Zusammenfassung M141

Datenbanksysteme in Betrieb nehmen

2009-06-24

Emanuel Duss

Über

Autor	Emanuel Duss
Erstellt	2009-03-18
Bearbeitet	2009-06-24
Heute	2009-06-24
Bearbeitungszeit	09:21:29
Lehrjahr des Moduls	3. Lehrjahr 2008 / 2009
Pfad	/ media/APACER_2GB/Schule/3_Lehrjahr/141_Datenbanksysteme_in_Betrieb_nehmen/M141_Zusammenfassung.odt
CC-Lizenz	 Creative Commons Namensnennung-Keine kommerzielle Nutzung-Weitergabe unter gleichen Bedingungen 2.5 Schweiz http://creativecommons.org/licenses/by-nc-sa/2.5/ch/
Powered by	OpenOffice.org ubuntu linux for human beings

Bearbeitungsprotokoll

Datum	Änderung(en)
2009-03-18	Erstellt
2009-03-31	Ergänzungen zum Backup; Neu: Indexe, Prozeduren
2009-06-24	Fertigstellung der Zusammenfassung (Modulschluss)

Inhaltsverzeichnis

1 Das Microsoft Datenbanksystem MS SQL....9
1.1 Diverse Informationen....9
1.2 Syntaxerklärungen....10

2 Repetition Datenbanken....11
2.1 Datenbank....11
2.1.1 Datenbank erstellen....11
2.1.2 Datenbank ändern: Wiederherstellungsmodell FULL....11
2.1.3 Datenbank löschen....11
2.2 Tabellen....12
2.2.1 Tabelle erstellen....12
2.2.2 Tabelle ändern: Primary Key hinzufügen....12
2.2.3 Tabelle abfragen....12

3 Backup und Restore von Datenbanken....13
3.1 Backup-Arten....13
3.2 Backup erstellen....13
3.3 Backup überprüfen....14
3.4 Backups anzeigen....14
3.5 Restore....15
3.5.1 Vollständiges Backup wiederherstellen....16
3.5.2 Differentielles Backup wiederherstellen....16
3.5.3 Transaktionsprotokoll-Backup wiederherstellen....16
3.5.4 Auf die Sekunde genau wiederherstellen....16
3.5.5 Syntaxerklärung....16

4 Index....17
4.1 Wann ist ein Index nötig und was macht ein Index?....17
4.1.1 Tabellengrösse....17
4.1.2 Beispiel....17
4.1.3 Mit dem Ausführungsplan verifizieren....17
4.2 Wie ist ein Index aufgebaut?....18
4.2.1 Einen Datensatz mit Hilfe eines Indexes finden....18
4.2.2 B-Baum / B-Tree / Balanced-Tree....18
4.3 Index erstellen....21
4.3.1 Nicht gruppierten Index....21
4.3.2 Gruppierten Index anlegen....21

5 Transact-SQL 22
5.1 Stored Procedure....22
5.1.1 Vorteile von Stored Procedures: 22
5.2 Stored Functions....22
5.3 Grundgerüst....22
5.4 Kommentare....23
5.5 Variablen....23
5.6 Anweisungen....23
5.7 Prozedur mit Datum....24
5.8 Beispiele....24
5.8.1 Zwei Strings zusammenhängen....24
5.8.2 Tag des Jahres....25
5.8.3 10 Tage zu einem Datum dazuzählen....25
5.8.4 Zahlen manipulieren....25
5.8.5 Zahlen manipulieren mit Case-Anweisung....26

5.8.6 Kreisfläche....26
5.8.7 Zahlen multiplizieren....27
5.8.8 Aufruf in Funktionen....27

6 Pivotieren....28
6.1 Einfach....28
6.2 Erweitert....28
6.2.1 Beispielaufgabe....29
6.2.2 Syntax....30

7 Rekursive Abfragen mit CTE....31
7.1 WTF?....31
7.2 Szenario....31
7.2.1 Mice-en-Place (Vorbereitung)....31
7.2.2 Rekursive Abfrage....32

8 Verschlüsselung von Daten....33
8.1 Allgemeines zur Verschlüsselung....33
8.1.1 Symmetrische und Asymmetrische Verschlüsselung....33
8.1.2 Verschlüsselungsstärke....33
8.1.3 Datenbank- und Diensthauptschlüssel....33
8.2 Daten ver- und entschlüsseln....34
8.2.1 Schlüssel erstellen....34
8.2.2 Daten verschlüsseln....34
8.2.3 Daten entschlüsseln....35
8.3 Schlüssel durch Zertifikat schützen....35
8.4 Mehrere Schlüssel verwenden....35
8.5 Datenbankhauptschlüssel....36

9 Eigene Datentypen mit UDT....37
9.1 Vorgehen....37

10 Sicherheit beim MS SQL-Server 2008....39
10.1 Usernameldungen....39
10.2 Der SQL Server-Konfigurations-Manager....40
10.3 Rechtvergabe des SQL-Server-Dienstes....41

11 Trigger....42
11.1 Vorbereitung....42
11.2 Trigger erstellen: Änderungen....42
11.3 Trigger erstellen: Löschungen....42

12 Diverses....44
12.1 Informationen zu den Tests....44
12.1.1 Test Nummer 1 vom 2009-04-29....44
12.1.2 Test Nummer 2 vom 2009-05-13....44
12.1.3 Test Nummer 3 vom 2009-06-10....44
12.2 TBD....45
12.3 Neuen Benutzer erstellen....45
12.4 Access....47
12.5 Daten importieren....48
12.6 SQL über die Kommandozeile....48

13 Codeschnipsel....49
13.1 .NET-Framework auf dem SQL-Server einschalten....49
13.2 Datensätze generieren....49
13.3 Passwortgenerator....49

14 Glossar.....52

15 Gute Links.....53

Abbildungsverzeichnis

Abbildung 1: Benannte Instanz....9
Abbildung 2: Unbenannte Instanz....9
Abbildung 3: Die Datenbank wird mit den zwei Dateien erstellt....11
Abbildung 4: Primary Key automatisch iterieren....12
Abbildung 5: Neues Backup erstellen....13
Abbildung 6: Pfad zur Backup-Datei....13
Abbildung 7: Das Backup wird hinten angefügt....13
Abbildung 8: Wir wählen die richtigen Backups für den Restore aus....15
Abbildung 9: Ausführungsplan erstellen....17
Abbildung 10: Table Scan ist eher langsam....17
Abbildung 11: Clustered Index Scan ist schneller....18
Abbildung 12: Jeder B-Tree mit Ordnungszahl = 1 und 3 bis 10 Elemente....20
Abbildung 13: Jeder B-Tree mit Ordnungszahl = 2 und 4 bis 10 Elemente....21
Abbildung 14: Die pivotierte Tabelle....28
Abbildung 15: Erstellte Tabelle....32
Abbildung 16: Ergebnis....32
Abbildung 17: Servereigenschaften....39
Abbildung 18: Eingerichtete User....39
Abbildung 19: Startmenüeintrag....40
Abbildung 20: Netzwerkeinstellungen....40
Abbildung 21: Neuen Benutzer erstellen....45
Abbildung 22: Rechte vergeben....46
Abbildung 23: Neues Access-Projekt....47
Abbildung 24: Mit MS SQL-Server verbinden....47

Modulbaukasten

Modulidentifikation

Modulnummer	141
Titel	Datenbanksysteme in Betrieb nehmen
Kompetenz	Datenbanksystem(e) installieren, konfigurieren, Urladung durchführen, Funktionalität sicherstellen und Übergabe in den produktiven Betrieb durchführen.
Handlungsziele	1. Datenbanksystem nach Vorgaben (Anzahl Benutzer, Speicherbedarf, Transaktionsvolumen, Verfügbarkeitsanforderungen usw.) installieren (Server und Clients) und für den produktiven Betrieb einrichten bzw. vorbereiten. 2. Datenbank (leere Datenbank) einrichten. Tabellen nach Vorgabe einpflegen, Testdaten laden und geforderte Funktionalität überprüfen. Urladung/Datenmigration vorbereiten, dokumentieren und durchführen. 3. Datenbanksystem für den operativen Betrieb vorbereiten, Security-, Backup-, Restart- und Recovery-Prozeduren bereitstellen und austesten. Systemtests (Volume-, Stress- und Crash-Tests) durchführen. 4. Mittels Standardreports Performance und Verfügbarkeit überprüfen und evtl. erforderliche Tuning-Massnahmen durchführen. 5. Grundberechtigungen für den produktiven Einsatz einrichten und Standardreports in Form von Views, Store Procedure und Scripts endbenutzergerecht zur Verfügung stellen. 6. Übergabe in die operative Umgebung planen, einleiten und durchführen und Abnahmeprotokoll erstellen.
Kompetenzfeld	System Management
Objekt	Datenbankserver in einer Client/Server Umgebung
Niveau	3
Voraussetzungen	• Datenmodelle (ERM) • Client/Server Konzepte • SQL Anwendung • Server-Betriebssysteme
Anzahl Lektionen	40
Anerkennung	Eidg. Fähigkeitszeugnis Informatiker/Informatikerin
Modulversion	1.1
MBK Release	R3
Harmonisiert am	30.04.07, HANOKs in Überarbeitung

Handlungsnotwendige Kenntnisse

Handlungsnotwendige Kenntnisse beschreiben Wissenselemente, die das Erreichen einzelner Handlungsziele eines Moduls unterstützen. Die Beschreibung dient zur Orientierung und hat empfehlenden Charakter. Die Konkretisierung der Lernziele und des Lernwegs für den Kompetenzerwerb sind Sache der Bildungsanbieter.

Modulnummer	141
Titel	Datenbanksysteme in Betrieb nehmen
Kompetenzfeld	System Management
Modulversion	1.1
MBK Release	R3

Handlungsziel	Handlungsnotwendige Kenntnisse
1.	1. Kennt Arten von Datenbanken (relationale-, objektorientierte-, hierarchische, Netzwerk-DB) und kann für diese typische Merkmale der physischen Datenorganisation nennen. 2. Kennt wichtige Parameter zur Konfigurierung eines Datenbanksystems und kann deren Bedeutung für die Funktionalität und Performance erläutern. 3. Kennt die Schritte zur Analyse einer Vorgabe und weiss, welchen Beitrag diese zur Definition eines klaren Auftrags liefern.
2.	1. Kennt die grundlegenden Aktionen und ihre Abfolgen, welche zum Aufsetzen einer Datenbank in einem Datenbanksystem notwendig sind. 2. Kennt den Normalisierungsprozess zur Überführung von logischen Schemas in Datenbankschemas und kann erläutern, warum diese Schritte erforderlich sind und welchen Beitrag diese für ein lauffähiges System leisten. 3. Kennt die Funktionen und Komponenten des Data Dictionary in einem Datenbanksystem und kann aufzeigen, welchen Beitrag das Data Dictionary für Aufbau und Betrieb einer Datenbank leistet.
3.	1. Kennt Sinn und Zweck von Security-, Backup-, Restart- und Recovery-Prozeduren in einem Datenbanksystem und kann deren Beitrag für die Sicherstellung des operativen Betriebs erläutern. 2. Kennt die grundlegenden Schritte, die bei einem Test durchlaufen werden müssen und kann aufzeigen, welchen Beitrag diese zu einem funktionsfähigen System leisten. 3. Kennt spezifische Testarten für Datenbanksysteme und kann an Beispielen aufzeigen, welchen Beitrag diese für die Performance und Verfügbarkeit eines Datenbanksystems leisten.
4.	1. Kennt Arten von Tuning- und Performance-Massnahmen und kann deren Wirkung und Stellenwert zur Optimierung eines Datenbanksystems erläutern.
5.	1. Kennt grundlegende Vorarbeiten und Schritte für die Betriebsfreigabe einer Datenbank und kann erläutern, welchen Beitrag diese Schritte für einen effiziente Inbetriebnahme leisten. 2. Kennt Möglichkeiten zur Vergabe von Zugriffsrechten und kann aufzeigen, wie damit der Datenschutz und die Konsistenz der Daten im Datenbanksystem gewährleistet werden kann.
6.	1. Kennt Strukturen und Bedeutung der für den Betrieb eines Datenbanksystems erforderlichen Dokumentationen (Installations-, Lizenz-, Betriebs- und Wartungsdokumentation) und kann deren Wichtigkeit für die Lizenzierung, Pflege und Weiterentwicklung belegen. 2. Kennt Aspekte, die bei der Übergabe eines Systems in den operativen Betrieb beachtet werden müssen und kann den Beitrag der erforderlichen Schritte zur Qualität der Abnahme darlegen.

1 Das Microsoft Datenbanksystem MS SQL

1.1 Diverse Informationen

- Der MS-SQL-Server-Dienst muss gestartet sein. Um diesen zu Starten führen wir folgenden Befehl aus: net start „SQL-Server (MSSQLSERVER)“. Dann wird der SQL-Server-Dienst gestartet.
- Der Standard-Port für den MS-SQL-Server ist 1433.
- Der Pfad einer Datenbank muss auf einem Lokalen Datenträger sein. Mit USB-Sticks, Netzlaufwerken oder dergleichen funktioniert es nicht.
- Der MS-SQL-Server ist ein Gauner und ein Halunke. Er klaut ihnen jedes Memory (RAM) und gibt es ihnen nie mehr zurück.
- MS SQL legt nie selber eine Ordnerstruktur an. Wenn man C:\tmp\Test als Pfad angibt und der Ordner Test existiert nicht, wird dieser nicht angelegt! Dasselbe gilt für Unterordner.
- Eine Datenbank besteht aus zwei Dateien. Eine Datei ist die eigentliche Datenbank mit allen Objekten und Daten. Die andere ist das Transaktionsprotokoll mit allen Änderunge. Das Transaktionsprotokoll kann für einen Sekundengenauen Restore verwendet werden (wenn man ein Vollbackup hat).
- 8KB wird pro Seite verbraucht. Wenn nur ein Bit geändert wird, dann wird aber schon 8KB verbraucht.. Deshalb: DB-Festplatte mit 8KB Clustergrösse formatieren um Performance zu gewinnen.
- Wenn wir uns mit dem SQL-Server verbinden, können wir das durch eine benannte Instanz (Bild links) oder durch eine unbenannte Instanz (Bild rechts) machen.

Abbildung 1: Benannte Instanz

Abbildung 2: Unbenannte Instanz

- Beim Anmelden können wir zwischen zwei Authentifizierung auswählen. Wenn man sich mit der Windows-Authentifizierung anmelden kann, sollte man dies nutzen, da das sicherer ist. Die SQL-Server-Authentifizierung ist weniger sicher.

1.2 Syntaxerklärungen

Was	Erklärung
GO	Der SQL-Server wartet, bis alle vorhergehenden Befehle fertig abgearbeitet wurden. Parallele Threads warten aufeinander (Der MS-SQL-Server arbeitet asynchron). Das ist nützlich, wenn diese voneinander abhängig sind.
N'Hallo Eris'	Das N bedeutet, dass der String Unicode kodiert ist. Ein Zeichen benötigt somit 16 Bit. Wenn man das weglasst, dann werden nur 8 Bit benötigt und es läuft theoretisch schneller.
-- Hallo Eris	So wird ein Kommentar eingesetzt.

2 Repetition Datenbanken

2.1 Datenbank

2.1.1 Datenbank erstellen

```
CREATE DATABASE [testA] ON  PRIMARY
( NAME='testA', FILENAME='c:\temp\DATA\testA.mdf' ,
      SIZE = 3072KB , FILEGROWTH=1024KB )
 LOG ON
( NAME='testA_log', FILENAME='c:\temp\LOGDATA\testA_log.ldf' ,
      SIZE=1024KB , FILEGROWTH=10%)
```

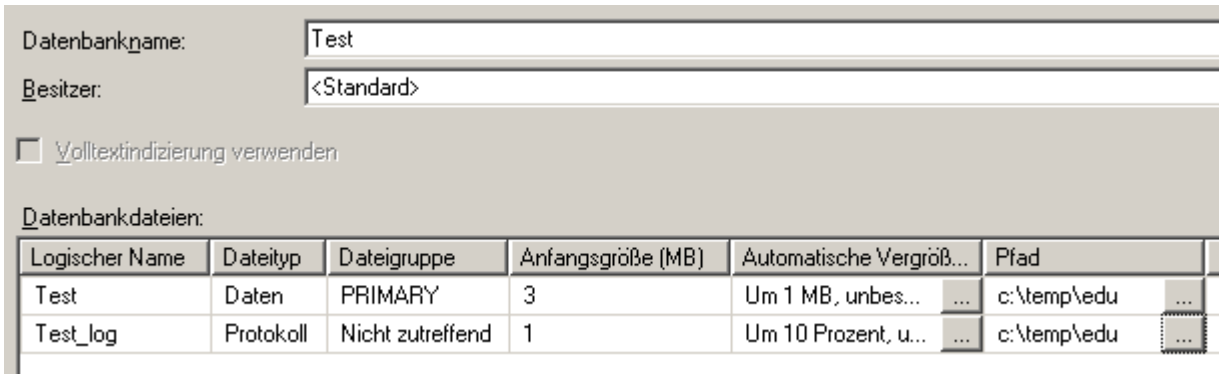


Abbildung 3: Die Datenbank wird mit den zwei Dateien erstellt

2.1.2 Datenbank ändern: Wiederherstellungsmodell FULL

```
ALTER DATABASE [testA] SET RECOVERY FULL
```

2.1.3 Datenbank löschen

```
DROP DATABASE [datenbankname]
```

2.2 Tabellen

2.2.1 Tabelle erstellen

```
CREATE TABLE [dbo].[Table_1](
      [id] [int] IDENTITY(100,10) NOT NULL,
      [Nachname] [varchar](50) NULL,
      [Vorname] [varchar](50) NULL
) ON [PRIMARY]
```

Das Schlüsselwort IDENTITY gibt an, in welchem Abstand der Primary Key erhöht werden soll.

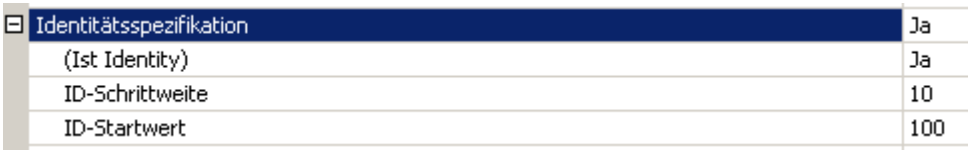


Abbildung 4: Primary Key automatisch iterieren

2.2.2 Tabelle ändern: Primary Key hinzufügen

```
ALTER TABLE dbo.Table_1 ADD CONSTRAINT
      PK_Table_1 PRIMARY KEY CLUSTERED
      (
            id
      )
```

2.2.3 Tabelle abfragen

```
SELECT [id]
      ,[Nachname]
      ,[Vorname]
  FROM [testA].[dbo].[Table_1]
```

3 Backup und Restore von Datenbanken

3.1 Backup-Arten

Vollständig	Diese Backupart muss man immer beim ersten Sichern verwenden! Dies sichert alle Daten komplett.
Differentiell	Sichert das Delta bis zum letzten vollständigen Backup (nicht bis zum vorhergehenden differentiellen Backup!)
Transaktionsprotokoll	Sichert den Änderungsablauf der Datenbank (z.B. Einfügen, Löschen, Ändern, DB-Design). Somit kann man auf die Sekunde genau ein Restore erstellen.

3.2 Backup erstellen

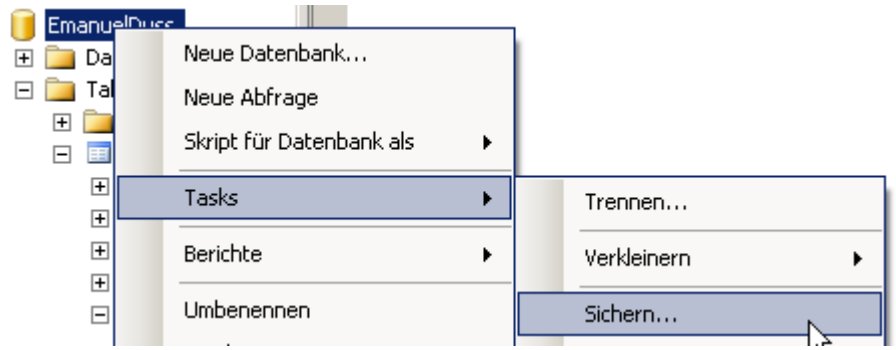


Abbildung 5: Neues Backup erstellen

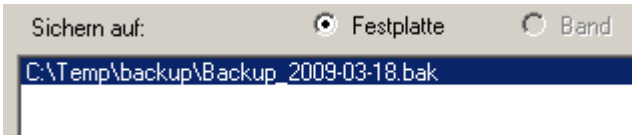


Abbildung 6: Pfad zur Backup-Datei

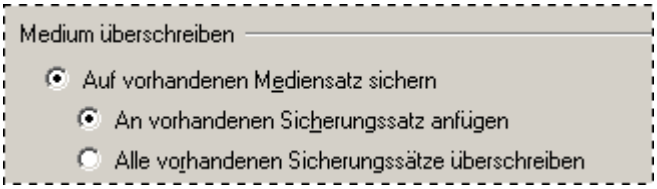


Abbildung 7: Das Backup wird hinten angefügt

Wir müssen uns das Backupfile wie ein Bandlaufwerk vorstellen. Wir fügen an die Backupdatei mehrere Backups an. Jedes Backup wird dann als wie ein eigenes File auf dem Bandlaufwerk behandelt. Das erste File auf dem „Bandlaufwerk“ beginnt mit der Nummer 1.

Vollständig

```
BACKUP DATABASE [datenbankname]
      TO  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
      WITH NOFORMAT, NOINIT,
      NAME = 'Vollständiges Backup'
```

Differentiell

Das differentielle Backup funktioniert genau gleich wie das Vollständige:

```
BACKUP DATABASE [datenbankname]
      TO  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
      WITH  DIFFERENTIAL , NOFORMAT, NOINIT,
      NAME = 'Differentielles Backup'
```

Transaktionsprotokoll

Das Transaktionsprotokoll sichern wir mir BACKUP LOG.

```
BACKUP LOG [datenbankname]
      TO  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
      WITH NOFORMAT, NOINIT,
      NAME = 'Transaktionsprotokoll'
```

3.3 Backup überprüfen

Man kann das Backup überprüfen, ob alle Daten richtig gesichert wurden.

```
declare @backupSetId as int
select @backupSetId = position from msdb..backupset where
database_name='datenbankname' and backup_set_id=(select max(backup_set_id) from
msdb..backupset where database_name='datenbankname' )
if @backupSetId is null begin raiserror(N'Fehler beim Überprüfen.
Sicherungsinformationen für die datenbankname-Datenbank wurden nicht gefunden.',
16, 1) end
RESTORE VERIFYONLY FROM  DISK = N'C:\Temp\backup\Backup_2009-03-18.bak' WITH
FILE = @backupSetId,  NOUNLOAD,  NOREWIND
```

3.4 Backups anzeigen

Hier kann man die Datenbank anzeigen, welche gesichert wurden. Diese Datenbank muss man natürlich auch sichern!

```
select position,*
      from msdb..backupset
      where database_name='datenbankname' and
            backup_set_id=(select max(backup_set_id) from msdb..backupset where
database_name='datenbankname')
```

3.5 Restore

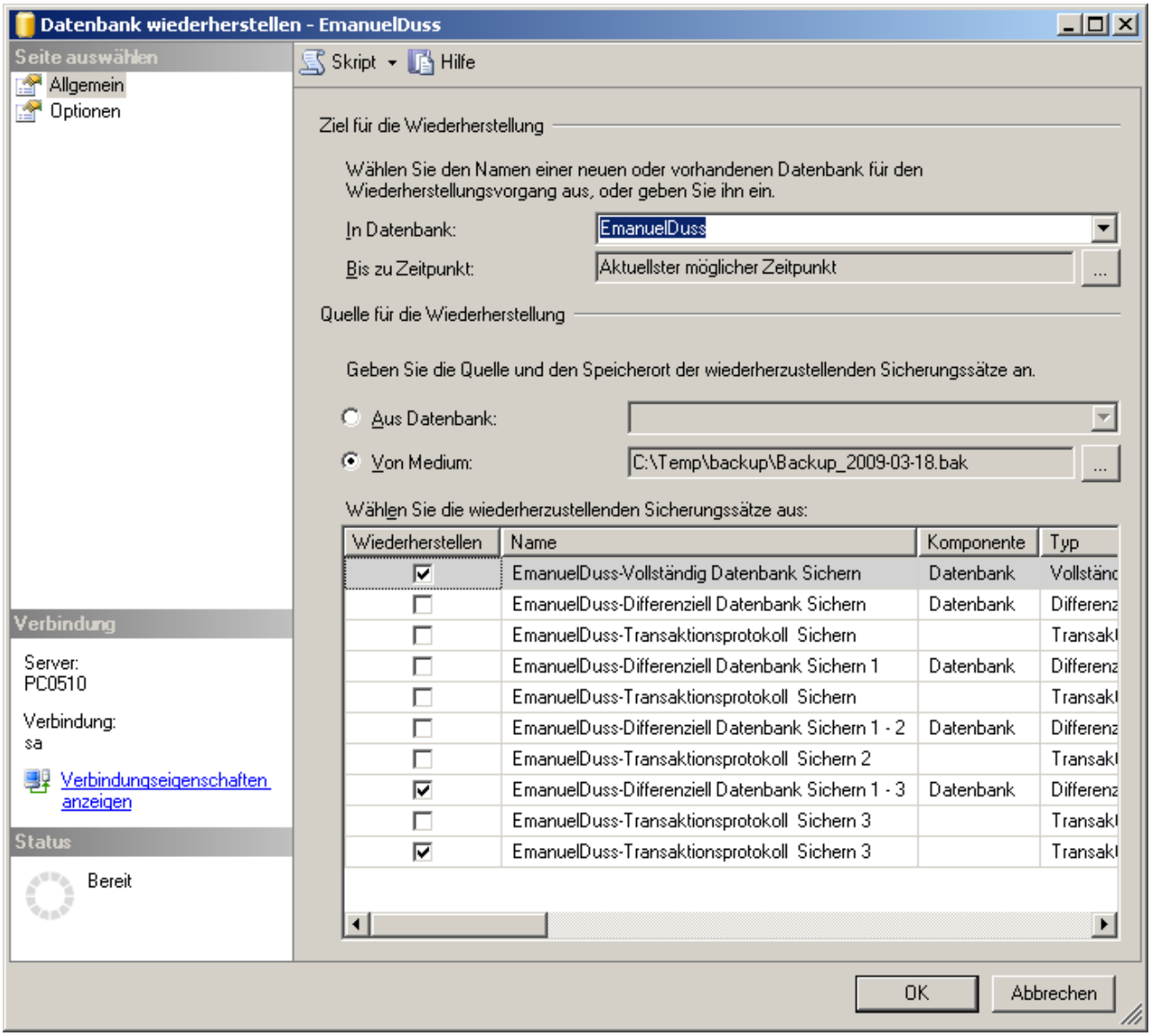


Abbildung 8: Wir wählen die richtigen Backups für den Restore aus

Wir können nun die verschiedenen Dateien innerhalb der Backupdatei wiederherstellen.

3.5.1 Vollständiges Backup wiederherstellen

```
RESTORE DATABASE [datenbankname]
FROM  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
WITH  FILE = 1,
MOVE 'datenbankname_log' TO 'c:\Temp\edu\EmanuelDuss_1.ldf',  NORECOVERY,
NOUNLOAD,  STATS = 10
```

3.5.2 Differentielles Backup wiederherstellen

Das funktioniert genau gleich wie das Wiederherstellen vom vollständigen Backup.

```
RESTORE DATABASE [datenbankname]
FROM  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
WITH  FILE = 8,
MOVE 'datenbankname_log' TO N'c:\temp\edu\EmanuelDuss_1.ldf',  NORECOVERY,
NOUNLOAD,  STATS = 10
```

3.5.3 Transaktionsprotokoll-Backup wiederherstellen

```
RESTORE LOG [datenbankname]
      FROM  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
      WITH  FILE = 10,  NOUNLOAD,  STATS = 10
```

3.5.4 Auf die Sekunde genau wiederherstellen

Backup zu einem bestimmten Zeitpunkt zurückholen:

1. Vollbackup und Differentielles Backup normal restoren.
2. Danach mit dem Transaktionsprotokoll (LOG) zu einem bestimmten Zeitpunkt restoren:

```
RESTORE LOG [datenbankname]
      FROM  DISK = 'C:\Temp\backup\Backup_2009-03-18.bak'
      WITH  FILE = 5,  NOUNLOAD,  STATS = 10,
      STOPAT = N'2009-03-18T11:22:30'
```

3.5.5 Syntaxerklärung

WITH FILE: Welches Backup wir aus unserer Backup-Datei wiederherstellen wollen.

NORECOVERY: Die Daten werden wiederhergestellt, die Datenbank jedoch noch nicht online gestellt. RECOVERY ist Standard.

MOVE: Wir speichern die Datenbank in der angegebenen Datei.

4 Index

4.1 Wann ist ein Index nötig und was macht ein Index?

4.1.1 Tabellengrösse

Man muss mindestens 8KB Daten haben, um Indexe nutzen zu können. 8KB entspricht der Grösse einer Datenbankseite. Beispieldatensätze können ganz einfach generiert werden (vgl. Anhang).

4.1.2 Beispiel

In einer Datenbank sind 23'500 Personen gespeichert. Nun will man die Person mit der ID 23'005 finden. Jetzt muss die Datenbank in der Tabelle von Oben nach unten durchgehen und bei jedem Datensatz überprüfen, ob dies der richtige Datensatz ist. Das dauert eindeutig zu lange.

Um den Suchprozess in einer Datenbank zu beschleunigen, erstellen wir Indexe.

4.1.3 Mit dem Ausführungsplan verifizieren

Mithilfe des Ausführungsplans können wir feststellen, ob ein Index auch wirklich nötig ist:

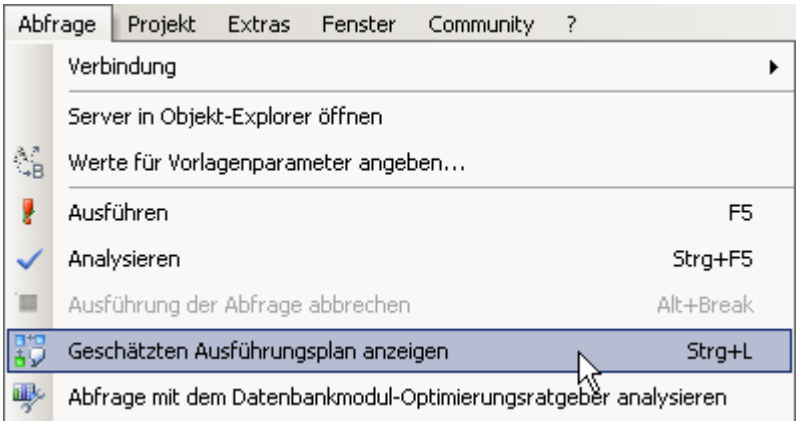


Abbildung 9: Ausführungsplan erstellen

Der Ausführungsplan wird von Rechts nach Links gelesen.

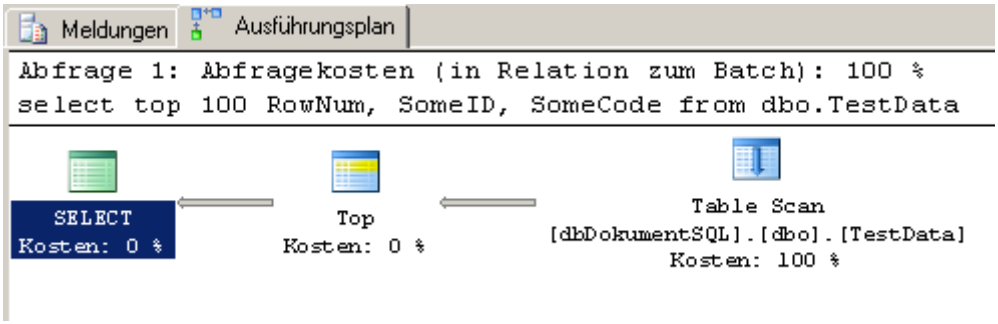


Abbildung 10: Table Scan ist eher langsam

Wir sehen einen Table Scan. Das ist schlecht, da dann das durchsuchen der Tabelle lange dauert.

Jetzt erstellen wir einen Index.

Wir indexieren die Daten, nach denen wir suchen wollen. Der Index erstellt eine neue „Tabelle“. Dort ist abgelegt, auf welcher Datenbankseite das Keyword gespeichert ist. Die Datenbankseite wird schneller gefunden, als wenn man die normale Tabelle durchsucht. Somit geht alles viel schneller.

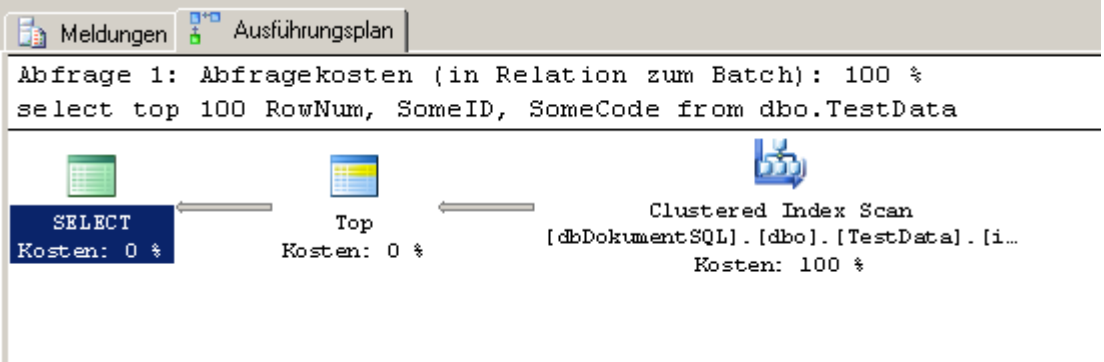


Abbildung 11: Clustered Index Scan ist schneller

4.2 Wie ist ein Index aufgebaut?

4.2.1 Einen Datensatz mit Hilfe eines Indexes finden

Die Datenbank schaut im Index nach, auf welcher Datenbankseite die Person mit der ID 23'005 zu finden ist. Dann geht die Datenbank direkt zu dieser Datenbankseite und durchsucht nur noch diese Datenbankseite.

Das durchsuchen eines Indexes geschieht wesentlich schneller als das Durchsuchen einer gesamten Tabelle.

4.2.2 B-Baum / B-Tree / Balanced-Tree

Hier wird gezeigt, wie MS SQL-Server ein Index erzeugt. Der Index ist ein B-Tree, welcher wie folgt aufgebaut ist:

Regeln

1. Ein Knoten der Ordnung d hat Platz für 2d Einträge, besitzt höchstens 2d+1 Zeiger und es müssen mindestens d Einträge enthalten sein.
2. Mindestens 1 Eintrag pro Block
3. Mindestens 50% gefüllt ausser in der Wurzel
4. Von links nach rechts aufsteigend
5. Unterste Ebene: Leaves; oberste Ebene: Root. Der Weg vom Root zum Leave ist immer gleich lang. (Alle Leaves sind auf der gleichen Ebene) Mit Leaves ist nicht ein einzelner Eintrag gemeint, sondern ein Block!

Folgende Einträge sind in einem Eintrag

1. Welche Person

2. Welche Partition
3. Welcher Cluster
4. Wo geht es weiter

Höhe eines Baumes

Die Höhe h eines Baumes gibt das Maximum der Weglängen aller Knoten an.

$$H_{max} = 1 + \log_{d+1}(\frac{n+1}{2})$$

Da der Taschenrechner nicht alle Logarythmen beherrscht, rechnen wir das nach folgendem Muster:

$$H_{max} = \frac{\ln(\frac{n+1}{2})}{\ln(d+1)} + 1$$ danach muss noch auf die nächst grössere Zahl aufgerundet werden.

n = Anzahl Blätter | d = Ordnung (bei zwei Kästchen ist es 1)

Beispiel:

Anzahl Blätter = n = 5 | Ordnung = d = 1

$$H_{max} = \frac{\ln(\frac{5+1}{2})}{\ln(1+1)} + 1 \quad = \quad H_{max} = \frac{\ln(3)}{\ln(2)} + 1$$ = Taschenrechner: (Ln((5+1)/2)/Ln(1+1))+1 = 2.58

Jetzt runden wir auf die nächst grössere ganze Zahl auf. Das ergibt 3. Und es stimmt.

Maximale Anzahl Knoten / Blöcke

Folgende Formel funktioniert nur bei d > 1.

$$AnzahlKonoten_{max} = \frac{d^{H+1} - 1}{d - 1}$$

Beispiel:

d = Ordnung = 2 | H = Höhe = 2

$$AnzahlKonoten_{max} = \frac{2^{2+1} - 1}{2 - 1}$$ = 7

Es können also maximal 7 Einträge gespeichert werden...

Maximale Anzahl Lesezugriffe

Max Anzahl Lesezugriffe = Höhe + 1

Wie viele Verzweigungen gibt es

Ordnung * 2 + 1 Striche gibt es.

Wie lange dauert ein Lesezugriff

Nach Herr Nyffenegger dauert ein Lesezugriff 20 ms. Das nehme ich einfach mal so hin...

Grösse

- 1 Partition a 500 GB mit 8 KB Cluster
- 4 Byte für eine Person, 1 Byte für den nächsten Cluster
- Jeder Strich braucht 4 Byte

Wichtig

Ein DB-Server zeichnet den B-Baum nicht neu, sondern fügt ihn mit einem speziellen Algorithmus ein. Dies wird hier nicht behandelt.

Prüfungs-Beschiss mit Master-Multi-B-Tree-Templates of non-confusion-Eris:

Um an der Modulprüfung gewinnen zu können, habe ich hier die Bäume mit der Ordnung d = 1 und d=2 und Elemente von 3 bis 10.

Das macht eine Riesenfreude! Natürlich. An dieser Stelle muss ich wohl nicht grüssen... :-D.

Howto use: An der Prüfung alle Zahlen der Reihe nach ordnen und in die oben gezeichneten Bäume einfüllen!

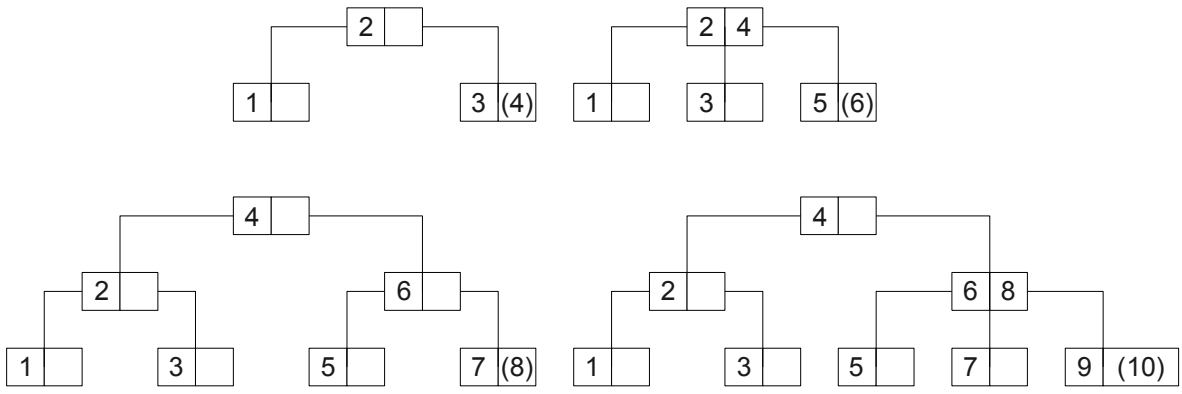


Abbildung 12: Jeder B-Tree mit Ordnungszahl = 1 und 3 bis 10 Elemente

Hier ist dasselbe mit der Ordnungszahl 2:

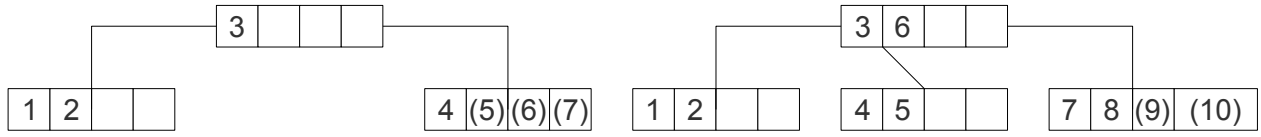


Abbildung 13: Jeder B-Tree mit Ordnungszahl = 2 und 4 bis 10 Elemente

4.3 Index erstellen

- Weitere Attrubute können mt Komma getrennt werden.
- Das generieren des Index kann eine weile Dauern.

4.3.1 Nicht gruppierten Index

Dies ist ein „normaler“ Index.

```
USE [datenbankname]
GO
CREATE UNIQUE INDEX [idxRowNum] ON [dbo].[TestData]
(
      [RowNum] ASC
)
GO
```

4.3.2 Gruppierten Index anlegen

Dieser Index kann verwendet werden, wenn neue Elemente immer am Schluss angefügt werden könne (z.B. Fortlaufendes Datum o. ä.)

```
USE [datenbankname]
GO
CREATE CLUSTERED INDEX [idxGrupiertSomeCode] ON [dbo].[TestData]
(
      [SomeCode] ASC
)
GO
```

- Wenn wir einen gruppierten Index erstellen, werden alles Indexe der gesamten Datenbank neu angelegt, weil die Adressen zu den Datensätzen nicht mehr stimmen.
- Nur Atribute mti Clustered indexen versehen, wenn ein neuer Datensatz immer in jedem Fall zu unterst angefügt wird (z.B. Eine fortlaufende Nummer, Datum oder der Primary Key). Es darf nicht dazwischen gefügt werden, weil dann der Clustered index neu generiert werden muss.

5 Transact-SQL

Weitere Infos: http://www.tsql.de/

5.1 Stored Procedure

Erstellt eine gespeicherte Prozedur. Eine gespeicherte Prozedur ist eine gespeicherte Auflistung von Transact-SQL-Anweisungen oder ein Verweis auf eine CLR-Methode (Common Language Runtime) von Microsoft .NET Framework, die vom Benutzer angegebene Parameter entgegennehmen und zurückgeben kann. Prozeduren können für die permanente oder temporäre Verwendung in einer Sitzung (lokale temporäre Prozeduren) oder für die temporäre Verwendung in allen Sitzungen (globale temporäre Prozeduren) erstellt werden.

Gespeicherte Prozeduren können auch so erstellt werden, dass sie beim Start einer Instanz von SQL Server automatisch ausgeführt werden.

5.1.1 Vorteile von Stored Procedures:

1. Geringere Netzwerklast, da nur das Ergebnis zum Client übertragen wird
2. Schnellere Ausführung als vergleichbare Einzelanweisung
3. Ergebnisse werden vom Datenbankserver im Cache gespeichert. Erneute Abfragen werden dadurch zusätzlich beschleunigt
4. Bei Veränderungen am Datenbankdesign müssen nur die Prozeduren am Server und nicht etwa Abfragen in Programmen geändert werden

5.2 Stored Functions

Erstellt eine benutzerdefinierte Funktion. Es handelt sich dabei um eine gespeicherte Transact-SQL- oder CLR-Routine (Common Language Runtime), die einen Wert zurückgibt. Mit benutzerdefinierten Funktionen können keine Aktionen ausgeführt werden, die den Status einer Datenbank ändern. Benutzerdefinierte Funktionen können, wie Systemfunktionen, aus einer Abfrage heraus aufgerufen werden. Skalarfunktionen können, wie gespeicherte Prozeduren, mit einer EXECUTE-Anweisung ausgeführt werden.

5.3 Grundgerüst

```
CREATE PROCEDURE MeineProzedur
-- Hier kommen die Übergabeparameter hin z.B.
    @Nachname NVARCHAR(50),
    @Vorname NVARCHAR(50)
AS
  -- Hier kommt der neue Code von unten hin. z.B.
SELECT *
    FROM Tabellenname
    WHERE Vorname = @Vorname AND Nachname = @Nachname;
RETURN
```

Ausführen:

```
EXEC MeineProzedur 'Eris', 'Confusion';
```

5.4 Kommentare

```
-- Das ist ein einzeiliger Kommentar
/* Und das ist ein
mehrzeiliger Kommentar */
```

5.5 Variablen

Datentypen

Folgende Datentypen können verwendet werden:

```
INT, FLOAT, DATE, DATETIME, CHAR,
BINARY, IMAGE, TIMESTAMP
```

Variablen Initialisieren

```
DECLARE @VARIABLE1 INT, @VARIABLE2
NVARCHAR(20), @VARIABLE3 NCHAR(3)
```

Variablen einen Wert geben

```
DECLARE @VARIABLE1 INT
SET @VARIABLE1 = 5
```

Variablen aus DB abfragen

```
DECLARE @VARIABLE1 INT
SELECT @VARIABLE1 = COUNT ( * ) FROM
Tabelle1
```

5.6 Anweisungen

If-Verzweigung

```
CREATE PROCEDURE Test3( @A INT, @B INT)
AS
  DECLARE @MAXIMUM INT
  IF @A < @B
    SET @MAXIMUM = @B
  ELSE
    SET @MAXIMUM = @SeA
RETURN @MAXIMUM
```

Select-Case-Anweisung

```
CREATE PROCEDURE TestCase ( @A INT, @B
INT)
AS
  DECLARE @MAXIMUM INT
  SELECT
    @MAXIMUM = CASE
      WHEN @A > @B THEN @A
      WHEN @A < @B THEN @B
  ELSE
    @A
  END
RETURN @MAXIMUM
```

While-Schleife

```
WHILE (SELECT AVG(price) FROM titles) <
$25
BEGIN
   UPDATE titles
      SET price = price * 1.10
   SELECT MAX(price) FROM titles
   IF (SELECT MAX(price) FROM titles) >
$70
      BREAK
   ELSE
      CONTINUE
END
```

Fehlerbehandlung

So kann man Fehler abfangen:

```
IF @@ERROR <>0
      RETURN (101)
ELSE
      RETURN(0)
```

5.7 Prozedur mit Datum

Hier habe ich mit einem Datum etwas herum gespielt. Ich konnte den Code nicht testen und bin deshalb auch nicht überrascht, wenn es noch Fehler hat... Aber so in etwa könnte eine Prüfungsaufgabe aussehen...

```
CREATE PROCEDURE DatumProzedur(
      @Datum1 DATE
      , @Datum2 DATE
      , @Delta INT
      , @Resultat VARCHAR(20)
)
AS
      -- SET @Datum1 = '2009-05-23'
      --, SET @Datum2 = '2009-05-28'
       , SET @Delta = @Datum2 - @Datum1

      IF @Delta < 5
            SET @Resultat = 'Der Datumsunterschied ist nicht sehr gross...'
      IF @Delta = 5
            SET  @Resultat = 'Der Datumsunterschied ist heute irgendwie sehr
eristisch!'
      ELSE
            SET @Resultat = 'Der Datumsunterschied ist mehr als 5 diskordische
Tage!'
SELECT @Resultat
```

Aufruf der Funktion:

```
DatumProzedur 2009-05-23, 2009-05-28
```

5.8 Beispiele

5.8.1 Zwei Strings zusammenhängen

```
Create PROCEDURE sp_add
@a varchar(20)OUTPUT,
@b varchar(20)
AS
BEGIN
      SET NOCOUNT ON;

      SELECT @a+@b;
      SET @a=@b;
END
GO

declare @a varchar(20)
declare @b varchar(20)
set @a='ab'
set @b='cd'
exec sp_add @a OUT,@b
select @a as Antwort
```

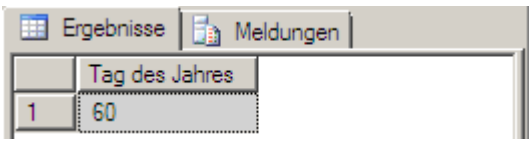
5.8.2 Tag des Jahres

Danke an Lex:

```
CREATE PROCEDURE sp_TagDesJahres
-- Bestimmen der Parameter, welche eingegeben werden müssen
@DateEingabe varchar(20)
AS
BEGIN
            -- Variable deklarieren
            DECLARE @Date DateTime

            -- Deutsche Sprache
            SET Language 'deutsch';

            -- Prüfen, ob Eingabe (Date) ein Datum ist
            IF ISDATE(@DateEingabe)>0
            BEGIN
                        -- Variable zuweisen
                        SET @Date=CAST(@DateEingabe as DateTime)
                        -- Ausgabe
                        SELECT CONVERT(varchar(20),DATENAME(DAYOFYEAR, @Date))
                        AS 'Tag des Jahres'
            END
                        -- Ausgabe, wenn kein Datum eingegeben
                        ELSE PRINT 'Ungültiges Datum'
END
GO
exec sp_TagDesJahres '01.3.2009'
```



5.8.3 10 Tage zu einem Datum dazuzählen

```
select dateadd(dd, 3, getdate())
```

5.8.4 Zahlen manipulieren

```
Alter PROCEDURE sp_manipulate(@A varchar(3), @B varchar(3))
as
begin
      SELECT CONVERT (int, @A);
      SELECT CONVERT (int, @B);

      DECLARE @Zahl1 int
      DECLARE @Zahl2 int
      DeCLARE @Zahl3 int

            set @Zahl1 = (@A*4)
            set @Zahl2 = (@B*2)

            set @Zahl3 = (@A + @B)
      Select @Zahl3

End
```

5.8.5 Zahlen manipulieren mit Case-Anweisung

```
Alter PROCEDURE sp_manipulate(@A varchar(3), @B varchar(3))
as
begin

      SELECT CONVERT (int, @A);
      SELECT CONVERT (int, @B);

      DECLARE @Zahl1 int
      DECLARE @Zahl2 int
      DeCLARE @Zahl3 int

      SELECT Case
            When @A = (int) then @Zahl1 = @A*4,
            When @B = (int) then @Zahl2 = @B*2

            set @Zahl3 = (@A + @B)
      Select @Zahl3

End
```

5.8.6 Kreisfläche

Danke an Lex:

```
CREATE FUNCTION lex_kreisflaeche
      -- Bestimmen der Parameter, welche eingegeben werden müssen
      (
            @GrundseiteE varchar(20),
            @HoeheE varchar(20)
      )

-- Bestimmen welchen Datentyp die Ausgabe haben soll
RETURNS float

BEGIN
-- Überprüfen, ob Eingabe das richtige Format hat
      BEGIN
      IF ISNUMERIC(@GrundseiteE)>0
            BEGIN
            IF ISNUMERIC(@HoeheE)>0
                  BEGIN
                        -- Variablen deklarieren
                        DECLARE @Hoehe float
                        DECLARE @Grundseite float
                        DECLARE @Flaeche float

                        --Berechnung
                        SET @Grundseite=CAST(@GrundseiteE as float)
                        SET @Hoehe=CAST(@HoeheE as float)
                        SET @Flaeche=(@Grundseite * @Hoehe / 2)

                  END
            -- Ausgabe, wenn Eingabe ungültig
            ELSE
            SET @Flaeche=NULL
            END
      -- Ausgabe, wenn Eingabe ungültig
      ELSE
```

```
      SET @Flaeche=NULL
      END

-- Ausgabe
RETURN CONVERT(varchar(20),(@Flaeche))

END

GO

SELECT dbo.lex_kreisflaeche('13.52342', '21.2263') AS 'Kreis Fläche';

SELECT ROUND(dbo.lex_kreisflaeche('13.522', '21.22'), 2) AS 'Kreis Fläche';

SELECT CEILING(dbo.lex_kreisflaeche('13.522', '21.22')) AS 'Kreis Fläche';
```

5.8.7 Zahlen multiplizieren

```
Create function mf_SpezialW (@ZZZ1 bigint, @ZZZ2 bigint)
returns bigint
as
begin
declare @ZZZ3 bigint

set @ZZZ3 = (@ZZZ1*@ZZZ2)*(@ZZZ1*@ZZZ2)*(@ZZZ1*@ZZZ2)

return @ZZZ3

end
```

Aufruf der Funktion:

```
SELECT
      dbo.mF_SpezialW(RowNum, SomeID) as  Zahl
From Test
```

5.8.8 Aufruf in Funktionen

```
SELECT
      MAX( dbo.mF_SinusPotenz(RowNum,SomeID)) as MaxWert,
      MIN( dbo.mF_SinusPotenz(RowNum,SomeID)) as MinWert
from Test
```

6 Pivotieren

6.1 Einfach

```
USE db;
GO
-- Spaltenüberschtiften
SELECT Artikelbezeichnung, [35], [36] , [37], [38], [39], [40], [41], [42],
[43], [44], [45], [46], [47]
FROM
(SELECT Artikelbezeichnung, KW, Menge -- Atribute
FROM dbo.Artikelstamm) p
PIVOT
(
SUM(Menge)
FOR KW IN
( [35], [36] , [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47]
)
) AS pvt
-- Nicht zwingend:
ORDER BY Artikelbezeichnung
```

	Artikelbezeichnung	35	36	37	38	39	40	41	42	43	44	45	46	47
1	D1252VL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	40	125	0	NULL	NULL
2	D1432BS	16	NULL	NULL	NULL	13	NULL	NULL	NULL	62	14	NULL	NULL	NULL
3	H3704ST15	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	K100SM	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL
5	U112PE	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	2	NULL	NULL	NULL
6	U123PE	10	NULL	NULL	NULL	10	NULL	NULL	NULL	1	NULL	NULL	NULL	NULL
7	U1310PE	NULL	NULL	NULL	NULL	12	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	U156ST15	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1	NULL	NULL	NULL	NULL
9	U205ST15	10	NULL	NULL	NULL	10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10	U249ST01	16	NULL	NULL	NULL	19	NULL	NULL	NULL	12	11	NULL	NULL	NULL

Abbildung 14: Die pivotierte Tabelle

6.2 Erweitert

Nummern der Pivot-Elemente selber machen.

```
use dbDokumentSQL
ALTER VIEW V_ModulDokuStatistik
AS
   SELECT [Dokument/e],
      [1] as 'Manual',[2] as 'SQL',[3] as 'Workshop',[4] as 'M239',[5] as
'M301',[0] as 'unbestimmt'
   FROM
      (
         select
            COUNT(*) as Anzahl,

            'Dokument/e' as [Dokument/e],

            case when [Stichwort-1]='Manual'
                  then 1
```

```
                else
                   case when [Stichwort-1]='SQL'
                         then 2
                         else
                            case when [Stichwort-1]='Workshop'
                                  then 3
                                  else
                                     case when [Stichwort-1]='M239'
                                           then 4
                                           else
                                              case when [Stichwort-1]='M301'
                                                    then 5
                                                    else 0
                                              end
                                     end
                            end
                   end
             end as Modul
          from Dokument with (nolock)
          group by Dokumentname,[Stichwort-1]
      )
      AS SourceTable
          PIVOT ( SUM(Anzahl) FOR Modul IN ([0],[1],[2],[3],[4],[5])
                ) AS PivotTable
```

6.2.1 Beispielaufgabe

Folgende Tabelle t_artikel ist gegeben:

Name	Artikel	Menge
Name_A	Artikel_A	10
Name_B	Artikel_B	2
Name_C	Artikel_B	10
Name_B	Artikel_B	20
Name_A	Artikel_A	30
Name_A	Artikel_A	1

Tabelle 1: Gegebene Tabelle

Bitte Pivotieren Sie!

So könnte eine Lösung aussehen. Man mus wissen, dass ich es nie ausprobiert habe, sondern eifach ein bizzli abgekupfert habe vom Lehrer...

```
USE datenbankname
ALTER VIEW v_artikel
AS
   SELECT [Name],
      [1] as 'Artikel_A',
      [2] as 'Artikel_B',
      [0] as 'Sonstige Artikel'
   FROM
      (
         select
            COUNT(*) as Menge,
            'Name' as [Name],
            case when [Artikel]='Artikel_A'
```

```
               then 1
               else
                  case when [Artikel]='Artikel_A'
                     then 1
                     else 0
                  end
            end as Modul
         from tabelenname with (nolock)
         group by Name,[Artikel]
      )
      AS SourceTable
         PIVOT ( SUM(Menge) FOR Artikel IN ([0],[1],[2])
                ) AS PivotTable
```

6.2.2 Syntax

Das ganze könnte man jetzt allgemein ausdrücken:

```
USE datenbankname
ALTER VIEW viewname
AS
   SELECT [nicht_pivotierte_spalte],
      [1] as 'neue_spalte_1_a',
      [2] as 'neue_spalte_2_b',
      [0] as 'neue_spalte_3_c'
   FROM
      (
         select
            COUNT(*) as spalte_gezaehlt_werden_soll,
            'nicht_pivotierte_spalte' as [nicht_pivotierte_spalte],
            case when [pivotierte_spalte]='inhalt_spalte_1_a'
               then 1
               else
                  case when [pivotierte_spalte]='inhalt_spalte_1_b'
                     then 1
                     else 0
                  end
             end as Modul
         from tabelenname with (nolock)
         group by nicht_pivotierte_spalte,[pivotierte_spalte]
      )
      AS SourceTable
         PIVOT ( SUM(spalte_die_gezaehlt_werden_soll) FOR
nicht_pivotierte_spalte IN ([0],[1],[2])
                ) AS PivotTable
```

Wahrscheinlich ist's ein risen huren Mist. Ich tschegge es nicht wirklich. Faen!

7 Rekursive Abfragen mit CTE

CTE = Common Table Expressions

7.1 WTF?

Was Microsoft dazu schreibt (http://msdn.microsoft.com/de-de/library/ms175972.aspx):

Gibt ein temporäres Resultset an, das als allgemeiner Tabellenausdruck (CTE, Common Table Expression) bekannt ist. Dieser wird von einer einfachen Abfrage abgeleitet und innerhalb des Ausführungsbereichs einer einzigen SELECT-, INSERT-, UPDATE- oder DELETE-Anweisung definiert. Diese Klausel kann auch in einer CREATE VIEW-Anweisung als Teil der definierenden SELECT-Anweisung verwendet werden. Ein allgemeiner Tabellenausdruck kann auch Verweise auf sich selbst enthalten. In diesem Fall handelt es sich um einen rekursiven allgemeinen Tabellenausdruck.

7.2 Szenario

7.2.1 Mice-en-Place (Vorbereitung)

Ich erstelle eine neue Tabelle

```
CREATE TABLE tabellenname (
      id INT IDENTITY PRIMARY KEY,
      was VARCHAR(15),
      abstammung INT)
```

Und befülle diese mit Inhalten:

```
INSERT INTO tabellenname VALUES ('Root', null)
INSERT INTO tabellenname VALUES ('Mensch', 1)
INSERT INTO tabellenname VALUES ('Tier', 1)
INSERT INTO tabellenname VALUES ('Pflanzen', 1)
INSERT INTO tabellenname VALUES ('Frau', 2)
INSERT INTO tabellenname VALUES ('Mann', 2)
INSERT INTO tabellenname VALUES ('Gemüse', 4)
INSERT INTO tabellenname VALUES ('Obst', 4)
INSERT INTO tabellenname VALUES ('Frucht', 4)
INSERT INTO tabellenname VALUES ('Hamster', 3)
INSERT INTO tabellenname VALUES ('Kuh', 3)
INSERT INTO tabellenname VALUES ('Esel', 3)
INSERT INTO tabellenname VALUES ('Gurke', 7)
INSERT INTO tabellenname VALUES ('Apfel', 8)
INSERT INTO tabellenname VALUES ('Birne', 8)
INSERT INTO tabellenname VALUES ('Erdbeere', 9)
INSERT INTO tabellenname VALUES ('Kiwi', 9)
INSERT INTO tabellenname VALUES ('Salat', 7)
```

Das ganze sieht danach so aus:

```
SELECT * FROM tabellenname
```

	id	was	abstammung
1	1	Root	NULL
2	2	Mensch	1
3	3	Tier	1
4	4	Pflanzen	1
5	5	Frau	2
6	6	Mann	2
7	7	Gemüse	4
8	8	Obst	4
9	9	Frucht	4
10	10	Hamster	3
11	11	Kuh	3
12	12	Esel	3
13	13	Gurke	7
14	14	Apfel	8
15	15	Birne	8
16	16	Erdbeere	9
17	17	Kiwi	9
18	18	Salat	7

Abbildung 15: Erstellte Tabelle

7.2.2 Rekursive Abfrage

Ich will nun alle Elemente aufgelistet haben, die vom Tier kommen:

```
WITH tabellennameH(id, was, abstammung) as (
      select id, was, abstammung
            from tabellenname
                  WHERE was = 'Tier'
union all
select e.id, e.was, e.abstammung
      from tabellenname e
      join tabellennameH eh on e.abstammung = eh.id
)
select id, was, abstammung from tabellennameH
```

Hinweis: Mit "tabellenname e" vergibt man "tabellenname" den Alias "e". Genau so mit "tabellennameH" und "eh".

	id	was	abstammung
1	3	Tier	1
2	10	Hamster	3
3	11	Kuh	3
4	12	Esel	3

Abbildung 16: Ergebnis

8 Verschlüsselung von Daten

Quelle und weitere Informationen: http://www.itrain.de/knowhow/sql/2005/tsql/encryption/encrypt.asp

8.1 Allgemeines zur Verschlüsselung

8.1.1 Symmetrische und Asymmetrische Verschlüsselung

Mit dem SQL-Server können Daten symmetrisch und asymmetrisch verschlüsseln. Bei der symmetrischen Verschlüsselung wird für die Ent- bzw. Verschlüsselung der selbe Schlüssel verwendet. Da die symmetrische Verschlüsselung schneller ist als die asymmetrische, wird diese Methode empfohlen. Das Problem der symmetrischen Verschlüsselung liegt jedoch beim Schlüsselaustausch. Das kann mit der asymmetrischen Verschlüsselung gelöst werden.

8.1.2 Verschlüsselungsstärke

Grundsätzlich ist die Verschlüsselung nur so stark wie der Schlüssel selbst. Der Zugriff auf die Schlüssel wird im SQL Server auf zwei Arten geschützt: Zum einen kann der Zugriff über Berechtigungen eingeschränkt werden. Dadurch erhalten nur autorisierte Benutzer Zugriff auf die Schlüssel; Datenbankbesitzer oder Systemadministratoren erhalten so aber in jedem Fall Zugriff auf alle Schlüssel der Datenbank. Daher wird der Schlüssel zusätzlich geschützt, indem der Schlüssel selbst wieder verschlüsselt gespeichert wird. Für die verschlüsselte Speicherung eines Schlüssels können andere Schlüssel oder ein Kennwort verwendet werden.

Um Daten zu verschlüsseln oder entschlüsseln muss grundsätzlich zunächst der Schlüssel geöffnet werden. Anschließend kann die Funktion zum Verschlüsseln (z.B. EncryptByKey) oder Entschlüsseln (z.B. DecryptByKey) aufgerufen werden. Um den Vorgang des Öffnens von Schlüsseln zu automatisieren, können die Schlüssel quasi automatisch geöffnet werden.

8.1.3 Datenbank- und Diensthauptschlüssel

Um die Schlüssel in einer Datenbank automatisch öffnen zu können, kann der sogenannte Datenbankhauptschlüssel verwendet werden. Wird ein Schlüssel mit dem Datenbankhauptschlüssel verschlüsselt abgespeichert, so kann er automatisch geöffnet werden, sobald der Datenbankhauptschlüssel geöffnet ist. Dadurch reicht es aus, einen Schlüssel zu öffnen, um Zugriff auf alle mit diesem Schlüssel verschlüsselten Schlüssel in der Datenbank zu erhalten (Voraussetzung dazu ist natürlich zusätzlich die Berechtigung auf die Schlüsselobjekte).

Um auch das Öffnen des Datenbankhauptschlüssels zu automatisieren, kann der Datenbankhauptschlüssel wiederum mit dem sogenannten Diensthauptschlüssel verschlüsselt abgespeichert werden. In diesem Fall muss auch der Datenbankhauptschlüssel nicht mehr explizit geöffnet werden. Der Diensthauptschlüssel wird automatisch mit Hilfe der DPAPI während der Serverinstallation erzeugt.

Diese Automatisierung bedeutet natürlich auch, dass die Daten jetzt eigentlich nur noch durch Berechtigungen geschützt sind, da keine Kennwörter benötigt werden, um einen Schlüssel in der Datenbank zu öffnen. Trotzdem hat natürlich auch dieses Vorgehen seine Berechtigung, denn die Daten können verschlüsselt abgespeichert werden, ohne dass die Anwendung explizit Anweisungen zum Öffnen und Schließen von Schlüsseln benötigt. Werden die Ver- und Entschlüsselungsmethoden in Sichten oder gespeicherten Prozeduren verwendet, so ist die Datenverschlüsselung für die Anwendung transparent. Dieses Vorgehen bietet auch hinreichenden Schutz für die Datenbanksicherungen, da eine Wiederherstellung der Daten auf einem anderen Server keinen Zugriff auf die Daten ermöglicht, denn in der Sicherung selbst ist der Diensthauptschlüssel natürlich nicht enthalten.

8.2 Daten ver- und entschlüsseln

Zuerst erstellen wir eine Tabelle. Die Datentypen der zu verschlüsselnden Inhalten ist varbinary.

```
CREATE TABLE tabellenname (Name varbinary(52), KontoNr varbinary(52));
```

8.2.1 Schlüssel erstellen

Jetzt erstellen wir den symmetrischen Schlüssel:

```
CREATE SYMMETRIC KEY schluesselname
    WITH ALGORITHM=TRIPLE_DES
    ENCRYPTION BY PASSWORD = 'just4us';
```

Wir können die Schlüsselinformaiton abrufen

```
SELECT * FROM sys.symmetric_keys
```

	name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date	modify_date
1	schluesselname	6	256	128	D3	TRIPLE_DES	2009-05-20 ...	2009-05-20 ...

8.2.2 Daten verschlüsseln

Wir öffnen den Schlüssel

```
OPEN SYMMETRIC KEY schluesselname
    DECRYPTION BY PASSWORD = 'just4us';
```

Wir können überprüfen, ob der Schlüssel geöffnet ist:

```
SELECT * FROM sys.openkeys
```

	database_id	database_name	key_id	key_name	key_guid	opened_date	status
1	20	EmanuelDB	256	schluesselname	75EBD900-3BB0-40C4-A745-4AC8B49CE55C	2009-05-20 12:39:21.033	1

Nun können wir Daten in die Tabelle einfügen:

```
INSERT INTO tabellenname (Name, KontoNr)
    VALUES (EncryptByKey(KEY_GUID('schluesselname'), 'Emanuel Duss'),
            EncryptByKey(KEY_GUID('schluesselname'), '5-23-042'))
INSERT INTO tabellenname (Name, KontoNr)
    VALUES (EncryptByKey(KEY_GUID('schluesselname'), 'Bill Gates' ),
             EncryptByKey(KEY_GUID('schluesselname'), '5-23-666' ));
```

Dann schliessen den Schlüssel wieder:

```
CLOSE SYMMETRIC KEY schluesselname;
```

Der Schlüssel ist jetzt nicht mehr offen:

```
SELECT * FROM sys.openkeys;
```

	database_id	database_name	key_id	key_name	key_guid	opened_date	status

Wir sehen in der Tabelle, dass die Daten verschlüsselt sind:

```
SELECT * FROM tabellenname;
```

	Name	KontoNr
1	0x00D9EB75B03BC440A7454AC8B49CE55C01000000FE89A09...	0x00D9EB75B03BC440A7454AC8B49CE55C0100000012D728...
2	0x00D9EB75B03BC440A7454AC8B49CE55C010000007C59EF1...	0x00D9EB75B03BC440A7454AC8B49CE55C0100000093A6734...

Die verschlüsselten Daten sind übrigens noch gesalzen, damit man nicht sieht, ob zwei Werte den gleichen Inhalt haben.

8.2.3 Daten entschlüsseln

Doch wie lesen wir jetzt die verschlüsselten Daten wieder aus? Das geht folgendernassen:

Wir öffnen den Schlüssel wieder:

```
OPEN SYMMETRIC KEY schluesselname
    DECRYPTION BY PASSWORD = 'just4us';
```

Die Daten können nun mit DecryptByKey herausgelesen werden:

```
SELECT CAST(DecryptByKey(Name) AS varchar(20)) AS Name,
       CAST(DecryptByKey(KontoNr) AS varchar(20)) AS KontoNr
       FROM tabellenname;
```

Der Schlüssel kann wieder geschlossen werden:

```
CLOSE SYMMETRIC KEY schluesselname;
```

Das entschlüsseln mit DecryptByKey funktioniert jetzt nicht mehr:

	Name	KontoNr
1	NULL	NULL
2	NULL	NULL

8.3 Schlüssel durch Zertifikat schützen

Zuerst erstellen wir eine Tabelle. Die Datentypen der zu verschlüsselnden Inhalten ist varbinary.

```
CREATE TABLE tabellenname (Name varbinary(52), KontoNr varbinary(52));
```

Dann erstellen wir ein neues Zertifikat:

```
CREATE CERTIFICATE zertifikatname
    ENCRYPTION BY PASSWORD = 'not4you'
    WITH SUBJECT = 'Zugriff auf schluesselname',
    START_DATE = '20050505',
    EXPIRY_DATE = '20121226';
```

Wir erstellen einen neuen symmetrischen Schlüssel, der durch das Zertifikat verschlüsselt wird:

```
CREATE SYMMETRIC KEY schluesselname
    WITH ALGORITHM = TRIPLE_DES ENCRYPTION
    BY CERTIFICATE zertifikatname;
```

Dieser Schlüssel wird geöffnet:

```
OPEN SYMMETRIC KEY schluesselname
    DECRYPTION BY CERTIFICATE zertifikatname
    WITH PASSWORD = 'not4you';
```

Jetzt kann genau gleich fortgefahren werden, wie weiter Oben beschrieben.

8.4 Mehrere Schlüssel verwenden

Natürlich kann man beim Verschlüsseln mehrere Schlüssel verwenden.

Beim Entschlüsseln der Daten wird automatisch der richtige Schlüssel verwendet. Dies ist möglich, da in den ersten 16-Bytes der verschlüsselten Daten hinterlegt ist, mit welchem Schlüssel die Daten verschlüsselt wurden.

Wenn nur ein Schlüssel von zwei geöffnet wurde, werden auch nur die Daten entschlüsselt, von denen der Schlüssel geöffnet ist. Da könnte das Ergebnis folgenermassen aussehen:

	BLZ	Kontonr
1	NULL	NULL
2	NULL	NULL
3	70040000	1122334455
4	20020020	66778899

Danach muss man nicht alle Schlüssel einzel schliessen, sondern kann man alle auf einmal schliessen.

```
CLOSE ALL SYMMETRIC KEYS;
```

8.5 Datenbankhauptschlüssel

Bei allen bisherigen Beispielen mussten die Schlüssel explizit durch Angabe eines Kennworts geöffnet werden. Mit Hilfe des Datenbankhauptschlüssels können asymmetrische Schlüssel und Zertifikate ohne Angabe eines Kennworts geöffnet werden. Um diese Möglichkeit nutzen zu können, muss in der Datenbank zunächst ein Datenbankhauptschlüssel angelegt werden. Dieser Schlüssel wird zum einen mit dem angegebenen Kennwort verschlüsselt gespeichert; zusätzlich wird automatisch eine durch den Diensthauptschlüssel der Instanz verschlüsselte Kopie gespeichert.

Nachdem der Datenbankhauptschlüssel erstellt ist, kann der Kennwortschutz vom Zertifikat entfernt werden. Das Zertifikat ist anschließend durch den Datenbankhauptschlüssel geschützt.

Wir erstellen einen Datenbankhauptschlüssel:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'just4us';
```

Wir erstellen ein neues Zertifikat:

```
CREATE CERTIFICATE zertifikatname
    ENCRYPTION BY PASSWORD = 'not4you'
    WITH SUBJECT = 'Zugriff auf schluesselname',
    START_DATE = '20050505',
    EXPIRY_DATE = '20121226';
```

Mit dem Statement WITH PRIVATE KEY schützen wir das Zertifikat durch den Datenbankhauptschlüssel.

```
ALTER CERTIFICATE zertifikatname
       WITH PRIVATE KEY (DECRYPTION BY PASSWORD = 'not4you');
```

Wir können die Zertifikate anzeigen lassen. Wir sehen, dass nun ein Privater Schlüssel verwendet wird:

```
select * from sys.certificates;
```

name	certificate_id	principal_id	pvt_key_encryption_type	pvt_key_encryption_type_desc
zertifikatname	256	6	MK	ENCRYPTED_BY_MASTER_KEY

Wir erstellen einen neuen Schlüssel:

```
CREATE SYMMETRIC KEY schluesselname
    WITH ALGORITHM = TRIPLE_DES ENCRYPTION
    BY CERTIFICATE zertifikatname;
```

Wir öffnen den Symmetrischen Schlüssel. Nun ist kein Kennwort mehr erforderlich!

```
OPEN SYMMETRIC KEY schluesselname
       DECRYPTION BY CERTIFICATE zertifikatname;
```

Und können Daten einfügen und abfragen wie oben beschrieben...

```
[...] -- Siehe weiter oben
```

Der Schlüssel kann wieder geschlossen werden:

```
CLOSE SYMMETRIC KEY schluesselname;
```

9 Eigene Datentypen mit UDT

UDT = User Defined Types

9.1 Vorgehen

Wir aktivieren diese Funktion:

```
EXEC sp_configure 'clr enabled' , '1'
reconfigure;
```

Dann erstellen wir eine neue Datenbank:

```
CREATE DATABASE [dbAHV] ON PRIMARY
( NAME = N'dbAHV', FILENAME = N'c:\temp\dbAHV.mdf' , SIZE = 3072KB , FILEGROWTH
=
1024KB )
LOG ON
( NAME = N'dbAHV_log', FILENAME = N'c:\temp\dbAHV_log.ldf' , SIZE = 1024KB ,
FILEGROWTH = 10%)
```

Jetzt müssen Assembly registrieren. Das geschieht pro Datenbank und nicht pro Server

```
create assembly [myAHV] from 'c:\temp\myAHV.dll'
create type [AhvNummer] external name [myAHV].[AhvNummer]
```

Wir erstellen eine neue Tabelle

```
create table Person (
id int,
Nachname varchar(20),
AHV AHVNummer
)
```

Jetzt können nur noch gültige AHV-Nummern eingegeben werden

```
insert into Person (id,Nachname,AHV) values(1,'Huber','456.56.111.678')
insert into Person (id,Nachname,AHV) values(2,'Meier','123.78.222.124')
insert into Person (id,Nachname,AHV) values(3,'Müller','890.09.333.123')
insert into Person (id,Nachname,AHV) values(4,'Nyffenegger','111.222.333.444')
```

-- Es wird nur ein Kräuterhaufen ausgegeben

```
SELECT * FROM person;
```

	id	Nachname	AHV
1	1	Huber	0x07312E312E302E3038006F00C801A602
2	2	Meier	0x07312E312E302E304E00DE007B007C00
3	3	Müller	0x07312E312E302E3009004D017A037B00
4	4	Nyffenegger	0x07312E312E302E304100BC0186006A03
5	4	Sigi	0x07312E312E302E304100BC01E7036A03
6	5	sowieso	0x07312E312E302E304100BC01E7036A03

So kann man das Kraut beheben: (Da C# CaseSensitive ist, muss man ToString genau so schreiben!). Das ist ein in C# programmierter Funktionsaufruf:

```
SELECT AHV.ToString() FROM person;
```

So wird der Jahrgang ausgelesen:

```
SELECT AHV.Jahrgang FROM person;
```

So sieht die ganze Tabelle aus:

```
SELECT id, Nachname, AHV.ToString() FROM person;
```

	id	Nachname	(Kein Spaltenname)
1	1	Huber	456.56.111.678
2	2	Meier	123.78.222.124
3	3	Müller	890.9.333.123
4	4	Nyffenegger	134.65.444.874
5	4	Sigi	999.65.444.874
6	5	sowieso	999.65.444.874

So kann man auch Filtern:

```
select Nachname, AHV.ToString() from person WHERE AHV.Jahrgang between 50 and
70;
```

Constraint (Einschränkungen) definieren

```
alter table Person Add Jahrgang as dbo.AhvNummer::GetJahrgang(AHV) persisted
alter table Person add constraint CheckJahrgang Check(AHV.Jahrgang>0)
```

Index erstellen

```
create index Idx_PersonJahrgang on Person(Jahrgang)
```

10 Sicherheit beim MS SQL-Server 2008

Folgendermassen kann man die Sicherheit vom MS SQL-Server erhöhen:

10.1 Usernameldungen

Wir gehen auf die Eigenschaften vom Server und setzen folgende Einstellungen:

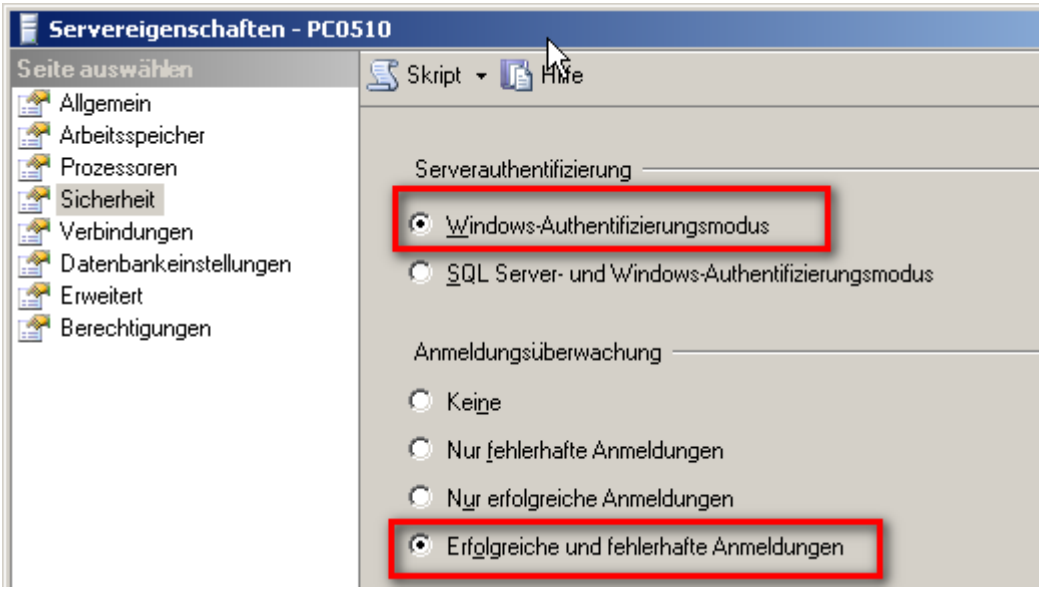


Abbildung 17: Servereigenschaften

- Wenn der SQL-Server mit einem Active-Directory verbunden ist, kann man sich nun mit einem Domänen-Account einloggen.
- Wir wollen jedes Login wechseln, damit wir später überprüfen können, wer sich wann eingeloggt hat.

Auf dem folgendem Screenshot kann man sehen, dass das Usermanagement nicht sehr optimal eingerichtet ist:

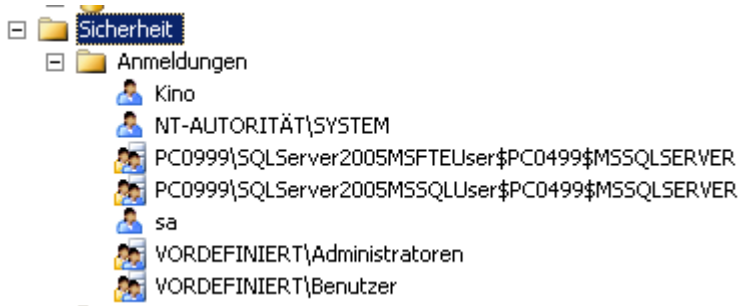


Abbildung 18: Eingerichtete User

- der SA-Account ist aktiviert
- Der vordefinierte Administrator ist aktiviert

10.2 Der SQL Server-Konfigurations-Manager

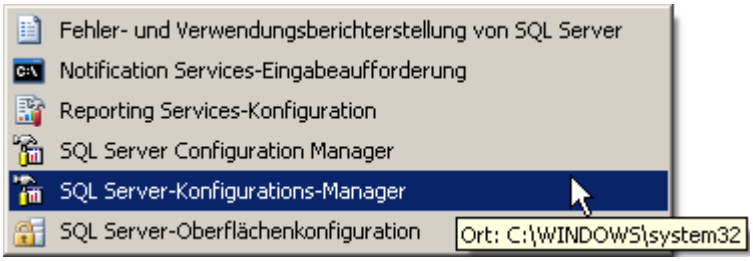


Abbildung 19: Startmenüeintrag

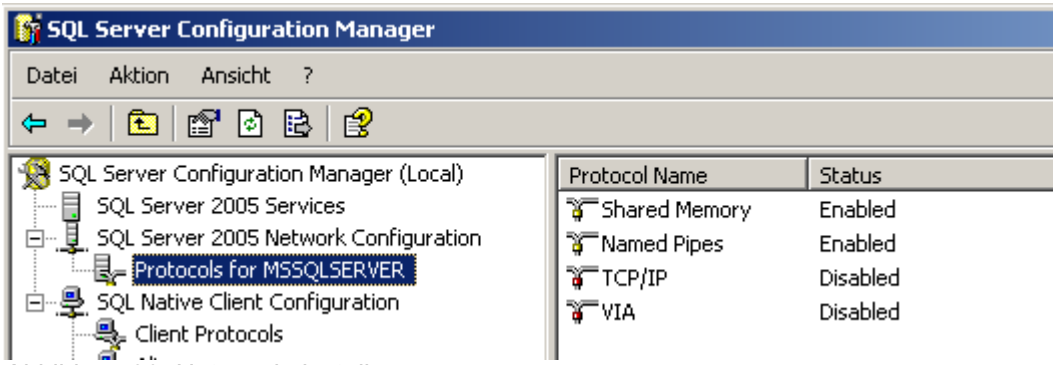
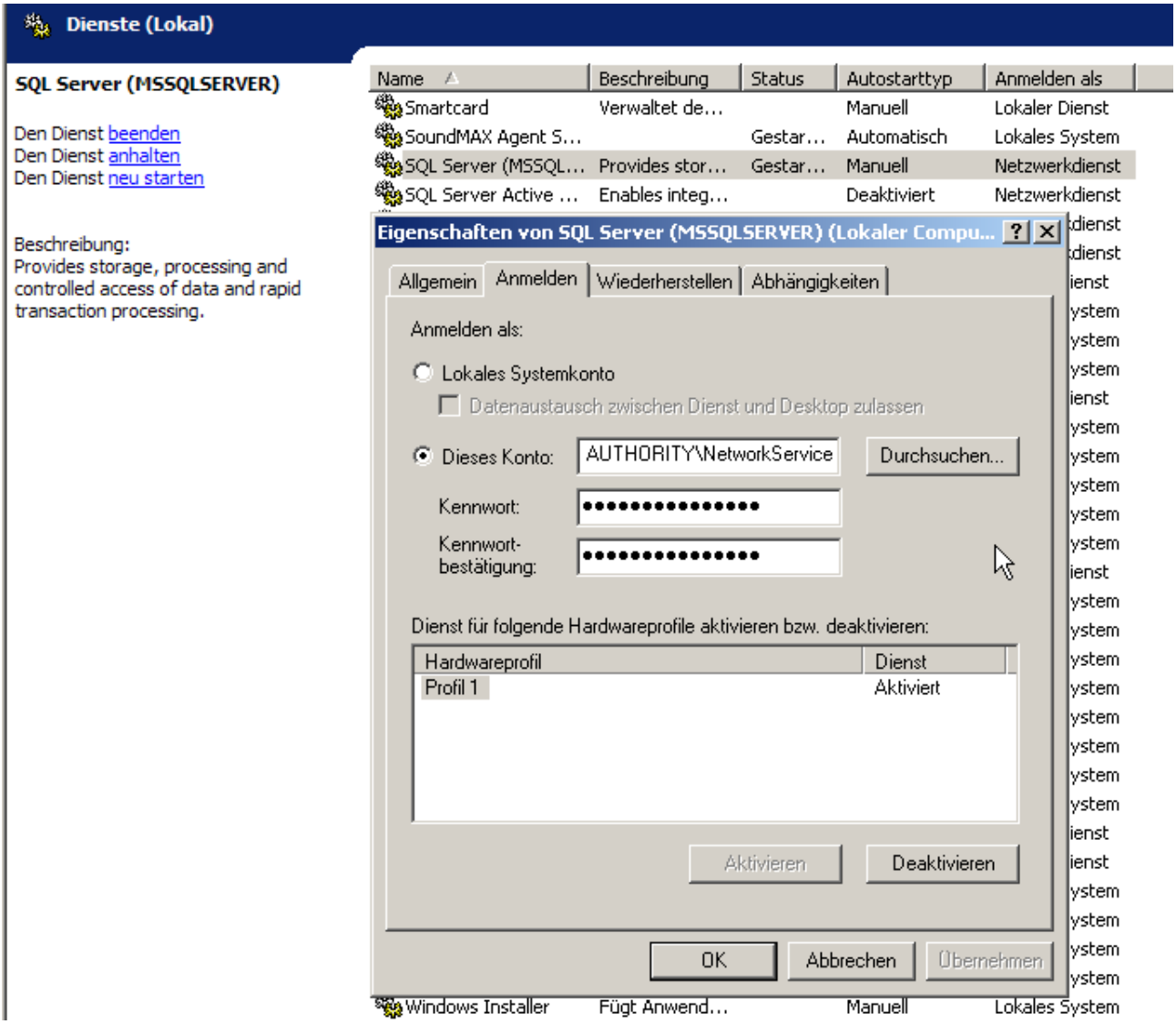


Abbildung 20: Netzwerkeinstellungen

10.3 Rechtvergabe des SQL-Server-Dienstes



Der Dienst vom SQL-Server muss unter einem anderen User laufen.

11 Trigger

11.1 Vorbereitung

Wir erstellen eine Tabelle:

```
create database KinoSQL
USE [KinoSQL]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Person](
      [id] [int] IDENTITY(1,1) NOT NULL,
      [Nachname] [varchar](400) NULL,
      [idVorgestzter] [int] NULL,
      [geAendertVon] [varchar](50) NULL,
      [geAendertAm] [datetime] NULL,
 CONSTRAINT [PK__Person__56B3DD81] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
```

11.2 Trigger erstellen: Änderungen

Falls ein Eintrag in der Tabelle geändert wird, dann wird in die Spalte geändert am das aktuelle Datum hinzugefügt und in die andere Spalte der Username eingefügt.

```
CREATE TRIGGER tr_insUP
      ON Person
      AFTER INSERT, UPDATE
AS
BEGIN
      SET NOCOUNT ON;
UPDATE Person
      SET geAendertAm=GETDATE(),
      geAendertVon=SYSTEM_USER FROM Person, inserted
WHERE Person.id=inserted.id
END
```

11.3 Trigger erstellen: Löschungen

Zuerst erstellen wir eine neue Tabelle für die gelöschten Objekte:

```
<tbd: CREATE TABLE Person_Papierkorb>
```

Und nun erstellen wir den Trigger:

```
CREATE TRIGGER tr_Papierkorb
```

```
      ON Person
      AFTER DELETE
AS
BEGIN
      SET NOCOUNT ON;
INSERT into Person_Papierkorb (id, Nachname, idVorgesetzter)
SELECT id, Nachname, idVorgesetzter from deleted
END
```

12 Diverses

12.1 Informationen zu den Tests

- Die Test finden auf dem Papier statt.
- Es dürfen Zusammenfassungen gebraucht werden. Auch fremde Zusammenfassungen.
- Ja, auch diese hier!
- So glaub es doch!

12.1.1 Test Nummer 1 vom 2009-04-29

- Backup / Restore
  - Datenbank von einem Backup-File restoren.
  - Der Restore-Zeitpunkt kann sekundengenau angegeben werden.
- Indexe
  - B-Baum Zeichen (Ordnung: eher 1, max 2; Anzahl Elemente: 6)
    - Screenshot mit B-Baum. Aufgabe: Lösche Nummer 42
    - Screenshot mit B-Baum: Aufgabe: Füge Nummer 23 ein
    - Baum neu zeichnen! Nicht ändern. Geht besser.
    - Verwende dazu doch bitte die Master-Tabellen-Ansicht of non confisuon Eris! Diese wird dich nicht verwirren und dir dabei viel helfen!
  - Wie viele Einträge können indexiert werden mit Ordnung d und x Ebenen
- Taschenrechner nicht vergessen!!!
- Prozeduren schreiben zu Datümmer (Wert zu Datum addieren) [Dumm, Dümmer, Datümmer]

12.1.2 Test Nummer 2 vom 2009-05-13

- Prozeduren schreiben zu Datümmer (Wert zu Datum addieren) [Dumm, Dümmer, Datümmer]
- Pivotieren mit Case-Struktur

12.1.3 Test Nummer 3 vom 2009-06-10

- Stored Procedures
- Stores Functions
- Pivot-Aufgabe

- Nicht: CTE

12.2 TBD

- Gebacktupte backups, DB mit BACKUPs sichern.. .<tbd>

12.3 Neuen Benutzer erstellen

Wir erstellen einen neuen Benutzer.

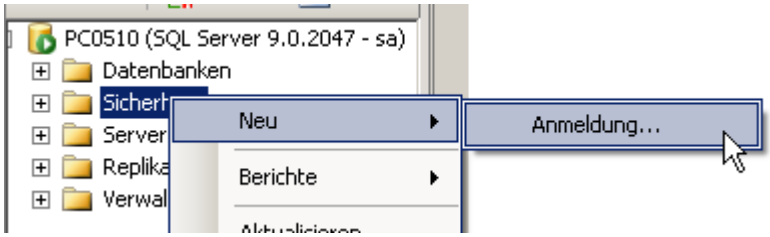


Abbildung 21: Neuen Benutzer erstellen

Und vergeben dem die nötigen Rechte:

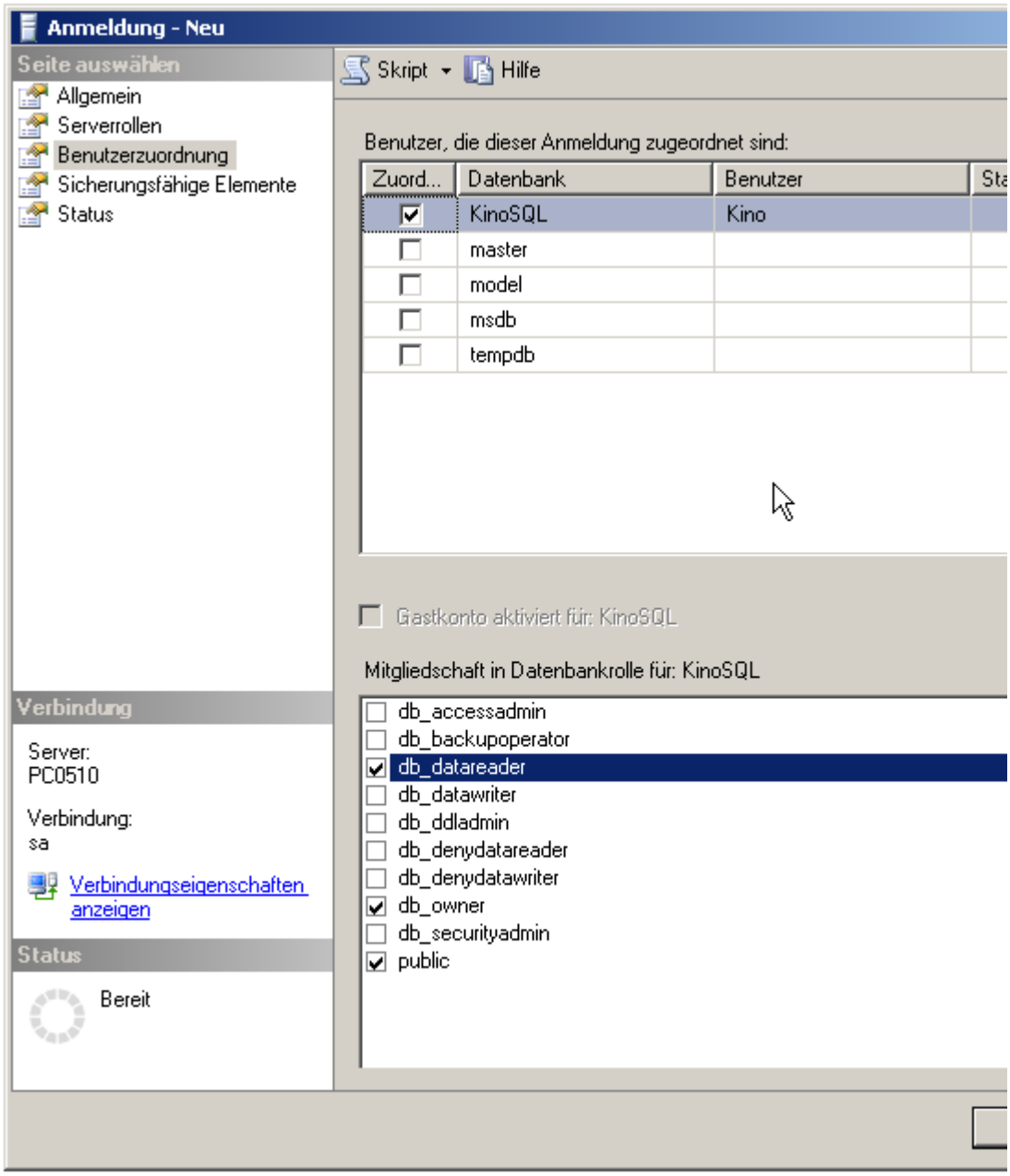


Abbildung 22: Rechte vergeben

12.4 Access

Mit Microsoft Access kann man ganz einfach eine Verbindung mit einem MS SQL-Server herstellen.

Dazu eröffnen wir ein neues Projekt. Wir speichern es im Dateiformat .adp ab.

Abbildung 23: Neues Access-Projekt

Jetzt können wir die Anmeldeinformationen eingeben:

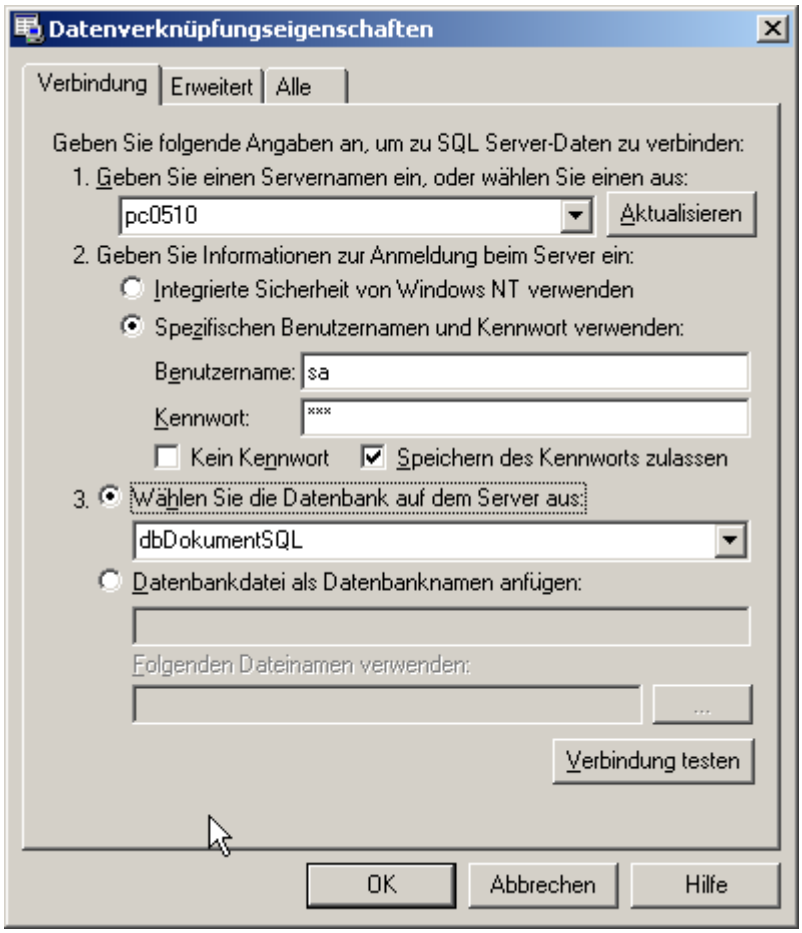
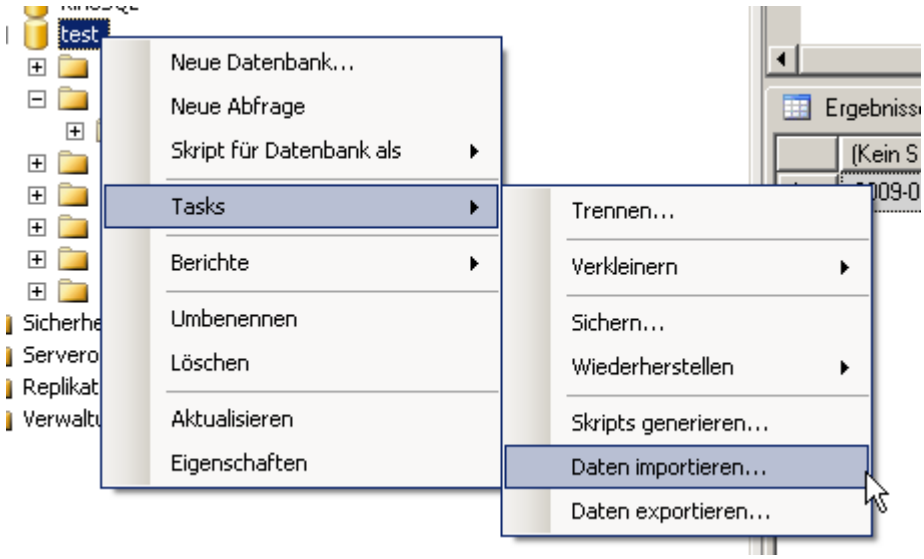


Abbildung 24: Mit MS SQL-Server verbinden

Dann sehen wir die Tabellen der Datenbank und können diese Manipulieren.

12.5 Daten importieren



12.6 SQL über die Kommandozeile

```
R:\>sqlcmd -H localhost -U sa -P sql
1> use KinoSQL;
2> go
Changed database context to 'KinoSQL'.
1> select * from Person;
2> go
id          Nachname

             idVorgestzter geAendertVon
geAendertAm
-----------
------------------------------------------------------------------------------
------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
----------- ------------- ------------------------------------------------
----------------------

(0 Zeilen betroffen)
1>
```

13 Codeschnipsel

13.1 .NET-Framework auf dem SQL-Server einschalten

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'clr enabled', 1;
GO
RECONFIGURE;
GO
```

13.2 Datensätze generieren

```
SELECT TOP 10000000 --<<<LOOK! CHANGE THIS NUMBER TO CHANGE THE NUMBER OF ROWS!
        RowNum   = IDENTITY(INT,1,1),
        SomeID   = ABS(CHECKSUM(NEWID()))%25+1, --<<<LOOK! CHANGE THIS NUMBER TO
1/400th THE ROW COUNT
        SomeCode = CHAR(ABS(CHECKSUM(NEWID()))%26+65) +
CHAR(ABS(CHECKSUM(NEWID()))%26+65)
   INTO dbo.TestData
   FROM Master.dbo.SysColumns t1,
        Master.dbo.SysColumns t2
GO
```

13.3 Passwortgenerator

Prozedur erstellen

```
CREATE PROCEDURE dbo.uspCreatePassword(
  @UpperCaseItems SMALLINT
  ,    @LowerCaseItems SMALLINT
  ,    @NumberItems SMALLINT
  ,    @SpecialItems SMALLINT)
 AS
 SET NOCOUNT ON
```

Variabeln Deklarieren

```
 DECLARE @UpperCase VARCHAR(26)
        , @LowerCase VARCHAR(26)
              , @Numbers VARCHAR(10)
              , @Special VARCHAR(13)
              , @Temp VARCHAR(8000)
              , @Password VARCHAR(8000)
              , @i SMALLINT
              , @c VARCHAR(1)
              , @v TINYINT
```

Variable setzen

```
-- Set the default items in each group of characters
SELECT @UpperCase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    ,    @LowerCase = 'abcdefghijklmnopqrstuvwxyz'
       ,    @Numbers = '0123456789'
       ,    @Special = '!@#$%&*()_+-='
       ,    @Temp = ''
```

```
      ,    @Password = ''
```

Verzweigungen

```
-- Enforce some limits on the length of the password
IF @UpperCaseItems > 20
  SET @UpperCaseItems = 20
IF @LowerCaseItems > 20
  SET @LowerCaseItems = 20
IF @NumberItems > 20
  SET @NumberItems = 20
IF @SpecialItems > 20
  SET @SpecialItems = 20
```

ABS = Betrag

```
-- Get the Upper Case Items
SET @i = ABS(@UpperCaseItems)
WHILE @i > 0 AND LEN(@UpperCase) > 0
  SELECT @v = ABS(CAST(CAST(NEWID() AS BINARY(16)) AS BIGINT)) % LEN(@UpperCase)
+ 1
  ,    @c = SUBSTRING(@UpperCase, @v, 1)
  ,    @UpperCase = CASE
       WHEN @UpperCaseItems < 0
               THEN STUFF(@UpperCase, @v, 1, '')
            ELSE @UpperCase
  END
  ,    @Temp = @Temp + @c
  ,    @i = @i - 1
-- Get the Lower Case Items
SET @i = ABS(@LowerCaseItems)
WHILE @i > 0 AND LEN(@LowerCase) > 0
  SELECT @v = ABS(CAST(CAST(NEWID() AS BINARY(16)) AS BIGINT)) % LEN(@LowerCase)
+ 1
  ,    @c = SUBSTRING(@LowerCase, @v, 1)
  ,    @LowerCase = CASE
       WHEN @LowerCaseItems < 0
              THEN STUFF(@LowerCase, @v, 1, '')
            ELSE @LowerCase
  END
  ,    @Temp = @Temp + @c
  ,    @i = @i - 1

-- Get the Number Items
SET @i = ABS(@NumberItems)
WHILE @i > 0 AND LEN(@Numbers) > 0
  SELECT @v = ABS(CAST(CAST(NEWID() AS BINARY(16)) AS BIGINT)) % LEN(@Numbers) +
1
  ,    @c = SUBSTRING(@Numbers, @v, 1)
  ,    @Numbers = CASE
       WHEN @NumberItems < 0
              THEN STUFF(@Numbers, @v, 1, '')
            ELSE @Numbers
  END
  ,    @Temp = @Temp + @c
  ,    @i = @i - 1

-- Get the Special Items
SET @i = ABS(@SpecialItems)
WHILE @i > 0 AND LEN(@Special) > 0
  SELECT @v = ABS(CAST(CAST(NEWID() AS BINARY(16)) AS BIGINT)) % LEN(@Special) +
```

```
1
  ,    @c = SUBSTRING(@Special, @v, 1)
  ,    @Special = CASE
       WHEN @SpecialItems < 0
               THEN STUFF(@Special, @v, 1, '')
          ELSE @Special
  END
  ,    @Temp = @Temp + @c
  ,    @i = @i - 1

-- Scramble the order of the selected items
WHILE LEN(@Temp) > 0
  SELECT @v = ABS(CAST(CAST(NEWID() AS BINARY(16)) AS BIGINT)) % LEN(@Temp) + 1
  ,    @Password = @Password + SUBSTRING(@Temp, @v, 1)
  ,    @Temp = STUFF(@Temp, @v, 1, '')
```

Folgender Befehlt gibt das Passwort aus:

```
SELECT @Password
```

Passwort generieren:

```
uspCreatePassword 3,3,2,2
```

14 Glossar

15 Gute Links

-

Stichwortverzeichnis