Mogelijkheid om intelligente operaties te definiërenl

StUF: inperking en uitbreiding op SQL

Henri Korver en Maarten van den Broek

StUF is een standaard die onder meer web-services definieert voor gegevensuitwisseling tussen systemen. Begin 2009 is StUF door het College Standaardisatie geplaatst op de lijst van open standaarden en is daarmee een officiële standaard voor gemeenten en het landelijk stelsel van basisregistraties.

Uitwisselingsstandaarden zijn vaak complex en lastig te begrijpen. Er is veel technische documentatie en je begrijpt de standaard pas echt als je er mee werkt. Voor StUF geldt dit ook. Toch is er de behoefte om de gedachte erachter te begrijpen zonder al te veel inspanning. Een aardige ingang hiervoor biedt de vergelijking met de CRUD-operaties (Create, Read, Update en Delete) toegepast in databases.
StUF gebruikt een CRUD-variant die veel overeenkomsten met SQL vertoont maar ook enkele belangrijke verschillen:

- StUF ondersteunt synchrone én asynchrone CRUD-operaties;
- Het CRUD-model van StUF is gebaseerd op hiërarchische XML-structuren in plaats van tabellen;
- SQL is als taal veel expressiever dan StUF.

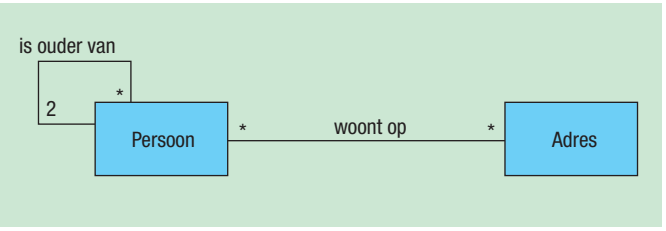
In dit artikel zullen we verder ingaan op deze verschillen.

Tabellen versus boomstructuren

Een database wordt veelal ontworpen op basis van een semantisch model. Bij StUF start het ontwerpen van berichten ook met een semantisch model. Afbeelding 1 toont een eenvoudig model in de vorm van een UML-klassendiagram. Het diagram definieert twee relaties: een relatie tussen personen en een relatie tussen personen en adressen. Een persoon heeft twee ouders en kan zelf kinderen hebben. Bovendien kan een persoon op meerdere adressen tegelijk wonen. En omgekeerd: een adres kan meerdere inwoners hebben.
Afbeelding 2 toont de vertaling van dit model in een database. Er zijn technische sleutels (ID) toegevoegd om de records te identificeren en om de relaties tussen de records te kunnen reconstrueren via het matchen op identifiers met behulp van JOIN's. De many-to-many relatie (woont op) is vertaald naar de hulptabel PersoonAdres. De relatie 'is ouder van' is vertaald naar de foreign keys vaderID en moederID.
Afbeelding 3 vertaalt hetzelfde semantische model naar twee boomstructuren die als basis dienen voor StUF-berichten. Het startpunt is het bovenste meest linker blokje; de eerste boom betreft personen en de tweede betreft adressen. De ongerichte associaties in het semantisch model zijn omgezet in gerichte associaties aangeduid door blokpijlen. In afbeelding 1 zitten twee loops. De eerste loop is de relatie 'is ouder van', een relatie van de klasse Persoon naar zich zelf en de tweede loop is de bidirectionele relatie tussen de klassen Persoon en Adres. Door deze loops zijn oneindig veel boomstructuren mogelijk. Het is de taak van de berichtontwerper om juist de boomstructuren relevant voor een bepaalde toepassing te definiëren. In de eerste boom van afbeelding 3 vond de berichtontwerper het blijkbaar niet nodig om de adresgegevens van de ouders op te nemen.
StUF eist dat de boomstructuren acyclisch zijn, ofwel geen oneindige paden bevatten. Het heeft immers weinig zin om een bericht van een oneindige omvang te versturen. StUF eist ook dat de bomen geordend zijn. De vertakking van boven naar beneden heeft altijd dezelfde volgorde. Een vaste structuur vereenvoudigt de verwerking van berichten. De berichtontwerper maakt keuzes door loops op een bepaald punt af te breken of door minder relaties of attributen in de boomstructuren op te nemen dan in het semantisch model voorhanden zijn.
Dit verscherpt de semantiek voor een bepaald toepassingsgebied: de boomstructuren in het berichtontwerp zijn de objecten die relevant zijn voor de business. Zij definiëren welke informatie in één keer kan worden opgevraagd en in de user interface getoond. In de boomstructuren zijn technische details als sleutels, hulptabellen en foreign keys niet relevant. De StUF standaard baseert zich daarmee op een model dat dichter bij het semantische model ligt dan SQL.

Interactiepatronen

StUF ondersteunt de CRUD operaties van SQL met synchrone kennisgeving- en vraagberichten. StUF biedt daarnaast de moge-



Afbeelding 1: Semantisch model.

lijkheid om systemen losjes te koppelen door een asynchroon interactiepatroon. De zender van het bericht zegt dan tegen de ontvanger "*ik* heb wat veranderd en via dit bericht geef ik je inzage" in plaats van "*jij* moet iets veranderen". De zender kan aangeven of de ontvanger de asynchrone kennisgeving moet overnemen of niet. Bij SQL heeft CUD[1] betrekking op een toestandsverandering van de server en bij StUF van de client. Een StUF-bericht bevat veelal een mutatie die reeds is doorgevoerd bij de verzender, een gebruikelijke manier van werken in publish/subscribe- en event-driven architecturen.
Het vraag- en antwoordbericht van StUF lijkt sterk op de Read-operatie van SQL (SELECT): de query wordt uitgevoerd op de server. StUF ondersteunt zowel een synchrone als een asynchrone interactiepatroon.
In afbeelding 4 bevindt SQL zich op de lijntjes die softwarecomponenten (kleine blokjes) verbinden met de database. SQL wordt normaliter toegepast binnen de grenzen van een applicatie (interne communicatie). De pijltjes symboliseren het StUF-berichtenverkeer via webservices en een dienstenbus. Door middel van StUF kunnen verschillende applicaties met elkaar communiceren (externe communicatie). StUF ontsluit een database met webservices op basis van business objecten gemodelleerd als boomstructuren. Deze business objecten zijn implementatie-onafhankelijk. Via StUF kunnen systemen met heel verschillende implementaties van een semantisch model in een database eenvoudig met elkaar communiceren.

Kennisgevingberichten

Deze paragraaf gaat dieper in op de semantiek voor kennisgevingberichten met wijzigingen in business objecten. Binnen kennisgevingberichten speelt het begrip *verwerkingssoort* een belangrijke rol. Hiermee wordt per entiteit (object of relatie, de blokjes c.q. blokpijlen in afbeelding 3) in een business object de CUD-mutatie bij de verzender aangegeven. StUF onderscheidt de volgende verwerkingssoorten:

T - Een entiteit is toegevoegd;
W - De attributen van een entiteit zijn gewijzigd;
V - Een entiteit is verwijderd;
E - Een relatie is beëindigd;
I - Er is niets gebeurd met de entiteit, deze bevat alleen identificerende gegevens;
R - Een relatie wordt vervangen door een andere relatie;
O - Twee objecten die naar hetzelfde ding in de werkelijkheid verwijzen zijn ontdubbeld;
S - De technische sleutel van een entiteit is gewijzigd.

De verwerkingssoorten T, W en V komen overeen met de CUD-operaties van SQL. Omdat een business object een complexere structuur heeft dan een tabel in SQL, kent StUF voor relaties twee extra verwerkingssoorten: beëindiging ('E') en vervanging ('R') van een relatie. Ze zijn relevant voor systemen die met historie werken. Verwerkingssoort 'E' is bijvoorbeeld van toepassing bij een echtscheiding die een relatie tussen twee personen beëindigt. Een beëindigde relatie kan nog steeds relevant zijn voor een registratie. Het verwijderen van een relatie (verwerkingssoort 'V') geeft aan dat deze niet meer relevant is voor de verzender. Bij verwerkingssoort 'R' wordt een relatie beëindigd en direct vervangen. Een verhuizing vervangt bijvoorbeeld de relatie naar het bestaande adres door een nieuw adres. Wat verwerkingssoort 'W' is voor attributen is 'R' voor relaties.
StUF kent een speciale verwerkingssoort 'S' voor een wijziging in de technische sleutel bij de zender en 'O' voor het ontdubbelen van een object bij de zender, dat wil zeggen het samenvoegen van twee records die verwijzen naar hetzelfde object in de werkelijkheid tot één record. Hoe een sleutelwijziging of ontdubbeling moet worden doorgevoerd bij de ontvanger is sterk afhankelijk van zijn implementatie van het semantisch model. Dit illustreert opnieuw dat StUF semantisch op een ander niveau werkt dan SQL.
De boomstructuren zoals in afbeelding 3 bevatten meerdere objecten en relaties en kunnen dus meerdere verwerkingssoorten bevatten. Als je alleen de attributen van een relatie wilt wijzigen dan wordt het startobject slechts gebruikt ter identificatie van de te wijzigen relatie en krijgt het startobject de verwerkingssoort 'I'. Ter illustratie volgen enkele voorbeelden.

Toevoeging van een object

Onderstaand XML-fragment is een vereenvoudigd voorbeeld van een asynchroon kennisgevingbericht (gekenmerkt door berichtcode Lk01) over de toevoeging van een persoon.

```
<ns:stufbericht ...>
<ns:stuurgegevens>
<StUF:berichtcode>Lk01</StUF:berichtcode>
<StUF:zender>A</StUF:zender>
<StUF:ontvanger>B</StUF:ontvanger>
...
<StUF:entiteittype>Persoon</StUF:entiteittype>
</ns:stuurgegevens>
<ns:parameters>
<StUF:mutatiesoort>T</StUF:mutatiesoort>
<StUF:indicatorOvername>I</StUF:indicatorOvername>
</ns:parameters>
<ns:object StUF:verwerkingssoort="T"
                        StUF:entiteittype="Persoon">
<ns:bsn>123456789</ns:bsn>
<ns:voornaam>Tof</ns:bsn>
<ns:voorvoegsel xsi:nil="true"
                         StUF:noValue="geenWaarde"/>
<ns:achternaam>Thissen</ns:achternaam>
<ns:beroep>Voorzitter Divosa</ns:beroep>
</ns:object>
</ns:stufbericht>
```

Een persoon (entiteittype 'Persoon') is toegevoegd (mutatiesoort 'T') in systeem A. Het bericht is informatief (indicatorOvername

'I') en hoeft niet verwerkt te worden door systeem B. Dit bericht kan verwerkt worden met onderstaand SQL-statement, als we ervan uitgaan dat de sleutel persoonID automatisch wordt gegenereerd:

```
INSERT INTO Persoon
(bsn, voornaam, tussenvoegsels, achternaam, beroep)
VALUES
('123456789', 'Tof', NULL, 'Thissen', 'Voorzitter
                                        Divosa')
```

Het belangrijkste verschil is het interactiepatroon: het SQL statement is synchroon en dient verplicht verwerkt te worden in de database. Bij het voorvoegsel zien we nog een verschil: in het StUF-bericht wordt expliciet aangegeven dat dit leeg is (StUF:noValue="geenWaarde"). In het SQL statement wordt de waarde op NULL gezet, wat ook kan betekenen dat het voorvoegsel onbekend is.

Wijziging object

Het volgende XML fragment betreft een wijziging van een persoon (verwerkingssoort 'W'). De persoon met bsn-nummer 123456789 is van beroep veranderd.

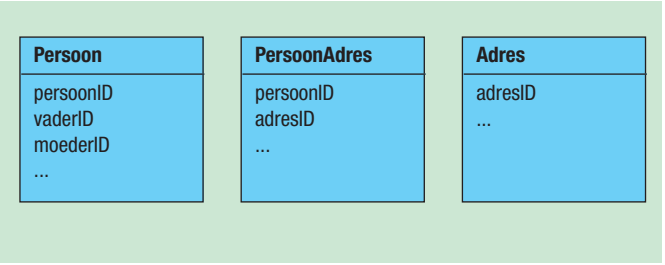
```
<ns:object StUF:verwerkingssoort="W"
                        StUF:entiteittype="Persoon">
<ns:bsn>123456789</ns:bsn>
<ns:beroep>Voorzitter Divosa</ns:beroep>
</ns:object>
<ns:object StUF:verwerkingssoort="W"
                        StUF:entiteittype="Persoon">
<ns:bsn>123456789</ns:bsn>
<ns:beroep>Directeur Kwaliteitsinstituut
                    Nederlandse Gemeenten</ns:beroep>
</ns:object>
```

In StUF worden wijzigingen doorgegeven door middel van een was/wordt constructie, omdat de verzender van een bericht niet weet hoe een ontvanger een object identificeert. Het element <ns:beroep> bevat een oude en nieuwe waarde. Het element <ns:bsn> is niet veranderd en wordt gebruikt voor de identificatie van het te wijzigen object. Onderstaand SQL-statement implementeert dit bericht:

```
UPDATE Persoon
SET beroep = 'Directeur Kwaliteitsinstituut
                                Nederlandse Gemeenten'
WHERE bsn = '123456789'
```

Vervanging van een relatie

Het volgende fragment betreft de verhuizing van een persoon. In StUF termen is dit een vervanging (verwerkingssoort 'R') van een relatie van het type woontOp.



Afbeelding 2: Afbeelding in een database.

```
<ns:object StUF:verwerkingssoort="I"
                        StUF:entiteittype="Persoon">
<ns:bsn>123456789</ns:bsn>
<ns:relatie StUF:verwerkingssoort="R"
                        StUF:entiteittype="woontOp">
<ns:gerelateerde StUF:verwerkingssoort="I"
                          StUF:entiteittype="Adres">
<ns:postcode>1058BK</ns:postcode>
<ns:huisnummer>27</ns:huisnummer>
<ns:toevoeging>huis</ns:toevoeging>
</ns:gerelateerde>
<ns:beginDatum>20010412</ns:beginDatum>
<ns:eindDatum>20070101</ns:eindDatum>
</ns:relatie>
</ns:object>
<ns:object StUF:verwerkingssoort="I"
                        StUF:entiteittype="Persoon">
<ns:bsn>123456789</ns:bsn>
<ns:relatie StUF:verwerkingssoort="R"
                        StUF:entiteittype="woontOp">
<ns:gerelateerde StUF:verwerkingssoort="I"
                          StUF:entiteittype="Adres">
<ns:postcode>1011AA</ns:postcode>
<ns:huisnummer>210</ns:huisnummer>
<ns:toevoeging>eerste etage</ns:toevoeging>
</ns:gerelateerde>
<ns:beginDatum>20070101</ns:beginDatum>
<ns:eindDatum xsi:nil="true"
                         StUF:noValue="geenWaarde"/>
</ns:relatie>
</ns:object>
```

De objecten waarnaar de relatie verwijst wijzigen niet en worden alleen gebruikt om de relatie te identificeren. Daarom hebben de objecten persoon en adres verwerkingssoort 'I' (Identificatie). De verzender verwacht dat het adresobject waarnaar de nieuwe relatie wordt gelegd, reeds een bekend object is voor de ontvanger van het bericht. Was dat niet zo en betrof het een nieuw object, dan had het adresobject de verwerkingssoort 'T' (Toevoeging) gekregen.

Hierna volgt het SQL-statement dat bovenstaand StUF-bericht implementeert, één van de vele mogelijke implementaties.

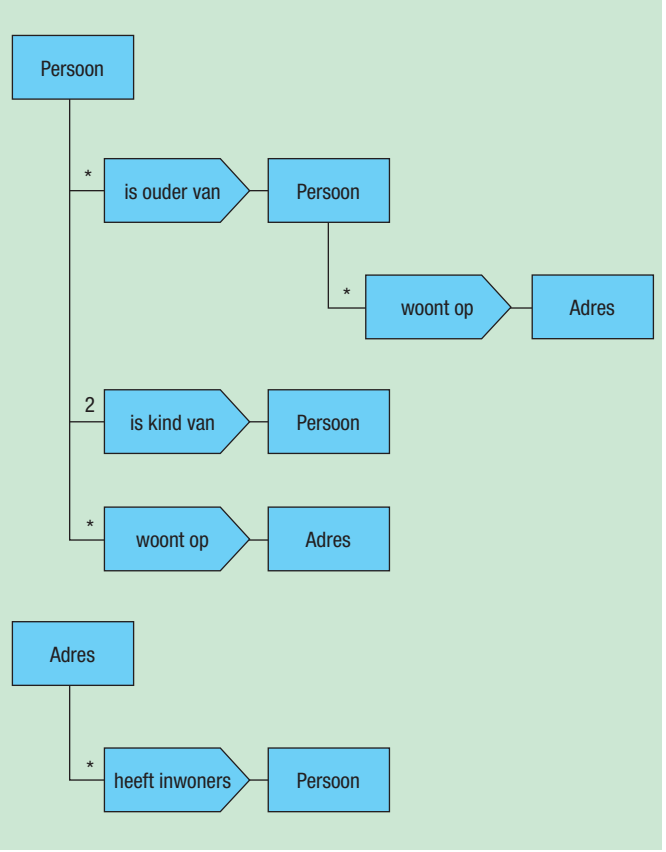
```
UPDATE PersoonAdres
SET
adres_id = (select adres_id from Adres where
                postcode="1011AA" and huisnummer=210
                      and toevoeging="eerste etage")
, beginDatum = 01-01-2007
WHERE
persoon_id = (select persoon_id from Persoon where
                                    bsn="123456789")
and adres_id = (select adres_id from Adres where
                 postcode="1058BK" and huisnummer=27
                                and toevoeging="huis")
```

Dit UPDATE-statement is in tegenstelling tot het StUF-bericht implementatiespecifiek. Het SQL-statement bevat nu ook technische sleutels. Het was-object in het StUF-bericht is in dit voorbeeld noodzakelijk om het te wijzigen record te identificeren via het persoon-object en het te vervangen adres. De einddatum wordt in het SQL statement niet gezet, omdat deze implementatie ervan uitgaat dat in een record alleen de begindatum wordt vastgelegd. In een database waarin historie wordt bijgehouden zal vaak ook een einddatum in het record aanwezig zijn om historische relaties te kunnen onderscheiden van actuele relaties. We zien hier opnieuw dat StUF op een hoger abstractieniveau leeft dan SQL.

Vraag- en antwoordberichten

Nu gaan we dieper in op de R-operatie (Read). Voor de CUD-operaties hebben we gezien dat StUF in een aantal opzichten semantisch rijker is dan SQL. Bij de R-operatie is StUF op een aantal punten bewust minder expressief dan SQL.
Vraagberichten specificeren de gevraagde elementen (de SELECT-clause) en de gevraagde objecten (de WHERE-clause) en de volgorde (de 'ORDER BY' clause). StUF perkt bewust de functionaliteit voor sorteren en selecteren in. De sorteringen zijn voorgedefinieerd en selecties op meer dan één criterium zijn altijd via AND aan elkaar gekoppeld. OR in de selectiecriteria wordt niet ondersteund. Zo nodig dienen meerdere vragen gesteld te worden en het resultaat samengevoegd. Deze beperkingen garanderen bij een goede implementatie dat binnen redelijke tijd elke vraag beantwoord kan worden.
In de SELECT-clause gebruikt StUF de boomstructuren uit afbeelding 3. De structuur van het business object ligt vast in de koppelvlakspecificatie. Bij de verwerking van een SQL-statement zijn willekeurig complexe selecties mogelijk via (low-level) JOIN-expressies om een complex business object samen te stellen.

Ook het antwoord wordt gegeven in de vorm van de boomstructuren. Ook dit is een verschil met SQL, waar de respons altijd bestaat uit *N* kolommen per voorkomen in de set. Het antwoord in een StUF-bericht heeft meer structuur en kan gemakkelijker vertaald worden naar een user interface.



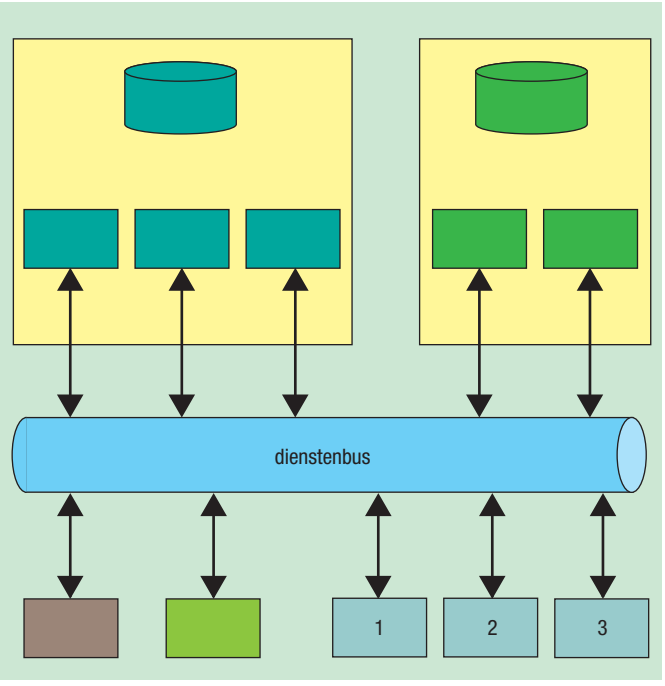
Afbeelding 3: Boomstructuren voor berichtuitwisseling.

Onderstaand voorbeeld is een synchroon vraagbericht (berichtcode Lv01) over personen. Voor Amsterdamse vrouwen tussen de 30 en 40 jaar worden alle persoonsgegevens opgevraagd.

```
<ns:npsLv01 ...>
<ns:stuurgegevens>
<StUF:berichtcode>Lv01</StUF:berichtcode>
<StUF:entiteittype>Persoon</StUF:entiteittype>
</ns:stuurgegevens>
<ns:parameters>
<StUF:sortering>1</StUF:sortering>
<StUF:indicatorVervolgvraag>false
                          </StUF:indicatorVervolgvraag>
</ns:parameters>
<ns:gelijk StUF:entiteittype="Persoon">
<ns:gemeente>Amsterdam</ns:gemeente>
<ns:geslacht>V</ns:geslacht>
</ns:gelijk>
<ns:vanaf StUF:entiteittype="Persoon">
<ns:geboortedatum>19700101</ns:geboortedatum>
</ns:vanaf>
<ns:totEnMet StUF:entiteittype="Persoon">
<ns:geboortedatum>19800101</ns:geboortedatum>
</ns:totEnMet>
<ns:scope StUF:scope="alles">
</ns:npsLv01>
```

Onderstaand voorbeeld laat één object zien uit het antwoordbericht. Het object van het entiteittype Persoon is een instantie van de boomstructuur in afbeelding 3. Het bericht bevat de gegevens van een moeder inclusief de gegevens van twee van haar kinderen die als gerelateerde objecten zijn opgenomen. Van het eerste kind zijn tevens via een gerelateerde de adresgegevens opgenomen. De adresgegevens van de moeder en het tweede kind zijn weggelaten om ruimte te besparen.

```
<ns:object StUF:entiteittype="Persoon">
<ns:bsn>123456700</ns:bsn>
<ns:gemeente>Amsterdam</ns:gemeente>
<ns:geslacht>V</ns:geslacht>
....
<ns:relatie StUF:entiteittype="heeftKinderen">
<ns:gerelateerde StUF:entiteittype="Persoon">
<ns:relatie StUF:entiteittype="woontOp">
<ns:gerelateerde StUF:entiteittype="Adres">
....
</ns:gerelateerde>
</ns:relatie>
<ns:bsn>123456701</ns:bsn>
</ns:gerelateerde>
</ns:relatie>
<ns:relatie StUF:entiteittype="heeftKinderen">
<ns:gerelateerde StUF:entiteittype="Persoon">
<ns:bsn>123456702</ns:bsn>
....
</ns:gerelateerde>
</ns:relatie>
</ns:object>
```



Afbeelding 4: StUF-berichtenverkeer via webservices en een dienstenbus.

We zien hier de kracht van geneste XML-structuren. In SQL kunnen door middel van SELECT-statements alleen maar platte recordsets teruggegeven worden. De antwoordberichten van StUF kunnen complexe (boom)structuren bevatten.

Tenslotte

StUF heeft enerzijds het CRUD-model van SQL behoorlijk ingeperkt om de verwerkbaarheid te garanderen. Wat heb je immers aan zulke complexe berichten dat andere systemen ze niet kunnen verwerken? Anderzijds voegt StUF de nodige functionaliteit aan SQL toe (ontdubbeling, het expliciet omgaan met relaties en met historie). Dit brengt StUF dichter bij het business niveau van gemeentelijke registraties. Eén StUF-bericht dat vraagt om historische voorkomens van een object op een bepaald tijdstip in het verleden, kan leiden tot vele low-level SQL-operaties (inner-joins, outer-joins, enzovoort).

Naast CRUD-operaties biedt StUF ook de mogelijkheid om intelligentere operaties te definiëren. Hiervoor wordt het zogenaamde vrije bericht gebruikt. In dat geval standaardiseert StUF alleen de datatypes (de business objecten als operanden) en niet de operaties zelf. De webservice ontwerper bepaalt de semantiek van de operatie voor een specifiek toepassingsgebied. Een voorbeeld hiervan is StUF-TIOX (Taxatie Incourante Objecten); dit is een sector-specifiek koppelvlak gebaseerd op StUF voor het uitrekenen van een WOZ-waarde van een (woon)object op basis van een complex taxatiemodel.[2] De functionaliteit van dit soort intelligente services is natuurlijk minder eenvoudig opnieuw te gebruiken in andere sectoren of toepassingen dan een standaard CRUD-operatie. In de praktijk is gebleken dat de boomstructuren van StUF ook prima bruikbaar zijn in niet-CRUD situaties en een goede basis vormen voor het definiëren van specifieke functionaliteit (maatwerk).

Literatuur

EGEM (2009). Standaard Uitwisselings Formaat voor applicaties. StUF 03.01: In gebruik. www.egem-iteams.nl/stuf-gebruik.
Henri Korver. StUF: De Betekenis van Berichten. Tijdschrift <!ELEMENT. Jaargang 15, Nummer 1, April 2009. www.egem-iteams.nl/system/files/Vakblad_Element_StUFartikel_0309.pdf.
Henri Korver en Maarten van den Broek. StUF mini-primer: Uitgewerkte voorbeelden voor de semantiek van XML-berichten. Tijdschrift <!ELEMENT. Jaargang 15, Nummer 3, December 2009. www.surfgroepen.nl/sites/stuf/Shared%20Documents/A_StUF_Achtergrondinformatie/StUF_Artikel_Vakblad_Element_2010.PDF
www.egem-iteams.nl/stuf
www.surfgroepen.nl/sites/stuf/default.aspx
www.noiv.nl/node/65915

Noten

1. *CUD staat voor CRUD zonder Read-operatie.*
2. *www.wozinformatie.nl/public/nieuwsarchief/item/?itemID=6895&bc=nieuws*

Henri Korver en Maarten van den Broek

Henri Korver is voorzitter van de StUF Expertgroep en adviseert namens KING over architectuur en standaarden voor de e-overheid. Maarten van den Broek is de geestelijk vader van StUF en zelfstandig adviseur.