Kapitel 4

Das Arbeitsbeispiel dieses Buchs

In diesem Kapitel:

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Wir haben uns für dieses Buch vorgenommen, Ihnen das Entwickeln mit SQL Server 2008 nicht trocken theoretisch, sondern anhand vieler praktischer Beispiele nahezubringen. Damit Sie beim Lesen die Demos und Codebeispiele sofort ausprobieren können, gibt es auf der CD zum Buch eine Beispieldatenbank, die in allen fünf Teilen des Buchs eine Rolle spielen wird. Von den Transact SQL-Beispielen, über die .NET-Programmierung bis hin zu den nicht relationalen Programmiermethoden: Sie können den Programmcode sofort laufen lassen, mit den Beschreibungen in den Kapiteln vergleichen und natürlich eigene Experimente anstellen. Die einzelnen Kapitel bauen, was die Demodaten angeht, nicht aufeinander auf. Sie müssen beim Lesen und Experimentieren keine feste Reihenfolge einhalten und können jederzeit aus der Datensicherung den ursprünglichen Zustand der Datenbank wiederherstellen, falls mal ein Unglück passiert ist.

Ganz bewusst haben wir uns dafür entschieden, *nicht* die Standard-Beispieldatenbank von SQL Server – AdventureWorks – zu verwenden. Diese ist sicher clever und vorbildlich aufgebaut. Für unsere Zwecke aber etwas zu komplex. Damit man aus den Beispielen des Buchs schnell Nutzen ziehen kann, sollte man sich möglichst schnell in das Datenmodell hineindenken können und deswegen gibt es bei uns zur Darstellung eines Kunden nur eine Tabelle und nicht gleich ein halbes Dutzend. Manchmal ist small eben doch beautiful. Ein Hinweis für die SQL Server-»Veteranen« in der Leserschaft. Falls Ihnen der ein oder andere Aspekt unserer *netShop*-Datenbank aus der guten alten Northwind-Datenbank[1] bekannt vorkommt, so ist das kein Zufall. Aufgabenstellung und Struktur der Datenbank sind ähnlich. Wir haben unsere Datenbank allerdings mit ein paar zusätzlichen Tabellen versehen, eine Menge Daten hineingepumpt und mit den neuen Möglichkeiten von SQL Server realisiert. Viel Spaß beim Ausprobieren!

Übersicht über die netShop-Datenbank

Eine neue Technologie im luftleeren Raum kennen zu lernen (oder ein Buch darüber zu schreiben), ist schwierig. Daher haben wir uns entschlossen, eine Beispieldatenbank für ein konkretes Anwendungsszenario zu entwickeln. Dessen Hintergrund und die daraus resultierende Datenbanklösung werden in den nächsten Abschnitten kurz beschrieben. Sie können den Text und die Diagramme als Referenz benutzen, wenn Sie die Demos in den verschiedenen Kapiteln bearbeiten.

Die Aufgabenstellung im Demo-Szenario

Die netShop-Datenbank soll die Datenhaltung für einen Webshop erledigen. Es geht um Artikel, Kunden und Bestellungen. Das liefert ein gutes Beispiel für ein übersichtliches Datenbankschema, da eine Webshop-Software in der Regel nicht ein komplettes Warenwirtschaftssystem abbildet, sondern häufig als Frontend für ein bestehendes ERP-System, wie SAP, Dynamics AX oder individuelle Geschäftslösungen eingesetzt wird. Die Datenbank bildet daher nur das ab, was mit der direkten Implementierung des Shops zu tun hat. Das macht die Aufgabenstellung übersichtlicher – dennoch sind die Anforderungen komplex genug, um alle Aspekte der Datenbankentwicklung mit SQL Server 2008 exemplarisch erörtern zu können: Es steckt wirklich viel drin.

Bei der Verwendung als »Store Frontend« müssen unter anderem die Daten des Webshops mit dem ERP-System synchronisiert werden. Dazu können die SQL Server-Integration Services verwendet werden, oder auch Webservices implementiert werden. Soll in kleineren Umgebungen der Webshop »stand alone« betrieben werden können, benötigt er eine Management-Oberfläche. Da kommt die Entwicklung eines .NET-Clients ins Spiel. Für geographische Auswertungen der Verkäufe bietet sich der neue Datentyp *geography* an. Und so weiter.

[1] Den legendären Kunden „ALFKI“ (Alfreds Futterkiste) aus Northwind konnten wir aber leider nicht reanimieren. Er ruhe in Frieden!

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Da ganz sicher keine zwei Implementierungen eines Shopsystems einander gleichen, soll das Datenbankschema unterschiedliche Arten der Verwendung zulassen. Zum Beispiel könnte auf der Grundlage der Struktur ein Webshop gebaut werden, bei dem der Kunde nach einer Anmeldung seine Kundendaten selbst pflegen kann. Es ist aber auch denkbar, dass ohne ein Kunden-Login direkt bestellt werden kann. Es ist möglich, dass bestimmte Tabellen für eine konkrete Umsetzung benutzt werden, andere wiederum nicht. Es ist auch denkbar, dass weitere Tabellen hinzugefügt werden müssen. Die netShop-Datenbank ist als kleinster gemeinsamer Nenner gedacht.

Installation der Beispieldatenbank(en)

Für die Installation der netShop-Beispieldatenbank benötigen Sie auf Ihrer Festplatte ca. 500 MByte für die SQL Server Datenbankdateien und noch einmal ca. 300 MByte für das Entpacken der Datensicherung – insgesamt also ungefähr 800 MByte. Für die Demos zum Thema Indizierung in Kapitel 12 (»Indizierung und Partitionierung«) können Sie bei dieser Gelegenheit noch die Beispieldatenbank *PerformanceDB* einspielen. Da es in dieser um »ordentliche« Datenmengen geht, müssen Sie für die installierte Datenbank etwa 3,5 GByte Plattenplatz einplanen. Für das Auspacken der Backup-Datei werden zusätzlich noch einmal ca. 3,5 GByte benötigt.

Da das Einspielen der beiden Datenbanken nahezu identisch verläuft, beschreibe ich an dieser Stelle nur den Umgang mit *netShop*.

Hier also die Schritt-für-Schritt-Anleitung für das Einrichten der netShop-Demodatenbank. Als Vorbereitung sollten zunächst einmal die Demodateien mithilfe des Tools von der Buch-CD auf die Festplatte kopiert werden. Das Standardverzeichnis *C:\SQLEntwicklerbuch2008* können Sie beliebig ändern, Sie müssen das nur bei den nachfolgenden Schritten berücksichtigen.

1. Packen Sie die Datei *netShop.zip*, die sich im Verzeichnis *C:\SQLEntwicklerbuch2008\Datenbanken* befindet, aus. Das Verzeichnis enthält jetzt die Dateien *netShop.bak* und *netShopRestore.sql*. Wenn es der Platz auf Ihrem PC hergibt, dann lassen Sie am besten die Backupdatei in diesem Verzeichnis stehen. So können Sie später jederzeit den Ausgangszustand der Demodaten rekonstruieren.
2. Die Wiederherstellung der netShop-Datenbank können Sie über das GUI des Management Studio mit dem entsprechenden Dialogfeld oder per T-SQL-Skript ausführen. Kapitel 21 (»Administration für Entwickler«) zeigt, wie ein Wiederherstellungsvorgang über die Benutzeroberfläche en detail abläuft. Schlagen Sie dieses auf, wenn Sie sich über die Feinheiten informieren möchten. Falls Sie den SQL Server bis jetzt noch gar nicht installiert haben, liefert dieses Kapitel die entsprechende Anleitung dazu. Im Moment kommen Sie aber ohne diese Informationen aus: Das Skript *netShopRestore.sql* lassen Sie im T-SQL-Editor des Management Studios ablaufen. Die ausführliche Beschreibung des Editors finden Sie in Kapitel 6 (»Werkzeuge für T-SQL-Entwickler«). Hier folgt aber jetzt gleich eine Schnellanleitung, damit Sie sofort durchstarten können und keine Zeit mit Blättern vergeuden.
3. Starten Sie das SQL Server Management Studio: *Start / Alle Programme / Microsoft SQL Server 2008 R2 / SQL Server Management Studio.*
4. Verbinden Sie sich mit Ihrem SQL Server 2008 R2. Verwenden Sie dazu die *Windows-Authentifizierung,* wenn Sie SQL Server mit den Standardvorgaben installiert haben (siehe Abbildung 4.1). Wahlweise können Sie natürlich mit einem anderen, von Ihnen festgelegten, SQL Server-Konto arbeiten.
5. Klicken Sie in der Standard-Symbolleiste des Management Studio auf *Neue Abfrage*, um ein neues T-SQL-Abfragefenster zu erzeugen.

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

6. Der Menübefehl *Datei / Öffnen / Datei…* ermöglicht Ihnen nun, die Skriptdatei *netShopRestore.sql* aus dem Verzeichnis *C:\SQLEntwicklerbuch2008\Datenbanken* zu laden. Das Editorfenster sieht dann so aus, wie in Abbildung 4.2 dargestellt.
7. Sie werden nun an den mit Kommentaren gekennzeichneten Stellen anstelle der Original-Dateipfade Ihre eigenen Pfade angeben müssen. Anstelle des Pfades *D:\Daten\netShop_data.mdf* geben Sie also beispielsweise den Pfad zu *Ihrem* Datenverzeichnis an (hinter dem Schlüsselwort TO). Die Dateinamen selbst sollten Sie nicht ändern. Zur *netShop*-Datenbank gehören insgesamt vier Dateien.
8. Führen Sie das Skript über das Symbol *Ausführen* aus (das ist das Symbol mit dem Ausrufezeichen). Beobachten Sie die Meldungen im Fenster unterhalb des Editors. Verläuft die Wiederherstellung erfolgreich, dann sehen Sie anschließend Zeilen in der Art:

```
RESTORE DATABASE hat erfolgreich X-tausend Seiten in Y Sekunden verarbeitet ( Z MByte/s).
Sie können jetzt mit dem netShop arbeiten!
```

9. Sie sollten jetzt Ihre angepasste Version des Wiederherstellungsskripts speichern. Es empfiehlt sich vor dem Bearbeiten jedes Kapitels, die Datenbank erneut aus der Sicherung einzuspielen, damit Sie einen einheitlichen Ausgangszustand für die Demos haben.
10. Für die Arbeit mit den Beispielen werden keine speziellen Benutzerkonten benötigt. Achten Sie darauf, in der Datenbank mit den Berechtigungen eines Datenbankbesitzers (*dbo*) zu arbeiten. Dies ist immer der Fall, wenn Sie als lokaler Administrator auf eine SQL Server-Instanz zugreifen. Sie können sich auch über das bei der SQL Server-Installation vergebene sa-Kennwort (siehe Kapitel 13 – »Sicherheit«) mit dem Server verbinden. Damit sind Sie Besitzer jeder Datenbank.

HINWEIS Beachten Sie, dass das Wiederherstellung-Skript eine vorhandene *netShop*-Datenbank löscht. Falls Sie also eigene Ergänzungen zu den Beispielen retten wollen, dann sollten Sie eine Sicherung durchführen, bevor Sie das Skript laufen lassen. Sie werden Probleme mit dem Wiederherstellung-Skript bekommen, falls es aktive Verbindungen zur *netShop*-Datenbank gibt. Falls Sie also gerade mit der Datenbank gearbeitet haben, dann schließen Sie zuerst alle offenen Fenster und beenden Sie alle Clientprogramme, die auf die Datenbank zugreifen. Erst dann sind das Löschen und die anschließende Wiederherstellung möglich.

Abbildung 4.1 Anmeldung beim SQL Server

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

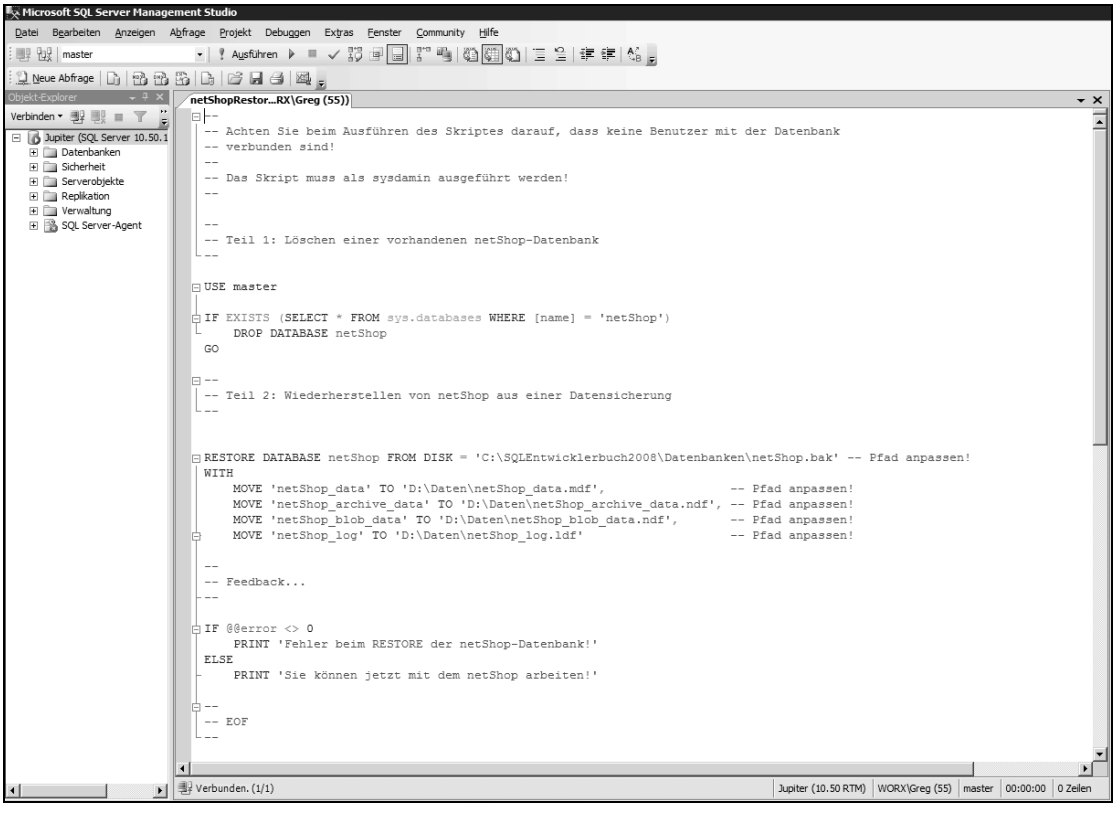

Abbildung 4.2 T-SQL-Editor mit geladenem Restore-Skript

Der Aufbau der netShop-Datenbank

Zu einer Beschreibung einer Datenbank gehört das so genannte Datenbankschema – das beschreibt den spaltenweisen Aufbau der einzelnen Tabellen und die Beziehungen zwischen diesen – sowie die Definition von Einschränkungen und Geschäftsregeln auf der Datenschicht. Für das Verständnis der Beispieldatenbank ist es zunächst einmal ausreichend, sich ein wenig im Datenbankschema zurechtzufinden. In der »Auslieferungsversion« der Demodatenbank sind zwar einige wenige Einschränkungen definiert (Spalten-Eindeutigkeit, Wertebereiche), aber keine komplexeren Geschäftsregeln; diese werden im Zuge der weiteren Implementierungen hinzugefügt.

Das Tabellenschema

Die folgende kurze Übersicht soll es Ihnen erleichtern, die Beispiele in diesem Buch besser nachvollziehen zu können. Sie können das Datenbankschema immer wieder gut als Referenz einsetzen. Bei der Beschreibung geht es an dieser Stelle nur um die Grundlagen, die Sie benötigen, um das Gesamtkonzept zu verstehen. In den verschiedenen Kapiteln des Buchs werden dann bei Bedarf weitere Details der Implementierung erläutert. Wichtig: Es gibt tausend (und eine) Art, ein Datenmodell zu einer gegebenen Aufgabenstellung zu entwickeln. Das Schema der netShop-Datenbank ist nur einer von vielen möglichen Lösungsansätzen. Manche Dinge hätte man sicher eleganter lösen können, manche Entwurfsprinzipien sind nicht durchgängig

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

eingehalten worden. In den meisten Fällen hat das damit zu tun, dass die Datenbank für die Demonstration verschiedener Techniken herhalten muss und wir für eine bestimmte SQL Server-Funktion eine passende Tabelle brauchten. Dennoch ist der Aufbau typisch für eine kleine Datenbanklösung und wurde so (oder so ähnlich) auch schon in der Praxis erprobt. Die Visio-Grafik in Abbildung 4.3 gibt eine Übersicht über die Haupttabellen der Datenbank. Diese Darstellung ist durch Reverse-Engineering aus der echten netShop-Datenbank entstanden und auf die Namen der Tabellen, Primär- und Fremdschlüssel reduziert.

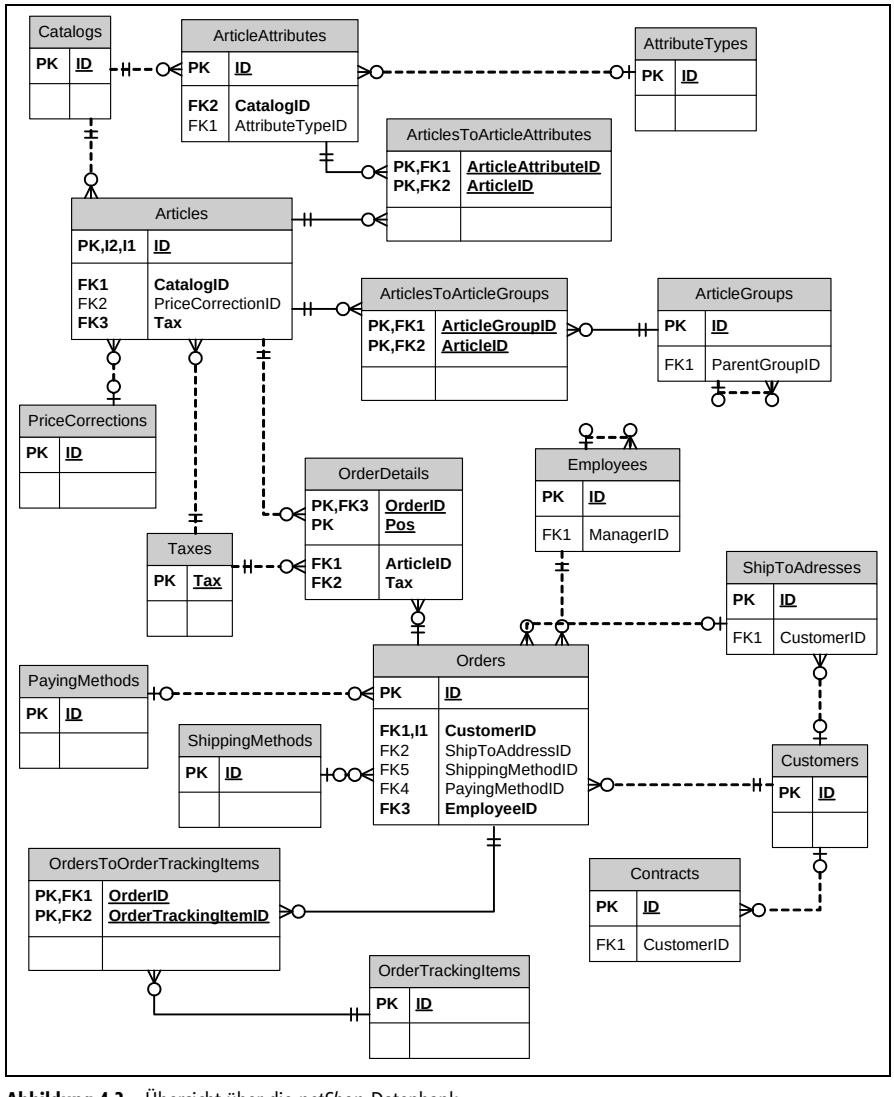

Abbildung 4.3 Übersicht über die *netShop*-Datenbank

Eine Besonderheit des Datenbankschemas ist die konsistente Verwendung künstlicher Primärschlüssel auf der 1-Seite einer 1-zu-N-Beziehung. Künstliche Schlüssel enthalten Werte, die nicht aus den Datensätzen selbst stammen, sondern automatisch erzeugt werden. Der Datentyp dieser Schlüssel ist ganzzahlig (*int*) mit der Eigenschaft *IDENTITY*. Auf diese Art hergestellte Verknüpfungen können von SQL Server besonders schnell ausgewertet werden. Für die Speicherung der *echten* Schlüssel wie Artikelnummer oder Kundennummer wird zusätzlich ein Feld *Code* vorgesehen.

Bestell- und Kundendaten

Die Datenbank lässt sich in drei Bereiche aufteilen: Bestellverwaltung, Katalogverwaltung und Kundenverwaltung. Die Speicherung der Bestellungen ist auf eine sehr direkte Art über zwei Tabellen gelöst. In der Tabelle *Orders* werden die zentralen Bestelldaten, wie Datum, Online-Bestellnummer, Kunde etc. gehalten (Abbildung 4.4). In der abhängigen Tabelle *OrderDetails* gibt es die Einzelheiten zu den Bestellpositionen. Die *Orders*-Tabelle bezieht sich auf verschiedene Referenztabellen wie *Employees* (Mitarbeiter, der die Bestellung bearbeitet), *ShippingMethods* (hinterlegte Versandarten) und *PayingMethods* (hinterlegte Bezahlarten). Einzig die Tabelle *Employees* weist eine kleine Besonderheit auf – die Tabelle referenziert sich selbst. In der Mitarbeiter-Datenbank sind die Vorgesetzten-Angestellten-Verhältnisse durch Referenzen innerhalb der Tabelle über die Spalten *ID* und *ManagerID* abgelegt.

Für die Nachverfolgung von Bestellungen eignen sich die beiden Tabellen *OrderTrackingItems* und *OrdersToOrderTrackingItems*. In *OrderTrackingItems* können Schlagworte für den Zustand einer Bestellung hinterlegt werden. Bei der ersten Installation sind hier die Werte »Eingang«, »Lieferung begonnen«, »Geliefert«, »Storniert« und »Abgeschlossen« hinterlegt. In der Verknüpfungstabelle *OrdersToOrderTrackingItems* lässt sich zu einem Schlüssel eines Datensatzes aus *Orders* (Feld *OrderID*) der entsprechende Schlüssel aus *OrderTrackingItems* (Feld *OrderTrackingItemID*) mit einem Datum hinterlegen. So lässt sich eine Bestellungsverfolgung implementieren. Soll das Tracking noch feiner sein, so kann ein Manager des Shops die Begriffe in *OrderTrackingItems* einfach erweitern.

Die zentrale Tabelle der Kundenverwaltung ist die Tabelle *Customers*. Diese ist ganz geradeaus entworfen worden und enthält keine Besonderheiten. Soll dem Kunden erlaubt werden, abweichende Lieferadressen einzugeben, dann landen diese in *ShipToAdresses*. Lieferadressen sind natürlich einerseits den Kunden zugeordnet, andererseits aber auch den Bestellungen, da die Adresse pro Bestellung ausgewählt werden kann. Ist das Feld *ShipToAddressID* in *Orders* leer, also (NULL), dann bedeutet dies, dass an die primäre Kundenadresse in *Customers* geliefert werden soll. Die Tabelle *Contracts* dient der Speicherung von Lieferverträgen (und ist vor allen Dingen ein Beispiel für die Volltextsuche – dazu gleich mehr).

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

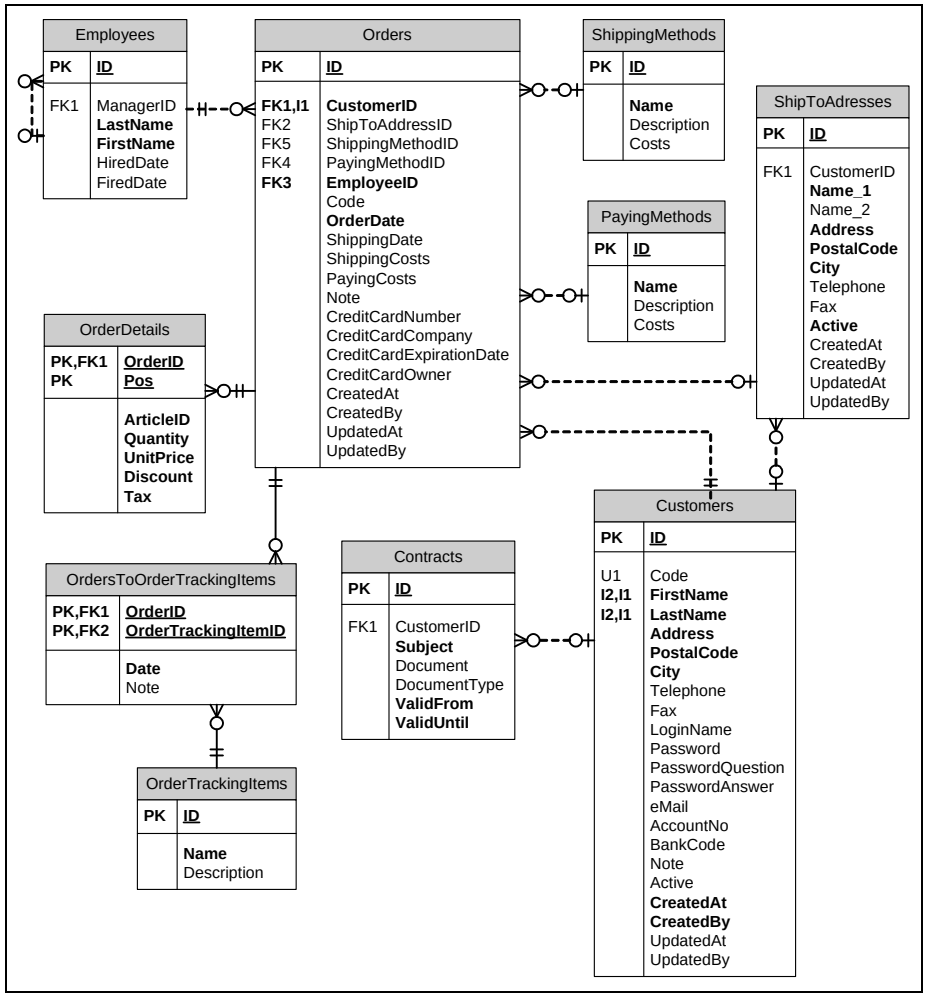

Abbildung 4.4 Tabellen für Bestellungen und Kunden

Katalogdaten

Die zentrale Tabelle für die Verwaltung der Katalogdaten ist die Artikeltabelle *Articles*. Um diese herum sind verschiedene Tabellen zur näheren Beschreibung des Warenbestandes angeordnet (Abbildung 4.5). Hier gibt es eine Konstruktion, die eine kurze Erläuterung verdient. Die netShop-Datenstruktur soll leicht auf die Bedürfnisse unterschiedlicher Nutzer anpassbar sein. So verlangen verschiedene Shop-Implementierungen abweichende Attribute für die Artikelbeschreibung. Während bei einem Buch Merkmale wie Autor, Verlag, Erscheinungsjahr oder Seitenzahl eine Rolle spielen können, sieht die Beschreibung für Oberhemden ganz anders aus. Dort interessieren der Hersteller, die Größe, die Farbe, das Material und anderes mehr. Im netShop ist dieser Wunsch nach Flexibilität dadurch umgesetzt worden, dass in der Tabelle *Articles* nur die allgemein gültigen Eigenschaften eines Artikels hinterlegt sind: Name, Preis, Bilder und Ähnliches. Die

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Beschreibungen der Attribute findet man dann in der getrennten Tabelle *ArticleAttributes*. Hier findet man die für eine Artikelklasse (Bücher, Hemden, etc.) hinterlegten Eigenschaften. In einem Shopsystem spricht man allerdings häufig nicht von einer Klasse, sondern von einem Katalog – was nicht so ganz der landläufigen Verwendung dieses Begriffs entspricht. Eine Menge von zusammengehörigen Attributen steht immer zu einem Datensatz in der Tabelle *Catalogs* in Beziehung. Dieser Eintrag gibt also die Art der Beschreibung eines Artikels vor. In der Tabelle *ArticlesToArticleAttributes* wird dann die Verknüpfung von Artikeln und Attributen vorgenommen. Diese Tabelle enthält die konkreten Werte für einen bestimmten Artikel. Damit an der Benutzeroberfläche einer Shopapplikation einfache Wertüberprüfungen vorgenommen werden können, muss jedem Attribut in *ArticleAttributes* über den Fremdschlüssel *AttributTypeID* noch ein »Datentyp« (zum Beispiel »Text«, »Zahl«, »Datum«) aus der Tabelle *AttributTypes* zugewiesen werden.

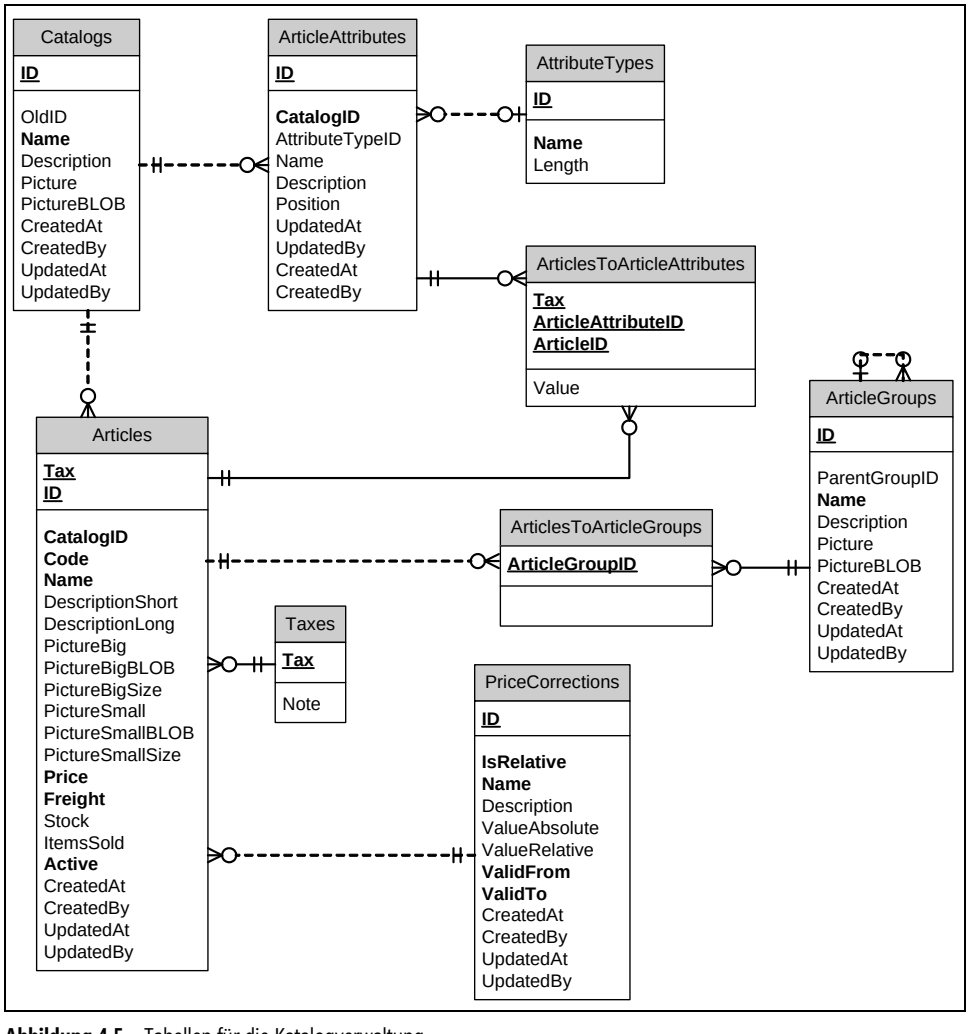

Abbildung 4.5 Tabellen für die Katalogverwaltung

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Würde man die Artikelbeschreibung nicht auf diese Art und Weise realisieren, so müssten für verschiedene Artikelklassen jeweils eigene Tabellen aufgebaut werden. Kommt eine neue Artikelklasse hinzu, so sind die Datenbank und – je nach Programmierung – auch die Applikation anzupassen. Diese Methode der Umsetzung von dynamischen Eigenschaften wird als *Open Schema* bezeichnet. Bei der Implementierung muss darauf geachtet werden, dass die Konsistenz der Datenbank nicht verletzt werden kann. Im Klartext ausgedrückt: Gehört ein Artikel (Datensatz in *Articles*) zu einem bestimmten Katalog (Datensatz in *Catalogs*), dann müssen diesem Artikel *jederzeit alle* Eigenschaften (Datensätze in *ArticleAttributes*) zugeordnet sein, die für diesen Katalog gesetzt wurden. Die Tabelle *ArticleToArticleAttributes* muss entsprechend gepflegt werden, wenn ein neuer Artikel angelegt wird, gelöscht wird oder einem anderen Katalog zugewiesen wird. Das ist eine Aufgabe, die in einem Datenbankserver hervorragend durch so genannte Trigger gelöst werden kann. Die stellen sicher, dass diese *Denormalisierung* der Datenstrukturen nicht zu fehlerhaften Zuständen in der Datenbank führt. In SQL Server 2008 gibt es weitere Möglichkeiten, mit einem Open Schema umzugehen: Sie könnten die Artikelbeschreibungen als XML-Dokumente ablegen. Sie könnten mit einer einzelnen, sehr breiten Tabelle unter Verwendung der *SPARSE*-Eigenschaft von Tabellenspalten arbeiten. Tabellen mit *SPARSE*-Spalten können bis zu 30.000 Spalten beinhalten und speichern leere Werte nicht ab. Oder Sie könnten einen »pfiffigen« .NET-basierten Benutzerdatentyp erfinden, der die Artikeleigenschaften dynamisch zur Verfügung stellt. Neben den Katalogen existiert in der netShop-Datenbank noch eine weitere Beschreibungsmöglichkeit für Artikel. Das sind die Artikelgruppen – dargestellt über die Tabelle *ArticleGroups*. In der realen Welt könnten Artikelgruppen so illustre Namen wie »Weiße Ware« oder »Braune Ware« tragen. In der brauen Ware (Unterhaltungselektronik) gäbe es dann wieder Artikel, die aus verschiedenen Katalogen stammen. LCD-Fernseher haben eben andere Eigenschaften (»Diagonale«, »Tuner«, »Surround«) als portable MP3-Player (»Speicher«, »Gewicht«, »Akkulaufzeit«). Ergibt sich langsam ein Bild? Artikelgruppen lassen sich für verschiedene Zwecke einsetzen. Neben der Zusammenfassung von Artikeln zu Gruppen, die es dem Käufer einfacher machen sollen, sich zu orientieren, können auch andere Gruppen gebildet werden: »Auslaufartikel«, »Neuheiten« und so weiter. So lässt sich die Präsentation im Shop steuern. Durch diese mehrfache Zuordnungsmöglichkeit entsteht zwischen den Artikeln und den Artikelgruppen insgesamt eine M:N-Beziehung. Diese wird über die Verknüpfungstabelle *ArticlesToArticleGroups* implementiert. Auch die *ArticleGroups*-Tabelle ist – genau wie *Employees* – selbstreferenzierend. Dadurch lassen sich aus Artikelgruppen so genannte »Artikelpyramiden« machen. Das sind Hierarchien von Artikelgruppen.

Schlussendlich wäre ein ordentlicher Shop nichts ohne häufige Rabattaktionen. In der Tabelle *PriceCorrections* lassen sich daher temporäre Korrekturen für Artikelpreise hinterlegen. Diese lassen sich als absoluter Wert (»Sparen Sie einen Euro!«) oder als relativer Wert (»Alles um 10% reduziert!«) darstellen. Die Einträge in *PriceCorrections* stehen in keiner Verbindung zu Artikelgruppen oder Katalogen. Sie sind eine weitere Dimension der Beschreibung von Artikeln. Die Zuordnung kann der Entwickler an andere Tabellen knüpfen oder es kann den Shopbetreibern vollständig selbst überlassen werden, welche Artikel eine Preisreduktion erfahren. Wie auch immer: Ein Datensatz in *PriceCorrections* hat immer ein Anfangs- und ein Enddatum, das auch eingegeben werden muss.

In verschiedenen Tabellen der Datenbank findet man Datentypen für das Ablegen großer Objekte – LOB (Large Objects) und BLOB-Daten (Binary Large Objects). In der Katalogverwaltung geht es dabei vor allen Dingen um das Ablegen von Bildinformationen. Die Spalte *PictureBigBLOB* kann beispielsweise eine Artikeldarstellung enthalten, die direkt in der Datenbank abgelegt ist. Die Spalte *PictureBig* liefert den ursprünglichen Dateinamen dazu. Eine Besonderheit stellt die Spalte *Document* in der *Contracts*-Tabelle dar. Hier können komplette Word-, PDF- oder andere Dokumente abgelegt werden. Diese werden von SQL Server nicht nur verwaltet, sondern auch durch Volltextindizierung inhaltlich erschlossen. Dazu benötigt die Datenbankmaschine zusätzlich noch die Spalte *DocumentType.* In dieser ist die Art des Dokuments hinterlegt: ».DOC«, ».TXT« und so weiter.

Archivdaten

Zu guter Letzt gibt es in der netShop-Datenbank noch einen Bereich für Daten, die nicht mehr aktuell sind, aber für Auswertungen noch im Zugriff gehalten werden sollen. Dieser setzt sich aus den Tabellen zusammen, deren Namen das Präfix »Archive_« enthalten (Abbildung 4.6). In diese Tabellen sollen ausschließlich zu archivierende Daten aus den aktiven Tabellen geladen und anschließend abgefragt werden. Änderungen und Löschungen gibt es nicht. Struktur der Tabellen und die Beziehungen untereinander entsprechen genau den Originaltabellen, allerdings werden in den Archivtabellen keine Schlüssel gebildet. Der Vorteil in dieser Abtrennung besteht darin, dass die archivierten Daten bei Abfragen nicht mehr durchsucht werden müssen, der Aufwand der Reindizierung sich vermindert und die Daten physisch auf andere Laufwerke gelegt werden können (über so genannte Dateigruppen).

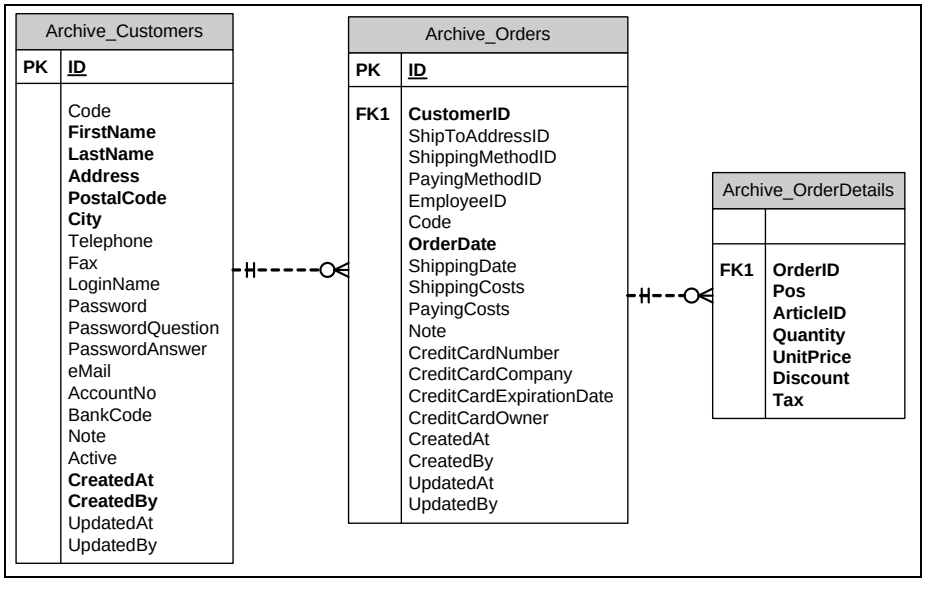

Abbildung 4.6 Tabellen für Archivdaten

Protokollierung

Einige wichtige Tabellen verfügen über vier gleichartige Spalten, die einem einfachen Tracking der Benutzeraktionen in der Datenbank dienen. Die beiden Spalten *CreatedAt* und *CreatedBy* halten fest, wann und durch wen der betreffende Datensatz angelegt wurde. In *UpdatedAt* und *UpdatedBy* sind das Datum und der Benutzer der letzten Änderung zu finden. Diese vier Felder werden auf der Datenbankebene vollautomatisch gepflegt, ohne dass sich die Frontend-Entwickler darum kümmern müssten. Als Beispiel für eine ausführliche Verfolgung von Aktionen in der Datenbank dient die Tabelle *Journal.* In Kapitel 17 (»Trigger«) wird vorgestellt, wie man so eine zentrale Protokolltabelle automatisch pflegt.

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Weitere Datenbankobjekte

Abgesehen von den Tabellen und den Beziehungen, die über Primär- und Fremdschlüssel definiert sind, findet man im Ausgangszustand der netShop-Datenbank nicht allzu viele Objekte. Im Wesentlichen gibt es noch *Trigger* und *Indizes*.

Trigger – das sind spezielle Prozeduren, die SQL Server ausführt, wenn sich die Daten in einer Tabelle ändern – werden eingesetzt, um Werte in der Datenbank automatisch zu setzen. Zum einen werden Trigger in den Tabellen *ArticleAttributes* und *Articles* einsetzt, um das Open Schema in der Datenbank zu pflegen. Zum anderen verfügen alle Tabellen, welche die Spalten *UpdatedAt* und *UpdatedBy* besitzen, über Trigger. Diese tragen den letzten Bearbeiter und das letzte Bearbeitungsdatum in die Tabellen ein (die Spalten *CreatedAt* und *CreatedBy* können über Standardwerte in den Tabellen vorgegeben werden).

Indizes werden für sämtliche Fremdschlüsselfelder in der Datenbank eingesetzt. Dies, in Verbindung mit dem »kurzen« Datentyp *int* für Schlüssel und Fremdschlüsselfelder, ermöglicht effektive Abfragen über verknüpfte Tabellen (JOINs). Weitere Indizes sind auf Spalten gesetzt, von denen bereits klar ist, dass sie in Suchen eine Rolle spielen werden: *Articles.Code, Articles.Name, Orders.Code* und so weiter.

Keine Sorge – wie Sie mit Triggern arbeiten und wie die Indizierung einer Datenbank an die speziellen Bedürfnisse einer Applikation angepasst wird, das erfahren Sie in den Kapiteln 17 (»Trigger«) und 12 (»Indizierung und Partitionierung«) noch ganz genau.

Berechtigungen, Tabellen, Schemata und Synonyme

Das Berechtigungssystem ist in der Demodatenbank recht einfach gehalten. Es gibt vier Benutzergruppen (*Rollen*), die unterschiedliche Aufgaben in einer Datenbankanwendung durchführen sollen.

- *Mitarbeiter in der Verkaufsabteilung* (Benutzerrolle *SalesDepartment*) können Kunden und Bestelldaten bearbeiten und Artikeldaten einsehen. Sie haben daher für die Tabellen in Abbildung 4.4 volle Berechtigungen und Leseberechtigungen für die Katalogtabellen.
- *Mitarbeiter in der Produktabteilung* (Benutzerrolle *ProductDepartment*) pflegen den Produktkatalog, legen Artikel an, setzen Preise fest und haben daher volle Zugriffsrechte auf die Tabellen in Abbildung 4.5
- *Shopmanager* (Benutzerrolle *ShopManagement*) dürfen alle Tabellen bearbeiten, auch die Nachschlagetabellen *ShippingMethods, PayingMethods, OrderTrackingItems* sowie die Tabelle *Employees*
- *Mitarbeiter in der Entwicklungsabteilung* (Benutzerrolle *DevelopmentDepartment*) sind in jeder Tabelle als Besitzer eingetragen und dürfen daher neben den Daten auch die Definition der Tabellen ändern

In der netShop-Datenbank wird mit so genannten Schemata gearbeitet, um die Rechte möglichst einfach verwalten zu können. Für den Augenblick können Sie sich ein Schema einfach als eine Sammlung von Objekten vorstellen, die einen gemeinsamen Besitzer und gemeinsame Anforderungen an die Benutzerrechte haben. Den Benutzergruppen werden in der Datenbank die notwendigen Rechte also nicht für jedes Objekt einzeln gegeben, sondern über Schemata. Tabelle 4.1 gibt eine Übersicht über die verwendeten Schemata und Datenbanktabellen.

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Tabelle	Schema	Beschreibung
Journal	Internal	Protokollierung von Datenänderungen
Employees	Management	Mitarbeiterinformationen
OrderTrackingItems	Management	Nachschlagetabelle für die Ablaufverfolgung
PayingMethods	Management	Nachschlagetabelle für Bezahlarten
ShippingMethods	Management	Nachschlagetabelle für Versandarten
ArticleAttributes	Products	Artikelmerkmale in einem Katalog
ArticleGroups	Products	Artikelgruppen
Articles	Products	Artikelinformationen
ArticlesToArticleAttributes	Products	Festlegung von Artikelmerkmalen
ArticlesToArticleGroups	Products	Zuordnung von Artikeln zu Artikelgruppen
AttributeTypes	Products	Datentypen für Artikelmerkmale
Catalogs	Products	Artikelkataloge
PriceCorrections	Products	Temporäre Preiskorrekturen
Taxes	Products	Nachschlagetabelle für Steuersätze
Archive_Customers	Sales	Archiv für Kundeninformationen
Archive_OrderDetails	Sales	Archiv für Bestellpositionen
Archive_Orders	Sales	Archiv für Bestellungen
Contracts	Sales	Lieferverträge
Customers	Sales	Kundeninformationen
OrderDetails	Sales	Bestellpositionen
Orders	Sales	Bestellungen
OrdersToOrderTrackingItems	Sales	Ablaufverfolgung von Bestellungen
ShipToAdresses	Sales	Versandadressen

Tabelle 4.1 Übersicht über die *netShop*-Tabellen

Ein Objekt wird in SQL Server 2008 über sein Schema und seinen Namen angesprochen. Die Kundentabelle also über *Sales.Customers*. Dies ist die empfohlene Art, SQL-Befehle zu formulieren und so wird auch in den Beispielen dieses Buchs gearbeitet. Um aber kürzere Schreibweisen zu ermöglichen, wurden in der netShop-Datenbank so genannte *Synonyme* vereinbart. Jeder Tabellenname existiert noch einmal als Synonym. Daher funktioniert prinzipiell die Bezeichnung *Customers* für die Kundentabelle ebenso.

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)

Georg Urban, Jörg Neumann, Klaus Löffelmann, Alexander Köller:
Microsoft SQL Server R2 - Das Entwicklerbuch, Microsoft Press 2011 (ISBN 978-3-86645-514-6)