XML 2010

Erik Tjong Kim Sang
erikt(at)xs4all.nl

11 oktober 2010

Vorige week

Zoeken in XML-documenten met XPath

Deze week

Opslaan van XML-data in databases

Zoeken in XML-data

De vorige week hebben we gezien hoe met XPath-expressies kan worden gezocht in XML-data

We hebben ook zoekopdrachten uitgevoerd met `xsltproc`

De opdrachten werden snel verwerkt maar we gebruikten ook kleine databestanden (bijvoorbeeld 200 zinnen geannoteerde tekst)

Wat gebeurt er als we zoeken in grotere documenten?

Hoe doorzoekt XPath XML-data?

XPath doorzoekt XML-data sequentieel: van alle elementen wordt 1 voor 1 gecontroleerd of ze voldoen aan de zoekopdracht

In 200 zinnen kan je zo snel iets terugvinden maar voor resultaten voor 1 miljoen zinnen moet je lang wachten

Dit probleem is ernstiger voor ingewikkelde zoekopdrachten (practicum-opdracht 5.10)

We moeten op zoek naar een manier van dataopslag waarin snel gezocht kan worden

Indexeren van tekst

De standaardmanier om teksten snel doorzoekbaar te maken, is door een **index** van de tekst te bouwen:

- maak een lijst van alle woorden in de teksten
- hou voor elk woord bij in welk document het staat
- zoek vervolgens in de woordenlijst (index) in plaats van in de teksten

Als je wil weten in welke documenten een woord staat dan kan je dat snel opzoeken in de index, net als bij een boek met een index

Deze methode wordt gebruikt door zoekmachines zoals Google

Indexeren van XML-documenten?

Indexeren werkt goed voor platte tekst: tekst zonder annotaties

Immers: in een index staan alleen individuele woorden

Maar een XML-document bevat zowel woorden, elementen als relaties tussen woorden en elementen

Deze complexe informatie is niet eenvoudig op te slaan in een index

Opslaan van XML-data in databases

Om de structuurinformatie van XML-data te behouden, gaan we de XML-data opslaan in databases

In databases kan gestructureerde informatie worden opgeslagen en geïndexeerd

Dit is precies wat we nodig hebben voor onze XML-data

De structuur van databases

In standaard databases (relationele databases) worden gegevens opgeslagen in de vorm van tabellen:

	kolom 1	**kolom 2**	**kolom 3**
rij 1	rij 1, kolom 1	rij 1, kolom 2	rij 1, kolom 3
rij 2	rij 2, kolom 1	rij 2, kolom 2	rij 2, kolom 3

Een tabel bestaat altijd uit een vooraf gedefinieerd aantal kolommen en een variabel aantal rijen

Passen XML-gegevens in tabellen?

Sommige XML-gegevens passen perfect in een tabel:

```
<results>
  <result>                              <!-- rij 1   -->
    <hometeam>De Graafschap</hometeam>  <!-- kolom 1 -->
    <awayteam>Feyenoord</awayteam>      <!-- kolom 2 -->
    <homegoals>1</homegoals>            <!-- kolom 3 -->
    <awaygoals>1</awaygoals>            <!-- kolom 4 -->
  </result>
  <result>                              <!-- rij 2   -->
    <hometeam>PSV</hometeam>            <!-- kolom 1 -->
    <awayteam>VVV-Venlo</awayteam>      <!-- kolom 2 -->
    <homegoals>3</homegoals>            <!-- kolom 3 -->
    <awaygoals>0</awaygoals>            <!-- kolom 4 -->
  </result>
</results>
```


Nog een voorbeeld van XML-data voor tabellen

Ook de volgende XML-gegevens passen in een tabel:

```
<eurovision year="2010">
  <country name="Moldova">              <!-- 5 x kolom 1  -->
    <points from="Azerbaijan">6</points>  <!-- kolom 2 + 3  -->
    <points from="Belarus">4</points>     <!-- kolom 2 + 3  -->
    <points from="Georgia">1</points>     <!-- kolom 2 + 3  -->
    <points from="Romania">10</points>    <!-- kolom 2 + 3  -->
    <points from="Portugal">6</points>    <!-- kolom 2 + 3  -->
  </country>
  <country name="Cyprus">               <!-- n x kolom 1  -->
    <points from="Greece">12</points>     <!-- kolom 2 + 3  -->
    <points from="Denmark">1</points>     <!-- kolom 2 + 3  -->
    <points from="Slovakia">2</points>    <!-- kolom 2 + 3  -->
    ...
```


Soms passen XML-gegevens niet in tabellen

Tekst verrijkt met XML-annotatie past niet goed in een tabel:

```
<toneelstuk>
  <beschrijving>                                    <!-- kolom 1 -->
    ANTONIO, SALARINO _en_ SOLANIO _komen op._      <!-- kolom 3 -->
  </beschrijving>
  <tekst spreker="SALARINO">                  <!-- kolommen 1 en 2 -->
     Ook niet? Komaan, dan zult ge somber zijn,     <!-- kolom 3 -->
     Wijl gij niet vroolijk zijt; 't ging even goed
     Te lachen en te dansen en te zeggen
     "'k Ben vroolijk," omdat gij niet somber zijt.
     Bij den tweehoofd'gen Janus,
     <beschrijving>                                 <!--   ???   -->
       [1]                                          <!--   ???   -->
     </beschrijving>
```


XML-shredding

Voor XML-informatie die niet op eenvoudige manier is om te zetten naar een tabel gebruiken we een algemene methode: XML-shredding

XML-shredding maakt gebruik van het feit dat XML-elementen een vast aantal attributen hebben: die zijn immers gedefinieerd in een DTD of schema

XML-shredding maakt voor elk XML-element een aparte tabel met de attributen als kolommen

Omkeerbaarheid van XML-shredding

We willen dat XML-shredding omkeerbaar is: dus dat uit de tabellen de originele documenten weer herleidbaar zijn

Om dit te bereiken moet het XML-document aan extra eisen voldoen:

- in tekst mogen geen XML-elementen staan
- elk element moet een attribuut `id` met een unieke waarde krijgen
- elk element moet een attribuut `parent` krijgen dat de waarde van het attribuut `id` van zijn ouder bevat

XML-document voor shredding

Deze XML-tekst is aangepast aan de drie hierboven genoemde eisen:

```
<toneelstuk id="id1" parent="id0">
  <beschrijving id="id2" parent="id1">
    ANTONIO, SALARINO _en_ SOLANIO _komen op._
  </beschrijving>
  <tekst spreker="SALARINO" id="id3" parent="id1">
     <tekstintekst id="id4" parent="id3">
       Ook niet? Komaan, dan zult ge somber zijn,
       Wijl gij niet vroolijk zijt; 't ging even goed
       Te lachen en te dansen en te zeggen
       "'k Ben vroolijk," omdat gij niet somber zijt.
       Bij den tweehoofd'gen Janus,
     </tekstintekst>
     <beschrijving id="id5" parent="id3">
       [1]
```


XML-document na shredding

We maken 1 tabel per XML-element met als kolommen de attributen van het element en de tekst (indien aanwezig):

toneelstuk

id	**parent**
id1	id0

beschrijving

id	**parent**	**text**
id2	id1	ANTONIO ...
id5	id3	[1]

tekst

id	**parent**	**spreker**
id3	id1	SALARINO

tekstintekst

id	**parent**	**text**
id4	id3	Ook niet? ...

Zoeken in databases

Voor het zoeken in de tabellen kunnen we standaarddatabasecommando's gebruiken zoals `SELECT`:

```
SELECT id                  -- selecteert een kolom
FROM tekst                 -- definieert de tabel
WHERE spreker='SALARINO'   -- selecteert rijen
```

Dit commando selecteert de waarden in de kolom `id` in de rijen van de tabel `tekst` waarvoor de kolom `spreker` de waarde `SALARINO` bevat

Dit correspondeert met de XPath-zoekopdracht:
`//tekst[@spreker='SALARINO']/@id`

Nog een voorbeeld van een zoekopdracht

De meeste zoekopdrachten combineren informatie van meerdere tabellen en zijn dan ingewikkelde in de databasetaal (SQL) dan in XPath

Bijvoorbeeld: `//tekst[@spreker='SALARINO']` wordt

```
SELECT tekstintekst.text
FROM tekstintekst, tekst
WHERE tekst.spreker='SALARINO' AND
      tekstintekst.parent=tekst.id
```

De omzetting van XPath naar de databasetaal doen we niet zelf maar gebeurt automatisch met andere software

Hoe ziet de uitvoer van een zoekopdracht eruit?

De uitvoer van een zoekopdracht in een database is een tabel

beschrijving

id	parent	text
id2	id1	ANTONIO ...
id5	id3	[1]

Sommige databasesystemen hebben ook de mogelijkheid om de uitvoer als XML te laten zien

Hierbij kan worden gekozen om kolommen te laten zien als attributen of als elementen

Voorbeelden van tabeluitvoer in XML-formaat

Kolommen als attributen:

```
<beschrijving id="id2" parent="id1" text="ANTONIO ..."/>
<beschrijving id="id5" parent="id3" text="[1]"/>
```

Kolommen als elementen:

```
<beschrijving>
  <id>id2</id>
  <parent>id</parent>
  <text>ANTONIO ...</text>
<beschrijving>
```

Het is ook mogelijk om combinaties hiervan te maken

Het databasesysteem `MySQL`

MySQL is een veelgebruikt gratis databasemanagementsysteem dat enige ondersteuning levert voor XML-data

Het bestaat uit een *client-server*-architectuur die het mogelijk maakt dat verschillende gebruikers tegelijkertijd met de databases werken

XML-data kan in de database worden opgeslagen in tabelvorm

Het kan ook worden opgeslagen in tekstvorm met XML-elementen
In dat geval kan in de tekstdata worden gezocht met XPath

Zoeken in MySQL met XPath

XPath is in MySQL beschikbaar via de functie `ExtractValue`

Hiermee kunnen delen van een kolom tekst worden geselecteerd:

```
mysql> SELECT ExtractValue(text,'//hometeam')
       FROM xmltekst
       WHERE id='id0';
```

Deze opdracht selecteert rijen uit de tabel `xmltekst` waarvoor het attribuut `id` de waarde `id0` heeft

Van deze rijen laat dit commando alleen de inhoud van de elementen `<hometeam>` uit de kolom `text` zien

Case-studie 1: XML-data in MySQL

We gaan nu het toneelstukdocument op 2 manieren in MySQL laden:

1. als onderdeel van een tekstveld (met alle XML-elementen)
2. als verschillende tabellen, na shredding:
   - eerst definieren we tabellen, bijvoorbeeld met:
     `CREATE TABLE toneelstuk ( id TINYTEXT );`
   - daarna zetten we XML met XSLT om naar tabelvorm:
     `INSERT INTO beschrijving VALUES ("id20","id18","[1]");`

Hierbij moeten we rekening houden met de representatie van quotes

XQuery

We hebben tot nu toe twee talen gezien voor de verwerking van XML-data:

- XPath: voor het zoeken in XML-data
- XSLT: voor het converteren van XML-structuren in andere structuren

We missen nog een taal voor het bewerken van (gedeeltes van) XML-documenten, zoals bijvoorbeeld SQL dat kan voor databases

XQuery is een taal waarmee XML-documenten kunnen worden bewerkt

Een voorbeeld van een XQuery-opdracht

De belangrijkste XQuery-instructie is de FLWOR-instructie:

```
for $tekst in doc("toneelstuk.xml")/toneelstuk/tekst
let $spreker:=$tekst/@spreker
where $spreker="SALARINO"
order by $tekst/@id
return <selected>{$tekst/tekstintekst/text()}</selected>
```

Bovenstaande voorbeeldinstructie selecteert de teksten van de spreker `SALARINO`, sorteert ze en plaatst ze in elementen `selected`

Een voordeel van Xquery boven XSLT is de beschikbaarheid van variabelen, zoals hier `$tekst` en `$spreker`

Zelf XQuery gebruiken

De eenvoudigste manier om zelf te experimenteren met XQuery is via het programma Saxon, beschikbaar op `http://saxon.sourceforge.net/`

Als je programmeertaal Java op je computer beschikbaar hebt dan kan je Saxon downloaden, uitpakken en XQuery-programma's draaien als:

```
java -cp saxon9he.jar net.sf.saxon.Query -q xquery.xq
```

Een programma zoals die van de vorige slide kan worden opgeslage in het bestand `xquery.xq`

Case-studie 2: Lassy

Lassy is een onderzoeksproject waarin Nederlandse tekst automatisch syntactisch is geannoteerd, onder andere door woordklassen (`pos`) en woordrelaties (`rel`) toe te voegen

De annotatie is gecodeerd in XML, bijvoorbeeld:

```
<node id="9" pos="noun" rel="hd" word="schrijver"/>
```

In totaal beschrijft de annotatie twee miljard woorden waarvan 1 miljoen woorden zijn gecontroleerd door mensen

Hoe kunnen we snel in deze verzameling zoeken?

De Lassydatabase

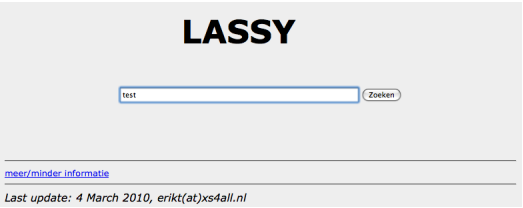
Alle woorden van LASSY-klein (1 miljoen woorden) zijn opgeslagen in een tabel van een MySQL-database

Een selectie van de XML-attributen die bij elk woord horen, komen terug in de tabel als kolommen

Daarnaast zijn bij elk woord attributen van het bijbehorende hoofdwoord toegevoegd als kolommen

Tenslotte is een webinterface gebouwd die zoekopdrachten van gebruikers omzet naar MySQL-instructies en zoekresultaten mooi laat zien

Demonstratie Lassy-webinterface



`http://urd.let.rug.nl/erikt/lassy/`

Achtergrondliteratuur

David Hunter, Jeff Rafter, Joe Fawcett and Eric van der Vlist, *Beginning XML (Programmer to Programmer)*. John Wiley & Sons, 2004 (chapter 10: XML and Databases; chapter 9: XQuery)

Anders Møller and Michael I. Schwartzbach, *An Introduction to XML and Web Technologies*. Addison-Wesley, 2006 (chapter 6.9: XML Databases)

Jon Stephens, *Using XML in MySQL 5.1 and 6.0*. `http://dev.mysql.com/tech-resources/articles/xml-in-mysql5.1-6.0.html` , retrieved in 2010.

EINDE