7 Big Data, Data Lakes en NoSQL-databases

ONDERWERPEN

- NoSQL-databases
- BASE en ACID
- CAP-theorema
- Big Data-ecosysteem

LEERDOELEN

- Je maakt kennis met Big Data en de complicaties die Big Data met zich mee brengt.
- Je weet dat de grote uitdagingen van dataverwerking niet alleen schuilen in de grote hoeveelheden data maar ook in de snelheid waarmee de data verwerkt moet worden.
- Je weet dat de data niet alleen uit relationele gestructureerde databases komt maar ook en steeds meer uit apparaten, vrije tekst en van symbolen op social media.
- Je maakt kennis met Data Lakes en andere databasearchitectuurelementen.
- Je kunt uitleggen dat deze nieuwe elementen nodig zijn om de nieuwe ontwikkelingen het hoofd te bieden.

(c) Pearson Benelux

In het vorige hoofdstuk is de traditionele binnenkant van het datawarehouse getekend. In dit hoofdstuk worden de datalakes en de nieuwe databasearchitectuurelementen behandeld. Dat zijn elementen die nodig zijn om de grote hoeveelheid data in allerlei structuren, zoals het internet die kan leveren, snel te verwerken.

7.1 Big Data

Big Data-technologie is ontstaan als een reactie op bepaalde tekortkomingen van het relationele en het multidimensionale model. Deze modellen sloten goed aan op de situatie van de meeste bedrijven in de vorige eeuw. Met de opkomst van internet, sociale media en bedrijven als Facebook, Google en Amazon is er inmiddels een situatie ontstaan waarin een groot deel van de data ongestructureerd is. We hebben het dan over tweets, foto's, afbeeldingen, video's, en dergelijke. Allemaal vormen van data die niet netjes passen in de rijen en kolommen waaruit een relationele database en ook een multidimensionaal model is opgebouwd. De ongestructureerde data is ook het deel van de data dat het hardst groeit.
Een andere uitdaging is dat de data inmiddels meer verspreid is dan ooit (globaal aanwezig). Zorgen dat al die data op alle punten consistent is en blijft, vergt veel inspanning en tijd.
Over hoe de data betrouwbaar te krijgen en consistent te houden is veel discussie. Deze uitdaging staat bekend als het *ACID versus BASE-debat*. ACID staat daarbij voor: atomic, consistent, isolated en durable.

- **A**tomic: de databasetransactie als geheel slaagt of mislukt.
- **C**onsistent: gedurende de databasetransactie is de staat van de database altijd valide.
- **I**solated: de databasetransactie vindt plaats in isolatie, niet gehinderd door andere bewerkingen.
- **D**urable: een geslaagde transactie blijft bestaan als record, ook bij systeemfalen of stroomstoring.

Een voorbeeld van een ACID-transactie is de overboeking van geld tussen twee personen. Persoon 1 moet 150 euro betalen aan persoon 2. Dit betekent dat het saldo op de rekening van persoon 1 afneemt met 150 euro en op de rekening van persoon 2 toeneemt met 150 euro.
De databasetransactie bestaat uit een aantal taken: het verminderen van het saldo van de rekening van persoon 1 en het vermeerderen van het rekeningsaldo van persoon 2. Als een van deze taken niet slaagt, is de hele transactie niet geslaagd. Taken die al zijn uitgevoerd moeten dan teruggedraaid worden. Bijvoorbeeld: boekingsopdracht en bijschrijven rekening persoon 2 lukt, maar afschrijven rekening persoon 1 mislukt. Alle taken moeten nu worden teruggedraaid. Door deze handelwijze (atomic) is de database altijd valide (consistent).

	Rekening	Saldo	Taakstatus	Transactiestatus
Taak 1	Persoon 1	- E150,-	OK	Open
Taak 2	Persoon 2	+ E150,-	OK	Open
Transactie				OK

TABEL 7.1 De transactie over de *nodes* is als geheel geslaagd of faalt als mislukt

(c) Pearson Benelux

Het CAP-theorema (ook wel Brewers Theorema, naar Eric Brewer die dit in 1981 als eerste presenteerde) stelt dat er drie aspecten zijn waaraan een succesvol systeem moet voldoen in een gedistribueerde omgeving. Dit zijn:

- **C**onsistency: de data is consistent;
- **A**vailability: de data is beschikbaar;
- **P**artition tolerance: het systeem blijft functioneren, ook als delen (partities) van het systeem uitvallen.

Het belang van het CAP-theorema is dat het duidelijk maakt dat in een gedistribueerde omgeving maar twee van de drie aspecten ingevuld kunnen worden. Bij grote, gedistribueerde omgevingen zijn beschikbaarheid (availability) en uitvaltolerantie (partition tolerance) van het grootste belang. Dit betekent dat de consistentie, zoals in ACID is vastgelegd, onbereikbaar wordt. Het antwoord hierop is BASE.
Grote gedistribueerde databasesystemen zoals Googles BigTable, Facebooks Cassandra en Amazon DynamoDB zijn gebaseerd op BASE.
BASE staat voor *Basic Availability*, *Soft state* en *Eventual consistency*.

- **B**asic **A**vailability: elk verzoek krijgt een antwoord, dit kan de gevraagde data zijn maar bijvoorbeeld ook de melding dat de data niet gevonden kan worden of een tijdelijke versie van de data.
- **S**oft state: de toestand van het systeem kan altijd veranderen (is 'soft').
- **E**ventual consistency: uiteindelijk is de data in het systeem consistent.

Een voorbeeld kan duidelijk maken waarin BASE verschilt van ACID. Stel dat een systeem bestaat uit twee *nodes* (N1 en N2), die beide dezelfde dataset D0 bevatten. Node N1 wordt gebruikt om geld op een rekening bij te storten of eraf te halen. Het saldo kan gelezen worden op node N2. Na bijstorting van 85 euro op node N1, is de dataset hier veranderd van D0 naar D1 (+ 85 euro). Na synchronisatie naar node N2 heb je ook hier de dataset D1.

Als na de wijziging van de dataset naar D1 (op node N1) er geen synchronisatie plaatsvindt (door bijvoorbeeld uitval van de machine), ontstaat de situatie dat N1 saldo S1 laat zien, maar N2 nog steeds saldo S0. Beide nodes zijn beschikbaar, maar er is nu (tijdelijk) inconsistentie in het saldo (dataset). Alternatief zou zijn dat het storten op N1 en bijwerken op N2 als één transactie wordt gezien. Zolang N2 niet is bijgewerkt (S1) wordt N1 niet vrijgegeven voor verdere transactie. In dit geval is er altijd consistentie, maar is de beschikbaarheid geringer. In een internetomgeving met e-commerce is beschikbaarheid van het allergrootste belang. Vandaar dat hier gekozen wordt voor de BASE-benadering, en niet voor ACID.

Node	Beginsaldo	Nieuw saldo	Taak node	Status
N1	D0	D1	Saldo aanpassen	OK
N2	D0	D0	Saldo lezen	Open

TABEL 7.2 Saldo op N1 aangepast, tijdelijk geen update N2 mogelijk (eventual consistency)

(c) Pearson Benelux

7.2 Hadoop en Map/Reduce

Map/Reduce is een framework dat in de jaren negentig door Google is ontwikkeld, speciaal voor het snel verwerken van grote hoeveelheden gegevens. Google gebruikte dat voor het indexeren van documenten om haar zoekmachines efficiënter te maken, en later ook voor het beschikbaar stellen van haar analyseplatform Google Analytics.
Map/Reduce kan grote hoeveelheden data snel verwerken door de taken die gedaan moeten worden over verschillende computers te verdelen. De taken worden vervolgens parallel uitgevoerd, waarbij de verschillende computers ook als back-up voor elkaar functioneren. Als een computer uitvalt, neemt een andere het over. Op die manier gaat ook de dataverwerking natuurlijk veel sneller.
Hadoop is op deze gedachte gebaseerd. Het is een Open-Source Java Framework, speciaal voor de opslag en verwerking van grote hoeveelheden data. Met de groei van de hoeveelheden gegevens voor data-analyse steeg de populariteit van Hadoop.
De opzet bij Map/Reduce en Hadoop is dat de totale verwerkingscapaciteit bestaat uit geclusterde, goedkope ('gewone') computers. Het voordeel van deze werkwijze is dat ze vele malen goedkoper is dan het inzetten van speciale hardware. Ook is ze makkelijk uitbreidbaar (horizontaal schaalbaar) door het toevoegen van meer machines.

Map/Reduce is geschikt voor **batch**processen, waarbij de machines in isolatie werken. Als je de data nog niet hebt of als je gegevens wilt afvangen tijdens een lopend proces, is het niet geschikt. Map/Reduce is met name bedoeld voor het werken met een beperkte set van grote data-intensieve transacties, die dan in stukken worden gehakt en separaat worden behandeld. Het is minder geschikt voor het bewerken van heel veel kleine transacties.
Map/Reduce is minder geschikt voor:

- Realtime processing (data wordt verwerkt zodra die beschikbaar komt);
- **Streaming data** (continue gegenereerde data);
- OLTP-transacties;
- Als de intermediate processen gegevens met elkaar moeten uitwisselen.

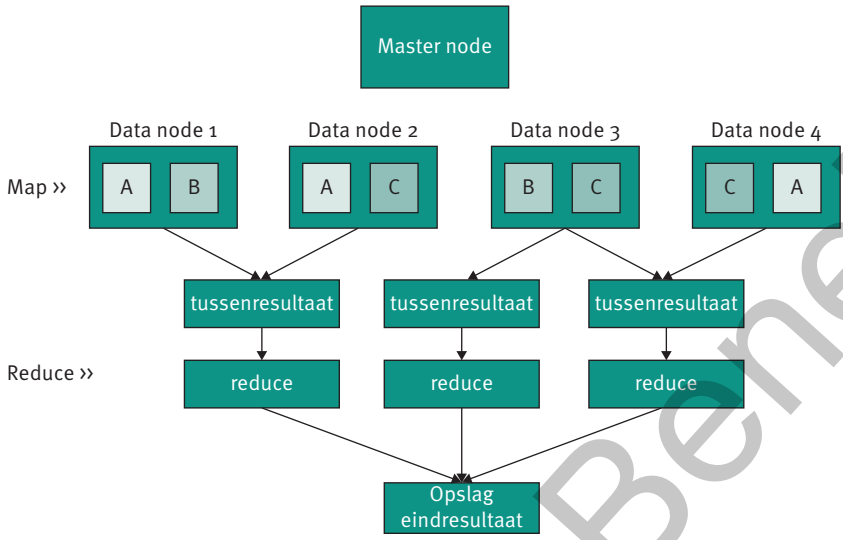
Een Hadoop-cluster bestaat uit vele gekoppelde machines die worden aangeduid als nodes. Zo'n cluster kent twee soorten nodes: masternodes en datanodes. Masternodes zijn er voor de organisatie, zij weten waar de data zich bevindt. Datanodes zijn, zoals het woord al aangeeft, de plaatsen waar de data is opgeslagen. De betreffende data wordt meerdere malen, verspreid over verschillende nodes opgeslagen. De voordelen hiervan zijn dat meerdere nodes parallel met deze data aan de slag kunnen (tijdwinst) en dat uitvallen van een node niet betekent dat vertraging wordt opgelopen (een andere node pakt dit op).

7.3 Het Big Data-ecosysteem

Big Data-technologie heeft voornamelijk een open-sourcekarakter. Veel software is ontwikkeld door de Apache Software Foundation. Dit is een non-profitorganisatie bestaande uit een netwerk van softwareontwikkelaars, die de software gratis ter beschikking stellen onder de Apache-licentie. Gevestigde leveranciers van relationele

(c) Pearson Benelux

databaseproducten bieden diensten en ondersteuning voor deze software, en hebben daarnaast hun eigen Big Data-software hierop gebaseerd door functionaliteit toe te voegen of af te stemmen op hun eigen softwareaanbod.



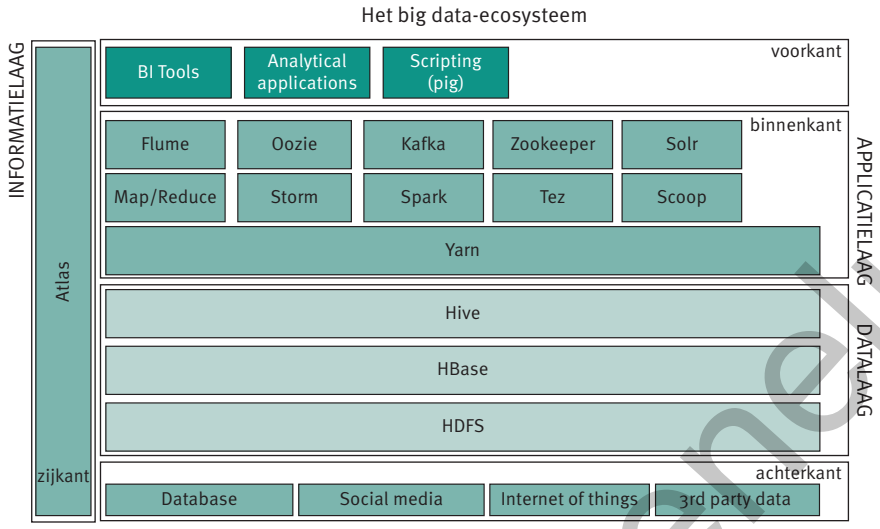
FIGUUR 7.1 Een Hadoop-cluster

Een typische Hadoop-distributie bevat de volgende componenten:

- Hdfs (Hadoop distributed file system): opslag van de data in een filestructuur;
- Map/Reduce: processing framework (alternatief: Spark);
- Hbase (NoSQL-database): opslag van de data in databaseformaat;
- Hive: datawarehouse, biedt SQL-toegang naar gedistribueerde data;
- Pig: scripting;
- Sqoop: transfer van gestructureerde data in/uit Hadoop;
- Solr: enterprise search;
- Oozie: voor het schedulen van jobs;
- Zookeeper: centraal management en services voor een Hadoop-cluster;
- Atlas: uitwisseling van metadata met tools en processen binnen en buiten Hadoop;
- Flume: logging van data;
- Kafka: publish and subscribe messaging.

Spark maakt gebruik van in-memory geheugen, waardoor het uitvoeren van query's beduidend sneller gaat dan bij Map/Reduce. Ook query's die gebruikmaken van gegevens opgeslagen op schijf (en niet aanwezig in geheugen), zijn door Spark veel sneller uit te voeren.
Spark en Map/Reduce zijn beide *computation engines* ('rekenmachines') die gebruikmaken van YARN (resource management). YARN is een laag bovenop de data (HDFS, Hbase, Hive).
Er zijn verschillende categorieën **NoSQL (Not Only SQL)-databases**. De bekendste staan hieronder vermeld.

(c) Pearson Benelux



FIGUUR 7.2 Onderdelen van een Hadoop-distributiesysteem: vele handen maken het werk licht

- Wide column stores

Bij de wide column stores wordt data niet georganiseerd in rijen, maar in kolommen. Dit levert met name voordelen op bij het analyseren van data, bij datacompressie en bij caching. Het idee om data in kolommen op te slaan bestaat dan ook al sinds de jaren negentig, maar heeft sinds 2000 een hoge vlucht genomen. Uiteraard heeft deze werkwijze ook nadelen: indien er gezocht wordt naar alle informatie die hoort bij 'Amsterdam' levert dit meer zoekwerk op vergeleken bij een rij-georiënteerde werkwijze.

- Document stores

Document stores zijn geschikt voor semigestructureerde data zoals documenten. In een document wordt alle data opgeslagen die bij elkaar hoort. De structuur van een document en het datatype kan per document verschillen en wordt bepaald door de verzameling data die je wilt opslaan. Document stores zijn anders gezegd **schemaless**. In een relationele database is er sprake van een uniforme opzet en structuur en worden bestanden met elkaar verbonden door Foreign Keys. Bij een document store zijn er geen Foreign Keys. Data die bij elkaar horen zijn opgeslagen in hetzelfde document.

- Key value databases

In een key value database wordt data (de value) opgeslagen aan de hand van een sleutel. Een document store kan worden gezien als een specifieke variant van een key value database. De waarde (value) is in dit geval een semigestructureerd document of gestructureerde data. Het voordeel van een document store versus een key value database is dat bij een document store de delen van de value (het document) opgevraagd kunnen worden, terwijl je bij een key value database altijd de hele waarde krijgt.

- Grafendatabases (graph databases)

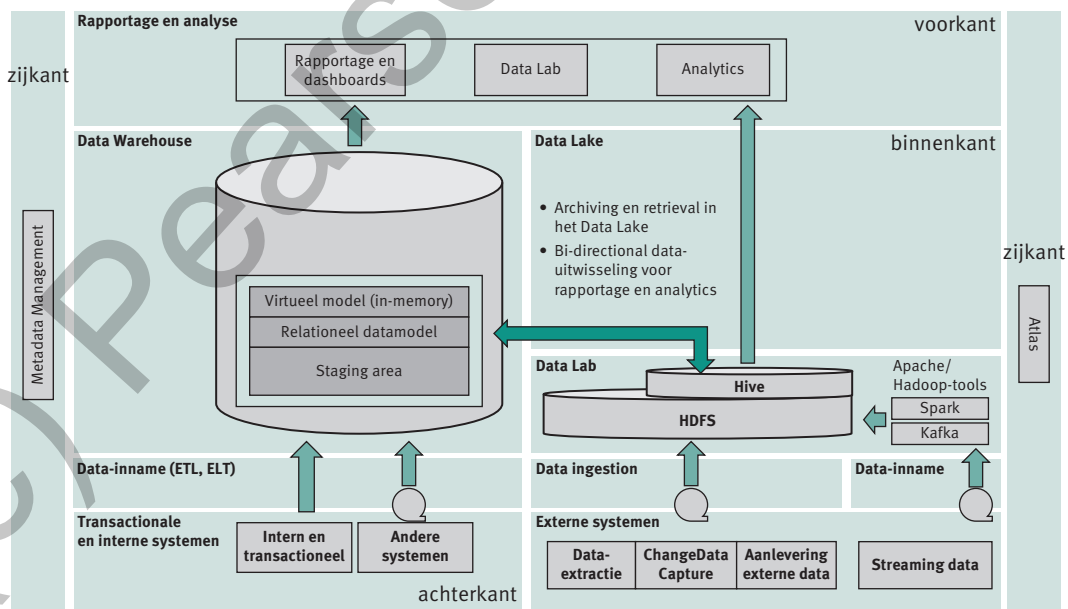
Een grafendatabase is een grafische weergave van nodes (entiteiten) en edges (relaties). Het onderscheidende van een grafendatabase is dat de relaties tussen de entiteiten

(c) Pearson Benelux

even belangrijk worden gevonden en apart worden opgeslagen. Hierdoor is het voor een grafendatabase veel makkelijker om relaties te leggen tussen entiteiten (nodes). In een relationeel model worden de relaties vastgelegd in de entiteit (als een Foreign Key). Dat heeft twee nadelen: naar de relatie moet worden gezocht in de entiteit, en de relaties liggen vast (zijn niet flexibel). Hiermee zijn grafendatabases beter geschikt om verbanden te leggen tussen entiteiten. Met name als veel entiteiten gekoppeld moeten worden loopt de performance van relationele systemen terug.
Opmerking: het begrip node in de context van grafendatabases duidt dus op een entiteit en wijkt daarmee af van het begrip node als onderdeel van Map/Reduce, waar het om een stuk hardware (server) gaat. De overeenkomst is dat in beide gevallen met node een onderdeel van een groter geheel wordt bedoeld.

7.4 Een hybride architectuur

De meeste organisaties hebben inmiddels de beschikking over een hybride architectuur. Het hybride karakter bestaat eruit dat het klassieke datawarehouse nu wordt gecompleteerd door een Data Lake. In de begindagen van het Data Lake was de opzet om gestructureerde data op te slaan in het datawarehouse en de semi- en ongestructureerde data in het Data Lake. Dat is nu niet meer zo. Opslag van gestructureerde data in het Data Lake is goedkoper. Ook kan data worden uitgewisseld tussen het datawarehouse en het Data Lake. Data die in Hive wordt opgeslagen kan met SQL worden benaderd. Vanwege al deze overwegingen is een hybride architectuur inmiddels een veel voorkomend verschijnsel.
Analytics en Business Intelligence vinden in zo'n omgeving vaak plaats in een Data Lab. Dit is de plaats waar alle data (gestructureerd, semigestructureerd en ongestructureerd) ter beschikking staan voor analyse en rapportage.



FIGUUR 7.3 Een voorbeeld van een hybride architectuur

(c) Pearson Benelux

7.5 Conclusie

In het huidige digitale tijdperk zijn de beperkingen van relationele databases duidelijk geworden. Relationele databases zijn nooit bedoeld voor het omgaan met ongestructureerde en semigestructureerde data. Waar data zich op vele verschillende plaatsen bevindt (gedistribueerd) en in (near-)realtime moet worden aangepast (e-commerce) zijn relationele oplossingen bovendien erg duur. Daarnaast laat het CAP-theorema zien dat de BASE-uitgangspunten waarop het relationale model is geënt niet houdbaar zijn in een gedistribueerde omgeving. Als oplossing en aanvulling zijn Data Lakes en Big Data-technologie ontstaan.

De opmars van hybride architecturen (het klassieke datawarehouse en een Data Lake) is niet verrassend. Een hybride architectuur biedt meer mogelijkheden om data op te slaan en te analyseren en de mogelijkheid om meer data en meer verschillende datatypen bij de analyse te betrekken.

VRAGEN

- Wat zijn de verschillen tussen BASE en ACID?
- Leg het CAP-theorema uit.
- Wat wordt bedoeld met een hybride architectuur?

OPDRACHTEN

1. Wide Column NoSQL-databases zijn bijvoorbeeld Hbase, Cassandra, Amazon, SimpleDB, Hypertable of Cloudera. Zet ze in een matrix en maak duidelijk wat de verschillen en de voor- en de nadelen zijn. Welke database zou je wanneer inzetten?
2. Redis, Chordless, Riak, Membase, Amazon DynamoDB, Voldemort en Scalaris zou je key value databases kunnen noemen. Welke zou jij kiezen en waarom?
3. CouchDB en MonetDB zijn document stores. Maak een kort essay over doel en inzet van deze twee databases. Wanneer zet je een dergelijke database in? Waar kun je dit soort databases op het internet met name vinden?
4. Voorbeelden van grafendatabases zijn AllegroGraph, DEX, HyperGraphDB, InfiniteGraph, InfoGrid of Neo. Zoek de *odd one out* en motiveer je keuze.
5. Wat is het verband tussen bitmapindexering en wide column databases?

Pearson | MyLab

Op **www.pearsonmylab.nl** vind je studiemateriaal en de eText
om je begrip en kennis van dit hoofdstuk uit te breiden en te oefenen.

(c) Pearson Benelux