Een abstracte en bijzonder krachtige constructie

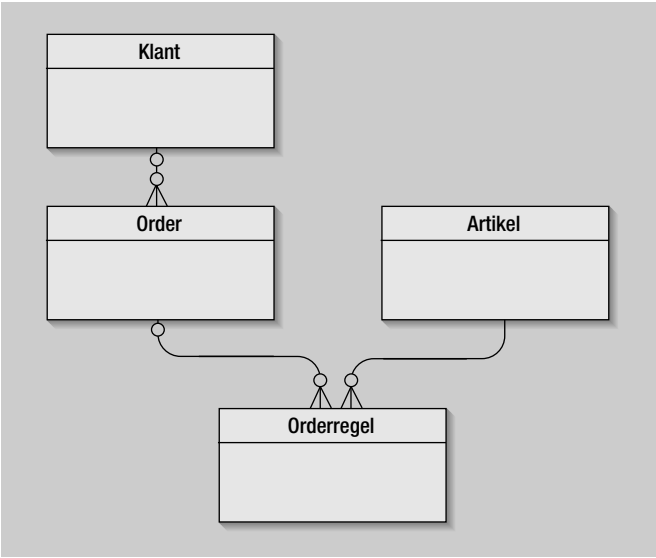
Recursieve relaties in het gegevensmodel

Toon Loonen

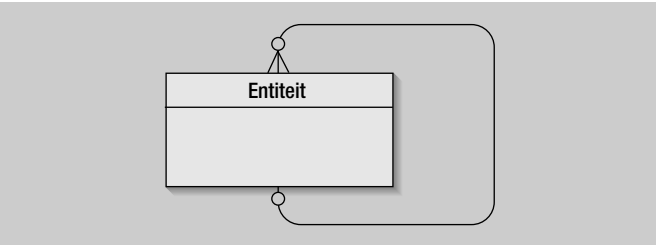
Het gegevensmodel is een abstractie van de werkelijkheid. Recursieve relaties vormen hierin een nog abstracter, dus voor velen ook moeilijker, constructie. Veel aankomende database administrators en anderen die de tekeningen moeten lezen, hebben moeite met deze constructie. Toch komen deze recursieve relaties veel voor in de praktijk en zijn ze bijzonder krachtig, mits goed toegepast. In dit artikel worden de modellen verder uitgewerkt, in een volgend artikel worden deze modellen vertaald naar coding.

De tekening van een gegevensmodel lijkt veel op een hiërarchisch structuurschema, bijvoorbeeld het organogram van een bedrijf. Het grote verschil is dat een element (object, entiteit) in een gegevensmodel meer *parents* kan hebben, terwijl in een zuiver hiërarchisch schema een element maar een parent (hoger niveau afdeling in het bedrijf) kan hebben.

Zie bijvoorbeeld afbeelding 1 met het gegevensmodel voor klanten, orders en artikelen. Hier is de structuur klant-order-orderregel nog hiërarchisch, maar orderregel heeft twee parents, namelijk order en artikel. Bij een recursieve relatie heeft een entiteit een relatie met zichzelf, zoals in afbeelding 2 is te zien.



Afbeelding 1: Hiërarchisch structuurschema.



Afbeelding 2: Recursieve relatie.

Enkele voorbeelden van recursieve structuren zijn:

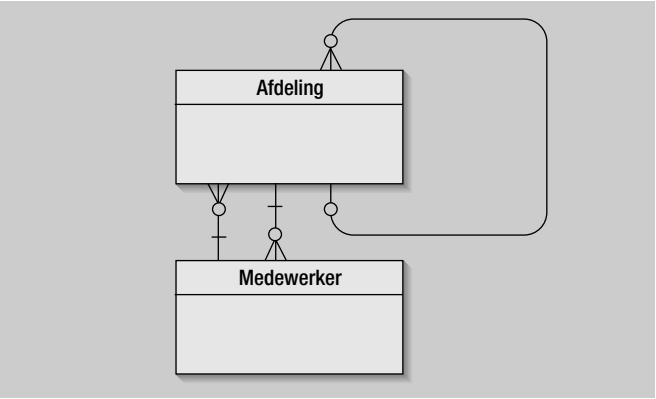
- De afdelingen van een bedrijf (afdeling met onderafdelingen);
- De *bill of material* van een artikel, bijvoorbeeld van een computer of auto;
- De directory-structuur op een computer ((sub)directory's met bestanden);
- Menustructuur in een applicatie (hoofdmenu, submenu's en functies);
- Formules met delen van formules.

Er zijn twee soorten recursieve structuren. In de eerste plaats is er de "1 op meer" recursieve relatie: een hiërarchische structuur; waarin een object maar één parent kan hebben, bijvoorbeeld de afdelingen in een bedrijf. De tweede structuur is de "meer op meer" recursieve relatie: een netwerkstructuur waarin een object meer parents kan hebben, bijvoorbeeld een submenu of functie kan op meer plaatsen in een menustructuur worden geplaatst. De bovengenoemde voorbeelden worden verder uitgewerkt in een serie schema's.

DE AFDELINGEN VAN EEN BEDRIJF

Het gegevensmodel zou er uit kunnen zien als in afbeelding 3. Er zijn drie relaties aanwezig, die toegelicht worden.

1. De relatie van afdeling naar zichzelf is hier de recursieve relatie. Deze is "1 op meer" en dus hiërarchisch. Een afdeling verwijst via deze relatie naar de hoger niveau afdeling waar deze onder valt. Andersom geeft deze relatie aan, welke afdelingen er onder een hoofdafdeling vallen. Het aantal niveaus is



AFBEELDING 3: MODEL AFDELINGEN VAN EEN BEDRIJF.

volgens dit model onbeperkt, maar zal in de praktijk vaak tussen de drie en de zes vallen.

2. De "1 op meer" relatie tussen afdeling en medewerker geeft aan tot welke afdeling een medewerker behoort.
3. De "1 op meer" relatie tussen medewerker en afdeling geeft hier aan wie de chef is van een afdeling. Hiervoor wordt in de tabel Afdeling een buitensleutel "medewerkercode-chef" opgenomen die verwijst naar een medewerkercode in de tabel Medewerker. Volgens dit model kan een chef (tijdelijk of na elkaar) meer dan een afdeling onder zich hebben.

Het alternatief voor een recursieve structuur is het hard coderen van de verschillende niveaus in het gegevensmodel. Bijvoorbeeld bij een bedrijf met afdelingen zou het nog mogelijk zijn om voor elk niveau een entiteit te definiëren: Bedrijf, Divisie, Sector, Afdeling en Onderafdeling.

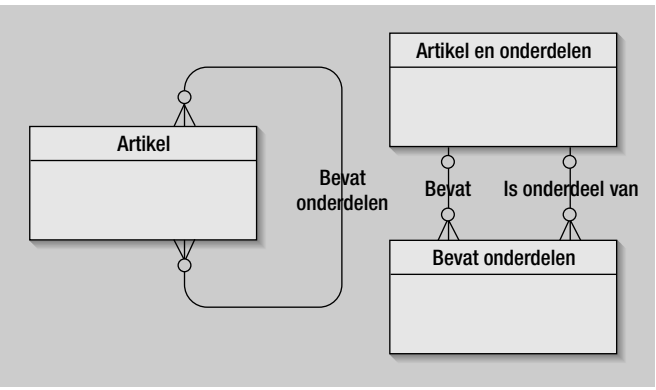
Maar dit is niet flexibel, want als het niveau niet overal even diep is moeten dummy-sectoren of -afdelingen in het systeem worden opgenomen om de ontbrekende (tussenliggende) niveaus op te vullen. Of er moeten extra relaties worden gedefinieerd, bijvoorbeeld tussen divisie en afdeling. De nette manier om dit uit te werken is dan ook het gebruik van de recursieve structuur.
Entiteit Afdeling bevat (fysiek) de attributen:

- Code afdeling: primaire sleutel;
- Code afdeling hoger niveau: buitensleutel, verwijst naar Code afdeling van een ander record. De waarde is NULL voor het hoogste niveau, anders ingevuld;
- Naam afdeling;
- Code medewerker van chef: buitensleutel.

Entiteit Medewerker bevat de attributen:

- Code medewerker: primaire sleutel;
- Code afdeling: buitensleutel;
- Naam medewerker.

De relatie van afdeling naar zichzelf is altijd zowel aan de "1 kant" als aan de "meer kant" optioneel. Immers de afdeling van het hoogste niveau (het bedrijf) heeft geen niveau meer boven zich en de afdelingen onder aan de hiërarchie hebben geen onderafdelingen meer. Dit geldt ook voor andere soorten gegevens.



AFBEELDING 4: MODEL "BILL OF MATERIAL".

DE BILL OF MATERIAL

Het gegevensmodel van een *bill of material* zou er uit kunnen zien als in afbeelding 4. Links staat het ongenormaliseerde model met een "meer op meer" relatie tussen de artikelen en onderdelen. Meer op meer relaties worden uitgenormaliseerd in twee "1 op meer" relaties en dat schema staat rechts. Deze twee relaties hebben een naam:

1. De relatie "Bevat" geeft aan welke onderdelen er in een artikel zitten;
2. De relatie "Is onderdeel van" geeft aan in welke artikelen een bepaald onderdeel wordt gebruikt.

Bijvoorbeeld, bij auto's kan een bepaald type motor in meer typen auto's worden gebruikt. Elk van deze auto's bevat behalve deze motor ook nog andere onderdelen. Van de onderdelen (de motor) kan desgewenst nog worden aangegeven welke (sub)onderdelen daar nog in thuis horen. Om verwarring te voorkomen is het gewenst om hier aan de relaties een duidelijke naam te geven.

Entiteit "Artikel en onderdelen" bevat de attributen:

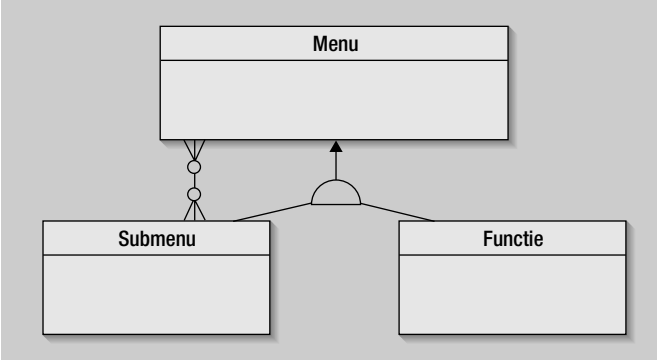
- Code Artikel;
- Naam;
- Prijs, voorraad, enzovoort.

Entiteit "Bevat onderdelen" bevat de attributen:

- Code Artikel hoger;
- Volgnummer;
- Code Artikel;
- Aantal.

Let op: hier is een recursieve relatie opgesplitst in twee "1 op meer" relaties en een nieuwe entiteit. In het eerste voorbeeld hadden we ook twee relaties tussen de tabellen "Afdeling" en "Medewerker". Dit zijn echter twee afzonderlijke (zelfstandige) relaties! Ze lopen ook niet beide in dezelfde richting, zoals bij de twee relaties in dit voorbeeld.

Bij een "bill of material" zit er functioneel nog enige hiërarchie in de gegevens (auto en onderdelen), maar met hetzelfde model kunnen we ook andere objecten met hun relaties modelleren, bijvoorbeeld Personen en hun familierelaties of Entiteiten in een gegevensmodel en hun onderlinge relaties, zie [1].



Afbeelding 5: Model menustructuur in een applicatie.

MENU- EN DIRECTORYSTRUCTUUR

Het gegevensmodel van een menustructuur kan gemodelleerd worden zoals in afbeelding 5. De tabel Menu bevat twee subtypes: Submenu en Functie. Dit betekent dat een record in de tabel Menu ofwel een Functie ofwel een Submenu is. Een Functie kan door de gebruiker worden uitgevoerd. Een Submenu zal weer andere Functies en/of subsub-menu's bevatten.

Fysiek kunnen het supertype Menu en de twee subtypes in een tabel worden opgeslagen. De "meer op meer" relatie tussen subtype "Submenu" en supertype "Menu" moet weer opgebroken worden in twee "1 op meer" relaties. We krijgen dan een fysiek model als in afbeelding 6. De entiteit "Menu en functies" bevat daarin de attributen:

- Code Menu of Functie;
- Type (M of F voor Menu of Functie);
- Naam.

Entiteit "Menu bevat regels" bevat de attributen:

- Code menu: buitensleutel naar menu;
- Volgnummer (van de regel op het menu);
- Code van de functie of het submenu: buitensleutel naar submenu of functies in de bovenstaande tabel.

Voor een verdere toelichting op de functionaliteit van dit model, zie [2] en op het gebruik van subtypes, zie [3].

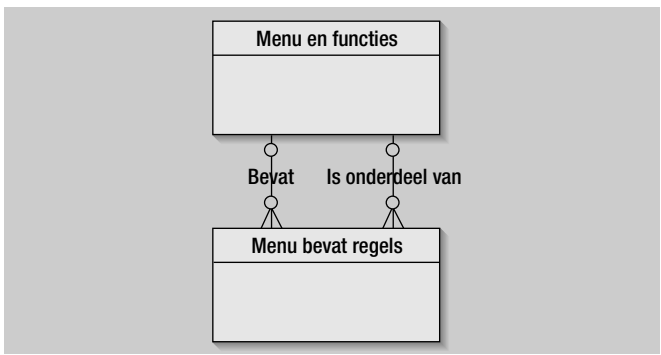
De structuur van directory's met subdirectory's en bestanden kan ook met subtypes worden gemodelleerd, zoals bij het menu hiervoor. Het kan echter ook zoals in afbeelding 7 worden uitgewerkt. Merk op dat dit weer een hiërarchisch model is, dus dat volgens dit model een subdirectory maar in één hoofddirectory kan worden opgenomen. De relatie van Directory direct naar zichzelf is een gewone hiërarchische recursieve relatie zoals in het eerste voorbeeld met afdelingen. De Entiteit "Snelkoppeling directory" is een soort "getrapte recursieve relatie". Via deze entiteit en twee relaties kan een directory ook naar andere directory's verwijzen. Logisch gezien kan hiermee een "meer op meer" directory-structuur worden gecreëerd. Er is echter een verschil met een echte "meer op meer" recursieve structuur.
Microsoft heeft in Windows de bestandsstructuur volgens dit schema uitgewerkt. De gebruiker kan nu, door een directory naar een andere directory te slepen, kiezen of hij de directory naar de andere directory wil kopiëren, verplaatsen of in de andere directory een snelkoppeling wil plaatsen. Bij het kopiëren blijft de structuur hiërarchisch omdat er nu twee fysieke directory's zijn, die elk naar hun eigen hoofddirectory verwijzen. De kopie kan gewijzigd worden zonder invloed op het origineel.

Bij de laatste keuze wordt de snelkoppeling gecreëerd, maar de werking is anders dan bij het fysiek plaatsen van de nieuwe directory in de hoofddirectory: Bij het kopiëren van de hoofd-directory (bijvoorbeeld ten behoeve van back-up naar een CD) worden subdirectory's wel automatisch mee gekopieerd, maar de directory's waar de snelkoppeling naar verwijst niet.

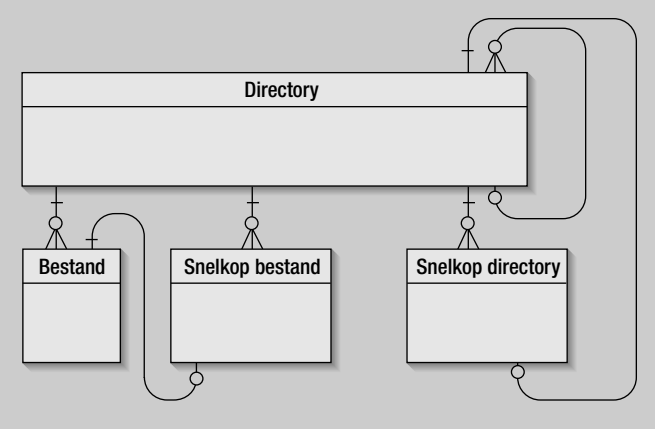
In UNIX is dit anders uitgewerkt, daar kan een bestand of directory wel in meer directory's staan! De *inode* tabel houdt bij waar (in welke directory's en met welke namen) een bestand gebruikt wordt. Pas als de laatste verwijzing naar het bestand wordt verwijderd, wordt het bestand zelf ook fysiek verwijderd.

FORMULES MET DELEN VAN FORMULES

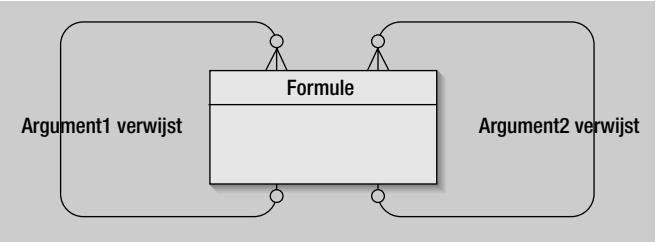
Een formule kan worden opgesplitst in delen, bijvoorbeeld de formule ((wortel(5)) * (2+3)) kan worden opgesplitst in de delen die hier tussen haakjes staan. Elke formule kan weer worden geschreven als <argument1> <operatie> <argument2>.
De operatie is dan + (optellen), - (aftrekken), * (vermenigvuldigen), / (delen) en ^ (machtsverheffen). De argumenten zijn getallen (constanten of waarden in een tabel) of weer subformules.
De bovenstaande formule kan dan als volgt worden opgebroken:
f1 = (2+3): argument1 = 2; argument2 = 3; operatie = optellen.
f2 = wortel(5): argument1 = 5; argument2 = 0,5; operatie = machtsverheffen (immers de wortel is hetzelfde als "macht een half").
f3 = (f2 * f1).
Hier verwijst de formule in het derde record naar de eerste en tweede formule. Door deze structuur in een gegevensmodel onder te brengen, kunnen ingewikkelde formules in een database worden opgeslagen. Het model is in feite erg eenvoudig (afbeelding 8), de uitwerking wat minder. De tabel Formule bevat nu twee recursieve relaties! Er zijn immers in elk record twee (potentiële) buitensleutels, dus attributen die verwijzen naar een code in een ander record. De tabel bevat dan (weer fysiek) de volgende attributen:



Afbeelding 6: Aangepast model menustructuur.



AFBEELDING 7: MODEL DIRECTORY-STRUCTUUR.



AFBEELDING 8: MODEL RECURSIE IN EEN FORMULE.

- Code formule;
- Waarde argument1;
- Waarde argument2;
- Type argument1 (F: subformule, C: constante of D: database veld);
- Type argument2;
- Operatie.

De waarde van argument (1 of 2) is alleen een correcte buitensleutel als het type veld aangeeft dat de waarde naar een subformule wijst. Deze controle kan alleen door het dbms (met een *foreign key constraint*) worden uitgevoerd als voor elk argument drie attributen worden gedefinieerd, namelijk:

- Waarde argument1 als subformule (Type argument1 = "F");
- Waarde argument1 als constante;
- Waarde argument1 als database veld.

Door de controle in de onderhoudsprogrammatuur te leggen in plaats van in de database, kan met dit ene attribuut worden volstaan. Het is aan de smaak van de DBA voor welke oplossing wordt gekozen.

We hebben de structuur hier weer hiërarchisch gemaakt, dus een subformule kan maar op een plaats gebruikt worden. In deze tijd met aandacht voor hergebruik is dat mogelijk niet meer acceptabel. Door de relaties "meer op meer" te maken wordt het mogelijk om een subformule op meer plaatsen (in meer formules) te gebruiken. Fysiek moeten elk van deze twee relaties dan weer worden opgebroken in een nieuwe entiteit en twee "1 op meer" relaties. We krijgen dan drie tabellen in plaats van de ene tabel hierboven. Voor een uitgebreider beschrijving van dit concept, zie [4]. De hierboven beschreven voorbeelden zijn binaire operaties. De operatie (+, *) heeft altijd twee argumenten. In de formule tabel kunnen ook enkelvoudige (unary) operaties (zoals "wortel van" of "sinus van") worden opgenomen. Alleen argument 1 zal dan ingevuld zijn en argument 2 (Type en waarde) blijft dan leeg (NULL).

Hiërarchisch tekenen

De auteur is gewend om gegevensmodellen zo te tekenen dat de "1 op meer" relaties altijd van boven naar beneden lopen. Alleen in enkele gevallen is dat niet mogelijk, bijvoorbeeld als een recursieve relatie "rond loopt ". Verder bevat het voorbeeld over Afdelingen hiervoor nog een relatie ("1 op meer" relatie tussen medewerker en afdeling, de "chef") die ook andersom loopt.

Niet elke DA of DBA denk hier hetzelfde over. Bij een quick scan van gegevensmodellen in DB/M jaargang 2002 bleken de meeste modellen zo getekend te zijn, dat "1 op meer" relaties zowel van boven naar beneden als ook van beneden naar boven liepen en van links naar rechts, enzovoort.

Toch is de auteur van mening dat een modellen waarbij de "1 op meer" relaties van boven naar beneden lopen, gemakkelijker te lezen zijn, zeker voor niet-DBA's. De belangrijkste "objecten" (bijvoorbeeld klanten en artikelen) staan dan boven in het model, waar men meestal ook begint te kijken en lezen. Het model naar beneden volgend, komt men dan bij details als orders, orderregels en afleveringen.

Uiteraard is het niet noodzakelijk om het model zo te tekenen. Het zal dus een kwestie van smaak blijven.

CONCLUSIE

Recursieve relaties vormen een abstracte maar bijzondere krachtige constructie bij het opstellen van een logisch of fysiek gegevensmodel. Het is voor een data(base) administrator belangrijk om hiermee goed om te gaan en deze constructie ook goed uit te leggen aan degenen voor wie het gegevensmodel wordt opgesteld: de functioneel ontwerpers, bouwers, testers en (andere) gebruikers.

In een volgend artikel zal verder ingegaan worden op het recursief aanroepen van coding. ●

LITERATUUR

1. *Loonen, Gegevensmodel van een distributed data dictionary. Database Magazine 1997/4.*
2. *Loonen, De ontwikkelstraat, hergebruik door inzet van architectuur. Software Release 2000/7-8.*
3. *Loonen, Modelleren van subtypes. Database Magazine 1999/1.*
4. *V.d.Graaf, Het adaptief ontwerpen van bedrijfsregels. Database Magazine 2003/3.*

Toon Loonen (toon.loonen@cgey.nl, toon.loonen@inter.nl.net) is als consultant werkzaam bij Cap Gemini Ernst & Young.