Toestandgeoriënteerde sterschema's roepen vragen op

Geschiedenis van de historie herhaalt zich

Karien Verhagen

Een uniform algoritme dat algemeen toepasbaar is voor "toestandgeoriënteerde sterschema's", zoals Harm van der Lek dat voorstelt, is een ambitieuze doelstelling. Bij nadere beschouwing riep dit bij Karien Verhagen een aantal praktische vragen op, en zij daagt Van der Lek uit om die te beantwoorden. Verder beschrijft Verhagen in dit artikel hoe zij in een vergelijkbare situatie tot een oplossing is gekomen.

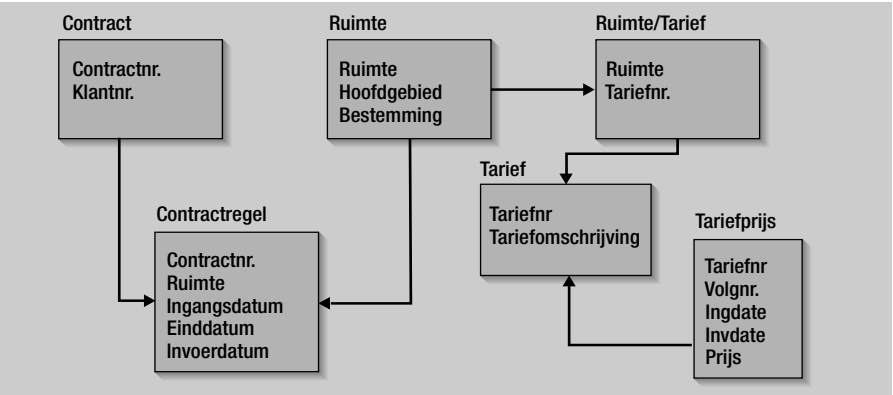
Ik weet niet hoe u erover denkt, maar ik ben dol op artikelen zoals die van Harm van der Lek (zie DB/M 6, 7 en 8 van 2001 onder 'Datawarehouse'). Als je in de praktijk veel met het modelleren en de techniek van het bouwen van datawarehouses bezig bent, weet je hoe lastig het is om het probleem van de historie op een nette en vooral ook 'performante' manier op te lossen.

Als je je centrale datawarehouse 'dweilt' of inwast, heb je nog een groot probleem

Een aantal van Van der Leks denkbeelden probeer ik overigens te verdiepen. De term 'aanbod- en vraaggericht model' heb ik meteen dankbaar geadopteerd.

VASTGOED

Mijn reactie wil ik graag illustreren met de beschrijving van een geval dat dezelfde problematiek, die van *historie van historie*, voor mij actueel maakte. Het betreft een transactioneel systeem voor de exploitatie van vastgoed, waarin de relationele database was ingericht om wijzigingen met terugwerkende kracht betrouwbaar te kunnen verwerken. Niet alleen het tarief, maar ook de contractdatum, de oppervlakte van de ruimte en zelfs de factureerfrequentie konden met terugwerkende kracht worden gewijzigd. De noodzaak om deze verregaande historie van historie vast te leggen, was natuurlijk -net als ongetwijfeld in het voorbeeld van de polisdekkingen- om correct te kunnen factureren. De factuurrun moest kunnen reconstrueren wat in het verleden al in rekening was gebracht en wat, indien van toepassing, op grond van wijzigingen in het verleden moest worden gecorrigeerd. Een sterk vereenvoudigd datamodel dat dit principe weergeeft, ziet eruit als in figuur 1. Een voorbeeld illustreert hoe gegevens worden onttrokken aan dit soort modellen (zie verder kader *Archeologie van de historie*).



Figuur 1: Gegevensmodel vastgoedexploitatiesysteem.

Rapporten opstellen met gegevens uit dit soort modellen geeft het volgende

Vragen aan Harm van der Lek

- Waar staat nu die 'toplaag'? Wat staat er precies in het transactionele systeem, wat in de backroom, wat in het centrale datawarehouse en wat in de eventuele datamart? En wie gebruikt dan wat?
- Als je je centrale datawarehouse 'dweilt' of inwast, heb je nog een groot probleem. Object 2 versie 3, die bij het dweilen in het putje verdwijnt, heeft misschien wel transacties gehad. Helaas heet object 2 versie 3 in het datawarehouse geen object 2 meer, maar heeft hij een system key en heet de bewuste versie 3 nu 57846. Moet je nu ook je je hele facttabel doordweilen? Welke system key moet het dan worden?

Archeologie van de historie

Een datawarehouse is in de tijd gepositioneerd. Het bevat niet alleen de laatste gegevenswaarden, maar alle waarden die een gegeven in de tijd heeft gehad. Als een gegeven ook nog eens met terugwerkende kracht veranderd kan worden, is er sprake van 'historie van historie'.

Willen we een overzicht hebben van de opbrengst van maart naar het inzicht van juni 1995, dan moeten we de prijs reconstrueren zoals die in juni 1995 was, dus zonder de wijzigingen van na die datum - in dit geval wijzigingen 5 en 6. In juni was immers nog niet bekend dat de prijs per 1 augustus 1994 met terugwerkende kracht 20 zou worden. Als we nu de query moeten samenstellen die de &prijsdatum geeft van maart, zoals die was volgens de situatie in juni 1995 (&rapportdatum), hebben we twee gegevens die we als variabelen kunnen meegeven (de prijs met ingang van maart 1995 was 15 gulden):

```
SELECT          PRIJS
   FROM         TARIEF  TA
   WHERE    VOLGNR  =
      SELECT    MAX(VOLGNR)
      FROM      TARIEF TA2
      WHERE     TA.TARIEFNR =
TA2.TARIEFNR
   AND  INVDATE #= &RAPPORTDATUM
   AND  INGDATE #= &PRIJSDATUM )
```

Dezelfde query geeft met rapportdatum juli 2002, dus naar het inzicht vandaag, voor maart 1995 een prijs van 20 gulden. De volledige historische lijn naar het inzicht van een bepaalde datum wordt in pseudocode bijvoorbeeld:

```
Select all into hulptabel
From TARIEF TA
Where INVDATE < &rapportdatum
Order by INGDATE

Dit geeft de hulptabel in de volgorde van
volgnummers 1, 3, 5, 4, 2 en 6.

Select count(*) as aantalrecs
from hulptabel
hoogvolgnr = 0

For teller = 1 to aantalrecs
    Lees record
    IF volgnr > hoogvolgnr
        hoogvolgnr = volgnr
   Else
      delete
next
```

Het record met volgnummer 4 en daarna het volgnummer 2 zullen als overschreven worden verwijderd. Resteert de volledige historische lijn met de volgnummers 1, 3, 5 en 6.

TARIEFNR	PRIJS	INGDATE	INVDATE	VOLGNR
100	10	01/01/1993	01/01/1995	1
100	15	01/03/1995	01/02/1995	2
100	20	01/01/1994	01/03/1995	3
100	15	01/01/1995	01/04/1995	4
100	20	01/08/1994	01/08/1995	5
100	25	01/05/1995	01/09/1995	6

Tabel 1: Voorbeeld records uit de tariefprijs-entiteit.

dilemma. Wilde ik de opbrengstrapportage kunnen reconstrueren zoals die geproduceerd is in de factuurrun van maart vorig jaar, dan *moest* ik de gehele historie van de historie in het datawarehouse 'meenemen'. Het volledige relationele model kopiëren levert echter allesbehalve een goede performance op... Een heel normale vraag van het management als: 'Hoe groot is de leegstandsderving dit jaar?', zou een batchrun vergen van de zwaarte van de hele factuurrun, wat al gauw een uur of vier kostte.

Het alternatief, alleen de maandsnapshots opslaan in het datawarehouse, zou de zekerheid geven dat de organisatie in de toekomst vast komt te zitten met onvoorziene informatiebehoeften. Dat noopt tot platslaan en het introduceren van system keys. Bovendien levert dat het probleem op dat het vastgoedobject afhankelijk van de rapportagedatum naar steeds andere tarieftoestanden moet verwijzen. De gekozen oplossing staat afgebeeld in figuur 2.

De functionele eisen die men aan een toepassing stelt, toekomstvast en waarheidsgetrouw enerzijds en een goede performance en toegankelijkjheid anderzijds, zijn tegenstrijdig. Ontwerp dus eerst een *toekomstlaag*, waarin -in dit geval- de volledige mutatiehistorie wordt bewaard. Dat is het *aanbodgerichte* datawarehousemodel. Deze laag geeft de garantie dat het

> *Een succesvolle BI-implementatie is vrucht van voortdurend geschipper, en dat is misschien juist de charme van het vak*

warehouse antwoord kan geven op alles wat de gebruikers in de toekomst nog aan vragen kunnen verzinnen. Dat lijkt duur, maar aanpassen van het centraal model van een datawarehouse dat al lang en breed in gebruik is, is vele malen rampzaliger. Het is dan ook zaak de hele historie op een zo waarheidsgetrouwe, toekomstvast mogelijke manier te modelleren.

En... ligt dat nu aan mij, heren Kimball en Inmon, maar ik kom dan altijd op een relationeel model!

Daarnaast is er de *vraaggerichte* laag. Die gaat uit van de *actuele* informatiebehoefte van een OLAP-gebruiker, internettoepassing, dashboard-applicatie of een te publiceren rapportage. Deze informatiebehoefte heeft betrekking op een bepaald thema uit het datawarehouse, zoals leegstandsderving. Daarvoor moet een horizontale of verticale subset gemodelleerd worden van de centrale datawarehousegegevens. Het vraaggerichte datamodel voedt zich uit het centrale datawarehouse.
Dat model kan een view zijn of een echt extract, een aggregaat of ook een snap-

Vervolg op pagina 33