Nuttige aanvulling op denkpalet van de modelleur

Reïficatie en navigatie van relaties

Maurice Gittens

De vaktechnische uitdrukkingsvaardigheid van informatiemodelleurs is te verbeteren wanneer het beschikbare palet aan modelleerbouwstenen wordt uitgebreid met hogere orde concepten.

In voorgaande artikelen zijn hogere orde concepten op basis van identificerende relaties geïntroduceerd. Dit artikel introduceert hogere orde concepten op basis van reïficerende relaties, en het concept van reïficatie via een toelichting op de functionele navigatiemogelijkheden die door respectievelijk niet-identificerende en identificerende relaties worden geboden. Een voorbeeld geeft intuïtief een beeld van de toepasbaarheid van reïficatie bij informatiemodellering.

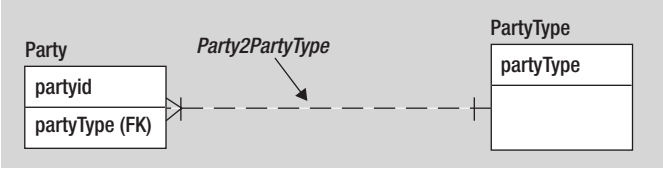
Niet-identificerende relaties

Niet-identificerende relaties maken unidirectionele navigatie tussen concepten mogelijk. De navigatierichting is hierbij van het child concept naar het parent concept. Een niet-identificerende relatie is dus te zien als een wiskundige functie die het child concept afbeeldt op het parent concept, zie afbeelding 1. Dit wil zeggen dat we de foreign key relatie PartyToPartyType mogen zien als een wiskundige functie met als domein het concept Party en als bereik het concept PartyType. Deze functie definieert voor iedere partij wat zijn type is. De navigatie van een instantie van Party naar het type van de betreffende instantie wordt in het volgende fragment geïllustreerd.[1]

```
print Party[1000].PartyToPartyType.name;
```

Identificerende relaties

Identificerende relaties faciliteren bidirectionele navigatie tussen concepten. Het is mogelijk om via wiskundige functies te navigeren van een instantie van de parent naar een verzameling instanties van de child en andersom van een instantie van de child naar een instantie van de parent.



Afbeelding 1: Niet-identificerende relatie als wiskundige functie.

Afbeelding 2: Twee wiskundige functies.

In afbeelding 2 zijn dus twee wiskundige functies te onderkennen. De ene functie doet een mapping van een Invoice naar een corresponderende verzameling InvoiceItems, terwijl omgekeerd er ook een wiskundige functie is aan te wijzen die een mapping doet van een InvoiceItem naar de Invoice die er bij hoort. Het volgende codefragment toont een aantal navigatie-statements die met afbeelding 3 corresponderen.

```
print Invoice[1000].item[2].description;
print Invoice[1000].item[2].invoice.total;
```

Reïficerende relaties

Reïficerende relaties maken ook bidirectionele navigatie mogelijk. Deze relaties maken het mogelijk om van het ene abstractieniveau te navigeren naar het andere abstractieniveau en ook vice versa. Een concreet voorbeeld maakt reïficatie-relaties iets toegankelijker. Als bijvoorbeeld in SQL een tabel Person wordt gecreëerd, geldt dat de catalogue van de database een tuple zal bevatten die met het concept Person overeenkomt. Deze tuple noemen we de attributie van het concept Person, terwijl de corresponderende SQL-tabel de extensie van het concept Person heet. Dat wil zeggen dat een create statement als

```
create table Person(name char(20), age integer
                                    primary key name);
```

conceptueel gezien de reïficatie (de tot stand brenging) van de extensie en attributie van het concept Person behelst. Hierbij wordt conceptueel gezien een reïficatie-relatie gecreëerd tussen de attributie van Person (aangegeven als rtablex[Person])[2] en zijn extensie (aangegeven door rtablex<Person>). De intensie van het concept Person wordt in dit voorbeeld gevormd door de attributen

name, age en ook de uniciteits-constraint. We voegen nu instanties van het concept Person toe aan onze voorbeeld-database.

```
insert into Person(name, age) { 'Jan', 20};
insert into rtablex<Person>(name, age)
                                    { 'Jannie', 21};
```

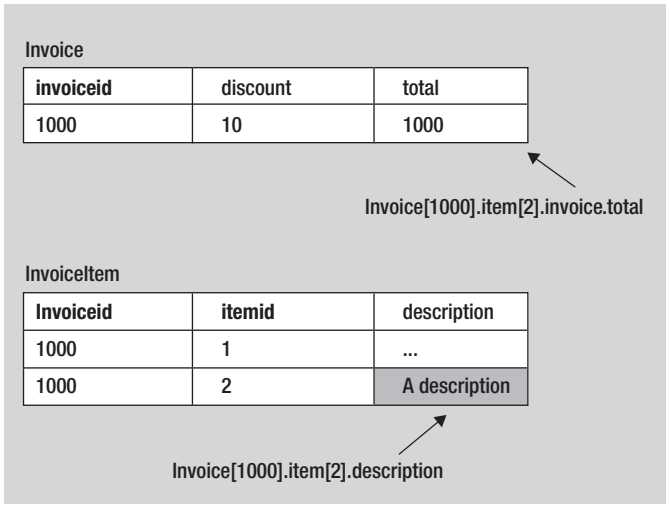
De bovenstaande insert statements hebben tot gevolg dat aan de extensie van rtablex<Person> twee tuples worden toegevoegd. Dit omdat de SQL-tabel Person in deze overeenkomt met de extensie van het concept Person dat door rtablex<Person> wordt aangeduid. Dit heeft tot gevolg dat de volgende statements

```
select * from rtablex<Person>;
select * from Person;
```

beide de volgende relatie opleveren, omdat de tabel Person in deze een alias vormt voor de extensie van het concept Person, in deze met rtablex<Person> aangeduid.

name	age
jan	20
jannie	21

In relationele databases vormen de extensies van concepten, de tabellen zo men wilt, gezamenlijk het bereik van een wiskundige functie waarvan het domein in het bovenstaande voorbeeld gevormd wordt door de extensie van de relatie rtablex. Anders gezegd, rtablex kan worden gezien als een wiskundige functie die een mapping maakt van de attributie van concepten (tuples uit de extensie van rtablex) naar de extensie van deze concepten (de corresponderende tabellen). Buiten de catalogue ondersteunen bestaande database management-systemen en datamanipulatietalen niet of nauwelijks reïficerende relaties. Ook geldt dat reïficatie bij de informatiemodellering in de regel niet wordt toegepast.



Afbeelding 3: Voorbeeld Invoice.

Een voorbeeld met reïficatie

Met een voorbeeld maken we inzichtelijk dat reïficerende relaties in de praktijk een nuttige aanvulling vormen op het conceptuele denkpalet van de informatiemodelleur. Beschouw eens een systeem dat de populatie van verschillende dieren via metingen bijhoudt. Centraal in dit voorbeeld is het hogere orde concept Population. De extensie van dit concept definieert een aantal diersoorten waarvan de populatiegegevens worden bijgehouden. Tegelijkertijd worden voor iedere diersoort de individuele exemplaren bijgehouden. We onderscheiden in dit voorbeeld twee intensies; respectievelijk corresponderend met dieren die door een naam en leeftijd worden gekenmerkt en dieren die door een Id en gewicht worden gekenmerkt.

```
intension idAndWeight {
   id integer;
   weight integer;
   key primary id;
}

intension nameAndAge {
   name char(20);
   age integer;
   key primary name;
}
```

De intensies voor deze soorten dieren zijn in het bovenstaande fragment gedefinieerd. Vervolgens reïficeren we in het volgende fragment de extensie van de concepten Shark en Person volgens de boven gedefinieerde intensies.

```
insert into Population { name('Shark'),
                 description('Lives in the ocean')}
                     with { intension(idAndName); }

insert into Population (name, Description)
                         { 'Horse', 'Lives on land'}
                    with { intension(nameAndAge); }
```

De bovenstaande statements hebben dus de attributie voor de concepten Shark en Horse geïnstantieerd, deze worden respectievelijk met Population[Shark] en Population[Horse] aangeduid. Tegelijkertijd worden door deze statements de corresponderende extensies Population<Shark> en Population<Horse> gecreëerd. De extensie van het concept Population is een normale relationele relatie.

```
select * from Population;
```

Het resultaat van dit statement wordt in de volgende tabel getoond.

name	description
Horse	lives on land
Shark	lives in the ocean

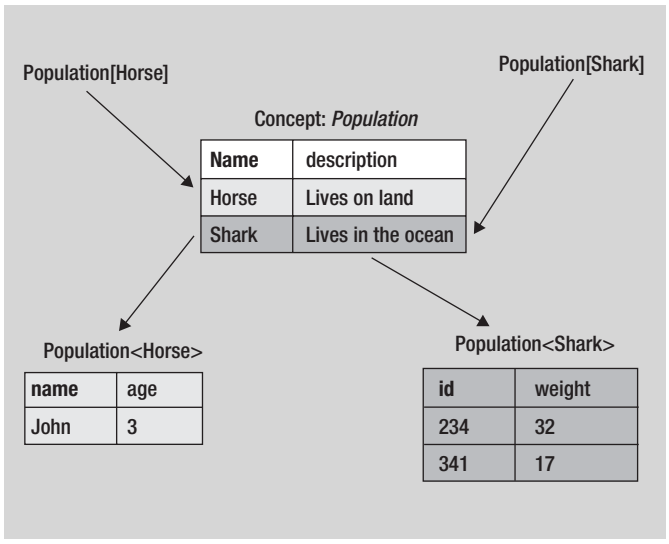
Zoals de volgende statements laten zien kunnen natuurlijk de extensies van de concepten Population<Horse> en Population<Shark> met tuples worden gevuld, zie ook afbeelding 4.

```
insert into Population<Shark>
                          { id(234), weight(32) };
insert into Population<Shark>
                          { id(341), weight(17) };
insert into Population<Horse>
                         { name("John"), age(3) };
```

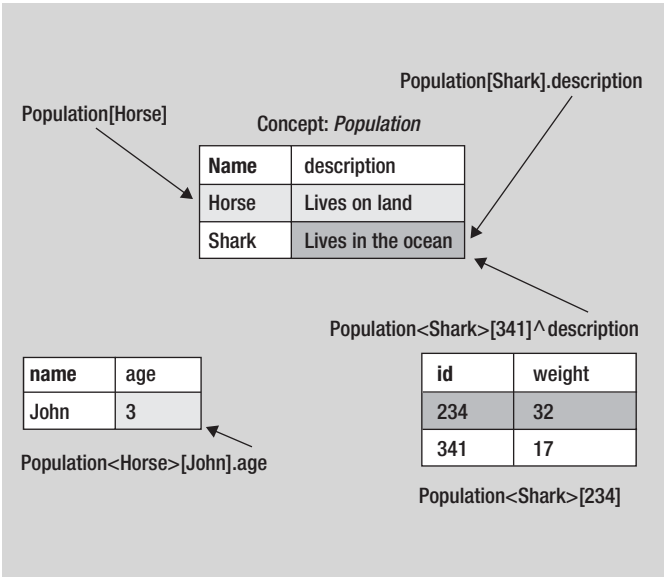
Zoals verwacht kunnen we deze ook weer met select statements uitlezen.

```
select * from Population<Shark>;
```

id	weight
234	32
341	17



Afbeelding 4: Extensies met tuples gevuld.



Afbeelding 5: Voorbeeld Population.

```
select * from Population<Horse>;
```

name	age
John	3

Navigatie van reïficatie-relaties

Zoals gesteld zijn ook reïficerende relaties te zien als directionele wiskundige functies. Deze functies zijn te gebruiken om navigatie over de grenzen van abstractieniveaus te faciliteren. Het volgende statement in de programmeertaal Michelle illustreert de navigatie naar een attribuut van een instantie.

```
print Population<Horse>[John].age;
```

De operator ^ uit het volgende fragment maakt het mogelijk om te navigeren van een element van de extensie van een concept naar een attribuut van de attributie van het concept.

```
print Population<Shark>[234]^description;
```

Afbeelding 5 geeft meer voorbeelden. Het mag evident zijn dat er over dit en verwante concepten meer te zeggen valt; in dit artikel is hiervoor niet de gelegenheid.

Extensionele en intensionele hogere orde concepten

De extensie van het concept Population is dus een normale relationele relatie. Het enige dat bijzonder is, en de reden dat we deze relatie een extensionele hogere orde concept noemen, is dat iedere tuple in de extensie van het concept Population ook de extensie van een ander concept identificeert.

In dit voorbeeld kunnen we dus stellen dat de tuple Population[Shark] het concept Population<Shark> identificeert.[3] Hogere orde concepten op basis van identificerende relaties zijn geïntroduceerd in eerdere artikelen in DB/M (zie het online-archief): 'Orthogonalen; een eerste introductie', DB/M 6, 2005 en 'Hogere orde concepten en hun visualisatie', DB/M 7, 2006. En deze heten voortaan intensionele hogere orde concepten.

Relaties of functies?

Het is belangrijk om te appreciëren dat de analyse en het ontwerp van informatiestructuren in termen van wiskundige functies, *niets* afdoet aan de flexibiliteit die gefaciliteerd wordt door de relationele algebra. Per slot van rekening is iedere wiskundige functie tevens ook een wiskundige relatie; niet waar?
Langs de ene dimensie structureren we en transformeren we conform relationele beginselen, terwijl we langs de andere dimensie navigeren, modelleren, analyseren en interpreteren door in deze relationele structuren wiskundige functies te herkennen. Dus niet relaties of functies maar eerder relaties en functies. Het is belangrijk om hierbij een struikeling te vermijden. De voordelen van de relationele algebra en relationele structuren zijn dus synergetisch met de functionele invalshoek die bij het ontwerp, de analyse, interpretatie en navigatie van informatiestructuren van pas komt. Een aardige bijkomstigheid van de functionele invalshoek is tevens dat het de ondersteuning van extensionele en verschillende soorten intensionele hogere orde concepten faciliteert.[4]

Conclusies

Via een toelichting op de functionele navigatiemogelijkheden die door respectievelijk niet-identificerende en identificerende relaties worden geboden, zijn reïficerende relaties geïntroduceerd. Aannemelijk is gemaakt dat de catalogue van relationele data-bases een voorbeeld van de toepassing van reïficatie in relationele databases vormt.

Literatuur

E.F. Codd [1979], 'Extending the Database Relational Model to Capture more Meaning'.

Noten

1. *De voorbeelden zijn in de taal Michelle geschreven, die wordt gebruikt om onderzoek te doen naar hogere orde concepten en hun toepassing.*
2. *rtablex is in dit artikel de naam van de metarelatie die een tuple bevat voor alle relaties in de database.*
3. *Zie hier een additionele grond om samen met Codd tuple identity te onderkennen.*
4. *Een uitwijding over andere soorten intensionele hogere orde concepten is voor een toekomstige gelegenheid gepland.*

Maurice Gittens is zelfstandig IT-consultant.