Banken zijn geen conservatieve IT-ers

Finance Data Portal belangrijke speler

Veel mensen zullen bij IT in de bankwereld niet gelijk denken aan innovatieve databasetechniek. Toch is die wel degelijk aanwezig. In twee artikelen proberen Kenneth Hammink en Gert-Jan Paulissen dat beeld bij te stellen. Zij beschrijven daarin de werking van de Oracle Finance Data Portal, een applicatie die de gegevensstroom goed kanaliseert en beheert. In dit nummer het eerste deel van hun tweeluik.

De Finance Data Portal is een Oracle-applicatie die alle in- en uitkomende data ontvangt en verzendt, voor zowel interne als externe partijen. Het grote voordeel hiervan is dat alles gecentraliseerd is. De andere interne applicaties hoeven slechts één type interface te hebben: de koppeling naar het Finance Data Portal. Het Finance Data Portal is een belangrijke speler in de wereldwijde verwerking van de financiële gegevens van de Nederlandse Grootbank.
Vanuit de verschillende front- en backoffice systemen wordt gedurende de dag data aangeleverd aan de Finance IT afdeling. Deze data heeft onder andere betrekking op alle financiële transacties gedaan voor en door klanten van de bank.
Binnen de Finance IT afdeling worden deze gegevens verwerkt door Financial Accounting-, Management Accounting-, Risk- en Reporting systemen. Tevens worden gegevens doorgeleverd aan applicaties verspreid over de hele wereld.
In het verleden kreeg ieder systeem zijn eigen data aangeleverd en wisselden applicaties ook nog onderling informatie uit. In de loop der jaren is hierdoor een complexe omgeving ontstaan waarbij gegevens middels een grote keur van formaten en protocollen werden uitgewisseld.

Om de gegevensstroom goed te kunnen kanaliseren en beheren is de Finance Data Portal ontwikkeld. Bij de eerste versie van het Finance Data Portal werd gebruik gemaakt van Oracle Warehouse Builder (OWB) om bestanden in te laden. OWB versie 10.1.0.3 bleek niet geschikt voor dit doel. Een nadeel was dat het laadproces voor veertig procent uit overhead bleek te bestaan. Verder was het verwerken van de maandelijkse wijzigingen in de aanlevering van verschillende back-office systemen bewerkelijk en tijdrovend. Ook kwam het regelmatig voor dat laadprocessen hingen.
Aan belangrijke eisen van de Finance Data Portal, zoals eenvoudig beheer en snelle en betrouwbare dataverwerking, werd hierdoor niet voldaan. Hierdoor is gekozen voor een compleet andere aanpak.

De voorwaarden die gesteld worden aan het Finance Data Portal zijn:
- data wordt ontvangen of verzonden via bestanden, message queues (IBM MQ en Advanced Queueing) of tabellen/views uit andere Oracle databases.
- data kan worden aangeleverd in binair formaat, in tekstformaat of als XML (met een XML schemadefinitie).
- data die binnenkomt kan in een andere vorm of ander formaat naar buiten.
- real-time aanleveringen kunnen worden doorgeleverd als batch aanleveringen.
- het toevoegen van een nieuwe interface moet een minimale impact op de software hebben.
- tabelaanpassingen als een kolom erbij of eraf moeten een minimale impact hebben.
- zoveel mogelijk functionaliteit in de database, ook O/S gerelateerde zaken als het kopiëren van bestanden middels FTP of opvragen van een lijst van bestanden in een folder.

Oplossingen

De oplossingen die zijn gekozen zijn:
1. Invoering van versiebeheer in het datamodel.
2. Bij het ontvangen van tekstbestanden worden ‘external tables’ gebruikt om ze in te lezen. Bij het verzenden wordt een tekstbestand weer op de juiste plaats gezet en kan de ‘external table’ weer gebruikt worden om de data te verzenden. Er is dus geen opslag in tabellen nodig voor tekstbestanden, hetgeen voordelig is voor de performance. Het gevolg hiervan is wel dat door de applicatie locks geplaatst moeten worden op de ‘external table’ bestandslocatie om ervoor te

zorgen dat op één moment slechts één proces een bestandslocatie kan gebruiken.
3. MQ Series aanleveringen worden via een MQ/AQ gateway overgezet naar een Oracle Advanced queue en vervolgens binnen de database verwerkt.
4. Aanleveringen vanuit andere Oracle databases via tabellen en views worden als bestand opgeslagen en vervolgens door een 'external table' verwerkt. Deze aanpak zorgt ervoor dat te allen tijde exact dezelfde gegevens van de tabel of view kunnen worden getoond.
5. XML wordt ingelezen via SAX en opgeslagen in (gegenereerde) tabellen. XML DB werkt gewoon niet met grote XML bestanden. Dit vertellen ze niet bij Oracle.
6. Generatie van code om foutgevoelig onderhoud te verminderen.
7. Alle verschillende typen interfaces worden verwerkt door Oracle Object Types. Deze object-relationele eigenschappen van Oracle zie je niet vaak in de praktijk gebruikt worden.
8. Java in de database wordt gebruikt om O/S functionaliteit aan te kunnen roepen. Het is echter behoorlijk trager dan rechtsreeks vanuit het O/S. Hier hebben we ook oplossingen voor moeten vinden.

In dit artikel behandelen we kort oplossingen 1,2,5 en 6. In het volgende nummer gaan we dieper in op de laatste twee van bovenstaande oplossingen.

Versiebeheer in het datamodel

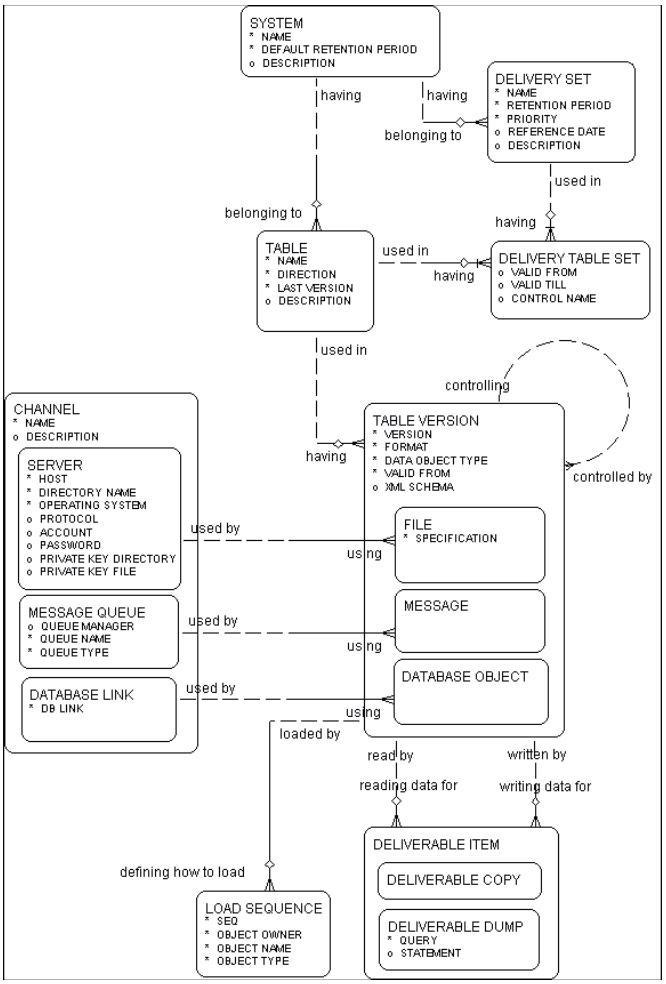
Aanleveringen komen binnen op de database servers in sets. Voor bestandsaanleveringen zijn dit normaliter dumps van een tabel in combinatie met controlegegevens waarin staat wat er is aangeleverd (geldigheidsperiode van de data, aantal rijen en een controlegetal). Het kan dat de data van opbouw verandert, bijvoorbeeld door een kolom erbij. Een nieuwe kolom is niet voor alle afnemers interessant waardoor de binnenkomende gegevensstructuur niet synchroon hoeft te lopen met de uitgaande. De applicatie moet hier goed mee overweg kunnen. Daarvoor zijn tabelversies geïntroduceerd: een versie van een tabel is geldig vanaf een datum totdat er een volgende versie is.

Gebruik van external tables

In de eerste versie van het Finance Data Portal werd met behulp van Oracle Warehouse Builder mappings de data geladen in een databasetabel. Dit gaf veel administratie bij wijzigingen en is ook niet snel bij grote tabellen. Het innovatieve idee is toen geboren om helemaal geen databasetabellen te gebruiken. De uitgaande data is tenslotte gebaseerd op de ingaande data. Dus als er een verzoek om de inkomende data ligt, wordt het betreffende bestand vanuit een gecomprimeerd archief op de plek van de external table gezet en kan er gelezen worden. Door middel van locking wordt ervoor gezorgd dat niet meerdere processen tegelijkertijd dezelfde bestandslocatie kunnen gebruiken. Dit hele proces van locken en plaatsen van het bestand wordt door middel van een functie gerealiseerd. Deze functionaliteit wordt ook weer gebruikt in (gegenereerde) table functies die de basis vormen van uitgaande aanleveringen via views.

XML en de database

Door een interne partij wordt van hun Oracle databasetabellen een XML dump gemaakt die is beschreven met een XML schemadefinitie. Dit is overigens helaas een overduidelijk voorbeeld van grote inefficiëntie: met een database link is de tabel van een andere Oracle database zo ingelezen. Budgettaire redenen lagen hieraan ten grondslag.
Maar het is wel een technische uitdaging om een bestand met slechts 50.000 contracten in te lezen. Het standaard Oracle type XMLType leent zich daar niet voor: het bestand is gewoon te groot voor de DOM parser die gebruikt worden door XMLType en andere PL/SQL programmatuur. Je krijgt het dus niet via SQL of PL/SQL de database in. Blijft over de SAX parser. Met behulp van een Java-programma wordt nu de XML gevalideerd tegen de schemadefinitie en daarna ingelezen in



Figuur 1- Finance Data Portal ERD Systeem Configuratie

tabellen. Deze tabellen zijn gegenereerd aan de hand van de schemadefinitie.

Generatie van code
Om het onderhoud qua gebruik zo simpel mogelijk te maken is ervoor gekozen zoveel mogelijk code te laten genereren. Het vele handmatige werk in de eerste versie van het Finance Data Portal is hier de oorzaak van. De applicatie werkt met CDM Ruleframe en kent veel zogenaamde Change Event regels, waarbij DML of DDL uitgevoerd kan worden.
Een voorbeeld is de generatie van uitgaande views die gegenereerd kunnen worden als er een link tussen in- en uitgaande tabel versies wordt gemaakt, waarbij de query wordt gebruikt om een table functie plus view aan te maken. Een ander voorbeeld is het aanmaken van tabellen op basis van een XML schemadefinitie.
Omdat dit DDL is en dus een impliciete commit veroorzaakt, wordt de generatie via een database job uitgevoerd.

Tips & trucs

Kopiëren van bestanden of verplaatsen
Op Unix is een verplaatsing van een bestand op hetzelfde bestandssysteem zeer snel, doordat er fysiek geen data wordt verplaatst maar alleen de interne administratie van Unix bijgewerkt hoeft te worden. In eerste instantie werden de bestanden, die op de database server binnenkwamen, gekopieerd naar de bestandslocatie van de externe tabel. Door dit te veranderen in een verplaatsing is de applicatie zo'n dertig procent sneller geworden.

Gebruik van database jobs
Bij een aanlevering, die bestaat uit meerdere tabellen, wordt de data tabel voor tabel ingelezen en gevalideerd. Op zich is het hierdoor simpel om het geheel als een transactie te behandelen, maar het heeft wel nadelen voor wat betreft de performance. Er is nu een voorziening gemaakt om per tabel een job te starten die het zware werk doet zodat alle tabellen parallel verwerkt kunnen worden. De applicatie zal dan wachten tot alle jobs klaar zijn. Als de job faalt, dan wordt door de job de SQL foutcode weggeschreven in een tabel die weer kan worden uitgelezen door het hoofdproces die deze fout dan escaleert.
Dit gebruik van database jobs brengt uiteraard extra werk met zich mee voor de database, maar de doorloopsnelheid wordt hierdoor wel verhoogd.

Unit testen
Voor de hele applicatie is 30.000 regels unit test code (utPLSQL) geschreven tegen 77.000 regels gewone code. Dit heeft er wel voor gezorgd dat het er heel weinig fouten in productie zijn opgetreden. Ook zorgt het ervoor dat 'refactoring', wat binnen de Java wereld regelmatig gebruikt wordt, ook veiliger met PL/SQL kan worden gedaan.
Het ontwikkelen en uitvoeren van unit testen vergt van de ontwikkelaar een bepaalde discipline. Het lijkt veel tijd te kosten, maar het betaalt zich wel uit. Na 1 jaar productie zijn nauwelijks incidenten (minder dan 10) opgetreden die te relateren zijn aan softwarefouten.

Conclusie

Door bij het ontwerp van een applicatie rekening te houden met de vele interfacewijzigingen is het gelukt om het onderhoud hiervoor flink te reduceren. Voor deze oplossing zijn de grenzen van de Oracle software opgezocht: het gebruik van bijvoorbeeld Oracle types, XML, Java, datamodellen met versiebeheer en codegeneratie zijn niet voor elke ontwikkelaar dagelijkse kost. De Finance Data Portal applicatie kan echter niet zonder. De bankwereld hoeft qua IT dus zeker niet conservatief te zijn.

Referenties

'Doodsbericht', Rene Veldwijk, DB/M Magazine nummer 3, 2008.

Kenneth Hammink en **Gert-Jan Paulissen** werken bij een Nederlandse Grootbank aan een zogenaamd Finance Data Portal.