Gestructureerde aanpak leidt tot hogere kwaliteit op alle fronten

Modelgedreven ontwerp van ETL-functies

Mark Zwijsen

Vrijwel iedere datawarehouse-implementatie bevat een ETL-component. Een ETL-programma onttrekt gegevens uit een bronsysteem (Extract), transformeert het in de gewenste vorm (Transform) en laadt de getransformeerde gegevens in een doelsysteem (Load).

Binnen de meeste datawarehouse-projecten wordt een zeer groot aantal ETL-programma's ontwikkeld. Een gemiddeld datawarehouse bevat al snel tientallen, zo niet honderden tabellen (feiten, dimensies, aggregaten), en het aantal ETL-programma's in een datawarehouse overstijgt vaak het aantal tabellen in dat datawarehouse. Dit komt omdat er vrijwel altijd voor gekozen wordt om per tabel een apart ETL-programma te bouwen dat het laadproces van die tabel verzorgt. En in veel gevallen blijft het niet bij één laadproces per tabel. Indien een tabel gegevens bevat die afkomstig zijn uit meer dan één bronsysteem, dan zal er per bronsysteem een apart laadproces worden gebouwd.
Een datawarehouse is gedurende de gehele levensduur aan verandering onderhevig. Door veranderende eisen van gebruikers, veranderingen in de bedrijfsprocessen of in de markt waarin het bedrijf opereert, zijn aanpassingen of uitbreidingen aan het datawarehouse nodig. Dit betekent dat er nieuwe tabellen en ETL-programma's gebouwd worden en dat er bij voortduring bestaande programma's aangepast worden.
Een efficiënte methode voor het ontwerp en de bouw van ETL-programma's levert dus winst op voor een datawarehouse-project.

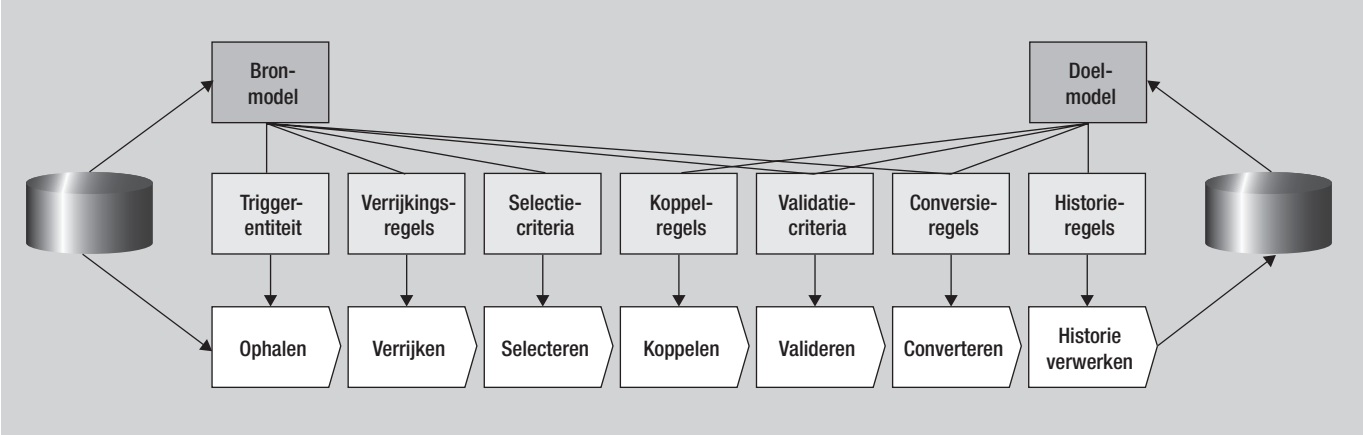
Logisch ontwerp versus technisch ontwerp

Het ontwerp- en bouwproces van ETL-programma's is in wezen niet anders dan dat van 'andere' gegevensverwerkende systemen. Ook hier wordt vanuit functionele eisen een technisch product vervaardigd. Voor het beschrijven van de gewenste transformatie-functionaliteit wordt echter vaak geen duidelijk onderscheid gemaakt tussen een logisch en een technisch niveau. Daarnaast worden voor het ontwerpen van een ETL-programma op logisch niveau geen formele specificatietalen gebruikt. Iets wat bij het modelleren van de gegevensstructuur vrijwel altijd gebeurt, ook bij datawarehouses.
Bij datamodellering wordt onderscheid gemaakt tussen een logisch en een technisch datamodel. Het logische niveau richt zich op de structuur, de betekenis en de samenhang van de gegevens, zonder enige wetenschap van, of rekenschap met de consequenties van het te gebruiken Database Management Systeem. Hier wordt vaak Entity Relationship-modellering als specificatietaal toegepast. Het technische niveau richt zich op de wijze waarop het gegevensmodel in een specifiek DBMS geïmplementeerd wordt.

Bij het ontwerp van ETL-programma's wordt deze scheiding tussen een logisch niveau en een technisch niveau weinig toegepast. De ontwerpdocumenten bevatten vaak een mengeling van functionele en technische beschrijvingen en specificaties. De gebruikte taal is informeel. Dit betekent dat deze vaak dubbelzinnig en onvolledig is, en niet volgens standaard jargon is opgebouwd, waardoor de programmeur zelf aannames moet maken en interpretaties moet doen. De kwaliteit van het eindresultaat (het ETL-programma) komt hierdoor in gevaar.
De ontwerpdocumentatie is ook in veel datawarehouse-projecten vaak het sluitstuk. Het wordt pas vervaardigd wanneer de programma's al gebouwd zijn, en dan is de verleiding groot om vanuit de software-code terug te redeneren (reverse engineering). Dat er in zulke ontwerpdocumenten veel tool-specifieke zaken voorkomen, is dan ook niet verwonderlijk.

Kwaliteitsaspecten van een ontwerp

Door deze praktijk ontstaat er nauwelijks kwalitatief goede functionele ontwerp-documentatie. De vraag kan worden gesteld waarom het belangrijk is om goede functionele documentatie te hebben. Hiervoor is een nadere detaillering van het begrip 'kwaliteit' nodig. In de publicaties van Cavano en McCall en van Boehm over aspecten van de kwaliteit van software-componenten staat het begip 'blijvende bruikbaarheid' centraal. De blijvende



Afbeelding 1: Het transformatiemodel.

bruikbaarheid van een product (in ons geval dus een functioneel ontwerp) wordt bepaald door de huidige bruikbaarheid, de onderhoudbaarheid en de overdraagbaarheid.
De ontwerpdocumentatie die er is, is vaak geheel toegeschreven naar de architectuur en toepassingswijze van de gekozen ETL-tool. Dit heeft negatieve gevolgen voor onder andere de onderhoudbaarheid op functioneel niveau (de structuur, begrijpelijkheid en de testbaarheid) en de overdraagbaarheid. Een overweging om (deels) over te stappen op een ander ETL-tool wordt dan belemmerd door de techniek-afhankelijkheid van de documentatie.
Zuiver functionele ontwerpen zijn dus van belang voor de onderhoudbaarheid en overdraagbaarheid van de datawarehouse-programmatuur.

Een universeel functioneel model voor ETL-programma's vormt een bruikbaar uitgangspunt voor goede functionele ontwerpen. Immers, dit heeft zich bij het modelleren van data al ruimschoots bewezen. De vraag die zich hierbij voordoet is hoe zo'n model er dan uit moet zien, wil het voldoen aan de hierboven beschreven kwaliteitseisen. Om deze vraag te beantwoorden, is een analyse gedaan van de verzameling van ETL-programma's van een aantal datawarehouse-implementaties. De gegevensmodellen van deze datawarehouses zijn gebaseerd op sterschema's.

De binnen deze datawarehouses gerealiseerde ETL-programma's vertonen een grote mate van functionele overeenkomst met elkaar. Hierbij geldt de overbekende 80-20 regel: 20 procent van de ETL-programma's doet iets zeer specifieks of uitzonderlijks, wat afwijkt van het algemene patroon.
Deze algemene werking van een ETL-programma kan op hoofdniveau in twee delen worden gesplitst:

1. In het eerste deel wordt een modeltransformatie toegepast. Men zou ook van een gedaantewisseling kunnen spreken: de gegevens hebben in aanvang de gedaante van het bronmodel, en door de transformatie wordt een nieuwe gedaante aangenomen, namelijk die van het doelmodel;
2. In het tweede deel worden de gegevens, die inmiddels de 'doelgedaante' hebben, verwerkt in het datawarehouse, waarbij onder andere zaken als historie-afhandeling (slowly changing dimensions, specifiek voor dimensies) worden gedaan.

Dit tweede deel is redelijk triviaal, en hiervoor wordt door veel ETL-tools geautomatiseerde ondersteuning geboden (wizards). Met een beperkte set regels kan dit tweede deel worden ontworpen en ontwikkeld.
Het eerste deel is eigenlijk de werkelijke transformatie. Hierin ondergaan de gegevens de werkelijke gedaantewisseling. De gegevens uit het bronsysteem zijn de kandidaatgegevens voor het datawarehouse. Een kandidaat ondergaat als het ware een gedaantewisseling om in de doelomgeving te worden opgenomen. De gedaante van de kandidaat verandert van *brongedaante* naar *doelgedaante*.

Mogelijke structurering van het transformatie-proces

De transformatie wordt onderverdeeld in een beperkt aantal deelfuncties, die transformatiestappen worden genoemd: 1. Ophalen; 2. Verrijken; 3. Selecteren; 4. Koppelen; 5. Valideren; 6. Converteren. (Zie afbeelding 1.) De weergegeven stap 'Historie verwerken' wordt in dit artikel verder buiten beschouwing gelaten.

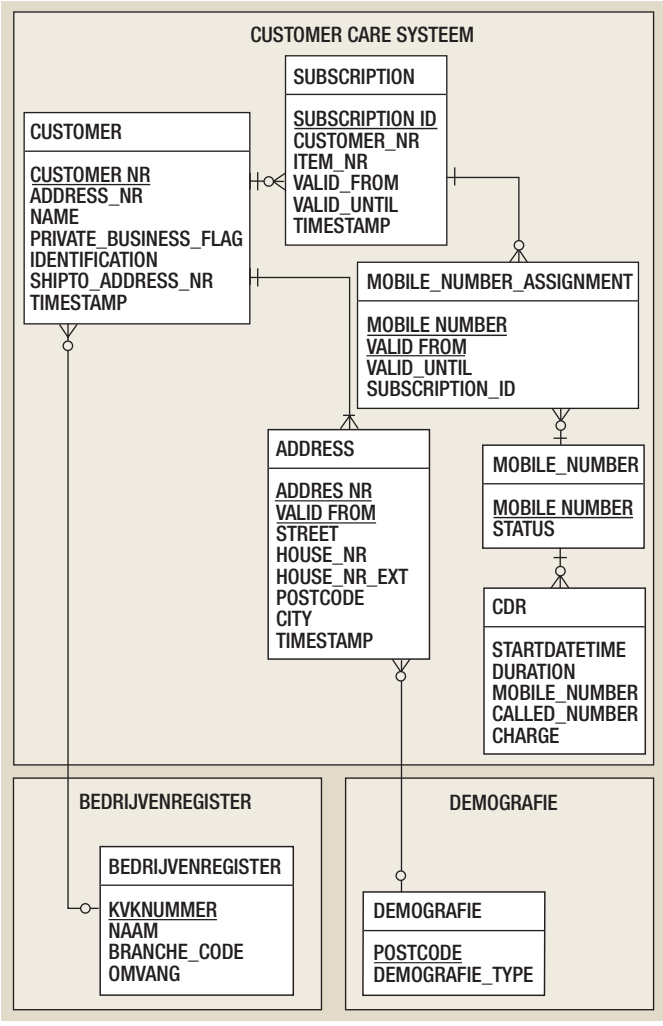
De transformatiestappen zijn als volgt te groeperen:

1. Bronmodel-geöriënteerd: De regels zijn voornamelijk gerelateerd aan het datamodel van de bronverzameling: Ophalen, Verrijken, Selecteren. Hier wordt de brongedaante van de kandidaten opgebouwd;
2. Doelmodel-geöriënteerd: De regels zijn voornamelijk gerelateerd aan het datamodel van de doelverzameling: Koppelen en Converteren. Hier wordt de doelgedaante van de kandidaten opgebouwd;
3. Kwaliteit-geörienteerd: De regels betreffen de kwaliteit (volledigheid, juistheid) van de doelgedaante en zijn gerelateerd aan correctieve of adaptieve acties: Valideren.

Het principe achter deze opdeling en volgorde is dat soortgelijke

Transformatiestap	Output
OPHALEN	Alle kandidaten met de impliciet aanwezige attributen
VERRIJKEN	Alle kandidaten met de direct *en indirect* gekoppelde attributen
SELECTEREN	Alle *relevante* kandidaten met de direct *en indirect* gekoppelde attributen
KOPPELEN	Alle *relevante* kandidaten met de direct *en indirect* gekoppelde attributen, *plus de verwijzende sleutelattributen van het doelmodel*
VALIDEREN	Alle *relevante en kwalitatief goedgekeurde* kandidaten met de direct *en indirect* gekoppelde attributen, *plus de verwijzende sleutelattributen van het doelmodel*
CONVERTEREN	Alle *relevante en kwalitatief goedgekeurde* kandidaten *met alle benodigde attributen voor het doelmodel*

Afbeelding 2: Resultaat van de functionele transformatiestappen.



Afbeelding 3: Sterschema.

beslissingen over, of bewerkingen op de te verwerken gegevens, bij elkaar gebracht zijn in één transformatiestap, en dat alle benodigde input voor een stap wordt gevormd door de output van de eraan voorafgaande stap. De output van elke stap is kort omschreven in afbeelding 2.

Een belangrijk uitgangspunt van deze functionele decompositie is dat deze weliswaar een globale volgtijdelijkheid in zich bergt, maar dat deze volgtijdelijkheid in de uiteindelijke technische vertaling er anders uit kan komen te zien. Dit technische beeld wordt namelijk, behalve door de te realiseren functionaliteit, ook gedreven door de mogelijkheden van de gebruikte ETL-tool en door 'tuning'-activiteiten, gericht op optimale prestatie in de operatie.

Bij het verder uitwerken en toelichten van de componenten van het transformatiemodel wordt de metafoor van de tabel gebruikt. De gegevens die zichtbaar gemaakt worden zijn symbolische gegevens. Dit vereist enige verbeeldings- en interpretatie-inspanning van de lezer. Het voordeel is dat de tabellen beperkt van omvang blijven, hetgeen het lezen weer prettiger maakt.

De koppeling naar de praktijk wordt gelegd met een voorbeeld dat afkomstig is uit de telecommunicatiewereld. Het betreft het ETL-proces voor een klant-dimensie. Een tweede voorbeeld met betrekking tot gespreksfeiten wordt in een vervolgartikel behandeld. Het sterschema met deze dimensietabel is in afbeelding 3 weergegeven.

De klant-dimensie wordt gevoed uit drie bronsystemen:

1. Het 'customer care' systeem, hierin bevinden zich de werkelijke klantgegevens;
2. Een tabel met demografische gegevens van consumenten, die periodiek wordt ververst;
3. Een tabel met bedrijfsgegevens, afkomstig van de Kamer van Koophandel. Ook deze wordt periodiek ververst.

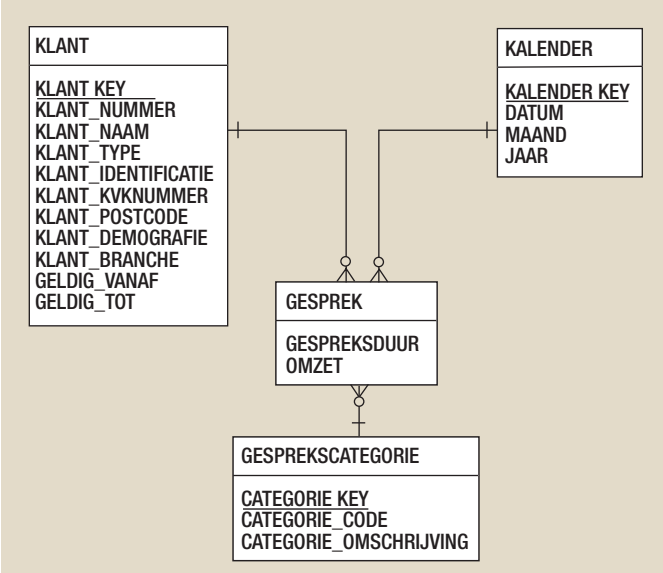
Een aansluiting is gekoppeld aan een klant. Het datamodel is weergegeven in afbeelding 4.

De voorkomens in de CUSTOMER tabel met een CUSTOMER_NR onder 1000 worden zo nu en dan voor testdoeleinden aangemaakt, en zijn niet relevant voor het datawarehouse. De klantdimensie-tabel wordt dagelijks bijgewerkt met de mutaties van de vorige dag.

Transformatiestap Ophalen: alle kandidaten uit de trigger-tabel

De eerste transformatiestap is het uitlezen van de *trigger-tabel*. Het Trigger-entiteittype representeert de wezenlijke kandidaten die verwerkt moeten worden. Deze wordt trigger-entiteittype genoemd omdat mutaties in entiteiten van dit type de oorsprong zijn van de mutaties in de doelverzameling. De bij deze stap behorende informatie in het functioneel ontwerp bestaat dus uit de naam van het trigger-entiteittype. Voor de klantdimensie is CUSTOMER het trigger-entiteittype.

De resultaatverzameling van deze transformatiestap bevat alle kandidaten met de impliciet aanwezige attributen, zie afbeelding 5.



Afbeelding 4: Datamodel brongegevens.

Verrijken: attributen toevoegen uit andere brontabellen

De kandidaten van het trigger-entiteittype bevatten nog niet hun gehele brongedaante. Het is nodig deze kandidaten te 'verrijken' met attributen uit gerelateerde entiteiten.
De redenen hiervoor zijn onder andere:

1. Denormalisatie (ten behoeve van dimensie in een dimensioneel model);
2. Combinatie: Samenvoegen van informatie uit verschillende bronnen.

Voor de klantdimensie dienen de kandidaten verrijkt te worden met adresgegevens uit ADDRESS, bedrijfsgegevens uit BEDRIJVENREGISTER en demografische gegevens uit DEMOGRAFIE.
Omdat de relatie tussen CUSTOMER en ADDRESS een 1-op-veel relatie is, kan niet zonder nadere restricties het geldige adres van de klant worden vastgesteld. Een restrictie op het attribuut VALID_FROM is nodig om eenduidig het adres te bepalen. Op basis van het attribuut POSTCODE van dit adres kan, via de relatie met DEMOGRAFIE, het attribuut DEMOGRAFIE_TYPE van de klant worden vastgesteld.
Het komt er bij het Verrijken dus feitelijk op neer, dat de relaties tussen het Trigger-Entiteittype en de overige relevante entiteittypen zodanig worden gespecificeerd, dat ondubbelzinnig de juiste attributen worden toegevoegd aan de brongedaante.
De resultaatverzameling van deze stap bevat alle kandidaten met de direct *en indirect* gekoppelde attributen, zie afbeelding 6.

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3	**Referentietabel B**	Attribuut B1	Attribuut B2	**Referentietabel C**	Attribuut C1	Attribuut C2
1	A	B	C		**P**	**Q**		**R**	**S**
2	D	E	F		**T**	**U**		**V**	**W**
3	G	H	I		**X**	**Y**		**Z**	**I**
4	J	K	L		**2**	**3**		**4**	**5**
5	M	N	O		**6**	**7**		**8**	**9**

Afbeelding 6: Tabelinhoud na Verrijking.

Selecteren: kandidaten uitsluiten

Het kan zijn dat niet alle aangeboden kandidaten relevant zijn voor verwerking in de doelverzameling. Dit kan bijvoorbeeld de volgende redenen hebben:

1. De gehele populatie van de brontabel wordt aangeboden, terwijl alleen de mutaties in een specifieke periode gewenst zijn;
2. De kandidaten zijn inhoudelijk niet interessant voor rapportagedoeleinden.

Door middel van criteria op de attributen wordt aangegeven welke kandidaten relevant zijn en welke niet. Voor de klantdimensie zijn alle kandidaten met een CUSTOMER_NR onder 1000 niet relevant. Daarnaast zijn alleen die kandidaten relevant, die binnen de verwerkingsperiode gemuteerd zijn. Dit wordt bepaald aan de hand van het attribuut TIMESTAMP.
In afbeelding 7 is aangegeven dat kandidaten 2 en 5 niet voldoen

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3
1	**A**	**B**	**C**
2	**D**	**E**	**F**
3	**G**	**H**	**I**
4	**J**	**K**	**L**
5	**M**	**N**	**O**

Afbeelding 5: Tabelinhoud na Ophalen.

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3	**Referentietabel B**	Attribuut B1	Attribuut B2	**Referentietabel C**	Attribuut C1	Attribuut C2
1	A	B	C		P	Q		R	S
2	D	E	F		T	~~U~~		V	W
3	G	H	I		X	Y		Z	I
4	J	K	L		2	3		4	5
5	M	~~N~~	O		6	7		~~8~~	9

Afbeelding 7: Tabelinhoud na Selecteren.

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3	**Referentietabel B**	Attribuut B1	Attribuut B2	**Referentietabel C**	Attribuut C1	Attribuut C2	**Doeltabel D**	FK1	FK2	FK3
1	A	B	C		P	Q		R	S		**a**	**b**	**c**
2	D	E	F		T	U		V	W				
3	G	H	I		X	Y		Z	I		**d**	**e**	
4	J	K	L		2	3		4	5		**g**	**h**	**i**
5	M	N	O		6	7		8	9				

Afbeelding 8: Tabelinhoud na Koppelen.

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3	**Referentietabel B**	Attribuut B1	Attribuut B2	**Referentietabel C**	Attribuut C1	Attribuut C2	**Doeltabel D**	FK1	FK2	FK3	Actie
1	A	B	C		P	Q		R	S		a	b	c	
2	D	E	F		T	U		V	W					
3	**G**	**H**	**I**		**X**	**Y**		**Z**	**I**		**d**	**e**		*herverwerken*
4	**J**	**K**	**L**		**2**	**3**		**4**	**5**		**g**	**h**	**i**	*corrigeren*
5	M	N	O		6	7		8	9					

Afbeelding 9: Tabelinhoud na Valideren. Alleen record 1 blijft over na deze stap.

aan de selectiecriteria. Na deze stap blijven dus drie kandidaten over (nummers 1, 3 en 4). De resultaatverzameling van deze stap bevat alle *relevante* kandidaten met de direct *en indirect* gekoppelde attributen.

Koppelen: toevoegen sleutelattributen van doelmodel

Indien de kandidaat in het doelmodel verwijzingen heeft naar andere entiteiten, dan worden in deze stap de verwijzende sleutelattributen per kandidaat vastgesteld. Deze stap zal voornamelijk gelden voor kandidaat feiten, en in sommige gevallen voor kandidaat dimensie-records.

Deze stap vertoont veel overeenkomsten met de stap Verrijken. Ook hier wordt gespecificeerd volgens welke criteria relaties van de kandidaat met andere entiteittypen worden gelegd. Het verschil met de stap Verrijken is dat hier wordt gewerkt aan de doelgedaante van de kandidaat, dat de Selectie dus al gepasseerd is, en dat we hier alleen geïnteresseerd zijn in de verwijzende sleutel, en niet in de overige attributen van de te koppelen entiteittypen.

In afbeelding 8 is te zien hoe de tabel gestaag groeit, en nu inmiddels al een deel van de doelgedaante heeft aangenomen. De resultaatverzameling van deze stap bevat alle *relevante* kandidaten met de direct *en indirect* gekoppelde attributen, *plus de verwijzende sleutelattributen van het doelmodel*.

Valideren: kandidaten controleren op volledigheid en juistheid

Iedere kandidaat dient aan vastgestelde kwaliteitscriteria te voldoen, om te kunnen worden opgenomen in de doel-gegevensverzameling. Deze kwaliteitscriteria worden vastgelegd in validatieregels. Een validatieregel kan betrekking hebben op een verrijking (is een verrijking geslaagd of niet), een koppeling (is een verwijzing gevonden of niet) of een attribuutwaarde (domeincontrole, verplichte waarde).

Deze stap vertoont overeenkomst met de stap Selecteren. Het verschil is dat bij het Selecteren een kandidaat op basis van criteria afvalt en niet verder behandeld wordt, en dat bij het Valideren iedere kandidaat relevant is, maar (nog) niet goed genoeg kan zijn om direct verder verwerkt te worden. Er zijn dan correctieve acties nodig om de kandidaat alsnog verder te kunnen verwerken. Een mogelijke correctieve actie is het op een later tijdstip opnieuw aanbieden van de kandidaat.

Voor de klantdimensie geldt dat de waarde van PRIVATE_BUSINESS_FLAG binnen het domein (P,Z) moet vallen. Indien een kandidaat hier niet aan voldoet, wordt deze geparkeerd om ter correctie te worden aangeboden. In afbeelding 9 is aangegeven dat kandidaten 3 en 4 niet voldoen aan de kwaliteitscriteria. Per kwaliteitscriterium is een vervolgactie gedefinieerd.

De resultaatverzameling van deze stap bevat: *alle relevante en kwalitatief goedgekeurde* kandidaten met de direct en indirect gekoppelde attributen, plus de verwijzende sleutelattributen van het doelmodel.

Converteren: vaststellen doelattributen, af te leiden van bron-attributen

De doelgedaante van de kandidaten wordt volledig gemaakt door het afleiden van alle attributen die in het bronmodel benodigd zijn (exclusief de verwijzende sleutelattributen, die zijn in de stap Koppelen al vastgesteld). Per doelgedaante-attribuut wordt gespecificeerd op welke wijze deze wordt afgeleid van één of meer brongedaante-attributen.

Trigger-tabel A	Attribuut A1	Attribuut A2	Attribuut A3	**Referentietabel B**	Attribuut B1	Attribuut B2	**Referentietabel C**	Attribuut C1	Attribuut C2	**Doeltabel D**	FK1	FK2	FK3	Attribuut D1	Attribuut D2	Attribuut D3
1	A	B	C		P	Q		R	S		a	b	c	**x**	**y**	**z**
2	D	E	F		T	U		V	W							
3	G	H	I		X	Y		Z	I		d	e				
4	J	K	L		2	3		4	5		g	h	i			
5	M	N	O		6	7		8	9							

Afbeelding 10: Tabelinhoud na Converteren.

Regel	Stap	Criterium	Evt. Actie
1	Ophalen	CUSTOMER	
2	Verrijken	CUSTOMER - BEDRIJVENREGISTER: CUSTOMER.IDENTIFICATION = BEDRIJVENREGISTER.KVKNUMMER -en- CUSTOMER.PRIVATE_BUSINESS_FLAG = 'Z'	
3	Verrijken	CUSTOMER - ADDRESS: CUSTOMER.ADDRESS_NR = ADDRESS.ADDRESS_NR -en- CUSTOMER.TIMESTAMP >= ADDRESS.VALID_FROM -en- kies MAX(ADDRESS.VALID_FROM)	
4	Verrijken	ADDRESS - DEMOGRAFIE: ADDRESS.POSTCODE = DEMOGRAFIEPOSTCODE	
5	Selecteren	CUSTOMER.CUSTOMER_NR >= 1000	
6	Selecteren	CUSTOMER.TIMESTAMP Binnen [VerwerkingsPeriode]	
7	Valideren	CUSTOMER.PRIVATE_BUSINESS_FLAG = 'Z' of 'P'	Parkeren voor correctie
8	Converteren	KLANT_NUMMER = CUSTOMER.CUSTOMER_NR	
9	Converteren	KLANT_NAAM = CUSTOMER.NAME	
10	Converteren	KLANT_TYPE = CUSTOMER.PRIVATE_BUSINESS_FLAG	
11	Converteren	KLANT_IDENTIFICATIE = CUSTOMER.IDENTIFICATION	
12	Converteren	KLANT_KVKNUMMER = BEDRIJVENREGISTER.KVKNUMMER	
13	Converteren	KLANT_POSTCODE = ADDRESS.POSTCODE	
14	Converteren	KLANT_DEMOGRAFIE = DEMOGRAFIE.DEMOGRAFIE_TYPE	
15	Converteren	KLANT_BRANCHE = BEDRIJVENREGISTER.BRANCHE_CODE	

Figuur 11: Transformatiemodel Klant-dimensie.

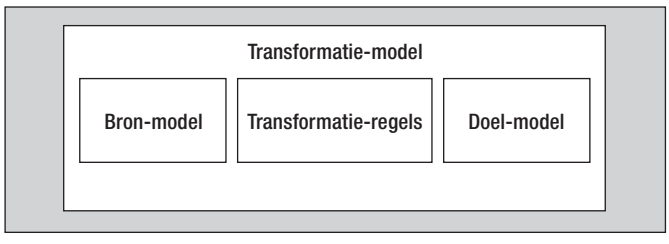
De resultaatverzameling van deze stap bevat alle *relevante en kwalitatief goedgekeurde* kandidaten *met alle benodigde attributen voor het doelmodel*.
Zoals beschreven, bij het uiteindelijk laden van de kandidaten in de doeltabel worden zaken als historie-afhandeling (slowly changing dimensions) meegenomen. Ook deze historie-afhandeling kan eenvoudig met regels gespecificeerd worden.

Transformatieregels

De transformatieregels, zoals deze per stap en aan de datamodellen van bron en doel zijn gerelateerd, kunnen overzichtelijk in tabelvorm worden weergegeven. In afbeelding 11 zijn de transformatieregels voor de Klant-dimensie opgenomen.
Samen met de datamodellen van zowel de bron-gegevensverzameling als de doel-gegevensverzameling, vormt deze tabel de volledige functionele beschrijving van het transformatieproces. Hiermee is een 'Platform Onafhankelijk Model' gecreëerd.
Deze modelgedreven aanpak biedt een aantal voordelen.

- Deze aanpak is, mits voldoende uitgekristalliseerd, te automatiseren. Er bestaan al geruime tijd case-tools waarmee vol- of semi-automatisch gegevensstructuren en de meer klassieke gegevensverwerkende programma's gegenereerd worden. Ook voor ETL-programma's is een dergelijke ontwikkeling mogelijk. Hiervoor dient echter nog meer onderzoek verricht te worden;
- Daarnaast voldoet deze modelgedreven aanpak (ook ongeautomatiseerd) in hogere mate aan de kwaliteitseisen zoals deze in dit artikel zijn benoemd. Met name op de aspecten gestructureerdheid, efficiëntie, consistentie, overdraagbaarheid en testbaarheid scoort deze aanpak hoog;
- Er is ook een financiële kant: uiteindelijk levert het wat op. Aspecten als volledigheid en juistheid kunnen met dit model veel beter worden bewaakt en besparen werk.

Het vertalen van dit functionele transformatie-ontwerp naar een technisch ontwerp is een onderwerp voor een volgend artikel.



Afbeelding 12: Transformatiemodel.

Mark Zwijsen (mark.zwijsen@atosorigin.com) is Senior Consultant Datawarehousing bij Atos Origin.