leren. durven. doen.

SYNTRA WEST

DATABANKEN - SQL

C# programmeur

SQL – INHOUD

- **Views**
- **T-SQL**
  - **Built-in functions**
  - **Variabelen**
  - **IF ...ELSE...**
  - **CASE**

leren. durven. doen.

SYNTRA WEST

DATABANKEN - SQL

Views

west-vlaanderen POM

Wat is een view?

= virtuele tabel gebaseerd op het resultaat van een SQL statement

- Dient om data in een natuurlijke of intuitieve manier te structureren
- Beperkt de toegang tot andere data zodat de gebruiker enkel dat ziet wat het mag/moet zien
- Verzamelt data van meerdere tabellen om reports te genereren

Opmerking

- Omdat views net als tabellen zijn, kan je ook queries uitvoeren op views.
- Een view wordt “bevroren” tijdens de aanmaak ervan: veranderingen aan de onderliggende tabellen worden niet automatisch opgenomen in de view (bv. het toevoegen van een kolom)

CREATE VIEW

Views kunnen gemaakt worden van een enkele of meerdere tabellen of een andere view

CREATE VIEW *view_name*
AS
SELECT...

Voorbeeld

CREATE VIEW Vw_BrouwersBeperkt
AS
SELECT B.BrouwerNr, B.BrNaam, B.Adres, B.PostCode, B.Gemeente
FROM Brouwers B

Resultaat

Een (updatable) view die alle velden van de tabel “Brouwers” toont met uitzondering van het veld “omzet”

CREATE VIEW

```
CREATE VIEW view_name
AS
SELECT...
```

OPGELET ! UPDATABLE vs READ-ONLY VIEW

Afhankelijk van de soort SELECT heeft een view de mogelijkheid om gegevens
toe te voegen, te wijzigen en te wissen.
criteria voor een **updatable view:**

- Gebouwd op 1 enkele tabel
- Geen GROUP BY, HAVING, UNION, SELECT DISTINCT clausules
- Geen aggregate functions zoals AVG(), SUM(), …
- Geen berekende kolommen
- Geen subquery in de SELECT
- Geen JOIN

CREATE VIEW

Views worden dikwijls gemaakt om gebruikers berekende gegevens te bieden zoals bij dit volgend voorbeeld

```
CREATE VIEW view_name
AS
SELECT...
```

Voorbeeld

```
CREATE VIEW vw_toptien
AS
SELECT TOP 10 brouwernr, brnaam, omzet FROM Brouwers
ORDER BY omzet DESC
```

Resultaat

Een view die een lijst van de 10 brouwers met de meeste omzet selecteert.

DROP VIEW

Verwijdert een view

```
DROP VIEW view_name
```

Voorbeeld

DROP VIEW vw_toptien

Resultaat

verwijdert de view **vw_toptien** uit de database

Oefeningen

Download van Github:

Oefeningen Views_3.pdf

https://www.w3schools.com/sql/sql_view.asp

leren. durven. doen.

SYNTRA WEST

DATABANKEN - SQL

T-SQL

west-vlaanderen

POM

Gebruik van commentaar

Inline commentaar

```
--commentaar
```

Block commentaar

```
/* commentaar*/
/*this code retrieves for each book
  ** the title and the number of books sold
 */
select titles.title_id, title, count(qty) --number of books sold
from titles left join sales on titles.title_id=sales.title_id
group by titles.title_id, title
```

Lokale variabelen: declaratie

- **Declaratie van variabele:**

naam variabele wordt steeds voorafgegaan door @
Voorbeeld:

```
DECLARE @custID VARCHAR(40), @rijtelling INT
```

Lokale variabelen: Toekenning waarde

- **Toekennen van waarde aan variabele**

set en select zijn gelijkaardig,
maar set is ANSI standaard, voorbeeld:

```
set @max = (select max(invoiceTotal) frominvoices)
select @max = max(invoiceTotal) frominvoices
```

via een select kan je meerdere variabelen in één keer waarde geven

```
select @max= max(invoiceTotal), @nrOfInvoices=
count(*) frominvoices
```

Lokale variabelen: Afprinten (SSMS)

```
PRINT string_expression
```

- binnen Management Studio wordt het bericht in het messages-venster getoond bij gebruik `PRINT`
- als alternatief kan je eveneens de **select** gebruiken:

```
DECLARE @lname VARCHAR(50)
SET @lname = 'Jan Janssens'

PRINT  'De naam van de werknemer is ' + @lname

select 'De naam van de werknemer is ' + @lname
```

Lokale variabelen: Afprinten (SSMS)

```
PRINT string_expression
```

Voorbeeld:

```
DECLARE @custID VARCHAR(40), @rijtelling INT
SET @custID ='ROMEY'

SELECT @rijtelling = count(*) FROM Orders
WHERE CustomerID = @custID;

PRINT @rijtelling

SELECT @rijtelling
```

Oefening T-sql declaratie en toekenning van variabelen

1. Open in SSMS(SQL Server Management Studio) de database Northwind en open een nieuw query window

2. Declareer in dit query window hierin een variabele van het type INT en ken de waarde 49 toe aan deze variabele.
Print deze variabele af via de instructie PRINT naar het message venster

3. Geef i.p.v. PRINT de waarde terug via een SELECT query.
Wat is het verschil met het resultaat uit vorige vraag?

4. Declareer een variabele waarin het aantal Employees zal worden opgeslagen.
Schrijf een SELECT query waarbij het aantal rijen van de tabel Employees wordt toegekend aan deze variabele.

PRINT daarna de waarde af van de variabele in het message venster

En geef de waarde van de variabele ook terug via een SELECT

Operatoren in Transact SQL

- **Rekenkundigeoperatoren**

-, +, *, /, %(modulo)

- **Vergelijkingsoperatoren**

<, >, =, ... , IS NULL, LIKE, BETWEEN, IN

- **Alfanumerieke operatoren**

+ (stringconcatenatie)

- **Logische operatoren**

AND, OR, NOT

Operator EXISTS

- De EXISTS-operator wordt gebruikt om het bestaan van een record in een subquery te testen.

- De operator EXISTS geeft true terug als de subquery 1 of meer records teruggeeft.

EXISTS Syntaxis

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

Operator EXISTS

Syntaxis

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

```
USE [northwind]
--Geeft de lijst van
--leveranciers op met een productprijs lager dan 20:
SELECT *
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products
WHERE Products.SupplierID = Suppliers.supplierID
AND Products.UnitPrice < 20);
```

Operator ANY en ALL

De operators ANY en ALL worden gebruikt met een WHERE- of HAVING-clausule.

De operator **ANY** geeft true terug als **1** van de subquerywaarden aan de voorwaarde voldoet.

De operator **ALL** geeft true als **alle** subquerywaarden aan de voorwaarde voldoen.

```
USE [northwind]

SELECT ProductName
FROM Products
WHERE ProductID = ANY
(SELECT ProductID FROM [Order Details]
WHERE Quantity = 10);
```

Operator ANY

Syntaxis

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);
```

```
USE [northwind]

SELECT ProductName
FROM Products
WHERE ProductID = ANY
(SELECT ProductID FROM [Order Details]
WHERE Quantity = 10);
```

Operator ALL

Syntaxis

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);
```

```
USE [northwind]

SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM [Order Details]
WHERE Quantity = 10);
```

Built-in Functions in T-SQL

Functies

Numerieke : `ROUND, POWER, COS, ...`

Datum/tijd: `DATEADD, DATEDIFF, GETDATE, DAY, MONTH,...`

Alfanumerieke: `LEFT, RIGHT, LTRIM, RTRIM, TRIM, REPLACE, UPPER, LOWER,...`

Systeem functies: `CAST, CONVERT, ISNUMERIC, ISDATE, PRINT,...`

https://www.w3schools.com/sql/sql_ref_sqlserver.asp

=> Voor een overzicht van de functies zie ook help/online books

Built-in Functies in T-SQL: convert

Voorbeeld:

```
DECLARE @custID VARCHAR(40), @rijtelling INT
SET @custID ='ROMEY'

SELECT @rijtelling = count(*) FROM Orders
WHERE CustomerID = @custID;

PRINT 'rijtelling ' + convert(varchar(8),@rijtelling)

SELECT 'rijtelling ' + convert(varchar(8),@rijtelling) AS result
```

Control flow met Transact SQL

Programma verloop kan je bepalen via o.a.

- **Instructie niveau**

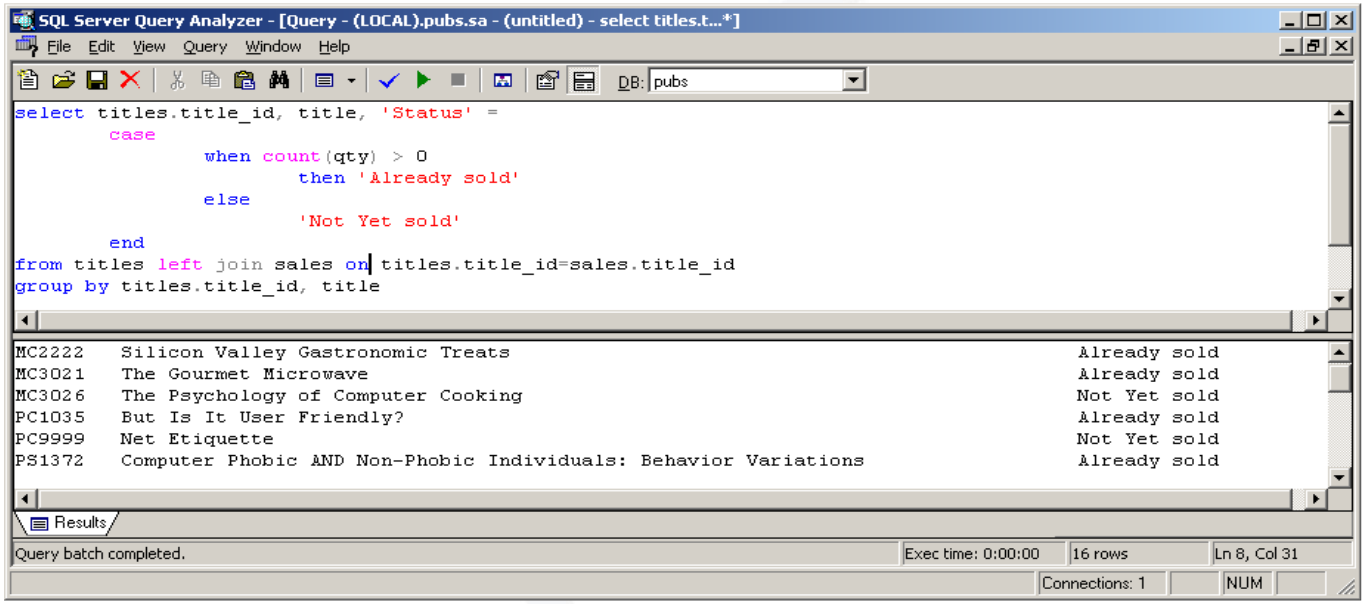
BEGIN ... END
IF .... ELSE
WHILE ...
break
continue
RETURN

- **Rij –niveau**

CASE ... END

```
if exists(select title_id from titles where pub_id=1)
        PRINT '*** Publisher cannot be deleted***'
else
        begin
                delete publishers where pub_id=1
                print '*** Publisher has been deleted***'
        end
```

```
*** Publisher has been deleted***
```



west-vlaanderen POM

SQL IF…END

Syntaxis

```
IF condition1
  BEGIN
          statement1;
          statement2
          …
  END
ELSE
  BEGIN
          statement1;
          statement2
          …

  END ;
```

```
IF condition1
   statement1;
ELSE
   statement2;
GO
```

```
DECLARE @Number INT;
SET @Number = 50;
IF @Number > 100
   PRINT 'The number is large.';
ELSE
   BEGIN
      IF @Number < 10
      PRINT 'The number is small.';
   ELSE
      PRINT 'The number is medium.';
   END ;
```

SQL WHILE

Syntaxis

```
WHILE condition
BEGIN
  sql_statement(s) | BREAK | CONTINUE

END
```

```
WHILE ( SELECT AVG(ListPrice) FROM dbo.DimProduct) < 300
BEGIN
    UPDATE dbo.DimProduct
        SET ListPrice = ListPrice * 2;
    SELECT MAX(ListPrice) FROM dbo.DimProduct
    IF ( SELECT MAX(ListPrice) FROM dbo.DimProduct) > 500
        BREAK;
END ;
```

SQL CASE Statement

Syntaxis

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

```
USE [northwind];
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'hoeveelheid is groter dan 30'
    WHEN Quantity = 30 THEN 'hoeveelheid is gelijk aan 30'
    ELSE 'hoeveelheid is kleiner dan 30'
END AS QuantityText
FROM [Order Details];
```

Foutafhandeling met Transact SQL

`RETURN`

–onmiddellijke beëindiging van de batch of procedure

`@@error`

–bevat de fout van de laatst uitgevoerde instructie

–waarde is 0 indien OK

`RAISERROR`

–retourneren van een user definedfout of systeemfout

Foutafhandeling met Transact SQL- voorbeeld

Voorbeeld

```
UPDATE HumanResources.EmployeePayHistory
SET PayFrequency = 4
WHERE BusinessEntityID = 1;

IF @@ERROR  = 547
        BEGIN PRINT 'A check constraint violation occurred.';
END GO
```

SQL – SAMENVATTING

- **Views**
- **T-SQL**
  - **Built-in functions**
  - **Variabelen**
  - **IF .... ELSE....**
  - **CASE**

VRAGEN?