Deel één van een drieluik over het genereren van ETL

ETL-Generator

Richard Puijk en Vincent Wylenzek

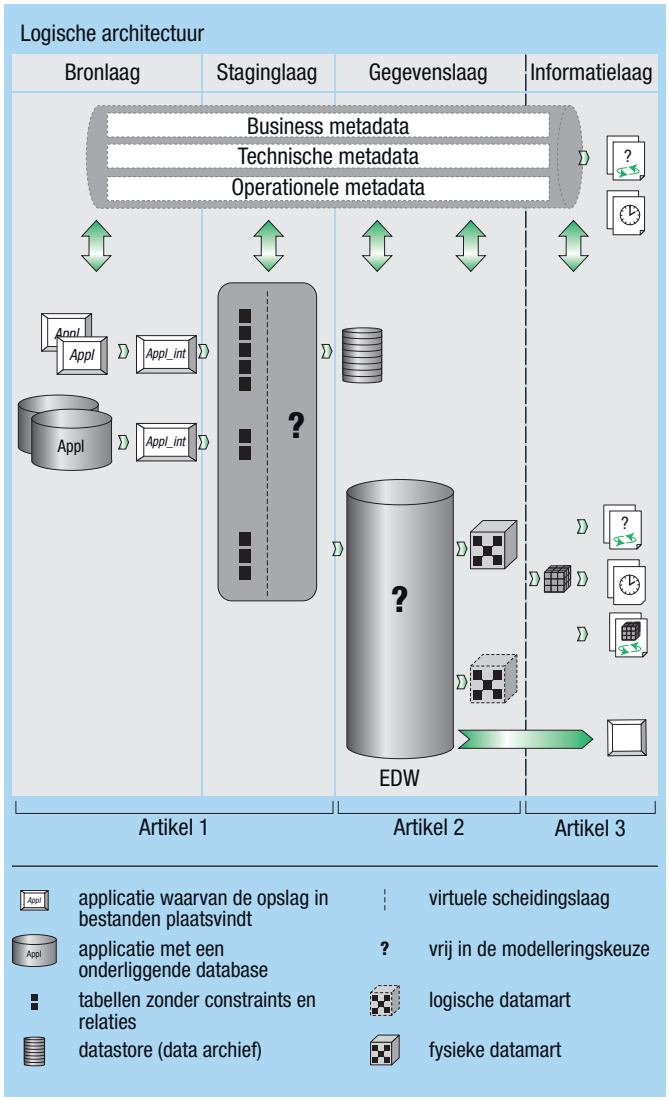
ETL-tools worden steeds geavanceerder, de processen steeds generieker en de behoefte om zo efficiënt mogelijk te werken steeds groter. Genereren van ETL is een trend en inmiddels zijn er al diverse leveranciers die hun generators op de markt uitgebracht hebben. Bent u al toe aan het ontwikkelen of kopen van een ETL-generator?

We kunnen in de huidige markt, met betrekking tot het efficiënter ontwikkelen van ETL, een tweetal verschillende producten onderscheiden, waarbij het ene product puur een versneller is en het andere de ETL bijna volledig automatiseert. De eerste categorie kenmerken we als de ETL-versneller en de tweede als de ETL-generator. Een voorbeeld van een ETL-versneller is een template ETL-mapping of een standaard herbruikbaar patroon binnen een dergelijke mapping. Een ETL-generator genereert de ETL-procedures op basis van metadata.
In dit artikel ligt de focus op de ETL-generatoren, omdat ETL-versnellers inmiddels gemeengoed zijn. Inmiddels zijn er zo'n vijftien leveranciers van ETL-generators, waarvan het overgrote deel op Microsoft technologie is gebaseerd. De oplossingen liggen op technologisch niveau wel redelijk ver uit elkaar; zo kiest de ene partij voor metadata-driven ETL in combinatie met tool-specifieke SSIS-templates met variabelen en kiest de ander ervoor om generieke ANSI-SQL ('92) code te gebruiken en laadprocessen via Stored Procedures aan te roepen. De ene partij heeft een volwassen GUI, waarmee de metadata eenvoudig op te voeren zijn, de ander werkt met platte stuurbestanden of Excel. De scope van de leveranciers met betrekking tot de ETL-generatoren loopt ook wijd uiteen. De ene partij heeft een sterke focus op het genereren van de transformaties van staging naar datawarehouse terwijl de andere partij het hele proces genereert.

Dit artikel is deel één van een drieluik over het genereren van ETL. In dit eerste artikel wordt een schets gegeven van de huidige markt, beschrijven we onze visie en lichten we een deel van de realisatie van onze ETL-generator toe. Tot slot komen ook nog enkele verbeterpunten aan bod. In het tweede en derde deel gaan we verder in op de realisatie.

Visie

In DB/M 4, 2009 schreven Vogt, Walraven en Wylenzek reeds over een 'generiek' ETL-proces waarbij het datawarehouse door middel van een aantal Microsoft SQL Server 2008 SSIS-templates geladen wordt. Elke template bevatte een aantal parameters, waarmee in slechts een beperkt aantal handelingen een nieuwe entiteit aan het datawarehouse kon worden toegevoegd.
Nu, inmiddels twee jaar later is het bovenstaande ETL-proces een aantal keer geïmplementeerd en hebben we zowel de voor- als nadelen van een gestandaardiseerd proces ondervonden. Het grootste nadeel is het gemis aan flexibiliteit. Zo was het ETL-proces destijds gebaseerd op een Enterprise DataWarehouse (EDW) op basis van Data Vault modellering, in de praktijk blijkt een EDW niet altijd noodzakelijk of sluit een relationeel (3NF) of dimensionaal gemodelleerd EDW beter aan op de klantvraag. Daarnaast waren de bronextracties buiten de scope van het ETL-proces gehaald, met de argumentatie dat dit de verantwoordelijkheid was van de broneigenaar. Dit bleek in de praktijk geen goede keus.
Flexibiliteit, of eigenlijk het gemis hiervan, was het grootste leerpunt van dit ETL-proces. De flexibiliteit was enerzijds beperkt, omdat de bestaande templates niet eenvoudig aan te passen waren. Wijzigingen in de templates werden namelijk niet direct doorgevoerd in de reeds ontwikkelde packages. Anderzijds moest er een betere afweging worden gemaakt in de mate van standaardisatie. Standaardisatie heeft een raakvlak met kwaliteit en efficiency. Impliceert volledige standaardisatie van ETL dan altijd een hogere kwaliteit en efficiency? Nee, volledige standaardisatie creëert starheid, schijnoptimalisatie en zorgt er in het ergste geval voor, dat er niet aan de requirements voldaan kan worden. Immers, alles wat voorafgaand aan de realisatie bekend is, kan meegenomen worden in de mate van flexibiliteit, maar omdat we vandaag niet weten hoe de wereld er in de toekomst



Afbeelding 1: Logische architectuur referentieproject.

uitziet, kunnen we niet alles flexibel meenemen in het standaard ETL-proces. Denk bijvoorbeeld aan wijzigende wet- en regelgeving of veranderende technologie.

Er moet dus een afweging worden gemaakt welke componenten generiek opgezet worden en wat specifiek blijft. Onderstaande vragen kunnen helpen in deze afweging. Indien de vragen positief beantwoord worden, dan leent de mapping zich voor generalisatie.

1. Wordt de logica uit de mapping meerdere keren toegepast? (vuistregel: minimaal vijf keer);
2. Is de structuur van deze mapping altijd gelijk en zit de afwijking alleen in de bron-, (eventueel hulp-) en doeltabellen?;
3. Wegen de kosten van het generaliseren op tegen de korte termijn baten die men ermee verkrijgt? (denk in termen van ontwikkel- en testtijd);
4. Wegen de kosten van het generaliseren op tegen de lange termijn baten die men ermee verkrijgt? (denk in termen van beheer- en beheersbaarheid, impact van wijzigingen et cetera).

Het voordeel van ETL-generatoren vanuit opdrachtgeverperspectief is enigszins beperkt. Opdrachtgevers zijn namelijk niet primair geïnteresseerd in de technische uitwerking, maar juist in de voor de business toegevoegde waarde. De relatief hoge ex ante investering voor het kopen of ontwikkelen van een generator wordt vaak één-op-één afgezet tegen de directe baten die men er mee realiseert. De hogere baten op lange termijn worden vaak beperkt meegenomen.

Dienstverlenerperspectief

Wanneer niet puur vanuit een project wordt geredeneerd, maar breder, zoals bijvoorbeeld vanuit een dienstverlenerperspectief, zijn aanvullende criteria relevant. Zoals uit de beschrijving van de huidige markt al duidelijk is geworden, zijn er inmiddels al vele leveranciers van ETL-generatoren. Een ETL-generator maakt het mogelijk om efficiënter te werken, waardoor meer resultaat behaald kan worden in een zelfde tijdsbestek. Dit heeft invloed op de kostprijs en daarmee mogelijk op de concurrentiepositie van de dienstverlener.

Vanuit het dienstverlenerperspectief moeten de volgende keuzes gemaakt worden; Wordt het als een zelfstandig product verkocht of is het een 'hulpmiddel' voor consultants, zodat zij hun werk efficiënter en beter kunnen doen? Wordt de generalisatieslag toolspecifiek of tooloverstijgend, lees SQL (bij voorkeur ANSI-92)? De beste oplossing voor een bepaald ETL-probleem in de ene tool kan sterk afwijken van de beste uitwerking in een andere tool. Met name de onderliggende metadata repository van de tools wijkt nogal af. De volgende keuze is dan ook logischerwijs of men een eigen metadata- en logging repository opzet, of juist die van de specifieke tool gebruikt. Tabel 1 geeft een overzicht van verschillen tussen de keuze SQL en toolspecifiek. Indien gekozen wordt voor toolspecifiek, dan zijn er diverse dialecten, zie tabel 2.

Keuze

Het antwoord op bovenstaande vragen vanuit onze visie als dienstverlener heeft ons geholpen om de keuze te maken met betrekking tot de realisatie van de generator.

Generiek versus specifiek – Hiervoor beantwoorden we vraag één en twee per mapping.

Kosten versus baten – Hiervoor beantwoorden we vraag drie en vier op basis van een referentieproject.

Doelgroep – De ETL-generator dient bij ons als hulpmiddel om efficiënter te kunnen werken tegen een betere kwaliteit. We zetten deze tool dan ook puur in om de BI-consultants te ondersteunen en verkopen het niet als opzichzelfstaand product.

Toolspecifiek/tooloverstijgend – Wij leveren dienstverlening voor meerdere tools. We hebben er toch voor gekozen om van de toolspecifieke mappings gebruik te maken en niet van een generieke ANSI-SQL opzet. De diverse dialecten en onderliggende softwarearchitectuur wijken zo ver van elkaar af, dat het niet mogelijk is om één uniforme code te ontwikkelen. We hebben wel een aantal onderdelen uniform opgezet. Zo is er een applicatiedatabase, waarin de verschillende instellingen voor de generator worden opgeslagen.

Eigenschap	ANSI-SQL	Toolspecifieke mapping
Hoeveelheid code	Eén dialect en één generiek script per mapping.	Elke tool heeft eigen code in eigen dialect.
Transparantie	Afhankelijk van uit welk paradigma benaderd; SQL is eenvoudig leesbaar, maar niet visueel.	Grafische weergave (geeft meer inzicht).
Afhankelijkheid (= continuïteit)	Alleen afhankelijk van SQL.	Afhankelijk van de tool en vaak ook nog een specifieke versie van een tool.
Efficiency	Veel code benodigd voor relatief eenvoudige bewerkingen.	De meest voorkomende transformaties zijn vaak als eenvoudig te parametriseren componenten opgenomen.
Performance	SQL is met name sneller met database georiënteerde acties (sorts, aggregates, joins).	Snel op in memory georiënteerde acties (non-blocking), traag met blocking acties.

Tabel 1: Verschillen ANSI-SQL of Toolspecifiek.

*Eigen metadata of toolspecifie*k – We kiezen voor een eigen metadata repository in verband met de ver van elkaar afwijkende onderliggende repository's van de diverse tools. Eén generieke repository helpt in het kader van transparantie en het raadplegen van de opgeslagen informatie.

Afbakening – We zijn gestart met de technische uitwerking op basis van Microsoft SQL Server 2008 R2, omdat hier nu concrete vraag naar is. In eventuele volgende incrementen wordt de generator ook voor andere tools uitgewerkt. Vanwege ons klantenbestand en expertise staan Oracle Warehouse Builder (OWB) en Informatica PowerCenter prominent op de lijst. Daarnaast hebben we parallel aan dit traject al bij diverse klanten ETL kunnen genereren op basis van OWB. De realisatie wordt uitgebreid besproken in de volgende paragraaf.

De realisatie

Zoals hiervoor reeds is beschreven, is de eerste versie van de ETL-generator op Microsoft technologie gebaseerd. De generator is ontwikkeld op basis van een project waarbij het standaard ETL-proces, zoals beschreven in DB/M 4, 2009, geïmplementeerd was en reeds circa 80 interfaces waren aangesloten.

Architectuur. De in afbeelding 1 getoonde logische architectuur (zie 'Generiek A&I-raamwerk bewijst zijn nut' DB/M 5, 2007) dient als basis voor de uitwerking van onze ETL-generator. Zie voor de toelichting van de symbolen DB/M 5, 2007. Zie de uitgangspunten en de toelichting bij de diverse ETL-stappen over hoe de diverse lagen worden gevuld en/of bijgewerkt, tabel 3.

De generator. De generator bestaat uit een grafische interface en een onderliggende applicatiedatabase. De instellingen van de generator, maar ook de mappingdefinities en configuraties (zoals bijvoorbeeld de instellingen voor de OTAP-straat) worden opgeslagen in deze onderliggende database.

De generator werkt in de basis als volgt; de GUI bevat de instellingen en parameters, in SSIS wordt de ETL-template ontwikkeld. Deze template wordt met behulp van de generator in de onderliggende applicatiedatabase ingeladen. Vervolgens wordt in de code aangegeven welke componenten/property's generiek zijn en aangepast moeten worden op basis van de ingestelde parameters. De relatie tussen parameters en SSIS-objecten ligt dus vast in de code. Bij het genereren van de packages wordt op basis van de template een nieuwe package aangemaakt, waarbij de ingevoerde parameters gemapped worden aan de juiste property's in de package.

Bronlaag – genereren interfaces. Het is niet altijd duidelijk wie de broneigenaar is en/of de bron is niet in staat om zelf een interface aan te bieden (push). Op basis van de ingevoerde gegevensleveringovereenkomst (GLO), kan de generator ook packages genereren voor de extractie vanuit het bronsysteem. De GLO kan met behulp van de GUI worden ingevoerd en wordt centraal in de repository opgeslagen (zie afbeelding 2). De gegenereerde extractiepackages worden ook onafhankelijk van het datawarehouse gescheduled, dit om aan uitgangspunt B.3 te kunnen voldoen.

Naar aanleiding van uitgangspunten B.1 en B.2, verwachten we dat een tweetal CSV-bestanden per interface gecreëerd wordt. Eén bevat de data en de ander de metadata. De data betreffen de headerregel (kolommen) en gegevensregels. De metadata bevatten de extractiedatum, het uitgevoerde SQL-statement (indien van toepassing), de naam van de bron en brontabel, het aantal rijen en eventueel nog enkele additionele attributen.

Met de GUI is het mogelijk om een aantal configuraties in te voeren. De belangrijkste worden nu besproken. Als eerste kunnen diverse bronsysteemtypes gekozen worden. Momenteel worden enerzijds flatfiles en anderzijds OLE_DB connecties ondersteund. Met deze laatste connectie zijn Oracle, .XLS en SQL Server als bronnen mogelijk. Ten tweede kunnen SQL-extractiequery's worden ingevoerd. Deze query's bestaan initieel vaak uit een selectie op alle kolommen van de brontabel. Kleine wijzigingen in de bron kunnen door middel van een wijziging op deze query eenvoudig worden opgelost. Ten derde kan het bron-, fout- en verwerkingspad opgegeven worden. Ten slotte worden de ingestelde waarden meegenomen bij het genereren van de package(s).

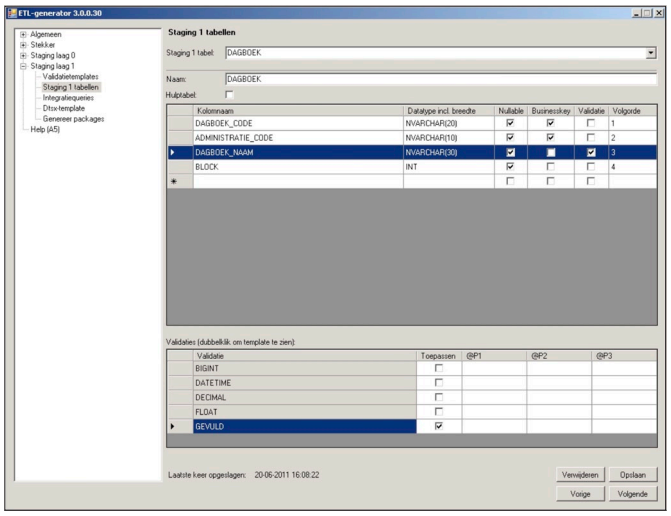
Tool	Dialect
Microsoft SQL Server SSIS (2005, 2008, 2008 R2)	DTS,VB.net of C# (Microsoft.SqlServer.Dts)
Oracle Warehouse Builder	Tcl (OMB*Plus)
Informatica PowerCenter	XML, (VB.net, C#,Vba)

Tabel 2: Dialecten.

Nr	Laag	Omschrijving
B.1	Bronlaag	Alle bronnen leveren platte bestanden aan (conform GLO).
B.2	Bronlaag	Alle bronnen leveren metadata aan (conform GLO).
B.3	Bronlaag	De bronlaag is zo ver mogelijk ontkoppeld van het datawarehouse.
S.1	Staginglaag	Alle aangeleverde data worden 1:1 ingelezen.
S.2	Staginglaag	Alle data moeten kunnen worden gevalideerd.

Tabel 3: Uitgangspunten bronlaag en staginglaag.

Staginglaag – genereren staging packages. De staging packages voldoen aan uitgangspunten S.1 en S.2. Om te waarborgen dat alle data één-op-één worden ingelezen (S.1) zijn alle attributen in de staging area als datatype nvarchar(2000) gedefinieerd. Wanneer 2000 tekens onvoldoende is, kan dit in de configuratie worden aangepast. De stagingtabellen worden aangemaakt aan de hand van de definitie in de centraal opgeslagen GLO. Alleen de kolomvolgordelijkheid en naamgeving zijn hierbij relevant, het datatype is immers altijd nvarchar(2000).
De gegenereerde staging package wordt gebaseerd op een specifieke staging template ETL-package. In deze package wordt een aantal generieke stappen uitgevoerd. De eerste stap is de controle of de metadata- en databestanden aanwezig zijn, de tweede stap betreft de controle of het aantal records overeenkomt met de metadata, in de derde stap worden de bestanden verplaatst naar het verwerkingspad, in de vierde stap wordt de doeltabel leeggehaald en tenslotte wordt de doeltabel gevuld door middel van bulk-load.
Validatie. Optioneel kan in de generator aangegeven worden of de in te lezen brondata gevalideerd moeten worden. Alhoewel het in verband met de betrouwbaarheid van de informatie altijd van belang is om de data te controleren, behoort datakwaliteit in verband met de vereiste investeringen niet altijd tot de scope van een project. Door de standaard werkwijze en het genereren is het altijd de moeite waard om validaties te doen op de aangeleverde data. Immers, als de ontwikkeltijd van een validatie beperkt blijft tot het aanvinken van de betreffende controle op een kolom, dan zullen de baten van de toegenomen datakwaliteit altijd groter zijn dan de extra kosten aan ontwikkeltijd.
De generator bevat standaard een aantal controles. Standaard zijn er diverse datatypegerichte controles aanwezig, zoals bigint, datetime, decimal en float. Daarnaast is er een not null en empty controle aanwezig.
De controles zijn zogenaamde SQL Server User-Defined Functies (UDF) gebaseerd op CLR. Deze zijn geschreven in C# en kunnen worden aangeroepen vanuit SQL. Wanneer een controle voor een bepaalde kolom staat aangevinkt (zie afbeelding 2), zal de generator dit verwerken in de gegenereerde package. Het is uiteraard mogelijk om extra (bijvoorbeeld klantspecifieke) controles toe te voegen. De performance van deze validaties is ook bij grote datasets (meer dan 1 miljard rijen) goed.



Afbeelding 2: Invoer GLO.

Verbeterpunten

Afhankelijkheid code – mapping. De afhankelijkheid tussen code en templates is nu te groot. Wanneer een template package wijzigt, moet tevens de code worden aangepast. Een vertaaltool (SSIS formaat naar C#) kan hier hulp bij bieden, helaas zijn de huidige tools op de markt nog te prematuur om in te zetten.
Afhankelijkheid generator – OTAP-straat. Op dit moment worden packages specifiek voor een bepaalde OTAP-omgeving gegenereerd. Ook worden de tabellen direct op deze omgeving aangemaakt. In de praktijk is het beter om packages te genereren die op basis van configuraties en de datawarehouse metadata op een bepaalde omgeving draaien. Zo kunnen de packages de OTAP-straat ook doorlopen. Om het proces van het promoten van deze packages te automatiseren kan gebruik gemaakt worden van de Business Intelligence Installer. (Wylenzek, Nijenhuis, Puijk in DB/M 5, 2010).
Invoeren configuratie. De GLO's inclusief de volledige attribuutdefinities, moeten nu handmatig worden ingevoerd. In de praktijk impliceert dit vele manuele handelingen. Een export vanuit een datadictionary van een bronsysteem biedt hiervoor wellicht een betere oplossing. Om deze export in te kunnen lezen, moet nog een importfunctie worden opgenomen.
Genereren GLO-document. Op basis van de ingevoerde GLO's is het vrij eenvoudig mogelijk om ook een document (PDF) te genereren, waarin de overeengekomen afspraken over de levering van de gegevens staan beschreven.

Wordt vervolgd

In het volgende artikel wordt ingegaan op het genereren van ETL van staging naar datawarehouse, het archiveren van data in de datastore en het loggen hiervan.

Richard Puijk is Medior Business Intelligence Consultant bij Ordina.
Vincent Wylenzek is Senior Business Intelligence Consultant bij Ordina.