VAN DER LANS

Information hiding

Lang, lang geleden bestond er een ontwerptechniek genaamd *Jackson Structured Programming* (JSP). Deze techniek was genoemd naar Michael A. Jackson (niet de zanger). Typerend aan deze techniek was dat de applicatiestructuur werd afgestemd op de structuur van de gegevensopslag, ofwel de structuur van de bestanden en de databases. In feite werden de structuur van het programma en die van de database op elkaar afgestemd. Het is duidelijk dat een dergelijke aanpak tot een goede performance leidt en zorgt dat de applicatie eenvoudig te programmeren is.
Maar al snel werden de nadelen van deze techniek duidelijk. Ten eerste, als de opslagstructuur vanwege performance of opslagaspecten veranderd moest worden, moest ook de applicatiestructuur aangepast worden. Een ander en misschien wel groter nadeel was dat als een nieuwe applicatie dezelfde opslagstructuur moest benaderen, de structuur voor de applicatie niet handig was. Het effect was meestal dat deze tweede applicatie een slechte performance en een lelijke interne structuur had.

In 1972 schreef David L. Parnas een artikel met de titel *On the Criteria to be Used in Decomposing Systems Into Modules*. Hierin introduceerde hij het concept *Information Hiding*. Hij legde uit dat het belangrijk is om de structuur van een applicatie zo onafhankelijk mogelijk van de opslagstructuur te maken. Ergens in de applicatie moet een stuk code zitten dat de opslagstructuur afschermt. Een prachtig concept dat later is overgenomen door de wereld van objectoriëntatie, component based development en SOA.
In feite is dit nu een oerprincipe voor elke software engineer geworden: de applicatiestructuur dient onafhankelijk van de opslagstructuur te zijn. Geen enkele programmeur zal het nut van dit principe nog aanvechten. Maar wel de datawarehouse-specialist.
Als we een klassieke datawarehouse-omgeving bestuderen, zien we dat de BI-applicaties (onze rapporten, KPI's, analytische applicaties) volledig aan de opslagstructuur van het datawarehouse gekoppeld zijn. Bijvoorbeeld, als we om performance redenen een tabel willen denormaliseren, dienen ook de BI-applicaties die gebruik maken van de tabel gewijzigd te worden. Als we aan een dimensie een aggregatieniveau willen toevoegen, moeten we alle BI-applicaties aanpassen, ook degene die niet eens in de nieuwe aggregatie geïnteresseerd zijn. Als we onze databaseserver willen vervangen door een speciale datawarehouse-appliance, moeten heel veel BI-applicaties aangepast worden, omdat de appliance waarschijnlijk een ander SQL-dialect ondersteunt. Als we een datamart willen verwijderen en de BI-applicaties direct het centrale datawarehouse willen laten benaderen, moeten veel BI-applicaties aangepast worden.

In feite zijn veel datawarehouse-omgevingen via een soort JSP-achtige benadering ontwikkeld: koppel de opslagstructuur aan de (BI-)applicatiestructuur. En hier moeten we nu voor boeten. De datawarehouse-omgeving is hierdoor verre van flexibel. Zelfs sommige onschuldige veranderingen leiden tot een enorme inspanning.
Omdat flexibiliteit steeds belangrijker wordt, wordt het tijd om een andere aanpak te kiezen. We moeten het concept van information hiding in de datawarehouse-wereld gaan toepassen. Dit betekent dat er tussen enerzijds onze datawarehouses en datamarts en anderzijds onze BI-applicaties een extra software-laag moet komen die de twee van elkaar gaat scheiden. Als we dan de opslagstructuur veranderen, dan zal dat waarschijnlijk ertoe leiden dat we de tussenlaag moeten aanpassen, maar de BI-applicaties waarvoor de verandering irrelevant is, moeten onveranderd blijven.

Sommige leveranciers zullen melden dat ze al zo'n laag hebben. Bijvoorbeeld, database-leveranciers zullen laten weten dat ze een view-mechanisme bieden waarmee veel veranderingen opgevangen kunnen worden. En leveranciers van BI-tools zullen melden dat ze een universe of iets dergelijks ondersteunen. Maar dit zijn voornamelijk oplossingen die alleen voor die database-server of dat BI-tool gelden.
We willen deze laag tool-onafhankelijk kunnen definiëren. Alleen dan kunnen we makkelijk veranderingen doorvoeren. Kortom, we moeten een tweede fase van het datawarehouse-tijdperk ingaan, één waarbij information hiding centraal staat, zodat we flexibelere datawarehouse-omgevingen kunnen ontwikkelen.

Rick van der Lans is zelfstandig IT-consultant.