Geen plek voor mannen met hamers

Relaties gerelativeerd

Rick van Rein

Op blogs buzzt het momenteel van NoSQL – een verzamelnaam voor alternatieven voor relationele databases. Tijd voor DB/M om de zin en onzin eens tegen elkaar af te zetten.

NoSQL klinkt alsof het een negatieve term is die relationele databases ter helle zou willen sturen, maar niets is minder waar. Het is een verzamelnaam voor een gemeenschappelijk gevoel dat niet alles in een relationele database thuishoort, en dat er noodzaak is voor andere manieren om data op te slaan.

Hamertje tik

Als je met Oracle spreekt, dan is snel duidelijk hoe zij tegen de wereld aankijken; kunnen ze XML in hun database kwijt, dan lonken ze meteen naar de volgende stap: office-documenten, het filesysteem, enzovoort. Met verve schetsen ze het beeld van een wereld waarin alles in een database zit. Een relationele database wel te verstaan, en bij voorkeur die van Oracle. Dit is natuurlijk het verhaal van de man met de hamer (voor wie alles op een spijker lijkt). Oracle zet zwaar in op het relationele model, en zal niet snel met een initiatief komen om andersvormige data te ondersteunen als het ook kan met een vertaalslag. Het is geweldig dat ze de moeite nemen om allerlei formaten te ondersteunen, maar tegelijkertijd is het verstandig om een oor te luisteren te leggen bij alternatieven.

> Het idee achter een object-database is een persistente onderlaag voor object-georiënteerde programmeertalen

Er zit natuurlijk heel veel in de benadering van Oracle. In tegenstelling tot bijvoorbeeld MySQL maken ze zeer sterk gebruik van de semantische constructies die bij het relationele model en de vraagtaal SQL horen. Door heel precies te begrijpen wat de verschillende constructies uitdrukken is het in allerlei omstandigheden mogelijk om daar slimme optimalisaties op toe te passen, en dat is precies wat Oracle doet. Zodat massale query's razendsnel kunnen verlopen. Als je ze precies goed opschrijft, tenminste.

Wie de schoen past

Als je MySQL met Oracle vergelijkt dan valt op dat die laatste uitstekend werkt voor grote datasets, en MySQL juist beter geschikt is voor kleinere operaties. Geen wonder, want MySQL doet bijna geen voorwerk om de query te versnellen maar gaat gewoon meteen aan de slag. Minder overhead dus, maar ook minder vruchten om te plukken.

MySQL is een grote stimulans geweest achter het ontstaan van dynamische websites, juist vanwege die verminderde overhead. Als je een webpagina wilt tonen met wat eenvoudige data die je zonder al te veel join- of rekenwerk wilt opdiepen uit je database, dan volstaat dat immers. Pas wanneer je ingewikkelde matches en selecties nodig hebt wordt Oracle voor zo'n toepassing interessant. MySQL is echter nog altijd een relationele database, en eigenlijk past dat model helemaal niet zo goed bij de gemiddelde webtoepassing. Websites sommen weleens lijstjes of rapportjes op, maar in het gros van de gevallen gaat het om een query die een klein aantal objecten afzonderlijk aanspreekt. En dan kom je terecht bij hele andere modellen dan het relationele. Het idee achter een object-database is een persistente onderlaag voor objectgeoriënteerde programmeertalen; dat wil zeggen dat de data in objecten zijn samengevoegd die naar elkaar verwijzen; bovendien kan een object meer of minder uitgebreid zijn in de velden die het bevat.

Verwijzingen tussen objecten passen echter beter bij een programmeertaal (waar het pointer of reference heet) dan bij een database (waar je eerder met een key wilt wijzen naar een andere klont data, of je die klont nu object noemt of niet). Pointers of referenties zijn bovendien binaire constructies (namelijk geheugenadressen) die niet geschikt zijn om via een datamodel zichtbaar te maken aan buitenstaanders.

Een aantal van de NoSQL databases werkt een dergelijk paradigma dan ook uit met een datastructuur die lijkt te zijn geïnspireerd op scripttalen. In Perl kent men de zogenaamde 'hash', een naam die geheel in lijn met het gebrek aan abstractie van de taal verwijst naar de implementatie; in PHP heet ditzelfde een array en in Python heeft het de meest logische naam van dictionary gekregen. Dit is een datastructuur waarin een key/value vertaling plaatsvindt. De key is vaak (maar niet altijd) een string, en de value kan een willekeurig complexe datastructuur zijn. Zoals gesteld wordt dit vaak efficiënt geïmplementeerd op basis van een hash-tabel.

Een paar voorbeelden van de waarden die met een key op te zoeken zijn in NoSQL databases:
'enkelvoudig'
['een', 'lijst', 'van waarden']
{ 'key1': 1, 'key2': ['val1b', 'val2b'] }
Hier zijn vierkante haken gebruikt om een lijst van waarden weer te geven, en accoladen om een (ingebed) object te noteren. Deze notaties zijn niet ongebruikelijk in de NoSQL-wereld; soms wordt de notatie JSON genoemd, kort voor JavaScript Object Notation. JSON lijkt conceptueel wel wat op XML.
Merk op dat elke string weer kan dienen als key waarmee weer een volgende datastructuur opgehaald kan worden; zo kunnen dus referenties tussen objecten worden gelegd die meer aansluiten bij de toegang per object en minder bij de afhandeling van hele verzamelingen gelijksoortige records, zoals in de relationele vorm normaal is.

Lonkend naar het web

De opslag van gegevens in key/value structuren op de manier van JSON sluit beter aan bij de structuur van het web, voor zover daarvoor toegang per object nodig is in plaats van toegang tot gegevens als verzamelingen. Niet toevallig ligt er dan ook zo'n sterke link met talen als JavaScript en Java, die geacht worden met dit soort objecten om te gaan. Dat is niet eens zo'n raar idee: hiermee wordt het mogelijk de browser te laten browsen over een dataset in plaats van puur een documentvertoner te zijn. We hebben het dan over typische AJAX-technieken.
Een vorm van NoSQL-database waar veel over wordt gesproken is de Cloud Store, wat neerkomt op een redundante opslag van min of meer ongestructureerde data. Het gaat hier om het soort structuren dat webgiganten als Amazon en Google aanspreekt; niet toevallig hebben zij dan ook allemaal systemen ontwikkeld die deze kant opgaan.
De term Cloud wordt hier losjes gebruikt op een manier die aansluit bij Cloud computing: gedistribueerd op basis van virtuele lokalisatie zodat er transparant verhuisd kan worden, en doorgaans is ook redundancy van opslag een belangrijk ontwerpcriterium geweest, zodat met minder betrouwbare (en dus goedkopere) componenten kan worden gewerkt. Cloud Stores vormen een soort middenweg tussen een gedistribueerd filesysteem en een database.

Bij toepassingen waarbij een gebruiker meestal op één plek zijn eigen gegevens beheert zijn lockingproblemen natuurlijk een stuk minder zwaarwegend dan bij een relationele aanpak waarin acties op grote sets van data ineens worden toegepast, en waarin dus moet worden voorkomen dat zulke wijzigingen in elkaars wielen rijden. Doordat de lockingwens doorgaans minder stringent is in Cloud Stores wordt dan ook afgeweken van het ACID-model. Logischerwijs volgt daaruit een grotere vrijheid tot distributie, doordat minder coördinatie nodig is vanuit een (al dan niet denkbeeldig) centraal punt. Een vaak voorkomend principe is daarom dat van 'eventual consistency': als je maar lang genoeg wacht dan zijn alle wijzigingen overal bekend en zijn de data overal gelijk.
Cloud Stores zijn natuurlijk niet de enige soorten NoSQL databases; ook object-databases en XML-databases horen er bij. Het is een kwestie van het gebruiken van het juiste databasemodel voor de juiste toepassing.

Voorbeelden van NoSQL-databases

CouchDB: Onderdeel van het Apache project, dus open source. Een database voor documentopslag, op basis van JSON.
MongoDB: Open source, schaalbaar. Een documentdatabase die zich erop beroept 'schema-free' te zijn.
HyperGraphDB: Open source netwerkdatabase. Integreert met Java.
Tuple space: Een elegante mengvorm tussen het relationele idee en de veel lossere alternatieven; tuple space is een verzameling tuples/records die uit te voeren werk representeren; om aan zo'n tuple te werken moet die uit de tuple space worden verwijderd, bij wijze van locking.
Voldemort: Een gedistribueerd opslagsysteem dat key/value structuren ondersteunt. Deze is van LinkedIn, maar Amazon heeft een soortgenoot in Dynamo en Google heeft broertje BigTable.
SimpleDB: Een gedistribueerde database die samenwerkt met Amazon's Elastic Compute Cloud. Een kenmerkend voorbeeld van de versoepeling ten opzichte van ACID is dat de data niet onmiddellijk overal gesynchroniseerd staan.
Db4o: Deze 'database for objects' is gericht op ontwikkelaars die .NET of Java gebruiken. Er is een GPL-variant, wat nogal opvallend is omdat die beperkingen oplegt aan de link met deze commerciële omgevingen. Het zal daarom zijn dat er ook andere licenties zijn.
Cassandra: Open source van Apache. Bedoeld om enorme datasets over veel eenvoudige servers te verspreiden. Wederom een key/value database met verlate consistentie. Er is geen enkelvoudig punt dat het hele systeem kan doen falen.
Mnesia: Tabellen met rijen, maar niet het gewone relationele model. Een record in Mnesia is ook hier weer een key/value combinatie.
InfoGrid: Deze 'internet graph database' is een netwerkdatabase die ernaar streeft eenvoudig bruikbaar te zijn in een webomgeving.

Blijf bij je leest

Hoewel het NoSQL-verhaal erg hip mag klinken, is het goed acht te slaan op mogelijke nadelen. Een database met losse structuren en verwijzingen op basis van strings die in tekstvelden voorkomen heeft als voordeel veel flexibiliteit voor de programmeur, maar als nadeel dat de database zelf geen kennis kan afleiden, waardoor dus ook minder te optimaliseren valt. Het zal dus meer lijken op MySQL dan op Oracle: voorspelbare performance die gelijkmatig opschaalt van kleine problemen

naar grote. Met de beperkte schaalbaarheid die daarbij hoort. Verder is het flexibele van deze structuren ook het recept voor meer werk voor de applicatieprogrammeur: die moet namelijk handmatig aangeven wat waaraan gekoppeld moet worden. En voorkomen dat niet-bestaande waarden in een als key bedoeld veld komen. En problemen op het vlak van security voorkomen. Het mag simpel lijken om een ander object via een tekstuele referentie te vinden, maar als daarmee handmatig een volgende lookup moet worden gedaan, en als referentiële integriteit handmatig gecontroleerd moet worden, dan is de pret voor veel applicaties snel voorbij. Of als de pret zegeviert, dan is het wellicht de betrouwbaarheid van de code die te lijden heeft.

Er zijn hele volksstammen die geloven dat XML de oplossing voor alles is. Onder hen zijn er velen die nooit doorvragen naar de DTD of het XML Schema met de datastructuur, wat veel van hen zegt. Juist de meta-informatie over de datastructuren maakt XML beter dan het erg vrije key/value systeem. Maar met meer structuur komt, ook bij XML, een meer stringente controle op diezelfde structuren. Er zijn natuurlijk hele goede toepassingen van XML, maar het is belangrijk te beseffen dat XML vooral geschikt is voor data die inherent hiërarchisch van structuur zijn. Er zijn in XML manieren gemaakt om buiten de boomstructuur om te verwijzen naar andere elementen, maar dan ben je eigenlijk al een ander datamodel aan het simuleren.
Dat lijkt de belangrijkste trigger om altijd in de gaten te houden: als er simulaties nodig zijn, zoals vertaalslagen van het ene model naar het andere, en zeker wanneer daar minder handige of minder logische constructies bij voorkomen, dan is vermoedelijk een verkeerd datamodel gekozen. En dan is het slechts een kwestie van tijd voor je het gevoel krijgt dat je een paar enorme platvoeten in ranke glazen muiltjes probeert te persen.
Zo ook het relationele model. Het is uitstekend dat er actief gezocht wordt naar alternatieve datamodellen, maar ze zullen evenmin het ultieme antwoord geven als het relationele model. Zolang we ons blijven beseffen dat we met een bepaalde structuur zitten en dat die voor bepaalde toepassingen geschikter is dan voor andere, kunnen we stevig met beide benen op de grond blijven staan. Minder structuur zoals in veel NoSQL omgevingen is geen echt handige oplossing; een andere structuur (dan relationeel) is dat soms wel. Als een structuur goed aansluit bij de wensen dan kan het haast niet strak genoeg worden nageleefd door de database-omgeving.
Zodra we een paradigma heilig verklaren, of dat nu het relationele model is of een andere variant, lopen we het risico onszelf klem te zetten en inefficiënte oplossingen te bedenken. En precies dat is de gedachte achter NoSQL waar we allemaal iets aan kunnen hebben. Een datamodel kan als een paar oogkleppen gaan werken, en niets werkt zo goed als het bestuderen van alternatieven om daar van af te komen.

Rick van Rein
Dr. ir. H. van Rein (rick@openfortress.nl) is ontwikkelaar en beheerder bij OpenFortress Digital signatures.