Alleen in te zetten bij standen-datawarehouse

Ruimtewinst met pointer-structuren

Oscar Zonneveld

In een vorig artikel is ingegaan op de karakteristieken van Stand- en stroominformatie. Bij standinformatie zijn verschillende methodieken beschreven. Eén van die methodieken die beknopt is beschreven, is het gebruik van pointer-structuren binnen een datawarehouse-omgeving. In dit artikel wordt dieper op de pointer-structuur ingegaan.

Samengevat wordt in het artikel in DB/M 7 gesteld, dat de opslag van standdata (het iedere dag opslaan van de data) een methodiek is die veel ruimte vergt. Om dit ruimtebeslag te beperken kan een andere techniek worden ingezet, namelijk de techniek van het gebruik van een pointer-structuur.
Een pointer-structuur maakt het mogelijk om één tabel in te richten, met daarin twee attributen, waarbij met behulp van een beperkte opslag toch voor iedere dag de totale stand beschikbaar is. Het voordeel van de pointer-structuur is de beperkte opslag voor stand-informatie. Daarentegen wordt de complexiteit van de transformatie wel enigszins vergroot. Het gebruik van de pointer-structuur wordt beschreven aan de hand van een case.

Case beschrijving

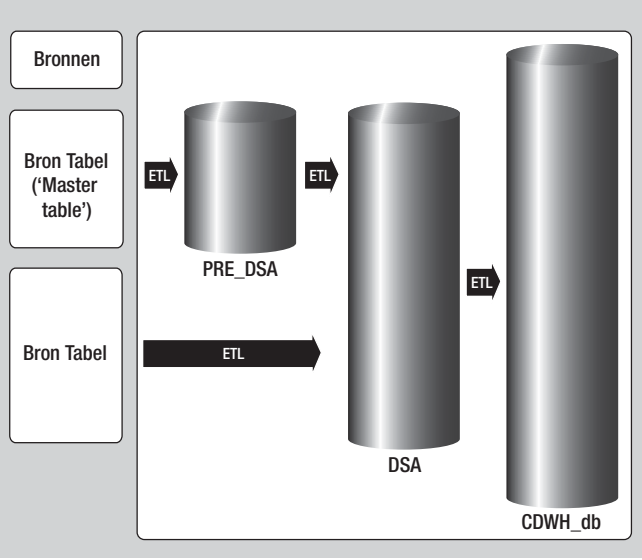
Een zorgverzekeraar wil eigenlijk elke dag in één oogopslag zien wat de huidige stand van de portefeuille is. Bovendien wil hij de mogelijkheid om vanuit de totaalstand verschillende doorsneden en analyses uit te voeren. Om aan zijn informatiebehoefte te kunnen voldoen, wordt een datawarehouse ontwikkeld. Bij analyse van de bronomgeving blijkt dat de huidige OLTP-applicatie een 'master' tabel heeft waarin de actuele stand is opgeslagen. Aan deze tabel zijn vervolgens weer verschillende referentietabellen gekoppeld. Hierbij moet worden gedacht aan bijvoorbeeld een producten- en klantentabel.

Wanneer we kijken naar de vulling van de brontabellen kunnen we vaststellen dat de betreffende 'master' tabel eigenlijk een aggregatie-tabel is. Op het moment dat namelijk een attribuut wijzigt, wordt deze wijziging niet opgeslagen maar wordt het resultaat van de wijziging geregistreerd in de 'master' tabel.
Dit betekent dus dat de oude stand na de mutatie niet meer te achterhalen is. Juist om inzicht te krijgen in het verloop van deze 'master' tabel wordt een datawarehouse ontwikkeld.

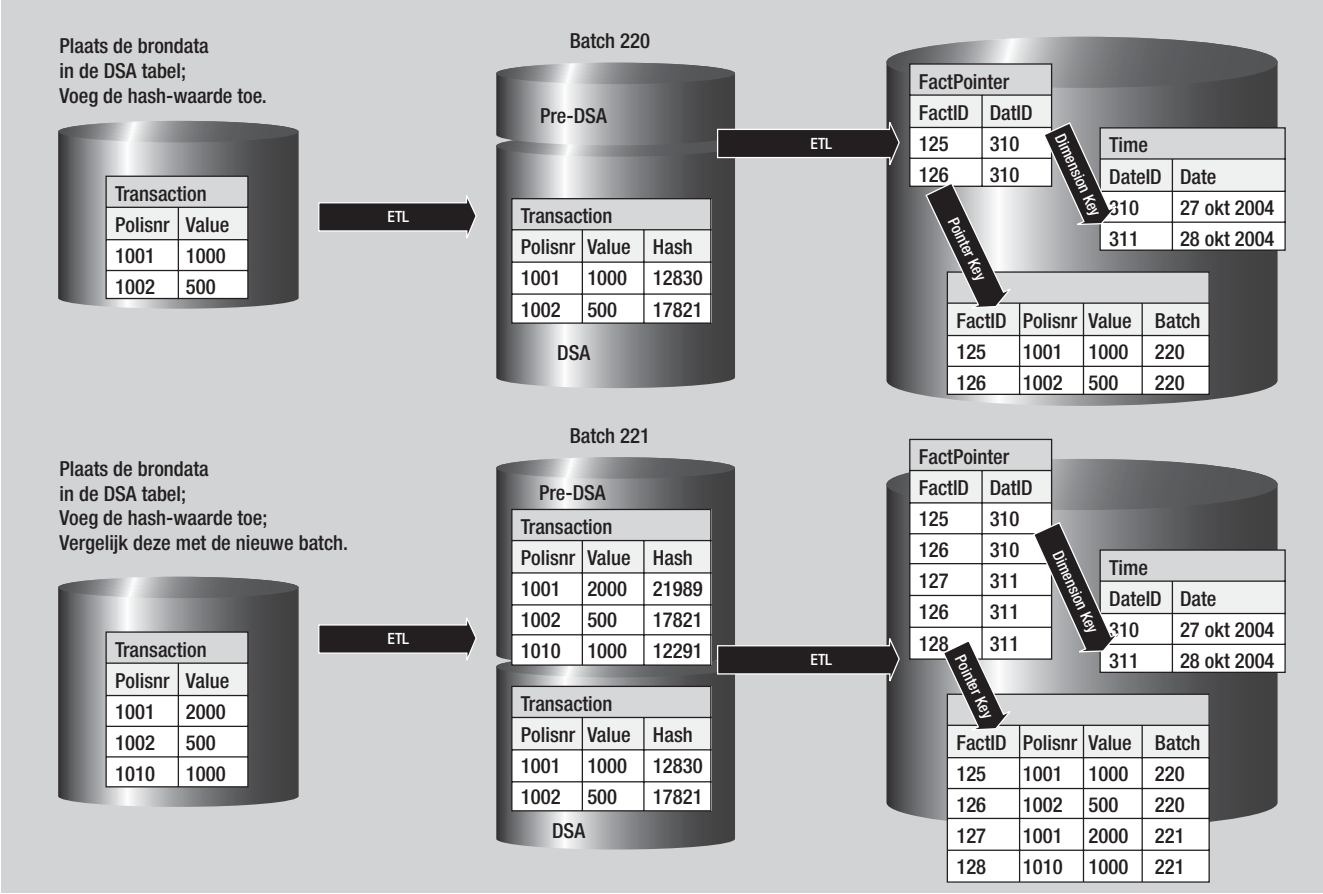
Architectuur

Bij het ontsluiten van de betreffende tabel wordt bij de initiële download de betreffende tabel overgezet in de Data Staging Area (DSA-omgeving). Tijdens de transformatie wordt aan iedere individuele rij een attribuut toegevoegd. Hierin wordt een hash-waarde gezet die bepaald wordt over de totale set aan attributen van de betreffende rij. Deze hash-waarde maakt het mogelijk om de totale betreffende rij één waarde te geven. Verschillen databases hebben functies die het mogelijk maken om de betreffende hash-waarde te bepalen (zoals de functie Binary_checksum).

Nadat getransformeerd is naar de DSA-omgeving en de hash-waarden zijn toegevoegd, kunnen eventuele cleansing-procedures gestart worden. Nadat cleansing-procedures hun werk hebben gedaan (zoals bijvoorbeeld het corrigeren van datumvelden), kunnen de data door middel van een ETL-procedure worden getransformeerd naar de centrale datawarehouse (CDWH) omgeving. Tijdens de transformatie wordt aan iedere individuele rij in de feitentabel een uniek nummer toegevoegd. Dit nummer (ook wel *surrogate key* genaamd) wordt gebruikt om te koppelen richting de pointer-tabel. Naast de surrogate key wordt aan iedere rij een attribuut toegevoegd om te bepalen in welke transformatie-batch deze rij aan het datawarehouse is toegevoegd.



Afbeelding 1: Pointer-architectuur.



Afbeelding 2: Transformatie-pointer.

De pointer-tabel bestaat uit twee attributen. Te weten het unieke nummer uit de feitentabel en de surrogate key van de representerende standdatum uit de datum tabel. Na de transformatie is in de pointer-tabel voor ieder feit uit de feitentabel een rij aanwezig met twee attributen.

Wanneer de initiële dataset is getransformeerd, kan de tweede set worden getransformeerd. Waarbij de transformatie van de initiële dataset een redelijk recht toe rechtaan transformatie is, is de tweede een veel complexere transformatie.

Complexe transformatie

Zoals beschreven in de case van de zorgverzekeraar, worden niet de individuele transacties opgeslagen, maar zorgt de OLTP-applicatie ervoor dat de transacties worden verwerkt in de data (het eindresultaat wordt dus opgeslagen). Om nu te kunnen bepalen of er mutaties zijn in de tweede set ten opzichte van de eerste set zijn er een aantal mogelijkheden.
Stel: betalingen vinden plaats op de verschillende polissen. Deze betalingen worden direct verwerkt in de 'master' tabel. Bij de transformatie van de 'master' tabel naar het datawarehouse worden de data initieel opgeslagen in de DSA. In deze DSA wordt naast de kopie van de 'master' tabel, nog een kopie van de 'master' tabel aangemaakt. In de eerste kopie-tabel (ook wel aangeduid als pre-DSA tabel) worden de data vanuit de bron geplaatst.

Bij het plaatsen van de data worden deze aangevuld met een extra attribuut, namelijk de eerder beschreven hash-total. Bovendien worden de data uit de eerdere batch (aangevuld met de hash-total) getransformeerd van de Pre-DSA naar de DSA-omgeving). De *delta* wordt bepaald door voor iedere rij uit de Pre-DSA omgeving te controleren of een rij in de DSA omgeving aanwezig is. Indien deze aanwezig is wordt aan de hand van de toegevoegde hash-totals bekeken of de betreffende rij gemuteerd is. Indien geconstateerd is dat de rij gemuteerd of nieuw aanwezig is in de Pre-DSA omgeving, kan deze worden getransformeerd naar de centrale datawarehouse (CDWH) omgeving.

De *delta* kan, zoals gesteld, bestaan uit een tweetal verschillende rijen, namelijk een volledig nieuwe rij of een mutatie op een bestaande rij. Rijen die niet gemuteerd worden tijdens de transformaties naar het CDWH niet meegenomen.
De transformatie van deze *delta* naar de CDWH-omgeving gebeurt in twee stappen. Als eerste stap worden de feiten met de daarbij behorende dimensiewaarden opgeslagen in de feitentabellen. Voor deze feiten worden bovendien nieuwe surrogate keys opgeslagen. De transformatie naar de pointer-tabel is echter een complexere aangelegenheid.
Voor het vullen van de pointer-tabel dient namelijk de laatste actuele set van de te transformeren dataset bepaald te worden. De laatste actuele set wordt opgebouwd uit twee verschillende data-

sets. Als eerste wordt alles van de laatste batch (zoals gesteld is dit de dataset waarin de laatste mutaties beschikbaar zijn) geselecteerd. Voor het tweede deel van de te transformeren dataset moet als eerste gelden dat het batch-nummer kleiner is dan die laatste dataset. Bovendien moet gelden dat de surrogate key van de te selecteren feiten de laatste is behorende bij een productsleutel. Kortom van ieder feit in de feitentabel wordt het laatste voorkomen geselecteerd.

> Bij gebruik van een stroom-datawarehouse is het niet verstandig om een pointer-structuur op te nemen

Bij het uitvoeren van selecties voor rapportages worden de gezochte feiten via de pointer-tabel gekoppeld aan de datum waarop het feit of de feiten hebben plaatsgevonden. Door een juiste indexering kunnen de betreffende selecties ondanks vaak grote hoeveelheden toch een optimale performance opleveren.

Het pointer vul-script ziet er als volgt uit:

```
INSERT INTO CDWH..Fact_Pointer
             --Voeg toe aan de feiten pointer-tabel
SELECT Fact_id, Datum_ID
  -- Selecteer de surrogate key van de feiten en de
                                          batchdatum
FROM CDWH..Fact_Polissen
                    -- Selecteer uit de feitentabel
WHERE
(   -- Alle feiten uit laatste batch
(
Batchnummer = @nmBatch_nummer
)
OR
-- Selecteer het laatste voorkomen uit de
           'oude dataset' van iedere productsleutel
(
Fact_id NOT IN (SELECT Fact_id
FROM CDWH..Fact_Polissen
WHERE Batchnummer = @nmBatch_nummer)
AND <Produkt_Key> NOT IN (SELECT <Produkt_Key>
FROM CDWH..Fact_Polissen
WHERE Batchnummer = @nmBatch_nummer)
AND Batchnummer < @nmBatch_nummer
)
)
```

Conclusie

Het gebruik van een pointer-tabel is niet in alle gevallen zinvol. Zoals reeds eerder gesteld, is het gebruik van een pointer-structuur alleen te gebruiken bij een standen-datawarehouse. Bij het gebruik van een stroom-datawarehouse is het niet verstandig om een pointer-structuur op te nemen. Dit omdat de feiten uit een stroom-datawarehouse doorgaans allemaal uniek in de tijd plaatsvinden.

Het gebruik van een pointer-structuur is dan vaak alleen bruikbaar in een omgeving waarin de mutatie op een dataset doorgaans beperkt is. Onder beperkt wordt verstaan 1 tot 10 keer per maand. Indien voor een omgeving de dataset in de bovenstaande frequentie muteert en er wordt niet gekozen voor een pointer-structuur maar wel voor standen, betekent dit dat de betreffende dataset iedere dag wordt opgeslagen. Als gekozen wordt voor de pointer-structuur, worden de feiten maar 1 tot 10 keer opgeslagen per maand (de feitentabel heeft doorgaans meer datacapaciteit), maar in de pointer-tabel wordt wel iedere rij geregistreerd. Echter, hier wordt alleen de surrogate key vanuit de feitentabel opgeslagen (beperkte opslagcapaciteit).

Oscar Zonneveld

Ing. O. Zonneveld (oscarz@infosupport.com) is consultant bij Info Support.