

Staging in het data warehouse

Grundsätzlich IT

M: 06-41966000

W: www.grundsatzlich-it.nl

E: info@grundsatzlich-it.nl



R. Kunenborg

Versie 2.1 / 3 augustus 2009 / Final

Licentie: deze versie mag vrij worden verspreid en gepubliceerd, onder voorwaarde dat er geen enkele wijziging in wordt aangebracht en bij de publicatie wordt verwezen naar de website van Grundsätzlich IT.

Contents

Staging in het data warehouse.....	3
Inleiding.....	3
Achtergrond.....	4
De principes.....	5
De inrichting.....	6
A. Controleerbaar – Staging moet ongeacht het formaat waarin de data wordt aangeleverd, zoveel metadata opslaan dat we de ingelezen data kunnen herleiden tot de aanlevering (of kunnen zien waar er iets niet is ingelezen).....	6
B. Onafhankelijke lagen - Staging moet een wijziging in de aanleveringen aankunnen zonder aanpassingen buiten het staginggebied.....	8
C. Onafhankelijke aanleveringen - Afhankelijkheden tussen aanleveringen moeten worden geminimaliseerd en waar ze aanwezig zijn mogen deze de verwerking niet verstoren.....	9
D. Herhaalbaar – Alle informatie die verwerkt moet worden wordt tijdig, in de goede volgorde en eenmalig verwerkt.....	11
E. Compleet – Alle aangeleverde data wordt ingelezen, tenzij het fysiek onmogelijk is.....	12
F. Beheersbaar – Staging levert de gegevens waarmee we de performance van de dataleverancier bewaken.....	13
Staging en realtime aanleveringen.....	14
Speciale gevallen.....	14
Conclusie.....	15
Voorbeeld.....	16
Brontabellen.....	16
Tabel: Customer.....	16
Tabel: STG_Customer.....	16
Ondersteunende tabellen.....	17
Tabel: Staging Delivery Audit.....	17
Tabel: Staging Delivery Field Audit.....	18
Tabel: Staging Delivery Line Audit.....	19
Tabel: Staging Delivery Contracts.....	19
Tabel: Staging Delivery Groups.....	19
Tabel: Staging Delivery Grouplink.....	19

Staging in het data warehouse

Inleiding

In dit artikel beschrijf ik wat een staginglaag is en waarom deze belangrijk is. Vervolgens ga ik verder en probeer op basis van een aantal principes een generieke staginglaag uit te werken, zo veel als mogelijk onafhankelijk van de verdere inrichting van het data warehouse.

Wat betreft voorgaande artikelen is er in DB/M de laatste jaren twee keer over staging geschreven. De eerste keer in 2007 in een artikel waar grote problemen in zitten¹, de tweede keer tijdens het schrijven van dit artikel in DB/M 4-2009². In het laatste artikel werd de benadering die ik in dit artikel kies ook genomen, zij het vooral gericht op de ETL. Buiten DB/M is er ook wel over gepubliceerd, maar bijna altijd wordt in navolging van Ralph Kimball de hele extractie- en laadprogrammatuur van staging naar data warehouse op één hoop geveegd^{3 4}.

Wat er tot nu toe is beschreven is naar mijn mening dan ook niet voldoende voor wie vanaf het nulpunt iets gaat implementeren. Daarom is dit artikel geschreven in de vorm van een gids, waarmee een data warehouse architect een deel van de architectuur 'out of the box' neer kan zetten. Het artikel is het beste te begrijpen voor mensen die al eerder zelfstandig een staginglaag hebben gebouwd of daarmee te maken hebben gehad.

Bij het schrijven van dit document ben ik geholpen door leden van BI United (www.bi-united.nl), in het bijzonder Rob Vonk en Bas Stiekema, evenals Patrick Leurs van de ING. Aan alle betrokkenen mijn dank voor de waardevolle input.

¹ "Mag het een (opslag)laagje meer zijn?", DB/M 3, mei 2007, Frank Habers

² "ETL in een paar minuten", DB/M 4, juni 2009, Matthijs Vogt, Onno Walraven en Vincent Wylenzek

³ "Mastering Data Extraction", DBMS Online, juni 1996, R. Kimball, <http://www.dbmsmag.com/9606d05.html>

⁴ "The Data Warehouse ETL Toolkit", R. Kimball, September 2004, ISBN: 978-0-7645-6757-5.

Achtergrond

Bij een data warehouse architectuur hoort een gebied, waar data wordt opgeslagen voordat ze in het echte data warehouse wordt geladen. Dit gebied, “staging”, vervult de functie van een buffer, waar technische problemen met dataleveringen worden opgelost. Het oplossen van de problemen in het staginggebied zorgt er voor dat de rest van het data warehouse met de aangeleverde gegevens kan werken zonder zich zorgen te maken over hoe die gegevens zijn aangeleverd en verwerkt.

De infrastructuur van een data warehouse is verantwoordelijk voor zo'n 70% van de kosten, waarvan een groot deel opgaat aan de bouw van programmatuur voor extractie, transformatie en laden (ETL) van gegevens⁵. Een goede staginglaag bepaalt in grote mate hoe het eindresultaat er uit gaat zien en het is dus belangrijk om dat goed in te richten en te beschrijven. In dit artikel beschrijf ik de staginglaag aan de hand van architectuurprincipes, die vervolgens worden uitgewerkt.

Allereerst dient een architectuur op basis van het principe van ‘loose coupling’ de lagen in het data warehouse te scheiden en zoveel mogelijk kennis binnen de laag te houden die zich daar mee bezig houdt. Voor het opschonen en klaarzetten van gegevens is veel kennis nodig van gegevens binnen het data warehouse. Dit zorgt er voor dat het schonen, als je dat al zou willen, een taak is voor de (ETL-)interface tussen staging en de data warehouselaag waar de kennis zit over beide gebieden.

Verder is het zo dat tegenwoordig de opvatting over ‘foute’ data aan het veranderen is. In de moderne opvatting zoals verwoord door o.a. Dan Linstedt bestaat er geen ‘foute’ data. Er bestaat alleen data die technisch niet klopt en waar staging iets mee moet doen, en data die niet voldoet aan business rules. Die laatste data kun je voorzien van indicatoren, en daarmee kan je zowel afwijkingsrapporten per bronsysteem maken, als kijken of de business rules misschien fout zijn. Deze data moet echter wel door naar het data warehouse. Dit kun je samenvatten in de woorden van Dan Linstedt als “100% van de data, 100% van de tijd”. De enige bewerking die is toegestaan is het zonder gegevensverlies gelijktrekken van veldlengtes en gegevenstypes.

Een voorbeeld van waar (te) rigoureuus opschonen en wegfilteren van gegevens toe kan leiden is het late ontdekken van het gat in de ozonlaag: in metingen was het gat al jaren zichtbaar, maar de grote gaten werden uit de eindresultaten gefilterd omdat ze afweken van wat er werd verwacht⁶. Pas toen andere metingen afweken kwamen de problemen aan het licht.

Voor we verder gaan, moeten we eerst even terug naar de basis. Wat is een data warehouse? De meest gebruikte definitie van Bill Inmon zegt “Een data warehouse is een onderwerpgeoriënteerde, geïntegreerde, statische en tijdgebonden verzameling gegevens ter ondersteuning van beslissingen op managementniveau.”

De voorgaande definitie laat zien dat er informatie in het data warehouse moet worden gestopt die tijdgebonden is, en geïntegreerd. Dat betekent onder andere dat de volgorde van het inladen van gegevens uit meer dan één bronsysteem belangrijk is. Als we persoonsgegevens inladen, dan willen we natuurlijk wel weten of adres A gevolgd wordt door adres B, of dat het juist andersom is. Wat betreft integratie, het is belangrijk dat als we bijvoorbeeld twee persoonsnummers hebben die fysiek dezelfde persoon aanduiden, dit ook tot uitdrukking komt in het data warehouse.

⁵ ETL Survey 2005, <http://www.passionned.nl/etlonderzoek.htm>

⁶ <http://nl.wikipedia.org/wiki/Ozongat>

Voor de ondersteuning van beslissingen op managementniveau geldt dat dit een belangrijke drijfveer is van business intelligence. In het kader van SOX en andere regelingen moeten we zekerstellen dat het management 'in control' is. Dat wil zeggen: het management dient te beschikken over de juiste cijfers, op de juiste tijd. En het moet ook verantwoord kunnen worden hoe je aan die cijfers komt. Dat is bijvoorbeeld ook een vereiste voor wie mee wil doen met een "Tax Control Framework", waarbij de belastingdienst niet langer alle cijfers wil weten, maar wil weten of je 'in control' bent⁷. 'In control' zijn betekent dat we door het hele data warehouse heen, aan moeten kunnen tonen hoe de cijfers tot stand zijn gekomen in relatie met het bronsysteem. Dat begint dus al in de staging laag.

Ten slotte moet het data warehouse proces zelf ook onderhevig zijn aan sturing en controle, dus moeten we ook 'in control' zijn op gebied van de levering en verwerking van de gegevens. We moeten dus weten in welke volgorde de aanleveringen moeten worden verwerkt en van welke andere aanleveringen deze aanleveringen afhankelijk zijn. Het 'in control' zijn moeten we ook aan kunnen tonen, dus een rapportage op de frequentie, volledigheid en kwaliteit van de aanlevering is noodzakelijk.

Bovenstaande zijn eisen die vanuit het oogpunt van bedrijfsbelang aan een data warehouse worden gesteld. Verder zijn er natuurlijk nog technische eisen die voortkomen uit de omgeving waarin een data warehouse opereert en te maken hebben met het inperken van de benodigde inspanningen voor bouw en onderhoud van het data warehouse. Er is namelijk één zekerheid voor elk data warehouse, en dat is dat de omgeving waarin het opereert aan verandering onderhevig is. Goed om kunnen gaan met verandering is dan ook een primaire eis voor elk data warehouse. Hiervoor grijp ik terug op het al eerder genoemde architectuurprincipe van 'loose coupling'.

De principes

De tot nu toe beschreven uitgangspunten leiden naar de volgende principes waaraan een staginggebied moet voldoen:

- A. Controleerbaar – Staging moet ongeacht het formaat waarin de data wordt aangeleverd, zoveel metadata opslaan dat we de ingelezen data kunnen herleiden tot de aanlevering (of kunnen zien waar er iets niet is ingelezen);
- B. Verticaal onafhankelijk – Staging moet een wijziging in de aanleveringen aankunnen zonder aanpassingen buiten het staginggebied;
- C. Horizontaal onafhankelijk – Afhankelijkheden tussen aanleveringen moeten worden geminimaliseerd en waar ze aanwezig zijn mogen deze de verwerking niet verstoren;
- D. Herhaalbaar – Alle informatie die verwerkt moet worden wordt tijdig, in de goede volgorde en eenmalig verwerkt;
- E. Compleet – Alle aangeleverde data wordt ingelezen, tenzij het fysiek onmogelijk is;
- F. Beheersbaar – Staging levert de gegevens waarmee we de performance van de dataleverancier bewaken.

Ten slotte nog een voor de hand liggende eis die eigenlijk voor elk systeem geldt en die ik verder niet zal behandelen:

- G. Integer – De cijfers worden beveiligd tegen ongeautoriseerde inkijk en wijziging.

⁷ Tax Control Framework brochure, <http://www.belastingdienst.nl/download/1884.html>

Functioneel beheerders van een aanlevering moeten bij “hun” staginggedeelte kunnen om de levering te beoordelen, maar het is uitdrukkelijk niet de bedoeling dat eindgebruikers toegang hebben tot staging, op welke manier dan ook. Het is de werkplaats van het data warehouse, met data in onbekende staat. Net als op een gewone werkplaats willen we in onze data-werkplaats ongeautoriseerde personen geen toegang verlenen – dat leidt maar tot data-ongelukken of data-diefstal. De data-klant is van harte welkom in de data-showroom, waar de gevaarlijke data buiten bereik is gezet, dure data achter slot en grendel zit en alles voorzien is van tekst en uitleg.

De inrichting

Hieronder beschrijf ik per principe, de gevolgen van dat principe voor de inrichting van het staginggebied.

A. Controleerbaar – Staging moet ongeacht het formaat waarin de data wordt aangeleverd, zoveel metadata opslaan dat we de ingelezen data kunnen herleiden tot de aanlevering (of kunnen zien waar er iets niet is ingelezen)

De eis om de ingelezen data te kunnen herleiden tot de aanlevering komt voort uit de wens om bij problemen eenduidig te kunnen communiceren met de aanleverende partij over waar de fout zit. We kunnen er op verschillende manieren voor zorgen dat we alle data kunnen herleiden. In de technisch meest eenvoudige vorm wordt er metadata meegeleverd door de dataleverancier die minimaal de volgende zaken beschrijft:

- Naam en/of code van het bronsysteem;
- De naam en/of code van de aanlevering zoals overeengekomen met de data warehouse ontwikkelaars;
- Een volgnummer;
- Datum en tijdstip van het ontstaan van de aanlevering;
- Het versienummer van de programmatuur waarmee de aanlevering tot stand is gekomen;
- Het aantal data-items (regels, XML-tags, records) in de aanlevering;
- Het dataformaat van de levering, bijvoorbeeld een regel met de geleverde velden, of een XML-schema.

Het doel hiervan is dat we aan de hand van de metadata kunnen herleiden hoe de aanlevering er uitzag, wat vooral belangrijk is als ergens in een aanlevering iets fout gaat. Het is ook nuttig als intern controlemiddel om het aantal geleverde data-items te kunnen vergelijken met het aantal dat uiteindelijk is verwerkt. Verder geeft het meeleveren van het dataformaat van de levering de mogelijkheid om hier ook processing op los te laten en bepaalde velden anders te verwerken op basis van deze informatie. Ten slotte is het een alarmsignaal voor het proces als het dataformaat opeens anders is dan verwacht.

Als dataleveranciers de gevraagde metadata niet aan kunnen leveren, dan zullen we zelf programmatuur moeten ontwikkelen die voorin de staginglaag voor zover mogelijk de genoemde informatie toevoegt. Het moet dan echter wel duidelijk worden gemeld in de audittabellen dat de metadata door onszelf is aangemaakt.

De gevraagde metadata slaan we op in een Delivery Audit tabel, waar we later nog meer informatie aan toevoegen.

Bij elke dataregel slaan we de volgende gegevens op:

- De Delivery Audit tabel key voor deze aanlevering, zodat we gegevens kunnen herleiden naar een aanlevering aan de hand van de key;
- Het nummer van de dataregel, zodat we de volgorde van aanlevering niet verliezen.

Het nummer van de dataregel is belangrijk bij het volgende scenario. Stel we hebben twee dataregels met een adres voor een persoon en beide regels hebben dezelfde mutatedatum, tot op de milliseconde gelijk (dit komt voor bij systemen met veel transacties per seconde), dan kun je vaak nog iets afleiden uit de volgorde van aanlevering. Als die verloren gaat is dat niet langer mogelijk.

Bij parallel inlezen van gegevens kun je zelf vaak geen eenduidige regelnummers toekennen. Daarom is het verstandig deze ook door de dataleverancier te laten meeleveren, anders moet dit voor in het proces worden toegevoegd aan binnenkomende bestanden in een sequentieel proces.

Om te bereiken dat we de ingelezen informatie tot de aanlevering kunnen herleiden voor elk data-item afzonderlijk, voeg ik voor elk veld in de stagingtabel audit-informatie toe. Dit is qua dataopslag geen groot probleem, omdat we er van uit gaan dat idealiter staging slechts tijdelijk gevuld is, tot de aanlevering is verwerkt. Daarom krijgt elk veld een extra veld met Field Audit ID mee. Het Field Audit ID verwijst naar een dimensietabel (Delivery Field Audit) met daarin allerlei gebeurtenissen die voor kunnen komen. Omdat er meerdere gebeurtenissen tegelijk plaats kunnen vinden is een Field Audit code niet genoeg.

In de Delivery Field Audit tabel vinden we sleutels voor onder andere de volgende omstandigheden:

- Data element corrupt of onleesbaar, behandeld als leeg;
- Data element verwacht maar onbekend, behandeld als leeg;
- Data element van het verkeerde type (niet numeriek, geen datum, geen tijd, etc.);
- Data element te groot, afgekapt;
- ... (overige problemen) ...

Deze gevallen kunnen tegelijk voorkomen, vandaar dat we er een dimensie van maken. Dit maakt het ook eenvoudig om later extra indicatoren toe te voegen.

Deze Delivery Field Audit Ids kunnen vervolgens gebruikt worden in het data warehouse om een volledige audit van de verwerking van de aanlevering te maken, van het niveau van de aanlevering zelf tot op het niveau van de individuele velden en de kwaliteit daarvan.

Wat we specifiek *niet* in de veldstatussen bijhouden, zijn fouten veroorzaakt door schending van "business rules". Een schending van de regel dat een waarde tussen 0 en 10 moet liggen, is niet een probleem van staging, of van het data warehouse. Het is een probleem van de business, waar dus in de rapportage en datamart iets mee moet gebeuren.

Zie verder ook Kimball's artikel 'Indicators of Quality'⁸ voor meer informatie over audit-velden en kwaliteitsmeting.

⁸ "Indicators of quality", R. Kimball, <http://www.intelligententerprise.com/000410/webhouse.jhtml>

B. Onafhankelijke lagen - Staging moet een wijziging in de aanleveringen aankunnen zonder aanpassingen buiten het staginggebied

Hoe minder afhankelijkheden tussen lagen in een architectuur bestaan, hoe eenvoudiger en dus goedkoper het is om wijzigingen door te voeren. Dit is een algemeen architectuurprincipe, ook bekend onder de naam 'loose coupling'. Staging is de interface laag tussen de aanleverende systemen en het data warehouse, en zal daarom mee moeten kunnen veranderen met het formaat van de aanlevering, maar heeft idealiter weinig te maken met een nieuwe opzet van de rapportages of met integratie van gegevens. Omgekeerd wil het data warehouse helemaal niet weten of de aanlevering in een tekstbestand zat, of via een ODBC-koppeling uit een andere database werd gehaald. Dat soort technische zaken is voor alle aanleveringen uiteindelijk gelijk en wil je dus op één plek afhandelen, op een consistente manier. Dan kan de ETL van het data warehouse zich concentreren op het afhandelen van business rules.

In DB/M 4-2009 beschrijven enkele consultants van Ordina een aanpak, waarin alle aanleveringen altijd plaatsvinden als platte tekstbestanden⁹. Dit zorgt voor een ETL in de staginglaag die voor alle aanleveringen hetzelfde is, ten koste van enige snelheid door conversies en controles die bij het direct uit een database lezen overbodig zouden zijn. Dat is op zich al genoeg reden om daar voor te kiezen, maar er zijn nog twee extra redenen om te kiezen voor aanlevering in bestanden in plaats van uit de database.

De belangrijkste reden om te kiezen voor aanlevering in tekstbestanden is politiek. De belangrijke vraag bij elke levering van gegevens is namelijk wie de problemen oplost die zich voor kunnen doen bij het verkrijgen van de gegevens. Als de partij die verantwoordelijk is voor het aanleveren van gegevens aan het data warehouse geen controle heeft over de hulpbronnen van dat proces, dan kunnen ze problemen niet zelf oplossen en zijn ze afhankelijk van een derde partij om hun proces draaiende te houden. Bij elke volgende partij neemt de complexiteit van het te managen leverproces toe. Dit zorgt uiteindelijk voor een onbeheersbaar leveringsproces. De partij die de controle heeft over de fysieke database, de toegangsrechten beheert en de tijd kiest wanneer de back-upsoftware de database uitzet, is daarom ook de partij die de verantwoordelijkheid moet krijgen voor het leveren van gegevens uit die database. Elke andere opzet leidt tot grote problemen voor het data warehouse.

De tweede reden om te kiezen voor tekstbestanden is dat het eenvoudiger wordt om bij fouten te herstellen zonder de staginglaag te belasten met ingewikkelde herstelmechanismen. Je wijzigt gewoon het tekstbestand.

Andere redenen om te kiezen voor tekstbestanden hebben te maken met het eenvoudige transport over allerlei verschillende computersystemen, de makkelijke manier om de inhoud te benaderen voor ontwikkelaars en beheerders, de eenvoud van het opnieuw inladen, het niet meer wijzigen van een eenmaal gemaakt bestand, en dergelijke.

In dit document ga ik dan ook uit van aanlevering in tekstbestanden. Aangezien dat ook het meer complexe scenario is, is als men toch een keuze maakt voor directe databaseaanlevering de aanpak simpel: laat eventueel overbodige controles weg.

⁹ "ETL in een paar minuten", DB/M 4, juni 2009, Matthijs Vogt, Onno Walraven en Vincent Wylenzek

Aan de uitgangszijde van de staginglaag zorgen we dat op het einde van het inlezen in staging de gegevens er zo uitzien als ze worden gebruikt in het data warehouse, los van hoe de aanlevering er uit zag. Natuurlijk behouden we daarnaast de oorspronkelijke aanlevering. Dit wordt verder uitgewerkt in het principe dat we aanleveringen compleet verwerken.

C. Onafhankelijke aanleveringen - Afhankelijkheden tussen aanleveringen moeten worden geminimaliseerd en waar ze aanwezig zijn mogen deze de verwerking niet verstoren

Er zijn verschillende soorten afhankelijkheden binnen het data warehouse die impact kunnen hebben op de architectuur.

Ten eerste kunnen er aanleveringen zijn die naar elkaar verwijzen maar los van elkaar worden geleverd. Indien men referentiële integriteit gebruikt in het data warehouse moeten we in dat geval de aanleveringen in de juiste volgorde verwerken. Dit lossen we niet op door staging aan te passen, maar door het data warehouse zo te modelleren dat er geen beperkingen ontstaan op de volgorde waarin we gegevens moeten inladen. Het modelleren van het data warehouse met behulp van bijvoorbeeld de Data Vault methode zorgt er voor dat onderlinge afhankelijkheden geen beperking hoeven te zijn voor het laden van aanleveringen zodra ze binnenkomen¹⁰. Ook zonder Data Vault kan gebruik worden gemaakt van 'plaatsvervangers' die worden gevuld met de echte informatie zodra deze binnenkomt. Voor deze soort afhankelijkheid hoeven we dus niets te doen in de staginglaag.

Afhankelijkheden kunnen ook ontstaan uit rapportage-eisen. Zo is het heel waarschijnlijk dat er geëist wordt dat een rapportage rapporteert op gegevens die in de bronsystemen beschikbaar waren op een bepaald moment, onafhankelijk van de laadtijd. Dat betekent dat voor elk gegeven beschikbaar moet zijn op welk moment het gegeven actueel was in het bronsysteem, op welk moment het uit het bronsysteem werd gehaald en in een aanlevering gestopt en op welk moment het gegeven actueel werd in het data warehouse. Op deze manier kunnen we zowel aanleveringen met terugwerkende kracht verwerken, als zorgen dat we consistente rapportages over een eerdere stand van een bronsysteem kunnen maken terwijl het data warehouse al weer is bijgewerkt.

Ten slotte zijn er nog afhankelijkheden tussen groepen aanleveringen. Deze kunnen op twee manieren ontstaan. Ten eerste kan het zijn dat bij meerdere filialen, ieder filiaal de verkoopgegevens van de vorige dag aanlevert. Dit zijn losse aanleveringen, maar een verkooprapport is pas compleet als elk filiaal de gegevens heeft aangeleverd. Op zijn minst moet je voor je rapporteert weten hoe compleet of incompleet de rapportage is.

Om te kunnen zien of we alles hebben, kunnen we een tabel maken met Staging Delivery Groups waarin we een naam geven aan een Delivery Group, en dan de aanleveringen beschrijven (naam en code zoals overeengekomen met de leverancier) die daar bij horen. Door de combinatie te maken met de Staging Delivery Audit tabel, kunnen we zien of voor de groep alle onverwerkte aanleveringen aanwezig zijn, of dat er nog aanleveringen ontbreken. De Staging Delivery Groups tabel komt er dan als volgt uit te zien:

- AanleveringsgroepID
- Aanleveringsgroep (Naam en code)

¹⁰ "Data Vault Series 4 – Link Tables", Dan Linstedt, 1-1-2004, <http://www.tdan.com/view-articles/5172/>

Een Staging Delivery Grouplink tabel linkt de aanleveringsgroepen aan de aanleveringen.

- AanleveringsgroepID
- Aanlevering (code, naam e.d.)

Bepalen of de groep gereed is voor verwerking doen we door te kijken naar de status van de aanleveringen in de Delivery Audit tabel. Als er van alle aanleveringen in de groep er een op 'gereed voor verwerking' staat, dan is de groep compleet.

In het data warehouse kunnen we vervolgens de aanleveringen verwerken zodra de groep compleet is, of op een bepaald moment (zeg maar de deadline). Op dat moment kan ook bij de auditgegevens worden vastgelegd hoe compleet de groep was en welke delen er wel of niet waren. Daarmee kunnen in de verdere verwerking een aantal beslissingen worden genomen: zo kan worden besloten om ontbrekende gegevens aan te vullen met bijvoorbeeld de gegevens van de dag er voor. Zo'n beslissing wordt dan natuurlijk ook weer bijgehouden in de auditgegevens.

Hoe weten we nu of een groep compleet is? Dat doen we door enkele velden toe te voegen aan de Delivery Audit tabel:

- Vroegste verwachte aanmaakdatum en -tijd van de bron
- Laatste verwachte aanmaakdatum en -tijd van de bron
- Leverstatus (bijvoorbeeld: nog niet geleverd, geleverd, bezig met verwerking, verwerkt, aanleverfout <buiten aanleverfrequentie>, aanleverfout <leesfout>, niet geleverd, etc.)

Vervolgens maken we een extra tabel aan, Staging Delivery Contracts, die er bijvoorbeeld als volgt uitziet:

- Aanlevering (naam en code zoals overeengekomen met de gegevensleverancier)
- Frequentie (dagelijks, elk uur, ad hoc, alleen weekeinde, etc.)
- Verwachte aanmaaktijd van de bron
- Speling in tijd voor de aanmaaktijd
- Speling in tijd na de aanmaaktijd
- Out of bounds afhandeling (afwijzen, accepteren)
- (Eventueel extra gegevens zoals maximaal toelaatbare uitval in regels, velden etc.)

We gaan uit van de aanmaaktijd in de bron, omdat we in die aanleveringen geen gaten willen hebben. Op de fysieke aflevertijd bij het data warehouse sturen we niet. De Frequentie is een tekst die aangeeft met frequentie (welkeweekday, weekend, Monday, Tuesday, etc. of combinaties, kan ook Hourly of Minute zijn) de aanlevering moet worden aangemaakt. Hier mag elke codering worden gebruikt waar de controlerende programmatuur mee overweg kan. De speling is het tijdsgebied waarbinnen we een aanlevering nog als de verwachte aanlevering zien, en verwerken. Let op dat de speling in tijd van aanleveringen er niet voor mag zorgen dat een late aanlevering kan worden verward met een vroege volgende aanlevering.

In de delivery audit tabel vullen we nu de tabel al vooraf met verwachte aanleveringen. Hier kunnen we natuurlijk lang niet alle gegevens invullen, maar op basis van de aanleverfrequentie kunnen we de te verwachten leveringen wel vast inplannen. Dit kun je bijvoorbeeld voor een maand vooraf doen. Bij binnenkomst van een aanlevering die niet voorkomt in de tabel zijn er dan twee mogelijkheden.

Optie één: het is een ad hoc aanlevering die onregelmatig binnenkomt. Dit kan worden gecontroleerd in de Staging Delivery Contracts tabel. Optie twee: het is een aanlevering die niet wordt verwacht. Hier kijken we dan naar de Out of bounds afhandeling om te zien of we deze moeten afwijzen, of alsnog inlezen.

Ook al worden bestandsgroepen meestal gelijktijdig aangemaakt, het kan toch gebeuren dat bestandsgroepen over bijvoorbeeld meerdere dagen komen en dan aan het eind van de week worden verwerkt. Daarom zal in de Staging Delivery Group tabel een verwerkingsfrequentie worden opgenomen. In data warehouses waar dit nooit zal gebeuren kan dat achterwege blijven en zetten we de Staging Delivery Contracts tabel op het niveau van de aanleveringsgroep in plaats van de individuele aanlevering.

Er is één soort van afhankelijke aanlevering die ik nog niet heb besproken. Als een bepaalde aanlevering in verband met de omvang wordt geplitst in losse bestanden, dan is het vaak pas mogelijk dit verder te verwerken als alle gegevens aanwezig zijn. Dit is in de database zelf bijna niet op te lossen omdat je daar te weinig (meta-)informatie voor hebt. Dit moet dus weer aan elkaar worden geplakt voordat het in staging komt, en het ontbreken van deelbestanden betekent daar ook dat de aanlevering niet mag worden afgeleverd aan het staginggebied.

D. Herhaalbaar – Alle informatie die verwerkt moet worden wordt tijdig, in de goede volgorde en eenmalig verwerkt

Een staginggebied zal alle informatie die wordt aangeleverd vasthouden totdat deze kan worden verwerkt. Er is geen vaste regel voor wanneer iets verwerkt moet zijn, dus het kan in uitzonderlijke gevallen dagen duren voor iets wordt verwerkt. Architecten zullen rekening moeten houden met de betrouwbaarheid van dataleveranciers, het geduld van hun eigen management, en dergelijke, bij het plannen van de verwerkingssnelheid. De meeste data warehouses kennen een nachtelijke verwerking van de aangeleverde gegevens maar een directe verwerking na aanlevering is vaak wenselijk. Dan Linstedt zegt “when the data is ready, load it.”¹¹ Al deze gevallen moeten kunnen worden afgehandeld. Belangrijk is om op te merken dat als je geen afhankelijkheden hebt tussen aanleveringen onderling, je kan inladen wanneer je wilt. Het vermijden van afhankelijkheden in de aanlevering is dan ook een belangrijk punt bij het opzetten van de dataleveranties.

De aanleveringen moeten niet alleen direct worden verwerkt, maar ook in de goede volgorde. Als we bijvoorbeeld bij de aanlevering van persoonsgegevens eerst adres B verwerken, en dan adres A, terwijl chronologisch gezien adres A voor B kwam, dan ontstaan er allerlei problemen. Daarom mag alleen de verwachte aanlevering in het data warehouse worden verwerkt. Hoe we daarmee omgaan bespreek ik bij het principe ‘Onafhankelijke lagen’.

Wat betreft de eenmalige verwerking moet er in de metadata worden bijgehouden welke aanlevering in het data warehouse is verwerkt, en deze aanleveringen mogen niet nogmaals worden verwerkt. Hier hebben we echter twee processtappen die beide kunnen worden herstart en beide maar een keer mogen plaatsvinden. Ten eerste het inlezen en technisch klaarzetten voor verwerking in het data warehouse. Ten tweede de verwerking in het data warehouse zelf.

¹¹ “Data Vault Series 5 – Loading practices”, Dan Linstedt, 1-1-2005, <http://www.tdan.com/view-articles/5285>

De implementatie hiervan is op meerdere manieren mogelijk maar een aanvulling op de Delivery Audit tabel is een eenvoudige optie:

- Datum en tijd van de start van het technisch verwerken van de aanlevering;
- Datum en tijd van het eind van het technisch verwerken van de aanlevering;
- Het versienummer van de programmatuur waarmee de aanlevering is verwerkt;
- Het resultaat van de technische verwerking (foutcode, tekst, etc.);
- Het aantal data-items (regels, XML-tags, records) in staging wat correct kon worden verwerkt;
- Het aantal data-items (regels, XML-tags, records) in staging dat niet correct kon worden verwerkt.

Zodra we starten met het verwerken van een aanlevering melden we dat in de Delivery Audit tabel. Als je een verwerking herstart en deze staat al gemerkt als gestart, dan kunnen we eenvoudig de delen van de aanlevering die al verwerkt zijn in de doeltabellen in staging gewoon verwijderen op basis van het Delivery Audit ID, en vervolgens een nieuwe startdatum en tijd schrijven.

De verwerking van de regels in het data warehouse zijn op zich het probleem van de ETL tussen staging en data warehouse. Als het data warehouse is gemodelleerd volgens Data Vault of Anchor modelling technieken doen we uitsluitend inserts, en is een herhaling van een load vrij eenvoudig: verwijder alle items met het juiste Delivery Audit ID uit het data warehouse en start de verwerking. Dit werkt echter niet als we ook updates en deletes uitvoeren. In dat geval dien je na elke succesvolle verwerkte regel die regel te markeren als 'verwerkt'. Na complete verwerking kunnen dan de verwerkte regels worden geschoond. Om dit te faciliteren nemen we in de stagingtabellen een 'regel verwerkt' markering op, naast Delivery Audit ID en regelnummer.

E. Compleet – Alle aangeleverde data wordt ingelezen, tenzij het fysiek onmogelijk is

Na toepassing van de overige principes zijn we zover dat we de aanleveringen netjes in kunnen lezen, op tijd en in de goede volgorde. Echter, bij het inlezen kunnen en zullen fouten optreden. De aanlevering kan bijvoorbeeld gevuld zijn met incorrect doorgegeven data. De data kan in EBCDIC staan, of velden zijn gevuld met binaire codes in plaats van tekst. Niet alleen bestanden zijn kwetsbaar, ook in een ODBC-koppeling kunnen velden in de bron opeens wijzigen van layout en te lang of te kort zijn. Al deze situaties moeten we kunnen behandelen op een gecontroleerde manier.

Waarom doen we zoveel moeite om alle informatie over te nemen, en laten we de ruwe informatie niet gewoon vallen als we deze moeten bewerken? Ten eerste omdat het uitermate lastig is om achteraf te reconstrueren waarom er twee jaar geleden een bepaalde beslissing is genomen als de oorspronkelijke data waarop dat was gebaseerd er niet meer is. Ten tweede omdat in geval van een rechtszaak het bedrijf dat de data niet meer heeft automatisch ongelijk krijgt. Het is dus belangrijk dit goed te regelen.

Bij een levering lezen we de velden in, maar we voorzien de regel in de aanlevering van een Delivery Line Audit ID naar een Delivery Line Audit dimensie, waarmee we aangeven welk probleem of problemen we in de regel hebben gevonden. Verder voegen we voor elke regel een overflow tekstveld toe, waarin 'extra' velden worden opgeslagen. Aan te weinig velden doen we weinig, maar soms voegt een partij opeens extra informatie toe. Daar willen we natuurlijk een waarschuwing op, en we willen weten wat voor informatie dat was. Overigens kan het overflow veld te krap zijn – in dat

geval verlies je informatie. Dit kan bijvoorbeeld gebeuren als de end-of-line tekens verkeerd zijn gecodeerd. Dan krijg je alle velden in de aanlevering van alle rijen na de eerste rij in het overflow veld. Deze gebeurtenis willen we natuurlijk terugzien middels het Delivery Line Audit ID.

In principe laden we de gegevens altijd in, hoe krom ze ook lijken te zijn. Verliezen we echter gegevens of zijn er hele grote problemen (de eerste 100 regels allemaal fout is een goede indicatie), dan kunnen we er voor kiezen de verwerking af te breken. Dit moet wel bij worden gehouden in de Delivery Audit tabel, want dit levert informatie over de performance van een dataleverancier. Het heeft in dat soort gevallen echter weinig zin om voor elke regel apart een foutmelding op te nemen.

De tabel waar we de regels inlezen heeft voor elk veld in de aanlevering drie losse velden. Ten eerste een tekstveld waar het origineel in gaat, ten tweede een veld van het datatype en de omvang zoals we dat in het data warehouse verwachten, en ten derde een Delivery Field Audit ID waar we kunnen aangeven of er problemen zijn met het veld, en zo ja welke dat dan zijn.

Samenvattend heeft een stagingtabel dan ook de volgende velden: een Delivery Audit ID, een Delivery Line Audit ID, een data-itemnummer, een verwerkingsindicator, een overflow-veld en voor elk verwacht data-item, drie velden zoals hierboven beschreven.

Op basis van de soort fouten en de velden waarin de fouten optreden kan de ETL een gefundeerd besluit nemen om de verwerking gewoon te stoppen na het inlezen. In het Delivery Contract kun je aangeven bij hoeveel regels fout of welk percentage fouten je de hele aanlevering afkeurt. Dat kan natuurlijk per regel maar ook per soort fout, als je echt wil. Meestal is het niet in kunnen lezen van regels een belangrijk foutsignaal en stopt de verwerking, maar er zijn uitzonderingen op die regel en het is handig om de mogelijkheid te hebben dit in het proces te regelen zonder de procescode aan te moeten passen.

Op basis van de foutgegevens uit het inleesproces kan de ETL die de informatie in het data warehouse plaatst vervolgens ook gefundeerde beslissingen nemen over de verwerking. Zo kan een veld met een afwijking worden overgenomen in een 'Error Satellite' of Error Datamart. Zolang Delivery Audit ID, Field Audit ID, veldnaam en de oorspronkelijke inhoud van de aanlevering aanwezig zijn, is alle informatie behouden voor auditing doeleinden. Ook kun je achteraf nog informatie aanvullen indien gewenst.

Overigens moet worden opgemerkt dat fouten in sleutelvelden heel wat vervelender zijn dan fouten in attribuutvelden. Het toevoegen van correcte attributen aan de verkeerde persoon is nauwelijks automatisch te herstellen. Daarom is het verstandig om bij het optreden van fouten in kritieke velden, de zaak niet verder te verwerken zonder handmatige inspectie.

F. Beheersbaar – Staging levert de gegevens waarmee we de performance van de dataleverancier bewaken

Aan het einde van het verwerken van de aanlevering in staging willen we weten wat de kwaliteit van de aanlevering was. Hiermee bewaken we de Service Level Agreement of Operational Level Agreement met de dataleveranciers. Een aantal vragen die daarbij voorkomen zijn bijvoorbeeld:

- Hoeveel aanleveringen hebben we gehad van een leverancier?
- Welke aanleveringen voldeden aan de eisen qua tijdigheid en correctheid?

- Welke niet?
- Hoe lang duurde het voordat de problemen werden opgelost?
- Waren alle velden technisch correct gevuld?

Om dit soort gegevens te kunnen leveren moet de programmatuur die de aanlevering in staging inleest en controleert, naast de al bekende auditgegevens ook nog de volgende gegevens wegschrijven:

- De verwachte tijd van aanlevering en de echte tijd van aanlevering
- Was de aanlevering compleet (alle bestanden, alle leden van een aanlevergroep)?
- Was de aanlevering inhoudelijk correct?
- Hoeveel data-items zaten er in de aanlevering?
- Hoeveel data-items waren correct?
- Hoeveel data-items bevatten fouten?

De fouten kun je eventueel uitsplitsen per soort fout, met behulp van de metadata die doorgaat naar het data warehouse.

De gegevens om die vragen te beantwoorden worden per aanlevering bijgehouden in de tabel waar we ook de overige informatie over de aanlevering bewaren: de Staging Delivery Audit tabel. Deze tabel dient samen met de informatie per aangeleverd veld als bron voor een Data warehouse Audit Mart waarmee we kunnen rapporteren over de processen in het data warehouse. Daar kunnen we natuurlijk ook de conclusies over de kwaliteit van het aanleverproces in kwijt. De precieze invulling van de Delivery Audit tabel is tot stand gekomen in de loop van dit document. De daarin genoemde velden voldoen om de vragen omtrent datakwaliteit van aanleveringen te kunnen beantwoorden.

Staging en realtime aanleveringen

Er is wel een kanttekening te maken wat betreft aanleveringen en dat gaat over real-time data warehouses. Hier geldt dat integratie, opschoning en foutafhandeling allemaal ondergeschikt zijn aan het grote doel van het van seconde op seconde volgen van de werkelijkheid. Bij een maximale vertraging die in seconden wordt gerekend is het duidelijk dat de ETL voor het laden in een data warehouse weinig meer kan doen dan de zaak zo snel mogelijk laden en records met grove fouten die echt niet kunnen worden geladen rapporteren in de audit tabellen. Niet om ze alsnog te laden (fouten komen sneller dan ze kunnen worden beoordeeld), maar om aan te geven waar de problemen zitten zodat de dataleverancier kan worden aangesproken. Voor real-time aanlevering is er ook geen staging, want we gaan niet wachten op andere aanleveringen of data waarmee moet worden geïntegreerd. Daarom laat ik real-time data warehouses hier verder buiten beschouwing. Hou overigens goed in de gaten dat het omzeilen van staging alleen werkt als de data warehouselaag zelf om kan gaan met niet-geïntegreerde aanleveringen. Dat gaat het beste als het data warehouse gemodelleerd is met behulp van Anchor Modelling, Data Vault of aanverwante methodieken. Databases gebouwd in de derde normaalvorm of op geconformeerde dimensies lopen hier tegen problemen op.

Speciale gevallen

Er zijn tegenwoordig steeds meer data warehouses waarbij gegevens worden aangeleverd in de vorm van hiërarchische (XML-)bestanden. Mijn advies is om dit op dezelfde wijze te behandelen als een gewoon tekstbestand, waarbij een deel van de velden nogal vaak dezelfde inhoud heeft.

Wil je echter de hiërarchische relaties in de aanlevering goed afhandelen en meteen in meerdere tabellen stoppen, met behoud van referentiële integriteit, dan krijg je ook alle bijbehorende problemen met afhankelijkheden weer terug. De verwerking van dit soort bestanden valt vanwege de complexiteit dan ook buiten het bereik van dit document.

Een soortgelijk verhaal als voor XML-bestanden geldt ook voor bestanden die uit meerdere recordtypes bestaan, de multirecordbestanden. Bij deze bestanden worden in één bestand meerdere soorten records opgeleverd, niet noodzakelijkerwijs afhankelijk van elkaar. Het eerste veld op de regel geeft dan meestal het recordtype aan. Een veel voorkomend gebruik van multirecordbestanden is om in één bestand zowel de ruwe gegevens als metadata over die gegevens te leveren.

Zolang er voor elk recordtype apart de metadata wordt geleverd die wij nodig hebben voor de verwerking, kunnen we in een voorbereiding het bestand opsplitsen. Vaak is er echter maar metadata voorhanden over het bestand zelf, zoals een totaal telling voor het aantal records. In dat geval zal er een voorbereiding plaats moeten vinden die de aanlevering splitst en tellingen maakt. Dat kan echter alleen als de totaal telling overeenkomt met de som van de tellingen van de delen, want anders verlies je informatie.

Het opsplitsen van de geleverde bestanden zorgt er voor dat het verdere proces gelijk blijft voor alle aanleveringen en heeft verder als voordeel dat je voor de aanlevering op elk recordtype apart kan rapporteren wat de kwaliteit was.

Conclusie

Met een goed opgezette staginglaag kunnen we een hele reeks veel voorkomende problemen in het data warehouse oplossen. Deze uitdagingen variëren van politieke overwegingen en 'best practices' in softwarearchitectuur tot praktische zaken. Op basis van praktijkervaring opgedaan bij meerdere grote ondernemingen draag ik daarvoor een robuuste architectuur aan. Met behulp van die architectuur kunnen de problemen die zich voordoen rond de aanlevering van gegevens in een data warehouse worden losgekoppeld van zaken die een rol spelen in andere delen van het data warehouse en de bronsystemen. Dit leidt tot een stabiel fundament voor het data warehouse.

Voorbeeld

Brontabellen

We gaan uit van de bekende AdventureWorks demonstratiedatabase van Microsoft¹². Deze database bevat een tabel genaamd Customer. Customer ziet er uit als volgt:

Tabel: Customer

Veld	Type	Null?	Opmerkingen
CustomerID	Int	Not null	Primary key
TerritoryID	Int	Null	ID of the territory in which the customer is located. Foreign key.
AccountNumber	Int	Not null	Unique key for the customer
CustomerType	nchar(1)	Not null	I = individual, S = store
Rowguid	uniqueidentifier	Not null	Unique ID for the row, used in replication
ModifiedDate	datetime	Not null	Timestamp of the last update of this row

Op basis van onze architectuurprincipes komen we tot de volgende staging-tabel, waarbij voor de duidelijkheid de namen van de nieuwe tabel en nieuwe velden zijn voorzien van een “STG” prefix. De lengte van de tekstvelden zijn afhankelijk van de maximale grootte van het datatype van het bijbehorende dataveld. Deze heb ik hier dus niet bij opgenomen.

Tabel: STG_Customer

Veld	Type	Null?	Opmerkingen
<i>STG_CustomerID</i>	Tekst	<i>Not null</i>	<i>Primary key</i>
<i>STG_CustomerID_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
CustomerID	Int	Not null	Primary key
<i>STG_TerritoryID</i>	Tekst	<i>Null</i>	<i>ID of the territory in which the customer is located. Foreign key.</i>
<i>STG_TerritoryID_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
TerritoryID	Int	Null	ID of the territory in which the customer is located. Foreign key.
<i>STG_AccountNumber</i>	Tekst	<i>Not null</i>	<i>Unique key for the customer</i>
<i>STG_AccountNumber_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
AccountNumber	Int	Not null	Unique key for the customer
<i>STG_CustomerType</i>	Tekst	<i>Not null</i>	<i>I = individual, S = store</i>
<i>STG_CustomerType_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
CustomerType	Nchar(1)	Not null	I = individual, S = store
<i>STG_Rowguid</i>	Tekst	<i>Not null</i>	<i>Unique ID for the row, used in replication</i>
<i>STG_Rowguid_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
Rowguid	Tekst	Not null	Unique ID for the row, used in replication
<i>STG_ModifiedDate</i>	Tekst	<i>Not null</i>	<i>Timestamp of the last update of this row</i>
<i>STG_ModifiedDate_SFA_ID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Field Audit table</i>
ModifiedDate	Tekst	Not null	Timestamp of the last update of this row
<i>STG_DeliveryLineAuditID</i>	Int	<i>Not null</i>	<i>Foreign key for Delivery Line Audit table</i>
<i>STG_DeliveryAuditID</i>	Int	<i>Not null</i>	<i>ID of the Delivery Audit description of this</i>

¹² Zie voor de database beschrijving <http://msdn.microsoft.com/en-us/library/ms124659.aspx>

			<i>data item.</i>
<i>STG_DataItemNumber</i>	Int	<i>Not null</i>	<i>Linenummer or other data-item number indicating the order in which this item was delivered.</i>
<i>STG_LineMovedToDWH</i>	Char	<i>Not null</i>	<i>Yes/No indicator – Yes when line has been successfully moved into the data warehouse</i>

Het overnemen van rowguid is een discussiepunt. Bij het signaleren van laadproblemen is het beslist een nuttige waarde dus in staging nemen we dit gewoon over. Het is echter onverstandig om technische sleutels ook echt als sleutels in het data warehouse te gebruiken omdat je jezelf daarmee afhankelijk maakt van de implementatie van een bronsysteem (zie principe A). Bij rowguid illustreer ik ook een ander punt, namelijk dat we soms in een brondatabase andere gegevenstypes ondersteunen dan in de doeldatabase.

Ondersteunende tabellen

Tabel: Staging Delivery Audit

Veld	Type	Null?	Opmerkingen
StagingDeliveryAuditID	Int	N	Primary key
SourceSystemName	Tekst		Naam van het bronsysteem van de aanlevering
SourceSystemName	Tekst		Code van het bronsysteem van de aanlevering
Name	Tekst	N	Naam van de aanlevering
Code	Tekst	N	Code voor de aanlevering
SequenceNumber	Int		Volgnummer van de aanlevering bij bron
CreationDate	Datum		Aanmaakdatum van aanlevering bij bron
CreationTime	Tijd		Aanmaaktijd van aanlevering bij bron
CreatingProgramVersion	Tekst		Versie van de aanleverprogrammaatuur
NrOfDataItemsDelivered	Int		Aantal data-items in de aanlevering
Dataformat	Tekst		Het gegevensformaat van de levering, bijvoorbeeld een regel met de geleverde velden, of een XML-schema. Hoeft niet hetzelfde te zijn als het <i>verwachtte</i> gegevensformaat. Eventueel normaliseren naar een subtabel.
StartLoadDateTime	Datum/ Tijd		Datum en tijd van de start van inlezen van de aanlevering in staging
EndLoadDateTime	Datum/ Tijd		Datum en tijd van het einde van inlezen van de aanlevering in staging
LoadingProgramVersion	Tekst		Versie van de ETL-software die is gebruikt om het bestand in te lezen
LoadingResult	Tekst		Het resultaat van het inleesproces (foutcode, tekst, etc.)
NrOfDataItemsLoadedCorrect	Int		Het aantal data-items (regels, XML-tags, records) in de aanlevering wat correct in staging is geplaatst
NrOfDataItemsLoadedError	Int		Het aantal data-items (regels, XML-tags, records) in de aanlevering wat niet kon worden ingelezen
QltyDeliveryOnTime	Char		Was de aanlevering op tijd? (Yes/No)

QltyDeliveryComplete	Char		Was de aanlevering compleet (waren alle bestanden aanwezig)? (Yes/No)
QltyContentComplete	Char		Waren alle data-items en data-elementen die nodig waren ook aanwezig in de aanlevering (is bijvoorbeeld van alle filialen ook data aangeleverd)? (Yes/No)
QltyContentCorrect	Char		Was de aanlevering inhoudelijk correct? (Yes/No)
QltyNrItemsCorrect	Int		Hoeveel data-items waren correct?
QltyNrItemsIncorrect	Int		Hoeveel data-items waren incorrect?

Om overweg te kunnen met deliverygroups en/of ontbrekende aanleveringen kunnen we de volgende velden toevoegen:

DeliveryStatus	Tekst		Status van de aanlevering: nog niet geleverd, geleverd, in verwerking, verwerkt, aanleverfout, niet geleverd, komt niet meer, etc.
EarliestExpectedCreationDate	Datum		Vroegst verwachte aanmaakdatum van aanlevering bij bron
EarliestExpectedCreationTime	Tijd		Vroegst verwachte aanmaaktijd van aanlevering bij bron
LastExpectedCreationDate	Datum		Laatst verwachte aanmaakdatum van aanlevering bij bron
LastExpectedCreationTime	Tijd		Laatst verwachte aanmaaktijd van aanlevering bij bron

De bovenstaande velden kunnen vooraf worden ingevuld in de Staging Delivery Audit tabel, conform de frequentie en leadtime/lagtime zoals aangegeven in het aflevercontract. Komt er dan een aanlevering, dan worden de overige gegevens gevuld. Zo niet, dan geeft de DeliveryStatus aan dat je iets hebt gemist.

Tabel: Staging Delivery Field Audit

Veld	Type	Null?	Opmerkingen
StagingDeliveryFieldAuditID	Int	N	Primary key
Ind_Unreadable	Char		The field was unreadable (yes/no)
Ind_Corrupt	Char		The field was corrupt (yes/no)
Ind_UnexpectedNull	Char		The field was empty when we expected a value (yes/no)
Ind_TooLarge	Char		The field was larger than the defined size of the target field (yes/no)
Ind_NotAnInteger	Char		The field was not an integer (yes/no)
Ind_NotAFloat	Char		The field was not a valid floating point number (yes/no)
Ind_NotADate	Char		The field was not a valid date (yes/no)
Ind_NotATime	Char		The field was not a valid time (yes/no)
...			

Tabel: Staging Delivery Line Audit

Veld	Type	Null?	Opmerkingen
StagingDeliveryLineAuditID	Int	N	Primary key
Ind_LineTooLarge	Char		The line was larger than we could store (Y/N)
Ind_LineCorrupt	Char		The line was unreadable
Ind_NotEnoughFields	Char		Not enough fields found in the line
Ind_TooManyFields	Char		Too many fields found in the line
....			

Tabel: Staging Delivery Contracts

Veld	Type	Null?	Opmerkingen
StgDeliveryContractId	Int	N	Primary key
DeliveryNaam	Tekst	N	Naam van de aanlevering
DeliveryCode	Tekst	N	Code voor de aanlevering
DeliveryFrequency	Tekst	N	Tekst die aangeeft hoe vaak we een delivery verwachten: daily, weekly, Sundays, realtime, ad-hoc, etc. De tekst moet worden begrepen door de programmatuur
DeliveryCreationTime	Tijd	N	Verwachte aanmaaktijd van de bron
PreDeliveryLag	Tijd	N	Aantal uren/minuten speling voor de geplande aanmaaktijd
PostDeliveryLag	Tijd	N	Aantal uren/minuten speling na de geplande aanmaaktijd
OutOfBoundsAction	Tekst	N	Tekst die beschrijft wat er moet gebeuren als de aanlevering buiten de aangegeven tijden is aangemaakt: 'accept', 'decline' o.i.d.
MaxNumberOfLineErrors	Int		Bij hoeveel foute regels in de aanlevering keuren we de hele aanlevering af
MaxPctOfLineErrors	Int		Bij hoeveel procent foute regels in de aanlevering keuren we de hele aanlevering af
... (verdere levercontractgegevens)

NB: Er zit geen relatie tussen deze tabel en Staging Delivery Audit omdat het mogelijk moet zijn een aanlevering in te lezen ondanks een fout in de meegeleverde metadata.

Tabel: Staging Delivery Groups

Veld	Type	Null?	Opmerkingen
StgDeliveryGroupId	Int	N	Primary key
DeliveryGroupName	Tekst	N	Naam van de aanleveringsgroep
DeliveryGroupCode	Tekst	N	Code van de aanleveringsgroep

Tabel: Staging Delivery Grouplink

Veld	Type	Null?	Opmerkingen
StgDeliveryGroupId	Int	N	Primary key voor Staging Delivery Groups
StgDeliveryContractId	Int	N	Primary key voor Staging Delivery Contracts