Kennismaking met de nieuwe versie van SQL Server

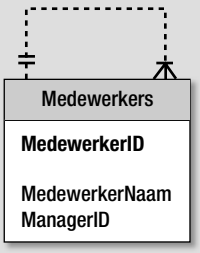
Codenaam Yukon

Astrid Hackenberg

Op de Microsoft Professional Developer Conference, afgelopen november in Los Angeles, hebben zo'n 7000 software-engineers en software-architecten kennis gemaakt met de volgende versie van SQL Server; codenaam Yukon. In dit artikel een overzicht van wat de database-ontwikkelaar kan verwachten in SQL Server Yukon.

Bij de planning van de SQL Server Yukon versie is vooral gekeken naar verbetering op twee gebieden. Ten eerste de database-programmeercapaciteiten en ten tweede de flexibiliteit in het ondersteunen van verschillende datamodellen. De belangrijkste nieuwe functionaliteiten zijn: hosting van het .NET framework; native XML support; gebruik als object database; meer conformiteit aan de ANSI-99 SQL-specificatie
Daarnaast is er een aantal nieuwe en vernieuwde tools: SQL Server Workbench, een omgeving voor het beheren en ontwikkelen van databases; SQL Server Business Intelligence Workbench, een omgeving voor het beheren en ontwikkelen van data cubes, datamining-modellen, datatransformatie packages en rapporten; Reporting Services, een platform voor het maken, beheren en uitleveren van rapportages; Data Transformation Services, een platform voor datatransformatie en data-integratie; Analysis Services, een platform voor OLAP en datamining; Notification Services, een platform voor het genereren en verzenden van berichten aan abonnees van een notificatie service; tenslotte Service Broker, een bericht-gebaseerd communicatieplatform voor communicatie tussen verschillende applicatie-componenten.

Afbeelding 1: De SQL Server workbench met object explorer, solution explorer en task page actief.



Afbeelding 2: Medewerkers-tabel met recursieve relatie.

Wat is er nieuw in Transact SQL?

SQL Server Yukon T-SQL bevat een heel aantal nieuwe taalconstructies. Deze toevoegingen zijn voornamelijk gericht op overeenstemming met de ANSI-99 SQL-specificatie en de ondersteuning van XML- en .NET-programmering. Het is niet mogelijk om alles wat nieuw is in T-SQL in dit artikel te bespreken.
Er is een selectie gemaakt van enkele opvallende nieuwe functionaliteiten.
Eén van de nieuwe taalconstructies is de Common Table Expression (CTE). De CTE maakt het mogelijk om in één query hiërarchieën van recursieve relaties te doorlopen. Met SQL Server Yukon kunnen we query's definiëren volgens onderstaande structuur.

```
WITH <CTEName> ( <column-list> )
AS
       <non-recursive SELECT>
    UNION ALL
       <SELECT referencing CTE>
```

Stel men heeft een medewerkers-tabel in de database, waarin naast de medewerker gegevens ook is vastgelegd wie de manager is van de medewerker, zoals weergegeven in afbeelding 2. Deze manager is zelf ook weer een medewerker die weer een manager kan hebben, enzovoort. Het selecteren van een medewerker met

de volledige hiërarchie van alle daaronder vallende medewerkers tot aan het laagste niveau, kan door middel van de onderstaande query worden uitgevoerd.

```
WITH MedewerkerCTE(MedewerkerID, MedewerkerNaam,
                                        ManagerID)
AS
(
       SELECT MedewerkerID, MedewerkerNaam,
                                        ManagerID
       FROM Mederwerkers
       WHERE MedewerkerID = '12345'
   UNION ALL
       SELECT MedewerkerID, MedewerkerNaam,
                                        ManagerID
       FROM Medewerkers AS M JOIN
                            MedewerkersCTE AS CTE
          ON M.ManagerID = CTE.MedewerkerID
)
SELECT * FROM MedewerkerCTE
```

Zonder de CTE-constructie zou hiervoor een iteratie met *while* statement, een tijdelijke tabel en hulpvariabelen benodigd zijn.
Een andere nieuwe taalconstructie is de OUTPUT clausule voor DML statements. Hiermee kan men een resultaatset verkrijgen als onderdeel van een UPDATE, DELETE of INSERT statement.
Ook de zogenaamde 'deleted' en 'inserted' tabellen zijn beschikbaar. Dit zijn views over het SQL Server transactie-log en hierdoor is het mogelijk om bijvoorbeeld de waarden voor en na de update van een kolom te retourneren. De OUTPUT clausule geeft bovendien de mogelijkheid om de resultaatset persistent te bewaren door de toevoeging INTO. In de onderstaande code wordt de prijs van een artikel aangepast en tegelijkertijd gerapporteerd wat de oude prijs en de huidige prijs is.

```
UPDATE Artikelen
   SET prijs = prijs*1.3
OUTPUT DELETED.artikelcode, DELETED.prijs,
                                    INSERTED.prijs
   INTO ArtikelPrijsHistorie
```

Een derde nieuwe taalconstructie is de TRY-CATCH. Door gebruik te maken van deze constructie wordt een exceptie gegooid, in het geval van een fout in T-SQL batch. In het BEGIN TRY - END TRY blok worden de statements geplaatst die eventueel een fout zouden kunnen opleveren. In het BEGIN CATCH - END CATCH blok wordt de fout afgevangen en kan een foutafhandeling worden geschreven.

```
BEGIN TRY
   BEGIN TRAN
      – Eén of meerdere INSERT, UPDATE, DELETE
                                        statements
   COMMIT TRAN
END TRY
BEGIN CATCH TRAN_ABORT
   – Foutafhandeling
   ROLLBACK
END CATCH
```

In de bovenstaande code wordt een fout die zou leiden tot het afbreken en terugdraaien van een transactie opgevangen. Na een eigen foutafhandeling wordt de transactie alsnog beëindigd.
SQL Server Yukon kent ook een aantal nieuwe datatypes. Er is een apart datatype *date* voor datum en *time* voor tijd.
De *varchar(), nvarchar()* en *varbinary()* datatypes zijn uitgebreid met een *max* indicator. Bij gebruik van de max indicator kan een kolom van dit datatype tot 2 Gigabyte aan data bevatten. Maar het meest interessante nieuwe datatype is ongetwijfeld het *xml* datatype. In een kolom van het type xml kan een volledig XML-document worden opgeslagen. Meer hierover verderop in dit artikel.
Als laatste kan nog een uitbreiding op de DDL genoemd worden. Alle DDL statements hebben events, die door middel van een trigger kunnen worden afgevangen. Trigger procedures kunnen worden gedefinieerd op database- en op server-niveau. Door deze DDL triggers kunnen ontwikkelstandaarden ten aanzien van database-objecten worden controleerd en afgedwongen. Maar het is bijvoorbeeld ook mogelijk om te voorkomen dat een database-object 'per ongeluk' wordt verwijderd.

Wat biedt de .NET framework integratie?

De Common Language Runtime (CLR) wordt in SQL Server Yukon gehost in de database engine. Hierdoor kan een database-ontwikkelaar stored procedures, triggers en user defined functions schrijven in iedere .NET-programmeertaal zoals C# of VB.NET.
De CLR is een executie omgeving die zorgt voor zaken als type-controle, geheugenbeheer, exceptie-afhandeling en codebeveiliging. Bovendien biedt de CLR een gezamenlijk type-systeem, het Common Type System, en hebben we de beschikking over een rijke collectie class library's. Programmacode die in de CLR draait noemen we managed code.
Door deze integratie van SQLCLR kunnen database-ontwikkelaars taken uitvoeren in programma's die in T-SQL ingewikkeld of zelfs onmogelijk zijn. Het ontwikkelen van databaseprogramma's is hiermee gelijk geworden aan de ontwikkeling van client- of business-componenten en services. Als men data-intensieve business-componenten heeft dan zijn dat de potentiële kandidaten om in SQL Server Yukon als database-component te gaan draaien.

Men kan stored procedures (SP), triggers en user-defined functions (UDF) dus schrijven als managed code. Hierbij is alles te gebruiken wat het .NET framework biedt aan functionaliteit.
Om vanuit managed code de SQL Server database-objecten te gebruiken is er zelfs een speciale versie van ADO.NET, de .NET data access-laag, die geoptimaliseerd is voor de SQLCLR.
De managed code is echter vooral geschikt voor programmeer-constructies zoals iteraties of het conditioneel uitvoeren van code.

Dit kan in managed code vaak in veel minder regels gecodeerd worden dan in T-SQL. Daarnaast kan het gebruik van standaard functionaliteit uit de .NET framework class library's het schrijven van code vereenvoudigen en versnellen.
SQL Server Yukon geeft in managed code ook nog de beschikking over een tweetal nieuwe database-objecten: *user-defined types* (UDT) en *user-defined aggregate functions* (UDA). User-defined types kunnen worden gebruikt als datatype in een tabel definitie aan de database server-kant en kunnen worden gemanipuleerd als objecten aan de client-kant. Een UDT is een eigen gedefinieerd type dat uit datavelden en functies kan bestaan, die intern of extern bruikbaar kunnen zijn. In .NET-termen is een UDT een *value type* dat private en public fields, properties en methods kan bevatten.

> Managed code is bedoeld voor procedureel programmeren en berekeningen

Een *user-defined aggregate function* is een .NET class die een mathematische berekening uitvoert op een set van waardes en 1 waarde retourneert. Een UDA kan op dezelfde wijze worden gebruikt als een standaard aggregaat, bijvoorbeeld als expressie in een SELECT-lijst of in GROUP BY.
Om managed code in een runtime-omgeving te gebruiken compileert men deze tot een assembly. Een assembly bevat naast de gecompileerde code ook de metadata van de types en een manifest met daarin informatie over afhankelijkheden met andere assembly's, een versienummer enzovoort. Deze assembly's moet men registeren in de SQL Server-database om ze te kunnen gebruiken. Hiervoor zijn speciale nieuwe DDL statements beschikbaar. Dit proces is weergegeven in afbeelding 3.

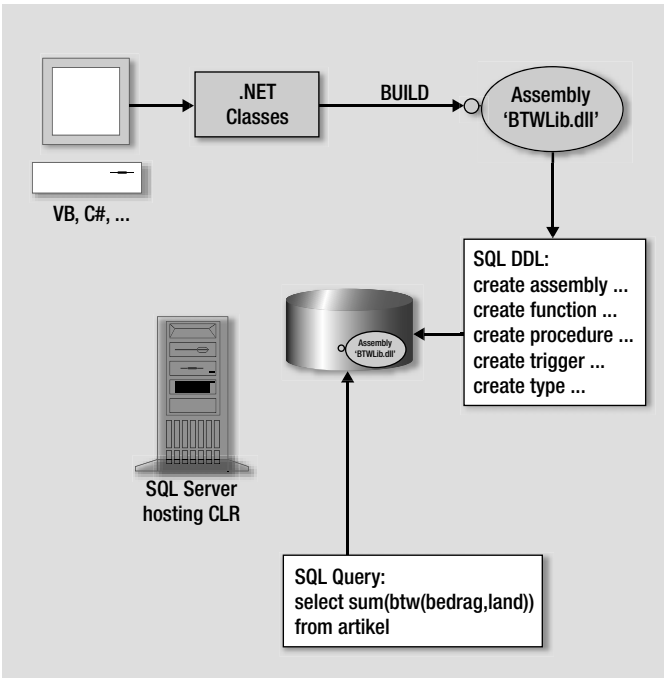
Is managed code beter dan T-SQL?

Het schrijven van complexe constructies in code wordt dus gemakkelijker door het gebruik van een .NET-programmeertaal. Maar de keuze tussen het schrijven van database-code in T-SQL versus managed code is er één die zorgvuldig moet worden gemaakt. T-SQL is geoptimaliseerd voor data access en set-gebaseerde operaties. Managed code is bedoeld voor procedureel programmeren en berekeningen. Wil men optimaal gebruik maken van de sterke kanten van beiden dan moet men query's schrijven in T-SQL die User-defined Functions en User-defined Aggregates aanroepen die geschreven zijn in managed code.

Hoe is XML geïntegreerd?

XML is in de tijd geëvolueerd van een representatietechnologie naar een opslagformaat. SQL Server Yukon heeft zoals eerder vermeld een nieuw *xml* datatype. Een kolom, parameter of variabele van dit datatype kan een XML-document of een XML-fragment bevatten. Er zijn veel mogelijke toepassingen voor de opslag van XML in een database, zoals bijvoorbeeld als het schema van een object onbekend is of als het schema dynamisch is en op regelmatige basis wijzigt. Daarnaast is XML-persistentie essentieel in applicaties die gebaseerd zijn op XML-documenten of XML-berichten.

We kennen in SQL Server Yukon *typed* en *untyped* XML. Door aan een XML-datatype een schema te koppelen wordt deze getypeerd. Het gebruik van getypeerde XML heeft een aantal voordelen ten opzichte van ongetypeerde XML. Ten eerste wordt getypeerde XML gevalideerd op basis van het schema. Ten tweede is de fysieke opslag van getypeerde XML veel compacter aangezien de data nu in native SQL-formaat wordt bewaard. Als men bijvoorbeeld een XML-element of -attribuut van het type *int* hebt dan wordt dat in de database ook als een integer opgeslagen.
Het schema dat apart van de XML-data in de database wordt opgeslagen bevat de informatie over de representatie als element of attribuut. Tenslotte is het uitvragen van getypeerde XML sneller, doordat de structuur van de XML bekend is.
Het xml datatype is een volwaardig datatype in SQL Server Yukon. Niet alleen kunnen rij- en kolom-constraints worden geïmplementeerd via XSD-schema's. Een XML-kolom kan worden geïndexeerd met het CREATE XML INDEX statement. Een XML-index indexeert de interne structuur van een XML-kolom, de elementen en attributen. Een XML-index is dan ook bedoeld voor het optimaliseren van XQuery query's en niet voor optimalisatie van SQL query's. Voor het optimaliseren van SQL query's op een XML-kolom kan een full-text index worden gebruikt.



Afbeelding 3: Het gebruik van managed code in SQL Server Yukon.

Afbeelding 4: De geïntegreerde XQuery designer.

Met XQuery is in SQL Server Yukon ook een geavanceerde query-mogelijkheid voor XML geïmplementeerd. XML Query Language (XQuery) is een specificatie in ontwikkeling onder verantwoordelijkheid van het W3C. Met XQuery kan men zowel getypeerde als ongetypeerde XML-kolommen en -variabelen uitvragen. XQuery gebruikt Xpath-expressies om nodes te selecteren maar voegt extra functionaliteiten toe aan Xpath, zoals de constructie van nieuwe elementen en/of attributen als onderdeel van de query.

SQL Server Yukon is uitgerust met een *XQuery designer*, een tool waarmee XQuery query's visueel kunnen worden geconstrueerd. Afbeelding 4 toont een query in de XQuery designer. Door een getypeerde XML-kolom op te nemen in het *source document* window, krijgt men een visuele representatie van de interne structuur van deze kolom. Door middel van drag & drop kan men nu een query definiëren. De XQuery designer geeft ook de source code en de mogelijkheid de query uit te voeren en het resultaat te zien. Met de komst van het xml-datatype, XQuery en de XQuery designer is SQL Server Yukon voor het bouwen van XML-databases een serieuze keuze geworden.

Wat biedt SQL Server Yukon nog meer?

In dit artikel is slechts een aantal van de nieuwe en spannende onderdelen van SQL Server Yukon behandeld. Duidelijk is dat er voor database-ontwikkelaars nog niet eerder zoveel beschikbaar was voor data access en data-opslag. Nieuwe SQL Server Yukon services zoals de Service Broker, Notification Services en Reporting Services maken meer en andere toepassingen van SQL Server mogelijk. Dit betekent dat er ook veel keuzes te maken zijn. Database-architectuur en design is dus belangrijker dan ooit. In volgende artikelen wordt dieper ingegaan op de verschillende functionaliteiten en tools.

Astrid Hackenberg (astrid.hackenberg@class-a.nl) is werkzaam bij Class-a training en coaching.