



D2.2 RNA Mappingmethoden

versie	datum	aute ur	reden
0.1	10-05-2007	Jeen	Eerste draft: outline van globale ideeën.
0.2	07-06-2007	Jeen	Uitbreiding sectie 3; plaatjes mapping methoden; documenteer verschillende algemene mapping technieken
0.3	03-07-2007	Jeen	Uitwerking sectie 5, uitbreiding secties 2 en 3, beginnetje sectie 4

1. RNA conceptuele architectuur

1.1. Domeinmodellen

Een *domeinmodel* is een conceptueel datamodel dat als voornaamste doel representatie van kennis van een specifiek domein heeft. Met domeinmodellen worden in deze context bedoeld:

- referentiestructuren
- typologische beschrijvingen
- objectbeschrijvingen
- contextbeschrijvingen

1.2. Applicatiemodel

Een *applicatiemodel* is een declaratieve structuur (in RDF) die een mapping aanbrengt tussen 'entiteiten' die de applicatie verwacht en het daadwerkelijke domeinmodel (of een set van domeinmodellen). In tegenstelling tot een domeinmodel bevat een applicatiemodel dus *taak-specifieke kennis*.

Voorbeeld: in een determinatie-applicatie is het van belang om gegeven een concept (bijv. 'huismus') te weten welke attributen en waarden daarbij van belang zijn (naam, plaatje, typering, context-referentie, enz.) en getoond moeten worden. Deze kennis wordt uitgedrukt in het applicatiemodel.

Voor het representeren van applicatiemodellen bestaat een standaard-vocabulair genaamd Fresnel (<http://www.w3.org/2005/04/fresnel-info/>). Gebruik hiervan in RNA maakt het mogelijk om gegeven een aantal standaard applicatiemodel-entiteiten snel een declaratieve mapping op te zetten voor een willekeurig domein en op die manier de RNA toolset snel te deployen op dat domein.

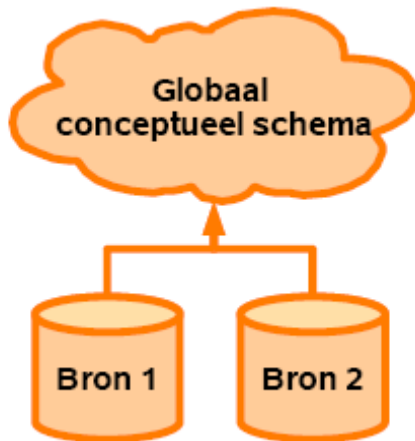


Illustration 1: directe mapping naar globaal schema

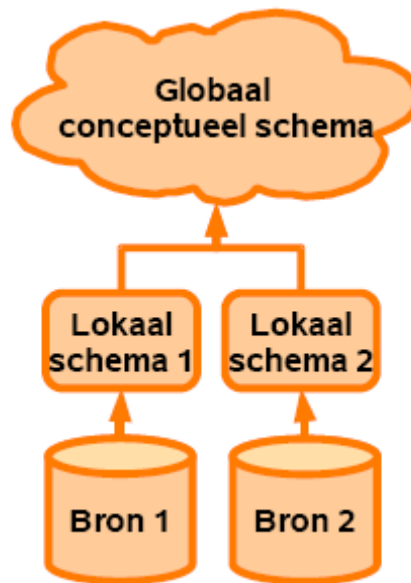


Illustration 2: lokaal schema per bron, mapping naar globaal schema

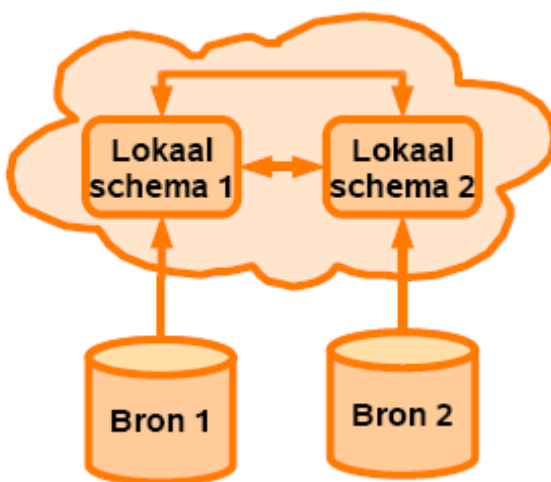


Illustration 2: lokaal schema per bron, directe onderlinge mappings

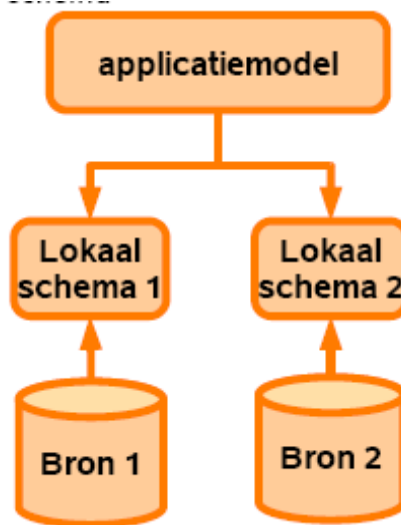


Illustration 1: lokaal schema per bron, mapping van applicatie naar schema's

2. Mapping

Een *mapping* is een definitie van een relatie tussen (twee of meer) concepten uit verschillende modellen. Deze relaties drukken uit hoe concepten zich tot elkaar verhouden en geven de mogelijkheid om objecten, gekarakteriseerd met concepten uit een bepaald model, te ontsluiten via concepten uit een ander, gemapped, model.

Er kunnen globaal gezien drie methoden onderscheiden worden voor het definiëren van mappings tussen verschillende modellen:

1. directe mapping van bron-data naar een globaal conceptueel schema (zie illustratie 1)
2. elk bronformaat heeft een eigen conceptueel schema; mapping vindt plaats door aanbrengen verbanden tussen deze schemas:
 - a) door middel van relatering aan een overkoepelend globaal schema (CIDOC-CRM) (zie illustratie 2);
 - b) door middel van directe mapping tussen concepten in verschillende schemas (zie illustratie 3)
3. elk bronformaat heeft een eigen conceptueel schema; mapping vindt plaats door definitie van applicatiemodel-concepten in termen van sets uit verschillende datasets (zie illustratie 4).

Methode 3 lijkt veel op methode 2, met als voornaamste verschil dat bij methode 2 de mapping taak-onafhankelijk is (we mappen immers 'pure' domein-modellen), terwijl bij methode 3 de mapping gerealiseerd wordt in het applicatie-model (het toepassingsgebied van de data maakt dus keuzes over welke data 'bij elkaar hoort').

2.1. Directe mapping naar globaal schema

Methode 1 (illustratie 1) vereist een globaal conceptueel schema om de bron(nen) naar te mappen. Het is belangrijk te vermelden dat dit *niet* betekent dat dit globaal schema een standaard-vocabulair dient te zijn zoals SKOS of CIDOC (hoewel dat uiteraard wel voordelen biedt), of zelfs maar dat het schema a priori beschikbaar is: er kan binnen een casus voor gekozen worden om een globaal schema te laten evolueren uit de brondata. Bij deze aanpak is het van belang vanaf het begin in het oog te houden welke informatie voor de doeleinden van de casus beschikbaar dient te zijn, en niet te 'nauw' te modelleren (dat wil zeggen, het op deze manier gedefinieerde globale schema dient breed genoeg te zijn om meerdere bronnen mee te kunnen ontsluiten).

2.2. Lokale schema's per bron

In methodes 2 en 3 is er sprake van een lokaal schema per bron. Bij deze aanpak is het in eerste instantie minder van belang dat de individuele schema's op elkaar afgestemd zijn. Echter, omdat er uiteindelijk een mapping nodig is, een integratie-stap, is het voordelig om al in een vroeg stadium enige 'synchronisatie' toe te passen, bijvoorbeeld in de granulariteit van de klassestructuur, of zelfs zaken als eenheden waarin bepaalde waarden worden uitgedrukt.

In methode 2a is er sprake van een globaal schema waarnaar gemapt wordt. Hier zijn dezelfde observaties op van toepassing als bij methode 1, met dien verstande dat door de tussenstap van de lokale modellen in te voegen de mapping conceptueel inzichtelijker is gemaakt (de mapping vindt nu plaats op 1 niveau).

2.3. Mixen van methoden

In de praktijk is het zeer goed denkbaar om een mix van alle drie hierboven beschreven methodes toe te passen. Het zal bijna altijd voor een deel van de data mogelijk zijn om, volgens methode 1, globale attributen of klassen te onderscheiden (zaken als 'name', 'identificer', bijvoorbeeld). Voor data waar zo'n directe mapping niet

wenselijk is (omdat bijvoorbeeld nuance-verschillen in betekenis verloren kunnen gaan), is methode 2 een optie. Methode 3 stelt ons in staat om voor bepaalde data die globaal gezien niet puur hetzelfde is taak-afhankelijke mappings te definiëren.

Merk op dat in het geval van een mix van methode 1 en 2a de applicatie *altijd* een globaal model kan queryen: er zal in de praktijk voor gezorgd moeten worden dat het deel van de data dat volgens methode 1 behandeld wordt direct in het globale model gerepresenteerd is. Methode 2b (binaire linking tussen lokale concepten onderling) geeft de applicatie meer vrijheid om al dan niet mappings mee 'in behandeling' te nemen, maar vergt anderszins wel dat de applicatie meer kennis heeft over hoe mappings werken.

Ons voorstel is om voor RNA projecten te standaardiseren op een combinatie van de drie methodes. In de volgende sectie gaan we dieper in op de stappen die hierbij van belang zijn.

3. RNA methodologie: hoe en wat

In de volgende secties nemen we een aantal stappen door in het proces om, gegeven een 'rauwe' dataset, tot een volledige modellering en applicatie te komen. Een centrale assumptie is dat alle modellen worden uitgedrukt in RDF[1].

3.1. Creatie van lokaal datamodel per bron

De eerste stap in onze methodologie is, voor elke te ontsluiten bron, te identificeren welke data 'opgewerkt' moet worden tot RDF en via een RDF model ontsloten dient te worden. Bij deze stap is het van belang om de doelen van de casus in de gaten te houden en er voor te zorgen dat de voor de beoogde functionaliteit noodzakelijke informatie op een voldoende rijke manier gemodelleerd wordt. Tevens is bij deze stap enige reflectie over de verschillen en overeenkomsten voor modellen van verschillende bronnen zeer nuttig, ter voorbereiding op de tweede stap.

3.2. Identificatie van een globaal domeinmodel

Gegeven een use case is een essentiële stap in een ontologiegebaseerde aanpak het identificeren en gebruiken van een toepasbaar globaal datamodel. In cultureel-erfgoed gerelateerde domeinen zijn meerdere algemeen toepasbare domeinmodellen beschikbaar, zoals CIDOC-CRM, AAT, enz. Daarnaast bestaan er volledig domein-onafhankelijke herbruikbare vocabulaires zoals Dublin Core en SKOS.

Zoals eerder opgemerkt is het niet altijd noodzakelijk dat een globaal datamodel (volledig) gemodelleerd is aan de hand van een standaard-vocabulair. We kunnen ervoor kiezen om, gegeven de lokale modellen gedefinieerd in de eerste stap, bepaalde properties en classes te abstraheren (naar een class of property in een globaal vocabulair) en andere rechtstreeks over te nemen. Via mappings kunnen we er tevens voor kiezen om abstracties zodanig te modelleren dat zowel de specifieke class (in het lokale model) als de 'globale' class (in het globale vocabulair) bewaard blijven.

3.3. Aanbrengen van mappings tussen datamodellen

Er zijn verschillende technieken om mappings tussen modellen te bewerkstelligen. In deze sectie bespreken we een aantal van deze mogelijkheden kort.

3.3.1. Schema Mapping door subsumptie

Mapping door subsumptie is een techniek waarbij concepten uit verschillende modellen gerelateerd worden door middel van subsumptie- of equivalentie-relaties. Mogelijke relaties binnen RDF en OWL hiervoor zijn:

- `X rdf:type Y`
deze relatie drukt uit dat X een instantie is van de klasse Y

Voorbeeld: `MonaLisa rdf:type Painting .`
- `X rdfs:subClassOf Y`
deze relatie drukt uit dat de klasse X een subklasse is van de klasse Y

Voorbeeld: `Painting rdfs:subClassOf Artwork .`
- `X rdfs:subPropertyOf Y`
deze relatie drukt uit dat X een sub-property (een specifiekere property) is van de property Y.

Voorbeeld: `paints rdfs:subPropertyOf creates .`
- `X owl:equivalentClass Y`
deze OWL relatie drukt uit dat de klassen X en Y dezelfde instanties hebben (extensionele gelijkheid).
- `X owl:sameAs Y`
deze OWL relatie drukt uit dat X en Y hetzelfde concept (intensionele gelijkheid) uitdrukken.

Mapping door subsumptie is met name gebruikelijk in het mappen van een domein-specifiek schema naar een globaal schema. Een veel in de praktijk voorkomende mapping is tussen de globale concepten in Dublin Core en specifiekere versies van die concepten in een domeinspecifiek model, bijvoorbeeld:

```
ex:socialSecurityNumber rdfs:subPropertyOf dc:identifier .
```

In deze mapping wordt uitgedrukt dat de (domein-specifieke) relatie `socialSecurityNumber` een specifiekere vorm van de Dublin Core `identifier`-relatie is.

Deze mapping methode biedt (naast eenvoud) als voordeel dat zowel domein-specifieke kennis als de meer globale, interoperabele kennis bewaard blijft: een andere applicatie die niets van social security numbers weet kan door de mapping naar Dublin Core wel interpreteren dat het hier om een identifier gaat. Een nadeel is de beperktheid: door deze methode zijn uitsluitend binaire mappings (tussen twee concepten) uitdrukbaar.

3.3.2. Schema Mapping door query/transformatie

De Semantic Web querytalen SeRQL[2] en SPARQL[3] introduceren beide functionaliteit voor het transformeren van RDF-modellen: door middel van zgn. 'CONSTRUCT'-queries kan informatie uit een datamodel opgehaald worden en direct worden gemapt naar een ander datamodel.

[TODO verdere uitwerking met voorbeelden]

3.4. Creatie van een applicatiemodel

Creatie van het schema van het applicatiemodel is in termen van de RNA toolset een eenmalige aangelegenheid. Immers, het navigatieparadigma en de gebruikte tools liggen vast, en hierin spelen entities en facets een hoofdrol. Onafhankelijk van de specifieke dataset kunnen we dus bedenken in welke termen de applicatie 'denkt' en hoe die met elkaar samenhangen.

Per use-case zal echter een koppeling moeten worden aangebracht tussen de termen in het applicatiemodel en de concepten in het domeinmodel. In deze koppeling komt tot uitdrukking welke data op welke manier gepresenteerd moet worden door de applicatie (bijv. Welke facetten er zijn en welke properties van entiteiten wel of niet getoond moeten worden).

Zoals eerder genoemd lijkt het Fresnel vocabulair bij uitstek geschikt om het applicatiemodel in uit te drukken. Gegeven een applicatiemodel-schema met de volgende entry:

```
amrna:entityLens a fresnel:Lens;  
                  fresnel:purpose amrna:EntityRendering .
```

Zou een voorbeeld van koppeling in de use case van de Groningendataset zijn:

```
amrna:entityLens fresnel:classLensDomain groningen:Gebouw ;  
                  fresnel:showProperties (  
                      "ccrm:P1F.is_identified_by[ccrm:E42s.  
id]/rdf:value"^^fresnel:fslselector ) .
```

In deze koppeling wordt uitgedrukt dat entities in de Groningen dataset instanties van de klasse 'groningen:Gebouw' zijn, en dat de waarde van de 'is_identified_by' property getoond moet worden als die waarde van het type E42s.id is.

[NOTE bovenstaand voorbeeld klopt waarschijnlijk op details niet helemaal maar geeft een globale indruk. Er zou eigenlijk het een en ander aan de modellering van de Groningen dataset verbeterd moeten worden, met name in de typering (via rdf:type) van objecten]

Een interessante observatie in dit voorbeeld is dat de FSL expressie volledig in termen van het CIDOC model is, en niet zozeer in Groningen-specifieke terminologie. Dit betekent dat het wellicht mogelijk is om Fresnel-koppelingen voor verschillende use-cases te hergebruiken, mits beide use cases van het CIDOC model gebruik maken. Er kan zo een basis-applicatiemodel ontwikkeld worden dat op elke op CIDOC gebaseerde dataset toepasbaar is en dat alleen in de usecase-specifieke details getuned moet

worden (bijvoorbeeld door extra 'showProperties' entries toe te voegen, of door de 'classLensDomain' aan te passen).

[TODO verdere uitwerking van gebruik Fresnel in RNA toolset, ism Herko]

4. SKOS, RDF en OWL

Simple Knowledge Organisation Systems (SKOS) [4] is een activiteit binnen het World Wide Consortium die zich bezig houdt met het bouwen van standaarden voor het ontsluiten van taxonomieën, thesauri, enz. op het Web. SKOS Core [5][6] is een definitie van een vocabulair, gemodelleerd in RDF, dat dient voor het uitdrukken van de structuur binnen dergelijke thesauri, taxonomieën, en andere vormen van controlled vocabularies.

Concreet biedt SKOS Core modelleerprimitieven die dienen om termen, beschrijvingen en concepten gestructureerd te modelleren en relateren. Zo biedt het relaties als `skos:narrower` en `skos:broader`, die uitdrukken dat twee concepten meer/minder specifiek qua betekenis zijn. Daarnaast biedt het functionaliteit voor het geven van labels aan concepten en daar een preferentie in aan te brengen (d.m.v. Properties als `skos:prefLabel` en `skos:altLabel`).

In het volgende voorbeeld demonstreren we het basis-idee achter SKOS Core door twee concepten en de relatie ertussen te modelleren (bron: [5]):

```
ex:concept1 skos:prefLabel "Tourism" ;
  skos:broader ex:concept2 .
```

```
ex:concept2 skos:prefLabel "Arts, recreation and travel" .
```

5. RDF Namespaces en Contexts

In de voorgaande secties hebben we besproken hoe conceptueel een domeinmodellering gesegmenteerd kan worden, waarbij er – afhankelijk van de gekozen modelleertechniek – sprake is van een aantal van de volgende segmenten:

- 1 of meer lokale datamodellen;
- 1 globaal datamodel;
- 1 applicatiemodel

In de praktijk zal de modellering gestalte krijgen in een of meer RDF datasets die zich in een of meer Sesame repositories bevinden. In deze sectie gaan we in op twee mechanismen die gebruikt kunnen worden om data binnen een RDF graaf te (onder)scheiden.

5.1. Namespaces

Namespaces zijn een XML syntax mechanisme[7]. Het doel van namespaces is een methode te bieden om XML element- en attribuutnamen te kwalificeren door associatie met een namespace URI. Aan de URI zelf worden geen eisen gesteld: het gaat hier puur om een identificatie-mechanisme, dus er is bijvoorbeeld geen eis dat de namespace-URI naar een daadwerkelijk bestaande pagina of definitie verwijst. Gebruikelijk is dat de namespace URI in ieder geval verwijst naar het domein van de organisatie die het betreffende document beheert.

In RDF, OWL en query-mechanismen zoals SeRQL en SPARQL worden XML namespaces (in RDF/XML[8]) en vergelijkbare mechanismen (in andere syntaxen zoals Turtle[9]) veel gebruikt als 'afkortingsmechanisme': de volledige identifier (URI) van, bijvoorbeeld, een RDF property of een RDFS class hoeft niet elke keer volledig uitgeschreven te worden, in plaats daarvan definiëren we een namespace (bestaande uit een korte prefix en de volledige URI van de namespace). Deze kan dan vervolgens gebruikt worden door de property/class te identificeren met zijn zgn. *QName* (qualified name).

Voorbeeld: een RDF statement (in Turtle syntax), zonder gebruik van namespaces en QNames:

```
<http://example.org/concept1>  
<http://www.w3.org/2004/02/skos/core#prefLabel> "Tourism".
```

Hetzelfde statement nogmaals, dit keer met namespaces en QNames:

```
@prefix ex: <http://example.org/> .  
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .  
  
ex:concept1 skos:prefLabel "Tourism".
```

Hoewel namespaces conceptueel gezien geen betekenis hebben (ze zijn immers puur een syntax-mechanisme), is de keuze van de juiste namespace identifiers voor modellen een zaak van belang. Voor standaard-vocabulaires, zoals bijvoorbeeld SKOS, bestaat meestal een voorgedefinieerde 'standaard' namespace. Correct hergebruik van deze voorgedefinieerde namespaces is cruciaal voor interoperabiliteit: immers, door het feit dat we bepaalde algemeen aanvaarde identifiers gebruiken kunnen we garanderen dat een andere partij die onze data leest deze juist kan interpreteren.

Voor delen van het model die niet uit standaard-vocabulaires komen (onze eigen properties, instanties en classes, zoals `ex:concept1` in het voorbeeld hierboven) is het van belang om een namespace te introduceren.

Naast het interoperabiliteits-aspect is er nog een zeer pragmatische reden om (handige) namespace-definities te gebruiken: ze verhogen de leesbaarheid van in RDF opgeschreven informatie. Hierdoor wordt niet alleen begrip van een modellering makkelijker gemaakt maar ook worden queries over het model simpeler te lezen en schrijven, aangezien dezelfde namespace-definities ook in queries gebruikt kunnen worden. Om dit te illustreren geven we een voorbeeld van een SeRQL query die, gegeven een concept (`concept1`) een via de 'broader' relatie gerelateerd concept en het bijbehorende label ophaalt. Eerst zonder namespace-gebruik:

```
SELECT
```

```

    concept, label
FROM
    {concept}
<http://www.w3.org/2004/02/skos/core#prefLabel> {label};
    <http://www.w3.org/2004/02/skos/core#broader>
{<http://example.org/concept1>}

```

Dezelfde query, dit keer inclusief namespace-definities:

```

SELECT
    concept, label
FROM
    {concept} skos:prefLabel {label};
    skos:broader {ex:concept1}
USING NAMESPACE
    skos = <http://www.w3.org/2004/02/skos/core#>,
    ex = <http://example.org/>

```

Zoals dit voorbeeld aantoont wordt gestructureerd en overzichtelijk werken met RDF in aanzienlijke mate gefaciliteerd door gebruik van namespaces.

Zoals eerder gezegd zijn namespaces geen conceptueel of semantisch maar een syntactisch mechanisme. Het definiëren van namespaces moet dan ook uitgaan van praktisch nut in syntax: het moet dienen tot heldere, gestructureerde kwalificatie van properties, instanties en classes, en in het algemeen de leesbaarheid van RDF modellen en over deze modellen gedefinieerde queries verhogen.

Zoals het voorbeeld hierboven illustreert is een veel voorkomende aanpak om per vocabulair, of per domeinmodel, een namespace te definiëren. Deze aanpak heeft als voordeel dat entiteiten die conceptueel bij elkaar horen (namelijk properties en classes uit hetzelfde model) ook dezelfde namespace-prefix gebruiken en dus ook syntactisch herkenbaar bij elkaar horen (zie bijvoorbeeld het gebruik van de twee SKOS properties in de SeRQL query hierboven).

In het RNA project is er gegeven een casus vaak sprake van meerdere domeinmodellen, zoals we in sectie 1 hebben gezien. In het modelleren van deze domeinmodellen in RDF is het van groot belang om namespace-definities correct en overzichtelijk te houden, en er voor te waken om niet in de verleiding te komen om voor elke fysieke RDF file een aparte namespace te introduceren. Beter is om te proberen namespaces zodanig te kiezen en te definiëren dat classes en properties die conceptueel bij elkaar gegroepeerd horen in dezelfde namespace te definiëren. Merk op dat er geen relatie is tussen de namespace-URI en de fysieke file: het is zeker niet onmogelijk (en in feit zelfs heel gebruikelijk) om dezelfde namespace-definitie voor de concepten in meerdere fysieke files te gebruiken.

5.2. Context

Context is een conceptueel mechanisme dat tot doel heeft een kwalificatie van subsets van een grote kennis-verzameling aan te brengen. In de betekenis waarin wij het hier gebruiken gaat het specifiek over het RDF context-mechanisme zoals dat in het Sesame framework (release 2) wordt aangeboden. Dit mechanisme maakt het mogelijk binnen

een RDF repository (een database gevuld met een of meer RDF modellen) subsets te onderscheiden, voor administratieve en/of conceptuele doelen. Dit gebeurt praktisch door elk RDF statement in een bepaalde context te annoteren met een *context identifier*, meestal een URI. Concreet zijn RDF statements dus niet langer triples (subject, predicate, object), maar *quads* (subject, predicate, object, context).

Dit mechanisme kan, zoals gezegd worden, gebruikt worden voor meerdere doeleinden. Als puur administratieve functionaliteit biedt het bijvoorbeeld de mogelijkheid om voor alle statements bij te houden wat hun fysieke bron-document is, en op basis van die informatie efficiënte updates te kunnen doen. Daarnaast zijn allerlei andere partitionering die in een dataset van belang kunnen zijn mogelijk: zo kan context gebruikt worden om doelgroepen te identificeren ('deze statements zijn voor kinderen, deze voor wetenschappers, deze alleen voor de redacteur').

[TODO uitwerking van gebruik namespaces en context in concreet voorbeeld]

Referenties

- 1: Graham Klyne and Jeremy Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, Recommendation, 2004. <http://www.w3.org/TR/rdf-concepts/>
- 2: Jeen Broekstra, SeRQL: A Second Generation RDF Query Language. Chapter 4 in Storage, Querying and Inferencing for Semantic Web Languages. ISBN 90-9019-236-0. 2005
- 3: Andy Seaborne and Eric Prud'hommeaux, SPARQL Query Language for RDF. World Wide Web Consortium, Candidate Recommendation, 2007. <http://www.w3.org/TR/rdf-sparql-query/>
- 4: , Simple Knowledge Organisation Systems. World Wide Web Consortium, . . <http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/>
- 5: Alistair Miles and Dan Brickley, SKOS Core Guide. World Wide Web Consortium, Working Draft, 2005. <http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/>
- 6: Alistair Miles and Dan Brickley, SKOS Core Vocabulary Specification. World Wide Web Consortium, Working Draft, 2005. <http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/>
- 7: Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, Namespaces in XML 1.0 (Second Edition). World Wide Web Consortium, Recommendation, 2006. <http://www.w3.org/TR/REC-xml-names/>
- 8: Dave Beckett, RDF/XML Syntax Specification (Revised). World Wide Web Consortium, Recommendation, 2004. <http://www.w3.org/TR/rdf-syntax-grammar/>
- 9: Dave Beckett, Turtle - Terse RDF Triple Language. ILRT Bristol, Draft, 2006. <http://www.dajobe.org/2004/01/turtle/>