<Oracle en GML>

Thijs Brentjens
Afdeling Geodesie
Faculteit Civiele Techniek en Geowetenschappen
Technische Universiteit Delft

Oracle en GML

Thijs Brentjens
Verslag Geo-DBMS casestudy 2002
Afdeling Geodesie
Faculteit Civiele Techniek en Geowetenschappen
Technische Universiteit Delft
April 2002

Samenvatting

In dit rapport worden het onderzoek en de onderzoeksresultaten van de volgende vraag behandeld:

> *Welke ondersteuning biedt Oracle voor het genereren en inlezen van GML uit een Oracle-Spatial database en hoe kan dit toegepast worden?*

Om deze vraag te beantwoorden is literatuuronderzoek gepleegd, documentatie van Oracle bestudeerd en in de praktijk zijn een aantal 'tools' getoetst. Daarbij is gebruik gemaakt van een Oracle-Spatial database, met TOP10Vector data.

XML en GML

Extensible Markup Language (XML) kan gebruikt worden om in een tekstformaat gegevens te structureren, bijvoorbeeld gegevens uit een database. Hiertoe wordt een verzameling tags en attributen ingezet, die naar gelang de toepassing kan worden ontworpen. XML maakt het mogelijk eenvoudig gegevens uit te wisselen. De *interpretatie* van de gegevens is echter geheel afhankelijk van de applicatie.

Aan XML zijn een aantal technologieën verbonden, die onder andere gebruikt kunnen worden voor de definitie en toetsing van de structuur van bepaalde XML-documenten of transformatie van XML bestanden naar een ander (XML-)formaat met een XSLTransformatie.

Geography Markup Language (GML) is een op XML gebaseerde codering voor het transport en de opslag van geografische informatie, die zowel de ruimtelijke als niet-ruimtelijke eigenschappen van geografische objecten beslaat. GML maakt hiervoor gebruik van een set tags en attributen die door het OpenGis Consortium zijn gedefinieerd. GML is onder andere ontworpen om interoperabiliteit te ondersteunen.

XML functionaliteit van Oracle

Oracle 9i ondersteunt XML in die zin dat het in staat is de opslag, bevraging, presentatie en manipulatie van XML-data uit te voeren. Oracle stelt daarvoor een aantal technologieën ter beschikking.

Om XML te genereren kan gebruik worden gemaakt van XML SQL Utility (XSU), SQL functies en/of packages als `DBMS_XMLGEN`. Met de XSU command line kan XML gegenereerd worden als een onderdeel van de SQL Query zelf. Voor veel toepassingen voldoet deze gegenereerde XML niet aan het formaat of de structuur die men wenst. De gegenereerde XML moet dan aangepast worden, bijvoorbeeld met een XSLTransformatie.

Voor het inlezen van XML kan de XSU command line gebruikt worden. Hierbij is het van belang dat de structuur van het XML-document en de doel-tabel overeenkomen. Om dit te bereiken kan eerst een XSLTransformatie worden toegepast.

GML genereren en inlezen

Om de geografische data uiteindelijk in XML/GML te krijgen, is ervoor gekozen het generatieproces op te splitsen in twee stappen. De eerste is extractie of generatie van de gegevens in XML-formaat ('platte XML'). Deze XML-gegevens worden in de tweede stap getransformeerd naar een GML-formaat, gebruikmakend van een XSL Transformatie. De XSU Command Line is voor beide stappen gebruikt, omdat deze de functionaliteit voor het genereren en het transformeren combineert.

Na generatie van de platte XML, moet deze worden omgezet in GML. Dit kan gedaan worden met een XSLTransformatie die:

a) De niet-geometrische XML-elementen van het object omzet naar GML-elementen.

b) De geometrische elementen transformeert tot geldige GML-geometrie.

Voor stap a) is eenvoudige XSLT-code voldoende. Stap b) vergt echter meer van de transformatie, omdat de structuur van de geometrische gegevens veel complexer is. Voor zowel eenvoudige als multipolygonen is het in dit onderzoek gelukt met XSLT de data naar een GML-formaat te transformeren. Hierbij kunnen de gegevens die het ruimtelijke type van Oracle (SDO) bevat, gebruikt worden. Er is echter nog geen volledig geldig GML-document gegenereerd, omdat er nog een aantal GML-elementen ontbreekt. Mogelijkheden hoe dergelijke elementen (bijvoorbeeld begin- en eindtags van een document, 'bounding boxes' of objecthiërarchie) worden in dit rapport nader uitgewerkt.

Om GML te kunnen inlezen kan men 2 stappen onderscheiden:

1) GML bewerken tot de juiste structuur in XML-formaat en
2) deze XML inlezen in de doel-tabel.

Bij de eerste stap lijkt XSLT de aangewezen technologie om GML te transformeren. Wanneer gebruik wordt gemaakt van de XSU command line, is Oracle goed in staat om niet-geometrische elementen in te lezen. Geometrische elementen daarentegen leveren problemen op, waarbij Oracle aangeeft dat toch succesvol is ingelezen. Dit lijkt te wijzen op een fout in XSU, die inmiddels bij Oracle als bug is erkend.

Conclusies en aanbevelingen

De belangrijkste conclusies en aanbevelingen zijn:

- Het genereren van XML-documenten vanuit Oracle Spatial-tabellen verloopt goed, zolang er geen overerving wordt gebruikt in de tabel.
- Het inlezen van XML verloopt ook goed, zolang men zich beperkt tot eenvoudige datastructuren.
- Oracle's XML functionaliteit biedt mogelijkheden om GML te genereren vanuit Oracle Spatial-tabellen, ook voor complexe geometrieën als multipolygonen. Of daadwerkelijk alle complexe geometrieën correct kunnen worden weergegeven, zal nader onderzocht moeten worden.
- Het inlezen van GML levert problemen op bij de geometrie, wanneer XSU wordt gebruikt. Dit leidt tot de aanbeveling om of XSU aan te passen of een andere oplossing te zoeken om GML in te lezen.

De slotconclusie moet zijn dat Oracle ondersteuning biedt om GML te genereren. Voor het inlezen van GML schiet Oracle's XSU echter (nog) tekort.

Inhoudsopgave

1 Inleiding

Bij de Topografische Dienst Nederland (TDN) bestaan momenteel initiatieven voor het gebruiken van GML (Geography Markup Language) als uitwisselingsformaat. Binnen de sectie GISt van de TU Delft is een prototype systeem gemaakt voor het genereren van GML uit een Oracle database [5]. Indien men gebruik wil maken van GML bij de uitwisseling van geografische data, is het daarnaast van belang dat een GML-bestand ingelezen kan worden in een ruimtelijke database.

In het prototype van de Technische Universiteit Delft werd de GML code op een vrij laag niveau gegenereerd door het uitvoeren van printstatements in een Java programma. GML is een op XML (Extensible Markup Language) gebaseerde standaard [4]. Omdat Oracle veel ondersteuning biedt voor XML bestaat het vermoeden dat er een betere manier is om met GML te werken dan tot nu toe gedaan is. In het kader van het vak geo-DBMS in de eindstudie Geodesie (TU Delft) is een onderzoek gedaan naar het genereren en inlezen van GML met behulp van Oracle en de XML tools die Oracle biedt. Daarbij is de volgende hoofdvraag onderzocht:

> *Welke ondersteuning biedt Oracle voor het genereren en inlezen van GML uit een Oracle-Spatial database en hoe kan dit toegepast worden?*

Om de hoofdvraag te beantwoorden is deze onderverdeeld in de volgende deelvragen:

a) Wat zijn XML en GML?
b) Welke XML-functionaliteit biedt Oracle?
c) Hoe kunnen GML-bestanden gegenereerd worden met Oracle's XML-functionaliteit?
d) Op welke wijze kan in Oracle GML worden ingelezen in een ruimtelijke database?

Met name om deelvraag a) te beantwoorden is literatuuronderzoek gepleegd. Voor de overige deelvragen is documentatie van Oracle bestudeerd en in de praktijk zijn een aantal 'tools' getoetst. Daarbij is gebruik gemaakt van een Oracle-Spatial database. In dit onderzoek is niet getracht een systeem te ontwikkelen dat volledig geldige GML kan genereren en kan inlezen met de tools die Oracle verschaft. Het uitgangspunt is juist dat de *geschiktheid* en werking van Oracle's tools onderzocht is om GML te genereren en in te lezen.

Dit rapport beschrijft het onderzoek en de onderzoeksresultaten. In hoofdstuk 2 wordt ingegaan op de vraag wat XML en GML zijn. Daarbij komen ook aan XML gerelateerde technologieën aan bod. Vervolgens wordt de XML functionaliteit van Oracle beschreven in hoofdstuk 3. Hoe XML gegenereerd en ingelezen kan worden staat daarbij centraal. Daarbij wordt ook ingegaan op enkele praktijkervaringen met gebruikte Oracle tools. Op welke wijze men nu GML kan genereren en inlezen, wordt in hoofdstuk 4 beschreven. De resultaten uit het praktijkonderzoek, de stappen die daarbij ondernomen zijn en enkele problemen en kanttekeningen komen aan bod. Tot slot worden in hoofdstuk 5 de conclusies weergegeven en enkele aanbevelingen gedaan.

2 XML en GML

Om inzicht te krijgen hoe XML en GML gebruikt kunnen worden bij de TDN is het van belang iets te weten van de (technische) achtergronden van XML en GML. In dit hoofdstuk wordt ingegaan op de vraag wat XML (§ 2.1) en aan XML gerelateerde technologieën (§ 2.2) inhouden en voor welke doelen zij gebruikt kunnen worden. Hierbij worden ook enkele voor- en nadelen gegeven. Vervolgens wordt in § 2.3 een beschrijving van GML gegeven. De OpenGIS standaard voor GML, welke doelen GML kan dienen en enkele technische principes worden daarbij behandeld.

2.1 Extensible Markup Language

Extensible Markup Language (XML) is net als HTML afgeleid van Standard Generalized Markup Language (SGML), een internationale standaard voor documentatie [2]. HTML bestaat uit een aantal statische, van tevoren gedefinieerde tags, zoals `<head>` en `<title>`. XML biedt daarentegen de mogelijkheid om tags te creëren en configureren. Met deze tags kunnen gegevens worden afgebakend volgens een eigen specificatie of structuur. Op deze wijze kunnen gestructureerde gegevens, bijvoorbeeld uit een database, in een tekstbestand worden gezet (zie figuur 1). De tag `<adres>` wordt in XML een XML-element genoemd. Een voorbeeld van een attribuut is te vinden in de tag `<afdeling>`. Tussen de `'<'` en `'>'` van de tag `<afdeling>`, staat het attribuut `'naam'`, met de waarde `"geodesie"`.

XML wordt omschreven als een verzameling regels voor het ontwerpen van tekstformaten voor gestructureerde data, op een dusdanige manier dat het eenvoudig is voor een computer bestanden te genereren en te lezen. De *interpretatie* van de gegevens is geheel afhankelijk van de applicatie [1]. De specificatie XML 1.0 van het World Wide Web Consortium [6] definieert wat de 'tags' en 'attributen' zijn.

Met XML is het mogelijk om op een eenvoudige wijze gegevens uit te wisselen, omdat gebruik wordt gemaakt van een tekstformaat. Hierdoor zijn de gegevens goed leesbaar en interpreteerbaar voor zowel applicaties als mensen. Om de data te bekijken hoeft men dus niet te beschikken over de applicatie die de gegevens heeft geproduceerd (zoals vaak nodig bij binaire formaten) [1].

```
<afdeling naam="geodesie">
      <adres>Thijsseweg 11</adres>
      <plaats>Delft</plaats>
      <sectie>
            <naam>GiGb</naam>
            <medewerkers>13</medewerkers>
      </sectie>
      <sectie>
            <naam>GIStechnology</naam>
            <medewerkers>9</medewerkers>
      </sectie>
      ...
</afdeling>
```

Figuur 1. Voorbeeld van gegevens in een XML-gestructureerd bestand. Elementen en attributen worden weergegeven als '<element attribuut="waarde">'.

Een XML-document moet *well-formed* zijn. Dit wil zeggen dat het document moet voldoen aan een aantal syntaxregels [2]. Bijvoorbeeld: elk element is gesloten door óf een openings

(<naam>) en een sluittag (</naam>) óf, als het element leeg is, door een forward-slash (/) aan het eind van het element (<naam />). Om te controleren of een document *well-formed* is, wordt een zogenaamde parser gebruikt.

De belangrijkste voordelen van XML zijn [7]:
- XML is ontworpen met internet in gedachte
- XML processing technologie is makkelijk verkrijgbaar en goedkoop
- XML is leesbaar voor mensen
- XML is flexibel, er kunnen andere talen mee gedefinieerd worden
- XML is open
- XML kan worden bekeken met eenvoudige tools, zoals internetbrowsers

De belangrijkste nadelen van XML zijn [7]:
- XML heeft veel ruimte (opslag, bandbreedte en rekencapaciteit) nodig
- XML is erg handig voor tekst, maar minder voor binaire formaten

XML wordt ook wel omschreven als een familie van technologieën [1]. Naast de XML-specificatie van het W3C bestaan verschillende andere technologieën die ook tot de 'familie' van XML behoren. Deze technologieën bestaan uit extra tags en attributen bepaalde taken uit te voeren en richtlijnen heo taken uitgevoerd moeten worden. Men kan hiermee bijvoorbeeld aangeven hoe gegevens weergeven moeten worden in een browser (met CSS of XSL), de algemene structuur vastleggen in een extern bestand of de structuur van bestanden wijzigen naar een gewenst formaat. Op dergelijke technologieën wordt in § 2.2 dieper ingegaan.

2.2 Aan XML gerelateerde technologieën

Zoals in de vorige paragraaf al aangegeven bestaan er verschillende technologieën die gerelateerd zijn aan XML. Deze technologieën bieden onder andere mogelijkheden om:
- De structuur van een bepaald XML-formaat te definiëren en documenten volgens dit formaat te toetsen op geldigheid, waarop verderop in deze paragraaaf wordt ingegaan (DTD en XML Schema).
- Stijlen te definiëren om XML-gegevens weer te geven (XSL / CSS).
- XML te transformeren (XSLT ofwel XSLTransformatie). De transformatieregels staan hierbij in een XSLT stylesheet. Deze regels worden door een XSLT processor uitgevoerd op het XML-document.
- Te verwijzen naar andere bestanden of gegevens (XLink, XPointer).

Hieronder zal per technologie worden beschreven wat deze inhoudt en waarvoor deze toegepast kan worden.

Met DTD's en XML Schema's kan de structuur van een XML-formaat gedefinieerd worden. Daarbij kunnen ook regels worden opgesteld voor de elementen, bijvoorbeeld het datatype van het element. XML Schema's bieden hierbij meer mogelijkheden.

Beide technologieën werken daarnaast als toetsingsinstrument voor XML-bestanden. Gegenereerde XML-bestanden kunnen namelijk getoetst worden op een geldige structuur, die opgegeven is in de DTD of het XML Schema. Als een bestand voldoet aan die structuur en de XML specificatie, wordt het document 'geldig' verklaard. Dit proces wordt valideren genoemd.

CSS (Cascading Stylesheet) wordt vaak gebruikt om een weergavestijl te definiëren voor webpagina's (in HTML), maar kan ook worden gebruikt voor XML-bestanden. XSL (Extensible Stylesheet Language) kan ook gebruikt worden voor de weergave van XML. In een XSL stylesheet wordt aangegeven hoe tags en de inhoud van tags moeten worden weergegeven. Zo kan men bijvoorbeeld voor een XML-document aangeven welke XML-tags HTML-tags moeten worden, zodat de inhoud van het XML-document in een HTML-browser kan worden weergegeven. In feite wordt het XML-document dan getransformeerd naar HTML. XSL kan gebruikt worden om XML-bestanden te transformeren naar elk ander (tekst)formaat. Een dergelijke transformatie wordt dan aangeduid met XSLT, oftewel XSL Transformatie. Wanneer XML-bestanden geconverteerd moeten worden, kan deze XSL Transformatie dus gebruikt worden.

In een XSLT stylesheet worden transformatieregels opgesteld, waarmee tags opgezocht en elementen gewijzigd kunnen worden (voor een voorbeeld zie § 4.2.2 of de bijlagen). Deze transformatieregels worden vervolgens door een XSLT processor toegepast op het brondocument, waardoor een nieuw document wordt geproduceerd [2]. Dit nieuwe document kan een andere structuur hebben en meer of minder tags en/of attributen bevatten. XSLT is dus een transformatietaal voor het hergroeperen, toevoegen of verwijderen van tags en attributen [1]. Een XSLT processor past deze regels toe op het document en voert zo de transformatie uit.

Met XPointer en XLink wordt verwezen naar gegevens in een XML-document of externe bestanden [2]. XPointer is ontworpen om een element of een verzameling elementen te lokaliseren in een XML-document. Hierbij wordt gebruik gemaakt van unieke identifiers of posities in een document. XLink verschaft geavanceerde mogelijkheden om te verwijzen, bijvoorbeeld in meerdere richtingen en extern. Het ligt buiten het bereik van dit rapport om hier dieper op in te gaan.

2.3 Geography Markup Language

Dit rapport behandelt de vraag welke ondersteuning Oracle biedt om GML te genereren en in te lezen. Deze paragraaf gaat daarbij in op de vraag wat GML nu is. Het OpenGis Consortium (OGC) definieert GML als volgt [4]:

> *Geography Markup Language (GML) is een op XML gebaseerde codering voor het transport en de opslag van geografische informatie, die zowel de ruimtelijke als niet-ruimtelijke eigenschappen van geografische objecten beslaat.*

Uit deze definitie volgt dat GML specificeert hoe geografische objecten moeten worden gemodelleerd en weergegeven in een tekstformaat dat gebruik maakt van tags. Er wordt immers gebruik gemaakt van XML. Daarbij worden ruimtelijke en niet-ruimtelijke eigenschappen van geografische objecten onderscheiden.

GML is onder andere ontworpen om interoperabiliteit te ondersteunen en doet dat door het verschaffen van een basisset geometrie tags (alle systemen die GML ondersteunen gebruiken dezelfde geometrie tags), zoals `<polygon>` of `<coordinates>`, een algemeen data model voor de objecten en eigenschappen en een mechanisme om schema's van applicaties te maken en te delen [4].

Geografische objecten worden in GML dus beschreven met behulp van tags. Een voorbeeld hiervan is in figuur 2 te vinden. De ruimtelijke eigenschappen zijn weergegeven

tussen <gml:geometryProperty> en </gml:geometryProperty>, de niet-ruimtelijke eigenschappen daaronder.

```
<tdn:waterdeel XMLns:tdn="http://www.gdmc.nl/tdn" id="TOP10.6300005">
   <gml:geometryProperty>
      <gml:Polygon srsName="EPSG:7048">
         <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:coordinates>119876.336,576934.906,0.0
               119881.605,576937.761,0.0 119887.313,576938.42,0.0
               119891.923,576937.541,0.0 119896.094,576933.369,0.0
               119896.972,576929.417,0.0 119894.777,576927.66,0.0
               119890.167,576926.342,0.0 119879.629,576927.221,0.0
               119876.555,576929.417,0.0 119874.579,576932.71,0.0
               119876.336,576934.906</gml:coordinates>
            </gml:LinearRing>
         </gml:outerBoundaryIs>
      </gml:Polygon>
   </gml:geometryProperty>
   <tdn:top10_id>6300005</tdn:top10_id>
   <tdn:begindatum>06 Jul 2001 08:28:02</tdn:begindatum>
   <tdn:einddatum>31 Dec 2999 23:59:59</tdn:einddatum>
   <tdn:type>Verbinding</tdn:type>
   <tdn:watertype>Meer, plas, ven vijver</tdn:watertype>
   <tdn:breedteklasse>Onbekend</tdn:breedteklasse>
   <tdn:breedte>Onbekend</tdn:breedte>
   <tdn:hoofdafwatering>Nee</tdn:hoofdafwatering>
   <tdn:zoutgehalte>Zoet</tdn:zoutgehalte>
   <tdn:toegankelijkheid>Openbaar</tdn:toegankelijkheid>
   <tdn:fysiek_voorkomen>Overig</tdn:fysiek_voorkomen>
   <tdn:gebruik>Overig</tdn:gebruik>
   <tdn:stroomrichting>Stilstaand</tdn:stroomrichting>
   <tdn:status>In gebruik</tdn:status>
   <tdn:naam>Onbekend</tdn:naam>
   <tdn:hoogteniveau>0</tdn:hoogteniveau>
   <tdn:tdncode>6113</tdn:tdncode>
</tdn:waterdeel>
```

Figuur 2 Een geografisch object in GML-formaat

In bovenstaand voorbeeld is duidelijk te zien dat GML gebaseerd is op XML. Oracle biedt veel ondersteuning voor XML. Daarom bestaat het vermoeden dat men met Oracle's XML-functionaliteit beter GML kan schrijven dan met de huidige methode, namelijk print-statements in een Java-programma. Op deze XML-functionaliteit wordt in het volgende hoofdstuk ingegaan.

3 Oracle's XML functionaliteit

Oracle biedt ondersteuning voor XML. In dit hoofdstuk wordt ingegaan op de algemene ondersteuning die Oracle biedt voor XML (§ 3.1) en de specifieke functionaliteit om XML te genereren (§ 3.2) en in te lezen (§ 3.3).

3.1 Ondersteuning van XML door Oracle

Oracle 9i ondersteunt XML in die zin dat het in staat is de opslag, bevraging, presentatie en manipulatie van XML-data uit te voeren [3]. Oracle noemt de technologieën die dit uitvoeren Oracle XML-Enabled Technologies. Er worden hierbij een aantal XML componenten onderscheiden. De belangrijkste soorten componenten zijn [3]:

- *Database XML Support.* Oracle rekent hiertoe onder andere:
  - een datatype voor opslag, bevraging en het ophalen van XML-documenten (`XMLType`)
  - SQL functies om XML-documenten te creëren en samen te voegen
  - een package om XML te creëren vanuit SQL queries (`DBMS_XMLGEN`).

- Verschillende *XML Developer's Kits* (XDK). De XDK voor Java is het meest uitgebreid. Deze bevat:
  - een XML parser
  - een XSLT processor
  - een XML Schema processor
  - een XML Class Generator
  - XML SQL Utility (XSU)

  Voor C, C++ en PL/SQL zijn ook XML Developer Kits beschikbaar. Deze bevatten echter minder componenten, bijvoorbeeld alleen een parser en Schema processor. Een XDK kan gebruikt worden om een (XML-)applicatie te maken met Oracle.

XML gegevens kunnen in twee verschillende vormen worden gebruikt in Oracle: als ontbonden XML of als geheel XML-document. Bij de eerste wordt een XML-document opgeslagen als losse componenten (XML-elementen en -attributen) in een object-relationele vorm (bijvoorbeeld in tabellen). De XML SQL Utility, SQL functies en/of packages kunnen worden gebruikt om XML-documenten te genereren en XML te plaatsen in de object-relationele vorm [3].

Deze ontbonden vorm is de meest geschikte manier, als de database of database-applicatie reeds bestaat (bijvoorbeeld Top10Vector in een object-relationele database). Omdat de gegevens geëxtraheerd worden uit de database en vervolgens in XML-formaat worden gezet, blijft de structuur van de gegevens in de database namelijk in tact. Hierdoor is bevraging en manipulatie (wijzigen, toevoegen, verwijderen, enz.) van de gegevens veel eenvoudiger, dan bij opslag van een geheel XML-document.

De tweede vorm (als geheel XML-document) gebruikt het XMLType of CLOB/BLOB's (Character Large OBject of Binary Large OBject) om een XML-documenten in zijn geheel op te slaan. Oracle biedt daarbij een aantal functies om deze gegevens te doorzoeken [3]. Manipulatie van de gegevens is omslachtig, omdat eerst het document moet worden opgehaald, dan de wijzigingen worden doorgevoerd en vervolgens het gehele document weer wordt teruggeplaatst. Voor grote hoeveelheden data (bijvoorbeeld een landelijk bestand van Top10Vector) is dit extra omslachtig. Het maakt ook een kleine mutatie duur, omdat het gehele document moet worden gebruikt. Ook de selectie van gegevens is weinig flexibel, omdat men werkt met van tevoren opgestelde documenten. Deze vorm is geschikt wanneer de

XML-gegevens niet vaak wijzigen, bijvoorbeeld een dataset op een vastliggend tijdstip (alle wegen uit TOP10Vector op 1 januari 2001).

De eerste vorm, als losse componenten, lijkt dus het meest geschikt voor geografische gegevens, omdat deze vaak wijzigen in beperkte gebieden en/of bevraagd worden. In het vervolg van het rapport wordt daarom ingegaan op tools om XML te genereren (§ 3.2) en in te lezen (§ 3.3) vanuit een object-relationele vorm.

3.2 XML genereren

Om XML te genereren, vanuit de losse componenten, kan gebruik worden gemaakt van XML SQL Utility (XSU), SQL functies en/of packages als `DBMS_XMLGEN` (zie § 3.1) [3]. Er is in dit onderzoek gekozen voor XSU omdat XSU de XML-functionaliteit eenvoudig toegankelijk maakt. SQL functies en de packages vereisen meer werk, met name programmeerwerk.

Met XSU kan XML gegenereerd worden als een onderdeel van de SQL Query zelf [3]. Er wordt hierbij (in een bepaalde toepassing) een SQL query uitgevoerd, die als resultaat de gegevens in XML oplevert. Het is mogelijk om tijdens het genereren eenvoudige transformaties toe te passen (bijvoorbeeld bepaalde tags herbenoemen) of een XSLTransformatie uit te voeren. Deze wordt vervolgens automatisch toegepast op de gegenereerde XML door de XSLT processor. Deze mogelijkheid blijkt nuttig bij het genereren van GML (hoofdstuk 4).

XSU is beschikbaar voor Java en PL/SQL. Daarbij ondersteunt XSU nagenoeg alle datatypen in de Oracle 9i database server [3]. XSU kan op drie manieren gebruikt worden:

- met een Java API (Application Program Interface, zie ook Bijlage IV)
- met een PL/SQL API
- met de Java command line

Voor de eerste twee moet dus respectievelijk een Java of een PL/SQL programma geschreven worden om deze te gebruiken, de Java command line was reeds geïnstalleerd in Oracle.

In dit onderzoek is gebruik gemaakt van XSU's Java command line, die vanaf hier met 'XSU command line' zal worden aangeduid. Deze XSU command line biedt namelijk een snelle en eenvoudige toegang tot de XSU functionaliteit. In een Unix-shell kan met het volgende commando bijvoorbeeld XML worden gegenereerd:

```
java OracleXML getXML –user "naam/wachtwoord" "select * from testtabel"
```

Vervolgens worden vier stappen uitgevoerd:

1) contact gelegd met de database voor gebruiker `"naam"` met wachtwoord `"wachtwoord"`
2) de SQL-query `"select * from testtabel"` uitgevoerd
3) het resultaat van de query geconverteerd naar XML en tenslotte
4) de XML weergegeven [3].

In Bijlage I staan meer details over het gebruik van de command line.

Voor veel toepassingen voldoet de gegenereerde XML niet aan het formaat of de structuur die men wenst. De gegenereerde XML moet dan aangepast worden. Oracle geeft hiervoor drie mogelijkheden [3]:

1) bron aanpassing ('source customization')
2) afbeeldingaanpassing ('mapping customization')
3) post-generatie aanpassing ('post-generation customization')

ad 1) In het database schema kan een 'view' worden gemaakt die de gewenste (XML) structuur weergeeft. Ook kan in de query door subqueries, aliassen van attribuutnamen (bijvoorbeeld `"select naam as achternaam from .."`) en aliassen om de waarde als XML-*attribuut* weer te geven in plaats van XML-*element* (bijvoorbeeld `"select naam, geslacht as '@naam' from .."`, dit geeft het volgende resultaat: `<naam geslacht="man">Karel</naam>`).

ad 2) Met de command line kunnen tijdens het genereren bijvoorbeeld tags worden verwijderd of de naam of het formaat van tags worden gewijzigd. Een voorbeeld hiervan is het argument `–rowTag "<doc>"` waarbij het rootelement `<Row>` (dit is de standaardwaarde) wordt vervangen door `<doc>`. Voor meer details zie Bijlage I.

ad 3) Door een XSLT stylesheet aan te roepen. Aan het commando moet `-setXSLT "naamstylesheet"` worden toegevoegd (Bijlage I). De XSLTransformatie in de stylesheet `"naamstylesheet"` wordt uitgevoerd na de conversie naar XML (stap 3), maar voor de weergave van de uiteindelijke XML.

Met de command line is de functionaliteit van XSU getest en deze is ook gebruikt om GML te genereren (hoofdstuk 4). De hier gebruikte functie getXML is overigens ook buiten de command line beschikbaar in XSU, dus ook voor Java en PL/SQL API's.

Bij het testen van de XSU command line ontstonden problemen, wanneer een tabel bevraagd werd waarin gebruik werd gemaakt van overerving. De gegenereerde XML bleek geërfde elementen niet weer te geven (Bijlage II). Ook in het XML Schema dat gegenereerd werd bleken de elementen niet voor te komen. Een Oracle database die gebruik maakt van overerving kan dus serieuze problemen ondervinden als XML gegenereerd moet worden.

3.3 XML inlezen

Zodra men XML wil gaan gebruiken voor de uitwisseling van gegevens, is het niet alleen van belang XML te kunnen genereren, maar ook te kunnen inlezen in de eigen database. Hiervoor kan in Oracle ook XSU (XML SQL Utility) ingezet worden. Net als bij het genereren kan hiervoor een Java of PL/SQL API of de XSU command line worden gebruikt [3].

XSU maakt op basis van het in te lezen XML-document en de doel-tabel, de tabel waarin de XML moet worden ingelezen, een SQL-instructie. Deze SQL-instructie plaatst met het INSERT-statement de gegevens vanuit het document in de overeenkomstige kolommen in de doel-tabel [3]. Het is daarom van belang dat de structuur van het XML-document en de doel-tabel overeenkomen. Concreet betekent dit dat de XML-elementen exact dezelfde namen moeten hebben als de kolommen in de doel-tabel. Om dit te bereiken kan eerst een XSLTransformatie worden toegepast.

Om het inlezen te testen is gebruik gemaakt van de XSU command line. Analoog aan de generatie-functie `getXML`, kan nu `putXML` gebruikt worden:

```
java OracleXML putXML –user "naam/wachtwoord" –filename "invoer.xml"
"doeltabel"
```

Het bestand `invoer.xml` wordt nu omgezet in een SQL-instructie (INSERT) voor de tabel `"doeltabel"`, die vervolgens wordt uitgevoerd. Er kunnen ook weer een aantal opties worden ingesteld. Zo kan een XSLT stylesheet worden aangeroepen om de structuur van de XML en de doel-tabel te laten overeenkomen. Deze XSLTransformatie wordt dan uitgevoerd, voordat de SQL-instructie wordt gemaakt.

Het inlezen van XML verloopt prima, zolang men zich beperkt tot eenvoudige structuren, bijvoorbeeld een rij zonder geneste typen, zoals het type SDO-Geometry in Oracle Spatial, of overerving. Er is geprobeerd dergelijke structuren in te lezen, maar dit is niet geheel gelukt. Voor meer details wordt verwezen naar § 4.4.

4 Van tabel naar GML en andersom

4.1 Introductie

Om geografische data uit een Oracle-Spatial tabel in GML-formaat te verkrijgen, zijn de volgende vragen relevant:

a) Hoe kunnen de niet geometrische attributen van de geografische objecten in GML-formaat gegenereerd worden?
b) Hoe kan de geometrie in GML-formaat gegenereerd worden?
c) Voor welke geografische object-typen (bijvoorbeeld lijnen, polygons, multilijnen of multipolygons) is automatische generatie met behulp van de Oracle-tools mogelijk?
d) Hoe kan extra objecthiërarchie en de basis GML structuren als Feature Collection worden gegenereerd en weergegeven in het uiteindelijke GML-document?

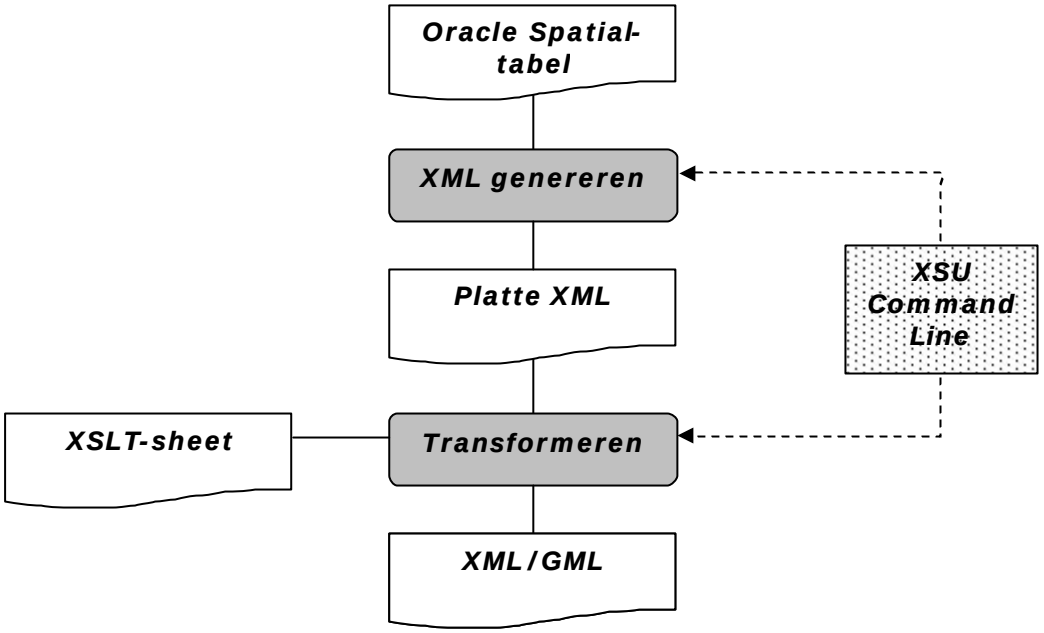
In het vorige hoofdstuk zijn tools ter sprake gekomen die gebruikt kunnen worden om binnen Oracle XML te kunnen genereren, inlezen en aanpassen. Met het oog op de beantwoording van de bovenstaande vragen, is een aantal van deze tools ingezet om GML te genereren en inlezen. De geschiktheid van de tools, de resultaten die ermee behaald zijn en de wijze waarop, worden in § 4.2 besproken.

Tijdens het onderzoek is uiteindelijk geen (volledig) geldig GML-document gemaakt. Mogelijkheden hoe, met behulp van de behaalde resultaten, dit alsnog bereikt kan worden, zijn het onderwerp van § 4.3. Daarbij wordt ook ingegaan op de vraag hoe extra structuurinformatie (zoals de objecthiërarchie) kan worden toegevoegd aan GML-bestanden.

Naast het genereren van GML is ook onderzocht hoe GML ingelezen kan worden in een Oracle Spatial-database. Welke tools hierbij gebruikt zijn, de genomen stappen en de problemen bij het inlezen, worden in § 4.4 behandeld. Vanwege de problemen en het gebrek aan tijd om dieper in te gaan op het inlezen van GML, is de paragraaf meer van conceptuele aard dan dat deze gericht is op behaalde, praktische resultaten.

4.2 GML genereren in de praktijk

Om de Oracle functionaliteit te testen is gebruik gemaakt van Top10vector data van de Topografische Dienst Nederland (TDN) in Oracle Spatial-tabellen. Om deze data uiteindelijk in XML/GML te verkrijgen, is ervoor gekozen het proces op te splitsen in twee stappen. De eerste is extractie / generatie van de gegevens in XML-formaat. Deze XML-gegevens worden in de tweede stap getransformeerd naar een GML-formaat, gebruikmakend van een XSL Transformatie (§ 2.2). In figuur 3 is dit proces schematisch weergegeven. De XSU Command Line is voor beide stappen gebruikt, omdat deze de functionaliteit voor het genereren en het transformeren combineert. Daarbij wordt in één commando de XML gegenereerd en gelijk getransformeerd.



Figuur 3 Schematische weergave van het generatie-proces

4.2.1 De geografische gegevens in XML

Met de XSU Command Line is uit een tabel met polygonen (de tabel 'waterdeel', zie figuur 4) 'platte XML' gegenereerd. Elke rij (geografisch object), die het resultaat is van de selectie-query, wordt een set XML-elementen, die begint met `<Row>` en eindigt met `</Row>`. De gegevens uit de oorspronkelijke tabel danwel query worden tussen tags geplaatst, die de structuur van de attributen weergeven (figuur 5). Hiervoor is het volgende commando gebruikt:

```
java OracleXML getXML –user "naam/wachtwoord" "select * from
waterdeel"
```

```
Naam attribuut                                  Type
----------------------------------------- -------- ----------------------
GEOMETRY                                        MDSYS.SDO_GEOMETRY
SHP_OID                                         NUMBER
SHP_TOP10_ID                                    NUMBER
SHP_BEGINDATUM                                  VARCHAR2(256)
SHP_EINDDATUM                                   VARCHAR2(256)
SHP_TYPE                                        VARCHAR2(256)
SHP_WATERTYPE                                   VARCHAR2(256)
SHP_BREEDTEKLA                                  VARCHAR2(256)
SHP_BREEDTE                                     VARCHAR2(256)
SHP_HOOFDAFWAT                                  VARCHAR2(256)
SHP_ZOUTGEHALT                                  VARCHAR2(256)
SHP_TOEGANKELI                                  VARCHAR2(256)
SHP_VOORKOMEN                                   VARCHAR2(256)
SHP_GEBRUIK                                     VARCHAR2(256)
SHP_STROOMRICH                                  VARCHAR2(256)
SHP_STATUS                                      VARCHAR2(256)
SHP_NAAM                                        VARCHAR2(256)
SHP_HOOGTENIVE                                  NUMBER
SHP_TDNCODE                                     NUMBER
```

Figuur 4 Beschrijving van de attributen van de tabel waterdeel, zoals deze gebruikt is in dit onderzoek

In onderstaand voorbeeld kan men zien dat niet-geometrische attributen per rij rechstreeks zijn omgezet in XML-elementen, zoals bijvoorbeeld het element <SHP_TDNCODE>6113</SHP_TDNCODE>. In de oorspronkelijke tabel komt het attribuut SHP_TDNCODE voor, met voor dit object een waarde '6113'. De geometrische attributen zijn te vinden tussen <geometry> en </geometry>. Hierbij worden de waarden van het ruimtelijke type SDO-geometry ook tussen tags gezet, bijvoorbeeld de coördinaten, die nu in een lijst van items tussen de tags <SDO_ORDINATES> en </SDO_ORDINATES> staan. Hierbij blijft de structuur van het type SDO-geometry behouden. Het gehele object wordt tenslotte afgebakend door de tags <ROW num="10"> en </ROW>.

```
<ROW num="10">
      <GEOMETRY>
         <SDO_GTYPE>2003</SDO_GTYPE>
         <SDO_ELEM_INFO>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
         </SDO_ELEM_INFO>
         <SDO_ORDINATES>
            <SDO_ORDINATES_ITEM>119876.336</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576934.906</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119881.605</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576937.761</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119887.313</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576938.42</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119891.923</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576937.541</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119896.094</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576933.369</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119896.972</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576929.417</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119894.777</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576927.66</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119890.167</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576926.342</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119879.629</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576927.221</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119876.555</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576929.417</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119874.579</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576932.71</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119876.336</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576934.906</SDO_ORDINATES_ITEM>
         </SDO_ORDINATES>
      </GEOMETRY>
      <SHP_OID>6300005</SHP_OID>
      <SHP_TOP10_ID>6300005</SHP_TOP10_ID>
      <SHP_BEGINDATUM>06 Jul 2001 08:28:02</SHP_BEGINDATUM>
      <SHP_EINDDATUM>31 Dec 2999 23:59:59</SHP_EINDDATUM>
      <SHP_TYPE>Verbinding</SHP_TYPE>
      <SHP_WATERTYPE>Meer, plas, ven vijver</SHP_WATERTYPE>
      <SHP_BREEDTEKLA>Onbekend</SHP_BREEDTEKLA>
      <SHP_BREEDTE>Onbekend</SHP_BREEDTE>
      <SHP_HOOFDAFWAT>Nee</SHP_HOOFDAFWAT>
      <SHP_ZOUTGEHALT>Zoet</SHP_ZOUTGEHALT>
      <SHP_TOEGANKELI>Openbaar</SHP_TOEGANKELI>
      <SHP_VOORKOMEN>Overig</SHP_VOORKOMEN>
      <SHP_GEBRUIK>Overig</SHP_GEBRUIK>
      <SHP_STROOMRICH>Stilstaand</SHP_STROOMRICH>
```

```
        <SHP_STATUS>In gebruik</SHP_STATUS>
        <SHP_NAAM>Onbekend</SHP_NAAM>
        <SHP_HOOGTENIVE>0</SHP_HOOGTENIVE>
        <SHP_TDNCODE>6113</SHP_TDNCODE>
</ROW>
```

Figuur 5 Een voorbeeld van een object (rij) uit de tabel 'waterdeel' in platte XML.

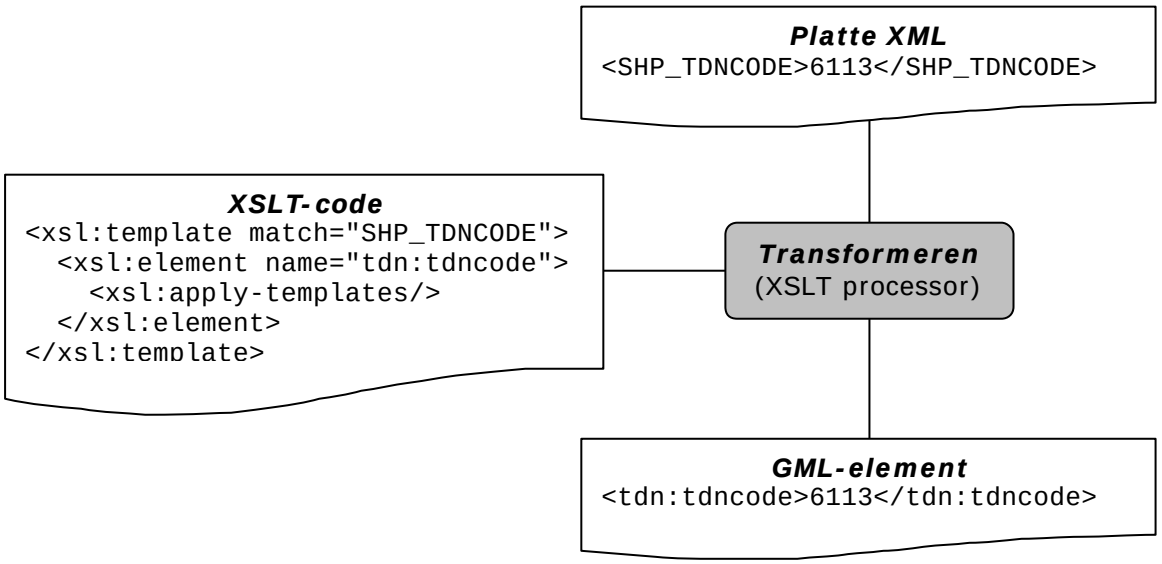
De platte XML moet worden omgezet in GML. Zoals eerder aangegeven, is gekozen voor een XSL Transformatie. Deze transformatie moet:

a) De niet-geometrische XML-elementen van het object omzetten naar GML-elementen, zoals die gedefinieerd zijn in het XML/GML Schema;
b) De geometrische elementen transformeren tot geldige GML-geometrie.

Beide transformatiestappen zullen hieronder beschreven worden.

4.2.2 Transformatie van niet-geometrische elementen

Het transformeren van een XML-element naar een XML-element in een ander formaat (zoals GML, waarbij in dit onderzoek het TDN Schema [5] is gebruikt), kan met behulp van XSLT op een eenvoudige wijze (hoofdstuk 3) geschieden. Het oorspronkelijke element wordt in de transformatie geïdentificeerd en de waarde wordt gekopieerd naar het nieuwe element, door XSLT-code toe te passen. Hieronder staat in figuur 6 schematisch uitgewerkt hoe dit proces verloopt voor één element. Voor de gehele XSLT-code om objecten uit de tabel 'waterdeel' te transformeren van platte XML naar GML, wordt verwezen naar Bijlage IIIc.



Figuur 6 Transformatieproces XML-element naar GML-element

De transformatie van niet-geometrische elementen is eenvoudig en 'straight-forward', zolang de niet-geometrische elementen geen bijzondere datastructuren hebben. In dat geval is de XSL Transformatie complexer. Echter, aangezien de meeste geografische objecten momenteel geen gebruik maken van ingewikkelde datastructuren voor niet-geometrische elementen, mag de transformatie hiervan naar GML weinig problemen opleveren.

4.2.3 Transformatie van geometrische elementen

De geometrie in Oracle's SDO-geometry type wordt beschreven aan de hand van 5 elementen. In de gebruikte dataset zijn er hier 3 van gebruikt, te weten:

- GTYPE (geometrie type);

- ELEM_INFO (meta-informatie over het soort geometrie, bijvoorbeeld of het element een punt, lijn of polygon is en uit welke objecten het bestaat en per object waar de ordinaten beginnen in de ordinatenlijst);
- ORDINATES (de ordinaten van het gehele object, weergegeven in een lijst).

Bij bestudering van de platte XML (figuur 5) valt te zien dat deze elementen terugkeren als elementen binnen <GEOMETRY>. Hierbij worden de waarden die behoren bij de 3 elementen weergegeven als sub-elementen voor de ELEM_INFO en ORDINATES. De ordinaten worden daarbij elk als een element (ITEM) aan deze lijst toegevoegd. Bovenstaande gegevens zijn 2-dimensionaal. Dit blijkt uit het GTYPE van de gegevens. Dit GTYPE is een 4-cijferige code voor het type geometrie. Het eerste cijfer geeft de dimensie van de data aan. Voor de ordinatenlijst betekent dit dat per (x,y)-coördinaat eerst de x- en vervolgens de y-ordinaat wordt weergegeven.

Deze gegevens moeten zodanig opnieuw gestructureerd worden dat de geometrie als geldige GML-geometrie in het document terechtkomt. Daartoe wordt in de XSL Transformatie de ELEM_INFO gebruikt om te bepalen wat voor geometrie het object heeft (2D of 3D, punt, lijn, polygon, enkelvoudig of meervoudig, enz) en worden aan de hand daarvan de vereiste elementen gemaakt en de ordinaten in het vereiste formaat geschreven (bijvoorbeeld: x1,y1,z1 x2,y2,z2, enz.). In figuur 6 is te zien hoe XSLT code eruit ziet om een polygon weer te geven, indien uit de ELEM_INFO blijkt dat het object een polygon is. Figuur 7 geeft de XSLT-code om ordinaten weer te geven als x,y,z-coördinaten. Bij 2-dimensionale objecten wordt hier de z-coördinaat op 0.0 gezet.

```
<xsl:template match="GEOMETRY"> <!-- zoek GEOMETRY op in XML-document
en plaats element gml:geometryProperty -->
 <xsl:element name="gml:geometryProperty">

  <xsl:if test="substring(string(SDO_GTYPE),4,1) = 3"> <!-- object is
een polygon als 4e cijfer in waarde van SDO_GTYPE 3 is, plaats dan
elementen voor polygon -->
   <xsl:element name="gml:Polygon">
    <xsl:attribute name="srsName">EPSG:7048</xsl:attribute>
    <xsl:element name="gml:outerBoundaryIs">
     <xsl:element name="gml:LinearRing">
      <xsl:element name="gml:coordinates">
       <xsl:apply-templates/> <!-- plaats alle waardes en eventuele
subelementen en –attributen van GEOMETRY in het element
gml:coordinates. Eventuele andere transformaties worden op een andere
plaats in de XSLT-code uitgevoerd. Sluit alle geopende elementen-->
      </xsl:element>
     </xsl:element>
    </xsl:element>
   </xsl:element>
  </xsl:if>

 </xsl:element>
</xsl:template>
```

*Figuur 6 XSLT code (**vet** gedrukt) om GML-elementen weer te geven voor een polygon. Commentaar is cursief weergegeven, tussen "<!--" en "-->".*

```
<xsl:template match="SDO_ORDINATES_ITEM"> <!-- zoek elk element
SDO_ORDINATES_ITEM op, dit is een ordinaat -->
  <xsl:apply-templates/> <!-- plaats het ordinaat in het document -->
  <xsl:if test="position() > 0"> <!-- na het eerste ordinaat moeten
de ordinaten gescheiden worden door komma's, de gehele coördinaten
door spaties -->
```

```
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 2"> <!--
object is 2-dimensionaal -->
      <xsl:if test="round(0.5 * (position())) = 0.5 *
      (position())">,0.0 </xsl:if>
<!-- plaats na elk 2e ordinaat (y) een komma en de z-waarde op 0.0-->
      <xsl:if test="round(0.5 * (position())) != 0.5 *
      (position())">,</xsl:if>
<!-- plaats na elk 1e ordinaat (x) een komma-->
   </xsl:if>
<!-- einde 2-dimensionale coördinaten -->
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 3"> <!--
object is 3-dimensionaal -->

      <xsl:if test="round(((position())) div 3) != ((position())) div
      3">,</xsl:if> <!-- elk x,y,z-ordinaat wordt gescheiden door een
      komma-->
      <xsl:if test="round(((position())) div 3) = ((position())) div
      3"><xsl:text> </xsl:text> <!-- elk geheel x,y,z-cöordinaat
      wordt van andere coördinaten gescheiden door een spatie-->
      </xsl:if>
    </xsl:if>

  </xsl:if> <!-- einde if, voor position groter dan 0 -->

</xsl:template>
```

*Figuur 7 XSLT code (**vet** gedrukt) om ordinaten te plaatsen als x,y,z-coördinaten, waarbij de z-coördinaat in 2-D op 0.0 wordt gezet. In 3-D worden de ordinaten gescheiden door komma's, de gehele coördinaten door spaties. Commentaar is cursief weergegeven, tussen "<!--" en "-->".*

Het voert te ver om op deze plaats de gehele XSLT-code te behandelen. Voor de exacte XSLT code wordt verwezen naar Bijlage IIIc. Het blijkt daarmee mogelijk om platte XML inclusief Oracle's SDO-geometry met een dergelijke XSLTransformatie om te zetten in GML-formaat. In figuur 8 wordt hiervan nog het resultaat gegeven van het gehele object uit figuur 5 na transformatie tot GML-formaat. Hierin is duidelijk te zien hoe de geometrie (van `<gml:geometryProperty>` tot en met `</gml:geometryProperty>`) en de overige attributen weergegeven worden.

```
<tdn:waterdeel XMLns:tdn="http://www.gdmc.nl/tdn" id="TOP10.6300005">
   <gml:geometryProperty>
      <gml:Polygon srsName="EPSG:7048">
         <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:coordinates>119876.336,576934.906,0.0
               119881.605,576937.761,0.0 119887.313,576938.42,0.0
               119891.923,576937.541,0.0 119896.094,576933.369,0.0
               119896.972,576929.417,0.0 119894.777,576927.66,0.0
               119890.167,576926.342,0.0 119879.629,576927.221,0.0
               119876.555,576929.417,0.0 119874.579,576932.71,0.0
               119876.336,576934.906</gml:coordinates>
            </gml:LinearRing>
         </gml:outerBoundaryIs>
      </gml:Polygon>
   </gml:geometryProperty>
   <tdn:top10_id>6300005</tdn:top10_id>
   <tdn:begindatum>06 Jul 2001 08:28:02</tdn:begindatum>
   <tdn:einddatum>31 Dec 2999 23:59:59</tdn:einddatum>
   <tdn:type>Verbinding</tdn:type>
```

```
      <tdn:watertype>Meer, plas, ven vijver</tdn:watertype>
      <tdn:breedteklasse>Onbekend</tdn:breedteklasse>
      <tdn:breedte>Onbekend</tdn:breedte>
      <tdn:hoofdafwatering>Nee</tdn:hoofdafwatering>
      <tdn:zoutgehalte>Zoet</tdn:zoutgehalte>
      <tdn:toegankelijkheid>Openbaar</tdn:toegankelijkheid>
      <tdn:fysiek_voorkomen>Overig</tdn:fysiek_voorkomen>
      <tdn:gebruik>Overig</tdn:gebruik>
      <tdn:stroomrichting>Stilstaand</tdn:stroomrichting>
      <tdn:status>In gebruik</tdn:status>
      <tdn:naam>Onbekend</tdn:naam>
      <tdn:hoogteniveau>0</tdn:hoogteniveau>
      <tdn:tdncode>6113</tdn:tdncode>
   </tdn:waterdeel>
```

Figuur 8 Resultaat na XSL Transformatie

Multipolygonen

In bovenstaande paragraaf is ervan uitgegaan dat een object in de tabel een enkelvoudig object (in de tabel ‘waterdeel’ een polygon) is. Maar stel dat het object een multipolygon is, oftewel een object dat bestaat uit meerdere polygonen. Kan XSLT dergelijke complexe objecten ook aan? In het onderzoek is de tabel ‘terrein’ gebruikt om dit te onderzoeken. In de tabel komen zowel enkelvoudige als multipolygonen voor. De platte XML van een enkelvoudig polygon uit ‘terrein’ en een multipolygon zijn te zien in figuur 9a en 9b. Hier zijn alleen de geometrische elementen weergegeven, omdat deze verschillen voor enkelvoudige en multipolygonen.

```
<POLYGONPROPERTY>
   <SDO_GTYPE>2003</SDO_GTYPE>
   <SDO_ELEM_INFO>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
   </SDO_ELEM_INFO>
   <SDO_ORDINATES>
      <SDO_ORDINATES_ITEM>159459.709</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432008.521</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159447.231</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159473.132</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159473.585</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000.169</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159473.648</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159490.159</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159482.1</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432020.726</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159467.839</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432013.12</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159459.709</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432008.521</SDO_ORDINATES_ITEM>
   </SDO_ORDINATES>
</POLYGONPROPERTY>
```

Figuur 9a De geometrie van een enkelvoudig polygon.

```
<POLYGONPROPERTY>
   <SDO_GTYPE>2003</SDO_GTYPE>
   <SDO_ELEM_INFO>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>23</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>2003</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>33</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>2003</SDO_ELEM_INFO_ITEM>
      <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
   </SDO_ELEM_INFO>
   <SDO_ORDINATES>
1→    <SDO_ORDINATES_ITEM>159562.536</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432010.179</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159565.26</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432003.328</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159566.925</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159619.485</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159618.319</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432002.273</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159619.671</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432024.717</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159604.528</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432065.549</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159595.881</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432067.764</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159583.148</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432063.436</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159596.957</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432021.742</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159562.536</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432010.179</SDO_ORDINATES_ITEM>
23→   <SDO_ORDINATES_ITEM>159600.198</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432004.01</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159598.219</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432009.947</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159608.685</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432013.435</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159610.664</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432007.498</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159600.198</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432004.01</SDO_ORDINATES_ITEM>
33→   <SDO_ORDINATES_ITEM>159596.557</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432029.969</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159590.841</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432050.761</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159599.353</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432053.101</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159605.069</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432032.309</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>159596.557</SDO_ORDINATES_ITEM>
      <SDO_ORDINATES_ITEM>432029.969</SDO_ORDINATES_ITEM>
   </SDO_ORDINATES>
</POLYGONPROPERTY>
```

Figuur 9b De geometrie van een multipolygon. In de lijst met ordinaten zijn de beginpunten van de 3 deel-polygonen aangegeven met het beginnummer gevolgd door →.

Uit de weergegeven XML blijkt dat een enkelvoudig polygon in de tabel 'terrein' 3 elementen heeft staan in ELEM_INFO, een multipolygon een veelvoud van 3, hier 9 elementen. Elke set van 3 elementen geeft informatie over een (deel-)polygon (een deel-polygon is een 'onderdeel' van een multipolygon). Elk eerste van die 3 elementen, geeft weer welk ordinaat het eerste ordinaat is van de (deel-)polygon in de ordinatenlijst `<SDO_ORDINATES>`. Voor een enkelvoudig polygon heeft dit eerste element altijd de waarde 1.

In figuur 9b is een object weergegeven dat bestaat uit 3 delen. Voor dit object geldt dat de eerste ordinaat van het eerste deel-polygon (logischerwijs) begint op positie 1 in de lijst met ordinaten, van de tweede deel-polygon vanaf positie 23 en van de derde vanaf positie 33.

Deze informatie kan gebruikt worden in de XSLTransformatie. Zo test bijvoorbeeld de volgende XSLT-code of een object een enkelvoudig of een mulipolygon (meer dan 3 elementen) is:

```
<xsl:if test="count(SDO_ELEM_INFO/SDO_ELEM_INFO_ITEM) > 3">
```

Aan de hand hiervan kunnen tags worden geplaatst die nodig zijn om een multipolygon weer te geven, bijvoorbeeld de tag `<multiPolygon>` voor het gehele object of `<polygonMember>` voor elk deel-polygon. Ook kunnen, op het moment dat de ordinaten moeten worden getransformeerd, in de ordinatenlijst de ordinaten worden geselecteerd die bij een deel-polygon worden weergegeven met de volgende XSLT-code:

```
<xsl:apply-templates
select="../../SDO_ORDINATES/SDO_ORDINATES_ITEM[position()  >=  $begin
and position() &lt; $end]"/>
```

Deze code maakt gebruik van twee variabelen (`$begin` en `$end`) die de positie in de ordinatenlijst aangeven. Deze variabelen worden ingesteld op het moment dat de ELEM_INFO per deel-polygon wordt doorlopen. In bijlage IVc is de gehele XSLT code te vinden. Met die XSLT code is het mogelijk om enkelvoudige en multipolygonen met de juiste tags en de juiste coördinaten weer te geven in GML-formaat. In bijlage IVd is het resultaat te zien van beide objecten na transformatie naar GML-formaat.

Vanwege het feit dat het dus mogelijk is ook multipolygonen te herkennen en in het juiste GML-formaat weg te schrijven, kan verwacht worden dat XSLT voldoende mogelijkheden biedt om ook allerlei andere complexe geometriën te transformeren vanuit Oracle naar GML met de hier gebruikte XML-tools.

De transformatie van geometrische elementen naar GML-geldige geometrie blijkt complexer dan van niet-geometrische elementen. Hoewel de stappen in het proces dezelfde zijn als bij de transformatie van niet-geometrische elementen (platte XML wordt getransformeerd tot GML door een XSLT-sheet toe te passen), is met name de XSLT-code een stuk ingewikkelder. Dit is te verklaren doordat de geometrie zowel in Oracle als in het GML-formaat een ingewikkeldere structuur heeft. Echter, XSLT biedt voldoende mogelijkheden hiermee om te gaan.

4.2.4 Samenvatting en kanttekeningen

In deze paragraaf is beschreven hoe vanuit een Oracle Spatial-tabel XML (en uiteindelijk GML) gegenereerd kan worden. Vervolgens kan deze XML door een XSLTransformatie worden getransformeerd naar elementen in GML-formaat. Het blijkt mogelijk dit zowel voor niet-geometrische als geometrische XML-elementen te kunnen doen met Oracle's XML-tools. Wat betreft de geometrie is aangetoond dat het mogelijk is met XSLT te kunnen herkennen watvoor geometrie een object heeft (2-D of 3-D, enkelvoudig of juist een meervoudig object)

en aan de hand daarvan elementen te kunnen structureren en weergeven in een GML-formaat, door de juiste tags te plaatsen en de coördinaten in het juiste formaat weg te schrijven.

Er is echter nog niet aangetoond dat het bovenstaande voor *alle* typen geometrie mogelijk is. Er bestaat wel het sterke vermoeden dat dit mogelijk is met de gebruikte tools en specifiek XSLT. Daarnaast is er nog geen geldig GML-document gegenereerd. De volgende paragraaf beschrijft daarom een wijze waarop een volledig geldig GML-document gegenereerd kan worden met de XML-tools die Oracle biedt. Daarbij wordt ook ingegaan op de vraag hoe extra structuurinformatie (met name objecthiërarchie) kan worden toegevoegd aan een GML-document.

4.3 Naar een geldig GML-document

Dit rapport heeft tot nu toe laten zien dat het mogelijk is vanuit een Oracle Spatial-tabel de gegevens te transformeren naar een formaat dat nagenoeg GML-geldig is. Om een geheel geldig GML-document te krijgen is het noodzakelijk nog enkele elementen toe te voegen aan de gegenereerde GML. Hierbij moet men denken aan bijvoorbeeld de begin- en eindtags van het document (header en root-element met de declaraties van de 'namespaces') en de zogenaamde bounding boxes van objecten. Hoe met de tot nu toe behaalde resultaten een geldig GML-document gegenereerd kan worden wordt hieronder besproken. Dit zijn uiteraard niet de enige mogelijkheden, maar slechts enkele mogelijke oplossingen en ideëen over oplossingen.

Daarnaast is het in bepaalde gevallen (bijvoorbeeld voor de TDN) gewenst of noodzakelijk om extra structuurinformatie (objecthiërarchie) te kunnen weergeven in een GML-document [5]. Het tweede deel van deze paragraaf geeft hiervoor een aantal oplossingsrichtingen.

4.3.1 Ontbrekende elementen toevoegen

Om een volledig geldig GML-document te krijgen moeten een aantal elementen worden toegevoegd. Daarbij kan men (naar mijn mening) gebruik maken van de volgende tools:

- Een eenvoudig programma (bijv. een shell-script) waarmee *de header en het rootelement* worden geschreven, vervolgens de XSU Command Line wordt aangeroepen om de *GML te genereren* en tot slot de *eindtags* worden geschreven. Dit is al met succes getest.
- *Oracle's Application Developer* om de '*bounding boxes*' te berekenen (zie ook het GML prototype van de TU Delft, waarbij Java-code is gebruikt om de bounding boxes te bepalen). Om vervolgens de bounding boxes in het GML-document te krijgen, lijkt het mij persoonlijk het meest geschikt om deze vóór de generatie van de platte XML toe te voegen aan de data. Dit heeft de volgende reden. Indien bij het genereren van de platte XML de bounding boxes per object (of collectie objecten) al bekend zijn (en bijvoorbeeld als element in de platte XML staan), kan dit na een (eenvoudige) XSL Transformatie worden weergegeven in het GML-document. Men kan er ook voor kiezen de bounding boxes na generatie van de XML te berekenen. Men kan hiervoor wellicht ook de Application Developer gebruiken.
- Een andere optie is te onderzoeken in hoeverre het mogelijk is met *XSLT* de bounding boxes te bepalen. Daarbij kan wellicht gebruik gemaakt worden van functies om het maximum danwel minimum ordinaat (voor de hoeken van de bounding box) uit een lijst ordinaten te bepalen.

Wellicht dat ook een aantal andere elementen nog toegevoegd moet worden aan het GML-document. Daarvoor kunnen de hierboven genoemde tools ook nuttig zijn. Ten overvloede misschien: bovenstaande ideëen zijn bij lange na niet de enige opties, maar pogen een oplossing te geven.

4.3.2 Objecthiërarchie toevoegen

Elementen die nog van belang kunnen zijn, zijn elementen in het GML-document die extra structuurinformatie, in het bijzonder objecthiërarchie, geven. Er zijn mijns inziens drie mogelijkheden om deze objecthiërarchie automatisch te kunnen genereren in het GML document, gebruik makend van Oracle's XML-tools.

De eerste is het toevoegen van een soort meta-tabel aan de database, die per tabel weergeeft tot welke verzameling klassen (voor een verzameling klassen [5]) de betreffende tabel behoort. Dit geldt alleen indien een tabel slechts één klasse van objecten beslaat, bijvoorbeeld wegen of water. Indien de XSU command line gebruikt wordt, moet deze meta-tabel in de SQL-queries worden betrokken om de informatie ook in de platte XML te krijgen. Dit maakt de SQL-queries complexer.

De tweede mogelijkheid is door (al dan niet automatisch) aan elk object als attribuut de klasse waaroe het behoort toe te voegen, bijvoorbeeld via een view. Bij het gebruik van de XSU command line wordt dan voor elk object in de platte XML de klasse toegevoegd. Nadeel van deze methode is dat gegevens (namelijk de objecthiërarchie) redundant worden opgeslagen. De SQL-queries blijven echter eenvoudig.

Voor beide bovenstaande methoden geldt dat het uitgangspunt is om de structuurinformatie in de platte XML te krijgen, voordat de XSL Transformatie plaats vindt. Zodoende kan namelijk met behulp van een XSL Transformatie vanuit de platte XML de objecthiërarchie in het GML-document geplaatst worden. In onderstaand kader wordt dit technisch nader uitgewerkt.

Van belang is dat XSLT voldoende mogelijkheden biedt om overtollige elementen (structuurinformatie) weg te filteren uit de platte XML. Bij de tot nu toe beschreven methoden, kan men zich voorstellen dat bij elk individueel object *in de platte XML een element wordt geplaatst dat weergeeft tot welke klasse de verzameling objecten behoort. Echter, de klasse moet in het GML-document maar* één keer *worden weergegeven voor de gehele verzameling. Het is nu als volgt mogelijk om de overbodige elementen weg te filteren met XSLT. Hierbij wordt XSLT (met XPath) gebruikt om de eerste en laatste 'instance' van een collectie objecten te indentificeren. Er worden hieronder 3 stappen onderscheiden, om uiteindelijk de klasse voor de gehele collectie één keer weer te geven.*

1) Men kan de eerste keer dat men een geografisch object van een bepaald type ('instance') tegenkomt, oftewel aan het begin van de collectie, in de platte XML de klasse weergeven. Concreet: de begintag van het element voor de klasse waarvan de objecten lid zijn, wordt weergegeven, bijvoorbeeld `<tdn:RuimtelijkeObjecten>`.

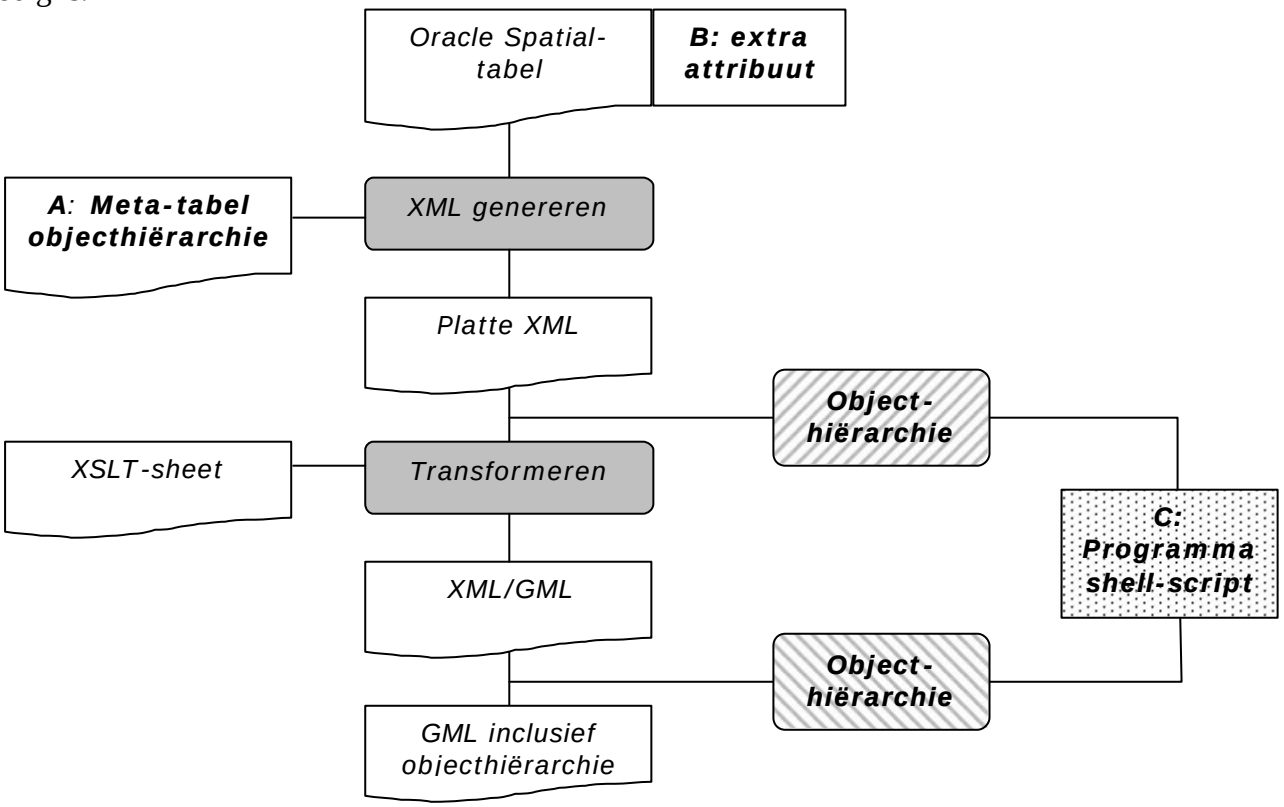
2) Vervolgens wordt voor elk volgend object het 'klasse-element' niet weergegeven, zodra dit gevonden wordt door de XSLT-sheet.

3) Aan het eind van de collectie, nadat het laatste object van de betreffende klasse op deze wijze is weergegeven in GML, wordt de eindtag weergegeven: `</tdn:RuimtelijkeObjecten>`. *Dit betekent dat of het einde bereikt is of een ander type objecten kan worden weergegeven.*

Beide bovenstaande methoden gaan ervan uit om de objecthiërarchie als gegevens toe te voegen voordat de platte XML gegenereerd wordt. Een derde mogelijkheid is om dit bij de transformatie van de platte XML naar XML/GML te doen. Per collectie objecten kan een programma (bijvoorbeeld een 'shell-script' die het gehele generatie-proces uitvoert), voordat de collectie wordt getransformeerd naar GML, de openingstag van de klasse weergeven en aan het eind van de transformatie de eindtag van de klasse. Zo wordt de verzamelingsstructuur niet in de database opgeslagen, maar feitelijk in een extern programma

dat de verzamelingsstructuur toevoegt. Dit houdt de queries eenvoudig en vereist geen extra opslagruimte in de database. Daarbij komt dat de XSLT-code niet hoeft te worden aangepast. Een nadeel kan zijn dat de verzamelingsstructuur nu extern 'beheerd' wordt, zodat deze ook niet meer uit de database te halen is. Wijzigingen in de verzamelingsstructuur van de database moeten daarom ook worden bijgehouden in het programma dat de verzamelingsstructuur toevoegt.

In figuur 10 zijn de geopperde methoden weergegeven in het generatie-proces als A (een meta-tabel), B (een extra attribuut per object) en C (verzamelingsstructuur in een extern programma of shell-script). Voor dit laatstgenoemde programma is in het schema alleen weergegeven op welke punten dit programma de verzamelingsstructuur moet toevoegen aan het uiteindelijke GML-document. Overige functies (bijvoorbeeld de eerder genoemde toevoeging van de header en het rootelement) worden in dit schema niet weergegeven. Voor methode A en B geldt dat direct na de transformatie de verzamelingsstructuur al is toegevoegd (bij 'XML/GML' in het schema). Daar staat tegenover dat extra XSLT-code nodig is.



Figuur 10 De plaats van drie methoden (A, B en C) om objecthiërarchie toe te voegen in het generatieproces

4.4 Het inlezen van GML

De voorgaande paragrafen hebben laten zien hoe GML gegenereerd kan worden met behulp van Oracle's XML-tools. Het is van belang ook GML te kunnen inlezen. Hoe dit kan met de eerder beschreven tools (§ 3.3) is onderwerp van deze paragraaf.

XSU (XML SQL Utility) biedt de mogelijkheid om XML in te lezen in een bestaande tabel, mits de structuur van het XML-document overeenkomt met de doel-tabel. GML is een op XML gebaseerde codering (§ 2.3). Wanneer men nu in staat is om een GML-document zodanig te bewerken dat het in de juiste XML-structuur staat, zou het daarom mogelijk moeten zijn GML in te lezen met XSU.

Om GML te kunnen inlezen moet men dus 2 stappen onderscheiden (vergelijk dit met het genereren van XML/GML):
1) GML bewerken tot de juiste structuur in XML-formaat en
2) deze XML inlezen in de doel-tabel.

Bij de eerste stap lijkt XSLT de aangewezen technologie om GML te transformeren, omdat de XSLTransformatie in staat is complexe transformaties uit te voeren en daarnaast door XSU ondersteund wordt. Bij de transformatie zullen de niet-geometrische elementen en attributen getransformeerd moeten worden naar elementen, die overeenkomen met de structuur van de doel-tabel (bijvoorbeeld: `<tdn:tdn_code>3199</tdn:tdn_code>` in het GML-document moet `<SHP_TDNCODE>3199</SHP_TDNCODE>` worden in het in te lezen XML-document, omdat in de doel-tabel de kolom `SHP_TDNCODE` voorkomt). De geometrische elementen moeten zodanig worden getransformeerd, dat Oracle met de SQL-instructie een SDO-geometry gaat aanmaken.

In het verrichte onderzoek is geen XSLTransformatie opgesteld, omdat eerst onderzocht is of het mogelijk is de XML in te lezen (stap 2). Daarbij is de XSU command line gebruikt (§ 3.3). De eerste gedachte was het in te lezen XML-document qua structuur exact te laten overeenkomen met een gegenereerd XML-document (platte XML met SDO-geometry). Immers, bij generatie worden de gegevens uit de doel-tabel geextraheerd. Daarom is met de XSU command line platte XML gegenereerd van de tabel 'waterdeel' (figuur 4 in § 4.2.1). Deze XML werd opgeslagen in het document "input.xml" en met het volgende commando weer ingelezen:

```
java OracleXML putXML –user "naam/wachtwoord" –filename "input.xml"
"waterdeel"
```

Er werd nu echter een foutmelding gegeven, waarbij het inlezen werd afgebroken. Er bleek geen enkele rij gevuld te zijn. Blijkbaar kan de XSU command line niet gebruikt worden om vanuit een Oracle Spatial-tabel gegenereerde XML in hetzelfde formaat direct weer in te lezen. De verwachting was dat dit te wijten was aan de geometrie, die niet in het juiste formaat staat voor XSU, om een juiste SQL-instructie te maken. De SQL-instructie om een ruimtelijk object toe te voegen, moet eruit zien als in figuur 11.

```
insert into waterdeel values
(
mdsys.sdo_geometry(2003,null,null,mdsys.sdo_elem_info_array(1,1003,1)
,mdsys.sdo_ordinate_array(119637.061, 576598.127, 119643.113,
576600.006, 119647.079, 576600.632, 119651.67, 576600.84, 119655.844,
576600.006, 119661.27, 576600.84, 119666.069, 576600.423, 119668.782,
576597.292, 119670.869, 576592.908, 119670.243, 576587.064,
119665.235, 576581.429, 119660.226, 576579.132, 119655.218,
576579.967, 119649.792, 576582.472, 119644.366, 576586.856,
119640.609, 576587.482, 119635.601, 576587.899, 119632.47,
576590.404, 119632.888, 576594.579, 119637.061, 576598.127)),
6300099,
06 Jul 2001 08:28:02,
31 Dec 2999 23:59:59,
Verbinding,
Zee,
&gt;6m,
Onbekend,
Nee,
Zoet,
Openbaar,
Overig,
```

```
Stilstaand,
In gebruik,
Onbekend,
0,
9999
);
```

Figuur 11 SQL-instructie om rij in te voegen in 'waterdeel' met SDO-geometry gearceerd

Oracle geeft echter in de eigen documentatie aan dat complexe typen moeten kunnen worden ingelezen met de XSU command line [3]. Als het XML-element `<GEOMETRY>` overeenkomt met de kolom die in de doel-tabel de geometrie bevat als SDO-geometry (de kolom `GEOMETRY` in 'waterdeel') zou dan dus ook de geometrie moeten kunnen worden ingelezen. Deze geometrie moet overeenkomen met de SQL-instructie om een SDO- geometry in te lezen (gearceerde deel in figuur 11). In het XML-document ziet dat er als in figuur 12 uit.

```
<GEOMETRY>mdsys.sdo_geometry(2003,null,null,mdsys.sdo_elem_info_array
(1,1003,1),mdsys.sdo_ordinate_array(119637.061, 576598.127,
119643.113, 576600.006, 119647.079, 576600.632, 119651.67, 576600.84,
119655.844, 576600.006, 119661.27, 576600.84, 119666.069, 576600.423,
119668.782, 576597.292, 119670.869, 576592.908, 119670.243,
576587.064, 119665.235, 576581.429, 119660.226, 576579.132,
119655.218, 576579.967, 119649.792, 576582.472, 119644.366,
576586.856, 119640.609, 576587.482, 119635.601, 576587.899,
119632.47, 576590.404, 119632.888, 576594.579, 119637.061,
576598.127))</GEOMETRY>
```

Figuur 12 SDO- geometry in een XML-element

Wanneer een dergelijk element wordt ingevoegd met `putXML` verschijnt nu de mededeling "`succesfully inserted 1 rows into waterdeel`". Echter, bij het opvragen van deze rij in zowel XML als in SQL blijkt dat de geometrie allemaal "null-waarden" bevat. De niet-geometrische waarden zijn daarentegen wel correct ingelezen. Ook een andere syntax (bijvoorbeeld met hoofdletters of zonder `mdsys.`) in het XML-element resulteert in een mededeling "`succesfully inserted 1 rows into waterdeel`", met desondanks null-waarden in de geometrie.

Uit het bovenstaande kan geconcludeerd worden dat XSU, wanneer gebruik wordt gemaakt van de command line, goed in staat is om niet-geometrische elementen in te lezen. Geometrische elementen daarentegen leveren problemen op, waarbij Oracle aangeeft dat toch succesvol is ingelezen. Dit lijkt te wijzen op een fout in XSU. Dir probleem is bij Oracel aangemeld als bug en inmiddels ook als bug erkend. Vanwege dit resultaat is niet verder ingegaan op een XSLTransformatie van GML naar het juiste XML-formaat, immers wat het juiste XML-formaat moet zijn is niet duidelijk.

5 Conclusies en aanbevelingen

Extensible Markup Language (XML) kan gebruikt worden om in een tekstformaat gegevens te structureren. Hiertoe wordt een verzameling tags en attributen ingezet, die naar gelang de toepassing kan worden ontworpen. XML maakt het mogelijk eenvoudig gegevens uit te wisselen. Aan XML zijn een aantal technologieën verbonden, die onder andere gebruikt kunnen worden voor de definitie en toetsing van de structuur van bepaalde XML-documenten (DTD en XML Schema) of transformatie van XML bestanden naar een ander (XML-)formaat (XSLT).

Geography Markup Language (GML) is een op XML gebaseerde codering voor het transport en de opslag van geografische informatie, die zowel de ruimtelijke als niet-ruimtelijke eigenschappen van geografische objecten beslaat. GML maakt hiervoor gebruik van een set tags en attributen die door het OpenGis Consortium zijn gedefinieerd.

Oracle 9i ondersteunt XML, wat concreet betekent dat er een aantal componenten aanwezig zijn om XML-documenten op te slaan, te bevragen, te presenteren en te manipuleren. Hiervoor worden onder andere XML Developer Kits aangeboden met XML functies.

Het genereren van XML-documenten vanuit Oracle-tabellen verloopt goed, zolang er geen overerving wordt gebruikt in de tabel. Tabellen met overerving worden niet volledig weergegeven in XML. Bij het genereren is gebruik gemaakt van een Java command line, die een onderdeel is van één van Oracle's XML-tools, namelijk XSU (XML SQL Utility). Of andere XML-functionaliteit (bijvoorbeeld met `DBMS_XMLGEN`) dezelfde problemen kent, zal onderzocht moeten worden. Wellicht dat de genoemde problemen in een vernieuwde versie van Oracle zijn verholpen.

Het inlezen van XML verloopt ook goed, zolang men zich beperkt tot eenvoudige datastructuren. Bij geneste typen in het XML-document ontstaan problemen. Dit heeft belangrijke gevolgen voor het inlezen van XML en daarom ook GML.

Oracle's XML functionaliteit biedt voldoende mogelijkheden om GML te genereren vanuit Oracle Spatial-tabellen. Met name XSU is hierbij nuttig. Hiermee kan XML gegenereerd worden ('platte XML') die vervolgens wordt omgezet in GML met behulp van een XSLTransformatie. De transformatie van geometrische elementen is complexer dan niet-geometrische elementen, mede vanwege de kaartlagen (Feature Collection) die voor kunnen komen. XSLT lijkt echter krachtig genoeg om ook ingewikkelde objecten te kunnen transformeren vanuit platte XML naar GML. Voor multipolygonen is het reeds gelukt om GML te genereren met een XSLTransformatie. Of het daadwerkelijk voor alle complexe objecten lukt, zal nog moeten blijken uit verder onderzoek.

In dit onderzoek is nog geen geldig GML-document gegenereerd. Voor een geldig GML-document ontbreken nog een aantal elementen. Hoe deze kunnen worden toegevoegd, zal nader onderzocht moeten worden. Een programma dat deze elementen tijdens het generatie-proces toevoegt, bijvoorbeeld een shell-script, is één van de mogelijkheden.

Het inlezen van GML levert problemen op bij de geometrie. De niet-geometrische elementen worden correct ingelezen, maar voor de geometrie worden alleen null-waarden ingelezen. Dit lijkt te wijzen op een fout in XSU. Het is echter wel van belang om GML te kunnen inlezen. Daarom zal er nog geprobeerd moeten worden om het probleem in XSU op te lossen of een andere oplossing te zoeken om GML in te lezen.

De slotconclusie moet zijn dat Oracle ondersteuning biedt om GML te genereren. Daarbij is het ook mogelijk om complexere objecten goed weer te geven in GML. De XML SQL Utility (XSU) is hierbij erg nuttig. Voor het inlezen van GML schiet XSU (nog) tekort.

I Gebruik van de XSU command line

UNIX classpath

Op het moment van schrijven (maart 2002) is, althans op de GISt-server van de TU Delft waarop Oracle draait, XSU geïnstalleerd. Daarom is het voldoende om de volgende regels toe te voegen aan het Java-classpath in Unix:

```
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/rdbms/jlib/xsu12.jar
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/lib/XMLparserv2.jar
```

Voor verdere details omtrent het installeren en gebruik van XSU wordt verwezen naar Oracle's Application Developer's Guide – XML [3].

Het gebruik van de XSU command Line

Voorbeeld syntax:

```
java OracleXML getXML –user "naam/wachtwoord" "select * from tabelnaam"
```

Bij de command line kunnen onderstaande opties ingesteld worden (onderstaande tekst is letterlijk overgenomen uit Oracle9i Application Developer's Guide – XML [3])

XSU's OracleXML getXML Options

- **-user "<username>/<password>"** Specifies the user name and password to connect to the database. If this is not specified, the user defaults to scott/tiger. Note that he connect string is also being specified, the user name and password can be specified as part of the connect string.
- **-conn "<JDBC_connect_string>"** Specifies the JDBC database connect string. By default the connect string is: "jdbc:oracle:oci8:@"
- **-withDTD** Instructs the XSU to generate the DTD along with the XML document.
- **-rowsetTag "<tag_name>"** Specifies rowset tag (the tag that encloses all the XML elements corresponding to the records returned by the query). The default rowset tag is ROWSET. Specifying an empty string for the rowset tells the XSU to completely omit the rowset element.
- **-rowTag "<tag_name>"** Specifies the row tag (the tag used to enclose the data corresponding to a database row). The default row tag is ROW. Specifying an empty string for the row tag tells the XSU to completely omit the row tag.
- **-rowIdAttr "<row_id-attribute-name>"** Names the attribute of the ROW element keeping track of the cardinality of the rows. By default this attribute is called num. Specifying an empty string (i.e. "") as the rowID attribute will tell the XSU to omit the attribute.
- **-rowIdColumn "<row Id column name>"** Specifies that the value of one of the scalar columns from the query should be used as the value of the rowID attribute.
- **-collectionIdAttr "<collection id attribute name>"** Names the attribute of an XML list element keeping track of the cardinality of the elements of the list (Note: the generated XML lists correspond to either a cursor query, or collection). Specifying an empty string (i.e. "") as the rowID attribute will tell the XSU to omit the attribute.
- **-useNullAttrId** Tells the XSU to use the attribute NULL (TRUE/FALSE) to indicate the nullness of an element.

- **-styleSheet "<stylesheet URI>"** Specifies the stylesheet in the XML PI (Processing Instruction).
- **-stylesheetType "<stylesheet type>"** Specifies the stylesheet type in the XML PI (Processing Instruction).
- **-errorTag "<error tag name>"** Secifies the error tag -- the tag to enclose error messages which are formated into XML.
- **-raiseNoRowsException** Tells the XSU to raise an exception if no rows are returned.
- **-maxRows "<maximum number of rows>"** Specifies the maximum number of rows to be retrieved and converted to XML.
- **-skipRows "<number of rows to skip>"** Specifies the number of rows to be skipped.
- **-encoding "<encoding name>"** Specifies the characterset encoding of the generated XML.
- **-dateFormat "<date format>"** Specifies the date format for the date values in the XML document.
- **-fileName "<SQL query fileName>" | <sql query>** Specifies the file name which contains the query or specify the query itself.

II XSU command line en overerving

Om te testen of XSU kan omgaan met tabellen met overerving, is de volgende SQL uitgevoerd in Oracle, om een tabel te creëren met overerving.

```
drop table persons;
drop type student;
drop type person;
--
-- Maak een type voor personen.
--
create or replace type person
as object
(
   naam varchar(40),
   leeftijd number
) not final;

/

--
-- Maak een type voor studenten; dit is een subtype
-- van persoon.
--
create type student under person
(
   startjaar number
);

/

--
-- Maak een tabel met personen (dus ook studenten).
--
create table persons of person;

--
-- Voeg een persoon en een student toe.
--
insert into persons
    values (person('wilko',33));

insert into persons
    values (student('thijs',23,97));
```

In Oracle geeft de SQL-query als resultaat:

```
VALUE(P)(NAAM, LEEFTIJD)
------------------------------------------------------------
PERSON('wilko', 33)
STUDENT('thijs', 23, 97)
```

Met de XSU command line wordt vervolgens XML gegenereerd. Hiervoor is het volgende commando gebruikt:

```
java OracleXML getXML -user "naam/wachtwoord" "select * from persons"
```

Dit levert het onderstaande resultaat in XML op, waarbij te zien is dat er geen waarden behorend bij een ‘student’ worden weergegeven. Overerving is dus niet weergegeven in XML.

```
<ROWSET>
      <ROW num="1">
         <WAARDE>
            <NAAM>wilko</NAAM>
            <LEEFTIJD>33</LEEFTIJD>
         </WAARDE>
      </ROW>
      <ROW num="2">
         <WAARDE>
            <NAAM>thijs</NAAM>
            <LEEFTIJD>23</LEEFTIJD>
         </WAARDE>
      </ROW>
   </ROWSET>
```

Ook in een gegeneererd XML Schema komen de attributen voor ‘student’ niet voor. Dit XML Schema is hieronder weergegeven.

```
<schema targetNamespace="http://ns.oracle.com/xdb/THIJS"
xmlns="http://www.w3.org/2000/10/XMLSchema" xmlns:THIJ
S="http://ns.oracle.com/xdb/THIJS">
      <complexType name="PERSON">
         <sequence>
            <element name="NAAM" type="string" nullable="true"
minOccurs="0"/>
            <element name="LEEFTIJD" type="double" nullable="true"
minOccurs="0"/>
         </sequence>
      </complexType>
   </schema>
   <xsd:schema xmlns:THIJS="http://ns.oracle.com/xdb/THIJS"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
      <xsd:import targetNamespace="http://ns.oracle.com/xdb/THIJS"
schemaLocation="#/DOCUMENT/xsd:schema[@targetNa
mespace=&apos;http://ns.oracle.com/xdb/THIJS&apos;]"/>
      <xsd:element name="ROWSET">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="ROW" minOccurs="0"
maxOccurs="unbounded">
                  <xsd:complexType>
                     <xsd:sequence>
                        <xsd:element name="WAARDE"
type="THIJS:PERSON" nullable="true" minOccurs="0"/>
                     </xsd:sequence>
                     <xsd:attribute name="num" type="xsd:integer"/>
                  </xsd:complexType>
               </xsd:element>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
```

De bovenstaande problemen zijn bij Oracle als bug aangemerkt en inmiddels ook zo erkend.

III Naar GML voor tabel 'waterdeel'

a Tabelstructuur 'waterdeel'

Naam attribuut	Type
GEOMETRY	MDSYS.SDO_GEOMETRY
SHP_OID	NUMBER
SHP_TOP10_ID	NUMBER
SHP_BEGINDATUM	VARCHAR2(256)
SHP_EINDDATUM	VARCHAR2(256)
SHP_TYPE	VARCHAR2(256)
SHP_WATERTYPE	VARCHAR2(256)
SHP_BREEDTEKLA	VARCHAR2(256)
SHP_BREEDTE	VARCHAR2(256)
SHP_HOOFDAFWAT	VARCHAR2(256)
SHP_ZOUTGEHALT	VARCHAR2(256)
SHP_TOEGANKELI	VARCHAR2(256)
SHP_VOORKOMEN	VARCHAR2(256)
SHP_GEBRUIK	VARCHAR2(256)
SHP_STROOMRICH	VARCHAR2(256)
SHP_STATUS	VARCHAR2(256)
SHP_NAAM	VARCHAR2(256)
SHP_HOOGTENIVE	NUMBER
SHP_TDNCODE	NUMBER

b Gegenereerde XML 'waterdeel'

```
<ROW num="10">
      <GEOMETRY>
         <SDO_GTYPE>2003</SDO_GTYPE>
         <SDO_ELEM_INFO>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
         </SDO_ELEM_INFO>
         <SDO_ORDINATES>
            <SDO_ORDINATES_ITEM>119876.336</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576934.906</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119881.605</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576937.761</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119887.313</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576938.42</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119891.923</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576937.541</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119896.094</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576933.369</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119896.972</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576929.417</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119894.777</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576927.66</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119890.167</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576926.342</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119879.629</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576927.221</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>119876.555</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>576929.417</SDO_ORDINATES_ITEM>
```

```
                <SDO_ORDINATES_ITEM>119874.579</SDO_ORDINATES_ITEM>
                <SDO_ORDINATES_ITEM>576932.71</SDO_ORDINATES_ITEM>
                <SDO_ORDINATES_ITEM>119876.336</SDO_ORDINATES_ITEM>
                <SDO_ORDINATES_ITEM>576934.906</SDO_ORDINATES_ITEM>
             </SDO_ORDINATES>
          </GEOMETRY>
          <SHP_OID>6300005</SHP_OID>
          <SHP_TOP10_ID>6300005</SHP_TOP10_ID>
          <SHP_BEGINDATUM>06 Jul 2001 08:28:02</SHP_BEGINDATUM>
          <SHP_EINDDATUM>31 Dec 2999 23:59:59</SHP_EINDDATUM>
          <SHP_TYPE>Verbinding</SHP_TYPE>
          <SHP_WATERTYPE>Meer, plas, ven vijver</SHP_WATERTYPE>
          <SHP_BREEDTEKLA>Onbekend</SHP_BREEDTEKLA>
          <SHP_BREEDTE>Onbekend</SHP_BREEDTE>
          <SHP_HOOFDAFWAT>Nee</SHP_HOOFDAFWAT>
          <SHP_ZOUTGEHALT>Zoet</SHP_ZOUTGEHALT>
          <SHP_TOEGANKELI>Openbaar</SHP_TOEGANKELI>
          <SHP_VOORKOMEN>Overig</SHP_VOORKOMEN>
          <SHP_GEBRUIK>Overig</SHP_GEBRUIK>
          <SHP_STROOMRICH>Stilstaand</SHP_STROOMRICH>
          <SHP_STATUS>In gebruik</SHP_STATUS>
          <SHP_NAAM>Onbekend</SHP_NAAM>
          <SHP_HOOGTENIVE>0</SHP_HOOGTENIVE>
          <SHP_TDNCODE>6113</SHP_TDNCODE>
   </ROW>
```

c XSLT stylesheet van XML naar GML voor ‘waterdeel’

```
<xsl:stylesheet version="1.0" xmlns:tdn="http://www.gdmc.nl/tdn"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:gml="http://www.opengis.net/gml"  exclude-result-prefixes = "gml
tdn">
<xsl:output method="html"/>

 <xsl:template match="ROW">

  <xsl:element name="tdn:waterdeel">
   <xsl:attribute name="fid">TOP10.<xsl:value-of
select="SHP_OID"/></xsl:attribute>
   <xsl:apply-templates/>

  </xsl:element>
 </xsl:template>

 <xsl:template match="SDO_ORDINATES_ITEM"> <!-- zoek elk element
SDO_ORDINATES_ITEM op, dit is een ordinaat -->
   <xsl:apply-templates/> <!-- plaats het ordinaat in het document -->
   <xsl:if test="position() > 0"> <!-- na het eerste ordinaat moeten de
ordinaten gescheiden worden door komma’s, de gehele coördinaten door
spaties -->
    <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 2"> <!--object
is 2-dimensionaal -->
  <xsl:if test="round(0.5 * (position())) = 0.5 * (position())">,0.0
</xsl:if>
<!-- plaats na elk 2e ordinaat (y) een komma en de z-waarde op 0.0-->
<xsl:if test="round(0.5 * (position())) != 0.5 * (position())">,</xsl:if>
    <!-- plaats na elk 1e ordinaat (x) een komma-->
   </xsl:if>
<!-- einde 2-dimensionale coördinaten -->
```

```
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 3"> <!--object is
3-dimensionaal -->

<xsl:if test="round(((position())) div 3) != ((position())) div
3">,</xsl:if> <!-- elk x,y,z-ordinaat wordt gescheiden door een komma-->
<xsl:if test="round(((position())) div 3) = ((position())) div
3"><xsl:text> </xsl:text> <!-- elk geheel x,y,z-cöordinaat wordt van andere
coördinaten gescheiden door een spatie-->
</xsl:if>
    </xsl:if>

  </xsl:if> <!-- einde if, voor position groter dan 0 -->

</xsl:template>


<xsl:template match="GEOMETRY"> <!-- zoek GEOMETRY op in XML-document en
plaats element gml:geometryProperty -->
 <xsl:element name="gml:geometryProperty">

  <xsl:if test="substring(string(SDO_GTYPE),4,1) = 3"> <!-- object is een
polygon als 4e cijfer in waarde van SDO_GTYPE 3 is, plaats dan elementen
voor polygon -->
   <xsl:element name="gml:Polygon">
    <xsl:attribute name="srsName">EPSG:7048</xsl:attribute>
    <xsl:element name="gml:outerBoundaryIs">
     <xsl:element name="gml:LinearRing">
      <xsl:element name="gml:coordinates">
       <xsl:apply-templates/> <!-- plaats alle waardes en eventuele
subelementen en –attributen van GEOMETRY in het element gml:coordinates.
Eventuele andere transformaties worden op een andere plaats in de XSLT-code
uitgevoerd. Sluit alle geopende elementen -->
      </xsl:element>
     </xsl:element>
    </xsl:element>
   </xsl:element>
  </xsl:if>

 </xsl:element>
</xsl:template>

 <xsl:template match="SHP_OID">
 </xsl:template>

 <xsl:template match="SDO_GTYPE">
 </xsl:template>

 <xsl:template match="SDO_ELEM_INFO">
 </xsl:template>

 <xsl:template match="SDO_ELEM_INFO_ITEM">
 </xsl:template>

 <xsl:template match="SHP_TOP10_ID">
  <xsl:element name="tdn:top10_id"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_BEGINDATUM">
  <xsl:element name="tdn:begindatum"><xsl:apply-templates/></xsl:element>
 </xsl:template>
```

```
 <xsl:template match="SHP_EINDDATUM">
  <xsl:element name="tdn:einddatum"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_TYPE">
   <xsl:element name="tdn:type"><xsl:apply-templates/></xsl:element>
 </xsl:template>

<xsl:template match="SHP_WATERTYPE">
   <xsl:element name="tdn:watertype"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_BREEDTEKLA">
   <xsl:element name="tdn:breedteklasse"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_BREEDTE">
   <xsl:element name="tdn:breedte"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_HOOFDAFWAT">
   <xsl:element name="tdn:hoofdafwatering"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_ZOUTGEHALT">
   <xsl:element name="tdn:zoutgehalte"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_TOEGANKELI">
   <xsl:element name="tdn:toegankelijkheid"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_VOORKOMEN">
   <xsl:element name="tdn:fysiek_voorkomen"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_GEBRUIK">
   <xsl:element name="tdn:gebruik"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_STROOMRICH">
   <xsl:element name="tdn:stroomrichting"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_STATUS">
   <xsl:element name="tdn:status"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_NAAM">
   <xsl:element name="tdn:naam"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="SHP_HOOGTENIVE">
   <xsl:element name="tdn:hoogteniveau"><xsl:apply-
templates/></xsl:element>
 </xsl:template>
```

```
 <xsl:template match="SHP_TDNCODE">
   <xsl:element name="tdn:tdncode"><xsl:apply-templates/></xsl:element>
 </xsl:template>

</xsl:stylesheet>
```

IV Naar GML voor tabel 'terrein' (multipolygons)

De tabel 'terrein' onderscheidt zich van 'waterdeel'doordat 'terrein'ook multipolygons bevat. In deze bijlage wordt de tabel beschreven, de gegenereerde XML van een multipolygon gegeven en de XSLT code om (eventueel voorkomende multi-)polygons te transformeren van XML naar GML. Tenslotte wordt het reultaat weergegeven voor een enkelvoudige en een multipolygon uit de tabel 'terrein'.

a Tabelstructuur 'terrein'

```
Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 OID                                                NUMBER(12)
 TOP10_ID                                           NUMBER(12)
 BEGINDATUM                                         VARCHAR2(20)
 EINDDATUM                                          VARCHAR2(20)
 BRONTYPE                                           VARCHAR2(32)
 BRONBESCHRIJVING                                   VARCHAR2(32)
 NAUWKEURIGHEID                                     VARCHAR2(32)
 ACTUALITEIT                                        VARCHAR2(20)
 LANDGEBRUIK                                        VARCHAR2(32)
 FYSIEK_VOORKOMEN                                   VARCHAR2(32)
 TOEGANKELIJKHEID                                   VARCHAR2(32)
 VOORKOMEN                                          VARCHAR2(32)
 NAAM                                               VARCHAR2(32)
 HOOGTENIVEAU                                       NUMBER(12)
 POLYGONPROPERTY                                    MDSYS.SDO_GEOMETRY
 TDNCODE                                            NUMBER(12)
```

b Gegenereerde XML 'terrein'

```
<?xml version = '1.0'?>
<ROWSET>
   <ROW num="1">
      <OID>5100295</OID>
      <TOP10_ID>5100295</TOP10_ID>
      <BEGINDATUM>24 JUN 2001 16:03:13</BEGINDATUM>
      <LANDGEBRUIK>Overig</LANDGEBRUIK>
      <FYSIEK_VOORKOMEN>Overig</FYSIEK_VOORKOMEN>
      <TOEGANKELIJKHEID>Openbaar</TOEGANKELIJKHEID>
      <VOORKOMEN>Overig</VOORKOMEN>
      <NAAM>Onbekend</NAAM>
      <HOOGTENIVEAU>0</HOOGTENIVEAU>
      <POLYGONPROPERTY>
         <SDO_GTYPE>2003</SDO_GTYPE>
         <SDO_ELEM_INFO>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
         </SDO_ELEM_INFO>
         <SDO_ORDINATES>
            <SDO_ORDINATES_ITEM>159459.709</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432008.521</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159447.231</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
```

```
            <SDO_ORDINATES_ITEM>159473.132</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159473.585</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000.169</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159473.648</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159490.159</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159482.1</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432020.726</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159467.839</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432013.12</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159459.709</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432008.521</SDO_ORDINATES_ITEM>
         </SDO_ORDINATES>
      </POLYGONPROPERTY>
      <TDNCODE>5263</TDNCODE>
   </ROW>
   <ROW num="2">
      <OID>5100159</OID>
      <TOP10_ID>5100159</TOP10_ID>
      <BEGINDATUM>24 JUN 2001 16:03:13</BEGINDATUM>
      <LANDGEBRUIK>Overig</LANDGEBRUIK>
      <FYSIEK_VOORKOMEN>Overig</FYSIEK_VOORKOMEN>
      <TOEGANKELIJKHEID>Openbaar</TOEGANKELIJKHEID>
      <VOORKOMEN>Overig</VOORKOMEN>
      <NAAM>Onbekend</NAAM>
      <HOOGTENIVEAU>0</HOOGTENIVEAU>
      <POLYGONPROPERTY>
         <SDO_GTYPE>2003</SDO_GTYPE>
         <SDO_ELEM_INFO>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>23</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>2003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>33</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>2003</SDO_ELEM_INFO_ITEM>
            <SDO_ELEM_INFO_ITEM>1</SDO_ELEM_INFO_ITEM>
         </SDO_ELEM_INFO>
         <SDO_ORDINATES>
            <SDO_ORDINATES_ITEM>159562.536</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432010.179</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159565.26</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432003.328</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159566.925</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159619.485</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432000</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159618.319</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432002.273</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159619.671</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432024.717</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159604.528</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432065.549</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159595.881</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432067.764</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159583.148</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432063.436</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159596.957</SDO_ORDINATES_ITEM>
```

```
            <SDO_ORDINATES_ITEM>432021.742</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159562.536</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432010.179</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159600.198</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432004.01</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159598.219</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432009.947</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159608.685</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432013.435</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159610.664</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432007.498</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159600.198</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432004.01</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159596.557</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432029.969</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159590.841</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432050.761</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159599.353</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432053.101</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159605.069</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432032.309</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>159596.557</SDO_ORDINATES_ITEM>
            <SDO_ORDINATES_ITEM>432029.969</SDO_ORDINATES_ITEM>
         </SDO_ORDINATES>
      </POLYGONPROPERTY>
      <TDNCODE>5263</TDNCODE>
   </ROW>
</ROWSET>
```

c XSLT stylesheet van XML naar GML voor 'terrein'

```
<xsl:stylesheet version="1.0" xmlns:tdn="http://www.gdmc.nl/tdn"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:gml="http://www.opengis.net/gml"  exclude-result-prefixes = "gml
tdn">
<xsl:output method="html"/>

<!-- map 'ROW' to 'tdn:terrein' as a row's root element and assign value to
attribute 'fid' -->
 <xsl:template match="ROW">
  <xsl:element name="tdn:terrein">
   <xsl:attribute name="fid">TOP10.<xsl:value-of
select="OID"/></xsl:attribute>
   <xsl:apply-templates/>
  </xsl:element>
  <!-- insert blank row -->
  <xsl:text>

</xsl:text>
</xsl:template>

 <!-- print ordinates to coordinates for multipolygons, -lines, or -points
-->
 <xsl:template match="SDO_ORDINATES_ITEM">
  <xsl:apply-templates />
  <xsl:if test="position() > 0">
   <!-- check dimension of data; for 2-dimensional data, print 0 for Z;
seperate X,Y,Z by comma's, coordinates by spaces -->
```

```
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 2"><xsl:if
test="round(0.5 * (position())) = 0.5 * (position())">,0.0 </xsl:if><xsl:if
test="round(0.5 * (position())) != 0.5 * (position())">,</xsl:if></xsl:if>
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 3"><xsl:if
test="round(((position())) div 3) != ((position())) div
3">,</xsl:if><xsl:if test="round(((position())) div 3) = ((position())) div
3"><xsl:text> </xsl:text></xsl:if></xsl:if>
  </xsl:if>
 </xsl:template>

 <!-- print ordinates to coordinates for multipolygons, -lines, or -points
-->
 <xsl:template name="ORDINATESMULTI">

  <!-- set default values for begin and end of range of ordinates -->
  <xsl:param name="end" select="last()"/>
  <xsl:param name="begin" select="1"/>

  <!-- select ordinates in xml-ordinates-array that belong to 1
(polygon)member -->
  <xsl:apply-templates
select="../../SDO_ORDINATES/SDO_ORDINATES_ITEM[position() >= $begin and
position() &lt; $end]"/>

  <xsl:if test="position() > 0 and position != last()">
   <!-- check dimension of data; for 2-dimensional data, print 0 for Z;
seperate X,Y,Z by comma's, coordinates by spaces -->
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 2"><xsl:if
test="round(0.5 * (position())) = 0.5 * (position())">,0.0 </xsl:if><xsl:if
test="round(0.5 * (position())) != 0.5 * (position())">,</xsl:if></xsl:if>
   <xsl:if test="substring(string(../../SDO_GTYPE),1,1) = 3"><xsl:if
test="round(((position())) div 3) != ((position())) div
3">,</xsl:if><xsl:if test="round(((position())) div 3) = ((position())) div
3"><xsl:text> </xsl:text></xsl:if></xsl:if>
  </xsl:if>
 </xsl:template>

 <!-- geometry, (multi)polygon-->
 <xsl:template match="POLYGONPROPERTY">
  <xsl:element name="gml:geometryProperty">

  <!-- test if feature is polygon (4th number of GTYPE-string represents
the type of feature e.g. 3 = polygon) -->
  <xsl:if test="substring(string(SDO_GTYPE),4,1) = 3">

   <!-- The number of elements in SDO_ELEM_INFO tells whether a polygon is
multi or not (each polygon or polygonmember has 3 values in SDO_ELEM_INFO).
A polygon is a multipolygon, if the SDO_ELEM_INFO contains more than 3
elements -->
   <xsl:if test="count(SDO_ELEM_INFO/SDO_ELEM_INFO_ITEM) > 3">
    <xsl:element name="gml:multiPolygon">
     <xsl:attribute name="srsName">EPSG:7048</xsl:attribute>

     <!-- each first, fourth, seventh, etc. element of the
SDO_ELEM_INFO_ITEM-array, gives the beginning of the polygon's ordinates in
the ordinates-array -->
     <xsl:for-each select="SDO_ELEM_INFO/SDO_ELEM_INFO_ITEM">
     <xsl:if test="round(((position()) -1) div 3) = ((position() -1) div
3)">
     <xsl:element name="gml:PolygonMember">
      <xsl:element name="gml:Polygon">
```

```
       <xsl:element name="gml:outerBoundaryIs">
        <xsl:element name="gml:LinearRing">
         <xsl:element name="gml:coordinates">

             <!-- use template ORDINATESMULTI for each polygonmember -->
             <xsl:call-template name="ORDINATESMULTI">
              <!-- set startingpoint and endpoint for the polygon's
ordinates in the ordinates-array -->
              <xsl:with-param name="begin">
               <xsl:value-of select="current()"/>
              </xsl:with-param>
              <xsl:with-param name="end">
               <!-- the ordinates for the last polygonmember go up to the
ordinates-array's end -->
               <xsl:if test="position() + 2 = last()"><xsl:value-of
select="count(../../SDO_ORDINATES/SDO_ORDINATES_ITEM)+1"/></xsl:if>
               <!-- the ordinates for all but the last polygonmember go up
to the next polygonmember's starting point -->
               <xsl:if test="position() + 2 != last()"><xsl:value-of
select="following-sibling::SDO_ELEM_INFO_ITEM[3]"/></xsl:if>
              </xsl:with-param>
             </xsl:call-template>

         </xsl:element>
        </xsl:element>
       </xsl:element>
      </xsl:element>
     </xsl:element>
    </xsl:if>
    </xsl:for-each>

    </xsl:element>
   </xsl:if>

   <!-- A polygon is a simple polygon, if the SDO_ELEM_INFO contains
precisely 3 elements -->
   <xsl:if test="count(SDO_ELEM_INFO/SDO_ELEM_INFO_ITEM) = 3">
    <xsl:element name="gml:Polygon">
     <xsl:attribute name="srsName">EPSG:7048</xsl:attribute>
     <xsl:element name="gml:outerBoundaryIs">
      <xsl:element name="gml:LinearRing">
       <xsl:element name="gml:coordinates">
        <xsl:apply-templates/>
       </xsl:element>
      </xsl:element>
     </xsl:element>
    </xsl:element>
   </xsl:if>

  </xsl:if>

  </xsl:element>
 </xsl:template>

 <!-- Following elements in the XML-output should not be printed in GML-
format -->
 <xsl:template match="OID">
 </xsl:template>

 <xsl:template match="SDO_GTYPE">
 </xsl:template>
```

```
 <xsl:template match="SDO_ELEM_INFO">
 </xsl:template>

 <xsl:template match="SDO_ELEM_INFO_ITEM">
 </xsl:template>

 <!-- Following elements in the XML-output should be printed as feature
attributes in GML-format -->
 <xsl:template match="TOP10_ID">
  <xsl:element name="tdn:top10_id"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="BEGINDATUM">
  <xsl:element name="tdn:begindatum"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="EINDDATUM">
  <xsl:element name="tdn:einddatum"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="LANDGEBRUIK">
   <xsl:element name="tdn:landgebruik"><xsl:apply-templates/></xsl:element>
 </xsl:template>

<xsl:template match="TOEGANKELIJKHEID">
   <xsl:element name="tdn:toegankelijkheid"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="VOORKOMEN">
   <xsl:element name="tdn:voorkomen"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="NAAM">
   <xsl:element name="tdn:naam"><xsl:apply-templates/></xsl:element>
 </xsl:template>

 <xsl:template match="FYSIEK_VOORKOMEN">
   <xsl:element name="tdn:fysiek_voorkomen"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="HOOGTENIVEAU">
   <xsl:element name="tdn:hoogteniveau"><xsl:apply-
templates/></xsl:element>
 </xsl:template>

 <xsl:template match="TDNCODE">
   <xsl:element name="tdn:tdncode"><xsl:apply-templates/></xsl:element>
 </xsl:template>

</xsl:stylesheet>
```

d Gegenereerde GML voor ‘terrein’

```
<tdn:terrein xmlns:tdn="http://www.gdmc.nl/tdn" fid="TOP10.5100295">
   <tdn:top10_id>5100295</tdn:top10_id>
   <tdn:begindatum>24 JUN 2001 16:03:13</tdn:begindatum>
   <tdn:landgebruik>Overig</tdn:landgebruik>
```

```
   <tdn:fysiek_voorkomen>Overig</tdn:fysiek_voorkomen>
   <tdn:toegankelijkheid>Openbaar</tdn:toegankelijkheid>
   <tdn:voorkomen>Overig</tdn:voorkomen>
   <tdn:naam>Onbekend</tdn:naam>
   <tdn:hoogteniveau>0</tdn:hoogteniveau>
   <gml:geometryProperty xmlns:gml="http://www.opengis.net/gml">
      <gml:Polygon srsName="EPSG:7048">
         <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:coordinates>159459.709,432008.521,0.0
159447.231,432000,0.0 159473.132,432000,0.0 159473.585,432000.169,0.0
159473.648,432000,0.0 159490.159,432000,0.0 159482.1,432020.726,0.0
159467.839,432013.12,0.0 159459.709,432008.521,0.0 </gml:coordinates>
            </gml:LinearRing>
         </gml:outerBoundaryIs>
      </gml:Polygon>
   </gml:geometryProperty>
   <tdn:tdncode>5263</tdn:tdncode>
</tdn:terrein>

<tdn:terrein xmlns:tdn="http://www.gdmc.nl/tdn" fid="TOP10.5100159">
   <tdn:top10_id>5100159</tdn:top10_id>
   <tdn:begindatum>24 JUN 2001 16:03:13</tdn:begindatum>
   <tdn:landgebruik>Overig</tdn:landgebruik>
   <tdn:fysiek_voorkomen>Overig</tdn:fysiek_voorkomen>
   <tdn:toegankelijkheid>Openbaar</tdn:toegankelijkheid>
   <tdn:voorkomen>Overig</tdn:voorkomen>
   <tdn:naam>Onbekend</tdn:naam>
   <tdn:hoogteniveau>0</tdn:hoogteniveau>
   <gml:geometryProperty xmlns:gml="http://www.opengis.net/gml">
      <gml:multiPolygon srsName="EPSG:7048">
         <gml:PolygonMember>
            <gml:Polygon>
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates>159562.536,432010.179,0.0
159565.26,432003.328,0.0 159566.925,432000,0.0 159619.485,432000,0.0
159618.319,432002.273,0.0 159619.671,432024.717,0.0
159604.528,432065.549,0.0 159595.881,432067.764,0.0
159583.148,432063.436,0.0 159596.957,432021.742,0.0
159562.536,432010.179,0.0 </gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </gml:PolygonMember>
         <gml:PolygonMember>
            <gml:Polygon>
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates>159600.198,432004.01,0.0
159598.219,432009.947,0.0 159608.685,432013.435,0.0
159610.664,432007.498,0.0 159600.198,432004.01,0.0 </gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </gml:PolygonMember>
         <gml:PolygonMember>
            <gml:Polygon>
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
```

```
                    <gml:coordinates>159596.557,432029.969,0.0
159590.841,432050.761,0.0 159599.353,432053.101,0.0
159605.069,432032.309,0.0 159596.557,432029.969,0.0 </gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </gml:PolygonMember>
      </gml:multiPolygon>
   </gml:geometryProperty>
   <tdn:tdncode>5263</tdn:tdncode>
</tdn:terrein>
```

V Lijst van gebruikte afkortingen en begrippen

Afkorting	Eventuele omschrijving
API Application Program Interface	
BLOB Binary Large Object	Datatype om grote hoeveelheden gegevens (bijv. een document) in een binair formaat op te slaan in de kolom van een tabel.
CLOB Character Large Object	Datatype om grote hoeveelheden gegevens (bijv. een document) in een character-formaat op te slaan in de kolom van een tabel.
CSS Cascading Style Sheets	Met CSS kunnen weergavestijlen worden gemaakt voor XML. CSS is echter het meest bekend door gebruik met HTML.
DTD Document Type Definition	Definieert de structuur van bepaalde XML-documenten. Met een DTD kan ook de structuur van die documenten getoetst worden. Dit wordt valideren genoemd.
GML Geography Markup Language	Geography Markup Language (GML) is een op XML gebaseerde codering voor het transport en de opslag van geografische informatie, die zowel de ruimtelijke als niet-ruimtelijke eigenschappen van geografische objecten beslaat.
HTML Hypertext Markup Language	HTML wordt gebruikt om structuur en weergavestijl (meestal voor webpagina's) vast te leggen. HTML is een vaste verzameling tags en attributen.
OGC OpenGis Consortium	Internationaal consortium van industrieen, overheden en universiteiten op het gebied van GIS met als doel publiek beschikbare specificaties voor 'geoprocessing' te ontwikkelen.
SDO Spatial Data Object	SDO-GEOMETRYwordt in Oracle gebruikt om ruimtelijke objecten te definiëren
SGML Standard Generalized Markup Language	Een internationale standaard voor documentatie. SGML wordt als basis gebruikt voor bijvoorbeeld HTML en XML.
SQL Standard Query Language	Vraagtaal voor databases
W3C World Wide Web Consortium	
XDK XML Developers Kit	Component in Oracle 9i waarin functionaliteit zit om met XML te werken. XDK's zijn beschikbaar voor meerdere programmeertalen (bijv. Java, C, C++, PL/SQL) en kunnen verschillende tools bieden.
Xlink Xlink	XLink verschaft geavanceerde mogelijkheden om vanuit XML-documenten te verwijzen naar andere (externe) bestanden of in meerdere richtingen.
XML Extensible Markup Language	XML wordt omschreven als een verzameling regels voor het ontwerpen van tekstformaten voor gestructureerde data, op een dusdanige manier dat het eenvoudig is voor een computer bestanden te genereren en te lezen. XML maakt gebruik van tags en attributen, die naar gelang de toepassing gedefinieerd kunnen worden.
XML Schema	Definieert de structuur van XML-documenten. Met een XML Schema kan ook de structuur van die documenten getoetst worden. Dit wordt valideren genoemd. Een XML Schema heeft meer mogelijkheden dan een DTD.
Xpointer Xpointer	XPointer is ontworpen om een element of een verzameling elementen te lokaliseren in een XML-document.
XSL Extensible Stylesheet Language	XSL wordt gebruikt om stijlbladen te definiëren voor XML. XSL is uitgebreider dan CSS. Wanneer XSL wordt gebruikt om een transformatie uit te voeren naar een ander bestandsformaat, wordt ook wel gesproken van XSLT.
XSLT of XSLTransformatie	Transformatie waarbij XSL wordt gebruikt om XML-documenten te transformeren naar een ander (XML-)formaat.
XSU XML SQL Utility	Component in Oracle 9i, waarmee XML kan worden gegenereerd, ingelezen en gewijzigd. XSU werkt op basis van SQL-instructies.

Literatuurlijst

[1] Bert Bos. *XML in 10 punten*. http://www.w3.org/XML/1999/XML-in-10-points, maart 1999.

[2] Robert Eckstein and Michel Casabianca. *XML Pocket Reference*. O'Reilly & Associates, Inc, april 2001, 2e editie.

[3] Shelley Higgins e.a. *Oracle 9i Application Developer's Guide – XML*. Oracle Corporation, juni 2001.

[4] Open GIS Consortium, Inc. Geography Markup Language (GML). Versie 2 (01-029), OGC, februari 2001.

[5] M.E. de Vries, T.P.M. Tijssen, J.E. Stoter, C.W. Quak, P.J.M. van Oosterom. *The GML Prototype of the new TOP10vector object model*. GISt Report No. 9, TU Delft, Faculteit CiTG, Afdeling Geodesie, december 2001.

[6] W3C. *Extensible Markup Language (XML) 1.0 (Second Edition)*. http://www.w3.org/TR/REC-xml, oktober 2000.

[7] ZapThink, LLC. *The "Pros and Cons" of XML*. Waltham, Massachusetts, 2001.