Modellering van scratch af aan

Objecten zonder klasse

Henk Jan Pels

Onze informatiemaatschappij wordt meer en meer gegevensintensief. Databases worden talrijker en groter. Wie databases wil begrijpen, en daarmee de informatiesystemen die ze gebruiken alsmede de informatiemaatschappij die erop draait, moet data kunnen modelleren. Maar niet alleen daarom. IT wordt steeds vaker gebruikt voor het bouwen van virtuele werelden. Conceptuele datamodellen helpen communiceren over abstracte zaken.

In de loop van de tijd heel wat gegevensmodellen voorgesteld, zoals het hiërarchische model, het netwerkmodel, NIAM, het model van entity en relationship (E/R), het relationele model, het objectgeoriënteerde model en UML. Mijn ervaring is dat het er niet zoveel toe doet welk model je gebruikt, als degene voor wie je dit maakt, het maar kan lezen. En als je maar heel precies definieert wat elk symbool in je model betekent.

Voor het eerste is het handig een standaard te hebben. Die standaard is momenteel UML, omdat software-ontwikkelaars het heel algemeen als zodanig aanvaarden. De gangbare leerboeken doorlezend, stuiten we op een bezwaar van UML. De betekenis lijkt niet zo precies gedefinieerd als we gewend waren van bijvoorbeeld het relationele model. Maar slaan we vervolgens de formele specificatie erop na, dan blijkt dat best mee te vallen. Daarom dient UML in het vervolg als basis voor onze discussie.

UML 'PLUS'

We gaan aan UML een paar dingen toevoegen: een tekstuele notatie en enkele constructies om beter met abstractie te kunnen omgaan. UML biedt alleen een grafische notatie voor datastructuren. Voor database-ontwerp is dat geen probleem; een ontwerp wordt in principe alleen aangeboden als een compleet en consistent geheel. Maar voor gebruik van het model als expressiemiddel in een discussie over zaken die te abstract zijn om te tekenen, is het handiger om met tekst te werken. Een model kan dan stap voor stap worden opgebouwd, bediscussieerd en aangepast.

Eerst een korte filosofie over wat de gegevens die we willen modelleren, eigenlijk zijn. Een datamodelleur beschouwt de werkelijkheid als een verzameling feiten. Een gegeven is de beschrijving van een feit. Zoiets wordt ook wel een bewering genoemd. Een bewering is waar als het beweerde feit er is en onwaar als het beweerde feit er niet is. Een feit kan zijn of niet zijn en is in die zin een binair begrip. Een bewering is ternair: hij kan niet alleen waar of onwaar zijn, maar ook onzin. Een bewering is onzin als het bedoelde feit niet past in de beschouwde werkelijkheid, zodat er over zijn of niet zijn niet zinnig valt te praten.

Neem de bewering "de zon is rood". Dat is een zinnige bewering in een wereld waarin de zon ook geel of oranje kan zijn. Die bewering is onwaar op een moment dat de zon waarneembaar geel is. In diezelfde wereld is de bewering "de zon is schuldig" waar noch onwaar. Schuld of onschuld van de zon is eenvoudig niet aan de orde (niet relevant). De bewering is dus onzin.

Het Nederlandse woord voor database was ooit -of is nog?-: gegevensverzameling. Elk gegeven is een bewering over een feit in een relevante wereld en is gecodeerd in structuren van herkenbare symbolen. De betekenis van een symbool hangt meestal af van de plaats van het symbool in de structuur. Daarom is het struc-

Datamodellen: nieuwe eisen (1)

Nieuwe producten en de productielijnen om die te maken, worden in 3D gevisualiseerd, lang voordat ze fysiek gemaakt worden. Datamodellen zijn een heel handig hulpmiddel om zichtbaar te maken hoe delen van het product en delen van de productielijn met elkaar samenhangen.
Een andere ontwikkeling is dat producten in steeds meer 'smaken' en 'kleuren' worden aangeboden. Om die variëteit te kunnen beheersen, moeten varianten in productfamilies worden geklasseerd. Conceptuele datamodellen zijn goed bruikbaar bij de communicatie over zulke abstracte zaken.
Voor al deze nieuwe toepassingen worden nieuwe eisen aan datamodellen gesteld. Ze moeten met name meer steun bieden bij het vormen van abstracties. Daarover schrijft Henk Jan Pels in dit en een aantal volgende artikelen.

tureren van gegevens zo belangrijk voor het bepalen van de betekenis. Heel belangrijk is ook het onderscheid tussen zin en onzin. Een database die onzinnige gegevens bevat, heet inconsistent en is in principe onbruikbaar. Daarom hoort bij een database altijd een verzameling integriteitsbeperkingen (integrity constraints) die nauwkeurig beschrijven aan welke maatstaven een gegeven moet voldoen om zinnig te zijn. Als een gegeven zinnig is, kan het waar of onwaar zijn.

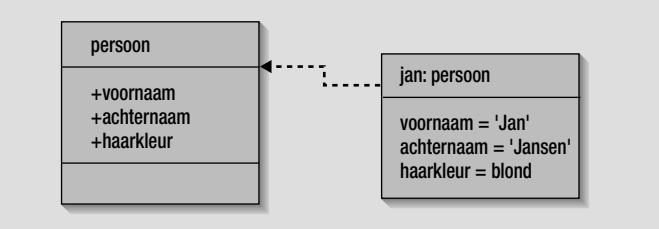
De fundamentele aanname waarop de betekenis van een database gebaseerd is, heet de *closed world assumption*. Deze stelt dat alle gegevens die in de database staan waar zijn, en dat alle volgens de integriteitsbeperkingen zinnige gegevens die niet in de database staan, onwaar zijn.

Op grond van deze aanname is het mogelijk logische redeneringen toe te passen op de inhoud van de database en daaruit conclusies af te leiden over de relevante wereld. Een database die onzinnige gegevens bevat is onbruikbaar, omdat logica niet bestand is tegen onzin. Redeneringen worden onbetrouwbaar als er onzin in de database voorkomt.

We lopen nu aan tegen de vraag wat eerst komt: de gegevens of de beperkingen. De beperkingen hebben we nodig om structuur en betekenis te geven aan gegevens. Anderzijds wordt het verschil tussen zin en onzin pas in de loop van de discussie duidelijk, en willen we dus eerst gegevens kunnen noteren voordat er beperkingen zijn waar ze inpassen. Kortom, we willen gegevens en beperkingen in willekeurige volgorde naar voren kunnen brengen. Hoe dat kan wordt in het volgende uitgelegd.

GEGEVENS EN BEPERKINGEN

Objectgeoriënteerde modellen als UML modelleren een wereld die bestaat uit entiteiten met eigenschappen. Een gegeven is een uitspraak over een eigenschap van een entititeit. Behalve uit feiten, bestaat de wereld van een datamodelleur uit entiteiten, zaken waarover feiten kunnen bestaan. Een object is een uniek symbool dat een specifieke entiteit representeert. Elke afzonderlijke relevante entiteit krijgt een uniek object toegewezen in de database.



FIGUUR 1: EEN SIMPEL UML-DIAGRAM.

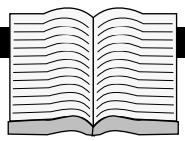
OO modellen doen vaak heel ingewikkeld over wat een object is, maar het is heel eenvoudig: een uniek symbool en meer niet. Men zegt vaak dat een object een unieke identiteit heeft, maar het is beter te zeggen dat elk object een uniek symbool is. In een fysieke database zal daarvoor een binair getal gebruikt worden, in een conceptueel model -en dus in onze conceptuele database- mag dat elk representeerbaar symbool zijn. Voor de hand liggende symbolen zijn letters en cijfers, maar ook woorden -combinaties van letters en cijfers- of grafische symbolen komen in aanmerking.

Figuur 1 toont een voorbeeld van een object, genoteerd in UML. We zien het object jan. Jan is van objecttype persoon. Kennelijk is er in de relevante wereld een persoon die wij kennen als Jan. De attribuutwaarden van object jan staan voor de beweringen: "De voornaam van jan is 'jan'", "De achternaam van jan is 'jansen'" en "De haarkleur van jan is blond".

Het is nuttig even te wijzen op enkele subtiele details. Er is een object jan. Hierin wordt het woord jan gebruikt als symbool. Het is een uniek symbool, in de zin dat het overal waar het in de database wordt gebruikt, verwijst naar dezelfde persoon in het UoD. De voornaam van jan is 'jan': de apostrofs geven aan dat hier niet het unieke symbool jan wordt bedoeld, maar de tekenrij bestaande uit de drie -op zichzelf unieke- symbolen j, a en n. De haarkleur van jan is blond, zonder apostrofs. Blond is hier dus een uniek symbool, dat verwijst naar een bepaalde kleur haar.

In UML-handleidingen wordt vaak gezegd dat het blok met in de bovenbalk de tekst 'jan: persoon' een object voorstelt. Dat is (in onze interpretatie) niet helemaal juist, want alleen het symbool jan is een object. Het blok stelt het object jan voor met een aantal attribuutwaarden van het object. We mogen dus wel zeggen: "een

AGENDA



Congressen, beurzen e.d.

8-11/7: TDWI World Conference
Wereldcongres van The Data Warehousing Institute. Londen, Olympia. Org./inf.: www.dw-institute.co.uk

Cursussen, seminars e.d.

25-26/6: DB2 Versie 7 Update
Cursus. Utrecht. Kosten: € 795,-. Org./inf.: KBCE, www.kbce.nl

25-26/6: E-business architectures
Seminar met Peter Hinssen. Neder-over-Heembeek (B), Business-faculteit, 14.00-21.00 uur. Kosten: € 920,-. Org./inf.: I.T. Works, www.itworks.be/e-architectures.html, (00) 32 9 2415613.

3-4/7: XML/XSL architectures in practice
Seminar met Paul Hermans en Steven Noels. Diegem (B), Sofitel Brussels Airport, 14.00-21.00 uur. Kosten: € 920,-. Org./inf.: I.T. Works, www.itworks.be/XML_Architectures.html, (00) 32 9 2415613.

16/7: Seminar voor banken en overige financiële instellingen
Georganiseerd door leverancier Sagent Technology Benelux. Met aandacht voor aspecten van het Basel II Capital Accord. Info: info@sagent-benelux.com, (010) 2582607.

object heeft attributen", maar niet "een object bestaat uit attributen". Een object is namelijk een ondeelbaar symbool.

VOORBEREIDING

Een moeilijkheid met UML is dat een object slechts wordt voorgesteld in combinatie met de objectklasse waartoe het behoort. Die klasse moet dus eerst bekend zijn. Bovendien kan een object alleen worden voorgesteld met al zijn attribuutwaarden. Dat betekent dat we -voordat we in UML formeel iets over een object kunnen zeggen- een heleboel moeten voorbereiden: er moet een passende klasse zijn en we moeten iets zeggen over elk attribuut dat in die klasse is gedefinieerd.

Het kan eenvoudiger door toe te staan elementaire gegevens of afzonderlijke attribuutwaarden te noteren. Dat kan het makkelijkst in de vorm van een drie-tupel:

```
<object, attribuut, waarde>
```

Hiermee kunnen we bijvoorbeeld direct in huis vallen met de bewering dat jan blond haar heeft:

```
<jan, haarkleur, blond>
```

Om het toch iets ingewikkelder te maken, bekenken we dat een database, die dan ook wel *knowledge base* wordt genoemd, vaak de mogelijkheid biedt nieuwe gegevens af te leiden met behulp van beperkingen, ook wel regels genoemd.

Het is comfortabel net te kunnen doen alsof al die afleidbare gegevens ook in de database staan. Omdat we het hebben over conceptuele modellen, hoeven we ons geen zorgen te maken over hoe die gegevens worden afgeleid. Wel moeten we onderscheid kunnen maken tussen gegevens die bewust door een gebruiker zijn ingevoerd en gegevens die op basis van regels zijn afgeleid. De laatste kunnen namelijk impliciet veranderen door nieuwe gegeven, de eerste niet. De oorspronkelijke gegevens noemen we 'gesteld' *asserted* en de andere afgeleid. Het verschil geven we aan met een vierde coördinaat in het tupel, die de waarde 1 krijgt voor gestelde en de waarde nul voor afgeleide gegevens:

```
<object, attribuut, waarde, gesteld>
```

Als de haarkleur van jan door een databasegebruiker is vastgesteld, schrijven we:

```
<jan, haarkleur, rood, 1>
```

Stel dat er regels zijn waaruit valt af te leiden dat bij een rode haarkleur altijd een zonnig humeur hoort, dan staat impliciet in de database ook:

```
<jan, humeur, zonnig, 0>
```

In de vorm van een verzameling van zulke gegevenselementen kan een database van willekeurige omvang worden gevormd, waarin alle vrijheid bestaat om aan objecten attribuutwaarden toe te kennen. Een object bestaat als er tenminste één attribuutwaarde is. Een object kan voor een bepaald attribuut een willekeurig aantal waarden hebben. De bekende puntnotatie:

```
object.attribuut
```

levert de verzameling gestelde en afgeleide waarden op. Als er geen waarde is gesteld of afgeleid, levert deze expressie de lege verzameling als waarde op.

VRIJHEID

Op de hierboven beschreven manier kunnen we databases bouwen met een veel grotere vrijheid dan het relationele model, waarin alles in vooraf gedefinieerde tabellen moet passen. Er is ook meer vrijheid dan in gangbare OO modellen, zoals UML, waarin objecten alleen in vooraf gedefinieerde klassen kunnen worden gecreëerd.

Uiteraard moeten aan deze vrijheid beperkingen kunnen worden opgelegd. Daarover meer in het volgende artikel uit deze reeks. ●

Dr. ir. H. J. Pels (H.J.Pels@tm.tue.nl) is werkzaam aan de Faculteit Technologie Management, Capaciteitsgroep Informatie & Technologie, van de Technische Universiteit Eindhoven.