Object Modellering en Object Relational Mapping (2)

De elementen van een klassediagram

Toon Loonen

In dit tweede artikel beschrijven we de afzonderlijke componenten van een klassediagram. Dit is de klasse zelf en de verschillende types van relaties ofwel associaties.

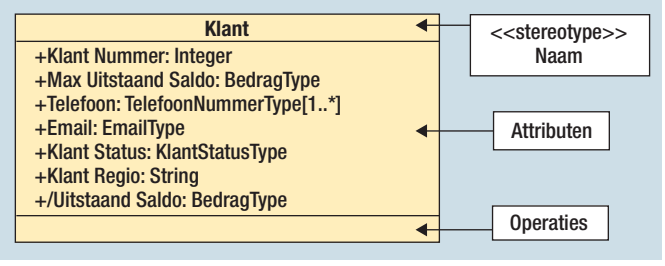
Een klasse bevat compartimenten met respectievelijk:
een naam en optioneel een (stereo)type; de attributen (of eigenschappen, property's); de operaties of methoden, zie afbeelding 1. Bij de naam kan ook een stereotype worden opgenomen zoals <<enumeration>> of <<type>> in het voorbeeld bij afbeelding 4 en 7 uit deel 1 in DB/M 5.
De attributen van de klasse komen overeen met de attributen van een entiteit in het ER model. Voor elke attribuut wordt vastgelegd:

- Eén teken '+' of '-' voor de zichtbaarheid van het attribuut;
- Een indicatie '/' om aan te geven dat het een afgeleid (uit andere gegevens berekend) attribuut betreft;
- De naam;
- Het type: voorgedefinieerd (Integer, Float, String of Boolean) of zelf gedefinieerd (BedragType, DatumType enzovoort);
- De multipliciteit: [0..-] of [1..-] betekent dat een attribuut meerdere keren kan voorkomen, zoals hier het telefoonnummer; [0..1] betekent dat een attribuut optioneel (niet verplicht) is. De default is [1] en het attribuut is dan wel verplicht.

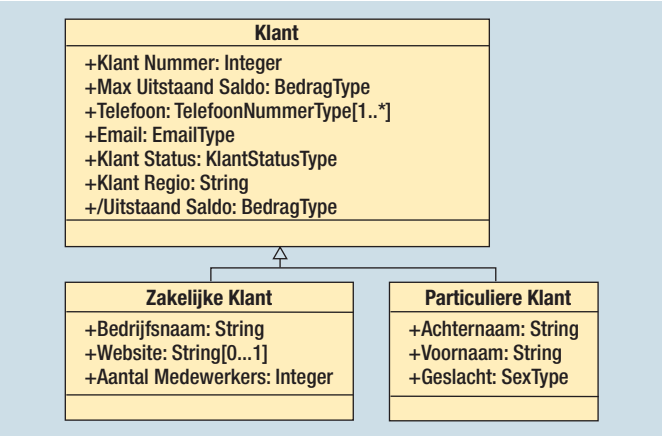
De tekens '+' of '-' voor de naam hebben de volgende betekenis:

- +: het attribuut is zichtbaar naar de buitenwereld; het is een 'publiek' attribuut.
- -: het attribuut is niet zichtbaar naar de buitenwereld; het is 'privé' voor de betreffende klasse.
- #: het attribuut is beschermd.

In de OO denkwijze kan een attribuut alleen opgevraagd worden via de operaties en derhalve moeten de attributen zelf allemaal



Afbeelding 1: Klasse.



Afbeelding 2: Inheritance.

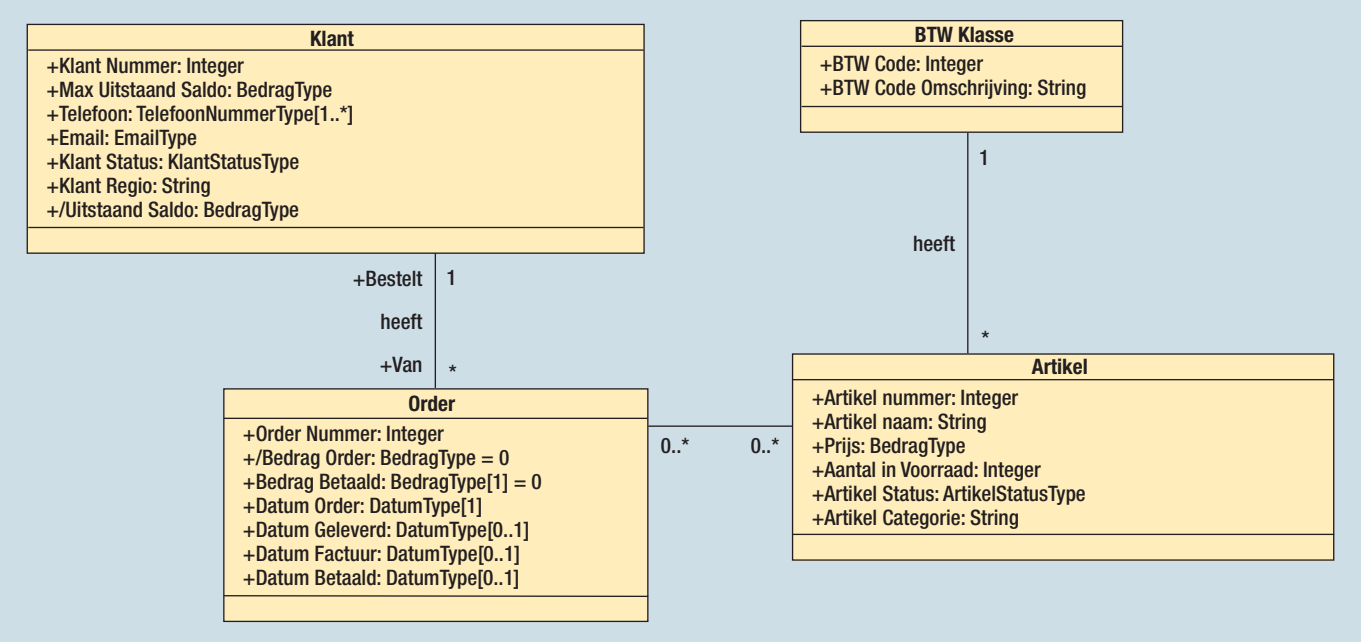
privé zijn. Maar in het relationele model en de fysieke database zijn alle attributen natuurlijk wel zichtbaar voor de buitenwereld en opvraagbaar via SQL query's. Volgens de strikte OO leer mag daarvan geen gebruik gemaakt worden.

De operaties op de klasse

Alle operaties op deze klasse moeten hier gedefinieerd worden. In een relationele database is het hele scala aan SQL query's (inclusief toevoegen, wijzigen en verwijderen) beschikbaar om op elk moment weer een andere vraag te stellen. Op een klasse moet men eerst een operatie definiëren en daarna implementeren vooraleer men op deze vraag een antwoord kan krijgen.
Voorbeelden van operaties zijn:

- Toevoegen van een nieuwe klant;
- Wijzig de status van een klant;
- Wijzig het adres, de telefoonnummers en/of het e-mailadres van een klant;
- Verwijder alle klanten die geen orders hebben;
- Tel de klanten met een uitstaand saldo groter dan een bepaalde waarde;
- Geef alle gegevens van de klant (of klanten) aan de hand van de postcode;
- Geef alle gegevens van de klant aan de hand van het klantnummer enzovoort.

Een operatie kan ook gebruik maken van gegevens in geassocieerde klassen, zoals hier de orders.
Door operaties als toevoegen, wijzigen, verwijderen, tellen, enzo-



Afbeelding 3: Associaties.

voort aan een hoger niveau (controler) toe te kennen, kunnen deze weer door alle klassen overerft worden en worden ze niet meer voor elke klasse gedefinieerd. Dit zijn dus voorgedefinieerde bewerkingen waar in de ontwerpen en de programmatuur weer gebruik van gemaakt kan worden. In de praktijk zullen niet altijd alle bewerkingen bij de klasse worden vastgelegd en zullen ook SQL query's (of een ander dialect, zoals HQL – Hibernate Query Language) gebruikt worden om gegevens uit de database op te halen, vaak ook om reden van performance.

Inheritance relatie

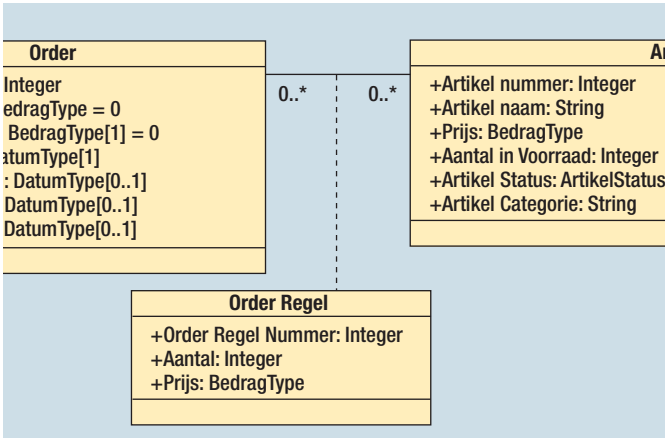
In het voorbeeld hebben we gezien dat een klant ofwel particuliere klant ofwel een zakelijke klant kan zijn, zie afbeelding 2. De zakelijke klanten 'erven' de attributen, operaties en relaties van de algemene klasse KLANT, maar hebben ook specifieke (eigen) attributen, operaties en eventueel relaties. Hetzelfde geldt voor de particuliere klanten:

- Klant is hier een 'supertype' en is een 'generalisatie' van de 'subtypes' particuliere klant en zakelijke klant;
- De particuliere klant en zakelijke klant zijn 'specialisaties' van het supertype klant.

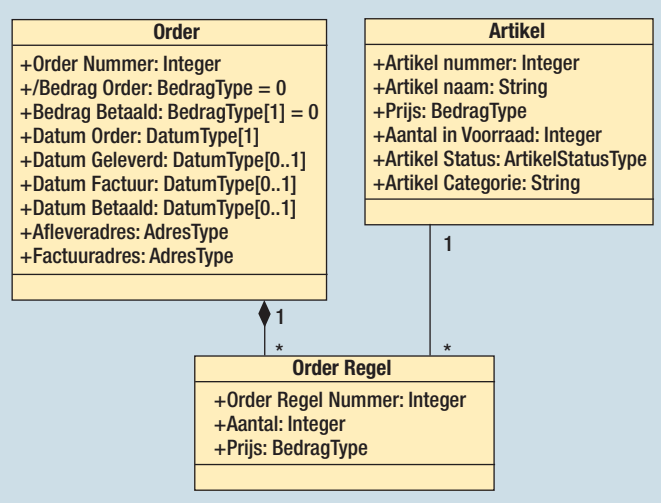
Deze relatie wordt aangegeven met de in afbeelding 2 getoonde pijl. In het voorbeeld (zie ook afbeelding 2 in deel 1) zien we nog een tweede inheritance relatie, namelijk tussen artikel en enkelvoudig respectievelijk samengesteld artikel. Hier hebben de subtypes geen eigen attributen maar wel eigen relaties. Om die relaties duidelijk in het diagram weer te geven is het vaak toch zinvol om de subtypes specifiek in het diagram op te nemen.
Subtypes sluiten elkaar uit: een klant is ofwel particuliere klant ofwel een zakelijke klant, nooit beide. Dit is een eis voor een inheritance relatie. Als dit niet het geval is, bijvoorbeeld 'Een boek is ofwel in drukvorm verkrijgbaar ofwel als PDF via Internet, ofwel beide', dan spreken we niet over een inheritance relatie. Hiervoor kun je wel een 'rolpatroon' gebruiken, zie [Ref 20].

Soms kan een subtype van type wijzigen, bijvoorbeeld een software wijziging is eerst als probleem (fout) vastgelegd maar blijkt later toch een change request te zijn, omdat er ook wijzigingen in het ontwerp nodig zijn.

Het is soms een project of bedrijfsstandaard, maar vaak een kwestie van smaak of gewoonte hoe ver men gaat bij het definiëren van subtypes. Bij klanten zou (naast de al gegeven indeling) ook een indeling gemaakt kunnen worden naar klantstatus. Een algemene regel is om alleen subtypes te definiëren als er ofwel specifieke relaties zijn, ofwel een significant aantal attributen maar voor 1 subtype van toepassing is. Het is handig om in het supertype aan te geven welk subtype het betreft, zoals hier voor het klanttype (Z of P) gedaan had kunnen worden. Zie verder ook [Ref 18].



Afbeelding 4: Associatie klasse.



Afbeelding 5: Alternatief.

Associaties

Het model in afbeelding 3 laat een aantal associaties (lijnen) zien tussen de klassen. De associaties zonder pijl zijn tweezijdig navigeerbaar, dat wil zeggen in dit voorbeeld:

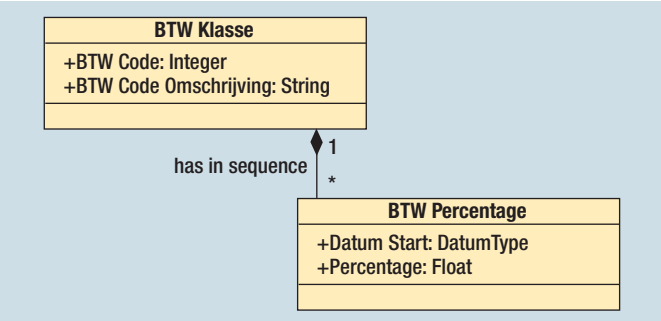
- Bij een klant kan men de orders van die klant opvragen (of beter: bij de klant is ook de informatie van de orders beschikbaar als een soort repeterend gegeven) en;
- Bij een order kan men de gegevens van de bijbehorende klant opvragen.

De associatie tussen Artikel en BTW Klasse is eenzijdig navigeerbaar:

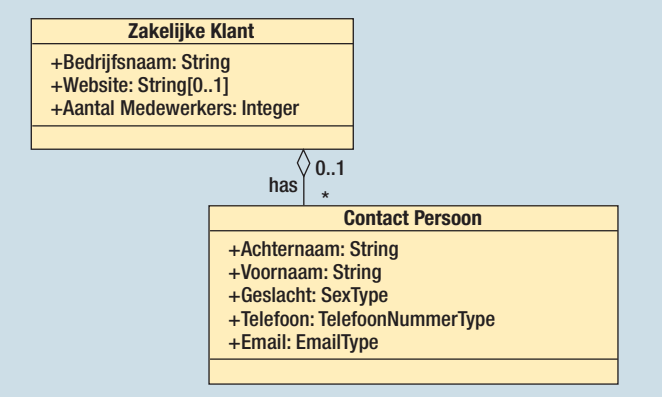
- Bij een artikel kan wel de BTW opgevraagd worden;
- Maar bij een BTW Klasse kunnen niet de artikelen met deze BTW Klasse worden opgevraagd.

De associatie heeft (optioneel) een naam en aan een of beide kanten een rolnaam:

- De naam, die halverwege de lijn wordt geschreven, geeft de relatie aan tussen de twee klassen, bijvoorbeeld een 'Een klant heeft orders' of 'Een artikel is samengesteld uit ..'
- De rolnaam, bijvoorbeeld de klant 'bestelt' een order of de order is 'van' een klant. Een rolnaam is vooral van belang als er twee of meer relaties tussen entiteiten lopen zoals bij (meer-op-meer) recursieve relaties (zie afbeelding 6 in deel 1 in DB/M 5).



Afbeelding 6: Compositie.



Afbeelding 7: Aggregatie.

Bij een associatie kan ook een multipliciteit worden aangegeven waarbij:

- Een - (of 0..-) staat voor 0, 1 of meer, bijvoorbeeld een klant heet 0, 1 of meer orders;
- 1..- staat voor 1 of meer (dus tenminste 1), bijvoorbeeld een order heeft tenminste 1 orderregel;
- 1 staat voor altijd 1, bijvoorbeeld een order hoort bij precies 1 klant;
- 0..1 staat voor 0 of 1, bijvoorbeeld een contactpersoon hoort optioneel (niet verplicht) een zakelijke klant.

Associatieklasse

Als bij een associatie nog extra attributen horen, zoals in afbeelding 4, dan kan dat worden aangegeven met een associatieklasse. In de praktijk zal men hiervoor meestal een gewone klasse definiëren met een associatie met beide oorspronkelijke klassen zoals in afbeelding 5.

Compositie en Aggregatie

Composities en aggregaties definiëren een soort 'Bevat, Bestaat uit of Hoort bij' relatie. De relaties tussen BTW code en BTW percentage (afbeelding 6) is van het type compositie, evenals de relatie tussen order en orderregel (afbeelding 5). Een orderregel kan niet bestaan zonder order, ofwel, als de order wordt verwijderd, dan worden de bijbehorende orderregels ook verwijderd. De compositie wordt aangegeven met een lijn met aan het einde een gesloten (dichte) ruit, zie afbeelding 6.
Een contactpersoon hoort bij een zakelijke klant, maar voor de relatie tussen zakelijke klant en contactpersoon hebben we (in dit voorbeeld) vastgelegd dat de contactpersoon niet verwijderd hoeft te worden als de zakelijke klant verwijderd wordt. Dit is een aggregatie en deze wordt aangegeven met een lijn met aan het einde een open ruit, zie afbeelding 7. Een aggregatie kan ook vaak als een gewone associatie getekend worden.
Zie voor de lijst van literatuurreferenties deel 1 in DB/M 5.
In het derde en laatste deel wordt de implementatie van het klassediagram in een fysieke relationele database behandeld.

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