Structureren van gegevens van het doelsysteem (I)

Object Modellering en Object Relational Mapping

Toon Loonen

Steeds vaker wordt de gegevensstructuur niet meer als een relationeel (Entity Relationship of ER) model vastgelegd of aangeleverd, maar in de vorm van een object- of klassediagram.

In een serie artikelen willen we ingaan op het object- of klassediagram, de overeenkomsten en verschillen met het ER model aanduiden en kijken hoe dit fysiek geïmplementeerd kan worden. Immers, in de praktijk zullen de gegevens meestal toch weer in een relationele database worden opgeslagen.

Object of klasse

In de OO (Object Oriented) terminologie is een object een ding in de werkelijkheid met kennis en gedrag, zoals een (specifieke) klant, een artikel of een order. Een klasse is de representatie van alle objecten van een soort, zoals 'KLANTEN', 'ARTIKELEN' of 'ORDERS' in het verderop te bespreken voorbeeld. En een object is dus weer een instantie van een klasse. Een klasse is voor het kennisdeel vergelijkbaar met een entiteit in ER terminologie en een object is vergelijkbaar met een voorkomen van deze entiteit of een record in een tabel. Hierbij beperken we ons bij het beschouwen van objecten en klassen alleen tot de gegevens (het kennisdeel). In OO is elk onderdeel van het systeem een object, dus ook schermen of knoppen op schermen enzovoort. Daarnaast bevat een klasse naast attributen ook gedrag ofwel operaties. Dit kennen we niet in het ER model en we zullen er in dit artikel ook niet op ingaan. Verder wordt hierna in het algemeen gesproken over het klassediagram, niet over het objectdiagram.

OO of UML

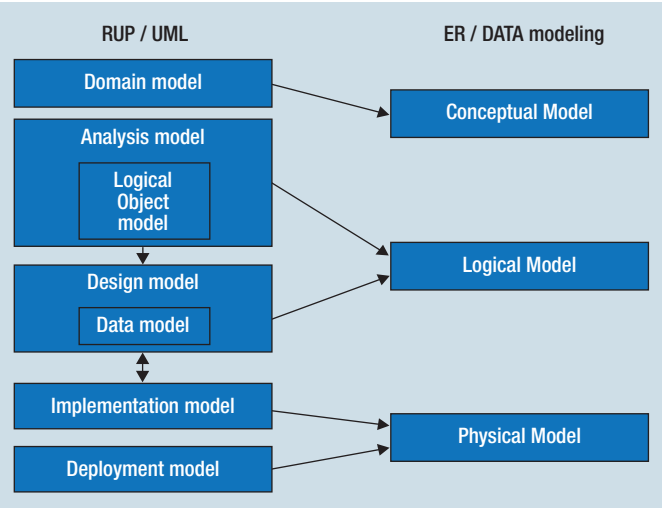
OO staat voor 'Object Oriented' en is een concept gebaseerd op een paradigma [Ref 22]. UML of Unified Modeling Language is de modelleertaal (met diagrammen enzovoort) die algemeen wordt gebruikt voor het tekenen van klasse- of objectdiagrammen. UML bevat een groot aantal modellen of diagramtechnieken. Voor een beschrijving van deze modellen; zie [Ref 1, 2, 23]. Hier zullen we ons weer beperken tot het klassediagram. De hierna gebruikte teken- en notatiewijze volgt de UML standaard.

RUP

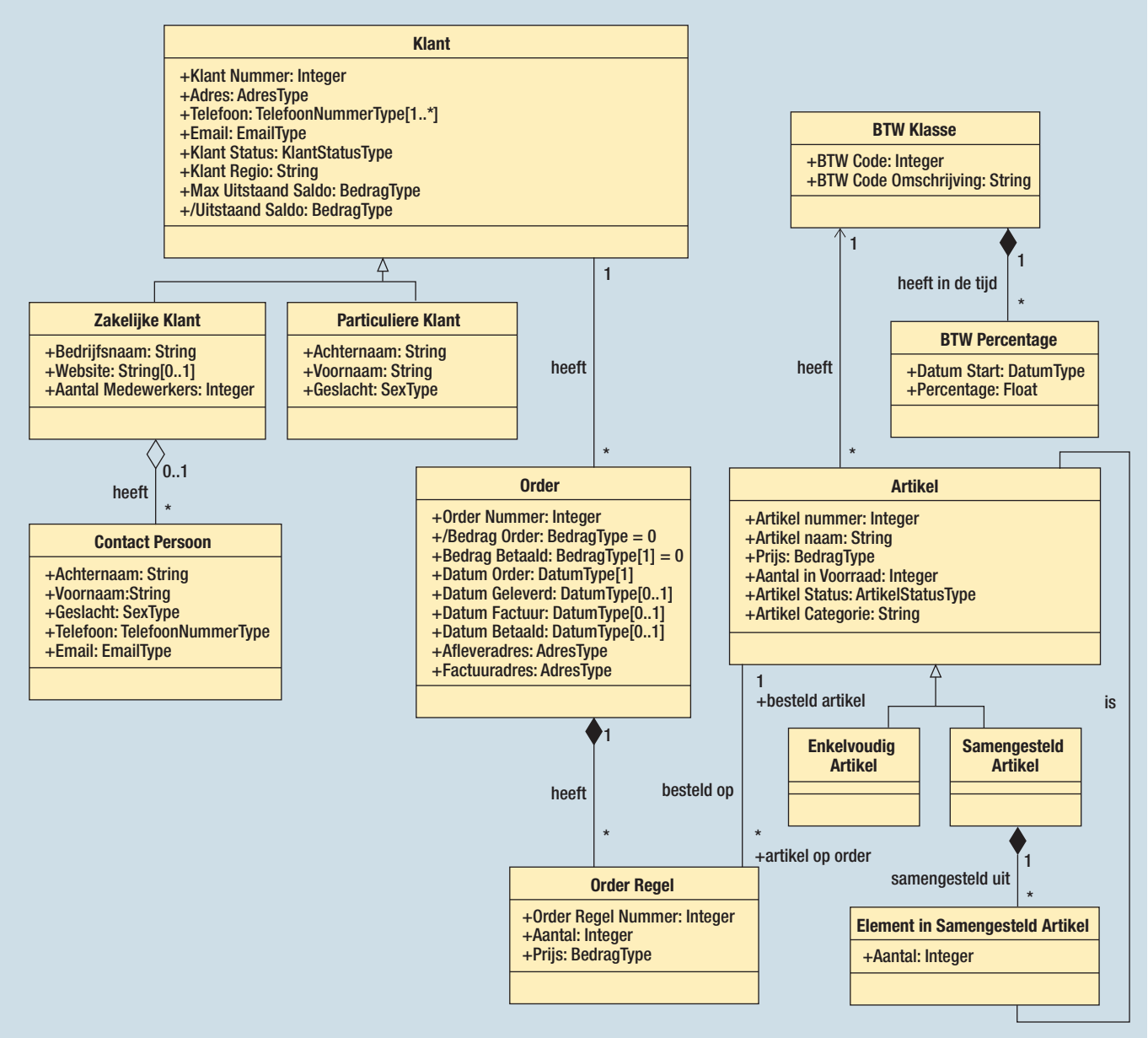
Rational Unified Process (RUP, zie [Ref 3]) is momenteel het meest algemeen gebruikte (iteratief) softwareontwikkelingsproces. Alhoewel RUP zich niet uitspreekt voor het gebruik van een specifieke modelleertechniek wordt het toch algemeen gebruikt in combinatie met UML diagramtechnieken en OO modellering. In RUP heeft het klassediagram niet de centrale plaats en aandacht die het gegevensmodel voorheen in de systeemontwikkeling had. Het klassediagram wordt er opgesteld aan de hand van de Use Case (functionele) beschrijvingen.
Wij geven er de voorkeur aan om het klassediagram eerst op te stellen inclusief een (al of niet gedetailleerde) beschrijving van de items. Dit kan aan de hand van een Enterprise model, een Cannonical model, een vorig systeem of via gegevensstromen, formulieren en andere requirements. Bij het opstellen van de Use Cases kan dan gebruik gemaakt worden van de centrale definities van de gegevens in het klassediagram (dus hergebruik en consistentie). Tevens wordt dan het klassediagram geverifieerd en gecompleteerd aan de hand van de nieuwe inzichten die bij het opstellen van de Use Cases worden opgedaan.
RUP hanteert voor de verschillende stadia in de verfijning van het model ook een andere terminologie dan bij een ER modellering gewoon is. In afbeelding 1 staat een overzicht van de modellen en hun relaties.

Modelleringstools

Er is een groot aantal tools beschikbaar voor het opstellen en tekenen van klassediagrammen (voor een overzicht van deze



Afbeelding 1: Modellen in RUP en UML versus traditioneel ER modellering.



Afbeelding 2: Klassediagram.

tools voor zowel ER als OO/UML modellen, zie [Ref 4]):

- ER: het relationeel model in dit artikel is getekend met Power Designer van Sybase [Ref 7];
- UML: het klassediagram van dit artikel is gemaakt met STARUML [Ref 6];
- UML: Enterprise Architect [Ref 8];
- UML: RSA (Rational Software Architect) van IBM [Ref 21].

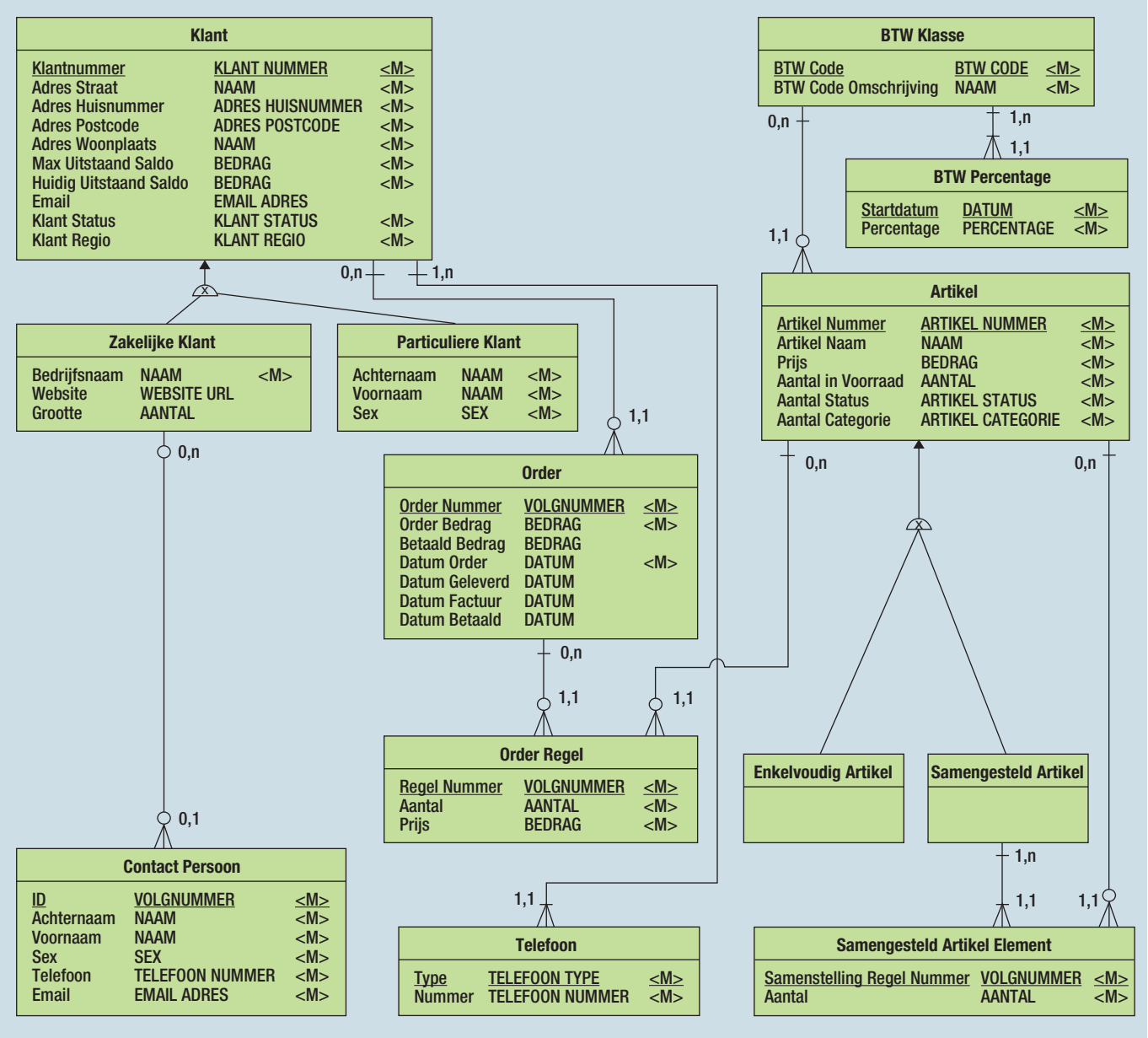
Voorbeeld UML klassediagram en ER model

Afbeelding 2 geeft een voorbeeld van een UML klassediagram en afbeelding 3 toont dezelfde functionaliteit in een (logisch) relationeel model. (N.B.: in de ORDER zijn de twee adressen weggelaten om het model overzichtelijk te houden.)

In beide modellen zien we de volgende informatie gemodelleerd:

- artikelen met een onderverdeling tussen enkelvoudige en samengestelde artikelen (bijvoorbeeld een computer met toetsenbord, muis en scherm);
- elk artikel heeft een BTW-code met een bijbehorend BTW-percentage; dit percentage wordt historisch bijgehouden;
- klanten met een onderverdeling in (de subtypes) zakelijke en particuliere klanten;
- zakelijke klanten hebben 0, 1 of meer contactpersonen;
- een klant is bereikbaar via 1 of meer telefoonnummers (thuis, mobiel, fax enzovoort);
- een klant heeft 0, 1 of meer orders waarop per order 0, 1 of meer artikelen besteld kunnen worden.

N.B.: het doel van deze modellen is om als voorbeeld gebruikt te worden voor dit artikel waarbij alle hierna te bespreken aspecten ook in het voorbeeld voorkomen. Het pretendeert zeker niet een volledig model voor een orderverwerkingssysteem te zijn.



Afbeelding 3: Relationeel model.

Hierna behandelen we enkele verschillen tussen de twee modelleringtechnieken. In het tweede artikel gaan we verder in op de details van het klassediagram. Daarna behandelen we de implementatie van het diagram in een fysieke (relationele) database.

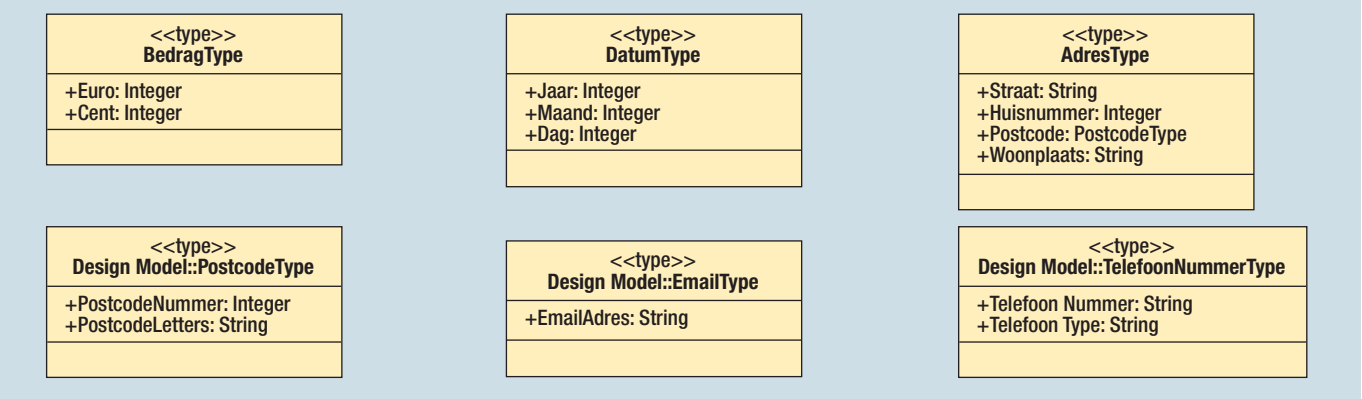
Subtypes in het relationeel model

De wijze waarop subtypes in een relationeel model worden gemodelleerd is erg afhankelijk van het gebruikte tool. Het voorbeeld hiervoor komt van Power Designer. Andere tools zullen tussen de entiteit KLANT en de subtypes ZAKELIJKE KLANTEN en PARTICULIERE KLANTEN een één-op-één relatie leggen met de opmerking (restrictie) erbij dat de twee relaties/subtypes elkaar uitsluiten. Bij een UML klassediagram worden de subtypes altijd als hierboven met een pijl aangegeven. Voor een toelichting op subtypes, zie ook [Ref 18].

Samengestelde attributen en datatypes

In het klassediagram in afbeelding 2 is het begrip ADRES als een afzonderlijk datatype gemodelleerd met als attributen straatnaam, huisnummer, postcode en woonplaats (zie ook afbeelding 4). Dit adres (met daarmee automatisch de vier genoemde attributen) kan op diverse plaatsen in het diagram worden opgenomen. Op deze manier wordt het adres altijd en overal eenduidig op dezelfde manier gedefinieerd. In een relationeel model is het niet zo gebruikelijk om een samengesteld attribuut als type te definiëren.

Andere samengestelde datatypes in dit voorbeeld zijn bedrag, datum, telefoon, postcode en e-mailadres. Hiermee kunnen op één plaats de naam, de definitie, de beschrijving en de validaties of constraints van zo'n type worden vastgelegd en kan het op meer plaatsen worden gebruikt zonder telkens deze informatie te

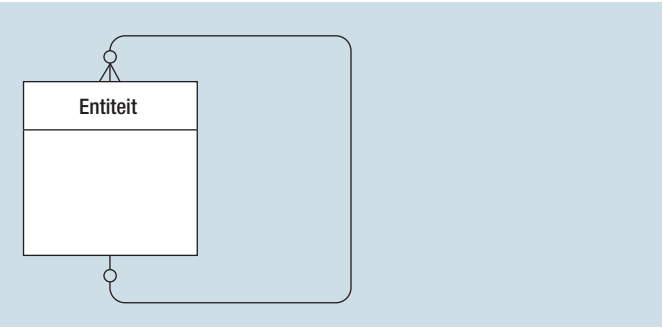


Afbeelding 4: Type Klassen.

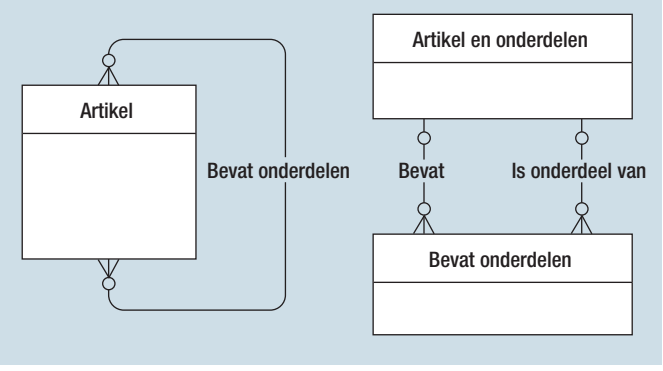
herhalen. De basistypen in UML zijn: in StarUML boolean, integer, real en string; in Enterprice Architect boolean, integer en string. Samengestelde typen, zoals in het voorbeeld een datum of bedrag, moeten uit deze basistypen worden opgezet.

Repeterende groepen

In een genormaliseerd relationeel model horen geen repeterende groepen thuis. In een klassediagram zijn wel repeterende groepen toegestaan, zoals in het voorbeeld voor telefoon.
Meer-op-meer relaties zijn ook toegestaan in een OO model, zoals in het voorbeeld tussen order en artikel. Heeft zo'n meer-op-meer relatie nog eigen attributen, dan kan dat worden gemodelleerd via een associatieklasse, maar meestal zal daarvoor een eigen object worden gedefinieerd.
In het voorbeeld is voor een artikel ook een (meer-op-meer) recursieve relatie opgenomen. Een artikel is samengesteld uit enkele andere artikelen, bijvoorbeeld een computeraanbieding bestaat uit een PC, een scherm, een toetsenbord en een muis. Een fiets wordt samengesteld uit een frame, twee wielen, stuur, zadel enzovoort. De artikelen in de samenstelling komen zelf ook weer als (los bestelbaar) artikel in de artikeltabel terug. Een één-op-meer (dus hiërarchische relatie, bijvoorbeeld afdelingen binnen een bedrijf) recursieve relatie wordt in het relationele model meestal getekend als in afbeelding 5.
Een meer-op-meer recursieve relatie moet vanwege de normaliseringsregels in het ER model opgesplitst worden in een nieuwe tabel en twee één-op-meer relaties als in afbeelding 6 en in het voorbeeldmodel voor het artikel [zie ook Ref 14]. In het voorbeeld van afbeelding 2 is dit gecombineerd met de subtypes voor enkelvoudige en samengestelde artikelen.



Afbeelding 5: Hiërarchische relatie.



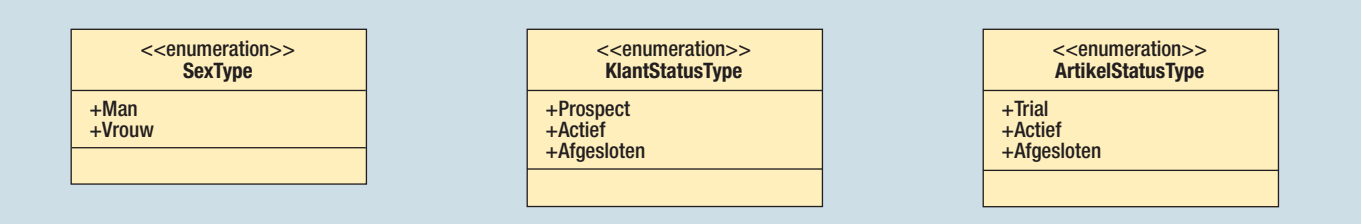
Afbeelding 6: Meer-op-meer recursieve relatie.

Enumeratie

De attribuuttypes Klantstatus en Artikelstatus zijn als enumeratie klassen gedefinieerd. Hierbij kan direct worden aangegeven wat de toegestane waarden zijn. In een relationeel (logisch) model wordt dit meestal als commentaar meegegeven en via rules of constraints in de database vastgelegd. Tevens zal men in het fysiek model en de database er vaak voor kiezen om een korte code te gebruiken (M, F) in plaats van de lange omschrijving (Male, Female), zie afbeelding 7.
De aanduiding <<enumeration>> (en <<type>> in afbeelding 4) betreft een voorbeeld van een 'stereotype'. Stereotypen geven een indeling in modelelementen om gegevensklassen te onderscheiden van interfaces, applicatieklassen, hardwarecomponenten enzovoort.

Navigatie in het klassediagram

In een klassediagram kan een 'geassocieerde klasse', dus een klasse die gerelateerd is aan een andere klasse, beschouwd worden als een attribuut van die andere klasse. Bijvoorbeeld bij een order zijn meteen de bijbehorende klantgegevens beschikbaar of bij een klant zijn de ordergegevens beschikbaar. De associatie tussen KLANT en ORDER is dan tweezijdig navigeerbaar.
De associatie tussen BTW-code en artikel is eenzijdig navigeer-



Afbeelding 7: Enumeratie Klassen.

baar, dat wil zeggen dat bij een artikel wel de bijbehorende BTW-code (en percentage) beschikbaar is maar bij de BTW-code kunnen niet de bijbehorende artikelen opgevraagd worden.
Bij een relationeel model wordt niet over navigeerbaarheid gesproken; via de where clause van een SQL select statement mag bijna alles aan elkaar geknoopt worden, hoe zinvol of zinloos ook. Het ER model is dus niet beperkt tot navigatie via een relatie of foreign key.

Primary en foreign keys

Het relationele model bevat in het logisch model een primary key. In het fysieke model komen daar in de gerelateerde tabellen ook de foreign keys bij. In het klassediagram wordt niet over sleutels gesproken. Hoe een geassocieerd gegeven bij een ander gegeven wordt opgehaald, daar bemoeit het model zich niet mee. Bij een vertaling van een klassediagram naar een relationeel schema (fysiek model of tabellen in de database) zullen daar natuurlijk wel de genoemde primary en foreign keys nodig zijn. In het algemeen zal daarvoor aan elke klasse (die naar een entiteit/tabel wordt vertaald) een technische sleutel worden toegevoegd: de object ID in de vorm van een betekenisloos volgnummer. Dit is niet nodig als een ander attribuut goed als primary key gebruikt kan worden. Bij elke gerelateerde klasse wordt de foreign key toegevoegd. Bij meer-op-meer relaties zal een nieuwe entiteit/tabel gedefinieerd worden met de twee ID's van de beide objecten als primaire sleutel.

Overeenkomsten

De belangrijkste overeenkomst tussen het opstellen van een relationeel model en een klassediagram is het natuurlijk structureren van de gegevens van het doelsysteem. Welke modelleringtechniek ook gebruikt wordt, welke tekeningtechniek ook gebruikt wordt om een klasse of entiteit vast te leggen, het belangrijkste is dat: 1. op een vroeg moment in de requirements analyse of het ontwerp een globaal (domein of conceptueel) model opgesteld wordt; 2. tijdens de volgende ontwerpfase(n) dit model verfijnd en gecompleteerd wordt tot een klassediagram of logisch gegevensmodel; 3. tenslotte er een fysiek model opgesteld wordt dat in de (relationele) database geïmplementeerd wordt.

Normaliseren

De eerste normaalvorm, dus geen repeterende groepen, wordt in het klassediagram niet toegepast. Maar de volgende normaalvormen, kortweg samengevat met: 'geen redundantie in het model', moeten wel worden gerespecteerd. Bijvoorbeeld, de naam van een klant hoort niet in de klasse order thuis, de datum van een order hoort niet in de orderregel thuis. Dit is in OO modellering minder strak gedefinieerd dan in het relationele model. Het bewust vastleggen van afgeleide attributen, zoals hier voor het uitstaand saldo van een klant, is wel mogelijk. Dit kan in het klassediagram voor het vastleggen van de definitie van dit begrip en later in het fysiek model ook in verband met performance (denormalisatie).
In het tweede deel worden de afzonderlijke componenten van een klassediagram beschreven.

Literatuur

1. *Warmer & Kleppe. Praktisch UML, 4de Editie. Pearson, Addison Wesley.*
2. *http://nl.wikipedia.org/wiki/Unified_Modeling_Language*
3. *http://nl.wikipedia.org/wiki/Rational_Unified_Process*
4. *www.databaseanswers.org/modelling_tools.htm*
5. *http://download.oracle.com/docs/cd/E15523_01/web.1111/b32441/undtl.htm*
6. *http://staruml.sourceforge.net/en/*
7. *www.sybase.nl/products/modelingdevelopment/powerdesigner/datamodeling*
8. *www.sparxsystems.com.au/ (voor Enterprise Architect)*
9. *www.intersystems.com/cache/ of www.intersystems.com/benchmark_lp/*
10. *https://www.hibernate.org/*
11. *Loonen. Gebruik van afgeleide gegevens in ontwerp en bouw. Database Magazine 1997/1.*
12. *Loonen. Datum en tijd in het logisch en fysiek gegevensmodel. Database Magazine 2001/6.*
13. *Loonen. Performance. Database Magazine 2002/1,2,3.*
14. *Loonen. Recursieve relaties en coding. Database Magazine 2003/4,6.*
15. *Loonen. NULL in het logisch en fysiek gegevensmodel. Database Magazine 2004/1,4.*
16. *Loonen. Logische en technische sleutels in het gegevensmodel. Database Magazine 2008/4.*
17. *Loonen. Kwaliteit van gegevens begint bij het begin. Database Magazine 2005/4,5.*
18. *Loonen. ILM in een database omgeving. Database Magazine 2008/7.*
19. *Loonen. Modelleren van subtypes. Database Magazine 1999/1.*
20. *Martin Fowler: Patterns of Enterprise Application Architecture. Zie: http://martinfowler.com*
21. *http://en.wikipedia.org/wiki/IBM_Rational_Software_Architect*
22. *http://en.wikipedia.org/wiki/Object-oriented_programming*
23. *www.uml.org/*
24. *De ANSI SQL 92 standaard: www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt hoofdstuk 4.7 Domains en 4.8.*
25. *Definitie van domeinen in PostgreSQL: http://archives.postgresql.org/pgsql-general/2005-12/msg01307.php*

Toon Loonen (toon.loonen@capgemini.com) is werkzaam bij Capgemini.