howest
hogeschool

Syllabus

Databases

Mileto Di Marco
Kristien Roels
Bart Soete

Inhoudsopgave

H8 - Programmeren in Transact-SQL 3

1 Inleiding 3

2 Batches en scripts 3

3 Lokale variabelen 3

4 De CASE-expressie 5

5 De statements IF en WHILE 5

6 @@functies (globale variabelen) 6

7 Tijdelijke tabellen 7

8 Dynamic SQL 8

H8 - Programmeren in Transact-SQL

1 Inleiding

De standaard SQL DML (SQL Data Manipulation Language) bevat de statements SELECT, INSERT, UPDATE en DELETE voor het afleveren of aanpassen van data. Deze standaard DML bevat geen procedurele statements (zoals IF of WHILE). Elk volledig DBMS moet de SQL-standaard uitbreiden. Transact-SQL is de Microsoft-implementatie van de standaard SQL met aanvullingen.

In dit hoofdstuk worden de programmatorische elementen die nodig zijn voor de ontwikkeling van server-side procedurele code bekeken. Dit is de basis voor de ontwikkeling van stored procedures, user-defined functions en triggers.

2 Batches en scripts

Een batch is een groep van één of meerdere Transact-SQL-statements die in 1 keer naar SQL Server gestuurd worden voor uitvoer. SQL Server compileert de statements en voert ze daarna uit.
Als er een compilatiefout optreedt, dan wordt dat gemeld en worden de statements niet uitgevoerd.
Als er een runtime-error optreedt (bijvoorbeeld een foute objectnaam), dan wordt het huidige statement gestopt en meestal worden ook de daaropvolgende statements niet meer uitgevoerd.

Sommige DDL-statements (Data Definition Language), zoals CREATE DATABASE, kunnen niet gecombineerd worden met andere statements in één batch.

Het commando GO signaleert het einde van een batch Transact-SQL-statements aan de SQL Server utilities. Zie ook: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/sql-server-utilities-statements-go

Een script is een bestand waarin meerdere Transact-SQL statements zijn opgeslagen. Een script kan één of meerdere batches bevatten, gescheiden van elkaar door het GO-commando.

Een script heeft vaak de volgende vorm:

```
batch1
GO
batch2
GO
batch3
GO
…
```

De voorbeelddatabank die in dit hoofdstuk gebruikt wordt is de GRADSchool-databank.

3 Lokale variabelen

Voor een volledige bespreking van (lokale) variabelen, zie https://docs.microsoft.com/en-us/sql/t-sql/language-elements/variables-transact-sql

Een Transact-SQL lokale variabele kan een waarde bevatten van een specifiek type en moet gedeclareerd worden:

DECLARE variabelenaam type [= waarde] [, …]

De naam van een lokale variabele moet beginnen met @.

Het type is één van de types die ook gebruikt worden bij de definitie van een kolom.

Een lokale variabele die gedeclareerd is, heeft de waarde NULL. Een lokale variabele kan geïnitialiseerd worden bij de declaratie.

Het bereik (scope) van een lokale variabele loopt vanwaar ze gedeclareerd is tot het einde van de batch of de procedure waarin ze gedeclareerd is.

Declareer (en initialiseer) lokale variabelen.
declare @betaald int declare @naam nvarchar(100) declare @zoeknaam nvarchar(10) = 'Van%', @geslacht nvarchar(10)

Het initialiseren na de declaratie kan gebeuren met het SET-statement. Het SET-statement initialiseert 1 lokale variabele met een expressie:

SET variabelenaam = expressie

Initialiseer een lokale variabele met het SET-statement. Gebruik deze lokale variabele in een query.
set @betaald = 100
select * from studenten where betaald < @betaald and familienaam like @zoeknaam

Het initialiseren na de declaratie kan ook gebeuren met het SELECT-statement. Het SELECT-statement kan meerdere lokale variabelen initialiseren met een expressie.
Het is ook mogelijk om de lokale variabele(n) te initialiseren met data uit een tabel. Als in dat geval het SELECT-statement meerdere rijen aflevert, wordt de data van de laatste rij opgeslagen in de lokale variabele(n).

SELECT variabelenaam = expressie [, …]
[FROM-clausule
WHERE-clausule
ORDER BY-clausule]

Initialiseer lokale variabelen met het SELECT-statement. Gebruik deze lokale variabelen in een query.
select @naam = familienaam, @geslacht = geslacht from studenten where studnr = 1
update studenten set betaald = betaald + 1 where familienaam = @naam and geslacht = @geslacht

4 De CASE-expressie

De corresponderende Help-pagina: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/case-transact-sql

Er zijn 2 formaten van de CASE-expressie:

- CASE input_expressie
  WHEN when_expressie THEN resultaat_expressie [...n]
  [ELSE else_resultaat_expressie]
  END

  De input_expressie wordt vergeleken met een when-expressie. Bij de eerste gelijkheid wordt de resultaat_expressie afgeleverd.

- CASE
  WHEN boolean_expressie THEN resultaat_expressie [...n]
  [ELSE else_resultaat_expressie]
  END

  De boolean_expressie wordt geëvalueerd. Bij de eerste TRUE wordt de resultaat_expressie afgeleverd.

De CASE-expressie kan gebruikt worden in elk statement en in elke clausule waar een expressie verwacht wordt. CASE kan dus gebruikt worden in een SELECT-statement, in een UPDATE-statement, … en in een ORDER BY-clausule, een WHERE-clausule, ….

Geef een lijst van alle studenten met de juiste aanspreking (Dhr. of Mevr.).

```
select *, case geslacht
            when 'M' then 'Dhr.'
            when 'V' then 'Mevr.'
            else ''
          end as aanspreking
from studenten
```

Geef een lijst van alle studenten met een tarief-aanduiding.

```
select *, case
            when betaald > 100 then 'Hoog tarief'
            when betaald > 50 then 'Midden tarief'
            when betaald > 0 then 'Laag tarief'
            else '0 tarief'
          end as tarief
from studenten
```

5 De statements IF en WHILE

Voor een overzicht van alle control-of-flow-statements: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/control-of-flow

Een selectie wordt geschreven met een IF-statement. De syntax van het IF-statement:

```
IF voorwaarde
    BEGIN
      statementblok
    END
[ELSE
    BEGIN
      statementblok
    END]
```

Ook geneste IF-statements zijn mogelijk.

IF EXISTS(SELECT-statement) gaat na of er rijen afgeleverd worden door het SELECT-statement.

Een iteratie wordt geschreven met een WHILE-statement. De syntax van het WHILE-statement:

```
WHILE voorwaarde
    BEGIN
      statementblok
    END
```

De selectie (IF) en de iteratie (WHILE) worden gebruikt in batches, stored procedures, user-defined functions en triggers. Zie verder in de cursus.

6 @@functies (globale variabelen)

De naam van sommige Transact-SQL systeemfuncties begint met @@. In eerdere versies van SQL Server werd de term globale variabele gebruikt voor deze functies. Deze @@functies zijn terug te vinden onder de node System Functions in de Object Explorer. Ze leveren informatie over het huidige systeem.

Hieronder enkele voorbeelden van @@functies:

@@identity	Levert de laatste identity waarde gegenereerd tijdens de huidige connectie. https://docs.microsoft.com/en-us/sql/t-sql/functions/identity-transact-sql
@@datefirst	Levert de dag van de week die de eerste dag van de week representeert. Hierbij is 1 maandag, 2 dinsdag, …. Voor taal US-English: standaard 7; voor taal Italian: standaard 1 https://docs.microsoft.com/en-us/sql/t-sql/functions/datefirst-transact-sql
@@error	Levert het errorgetal van het vorig uitgevoerde Transact-SQL statement. De returnwaarde is 0 indien het vorig uitgevoerde Transact-SQL statement succesvol was. Overzicht van alle errorgetallen: select * from sys.messages https://docs.microsoft.com/en-us/sql/t-sql/functions/error-transact-sql
@@rowcount	Levert het aantal rijen, afgeleverd door het vorige Transact-SQL statement. https://docs.microsoft.com/en-us/sql/t-sql/functions/rowcount-transact-sql

Merk op dat foutafhandeling vroeger gedaan werd met o.a. @@error en @@rowcount. In SQL Server 2005 werd TRY … CATCH geïntroduceerd. Deze manier van foutafhandeling wordt vanaf dan aangeraden.

Voeg een nieuwe student toe. Schrijf deze nieuwe student in voor de cursus 1.
insert into studenten(voornaam, familienaam, geslacht) values ('Valerie','Delange','V') --studnr is een identity-kolom insert into studenten_cursussen(studnr, cursusnr) values(@@identity, 1) --het zojuist gegenereerde studnr wordt gebruikt

7 Tijdelijke tabellen

Een tijdelijke tabel wordt gecreëerd als een gewone tabel, maar de naam moet beginnen met #. De tijdelijke tabellen worden gecreëerd in de databank tempdb.

Een tijdelijke tabel heeft een korte levensduur. Als een tijdelijke tabel in een stored procedure wordt aangemaakt, dan wordt de tijdelijke tabel gewist als de stored procedure eindigt. Andere tijdelijke tabellen worden automatisch verwijderd bij het einde van de sessie. Een tijdelijke tabel heeft ook een beperkt bereik. Alleen de connectie die de tabel maakte, kan de tabel zien.

Als een stored procedure wil verder werken op de resultaatset van een andere stored procedure, dan wordt een tijdelijke tabel gebruikt. Deze tijdelijke tabel wordt opgevuld met de resultaatset van de stored procedure. Zie hoofdstuk 9 Stored procedures.

De corresponderende Help-pagina: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql#temporary-tables

Maak een tijdelijke tabel met de vrouwelijke studenten uit de GRADSchool-databank. Gebruik deze tijdelijke tabel. Bekijk tot slot de gegevens van de tijdelijke tabel in de databank tempdb.
create table #vrouwen(studnr int primary key, familienaam nvarchar(50)) insert into #vrouwen(studnr, familienaam) --tijdelijke tabel opvullen select studnr, familienaam from studenten where geslacht = 'V' select * from #vrouwen v left join studenten_cursussen sc on v.studnr = sc.studnr left join cursussen c on sc.cursusnr = c.cursusnr select * from tempdb.sys.objects where name like '#v%'

8 Dynamic SQL

Dynamic SQL betekent dat het mogelijk is om een SQL DML-statement dynamisch (at run-time) te vormen als een string. Eens deze string volledig is, kan het SQL-statement uitgevoerd worden met de systeem stored procedure sp_executesql.

SQL-injectie is een reële bedreiging: let dus op dat invoer van de gebruiker nooit rechtstreeks gebruikt wordt om een dynamische SQL-string op te bouwen. Zorg dat de gebruikersinvoer steeds via parameters aan een stored procedure wordt doorgegeven. In deze stored procedure kunnen de ontvangen waarden gebruikt worden om een dynamische SQL-string op te bouwen.

Schrijf een batch waarin een SELECT-statement wordt opgebouwd als een string. Het SELECT-statement levert de studenten die geboren zijn na het opgegeven geboortejaar en met het opgegeven geslacht. Als er geen geboortejaar is opgegeven, dan wordt geen rekening gehouden met de geboortedatum.
Voer tot slot het gevormde SQL-statement uit.

```
--declaratie van de variabelen
declare @geboortejaar int = NULL
declare @geslacht char(1) = 'M'
declare @sql nvarchar(500)

--opstellen van het SELECT-statement
set @sql = 'select * from studenten where geslacht = '''+ @geslacht + ''''
if @geboortejaar is not null
begin
	set @sql = @sql + ' and year(geboortedatum) > ' + cast(@geboortejaar as nvarchar(4))
end

--om te testen
select @sql

--uitvoeren van het gevormde SQL-statement
exec sp_executesql @sql
```