Total Recall = geschiedenis van de historie = bi-temporele tabellen

Historie in het platte vlak

Maarten Zaanen

**Historie in applicaties is een onderwerp dat al vaak is beschreven. Toch blijkt er veel onbegrip te zijn en worden er bij nieuwe systemen vaak nog suboptimale ontwerpkeuzes gemaakt. Met behulp van grafische presentatie is het basisprincipe eenvoudig te begrijpen.
Met de plaatjes in het achterhoofd wordt het lezen van verdiepende literatuur (waar naar wordt verwezen) hopelijk eenvoudiger.**

Veldwijk definieert in zijn artikel 'De Polis Papers (1)' (DBM februari 2009) een Total Recall database als zijnde een database waarin elke toestand is te reconstrueren. Van der Lek noemt ditzelfde principe 'Geschiedenis van de historie' (DBM november 2001). Snodgrass bespreekt in zijn boek 'Developing Time-Oriented Database Applications in SQL' hetzelfde fenomeen uitvoerig en noemt het databases met 'bi-temporele' tabellen.
Als we het woord bi-temporeel ontleden, dan staat 'bi' voor twee en 'temporeel' voor tijd. Kortom, tabellen met twee tijddimensies. De tijddimensies waar het hierbij om gaat zijn 'valid time' en 'transaction time'.

Valid time en transaction time

Allereerst zullen we de termen 'valid time' en 'transaction time' toelichten.
Valid time is de periode[1] waarin iets geldig is. Een voorbeeld van valid time is 'Een hypotheek is voor 30 jaar afgesloten met startdatum 1 jan 2009 en einddatum 31 dec 2038'.
Transaction time[2] is het tijdstip waarop een observatie in de database is geplaatst. Vrijwel zonder uitzondering is dit in transactionele systemen de systeemklok. Een voorbeeld van transaction time is 'Bovengenoemde hypotheek is op 15 dec 2008 15.32 uur in het systeem ingevoerd'. Als de systeemklok wordt gebruikt voor nieuwe transacties dan ligt iedere nieuwe transactie ná alle vorige transactietijdstippen en liggen ze nooit in de toekomst.[3]
Er bestaat over het algemeen een sterke correlatie tussen valid time en transaction time. Echter, principieel zijn ze onafhankelijk van elkaar. In het bovenstaande voorbeeld: de datum waarop een hypotheek in een systeem wordt opgeslagen kan maanden voor de ingangsdatum van de hypotheek zelf liggen, maar wellicht ook slechts een paar dagen ervoor.

Zoals we hierboven hebben gedefinieerd, is 'valid time' een periode en 'transaction time' een tijdstip. Echter, de 'transaction time' kan ook gezien worden als het begin van een periode. Deze periode wordt dan afgesloten door de transaction time van de volgende transactie. Zolang er nog geen volgende transaction time is, zien we die afsluiting virtueel als zijnde in de oneindige toekomst.

Een uitgewerkt voorbeeld

Een voorbeeld spreekt het best tot de verbeelding. Hiertoe nemen we hetzelfde voorbeeld als Snodgrass en Veldwijk ook gebruikten: hoe registreert men het eigendom van vastgoed zodat de geschiedenis van de historie bewaard blijft.
We bestuderen de (niet genormaliseerde) tabel 'vastgoed_eigenaar'. In deze tabel komen de volgende velden voor:

- Vastgoed, beschrijving van stuk vastgoed;
- Eigenaar, voornaam van de eigenaar;
- VT_start, valid time start datum (of datumtijd, afhankelijk van de situatie);
- VT_end, valid time eind datum (of datumtijd, afhankelijk van de situatie);
- TT_start, de transactie datumtijd waarop de row in de tabel geplaatst is;[4]
- TT_end, de volgende transactie datumtijd; als er nog geen volgende transactie is, een 'high date'.

Transactie nr. 1: Op 5 jan 2009 wordt alvast in het systeem ingebracht dat de woning op 10 jan 2009 wordt opgeleverd aan de eerste eigenaar, Richard. Omdat we nu nog niet weten wanneer hij het gaat verkopen worden de VT_end en TT_end op 'high_date' gezet.[5] Zie tabel 1.

Vastgoed	Eigenaar	VT_start	VT_end	TT_start	TT_end
FlatA1	Richard	10 jan 2009	High_date	5 jan 2009	High_date

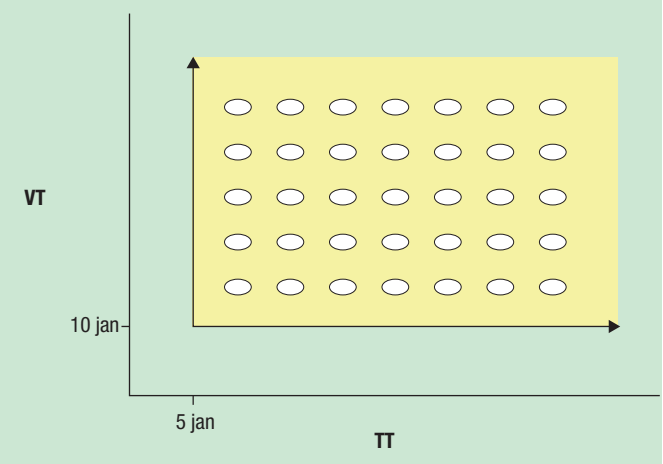
Tabel 1: De inhoud van tabel 'vastgoed_eigenaar' na de eerste transactie voor FlatA1.

Vastgoed	Eigenaar	VT_start	VT_end	TT_start	TT_end	Row
FlatA1	Richard	10 jan 2009	High_date	5 jan 2009	15 jan 2009	1
FlatA1	Richard	10 jan 2009	15 jan 2009	15 jan 2009	High_date	2
FlatA1	René	15 jan 2009	High_date	15 jan 2009	High_date	3

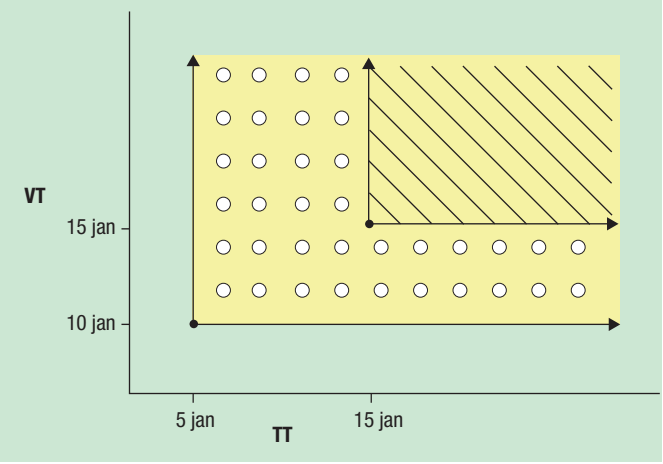
Tabel 2: De tabel 'vastgoed_eigenaar' na de tweede transactie. Het veld 'row' is hier slechts toegevoegd om de relatie met afbeelding 3 te vereenvoudigen.

Grafisch is de situatie voor FlatA1 weer te geven als in afbeelding 1. Het begin van een pijl is hierbij 'VT_start' respectievelijk 'TT_start', het einde van een pijl is vervolgens de 'VT_end' respectievelijk 'TT_end'. Het gebied met de bolletjes is het tijdvak waarin Richard eigenaar is.
Nu zijn vragen als hieronder grafisch inzichtelijk te maken:

- "Was de flat al bekend binnen het systeem op 2 jan 2009?" Antwoord: Op TT = 2 jan 2009, trek een verticale lijn, die lijn blijft in het witte gebied, dus de flat was nog niet bekend in het systeem.
- "Met de kennis zoals in het systeem is opgeslagen op 15 jan 2009, wie was de eigenaar op 11 jan 2009?" Antwoord: Het punt TT = 15 jan 2009 en VT=11 jan 2009 ligt in het gebied met de bolletjes, kortom, Richard was de eigenaar.



Afbeelding 1: Grafische weergave na de eerste transactie FlatA1.



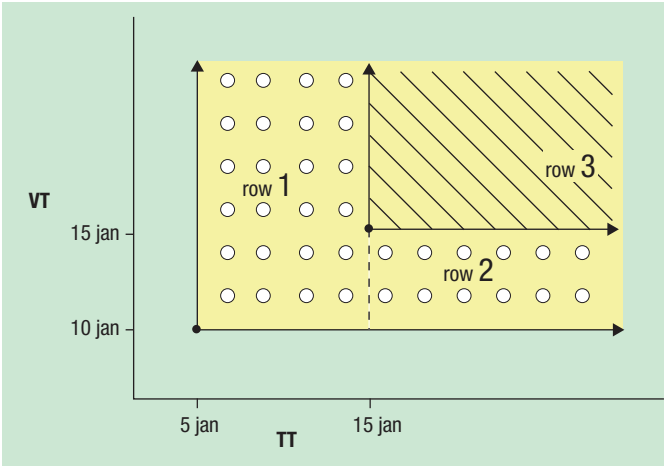
Afbeelding 2: Grafische weergave na de tweede transactie FlatA1.

Transactie nr. 2: Op 15 jan 2009 wordt in het systeem ingebracht dat FlatA1 op 15 jan 2009 wordt verkocht aan de tweede eigenaar, René. Grafisch kan men dat weergeven als in afbeelding 2. Het gebied met bolletjes geeft de historie van Richard weer en het gearceerde vlak die van René. In tabel 2 zien we de 'vastgoed_eigenaar' tabel na transactie nr. 2. Het oorspronkelijke bolletjes gebied van Richard (één row voor de verkoop) is opgesplitst in twee gebieden (twee rows), zie afbeelding 3.

De volgende vraag kan wederom grafisch inzichtelijk gemaakt en beantwoord worden:

- "Met de kennis zoals opgeslagen in het systeem op 20 jan 2009, wie was de eigenaar op 17 jan 2009?" Antwoord: Zie afbeelding 4. We bepalen het kruispunt van TT=20 jan 2009 met VT=17 jan 2009 en zien dat het antwoord René is.

We brengen nog een transactie in het systeem, transactie nr. 3. Het blijkt dat er tussen Richard en René nog een andere eigenaar was: Harm. Op 25 jan 2009 wordt in het systeem ingevoerd dat de woning op 13 jan 2009 is verkocht aan Harm. Grafisch ziet dit er uit als in afbeelding 5.
Na transactie nr. 3, zie tabel 3, begint het al aardig ingewikkeld te worden om te programmeren.



Afbeelding 3: Grafische weergave na de tweede transactie. Het gebied waarin Richard eigenaar was is, database-technisch gezien, gesplitst in twee rechthoekige gebieden. Iedere rechthoek is een row in de tabel.

Vastgoed	Eigenaar	VT_start	VT_end	TT_start	TT_end	Row nr.
FlatA1	Richard	10 jan 2009	High_date	5 jan 2009	15 jan 2009	1
FlatA1	Richard	10 jan 2009	15 jan 2009	15 jan 2009	25 jan 2009	2
FlatA1	René	15 jan 2009	High_date	15 jan 2009	High_date	3
FlatA1	Richard	10 jan 2009	13 jan 2009	25 jan 2009	High_date	4
FlatA1	Harm	13 jan 2009	15 jan 2009	25 jan 2009	High_date	5

Tabel 3: Inhoud tabel 'vastgoed_eigenaar' na transactie nr. 3.

Terugwerkende kracht

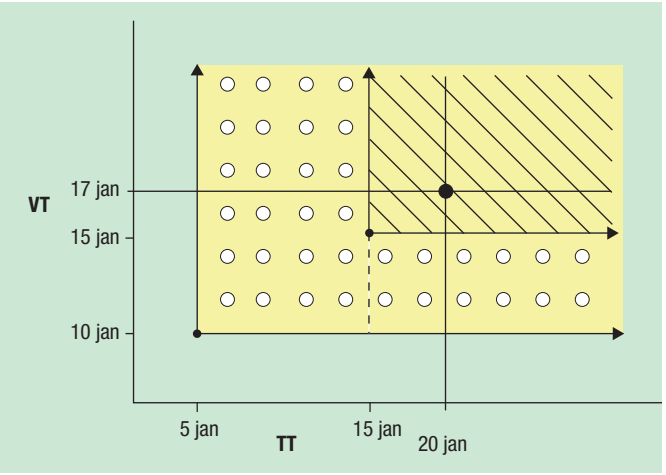
Transactie nr. 3 is een 'eenvoudige transactie met terugwerkende kracht' (TWK). TWK-transacties zijn transacties waarbij de VT_start kleiner is dan minstens één nog geldige VT_start op betreffende TT_start. Grafisch gezien betekent dit dat het beginpunt van de nieuwe toestand (het startpunt van de pijlen van gebiedje row 5 in afbeelding 5) niet in het bovenste rechtervlak begint (het gebied van René), maar in een bestaand horizontaal blok (in dit geval het gebied van Richard).

Het bovenstaande is een 'eenvoudige' TWK-transactie omdat de bestaande toestand die na deze TWK-transactie komt (in VT richting bekeken) niet afhankelijk is van de toestand van deze transactie; René blijft de eigenaar na VT = 15 jan 2009. Deze afhankelijkheid is in vele systemen echter wel aanwezig. Stel we hebben een polissysteem en er wordt een transactie ingevoerd op het punt aangegeven door de ster in afbeelding 6. Voorbeeld: de medewerker blijkt op de transaction time van de ster al vier maanden getrouwd te zijn.

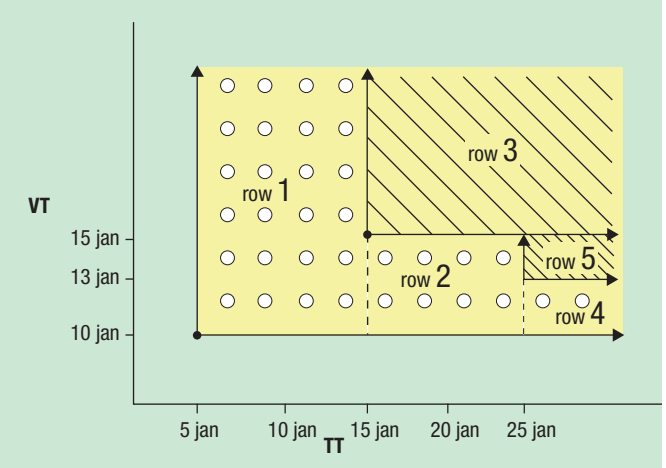
Alle toestanden met een VT_start groter dan de VT_start van de ster moeten in zo'n geval eerst ongedaan worden gemaakt (rewind) om dan, rekening houdend met de TWK-transactie, weer uitgerekend te worden (wind). Grafisch is in afbeelding 7 te zien wat er gebeurt met deze ene TWK-transactie:

- er moeten vier bestaande rows worden afgesloten met een wijziging op TT_end (A1 t/m A4 in afbeelding 7);
- vijf nieuwe rows worden uitgerekend en ge-insert (N1 t/m N5 in afbeelding 7);
- het gebied waar de TWK in valt moet in twee gebieden worden gesplitst (S1 en S2).

Hier stuiten we op een praktisch probleem: één TWK-transactie kan leiden tot vele wijzigingen in de database. De auteur heeft een total recall systeem gezien waar, als men administratief minder handig bezig was en meerdere TWK-transacties uitvoerde, er



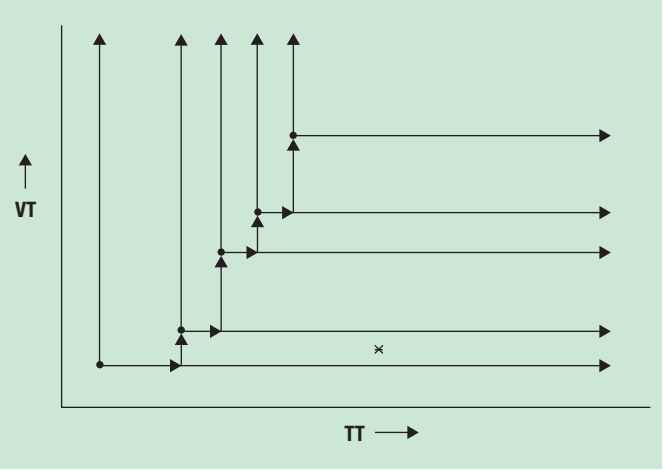
Afbeelding 4: Antwoord op de vraag: "Met de kennis opgeslagen in het systeem op 20 jan 2009, wie is de eigenaar op 17 jan 2009?"



Afbeelding 5: Grafische weergave na het toevoegen van de periode met Harm als eigenaar. Het gebied van Richard is nogmaals verdeeld. Het gebied van René is niet gewijzigd.

Kort parkeren

Er zijn systemen waar valid time en transaction time over elkaar heen vallen. Een voorbeeld hiervan is *kort parkeren* in parkeergarages. Op het moment dat er een kaartje bij een slagboom getrokken wordt begint de parkeertransactie (VT_start) en tevens wordt de transactie op dat moment in het systeem geregistreerd (TT_start). Bovendien is er bij parkeren nooit sprake van TWK-transacties. Het 'geschiedenis van de historie' probleem bestaat in dergelijke systemen niet.

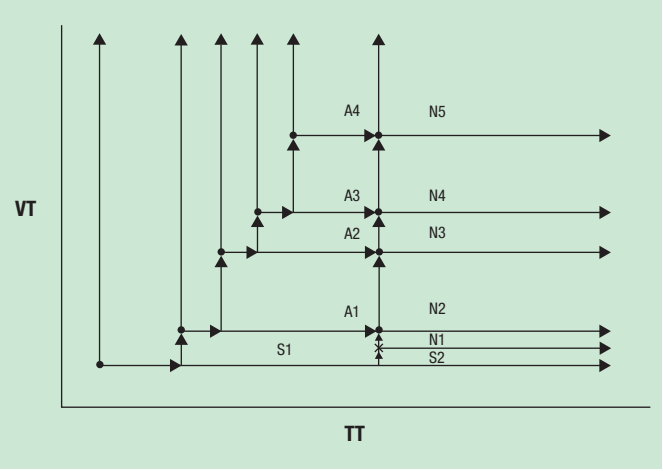


Afbeelding 6: Een TWK-transactie.

vele honderden of zelfs duizenden records voor een enkele polis werden gewijzigd en aangemaakt (en dat vaak ook nog op één en dezelfde TT dag). De praktische gevolgen hiervan laten we aan de verbeelding van de lezer over.

Constraints

Als laatste volgt een korte verhandeling over constraints. Stel, we hebben de constraint 'De flat kan tussen oplevering en sloop nooit zonder eigenaar zijn'. Grafisch betekent dit 'De gebieden moeten zonder gaten op elkaar aansluiten (A)'. Een andere constraint is 'De flat heeft op ieder moment maximaal één eigenaar'. Dat is grafisch gezien 'De gebieden mogen elkaar niet overlappen (B)'. Kort samengevat: we zien dus dat alles er op neer komt om het platte vlak in te delen in 'vlakdelen' met de eigenschappen (A) de vereniging van alle vlakdelen overdekt het hele vlak en (B) de vlakdelen overlappen elkaar niet. Dit komt namelijk precies overeen met de logische eis dat er voor elk punt in het vlak precies één vlakdeel (= rij in tabel) is waar dit punt in ligt. Met ander woorden: de informatie, de waarden van de attributen in de rij, is uniek bepaald.



Afbeelding 7: Eén TWK-wijziging kan vele wijzigingen in de database tot gevolg hebben.

Vastgoed	Eigenaar		BiTempKey
FlatA1	Richard	Meer velden	1
FlatA1	René	Meer velden	2
FlatA1	Harm	Meer velden	3

Tabel 5: Vastgoed_eigenaar.

BiTempKey	VT_start	VT_end	TT_start	TT_end
1	10 jan 2009	High_date	5 jan 2009	15 jan 2009
1	10 jan 2009	15 jan 2009	15 jan 2009	25 jan 2009
2	15 jan 2009	High_date	15 jan 2009	High_date
1	10 jan 2009	13 jan 2009	25 jan 2009	High_date
3	13 jan 2009	15 jan 2009	25 jan 2009	High_date

Tabel 6: Vastgoed_eigenaar_biTemp.

Aanbevolen publicaties

Hierboven is slechts een tipje van de sluier opgelicht. Wil de lezer dit onderwerp echt bestuderen dan kunnen wij een aantal (gratis te downloaden) documenten sterk aanbevelen.

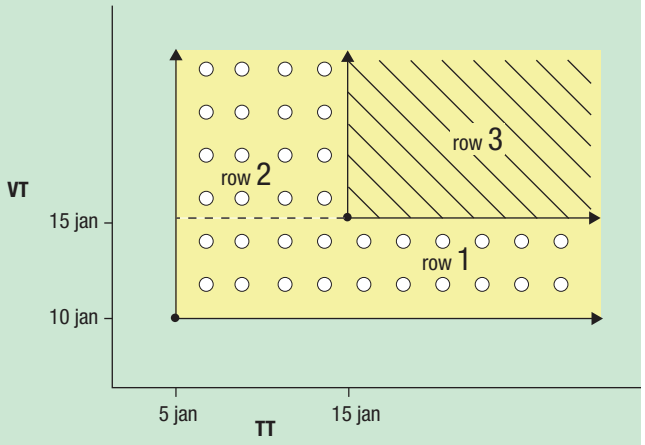
1. Richard T. Snodgrass (1999) 'Developing Time-Oriented Database Applications in SQL'. Snodgrass heeft vele jaren van zijn werkzame leven besteed aan onderzoek van tijdsaspecten in databases en applicaties. Genoemd document beslaat maar liefst 520 pagina's. Het boek heeft in onze ogen in de 11 jaar die verstreken zijn sinds haar publicatie niet veel aan waarde verloren. De grafische presentaties die wij in dit artikel hebben gebruikt zijn gebaseerd op ideeën uit dit boek.
   'Developing Time-Oriented Database Applications in SQL' is oorspronkelijk in boekvorm uitgekomen, maar omdat het niet meer gedrukt wordt is het nu vanaf de website van de auteur gratis te downloaden.
   Ga naar www.cs.arizona.edu/~rts/publications.html en zoek op 'The PDF of this book is here (5MB)'. Klik op 'here (5MB)'.
2. Veldwijk, Cannan, van Orden (1998) 'Tijd in de database'. Veel komt (uiteraard) overeen met Snodgrass. Dat ook dit boek niet veel aan waarde heeft verloren blijkt uit een recent artikel van dhr. Veldwijk, 'De Polis Papers (1): Let's make

Vastgoed	Eigenaar		VT_start	VT_end	TT_start	TT_end	VT2_end	TT2_end
FlatA1	Richard	Meer velden	10 jan 2009	High_date	5 jan 2009	15 jan 2009	15 jan 2009	High_date
FlatA1	René	Meer velden	15 jan 2009	High_date	15 jan 2009	High_date	NULL	NULL

Tabel 4.

Vastgoed	Eigenaar	VT_start	VT_end	TT_start	TT_end	Row
FlatA1	Richard	10 jan 2009	15 jan 2009	5 jan 2009	High_date	1
FlatA1	Richard	15 jan 2009	High_date	5 jan 2009	15 jan 2009	2
FlatA1	René	15 jan 2009	High_date	15 jan 2009	High_date	3

Tabel 7: Inhoud 'vastgoed_eigenaar' na de tweede transactie en ingedeeld als afbeelding 8.



Afbeelding 8: Grafische weergave na de tweede transactie. Het gebied van Richard is nu anders verdeeld; de records in de database zijn verschillend. Het totale oppervlak blijft hetzelfde.

History'. Hierin beschrijft Veldwijk een total recall systeem bij het UWV. Na een bespreking waaruit blijkt dat vrijwel alle tabellen 'dat_van' (VT_start), 'dat_tot' (VT_end) en 'TS_mut' (TT_start) hebben, staat in afbeelding 4 de opmerking: "NB: TS_MUT bestaat feitelijk uit twee timestamps waarmee een interval in de beschouwingtijd wordt weergegeven". Veldwijk bedoelt dat in de tabellen ook TT_end is opgenomen. Een mooi moment van herkenning. 'Tijd in de database' en 'De Polis Papers (1)' zijn beide via www.dbm.nl te downloaden.

Conclusie

In de praktijk blijkt er verwarring te bestaan over de wijze waarop total recall systemen moeten worden geïmplementeerd. De begrippen 'valid time' en 'transaction time' worden vaak niet goed uit elkaar gehouden of de 'vlakken zonder gaten of overlap' worden niet goed ontworpen. Wij hopen met dit artikel het probleem gevisualiseerd en toegankelijker te hebben gemaakt.

Andere weergave

Een andere manier om de situatie uit afbeelding 3 weer te geven is te zien in afbeelding 8. Het gebied van Richard is hierin anders ingedeeld. Zowel de indeling van afbeelding 3 als afbeelding 8 zijn correct. Let echter wel op: de keuze heeft wel invloed op de wijze waarop query's op deze tabel opgebouwd moeten worden. Zie tabel 7.

Slimmer opslaan

Kijk nog eens goed naar tabel 2. Stelt u zich voor dat in deze tabel nog veel meer variabelen zouden staan. Die zouden dan voor de twee rijen waar Richard eigenaar is (row 1 en 2) geheel identiek zijn. Row 1 en 2 in tabel 2 zijn ontstaan uit row 1 van tabel 1, waarbij er alleen mutaties op VT en TT data hebben plaats gevonden. Alle overige variabelen zijn onberoerd gebleven. Als er veel opslagruimte nodig is voor al die extra variabelen, en dit soort splitsingen komt vaak voor in een systeem, dan begrijpt u dat er manieren zijn om dit slimmer op te slaan.

Methode 1: Merk in tabel 2 op dat de VT_start van beide rijen gelijk zijn en dat de TT_end van row 1 aansluit op de TT_Start van row 2. We kunnen met deze kennis tabel 2 omvormen tot tabel 4.

Het idee om de twee rechthoeken die oorspronkelijk een 'L' vormden in de afbeelding, samen te voegen tot een row is afkomstig van Harm van der Lek. Het is verwerkt in BIReady. Het voordeel van deze methode is dat het aantal rijen in de tabel flink kan verminderen, hetgeen opslagcapaciteit spaart en de performance zeer ten goede komt. Bij TWK-transacties (in ons voorbeeld transactie nr. 3) komen we met deze methodiek niet uit onder een nieuwe row voor Richard. Kent een systeem echter geen of weinig TWK-transacties, dan is dit is een goede, efficiënte oplossing.

Methode 2: Splits de oorspronkelijke tabel op in een tabel met de vier datumvelden en een tabel met de overige velden en koppel de tabellen via een gegenereerde, betekenisloze key. We geven als voorbeeld aan hoe tabel 3 er dan uit komt te zien, zie tabel 5 en 6.

Noten

1. *Een periode wordt gekenmerkt door een begin-tijdstip en een eind-tijdstip.*
2. *In Nederland noemen we dit meestal 'mutatieDatumTijd' of een vergelijkbare term met 'mutatie' erin. De auteur kent echter een systeem waar 'mutatiedatum' wordt gebruikt, terwijl 'valid date' wordt bedoeld. Daarom blijven we in dit artikel maar bij de termen die Snodgrass gebruikt: valid time en transaction time.*
3. *We laten tijdzones nu even buiten beschouwing.*
4. *De transaction time zal meestal een datumtijd zijn omdat, vrijwel zonder uitzondering, de systeemklok hiervoor wordt gebruikt. Dit omdat het in de meeste systemen mogelijk moet zijn om meerdere malen op een dag te muteren. Het gebruik van datumtijd is dan een eenvoudige manier om de volgordelijkheid te garanderen. Met het oog op de leesbaarheid van dit artikel tonen we de tijd niet.*
5. *High date is bijvoorbeeld '31 dec 2199'. De discussie of hier een high date of NULL moet staan gaan we hier niet aan, we verwijzen daarvoor naar de literatuur.*

Maarten Zaanen is senior consultant bij PZvK. Met dank aan Harm van der Lek voor de constructieve opmerkingen.