Edwin Smit is technisch architect bij Qurius ETX

Integreren is kiezen

MICROSOFT INTEGRATIETECHNOLOGIEËN: OVERWEGINGEN EN ALTERNATIEVEN

De opdracht klinkt eenvoudig: koppel ons systeem A aan het nieuwe systeem B. Vol goede moed, maar duizelend van de mogelijkheden slaat de twijfel toe. Gaan we een product inzetten of kiezen we voor zelf bouwen? BizTalk Server 2006? Kunnen we SQL Server Integration Services inzetten? Hoe zit het met Windows Communication Foundation?

Dit artikel geeft een overzicht van de integratietechnologieën die Microsoft biedt en bespreekt hun belangrijkste voor- en nadelen. Het probeert zo duidelijkheid te scheppen in het totale aanbod en geeft adviezen over de optimale inzet van de beschikbare alternatieven op het Microsoft-platform. De beschreven situaties en keuzemogelijkheden zullen menig ontwikkelaar bekend voorkomen. Met dit artikel kan hij de alternatieve scenario's met elkaar vergelijken om zo tot een leidraad te komen voor zijn eigen, specifieke oplossing. Met in ieder geval één overeenkomst: simpel is het nooit, uitdagend des te meer. We kunnen integratieprojecten grofweg indelen in twee soorten: integratie van applicaties en integratie van data. Bij integratie van applicaties ligt de primaire focus op het hergebruik van de logica van de te integreren applicatie, zonder deze opnieuw te hoeven bouwen. Dit kan via directe aanroepen, via een wachtrij of door gebruik te maken van de services van een broker. Al deze opties komen hieronder aan bod.

Directe integratie van applicaties

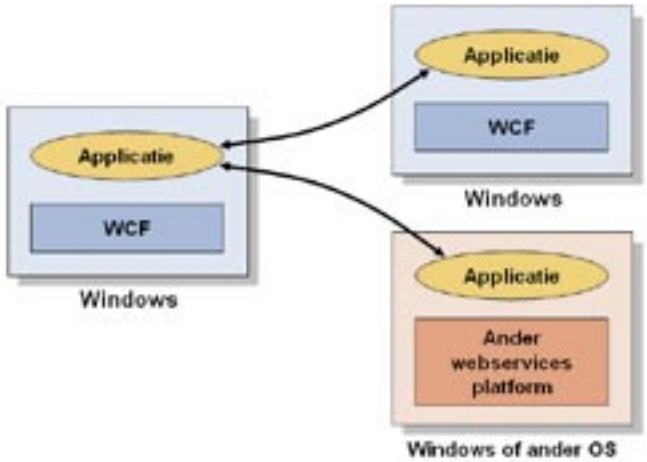
Directe integratie is de eenvoudigste vorm van integratie: we roepen functies in het oude systeem aan en hergebruiken op deze manier de logica. De communicatie is synchroon van aard, als Remote Procedure Calls (RPC). Voor iedere koppeling is een aparte interface nodig. Een voorwaarde is dat het gebruikte protocol en het formaat van te voren vaststaan. Als ontwikkelaar hebben we dan de volgende technologieën tot onze beschikking.

Webservices: Microsoft heeft het in Visual Studio bijzonder gemakkelijk gemaakt om een methode beschikbaar te stellen als webservice.Via ASP.NET-webservices kunnen we via SOAP koppelen aan externe systemen. Omdat dit via Web Services Interoperability-standaarden (WS-I) gebeurt, is het in theorie goed mogelijk om zo aan een ander webservice-platform te koppelen zoals J2EE op Linux. We kunnen communiceren via poorten 80 (http) of 443 (https); deze staan meestal toch al open. In de praktijk zien we echter dat webservices vooral worden gebruikt voor koppelingen tussen twee Windows-servers. Microsoft heeft in zijn samples en documentatie een sterke voorkeur om data uit te wisselen via datasets. Methodes die datasets gebruiken kunnen vrij eenvoudig via webservices beschikbaar worden gesteld. Door het .NET Framework worden die datasets daarvoor zodanig geserializeerd dat het erg lastig is om er op een ander platform gebruik van te maken.

Remoting: Remoting houdt in dat we via een proxy-en-stubmechanisme Remote Procedure Calls kunnen doen van windows-applicatie naar windows-applicatie. Dit geeft echter problemen met de security en vraagt om firewall-aanpassingen. Aangezien het daarnaast veel gemakkelijker is om een webservice te bouwen, kunnen we Remoting beschouwen als een alternatief dat op sterven na dood is.

Enterprise Services / COM+: Enterprise Services, de .NET-wrapper om COM+, stelt ons in staat gebruik te maken van gedistribueerde transacties. We kunnen er ook een beroep op doen als we in ons nieuwe .NET-project niet de tijd hebben om oude COM+ componenten te herschrijven naar .NET of wanneer we moeten verdedigen dat de grote investering in Visual Studio 6 COM+ componenten niet voor niets is geweest. Deze technologie heeft dezelfde nadelen als Remoting voor wat betreft de configuratie van de firewall.

Windows Communication Foundation: Voorgenoemde technologieën zijn opgevolgd door Windows Communication Foundation (WCF), voorheen bekend als Indigo. Het maakt gebruik van SOAP en de Web Services Interoperability- (WS-I) specificaties om betrouwbare, veilige en transactionele communicatie met andere systemen of een ander platform zoals Linux mogelijk te maken. Hierbij kan het uit te wisselen Message-formaat losgekoppeld worden van het transportmechanisme. Het standaard op XML-gebaseerde SOAP-formaat kan gebruikt worden voor bijvoorbeeld Windows naar J2EE-communicatie, en binair SOAP voor Windows naar Windows-communicatie. Los daarvan kunnen we kiezen voor een HTTP-kanaal voor standaard WS-I communicatie of andere kanalen met bijvoorbeeld MSMQ- of SQL Server Service Broker-ondersteuning. Zie afbeelding 1 voor de mogelijkheden van integratie met Windows Communication Foundation.



Afbeelding 1. Directe integratie met Windows Communication Foundation

WCF heeft de voorkeur in de volgende scenario's:

- Directe webservice-integratie tussen Windows en een applicatie op een ander platform.
- Directe communicatie tussen twee Windows-applicaties.
- Indirecte communicatie tussen twee Windows-applicaties via WCF over MSMQ of WCF over SQL Server Broker. WCF zorgt hierbij voor de interface voor zowel de directe als de indirecte integratie. Op de indirecte integratie kom ik later in dit artikel terug.

De inzet van WCF biedt ons als voordeel dat we gebruik kunnen maken van een uniform ontwikkelmodel: we bouwen onze applicatie met Visual Studio 2005. Dat er onderliggend van communicatiemedium wordt gewisseld is een kwestie van configureren en daarmee voor ons als ontwikkelaars dus minder van belang. Hierover meer als we het over applicatie-integratie via een wachtrij hebben.

Applicatie-integratie via een wachtrij

Directe integratie is niet altijd de beste keuze: de vraag is of de externe service die we aanroepen wel kan voldoen aan de beschikbaarheidseisen van onze applicatie op het moment dat we de aanroep doen. Die onzekerheid voorkomen we wanneer we koppelen door het sturen van boodschappen via een wachtrij (queue). Boodschappen blijven in de wachtrij staan tot het doelsysteem ze kan oppakken, waarna het resultaat teruggestuurd wordt naar het bronsysteem via hetzelfde mechanisme. Op deze manier wordt de koppeling asynchroon gemaakt. De aanroep is nog steeds direct gekoppeld via de interface, maar losgekoppeld in tijd. Bij dit scenario kennen we de volgende opties: MSMQ/System.Messaging, SQL Service Broker (SSB) en Windows Communication Foundation.

MSMQ/System.Messaging

Microsoft Message Queuing (MSMQ) is ingebouwd in Windows. De System.Messages-namespace maakt gebruik van MSMQ. Zo is het vrij eenvoudig om asynchrone koppelingen op te zetten; zie afbeelding 2. MSMQ heeft geen notie van sessie. Het kan in-memory werken, wat een goede performance oplevert, maar waarbij het gevaar bestaat dat boodschappen verloren raken bij een servercrash. Zie codevoorbeeld 1 voor een voorbeeld van het verzenden van een boodschap via System.Messaging.

Wanneer gebruiken we MSMQ?

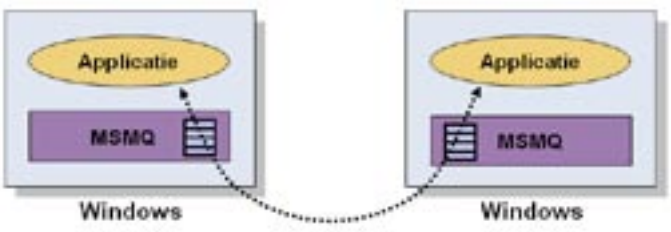
- Als er asynchrone communicatie nodig is tussen twee Windows-applicaties.
- Als zender en ontvanger niet altijd gelijktijdig in de lucht zijn.
- Als er extra functionaliteit nodig is voor de uitgewisselde boodschappen, zoals logging.

SQL Service Broker (SSB)

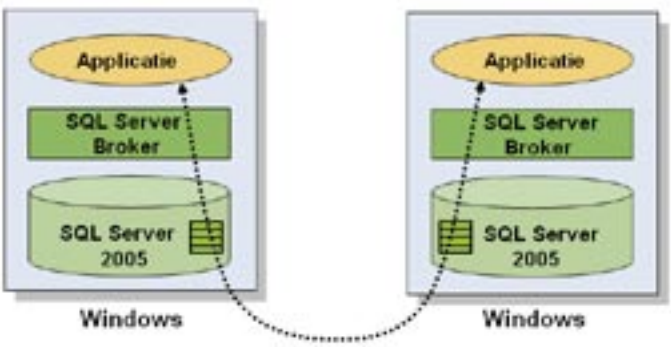
Waarom op het OS of een ander extern mechanisme vertrouwen als je al SQL Server hebt? SQL Server Service Broker stelt je in staat 'conversaties' op te stellen tussen SQL Server 2005-machines. SSB is in alle versies van SQL 2005 aanwezig, maar voor het opzetten van een conversatie moet ten minste één van de twee servers een versie zijn die in licentie is. Zie afbeelding 3 voor integratie via SQL Service Broker. Er worden intern databasetabellen gebruikt voor het wachtrijmechanisme, waardoor je gebruik kunt maken van SQL-transacties, back-up en restore. De boodschappen worden intern opgeslagen in een tabel. Dat maakt het lastig de conversatie te volgen. Er is wel een tooltje te vinden om dit te vergemakkelijken: http://www.sqljunkies.com/WebLog/nielsb/archive/2005/12/27/17701.aspx Voor het implementeren van SSB is TSQL uitgebreid. Zie codevoorbeeld 2 voor code die hetzelfde doet als het MSMQ-scenario, maar dan in een geheel andere taal en in je database in plaats van in je applicatie.

```
using System;
using System.Messaging;

[assembly: CLSCompliant(true)]
namespace ConsoleApplicationMSMQSend
{
  class Program
  {
    static void Main(string[] args)
    {
      MessageQueue messageQueue = null;

      try
      {
        string messageQueuePath = @".\private$\MSMQSample";
        if (System.Messaging.MessageQueue.Exists(messageQueuePath))
        {
          messageQueue = new MessageQueue(messageQueuePath);
        }
        else
        {
          messageQueue = MessageQueue.Create(messageQueuePath);
        }

        Message message = new Message("Goedemiddag",
             new System.Messaging.ActiveXMessageFormatter());
        messageQueue.Send(message);

      }
      catch (Exception ex)
      {
        // ExceptionManager.Publish(ex);
        throw ex;
      }

      finally
      {
        messageQueue.Close();
      }
    }
  }
}
```

Codevoorbeeld 1.
Boodschappen versturen via een wachtrij met System.Messaging



Afbeelding 2. Integratie via MSMQ



Afbeelding 3. SQL Server Service Broker

```
/* Maak twee queues Send en Receive*/
CREATE QUEUE GoedemorgenSQ
CREATE QUEUE GoedemorgenRQ

/* Definieer een message type */
CREATE MESSAGE TYPE Groet

/* Maak het contract voor de message */
CREATE CONTRACT GroetContract(Groet SENT BY ANY)

/* Zorg dat er ook services sturen en luisteren */
CREATE SERVICE GroetZendService ON QUEUE GoedemorgenSQ
CREATE SERVICE GroetOntvangService ON QUEUE GoedemorgenRQ (GroetContract)

/* Ga praten */
DECLARE @conversatie uniqueidentifier
/* zet de dialoog op */
BEGIN DIALOG CONVERSATION @conversatie
   FROM SERVICE GroetZendService
   TO SERVICE 'GroetOntvangService'
   ON CONTRACT GroetContract
WITH ENCRYPTION = OFF
;

   /* stuur het bericht */
   SEND ON CONVERSATION @conversatie MESSAGE TYPE Groet ('Goedemorgen')
END CONVERSATION @conversatie WITH CLEANUP;

/* bekijken berichten op dit moment in de queue via SSB Admin:
   http://www.sqljunkies.com/WebLog/nielsb/archive/2005/12/27/17701.aspx */

/* haal de queue leeg */
/* message_body is een keyword */
WAITFOR (RECEIVE TOP(1)  CAST(message_body as VARCHAR(255))
   FROM GoedemorgenRQ);
```

Codevoorbeeld 2.
Boodschappen versturen via een wachtrij met SSB

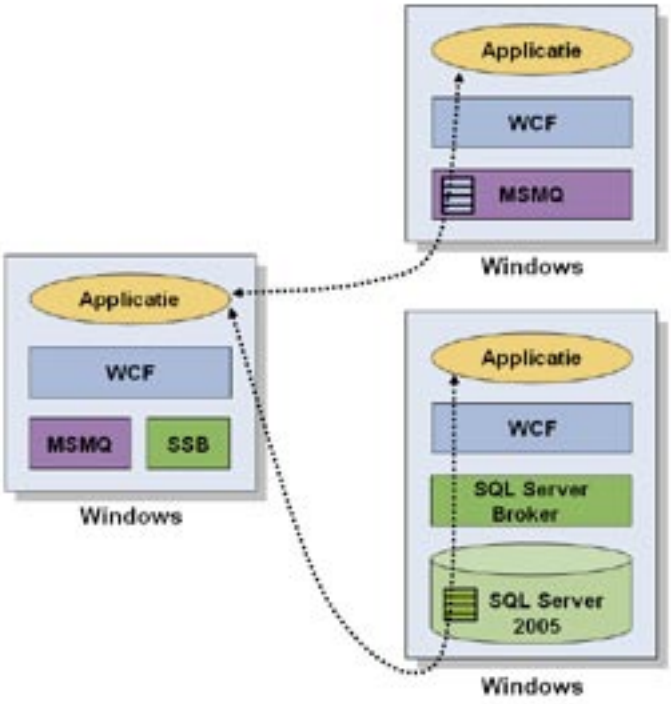
Wanneer gebruiken we SSB?

- Als de te koppelen applicaties geschreven zijn als Stored Procedures in T-SQL of in een taal die in de CLR-omgeving in SQL draait.
- Als businesslogica beheerd wordt door DBA's.

De vraag is of je uit architectuuroverwegingen wel wilt dat applicatielogica wordt gehost in SQL Server: applicatielogica hoort in de businesslaag en niet in de datalaag en dus in een Visual Studio-project. Daarnaast kost het gebruik van de CLR-omgeving in SQL Server behoorlijk wat performance. Businesslogica door DBA'ers laten beheren moet een bewuste keuze zijn.

Windows Communication Foundation

WCF kan gebruikmaken van MSMQ of SSB; zie afbeelding 4. Je kunt in WCF een applicatiekoppeling schrijven en vervolgens in de configuratie aangeven of je voor de communicatie MSMQ of SSB wilt gebruiken. WCF gebruikt hetzelfde programmeermodel voor communicatie met MSMQ en SSB, ondanks het feit dat de syntax voor MSMQ en SSB erg verschillend is. Ook voor synchrone RPC gebruikt WCF hetzelfde programmeermodel. Laten we als voorbeeld naar codevoorbeeld 3 kijken om te zien hoe we de hiervoor genoemde implementaties in WCF zouden doen. Het mooie is dat we de code niet meer hoeven aan te passen als we besluiten van MSMQ naar SSB te wisselen, maar dat is een kwestie van configuratie. Een stuk beter dan je Visual Studio-code zoals in codevoorbeeld 1 te moeten ombouwen naar T-SQL-code als in codevoorbeeld 2.



Afbeelding 4. Windows Communication Foundation: MSMQ en SSB agnostisch

Wanneer zou ik MSMQ of SSB dan toch nog los gebruiken van WCF?

- Niet alle functionaliteit van MSMQ is beschikbaar in WCF: je mist het bekijken van verzonden boodschappen in de wachtrij, het gebruik van journaling of queued ontvangstberichten, of het door de verzender laten kiezen van de wachtrij voor het antwoord.
- Een DBA'er zal zich wellicht beter thuis voelen in SQL Server voor het gebruiken van alle SSB-functionaliteit. Dat is vooral een overweging waard als het beheer van je applicatie onder de verantwoordelijkheid van DBA'ers valt.

Applicatie-integratie via een broker

Tot nu toe hebben we gezien dat we applicaties direct kunnen koppelen of ze qua tijd kunnen loskoppelen door een wachtrijmechanisme te implementeren, maar het is ook mogelijk om een broker in te zetten. Dit is zinnig wanneer verschillende applicaties met elkaar moeten communiceren. In plaats van elke applicatie met alle andere applicaties afzonderlijk te laten praten, laat je ze via de broker met elkaar communiceren. Per koppeling is dan maar één interface nodig, namelijk naar de broker. De broker is als het ware de spin in het web van applicaties. De broker kan het communicatieprotocol loskoppelen van het message-formaat, assisteren in de conversie van verschillende formaten naar een gemeenschappelijk formaat en als platform dienen voor het businessproces. Een broker kan namelijk ook zelf businesslogica bevatten. In de praktijk zal dit de businesslogica zijn die gebruikt wordt voor het uitwisselen van boodschappen; niet de businesslogica van de applicaties zelf. In dat perspectief is de naam SQL Server Service Broker wat ongelukkig gekozen. Bij een broker is het communicatieprotocol los te koppelen van het message-formaat, terwijl bij SSB het datatype en de interface al vast staan. Er kan slechts gekoppeld worden met een andere SQL 2005-instantie, er is geen sprake van procesondersteuning: de naam SQL Server Message Queue zou beter passen. Wat zijn de keuzes voor applicatie-integratie via een broker?

Eigen broker

Soms lijkt het eenvoudiger om zelf een broker te ontwikkelen. Dit is bijvoorbeeld het geval als de hoeveelheid functionaliteit in BizTalk Server 2006 niet nodig is, of wanneer er goede redenen zijn om aan te nemen dat het beheer van BizTalk in de organisatie van de klant problemen zal opleveren vanwege de complexiteit. Als ontwikkelaar moet je sterk in je schoenen staan wil je deze optie

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;
using System.Transactions;

[assembly: CLSCompliant(true)]
namespace ConsoleApplicationWCF
{
  class Program
  {
    static void Main(string[] args)
    {
      //maak de service
      ServiceHost<Goedemorgen> host = new ServiceHost<Goedemorgen>();
      try
      {
        //open zoewl de directe binding als de wachtrijbinding
        host.Open();

        //directe aanroep
        GoedemorgenProxy p1 = new GoedemorgenProxy("WS");
        p1.ZegGoedemorgen();

        Console.WriteLine("directe aanroep via webservice");
        Console.ReadLine();

        //exact zelfde via wachtrijbinding, transactie toegevoegd
        using (TransactionScope ts = new TransactionScope())
        {
          GoedemorgenProxy p2 = new GoedemorgenProxy("Queue");
          p2.ZegGoedemorgen();
          ts.Complete();

          //via wachtrij
          Console.WriteLine("aanroep via wachtrij");
          Console.ReadLine();
        }
      }
      catch (Exception ex)
      {
        //  ExceptionManager.Publish(ex);
        throw ex;
      }
      finally {
           host.close();
      }
    }
  }
}
```

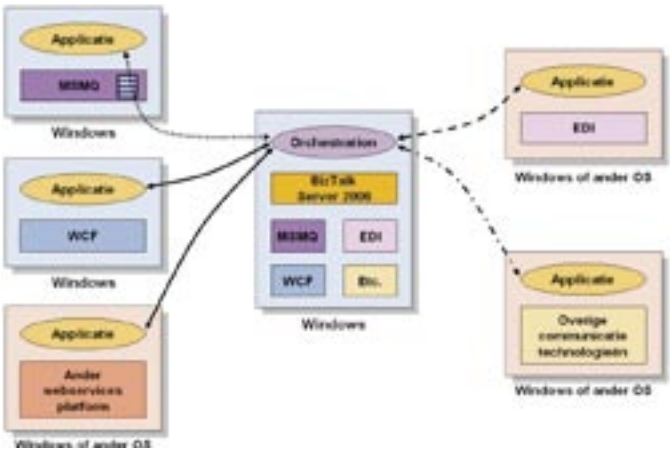
Codevoorbeeld 3.
Boodschappen versturen via Windows Communication Foundation

succesvol verdedigen. Zo duur zijn de licenties voor de standaard versie van BizTalk namelijk niet.

BizTalk Server 2006

BizTalk Server 2006 kan (net als BizTalk Server 2004) als integratieplatform fungeren; zie afbeelding 5. Er zijn adapters beschikbaar voor verschillende communicatiemechanismen en -formaten (WCF, MSMQ, FTP, HTTP, EDI, XML). Maar het is ook mogelijk eigen adapters te ontwikkelen, of gebruik te maken van extern ontwikkelde adapters zoals de EDI-adapter van Covast. Naast adapters kent BizTalk Server 2006 ook extra integratieservices:

- Het ontwerpen van het businessproces aan de hand van 'orchestrations' die diverse externe applicaties kunnen aansturen en gebruik kunnen maken van de services voor state management en langlopende transacties.



Afbeelding 5. BizTalk Server 2006: spin in het web

- Het ontwerpen van XML-schema's voor berichten, en de transformaties van berichten.
- Rapportage over het proces met Business Activity Monitoring. Dit levert voor managers mooie rapportages op.
- Een Business Rules-engine voor het centraal beheren van complexe regels.

Wanneer gebruiken we BizTalk Server?

- Als een groot aantal te koppelen applicaties wordt verwacht.
- Bij verwachte wisselingen van berichtstructuur of communicatiemechanisme.
- Als er een integratieproces ingericht moet worden, met transacties die uren tot weken gaan duren of met ingewikkelde businesslogica.
- Als er met verschillende externe partners gekoppeld moeten worden, eventueel met eisen als het gebruik van standaarden als EDI of Rosettanet.

Een veelvoorkomende fout is het gebruik van BizTalk Server als een database-importtool. Dit kun je vergelijken door met een gereedschapskist een spijker in de muur te slaan. Dat lukt uiteindelijk wel, maar de kist (lees: BizTalk Server) is bedoeld om spijkers in te vervoeren, niet om er spijkers mee in de muur te slaan. Op het moment dat dit nodig is pak je de hamer (lees: SQL Server Integration Services (SSIS) of de voorloper DTS), zeker als je wilt dat het snel gebeurt. In dit laatste geval hebben we het over de integratie van data.

Integratie van data

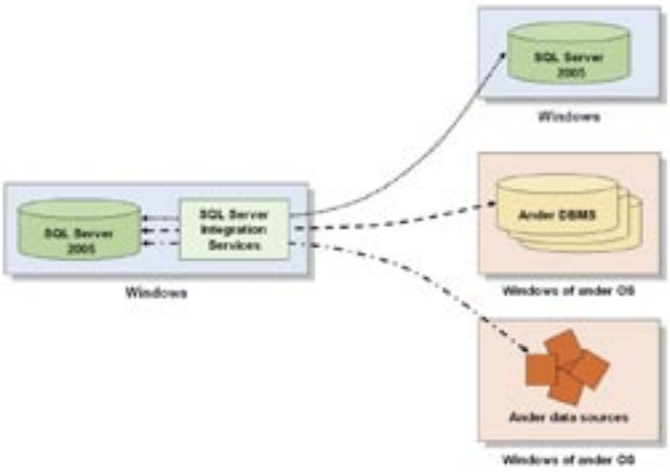
We hebben hiervoor gesproken over het koppelen van applicatielogica. In andere gevallen ligt de focus van het integratieproject op de applicatiedata. Data zijn van zichzelf niet intelligent. Het integreren van data komt neer op het heen en weer schuiven van passieve informatie, tussen dataopslagsystemen, los van de businesslogica. Vandaar dat we de twee data-integratietools van Microsoft terugvinden in SQL Server: SQL Server Integration Services en SQL Replicatie.

SQL Server Integration Services

SQL Server Integration Services (SISS), de opvolger van SQL Data Transformation Services (DTS), geeft ons gereedschap om data uit diverse bronnen te extraheren, te transformeren en in te laden (ETL). Daarnaast biedt SSIS mogelijkheden gegevens op te schonen en fouten gestructureerd af te handelen.

Wanneer gebruiken we SSIS?

- Voor initiële bulkload van historische gegevens.
- Bij hoge performance-eisen of grote hoeveelheden te importeren data in flat file of een ander formaat zoals Excel.
- Als data uit diverse bronnen (flat file, spreadsheets) of DBMS'en samengevoegd moet worden.



Afbeelding 6. SQL Server Integration Services

- Voor het opzetten van een datawarehouse voor data mining of voor online analytical processing (OLAP).
- Voor het omzetten van het ene naar het andere DBMS.

SSIS of BizTalk: sloot of druppel?

Het is goed mogelijk om in BizTalk een orchestration te definiëren waarmee diverse databronnen worden geraadpleegd om aan de hand daarvan een database te vullen. BizTalk is echter vooral goed in het faciliteren van real time-communicatie voor het koppelen van applicatielogica of voor communicatie tussen zakenpartners. SISS echter is geoptimaliseerd voor het importeren van grote hoeveelheden data. Om een andere vergelijking te maken: BTS is geschikt voor druppelsgewijze verwerking voor het koppelen van applicatielogica, SSIS voor het integreren van een hele stroom applicatiedata ineens. Naar mijn mening is het gebruikersgemak van de BizTalk Mapper één van de oorzaken voor het veelvoorkomende 'misbruik' van BizTalk als importeer-tool. Als je weet wat je doet, kun je in betrekkelijk weinig tijd op heel inzichtelijke wijze het ene formaat in het andere formaat omzetten. Met de SQL Adapter kan het in theorie dan zo de database in. De functionaliteit van de mapper is goed, maar je gebruikt het voor het omzetten van bulkdata en daar is BizTalk niet geschikt voor. Gelukkig hebben we sinds SQL2005 SSIS met daarin een soortgelijke Mapper.

SQL Server Replicatie

We hebben gezegd dat SISS geschikt is voor het importen en exporteren van grote hoeveelheden data. Als dit een één-op-één kopie moet opleveren, is SQL Server Replicatie echter een betere optie; zie afbeelding 7. Replicatie zorgt ervoor dat dezelfde data, of een subset van de data, op verschillende servers beschikbaar zijn. SQL Server Replicatie heeft onder meer een user-interface om aan te geven welke tabellen (en welke kolommen van die tabellen) gerepliceerd dienen te worden, wat het verschil tussen de tabellen is en welke conflicten zijn opgetreden tijdens de replicatie. Met replicatie worden alleen gewijzigde of toegevoegde tabelregels doorgegeven aan de andere SQL Servers: dit maakt het bijna real-time en stukken sneller dan bijvoorbeeld een tabel legen en opnieuw vullen met alle data, zoals dat via SSIS zou worden geïmplementeerd.

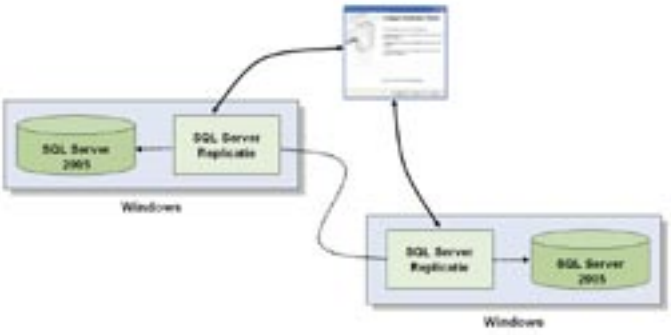
Wanneer gebruiken we replicatie?

- Als er behoefte is aan identieke data op verschillende SQL-instanties. Denk hierbij ook aan SQL Server CE als middel om mobiele applicaties bijgewerkt te houden.
- Om SQL Server als bron voor IBM- of Oracle-databases te gebruiken.
- Om Oracle als bron voor een SQL Server-, IBM- of Oracle-doeldatabase te gebruiken (waarbij in de laatste twee gevallen SQL Server als tussenpersoon wordt gebruikt).

SQL Replicatie zorgt voor het gelijkhouden van data op tabelniveau, met de mogelijkheid om te kiezen welke kolommen van een tabel gerepliceerd worden. Het datatype blijft hetzelfde. Als er wijzigingen in het datatype of transformaties nodig zijn, is SSIS

Integratiestijl	Technologie	Scenario
Directe applicatie-integratie	ASP.NET Web Services (ASMX)	Het verbinden van Windows-applicaties met Windows- en non-Windows-applicaties via SOAP.
	.NET Remoting	Het via gedistribueerde objecten verbinden van Windows-applicaties.
	Enterprise Services	Het integreren van Windows-applicaties die gebruikmaken van gedistribueerde transacties en object lifetime management, etc.
	Windows Communication Foundation (Indigo)	Het integreren van Windows- en non-Windows-applicaties, gebruikmakend van webservices, gedistribueerde transacties, lifetime management, etc. (opvolger van ASMX,.NET Remoting en Enterprise Services) met een uniform programmeermodel.
Applicatie-integratie via een wachtrij	Microsoft Message Queuing	Het verbinden van Windows-applicaties met Windows- applicaties via een wachtrij.
	SQL Service Broker	Het verbinden van SQL Server 2005-applicaties met andere SQL Server 2005-applicaties via een wachtrij.
	Windows Communication Foundation	Het verbinden van Windows-applicaties via een wachtrij (via MSMQ en/of SSB) met een uniform programmeermodel.
Integreren van applicaties via een broker	BizTalk Server 2006	Windows- en niet-Windows-applicaties verbinden met gebruikmaking van diverse protocollen. Omzettingen opzetten tussen verschillende boodschapformaten. Businessprocessen besturen via uitgetekende orchestrations. Voor het aanbieden van Business process services zoals Business Activity Monitoring en de Business Rules Engine. Businesspartners verbinden met gebruikmaking van industriestandaarden zoals RosettaNet.
Het integreren van data	SQL Server Integration Services	Het combineren en transformeren van verschillende databronnen naar SQL Server 2005-data.
	SQL Server Replication	Het synchroniseren van SQL Server-data met kopieën van die data in andere instanties van SQL Server, Oracle of DB2.
Integratie met applicaties en data op IBM-systemen	Host Integration Server 2004	Windows-applicaties verbinden met IBM zSeries- en iSeries- applicaties en data MSMQ verbinden met IBM WebSphere MQ.

Tabel 1. Overzicht met technologieën



Afbeelding 7. SQL Server replicatie

een betere keuze. SSIS blijft echter een bulkgeoriënteerde technologie: er is geen mogelijkheid dat per tabelregel te doen. SSIS is dus voor het integreren van uiteenlopende data, replicatie is voor identieke data. Toch heeft ook replicatie beperkingen: het is lastig om te gaan met identity-kolommen. Er is een limiet van maximaal 246 kolommen in een te repliceren tabel, maar als dat probleem optreedt, wijst dat op een suboptimaal datamodel (en dat terwijl Active Directory juist voor de snelheid met een platgeslagen formaat werkt).

Host Integration Server 2004

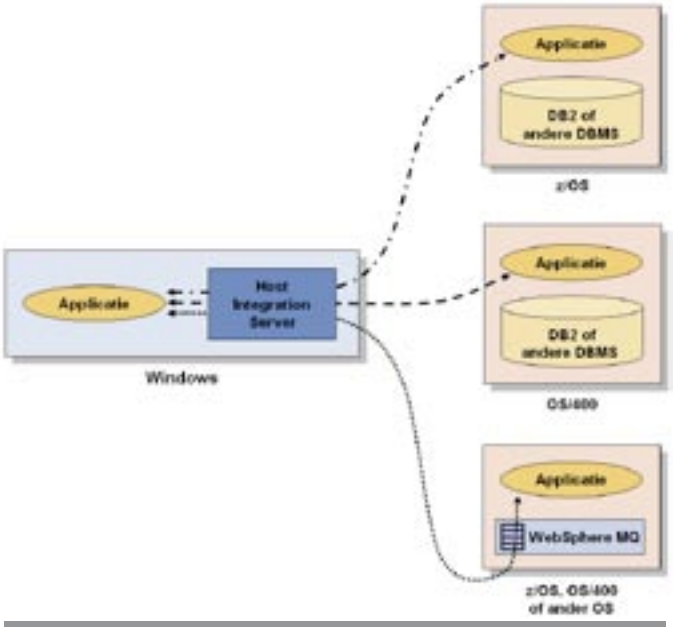
Voor de volledigheid is het goed om te weten dat Microsoft ook een product heeft voor het integreren van zowel applicaties als data op IBM-systemen: Host Integration Server (HIS). Zie ook afbeelding 8 voor de integratiemogelijkheden van Host Integration Server 2004.

Wanneer gebruiken we HIS:

- bij directe applicatiekoppeling met IBM-mainframes (z/OS) of midrange- systemen (OS/400).
- bij asynchrone applicatiekoppeling MSMQ met Websphere MQ (voorheen MQSeries).
- bij koppeling met CICS/IMS-applicaties, direct of via webservices.
- bij integratie van data in bijvoorbeeld VSAM of DB2.

De kracht van de keuze

Microsoft biedt ons nogal wat mogelijkheden om applicaties en/of data te integreren. Het is goed om te zien dat deze technologieën convergeren naar een eenvormige set van tools op basis van het .NET Framework, allemaal benaderbaar vanuit Visual Studio 2005. In tabel 1 zijn deze technologieën en de hoofdscenario's samengevat. Aan de hand van het hoofdscenario kun je bepalen wat in jouw geval de meest voor de hand liggende oplossingsrichting is. Voor ieder wat wils, zou ik zeggen, maar juist in de omvang van het platform ligt de uitdaging voor de ontwikkelaar. In dit artikel heb ik een overzicht gegeven van de verschillende alternatieven en van de situaties waarin je ze wel of juist beter niet kunt inzetten. Aan jou nu de keuze, want juist op een weloverwogen, strategische beslissing op grond van argumenten worden de krachtigste integratieoplossingen gebouwd.



Afbeelding 8. Host Integration Server

Edwin Smit is Technisch Architect bij Qurius ETX (www.quirusetx.nl) en bereikbaar via edwins@quriusetx.nl

Referenties

Voor het artikel is gebruik gemaakt van het white paper over integratie, http://msdn.microsoft.com/library/en-us/bts_2004wp/html/14bc36a8-69a9-48ed-8e4c-1c85202544c0.asp

Een goede presentatie over hetzelfde onderwerp maar meer gefocust op messaging versus data: Scott Woodgate, MSDN Architecture Webcast: Microsoft Integration Technologies: When to Use What: http://msevents.microsoft.com/cui/WebCastEventDetails.aspx?EventID=1032281670&EventCategory=5&culture=en-US&CountryCode=US)

Introductie Windows Communication Foundation: http://msdn.microsoft.com/library/en-us/dnlong/html/introtowcf.asp