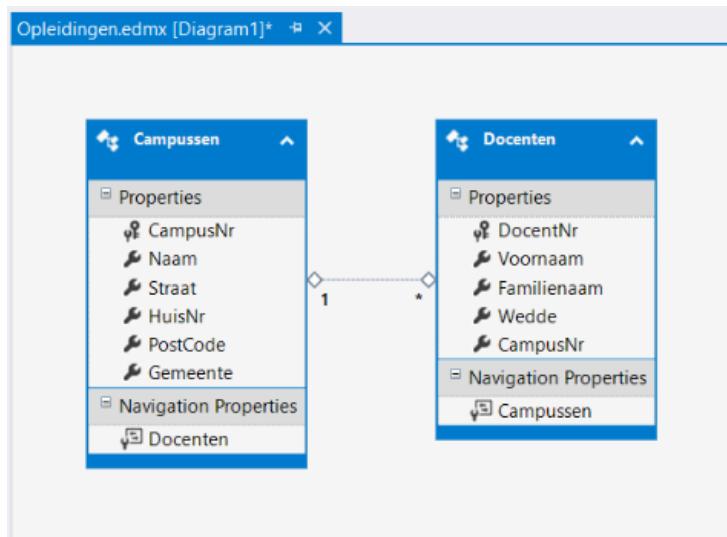
- Je kiest **Finish**. Opgepast ! Hier moet je eventjes geduld hebben.



- Je kiest **OK** bij de **Security warnings**.



Je ziet in de designer van **Opleidingen.edmx** het ontwerp van de entity classes **Campussen** en **Docenten**. Ze lijken op de structuur van de tables **Campussen** en **Docenten** in de database:



Er is een 1-op-veel-associatie $1 \cdots \cdots *$ tussen **Campussen** en **Docenten**. Visual Studio maakte die omdat de database een één-op-veel-relatie bevat tussen de tables **Campussen** en **Docenten**.

Je ziet per entity class twee soorten properties:

- **Gewone properties**

Deze bevatten data die geen references zijn naar andere entity classes.

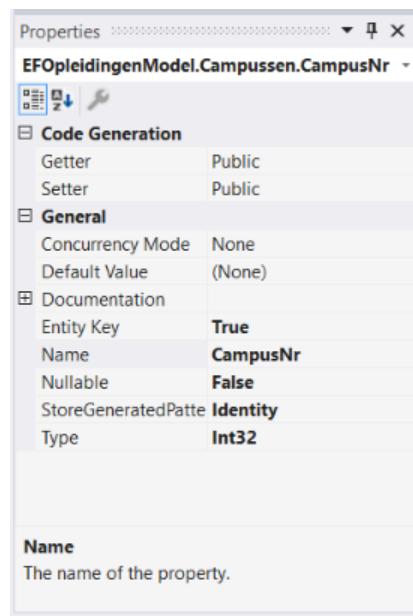
- **Navigation properties**

Dit zijn references naar geassocieerde entity classes.

- De navigation property **Docenten** in de entity class **Campussen** verwijst naar de bijbehorende **Docenten**-objecten.
- De navigation property **Campussen** in de entity class **Docenten** verwijst naar de bijbehorende **Campus**-entity.

Als je een property aanklikt, zie je in het **Properties** venster detailinformatie over die property.

Voorbeeld: de property **CampusNr** van **Campussen**

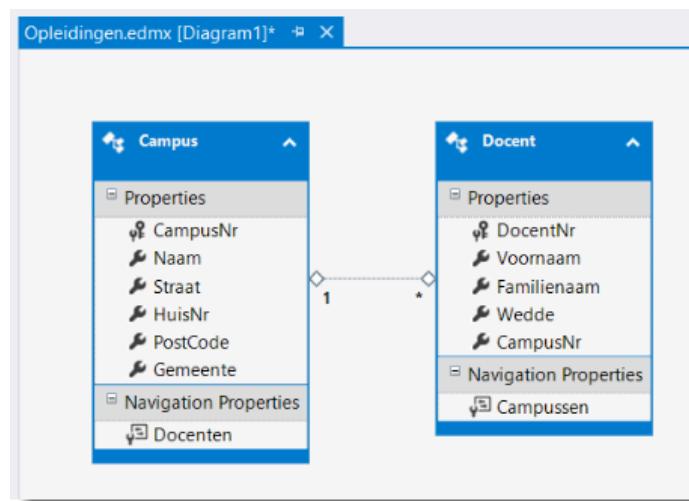


De namen van de entity classes (**Campussen**, **Docenten**) zijn overgenomen uit de database.

De entity class **Campussen** stelt echter één campus voor, geen verzameling campussen en de entity class **Docenten** stelt één docent voor.

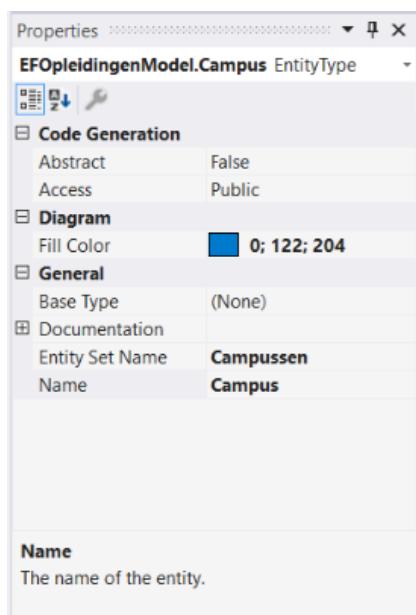
Je wijzigt daarom de namen van deze entity classes:

- Je dubbelklikt in de hoofding van de entity class **Campussen** het woord **Campussen**, je corrigeert naar **Campus** en je drukt **Enter**.
- Je dubbelklikt in de hoofding van de entity class **Docenten** het woord **Docenten**, je corrigeert naar **Docent** en je drukt **Enter**.



Als je de hoofding van een entity class aanklikt, zie je in het **Properties** venster detailinformatie over die entity class.

Voorbeeld: de entity class **Campus**:



Eén van deze properties heet **Entity Set Name**.

Deze naam (**Campus**) staat voor de verzameling met alle entites uit de database.

Het is ook beter de namen van enkele **Navigation Properties** te corrigeren:

- De naam van de Navigation Property **Docenten** in de entity class **Campus** is OK: één **Campus** heeft meerdere **Docenten**.
- De naam van de Navigation Property **Campus** in de entity class **Docent** is niet OK: een **Docent** heeft maar één **Campus**.

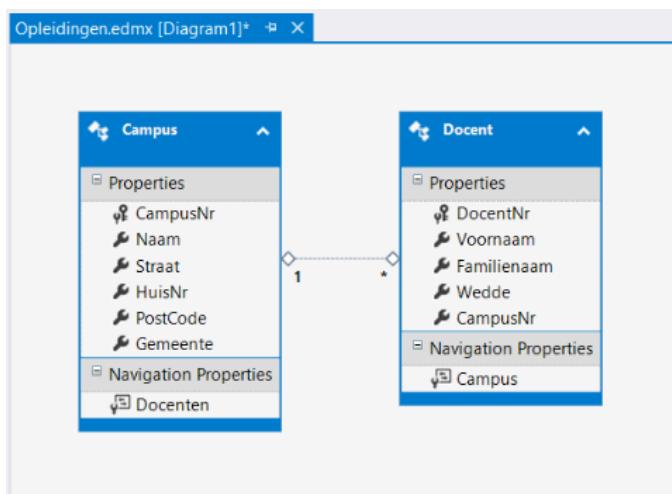
Je wijzigt daarom de naam van deze Navigation Property naar **Campus**:

- Je klikt op de Navigation Property **Campus** in de entity class **Docent**.
- Je klikt nog eens op dezelfde Navigation Property
- Je corrigeert naar **Campus** en je drukt **Enter**.

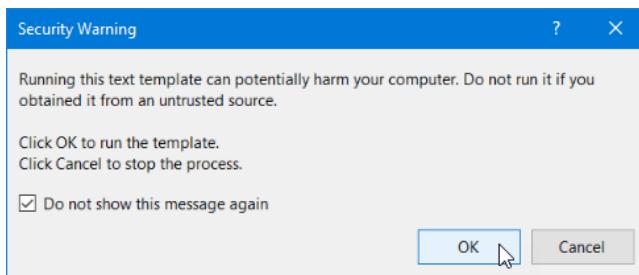
Opmerking:

Het is belangrijk deze naamcorrecties te doen vooraleer je het EDM gebruikt in je code. Als je achteraf de namen in het EDM wijzigt, moet je ook de verwijzingen in je code wijzigen !

Resultaat:



Je slaat het EDM op met de knop in de toolbar. Je krijgt hierbij een security warning, die niet belangrijk is. Je krijgt deze warning iedere keer wanneer je het EDM opslaat. Je plaatst daarom een vinkje bij **Do not show this message again** en je kiest **OK**



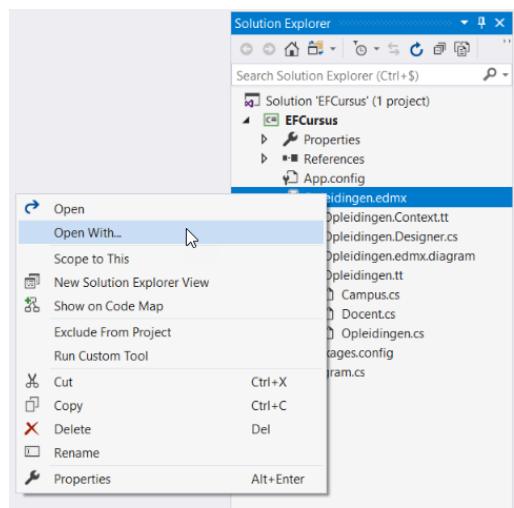
4.3 HET EDM ALS XML-BESTAND

4.3.1 ALGEMEEN

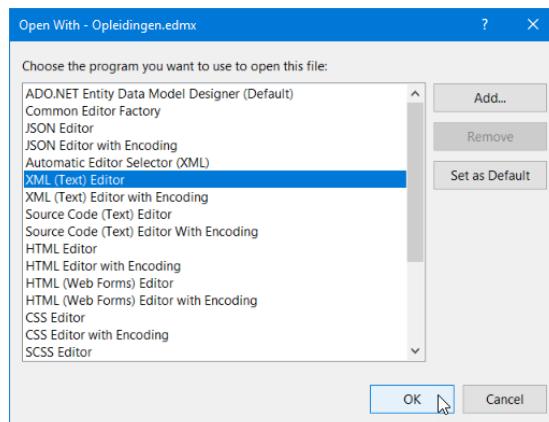
Je bekijkt het EDM tot nu met de grafische designer. Het EDM is in feite een XML-bestand. Je krijgt hier wat inzicht in de opbouw van het XML-bestand.

Je bekijkt het EDM als XML:

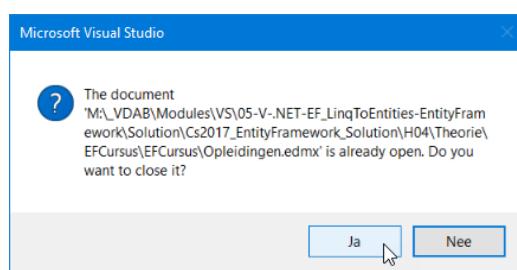
- Je klikt in de **Solution Explorer** met de rechtermuisknop op **Opleidingen.edmx**.
- Je kiest **Open With**.



- Je kiest **XML (Text) Editor**.
- Je kiest **OK**.



- Je kiest **Yes** op de vraag **The document ... is already open. Do you want to close it?**



Je klapt met de onderdelen `<edmx:Runtime>` en `<Designer ...>` dicht.

Dan zie je de grove opbouw van het EDMX-bestand:

```

<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="3.0" xmlns:edmx="http://schemas.microsoft.com/ado/2009/11/edmx">
  <!-- EF Runtime content -->
  <!-- EF Designer content (DO NOT EDIT MANUALLY BELOW HERE) -->
  <Designer xmlns="http://schemas.">...</Designer>
</edmx:Edmx>

```

- Het onderdeel `<edmx:Runtime>` bevat het eigenlijke ontwerp van het EDM. Je leert dit verder in detail kennen.
- Visual Studio houdt in het onderdeel `<Designer ...>` de positie van de entity classes bij in de grafische designer en is niet zo interessant.

Je klapst met het onderdeel `<edmx:Runtime>` open. Je klapst daarin de onderdelen `<edmx:StorageModels>`, `<edmx:ConceptualModels>` en `<edmx:Mappings>` dicht.

Je ziet dan de grove opbouw van het onderdeel `<edmx:Runtime>`:

```

<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="3.0" xmlns:edmx="http://schemas.microsoft.com/ado/2009/11/edmx">
  <!-- EF Runtime content -->
  <edmx:Runtime>
    <!-- SSDL content -->
    <edmx:StorageModels>...</edmx:StorageModels>
    <!-- CSDL content -->
    <edmx:ConceptualModels>...</edmx:ConceptualModels>
    <!-- C-S mapping content -->
    <edmx:Mappings>...</edmx:Mappings>
  </edmx:Runtime>
  <!-- EF Designer content (DO NOT EDIT MANUALLY BELOW HERE) -->
  <Designer xmlns="http://schemas.">...</Designer>
</edmx:Edmx>

```

- StorageModels Bevat de databasestructuur.
- ConceptualModels Bevat de structuur van de entity classes.
- Mappings Bevat de verbanden tussen de entity classes en de database.

4.3.2 STORAGE MODELS

Het onderdeel StorageModels bevat de structuur van iedere table.

De structuur van de table **Campusen**:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <edmx:Edmx Version="3.0" xmlns:edmx="http://schemas.microsoft.com/ado/2009/11/edmx">
3      <!-- EF Runtime content -->
4      <edmx:Runtime>
5          <!-- SSDL content -->
6          <edmx:StorageModels>
7              <Schema Namespace="EF0pleidingenModel.Store" Provider="System.Data.SqlClient" ProviderManifestType="T-SQL">
8                  <EntityType Name="Campussen">
9                      <Key>
10                         <PropertyRef Name="CampusNr" />
11                     </Key>
12                     <Property Name="CampusNr" Type="int" StoreGeneratedPattern="Identity" Nullable="false" />
13                     <Property Name="Naam" Type="nvarchar" MaxLength="50" Nullable="false" />
14                     <Property Name="Straat" Type="nvarchar" MaxLength="50" Nullable="false" />
15                     <Property Name="HuisNr" Type="nvarchar" MaxLength="10" Nullable="false" />
16                     <Property Name="PostCode" Type="nvarchar" MaxLength="10" Nullable="false" />
17                     <Property Name="Gemeente" Type="nvarchar" MaxLength="50" Nullable="false" />
18                 </EntityType>
19             <EntityType Name="Docenten">
20             </EntityType>
21         </Schema>
22     </edmx:StorageModels>
23 </edmx:Edmx>

```

Het onderdeel StorageModels bevat ook informatie over de relaties tussen de tables.

De relatie tussen de tables **Campussen** en **Docenten**:

```

28             </EntityType>
29             <Association Name="FK_Docenten_Campus_398D8EEE">
30                 <End Role="Campussen" Type="Self.Campussen" Multiplicity="1" />
31                 <End Role="Docenten" Type="Self.Docenten" Multiplicity="*" />
32                 <ReferentialConstraint>
33                     <Principal Role="Campussen">
34                         <PropertyRef Name="CampusNr" />
35                     </Principal>
36                     <Dependent Role="Docenten">
37                         <PropertyRef Name="CampusNr" />
38                     </Dependent>
39                 </ReferentialConstraint>
40             </Association>
41         <EntityType Name="EF0pleidingenModel.StoreContainer">

```

4.3.3 CONCEPTUAL MODELS

Het onderdeel ConceptualModels bevat informatie over de opbouw van de entity classes.

De entity class **Campus**:

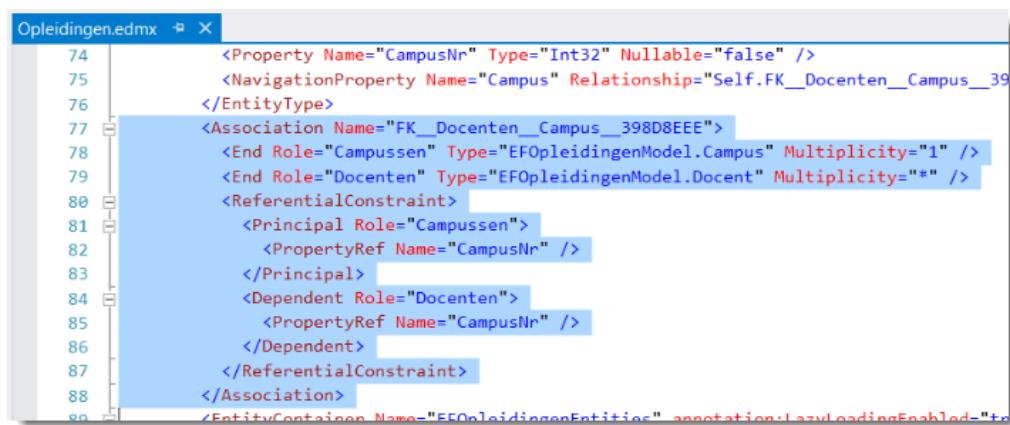
```

50             </edmx:StorageModels>
51             <!-- CSDL content -->
52             <edmx:ConceptualModels>
53                 <Schema Namespace="EF0pleidingenModel" Alias="Self" annotation:UseStrongSpatialTypes="false" xmlns:annotation="http://schemas.microsoft.com/ado/2009/11/edmx">
54                     <EntityType Name="Campus">
55                         <Key>
56                             <PropertyRef Name="CampusNr" />
57                         </Key>
58                         <Property Name="CampusNr" Type="Int32" Nullable="false" annotation:StoreGeneratedPattern="Identity" />
59                         <Property Name="Naam" Type="String" MaxLength="50" FixedLength="false" Unicode="true" Nullable="false" />
60                         <Property Name="Straat" Type="String" MaxLength="50" FixedLength="false" Unicode="true" Nullable="false" />
61                         <Property Name="HuisNr" Type="String" MaxLength="10" FixedLength="false" Unicode="true" Nullable="false" />
62                         <Property Name="PostCode" Type="String" MaxLength="10" FixedLength="false" Unicode="true" Nullable="false" />
63                         <Property Name="Gemeente" Type="String" MaxLength="50" FixedLength="false" Unicode="true" Nullable="false" />
64                         <NavigationProperty Name="Docenten" Relationship="Self.FK_Docenten_Campus_398D8EEE" FromRole="Campussen" ToRole="Docenten" />
65                     </EntityType>
66                 <EntityType Name="Docent" />
67             </Schema>
68         </edmx:ConceptualModels>
69     </edmx:Edmx>

```

Het onderdeel ConceptualModels bevat ook informatie over de associaties tussen entity classes.

De associatie tussen de entites **Campus** en **Docent**:



```

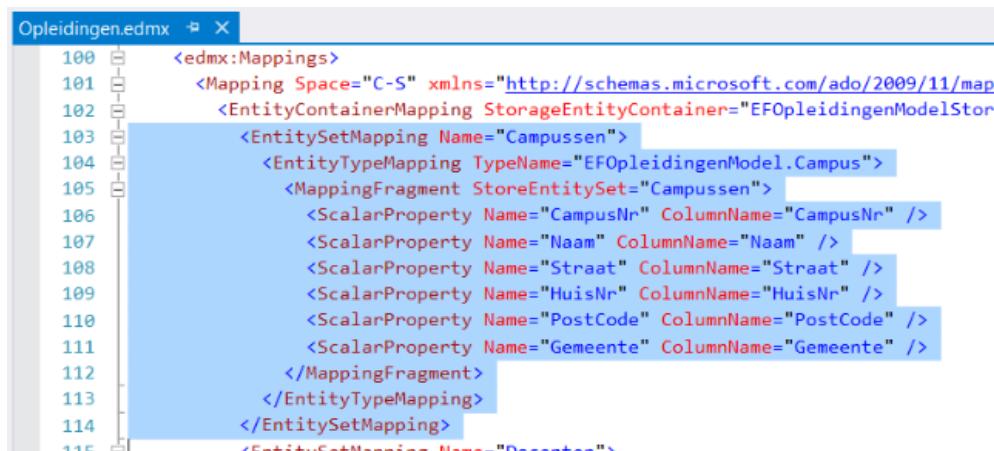
<Property Name="CampusNr" Type="Int32" Nullable="false" />
<NavigationProperty Name="Campus" Relationship="Self.FK_Docenten_Campus_398D8EEE" />
</EntityType>
<Association Name="FK_Docenten_Campus_398D8EEE">
<End Role="Campussen" Type="EFOPleidingenModel.Campus" Multiplicity="1" />
<End Role="Docenten" Type="EFOPleidingenModel.Docent" Multiplicity="*" />
<ReferentialConstraint>
<Principal Role="Campussen">
<PropertyRef Name="CampusNr" />
</Principal>
<Dependent Role="Docenten">
<PropertyRef Name="CampusNr" />
</Dependent>
</ReferentialConstraint>
</Association>
<EntityContainer Name="EFOPleidingenEntities" annotation:LazyLoadingEnabled="true" />

```

4.3.4 MAPPINGS

Het onderdeel **Mappings** bevat informatie over welke table bij welke entity class hoort.

Het verband tussen de table **Campussen** en de entity class **Campus**:



```

<edmx:Mappings>
<Mapping Space="C-S" xmlns="http://schemas.microsoft.com/ado/2009/11/mapping/schemas">
<EntityContainerMapping StorageEntityContainer="EFOPleidingenModelStore">
<EntityTypeMapping TypeName="EFOPleidingenModel.Campus">
<EntitySetMapping Name="Campussen">
<EntityTypeMapping TypeName="EFOPleidingenModel.Campus">
<MappingFragment StoreEntitySet="Campussen">
<ScalarProperty Name="CampusNr" ColumnName="CampusNr" />
<ScalarProperty Name="Naam" ColumnName="Naam" />
<ScalarProperty Name="Straat" ColumnName="Straat" />
<ScalarProperty Name="HuisNr" ColumnName="HuisNr" />
<ScalarProperty Name="PostCode" ColumnName="PostCode" />
<ScalarProperty Name="Gemeente" ColumnName="Gemeente" />
</MappingFragment>
</EntityTypeMapping>
</EntitySetMapping>
</EntityContainerMapping>
</Mapping>

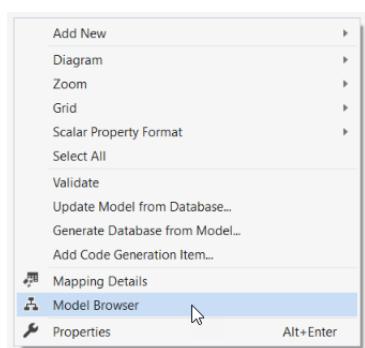
```

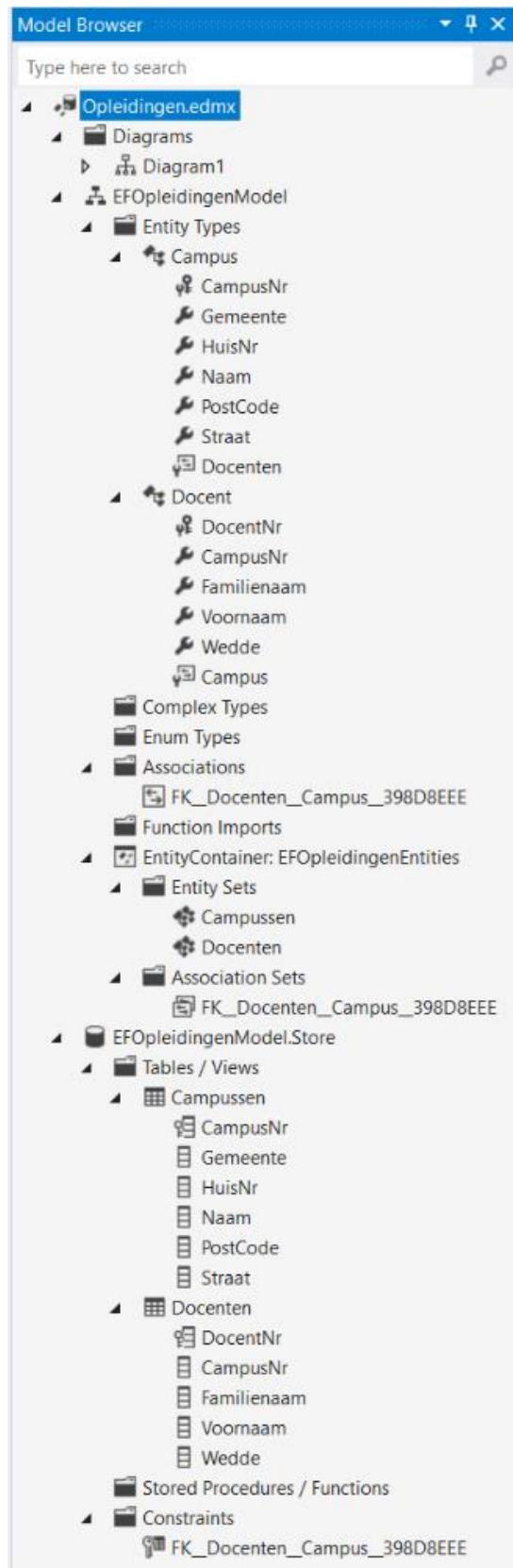
Je mag het venster met deze XML sluiten en **Opleidingen.edmx** in de Solution Explorer dubbel klikken. Visual Studio opent dit bestand dan terug in de designer.

4.4 DE MODEL BROWSER

De Model Browser toont de opbouw van het EDM op nog een andere manier.

Je ziet de Model Browser door in de achtergrond van het EDM te klikken met de rechtermuisknop en **Model Browser** te kiezen:





Je ziet bij **EFOpleidingenModel** de classes die het EDM aanmaakt:

- **Campus** (entity class)
- **Docent** (entity class)

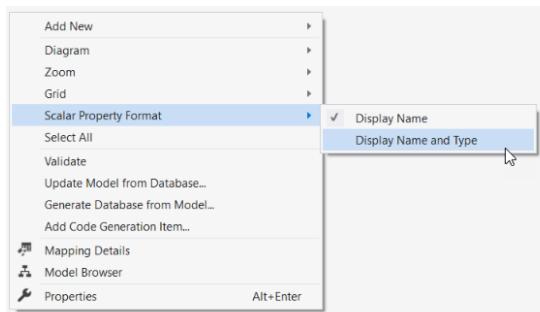
Je ziet bij **EFOpleidingenModel.Store** de database tables die bij het EDM horen:

- **Campussen**
- **Docenten**

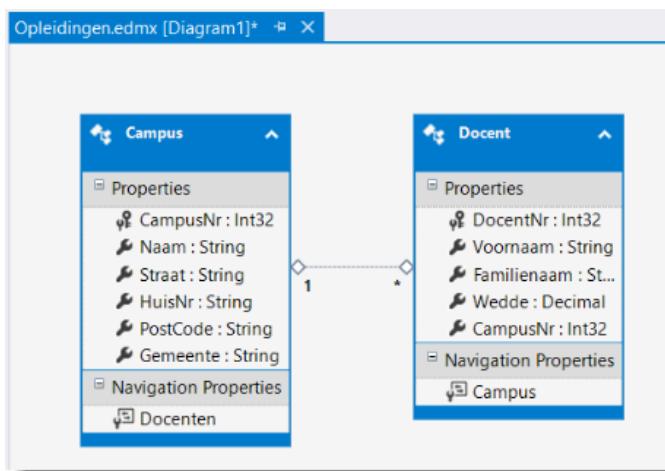
4.5 EXTRA TECHNIEKEN IN DE GRAFISCHE DESIGNER

Je kan in die grafische designer ook de types van de entity properties zien:

- Je klikt in het achtergrond van de designer met de rechtermuisknop.
- Je kiest **Scalar Property Format** en daarna **Display Name and Type**.



Resultaat:

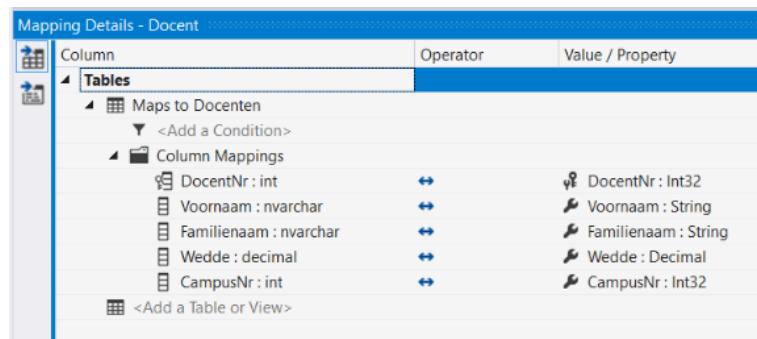


Je kan ook de mapping details zien:

welke table hoort bij welke entity class en welke kolom binnen die table hoort bij één property van de entity class:

- Je klikt met de rechtermuisknop ergens in de entity class **Docent**.
- Je kiest **Table Mapping**.

Resultaat:



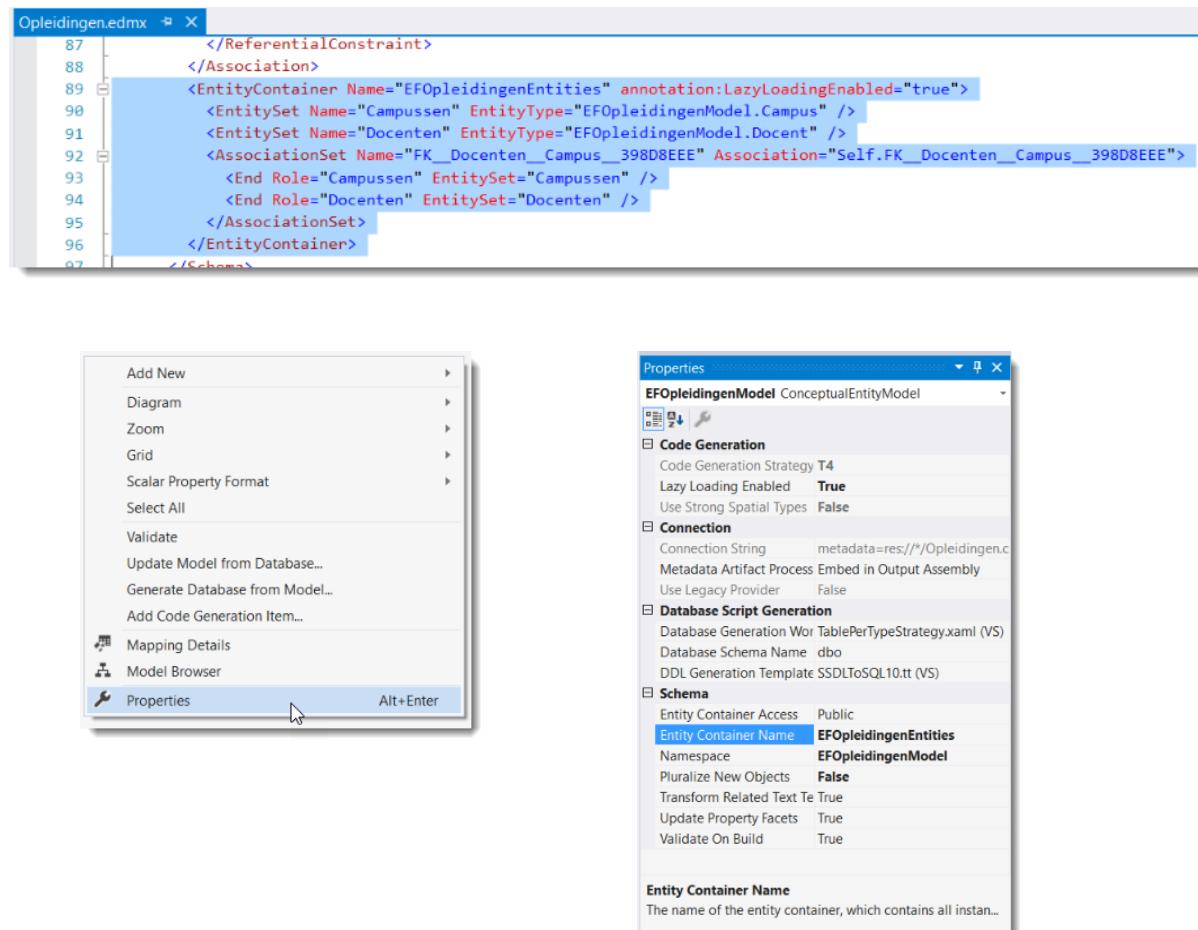
4.6 DE DBCONTEXT CLASS

Visual Studio maakt per **Entity Data Model** (in ons geval **Opleidingen.edmx**) één class, die erft van **DbContext**.

Je gebruikt deze class om vanuit je applicatie de database aan te spreken.

De property **Entity Container Name** van het EDM bevat de naam van deze class.

Je ziet de properties van het EDM als je in de achtergrond van het EDM klikt en daarna in het properties-venster kijkt. Bij ons is de naam **EFOpleidingenEntities**.



Een **DbContext** class implementeert de interface **IDisposable**.

Je moet een **DbContext**-object dus 'opkuisen' na gebruik.

Indien je het **DbContext**-object aanmaakt met het sleutelwoord **using**, gebeurt deze opkuis automatisch:

```

using (var entities = new EFOpleidingenEntities())
{
    // Je doet hier op de variabele entities één of meerdere bewerkingen
}

```

Intern gebruikt een **DbContext** databaseconnecties.

Bij de opkuis van de `DbContext` sluit `.NET` deze connecties. Het is belangrijk de `DbContext` niet langer dan noodzakelijk levend te houden, zodat .net connecties vlot kan sluiten.

4.7 ENTITY CLASSES UITBREIDEN MET PROPERTIES EN METHODS

Je kan de entity classes uitbreiden met extra properties en methods.

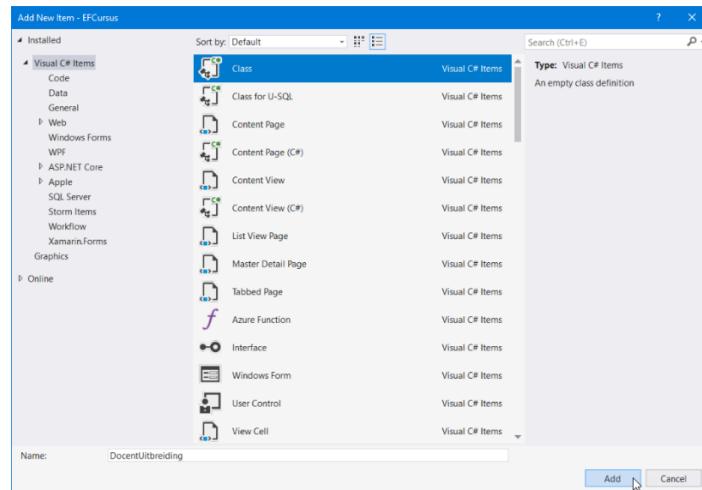
De entity classes die het EDM heeft aangemaakt zijn partial classes. Jij kan een source file toevoegen aan het project. Je beschrijft in die source file dezelfde class ook als partial class.

Je voegt in deze source file properties en of methods toe aan de entity class.

Voorbeeld:

Je voegt een readonly property `Naam` en een method `Opslag` toe aan de class `Docent`:

- Je kiest in het menu `project` de opdracht `Add Class`
- Je tikt `DocentUitbreiding` bij Name en je kiest `Add`



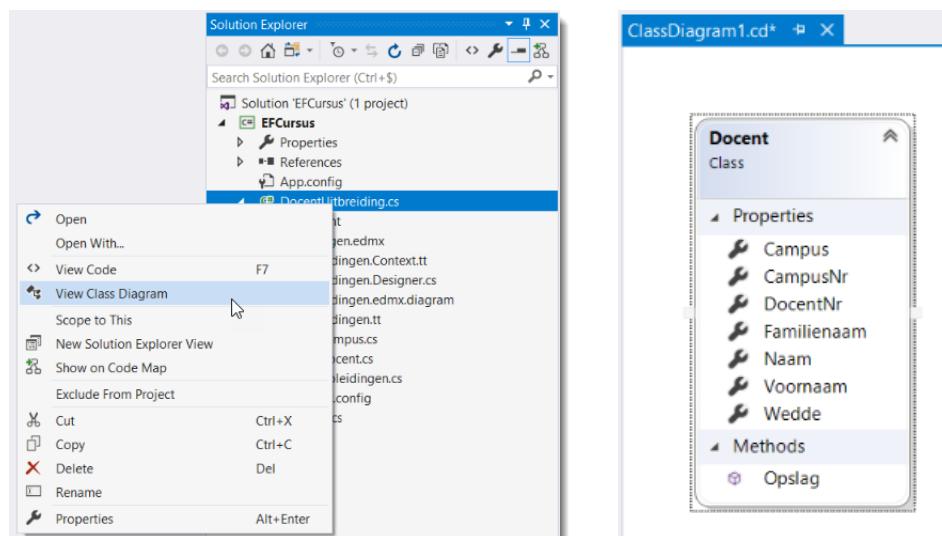
- Je wijzigt deze source file:

```
namespace EFCursus
{
    public partial class Docent
    {
        public string Naam
        {
            get
            {
                return Voornaam + ' ' + Familiennaam;
            }
        }

        public void Opslag(decimal bedrag)
        {
            Wedde += bedrag;
        }
    }
}
```

Je kan zien dat de class `Docent` de extra property en de extra method bevat:

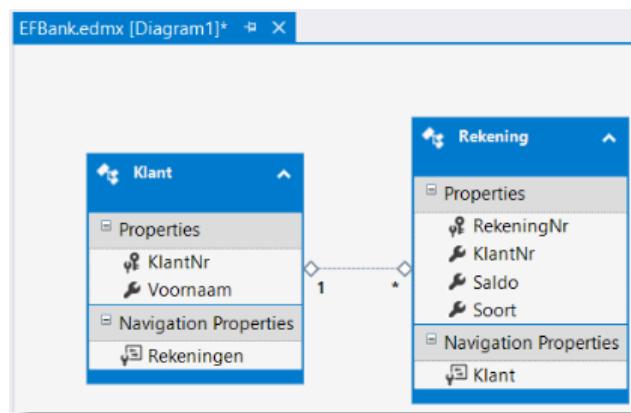
- Je klikt in de Solution Explorer met de rechtermuisknop op `DocentUitbreiding.cs`.
- Je kiest `View Class Diagram`.



4.8 TAAK 01 : EFBANK MAKEN

Het script **CreateBank.sql** maakt de database **EFBank**. Je voert dit script uit.

De database bevat twee tables:



De kolom **Soort** van de table **Rekeningen** bevat:

- **S** als het een spaarrekening betreft
- **Z** als het een zichtrekening betreft

Je maakt in Visual Studio een console-applicatie.

Je voegt aan deze console-applicatie een EDM toe met de entity classes:

- **Klant** (die hoort bij de table **Klanten**) en
- **Rekening** (die hoort bij de table **Rekeningen**).

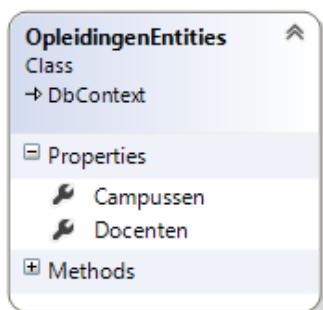
Je maakt in de class **Rekening** een method **Storten**.

- Deze method heeft een parameter **Bedrag**.
- Deze method verhoogt het **Saldo** met dit **Bedrag**.

5 QUERIES

5.1 ALGEMEEN

De `DbContext` class bevat per entity class een property waarvan de naam gelijk is aan de `Entity Set Name` property van die entity class.



Onze `DbContext` class `EFopleidingenEntities` bevat de properties:

- **Campussen**
Je spreekt met die property de records uit de table `campusen` aan.
- **Docenten**
Je spreekt met die property de records uit de table `docenten` aan.

5.1.1 EEN FOREACH ITERATIE

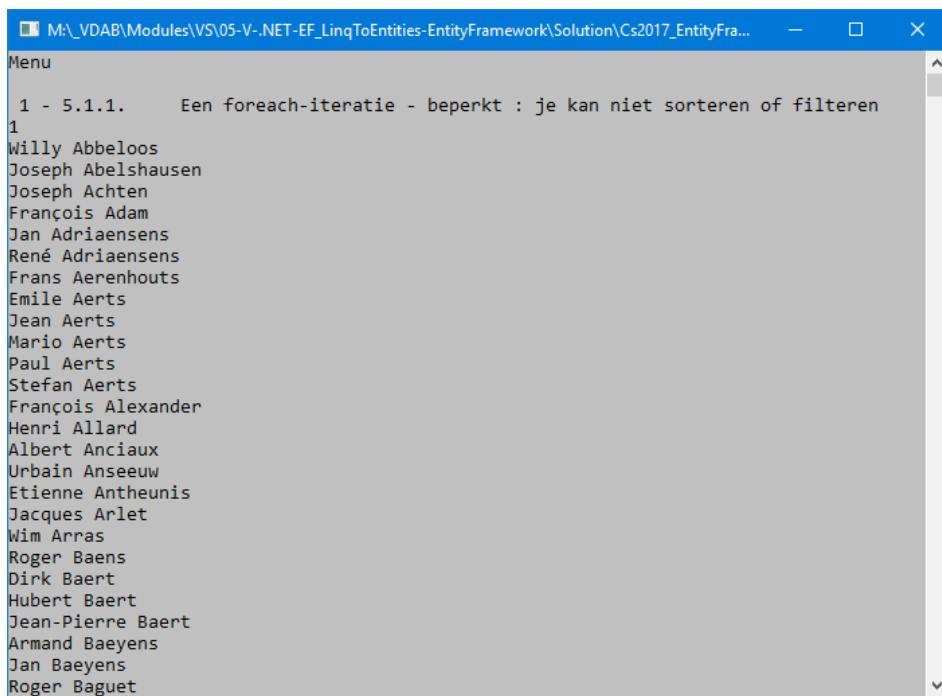
Als je op de property `Docenten` van `EFopleidingenEntities` een foreach-iteratie uitvoert, doet EF volgende stappen:

- Het stuurt een SQL-select-statement naar de database om álle records uit de bijbehorende table (`Docenten`) op te vragen.
- Het maakt van ieder record een `Docent`-entity.
- Het verzamelt deze entities in de property `Docenten` van het `EFopleidingenEntities`-object.

Je itereert dus met deze foreach over alle `docenten` uit de database.

Je kan dit uitproberen met volgende code in de method `Main` van `Program.cs`:

```
using (var entities = new EFopleidingenEntities())
{
    foreach (var docent in entities.Docenten)
    {
        Console.WriteLine(docent.Naam);
    }
}
```



The screenshot shows a Windows application window with a title bar 'M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs2017_EntityFra...' and a menu bar 'Menu'. The main content area contains a list of names, each preceded by a number from 1 to 10. The names listed are: 1 - 5.1.1. Een foreach-iteratie - beperkt : je kan niet sorteren of filteren, followed by 10 names: Willy Abbeloos, Joseph Abelhausen, Joseph Achten, François Adam, Jan Adriaensens, René Adriaensens, Frans Aerenhouts, Emile Aerts, Jean Aerts, Mario Aerts, Paul Aerts, Stefan Aerts, François Alexander, Henri Allard, Albert Anciaux, Urbain Anseeuw, Etienne Antheunis, Jacques Arlet, Wim Arras, Roger Baens, Dirk Baert, Hubert Baert, Jean-Pierre Baert, Armand Baeyens, Jan Baeyens, Roger Baguet.

Deze manier van records lezen is beperkt:

- je kan niet sorteren of filteren (bvb. enkel de **docenten** met een wedde vanaf 2000).

5.1.2 EEN LINQ-QUERY

Als je een LINQ-query uitvoert op de property **Docenten** van **EFOpleidingenEntities**, doet EF volgende stappen:

- Het vertaalt de LINQ-query naar een SQL-select-statement.
- Het stuurt dit SQL-statement naar de database om records uit de bijbehorende table (**Docenten**) op te vragen.
- Het maakt van ieder gevonden record een **Docent**-entity.
- Het verzamelt deze **Docent**-entities in één verzameling.
Deze verzameling, van het type **IQueryable<Docent>**, is het resultaat van de LINQ-query.
- Je kan met een foreach itereren over deze verzameling.

Je kan dit uitproberen in de method **Main** van **Program.cs**:

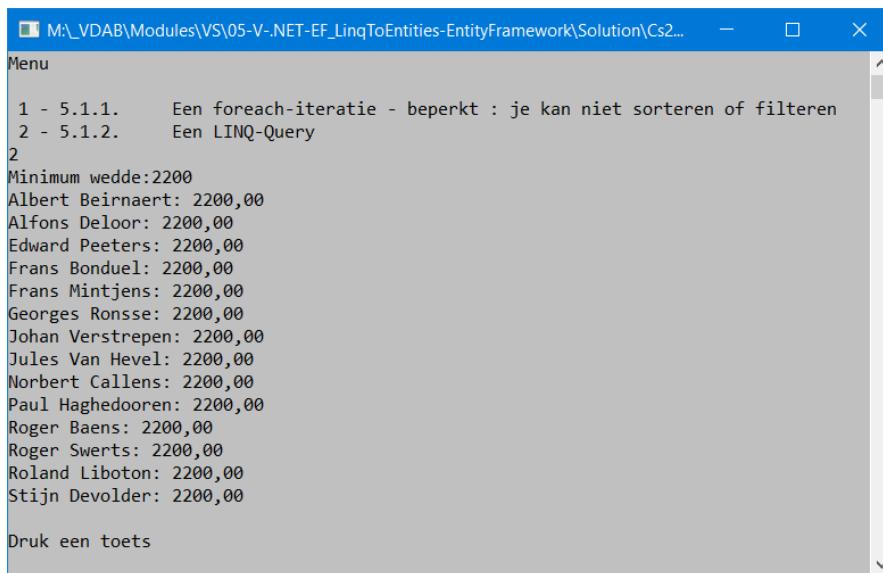
```
.....
using System.Linq;
.....

Console.WriteLine("Minimum wedde:");

decimal minWedde;

if (decimal.TryParse(Console.ReadLine(), out minWedde))
{
    using (var entities = new EFOpleidingenEntities())
    {
        var query =      from docent in entities.Docenten
                        where docent.Wedde >= minWedde
                        orderby docent.Voornaam, docent.Familienaam
                        select docent;
```

```
        foreach (var docent in query)
    {
        Console.WriteLine("{0}: {1}", docent.Naam, docent.Wedde);
    }
}
else
{
    Console.WriteLine("Tik een getal");
}
```



The screenshot shows a Windows application window titled 'M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs2...'. The window contains a menu with options 1 - 5.1.1. and 2 - 5.1.2. Below the menu, there is a list of names and their bet amounts. At the bottom of the window, there is a button labeled 'Druk een toets'.

Naam	Wedde
Albert Beirnaert	2200,00
Alfons Deloor	2200,00
Edward Peeters	2200,00
Frans Bonduel	2200,00
Frans Mintjens	2200,00
Georges Ronsse	2200,00
Johan Verstrepen	2200,00
Jules Van Hevel	2200,00
Norbert Callens	2200,00
Paul Hagedooren	2200,00
Roger Baens	2200,00
Roger Swerts	2200,00
Roland Liboton	2200,00
Stijn Devolder	2200,00

Je ziet verder in de cursus meer gespecialiseerde LINQ-queries.

5.1.3 QUERY-METHODS

In plaats van een query te definiëren als een LINQ-query, kan je dezelfde query ook definiëren met query-methods (bv. de method **Where** en de method **OrderBy**).

Je probeert dit uit door de opdracht

```
var query
```

(over de volledige drie regels) te vervangen door:

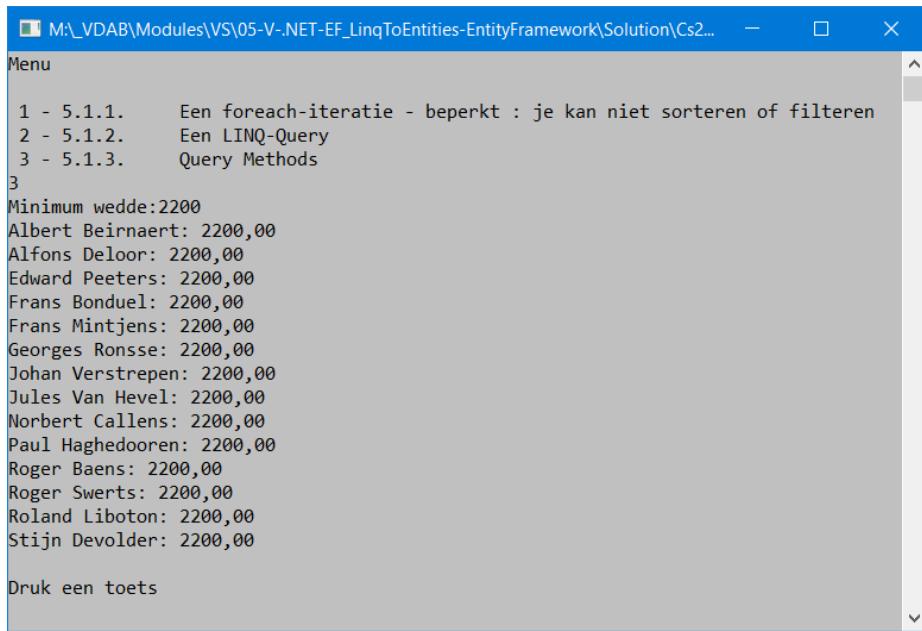
```
Console.Write("Minimum wedde:");

decimal minWedde;

if (decimal.TryParse(Console.ReadLine(), out minWedde))
{
    using (var entities = new EFopleidingenEntities())
    {
        var query = entities.Docenten
            .Where(docent => docent.Wedde >= minWedde)           // (1)
            .OrderBy(docent => docent.Voornaam)                     // (2)
            .ThenBy(docent => docent.Familienaam);                  // (3)

        foreach (var docent in query)
        {
            Console.WriteLine("{0}: {1}", docent.Naam, docent.Wedde);
        }
    }
}
```

```
else
{
    Console.WriteLine("Tik een getal");
}
```



The screenshot shows a Windows command-line interface window. The title bar reads 'M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs2...'. The window contains a menu with items 1 - 5.1.1 through 3. Below the menu, a list of names and their minimum bet amounts is displayed:

Naam	Minimum wedde
Albert Beirnaert	2200,00
Alfons Deloor	2200,00
Edward Peeters	2200,00
Frans Bonduel	2200,00
Frans Mintjens	2200,00
Georges Ronsse	2200,00
Johan Verstrepen	2200,00
Jules Van Hevel	2200,00
Norbert Callens	2200,00
Paul Haghedooren	2200,00
Roger Baens	2200,00
Roger Swerts	2200,00
Roland Liboton	2200,00
Stijn Devolder	2200,00

At the bottom of the window, there is a message 'Druk een toets'.

- (1) Je gebruikt de method **Where** om records te filteren.
Je geeft een lambda-expressie mee, waarin je de filter definieert.
De lambda-expressie krijgt één entity als parameter binnen. Je noemt deze parameter **docent**.
Je geeft **true** terug als de entity in het resultaat mag voorkomen.
Je geeft **false** terug als de entity niet in het resultaat mag voorkomen.
- (2) Je gebruikt de method **OrderBy** om records te sorteren.
Je geeft een lambda-expressie mee, waarin je sortering definieert.
De lambda-expressie krijgt één entity als parameter binnen. Je noemt deze parameter **docent**.
Je geeft in deze lambda-expressie een property van deze entity terug.
EF sorteert records op de kolom die hoort bij die property.
- (3) Als je op meerdere properties wil sorteren (in dit voorbeeld op voornaam én familienaam), pas je op het resultaat van de **OrderBy**-method de method **ThenBy** toe. Je geeft terug een lambda-expressie. Deze lambda-expressie werkt op dezelfde manier als de lambda-expressie van de **OrderBy**-method.

Opmerking:

De methods **OrderBy** en **ThenBy** sorteren oplopend. De methods **OrderByDescending** en **ThenByDescending** sorteren aflopend.

5.1.4 LINQ-QUERIES EN QUERIES MET METHODS VERGELEKEN

- LINQ-queries zijn meestal leesbaarder dan queries gedefinieerd met query-methods.
- Sommige programmaonderdelen zijn meer onderhoudbaar met query-methods dan met LINQ-queries. In het volgende voorbeeld (in de method **Main**) ziet de gebruiker de **docenten**

met een **wedde** vanaf een in te tikken grens. De gebruiker kiest daarna hoe hij die **docenten** sorteert. De versie met een LINQ-query bevat drie sterk gelijkaardige queries:

```
Console.WriteLine("Minimum wedde:");
decimal minWedde;

if (decimal.TryParse(Console.ReadLine(), out minWedde))
{
    Console.WriteLine("Sorteren:1=op wedde, 2=op familienaam, 3=op voornaam:");
    var sorterendOp = Console.ReadLine();

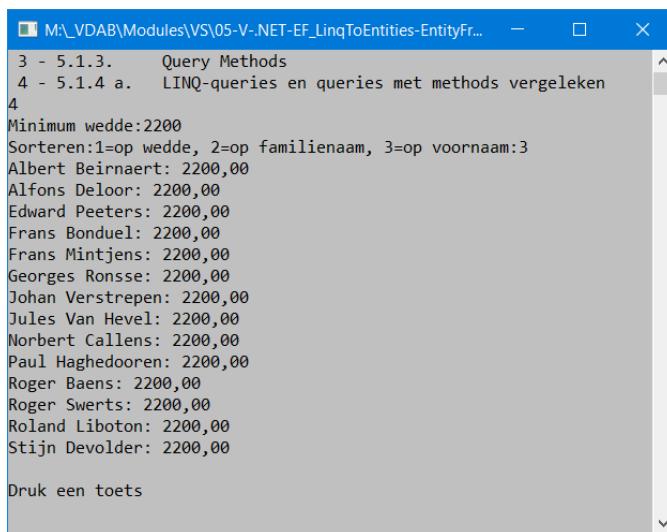
    using (var entities = new EFopleidingenEntities())
    {
        IQueryable<Docent> query;           // Het type van de variabele query is een
                                            // LINQ-query die Docent-entities teruggeeft
        switch (sorterendOp)
        {
            case "1":
                query = from docent in entities.Docenten
                         where docent.Wedde >= minWedde
                         orderby docent.Wedde
                         select docent;
                break;

            case "2":
                query = from docent in entities.Docenten
                         where docent.Wedde >= minWedde
                         orderby docent.Familienaam
                         select docent; // deze query lijkt sterk op de vorige
                break;

            case "3":
                query = from docent in entities.Docenten
                         where docent.Wedde >= minWedde
                         orderby docent.Voornaam
                         select docent; // deze query lijkt sterk op de vorige
                break;

            default:
                Console.WriteLine("Verkeerde keuze");
                query = null;
                break;
        }

        if (query != null)
        {
            foreach (var docent in query)
            {
                Console.WriteLine("{0}: {1}", docent.Naam, docent.Wedde);
            }
        }
        else
        {
            Console.WriteLine("U tikte geen getal");
        }
    }
}
```



The screenshot shows a Windows application window titled 'Query Methods'. It displays a list of names and amounts, ordered by amount. At the bottom of the window, there is a message 'Druk een toets' (Press a key).

Naam	Wedde
Albert Beirnaert	2200,00
Alfons Deloor	2200,00
Edward Peeters	2200,00
Frans Bonduel	2200,00
Frans Mintjens	2200,00
Georges Ronsse	2200,00
Johan Verstrepen	2200,00
Jules Van Hevel	2200,00
Norbert Callens	2200,00
Paul Haghedooren	2200,00
Roger Baens	2200,00
Roger Swerts	2200,00
Roland Liboton	2200,00
Stijn Devolder	2200,00

- De versie met query-methods bevat maar één query-definitie:

```
Console.WriteLine("Minimum wedde:");

decimal minWedde;

if (decimal.TryParse(Console.ReadLine(), out minWedde))
{
    Console.WriteLine("Sorteren:1=op wedde, 2=op familienaam, 3=op voornaam:");
    var sortererenOp = Console.ReadLine();
    Func<Docent, Object> sorteerLambda;

    switch (sortererenOp)
    {
        case "1":
            sorteerLambda = (docent) => docent.Wedde;
            break;

        case "2":
            sorteerLambda = (docent) => docent.Familienaam;
            break;

        case "3":
            sorteerLambda = (docent) => docent.Voornaam;
            break;

        default:
            Console.WriteLine("Verkeerde keuze");
            sorteerLambda = null;
            break;
    }
    if (sorteerLambda != null)
    {
        using (var entities = new EFopleidingenEntities())
        {
            var query = entities.Docenten
                .Where(docent => docent.Wedde >= minWedde)
                .OrderBy(sorteerLambda);

            foreach (var docent in query)
            {
                Console.WriteLine("{0}: {1}", docent.Naam, docent.Wedde);
            }
        }
    }
    else
    {
        Console.WriteLine("U tikte geen getal");
    }
}
```

```
}
```

5.2 EEN ENTITY ZOEKEN OP ZIJN PRIMAIRE-KEY-WAARDE

Je hoeft geen LINQ-query te schrijven om een entity te zoeken op zijn **primary-key**-waarde. Voor zo'n zoekoperatie kan je de **Find**-method gebruiken van de property in de **DbContext**-class die de verzameling entities voorstelt.

Voorbeeld in de method **Main**: je gebruikt de **Find**-method op de property **Docenten** van **EFOpleidingenEntities**, om een docent te zoeken op zijn **docentnummer**.

- Je geeft als parameter aan de **Find**-method de primary-key-waarde mee van de te zoeken entity.
- Deze **Find**-method geeft de **entity** terug, als deze entity voorkomt in de database.
- Deze **Find**-method geeft **null** terug, als deze entity niet voorkomt in de database.

Je kan dit uitproberen in de method **Main** van **Program.cs**:

```
using (var entities = new EFOpleidingenEntities())
{
    Console.Write("DocentNr.:");
    if (int.TryParse(Console.ReadLine(), out int docentNr))
    {
        var docent = entities.Docenten.Find(docentNr);
        Console.WriteLine(docent == null ? "Niet gevonden" : docent.Naam);
    }
    else
    {
        Console.WriteLine("U tikte geen getal");
    }
}
```

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-Entit... - □ X
6 - 5.2.      Een entity zoeken op zijn primary-key - waarde
6
DocentNr.:123456789
Niet gevonden
Druk een toets
```

5.3 GEDEELTELIJKE OBJECTEN OPHALEN

De queries die je tot nu maakte, lezen uit de records alle kolommen en vullen hiermee per entity alle bijbehorende properties. Dit kan de performantie benadelen als je in een programma-onderdeel slechts enkele properties per entity nodig hebt.

Je gebruikt als oplossing een LINQ-query, waarin je slechts enkele properties opvraagt. EF vertaalt zo'n LINQ-query naar een SQL-select-statement dat enkel de kolommen leest die bij die properties horen.

Een voorbeeld in de method **Main**

```
using (var entities = new EFopleidingenEntities())
{
    var query = from campus in entities.Campussen
                orderby campus.Naam
                select new { campus.CampusNr, campus.Naam }; // (1)

    foreach (var campusDeel in query)
    {
        Console.WriteLine("{0}: {1}", campusDeel.CampusNr, campusDeel.Naam);
    }
}
```

- (1) Je vraagt enkel de properties **CampusNr** en **Naam**, niet de overige properties. EF vertaalt dit naar een SQL-statement dat ook enkel leest uit de kolommen **CampusNr** en **Naam**. Het resultaat van deze query is een verzameling objecten. Het type van deze objecten is een anonieme tijdelijke class, aangemaakt door EF. Deze class heeft twee properties: **CampusNr** en **Naam**.

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToE... - □ X
7 - 5.3.a.      Gedeeltelijke objecten ophalen - LINQ
7
1: Andros
2: Delos
3: Gavdos
4: Hydra
5: Ikaria
6: Oinouses
Druk een toets
```

Je kan ook deze query schrijven met query-methods in plaats van een LINQ-query.

Je probeert dit uit door de opdracht

```
var query
```

(over de volledige drie regels) te vervangen door:

```
var query = entities.Campussen .OrderBy(campus => campus.Naam)
            .Select(campus => new { campus.CampusNr, campus.Naam });
```

```
using (var entities = new EFopleidingenEntities())
{
    //var query = from campus in entities.Campussen
    //            orderby campus.Naam
    //            select new { campus.CampusNr, campus.Naam }; // (1)

    var query = entities.Campussen .OrderBy(campus => campus.Naam)
        .Select(campus => new { campus.CampusNr, campus.Naam });

    foreach (var campusDeel in query)
    {
        Console.WriteLine("{0}: {1}", campusDeel.CampusNr, campusDeel.Naam);
    }
}
```

5.4 GROEPEREN IN QUERIES

Je kan in een query objecten groeperen met de combinatie van de sleutelwoorden **group by** en **into**. Je vermeldt na **by** de entity-property waarop je wil groeperen.

Een voorbeeld in de method **Main**. Je groepeert de **docenten** op **voornaam**:

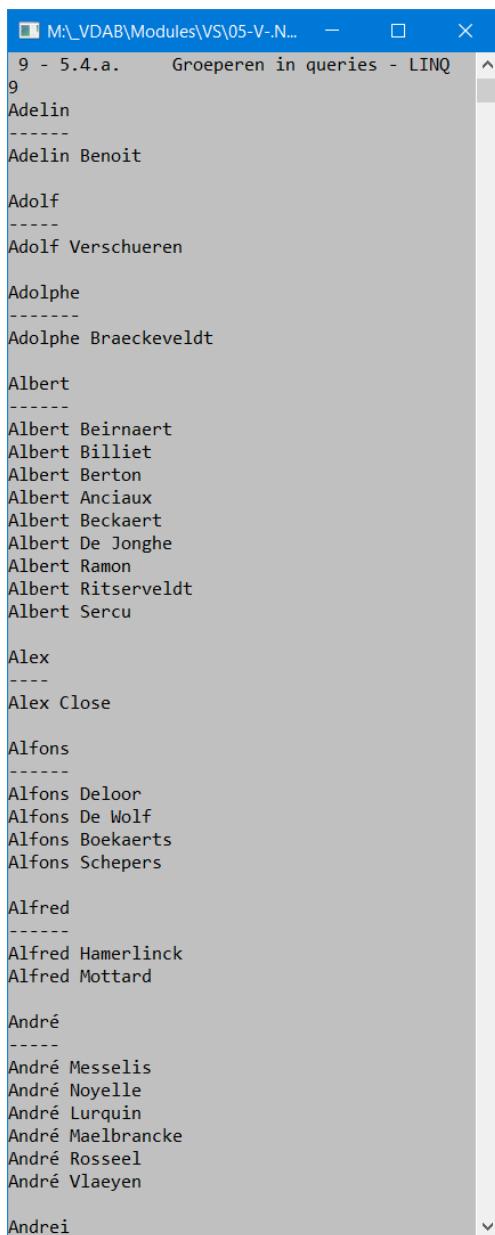
```
using (var entities = new EFopleidingenEntities())
{
    var query = from docent in entities.Docenten
                group docent by docent.Voornaam into VoornaamGroep
                select new { VoornaamGroep, Voornaam = VoornaamGroep.Key }; // (1)

    foreach (var voornaamStatistiek in query)
    {
        Console.WriteLine(voornaamStatistiek.Voornaam);
        Console.WriteLine(new string('-', voornaamStatistiek.Voornaam.Length));

        foreach (var docent in voornaamStatistiek.VoornaamGroep)
        {
            Console.WriteLine(docent.Naam);
        }

        Console.WriteLine();
    }
}
```

- (1) Het resultaat van deze query is een verzameling objecten. Deze hebben als type een anonieme tijdelijke class. Deze class heeft twee properties: **Voornaam** en **VoornaamGroep**. **Voornaam** is een voornaam die bij één of meerdere docenten voorkomt. **VoornaamGroep** is de verzameling **Docent** entities met deze voornaam.



The screenshot shows a command-line interface with the title 'M:_VDAB\Modules\VS\05-V.N... - Groeperen in queries - LINQ'. The output lists names grouped by their first name, separated by dashed lines:

- 9
- Adelin
-
- Adelin Benoit
- Adolf
-
- Adolf Verschueren
- Adolphe
-
- Adolphe Braeckeveldt
- Albert
-
- Albert Beirnaert
- Albert Billiet
- Albert Berton
- Albert Anciaux
- Albert Beckaert
- Albert De Jonghe
- Albert Ramon
- Albert Ritserveldt
- Albert Sercu
- Alex
-
- Alex Close
- Alfons
-
- Alfons Deloor
- Alfons De Wolf
- Alfons Boekaerts
- Alfons Schepers
- Alfred
-
- Alfred Hamerlinck
- Alfred Mottard
- André
-
- André Messelis
- André Noyelle
- André Lurquin
- André Maelbrancke
- André Rosseel
- André Vlaeyen
- Andrei

Een voorbeeld van twee objecten uit het resultaat van de query:

Voornaam (String)	VoornaamGroup (een verzameling Docent entities)		
Armand	260	Armand	Van Bruaene
	24	Armand	Bayens
Arsène	32	Arsène	Bauwens

Je kan ook deze query schrijven met query-methods in plaats van een LINQ-query.

Je kan dit uitproberen door de opdracht

```
var query
```

(over de volledige drie regels) te vervangen door:

```
var query = entities.Docenten.GroupBy((docent) => docent.Voornaam,
                                         (Voornaam, docenten) => new { Voornaam, VoornaamGroep = docenten });
```

```

using (var entities = new EF0pleidingenEntities())
{
    //var query = from docent in entities.Docenten
    //            group docent by docent.Voornaam into VoornaamGroep
    //            select new { VoornaamGroep, Voornaam = VoornaamGroep.Key }; // (1)

    var query = entities.Docenten.GroupBy((docent) => docent.Voornaam,
                                         (Voornaam, docenten) => new { Voornaam, VoornaamGroep = docenten });

    foreach (var voornaamStatistiek in query)
    {
        Console.WriteLine(voornaamStatistiek.Voornaam);
        Console.WriteLine(new string('-', voornaamStatistiek.Voornaam.Length));

        foreach (var docent in voornaamStatistiek.VoornaamGroep)
        {
            Console.WriteLine(docent.Naam);
        }

        Console.WriteLine();
    }
}

```

5.5 LAZY LOADING

Je kan ná het uitvoeren van een query op een entity, die je van de query krijgt, een geassocieerde entity lezen uit de database. Dit heet **lazy loading**. Daarbij stuurt EF een nieuw SQL-select-statement naar de database, om de geassocieerde entiteit(s) te lezen.

Een voorbeeld in de method `Main`. Je leest in de query zelf `Docent`-entities. Je leest pas ná de query de geassocieerde `Campus`-entities:

```

using (var entities = new EF0pleidingenEntities())
{
    Console.Write("Voornaam:");
    var voornaam = Console.ReadLine();

    var query = from docent in entities.Docenten
               where docent.Voornaam == voornaam
               select docent; // (1)

    foreach (var docent in query)
    {
        Console.WriteLine("{0} : {1}", docent.Naam, docent.Campus.Naam); // (2)
    }
}

```

- (1) Je leest in de query `Docent` entities.
- (2) Je spreekt het geassocieerde `Campus` object aan. EF stuurt op dat moment een SQL-select-statement naar de database om het juiste record uit de table `Campussen` te lezen. Je moet dit doen terwijl de `DbContext` nog niet gesloten is. Anders krijg je een exception.

Lazy loading kan een performantieprobleem veroorzaken.

Als je het programma uitvoert en `Roger` intikt, krijg je volgende output:

```
M:\_VDAB\Mod... 11 - 5.5. Lazy Loading
11
Voornaam:Roger
Roger Baens : Delos
Roger Baguet : Andros
Roger Blockx : Ikaria
Roger Decock : Andros
Roger De Vlaeminck : Delos
Roger Gyselinck : Delos
Roger Lambrecht : Gavdos
Roger Swerts : Delos

Druk een toets
```

EF heeft 9 SQL-select-statements naar de database gestuurd:

- Één SQL-statement dat de records met als voornaam Roger leest uit de table **Docenten**. Dit gebeurde bij het uitvoeren van de LINQ-query.
- Acht SQL-statements die elk één record lezen uit de table **Campus**. Dit gebeurde in de foreach loop bij het lezen van **docent.Campus.Naam**.

Je kan het aantal SQL-select-statements terugbrengen tot één met **eager loading** (zie hieronder)

5.6 EAGER LOADING LOST HET PERFORMANTIEPROBLEEM OP

Je kan het aantal SQL-select-statements terugbrengen tot één, door in de LINQ-query niet enkel de **Docent**-entities te lezen, maar ook al de geassocieerde **Campus**-entities. Dit heet **eager loading**.

Je gebruikt daarvoor in de query de **Include**-method:

```
using (var entities = new EFopleidingenEntities())
{
    Console.Write("Voornaam:");
    var voornaam = Console.ReadLine();

    var query = from docent in entities.Docenten.Include("Campus") // (1)
               where docent.Voornaam == voornaam
               select docent;

    foreach (var docent in query)
    {
        Console.WriteLine("{0}:{1}", docent.Naam, docent.Campus.Naam);
    }
}
```

- (1) De **DbContext** heeft properties waarop je LINQ-queries uitvoert.
In ons voorbeeld zijn dit de properties **Docenten** en **Campus**.
Deze properties bevatten een method **Include**. Je geeft aan deze method een **String** mee met de naam van een associatie die voorkomt in de bijbehorende entity class:

- Bij de **DbContext** property **Docenten** hoort de class **Docent**. Deze class bevat een associatie **Campus** (die verwijst naar de bijbehorende **Campus**-entity).
- Bij de **DbContext** property **Campus** hoort de class **Campus**. Deze class bevat een associatie **Docenten** (die verwijst naar de bijbehorende **Docent**-entities).

Bij het uitvoeren van de LINQ-query, zal EF ook de records lezen die bij deze associatie horen. In onze code leest EF niet enkel records uit de table **Docenten**, maar ook de gerelateerde records uit de table **Campus**. EF doet dit met één SQL-select-statement, met daarin het

sleutelwoord JOIN.

```
12 - 5.6.a.      Eager loading lost het performantieprobleem op (1)
12
Voornaam:Roger
Roger Baens:Delos
Roger Baguet:Andros
Roger Blockx:Ikaria
Roger Decock:Andros
Roger De Vlaeminck:Delos
Roger Gyselinck:Delos
Roger Lambrecht:Gavdos
Roger Swerts:Delos

Druk een toets
```

Je maakt een tweede voorbeeld in de method **Main**.

- Je leest in de query de **Campus**-objecten waarvan in de naam een zoekwoord voorkomt.
- Je leest in de query ook onmiddellijk de gerelateerde **Docenten**.
- EF zal dit vertalen naar één SQL-select-statement.

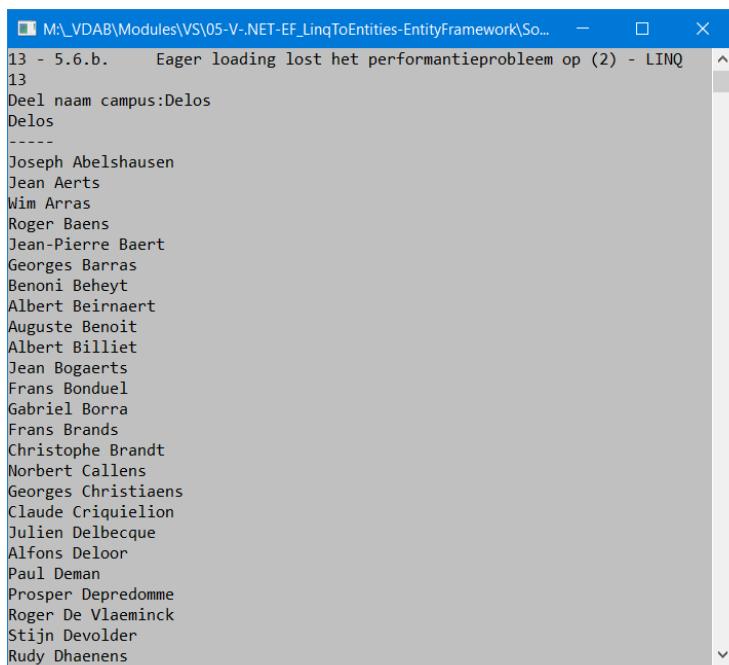
```
using (var entities = new EFopleidingenEntities())
{
    Console.Write("Deel naam campus:");
    var deelNaam = Console.ReadLine();

    var query = from campus in entities.Campussen.Include("Docenten")
                where campus.Naam.Contains(deelNaam)
                orderby campus.Naam
                select campus;

    foreach (var campus in query)
    {
        var campusNaam = campus.Naam;
        Console.WriteLine(campusNaam);
        Console.WriteLine(new string('-', campusNaam.Length));

        foreach (var docent in campus.Docenten)
        {
            Console.WriteLine(docent.Naam);
        }

        Console.WriteLine();
    }
}
```



The screenshot shows a Windows command prompt window with the title 'M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\So...'. The window contains a list of names, likely from a database query result. The names listed are: Deel naam campus:Delos, Delos, ----, Joseph Abelshausen, Jean Aerts, Wim Arras, Roger Baens, Jean-Pierre Baert, Georges Barras, Benoni Beheyt, Albert Beirnaert, Auguste Benoit, Albert Billiet, Jean Bogaerts, Frans Bonduel, Gabriel Borra, Frans Brands, Christophe Brandt, Norbert Callens, Georges Christiaens, Claude Criquielion, Julien Delbecque, Alfons Deloor, Paul Deman, Prosper Depredomme, Roger De Vlaeminck, Stijn Devolder, Rudy Dhaenens.

Je kan ook deze query schrijven met query methods in plaats van een LINQ-query.

Je probeert dit uit door de opdracht

```
var query
```

(over de volledige drie regels) te vervangen door:

```
var query = entities.Campussen.Include("Docenten")
    .Where(campus => campus.Naam.Contains(deelNaam))
    .OrderBy(campus => campus.Naam);
```

```
using (var entities = new EFopleidingenEntities())
{
    Console.Write("Deel naam campus:");
    var deelNaam = Console.ReadLine();

    //var query = from campus in entities.Campussen.Include("Docenten")
    //            where campus.Naam.Contains(deelNaam)
    //            orderby campus.Naam
    //            select campus;

    var query = entities.Campussen.Include("Docenten")
        .Where(campus => campus.Naam.Contains(deelNaam))
        .OrderBy(campus => campus.Naam);

    foreach (var campus in query)
    {
        var campusNaam = campus.Naam;
        Console.WriteLine(campusNaam);
        Console.WriteLine(new string('-', campusNaam.Length));

        foreach (var docent in campus.Docenten)
        {
            Console.WriteLine(docent.Naam);
        }

        Console.WriteLine();
    }
}
```

Per programma-onderdeel kan lazy loading of eager loading de beste oplossing zijn.

5.7 DE METHOD `ToListAsync` VAN EEN QUERY

Wanneer je met `foreach` itereert over een query, doet EF volgende stappen:

- De query omzetten naar een SQL-select-statement.
- Dit SQL-select-statement naar de database sturen.
- Bij iedere iteratie van jouw `foreach` een volgend record lezen uit het resultaat van dit SQL-select-statement.
- Dit record omzetten naar een entity.
- Jij kan deze entity binnen je `foreach`-iteratie aanspreken.
- Op het einde van iedere `foreach`-iteratie wordt deze entity vergeten.

Deze werkwijze heeft gevolgen:

- Als je twee keer itereert over een query, worden de voorgaande stappen twee keer allemaal uitgevoerd. Dit houdt in dat het SQL-select-statement ook twee keer uitgevoerd wordt. Je wil soms twee keer itereren over het resultaat van een query, zonder de query een tweede keer als SQL-select-statement uit te voeren (wegens bvb. performantieredenen).
- Je kan enkel itereren over een query binnen de `using` van de `DbContext` waarmee je query opbouwde, niet daarbuiten. Volgende code veroorzaakt een exception:

```
IQueryable<Campus> query;

using (var entities = new EFopleidingenEntities())
{
    query = from campus in entities.Campussen
            orderby campus.Naam
            select campus;
}

// Itereren na het sluiten van de DbContext (entities) kan niet
//foreach (var campus in query)
//{
//}
```

Dit houdt in dat het niet mogelijk is de opbouw van de query in één method te schrijven en het itereren over de query in een andere method te schrijven.

De oplossing voor deze problemen is de method `ToListAsync` van een query.

Als je deze method uitvoert, doet EF volgende stappen:

- De query omzetten naar een SQL-select-statement.
- Dit SQL-select-statement naar de database sturen.
- Itereren over de records uit het resultaat van het SQL-select-statement.
Van ieder record een entity maken. Deze entity toevoegen aan een List.
- Op het einde van de method geeft de method `ToListAsync` deze List terug.

Als je itereert over de List die je van de method `ToListAsync` krijgt, itereer je over de entities in deze List (in het RAM-geheugen).

Als je twee keer over deze List itereert, itereer je twee keer over entities in het RAM-geheugen, en stuur je geen SQL-select-statement naar de database:

```
    . . . .
using System.Collections.Generic;
. . . .

List<Campus> campussen;

using (var entities = new EFopleidingenEntities())
{
    var query = from campus in entities.Campussen
                orderby campus.Naam
                select campus;

    campussen = query.ToList();
}

foreach (var campus in campussen)
{
    Console.WriteLine(campus.Naam);
}

Console.WriteLine();

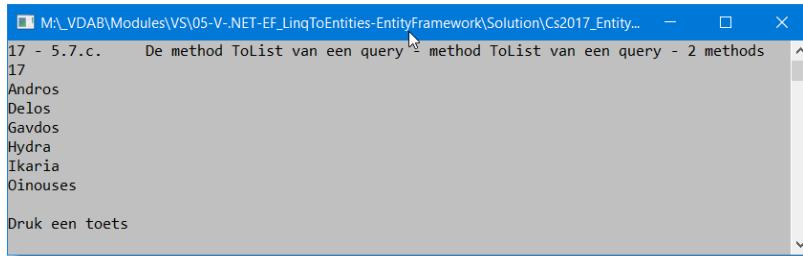
foreach (var campus in campussen)
{
    Console.WriteLine(campus.Naam);
}
```



Je kan de query én de List opbouwen in één method en de List doorgeven aan een andere method, die itereert over de List:

```
foreach (var campus in FindAllCampussen())
{
    Console.WriteLine(campus.Naam);
}
```

```
// FindAllCampussen
static List<Campus> FindAllCampussen()
{
    using (var entities = new EFopleidingenEntities())
    {
        return (from campus in entities.Campussen
                orderby campus.Naam
                select campus).ToList();
    }
}
```



M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs2017_Entity... - De method ToList van een query ↵ method ToList van een query - 2 methods
17
Andros
Delos
Gavdos
Hydra
Ikaria
Oinouses

Druk een toets

Opmerking:

Ook op queries die je definieert met query-methods kan je de method **ToList** uitvoeren.

5.8 TAAK 02 : KLANTEN EN HUN REKENINGEN

Je toont in de console een alfabetische lijst van de klanten.

Je toont per klant zijn naam, zijn rekeningen en het totale saldo van de rekeningen van de klant.

```
Bart
Totaal:0

Homer
345-6789012-12:500,00
Totaal:500,00

Lisa
Totaal:0

Maggie
Totaal:0

Marge
123-4567890-02:1000,00
234-5678901-69:2000,00
Totaal:3000,00
```

6 ENTITIES TOEVOEGEN

6.1 ÉÉN ENTITY TOEVOEGEN

Je doet volgende stappen om een entity toe te voegen aan de database:

- Je maakt de entity aan in het interne geheugen en je vult de properties van die entity in.
- Je voegt deze entity toe aan de verzameling gelijkaardige entities in de **DbContext**. Je doet dit met de method **Add** van deze verzameling. In ons voorbeeld bevatten de properties **Docenten** en **Campussen** van **EFopleidingenEntities** een method **Add**.
- Je roept op de **DbContext** de method **SaveChanges** op. EF stuurt op dat moment een **insert**-SQL-statement naar de database om de entity als een record toe te voegen.

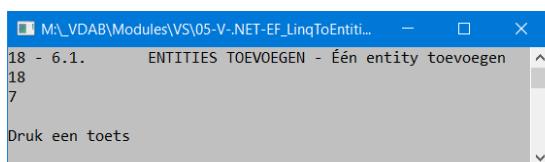
Voorbeeld in de method **Main**:

```
var campus = new Campus
{
    Naam = "Naam1",
    Straat = "Straat1",
    HuisNr = "1",
    PostCode = "1111",
    Gemeente = "Gemeente"
};

using (var entities = new EFopleidingenEntities())
{
    entities.Campussen.Add(campus);           // (1)
    entities.SaveChanges();                   // (2)
    Console.WriteLine(campus.CampusNr);       // (3)
}
```

- (1) Je hebt de entity toegevoegd aan de **DbContext**. De entity is dan nog niet opgeslagen in de database.
- (2) Je slaat de entity op in de database.
- (3) EF vult na het toevoegen van een record het autonummer van dit nieuwe record automatisch in bij de property die hoort bij de autonumber kolom: **CampusNr**.

Je voert het programma uit:



Je ziet daarna in de Server Explorer een nieuw record in de table **campussen**:

	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente
1	Andros	Somersstraat	22	2018	Antwerpen	
2	Delos	Oude Vest	17	9200	Dendermon...	
3	Gavdos	Europalaan	37	3600	Genk	
4	Hydra	Interleuvenl...	2	3001	Heverlee	
5	Ikaria	Vlamingstra...	10	8560	Wevelgem	
6	Oinouses	Archimedes...	4	8400	Oostende	
7	Naam1	Straat1	1	1111	Gemeente	
*	NULL	NULL	NULL	NULL	NULL	NULL

6.2 MEERDERE ENTITIES TOEVOEGEN

Als je meerdere entities met de `Add...` methods verbindt met de `DbContext`, moet je maar één keer de method `SaveChanges` uitvoeren om de bijbehorende SQL-insert-statements naar de database te sturen.

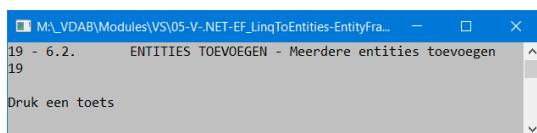
Voorbeeld in de method `Main`:

```
var campus2 = new Campus
{
    Naam = "Naam2",
    Straat = "Straat2",
    HuisNr = "2",
    PostCode = "2222",
    Gemeente = "Gemeente2"
};

var campus3 = new Campus
{
    Naam = "Naam3",
    Straat = "Straat3",
    HuisNr = "3",
    PostCode = "3333",
    Gemeente = "Gemeente3"
};

using (var entities = new EFopleidingenEntities())
{
    entities.Campussen.Add(campus2);
    entities.Campussen.Add(campus3);
    entities.SaveChanges();
}
```

Je voert het programma uit:



Je ziet daarna in de Server Explorer twee nieuwe records in de table `campussen`.

	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente
1	Andros	Somersstraat	22	2018	Antwerpen	
2	Delos	Oude Vest	17	9200	Dendermon...	
3	Gavdos	Europalaan	37	3600	Genk	
4	Hydra	Interleuvenl...	2	3001	Heverlee	
5	Ikaria	Vlamingstra...	10	8560	Wevelgem	
6	Oinouses	Archimedes...	4	8400	Oostende	
7	Naam1	Straat1	1	1111	Gemeente	
8	Naam2	Straat2	2	2222	Gemeente2	
9	Naam3	Straat3	3	3333	Gemeente3	
*	NULL	NULL	NULL	NULL	NULL	NULL

6.3 ENTITIES MET NIEUWE GEASSOCIEERDE ENTITIES TOEVOEGEN

Je kan in het interne geheugen een nieuwe entity én een nieuwe geassocieerde entity maken.

Het volstaat één van beide entities toe te voegen aan de `DbContext` met een `Add` method. Als je daarna op de `DbContext` de method `SaveChanges` uitvoert, voegt EF twee records toe aan de

database.

Voorbeeld 1 in de method Main:

je maakt een nieuwe **campus**. Je maakt een nieuwe **docent**.

Je associeert de **docent** met die **campus** vanuit het standpunt van de **campus**:

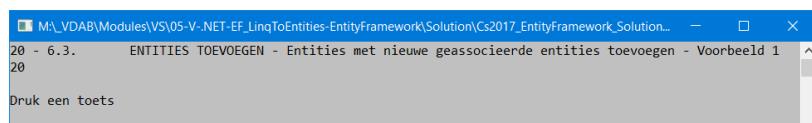
```
var campus4 = new Campus
{
    Naam = "Naam4",
    Straat = "Straat4",
    HuisNr = "4",
    PostCode = "4444",
    Gemeente = "Gemeente4"
};

var docent1 = new Docent
{
    Voornaam = "Voornaam1",
    Familienaam = "Familienaam1",
    Wedde = 1
};

// docent associëren met campus door hem toe te voegen aan de verzameling docenten van die campus
campus4.Docenten.Add(docent1);

using (var entities = new EFopleidingenEntities())
{
    entities.Campussen.Add(campus4);
    entities.SaveChanges();
}
```

Je ziet na het uitvoeren in de Server Explorer een nieuw record in de table **campussen** én een nieuw geassocieerd record in de table **docenten**.



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	308	Daniel	Willems	1600,00	6
	309	Jozef	Wouters	1100,00	1
▶	310	Voornaam1	Familienaam1	1,00	10
*	NULL	NULL	NULL	NULL	NULL

	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente
	1	Andros	Somersstraat	22	2018	Antwerpen
	2	Delos	Oude Vest	17	9200	Dendermon...
	3	Gavdos	Europalaan	37	3600	Genk
	4	Hydra	Interleuvenl...	2	3001	Heverlee
	5	Ikaria	Vlamingstra...	10	8560	Wevelgem
	6	Oinouses	Archimedes...	4	8400	Oostende
	7	Naam1	Straat1	1	1111	Gemeente
	8	Naam2	Straat2	2	2222	Gemeente2
	9	Naam3	Straat3	3	3333	Gemeente3
▶	10	Naam4	Straat4	4	4444	Gemeente4
*	NULL	NULL	NULL	NULL	NULL	NULL

Voorbeeld 2 in de method Main:

je maakt een nieuwe **campus**. Je maakt een nieuwe **docent**.

Je associeert de **docent** met die **campus** vanuit het standpunt van de **docent**:

```
var campus5 = new Campus
{
    Naam = "Naam5",
    Straat = "Straat5",
    HuisNr = "5",
    PostCode = "5555",
    Gemeente = "Gemeente5"
};

var docent2 = new Docent
{
    Voornaam = "Voornaam2",
    Familienaam = "Familienaam2",
    Wedde = 2
};

// docent associëren met campus door de property Campus van de docent in te vullen:
docent2.Campus = campus5;

using (var entities = new EFopleidingenEntities())
{
    entities.Docenten.Add(docent2);
    entities.SaveChanges();
}
```

Je ziet na het uitvoeren in de Server Explorer een nieuw record in de table **campussen** én een nieuw geassocieerd record in de table **docenten**.



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	308	Daniel	Willem	1600,00	6
	309	Jozef	Wouters	1100,00	1
	310	Voornaam1	Familienaam1	1,00	10
▶	311	Voornaam2	Familienaam2	2,00	11
*	NULL	NULL	NULL	NULL	NULL

	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente
	1	Andros	Somersstraat	22	2018	Antwerpen
	2	Delos	Oude Vest	17	9200	Dendermon...
	3	Gavdos	Europalaan	37	3600	Genk
	4	Hydra	Interleuven...	2	3001	Heverlee
	5	Ikaria	Vlamingstra...	10	8560	Wevelgem
	6	Oinouses	Archimedes...	4	8400	Oostende
	7	Naam1	Staat1	1	1111	Gemeente
	8	Naam2	Staat2	2	2222	Gemeente2
	9	Naam3	Staat3	3	3333	Gemeente3
	10	Naam4	Staat4	4	4444	Gemeente4
▶	11	Naam5	Straat5	5	5555	Gemeente5
*	NULL	NULL	NULL	NULL	NULL	NULL

6.4 EEN ENTITY TOEVOEGEN MET EEN ASSOCIATIE NAAR EEN BESTAANDE ENTITY

6.4.1 EEN ENTITY TOEVOEGEN EN DE ASSOCIATIE DEFINIËREN VANUIT DE VEEL-KANT

Je zal als voorbeeld een nieuwe **docent** toevoegen en vanuit die **docent** (de veel kant van de associatie) een associatie leggen naar een bestaande **campus**.

Er bestaan hier toe twee methodes:

- De geassocieerde entity lezen en associëren aan de nieuwe entity
- De geassocieerde entity associëren met de foreign-key-property

6.4.1.1 DE GEASSOCIEERDE ENTITY LEZEN EN ASSOCIËREN AAN DE NIEUWE ENTITY

Je doet bij deze methode stappen:

- Je maakt de nieuwe entity.
- Je leest de bestaande entity waarmee je de nieuwe entity wil associëren.
- Je associeert de bestaande entity met de nieuwe entity.
- Je voert op de object context de method **SaveChanges** uit.

Je moet deze stappen op dezelfde object context uitvoeren.

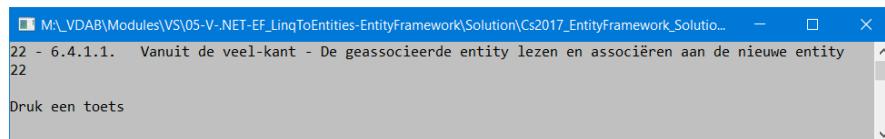
Voorbeeld in de method **Main**:

je maakt een **docent** en associeert deze **docent** met de **campus 1**.

```
var docent3 = new Docent
{
    Voornaam = "Voornaam3",
    Familienaam = "Familienaam3",
    Wedde = 3
};

using (var entities = new EFopleidingenEntities())
{
    var campus1 = entities.Campussen.Find(1);

    if (campus1 != null)
    {
        entities.Docenten.Add(docent3);
        docent3.Campus = campus1;
        entities.SaveChanges();
    }
    else
    {
        Console.WriteLine("Campus 1 niet gevonden");
    }
}
```



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	309	Jozef	Wouters	1100,00	1
	310	Voornaam1	Familienaam1	1,00	10
	311	Voornaam2	Familienaam2	2,00	11
▶	312	Voornaam3	Familienaam3	3,00	1
*	NULL	NULL	NULL	NULL	NULL

6.4.1.2 DE GEASSOCIEERDE ENTITY KOPPELEN MET DE FOREIGN-KEY-PROPERTY

Je leest bij deze methode de geassocieerde entity niet, wat performantiewinst oplevert.

Je vult de primary key van de geassocieerde entity in bij de property in de nieuwe entity die de foreign key naar de geassocieerde entity voorstelt:

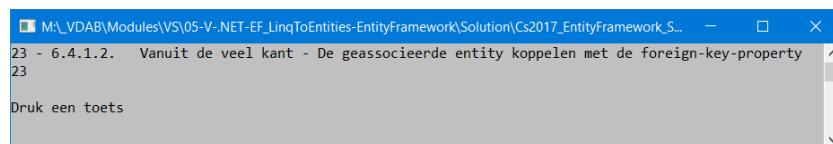
```
eenNieuweDocent.CampusNr = 1; // Campus 1 wordt de campus van de nieuwe docent
```

Voorbeeld in de method Main:

je maakt een **docent** en associeert die met de bestaande **campus 1**:

```
var docent4 = new Docent
{
    Voornaam = "Voornaam4",
    Familienaam = "Familienaam4",
    Wedde = 4,
    CampusNr = 1
};

using (var entities = new EFopleidingenEntities())
{
    entities.Docenten.Add(docent4);
    entities.SaveChanges();
}
```



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
309	Jozef	Wouters	1100,00	1	
310	Voornaam1	Familienaam1	1,00	10	
311	Voornaam2	Familienaam2	2,00	11	
312	Voornaam3	Familienaam3	3,00	1	
▶ 313	Voornaam4	Familienaam4	4,00	1	
*	NULL	NULL	NULL	NULL	NULL

6.4.2 EEN ENTITY TOEVOEGEN EN DE ASSOCIATIE DEFINIËREN VANUIT DE ÉÉN-KANT

Je doet bij deze methode volgende stappen:

- Je maakt de nieuwe entity.
- Je leest de bestaande entity waarmee je de nieuwe entity wil associëren.
- Je voegt de nieuwe entity toe aan de associatie in de bestaande entity met de **Add**-method.
- Je voert op de object context de method **SaveChanges** uit.

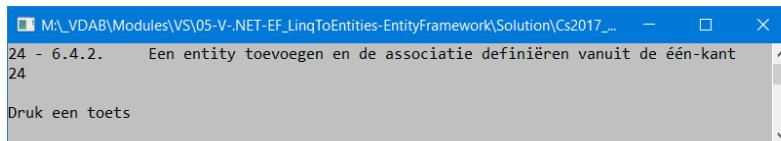
Voorbeeld in de method Main:

je maakt een **docent** en associeert die met de bestaande **campus 1**:

```
var docent5 = new Docent { Voornaam = "Voornaam5", Familienaam = "Familienaam5", Wedde = 5 };

using (var entities = new EFopleidingenEntities())
```

```
{  
    var campus1 = entities.Campussen.Find(1);  
  
    if (campus1 != null)  
    {  
        campus1.Docenten.Add(docent5);  
        entities.SaveChanges();  
    }  
    else  
    {  
        Console.WriteLine("Campus 1 niet gevonden");  
    }  
}
```



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	309	Jozef	Wouters	1100,00	1
	310	Voornaam1	Familienaam1	1,00	10
	311	Voornaam2	Familienaam2	2,00	11
	312	Voornaam3	Familienaam3	3,00	1
	313	Voornaam4	Familienaam4	4,00	1
▶	314	Voornaam5	Familienaam5	5,00	1
*	NULL	NULL	NULL	NULL	NULL

6.5 TAAK 03 : ZICHTREKENING TOEVOEGEN

De gebruiker kan een zichtrekening toevoegen. Je toont eerst een alfabetische lijst van de klanten. De gebruiker kiest daaruit één klant, door zijn klantnr. in te tikken:

```
5:Bart  
2:Homer  
3:Lisa  
4:Maggie  
1:Marge  
KlantNr:
```

Als de gebruiker geen getal tikt, toon je de foutmelding **Tik een getal.**

Als de gebruiker een onbestaand klantnr. intikt, toon je de foutmelding **Klant niet gevonden.**

Je vraagt daarna het rekeningnr. van de nieuwe rekening. Je moet op dit rekeningnr. geen invoercontrole doen.

Je maakt met deze gegevens een nieuwe zichtrekening die bij de gekozen klant hoort. Het saldo van de rekening staat op nul.

7 ENTITIES WIJZIGEN

7.1 ÉÉN ENTITY WIJZIGEN

Je doet volgende stappen om een entity te wijzigen in de database:

- Je leest de entity vanuit de database.
- Je wijzigt deze entity in het interne geheugen.
- Je roept op de object context de method `SaveChanges` op. EF stuurt op dat moment een `update`-SQL-statement naar de database om het record dat bij de entity hoort te wijzigen.

Voorbeeld in de method `Main`:

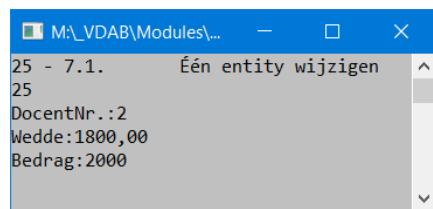
één docent opslag geven:

```
Console.WriteLine("DocentNr.:");

if (int.TryParse(Console.ReadLine(), out int docentNr))
{
    using (var entities = new EFopleidingenEntities())
    {
        var docent = entities.Docenten.Find(docentNr);

        if (docent != null)
        {
            Console.WriteLine("Wedde:{0}", docent.Wedde);
            Console.Write("Bedrag:");

            if (decimal.TryParse(Console.ReadLine(), out decimal bedrag))
            {
                docent.Opslag(bedrag);
                entities.SaveChanges();
            }
            else
            {
                Console.WriteLine("Tik een getal");
            }
        }
        else
        {
            Console.WriteLine("Docent niet gevonden");
        }
    }
}
else
{
    Console.WriteLine("Tik een getal");
}
```



Je kan in de Server Explorer nazien of het juiste record in de table docenten aangepast is.

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	1	Willy	Abbeloos	1400,00	4
▶	2	Joseph	Abelhausen	3800,00	2
	3	Joseph	Achten	1300,00	3
	4	François	Adam	1700,00	1

7.2 MEERDERE ENTITIES LEZEN EN SLECHTS ENKELE DAARVAN WIJZIGEN

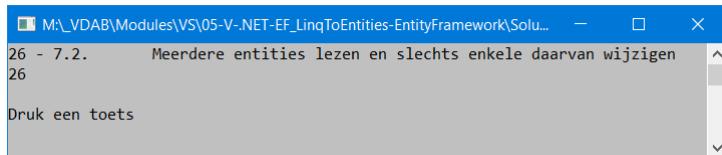
Het kan gebeuren dat je met de `DbContext` meerdere entites leest en slechts enkele wijzigt. Wanneer je op de `DbContext` de method `SaveChanges` uitvoert, stuurt EF enkel voor de aangepaste entites SQL-update-statements naar de database.

Voorbeeld in de method `Main`:

Je leest de `docenten` met de nummers **1** en **2** uit de database. Je wijzigt enkel de docent met het nummer **2**. De method `SaveChanges` stuurt één SQL-update-statement naar de database om het record met `DocentNr` **2** te wijzigen:

```
using (var entities = new EFopleidingenEntities())
{
    var docent1 = entities.Docenten.Find(1);
    var docent2 = entities.Docenten.Find(2);

    docent2.Opslag(10m);
    entities.SaveChanges();
}
```



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
	1	Willy	Abbeloos	1400,00	4
▶	2	Joseph	Abelhausen	3810,00	2
	3	Joseph	Achten	1300,00	3
	4	François	Adam	1700,00	1

7.3 ENTITIES WIJZIGEN DIE JE INDIRECT GELEZEN HEBT MET ASSOCIATIES

Soms lees je een entity en lees je met een `Navigation Property` van die entity een geassocieerde entity of een verzameling geassocieerde entites.

Je kan bijvoorbeeld een `Campus`-entity lezen met een query. Als je de property `Docenten` van deze `Campus`-entity aanspreekt, leest EF automatisch de `Docent`-entities die bij de `Campus`-entity horen.

Ook als je zo'n geassocieerde entites (in dit voorbeeld `Docent`-entities) wijzigt en op de `DbContext` de method `SaveChanges` uitvoert, wijzigt EF de records die horen bij deze geassocieerde entites:

Voorbeeld in de method `Main`:

Je geeft 10 € opslag aan de docenten uit `campus 1`

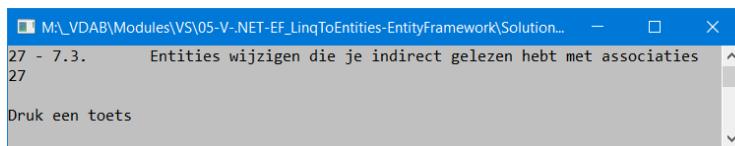
```
using (var entities = new EFopleidingenEntities())
{
    var campus1 = entities.Campussen.Find(1);
```

```
if (campus1 != null)
{
    foreach (var docent in campus1.Docenten)
    {
        docent.Opslag(10M);
    }

    entities.SaveChanges();
}
```

Je kan in de **Server Explorer** nazien of de juiste records in de table **docenten** aangepast zijn.

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
1	Willy	Abbeloos	1400,00	4	
2	Joseph	Abelshausen	3810,00	2	
3	Joseph	Achten	1300,00	3	
4	François	Adam	1700,00	1	
5	Jan	Adriaensens	2100,00	1	
6	René	Adriaensens	1600,00	6	
7	Frans	Aerenhouts	1300,00	3	
8	Emile	Aerts	1700,00	1	
9	Jean	Aerts	1200,00	2	
10	Mario	Aerts	1600,00	6	
11	Paul	Aerts	2000,00	5	
12	Stefan	Aerts	1500,00	5	
13	François	Alexander	1900,00	3	
14	Henri	Allard	1600,00	6	
15	Albert	Anciaux	1100,00	1	
16	Urbain	Anseeuw	1500,00	5	
17	Etienne	Antheunis	1900,00	3	
18	Jacques	Arlet	1400,00	4	
19	Wim	Arras	1800,00	2	
20	Roger	Baens	2200,00	2	
21	Dirk	Baert	1000,00	5	
22	Hubert	Baert	1400,00	4	
23	Jean-Pierre	Baert	1800,00	2	
24	Armand	Baeyens	1300,00	3	
25	Jan	Baevens	1700,00	1	
▶ 26	Roger	Baguet	2100,00	1	
27	Serge	Baguet	1600,00	6	
28	Gérard	Baldewyns	1200,00	2	



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
1	Willy	Abbeloos	1400,00	4	
2	Joseph	Abelshausen	3810,00	2	
3	Joseph	Achten	1300,00	3	
4	François	Adam	1710,00	1	
5	Jan	Adriaensens	2110,00	1	
6	René	Adriaensens	1600,00	6	
7	Frans	Aerenhouts	1300,00	3	
8	Emile	Aerts	1710,00	1	
9	Jean	Aerts	1200,00	2	
10	Mario	Aerts	1600,00	6	
11	Paul	Aerts	2000,00	5	
12	Stefan	Aerts	1500,00	5	
13	François	Alexander	1900,00	3	
14	Henri	Allard	1600,00	6	
15	Albert	Anciaux	1110,00	1	
16	Urbain	Anseeuw	1500,00	5	
17	Etienne	Antheunis	1900,00	3	
18	Jacques	Arlet	1400,00	4	
19	Wim	Arras	1800,00	2	
20	Roger	Baens	2200,00	2	
21	Dirk	Baert	1000,00	5	
22	Hubert	Baert	1400,00	4	
23	Jean-Pierre	Baert	1800,00	2	
24	Armand	Baeyens	1300,00	3	
25	Jan	Baeyens	1710,00	1	
▶ 26	Roger	Baguet	2110,00	1	
27	Serge	Baguet	1600,00	6	
28	Gérard	Baldus	1300,00	2	

Opmerking

EF stuurt in dit voorbeeld evenveel **update**-SQL-statements naar de database als er **docenten** behoren tot **campus 1**. Een snellere oplossing is vanuit EF een stored procedure op te roepen die alle **docenten** van **campus 1** opslag geeft met één **update**-SQL-statement. Je ziet verder in de cursus hoe je een stored procedure oproept.

7.4 EEN ASSOCIATIE VAN EEN ENTITY WIJZIGEN

7.4.1 DE ASSOCIATIE WIJZIGEN VANUIT DE VEEL-KANT

Je zal als voorbeeld een **docent** verhuizen naar een andere **campus**.

Er bestaan hiertoe twee methodes:

- De te associëren entity lezen en associëren aan de te wijzigen entity
- De te associëren entity associëren met de foreign-key-property

7.4.1.1 DE TE ASSOCIËREN ENTITY LEZEN EN ASSOCIËREN AAN DE TE WIJZIGEN ENTITY

Je doet bij deze methode volgende stappen:

- Je leest de te wijzigen entity.
- Je leest de entity waarmee je de te wijzigen entity wil associëren.
- Je legt de associatie.
- Je voert op de object context de method **SaveChanges** uit.

Voorbeeld in de method **Main**:

Je verhuist **docent 1** naar **campus 6**:

```

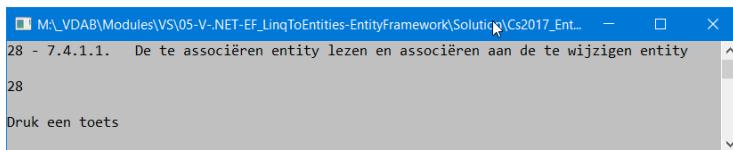
using (var entities = new EFopleidingenEntities())
{
    var docent1 = entities.Docenten.Find(1);

    if (docent1 != null)
    {
        var campus6 = entities.Campussen.Find(6);

        if (campus6 != null)
        {
            docent1.Campus = campus6;
            entities.SaveChanges();
        }
        else
        {
            Console.WriteLine("Campus 6 niet gevonden");
        }
    }
    else
    {
        Console.WriteLine("Docent 1 niet gevonden");
    }
}

```

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	4
	2	Ioseph	Ahelshausen	3810,00	2



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	6
	2	Ioseph	Ahelshausen	3810,00	2

7.4.1.2 DE TE ASSOCIËREN ENTITY ASSOCIËREN MET DE FOREIGN-KEY-PROPERTY

Je leest bij deze methode de te associëren entity niet, wat performantiewinst oplevert.

- Je vult de primary key van de te associeën entity in bij de property in de te wijzigen entity die de foreign key naar de te associeën entity voorstelt:

```
teWijzigenDocent.CampusNr = 1; // Campus 1 wordt de campus gewijzigde docent
```

Voorbeeld in de method Main:

Je verhuist docent 1 naar campus 2:

```

using (var entities = new EFopleidingenEntities())
{
    var docent1 = entities.Docenten.Find(1);

    if (docent1 != null)
    {
        docent1.CampusNr = 2;
        entities.SaveChanges();
    }
    else
    {

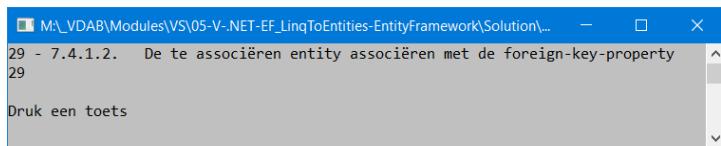
```

```

        Console.WriteLine("Docent 1 niet gevonden");
    }
}

```

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	6
	2	Ioseph	Abelhausen	3810,00	2



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	2
	2	Ioseph	Abelhausen	3810,00	2

7.4.2 DE ASSOCIATIE WIJZIGEN VANUIT DE ÉÉN-KANT

Je zal als voorbeeld een **docent** verhuizen naar een andere **campus**.

Je doet hierbij volgende stappen:

- Je leest de te wijzigen entity.
- Je leest de entity waarmee je de eerste entity wil associëren.
- Je voegt de eerste entity toe aan de associatie in de tweede entity met de **Add**-method.
- Je voert op de object context de method **SaveChanges** uit.

Voorbeeld in de method **Main**:

Je verhuist **docent 1** naar **campus 3**:

```

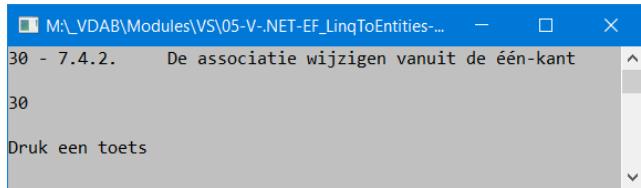
using (var entities = new EFopleidingenEntities())
{
    var docent1 = entities.Docenten.Find(1);

    if (docent1 != null)
    {
        var campus3 = entities.Campussen.Find(3);

        if (campus3 != null)
        {
            campus3.Docenten.Add(docent1);
            entities.SaveChanges();
        }
        else
        {
            Console.WriteLine("Campus 3 niet gevonden");
        }
    }
    else
    {
        Console.WriteLine("Docent 1 niet gevonden");
    }
}

```

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	2
	2	Ioseph	Abelhausen	3810,00	2



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
▶	1	Willy	Abbeloos	1400,00	3
	2	Joseph	Abelshausen	3810,00	2

7.5 TAAK 04 : STORTEN

De gebruiker kan geld storten op een rekening.

Je vraagt eerst het rekeningnr. van de rekening waarop de gebruiker het geld wil storten. Als de gebruiker een onbestaand rekeningnr. intikt, toon je de foutmelding **Rekening niet gevonden**.

Je vraagt daarna het te storten bedrag. Als de gebruiker geen getal intikt, toon je de foutmelding **Tik een getal**. Als de gebruiker een getal intikt kleiner of gelijk aan nul, toon je de foutmelding **Tik een positief getal**.

8 ENTITIES VERWIJDEREN

Je doet volgende stappen om een entity te verwijderen uit de database:

- Je leest de entity vanuit de database.
- Je voert de Remove-method uit van verzameling in de **DbContext** die soortgelijke entities bevat. Je geeft de gelezen entity mee als parameter.
- Je roept op de **DbContext** de method **SaveChanges** op. EF stuurt dan een **delete**-SQL-statement naar de database om het record dat bij de entity hoort te verwijderen.

Belangrijk:

Je moet deze stappen op dezelfde **DbContext** uitvoeren.

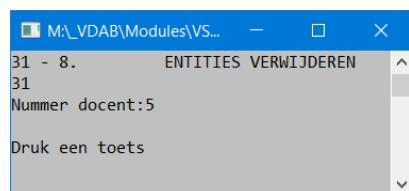
Voorbeeld inde method Main:

```
Console.WriteLine("Nummer docent:");

if (int.TryParse(Console.ReadLine(), out int docentNr))
{
    using (var entities = new EFopleidingenEntities())
    {
        var docent = entities.Docenten.Find(docentNr);

        if (docent != null)
        {
            entities.Docenten.Remove(docent);
            entities.SaveChanges();
        }
        else
        {
            Console.WriteLine("Docent niet gevonden");
        }
    }
}
else
{
    Console.WriteLine("Tik een getal");
}
```

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
1	Willy	Abbeloos	1400,00	3	
2	Joseph	Abelhausen	3810,00	2	
3	Joseph	Achten	1300,00	3	
4	François	Adam	1710,00	1	
5	Jan	Adriaensens	2110,00	1	
6	René	Adriaensens	1600,00	6	
7	Frans	Aerenhouts	1300,00	3	
8	Eduardo	Asante	1710,00	1	



	DocentNr	Voornaam	Familienaam	Wedde	CampusNr
1	Willy	Abbeloos	1400,00	3	
2	Joseph	Abelshausen	3810,00	2	
3	Joseph	Achten	1300,00	3	
4	François	Adam	1710,00	1	
6	René	Adriaensens	1600,00	6	
7	Frans	Aerenhouts	1300,00	3	
9	Emile	Aerts	1710,00	1	

8.1 TAAK 05 : KLANT VERWIJDEREN

De gebruiker kan een klant verwijderen, op voorwaarde dat hij geen rekeningen meer heeft.

Je vraagt het klantnr. van de te verwijderen klant. Als de gebruiker geen getal intikt, toon je de foutmelding **Tik een getal**. Als de gebruiker een onbestaand klantnr. intikt, toon je de foutmelding **Klant niet gevonden**. Als de gebruiker nog rekeningen heeft, toon je de foutmelding **Klant heeft nog rekeningen**.

9 TRANSACTIES

9.1 ALGEMEEN

Het doel van een transactie is meerdere SQL-statements als één geheel te aanziën. Het is de verantwoordelijkheid van de database ervoor te zorgen dat:

- Ofwel de volledige transactie lukt, wat wil zeggen dat alle SQL-statements binnen de transactie uitgevoerd zijn. Dit noemt men een **commit** van de transactie.
- Ofwel de volledige transactie mislukt (bvb. bij fout in de database, fout in jouw applicatie, stroomuitval), wat wil zeggen dat de aanpassingen van alle SQL-statements binnen de transactie ongedaan gemaakt worden. Dit noemt men een **rollback** van de transactie.

Een voorbeeld van een transactie is het overschrijven van geld van een spaarrekening naar een zichtrekening bij dezelfde bank. Hiervoor zijn twee **update**-statements nodig:

- Een statement dat het te transfereren geld aftrekt van het saldo van de spaarrekening.
- Een statement dat het te transfereren geld bijtelt bij het saldo van de zichtrekening.

Voorbeeld:

om 10 € over te schrijven van spaarrekening 111-1111111-70 naar zichtrekening 222-2222222-43 heb je twee update-statements nodig:

Eerste statement:

```
update Rekeningen  
set saldo=saldo - 10  
where RekeningNr = '111-1111111-70'
```

Tweede statement:

```
update Rekeningen  
set saldo=saldo + 10  
where RekeningNr = '222-2222222-43'
```

Fouten die kunnen optreden:

- Een fout in de databasesoftware
- Een fout in jouw applicatie
- Stroomuitval
- Het rekeningnummer van één van de rekeningen bestaat niet.
- Één van de records is te lang gelockt door een andere applicatie.

Indien één van deze twee statements mislukt, mag het andere statement ook niet uitgevoerd worden.

Door de SQL-statements te verzamelen in een transactie, zorgt de database er voor dat ofwel beide SQL-statements uitgevoerd worden, ofwel geen van beide.

Transacties hebben vier kenmerken (gekend als de **ACID** kenmerken):

- **Atomicity**

De SQL-statements die tot de transactie behoren vormen één geheel. Een database voert een transactie helemaal, of niet uit. Als halverwege de transactie een fout gebeurt, brengt de database de bijgewerkte records terug in hun toestand juist voor de transactie begon.

- **Consistency**

De transactie breekt geen databaseregels. Als een kolom bijvoorbeeld geen duplicaten kan bevatten, zal de database de transactie afbreken (rollback) op het moment dat je toch probeert duplication toe te voegen.

- **Isolation**

Gedurende een transactie zijn de bewerkingen van de transactie niet zichtbaar voor andere lopende transacties. Om dit te bereiken vergrendelt de database de bijgewerkte records tot het einde van de transactie (locking).

- **Durability**

Een voltooide transactie is definitief vastgelegd in de database, zelfs al valt de computer uit juist na het voltooien van de transactie.

9.2 ISOLATION LEVEL

Het isolation level van een transactie definieert hoe de transactie beïnvloed wordt door handelingen van andere gelijktijdige transacties.

Als meerdere transacties op eenzelfde moment in uitvoering zijn, kunnen volgende problemen optreden:

- **Dirty read**

Dit gebeurt als een transactie data leest die een andere transactie geschreven heeft, maar nog niet gecommit heeft. Als die andere transactie een rollback doet, is de data gelezen door de eerste transactie verkeerd.

- **Nonrepeatable read**

Dit gebeurt als een transactie meerdere keren dezelfde data leest en per leesopdracht deze data wijzigt. De oorzaak zijn andere transacties die tussen de leesoperaties van de eerste transactie dezelfde data wijzigen.

De eerste transactie krijgt geen stabiel beeld van de gelezen data.

- **Phantom read**

Dit gebeurt als een transactie meerdere keren dezelfde data leest en per leesoperatie meer records leest. De oorzaak zijn andere transacties die records toevoegen tussen de leesoperaties van de eerste transactie.

De eerste transactie krijgt geen stabiel beeld van de gelezen data.

Je verhindert één of meerdere van deze problemen door het isolation level van de transactie in te stellen:

↓ Isolation level ↓	Dirty read kan optreden	Nonrepeatable read kan optreden	Phantom read kan optreden
Read uncommitted	Ja	Ja	Ja
Read committed	Nee	Ja	Ja
Repeatable read	Nee	Nee	Ja

Serializable	Nee	Nee	Nee
--------------	-----	-----	-----

Het lijkt aanlokkelijk altijd het isolation level **Serializable** te gebruiken, want deze keuze lost alle problemen op. Het is echter zo dat **Serializable** het traagste isolation level is. De isolation levels van snel naar traag:



Read uncommitted → Read committed → Repeatable read → Serializable



Je moet dus per programma-onderdeel analyseren welke problemen (dirty read, ...) de goede werking van dat programma-onderdeel benadelen. Je kiest daarna een isolation level dat deze problemen oplost. Het isolation level read uncommitted wordt zelden gebruikt, omdat het geen enkel probleem oplost.

9.3 DE METHOD **SAVECHANGES**

Je leerde al de method **SaveChanges** van de **DbContext** kennen.

We herhalen nog eens wat deze method doet:

- Voor iedere entity die je aan de **DbContext** toegevoegd hebt (met de **Add...** methods) een **insert**-SQL-statement naar de database sturen.
- Voor iedere entity die je gelezen én **gewijzigd** hebt een **update**-SQL-statement naar de database sturen.
- Voor iedere entity die je **verwijderd** hebt ten opzichte van de object context een **delete**-SQL-statement naar de database sturen.

De method **SaveChanges** verzamelt al deze bewerkingen zelf in één transactie.

Jij hoeft dus in veel gevallen geen transactiebeheer te doen.

De method **SaveChanges** gebruikt **read committed** als transaction isolation level.

9.4 EIGEN TRANSACTIEBEHEER MET TRANSACTIONSCOPE

9.4.1 ALGEMEEN

Je kan soms de ingebouwde transacties van de method **SaveChanges** niet gebruiken.

Je moet dan zelf transactiebeheer doen. Voorbeelden waarom je zelf de transactie beheert:

- Je wil een ander isolation level dan **read committed** gebruiken bij de databasebewerkingen.
- Je wil een **distributed transaction** doen. Bij een distributed transactie bevinden de records die tot de transactie behoren zich niet in één, maar in meerdere databases.

Je doet eigen transactiebeheer met de class **TransactionScope**.

De method **SaveChanges** detecteert automatisch een lopende transactie die je met **TransactionScope** gestart hebt. De method **SaveChanges** start dan geen eigen transactie, maar doet zijn bewerkingen binnen jouw transactie.

Je maakt een `TransactionScope`-object binnen een `using`-structuur.

```
using (var transactionScope = new TransactionScope())
{
    // ...
}
```

Alle databasebewerkingen die je binnen deze `using`-structuur uitvoert, behoren automatisch tot één en dezelfde transactie.

Nadat alle bewerkingen goed aflopen, voer je op je `TransactionScope`-object de method `Complete` uit:

```
transactionScope.Complete();
```

Bij het uitvoeren van deze method gebeurt een `commit` van alle bewerkingen van alle databaseconnecties die je uitgevoerd hebt binnen de `using`-structuur.

Als je de `using`-structuur verlaat zonder de method `Complete()` uit te voeren (vb. een exception treedt op), gebeurt automatisch een `rollback` van alle bewerkingen van alle databaseconnecties die je opende binnen de `using`-structuur.

`TransactionScope`-objecten kunnen in elkaar genest zijn:

```
using (var transactionScope = new TransactionScope())
{
    using (var transactionScope2 = new TransactionScope())
    {
        ...
    }
}
```

Er bestaan `TransactionScope`-constructors waarbij je een parameter meegeeft van het type `TransactionScopeOption`. Je beslist met deze parameter hoe het `TransactionScope`-object zich gedraagt ten opzichte van andere `TransactionScope`-objecten waarbinnen het genest is.

De parameter kan volgende waarden bevatten:

- Required

Het `TransactionScope` object start een transactie als het object niet in een ander `TransactionScope` object genest is. Als er wel genest is, gebruikt het object de transactie die al door het omringende `TransactionScope`-object gestart werd.

- RequiresNew

Het `TransactionScope` object start een nieuwe transactie, zelfs als het object genest is in een ander `TransactionScope`-object.

- Suppress

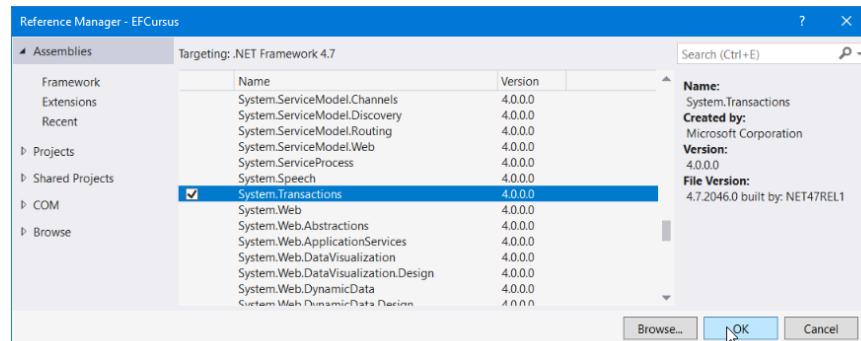
Alle databasebewerkingen binnen dit `TransactionScope`-object behoren niet tot een transactie, zelfs als het `TransactionScope`-object genest is in een ander `TransactionScope`-object.

Om het isolation level van de transactie in te stellen bestaan er `TransactionScope`-constructors die een parameter van het type `TransactionOption` aanvaarden. Een `TransactionOption` object heeft een `IsolationLevel` property om het isolation level in te stellen.

De class `TransactionScope` bevindt zich in de DLL `System.Transactions`.

Je legt in je project een referentie naar **System.Transactions.dll**:

- Je klikt in de **Solution Explorer** met de rechtermuisknop op je project.
- Je kiest **Add Reference**.
- Je plaatst in de lijst een vinkje bij **System.Transactions**
- Je klikt op **OK**.



9.4.2 VOORBEELD

Het script **VoorradenToevoegen.sql** voegt aan de database **EFOpleidingen** de table **Voorraden** toe. Je voert dit script uit in de **SQL Server Management Studio**.

De nieuwe table heeft volgende structuur:

Voorraden			
Column Name	Data Type	Allow Nulls	
MagazijnNr	int	<input type="checkbox"/>	
ArtikelNr	int	<input type="checkbox"/>	
AantalStuks	int	<input type="checkbox"/>	
RekNr	int	<input type="checkbox"/>	

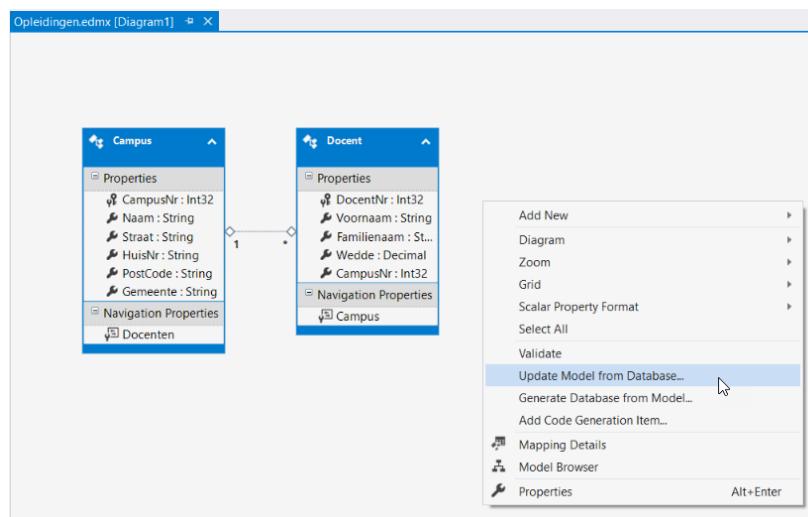
De table houdt bij hoeveel voorraad er per artikel per magazijn is.

De kolom **RekNr** geeft aan in welk rek van een magazijn een artikel te vinden is.

Je zal een hoeveelheid voorraad van één artikel overbrengen van één magazijn naar een ander magazijn. Je moet hierbij twee records aanpassen.

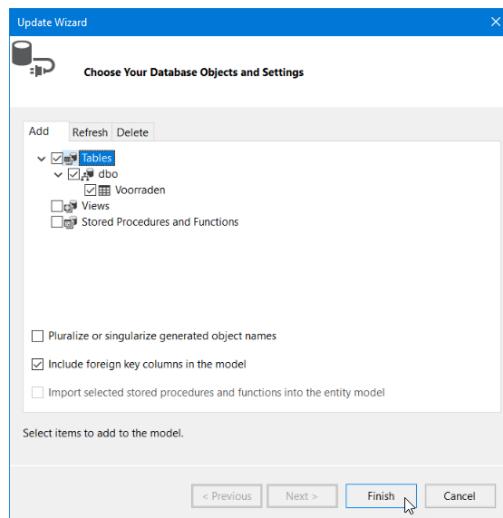
Je voegt aan **Opleidingen.edmx** de entity toe die hoort bij de table **Voorraden**:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Update Model from Database**.

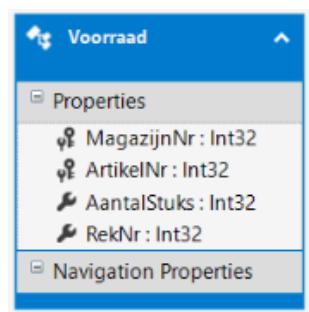


Je breidt met deze opdracht het [Entity Data Model](#) uit met informatie over nieuwe tables, relaties, kolommen, views en stored procedures uit de database.

- Je klapst in het tabblad **Add** het onderdeel **Tables** open.
- Je klapst daarbinnen **dbo** open.
- Je plaatst een vinkje bij **Voorraden**.
- Je kiest **Finish**.



Je corrigeert de naam van de entity **Voorraden** naar **Voorraad**



Je doet in de **eerste versie** van het programma geen eigen transactiebeheer, maar je gebruikt het ingebouwde transactiebeheer van de method **SaveChanges**:

```
try
{
    Console.Write("Artikel nr.:");
    var artikelNr = int.Parse(Console.ReadLine());

    Console.Write("Van magazijn nr.:");
    var vanMagazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Naar magazijn nr.:");
    var naarMagazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Aantal stuks:");
    var aantalStuks = int.Parse(Console.ReadLine());

    VoorraadTransfer1(artikelNr, vanMagazijnNr, naarMagazijnNr, aantalStuks);
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}
```

```
static void VoorraadTransfer1(int artikelNr, int vanMagazijnNr, int naarMagazijnNr, int aantalStuks)
{
    using (var entities = new EFopleidingenEntities())
    {
        var vanVoorraad = entities.Voorraden.Find(vanMagazijnNr, artikelNr);

        if (vanVoorraad != null)
        {
            if (vanVoorraad.AantalStuks >= aantalStuks) // (1)
            {
                vanVoorraad.AantalStuks -= aantalStuks; // (2)

                var naarVoorraad = entities.Voorraden.Find(naarMagazijnNr, artikelNr);

                if (naarVoorraad != null) // voorraad aanpassen
                {
                    naarVoorraad.AantalStuks += aantalStuks;
                }
                else // nieuwe voorraad aanmaken
                {
                    naarVoorraad = new Voorraad
                    {
                        ArtikelNr = artikelNr,
                        MagazijnNr = naarMagazijnNr,
                        AantalStuks = aantalStuks
                    };
                    entities.Voorraden.Add(naarVoorraad);
                }
            }

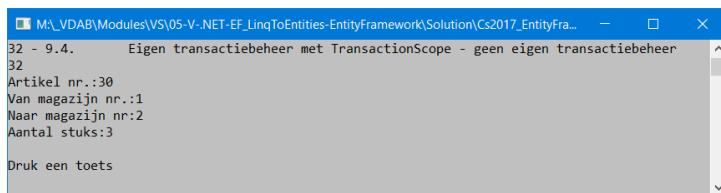
            entities.SaveChanges(); // (3)
        }
        else
        {
            Console.WriteLine("Te weinig voorraad voor transfer");
        }
    }
    else
    {
        Console.WriteLine("Artikel niet gevonden in magazijn {0}", vanMagazijnNr);
    }
}
```

Het programma doet de voorraadaanpassing correct, op voorwaarde dat geen andere gebruikers tegelijk aanpassingen doen. Als dit wel het geval is, kan volgende fout optreden:

- (1) Je controleert hier of er genoeg voorraad ligt in het van-magazijn.
- (2) Je past hier de bijbehorende entity aan in het interne geheugen.
- (3) EF past het bijbehorende record pas aan op het moment dat je **SaveChanges** uitvoert.

Tussendoor is het record niet gelockt. Een andere gebruiker kan dus tussendoor het aantal stuks van hetzelfde record verlagen. Zo kan uiteindelijk het aantal stuks negatief worden!

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	300	4
	2	10	1000	17
	2	20	2000	23
▶	2	30	3000	9
*	NULL	NULL	NULL	NULL



	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	297	4
	2	10	1000	17
	2	20	2000	23
▶	2	30	3003	9
*	NULL	NULL	NULL	NULL

Je beheert in de **tweede versie** van het programma de transactie zelf, om dit probleem op te lossen.

- Je plaatst het isolation level van de transactie op **Repeatable read**.
- Dan lockt de database de records die je leest tot het einde van de transactie.
- Zo kunnen andere gebruikers tussendoor de zelfde records niet wijzigen.

Je voegt boven in de source volgende opdracht toe:

```
using System.Transactions;
```

Enkel de method **VoorraadTransfer** is uitgebreid (aangeduide regels):

```
try
{
    Console.Write("Artikel nr.:");
    var artikelNr = int.Parse(Console.ReadLine());

    Console.Write("Van magazijn nr.:");
    var vanMagazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Naar magazijn nr.:");
    var naarMagazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Aantal stuks:");
    var aantalStuks = int.Parse(Console.ReadLine());

    VoorraadTransfer2(artikelNr, vanMagazijnNr, naarMagazijnNr, aantalStuks);
}
```

```
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een getal");
    }
}
```

```
.....
using System.Transactions;
.....

static void VoorraadTransfer2(int artikelNr, int vanMagazijnNr, int naarMagazijnNr, int aantalStuks)
{
    var transactionOptions = new TransactionOptions
    {
        IsolationLevel = System.Transactions.IsolationLevel.RepeatableRead
    };

    using (var transactionScope = new TransactionScope(TransactionScopeOption.Required,
                                                       transactionOptions))
    {
        using (var entities = new EFopleidingenEntities())
        {
            var vanVoorraad = entities.Voorraden.Find(vanMagazijnNr, artikelNr);

            if (vanVoorraad != null)
            {
                if (vanVoorraad.AantalStuks >= aantalStuks)
                {
                    vanVoorraad.AantalStuks -= aantalStuks;

                    var naarVoorraad =
                        entities.Voorraden.Find(naarMagazijnNr, artikelNr);

                    if (naarVoorraad != null)          // voorraad aanpassen
                    {
                        naarVoorraad.AantalStuks += aantalStuks;
                    }
                    else                            // nieuwe voorraad aanmaken
                    {
                        naarVoorraad = new Voorraad
                        {
                            ArtikelNr = artikelNr,
                            MagazijnNr = naarMagazijnNr,
                            AantalStuks = aantalStuks
                        };
                        entities.Voorraden.Add(naarVoorraad);
                    }

                    entities.SaveChanges();
                    transactionScope.Complete();
                }
            }
            else
            {
                Console.WriteLine("Te weinig voorraad voor transfer");
            }
        }
        else
        {
            Console.WriteLine("Artikel niet gevonden in magazijn {0}",
                             vanMagazijnNr);
        }
    }
}
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	297	4
	2	10	1000	17
	2	20	2000	23
▶	2	30	3003	9
*	NULL	NULL	NULL	NULL

```
M:\VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs2017_EntityFr...
33 - 9.4. Eigen transactiebeheer met TransactionScope - wel eigen transactiebeheer
33
Artikel nr.:30
Van magazijn nr.:1
Naar magazijn nr:2
Aantal stuks:4

Druk een toets
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	293	4
	2	10	1000	17
	2	20	2000	23
▶	2	30	3007	9
*	NULL	NULL	NULL	NULL

9.5 TAAK 06 : OVERSCHRIJVEN

De gebruiker kan geld overschrijven van een rekening naar een andere rekening.

Je vraagt het rekeningnr. van de rekening waarvan het geld wordt overgeschreven.

Je vraagt het rekeningnr. van de rekening naar waar het geld wordt overgeschreven.

Je vraagt het over te schrijven bedrag.

Als de gebruiker een onbestaand van-rekeningnr. intikt, toon je de foutmelding **Van-rekening niet gevonden**.

Als de gebruiker een onbestaand naar-rekeningnr. intikt, toon je de foutmelding **Naar-rekening niet gevonden**.

Als de gebruiker geen getal intikt bij bedrag, toon je de foutmelding **Tik een getal**.

Als de gebruiker een getal intikt bij bedrag dat kleiner is of gelijk aan nul, toon je de foutmelding **Tik een positief bedrag**.

Als het saldo van de van-rekening kleiner is dan het over te schrijven bedrag, toon je de foutmelding **Saldo ontoereikend**.

10 OPTIMISTIC RECORD LOCKING

10.1 ALGEMEEN

Je doet volgende stappen om een record te wijzigen met EF:

- Je leest het te wijzigen record als een entity in het interne geheugen.
- Je wijzigt de entity in het interne geheugen.
- Je voert op de object-context de method `SaveChanges` uit. Deze method stuurt een `update`-SQL-opdracht naar de database om het record te wijzigen dat bij de entity hoort.

Standaard doet EF geen controle of het record tussendoor niet door een andere gebruiker werd bijgewerkt. Dit houdt in dat je bij stap 3 de wijzigingen van die andere gebruiker overschrijft!

Je kan dit zien met volgend voorbeeld, waarin de voorraad bijgevuld wordt:

```
try
{
    Console.Write("Artikel nr.:");
    var artikelNr = int.Parse(Console.ReadLine());

    Console.Write("Magazijn nr.:");
    var magazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Aantal stuks toevoegen:");
    var aantalStuks = int.Parse(Console.ReadLine());

    VoorraadBijvulling1(artikelNr, magazijnNr, aantalStuks);
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}
```

```
static void VoorraadBijvulling1(int artikelNr, int magazijnNr, int aantalStuks)
{
    using (var entities = new EFopleidingenEntities())
    {
        var voorraad = entities.Voorraden.Find(magazijnNr, artikelNr);

        if (voorraad != null)
        {
            voorraad.AantalStuks += aantalStuks;
            Console.WriteLine("Pas nu de voorraad aan met de Server Explorer," +
                " druk daarna op Enter");
            Console.ReadLine();

            entities.SaveChanges();
        }
        else
        {
            Console.WriteLine("Voorraad niet gevonden");
        }
    }
}
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶	2	20	2000	23
2	30	3007	9	
*	NULL	NULL	NULL	NULL

Je voert het programma uit. Op het moment dat de boodschap **Pas nu de voorraad aan ...** verschijnt, schakel je over naar **Visual Studio**.

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFrame...
34 - 10.1.      OPTIMISTIC RECORD LOCKING
34
Artikel nr.:20
Magazijn nr.:2
Aantal stuks toevoegen:2020
Pas nu de voorraad aan met de Server Explorer, druk daarna op Enter
```

Je wijzigt in de **Server Explorer** de voorraad van hetzelfde record.

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶	2	20	2010	23
2	30	3007	9	
*	NULL	NULL	NULL	NULL

Je schakelt daarna terug naar de **console** applicatie en je drukt op **Enter**.

Na het uitvoeren van de console-applicatie keer je terug naar de Server Explorer. Als het venster met de table **Voorraden** nog openstaat geef je een rechtermuisklik in de table en je kiest **Refresh**. Je ziet dat de wijziging die je deed in de **Server Explorer** overschreven is door de **console**-applicatie.

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶	2	20	4020	23
2	30	3007	9	
*	NULL	NULL	NULL	NULL

EF stuurt hier volgend **update**-SQL-statement naar de database:

```
update Voorraden
set AantalStuks = @0
where MagazijnNr = @1 and ArtikelNr = @2
```

EF vult volgende waarden in bij de parameters:

- @0 Het aantal stuks zoals aangepast in de bijbehorende entity

- @1 De waarde die de kolom **MagazijnNr** had bij het lezen van het record
- @2 De waarde die de kolom **ArtikelNr** had bij het lezen van het record

Een eerste oplossing voor dit probleem is zelf transactiebeheer te doen, en het isolation level **Repeatable read** te gebruiken. De console-applicatie vergrendelt dan het record tot het einde van de transactie. Tussendoor kan niemand anders het record wijzigen.

Je ziet een tweede oplossing in dit hoofdstuk: **optimistic record locking**.

Bij optimistic record locking wordt het gelezen record niet echt gelockt. Bij het aanpassen van het record controleert EF of de waarden in het record nog dezelfde zijn als op het moment dat het record gelezen werd.

Als dit niet het geval is (omdat ondertussen een andere applicatie dit record wijzigde), werpt EF een exception.

Optimistic record locking is performant ten opzichte van een transactie met isolation level **Repeatable read** omdat er geen records gelockt worden.

De manier waarop je optimistic record locking activeert verschilt een beetje, naargelang de table al of niet een **timestamp**-kolom heeft.

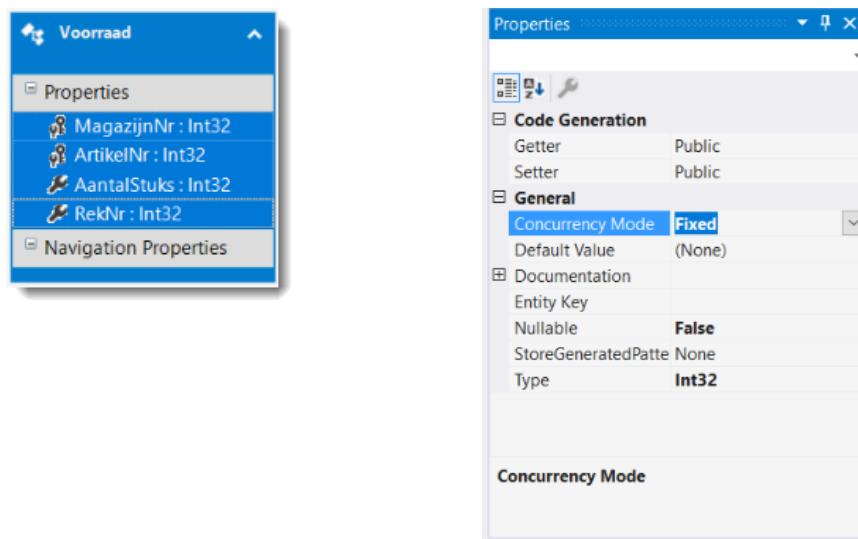
Een **timestamp**-kolom is een kolom in de database waarin de database zelf bij iedere recordwijziging een andere waarde invult.

10.2 TABLE ZONDER TIMESTAMP-KOLOM

Op dit moment heeft de table **Voorraden** geen **timestamp**-kolom.

Je activeert optimistic record locking dan op de volgende manier:

- Je opent het ontwerp van de entities (**Opleidingen.edmx**).
- Je klikt alle properties in de entity **Voorraad** aan en je wijzigt in het properties-venster de property **Concurrency Mode** naar **Fixed**.



Bij het lezen van het record, onthoudt EF de kolomwaarden van het gelezen record.

- Vb.: MagazijnNr: 1, ArtikelNr: 10, AantalStuks: 100, RekNr: 3

Bij het wijzigen van het record stuurt EF volgend **update**-SQL-statement naar de database:

```
update Voorraden
set AantalStuks = @0
where MagazijnNr=@1
and ArtikelNr=@2
and AantalStuks = @3
and RekNr = @4
```

EF vult volgende waarden in bij de parameters:

- @0 Het aantal stuks zoals aangepast in de bijbehorende entity
- @1 De waarde die de kolom **MagazijnNr** had bij het lezen van het record
- @2 De waarde die de kolom **ArtikelNr** had bij het lezen van het record
- @3 De waarde die de kolom **AantalStuks** had bij het lezen van het record
- @4 De waarde die de kolom **RekNr** had bij het lezen van het record

Als een andere applicatie ondertussen één van deze kolommen wijzigde bij hetzelfde record, vindt het **update**-SQL-statement geen record meer om aan te passen, omdat één of meerdere **where**-voorwaarden false teruggeven.

Zo detecteert EF dat een andere applicatie hetzelfde record wijzigde.

- Dan werpt EF een **DbUpdateConcurrencyException**.

Je voegt boven in de source volgende opdracht toe:

```
using System.Data.Entity.Infrastructure;
```

Enkel de method **VoorraadBijvulling** is gewijzigd:

```
try
{
    Console.Write("Artikel nr.:");
    var artikelNr = int.Parse(Console.ReadLine());

    Console.Write("Magazijn nr.:");
    var magazijnNr = int.Parse(Console.ReadLine());

    Console.Write("Aantal stuks toevoegen:");
    var aantalStuks = int.Parse(Console.ReadLine());

    VoorraadBijvulling2(artikelNr, magazijnNr, aantalStuks);
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}
```

```
.....
using System.Data.Entity.Infrastructure;
.....

static void VoorraadBijvulling2(int artikelNr, int magazijnNr, int aantalStuks)
{
    using (var entities = new EFopleidingenEntities())
    {
        var voorraad = entities.Voorraden.Find(magazijnNr, artikelNr);

        if (voorraad != null)
        {
            voorraad.AantalStuks += aantalStuks;
            Console.WriteLine("Pas nu de voorraad aan in de Server Explorer," +
                voorraad.AantalStuks);
        }
    }
}
```

```
        " druk daarna op Enter");
        Console.ReadLine();

        try
        {
            entities.SaveChanges();
        }
        catch (DbUpdateConcurrencyException)
        {
            Console.WriteLine("Voorraad werd door andere applicatie aangepast.");
        }
    }
    else
    {
        Console.WriteLine("Voorraad niet gevonden");
    }
}
```

Je voert het programma uit. Op het moment dat de boodschap **Pas nu de voorraad aan ...** verschijnt, schakel je over naar **Visual Studio**. Je wijzigt in de **Server Explorer** de voorraad van hetzelfde record. Daarna schakel je terug naar de console-applicatie en druk je op **Enter**.

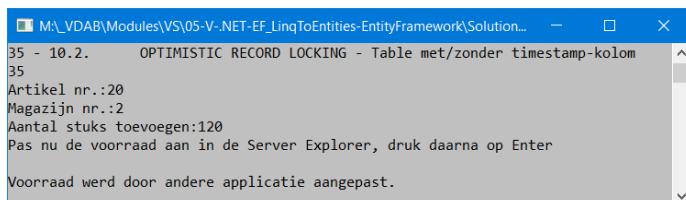
	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	293	4
	2	10	1000	17
▶	2	20	4020	23
	2	30	3007	9
*	NULL	NULL	NULL	NULL



```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution... - □ ×
35 - 10.2.      OPTIMISTIC RECORD LOCKING - Table met/zonder timestamp-kolom
35
Artikel nr.:20
Magazijn nr.:2
Aantal stuks toevoegen:120
Pas nu de voorraad aan in de Server Explorer, druk daarna op Enter
```

```
UPDATE dbo.Voorraden
SET AantalStuks=2010
WHERE MagazijnNr=2
AND ArtikelNr=20
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
	1	10	100	3
	1	20	200	12
	1	30	293	4
	2	10	1000	17
▶	2	20	2010	23
	2	30	3007	9
*	NULL	NULL	NULL	NULL



```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution... - □ ×
35 - 10.2.      OPTIMISTIC RECORD LOCKING - Table met/zonder timestamp-kolom
35
Artikel nr.:20
Magazijn nr.:2
Aantal stuks toevoegen:120
Pas nu de voorraad aan in de Server Explorer, druk daarna op Enter
Voorraad werd door andere applicatie aangepast.
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶	2	20	2010	23
2	30	3007	9	
*	NULL	NULL	NULL	NULL

10.3 TABLE MET TIMESTAMP-KOOLM

Als een table geen **timestamp**-kolom heeft wordt het **where**-deel van het **update**-SQL-statement bij **optimistic record locking** langer naargelang de table meer kolommen bevat. Het uitvoeren van een **update**-SQL-statement met een lang **where**-deel is nadelig voor de performantie.

Een oplossing is het toevoegen van een **timestamp**-kolom. Bij iedere wijziging van een record plaatst de database zelf een andere waarde in deze **timestamp**-kolom.

Je moet in het **where**-deel van het **update**-statement niet meer op iedere kolomwaarde controleren of het door een andere gebruiker werd bijgewerkt, maar enkel op deze **timestamp**-kolom.

Het script **TimestampInVoorradenToevoegen.sql** voegt aan de table **Voorraden** een **timestamp** kolom **Aangepast** toe.

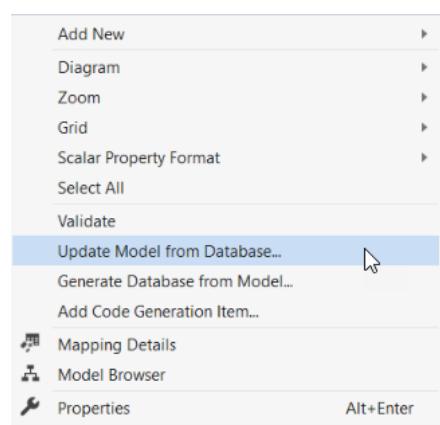
Je voert dit script uit in de **SQL Server Management Studio**.

De table heeft nu volgende structuur:

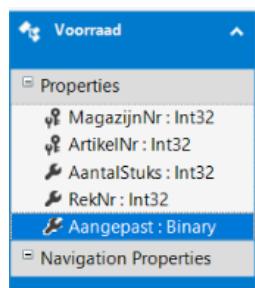
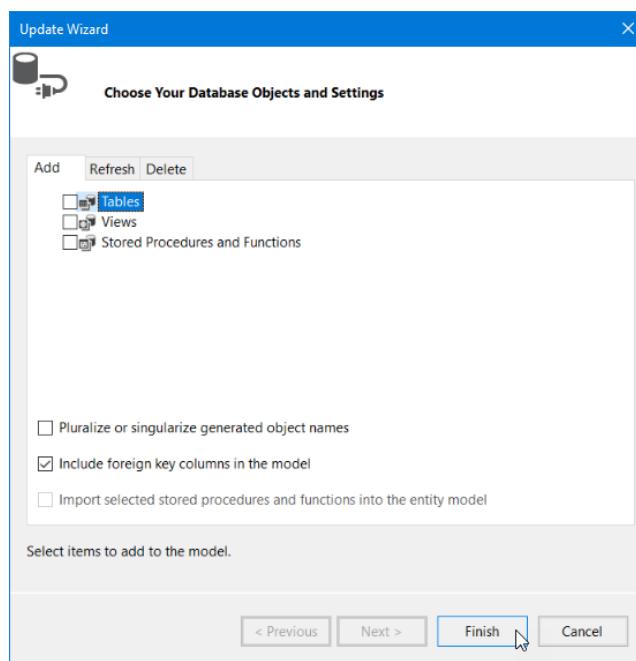
Voorraden		
Column Name	Data Type	Allow Nulls
MagazijnNr	int	<input type="checkbox"/>
ArtikelNr	int	<input type="checkbox"/>
AantalStuks	int	<input type="checkbox"/>
RekNr	int	<input type="checkbox"/>
Aangepast	timestamp	<input type="checkbox"/>

Je zorgt ervoor dat in **Opleidingen.edmx** de entity **Voorraad** ook een extra property **Aangepast** heeft:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest [Update Model from Database](#).



- Je kiest **Finish**.



De optimistic record locking moet nu enkel nog dit veld controleren:

- Je selecteert in de entity **Voorraad** de properties **MagazijnNr**, **ArtikelNr**, **AantalStuks** en **RekNr**, en je wijzigt in het properties venster de property **Concurrency Mode** terug naar **None**
- Je selecteert de property **Aangepast**, en je wijzigt in het properties venster de property **ConcurrencyMode** naar **Fixed**.

Als EF nu een record wijzigt, stuurt het volgend **update**-SQL-statement naar de database:

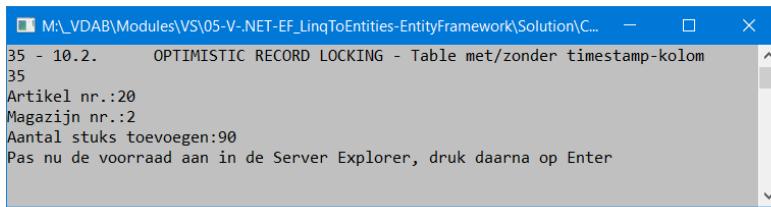
```
update Voorraden
set AantalStuks = @0
where MagazijnNr=@1
AND ArtikelNr=@2
AND Aangepast=@3
```

EF vult volgende waarden in bij de parameters:

- @0 Het aantal stuks zoals aangepast in de bijbehorende entity
- @1 De waarde die de kolom **MagazijnNr** had bij het lezen van het record
- @2 De waarde die de kolom **ArtikelNr** had bij het lezen van het record
- @3 De waarde die de kolom **Aangepast** had bij het lezen van het record

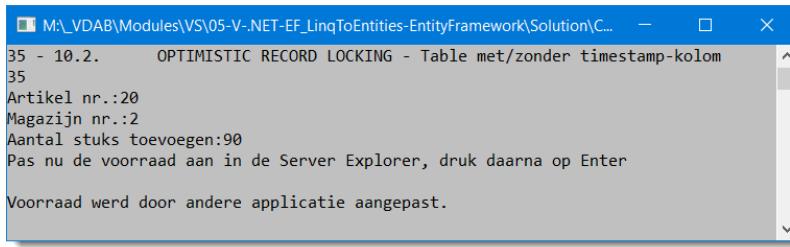
Als er nog kolommen bijkomen in de table **Voorraden**, blijft het **where**-deel van het **update-SQL**-statement zoals nu: kort en daarom ook performant.

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶ 2	20	2010	23	
2	30	3007	9	
*	NULL	NULL	NULL	NULL



```
SET AantalStuks=1900
WHERE MagazijnNr=2
AND ArtikelNr=20
```

	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶ 2	20	1900	23	
2	30	3007	9	
*	NULL	NULL	NULL	NULL



	MagazijnNr	ArtikelNr	AantalStuks	RekNr
1	10	100	3	
1	20	200	12	
1	30	293	4	
2	10	1000	17	
▶ 2	20	1900	23	
2	30	3007	9	
*	NULL	NULL	NULL	NULL

10.4 TAAK 07 : KLANT WIJZIGEN

De gebruiker kan de **voornaam** van een **klant** corrigeren.

Je vraagt het klantnr. van de te wijzigen klant.

Als de gebruiker geen getal intikt, toon je de foutmelding **Tik een getal**.

Als de gebruiker een onbestaand klantnr. intikt, toon je de foutmelding **Klant niet gevonden**.

Je vraagt de gecorrigeerde voornaam van de klant.

Tussen het lezen van de klant en het wijzigen van de klant, kan een andere gebruiker dezelfde klant wijzigen. Je vangt deze fout op met **optimistic record locking**. Je toont dan de foutmelding **Een andere gebruiker wijzigde deze klant**

11 ASSOCIATIES

11.1 ALGEMEEN

Je leerde reeds werken met een één-op-veel associatie (tussen campus en docenten).

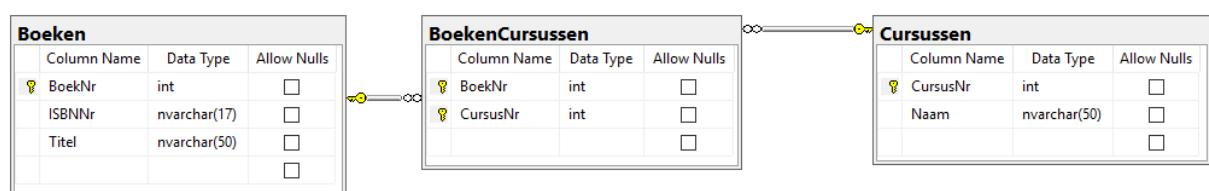
Je leert in dit hoofdstuk andere types associaties kennen.

11.2 VEEL-OP-VEEL-ASSOCIATIES ZONDER EXTRA ASSOCIATIEINFORMATIE

11.2.1 DE TABLES BOEKEN, CURSUSSEN EN BOEKENCURSUSSEN TOEVOEGEN AAN DE DATABASE

Het script `BoekenEnCursussenToevoegen.sql` voegt aan de database `EFOpleidingen` de tables `Boeken`, `Cursussen` en `BoekenCursussen` toe. Je voert dit script uit in de `SQL Server Management Studio`.

De nieuwe tables hebben volgende structuur:



Er is een veel-op-veel-relatie tussen `Boeken` en `Cursussen`:

- Één `boek` kan in meerdere cursussen gebruikt worden
- In één `cursus` kunnen meerdere `boeken` gebruikt worden.

Twee tables in een database kunnen geen directe veel-op-veel-relatie hebben.

De oplossing is een tussentable (`BoekenCursussen`) die een veel-op-één-relatie heeft naar `Boeken` én een veel-op-één-relatie naar `Cursussen`.

De tussentable bevat enkel de kolommen die de associatie definiëren:

- `BoekNr` en `CursusNr`. Er zijn hier geen kolommen met extra associatieinformatie.

11.2.2 DE ENTITIES DEFINIËREN DIE BIJ DE TABLES BOEKEN EN CURSUSSEN HOREN

Je voegt aan `Opleidingen.edmx` entities toe die horen bij de tables `Boeken` en `Cursussen`:

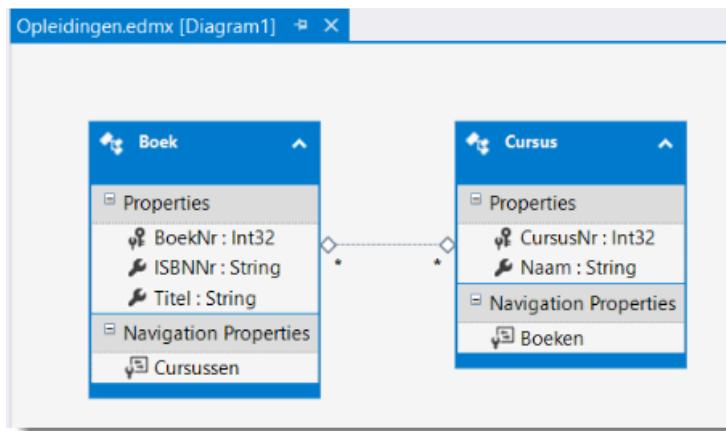
- Je klikt met de rechtermuisknop in de achtergrond van `Opleidingen.edmx`.
- Je kiest `Update Model from Database`.
- Je klapte in het tabblad `Add` het onderdeel `Tables` open.
- Je klapte daarbinnen het onderdeel `dbo` open.
- Je plaatst een vinkje bij `Boeken`, `BoekenCursussen` en `Cursussen` en je kiest `Finish`.

Je ziet de entities `Boeken` en `Cursussen` met een veel-op-veel associatie (* *). Er is geen tussentable nodig, omdat er geen associatieinformatie is

Je corrigeert enkele namen:

- Je wijzigt de naam van de entity `Boeken` naar `Boek`

- Je wijzigt de naam van de entity **Cursussen** naar **Cursus**



De navigation property **Cursussen** in de entity **Boek** verwijst vanuit een **boek** naar de **cursussen** waarin dit **boek** gebruikt wordt.

De navigation property **Boeken** in de entity **Cursus** verwijst vanuit een **cursus** naar de **boeken** die in die **cursus** gebruikt worden.

11.2.3 DE ENTITIES GEBRUIKEN VANUIT JE CODE

Je kan deze nieuwe entites nu gebruiken vanuit je code ([Program.cs](#))

Voorbeeld 1 in de method **Main**:

Je toont elk boek en de cursussen waarin dit boek gebruikt wordt:

```
using (var entities = new EFopleidingenEntities())
{
    var query = from boek in entities.Boeken.Include("Cursussen")
                orderby boek.Titel
                select boek;

    foreach (var boek in query)
    {
        Console.WriteLine(boek.Titel);

        foreach (var cursus in boek.Cursussen)
        {
            Console.WriteLine("\t{0}", cursus.Naam);
        }
    }
}
```

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\S... - X
11.      Associaties
11.2.     Veel-op-veel-associaties zonder extra associatieinformatie
36 - 11.2.3. De Entities gebruiken vanuit je code - Voorbeeld 1
36
Access from the Ground Up
Access
C++ : For Scientists and Engineers
C++
C++ : The Complete Reference
C++
C++ : The Core Language
C++
Oracle : A Beginner's Guide
Oracle
Oracle : The Complete Reference
Oracle
Oracle Backup & Recovery Handbook
Oracle
Relational Database Systems
Access
Oracle
Druk een toets
```

Voorbeeld 2 in de method Main:

Je toont elke cursus en de daarin gebruikte boeken:

```
using (var entities = new EFopleidingenEntities())
{
    var query = from cursus in entities.Cursussen.Include("Boeken")
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        Console.WriteLine(cursus.Naam);

        foreach (var boek in cursus.Boeken)
        {
            Console.WriteLine("\t{0}", boek.Titel);
        }
    }
}
```

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\S... - X
11.      Associaties
11.2.     Veel-op-veel-associaties zonder extra associatieinformatie
36 - 11.2.3. De Entities gebruiken vanuit je code - Voorbeeld 1
37 - 11.2.3. De Entities gebruiken vanuit je code - Voorbeeld 2
37
Access
    Relational Database Systems
    Access from the Ground Up
C++
    C++ : For Scientists and Engineers
    C++ : The Complete Reference
    C++ : The Core Language
Oracle
    Relational Database Systems
    Oracle : A Beginner's Guide
    Oracle : The Complete Reference
    Oracle Backup & Recovery Handbook
Druk een toets
```

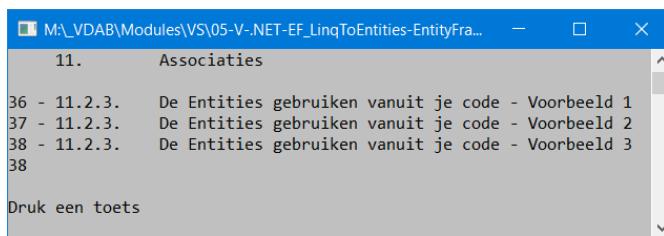
Voorbeeld 3 in de method Main:

Je maakt een nieuw boek en associeert dit met de cursus Oracle:

```
using (var entities = new EFopleidingenEntities())
{
    var nieuwBoek = new Boek
    {
        ISBNNr = "0-0788210-6-1",
        Titel = "Oracle Backup & Recovery Handbook"
    };

    var oracleCursus = (from cursus in entities.Cursussen
                        where cursus.Naam == "Oracle"
                        select cursus).FirstOrDefault();

    if (oracleCursus != null)
    {
        oracleCursus.Boeken.Add(nieuwBoek);
        entities.SaveChanges();
    }
    else
    {
        Console.WriteLine("cursus Oracle niet gevonden");
    }
}
```



	CursusNr	Naam
1	C++	
2	Access	
▶ 3	Oracle	
*	NULL	NULL

	BoekNr	ISBNNr	Titel
1	0-0705918-0-6	C++ : For Scientists and Engineers	
2	0-0788212-3-1	C++ : The Complete Reference	
3	1-5659211-6-X	C++ : The Core Language	
4	0-4448771-8-5	Relational Database Systems	
5	1-5595851-1-0	Access from the Ground Up	
6	0-0788212-2-3	Oracle : A Beginner's Guide	
7	0-0788209-7-9	Oracle : The Complete Reference	
▶ 8	0-0788210-6-1	Oracle Backup & Recovery Handbook	
*	NULL	NULL	NULL

11.3 VEEL-OP-VEEL-ASSOCIATIES MET EXTRA ASSOCIATIEINFORMATIE

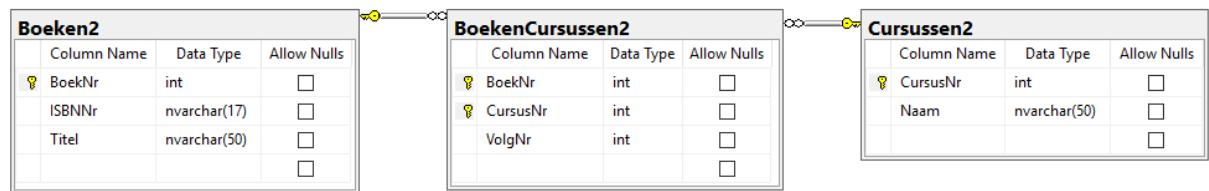
11.3.1 DE TABLES BOEKEN2, CURSUSSSEN2 EN BOEKENCURSUSSSEN2 TOEVOEGEN AAN DE DATABASE

Je leert hier hoe EF werkt als de tussentable in de veel-op-veel-relaties extra associatieinformatie bevat.

Het script `BoekenEnCursussenMetAssociatieInformatieToevoegen.sql` voegt aan de database `EFopleidingen` de tables `Boeken2`, `Cursussen2` en `BoekenCursussen2` toe.

Je voert dit script uit in de **SQL Server Management Studio**.

De nieuwe tables hebben volgende structuur:



De tussentable **BoekenCursussen2** bevat een extra kolom **VolgNr**.

Deze kolom geeft aan in welke volgorde boeken gebruikt worden in één **cursus**.

Je ziet deze informatie met volgend SQL-statement:

```

select Cursussen2.Naam, BoekenCursussen2.VolgNr, Boeken2.Titel
from Cursussen2
inner join BoekenCursussen2 on Cursussen2.CursusNr = BoekenCursussen2.CursusNr
inner join Boeken2 on Boeken2.BoekNr = BoekenCursussen2.BoekNr
order by Cursussen2.Naam, BoekenCursussen2.VolgNr
  
```

Het resultaat:

	Naam	VolgNr	Titel
1	Access	1	Relational Database Systems
2	Access	2	Access from the Ground Up
3	C++	1	C++ : The Core Language
4	C++	2	C++ : The Complete Reference
5	C++	3	C++ : For Scientists and Engineers
6	Oracle	1	Relational Database Systems
7	Oracle	2	Oracle : A Beginner's Guide
8	Oracle	3	Oracle : The Complete Reference

Opmerking:

BoekenCursussen2 bevat een samengestelde index op **CursusNr** en **VolgNr** die unieke waarden vereist. Zo kan een **cursus** geen twee keer hetzelfde **VolgNr** hebben.

11.3.2 DE ENTITIES DEFINIËREN DIE BIJ DE TABLES BOEKEN2, CURSUSSEN2 EN BOEKENCURSUSSEN2 HOREN

Je voegt aan **Opleidingen.edmx** entities toe die horen bij de tables **Boeken2**, **Cursussen2** en **BoekenCursussen2**:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Update Model from Database**.
- Je klapst in het tabblad **Add** het onderdeel **Tables** open.
- Je klapst daarbinnen het onderdeel **dbo** open.
- Je plaatst een vinkje bij **Boeken2**, **BoekenCursussen2** en **Cursussen2**
- Je kiest **Finish**.

Je ziet de entities **Boeken2**, **Cursussen2** én een associatieentity **BoekenCursussen2**. De associatieentity bevat de associatieinformatie **VolgNr**.

Er is een één-op-veel-associatie tussen **Boeken2** en **BoekenCursussen2** én een één-op-veel-

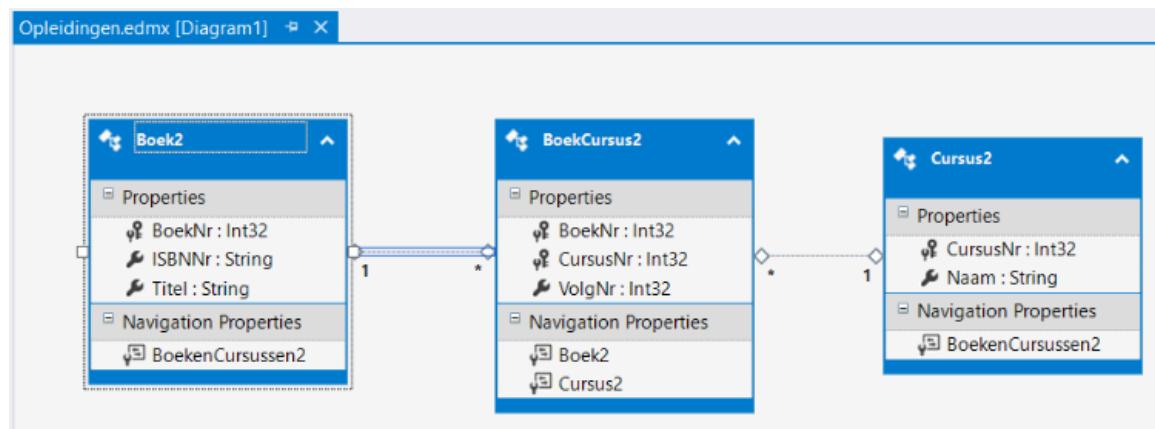
associatie tussen **Cursussen2** en **BoekenCursussen2**.

Deze twee associaties samen vormen een veel-op-veel-associatie tussen **Boeken2** en **Cursussen2**.

Je corrigeert enkele namen:

- Je wijzigt de naam van de entity **Boeken2** naar **Boek2**
- Je wijzigt in het properties venster de **Entity Set Name** naar **Boeken2**
- Je wijzigt de naam van de navigation property **BoekenCursussen2** naar **BoekenCursussen2**.
- Je wijzigt de naam van de entity **Cursussen2** naar **Cursus2**
- Je wijzigt in het properties venster de **Entity Set Name** naar **Cursussen2**
- Je wijzigt de naam van de navigation property **BoekenCursussen2** naar **BoekenCursussen2**.
- Je wijzigt de naam van de entity **BoekenCursussen2** naar **BoekCursus2**
- Je wijzigt in het properties venster de **Entity Set Name** naar **BoekenCursussen2**
- Je wijzigt de naam van de navigation property **Boeken2** naar **Boek2**.
- Je wijzigt de naam van de navigation property **Cursussen2** naar **Cursus2**.

Het eindresultaat:



De navigation property **BoekenCursussen** in de entity **Boek** laat toe vanuit een **boek** te verwijzen naar de associatieinformatie **BoekCursus**. Hier vind je het **volgnr.** van iedere **cursus** die dit **boek** gebruikt. Om andere informatie van deze **cursus** op te halen gebruik je de navigation property **Cursus** van **BoekCursus**.

De navigation property **BoekenCursussen** in de entity **Cursus** laat toe vanuit een **cursus** te verwijzen naar de associatieinformatie **BoekCursus**. Hier vind je het **volgnr.** van ieder **boek** die in de **cursus** gebruikt wordt. Om andere informatie van dit **boek** op te halen gebruik je de navigation property **Boek** van **BoekCursus**.

11.3.3 DE ENTITIES GEBRUIKEN VANUIT JE CODE

Je kan deze nieuwe entites nu gebruiken vanuit je code ([Program.cs](#))

Voorbeeld 1 in de method **Main**:

Je toont elke **cursus** en de daarin gebruikte **boeken**:

```

using (var entities = new EFopleidingenEntities())
{
    var query = from cursus2 in entities.Cursussen2.Include("BoekenCursussen2.Boek2")           // (1)
                orderby cursus2.Naam
                select cursus2;

    foreach (var cursus in query)
    {
        Console.WriteLine(cursus.Naam);

        foreach (var boekCursus in cursus.BoekenCursussen2)
        {
            Console.WriteLine("\t{0}:{1}", boekCursus.VolgNr, boekCursus.Boek2.Titel);
        }
    }
}

```

BoekenCursussen2.Boek2 betekent: Laadt van iedere **cursus** de gerelateerde informatie van de navigation property **BoekenCursussen2** én van de gerelateerde informatie nog eens de gerelateerde data van de navigation property **Boek2**.

Je ziet volgende informatie:

```

11.3.      Veel-op-veel-associaties met extra associatieinformatie
39 - 11.3.3. De Entities gebruiken vanuit je code - Voorbeeld 1
39
Access
  1:Relational Database Systems
  2:Access from the Ground Up
C++
  3:C++ : For Scientists and Engineers
  2:C++ : The Complete Reference
  1:C++ : The Core Language
Oracle
  1:Relational Database Systems
  2:Oracle : A Beginner's Guide
  3:Oracle : The Complete Reference
Druk een toets

```

Voorbeeld 2 in de method **Main**:

Je voegt een **boek** toe:

```

var nieuwBoek = new Boek2() { ISBNNr = "0-201-70431-5", Titel = "Modern C++ Design" };

var transactionOptions = new TransactionOptions
{
    IsolationLevel = IsolationLevel.Serializable
};

using (var transactionScope = new TransactionScope(TransactionScopeOption.Required,
                                                    transactionOptions))
{
    using (var entities = new EFopleidingenEntities())
    {
        // Cursus C++ ophalen
        // én het hoogste volgnr. van boek gebruikt in die cursus.
        // Met transactie met isolation level Serializable
        // kan daarna niemand anders een boek toevoegen aan C++ cursus
        // en is het nieuwe volgnr gelijk aan 1 + hoogst gelezen volgnr
        var query = from cursus2 in entities.Cursussen2.Include("BoekenCursussen2")
                    where cursus2.Naam == "C++"
                    select new
                    {
                        Cursus = cursus2,

```

```

        HoogsteVolgnr = cursus2.BoekenCursussen2.Max(boekCursus =>
        boekCursus.VolgNr)
    };

    var queryResult = query.FirstOrDefault();

    if (queryResult != null)
    {
        entities.BoekenCursussen2.Add(new BoekCursus2
        {
            Boek2 = nieuwBoek,
            Cursus2 = queryResult.Cursus,
            VolgNr = queryResult.HoogsteVolgnr + 1
        });

        entities.SaveChanges();
    }

    transactionScope.Complete();
}
}

```

Cursussen2

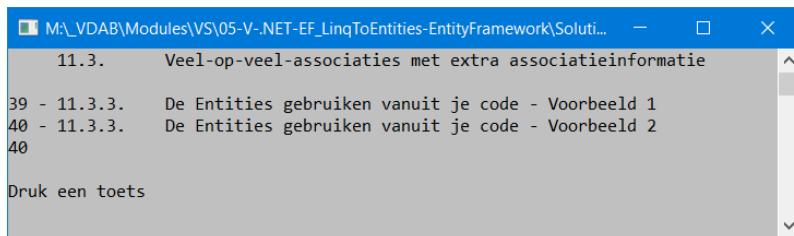
	CursusNr	Naam
▶	1	C++
	2	Access
	3	Oracle
*	NULL	NULL

BoekenCursussen2

	BoekNr	CursusNr	VolgNr
	3	1	1
	2	1	2
▶	1	1	3
	4	2	1
	5	2	2
	4	3	1
	6	3	2
	7	3	3
*	NULL	NULL	NULL

Boeken2

	BoekNr	ISBNNr	Titel
	1	0-0705918-0-6	C++ : For Scientists and Engineers
	2	0-0788212-3-1	C++ : The Complete Reference
▶	3	1-5659211-6-X	C++ : The Core Language
	4	0-4448771-8-5	Relational Database Systems
	5	1-5595851-1-0	Access from the Ground Up
	6	0-0788212-2-3	Oracle : A Beginner's Guide
	7	0-0788209-7-9	Oracle : The Complete Reference
*	NULL	NULL	NULL

BoekenCursussen2Boeken2

	BoekNr	CursusNr	VolgNr
3	1	1	1
2	1		2
1	1		3
► 8	1		4
4	2		1
5	2		2
4	3		1
6	3		2
7	3		3
* NULL	NULL	NULL	NULL

	BoekNr	ISBNNr	Titel
1	0-0705918-0-6	C++ : For Scientists and Engineers	
2	0-0788212-3-1	C++ : The Complete Reference	
3	1-5659211-6-X	C++ : The Core Language	
4	0-4448771-8-5	Relational Database Systems	
5	1-5595851-1-0	Access from the Ground Up	
6	0-0788212-2-3	Oracle : A Beginner's Guide	
7	0-0788209-7-9	Oracle : The Complete Reference	
► 8	0-201-70431-5	Modern C++ Design	
* NULL	NULL	NULL	NULL

11.4 EEN ASSOCIATIE VAN EEN ENTITY-CLASS NAAR ZICHZELF

11.4.1 ALGEMEEN

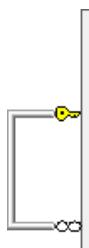
Een table in een relationele database kan een foreign-key-kolom hebben die verwijst naar de primary key van dezelfde table. De bijbehorende entity class heeft dan een navigation property die verwijst naar dezelfde entity class.

11.4.2 DE TABLE CURSISTEN TOEVOEGEN AAN DE DATABASE

Het script `CursistenToevoegen.sql` voegt aan de database `EFopleidingen` de table `Cursisten` toe.

Je voert dit script uit in de `SQL Server Management Studio`.

De nieuwe table heeft volgende structuur:



Cursisten			
Column Name	Data Type	Allow Nulls	
CursistNr	int	<input type="checkbox"/>	
Voornaam	nvarchar(50)	<input type="checkbox"/>	
Familienaam	nvarchar(50)	<input type="checkbox"/>	
MentorNr	int	<input checked="" type="checkbox"/>	

Een `cursist` kan een andere `cursist` als `mentor` hebben. De kolom `MentorNr` bevat dan het `CursistNr` van de `mentor`. Een `mentor` kan meerdere `cursisten` hebben.

11.4.3 DE ENTITY DEFINIËREN DIE BIJ DE TABLE CURSISTEN HOORT

Je voegt aan `Opleidingen.edmx` een entity toe die hoort bij de table `Cursisten`:

- Je klikt met de rechtermuisknop in de achtergrond van `Opleidingen.edmx`.
- Je kiest `Update Model from Database`.
- Je klapte in het tabblad `Add` het onderdeel `Tables` open.
- Je klapte daarbinnen `dbo` open.
- Je plaatst een vinkje bij `Cursisten`
- Je kiest `Finish`.

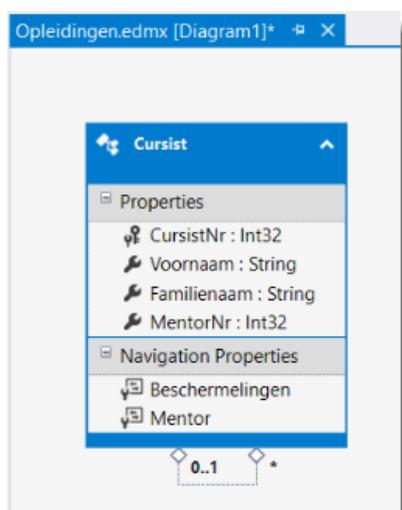
Je ziet een nieuwe entity `Cursisten`

Je ziet in deze entity twee navigation properties: `Cursisten1` en `Cursisten2`. Aan de hand van deze property namen kan je de betekenis niet inschatten.

- Als je Cursisten1 aanklikt, zie je in het properties venster bij **Multiplicity * (Many)**
 - Hieruit kan je afleiden dat de navigation property **Cursisten1** de **cursisten** voorstelt die bij de **mentor** horen (de beschermelingen van de mentor).
 - Je hernoemt de navigation property **Cursisten1** dus naar **Beschermelingen**
- Als je Cursisten2 aanklikt, zie je in het properties venster bij **Multiplicity 0..1 (Zero or One)**
 - Je kan hieruit afleiden dat de navigation property **Cursisten2** de mentor voorstelt van een cursist (als die een mentor heeft).
 - Je hernoemt de navigation property **Cursisten2** dus naar **Mentor**

Je wijzigt de naam van de entity **Cursisten** naar **Cursist**.

Het eindresultaat:



11.4.4 DE ENTITIES GEBRUIKEN VANUIT JE CODE

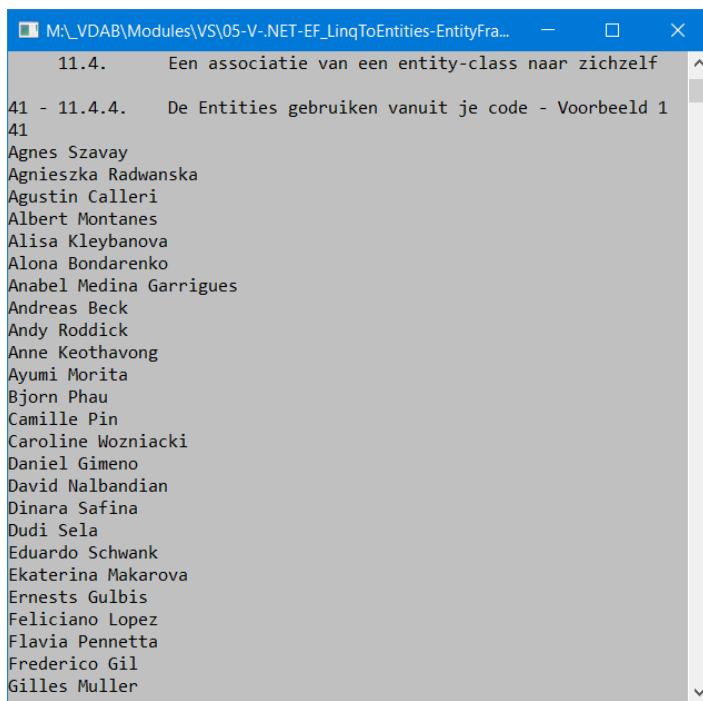
Je kan deze nieuwe entities nu gebruiken vanuit je code (**Program.cs**)

Voorbeeld 1 in de method **Main**:

Je toont de **cursisten** die nog geen **mentor** hebben:

```
using (var entities = new EFopleidingenEntities())
{
    var query = from cursist in entities.Cursisten
                where cursist.Mentor == null
                orderby cursist.Voornaam, cursist.Familienaam
                select cursist;

    foreach (var cursist in query)
    {
        Console.WriteLine("{0} {1}", cursist.Voornaam, cursist.Familienaam);
    }
}
```



The screenshot shows a Windows Notepad window with the following content:

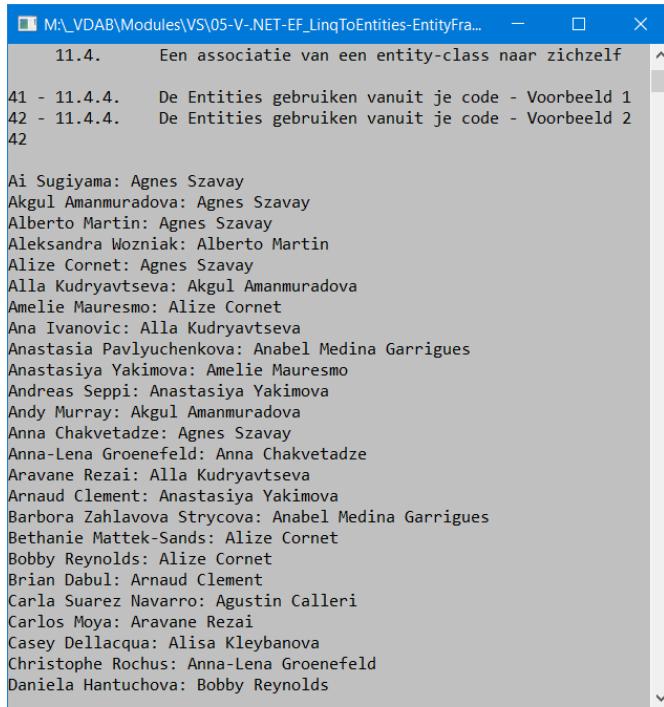
```
M:\VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFra...
11.4. Een associatie van een entity-class naar zichzelf
41 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 1
41
Agnes Szavay
Agnieszka Radwanska
Agustin Calleri
Albert Montanes
Alisa Kleybanova
Alona Bondarenko
Anabel Medina Garrigues
Andreas Beck
Andy Roddick
Anne Keothavong
Ayumi Morita
Bjorn Phau
Camille Pin
Caroline Wozniacki
Daniel Gimeno
David Nalbandian
Dinara Safina
Dudi Sela
Eduardo Schwank
Ekaterina Makarova
Ernests Gulbis
Feliciano Lopez
Flavia Pennetta
Frederico Gil
Gilles Muller
```

Voorbeeld 2 in de method Main:

Je toont de **cursisten** die wel een **mentor** hebben én hun **mentor**:

```
using (var entities = new EFopleidingenEntities())
{
    var query = from cursist in entities.Cursisten.Include("Mentor")
                where cursist.Mentor != null
                orderby cursist.Voornaam, cursist.Familienaam
                select cursist;

    foreach (var cursist in query)
    {
        var mentor = cursist.Mentor;
        Console.WriteLine("{0} {1}: {2} {3}", cursist.Voornaam, cursist.Familienaam,
                           mentor.Voornaam, mentor.Familienaam);
    }
}
```



The screenshot shows a Windows command prompt window with the title '11.4. Een associatie van een entity-class naar zichzelf'. The window displays a list of names and their associations:

```
M:\_VDAB\Modules\VS\05-V-NET-EF_LinqToEntities-EntityFra... - X
11.4. Een associatie van een entity-class naar zichzelf
41 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 1
42 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 2
42

Ai Sugiyama: Agnes Szavay
Akgul Amanmuradova: Agnes Szavay
Alberto Martin: Agnes Szavay
Aleksandra Wozniak: Alberto Martin
Alize Cornet: Agnes Szavay
Alla Kudryavtseva: Akgul Amanmuradova
Amelie Mauresmo: Alize Cornet
Ana Ivanovic: Alla Kudryavtseva
Anastasia Pavlyuchenkova: Anabel Medina Garrigues
Anastasiya Yakimova: Amelie Mauresmo
Andreas Seppi: Anastasiya Yakimova
Andy Murray: Akgul Amanmuradova
Anna Chakvetadze: Agnes Szavay
Anna-Lena Groenefeld: Anna Chakvetadze
Aravane Rezai: Alla Kudryavtseva
Arnaud Clement: Anastasiya Yakimova
Barbora Zahlavova Strycova: Anabel Medina Garrigues
Bethanie Mattek-Sands: Alize Cornet
Bobby Reynolds: Alize Cornet
Brian Dabul: Arnaud Clement
Carla Suarez Navarro: Agustin Calleri
Carlos Moya: Aravane Rezai
Casey Dellacqua: Alisa Kleybanova
Christophe Rochus: Anna-Lena Groenefeld
Daniela Hantuchova: Bobby Reynolds
```

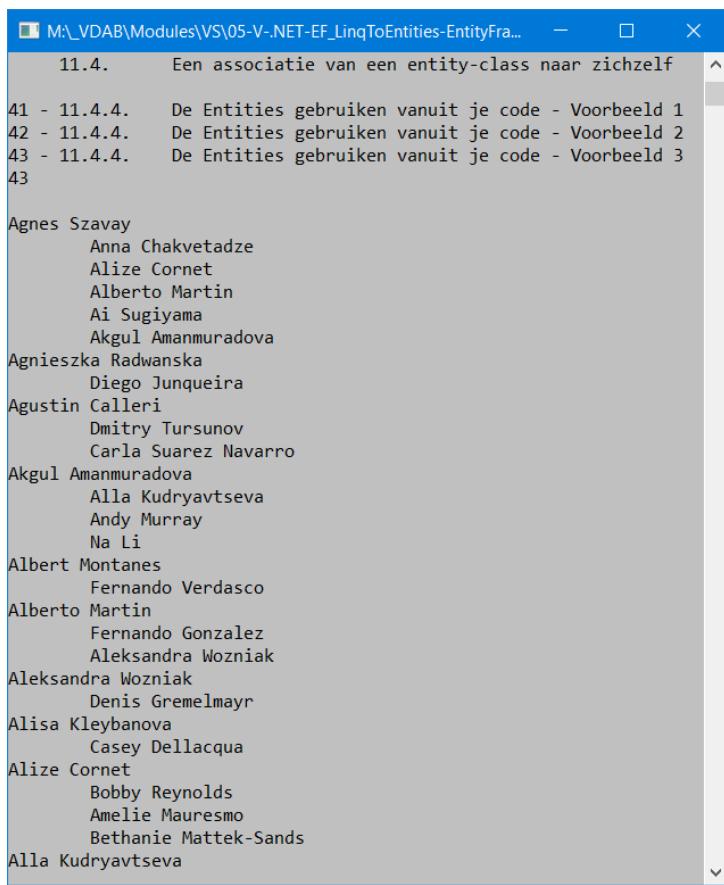
Voorbeeld 3 in de method Main:

Je toont **cursisten** die **mentor** zijn en hun **beschermelingen**:

```
using (var entities = new EFopleidingenEntities())
{
    var query = from mentor in entities.Cursisten.Include("Beschermelingen")
                where mentor.Beschermelingen.Count != 0
                orderby mentor.Voornaam, mentor.Familienaam
                select mentor;

    foreach (var mentor in query)
    {
        Console.WriteLine("{0} {1}", mentor.Voornaam, mentor.Familienaam);

        foreach (var beschermeling in mentor.Beschermelingen)
        {
            Console.WriteLine("\t{0} {1}", beschermeling.Voornaam,
                            beschermeling.Familienaam);
        }
    }
}
```



The screenshot shows a Windows Notepad window with the following content:

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFra...
11.4. Een associatie van een entity-class naar zichzelf
41 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 1
42 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 2
43 - 11.4.4. De Entities gebruiken vanuit je code - Voorbeeld 3
43

Agnes Szavay
    Anna Chakvetadze
    Alize Cornet
    Alberto Martin
    Ai Sugiyama
    Akgul Amanmuradova
Agnieszka Radwanska
    Diego Junqueira
Agustin Calleri
    Dmitry Tursunov
    Carla Suarez Navarro
Akgul Amanmuradova
    Alla Kudryavtseva
    Andy Murray
    Na Li
Albert Montanes
    Fernando Verdasco
Alberto Martin
    Fernando Gonzalez
    Aleksandra Wozniak
Aleksandra Wozniak
    Denis Gremelmayr
Alisa Kleybanova
    Casey Dellacqua
Alize Cornet
    Bobby Reynolds
    Amelie Mauresmo
    Bethanie Mattek-Sands
Alla Kudryavtseva
```

Voorbeeld 4 in de method **Main**:

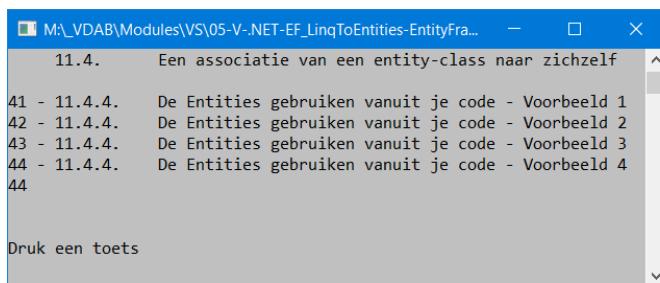
cursist 5 wordt de mentor van cursist 6:

```
using (var entities = new EFopleidingenEntities())
{
    var cursist5 = entities.Cursisten.Find(5);

    if (cursist5 != null)
    {
        var cursist6 = entities.Cursisten.Find(6);

        if (cursist6 != null)
        {
            cursist5.Beschermelingen.Add(cursist6);
            entities.SaveChanges();
        }
        else
        {
            Console.WriteLine("Cursist 6 niet gevonden");
        }
    }
    else
    {
        Console.WriteLine("Cursist 5 niet gevonden");
    }
}
```

	CursistNr	Voornaam	Familienaam	MentorNr
1	Agnes	Szavay	NULL	
2	Agnieszka	Radwanska	NULL	
3	Agustin	Calleri	NULL	
4	Ai	Sugiyama	1	
5	Akgul	Amanmura...	1	
▶ 6	Albert	Montanes	NULL	
7	Alberto	Martin	1	
8	Aleksandra	Wozniak	7	
9	Alisa	Kleybanova	NULL	



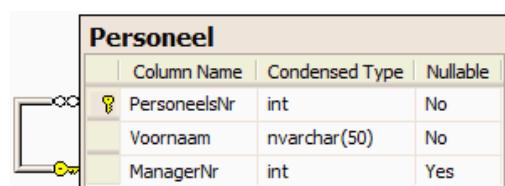
	CursistNr	Voornaam	Familienaam	MentorNr
1	Agnes	Szavay	NULL	
2	Agnieszka	Radwanska	NULL	
3	Agustin	Calleri	NULL	
4	Ai	Sugiyama	1	
5	Akgul	Amanmura...	1	
▶ 6	Albert	Montanes	5	
7	Alberto	Martin	1	
8	Aleksandra	Wozniak	7	
9	Alisa	Kleybanova	NULL	

11.5 TAAK 08 : PERSONEEL

Het script **PersonnelToevoegen.sql** voegt aan de database **EFBank** een table **Personnel** toe.

Je voert dit script uit.

De table **Personnel**:



Personnelsleden waarvan de kolom **ManagerNr** leeg is, staan hoogst in de hiërarchie.

Je voegt aan je EDM een bijbehorende entity class **Personnelslid** toe.

Je toont alle personeelsleden, beginnend met personeelsleden hoogst in hiërarchie. Je toont per personeelslid de ondergeschikten. Iedere keer je een lager niveau toont, toon je een extra tab-teken voor de voornaam:

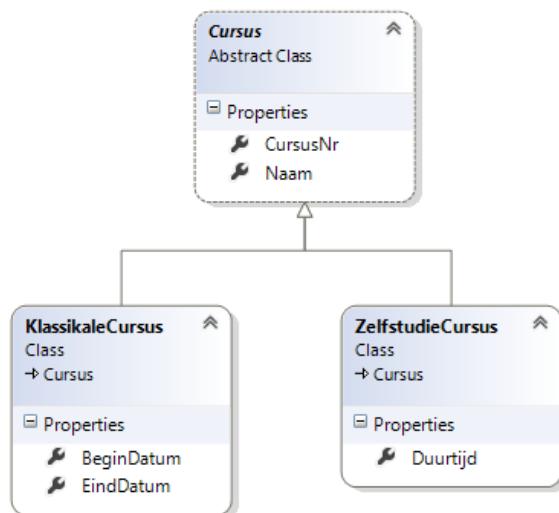


	Andy
	Peter
	Tom
Gerard	
	Loui
	Pamela
	Larry
	Barry
	Martin
Anthony	
	Leslie
	July
	Steve
	Foon Yue
	George
Mami	
	Yoshimi
Jeff	

12 INHERITANCE

12.1 ALGEMEEN

Inheritance bestaat in C# maar niet in een database. Je kan inheritance nabootsen in de database, op drie manieren. Je leert in dit hoofdstuk hoe EF daarmee omgaat. De voorbeeld-entities:



12.2 TABLE PER CONCRETE CLASS (TPC)

12.2.1 ALGEMEEN

Table per concrete class is de eerste manier om inheritance na te bootsen in de database.

- Er is één table per concrete class (**TPCKlassikaleCursus**, **TPCZelfstudieCursus**).
- Er is geen table voor abstracte classes (**TPCCursus**).

12.2.2 DE TABLES TPCKLASSIKALECURSUSSEN EN TPCZELFSTUDIECURSUSSEN TOEVOEGEN AAN DE DATABASE

Het script **TablePerConcreteClass.sql** maakt in de database **EFOpleidingen** de tables **TPCKlassikaleCursussen** en **TPCZelfstudieCursussen** toe. Je voert dit script uit in de **SQL Server Management Studio**.

Deze tables hebben volgende structuur:

TPCKlassikaleCursussen			TPCZelfstudieCursussen		
Column Name	Data Type	Allow Nulls	Column Name	Data Type	Allow Nulls
CursusNr	int	<input type="checkbox"/>	CursusNr	int	<input type="checkbox"/>
Naam	varchar(...)	<input type="checkbox"/>	Naam	varchar(50)	<input type="checkbox"/>
Van	datetime	<input type="checkbox"/>	Duurtijd	int	<input type="checkbox"/>
Tot	datetime	<input type="checkbox"/>			<input type="checkbox"/>

Klassikale cursussen

	CursusNr	Naam	Van	Tot
▶	1	Frans voor beginners	1/06/2009 0:00:00	5/06/2009 0:00:00
	3	Frans voor gevorderden	8/06/2009 0:00:00	12/06/2009 0:00:00
	5	Engels voor beginners	15/06/2009 0:00:00	19/06/2009 0:00:00
	7	Engels voor gevorderden	22/06/2009 0:00:00	26/06/2009 0:00:00

Zelfstudie cursussen

	CursusNr	Naam	Duurijd
▶	2	Franse correspondentie	5
	4	Engelse correspondentie	5

Bij table per concrete class mogen de entities over de bijbehorende tables heen geen dubbele primary-key-waarden hebben. In ons voorbeeld mag een record uit de table **TPCKlassikaleCursussen** niet eenzelfde primary key waarde hebben als een record uit de table **TPCZelfstudieCursussen**. Anders krijg je runtime-fouten als je straks de records aanspreekt vanuit EF.

De oplossing in ons voorbeeld is als volgt:

- De primary key van de table **TPCKlassikaleCursussen** is gebaseerd op de kolom **CursusNr**.

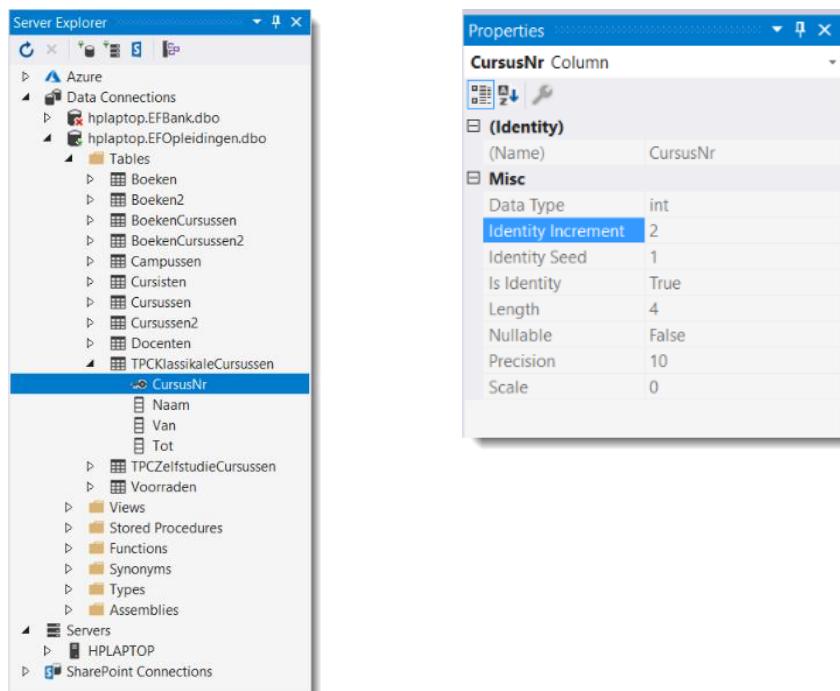
Dit is een autonumber-kolom. De nummering begint vanaf 1 en verhoogt per nieuw record met 2. De nummering van de records is dus: 1, 3, 5, 7, ...

- De primary key van de table **TPCZelfstudieCursussen** is ook gebaseerd op de kolom **CursusNr**.

Dit is een autonumber-kolom. De nummering begint vanaf 2 en verhoogt per nieuw record met 2. De nummering van de records is dus: 2, 4, 6, 8, ...

Je kan dit inzien vanuit Visual Studio:

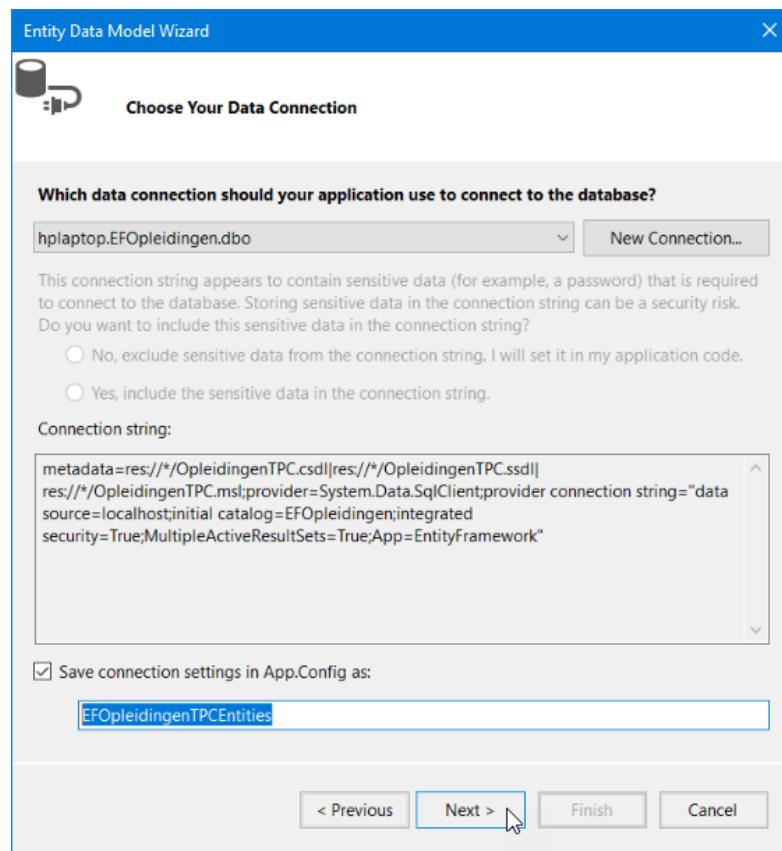
- Je klapst in de **server explorer** de databaseconnectie naar de database **EFopleidingen** open.
- Je klapst daarbinnen het onderdeel **Tables** open.
- Je klapst daarbinnen de table **TPCKlassikaleCursussen** open.
- Je selecteert de kolom **CursusNr**.
- Je ziet in het Properties-venster bij **Identity Seed 1** (begin nummering).
- Je ziet bij **Identity Increment 2** (verhoging van de nummering).



12.2.3 DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TABLES TPCKLASSIKALECURSUSSEN, EN TPCZELFSTUDIECURSUSSEN

Je voegt aan het project een **Entity Data Model** toe:

- Je klikt in de **Solution Explorer** met de rechtermuisknop op je project (**EFCursus**), je kiest **Add** en daarna **New Item**.
- Je kiest in het middendeel **ADO.NET Entity Data Model**.
- Je tikt bij **Name** **OpleidingenTPC** en je kiest **Add**.
- Je krijgt een vraag: **What should the model contain?**
- Je kiest **EF Designer from database** en je kiest **Next**.
- Je kiest bij **Which data connection should your application use to connect to the database?** De connectie **EFOpleidingen.dbo** zoals die gedefinieerd is in de **Server Explorer**.
- Als de vraag **Do you want to include this sensitive data in the connection string?** beschikbaar is, antwoord je **Yes**.
- Je laat het vinkje staan bij **Save connection settings in App.Config as** staan.
- Visual Studio maakt in dit configuratiebestand van je applicatie een onderdeel **EFOpleidingenTPCEntities** waarin de databaseconnectie gedefinieerd is. Pas dat aan.
- Je kiest **Next**.

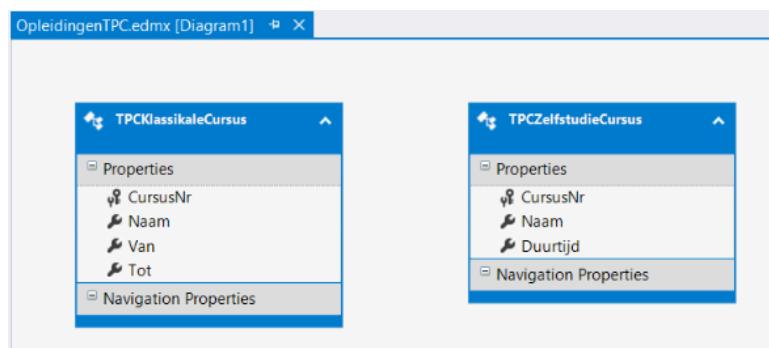


- Indien gevraagd kies je **Entity Framework 6.x** en je kiest **Next**
- Je klap bij **Which database objects do you want to include in your model?** het onderdeel **Tables** open. Je klap daarbinnen **dbo** open.
- Je plaatst een vinkje bij **TPCKlassikaleCursussen** en **TPCZelfstudieCursussen**.
- Je plaatst geen vinkje bij **Pluralize or singularize generated object names**.
- Je laat het vinkje staan bij **Include foreign key columns in the model**.
- Je zet **Model Namespace** op **EFOpleidingenModelTPC**.
- Je kiest **Finish**. Opgepast ! Hier moet je eventjes geduld hebben.
- Indien nodig kies je **OK** bij de **Security warnings**.

Je ziet in de designer van **OpleidingenTPC.edmx** het ontwerp van de entity classes **TPCKlassikaleCursussen** en **TPCZelfstudieCursussen**.

Je corrigeert enkele namen:

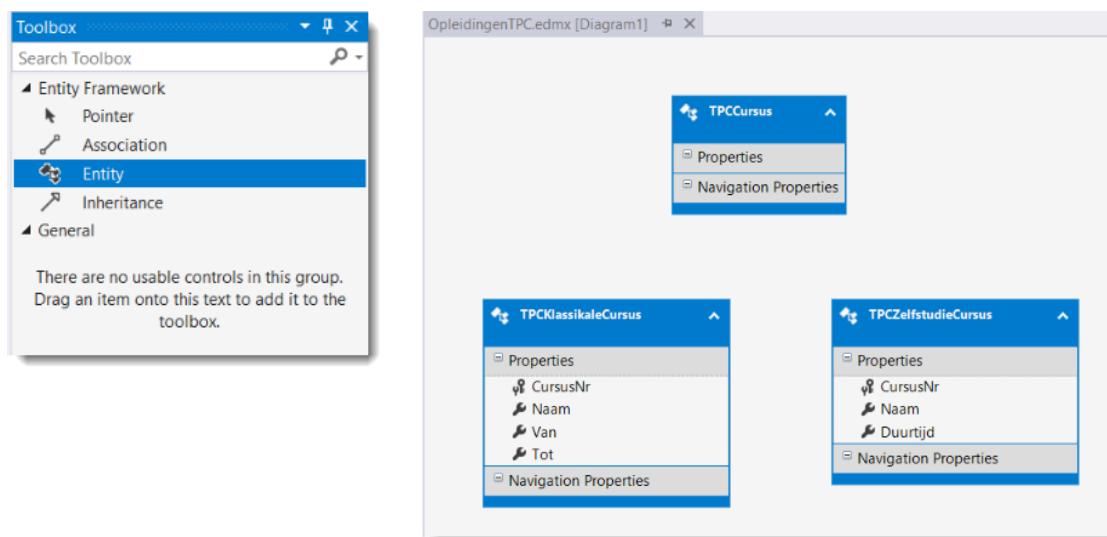
- Je wijzigt de naam van de entity **TPCKlassikaleCursussen** naar **TPCKlassikaleCursus**
- Je wijzigt de naam van de entity **TPCZelfstudieCursussen** naar **TPCZelfstudieCursus**



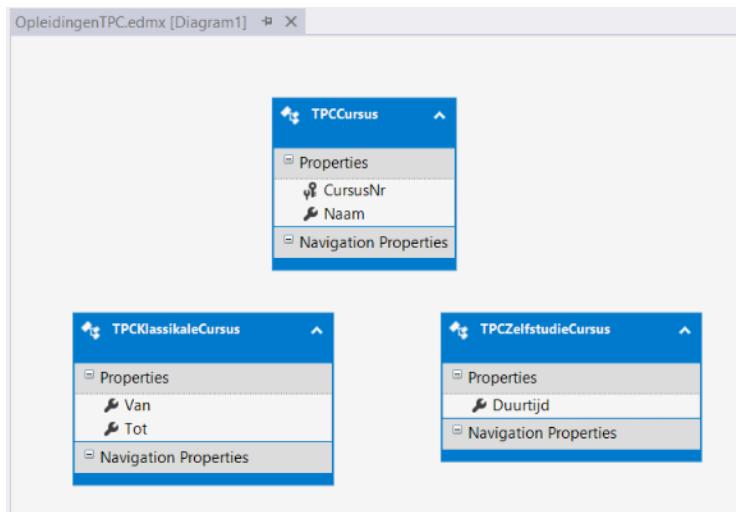
12.2.4 DE ENTITY CURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN

Op dit moment is er nog geen inheritance verband tussen **TPCKlassikaleCursus** en **TPCZelfstudieCursus**. Je maakt nu dit verband:

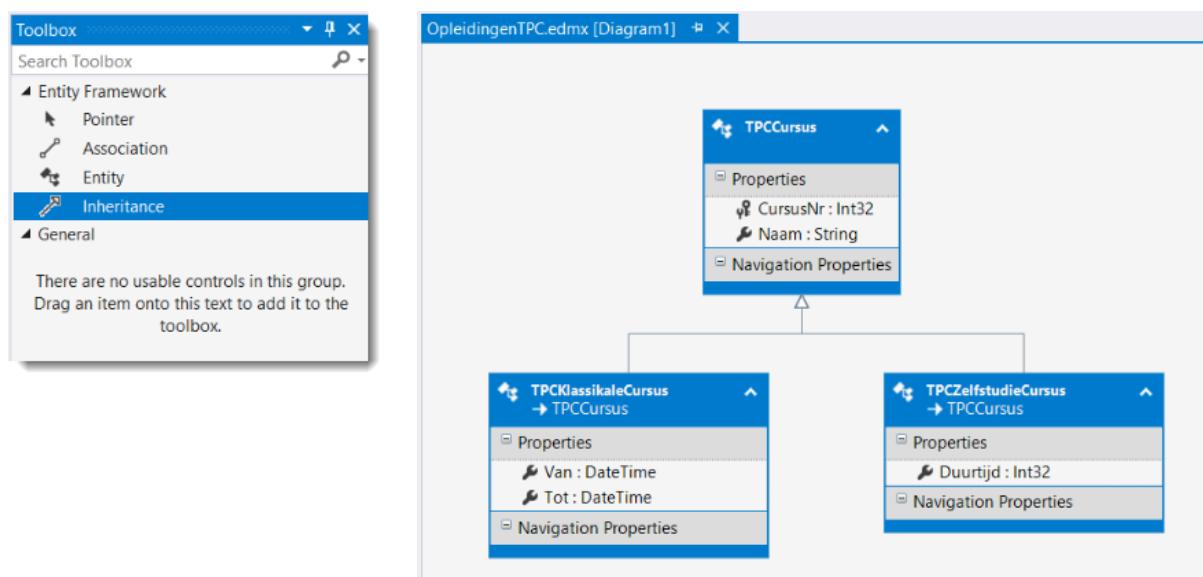
- Je maakt een nieuwe entity **TPCCursus**:
 - Je sleept vanuit de toolbox een **Entity** op de designer.
 - Je wijzigt de naam van **Entity1** naar **TPCCursus**
 - Je wijzigt in het properties venster **Entity Set Name** naar **TPCCursussen**
 - Je wijzigt in het properties venster **Abstract** naar **True**. Nu is **TPCCursus** abstract.
 - Je verwijdert in de entity **TPCCursus** de property **Id**



- Je brengt de gemeenschappelijke properties **CursusNr** en **Naam** van **TPCKlassikaleCursus** en **TPCZelfstudieCursus** over naar **TPCCursus**:
 - Je selecteert de properties **CursusNr** en **Naam** in **TPCKlassikaleCursus**. Je klikt met de rechtermuisknop in deze selectie en je kiest **Cut**
 - Je selecteert **Cursus** met de rechtermuisknop en je kiest **Paste**.
 - Je verwijdert de properties **CursusNr** en **Naam** in **TPCZelfstudieCursus**



- Je definieert de inheritance tussen **TPCKlassikaleCursus** en **TPCCursus** en tussen **TPCZelfstudieCursus** en **TPCCursus**:
 - Je kiest in de toolbox een Inheritance en sleept van **TPCKlassikaleCursus** naar **TPCCursus**.
 - Je kiest in de toolbox een Inheritance en sleept van **TPCZelfstudieCursus** naar **TPCCursus**.



Je moet nog definiëren welke kolommen (uit de tables van de database) horen bij de properties **CursusNr** en **Naam** van de entity **TPCCursus**.

Je kan dit niet definiëren in de designer, wel in de achterliggende XML:

- Je sluit het venster met de designer van **OpleidingenTPC.edmx** en je slaat daarbij op.
- Je klikt in de **Solution Explorer** met de rechtermuisknop op **OpleidingenTPC.edmx**.
- Je kiest **Open With**
- Je kiest **XML (Text) Editor** en je kiest **OK**.
- De XML bevat een element **edmx:Runtime**
- Dit element bevat een element **edmx:Mappings**
In dit element worden onder andere properties aan kolommen gekoppeld.
- Dit element bevat een element **EntitySetMapping Name="TPCCursussen"**
- Je wijzigt dit element als volgt:

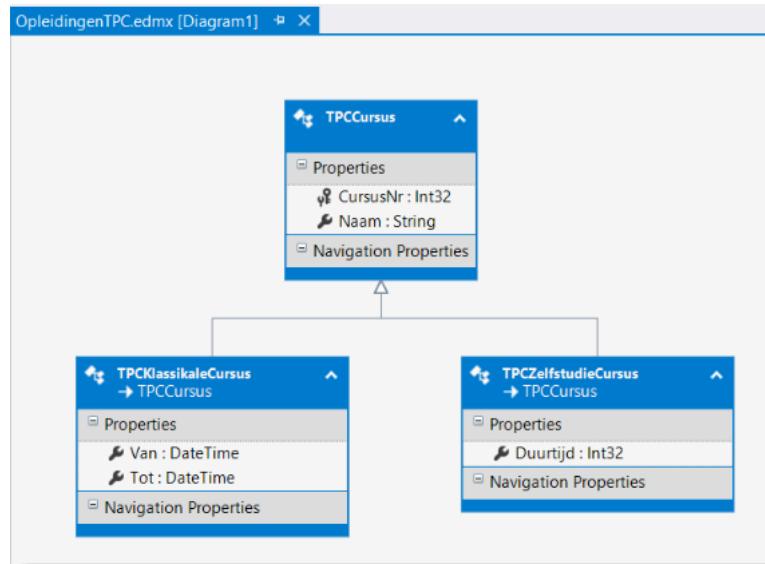
```

<EntitySetMapping Name="TPCCursussen">
  <EntityTypeMapping
    TypeName="IsTypeOf(EF0pleidingenModelTPC.TPCKlassikaleCursus)">
    <MappingFragment StoreEntitySet="TPCKlassikaleCursussen">
      <ScalarProperty Name="CursusNr" ColumnName="CursusNr"/>      <!--(1)-->
      <ScalarProperty Name="Naam" ColumnName="Naam"/>            <!--(2)-->
      <ScalarProperty Name="Tot" ColumnName="Tot" />
      <ScalarProperty Name="Van" ColumnName="Van" />
    </MappingFragment>
  </EntityTypeMapping>
  <EntityTypeMapping
    TypeName="IsTypeOf(EF0pleidingenModelTPC.TPCZelfstudieCursus)">
  
```

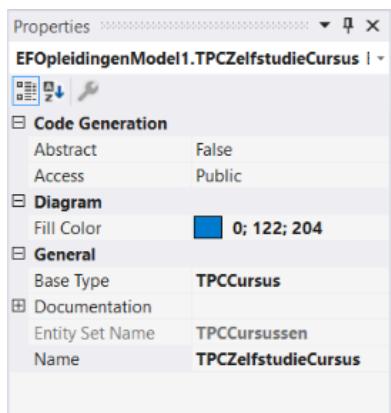
```
<MappingFragment StoreEntitySet="TPCZelfstudieCursussen">
    <ScalarProperty Name="CursusNr" ColumnName="CursusNr" />      <!--(3)-->
    <ScalarProperty Name="Naam" ColumnName="Naam" />            <!--(4)-->
    <ScalarProperty Name="Duurtijd" ColumnName="Duurtijd" />
</MappingFragment>
</EntityTypeMapping>
</EntitySetMapping>
```

- (1) Je geeft op deze extra lijn aan dat de property **CursusNr** van de entity **TPCKlassikaleCursus** hoort bij de kolom **CursusNr** van de table **TPCKlassikaleCursussen**.
- (2) Je geeft op deze extra lijn aan dat de property **Naam** van de entity **TPCKlassikaleCursus** hoort bij de kolom **Naam** van de table **TPCKlassikaleCursussen**.
- (3) Je geeft op deze extra lijn aan dat de property **CursusNr** van de entity **TPCZelfstudieCursus** hoort bij de kolom **CursusNr** van de table **TPCZelfstudieCursussen**.
- (4) Je geeft op deze extra lijn aan dat de property **Naam** van de entity **TPCZelfstudieCursus** hoort bij de kolom **Naam** van de table **TPCZelfstudieCursussen**.

Je sluit de XML visie van **OpleidingenTPC.edmx**. In de design view zien de entites er als volgt uit:



Als je de entity **TPCKlassikaleCursus** of de entity **TPCZelfstudieCursus** selecteert, zie je in het properties-venster dat de **Entity Set Name** property de waarde **TPCCursussen** bevat. Je kan dit niet wijzigen.



De property **Entity Set Name** van derived entities (**TPCKlassikaleCursus**, **TPCZelfstudieCursus**) is altijd gelijk aan de **Entity Set Name** property van hun base entity (**TPCCursus**).

Een instance van de object context heeft dus een property **TPCCursussen**, maar geen property **TPCKlassikaleCursussen** of **TPCZelfstudieCursussen**.

```
using (var entities = new OpleidingenTPCEntities())
{
    var query = from cursus in entities.TPCCursussen select cursus;
    --var query = from cursus in entities.KlassikaleCursussen select cursus;
}
```

Je kan dus enkel een **LINQ** query uitvoeren op de property **Cursussen**.

Als je enkel de **klassikale cursussen** wil lezen, geef je dit aan in het where deel van je query:

```
using (var entities = new OpleidingenTPCEntities())
{
    var query = from cursus in entities.TPCCursussen
                where cursus is TPCKlassikaleCursus
                select cursus;
}
```

12.2.5 DE ENTITIES GEBRUIKEN VANUIT JE CODE

Voorbeeld 1 in de method **Main**:

Je toont de **naam** van alle **cursussen**:

```
using (var entities = new EFopleidingenTPCEntities())
{
    var query = from cursus in entities.TPCCursussen
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        Console.WriteLine(cursus.Naam);
    }
}
```

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solu...
12.      Inheritance
12.2.     Veel-op-veel-associaties zonder extra associatieinformatie
45 - 12.2.5. TPC - De entities gebruiken vanuit je code - Voorbeeld 1
45

Engels voor beginners
Engels voor gevorderden
Engelse correspondentie
Frans voor beginners
Frans voor gevorderden
Franse correspondentie

Druk een toets
```

Je toont in dit voorbeeld de naam van iedere **cursus**, maar niet of het om een **klassikale cursus** of een **zelfstudiecursus** gaat.

Je kan aan een object de naam van de class vragen waartoe het object behoort. Je voert daar toe op dat object de method **GetType()** uit. Deze method geeft je een object van de class **Type**. Je vraagt aan dit object de property **Name**. Je wijzigt de regel

```
Console.WriteLine(...)
```

naar

```
Console.WriteLine(cursus.Naam + ' ' + cursus.GetType().Name);
```

```
using (var entities = new EFopleidingenTPCEntities())
{
    var query = from cursus in entities.TPCCursussen
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        //Console.WriteLine(cursus.Naam);
        Console.WriteLine(cursus.Naam + ' ' + cursus.GetType().Name);
    }
}
```

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solu...
12.      Inheritance
12.2.     Veel-op-veel-associaties zonder extra associatieinformatie
45 - 12.2.5. TPC - De entities gebruiken vanuit je code - Voorbeeld 1
45

Engels voor beginners TPCKlassikaleCursus
Engels voor gevorderden TPCKlassikaleCursus
Engelse correspondentie TPCZelfstudieCursus
Frans voor beginners TPCKlassikaleCursus
Frans voor gevorderden TPCKlassikaleCursus
Franse correspondentie TPCZelfstudieCursus

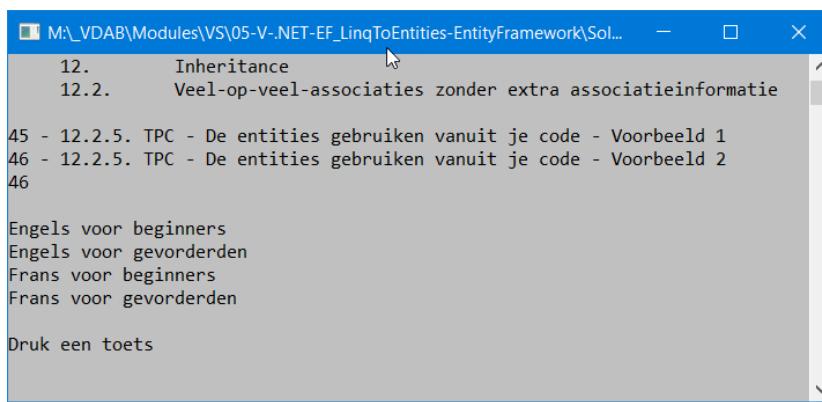
Druk een toets
```

Voorbeeld 2 in de method **Main**:

Je toont enkel **klassikale cursussen**:

```
using (var entities = new EFopleidingenTPCEntities())
```

```
{  
    var query = from cursus in entities.TPCCursussen  
                where cursus is TPCKlassikaleCursus  
                orderby cursus.Naam  
                select cursus;  
  
    foreach (var cursus in query)  
    {  
        Console.WriteLine(cursus.Naam);  
    }  
}
```



```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Sol...  
12.      Inheritance  
12.2.     Veel-op-veel-associaties zonder extra associatieinformatie  
45 - 12.2.5. TPC - De entites gebruiken vanuit je code - Voorbeeld 1  
46 - 12.2.5. TPC - De entites gebruiken vanuit je code - Voorbeeld 2  
46  
Engels voor beginners  
Engels voor gevorderden  
Frans voor beginners  
Frans voor gevorderden  
Druk een toets
```

Voorbeeld 3 in de method Main:

Je voegt een **zelfstudie cursus** toe:

```
using (var entities = new EFopleidingenTPCEntities())  
{  
    Console.WriteLine("Voor aanpassing:");  
  
    var query = from cursus in entities.TPCCursussen  
                where cursus is TPCZelfstudieCursus  
                orderby cursus.Naam  
                select cursus;  
  
    foreach (var cursus in query)  
    {  
        Console.WriteLine(cursus.Naam);  
    }  
  
    entities.TPCCursussen.Add(new TPCZelfstudieCursus  
    { Naam = "Spaanse correspondent", Duurtijd = 6 });  
  
    entities.SaveChanges();  
  
    Console.WriteLine("\nNa aanpassing:");  
  
    query = from cursus in entities.TPCCursussen  
            where cursus is TPCZelfstudieCursus  
            orderby cursus.Naam  
            select cursus;  
  
    foreach (var cursus in query)  
    {  
        Console.WriteLine(cursus.Naam);  
    }  
}
```

```

M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Sol...
12. Inheritance
12.2. Veel-op-veel-associaties zonder extra associatieinformatie

45 - 12.2.5. TPC - De entites gebruiken vanuit je code - Voorbeeld 1
46 - 12.2.5. TPC - De entites gebruiken vanuit je code - Voorbeeld 2
47 - 12.2.5. TPC - De entites gebruiken vanuit je code - Voorbeeld 3

47

Voor aanpassing:
Engelse correspondentie
Franse correspondentie

Na aanpassing:
Engelse correspondentie
Franse correspondentie
Spaanse correspondentie

```

12.3 TABLE PER HIERARCHY (TPH)

Dit is de tweede manier om inheritance na te bootsen in de database. Hierbij bevat de database één table voor alle classes uit de inheritance hiérarchy.

Het script [TablePerClassHierarchy.sql](#) voegt aan de database [EFopleidingen](#) de table [TPHCursussen](#) toe. Je voert dit script uit in de [SQL Server Management Studio](#).

Je houdt in de kolom [SoortCursus](#) bij of het om een [klassikale cursus](#) of een [zelfstudie cursus](#) gaat:

- [K](#) betekent [klassikale cursus](#).
- [Z](#) betekent [zelfstudie-cursus](#).

TPHCursussen			
	Column Name	Data Type	Allow Nulls
1	CursusNr	int	<input type="checkbox"/>
	Naam	varchar(50)	<input type="checkbox"/>
	Van	datetime	<input checked="" type="checkbox"/>
	Tot	datetime	<input checked="" type="checkbox"/>
	Duurtijd	int	<input checked="" type="checkbox"/>
	SoortCursus	char(1)	<input type="checkbox"/> <input type="checkbox"/>

	CursusNr	Naam	Van	Tot	Duurtijd	SoortCursus
▶	1	Frans voor beginners	1/06/2009 0:00:00	5/06/2009 0:00:00	NULL	K
	2	Frans voor gevorderden	8/06/2009 0:00:00	12/06/2009 0:00:00	NULL	K
	3	Engels voor beginners	15/06/2009 0:00:00	19/06/2009 0:00:00	NULL	K
	4	Engels voor gevorderden	22/06/2009 0:00:00	26/06/2009 0:00:00	NULL	Z
	5	Franse correspondentie	NULL	NULL	5	Z
	6	Engelse correspondentie	NULL	NULL	5	Z

12.3.1 DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TPHCURSUSSEN

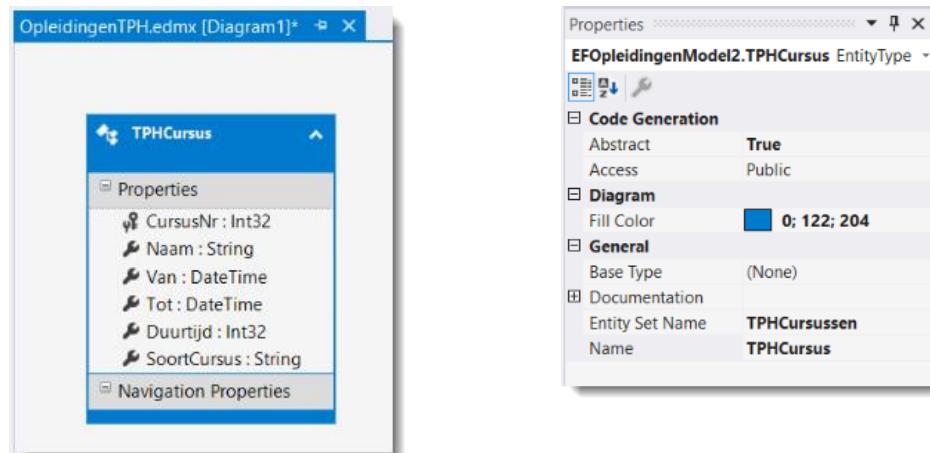
Je voegt aan het project een [Entity Data Model](#) toe:

- Je klikt in de [Solution Explorer](#) met de rechtermuisknop op je project ([EFCursus](#)), je kiest [Add](#) en daarna [New Item](#).

- Je kiest in het middendeel **ADO.NET Entity Data Model**.
- Je tikt bij **"Name"** **OpleidingenTPH** en je kiest **Add**.
- Je krijgt een vraag: **What should the model contain?**
- Je kiest **EF Designer from database** en je kiest **Next**.
- Je kiest bij **Which data connection should your application use to connect to the database?** De connective **EFOpleidingen.dbo** zoals die gedefinieerd is in de **Server Explorer**.
- Als de vraag **Do you want to include this sensitive data in the connection string?** beschikbaar is, antwoord je **Yes**.
- Je laat het vinkje staan bij **Save connection settings in App.Config as** staan.
- Visual Studio maakt in dit configuratiebestand van je applicatie een onderdeel **EFOpleidingenTPHEntities** waarin de databaseconnectie gedefinieerd is. Pas dat aan.
- Je kiest **Next**.
- Indien gevraagd kies je **Entity Framework 6.x** en je kiest **Next**
- Je klapt bij **Which database objects do you want to include in your model?** het onderdeel **Tables** open. Je klapt daarbinnen **dbo** open.
- Je plaatst een vinkje bij **TPHCursussen**.
- Je plaatst geen vinkje bij **Pluralize or singularize generated object names**.
- Je laat het vinkje staan bij **Include foreign key columns in the model**.
- Bij **Model Namespace** vermeld je **EFOpleidingenModelTPH**.
- Je kiest **Finish**. Opgepast ! Hier moet je eventjes geduld hebben.
- Je kiest **OK** bij de eventuele **Security warnings**.

Je doet enkele aanpassingen in de entity **TPHCursussen**.

- Je wijzigt de naam van de entity **TPHCursussen** naar **TPHCursus**
- Je wijzigt in het properties venster **Entity Set Name** naar **TPHCursussen**.
- Je wijzigt in het properties venster **Abstract** naar **True**.

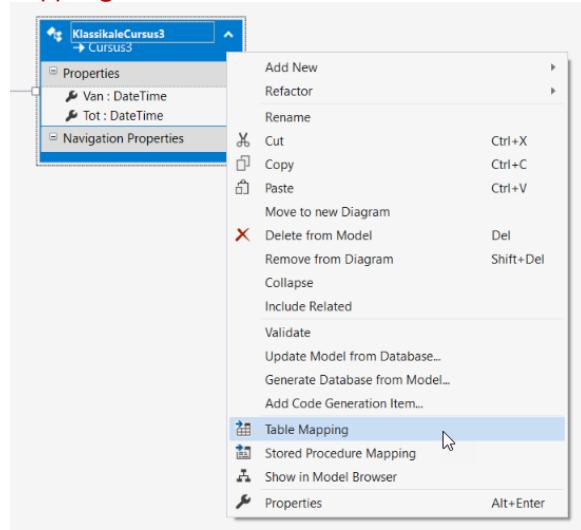


12.3.2 DE ENTITY KЛАSSIKALECURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN

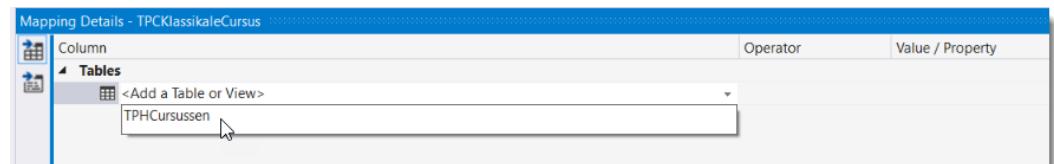
- Je maakt een nieuwe entity **TPHKlassikaleCursus**:
 - Je sleept vanuit de toolbox een **Entity** op de designer.
 - Je wijzigt de naam van **Entity1** naar **TPHKlassikaleCursus**
 - Je verwijdert de property **Id**
- Je definieert de **inheritance** tussen **TPHKlassikaleCursus** en **TPHCursus**
 - Je kiest in de toolbox een **Inheritance** en sleept van **TPHKlassikaleCursus**

naar **TPHCursus**.

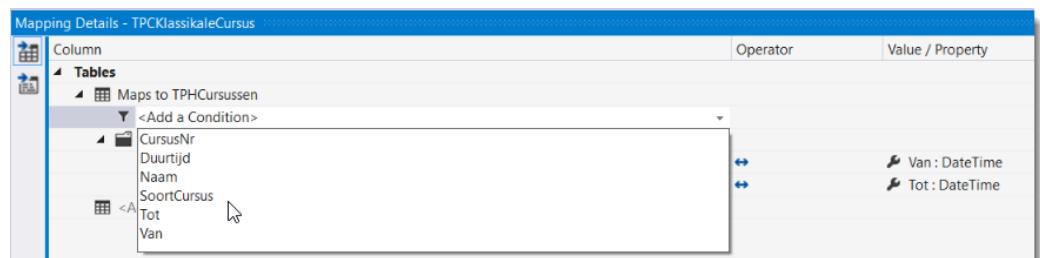
- Je brengt de properties die specifiek zijn voor een **klassikale cursus** over van **TPCHursus** naar **TPHKlassikaleCursus**
 - Je selecteert de properties **Van** en **Tot** in **TPHCursus**.
Je klikt met de rechtermuisknop in deze selectie en je kiest **Cut**
 - Je selecteert **TPHKlassikaleCursus** met de rechtermuisknop en je kiest **Paste**.
- Je associeert **TPHKlassikaleCursus** met de records uit de table **TPHCursussen** die in de kolom **SoortCursus** de waarde **K** hebben
 - Je selecteert **TPHKlassikaleCursus** met de rechtermuisknop en je kiest **Table Mapping**



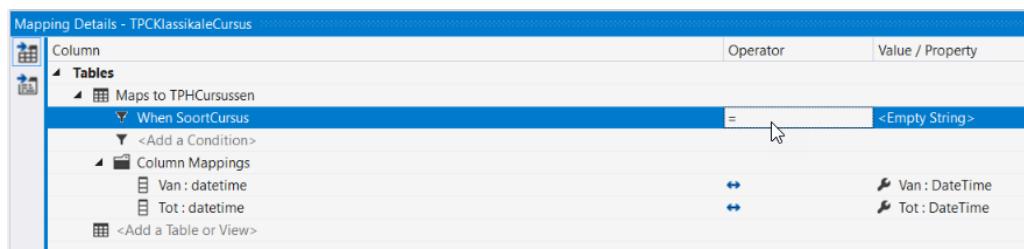
- Je selecteert <Add a Table or View> en je kiest **TPHCursussen**.



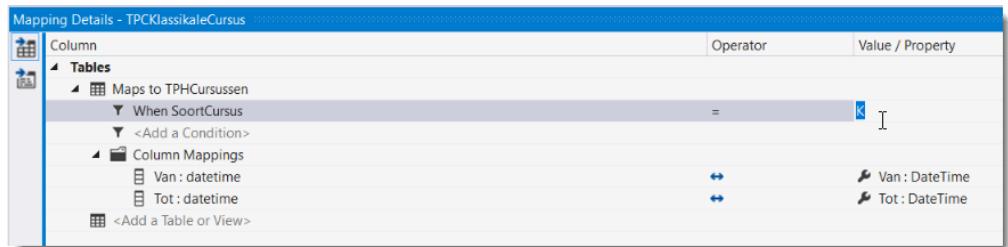
- Je selecteert <Add a Condition> en je kiest **SoortCursus**



- Je laat Operator op de waarde =

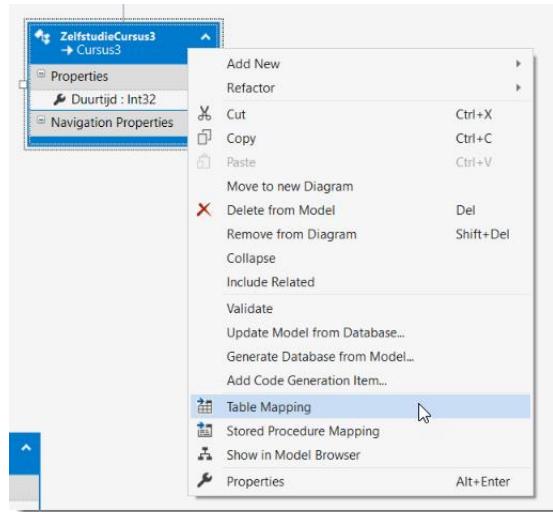


- Je tikt bij **Value/Property** de waarde **K**

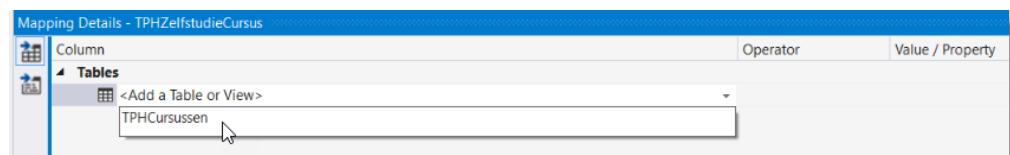


12.3.3 DE ENTITY TPHZELFSTUDIECURSUS TOEVOEGEN EN INHERITANCE DEFINIËREN

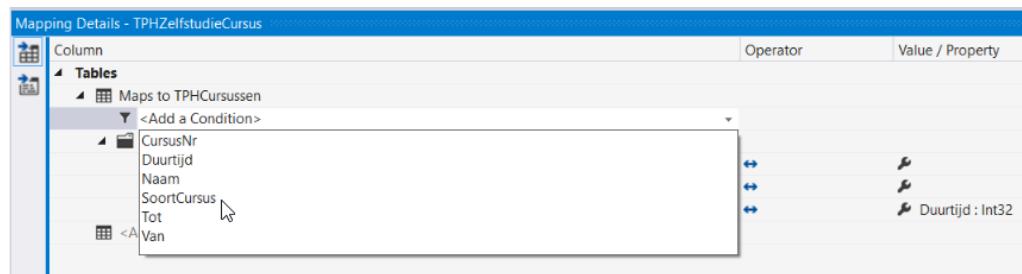
- Je maakt een nieuwe entity **TPHZelfstudieCursus**:
 - Je sleept vanuit de toolbox een **Entity** op de designer.
 - Je wijzigt de naam van **Entity1** naar **TPHZelfstudieCursus**
 - Je verwijdert de property **Id**
- Je definieert de **inheritance** tussen **TPHZelfstudieCursus** en **TPHCursus**
 - Je kiest in de toolbox een Inheritance en sleept van **TPHZelfstudieCursus** naar **TPHCursus**.
- Je brengt de property die specifiek is voor een **zelfstudie cursus** over van **TPHCursus** naar **TPHZelfstudieCursus**
 - Je selecteert de property **Duurijd** in **TPHCursus** met de rechtermuisknop en je kiest **Cut**
 - Je selecteert **TPHZelfstudieCursus** met de rechtermuisknop en je kiest **Paste**.
- Je associeert **TPHZelfstudieCursus** met de records uit de table **TPHCursussen** die in de kolom **SoortCursus** de waarde **Z** hebben
 - Je selecteert **TPHZelfstudieCursus** met de rechtermuisknop en je kiest **Table Mapping**



- Je selecteert <Add a Table or View> en je kiest TPHCursussen.



- Je selecteert <Add a Condition> en je kiest SoortCursus



- Je laat Operator op de waarde =



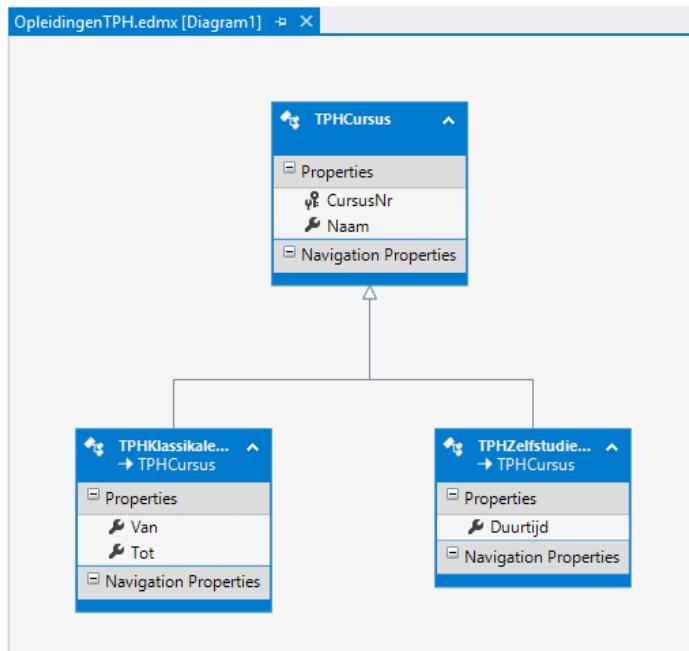
- Je tikt bij Value/Property de waarde Z



- Je verwijdert de property **SoortCursus** uit **TPHCursus**. Deze dient enkel om intern in EF het onderscheid te maken tussen een **klassikale cursussen** en **zelfstudie cursussen**.

Je hoeft de letter in de kolom niet rechtstreeks te lezen in de rest van de applicatie.

Het eindresultaat in de designer is hetzelfde als op het einde van hoofdstuk 12.2.4



12.3.4 DE ENTITIES GEBRUIKEN VANUIT JE CODE

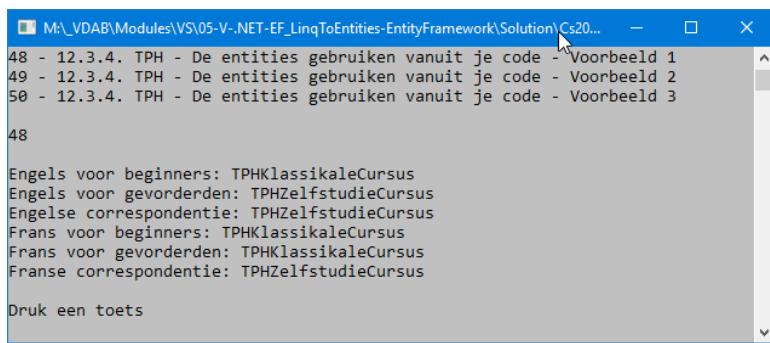
Je gebruikt deze entites op juist dezelfde manier als bij Table per concrete class.

Voorbeeld 1 in de method **Main**:

Je toont de naam en soort van alle **cursussen**:

```
using (var entities = new EFOpleidingenTPHEntities())
{
    var query = from cursus in entities.TPHCursussen
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        Console.WriteLine("{0}: {1}", cursus.Naam, cursus.GetType().Name);
    }
}
```



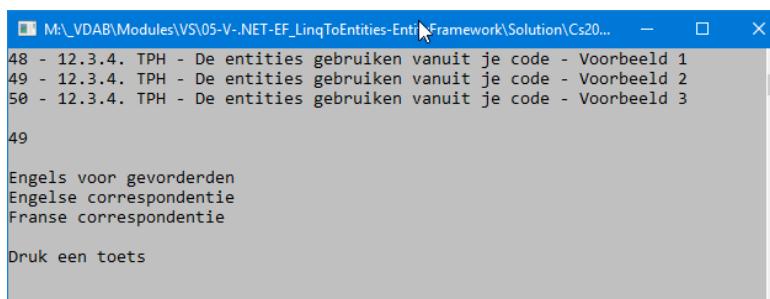
```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs20...
48 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 1
49 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 2
50 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 3
48
Engels voor beginners: TPHKlassikaleCursus
Engels voor gevorderden: TPHZelfstudieCursus
Engelse correspondentie: TPHZelfstudieCursus
Frans voor beginners: TPHKlassikaleCursus
Frans voor gevorderden: TPHKlassikaleCursus
Franse correspondentie: TPHZelfstudieCursus
Druk een toets
```

Voorbeeld 2 in de method Main:

Je toont enkel van **zelfstudie cursussen** de naam:

```
using (var entities = new EFopleidingenTPHEntities())
{
    var query = from cursus in entities.TPHCursussen
                where cursus is TPHZelfstudieCursus
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        Console.WriteLine(cursus.Naam);
    }
}
```



```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs20...
48 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 1
49 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 2
50 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 3
49
Engels voor gevorderden
Engelse correspondentie
Franse correspondentie
Druk een toets
```

Voorbeeld 3 in de method Main:

Je voegt een **zelfstudie cursus** toe:

```
using (var entities = new EFopleidingenTPHEntities())
{
    entities.TPHCursussen.Add(new TPHZelfstudieCursus
        { Naam = "Duitse correspondentie", Duurtijd = 6 });
    entities.SaveChanges();
}
```

CursusNr	Naam	Van	Tot	Duurijd	SoortCursus
1	Frans voor beginners	1/06/2009 0:00:00	5/06/2009 0:00:00	NULL	K
2	Frans voor gevorderden	8/06/2009 0:00:00	12/06/2009 0:00:00	NULL	K
3	Engels voor beginners	15/06/2009 0:00:00	19/06/2009 0:00:00	NULL	K
4	Engels voor gevorderden	22/06/2009 0:00:00	26/06/2009 0:00:00	NULL	Z
5	Franse correspondentie	NULL	NULL	5	Z
6	Engelse correspondentie	NULL	NULL	5	Z

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs20...
48 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 1
49 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 2
50 - 12.3.4. TPH - De entites gebruiken vanuit je code - Voorbeeld 3
50

Druk een toets
```

	CursusNr	Naam	Van	Tot	Duurijd	SoortCursus
	1	Frans voor beginners	1/06/2009 0:00:00	5/06/2009 0:00:00	NULL	K
	2	Frans voor gevorderden	8/06/2009 0:00:00	12/06/2009 0:00:00	NULL	K
	3	Engels voor beginners	15/06/2009 0:00:00	19/06/2009 0:00:00	NULL	K
	4	Engels voor gevorderden	22/06/2009 0:00:00	26/06/2009 0:00:00	NULL	Z
	5	Franse correspondentie	NULL	NULL	5	Z
	6	Engelse correspondentie	NULL	NULL	5	Z
▶	7	Duitse correspondentie	NULL	NULL	6	Z

12.4 TABLE PER TYPE (TPT)

12.4.1 DE TABLES TPTCURSUSSEN, TPTKLASSIKALECURSUSSEN EN TPTZELFSTUDIECURSUSSEN TOEVOEGEN AAN DE DATABASE

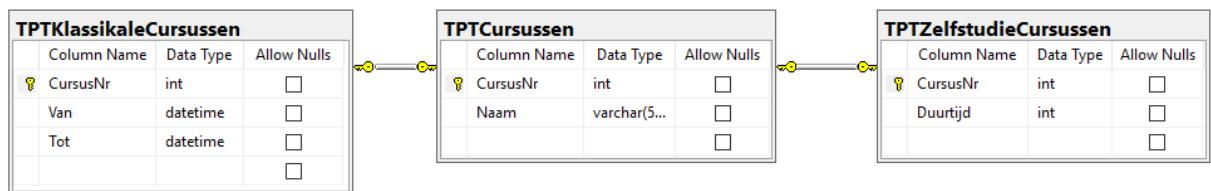
Table per type is de derde manier om inheritance na te bootsen in de database.

Bij deze manier bevat de database één table per class uit de inheritance hiërarchy. (**TPTCursus**, **TPTKlassikaleCursus** en **TPTZelfstudieCursus**).

Het script **TablePerSubClass.sql** voegt aan de database **EFOpleidingen** volgende tables toe: **TPTCursussen**, **TPTKlassikaleCursussen** en **TPTZelfstudieCursussen**.

Je voert dit script uit in de **SQL Server Management Studio**.

Deze tables hebben volgende structuur:



- Per **klassikale cursus** is er één record in de table **TPTCursussen** én één bijbehorend record in de table **TPTKlassikaleCursussen**. Beide records hebben dezelfde waarde in de kolom **CursusNr**.
- Per **zelfstudie-cursus** is er één record in de table **TPTCursussen** én één bijbehorend

record in de table **TPTZelfstudieCursussen**. Beide records hebben dezelfde waarde in de kolom **CursusNr**.

- De kolom **CursusNr** is in de table **TPTCursussen** een autonumber-kolom. De kolom **CursusNr** is in de tables **TPTKlassikaleCursussen** en **TPTZelfstudieCursussen** geen autonumber-veld.

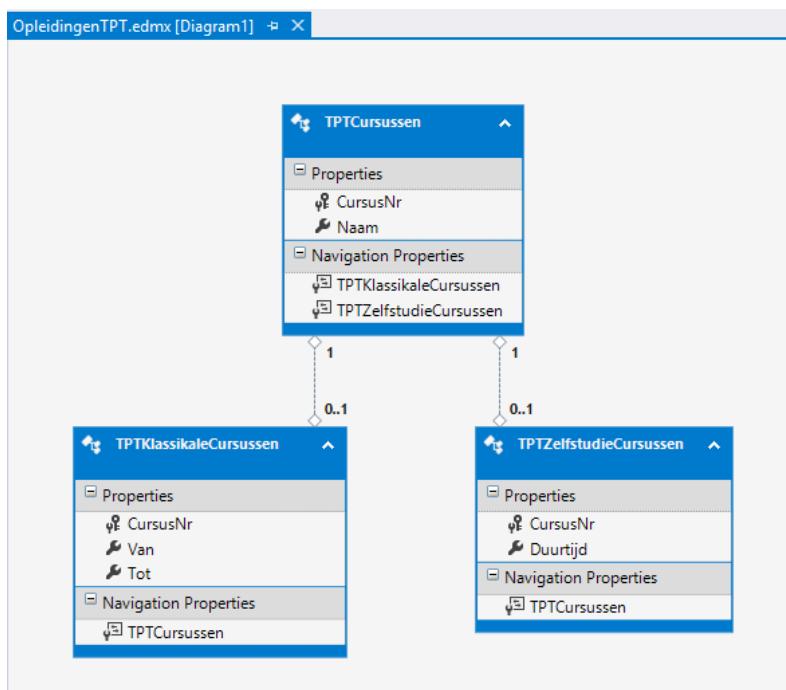
Het veld is wel primary key én tegelijk een foreign key die verwijst naar de kolom **CursusNr** in de table **TPTCursussen**. Je verzekert met deze foreign key dat in de tables **TPTKlassikaleCursussen** en **TPTZelfstudieCursussen** enkel records kunnen voorkomen die een bijbehorend record hebben in de table **TPTCursussen**.

12.4.2 DE ENTITIES DEFINIËREN DIE HOREN BIJ DE TABLES TPTCURSUSSEN, TPTKLASSIKALECURSUSSEN EN TPTZELFSTUDIECURSUSSEN

Je voegt aan het project een **Entity Data Model** toe:

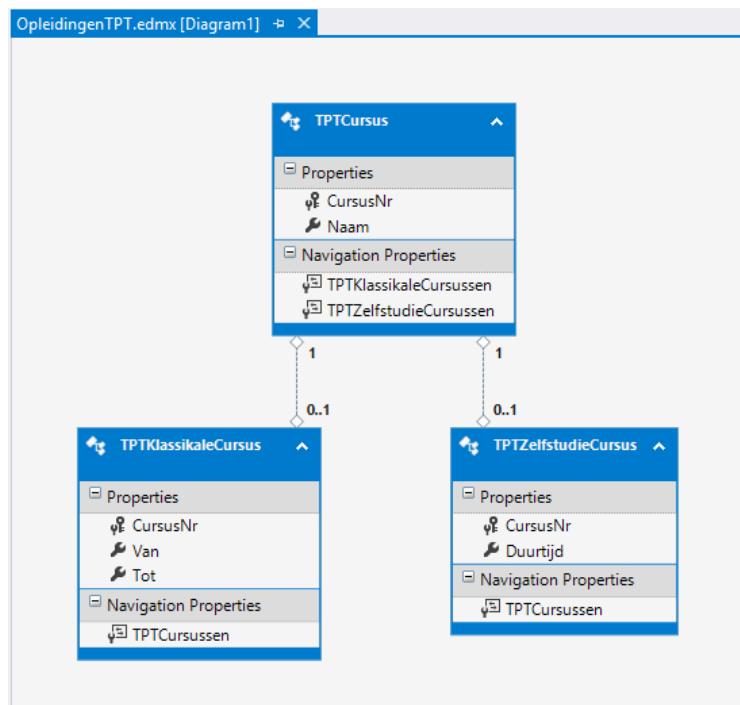
- Je klikt in de **Solution Explorer** met de rechtermuisknop op je project (**EFCursus**), je kiest **Add** en daarna **New Item**.
- Je kiest in het middendeel **ADO.NET Entity Data Model**.
- Je tikt bij **Name** **OpleidingenTPT** en je kiest **Add**.
- Je krijgt een vraag: **What should the model contain?**
- Je kiest **EF Designer from database** en je kiest **Next**.
- Je kiest bij **Which data connection should your application use to connect to the database?** De connectie **EFOpleidingen.dbo** zoals die gedefinieerd is in de **Server Explorer**.
- Als de vraag **Do you want to include this sensitive data in the connection string?** beschikbaar is, antwoord je **Yes**.
- Je laat het vinkje staan bij **Save connection settings in App.Config as** staan.
- Visual Studio maakt in dit configuratiebestand van je applicatie een onderdeel **EFOpleidingenTPTEntities** waarin de databaseconnectie gedefinieerd is. Pas dat aan.
- Je kiest **Next**.
- Indien gevraagd kies je **Entity Framework 6.x** en je kiest **Next**.
- Je klap bij **Which database objects do you want to include in your model?** het onderdeel **Tables** open. Je klap daarbinnen **dbo** open.
- Je plaatst een vinkje bij **TPTCursussen**, **TPTZelfstudieCursussen** en **TPTKlassikaleCursussen**.
- Je plaatst geen vinkje bij **Pluralize or singularize generated object names**.
- Je laat het vinkje staan bij **Include foreign key columns in the model**.
- Bij **Model Namespace** vermeld je **EFOpleidingenModelTPT**.
- Je kiest **Finish**. Opgepast ! Hier moet je eventjes geduld hebben.
- Je kiest **OK** bij de eventuele **Security warnings**.

Je ziet de entites **TPTCursussen**, **TPTKlassikaleCursussen** en **TPTZelfstudieCursussen**



Je corrigeert enkele namen:

- Je wijzigt de naam van de entity **TPTCursussen** naar **TPTCursus**
- Je wijzigt in het properties venster **Entity Set Name** naar **TPTCursussen**
- Je wijzigt in het properties venster **Abstract** naar **True**.
- Je wijzigt de naam van de entity **TPTKlassikaleCursussen** naar **TPTKlassikaleCursus**
- Je wijzigt de naam van de entity **TPTZelfstudieCursussen** naar **TPTZelfstudieCursus**



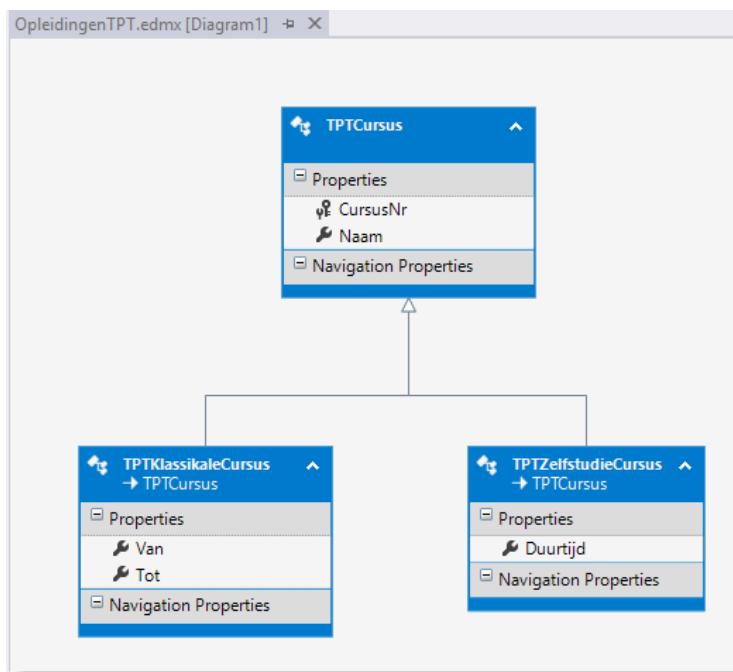
12.4.3 INHERITANCE DEFINIËREN

- Op dit moment is het verband tussen **TPTCursus** en **TPTKlassikaleCursus** geen

inheritance, maar een associatie **1 – 0..1**. Je verwijdert deze associatie

- Op dit moment is het verband tussen **TPTCursus** en **TPTZelfstudieCursus** geen inheritance, maar een associatie **1 – 0..1**. Je verwijdert deze associatie
- Je definieert een **inheritance** verband tussen **TPTKlassikaleCursus** en **TPTCursus**
 - Je kiest in de toolbox een Inheritance en sleept van **TPTKlassikaleCursus** naar **TPTCursus**.
 - Je verwijdert in de **TPTKlassikaleCursus** de property **CursusNr**. **TPTKlassikaleCursus** erft al een **CursusNr** van **Cursus**.
- Je definieert de **inheritance** tussen **TPTZelfstudieCursus** en **TPTCursus**
 - Je kiest in de toolbox een Inheritance en sleept van **TPTZelfstudieCursus** naar **TPTCursus**.
 - Je verwijdert in de **TPTZelfstudieCursus** de property **CursusNr**. **TPTZelfstudieCursus** erft al een **CursusNr** van **Cursus**.

Het eindresultaat in de designer is hetzelfde als op het einde van hoofdstuk 12.2.4



12.4.4 DE ENTITIES GEBRUIKEN VANUIT JE CODE

Je gebruikt deze entites op juist dezelfde manier als bij Table per concrete class.

Voorbeeld 1 in de method **Main**:

Je toont de **naam** en **soort** van alle **cursussen**:

```

using (var entities = new EFOPleidingenTPTEntities())
{
    var query = from cursus in entities.TPTCursussen
                orderby cursus.Naam
                select cursus;
  
```

```
foreach (var cursus in query)
{
    Console.WriteLine("{0}: {1}", cursus.Naam, cursus.GetType().Name);
}
```

```
51 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 1
52 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 2
53 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 3

51
Engels voor beginners: TPTKlassikaleCursus
Engels voor gevorderden: TPTKlassikaleCursus
Engelse correspondentie: TPTZelfstudieCursus
Frans voor beginners: TPTKlassikaleCursus
Frans voor gevorderden: TPTKlassikaleCursus
Franse correspondentie: TPTZelfstudieCursus

Druk een toets
```

Voorbeeld 2 in de method Main:

Je toont van alle **cursussen**, behalve **zelfstudiecursussen** de **naam**:

```
using (var entities = new EFopleidingenTPTEntities())
{
    var query = from cursus in entities.TPTCursussen
                where !(cursus is TPTZelfstudieCursus)
                orderby cursus.Naam
                select cursus;

    foreach (var cursus in query)
    {
        Console.WriteLine(cursus.Naam);
    }
}
```

```
51 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 1
52 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 2
53 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 3

52
Engels voor beginners
Engels voor gevorderden
Frans voor beginners
Frans voor gevorderden

Druk een toets
```

Voorbeeld 3 in de method Main:

Je voegt een **zelfstudie-cursus** toe:

```
using (var entities = new EFopleidingenTPTEntities())
{
    entities.TPTCursussen.Add(new TPTZelfstudieCursus
        { Naam = "Italiaanse correspondentie", Duurtijd = 6 });
    entities.SaveChanges();
}
```

CursusNr	Duurijd
5	5
6	5
8	6

CursusNr	Naam
1	Frans voor beginners
2	Engels voor beginners
3	Frans voor gevorderden
4	Engels voor gevorderden
5	Franse correspondentie
6	Engelse correspondentie

```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFramework\Solution\Cs20...
51 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 1
52 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 2
53 - 12.4.4. TPT - De entites gebruiken vanuit je code - Voorbeeld 3
53

Druk een toets
```

CursusNr	Duurijd
5	5
6	5
7	6

CursusNr	Naam
1	Frans voor beginners
2	Engels voor beginners
3	Frans voor gevorderden
4	Engels voor gevorderden
5	Franse correspondentie
6	Engelse correspondentie
8	Italiaanse correspondentie

12.5 TAAK 09 : ZICHTREKENINGEN – SPAARREKENINGEN

Je leidt twee derived classes af van de class **Rekening**:

- **Zichtrekening** De bijbehorende records hebben de waarde **Z** in de kolom **Soort**
- **Spaarrekening** De bijbehorende records hebben de waarde **S** in de kolom **Soort**

Je maakt van de class **Rekening** een abstract class.

Je toont daarna van **zichtrekeningen** het **rekeningnr.** en het **saldo**.

13 COMPLEX TYPES

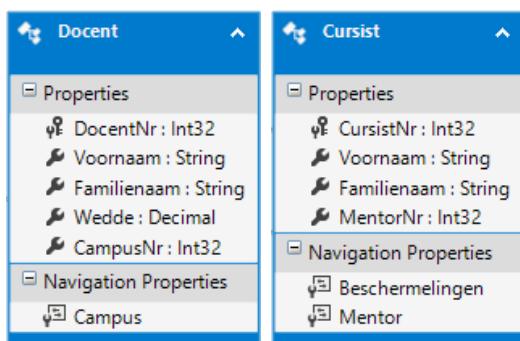
13.1 ALGEMEEN

Je leerde in het hoofdstuk **Error! Reference source not found.** (**Error! Reference source not found.**) dat je de data in een database maar op twee niveaus kan opsplitsen: tables en columns (binnen die tables).

Je hebt bij object-oriëntatie geen beperking bij het opsplitsen van data:

- een class kan gewone properties bevatten (`int, string, ...`)
- een class kan ook navigation properties bevatten die verwijzen naar andere classes. Deze andere classes kunnen zelf gewone properties en navigation properties bevatten ...

In het `Entity Data Model Opleidingen.edmx` komen de properties `Voornaam` en `Familienaam` als scalar properties voor in de entities `Docent` én `Cursist`:



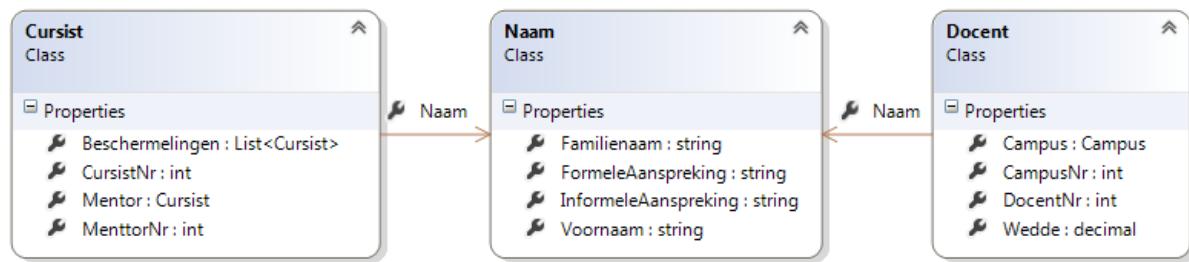
Een betere object-georiënteerde voorstelling van deze twee properties is als volgt:

- Je maakt een class `Naam`, die deze properties `Voornaam` en `Familienaam` bevat.
- Je kan in deze class ook properties toevoegen die typisch zijn voor een naam.

Voorbeelden:

- de property `InformeleAansprek` (deze geeft `Hello Eddy` terug als de voornaam `Eddy` is)
- de property `FormeleAansprek` (deze geeft `Geachte Eddy Wally` terug als de voornaam `Eddy` is en de familienaam `Wally` is).
- Je vervangt in `Docent` én `Cursist` de properties `Voornaam` en `Familienaam` door één property van het type `Naam`.
- Op deze manier hebben deze entites met deze property nog altijd een `voornaam` en een `familienaam`, maar je kan vanuit deze entites ook de properties `InformeleAansprek` en `FormeleAansprek` gebruiken, die één keer gedefinieerd zijn in de class `Naam`.

`Docent`, `Cursist` en `Naam` als een class diagram:



Een class, zoals **Naam**, die je gebruikt vanuit één of meerdere entities, noemt men een **complex type**.

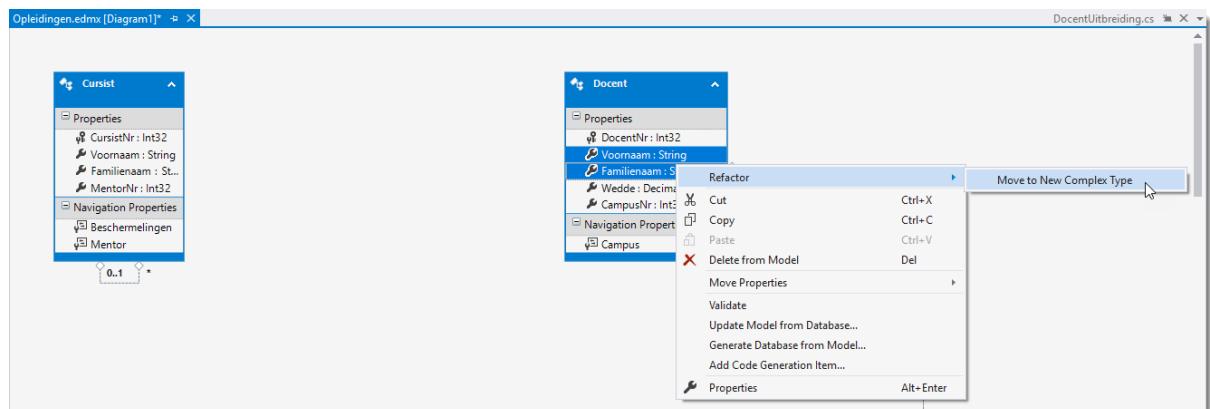
Een complex type heeft volgende beperkingen:

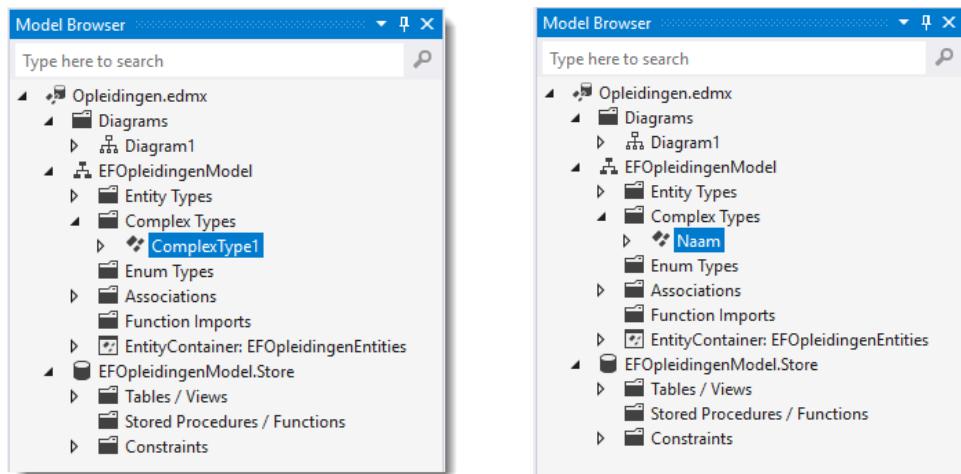
- Een complex type kan niet erven van een ander complex type.
- Een property die als type een complex type heeft, de property **Naam** in de entities **Docent** en **Cursist**, kan niet leeg zijn (nullable).

13.2 EEN COMPLEX TYPE MAKEN OP BASIS VAN EEN BESTAANDE ENTITY

Je kan een complex type maken op basis van een bestaande entity:

- Je kiest de entity properties die je wil overbrengen naar een complex type:
 - Je klikt in **Docent** op de property **Voornaam**.
 - Je geeft een **Ctrl+klik** op de property **Familienaam**.
- Je klikt met de rechtermuisknop op één van deze properties.
- Je kiest **Refactor** en daarna **Move to New Complex Type**.
- Je wijzigt in het venster **Model Browser** de naam van het nieuwe complex type van **ComplexType1** naar **Naam**.





Als je dit complex type openklapt in de **Model Browser**, zie je dat het de properties **Familienaam** en **Voornaam** bevat.



Visual Studio heeft in **Docent** de properties **Voornaam** en **Familienaam** vervangen door een property **ComplexProperty**.

Deze property heeft als type het complex type **Naam**.

Je wijzigt de property naam naar **Naam** (de property beschrijft de naam van de docent).



Visual Studio heeft de table mappings van de **Docent** juist aangepast:

- De property **Naam.Familienaam** hoort bij de kolom **Familienaam**
- De property **Naam.Voornaam** hoort bij de kolom **Voornaam**

Column	Operator	Value / Property
DocentNr : int		DocentNr : Int32
Voornaam : nvarchar		Naam.Voornaam : String
Familienaam : nvarchar		Naam.Familienaam : String
Wedde : decimal		Wedde : Decimal
CampusNr : int		CampusNr : Int32

Je verwijdert in **DocentUitbreiding.cs** de readonly property **Naam** die je vroeger maakte.

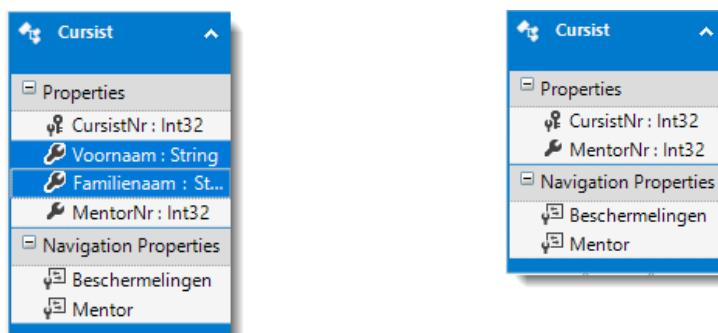
```
namespace EFCursus
{
    public partial class Docent
    {
        //public string Naam
        //{
        //    get
        //    {
        //        return Voornaam + ' ' + Familienaam;
        //    }
        //}

        public void Opslag(decimal bedrag)
        {
            Wedde += bedrag;
        }
    }
}
```

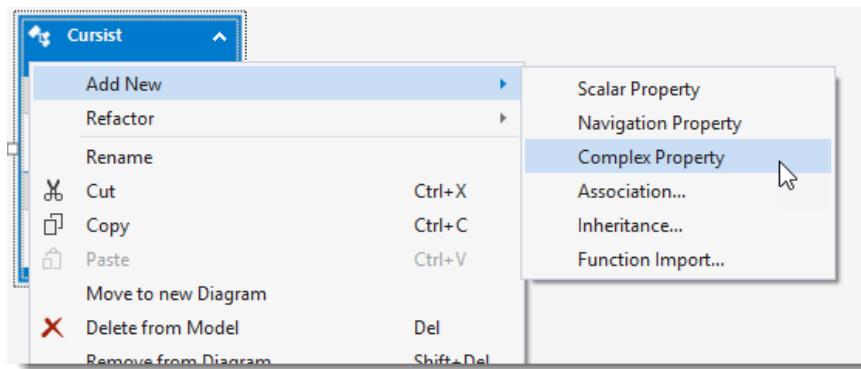
13.3 EEN COMPLEX TYPE HERBRUIKEN IN EEN ANDERE ENTITY

Je zal nu het complex type **Naam** herbruiken in **Cursist**:

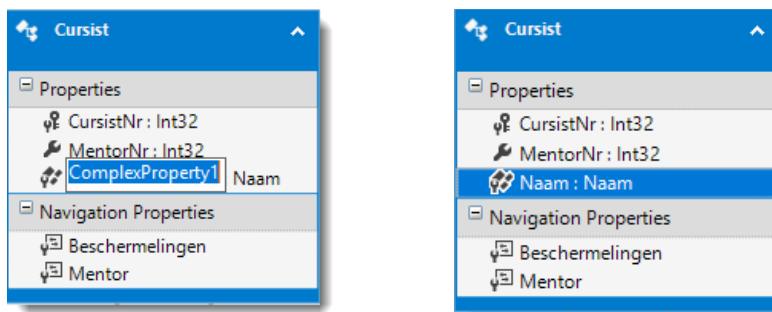
- Je verwijdert in **Cursist** de properties **Voornaam** en **Familienaam**



- Je voegt een property met als naam **Naam** en als type **Naam** (het complex type) toe:
 - Je klikt met de rechtermuisknop ergens in **Cursist**.
 - Je kiest **Add New** en daarna **Complex Property**

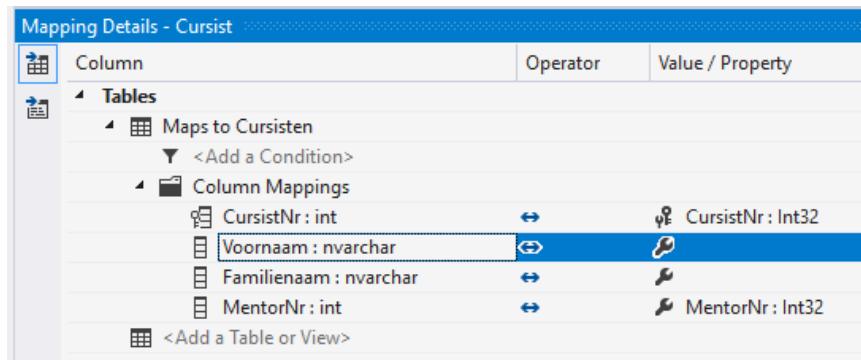


- Je wijzigt de naam van de property van **ComplexProperty** naar **Naam**. Het type van de property is het complex type **Naam**, omdat er momenteel maar één complex type is. Als je diagram meerdere complex types bevat, kan je het type van de property instellen in het properties-venster.



- Je corrigeert de table mappings van **Cursist**:

- Je klikt met de rechtermuisknop in **Cursist**.
- Je kiest **Table Mapping**.
- Je ziet dat de kolommen **Voornaam** en **Familienaam** niet meer gekoppeld zijn aan **Cursist** properties
- Je kiest bij de kolom **Voornaam** de property **Naam.Voornaam**
- Je kiest bij de kolom **Familienaam** de property **Naam.Familienaam**.



Column	Operator	Value / Property
CursistNr : int		CursistNr : Int32
Voornaam : nvarchar		Naam.Voornaam : String
Familienaam : nvarchar		Naam.Familienaam : String
MentorNr : int		MentorNr : Int32

13.4 COMPLEX TYPE ALS PARTIAL CLASS

Het complex type **Naam** is een **partial class**. Je kan deze class uitbreiden door een source file **NaamUitbreiding.cs** toe te voegen aan het project, waarin je dezelfde class definieert als een partial class. Je voegt in deze source properties en of methods toe:

```
namespace EFCursus
{
    public partial class Naam
    {
        public override string ToString()
        {
            return Voornaam + ' ' + Familienaam;
        }

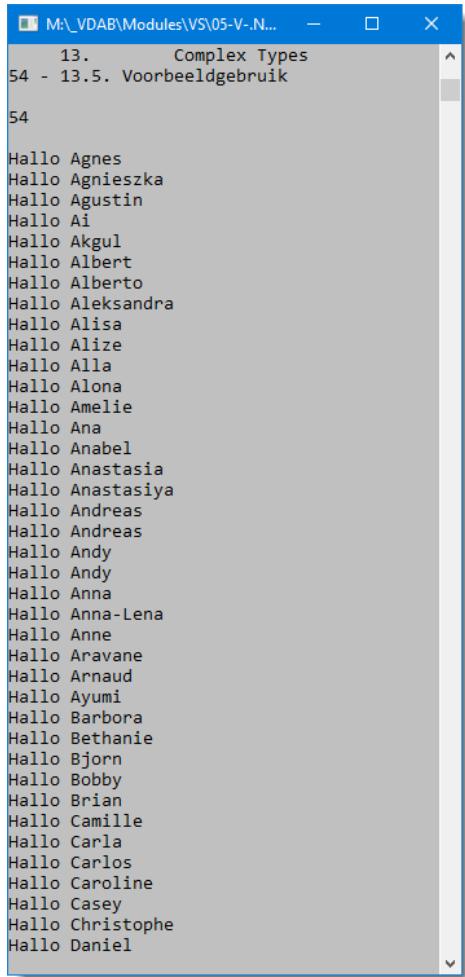
        public string InformeleBegroeting
        {
            get
            {
                return "Hallo " + Voornaam;
            }
        }

        public string FormeleBegroeting
        {
            get
            {
                return "Geachte " + Voornaam + ' ' + Familienaam;
            }
        }
    }
}
```

13.5 VOORBEELDGEbruik

Je kan als voorbeeld de cursisten aanspreken met hun informele begroeting in de method **Main**:

```
using (var entities = new EFopleidingenEntities())
{
    foreach (var cursist in (from eenCursist in entities.Cursisten select eenCursist))
    {
        Console.WriteLine(cursist.Naam.InformeleBegroeting);
    }
}
```



A screenshot of a Windows command-line window titled "13. Complex Types" with the subtitle "54 - 13.5. Voorbeeldgebruik". The window displays a list of names, each preceded by the text "Hallo". The names listed are: Agnes, Agnieszka, Agustin, Ai, Akgul, Albert, Alberto, Aleksandra, Alisa, Alize, Alla, Alona, Amelie, Ana, Anabel, Anastasia, Anastasiya, Andreas, Andreas, Andy, Andy, Anna, Anna-Lena, Anne, Aravane, Arnaud, Ayumi, Barbora, Bethanie, Bjorn, Bobby, Brian, Camille, Carla, Carlos, Caroline, Casey, Christophe, Daniel.

14 ENUMS

14.1 ALGEMEEN

Als de inhoud van een variabele één waarde kan zijn die je kiest uit een beperkt aantal waarden, is het in C# aan te raden

deze waarden te definiëren in een enum.

deze enum te gebruiken als type voor de variabele.

Voorbeelddeclaratie van een enum:

```
public enum Geslacht
{
    Man, Vrouw
}
```

Voorbeeldgebruik van deze enum in een variabele mijnGeslacht:

```
Geslacht mijnGeslacht = Geslacht.Man;
```

14.2 VOORDELEN VAN EEN ENUM

14.2.1 DE COMPILER CONTROLEERT DE INHOUD VAN EEN ENUM-VARIABELE

De C#-compiler controleert dat de variabele `mijnGeslacht` enkel de waarde `Geslacht.Man` of `Geslacht.Vrouw` kan bevatten. Je hebt meer kans op tikfouten als je geen `enum` gebruikt als type van de variabele `mijnGeslacht`, maar een `string`:

```
string mijnGeslacht = "D";
```

- De variabele bevat een verkeerde waarde (je verwachtte enkel “M” of “V”).
- De C#-compiler controleert de inhoud van een string echter niet.
- Je ontdekt deze fout hopelijk tijdens het uittesten van de code!

14.2.2 VISUAL STUDIO HELPT BIJ HET INVULLEN VAN EEN ENUM-VARIABELE

Visual Studio toont bij het invullen van een enum-variabele een popup venstertje met de mogelijkwaarden.



Visual Studio kan dit niet bij het invullen van een string-variabele.

14.3 ENUMS EN EF

- Het type van een property van een entity of complex type kan een enum zijn.
- Ook bij zo'n property hoort een kolom in de table in de database.
- Een database kent geen enum als type van een kolom van een table.
- Sommigen kiezen dan als kolomtype varchar, anderen kiezen int.
- EF kan een enum property associëren met een kolom in de table in de database, als het type van die kolom een **geheel getal** type is (jammer genoeg niet als het type van de kolom varchar)

is).

Je zal bijvoorbeeld van iedere **Docent**-entity een extra property **Geslacht** bijhouden.

Het type van deze property zal een enum **Geslacht** zijn.

Het script **GeslachtToevoegen.sql** voegt aan de table **Docenten** van de database **Opleidingen** een kolom **Geslacht** toe. Het type van deze kolom is **int**.

Dit script vult bij de bestaande records (mannen) de inhoud van deze kolom met **1**.

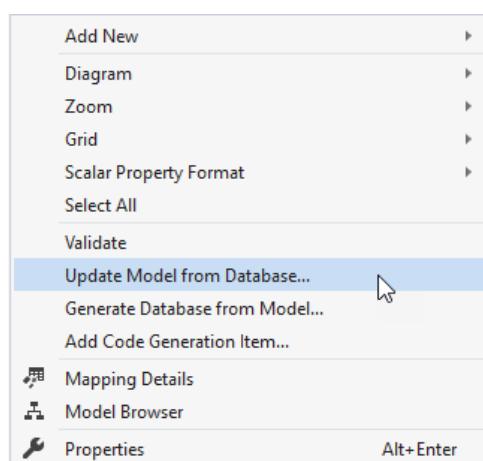
Dit script voegt ook enkele records met vrouwelijke docenten toe en vult de inhoud van de kolom **Geslacht** in deze records met **2**.

Het geslacht wordt dus in de kolom **Geslacht** voorgesteld met **1 (Man)** of **2 (Vrouw)**.

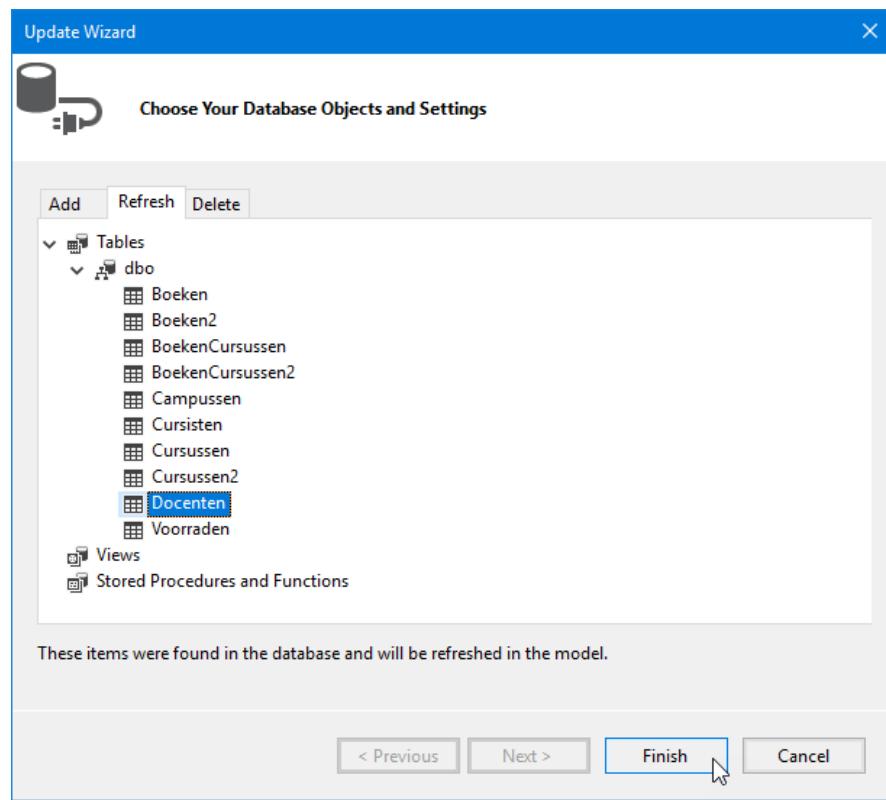
Je voert dit script uit in de **SQL Server Management Studio**.

Je zal dit getal in **Docent** voorstellen met een enum **Geslacht**:

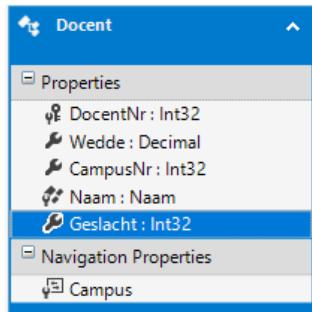
- Je klikt met de rechtermuisknop in **Opleidingen.edmx** en je kiest **Update Model from Database**.



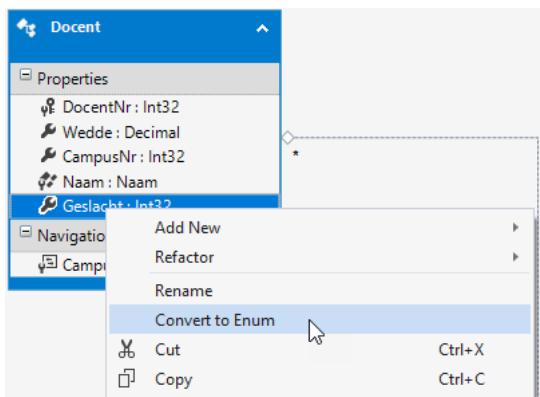
- Je kiest **Finish**



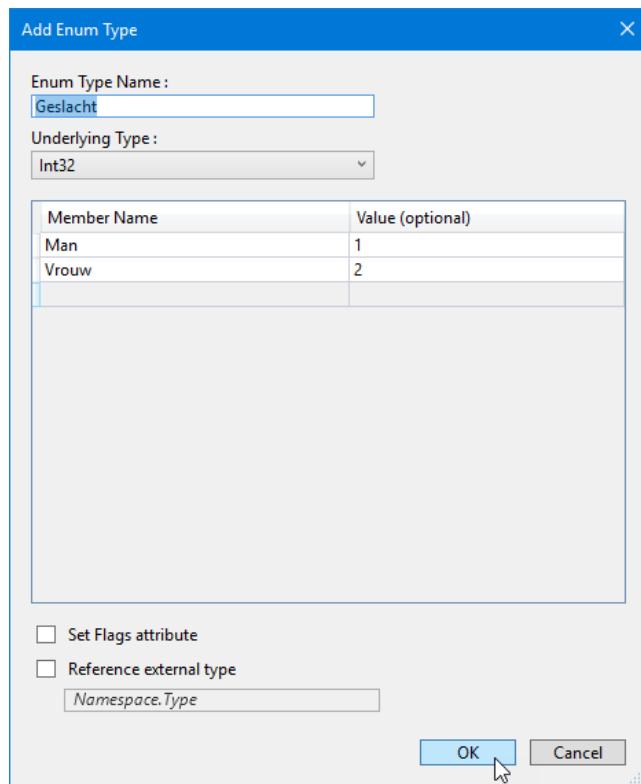
- Visual Studio heeft aan **Docent** een property **Geslacht** van het type **int32** toegevoegd.



- Je klikt met de rechtermuisknop op deze property en je kiest **Convert to Enum**.



- Je tikt bij **Enum Type Name** de naam van de enum die je wenst aan te maken: **Geslacht**
- Je tikt onder **Member Name** de eerste mogelijke waarde in deze enum: **Man**
Je tikt daarnaast (onder **Value**) het getal dat **Man** voorstelt in de database: **1**
- Je tikt op een nieuwe regel onder **Member Name** de tweede mogelijke waarde in deze enum: **Vrouw**. Je tikt daarnaast (onder **Value**) het getal dat **Man** voorstelt in de database: **2**



Opmerking:

Als je **Value** leeg laat, associeert EF de eerste enum waarde (**Man**) met de waarde **0** in de database, de tweede enum waarde (**Vrouw**) met de waarde **1**, ...

Als EF records leest uit de tabel **docenten**, zet hij de getallen in de kolom **Geslacht** om naar de bijbehorende enum-waarden in de property **Geslacht** van **Docent**-entities.

Je ziet dit met volgend voorbeeld in de method **Main**:

```
using (var entities = new EFopleidingenEntities())
{
    foreach (var docent in entities.Docenten)
    {
        Console.WriteLine("{0}:{1}", docent.Naam, docent.Geslacht);
    }
}
```

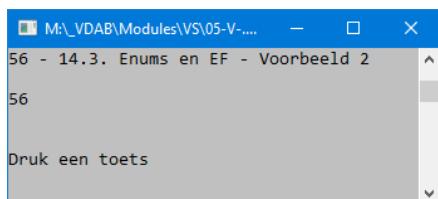


Als EF een **Docent**-entity toevoegt als een nieuw record in de table **docenten**, zet hij de enum-waarde in de property **Geslacht** om naar het bijbehorend getal in de kolom **Geslacht**.

Je ziet dit met volgend voorbeeld in in de method **Main**:

```
using (var entities = new EFopleidingenEntities())
{
    entities.Docenten.Add
    (
        new Docent
        {
            Naam = new Naam { Voornaam = "Brigitta", Familienaam = "Roos" },
            Wedde = 2000,
            Geslacht = Geslacht.Vrouw,
            CampusNr = 1
        }
    );
    entities.SaveChanges();
}
```

317	Grace	Verbeke	2200,00	1	2
NULL	NULL	NULL	NULL	NULL	NULL



DocentNr	Voornaam	Familienaam	Wedde	CampusNr	Geslacht
317	Grace	Verbeke	2200,00	1	2
318	Brigitta	Roos	2000,00	1	2

15 VIEWS

15.1 ALGEMEEN

Een database kan naast tables ook views en stored procedures bevatten.

Je ziet in dit hoofdstuk hoe je views aanspreekt met EF.

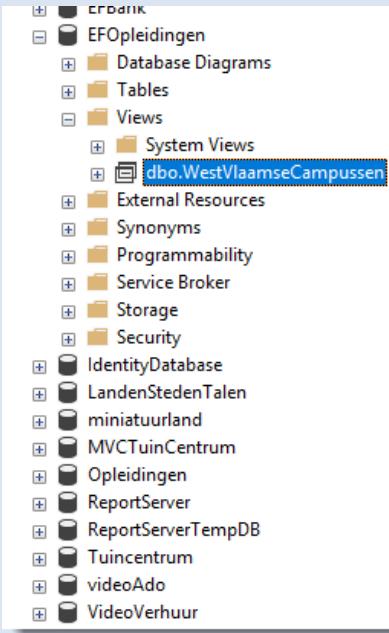
Een view is een virtuele table die data voorstelt uit één of meerdere echte tables.

15.2 EEN VIEW AANMAKEN

Je maakt met het volgend SQL-statement bijvoorbeeld een view met de naam **WestVlaamseCampussen** die de campussen voorstelt uit West-Vlaanderen:

```
use EFOpleidingen
go

create view WestVlaamseCampussen as
select * from Campussen
where PostCode between '8000' and '8999'
```



Er zijn beperkingen bij het aanmaken van een view, waaronder :

- Je kan **order by** niet gebruiken
- Je kan geen parameters definiëren

15.3 DE DATA VAN EEN VIEW LEZEN VANUIT SQL

Je leest met het volgend SQL statement de data van de view:

```
select * from WestVlaamseCampussen
```

	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente
1	5	Ikaria	Vlamingstraat	10	8560	Wevelgem
2	6	Oinouses	Archimedesstraat	4	8400	Oostende

15.4 DE VOORBEELDVIEW

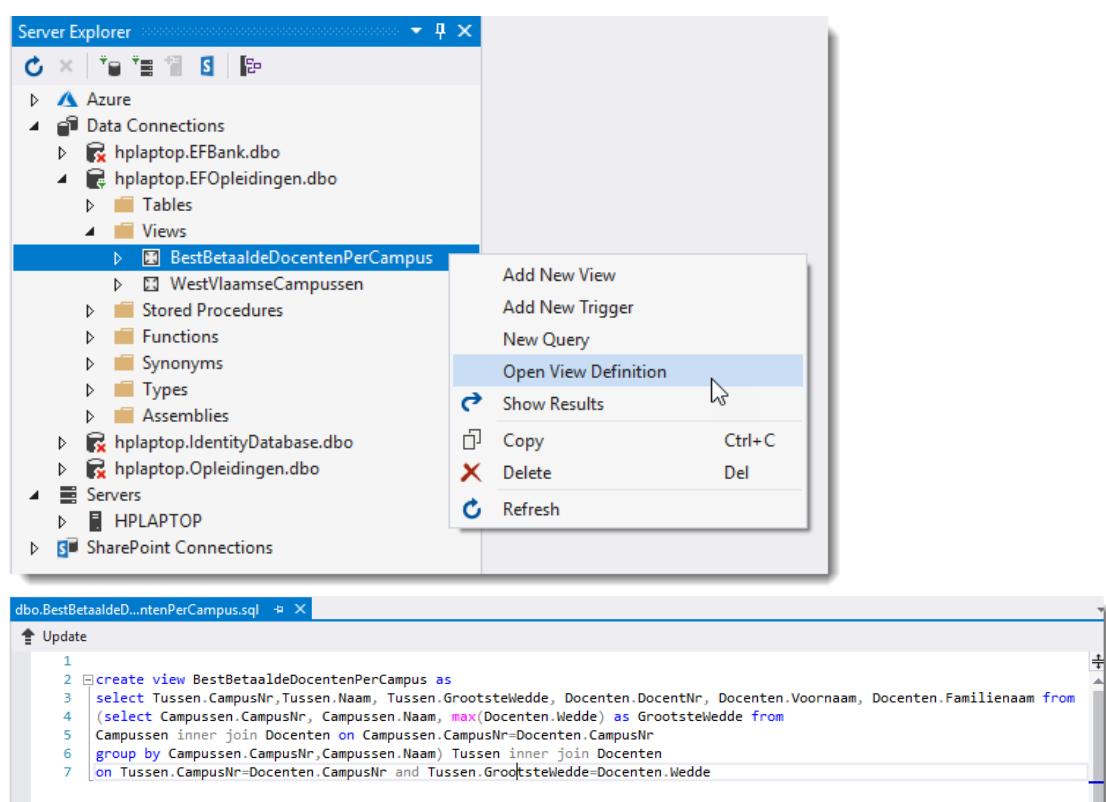
Het script **BestBetaaldeDocentenPerCampusMaken.sql** voegt aan de database een view toe met de naam **BestBetaaldeDocentenPerCampus**.

Deze view leest per campus de docent(en) met de hoogste wedde.

Je voert dit script uit in de **SQL Server Management Studio**.

Je kan de definitie van de view zien in **Visual Studio** op volgende manier:

- Je klap in de **Server Explorer** de databaseconnectie **EFOpleidingen** open.
- Je klap daarbinnen het onderdeel **Views** open.
- Je selecteert de view **BestBetaaldeDocentenPerCampus** met de rechtermuisknop en je kiest **Open View Definition**.



Je kan het resultaat van de view zien in **Visual Studio** op volgende manier:

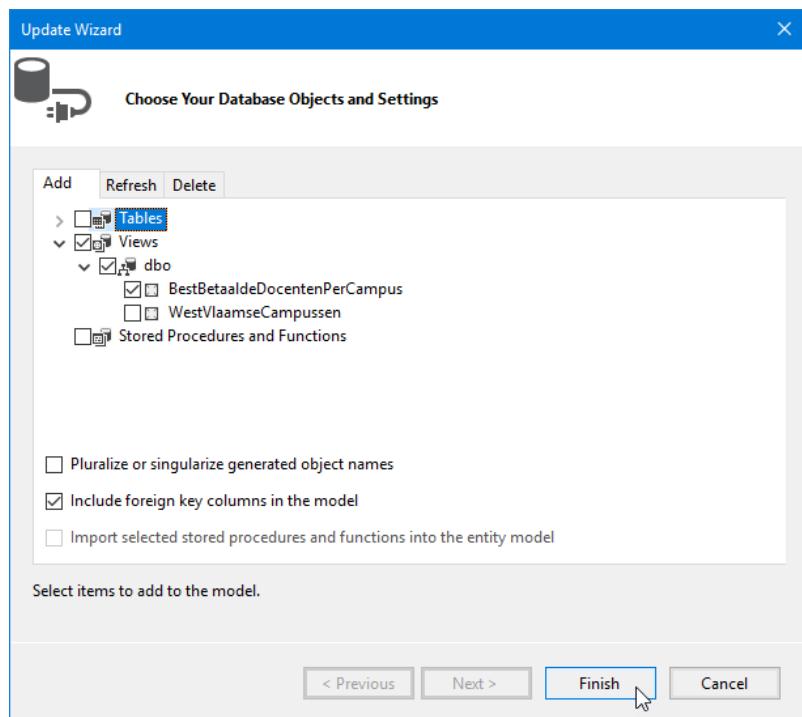
- Je selecteert in de **Server Explorer** de view **BestBetaaldeDocentenPerCampus** met de rechtermuisknop en je kiest **Show Results**. Je ziet dat het resultaat niet gesorteerd is. Je lost dit straks op door te sorteren in een **LINQ**-query.

	CampusNr	Naam	GrootsteWedde	DocentNr	Voornaam	Familienaam
►	2	Delos	3810,00	2	Joseph	Abelhausen
	6	Oinouses	1600,00	6	René	Adriaensens
	6	Oinouses	1600,00	10	Mario	Aerts
	5	Ikaria	2000,00	11	Paul	Aerts
	3	Gavdos	1900,00	13	François	Alexander
	6	Oinouses	1600,00	14	Henri	Allard
	3	Gavdos	1900,00	17	Etienne	Antheunis
	4	Hydra	1400,00	18	Jacques	Arlet
	4	Hydra	1400,00	22	Hubert	Baert
	6	Oinouses	1600,00	27	Serge	Baguet
	6	Oinouses	1600,00	31	Auguste	Baumans
	5	Ikaria	2000,00	22	Arène	Bauwens

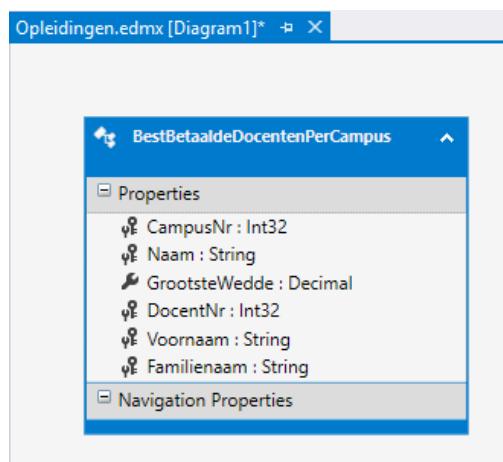
15.5 DE ENTITIES DEFINIËREN DIE HOREN BIJ DE VIEW

Je voegt aan **Opleidingen.edmx** een entity toe die hoort bij de view die je maakte:

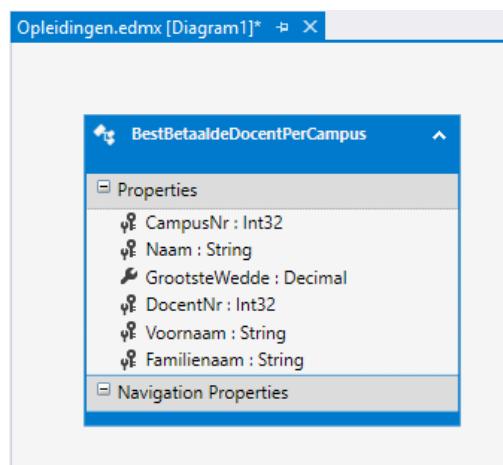
- Je klikt met de rechtermuisknop in **Opleidingen.edmx**.
- Je kiest **Update Model from Database**.
- Je klapt in het tabblad **Add** het onderdeel **Views** open.
- Je klapt daarbinnen het onderdeel **dbo** open.
- Je plaatst een vinkje bij **BestBetaaldeDocentenPerCampus**
- Je kiest **Finish** en je negeert de warning.



- Je ziet de entity **BestBetaaldeDocentenPerCampus**



- Je wijzigt de naam van de entity naar **BestBetaaldeDocentPerCampus**.



15.6 DE ENTITIES AANSPREKEN VANUIT CODE

```
using (var entities = new EFopleidingenEntities())
{
    var query =      from      bestBetaaldeDocentPerCampus
                     in       entities.BestBetaaldeDocentenPerCampus
                     orderby  bestBetaaldeDocentPerCampus.CampusNr,
                               bestBetaaldeDocentPerCampus.Voornaam,
                               bestBetaaldeDocentPerCampus.Familienaam
                     select  bestBetaaldeDocentPerCampus;

    var vorigCampusNr = 0;

    foreach (var bestbetaaldeDocentPerCampus in query)
    {
        if (bestbetaaldeDocentPerCampus.CampusNr != vorigCampusNr)
        {
            Console.WriteLine("{0} {1} Grootste wedde:",
                bestbetaaldeDocentPerCampus.Naam, bestbetaaldeDocentPerCampus.GrootsteWedde);
            vorigCampusNr = bestbetaaldeDocentPerCampus.CampusNr;
        }

        Console.WriteLine("\t{0} {1}",
            bestbetaaldeDocentPerCampus.Voornaam, bestbetaaldeDocentPerCampus.Familienaam);
    }
}
```

```
15.       Views
57 - 15.6. De entites aanspreken vanuit code
57

Andros 2200,00 Grootste wedde:
Grace Verbeke
Delos 3810,00 Grootste wedde:
Joseph Abelshausen
Gavdos 1900,00 Grootste wedde:
Albert Ritserveldt
André Messelis
Arthur Decabooter
Cesar Bogaert
Charles Deruyter
Dries Govaerts
Emile Bruneau
Ernest Sterckx
Etienne Antheunis
François Alexander
Gaston Rebray
Guy Nulens
Henri Garnier
Johan Bruyneel
Jules Bayens
Kamiel Beeckman
Léon Despontin
Marc Wauters
Martin Van Geneugden
Maurice Blomme
Nico Mattan
Norbert Kerckhove
Odiel Defraeye
Odiel Vanden Meerschaut
Patrick Sercu
Petrus Oellibrandt
Rik Verbrugge
Roger Lambrecht
Steve Vermaut
Hydra 1400,00 Grootste wedde:
Adelin Benoit
Adolf Verschueren
Albert De Jonghe
Alfons De Wolf
André Lurquin
Andrei Tchmil
Bert Roesems
```

15.7 TAAK 10 : TOTALE SALDO PER KLANT

Het script `TotaleSaldoPerKlantMaken.sql` voegt aan de database `EFBank` een view `TotaleSaldoPerKlant` toe. Je voert dit script uit.

De view bevat 3 kolommen: `KlantNr`, `Voornaam`, `TotaleSaldo` (het totale saldo van de rekeningen van één klant).

Voeg aan het EDM een entity toe die bij deze view hoort.

Toon daarna per klant de voornaam en het totale saldo van die klant.

16 STORED PROCEDURES

16.1 ALGEMEEN

Een stored procedure is een procedure die in de database is opgeslagen.

- Een stored procedure kan één of meerdere SQL-statements bevatten.
Dit kunnen select-, insert-, update- en/of delete-statements zijn.
- Een stored procedure kan ook parameters bevatten.

16.2 EEN STORED PROCEDURE AANMAKEN

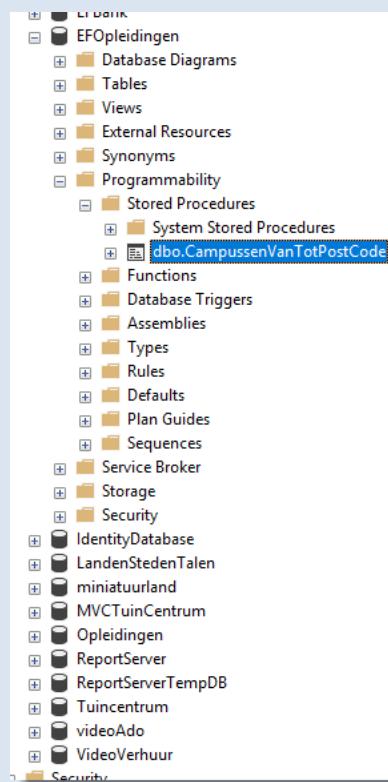
Je maakt met het volgend SQL-statement een stored procedure.

Deze stored procedure heeft twee parameters **VanPostCode** en **TotPostCode**.

De stored procedure leest de **campussen** met een **postcode** die valt tussen deze parameters:

```
use EFOpleidingen
go

create procedure CampussenVanTotPostCode(@VanPostCode nvarchar(10),
                                         @TotPostCode nvarchar(10))
as
select * from Campussen
where Postcode between @VanPostCode and @TotPostCode
order by PostCode, Naam
```



16.3 EEN STORED PROCEDURE OPROEPEN VANUIT SQL

Je voert met het volgend statement de stored procedure uit:

<pre>execute campussenvantotpostcode '9000', '9999'</pre>														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th> <th>CampusNr</th> <th>Naam</th> <th>Straat</th> <th>HuisNr</th> <th>PostCode</th> <th>Gemeente</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2</td> <td>Delos</td> <td>Oude Vest</td> <td>17</td> <td>9200</td> <td>Dendermonde</td> </tr> </tbody> </table>		CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente	1	2	Delos	Oude Vest	17	9200	Dendermonde
	CampusNr	Naam	Straat	HuisNr	PostCode	Gemeente								
1	2	Delos	Oude Vest	17	9200	Dendermonde								

Het script [CampussenVanTotPostCodeMaken.sql](#) voegt aan de database de stored procedure [CampussenVanTotPostCode](#) toe. Je voert dit script uit in de [SQL Server Management Studio](#).

16.4 STORED PROCEDURES OPROEPEN MET EF

Je kan een stored procedure oproepen vanuit EF als:

- De stored procedure data terug geeft in de vorm van entities, of
- De stored procedure data terug geeft in de vorm verschillend van entities, of
- De stored procedure geen data terug geeft, of
- De stored procedure een scalar value terug geeft

16.4.1 EEN STORED PROCEDURE DIE DATA TERUG GEEFT IN DE VORM VAN ENTITIES

Sommige stored procedures lezen data en geven die aan de applicatie die de procedure oproept.

Deze data kunnen dezelfde kolommen bevatten als één van de tables van de database. Dan kunnen deze data ook voorgesteld worden als een verzameling entities.

De stored procedure [CampussenVanTotPostCode](#) leest bijvoorbeeld alle kolommen uit de table [Campussen](#). Als je deze stored procedure oproept vanuit je applicatie, kan het resultaat van deze stored procedure voorgesteld worden als een verzameling [Campus](#)-entities.

Je ziet hier hoe je zo'n stored procedure oproept met EF.

Je voegt de stored procedure toe aan [Opleidingen.edmx](#):

- Je klikt met de rechtermuisknop in de achtergrond van [Opleidingen.edmx](#).
- Je kiest [Update Model from Database](#).
- Je klapt in het tabblad [Add](#) het onderdeel [Stored Procedures](#) open.
- Je klapt daarbinnen het onderdeel [dbo](#) open.
- Je plaatst een vinkje bij [CampussenVanTotPostCode](#) en je kiest [Finish](#).

Je ziet geen nieuw onderdeel in de designer, maar [Visual Studio](#) heeft aan de object context class ([OpleidingenEntities](#)) een method toegevoegd die zelf de stored procedure oproept. Jij zal vanuit je code de stored procedure oproepen door deze nieuwe method op te roepen.

```
/*
// <auto-generated>
// This code was generated from a template.
//
// Manual changes to this file may cause unexpected behavior in your application.
// Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//

namespace EFCursus
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;
    using System.Data.Entity.Core.Objects;
    using System.Linq;
```

```
public partial class EFOPleidingenEntities : DbContext
{
    public EFOPleidingenEntities()
        : base("name=EFOPleidingenEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }

    public virtual DbSet<Campus> Campussen { get; set; }
    public virtual DbSet<Docent> Docenten { get; set; }
    public virtual DbSet<Voorraad> Voorraden { get; set; }
    public virtual DbSet<Boek> Boeken { get; set; }
    public virtual DbSet<Cursus> Cursussen { get; set; }
    public virtual DbSet<Boek2> Boeken2 { get; set; }
    public virtual DbSet<BoekCursus2> BoekenCursussen2 { get; set; }
    public virtual DbSet<Cursus2> Cursussen2 { get; set; }
    public virtual DbSet<Cursist> Cursisten { get; set; }
    public virtual DbSet<BestBetaaldeDocentPerCampus> BestBetaaldeDocentenPerCampus { get; set; }

    public virtual ObjectResult<CampussenVanTotPostCode_Result> CampussenVanTotPostCode(string
vanPostCode, string totPostCode)
    {
        var vanPostCodeParameter = vanPostCode != null ?
            new ObjectParameter("VanPostCode", vanPostCode) :
            new ObjectParameter("VanPostCode", typeof(string));

        var totPostCodeParameter = totPostCode != null ?
            new ObjectParameter("TotPostCode", totPostCode) :
            new ObjectParameter("TotPostCode", typeof(string));

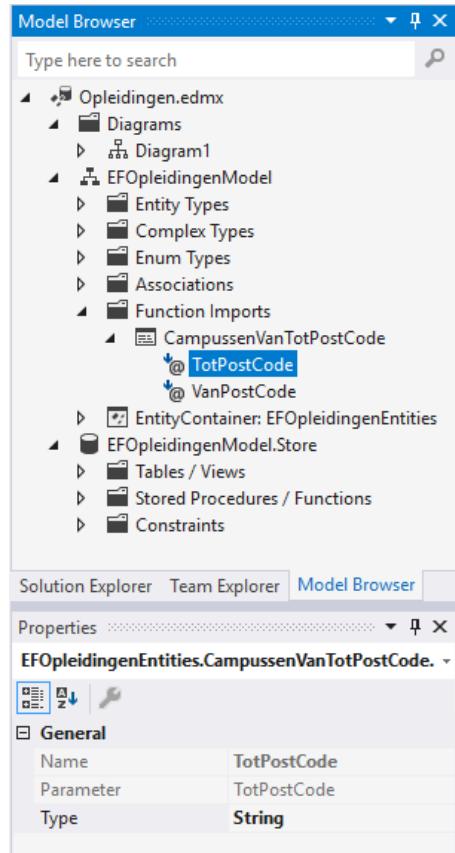
        return
((IObjectContextAdapter)this).ObjectContext.ExecuteFunction<CampussenVanTotPostCode_Result>("CampussenVanTotPostCode",
    vanPostCodeParameter, totPostCodeParameter);
    }
}
```

De method-naam is gelijk aan de bijbehorende stored-procedure-naam:

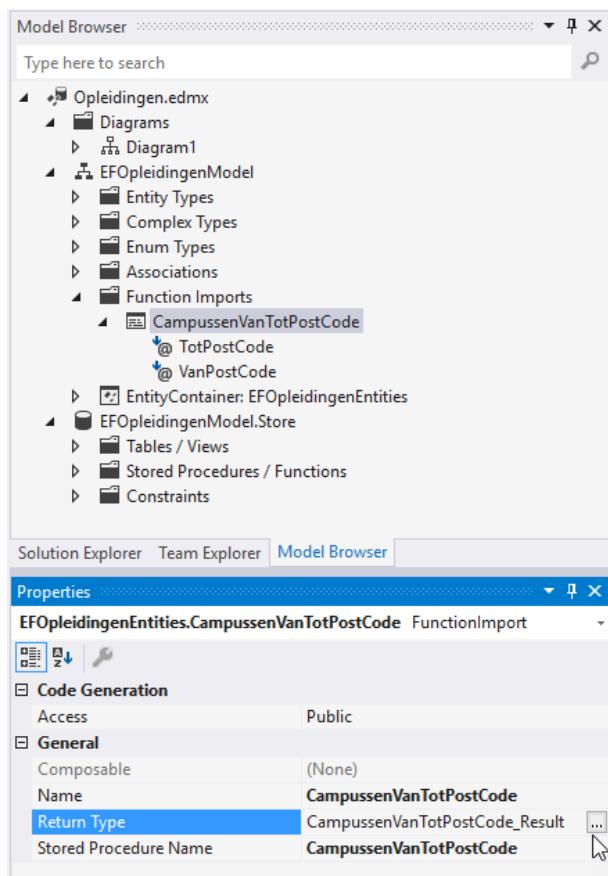
- **CampussenVanTotPostCode**

Je ziet dit op volgende manier:

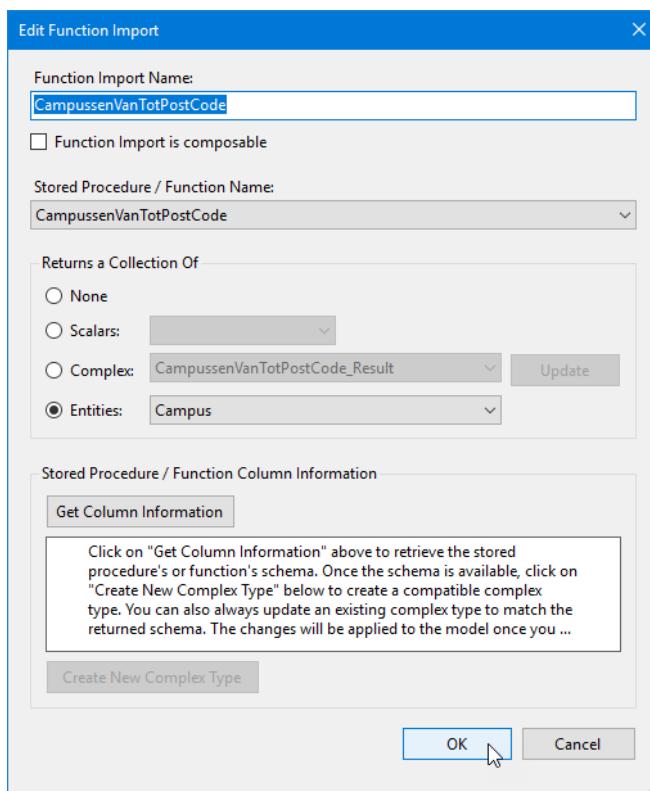
- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Model Browser**.
- Je klapt **EFOPleidingenModel** open.
- Je klapt daarbinnen **Function Imports** open
- Je ziet de function import **CampussenVanTotPostCode**
- Je klapt **CampussenVanTotPostCode** open.
- Je ziet de twee parameters van de method: **@VanPostCode** en **@TotPostCode**.
- Als je één van deze parameters aanklikt, zie je in het properties-venster bij **Type** dat deze parameters in de method het type **String** hebben.



- Je geeft aan dat de function **Campus** Entities teruggeeft:
 - Je selecteert **CampussenVanTotPostCode** in de **Model Browser**
 - Je klikt in het properties-venster op de knop **...** bij **Return Type**

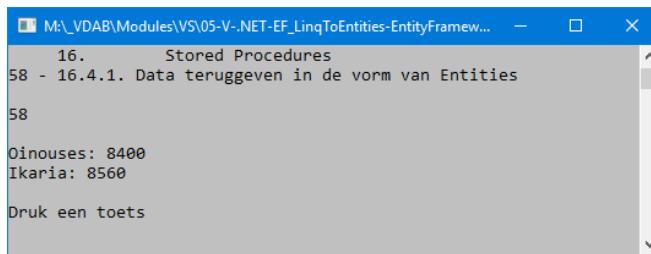


- Je kiest bij **Returns a Collection of** voor **Entities**
- Je kiest bij **Entities** voor **Campus**
- Je kiest **OK**



Je kan in de `Main` van `Program.cs` de method `CampussenVanTotPostCode` oproepen, die op zijn beurt de stored procedure oproept. De method geeft je een verzameling `Campus` entities, waarover je bijvoorbeeld kan itereren met foreach:

```
using (var entities = new EFopleidingenEntities())
{
    foreach (var campus in entities.CampussenVanTotPostCode("8000", "8999"))
    {
        Console.WriteLine("{0}: {1}", campus.Naam, campus.PostCode);
    }
}
```



16.4.2 EEN STORED PROCEDURE DIE DATA TERUG GEEFT IN EEN VORM DIE NIET OVEREENSTEMT MET DE STRUCTUUR VAN EEN ENTITY

Sommige stored procedures bieden data aan, waarvan de kolomstructuur niet overeenstemt met de property-structuur van één van je entity classes.

Het script `AantalDocentenPerVoornaamMaken.sql` voegt aan de database `EFopleidingen` de stored procedure `AantalDocentenPerVoornaam` toe.

Je voert dit script uit in de `SQL Server Management Studio`.

Deze stored procedure geeft een lijst met per voornaam de voornaam en het aantal docenten die deze voornaam heeft:

	Voornaam	Aantal
1	Adelin	1
2	Adolf	1
3	Adolphe	1
4	Albert	9
5	Alex	1
6	Alfons	4
7	Alfred	2
8	André	6
9	Andrei	1
10	Andy	1
11	Armand	2
12	Arsène	1
13	Arthur	3
14	Auguste	3
15	Aviel	1
	Return Value	
1		0

De structuur van één rij uit deze lijst komt niet overeen met de structuur van één van je entities.

Als je deze stored procedure integreert in **EDMX**, maakt **Visual Studio** een class die één rij uit deze lijst weerspiegelt: een class met een **string**-property **Voornaam** en een **int**-property **Aantal**.

Je voegt de stored procedure toe aan **Opleidingen.edmx**:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Update Model from Database**.
- Je klapt in het tabblad **Add** het onderdeel **Stored Procedures** open.
- Je klapt daarbinne het onderdeel **dbo** open.
- Je plaatst een vinkje bij **AantalDocentenPerVoornaam**
- Je kiest **Finish**.

Visual Studio heeft aan de object context class (**OpleidingenEntities**) een method toegevoegd die zelf de stored procedure oproept. Jij zal vanuit je code de stored procedure oproepen door deze nieuwe method op te roepen.

De method-naam is dezelfde als de bijbehorende stored-procedure-naam:

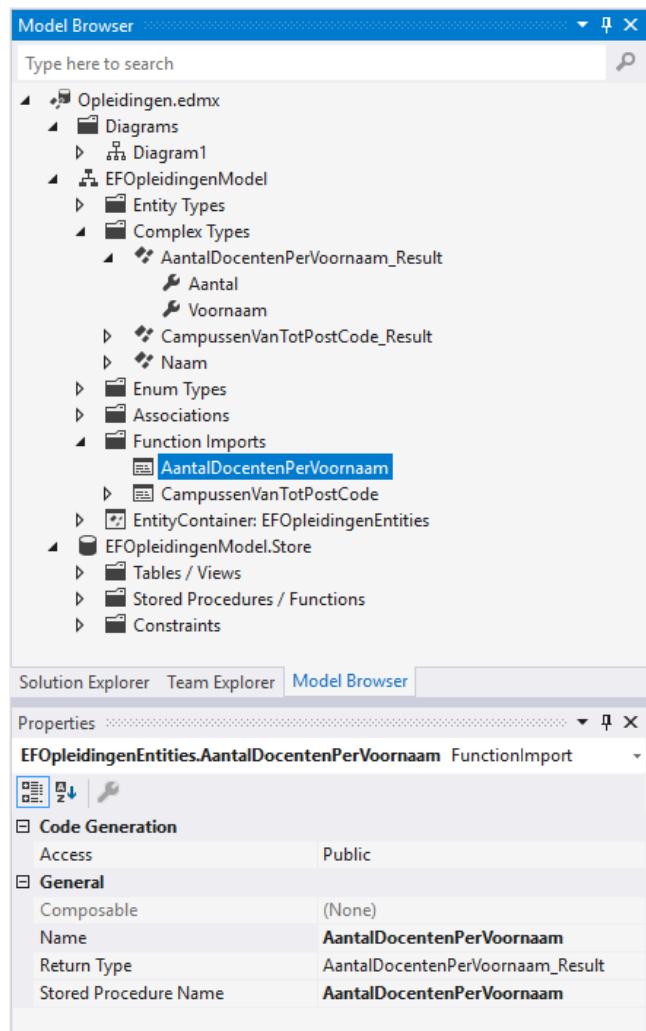
- **AantalDocentenPerVoornaam**

Je ziet dit op volgende manier:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Model Browser**.
- Je klapt **EFOpleidingenModel** open.
- Je klapt daarbinnen **Function Imports** open
- Je selecteert de function import **AantalDocentenPerVoornaam**
- Je ziet in het Properties-venster bij **Return Type**
AantalDocentenPerVoornaam_Result

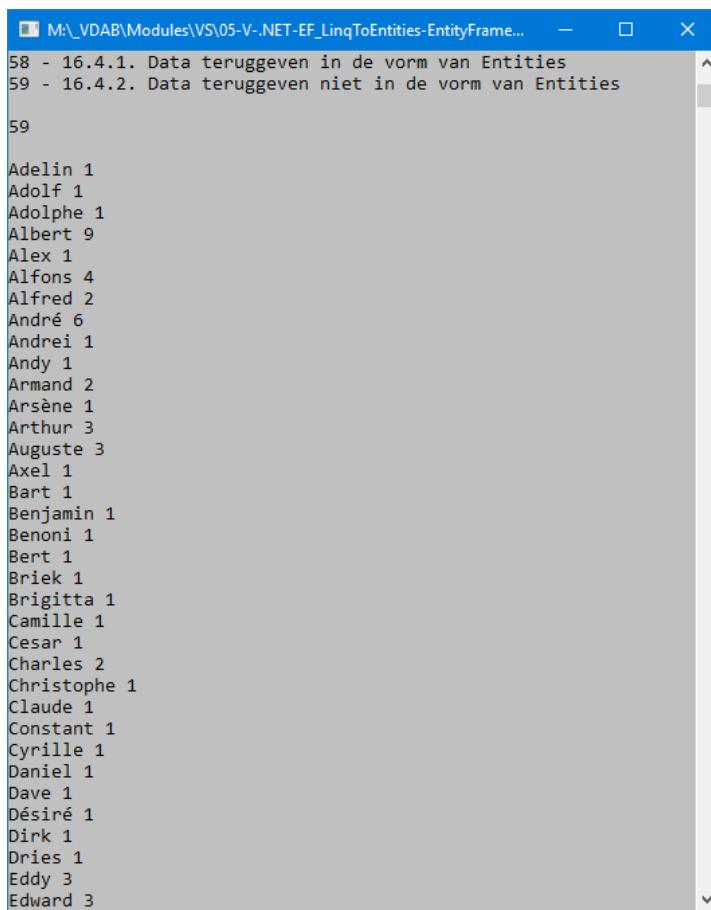
Dit is de class die **Visual Studio** heeft aangemaakt als een weerspiegeling van één rij uit de resultaatdata van de stored procedure.

- Je ziet de structuur van deze class ook in de **Model Browser**, in het onderdeel **Complex Types** van **OpleidingenModel**



Je kan in de [Main](#) van [Program.cs](#) de method [AantalDocentenPerVoornaam](#) oproepen, die op zijn beurt de stored procedure oproept. De method geeft je een verzameling objecten van het type [AantalDocentenPerVoornaam_Result](#) terug:

```
using (var entities = new EFModel1())
{
    foreach (var voornaamAantal in entities.AantalDocentenPerVoornaam())
    {
        Console.WriteLine("{0} {1}", voornaamAantal.Voornaam, voornaamAantal.Aantal);
    }
}
```



The screenshot shows a Windows command prompt window with the title bar 'M:_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFrame...'. The window displays a list of names and their counts, likely from a database query. The data is as follows:

Name	Count
Adelin	1
Adolf	1
Adolphe	1
Albert	9
Alex	1
Alfons	4
Alfred	2
André	6
Andrei	1
Andy	1
Armand	2
Arsène	1
Arthur	3
Auguste	3
Axel	1
Bart	1
Benjamin	1
Benoni	1
Bert	1
Briek	1
Brigitta	1
Camille	1
Cesar	1
Charles	2
Christophe	1
Claude	1
Constant	1
Cyrille	1
Daniel	1
Dave	1
Désiré	1
Dirk	1
Dries	1
Eddy	3
Edward	3

16.4.3 EEN STORED PROCEDURE DIE GEEN DATA TERUG GEEFT

Sommige stored procedures voegen records toe, wijzigen records of verwijderen records, maar geven daarna geen data terug aan de applicatie die de stored procedure heeft opgeroepen.

De volgende stored procedure geeft een weddeverhoging aan alle docenten:

```
create procedure WeddeVerhoging(@Percentage decimal(5,2))
as
update Docenten
set wedde = wedde * (1 + @Percentage / 100)
```

Het script `WeddeVerhogingMaken.sql` voegt aan de database `EFOpleidingen` de stored procedure `WeddeVerhoging` toe.

Je voert dit script uit in de `SQL Server Management Studio`.

Je ziet nu hoe je zo'n stored procedure oproeft met EF.

Je voegt de stored procedure toe aan `Opleidingen.edmx`:

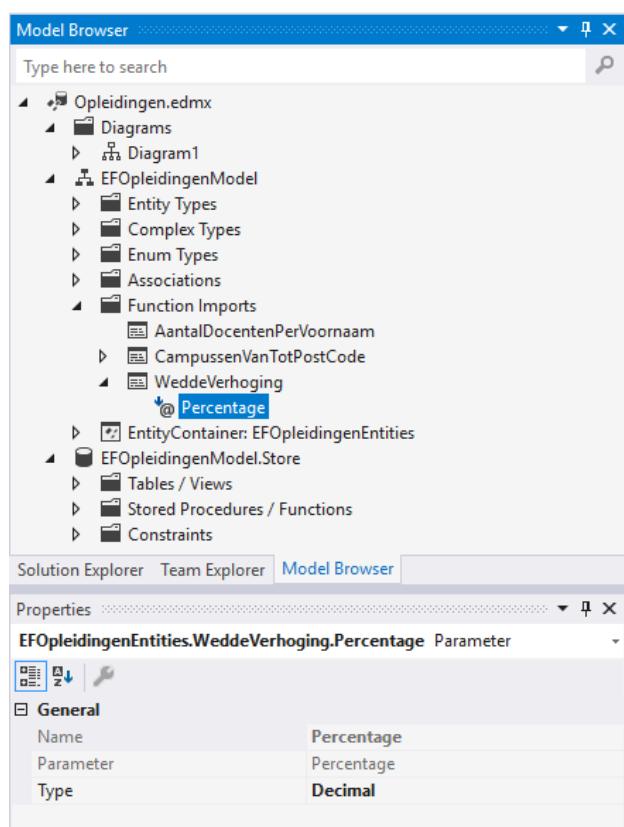
- Je klikt met de rechtermuisknop in de achtergrond van `Opleidingen.edmx`.
- Je kiest `Update Model from Database`.
- Je klapt in het tabblad `Add` het onderdeel `Stored Procedures` open.
- Je klapt daarbinnen het onderdeel `dbo` open.
- Je plaatst een vinkje bij `WeddeVerhoging`
- Je kiest `Finish`.

Visual Studio heeft aan de object context class (`OpleidingenEntities`) een method toegevoegd die zelf de stored procedure oproept. Jij zal vanuit je code de stored procedure oproepen door deze nieuwe method op te roepen.

De method naam is gelijk aan de achterliggende stored procedure naam: `WeddeVerhoging`

Je ziet dit op volgende manier:

- Je klikt met de rechtermuisknop in de achtergrond van `Opleidingen.edmx`.
- Je kiest **Model Browser**.
- Je klapte `EFOpleidingenModel` open.
- Je klapte daarbinnen `Function Imports` open
- Je ziet de function import `WeddeVerhoging`
- Je klapte `WeddeVerhoging` open.
- Je ziet de parameter van de method: `@Percentage`.
- Als je deze parameter aanklikt, zie je in het properties venster bij **Type** dat deze parameter in de method het type `Decimal` heeft.



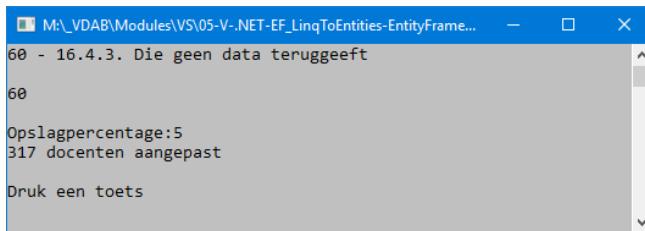
Je kan in de `Main` van `Program.cs` de method `WeddeVerhoging` oproepen, die op zijn beurt de stored procedure oproept. De method geeft je het aantal aangepaste records terug:

```
Console.WriteLine("Opslagpercentage:");

decimal percentage;

if (decimal.TryParse(Console.ReadLine(), out percentage))
{
    using (var entities = new EFOpleidingenEntities())
    {
        var aantalDocentenAangepast = entities.WeddeVerhoging(percentage);
        Console.WriteLine("{0} docenten aangepast", aantalDocentenAangepast);
    }
}
```

```
}
else
{
    Console.WriteLine("Tik een getal");
}
```



16.4.4 EEN STORED PROCEDURE DIE DATA LEEST ALS EEN SCALAR VALUE

Een scalar value is één enkele waarde: data met één rij én één kolom.

De volgende stored procedure geeft een scalar value terug:

```
create procedure AantalDocentenMetFamilienaam(@Familienaam nvarchar(50))
as
select count(*)
from Docenten
where Docenten.Familienaam = @Familienaam
```

Het script [AantalDocentenMetFamilieNaamMaken.sql](#) maakte in de database **EFopleidingen** de stored procedure **AantalDocentenMetFamilieNaam**.

Je voert dit script uit in de **SQL Server Management Studio**.

Je ziet nu hoe je zo'n stored procedure oproept met EF.

Je voegt de stored procedure toe aan **Opleidingen.edmx**:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Update Model from Database**.
- Je klapt in het tabblad **Add** het onderdeel **Stored Procedures** open.
- Je klapt daarbinnen het onderdeel **dbo** open.
- Je plaatst een vinkje bij **AantalDocentenMetFamilieNaam**

Visual Studio heeft aan de object context class (**OpleidingenEntities**) een method toegevoegd die zelf de stored procedure oproept. Jij zal vanuit je code de stored procedure oproepen door deze nieuwe method op te roepen.

De method naam is gelijk aan de achterliggende stored procedure naam:

- **AantalDocentenMetFamilieNaam**

Je ziet dit op volgende manier:

- Je klikt met de rechtermuisknop in de achtergrond van **Opleidingen.edmx**.
- Je kiest **Model Browser**.
- Je klapt **EFopleidingenModel** open.
- Je klapt daarbinnen **Function Imports** open
- Je ziet de function import **AantalDocentenMetFamilieNaam**
- Je klapt **AantalDocentenMetFamilieNaam** open.

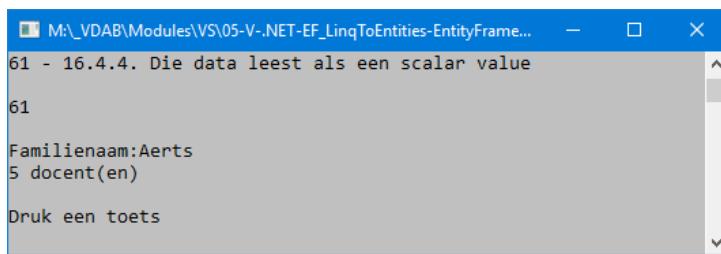
- Je ziet de parameter van de method: `@Familienaam`.
- Als je deze parameter aanklikt, zie je in het properties-venster bij **Type** dat deze parameter in de method het type **String** heeft.

Je kan in de method **Main** de method **AantalDocentenMetFamilienaam** oproepen, die op zijn beurt de stored procedure oproept. De method geeft je een object van het type **ObjectResult** terug. Je voert op dit object de First-method uit om de scalar-returnwaarde van de stored procedure te lezen.

```
Console.WriteLine("Familienaam:");
var familienaam = Console.ReadLine();

using (var entities = new EFopleidingenEntities())
{
    var aantalDocenten = entities.AantalDocentenMetFamilienaam(familienaam);
    Console.WriteLine("{0} docent(en)", aantalDocenten.First());
}
```

	DocentNr	Voornaam	Familienaam	Wedde	CampusNr	Geslacht
1	1	Willy	Abbeloos	1470.00	3	1
2	2	Joseph	Abelshausen	4000.50	2	1
3	3	Joseph	Achten	1365.00	3	1
4	4	François	Adam	1795.50	1	1
5	6	René	Adriaensens	1680.00	6	1
6	7	Frans	Aerenhouts	1365.00	3	1
7	8	Emile	Aerts	1795.50	1	1
8	9	Jean	Aerts	1260.00	2	1
9	10	Mario	Aerts	1680.00	6	1
10	11	Paul	Aerts	2100.00	5	1
11	12	Stefan	Aerts	1575.00	5	1
12	13	François	Alexander	1995.00	3	1
13	14	Wim	Van den Berghe	1000.00	6	1



```
M:\_VDAB\Modules\VS\05-V-.NET-EF_LinqToEntities-EntityFrame...
61 - 16.4.4. Die data leest als een scalar value
61
Familienaam:Aerts
5 docent(en)

Druk een toets
```

16.5 TAAK 11 : ADMINISTRATIEVE KOST

Het script **AdministratieveKostMaken.sql** maakt in de database **Bank** een stored procedure **AdministratieveKost**. Je voert dit script uit.

De stored procedure bevat één invoerparameter: **Kost (decimal)**.

De stored procedure vermindert het saldo van alle rekeningen met het bedrag in deze invoerparameter Kost.

Je vraagt aan de gebruiker het bedrag van de kost.

Als de gebruiker geen getal intikt, toon je de foutmelding Tik een getal.

Anders roep je de stored procedure op, waarbij je het ingetikte getal als parameter meegeeft.

Je toont aan de gebruiker het aantal aangepaste rekeningen.

17 CODE FIRST

17.1 ALGEMEEN

Je gebruikte tot nu een **EDMX-bestand** om het verband te leggen tussen tables uit de database en classes in je programma.

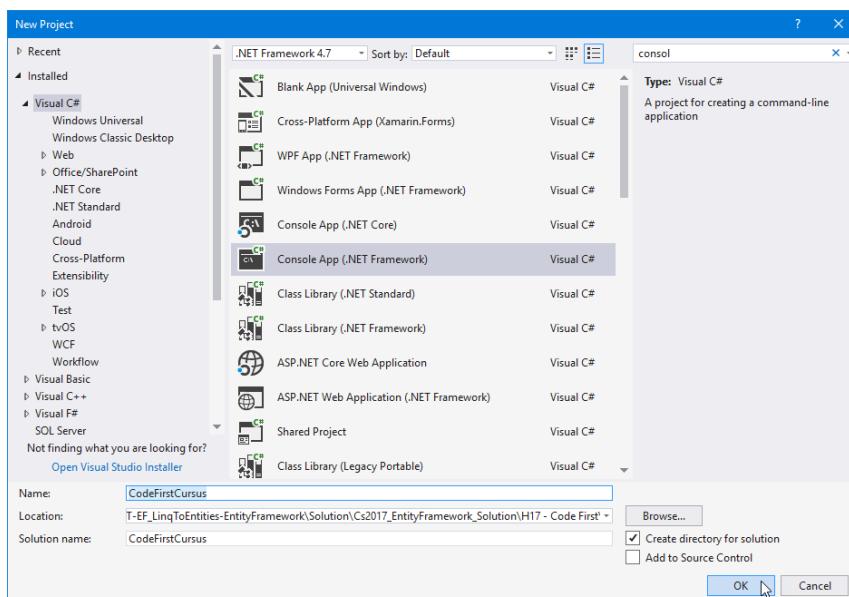
Je zal nu de verbanden uitdrukken in **C#-code**, onder andere met attributen (woorden tussen [en] die je voor een class of een variabele schrijft).

Ook de namen van classes en variabelen zijn belangrijk voor de configuratie. Dit heet "**Configuration by Convention**".

Bij Code First kan je een nieuwe database maken op basis van de C#-code of met een bestaande database werken.

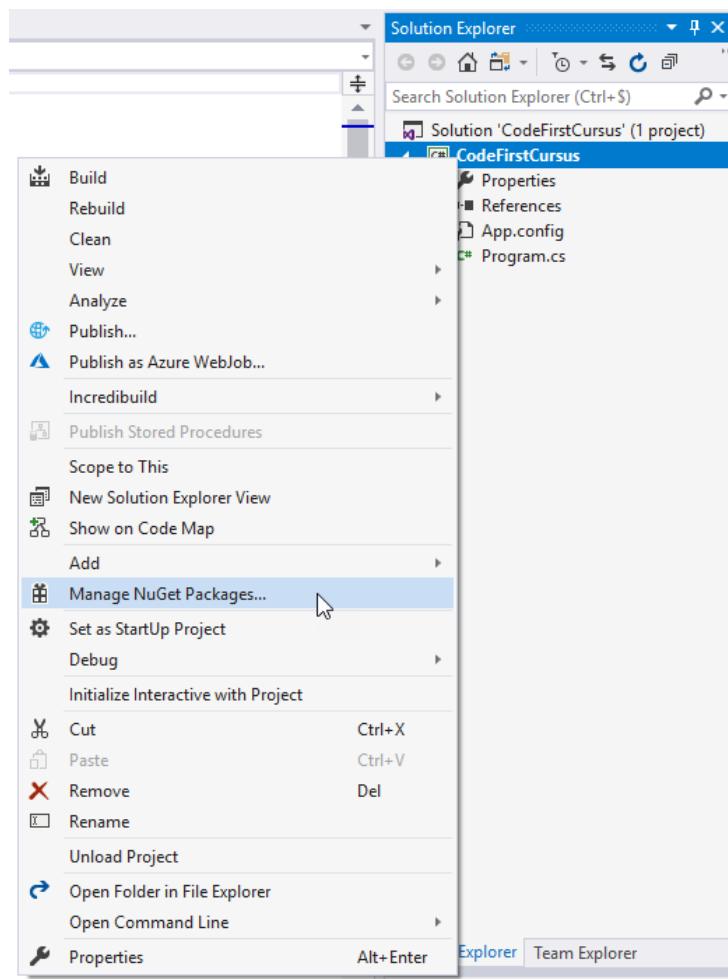
17.2 DE ENTITY CLASSES VOOR DE NIEUWE DATABASE

Je maakt in Visual Studio een **Console Application** project met de naam **CodeFirstCursus**.



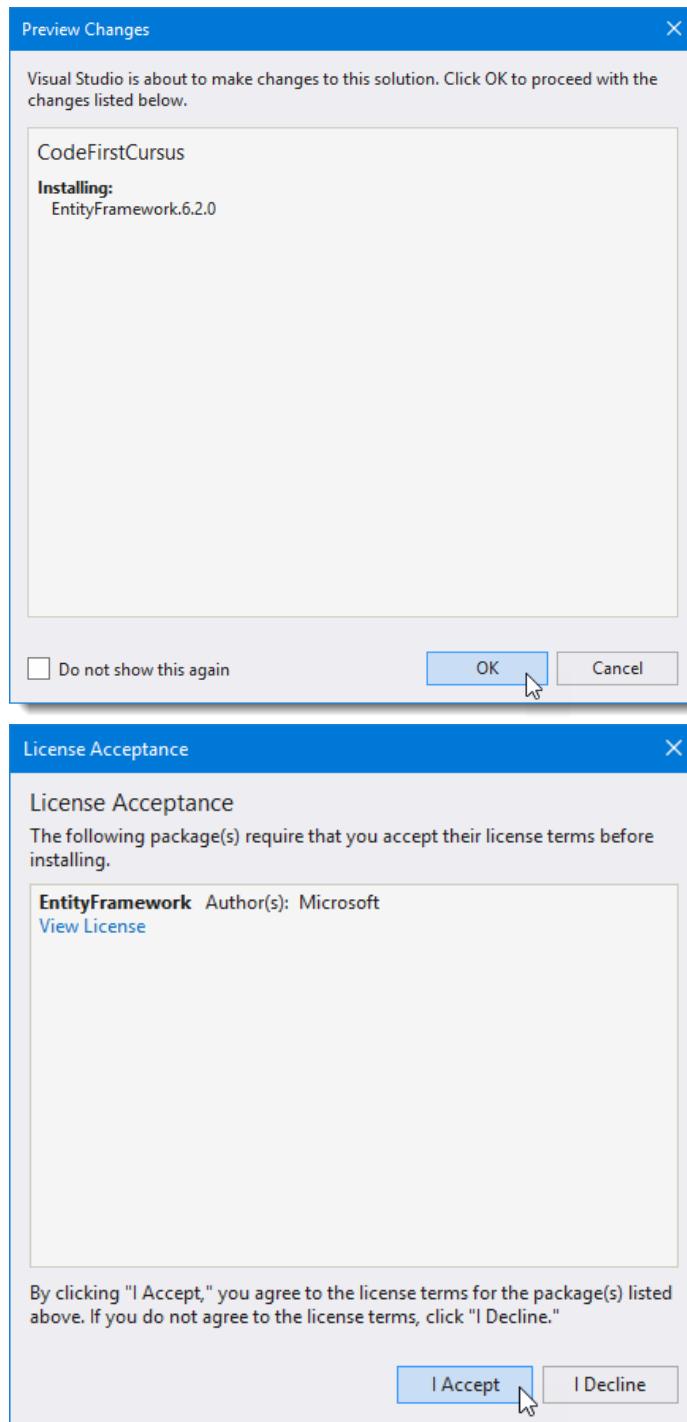
Je downloadt de laatste versie van de **EF library** en gebruikt die in het project

- Je klikt met de rechtermuisknop op het project in de **Solution Explorer**
- Je kiest **Manage NuGet Packages**



- Je kiest links **Browse**
- Je kiest links **EntityFramework**
- Je kiest daaronder **Install**

A screenshot of the NuGet Package Manager. On the left, the search results for 'EntityFramework' are shown, listing packages like Newtonsoft.Json, NUnit, EntityFramework, MySql.Data, bootstrap, and jQuery. On the right, the details page for the EntityFramework package is displayed, showing version 6.2.0, author Microsoft, and download statistics. The 'Install' button is highlighted with a blue selection bar.



Je voegt een entity class **Instructeur** toe:

```
using System;

namespace CodeFirstCursus
{
    public class Instructeur           // (1)
    {
        public int Id { get; set; }      // (2)
        public string Voornaam { get; set; } // (3)
        public string Familienaam { get; set; }
        public decimal Wedde { get; set; }
        public DateTime InDienst { get; set; }
```

```

        public void Opslag(decimal percentage)
    {
        Wedde *= (1M + percentage / 100M);
    }
}

```

- (1) CodeFirst aanziet de **naam van de class** + het teken **s** als de naam van de table (**Instructeurs**) die bij de **entity class** zal horen.
- (2) Als CodeFirst een property vindt met de naam **Id**, aanziet CodeFirst die als de property die hoort bij de **primary-key-kolom**. De kolomnaam is ook **Id**.
- (3) De naam van de kolom die hoort bij een property is dezelfde als de naam van de property.

Je voegt een entity class **Campus** toe:

```

namespace CodeFirstCursus
{
    public class Campus
    {
        public int CampusId { get; set; }          // (1)
        public string Naam { get; set; }           // (2)
    }
}

```

- (1) CodeFirst aanziet de naam van de class als de naam van de table (**Campus**) die bij de entity class hoort. CodeFirst voegt geen **s** toe, omdat de naam van de class reeds eindigt op **s**.
- (2) Als CodeFirst een property vindt waarvan de naam gelijk is aan de naam van de class, gevuld door **Id**, aanziet CodeFirst deze property als degene die bij de **primary key** hoort. De kolomnaam is ook **CampusId**. Er ligt nog geen verband tussen **Instructeur** en **Campus**. Je legt dit verband verder in de cursus.

17.3 DE DBCONTEXT CLASS

Bij een **EDMX** maakte **Visual Studio** een **DbContext** class voor je.

De **DbContext** class is het **aanspreekpunt** naar de database.

Bij **CodeFirst** maak je de **DbContext** class zelf.

Je voegt een class **EFCFContext** toe:

```

using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext
    {
        public DbSet<Instructeur> Instructeurs { get; set; }      // (1)
        public DbSet<Campus> Campussen { get; set; }            // (2)
    }
}

```

- (1) Je maakt een class die afgeleid is van **DbContext**.
- (2) Je maakt per **entity class** een property in je **DbContext** class. Deze property is van het type **DbSet**.

17.4 DE CONNECTIONSTRING

Je maakt in `App.config` een **connectionstring** naar de nieuwe database, onder `</startup>`

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0,
      Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
  </startup>

  <connectionStrings>
    <!--<add name="EFCFContext"
      providerName="System.Data.SqlClient"
      connectionString="Server=.\SQLEXPRESS;Database=EFCF;
      Trusted_Connection=true;"/>
    -->

    <add name="EFCFContext"
      providerName="System.Data.SqlClient"
      connectionString="Server=localhost;Database=EFCF;
      Trusted_Connection=true;" />
  </connectionStrings>

  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
    EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
      type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>

```

- (1) De name van de **connectionString** moet gelijk zijn aan je **DbContext** class.

17.5 DE DBCONTEXT GEBRUIKEN

Je wijzigt `Program.cs` :

```

using System;

namespace CodeFirstCursus
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var context = new EFCFContext())
            {
                var jean = new Instructeur
                {
                    Voornaam = "Jean",
                    Familienaam = "Smits",
                    Wedde = 1000,
                    InDienst = new DateTime(1994, 8, 1)
                };

```

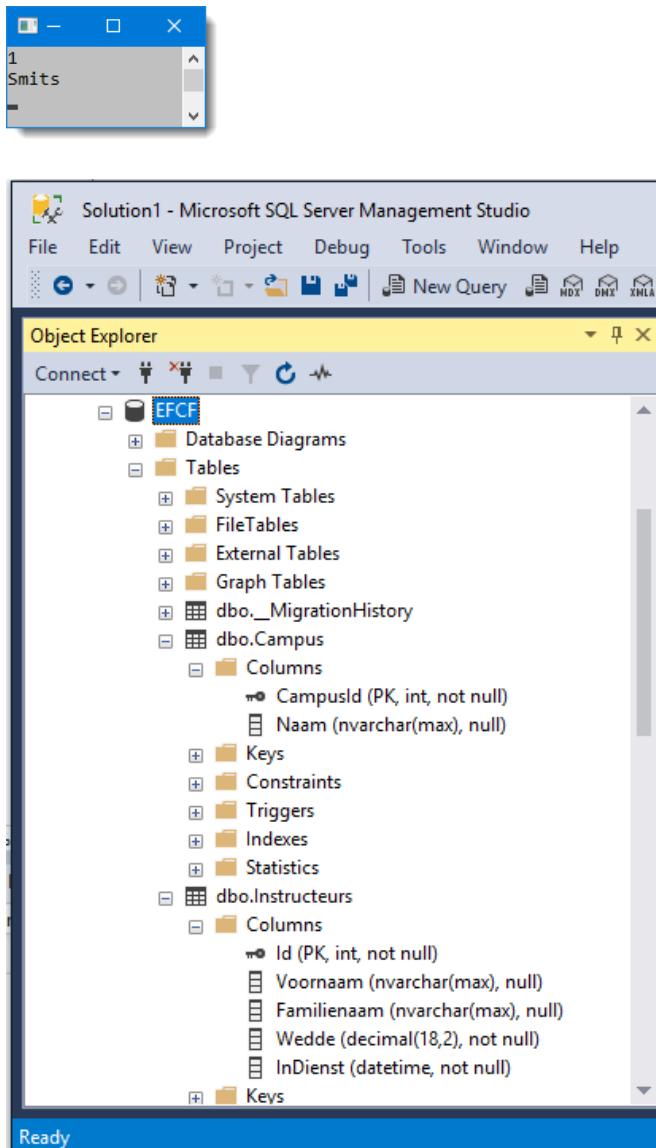
```
        context.Instructeurs.Add(jean);
        context.SaveChanges();

        Console.WriteLine(jean.Id);

        // zoeken op primary key
        Console.WriteLine(context.Instructeurs.Find(1).Familienaam);
        Console.ReadKey();
    }
}
```

Je ziet dat je de **entity classes** en de **context** gebruikt op dezelfde manier als ze zouden gemaakt zijn met een **EDMX**-bestand.

Je voert het programma uit. Je ziet daarna met de **SQL Server Management Studio** een nieuwe database **VDAB**, met de tables **Instructeurs** en **Campussen**:



17.6 DE DATABASE OPNIEUW MAKEN

Je voegt aan de class Instructeur een property **HeeftRijbewijs** toe :

```
using System;
namespace CodeFirstCursus
{
    public class Instructeur          // (1)
    {
        public int Id { get; set; }      // (2)
        public string Voornaam { get; set; } // (3)
        public string Familienaam { get; set; }
        public decimal Wedde { get; set; }
        public DateTime InDienst { get; set; }
        public bool HeeftRijbewijs { get; set; }

        public void Opslag(decimal percentage)
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
}
```

Je vult deze property in bij de entity die je maakt in Program.cs

```
var jean = new Instructeur
{
    Voornaam = "Jean",
    Familienaam = "Smits",
    Wedde = 1000,
    InDienst = new DateTime(1994, 8, 1),
    HeeftRijbewijs = true
};
```

De table **Instructeurs** bevat echter geen kolom **HeeftRijbewijs**.

Je kan dit oplossen door volgende regel als eerste te tikken in de method **Main** van **Program.cs** :

```
using System;
using System.Data.Entity;

namespace CodeFirstCursus
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Data.Entity.Database.SetInitializer(
                new DropCreateDatabaseIfModelChanges<EFCFContext>()); // (1)

            using (var context = new EFCFContext())
            {
                var jean = new Instructeur
                {
                    Voornaam = "Jean",
                    Familienaam = "Smits",
                    Wedde = 1000,
                    InDienst = new DateTime(1994, 8, 1),
                    HeeftRijbewijs = true
                };

                context.Instructeurs.Add(jean);
                context.SaveChanges();

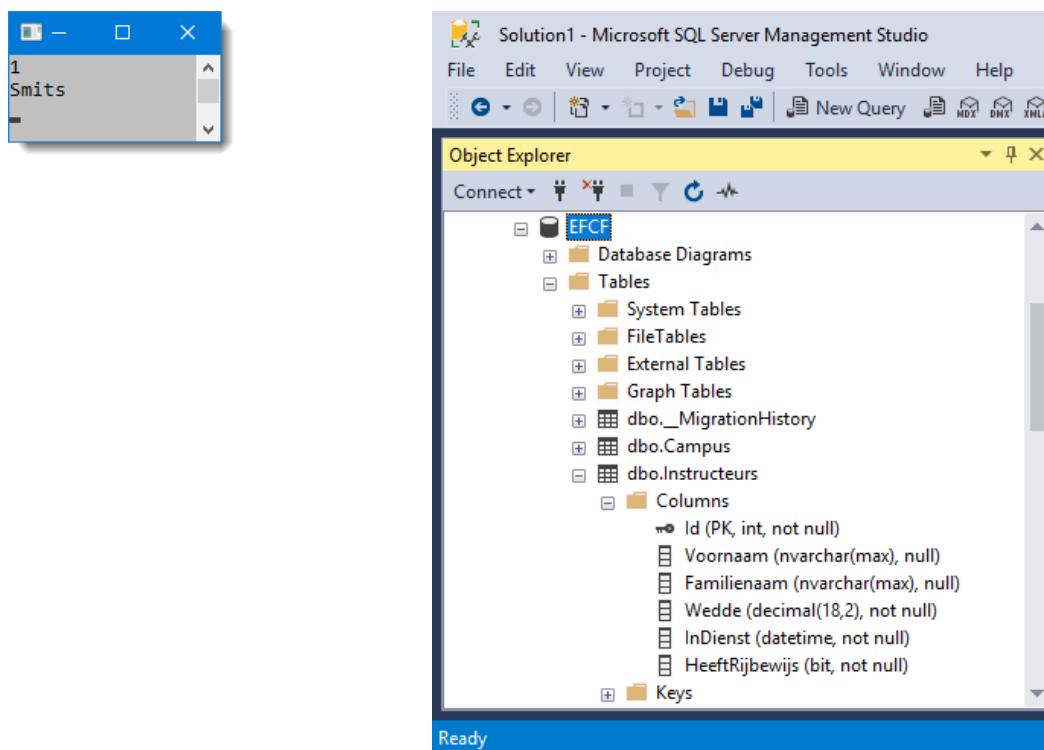
                Console.WriteLine(jean.Id);

                // zoeken op primary key
            }
        }
    }
}
```

```
        Console.WriteLine(context.Instructeurs.Find(1).Familienaam);
        Console.ReadKey();
    }
}
```

- (1) Met deze opdracht controleert CodeFirst of de tables in de database nog dezelfde structuur hebben als de bijbehorende entity classes.
Als dit niet het geval is, verwijdert CodeFirst de database en maakt hem opnieuw aan.
Het verwijderen lukt enkel als geen ander programma (bijvoorbeeld **SQL Server Management Studio**) de database heeft geopend.

Je voert het programma uit :



De table **Instructeurs** bevat daarna ook een kolom **HeeftRijbewijs**.

17.7 DE AANGEMAAKTE TABLE STRUCTUUR VERFIJNEN

17.7.1 EXPLICIET DE TABLE-NAAM INSTELLEN

Als je de automatische table-naam niet goed vindt, kan je die instellen met het attribuut **Table** bij de entity class. Je tikt voor class **Campus** :

```
[Table("Campussen")] //using System.ComponentModel.DataAnnotations.Schema;
```

```
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace CodeFirstCursus  
{  
    [Table("Campussen")] // using System.ComponentModel.DataAnnotations.Schema;  
    public class Campus
```

```
{  
    public int CampusId { get; set; }  
    public string Naam { get; set; }  
}
```

17.7.2 EXPLICIET DE KOLOMNAAM INSTELLEN

Als je de automatische kolomnaam niet goed vindt, kan je die instellen met het attribuut **Column** bij de property. Je tikt voor de property **Wedde** in de class **Instructeur** :

```
[Column("maandwedde")] //using System.ComponentModel.DataAnnotations.Schema;
```

```
using System;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace CodeFirstCursus  
{  
    public class Instructeur  
    {  
        public int Id { get; set; }  
        public string Voornaam { get; set; }  
        public string Familienaam { get; set; }  
  
        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;  
        public decimal Wedde { get; set; }  
  
        public DateTime InDienst { get; set; }  
        public bool HeeftRijbewijs { get; set; }  
  
        public void Opslag(decimal percentage)  
        {  
            Wedde *= (1M + percentage / 100M);  
        }  
    }  
}
```

17.7.3 EEN KOLOM INSTELLEN ALS VERPLICHT IN TE VULLEN

Als je de kolom, die bij een property hoort, wil instellen als verplicht in te vullen, doe je dit met het attribuut **Required** bij de property. Je tikt voor de property **Naam** in de class **Campus** :

```
[Required] //using System.ComponentModel.DataAnnotations
```

```
using System.ComponentModel.DataAnnotations.Schema;  
using System.ComponentModel.DataAnnotations;  
  
namespace CodeFirstCursus  
{  
    [Table("Campussen")] // using System.ComponentModel.DataAnnotations.Schema;  
    public class Campus  
    {  
        public int CampusId { get; set; }  
  
        [Required] // using System.ComponentModel.DataAnnotations;  
        public string Naam { get; set; }  
    }  
}
```

17.7.4 EEN KOOLM INSTELLEN ALS NIET VERPLICHT IN TE VULLEN

Een kolom die bij een property hoort met een primitief data-type (`int`, `long`, `bool`, ...) is standaard verplicht in te vullen. Je kan dit aanpassen door de property `nullable` te maken. Je wijzigt de property `HeeftRijbewijs` in de class `Instructeur` :

```
public bool? HeeftRijbewijs { get; set; }
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    public class Instructeur
    {
        public int Id { get; set; }
        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
        public decimal Wedde { get; set; }

        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public void Opslag(decimal percentage)
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
}
```

17.7.5 HET MAXIMUM AANTAL TEKENS IN EEN VARCHAR-KOOLM INSTELLEN

Je kan met het attribuut `StringLength` het maximum aantal tekens in een `varchar`-kolom instellen. Je tikt voor de property `Naam` in de class `Campus` volgende regel :

```
[StringLength(50)]
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("Campussen")]
    public class Campus
    {
        public int CampusId { get; set; }

        [Required]
        [StringLength(50)]
        public string Naam { get; set; }
    }
}
```

17.7.6 HET KOLOMTYPE INSTELLEN

Je kan met het attribuut `Column` het kolomtype instellen dat bij een property hoort. Bij een `Date`-property hoort standaard een `datetime`-kolom. Je tikt voor de property `InDienst` van de class

Instructeur volgende regel, om een date-kolom te bekomen :

```
[Column(TypeName="date")]
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    public class Instructeur
    {
        public int Id { get; set; }

        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
        public decimal Wedde { get; set; }

        [Column(TypeName = "date")]
        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public void Opslag(decimal percentage)
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
}
```

17.7.7 DE PROPERTY INSTELLEN DIE BIJ DE PRIMARY KEY HOORT

Je kan met het attribuut **Key** instellen welke property bij de **primary key** hoort. Je wijzigt de property **Id** van de class **Instructeur** naar :

```
[Key] // using System.ComponentModel.DataAnnotations;
public int InstructeurNr { get; set; }
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    public class Instructeur
    {
        //public int Id { get; set; }
        [Key] // using System.ComponentModel.DataAnnotations;
        public int InstructeurNr { get; set; }

        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
        public decimal Wedde { get; set; }

        [Column(TypeName = "date")]
        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public void Opslag(decimal percentage)
```

```
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
```

Je wijzigt `jean.Id` naar `jean.InstructeurNr` in `Program.cs`

```
using System;
using System.Data.Entity;

namespace CodeFirstCursus
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Data.Entity.Database.SetInitializer(
                new DropCreateDatabaseIfModelChanges<EFCFContext>());

            using (var context = new EFCFContext())
            {
                var jean = new Instructeur
                {
                    Voornaam = "Jean",
                    Familienaam = "Smits",
                    Wedde = 1000,
                    InDienst = new DateTime(1994, 8, 1),
                    HeeftRijbewijs = true
                };

                context.Instructeurs.Add(jean);
                context.SaveChanges();

                //Console.WriteLine(jean.Id);
                Console.WriteLine(jean.InstructeurNr);

                // zoeken op primary key
                Console.WriteLine(context.Instructeurs.Find(1).Familienaam);
                Console.ReadKey();
            }
        }
    }
}
```

17.7.8 EEN PRIMARY KEY DIE GEEN INT MET AUTONUMBER IS

Default is de primary key bij CodeFirst een `int` met `autonumber`. Als je de primary key van nieuwe entiteiten zelf invult in je C#-code, duid je dit aan met het attribuut `DatabaseGenerated`.

Je voegt volgende class `Land` toe:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("Landen")]
    public class Land
    {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
        public string LandCode { get; set; }

        public string Naam { get; set; }
    }
}
```

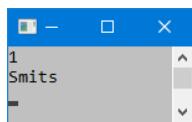
Je voegt volgende property toe aan de class EFCFContext

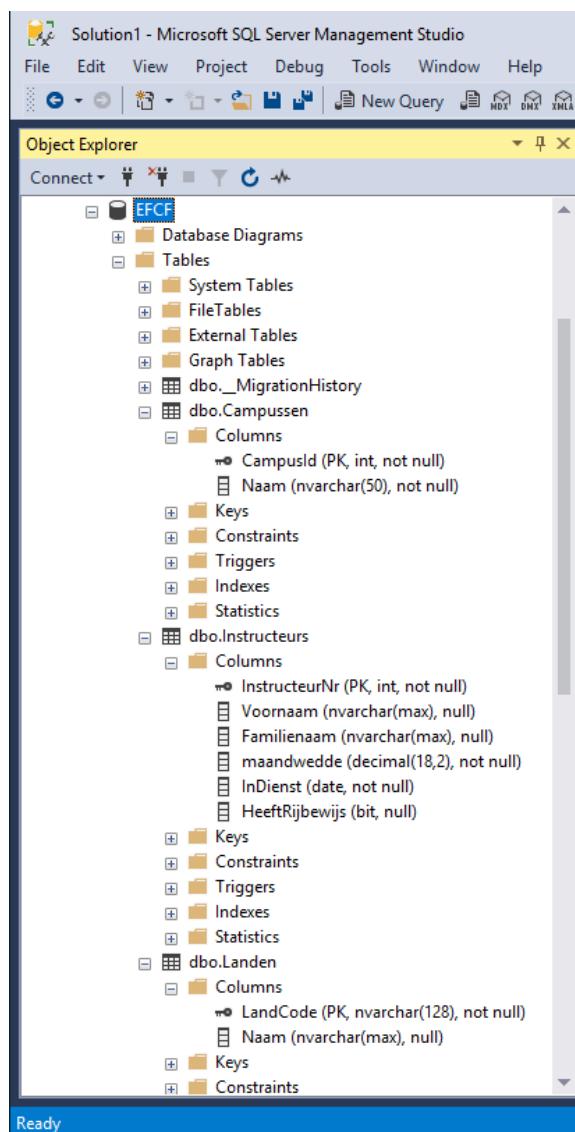
```
public DbSet<Land> Landen { get; set; }
```

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext:DbContext
    {
        public DbSet<Instructeur> Instructeurs { get; set; }
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }
    }
}
```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.





17.8 COMPLEX TYPE

Zowel een **instructeur** als een **campus** krijgen een **adres**, bestaande uit **straat**, **huisnummer**, **postcode** en **gemeente**. Dit worden in beide tables **Instructeurs** en **Campussen** 4 extra kolommen. In de classes is het vervelend om aan beide classes vier properties toe te voegen.

Een betere oplossing is deze vier properties één keer te beschrijven in een class **Adres** (een complex type). De class **Instructeur** en **Campus** hebben dan één nieuwe property, van het type **Adres**.

Je voegt de class **Adres** toe:

```
using System.ComponentModel.DataAnnotations.Schema;

[ComplexType] // (1)
public class Adres
{
    public string Straat { get; set; }
    public string HuisNr { get; set; }
    public string PostCode { get; set; }
    public string Gemeente { get; set; }
}
```

- (1) Je tikt **ComplexType** bij een class die een complex type voorstelt.

Je voegt aan de class **Campus** een property toe:

```
public Adres Adres { get; set; }
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("Campussen")]
    public class Campus
    {
        public int CampusId { get; set; }

        [Required]
        [StringLength(50)]
        public string Naam { get; set; }

        public Adres Adres { get; set; }
    }
}
```

Je voegt aan de class **Instructeur** een property toe:

```
public Adres Adres { get; set; }
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    public class Instructeur
    {
        //public int Id { get; set; }
        [Key]
        public int InstructeurNr { get; set; }

        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")]
        public decimal Wedde { get; set; }

        [Column(TypeName = "date")]
        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public Adres Adres { get; set; }

        public void Opslag(decimal percentage)
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
}
```

Je wijzigt **in Program.cs** de variabele **jean**:

```
using (var context = new EFCFContext())
{
```

```

var jean = new Instructeur
{
    Voornaam = "Jean",
    Familienaam = "Smits",
    Wedde = 1000,
    InDienst = new DateTime(1994, 8, 1),
    HeeftRijbewijs = true,
    Adres = new Adres
    {
        Straat = "Keizerslaan",
        HuisNr = "11",
        PostCode = "1000",
        Gemeente = "Brussel"
    }
};

```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.

De tables hebben nu volgende structuur :

Instructeurs	Campussen
InstructeurNr	CampusId
Voornaam	Naam
Familienaam	Adres_Straat
maandwedde	Adres_HuisNr
InDienst	Adres_PostCode
HeeftRijbewijs	Adres_Gemeente
Adres_Straat	
Adres_HuisNr	
Adres_PostCode	
Adres_Gemeente	

Je kan het herhalend woord **Adres_** in de kolomnamen verwijderen door bij de properties van de class **Adres** het attribuut **Column** te tikken:

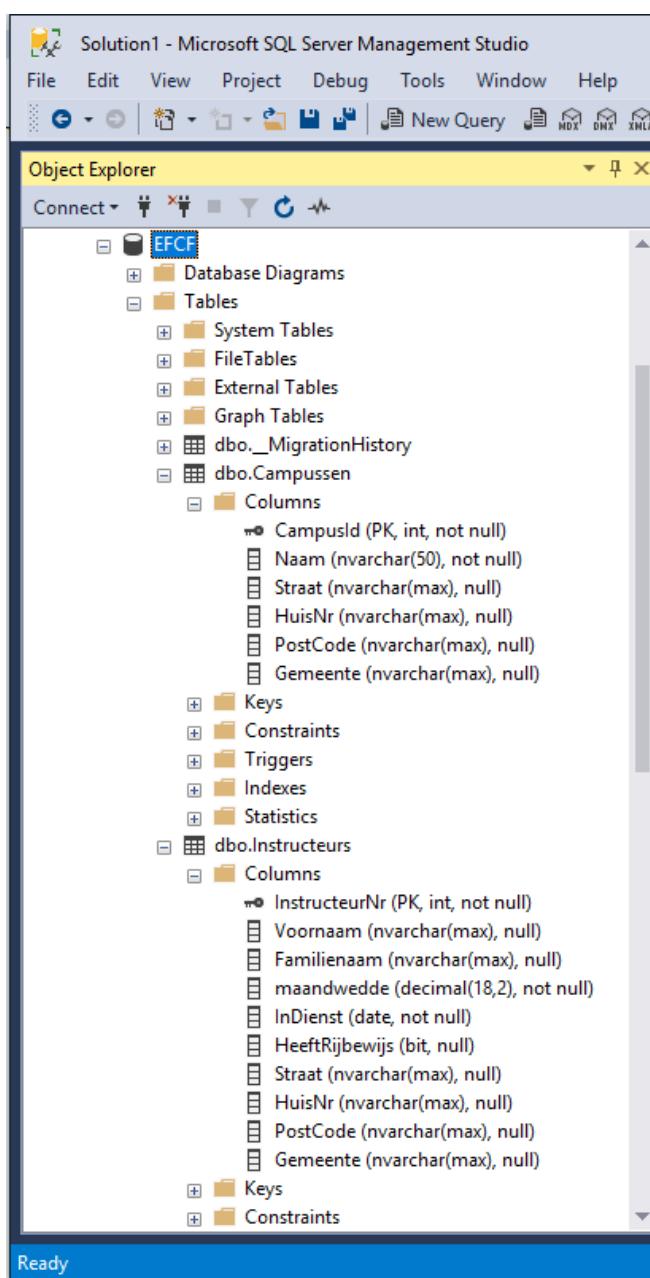
```
using System.ComponentModel.DataAnnotations.Schema;

[ComplexType]
public class Adres
{
    [Column("Straat")]
    public string Straat { get; set; }

    [Column("HuisNr")]
    public string HuisNr { get; set; }

    [Column("PostCode")]
    public string PostCode { get; set; }

    [Column("Gemeente")]
    public string Gemeente { get; set; }
}
```



17.9 INHERITANCE

17.9.1 TABLE PER HIERARCHY (TPH)

Deze manier om inheritance voor te stellen is de **default** manier in CodeFirst. Er is één table voor alle classes met een inheritance-verband.

Je voegt volgende classes toe :

TPHCursus

```
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("TPHCursussen")]
    public abstract class TPHCursus
    {
        public int Id { get; set; }
        public string Naam { get; set; }
    }
}
```

TPHKlassikaleCursus

```
using System;

namespace CodeFirstCursus
{
    public class TPHKlassikaleCursus : TPHCursus
    {
        public DateTime Van { get; set; }
        public DateTime Tot { get; set; }
    }
}
```

TPHZelfstudieCursus

```
namespace CodeFirstCursus
{
    public class TPHZelfstudieCursus : TPHCursus
    {
        public int AantalDagen { get; set; }
    }
}
```

Je voegt volgende property toe aan de class **EFCFContext** :

```
public DbSet<TPHCursus> TPHCursussen { get; set; }
```

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext:DbContext
    {
        public DbSet<Instructeur> Instructeurs { get; set; }
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }
    }
}
```

```
        public DbSet<TPHCursus> TPHCursussen { get; set; }  
    }
```

Je wijzigt de code in de using van de Main van **Program.cs**:

```
using System;  
using System.Data.Entity;  
  
namespace CodeFirstCursus  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            System.Data.Entity.Database.SetInitializer(  
                new DropCreateDatabaseIfModelChanges<EFCFContext>()); // (1)  
  
            using (var context = new EFCFContext())  
            {  
                . . . . .  
  
                // ======  
                // Inheritance TPH  
                // ======  
                context.TPHCursussen.Add(new TPHKlassikaleCursus  
                {  
                    Naam = "Frans in 24 uur",  
                    Van = DateTime.Today,  
                    Tot = DateTime.Today  
                });  
  
                context.TPHCursussen.Add(new TPHZelfstudieCursus  
                {  
                    Naam = "Engels in 24 uur",  
                    AantalDagen = 1  
                });  
  
                context.SaveChanges();  
  
                Console.WriteLine("Einde");  
                Console.ReadKey();  
            }  
        }  
    }  
}
```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.

De bijbehorende table:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the top right, the title bar reads "HPLAPTOP.EFCF - Diagram_0* - Microsoft SQL Server Management Studio". Below it is the "File" menu and other standard toolbar icons. The main area is the "Object Explorer" pane, which is expanded to show the database structure. Under the "Tables" node, there is a sub-node for "dbo" which contains "TPHCursussen". This table is further expanded to show its columns: Id, Naam, Van, Tot, AantalDagen, and Discriminator. The "Discriminator" column is highlighted with a red border. To the left of the Object Explorer is a small preview window titled "TPHCursussen" showing a grid of data with the same six columns.

CodeFirst heeft een kolom **Discriminator** toegevoegd om het onderscheid te maken tussen **klassikale cursussen** en **zelfstudiecursussen**.

CodeFirst vult deze kolom met de waarde **TPHKlassikaleCursus** of **TPHZelfstudieCursus**.

	Id	Naam	Van	Tot	AantalDagen	Discriminator
1	1	Frans in 24 uur	2018-04-06 00:00:00.000	2018-04-06 00:00:00.000	NULL	KlassikaleCursusTPH
2	2	Engels in 24 uur	NULL	NULL	1	ZelfstudieCursusTPH

Je kan de kolomnaam en de kolomwaarden instellen in je **DbContext** class

Je voegt volgende method toe aan **EFCFContext**

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext // (1)
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (2)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder) // (3)
        {
            modelBuilder.Entity<TPHKlassikaleCursus>()
                .Map(m => m.Requires("Soort").HasValue("K")); // (4)

            modelBuilder.Entity<TPHZelfstudieCursus>()
                .Map(m => m.Requires("Soort").HasValue("Z"));
        }
    }
}
```

```
}
```

[3] CodeFirst voert deze method uit bij het aanmaken van de database.

[4] Met deze opdracht geef je de **discriminator** kolom de naam **Soort** en geef je aan dat deze kolom de waarde **K** moet bevatten bij een **klassikale cursus**.

Je kan het programma uitproberen.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the 'EFCF' database, there is a 'Tables' node expanded, showing several tables like 'System Tables', 'FileTables', 'External Tables', 'Graph Tables', 'dbo._MigrationHistory', 'dbo.Campussen', 'dbo.Instructeurs', 'dbo.Landen', and 'dbo.TPHCursussen'. The 'dbo.TPHCursussen' table is selected and its details are shown in the main pane. The table has columns: Id (PK, int, not null), Naam (nvarchar(max), null), Van (datetime, null), Tot (datetime, null), AantalDagen (int, null), and Soort (nvarchar(128), not null). The 'Soort' column is highlighted with a blue selection bar. Below the table definition, there are sections for Keys, Constraints, Triggers, and Indexes. At the bottom of the main pane, there is a grid showing data for the 'TPHCursussen' table:

	Id	Naam	Van	Tot	AantalDagen	Soort
1	1	Frans in 24 uur	2018-04-06 00:00:00.000	2018-04-06 00:00:00.000	NULL	K
2	2	Engels in 24 uur	NULL	NULL	1	Z

17.9.2 TABLE PER TYPE (TPT)

Bij TPT is er één table per class in de inheritance-structuur.

CodeFirst past TPT toe als je ook een **Table**-attribuut tikt bij de derived classes.

Maak volgende classes aan:

Class **TPTCursus**

```
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
```

```
[Table("TPTCursussen")]
public abstract class TPTCursus
{
    public int Id { get; set; }
    public string Naam { get; set; }
}
```

Class **TPTKlassikaleCursus**

```
using System;

namespace CodeFirstCursus
{
    public class TPTKlassikaleCursus : TPTCursus
    {
        public DateTime Van { get; set; }
        public DateTime Tot { get; set; }
    }
}
```

Class **TPTZelfstudieCursus**

```
namespace CodeFirstCursus
{
    public class TPTZelfstudieCursus : TPTCursus
    {
        public int AantalDagen { get; set; }
    }
}
```

Je tikt voor de class **TPTKlassikaleCursus** volgende regel:

```
[Table("TPTKlassikalecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("TPTKlassikalecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
    public class TPTKlassikaleCursus : TPTCursus
    {
        public DateTime Van { get; set; }
        public DateTime Tot { get; set; }
    }
}
```

Je tikt voor de class **TPTZelfstudieCursus** volgende regel:

```
[Table("TPTZelfstudiecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
```

```
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("TPTZelfstudiecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
    public class TPTZelfstudieCursus : TPTCursus
    {
        public int AantalDagen { get; set; }
    }
}
```

}

Je wijzigt de class **EFCFContext**:

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext
    {
        public DbSet<Instructeur> Instructeurs { get; set; }
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }
        public DbSet<CursusTPT> TPTCursussen { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<KlassikaleCursusTPH>()
                .Map(m => m.Requires("Soort").HasValue("K"));

            modelBuilder.Entity<ZelfstudieCursusTPH>()
                .Map(m => m.Requires("Soort").HasValue("Z"));
        }
    }
}
```

Je wijzigt de **Main** methode van de class **Program.cs**

```
}
```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.

De bijbehorende tables:

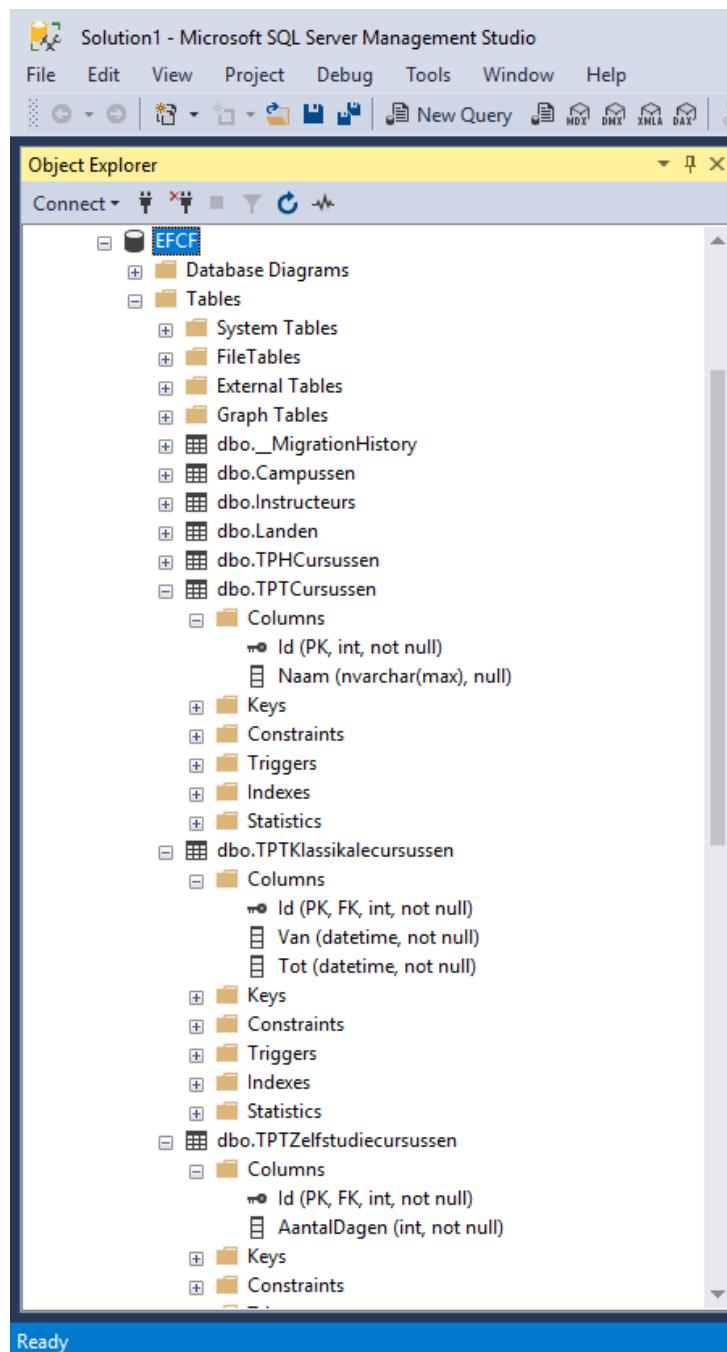
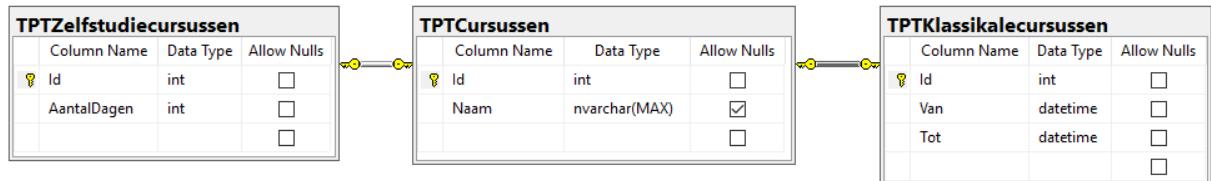


Table TPTTurssussen

	Id	Naam
1	1	Frans in 24 uur
2	2	Engels in 24 uur

Table TPTKlassikaleCursussen

	Id	Van	Tot
1	1	2018-04-06 00:00:00.000	2018-04-06 00:00:00.000

Table TPTZelfstudieCursussen

	Id	AantalDagen
1	2	1

17.9.3 TABLE PER CONCRETE CLASS (TPC)

Bij TPC is er één table per concrete subclass in de inheritance-structuur.

CodeFirst past TPC toe als de primary key geen int met autonumber is.

Maak volgende classes aan:

Class TPCCursus

```
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("TPCCursussen")]
    public abstract class TPCCursus
    {
        public int Id { get; set; }
        public string Naam { get; set; }
    }
}
```

Class TPCKlassikaleCursus

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("TPCKlassikalecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
    public class TPCKlassikaleCursus : TPCCursus
    {
        public DateTime Van { get; set; }
        public DateTime Tot { get; set; }
    }
}
```

Class TPCZelfstudieCursus

```
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("TPCZelfstudiecursussen")] // using System.ComponentModel.DataAnnotations.Schema;
    public class TPCZelfstudieCursus : TPCCursus
    {
        public int AantalDagen { get; set; }
    }
}
```

Je verwijdert het attribuut **Table** bij de class **TPCCursus** en je wijzigt de property **Id** van die class

```
[DatabaseGenerated(DatabaseGeneratedOption.Identity)] // (1)
public Guid Id { get; set; } // using System; // (2)
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System;

namespace CodeFirstCursus
{
    //#[Table("TPCCursussen")]
    public abstract class TPCCursus
    {
        //public int Id { get; set; }
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)] // (1)
        public Guid Id { get; set; } // using System; // (2)

        public string Naam { get; set; }
    }
}
```

- (1) Je geeft aan dat de inhoud van de primary key bij een nieuw record door de database wordt ingevuld (en niet in je code).
- (2) Je kiest **Guid** als type van de property.
SQL Server zal bij een nieuw record een unieke string invullen in de primary key.

Je voegt aan de class **EFCFContext** de method **onModelCreating** toe:

```
modelBuilder.Entity<KlassikaleCursus>().Map(m => m.MapInheritedProperties()); // (1)
modelBuilder.Entity<ZelfstudieCursus>().Map(m => m.MapInheritedProperties());
```

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext // (1)
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (2)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }
        public DbSet<TPTCursus> TPTCursussen { get; set; }
        public DbSet<TPCCursus> TPCCursussen { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder) // (3)
        {
            modelBuilder.Entity<KlassikaleCursusTPH>()
                .Map(m => m.Requires("Soort").HasValue("K")); // (4)

            modelBuilder.Entity<ZelfstudieCursusTPH>()
                .Map(m => m.Requires("Soort").HasValue("Z"));

            modelBuilder.Entity<KlassikaleCursusTPC>()
                .Map(m => m.MapInheritedProperties()); // (5)
        }
    }
}
```

```
        modelBuilder.Entity<ZelfstudieCursusTPC>()
            .Map(m => m.MapInheritedProperties());
    }
}
```

- [5] Je geeft met de method `MapInheritedProperties` aan dat Code First alle properties (ook de geërfde properties) in de table moet plaatsen die bij de class hoort.

Je wijzigt de **Main** methode van de class **Program.cs**

```
using System;
using System.Data.Entity;

namespace CodeFirstCursus
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Data.Entity.Database.SetInitializer(
                new DropCreateDatabaseIfModelChanges<EFCFContext>()); // (1)

            using (var context = new EFCFContext())
            {

                . . . . . . . . .

                // =====
                // Inheritance TPC
                // =====
                context.TPCCursussen.Add(new TPCKlassikaleCursus
                {
                    Naam = "Frans in 24 uur",
                    Van = DateTime.Today,
                    Tot = DateTime.Today
                });

                context.TPCCursussen.Add(new TPCZelfstudieCursus
                {
                    Naam = "Engels in 24 uur",
                    AantalDagen = 1
                });

                context.SaveChanges();

                Console.WriteLine("Einde");
                Console.ReadKey();
            }
        }
    }
}
```

Je kan het programma uitproberen.

De bijbehorende tabels:

TPCKlassikalecursussen			
	Column Name	Data Type	Allow Nulls
	Id	uniqueidentifier	<input type="checkbox"/>
	Naam	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Van	datetime	<input type="checkbox"/>
	Tot	datetime	<input type="checkbox"/>
			<input type="checkbox"/>

TPCZelfstudiecursussen			
Column Name	Data Type	Allow Nulls	
Id	uniqueidentifier	<input type="checkbox"/>	
Naam	nvarchar(MAX)	<input checked="" type="checkbox"/>	
AantalDagen	int	<input type="checkbox"/>	
		<input type="checkbox"/>	

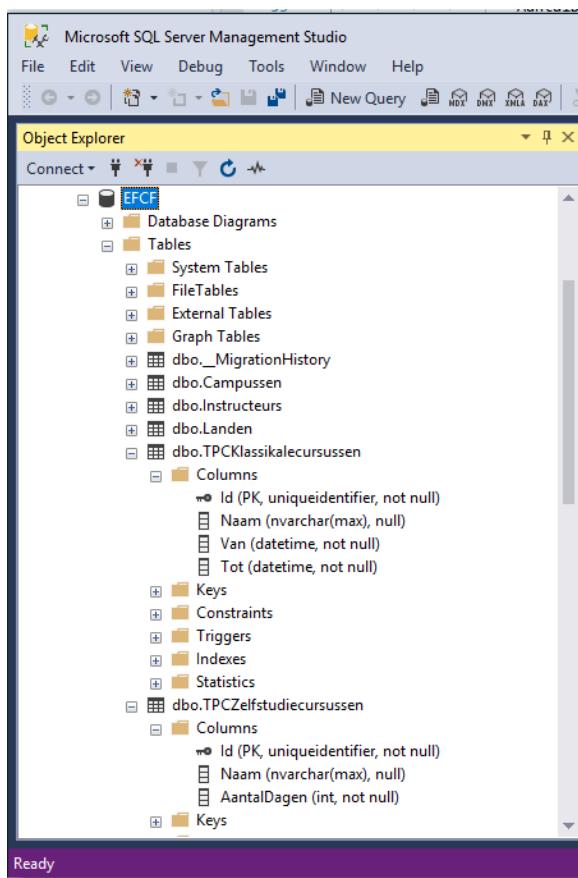


Table TPCKlassikaleCursussen

	Id	Naam	Van	Tot
1	34766581-9A39-E811-A50C-F0D5BF00A1F3	Frans in 24 uur	2018-04-06 00:00:00.000	2018-04-06 00:00:00.000

Table TPCZelfstudieCursussen

	Id	Naam	AantalDagen
1	35766581-9A39-E811-A50C-F0D5BF00A1F3	Engels in 24 uur	1

17.10 ASSOCIATIES TUSSEN ENTITIES

17.10.1 ÉEN-OP-VEEL-ASSOCIATIES

Je voegt een één-op-veel-relatie toe tussen **ASSCampus** en **ASSInstructeur** :

Voeg een nieuwe class toe met de naam **ASSCampus** en copieer de code van de class **Campus**.

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("ASSCampussen")] // using System.ComponentModel.DataAnnotations.Schema;
    public class ASSCampus
    {
        public int ASSCampusId { get; set; } // (2)
```

```

[Required]           // using System.ComponentModel.DataAnnotations;
[StringLength(50)]
public string Naam { get; set; }

public Adres Adres { get; set; }
}
}

```

Voeg een nieuwe class toe met de naam **ASSInstructeur** en copieer de code van de class **Instructeur**.

```

using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("ASSInstructeurs")] // using System.ComponentModel.DataAnnotations.Schema;
    public class ASSInstructeur
    {
        //public int Id { get; set; }

        [Key]
        // using System.ComponentModel.DataAnnotations;
        public int InstructeurNr { get; set; }

        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
        public decimal Wedde { get; set; }

        [Column(TypeName = "date")]
        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public Adres Adres { get; set; }

        public void Opslag(decimal percentage)
        {
            Wedde *= (1M + percentage / 100M);
        }
    }
}

```

Voeg volgende code toe aan de class **EFCFContext** :

```

public DbSet<ASSInstructeur> ASSInstructeurs { get; set; }
public DbSet<ASSCampus> ASSCampussen { get; set; }

```

```

using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext // (1)
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (2)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }
        public DbSet<TPTCursus> TPTCursussen { get; set; }
        public DbSet<TPCCursus> TPCCursussen { get; set; }

        public DbSet<ASSInstructeur> ASSInstructeurs { get; set; }
    }
}

```

```
public DbSet<ASSCampus> ASSCampussen { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder) // (3)
{
    modelBuilder.Entity<TPHKlassikaleCursus>()
        .Map(m => m.Requires("Soort").HasValue("K")); // (4)

    modelBuilder.Entity<TPHZelfstudieCursus>()
        .Map(m => m.Requires("Soort").HasValue("Z"));

    modelBuilder.Entity<TPCKlassikaleCursus>()
        .Map(m => m.MapInheritedProperties()); // (5)

    modelBuilder.Entity<TPCZelfstudieCursus>()
        .Map(m => m.MapInheritedProperties());
}
```

Je voegt volgende property toe aan de class **ASSCampus**:

```
public virtual ICollection<ASSInstructeur> ASSInstructeurs { get; set; } // (3)  
// using System.Collections.Generic;
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Collections.Generic;

namespace CodeFirstCursus
{
    [Table("ASSCampussen")]           // using System.ComponentModel.DataAnnotations.Schema;
    public class ASSCampus           // (1)
    {
        public int ASSCampusId { get; set; } // (2)

        [Required]                      // using System.ComponentModel.DataAnnotations;
        [StringLength(50)]               // using System.ComponentModel.DataAnnotations;
        public string Naam { get; set; }

        public Adres Adres { get; set; }

        // -----
        // Associaties
        // -----
        public virtual ICollection<ASSInstructeur> ASSInstructeurs { get; set; } // (3) -
    }
}
```

- [3] Deze property stelt de verzameling **instructeurs** voor die bij de huidige **campus** horen. Het type van zo'n property moet **ICollection** zijn bij Code First.
Je kan met **foreach** itereren over een **ICollection**.
Zo'n property moet **virtual** zijn, anders werkt **lazy loading** niet.

Je voegt volgende property toe aan de class **ASSInstructeur** :

```
public virtual ASSCampus Campus { get; set; } // (1)  
public int ASSCampusId { get; set; } // (2)
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("ASSInstructeurs")]
        // using System.ComponentModel.DataAnnotations.Schema;
```

```
public class ASSInstructeur
{
    //public int Id { get; set; }
    [Key]                                     // using System.ComponentModel.DataAnnotations;
    public int InstructeurNr { get; set; }

    public string Voornaam { get; set; }
    public string Familiennaam { get; set; }

    [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
    public decimal Wedde { get; set; }

    [Column(TypeName = "date")]
    public DateTime InDienst { get; set; }

    public bool? HeeftRijbewijs { get; set; }
    //public bool HeeftRijbewijs { get; set; }

    public Adres Adres { get; set; }

    // -----
    // Associaties
    // -----
    public virtual ASSCampus Campus { get; set; }    // (1)
    public int ASSCampusId { get; set; }              // (2)

    public void Opslag(decimal percentage)
    {
        Wedde *= (1M + percentage / 100M);
    }
}
```

- (1) Deze property stelt de **campus** voor die bij de **instructeur** hoort.
Zo'n property moet **virtual** zijn, anders werkt **lazy loading** niet.
 - (2) Deze property stelt het **campusnummer** voor van de **campus** die bij de **instructeur** hoort. De property-naam moet gelijk zijn aan de property die in **Campus** de primary key voorstelt.

Je wijzigt de code in de **using** van de **Main** van **Program.cs**:

```
        }

        var instructeur = new ASSInstructeur
        {
            Voornaam = "Marcel",
            Familienaam = "Kiekeboe",
            Wedde = 100,
            InDienst = new DateTime(1955, 1, 1),
            HeeftRijbewijs = true,
            Adres = new Adres
            {
                Straat = "Merholaan",
                HuisNr = "1B",
                PostCode = "2981",
                Gemeente = "Zoersel-Parwijs"
            },
            Campus = campus
        };

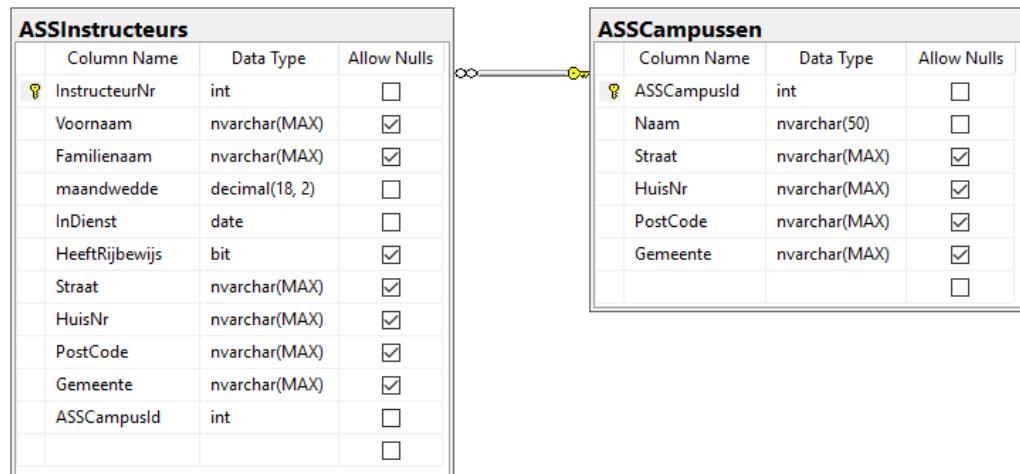
        context.ASSCampussen.Add(campus);
        context.ASSInstructeurs.Add(instructeur);

        // =====
        context.SaveChanges();

        Console.WriteLine("Einde");
        Console.ReadKey();
    }
}
```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.

De table **ASSInstructeurs** bevat nu een foreign-key-kolom **ASSCampusId**, die verwijst naar de table **ASSCampussen**:



Object Explorer

- EFCF
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo._MigrationHistory
 - dbo.ASSCampussen
 - Columns
 - ASSCampusId (PK, int, not null)
 - Naam (nvarchar(50), not null)
 - Straat (nvarchar(max), null)
 - HuisNr (nvarchar(max), null)
 - PostCode (nvarchar(max), null)
 - Gemeente (nvarchar(max), null)
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.ASInstructeurs
 - Columns
 - InstructeurNr (PK, int, not null)
 - Voornaam (nvarchar(max), null)
 - Familienaam (nvarchar(max), null)
 - maandwedde (decimal(18,2), not null)
 - InDienst (date, not null)
 - HeeftRijbewijs (bit, null)
 - Straat (nvarchar(max), null)
 - HuisNr (nvarchar(max), null)
 - PostCode (nvarchar(max), null)
 - Gemeente (nvarchar(max), null)
 - ASSCampusId (FK, int, not null)
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.Campussen
 - dbo.Instructeurs
 - dbo.Landen

Table ASSCampussen

	ASSCampusId	Naam	Straat	HuisNr	PostCode	Gemeente
1	1	Delos	Vlamingstraat	10	8560	Wevelgem

Table ASInstructeurs

	InstructeurNr	Voornaam	Familienaam	maandwedde	InDienst	HeeftRijbewijs	Straat	HuisNr	PostCode	Gemeente	ASSCampusId
1	1	Marcel	Kiekeboe	100.00	1955-01-01	1	Merholaan	1B	2981	Zoersel-Parwijs	1

17.10.2 VEEL-OP-VEEL-ASSOCIATIES

Als twee classes een veel-op-veel-associatie hebben, bevatten beide classes een **ICollection** property die verwijst naar de andere class.

Het voorbeeld is een veel-op-veel-associatie tussen **verantwoordelijkheden** en **instructeurs**.

Je voegt een class **ASSVerantwoordelijkheid** toe:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstCursus
{
    [Table("ASSVerantwoordelijkheden")]
    public class ASSVerantwoordelijkheid
    {
        public int ASSVerantwoordelijkheidId { get; set; }
        public string Naam { get; set; }
        public virtual ICollection<ASSInstructeur> Instructeurs { get; set; }
    }
}
```

Je voegt aan de class **ASSInstructeur** een property **Verantwoordelijkheden** toe:

```
public virtual ICollection<ASSVerantwoordelijkheid> Verantwoordelijkheden { get; set; } //  
using System.Collections.Generic;
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Collections.Generic;

namespace CodeFirstCursus
{
    [Table("ASSInstructeurs")] // using System.ComponentModel.DataAnnotations.Schema;
    public class ASSInstructeur
    {
        //public int Id { get; set; }
        [Key] // using System.ComponentModel.DataAnnotations;
        public int ASSInstructeurNr { get; set; }

        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        [Column("maandwedde")] // using System.ComponentModel.DataAnnotations.Schema;
        public decimal Wedde { get; set; }

        [Column(TypeName = "date")]
        public DateTime InDienst { get; set; }

        public bool? HeeftRijbewijs { get; set; }
        //public bool HeeftRijbewijs { get; set; }

        public Adres Adres { get; set; }

        // -----
        // Associaties
        // -----
        public virtual ASSCampus Campus { get; set; } // (1)
        public int ASSCampusId { get; set; } // (2)
        public virtual ICollection<ASSVerantwoordelijkheid> ASSVerantwoordelijkheden { get;
set; } // using System.Collections.Generic;

        public void Opslag(decimal percentage)
        {
    }
```

```
        Wedde *= (1M + percentage / 100M);  
    }  
}
```

Je voegt volgende property toe aan de class **EFCFContext**:

```
public DbSet<ASSVerantwoordelijkheid> Verantwoordelijkheden { get; set; }
```

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext // (1)
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (2)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }
        public DbSet<TPTCursus> TPTCursussen { get; set; }
        public DbSet<TPCCursus> TPCCursussen { get; set; }

        public DbSet<ASSInstructeur> ASSInstructeurs { get; set; }
        public DbSet<ASSCampus> ASSCampussen { get; set; }
        public DbSet<ASSVerantwoordelijkheid> ASSVerantwoordelijkheden { get; set; } // (3)

        protected override void OnModelCreating(DbModelBuilder modelBuilder) // (3)
        {
            modelBuilder.Entity<TPHKlassikaleCursus>()
                .Map(m => m.Requires("Soort").HasValue("K")); // (4)

            modelBuilder.Entity<TPHZelfstudieCursus>()
                .Map(m => m.Requires("Soort").HasValue("Z"));

            modelBuilder.Entity<TPCKlassikaleCursus>()
                .Map(m => m.MapInheritedProperties()); // (5)

            modelBuilder.Entity<TPCZelfstudieCursus>()
                .Map(m => m.MapInheritedProperties());
        }
    }
}
```

Je voegt volgende code toe in `Program.cs`, vóór `context.SaveChanges()`:

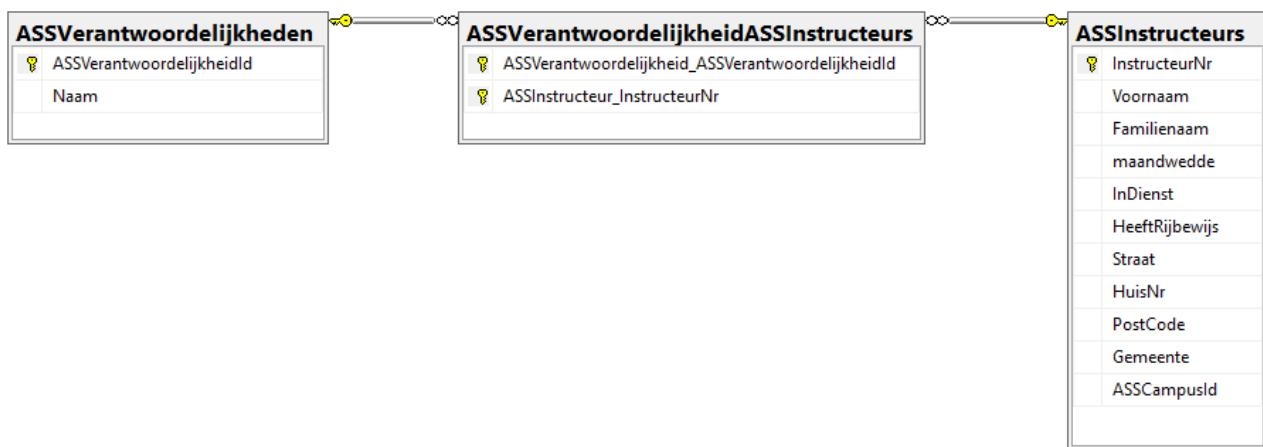
```
        instructeur.ASSVerantwoordelijkheden =
            new List<ASSVerantwoordelijkheid> {verantwoordelijkheid};
            // using System.Collections.Generic;

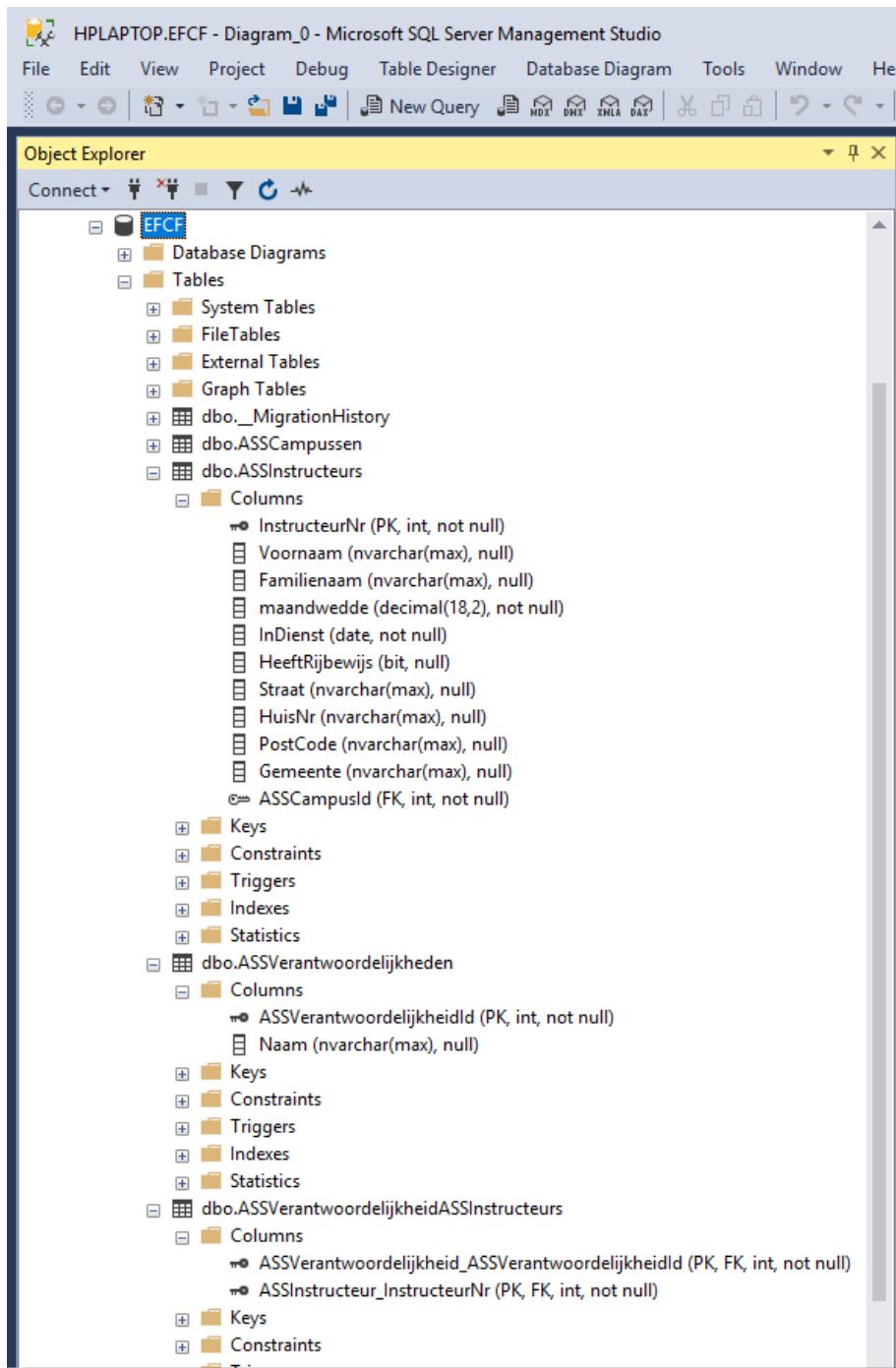
            context.ASSVerantwoordelijkheden.Add(verantwoordelijkheid);

            // =====
            context.SaveChanges();

            Console.WriteLine("Einde");
            Console.ReadKey();
        }
    }
}
```

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.





Je kan de naam van de tussentabel en de namen van de kolommen in die tussentabel instellen, met een extra opdracht in de method `onModelCreating` van de class `EFCFContext`:

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (1)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; } // (2)
        public DbSet<PTPCursus> TPTCursussen { get; set; }
    }
}
```

```

public DbSet<TPCCursus> TPCCursussen { get; set; }

public DbSet<ASSInstructeur> ASSInstructeurs { get; set; }
public DbSet<ASSCampus> ASSCampussen { get; set; }
public DbSet<ASSVerantwoordelijkheid> ASSVerantwoordelijkheden { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder) // (3)
{
    modelBuilder.Entity<TPHKlassikaleCursus>()
        .Map(m => m.Requires("Soort").HasValue("K")); // (4)

    modelBuilder.Entity<TPHZelfstudieCursus>()
        .Map(m => m.Requires("Soort").HasValue("Z"));

    modelBuilder.Entity<TPCKlassikaleCursus>()
        .Map(m => m.MapInheritedProperties()); // (5)

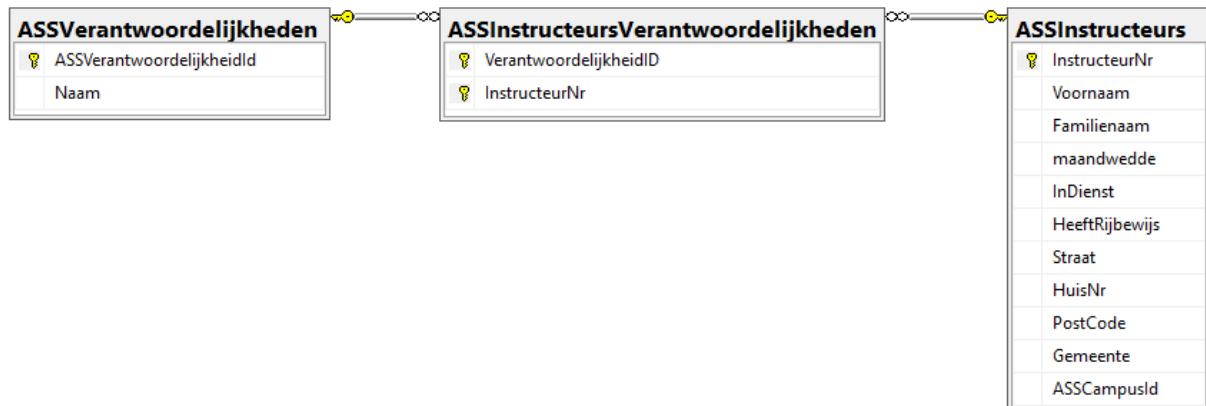
    modelBuilder.Entity<TPCZelfstudieCursus>()
        .Map(m => m.MapInheritedProperties());

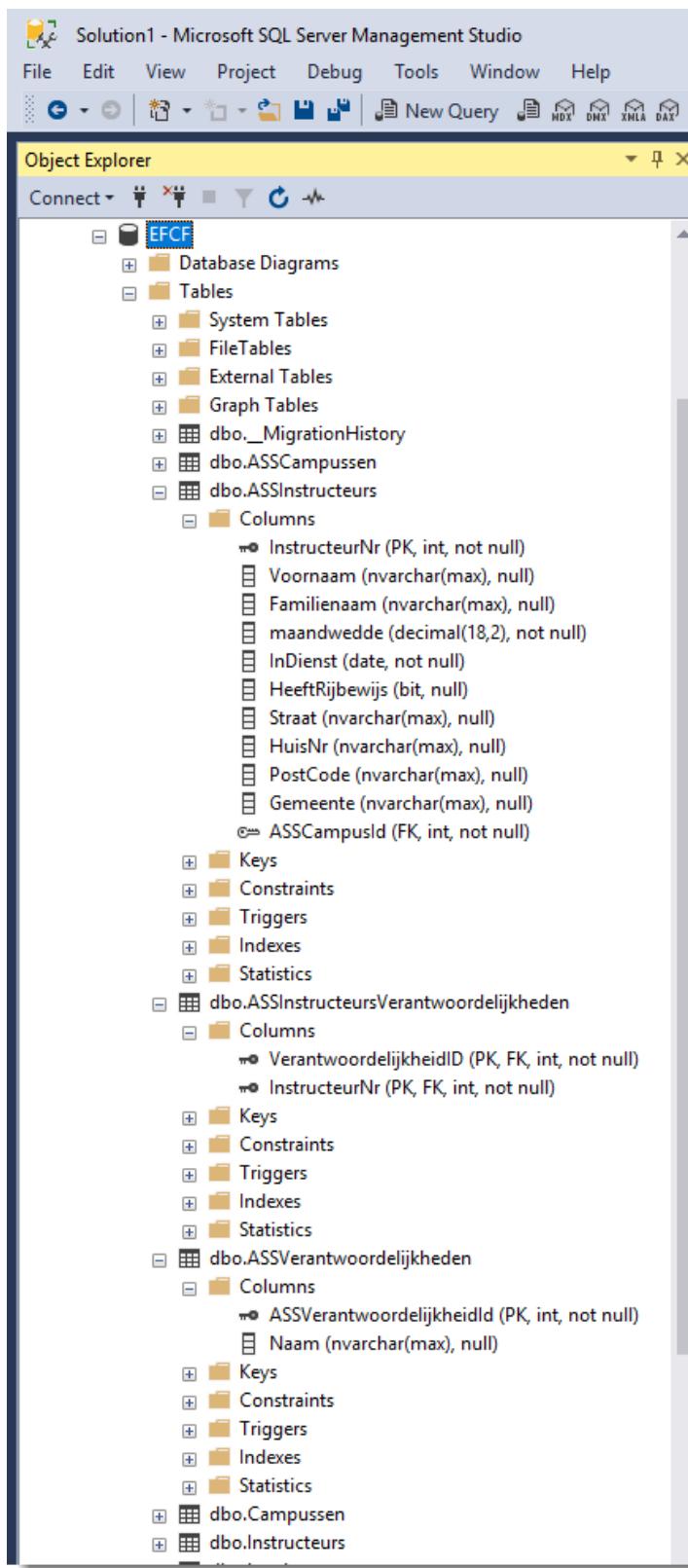
    modelBuilder.Entity<ASSInstructeur>() // (6)
        .HasMany(i => i.ASSVerantwoordelijkheden) // (7)
        .WithMany(v => v.ASSInstructeurs) // (8)
        .Map(c => c.ToTable("InstructeursVerantwoordelijkheden")) // (9)
        .MapLeftKey("VerantwoordelijkheidID") // (10)
        .MapRightKey("InstructeurNr"); // (11)
}
}
}

```

- [6] Je vermeldt bij **Entity** tussen < en > de naam van één van beide classes.
- [7] Je vermeldt bij **HasMany** de property in de class bij (6) die verwijst naar de tweede class.
- [8] Je vermeldt bij **WithMany** de property in de tweede class die verwijst naar de class bij (6)
- [9] Je vermeldt bij **ToTable** de naam van de tussentabel.
- [10] Je vermeldt bij **MapLeftKey** de kolomnaam die hoort bij de tweede class in de tussentabel.
- [11] Je vermeldt bij **MapRightKey** de kolomnaam die hoort bij de class bij (6) in de tussentabel.

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.





17.10.3 EEN ASSOCIATIE NAAR DEZELFDE TABLE

Het voorbeeld is een class **ASSCursist**. Iedere **cursist** heeft een **mentor**, een **mentor** kan meerdere **cursisten** als **beschermeling** hebben.

Je voegt de class **ASSCursist** toe:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CodeFirstCursus
{
    [Table("ASSCursisten")]
    public class ASSCursist
    {
        [Key]
        public int ASSCursistId { get; set; }
        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        public virtual ICollection<ASSCursist> Beschermelingen { get; set; } // (1)

        [InverseProperty("Beschermelingen")]
        public virtual ASSCursist Mentor { get; set; } // (2) // (3)
    }
}
```

- (1) Deze property stelt de **beschermelingen** van één **mentor** voor.
- (2) De property bij (3) stelt de **mentor** van een **cursist** voor. Je vermeldt bij **InverseProperty** de property (1) die de andere kant van de associatie voorstelt.

Je voegt volgende property toe aan de class **EFCFContext**:

```
public DbSet<ASSCursist> Cursisten {get; set;}
```

```
using System.Data.Entity;

namespace CodeFirstCursus
{
    class EFCFContext : DbContext // (1)
    {
        public DbSet<Instructeur> Instructeurs { get; set; } // (2)
        public DbSet<Campus> Campussen { get; set; }
        public DbSet<Land> Landen { get; set; }

        public DbSet<TPHCursus> TPHCursussen { get; set; }
        public DbSet<TPTCursus> TPTCursussen { get; set; }
        public DbSet<TPCCursus> TPCCursussen { get; set; }

        public DbSet<ASSInstructeur> ASSInstructeurs { get; set; }
        public DbSet<ASSCampus> ASSCampussen { get; set; }
        public DbSet<ASSVerantwoordelijkheid> ASSVerantwoordelijkheden { get; set; }

        public DbSet<ASSCursist> Cursisten { get; set; } // (3)

        protected override void OnModelCreating(DbModelBuilder modelBuilder) // (4)
        {
            modelBuilder.Entity<TPHKlassikaleCursus>()
                .Map(m => m.Requires("Soort").HasValue("K"));

            modelBuilder.TPHEntity<ZelfstudieCursus>()
                .Map(m => m.Requires("Soort").HasValue("Z"));

            modelBuilder.TPCEntity<KlassikaleCursus>()
                .Map(m => m.MapInheritedProperties()); // (5)

            modelBuilder.Entity<TPCZelfstudieCursus>()
                .Map(m => m.MapInheritedProperties());

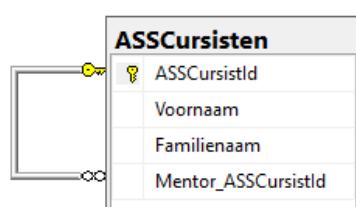
            modelBuilder.Entity<ASSInstructeur>() // (6)
                .HasMany(i => i.Verantwoordelijkheden) // (7)
                .WithMany(v => v.Instructeurs) // (8)
                .Map(c =>
```

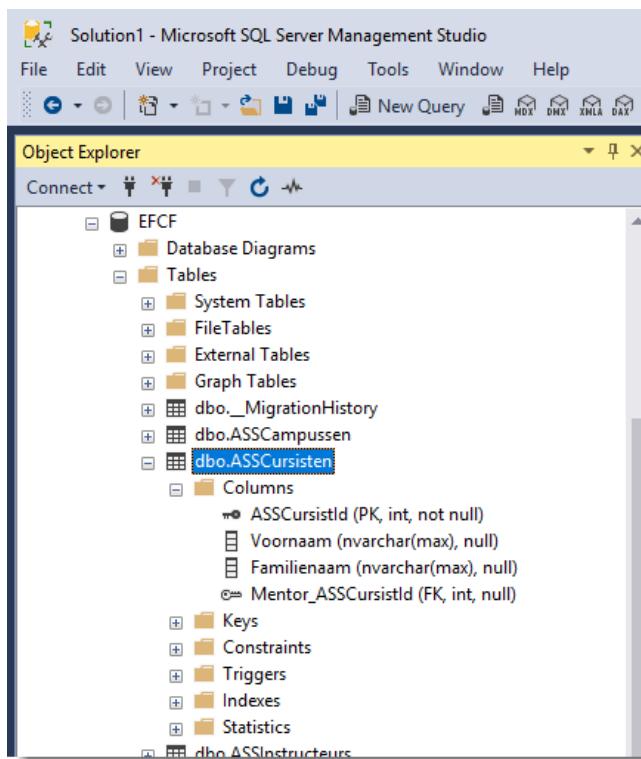
```
        c.ToTable("InstructeursVerantwoordelijkheden") // (9)
        .MapLeftKey("VerantwoordelijkheidID")           // (10)
        .MapRightKey("InstructeurNr"));                  // (11)
    }
}
```

Je wijzigt de code in de using van de Main van Program.cs:

Je kan het programma uitproberen. De database wordt geschrapt en opnieuw aangemaakt.

Je krijgt volgende nieuwe tabel:





Je ziet dat EF zelf een kolomnaam verzonnen heeft die hoort bij de private variabele **Mentor**. Je kan deze kolomnaam zelf kiezen met het attribuut **ForeignKey**.

Je tikt

`[ForeignKey("MentorId")]`

voor de private variabele **Mentor** en je voegt volgende private variabele toe:

`public int? MentorId { get; set; }`

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

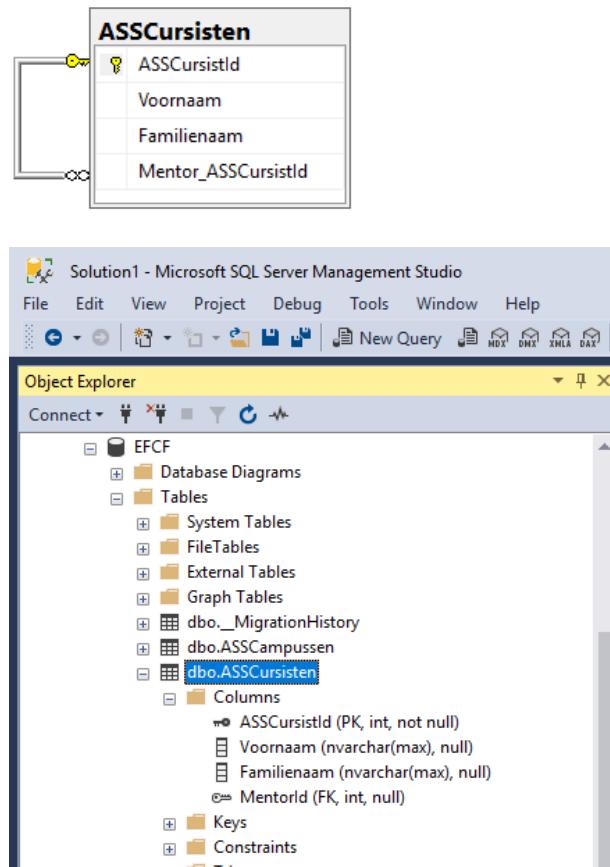
namespace CodeFirstCursus
{
    [Table("ASSCursisten")]
    public class ASSCursist
    {
        [Key]
        public int ASSCursistId { get; set; }
        public string Voornaam { get; set; }
        public string Familienaam { get; set; }

        public virtual ICollection<ASSCursist> Beschermlingen { get; set; } // (1)

        [InverseProperty("Beschermlingen")]
        [ForeignKey("MentorId")]
        public virtual ASSCursist Mentor { get; set; } // (2)

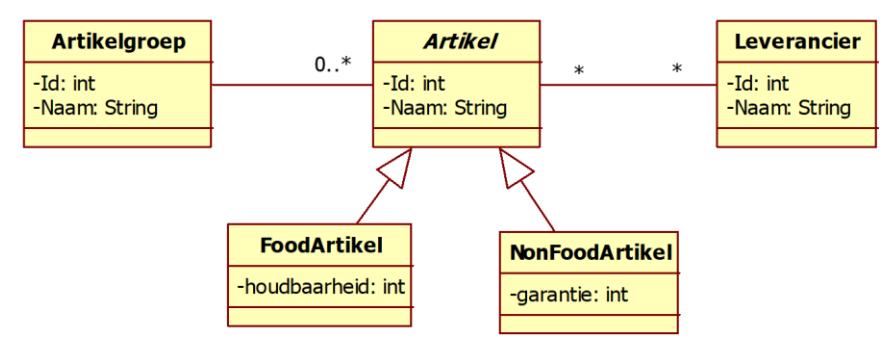
        public int? MentorId { get; set; } // (3)
    }
}
```

Als je nu het programma uitvoert, ziet de table er als volgt uit:



17.11 TAAK 12 : CODE FIRST

Je maakt in een nieuw project volgende classes met code first en een bijbehorende database.



18 WPF

Om deze cursus af te ronden gebruik je EF in een WPF-applicatie.

Je kan daarbij zowel EDMX als Code First gebruiken. Je gebruikt in het voorbeeld Code First. Je kan als voorbeeld eveneens de table **Cursisten** van de database **EFOpleidingen** gebruiken (zonder initializer).

18.1 DE ENTITY CLASSES VOOR DE NIEUWE DATABASE

Je maakt in Visual Studio een WPF-project met de naam **EFinWPF**.

Je downloadt de laatste versie van de EF-library en gebruikt die in het project

- Je klikt met de rechtermuisknop op het project in de **Solution Explorer**
- Je kiest **Manage NuGet Packages**
- Je kiest **Browse**
- Je kiest **EntityFramework**
- Je kiest **Install**

Je voegt een entity class **Cursist** toe :

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EFinWPF
{
    [Table("Cursisten")]
    public class Cursist
    {
        [Key]
        public int CursistNr { get; set; }

        public string Voornaam { get; set; }

        public string Familienaam { get; set; }

        public virtual ICollection<Cursist> Beschermlingen { get; set; }

        [InverseProperty("Beschermlingen")]
        [ForeignKey("MentorNr")]
        public virtual Cursist Mentor { get; set; }

        public int? MentorNr { get; set; }

        public string Naam
        {
            get
            {
                return Voornaam + ' ' + Familienaam;
            }
        }
    }
}
```

18.2 DE DBCONTEXT CLASS

Je voegt een class **EFinWPFContext** toe :

```
using System.Data.Entity;
```

```
namespace EFInWPF
{
    class EFInWPFCContext : DbContext
    {
        public DbSet<Cursist> Cursisten { get; set; }
    }
}
```

18.3 DE CONNECTIONSTRING

Je maakt in `App.config` een **connectionstring** naar de database, onder `</startup>`

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkId=237468 -->
        <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    </configSections>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
    </startup>

    <connectionStrings>
        <add name="EFInWPFCContext"
            providerName="System.Data.SqlClient"
            connectionString=
                "Server=localhost;Database=EFInWPF;Trusted_Connection=true;"/>
        <!-- "Server=localhost;Database=EFopleidingen;Trusted_Connection=true;"/>-->
        <!-- "Server=.\SQLEXPRESS;Database=EFInWPF;Trusted_Connection=true;"/>-->
    </connectionStrings>

    <entityFramework>
        <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
EntityFramework">
            <parameters>
                <parameter value="mssqllocaldb" />
            </parameters>
        </defaultConnectionFactory>
        <providers>
            <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
        </providers>
    </entityFramework>
</configuration>
```

18.4 EEN INSTANTIE VAN DE DBCONTEXT CLASS

Je voegt aan `MainWindow.xaml.cs` een instantie toe van de `EFInWPFCContext` class :

```
private EFInWPFCContext context = new EFInWPFCContext();
```

Je sluit deze `EFInWPFCContext` wanneer de gebruiker het venster sluit:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
```

```

using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

using System.Data.Entity;

namespace EFInWPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private EFInWPFCContext context = new EFInWPFCContext();

        public MainWindow()
        {
            InitializeComponent();
        }

        protected override void OnClosing(System.ComponentModel.CancelEventArgs e)
        {
            context.Dispose();
        }
    }
}

```

18.5 EEN LISTBOX TONEN MET DATA UIT DE DATABASE

Je toont in het venster de namen van **cursisten** die zelf **mentor** zijn.

Je voegt daartoe volgende regels toe aan **MainWindow.xaml**, onder **<Grid>** :

```
<ListBox Name="listMentors" DisplayMemberPath="Naam"/>
```

```

<Window x:Class="EFInWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:EFInWPF"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <ListBox Name="listMentors" DisplayMemberPath="Naam"/>
    </Grid>
</Window>

```

Je maakt de class **EFInWPFIinitializer** om de database van testgegevens te voorzien:

```

using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EFInWPF
{
    class EFInWPFIinitializer : DropCreateDatabaseAlways<EFInWPFCContext>
    {
        protected override void Seed(EFInWPFCContext context)
        {
            context.Cursisten.Add(new Cursist
            {
                CursistNr = 1,
                Voornaam = "Marcel",

```

```
        Familienaam = "Kiekeboe"
    });

    context.Cursisten.Add(new Cursist
    {
        CursistNr = 2,
        Voornaam = "Charlotte",
        Familienaam = "Kiekeboe"
    });

    context.Cursisten.Add(new Cursist
    {
        CursistNr = 3,
        Voornaam = "Fanny",
        Familienaam = "Kiekeboe"
    });

    context.Cursisten.Add(new Cursist
    {
        CursistNr = 4,
        Voornaam = "Konstantinopel",
        Familienaam = "Kiekeboe"
    });

    context.Cursisten.Add(new Cursist
    {
        CursistNr = 5,
        Voornaam = "Leon",
        Familienaam = "Van der Neffe"
    });

    context.Cursisten.Add(new Cursist
    {
        CursistNr = 6,
        Voornaam = "Firmin",
        Familienaam = "van de Kasseien"
    });

    context.SaveChanges();
}
}
```

Voeg een methode OnStartup toe in de `App.xaml.cs` :

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;

using System.Data.Entity;

namespace EFInWPF
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);
            Database.SetInitializer(new EFInWPFIInitializer());
        }
    }
}
```

Je voegt volgende method toe aan de class **MainWindow**.

Je zoekt in die code de **cursisten** die **mentor** zijn.

Je maakt van die **cursisten** een **List** en gebruikt die als bron voor de **ListBox**

```
private void VulListMentors()
{
    listMentors.ItemsSource = (from cursist in context.Cursisten
                                where cursist.Beschermlingen.Count() != 0
                                orderby cursist.Voornaam, cursist.Familienaam
                                select cursist).ToList();
}
```

Je roept die method op in de constructor van de class **MainWindow**.

```
public MainWindow()
{
    InitializeComponent();
    VulListMentors();
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EFInWPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private EFInWPFCContext context = new EFInWPFCContext();

        public MainWindow()
        {
            InitializeComponent();
            VulListMentors();
        }

        protected override void OnClosing(System.ComponentModel.CancelEventArgs e)
        {
            context.Dispose();
        }

        private void VulListMentors()
        {
            listMentors.ItemsSource =
                (from cursist in context.Cursisten
                 where cursist.Beschermlingen.Count() != 0
                 orderby cursist.Voornaam, cursist.Familienaam
                 select cursist).ToList();
        }
    }
}
```

Je kan de applicatie uitproberen.

	CursistNr	Voornaam	Familienaam	MentorNr
1	1	Charlotte	Kiekeboe	NULL
2	2	Marcel	Kiekeboe	1
3	3	Fanny	Kiekeboe	2
4	4	Konstantinopel	Kiekeboe	2
5	5	Leon	Van der Neffe	NULL
6	6	Firmin	van de Kasseien	NULL

18.6 EEN LISTBOX MET GERELATEERDE DATA TONEN

Als de gebruiker in de **ListBox** een **mentor** selecteert, toon je in een tweede **ListBox** de **beschermelingen** van die **mentor**.

Je vervangt in **MainWindow.xaml** de regel `<ListBox .../>` door :

```

<Window x:Class="EFInWPF.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:EFInWPF"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
    <Grid>
        <!-- <ListBox Name="listMentors" DisplayMemberPath="Naam"/> -->

        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="5"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <ListBox      Name="listMentors"
                      DisplayMemberPath="Naam"
                      Grid.Row="0"
                      SelectionChanged="ListMentorsSelectionChanged"/>
    

```

```

<GridSplitter Grid.Row="1" HorizontalAlignment="Stretch"/>

<ListBox Name="listBeschermelingen"
DisplayMemberPath="Naam"
Grid.Row="2"/>
</Grid>
</Window>

```

Je voegt volgende method toe aan de class **MainWindow**.

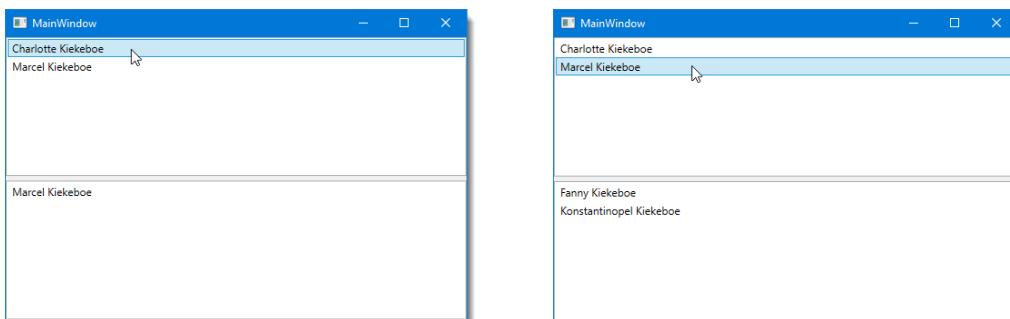
Als de gebruiker in de **ListBox** met **mentors** een item selecteert, vraag je de **beschermelingen** van die **mentor** en je gebruikt die **beschermelingen** als bron voor de tweede **ListBox**:

```

private void ListMentorsSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (listMentors.SelectedItem != null)
    {
        var mentor = (Cursist)listMentors.SelectedItem;
        listBeschermelingen.ItemsSource = mentor.Beschermelingen;
    }
}

```

Je kan de applicatie uitproberen.



18.7 EEN DATAGRID TONEN MET DATA UIT DE DATABASE

Je zal de **beschermelingen** tonen in een **DataGridView**.

Je vervangt in **MainWindow.xaml** de regel **<ListBox Name="listBeschermelingen" ... />** door:

```

<Window x:Class="EFInWPF.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:EFInWPF"
mc:Ignorable="d"
Title="MainWindow" Height="350" Width="525">
<Grid>
    <!--<ListBox Name="listMentors" DisplayMemberPath="Naam"/><!--&gt;
    &lt;Grid.RowDefinitions&gt;
        &lt;RowDefinition Height="*"/&gt;
        &lt;RowDefinition Height="5"/&gt;
        &lt;RowDefinition Height="*"/&gt;
    &lt;/Grid.RowDefinitions&gt;
</pre>

```

```

<ListBox Name="listMentors"
         DisplayMemberPath="Naam"
         Grid.Row="0"
         SelectionChanged="ListMentorsSelectionChanged"/>

<GridSplitter Grid.Row="1" HorizontalAlignment="Stretch"/>

<!-- <ListBox Name="listBeschermelingen"
             DisplayMemberPath="Naam"
             Grid.Row="2"/>
-->

<DataGrid Name="gridBeschermelingen"
          Grid.Row="2"
          IsReadOnly="True"
          AutoGenerateColumns="False">
    <DataGrid.Columns>
        <DataGridTextColumn Binding="{Binding Voornaam}"
                            Header="Voornaam"
                            Width="*"/>
        <DataGridTextColumn Binding="{Binding Familienaam}"
                            Header="Familienaam"
                            Width="*"/>
    </DataGrid.Columns>
</DataGrid>
</Grid>
</Window>

```

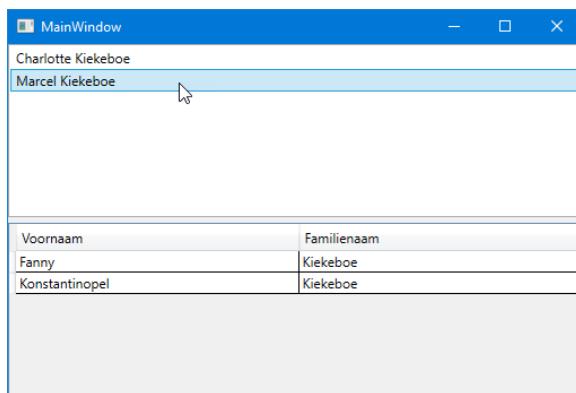
Je vervangt in de method `ListMentorsSelectionChanged` van de class `MainWindow` `listMentors` door `gridBeschermelingen`

```

private void ListMentorsSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (listMentors.SelectedItem != null)
    {
        var mentor = (Cursist)listMentors.SelectedItem;
//        listBeschermelingen.ItemsSource = mentor.Beschermelingen;
        gridBeschermelingen.ItemsSource = mentor.Beschermelingen;
    }
}

```

Je kan de applicatie uitproberen.



18.8 DATA WIJZIGEN

De gebruiker zal de voornaam en/of familienaam van beschermelingen kunnen corrigeren. Hij drukt daarna op een knop **Opslaan** in het menu om deze correcties in de database op te slaan.

Je doet volgende wijzigingen in `MainWindow.xaml`

- Je verwijdert in de regel `<DataGrid ... IsReadOnly="True"`
 - Je tikt voor `<Grid>` volgende regels:
`<DockPanel>`
 `<Menu DockPanel.Dock="Top">`
 `<MenuItem Header="Opslaan" Click="Opslaan" />`
 `</Menu>`
 - Je tikt na `</Grid>` volgende regel:
`</DockPanel>`

```
<Window x:Class="EFInWPF.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:EFInWPF"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
    <DockPanel>
        <Menu DockPanel.Dock="Top">
            <MenuItem Header="Opslaan" Click="Opslaan" />
        </Menu>

        <Grid>
            <!--<ListBox Name="listMentors" DisplayMemberPath="Naam"/><!--

            &lt;Grid.RowDefinitions&gt;
                &lt;RowDefinition Height="*"/&gt;
                &lt;RowDefinition Height="5"/&gt;
                &lt;RowDefinition Height="*"/&gt;
            &lt;/Grid.RowDefinitions&gt;

            &lt;ListBox Name="listMentors"
                    DisplayMemberPath="Naam"
                    Grid.Row="0"
                    SelectionChanged="ListMentorsSelectionChanged"/&gt;

            &lt;GridSplitter Grid.Row="1" HorizontalAlignment="Stretch"/&gt;

            &lt;!--      &lt;ListBox Name="listBeschermerlingen"
                    DisplayMemberPath="Naam"
                    Grid.Row="2"/&gt;
            --&gt;

            &lt;DataGrid Name="gridBeschermerlingen"
                      Grid.Row="2"
                      AutoGenerateColumns="False"&gt;
                &lt;!--      IsReadOnly="True" --&gt;
                &lt;DataGrid.Columns&gt;
                    &lt;DataGridTextColumn Binding="{Binding Voornaam"
                                         Header="Voornaam"
                                         Width="*"/&gt;
                    &lt;DataGridTextColumn Binding="{Binding Familienaam"
                                         Header="Familienaam"
                                         Width="*"/&gt;
                &lt;/DataGrid.Columns&gt;
            &lt;/DataGrid&gt;
        &lt;/Grid&gt;
    &lt;/DockPanel&gt;
&lt;/Window&gt;</pre>
```

Je voegt volgende method toe aan de class **MainWindow**.

Deze method wordt opgeroepen als de gebruiker in het menu **Opslaan** kiest :

```
private void Opslaan(object sender, RoutedEventArgs e)
{
    context.SaveChanges();
```

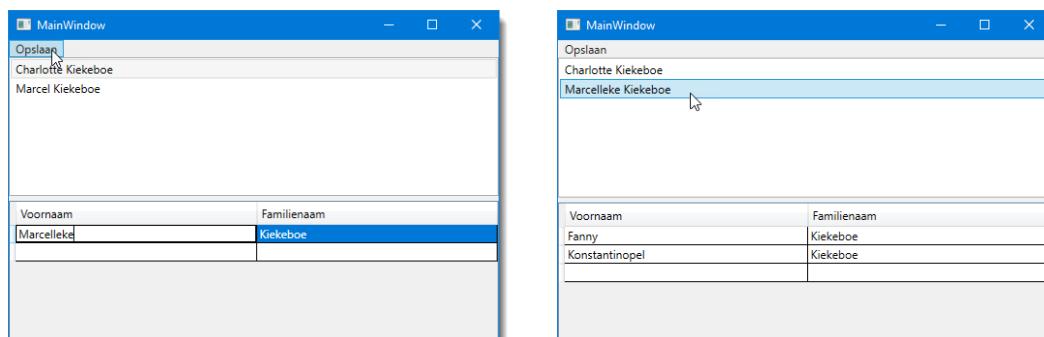
```
// Opdat de wijzigingen die je in de grid deed ook zichtbaar zijn in de list:  
VulListMentors();  
}
```

Je wijzigt in de class **Cursist ICollection** naar **ObservableCollection**.

- Het type **ObservableCollection** stelt, zoals **ICollection** een verzameling voor.
- Je kan data in een **ICollection** tonen in WPF-controls, maar niet wijzigen.
- Je kan data in een **ObservableCollection** in WPF-controls tonen én wijzigen.

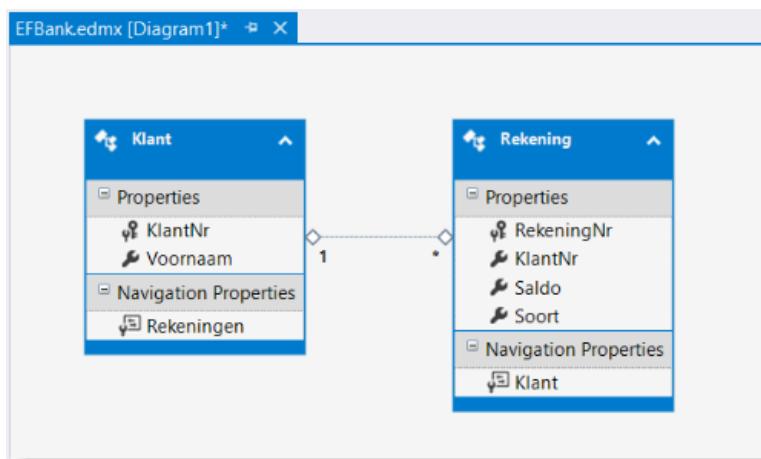
```
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
  
using System.Collections.ObjectModel;  
  
namespace EFInWPF  
{  
    [Table("Cursisten")]  
    public class Cursist  
    {  
        [Key]  
        public int CursistNr { get; set; }  
  
        public string Voornaam { get; set; }  
  
        public string Familienaam { get; set; }  
  
        //  
        public virtual ICollection<Cursist> Beschermeringen { get; set; }  
        public virtual ObservableCollection<Cursist> Beschermeringen { get; set; }  
  
        [InverseProperty("Beschermeringen")]  
        [ForeignKey("MentorNr")]  
        public virtual Cursist Mentor { get; set; }  
  
        public int? MentorNr { get; set; }  
  
        public string Naam  
        {  
            get  
            {  
                return Voornaam + ' ' + Familienaam;  
            }  
        }  
    }  
}
```

Je kan de applicatie uitproberen.



19 VOORBEELDOPLOSSINGEN

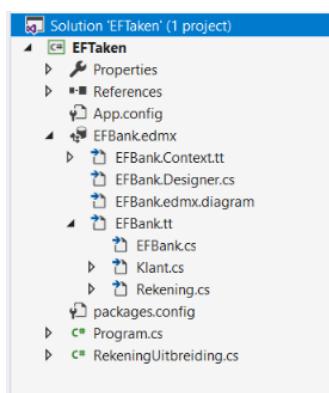
19.1 TAAK 01 : BANK MAKEN



- De class **Klanten** is hernoemd naar **Klant**
- De class **Rekeningen** is hernoemd naar **Rekening**
- De navigation property **Klanten** van de class **Rekening** is hernoemd naar **Klant**.

De method **Storten** van de class **Rekening** in een apart bestand **RekeningUitbreiding.cs**

```
namespace EFTaken
{
    public partial class Rekening
    {
        public void Storten(decimal bedrag)
        {
            this.Saldo += bedrag;
        }
    }
}
```



19.2 TAAK 02 : KLANTEN EN HUN REKENINGEN

```
.....
using System.Linq;
.....
```

```
using (var entities = new EFBankEntities())
{
    var query = from klant in entities.Klanten.Include("Rekeningen")
                orderby klant.Voornaam
                select klant;

    foreach (var klant in query)
    {
        Console.WriteLine(klant.Voornaam);
        var totaleSaldo = Decimal.Zero;

        foreach (var rekening in klant.Rekeningen)
        {
            Console.WriteLine("{0}: {1}", rekening.RekeningNr, rekening.Saldo);
            totaleSaldo += rekening.Saldo;
        }

        Console.WriteLine("Totaal: {0}", totaleSaldo);
        Console.WriteLine();
    }

    Console.ReadKey();
}
```

```
1 - Taak 2 : Klanten en hun rekeningen
1
Bart
Totaal: 0

Homer
345-6789012-12: 500,00
Totaal: 500,00

Lisa
Totaal: 0

Maggie
Totaal: 0

Marge
123-4567890-02: 1000,00
234-5678901-69: 2000,00
Totaal: 3000,00
```

19.3 TAAK 03 : ZICHTREKENING TOEVOEGEN

```
using (var entities = new EFBankEntities())
{
    var query = from klant in entities.Klanten orderby klant.Voornaam select klant;

    foreach (var klant in query)
    {
        Console.WriteLine("{0}: {1}", klant.KlantNr, klant.Voornaam);
    }
    try
    {
        Console.Write("KlantNr:");
        var klantNr = int.Parse(Console.ReadLine());
        var klant = entities.Klanten.Find(klantNr);

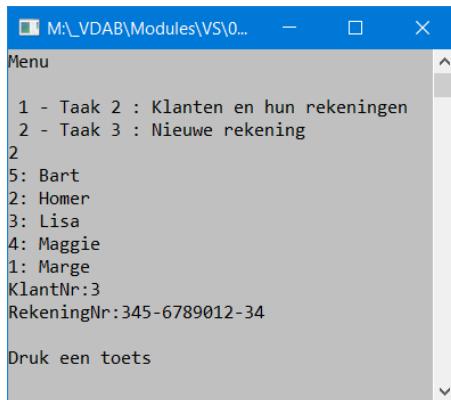
        if (klant == null)
        {
            Console.WriteLine("Klant niet gevonden");
        }
        else
        {
            Console.Write("RekeningNr:");
        }
    }
}
```

```

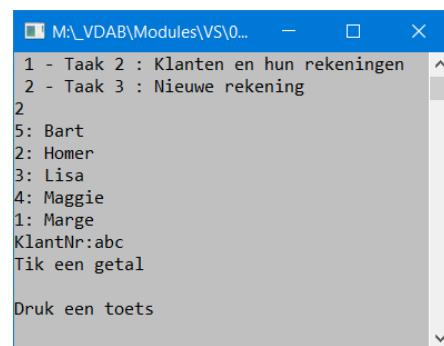
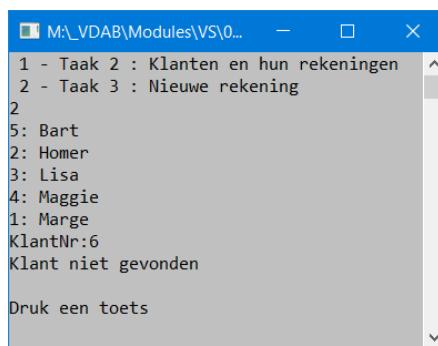
        var rekeningNr = Console.ReadLine();
        var rekening = new Rekening { RekeningNr = rekeningNr, Soort = "Z" };

        klant.Rekeningen.Add(rekening);
        entities.SaveChanges();
    }
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}
}

```



	RekeningNr	KlantNr	Saldo	Soort
	123-4567890-02	1	1000,00	Z
	234-5678901-69	1	2000,00	S
	345-6789012-12	2	500,00	S
▶	345-6789012-34	3	0,00	Z
*	NULL	NULL	NULL	NULL



19.4 TAAK 04 : STORTEN

```

using (var entities = new EFBankEntities())
{
    Console.Write("RekeningNr:");
    var rekeningNr = Console.ReadLine();
    var rekening = entities.Rekeningen.Find(rekeningNr);

    if (rekening == null)
    {
        Console.WriteLine("Rekening niet gevonden");
    }
}

```

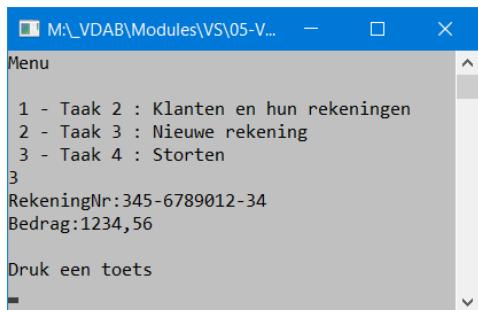
```

else
    try
    {
        Console.WriteLine("Bedrag:");
        var bedrag = decimal.Parse(Console.ReadLine());

        if (bedrag <= Decimal.Zero)
        {
            Console.WriteLine("Tik een positief bedrag");
        }
        else
        {
            rekening.Storten(bedrag);
            entities.SaveChanges();
        }
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een bedrag");
    }
}

```

	RekeningNr	KlantNr	Saldo	Soort
	123-4567890-02	1	1000,00	Z
	234-5678901-69	1	2000,00	S
	345-6789012-12	2	500,00	S
▶	345-6789012-34	3	0,00	Z
*	NULL	NULL	NULL	NULL



	RekeningNr	KlantNr	Saldo	Soort
	123-4567890-02	1	1000,00	Z
	234-5678901-69	1	2000,00	S
	345-6789012-12	2	500,00	S
▶	345-6789012-34	3	1234,56	Z
*	NULL	NULL	NULL	NULL

19.5 TAAK 05 : KLANT VERWIJDEREN

```

try
{
    Console.WriteLine("KlantNr:");
    var klantNr = int.Parse(Console.ReadLine());

    using (var entities = new EFBankEntities())
    {
        var klant = entities.Klanten.Find(klantNr);

        if (klant == null)
        {
            Console.WriteLine("Klant niet gevonden");
        }
        else
        {

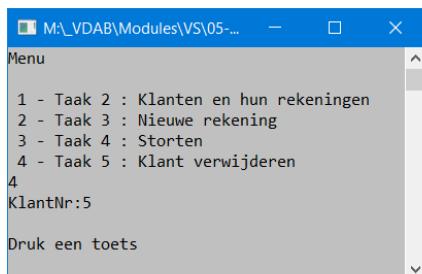
```

```

        if (klant.Rekeningen.Count != 0)
        {
            Console.WriteLine("Klant heeft nog rekeningen");
        }
        else
        {
            entities.Klanten.Remove(klant);
            entities.SaveChanges();
        }
    }
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}

```

	KlantNr	Voornaam
1	Marge	
2	Homer	
3	Lisa	
4	Maggie	
5	Bart	
*	NULL	NULL



	KlantNr	Voornaam
1	Marge	
2	Homer	
3	Lisa	
4	Maggie	
*	NULL	NULL

19.6 TAAK 06 : OVERSCHRIJVEN

Class [SaldoOntoereikendException.cs](#)

```

using System;
namespace EFTaken
{
    class SaldoOntoereikendException: Exception
    {
    }
}

```

Een extra method in de partial class [Rekening \(RekeningUitbreiding.cs\)](#)

```

namespace EFTaken
{
    public partial class Rekening

```

```
{  
    public void Storten(decimal bedrag)  
    {  
        this.Saldo += bedrag;  
    }  
  
    public void Overschrijven(Rekening naarRekening, decimal bedrag)  
    {  
        if (this.Saldo < bedrag)  
        {  
            throw new SaldoOntoereikendException();  
        }  
        else  
        {  
            this.Saldo -= bedrag;  
            naarRekening.Saldo += bedrag;  
        }  
    }  
}
```

Program.cs

```
.....  
using System.Transactions;  
.....  
  
Console.Write("RekeningNr. van rekening:");  
var vanRekeningNr = Console.ReadLine();  
  
Console.Write("RekeningNr. naar rekening:");  
var naarRekeningNr = Console.ReadLine();  
  
try  
{  
    Console.Write("Bedrag:");  
    var bedrag = decimal.Parse(Console.ReadLine());  
  
    if (bedrag <= decimal.Zero)  
        Console.WriteLine("Tik een positief bedrag");  
    else  
    {  
        var transactionOptions = new TransactionOptions  
        {  
            IsolationLevel = IsolationLevel.RePEATABLEREAD  
        };  
  
        using (var transactionScope = new TransactionScope(  
            TransactionScopeOption.REQUIRED, transactionOptions))  
        {  
            using (var entities = new EFBankEntities())  
            {  
                var vanRekening = entities.Rekeningen.Find(vanRekeningNr);  
  
                if (vanRekening == null)  
                    Console.WriteLine("Van rekening niet gevonden");  
                else  
                {  
                    var naarRekening = entities.Rekeningen.Find(naarRekeningNr);  
  
                    if (naarRekening == null)  
                        Console.WriteLine("Naar rekening niet gevonden");  
                    else  
                        try  
                        {  
                            vanRekening.Overschrijven(naarRekening,  
                                bedrag);  
                            entities.SaveChanges();  
                            transactionScope.Complete();  
                        }  
                }  
            }  
        }  
    }  
}
```

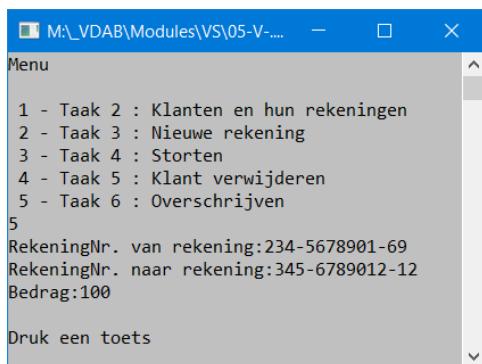
```

        catch (SaldoOntoereikendException)
        {
            Console.WriteLine("Saldo ontoereikend");
        }
    }
}
catch (FormatException)
{
    Console.WriteLine("Tik een bedrag");
}

```

(Vergeet niet: Add Reference..., Assemblies, Framework, **System.Transactions** aanvinken)

	RekeningNr	KlantNr	Saldo	Soort
	123-4567890-02	1	1000,00	Z
▶	234-5678901-69	1	2000,00	S
	345-6789012-12	2	500,00	S
*	345-6789012-34	3	1234,56	Z
	NULL	NULL	NULL	NULL



	RekeningNr	KlantNr	Saldo	Soort
	123-4567890-02	1	1000,00	Z
▶	234-5678901-69	1	1900,00	S
	345-6789012-12	2	600,00	S
*	345-6789012-34	3	1234,56	Z
	NULL	NULL	NULL	NULL

19.7 TAAK 07 : KLANT WIJZIGEN

Aanpassingen aan het **edm**

Je wijzigt de property **Concurrency Mode** van de property **Voornaam** van de class **Klant** naar **Fixed**.

Opmerking:

Je hoeft dit niet te doen voor de property **KlantNr**.

De bijbehorende kolom is een **autonumber**-kolom en kan dus niet gewijzigd worden bij het wijzigen van een record.

```

    ...
using System.Data.Entity.Infrastructure;

```

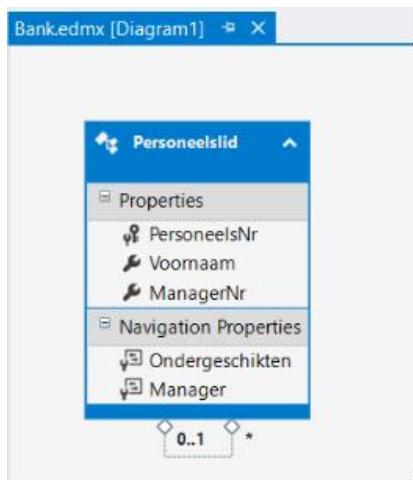
```
Console.WriteLine("KlantNr:");

try
{
    var klantNr = int.Parse(Console.ReadLine());

    using (var entities = new EFBankEntities())
    {
        var klant = entities.Klanten.Find(klantNr);

        if (klant == null)
        {
            Console.WriteLine("Klant niet gevonden");
        }
        else
        {
            Console.WriteLine("Voornaam:");
            klant.Voornaam = Console.ReadLine();
            entities.SaveChanges();
        }
    }
}
catch (DbUpdateConcurrencyException)
{
    Console.WriteLine("Een andere gebruiker wijzigde deze klant");
}
catch (FormatException)
{
    Console.WriteLine("Tik een getal");
}
```

19.8 TAAK 08 : PERSONEEL

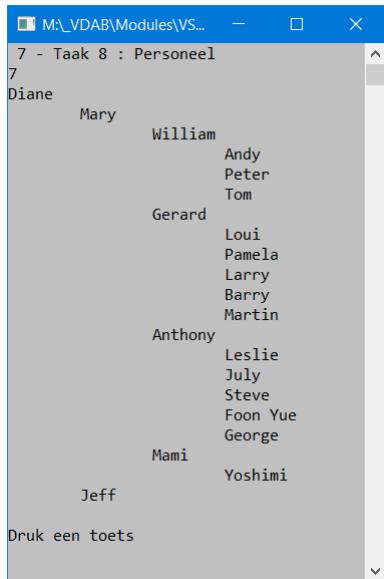


```
using (var entities = new EFBankEntities())
{
    var hoogstenInHierarchie = (from personeelslid in entities.Personeel
                                 where personeelslid.Manager == null
                                 select personeelslid).ToList();

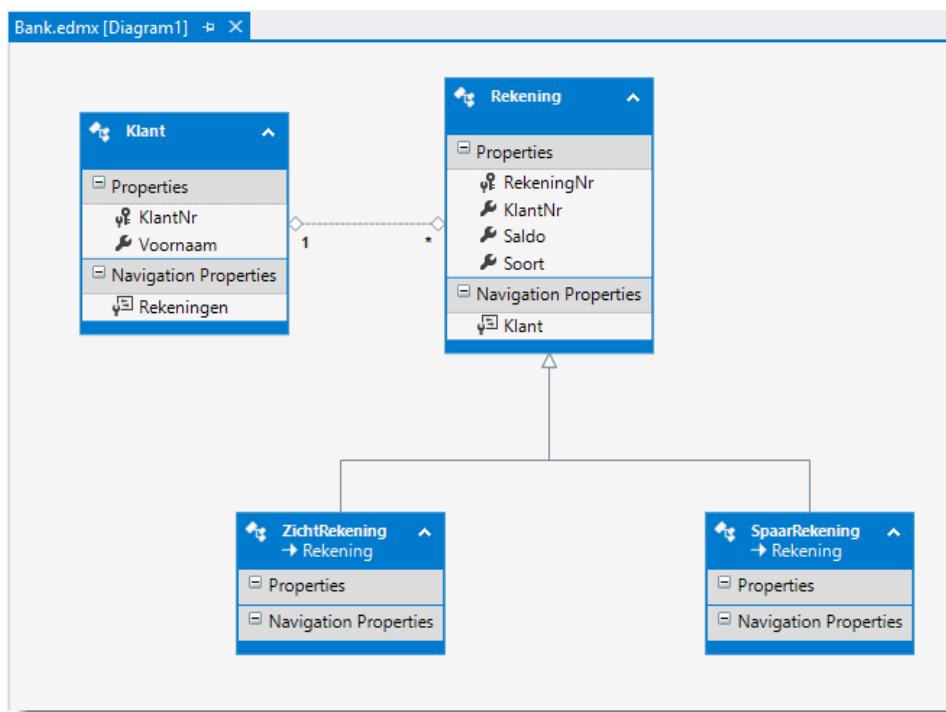
    Afbeelden(hoogstenInHierarchie, 0);
}
```

```
static void Afbeelden(List<Personeelslid> personeel, int insprong)
{
    foreach (var personeelslid in personeel)
```

```
{  
    Console.Write(new String('\t', insprong));  
    Console.WriteLine(personneelslid.Voornaam);  
  
    if (personneelslid.Ondergeschikten.Count != 0)  
    {  
        Afbeelden(personneelslid.Ondergeschikten.ToList(), insprong + 1);  
    }  
}
```

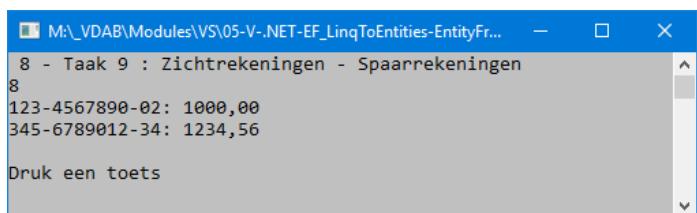


19.9 TAAK 09 : ZICHTREKENINGEN – SPAARREKENINGEN



```
using (var entities = new EFBankEntities())
{
    var query =     from rekening in entities.Rekeningen
                    where rekening is ZichtRekening
                    select rekening;

    foreach (var zichtrekening in query)
    {
        Console.WriteLine("{0}: {1}", zichtrekening.RekeningNr, zichtrekening.Saldo);
    }
}
```



RekeningNr	KlantNr	Saldo	Soort
123-4567890-02	1	1000,00	Z
234-5678901-69	1	1900,00	S
345-6789012-12	2	600,00	S
345-6789012-34	3	1234,56	Z

19.10 TAAK 10 : TOTALE SALDO PER KLANT

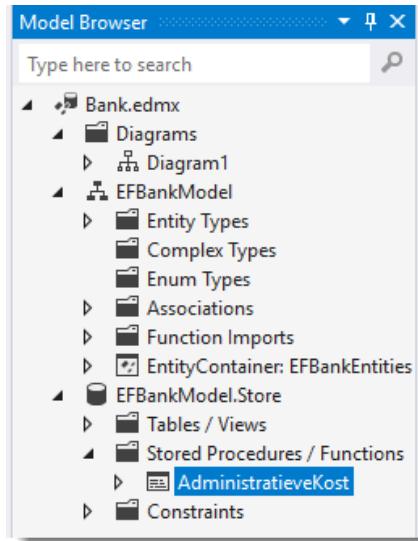
De property **Entity Set Name** is hernoemd naar **TotaleSaldosPerKlant**.

```
using (var entities = new EFBankEntities())
{
    var query =     from totaleSaldoPerKlant
                    in     entities.TotaleSaldosPerKlant
                    orderby totaleSaldoPerKlant.Voornaam
                    select totaleSaldoPerKlant;

    foreach (var totaleSaldoPerKlant in query)
        Console.WriteLine("{0}: {1}", totaleSaldoPerKlant.Voornaam,
                          totaleSaldoPerKlant.TotaleSaldo);
}
```

19.11 TAAK 11: ADMINISTRATIEVE KOST

Bank.edmx



Bank.Context.cs

```
//----------------------------------------------------------------------------  
// <auto-generated>  
//     This code was generated from a template.  
//  
//     Manual changes to this file may cause unexpected behavior in your application.  
//     Manual changes to this file will be overwritten if the code is regenerated.  
// </auto-generated>  
//----------------------------------------------------------------------------  
  
namespace EFTaken  
{  
    using System;  
    using System.Data.Entity;  
    using System.Data.Entity.Infrastructure;  
    using System.Data.Entity.Core.Objects;  
    using System.Linq;  
  
    public partial class EFBankEntities : DbContext  
    {  
        . . . . .  
  
        public virtual int AdministratieveKost(Nullable<decimal> kost)  
        {  
            var kostParameter = kost.HasValue ?  
                new ObjectParameter("Kost", kost) :  
                new ObjectParameter("Kost", typeof(decimal));  
  
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("AdministratieveKost",  
kostParameter);  
        }  
    }  
}
```

Program.cs

```
try  
{  
    Console.Write("Kost:");
```

```

        var kost = decimal.Parse(Console.ReadLine());

        using (var entities = new EFBankEntities())
        {
            Console.WriteLine("{0} rekeningen aangepast",
                entities.AdministratieveKost(kost));
        }
    catch (FormatException)
    {
        Console.WriteLine("Tik een getal");
    }
}

```

	RekeningNr	KlantNr	Saldo	Soort
1	123-4567890-02	1	1000.00	Z
2	234-5678901-69	1	1900.00	S
3	345-6789012-12	2	600.00	S
4	345-6789012-34	3	1234.56	Z

```

M:\_VDAB\Modules\VS\05-V...
10 - Taak 11 : Administratieve kost
10
Kost:55
4 rekeningen aangepast

Druk een toets

```

	RekeningNr	KlantNr	Saldo	Soort
1	123-4567890-02	1	945.00	Z
2	234-5678901-69	1	1845.00	S
3	345-6789012-12	2	545.00	S
4	345-6789012-34	3	1179.56	Z

19.12 TAAK 12 : CODE FIRST

19.12.1 INSTALLEREN EF

19.12.2 ARTIKELGROEP

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstTaak
{
    [Table("Artikelgroepen")]
    public class Artikelgroep
    {
        public int Id { get; set; }
        public string Naam { get; set; }
        public ICollection<Artikel> Artikels { get; set; }
    }
}

```

19.12.3 ARTIKEL

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace CodeFirstTaak
{
    [Table("Artikels")]
    public abstract class Artikel
    {
    }
}

```

```
{  
    public int Id { get; set; }  
    public string Naam { get; set; }  
    public virtual Artikelgroep Artikelgroep { get; set; }  
    public int ArtikelgroepId { get; set; }  
    public ICollection<Leverancier> Leveranciers { get; set; }  
}  
}
```

19.12.4 LEVERANCIER

```
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace CodeFirstTaak  
{  
    [Table("Leveranciers")]  
    public class Leverancier  
    {  
        public int Id { get; set; }  
        public string Naam { get; set; }  
        public ICollection<Artikel> Artikels { get; set; }  
    }  
}
```

19.12.5 FOODARTIKEL

```
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace CodeFirstTaak  
{  
    [Table("FoodArtikels")]  
    public class FoodArtikel : Artikel  
    {  
        public int Houdbaarheid { get; set; }  
    }  
}
```

19.12.6 NONFOODARTIKEL

```
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace CodeFirstTaak  
{  
    [Table("NonFoodArtikels")]  
    public class NonFoodArtikel : Artikel  
    {  
        public int Garantie { get; set; }  
    }  
}
```

19.12.7 CONTEXT CLASS

```
using System.Data.Entity;  
  
namespace CodeFirstTaak  
{  
    class EFTaakContext : DbContext  
    {  
        public DbSet<Artikelgroep> Artikelgroepen { get; set; }  
        public DbSet<Artikel> Artikels { get; set; }  
        public DbSet<Leverancier> Leveranciers { get; set; }  
    }  
}
```

19.12.8 APP.CONFIG

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkId=237468 -->
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0,
      Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
  </startup>

  <connectionStrings>
    <add name="EFTaakContext"
      providerName="System.Data.SqlClient"
      connectionString="Server=localhost;Database=EFTaak;Trusted_Connection=true;" />
    <!-- connectionString="Server=.\SQLEXPRESS;Database=EFTaak;Trusted_Connection=true;" /> -->
  </connectionStrings>

  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
      EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
      type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

19.12.9 PROGRAM

```
using System;
using System.Data.Entity;

namespace CodeFirstTaak
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Data.Entity.Database.SetInitializer(
                new DropCreateDatabaseIfModelChanges<EFTaakContext>());

            using (var context = new EFTaakContext())
            {
                var leverancier = new Leverancier
                {
                    Id = 1,
                    Naam = "Marcel Kiekeboe"
                };

                context.Leveranciers.Add(leverancier);
                context.SaveChanges();

                Console.WriteLine("EFTaak Done");
                Console.ReadKey();
            }
        }
    }
}
```

19.12.10 RESULTAAT IN SQL SERVER MANAGEMENT STUDIO

Table Leveranciers

	Id	Naam
1	1	Marcel Kiekeboe

Microsoft SQL Server Management Studio

File Edit View Debug Tools Window Help

New Query MDX DMX XML

Object Explorer

Connect

FileTables

External Tables

Graph Tables

dbo._MigrationHistory

dbo.Artikelgroepen

Columns

Id (PK, int, not null)

Naam (nvarchar(max), null)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Artikels

Columns

Id (PK, int, not null)

Naam (nvarchar(max), null)

ArtikelgroepId (FK, int, not null)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.FoodArtikels

Columns

Id (PK, FK, int, not null)

Houdbaarheid (int, not null)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.LeverancierArtikels

Columns

Leverancier_Id (PK, FK, int, not null)

Artikel_Id (PK, FK, int, not null)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Leveranciers

Columns

Id (PK, int, not null)

Naam (nvarchar(max), null)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.NonFoodArtikels

Columns

Id (PK, FK, int, not null)

Garantie (int, not null)

Keys

Constraints

Triggers

20 COLOFON

Domeinexpertisemanager: Jean Smits
Moduleverantwoordelijke: Hans Desmet
Medewerkers: Hans Desmet
Johan Vandaele
Versie: 01-mei-2018
Nummer dotatielijst: