Contactlenzen en datamarts vertonen overeenkomst

Sterschema's als starschema's

Harm van der Lek

In dit artikel gaan we eerst de klassieke discussie nog even recapituleren: moet je een datawarehouse nu genormaliseerd ontwerpen (Inmon) of niet (Kimball). En zo ja, hoe ziet zo'n genormaliseerd datawarehouse er dan uit? Voor dit laatste is tegenwoordig het toverwoord: Data Vault oftewel gegevenskluis. Dan Linstedt is de bedenker van deze modelleringstechniek en in een volgend artikel gaan we daar dieper op in.

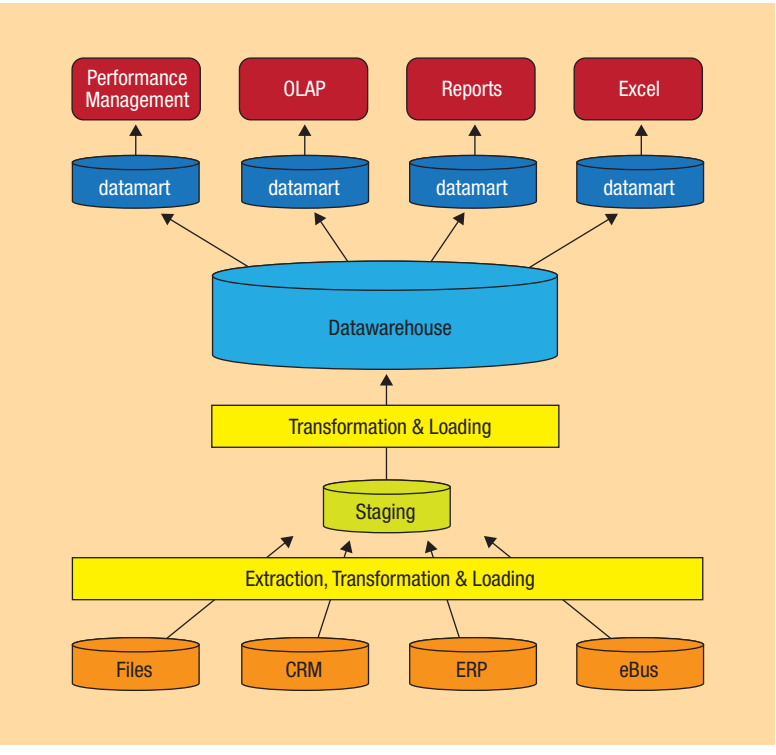
Hier bekijken we, na bovenstaande discussie beslecht te hebben, de consequenties daarvan voor het denken over datamarts. Deze blijken vaker wegwerpartikelen te worden.

Velen zijn het erover eens dat de ideale architectuur voor een standaard BI-omgeving er een is zoals in afbeelding 1 is te zien. Het bestaat uit een centraal datawarehouse (ook wel Enterprise Datawarehouse genoemd: EDWH) en daarbovenop één of meer datamarts. De gedachte hierachter is dat in het EDWH één versie van de waarheid kan worden opgebouwd, inclusief de historische ontwikkeling van bepaalde data. De verschillende datamarts, per definitie op specifieke gebruikersgroepen gerichte omgevingen, kunnen hier dan uit worden afgeleid (fysiek of via views). Zo kan worden gegarandeerd dat de resultaten van de verschillende datamarts, voor zover vergelijkbaar, met elkaar in overeenstemming zijn. Waar kan dit goed voor zijn?

Stel u eens een directiemeeting voor waarin de omzetcijfers van de laatste maand worden besproken. Nu blijkt dat de totalen uit de datamart van de productiemanager verschillen van die van de salesmanager (guess what, die van de laatste blijken hoger dan die van de eerste). Ruzie in de tent, opdrachten van boven naar beneden om uit te zoeken hoe dit kan, overuren van stafmedewerkers en IT'ers. Voor nieuwsgierigen onder u: het bleek uiteindelijk dat bij de bouw van de datamart voor de verkoopbaas geen rekening was gehouden met de retouren. Dit voorbeeld laat zien dat er in onderlinge inconsistente datamarts een gevaar voor de business schuilt, want onenigheid kost geld.

Bedrijfsbreed onhaalbaar

Alhoewel dit dus een ideale opzet lijkt, komt het in de praktijk toch maar erg weinig voor dat zo'n EDWH echt bedrijfsbreed is ingezet. Eigenlijk is dat wel begrijpelijk; vergelijk maar met 'gewone' informatiesystemen. Vroeger hadden we ook nog wel ideeën over een bedrijfsbrede operationele database, waar dan alle applicaties op zouden aansluiten: alle gegevens redundantievrij op precies één plaats onderhouden en gebruikt. Dit ideaal is natuurlijk niet haalbaar gebleken, en wellicht zelfs niet helemaal wenselijk. Bovendien wat is 'het bedrijf'? Is dat heel Philips of Shell? Nee, dus daar zou het hooguit per business unit haalbaar zijn. Bovendien, bedrijven splitsen, fuseren enzovoort. En verder is er altijd de vraag wie het allemaal betaalt. Dit geldt natuurlijk ook voor een datawarehouse. De enigen die het belang zouden moeten inzien van een EDWH zijn de topmanagers.



Afbeelding 1: De hub-and-spoke architectuur.

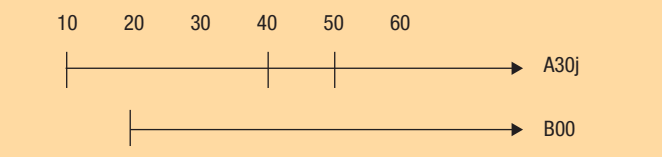
Product Key	Product Code	Geldig Van	Geldig Tot	Courant	Verpakking
1	A30j	10	40	Ja	Dozen
2	A30j	40	50	Ja	Kratten
3	A30j	50	9999	Nee	Kratten
4	B00	20	9999	Ja	Flessen

Afbeelding 2: Tabel met producthistorie.

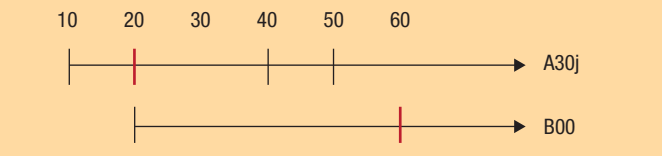
De onhaalbaarheid van een bedrijfsbreed EDWH mag echter niet het streven ernaar in de weg staan. Vooral organisaties (of bepaalde grote onderdelen daarvan) met een aantal invloedrijke informatiearchitecten slagen er dan ook wel degelijk in. De sleutel tot succes is ook hier weer 'denk groot, maar begin klein'. Het EDWH hoeft niet meteen echt bedrijfsbreed te zijn, zolang de opzet maar zodanig is dat het daar heen kan groeien. Dit is een belangrijk punt, want dat betekent dat die uitbreiding mogelijk moet zijn. Met andere woorden: de opzet moet 'flexibel' zijn. Bovendien heeft de hub-and-spoke architectuur van afbeelding 1 nog wel andere voordelen, ook als hij wat kleinschaliger wordt opgezet. Met name over die voordelen willen wij het hier hebben.

Oorzaken inflexibiliteit

De redenen waardoor een BI-omgeving inflexibel kan zijn kan men grofweg in twee categorieën indelen: 1. onderlinge afhankelijkheid onderdelen; 2. modellering van historie.
De eerste categorie leidt tot inflexibiliteit, doordat voor zelfs relatief kleine wijzigingen een hele keten van technische veranderingen moet worden doorgevoerd. Stel dat een bepaald attribuut in het operationele systeem wijzigt (van datatype bijvoorbeeld). Dan moet de ETL van de sources naar de staging worden aangepast: de staging tabel zelf, alsmede de DWH tabel en ook de betreffende datamart tabel. Verder natuurlijk nog de ETL van staging naar DWH en van DWH naar Datamart. Dit is een hoop werk voor zo'n relatief kleine wijziging. In verband met deze problematiek heeft men het vaak over de noodzaak van het vastleggen van metagegevens. Als dit netjes gebeurt kan men een



Afbeelding 3: Tijdlijnen van beide producten.



Afbeelding 4: Tijdlijnen na uitbreiding DWH.

'impactanalyse' laten doen. Alle tools die betrokken waren bij de implementatie laten een spoor van metadata achter. Er zijn dan ook tools die claimen dit hele woud van metagegevens te kunnen overzien (MetaCenter, SuperGlue). Terzijde zij opgemerkt dat dit mijns inziens ook de verkeerde weg is. In plaats van met dure kijkers naar de troep te kijken die men heeft achtergelaten, kan men beter beginnen met een relatief kleine hoeveelheid metadata (een 'model'). Van daaruit kan men de implementatie zoveel mogelijk genereren.

Sterschema's als starschema's

Gaan we over tot de bespreking van de tweede categorie. Hoe kan het dat vastlegging van historie (en de modellering daarvan in het datawarehouse) tot starheid leidt? Laten we daartoe een simpel voorbeeld bekijken. In afbeelding 2 zien we een tabel waarin historie van enkele attributen ('Courant' en 'Verpakking') van onze producten wordt bijgehouden. Producten zelf worden geïdentificeerd door een 'Productcode'. De versies van deze producten in de loop van de tijd kunnen worden geïdentificeerd door deze productcode, tezamen met de ingangsdatum waarop de versie geldig werd. Met andere woorden: de kolommen 'Productcode' en 'Geldig van' vormen de primaire sleutel van deze tabel. In de praktijk is het vaak een secundaire sleutel, omdat men elke versie ook nog een absoluut nummertje geeft, wat dan de primaire sleutel is. Dit namelijk als het een 'slowly changing' dimensie is in een sterschema. In afbeelding 3 hebben we de tijdlijnen van beide aanwezige producten getekend. Er zijn nog niet zoveel attributen aanwezig, want dit is ons eerste increment en we hebben ons gehouden aan het adagium 'start klein, denk groot'. Dit betekent echter, dat we wel in staat moeten zijn om een datawarehouse, reeds in productie zijnde, uit te breiden, en uiteraard onder behoud van de reeds aanwezig data inclusief historie. Nu zult u zeggen: "Zo moeilijk is dat toch niet, we hebben toch een statement als ALTER TABLE ADD COLUMN"!

Laten we eens kijken hoe dat uitpakt. Stel nu dat we een attribuut 'Productie plaats' willen toevoegen. Verder weten we nog dat A30j tot datum 20 in Alkmaar werd geproduceerd en daarna in Zaandam. B00 ook eerst in Alkmaar, maar vanaf 60 in Purmerend. Dit betekent dat in beide tijdlijnen een extra breukpunt verschijnt (afbeelding 4). We zien dus dat de nieuwe historietabel er uit zal moeten zien als in afbeelding 5. We moeten dus niet alleen een kolom toevoegen, maar ook rommelen met nieuwe rijen. Als dit een langzaam veranderende dimensie van een sterschema is hebben we helemaal een probleem, want we zullen strikt genomen ook nog allerlei updates moeten doen op verwijzingen in de feitentabel naar deze dimensie.

Obstakel

Nu zou u kunnen vinden dat ik wat overdrijf. Meestal is dat soort historie van de nieuwe attributen toch nergens bewaard. Voor historie gebruiken we nu juist het datawarehouse, en daar zat het nog niet in. We voegen gewoon een kolom toe en zetten in

Product Code	Geldig Van	Geldig Tot	Courant	Verpakking	Productie Plaats
A30j	10	20	Ja	Dozen	Alkmaar
A30j	20	40	Ja	Dozen	Zaandam
A30j	40	50	Ja	Kratten	Zaandam
A30j	50	9999	Nee	Kratten	Zaandam
B00	20	60	Ja	Flessen	Alkmaar
B00	60	9999	Ja	Flessen	Purmerend

Afbeelding 5: Aangepaste historietabel.

elke rij die de meest recente versie representeert de huidig bekende waarde van 'Productie plaats' en klaar is Kees. Maar stel dat er een saai product is, waaraan nog nooit iets is veranderd, althans niet de huidige aanwezige attributen. Er is dus maar één rij voor en dat is dan natuurlijk ook de meest recente versie. Laat nu juist voor dit product de 'Productie plaats' in het (recente) verleden zijn gewijzigd. Query's die transacties in het verleden raken zullen dus al heel gauw feitelijk verkeerde informatie geven, namelijk met de huidige 'Productie plaats' in plaats van de juiste. Er zijn nog wel pogingen te doen om de zaak te redden. Men kan bijvoorbeeld verheugd roepen dat nu juist hier een mooie toepassing van het goed bijhouden van metadata ligt. Immers laten wij van elk attribuut vastleggen vanaf welke datum het in het gegevenspakhuis is opgenomen. Dan kan de gebruiker hiermee rekening houden bij zijn query's. Dit gaat natuurlijk niet werken, want gebruikers gaan niet eerst netjes de metagegevens raadplegen, voordat ze een rapport draaien. Met andere woorden; de front-end tool zou dan moeten waarschuwen.
Hoe het ook zij, ik hoop op zijn minst aannemelijk te hebben gemaakt dat opslag van historie en met name sterschema's met slowly changing dimensions een obstakel vormen voor de flexibiliteit. Alhoewel ik een groot voorstander ben van sterschema's moet ik dus toegeven dat het een beetje starschema's zijn. Ik herinner mij nog goed een belangrijk moment in mijn eigen meningsvorming. Ik had mij bovenstaande bedacht maar was de problemen nog nooit in de praktijk tegengekomen, omdat ik in die tijd meestal betrokken was bij nieuwbouwprojecten en daar doet zich deze moeilijkheid per definitie nog niet voor.

Daarom vroeg ik bij gelegenheid eens aan een gerespecteerd collega[1] of hij dit soms al eens tegengekomen was. Met andere woorden, mijn vraag was: "Heb je ooit al eens in de praktijk een draaiend sterschema van structuur moeten veranderen of zelfs maar uitbreiden, uiteraard onder behoud van de reeds bestaande data met historie"? Antwoord: "Ja". Ik: "En was dat dan niet lastig"? Het antwoord was wederom bevestigend. Op mijn vraag hoe hij deze kwestie dan had opgelost, kwam de verrassende uitspraak: "Dan gooien we de tabellen gewoon weg en herontwerpen we het sterschema". Op mijn bezorgde opmerking dat je dan toch alle gegevens inclusief historie kwijt bent, zei hij laco-

niek: "Nee hoor, die zitten nog in het centrale datawarehouse. We hebben alleen de datamart (!) weggegooid en opnieuw ontworpen. Daarna is deze opnieuw initieel geladen vanuit het datawarehouse". Ik probeerde nog zwakjes tegen te werpen dat je dan hetzelfde probleem toch hebt in dat datawarehouse, want dat zijn toch ook sterschema's? Maar ik bleek in de verkeerde veronderstelling te verkeren, dat hij ook tot het Kimball kamp behoorde. Toen begon het mij te dagen. Dat was dus de reden waarom Bill Inmon en anderen de CIF (Common Information Factory) architectuur prediken!

Wegwerpartikel

Hierboven stond dat zo'n herontworpen datamart opnieuw initieel geladen wordt vanuit het datawarehouse. Gaat dat dan niet erg lang duren? Dat hangt natuurlijk af van de omvang van de datamart en met name van het aantal rijen in de feitentabel. De literatuur heeft het vaak over de gigavoorbeelden met vele miljarden rijen in de feitentabel (Walmart, Telecom, banktransacties, enzovoort). Ten eerste blijkt ook daar al veel mogelijk met de huidige hard- en software, maar ten tweede is de grote meerderheid van de gebruikte datamarts lang niet zo groot. Het opnieuw vullen van een feitentabel met tientallen miljoenen rijen, daar kun je vandaag de dag gewoon op wachten. Er is een aardige parallel te trekken met de ontwikkelingen op het gebied van contactlenzen. Enige tijd geleden waren dat dure dingen, waar je zo'n beetje je hele leven mee deed. Tegenwoordig gebruikt men zogenaamde daglenzen, partylenzen enzovoort. Lenzen zijn dus wegwerpartikelen geworden en zo is het ook met datamarts. Ik zie dan ook nu situaties waarbij men zelfs zonder noodzaak van herontwerp er voor kiest om een datamart elke nacht, na het incrementeel bijwerken van het datawarehouse, volledig opnieuw aan te maken en te vullen (te 'rebuilden'). Dit heeft het bijkomende voordeel dat men er meer op kan vertrouwen dat de data in de datamart 'in sync' zullen zijn met het datawarehouse. In de klassieke situatie, waar de datamart incrementeel werd bijgewerkt, wist je dat maar nooit. Daarnaast zien we ook de ontwikkeling van zogenaamde virtuele datamarts. In plaats van een fysieke nieuwe database creëert (of genereert) men views op het datawarehouse. Met name bij de zogenaamde appliances snijdt het mes dan aan twee kanten: voor de performance is het niet meer nodig om een fysiek apart sterschema te gebruiken. De gespecialiseerde hardware en software is, met name door het toepassen van massieve parallelliteit, zo snel dat de query's best op het genormaliseerde datawarehouse kunnen worden afgevuurd. Aan de andere kant is de eenvoud van sterschema's weer aantrekkelijk om de eindgebruikertools op te implementeren.

Doordat deze views op het datawarehouse de structuur van een sterschema hebben, heeft men dus het beste van twee werelden gecombineerd. Verder zien we ook het ontstaan van verschillende soorten datamarts. Grofweg kan men deze indelen langs twee assen: 1. het hierboven genoemde onderscheid tussen fysiek versus virtueel; 2. 'Full Blown' versus beperkt. Met 'Full Blown' bedoelen we dat alle attributen die vanuit de transacties te bereiken zijn, ook zijn opgenomen. Verder zijn er geen selecties, noch aggregaten. Bij een beperkte datamart is er juist wel sprake van een beperkt aantal attributen en kan er sprake zijn van een selectie (de datamart van de verkopen aan Noord Amerika). Als er ook nog sprake is van geaggregeerde gegevens, dan wordt de hoeveelheid data al vaak zo klein dat het met gemak in een Excel-draaitabel past, zodat de eindgebruikers er meteen mee aan de slag kunnen. In dit geval spreken we dan van een datashop. In afbeelding 6 staat het een en ander samengevat.

	Fysiek	Virtueel
Full Blown *Alle bereikbare attributen*	Klassieke sterschema	Appliances: Teradata, Netezza, Exadata, ...
Beperkt *Niet alle attributen, Selecties op waarden, aggregaten*	1 tabel, Draaitabel (Datashop)	Ad hoc view op DWH, ook voor 'gewoon' RDBMS

Afbeelding 6: Datamart soorten.

De oplossing

Het algemene idee is dus als volgt: het EDWH is bedoeld voor de (centrale) opslag van gegevens uit de bronsystemen, inclusief historie. De integratie van data is hier ook geregeld. Het is echter niet de bedoeling om hierop rechtstreeks toepassingen voor eindgebruikers te implementeren. Daarvoor zijn de datamarts bedoeld. En hier kan men (naast andere methoden) kiezen voor sterschemastructuren, opdat die eenmaal makkelijker te bevragen zijn en in het algemeen een betere performance geven. Rest natuurlijk de vraag hoe dit centrale datawarehouse gemodelleerd moet worden als dat geen sterschema's bevat. Het probleem met de boeken van Bill Inmon is dat hij nergens concrete voorbeelden geeft van een dergelijk datawarehouse.

Uitgangspunt is: genormaliseerd met historie. Heel lang heb ik zelf gedacht dat dit betekent dat je zo'n beetje voor elk attribuut een aparte tabel moet aanleggen en dat het dus veel te ingewikkeld wordt. Dat valt echter wel mee. Dan Linstedt heeft precies alle consequenties hiervan op een rijtje gezet en zijn manier van modelleren een naam gegeven: Data Vault ofwel gegevenskluis. In een volgend artikel gaan we hier dieper op in en laten we zien hoe via een eenvoudig stappenplan uit een gegeven informatiemodel (ER-model) een Data Vault model is af te leiden.

Noot

1. Dit was Alexander van Helm, tegenwoordig werkzaam bij Kadenza.

Harm van der Lek

Dr. H. van der Lek (harm.vanderlek@biready.com) is directeur van BIReady.

BPMN 1.2 in de praktijk

met Christian Gijsels

8 oktober 2009
Hotel Lapershoek Hilversum

BPMN staat voor "Business Process Modeling Notation". Deze notatie is uitgegroeid tot dé standaard voor het modelleren van bedrijfsprocessen. Door de toepassing van grafische voorstellingen begrijpen organisaties hun interne bedrijfsprocedures beter en kunnen ze er op een gestandaardiseerde manier over communiceren. Ook maken deze grafische voorstellingen het gemakkelijker om inzicht te krijgen in samenwerkingsverbanden en zakelijke transacties tussen verschillende organisaties. Hierdoor verwerven organisaties beter begrip van hun processen en alle deelnemers daaraan en kunnen ze zich snel aanpassen aan nieuwe interne en B2B situaties (bron: BPMN.org).

Het seminar is een inleiding tot het "Business Process Modeling Notation" formaat. In het programma komt aan de orde:

- positionering van BPMN
- de syntax van BPMN en de relatie met UML
- BPMN case
- BPMN tools & demo

Bestemd voor ú

Dit seminar is bedoeld voor ondernemingen en overheidsorganisaties die een formele manier zoeken om elk aspect van hun bedrijfsprocessen met inbegrip van bedrijfsactiviteiten, organisatie, berichten inclusief hun doorstroom, deelnemers en regels te beschrijven. Ook is het seminar bedoeld voor projectgroepen die op zoek zijn naar een betere manier van communicatie tussen technologie en business.

Array Seminars: specialist in IT-vakkennis!

Kijk snel op www.arrayseminars.nl voor het complete programma!

Array SEMINARS

COMPUTABLE