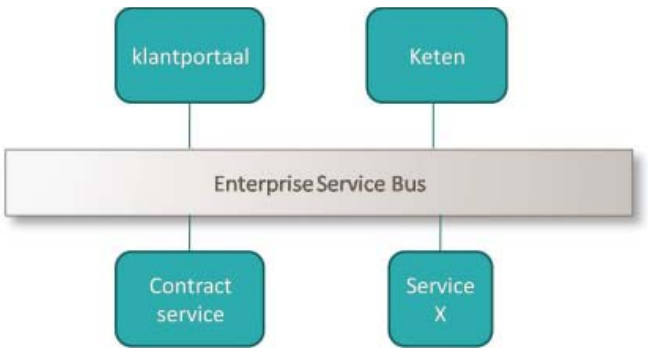
Gebruik database applicaties in SOA

Belangrijke voordelen van een service georiënteerde architectuur voor een organisatie zijn flexibiliteit en efficiëntie. Efficiëntie wordt bereikt doordat functionaliteit maar één keer gerealiseerd hoeft te worden en vervolgens in verschillende processen hergebruikt kan worden. Flexibiliteit wordt bereikt, omdat veranderingen eenvoudig te realiseren zijn door componenten als losse onderdelen te bouwen. Naast flexibiliteit en efficiëntie kan ook ketenintegratie bereikt worden in een SOA-omgeving, door het gebruik van standaarden, zoals webservices. Dit artikel beschrijft een aantal mogelijkheden waarop bestaande bedrijfslogica in database systemen hergebruikt kan worden in een service gerichte omgeving.

Hergebruik van bestaande IT-componenten in een SOA is belangrijk: het zorgt ervoor dat een organisatie niet alle IT-systemen hoeft te vervangen, maar een groeipad kan gebruiken voor het bereiken van de gewenste architectuur. Bescherming van bestaande systemen wordt gewaarborgd door een servicelaag om de systemen te plaatsen.

Situatie

In dit artikel gaan we uit van een fictieve, maar ongetwijfeld herkenbare, situatie: een bestaande Oracle Forms applicatie wordt gebruikt bij een aantal afdelingen van een organisatie. De meeste bedrijfslogica is ontwikkeld in PL/SQL en database triggers. De organisatie heeft een aantal speerpunten in de uitvoering van processen benoemd, waaronder selfservice en ketenintegratie. Dit betekent dat klanten en partners zelf gegevens kunnen doorgeven en wijzigen en dat dit doorgevoerd wordt zonder tussenkomst van een medewerker (tenzij er sprake is van uitval). Daarnaast worden processen met partners geïntegreerd, zowel in de systemen als in de organisatie.
Het bedrijf heeft in het kader van de selfservice besloten om een nieuwe website te bouwen, waarin klanten en partners gegevens kunnen wijzigingen. In een ander project wordt invulling gegeven aan het begrip ketenintegratie door een geautomatiseerd proces in te richten om gegevens die worden aangeleverd door partners automatisch en direct te verwerken en door te spelen naar een volgende partner in het proces.
Omdat het bedrijf enige tijd geleden gestart is met een service georiënteerde architectuur, heeft het een Enterprise Service Bus (ESB) ingericht die de communicatie tussen services afhandelt (zie Figuur 1).
Om te onderzoeken hoe beide projecten (klantportaal en ketenintegratie) het best ondersteund kunnen worden met zoveel mogelijk hergebruik van bestaande systemen, wordt één service gerealiseerd: de zogeheten 'klantservice'. In dit artikel worden de mogelijke oplossingen om de klantservice te realiseren, beschreven.



Figuur 1. Klant service en ESB

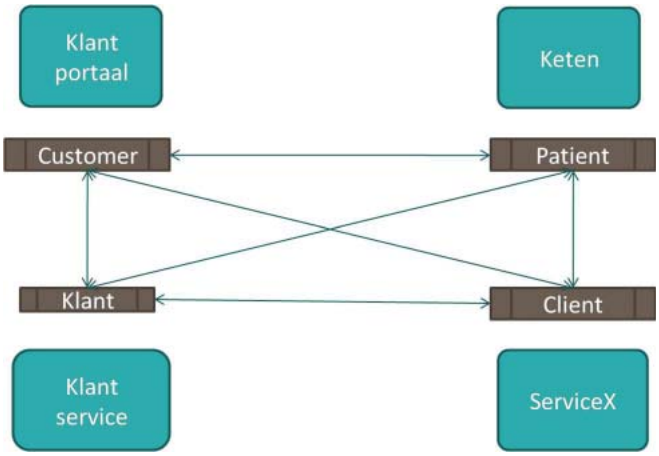
De PL/SQL functie die we hergebruiken ziet er als volgt uit:

```
PACKAGE BODY optimize AS

FUNCTION geef_klant (naam varchar2)
    RETURN varchar2  IS

          v_klantnaam varchar2(50);
        BEGIN
        v_klantnaam:= CONCAT('ik ben klant ', naam);
        RETURN v_klantnaam;

    END geef_klant;
END optimize;
```

We gebruiken hier een simpel voorbeeld om een en ander te illustreren. In werkelijkheid zijn de functies en procedures natuurlijk veel complexer. De simpele service die we willen aanbieden aan afnemers is een webservice die aan het volgende schema voldoet:

```
<element name="geefKlantRequest">
                <complexType>
                    <sequence>
                        <element name="header" type="string"
fixed="demo"/>
                        <element name="klantnaam" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="geefKlantResponse">
                <complexType>
                    <sequence>
                        <element name="header" type="string"
fixed="demo"/>
                        <element name="klant" minOccurs="0" maxOc-
curs="1">
                            <complexType>
                                <sequence>
                                    <element name="achternaam"
type="string"
                                             minOccurs="0" maxOc-
curs="1"/>
                                    <element name="tussenvoegsel"
type="string"
                                             minOccurs="0" maxOc-
curs="1"/>
                                    <element name="voorvoegsel"
type="string"
                                             minOccurs="0" maxOc-
curs="1"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
```

Zoals uit de bovenstaande beschrijving is af te lezen, bevat 'klant' in het bericht anders gedefinieerde eigenschappen (achternaam, tussenvoegsel, voornaam) dan de database procedure die klantgegevens (naam) teruggeeft. De organisatie heeft in het kader van de SOA en ESB realisatie besloten een zogeheten 'canonical datamodel' (CDM) te implementeren om afhankelijkheden tussen applicaties nog verder te ontkoppelen. Dit betekent dat er afspraken worden gemaakt over de vorm, inhoud, en betekenis van gegevens waarover gecommuniceerd wordt. Iedere afnemer weet daardoor direct hoe bijvoorbeeld een 'klant' of een 'contract' er uitziet. Een bericht wordt altijd vertaald naar het 'canonical' formaat. Van daaruit kan het dan eventueel vertaald worden naar een afnemerspecifiek formaat. Dit zorgt ervoor dat het aantal benodigde transformaties veel kleiner is dan wanneer geen CDM gebruikt wordt: in plaats van een transformatie per koppeling zoals in Figuur 2, hebben we een transformatie per service (zie Figuur 3).



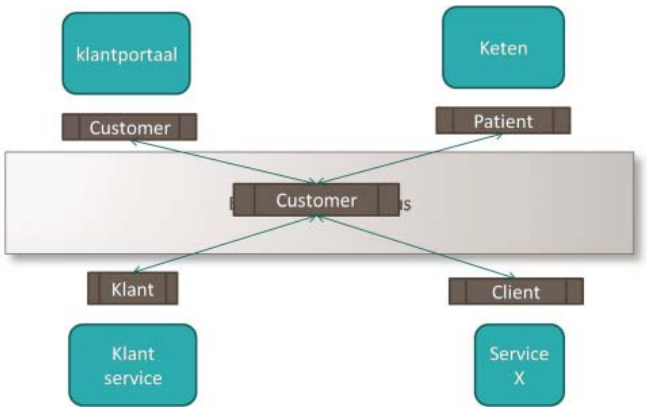
Figuur 2. Transformaties per koppeling zonder CDM

Een CDM brengt natuurlijk wel beheer met zich mee: aanvullingen moeten worden opgenomen, wijzigingsverzoeken beoordeeld, enzovoorts.

Mogelijkheden

Er zijn een aantal mogelijkheden bij het ontwerp van een dergelijke service. De oplossingen verschillen in de mate van verantwoordelijkheid die aan de database applicatie opgelegd wordt ten aanzien van de interface specificatie en het protocol:

1) Applicatie. De klant-relatie-applicatie stelt de service beschikbaar door een webservice te bieden volgens de interface afspraken, protocol en het CDM.
2) Enterprise Service Bus. De klant applicatie stelt de service beschikbaar als PL/SQL procedure. De ESB is verantwoordelijk voor zowel de data transformatie naar het CDM als de protocol transformatie naar SOAP over HTTP.
3) Tussenvorm.
    i. De klant-relatie-applicatie biedt de service aan via een PL/SQL procedure die een XML bericht teruggeeft volgens het canonical formaat. De ESB is vervolgens



Figuur 3. Transformatie per service met CDM

verantwoordelijk voor de protocol transformatie, **of**

ii. De klant applicatie biedt een webservice, de Enterprise Service Bus is verantwoordelijk voor de datatransformatie naar het CDM.

De voor en nadelen van elk van deze opties, en de mogelijkheden die Oracle hiervoor biedt, worden hieronder besproken. Belangrijke criteria hierbij zijn:

- Beheer. Services zullen, los van de bedrijfslogica, beheerd worden. Het is belangrijk dat logging, beveiliging, foutafhandeling en andere quality of service (QoS) aspecten van de services eenduidig belegd kunnen worden.
- Ontwikkelen. Het beschikbaar stellen in een SOA-omgeving vereist kennis van (web)services, XML en allerlei standaarden. Deze moet aanwezig zijn bij de groep die verantwoordelijk is voor een bepaalde oplossing.
- Schaalbaarheid. Services zijn bedoeld om hergebruikt te worden. Het is belangrijk dat een toename in afnemers en aantallen berichten opgevangen kan worden en dat wijzigingen in de bedrijfslogica en de interface onafhankelijk van elkaar veranderd kunnen worden.
- Tooling. Het genereren van XML documenten, webservices en andere specifieke webservice producten wordt door sommige tools beter ondersteund dan door andere. Denk hierbij aan het wijzigen van een namespace, URL of element. Ook het eenduidig gebruiken van het CDM moet ondersteund worden.

Verantwoordelijkheid bij Applicatie

Als de verantwoordelijkheid om te voldoen aan de interface-eisen volledig bij de applicatie ligt, betekent dit dat we een webservice en een transformatie moeten bouwen in de klant-relatie-applicatie om de datastructuur van de procedure om te vormen naar het canonical formaat.

Oracle biedt een aantal mogelijkheden om dit te realiseren:

- 'Database native webservices'. XMLDB biedt vanaf Oracle database 11g zogeheten 'database native webservices' . Hierbij heeft de ontwikkelaar weinig controle over de gegenereerde payload van het bericht. Dit betekent dat het lastig is om de webservice exact (inclusief imports, namespaces, operatie namen) volgens de interface specificatie te bouwen.
- Java Webservice. Een Java applicatie bouwen, die een PL/SQL procedure aanroept, en het resultaat vertaalt naar een XML document. Hier omheen kan een Java webservice gegenereerd worden met JDeveloper. De ontwikkelaar heeft hierbij volledige controle over de interface, omdat de WSDL het uitgangspunt is bij de generatie van de Java code.
- XMLType. De derde mogelijkheid is een PL/SQL procedure bouwen die XML teruggeeft in het canonical formaat . Op basis van de PL/SQL procedure, kan dan een PL/SQL Webservice gegenereerd worden in JDeveloper. Hierbij heeft de ontwikkelaar, net als bij de 'database native webservice', weinig controle over de payload en de operatie namen die gegenereerd worden (Zie tabel 1).

Verantwoordelijkheid bij de Enterprise Service Bus

Wanneer de verantwoordelijkheid voor transformatie van zowel het formaat als het protocol bij de Enterprise Service Bus ligt, kan een Databaseadapter ingezet worden. Oracle heeft op dit moment twee service bus producten: de Oracle

Optie	Beheer	Kennis	Schaalbaarheid	Tooling	Opmerking
Database native webservice	Op locatie: bij de database. Niet centraal bij de services op de Applicatieserver. Geen hergebruik CDM (moet bekend zijn in de database en in de ESB/appserver)	Bijscholing van PL/SQL ontwikkelaars op gebied van XML Standaarden	Database connecties is bottleneck. Verandering is een wijziging die in de database moet plaatsvinden.	Extra tooling nodig, voor het maken van WSDL, XML, XSD's, testen	Voor kleine toepassingen (niet enterprise SOA) geschikt.
Java Applicatie	Centraal beheer van services: Applicatieserver.	Kennis van webservices standaard uitrusting van Java programmeurs.	Gegevens kunnen gecached worden. Applicatieserver load balanced. Connection pooling.	JDeveloper en andere java tooling bieden XML mogelijkheden.	Geschikt voor bedrijven met goede Java kennis
XML uit PL/SQL procedure	Disjunct: deels bij de DB, deels bij de Applicatieserver. Geen hergebruik CDM (moet bekend zijn in de database en in de ESB/appserver)	Bijscholing van PL/SQL ontwikkelaars op gebied van XML Standaarden	Gegevens kunnen gecached worden. Applicatie server load balanced. Connection pooling.	Extra tooling nodig, voor het maken van WSDL, XML, XSD's, testen	Erg veel nadelen in onderhoud, tooling en beheer.

Tabel 1.

Optie	Beheer	Kennis	Schaalbaarheid	Tooling	Opmerking
Oracle Enterprise Service Bus (OESB)	Geen hergebruik van transformaties, WSDLs etc. Aan WSDL, XSLT wordt gerefereerd op basis van locatie, niet logische naam.	Kennis van OESB Adapters en XML familie nodig	Connection pooling. Interface van procedure kan los van de webservice interface veranderd worden.	JDeveloper	Zeer geschikt als adapters gebruikt worden.
Oracle Service Bus (OSB)	Hergebruik van resources. Aan WSDL, XSLT wordt gerefereerd op basis van logische naam.	Kennis van OSB, adapters en XML familie nodig	Connection pooling. Interface van procedure kan los van de webservice interface veranderd worden.	Eclipse en Console	Pas geschikt als de JCA adapters beschikbaar zijn.

Tabel 2.

Optie	Beheer	Kennis	Schaalbaarheid	Tooling	Opmerking
Genereren Webservice, transformatie in ESB	Centraal beheer services (Applicatieserver/ESB)	Geen extra kennis nodig.	Connection pooling	Ondersteuning in JDeveloper	Geschikte oplossing, zeker als ESB geen database adapter heeft
PL/SQL procedure geeft XML terug	Disjunct: Beheer bij database en Applicatieserver/ESB	XML kennis bij DB ontwikkelaars	Wijziging van interface moet op twee plaatsen gebeuren (DB en Applicatieserver)	Extra tooling nodig, voor het maken van WSDL, XML, XSDs, testen.	Atypische, lastige oplossing.

Tabel 3.

Enterprise Service Bus (OESB) en de Oracle Service Bus (OSB). Deze laatste is als strategisch product aangewezen in de roadmap die Oracle heeft gepubliceerd. De verschillende JCA adapters, waaronder de databaseadapter, zijn in de huidige release (Oracle Service Bus 10g R3) nog niet zichtbaar voor gebruikers. De komende maanden zullen deze gecertificeerd worden voor OSB. Met OESB, zetten we de databaseadapter als volgt in: We definiëren met de wizard een databaseadapter die de PL/SQL functie, die we eerder gedefinieerd hebben, aanroept. Vervolgens zetten we er een routing service voor, die voldoet aan de WSDL zoals eerder beschreven. Met behulp van een XSL map vertalen we de klantdefinitie van de WSDL naar de gegevens die worden teruggeven uit de databaseadapter.

Zoals te zien is in tabel 2, heeft de Oracle Service Bus de voorkeur zodra de JCA-adapters beschikbaar zijn.

Tussenvorm: gedeelde verantwoordelijkheid

Een variant op de twee bovenstaande mogelijkheden is dat de verantwoordelijkheden gedeeld worden. Een webservice kan gecreëerd worden op basis van een bestaande PL/SQL functie. Dit kan via een 'database native webservice', of via een PL/SQL webservice die vanuit JDeveloper wordt gegenereerd. De transformatie naar het CDM is met behulp van een XSL map en een routing service te realiseren. Andersom kan ook, de applicatie kan een XML bericht teruggeven volgens het CDM. De ESB is dan verantwoordelijk voor de protocolvertaling van PL/SQL naar een Webservice (Zie tabel 3).

Conclusie

Er zijn een aantal mogelijkheden wanneer een organisatie bestaande bedrijfslogica uit de database wil hergebruiken in een SOA-omgeving. Wanneer de verantwoordelijkheid compleet bij de bestaande applicatie gelegd wordt om aan de interface te voldoen, moet met een aantal zwaarwegende nadelen rekening gehouden worden. Het verdient daarom de voorkeur om de tussenvorm te kiezen of de verantwoordelijkheid voor de data en protocol transformatie bij de ESB te leggen. Welke van deze twee opties de voorkeur heeft, hangt sterk af van de situatie: de gekozen service bus, de kennis en ervaring van de Oracle ontwikkelaars, de load en performance eisen.

Voor meer informatie:

- XML DB developers guide. http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28369/toc.htm
- First Tests of 11g Native Web Services: http://tardate.blogspot.com/2007/08/first-tests-of-11g-native-web-services.html.
- Oracle Service Bus. http://www.oracle.com/technologies/soa/service-bus.html

Lonneke Dikmans is Oracle Ace Director en managing partner bij Approach Alliance; een IT bedrijf dat gespecialiseerd is in SOA en BI.