Robert Tusveld is als softwarearchitect werkzaam bij Capgemini

CLR-integratie versus Transact-SQL

MAAK DE JUISTE KEUZE!

Met de komst van SQL Server 2005 is een groot aantal nieuwe functionaliteiten aan SQL Server toegevoegd. Een van de belangrijkste ontwikkelingen is dat de Common Language Runtime (CLR) een integraal onderdeel is geworden van de database-engine. Dit heeft tot gevolg dat databasefunctionaliteit voortaan eenvoudig in C# of in elke andere .NET-taal geprogrammeerd kan worden.

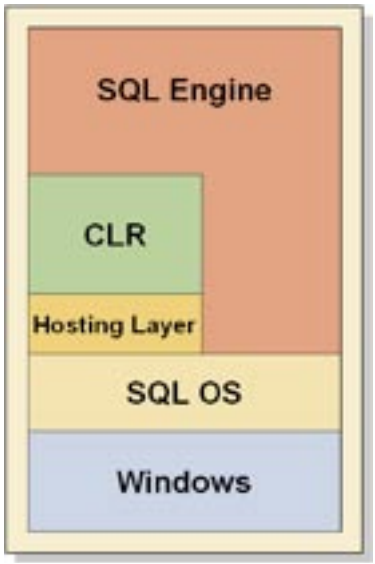
Bij het schrijven van stored procedures, triggers en door de gebruiker gedefinieerde functies in SQL Server 2005, moet de programmeur beslissen of traditiegetrouw gebruik wordt gemaakt van Transact-SQL (T-SQL) of van een .NET-taal. De uitkomst hangt af van de specifieke situatie die aan de orde is. In de ene situatie zal de ontwikkelaar T-SQL willen gebruiken, in de andere situatie juist managed code. In dit artikel wordt allereerst een overzicht gegeven van de voordelen van CLR-integratie en van de veranderingen in SQL Server 2005 met betrekking tot T-SQL. Daarnaast wordt beschreven op basis van welke criteria de ontwikkelaar een keuze kan maken tussen het gebruik van T-SQL of managed code.

CLR-integratie in SQL Server 2005

SQL Server 2005 is uitgerust met Microsoft .NET Framework versie 2.0; dit is dezelfde versie die ook gebruikt wordt door Visual Studio 2005. De CLR is de motor van het Microsoft .NET Framework, dat zorg draagt voor het uitvoeren van managed code. Met SQL Server 2005 wordt managed code een vast onderdeel in de SQL-engine. De CLR vertaalt de managed code naar instructies, die door de processor worden uitgevoerd. De architectuur van SQL Server en de opbouw in lagen wordt getoond in afbeelding 1.

In dit overzicht wordt de SQL Engine getoond die gebruikmaakt van het SQL Operating System (SQL OS) en van het Windows-platform. In de SQL-engine is de CLR aanwezig die samen met de hosting-layer zorgt voor het uitvoeren van managed code binnen SQL Server. De CLR is verantwoordelijk voor het op een efficiënte en veilige manier uitvoeren van applicaties. De CLR beheert daarom het geheugen, controleert of de applicatie wel voldoende rechten heeft om bepaalde acties uit te voeren en of er niets gedaan wordt waardoor het systeem instabiel kan worden. Voor het uitvoeren van deze taken maakt de CLR gebruik van de hosting-layer, die voor de uitvoering van de volgende taken zorgt:

- Assembly Loading
- Memory management
- Security Model
- Reliability
- Threads & Fibers
- Deadlock detection
- Execution context



Afbeelding 1: SQL Server-architectuur

De hosting-layer werkt bovenop het SQL-operating system en het Windows-operating system, zodat managed code niet direct met het operating system kan communiceren. Met CLR-integratie kunnen databaseontwikkelaars hun opgeslagen procedures, functies, triggers en door de gebruiker gedefinieerde typen coderen in de .NET-taal waar zij de voorkeur aan geven. Met Visual Basic .NET en C# beschikken zij over objectgeoriënteerde constructies, gestructureerde afhandeling van fouten, namespaces en klassen. Bovendien bevat het .NET Framework diverse .NET Base Class Libraries met allerlei klassen en methoden die eenvoudig aan de serverzijde kunnen worden gebruikt.

Het volgende overzicht toont de voordelen die de integratie van SQL Server 2005 en het .NET Framework biedt:

- *Verbeterd programmeermodel:* de .NET-talen zijn in veel opzichten rijker dan T-SQL. Ze bieden constructies en andere mogelijkheden die de databaseontwikkelaar eerder niet ter beschikking stonden, zoals encapsulatie, overerving en polymorfisme. Met de functionaliteit in de .NET Framework Base Class Library hebben databaseontwikkelaars nu onbeperkt toegang tot duizenden vooraf gemaakte klassen en routines die eenvoudig kunnen worden benaderd vanuit een opgeslagen procedure, trigger of een door de gebruiker gedefinieerde functie.
- *Verbeterde veiligheid en beveiliging:* managed code wordt uitgevoerd in een CLR-omgeving, waarvoor de database-engine als host optreedt. Daardoor zijn .NET-databaseobjecten veiliger en beter beveiligd dan de uitgebreide opgeslagen procedures uit eerdere versies van SQL Server. Een van de belangrijkste voordelen van managed code is typebeveiliging. Voordat managed code wordt uitgevoerd, verifieert de CLR dat de code veilig is. Dit proces wordt 'verificatie' genoemd. Tijdens de verificatie

voert de CLR verschillende controles uit om ervoor te zorgen dat de code veilig kan worden uitgevoerd.
- *Door de gebruiker gedefinieerde typen en samenvoegingen:* doordat SQL Server als host optreedt voor de CLR, beschikt SQL Server over twee nieuwe databaseobjecten waarmee de mogelijkheden voor opslag en opvragen worden vergroot.
- *Gemeenschappelijke ontwikkelomgeving:* in Microsoft Visual Studio 2005 wordt de .NET-ontwikkelomgeving en database-ontwikkeling geïntegreerd. Ontwikkelaars gebruiken dan voor de ontwikkeling en foutopsporing van databaseobjecten en -scripts dezelfde gereedschappen als voor het schrijven van .NET-componenten.
- *Prestaties en schaalbaarheid:* omdat managed code voorafgaand aan de uitvoering naar machinetaal wordt gecompileerd, kan in sommige scenario's prestatieverbeteringen worden bereikt.

Ontwikkelde code kan nu eenvoudig worden georganiseerd in klassen en namespaces. Bij grote hoeveelheden servercode kan deze code eenvoudiger worden georganiseerd en onderhouden. De mogelijkheid code logisch en/of fysiek te organiseren in assemblies en namespaces is een voordeel ten opzichte van de mogelijkheden die SQL Server 2000 bood.

Transact-SQL in SQL Server 2005

Is T-SQL, met de soms omslachtige en ingewikkelde syntax, door de komst van CLR-integratie overbodig geworden? Het antwoord hierop luidt: *absoluut niet!* T-SQL is en blijft dé programmeertaal in SQL Server. T-SQL kan worden gebruikt voor het manipuleren en bewerken van data en voor het definiëren van data. Er zijn twee manieren voor het manipuleren van data: declaratief (het gebruik van SELECT/INSERT/UPDATE/DELETE-statements) of procedureel (het gebruik van WHILE, assignment, triggers, cursors, enzovoort.) In het kort kan worden gezegd dat CLR-integratie in SQL Server slechts een alternatief is voor T-SQL om data procedureel te bewerken. Het blijft ook een belangrijk gegeven dat zowel op T-SQL als op CLR gebaseerde code gebruikmaken van dezelfde SQL-taal; alleen het procedurele deel verschilt.

Microsoft SQL Server 2005 biedt veel nieuwe taalmogelijkheden en verbeteringen die gezamenlijk kunnen zorgen voor meer stabiliteit en een betere performance. De uitbreiding en verbetering van de taalmogelijkheden zijn onder andere terug te vinden in foutafhandeling met try/catch-blokken, nieuwe recursieve query-mogelijkheden, het gebruik van snapshot isolation, WAITFOR-, PIVOT- en UNPIVOT- functies. De verbeteringen van T-SQL in SQL Server 2005 vergroten de uitdrukkingsmogelijkheden bij het schrijven van queries, waardoor de ontwikkelaar de prestaties van de code kan verbeteren en het afvangen van fouten kan uitbreiden. De voortdurende inspanningen die door Microsoft worden geleverd om T-SQL te verbeteren, demonstreren een sterk geloof in het belang van T-SQL voor SQL Server.

De keuze: wanneer T-SQL en wanneer CLR?

Een databaseontwikkelaar die op het punt staat gebruik te maken van een specifieke CLR-functionaliteit, dient zich er zeer goed van bewust te zijn hoe hij deze functionaliteit moet gebruiken en wat de gevolgen van het gebruik zijn voor SQL Server. Hoewel er veel voordelen zijn voor het gebruik van CLR-functionaliteit kleeft er ook een aantal nadelen aan. Afhankelijk van de wijze waarop gebruikgemaakt wordt van CLR-integratie zullen sommige voordelen niet opwegen tegen de nadelen. Het kan hierdoor beter zijn om voor een oplossing met behulp van het vertrouwde T-SQL te kiezen in plaats van een op CLR-gebaseerde oplossing.
Zelfs met de aanwezigheid van CLR-ondersteuning is het goed om er vanuit te gaan dat het declaratieve deel van T-SQL zoveel mogelijk wordt gebruikt. Dit deel is in staat om de kracht van de query-processor volledig te gebruiken. De query-processor verzorgt de optimalisatie en uitvoering van bulkoperaties. Over het algemeen kan worden gesteld dat databaseapplicaties alleen gebruik moeten maken van een procedurele programmering als de realisatie niet mogelijk is met het declaratieve deel van de gebruikte query-taal. Het is dus niet aan te raden om alle stored procedures met INSERT-, UPDATE- en DELETE-operaties te herschrijven. Dit voegt niets toe. Samengevat kan worden gezegd: *managed code moet alleen in SQL Server worden gebruikt als het niet met de declaratieve eigenschappen van T-SQL kan worden gerealiseerd.*

T-SQL kan het best gebruikt worden in situaties waarbij de code hoofdzakelijk gegevenstoegang verzorgt en weinig tot geen procedurele logica bevat. De .NET-talen zijn het geschiktst voor CPU-intensieve functies en procedures die worden gekenmerkt door complexe logica. Ook kan gebruik worden gemaakt van een uitgebreide .NET Framework Base Class Library. Managed code heeft duidelijk een betere performance dan T-SQL als het gaat om het procedureel bewerken van data. Voor het verwerken van grote hoeveelheden data is T-SQL sneller. Bij het gebruik van functionaliteit zoals het versleutelen van data, tekstmanipulatie, I/O-operaties, en webservices, biedt de CLR een uitgebreide interface en mogelijkheden die in voorgaande versies van SQL Server en met T-SQL niet mogelijk zijn. Als regel kan worden gezegd dat *voor het realiseren van code met intensieve berekening en bewerkingen het beste voor managed code kan worden gekozen. Voor het realiseren van code met intensieve datatoegang (lezen/schrijven) is T-SQL de beste oplossing.*

Er is niets tegen het ontwikkelen van n-tier-applicaties. Door het gebruik van CLR-functionaliteit treedt er een verplaatsing op van processing van de client naar de server. Dit betekent dat netwerk-roundtrips en netwerkbelasting door de applicatie worden ingeruild tegen processortijd en geheugen van de server. Afhankelijk van het gebruik en van het aantal gebruikers kan dit zowel resulteren in een verbetering als een verslechtering van de performance. Hierbij is het belangrijkste - als het gaat om processorcapaciteit op de server - dat er rekening wordt gehouden met de afweging of het om data-intensieve of reken-intensieve logica gaat. En als het om netwerkbelasting gaat, dat rekening wordt gehouden met de geschatte hoeveelheid data die over het netwerk naar de client moet worden verstuurd. Samengevat leidt dit tot de uitspraak: *gaat het om kleine hoeveelheden data en complexe berekeningen, voer ze uit op de client, gaat het om grote hoeveelheden data en kleinere berekeningen, voer ze uit op de server.*

Processorcapaciteit is ook van belang bij het maken van de keuze tussen T-SQL en CLR. T-SQL en managed code kunnen beide op de server worden uitgevoerd. Op deze manier bevinden functionaliteit en data zich dicht bij elkaar, en kan hierdoor voordeel worden verkregen door gebruik te maken van de verwerkingskracht van de servermachine. Er kan ook voor worden gekozen om juist geen processorintensieve taken op de databaseserver te plaatsen. Tegenwoordig zijn de meeste clientmachines erg krachtig en daar kan gebruik van worden gemaakt door zoveel mogelijk code op de client te plaatsen. Afhankelijk van de beschikbare processorcapaciteit op de client en op de server, in combinatie met de aanwezigheid van data-intensieve of rekenintensieve logica, zal een keuze moeten worden gemaakt. Hiervoor is geen pasklaar antwoord. Functies kunnen het beste in managed code worden geïmplementeerd, omdat er zelden informatie uit de database nodig is, deze worden namelijk via parameters doorgegeven aan de functie. Functies bevatten voornamelijk berekeningen, wat nu juist de kracht is van managed code. Managed code is hierin een stuk sneller in dan T-SQL.
Wordt in SQL Server 2000 gebruik gemaakt van extended stored-procedures (in C/C++), overweeg dan ze te vervangen door CLR-procedures en -functies. Extended stored procedures zijn unmanaged en draaien in het proces en in de security-context van SQL Server; dit betekent dat slecht geschreven extended stored-pro-

```
--Inschakelen van clr integratie
sp_configure 'clr enabled', 1
go

reconfigure
go
```

Codevoorbeeld 1.
Inschakelen van CLR-integratie in SQL Server 2005

```
--Uitschakelen van clr integratie
sp_configure 'clr enabled', 0
go

reconfigure
go
```

Codevoorbeeld 2.
Uitschakelen van CLR-integratie in SQL Server 2005

cedures ervoor kunnen zorgen dat het gehele SQL Server-proces onderuit gaat. Het gebruik van de CLR levert een veiliger en beter schaalbaar resultaat op. Aangezien CLR-functionaliteit in een eigen applicatiedomein wordt uitgevoerd, heeft het zijn eigen security-context en kan het niet bij SQL Server-geheugen; het betreft safe code. Daarnaast is .NET-code eenvoudiger uit te deployen. Ook COM-objecten die via sp_OA stored-procedures worden aangeroepen via het late-binding-model en in SQL Server 2000 gebruikt werden, kunnen het beste worden vervangen door CLR-classes. Probeer ook het gebruik van COM-interop te voorkomen. In veel gevallen is de gewenste functionaliteit ook in de .NET-klassenboom aanwezig. Is het vervangen van het COM-object niet mogelijk, zorg er dan voor het COM-object vanuit een CLR-procedure of -functie aan te roepen en gebruik te maken van het early-binding-model. Dit zorgt voor een betere performance en vermindert de kans op programmeer- en executiefouten.

Installatie

Mocht je om een bepaalde reden niet tevreden zijn over het gebruik van de CLR-functionaliteit in SQL Server 2005 of het om veiligheidsredenen niet willen gebruiken, dan kun je deze altijd uitschakelen. Microsoft heeft er zelfs bij installatie van SQL Server 2005 voor gekozen, om de CLR-functionaliteit (ook een groot aantal andere nieuwe features, zoals databasemail, XML-webservice-functionaliteit, enzovoort) uitgeschakeld te houden. Wil je deze activeren, dan moet je ze dus eerst inschakelen. Inschakelen van CLR-functionaliteit doe je door de commando's in codevoorbeeld 1 uit te voeren.

Het uitvoeren van het eerste commando zorgt voor het inschakelen van de CLR-functionaliteit, echter niet in de huidige sessie. Het tweede commando start de database-engine met de aanwezige configuraties. CLR-functionaliteit kan dus worden in- en uitgeschakeld zonder de hele server opnieuw op te starten. De CLR-functionaliteit kan direct worden gebruikt. Uitschakelen van CLR-functionaliteit kun je doen door de commando's in codevoorbeeld 2 uit te voeren.

Optimale mix

Hoewel CLR-integratie veel enthousiasme opwekt bij SQL Server-developers, betekent de introductie van het .NET Framework in SQL Server nog niet dat hiermee automatisch alle T-SQL-code vervangen moet worden. CLR-functionaliteit kan ervoor zorgen dat een productiviteitsverbetering kan worden gerealiseerd op een manier die in voorgaande versies van SQL Server niet mogelijk was. Daarnaast bevat SQL Server 2005 ook T-SQL-verbeteringen, die ervoor zorgen dat T-SQL stabieler en uitgebreider wordt. De uitdaging is de voordelen van de CLR en het .NET Framework en de verbeteringen van T-SQL zo goed mogelijk te benutten om tot de optimale mix te komen.
Wat is nu de afweging die moet worden gemaakt om kiezen tussen T-SQL enerzijds en managed code anderzijds?

- Managed code moet alleen in SQL Server worden gebruikt als deze niet met de declaratieve eigenschappen van T-SQL gerealiseerd kan worden.
- Voor het realiseren van code met intensieve berekening en bewerkingen kan het beste voor managed code worden gekozen. Voor het realiseren van code met intensieve datatoegang (lezen/schrijven) is T-SQL de beste oplossing.
- Voer kleine hoeveelheden data en complexe berekeningen uit op de cliënt. Voer grote hoeveelheden data en kleinere berekeningen ze uit op de server.
- Op basis van beschikbare processorcapaciteit op zowel client als server, en in combinatie met de aanwezigheid van data-intensieve of rekenintensieve logica, moet de ontwikkelaar bepalen waar de code uitgevoerd moet worden. Hiervoor is geen pasklaar antwoord.

Functies zijn de beste kandidaten om in managed code te worden geïmplementeerd. Functies bevatten voornamelijk berekeningen, wat nu juist de kracht is van managed code. Extended stored-procedures en COM-objecten moeten niet meer worden gebruikt en moeten in managed code worden geïmplementeerd. Bij migratie naar SQL Server 2005 zijn dit de onderdelen waarbij herschrijven van de code een zeer positief effect kan hebben op zaken als performance en stabiliteit. De discussie over het gebruik van CLR-integratie of T-SQL in SQL Server 2005 is nog niet voorbij, maar het is wel duidelijk dat het T-SQL-tijdperk zeker nog niet kan worden afgesloten. T-SQL is springlevend en biedt ontwikkelaars verbeteringen en uitbreidingen die voor nog meer functionaliteit en mogelijkheden in gebruik zorgen. Bovendien kan worden gesteld dat de toevoeging van CLR-functionaliteit in SQL Server 2005 tot nieuwe mogelijkheden en tot fundamentele wijzigingen zal leiden in de manier waarop architecten en ontwikkelaars databaseapplicaties zullen ontwikkelen.

Robert Tusveld is als softwarearchitect werkzaam bij Capgemini (www.nl.capgemini.com). Hij heeft zich gespecialiseerd op het gebied van Microsoft-producten en -technologieën. Robert is te bereiken via e-mail op Robert.Tusveld@Capgemini.com

Interessante links:

- www.sqljunkies.com
- www.sqlteam.com
- www.microsoft.com/SQL
- www.microsoft.com/SQL/2005
- msdn.microsoft.com/SQL
- msdn.microsoft.com/SQL/2005