



SQL Server 2005

Neuerungen in T-SQL



SQL Server 2005

Agenda

- neue Datentypen
- neue T-SQL Befehle
- Fehlerbehandlung



SQL Server 2005

neue Datentypen - Large-Value Data Types

- `varchar(max)` und `nvarchar(max)`
- `varbinary(max)`
- können in Variablen verwendet werden
- können als Parameter verwendet werden
- bis zu 2 GByte
- `varchar(max)` und `nvarchar(max)` unterstützen String-Funktionen
- Sonderfunktionen wie `UPDATETEXT` u.a. werden nicht mehr benötigt



SQL Server 2005

Ranking - row_number()

- liefert zu jeder Zeile des Resultsets eine fortlaufende Nummer
- PARTITION liefert zu jeder Gruppe eine fortlaufende Nummer

```
SELECT  LastName, FirstName,
        row_number() OVER
          (ORDER BY Lastname, Firstname) AS Num,
        row_number() OVER
          (PARTITION BY LastName ORDER BY LastName) AS NumPart
FROM    Person.Contact As x
ORDER   BY LastName, FirstName
```



SQL Server 2005

Ranking – Rank()

- liefert die jeweiligen Zeilennummern je ausgegebener Zeile
- bei Gruppierung wird die Zeilennummer der ersten Zeile der Gruppe bei den anderen Zeilen beibehalten
- bei Gruppenwechsel wird mit der eigentlichen Zeilennummer fortgefahren

```
SELECT LastName, FirstName,  
       RANK() OVER (order by LastName) as r_rank  
FROM   Person.Contact As x  
ORDER  BY Lastname, Firstname
```



SQL Server 2005

Ranking – DENSE_RANK

- liefert die jeweiligen Zeilennummern je ausgegebener Zeile

```
SELECT  Lastname, Firstname,  
        DENSE_RANK() OVER (ORDER BY Lastname) As r_dense_rank  
FROM    Person.Contact As PersonRank  
ORDER  BY Lastname, Firstname
```



SQL Server 2005

Ranking – NTILE()

- verteilt die Zahlen der Nummerierung in die angegebene Anzahl der Gruppen
 - ntile(4) liefert 4 Ziffern und somit 4 Gruppen

```
SELECT  Lastname, Firstname,  
        NTILE(4) OVER (ORDER BY Lastname, Firstname) As r_ntile  
FROM    Person.Contact as RankNtile  
ORDER   BY Lastname, Firstname
```



SQL Server 2005

TOP (expression)

- (expression) kann eine Variable sein

```
DECLARE @topN As int
SET @topN =10
SELECT TOP(@topN) * FROM HumanResources.Employee
```




SQL Server 2005

TOP (expression)

- neben SELECT auch bei UPDATE, INSERT, DELETE möglich
- Vorsicht!
Bei UPDATE, INSERT, DELETE gibt TOP(n) die Zeilen nach einem Zufallsprinzip zurück. Es sind nicht die ersten n Datensätze!

```
INSERT TOP (2) INTO Table2 (ColumnB)  
SELECT ColumnA FROM Table1 ORDER BY ColumnA
```

- Liefert NICHT die ersten beiden Datensätze

```
INSERT INTO Table2 (ColumnB)  
SELECT TOP (2) ColumnA FROM Table1 ORDER BY ColumnA
```

- Liefert die ersten beiden Datensätze



SQL Server 2005

CTE (Common Table Expression)

- Möglichkeit eine Abfrage in einem SQL-Statement zu definieren
- Abfrage -Definition wird nur im Ausführungskontext des SQL-Statements genutzt
- Abfrage -Definition wird nicht permanent gespeichert
- temporäres Resultset („adhoc-Views“)
- Ersatz für gespeicherte Views
- bessere Lesbarkeit
- vereinfachte Wartbarkeit von komplexen Abfragen
- gut geeignet für rekursive Abfragen
- kann mehrfach in derselben Abfrage referenziert werden



SQL Server 2005

CTE (Common Table Expression)

- die Struktur einer CTE besteht aus:
 - dem Namen des Ausdrucks
 - einer optionalen Spaltenliste
 - einem SQL-Statement mit
SELECT, INSERT, UPDATE, DELETE, CREATE VIEW

```
WITH cte (EmpId, MgrId, Title, Gender) AS
(
    SELECT E.EmployeeId, E.ManagerId, E.Title, E.Gender
    FROM HumanResources.Employee As E
)

SELECT * FROM cte ORDER BY mgrid
```



SQL Server 2005

rekursive CTE (Common Table Expression)

- CTE's können rekursiv genutzt werden
- die SQL-Statements müssen dieselbe Spaltenanzahl haben
- die Spalten müssen dieselben Datentypen haben
- Rekursion wird beendet, wenn
 - das zweite SELECT-Statement keine Ergebnisse mehr liefert
 - nach 100 Rekursionen, sofern nicht MAXRECURSION = 0 definiert ist (0 = unendlich)
 - die in der Option MAXRECURSION definierte Rekursionsebene erreicht ist



SQL Server 2005

rekursive CTE (Common Table Expression)

- Struktur einer rekursiven CTE:
 - <nicht-rekursives SELECT>
 - muss immer vor dem rekursiven CTE-Aufruf stehen
 - UNION ALL
 - <SELECT mit Referenz auf CTE>
 - rekursive Elementdefinition, die sich selbst referenziert



SQL Server 2005

OUTPUT

- das Resultset einer DML-Anweisung wird ausgegeben
- gilt für INSERT, DELETE, UPDATE
- Ausgabe an die Anwendung
- Ausgabe in Tabellen
- Ausgabe in Variablen
- dazu werden die Tabellen „inserted“ und „deleted“ genutzt
- OUTPUT funktioniert auch in Triggern
 - per sp_configure und der Option DISALLOW RESULTS FROM TRIGGERS kann OUTPUT in Triggern deaktiviert werden; OUTPUT erzeugt dann einen Fehler.



SQL Server 2005

APPLY

- In einem SQL-Statement können einer Table-Valued-Function Daten aus dem Resultset des SQL-Statements als Parameter übergeben werden.
- Das Resultset der Abfrage wird durch die Spalten der Table-Valued-Function erweitert.
- Das Resultset der Abfrage wird durch die Zeilen der Table-Valued-Function erweitert.



SQL Server 2005

APPLY

■ CROSS APPLY

- gibt keine Zeile zurück, wenn right_table_source kein Ergebnis liefert
- entspricht einem INNER JOIN

■ OUTER APPLY

- gibt für left_tables_source Zeilen zurück, auch wenn right_table_source kein Ergebnis liefert
- entspricht einem LEFT JOIN



SQL Server 2005

APPLY

- SELECT-Statement liefert Daten der Tabelle Person.Contact
- UDF bereitet zu jeder übergebenen ContactId Daten auf, mit denen das Resultset ergänzt wird

```
SELECT * FROM Person.Contact as C  
CROSS APPLY  
ufnGetContactInformation(c.contactid) as ST
```



SQL Server 2005

PIVOT

- Beispiel mit CTE:

```
WITH cte (ProductId, Name, theYear, TotalDue) AS
  (SELECT P.ProductId, P.Name, Year(O.OrderDate) As theYear, O.TotalDue
   FROM Production.Product As P
   INNER JOIN Sales.SalesOrderDetail As OD ON OD.ProductId = P.ProductId
   INNER JOIN Sales.SalesOrderHeader As O ON O.SalesOrderId = OD.SalesOrderId
   WHERE P.ProductId BETWEEN 779 AND 784)

SELECT ProductId, Name, [2002], [2003], [2004] FROM cte
PIVOT (SUM(TotalDue) FOR theYear IN ([2002], [2003], [2004])) As pvt
ORDER BY ProductId
```



SQL Server 2005

UNPIVOT

- kehrt PIVOT wieder um

```
SELECT  Year, Quarter, Amount
FROM    Piv
UNPIVOT (Amount FOR Quarter In (Q1, Q2, Q3, Q4) As UnPiv
```



SQL Server 2005

UNPIVOT

- andere Möglichkeiten mit UNPIVOT
 - liefert eine Liste mit allen Vornamen aus den Spalten FirstName und MiddleName:

```
SELECT DISTINCT AlleVornamen FROM
(SELECT * FROM Person.Contact
UNPIVOT
(AlleVornamen FOR NameType
IN (MiddleName, FirstName)) AS PVT) As Names
ORDER BY AlleVornamen
```



SQL Server 2005

EXCEPT – INTERSECT

- Vergleich von Daten auf Basis mehrerer Datenmengen
- Verwendung und Regeln entsprechen UNION
 - gleiche Anzahl Spalten
 - Spalten müssen vom gleichen Datentyp sein
- Es werden nur DISTINCT Werte geliefert.
- INTERSECT entspricht WHERE EXISTS bzw. INNER JOIN.
- EXCEPT entspricht WHERE NOT EXISTS.
- ORDER BY, GROUP BY und HAVING können weiterhin genutzt werden.



SQL Server 2005

DDL–Trigger

- Trigger für Data Definition Language
- Änderungen an Datenbank-Objekten können Trigger auslösen
 - CREATE, ALTER, DROP, GRANT, DENY, REVOKE

```
CREATE TRIGGER FingerWeg
ON DATABASE FOR DROP_TABLE, ALTER_TABLE
AS
PRINT 'Finger weg'
ROLLBACK
```



SQL Server 2005

Indizes – INCLUDE

- nonclustered Index kann um Spalten erweitert werden
 - Index beinhaltet neben den indizierten Spalten noch weitere Spalten
 - durch die zusätzlichen Informationen kann ein Zugriff von der Indexpage auf die Datapage vermieden werden
 - Die zusätzliche Spalten werden nicht durchsucht!

```
CREATE INDEX IX_Customers_Names  
ON Customers (CompanyName, Contactname)  
INCLUDE (Address, City)
```



SQL Server 2005

Fehlerbehandlung

- es gibt eine Fehlerbehandlung!
 - ... für nicht kritische Fehler
- mit BEGIN TRY ... END TRY und BEGIN CATCH ... END CATCH
 - BEGIN TRY ... END TRY beinhaltet die eigentliche Anweisung
 - BEGIN CATCH ... END CATCH beinhaltet die Fehlerbehandlung
- Im CATCH-Block stehen folgende Systemfunktionen zur Verfügung:
 - Error_Number() = Fehlernummer
 - Error_Message() = ausführliche Fehlermeldung
 - Error_Severity() = Schweregrad des Fehlers
 - Error_State() = Fehlerstatusnummer
 - Error_Line() = Zeilennummer der fehlerhaften Zeile der Routine
 - Error_Procedure() = Name der Stored Procedure oder des Triggers



SQL Server 2005

Danke für die Aufmerksamkeit ☺