Modelleringsmethode gaat stap verder dan de Data Vault

Anchor Modelling

Ronald Kunenborg

In een modern datawarehouse zijn enkele lagen te onderscheiden, waarvan de gegevensopslaglaag de belangrijkste is. Er zijn verschillende methoden om de opslaglaag te modelleren, variërend van Inmon's derde normaalvorm tot de recente Data Vault methode. Sinds kort is er ook een nieuwe methode, genaamd Anchor Modelling. Hoe deze methode werkt en wat de sterke en zwakke punten zijn staat in dit artikel centraal.

Eerst kijken we naar hoe modelleermethoden ontstaan. Modelleermethoden ontstaan namelijk als reactie op de verschillende eisen waaraan de gegevensopslaglaag van een datawarehouse moet voldoen. Deze eisen zijn in de loop der tijd verschoven, onder invloed van zowel de wijzigende techniek als de veranderende bedrijfsomgeving.

Geschiedenis

In de jaren zeventig waren performance voor gebruikers, zorgen voor een gemeenschappelijk idee van wat de data betekenden en vereenvoudiging van de rapportagebouw belangrijke eisen. Ralph Kimball populariseerde het gebruik van dimensies en feitentabellen als oplossing. Bill Inmon kwam, in reactie op enkele problemen die daarbij ontstonden, met een gegevenslaag in de derde normaalvorm, zonder dimensies.

In de jaren negentig begon de wens op gebied van performance langzaam te verschuiven naar real-time datawarehouses. Mede door steeds grotere hoeveelheden data werd het belangrijk gegevens zo snel mogelijk te ontladen vanuit de operationele systemen. Verder werd het eind jaren negentig door allerlei schandalen en nieuwe wettelijke eisen essentieel dat gegevens waarop wordt gerapporteerd, traceerbaar en controleerbaar zijn. Ten slotte zijn de kosten, vooral personeelskosten, altijd een punt gebleven bij datawarehouses. Om alle wijzigingen in de bronsystemen en rapportages op te kunnen vangen in een klassiek model kost veel tijd en dus geld. Moderne modelleermethoden zoals Data Vault en Anchor Modelling richten zich daarom ook op het beperken van de gevolgen van wijzigingen in de bedrijfsomgeving.

Tabel 1A - 3NF

Naam (Key)	Straat (attribuut)	Geboortedatum (attribuut)
Daniël	Datastraat	13-dec-2002

Afbeelding 1: Derde normaalvorm

Tabel 1B - 6NF

Naam (Key)	Geboortedatum (attribuut)
Daniël	13-Dec-2002

Afbeelding 2: Zesde normaalvorm.

Tabel 1C – 6NF

Naam (Key)	DatumVanaf (Key)	DatumTot (Key)	Straat (attribuut)
Daniël	5-Aug-2009	31-12-9999	Datastraat

Afbeelding 3: Zesde normaalvorm.

Anchor Modelling

Anchor Modelling is een methode die door Olle Regardt is bedacht rond 2002. Deze is geformaliseerd en in 2007 gepubliceerd door Lars Rönnbäck (Affecto) via whitepapers.[1] Anchor Modelling probeert aan diverse eisen tegemoet te komen: goede performance bij laden en opvragen van gegevens; eenvoudig bruikbaar; onderhoudbaar, flexibel en voegt snel waarde toe. Men wil voldoen aan de eisen door datamodellering in de zesde normaalvorm te combineren met een aantal pragmatische uitgangspunten.

Een tikje kort door de bocht geformuleerd staat een tabel in de zesde normaalvorm (6NF) als de tabel alleen de primaire sleutel bevat, en hooguit één attribuut.[2] Zie als voorbeeld de tabellen 1A (het origineel), 1B en 1C in afbeelding 1, 2 en 3.

Tabel 1B en 1C staan in 6NF, want ze bevatten de primaire sleutel en één attribuut. De primaire sleutel kan natuurlijk best uit

Anchor PE_Persoon

PE_ID (PK)	_metadata
#1	#1

Afbeelding 4.

Attribute PENAM_PersoonNaam

PE_ID (PK)	PENAM_Naam (attribuut)	_metadata
#1	Daniël	#1

Afbeelding 5.

meerdere velden bestaan. Tabel 1A staat niet in 6NF want deze tabel bevat teveel attributen en kan worden opgesplitst in de twee tabellen 1A en 1B.
Het voorbeeld maakt meteen de voordelen duidelijk van gegevensopslag in 6NF. Ten eerste kan bij elk attribuut los worden bepaald of dit historisch moet worden vastgelegd, of niet. Een attribuut dat snel verandert zal natuurlijk meerdere keren worden opgeslagen, maar de overige attributen worden niet gedupliceerd. Hierdoor is minder gegevensopslagruimte nodig dan bij andere methoden.

Aanvullingen

Anchor Modelling gebruikt de zesde normaalvorm, maar met wat aanvullingen. Ten eerste komt er een extra veld voor metadata in elke tabel, waardoor de tabellen strikt genomen niet meer in de zesde normaalvorm staan. De metadata worden geïmplementeerd als een tabel met een surrogate key '_metadata', die in alle tabellen wordt gebruikt.
Ook de omgang met tijd in Anchor Modelling is iets anders dan bij de standaard zesde normaalvorm. In het voorbeeld zien we twee datumvelden om de geldigheid van een attribuut in de database aan te geven als een tijdsinterval. In Anchor Modelling gebruiken we alleen de 'datum vanaf'. De einddatum van het record wordt impliciet bepaald door de startdatum van het volgende record. Hiermee voorkom je dat je bij het laden van nieuwe gegevens ook oudere records moet aanpassen. Dit is wat men in Anchor Modelling een 'zero update strategie' noemt. Het bijwerken van bestaande gegevens is hierdoor alleen nodig als er verkeerde gegevens zijn geladen.
Wat betreft tijd zijn er verschillende soorten tijd: ten eerste de tijd waarop dingen in de werkelijkheid plaatsvinden, 'valid time'. Dit wordt ondergebracht in Attributes zoals OrderDatum. De tijd waarop de vastlegging van het feit geldig is, de 'transaction time', leggen we vast in de DatumVanaf velden die bij het attribuut horen. Ten slotte hebben we ook nog de tijd waarop de gegevens in het datawarehouse bekend werden. Deze tijd slaan we op in de metadata.
Overigens verschil ik op dit punt van mening met Lars Rönnback, die 'valid time' in de DatumVanaf velden plaatst, de 'transaction time' in de metadata, en vervolgens een derde tijdsoort ('happening time') definieert die dan in de attributes moet worden vastgelegd. Een uitwerking van zijn standpunt in onze lopende discussie hierover staat momenteel op de Anchor Modelling website.[3] Gezien de lastige constructies die zijn manier van werken noodzaakt zou ik adviseren hem hierin niet te volgen.

Sleutels

Een belangrijk onderwerp in elke methode is de omgang met sleutels. Anchor Modelling behandelt deze 'business keys', de attributen die voor de medewerkers in een organisatie het mogelijk maken iets terug te vinden, als gewone attributen. De primaire sleutel die de attributen van een business key met elkaar verbindt is een surrogaatsleutel, een ID. Deze wordt opgeslagen in een Anchor, waar de methode haar naam aan ontleent. Anchors krijgen in de naam een prefix van twee letters, die terugkomt in alle daarmee verbonden tabellen. De sleutelvelden zelf worden opgeslagen als attributen op het Anchor en niet speciaal behandeld. Op het uitdelen van surrogaatsleutels kom ik later nog terug.
De attributen worden opgeslagen in tabellen die, heel verrassend, Attributes heten. Deze bevatten de surrogaatsleutel en één attribuut. Verder krijgen Attributes een naam met een prefix van vijf letters, waarvan de eerste twee letters afkomstig zijn van het Anchor waar ze bij horen. De overige drie letters in de prefix baseert men op het attribuut.
Voor alle tabellen wordt de prefix in de naam gevolgd door een underscore en een beschrijvende naam in 'CamelCase', waarbij de woorden aan elkaar worden geplakt en met een hoofdletter beginnen. Het gebruik van een consistente naamgeving met pre-

Attribute PEGEB_PersoonGeboortedatum

PE_ID (PK)	PEGEB_Geboortedatum (attribuut)	_metadata
#1	13-Dec-2002	#2

Afbeelding 6.

Attribute PESTR_PersoonStraat

PE_ID (PK)	PESTR_Vanaf (PK)	PESTR_Straat (attribuut)	_metadata
#1	5-Aug-2009	Datastraat	#3

Afbeelding 7.

Metadata

_metadata (PK)	LaadDatum	Bron	Type
#1	4-Aug-2009	PersoonsRegister	Handmatig
#2	4-Aug-2009	PersoonsRegister	Automatisch
#3	5-Aug-2009	AdresSysteem	Automatisch
...	...	...	...

Afbeelding 8: De Metadatatabel met enkele voorbeeldkolommen. Deze kunnen naar believen worden uitgebreid.

Anchor OR_Order

OR_ID (PK)	_metadata
#1	#4

Afbeelding 9.

Attribuut ORNUM_OrderNummer

OR_ID (PK)	ORNUM_Nummer (attribuut)	_metadata
#1	#101	#5

Afbeelding 10.

Tie ORPE_Order_Persoon

OR_ID (PK)	PE_ID (PK)	ORPE_Vanaf (PK)	_metadata
#1	#1	5-Aug-2009	#6

Afbeelding 11.

fixen in de tabelnaam en veldnamen zorgt er voor dat je meteen conflicten krijgt in de naamgeving als je het model niet goed opzet.[4]

Als we het eerder genoemde voorbeeld omschrijven op de hierboven beschreven manier dan krijgen we de situatie zoals weergegeven in afbeelding 4 tot en met 8. Tot zover hebben we nu de Anchors voor keys en de Attributes voor de attributen. Maar wat doen we als we een Persoon een order laten plaatsen? Allereerst hebben we een Anchor nodig voor de orders, zie de tabellen in afbeelding 9 en 10. Nu moeten we Persoon en Order nog verbinden. Om twee of meer Anchors te verbinden gebruiken we een *Tie*. Een Tie bestaat uit de ID's van de verbonden Anchors, gecombineerd met metadata en eventueel een Vanaf-datum voor relaties die over de tijd heen veranderen. Een Tie heeft geen Attributes.

Ties krijgen een naam met een prefix van vier letters, overgenomen uit de letters van de verbonden Anchors. In het geval dat er meer dan twee Anchors met elkaar worden verbonden, moet men een conventie afspreken over welke Anchors daarvoor worden gebruikt, bijvoorbeeld de eerste twee op alfabetische volgorde. De beschrijvende naam bestaat uit de namen van de verbonden Anchors, verbonden met een underscore, zie de tabel in afbeelding 11.

Stel dat we nu de relatie tussen Order en Persoon willen voorzien van een type, bijvoorbeeld 'spoedorder'. Dat kunnen we prima doen door een Anchor te maken voor Ordertypes, en vervolgens dat Anchor in de Tie te hangen. Deze constructie komt echter zo veel voor dat er in Anchor Modelling een speciale constructie voor is gemaakt, de *Knot*.

De Knot is een combinatie van een Anchor met een attribuut, waarbij het attribuut niet mag veranderen in de tijd. Dit is eigenlijk een klassieke referentietabel. Deze kunnen we overal toepassen waar Anchors, Tie's of Attributes moeten worden getypeerd. De Knot heeft een prefix van drie letters, gebaseerd op de typenaam. Een Knot voor bijvoorbeeld Ordertype (spoed, regulier) wordt dan bijvoorbeeld als volgt opgezet. Er wordt geen geschiedenis bijgehouden – als je dat wel zou willen, dan moet je overwegen een Attribute te maken met historie, waar de Knot aan wordt vastgeknoopt, zie afbeelding 12. Deze Knot moet natuurlijk ook aan de Tie worden geknoopt, zie afbeelding 13. Zoals men kan zien krijgen Ties of Attributes die worden verbonden met een Knot geen naamswijzigingen.

Het grote voordeel van de eenvoud van deze tabellen is dat je ze automatisch kan aanmaken. Wie een tabel met 100 attributen handmatig wil omzetten naar een Anchor model, heeft daar best een behoorlijke klus aan. Gelukkig zijn er XML- en XSLT-bestanden beschikbaar op de Anchor Modelling website die vanuit een modelbeschrijving een create script voor de views en tables aanmaken (op dit moment overigens alleen voor Microsoft SQL Server).

Sleutelmanagement

Bij het inladen van gegevens moeten de surrogaatsleutels van de Anchors worden uitgedeeld. Dat betekent dat we de natuurlijke sleutels ofwel in de database, ofwel in de ETL moeten bijhouden. Het is vrij eenvoudig een view te bouwen op een Anchor en de bijbehorende attributen die de natuurlijke sleutel vormen, zodat extra opslag van de sleutels in principe niet noodzakelijk is.

Views en functies

Views zijn overigens hard nodig om de enorme hoeveelheid tabellen te kunnen hanteren. Anchor Modelling definieert twee soorten views die hierbij nuttig zijn. De 'complete' view denormaliseert de Anchor tabel en de bijbehorende attributen en wordt gevormd door een left outer join van het Anchor met zijn Attributes. De 'latest' view is een overzicht van de laatste stand van zaken en wordt gevormd door de 'complete' view in te perken tot alleen de meest actuele historische waarden.

Verder wordt binnen Anchor Modelling gebruik gemaakt van een 'point-in-time' functie om de attributen op een bepaald moment in de tijd te kunnen bekijken. Deze functie neemt van elk attribuut de waarde die op het gegeven moment actueel was. Ook is er nog de Interval functie. Hiermee worden uit de

Knot ORT_Ordertype

ORT_ID (PK)	ORT_Ordertype (attribuut)	_metadata
#1	Spoed	#7
#2	Regulier	#8

Afbeelding 12: Knot.

Tie ORPE_Order_Persoon

OR_ID (PK)	PE_ID (PK)	ORT_ID	ORPE_Vanaf (PK)	_metadata
#1	#1	#1	5-Aug-2009	#6

Afbeelding 13:Tie.

'complete' view alle waarden van de attributen van een Anchor getoond die geldig waren binnen een bepaald tijdsinterval.
Met bovenstaande functies en views, die ook automatisch kunnen worden aangemaakt, is een goed historisch beeld te krijgen van de inhoud van het datawarehouse. Eventueel kunnen hier bovenop weer sterschema's worden gebouwd, als de gekozen OLAP-tools dit nodig hebben.

Performance

Om de performance van de methode te verhogen worden bij de primaire sleutels de DatumVanaf velden altijd als 'descending' opgeslagen. Dit zorgt voor snellere 'latest' views. Ook wordt stevig gebruik gemaakt van indexen, waarbij de index bij voorkeur gebruik maakt van de al opgeslagen gegevens. Verder wordt er geen gebruik gemaakt van referentiële integriteit in de database. Ondanks deze maatregelen moeten query's erg veel tabellen aflopen. Gelukkig kunnen moderne query optimizers overweg met een techniek die 'tabel eliminatie' heet.[5] Hierbij worden alle tabellen waar geen veld van wordt aangesproken uit de query verwijderd. Dit zorgt er voor dat bij een query over enkele attributen de performance beter is dan bij een database met dezelfde attributen in slechts enkele tabellen. Het verschil wordt groter naarmate meer attributen worden opgeslagen. Meer materiaal hierover is te vinden op de Anchor Modelling website.[6]

Voordelen en nadelen

De voordelen van Anchor Modelling zijn onder andere de eenvoud in ontwerp en bouw, evenals de goede performance bij query's. Door het gebruik van losse attributen en meer-op-meer relaties tussen Anchors kan men data uit meerdere bronnen asynchroon inladen en elk attribuut onafhankelijk van elkaar voorzien van versiegeschiedenis en typering. Het beperkt aantal soorten tabelvormen en de gehanteerde naamsconventie zorgen er voor dat de methode foutbestendig en goed te automatiseren is. Verder zorgt het splitsen van alle attributen in losse tabellen er voor dat ontwikkelaars vrijwel alle aanpassingen los van elkaar kunnen uitvoeren. Dit maakt het model ook minder gevoelig voor wijzigingen in de bronsystemen. Het verdwijnen van attributen uit de bron zorgt niet voor gaten in de data omdat we geen 'null'-waarden opslaan en het toevoegen van attributen heeft geen impact op bestaande Attributes.
Een nadeel van Anchor Modelling is dat het niet geschikt is voor databases die niet aan 'table elimination' doen, zoals MySQL, dus je moet van tevoren testen of dit concept hanteerbaar is voor de gebruikte database. Voor datamining is de methode ook minder geschikt. Ten slotte blijft de overmaat aan tabellen, functies en views ondanks automatisering lastig voor de ontwikkelaars.

Conclusie

De Anchor Modelling methode lijkt sterk op Data Vault modelling. De verschillen zitten voornamelijk in de naamgevingsconventie, de tot in het uiterste doorgevoerde opsplitsing van attributen en de omgang met de 'business keys'. Een aantal van de sterke punten, zoals onafhankelijk laden van gegevens, is in beide methoden aanwezig. Anchor Modelling voert het opsplitsen van attributen echter verder door en heeft daardoor enkele voordelen. De methode is duidelijk nog in ontwikkeling, maar is in Zweden al enkele jaren in gebruik bij redelijk grote (4 TB) datawarehouses. Wie nu nieuwbouw of uitbreiding overweegt zou daarom naar deze methode moeten kijken als solide alternatief voor oudere methoden en zelfs voor Data Vault.

Noten

1. *Anchor Modelling in the data warehouse, Lars Rönnback, 2007. www.anchormodelling.com*
2. *C.J. Date, H. Darwen, and N.A. Lorentzos, Temporal Data and the Relational Model, Elsevier Science, 2003 (Ch. 10.4).*
3. *Three concepts of Time in Anchor Models, Lars Rönnback, Augustus 2009. www.anchormodelling.com.*
4. *Anchor Modelling Cheat Sheet, TDWI, 2007, www.anchormodelling.com.*
5. *Why are some of the tables in my query missing from the plan?, Juni 2008, Inside the Oracle Optimizer – Removing the Black Magic, Oracle Optimizer Development Group. (http://optimizermagic.blogspot.com/2008/06/why-are-some-of-tables-in-my-query.html).*
6. *Anchor Modelling Performance, www.anchormodelling.com.*

Ronald Kunenborg

Drs. R. Kunenborg (info@grundsatzlich-it.nl) is zelfstandig BI-consultant bij Grundsätzlich IT.