Afstudeer verslag

Semantische webtechnologie &

Federatieve databases

Joris Lops

16-8-2008 Groningen

Dit verslag is het resultaat van mijn afstudeerstage bij Ordina Oracle Solutions te Groningen. Het project is onderdeel van mijn mastersopleiding system en software engineering aan de Rijks Universiteit Groningen. Het project is begeleid door Jelmer Sjollema en Jan van de Ven vanuit Ordina en door Jan Jongejan en Herman Balsters vanuit de universiteit. In dit verslag wordt de bruikbaarheid van semantische webtechnologie geëvalueerd en een methode gedemonstreerd om schema-integratie te realiseren voor federatieve database systemen.

De volgende mensen wil ik bedanken voor hun tijd en hulp

J. Sjollema, J. van der Ven, J. Jongejan, H. Balsters

1 Inleiding 5
2 Opdracht omschrijving 6
3 Kidos 6
4 Ontologie technologie 6
4.1 Inleiding 7
4.2 Semantisch Web 7
4.3 Informatie versus kennis 10
4.3.1 Kennis representatie 11
4.3.2 Levenscyclus van een ontologie 12
4.3.3 RDF & RDFS 13
4.3.4 Resource Description Framework 13
4.3.5 RDF + RDFS 15
4.3.6 SPARQL – Query taal voor kennisdatabase 17
4.3.7 Inferentie 19
4.3.8 Ontologie gecombineerd met database 20
4.3.9 Voorbeeld van database en ontologie gecombineerd 21
4.3.10 Toepassingen 21
4.3.11 Kennis life-cycle 26
4.3.12 Aanverwante technologieën van het Semantisch Web 28
4.3.13 Agency and reasoning 28
4.3.14 Kennis services 28
4.4 Structurering van gegevens 29
4.5 Problemen met ontologie technologie 30
5 Ontologie technologie in het kort 31
5.1 Toepassingsgebieden van ontologieën 32
5.2 De opdracht 33
5.3 Complexiteit 33
5.4 Toepassingen voor Kidos 33
5.5 Conclusie 34
6 Data Integratie 35
7 Ontologie technologie en federatief database systemen. 37
8 Inleiding - Object Role Model 38
8.1 Specificatie van Voorbeeld 39
8.2 Elementaire feiten en de structuur 40
8.3 Afbeelden van Concepten, waarden en rollen 41
8.4 Attribuut of Object 43
8.5 Uitbreiding van ORM schema 44
8.6 Objectificatie 46
8.7 Uniciteitbeperking 47
8.8 Mandatory rol constraints 48
8.9 Generalisatie en specialisatie relatie 49
8.10 Valideren van ORM model 50
8.11 RMAP 51
8.12 Conclusie 51
9 Schema integratie 52
9.1 ORM 52
9.2 Schema Integratie 52
9.3 Plan van aanpak 54
9.4 Schoolresultaten 54

9.5 Reverse Engineering ... 56
9.6 Rmap Inverse ... 58
9.7 Reverse engineering in ORM door middel van patroon herkenning ... 59
9.8 Reverse Engineering van het Kidos Schema ... 64
9.8.1 Stap 1 ... 65
9.8.2 Stap 2 ... 66
9.8.3 Stap 3 en 4 ... 67
9.8.4 Tussenstap – Controleren van equivalentie ... 68
9.8.5 Stap 5 ... 68
9.9 Conclusie ... 69
10 Schema Integratie ... 71
10.1 Bruikbaarheid van Globale Schema ... 73
10.2 Virtuoso Virtual Database – Implementatie van het ontwerp ... 74
10.3 Vervolg ... 75
10.4 Conclusie ... 75
Appendix A - Requirements Document ... 76
Inleiding ... 76
Beschrijving probleemdomein ... 76
Probleemomschrijving ... 77
Mogelijk voorbeeld voor het gebruik van KidosR ... 78
Use-case ... 79
Visualisatie en Reportage ... 79
Relatiebeheer ... 80
Use case - Bekijken netwerk ... 82
Use case - Hypothese aanmaken ... 83
Use case - Hypothese bevestigen ... 84
Use case Gegevens invullen ... 85
Use case - Relatie beheer ... 86
Use case - Rapportage ... 87
Plan van aanpak ... 88
Doel ... 88
Kidos ... 88
Mijlpalen ... 91
Requirements definition ... 91
Functioneel ontwerp ... 91
Technisch ontwerp ... 91
Prototype ... 91
Beperkingen en aannamen ... 92
Risico’s ... 93
Doelstellingen ... 93
Aanpak - Projectorganisatie ... 93
Appendix B Referenties ... 94

1 Inleiding

Mijn afstudeerstage bij Ordina Oracle Solutions had als doel te onderzoeken wat de bruikbaarheid en toepasbaarheid van RDF technologie is. Dit onderzoek had als voornaamste doel om de mogelijkheden van RDF technolgie te onderzoeken en te kijken of de problematiek, zoals aanwezig bij het Kidos project, opgelost zou kunnen worden met RDF.

De afstudeerstage heeft een aantal fasen doorlopen:

- Het analyseren van het op te lossen probleem en het opstellen van de vraagstelling en het plan van aanpak zoals respectievelijk beschreven zijn in hoofdstuk 2 en 0
- Het onderzoeken van de mogelijkheden van RDF voor het oplossen van de problematiek in het Kidos project. RDF kan niet gezien worden als individuele techniek maar is onderdeel van een verzameling technologieën die gezamelijk het semantische web vormen. De mogelijkheden, werking en voor- en nadelen van deze verzameling van technologieën worden besproken in hoofdstuk 4 en 5.
- Na de afronding van het bruikbaarheidsonderzoek van RDF technologie is, in overleg met de opdrachtgever, besloten een onderzoek naar een schema-integratie methode uit te voeren. De werking ervan wordt gedemonstreerd aan de hand van een voorbeeld case.

Een Federatief database systeem maakt het mogelijk om meerdere heterogene databasesystemen te ontsluiten door middel van één globaal databasesysteem. Een dergelijk systeem heeft een globaal databaseschema nodig.
De beschreven methode maakt gebruik van een conceptuele modelleertechniek, namelijk Object Role Model. Omdat van bestaande databaseschema's niet altijd een conceptueel model beschikbaar is, zoals een ORM schema, wordt een methode beschreven om een dergelijk conceptueel model in ORM af te leiden van een bestaand relationeel databaseschema.
De bruikbaarheid van deze methoden, namelijk het afleiden van een conceptueel schema en het samenvoegen van conceptuele schema's tot één globaal schema, wordt gedemonstreerd met een voorbeeld case.

Technieken uit het Semantisch Web en uit schema integratie met behulp van ORM bieden beide mogelijkheden verschillende gegevensbronnen te koppelen op een globaal niveau.

2 Opdracht omschrijving

Het RDF (Resource Description Framework) maakt het mogelijk om relaties tussen "resources" aan te brengen. Een resource kan alles zijn wat te identificeren is. Resources kunnen dus gezien worden als verwijzingen naar abstracte objecten die een rol spelen in het probleemdomein van de applicatie (Universe of Discourse).

Vervolgens kunnen relaties afgeleid worden door middel van regels (inferentie). Hierdoor zou het mogelijk moeten zijn om op bestaande gegevensbronnen, zoals een database, extra relaties aan te brengen om hierop inferentie toe te passen.

Oracle biedt de mogelijkheid om RDF en inferentie te combineren met hun (object) relationele database management systeem. De afstudeer opdracht zal bestaan uit het onderzoeken van de mogelijkheden van RDF en de bruikbaarheid van deze technologie. De opdracht is opgedeeld in de volgende deelvragen:

- Wat is RDF en inferentie?
- Het uiteindelijke doel is om vast te stellen of RDF bruikbaar is voor projecten binnen Ordina Oracle Solutions (OOS) zoals bijvoorbeeld het Kidos project.

Concluderend, in deze afstudeeropdracht wordt een haalbaarheidsstudie gedaan naar de bruikbaarheid van RDF. In deze haalbaarheidsstudie zullen de functionele, technische en theoretische aspecten van RDF geanalyseerd worden. Uiteindelijk zal de vraag beantwoord worden wat de toegevoegde waarde van RDF is voor projecten binnen OOS en Kidos in het bijzonder.

De complete opdracht omschrijving is te vinden in [10] en het plan van aanpak is beschreven in hoofdstuk 0.

3 Kidos

Verschillende partijen in de jeugdgezondheidszorg (JGZ) hebben besloten om één landelijk Elektronisch Kinddossier (EKD) in te voeren. Kidos is de naam van het elektronische cliënt volgsysteem wat gebruikt wordt voor de registratie van de medische en psychosociale ontwikkelingen van 0- tot 19-jarigen door GGD's en thuiszorgorganisaties in heel Nederland. Dit systeem wordt ontwikkeld door Ordina Oracle Solutions (OOS).

Omdat het Kidos systeem landelijk is, wordt het mogelijk dat verschillende rechthebbenden toegang hebben tot het systeem waardoor een compleet beeld ontstaat van de medische en psychosociale toestand van de cliënten [25].

4 Ontologie technologie

Dit hoofdstuk beschrijft de principes die ten grondslag liggen aan RDF. Het Semantisch Web [2, 3, 4] maakt gebruik van ontologie technologie. Deze technologie is gebaseerd op o.a. RDF. RDF is namelijk een taal om een ontologie mee te definiëren. Echter RDF heeft niet de uitdrukkingskracht om een complexe ontologie mee te definiëren. Om dit probleem op te

lossen zijn er andere ontologie talen gedefinieerd die als basis RDF gebruiken. Dit hoofdstuk beschrijft dus de verschillende technologieën die gerelateerd zijn aan RDF en Semantisch Web (SW). Het uiteindelijke doel is om de lezer een overzicht te geven van deze technologieën om daarna bruikbare toepassingen te identificeren.

Dit gebeurt aan de hand van een uiteenzetting van de verschillende technologieën die het SW vormen. Om het geheel niet heel abstract te houden worden een aantal voorbeeld toepassingen besproken. Vervolgens wordt geanalyseerd of deze verzameling van beschreven technologieën (die gerelateerd zijn aan het Semantisch Web) aansluiten bij de vraagstelling en voor het probleem een mogelijke oplossing zijn.

4.1 Inleiding

Het Internet is momenteel hoofdzakelijk bedoeld voor menselijke consumptie. Computers worden gebruikt om de informatie, die te vinden is op het Web, te verspreiden. Computers worden niet gebruikt om informatie/gegevens te consumeren. Het resultaat is dat het potentieel van het Web onbenut blijft.

Als computers informatie zouden kunnen verwerken, zoals mensen, zou er meer mogelijk zijn met het Internet. Echter computers zouden de gegevens dan op een semantisch niveau moeten begrijpen.

Als een programmeur informatie zoekt over een bepaalde ziekte in een medische encyclopedie kan deze dit wel lezen. Echter de inhoud begrijpt hij waarschijnlijk niet omdat hij geen medische opleiding heeft genoten. Dit voorbeeld valt te vergelijken met hoe computers gegevens verwerken. De computer verwerkt de gegevens wel maar begrijpt deze niet op semantische niveau (betekenis) en de context waarbinnen ze geplaatst moeten worden al helemaal niet. De gegevens hebben dus geen semantische betekenis voor de computer.

Het gevolg is dat zoekmachines wel statische informatie kunnen opslaan en gebruiken, echter de woorden begrijpen ze niet en de context waarin de woorden gebruikt worden al helmaal niet. Het gevolg is dan ook dat de meeste zoekmachines slecht presteren. De gebruiker moet vervolgens vaak nog handmatig de documenten scannen op bruikbaarheid. Het cruciale punt hierin is dat mensen betekenis aan de gegevens toekennen en over deze gegevens en de betekenis hiervan kunnen redeneren. Door betekenis (semantiek) aan informatie toe te kennen, wordt het mogelijk om computers de gegevens te laten ‘begrijpen’ en over deze gegevens te redeneren.

4.2 Semantisch Web

Om RDF in het juiste perspectief te kunnen plaatsen is het noodzakelijk om het Semantisch Web (SW) te bestuderen, RDF is één van de technologieën die onderdeel zijn van het semantisch web [2, 3, 4].

Het huidige Internet maakt over het algemeen gebruik van HTML pagina’s om gegevens weer te geven. Het probleem is dat HTML alleen bedoeld is voor de opmaak van een webpagina zodat de web browser weet hoe de pagina weergegeven moet worden op het scherm. Om het mogelijk te maken webpagina’s automatisch te laten verwerken door computers is het

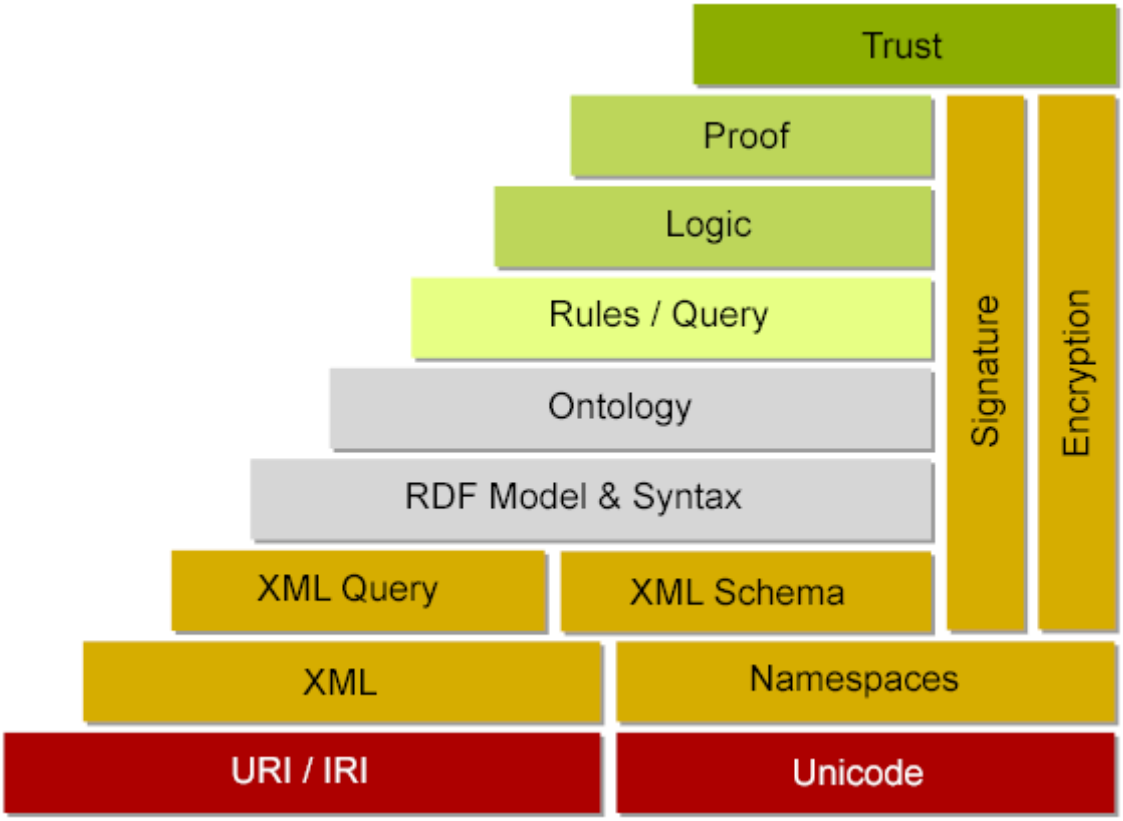
noodzakelijk om niet alleen de opmaak te hebben maar ook de betekenis van de gegevens te weten. De betekenis van data kan worden vastgelegd door middel van metadata.

Dit wordt verduidelijkt aan de hand van het volgende voorbeeld. Om een website te maken die prijzen vergelijkt van verschillende webwinkels is een ingewikkeld proces. De reden hiervoor is dat het in HTML niet gebruikelijk is om aan te geven wat gegevens precies betekenen. X13231 kan best een artikel nummer zijn van een computer scherm dat $232 kost. Echter in HTML is het helemaal niet duidelijk dat X13231 een artikel is en $232 een prijs van dit artikel. Dit betekent dus dat indien een website toegevoegd dient te worden aan de prijsvergelijkingsite er gekeken moet worden door mensen wat de betekenis is van de gegevens en waar deze staan, immers HTML specificeert alleen de opmaak en niet de betekenis en de structuur van gegevens. Deze structuur en hoe deze geïnterpreteerd dient te worden moet kenbaar gemaakt worden aan een computer.

Een mooie oplossing voor dit probleem zou zijn dat de betekenis van gegevens losgekoppeld zou zijn van de opmaak. Indien de betekenis van de gegevens bekend is zou deze informatie automatisch verwerkt kunnen worden door een computer. Het zou dus mogelijk moeten zijn voor een computer om een artikel automatisch te kunnen identificeren en de bijbehorende relevante informatie zoals prijs, productcategorie, garantie, etc. te kunnen identificeren.

Het SW bestaat uit een aantal technologieën waarmee het mogelijk moet worden om betekenis aan gegevens toe te kennen, waardoor het automatisch verwerken van gegevens mogelijk wordt.

De verschillende componenten waaruit het semantisch web is opgebouwd zijn weergegeven in Figuur 1 Het SW bestaat uit verschillende standaarden en technieken die te samen het SW vormen. De ontwikkeling van het SW geschiedt in stappen. De gelaagde structuur vergemakkelijkt het proces om consensus te bereiken. Er zijn verschillende onderzoeksrichtingen bezig met de ontwikkelingen van SW. Welke richting het SW uiteindelijk op gaat is niet bekend. Om het SW bruikbaar te maken is het noodzakelijk dat er standaarden worden opgesteld.



Figuur 1 Het SW bestaat uit verschillende standaarden en technieken die te samen het SW vormen.

Er zijn twee principes die in deze aanpak van het SW belangrijk zijn, namelijk:
Downwards compatibility: een agent die om kan gaan met ontologieën zal ook kunnen omgaan met de laag daar beneden, namelijk RDF.
Upward partial understanding: een agent die om kan gaan met RDF kan deels omgaan met een ontologie omdat een ontologie gerepresenteerd kan worden door gebruik te maken van RDF technologie.

Het semantische web bestaat uit de volgende (technologie) lagen:

- **XML & XML Schema:** XML maakt het mogelijk om gestructureerd documenten te definiëren in een vocabulaire dat gebruikt wordt in een specifiek probleem domein. Deze structuur en het gebruik van vocabulaire maakt XML geschikt om gegevens te verzenden over het Internet. Een mens kan een XML document begrijpen maar een computer heeft geen idee wat de betekenis van een XML document is. Dit betekent dat indien een XML document verwerkt moet worden, een programmeur een programma moet schrijven om XML bestanden te verwerken. De programmeur begrijpt de betekenis van de gegevens in het document en weet hoe deze data verwerkt dient te worden. XML Schema kan gebruikt worden om de structuur waaraan XML gegevens moeten voldoen vast te leggen.
- **XMLQuery:** maakt het mogelijk om een XML document te bevragen. Kenmerkend is dat er alleen vragen gesteld kunnen worden die betrekking hebben op de structuur van de gegevens.
- **RDF:** XML maakt het mogelijk om structuur in documenten aan te brengen, echter het kent geen betekenis toe aan de gegevens. Om de structuur en inhoud van XML documenten te beperken is het mogelijk om XML Schema's te gebruiken. Echter

hiermee is het nog steeds niet mogelijk om betekenis toe te kennen aan de gegevens in een XML document. RDF is een taal die het mogelijk maakt om het datamodel te beschrijven waarmee het mogelijk is om objecten (entiteiten) en hun relaties te beschrijven. RDF kan gezien worden als een data model om simpele statements over resources te beschrijven. RDF is niet afhankelijk van XML maar wordt gerepresenteerd in XML notatie. Vandaar dat RDF bovenop de XML laag is gepositioneerd.

- **RDFS:** RDF Schema maakt het mogelijke om de resources te beschrijven in een structuur. RDF Schema (RDFS) is gebaseerd op RDF. RDFS kan gezien worden als een taal om primitieve ontologieën mee te definiëren.
- **Ontology:** Omdat combinatie van RDF & RDFS niet expressief genoeg is, zijn er andere ontologie talen gedefinieerd zoals de OWL talen familie [3].
- **Query/rulebase:** Het uiteindelijke doel is natuurlijk om informatie te kunnen verkrijgen uit semantisch (data) geannoteerde data; dit kan gedaan worden met SPARQL wat onder andere een query taal is. SPARQL kan gezien worden als een soort van SQL voor semantische data bronnen. Met SPARQL is het ook mogelijk om gegevens te verkrijgen door middel van logisch redeneren (inferentie). Ook is het mogelijk om SPARQL queries te gebruiken om regels te definiëren die nieuwe gegevens kunnen afleiden (rule base).
- **Logic Layer:** De logica laag wordt gebruikt om over ontologie te redeneren (inferentie).
- **Proof Layer:** De proof layer bevat de deductieve processen en ook de bewijsvoering.
- **Trust Layer:** De trust laag is noodzakelijk om te kunnen vaststellen of gegevens betrouwbaar zijn. Deze laag maakt gebruik van digitale handtekeningen en certificaten. Gegevens kunnen alleen gebruikt worden als aanbieders te vertrouwen zijn, de gegevens correct zijn en de beveiliging in orde is.

Kort samengevat bestaat het SW uit een aantal technologieën waarmee het mogelijk is om ‘betekenis’ aan gegevens te kunnen toekennen waardoor de computer deze gegevens op een intelligente manier kan verwerken.

4.3 Informatie versus kennis

Op het Web zijn grote hoeveelheden gegevens te vinden, echter gegevens worden pas waardevol indien ze omgezet kunnen worden in informatie. Om deze omzetting te realiseren is kennis nodig. Kennis valt te relateren aan begrippen zoals context, vertrouwen, perspectief, methodologie en concepten. In de huidige situatie moet de gebruiker van het Internet deze kennis toevoegen om de gegevens, die beschikbaar zijn op het Internet, om te zetten in informatie. Als er semantiek wordt toegevoegd aan het Internet dan zou de computer de betekenis van gegevens kunnen begrijpen, daarmee kunnen deze gegevens opgewaardeerd worden tot relevante informatie. Door het toevoegen van semantische gegevens aan het Internet wordt verwacht dat nieuwe intelligente webtoepassingen mogelijk zullen worden. Hiermee zal het Web een nieuwe dimensie krijgen. Namelijk dat het niet langer alleen een bak met gegevens is, maar dat deze gegevens informatie en kennis gaan representeren. Hierdoor zal het Internet kunnen veranderen van een bak met gegevens in een kennis database. Deze visie op het ‘nieuwe’ Web wordt in [1] beschreven door de vader van het Internet, Tim Berners-Lee.

Binnen het semantische web zijn er twee fundamentele vragen, namelijk:

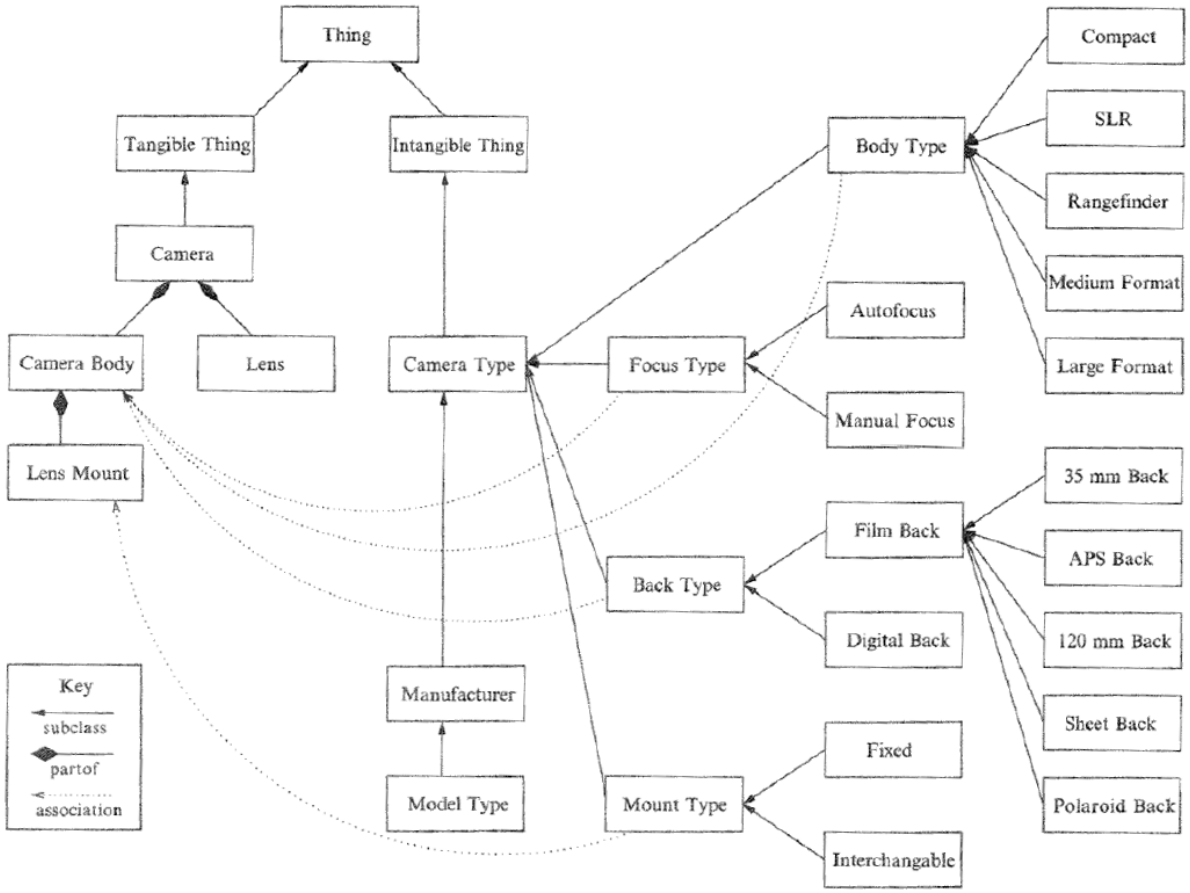
- Hoe moet kennis (informatie + semantiek) gerepresenteerd worden zodat het verwerkt kan worden door computers?
- Indien informatie omgezet wordt naar kennis, hoe moet dan over deze kennis geredeneerd worden?

4.3.1 Kennis representatie

Concluderend kan er gezegd worden dat kennis de sleutel is tot het toegankelijk maken van informatie [5] op het Web. Als informatie op de juiste manier gerepresenteerd wordt is het mogelijk om hiermee te rederenen. Door te redeneren met bestaande kennis kan nieuwe kennis worden afgeleid. Kennis is gerelateerd aan de wereld om ons heen; de wereld om ons heen kan vastgelegd worden door een conceptualisatie van de wereld te maken. Een conceptualisatie is een versimpelde voorstelling van een deel van de wereld. Mensen relateren concepten door middel van associaties. Een conceptualisatie bestaat daarom ook uit een netwerk van concepten die aan elkaar relateerd zijn; dit valt te visualiseren als een gericht netwerk. Dit netwerk bestaat uit concepten die met elkaar verbonden zijn. Een dergelijke conceptualisatie hiërarchie (netwerk) wordt een ontologie genoemd. Een ontologie is een specificatie van een conceptualisatie.

Een voorbeeld van een conceptualisatie is de camera ontologie. Het domein, ook wel ‘universe of discourse’ genoemd, beschrijft concepten die betrekking hebben op camera’s en de relaties tussen verschillende concepten uit het domein, zie Figuur 16.

Een ontologie wordt gevisualiseerd als een gerichte graaf die opgebouwd is uit concepten en relaties tussen de concepten (gerichte paden). Vervolgens kan informatie gerelateerd worden aan een ontologie, dit betekent dat informatie geclassificeerd wordt door gebruik te maken van de conceptualisatie. Door het maken van een ontologie over camera’s ontstaat er kennis over het domein van camera’s. Vervolgens kunnen we data hieraan relateren (classificeren). Als we een conceptualisatie hebben en hieraan informatie relateren wordt het mogelijk om intelligente vragen te stellen. Een voorbeeld van een dergelijke vraag is: ‘Welke camera’s voldoen aan mijn wensen en sorteer deze op prijs?’.



Figuur 2 De camera ontologie maakt het mogelijk om verschillende concepten met betrekking tot fotografie ter relateren en daarover te redeneren.

Concluderend: om computers automatisch gegevens op een slimme manier te laten verwerken is het noodzakelijk om een conceptualisatie (ontologie) te maken en classificatie op gegevens uit te voeren. Er blijven twee praktische vragen onbeantwoord over ontologieën, namelijk:

- hoe leggen we een ontologie vast (representatie)?
- hoe moet de classificatie van gegevens plaats vinden?

4.3.2 Levenscyclus van een ontologie

Een ontologie is een specificatie van een conceptualisatie. Een ontologie is een verzameling van concepten (objecten) en relaties daartussen. Een ontologie wordt meestal gebruikt om een kennis vocabulaire te definiëren en kan gevisualiseerd worden met een gericht netwerk.

Om een ontologie te representeren is een ontologie taal nodig, zoals RDF, RDFS en de OWL talen familie. Dit zijn de ontologie talen die momenteel gebruikt worden, de talen verschillen in de mate van uitdrukkingskracht en zijn gebaseerde op XML. De semantiek van deze talen is gebaseerd op grafen theorie en descriptive logic [3]. De talen verschillen in de uitdrukkingskracht. RDFS heeft de minste uitdrukkingskracht en de OWL talen familie heeft drie verschillende taal varianten die allemaal verschillend in uitdrukkingskracht zijn. Echter hoe meer uitdrukkingskracht een ontologie taal heeft hoe preciezer een conceptualisatie gemaakt kan worden. Het lijkt verleidelijk om dan de ontologie taal te kiezen met de meeste

uitdrukkingskracht. Echter hoe meer uitdrukkingskracht een ontologie taal heeft hoe moeilijker (computational expensive) het redeneren ermee wordt. Daarom is het verstandig om een ontologie taal te kiezen die genoeg uitdrukkingskracht heeft om het probleem domein adequaat te modeleren. Het uiteindelijke doel is dat een ontologie wordt gebruikt door een applicatie. De mate van uitdrukkingskracht die applicaties nodig hebben hangt af van de (functionele) eisen die de applicaties stellen aan de ontologie. Het is belangrijk om te onthouden: kies de taal met de minste uitdrukkingskracht indien de functionele eisen van de applicaties dit toelaten.

Het ontwerpen (definiëren) van ontologieën is een tijdrovend en ingewikkeld proces. Mede hierom is het verstandig om ontologieën te delen en te combineren. Het is namelijk wel voor te stellen dat verschillende domeinen op de een of andere manier overlap met elkaar hebben of gerelateerd zijn.

4.3.3 RDF & RDFS

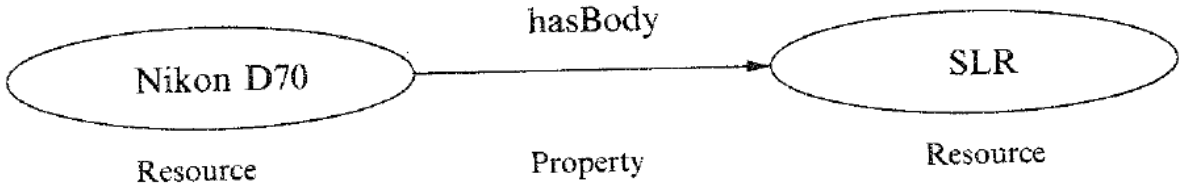
Met het Resource Description Framework (RDF) in combinatie met Resource Description Framework Schema (RFDS) is het mogelijk om een kennis database te bouwen. Met RDF is het mogelijk om gegevens te representeren en met RDFS is het mogelijk om een simpele ontologie te definiëren. Eerst wordt uitgelegd wat RDF is en op welke principes het gebaseerd is, daarna wordt ingegaan op RDFS en de samenhang tussen RDF en RDFS.

4.3.4 Resource Description Framework

RDF is een standaard voor het representeren van kennis. Met RDF kunnen resources beschreven worden door gebruik te maken van metadata. RDF gebruikt als bouwsteen de **tripel**. Een RDF tripel is een statement over resources. Een statement koppelt altijd een eigenschap aan een resource. Daarnaast zijn RDF statements gerelateerd aan statements in de natuurlijke taal. Een voorbeeld van een statement in de natuurlijke taal: ‘Een auto heeft vier wielen’.

Om statements automatisch te kunnen verwerken en er over te kunnen redeneren, moet er een vaste structuur zijn. In RDF heeft ieder statement een vaste structuur die bestaat uit een onderwerp en een predicaat dat een eigenschap toekent aan het onderwerp. De waarde van die eigenschap is eveneens een resource. Deze structuur vormt het RDF tripel.

Het onderwerp is een resource waar het statement over gaat. Aan dit onderwerp kent het predicaat een eigenschap toe, de waarde van de eigenschap wordt gerepresenteerd door het object. Ieder element van de triple is een resource dat te identificeren valt door middel van een Uniform Resource Identifier (URI). Een triple kan ook gezien worden als een binaire relatie, P(S,0) waarbij P het predicaat is, S het onderwerp en O het object. Echter om de tripels in een RDF te visualiseren is het verstandig om deze te beschouwen als een gerichte graaf, zie Figuur 3.

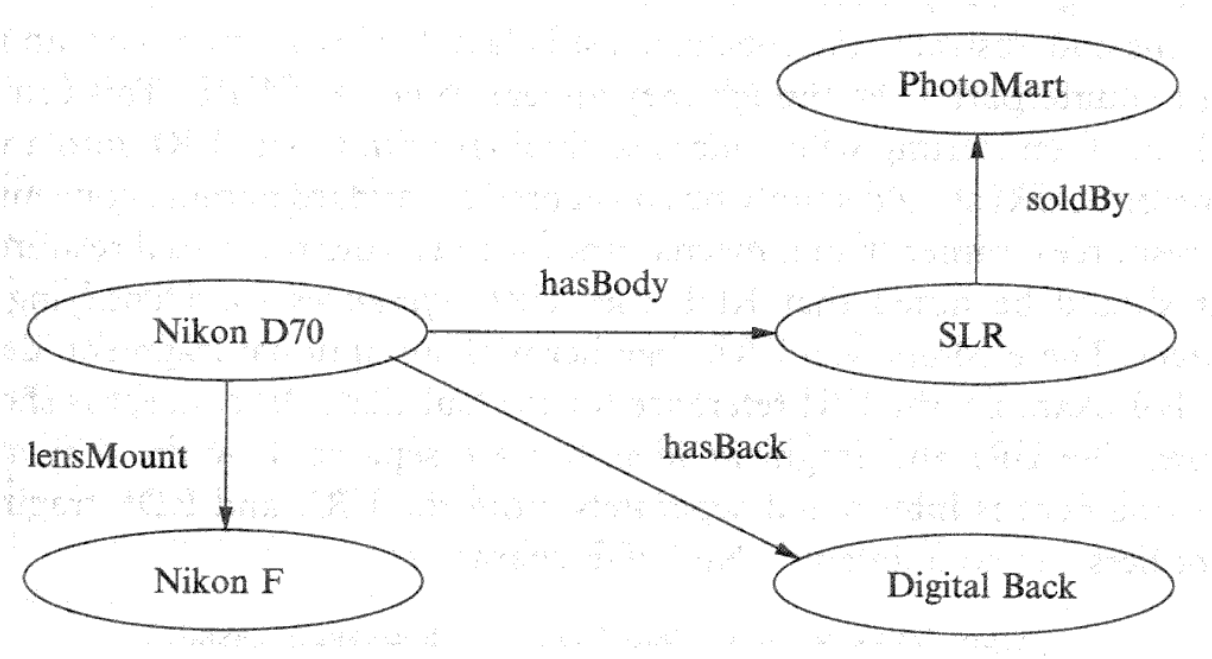


Figuur 3 Voorbeeld van de visualisatie van een RDF tripel. De resources worden weergegeven door een gericht pad (predicaat) met het object (rechter ellips).

RDF wordt opgeslagen door gebruik te maken van de XML syntax. Zoals we hebben gezien zijn RDF triples opgebouwd uit resources. Deze resources kunnen geïdentificeerd worden door gebruik te maken van URI's (Universal Resource Identifier).

```
<rdf:Description rdf:about="Nikon D70">
      <camera:hasbody>SLR</camera:hasbody>
</rdf:Description>
```

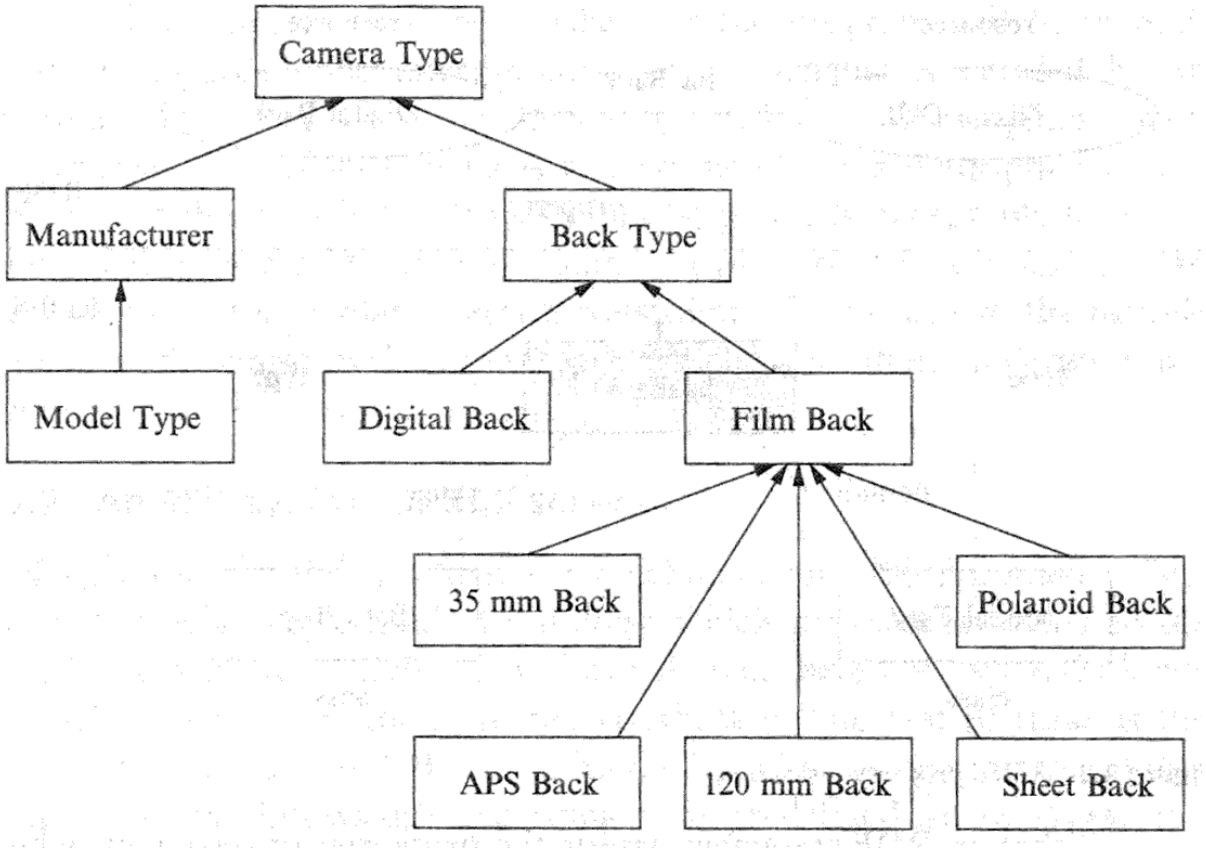
De triples worden gedefinieerd door de *rdf:Description* tag met als predicaat en waarde *<camera:hasbody>SLR</camera:hasbody>.* Waarbij het predicaat camera:hasbody is en de waarde SLR. In Figuur 4 is een semantisch netwerk van RDF tripels te zien.



Figuur 4 Semantisch netwerk dat bestaat uit meerdere RDF tripels.

RDF Schema (RDFS) maakt het mogelijk om het vocabulaire van een bepaald domein vast te leggen. Dit gebeurt door gebruik te maken van een overervinghiërarchie (generalisatie/specialisatierelatie). RDFS is eigenlijk niet expressief genoeg om een ontologie te kunnen representeren. Het is wel mogelijk om een taxonomie (hiërarchie van gerelateerde concepten) te definiëren met RDFS. De OWL talen familie biedt meer uitdrukkingskracht dan RDFS, ze zijn gebaseerd op RDFS en kunnen dan ook gezien worden als een uitbreiding op RDFS.

Het doel van RDFS is om de verschillende concepten in de kennis database vast te leggen door middel van generalisatie/specialisatierelaties aan te geven tussen de concepten in het domein. Een voorbeeld van een dergelijke hiërarchie is te zien in Figuur 5.

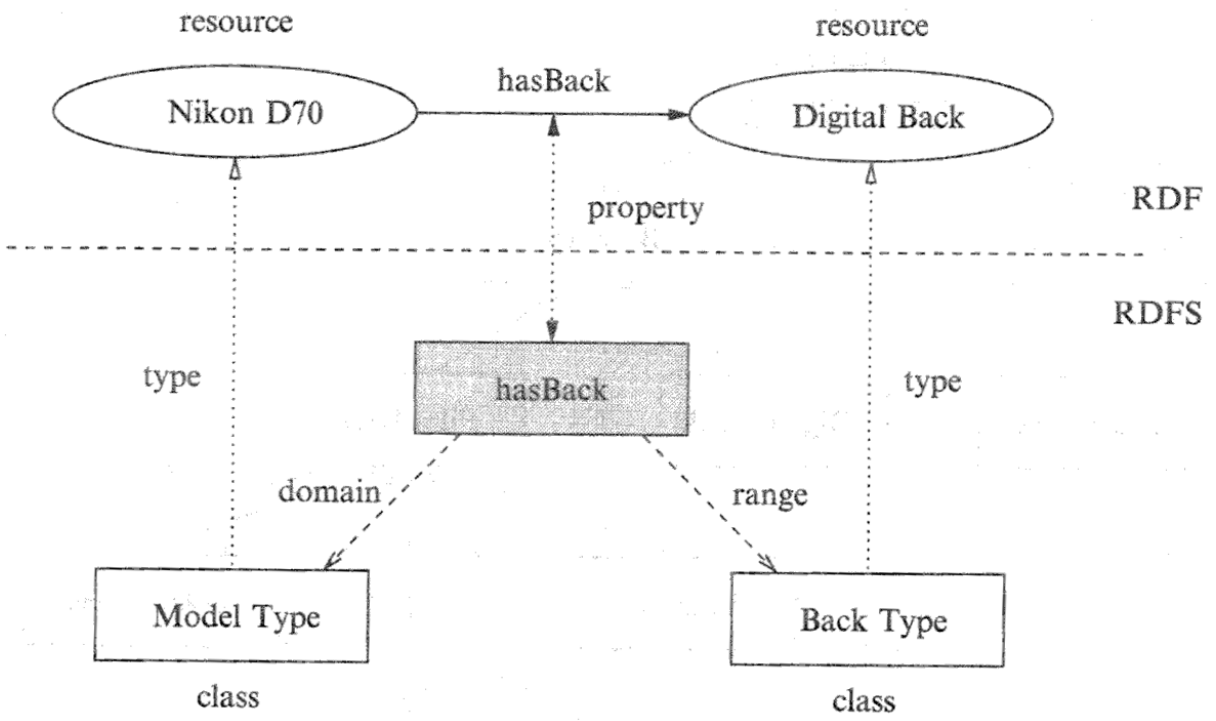


Figuur 5 Voorbeeld taxonomie gemaakt met behulp van RDFS. Er wordt alleen gebruik gemaakt van generalisatie/specialisatie relaties.

4.3.5 RDF + RDFS

In RDF documenten worden instanties gedefinieerd van classes die in RDFS zijn gedefinieerd. Een instantie is een concrete representatie van een klasse. In Figuur 6 is het verband te zien tussen RDF en RDFS. Alle resources worden gekoppeld aan een klasse. Dit is een gebruikelijke techniek om alle componenten van een RDF tripel te koppelen aan een klasse; dit gebeurt met *rdf:type*.

Met RDFS is het mogelijk een eigenschappen (predicaat) hiërarchie te maken (sub-properties). Het probleem is dat de eigenschappen op globaal niveau worden gedefinieerd en niet op klasse niveau zoals in OO programmeren. Hierdoor is het onmogelijk om de range van een eigenschap per klasse te specificeren.



Figuur 6 Verband tussen RDF en RDFS.

Een ontologie maakt het mogelijk om de structuur van kennis te representeren. Een kennisdatabase (ontologie + instanties) wordt gebruikt om gestructureerd gegevens op te slaan. Bij een database is het gebruikelijk om vragen (query) te stellen. Bij een kennisdatabase willen we niet alleen vragen stellen maar ook redeneren over de kennis.

Queries worden gebruikt om specifieke vragen te stellen aan een kennisdatabase, een aantal voorbeelden van dergelijke vragen zijn:

- Welke soorten camera behuizingen zijn er?
- Welke normaal formaat camera's zijn digitaal?
- Welke digitale SLR camera heeft de laagste prijs bij Photomart?

De eerste vraag kan beantwoord worden door de ontologie te gebruiken. De tweede vraag wordt beantwoord door naar alle camera's in de kennisdatabase te zoeken die als normaalformaat en digitaal zijn geclassificeerd. De laatste vraag kan beantwoord worden door de camera kennisdatabase te gebruiken en een andere kennisdatabase waarin prijsinformatie te vinden is. Bij het laatste voorbeeld is dan ook sprake van het samenvoegen van ontologieën. Ontologieën samenvoegen maakt het mogelijk om verschillende kennisdatabase systemen te combineren. Bij Federatieve databasesystemen is er sprake van het samenvoegen van verschillende heterogene databasesystemen. In hoofdstuk 9 wordt een methode beschreven om een dergelijk systeem te ontwerpen.

De manier van data bevragen (queries) in een kennisdatabase lijkt op hoe dit in Relationele Database Management Systemen (RDBMS) gebeurt. Een ontologie binnen een kennisdatabase heeft dezelfde functionaliteit als een database schema in een RDBMS. Echter er zijn natuurlijk verschillen tussen een kennisdatabase en RDBMS, namelijk:

- In een kennisdatabase wordt het schema gevormd door een semantisch netwerk. In een RDBMS systeem wordt een relationeel model gebruikt. Bij een RDBMS worden er vragen gesteld aan relaties (tabellen) en in een kennisdatabase worden vragen gesteld aan delen van het semantische netwerk. Een antwoord op een vraag aan een kennisdatabase kan gerepresenteerd worden als een relatie (tabel).
- Een kennisdatabase is minder gestructureerd dan een database, met andere woorden een kennisdatabase is semi-structured. In een kennisdatabase kan kennis ontbreken en inconsistent zijn, hierdoor is het resultaat van een query niet altijd even betrouwbaar.
- Gegevens zijn gedefinieerd of worden afgeleid (inference) van bestaande gegevens. Er is geen verschil te zien voor de gebruiker tussen afgeleide gegevens en gedefinieerde gegevens. In een relationeel databases systeem zijn geen gegevens beschikbaar die verkregen zijn door (logisch) redeneren.
- Kennis wordt gerepresenteerd door verschillende syntactische vormen (RDF, OWL, etc.). Het gevolg is dat queries aangepast moeten worden aan de onderliggende structuur. Hiermee ontstaat een afhankelijkheid tussen de query taal en de representatie van de ontologie en de instantie data.
- Databases zijn gebouwd met een speciaal doel (representatie van een eindig domein), echter bij kennisdatabases is van te voren niet bekend wat er wel en niet opgeslagen gaat worden (oneindig domein).

4.3.6 SPARQL – Query taal voor kennisdatabase

In de vorige paragraaf zijn de belangrijkste kenmerken van een kennisdatabase en de belangrijkste verschillen ten opzichte van een relationeel database systeem behandeld. Het doel van een (kennis) database is natuurlijk het stellen van vragen over de opgeslagen gegevens, met als doel het verkrijgen van informatie.

In deze paragraaf wordt behandeld hoe een kennisdatabase bevraagd kan worden. Omdat RDF syntactisch op XML gebaseerd is, zou een XML query taal gebruikt kunnen worden. Echter dit is niet het geval omdat de XML query talen het mogelijk maken om vragen te stellen die gebaseerd zijn op de structuur van het onderliggende document. Echter het is niet relevant om vragen te stellen over de structuur van het document, het is tenslotte de betekenis van de gegevens die belangrijk is in een kennisdatabase. Voordat de query taal om kennisdatabase systemen mee te bevragen bestudeerd wordt, wordt een dergelijke query taal eerst op logisch niveau gedefinieerd.

Een kennisdatabase bestaat uit een verzameling van statements (triples) T. Ieder statement $t \varepsilon T$ heeft de vorm $P(S, O)$, met P het predicaat O het object en S het onderwerp. Een query wordt gespecificeerd door een patroon q, dit is een statement met eventueel vrije variabelen.

Een voorbeeld van een dergelijke query is *?hasBack(x, DigitalBack).* Stelt de vraag: 'Welke digitale camera's zijn er?'

Het resultaat van q gesteld aan T is een verzameling Q die voldoet aan het patroon van q, $Q = match(q,T)$. Voor elke match (patroon van query is voldaan) worden eventuele vrije variabelen in q vervangen, zodat de vraag beantwoord wordt.

?hasBack(X, DigitalBack) heeft als antwoord bijvoorbeeld *X={Canon350D, Sony1000Alpha}*.

Simple Protocol And RDF Query Language (SPARQL) is een standaardisatie van: een simpel netwerk protocol (om gegevens over queries uit te wisselen) en een query taal. Alleen de query taal wordt nu behandeld. De query taal lijkt sterk op SQL en dus niet op XML ook al is RDF gebaseerd op de XML-syntax. Met SPARQL wordt geprobeerd de verschillende query talen voor kennisdatabases die op RDF gebaseerd zijn. te vervangen door één standaard. [11] bespreekt in detail meerdere talen en methodieken voor het bevragen van kennisdatabases.

Een voorbeeld van een SPARQL query is:

```
PREFIX camera: <http://www.mycamera.org/photo#>
SELECT ?x
WHERE { ?x camera:hasback camera:digitalback }
```

Wordt deze query uitgevoerd op de triples uit Tabel 1 dan is het resultaat Tabel 2.

Subject	**Predicaat**	**Object (waarde)**
canon:CanonEOS300D	camera:manufacturer	canon:canonInc
canon:CanonEOS300D	camera:hasBack	camera:DitigtalBack
canon:CanonEOS300D	camera:resolution	“6.3”^^xsd:decimal
nikon:NikonD70	camera:manufacturer	Nikon:NikonCorporation
nikon:NikonD70	camera:hasBack	camera:DitigtalBack
nikon:NikonD70	camera:resolution	“6.1”^^xsd:decimal
pentax:PentaxK1000	camera:manufacturer	pentax:PentaxCorporation
pentax:PentaxK1000	camera:hasBack	camera:35 mm

Tabel 1 Triples in kennisdatabase.

?x
canon:CanonEOS300D
nikon:NikonD70

Tabel 1 Resultaat van Query.

De accolades na het *WHERE* keyword geven een verzameling van RDF tripels weer, {} staat voor een lege verzameling van triples. Het patroon van de triples in de *WHERE* conditie zal gematched worden met de data die beschikbaar is in de kennisdatabase. Tijdens de uitvoering van een dergelijke query zullen de variabelen gebonden worden aan RDF tripel componenten. Variabelen worden aangeduid met een ‘*?*’ gevolgd door een identifier. Wat opvalt, is dat er geen *FROM* keyword wordt gebruikt zoals in SQL maar PREFIX.

Normaal gesproken is een *PREFIX* keyword niet nodig, indien bekend is uit welke kennisdatabase de gegevens moeten komen.

In de *WHERE* conditie kunnen ook ingewikkelder patronen gedefinieerd worden, zie het voorbeeld hieronder.

```
PREFIX camera: <http://www.mycamera.org/photo#>
PREFIX nikon: <http://www.mycamera.org/nikon#>
SELECT ?x ?y
WHERE { ?x camera:hasBack ?y.
?x camera:manufacturer nikon:NikonCorporation }
```

Deze query geeft alle camera’s van Nikon terug, het moet opgemerkt worden dat zowel de eerste conditie (tripel patroon) als de tweede conditie in de *WHERE* clause gematched moeten worden. Door deze structuur is het mogelijk om patroon herkenning te doen.

Sommige objecten hebben niet een volledig netwerk, er zijn bijvoorbeeld camera's waarvoor er niet is gespecificeerd wat de resolutie is. Als de resolutie weergegeven moet worden van camera's, waarvan dit bekend is, moet het *OPTIONAL* keyword gebruikt worden, zie voorbeeld hieronder.

```
PREFIX camera: <http://www.mycamera.org/photo#>
SELECT ?model ?resolution
WHERE {?model camera:manufacturer ?manufacturer.
OPTIONAL {?model camera:resolution ?resolution} }
```

Indien het *OPTIONAL* keyword niet gebruikt was zou het resultaat Tabel 3 zijn maar met het *OPTIONAL* keyword is het resultaat Tabel 4.

?model	**?resolution**
canon:CanonEOS300D	"6.3"^^xsd:decimal
nikon:NikonD70	"6.1"^^xsd:decimal

Tabel 3 Resultaat zonder gebruik te maken van OPTIONAL keyword.

?model	**?resolution**
canon:CanonEOS300D	"6.3"^^xsd:decimal
nikon:NikonD70	"6.1"^^xsd:decimal
pentax:PentaxK1000	

Tabel 4 Resultaat met gebruik van het keyword OPTIONAL.

Soms is het handig om in een query een conditie op te nemen die betrekking heeft op een literal (rdf component dat geen URI is, bijvoorbeeld een getal, datum, etc.). Dit kan met het keyword FILTER, het voorbeeld hieronder verduidelijkt dit.

```
PREFIX camera: <http://www.mycamera.org/photo#>
SELECT ?model ?resolution
WHERE { ?model camera:resolution ?resolution.
FILTER (?resolution >= 6.2) }
```

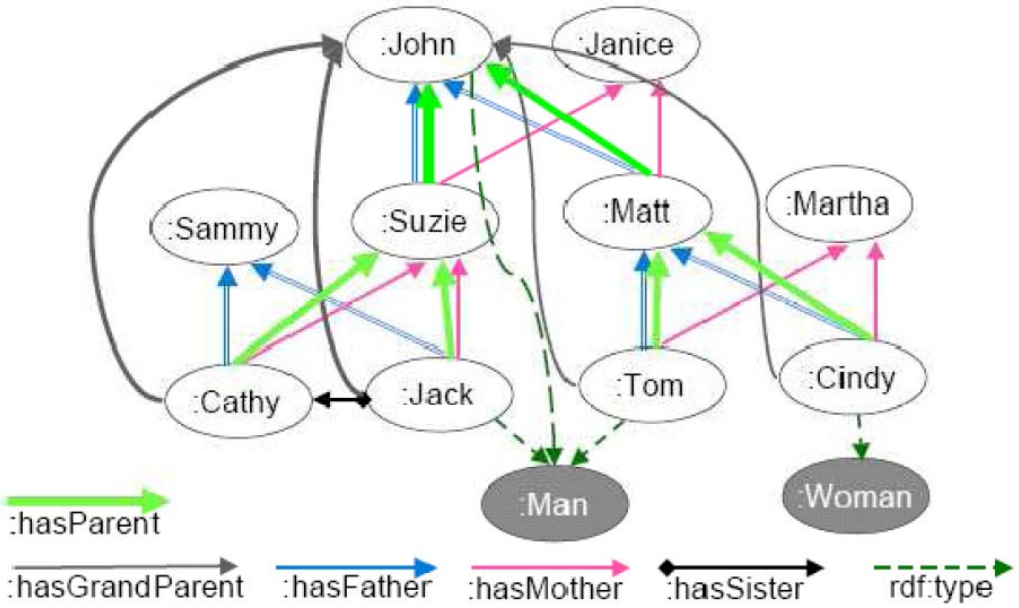
Meer informatie en voorbeelden over SPARQL kunnen gevonden worden in de literatuur [3].

Er zijn nog een aantal andere keywords die in SPARQL gebruikt kunnen worden en die eigenlijk voor zich spreken, namelijk: *DISTINCT* (geen duplicaten in resultaat), *LIMIT* (aantal records in resultaat beperken), *ORDER BY* (sorteren van het resultaat).

4.3.7 Inferentie

Inferentie betekent dat het mogelijk is om conclusies te trekken uit bestaande relaties (gegevens). Het is dus mogelijk om relaties vast te leggen door middel van een ontologie taal. Met een dergelijke taal definieert de gebruiker een netwerk van relaties tussen objecten. Met een dergelijk netwerk wordt het mogelijk om vragen te stellen over relaties en objecten. Het vragen stellen in een RDF netwerk kan gedaan worden door een speciale query taal zoals SPARQL. Echter in de meeste Kennisdatabase systemen (zoals OracleRDF) is het ook mogelijk om regels op te nemen in een rulebase. Een rulebase bestaat in feite uit SPARQL queries die het systeem altijd gebruikt indien er vragen gesteld worden.

Een dergelijke rule ziet er als volgt uit: A :hasGrandParent C → A :hasParent B and B :hasParent C, zie Figuur 7.



Figuur 7 Familie relaties met de toegevoegde relatie hasGrandParent, deze relatie wordt niet opgeslagen.

Het is dus mogelijk om aan een kennisdatabase vragen te stellen, het antwoord kan bestaan uit bestaande gegevens en/of afgeleide gegevens.

Opmerking: een regel is eigenlijk ook een relatie die niet expliciet is aangebracht in de gerichte graaf die een ontologie representeert. Zulke regels kunnen worden vastgelegd in een rulebase, dat een onderdeel is van het RDF database systeem.

4.3.8 Ontologie gecombineerd met database

Als we ontologie technologie vergelijken met een database schema is er geen twijfel mogelijk dat met een ontologie een betere beschrijving (model) van de wereld gemaakt kan worden. Bij een kennisdatabase kunnen ook vragen gesteld worden die niet (of zeer moeilijk) te beantwoorden zijn met een relationele database. In 4.3.9 wordt een voorbeeld gegeven van een dergelijke vraag.

Een ander belangrijk aspect van het relationele database management model is dat het goed te begrijpen is. Tussen de complexiteit/omvang en begrijpbaarheid van een relationeel schema zit een lineair verband. Bij een kennisdatabase is er geen lineair verband tussen de omvang en de complexiteit. Deze complexiteit neemt vrij snel toe.

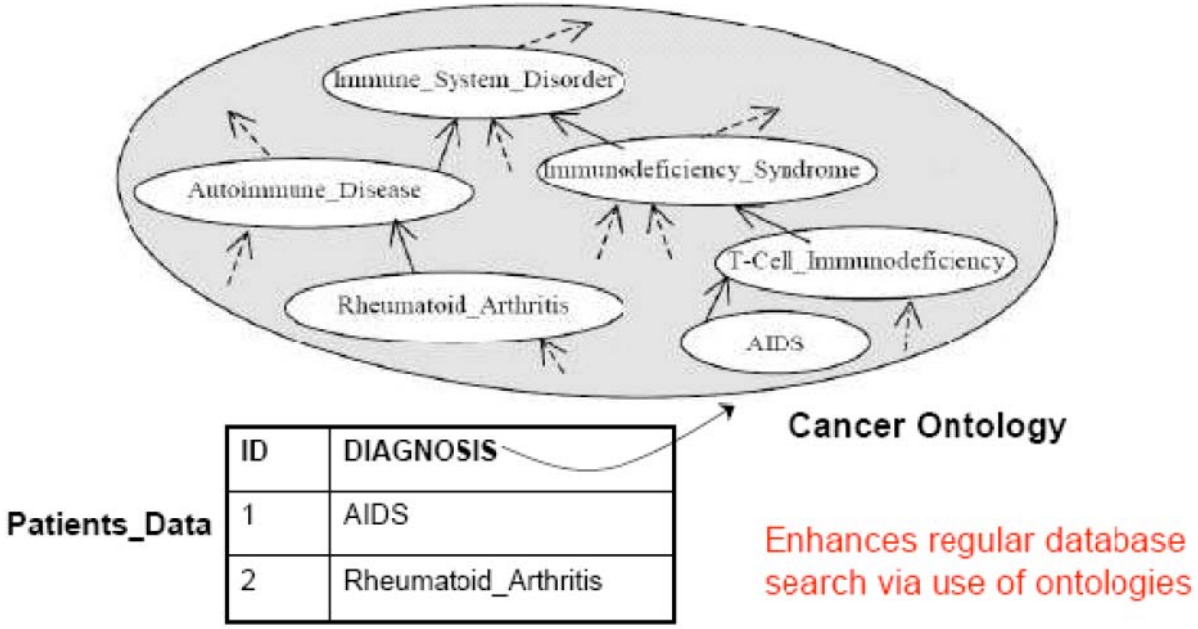
Het mooie van het relationele systeem is dat het voorspelbaar, onderhoudbaar en te verifiëren valt door mensen met een informatica opleiding of exacte opleiding (zeker abstractie niveau is nodig). Een simpele ontologie die gemaakt is in OWL DL zal een goed opgeleide ontology engineer meer moeite kosten om te ontwerpen en deze zal moeilijker te onderhouden zijn dan een gewoon database schema/systeem. Het probleem is dat de complexiteit bij ontologie systemen niet lineair toeneemt ten opzicht van de omvang van een dergelijk systeem zoals bij relationele databases het geval is.

Reden: Een ontologie is gebaseerd op wiskundige logica en dit is moeilijker te begrijpen dan relationele database systemen (relationele algebra). Daarnaast is bij een relationeel schema het domein eindig.

4.3.9 Voorbeeld van database en ontologie gecombineerd

Als ik in een 'simpele' zoekmachines zoek naar een artikel over AIDS dan krijg ik alleen artikelen over AIDS. In een zoekmachine die gebruik maakt van een ontologie, weet de zoekmachine dat AIDS een ziekte is in de T-Cell'Immumodeficiency' categorie en kunnen hieraan gerelateerde artikelen opgenomen worden in het zoekresultaat.

Oracle Enterprise Edition heeft de mogelijkheid om SPARQL queries (bevragen van triplestore) en rulebase te combineren met gegevens uit de database. Het is zelfs mogelijk om SQL queries te schrijven met daarin SPARQL queries, zie Figuur 8.



Figuur 8 Een zoekopdracht naar een bepaalde diagnose kan worden verbeterd doordat ontologieën gebruikt worden.

4.3.10 Toepassingen

Ontologie technologie wordt onder andere gebruikt in kennis systemen. Om dit te illustreren worden twee voorbeelden van toepassingen besproken; voor een gedetailleerdere behandeling van deze voorbeelden zie [12, 8]. Daarnaast worden in [3] ook toepassingen besproken maar deze zijn niet gedetailleerd beschreven. In [2] worden twee simpele voorbeeld applicaties beschreven die de werking en principes van het semantisch web verduidelijken.

4.3.10.1 De digitale bibliotheek

Door het toepassen van zoekmachines is het mogelijk geworden om verschillende boeken, journaals, multimedia producten, en dergelijke snel te doorzoeken. Het zoeken van informatie in een bibliotheek is makkelijker geworden door gebruik te maken van zoekmachine technologie. Verschillende bibliotheken gebruiken verschillende database schema's voor de catalogi. De verschillende bibliotheken houden ongeveer dezelfde gegevens bij in de catalogi, echter de manier waarop deze gegevens verwerkt en opgeslagen worden is verschillend, met andere woorden er is sprake van dezelfde semantiek maar andere schema's.

De gebruiker zou door gebruik te maken van één applicatie alle catalogi op een uniforme manier willen doorzoeken. Het federeren van de verschillende catalogus systemen kan gedaan worden door gebruik te maken van ontologie technologie.

In [8] wordt beschreven hoe dit probleem opgelost kan worden met behulp van ontologie technologie. Andere mogelijkheden voor het oplossen van een dergelijk probleem kunnen gevonden worden in [26].

Er zijn vijf moeilijkheden/vraagstukken gedefinieerd door de Goverment's Information Infrastructure Technology and Application Working Group [9].

1. Interoperability: hoe moeten systemen en software samenwerken. Op een ander niveau is het meer de vraag hoe kan semantische samenwerking (Interoperability) gerealiseerd worden. De grote uitdaging in semantische applicaties is dan ook 'deep semantic interoperability'.
2. Description of objects and repositories: de noodzaak om een gemeenschappelijk schema te definiëren om het mogelijk te maken gedistribueerd te zoeken en delen van gegevens mogelijk te maken. Hoe maken we een ontologie zodanig dat de verschillende, lokale ontologieën gecombineerd kunnen worden tot één globale ontologie? Met één dergelijke globale ontologie is het mogelijk te kunnen zoeken door meerdere lokale ontologieën [26].
3. The collection and management of nontextual information: hoe verzamel, onderhoud en presenteer je informatie die niet bestaat uit tekst. Het probleem is dat er verschillende hardware en sofware systemen en bestandsformaten zijn. Daarnaast is een onbeantwoorde open vraag hoe je niet-tekst gegevens kan relateren aan tekst. Met als doel dat de twee representatie vormen elkaar aanvullen en zodoende aanvullende informatie bevatten. Dit is een onderzoekvraag die zich buiten de scoop van ontologie technologie bevindt.
4. Gebruikers interface: hoe representeer je deze informatie op een uniforme manier aan de gebruiker. Een ontologie kan helpen in het aanbrengen van de basis voor het weergeven van informatie (de structuur van het weergeven, dus niet de opmaak).
5. The social context: hoe dient er omgegaan te worden met vraagstukken zoals sociale, juridische en economische aspecten.

Nu volgt een beschrijving van het bibliotheek systeem. In dit systeem worden de documenten (boeken, multimedia, etc.) gecategoriseerd aan de hand van 9000 categorieën. Deze categorieën vormen een hiërarchie. In [26] wordt beschreven hoe zulke hiërarchieën semi-automatisch gegeneerd en aangepast kunnen worden door middel van ontologie learning & discovery software. Deze categorieën hiërarchie bevat dus verschillende onderwerpen en maakt slim zoeken mogelijk.

Daarnaast wordt er gekeken naar wat gebruikers in het verleden gezocht hebben, met deze informatie kan een profiel van een gebruiker gemaakt worden. Dit profiel kan dan gebruikt worden om de zoekmachine te voorzien van additionele informatie.

De documenten worden geannoteerd met ontologie informatie zodat er gezocht kan worden in de documenten met behulp van de categorieën hiërarchie (ontologie). Door de toepassing van hiërarchie van onderwerpen en de documenten te annoteren met onderwerpen informatie kan de zoekfunctionaliteit verbeterd worden.

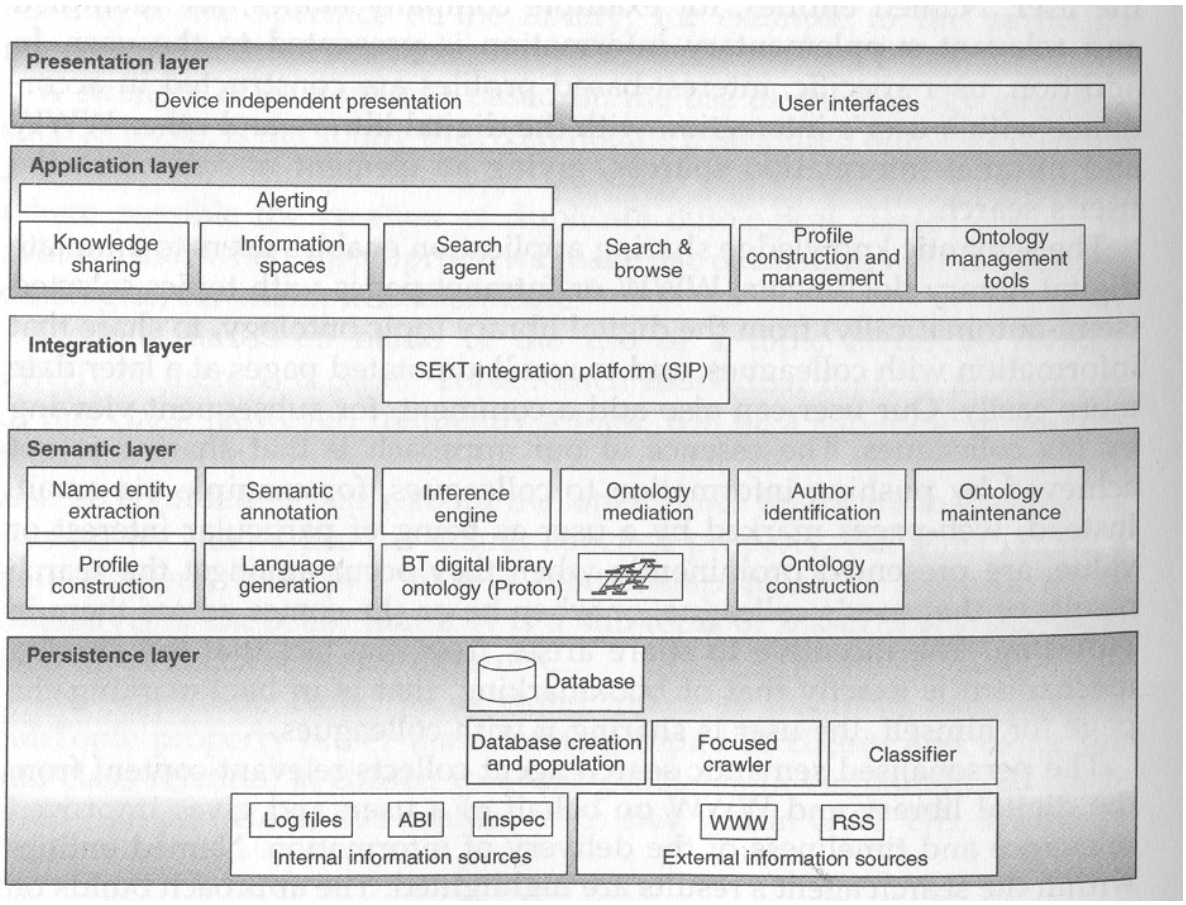
4.3.10.1.1 De implementatie van de digitale bibliotheek

De digitale bibliotheek is gebaseerd op technologieën die ontwikkeld zijn door het SEKT project. Een korte beschrijving van SEKT kan gevonden worden in [27]. De ontologie die ontwikkeld is maakt gebruik van de PROTON [28] knowledge manamagement ontologie en DUBLIN [29] core ontologie.

4.3.10.1.2 Architectuur

De architectuur van de digitale bibliotheek bestaat uit vijf lagen (zie Figuur 9):

1. The persistence layer: hier worden de gegevens uit verschillende databronnen gecombineerd tot één database.
2. The semantic layer: is verantwoordelijk voor de communicatie tussen de lokale ontologieën, hieronder vallen maken, verbeteren en bevragen van de informatie in een ontologie.
3. The integration layer: is de middleware tussen de applicatie laag en de ontologie laag.
4. Application Layer: de applicaties die gebruik maken van de ontologieën. De applicaties krijgen toegang tot de gegevens (semantische) laag door gebruik te maken van de functionaliteiten die de integratie laag aanbied.
5. User interface: de presentatie laag die ervoor zorgt dat de applicaties op verschillende platformen weergegeven kunnen worden (pc, mobieltje, etc.).



Figuur 9 Architectuur van de digitale bibliotheek.

4.3.10.2 Legal case

Rechters doen uitspraken aan de hand van het wetboek, juridische kennis (procedures/jurisprudentie), maatschappelijke-, sociale-kenmerken, etc. Deze kennis is verspreid en het is niet gemakkelijk voor rechters om de juiste informatie te vinden binnen de beperkte tijd die hiervoor beschikbaar is. Mede doordat de werkdruk hoog is zijn ze niet in staat om grondig te zoeken naar relevante informatie zoals jurisprudentie, procedures en gelijksoortige gevallen. Om dit probleem op te lossen is er gekozen voor het maken van een juridisch portal voor de Spaanse Justitie die rechters kunnen gebruiken voor het opzoeken van relevante informatie. Huidige zoekmachines bleken voor het vinden van relevante informatie niet toereikend.

Door rechters te interviewen is een ontologie gedefinieerd (Ontology for Professional Juridical Knowledge). Deze ontologie bestaat uit zevenhonderd termen. De juridische gegevens werden vervolgens geannoteerd met behulp van deze ontologie.

4.3.10.2.1 Functionaliteit & Architectuur

Het systeem kan gezien worden als een FAQ platform, de rechter stelt een vraag in de natuurlijke taal. Het systeem probeert bij deze vraag het best passende antwoord te vinden.

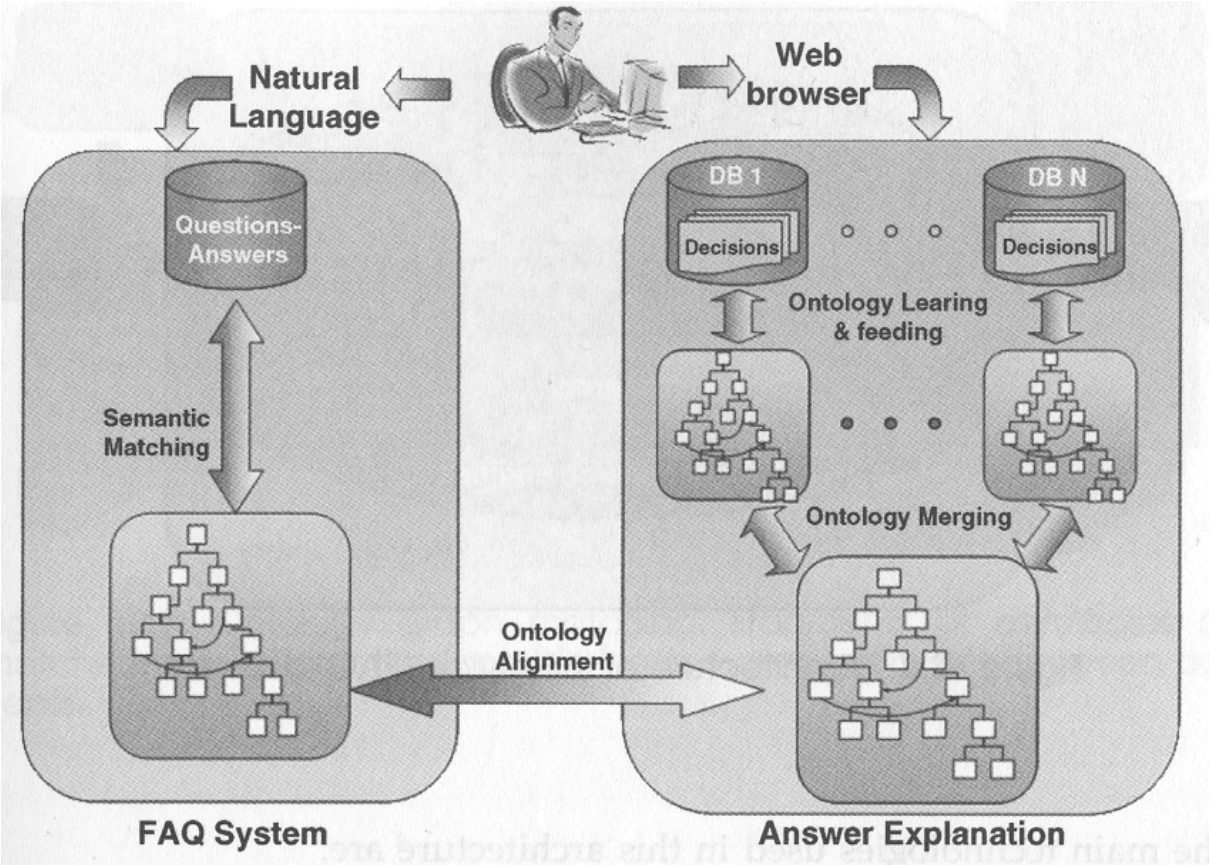
Vervolgens kan het antwoord verduidelijkt worden door de ondersteunende documenten te tonen die bijgedragen hebben tot het antwoord. Het systeem heeft kennis van juridische begrippen (door de ontologie), de ontologie wordt gebruikt om het beste antwoord op een vraag te vinden, daarna worden de verklarende documenten opgehaald. Deze aanpak is terug te vinden in het ontwerp van de architectuur, zie Figuur 10.

- **FAQ System:** Verschillende zoeksystemen die gebruik maken van de juridische ontologie worden gebruik om het antwoord op de vraag te vinden. Dit systeem is gebaseerd op een Multistage search chain, zie Figuur 11.

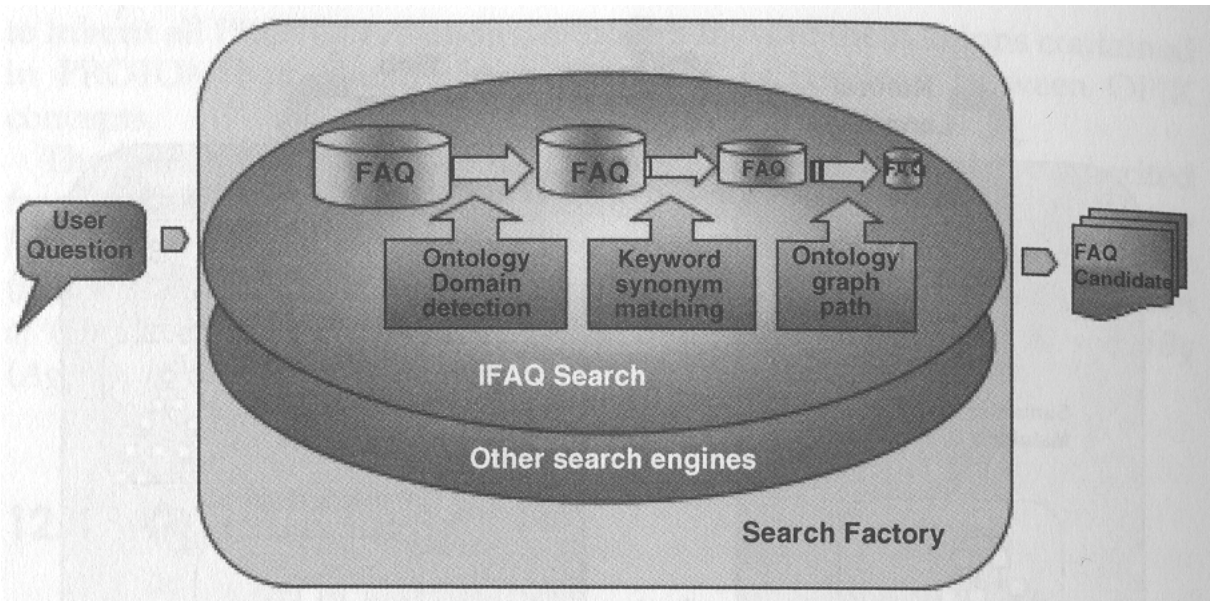
  Ieder onderdeel heeft zijn eigen verantwoordelijkheid. Het eerste onderdeel ontdekt het domein van de vraag (discriminatie, seksuele mishandeling, wetten, etc). In stap twee worden de domeinen eruit gefilterd die van toepassing zijn op de vraag. In de laatste stap wordt de semantische afstand bepaald met de overgebleven resultaten. Vervolgens worden deze resultaten gebruikt voor de ranking van de antwoorden op de vragen.

- **Answer explanation system:** Hierin worden de vragen verduidelijkt door te zoeken in verschillende databases met de semantische zoekmachines. Het gaat hierbij voornamelijk om databases met vonnissen en de beschrijving van het proces. Dit betekent dat de data uit de verschillende systemen eerst wordt geannoteerd met de juridische ontologie. Doordat zowel de vraag als juridische gegevens gerelateerd worden aan dezelfde ontologie kan er beter gezocht worden in de juridische gegevens. Het relateren van twee verschillende soorten gegevens met een en dezelfde ontologie wordt in de literatuur ontology alignment genoemd. In [26] wordt beschreven hoe ontology alignment werkt.

Het resultaat is dat het systeem dezelfde terminologie gebruikt als de rechters. Doordat zowel de vraag als de gegevens gerelateerd kunnen worden aan dezelfde ontologie ontstaat een systeem dat bruikbare resultaten oplevert.



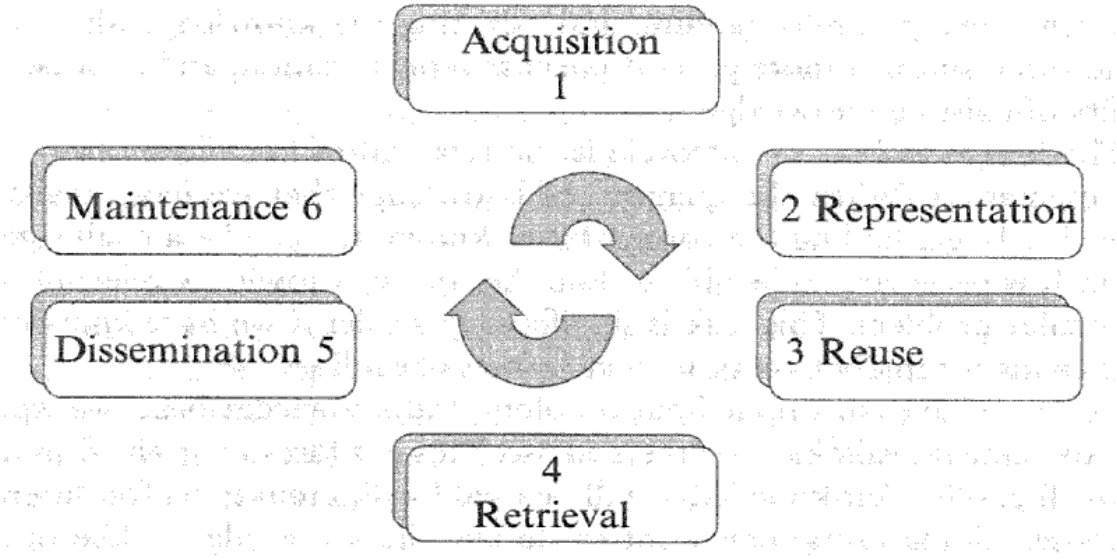
Figuur 10 Architectuur van legal case study.



Figuur 11 Architectuur van het FAQ systeem.

4.3.11 Kennis life-cycle

Het is belangrijk te realiseren dat toepassingen van ontologie technologie kennis gebaseerd toepassingen zijn. Dit soort systemen hebben een speciaal karakter en omdat kennis beheerd moet worden is er een kennis life-cyle gedefinieerd die houvast kan geven aan het ontwerp en beheer proces. Dit proces bestaat uit zes taken die iteratief uitgevoerd worden, zie Figuur 12, zoals wordt beschreven in [4].



Figuur 12 Life-cycle van kennis.

1. **Acquisition:** het verzamelen van de informatie (gegevens) die gerepresenteerd/geclassificeerd moet worden door de te ontwikkelen ontologie.
2. **Representation:** kijken hoe de verzamelde informatie gerepresenteerd kan worden in de te maken ontologie.
3. **Reuse:** het ontwerpen van een ontologie is een tijdrovend en ingewikkeld proces. Daarom is het verstandig om bestaande ontologieën te gebruiken of te combineren. Hergebruik en het combineren van ontologieën is goed mogelijk omdat voor sommige domeinen al ontologieën gedefinieerd zijn. Daarnaast zijn er waarschijnlijk al ontologieën die een zekere overlap hebben met de ontologie die voor een bepaald domein gemaakt gaat worden.
4. **Retrieval:** het uiteindelijke doel van een kennisdatabase is het stellen van vragen, deze vragen kunnen, als het systeem goed ontworpen is beantwoord worden. Bij deze stap moet gekeken worden of de te stellen vragen aan het systeem correcte antwoorden opleveren.
5. **Dissemination:** als de kennisdatabase en ontologie klaar zijn wordt deze beschikbaar gesteld aan de gebruikers.
6. **Maintenance:** het onderhouden van het systeem om de kennisdatabase bruikbaar te houden: hierbij moet gedacht worden aan het up-to-date houden van de opgeslagen gegevens, de correctheid van de gegevens en verificatie van de ontologie (representeert de ontologie nog wel de conceptualisatie). Sommige domeinen zijn onderhevig aan verandering, het is belangrijk dat een ontologie de huidige situatie representeert om bruikbaar te zijn en te blijven.

Een andere belangrijke en veel toegepaste methodiek om ontologien te ontwerpen is de DILIGENT Methode [30].

4.3.12 Aanverwante technologieën van het Semantisch Web

Dit hoofdstuk beschrijft technologiëen die gebruik maken van het Semantisch Web. Hierbij moet gedacht worden aan agents [4] die zelfstandig of in groepsverband het Semantisch Web afspeuren om een gedefinieerd probleem op te lossen. Daarnaast worden webservices behandeld die gebruik maken van technologieën die onderdeel zijn van het Semantisch Web.

4.3.13 Agency and reasoning

Een ontologie maakt het mogelijk om kennis te representeren zodat de computer de betekenis begrijpt. In de visie over het Semantisch Web wordt gesproken over agents. Agents zijn autonome programma's die kennis consumeren en hierover rationeel kunnen redeneren. Agents zijn dus software programma's die het Semantisch Web kunnen doorzoeken om antwoord te vinden op vragen of die bepaalde taken kunnen uitvoeren. De moeilijkheid bij het maken van een agent is dat deze onafhankelijk beslissingen moet kunnen nemen, dit lijkt sterk op de manier zoals mensen redeneren en handelen. Dit is anders dan normale programma's waarin precies wordt vastgelegd wat software doet.

Als mensen redeneren zijn er twee belangrijke dingen die onbewust plaats vinden:
- naar welke toestand willen we toe, dit kan ook gezien worden als een verzameling van voornemens.
- hoe willen we deze toestand gaan realiseren, dit zal uiteindelijk moeten resulteren in een actieplan.

Om het menselijk redeneren na te bootsen worden methodieken gebruikt die gebaseerd zijn op logica. De computer kan dus redeneren door gebruik te maken van logica. Een populaire implementatie hiervan is bekend als de Belief-Desire-Intention methodiek.

- **Beliefs:** de kennis die een agent heeft van het domein waarin hij opereert (ontologie + geclassificeerde gegevens).
- **Desire:** de toestanden die bepalen welke doelen de agent heeft en welke resultaten bereikt moeten worden.
- **Intentions:** de toestanden waar de agent zich aan zal toewijden, het is namelijk niet altijd mogelijk alle voornemens (desires) te verwezenlijken.

Agents kunnen alleen opereren, maar indien ze in groepen opereren wordt er gesproken over Multi-agent systems of socities of agents. Om een dergelijke Multi-agent systeem te realiseren moeten de agents met elkaar kunnen communiceren. Voor zulke communicatie zijn er protocollen gedefinieerd.

4.3.14 Kennis services

De visie van het Semantisch Web omvat drie technische componenten (technologieën), namelijk ontologieën, agents en kennis services [7, 4]. Een ontologie wordt gebruikt om

kennis te representeren; vervolgens kan een agent deze ontologieën gebruiken om over kennis te redeneren. Een kennis service kan benaderd worden om kennis te verkrijgen (kennisdatabase te bevragen) of kennis te transformeren (met kennis redeneren). Deze kennis services worden meestal geïmplementeerd door gebruik te maken van webservice technologieën. Semantische webapplicaties kunnen geconstrueerd worden door kennis services te combineren. Het combineren van verschillende services zou automatisch moeten kunnen gebeuren door één of meerdere agents.

4.4 Structurering van gegevens

Gegevens kunnen ingedeeld worden aan de hand van drie begrippen, namelijk semantiek, structuur en schema [5].

Bij kennis (gegevens) representatie gaat het erom om een deel van de wereld (probleemdomein) te modelleren. Zulke formele modellen worden ontologieën genoemd. Een ontologie kan gebruikt worden om semantische gegevens toe te voegen aan data zoals: databases, catalogussen, documenten, webpagina's, etc.

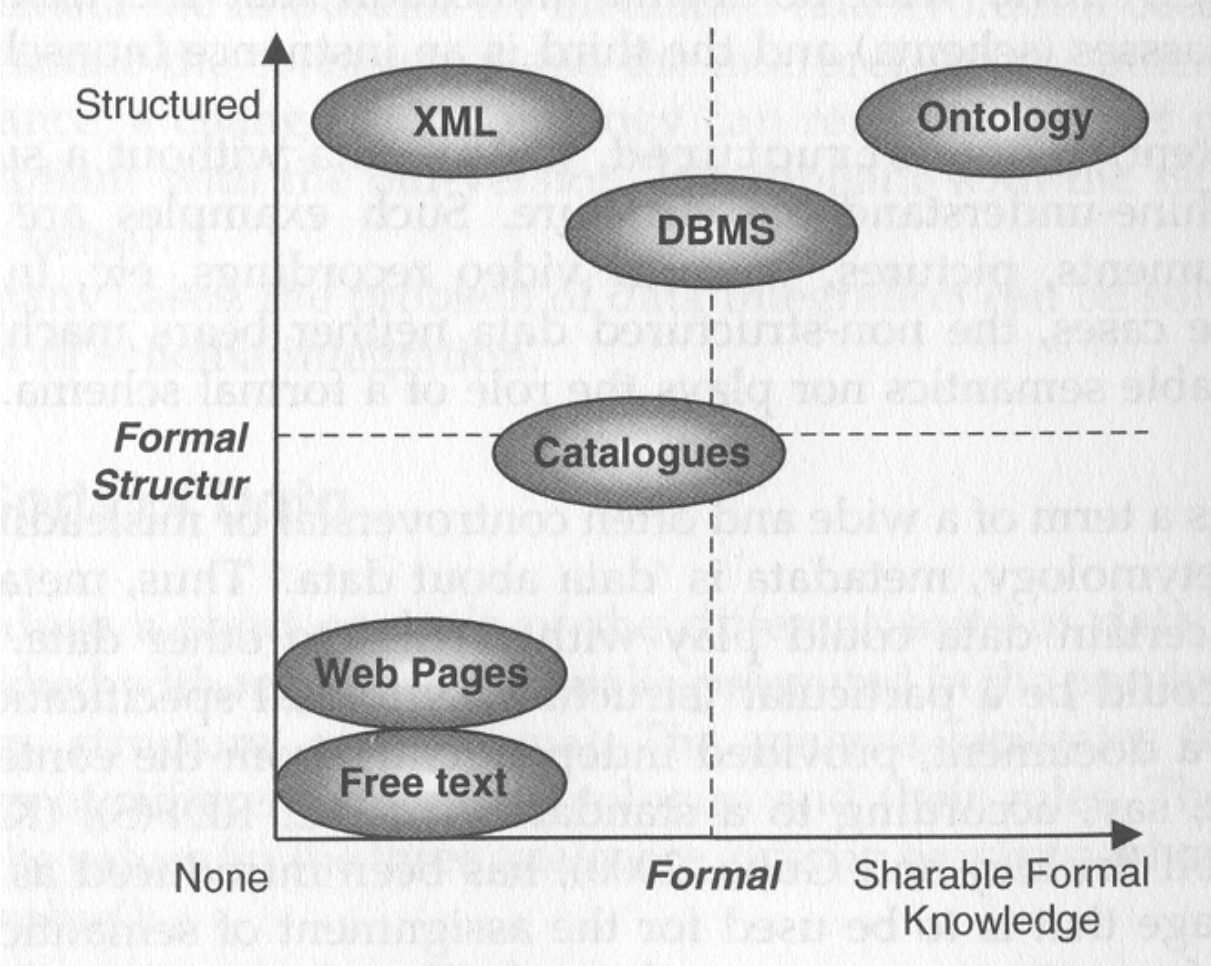
Semantische databronnen kunnen overweg (opslaan, bevragen en beheren) met gestructureerde data. Daarnaast bieden zulke semantische database systemen andere analytische mogelijkheden (inferentie).

Semantiek: semantiek legt de betekenis van de data formeel vast.
Structuur: of er een formele structuur in de data zit.
Schema: het formaat waarin de gegevens opgeslagen kunnen worden.

Dit betekent echter nog niet dat het conform een bepaalde structuur opgeslagen moet worden. Indien de structuur van de gegevens vastgelegd wordt is er sprake van een schema. Voorbeelden van schema's zijn relationele schema's, xml schema's en RDFS.

Nu volgt een indeling van gegevens aan de hand van bovengenoemde eigenschappen.
Data(_, _, _): alle soorten gegevens
Datasets(_, structured, _): alle gegevens die opgeslagen zijn volgens bepaalde structuur
Knowledgebase(semantic, structured,_): iedere dataset met een gedefinieerde betekenis (semantiek). Worden in de literatuur vaak instance datasets of instance knowledge genoemd.
Ontologies(semantic, structured, schema): een ontologie beschrijft zowel een structuur als de semantiek. Een ontologie kan een object beschrijven en de betekenis.
Nonsemantic schemata (nonsemantic, structured): database en XML schemata.
Databases (nonsemantic, structured, schema): hier worden databases gezien als relationele databases, xml-bestanden, comma-separated files en andere gestructureerde data.

Een databron kan gestructureerd zijn of ongestructureerd. Een goed voorbeeld van een gestructureerde databron is XML, een minder gestructureerde databron is een tekstdocument. Formele kennis is de mate waarin er over data geredeneerd kan worden. Met een ontologie is het mogelijk om te redeneren door gebruik te maken van formele logica. Met een tekstbestand wordt het al moeilijk of onmogelijk om daar een computer over te laten redeneren. In Figuur 13 zijn verschillende databronnen afgezet aan de hand van deze twee eigenschappen.



Figuur 13 De structuur en semantische positie van verschillende soorten gegevens.

Structuur van data: het idee is, dat in databases de data is opgeslagen in een strikte structuur (relaties, attributen en records). Een ontologie wordt gerepresenteerd door middel van een graaf die de structuur van de data vastlegt. Het grote voordeel van een graaf model ten opzicht van een database is dat relaties dynamisch aangemaakt en opgeslagen kunnen worden, hierdoor ontstaan zogenaamde dynamische relatie netwerken (semantisch netwerk), waarmee het mogelijk is om ad-hoc relaties aan te brengen en te veranderen.

4.5 Problemen met ontologie technologie

Het probleem van ontologieën is dat de relaties tussen objecten handmatig aangebracht moeten worden en dit is een kostbaar en tijdrovend proces. Dit is dus een groot probleem voor de bruikbaarheid van ontologieën in de praktijk. Momenteel is het automatisch genereren van ontologieën uit bestaande data een academisch vraagstuk.

Echter er blijven nog twee problemen met ontologieën, namelijk:

- het analyseren van documenten
- het maken van queries.

Een persoon die informatie wil vinden gebruikt waarschijnlijk een andere ontologie (vocabulaire) dan het systeem gebruikt. Dit betekent dat verschillende mensen andere

woorden, concepten en associaties gebruiken. Daarnaast zijn woorden vaak context gerelateerd, afhankelijk van de query en de verwachtingen van de gebruiker. Hieruit valt te concluderen dat ontologieën eigenlijk alleen bruikbaar zijn voor het zoeken in gesloten systemen waarbij de vocabulaire klein is.

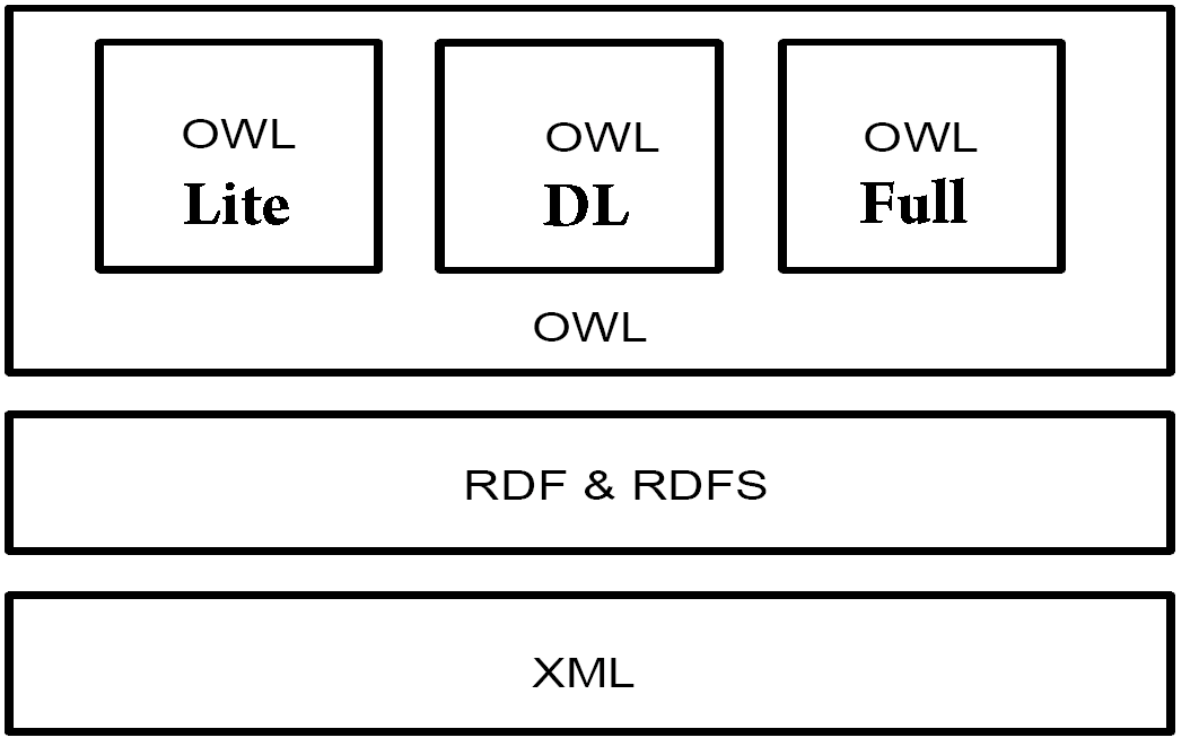
5 Ontologie technologie in het kort

Dit hoofdstuk beschrijft de toepassingsgebieden van ontologie technologie. Uiteindelijk wordt er gekeken of de RDF technologie toepasbaar is op de vraagstelling zoals gedefinieerd in 2. Deze vraagstelling is gedetailleerder geformuleerd in de opdrachtomschrijving en het requirements document [1, 2]. Eerst wordt een beschrijving gegeven van RDF technologie en bijbehorende toepassingsgebieden. Vervolgens wordt een alternatieve oplossing beschreven die beter aansluit bij de vraagstelling.

RDF moet gezien worden binnen de context van ontologie technologieën. Met een ontologie kan een conceptualisatie gemaakt worden van een domein. De conceptualisatie van een domein kan beschreven worden met behulp van RDFS. Instanties van de concepten (classes) kunnen beschreven worden met RDF.

Een dergelijke conceptualisatie (ontologie) is een representatie van kennis uit een probleem domein. De conceptualisatie tezamen met de instanties is een kennisdatabase. Deze kan dan bevraagd worden met een speciale query taal, zoals SPARQL. Het is ook mogelijk om over kennis te redeneren (inferencing) waarmee indirecte kennis afgeleid wordt uit aanwezige kennis. Dit rederenen met kennis is mogelijk omdat ontologie talen gebaseerd zijn op (descriptieve) logica [4, 5].

Op RDF zijn verschillende ontologie talen gebaseerd, namelijk de OWL talen in drie varianten (Lite, DL en Full). De ontologietalen uit de OWL familie zijn gebaseerd op RDF. RDF is gebaseerd op XML, zie Figuur 14. RDFS kan niet gezien worden als een ontologie taal omdat het te weinig uitdrukkingskracht heeft [4, 5]. Het is voornamelijk bedoeld om hiërarchische structuren aan te brengen tussen concepten. RDF(S) kan gebruikt worden om een taxonomie mee te beschrijven. Het is dan misschien ook beter om RDF(S) niet te zien als ontologie taal maar meer als de bouwsteen voor de OWL ontologie talen familie.



Figuur 14 OWL is gebaseerd op RDF/RDFS wat weer gebaseerd is op XML. RDF/RDFS kunnen dan ook gezien worden als de bouwstenen voor de ontologie talen uit de OWL familie.

Het voornaamste doel van RDF(S) is om concepten te identificeren die we willen representeren in de kennis database en hiërarchische relaties aanbrengen tussen de concepten. RDFS is gebaseerd op een class-hiërarchie, het is mogelijk om verbanden aan te brengen tussen verschillende classes (concepten) door middel van eigenschappen [4].

Met RDFS is het mogelijk een simpele ontologie te definiëren (officieel geen ontologie). De instanties van objecten en relaties tussen de objecten worden vastgelegd met behulp van RDF.

5.1 Toepassingsgebieden van ontologieën

Ontologie technologie wordt momenteel gebruikt in enkele toepassingsgebieden [9]:

- data integratie (biomedische sector)
- kennissystemen, meestal voor Intranet applicaties
- e-Science met name life-sciences

Het is belangrijk om te realiseren dat tot nu toe de succesvolle applicaties gemaakt zijn voor gesloten gemeenschappen.

Bij data integratie wordt het mogelijk om verschillende schema's te mappen naar een ontologie zodat een uniform beeld ontstaat van de verschillende databronnen. Voor kennissystemen is het mogelijk om bepaalde data te relateren aan een ontologie en hierop slimme zoekfunctionaliteit toe te passen [7]. In e-Science is eigenlijk sprake van data integratie op grote schaal omdat uit veel verschillende databronnen gegevens gecombineerd dienen te worden. Ontologieën kunnen ook gebruikt worden voor slimme webservices [5],

hiermee moet het mogelijk worden dat agents automatisch de juiste webservices kunnen samenstellen en configureren om een probleem op te lossen.

In [7, 4] worden meerdere praktijksituaties beschreven die gebaseerd zijn op ontologie technologie. Het gaat hier voornamelijk om systemen die informatie (documenten) classificeren aan de hand van een ontologie, hierdoor wordt het mogelijk om slimmer te zoeken dan met traditionele zoekmachines. De informatie komt uit verschillende bronnen en er is dus meestal ook sprake (op semantisch niveau) van data integratie bij deze projecten.

In [9] worden vier vragen gesteld die betrekking hebben op de bruikbaarheid van ontologie technologie. De antwoorden op deze vragen kunnen geïnterpreteerd worden als de huidige status van ontologie technologie. Uit het antwoord blijkt onder andere dat er nog een aantal hobbels overwonnen moeten worden voordat ontologie technologie volwassen is.

5.2 De opdracht

‘Tussen de regels door’ is de naam van de afstudeeropdracht [1]. Zowel uit de opdracht omschrijving als de functionele specificatie [2] komt de wens naar voren dat bepaalde patronen vroegtijdig opgespoord worden. Daarnaast moet het makkelijker worden om nieuwe gegevens (relaties) toe te voegen. Deze zijn momenteel moeilijk vast te leggen door de complexiteit van het datamodel en de applicatie van Kidos.

5.3 Complexiteit

Uit [7] blijkt dat het ontwerpen, onderhouden en gebruiken van ontologieën moeilijker is dan van een relationeel datamodel. De complexiteit van een relationeel model neemt lineair toe in verhouding tot de omvang. Bij ontologieen is de complexiteit groter dan bij het relationele model (niet lineair). De kennis die nodig is om applicaties te ontwikkelen, die gebruik maken van ontologie technologie, is niet of onvoldoende aanwezig bij softwareontwikkelaars.

Datamining [14] is de richting in de informatica die zich bezig houdt met het analyseren van data. Dit betekent dat zowel classificatie als identificatie van klassen gerealiseerd kunnen worden met datamining technologieën. Met dataminingtechnologie is het mogelijk om patronen in gegevens te ontdekken en deze te gebruiken voor classificatie. Ook is het mogelijk om zelfgedefinieerde patronen te gebruiken als classificatie criteria [6]. Datamining wordt meestal toegepast op een datawarehouse [15]. In een datawarehouse wordt vaak het dimensional databasemodel (ster model) gebruikt. Dit model is geschikt voor het analytisch verwerken en opslaan van gegevens [8].

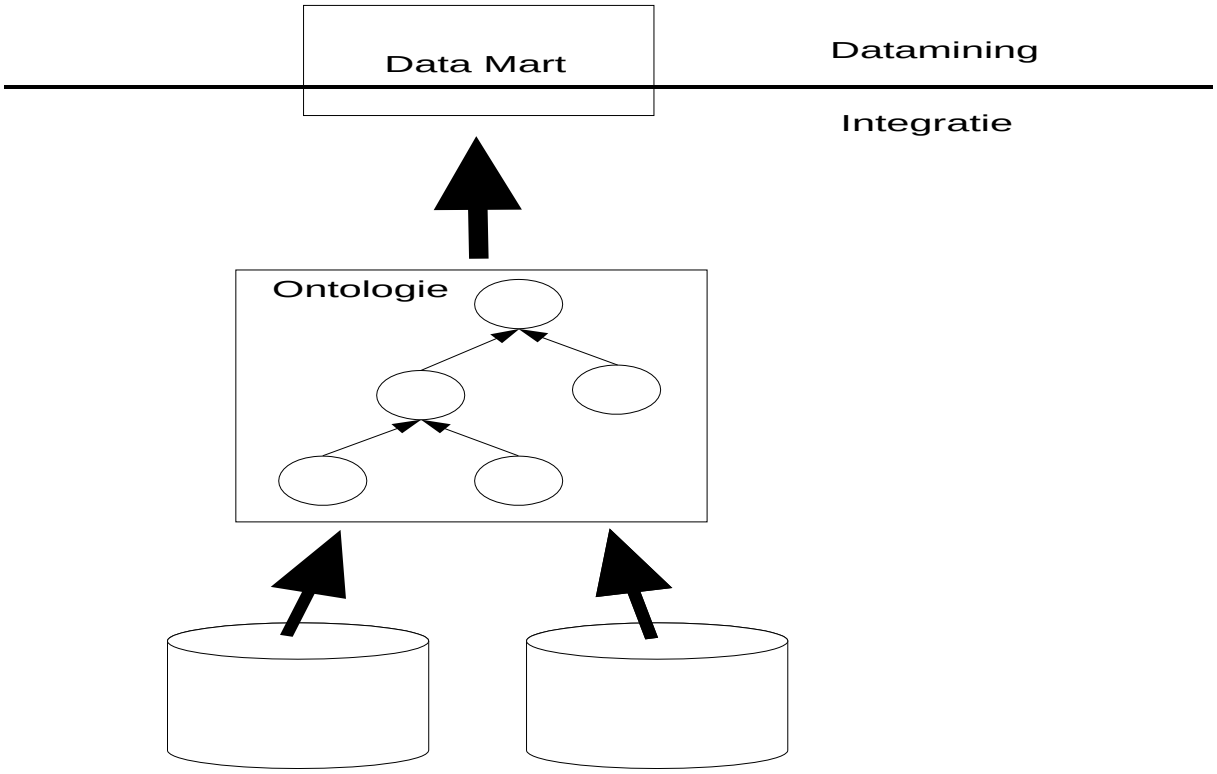
5.4 Toepassingen voor Kidos

Het zal intressant zijn om naar een aantal potentiële toepassingen van RDF te kijken binnen het Kidos Project:

- Data integratie door middel van een ontologie
    - Het onderzoeken van hoe data integratie kan plaats vinden bij een datawarehouse die gevuld wordt vanuit verschillende databronnen.

    - Het integreren van data uit meerdere bronnen om de operationele database te vullen.
- Het maken van een ontologie (conceptueel domein model van Kidos) om dit in de toekomst te kunnen gebruiken voor applicaties en het probleem domein te verduidelijken door middel van de conceptualisatie.

Bij het integreren van data bronnen moeten er wel meerdere databronnen zijn. Voor zover ik kan overzien is dat momenteel niet het geval binnen Kidos. Dit zou misschien in de (nabij) toekomst kunnen veranderen. Er moet dan ook gerealiseerd worden dat data integratie niet alleen interessant is voor Kidos maar ook voor andere projecten. Zie Figuur 15 voor een mogelijke toepassing van ontologie technologie binnen de data-integratie. Ontologie technologie maakt het mogelijk om data uit verschillende schema's te mappen naar één ontologie. Deze ontologie kan dan gezien worden als het gemeenschappelijke schema van de verschillende databronnen.



Figuur 15 Een datamart wordt gevuld vanuit twee databronnen. De ontologie zorgt voor Schema integratie, waardoor de data op een uniforme manier gerepresenteerd wordt.

5.5 Conclusie

De opdrachtomschrijving en het requirements document suggereren dat de problemen die o.a. opgelost moeten worden te maken hebben met het analyseren en classificeren van data uit de Kidos omgeving. Bijvoorbeeld het identificeren en classificeren van kinderen die in aanmerking komen voor voorschoolse opvang. Ontologie technologie is hiervoor niet bedoeld. Dit soort vragen kunnen beantwoord worden door gebruik te maken van datamining technologie.

Het eenvoudig aanbrengen van relaties is niet makkelijker in een ontologie dan in relationele database systemen. Met andere woorden een goed ontwerp blijft cruciaal voor zowel een relationeel model als een ontologie. Een ontologie technologie (gecombineerd met een relationeel data model) kan niet de tekortkomingen in het ontwerp van het relationele systeem verhelpen.

Daarnaast is het ook nog zo dat ontologiesystemen ingewikkelder zijn om te ontwerpen, teimplementeren en te onderhouden (dan relationele systemen). De wetenschappelijke gemeenschap heeft nog niet een passend antwoord gevonden op belangrijke vragen die ontologie technologie veel meerwaarde geven ten opzichte van relationele systemen. Daarnaast zijn er nog een aantal hobbels [16] die overwonnen moeten worden voordat de technologie volwassen is.

De meeste toepassingen van ontologie technologie hebben betrekking op het verbeteren van zoekfunctionaliteit in bestaande data. Door de bestaande data aan een ontologie te relateren wordt het mogelijk om slimmer te zoeken en data te combineren uit verschillende bronnen omdat de data aan dezelfde of een gecombineerde ontologie wordt gerelateerd. Ontologie technologie is dan ook geschikt voor kennis portals zoals het Juridische portaal en het bibliotheken portaal. Andere toepassingen staan nog in de kinderschoenen. Ook moet er afgewacht worden of ontologie technologie en het Semantisch Web in het algemeen echt een succes zullen worden.

Concluderend: ontologie technologie heeft enkele toepassingsgebieden, de opdrachtomschrijving komt niet overeen met een van deze toepassingsgebieden. Datamining technologie biedt wel de mogelijkheid om een deel van de problemen op te lossen. Het komt er op neer dat de juiste technologie en methoden gekozen moeten worden om de problemen op te lossen.

6 Data Integratie

Ontologie technologie wordt meestal ingezet als er ook sprake is van data integratie. Dit betekent dat meerdere databronnen geïntegreerd worden door de gegevens te mappen naar een globale ontologie, zie voorbeelden kennis systeem voor rechters §4.3.10.2 en digitale bibliotheek §4.3.10.1. In beide systemen wordt gebruik gemaakt van verschillende databronnen die geannoteerd worden met semantische data. Hierdoor is het mogelijk dat dezelfde vocabulaire wordt gebruikt, ook al is de data afkomstig uit verschillende systemen.

Data integratie vanuit verschillende systemen is ook op andere manieren mogelijk dan door gebruik te maken van ontologie technologie. In het volgende hoofdstuk zullen andere manieren van data integratie kort besproken worden. Een methode wordt uitvoerig geïllustreerd aan de hand van een case studie die twee lokale database schema’s samenvoegt tot een globaal database schema. Het verschil tussen deze methode en een ontologie methode is het volgende:

- Bij de ontologie methode wordt integratie gerealiseerd door de data te mappen naar een vocabulaire ontologie. Bij de andere methode wordt gebruik gemaakt van schema integratie. Dit betekent dat ontologie data gekoppeld wordt op betekenis en bij schema integratie op structuur.

Beide methoden kunnen gebruikt worden om hetzelfde soort probleem op te lossen, namelijk data integratie. Voor de volledigheid van dit onderzoek wordt daarom ook de methode beschreven die lokale database schema's kan integreren tot een globaal schema.

7 Ontologie technologie en federatief database systemen.

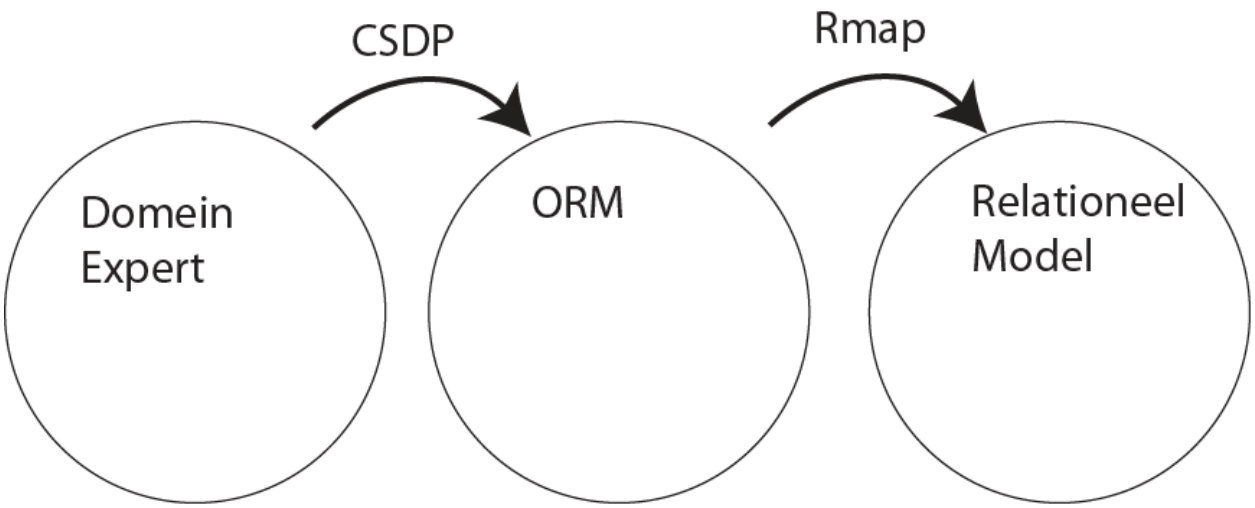
In de voorgaande hoofdstukken is gekeken naar de toepasbaarheid van ontologie technologie. Dit is o.a. toepasbaar op data integratie. Echter bij ontologie technologie wordt data integratie toegepast met als doel het maken van een kennis database systemen. Een kenmerk van kennis database systemen is dat ze meestal niet geschikt zijn voor operationele doeleinden waarvoor de meeste gewone databasesystemen worden ingezet. Operationele database systemen worden meestal ingezet om een real-time process te ondersteunen. Gegevens die zich in dergelijke database systemen bevinden representerern de actuele toestand van de werkelijkheid. In dergelijke systemen is het niet mogelijk of gewenst data te annoteren zoals bij kennis database systemen gebeurd. Willen we echter toch meerdere database systemen die voor opertionele doeleinden gebruikt worden integreren dan is ontologie technologie geen geschikte kandidaat.

Voor data integratie uit verschillende heterogene (operationele) database systemen worden federatieve database systemen gebruikt. Een dergelijke systeem maakt het mogelijk om meerdere en verschillende (heterogene) databasesystemen te ontsluiten met behulp van één globaal database schema. Echter om een federatief database systeem effictief te kunnen gebruiken is het noodzakkelijk om een globaal database schema te ontwerpen. Een globaal databaseschema kan worden opgevat als een verzameling van de lokale database schema's die het federatieve database systeem ontsluit. Het ontwerpen van een dergelijk schema kan gedaan worden aan de hand van een methode zoals beschreven staat in [17]. Het ontwerpen van een database schema dient te geschieden met behulp van een methode die ondersteunt wordt door visuele representatie. Een dergelijke methode en visualisatie biedt het Object Role Model. ORM wordt vervolgens gebruikt als tool om een globaal database schema te ontwerpen.

In hoofdstuk 8 volgt een inleiding over Object Role Model. Deze inleiding moet de lezer in staat stellen om hoofdstuk 9 te begrijpen waarin wordt behandeld hoe schema integratie plaats vindt met behulp van ORM. Voor een gedetaileerde bespreking van database schema ontwerp met behulp van ORM kan [17] geraadpleegd worden.

8 Inleiding - Object Role Model

Dit hoofdstuk behandeld de basis notatie van Object Role Model (ORM) en een bijbehorende methode Conceptual Design Procedure (CDSP) waarmee het mogelijk is om een conceptueel model te ontwerpen. Vervolgens kan een dergelijk ORM model omgezet worden in een equivalent relationeel model door middel van de RMap procedure. Object Role Model (ORM) is een grafische notatie in combinatie met een methodiek om een conceptueel model te maken van het Universe of Discource (UoD).
Een functionele specificatie van het te modelleren domein (UoD) wordt gebruikt als input voor het CSDP om een conceptueel ORM model te ontwerpen. Het opstellen van een functioneel ontwerp voor het datamodel is eigenlijk niet noodzakkelijk. De reden hiervoor is dat het doorlopen van de CSDP met de domein expert de input levert waarmee het ORM model afgeleid kan worden. Het functioneel ontwerp wordt onderdeel van het ORM model. Een ORM model valt eenvoudig te verwoorden in natuurlijke taal zodat een ORM model geverifieerd kan worden door de domein expert. ORM biedt dus de mogelijkheid om communicatie georiënteerd te ontwerpen. Het verkregen ORM diagram is een conceptuele representatie van het UoD. Hierin worden de verschillende concepten uit het UoD aan elkaar gerelateerd. Het grote voordeel van ORM is dat de (CSDP) methodiek een gestructureerde manier biedt om een database schema te modelleren. Bij UML en ER modelleren is de stap van functionele specificatie naar ontwerp onduidelijk. Dit wordt meestal gedaan op gevoel en het succes is vaak afhankelijk van de ervaring van de modelleur. Daarnaast is het ook moeilijke dergelijke modellen te controleren met de domeinexpert omdat deze notaties moeilijk te verwoorden zijn in natuurlijke taal. Een andere belangrijk pluspunt ten opzicht van UML en ER is dat ORM een expressievere notatie heeft. Dit maakt het mogelijk om constructies die in UML en/of ER diagrammen niet gemodelleerd kunnen wel in ORM gemodelleerd kunnen worden. Een ORM diagram kan snel omvangrijk worden in tegenstelling tot een UML/ER diagram, dit wordt meestal gezien als een nadeel maar wanneer je het UoD precies wilt vastleggen dan ontkom je daar niet aan. Het volledig in detail vastleggen van het UoD kan het beste met ORM!
UML en ER diagrammen zijn modelleer technieken die zich op implementatie niveau bevinden en ORM op conceptueel niveau. Er wordt dus in een hoger abstractie niveau gemodelleerd met ORM dan met ER en UML. Een ORM model wordt door middel van een algoritmische methode omgezet in een relationeel database schema waarin geen redundantie voorkomt! Dit wordt gegarandeerd door de manier waarop het CSDP proces werkt en de RMap procedure.
Het proces om een ORM model af te leiden is gevisualiseerd in Afbeelding 1.

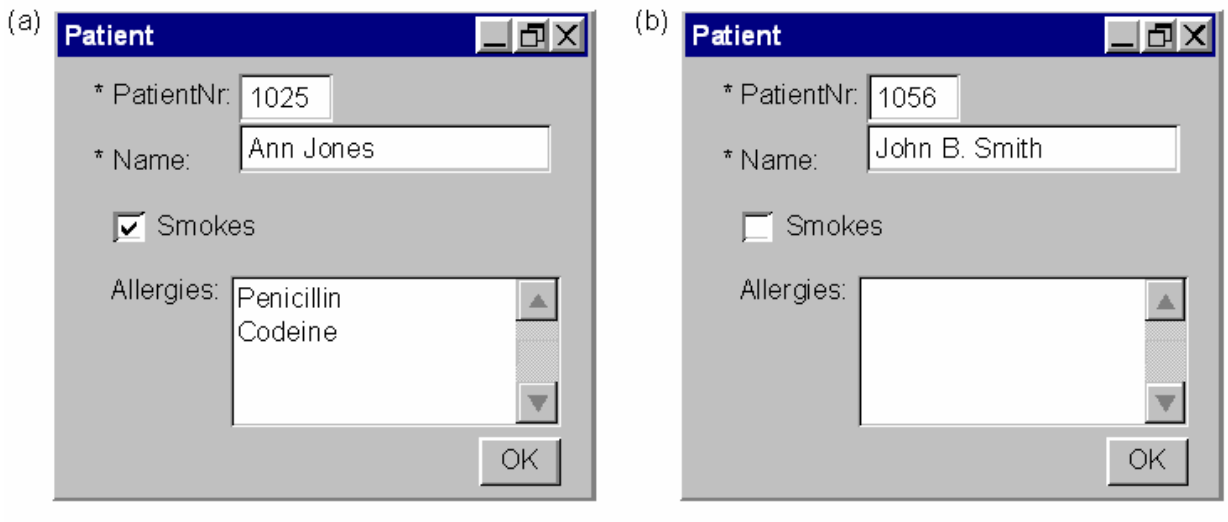


Afbeelding 1 Het proces van het database schema's ontwerpen met behulp van een ORM model.

Aan de hand van de onderstaande specificatie zal het ORM schema afgeleid worden. Bij het afleiden wordt een groot deel van het CSDP doorlopen. Echter niet iedere stap wordt volledig gedaan of behandeld omdat dit buiten het bereik van deze inleiding valt. Alleen de basiselementen uit ORM worden behandeld. Er wordt ook niet expliciet aangegeven welke stap wanneer uitgevoerd wordt.
Voor meer geavanceerde concepten en een volledige beschrijving van ORM doet de lezer er goed aan [19] te raadplegen. Na het lezen van deze inleiding moet de lezer in staat zijn zelf een simpel ORM schema te ontwerpen en te kunnen begrijpen wat een ORM schema communiceert (betekend).

8.1 Specificatie van Voorbeeld

Het te bouwen systeem moet gegevens registeren in een ziekenhuis. De tekst hieronder is het resultaat van het gesprek met de domein expert over het UoD.
Er moet worden bijgehouden welke aandoeningen er behandeld kunnen worden in het ziekenhuis. Een behandeling wordt uitgevoerd door één verantwoordelijke arts. Soms gebeurd het dat een complicatie optreed bij een behandeling van een aandoening. Een dergelijke complicatie wordt geïdentificeerd door middel van de medische naam.
Patiënten ondergaan behandelingen bij een arts voor een bepaalde aandoening. Van een patiënt wordt bijgehouden of deze rookt ja of de nee. Ook kan het gebeuren dat een patiënt overlijd.
De overige informatie is gevisualiseerd in Afbeelding 2 een screenshot van de UI die de domeinexpert voor ogen had.
Gedurende het ontwerpen van het ORM diagram in de volgende hoofdstukken kunnen wijzigingen en nieuwe inzichten optreden in dat ontwerp. Naarmate de modelleur en de domeinexpert het te modelleren domein beter begrijpen zal het ontwerp door voortschrijdend inzicht veranderen.



Afbeelding 2 Voorbeeld scherm van de te bouwen applicatie. Twee verschillende instanties van patiënten informatie.

8.2 Elementaire feiten en de structuur

De eerste stap is het identificeren van elementaire (statement) uitspraken. Een elementaire uitspraak is niet verder op te delen in kleinere uitspraken zonder dat er informatie verloren gaat. In Afbeelding 2 staat informatie over patiënten. Deze informatie is te beschrijven met de volgende uitspraken:

1. Er is een patiënt Ann Jones met patiëntnummer 1025 die rookt en allergisch is voor Penicilline en Codeïne.
2. Er is een patiënt John B. Smith met patiëntnummer 1056 die niet rookt en geen allergieën heeft.

Uitspraken 1 en 2 zijn niet elementair, omdat de uitspraken op te splitsen zijn in meerdere uitspraken die tezamen dezelfde informatie communiceren. De volgende uitspraken zijn een opsplitsingen van uitspraken 1 en 2:

3. Er is een patiënt Ann Jones met patiëntnummer 1025.
4. Patiënt 1025 rookt.
5. Patiënt 1025 is allergisch voor medicijnen: Penicilline en Codeïne.
6. Er is een patiënt John B. Smith met patiëntnummer 1056.
7. Patiënt 1056 rookt niet.
8. Patiënt 1056 is niet allergische voor medicijnen.

De vraag is dan of de uitspraken nu wel elementair zijn. Dat is niet zo, uitspraak 5 moeten worden opgedeeld. De uitspraken 9 t/m 17 zijn elementair, dat is ze zijn niet verder opdeelbaar zonder informatie verlies. Uitspraak 9 is een speciaal geval omdat dit het bestaan vertelt van een patiënt met een bepaald nummer. Een dergelijke uitspraak wordt een existentieel feit genoemd. Het patiëntnummer wordt vervolgens gebruikt om een patiënt uniek te identificeren.

9. Er is een patiënt met patiëntnummer 1025.
10. Patiënt 1025 heeft naam Ann Jones.
11. Patiënt 1025 rookt.
12. Patiënt 1025 is allergisch voor medicijn Penicilline

13. Patiënt 1025 is allergisch voor medicijn Codeïne.
14. Er is een patiënt patiëntnummer 1056.
15. Patiënt 1056 heeft naam John B. Smith.
16. Patiënt 1056 rookt niet.
17. Patiënt 1056 is niet allergische voor medicijnen.

Nu alle uitspraken elementair zijn kunnen de uitspraken met dezelfde betekenis samengenomen worden om er structuur in aan te brengen. Het resultaat van uitspraken 9 t/m 17 zijn de uitspraken 18 t/m 21.

18. Er is een patiënt met <patiëntnummer>. (9 en 14)
19. Patiënt <patiëntnummer> heeft naam <patiëntnaam>. (10 en 15)
20. Patiënt <patiëntnummer> rookt <wel/niet>. (11 en 16)
21. Patiënt <patiëntnummer> is allergisch voor <medicijn>. (12, 13 en 17)

Nu de structuur is aangebracht in de uitspraken is het verstandig om de gegevens erbij te zetten, dit is te zien in uitspraken 22 t/m 25.

22. Er is een patiënt met <patiëntnummer>. (9 en 14)
    1. 1025
    2. 1056
23. Patiënt <patiëntnummer> heeft naam <patiëntnaam>. (10 en 15)
    1. 1025 Ann Jones
    2. 1056 John B. Smith
24. Patiënt <patiëntnummer> rookt <wel/niet> (11 en 16)
    1. 1025 wel
    2. 1056 niet
25. Patiënt <patiëntnummer> is allergisch voor <medicijn>. (12, 13 en 17)
    1. 1025 Penicilline
    2. 1025 Codeïne

Nu er structuur is aangebracht in de uitspraken kan een ORM diagram getekend worden. In een dergelijk diagram wordt de structuur zoals in de uitspraken 22 t/m 25 gevisualiseerd. Een dergelijke visualisatie geeft een beter inzicht in hoe de concepten (patiënten, medicijnen, roken) aan elkaar gerelateerd zijn.

8.3 Afbeelden van Concepten, waarden en rollen

Als we kijken naar uitspraak 22 hebben we het over een patiënt die geïdentificeerd wordt door middel van een patiëntnummer. Uitspraak 22 communiceert het bestaan van een concept Patiënt en instanties (lees database records) van het concept Patiënt worden uniek geïdentificeerd door middel van patiëntnummer. Een dergelijk concept, zoals Patiënt, wordt in een ORM diagram weergegeven met behulp van een rechthoek met afgeronde hoeken. Indien instanties van een dergelijk object uniek geïdentificeerd kunnen worden met behulp van één waarde wordt deze identificatie tussen haakjes onder de naam van het object geplaatst. Het label van de identificatie in ons voorbeeld is patiëntnummer. Omdat dit een wat lang label is voor de identificatie wordt dit genoteerd als ".nr". Waarbij de punt staat voor de naam van het Object in dit geval "Patiënt". Dus ".nr" is eigenlijk "Patiëntnr.". Zie Afbeelding 3 voor de

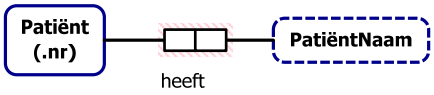
visuele weergave van een object (concept) in ORM en het bijbehorende identificatie (patiëntnr).

Afbeelding 3 Het patiënt object (concept) met bijbehorende identificatie patiëntnr, afgekort als .nr.

Een patiënt heeft in ons voorbeeld een patiëntnaam. Een patiëntnaam is geen concept (object) zoals een patiënt maar een waarde die een bepaalde relatie (rol) heeft met het concept Patiënt. In ORM wordt een dergelijke waarde aangegeven met behulp van een rechthoek met afgeronde hoeken en een stippellijn. Een dergelijke waarde heeft een type, bijvoorbeeld string, int, etc. Een dergelijk attribuut (waarde) is afgebeeld in Afbeelding 4.

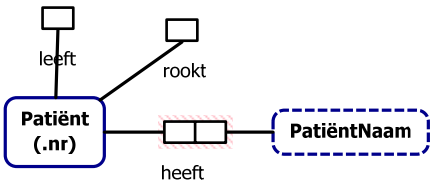
Afbeelding 4 Attribuut (waarde) patiëntnaam, wat niet te zien is in de afbeelding is dat patiëntnaam van het type string is.

Om de relatie (rol) tussen het attribuut PatiëntNaam en Patiënt aan te geven wordt er gebruik gemaakt van een verbinding (relatie) zoals afgebeeld in Afbeelding 5. In ORM wordt een dergelijke relatie een rol genoemd. Om het doel van de rol te verduidelijk wordt er een label gebruikt.



Afbeelding 5 Relatie tussen Patiënt en PatiëntNaam wordt weergegeven met behulp van een verbinding met daartussen twee rechthoekjes, de rol die dergelijke objecten/attributen met elkaar aangaan wordt verduidelijkt door een label. In dit specifieke geval is het label “heeft”.

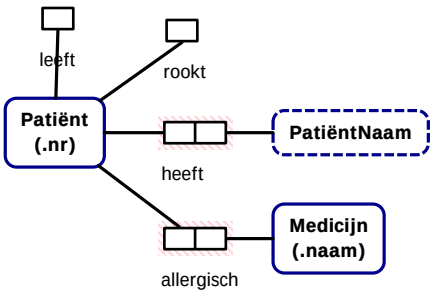
Uitspraak 24 communiceert dat een patiënt wel of niet rookt. Een dergelijk feit kan worden afgebeeld als een unaire-rol (één klein rechthoekje). Unaire feiten worden meestal gebruikt in situaties waarin er sprake is van één simpele eigenschap, namelijk of een eigenschap wel of niet van toepassing is; er is dus altijd sprake van een booleaanse waarde. Een ander goed voorbeeld van een unaire-rol is het feit of een patiënt nog leeft of niet, zie Afbeelding 6.



Afbeelding 6 Twee unaire rollen. Unaire rollen worden gebruikt om eigenschappen te communiceren die alleen wel of niet van toepassing kunnen zijn.

8.4 Attribuut of Object

Uitspraak 25 communiceert het (elementaire) feit dat een patiënt allergisch is voor bepaalde medicijnen. Nu had de modelleur er ook voor kunnen kiezen om Medicijn een attribuut te maken in plaats van het object Medicijn(.naam) te gebruiken, zoals in Afbeelding 5. Het verschil is dat indien een object gebruikt wordt deze een identificatie moet hebben, namelijk de naam van het medicijn in dit geval. Het resultaat is dat indien meerdere patiënten allergisch zijn voor hetzelfde medicijn de naam van het medicijn maar één keer wordt opgeslagen. Als Medicijn een attribuut was wordt de naam van het medicijn elke keer opgeslagen wanneer een patiënt daar allergisch voor is. Daarnaast worden op deze manier alleen de medicijnen geregistreerd waar patiënten allergisch voor zijn. Wanneer alle bestaande medicijnen geregistreerd moeten worden is dit geen goede werkwijze. In het geval dat Medicijn een object is zijn medicijn en patiënt niet van elkaars bestaan afhankelijk.

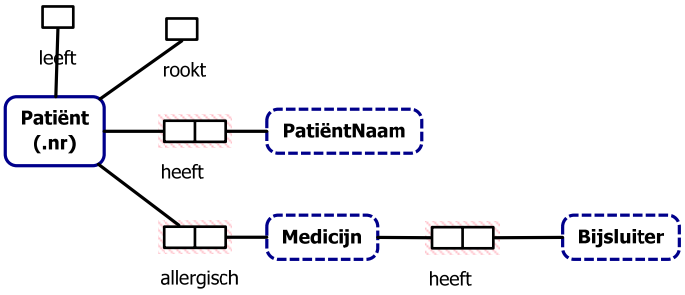


Afbeelding 7 Het ORM diagram dat uitspraken 22 t/m 26 gevisualiseerd.

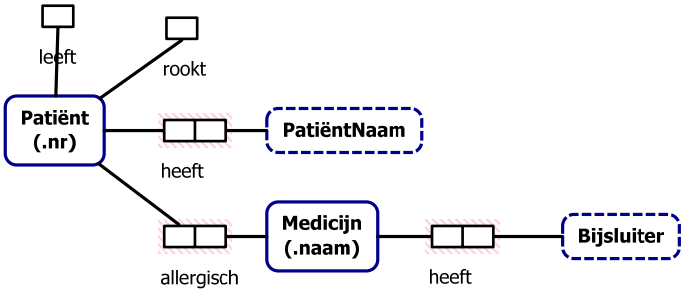
Zowel de keuze voor een attribuut als een object was in dit geval goed. In de situatie zoals in Afbeelding 7 kunnen medicijnen opgeslagen worden zonder dat er een patiënt er allergisch voor is. De modelleur heeft meestal de keuze tussen een attribuut of een object, maar niet altijd. Als voorbeeld breiden we het systeem uit met het volgende statement:

26. Medicijn heeft bijsluiter.

Dit statement lijkt op een unaire rol namelijk er is wel of niet een bijsluiter bij een medicijn. Maar wanneer er wel een bijsluiter is dan willen we de tekst ervan opnemen in het systeem. Dus dit feit wordt gemodelleerd als een attribuut dat we een binaire rol (twee kleine rechthoekjes) toebedeeld krijgt met Medicijn. Het resultaat hiervan is te zien in Afbeelding 8. Echter nu is er iets raars aan de hand namelijk dat bijsluiter een attribuut is dat iets zegt over het attribuut medicijn. Dit is eigenlijk niet mogelijk. Het gevolg is dat Medicijn moet veranderen in een object. Meestal bepaalt dus de context of een attribuut of object gebruikt moet worden. Dit wordt in de literatuur schematische stabiliteit genoemd. Het correcte resultaat is te zien in Afbeelding 9.



Afbeelding 8 De Bijsluiter is toegevoegd aan het Schema.



Afbeelding 9 Bijsluiter is een attribuut van het object Medicijn. DIt is correct.

8.5 Uitbreiding van ORM schema.

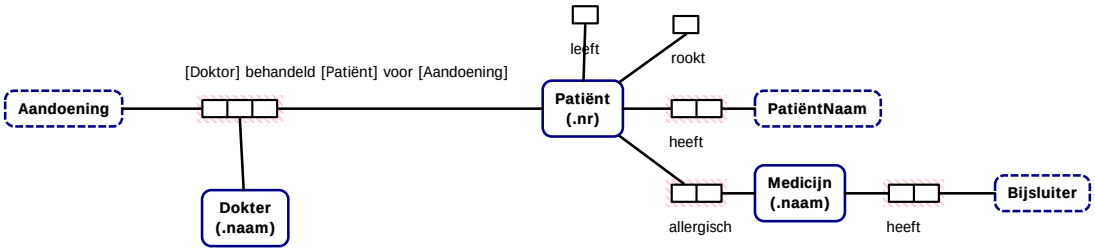
De volgende twee uitspraken komen voor in het functioneel ontwerp van het systeem.

27. Patiënt 1025 wordt behandeld door dokter De Wit voor longkanker.
28. Patiënt 1025 wordt behandeld door dokter De Groot voor niersteen.

Deze twee uitspraken hebben de volgende structuur:

29. Patiënt <patiëntnr> wordt behandeld door Dokter <DokterNaam> voor <aandoening>.

Het resultaat hiervan is weergegeven in Afbeelding 10. Hierbij wordt er van uitgegaan dat een dokter uniek valt te identificeren door middel van zijn naam. Met andere woorden: binnen de applicatie is het niet mogelijk dat twee Dokteren dezelfde naam hebben.



Afbeelding 10 ORM diagram is uitgebreid met de structuur van uitspraak 29.

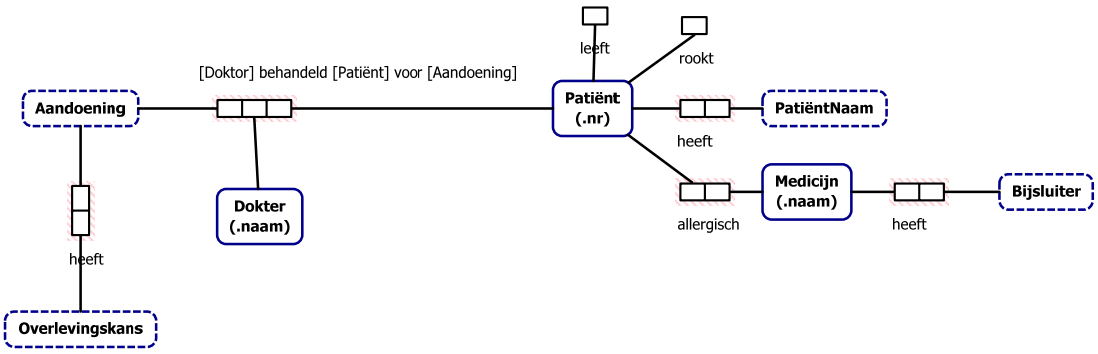
Ook de volgende uitspraak moet vastgelegd worden in het model:

30. Voor Longkanker is overleevingskans 70%.

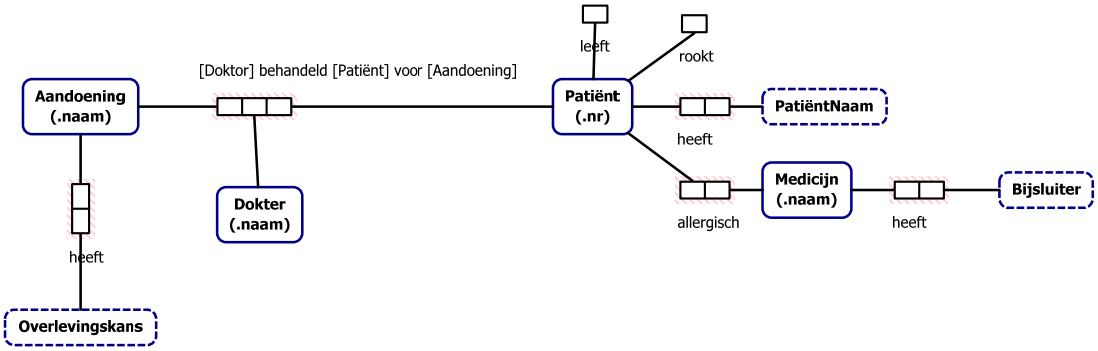
De structuur is dan

31. <Aandoening> overlevingskans <kans percentage>

Dit kan weergeven worden in een ORM diagram zoals in Afbeelding 11. Echter het aanbrengen van een rol tussen attributen is niet correct. Als “regel” kunnen we zeggen dat één van de twee attributen veranderd in een Object. In dit geval veranderen we Aandoening in een object dat geïdentificeerd wordt door de naam, zie Afbeelding 12. De logica of iets een object of attribuut is hangt af van de context en moet bepaald worden door de modelleur. Maar het is niet gewenst dat een attribuut een relatie (rol) aangaat met een attribuut. Een dergelijke situatie kwam ook ter sprake bij het aanbrengen van uitspraken die betrekking hadden op Medicijnen en bijsluiter.



Afbeelding 11 Er is een rol aangebracht tussen aandoening en overlevingskans. Echter een relatie aanbrengen tussen twee attributen is niet gebruikelijk.



Afbeelding 12 Het is noodzakelijk om Aandoening als object te modelleren.

8.6 Objectificatie

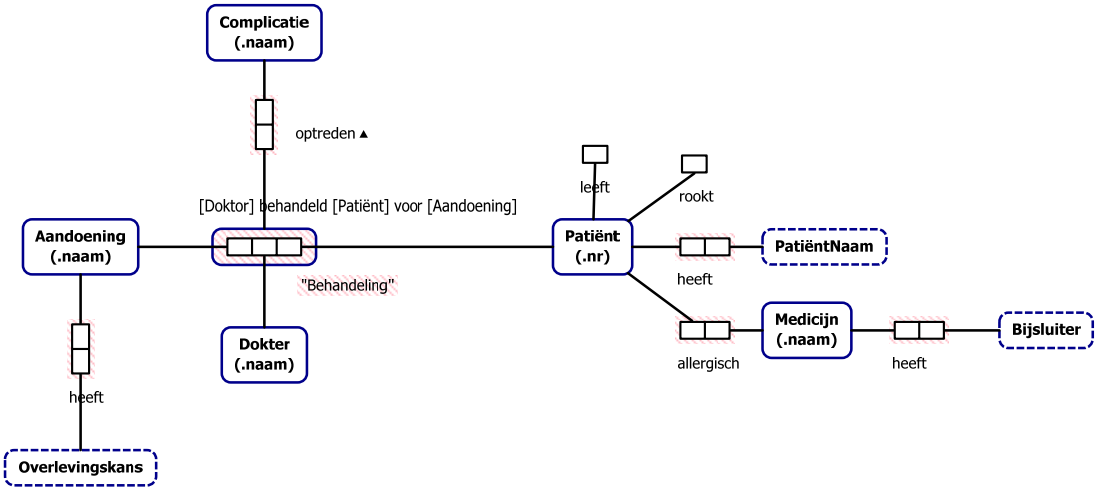
Als een patiënt behandeld wordt voor een bepaalde aandoening wil de arts eventuele complicaties, bijvoorbeeld een ontsteking kunnen registreren.

32. Patiënt 1025 wordt behandeld door Dokter De Wit voor longkanker en bij de behandeling treedt een ontsteking op.

De structuur van deze uitspraak is

33. **Patiënt <patiëntnr> wordt behandeld door Dokter <DokterNaam> voor <Aandoening>** hierbij treed <Complicatie> op.

Het eerste deel van de structuur (vet gedrukt) bevind zich al in het ORM diagram (Afbeelding 13) als een rol tussen de objecten Aandoening, Dokter en Patiënt. Het nieuwe deel van deze uitspraak (niet vet) verteld dus eigenlijk wat over de relaties (rol) tussen deze objecten. Indien dit het geval is kan een relatie geobjectificeerd worden. Dit wordt gedaan door om de rol een Object symbool te tekenen. De geobjectificeerde rol heeft de naam “Behandeling” gekregen. Vervolgens wordt daaraan het object complicatie gerelateerd.



Afbeelding 13 behandeling is een Geobjectificeerde rol.

8.7 Uniciteitbeperking

De begrippen uniciteitbeperking en mandatory rol constraints worden uitgelegd aan de hand van een voorbeeld waarin wordt bijgehouden welke werknemer(s) welke auto's gebruiken. Laten we eens kijken naar de relatie tussen auto en persoon. We beschouwen dit voorbeeld in het UoD van een bedrijf dat in de personeelsadministratie bijhoudt welke personeelsleden welke auto's gebruiken. Een statement dat deze situatie beschrijft is:

34. Persoon met Personeelsnummer 123 rijdt een auto met kenteken x24wa

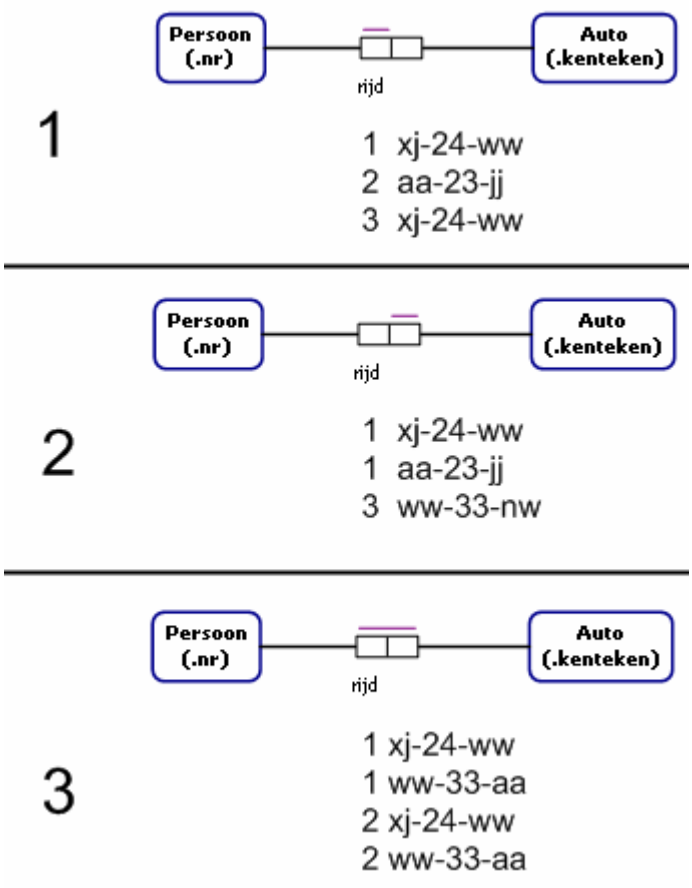
Dit feit heeft de volgende structuur:

35. Persoon <persnr> heeft Auto <kenteken>.

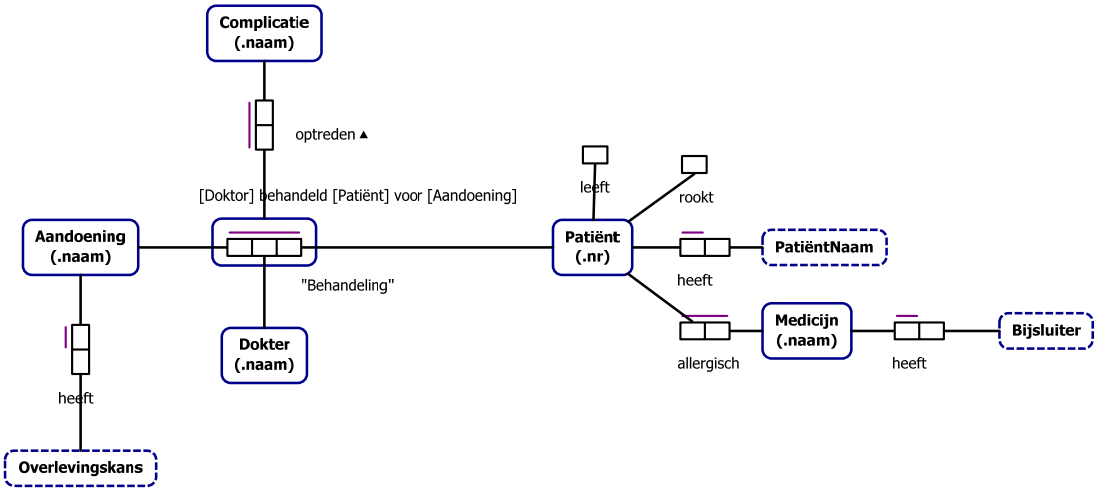
Er is nu nog maar weinig bekend over de relatie tussen Auto en Persoon en er kunnen vragen ontstaan zoals:

- Mag een auto door meerdere personen gebruikt worden?
- Mogen personen meerderen auto's gebruiken?

Deze vragen suggereren bepaalde beperkingen, dergelijke beperkingen kunnen vastgelegd worden met een zogenaamde uniciteitbeperking. Deze worden gevisualiseerd in een ORM schema met behulp van een streep die bepaalt dat de waardes daaronder uniek moeten zijn. Door goed te kijken naar Afbeelding 14 en de voorbeeldpopulaties bij de verschillende gevallen moet duidelijke maken hoe uniciteitbeperkingen invloed hebben op de mogelijke waarden en dus op het type relatie dat gerealiseerd wordt. Hierin zijn verschillende soorten relaties te herkennen, namelijk 1-op-n (1) en n-op-1 (2) en n-op-n (3). Kort samengevat; de waardes onder de uniciteitbeperkingen, dat zijn de strepen boven de rol, moeten uniek zijn. Het aantal objecten/attributen dat een rol speelt in de relatie (feit) wordt aangeduid met de cardinaliteit. De cardinaliteit van *"<Patiëntnr> heeft <PatiëntNaam>"* is twee en van *"<Dokter> behandeld <Patiënt> voor <Aandoening>"* is drie.
Uniciteitbeperking moeten minimaal de cardinaliteit verminderd met één rollen bedekken. Dus het feit *<Patiëntnr> heeft <PatiëntNaam>* heeft een uniciteitbeperking die minimaal één rol overspant.
Om de uniciteitbeperkingen te kunnen bepalen is het raadzaam een representatieve populatie te gebruiken. Door deze te analyseren, onder welke rol(len) mogen geen dubbele waarden voorkomen, kan bepaald worden op welke rol(len) de unicitietsbeperkingen van toepassing zijn. Indien een representatieve populatie niet aanwezig is doet men er verstandig aan deze samen met de domeinexpert te bedenken.



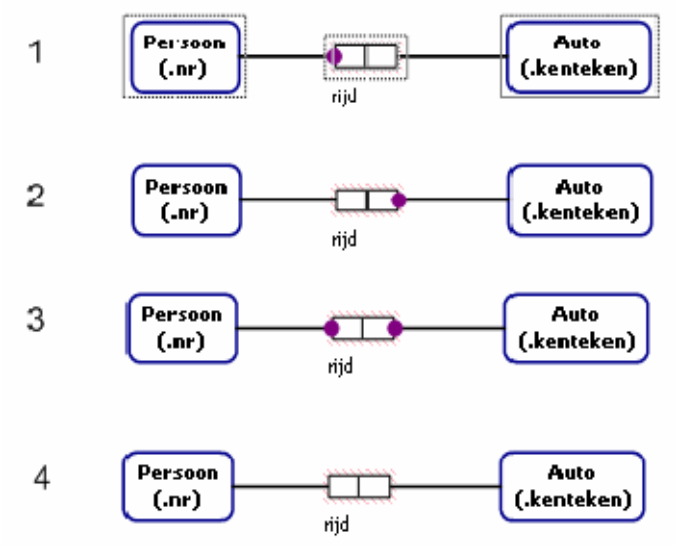
Afbeelding 14 Verschillende uniciteitbeperking. (1) 1 Persoon kan 1 Auto rijden. 1 Auto kan door meerdere Personen worden gedeeld. (2) 1 Persoon kan meerdere Auto gebruiken. 1 Auto wordt maximaal gebruikt door 1 Persoon. (3) 1 Persoon mag meerdere Auto's gebruiken . 1 Auto kan door meerdere personen worden gereden.



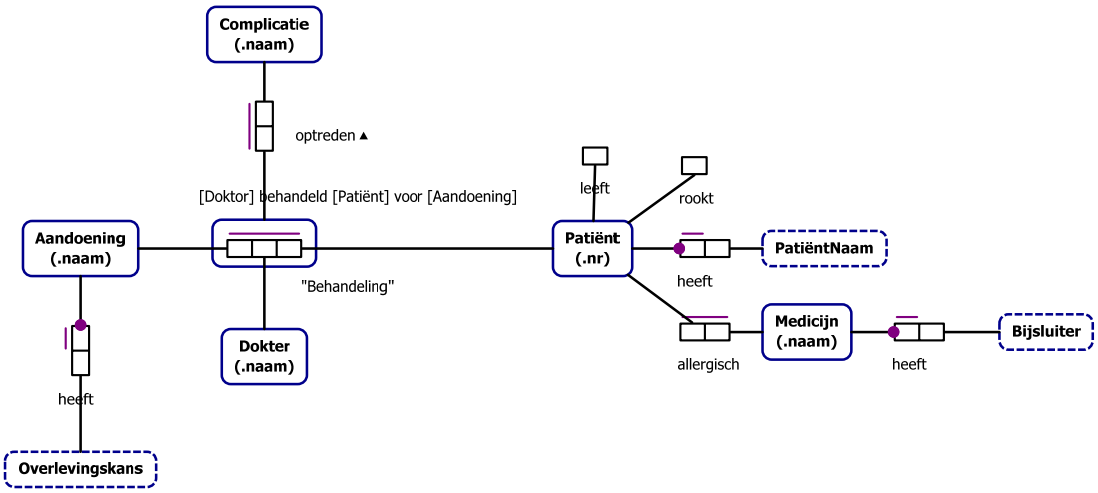
Afbeelding 15 Er zijn nu verschillende uniciteitbeperkingen toegevoegd.

8.8 Mandatory rol constraints

Als we kijken naar het voorbeeld met de auto's (Afbeelding 14) kunnen we niks zeggen over of een relatie tussen één auto en één persoon moet voorkomen. Concreet betekent dit dat indien er een instantie van een auto is er niks in het conceptuele model is vastgelegd over het feit of deze moet toegekend zijn aan één of meerdere personen en indien er een Persoon is dat die één, geen of meerdere auto's kan gebruiken.
Met behulp van mandatory rol constraints wordt het mogelijk in een ORM diagram aan te geven of een rol optioneel of verplicht vervult moet worden. Met een stip wordt aangegeven of een rol optioneel is of niet. Om dit te begrijpen wordt dit geïllustreerd aan de hand van een voorbeeld Afbeelding 16.



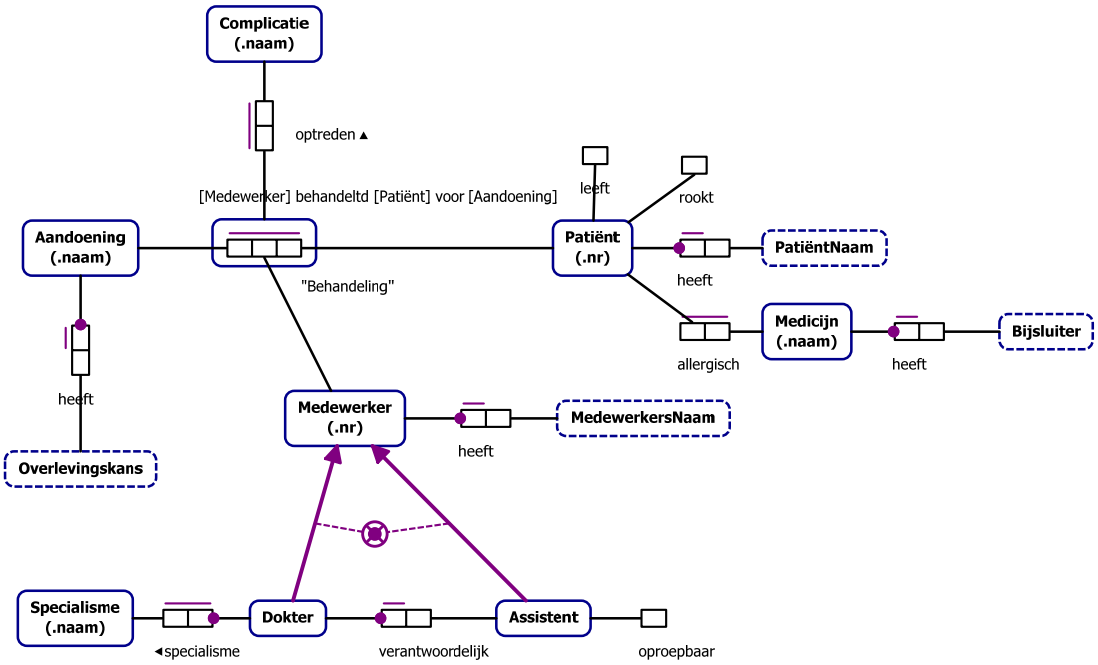
Afbeelding 16 Een Persoon rijdt altijd één of meerdere auto's en Auto kan gebruikt worden door Persoon (2) Iedere Auto wordt gebruikt door minimaal één Persoon en een Persoon kan een Auto rijden (3) Ieder Persoon rijdt minimaal één Auto en iedere Auto wordt gebruikt door minimaal één Persoon. (4) Persoon kan een auto rijden en een Auto kan bestuurd worden door een Persoon.



Afbeelding 17 Verschillende mandatory role constraints toegevoegd (paarse stippen).

8.9 Generalisatie en specialisatie relatie

Van artsen willen we echter gaan bijhouden welke specialisme ze hebben. Een arts moet minimaal één specialisme hebben om in het ziekenhuis te mogen werken. Er zijn ook assistenten werkzaam voor artsen. Iedere assistent heeft precies één arts die verantwoordelijk is voor het medische handelen van de assistent. Sommige assistenten mogen in noodgevallen gebeld worden in de nacht om simpele ingrepen te verrichten. Mederwerkers, zowel artsen als assistenten, worden vanaf nu uniek geïdentificeerd door middel van het medewerkersnummer. Van iedere medewerker wordt de naam bijgehouden. Het resultaat van deze aanpassingen is te zien in Afbeelding 18. Er is gebruik gemaakt van een generalisatie/specialisatie relatie. Een dergelijke relatie wordt gevisualiseerd door gebruik te maken van een dikke pijl. Een arts is een Medewerker en een Assistent is ook een Medewerker. De gemeenschappelijk rollen die Arts en Medewerker aangaan worden gekoppeld aan Medewerker (de generalisatie). De rollen die specifiek zijn voor een Arts en Assistent worden gekoppeld aan respectievelijk Arts en Medewerker (de specalisaties). Een Medewerker is dus een Arts of een Assistent maar niet beide tegelijkertijd, dit wordt gevisualiseerd d.m.v. exclusion beperking wat zich tussen de twee pijlen bevind. Het is belangrijk te onthouden dat indien meerdere objecten dezelfde rollen aangaan dat er sprake kan zijn van een generalisatie/specialisatie relatie. Om er achter te komen of een dergelijke relatie geoorloofd is dient onderzocht te worden of er sprake is van een “is-een” relatie. Bijvoorbeeld een dokter is een medewerker en assistent is een medewerker. Echter patiënt is geen medewerker en daarom is er ook geen generalisatie/relatie gevisualiseerd tussen patiënt en medewerker.



Afbeelding 18 De generalisatie/specialisatie zijn toegevoegd.

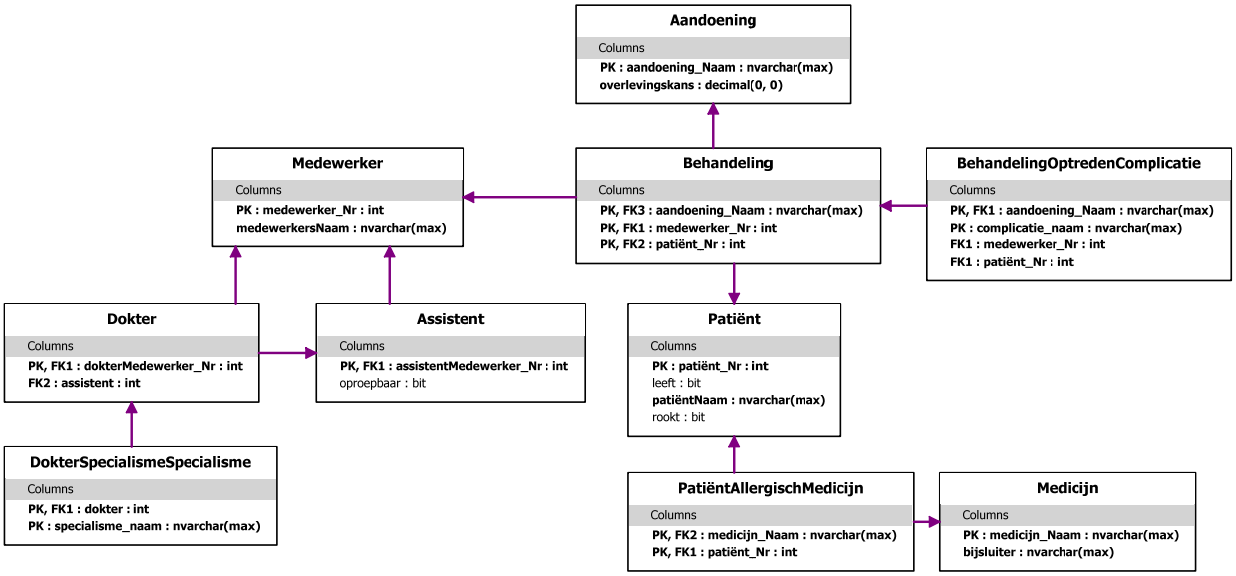
8.10 Valideren van ORM model

Het ORM model is makkelijk te verifiëren op correctheid. Met de NORMA tool kunnen zeer eenvoudig verwoordingen gegenereerd worden. Laten we eens een stukje van een dergelijke verwoording bekijken:

Dokter verantwoordelijk Assistent.
Each Dokter verantwoordelijk **exactly one** Assistent.
It is possible that more than one Dokter verantwoordelijk **the same** Assistent.

8.11 RMAP

Nu het ORM diagram volledig is kan het worden getransformeerd in een equivalent relationeel model. Dit gebeurt algoritmisch en wordt gedaan met behulp van de NORMA tool. De modelleur kan door middel van schema optimalisatie wel enige invloed uitoefenen op de transformatie van ORM naar het relationele model. Schema optimalisatie valt buiten het bereik van deze inleiding. In Afbeelding 19 is het equivalente relationele model te zien.



Afbeelding 19 Het relationele model.

8.12 Conclusie

Door het proces (CDSP) te doorlopen kan de modelleur een conceptueel ORM model maken van het UoD. Dit model wordt omgezet in een relationeel database schema waarin geen redundantie voorkomt. Zowel het feit dat er geen redundantie voorkomt en dat met het ORM model de functionele specificatie verwoord kan worden maakt de ORM methode geschikter voor het ontwerpen van database schema's dan andere technieken zoals ER en UML. Hopelijk is de lezer nu in staat om simpele ORM modellen te maken en de betekenis van ORM diagrammen te begrijpen.

9 Schema integratie

Om gegevens uit verschillende database systemen te kunnen combineren is het wenselijk om schema-integratie toe te passen. Dit betekent dat twee of meer lokale databaseschema's worden samengevoegd in één globaal databaseschema. In dit document wordt een casestudie beschreven die twee lokale databaseschema's integreert tot één globaal databaseschema. De methode die hiervoor gebruikt is staat beschreven in [17].

9.1 ORM

Deze methode is gebaseerd op ORM (Object Role Modelling). Met ORM is het mogelijk om op een conceptueel niveau te modelleren. Uit een ORM model kan vervolgens een databaseschema gegenereerd worden zonder redundantie (5de normaalvorm). Het voordeel ten opzichte van andere modeleermethoden (zoals ER diagrammen en UML) is dat er in ORM gemodelleerd wordt met concepten (objecten) en relaties (rollen) tussen concepten. Dit betekent dat het conceptuele model beschreven wordt in de natuurlijke taal en dat het functionele ontwerp onderdeel wordt van het ORM schema. Daarnaast biedt ORM een methode om systematisch een datamodel te maken van de Universe of Discourse (het probleem domein), namelijk de Conceptuel Schema Design Procedure (CSDP).

In [19] wordt ORM gedetailleerd beschreven. Dit document maakt gebruik van de ORM2 notatie [20]. Het programma dat gebruikt is om de ORM diagrammen mee te maken is de NORMA tool [22].

9.2 Schema Integratie

Bij dataintegratie is sprake van een verzameling lokale schema's en één globaal schema. Er zijn drie verschillende methoden [18] om data integratie te verwezenlijken, namelijk:

1. Global as View (GAV): het globale schema wordt gedefinieerd als een verzameling van views op de lokale schema's.

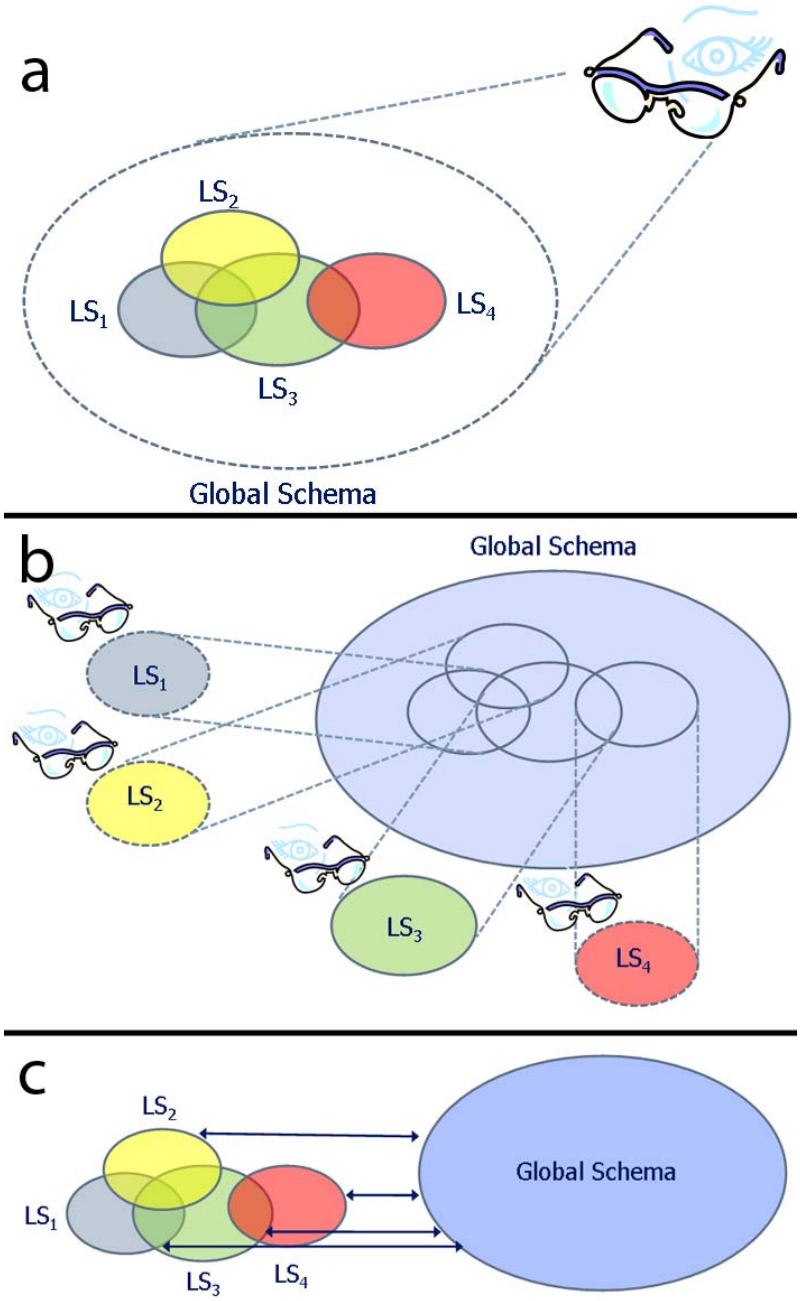
2. Local as View (LAV): de locale schema's worden gedefinieerd als views op het globale schema.

3. Replicatie technieken: data wordt tussen het globale systeem en de lokale systemen gesynchroniseerd en/of gerepliceerd.

In Figuur 16 worden drie verschillende data-integratie methoden afgebeeld. In dit document wordt data-integratie gerealiseerd door de GAV methode te gebruiken. Deze methode in combinatie met ORM wordt beschreven in [17, 18]. Voor de integratie worden twee lokale databaseschema's gebruikt:

1. Een subset van het Kidos Schema

2. School resultaten registratie systeem



Figuur 16 Verschillende data-integratie methoden.

(a) Global-As-View (GAV): het globale schema is gedefinieerd als een view op de lokale schema's

(b) Local-As-View (GAV): lokale schema's worden gedefinieerd als view op het globale schema

(c) Data Exchange: de data wordt gesynchroniseerd/gerepliceerd tussen het globale en lokale schema's

9.3 Plan van aanpak

Om het samenvoegen van de twee schema's te realiseren zijn de volgende stappen doorlopen:

- Het ontwerpen van een database schema (met behulp van ORM) waarin schoolresultaten van leerlingen worden opgeslagen.
- Reverse engineering gebruiken om een subset van het Kidos databaseschema te converteren naar een ORM model.
- Twee ORM schema's samenvoegen tot één globaal schema aan hand de van de methode zoals beschreven in [17].

9.4 Schoolresultaten

Het doel van het schoolresultaten registratie systeem is resultaten van leerlingen van middelbare scholen te registreren. Vervolgens kan het systeem gebruikt worden om prestaties van leerlingen op te vragen. Het conceptuele ontwerp (ORM diagram) van dit systeem staat in Figuur 17.

In het schoolresultaten (SR) systeem staan de resultaten van leerlingen centraal.

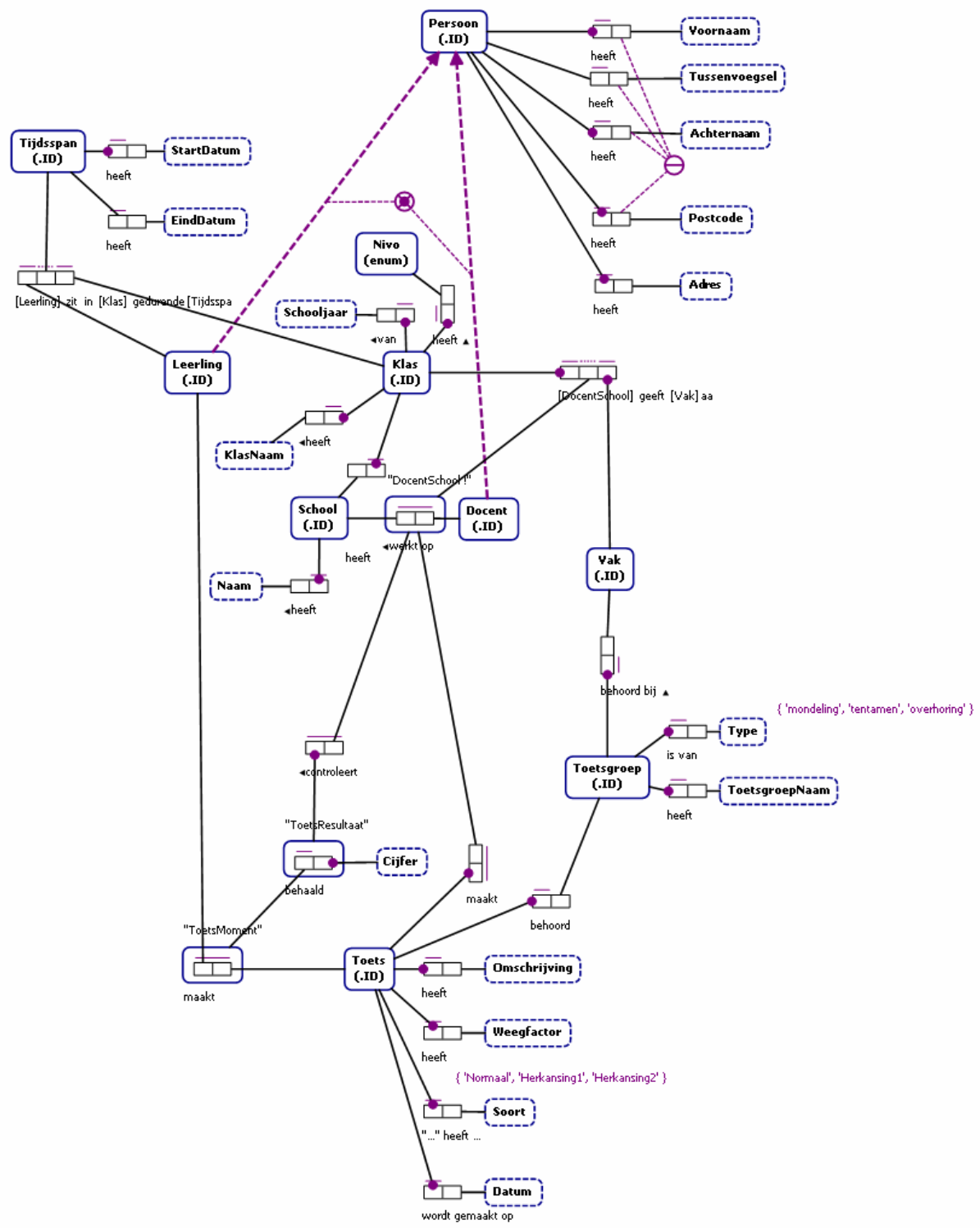
Er zijn twee types personen aanwezig in het systeem (*Leerlingen* en *Docenten*). *Leerlingen* en *Docenten* hebben een aantal gemeenschappelijke attributen; daarom is er voor gekozen om de generalisatie relatie *Persoon* op te nemen in het ontwerp.

Een *leerling* maakt tijdens zijn schoolcarrière verschillende *Toets*en (*ToetsMoment*). Een *ToetsGroep* representeert een verzameling van toetsen die bij elkaar horen. Bijvoorbeeld een *Toets* Wiskunde1 heeft ook herkansing één en twee. Door toetsen te groeperen in *Toetsgroepen* is het uiteindelijke resultaat voor een bepaalde toets te berekenen: voor iedere Toets wordt een *weegfactor* en *omschrijving* vastgelegd waardoor het eindcijfer voor een bepaald vak te berekenen valt.

Een *ToetsMoment* legt een relatie tussen een *Leerling* en een *Toets*. Vervolgens kan aan een dergelijk Toetsmoment (door gebruik te maken van het Objectify fact type) een resultaat (cijfer) worden toegekend. Om zowel vast te leggen wat het resultaat is en welke *Docent*(en) de *Toets* hebben nagekeken wordt gebruik gemaakt van een Objectify fact type, namelijk *Toetsresultaat*.

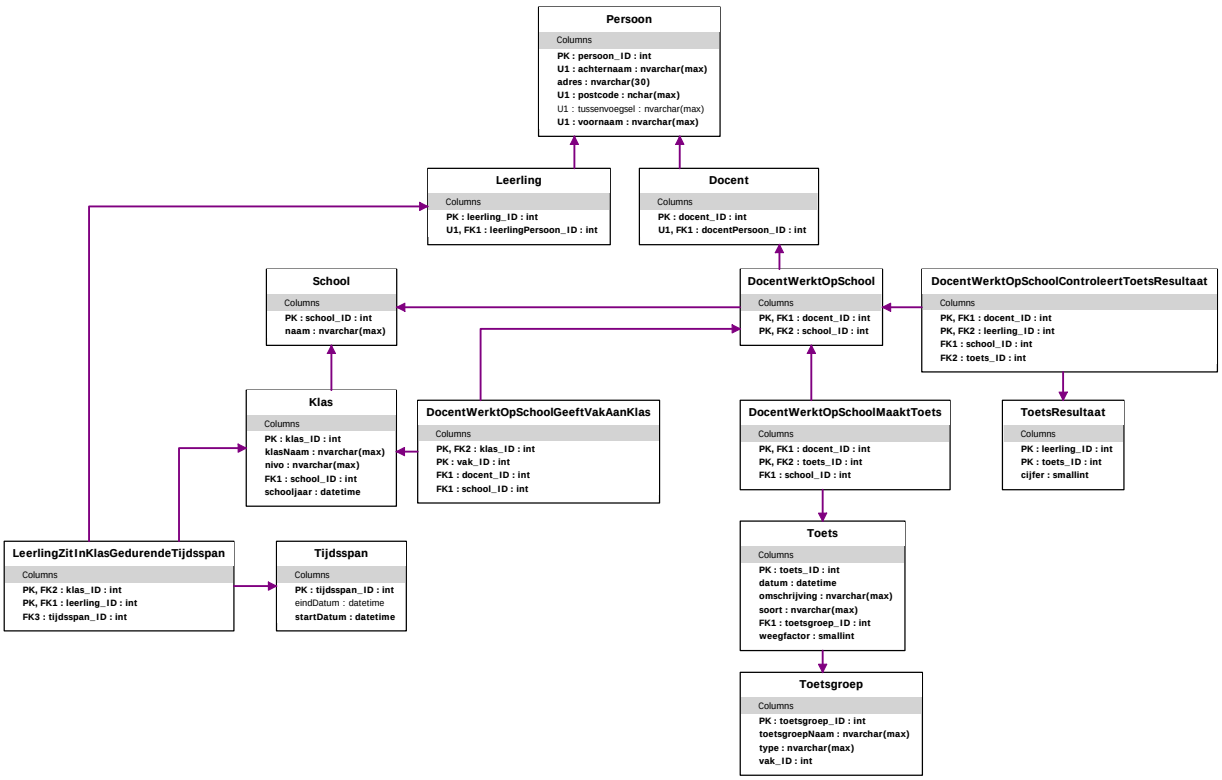
Van een *Leerling* wordt bijgehouden in welke *Klas* hij/zij zit en gedurende welke periode (*Tijdsspanne*). Een *Klas* is altijd gerelateerd aan een *School*.

Van de combinatie *Docent*en, *Scholen* wordt geregistreerd welke *Vakken* ze aan welke *Klas*sen geven. Ook wordt geregistreerd welke *Toetsen* een *Docent* controleert en/of opstelt.



Figuur 17 Conceptueel model van School Resultaten schema.

Het ORM ontwerp (Figuur 17) van het systeem kan algoritmisch geconverteerd (door NORMA) worden naar een relationeel model in de 5de normaal vorm, zie Figuur 18.



Figuur 18 Database schema van het Schoolregistratie systeem.

9.5 Reverse Engineering

Reverse Engineering wordt onder andere toegepast om de volgende reden:

- Incorrect model van de Universe of Discourse. Het database model komt niet overeen met de werkelijkheid en dit vormt een probleem.
- Inefficiënt database schema. Indien andere optimalisatietechnieken niet voldoende werken om de beoogde performanceverbetering te realiseren kan herontwerp van het database schema een oplossing bieden.
- Onduidelijkheid. Het is niet inzichtelijk of begrijpbaar hoe het systeem werkt. Van legacy systemen is meestal geen documentatie of kennis meer beschikbaar. Om deze systemen toch te kunnen begrijpen en uit te faseren is het afleiden van het datamodel noodzakelijk.

Meestal is het een combinatie van factoren die ertoe leiden dat er wordt overgegaan op roundtrip engineering. Bij roundtrip engineering wordt op het huidige systeem reverse engineering toegepast. Hierdoor wordt een model verkregen van het systeem, bijvoorbeeld een ORM model. Daarna worden er verbeteringen aangebracht in het model om vervolgens de verbeteringen toe te passen op het bestaande systeem.

In [19] wordt een het roundtrip engineering proces beschreven in vier stappen:

1. Conceptualisatie: in deze stap wordt de Conceptual Schema Design Procedure (CSDP) toegepast om een conceptueel model te maken van het bestaande database schema. De bestaande database kan gezien worden als een verzameling van tabellen die als inputformulieren gebruikt worden voor de CSDP.

2. Transform: het initiële conceptuele schema dat verkregen is in stap 1 wordt conceptueel geoptimaliseerd. Dit betekent dat een conceptueel model wordt geconverteerd naar een equivalent conceptueel model. Met equivalent wordt hier bedoelt dat het conceptuele model dezelfde betekenis (semantiek) maar een andere structuur heeft. Dergelijke transformaties zijn noodzakelijk voor:

    a. Het verduidelijken (leesbaarder maken) van het conceptuele schema.

    b. Het optimaliseren van het ORM model zodat het efficiënter mapt naar een relationeelschema door middel van de RMap procedure.

    c. Om performanceproblemen te verhelpen kan gebruik worden gemaakt van gecontroleerde denormalisatie. Dit betekent dat op een gecontroleerde manier redundantie wordt geïntroduceerd in het systeem. Alleen indien performance problemen echt niet anders verholpen kunnen worden en bij bijzondere toepassingen zoals datawarehouses is denormalisatie een optie.

3. RMap: het verbeterde conceptuele ORM schema wordt geconverteerd met behulp van de RMap naar een semantisch equivalent relationeel schema.

4. Data migratie & query transformatie: uiteindelijk moet de data uit het originele systeem geconverteerd en/of aangevuld worden voor de nieuwe situatie. Als het relationele schema verandert zullen ook queries en eventuele business logica aangepast moeten worden aan de nieuwe situatie.

Om de reverse engineering-stap toe te passen op het schema van Kidos is een algoritmische aanpak gebruikt. Dit proces is sterk gebaseerd op de CSDP. In de volgende paragraaf wordt de inverse RMap ($RMap^{-1}$) beschreven waarop het stappenplan hieronder gebaseerd is.

1. De database tabellen worden afgebeeld als objecten in het ORM model.

2. Vervolgens worden de referentiële sleutels één op één afgebeeld als een rol tussen twee objecten in het ORM model. Indien een referentiële sleutel gebruikt wordt met daarin het woord “super” is er vanuit te gaan dat een tabel waarnaar verwezen wordt een generalisatie relatie is van de tabel waarin een dergelijke referentiële sleutel voorkomt. Indien dit het geval is wordt een generalisatie/specialisatierelatie gemodelleerd in het ORM model. Deze methode wordt verduidelijkt in de volgende twee paragrafen.

3. Het toevoegen van de mandatory-constraints wordt bepaald door te kijken of een referentiële sleutel altijd gevuld is.

4. Het toevoegen van de uniciteitbeperkingen wordt bepaald door te kijken of een referentiële sleutel unieke waarden moet bevatten.

5. De attributen van tabellen worden toegevoegd als rollen met value-types aan het desbetreffende object. Deze methode wordt verduidelijkt in de volgende twee paragrafen.

6. Om te controleren of het gemaakte ORM model semantisch equivalent is met het relationele model waarop het gebaseerd is wordt de volgende procedure gebruikt:

    a. Genereer het relationele database schema vanuit het ORM model.

    b. Controleer of het relationele schema dat vanuit het ORM model gegenereerd is semantisch equivalent is aan het oorspronkelijke relationele model. Controleer of beide schema's dezelfde tabellen hebben en of de referentiële sleutels ook hetzelfde zijn.

Door te analyseren wat er gebeurt bij de RMap procedure is de omgekeerde weg, van relationeel naar conceptueel ORM model, ook te verklaren. De Rmap procedure garandeert dat een redundantievrij relationeel schema wordt gegenereerd in de 5de normaalvorm. Dit is mogelijk omdat ieder (elementair) feit precies naar één tabel wordt geconverteerd zodat instanties van dergelijke feiten maar één keer worden opgeslagen.
De procedure die hiervoor beschreven is is wat kort door de bocht. Echter het principe waarop deze procedure werkt is het herkennen van patronen in het relationele schema en dan te converteren naar een ORM model.

In §9.6 en §9.7 wordt beschreven welke patronen er in een relationeel database schema te vinden zijn en hoe deze geconverteerd kunnen worden naar een semantisch equivalent ORM schema.

9.6 Rmap Inverse

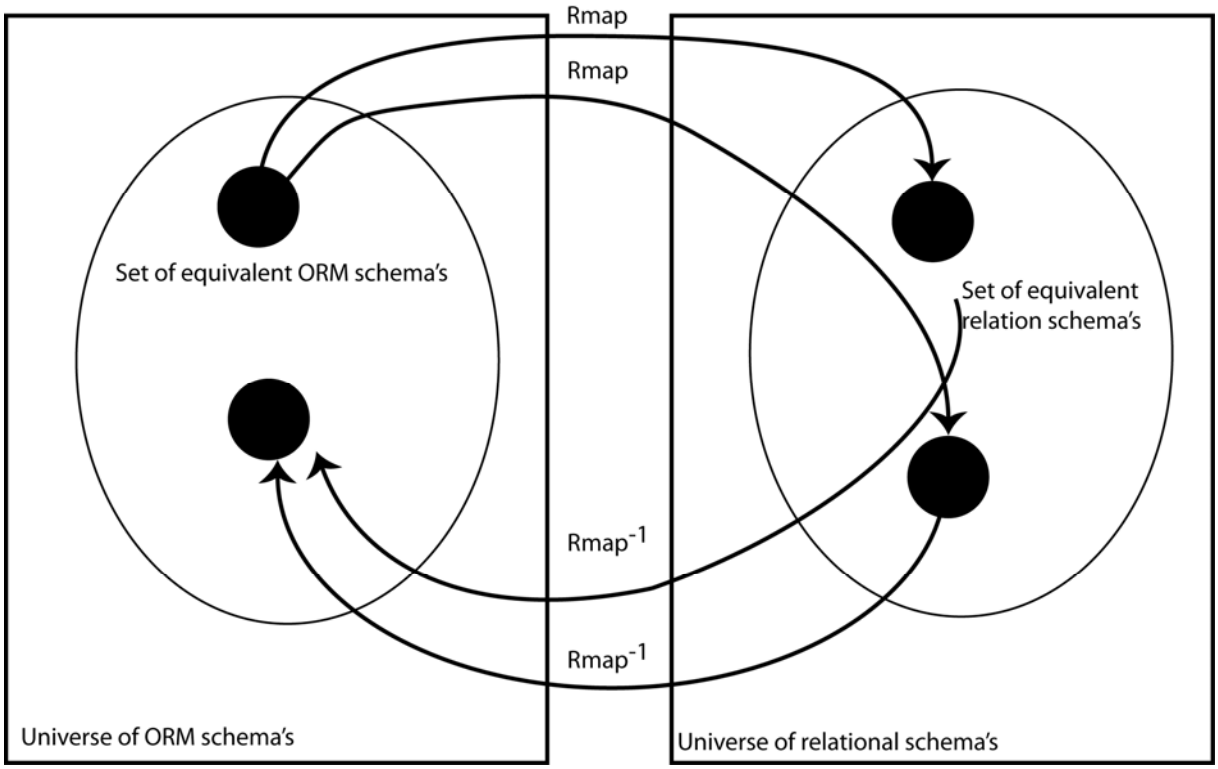
Als we kijken naar de RMap [19] procedure dan valt te constateren dat een ORM schema op één manier wordt geconverteerd naar een relationeel model. Dit betekent dat ieder ORM schema naar precies één equivalent relationeel schema wordt geconverteerd. Er moet worden opgemerkt dat de ontwerper bij generalisatie/specialisatie relaties de gewenste methode kan kiezen om een dergelijke relatie te transformeren naar het relationele model. Op dit detail na wordt dus ieder ORM schema op precies één relationeel model in 5de normaalvorm afgebeeld.

Echter het omgekeerde proces (RMap^{-1}) waarbij een relationeel model geconverteerd wordt naar een ORM model is niet eenduidig. Indien een relationeel model geconverteerd wordt naar een ORM model zijn er meerdere equivalente ORM modellen mogelijk die dezelfde semantische betekenis hebben als het originele model.

Een ORM schema wordt afgebeeld op precies één relationeel schema. Zowel ORM schema's als relationele schema's kunnen semantisch equivalente schema's hebben. Indien een relationeel systeem problemen heeft met de performance kan het noodzakelijk zijn om het bestaande schema te transformeren in een semantisch equivalent schema dat beter presteert. In

de praktijk komt dit er meestal op neer dat het aantal tabellen geminimaliseerd wordt en/of dat er gecontroleerde denormalisatie plaats vindt.

Omdat één ORM schema precies naar één semantisch equivalent relationeel model wordt geconverteerd loont het de moeite om het ORM schema te optimaliseren zodat het een efficiënt relationeel schema oplevert. In [19] wordt dit proces beschreven als schema optimalisatie. Hierbij worden transformaties voorgesteld in het ORM model zodat een nieuw semantisch equivalent model ontstaat. Dit nieuwe semantisch equivalente ORM model wordt geconverteerd naar een efficiënter relationeel schema. Het is dus de truc om een ORM model te maken dat semantisch equivalent is aan het oorspronkelijke ORM model dat de Universe of Discourse beschrijft maar dat wel een efficiënter relationeel schema oplevert.



Figuur 19 Het verband tussen RMap en $RMap^{-1}$. Indien de RMap uitgvoerd wordt op een ORM schema leverd dit een relationeel schema oplevert. Als vervolgens dit relationele schema wordt geconverteerd door de $Rmap^{-1}$ procedure naar een ORM schema wordt een ORM schema verkregen dat equivalent is met ORM schema waarmee begonnen is.

In Figuur 19 valt te zien hoe de RMap en de $RMap^{-1}$ zich gedragen.

In [19] wordt niet gesproken over $RMap^{-1}$. Om reverse engineering te realiseren wordt voorgesteld om Conceptuel Schema Design Procedure (CSDP) te volgen. Als input voor deze procedure worden de tabellen als formulieren gebruikt. De volgende paragraaf geeft een beschrijving van een alternatieve methode die gebruik maakt van patroonherkenning in het relationele database schema om een semantisch equivalent ORM model af te leiden.

9.7 Reverse engineering in ORM door middel van patroon herkenning

In dit hoofdstuk wordt beschreven hoe reverse engineering gerealiseerd kan worden door goed naar de structuur van het relationele schema te kijken. Hierdoor valt een semantisch equivalent ORM schema af te leiden. Deze procedure kan gezien worden als de inverse van de RMap ($RMap^{-1}$) procedure.
In een relationeel model (schema) kunnen we de volgende types relatiestructuren onderscheiden:

- 1:1
- 1:N
- M:N
- Generalisatie/specialisatie (meestal een speciaal geval van 1:1).

Deze structuren zijn duidelijk te herkennen in een relationeel schema (database). De truc is nu om deze verschillende relatietypes te converteren naar een ORM model.
De opsomming beschrijft de verschillende soorten relaties tussen tabellen uit het relationele schema en hoe deze afgebeeld worden op het ORM model:

- Een 1-op-n relatie (Figuur 20.1 en Figuur 20.2) is de enige relatie die op database niveau kan worden vastgelegd. De andere type relatie (1-op-1 en m-op-n) worden opgebouwd uit de 1-op-n relatie. Een 1-op-n relatie wordt geïmplementeerd in het relationeel database systeem door middel van een foreign key constraint (referentiële sleutel). In Figuur 20 zijn de verschillende soorten relaties weergegeven met in de rechter kolom een mogelijk semantisch equivalent ORM model.

- Een 1-op-1 relatie (Figuur 20.3) kan opgevat worden als de vereniging van de twee relaties, 1-op-n (Figuur 20.1) en n-op-1 (Figuur 20.2).

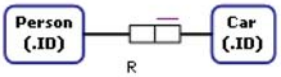
- Een m-op-n relatie (Figuur 20.4) kan worden opgevat als de samenvoeging van de twee relaties 1-op-n (Figuur 20.1) en n-op-1 (Figuur 20.2) door een tussentabel (associatietabel) toe te voegen tussen de bestaande tabellen. Bij een m-op-n relatie zijn er eigenlijk twee mogelijkheden. De tussentabel bevat alleen de attributen die ook een onderdeel zijn van de referentiële sleutel, zoals in (Figuur 20.5). Dan kan deze afhankelijkheid afgebeeld worden op een binaire rol met een uniciteitsconstraint over de gehele rol (Figuur 20.5). Indien de tussentabel ook attributen heeft die niet onderdeel zijn van de referentiële sleutel is er sprake van een associatie relatie/klasse (Figuur 20.6). Deze relatie wordt afgebeeld in ORM als geöbjectificeerde rol.

- Een generalisatie/specialisatie relatie wordt niet in alle relationele database systemen ondersteund. Indien het wel ondersteund wordt door het DBMS is het natuurlijk duidelijk wanneer er sprake is van een dergelijke relatie. Echter als het niet ondersteund wordt en ontwerpers toch gebruik willen maken van dergelijke structuren wordt het gerealiseerd door middel van 1-op-n relaties. Een tabel die de specialisatie vervult heeft dan een referentiële sleutel naar de tabel die de generalisatie vervult, zoals afgebeeld in Figuur 20.7. Een dergelijk gebruik van een sleutel voor generalisatie doeleinde moet blijken uit de context. Veel ontwikkelaars gebruiken een

naamgevingsmethode om dergelijke relaties aan te duiden. Dit uit zich meestal in de naam van de referentiële sleutel zoals super, parent, base, extends, etc.

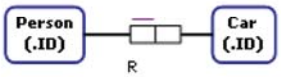
1 Person 1 N Car

Person(**personId**, ...)
personIdCar(**carId**, *personIc*



2 Person N 1 Car

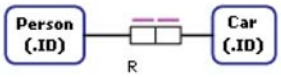
Person(**personId**, *carId*, ...)
Car(**carId**, ...)



---

3 Person 1 1 Car

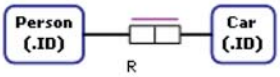
Person(**personId**, *carId*, ...)
Car(**carId**, ...)
Of
Person(**personId**, ...)
Car(**carId**, *personId*, ...)



---

4 Person M N Car

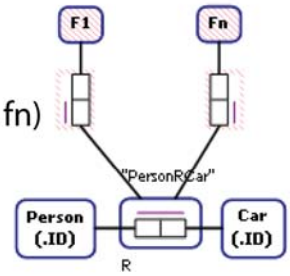
5 Person 1 N PersCar N 1 Car



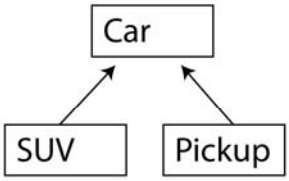
Person(**personId**, ...)
PersCar(***personId***, ***carId***)
Car(**carId**, ...)

6 Person 1 N PersCar N 1 Car

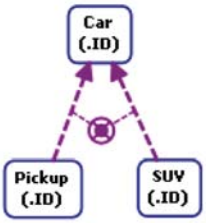
Person(**personId**, ...)
PersCar(***personId***, ***carId***, f1, ..., fn)
Car(**carId**, ...)



7



Car(**carId**, ...)
Suv(**suvId**, *carId*, ...)
Pickup(**pickupId**, *carId*, ...)



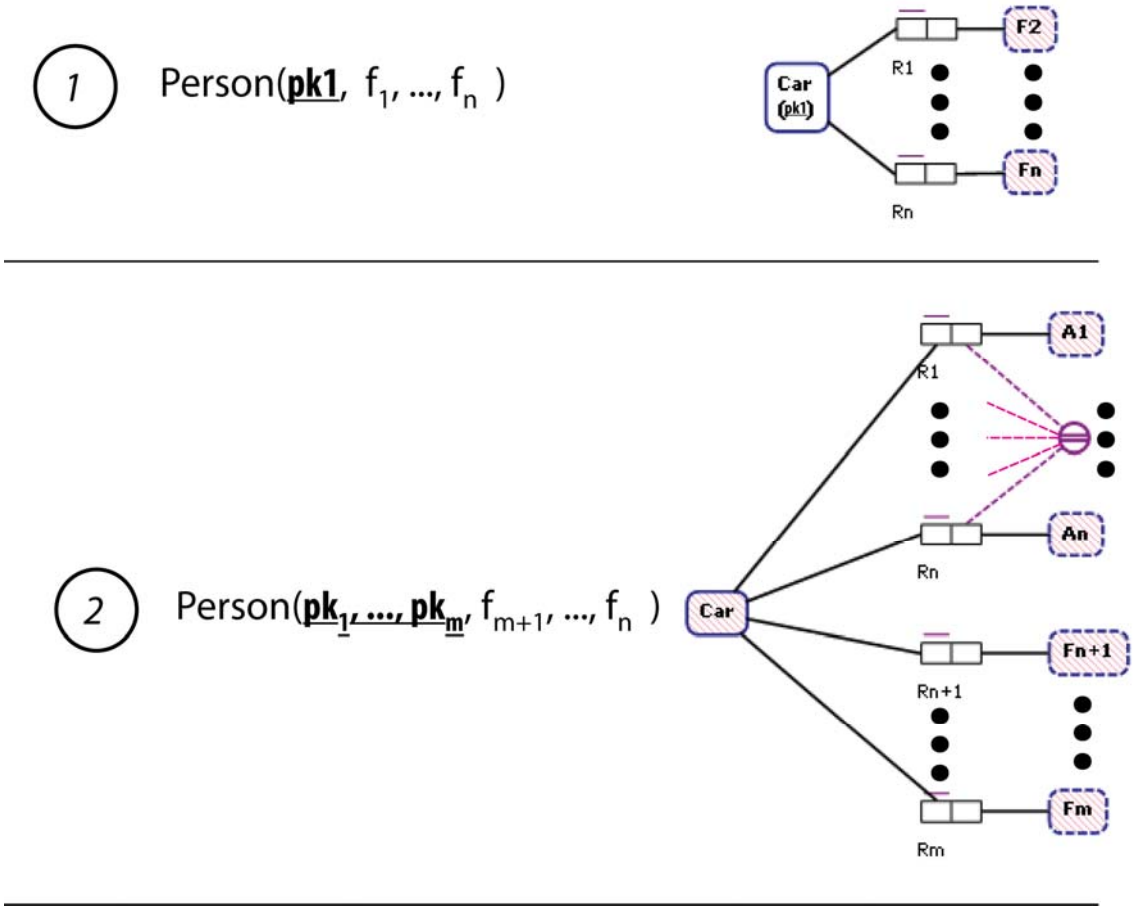
Legenda

Vet = Primary sleutel
Cursief = Referentiële sleutel
underline = Unieke sleutel

Figuur 20 De linker kolom geeft een visualisatie van het type relatie. In de middelste kolom is dit afgebeeld in het relationele model. In de rechter kolom is een equivalent ORM model te vinden. In het relationele model is de primaire sleutel onderlijnd en dik gedrukt. Een referentiële sleutel is cursief en dik gedrukt en onderlijnd.

Attributen in een relatie (tabel) vervullen een functionele rol. Bijvoorbeeld het attribuut salaris specificeert in de relatie Medewerker het loon van een medewerker. Dit kan opgevat worden als een functie, namelijk salaris(Medewerker) → bedrag, met als domein Medewerker en als bereik de salarisschaal die het bedrijf hanteert.

In de RMap procedure worden functionele rollen samengevoegd in één tabel. Daarom ligt het voor de hand om in de RMap^{-1} deze attributen te converteren naar functionele rollen. In Figuur 21.1 is een tabel afgebeeld die maar één attribuut heeft dat onderdeel is van de primaire sleutel. In Figuur 21.2 is een tabel afgebeeld dat een primaire sleutel heeft die uit meerdere attributen bestaat en het daarbij behorende ORM model.



Figuur 21 Rmap-1 voor functionele rollen.

De afbeeldingen Figuur 20 en Figuur 21 visualiseren de patronen die gebruikt worden voor de RMap^{-1} procedure. Het idee is dus dat patronen in het relationele model herkend worden en worden omgezet naar het ORM model. Op deze manier is het dus mogelijk een equivalent ORM model te maken van een bestaand relationeel database schema.

Het is niet eenvoudig om te bewijzen dat een RMap procedure gevolgd door een RMap^{-1} procedure een schema oplevert dat equivalent is aan het schema dat als input diende voor de RMap procedure. De voorgestelde RMap^{-1} procedure heeft zulke triviale transformaties dat

het zeer aannemelijk is dat dit het geval is. In §9.8.4 is met behulp van de $RMap^{-1}$ en de RMap procedures gecontroleerd of deze aanname klopt met behulp van een voorbeeld. Dit proces levert inderdaad hetzelfde schema op als waarmee begonnen was.

De NORMA-tool maakt het mogelijk om automatische relationele database schema's te converteren naar een equivalent ORM model. Momenteel worden MySQL en Microsoft SQL Server ondersteund in NORMA om reverse engineering toe te passen. Reverse engineering van een relationeel database schema naar een ORM model maakt het mogelijk om het bestaande schema te evalueren op kwaliteit en bruikbaarheid. Na het afleiden van een ORM model uit een bestaand database schema kan een dergelijk model gebruikt worden voor data integratie met andere ORM modellen.

9.8 Reverse Engineering van het Kidos Schema

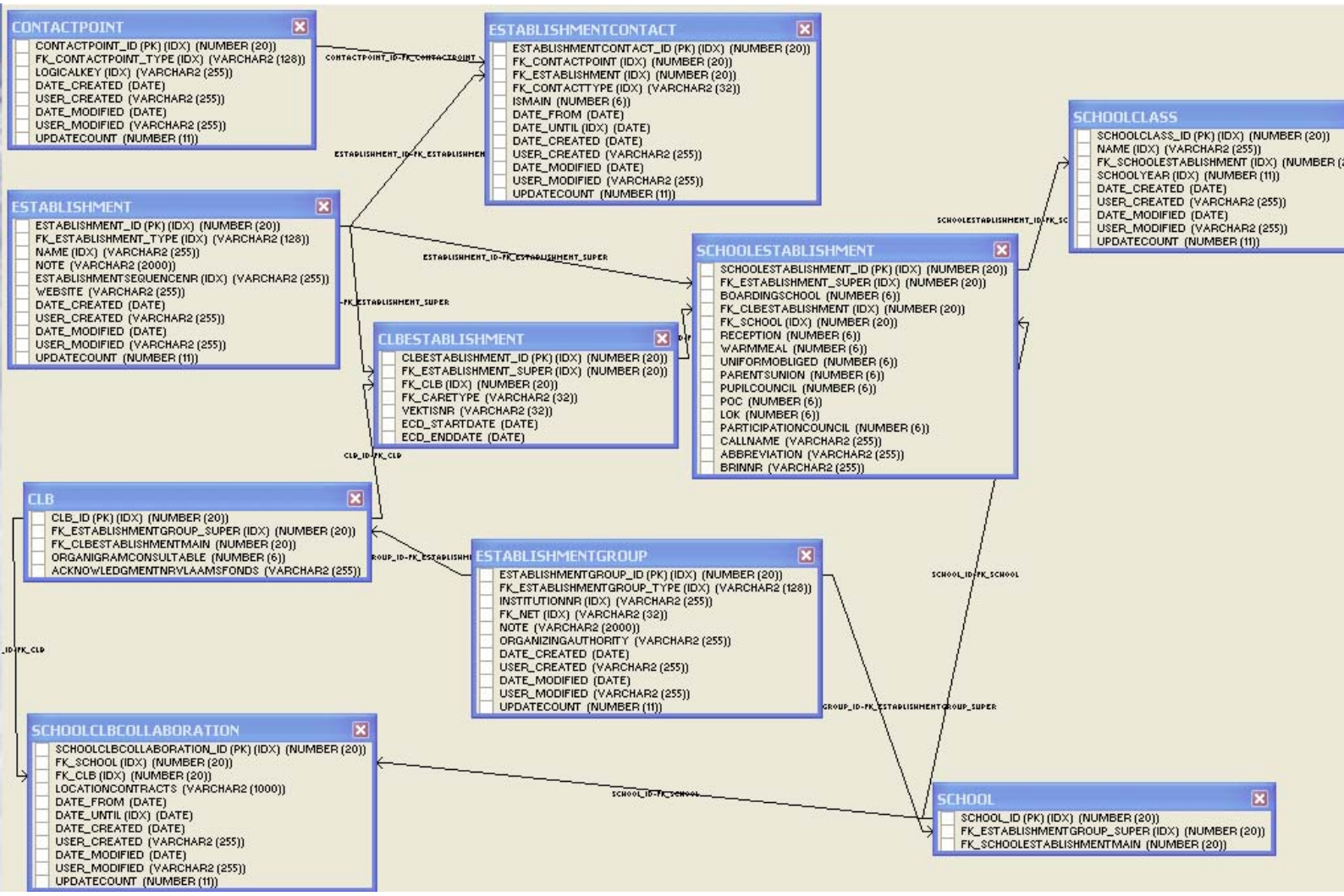
Kidos is een systeem waarin gegevens van kinderen worden bijgehouden voor de Jeugd Gezondheids Zorg (JGZ). Om een beter beeld te krijgen van de kinderen is het wenselijk om de gegevens uit Kidos te koppelen met andere systemen. Een voorbeeld hiervan is de koppeling van de schoolresultaten met een kind dossier uit het Kidos systeem. Dit betekent dat leerlingen uit het School Resultaten (SR) systeem gerelateerd kunnen worden aan kinderen uit het Kidos schema.

Kidos heeft een omvangrijk databaseschema om verschillende gegevens over kinderen bij te kunnen houden. In het geïntegreerde systeem staan schoolgegevens centraal. Daarom is ervoor gekozen om een subset te nemen van het Kidos schema. Met deze subset van Kidos is het mogelijk de schoolresultaten van kinderen te relateren aan kinderen in het Kidos systeem.

Het Kidos systeem gebruikt een Oracle DBMS. Om het relationele Kidos schema te converteren naar een equivalent ORM model wordt daarom de algoritmische aanpak die in het paragraaf 9.5 beschreven is gebruikt. In de rest van dit hoofdstuk wordt het stappenplan toegepast op het Kidos schema met als doel dit om te zetten in een equivalent ORM model. Om dit begrijpbaar te houden wordt een subset van de Kidos database schema genomen. Deze subset is gekozen zodat het hiermee mogelijk is om de twee database schema's (subset Kidos schema en School registratie schema) te integreren tot een globaal schema. Dit betekent dat hierna de kinderen uit de verschillende systemen gerelateerd kunnen worden.

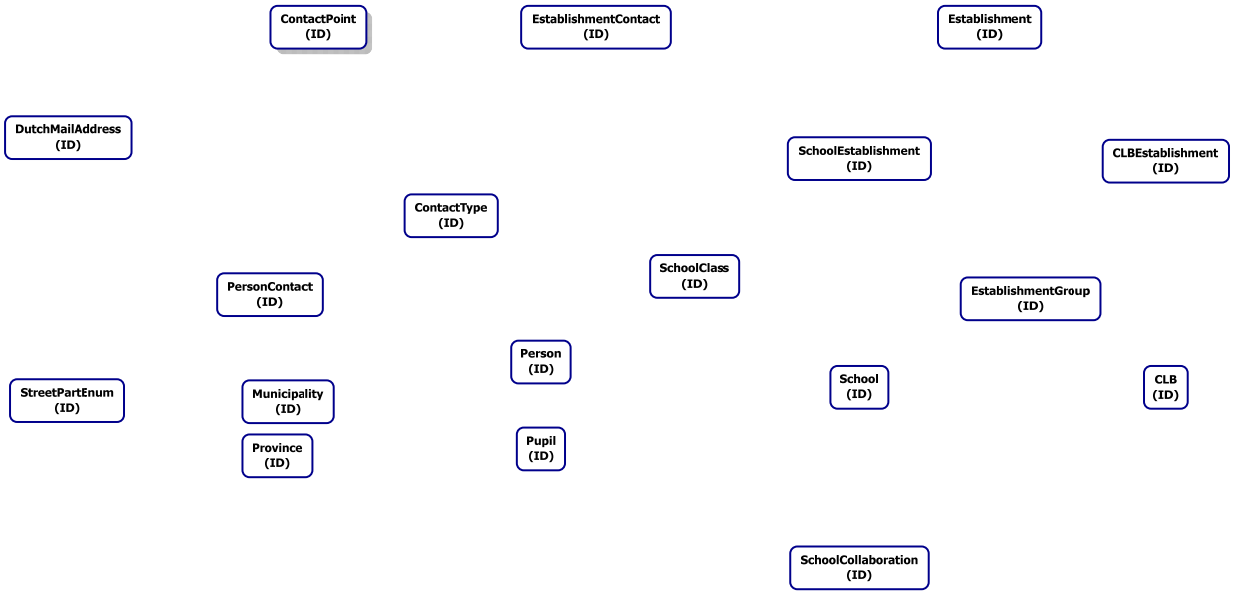
9.8.1 Stap 1

In Figuur 22 zijn de tabellen van het Kidos schema gegeven die gebruikt gaan worden voor de schema integratie. Deze tabellen worden afgebeeld op ORM objecten.



Figuur 22 Relationeel database schema van subset van Kidos Systeem.

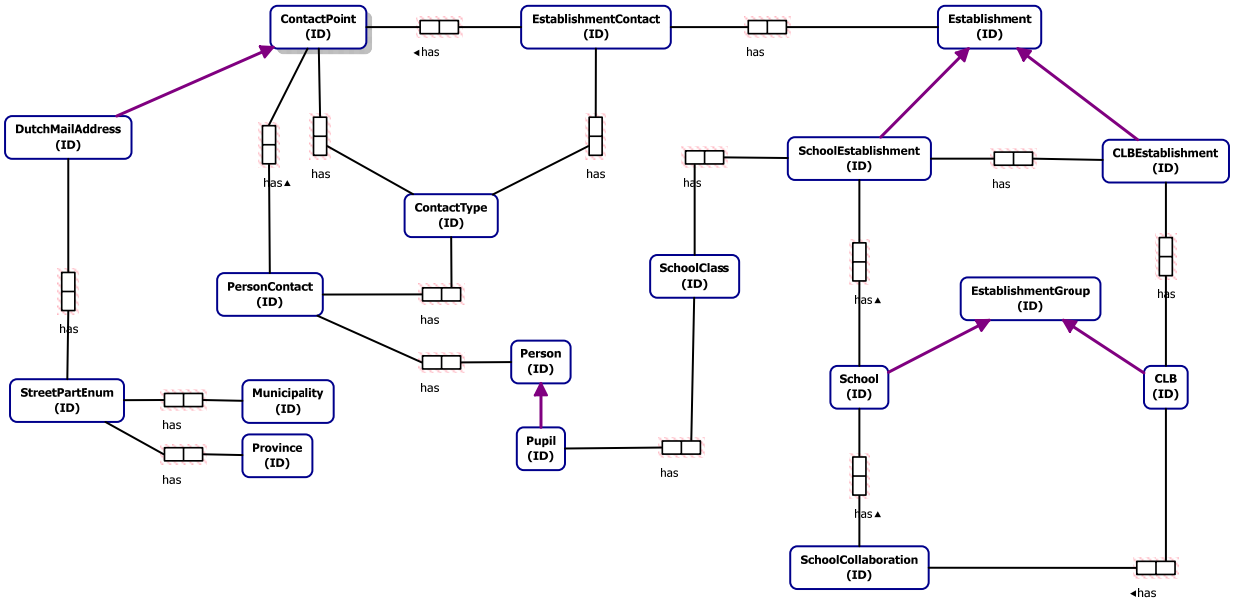
De database tabellen worden afgebeeld op objecten in een ORM model. Ieder object wordt uniek geïdentificeerd door een uniek ID. Het resultaat is te zien in Figuur 23.



Figuur 23 Tabellen uit Afbeelding 7 afgebeeld als ORM objecten.

9.8.2 Stap 2

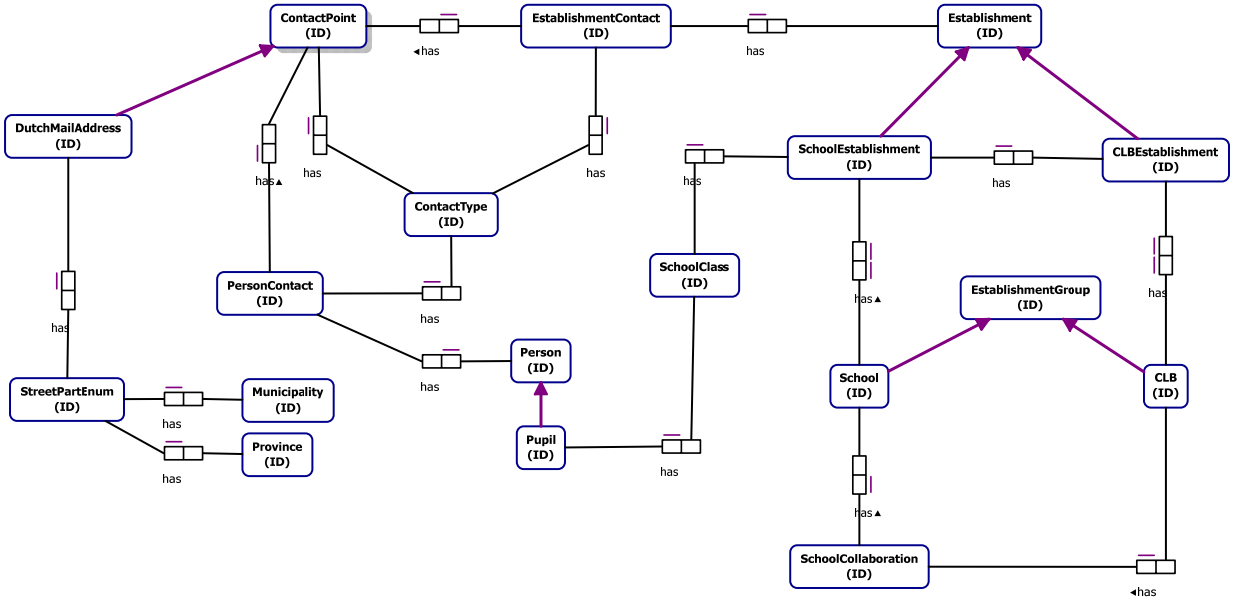
De referentiële sleutels uit Figuur 22 worden aangebracht in het ORM model als rollen tussen de desbetreffende objecten (tabellen in het originele relationele schema), zie Figuur 24. Indien de naam van een attribuut dat onderdeel is van een referentiële sleutel het woord *super* bevat wordt er vanuit gegaan dat er sprake is van een specialisatie/generalisatierelatie.



Figuur 24 De sleutels die afgebeeld zijn in Figuur 22 zijn aangebracht in het ORM model.

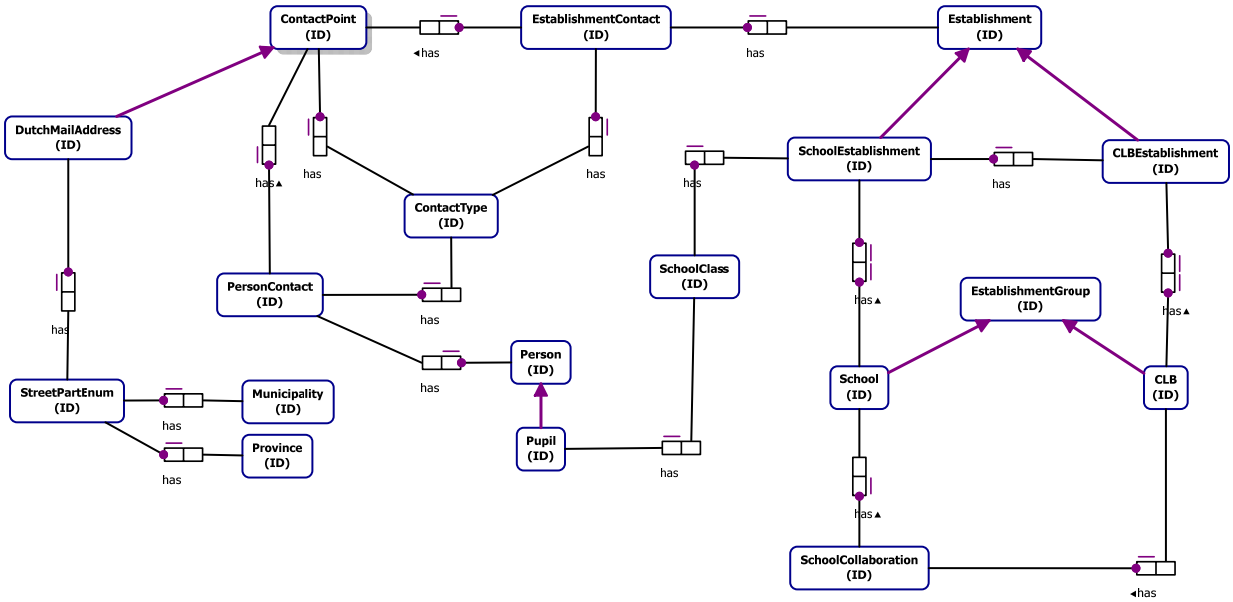
9.8.3 Stap 3 en 4

Het toevoegen van uniciteitsconstraints en mandatory-role-constraints.



Figuur 25 In deze afbeelding zijn de uniciteitsconstraints toegevoegd.

Het probleem met het oorspronkelijke database schema (Figuur 22) is dat de mandatory-role-constraints niet aanwezig zijn. In Figuur 22 zijn alle attributen van tabellen standaard NULL met uitzondering van primaire sleutels en met het attribuut FK_Establishment_Type van de Tabel Establishment. Deze situatie is niet wenselijk, daarom is er gekeken of de attributen wel of niet een waarde hebben voor alle records in een tabel. Aan de hand daarvan zijn de mandatory-constraints afgeleid en toegevoegd aan het ORM model. Zo blijkt dat het merendeel van attributen die gebruikt worden als referentiële sleutels altijd een waarde bevat. Het resultaat hiervan is te zien in Figuur 26.



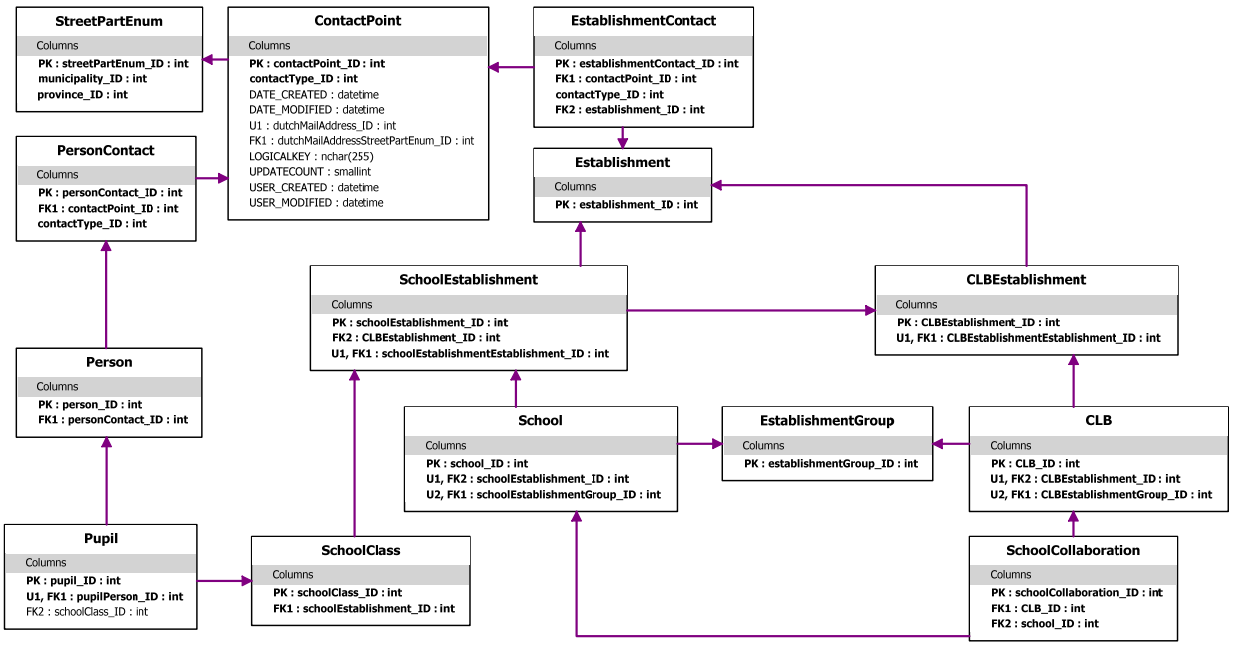
Figuur 26 Het ORM model aangevuld met mandatory-role-constraints.

9.8.4 Tussenstap – Controleren van equivalentie

Het is handig om regelmatig te controleren of het ORM model equivalent is aan het database schema. Dit wordt als volgt gedaan:

- genereer uit het ORM model het relationele schema,
- vergelijk dit schema met het originele schema.

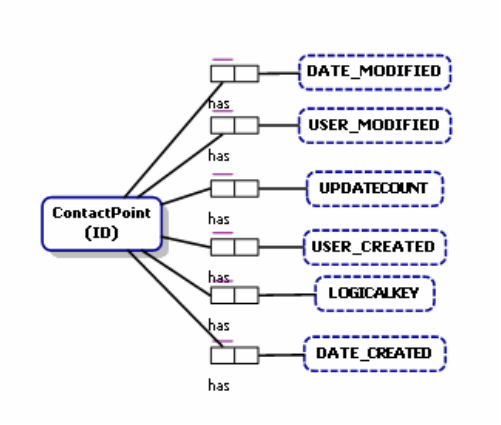
Het schema uit Figuur 27 is hetzelfde als dat uit Figuur 22 met uitzondering van de naamgeving van de attributen. Dit betekent dat het ORM schema equivalent is aan het relationele database schema waarmee begonnen is.



Figuur 27 Het relationeel schema gegenereerd uit het ORM model.

9.8.5 Stap 5

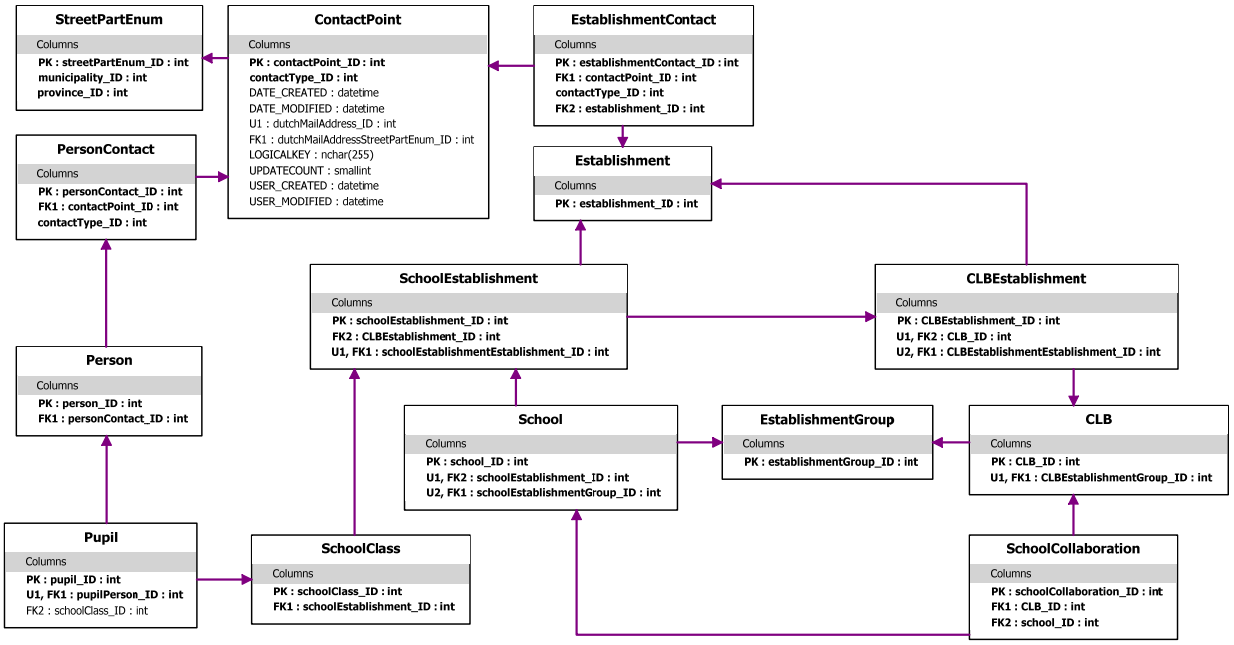
In stap 1 zijn de tabellen omgezet naar ORM objecten. In stap 5 worden de attributen van de tabellen toegevoegd aan objecten in het ORM model. Dit resulteert er in dat ieder attribuut met uitzondering van referentiële sleutels en primaire sleutels wordt afgebeeld als een value-type die verbonden is d.m.v. een rol met een object. Om het ORM model overzichtelijk te houden wordt per object (tabel in het originele relationele schema) een nieuw diagram gemaakt. De objecten/rollen/value-types die in meerdere diagrammen gebruikt worden, worden gevisualiseerd met een schaduw. Alleen voor *ContactPoint* zijn value-types toegevoegd, zie Figuur 28. Bij de overige objecten kunnen op dezelfde wijze value-types worden toegevoegd.



Figuur 28 Bij het object ContactPoint zijn de attributen toegevoegd.

9.9 Conclusie

In Figuur 26 is het uiteindelijke ORM model weergegeven. Hieruit kan dan het Relationele schema gegeneerd (Figuur 29) worden wat equivalent (met uitzondering van de naamgeving) is aan het oorspronkelijke database schema (Figuur 22). Zoals eerder aangegeven is, zijn alleen voor de tabel *ContactPoint* attributen toegevoegd die niet onderdeel zijn van de primaire of referentiële sleutel.



Figuur 29 Het Complete relationele database schema.

Het handmatig reconstrueren van een ORM model dat equivalent is aan een relationeel database schema is arbeidsintensiever dan het automatisch generen van een dergelijk model met een tool zoals NORMA. Echter dit handmatige proces heeft ook een aantal voordelen ten opzichte van het automatisch generen van een ORM diagram, namelijk:

- Het is niet altijd duidelijk of een referentiële sleutel wordt gebruikt voor een generalisatie/specialisatie relatie of een associatie relatie. Bij het automatisch generen van een dergelijk model wordt altijd een associatie relatie gekozen. Dit valt wel te verklaren omdat zowel een associatie relatie als een generalisatie/specialisatie relatie in een relationeel database schema gemodelleerd kunnen worden met behulp van een referentiële sleutel. Er zijn echter database management systemen waarin een generalisatie/specialisatie relatie onderdeel kan zijn van het schema. Postgres is zo'n DBMS dat dergelijke relaties ondersteunt. Onder de motorkap wordt een dergelijke generalisatie/specialisatierelatie door het DBMS geïmplementeerd als een associatie relatie. Concluderend kan gesteld worden dat het niet altijd duidelijk is of een referentiële sleutel gebruikt wordt als een associatierelatie of een generalisatie/specialisatie relatie.

- In de praktijk gebeurt het nogal eens dat mandatory-role-constraints (NULL/NOT NULL constraints op attributen) niet altijd consequent worden gebruikt in het relationele schema. Door de data in de tabel te inspecteren en dus niet alleen de schema structuur kan beter worden vastgesteld wat voor mandatory-role-constraint er impliciet gebruik worden.

Bij reverse engineering van het bestaande Kidos schema zijn een aantal slordigheden ontdekt in het Kidos model.

- Als we kijken naar de tabellen (CLBEstablishment, CLB) en (SchoolEstablishment, School) dan ontdekken we dat er een referentiële cycle is.

- In de tabel Contactpoint is een logische sleutel (manier om referentiële sleutel na te bootsen) gedefinieerd. Deze verwijst naar een andere tabel/object. Echter het is niet duidelijk waarnaar.

- Er zitten tabellen in het schema zonder data (niet opgenomen in het ORM schema).

- Data duplicatie (redundantie is geen uitzondering), zie bijvoorbeeld dat er onnodig vaak verwezen wordt naar bijvoorbeeld *ContactType*.

- Het ontbreken van NOT NULL constraints op attributen is slordig en kan leiden tot het ontbreken van gegevens. Het ontbreken van waardes (NULL values) kan gebruik van aggregatie bij functies in queries onnauwkeurige/foute antwoorden opleveren. In [24] wordt uiteengezet waarom het af te raden valt om geen NOT NULL constraints te gebruiken.

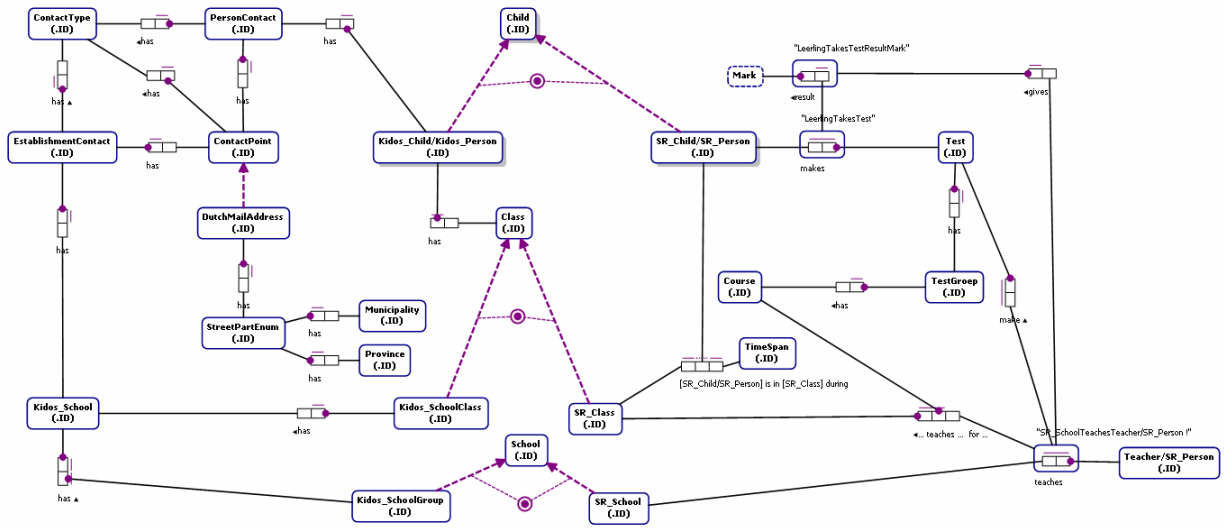
Het is misschien niet onverstandig om het database schema van Kidos eens goed te evalueren en herontwerp toe te passen waar nodig. Dergelijke verbeteringen op het database schema zullen uiteindelijk leiden tot beter te onderhouden applicaties.

10 Schema Integratie

Het integreren van de twee schema's is gedaan met de methode zoals beschreven is in [17]. Het resultaat hiervan is te zien in Figuur 30. Hieronder volgt een omschrijving van de verschillende integratiemethoden zoals beschreven zijn in [17], namelijk:

- Homoniem (hom): twee dezelfde namen in verschillende lokale schema's maar met andere semantiek, deze moeten met verschillende namen in het globale schema opgenomen worden.
- Synoniem (syn): twee verschillende namen met dezelfde semantiek worden in het globale schema opgenomen als één naam (samenvoegen).
- Conversie (conv): semantisch gerelateerde typen, die in verschillende eenheden of detailniveau (korreligheid) gerepresenteerd zijn, moeten een conversie ondergaan om opgenomen te kunnen worden in het globale schema.
- Default value conflicts (def): sommige attributen in lokale schema's gebruiken default values. Indien twee attributen van lokale schema's naar één attribuut in het globale schema geconverteerd worden moet wel duidelijk zijn welke default value wordt gebruikt.
- Differentiatie (diff): als in lokale schema's twee of meer entiteiten dezelfde rol vervullen kunnen ze verschillen in de verzameling attributen. Het verschil in attributenverzamelingen kan opgelost worden op twee manieren:
    - De vereniging van alle attriubeten uit de lokale schema's opnemen in het globale schema. Indien een attribuut geen waarde heeft in het lokale schema kan een default waarde gebruikt worden.
    - Het toepassen van specialisatie/generalisatie, waarbij de gemeenschappelijke attributen opgenomen worden in de generalisatie objecten en de niet gemeenschappelijke attributen per entiteit worden opgenomen in specialisatie objecten.

Hierna volgt een korte uitleg van het geïntegreerde schema. De uitleg is opgedeeld in de verschillende integratiemethoden zoals hierboven beschreven.



Figuur 30 Samenvoegen van de twee schema's in een globaal schema.

Differentiatie

In de twee lokale schema's zijn drie soorten personen te identificeren, namelijk Kidos_Child *(Pupil*), *SR_Child* (*Leerling*), *Teacher* (SR_Docent). In het globale schema is *Child* de generalisatie van *Kidos_Child* en *SR_Child*.

Om records in het globale schema 1 op 1 te kunnen relateren aan het lokale schema wordt in het globale schema bij generalisaties een attribuut (rol) toegevoegd met daarin het ID van de lokale waarde.

Dit zelfde principe van differentiatie wordt toegepast op *Class* met subtypes *Kidos_Class* en *SR_Class* en *School* met subtypes *SR_School* en *Kidos_School.*

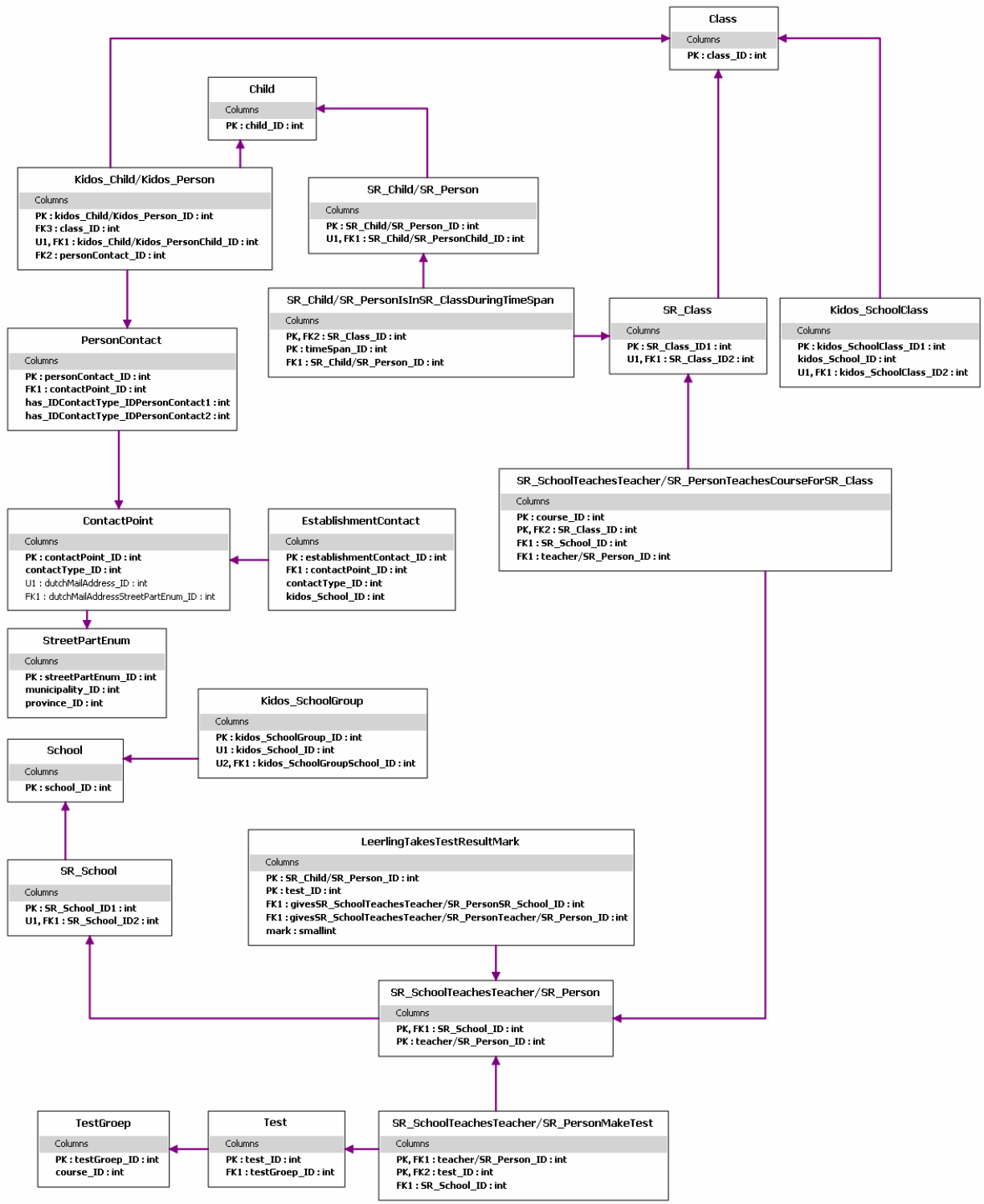
Synoniem

In het SR schema worden Nederlandse namen gebruikt, deze worden vertaald naar het Engels en worden gebruikt voor het mappen van de attributen. Voorbeelden van synoniemen zijn *Tijdsspanne* en *Timespan*, *Schooljaar* en *Schoolyear*, enz.

Bij nadere inspectie en navraag blijkt dat verschillende concepten binnen de twee lokale schema's equivalent zijn (homoniemen). Om deze problemen op te lossen worden de conversies gedaan die bij Conversies beschreven worden.

Conversies

In het Kidos en SR-systeem wordt totaal anders omgegaan met het begrip scholen. In het globale schema zijn de lokale Kidos objecten *School* en *Establishmentgroup* samengevoegd tot een *Schoolgroup*. Een Schoolgroup kan gezien worden als een scholengemeenschap. *Establishment* en *SchoolEstablishment* worden samengevoegd tot een *Kidos_School* in het globale schema.



Figuur 31 Het globale relationele database schema.

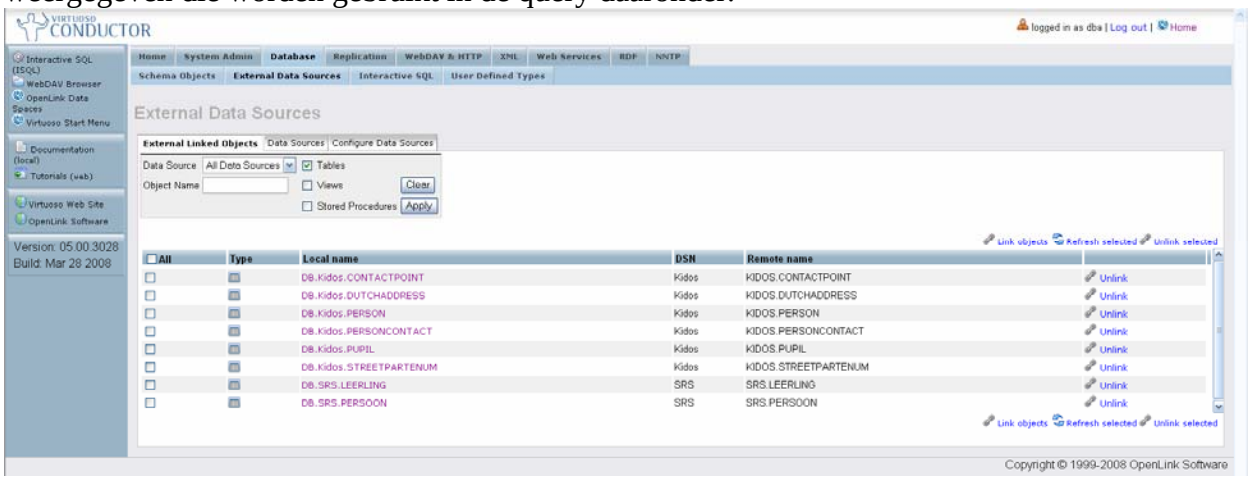
10.1 Bruikbaarheid van Globale Schema

Met het globale schema wordt het mogelijk om queries uit te voeren waarin Kidos_Child gerelateerd wordt aan SR_Child. Dit relateren zal kunnen gebeuren door de postcode van SR_Persoon te vergelijken met het “zip” van het Kidos_Child.

Doordat op een dergelijke manier kinderen uit de twee lokale schema's gerelateerd kunnen worden in het globale schema is het mogelijk om queries te maken die gegevens uit de lokale schema's relateren aan het globale schema. Zo wordt het mogelijk om gegevens op te vragen over de schoolresultaten van kinderen vanuit het Kidos Programma. Deze gegevens zijn fysiek alleen beschikbaar op het SR-systeem. Nu kunnen vragen gesteld worden zoals: is er een verband tussen de thuissituatie en de schoolresultaten? Ook zouden schoolresultaten beschikbaar gemaakt worden voor gebruikers van het Kind dossier.

10.2 Virtuoso Virtual Database – Implementatie van het ontwerp

Om de bruikbaarheid van het ontwerp te illustreren is er met Virtuoso Universal Server (Openlink) een implementatie gemaakt. In het voorbeeld worden de twee verschillende lokale database systemen gekoppeld. Ter illustratie wordt een deel van het globale database schema geïmplementeerd. In het globale systeem worden kinderen uit de twee lokale schema's gekoppeld door middel van de postcode. Om het voorbeeld illustratief te houden zijn alleen de tabellen opgenomen in het globale schema die gebruikt worden om de kinderen uit de twee verschillende systemen te koppelen. In Figuur 32 zijn de externe database tabellen weergegeven die worden gebruikt in de query daaronder.




```
SELECT DISTINCT
      SRS.voornaam, SRS.achternaam, SRS.postcode,
      DB.Kidos.DutchAddress.zip, DB.Kidos.Person.firstname, DB.Kidos.Person.middleName, DB.Kidos.Person.familyName
FROM    DB.Kidos.Person
      JOIN DB.Kidos.Pupil ON (PUPILPERSON_ID = PERSON_ID)
      JOIN DB.Kidos.PersonContact ON (DB.Kidos.Person.PERSONCONTACT_ID = Kidos.PersonContact.PERSONCONTACT_ID)
      JOIN DB.Kidos.ContactPoint USING (CONTACTPOINT_ID)
      JOIN DB.Kidos.DutchAddress ON (DB.Kidos.DutchAddress.DUTCHADDRESSCONTACTPOINT_ID = DB.Kidos.ContactPoint.CONTACTPOINT_ID)
      JOIN DB.SRS.PERSOON as pSRS ON (postcode = zip)
      JOIN DB.SRS.LEERLING USING (persoon_id)
```

Figuur 32 Overzicht van de verschillende tabellen uit de twee lokale database systemen.

Het mag duidelijk zijn dat een dergelijke query die leerlingen relateert uit de twee lokale systemen niet ideaal is. Om het globale schema goed te implementeren is het noodzakelijk om views te maken van de externe tabellen in het globale database systeem. In deze views worden de problemen zoals conversie en default value conflict opgelost.

Nadat de tabeldefinities van de lokale tabellen in het globale schema zijn opgenomen en conflicten zijn verholpen met betrekking tot conversie, default value en type kan er begonnen worden met het maken van views die de data uit de verschillende systemen relateert. Dit is

dus de daadwerkelijke implementatie van het globale database schema. Dit relateren gebeurt in dit voorbeeld door middel van de postcode. Het is echter wenselijk gebruik te maken van een gemeenschappelijke identificatie wanneer data uit de verschillende bronnen gerelateerd moet worden. In dit voorbeeld is dit gedaan met behulp van de postcode, ook al is dit niet ideaal. Een postcode identificeerd een persoon niet echt uniek echter. Een betere keuze (niet beschikbaar in dit voorbeeld) zou het sofinummer (burgerservicenummer) zijn omdat dit een persoon uniek identificeerbaar maakt. Het zou een idee zijn om het burgerservicenummer toe te voegen aan de lokale database systemen. Een andere optie zou zijn om meerdere attributen te combineren zoals postcode, huisnummer en geboortedatum. Dit vergroot de kans dat de juiste data wordt gerelateerd.

Een ORM model dat een globaal schema representeert kan omgezet worden in een relationeel model, zie Figuur 31. Echter de SQL code moet nog handmatig bewerkt/gemaakt worden om de tabeldefinities om te zetten in views die gebruik maken van de lokale schema's. Het uiteindelijke globale database ontwerp wordt dus omgezet in code die bruikbaar is voor een Federated Database System.

10.3 Vervolg

Om de verschillende federatieve database producten (DB2 Federated Server, Virtuoso Virtual Database en Oracle Gateway) te onderzoeken op bruikbaarheid en toepasbaarhied is het noodzakkelijk om criteria op te stellen. Aan de hand van deze criteria kunnen de verschillende producten vergeleken worden op bruikbaarheid en toepasbaarheid.
Criteria ter evaluatie van federatieve database producten:

- global query optimalisatie
- ondersteuning om conflicten (conversie, default value) al op tabel/view niveau op te kunnen lossen in het globale schema.
- welke soort gegevensbronnen ontsloten kunnen worden
- wat is de performance (doorvoersnelheid) van gegevens
- is transactie verwerking mogelijk (wel of geen 2phase commit)
- ondersteuning van tooling

Het mag duidelijk zijn dat eventuele voordelen of tekortkomingen tussen de verschillende producten pas aan het licht komen bij het implementeren van een oplossing van een redelijke omvang. Een goed ontwerp van het globale database schema is belangrijk voor het succes en de begrijpbaarheid van een dergelijk systeem. De methode om meerdere lokale database schema's te integreren, zoals beschreven is in dit document, kan hier dan ook in faciliteren.

10.4 Conclusie

Dit document laat zien hoe op een systematische wijze schema integratie kan plaatsvinden aan de hand van een praktijkvoorbeeld. Eerst wordt op een systematische wijze reverse engineering toegepast om een lokaal database schema te converteren naar een ORM model, voordat twee lokale ORM diagrammen gecombineerd kunnen worden tot één globaal ORM model dat vervolgens weer leidt tot één Federated database implementatie.

Appendix A - Requirements Document

Inleiding

Dit document beschrijft de functionele requirements van het KidosR systeem aan de hand van usecases. Het KidosR systeem is een uitbreiding van het Kidos systeem waarmee elektronische kind dossiers (EKD) worden bijgehouden. De R in KidosR staat voor Relaties, het kenmerk van de uitbreiding is dat het mogelijk wordt om dynamische relaties aan te brengen. Het is buiten het bereik van dit document om een gedetailleerde omschrijving te geven van deze dynamische relatie structuur omdat dit te maken heeft met de technische implementatie, zie hiervoor het technisch ontwerp.

Om de use-cases goed te begrijpen wordt eerst het probleemdomein van Kidos verduidelijkt. Uit deze beschrijving volgt voor welke partijen (actors) een belangrijke rol is weggelegd binnen KidosR. Vervolgens worden de requirements verduidelijkt door middel van functionele voorbeelden van het KidosR systeem. Uit deze voorbeelden komt duidelijk naar voren dat het systeem generiek opgebouwd dient te worden. Aan het eind van dit document volgen de use-cases met een beschrijving. Deze usecases beschrijven de functionele requirements voor KidosR.

Beschrijving probleemdomein

Kinderen in Nederland hebben recht op preventieve jeugd gezondheids zorg (JGZ). De JGZ is verantwoordelijk voor het realiseren van preventieve zorg voor kinderen (0-19 jaar).

De gemeentes bieden preventieve jeugd zorg aan. Om deze zorg te realiseren zijn er o.a. de volgende twee instanties: de gemeentelijke gezondheidsdienst (GGD) en de thuiszorg.

De thuiszorg is verantwoordelijk vanaf de geboorte tot de basisschoolleeftijd. Daarna neemt de GGD deze taak over. Samen verwezenlijken ze dus het recht op de JGZ voor kinderen in opdracht van een gemeente.

De GGD en thuiszorg houden een elektronisch kind dossier (EKD) bij voor alle kinderen in een gemeente. Er wordt vaak de fout gemaakt dat het EKD en het elektronisch patiënten dossier (EPD) hetzelfde zijn. Dit is niet het geval, omdat het EPD alleen ‘zieke’ mensen betreft (patiënten) en het EKD alle kinderen betreft en een preventief karakter heeft. In het EKD worden gegevens over kinderen bijgehouden. Deze gegevens worden gebruikt door de GGD en thuiszorg om preventieve zorg te bieden.

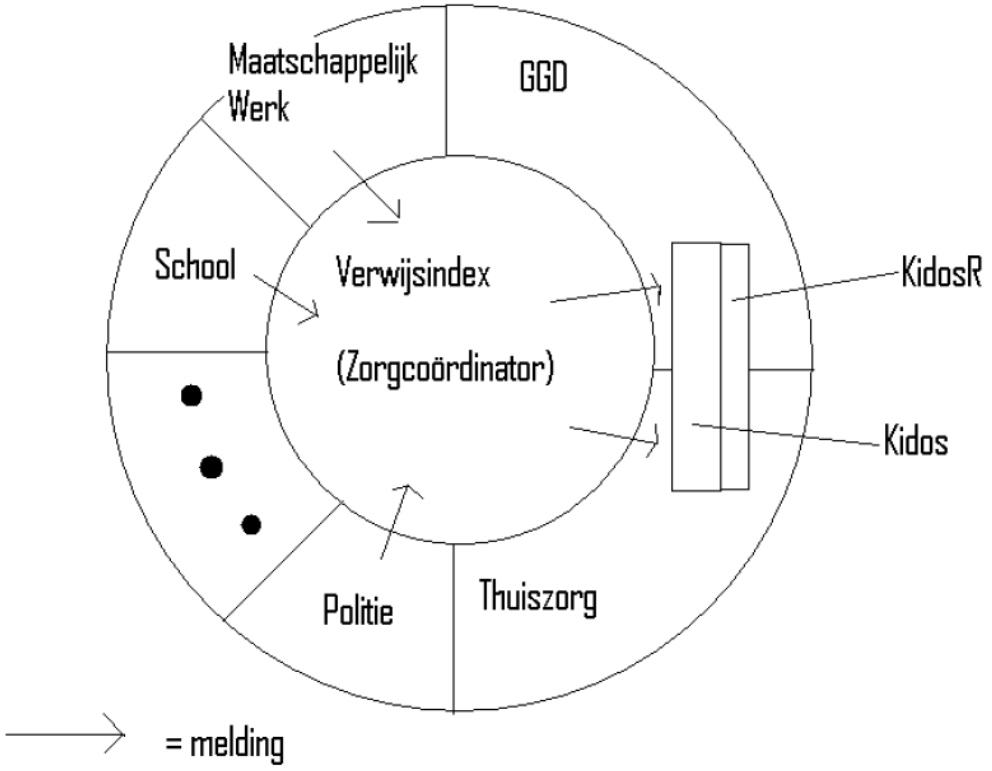
Naast de GGD en thuiszorg zijn er andere instanties betrokken bij de JGZ. Een aantal van deze organisaties is aangegeven in de zorgcirkel, zie Figuur 33. De verwijsindex maakt het mogelijk dat verschillende instanties informatie uitwisselen (berichten) en dat de zorg voor kinderen wordt gecoördineerd door de zorgcoördinator. In de praktijk betekent dit, dat professionele hulpverleners een melding kunnen doen bij de verwijsindex, de zorgcoördinator moet de zorg die nodig is gaan coördineren met de verschillende instanties. Een voorbeeld is dat justitie meldt dat Jantje al voor de derde keer opgepakt is of een docent ziet dat Jantje alweer blauwe plekken heeft op zijn armen. Deze melding wordt vervolgens doorgegeven aan

het EKD en indien noodzakelijk coördineert de zorgcoördinator de hulp die geleverd moet worden door de verschillende instellingen.

Indien een kind voor een consult bij de jeugdarts komt krijgt deze o.a. informatie over de melding(en) uit de verwijsindex. Ook is het mogelijk dat de GGD en thuiszorg zelf meldingen doorgeven aan de verwijsindex.

De zorgcoördinator zorgt er voor dat op een melding een actie volgt.

Om een kind beter te kunnen volgen is het noodzakelijk om zijn psychosociale netwerk in kaart te brengen. Door bestudering van een dergelijk netwerk kunnen aannames (hypotheses) gedaan worden over een kind. Een voorbeeld van een dergelijke hypothese kan zijn dat, indien een kind uit een gebroken gezin komt, deze een grotere kans heeft om een probleem te hebben. Een ander hypothese zou kunnen zijn dat een kind dat regelmatig van woonsituatie verandert meer kans heeft om verwaarloosd te worden.



Figuur 33 Zorgcirkel. De meldingen van verschillende instanties worden doorgegevens aan de GGD door **de verwijsindex. De zorgcoördinator is verantwoordelijke dat er actie wordt ondernomen op een melding. De GGD en thuiszorg gebruiken het Kidos systeem voor het EKD.**

Probleemomschrijving

Om een compleet psychosociaal beeld te krijgen van cliënten (kinderen) is het noodzakelijk om het sociale netwerk van een cliënt in kaart te brengen. Het Kidos systeem wordt gebruiken door medewerkers van de GGD en thuiszorg om het EKD van kinderen bij te houden en te raadplegen. Om een cliënt goed te kunnen volgen moet het mogelijk zijn om de relaties die een kind heeft met zijn omgeving vast te leggen.

Door het vastleggen van verschillende relaties tussen cliënten en de omgeving wordt het sociale netwerk van een kind inzichtelijker. Uit het beeld wat ontstaat uit de aangebrachte relaties kunnen dan conclusies getrokken worden door te kijken naar afwijkende patronen. Hierbij moet gedacht worden aan het al dan niet aanmerken van een kind als een potentieel probleemgeval.

Het sociale netwerk van een kind valt te verdelen in een aantal categorieën (niveaus):

- familie: wie zijn de opvoeders, broertjes, zusjes.
- instellingen: met welke instellingen heeft de cliënt te maken, zoals school, kinderdagverblijf, huiszorg, maatschappelijk jeugd werk, etc.
- leefomgeving: in welke omgeving bevindt de cliënt zich, zowel sociaal als geografisch.

Het kan belangrijk zijn om de relaties te voorzien van tijdsgegevens zodat er een volledig beeld ontstaat over het verleden en de huidige situatie van een kind.

Mogelijk voorbeeld voor het gebruik van KidosR

Door de thuissituatie in kaart te brengen is bekend wie, waar en wanneer een kind opvoedt. Indien er zich onregelmatigheden voordoen in een dergelijke situatie zou dit opgemerkt kunnen worden door KidosR. Voorbeelden van onregelmatigheden zijn: vaak verhuizen, regelmatig wisselen van school, gebroken gezinnen, etc.

Dit soort onregelmatigheden kunnen gebruikt worden om potentiële probleemgevallen te identificeren.

Een gemeente beslist dat ze sport willen promoten onder de jeugd. Om dit te verwezenlijken worden in scholen bij alle kinderen in de gemeente enquêtes afgenomen. Deze enquêtes worden verwerkt met de computer en de uitslagen van de enquêtes worden opgeslagen in het KidosR systeem.

Doordat de gemeente weet welke kinderen wel en niet sporten en wat voor sport de verschillende kinderen doen is het mogelijk om deze gegevens te benutten. Een gemeente kan beslissen om bepaalde groepen te motiveren om te gaan sporten door bijvoorbeeld sport na schooltijd aan te bieden. Deze gegevens worden opgenomen in het KidosR systeem waardoor de instanties een beter psychosociaal beeld krijgen van cliënten. Het zal bijvoorbeeld kunnen blijken dat kinderen uit bepaalde buurten en sociale lagen van de samenleving niet of nauwelijks sporten.

Cliënten kunnen kinderen krijgen (tienerzwangerschap), het is voor de instellingen handig om te weten dat een cliënt een vader of moeder is en wie dan de zoon(s)/dochter(s) (tevens een cliënt) zijn.

Indien een cliënt hulp krijgt van een instelling is belangrijk om te weten welke hulpverleners hierbij betrokken zijn en van welke instellingen. Door relaties tussen de cliënt en de hulpverleners vast te leggen wordt het duidelijk wie, waar en wanneer verantwoordelijk is.

Indien onderwijsinstellingen gegevens aanleveren met betrekking tot de prestaties en het gedrag van kinderen kunnen sommige problemen vroegtijdig gesignaleerd worden. Een

docent zal het sneller zien dat een kind vaak afgeleid is of niet sociaal gedrag vertoont dan een arts die een kind kortdurend ziet tijdens een bezoek. Deze hypothesen zouden dan opgenomen moeten worden in het systeem, een arts kan hierop participeren door eventuele constateringen te bevestigen.

Wanneer er wordt vastgelegd met welke mensen een bepaalde cliënt bevriend is kunnen sociale netwerken in kaart gebracht worden. Sommige gemeenten hebben problemen met bepaalde groepen jongeren die overlast veroorzaken. Als een dergelijke groep jongeren te identificeren valt met KidosR kan het probleem opgelost worden.

Door het bijhouden van gegevens over seksueel overdraagbare aandoeningen onder de jeugd kan inzicht verkregen worden in dergelijke ziektes onder de jeugd. Het kan bijvoorbeeld blijken dat bepaalde sociale lagen uit de samenleving een verhoogd risico hebben om een dergelijke ziekte op te lopen.

De actors voor het KidosR systeem:

- **Professional:** de professionals mogen alleen gegevens in een EKD inzien maar niet wijzigen.
- **Arts:** mag zowel de dossiers van zijn cliënten inkijken als wijzigen, een arts is ook een professional maar dan met meer rechten.
- **Project medewerker Kidos (beheerder):** beheerder van het systeem. Een project medewerker van Kidos zal het KidosR systeem gaan onderhouden en wijzigen.
- **Onderzoeker:** om rapportage over alle kinderen te maken zal een onderzoeker de gegevens kunnen onderzoeken. Hierbij moet worden opgemerkt dat een onderzoeker alleen anonieme statistische gegevens krijgt (gegevens vallen niet te relateren aan individuen).

Het is belangrijk te realiseren dat alleen personen binnen de GGD en thuiszorg toegang hebben tot het Kidos systeem. Informatie van derde partijen komt dus binnen via de verwijsindex en zal ingevoerd moeten worden door een arts die een constatering van een derde kan bevestigen en dat opnemen in het dossier.

Use-case

De use-cases zijn generiek van aard omdat de functionele eisen afhankelijk zijn van wat de gebruikers van KidosR willen gaan vastleggen over kinderen.
De use-cases zijn verdeeld in twee categorieën, namelijk:

- Visualisatie en Rapportage
- Relatiebeheer

In Figuur 34 worden de verschillende use-cases weergegeven. Vervolgens volgt een korte beschrijving van de verschillende use-cases en daarna volgt een gedetailleerde beschrijving.

Visualisatie en Reportage

Artsen, professionele hulpverleners en onderzoekers moeten de mogelijkheid hebben om de aangebrachte gegevens van cliënt(en) te bestuderen. Voor het evalueren van een dergelijk relatie netwerk is het noodzakelijk om dit netwerk op verschillend niveau te bekijken. Zowel op globaal als op detail niveau moet een dergelijk netwerk geanalyseerd kunnen worden.

Hierdoor moet de medewerker een compleet beeld kunnen vormen van een cliënt. Het systeem moet in staat zijn om eventuele afwijkingen in het netwerk te constateren, die een mogelijk probleem kunnen identificeren. Als een afwijking geïdentificeerd is, moet er een hypothese rapport aangemaakt worden. Een hypothese die niet door het systeem wordt geïdentificeerd moet ook door een medewerker aangebracht kunnen worden. In een hypotheserapport moet een deel van het netwerk gemarkeerd worden, dat volgens de medewerker een afwijkend patroon bevat, waarop de hypothese gebaseerd is. Uiteindelijk moet een hypothese bevestigd of ontkracht worden door een rechthebbende (arts).

Het moet mogelijk zijn om rapportages over aangebrachte relaties en hypothese te maken.

Er zijn de volgende functionele eisen met betrekking tot visualisatie en rapportage voor het KidosR systeem:

- Bekijken van netwerk op verschillend detail niveau
- Labelen van (een deel van) het netwerk met commentaar en/of status informatie om eventuele risico gevallen te identificeren. Dit betekent dus dat een hypothese (aanname) wordt gedaan aan de hand van het netwerk met labels.
    - Bevestiging/ontkrachten van een dergelijke hypothese
- Afwijkende patronen moeten door het systeem gemarkeerd worden zodat eventuele risico
- gevallen direct zichtbaar zijn.
- Genereren van rapportages uit de aangebrachte relaties en gegevens.

Relatiebeheer

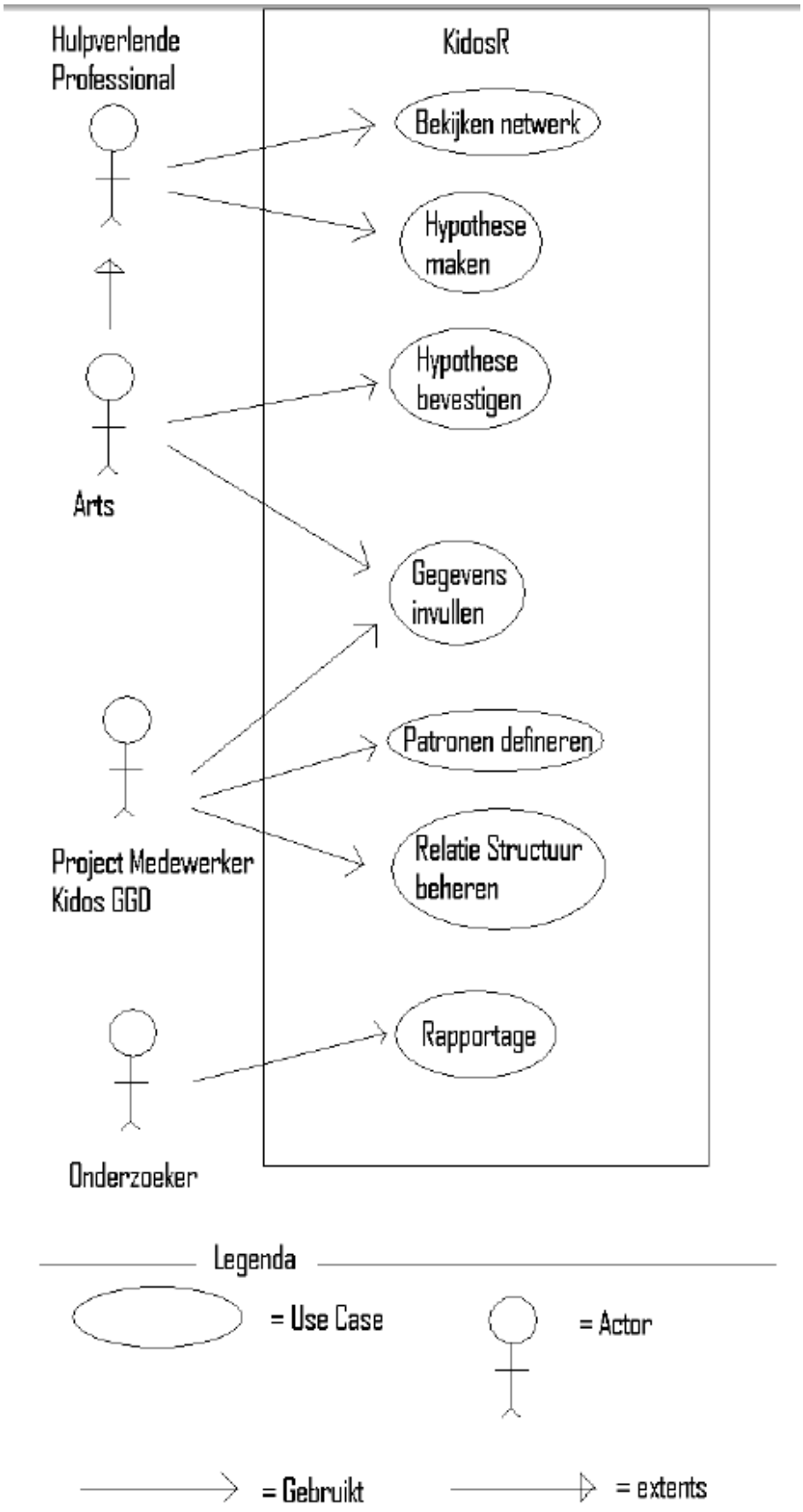
Het definiëren van relaties betekent dat bepaalde relaties beschreven moeten worden. Bijvoorbeeld een medewerker wil iets vastleggen over cliënten; hiervoor moet het mogelijk zijn relaties te definiëren. Nadat relaties gedefinieerd zijn moeten de gegevens ingevuld kunnen worden (instanties van relaties). Ook moeten relaties verwijderd of aangepast kunnen worden.

Een medewerker zal bepaalde afwijkende patronen of standaard patronen moeten definiëren met betrekking tot relaties, om eventuele afwijkingen in het sociale netwerk te kunnen detecteren (patroonherkenning).

Er zijn de volgende functionele eisen met betrekking tot relatiebeheer voor het KidosR systeem:

- Definiëren, verwijderen en aanpassen van relaties
- Vullen van relaties met (instantie) data, eventueel deze data uit de externe omgeving importeren
- Wijzigen/verwijderen van relaties

Aanbrengen van afwijkende en standaard patronen om risico gevallen te kunnen herkennen door gebruik te maken van patroonherkenning.



Figuur 34 Use cases voor het te bouwen systeem.

Use case - Bekijken netwerk

Korte omschrijving

Een medewerker wil het sociale netwerk bestuderen om hieruit conclusies en informatie te kunnen afleiden.

Acteurs

artsen, hulpverlenende professionals

Precondities

de medewerker moet toegang hebben tot een dossier en relaties moeten aangebracht zijn (zowel de structuur als de instanties)

Main flow

1. Bekijken van netwerk
    a. op detail niveau en globaal niveau
    b. filters te definiëren om bepaalde details in netwerk beter te kunnen analyseren
2. Indien afwijkende patronen zijn geconstateerd, moet een hypotheserapport aangemaakt worden
    a. Selecteren van aandachtsgebied in netwerk waarop de hypothese gebaseerd is.
    b. Eventuele opmerkingen toevoegen in het hypotheserapport

Een medewerker kan de verschillende relaties bekijken, zowel op detail als globaal niveau. Door filters te definiëren kan een netwerk overzicht veranderd worden door de gebruiker. Dit is noodzakelijk om bepaalde vragen en/of aandachtsgebieden te bestuderen.

Eventuele afwijkende patronen moet geaccentueerd worden door het systeem, eventueel met verduidelijking waarom een patroon afwijkend is. Dit kan dan onderdeel vormen van een hypothese.

Het idee is dat de medewerkers bepaalde problemen kunnen identificeren of aangeboden krijgen door het systeem.

Alternatieve flows

Postconditie

Er moet een duidelijk beeld over de psychosociale toestand ontstaan van een cliënt. Indien een afwijkend patroon is gevonden is een hypotheserapport aangemaakt.

Use case - Hypothese aanmaken

Korte omschrijving

Indien een professional een netwerk bestudeert kan er geconcludeerd worden dat er iets aan de hand is. Dit kan worden vastgelegd door een hypothese te maken van de constatering. Deze hypothese kan vervolgens bevestigd worden door een arts.

Actors

arts, professional

Precondities

Er is bij het bekijken van een netwerk wat opgevallen dat afwijkt, dit kan zowel door het systeem aangegeven worden (d.m.v. patroonherkenning) als door een medewerker.

Mainflow of events

1. Verdacht patroon markeren
2. Hypothese rapport aanvullen met omschrijving.

Een verdacht patroon moet gemarkeerd worden door dat deel van de relaties in het netwerk een speciale status te geven. Hierdoor is het mogelijk om deze relaties anders weer te geven aan de gebruiker (kleur). Vervolgens kan er een aantekening bij de hypothese gemaakt worden.

Alternatieve flows

Postcondities

Een hypotheserapport, dit bevat:

- een deel of het gehele netwerk is gemarkeerd als onderdeel van de hypothese
- beschrijving van de hypothese

Use case - Hypothese bevestigen

Korte omschrijving	Een hypothese wordt bevestigd of als onterecht beoordeeld. Indien een hypothese is bevestigd moeten er acties gedefinieerd worden om het probleem op te lossen.
Actors	Arts
Precodities	Er moet een hypothese aangemaakt zijn. Dit aanmaken kan gebeuren door het systeem dat afwijkende patronen herkent of door een hulpverlenende professional (aanmaken hypothese).
Mainflow of events	1. Zoek hypothese op voor bepaald kind 2. Bekijk hypothese 3. Beoordeel hypothese a. Bevestig hypothese → Acties ondernemen b. Hypothese verworpen 4. Documenteren van redenen verwerping of bevestiging De hypothese wordt bekeken en er wordt aangegeven of de hypothese klopt of niet. Indien een hypothese klopt zal er dus actie ondernomen moeten worden door de medewerker die de hypothese beoordeelt. De eventuele actie kan omschreven worden in het hypothese rapport. In het rapport moet ook de reden voor verwerping of bevestiging van de hypothese worden opgenomen.
Alternatieve flows	
Postconditie	Een hypothese is beoordeeld op correctheid (bevestigt of verworpen). Indien een hypothese bevestigd is, moeten er acties gedefinieerd zijn om het probleem te verhelpen.

Use case Gegevens invullen

Korte omschrijving	Invullen van de gegevens in het systeem.
Actors	Arts, Project Medewerker GGD Kidos.
Precodities	Relatie structuur moet aanwezig zijn.
Mainflow of events	1. opzoeken van relatie of kind waarvan gegevens ingevuld dienen te worden 2. invullen gegevens 3. opslaan gegevens Opmerking: een arts zal altijd als ingang voor het invullen van gegevens een kind dossier gebruiken.
Alternatieve flows	Het inladen van gegevens uit externe databronnen (tekstbestand, database, Excel bestand, etc). Dit kan alleen gedaan worden door de Project Medewerker GGD Kidos.
Postconditie	Instanties van objecten en relaties zijn aangemaakt.

Use case - Relatie beheer

Korte omschrijving	Relaties moeten toegevoegd, gewijzigd en verwijderd kunnen worden.
Actors	Projectmedewerker Kidos of GGD
Preconditie	De relatie structuur moet aangebracht zijn.
Mainflow of events	Definiëren en aanpassen van de relatie structuur. Het vullen van de relaties met instanties, bijvoorbeeld door deze gegevens te importeren.
Postconditie	Relaties zijn toegevoegd, verwijderd of aangepast in het netwerk, het systeem moet de correctheid van de relaties waarborgen.

Use case - Rapportage

Korte omschrijving	Maken van rapportage, bijvoorbeeld hoeveel procent van alle hypotheses wordt bevestigd.
Actors	Onderzoeker
Preconditie	De relatie structuur moet aangebracht zijn en de instanties moeten beschikbaar zijn.
Mainflow of events	1. Stellen van een vraag aan het systeem. 2. Het resultaat (gegevens) op deze vraag exporteren naar een extern programma. 3. Het importeren en verwerken van deze resultaten in een extern programma om daarmee een rapportage te maken
Postconditie	Een rapportage is gemaakt die bepaalde vragen beantwoordt.

Plan van aanpak

Doel

RDF (resource description framework) maakt het mogelijk om relaties tussen “resources” aan te brengen. Een resource kan alles zijn als het maar te identificeren is. Resources kunnen dus gezien worden als verwijzingen naar abstracties/objecten die een rol spelen in het probleemdomein van de applicatie (Universe of Discourse).

Vervolgens kunnen relaties afgeleid worden door middel van regels (inferentie). Hierdoor zou het mogelijk moeten zijn om op bestaande gegevens, zoals uit een database, extra relaties aan te brengen om vervolgens hierop inferentie toe te passen. In de rest van dit document wordt meestal alleen de afkorting RDF gebruikt terwijl daar ook vaak inferentie impliciet mee wordt bedoeld.

Oracle biedt de mogelijkheid om RDF en inferentie te combineren met hun (object) relationele database management systeem. De afstudeeropdracht zal bestaan uit het onderzoeken van de mogelijkheden van RDF. Dit is ingedeeld in de volgende deelopdrachten:

- Wat is RDF en inferentie? Hiervoor zal een literatuur studie gedaan worden. Hierdoor moet duidelijkheid verkregen worden in de principes van RDF en inferentie.
- Om de bruikbaarheid niet alleen theoretisch maar ook praktisch te toetsen zal een prototype gemaakt worden. Dit prototype zal een module voor het Kidos systeem zijn. Deze module zal een aantal extra relaties aanbrengen, d.m.v. RDF, met als doel het in kaart kunnen brengen van het sociale netwerk van een cliënt. Momenteel is het alleen mogelijk om een relatie tussen kinderen en hun biologische ouders te registreren. Uit deze nieuwe relaties zouden dan bepaalde conclusies getrokken kunnen worden met behulp van inferentie.
- Het uiteindelijke doel is om vast te stellen of RDF potentie heeft om gebruikt te worden voor projecten binnen Ordina Oracle Solutions (OOS) zoals het Kidos project.

Concluderend, in deze afstudeeropdracht word een haalbaarheidsstudie gedaan naar de bruikbaarheid van RDF. In deze haalbaarheidsstudie zullen de functionele, technische en theoretische aspecten van RDF geanalyseerd worden. Uiteindelijk zal de vraag beantwoord worden wat de toegevoegde waarde van RDF is voor projecten binnen OOS en Kidos in het bijzonder.

Kidos

De afstudeeropdracht zal zes maanden duren en daarom is het belangrijk om vast te stellen wat er wel en niet gedaan moet worden. Om de onderzoeksvraag goed te kunnen beantwoorden is belangrijk om relatief veel tijd te besteden aan het maken van een prototype.

Om de werking en mogelijkheden van RDF te begrijpen zullen de volgende deelopdrachten uitgevoerd worden:

- literatuurstudie van RDF en inferentie

- experimenteren met Oracle RDF software
- maken van een prototype
- analyseren van eisen en wensen voor het prototype
- functioneel ontwerp maken voor het prototype
- implementeren van prototype aan de hand van functioneel ontwerp
- testen van het prototype op: performance, gebruikersgemak,
- toegevoegde waarde
- schrijven van het stageverslag

Om het onderzoek goed af te bakenen worden hieronder de deelvragen gespecificeerd die als doel hebben de onderzoekvragen te beantwoorden. De vragen zijn ingedeeld door middel van het MoSCoW (M= Must have, S = Should Have, C = Could have, W = Won't have) principe.

- (M) Wat is RDF?
    - Wat zijn de (wiskundige) principes waarop RDF gebaseerd is?

Bruikbaarheid:

- (C) Zijn er toepassingen, in de praktijk, waar RDF een toegevoegde waarde kan hebben?
    - Is RDF wel de juiste technologie om het Kidos
    - project aan te vullen (een schroevendraaier als
    - hamer gebruiken).
    - Zijn er binnen OOS projecten die potentie hebben
    - om RDF technologie te kunnen gebruiken?
    - Zijn er toepassingen bekend en gebouwd
    - waarbinnen het nut van RDF zich al bewezen heeft?
    - Zijn er ook bepaalde criteria om te vast te kunnen
    - stellen of RDF wel of niet goed toepasbaar is een in
    - bepaalde context (lakmoesproef)?
- (S) Wat is de toegevoegde waarde ten opzichte van andere technologieën?
- (S) Is RDF een technologie die toepasbaar is binnen bestaande database producten, data modellen en software?
    - Met andere woorden vraagt het niet om een geheel nieuwe manier van denken, ontwerpen en implementeren van een ICT systeem als RDF gebruikt zal gaan worden?
- (M) Is de performance van RDF producten wel goed genoeg om gebruikt te kunnen worden in ICT systemen?
    - Wat beïnvloedt de performance van RDF systemen?
    - Hoe kunnen deze performance factoren gereduceerd worden?
    - Hoe zit het met performance van Oracle RDF en eventueel andere RDF producten?
- (C) Is er diepgaande wiskundige kennis nodig van de principes waarop de semantiek van RDF is gebaseerd om RDF in de praktijk te kunnen gebruiken?
    - Is deze wiskundige onderbouwing niet te ingewikkeld?
- (C) Zijn RDF producten wel ver genoeg doorontwikkeld om bruikbaar te zijn?
- (S) Is RDF wel expressief, flexibel en wel begrijpbaar genoeg om functioneel bruikbaar te zijn?
    - Is het mogelijk om complexe relaties aan te brengen?

- (S) Hoe betrouwbaar zijn de conclusies die uit de feiten (relaties) worden getrokken van RDF systemen?
    - Welke invloed heeft ruis op de conclusies?
- (S) Hoe verhoudt Oracle RDF zich ten aanzien van andere RDF producten met betrekking tot performance en functionaliteit?
- (M) Mocht RDF praktisch toepasbaar zijn wat zouden dan standaard methodieken (richtlijnen) zijn die gebruikt kunnen worden bij het gebruik van RDF?
    - Hoe moet een RDF applicatie getest worden?
    - Hoe moet een RDF applicatie ontwikkeld worden?
    - Hoe moet functionele eisen van een RDF applicatie gespecificeerd worden?
    - Welke modelleertechnieken zijn er om de structuur van RDF weer te geven (UML)?
    - Hoe moet een technisch ontwerp worden gemaakt?
- (M) Mocht RDF praktisch toepasbaar zijn wat zouden dan standaard methodieken (richtlijnen) zijn die gebruikt kunnen worden bij het gebruik van RDF?

Omdat het prototype een “proof of technology” is wordt hieronder beschreven wat voor “problemen” mogelijk opgelost kunnen worden met RDF in het Kidos project. Daarna volgt een opsomming van de deelvragen die betrekking hebben op het prototype.

Binnen het Kidos systeem worden gegevens bijgehouden over cliënten (kinderen). Momenteel is het moeilijk om relaties aan te brengen tussen een cliënt en zijn sociale context (netwerk). Dit betekend nu in de praktijk dat er alleen een relatie aangebracht kan worden tussen de cliënt en de biologische ouders. Echter het blijkt dat de sociale context niet alleen bestaat uit biologische ouders maar ook uit andere personen.

Wat er in kaart gebracht moet worden is welke personen een cliënt opvoeden en waar dit gebeurt. Er moeten dus relaties aangebracht worden over onder andere personen en woonadressen. Uit deze relaties zou het mogelijk moeten worden om conclusies te kunnen trekken. Categorisering van cliënten zou ook mogelijk zijn, dit zou kunnen gebeuren door het bestuderen van het sociale netwerk. Een voorbeeld hiervan is dat de stabiliteit van de leefomgeving bepaald kan worden. Door toepassing van RDF moeten hulpverleners het sociale netwerk van een cliënt in kaart kunnen brengen. Dit zal in de praktijk betekenen dat er relaties aangebracht kunnen worden tussen de cliënt en derden. Hierbij is het van belang dat hulpverleners zowel het netwerk kunnen beheren als raadplegen.

De relaties tussen cliënten en derden kunnen verschillend van karakter zijn, waarbij het van te voren niet bekend is wat voor type relaties er zijn en welke gegevens er bijgehouden gaan worden. Omdat RDF een ander concept is dan een relationeel database model moet er gezocht worden naar andere manieren om data te presenteren aan de gebruiker. Hieronder staan deelvragen die betrekking hebben op het prototype:

(M) Is het mogelijk om nieuwe conclusies af te leiden uit de aangebrachte relaties of uit het bestaande relationele database model?

(M) Is RDF een betere technologie om het sociale netwerk in kaart te brengen dan gebruik te maken van relationele database technologie?

(M) Hoe verloopt de interactie tussen de gebruiker en RDF, met andere woorden hoe moet de grafische gebruikersinterface (UI) de data naar de gebruik presenteren?
    - Kan dit met de bestaande technieken zoals gebruikelijk bij het weergeven van data vanuit relationele databases of moet dit anders gedaan worden

Mijlpalen

Door middel van een literatuurstudie moet er inzicht verkregen worden in de principes van RDF. Daarna wil ik de opgedane kennis toepassen bij het experimenteren met Oracle RDF. Hierdoor moet inzicht verkregen worden die gebruikt zal gaan worden bij het ontwikkelen van het prototype. Het uiteindelijke doel van het prototype is om de mogelijkheden van RDF vast te kunnen stellen in de praktijk. De volgende mijlpalen hebben betrekking op het prototype.

Naast de mijlpalen zoals hieronder zijn beschreven worden in de Gebiedsafbakening de belangrijkste deelvragen beschreven, iedere deelvraag kan gezien worden als een mijlpaal. De deelvragen zijn ingedeeld volgens het MoSCoW principe. Hierbij moet in acht genomen worden dat de deelvragen die “Must have” gekwalificeerd zijn de hoogste prioriteit hebben en dus als eerste gerealiseerd dienen te worden.

Requirements definition

De eisen voor het prototype worden gedocumenteerd in het requirements document. Deze (functionele) eisen worden in overleg met de opdrachtgever en andere belanghebbende partijen opgesteld d.m.v. interviews. Voor de functionele eisen moet worden gespecificeerd wat de testcriteria zijn, zodat achteraf vast te stellen is of de functionele eisen wel gerealiseerd zijn in het prototype.

Functioneel ontwerp

Op basis van het requirements document zal het ontwerp ingedeeld worden in modules. Tevens wordt aan de hand van de requirements voor iedere module een testplan gemaakt.

Technisch ontwerp

In het technische ontwerp wordt bekeken hoe de software (Oracle RDF) werkt en wat de mogelijkheden zijn. Daarnaast zullen tools onderzocht worden om het ontwikkelproces te ondersteunen. Indien de tijd het toelaat kunnen andere RDF producten bekeken worden. Ook moet gekeken worden wat voor infrastructuur nodig is voor Oracle RDF.

Prototype

Omdat RDF een redelijk nieuwe en onbekende technologie is stel ik voor om het project iteratief aan te pakken, dit omdat er van te voren niet precies duidelijk is wat er gedaan zal moeten worden, om uiteindelijk de onderzoeksvraag te kunnen beantwoorden.

De iteraties zullen bestaan uit de volgende stappen:

- specificeren van functionaliteit
- technisch ontwerp
- implementeren

- testen
- documenteren van bevindingen in het verslag

Na iedere iteratie zou nieuwe functionaliteit toegevoegd kunnen worden indien de tijd dit toelaat.

Uiteindelijk dient het stage verslag geschreven te worden. Dit verslag heeft de volgende voorlopige structuur.

- Inleiding
  - context
  - probleemstelling
- RDF literatuur studie
  - Inleiding
  - Toepassingen van RDF
  - Wiskundige model van RDF
  - RDF+
- Prototype
  - Requirements analyse
  - Functioneel werking
  - Technisch ontwerp
  - Testen
  - Performance
  - Problemen
- Toepasbaarheid
  - Toepasbaarheid van RDF in Kidos
  - Toepasbaarheid van RDF binnen OOS
- Conclusie

Deadlines van verslag:

- eind 3de maand eerste versie
- eind 5de maand verslag voor review bijgeleiders
- eind 6de maand uiteindelijke verslag

Beperkingen en aannamen

De volgende aannames zijn gemaakt:

1. Er kan gebruik worden gemaakt van Oracle RDF (zie middelen hieronder).
2. Het datamodel van Kidos is niet wijzigbaar
3. Er is een ontwikkelplatform beschikbaar

Middelen: om het prototype te kunnen maken en te experimenteren met Oracle RDF is er een ontwikkelomgeving nodig. Hierop is een Oracle installatie nodig die RDF ondersteunt. Daarnaast is het noodzakelijk om de beschikking te hebben het database schema van Kidos over en data om mee te werken. Om te kunnen ontwikkelen is het noodzakelijk om een ontwikkelomgeving te hebben waarin het niet erg is om dingen kapot te maken, dit kan bijvoorbeeld door een nieuwe ontwikkelomgeving in te richten met een kopie van het dataschema en data of door een account te krijgen op een bestaande ontwikkelomgeving.

Risico's

- Opstellen van requirements-, functioneel- en technisch-ontwerp kan moeilijk zijn omdat van te voren niet bekend is wat wel en niet kan met RDF. Daarnaast zijn er op dit moment nog geen requirements bekend voor het prototype.
- Waarschijnlijk is het maken van een requirements anlyse, functioneel- en technisch-ontwerp voor het prototype niet de geschikte weg om te bewandelen. Omdat dit project niet echt een standaard software ontwikkeltraject is maar een onderzoeksproject.
- De theorie achter RDF is wiskundig van aard en redelijk ingewikkeld. Te weinig kennis hiervan kan leiden tot het niet volledig kunnen benutten van RDF. Ook indien er veel tijd wordt besteed aan het begrijpen van de theorie blijft er minder tijd over voor bouwen van het prototype.
- Is er wel genoeg documentatie beschikbaar over (Oracle) RDF en inferentie.
- Onvoldoende kennis in directe omgeving over RDF. Misschien heeft Oracle Nederland wel kennis met betrekking tot het gebruik van RDF in de praktijk. Het is wellicht interessant om contact op te nemen om te kijken wat Oracle te bieden heeft.

Deze risico's zullen beperkt kunnen worden door het regelmatig overleggen met begeleiders. Hierbij zal goed gekeken moeten worden welke richting het project op gaat en zou moeten gaan om tijdig te kunnen bij sturen. Het uiteindelijke doel is natuurlijk om duidelijkheid te verschaffen in de bruikbaarheid van RDF voor OOS.

Doelstellingen

a) Projectdoelstellingen

1. Het onderzoeken van de geschiktheid van RDF.
2. Het ontwikkelen van een prototype.
3. Verhogen en overbrengen van de kennis en mogelijkheden van RDF.

b) Kritieke succesfactoren

1. Begrijpbaarheid van de theorie van RDF en inferentie.
2. Het duidelijk overbrengen van de bevindingen op de onderzoeksvragen.
3. Of het Kidos project wel mogelijkheden biedt om RDF in de juiste context te kunnen uitproberen. Met andere woorden is het wel de juiste technologie voor de problemen die er mee opgelost zouden moeten worden.

Aanpak - Projectorganisatie

In het project zijn o.a. de volgende partijen betrokken: opdrachtgever, begeleiders, ontwikkelaars (Kidos) en leverancier van RDF producten.
Jelmer is de opdrachtgever binnen Ordina. Jelmer zal uiteindelijk beoordelen of de opdracht naar behoren is uitgevoerd. Hij representeert dan ook de klant van deze opdracht.

Jurjen is de begeleider vanuit Ordina en met hem zal regelmatig de voortgang besproken worden.

Ontwikkelaars Kidos: deze groep personen is verantwoordelijke voor het

ontwikkelen van het Kidos systeem en kunnen mij helpen dit systeem te begrijpen (datamodel, code, etc). Om een werkend prototype te maken zal er ongetwijfeld hulp nodig zijn van deze groep mensen en misschien ook wel andere personen zoals architecten.

Begeleiders:

- J. Sjollema opdrachtgever, Ordina, sjollemaj@vertis.nl
- J. Jongejan eerste afstudeercoördinator, Rijks Universiteit Groningen, j.h.jongejan@rug.nl
- H. Balsters tweede afstudeercoördinator, Rijks Universiteit Groningen h.balsters@rug.nl
- J. van der Ven begeleider, Ordina Jan.van.der.Ven@ordina.nl

Appendix B Referenties

[1] Tim Berners-Lee, James Hendler and Ora Lassila, The Semantic Web, 2001 Mei, Scientific American Magazine, http://www.sciam.com/article.cfm?id=the-semantic-web
[2] Thomas B. Passin, To the semantic web, 2004 Manning Publication Co, ISBN 1-932394-20-6
[3] Grigoris Antoniou and Frank van Harmelen, A Semantic Web Primer, 2004 The MIT Press, ISBN 0-262-01210-3
[4] Christopher Walton, Agency and the semantic Web, 2007 Oxford University Press, ISBN 0-19-9292248
[5] Atanas Kiryakov, Ontologies for Knowledge Management, 2006, Semantic Web technologies, trends and research in ontology-based systems, 2007 Wiley, ISBN 978-0-470-02596-3
[7] Semantic Web Services – Approaches and Perspectives, Dumitru Roman, Jos de Bruijn, Adrian Mocan, loan Toma, HolgerLausen, Jacek Kopecky,Christoph Bussler, Dieter Fensel, John Domingue, Stefania Galizia and Lillana Cabral, Semantic Web technologies, trends and research in ontology-based systems, Wiley, 2006, ISBN 978-0-470-02596-3
[8] Applying Semantic Technology to a Digital Library, Paul Warren, Ian Thurlow, David Alsmeyer, Semantic Web technologies, trends and research in ontology-based systems, Wiley, 2006, ISBN 978-0-470-02596-3
[9] Lynch C, Garcia-Molina H. 1995. Interoperability, Scaling and the Digital Libraries Research Agenda.
[10] Opdracht Omschrijving; “Tussen de Regels door”; Ordina.
[11] Storage, querying and inferencing for semantic web languages; Jeen broekstra, ISBN 9090192360; SIKS Dissertation Series, Sep 2005.
[12] Semantic Web: A Legal Case study, Pompeu Casanovas, Núria Casellas, Joan-Josep Vallbé, Marta Poblet, V. Richard Benjamins, Mercedes Blázquez, Raúl Poblet and Jesús Contreras, Semantic Web technologies, trends and research in ontology-based systems, Wiley, 2006, ISBN 978-0-470-02596-3
[13] A Sementic Service-Oriented Architecture for the Telecommunications Industry, Alistair Duke and Marc Richardson, Semantic Web technologies, trends and research in ontology-based systems, Wiley, 2006, ISBN 978-0-470-02596-3

[14] Data Mining Technologies, Techniques, Tools, and Trends, Bhavani Thuraisingham; ISBN 0849318157; CRC Press; 1999.
[15] The Data Warehouse Toolkit, Lifecycle Toolkit; Ralph Kimball, Laura Reeves, Margy Ross and Warren Thornthwaite; ISBN 047125547-5; John Wiley & Sons; 1998.
[16] Semantic Web Research anno 2006: main stream, popular fallacies, current status and future challenges; Frank van Harmelen; 2006.
[17] Balsters, H. & Halpin, T. 2007, 'Modeling Data Federations in ORM', On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops
[18] Herman Balster, H & Halpin , T, Modeling Data Federations in ORM, presentatie dataIntegration2.ppt
[19] Halpin T. & Morgan T., March, 2008, Information Modeling and Relational Databases, Second Edition, ISBN 0123735688.
[20] Terry Halpin, 2005, 'ORM 2', On the Move to Meaningful Internet Systems 2005
[21] T. Halpin, T. 1998, 'Conceptual Queries', Database Newsletter, vol. 26, no. 2, ed. R.G. Ross, Database Research Group, Inc., Boston MA (March/April 1998).
[22] http://sourceforge.net/projects/orm/
[23] http://virtuoso.openlinksw.com/
[24] http://www-28.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html
[26] Ontology Mediation, Merging and Aligning, Jos de Bruijn,Marc Ehrig, Cristina Feier, Francisco Martín-Recuerda, François Scharffe and Moritz Weiten.
[27] http://www.sekt-project.org/project/SEKT_flyer.pdf
[28] http://proton.semanticweb.org/
[29] http://www.w3.org/Submission/sioc-related/
[30] Evaluating DILIGENT Ontology Engineering in a Legal Case Study, , Christoph Tempich, H. Sofia Pinto, York Sure, Denny Vrandeˇci´c, N´uria Casellas, and Pompeu Casanovas, http://www.aifb.uni-karlsruhe.de/WBS/dvr/publications/ivrtempich.pdf