SQL alleen is niet meer voldoende voor ontsluiten van data

Data voor het Web: snapt u het nog?

Roland Bouman en Jos van Dongen

BI- en database professionals denken bij het woord 'data' vooral aan mooie gestructureerde gegevensverzamelingen die via een standaard querytaal en dito connectiviteitssoftware op simpele wijze toegankelijk zijn om mooie rapporten van te bouwen.

Deze wereld verandert echter snel; toepassingen voor bijvoorbeeld Competitive Intelligence dwingen tot een blik over de eigen schutting, en wanneer men bijvoorbeeld voor sentimentanalyse gebruik wil maken van de oneindige databronnen die door 'social media' gevormd worden (denk aan blogs, Twitter en Facebook) moeten er wat nieuwe kunstjes geleerd worden.
De vraag is dan ook hoe het fenomeen 'data voor het web' in te passen valt in onze praktijken voor data integratie en datamanagement. In dit artikel nemen we een aantal technieken en trends onder de loep.

De bouwstenen van het web

Op het eerste gezicht lijkt informatieuitwisseling voor het web ingericht te zijn volgens een aantal duidelijke standaarden, maar schijn bedriegt. De factor die zowat alles wat web heet met elkaar verbindt, is het *Hypertext Transfer Protocol* (HTTP). Bijna even belangrijk is de *Hypertext Markup Language* (HTML). Zowel HTTP als HTML zijn ontstaan in het begin van de jaren negentig van de vorige eeuw. Om het ontstaan en functioneren van beide te begrijpen is het nuttig om te kijken naar het gemeenschappelijk concept *hypertext*. Hypertext is een algemeen concept dat veel verder teruggaat dan zowel HTTP als HTML. Het gaat hier primair om gecomputeriseerde maar voor mensen leesbare tekst die speciale verwijzingen bevat, die direct toegang geven tot andere informatiebronnen, de zogenaamde *hyperlinks*. De term hypertext en het eerste (experimentele) hypertextsysteem stammen uit de tweede helft van de jaren zestig van de vorige eeuw. Pas zo'n 15 tot 20 jaar daarna zijn hypertextsystemen commercieel beschikbaar geworden, met de echte doorbraak voor het grote publiek in 1993, toen de Mosaic browser het levenslicht zag.
HTTP is een communicatieprotocol dat vastlegt hoe een verzoek (*request*) van een client (zoals bijvoorbeeld een web browser) moet worden geïnterpreteerd door een server (een website) om een passend antwoord (*response*) te geven. Belangrijke kenmerken van HTTP zijn diens interoperabiliteit en simpelheid. Zo is het protocol zelf volledig gebaseerd op (8-bits) tekst. Verder is HTTP *stateless*: de eenheid van communicatie is één request/response cyclus en het protocol zelf voorziet niet in de mogelijkheid om een sessie over meerdere request/response cycli heen te onderhouden. Tot slot is HTTP generiek: het protocol is onafhankelijk van de aard en inhoud van de resources en kan dus altijd worden toegepast.

Kerngegevens van ieder HTTP-verzoek zijn de URL (Uniform Resource Locator) en de *method*, ook wel aangeduid als *verb* (werkwoord). De URL identificeert een *resource* en specificeert zowel het adres alsook het overdrachtsprotocol voor een collectie data (bijvoorbeeld een bestand) of een service. URL's zijn breder inzetbaar dan alleen voor het HTTP-protocol en hun precieze formaat en semantiek zijn vastgelegd in een eigen, aparte standaard. De method geeft aan welke actie de client wenst uit te voeren op de resource. De HTTP-specificatie definieert een lijst van slechts enkele methods, waarvan GET (verkrijgen), POST (posten, een actie die van alles en nog wat kan betekenen) en PUT (plaatsen) het meest gebruikt zijn. Afhankelijk van de gebruikte method wordt in het HTTP-verzoek ook de aanwezigheid van een *entity* (entiteit) verondersteld: een in het bericht ingesloten verzameling data die door de server verwerkt dient te worden. Bijvoorbeeld, de ingevulde formulieren op webpagina's worden meestal als entity in het HTTP-verzoek naar de server gestuurd. In het minimale geval bestaat een HTTP-antwoord alleen uit een *status-line*: een code en een tekstregel die aangeven in hoeverre het verzoek is geslaagd. In de meeste gevallen bevat een HTTP-antwoord echter ook een entity, bijvoorbeeld om (een representatie van) de resource naar de client te zenden.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Afbeelding 1: Formule

HTML vervolgens is een taal om documenten op te maken en te voorzien van hypertext: 'actieve' tekst die door middel van een koppeling direct toegang geeft naar informatie die zich elders op het web bevindt. Al sinds jaar en dag is HTML de dominante standaard voor webpagina's. In het klassieke geval worden HTML-documenten via HTTP als entity meegestuurd in een response ter beantwoording van een GET request.
HTML is een karakteristieke *markup language*: de opmaak en structuur van een document worden gedefinieerd met behulp van annotaties die direct, te midden van de eigenlijke documenttekst zelf, in het document zijn opgenomen. De annotaties zijn van de eigenlijke documenttekst te onderscheiden door het gebruik van speciale (eveneens tekstuele) codes, de zogenaamde *tags*. Vandaar de term markup; door het gebruik van tags worden bepaalde onderdelen en/of gebieden in het document gemarkeerd, zodat deze een speciale betekenis toegekend krijgen. Bijvoorbeeld, door een deel van het document in te sluiten in <P> … </P> tags wordt het ingesloten gebied gemarkeerd als 'paragraaf'. HTML is zelf een toepassing van de algemene *Standard Generalized Markup Language* (SGML) standaard, hetgeen een zogenaamde *meta-markup* taal is (een taal om opmaaktalen in te definiëren).

HTML over HTTP en data overdracht?

De combinatie van HTTP en HTML is in technologisch opzicht nauwelijks ideaal te noemen voor uitwisseling van gestructureerde data. Hoewel tekstgebaseerde formaten zoals HTTP en HTML de interoperabiliteit ten goede komen, lenen zij zich niet zo goed voor efficiënte data overdracht. Met de groei van het aantal, en de toename van de diversiteit in het soort webtoepassingen zijn er steeds meer problemen in het gebruik van HTML aan licht gekomen:

- onvoldoende scheiding van structuur en presentatie;
- weinig strikte syntax leidt tot sluimerende fouten en onnodig complexe programmatuur;
- het formaat is niet uitbreidbaar en dus niet aan te passen aan specifieke soorten content.

Echter, het bloed kruipt waar het niet gaan kan: de populariteit van het world wide web is zelf een drijvende kracht gebleken die puur technologische overwegingen van mindere betekenis heeft gemaakt.

Eén van de problemen is dat HTML als documenttaal eigenlijk tekortschiet. De HTML-syntax omvat enerzijds constructen die er toe dienen om documenten te structureren (semantische opmaak), en anderzijds constructen die tot doel hebben om de presentatie van het document te controleren (typografische opmaak). Het gevolg is dat er vaak presentatieconstructen worden gebruikt waar semantische constructen eigenlijk beter op hun plaats zijn. Bijvoorbeeld, in plaats van een <P> element voor het structureren van paragrafen wordt er gebruik gemaakt van een
 element om een hard regeleinde te forceren.
Een probleem van technische aard is dat de HTML-syntax tamelijk vergevingsgezind is. Eenzelfde construct kan vaak op verschillende manieren worden genoteerd, zelfs binnen hetzelfde document. Zo kunnen paragrafen worden genoteerd door een <P> tag aan de paragraaf tekst vooraf te laten gaan, of door de paragraaf tekst in zijn geheel in te sluiten in een paar corresponderende <P> … </P> tags. Als je niet van hoofdletters houdt, is ook <p> acceptabel, en zelfs een mengelmoesje van hoofd- en kleine letters, zoals <p>...</P>, is gewoon geldig HTML. Om het nog erger te maken worden syntactisch incorrecte constructies vaak door populaire browsers gemaskeerd: browsers rapporteren zelden of nooit fouten in HTML-documenten, maar doen in plaats daarvan een 'best effort' om iets te tonen dat waarschijnlijk in de buurt komt van hetgeen bedoeld werd door de auteur van het document. Het gevolg is dat op het web een grote hoeveelheid te vinden is van documenten die niet syntactisch correct zijn. In het ergste geval leidt dit tot ambiguïteit: eenzelfde stuk HTML kan op verschillende manieren geïnterpreteerd worden, en kan afhankelijk van de parser tot verschillende eindresultaten leiden. In ieder geval maakt de weinig strikte syntax het moeilijker dan strikt genomen noodzakelijk om HTML te interpreteren. Dit uit zich niet alleen in meer complexe programmatuur om HTML te lezen: het wordt ook veel moeilijker om dat soort programmatuur te testen omdat het aantal te testen mogelijkheden veel groter is.

```
<math mode="display" xmlns="http://www.w3.org/1998/Math/MathML">
 <mrow>
   <mi>x</mi>
   <mo>=</mo>
   <mfrac>
     <mrow>
       <mo form="prefix">&minus;</mo>
       <mi>b</mi>
       <mo>&PlusMinus;</mo>
       <msqrt>
         <msup>
           <mi>b</mi>
           <mn>2</mn>
         </msup>
         <mo>&minus;</mo>
         <mn>4</mn>
         <mo>&InvisibleTimes;</mo>
         <mi>a</mi>
         <mo>&InvisibleTimes;</mo>
         <mi>c</mi>
       </msqrt>
     </mrow>
     <mrow>
       <mn>2</mn>
       <mo>&InvisibleTimes;</mo>
       <mi>a</mi>
     </mrow>
   </mfrac>
 </mrow>
</math>
```

Afbeelding 2: Formule in MathXML.

De XML belofte

De HTML-documenttaal is primair gericht op voor mensen leesbare documenten en niet op uitwisseling van abstracte data. De verzameling HTML tags, en daarmee de metadata, is eindig en tamelijk beperkt: in ieder geval te beperkt om te kunnen gebruiken als een generiek data-uitwisselingsformaat. Om aan deze beperking tegemoet te komen is in de periode 1996-1998 Extensible Markup Language (XML) gedefinieerd. XML is een subset van SGML die speciaal met het oog op het gebruik voor het world wide web werd ontworpen. Behalve toepasbaarheid voor het web en compatibiliteit met SGML, stonden de ontwerpers van XML nog andere doelen voor ogen. Enerzijds moest de XML specificatie zelf snel tot stand komen, en simpel maar formeel zijn. Als expliciete doelstelling werd opgenomen dat XML een minimaal aantal optionele features zou bevatten (en idealiter geen) – dit in scherp contrast met SGML. Anderzijds moest het makkelijk worden om XML documenten te creëren, maar ook te lezen. Niet alleen door machines, maar ook door mensen. Al met al een tamelijk ambitieuze lijst van doelstellingen.

Ondanks (of wellicht wel dankzij) deze ambities is XML een doorslaand succes geworden. In vergelijking met SGML is XML een drastische versimpeling, die zowel het ontwerpen van nieuwe documentformaten alsook het schrijven van programmatuur om XML documenten te verwerken veel gemakkelijker maakt. Wat zeker ook aan het succes van XML heeft bijgedragen is de adoptie van de standaard door grote vendors als Sun, IBM en Microsoft. Dit uit zich onder meer in XML ondersteuning in nieuwe en reeds bekende producten en het gebruik van XML voor de definitie van eigen domeinspecifieke documentformaten. In de loop der tijd zijn er vele succesvolle XML toepassingen ontworpen en ingezet. Bekende voorbeelden zijn XHTML (zeg maar HTML, maar dan genoteerd met de meer strikte XML syntax), SVG (Scalable Vector Graphics), ODF (Open Document Format, het native formaat van OpenOffice), MathML (een formaat waarmee wiskundige formules kunnen worden genoteerd), PMML (Predictive Model Markup Language voor datamining tools) en natuurlijk XBRL (eXtensible Business Reporting Language).
Toch is er ook een aantal kanttekeningen bij XML en het succes hiervan te plaatsen. Een punt van kritiek dat vaak terugkeert is dat veel XML toepassingsformaten tamelijk 'breedsprakig' zijn: zelfs om relatief simpele resultaten te boeken is er relatief veel XML code nodig. In deze vorm wordt dit argument vaak gebezigd door auteurs die met de hand XML documenten opmaken. Het voorbeeld in afbeelding 1 uit de wikipedia (http://en.wikipedia.org/wiki/MathML) moge dit illustreren. Indien we de formule in het MathML XML formaat zouden noteren, dan zou de code er ongeveer uitzien als in afbeelding 2. Om het vergelijkbare resultaat in de vanouds bekende LaTeX taal te bereiken, hoeven we slechts het volgende te noteren:

```
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
```

Hetzelfde 'breedsprakigheid' argument weerklinkt ook uit een geheel andere hoek: juist vanuit webtoepassingen geredeneerd is de breedsprakigheid van XML een handicap, omdat het uiteindelijk in een groter datavolume resulteert. Ondanks allerlei slimme compressiealgoritmes om het datavolume bij overdracht te beperken, de toename van de fysieke bandbreedte van het internet en de afname van kosten van opslag op schijf, geldt dat voor de echt grote sites – vooral indien het een gratis dienst betreft – elke byte telt. XML steekt in dit opzicht niet zo gunstig af (temeer daar toepasselijkheid voor het web één van de prominente doelstellingen van XML is).

```
<?xml version="1.0" encoding="utf-8"?>
  <feed xmlns="http://www.w3.org/2005/Atom">

    <title>Example Feed</title>
    <link href="http://example.org/"/>
    <updated>2003-12-13T18:30:02Z</updated>
    <author>
      <name>John Doe</name>
    </author>
    <id>urn:uuid:60a76c80-d399-11d9-b93C-
                                0003939e0af6</id>
    <entry>
      <title>Atom-Powered Robots Run Amok</title>
      <link href="http://example.org/2003/12/13/
                                        atom03"/>
      <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-
                                80da344efa6a</id>
      <updated>2003-12-13T18:30:02Z</updated>
      <summary>Some text.</summary>
    </entry>

  </feed>
```

Afbeelding 3: Atom voorbeeld.

Feeds: RSS en Atom

Over het algemeen werkt een website als volgt: u klikt en de pagina verschijnt. Om het laatste nieuws of de blog van de auteurs te lezen dient dus een browser gestart te worden en naar de desbetreffende pagina genavigeerd te worden, iets wat een specifieke actie van een gebruiker vereist. Het zou natuurlijk veel eenvoudiger zijn als nieuwe content automatisch wordt aangeboden, en dat is precies wat er met 'feeds' gebeurt. In dit geval is er een feed reader (de Google Reader is hier een goed voorbeeld van) waarin aangegeven wordt welke 'feeds' men bij wil houden. Voor het definiëren van deze feeds zijn twee op XML gebaseerde standaarden beschikbaar: RSS (Really Simply Syndication) en Atom. Atom is bedoeld als opvolger van RSS maar zeker gezien de populariteit van RSS voor bijvoorbeeld Podcasts is het merendeel van het feed-aanbod op het web nog

steeds in RSS formaat. Atom biedt ten opzichte van RSS enige verbeteringen, met name op de het gebied van internationalisatie en structureren van content. Het mooie van beide protocollen is echter de eenvoud, waardoor het voor bijvoorbeeld ETL-tools ook mogelijk wordt om feed content te extraheren. In afbeelding 3 is een Atom feed weergegeven met één item (entry) waar alle elementen duidelijk te zien zijn. In een volgend nummer wordt getoond dat het uitlezen en verwerken van deze data een fluitje van een cent is.

Atom en RSS worden niet alleen gebruikt voor feeds, maar er kunnen, dankzij het Atom Publishing Protocol, ook complete interfaces mee gedefinieerd worden. Een bekend voorbeeld hiervan betreft de Google Data API's waarmee gecommuniceerd kan worden met de complete verzameling door Google aangeboden applicaties.

Dataformaten: JSON en YAML

Het is al eerder aangegeven dat XML als opslag- en uitwisselingsformaat van gestructureerde data wellicht niet het meest handige vehikel is. En hoewel de 'X' in AJAX staat voor 'XML' zijn het momenteel toch de nieuwe compactere formaten JSON (JavaScript Object Notation) en YAML (YAML Ain't Markup Language) die aan een gestage opmars op het Web bezig zijn. Beide pogen om de breedsprakigheid van XML te omzeilen en beide bieden ook een formaat dat zowel 'human' als 'machine' readable is. Toch zit met name in dit laatste onderdeel het verschil. JSON is meer een machinetaal die ook voor mensen leesbaar is, terwijl YAML poogt om juist toegankelijk te zijn voor menselijk gebruik, daarbij voor lief nemend dat processing door een computer wat complexer wordt. YAML maakt voor deze leesbaarheid gebruik van spaties en inspringingen, iets wat puristen behoorlijk tegen de borst stuit, terwijl JSON er wat cryptischer uitziet. Hét grote voordeel van deze laatste is echter de compactheid, niet alleen van de objecten, maar ook van JSON zelf. De gehele definitie past dan ook op de achterkant van een visitekaartje dat de mensen achter JSON trots met zich meedragen.

Er is echter meer: alle SGML-afgeleiden zoals HTML en XML zijn bedoeld als opmaaktaal voor documenten, niet noodzakelijkerwijs voor data. Applicaties en andere dataverwerkende processen werken veel efficiënter met een objectserialisatieformaat zoals JSON of YAML, omdat er dan aan de applicatiekant nauwelijks parsing van (XML) documenten plaats hoeft te vinden. Op dit moment is het nog vooral JSON dat de boventoon voert en is er een flinke hoeveelheid tools die dit ondersteunen of zelfs als enige format gebruiken. Een voorbeeld van deze laatste is de Javascript InfoVis Toolkit, maar ook de Freebase web database gebruikt JSON als basis. Sterker nog, Freebase heeft een eigen querytaal ontwikkeld (de MetaWeb Query Language, MQL) die volledig op JSON is gebaseerd. Een andere interessante ontwikkeling is de Apache CouchDB, een schemaloze document database die via een JSON API te bevragen is. Zowel Freebase als CouchDB zullen in één van de volgende nummers nog uitgebreid aan bod komen. In datzelfde artikel zal meteen uitgelegd worden wat nu precies met REST (REpresentational State Transfer) bedoeld wordt en hoe dit architectuurprincipe het mogelijk maakt om gegevens uit te wisselen tussen heterogene sytemen. REST is namelijk één van de pijlers waarop Web 2.0 applicaties rusten en een aanzienlijke vereenvoudiging ten opzichte van het allesbehalve simpele Simple Object Access Protocol (SOAP)

Conclusie

Een compleet artikel over webdata en niet één keer sprake van SQL; dat kan geen toeval zijn en dat is het ook niet. Als er al een conclusie getrokken kan worden, dan is het wel dat SQL alleen al lang niet meer voldoende is voor het ontsluiten van data. Semigestructureerde, schemaloze dataverzamelingen verhouden zich namelijk niet zo goed met de 'S' in SQL en de bijbehorende relationele databases. Daarnaast zijn connectiviteits-API's zoals ODBC en JDBC ook niet geschikt om de data op het web te ontsluiten, aangezien er momenteel nog geen algemeen toepasbare implementaties over HTTP beschikbaar zijn. Aan de andere kant hebben XPath en XQuery, oorspronkelijk bedoeld voor het ontsluiten van XML documenten, het (nog?) niet tot een brede adoptie gebracht. Wellicht wordt het tijd voor UQL, een 'Unified Query Language', waarmee zowel strikt gestructureerde SQL databases als de vrijere varianten als CouchDB en Freebase op een uniforme manier benaderd kunnen worden. In afwachting daarvan zullen we voorlopig moeten leren werken met RSS, Atom, JSON, YAML en REST-style API's.

Referenties

Hypertext – http://en.wikipedia.org/wiki/Timeline_of_hypertext_technology
HTTP pre-spec – www.w3.org/Protocols/HTTP/AsImplemented.html
HTTP 1.1 specificatie – www.w3.org/Protocols/rfc2616/rfc2616.html
URI specifiatie – http://labs.apache.org/webarch/uri/rfc/rfc3986.html
SGML (ISO 8879:1986) – www.iso.org/iso/catalogue_detail.htm?csnumber=16387
XML – www.w3.org/TR/REC-xml/
XML Doelstellingen – www.w3.org/TR/REC-xml/#sec-origin-goals
MathML – www.w3.org/Math/
LaTeX – www.latex-project.org/
MathML en LaTeX formule voorbeelden – http://en.wikipedia.org/wiki/MathML
RSS – www.rssboard.org/
ATOM – http://tools.ietf.org/html/rfc4287
Google API's – http://code.google.com/apis/gdata/
JSON – www.json.org/
JSON businesskaartje – www.flickr.com/photos/equanimity/3763158824/
YAML – www.yaml.org/
Infovis toolkit – http://thejit.org
Apache CouchDB – http://couchdb.apache.org/
Freebase – www.freebase.com/
MQL specificatie – www.freebase.com/docs/mql
Freebase MQL Query Editor – http://freebase.com/app/queryeditor/

Roland Bouman is webapplicatie- en BI-ontwikkelaar voor Strukton Rail en auteur.

Jos van Dongen (jos@tholis.com) is onafhankelijk adviseur, auteur en spreker.