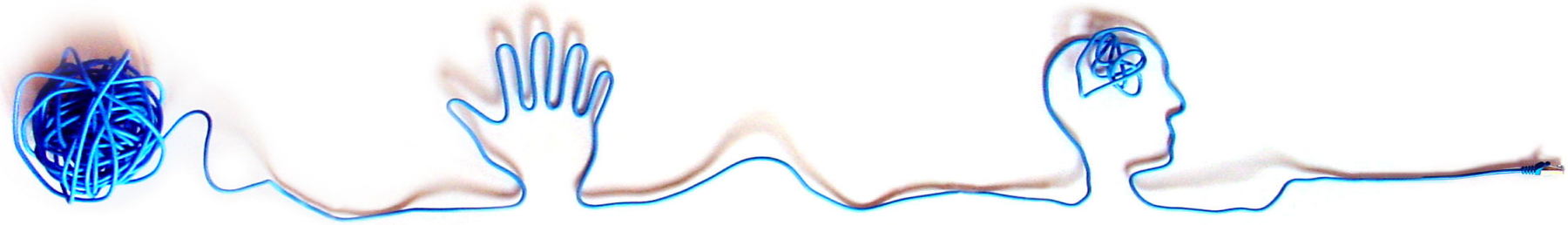
SYSTEMINTEGRATION

Know-how in gns IT-Services

SQL Server 2005 aus Sicht von Entwicklern

Uwe Ricken
GNS GmbH



Know-how in gns IT-Services

Agenda

- Welche Version für welche Ansprüche
- SQL Server Management Studio und T-SQL
- T-SQL Erweiterungen
- Sicherheit

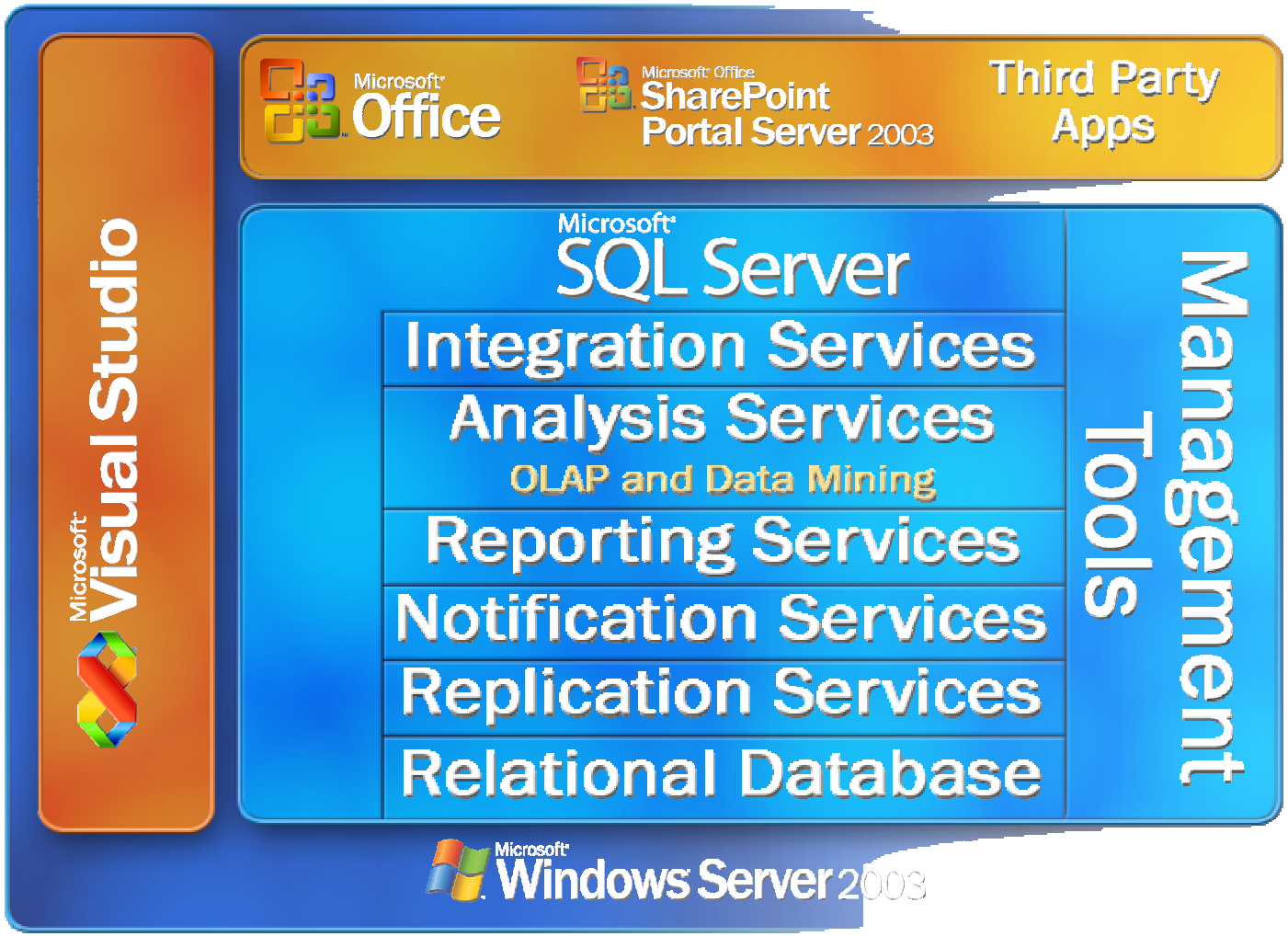
Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Welche Version für welche Ansprüche

Feature	Express	Workgroup	Standard	Enterprise
CPU	1	2	4	--
RAM	1 GB	3 GB	--	--
DB Size	4 GB	--	--	--
DB Mirroring			X	X
Express Manager	X	X	X	X
Management Studio		X	X	X
SQL Agent		X	X	X
Notification Services			X	X
ORACLE Replication				X
BI Services			X	X

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

SQL Server 2005 Plattform



© GNS GmbH; Robert-Bosch-Straße 8; 63477 Maintal

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Administrative Tools

- **SQL Server Management Studio**
  - Ersetzt Enterprise Manager and Query Analyzer
- **SQL Computer Manager**
  - Ersetzt Service Manager
- **SQLCMD utility**
  - Ersetzt ISQL and OSQL
- **SMO**
  - Ersetzt of DMO

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

SQL Server Management Studio und T-SQL

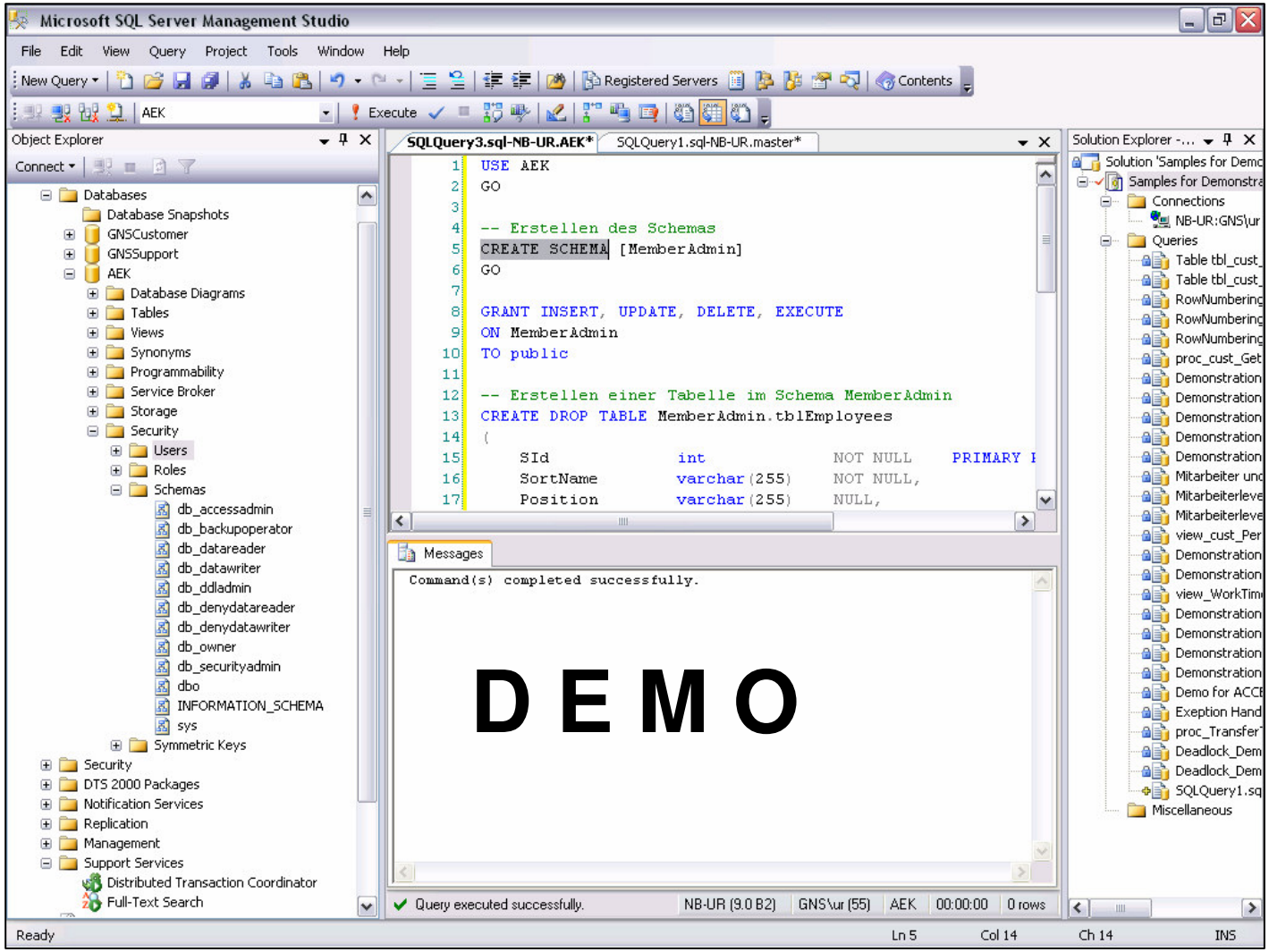
- **Was ist SQL Server Management Studio**

(Neue Besen kehren gut – alte Funktionalität in neuem Gewand?)

- Neue umfassende GUI
- Integrierte Entwicklung UND Verwaltung
- Unterstützung ALLER SQL-Server Komponenten
- Verwaltung mehrerer SQL-Server
- Ersatz für EM und QA
- Integration von Visual Source Safe

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

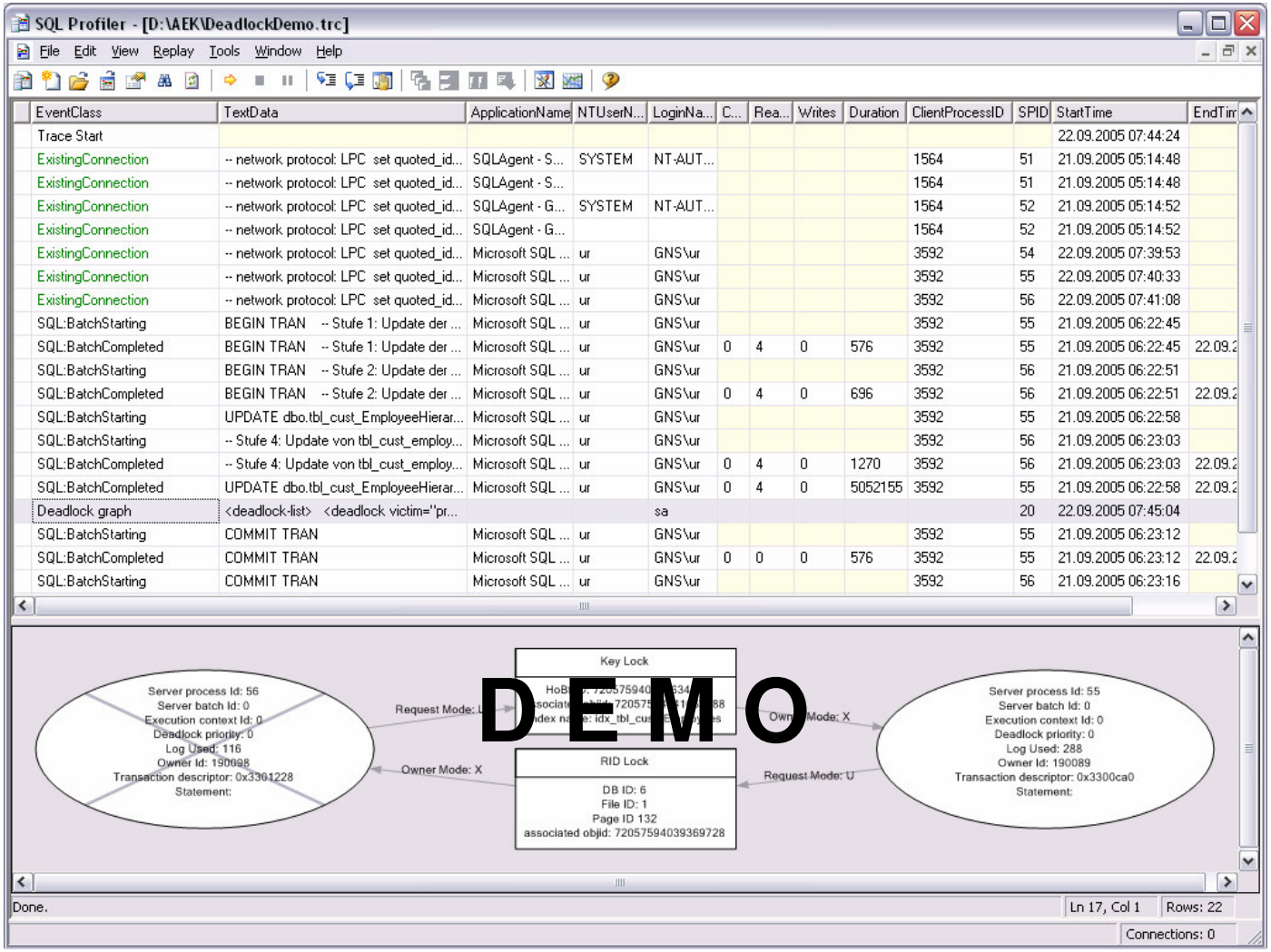
SQL Server Management Studio und T-SQL



© GNS GmbH; Robert-Bosch-Straße 8; 63477 Maintal

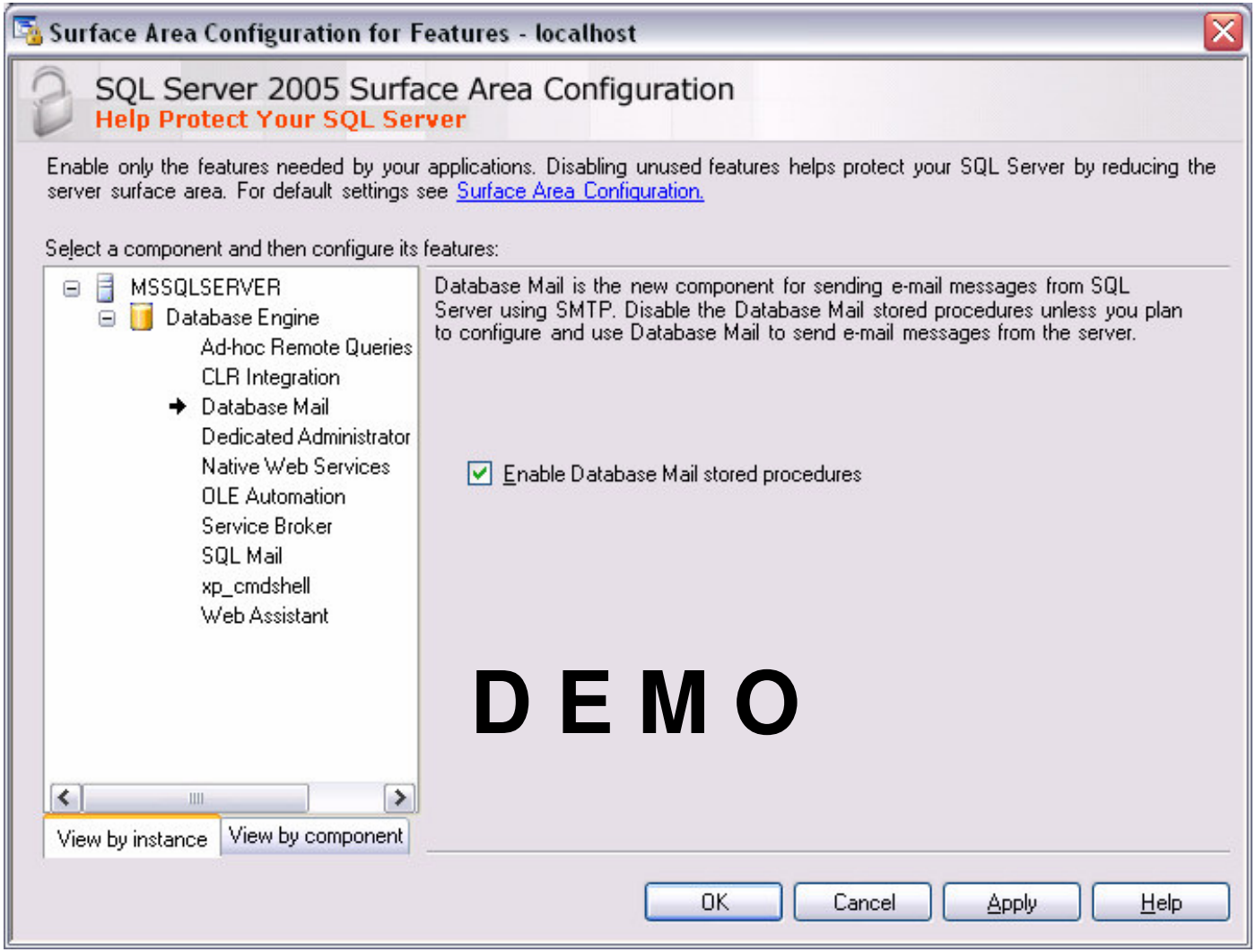
Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

SQL Server Profiler



Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

SQL Server Surface Manager



Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Erweiterungen in T-SQL

- Ranking Funktionen
- Neue Datentypen sind vorhanden
- Verwendung von Common Table Expression (CTE)
- Neue SET Operatoren (EXCEPT und INTERSECT)
- Erstellung von PIVOT Tabellen
- DML (Data Manipulation Language) mit Ausgabemöglichkeiten
- Verbesserung der TOP-Klausel
- Fehlerbehandlung in Stored Procedures ist möglich

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Ranking Functions – ROW_Number()

- Erstellt für jede Zeile eine sequentielle Zeilennummer
- Erstellung mit SQL-Server 2000 nur mittels „Derived Table“ möglich

```
-- Numerierung Pre SQL 2005
SELECT
          (
           SELECT COUNT(e1.SId) FROM dbo.tbl_cust_Employees AS e1
           WHERE e1.SortName <= e2.SortName
          ) AS RowNumber,
          e2.SortName,
          e2.Position
FROM      dbo.tbl_cust_Employees e2
ORDER BY
          e2.SortName
```

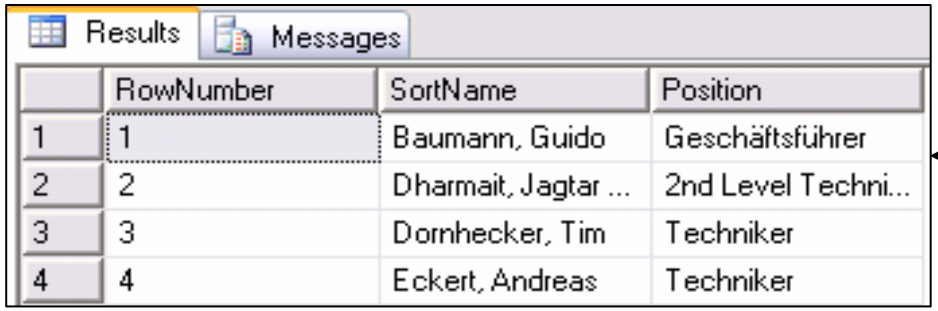
Results | Messages

	RowNumber	SortName	Position
1	1	Baumann, Guido	Geschäftsführer
2	2	Dharmait, Jagtar ...	2nd Level Techni...
3	3	Dornhecker, Tim	Techniker
4	4	Eckert, Andreas	Techniker

Ranking Functions – ROW_Number()

- **Erstellt für jede Zeile eine sequentielle Zeilennummer**
- **Erstellung mit SQL-Server 2005 mittels Funktion ROW_NUMBER()**

```
-- Numerierung SQL 2005
SELECT    ROW_Number() OVER (ORDER BY SortName) AS RowNumber,
          SortName,
          Position
FROM      dbo.tbl_cust_Employees
```



Results | Messages

	RowNumber	SortName	Position
1	1	Baumann, Guido	Geschäftsführer
2	2	Dharmait, Jagtar ...	2nd Level Techni...
3	3	Dornhecker, Tim	Techniker
4	4	Eckert, Andreas	Techniker

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Ranking Functions – ROW_Number()

- **Filtern von Zeilen auf Basis der Zeilennummer mit Gruppierungen**

```
-- Numerierung SQL 2005
SELECT    ROW_Number() OVER (PARTITION BY Position ORDER BY SortName) AS
Row,
          SortName,
          Position
FROM      dbo.tbl_cust_Employees
```

2	2	Küchenberg, Wal...	NULL
3	3	Kunert, Werner	NULL
4	4	Silberhorn, Norman	NULL
5	1	Dharmait, Jagtar ...	2nd Level Techni...
6	1	Stumpf, Miriam	Assistentin der Ge...
7	1	Müller, Michaela	Auszubildende Se...
8	1	Mierke, Stefan	Auszubildender

Ranking Functions – ROW_Number()

- **Filtern von Zeilen auf Basis der Zeilennummer**

```
SELECT    ROW_NUMBER() OVER (ORDER BY SortName) AS RowNumber,
          SortName,
          Position
FROM      dbo.tbl_cust_Employees
WHERE     RowNumber BETWEEN 5 AND 10
```

Die obige Abfrage kann nicht funktionieren, da die Filterung der Zeilen in der WHERE-Klauses stattfindet. Die WHERE-Klauses wird jedoch vor der Ermittlung der Zeilennummern ausgeführt. Zu diesem Zeitpunkt existieren die Zeilennummern noch nicht und die Abfrage wird mit einem Fehler beendet

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Ranking Functions – ROW_Number()

- **Filtern von Zeilen auf Basis der Zeilennummer mittels „Derived Table“**

```
-- Filtern von Datensätzen auf Basis von ermittelten Zeilennummern
-- mittels Verwendung einer Derived Table
SELECT * FROM
            (
             SELECT    ROW_NUMBER() OVER (ORDER BY SortName) AS RowNumber,
                       SortName,
                       Position
             FROM      dbo.tbl_cust_Employees
            ) AS Result
WHERE       RowNumber BETWEEN 5 AND 10
ORDER BY
            Position DESC
```

Mit Hilfe einer „Derived Table“ kann dieses Problem umgangen werden und die Daten basierend auf einem Filter der Zeilennummern ermittelt werden.

Eine weitere Möglichkeit besteht in der Verwendung einer CTE (Common Table Expression)

Ranking Functions – ROW_Number()

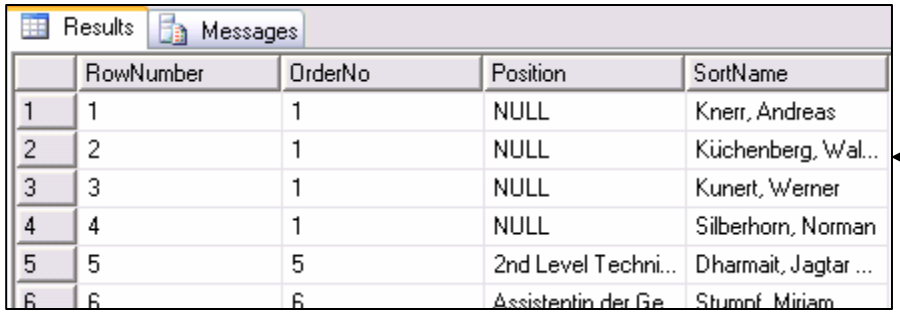
- **Beispiel für die Verwendung von ROW_NUMBER in einer Stored Procedure**

```
CREATE PROC dbo.proc_cust_GetEmployees
    @StartNumber   int,
    @RowCount      int
AS
    SET NOCOUNT ON

    SELECT * FROM
    (
     SELECT ROW_NUMBER() OVER (ORDER BY SortName) AS RowNumber,
          SortName,
          Position
     FROM dbo.tbl_cust_Employees
    ) AS Result
    WHERE RowNumber BETWEEN @StartNumber AND @StartNumber + @RowCount - 1
    ORDER BY SortName

    SET NOCOUNT OFF
GO
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Ranking Functions – RANK()

```
/*
    Beispiel für die Verwendung von Ranking Funktionen

    RANK(): Liefert den Rang von jeder Zeile innerhalb
            einer Partition eines
            Resultsets.
*/

SELECT    ROW_NUMBER() OVER (ORDER BY Position)AS RowNumber,
          RANK() OVER (ORDER BY Position)      AS OrderNo,
          Position,
          SortName
FROM      dbo.tbl_cust_Employees
```



Results | Messages

	RowNumber	OrderNo	Position	SortName
1	1	1	NULL	Knerr, Andreas
2	2	1	NULL	Küchenberg, Wal...
3	3	1	NULL	Kunert, Werner
4	4	1	NULL	Silberhorn, Norman
5	5	5	2nd Level Techni...	Dharmait, Jagtar ...
6	6	6	Assistentin der Ge...	Stumpf, Miriam

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Ranking Functions – DENSE_RANK()

```
/*
   Beispiel für die Verwendung von Ranking Funktionen

   DENSE_RANK(): Liefert den Rang einer Zeile innerhalb einer Gruppe
                 von Records (Partition) in fortlaufender Numerierung
*/

SELECT    ROW_NUMBER() OVER (ORDER BY Position) AS RowNumber,
     DENSE_RANK() OVER (ORDER BY Position) AS Rank,
     Position,
     SortName
FROM dbo.tbl_cust_Employees
```

Results | Messages

	RowNumber	Rank	Position	SortName
1	1	1	NULL	Knerr, Andreas
2	2	1	NULL	Küchenberg, Wal...
3	3	1	NULL	Kunert, Werner
4	4	1	NULL	Silberhorn, Norman
5	5	2	2nd Level Techni...	Dharmait, Jagtar ...
6	6	3	Assistentin der Ge...	Stumpf, Miriam

Ranking Functions – NTile(@n)

```
/*
    Beispiel für die Verwendung von Ranking Funktionen
    NTile(): Veröffentlicht Datenreihen in in einer Anordnung
*/

SELECT    ROW_NUMBER() OVER (ORDER BY Position) AS RowNumber,
          NTile(3) OVER (ORDER BY Position) AS Rank,
          Position,
          SortName
FROM      dbo.tbl_cust_Employees
```

Results | Messages

	RowNumber	Rank	Position	SortName
7	7	1	Auszubildende Se...	Müller, Michaela
8	8	1	Auszubildender	Nicotera, Pascal
9	9	1	Auszubildender	Wojnar, Alexander
10	10	2	Auszubildender	Mierke, Stefan
11	11	2	Freiberuflicher Ent...	Marktscheffel, Pe...

Neue Datentypen

- **SQL Server 2005 unterstützt drei weitere Datentypen für mehr Flexibilität**
  - **varchar (max)**
  - **nvarchar (max)**
  - **varbinary (max)**
- **Die neuen Datentypen wurden erweitert, um bis zu 2 GByte Daten zu speichern**
- **Daten müssen nicht mehr „stückweise“ ein- und ausgelesen werden**
- **Die bisherigen Datentypen werden in weiteren Versionen von SQL-Server nicht mehr unterstützt**

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Neue Datentypen

```
/*
    Beispiel für die Vewendung der neuen Datentypen
    - varchar (max)
    - nvarchar (max)
    - varbinary (max)
*/

DECLARE   @vmax varchar(max)
SET       @vmax = REPLICATE('a', 16000)

SELECT    SUBSTRING(@vmax, 9000, 10)  AS PartOfString,
          LEN(@vmax)                  AS LengthOfString


SET       @vmax = REPLICATE('a', 8000)
SET       @vmax = @vmax + REPLICATE('b', 8000)
SET       @vmax = @vmax + REPLICATE('c', 8000)

SELECT    SUBSTRING(@vmax, 17000, 30), LEN(@vmax)
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

CTE – Common Table Expression

- **Common Table Expression (CTE) ist ein neues Feature in Microsoft SQL-Server 2005, das es ermöglicht, mit „virtuellen“ Tabellen in anderen DML-Statements zu arbeiten.**
- **Funktionalität ist ähnlich der Verwendung von „Derived Tables“**
- **Besondere Verwendung von CTE im Zusammenhang mit rekursiven Abfragen**
- **Folgende Statements dürfen in CTE‘s nicht verwendet werden:**
  - **COMPUTE oder COMPUTE BY**
  - **ORDER BY (außer in Verbindung mit TOP)**
  - **INTO**
  - **OPTION –Klausel mit Abfragehinweisen**

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

CTE – Common Table Expression

```
-- Beispiel für die Vewendung von CTE (1)
WITH R (RowNumber, Position, SortName)
AS (
          SELECT    ROW_NUMBER() OVER (ORDER BY SortName),
                    Position,
                    SortName
          FROM      dbo.tbl_cust_Employees
          )
SELECT * FROM R
WHERE     RowNumber BETWEEN 5 AND 10
```

Results | Messages

	RowNumber	Position	SortName
1	5	Leiter Entwicklung	Hufschmidt, Tho...
2	6	Techniker	Junker, Lars
3	7	NULL	Knerr, Andreas
4	8	NULL	Küchenberg, Wal...
5	9	Techniker	Kühlwei, Matthias
6	10	NULL	Kunert, Werner

CTE – Common Table Expression (Rekursionen)

Results | Messages

	SId	SortName	Position	ManagerId	InsertUser	InsertDate
1	3	Baumann, Guido	Geschäftsführer	NULL	GNS\ur	2005-09-21 07:19...
2	292	Dharmait, Jagtar ...	2nd Level Techni...	3	GNS\ur	2005-09-21 07:19...
3	6142	Dornhecker, Tim	Techniker	303	GNS\ur	2005-09-21 07:19...

```
/*
   Problemstellung: Zeige alle (!!!) Mitarbeiter und deren Vorgesetzte
   Das nachfolgende SQL-Statement zeigt Unternehmenshierarchien bis in
   den dritten Level
*/

SELECT    e.SortName AS Mitarbeiter,
          m1.SortName AS Manager1,
          m2.SortName AS Manager2
FROM      dbo.tbl_cust_Employees e LEFT JOIN dbo.tbl_cust_Employees m1
          ON (e.ManagerId = m1.SId) LEFT JOIN dbo.tbl_cust_Employees m2
          ON (m1.ManagerId = m2.SId)
ORDER BY  Manager1
```

2	Ricken, Uwe	NULL	NULL
3	Kunert, Werner	Baumann, Guido	NULL
4	Dharmait, Jagtar ...	Baumann, Guido	NULL

CTE – Common Table Expression (Rekursionen)

```
-- Problemstellung: Zeige alle Mitarbeiter und deren Hierarchie-Level
-- im Unternehmen (bis maximal 3 Level!!!)

SELECT     SId, SortName, Position, ManagerId, MIN(Level) AS Level
FROM (
           SELECT     e3.SId, e3.SortName, e3.Position, e3.ManagerId, 1 AS Level
           FROM       dbo.tbl_cust_Employees e1 LEFT JOIN dbo.tbl_cust_Employees e2
                      ON (e1.ManagerId = e2.SId) LEFT JOIN dbo.tbl_cust_Employees e3
                      ON (e2.ManagerId = e3.SId)

           UNION

           SELECT     e2.SId, e2.SortName, e2.Position, e2.ManagerId, 2 AS Level
           FROM       dbo.tbl_cust_Employees e1 LEFT JOIN dbo.tbl_cust_Employees e2
                      ON (e1.ManagerId = e2.SId) LEFT JOIN dbo.tbl_cust_Employees e3
                      ON (e2.ManagerId = e3.SId)

           UNION

           SELECT     e1.SId, e1.SortName, e1.Position, e1.ManagerId, 3 AS Level
           FROM       dbo.tbl_cust_Employees e1 LEFT JOIN dbo.tbl_cust_Employees e2
                      ON (e1.ManagerId = e2.SId) LEFT JOIN dbo.tbl_cust_Employees e3
                      ON (e2.ManagerId = e3.SId)
           )          AS tempTable
WHERE      SId IS NOT NULL
GROUP BY SId, SortName, Position, ManagerId
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

CTE – Common Table Expression (Rekursionen)

```
-- Beispiel der Mitarbeiter und Hierarchie-Level mit CTE's

WITH el (SId, SortName, Position, Level)
AS
   (
         SELECT    SId, SortName, Position, 1  AS Level
         FROM      dbo.tbl_cust_Employees
         WHERE     ManagerId IS NULL

         UNION ALL

         SELECT    e.SId, e.SortName, e.Position, Level + 1
         FROM      dbo.tbl_cust_Employees e INNER JOIN el
                   ON (el.SId = e.ManagerId)
   )

SELECT SId, SortName, Position, Level
FROM el
ORDER BY Level, SortName
```

Results | Messages

	SId	SortName	Position	Level
1	3	Baumann, Guido	Geschäftsführer	1
2	2	Ricken, Uwe	Geschäftsführer	1
3	292	Dharmait, Jagtar ...	2nd Level Techni...	2

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

EXCEPT und INTERSECT Operatoren

- **Neue Operatoren in Microsoft SQL Server 2005**
- **Vergleich von Daten auf Basis mehrerer Datenmengen (Tabellen, Views, …)**
- **Regeln der Verwendung entsprechen den Regeln für den UNION-Operator**
  - Beide Datenmengen müssen die gleiche Anzahl von Attributen besitzen
  - Die zu vergleichenden Attribute müssen vom gleichen Datentypen sein
- **INTERSECT liefert die Daten, die in beiden Datenmengen vorhanden sind (INNER JOIN)**
- **EXCEPT liefert die Daten, die in der zu vergleichenden Datenmenge nicht vorhanden ist (LEFT JOIN / NOT EXISTS(**

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

INTERSECT Operator

```
/*
    Beispiel für die Verwendung von INTERSECT
    Problemstellung: Zeige alle Daten aus der Mitarbeitertabelle
    (tbl_cust_Employees) die einen identischen Eintrag in der
    Personentabelle besitzen
*/

-- Bisherige Möglichkeiten
SELECT    e.SId, e.SortName, e.Position, e.ManagerId
FROM      dbo.tbl_cust_Employees e INNER JOIN dbo.view_cust_Persons vp
          ON (
                    e.SortName = vp.SortName AND
                    ISNULL(e.Position, '') = ISNULL(vp.Position, '')
             )

SELECT    e.SortName, e.Position
FROM      dbo.tbl_cust_Employees e
WHERE     EXISTS (
                    SELECT    *
                    FROM      dbo.view_cust_Persons vp
                    WHERE     vp.SortName = e.SortName AND
                              ISNULL(e.Position, '') =
                              ISNULL(vp.Position, '')
                    )
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

INTERSECT Operator

```
/*
    Beispiel für die Verwendung von INTERSECT
    Problemstellung: Zeige alle Daten aus der Mitarbeitertabelle
    (tbl_cust_Employees) die einen identischen Eintrag in der
    Personentabelle besitzen
*/

-- Verwenduung von INTERSECT
SELECT    e.SortName, e.Position FROM dbo.tbl_cust_Employees e
INTERSECT
SELECT    v.SortName, v.Position FROM dbo.view_cust_Persons v
```

Results | Messages

	SortName	Position
1	Baumann, Guido	Geschäftsführer
2	Dharmait, Jagtar ...	2nd Level Techni...
3	Dornhecker, Tim	Techniker
4	Eckert, Andreas	Techniker
5	Hufschmidt, Thom	Leiter Entwicklung

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

EXCEPT Operator

```
/*
    Beispiel für die Verwendung von EXCEPT
    Problemstellung: Es sollen aus der Mitarbeitertabelle alle Personen
    gezeigt werden, die NICHT in der Personentabelle vorhanden sind
*/

-- Bisherige Möglichkeiten
SELECT    e.SId, e.SortName, e.Position, e.ManagerId
FROM      dbo.tbl_cust_Employees e LEFT JOIN dbo.view_cust_Persons vp
          ON (
                    e.SortName = vp.SortName AND
                     ISNULL(e.Position, '') = ISNULL(vp.Position, '')
             )
WHERE vp.SId IS NULL

SELECT    e.SortName, e.Position
FROM      dbo.tbl_cust_Employees e
WHERE     NOT EXISTS (
                    SELECT    *
                    FROM      dbo.view_cust_Persons vp
                    WHERE     vp.SortName = e.SortName AND
                              ISNULL(e.Position, '') =
                              ISNULL(vp.Position, '')
                    )
```

EXCEPT Operator

```
/*
    Beispiel für die Verwendung von EXCEPT
    Problemstellung: Es sollen aus der Mitarbeitertabelle alle Personen
    gezeigt werden, die NICHT in der Personentabelle vorhanden sind
*/

-- Verwenduung von EXCEPT
SELECT    e.SortName, e.Position FROM dbo.tbl_cust_Employees e
EXCEPT
SELECT    v.SortName, v.Position FROM dbo.view_cust_Persons v
```

Results | Messages

	SortName	Position
1	DeBetta, Peter	Senior Developer
2	Donaubauer, Karl	Senior Developer
3	Ferrari, Enzo	Senior Developer

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

PIVOT-Tabellen

```
/*
          Beispiel für die Erstellung einer PIVOT-Tabelle
          nach alten "Spielregeln"
*/

SELECT Kunde,
       SUM(CASE WHEN Jahr = 2004 THEN Arbeitszeit ELSE 0 END) AS [2004],
       SUM(CASE WHEN Jahr = 2005 THEN Arbeitszeit ELSE 0 END) AS [2005]
FROM   dbo.view_WorkTime
WHERE  Jahr BETWEEN 2004 AND 2005
GROUP BY Kunde
```

Results | Messages

	Kunde	2004	2005
1	Motorola GmbH	1065	390
2	amcham.ch | Swis...	2995	1249
3	Brockhaus Privat...	0	130
4	Dieffenbacher Gm...	410	55

PIVOT-Tabellen

```
SELECT    Kunde,
          SUM(CASE WHEN Monat = 1 THEN Arbeitszeit ELSE 0 END) AS [Jan],
          SUM(CASE WHEN Monat = 2 THEN Arbeitszeit ELSE 0 END) AS [Feb],
          SUM(CASE WHEN Monat = 3 THEN Arbeitszeit ELSE 0 END) AS [Mae],
          SUM(CASE WHEN Monat = 4 THEN Arbeitszeit ELSE 0 END) AS [Apr],
          SUM(CASE WHEN Monat = 5 THEN Arbeitszeit ELSE 0 END) AS [Mai],
          SUM(CASE WHEN Monat = 6 THEN Arbeitszeit ELSE 0 END) AS [Jun]
FROM      dbo.view_WorkTime
WHERE     Jahr = 2004
GROUP BY Kunde
ORDER BY Kunde
```

Results | Messages

	Kunde	Jan	Feb	Mae	Apr	Mai	Jun
1	amcham.ch | Swiss-American Chamber of ...	60	10	40	75	60	25
2	amcham.de | American Chamber of Comm...	115	2190	640	1705	1100	1480
3	amchamfrance | American Chamber of Co...	85	120	135	30	10	0
4	Behindertenwerkstatt e.V.	0	0	210	735	0	180
5	Dieffenbacher GmbH & CO	0	0	50	0	0	0
6	Downer & Company	0	0	0	150	0	0
7	DWS Investment GmbH	0	0	0	0	0	975

PIVOT-Tabellen

```
SELECT * FROM dbo.view_WorkTime
PIVOT (
        SUM([ArbeitsZeit])
        FOR Mitarbeiter IN ([Ricken, Uwe], [Baumann, Guido])
      ) AS PVT
WHERE [Ricken, Uwe] IS NOT NULL OR
      [Baumann, Guido] IS NOT NULL
ORDER BY Kunde, Jahr
```

Results | Messages

	Kundennummer	Kunde	MA_Nummer	Monat	Jahr	Ricken, Uwe	Baumann, Gu
1	413	amcham.be | Ame...	2	2	2005	1560	NULL
2	508	amcham.ch | Swis...	2	4	2005	20	NULL
3	29	amcham.de | Ame...	2	2	2004	960	NULL
4	29	amcham.de | Ame...	2	3	2004	195	NULL
5	29	amcham.de | Ame...	2	4	2004	1245	NULL
6	29	amcham.de | Ame...	2	5	2004	75	NULL
7	29	amcham.de | Ame...	2	6	2004	465	NULL

PIVOT-Tabellen

```
SELECT Kunde, Jahr, [Ricken, Uwe], [Baumann, Guido]
FROM dbo.view_WorkTime
PIVOT(
          SUM([ArbeitsZeit])
          FOR Mitarbeiter IN ([Ricken, Uwe], [Baumann, Guido])
     ) AS PVT
WHERE     [Ricken, Uwe] IS NOT NULL OR
          [Baumann, Guido] IS NOT NULL
ORDER BY Kunde, Jahr
```

Results | Messages

	Kunde	Jahr	Ricken, Uwe	Baumann, Guido
1	amcham.be | Ame...	2005	1560	NULL
2	amcham.ch | Swis...	2005	20	NULL
3	amcham.de | Ame...	2004	960	NULL
4	amcham.de | Ame...	2004	195	NULL
5	amcham.de | Ame...	2004	1245	NULL
6	amcham.de | Ame...	2004	75	NULL

PIVOT-Tabellen

```
SELECT Kunde, SUM([Ricken, Uwe]) AS UR, SUM([Baumann, Guido]) AS GB
FROM dbo.view_WorkTime
PIVOT (
          SUM(Arbeitszeit)
          FOR Mitarbeiter IN ([Ricken, Uwe], [Baumann, Guido])
      ) AS PVT
WHERE ISNULL([Ricken, Uwe], 0) > 0 OR
      ISNULL([Baumann, Guido], 0) > 0
GROUP BY Kunde
```

Results | Messages

	Kunde	Jahr	Ricken, Uwe	Baumann, Guido
1	amcham.be | Ame...	2005	1560	NULL
2	amcham.ch | Swis...	2005	20	NULL
3	amcham.de | Ame...	2004	960	NULL
4	amcham.de | Ame...	2004	195	NULL
5	amcham.de | Ame...	2004	1245	NULL
6	amcham.de | Ame...	2004	75	NULL

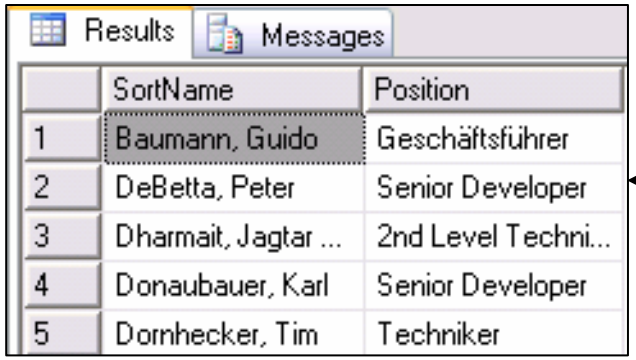
TOP-Klausel

- **Einschränkung der Verwendung von Konstanten ist aufgehoben**
- **TOP kann nun mit variablen Argumenten arbeiten**
  - Variablen (Datentyp int)
  - SELECT – Statement
  - Functions

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

TOP-Klausel

```
-- Standard in Microsoft SQL-Server <=2000
-- absolute Anzahl

SELECT TOP 5 (PERCENT) SortName, Position
FROM dbo.tbl_cust_Employees
ORDER BY SortName ASC

-- oder . . .
SET ROWCOUNT 5

SELECT SortName, Position
FROM dbo.tbl_cust_Employees
ORDER BY SortName ASC

SET ROWCOUNT 0
```

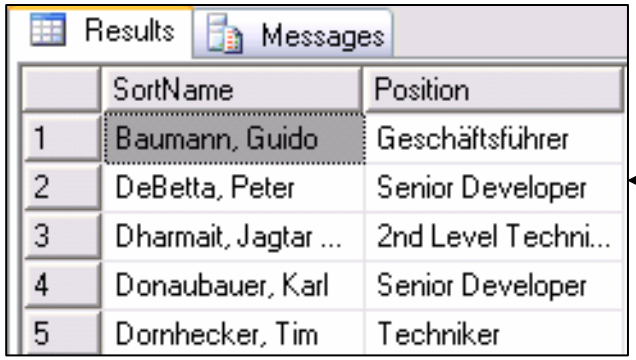


Results | Messages

	SortName	Position
1	Baumann, Guido	Geschäftsführer
2	DeBetta, Peter	Senior Developer
3	Dharmait, Jagtar ...	2nd Level Techni...
4	Donaubauer, Karl	Senior Developer
5	Dornhecker, Tim	Techniker

TOP-Klausel

```
-- Standard in Microsoft SQL-Server <=2000
-- absolute Anzahl

DECLARE   @Rows int
SET       @Rows = 5

SET ROWCOUNT @Rows

SELECT SortName, Position
FROM dbo.tbl_cust_Employees
ORDER BY SortName ASC

SET ROWCOUNT 0
```



Results | Messages

	SortName	Position
1	Baumann, Guido	Geschäftsführer
2	DeBetta, Peter	Senior Developer
3	Dharmait, Jagtar ...	2nd Level Techni...
4	Donaubauer, Karl	Senior Developer
5	Dornhecker, Tim	Techniker

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

TOP-Klausel

```
-- Diese Variante funktioniert bis Microsoft SQL Server 2000 NICHT!

DECLARE   @Rows int
SET       @Rows = 5

SELECT TOP @Rows SortName, Position
FROM      dbo.tbl_cust_Employees
ORDER BY  SortName
```

```
Msg 102, Level 15, State 1, Line 4
Incorrect syntax near '@Rows'.
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

TOP-Klausel

```
-- Eine Alternative war die Ausführung von dynamischem SQL

DECLARE   @Rows     int,
          @SQL      varchar(1000)

SET       @Rows = 5
SET       @SQL =    'SELECT TOP ' + CONVERT(varchar, @Rows) + ' ' +
                    'SortName, Position FROM dbo.tbl_cust_Employees ' +
                    'ORDER BY SortName'

EXEC (@SQL)
```

Results | Messages

	SortName	Position
1	Baumann, Guido	Geschäftsführer
2	DeBetta, Peter	Senior Developer
3	Dharmait, Jagtar ...	2nd Level Techni...
4	Donaubauer, Karl	Senior Developer
5	Dornhecker, Tim	Techniker

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

TOP-Klausel

```
-- TOP wird nun als „Funktion“ verwendet!!!

DECLARE   @Rows int
SET       @Rows = 5

SELECT TOP (@Rows) SortName, Position
FROM      dbo.tbl_cust_Employees
ORDER BY  SortName


SELECT TOP (dbo.fn_myFunction()) SortName, Position
FROM      dbo.tbl_cust_Employees
ORDER BY  SortName
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

DML mit Ausgabemöglichkeiten

```
/*
    Beispiele für DML-Befehle und entsprechende Ausgaben

    Vorgehen in Microsoft SQL-Server <= 2000
    Das nachfolgende Beispiel löscht einen Datensatz
    aus der Tabelle tbl_cust_Employees und gibt NACH
    dem Löschvorgang den Inhalt des Datensatzes aus
*/

DECLARE   @TempTable TABLE (
                            SId                 int,
                            SortName            varchar(255),
                            Position            varchar(255),
                            ManagerId           int,
                            InsertUser          sysname,
                            InsertDate          smalldatetime
                            )

-- Eintragen eines Datensatzes aus tbl_cust_Employees
-- in die temporäre Tabelle
INSERT INTO @TempTable
SELECT * FROM dbo.tbl_cust_Employees
WHERE SId = 2
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

DML mit Ausgabemöglichkeiten

```
/*
    Beispiele für DML-Befehle und entsprechende Ausgaben

    Vorgehen in Microsoft SQL-Server <= 2000
    Das nachfolgende Beispiel löscht einen Datensatz
    aus der Tabelle tbl_cust_Employees und gibt NACH
    dem Löschvorgang den Inhalt des Datensatzes aus
*/

-- Anzeigen des Inhaltes aus @TempTable
SELECT * FROM @TempTable

-- Löschen des Inhaltes aus der Tabelle tbl_cust_Employees
DELETE    dbo.tbl_cust_Employees WHERE SId = 2

-- Bestätigung (Es dürfte nichts angezeigt werden
SELECT * FROM dbo.tbl_cust_Employees WHERE SId = 2

-- Nun wieder den Eintrag aus @TempTable hinzufügen
INSERT INTO dbo.tbl_cust_Employees
SELECT * FROM @TempTable
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

DML mit Ausgabemöglichkeiten

```
/*
    Beispiele für DML-Befehle und entsprechende Ausgaben

    Vorgehen in Microsoft SQL-Server 2005
    Das nachfolgende Beispiel löscht einen Datensatz
    aus der Tabelle tbl_cust_Employees und gibt NACH
    dem Löschvorgang den Inhalt des Datensatzes aus
*/

-- Erstellen der temporären Tabelle
DECLARE   @TempTable TABLE (
                            SId             int,
                            SortName        varchar(255),
                            Position        varchar(255),
                            ManagerId       int,
                            InsertUser      sysname,
                            InsertDate      smalldatetime
                            )

-- Löschen des Datensatzes UND Eintragung in temporärer Tabelle
DELETE    dbo.tbl_cust_Employees
OUTPUT    deleted.* INTO @TempTable
WHERE     SId = 2
```


Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

DML mit Ausgabemöglichkeiten

D E M O

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Exception Handling

- **Fehlerbehandlung von nichtkritische Ausnahmen möglich**
- **Fehlerbehandlung kann Fehlernummern auswerten und darauf reagieren**
- **Rumpf für Fehlerbehandlung**
  - Begin Try
    - Anweisung A
    - Anweisung B
  - End Try
  - Begin Catch
    - Anweisung A
    - Anweisung B
  - End Catch

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Exception Handling

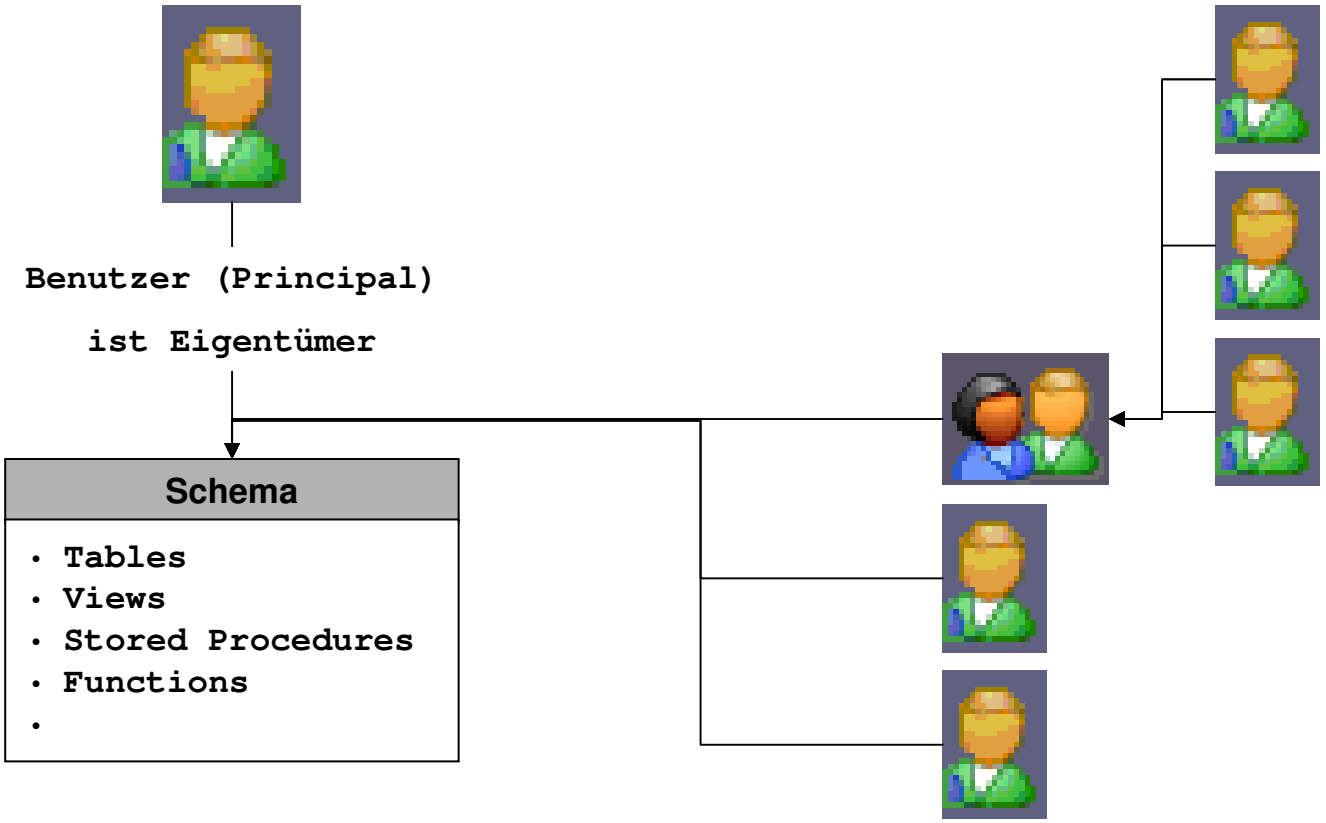
```
CREATE PROC dbo.proc_TransferToSavings
          @KundenId           int,
          @Amount             smallmoney
AS
          -- Automatisches Rollback im Falle eines Fehlers
          SET XACT_ABORT ON

          BEGIN TRY
             BEGIN TRANSACTION Transfer
             -- Zunächst wird der Betrag vom Bankkonto abgebucht
             UPDATE dbo.tblBankKonto
             SET    Kontostand = Kontostand - @Amount
             WHERE  KundenId = @KundenId
             -- um ihn anschließend dem Sparkonto zuzuordnen
             UPDATE dbo.tblSparbuch
             SET    Kontostand = Kontostand + @Amount
             WHERE  KundenId = @KundenId
             COMMIT TRANSACTION Transfer
          END TRY

          BEGIN CATCH
             ROLLBACK TRANSACTION Transfer
             RAISERROR ('Nicht genügend Geld auf dem Bankkonto', 11, 1)
          END CATCH
```

Sicherheit

- **Neues Sicherheitsmodell durch die Verwendung von Schema**
  - Abstraktion von von Benutzers und Objekten in der Datenbank
  - Neuer Ansatz für die „Referenzierung“ von Objekten
- **Möglichkeit, Stored Procedures im Kontext eines anderen Benutzers auszuführen:**
  - EXECUTE AS Caller
  - EXECUTE AS „Benutzer“
  - EXECUTE AS SELF
  - EXECUTE AS OWNER

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Neues Sicherheitsmodell durch die Verwendung von Schema



Berechtigungen werden nur noch auf Schema nicht mehr auf DB-Objekte vergeben!

© GNS GmbH; Robert-Bosch-Straße 8; 63477 Maintal

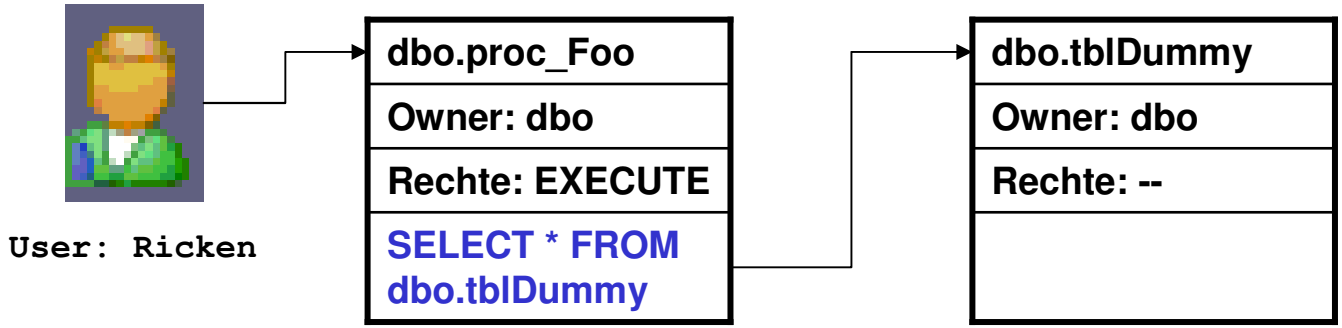
Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Neues Sicherheitsmodell durch die Verwendung von Schema

SQL 2000	SQL 2005
SELECT * FROM master.dbo.sysObjects - Datenbank „Master“ - Eigentümer „dbo“ - Objekt „sysObjects“	SELECT * FROM master.sys.Objects - Datenbank „Master“ - Schema „sys“ - Objekt „Objects“

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

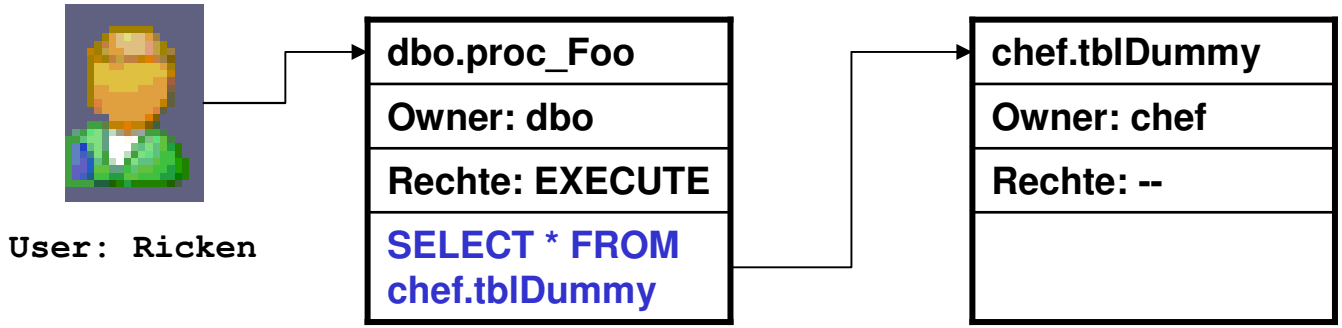
Sicherheitsmodell in SQL 2000



Benutzer „Ricken“ erhält Daten aus Tabelle „tblDummy“ durch die Besitzverkettung der einzelnen Objekte

Die Berechtigung auf dbo.proc_Foo „gewährt“ Rechte auf Objekte, die in dbo.proc_foo adressiert sind

Sicherheitsmodell in SQL 2000



Benutzer „Ricken“ erhält keine Daten aus Tabelle „tblDummy“, da das Objekt tblDummy dem Benutzer „chef“ zugeordnet ist und Benutzer „Ricken KEINE Rechte auf diesem Objekt besitzt!

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de

Neues Sicherheitsmodell durch die Verwendung von Schema

D E M O

© GNS GmbH; Robert-Bosch-Straße 8; 63477 Maintal

Tel. +49 (0) 61 81-4 24 31-0; Fax + 49 (0) 61 81-4 24 31-29 ; www.gns-online.de