Schaalbare oplossing slaat data als entiteiten op

# Google datastore: mogelijkheden en beperkingen

Bas Peters

**De Google datastore is de database voor Google App Engine applicaties. Google App Engine is een platform om webtoepassingen te ontwikkelen en te hosten via Google. Je ontwikkelt een toepassing met de programmeertaal Python of met Java. De datastore heeft sinds de introductie in april 2008 tot veel discussie geleid. De kritiek is met name gericht op ontbrekende functionaliteiten ten opzichte van relationele databases.**

Dit artikel gaat in op de mogelijkheden en beperkingen van de datastore. Wat is het voor een database, welke functionaliteiten worden geboden, voor welke toepassingen kun je de datastore gebruiken en voor welke toepassingen is het minder geschikt?

## Misbruik

Een relationele database is niet altijd even geschikt als database achter een website. De opkomst van de *user generated content* heeft veel architecten gedwongen uitgangspunten als databasenormalisatie overboord te zetten om de bezoekersaantallen aan te kunnen. Een interessante case op dit gebied is Flickr, de populaire website voor het delen van foto's. Flickr architect Cal Henderson heeft eens pesterig gezegd dat normaliseren iets is voor watjes. Flickr maakt gebruik van MySQL, een relationele database, maar in het geval van Flickr kun je misschien beter spreken van misbruik. Data zijn niet of nauwelijks genormaliseerd en worden gekopieerd naar verschillende databases die voor verschillende doelen worden ingezet. Bij Flickr garandeert de applicatie dan ook de integriteit van de data en niet de database.

Geen databasenormalisatie toepassen heeft het nadeel dat tabellen erg veel rijen kunnen bevatten, maar ook voor dit probleem bestaat een oplossing. De data worden simpelweg verdeeld over meerdere tabellen en databases. Dit principe wordt *database sharding* genoemd, te vertalen als versplinteren.

Daar komt bij dat websites, naast meer gestructureerde data, vaak ook minder gestructureerde data bevatten en met deze *content* kunnen relationele databases ook niet altijd even goed omgaan. Om content te ontsluiten moet je bijvoorbeeld goed kunnen zoeken op woorden uit de tekst. Bij Flickr wordt content geschreven naar een database met een tabelindeling die is geoptimaliseerd voor schrijfacties en vervolgens gekopieerd naar een database met een tabelindeling die is geoptimaliseerd voor zoekacties.

## Schemaloze databases

De specifieke eisen die webtoepassingen stellen aan databases hebben niet alleen geleid tot het misbruiken van relationele databases, maar ook tot een nieuwe generatie producten die volledig ontworpen zijn voor webtoepassingen. Deze databases zijn geschikter voor het opslaan van minder gestructureerde data en ze zijn zo ontworpen dat ze eenvoudig te integreren zijn, weinig administratie nodig hebben, een hoge beschikbaarheid en performance garanderen en schaalbaar zijn. Voorbeelden zijn Amazon SimpleDB, Apache CouchDB en de Google datastore.

> De applicatie is de eigenaar van de data en niet de database

Deze databases werken zonder schema. Je kunt dus niet de structuur van de database vastleggen zoals bij een relationele database. Ze zijn doorgaans simpel van opzet en stellen de gebruiker in staat entiteiten op te slaan als een verzameling sleutel/waardeparen. De sleutel is vergelijkbaar met de naam van een kolom in een tabel. De waarde kan, net als bij een relationele database, verschillende soorten data bevatten zoals een getal, tekst, een datum of binaire data zoals bijvoorbeeld een afbeelding. Schemaloze databases kunnen vaak goed omgaan met data die nogal variëren van entiteit tot entiteit, denk bijvoorbeeld aan het opslaan van tekstwaarden die per entiteit sterk verschillen in lengte. Databases die data opslaan als sleutel/waardeparen bestaan naast relationele databases, omdat ze eenvoudig te gebruiken zijn, geschikt zijn voor opslag van minder gestructureerde data en ook bij de opslag van Terabytes aan informatie nog een goede performance bieden.

## BigTable

De Google datastore is gebaseerd op Google BigTable. Dit is een schaalbare database die Petabytes aan data kan opslaan in een gedistribueerde omgeving op basis van het Google File System (GFS). BigTable wordt toegepast in een groot aantal Google pro-

jecten waaronder Google Search, Google Maps, Google Earth en YouTube. De Google datastore draait dan ook alleen binnen de infrastructuur van Google. Je kunt de software niet downloaden en installeren op een eigen server. Wanneer je gebruik maakt van de Google infrastructuur hoef je je geen zorgen te maken over *load balancing* of replicatie en dat maakt het beheer van de omgeving eenvoudig. Een nadeel is natuurlijk wel de *vendor lock-in*. Je kunt je applicatie niet snel elders onderbrengen als Google de dienst beëindigt of er te veel voor gaat vragen. Ook moet je voldoende kunnen vertrouwen op Google, want je data staan op Google servers in een gedistribueerde omgeving. Dit betekent dat je data op willekeurige locaties binnen het GFS worden weggeschreven. Voor veel bedrijven en instellingen zal dit een bezwaar zijn. Overigens zijn niet alle functies van BigTable beschikbaar in de Google datastore. BigTable biedt bijvoorbeeld ook versiebeheer, iets wat de Google datastore helaas niet ondersteunt.

## Entiteiten

Binnen een Google datastore worden data opgeslagen in entiteiten. Een entiteit kun je vergelijken met een record in een database. Datastore entiteiten hebben een primaire sleutel om ze uniek identificeerbaar te maken. Als de sleutel niet wordt opgegeven bij het aanmaken van een entiteit, dan kent het systeem er zelf een toe. Je kunt een entiteit ook een soortnaam meegeven, zodat je verzamelingen kunt maken van bepaalde soorten informatie, bijvoorbeeld producten, vacatures, klanten, kandidaten, werkgevers, orders of nieuwsberichten. Een verzameling entiteiten van een bepaalde soort kun je tot op zekere hoogte vergelijken met een tabel in een relationele database.

## Groepen

Entiteiten in de datastore kunnen een ouder hebben, een grootouder of een overgrootouder, zodat je hiërarchische structuren kunt vastleggen. De hiërarchie wordt samen met de soortnaam vastgelegd in de sleutel van een entiteit. De ouder van een entiteit kan later niet gewijzigd worden. Een entiteit zonder ouder wordt een *root* entiteit genoemd.

De hiërarchische structuur is een fundamentele eigenschap van de datastore, want entiteiten met een gemeenschappelijke ouder vormen samen een groep die binnen de gedistribueerde omgeving een fysieke eenheid vormt. Een gevolg van deze opzet is dat transacties alleen uitgevoerd kunnen worden op entiteiten die deel uitmaken van dezelfde groep. Bij het ontwerp van een database op basis van de datastore groepeer je dus entiteiten die als verzameling gewijzigd moeten kunnen worden. Bijvoorbeeld een kandidaat met zijn profielgegevens, c.v., sollicitatiebrieven en sollicitaties, een werkgever met zijn bedrijfsprofiel en vacatures of een klant met zijn orders.

## Sleutel/waardeparen

Entiteiten kunnen eigenschappen hebben. Een eigenschap bestaat uit een sleutel en een waarde. Vergelijk een kolom in een tabel. Eigenschappen van entiteiten kunnen heterogeen zijn,

| Gegevenstype | Omschrijving |
|---|---|
| Blob | Een binair object (niet indexeerbaar) |
| Boolean | Gegevenstype met waarde ja of nee |
| Category | Een categorie of onderwerp |
| Date | Een datum |
| Double | Een getal met een decimale breuk |
| Email | Een emailadres |
| GeoPt | Een geografische locatie |
| Key | Een sleutel van een entiteit |
| Link | Een verwijzing |
| PhoneNumber | Een telefoonnummer |
| PostalAddress | Een adres |
| Rating | Een waarde tussen 0 en 100 |
| String | Een tekst van maximaal 500 tekens |
| ShortBlob | Een binair object van maximaal 500 bytes (niet indexeerbaar) |
| Short, Long, Integer | Een heel getal |
| Text | Een grote hoeveelheid tekst (niet indexeerbaar) |
| URL | Een webadres |
| User | Een Google gebruiker |

**Afbeelding 1**: Overzicht gegevenstypen.

want de toegestane eigenschappen van een entiteit zijn niet in een schema vastgelegd. Stel; je slaat de leeftijd op in een gebruikersprofiel, dan zal de datastore het geen probleem vinden wanneer je de leeftijd de ene keer als heel getal opslaat (40) en de volgende keer met een decimale breuk (8,5). Eigenschappen zijn ook variabel. Als je een entiteit van het soort 'kandidaat' aanmaakt met een eigenschap 'werkervaring', dan kun je vervolgens een tweede entiteit van hetzelfde soort aanmaken zonder deze eigenschap. De waarde van een eigenschap kan verschillende typen gegevens bevatten. Afbeelding 1 geeft een overzicht van de gegevenstypen die beschikbaar zijn in de Java implementatie van de datastore.

Wanneer een waarde wordt gewijzigd, dan mag het gegevenstype ook wijzigen. De bestaande waarde met dezelfde sleutel wordt gewoon vervangen door de nieuwe waarde. Naast de bovenstaande gegevenstypen kan een eigenschap ook een lijst met waarden bevatten. Denk bijvoorbeeld aan een aantal *tags* om de onderwerpen van een nieuwsbericht vast te leggen. In een relationele database moet hiervoor een tweede tabel aangelegd worden met een vreemde sleutel. Bij het opslaan van meerdere waarden is er geen garantie dat de volgorde behouden blijft. In het geval van Java kun je gebruik maken van een implementatie van de List of Set interface om de lijst met waarden vast te leggen.

## Data-integriteit

Als je ervaring hebt met relationele databases, dan zul je je inmiddels wel afvragen hoe je de integriteit van de data kunt bewaken met de Google datastore. Je kunt voor entiteiten van dezelfde soort willekeurig sleutel/waardeparen aanmaken, je kunt voor sleutels met dezelfde naam willekeurige gegevenstypen gebruiken en je mag zelfs bij het wijzigen van een eigenschap van een entiteit het gegevenstype veranderen. Het antwoord hierop is simpel: je bewaakt de integriteit van de data niet in de database, maar in de applicatie die van de database gebruik maakt. De applicatie is de eigenaar van de data en niet de database. Het gevolg hiervan is dat de database onlosmakelijk verbonden is met de applicatie die van de database gebruik maakt.

## Domeinmodel

Dit is de belangrijkste reden waarom de Google App Engine voor de Java implementatie, naast de *low-level interface*, gebruik maakt van standaard Java interfaces om objecten in de database op te slaan en te ontsluiten. Deze interfaces bieden een raamwerk waarin de integriteit van de data bewaakt kan worden in het domeinmodel. In een domeinmodel leg je vast welke entiteiten binnen je domein bestaan, wat hun eigenschappen zijn en welke relaties deze entiteiten hebben. Bijvoorbeeld klanten met een relatie naar de order en producten waarnaar verwezen wordt vanuit de orders.

Vanuit de rol van de ontwikkelaar bekeken, biedt deze aanpak het voordeel dat de ontwikkelaar zich alleen maar hoeft bezig te houden met de applicatielaag. Als het domeinmodel eenmaal is vastgelegd, dan kunnen objecten opgeslagen worden in de datastore. Je hoeft je geen zorgen te maken over de structuur in de achterliggende database, de relaties tussen entiteiten of verwijzingen naar entiteiten. Objecten die relaties hebben worden automatisch in dezelfde groep entiteiten geplaatst (ze hebben een gemeenschappelijke ouder in de hiërarchie), denk bijvoorbeeld aan de relatie tussen een klant en zijn orders. Wanneer je een klant uit het systeem verwijdert, dan verwijdert het systeem ook de orders van deze klant. Verwijzingen komen weer niet in dezelfde groep terecht. De bestelde producten blijven in het systeem als de orders voor die producten verwijderd worden.

> Schemaloze databases kunnen vaak goed omgaan met data die nogal variëren van entiteit tot entiteit

Veel Java programmeurs zijn vertrouwd met de standaard interfaces, wat het gemakkelijk maakt om met Google App Engine te starten. Door deze interfaces te bieden is er volgens Google ook geen sprake van een vendor lock-in. Ze werken immers ook op andere databases. Dit is natuurlijk maar ten dele waar, want je zult bij een migratie naar een andere omgeving ook de bestaande data willen overzetten. De kans is groot dat de data dan van de datastore naar een relationele database gemigreerd moeten worden. Google biedt hier wel wat tools voor.

## Functies

Veel ontwikkelaars zijn teleurgesteld in de mogelijkheden van de Google datastore in vergelijking tot relationele databases, omdat de datastore bijvoorbeeld geen functies kent. Wanneer je bijvoorbeeld een totaalbedrag voor een order wilt tonen, dan zul je de berekening in je applicatie moeten doen. Eventueel kun je de totalen vervolgens opslaan in de database, om te voorkomen dat de berekening steeds moet worden uitgevoerd. Je moet bij het werken met de datastore niet bang zijn om afgeleide gegevens apart op te slaan. Zolang je er maar voor zorgt dat de applicatielaag over de integriteit van de data waakt, door bijvoorbeeld bij iedere wijziging in een bestelling de totalen opnieuw te berekenen en weg te schrijven. Programmeurs van met name bedrijfskritische webtoepassingen zijn vaak gewend om de integriteit van de data bij de applicatie te beleggen en ervaren het vaak als dubbel werk als daarnaast ook een schema onderhouden moet worden.

## Zoeken

De Google datastore ondersteunt de meest gangbare zoekfunctionaliteiten, maar er zijn ook beperkingen. De belangrijkste beperking is dat je niet goed kunt zoeken op woorden of delen van woorden in de tekst, terwijl dit nu juist een belangrijke eis is

voor veel webtoepassingen. Het is ook niet iets wat je zou verwachten van een bedrijf als Google dat groot is geworden met een zoekmachine. Mogelijk is de impact op de huidige architectuur te groot om deze functionaliteit te bieden. Er zijn wel creatieve oplossingen te bedenken voor *full-text search*, bijvoorbeeld door de woorden uit de tekst in een aparte eigenschap op te slaan met een waarde voor iedere woord.
Een query bestaat uit een optionele soortnaam, een optionele ouder en nul of meer filters die zijn samengesteld uit de naam van een eigenschap, een operator en een bepaalde waarde. De datastore ondersteunt verschillende operatoren, zoals kleiner dan, kleiner of gelijk aan, gelijk aan, niet gelijk aan, groter dan, groter of gelijk aan. De meeste filters zijn ook te combineren om meer complexe zoekacties uit te voeren.

Zoekresultaten kunnen gesorteerd worden en je kunt met een cursor werken. Door gebruik te maken van een cursor kun je steeds een beperkte hoeveelheid entiteiten ophalen, bijvoorbeeld ten behoeve van paginering. Door gebruik te maken van de 'groter dan' en 'kleiner dan' operatoren kun je wel op woorden zoeken die beginnen met bepaalde letters, vergelijk het gebruik van een *wild card* bij SQL. De datastore ondersteunt ook de zogenaamde SQL IN operator, bijvoorbeeld om te zoeken op klanten met de achternaam 'Janssen' of 'Pietersen'. Je kunt bij een zoekactie de volledige entiteiten opvragen, of alleen de primaire sleutels en zoekacties kunnen ook binnen een transactie uitgevoerd worden.
Voor iedere soort query legt de datastore een index aan. Via de ontwikkelomgeving van de Google App Engine verloopt dit proces automatisch, in andere gevallen kun je de indexen ook handmatig configureren. Een aantal voor de hand liggende indexen wordt altijd aangemaakt. Iedere index bevat kolommen voor alle eigenschappen die gebruikt worden in een filter of sorteervolgorde. De rijen zijn op een logische volgorde gezet op ouder, eigenschappen waarvan de waarde gelijk moet zijn, eigenschappen waarvan de waarde niet gelijk moet zijn en de eigenschappen waarop gesorteerd moet worden. Het zoeken verloopt in drie stappen:

1. Eerst bepaalt de datastore de juiste index op grond van de soortnaam van de entiteit, de eigenschappen waarop gezocht wordt, de operatoren en de sorteervolgorde.
2. Vervolgens scant de datastore de index tot de eerste entiteit gevonden wordt die aan alle criteria voldoet.
3. De datastore haalt vervolgens de entiteiten op uit de index tot de entiteit die niet aan alle criteria voldoet.

Een gevolg van deze opzet is dat de Google datastore nooit entiteiten kan teruggeven waarvan een van de eigenschappen uit de zoekactie geen waarde heeft. Er moet immers aan alle criteria van de zoekactie voldaan worden. Wanneer je bijvoorbeeld een verzameling entiteiten hebt met de soortnaam 'klant' en je maakt een index voor een zoekactie op achternaam en woonplaats, dan zullen de klanten waarvan de woonplaats niet is ingevuld niet worden opgenomen in de index. Voor gebruik met de standaard interfaces is dit geen probleem, want die kennen 'onder water' altijd een waarde toe (eventueel dus een nulwaarde).
Het is ook niet mogelijk in een zoekactie op meer dan één eigenschap een operator te gebruiken als kleiner dan, niet gelijk aan of groter dan. Je kunt dus wel zoeken op een leeftijd die tussen de 20 en de 40 ligt, maar niet op iemand die ouder is dan 20 en een salaris heeft van minder dan 40.000 euro. Je kunt deze groep operatoren overigens wel combineren met andere operatoren. Bijvoorbeeld zoeken op personen met achternaam 'Janssen', woonplaats 'Amsterdam' en een leeftijd van tussen de 20 en 40. Google adviseert overigens altijd een sorteervolgorde mee te geven, omdat de volgorde uit de index kan veranderen voor nieuwere versies van de datastore.

## Conclusie

Je kunt de Google datastore niet vergelijken met een relationele database. Het is begrijpelijk dat Google heeft gekozen voor standaard interfaces die vooral worden gebruikt voor de opslag van objecten in relationele databases, maar het is ook verwarrend. Ontwikkelaars verwachten hierdoor dat deze interfaces dezelfde functionaliteiten bieden als ze voor de relationele databases doen. Dit is ook de belangrijkste reden waarom er zoveel kritiek is geuit op het product. Misschien had Google er beter voor kunnen kiezen om een eigen interface te bieden, zoals wel het geval is met een product als Oracle Berkeley DB. Oracle biedt met de DPL (Direct Persistence Layer) een eenvoudige Java interface om objecten op te slaan in een Berkeley DB database. Dit is ook een schemaloze database die gebruik maakt van sleutel/waardeparen. Verder is het teleurstellend dat je niet op de volledige tekst kunt zoeken, iets wat je wel zou verwachten van een product van Google. Als Google deze functionaliteit zou toevoegen, dan zou de datastore veel beter geschikt zijn als database voor webtoepassingen.

Toch biedt Google met de datastore een interessant product. Je kunt de meeste data of content die binnen webtoepassingen gebruikt wordt goed opslaan en je kunt de meest gangbare zoekacties uitvoeren. Wel moet je goed weten wat de beperkingen zijn, met name van de indexen. De datastore is ook eenvoudig te gebruiken en vanaf het moment dat je applicatie is gepubliceerd hoef je de omgeving alleen nog maar te monitoren. De Google infrastructuur draagt zorg voor de beschikbaarheid en de schaalbaarheid. Het kan ook een betaalbare oplossing zijn. Tot ongeveer vijf miljoen page views per maand heb je geen kosten op een datastore van maximaal 1 Gigabyte. Pas daarna gaat de teller lopen. Google kan deze quota's wel wijzigen. Er komen regelmatig nieuwe releases uit van de Google App Engine en de datastore. Ik verwacht dan ook wel dat veel van de beperkingen die ontwikkelaars nu ervaren op termijn verdwenen zullen zijn.

**Bas Peters** (bas.peters@vlc.nl) is senior consultant bij VLC en gespecialiseerd in content management.