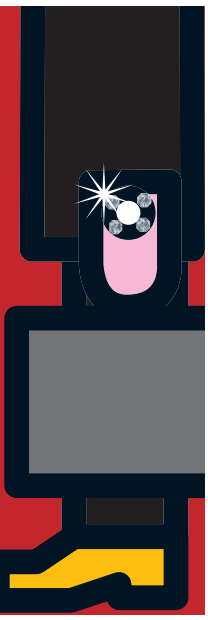

Database-technologie kraakt in al haar voegen

Anders denken over databases en query's

Martin Kersten en Teus Molenaar

In de loop der tijd zijn er meer toepassingen gekomen waarmee traditionele database systemen niet goed uit de voeten kunnen. Ze lopen op vele vlakken tegen hun (architecturale) beperkingen aan. Een andere manier van omgaan met data en query's is noodzakelijk. Evenals de gedachte dat er meerdere typen database systemen nodig zijn: 'one size does not fit all'.

De database management-systemen zoals we die in groten getale tegenkomen bij ondernemingen en instellingen (denk aan Oracle, DB2, SQL Server) zijn ontworpen in de jaren tachtig/negentig; voor hardware uit die tijd en voor de toenmalige applicaties. Lang is gedacht dat deze technologieën het eeuwige leven hebben en overal voor toepasbaar zijn. In de wetenschappelijke wereld heeft enkele jaren geleden al de opvatting post gevat dat *one size does not fit all* het credo moet zijn.
Een opvallende tekortkoming van de genoemde databases is dat zij alle records opslaan in rijen. Alle gegevens die bij een bepaald kenmerk horen, staan op een – als we het aanschouwelijk maken – horizontale lijn. Dat is uiterst lastig en inefficiënt als we deze databases gebruiken voor bijvoorbeeld Business Intelligence-toepassingen. Dan ben je namelijk niet geïnteresseerd in alle gegevens die bij een bepaald rekeningnummer horen – om een voorbeeld te geven – maar in alle transacties op een zekere dag of maand over alle rekeningnummers heen.

> De winst zit in de talloze uren die niet meer nodig zijn om een index te bepalen

Beter is het dan om een kolomgebaseerde database te hebben; juist voor BI-toepassingen. Daarbij worden de records verticaal opgeslagen en hoeft de database alleen een beperkt aantal kolommen te doorzoeken. Hier zien we al een onderscheid ontstaan in de soorten databases en hun bijzondere nut. De MonetDB, ontwikkeld op het CWI in Amsterdam, is een van de pioniers op dit vlak. In de praktijk blijkt dat een dergelijk ingerichte database een tijdwinst bij query's geeft van een factor tien en meer. Met een decennium aan ontwikkelingservaring is deze technologie stabiel genoeg voor concrete bedrijfstoepassingen.
Maar dat is niet genoeg voor toepassingen die hun intrede doen in het dagelijks leven.

Lampen en auto's

Stelt u zich een kantoorruimte voor met tal van lichtpunten en in de armaturen van de lampen zit een database die allerlei omgevingsfactoren, opgevangen door sensoren, opslaat. Tientallen lampen zijn uitgerust met zo'n 'ambient' (omgevingsgevoelige) database die vaak niet groter is dan enkele kilobytes. De lampen vormen samen een gedistribueerde database, want ze beïnvloeden elkaar. Er moet een regie zijn tussen al die *embedded* systeempjes om de goede balans te vinden van de lichtafgifte naar de kantoorruimte.
Eenzelfde soort embedded systemen is verwerkt in uw auto. In toenemende mate zijn voertuigen voorzien van sensoren en databases om bijvoorbeeld zuinig rijden af te dwingen of om zelfstandig te remmen als u te dicht op een voorganger rijdt of onhandig inparkeert.

Beide systemen kenmerken zich door grote aantallen kleine gegevensstromen die met elkaar te maken hebben om acties op elkaar af te stemmen. De historische informatie is hierbij een essentieel onderdeel en de database zorgt voor langdurige beschikbaarheid. Historische gegevens zijn bijvoorbeeld nuttig voor het realiseren van energiebesparing op basis van seizoenen. Ook zal er applicatiespecifieke code onder tijdsdruk moeten worden uitgevoerd.
Daar hoeft je niet aan te komen met de marktleiders in database management, door Mike Stonebraker aangeduid als de 'elephants'. SQLite komt misschien nog in de buurt. Natuurlijk

zijn er de afgelopen jaren innovatieve bedrijfjes geweest die een antwoord trachtten te formuleren voor deze uitdaging. En die worden dan weer opgekocht door de grotere, zoals Oracle dat deed met TimeStamp en BerkeleyDB, waardoor een gedifferentieerd pallet aan database functionaliteiten is aan te bieden.
Een breed geaccepteerde oplossing voor deze embedded toepassingen is er overigens nog niet.

Astronomisch

Aan de andere kant van het spectrum vinden we databases die in staat moeten zijn om juist erg grote bestanden op te slaan. Dat is het geval in de astronomie. Neem LOFAR: dit is een stelsel van duizenden gekoppelde kleine antennes met het doel signalen uit het heelal op te vangen. Het idee is terug te kunnen kijken naar het begin van de kosmos. De antennetjes vormen samen een 'software telescoop' met een diameter van 350 kilometer; met het centrum in Dwingeloo waar Astron (Astronomisch Onderzoek in Nederland) is gevestigd.
Hier gaat het inderdaad om astronomische getallen. De database moet kunnen beschikken over 500 TB aan opslag. En dagelijks komt er 1 tot 3 TB bij in de database-engine. Tegelijk mogen de data niet verloren gaan, want een signaal dat bijvoorbeeld 6 miljard jaar geleden is uitgezonden kan niet nog eens worden geregistreerd. LOFAR stelt de traditionele aanpak van databases voor grote uitdagingen. Prijs en performance voor wetenschappelijke applicaties zijn hier de doorslaggevende argumenten.

Vaak neigen gebruikers naar open source oplossingen. Zo is MySQL onbetwist dominant in de wereld van internet. Maar in beginsel wijkt de architectuur van MySQL niet af van de exemplaren die Oracle en IBM op de markt brengen.
Hoe lossen we dit op? Google en Microsoft kijken bijvoorbeeld naar de Big Table-benadering. Daarbij worden database-tabellen gesplitst in meerdere kleinere stukken, tabletten genaamd, die worden opgeslagen op verschillende computers binnen een Google File System cluster. Op die manier kunnen hele grote databestanden worden doorgeploegd, omdat het parallel gebeurt en het vrijwel nooit voorkomt dat alle servers tegelijk traag zijn met het geven van een antwoord. Dit werkt goed als je op zoek bent naar één bepaalde waarde (bijvoorbeeld een woord), een value query, maar schiet tekort als je een query doet naar meerdere waardes en hun onderlinge relatie.
De wetenschap zoekt de randen van wat nu mogelijk is op en probeert een antwoord te formuleren op de hedendaagse uitdagingen. Nederland, met het Centrum voor Wiskunde en Informatica, heeft in dit onderzoek wereldwijd overigens een leidende positie.

Bolwerk van database innovatie

Kortweg zijn er drie wegen die we kunnen inslaan. Alle drie vatten we samen onder de term *database cracking*. Een 'krakersbolwerk' dat de grote hoeveelheden gegevens op hun knieën krijgt.

Als eerste noemen we *index cracking* om het kostbare indexeringsprobleem de wereld uit te helpen. Vervolgens gaan we in op *query cracking* dat ertoe moet leiden dat query's effciënt worden uitgevoerd, of helemaal niet als de database weet dat het vinden van een antwoord op de gestelde vraag een tijdslimiet overschrijdt. Tenslotte gaan we in op *schema cracking*, dat de weg inslaat naar zelforganiserende (gedistribueerde) databases.

De kosten van het sorteren worden niet op voorhand gemaakt

Database-beheerders zijn veel tijd kwijt aan het kiezen van de juiste index voor de database-tabellen. Een gangbare methode is om tabellen te sorteren op een sleutelkenmerk, zodat records dan ook bij elkaar in de buurt blijven Dan kan het systeem immers snel zijn weg vinden als hij een query tracht te beantwoorden.
Bij een eenvoudige database is dat nog wel te doen, maar voor een goed gevulde database, laat staan een datawarehouse, is het heel veel werk. En als je het dan ook nog eens gaat gebruiken voor Business Intelligence wordt het monnikenwerk. Want het is welhaast onmogelijk tevoren in te schatten welke query's er dan komen en op grond daarvan te bepalen welke waarden of groepen van waarden bij elkaar moeten staan.
Ook de leveranciers worstelen met dit vraagstuk. Welke indices kun je laten vallen om de performance te verbeteren? Welke moet je toevoegen? En kan dit zonder het systeem onnodig te belasten of zelfs compleet stil te zetten? De oplossing wordt gezocht in het bouwen van *wizards*, een soort alter ego van de DBA die het handwerk overneemt.

Index cracking

Met 'index cracking' pakken we dit heel anders aan: we laten de index weg. Is het wel nodig om tevoren een index te bouwen, terwijl je niet eens weet waarnaar gevraagd gaat worden? In plaats hiervan nemen wij de applicatie (de query) als uitgangspunt. We hebben een database zonder index; alle gegevens worden op één hoop gegooid; zonder een directe adressering. Alles wordt simpelweg aan een tabel toegevoegd.
De eerste die een vraag stelt aan de database heeft pech, want hij kan wel even koffie gaan halen voordat er een antwoord komt. Zijn query wordt gebruikt om de database fysiek te reorganiseren, zodat het antwoord op de vraag een fysiek geclusterde deeltabel is. Als dat antwoord er eenmaal is, dan heeft de database al een verfijning aangebracht in zichzelf.
Op grond van een reëel gestelde vraag vanuit het gebruikersveld, gevolgd door de reorganisatie, is de volgende vraag sneller te beantwoorden.
De tweede die een vraag stelt, heeft al geen tijd meer om suiker in de koffie te doen. Naarmate er meer query's op de database worden losgelaten, zal hij zichzelf gaan organiseren naar de

reëel gestelde vragen. Uiteindelijk doet de indexloze database er net zo lang of kort over om met een antwoord te komen als zijn geïndexeerde broertje.
De winst van deze aanpak zit in de talloze uren die niet meer nodig zijn om een index te bepalen, te realiseren, en het effect ervan op de applicatie te controleren. De database-beheerder kan zijn tijd dan aan nuttiger zaken besteden.

Dit verhaal is goed te volgen als we uitgaan van een stabiele database. Maar die bestaan niet in de werkelijkheid. Elke dag, vaak elke seconde melden zich nieuwe tupels aan. Werkt het dan nog? Stel dat je bent begonnen met een hoeveelheid records die zichzelf aan het rangschikken is onder invloed van query's; een nuttig proces dat je vooral niet moet verstoren. Dat hoeft ook niet, want alle nieuwe tupels stop je in een aparte verzameling die ook wordt doorzocht bij elke query.

> Vasthouden van antwoorden en hergebruiken is bij de volgende query zeer profijtelijk gebleken

Pas op het moment dat er iets in dat aparte stapeltje zit dat van belang is voor een query wordt dit gedeelte overgeheveld naar de zichzelf organiserende tabel. Ook hier geldt dus dat de gegevens in de database pas een 'plek' krijgen als ze bewezen nuttig zijn. De kosten van het sorteren worden niet op voorhand gemaakt door te gaan gokken welke velden van belang zijn, maar pas op het moment dat het er werkelijk toe doet.
Experimenten hebben overtuigend aangetoond dat deze benadering realiseerbaar is in een open source DBMS, en zijn vruchten afwerpt.

Query cracking

Een volgende methode om de huidige database-uitdagingen aan te gaan, is 'query cracking'. Het onderliggende probleem is dat een database systeem elke query met evenveel respect behandelt. Hij zal steeds proberen op de beste en snelste manier tot een antwoord te komen. Zelfs al is de query op zo'n manier opgesteld dat het dagen duurt aleer een antwoord valt te verwachten, of als er een heel domme vraag wordt gesteld.
De bedoeling is om tot een systeem te komen waarbij het database management-systeem de query's in partjes opdeelt en één voor één uitrekent. Na elke stap krijgt de gebruiker een deelresultaat en een advies over de verwachte tijd om de rest uit te rekenen. Hiervoor moet het systeem wel in staat zijn een SQL-query op te breken in deelvragen en na elk antwoord op een deelvraag de gebruiker voor te leggen of hij met dat antwoord uit de voeten kan, of toch nog een specifieker antwoord zoekt. Waarna de query zich eventueel voortzet.

Vaak hebben mensen genoeg aan een 'bij benadering'-antwoord – bijvoorbeeld als ze op zoek zijn naar een trend – en is het niet nodig door te gaan tot vijf cijfers achter de komma. De architectuur van de database moet dan wel zo zijn gemaakt dat de database met (deel)antwoorden komt, en doorgaat met zoeken als blijkt dat een gebruiker nog niet tevreden is. De traditionele databases missen deze mogelijkheden. Zeker als een deelvraag pas op een veel later tijdstip wordt geactiveerd. MonetDB is hiervoor mogelijk beter geschikt, omdat de interne verwerking niet gebaseerd is op een *tuple pipeline.* Deelresultaten worden altijd gematerialiseerd en de code die de kernel aanstuurt is beter beïnvloedbaar. Om dit te bevestigen loopt er een proef met de Sloan Digital Sky Survey (SDSS). Deze wereldwijd beschikbare database van 'de sterren' bevat 2 TB catalogusdata en 9 TB beeldgegevens. De referentie implementatie werkt op SQL-server en de ontwerpers hebben de regel gesteld dat elke niet-astronoom alleen query's mag loslaten die binnen tien minuten een antwoord genereren. Als het langer duurt, dan wordt de query gewoon afgebroken. Astronomen krijgen maximaal een uur om de database lastig te vallen.
Een analyse van een maand query's (1,3 miljoen) laat zien dat sterrenkundigen eigenlijk steeds inzoomen op de database. Ze willen vrijwel altijd maar een stukje zien van de vorige selectie. Met dit inzicht is het vasthouden van antwoorden en hergebruiken bij de volgende query zeer profijtelijk gebleken. Anekdotisch is de gebruiker die zonder er verder bij na te denken een cross-product query over de centrale feitentabel wilde uitrekenen. Dat zou 200 x 200 miljoen kandidaat antwoorden opleveren als het systeem niet de time out had gebruikt om dit proces te stoppen.

Schema cracking

Een derde methode is 'schema cracking'. Ook hierbij gaat het om zelforganisatie van databases. Gewoonlijk, als een database erg groot wordt, maakt een database-beheerder een gedistribueerde of gepartitioneerde database. Hij bepaalt waar wat op welke machine komt te staan, opdat later alles is terug te vinden of op de juiste manier is te doorzoeken. Dat is, op zijn zachtst gezegd, een heel karwei.
'Schema cracking' is een techniek om databases in onderdelen te splitsen en te beheren zonder tussenkomst van een beheerder. We beginnen met een limiet te stellen aan de hoeveelheid records in een database-instantie. Als dit een niveau bereikt dat de doorzoekbaarheid bemoeilijkt en waardoor de prestaties afnemen, dan brengt het database-systeem zelf een deel over naar een andere machine. En dat proces herhaalt zich. Er ontstaat vanzelf een gedistribueerde database. Elke extra database weet alleen maar dat hij is afgescheiden van zijn moeder (of vader); die relatie is bekend. Een query kan nu naar elk machine worden gestuurd, waar een deeloplossing wordt gegenereerd en een plan de campagne om de rest op te halen. Query cracking is hier dus noodzakelijk, evenals de feedback aan de gebruiker op tussenresultaten.

Anders denken

De aangedragen oplossingen verkeren nog in het stadium 'doe dit niet thuis' of 'raadpleeg uw huisarts', maar wijzen niettemin een uitweg uit de bestaande prestatieproblemen van traditionele databases in vooral BI-toepassingsgebieden. Ze zijn hoopgevend en zonder twijfel toepasbaar, zoals aangetoond in de context van ons onderzoek met MonetDB.

De gemeenschappelijke factor van de drie aanpakken is dat ze het anders denken over databases bevorderen en aantonen dat vasthouden aan bestaande technologieën een belemmering kan zijn voor innovatie. Het toepassen van deze open source oplossingen in delen van de bedrijfsautomatisering is niet meer een kwestie van 'of', maar van beleid bepalen 'wanneer'. *First Market Movers* hebben hiermee een streepje voor.

Martin Kersten en Teus Molenaar

Martin Kersten is hoofd van de afdeling Informatiesystemen bij het Centrum voor Wiskunde en Informatica. Teus Molenaar is freelance journalist.

Update

SQL Anywhere op weg naar versie 11

Een database die we wat uit het oog zijn verloren is SQL Anywhere en zijn derivaten Ultralite en UltraliteJ. Ontwikkelaar iAnywhere heeft hiermee een erg complete database op de markt gezet dat wordt toegepast in mobiele omgevingen; het concept van iAnywhere is gebaseerd op het gebruik van remote entry devices met SQL Anywhere als basis en MobiLink als regisseur van het draadloze netwerk, dat kan bestaan uit laptops, mobiele telefoons, PDA's, enzovoort. Dat de technologie wereldwijd zijn weg naar de gebruikers heeft gevonden blijkt uit de jaarlijkse stijging van omzet en winst, in 2005 werd een groei van 20 procent gerealiseerd – en zeker ook uit het feit dat inmiddels versie 10 op de markt is gebracht. Naast 2500 nieuwe klanten werden onlangs de twee grootste mobiele implementaties ter wereld gerealiseerd, de een bij het Amerikaanse Census met ruim een half miljoen gebruikers, de ander bij de Chinese douane waar alle in- en uitgaande goederen op zwaar beveiligde manier via iAnywhere worden geregistreerd. Er kwamen het afgelopen jaar 148 nieuwe partners bij en de community begroette bijna 23 duizend nieuwe leden.

iAnywhere werkt op dit moment onder de codenaam Panorama aan de ontwikkeling van versie 11.

iAnywhere is in de VS en het Verre Oosten een bijna volle dochtermaatschappij van Sybase; in Europa is iAnywhere niet meer dan een min of meer zelfstandige divisie waarbij Sybase zelf de grootste reseller van de mobiele productlijn is. Daarnaast afficheert moederbedrijf Sybase zich als de marktleider op het gebied van mobiele databases, terwijl die kwalificatie feitelijk dochter/divisie iAnywhere toekomt. Ingaande 2008 wil men een einde maken aan deze voor resellers, partners en gebruikers onduidelijke en verwarrende toestand: iAnywhere zal zich geheel toeleggen op de mobiele lijn, terwijl Sybase zich verder focust op Enterprise Solutions zoals de ASE en IQ.

Sun en Oracle leveren gebruiksklaar datawarehouse

Oracle en Sun Microsystems kondigen de beschikbaarheid aan van het Oracle Optimized Warehouse for Sun. Dit is een complete en gebruiksklare datawarehouse-oplossing, een geoptimaliseerde combinatie van de Oracle database, het Solaris-besturingssysteem en Sun's krachtige hardware.

De totale oplossing bestaat uit de Oracle Database Enterprise Edition, Oracle Real Application Clusters, Oracle Partitioning op een Sun Fire E20K server, het Sun open source Solaris besturingssysteem (OS) en Sun StorageTek 6540 array's.

Met alle hardware/softwarecomponenten vooraf geïnstalleerd, geconfigureerd en geoptimaliseerd, is Oracle Optimized Warehouse for Sun een snelle, eenvoudige en veilige manier om een datawarehouse in te richten. Oracle Optimized Warehouse for Sun is bedoeld voor datawarehouses met een omvang van 10 TB aan ruwe data. Toekomstige configuraties kunnen meer data aan door het gebruik van modulaire bouwstenen.

HP en Informatica leveren BI-oplossing

Informatica PowerCenter is een geïntegreerd onderdeel in de nieuwe Enterprise Risk Management-oplossing van HP; een BI-oplossing voor financiële dienstverleners. PowerCenter levert het fundament voor high-performance data-integratie van deze oplossing.

Hierdoor wordt het in kaart brengen van bedrijfsrisico's vereenvoudigd, waardoor onder andere de kwaliteit van data in wet- en regelgeving-initiatieven gegarandeerd kan worden.

De HP Enterprise Risk Management-oplossing is 'pre-integrated' met HP Neoview, Informatica PowerCenter, het analytische dashboard van MicroStrategy en het datamodel van Quadrant.

Nieuw op de website van DBM

Weblogs van Hans Lamboo en Paul van der Linden