Reactie op Asymmetrische links in de Data Vault

Een risicovolle uitbreiding op Data Vault

Ronald Kunenborg

Harm van der Lek heeft in twee artikelen (in DB/M 1 en 2) uiteengezet waarom hij van mening is dat asymmetrische links een goede en theoretisch correcte uitbreiding zijn op de Data Vault methodiek. Mijn mening is dat zijn toevoeging in twee delen uiteenvalt, waarvan het asymmetrische gedeelte rechtstreeks strijdig is met de uitgangspunten van de Data Vault architectuur en het gedeelte met historie in de links soms toepasbaar is, maar op termijn vaak zal leiden tot grote problemen in de opslaglaag.

Allereerst lijkt het me verstandig om nog even kort de voornaamste uitgangspunten van de Data Vault methode te herhalen. Ten eerste slaat Data Vault de gegevens historisch op in een model met een paar aparte eigenschappen. Hierbij zijn de sleutels waar alles aan wordt opgehangen onafhankelijk van elkaar opgeslagen in eigen tabellen (hubs). Verder liggen de koppelingen (links) tussen sleuteltabellen buiten de sleuteltabellen. Ook ligt de historie niet vast in de ankers waar het model aan hangt, maar in attribuuttabellen (satellites). Deze constructie zorgt ervoor dat de onvermijdelijke wijzigingen voornamelijk impact hebben op de attribuuttabellen en het model zelf niet aantasten. Verder kun je de attribuuttabellen splitsen of er nieuwe naast zetten, zodat je zelfs die tabellen meestal niet hoeft aan te passen. Ook zorgt de ontkoppeling tussen de tabellen voor een hele grote schaalbaarheid, omdat je heel veel parallel kan laden – de architectuur zelf legt daaraan bijna geen beperking op.
Dit maakt real-time laden ook veel eenvoudiger dan in vrijwel alle andere EDW-architecturen.

Ten tweede heeft Data Vault een agressieve focus op vastleggen van alle data, vanuit het standpunt dat bedrijfssleutels kunnen wijzigen, maar feiten niet. Daarom worden geschonden business rules ook niet afgedwongen tijdens het laden, maar hooguit gerapporteerd. Elke keer dat een laadproces data 'schoont' van vreemde smetten, verliezen we gegevens. Elke keer dat een laadproces voor een EDW stopt om even te vragen of het wel klopt wat er staat, verliezen we tijd.
Beide kosten uiteindelijk geld en in het geval van real-time bijladen kan het zelfs niet eens.

Tenslotte is het uitgangspunt: 'All the facts, all the time!' Als dit uitgangspunt goed is doorgevoerd dan kun je altijd de gegevens in de rapportages herleiden tot de gegevens op elk moment in het bronsysteem. Dit is een krachtig verkoopargument voor een EDW, in het kader van Sarbanes-Oxley, Basel II en andere eisen die tegenwoordig aan de auditing van interne processen worden gesteld.

Problemen met asymmetrische links – unieke sleutels en bedrijfsregels

Vanuit die uitgangspunten zal het nu al wel duidelijk zijn waar mijn problemen met de asymmetrische links vandaan komen. Maar voor de volledigheid licht ik ze toch even toe, te beginnen bij het 'asymmetrische' gedeelte waarbij we een unieke sleutel op de tabel zetten.
Allereerst is er met deze unieke sleutel een bedrijfsregel geïntroduceerd in het gedeelte van het EDW waar we feiten willen laden. Maar wat doe je met feiten die niet voldoen aan de wensdroom van de bedrijfsregels? Data Vault is heel expliciet: laden! En als je ze moet laden, dan moet de unieke sleutel er af. Het alternatief is dat je de gegevens niet gaat laden, maar corrigeert. Hiermee ondergraaf je zoveel kernelementen van de Data Vault methode dat we het maar beter de 'HarmVanDerLek-methode' kunnen noemen, want Data Vault is het niet meer. En nog los van het ondergraven van het hele uitgangspunt dat je de feiten altijd ruw inlaadt, zorgt het er ook voor dat je niet langer real-time kan laden: je zal bij laadfouten op die sleutel handmatig actie moeten ondernemen. En dat kan nooit real-time. Dus deze kleine aanpassing van het model zorgt ervoor dat je jezelf stevig inperkt in de mogelijkheden die de methode te bieden heeft. Zo kun je bijvoorbeeld niet meer naadloos een real-time webservice integreren met het hele laadproces.

Het argument dat je zonder deze unieke sleutels geen structuur in je database hebt is weinig steekhoudend: de hubs, links en satellites bieden juist een hele degelijke, schaalbare en consistente structuur, voorzien van metadata, over de ingeladen gegevens heen. En het argument is ook weinig consistent: waarom wel deze bedrijfsregel, maar geen andere? Omdat deze regel blijkbaar minder wijzigt, of net iets handiger te implementeren valt? Het zal aan mij liggen, maar ik zie de dringende noodzaak niet.

Tenslotte is het heel vreemd dat het EDW, het rapportagesysteem, de bedrijfssleutels voor een bronsysteem moet afdwingen. Blijkbaar was de bedrijfssleutel dus gewijzigd na implementatie van het bronsysteem, of bewaakt het bronsysteem die bedrijfsregel niet: een loeiende alarmsirene wat betreft de stabiliteit van die regel en een hele duidelijke validering van het principe dat je de bedrijfsregels pas in acht neemt bij het opbouwen van de datamarts. De opmerking van Harm "de regel kan bij de business allang zijn gewijzigd zonder dat we dat (op deze manier althans) in het datawarehouse merken. En dat is pas echt eng" spreekt wat mij betreft dan ook boekdelen. Blijkbaar moet de IT-afdeling opmerken dat de gebruikers van een bronsysteem hun eigen bedrijfsregels aanpassen. En er dan ook iets mee doen – op eigen initiatief? De IT-afdeling zal het aardig druk krijgen als je meer dan twee bronsystemen moet ontsluiten op die basis. En dan ga ik maar niet in op de impliciete aanname dat de IT-afdeling verantwoordelijk is voor correcte rapporten, in plaats van ondersteunend en uitvoerend. Maar genoeg over de unieke sleutel. Het andere onderdeel van het concept is de historisering van de link.

Problemen met asymmetrische links – geschiedenis van de relatie

Als we een link met start- en einddatum hebben, dan is het helemaal niet ondenkbaar dat er een keer attributen op de link komen. Met name wanneer je een nieuw bronsysteem ontsluit is die kans aanwezig omdat we nu eenmaal op bedrijfssleutels integreren en daar zijn er minder van dan technische sleutels. De kans op een treffer op een link is dan ook redelijk hoog. In dat geval zit je met een probleem met de historische links. Allereerst moet je een extra satelliet maken voor de nieuwe attributen. Vervolgens moet je de historie van de link migreren naar een satelliet met een geldigheidsindicator. En vervolgens moet je alle downstream datamarts aanpassen.

Het sterke punt van Data Vault is dat de onvermijdelijke wijzigingen in het model zoveel mogelijk worden beperkt tot alleen de plek die het betreft. Maar deze uitbreiding van het model introduceert extra afhankelijkheden die ook elders in het datawarehouse nog flinke impact kunnen hebben.
Natuurlijk mag je stellen dat historie op de links beter te begrijpen is door gebruikers dan historie in de satelliet op de link. Maar de focus van Data Vault heeft nooit op begrijpelijke query's of goede query performance gelegen. De focus ligt op correcte historische opslag met zo eenvoudig mogelijke ETL en zo min mogelijk onderhoud in een snel wijzigende omgeving. Verder is het natuurlijk helemaal geen probleem om views aan te bieden op de Links en Hubs die zorgen voor een correcte weergave van de historische werkelijkheid. Die kun je ook prima automatisch genereren.
Je zou wel kunnen overwegen om historie op de links te zetten in een Raw Source Data Vault, die in feite een historische kopie van de bron opslaat in een Data Vault-structuur. Hier geldt het argument dat je inderdaad niet snel wijzigingen verwacht op zo'n plek. Maar stel dat je ooit wél attributen op de link krijgt? De verleiding om die aan een Hub te knopen is dan opeens wel heel sterk. Op termijn gaat dat wringen en voor je het weet zit je weer met een niet-onderhoudbare opslag waar wijzigingen impact hebben op onverwachte plekken. Precies het scenario wat je met het inzetten van Data Vault wil voorkomen, nog los van de overige bezwaren die er zijn tegen de inzet van Raw Source Data Vaults/Staging Vaults.[1]
Samengevat komen mijn bezwaren tegen asymmetrische links op het volgende neer: je verkrijgt extra beperkingen die de architectuur niet had, in ruil voor niet veel meer dan het verdwijnen van één tabel per link. Zelfs op plekken waar het geen acute problemen oplevert, is het risicovol op lange termijn en in ieder geval niet iets waar je zonder diepgaande kennis van de Data Vault architectuur aan zou moeten beginnen.
Wat mij betreft kunnen we dan ook zonder deze uitbreiding.

Noot

1. http://prudenza.typepad.com/dwh/2011/02/dv-the-case-against-a-raw-data-vault.html

Ronald Kunenborg

Drs. R. Kunenborg is zelfstandig BI-consultant bij Grundsätzlich IT.

Met dank aan Ronald Damhof, Peter de Leeuw van Weenen, en Martijn Evers voor hun inbreng.