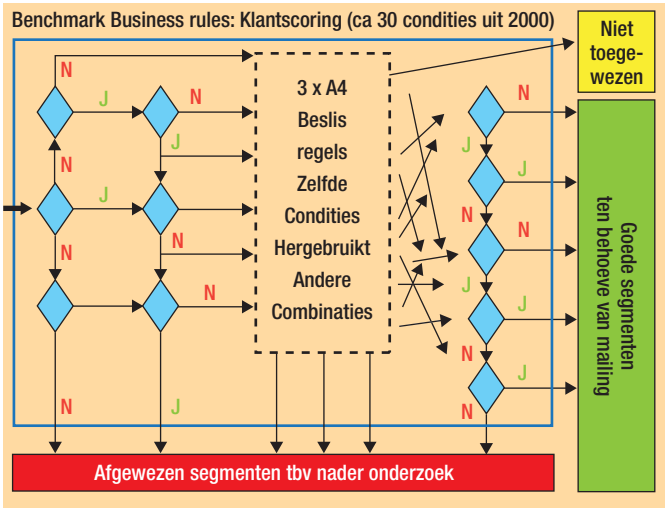
Haal meer uit de systemen die je in huis hebt

De vergeten kracht van het DBMS

Huub Hillege

Het gemak dat het gebruik van Extract, Transform, Load (ETL) tools met zich meebrengt heeft ervoor gezorgd dat het onderliggende Database Management Systeem tegenwoordig vaak wordt vergeten. Dat is jammer, want bedrijven kunnen veel meer uit het DBMS halen dan men zich realiseert, met meer efficiëntie, inzichtelijkheid en zelfs betrouwbaarder bedrijfsresultaten tot gevolg.

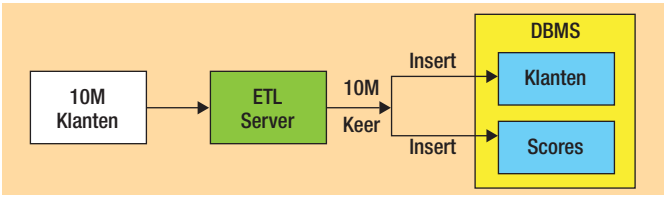
Ook wordt te snel met het bouwen van ETL-tools begonnen, vooral bij realisatie van complexere processen blijkt dat zo'n tool het DBMS niet altijd correct aanstuurt. Problemen komen pas later boven tafel, als blijkt dat de ETL-processen niet in het beschikbare nachtslot passen of als uit definitieve rapportages naar voren komt dat de data niet volledig of zelfs incorrect zijn. Vanwege de complexe structuur is het haast onmogelijk de fout te vinden en zit er weinig anders op dan de ETL-processen te herzien. Opnieuw beginnen dus, maar het kan anders!
De twee belangrijkste problemen bij het gebruik van ETL-tools voor het verwerken van complexe processen zijn de beschikbare tijd en de inzichtelijkheid van de resultaten. De tijd die nodig is om de dagelijkse *delta* aan wijzigingen toe te voegen aan het aanwezige datawarehouse kan onvoldoende zijn. Zo kan één ETL-proces al langer duren dan de totale beschikbare tijd buiten kantooruren. Het tweede probleem is de onleesbaarheid van de processen, waardoor de resultaten niet controleerbaar zijn.
De mogelijkheden van ETL- en BI-tools zijn de laatste jaren dusdanig uitgebreid dat bedrijven voorbij gaan aan de aanwezigheid van het onderliggende DBMS en bijbehorende functies. Met andere woorden: bedrijven hebben vaak meer in huis dan ze denken.
Het DBMS wordt meer en meer het ondergeschoven kindje. Een voorbeeld uit de praktijk wijst uit dat verkeerde ETL-keuzes, gemaakt in samenhang met het DBMS, tot ernstige vertragingen kunnen leiden en zelfs incorrecte resultaten. Een grote internationale bank wil een keuze maken uit een selectie van verschillende hardware/DBMS leveranciercombinaties. Om de juiste optie te kiezen worden teams samengesteld om de beschikbare mogelijkheden te testen. Eén van de belangrijkste beslispunten is de vraag hoe bestaande scoringssegmentatie-algoritmes in een datawarehouse-omgeving kunnen worden gebracht en onderhouden. De test moet aantonen dat een complexe klantscoring voor 10 miljoen klanten in de nachtverwerking zou kunnen plaatsvinden, zodat dagelijks specifieke scores kunnen worden opgebouwd voor trend-analyses, fraude-aspecten en mailingcampagnes. Vanuit deze vraag geven we hieronder vier mogelijke oplossingen weer, waarbij veelvoorkomende problemen met het gebruik van ETL-tools naar voren komen.



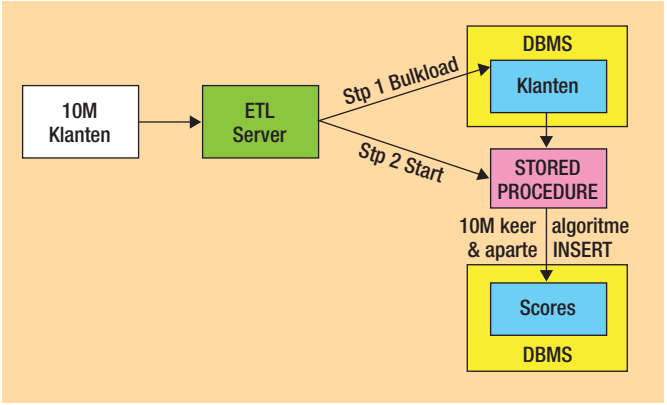
Afbeelding 1: Logisch beslissingsalgoritme.

Testcase klantscoring-algoritme

Uit de totale klantgegevens, circa tweeduizend attributen en scores, werd aan de testteams een 'logisch beslissingsalgoritme' gegeven met een keuze van 32 condities en 13 acties: een samenstelling van IF-THEN-ELSE, zie afbeelding 1. Volgens



Afbeelding 2: Eerste mogelijkheid. RUN TIJD 15:01:29 (hh:mm:ss). Kenmerken: naast de data van elk rijtje gaat ook twee keer een INSERT INTO statement door het netwerk; voor elk rijtje moet het DBMS een transactie backout-mechanisme activeren voor het geval er fouten optreden; de transactie overhead geldt zowel voor het aanbrengen van de klanten als de scores.



Afbeelding 3: Tweede mogelijkheid. RUN TIJD 04:18:48 (hh:mm:ss). Kenmerken: data worden door de bulkload-faciliteit van het DBMS sneller binnengehaald; er worden geen data uit het DBMS gehaald voor controle van het ETL-proces; voor elke rij wordt de SP logica opnieuw doorlopen. Na bepaling van de score wordt een INSERT statement gemaakt; voor elke rij INSERT wordt het transaction backout-mechanisme geactiveerd; om te weten hoever de SP onderweg is, moet een teller worden bijgehouden. Ook herstart moet je zelf programmeren.

dit algoritme bevindt een klantscore zich in een goed 'groen segment', in een afgewezen 'rood segment', of in het 'niet toegewezen' segment. Het algoritme is afgeleid van een lijst met geschreven business rules, zoals:

- alle mannelijke klanten tussen 21 en 35 jaar die een hypotheek en geen eigen creditcard van de bank hebben;
- mocht de partner wel een creditcard van de bank hebben en deze partner is ouder dan 40 jaar en er is geen en/of-rekening, dan moet de mannelijke klant alsnog worden geselecteerd.

Bij de verschillende mogelijkheden om de gegevens via dit algoritme in het DBMS te laden, komt steeds het aspect van de benodigde tijd en de bewijsbaarheid van de resultaten naar voren. We beschrijven alle vier de mogelijkheden om aan te geven hoe het in de praktijk kan werken, zonder dat men bij eventuele verbetering van efficiëntie of foutkans stil staat. Processen kunnen in de loop der jaren complexer en daardoor minder inzichtelijk worden. Dan is de bewijsbaarheid van de resultaten van groot belang. Ook is het in een dergelijk geval wenselijk het maximale uit de beschikbare systemen te halen, zodat het laden van gegevens via dit algoritme in het DBMS zo min mogelijk tijd in beslag neemt.

De eerste mogelijkheid is het algoritme direct in de ETL-tool op de ETL-server in te brengen (DBMS record-at-a-time). De ETL-server krijgt bestanden en/of leest tabellen uit allerlei bronsystemen, waarna ETL-processen de inkomende records/rijen individueel inspecteren, controleren, schonen en uiteindelijk apart doorzenden naar het DBMS. Zie afbeelding 2. In de praktijk wordt dit principe het meest toegepast, terwijl het erg veel tijd in beslag neemt en niet direct is te bewijzen dat het gebouwde algoritme ook daadwerkelijk het juiste resultaat oplevert. Een tweede mogelijkheid is in een bulk-load operatie de klantgegevens in een tabel binnen het DBMS laden. Het algoritme wordt als Stored Procedure (SP) in het DBMS opgeslagen, zie afbeelding 3.

Ook de SP is een complex, genest IF-THEN-ELSE in de geest van:

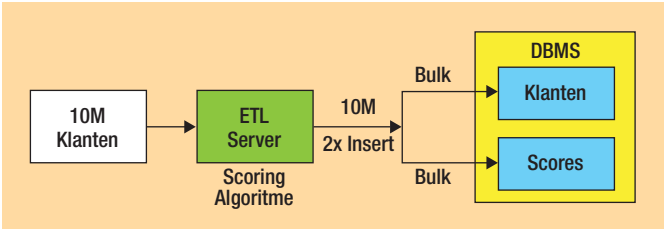
```
BEGIN
Declare  @ClientID =..., @Vector = ...,@A01 =
                                  ... etc... ,@A32 = ...
SELECT
@A01 = A01
,.....etc.....
,@A32 = A32
FROM Klantscore_10K
WHERE KLANT = @ClientID

SET @Vector = 'V'
IF @A01 = 1
BEGIN
--Vector 2, 3, 4, 5, 6, 7, 8, 9, 14--
IF @A02 = 2
BEGIN
IF @A03 <> 3 AND @A04 <> 4 SET @Vector = 'V6'
IF @A05 <> 5
.....etc.....

/* voor elke rij apart */
INSERT INTO KlantVector (KlantID, Score) VALUES
                                  (@ClientID, @Vector)
END
```

Hoewel het goed is dat het DBMS bij deze oplossing aan het werk wordt gezet, zijn ook bij deze mogelijkheid kanttekeningen te plaatsen. Een reden waarom bedrijven voor deze oplossing kiezen is dat een SP zich in het binnenste van het DBMS bevindt en daarom door iedereen met toegang tot het DBMS kan worden geactiveerd. Het grote nadeel blijft wederom dat ook hier niet valt te bewijzen dat het gebouwde algoritme daadwerkelijk het juiste resultaat heeft opgeleverd.

Een derde mogelijkheid, een combinatie van de twee bovenstaande oplossingen, komen we ook vaak tegen in het bedrijfsleven. Het scoringsalgoritme wordt op de ETL-server in een proces ingebracht. De scoring wordt lokaal per rij uitgevoerd en het resultaat



Afbeelding 4: Derde mogelijkheid. RUN TIJD 00:27:30 (hh:mm:ss). Kenmerken: scoring lokaal in ETL; voor elke tabel slechts één INSERT INTO; transactie backout-mechanisme voor totaal aantal rijen per tabel; herstart alleen vanaf het begin.

Vector	V1	V6	V10	V3	V8	V14	V2	V7	Geen score
Bouwer1	7724	94	16	10	115	1616	83	45	297
SQL Controle	7724	248	16	20	3	1616	83	3	287
Bouwer2	5710	307	15	79		1244	78	2	2565

Afbeelding 5.

Conditie		Vector1	Vector2	Vector3	Vector4	Vector5
		Status	Status	Status	Status	Status
C1	Geslacht = 'M'	JA	NEEN	JA	NEEN	NEEN
C2	Geslacht = 'V'	NEEN	JA	NEEN	NEEN	NEEN
C3	Geslacht = 'O'	NEEN	NEEN	NEEN	JA	JA
C4	Leeftijd > 65	NEEN	NEEN	JA	JA	NEEN
Actie:						
A1	Uitnodiging	X	X			
A2	Stropdas			X		
A3	Bellen				X	X

Afbeelding 6.

lokaal op de ETL-server gebufferd. Uiteindelijk worden de klanten en bijbehorende scores via een bulkload-operatie met een enkele INSERT opdracht per tabel vanuit een ETL-proces geladen, zie afbeelding 4. Ook deze mogelijkheid heeft voor- en nadelen. Een voordeel is dat de ETL-server voor elke doeltabel slechts één INSERT naar het DBMS hoeft te sturen. Dat scheelt een aanzienlijke hoeveelheid tijd. Het nadeel is dat niet direct te bewijzen is dat het gebouwde algoritme de juiste resultaten gaat opleveren. Behalve de gebrekkige controleerbaarheid van de eerste drie genoemde mogelijkheden is er nog een ander belangrijk nadeel: de foutkans. We illustreren dit met een voorbeeld uit de testcase. Het omzetten van het algoritme naar een IF-THEN-ELSE structuur in een ETL-tool of SP in het DBMS kost veel tijd. In de testcase is er bij elke implementatie een andere uitkomst. Ook de bouwer met de meeste ervaring (bouwer één) bleek fouten te maken, bouwer twee had zelfs geen enkele uitkomst goed, zie afbeelding 5. Daarbij geven de bouwers aan bij wijzigingen genoodzaakt te zijn opnieuw te beginnen, aangezien het te veel tijd kost het oorspronkelijke werk weer te begrijpen.

Bewezen resultaat

Er is nog een vierde mogelijkheid. Eén waarbij de resultaten wel bewijsbaar zijn en de foutkans aanzienlijk kleiner. Bij de eerste twee opties hebben we gezien dat het DBMS iedere keer slechts één rij uit het externe bestand gebruikt. Per rij kan maar een beperkte optimalisatie worden doorgevoerd, omdat het DBMS geen kennis heeft over de andere rijen. Deze zijn immers nog niet aangeboden. Om het DBMS toch een verzameling gegevens te laten verwerken, en daar dan slimme optimalisaties mee te kunnen doen, moeten we het algoritme verder ontleden. Hiertoe zetten we de gegevens van het algoritme om naar een beslissingstabel representatie in een Excel spreadsheet. Op deze manier kunnen we van elkaar onafhankelijke taken voor het DBMS ontdekken. Een voorbeeld:

Klant#	Klantnummer
Geslacht	M, V of O
Datum_Geboorte	Nodig voor leeftijdsbepaling
Datum_Vandaag	Nodig voor leeftijdsbepaling

Er zijn vier condities:

C1. Is de Klant een Man (Geslacht = M);

C2. Is de Klant een Vrouw (Geslacht = V);

C3. Is het Geslacht onbekend (Geslacht = O);

C4. Is de Leeftijd van de klant > 65
(Datum_Vandaag - Datum_Geboorte > 65 jaar).

We hebben drie acties die alleen afhankelijk zijn van de voornoemde condities:

A1: Stuur een uitnodiging aan alle mannen en vrouwen van 65 jaar en jonger;

A2: Alle mannen boven de 65 jaar krijgen een stropdas;

A3: Alle personen waarvan het geslacht onbekend is, worden gebeld.

Een statuskolom van de beslissingstabel (vector) geeft voor elke conditie aan of deze waar (JA) of niet waar (NEEN) is en welke actie moet worden uitgevoerd, zie afbeelding 6. Op basis van gelijke acties kunnen verschillende vectoren worden samengevoegd, omdat bepaalde condities zowel JA als NEEN kunnen zijn om dezelfde actie als resultaat te krijgen. In dit geval vullen we NVT (niet van toepassing) in. Afbeelding 6 wordt dan een stuk eenvoudiger, zie afbeelding 7. Als we deze aanpak toepassen op de testcase ziet het vereenvoudigde resultaat, waarbij het maximaal aantal vectoren is samengevoegd, er uit als in afbeelding 8. Uiteindelijk blijven 14 vectoren over. Voor elke vector kunnen we de score bepalen en tevens bewijzen dat het resultaat goed is. Ter illustratie bepalen we de score van vector 5, de meest complexe:

Conditie		Vector12	Vector3	Vector45
		Status	Status	Status
C1	Geslacht = 'M'	N.V.T.	JA	NEEN
C2	Geslacht = 'V'	N.V.T.	NEEN	NEEN
C3	Geslacht = 'O'	NEEN	NEEN	JA
C4	Leeftijd > 65	NEEN	JA	N.V.T.
Actie:				
A1	Uitnodiging	X		
A2	Stropdas		X	
A3	Bellen			X

Afbeelding 7.

```
INSERT INTO 10M_Score
SELECT
Klant
,'V5-G5'
from "Test-Klant"
where A01 =   1
and   A02 =   2
and   A03 =   3
and   A04 =   4
and   A05 <>  5
and   A06 <>  6
and   A07 <>  7
and   A08 =   8
and   A09 =   9
and   A10 =  10
and   A11 =  11
and   A12 =  12
and   A13 =  13
and   A14 =  14
and   A15 =  15
and   A16 =  16
and   A17 =  17
and   A20 <> 20
and   A31 =  31
and   A32 =  32
```

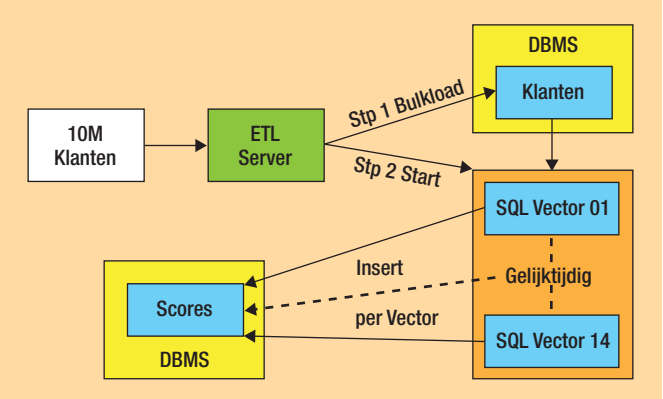
Dit is een zeer eenvoudig statement dat zelfs door niet SQL-kenners te begrijpen valt en ook na lange tijd indien nodig zonder problemen is te wijzigen. De uitvoering is volgens het volgende principe gedaan:

VectorNummer	Aantal Rijen	RunTijd
V01	9690135	2:55
V10	16527	2:25
V11	44	2:36
V12	34	2:33

Conditie	Conditie Nummer	Vector 5	Vector 4	Vector 9	Vector 8	Vector 7	Vector 6	Vector 2	Vector 3	Vector 14	Vector 13	Vector 12	Vector 11	Vector 10	Vector 1	Conditie Nummer	Conditie
A01 = 1	C01	JA	JA	JA	JA	JA	JA	JA	JA	JA	NEEN	NEEN	NEEN	NEEN	NEEN	C01	A01 = 1
A02 = 2	C02	JA	JA	JA	JA	JA	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C02	A02 = 2
A03 = 3	C03	JA	JA	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C03	A03 = 3
A04 = 4	C04	JA	JA	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C04	A04 = 4
A05 = 5	C05	NEEN	NEEN	NEEN	NEEN	NEEN	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C05	A05 = 5
A06 = 6	C06	NEEN	NEEN	NEEN	NEEN	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C06	A06 = 6
A07 = 7	C07	NEEN	NEEN	NEEN	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C07	A07 = 7
A08 = 8	C08	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C08	A08 = 8
A09 = 9	C09	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C09	A09 = 9
A10 = 10	C10	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C10	A10 = 10
A11 = 11	C11	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C11	A11 = 11
A12 = 12	C12	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C12	A12 = 12
A13 = 13	C13	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C13	A13 = 13
A14 = 14	C14	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C14	A14 = 14
A15 = 15	C15	JA	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C15	A15 = 15
A16 = 16	C16	JA	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C16	A16 = 16
A17 = 17	C17	JA	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	JA	JA	JA	N.V.T.	C17	A17 = 17
A18 = 18	C18	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	JA	N.V.T.	NEEN	N.V.T.	C18	A18 = 18
A19 = 19	C19	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	JA	N.V.T.	NEEN	N.V.T.	C19	A19 = 19
A20 = 20	C20	NEEN	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	NEEN	NEEN	NEEN	N.V.T.	N.V.T.	C20	A20 = 20
A21 = 21	C21	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	NEEN	NEEN	JA	N.V.T.	N.V.T.	C21	A21 = 21
A22 = 22	C22	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	NEEN	NEEN	JA	N.V.T.	N.V.T.	C22	A22 = 22
A23 = 23	C23	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C23	A23 = 23
A24 = 24	C24	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C24	A24 = 24
A25 = 25	C25	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C25	A25 = 25
A26 = 26	C26	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C26	A26 = 26
A27 = 27	C27	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C27	A27 = 27
A28 = 28	C28	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C28	A28 = 28
A29 = 29	C29	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C29	A29 = 29
A30 = 30	C30	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	NEEN	N.V.T.	N.V.T.	N.V.T.	C30	A30 = 30
A31 = 31	C31	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	NEEN	N.V.T.	N.V.T.	N.V.T.	N.V.T.	C31	A31 = 31
A32 = 32	C32	JA	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	N.V.T.	JA	JA	JA	JA	NEEN	C32	A32 = 32

Score Naam		Vector 5	Vector 4	Vector 9	Vector 8	Vector 7	Vector 6	Vector 2	Vector 3	Vector 14	Vector 13	Vector 12	Vector 11	Vector 10	Vector 1		Score Naam
G1															XXX		G1
G2								XXX									G2
G3									XXX								G3
G4			XXX														G4
G5		XXX															G5
F1							XX										F1
F2						XX											F2
F3					XX												F3
F4				XX-A							XX-B						F4
F5														XX			F5
F6													XX				F6
F7												XX					F7
F8										XX							F8

Afbeelding 8.



Afbeelding 9.

V13	0	2:32
V02	0	0:25
V03	0	0:15
V04	0	0:35
V05	0	0:16
V06	0	0:24
V07	0	0:23
V08	0	0:14
V09	0	0:07
V14	0	0:21

De totale runtijd is minder dan drie minuten als we alle 14 query's tegelijk laten starten, zie afbeelding 9. Door alle vectoren tegelijk aan te bieden, krijgt het DBMS maximale informatie en kunnen optimalisaties plaatsvinden, zoals hergebruik van buffers en disk cache gebruik. Het spreadsheet met de vectoren en de daaruit afgeleide SQL zijn zo eenvoudig dat ook na een zeer lange periode te begrijpen is wat er gebeurt en waar mogelijke wijzigingen moeten worden aangebracht. Zelfs met honderden variabelen blijven de beslissingstabel en separate vector-definities overzichtelijk.

Meer in huis

Het bedrijfsleven besteedt jaarlijks 10 miljard euro aan ICT, maar ziet onvoldoende rendement waardoor ICT vaak als kostenpost wordt gezien. Er is behoefte aan tastbaarheid, meetbaarheid en geloofwaardigheid. Door kleinere stappen met beter inzicht te maken en efficiënter te werk te gaan, kunnen aanzienlijke kostenbesparingen worden bereikt. Bijvoorbeeld door eerst de mogelijkheden van wat er in huis is optimaal te benutten. Bij het gebruik van ETL-tools betekent dit eerst een gedegen informatie-analyse maken. Zo kan de optimale strategie tussen ETL-tools en DBMS bepaald worden en kan het DBMS in zijn ultieme kracht worden gebruikt. Als deze afweging wordt gemaakt kan een investering soms onnodig blijken. Bezint eer ge met ETL bouwen begint, dus!

Huub Hillege is manager Business Development BI bij Pecoma Business Technology. Met dank aan Herman Berger.