PEARSON Education

ADDISON-WESLEY

Klemens Konopasek
Ernst Tiemeyer

SQL Server 2005 – Der schnelle Einstieg

Abfragen, Transact-SQL, Entwicklung und Verwaltung

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

3 Eine neue Datenbank erstellen

Nachdem wir den SQL Server 2005 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 einerseits, wie Sie dazu vorgehen, andererseits möchten wir Sie auch 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 Abfrage-Editorfenster mit der Anweisung `CREATE DATABASE` die neue Datenbank erstellen. Dazu würde es genügen, den Namen der Datenbank zusätzlich zur Anweisung anzugeben. Diese wird dann mit allen Standardeinstellungen erstellt.

Probieren Sie es ruhig aus! Geben Sie in einem Abfrage-Editorfenster die Anweisung `CREATE DATABASE kapitel3` ein. Führen Sie diese Anweisung aus, haben Sie schon Ihre erste Datenbank erstellt. Die Dateien dieser Datenbank finden Sie im Ordner …\MSSQL.1\MSSQL\DATA unter jenem Verzeichnis, das Sie während des Setups als Datenverzeichnis angegeben haben.

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 Sinn machen, 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 wird sie auch die einzige Datendatei sein. In dieser werden neben den Benutzerdatenbankobjekten (Tabellen etc.) auch die Systemobjekte der Datenbank gespeichert. In den Systemtabellen werden zum Beispiel die ganze 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.
Transaktionsprotokoll-Dateien	LDF	Es können eine oder mehrere Transaktionsprotokolldateien für eine Datenbank festgelegt werden. Im Transaktionsprotokoll werden alle Schreibvorgänge in der Datenbank protokolliert. Diese Informationen dienen der 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 und in Kapitel 8 zum Thema Sicherung und 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 in der Regel nur bei größeren Systemen verwendet. Unter folgenden 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, kann bei Lesevorgängen parallelisiert werden. Nachdem ein Indexeintrag auf dem einen Datenträger gelesen worden ist, kann bereits der nächste Indexeintrag gesucht und gelesen werden, während in der Zwischenzeit bereits 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 solchen Dateien unterbringen.

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

Für jede Datei einer Datenbank können folgende 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 deren 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, wenn sie „voll" ist. Die Vergrößerung kann als Prozentsatz der bisherigen Größe oder als fixe Größe in Megabyte erfolgen. Zusätzlich kann festgelegt werden, 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, kann in der Datenbank nur mehr gelesen werden. Schreibvorgänge sind erst nach Schaffen von weiterem Speicherplatz möglich. Dies kann zum Beispiel durch Hinzufügen von weiteren Dateien oder 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 Filesystem mit der Erweiterung 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 immer in der Dateigruppe PRIMARY gespeichert.

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

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

- *Gemeinsame Dateigruppe*: Eine Dateigruppe werden Sie dann gemeinsam für mehrere Datendateien verwenden, wenn mehrere Dateien nur aufgrund des Speicherplatzmangels erstellt worden sind. In diesem Fall verteilt der SQL Server die Daten selber auf diese Dateien. Sie haben keinen Einfluss darauf, in welcher Datei zum Beispiel eine Tabelle physisch abgelegt wird. Wenn Sie zum Beispiel eine weitere Datei ergänzen, um den zur Verfügung stehenden Speicherplatz nur zu erweitern, werden Sie diese derselben Dateigruppe anfügen.
- *Getrennte Dateigruppe*: Sie verwenden eine eigene Dateigruppe für eine Datei, wenn 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.

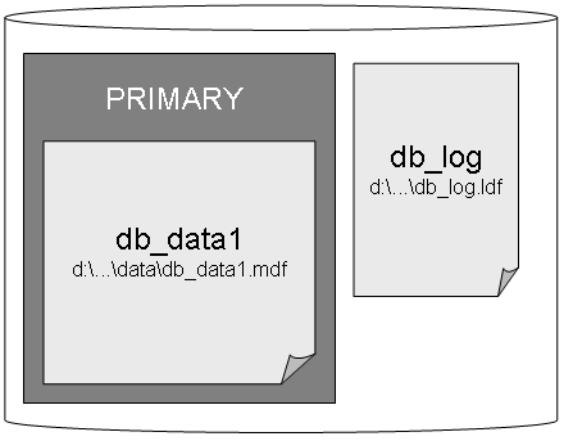

Abbildung 3.1: Einfache Datenbank mit einer Datendatei in einer Dateigruppe

Die Abbildung 3.2 zeigt eine möglich 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 außerdem auf einem anderen Datenträger. Das Transaktionsprotokoll kann keiner Dateigruppe zugeordnet werden.

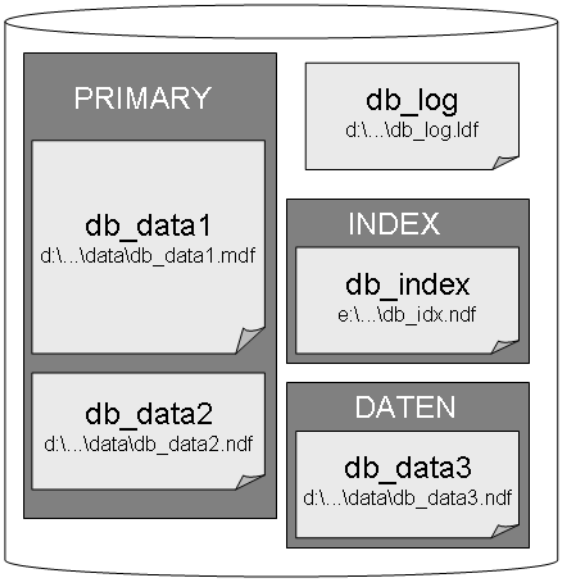

Abbildung 3.2: Datenbank mit mehreren Datendateien und Dateigruppen

Nach diesem einleitenden Überblick über den Aufbau einer Datenbank möchten wir im nächsten Schritt zum Anlegen einer solchen 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 jenem SQL Server an, auf dem Sie die Datenbank erstellen möchten.
2. Markieren Sie den Ordner *Datenbanken*, 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.

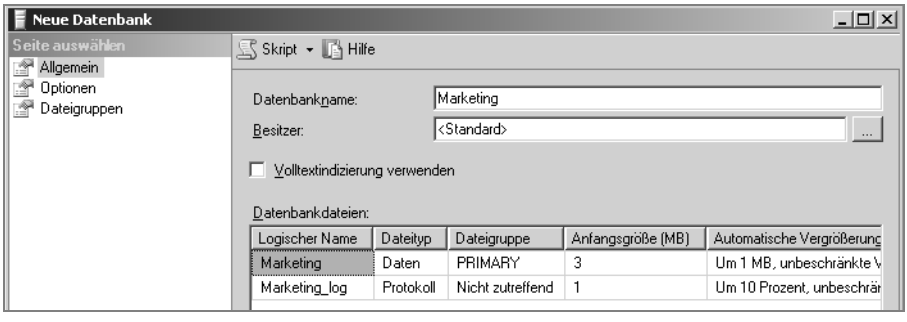

Abbildung 3.4: Name für Datenbank vergeben

4. Wenn Sie möchten, können Sie einen Benutzer als Besitzer für die Datenbank angeben. Tun Sie das nicht und übernehmen 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.

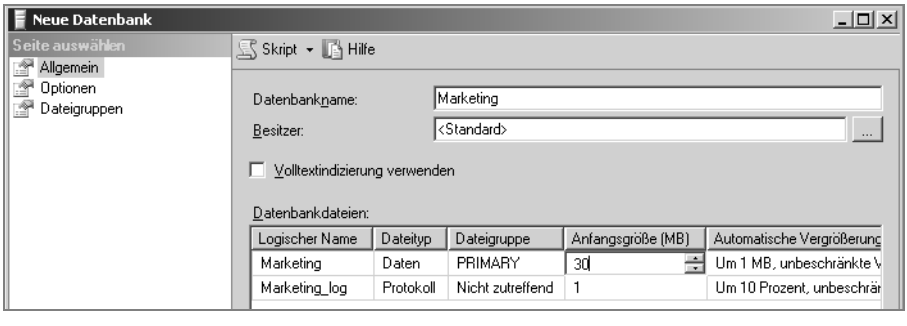

Abbildung 3.5: Anfangsgröße der Datendatei anpassen

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.

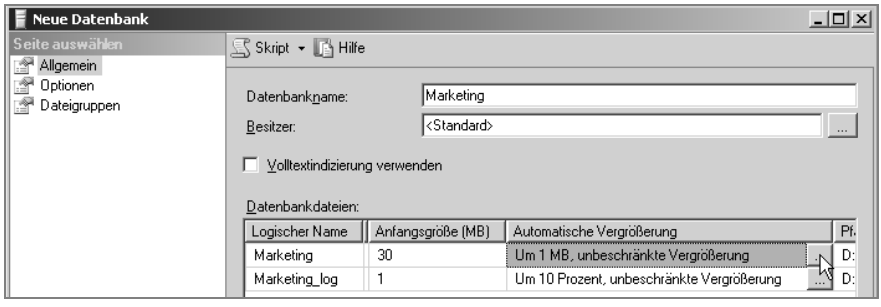

Abbildung 3.6: Automatische Vergrößerung einstellen

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 die maximale Dateigröße von 500 MB.

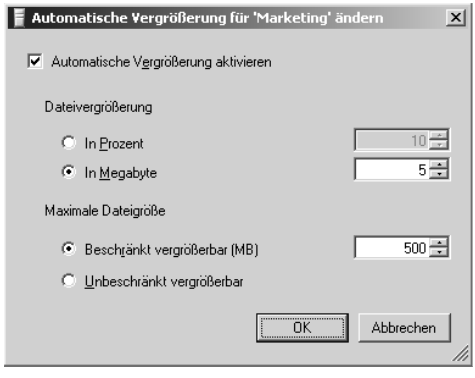

Abbildung 3.7: Einstellungen für automatische Vergrößerung ändern

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 jene Ihres Rechners.

Als Ziel können lediglich lokale Pfade, keine Netzlaufwerke verwendet werden.

Der Dateiname kann im Management Studio leider nicht direkt ins Feld eingegeben werden. Ändern Sie ihn also indirekt über den logischen Namen. Dieser wird für die Datendatei mit der Dateierweiterung MDF sowie für die Protokolldatei mit der Erweiterung LDF übernommen.

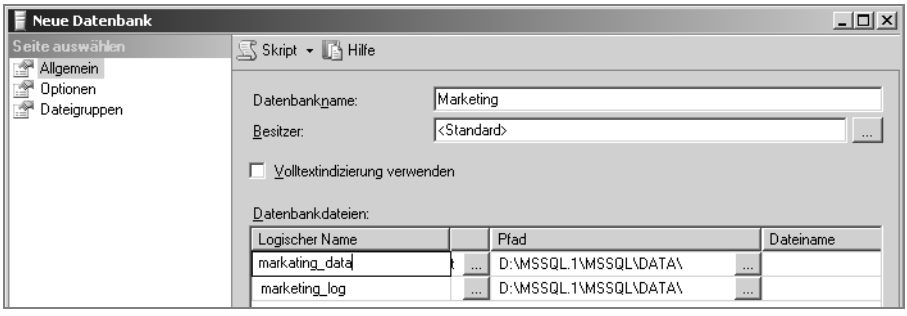

Abbildung 3.8: Pfad und Name für Datenbankdateien angeben

9. Wenn Sie möchten, können Sie noch auf die Seite *Optionen* wechseln. Hier können Sie verschiedene Einstellungen wie zum Beispiel die *Sortierung* (Collation), das *Wiederherstellungsmodell* und den *Kompatibilitätsgrad* vornehmen.

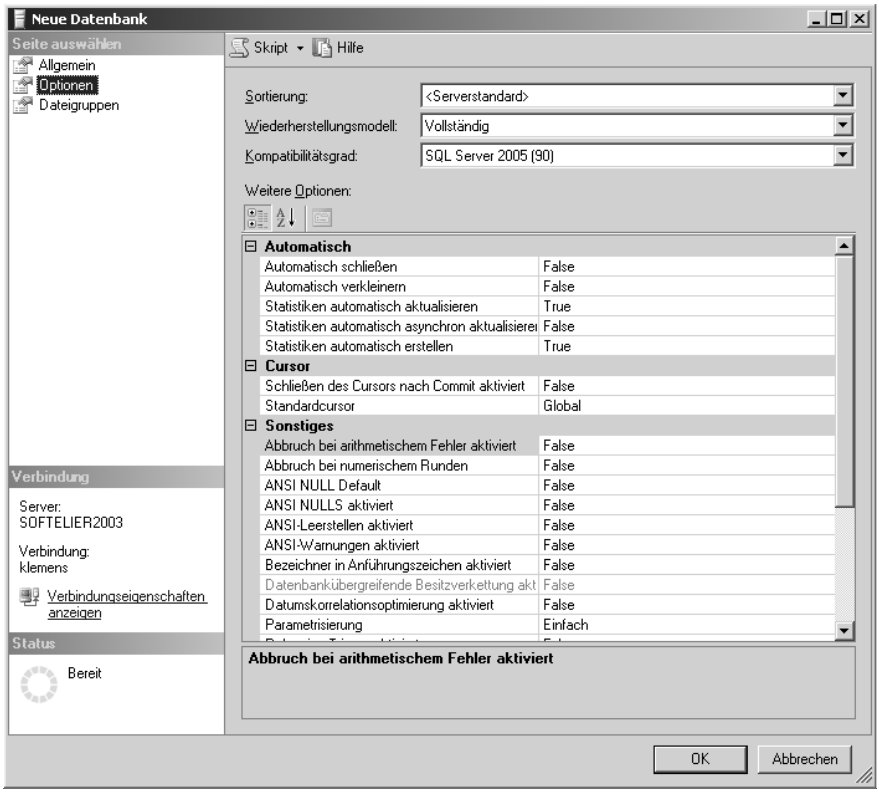

Abbildung 3.9: Datenbankoptionen einstellen

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 den Features welcher Version diese Datenbank kompatibel ist. Hier könnte auch eine der beiden Vorversionen 7.0 und 2000 gewählt werden. Dies ist aber nur in Ausnahmefällen sinnvoll.

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

Die neue Datenbank wird nun im Objekt-Explorer angezeigt.

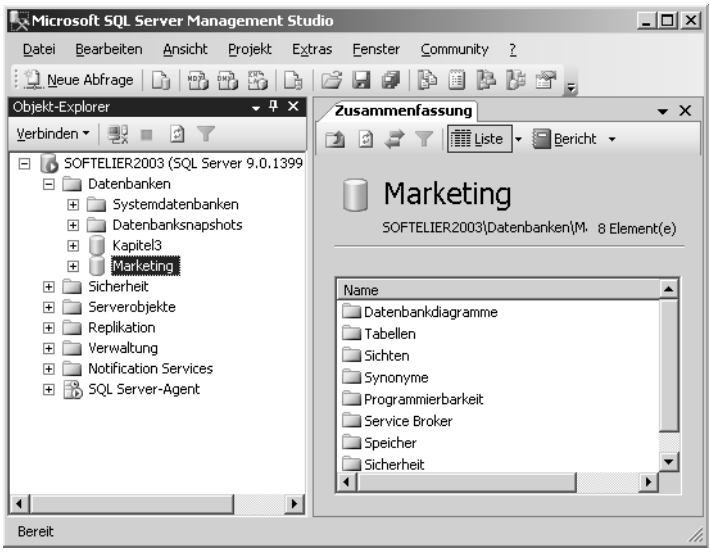

Abbildung 3.10: Neue Datenbank im Management Studio

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 *Optionen* vorerst die benötigten Dateigruppen über die Schaltfläche HINZUFÜGEN. In unserem Beispiel sind dies die Dateigruppen *DATEN* und *INDEX*.

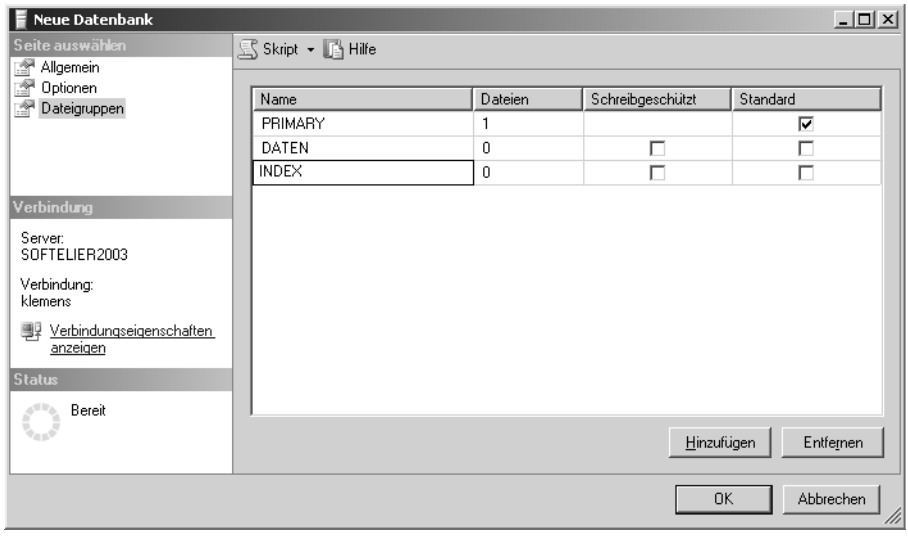

Abbildung 3.11: Dateigruppen anlegen

Auf der Seite *Allgemein* können Sie nun beim Hinzufügen von weiteren Datendateien diese Dateigruppen aus der Liste auswählen. Wie Sie in Abbildung 3.12 sehen, kann bei einer Datenbankdatei, die als Typ *Protokoll* definiert ist, keine Dateigruppe ausgewählt werden.

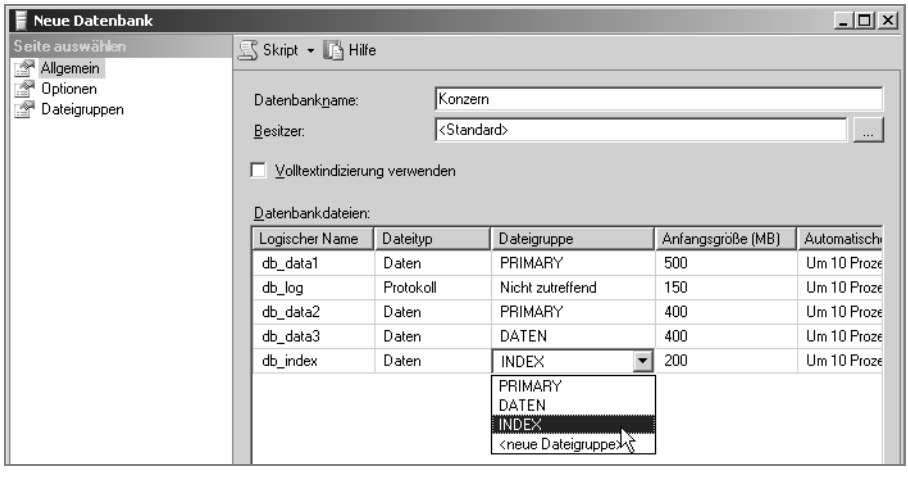

Abbildung 3.12: Datendateien ergänzen und Dateigruppen zuordnen

Wenn 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, gilt folgende Faustregel:

- Bei einer standardmäßigen OLTP-Anwendung beträgt die Größe des Transaktionsprotokolls circa ein Drittel der Größe der Datendateien. (Zum Beispiel eine Warenwirtschaftsanwendung.)
- Bei Archivdatenbanken 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 eine Datenbank für die Verarbeitung von Messwerten einer Produktion.)

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

Da im Transaktionsprotokoll sehrt viele Schreibzugriffe erfolgen, ist es hier von besonderer Bedeutung, dass die Datei nicht fragmentiert ist. Wählen Sie daher im Zweifel – wenn möglich – eine etwas größere Anfangsgröße.

3.1.3 Datenbank über ein SQL-Anweisung erstellen

Wie bereits früher in diesem Kapitel erwähnt, lässt sich eine Datenbank ganz schnell mit der Anweisung `CREATE DATABASE name` über ein Abfrage-Editorfenster (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 Standarddatenbankordner 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. Deren Anfangsgröße beträgt 1 MB, sie wächst unbeschränkt jeweils um 10%. Auch sie wird im Standarddatenbankordner angelegt.
- Sie selber sind der Besitzer der Datenbank.
- Volltextindizierung ist aktiv.
- Der Serverstandard wird für die Sortierung herangezogen, ebenso das vollständige Wiederherstellungsmodell und der Kompatibilitätsgrad für SQL Server 2005 (9.0).

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:\MSSQL.1\MSSQL\DATA\ma_data.mdf',
    SIZE = 30720KB , MAXSIZE = 512000KB , FILEGROWTH = 5120KB)
LOG ON
(   NAME = N'marketing_log',
    FILENAME = N'D:\MSSQL.1\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.

Lässt der grafische Editor zwar keinen vom logischen Namen abweichenden physischen Dateinamen zu, ist dies beim CREATE DATABASE-Kommando jedoch möglich, wie das vorige Beispiel zeigt.

Soll eine Datenbank mit mehreren Dateigruppen und mehreren Datendateien angelegt werden, ist diese Anweisung so zu erweitern, wie dies das nachfolgende Beispiel zeigt:

```
CREATE DATABASE Konzern ON PRIMARY
(   NAME = 'db_data1',
    FILENAME = 'D:\MSSQL.1\MSSQL\DATA\db_data1.mdf',
    SIZE = 512000KB , FILEGROWTH = 10%),
(   NAME = N'db_data2',
    FILENAME = 'D:\MSSQL.1\MSSQL\DATA\db_data2.ndf',
    SIZE = 409600KB , FILEGROWTH = 10%),
FILEGROUP DATEN
(   NAME = 'db_data3',
    FILENAME = 'D:\MSSQL.1\MSSQL\DATA\db_data3.ndf',
    SIZE = 409600KB , FILEGROWTH = 10%),
FILEGROUP INDEX
(   NAME = 'db_index',
    FILENAME = 'D:\MSSQL.1\MSSQL\DATA\db_index.ndf',
    SIZE = 204800KB , FILEGROWTH = 10%)
LOG ON
(   NAME = 'db_log',
    FILENAME = 'D:\MSSQL.1\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.

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 möchten wir Ihnen zeigen, wie Sie eine Tabelle anlegen, wie Sie diese indizieren und mit Gültigkeitsregeln versehen. Ein besonderes Augenmerk wird auch auf das Erstellen von Beziehungen gelegt werden.

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

Wie möchten in der im vorigen Abschnitt angelegten 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.

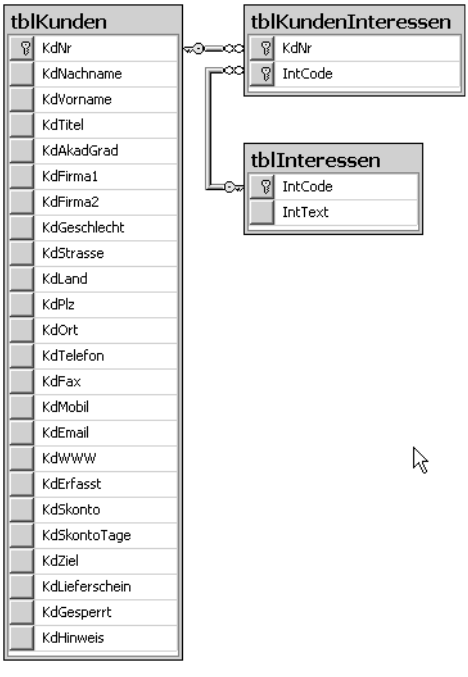

Abbildung 3.13: Anzulegende Beispieltabellen

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, verwenden wir 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 Zusammenfassung wählen Sie im Kontextmenü den Befehl NEUE TABELLE… aus.

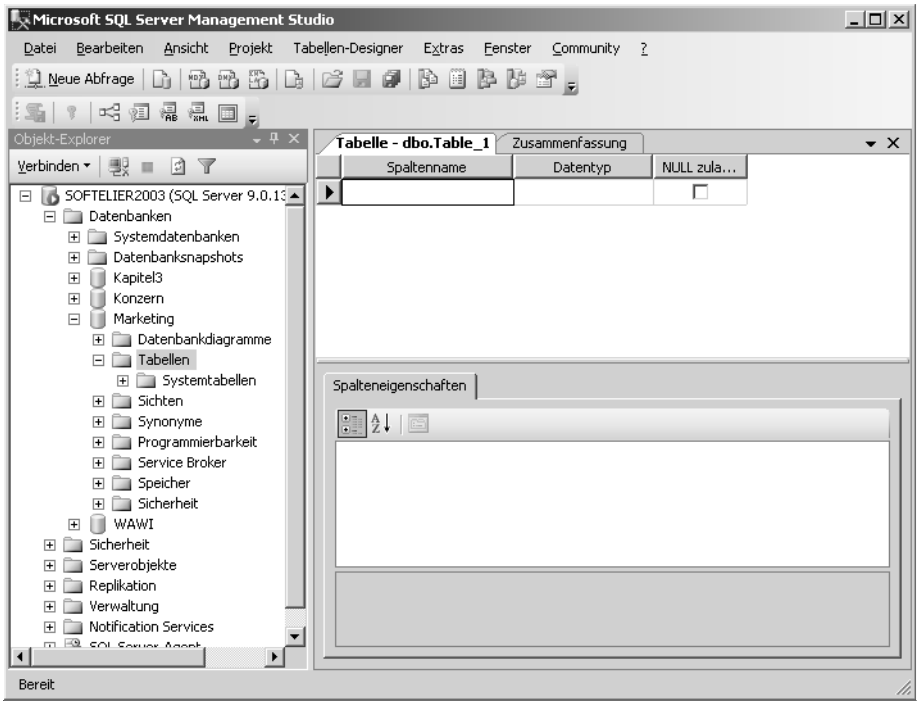

Abbildung 3.14: Neue Tabelle anlegen

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

- Vermeiden Sie *Leer- und Sonderzeichen*, der Unterstrich gilt dabei nicht als Sonderzeichen. Feldnamen in Sonderzeichen müssten immer in eckigen Klammern oder unter doppelten Hochkommata (so genannte *quoted identifiers* in ANSI-SQL) geschrieben werden.
- Feldnamen sollten *sprechend* sein, aber auch möglichst kurz und prägnant. Die maximale Länge beträgt 128 Zeichen.

- Legen Sie sich ein Namenschema zurecht, nach dem Sie die Namen vergeben. Sie merken sich diese dann leichter, wenn Sie sie im Zweifel anhand Ihrer Namenslogik herleiten können. Wir verwenden beispielsweise Präfixe, die den Namen der Tabelle widerspiegeln.

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

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 und 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. Die Typen mit dem fixen Parameter *max* sind in dieser Version neu. Sie können maximal 2 GB an Daten aufnehmen.
Datum/ Uhrzeit	datetime smalldatetime	Datums- und Zeitangaben. *Datetime* reicht von 01.01.1753 bis 31.12.9999 auf 3,33 Millisekunden genau. *Smalldatetime* kommt mit 4 Byte aus, reicht dafür aber mit Minutengenauigkeit nur von 01.01.1900 bis 06.06.2079.
Zahlen	decimal(Genauigkeit, Dezimalstellen) numeric(Genauigkeit, Dezimalstellen) float real bigint int smallint tinyint	*Decimal* und *numeric* sind derselbe Datentyp. Ihre Größe wird durch die Genauigkeit in Stellen (maximal 38) und die darin enthaltenen Dezimalstellen angegeben. *Real* und *float* repräsentieren Gleitkommazahlen. Da nicht alle Werte im Bereich exakt dargestellt werden können, eignen sich diese Datentypen nicht für Primärschlüssel. Die Integer-Typen repräsentieren ganze Zahlen. *Bigint* hat einen Wertebereich von –2 ^ 63 bis 2 ^ 63 und benötigt dafür 8 Byte. *Int* kann mit 4 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 2 Byte je Zahl aus. *Tinyint* benötigt 1 Byte für die Abdeckung des Bereiches 0–255.

Kategorie	Datentyp	Beschreibung
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 4 Byte benötigt. Ist dieser Wertebereich zu gering, müssen Sie *money* mit einem Speicherbedarf von 8 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.
Text und Image	text image	BLOBs (Binary Large Objects) zur Speicherung von Daten bis zu 2 GB Größe. *Text* kann beim SQL Server 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 neue Datentyp *varbinary(max)* kann bis zu 2 ^ 31–1 Bytes Daten aufnehmen.
XML	xml	In diesem neuen 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 an den Inhalt an. So kann zum Beispiel mit einer unter diesem Datentyp gespeicherter Zahl sofort gerechnet werden, was bei einem Charakter-Datentyp nicht möglich ist.

Tabelle 3.2: SQL Server-Datentypen

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 und dürfen nicht leer sein.

Legen Sie die Felder gemäß der nachfolgenden Grafik an.

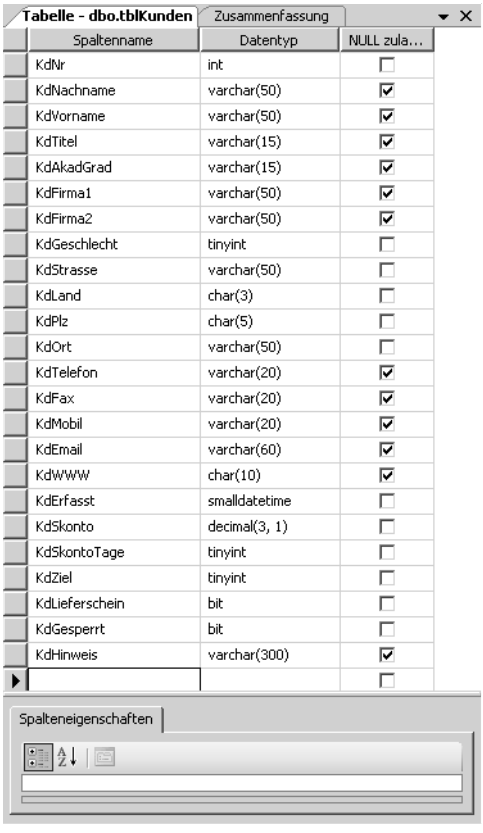

Abbildung 3.15: Kundentabelle

Speichern Sie die Tabelle schon 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 weitere Eigenschaften für einzelne Spalten der Tabelle festgelegt werden. Da das SQL Server Management Studio das grafische User-Interface des Visual Studios verwendet, können auch hier die Eigenschaften entweder nach Kategorie oder alphabetisch sortiert angezeigt werden. Die nachfolgende Abbildung zeigt die Darstellung nach Kategorie. 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 Plus und Minus eingeklappt und expandiert werden.

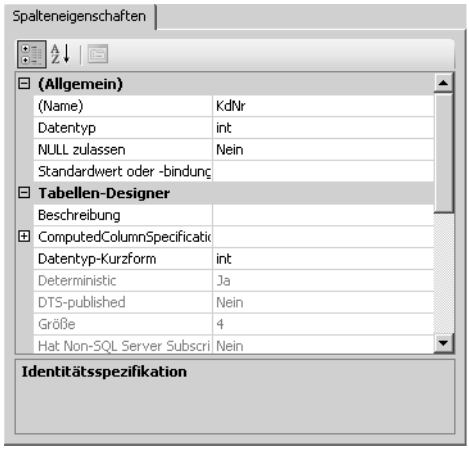

Abbildung 3.16: Spalteneigenschaften

Wenn Sie möchten, können Sie den Datentyp auch in den Spalteneigenschaften eingeben.

Folgende Eigenschaftseinstellungen können für Spalten vorgenommen werden:

- *Beschreibung*: Dieser Erläuterungstext dient der eigenen 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`. (Unter *Sonstiges* sind zum Beispiel Vereine, öffentliche Einrichtungen und Ähnliches gemeint.)
- *ComputedColumnSpecification*: Der SQL Server unterstützt berechnete Spalten in Tabellen. Über die Eingabe einer Formel werden hier direkt in der Tabelle die berechneten Werte 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 so genannte *Identität* (Identity) definiert werden. Für diese werden zusätzlich ein *Startwert* und eine *Schrittweite* definiert. Standardmäßig sind diese beiden mit 1 vorbelegt. Beim Einfügen von Datensätzen wird eine Identitätsspalte ausgehend vom Startwert automatisch befüllt. In eine als Identität festgelegte Spalte können manuell keine Werte eingetragen werden. Verwenden Sie diese Eigenschaft immer, wenn Sie eine fortlaufende Nummerierung benötigen. In der Regel wird diese Eigenschaft für Primärschlüsselspalten verwendet.

- *Volltextspezifikation*: Ist die Volltextindizierung für die Datenbank aktiviert, kann über diese Eigenschaft festgelegt werden, ob diese Spalte volltextindiziert werden soll.
- *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 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.
- *Sortierreihenfolge*: Diese Eigenschaft ist nur für Spalten mit Charakter-Datentypen verfügbar. So wie beim Anlegen der Datenbank eine Sortierreihenfolge festgelegt worden ist, kann diese auch für jede einzelne Spalte mit einem Charakter-Datentyp festgelegt werden. In der Praxis wird man hier jedoch die Standardeinstellung übernehmen. Es macht nur in Ausnahmefällen Sinn, für einzelne Spalten innerhalb einer Datenbank unterschiedliche Sortierreihenfolgen (Collationen) einzustellen.

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 fix 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 an der Oberfläche keine Möglichkeit, einen solchen Namen einzugeben. Dass Constraints wirklich eigene Objekte sind, gehrt in der grafischen Oberfläche durch die Integration der Erstellung ab. Deutlicher wird dies bei der Erstellung von Tabellen über SQL-Anweisungen.

Primärschlüssel

Pro Tabelle kann es nur einen Primärschlüssel geben, der allerdings auch aus mehreren Spalten bestehen kann. 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 Sie im Kontextmenü den Befehl PRIMÄRSCHLÜSSEL FESTLEGEN.

Abbildung 3.17: 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. Wie vom Enterprise Manager der Vorversionen gewohnt, wird der Primärschlüssel durch ein Schlüsselsymbol optisch hervorgehoben.

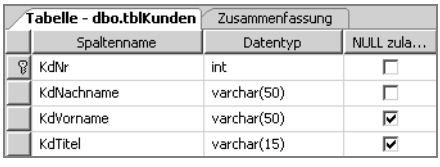

Abbildung 3.18: Primärschlüssel

Gültigkeitsregel

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 Inhalte anderer Tabellen zugreifen. Folgende Regeln lassen sich zum Beispiel 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 zumindest einer von zwei definierten Spalten muss ein Eintrag vorgenommen werden.

Nicht realisierbar sind Aufgabenstellungen wie folgende:

- 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 z.B.
	`KdGeschlecht > 0 AND KdGeschlecht < 6`
	oder z.B.
	`KdGeschlecht IN(1, 2, 3, 4, 5)`

Das Skonto darf nicht negativ sein und nicht über 5% liegen.	`KdSkonto BETWEEN 0 AND 5`
	oder z.B.
	`KdSkonto >= 0 AND KdSkonto <= 5`

Die Skontotage dürfen nicht negativ sein und maximal 30 Tage ausmachen.	`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 auch ausgefüllt sein.	`(KdGeschlecht <= 3 AND KdNachname IS NOT NULL AND KdVorname IS NOT NULL) OR (KdGeschlecht >= 4 AND KdFirma1 IS NOT NULL)`

Tabelle 3.3: Check-Einschränkungen

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

1. Klicken Sie (irgendwo) im Tabellen-Designer in den Tabellenentwurf und wählen im Kontextmenü den Befehl CHECK-EINSCHRÄNKUNGEN… aus.
2. Im Dialog klicken Sie 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.

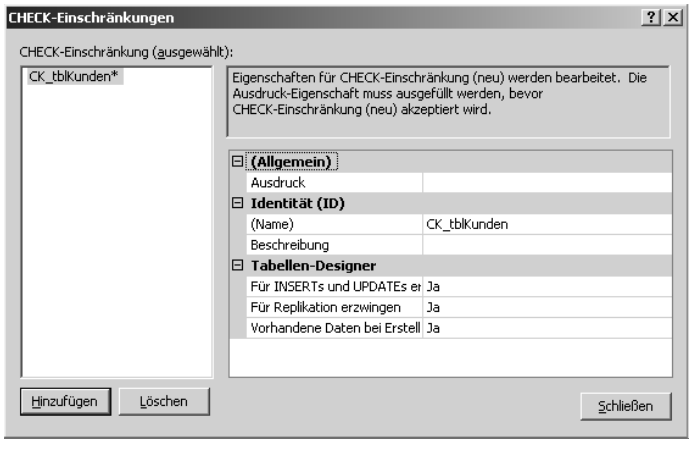

Abbildung 3.19: Neue Check-Einschränkung

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.

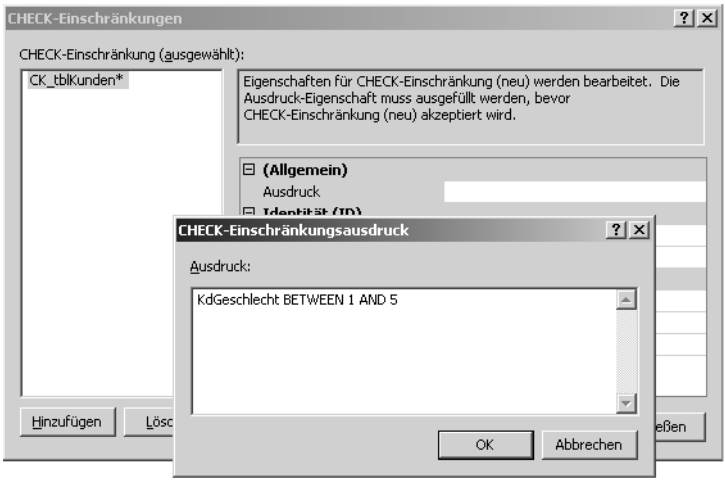

Abbildung 3.20: Einschränkungsausdruck eintragen

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

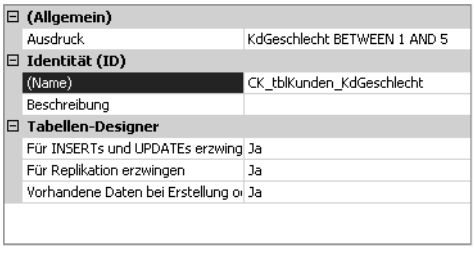

Abbildung 3.21: Einschränkungsname festlegen

Standardmäßig ist die Eigenschaft *Vorhandene Daten bei Erstellung oder Reaktivierung überprüfen* aktiviert. Sind in der Tabelle nun bereits Daten enthalten, die der Regel nicht entsprechen, kann die Einschränkung nicht erstellt werden. In diesem Fall stellen Sie diese Eigenschaft auf *Nein*. Bei Zeiten sollten Sie jedoch diese Daten dann auch in Ordnung bringen.

Ergänzen Sie die übrigen in Tabelle 3.3 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 selber nicht erfassen, solange die Spaltennamen keine Leer- und 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 3.13 dargestellt sind, erstellen zu können, müssen Sie vorerst noch die zwei weiteren 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 *InText* gespeichert werden.

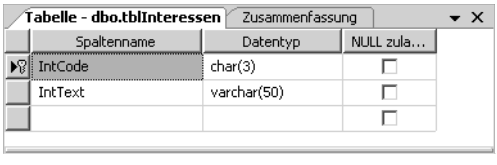

Abbildung 3.22: Tabelle tblInteressen

Die Tabelle *tblKundenInteressen* dient der 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 Kundentabelle, die Spalte *IntCode* als Fremdschlüsselspalte für die Interessentabelle. Beide gemeinsam werden als zusammengesetzter Primärschlüssel für diese Tabelle definiert. Da der Primärschlüssel ja eindeutig sein muss, ist dadurch ausgeschlossen, dass einem Kunden ein Interesse mehrmals zugeordnet wird.

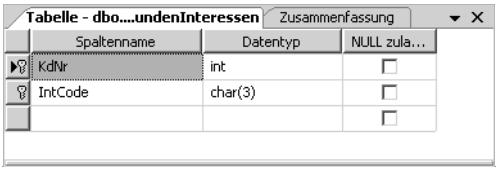

Abbildung 3.23: Tabelle tblKundenInteressen

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 wir, wie in jeder Zeile eine Kundennummer einem Interessenscode zugewiesen ist.

Abbildung 3.24: Daten der Tabelle tblKundeninteressen

Wenn 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 per se einen Eintrag enthalten. Wenn Sie dies möchten – was in der Praxis oft der Fall ist –, müssen Sie die Fremdschlüsselspalte extra noch als NOT NULL definieren.

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

- Die Anzahl und Reihenfolge der Spalten von Fremdschlüssel und referenziertem Primärschlüssel müssen 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 vorerst einfacher und übersichtlicher, wenn sie es sind.

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. Diese Variante lernen Sie am Ende dieses Kapitels 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, wählen Sie die Tabelle im Objekt-Explorer aus und wählen im Kontextmenü den Befehl ÄNDERN. Gehen Sie danach nach folgenden Schritten vor:

1. Über das Kontextmenü wählen Sie im Tabellenentwurf den Befehl BEZIEHUNGEN… aus. Im Dialog FREMDSCHLÜSSELBEZIEHUNGEN klicken Sie 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.

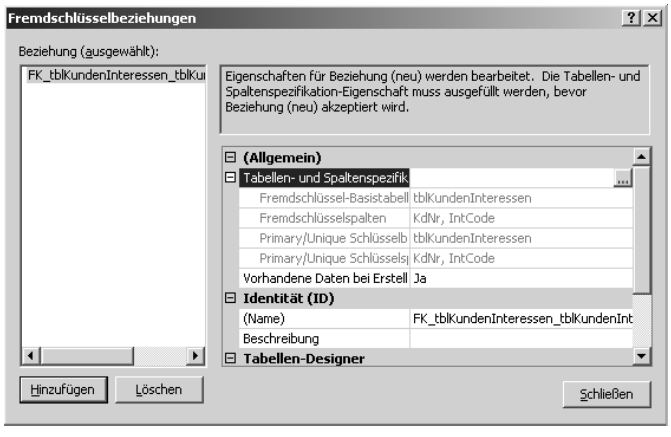

Abbildung 3.25: Neue Fremdschlüsselbeziehung

2. Im Dialog *Tabellen und Spalten* wählen Sie die Tabelle *tblKunden* als Primärschlüsseltabelle aus. Der Beziehungsname passt sich sofort an diese Änderung an. Diesen sollten Sie dann auch so belassen, weil der vorgeschlagene Name den allgemeinen Namenskonventionen für Einschränkungen entspricht. Stellen Sie die Namen für die Beziehung jeweils in den Spalten *KdNr* ein.

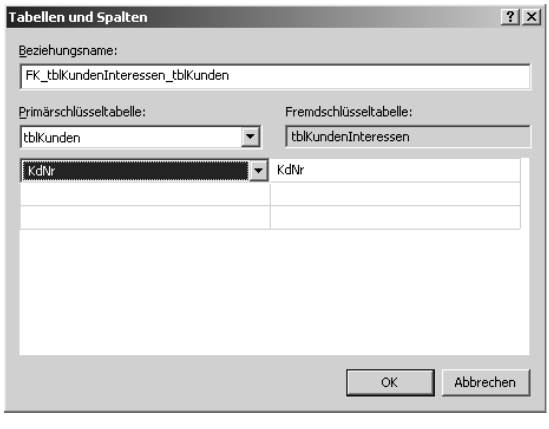

Abbildung 3.26: Tabellen und Spalten auswählen

Im Handling 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 hinten in der Liste vorfinden.

Bestätigen Sie Ihre Eingaben mit *OK*.

3. Wenn 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 *Überlappend*.

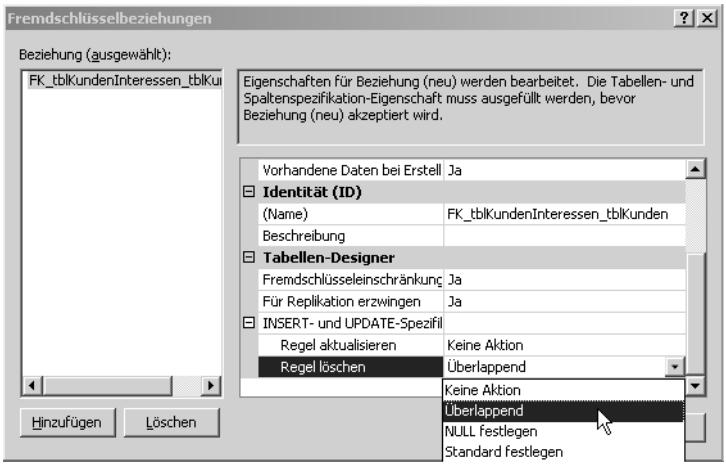

Abbildung 3.27: Änderungs- und Löschweitergabe festlegen

4. 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 anderer Ausdruck für die Beziehung zwischen zwei Tabellen ist die *referenzielle Integrität*. Deshalb wird ein Foreign Key-Constraint auch als *Referential Integrity-Constraint* bezeichnet.

Die referenzielle Integrität macht eine Beziehung zwischen Tabellen erst zu dem, was sie ausmacht. 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. Umgelegt auf unser Beispiel bedeutet dies, dass Sie keinen Kunden aus der Tabelle *tblKunden* löschen können, wenn diesem Kunden in der Tabelle *tblKundenInteressen* Interessensgebiete zugeordnet sind.
- Die referenzielle Integrität verhindert, dass der *Inhalt der Primärschlüsselspalte der Mastertabelle geändert* wird, wenn für diesen Datensatz Einträge in der Detailtabelle existieren. (Dies bedeutet, dass Sie keinem Kunden, der Interessen zugeordnet hat, eine neue Kundennummer geben können.)
- Die referenzielle Integrität erzwingt, dass in die Fremdschlüsselspalte der Detailtabelle *nur Einträge, die in der Mastertabelle auch 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 mitgelöscht. In unserem Beispiel würde dies bedeuten, dass, wenn ein Kunde gelöscht wird, auch alle seine Interessenzuordnungen gelöscht werden. Dieses Verhalten wird durch die Wahl des Eintrages *Überlappend (Cascade)* erreicht. (Ich kann es mir einfach nicht verkneifen, an dieser Stelle zu erwähnen, dass die Übersetzung für die deutsche Version hier mehr als unglücklich ist.)

In dieser Version sind zwei neue Einstellungsvarianten dazugekommen:

- *NULL festlegen (Set NULL)*: Wird eine Eintrag in der Mastertabelle gelöscht, werden Detaildatensätze zwar nicht gelöscht, aber der Inhalt der Fremdschlüsselspalte geleert. Dies ist allerdings nur möglich, wenn diese 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.

- *Änderungsweitergabe (Regel ändern)*: Die Änderungsweitergabe bewirkt, 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 mit geändert. Dadurch gehören auch nach der Änderung noch dieselben Datensätze zusammen.

  Wie beim Löschen stehen auch hier neu die Optionen *NULL festlegen* und *Standard festlegen* zur Auswahl. Deren Bedeutung ist analog zu sehen.

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 ausarten kann.

Eindeutiger Schlüssel

Den eindeutigen Schlüssel haben wir uns bis zum Schluss aufgehoben, da er wie ein Index anzulegen ist. Dies wird im darauf folgenden Punkt erläutert.

Ein *Eindeutiger Schlüssel* (Unique Key) unterscheidet sich von einem Primärschlüssel durch folgende zwei Punkte:

- NULL-Werte sind im eindeutigen Schlüssel zugelassen.
- 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, 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 durchsuchen. 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 oder nicht. Die Tatsache, dass Sie ihm einen Index für den Suchvorgang zur Verfügung stellen, bedeutet noch lange nicht, dass der Server diesen auch verwendet. (Wir haben uns bei der Erstellung des Index für dieses Buch auch sehr bemüht, haben jedoch keinen Einfluss darauf, ob Sie ihn nutzen oder nicht.)

Nach folgenden Kriterien sollten Sie zu indizierende Spalten auswählen:

- Diese Spalte wird *häufig als Suchkriterium* oder als Verknüpfungskriterium in Abfragen verwendet.
- Die *erwartete Trefferquote* bei der 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 unserem 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, denn 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 das nur für dickere 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 Nachname und Vorname 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 zu indizieren. 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 diesem nochmals gespeichert werden.
- Bei jeder Datenänderung müssen davon betroffene Indizes aktualisiert werden, was wiederum Systemressourcen belegt.

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 untersten 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 schon der Datensatz selber. Der Index ist demnach eng mit der Speicherstruktur der Tabelle verwoben. Dieser Index ist daher in Summe 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, wenn es für diese Tabelle nicht schon einen gruppierten Index gibt.

Erstellen eines Index

Einen Index haben wir indirekt schon 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 ist, hat er auch einen Namen. Der Name des für den Primärschlüssel angelegten Index entspricht jenem für den Primärschlüssel selbst.

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. Über das Kontextmenü wählen Sie 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 von den vorigen Beispielen bekannt, mit Standardwerten angezeigt.

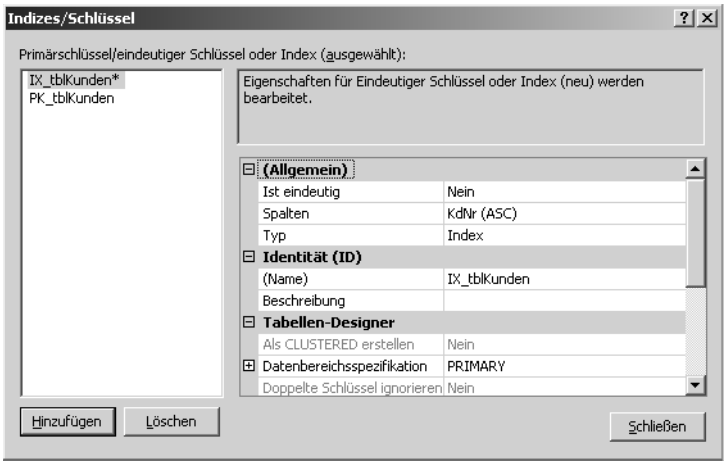

Abbildung 3.28: Neuen Index anlegen

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 *KdNachname* aus. Belassen Sie die Sortierreihenfolge auf *Aufsteigend* (ASC = ascending) eingestellt.

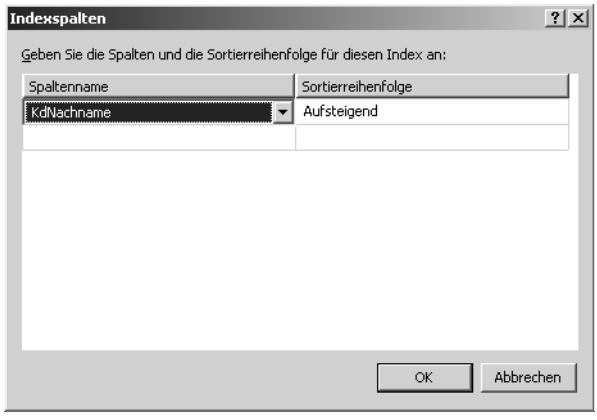

Abbildung 3.29: Indexspalten auswählen

In diesem Beispiel belassen wir aber die Einstellung *Index*.

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

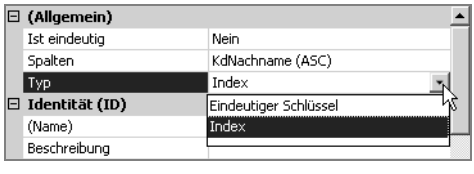

Abbildung 3.30: Index oder eindeutiger Schlüssel

4. Bauen Sie den Namen der indizierten Spalte in den Indexnamen ein.

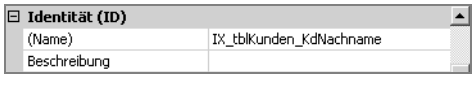

Abbildung 3.31: Indexname

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

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

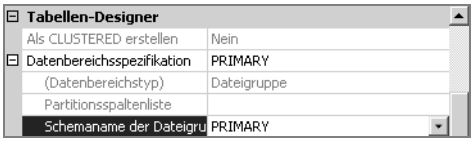

Abbildung 3.32: Dateigruppe für Index

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

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

Noch ein Tipp zum Abschluss: Die in den vorigen Abschnitten verwendeten Dialoge können nicht nur über das Kontextmenü, sondern auch über die Symbolleiste *Tabellen-Designer* geöffnet werden.

Abbildung 3.33: 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

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 doch, um ebenso wie jetzt „quick and dirty" in Tabellen zu schreiben.

Um Daten zu erfassen, öffnen Sie die Tabelle über den Befehl TABELLE ÖFFNEN des Kontextmenüs. 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 3.34: 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 und 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 von weiteren Datensätzen abbrechen, wenn Sie eine sehr große Tabelle geöffnet haben und es Ihnen zu lange dauert, bis alle Datensätze in den Speicher geladen sind.

Abbildung 3.35: 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. (Die Tools der Vorgängerversionen haben für diese Kommunikation noch ODBC verwendet, weshalb die Fehlermeldungen anders aufgebaut gewesen sind.)

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

Erfassen Sie 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 absichtlich solche Fehler ein, um mit dem Umgang mit Fehlermeldungen und deren Interpretation vertraut zu werden.

Ein kleiner Unterschied zu den Editoren der Vorversionen birgt noch eine Fehlerquelle bei der Eingabe: 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 nun statt 1 und 0 abweichend davon *True* und *False* eingegeben werden. Sonst erhalten Sie auch hierbei einen Fehler.

3.3 Datenbankdiagramme einsetzen

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

Ein wichtiger Hinweis für Benutzer der Vorversion: Die Datenbankdiagramme der Vorversionen sind nicht kompatibel und können nicht übernommen werden. Sie müssen leider neu erstellt 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.

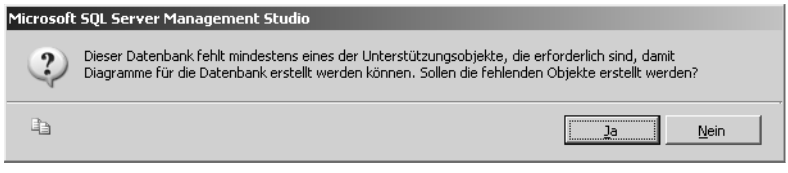

Abbildung 3.36: Unterstützungsobjekte für Datenbankdiagramme anlegen

Unter folgenden Umständen tritt dabei jedoch ein Fehler auf:

- Der *Kompatibilitätsgrad* der Datenbank ist nicht auf SQL Server 2005 eingestellt. Öffnen Sie in diesem Fall die Datenbankeigenschaften über das Kontextmenü, und wechseln Sie auf die Seite *Optionen*. Stellen Sie dort den Kompatibilitätsgrad der Datenbank auf *SQL Server 2005 (90)*.
- Die Datenbank hat *keinen gültigen Eigentümer*. Dies ist meist vor allem 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, und tragen Sie diesen als Besitzer der Datenbank auf der Seite *Dateien* der Datenbankeigenschaften ein. Wenn Sie Informationen zum Anlegen eines Anmeldenamens benötigen, finden Sie diese in Kapitel 9.

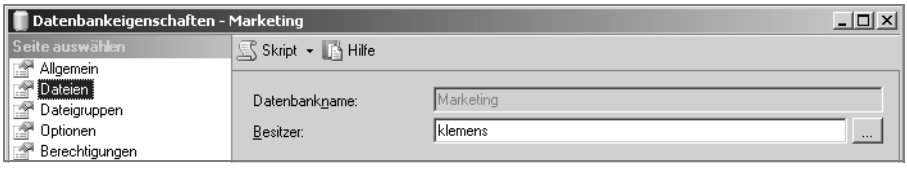

Abbildung 3.37: Zuweisen des Datenbankbesitzers

Haben Sie die Unterstützungsobjekte hinzugefügt, können Sie nun 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.

Im Dialog *Tabelle hinzufügen* wählen Sie jene Tabellen aus, die Sie im Diagramm darstellen möchten.

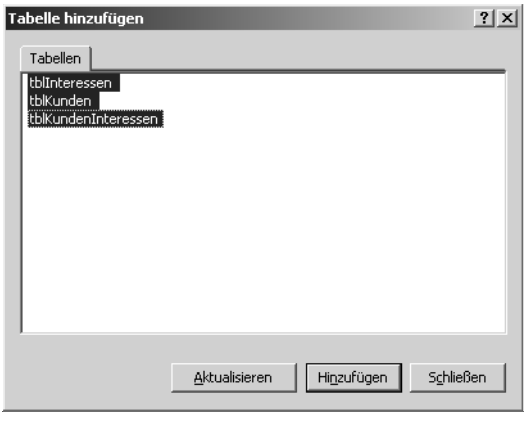

Abbildung 3.38: Tabellen für Datenbankdiagramm auswählen

Bestehende Beziehungen werden im Datenbankdiagramm bereits 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.

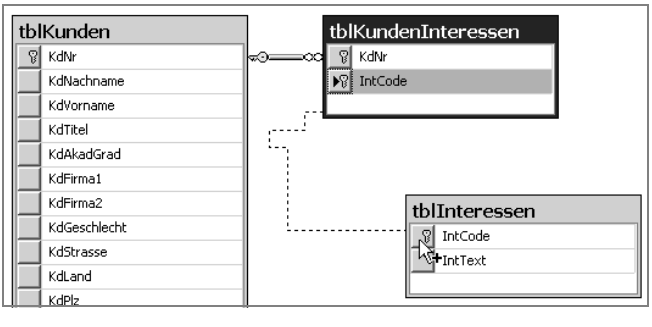

Abbildung 3.39: Beziehung per Drag & Drop erzeugen

Nach dem Loslassen der Maustaste öffnet sich derselbe Dialog, den Sie vom Tabellen-Designer 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.

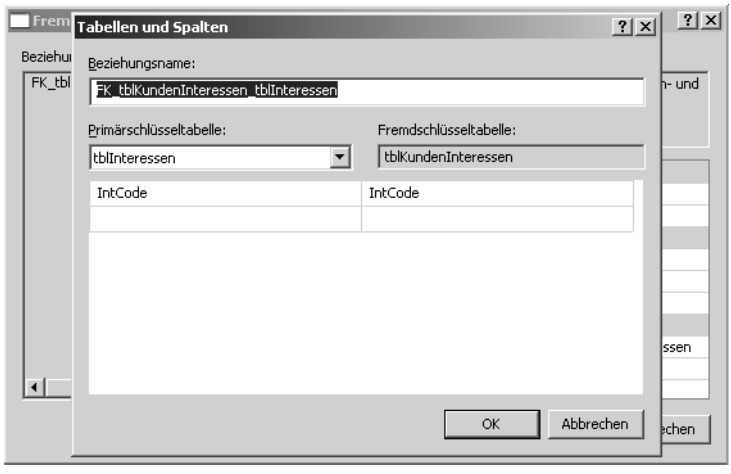

Abbildung 3.40: Beziehung erstellen

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 vorbereiten zu können

Abbildung 3.41 zeigt ein Datenbankdiagramm der Beispieldatenbank WAWI mit eingeblendeten Seitenumbrüchen.

Außerdem können Sie direkt in einem Diagramm neue Tabellen anlegen und auch 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, Sie können Tabellen auch in mehreren Diagrammen anzeigen. In der Praxis hat es sich bewährt, nicht nur ein Diagramm zu erstellen, in dem alle Zusammenhänge dargestellt werden. 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 übernommen.

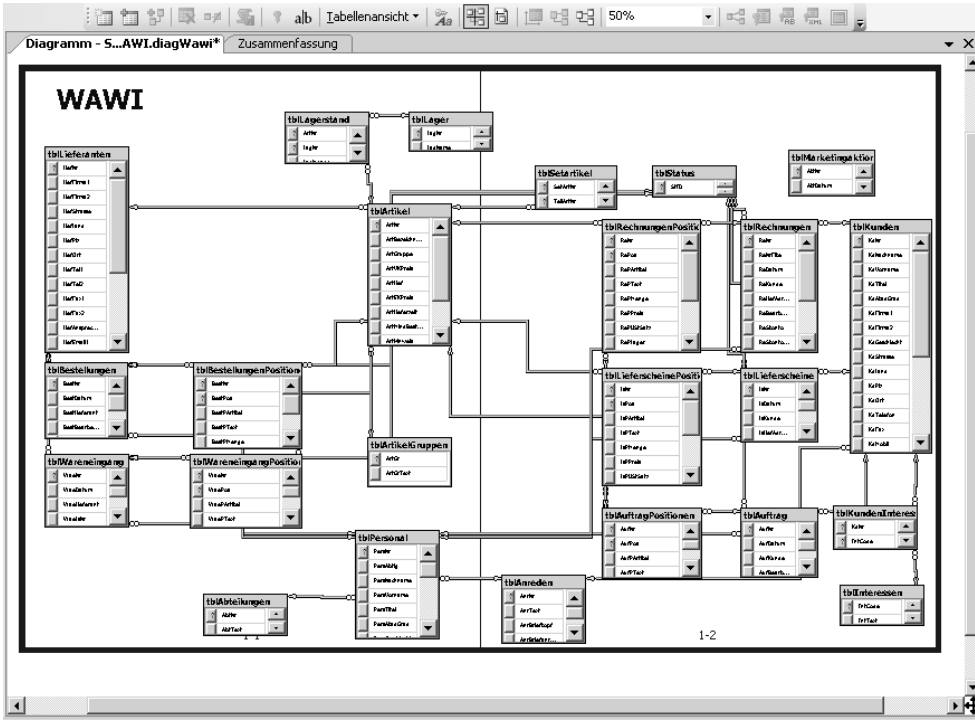

Abbildung 3.41: Datenbankdiagramm mit Seitenumbrüchen

3.4 Was Sie noch wissen sollten…

Zum Abschluss des Kapitels möchten wir noch drei Aspekte beleuchten, die im praktischen Arbeiten mit Datenbanken von Bedeutung sein können.

3.4.1 Tabellen in anderen Dateigruppen speichern

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

Wie können Sie beim Anlegen einer Tabelle festlegen, in welcher Dateigruppe sie angelegt werden soll?

Wenn Sie keine spezielle Dateigruppe angeben, wird das Objekt immer in der Dateigruppe PRIMARY erstellt.

Öffnen Sie beim Erstellen der Tabelle im Tabellen-Designer das Eigenschaftsfenster über das Menü ANSICHT oder die Taste F4. In der Rubrik *Reguläre Datenbereichsspezifikationen* wählen Sie die Dateigruppe in der Liste bei der Eigenschaft *Schemaname der Dateigruppe* aus.

Abbildung 3.42: Tabelleneigenschaften

Beim Speichern wird die Tabelle in der gewählten Dateigruppe angelegt.

3.4.2 Tabellen direkt mit DDL-Anweisungen erstellen

Alternativ zum Tabellen-Designer können Sie eine Tabelle in einem Abfrage-Editorfenster direkt über *DDL* (*Data Definition Language*, Teilbereich der Sprache SQL) Anweisungen erstellen.

Ein einfaches Beispiel soll Ihnen einen kleinen Einblick in dieses Thema geben. Dies kann keine umfassende Abhandlung sein, denn diese würde den Rahmen sprengen. Ziel ist es, 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 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 smalldatetime 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 `ALTER TABLE`-Anweisung. Das nachfolgende Beispiel ergänzt eine Spalte für das Austrittsdatum.

```
ALTER TABLE dbo.tblPersonal
ADD PersAustritt smalldatetime
```

Auch Einschränkungen können für eine Tabelle nachträglich ergänzt werden. Diese Anweisung erzeugt eine Beziehung zwischen der Personaltabelle und der Abteilungstabelle.

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

3.4.3 Gefahren der grafischen Oberfläche

Grafische Oberflächen erleichtern zwar die Arbeit ungemein und sind in der Regel sehr komfortabel. Aber sie können auch zum Fluch werden, wenn man versteckte Gefahren nicht kennt.

Viele Dinge, die über die grafische Oberfläche möglich sind, werden direkt 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 folgendes Beispiel veranschaulichen:

Neue Spalten können an eine Tabelle nur am Ende angefügt 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 zwar 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 Vorgänge unbedingt zu Zeiten durchführen, in denen nicht in der Datenbank gearbeitet wird.

Wie kann man solche Vorgänge von ungefährlichen 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, müssen die Alarmglocken läuten. Die nachfolgende Meldung erscheint, wenn in der Artikeltabelle der Beispieldatenbank WAWI die Reihenfolge von Spalten verändert wird. Warum müssen so viele Tabellen gespeichert werden, wenn nur Änderungen an einer Tabelle vorgenommen worden sind? – Weil alle auf diese Tabelle verweisenden Fremdschlüssel vorübergehend gelöscht werden müssen, da sonst die alte Artikeltabelle nicht gelöscht werden könnte.

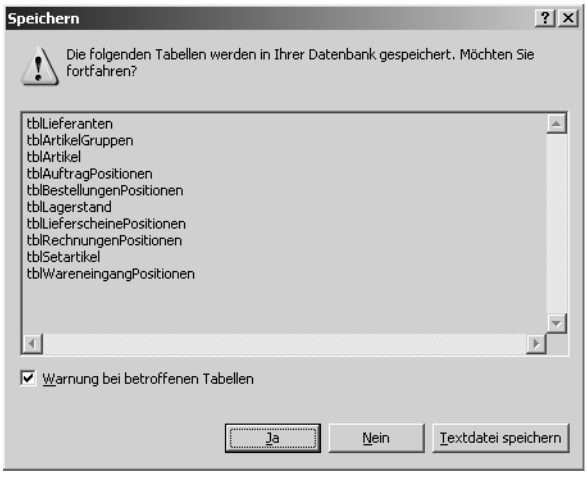

Abbildung 3.43: Abfrage beim Speichern

- *Analyse des Änderungsskripts*: Wenn Sie Ihre Änderungen fertig eingegeben, aber noch nicht gespeichert haben, können Sie sich das Änderungsskript, das alle ausgeführten Anweisungen enthält, anzeigen lassen. Wählen Sie im Kontextmenü den Befehl ÄNDERUNGSSKRIPT GENERIEREN…, oder klicken Sie dazu auf das entsprechende Symbol. Wenn 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 3.44: Änderungsskript

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.