VELDWIJK

ESLP: Game, Set and Batch!

Twee columns terug schreef ik over de problemen die ontstaan wanneer een informatiesysteem niet kan worden gebaseerd op één programmeerparadigma. In die column beschreef ik mijn rol bij een ontwikkelproject als vertegenwoordiger van de relationele visie op applicatie-ontwikkeling. Het systeem dat mijn collega's en ik ontwikkelden was gebaseerd op Java/Oracle en het was onze bedoeling om het maximale uit beide omgevingen te halen: Java als basis voor schermen en workflow en Oracle rdbms als basis voor gegevensopslag (natuurlijk) en batchverwerking. Over die batchverwerking wil ik het hier hebben.

Mijn stelling hier is dat relationele technologie in zijn meest pure vorm, dus relationele technologie die is ontdaan van alle toeters en bellen, in de praktijk de optimale basis vormt voor zware, complexe, maar verder recht-toe-recht-aan, batchverwerking. Met de meest pure relationele technologie doel ik op SQL zonder enige 3GL-achtige toevoeging; in een Oracle-omgeving met meestal PL/SQL, maar vroeger ook vaak talen als Cobol en C, waarin SQL statements worden ingebed.
Met zware, complexe, maar recht-toe-recht-aan, batchverwerking doel ik op batchverwerking die wordt gekenmerkt door veel relatief eenvoudige operaties als aggregaties, selecties en joins die een zwaar beslag op de rdbms *resources* doen.
En omdat elk ICT concept een naam moet hebben, noem ik de aanpak die ik hier beschrijf *Extreme Set Level Programming*, ofwel ESLP.
Hoe werkt ESLP? Stel dat je een overzicht wilt maken van de afdelingen in een bedrijf en per afdeling gegevens over de aan die afdelingen verbonden medewerkers wilt afdrukken. De normale aanpak zou daaruit bestaan dat je met PL/SQL, een objectgeoriënteerde taal als Java, of een klassieke 3GL, een cursor opent op de tabel met afdelingen. Voor elke gelezen afdeling wordt vervolgens een cursor geopend op de tabel met medewerkers. Het rdbms levert eenmalig een verzameling (set) met afdelingrecords op en voor elk afdelingrecord een set met medewerker records. Heel simpel allemaal. En 40-plussers die nog gestructureerd Cobol hebben leren programmeren weten ook dat het best mooi is wanneer de structuur van je programma identiek is aan de structuur van je batch output.
In de ESLP-benadering zien we zoals gezegd af van PL/SQL, Java of een 3GL en doen we alles met SQL. We schrijven één SQL statement dat de afdelingsgegevens ophaalt en één SQL statement dat de medewerkergegevens ophaalt. De resultaten stoppen we in een output-tabel, bij voorkeur een met een hiërarchische structuur zodat het programma dat de daadwerkelijke output verzorgt zo dom mogelijk kan blijven.
Misschien levert deze aanpak bij u het gevoel op dat er met een kanon op een mug wordt geschoten, maar dan moet u bedenken dat ik hier een zeer eenvoudig voorbeeld geef.
Het systeem dat wij bouwden moet uiteindelijk in staat zijn om dagelijks tot 100.000 brieven, accepten en rapportages uit te draaien, waarbij het in totaal ook nog om enkele honderden *soorten* output gaat. Dat bedoelde ik hiervoor met zware, complexe, maar verder recht-toe-recht-aan batchverwerking.
En voor het genereren van journaalposten in het grootboeksubsysteem gold iets vergelijkbaars. Bij de klassieke programmeeraanpak zoals hierboven beschreven loop je in dit soort gevallen snel tegen performance-problemen aan.
De ESLP-aanpak is hier altijd superieur en wel om vier redenen. Allereerst haalt ESLP het onderste uit het rdbms: met een minimaal aantal SQL statements wordt een maximaal aantal gegevens opgehaald: vandaar ook 'Extreme Set Level Programming'. Een tweede, minstens even belangrijk argument is dat er nauwelijkse sprake is van *context switching* (ge-ping-pong tussen rdbms en 3GL) en dat er al helemaal geen problemen zijn met alle overhead die objectgeoriënteerde talen als Java met zich meebrengen. Ten derde gebeuren alle interessante zaken op de database server en is het netwerkverkeer dus meestal minimaal.
En last but not least maakt de ESLP-stijl van programmeren het heel eenvoudig om performance bottlenecks op te sporen en weg te nemen. Log de uitvoeringstijd per SQL statement, gebruik *Explain Plan* en schaaf aan het statement of aan de database-inrichting. Bijna altijd kost dit veel minder inspanning dan een complex programma analyseren en herschrijven.
ESLP komt kort en goed neer op het in bulk en ineens ophalen van alle gelijksoortige gegevens in plaats van het ophalen van gegevens, conform de structuur van de aan te maken output. Anders gezegd: ESLP is database-gericht in plaats van output-gericht.
De kritische lezer van deze column zou er voor kunnen kiezen om de filosofie van ESLP toe te passen binnen de gangbare *Embedded SQL-benadering*. Wie dat doet zal merken dat er dan weinig niet-SQL hoeft te worden geprogrammeerd. De reden waarom ik in overweging geef om je te beperken tot de extreme benadering van *puur* SQL is dat alleen dit garandeert dat de programmeur afziet van alle slimmigheden die 3GL's en talen als Java bieden. Ook de 'performance' van de programmeur is vaak beter wanneer de toolbox minder rijk gevuld is. Minder is hier sneller én meer!

René Veldwijk
Dr. R.J. Veldwijk (rene.veldwijk@faapartners.com) is partner bij FAA Partners, een onderdeel van de Ockham Groep.