Informatiesystemen 1

Thom Wiggers
s4119444

27 januari 2013

Inhoudsopgave

1 Huishoudelijke mededelingen

Dit project maak ik individueel.

2 Taak 1 - ORC vs. SQL

2.1 Inleiding

ORM is een methode om door middel van modellen systemen te ontwikkelen waarbij gepoogd wordt zo min mogelijk fouten toe te laten en zo veel mogelijk redundantie in de opgeslagen gegevens te voorkomen.

ORM bestaat voornamelijk uit een verzameling afspraken over taalgebruik en notaties. Hierdoor zou een goed model ook voor niet-domeinexperts leesbaar en begrijpbaar moeten zijn.

Hoewel ORM modellen vooral worden omgezet naar klassieke relationele (SQL)-databases, is het ook mogelijk om direct 'vragen' te stellen aan een dataset in ORM. Deze querytaal staat bekend als Object-Role Calculus (ORC).

Ik ga hier proberen ORC te vergelijken met de SQL taal, door middel van het vergelijken van enkele verschillende queries, zoekvragen, waarbij ik ook in ga op de fundamentele verschillen tussen de twee verschillende systemen. Hiervoor zal ik een voorbeeldsysteem beschrijven, zowel uitgevoerd in ORM als draaiende op de populaire relationele database postgreSQL.

2.2 Systeem

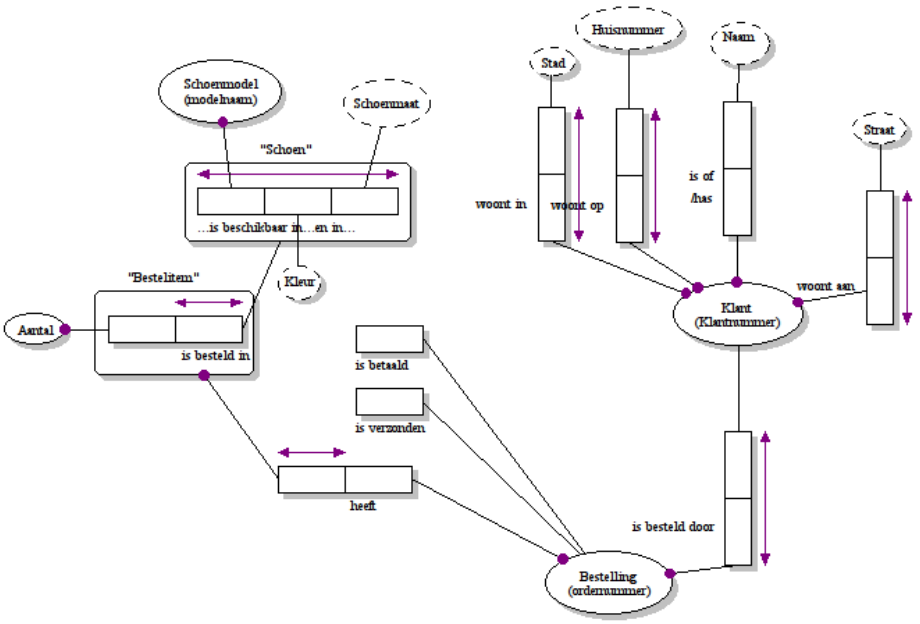
Ik ga hier een voorbeeldsysteem beschrijven van een webwinkel waar men schoenen verkoopt. In deze webwinkel houdt men bestellingen bij, en profielen van klanten. Bestellingen kunnen bestaan uit een of meerdere paren schoenen, in verschillende maten en aantallen.

2.2.1 ORM

```
Schoen: Schoenmodel (modelnaam) is beschikbaar
    in Schoenmaat en is beschikbaar in Kleur.
Bestelitem: Schoen is besteld in aantal.
Bestelling (ordernummer) heeft Bestelitem.
Bestelling (ordernummer) is besteld door
   Klant (klantnummer).
Bestelling is betaald.
Bestelling is verzonden.
Klant (klantnummer) woont aan
   Straat (straatnaam).
Klant (klantnummer) woont op
   Huisnummer (nr).
Klant (klantnummer) woont in Stad (naam).
Klant (klantnummer) heet Naam.
```

2.2.2 SQL Tabellen

Het transformeren van het ORM model naar SQL tabellen is redelijk eenvoudig, maar om dubbel voorkomende gegevens te voorkomen heb ik op verschillende



Figuur 1: ORM Model van het voorbeeldsysteem

plaatsen een identificatienummer aan tabellen toegevoegd als vervangende primary key.

Hier wordt al een duidelijk nadeel van SQL-gebaseerde databases zichtbaar: het is niet mogelijk om gegevens te objectiviseren.

Zie hiervoor de tabellen 1, 2, 3, 4 en 5.

id	**Schoenmodel**	**Kleur**	**Schoenmaat**
1	Model a	Zwart	43
2	Model a	Zwart	42
3	Model a	Wit	43
...	...	...	...

Tabel 1: Schoen tabel

id	schoenid	Aantal
1	2	1
2	2	2
3	3	1
...	...	...

Tabel 2: Bestelitem Tabel

Bestelnummer	Bestelddoor	Verzonden	Betaald
1	1	Y	Y
2	1	N	Y
3	2	Y	N
...	...	...	...

Tabel 3: Bestellingen

Bestelitem	Bestelling
1	1
2	1
3	2
...	...

Tabel 4: BestelitemBestelling: Bestelde items horende bij bestellingen

Klantnummer	Naam	Straat	Huisnummer	Stad
1	John Doe	Heyendaalseweg	91	Nijmegen
2	Steve Foo	Asselsestraat	34	Apeldoorn
3	Jane Bar	Kanaalstraat	33	Amsterdam
...	...	...	...	...

Tabel 5: Klant tabel

2.3 Eenvoudige gegevens uit het systeem halen

2.3.1 SQL

Eenvoudige gegevens uit het informatiesysteem halen is eenvoudig in SQL. Een `SELECT` statement is erg eenvoudig voor elkaar te krijgen. Bijvoorbeeld het selecteren van alle verschillende schoenen die in de winkel te koop worden aangeboden:

```
SELECT Schoenmodel, Kleur, Schoenmaat
FROM Schoen;
```

2.3.2 ORC

In ORC is het ook eenvoudig om dezelfde gegevens op te halen:

```
Schoenmodel, Kleur, Schoenmaat FROM Schoen
```

Het verschil tussen ORC en SQL is hier niet zo groot. SQL heeft `SELECT`, maar verder zou men bijna copy/paste kunnen doen.

2.4 Conditioneel gegevens uit het systeem halen

Men wil niet altijd alle gegevens uit een informatiesysteem hebben. Daarom is het in SQL en in ORC mogelijk om condities op te geven waaraan de op te vragen informatie moet voldoen.

2.4.1 SQL

Stel, ik wil alle verschillende modellen van zwarte schoenen hebben in maat 43.

```
SELECT Schoenmodel
FROM Schoen
WHERE Schoenmaat = '43'
  AND Kleur='Zwart';
```

2.4.2 ORC

In ORC gaat het zo:

```
Schoenmodel FROM Schoen in Kleur 'Zwart'
  AND in Schoenmaat '43'
```

Ook hier is er een grote overeenkomst tussen SQL en ORC. SQL heeft hier het `WHERE` keyword om onderscheid te maken tussen de condities en de tabel, maar verder is het grotendeels hetzelfde.

2.5 Complexe informatie uit het systeem halen

Stel, ik wil weten uit welke steden de mensen komen die zwarte schoenen besteld hebben.

2.5.1 SQL

```
SELECT stad
FROM Klant
WHERE Klant.Klantnummer IN
  (SELECT Bestelddoor
   FROM   Bestellingen
   JOIN   BestelitemBestelling
   ON   Bestelling = Bestelnummer
   WHERE  Bestelitem IN
     (SELECT id
      FROM Bestelitem
      WHERE Schoenid IN
        (SELECT id
         FROM Schoen
         WHERE Kleur = 'Zwart')
     )
  )
```

Dit is duidelijk een erg complexe query, omdat men vele tabellen door moet omdat de koppeling tussen de informatie vrij zwak is in Relationele Databases.

2.5.2 ORC

```
Stad woonplaats van Klant
  die besteld heeft Bestelling
    die Bestelitem heeft
    met Schoen in kleur Zwart
```

De ORC query is hier duidelijk een stuk eenvoudiger. Dit komt doordat in een ORM-schema de relaties tussen verschillende gegevens een stuk duidelijker is edefineerd. Waar SQL steeds in verschillende lijsten moet zoeken die allemaal apart gemaakt dienen te worden, is het hier iets wat impliciet gebeurt.

2.6 Conclusie

Waar ORC en SQL vaak erg op elkaar lijken, is ORC soms in staat om veel duidelijker queries op te stellen, vooral wanneer het om complexe informatie gaat. Relationele Databases zijn minder sterk in het weergeven van de verbanden tussen informatie, en er moeten vaak extra gegevenstypen geintroduceerd worden (identificatienummers bijvoorbeeld) om dataduplicatie te voorkomen.

3 Taak 2 - Typegerelateerdheid

3.1 Inleiding

Omdat ik denk dat dit een belangrijk onderdeel is van de stof, en veel van de problemen behandeld in het college een oplossing hebben die steeds subtiel is en niet per se direct voor de hand ligt, ga ik hier proberen een paar extra voorbeelden te maken en uit te werken, om zo mijn begrip van typegerelateerdheid wat scherper te krijgen.

Ik ga proberen om de afleidingsregels zoals vermeld in het dictaat [1, p. 41] opnieuw af te leiden.

3.2 Regels

3.2.1 T1

$$\vdash x \sim x$$

waar $\sim$ typegerelateerd betekent.

Type x is vanzelfsprekend typegerelateerd met zichzelf. De typegerelateerdheidsrelatie is *reflexief*.

3.2.2 T2

$$x \sim y \vdash y \sim x$$

Typegerelateerdheid is ook een *symmetrische* relatie.

3.2.3 T3

$$\sqcap(x) = \sqcap(y) \wedge y \sim z \vdash x \sim z$$

waar $\sqcap(x)$ de *Pater familias* van x is.

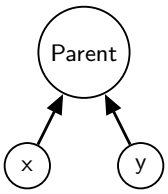
Dit betekent dat als x en y specialisaties van hetzelfde type zijn, en y is typegerelateerd aan z, x ook typegerelateerd is aan z. Zie hiervoor ook figuur 2. Elementen in "Parent" ($\sqcap(x)$ en $\sqcap(y)$) komen ook voor in x en in y, daar zij specialisaties van "Parent" zijn, ze hebben grotendeels dezelfde eigenschappen, alleen een paar meer. Elementen die typegerelateerd zijn aan "Parent" en aan één van de subtypes van "Parent" zijn dus ook typegerelateerd aan de andere subtypes.

Er is ook het bijzondere geval waarbij $y = z$, waarmee bewezen kan worden dat $x \sim y$.

3.2.4 T4

$$x \,\mathrm{Gen}\, y \wedge y \sim z \vdash x \sim z$$

waar $x \,\mathrm{Gen}\, y$ betekent "x is een generalisatie van y".



Figuur 2: Specialisatie

Op dezelfde manier als bij T3 (3.2.3), komen elementen van generalisaties voor in het gegeneraliseerde type. Typen die gerelateerd zijn aan een bepaald type, zijn dus ook gerelateerd aan de generalisaties van dat type.

Een variant van deze waarbij $y = z$ bewijst dat $x \operatorname{Gen} y \vdash x \sim y$.

3.2.5 T5

$$x, y \in \mathcal{G} \wedge \operatorname{Elt}(x) \sim \operatorname{Elt}(y) \vdash x \sim y$$

Als x en y powertypen zijn, en hun elementen typegerelateerd, dan zijn deze powertypen, verzamelingen van hun elementtypen, natuurlijk ook gerelateerd.

3.2.6 T6

$$x, y \in \mathcal{S} \wedge \operatorname{Elt}(x) \sim \operatorname{Elt}(y) \vdash x \sim y$$

Zoals bij T5 (3.2.5), zijn als de elementen van sequentietypen typegerelateerd zijn, de sequentietypen zelf ook typegerelateerd.

3.2.7 T7

$$\mathcal{O}_x = \mathcal{O}_y \vdash x \sim y$$

Als het objecttype van x en die van y dezelfde zijn, zijn x en y typegerelateerd.

Figuur 3: Generalisatie

3.3 Voorbeeld

Ik gebruik hier [1, figuur 2.26] uit het dictaat. Ik ga de bewering dat alle typen behalve C en B, C en E, en C en F, typegerelateerd zijn toetsen.

$$
\begin{aligned}
A \operatorname{Gen} B &\vdash A \sim B \\
A \operatorname{Gen} C &\vdash A \sim C \\
\sqcap(D) = \sqcap(A) \wedge A \sim C &\vdash D \sim C \\
\sqcap(D) = \sqcap(A) \wedge A \sim B &\vdash D \sim B \\
\sqcap(A) = \sqcap(D) \wedge D \sim D &\vdash A \sim D \\
B, E \in \mathcal{G} \wedge A \sim D &\vdash B \sim E \\
F, B \in \mathcal{G} \wedge A \sim A &\vdash F \sim B \qquad (B \in A) \\
\sqcap(D) = \sqcap(E) \wedge D \sim A &\vdash E \sim A \\
F, B \in \mathcal{G} \wedge E \sim A &\vdash F \sim A \\
\sqcap(D) = \sqcap(E) \wedge D \sim D &\vdash D \sim E \\
E, F \in \mathcal{G} \wedge D \sim E &\vdash E \sim F \\
\sqcap(D) = \sqcap(F) \wedge D \sim A &\vdash D \sim F
\end{aligned}
$$

3.4 Conclusie

Typegerelateerdheid is iets wat op een redelijk intuïtief vlak werkt, maar ook een aantal harde regels kent. Veel van die regels zijn makkelijk beredeneerbaar, zoals bijvoorbeeld typegerelateerdheid van powertypen, en sommige anderen zijn iets moeilijker om te onderbouwen. Het consequent toepassen van die regels is echter vaak een flinke uitdaging, en waar het vaak makkelijk is om te zeggen "x is gerelateerd aan y, want zus en zo", is dit opschrijven volgens de bovengenoemde regels (3.2) vaak een uitdaging.

4 Taak 3 - Constraints en populeerbaarheid

4.1 Inleiding

In deze taak probeer ik wat ingewikkelder contraints en restricties op domeinmodellen te vinden, en daar wat voorbeelden bij te maken en uit te werken. In het bijzonder ga ik het deze keer proberen op te schrijven zonder gebruik te maken van afbeeldingen[1]. Ik ga ook proberen verschillende constraints in een model te gebruiken en kijk naar de populeerbaarheid.

[1] Ik maak wel gebruik van schema's in mijn notities, omdat het bedenken van een kloppend schema door willkeurig opschrijven van symbolen niet echt doenbaar is.

4.2 Systeem 1

$$
\begin{aligned}
\mathcal{P} &= \{p1, p2, p3, p4\} \\
\mathcal{E} &= \{A, B, C\} \\
\mathcal{F} &= \{f, g\} \\
\mathcal{G} &= \emptyset \\
\mathcal{S} &= \emptyset \\
\text{Spec} &\Rightarrow B \,\text{Spec}\, A \\
f &= \{p1, p2\} \\
g &= \{p3, p4\} \\
\text{Base}(p1) &= A \\
\text{Base}(p2) &= B \\
\text{Base}(p3) &= A \\
\text{Base}(p4) &= C \\
\text{total}(\tau) &\text{ where } \tau = \{p1\} \\
\text{total}(\tau) &\text{ where } \tau = \{p2\}
\end{aligned}
$$

Instanties van A moeten in f een rol spelen (total($p1$)) en ook alle elementen in B moeten dat (total($p2$)). B is daarnaast een specialisatie van A, met als gevolg dat instanties van B ook in A voorkomen. Dit betekent echter niet dat de populaties van A en B gelijk moeten zijn: B kan een subset zijn van de instanties van A. Een voorbeeld zou zijn dat A studenten betekent, en B studentmentoren.

Dit systeem is Globaal Objectpopuleerbaar: want er is een voorbeeld populatie:

$$
\begin{aligned}
\text{Pop}(A) &= \{a1, a2\} \\
\text{Pop}(B) &= \{a2\} \\
\text{Pop}(C) &= \{c\} \\
\text{Pop}(f) &= \{\{a1, a2\}, \{a2, a2\}\} \\
\text{Pop}(g) &= \{\{a1, c\}\}
\end{aligned}
$$

Doordat dit model Globaal Objectpopuleerbaar is, is het direct ook Globaal Atomair populeerbaar, Lokaal Atomair populeerbaar, en Lokaal Objectpopuleerbaar. Voor populaties in de bovenstaande klassen kan men hetzelfde voorbeeld gebruiken.

Stel we zouden de constraint unique($\{p1\}$) $\wedge$ unique($\{p2\}$) toevoegen. Dan zou er een 1-op-1 mapping van A op B ontstaan, waaraan door de totale rol

constraints alle elementen van A en B mee moeten doen. Dan creëert men dus een situatie waarin $\mathrm{Pop}(A) = \mathrm{Pop}(B)$.

Als men in dit systeem $\mathrm{total}(p1)$ zou weglaten, en $\mathrm{unique}(\{p2, p4\})$ toevoegt, dan ontstaat een leuk nieuw systeem, waarin men $\xi(\{f, g\})$ kan doen.

$$\xi(\tau) = \sigma_{p1=p3}(f \bowtie g)$$

Hierdoor krijgt men het volgende systeem:

$$\begin{aligned}
\mathcal{P} &= \{p1, p2, p4\} \\
\mathcal{E} &= \{A, B, C\} \\
\mathcal{F} &= \{f \bowtie g\} \\
\mathrm{Spec} &\Rightarrow B \,\mathrm{Spec}\, A \\
f \bowtie g &= \{p1, p2, p4\} \\
\mathrm{Base}(p1) &= A \\
\mathrm{Base}(p2) &= B \\
\mathrm{Base}(p3) &= C \\
\mathrm{unique}(\tau) &\text{ where } \tau = \{p2, p4\}
\end{aligned}$$

Uiteraard is dit systeem Globaal Objectpopuleerbaar:

$$\mathrm{Pop}(A) = \{a\},\ \mathrm{Pop}(B) = \{b\},\ \mathrm{Pop}(C) = \{c\}$$

4.3 Systeem 2

In dit onderdeel gebruik ik het volgende systeem:

$$\begin{aligned}
\mathcal{P} &= \emptyset \\
\mathcal{E} &= \{A, B, C\} \\
\mathcal{F} &= \emptyset \\
\mathcal{G} &= \{A, C\} \\
\mathcal{S} &= \emptyset \\
\mathrm{Spec} &\Rightarrow \mathcal{G}(A) \,\mathrm{Spec}\, C \\
&\Rightarrow B \,\mathrm{Spec}\, A \\
\mathrm{Gen} &\Rightarrow C \,\mathrm{Gen}\, B \\
\mathrm{total}(\tau) &\text{ where } \tau = \{\mathcal{G}(C)\}
\end{aligned}$$

Populeerbaarheid van dit systeem is gemakkelijk, men stopt gewoon overal een item in. Alle objecttypen zijn wel typegerelateerd, door de vele generalisaties en specialisaties.

5 Taak 4 - Metamodel van XML

5.1 Inleiding

In deze taak ga ik ter oefening een metamodel en een grammatica maken voor XML.

> Extensible Markup Language (XML) is een standaard van het World Wide Web Consortium voor de syntaxis van formele opmaaktalen waarmee men gestructureerde gegevens kan weergeven in de vorm van platte tekst. Deze presentatie is zowel machineleesbaar als leesbaar voor de mens. Het XML-formaat wordt gebruikt om gegevens op te slaan (zoals in het OpenDocument-formaat) en om gegevens over het internet te versturen (zoals in AJAX).[2]

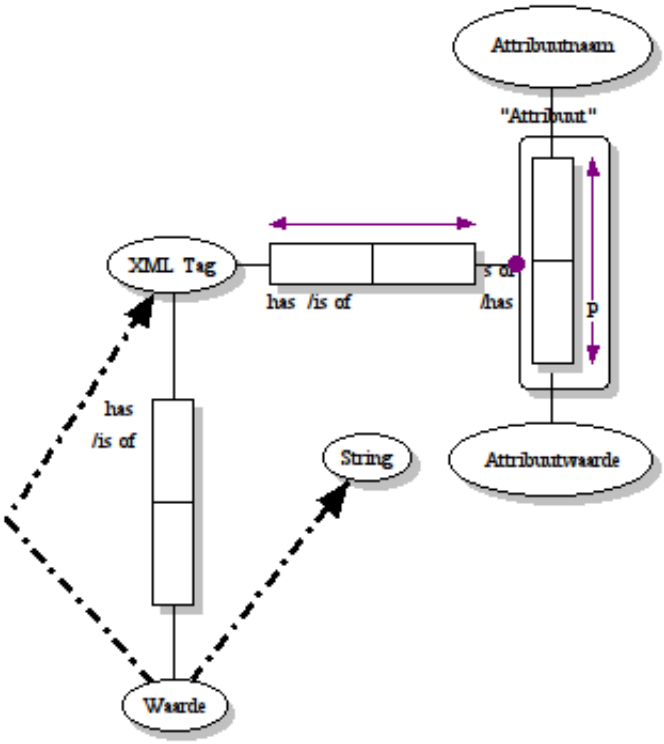
5.2 Aannames en definities

Ik laat bij de definitie van XML DTDs weg, en ga uit van elementen die attributen hebben met bepaalde waarden, en die elementen hebben zelf nul, een of meer waarden, die eventueel elementen zijn. Een XML-document moet precies 1 root-element bevatten, welke alle andere elementen bevat.

```
<element attribuut="waarde">
  foo
</element>
```

5.3 Schema

Een XML-document bevat XML-tags, die attributen hebben en een waarde kunnen bevatten. Die waarde kunnen zelf weer XML-tags zijn, of Strings. De waarde is dus een generalisatie van Strings en van Tags. Attributen kunnen niet op zichzelf staan, en zijn naam-waarde paren. Attributen zijn dus een complex object. Een tag kan meer dan een waarde bevatten en volgorde maakt uit. Daarom is Waarde een sequencetype. Dat is echter niet iets wat Visiomodeler ondersteunt, en daarom is het niet opgenomen in afbeelding 4.



Figuur 4: Metamodel voor XML

5.4 Grammatica

$$
\begin{aligned}
S &: tag \\
tag &: <tagnaam\ attributen>waarde</tagnaam> \\
attributen &: attribuutnaam="attribuutwaarde" \\
attribuutnaam &: naamstring \\
attribuutwaarde &: string \\
waarde &: tag|string \\
tagnaam &: naamstring \\
naamstring &: a-z|- \\
string &: a-z|A-Z|0-9|\dots
\end{aligned}
$$

5.5 Conclusie

XML is eenvoudig te verwerken door andere systemen: het bevat maar weinig "rare dingen". Dit is opnieuw gebleken in deze taak: XML is in weinig velden op te slaan, en gemakkelijk te parseren met een grammatica.

6 Taak 5 - Oefening

6.1 Inleiding

In deze taak ga ik het tentamen oefenen door het tentamen van 2011–2012[3] uit te pluizen.

6.2 Opgave 1

6.2.1 a

> Teken een voorbeeld van een informatiestructuur $\mathcal{I}$ waarin het aantal objecttypen $|\mathcal{O}| = 5$. Geef daarna de volledige definitie van die $\mathcal{I}$.[3]

Deze opdracht is behoorlijk vrij. Aangezien er geen eisen zijn anders dan $|\mathcal{O}| = 5$, voldoen 5 objecten zonder relaties al.

$$
\begin{aligned}
\mathcal{E} &= \{A, B, C, D, E\} \\
\mathcal{P} &= \emptyset \\
\mathcal{F} &= \emptyset \\
\mathcal{G} &= \emptyset \\
\mathcal{S} &= \emptyset
\end{aligned}
$$

Figuur 5: Opdracht 1 a

Er zijn onnoemelijk veel oplossingen voor deze vraag: namelijk zijn feittypen $\mathcal{F}$ ook onderdeel van $\mathcal{O}$. Verder kunnen er in plaats van entiteittypen, zoals in mijn oplossing, ook Labeltypen gebruikt worden.

6.2.2 b

> In een informatiestructuur ontdekken we een gedeelte waarin objecttypen x en y voorkomen met $y \,\mathrm{Gen}\, x$ en $\mathrm{Elt}(y) = x$. Is dat toegestaan? [3]

Het antwoord uit deze vraag hangt op de definitie van $\mathrm{Elt}(y)$. $\mathrm{Elt}(y)$ geeft het elementtype van powertype y terug, wat dus betekent dat in deze vraag y een powertype over x is.

Generalisaties ontlenen hun identificatie aan het type waarvan gegeneraliseerd wordt. Maar omdat y als powertype toch al van x afhangt, is dat geen probleem. De gegeven structuur is dus toegestaan.

6.2.3 c

> In een informatiestructuur ontdekken we een gedeelte waarin de objecttypen x en y voorkomen met $x \,\mathrm{Spec}\, y$ en $y \in \mathcal{F}$. Is dat toegestaan? [3]

y is dus een feittype dat als objecttype gebruikt wordt en wat vervolgens gespecialiseerd wordt. Maar dat is gewoon toegestaan: $\mathcal{F} \subseteq \mathcal{O}$, en $\mathrm{Spec} \subseteq \mathcal{E} \times \mathcal{O}$. Als de relatie omgekeerd was, dan was het niet toegestaan: naar feittypen kan men niet specialiseren, enkel naar entiteittypen.

6.2.4 d

> Laat $\mathcal{I}$ een informatiestructuur zijn met feittypen $\mathcal{F} = \{\{p_1, p_2\}, \{p_3, p_4\}\}$ waarbij $\mathrm{Base}(p_1) = \mathrm{Base}(p_3) = A$ en $\mathrm{Base}(p_2) = \mathrm{Base}(p_4) = B$. Veronderstel dat we de constraints enumeration$(A, \{a\})$ en exclusion$(\{p_1\}, \{p_3\})$ hebben. Welke van de volgende eigenschappen zijn nu geldig:
>
> i. Het schema is lokaal-atomair populeerbaar.
>
> ii. Het schema is globaal-atomair populeerbaar.
>
> iii. Het schema is lokaal-object populeerbaar.
>
> iv. Het schema is globaal-object populeerbaar.
>
> [3]

De enumeration constraint zorgt ervoor dat A alleen het object a bevat. Verder mogen p_1 en p_3 niet hetzelfde element bevatten. Globale objectpopuleerbaarheid is dus in ieder geval onhaalbaar: er ontstaat dan een illegale populatie waarin $p_1 = p_3$.

Lokaal-atomaire populeerbaarheid is wel haalbaar: $\text{Pop}(A) = \{a\}, \text{Pop}(B) = \{b\}$. Dit is ook direct een populatie die globaal-atomaire populeerbaarheid aantoont. Voor lokaal-object populeerbaarheid kan men de populaties $\text{Pop}(A) = \{a\}, \text{Pop}(B) = \{b\}$ gebruiken, en dan ofwel $\text{Pop}(\{p_1, p_2\}) = \{a, b\}$, ofwel $\text{Pop}(\{p_3, p_4\}) = \{a, b\}$.

6.3 Opgave 2

6.3.1 a

> Laat $\mathcal{I}$ een informatiestructuur zijn met feittypen $\mathcal{F}$ en predicatoren $\mathcal{P}$. Is het nu mogelijk dat $|\mathcal{F}| = |\mathcal{P}|$? [3]

Ja, bijvoorbeeld een $\mathcal{I}$ waarin $\mathcal{F} = \{\{p_1\}\}$ en $\mathcal{P} = \{p_1\}$. Hier bevat $\mathcal{F}$ enkel een unair feittype.

6.3.2 b

> Wanneer zijn twee feittypen type-gerelateerd? Licht je antwoord toe met behulp van een kleine informatiestructuur. [3]

Twee feittypen zijn gerelateerd als hun Base gerelateerd zijn. Dit is dus bijvoorbeeld in de structuur van vraag 1d in paragraaf 6.2.4. Daar zijn de twee feittypen typegerelateerd, omdat $\vdash x \sim x$.

6.3.3 c

> Laat x een powertype zijn en laat $f = \{p_1, p_2\}$ een feittype zijn met $\text{Base}(p_1) = x$ en $\text{Base}(p_2) = y$. Verder geldt het volgende:
>
> i. Alle predicatoren zijn totaal en uniek.
>
> ii. Objecttype y is identificeerbaar.
>
> Is x nu identificeerbaar? [3]

x is identificeerbaar, omdat y dat is, en er door de totale rollen en uniqueness een 1-op-1 relatie tussen elke x en een identificeerbare y is.

6.3.4 d

> Laat $f = \{p_1, p_2\}$ een feittype zijn met $\text{Base}(p_1) = A$ en $\text{Base}(p_2) = B$ waarbij B een schematype is. Veronderstel dat de informatiestructuur van schematype B twee objecttypen bevat:

$$\mathcal{O}_B = \{C, D\}$$

Geef een definitie vam de gehele populatie Pop waarbij $|\,\mathrm{Pop}(f)| = 2$. Bij de definitie van deze populatie is het de bedoeling dat je de formele notatie gebruikt. [3]

$$\begin{aligned}
\mathrm{Pop}(A) &= \{a_1, a_2\} \\
\mathrm{Pop}(C) &= \{c_1\} \\
\mathrm{Pop}(D) &= \{d_1\} \\
\mathrm{Pop}(f) &= \{\{p_1 : a_1, p_2 : \{c_1, d_1\}\}, \{p_1 : a_2, p_2 : \{c_1, d_1\}\}\}
\end{aligned}$$

6.4 Opgave 3

6.4.1 a

Laat $\tau = \{p_1, p_2\}$ een verzameling predicatoren zijn, zodanig dat:

$$\mathrm{Fact}(p_1) \neq \mathrm{Fact}(p_2) \text{ en } \mathrm{Base}(p_1) = \mathrm{Fact}(p_2)$$

Geef een voorbeeldschema waarin de constraint unique(τ) voorkomt. Is die constraint syntactisch correct? Zo ja, geef dan de bijbehorende semantiek van de constraint. Zo nee, geef dan aan hoe de fout verbeterd kan worden. [3]

$\mathrm{Base}(p_1) = \mathrm{Fact}(p_2)$ betekent dat p_2 in een feittype zit, waarmee p_1 een relatie heeft. De constraint unique(τ) is niet syntactisch correct: dit omdat p_2 in p_1 zit. Deze fout zou opgelost kunnen worden door p_1 aan de andere kant van zijn feittype te zetten, en dus niet meer $\mathrm{Base}(p_1) = \mathrm{Fact}(p_2)$.

6.4.2 b

Wanneer noemen we een datamodel conceptueel? Gebruik bij je antwoord een isomorfisme. [3]

Een conceptueel model is een datamodel dat slechts het universe of discourse modelleert en niets anders. Als een datamodel conceptueel is, behandelt het slechts het datamodel, en andersom.

6.4.3 c

We bekijken een informatiestructuur $\mathcal{I}$ met vier verschillende objecttypen x, y, z, f waarbij $f = \{p_1, p_2\}$ het enige (niet impliciete) feittype is. Laat x een powertype zijn met elementtype y. Laat $\text{Base}(p_1) = y$ en $\text{Base}(p_2) = z$. Naast de vier genoemde objecttypen, is er ook nog het impliciete feittype van x, dat is $\in_x = \{\in_x^e, \in_x^p\}$. Geef een populatie van $\mathcal{I}$, waarin elk objecttype tenminste 1 instantie heeft, zodanig dat de constraint $\text{subset}(\{p_1\}, \{\in_x^e\})$ wordt overtreden. Bij de definitie van deze populatie moet je de formele notatie gebruiken. [3]

$$\begin{aligned}
\text{Pop}(x) &= \{\{y_1\}\} \\
\text{Pop}(y) &= \{y_1, y_2\} \\
\text{Pop}(\in_x) &= \{\{y_1, \{y_1\}\}\} \\
\text{Pop}(z) &= \{z_1\} \\
\text{Pop}(f) &= \{\{y_2, z_1\}\}
\end{aligned}$$

Nu geldt $p_1 \not\subset \in_x^e$.

6.4.4 d

Laat $\mathcal{I}$ een informatiestructuur zijn met een atomair objecttype A en twee feittypen $f = \{p_1, p_2\}$ en $g = \{p_3, p_4\}$ waarbij $\text{Base}(p_i) = A$ voor alle $1 \leq i \leq 4$. Welke total role constraints in $\mathcal{I}$ zijn nu mogelijk? [3]

Er zijn geen andere constraints in $\mathcal{I}$, wat dus betekent dat er op elke predicator een total role constraint kan liggen. Verder kunnen er nog op elke combinatie van predicatoren uit beide feittypen een total role constraint liggen. Dit omdat ze allen van hetzelfde type zijn.

6.5 Opgave 4

6.5.1 a

Waarom heeft een sequentietype twee impliciete feittypen nodig? Licht je antwoord toe met een kleine voorbeeldpopulatie. Bij de definitie van deze populatie is het de bedoeling dat je de formele notatie gebruikt. [3]

Sequentietypen zijn zoals powertypen, maar met de toevoeging dat volgorde uitmaakt. Daarom is er de tweede impliciete predicator $@_x = \{@_x^s, @_x^i\}$.

$$\begin{aligned}\mathcal{E} &= \{x\}\\ \mathcal{S} &= \{y\}\\ \text{Pop}(x) &= \{a\}\\ \text{Pop}(y) &= \{\{a\}\}\\ \text{Pop}(\in_y) &= \{a, \{a\}\}\\ \text{Pop}(@_y) &= \{\{a, \{a\}\}, 1\}\end{aligned}$$

6.5.2 b

> In een metamodel kan een feittype Gen = $\{p_1, p_2\}$ voorkomen met Base(p_1) = $\mathcal{E}$ en Base(p_2) = $\mathcal{O}$, waarbij $\mathcal{E}$ entiteittypen zijn en $\mathcal{O}$ objecttypen. Omdat in het metamodel tevens $\mathcal{O}\,\text{Gen}\,\mathcal{E}$ geldt, moet het mogelijk zijn om het genomede feittype daarmee te populeren. Laat nu $t \in$ Pop(Gen) een instantie van het feittype Gen zijn, waarmee we $\mathcal{O}\,\text{Gen}\,\mathcal{E}$ willen specificeren. Geef de definitie van t. Bij de definitie van deze populatie is het de bedoeling dat je de formele notatie gebruikt. [3]

Eerst moeten we vaststellen wat $\mathcal{O}$ en $\mathcal{E}$ zijn. Dit zijn de objecttypen, die de instanties van $\mathcal{O}$ en $\mathcal{E}$ bevatten. De $\mathcal{O}$ en $\mathcal{E}$ in $\mathcal{O}\,\text{Gen}\,\mathcal{E}$ zijn instanties in het metamodel, en in dit geval is $\mathcal{O}$ een instantie van $\mathcal{O}$ en $\mathcal{E}$ een instantie van $\mathcal{E}$. Dus $t = \{\mathcal{E}, \mathcal{O}\}$.

6.5.3 c

> Laat $p_1 \circ p_2 \circ p_3^{\leftarrow}$ een padexpressie zijn met als semantiek $\{(a, b)\}$. Geef een informatiestructuur met populatie, zodanig dat de gegeven padexpressie inderdaad de gegeven semantiek heeft. Bij de definitie van de populatie hoef je niet de formele notatie te gebruiken, maar je moet wel de stapsgewijze evaluatie van de padexpressie uitwerken. [3]

Opvallend aan deze expressie is dat er tweemaal een feittype binnen wordt gegaan. Dit kan dus niet anders dan een genest feittype betekenen.

Populaties:

$$\begin{aligned}\text{Pop}(\{p_1, p_4\}) &= \{a, x\}\\ \text{Pop}(\{p_2, p_3\}) &= \{\{a, x\}, b\}\end{aligned}$$

Evaluatie:

$$p_1 \Rightarrow \{a, \_\}$$
$$p_1 \circ p_2 \Rightarrow \{\{a, \_, \_\}$$
$$p_1 \circ p_2 \circ p_3^{\leftarrow} \Rightarrow \{a, b\}$$

6.5.4 d

Laat G de grammatica zijn met de volgende regels:

$$R_1 : X \to yY$$
$$R_2 : Y \to xX$$
$$R_3 : Y \to Y$$

> Bepaal welke feittypen er in het schema $\Delta(G)$ moeten komen. Geef dan het hele schema en bepaal of dat schema identificeerbaar is.

Voor elk terminale symbool is er een entiteittype met labeltype nodig, en voor elk productiesymbool is er een entiteittype nodig. Elke productieregel generaliseert naar het product.

Productieregel R_3 zorgt ervoor dat dit schema niet identificeerbaar is, namelijk is het niet mogelijk om een Y te identificeren die naar zichzelf generaliseert. Het is mogelijk een n aantal stappen door R_3 te doen, maar dat is niet uit elkaar te houden van een schema dat $n + 1$ stappen erdoor doet.

6.6 Conclusie

De meeste stof beheers ik, maar sommige dingen zijn nog wat vaag. Dit tentamen maken heeft wel geholpen met de stof bevatten.

Referenties

[1] Advanced Information Models, Arthur ten Hofstede en Patrick van Bommel, Radboud Universiteit Nijmegen, September 2011.

[2] Extensible Markup Language, Wikipedia, https://nl.wikipedia.org/wiki/XML, Opgehaald 13 januari 2013.

[3] Tentamen Informatiesystemen (IS1), 2011.