Gebrek aan standaard bemoeilijkt keuze multidimensionale oplossing

De magie van de kubus verdwijnt

Frank Habers

De kubus (of multidimensionele database) is jarenlang de succesfactor geweest voor veel Business Intelligence-oplossingen. Na het verzamelen en integreren van data in een centraal relationeel datawarehouse, worden deze data vervolgens vaak in een kubus aan de gebruiker beschikbaar gesteld. De kubus gebruikt daarbij een eigen (proprietary) opslagstructuur, waardoor een hoge performance gegarandeerd is. Bovendien bieden de (kubus gerelateerde) OLAP-applicaties vaak uitgebreidere analysefunctionaliteit dan relationele (op SQL gebaseerde) toepassingen. Deze twee kenmerken doen het uiteraard erg goed bij gebruikers, die dan ook erg enthousiast zijn over hun kubussen.

Het succes van de kubus heeft veel leveranciers van relationele databases ertoe bewogen kubustechnologie in te kopen. Waar de kubus in eerste instantie een los product was in het aanbod van de database-leveranciers, wordt de kubustechnologie inmiddels steeds verder geïntegreerd in de relationele database. Daarnaast wordt vaker functionaliteit voor het analyseren van relationele data met een hoge performance in de database geïmplementeerd. De grenzen tussen relationele en dimensionele technologie vervagen daarmee en een keuze voor een van de twee is steeds moeilijker te maken. Dit artikel zet de belangrijkste ontwikkelingen rond de kubus op een rijtje en geeft handvatten bij de keuze voor een relationele of multidimensionale oplossing.

Nadelen

In de jaren negentig heeft een groot aantal leveranciers kubusproducten (ook wel MDB OLAP- of MOLAP-toepassingen genoemd) op de markt gebracht. Deze producten bestaan uit een combinatie van een gegevenslaag (de kubus) en een applicatie- en presentatielaag. Ieder product heeft zijn eigen kenmerken met bijbehorende voor- en nadelen. Alle producten boden in die tijd echter twee aansprekende voordelen ten opzichte van de traditionele, op SQL gebaseerde oplossingen, namelijk een hoge performance en meer analytische functionaliteit. Het zijn ook deze twee aspecten die de kubus zo succesvol hebben gemaakt.

Tijdens diverse implementaties werden echter ook de nadelen en de beperkingen van de kubussen duidelijk. De nadelen verschillen per product, maar ieder product brengt in ieder geval meerdere van de volgende nadelen (en beperkingen) met zich mee:

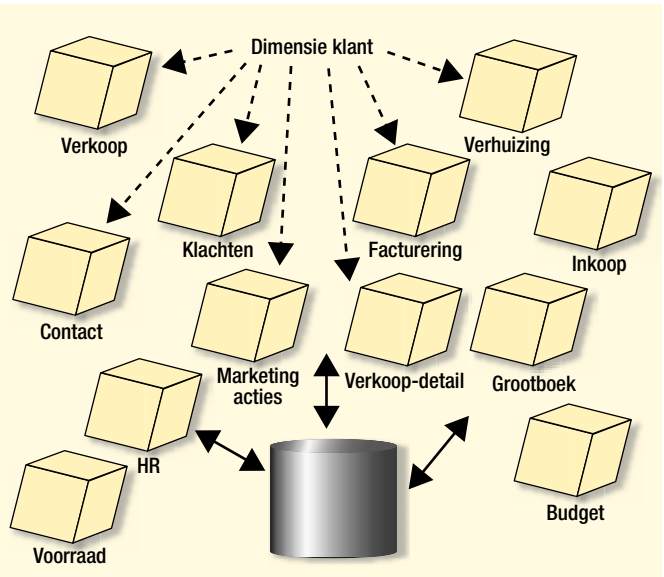
- Een kubus bevat 'beperkte' data. Kubussen bevatten vaak wel analyse-objecten als product, productgroep, klant, regio en woonplaats, maar geen beschrijvende attributen, zoals het telefoonnummer, e-mailadres en straat en huisnummer van een klant. Een adressenlijst uitdraaien na een analyse is dus niet mogelijk vanuit een kubus;
- Analyseren over meerdere kubussen is lastig. Een kubus bevat meestal één onderwerp per kubus, bijvoorbeeld een kubus voor verkoop, facturatie, inkoop, contactmomenten, enzovoort. Het is vaak lastig om eenvoudig over kubussen heen te rapporteren, of dat op zodanige wijze te doen dat dit niet zichtbaar is voor een eindgebruiker (conform het analyseren over meerdere relationele tabellen);

> Performance wordt daarmee steeds minder een argument om voor kubussen te kiezen

- Omvang van kubussen. Een bekend fenomeen is dat kubussen sterk in omvang kunnen toenemen. Dit is vaak het gevolg van het opnemen van veel en grote dimensies in combinatie met veel meetwaarden. De mogelijke omvang van de kubus is echter beperkt, waardoor alleen geaggregeerde informatie in de kubussen kan worden opgenomen en de analysemogelijkheden beperkt worden;
- Sterke groei van het aantal kubussen. Door het grote aantal onderwerpsgebieden en de opdeling van kubussen als gevolg van de beperking ten aanzien van omvang, kan het aantal te beheren kubussen sterk groeien (zie voorbeeld afbeelding 1);

- Redundante informatie in verschillende kubussen. Geconformeerde dimensies komen in veel kubussen terug. De klantdimensie zal bijvoorbeeld zijn opgenomen in de verkoop-kubus, de marketingactie-kubus, de financiële kubus en de klachtenkubus. En dan worden de andere dimensies nog buiten beschouwing gelaten. Het nadeel is dat veel redundante informatie ontstaat en dat de definitie van de dimensie (inclusief levels, hiërarchieën en dergelijke) op meerdere plaatsen onderhouden moet worden (zie afbeelding 1);
- Impact van wijzigingen op opslag. De kubusdefinitie kan in de loop van de tijd wijzigen. Bij sommige producten betekent een wijziging (bijvoorbeeld een dimensie toevoegen, een hiërarchie wijzigen, meetwaarden verwijderen) dat de kubus geheel opnieuw moet worden opgebouwd. Dit kan de nodige doorlooptijd en inspanning vergen;
- Analyseren over historie van dimensies. Met behulp van een kubus kunnen heel goed trendanalyses op meetwaarden worden gedaan via de tijdsdimensie. Er bestaat echter ook een andere vorm van historie en dat is de historie van dimensies (slowly changing dimensions). Kubussen hebben veelal geen faciliteiten om deze historie elegant vast te leggen en te analyseren;
- Beperkingen van de kubus-interface. De kubus-interface is logischerwijs gebaseerd op het analyseren van meetwaarden door het combineren van dimensies. De applicatie-interface is hier dan ook specifiek op gebaseerd en de opmaakmogelijkheden zijn vaak beperkt.

De bovenstaande lijst maakt duidelijk dat kubussen de nodige beperkingen met zich mee brengen en zwaar drukken op de inspanningen voor beheer en onderhoud. Gelukkig staan de ontwikkelingen in de techniek niet stil. Doordat database-leveranciers de kubus integreren met de relationele database, gelden de hiervoor genoemde nadelen en beperkingen in mindere mate. Het is ontegenzeggelijk een voordeel dat relationele en multi-



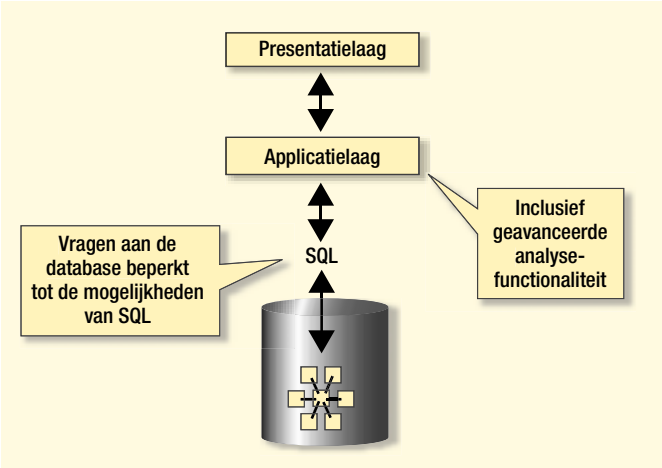
Afbeelding 1: Kubus per onderwerp.

dimensionele data (en metadata) worden beheerd in één omgeving. De relationele database zelf heeft zich echter ook sterk ontwikkeld op het gebied van performance en analytische functionaliteit, waardoor de behoefte aan kubussen misschien wel veel minder is. We komen daarmee voor een moeilijke keuze te staan: relationeel of dimensioneel? Hebben we aan één van de twee voldoende of hebben we misschien allebei nodig?

ETL-hulpmiddelen

In de praktijk wordt vaak gekozen voor een datawarehouse-architectuur, waarbij de data in het (centrale) datawarehouse relationeel worden opgeslagen. Gezien de beperkingen van kubussen is het ook niet aan te raden om hierbij te kiezen voor multidimensionele opslag. De relationele omgeving biedt immers veel uitgebreidere mogelijkheden voor het beheer van data en autorisaties, het wijzigen van gegevensstructuren, het vastleggen van historie en het vastleggen van het laagste niveau van de feiten (bijvoorbeeld orderregels, financiële boekingen, call detail records, enzovoort). Bovendien ondersteunen de ETL-hulpmiddelen die in gebruik zijn met name relationele databases.
Zijn er naast de relationele opslag dan nog kubussen nodig? De laatste jaren hebben database-leveranciers immers een groot aantal features ontwikkeld om de query-performance te verbeteren, zoals:

- Aggregatietabellen. Databases bieden de mogelijkheid om automatisch aggregatietabellen (ook wel materialized views, automatic summary tables en indexed views genoemd) bij te werken en te onderhouden. De query-optimizer bepaalt automatisch aan de hand van het SQL-statement welke aggregatietabel kan worden gebruikt. Het elegante aan deze oplossing is dat de functionaliteit transparant is voor de gebruiker en applicaties (want die verwijzen gewoon naar de detailtabel, terwijl de aggregatietabel wordt gebruikt). De oplossing kan ook worden gezien als een soort eenvoudige kubus in relationeel formaat;
- Starjoins. Indien een sterschema wordt gehanteerd, kan de query-optimizer een efficiënter joinpad gebruiken, namelijk door eerst een cartesiaans product van de relatief kleine dimensies te maken en vervolgens met dit resultaat een join naar de (grote) feitentabel te maken;
- Partitionering. Tabellen worden hierbij opgedeeld, zodat de afzonderlijke tabellen kleiner zijn en de performance verbetert als een SQL-statement een enkele partitie vereist. De kunst is dus om de tabellen zodanig in te delen, dat het SQL-statement zo min mogelijk partities nodig heeft. Indelingen in de tijd (partitie per maand, jaar) zijn zeer gebruikelijk, omdat analyses zich vaak beperken tot een periode;
- Parallel processing. De query's worden opgedeeld in deelvragen die tegelijkertijd worden uitgevoerd. Het resultaat van de deelvragen wordt vervolgens weer gecombineerd. De performance verbetert door de parallelle uitvoering van het statement;
- Indexering. Ten behoeve van query-performance zijn specifieke indexen ontwikkeld, bijvoorbeeld zeer snelle en efficiënte bitmap-indexen.



Afbeelding 2: Analytische SQL-gebaseerde toepassing.

Deze lijst is niet compleet, maar geeft een goed beeld van de mogelijkheden om de performance in een relationele omgeving op efficiënte wijze sterk te verbeteren.
Naast deze verbeteringen wordt hardware ook steeds sneller. De combinatie van deze factoren maakt het in veel situaties mogelijk om via een relationele oplossing voldoende performance te bieden. Performance wordt daarmee steeds minder een argument om voor kubussen te kiezen.
Als performance geen argument meer is om voor een kubus te kiezen, is het kennelijk een keuze voor functionaliteit geworden. Traditioneel bieden OLAP-applicaties vaak meer analytische functionaliteit dan op SQL gebaseerde oplossingen. Niet alleen op het gebied van *drill down* en *slice & dice*, maar ook wat betreft analytische functies als tijdreeks- en statistische analyses. OLAP-toepassingen bieden vaak veel standaardfunctionaliteit die met een op SQL gebaseerde oplossing slechts moeizaam te implementeren is (denk bijvoorbeeld aan standaard *year to date* analyses over de tijddimensie). Is dit soort specifieke functionaliteit vereist, dan zijn we dus aangewezen op een OLAP-toepassing, die ons wel verplicht de data eerst in een kubus te stoppen. Er wordt dus niet zozeer gekozen voor een ander opslagformaat (de kubus) maar voor specifiek benodigde functionaliteit.
Uiteraard onderkennen de leveranciers van (op SQL gebaseerde) query- en reportingtools deze problematiek ook en proberen ze hun producten functioneel uit te breiden. De nadelen van SQL als vraagtaal worden daarbij omzeild door functionaliteit op applicatieniveau in te bouwen waarmee geavanceerde analyses op de *result set* (het resultaat van de SQL query) mogelijk zijn. Hiermee is zonder gebruik te maken van fysieke opslag in aparte kubussen, toch geavanceerde analytische functionaliteit beschikbaar (zie afbeelding 2). De beperking van deze oplossing zit vooral in het feit dat alleen relatief eenvoudige vragen (inclusief filters) aan de database kunnen worden gesteld (de beperking is immers SQL), waardoor in sommige situaties inefficiënt veel data aan de applicatielaag worden aangeboden. In de praktijk blijkt dat deze oplossingen vaak een heel groot deel van de benodigde functionaliteit afdekken, waarbij de kosten voor ontwikkeling en onderhoud relatief laag zijn (er hoeven immers geen aparte kubussen geïmplementeerd en onderhouden te worden). Bovendien bieden dit soort hulpmiddelen veel functionaliteit die in OLAP-applicaties ontbreekt (met name voor de opmaak, scheduling en distributie van grote hoeveelheden rapportages). Als er een datawarehouse geïmplementeerd is, is het dus mogelijk om met een relatief kleine investering veel functionaliteit te bieden via één hulpmiddel. Vanuit kostenoogpunt is het daarom aan te raden altijd een dergelijk hulpmiddel te implementeren voor de 'basisfunctionaliteit'. Daarnaast hoeft alleen naar een kubusoplossing gegrepen worden als deze functionaliteit biedt die ontbreekt in het geïmplementeerde query tool en die onmisbaar is voor de Business Intelligence-toepassing.
Als voor het gebruik van een kubus wordt gekozen, lijkt het meer en meer voor de hand te liggen om voor een oplossing van een database-leverancier te kiezen, gezien de eerder genoemde voordelen. Een bijkomend voordeel van de integratie van de kubus in de relationele database is dat de relationele- en OLAP-metadata nu op één centrale plek vastliggen. Alle applicaties op het datawarehouse kunnen gebruik maken van deze metadata zonder deze nog een keer apart te definiëren en belangrijker, te onderhouden.

> De nadelen van SQL worden omzeild door functionaliteit op applicatieniveau

De keuze voor een specifieke database-leverancier is echter automatisch ook een keuze voor een specifieke architectuur.
Een standaard als SQL is er voor kubusimplementaties nog niet en database-leveranciers als IBM, Oracle en Microsoft volgen nog verschillende strategieën op dit gebied.

IBM en Oracle

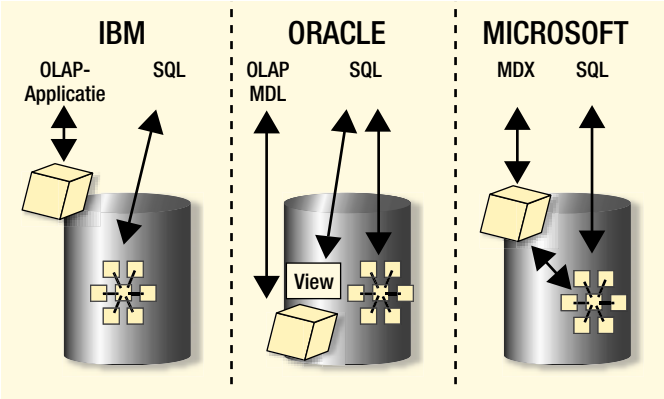
IBM neigt sterk naar een strategische keuze voor een relationele oplossing. IBM heeft als kubusoplossing DB2 OLAP Server, maar biedt in DB2 UDB versie 8.1 zeer sterke functionaliteit om te concurreren met de kubus. De naam van deze oplossing is veelzeggend: Cubeview. De analysemogelijkheden zijn hierbij sterk uitgebreid door SQL-extensies op te nemen (bijvoorbeeld rank, rownumber en geavanceerde aggregatiefuncties) en OLAP-metadata in de database op te nemen. Daarnaast biedt IBM slimme functionaliteit om de performance van de relationele omgeving sterk te verbeteren. Naast de eerder genoemde functies biedt DB2 UDB mogelijkheden als *multidimensional clustering* (MDC, een slimme en efficiënte opslagmethode), maar ook SQL-extensies die de efficiency vergroten, zoals *rollup* (waarmee via één SQL-statement gegevens op verschillende aggregatieniveaus worden berekend, dit in tegenstelling tot *group by*).

Oracle biedt in Oracle9i release 2 ook een nieuwe oplossing. Oracle Express is als apart product verdwenen en is opgenomen in de relationele database in een zogenaamde *analytical workspace*. Alle faciliteiten van het DBMS (onder andere managementtools, backup/recovery en autorisatie/authenticatie) zijn automatisch ook beschikbaar voor de kubussen. Via de vraagtaal Oracle DML (de multidimensionale tegenhanger van SQL) kunnen complexe analyses op kubussen worden uitgevoerd. Maar Oracle gaat nog verder, het is namelijk mogelijk om een relationele view van de kubussen te maken, zodat met SQL vragen aan kubussen kunnen worden gesteld. Daarbij is het zelfs mogelijk multidimensionale data te koppelen met relationele data. Daarnaast heeft ook Oracle de SQL-functionaliteit uitgebreid. Met deze uitbreidingen biedt Oracle de mogelijkheid om bestaande investeringen in SQL (in tools, kennis en methodieken) te blijven benutten. Daarnaast wordt het onderscheid tussen relationeel en kubus voor de gebruiker onzichtbaar. Geconcludeerd kan worden dat Oracle de keuze voor relationele of multidimensionale opslag aan de klant laat, waarbij de keuze voor multidimensioneel niet automatisch de keuze voor de vraagtaal (of applicatie) bepaalt en dat is zeker een pluspunt.

Microsoft biedt de klant op eenvoudige wijze de keuze voor multidimensionale (MOLAP), relationele (ROLAP) en Hybride data-opslag (HOLAP), waarbij de keuze transparant is voor de gebruiker. Deze keuze is een afweging tussen performance en opslag. De vraagtaal voor de (virtuele) kubussen is MDX (Multidimensional Expression). MDX lijkt wat betreft syntax-opbouw sterk op SQL (SELECT.. FROM ..WHERE..), maar biedt net als Oracle MDL meer analyse- mogelijkheden dan SQL. Alhoewel het technisch mogelijk is om met een subset van SQL de kubussen te benaderen, kiest Microsoft duidelijk voor MDX als vraagtaal voor analytische functionaliteit. Dit betekent dat ook bij een ROLAP oplossing (virtuele kubussen) gebruik wordt gemaakt van MDX.

Geen standaard

Met de bovenstaande opsomming wordt niet beoogd een voorkeur uit te spreken voor een leverancier. Belangrijk is dat duidelijk wordt dat de keuze voor een kubus van een database-leverancier andere keuzes tot gevolg heeft (zie ook afbeelding 3). Het grootste probleem daarbij is dat de vraagtaal voor kubussen niet gestandaardiseerd is. Microsoft en Oracle bieden hiervoor immers twee verschillende oplossingen (MDX versus DML). De keuze voor een kubus van een database-leverancier, betekent dus automatisch een keuze voor een vraagtaal en daarmee een beperking in de beschikbare applicaties die op de kubus draaien. Dit in tegenstelling tot relationele data, waarvoor gebruik wordt gemaakt van de standaard vraagtaal SQL. Het mooiste zou eigenlijk een combinatie van de oplossingen van Oracle en Microsoft zijn, waarbij de grens tussen relationele en multidimensionele opslag steeds verder vervaagt en er een standaard ontstaat als SQL maar dan voor OLAP-analyses. Deze vraagtaal kan dan met name worden



Afbeelding 3: Vergelijking SQL gebaseerde toepassing.

gebruikt voor analytische functionaliteit die niet eenvoudig met SQL is op te lossen. De keuze voor relationele of multidimensionele opslag wordt dan puur een kwestie van performance. Voor analytische toepassingen kan dan immers zowel voor SQL als de multidimensionele vraagtaal gekozen worden, zonder rekening te houden met de fysieke opslag van de data.

Conclusie

Geconcludeerd kan worden dat het niet meer vanzelfsprekend is om kubussen te gebruiken in datawarehouse-omgevingen. Door de snelle ontwikkelingen van relationele databases, SQL, hardware en query tools wegen de twee belangrijkste argumenten om voor een kubus te kiezen, performance en functionaliteit, vaak niet meer op tegen de nadelen. Door een geavanceerd query- en reportingtool op een goed geoptimaliseerde relationele database te implementeren kan vaak met beperkte inspanning en tegen lage kosten een groot deel van de gewenste (analyse)functionaliteit worden geboden. Is deze functionaliteit echt niet voldoende, dan kan daarnaast voor een OLAP-oplossing worden gekozen waarbij gegevens in een kubus (of virtuele kubus) opgeslagen moeten worden. Welke technologie daarvoor gebruikt moet worden is echter een moeilijke keuze. Het ligt voor de hand om te kiezen voor de oplossing van een database-leverancier waarbij de kubus is geïntegreerd in de database met alle voordelen daarvan. Omdat een standaard ontbreekt, betekent dit echter meteen een keuze voor een specifieke vraagtaal en architectuur en daarmee een beperking in beschikbare toepassingen. Een reden te meer om een weloverwogen keuze te maken of deze keuze nog even uit te stellen totdat er meer zicht is op een standaard. Duidelijk is in ieder geval dat de relationele database de aanval van de kubus heeft gepareerd en heeft omgezet in een samenwerking. Daarmee verdwijnt definitief de magie van een kubus.

Drs. Frank Habers (fhabers@inergy.nl) is directeur van Inergy Analytical Solutions en consultant op het gebied van Business Intelligence en Datawarehousing.

Informatie over verschillende OLAP-oplossingen is te vinden op de website www.olapreport.com, van onafhankelijk consultant Nigel Pendse.