Ontwikkeling van mobiele toepassingen met SQL Anywhere 11

Sybase UltraLite en QAnywhere in de praktijk

Frans Kouwenhoven

Sinds jaar en dag levert Sybase enterprise-size databases, de laatste versie is de vorig jaar gelanceerde SQL Anywhere versie 11. Van de bijna tweehonderd nieuwe mogelijkheden in deze versie zijn full-text search en regular expressions het meest in het oog springend. In dit artikel wordt beschreven wat er zoal komt kijken bij het bouwen van een real-life mobiele applicatie gebaseerd op een krachtige minidatabase mét berichtenverkeer.

Naast een krachtige database levert Sybase standaard een uitgebreide set tooling mee. Vanuit Sybase Central zijn allerhande handelingen op de databases uit te voeren, zoals creëren en aanpassen, wizards voor deployment en een query tool (Interactive SQL). Vanuit Central zijn ook de andere megeleverde producten (zoals UltraLite en QAnywhere die hieronder worden beschreven) te onderhouden.
Voor Visual Studio heeft Sybase plugins zoals een explorer window om de database in te zien en *drag-n-drop* van tabellen in de code. Hierbij wordt de connectie en SELECT query op deze tabel automatisch in code gegenereerd. Men moet hier geen wonderen van verwachten, maar het werkt wel.
Het voordeel van SQL Anywhere is het cross-platform karakter. Alle producten en tools zijn te gebruiken op de meest uiteenlopende besturingssystemen. Dit is erg handig als je ontwikkelt op Windows en uitrolt op Linux.

Mobiele toepassingen

Waarmee Sybase met afstand op kop loopt, is het gemak van het ontwikkelen van mobiele toepassingen. De krachtigste tooling hierbij is het kunnen synchroniseren van server- en mobiele databases: MobiLink. Het maakt voor MobiLink niet uit welke database er op de server draait: SQL Anywhere, MS-SQL, Oracle, MySQL enzovoort.
MobiLink werkt op basis van scripts die voor de tweeweg synchronisatie zorgen. In versie 11 zijn nieuwe wizards toegevoegd in Sybase Central die het ontwikkelen van deze scripts faciliteren. Maar voor het betere tweak werk kunnen de gegenereerde scripts nog aangepast worden. De praktijk leert dat ontwikkelaars graag die controle willen houden. Het formaat van de scripts maakt dat ze prima leesbaar en aanpasbaar zijn.

Ondanks deze tooling blijft het opletten. We hebben situaties gehad dat MobiLink een foutloze synchronisatie uitvoerde, maar dat toch de gewenste rijen niet in de mobiele database verschenen. Na veel speurwerk bleek dat bij de insert van de rijen op de mobiele database (dat verzorgt MobiLink) er een *violation* optrad vanwege een *foreign key* verwijzing naar een niet-bestaande rij. Dit valt MobiLink niet te verwijten, maar het blijft riskant!
MobiLink is heel transparant. We noemden zojuist de onafhankelijkheid van de database, een ander goed voorbeeld is het gebruik van MobiLink's achterdeur: vanuit verschillende stappen in het synchronisatieproces zijn uitstapjes te maken naar .NET- of Java-code. Wij gebruiken bijvoorbeeld deze achterdeur om MobiLink in te laten haken in ons standaard authorisatiemechanisme. Bij de eerste aanroep van het mobiele device wordt door ons het device (of de gebruiker) uniek geïdentificeerd en bepalen we of dit device (of gebruiker) toegang mag krijgen tot de server. In het MobiLink script staat aangegeven welke class moet worden aangeroepen. De returnwaarde van deze class geeft vervolgens aan of MobiLink de synchronisatie mag vervolgen of dat deze wordt afgebroken. Treedt er in de class een fout of een waarschuwing op, dan kan met de foutcode die wordt teruggestuurd in de applicatie netjes een afhandeling plaatsvinden.

Een punt waarin MobiLink wel kan verbeteren is de mogelijkheid om de synchronisatiescripts te kunnen debuggen. Momenteel moeten we het doen met de standaard logging die MobiLink genereert. Deze logging kan dan wel op heel uitgebreid gezet worden, als verwende programmeur wil je eigenlijk het mechanisme van *breakpoints* hebben. We hebben goede hoop dat in komende versies van MobiLink hier een flinke stap in gemaakt wordt.

Groot, kleiner, kleinst

Vanwege het cross-platform karakter van de producten is het laten draaien van een SQL Anywhere (SA) database op een server of mobiel device even eenvoudig: het is namelijk hetzelfde product. Dit is handig, maar hiermee heb je natuurlijk nog geen mobiele applicatie. De keerzijde van het hebben van een enterprise-size database op je mobiele device is natuurlijk direct de footprint, maar ook het feit dat de database engine als apart proces draait. Omdat je wilt dat je lang op een volle batterij kan werken wil je de engine alleen gestart hebben op het moment

dat je applicatie draait. Een engine in hetzelfde proces vereenvoudigt dit principe.
Voor mobiele toepassingen wil je vaak geen SA database op je device draaien: dat is overkill wat betreft functionaliteit én footprint. Sybase heeft een sterk alternatief: UltraLite. Zoals de naam al suggereert heeft deze database een kleine footprint (kleiner dan 500 KB) met behoud van veel van de functionaliteit van de SQL Anywhere database. Wat UltraLite daarvoor moet missen zijn bijvoorbeeld triggers, stored procedures en views, maar ook de standaard ingebouwde webserver die SA wel kent. Aan de andere kant heeft UltraLite een ingebouwde synchronisatie die communiceert met MobiLink op de server. Ook draait UltraLite in hetzelfde proces als de applicatie.
Programmeren met UltraLite is heel eenvoudig. Voor .NET-ontwikkeling is de ADO.NET interfacing geïmplementeerd. Voor wie vertrouwd is met ADO.NET is de overstap naar UltraLite heel eenvoudig.

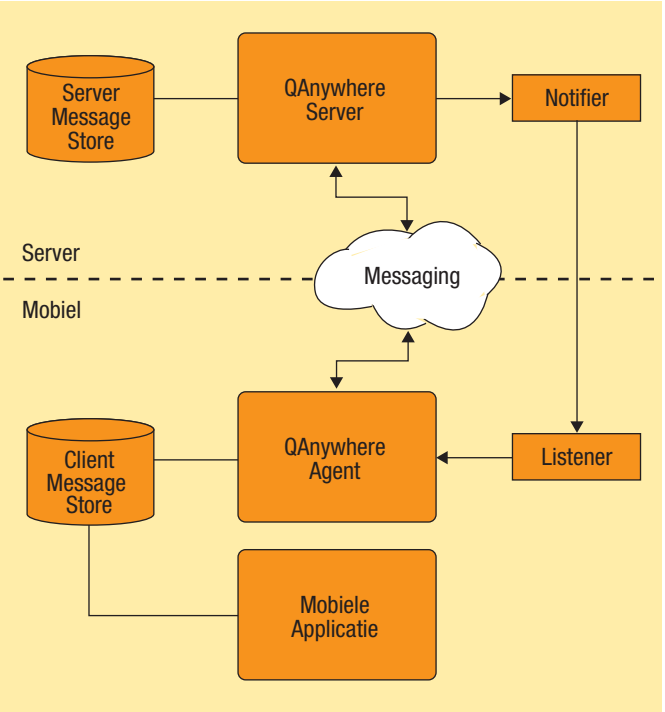
```
ULConnection conn =
   new ULConnection("dbf=MyDB.udb;uid=dba;pwd=sql");
conn.Open();

ULCommand cmd = conn.CreateCommand();
cmd.Command = "INSERT INTO MyTable(MyColumn)
                                        values (?)";
cmd.Parameters.add("", someValue);

int rowsInserted = cmd.ExecuteNonQuery();
```

Ook voor C en C++ zijn er API's beschikbaar.

Voor Java-gebaseerde systemen heeft Sybase UltraLiteJ ontwikkeld. Dit is niet een full-fledged relationele database, maar het biedt een heel goed alternatief voor de lastige low-level RMS die standaard onder J2ME te vinden is. Voor ontwikkeling op smartphones (zoals bijvoorbeeld de BlackBerry) is UltraLiteJ een uitkomst. Vanwege het cross-platform karakter is de database ook op J2SE systemen te gebruiken. Dat opent bijvoorbeeld de deur naar Android bijvoorbeeld.

QAnywhere

Voor het synchroniseren van een server- en mobiele database is MobiLink dé aangewezen tool. Heeft de informatie meer het karakter van berichten, dan is er een beter mechanisme voorhanden: QAnywhere. Dit is een applicatie-naar-applicatie systeem voor het versturen van berichten tussen mobiele devices onderling en van en naar een server.
QAnywhere heeft standaard aansluitingen naar systemen die de JMS-standaard ondersteunen, zoals JBoss JMS en WebSphere MQ. Microsoft's MQ heeft geen ondersteuning voor JMS, maar er zijn bridging tools verkrijgbaar voor MS-MQ naar JMS.
Men kan QAnywhere (QA) beschouwen als een adapter die het MQ-systeem van de server doortrekt naar mobiele devices.



Afbeelding 1: QAnywhere: schematische weergave van messaging van en naar een mobiele applicatie.

Daarbij worden de standaard mobiele uitdagingen (slechte dekking, kleine bandbreedte, security en dergelijke) opgelost. QA werkt volgens het store-and-foreward principe met gegarandeerde aflevering. Als een mobiel device niet verbonden is dan worden de berichten opgeslagen, zowel lokaal (te verzenden) als op de server (te ontvangen).
Het is mogelijk om meerdere berichtkanalen te gebruiken; bij elk kanaal kunnen weer andere devices aangesloten zijn, reageert een andere applicatie op het inkomende bericht of reageert dezelfde applicatie op een andere manier. Ook is het mogelijk om berichten te versturen tussen mobiele devices. Hierbij wordt de centrale server gebruikt als relay om te voorkomen dat de mobiele devices tegelijkertijd connected moeten zijn.

De meest eenvoudige configuratie van QA is die waar mobiele devices berichten kunnen versturen, en kunnen ontvangen op basis van policy's. Deze policy's kunnen bijvoorbeeld een interval zijn, een statusverandering van een bericht, of (en dit is echt een heel mooie optie) het moment dat het type verbinding wijzigt. Ga je van GRPS naar Wifi? Verstuur je berichten en check je inbox!
Ik denk dat bovenstaande configuratie voor de meeste toepassingen ontoereikend is, omdat je op commando berichten naar een mobiel device wil kunnen sturen. Voor deze message push zal je de configuratie moeten uitbreiden met listeners (op mobiele devices) en een notifier (op de server). MobiLink neemt het notificeren voor zijn rekening (opstarten met de -m optie). Op de server is tevens een Server Message Store nodig, een lokale database waarin de berichten worden gecached. Vanuit Sybase Central is

deze SMS te maken in een bestaande of een nieuwe database. Sybase levert echter geen scripts om het omgekeerde te bewerkstelligen: het verwijderen van de tabellen en triggers van QAnywhere. Omdat wij hier wel behoefte aan hadden hebben we deze scripts zelf ontwikkeld, door ons flauw gedoopt als qnowhere.sql.
Op de mobiele devices dient zowel een QA agent als een listener gestart te worden. De agent moet wel als een apart proces worden opgestart. De reden hiervoor is dat in principe meerdere applicaties van dezelfde agent gebruik kunnen maken.
Het opstarten van de listener hoeft niet handmatig, door het instantiëren van het object MessageListener gebeurt dit door QAnywhere zelf (zie C# voorbeeld);

```
receiveListener = new MessageListener(onMessage);
```

waar de methode 'onMessage' de afhandeling verzorgt van het binnengekomen bericht.
Spin in het berichtenweb is de QA Manager die de koppeling tussen de applicatie en QA onderhoudt:

```
QAManager qaManager =
        QAManagerFactory.Instance.CreateQAManager();
qaManager.SetMessageListener("ExampleQueue",
                                    receiveListener);
```

Het verzenden van een bericht gaat al even eenvoudig;

```
qaManager.PutMessage("ServerID\\ExampleQueue",
                                            myMessage);
```

waarbij myMessage zowel tekst als binary kan zijn.

Het mobiele device heeft 'onder water' ook een lokale cache-database nodig: de Client Message Store. Voor het eerste gebruik dient deze database gecreëerd te worden en voorzien van de unieke client-id. Deze client-id is hierna niet meer te wijzigen. Dit creëren van de CMS is omslachtig en niet echt nodig. Liever zie ik de mogelijkheid om bij het opstarten van QA agent de optie 'wel of niet een CMS aanmaken indien geen beschikbaar' mee te sturen.
Werkt de gewenste configuratie, dan is ook meteen gebruik te maken van het berichtenverkeer en kom je er achter hoe eenvoudig dit systeem in de praktijk werkt. Doordat het push-mechanisme werkt op basis van UDP (kleine pakketjes), hoef je niet bang te zijn voor torenhoge rekeningen van je GPRS-provider.
Sybase heeft vrij onopgemerkt het systeem van QA uitgebreid met een heel mooi feature: het mobiel gebruik van webservices. Door via QA webservices aan te spreken kun je gebruik maken van gegarandeerde aflevering van berichten. Voor de developers zijn standaard tools meegeleverd die WSDL omzetten in proxy classes (voor zowel .NET als Java).

De praktijk

Voor het ontwikkelen van een field service-oplossing voor een klant waren de volgende requirements nodig: een enterprise-size database op Linux-servers, synchronisatie naar duizenden PDA's in het veld en het kunnen versturen en ontvangen van urgente berichten. Omdat op dat moment SA versie 11 nog niet officieel uitgekomen was, stonden wij voor de keuze: gaan we voor versie 10 die zich reeds bewezen heeft of durven we de sprong naar 11 aan ondanks dat deze nog een bèta-versie is?
Een goede reden om toch voor 11 te kiezen was het feit dat hierin QAnywhere in UltraLite is geïntegreerd. Met de wetenschap dat support van Sybase goed en snel is, hebben we op de koop toe genomen met een bèta-product aan de slag te gaan. En natuurlijk lopen we tegen onvolkomenheden aan. Zoals voorbeelden die niet werken, of instellingen die niet duidelijk zijn. Maar er was één probleem waar we pas tijdens deployment achter kwamen: er is geen QA agent beschikbaar onder Linux! We dachten eerst dat Sybase deze nog even over het hoofd had gezien, maar na veel contact met de developers van Sybase weten we de echte reden: Sybase gaat er vanuit dat op de server de QA agent met een Java- of .NET-proces praat, maar wij gebruiken de SQL API om meteen informatie in en uit de database te halen. De QA agent gebruiken wij dus min of meer onbedoeld als 'mobiele agent op de server'.
Omdat we niet veel tijd meer over hadden voor een alternatief, zijn we creatief aan de slag gegaan en hebben we één van de weinige Windows-systemen in ons serverpark gebruikt om de QA agent te installeren. Vanuit hier maken we een brug naar de database die wel op Linux draait.

Toekomst

Wij hebben ondanks het uitzoekwerk de applicatie op tijd kunnen opleveren aan de klant en het systeem werkt nog steeds naar behoren. Ondertussen heeft Sybase een nieuwe versie uitgebracht waarin reeds een verzoek van ons gehonoreerd is: de QA agent is geïntegreerd in hetzelfde proces als de applicatie.

Wat heeft Sybase voor de korte termijn in petto? Enkele maanden terug is SUP gelanceerd, het Sybase Unwired Platform. Dit platform is een paraplu waaronder veel van bovenstaande technieken naadloos met elkaar zijn geïntegreerd. Als een bonus is ook het device management en security tool toegevoegd (Afaria). Volledig nieuw en erg welkom is het gebruik van Mobile Business Objects, de mobiele variant van een Object-relational mapper (O/RM) zoals het Entity Framework van Microsoft. Hiermee kunnen met drag-n-drop de op het mobiele device gewenste tabellen aangegeven worden, waarmee 'onder water' de volledig werkende synchronisatiescripts worden gegenereerd. Dit platform schept verwachtingen.

Frans Kouwenhoven is mede-oprichter en technisch directeur van 2go-mobile en lid van het Customer Advisory Board van Sybase Europe.