net.com
networking & communications

KLEMENS KONOPASEK

4., aktualisierte Auflage

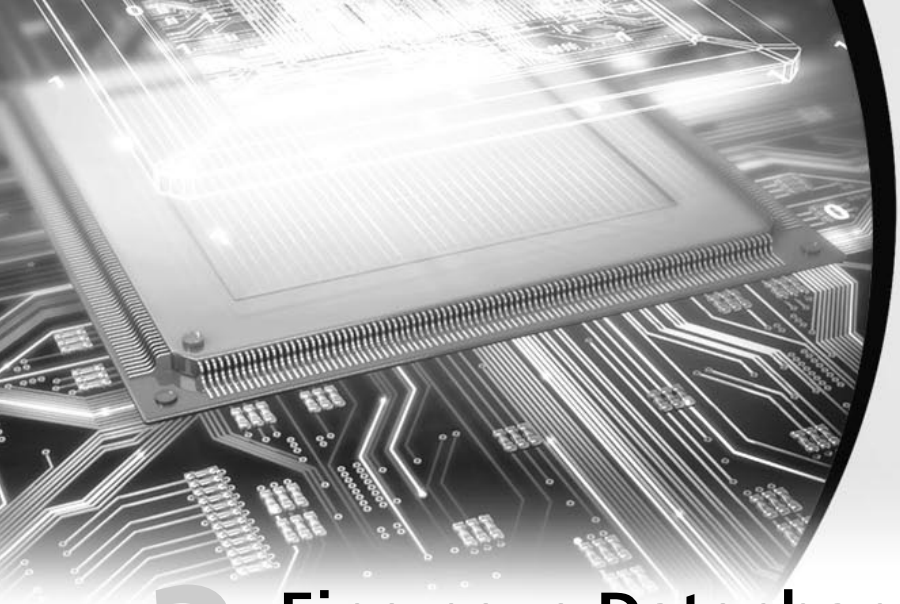
SQL Server 2008 R2
Der schnelle Einstieg

Abfragen, Transact-SQL, Entwicklung und Verwaltung

Für SQL Server 2008 und 2008 R2

ADDISON-WESLEY

CD



3 Eine neue Datenbank erstellen

Nachdem wir den SQL Server 2008 R2 installiert und uns ein wenig mit den grafischen Tools vertraut gemacht haben, ist es höchste Zeit, eine erste Datenbank zu erstellen. In diesem Kapitel lesen Sie zum einen, wie Sie dazu vorgehen, und zum anderen möchte ich Sie mit den Hintergründen vertraut machen.

3.1 Erstellen einer neuen Datenbank

Zum Anlegen einer neuen Datenbank verwenden Sie vorzugsweise das SQL Server Management Studio. Hier können Sie wahlweise die grafische Oberfläche verwenden oder in einem Abfrageeditor-Fenster mit der Anweisung `CREATE DATABASE` eine neue Datenbank erstellen. Hierbei würde es genügen, zusätzlich den Namen der Datenbank anzugeben. Diese wird dann mit allen Standardeinstellungen erstellt.

Probieren Sie es ruhig aus! Geben Sie in einem Abfrageeditor-Fenster die Anweisung `CREATE DATABASE kapitel3` ein. Führen Sie diese Anweisung aus, so haben Sie schon Ihre erste Datenbank erstellt. Die Dateien dieser Datenbank finden Sie im Ordner *...\MSSQL10.MSSQLSERVER\MSSQL\DATA* unter dem Verzeichnis, das Sie während des Setups als Datenverzeichnis angegeben haben. Der Teil des Ordnernamens *MSSQLSERVER* kann variieren, sofern Sie eine benannte Instanz installiert haben. Die R2 des SQL Server 2008 verwendet anstelle des Ordners MSSQL10.MSSQLSERVER nun das Verzeichnis MSSQL10_50.MSSQLSERVER.

3.1.1 Bestandteile einer Datenbank

Eine SQL Server-Datenbank besteht aus mehreren Dateien, mindestens sind es immer zwei. Bei größeren Datenbanken kann es aber aus Speicherplatz- oder Performancegründen sinnvoll sein, mehrere Dateien für eine Datenbank zu verwenden. Diese können zu logischen Dateigruppen zusammengefasst werden.

Datenbankdateien

Eine Datenbank besteht aus Daten- und Transaktionsprotokolldateien. Eine Übersicht über diese Datenbankdateien liefert Ihnen die nachfolgende Tabelle.

Datei	Typ	Beschreibung
Primäre Datendatei	MDF	Die primäre Datendatei (Master Data File) gibt es in jeder Datenbank. Bei Datenbanken von kleinerer und mittlerer Größe ist sie auch die einzige Datendatei. In ihr werden neben den Benutzerdatenbankobjekten (Tabellen usw.) auch die Systemobjekte der Datenbank gespeichert. In den Systemtabellen werden unter anderem die gesamte Struktur der Datenbank, deren Benutzer sowie alle Berechtigungen gespeichert. Die primäre Datendatei wird immer in der Dateigruppe *PRIMARY* gespeichert.
Weitere Datendateien	NDF	Bei großen Datenbanken können weitere Datendateien ergänzt werden, um Datenbankobjekte auf diese zu verteilen.
Transaktionsprotokolldateien	LDF	Es können eine oder mehrere Transaktionsprotokolldateien für eine Datenbank festgelegt werden. Im Transaktionsprotokoll werden alle Schreibvorgänge in der Datenbank protokolliert. Diese Informationen dienen zur Steuerung von Transaktionen. Fällt der Datenbankserver aus – zum Beispiel durch einen ungesicherten Stromausfall –, werden beim Neustart des Systems alle nicht abgeschlossenen Transaktionen automatisch zurückgesetzt. Des Weiteren wird das Transaktionsprotokoll für Backup- und Recovery-Vorgänge benötigt. (Mehr über die Bedeutung des Transaktionsprotokolls lesen Sie in Kapitel 5 zu den Transaktionen; in Kapitel 8 erfahren Sie mehr zum Thema Sicherung und zum Wiederherstellen von Datenbanken.)

Tabelle 3.1: *Dateien einer Datenbank*

Eine Standarddatenbank besteht in der Regel aus der *MDF*-Datei sowie einer Transaktionsprotokolldatei (*LDF*). Weitere Dateien werden meist nur bei größeren Systemen verwendet. Unter den nachfolgenden Voraussetzungen kann es sinnvoll oder sogar notwendig sein, mehrere Datendateien für eine Datenbank einzusetzen:

- Der benötigte Speicherplatzbedarf kann auf einem Datenträger nicht zur Verfügung gestellt werden, womit eine Aufteilung auf mehrere Dateien auf unterschiedlichen Datenträgern unumgänglich ist.
- Performancevorteile können erzielt werden, wenn zum Beispiel Tabellen in einer Datei und Indizes in einer anderen Datei gespeichert werden. Liegen diese auf unterschiedlichen Datenträgern, die an unterschiedlichen Controllern im Server angeschlossen sind,

können Lesevorgänge parallelisiert werden. Nachdem ein Indexeintrag auf einem Datenträger gelesen worden ist, kann bereits der nächste Indexeintrag gesucht und gelesen werden, während in der Zwischenzeit die Daten zum ersten Indexeintrag vom anderen Datenträger eingelesen werden.

- Durch die Aufteilung der Daten auf mehrere Dateien und Dateigruppen können diese separat gesichert (Dateigruppensicherung) und wiederhergestellt werden. Dies ist vor allem von Vorteil, wenn die Datenbank aufgrund ihrer Größe in einem Durchgang nicht komplett gesichert werden könnte.
- Einzelne Dateien können in schreibgeschützten Dateigruppen enthalten sein. Dort können Sie Daten unterbringen, die unveränderlich bleiben sollen. Sie können zum Beispiel Archivdaten in solche Dateien einlagern.

Für eine kleine Datenbank bis zu einer Größe von einem Gigabyte wird üblicherweise noch keine Aufteilung in mehrere Dateien vorgenommen.

Für jede Datei einer Datenbank können die folgenden Parameter vergeben werden:

- *Logischer Name*: Der logische Name ist der interne Name der Datei, über den sie mit SQL-Anweisungen angesprochen werden kann. Dieser dient quasi als Brücke zwischen der Datenbank und den physischen Datenbankdateien. Dieser Name wird zum Beispiel bei der Anweisung `RESTORE DATABASE` verwendet, wenn beim Wiederherstellen der Datenbank die Datei an einen anderen Pfad verschoben werden soll.
- *Anfangsgröße*: Die Anfangsgröße bestimmt den Speicherplatz, den die Datei bei ihrer Erstellung auf dem Datenträger belegt.

Verwenden Sie hier gleich eine angemessene Größe, um eine Fragmentierung der Datei durch viele kleine Vergrößerungen zu vermeiden.

- *Automatische Vergrößerung*: Eine Datei kann automatisch vergrößert werden, sobald sie voll ist. Die Vergrößerung kann als Prozentsatz der bisherigen Größe oder als fixe Größe in Megabyte angegeben werden. Zusätzlich lässt sich festlegen, ob dieses Wachstum unbeschränkt oder bis zu einer gewissen Maximalgröße erfolgen soll.

Ist die Maximalgröße erreicht, die automatische Vergrößerung nicht aktiviert oder einfach nur der Datenträger voll, so kann in der Datenbank nur noch gelesen werden. Schreibvorgänge sind erst nach Schaffen weiteren Speicherplatzes möglich. Dies kann zum Beispiel durch Hinzufügen von zusätzlichen Dateien oder durch das Verschieben von Dateien auf andere Datenträger erfolgen. Für Letzteres muss die Datenbank allerdings offline sein.

- *Physischer Dateiname*: Dies ist der Name und Pfad der Datei auf dem Dateisystem mit einer der Erweiterungen *MDF*, *NDF* oder *LDF*.

Dateigruppen

Jede Datendatei einer Datenbank wird in einer *Dateigruppe* gespeichert. Dabei können Sie selbst entscheiden, ob Sie mehrere Dateien in einer Dateigruppe oder in jeweils einer eigenen Dateigruppe anlegen möchten.

Jede Datenbank enthält die Standarddateigruppe *PRIMARY*. Diese kann nicht gelöscht werden, da die primäre Datendatei (*MDF*) immer in dieser Dateigruppe gespeichert wird. Jedes Datenbankobjekt, dem beim Erstellen keine Dateigruppe zugewiesen wird, wird standardmäßig in der Dateigruppe *PRIMARY* gespeichert.

Beim Anlegen eines Datenbankobjekts (Tabelle, Index usw.) kann die Dateigruppe (nicht die Datendatei!) angegeben werden, in der das Objekt gespeichert werden soll. Bei Tabellen legen Sie somit fest, wo die Daten physisch abgelegt werden.

Wann sollten mehrere Dateien in einer Dateigruppe gespeichert und wann auf mehrere Dateigruppen aufgeteilt werden?

- *Gemeinsame Dateigruppe*: Eine Dateigruppe werden Sie dann gemeinsam für mehrere Datendateien verwenden, falls mehrere Dateien nur aufgrund von Speicherplatzmangel erstellt wurden. In diesem Fall verteilt der SQL Server die Daten selbst auf diese Dateien. Sie haben keinen Einfluss darauf, in welcher Datei zum Beispiel eine Tabelle physisch abgelegt wird. Wenn Sie beispielsweise eine weitere Datei ergänzen, um den zur Verfügung stehenden Speicherplatz zu erweitern, fügen Sie diese derselben Dateigruppe an.
- *Getrennte Dateigruppe*: Sie verwenden eine eigene Dateigruppe für eine Datei, falls Sie diese gezielt als Speicherort für Datenbankobjekte angeben möchten. Dies ist zum Beispiel der Fall, wenn Sie Tabellen auf einem Laufwerk und Indizes auf einem anderen Laufwerk speichern möchten.

Dateigruppen werden nur für Datendateien verwendet. Transaktionsprotokolldateien werden nicht in Dateigruppen, sondern gesondert gespeichert.

Die beiden nachfolgenden Grafiken sollen Ihnen noch einmal einen Überblick über mögliche Realisierungsvarianten geben. Abbildung 3.1 zeigt Ihnen eine Standarddatenbank, die aus der primären Datendatei mit dem logischen Namen *db_data1* in der primären Dateigruppe und einer Transaktionsprotokolldatei besteht.

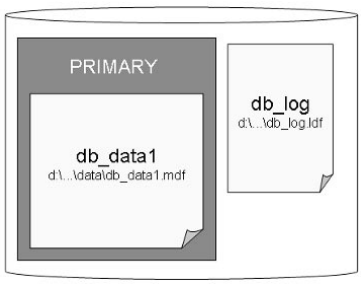


Abbildung 3.1: *Einfache Datenbank mit einer Datendatei in einer Dateigruppe*

Abbildung 3.2 zeigt eine mögliche Variante für eine Datenbank aus mehreren Datendateien. Die primäre Datendatei *db_data1* sowie die weitere Datendatei *db_data2* gehören der Dateigruppe *PRIMARY* an. Die Datendatei *db_data3* befindet sich in einer eigenen Dateigruppe mit dem Namen *DATEN*. Für die Datendatei *db_index* ist ebenfalls eine eigene Dateigruppe mit dem Namen *INDEX* angelegt worden, damit diese beim Erstellen von Indizes als Zieldateigruppe angegeben werden kann. Letztere befindet sich physisch auf einem anderen Datenträger. Das Transaktionsprotokoll kann keiner Dateigruppe zugeordnet werden.

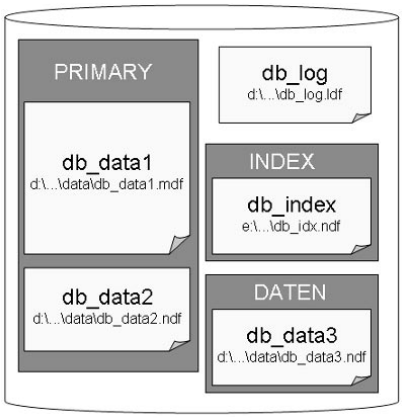


Abbildung 3.2: *Datenbank mit mehreren Datendateien und Dateigruppen*

Nach diesem einleitenden Überblick über den Aufbau einer Datenbank möchte ich im nächsten Schritt zum Anlegen einer Datenbank mit dem Management Studio kommen.

3.1.2 Datenbank mit dem grafischen Tool anlegen

Das Anlegen einer Datenbank mit dem Management Studio ist eine sehr einfache Angelegenheit. Wenn Sie sich die physische Struktur – wie im vorigen Abschnitt beschrieben – der Datenbank schon überlegt haben, können Sie sogleich loslegen.

Im ersten Schritt legen wir eine Datenbank an, die aus lediglich einer Datendatei besteht.

1. Öffnen Sie das Management Studio und melden Sie sich an dem SQL Server an, auf dem Sie die Datenbank erstellen möchten.
2. Markieren Sie den Ordner *Datenbanken* im Objekt-Explorer und wählen Sie im Kontextmenü den Befehl NEUE DATENBANK... aus.

Abbildung 3.3: *Dialog »Neue Datenbank«*

3. Im Dialog NEUE DATENBANK tragen Sie vorerst *Marketing* als Namen für die neue Datenbank ein. Dieser wird automatisch als logischer Name für die primäre Datendatei übernommen. Das Transaktionsprotokoll erhält denselben Namen mit dem Zusatz *_log*.

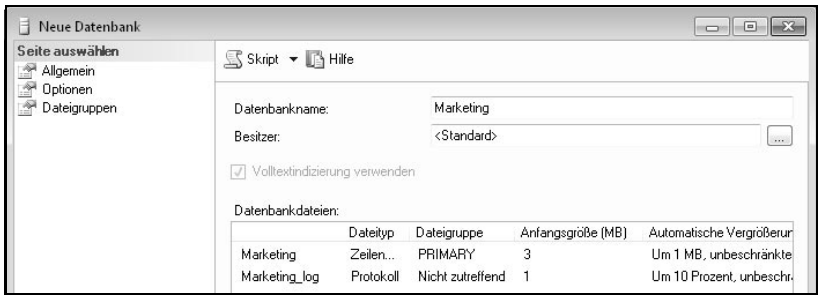


Abbildung 3.4: *Name für Datenbank vergeben*

4. Wenn Sie möchten, können Sie einen Benutzer als Besitzer für die Datenbank angeben. Tun Sie das nicht und übernehmen Sie den Eintrag <STANDARD>, werden Sie beim Anlegen selber als Datenbankbesitzer übernommen.
5. Ändern Sie die Anfangsgröße der primären Datendatei zum Beispiel auf 30 MB. Als Standardwert wird an dieser Stelle lediglich eine Größe von 3 MB vorgeschlagen.

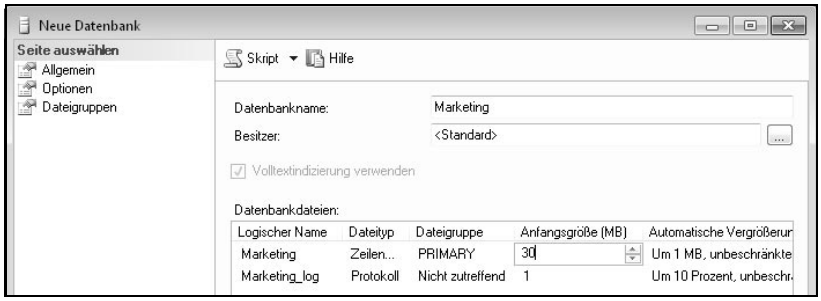


Abbildung 3.5: *Anfangsgröße der Datendatei festlegen*

6. Standardmäßig ist für Datendateien die automatische Vergrößerung aktiviert, und zwar unbeschränkt um jeweils ein Megabyte. Um diese Einstellung anzupassen, klicken Sie auf die Schaltfläche mit den drei Punkten in der betreffenden Zeile.

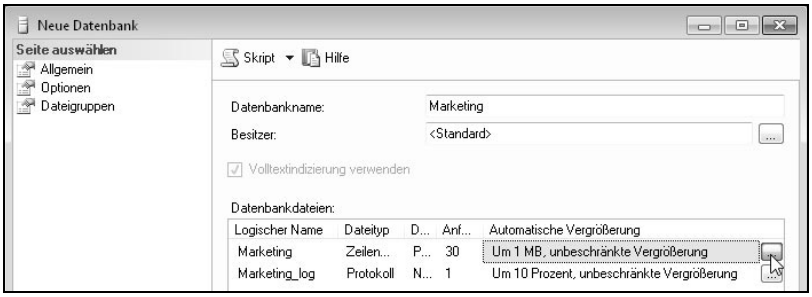


Abbildung 3.6: *Automatische Vergrößerung einstellen*

7. Im Dialog AUTOMATISCHE VERGRÖSSERUNG ÄNDERN können Sie die gewünschten Einstellungen vornehmen. Ändern Sie die Dateivergrößerung zum Beispiel auf 5 MB und beschränken Sie das Wachstum auf eine maximale Dateigröße von 500 MB.

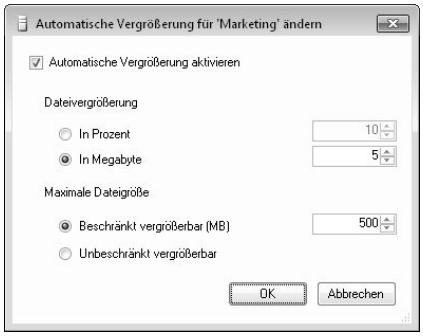


Abbildung 3.7: *Einstellungen für die automatische Vergrößerung anpassen*

Achten Sie darauf, dass bei einem Prozentwachstum das Ausmaß der Vergrößerung bei jedem Vorgang ebenso anwächst.

8. Legen Sie nun den Pfad für die Datendatei sowie für die Protokolldatei fest. Sie können jeden lokalen Pfad auf dem Server-Rechner auswählen. Greifen Sie remote auf den Server zu, sehen Sie bei der Auswahl des Pfades die Verzeichnisstruktur des Servers, nicht jedoch die Ihres Rechners.

Als Ziel können lediglich lokale Pfade und keine Netzlaufwerke verwendet werden. Wenn Sie den SQL Server-Dienst mit einem Domänenkonto ausführen, muss dieses Konto über Vollzugriff auf die verwendeten Pfade verfügen.

Der Dateiname wird im Management Studio direkt in das dafür vorgesehene Feld eingegeben. Tragen Sie hier keinen Namen ein, wird später der logische Name für die Datendatei mit der Dateierweiterung *MDF* sowie für die Protokolldatei mit der Erweiterung *LDF* übernommen. Da die Dateierweiterungen automatisch ergänzt werden, müssen Sie diese bei der Eingabe der Dateinamen nicht berücksichtigen.

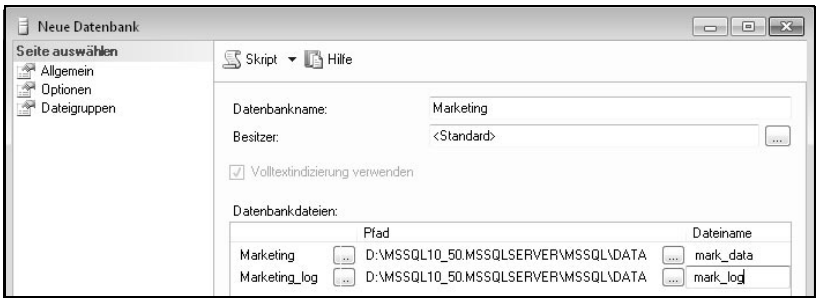


Abbildung 3.8: *Pfad und Name für Datenbankdateien angeben*

9. Falls Sie möchten, können Sie noch auf die Seite OPTIONEN wechseln. Dort können Sie verschiedene Einstellungen wie zum Beispiel die SORTIERUNG (Collation), das WIEDERHERSTELLUNGSMODELL und den KOMPATIBILITÄTSGRAD vornehmen.

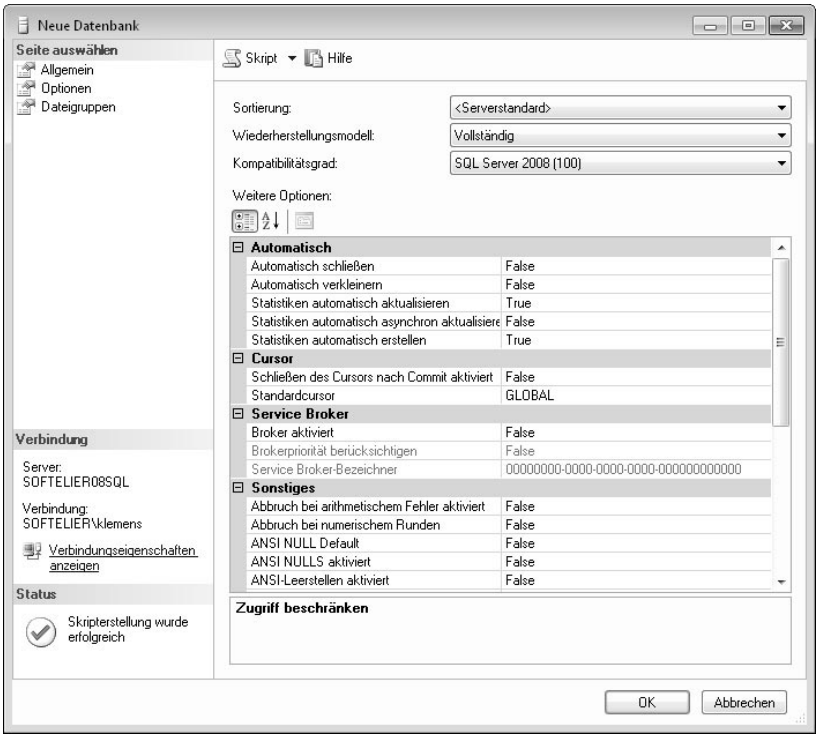


Abbildung 3.9: *Datenbankoptionen einstellen*

Übernehmen Sie bei der SORTIERUNG – Sie erinnern sich, diese legt fest, nach welchen Sprachgegebenheiten der Vergleich von Texten vorgenommen wird – den Serverstandard. Dies ist jene Einstellung, die Sie beim Setup des Servers festgelegt haben. Als Wiederherstellungsmodell sollten Sie ebenfalls die Voreinstellung *Vollständig* beibehalten. Damit kann die Datenbank bei einem Crash bis zum Zeitpunkt desselben wiederhergestellt werden. Was diese Einstellung im Detail bedeutet, wird in Kapitel 8 erläutert. Der Kompatibilitätsgrad legt fest, mit welchen Features welcher Version diese Datenbank kompatibel ist. Hier könnte auch eine der beiden Vorgängerversionen 2000 und 2005 gewählt werden. Dies ist aber nur in Ausnahmefällen sinnvoll, wenn zum Beispiel die verwendete Clientanwendung nur für den Server einer Vorversion zertifiziert ist. Damit werden nur Features der jeweiligen Version unterstützt. Der SQL Server 2008 und 2008 R2 verwenden denselben Kompatibilitätsgrad, hier wird nicht unterschieden.

10. Legen Sie die Datenbank nun an, indem Sie Ihre Eingaben mit einem Klick auf die Schaltfläche OK abschließen.

Die neue Datenbank wird anschließend im Objekt-Explorer angezeigt.

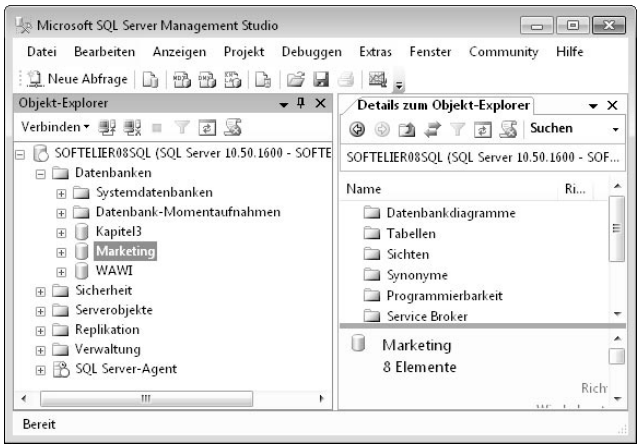


Abbildung 3.10:
Neue Datenbank im Management Studio

Um eine Datenbank nach dem Muster von Abbildung 3.2 mit mehreren Datendateien und Dateigruppen zu erstellen, definieren Sie beim Anlegen der Datenbank über die Seite DATEIGRUPPEN zuerst die benötigten Dateigruppen über die Schaltfläche HINZUFÜGEN. In unserem Beispiel sind dies die Dateigruppen *DATEN* und *INDEX*.

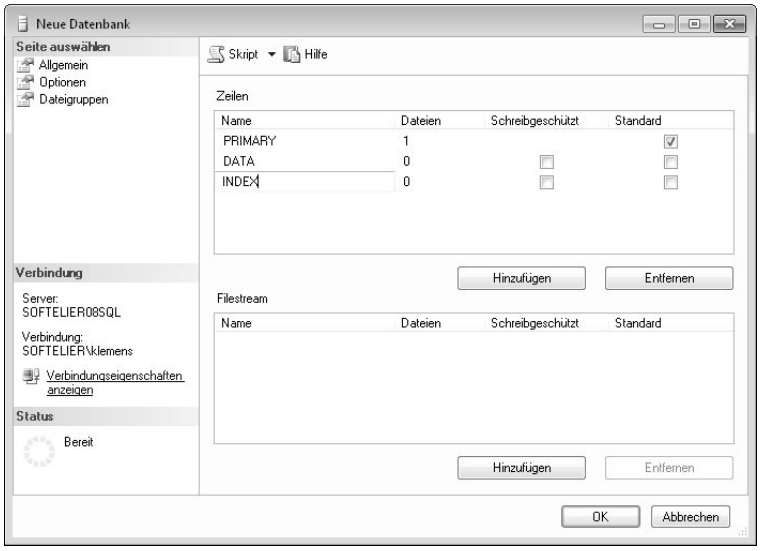


Abbildung 3.11: *Dateigruppen anlegen*

NEU 2008

An dieser Stelle sehen Sie auch ein neues Feature der Version 2008. Hier können Sie im unteren Dialogbereich auch Dateigruppen für *Filestream* hinzufügen. Auf das Thema *Filestream* – hierbei geht es zum Beispiel darum, bestimmte Objekte außerhalb der Datenbank zu speichern, aber automatisiert mit dieser gemeinsam zu sichern – werden wir später in diesem Kapitel noch zu sprechen kommen.

Auf der Seite ALLGEMEIN können Sie beim Hinzufügen von weiteren Datendateien die zuvor angelegten Dateigruppen aus der Liste auswählen. Wie Sie in der Abbildung sehen, kann bei einer Datenbankdatei, deren Typ als PROTOKOLL definiert ist, keine Dateigruppe ausgewählt werden.

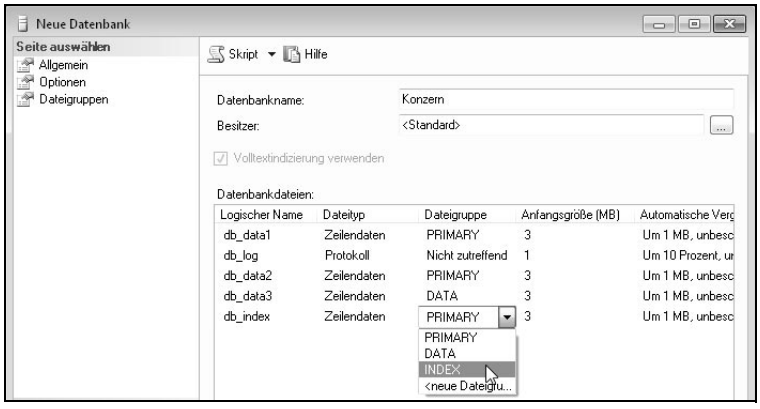


Abbildung 3.12: *Datendateien ergänzen und Dateigruppen zuordnen*

Falls Sie möchten, können Sie eine neue Dateigruppe auch direkt beim Ergänzen einer Datendatei hinzufügen. Wählen Sie dazu in der Liste den Eintrag *<neue Dateigruppe>* aus und legen Sie die Dateigruppe über den nachfolgenden Dialog an.

Welche Anfangsgröße sollte man für das Transaktionsprotokoll wählen?

Was die Größe des Transaktionsprotokolls betrifft, so gilt folgende Faustregel:

- Bei einer standardmäßigen OLTP-Anwendung (OLTP = Online Transactional Processing) beträgt die Größe des Transaktionsprotokolls etwa ein Drittel der Größe der Datendateien. (Zum Beispiel bei einer Warenwirtschaftsanwendung).
- Bei einer Archivdatenbank mit einem geringen Anteil an Schreibvorgängen, die in erster Linie für Suchen und Recherchen verwendet wird, wird der reale Wert weit darunter liegen. (Zum Beispiel eine Datenbank für die Verwaltung von Museumsbeständen.)
- Bei Datenbanken mit einem enormen Schreibaufkommen kann das Transaktionsprotokoll auch größer als die Datendateien sein. (Zum Beispiel bei einer Datenbank für die Verarbeitung von Messwerten eines Produktionsvorgangs.)

Je nach Ihrer Anwendung wählen Sie einen Ihnen passend erscheinenden Wert für die Anfangsgröße des Transaktionsprotokolls.

Da im Transaktionsprotokoll sehr viele Schreibzugriffe erfolgen, ist es hier von besonderer Bedeutung, dass die Datei nicht fragmentiert ist. Wählen Sie daher im Zweifelsfall – sofern möglich – eine etwas höhere Anfangsgröße. Aus Performancesicht ist es sinnvoll, das Transaktionsprotokoll nicht auf demselben Datenträger mit den Datendateien zu speichern.

3.1.3 Datenbank über eine SQL-Anweisung erstellen

Wie bereits früher in diesem Kapitel erwähnt, lässt sich eine Datenbank ganz schnell mit der Anweisung `CREATE DATABASE name` über ein Abfrageeditor-Fenster (wie in Kapitel 2 beschrieben) erstellen.

Wenn Sie mit dieser kurzen Anweisung eine Datenbank anlegen, werden für alle Einstellungen Standardwerte herangezogen:

- Es gibt nur die Dateigruppe *PRIMARY* und eine Datendatei.
- Der Name der Datenbank wird als logischer Name für die primäre Datendatei verwendet. Er wird ebenso für den physischen Namen der Datei herangezogen, die im Standard-Datenbankordner angelegt wird. Die Anfangsgröße dieser Datei beträgt 3 MB, sie wird unbeschränkt um jeweils 1 MB wachsen.
- Für die Transaktionsprotokolldatei wird der Datenbankname mit der Erweiterung *_LOG* ergänzt. Ihre Anfangsgröße beträgt 1 MB, sie wächst unbeschränkt jeweils um 10 Prozent. Auch sie wird im Standard-Datenbankordner angelegt.
- Sie selber sind der Besitzer der Datenbank.

- Der Serverstandard wird für die Sortierung herangezogen, ebenso das vollständige Wiederherstellungsmodell und der Kompatibilitätsgrad für SQL Server 2008 (100).

Erweitern Sie die Anweisung wie im nachfolgenden Beispiel, wird die Beispieldatenbank *Marketing* analog zum letzten Abschnitt erzeugt.

```
CREATE DATABASE Marketing ON PRIMARY
(   NAME = 'marketing_data',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\ma_data.mdf',
    SIZE = 30720KB , MAXSIZE = 512000KB , FILEGROWTH = 5120KB)
LOG ON
(   NAME = 'marketing_log',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\ma_log.ldf',
    SIZE = 1024KB , FILEGROWTH = 10%);
```

Für jede der zwei Datenbankdateien werden hier der logische Name (`NAME`), der physische Dateiname (`FILENAME`) sowie die Faktoren für die automatische Vergrößerung angegeben. `ON PRIMARY` gibt die Dateigruppe für die primäre Datendatei an. Hinter `LOG ON` wird das Transaktionsprotokoll angegeben.

Soll eine Datenbank mit mehreren Dateigruppen und mehreren Datendateien angelegt werden, ist die Anweisung so zu erweitern, wie im nachfolgenden Beispiel gezeigt:

```
CREATE DATABASE Konzern ON PRIMARY
(   NAME = 'db_data1',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\db_data1.mdf',
    SIZE = 512000KB , FILEGROWTH = 10%),
(   NAME = N'db_data2',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\db_data2.ndf',
    SIZE = 409600KB , FILEGROWTH = 10%),
FILEGROUP DATEN
(   NAME = 'db_data3',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\db_data3.ndf',
    SIZE = 409600KB , FILEGROWTH = 10%),
FILEGROUP [INDEX]
(   NAME = 'db_index',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\db_index.ndf',
    SIZE = 204800KB , FILEGROWTH = 10%)
LOG ON
(   NAME = 'db_log',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\db_log.ldf',
    SIZE = 153600KB , FILEGROWTH = 10%);
```

Da bei diesem Beispiel zwei Datendateien in der Dateigruppe *PRIMARY* angelegt werden, werden diese hintereinander angegeben. Datendateien, die in einer eigenen Dateigruppe angelegt werden, erhalten den Namen der Dateigruppe mit dem Schlüsselwort `FILEGROUP` vorangestellt.

Da `INDEX` ein reservierter Begriff in SQL ist, muss es als Name wie im Beispiel in eckige Klammern gesetzt werden. Sonst würde die Anweisung beim Ausführen einen Fehler erzeugen.

3.1.4 Datenbank mit Filestream ausstatten

NEU 2008

Ein besonderes neues Feature von SQL Server 2008 besteht darin, Binärdaten im Dateisystem ablegen zu können, und dem Anwender das so erscheinen zu lassen, als wären diese Dateien noch immer in der Datenbank gespeichert.

Bisher konnten Sie zwischen zwei Methoden auswählen, um binäre Daten wie Dokumente, Kalkulationstabellen, Videos und Ähnliches über eine Datenbankanwendung zur Verfügung zu stellen.

1. Sie speichern Binärobjekte (BLOBs = Binary Large Objects) in der Datenbank und verwenden dafür Tabellenspalten mit dem Datentyp *varbinary(max)*. Der Vorteil besteht darin, dass die Objekte durch die Speicherung in der Datenbank mit dieser mit gesichert und gemeinsam mit den Daten verändert werden, und daher immer konsistent mit diesen sind. Änderungen an Binärobjekten sind ebenfalls Bestandteile von Transaktionen. Dem stehen aber folgende Nachteile gegenüber:
   - Die Datenbankgröße nimmt enorm zu.
   - Die Größe der gespeicherten Objekte ist auf jeweils 2 GB beschränkt.
   - Der Zugriff auch größere Objekte ist über das Dateisystem schneller als über die Datenbank.
2. Als zweite Variante werden Objekte als Dateien im Dateisystem abgelegt; in der Datenbank werden die Pfade bzw. Links zu diesen Dateien als *Character*-Werte gespeichert. Damit können die bei der Speicherung in der Datenbank auftretenden Nachteile umgangen werden. Jedoch treten hierbei andere Nachteile auf, nämlich jene Punkte, die bei der Speicherung in der Datenbank als Vorteile anzusehen sind:
   - Daten und Dateien werden separat angelegt, gespeichert, verwaltet und gesichert. Es gibt keinen Mechanismus, der sicherstellt, dass in der Datenbank abgelegte Links nicht ins Leere gehen.
   - Für den Zugriff auf die Daten in der Datenbank und die verlinkten Dateien gelten unterschiedliche Berechtigungssysteme. Wie kann man also sicherstellen, dass jeder, der Zugriff auf die Daten hat, auch die dazugehörigen Dateien lesen kann?

Der SQL Server 2008 bietet über Filestream die Möglichkeit, die Vorteile beider Welten gemeinsam zu nutzen: Speicherung der Objekte im Dateisystem und dennoch volle Integration in die Datenbank. Dies bedeutet: Auch wenn Objekte direkt auf einem Speichermedium gespeichert werden, stehen sie innerhalb der Datenbank so zur Verfügung, als wären sie in der Datenbank abgelegt. Auch bei der Sicherung der Datenbank werden diese Daten mit gesichert.

Ich möchte Ihnen nun zeigen, welche Vorkehrungen Sie in der Datenbank treffen müssen, um diese Möglichkeiten nutzen zu können.

Zunächst muss auf Serverebene Filestream aktiviert sein. Wenn dies noch nicht beim Setup geschehen ist, öffnen Sie bitte auf dem Server den Konfigurations-Manager und die Eigenschaften des betroffenen Server-Dienstes. Auf dem Register FILESTREAM können Sie denselben für diese Instanz aktivieren.

Soll der Zugriff über .NET-Programmierung direkt über die API auf Filestream-Daten erfolgen können, so aktivieren Sie zusätzlich die Option FILESTREAM FÜR E/A-STREAMINGZUGRIFF AUF DATEIEN AKTIVIEREN und legen einen Freigabenamen fest. Dann können Sie bei Ihrer Anwendungsentwicklung nicht nur via Transact-SQL, sondern auch auf Dateisystemebene direkt auf Dateien zugreifen. Dies bringt insbesondere bei großen Dateien über 100 MB Vorteile bei der Leistung.

Soll der direkte Streamingzugriff auch von anderen Rechnern im Netzwerk aus möglich sein, müssen Sie auch die letzte Option in diesem Dialog aktivieren. Andernfalls funktioniert dies lediglich lokal.

Beim erstmaligen Aktivieren von Filestream kann es erforderlich sein, den Rechner neu zu starten, da neue Komponenten installiert werden müssen. Auf jeden Fall sollten Sie den SQL Server-Dienst neu starten.

Ist der Server für den Filestream aktiviert, ist nun die Datenbank an der Reihe. Davor müssen wir nur noch auf dem Server einen Basisordner für die Aufnahme der Dateien anlegen. Ich habe bei mir auf dem Server *Softelier03sql* dafür auf der Festplatte *D* den Ordner *DB_FS* erstellt.

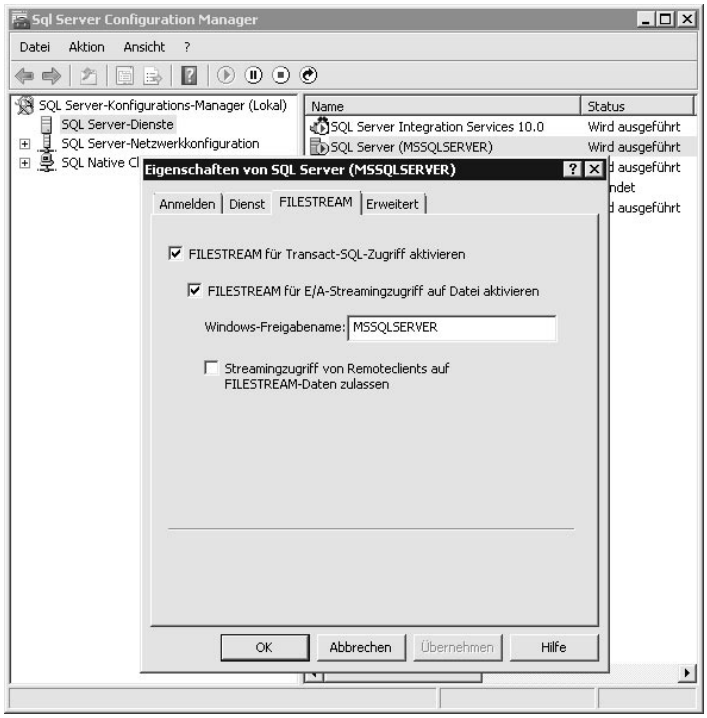


Abbildung 3.13: *Server für Filestream konfigurieren*

Wir erstellen über das Management Studio eine neue Datenbank mit dem Namen *Video*. Wie zuvor legen wir wieder eine zusätzliche Dateigruppe an, diesmal aber für Filestream. Dazu klicken Sie im Dialog NEUE DATENBANK im unteren Teil der Seite DATEIGRUPPEN auf HINZUFÜGEN und geben der Dateigruppe beispielsweise den Namen *VIDEOFILES*.

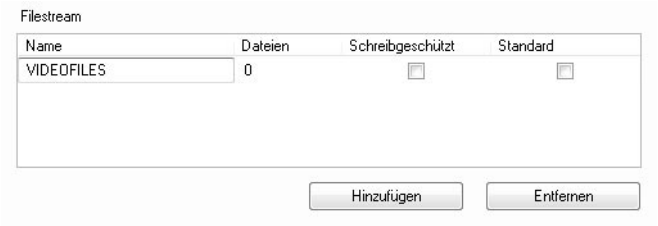


Abbildung 3.14: *Dateigruppe für Filestream anlegen*

Nun muss noch eine neue Datei in dieser Dateigruppe angelegt werden. Dies geschieht wieder auf der Seite ALLGEMEIN des Dialogs zum Anlegen einer neuen Datenbank. Diese bekommt so wie Daten- und Log-Dateien einen logischen Namen, da sie intern wie eine solche behandelt wird. Ich gebe ihr hier den Namen *Video_Stream*. Als Dateityp wird diesmal *Filestream-Daten* ausgewählt. Als Dateigruppe wird sofort die zuvor angelegte Gruppe vorgeschlagen. Haben Sie in Ihrem Beispiel mehrere Dateigruppen für Filestream angelegt, können Sie natürlich zwischen diesen auswählen. *Anfangsgröße* und *Automatische Vergrößerung* entfallen bei diesem Dateityp. Alledings muss der Pfad angegeben werden. Ich verwende den zuvor angelegten Ordner. So wie bei anderen Datenbankdateien auch muss der angegebene Pfad bei der Erstellung bereits bestehen.

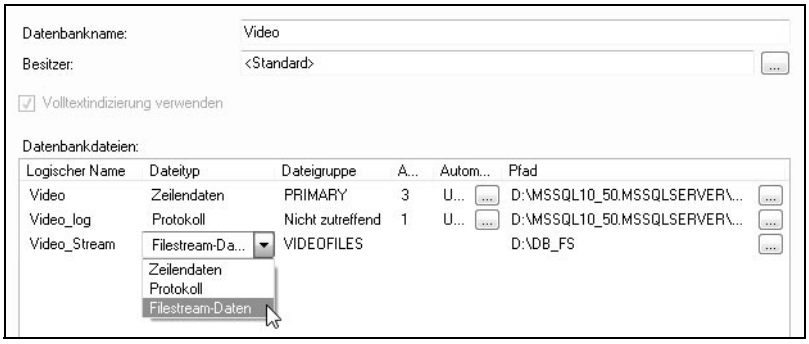


Abbildung 3.15: *Datenbankdatei für Filestream anlegen*

Wenn die Datenbank angelegt wird, wird in dem Basisordner ein weiterer Ordner mit dem Namen der Dateigruppe angelegt. Wenn wir die Datenbank nicht mit dem grafischen Tool, sondern mit einer Anweisung in einem Abfrageeditor-Fenster erstellen, können wir auch diesen Ordnernamen frei vergeben. Lediglich das grafische Tool verwendet den logischen Namen der Datenbankdatei.

Wird die Datenbank gelöscht, verschwindet dieser Ordner mit all seinem Inhalt.

Die nachfolgende Anweisung entspricht den Eingaben, die wir im Dialog NEUE DATENBANK vorgenommen haben, und kann stattdessen gleichbedeutend verwendet werden.

```
CREATE DATABASE Video ON PRIMARY
(   NAME = 'Video',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Video.mdf',
    SIZE = 3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB ),
FILEGROUP VIDEOFILES CONTAINS FILESTREAM
(   NAME = 'Video_Stream',
    FILENAME = 'D:\DB_FS\Video_Stream')
LOG ON
(   NAME = 'Video_log',
    FILENAME = 'D:\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Video_log.ldf',
    SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%);
```

Der Anweisungsteil zur Definition einer neuen Dateigruppe wird um den Zusatz `CONTAINS FILESTREAM` ergänzt. Ansonsten gleicht er großenteils den bereist zuvor verwendeten Anweisungen.

Wie Sie Filestream beim Anlegen von Tabellen nutzen können, lesen Sie am Ende des Kapitels.

3.2 Tabellen in der Datenbank erstellen

Da eine Datenbank erst mit Tabellen zu einer solchen wird, werden wir nun unsere Datenbank mit Leben füllen. Im folgenden Abschnitt möchte ich Ihnen zeigen, wie Sie eine Tabelle anlegen, wie Sie diese indizieren und mit Gültigkeitsregeln versehen. Ein besonderes Augenmerk werde ich auf das Erstellen von Beziehungen legen.

Für die Arbeit mit diesem Kapitel ist es von Vorteil, wenn Sie mit den Grundzügen der Theorie relationaler Datenbanken vertraut sind. Wenn Sie zum Beispiel bereits ein wenig Erfahrung mit der Arbeit mit MS Access haben, ist dies ausreichend.

Wir möchten in der im vorigen Abschnitt erstellten Marketing-Datenbank folgende Tabellen anlegen:

- *Kunden*
- *Interessen*
- *Kundeninteressen*

In diesem Beispiel werden neben den Kunden (*tblKunden*) deren Interessen (*tblInteressen*) in der Datenbank gespeichert. Die m:n-Beziehung zwischen diesen beiden Tabellen wird über die Zwischentabelle *tblKundenInteressen* aufgelöst.

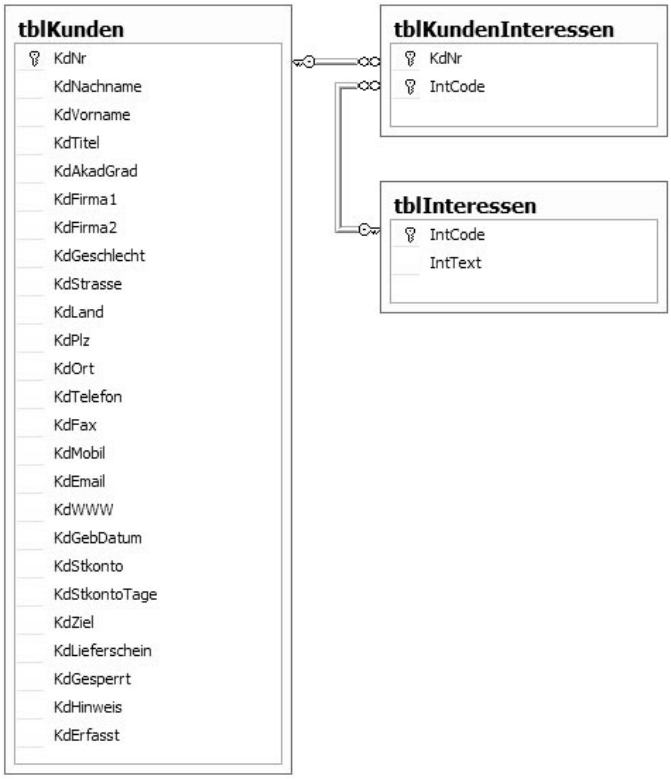


Abbildung 3.16: *Anzulegende Beispieltabellen*

3.2.1 Tabellenfelder definieren

Eine Tabelle besteht aus einzelnen *Feldern*. Andere Ausdrücke dafür sind auch *Datenfelder* oder *Spalten*. Auch Tabellenspalten und Tabellenfelder sind gebräuchliche Ausdrücke. Da im Tabellen-Designer des Management Studios der Begriff *Spalte* verwendet wird, verwende ich zumeist auch diesen Ausdruck.

Um eine neue Tabelle anzulegen, erweitern Sie die Ordnerstruktur der neuen Datenbank bis zu den Tabellen. Entweder über den Ordner *Tabellen* oder das Register DETAILS ZUM OBJEKT-EXPLORER wählen Sie im Kontextmenü den Befehl NEUE TABELLE... aus.

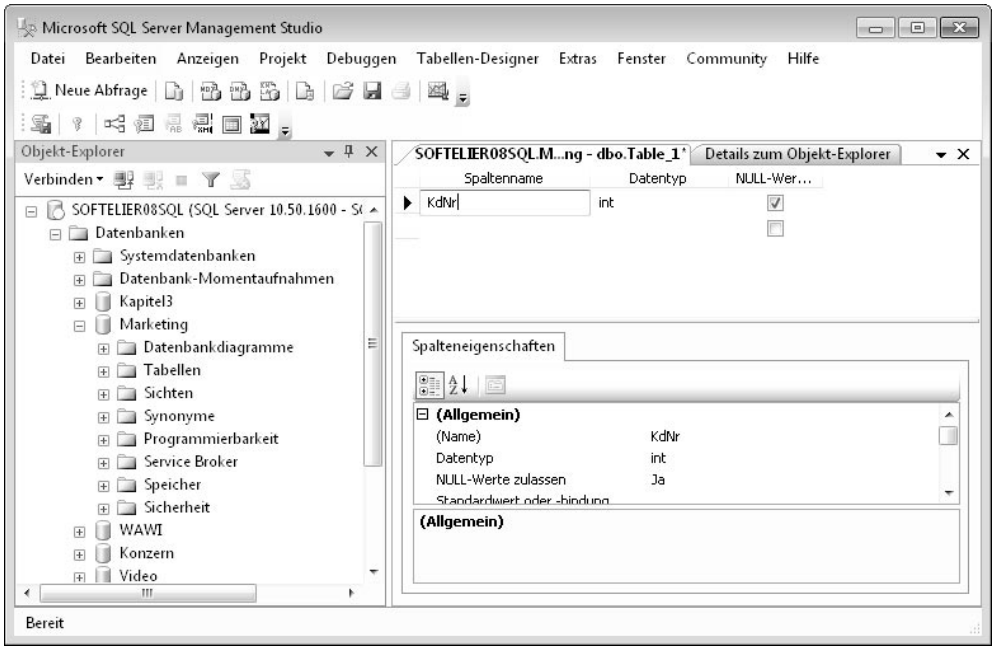


Abbildung 3.17: *Neue Tabelle anlegen*

Im Raster können Sie nun die Spalten der Tabelle eintragen. Für die Vergabe der Spaltennamen sollten Sie folgende Regeln beachten:

- Vermeiden Sie Leer- und Sonderzeichen. Der Unterstrich ist erlaubt, er gilt nicht als Sonderzeichen. Feldnamen mit Sonderzeichen müssten nämlich immer in eckigen Klammern oder unter doppelten Hochkommata (sogenannte *quoted identifiers* in ANSI-SQL) geschrieben werden. Dies würde mit der Zeit sehr lästig werden.
- Feldnamen sollten sprechend sein, aber auch möglichst kurz und prägnant. Die maximale Länge beträgt 128 Zeichen.
- Legen Sie sich ein Namensschema zurecht, nach dem Sie die Namen vergeben. Sie merken sich diese dann leichter, wenn Sie sie im Zweifel anhand Ihrer Namenslogik herleiten. Wir verwenden beispielsweise Präfixe, die den Namen der Tabelle widerspiegeln.

Welches Benennungsschema Sie für Ihre Tabellen- und Spaltennamen verwenden, ist nicht so wichtig. Viel wichtiger ist, dass Sie überhaupt ein Benennungsschema verwenden – und in einem Team alle dasselbe!

Neu in ab Version 2008 ist, dass Sie bestimmte Benennungsregeln über Richtlinien erzwingen können. Dazu später mehr in diesem Kapitel.

Für die Definition der Spalten stellt der SQL Server folgende Datentypen zur Verfügung:

Kategorie	Datentyp	Beschreibung
Character	char (Länge) varchar (Länge) nchar (Länge) nvarchar (Länge) varchar (max) nvarchar (max)	Text mit fixer oder variabler (var) Länge. Als maximale Länge können 8.000 Zeichen definiert werden. Typen mit dem Präfix *n* (für national) verwenden Unicode und belegen den doppelten Speicherplatz. Die maximale Länge beträgt daher 4.000 Zeichen. Typen mit dem fixen Parameter *max* sind seit der Version 2005 verfügbar. Sie können maximal 2 GB an Daten aufnehmen und vereinen die Vorteile von normalem Character und sogenannten CLOBs (Charakter Large Objects) – Text beim SQL Server genannt.
Datum/ Uhrzeit	datetime smalldatetime	Datums- und Zeitangaben. *datetime* reicht von 01.01.1753 bis 31.12.9999 auf 3,33 Millisekunden genau und belegt acht Byte, *smalldatetime* kommt mit vier Byte aus, reicht dafür aber mit Minutengenauigkeit nur von 01.01.1900 bis 06.06.2079.
Zahlen	decimal (Genauigkeit, Dezimalstellen) vardecimal (nur Enterprise Edition mit SP2) numeric (Genauigkeit, Dezimalstellen)	*decimal* und *numeric* sind derselbe Datentyp. Ihre Größe wird durch die Genauigkeit in Stellen (maximal 38) und die darin enthaltenen Dezimalstellen angegeben.
	float real	*real* und *float* repräsentieren Gleitkommazahlen. Da nicht alle Werte im Bereich exakt dargestellt werden können, eignen sich diese Datentypen nicht für Primärschlüssel.
	bigint int smallint tinyint	Die Integer-Typen repräsentieren ganze Zahlen. *bigint* hat einen Wertebereich von -2^{63} bis 2^{63} und benötigt dafür acht Byte. *int* kann mit vier Byte Platzbedarf einen Bereich von –2.147.483.648 bis 2.147.483.647 abdecken. Für den Bereich von –32.768 bis –32.767 kommen Sie mit *smallint* und zwei Byte je Zahl aus. *tinyint* benötigt ein Byte für die Abdeckung des Bereichs von 0 bis 255.
Währung	money smallmoney	Währungen speichern Daten auf vier Nachkommastellen genau. Für kleinere Beträge können Sie *smallmoney* mit einem Wertebereich von –214.748.3648 bis 214.748.3647 verwenden. Dafür werden vier Byte benötigt. Ist dieser Wertebereich zu gering, müssen Sie *money* mit einem Speicherbedarf von acht Byte verwenden. Dafür können Sie einen Bereich von circa –922 Billionen bis 922 Billionen abbilden.
Boolean	bit	Dieser Datentyp kann die Werte *Wahr* (1), *Falsch* (0) und *NULL* darstellen.

Tabelle 3.2: *SQL Server-Datentypen*

Kategorie	Datentyp	Beschreibung
Text und Image	text image	BLOBs (Binary Large Objects) und CLOBs zur Speicherung von Daten bis zu 2 GB Größe. *text* sollte beim SQL Server ab der Version 2005 durch den im Einsatz flexibleren Datentyp *varchar(max)* ersetzt werden.
Binär	binary(Länge) varbinary(Länge) varbinary(max)	Datentypen zur Speicherung von Binärdaten mit maximal 8.000 Byte. Der Datentyp *varbinary(max)* kann bis zu 2^{31}–1 Byte Daten aufnehmen.
XML	xml	In diesem Datentyp können XML-Daten bis zu einer Größe von maximal 2 GB gespeichert werden.
Variant	sql_variant	Mit diesem Datentyp können unterschiedliche Daten wie zum Beispiel *varchar* oder *int* gespeichert werden. Er passt sich dem Inhalt an. Allerdings muss dieser Datentyp vor der Verwendung in einem Ausdruck explizit in einen passenden anderen Datentyp konvertiert werden. Zum Beispiel in einen *int*, um in einer mathematischen Berechnung verwendet werden zu können.

Tabelle 3.2: *SQL Server-Datentypen (Forts.)*

Der SQL Server 2008 wartet erneut mit einigen neuen Datentypen auf. Um diese von den bisher schon verfügbaren Typen besser abzuheben, führe ich sie in einer separaten Übersicht auf.

Die neuen Datentypen des SQL Server 2008 – in R2 sind keine weiteren neuen Datentypen ergänzt worden – sind:

Datentyp	Beschreibung
date	Der Datentyp *date* bietet die Möglichkeit, einen reinen Datumswert ohne eine Uhrzeitkomponente zu verwenden. Er sollte also ab Version 2008 in all jenen Fällen verwendet werden, bei denen schon bisher nur das Datum benötigt wurde und die Uhrzeit nur Ballast war. Er belegt lediglich drei Byte – also weniger als der bisher für diese Anwendungsfälle verwendete Datentyp *smalldatetime* – und hat einen wesentlich weiteren Geltungsbereich: von 01.01.0001 bis 31.12.9999

Tabelle 3.3: *Neue Datentypen ab SQL Server 2008*

Datentyp	Beschreibung
time(länge)	So wie es mit *date* einen neuen Datentyp gibt, der nur ein Datum ohne eine Uhrzeit speichert, so gibt es auch den neuen Datentyp *time*, der eben nur eine Zeit ohne ein Datum enthält. Über den Längenparameter wird die Genauigkeit in Sekundenbruchteilen festgelegt. Dabei entspricht 0 ganzen Sekunden im Format *hh:mm:ss* und 7 der maximalen Genauigkeit von 100 Nanosekunden (1 Nanosekunde = 10^{-9} Sekunden) im Format *hh:mm:ss.nnnnnnn*. Der belegte Speicherplatz ist vom Längenparameter abhängig. Bei 0-2 werden drei Byte belegt, bei 3-4 sind es vier Byte und darüber hinaus werden fünf Byte benötigt. Wird der Parameter bei der Definition nicht angegeben, entspricht dies einer Länge von sieben. Mit dieser Genauigkeit ist der SQL Server nun viel besser als bisher für wissenschaftliche Anwendungen geeignet.
datetime2(länge)	Der Datentyp *datetime2* ist eine Erweiterung des schon bekannten Datentyps *datetime*. Der Datumsbereich entspricht dem von *date*, der Zeitbereich jenem von *time* – so als würde man einfach *date* und *time* zusammenfügen. Der Längenparameter entspricht eins zu eins jenem von *time*. Auch der Speicherplatzbedarf ergibt sich aus der Addition der beiden. Je nach Genauigkeit der Zeit ergibt sich daher ein Bedarf zwischen sechs und acht Byte.
datetimeoffset(länge)	Der vierte im Bunde der neuen Datentypen im Bereich Datum und Uhrzeit ist *datetimeoffset*. Dieser entspricht dem um die Zeitzone ergänzten Datentyp *datetime2*. Er wird im Format *YYYY-MM-DD hh:mm:ss.nnnnnnn* {+|-}*hh:mm* dargestellt.
geography	Die Datentyp *geography* dient zum Speichern von geographischen Daten der Erdkugel.
geometry	*Geometry* ist in der Lage, Geometriedaten zu speichern und kann ähnlich wie *geography* verwendet werden. Hierbei wird aber immer von der Ebenenform, und nicht von der Ellipsenform der Erde ausgegangen.
hierarchyid	Um Positionen innerhalb einer Hierarchie zu beschreiben, kann der neue Datentyp *hierarchyid* verwendet werden. Der Aufbau der Hierarchie muss allerdings über die Anwendung definiert werden.

Tabelle 3.3: *Neue Datentypen ab SQL Server 2008 (Forts.)*

Wenn Sie die Spaltennamen eintragen und den Datentyp auswählen, definieren Sie direkt in der dritten Spalte, ob dieses Feld `NULL` zulassen soll oder nicht. Spalten, die `NULL` nicht zulassen, müssen einen Eintrag erhalten; sie dürfen also nicht leer sein.

Legen Sie die Felder gemäß der nachfolgenden Grafik an.

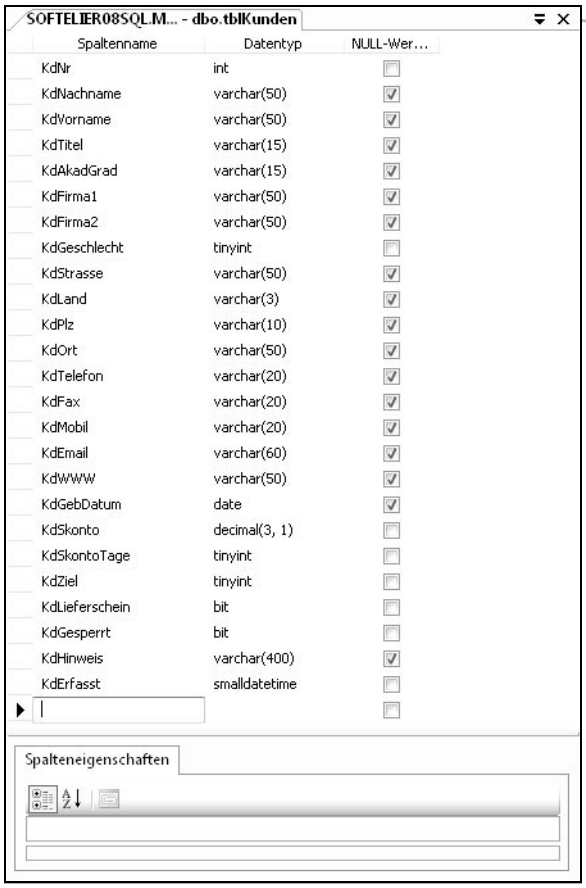


Abbildung 3.18: *Kundentabelle*

Speichern Sie die Tabelle unter dem Namen *tblKunden* ab, auch wenn sie noch nicht ganz fertig ist. Verwenden Sie dazu das Diskettensymbol in der Symbolleiste.

3.2.2 Spalteneigenschaften

In Abhängigkeit vom gewählten Felddatentyp können Sie weitere Eigenschaften für einzelne Spalten der Tabelle festlegen. Da das SQL Server Management Studio das grafische User-Interface (Shell) von Visual Studio verwendet, können auch hier die Eigenschaften entweder nach Kategorien oder alphabetisch sortiert angezeigt werden. Die nachfolgende Abbildung zeigt die Darstellung nach Kategorien. Über die Symbole links oben auf dem Register kann die Darstellung angepasst werden. Eigenschaften, die mehrere Einstellungen erfordern, sind standardmäßig eingeklappt und können über das Pluszeichen ausgeklappt werden. Auch die einzelnen Kategorien können über das Plus- und das Minuszeichen eingeklappt bzw. expandiert werden.

Abbildung 3.19: *Spalteneigenschaften nach Kategorie*

Wenn Sie möchten, können Sie den Datentyp auch in den Spalteneigenschaften eingeben. Dies kann vor allen bei Datentypen wie *character*, bei denen eine Feldgröße angegeben werden muss, ein wenig komfortabler sein.

Folgende Eigenschaftseinstellungen können Sie für Spalten vornehmen:

- **(Name):** Diese Eigenschaft entspricht dem Spaltennamen, der bereits in der Übersichtsdarstellung erfasst worden ist.
- **Datentyp:** Auch hierbei handelt es sich um eine alternative Eingabemöglichkeit für den Datentyp. Dieser wird an dieser Stelle aber ohne den Längen-Parameter ausgewählt, da dieser über eine eigene Eigenschaft eingetragen werden kann bzw. muss.
- **Länge:** Hier wird die Länge zum gewählten Datentyp eingetragen, falls ein solche erforderlich ist.
- **NULL-Werte zulassen:** Auch hierbei handelt es sich um eine alternative Eingabemöglichkeit für das Kontrollkästen in der Übersichtsdarstellung.
- **Beschreibung:** Dieser Erläuterungstext dient Ihrer Dokumentation. Hier können Sie einen beliebigen Text eintragen. So könnten Sie beispielsweise für das Geschlecht die verwendeten Kürzel vermerken, z. B.: `1=Frau; 2=Herr; 3=Famile; 4=Firma; 5=Sonstiges.` (Mit *Sonstiges* sind zum Beispiel Vereine, öffentliche Einrichtungen und Ähnliches gemeint.)

- **Standardwert oder -bindung:** Diese Eigenschaft wird in der Praxis sehr oft verwendet. Hier können Sie Werte für eine Spalte definieren, mit denen die Spalte bei der Neuerfassung eines Datensatzes bereits vorbelegt wird. In unserer Kundentabelle könnte zum Beispiel das Länderkürzel (*KdLand*) mit `'D'` vorbelegt werden. Um das Erfassungsdatum (*KdErfasst*) mit dem aktuellen Datum vorzubelegen, verwenden Sie die Funktion `GETDATE()` als Standardwert.
- **ComputedColumnSpecification:** SQL Server unterstützt berechnete Spalten in Tabellen. Über die Eingabe einer Formel werden hier die berechneten Werte direkt in der Tabelle angezeigt. Beispielsweise könnte der Kundentyp in Abhängigkeit vom Feld *KdGeschlecht* angezeigt werden. Für die Werte 1 bis 3 soll *privat*, für die anderen Werte *Firma* angezeigt werden. Dazu müsste die Formel `CASE WHEN KdGeschlecht <= 3 THEN 'privat' ELSE 'Firma' END` in der Eigenschaft *Formel* eingetragen werden. Der Vorteil dieser berechneten Spalten gegenüber an anderen Stellen berechneten Werten ist, dass für sie ein Index erstellt werden kann.
- **Identitätsspezifikation:** Diese Eigenschaft ist nur für Zahlenspalten verfügbar. Pro Tabelle kann eine Spalte als sogenannte *Identität* (Identity) definiert werden. Für diese werden zusätzlich ein Startwert (*ID-Ausgangswert*) und eine Schrittweite (*ID-Inkrement*) definiert. Standardmäßig sind beide mit dem Wert 1 vorbelegt. Beim Einfügen von Datensätzen wird eine Identitätsspalte ausgehend vom Startwert automatisch befüllt. In eine als Identität festgelegte Spalte können ohne besondere Vorkehrungen manuell keine Werte eingetragen werden. Verwenden Sie diese Eigenschaft immer, falls Sie eine fortlaufende Nummerierung benötigen, bei der verursachte Lücken durch das Löschen von Datensätzen und zurückgerollte Transaktionen kein Problem darstellen. In der Regel wird diese Eigenschaft für Primärschlüsselspalten verwendet.
- **Sortierung:** Diese Eigenschaft ist nur für Spalten mit *character*-Datentypen verfügbar. So wie beim Anlegen der Datenbank eine Sortierreihenfolge festgelegt worden ist, kann diese auch für jede einzelne Spalte mit einem *character*-Datentyp festgelegt werden. In der Praxis wird man hier jedoch die Standardeinstellung übernehmen. Es ist nur in Ausnahmefällen sinnvoll, für einzelne Spalten innerhalb einer Datenbank unterschiedliche Sortierreihenfolgen (Collationen) einzustellen.

Sie können zum Beispiel über eine andere Sortierung für eine Kennwortspalte festlegen, dass in dieser – im Unterschied zu allen anderen Spalten – bei Vergleichen sehr wohl zwischen Groß- und Kleinbuchstaben unterschieden wird. Dazu müssen Sie lediglich dieselbe Sortierung wie jene des Datenbankstandards, jedoch mit *CS* (= context sensitive) anstelle von *CI* (= context insensitive) in der Zeichenfolge verwenden. (Zum Beispiel `Latin1_General_CS_AS` anstelle von `Latin1_General_CI_AS`.)

- **Volltextspezifikation:** Ist die Volltextindizierung für die Datenbank aktiviert, kann über diese Eigenschaft festgelegt werden, ob diese Spalte volltextindiziert werden soll.

3.2.3 Constraints

Um Geschäftsregeln in Tabellen zu erzwingen, werden Einschränkungen (Constraints) benötigt. Diese sind zwar eigenständige Objekte mit einem eigenen Namen, sind aber fest mit einer Tabelle verbunden. Wird diese Tabelle gelöscht, werden alle Einschränkungen ebenfalls mitgelöscht. Der SQL Server kennt folgende Einschränkungstypen:

- Primary Key (Primärschlüssel)
- Unique Key (eindeutiger Schlüssel)
- Foreign Key (Fremdschlüssel)
- Check (Gültigkeitsregel)
- Default (Standardwert)

Im grafischen Tool gibt es keine einheitliche Oberfläche für die Erstellung von Einschränkungen. Jeder der fünf Typen wird an einer anderen Stelle erzeugt. Standardwerte werden zum Beispiel wie zuvor beschrieben über Spalteneigenschaften angelegt. Der Name für eine Standardwert-Einschränkung wird vom Management Studio automatisch vergeben. Es gibt in der grafischen Umgebungt keine Möglichkeit, einen solchen Namen einzugeben. Dass Constraints wirklich eigene Objekte sind, ist in der grafischen Oberfläche durch die Integration der Erstellung kaum erkennbar. Deutlicher wird dies erst beim Erstellen von Tabellen mittels SQL-Anweisungen.

Primärschlüssel

Pro Tabelle kann es nur einen Primärschlüssel geben, der allerdings auch aus mehreren Spalten bestehen darf. Ein Primärschlüssel weist folgende Merkmale auf:

- Er darf nicht `NULL` sein.
- Er muss eindeutig sein.
- Es wird automatisch ein Index erstellt.
- Er wird für Beziehungen benötigt.

Um eine oder mehrere Spalten als Primärschlüssel zu definieren, markieren Sie die betroffene(n) Spalte(n) und wählen im Kontextmenü den Befehl PRIMÄRSCHLÜSSEL FESTLEGEN.

Abbildung 3.20: *Einen Primärschlüssel festlegen*

Auch hier wird der Name für die Primärschlüssel-Einschränkung vom Management Studio automatisch vergeben. Es wird hierfür der Name der Tabelle mit dem Präfix *PK_* verwendet. Der Primärschlüssel wird durch ein Schlüsselsymbol optisch hervorgehoben.

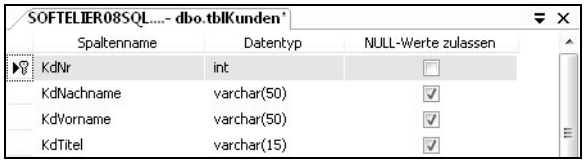


Abbildung 3.21: *Primärschlüssel*

Natürlich können Sie die markierte(n) Spalte(n) auch über das Symbol mit dem Schlüssel in der Symbolleiste als Primärschlüssel für diese Tabelle festlegen.

Gültigkeitsregeln

Mit Check-Einschränkungen werden Gültigkeitsregeln definiert, die auf Datensatzebene wirken. Diese Regeln müssen durch einen Ausdruck abzubilden sein und sich auf den Datensatz beschränken. Das heißt, Sie können bei der Prüfung einer Check-Einschränkung nur auf die Werte innerhalb des Datensatzes zugreifen. Sie können dabei nicht auf andere Datensätze der Tabelle oder gar auf die Inhalte anderer Tabellen zugreifen. Folgende Regeln lassen sich beispielsweise mit einer Check-Einschränkung prüfen:

- In einem Feld dürfen nur Werte von ... bis ... erfasst werden.
- Der Wert in einem Feld muss größer oder kleiner als der eines anderen Feldes sein.
- Der eingegebene Wert muss einer Eingabemaske entsprechen. Dies könnte zum Beispiel für die Prüfung einer E-Mail-Adresse verwendet werden.
- In mindestens einer von zwei definierten Spalten muss ein Eintrag vorgenommen werden.

Nicht direkt realisierbar – außer über den Einsatz von benutzerdefinierten Funktionen – sind Aufgabenstellungen wie die nachfolgenden:

- Das Format der Postleitzahl muss dem im Länderkürzel eingetragenen Land entsprechen. (Die Formate für alle Länder sind in einer anderen Tabelle gespeichert.)
- Die Reservierung einer Ressource überschneidet sich mit einer anderen Reservierung in derselben Tabelle.

Wir möchten in der Kundentabelle folgende Gültigkeitsregeln implementieren.

Regel	Ausdruck

Das Geschlecht darf die Werte 1 bis 5 enthalten.	`KdGeschlecht BETWEEN 1 AND 5`
	oder zum Beispiel:
	`KdGeschlecht > 0 AND KdGeschlecht < 6`
	oder zum Beispiel:
	`KdGeschlecht IN(1, 2, 3, 4, 5)`

Tabelle 3.4: *Check-Einschränkungen*

Regel	Ausdruck

Der Skonto darf nicht negativ sein und nicht über 5% liegen.	`KdSkonto BETWEEN 0 AND 5`
	oder zum Beispiel:
	`KdSkonto >= 0 AND KdSkonto <= 5`

Die Skontotage dürfen nicht negativ sein und maximal 30 Tage betragen.	`KdSkontoTage BEWTEEN 0 AND 30`

Die E-Mail-Adresse muss gültig sein.	`KdEmail LIKE '%__@%__.__' OR`
	`KdEmail LIKE '%__@%__.___' OR`
	`KdEmail LIKE '%__@%__.____'`

Ist im Geschlecht *Herr*/*Frau*/*Familie* ausgewählt, müssen Nachname und Vorname erfasst werden. Ist *Firma*/*Sonstiges* eingetragen, muss die Spalte *Firma* ebenfalls ausgefüllt sein.	`(KdGeschlecht <= 3 AND KdNachname IS NOT NULL AND KdVorname IS NOT NULL)` `OR (KdGeschlecht >= 4 AND KdFirma1 IS NOT NULL)`

Tabelle 3.4: *Check-Einschränkungen (Forts.)*

Gehen Sie wie folgt vor, um eine neue Check-Einschränkung zu erstellen:

1. Klicken Sie (irgendwo) im Tabellen-Designer in den Tabellenentwurf und wählen Sie im Kontextmenü den Befehl CHECK-EINSCHRÄNKUNGEN aus.
2. Klicken Sie im Dialog auf HINZUFÜGEN, um eine neue Einschränkung zu erzeugen. Wie die nachfolgende Abbildung zeigt, wird standardmäßig für die neue Einschränkung der Tabellenname mit dem Präfix *CK_* verwendet. Der Stern rechts neben dem Namen zeigt, dass diese Einschränkung noch nicht gespeichert worden ist.

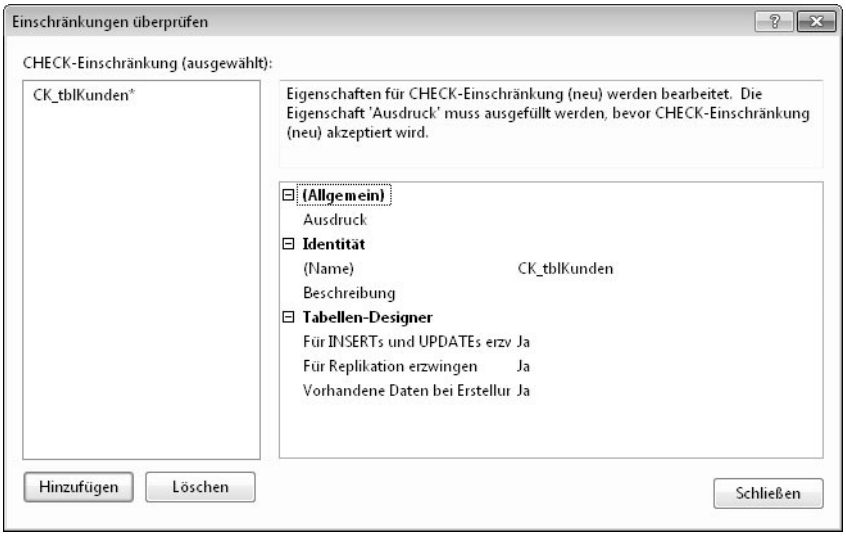


Abbildung 3.22: *Neue CHECK-Einschränkung*

3. Tragen Sie in der ersten Zeile AUSDRUCK den Einschränkungsausdruck `KdGeschlecht BETWEEN 1 AND 5` ein. Da dieses Eingabefeld relativ klein ist, klicken Sie wahlweise auf die Schaltfläche mit den drei Punkten, die am rechten Rand des Eingabefeldes auftaucht. Es öffnet sich ein Dialog mit einem größeren Eingabefeld. Tragen Sie alternativ hier den Einschränkungsausdruck ein.

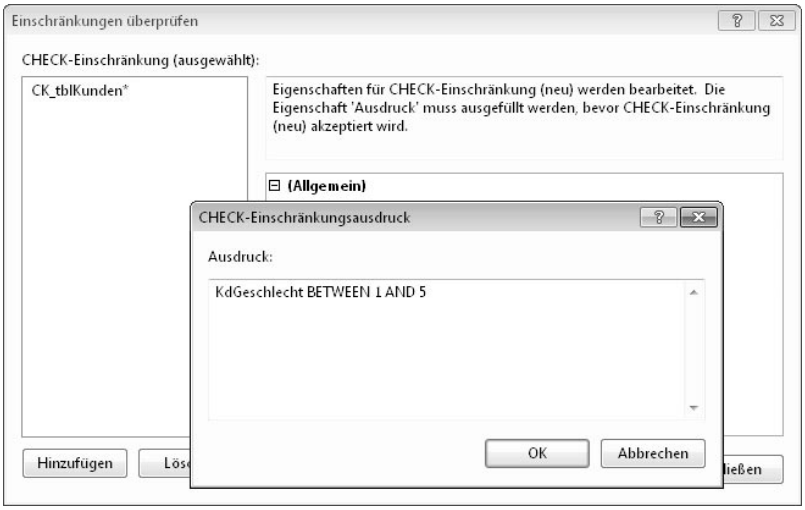


Abbildung 3.23: *Einschränkungsausdruck eintragen*

4. Ergänzen Sie den vom System vorgeschlagenen Namen um den Namen der betroffenen Spalte *KdGeschlecht*. Damit ermöglichen Sie eine saubere Namensgebung, falls Sie mehrere Check-Einschränkungen für eine Tabelle erstellen.

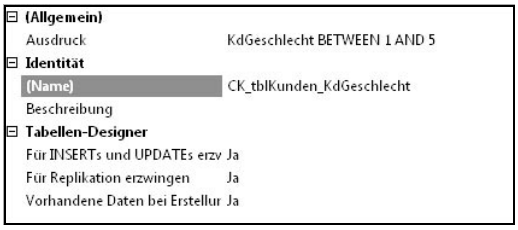


Abbildung 3.24: *Name für einen Constraint festlegen*

Standardmäßig ist die Eigenschaft VORHANDENE DATEN BEI ERSTELLUNG ODER REAKTIVIERUNG ÜBERPRÜFEN aktiviert. Sind in der Tabelle bereits Daten enthalten, die der Regel nicht entsprechen, kann die Einschränkung nicht erstellt werden. In diesem Fall stellen Sie diese Eigenschaft bitte auf *Nein*. Bei Zeiten sollten Sie jedoch diese Daten in Ordnung bringen.

Ergänzen Sie die übrigen in Abbildung 3.21 dargestellten Check-Einschränkungen für die Tabelle *tblKunden*.

Wenn Sie weitere Check-Einschränkungen erstellen und in den Dialog zur Erstellung derselben zurückkehren, wird Ihnen vielleicht auffallen, dass der SQL Server den zuvor von Ihnen eingetragenen Einschränkungsausdruck in die Form `([KdGeschlecht]>=(1) AND [KdGeschlecht]<=(5))` umgeschrieben hat. Dies ist nichts Außergewöhnliches und muss Sie nicht beunruhigen. Auch wenn die eckigen Klammern um die Spaltennamen ergänzt werden, müssen Sie diese nicht selber erfassen, solange die Spaltennamen keine Leer- oder Sonderzeichen enthalten.

Fremdschlüssel

Der Fremdschlüssel ist die technische Umsetzung einer Beziehung zwischen zwei Tabellen. Dabei wird von einer untergeordneten *Detailtabelle* mit einem Fremdschlüssel auf den Primärschlüssel einer übergeordneten *Mastertabelle* referenziert.

Um die Beziehungen, die im Diagramm in Abbildung 3.16 dargestellt sind, erstellen zu können, müssen Sie zunächst noch die beiden Tabellen *tblInteressen* und *tblKundenInteressen* anlegen. Die Tabelle *tblInteressen* enthält den Interessenscode als Primärschlüssel, der aus drei Buchstaben bestehen soll. Die Bezeichnung des Interesses soll in der Spalte *IntText* gespeichert werden.

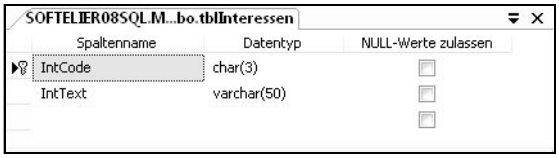


Abbildung 3.25: *Tabelle »tblInteressen«*

Die Tabelle *tblKundenInteressen* dient zur Auflösung der m:n-Beziehung zwischen Kunden und Interessen und ordnet so die Interessen den Kunden zu. Die Spalte *KdNr* dient als Fremdschlüsselspalte für die Beziehung zur Kundentabelle, die Spalte *IntCode* als Fremdschlüsselspalte für die Beziehung zur Interessentabelle. Beide gemeinsam werden als zusammengesetzter Primärschlüssel für diese Tabelle definiert. Da ein Primärschlüssel eindeutig sein muss, ist dadurch ausgeschlossen, dass einem Kunden dasselbe Interesse mehrmals zugeordnet werden kann.

Abbildung 3.26: *Tabelle »tblKundenInteressen«*

Da diese Tabellen auch in unserer Beispieldatenbank *WAWI* enthalten sind, können Sie sich die Inhalte dieser Tabellen für ein besseres Verständnis der Zusammenhänge ansehen. Die nachfolgende Abbildung zeigt zum Beispiel ein paar Einträge der Tabelle *tblKundenInteressen*. Hier sehen Sie, wie in jeder Zeile eine Kundennummer einem Interessenscode zugewiesen ist.

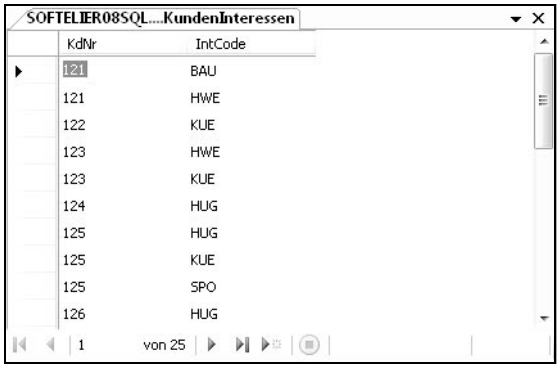


Abbildung 3.27: *Beispieldaten der Tabelle »tblKundenInteressen«*

Sobald Sie diese Tabellen als Voraussetzung für unser Beispiel angelegt haben, können wir uns nun dem eigentlichen Thema, dem Fremdschlüssel, widmen.

Ein Fremdschlüssel weist folgende Eigenschaften auf:

- Er darf nur Werte enthalten, die in der Primärschlüsselspalte der referenzierten Tabelle vorkommen.
- Er darf `NULL` sein. Anders als ein Primarschlüssel muss er nicht zwingend einen Eintrag enthalten. Falls Sie dies möchten – was in der Praxis oft der Fall ist –, müssen Sie die Fremdschlüsselspalte zusätzlich als `NOT NULL` definieren.

Damit ein Fremdschlüssel erstellt werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Die Anzahl und Reihenfolge der Spalten von Fremdschlüssel und referenziertem Primärschlüssel muss identisch sein.
- Primär- und Fremdschlüssel müssen dieselben Datentypen und Feldgrößen haben.
- Die Feldnamen von Primär- und Fremdschlüsselspalten müssen nicht dieselben sein. Jedoch ist es für Datenbankneulinge zu Beginn einfacher und übersichtlicher, wenn dies der Fall ist.

Fremdschlüssel erstellen Sie können einen Fremdschlüssel beziehungsweise eine Beziehung im Management Studio auf zwei Arten erstellen:

- Im *Entwurf der Fremdschlüsseltabelle (Tabellen-Designer)*. Diese Variante werden wir uns als erste ansehen.
- Über ein *Datenbankdiagramm*. Diese Möglichkeit ist wegen der sehr guten grafischen Aufbereitung und wegen der Erstellung der Beziehung per Drag & Drop sehr intuitiv. Sie lernen diese Variante später in diesem Kapitel kennen.

Da ein Fremdschüssel immer zur Detailtabelle gehört, müssen Sie diesen in unserem Beispiel für die Tabelle *tblKundenInteressen* anlegen. Wenn Sie den Tabellenentwurf dieser Tabelle nicht vor sich haben, so wählen Sie bitte die Tabelle im Objekt-Explorer aus und wählen danach im Kontextmenü den Befehl ENTWERFEN aus. Gehen Sie anschließend nach folgender Reihenfolge vor:

1. Wählen Sie über das Kontextmenü im Tabellenentwurf den Befehl BEZIEHUNGEN... aus. Klicken Sie im Dialog FREMDSCHLÜSSELBEZIEHUNGEN auf die Schaltfläche HINZUFÜGEN. Wie schon bei der Erstellung einer Check-Einschränkung wird ein neuer Eintrag mit Standardeinstellungen erzeugt, den Sie nun noch anpassen müssen.

   Klicken Sie dazu in der Zeile TABELLEN- UND SPALTENSPEZIFIKATION auf die Schaltfläche mit den drei Punkten.

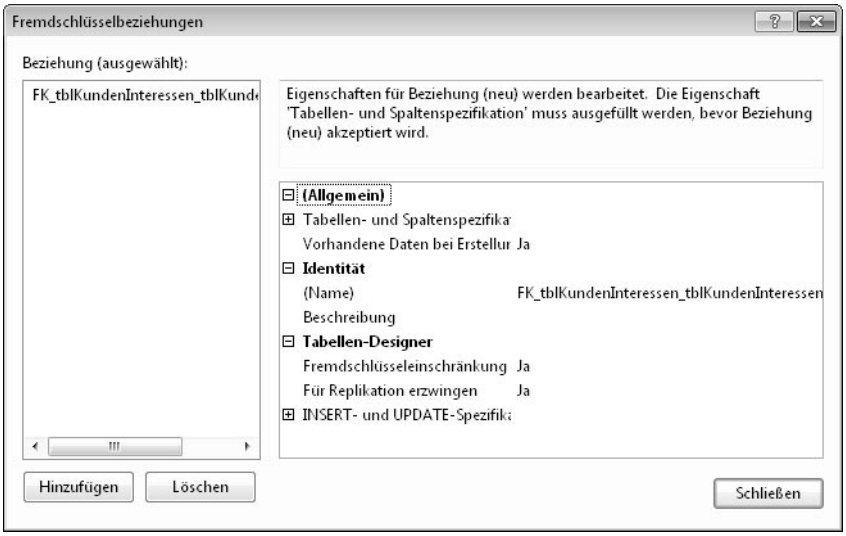


Abbildung 3.28: *Neue Fremdschlüsselbeziehung*

2. Wählen Sie im Dialog TABELLEN UND SPALTENSPEZIFIKATION die Tabelle *tblKunden* als Primärschlüsseltabelle aus. Der Beziehungsname passt sich sofort an diese Änderung an. Sie sollten ihn dann auch so belassen, weil der vorgeschlagene Name den allgemeinen Namenskonventionen für Einschränkungen entspricht. Stellen Sie die Namen für die Beziehung jeweils in der Spalte *KdNr* ein.

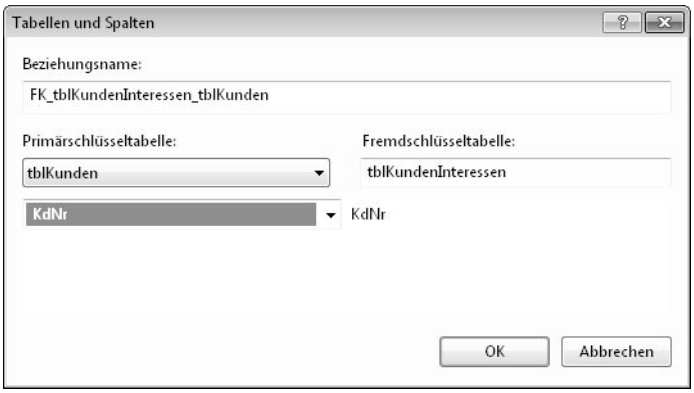


Abbildung 3.29: *Tabellen und Spalten für die Beziehung auswählen*

Bei der Handhabung ist es etwas verwirrend, dass die Spaltennamen in der Auswahlliste nicht, wie man es erwarten würde, nach der Position in der Tabelle, sondern alphabetisch angeordnet sind. Wundern Sie sich also nicht, dass Sie den Spaltennamen *KdNr* erst etwas weiter unten in der Liste vorfinden.

Bestätigen Sie Ihre Eingaben mit OK.

3. Falls Sie möchten, können Sie abschließend noch die Änderungs- oder Löschweitergabe aktivieren. Diese finden Sie in der Rubrik INSERT- UND UPDATE-SPEZIFIKATIONEN unter der Bezeichnung REGEL AKTUALISIEREN beziehungsweise REGEL LÖSCHEN. (Eine Erklärung für diese Einstellungen finden Sie im Anschluss.) Stellen Sie zum Beispiel REGEL LÖSCHEN auf WEITERGABE.

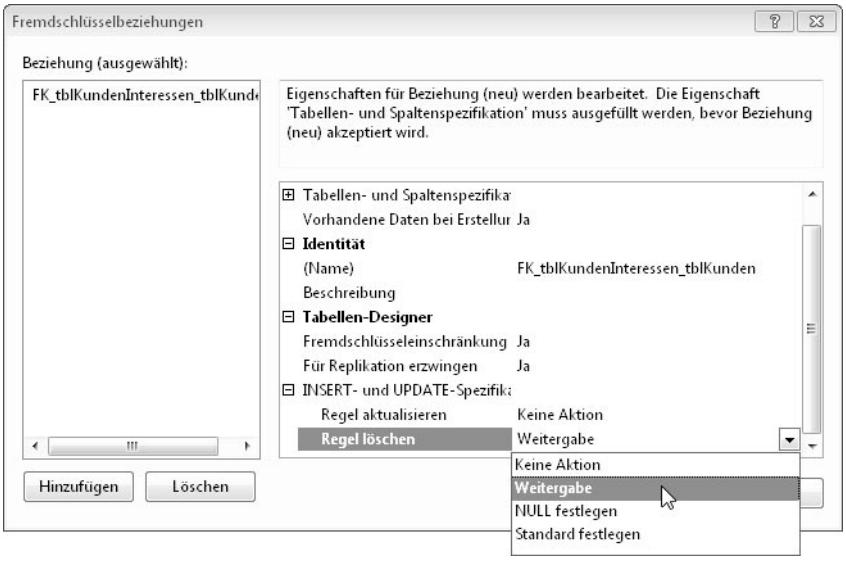


Abbildung 3.30: *Änderungs- oder/und Löschweitergabe festlegen*

Erzeugen Sie noch den zweiten Fremdschlüssel zur Tabelle *tblInteressen* und schließen Sie danach die Eingabe.

Die tatsächliche Erstellung der Fremdschlüsselbeziehungen erfolgt erst, wenn Sie die Änderungen an der Tabelle – zum Beispiel über das Diskettensymbol – speichern.

Referenzielle Integrität Ein Begriff, der auch im Zusammenhang mit der Beziehung zwischen zwei Tabellen steht, ist die *referenzielle Integrität*. Deshalb wird ein Foreign Key-Constraint auch als *Referential Integrity-Constraint* bezeichnet. Die referenzielle Integrität erzwingt, dass für jeden Eintrag in einer Fremdschlüsselspalte ein Eintrag in der Primärschlüsselspalte der referenzierten Tabelle vorhanden ist.

- Die referenzielle Integrität verhindert, dass Datensätze aus der Primärschlüsseltabelle (Mastertabelle) gelöscht werden, wenn dazugehörige Einträge in der Fremdschlüsseltabelle (Detailtabelle) vorhanden sind. Bezogen auf unser Beispiel bedeutet dies, dass Sie keinen Kunden aus der Tabelle *tblKunden* löschen können, falls diesem Kunden in der Tabelle *tblKundenInteressen* Interessensgebiete zugeordnet sind.
- Die referenzielle Integrität verhindert, dass der Inhalt der Primärschlüsselspalte der Mastertabelle geändert wird, falls für diesen Datensatz Einträge in der Detailtabelle existieren. Dies bedeutet in unserem Beispiel, dass Sie keinem Kunden, dem Interessen zugeordnet sind, eine neue Kundennummer geben können.
- Die referenzielle Integrität erzwingt, dass in die Fremdschlüsselspalte der Detailtabelle nur Einträge, die auch in der Mastertabelle vorhanden sind, geschrieben werden können.

Ausnahmen zu diesen Regeln können über die Änderungs- und Löschweitergabe realisiert werden. Diese schaffen eine Art Workaround, wodurch besagte Änderungen zwar möglich sind, aber die aufgestellten Regeln nicht verletzen:

- *Löschweitergabe (Regel löschen)*: Wird ein Datensatz in der Mastertabelle gelöscht, werden alle Detaildatensätze in der Fremdschlüsseltabelle mit gelöscht. In unserem Beispiel würde dies bedeuten, dass, wenn ein Kunde gelöscht wird, auch alle seine Interessenszuordnungen gelöscht werden. Dieses Verhalten wird durch die Wahl des Eintrags *Weitergabe (Cascade)* erreicht. In dieser Version ist zum Glück eine saubere Übersetzung ins Deutsche erfolgt. In der Vorgängerversion wurde der englische Begriff »Cascade« etwas unglücklich wörtlich mit »überlappend« übersetzt.

  Neben *Weitergabe* gibt es noch die beiden folgenden Varianten zur Auswahl:

  - *NULL festlegen (Set NULL)*: Wird ein Eintrag in der Mastertabelle gelöscht, werden die Detaildatensätze zwar nicht gelöscht, aber der Inhalt der Fremdschlüsselspalte geleert. Dies ist allerdings nur möglich, wenn diese auch `NULL`-Werte zulässt.
  - *Standard festlegen (Set Default)*: Diese Einstellung bewirkt ein ähnliches Verhalten wie die vorige Option. Der Unterschied besteht nur darin, dass der Inhalt der Spalte nicht geleert, sondern auf den definierten Standardwert zurückgesetzt wird. Dies ist allerdings nur möglich, wenn es überhaupt einen Standardwert für diese Spalte gibt und dieser referenziert werden kann. Das heißt, dieser muss als Eintrag in der referernzierten Tabelle vorhanden sein.

- *Änderungsweitergabe (Regel ändern)*: Die Änderungsweitergabe bewirkt bei der Auswahl von *Weitergabe*, dass bei einer Änderung im Masterdatensatz diese im Detaildatensatz mitgezogen wird. Ändern wir in unserem Beispiel die Kundennummer eines Kunden, wird die Kundennummer bei den Interessenszuordnungen ebenfalls geändert. Dadurch gehören auch nach einer Änderung noch dieselben Datensätze zusammen.

  Wie beim Löschen stehen auch hier die Optionen *NULL festlegen* und *Standard festlegen* zur Auswahl. Ihre Bedeutung ist analog.

Seien Sie mit der Aktivierung der Löschweitergabe sehr vorsichtig. Diese wird in der Praxis nur in sehr wenigen Fällen eingesetzt, da sie zu einem unkontrollierten Löschen von Daten führen kann.

Eindeutiger Schlüssel

Den eindeutigen Schlüssel habe ich mir bis zum Schluss aufgehoben, da er wie ein Index anzulegen ist.

Ein eindeutiger Schlüssel (Unique Key) unterscheidet sich von einem Primärschlüssel durch die folgenden beiden Punkte:

- `NULL`-Werte sind in einem eindeutigen Schlüssel zugelassen. Allerdings im Gegensatz zu anderen Datenbanksystemen nur ein einziges Mal. Dies schränkt seine Verwendbarkeit stark ein.
- Es kann in jeder Tabelle mehrere eindeutige Schlüssel geben.

Eindeutige Schlüssel werden dann verwendet, wenn Sie in einer Spalte zwar eindeutige Werte haben möchten, diese aber nicht als Primärschlüssel definieren möchten oder können.

Benötigen Sie zusätzlich zum Primärschlüssel Spalten, die eindeutig sind, so verwenden Sie ebenfalls einen eindeutigen Schlüssel, da ja pro Tabelle nur ein Primärschlüssel erstellt werden kann. Wie Sie einen eindeutigen Schlüssel erstellen, lesen Sie im folgenden Abschnitt.

3.2.4 Indizierung

Wenn Sie in einer Tabelle einen oder mehrere Werte suchen, muss die Datenbank-Engine alle in dieser Spalte gespeicherten Werte lesen, um die Treffer zu ermitteln. Man nennt dies im Fachjargon einen *Full Table Scan*.

Wenn Sie in diesem Buch nach einer ganz bestimmten Information suchen, werden Sie deshalb nicht das ganze Buch von der ersten bis zur letzten Seite durchsuchen, um diese Information zu finden. Vielmehr werden Sie den Index am Ende des Buches durchschauen. Dies hat für Sie zwei Vorteile:

- Sie finden den gesuchten Begriff aufgrund der Sortierung schneller.
- Durch die Angabe der Seitenzahl(en) können Sie sofort den gesuchten Text finden, ohne alle Seiten durchschauen zu müssen.

Nach demselben Grundprinzip werden Indizes in Datenbanken verwendet. Um Suchvorgänge zu beschleunigen, können gezielt einzelne Spalten einer Tabelle indiziert werden.

Sie sollten beachten, dass der SQL Server aufgrund der Statistiken, die er über jeden Index führt, vor dem Ausführen der Abfrage selber entscheidet, ob er den Index verwendet. Die Tatsache, dass Sie ihm einen Index für den Suchvorgang zur Verfügung stellen, bedeutet noch lange nicht, dass der Server diesen auch verwendet. (Ich habe mich bei der Erstellung des Index für dieses Buch auch sehr bemüht, habe jedoch keinen Einfluss darauf, ob Sie ihn nutzen oder nicht.)

Nach folgenden Kriterien sollten Sie eine zu indizierende Spalte auswählen:

- Die Spalte wird häufig als Suchkriterium oder als Verknüpfungskriterium in Abfragen verwendet.
- Die erwartete Trefferquote bei einer Suche in der indizierten Spalte ist sehr gering. Bei einer erwarteten großen Trefferanzahl ist ein Full Table Scan schneller als eine Suche über den Index. (Wenn Sie in diesem Buch viele Informationen benötigen, werden Sie diese auch nicht der Reihe nach im Index suchen und dann x-mal nach vorne blättern, sondern das Buch von vorne nach hinten durchgehen und jene Seiten, die Sie nicht interessieren, überspringen.)
- Die Spalte enthält viele unterschiedliche Werte. Je eindeutiger die Werte in einer Spalte sind, umso effizienter ist ein Index. Ein Index für eine Spalte mit fast gleichen Werten bringt nicht viel. (Sie würden wahrscheinlich auch nicht auf die Idee kommen, den Begriff »SQL Server« in diesem Buch über einen Index zu suchen. Da dieser Begriff enorm oft vorkommt, wäre das alles andere als effizient.)
- Es handelt sich nicht um sehr kleine Tabellen. (Sie würden auch nicht auf die Idee kommen, für eine vierseitige Broschüre einen Index zu erstellen, sondern dies nur für umfangreiche Bücher tun.)
- Ziehen Sie zusammengesetzte Indizes in Betracht. In einem zusammengesetzten Index – das ist ein Index, der über mehrere Spalten erstellt worden ist – können Sie nach führenden Spalten suchen. Haben Sie zum Beispiel einen zusammengesetzten Index für die Spalten *Nachname*, *Vorname* und *Postleitzahl* erstellt, kann dieser verwendet werden, wenn Sie nach allen drei Spalten, nach dem Nachnamen und dem Vornamen oder dem Nachnamen alleine suchen.

Da ein Index nicht nur Vorteile, sondern auch Kosten verursacht, wäre es absolut falsch, alle oder einen Großteil der Spalten indizieren zu wollen. In diesem Fall würden die Nachteile die gewonnenen Vorteile mehr als aufwiegen. (Sie kaufen am Samstag auch nicht eine ganze Filiale eines Lebensmittelgeschäftes auf, bloß weil Sie noch nicht wissen, was Sie am Sonntag vielleicht essen werden.)

Die Nachteile eines Index sind:

- Er benötigt Speicherplatz, da alle Werte der indizierten Spalte(n) in dieser(n) nochmals im Index gespeichert werden.
- Bei jeder Datenänderung müssen davon betroffene Indizes aktualisiert werden, was wiederum Systemressourcen in Anspruch nimmt.

Gruppierter und nicht gruppierter Index

Der SQL Server unterstützt zwei Arten von Indizes: gruppierte und nicht gruppierte Indizes. Beide Arten sind nach einer Baumstruktur aufgebaut, die beginnend von einem Wurzelelement je nach Tabellengröße in einer unterschiedlichen Anzahl an Ebenen verzweigt.

- *Nicht gruppierter Index (Nonclustered Index)*: Wird der Index durchsucht und am unteren Ende des Baumes (Blatt-Level; man muss sich diesen Baum auf den Kopf gestellt vorstellen) ein Treffer erzielt, findet man bei diesem die Row-ID des betreffenden Datensatzes. Anhand dieser ID kann dann der Satz gelesen werden.
- *Gruppierter Index (Clustered Index)*: Pro Tabelle kann es nur einen gruppierten Index geben. Dies liegt daran, dass die Daten dieser Tabelle physisch gemäß der Indexreihenfolge gespeichert werden. Am Blatt-Level des Index steht nicht die Row-ID, die angibt, wo der Datensatz zu finden ist, sondern der Datensatz selber. Der Index ist demnach eng mit der Speicherstruktur der Tabelle verwoben. Dieser Index ist daher insgesamt schneller bei der Rückgabe von Werten als ein nicht gruppierter Index.

Wenn Sie einen Primärschlüssel für eine Tabelle erzeugen, wird dieser standardmäßig als gruppierter Index angelegt, sofern es für diese Tabelle noch keinen gruppierten Index gibt.

Genau genommen kennt der SQL Server nicht nur zwei Indexarten, sondern eigentlich vier. Die beiden anderen Indexarten sind aber auf spezielle Datentypen und Anwendungsfälle ausgelegt. Sie können nicht für klassische Datenbankspalten verwendet werden; daher habe ich sie in diese Betrachtung nicht mit einbezogen.

Diese zwei weiteren Indexarten sind:

- **XML-Index:** Dies ist ein spezieller Index für Spalten mit dem Datentyp *xml*, welcher ab der Version 2005 zur Verfügung steht.
- **Räumlicher Index:** Dieser Index ist speziell für Spalten mit einem der neuen Datentypen *geography* oder *geometry* gedacht. Er besteht aus mehreren Ebenen, deren Anzahl festgelegt werden kann, in denen räumliche Daten in immer kleinere Teile zerlegt werden.

Erstellen eines Index

Einen Index haben wir indirekt bereits erzeugt: Jedes Mal, wenn wir einen Primärschlüssel anlegen, wird automatisch auch ein Index für diese Spalte angelegt. Da ein Index wie eine Einschränkung ein eigenes Objekt darstellt, besitzt er auch einen Namen. Der Name des für den Primärschlüssel angelegten Index entspricht jenem für den Primärschlüssel.

Um einen Index anzulegen, müssen Sie wie schon zuvor beim Erstellen einer Check-Einschränkung und einer Fremdschlüsselbeziehung die Tabelle im Entwurf geöffnet haben. Gehen Sie danach wie folgt vor, um einen Index für die Spalte *KdNachname* der Tabelle *tblKunden* zu erzeugen:

1. Wählen Sie über das Kontextmenü im Tabellenentwurf den Befehl INDIZES/SCHLÜSSEL... aus. Im Dialog sehen Sie den für den Primärschlüssel bereits erstellen Index (*PK_tblKunden*). Fügen Sie einen neuen Index hinzu. Dieser wird, wie bereits aus den vorigen Beispielen bekannt, mit Standardwerten angezeigt.

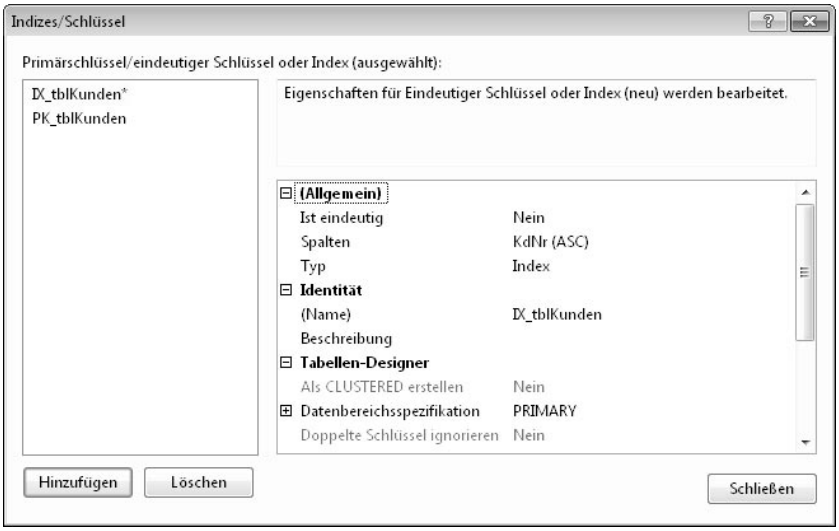


Abbildung 3.31: *Neuen Index anlegen*

2. Klicken Sie in die Zeile SPALTEN und danach auf die Schaltfläche mit den drei Punkten, um die Spalte für den Index auszuwählen. Wählen Sie in der Liste SPALTENNAME den Eintrag *KdNachname* aus. Belassen Sie die Sortierreihenfolge auf *Aufsteigend* (ASC = ascending).

Abbildung 3.32: *Indexspalte(n) auswählen*

Um an anderer Stelle einen mehrspaltigen Index zu erzeugen, wählen Sie dort einfach alle Spalten in der gewünschten Reihenfolge aus. Zu Ihrer Erinnerung: Die Reihenfolge ist von Bedeutung, da ein mehrspaltiger Index nicht nur bei einer Suche nach allen enthalten Spalten, sondern auch bei einer Suche nach führenden Kopfspalten verwendet werden kann.

3. Um aus dem Index einen eindeutigen Schlüssel zu machen, wie im vorigen Abschnitt erläutert, stellen Sie den Typ auf *Eindeutiger Schlüssel*.

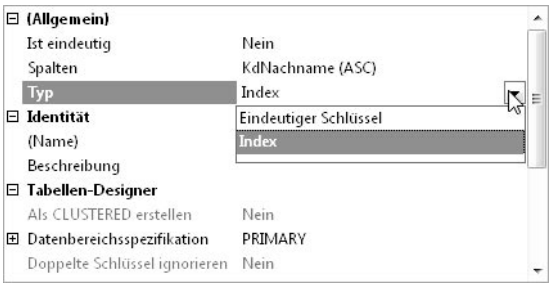


Abbildung 3.33: *Index oder eindeutiger Schlüssel*

In diesem Beispiel belassen wir die Einstellung *Index*.

4. Bauen Sie den Namen der indizierten Spalte in den Indexnamen ein. Das typischerweise dabei verwendete Namensgebungsschema gleicht dem von Constraints.

Abbildung 3.34: *Indexname*

5. Da der gruppierte Index in dieser Tabelle für den Primärschlüssel bereits vergeben ist, ist die Einstellung *Als CLUSTERED erstellen* nicht mehr aktiv.

   In der Datenbereichsspezifikation können Sie die Dateigruppe auswählen, in der der Index angelegt werden soll. Sie werden sich erinnern, dass wir am Beginn dieses Kapitels die Möglichkeit besprochen haben, einen Index auch in einer anderen Dateigruppe zu platzieren.

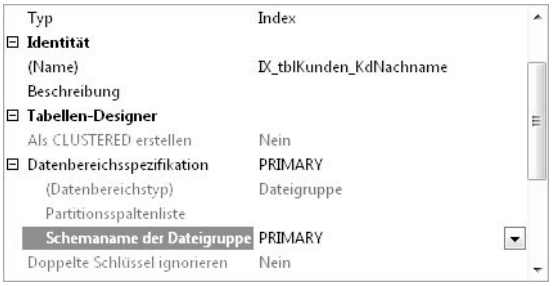


Abbildung 3.35: *Dateigruppe für Index*

Erstellen Sie noch weitere Indizes für die Spalten *KdFirma1* und *KdPlz*.

Die Indizes werden angelegt, sobald Sie die vorgenommenen Änderungen im Tabellen-Designer speichern.

Sie können die in den vorigen Abschnitten verwendeten Dialoge nicht nur über das Kontextmenü, sondern auch über die Symbolleiste *Tabellen-Designer* öffnen.

Abbildung 3.36: *Symbolleiste »Tabellen-Designer«*

Die Symbole in dieser Symbolleiste von links nach rechts:

- Änderungsskript generieren
- Primärschlüssel festlegen
- Beziehungen
- Indizes und Schlüssel verwalten
- Volltextindex verwalten
- XML-Indizes verwalten
- CHECK-Einschränkungen verwalten
- Räumliche Indizes verwalten

3.2.5 Erste Daten erfassen

Nachdem wir nun unsere ersten Tabellen angelegt haben, möchten wir nun ein paar Testdaten in diesen Tabellen erfassen. Auch wenn das Management Studio kein Tool zur Datenerfassung ist, besteht diese Möglichkeit dennoch, um wie jetzt »quick and dirty« in Tabellen zu schreiben. Um Daten zu erfassen, öffnen Sie die Tabelle über den Befehl OBERSTE 200 ZEILEN BEARBEITEN des Kontextmenüs.

NEU 2008

Der Befehl OBERSTE 200 ZEILEN BEARBEITEN ersetzt den Befehl TABELLE ÖFFNEN der Version 2005. Damit wird beim Öffnen die Anzeige der Daten auf die ersten 200 der von der Datenbank gelesenen Datensätze beschränkt. Zuerst skeptisch, empfinde ich diese Änderung inzwischen als absoluten Vorteil. Denn bei kleinen Tabellen stört diese Beschränkung nicht, und bei großen Tabellen erspart man sich das manuelle Abbrechen beim Laden der Daten. Sehr oft habe ich eine Tabelle geöffnet und dann das Laden der Datensätze manuell abgebrochen, um dann erst über die Eingabe von Kriterien jene Daten zu selektieren, die ich wirklich bearbeiten wollte. Dieses Abbrechen spare ich mir nun, und das ist ein Arbeitsschritt weniger. Und in den sehr wenigen Fällen, in denen ich mit diesem Tool tatsächlich alle Datensätze anzeigen möchte, ist es kein Problem, diese Einschränkung manuell zu entfernen.

Erfassen Sie Daten, werden diese mit einem Stift am Zeilenkopf dargestellt, solange sie nicht gespeichert sind. Geänderte Feldinhalte werden mit einem Rufzeichen versehen, bis die Änderungen gespeichert werden. Dies erfolgt in diesem Editor durch das Verlassen der Zeile.

Abbildung 3.37: *Neuen Datensatz erfassen*

Haben Sie schon mehrere Datensätze in der Tabelle, können Sie die Navigationsschaltflächen am unteren Tabellenrand benutzen. Sie können damit zum ersten, vorigen, nächsten oder letzten Datensatz springen. Über das Symbol mit dem Rechtspfeil und dem gelben Stern gelangen Sie zu einem neuen Datensatz. Mit der letzten Schaltfläche können Sie das Laden weiterer Datensätze abbrechen, wenn Sie sehr große Datenmengen hereinladen. Standardmäßig werden in dieser Version beim Öffnen ja ohnehin nur 200 Zeilen geladen.

Abbildung 3.38: *Navigation*

Geben Sie ungültige Daten ein, erhalten Sie eine Fehlermeldung, die auf den *.NET SqlClient Data Provider* verweist und oft Angaben mit .NET-Datentypen enthält. Dies liegt daran, dass das Management Studio über ADO.NET mit dem SQL Server kommuniziert.

Eine Gegenüberstellung der SQL Server-Datentypen mit den .NET-Datentypen finden Sie in einem anderen Zusammenhang in Kapitel 7. Sie hilft Ihnen sicher bei der Interpretation der Fehlermeldungen.

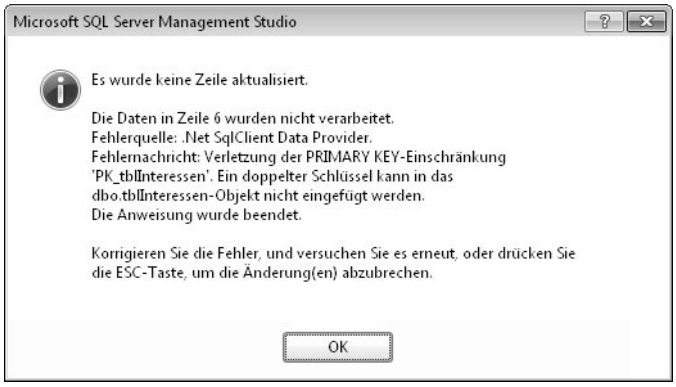


Abbildung 3.39: *Fehlermeldung bei der Eingabe ungültiger Daten*

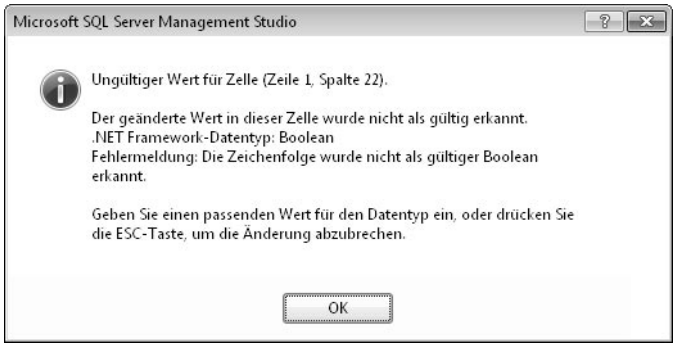


Abbildung 3.40: *Eingabefehler beim Datentyp*

Abbildung 3.40 zeigt eine Fehlermeldung, welche auf einen ungültigen Wert für den Datentyp *Boolean* hinweist. Selbstverständlich gibt es diesen Datentyp beim SQL Server nicht. Dies ist die .NET-Ensprechung für den SQL Server-Datentyp *bit*.

Erfassen Sie bitte Testdaten in den drei Tabellen und achten Sie dabei auf die Regeln, die wir für diese Tabellen festgelegt haben. Verstoßen Sie gegen diese, erhalten Sie eine Fehlermeldung. Bauen Sie zum Test bitte absichtlich solche Fehler ein, um mit dem Umgang mit Fehlermeldungen und deren Interpretation vertraut zu werden. Falsche Eingaben können Sie durch Drücken der Taste Esc verwerfen.

Auch wenn für eine Spalte mit dem Datentyp *bit* die Werte wahr und falsch nach wie vor in den Tabellen mit 1 und 0 gespeichert werden, muss im Grid anstelle von 1 und 0 abweichend davon *True* und *False* eingegeben werden. Sonst erhalten Sie auch hierbei einen Fehler. Wenn Sie einen Feldeintrag löschen möchten, müssen Sie explizit den Ausdruck `NULL` in das Feld eintragen. Außerdem muss `NULL` auch unbedingt in Großbuchstaben eingetippt werden, um erkannt zu werden. Es ist nicht ausreichend, den Inhalt einfach zu entfernen, denn dann würde das Tool versuchen, einen Leerstring zu speichern. Das wiederum erzeugt bei allen Feldern, die keinen *Character*-Datentyp aufweisen, eine Fehlermeldung.

3.3 Datenbankdiagramme einsetzen

Datenbankdiagramme sind ein sehr praktisches Tool sowohl zur Wartung als auch zur Dokumentation des Aufbaus einer Datenbank. Durch die grafische Darstellung der Tabellen und Beziehungen sind sie ein beliebtes Werkzeug zur Anzeige der Struktur der Datenbank.

Datenbankdiagramme, die mit der Version 2000 erstellt worden waren, konnten im Management Studio der Version 2005 nicht angezeigt werden und mussten nach der Übernahme der Datenbank neu erstellt werden. Ab der Version 2008 können Diagramme der Version 2000 jedoch wieder problemlos geöffnet und bearbeitet werden.

Wenn Sie den Ordner *Datenbankdiagramme* für Ihre Datenbank das erste Mal öffnen, werden Sie aufgefordert, die dafür notwendigen Unterstützungsobjekte in dieser Datenbank zu erstellen. Bestätigen Sie die Aufforderung einfach mit JA, um dies zu tun.

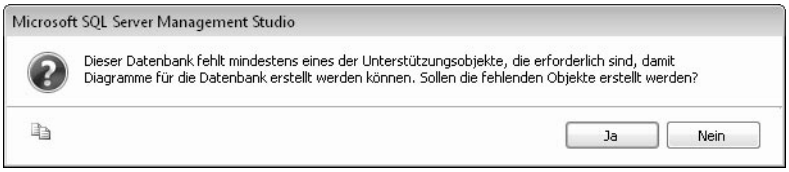


Abbildung 3.41: *Unterstützungsobjekte für Datenbankdiagramme anlegen*

Es tritt dabei jedoch ein Fehler auf, wenn die Datenbank keinen gültigen Eigentümer hat. Dies ist meistens dann der Fall, wenn Sie die Datenbank von einem anderen Server übernommen haben, wie zum Beispiel die *WAWI*-Beispieldatenbank von der Buch-CD. (Der ursprüngliche Besitzer ist ja auf Ihrem Rechner nicht vorhanden.) Da Sie auf Ihrem Rechner in der Regel Administrator sind, können Sie ungehindert mit dieser Datenbank arbeiten – bis eben auf diesen Punkt. Legen Sie daher einen neuen Anmeldenamen an oder verwenden Sie einen bestehenden, um ihn als Besitzer der Datenbank auf der Seite DATEIEN der Datenbankeigenschaften einzutragen. Informationen zum Anlegen eines Anmeldenamens finden Sie in Kapitel 9.

Abbildung 3.42: *Zuweisen des Datenbankbesitzers*

Haben Sie die Unterstützungsobjekte hinzugefügt, können Sie anschließend ein neues Datenbankdiagramm erstellen. Auch bestehende Diagramme einer übernommenen Datenbank können nun angezeigt werden. Klicken Sie auf den Ordner *Datenbankdiagramme* der Datenbank *Marketing* und wählen Sie im Kontextmenü den Befehl NEUES DATENBANKDIAGRAMM aus.

Im Dialog TABELLE HINZUFÜGEN wählen Sie jene Tabellen aus, die Sie im Diagramm darstellen möchten.

Abbildung 3.43: *Tabellen für ein Datenbankdiagramm auswählen*

Bestehende Beziehungen werden im Datenbankdiagramm angezeigt.

Das Datenbankdiagramm bietet eine wesentlich komfortablere Oberfläche zum Erzeugen von Beziehungen als der Tabellen-Designer. Hier können Beziehungen per Drag & Drop erstellt werden. Ziehen Sie dazu einfach die Fremdschlüsselspalte der Detailtabelle mit gedrückter linker Maustaste auf die Primärschlüsselspalte jener Tabelle, zu der Sie die Beziehung erstellen möchten.

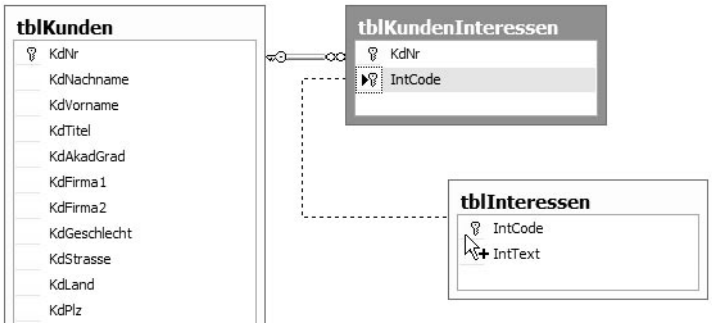


Abbildung 3.44: *Beziehung per Drag & Drop erzeugen*

Nach dem Loslassen der Maustaste öffnet sich derselbe Dialog, den Sie vom Tabellen-Designer her kennen. Wenn Sie beim Ziehen die korrekten Felder treffen, müssen Sie hier keine Einstellungen mehr vornehmen. Wenn Sie möchten, können Sie noch die Änderungs- oder Löschweitergabe aktivieren.

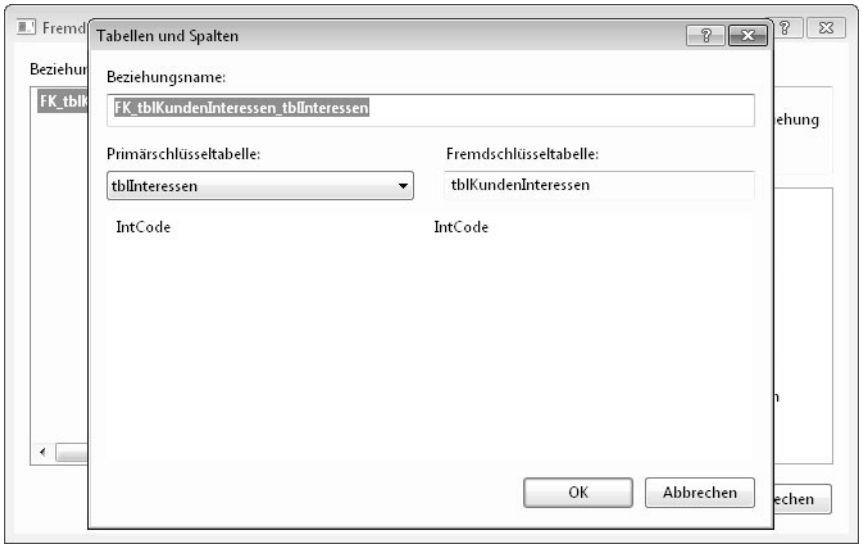


Abbildung 3.45: *Beziehung erstellen*

Um das Diagramm übersichtlich zu gestalten, können Sie:

- die Tabellen anordnen,
- die Beziehungslinien so verschieben, dass eine gute Übersicht erreicht wird,
- über das Kontextmenü neue Textanmerkungen ergänzen und auch formatieren,
- Seitenumbrüche anzeigen, um den Ausdruck vorzubereiten.

Abbildung 3.46 zeigt ein Datenbankdiagramm der Beispieldatenbank *WAWI* mit eingeblendeten Seitenumbrüchen.

Außerdem können Sie direkt in einem Diagramm neue Tabellen anlegen und bestehende bearbeiten. Dazu können Sie über das Kontextmenü die Anzeige der Tabellen so anpassen, wie es für Sie – zum Beispiel zum Ergänzen einer neuen Spalte – am passendsten ist.

Sie können in einer Datenbank nicht nur mehrere Diagramme erstellen, sondern auch Tabellen in mehreren Diagrammen anzeigen. In der Praxis hat es sich bewährt, nicht nur ein Diagramm, in dem alle Zusammenhänge dargestellt werden, zu erstellen. Zusätzlich bieten weitere kleinere Diagramme die Möglichkeit, Detailaspekte übersichtlicher darzustellen. Nehmen Sie Änderungen an Tabellen und Beziehungen in einem Diagramm vor, werden diese in allen anderen Diagrammen automatisch mit übernommen.

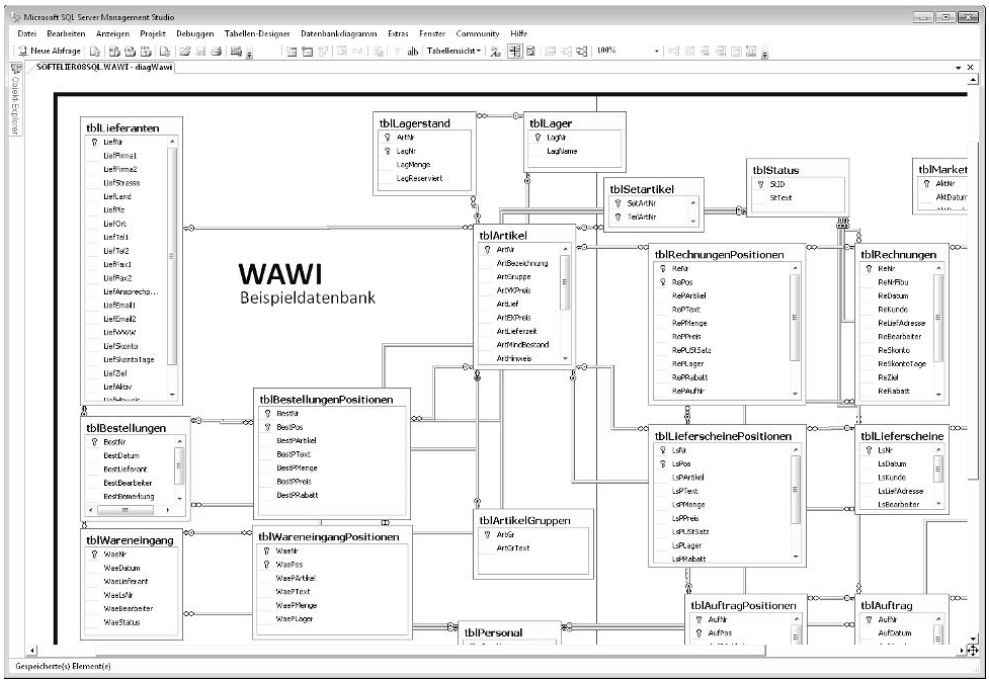


Abbildung 3.46: *Datenbankdiagramm mit Seitenumbrüchen*

3.4 Richtlinien für Benennungsregeln einsetzen

NEU 2008

Ein neues Feature der Version 2008 sind *Richtlinien*, welche auf verschiedenen Ebenen für unterschiedlichste Dinge definiert werden können. In der englischen Version werden Richtlinien als *Policies* bezeichnet.

Da wir ums in diesem Kapitel mit dem Erstellen von Tabellen beschäftigen, möchte ich Ihnen zeigen, wie eine Richtlinie für die Benennung von Tabellen erstellt wird.

Richtlinien sind im Management Studio unter dem Ordner *Verwaltung* zu finden. Folgende Objekte sind hier vorhanden:

- **Richtlinien:** Richtlinien sind die Zusammenfassung verschiedener Regeln, die gemeinsam angewandt und geprüft werden sollen.
- **Bedingungen:** Bedingungen definieren über Ausdrücke diejenigen Kriterien, die geprüft werden sollen. Zum Beispiel, dass der Name eine Mindestlänge von fünf Zeichen aufweist.
- **Facets:** Facets sind diejenigen Objekte, auf die Bedingungen angewandt werden können. Diese sind vordefiniert und werden bei der Erstellung ausgewählt und zugewiesen. Ein Facet ist zum Beispiel eine Datenbank, eine Tabelle oder ein Benutzer. In Abbildung 3.46 sehen Sie einige vordefinierte Facets.

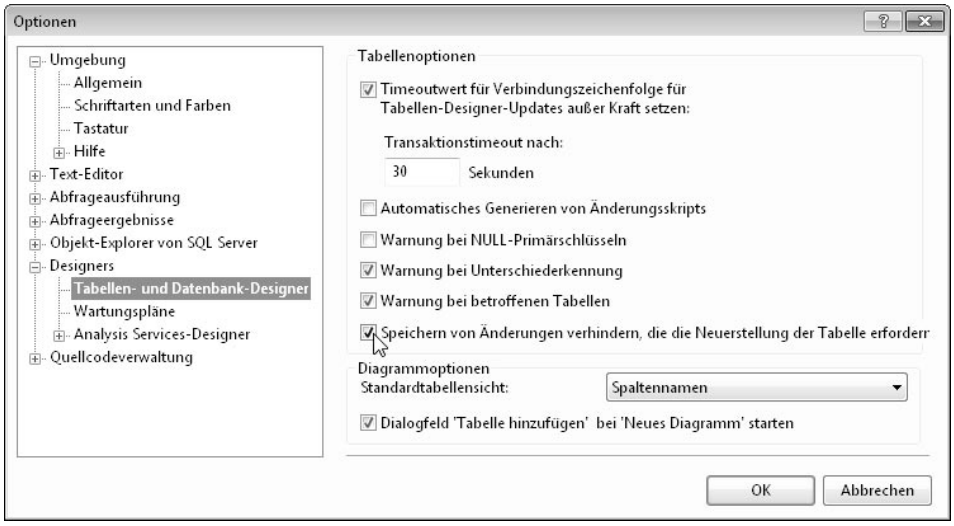


Abbildung 3.47: *Richtlinienverwaltung*

Betrachten wir zur Veranschaulichung des Themas ein kleines Beispiel. Wir wollen eine Benennungsregel für Tabellen in der Beispieldatenbank *WAWI* definieren. Wählen Sie im Kontextmenü dazu den Befehl NEUE RICHTLINIE... aus.

Als Erstes vergeben wir für die neue Richtlinie einen Namen, zum Beispiel *Tabellenname_WAWI*. Die zur Überprüfung angewandte Richtlinie kann nicht nur neu erstellt, sondern auch aus den vorhandenen Bedingungen ausgewählt werden. Damit können Sie verallgemeinerte Bedingungen erstellen, die Sie in mehreren Richtlinien verwenden. Wir haben allerdings noch keine Bedingung und erstellen daher eine neue.

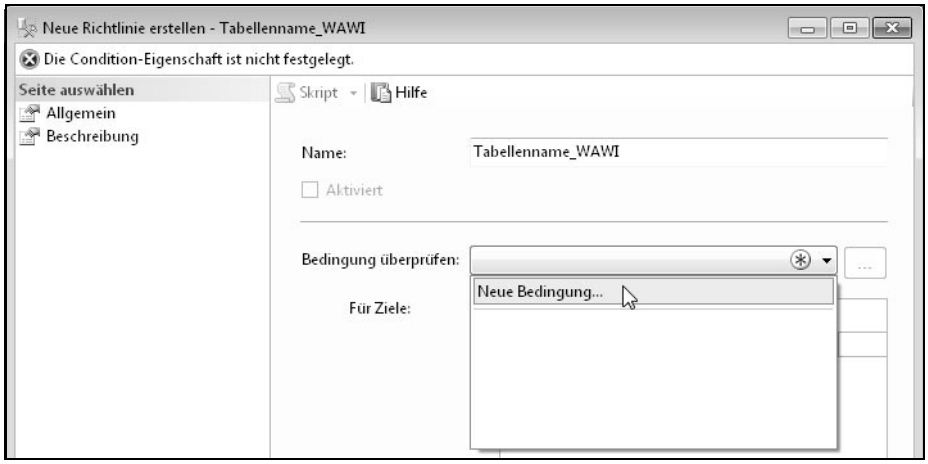


Abbildung 3.48: *Neue Bedingung für neue Richtlinie*

Geben Sie den Namen für die neue Bedingung ein und wählen Sie das Facet als Ziel aus. In diesem Beispiel wählen wir das Facet *Tabelle*.

Nun sind noch die Bedingungsausdrücke zu definieren. Mehrere Ausdrücke sind jeweils mit `AND` oder `OR` zu kombinieren. Die Ausdrücke müssen hier nicht als Ganzes eingetragen werden, sondern werden aus den Komponenten zusammengesetzt:

- **AndOr:** Wird mehr als ein Ausdruck verwendet, muss ab dem zweiten festgelegt werden, ob dieser mit AND oder mit OR verknüpft wird.
- **Feld:** Aus einer umfangreichen Auswahl ist ein betroffenes Feld auszuwählen. Dieses kann alleine verwendet oder wieder in einen Ausdruck eingebaut werden.
- **Operator:** Die klassischen SQL-Operatoren stehen in einer Liste zur Auswahl.
- **Wert:** Der Wert, mit dem verglichen werden soll. Auch hier können nicht nur konstante Ausdrücke, sondern auch Felder eingebaut werden.

Über die Schaltfläche mit den drei Punkten gelangen Sie sowohl beim FELD als auch beim WERT in den Dialog ERWEITERTE BEARBEITUNG. Hier lässt sich der Ausdruck komfortabel eintragen. Funktionen und Eigenschaften können aus einer Liste ausgewählt werden. Zur Unterstützung wird beim Markieren einer solchen ein ausführlicher Erläuterungstext, der sogar eine beispielhafte Verwendung enthält, angezeigt.

Unser Beispiel soll prüfen, ob Tabellennamen mit dem Präfix *tbl* beginnen und eine Mindestlänge von fünf Zeichen aufweisen. Dafür sind folgende Ausdrücke einzutragen:

- `@Name LIKE 'tbl%'`
- `AND LEN(@Name) >= 5`

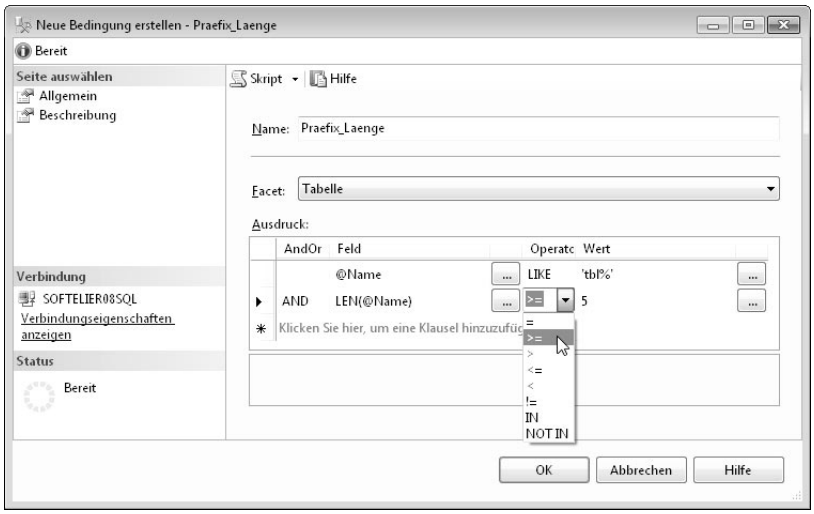


Abbildung 3.49: *Ausdrücke für eine Bedingung definieren*

Nach dem Speichern der Bedingung fehlt noch der Wirkungsbereich. Standardmäßig wird als Ziel für die Anwendung die Richtlinie *Alle Tabellen* in *Allen Datenbanken* vorgeschlagen. Da wir die Richtlinie ausschließlich auf unsere Beispieldatenbank *WAWI* anwenden wollen, klicken wir im Dialog auf das blau unterstrichene *Alle* vor dem Begriff *Datenbanken*. Nun können wir eine weitere Bedingung für die Auswahl der Datenbank anlegen. Ich habe, wie aus Abbildung 3.50 ersichtlich, für diese neue Bedingung den Namen *Datenbank_WAWI* vergeben. Die Bedingung enthält den Ausdruck `@Name = 'wawi'`, der auf das Facet *Datenbank* angewendet wird.

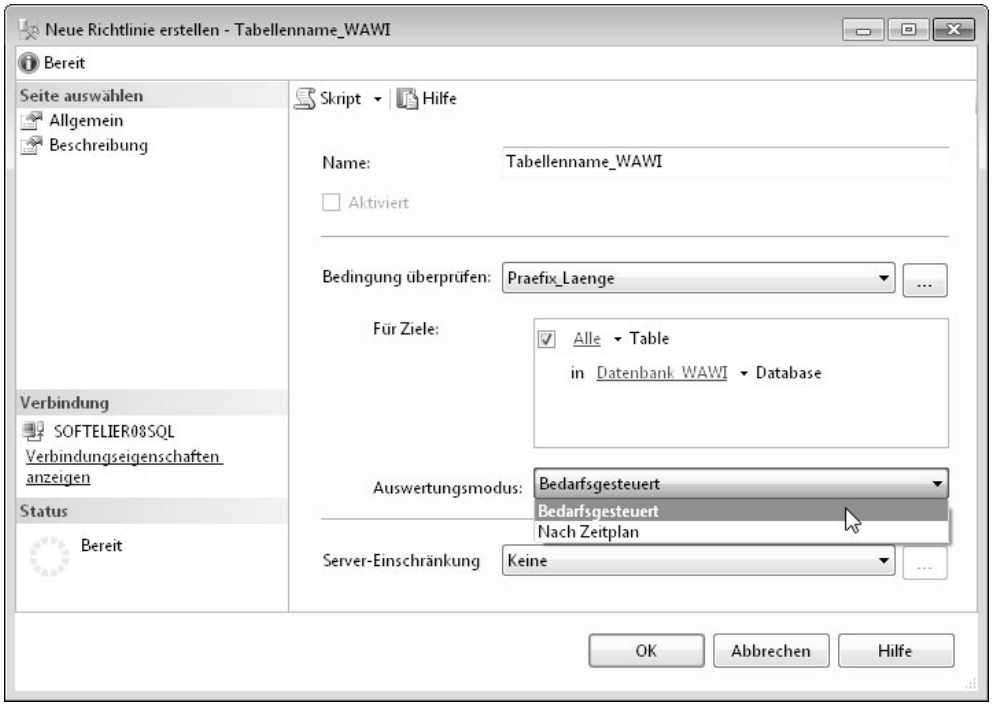


Abbildung 3.50: *Ziele und Auswertungsmodus definieren*

Der letzte Schritt ist für uns die Auswahl des *Auswertungsmodus*. Dieser legt fest, wann die Richtlinien geprüft werden sollen. Hier stehen zwei Varianten zur Auswahl:

- **Bedarfsgesteuert:** Die Prüfung der Richtlinie erfolgt durch manuelles Starten über den Objekt-Explorer über den Befehl AUSWERTEN im Kontextmenü.
- **Nach Zeitplan:** Sie können einen Zeitplan auswählen oder erstellen, zu dem die Auswertung automatisch gestartet werden soll. Hier wird ein Job für den SQL Server Agent eingerichtet. Nähere Informationen zum SQL Server Agent und zu Aufträgen finden Sie in Kapitel 8.

Da der SQL Server Agent in der Express Edition nicht enthalten ist, steht in dieser die Option *Nach Zeitplan* nicht zur Verfügung.

Ich habe für unser Beispiel die Option *Bedarfsgesteuert* gewählt. Nach dem Speichern sind sowohl die neue Richtlinie als auch die beiden in ihr verwendeten Bedingungen im Objekt-Explorer zu sehen. Bedingungen können hier auch losgelöst von Richtlinien erstellt und verändert werden. Gelöscht können sie allerdings nur werden, sofern Sie in keiner Richtlinie mehr verwendet werden.

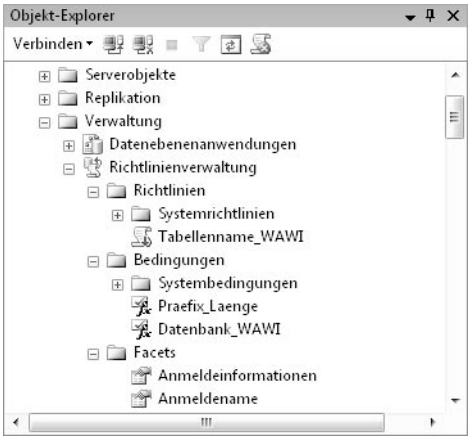


Abbildung 3.51: *Neue Richtlinie und Bedingungen*

Vielleicht fällt Ihnen im Symbol der Richtlinie der rote Abwärtspfeil auf. Dieser zeigt an, dass diese Richtlinie zurzeit deaktiviert ist. Dies ist der Normalzustand für Richtlinien, die keinem Zeitplan zugeordnet sind. Nur Richtlinien mit Zeitplan können über das Kontextmenü aktiviert werden. Dies stört aber weiter nicht, da wir die Richtlinie ja ohnehin manuell auswerten möchten. Dazu setzen wir den gleichnamigen Befehl im Kontextmenü ein.

Werten wir die Richtlinie aus, bekommen wir ein Ergebnis angezeigt. Unser Beispiel liefert das Ergebnis, dass die Richtlinie nicht erfüllt ist. Im unteren Bildschirmbereich werden die Detailergebnisse für alle Tabellen der betroffenen Datenbank *WAWI* aufgelistet. Am unteren Ende dieser Liste finden wir zwei Einträge, für die die Richtlinie nicht erfüllt ist.

Abbildung 3.52: *Ergebnis der Richtlinienprüfung*

Um Details für diese Verstöße anzuzeigen, klicken wir in der betreffenden Zeile auf SICHT... beziehungsweise ANZEIGEN... in R2. Aus der Detailanzeige können wir entnehmen, welche

Bedingung nicht erfüllt ist. Hier sehen Sie, dass es die Tabelle *xGruppenMerge1* gibt, welche der Bedingung widerspricht, dass Tabellennamen mit *tbl* beginnen sollen. Diese temporäre Tabelle benötigen wir für ein Beispiel im nächsten Kapitel. Ebenso wie eine weitere Tabelle, deren Name mit *x* beginnt. Schön in dieser Darstellung ist, dass nicht nur der bloße Verstoß aufgezeigt wird, sondern auch detaillierte Informationen zum erwarteten Wert und zum Istwert einander gegenübergestellt werden.

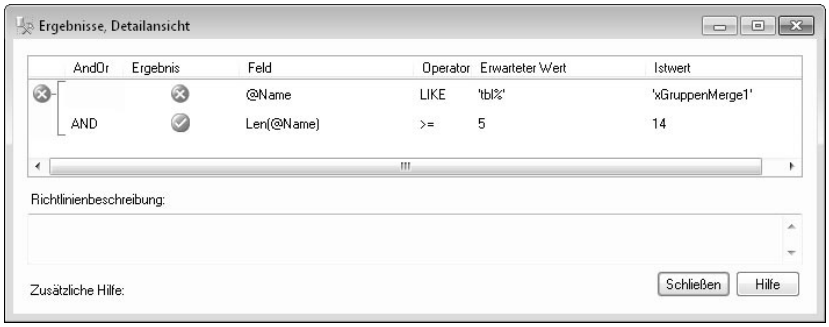


Abbildung 3.53: *Details zu einem Richtlinienverstoß*

Richtlinien dienen also zum Aufzeigen von Verstößen gegen bestimmte Regeln. Aber leider nicht dazu, diese bei der Erstellung von Objekten zu erzwingen.

Richtlinien werden nicht in Benutzerdatenbanken, sondern in der Systemdatenbank *msdb* gespeichert. Um diese auf ein anderes System zu übertragen, können Sie entweder einen Export vornehmen oder ein SQL-Skript für die Richtlinie generieren. Beide Vorgänge führen Sie über das Kontextmenü aus. Beim Export werden die benötigten Informationen in einer XML-Datei gespeichert.

Das SQL-Skript zum Anlegen der Richtlinie in unserem Beispiel finden Sie auf der Buch-CD als Teil der Datei *Kapitel3.sql* vor.

3.5 Was Sie noch wissen sollten

Gegen Ende des Kapitels möchte ich noch drei Aspekte beleuchten, die im praktischen Arbeiten mit Datenbanken von Bedeutung sein können.

3.5.1 Tabellen in anderen Dateigruppen speichern

Zu Beginn des Kapitels haben Sie erfahren, wie Sie für eine Datenbank mehrere Dateigruppen einrichten. Beim Anlegen von Indizes haben Sie gelernt, wie Sie diese in einer anderen als der Standarddateigruppe *PRIMARY* speichern.

Wie können Sie beim Anlegen einer Tabelle festlegen, in welcher Dateigruppe sie angelegt werden soll? Falls Sie keine spezielle Dateigruppe angeben, wird das Objekt immer in der Dateigruppe *PRIMARY* erstellt.

Öffnen Sie beim Erstellen der Tabelle im Tabellen-Designer das Eigenschaftenfenster über das Menü ANSICHT (in R2 lautet der Menübefehl ANZEIGEN) oder mit Hilfe der Taste F4. In der Rubrik REGULÄRE DATENBEREICHSSPEZIFIKATION wählen Sie die Dateigruppe in der Liste bei der Eigenschaft *Schemaname der Dateigruppe* aus.

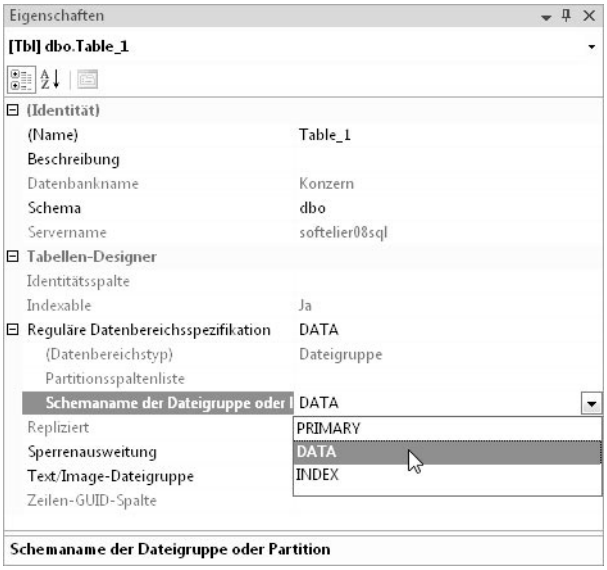


Abbildung 3.54: *Tabelleneigenschaften*

Beim Speichern wird die Tabelle in der gewählten Dateigruppe angelegt.

3.5.2 Tabellen direkt mit DDL-Anweisungen erstellen

Alternativ zum Tabellen-Designer können Sie eine Tabelle in einem Abfrageeditor-Fenster direkt über DDL-Anweisungen erstellen. DDL steht für Data Definition Language und ist ein Teilbereich der Abfragesprache SQL.

Ein einfaches Beispiel soll Ihnen einen kleinen Einblick in dieses Thema geben. Dies kann allerdings keine umfassende Abhandlung sein, denn diese würde über den Rahmen des Buches hinausgehen. Ziel ist vielmehr, Ihnen einen Eindruck zu vermitteln.

Mit der nachfolgenden Anweisung wird eine Personaltabelle mit folgenden Parametern angelegt:

- Für die Personalnummer wird eine Identität mit dem Startwert 1000 und der Schrittweite 1 festgelegt.

- Die erforderliche Eingabe wird bei den Spalten durch den Zusatz `NOT NULL` festgelegt. Bei der Personalnummer kann dies entfallen, weil sie später ohnehin als Primärschlüssel definiert wird.

Für das Eintrittsdatum wird das aktuelle Datum als Standardwert über die Funktion `GETDATE()` festgelegt. (Hier wäre auch die erweiterte Syntax `CONSTRAINT df_personal_eintritt DEFAULT GETDATE()` denkbar, um der Einschränkung einen sauberen Namen anstelle des vom System generierten zu geben.)

- Nach den Spalten werden die Primärschlüssel-Einschränkung sowie eine Check-Einschränkung für die Geschlecht-Spalte definiert.
- Über den Zusatz `ON DATEN` am Ende der Anweisung wird die Tabelle in der Dateigruppe `DATEN` erstellt.

```
CREATE TABLE dbo.tblPersonal
(   PersNr int IDENTITY(1000, 1),
    PersNachname varchar(50) NOT NULL,
    PersVorname varchar(50) NOT NULL,
    PersGeschlecht char(1) NOT NULL,
    PersAbtlg char(2),
    PersDW varchar(5) NOT NULL,
    PersEmail varchar(60),
    PersEintritt date DEFAULT GETDATE(),
    CONSTRAINT pk_personal PRIMARY KEY (PersNr),
    CONSTRAINT ck_personal_geschlecht
    CHECK (PersGeschlecht IN('w','m'))
) ON DATEN;
```

Sie können einer Tabelle auch nach dem Anlegen eine neue Spalte hinzufügen, allerdings nur am Ende. Dazu benötigen Sie die Anweisung `ALTER TABLE`. Das nachfolgende Beispiel ergänzt die Tabelle um eine Spalte für das Austrittsdatum.

```
ALTER TABLE dbo.tblPersonal
ADD PersAustritt date;
```

Auch Einschränkungen können für eine Tabelle nachträglich ergänzt werden. Die folgende Anweisung stellt eine Beziehung zwischen der Personaltabelle und der Abteilungstabelle her.

```
ALTER TABLE dbo.tblPersonal
ADD CONSTRAINT fk_personal_abteilung FOREIGN KEY (PersAbtlg)
REFERENCES dbo.tblAbteilungen (AbtNr);
```

Wenn Sie einen Primärschlüssel mit dieser Methode im Nachhinein ergänzen, muss diese Spalte im Gegensatz zur Beispielanweisung sehr wohl schon als `NOT NULL` definiert sein. Denn nur, wenn der Primärschlüssel in derselben Anweisung definiert wird, kann diese Festlegung entfallen.

Sie müssten also bei Bedarf zuerst die Spalte als `NOT NULL` festlegen, um danach den Primärschlüssel ergänzen zu können:

```
ALTER TABLE dbo.tblPersonal
ALTER COLUMN PersNr int NOT NULL;

ALTER TABLE dbo.tblPersonal
ADD CONSTRAINT pk_personal PRIMARY KEY (PersNr);
```

3.5.3 Gefahren der grafischen Oberfläche

Grafische Oberflächen erleichtern zwar die Arbeit ungemein und sind in der Regel sehr komfortabel; sie können aber auch zu einem Fluch werden, wenn man versteckte Gefahren nicht kennt. Viele Dinge, die über die grafische Oberfläche möglich sind, werden von der Datenbank gar nicht unterstützt. Sie funktionieren nur, weil das grafische Tool auf einen Workaround zurückgreift, der in der Datenbank umfangreiche Änderungen vornimmt. Dies soll Ihnen das folgende Beispiel veranschaulichen:

Neue Spalten können in eine Tabelle nur am Ende eingefügt werden. Es ist in keiner Datenbank möglich, eine Spalte an vorderer Stelle zu ergänzen. Das grafische Tool lässt dies ohne Weiteres zu. Sie können im Tabellen-Designer neue Spalten über das Kontextmenü einfügen oder die Reihenfolge von Spalten mit der Maus verschieben.

Was geschieht, wenn Sie diese Änderungen speichern?

Da die Änderung direkt nicht möglich ist, erstellt das Tool eine völlig neue Tabelle, kopiert alle Daten von der ursprünglichen Tabelle in diese hinein, hebt alle Beziehungen der alten Tabelle auf und erstellt sie für die neue Tabelle. Schließlich wird die alte Tabelle gelöscht und die neue umbenannt. Dies hört sich toll an, wenn man sich vorstellt, alle diese Schritte selber manuell umsetzen zu müssen. Die Gefahr liegt allerdings darin, dass solche Vorgänge ausgeführt werden, während andere Benutzer in der Datenbank arbeiten. In diesem Fall führt dieser Vorgang verständlicherweise zu Problemen. Deshalb sollten Sie solche Änderungen unbedingt zu Zeiten durchführen, in denen nicht in der Datenbank gearbeitet wird.

Wie kann man solche Vorgänge von ungefährlichen Änderungen unterscheiden?

Es gibt zwei Methoden, um diese Vorgänge vor dem Ausführen zu erkennen:

- Meldung beim Speichern: Erhalten Sie beim Speichern eine Meldung, dass die angeführten Tabellen gespeichert werden, sollten bei Ihnen die Alarmglocken läuten. Die nachfolgende Meldung erscheint, wenn in der Artikeltabelle der Beispieldatenbank *WAWI* die Reihenfolge der Spalten verändert wird. Warum müssen so viele Tabellen gespeichert werden, wenn nur Änderungen an einer Tabelle vorgenommen wurden? – Weil alle auf diese Tabelle verweisenden Fremdschlüssel vorübergehend gelöscht werden müssen, da sonst die alte Artikeltabelle nicht gelöscht werden kann.

Abbildung 3.55: *Abfrage beim Speichern*

- Analyse des Änderungsskripts: Wenn Sie Ihre Änderungen eingegeben, aber noch nicht gespeichert haben, können Sie sich das Änderungsskript, das alle ausgeführten Anweisungen enthält, anzeigen lassen. Wählen Sie dazu im Kontextmenü den Befehl ÄNDERUNGSSKRIPT GENERIEREN... oder klicken Sie auf das entsprechende Symbol. Sofern Sie im Skript die Anweisungen `DROP TABLE` finden, handelt es sich um den beschriebenen Vorgang.

Speichern Sie das Änderungsskript als Textdatei, dann können Sie es einfacher durchsuchen.

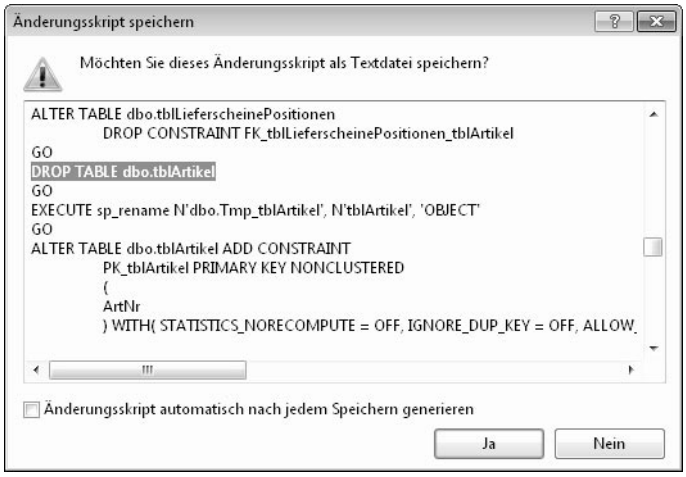


Abbildung 3.56: *Änderungsskript mit DROP TABLE*

NEU 2008

Neu in der Version ist, dass sie diese gefährlichen Änderungen generell unterbinden können. In diesem Fall erhalten Sie eine Fehlermeldung beim Speichern, falls eine Neuerstellung der Tabelle notwendig ist.

Diese Einstellung finden Sie im Menü EXTRAS unter dem Befehl OPTIONEN... In der Rubrik DESIGNER, in der Unterkategorie TABELLEN- UND DATENBANK-DESIGNER finden Sie die Option SPEICHERN VON ÄNDERUNGEN VERHINDERN, DIE DIE NEUERSTELLUNG DER TABELLE ERFORDERN. Aktivieren Sie diese Option, um solche Vorgänge zu unterbinden.

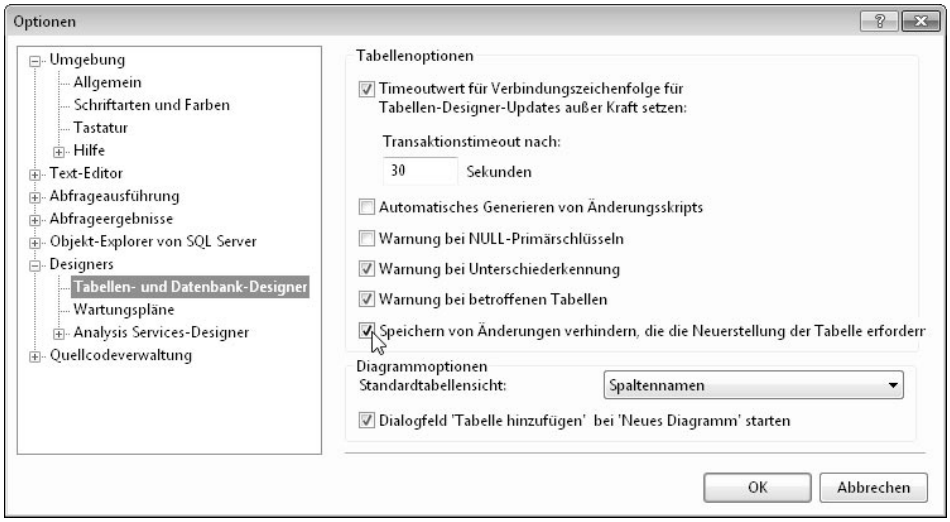


Abbildung 3.57: *Sicherheitseinstellung über die Optionen*

Ist diese Option aktiv, so kann die Änderung nicht gespeichert werden. Sie müssen sie daher bei Bedarf deaktivieren.

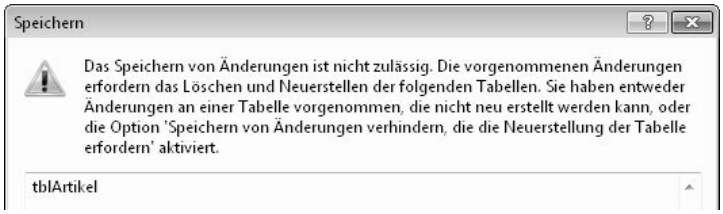


Abbildung 3.58: *Verhindern des Neuerstellens von Tabellen*

Vorgänge mit den beschriebenen Merkmalen sollten Sie nur ausführen, wenn sonst keine Aktivitäten in der Datenbank stattfinden. Das Anfügen einer neuen Tabellenspalte am Ende kann hingegen gefahrlos im laufenden Betrieb erfolgen.

Leider arbeitet das grafische Tool nicht immer sauber. In manchen Situationen möchte es die Tabelle neu erstellen, ohne dass dies wirklich erforderlich wäre. Ändern Sie zum Beispiel den Datentyp der Spalte *ArtEKPreis* in der Tabelle *tblArtikel* der Beispieldatenbank

WAWI von *smallmoney* auf *money*, so kommt es zu einer Neuerstellung der Tabelle durch das grafische Tool. Dies wäre aber gar nicht notwendig, da die Änderung auch durch eine einfache `ALTER TABLE`-Anweisung möglich wäre.

```
ALTER TABLE dbo.tblArtikel
ALTER COLUMN ArtEKPreis money;
```

Leider ist das grafische Tool hier nicht so sauber programmiert, dass alle Möglichkeiten von SQL ausgeschöpft werden, bevor es zur Neuerstellung der Tabelle kommt. Der SQL Server kann hier nämlich einiges mehr, als sein Management Studio vorgibt.

3.6 Tabelle mit Filestream erstellen

In einem früheren Abschnitt dieses Kapitels habe ich ihnen gezeigt, wie man eine Datenbank mit einer Filestream-Dateigruppe anlegt, und Ihnen das grundlegende Konzept von Filestream erläutert. Nun möchte ich Ihnen zeigen, wie man Filestream in einer Tabelle nutzt. Leider bietet der Tabellendesigner auch in der R2 noch immer keine Möglichkeit, eine Spalte zur Filestream-Nutzung zu definieren. Daher muss man im Moment eine Tabelle, die Filestream enthält, mit einer SQL-Anweisung erstellen. Das ist auch der Grund, warum ich dieses Thema erst am Ende dieses Kapitels behandle.

Ich möchte nun in der Video-Datenbank eine Tabelle erstellen, in der Video-Dateien mit ein wenig Zusatzinformation versehen gespeichert werden sollen. Für diese Video-Dateien soll Filestream genutzt werden.

Um Filestream zu nutzen, muss eine Tabelle folgende Voraussetzungen erfüllen:

- Eine Spalte vom Datentyp *varbinary(max)* muss mit dem Attribut `FILESTREAM` versehen werden.

Filestream ist nur für den Datentyp *varbinary(max)* und nicht für *varchar(max)* verfügbar. Wenn Sie Filestream für umfangreiche Texte nutzen möchten, müssen Sie für diese anstelle von *varchar(max)* daher *varbinary(max)* verwenden.

- Die Tabelle muss eine Spalte enthalten, die als *UNIQUEIDENTIFIER* und *ROWGUID* definiert ist. Ein *UNIQUEIDENTIFIER* ist ein Wert, der aus 32 hexadezimalen Stellen besteht. Nach der achten, zwölften, sechzehnten und zwanzigsten Stelle kommt jeweils ein Bindestrich (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx). Während eine als *IDENTITY* definierte Spalte immer innerhalb der Tabelle eindeutige Werte enthält, enthält eine als *UNIQUEIDENTIFIER* festgelegte Spalte eindeutige Werte über alle Tabellen einer Datenbank hinweg. Damit ist ein absolut eindeutiger Wert innerhalb der Datenbank gegeben. Während *IDENTITY*-Spalten beim Anlegen eines neuen Datensatzes automatisch mit einem neuen Wert versehen werden, geschieht dies bei *UNIQUEIDENTIFIER*-Spalten nicht. Damit diese auch vom System einen Wert zugewiesen bekommen, muss zusätzlich die *ROWGUID*-Eigenschaft gesetzt werden.

  Dieser eindeutige Wert wird für die Assoziierung der im Dateisystem gespeicherten Datei mit dem Datensatz verwendet. Damit ist ausgeschlossen, dass es auf dieser

Ebene zu Namenskonflikten kommt, auch wenn in mehreren Tabellen Filestream genutzt wird. (Uniqueidentifier werden ja auch bei Replikation genutzt, um Datensätze eindeutig zu identifizieren.)

- Die *UNIQUEIDENTIFIER*-Spalte darf nicht leer sein und muss daher als *NOT NULL* definiert werden. Außerdem muss sie mit einem *UNIQUE KEY-Constraint* versehen werden, um Eindeutigkeit zu erzwingen. Zwar werden vom System nur eindeutige Werte generiert, aber anders als bei einer Indentity kann ein Uniqueidentifier auch manuell eingetragen werden. Andernfalls könnte sonst gegen die Eindeutigkeit verstoßen werden.

Erstellen wir in der Datenbank *Video* nun eine neue Tabelle mit dem Namen *tblMedien*. Dazu verwenden wir folgende Anweisung. In der ersten Variante verwenden wie die Spalte *ID* als `UNIQUEIDENTIFIER`. Anstelle eines `UNIQUE KEY`-Constraints mit `NOT NULL` versehen kann natürlich auch gleich ein `PRIMARY KEY`-Constraint verwendet werden.

```
CREATE TABLE dbo.tblMedien
(   ID UNIQUEIDENTIFIER ROWGUIDCOL CONSTRAINT pk_medien PRIMARY KEY,
    Titel varchar(50) NOT NULL,
    Beschreibung varchar(1000),
    Aufnahmedatum date NOT NULL,
    Video varbinary(max) FILESTREAM
);
```

Die Spalte *Titel* soll die Kurzbezeichnung des Videos aufnehmen. Deren Eingabe soll verpflichtend sein. Die *Beschreibung* kann einen längeren Beschreibungstext aufnehmen. Ebenso soll das *Aufnahmedatum* nicht fehlen. Die Spalte *Video* muss mit dem Datentyp *varbinary(max)* definiert werden. Entscheidend ist der Zusatz `FILESTREAM` dahinter. Durch diesen wird diese Spalte für Filestream definiert, sonst würden die Videoclips später direkt in der Datenbank gespeichert, und die maximale Größe pro Video wäre 2 GB. Da wir in der Datenbank nur eine Dateigruppe für Filestream definiert haben, ist diese die Default-Dateigruppe und muss daher in der SQL-Anweisung nicht extra als Ziel angegeben werden.

Falls Sie als Primärschlüssel beziehungsweise als ID gerne einen anderen Wert als den kryptischen 32-stelligen Uniqueidentifier verwenden möchten, können Sie auch die Funktionen von Primärschlüssel und Uniqueidentifier von einander trennen.

Dafür käme dann zum Beispiel folgende Tabellendefinition zum Einsatz:

```
CREATE TABLE dbo.tblMedien
(   ID int IDENTITY CONSTRAINT pk_medien PRIMARY KEY,
    Titel varchar(50) NOT NULL,
    Beschreibung varchar(1000),
    Aufnahmedatum date NOT NULL,
    Video varbinary(max) FILESTREAM,
    StreamID UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL
    CONSTRAINT uk_medien_stream UNIQUE
);
```

Gegenüber der ersten Variante habe ich hier die ID als Identität mit dem Datentyp *int* definiert. Für den benötigten Uniqueidentifier habe ich die Spalte *StreamID* ergänzt. Wieder wird die `RowGuid`-Eigenschaft über den Zusatz `ROWGUIDCOL` gesetzt. Der für den Anwendungsfall geforderte Unique Key-Constraint ersetzt hier den Primärschlüssel. Wichtig ist auch die Definition dieser Spalte als `NOT NULL`.

Für die weiteren Schritte des Beispiels habe ich die zweite Variante verwendet. Um Ihnen eine mögliche Verwendung von Filestream in einer Anwendung zu zeigen, möchte ich eine »quick and dirty« erstellte Access-Anwendnug verwenden. Es handelt sich dabei um eine ADP-Datei (ADP = Access Data Project), welche direkt auf den SQL Server zugreift.

Sie finden diese Datei mit dem Namen *filestream.adp* auf der CD und können sie ab der Version Access 2000 einsetzen. Ebenso auf der CD finden Sie das Skript *Kapitel3.sql*, welches die Anweisungen zum Erstellen der Datenbank *Video* und der Tabelle *tblMedien* enthält. Diese Anweisungen können Sie verwenden, um die in diesem Kapitel beschriebenen Befehle nicht abtippen zu müssen. Achten Sie aber bitte darauf, dass Sie bei Ihrer Konfiguration gegebenenfalls die Pfade der Dateigruppen an Ihre Gegebenheiten anpassen. Die Access-Datei nutzt die zweite vorgestellte Variante der Tabelle *tblMedien* mit dem separaten Primärschlüssel.

Wenn Sie die Datei öffnen, muss diese mit Ihrer Datenbank auf Ihrem Server verbunden werden. Der Dialog hierfür wird automatisch geöffnet. Verwenden Sie dieselben Anmeldeinformationen, die Sie für das Management Studio verwenden.

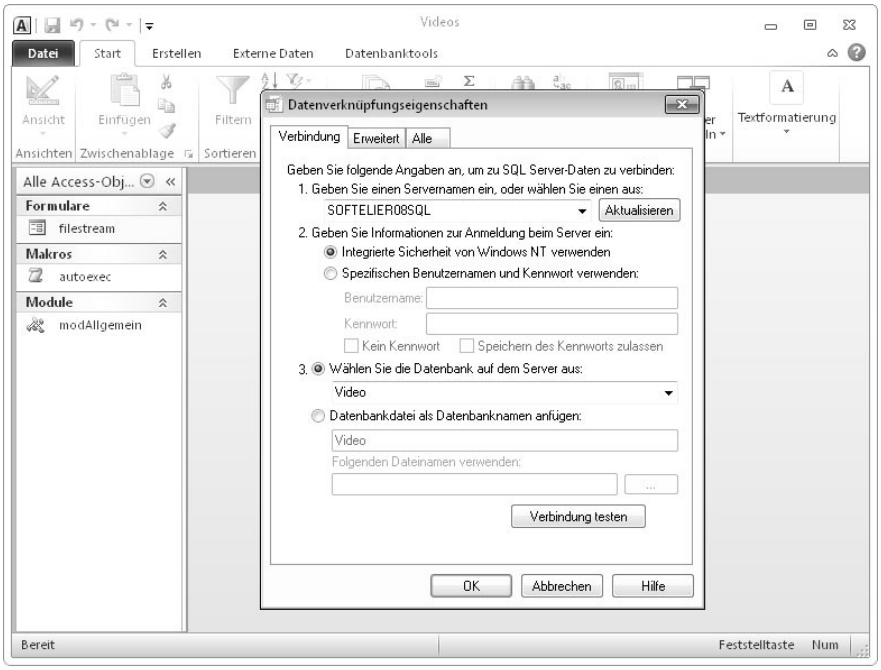


Abbildung 3.59: *Verbindung von Access-ADP mit SQL Server herstellen*

Ohne eine gültige Verbindung wird die Anwendung sofort wieder beendet. Wenn Sie eine gültige Verbindung hergestellt haben, wird das Formular zur Erfassung der Mediendaten angezeigt. Sie können nun einen ersten Datensatz erfassen. Sie finden auf der Buch-CD ein paar Videoclips im Format WMV, die sie hierzu verwenden können. Natürlich können Sie auch jede beliebige Mediadatei verwenden.

Bevor wir einen ersten Datensatz anlegen, werfen wir einen Blick auf das Verzeichnis, das wir für die Filestream-Dateigruppe festgelegt haben. Bei mir ist dies der Ordner *D:\DB_FS\Video_Stream* auf dem Server. Ich verbinde mich über *$-Share* dorthin. Dieser Ordner ist – wie erwartet – im Moment noch weitgehend leer:

- Nach dem Anlegen der Datenbank *Video* befinden sich im Ordner ein Unterordner mit dem Name *$FSLOG* und die Datei *filestream.hdr*.
- Nach dem Anlegen der Tabelle *tblMedien* kommt ein Ordner, dessen Name das Format eines Uniqueidentifiers aufweist, hinzu.

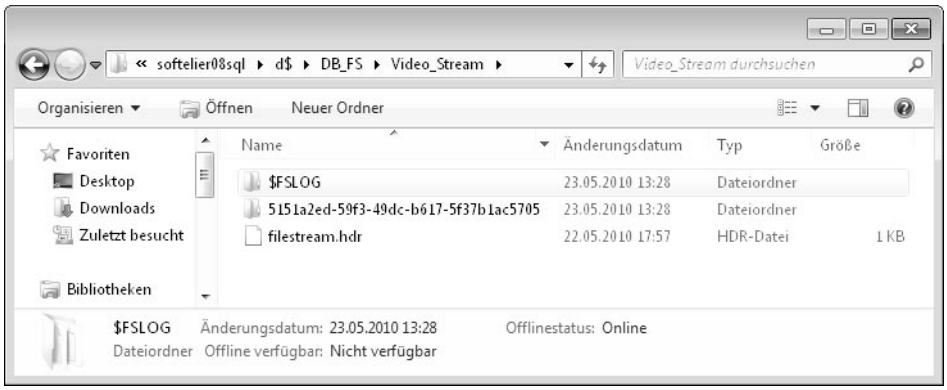


Abbildung 3.60: *Basisordner für Filestream, bevor Daten erfasst werden*

Erfassen Sie nun über die Access-Anwendung Titel, Bezeichnung und Aufnahmedatum für den ersten Clip. Fügen Sie den Clip entweder per Drag & Drop ein oder verwenden Sie dazu im Kontextmenü den Befehl OBJEKT EINFÜGEN.

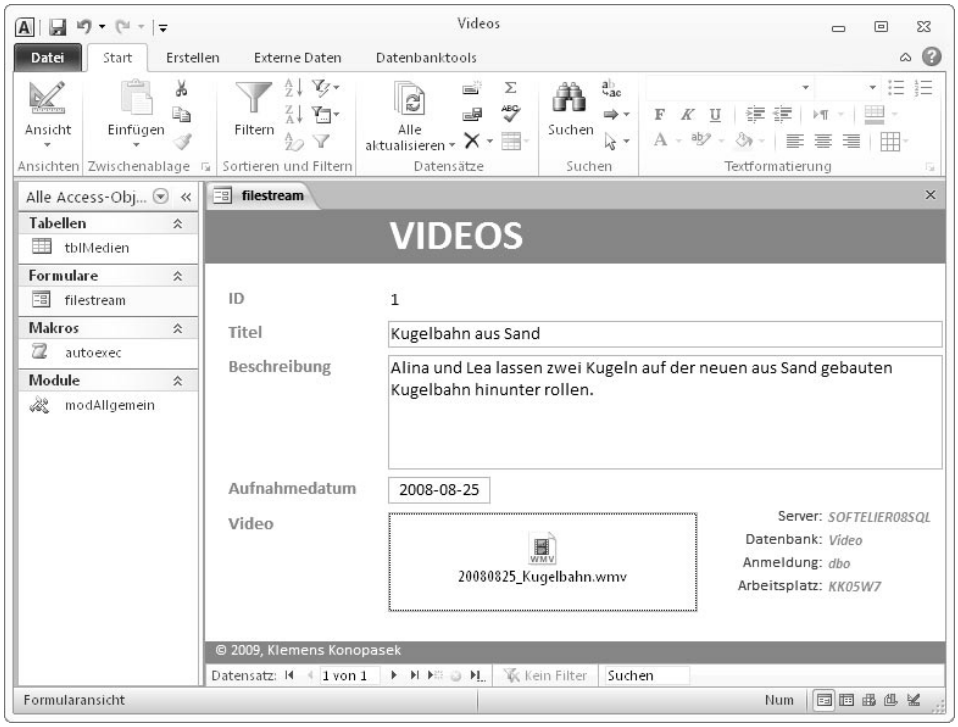


Abbildung 3.61: *Datensatz über Access-Formular erfasst*

Für die Anwendung (in diesem Fall MS Access) ist Filestream vollkommen transparent. Es macht für Sie keinen Unterschied, ob die Video-Datei in der Datenbank oder auf dem Server im Dateisystem gespeichert wird. Wo finden wir diese Datei auf dem Server? – In einem der Unterordner. Der Dateiname ist für uns nicht sprechend; er muss es aber auch nicht sein.

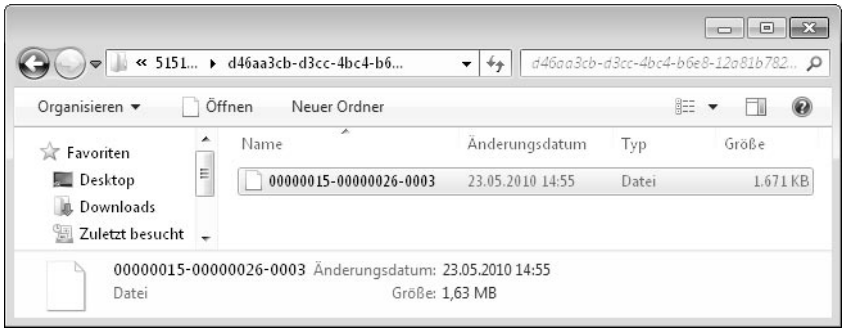


Abbildung 3.62: *Filestream-Objekt im Dateisystem des Servers*

Wenn Sie diesen Datensatz löschen, verschwindet diese Datei wieder. Aktivieren Sie die Datei über ihre Anwendung, zum Beispiel indem Sie sie im Access-Formular doppelt ankli-

cken, wird sie vom SQL Server aus dem Dateisystem gestreamt und an den Client übertragen. Wie gesagt, ist dies für die Clientanwendung transparent. Wenn wir den Datensatz im Abfrageeditor des Management Studios anzeigen, sehen wir die ID des Streamings in Form des Uniqueidentifiers.

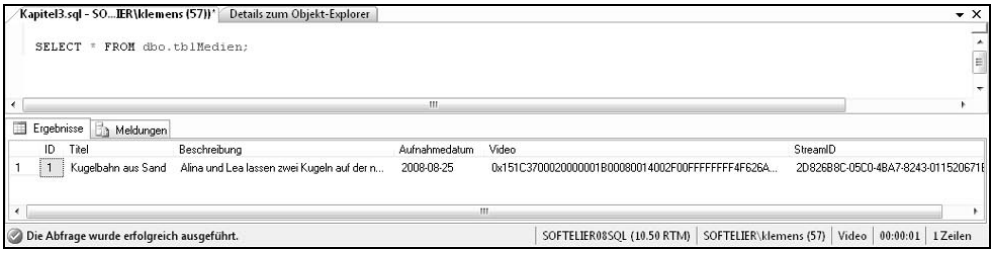


Abbildung 3.63: *Datensatz mit Filestream*

Um den Film in der Beispielanwendung anzusehen, müssen Sie das Symbol im Formular lediglich doppelt anklicken. Der SQL Server streamt die Datei dann vom Dateisystem an die Anwendung, in diesem Fall an MS Access. Das Objekt wird anschließend mit der mit dem Dateityp assoziierten Anwendung (in diesem Fall dem Media Player) abgespielt. Entscheidend ist, dass hier aus der Anwendung heraus kein Unterschied gegenüber einer in der Datenbank gespeicherten Datei zu erkennen ist.

Abbildung 3.64: *Wiedergabe der vom Dateisystem des Servers gestreamten Datei*

Legen Sie bitte zur Übung noch weitere Datensätze an und löschen Sie auch wieder welche, um den Vorgang zu testen!

Folgendes sollten Sie beim Einsatz von Filestream unbedingt beachten:

- Die vom SQL Server angelegten Ordner und Dateien sind nicht für einen direkten Zugriff gedacht. Tun Sie dies bitte ausschließlich in Testumgebungen und greifen Sie auch nur lesend darauf zu. In Produktivsystemen sollten keine Benutzer Zugriffsrechte auf diese Ordnerstruktur haben.
- Der Tabellendesigner kann mit Filestream (noch) nicht umgehen. Wenn Sie über ihn eine Änderung an der Tabelle vornehmen, die wie beschrieben eine Neuerstellung der Tabelle erfordert, fehlt der Filestream danach bei der neu angelegten Tabelle. Zumindest erfolgt das so sauber, dass bei meinem Test die Datei vom Dateisystem in die Datenbank kopiert worden ist. Wenn auch kein Filestream mehr vorhanden ist, so sind wenigstens keine Daten verloren gegangen. Dennoch ist das natürlich nicht erwünscht. Bearbeiten Sie daher in der Praxis Tabellen mit Filestream bitte mit DDL-Anweisungen wie `ALTER TABLE` und nicht über den grafischen Tabellendesigner.

Wie Sie Datenbanken mit Filestream sichern und wiederherstellen und damit auf einen anderen Server transferieren können, lesen Sie in Kapitel 8, welches sich unter anderem mit dem Thema Sicherung beschäftigt.