KAPITEL 1

Einführung in die Datenmodellierung

In diesem Buch geht es um Datenmodellierung. Als Erstes stellt sich die Frage, warum Sie sich überhaupt damit beschäftigen sollten. Schließlich können Sie auch einfach Erkenntnisse aus Ihren Daten gewinnen, indem Sie in Excel eine Abfrage laden und eine Pivottabelle daraus erstellen. Wozu brauchen Sie da Datenmodellierung?

Als Berater werden wir täglich von Einzelpersonen oder Unternehmen beauftragt, die Schwierigkeiten damit haben, die erforderlichen Zahlen zu berechnen. Sie haben das Gefühl, dass die Zahl, nach der sie suchen, existiert und berechnet werden kann, aber entweder die Formeln zu kompliziert sind oder die Zahlen nicht stimmen. In 99 % der Fälle liegt das an einem Fehler im Datenmodell. Wenn Sie das Modell korrigieren, lässt sich die Formel leicht aufstellen und verstehen. Wenn Sie Ihre Analysemöglichkeiten verbessern und sich lieber auf die Entscheidungsfindung konzentrieren möchten anstatt darauf, eine komplizierte DAX-Formel auszutüfteln, müssen Sie daher Datenmodellierung lernen.

Datenmodellierung gilt gewöhnlich als schwer zu erlernen. Wir werden Ihnen nicht einreden, dass das nicht so wäre. Datenmodellierung ist ein vielschichtiges Thema. Es ist anspruchsvoll und es erfordert einige Anstrengung, um es zu lernen und um Ihr Gehirn darauf zu trainieren, bei der Betrachtung eines Szenarios das Modell im Geiste vor sich zu sehen. Es stimmt, Datenmodellierung ist kompliziert, anspruchsvoll und erweitert den Geist. Mit anderen Worten, es macht viel Spaß!

In diesem Kapitel finden Sie einige einfache Beispiele von Berichten, bei denen das richtige Datenmodell zu einfacheren Formeln führt. Da es sich um Beispiele handelt, lassen sie sich natürlich nicht vollständig auf Ihr Geschäft übertragen. Dennoch hoffen wir, dass sie Ihnen eine gute Vorstellung davon geben, warum Datenmodellierung eine so wichtige Fähigkeit ist. Ein guter Datenmodellierer zu sein, bedeutet im Grunde genommen, Ihr spezifisches Modell einem der vielen verschiedenen Muster zuzuordnen, die bereits von anderen untersucht und eingerichtet worden sind. Ihr Modell unterscheidet sich gar nicht so stark von anderen. Es hat sicherlich einige Eigenheiten, aber es ist sehr wahrscheinlich, dass Ihr Problem bereits von jemand anderem gelöst worden ist. Zu lernen, wie Sie Ähnlichkeiten zwischen Ihren Datenmodellen und denen in den Beispielen finden, ist nicht einfach, aber sehr befriedigend. Die Lösung erscheint dann vor Ihren Augen, und die meisten Probleme mit Ihren Berechnungen verschwinden schlagartig.

Für die meisten unserer Beispiele verwenden wir die Datenbank von Contoso. Dabei handelt es sich um ein fiktives Unternehmen, das über verschiedene Vertriebskanäle elektronische Geräte in aller Welt verkauft. Ihr Geschäft unterscheidet sich sehr wahrscheinlich davon, wes-

halb Sie die Contoso-Berichte und die daraus gewonnenen Erkenntnisse auf Ihren Fall übertragen müssen.

Da dies das erste Kapitel ist, beschäftigen wir uns zunächst mit der Terminologie und den Grundprinzipien. Wir erklären, was ein Datenmodell ist und warum Beziehungen so wichtige Bestandsteile davon sind. Außerdem führen wir die Begriffe Normalisierung, Denormalisierung und Sternschema ein. Die Vorgehensweise, Prinzipien anhand von Beispielen vorzuführen, halten wir im ganzen Buch ein, aber hier, in den ersten Schritten, ist sie viel offensichtlicher.

Schnallen Sie sich an und tauchen Sie ein in die Geheimnisse der Datenmodellierung!

Arbeiten mit einer einzelnen Tabelle

Wenn Sie Excel und Pivottabellen verwenden, um Erkenntnisse aus Ihren Daten zu ziehen, laden Sie diese Daten wahrscheinlich mithilfe einer Abfrage aus einer Quelle, gewöhnlich einer Datenbank. Anschließend erstellen Sie eine Pivottabelle aus diesem Dataset (Datenmenge) und beginnen mit Ihren Nachforschungen. Dabei unterliegen Sie natürlich den üblichen Einschränkungen von Excel, wobei die wichtigste lautet, dass das Dataset nicht mehr als 1.000.000 Zeilen umfassen darf, da es sonst nicht in ein Arbeitsblatt passt. Ehrlich gestanden, als wir zum ersten Mal von dieser Einschränkung hörten, hielten wir sie nicht einmal für eine Einschränkung. Warum um alles in der Welt sollte jemand 1.000.000 Zeilen in Excel laden wollen, anstatt eine Datenbank zu verwenden? Man könnte meinen, der Grund für ein solches Vorgehen liege daran, dass Excel im Gegensatz zu Datenbanken keine Kenntnisse über Datenmodellierung erfordert.

Wenn Sie tatsächlich Excel verwenden wollen, kann dies jedoch eine wirklich schwere Einschränkung darstellen. In der Contoso-Datenbank, die wir für unsere Beispiele verwenden, umfasst die Verkaufstabelle 12.000.000 Zeilen. Damit ist es nicht möglich, sie komplett in Excel zu laden, um mit der Analyse zu beginnen. Für dieses Problem gibt es jedoch eine einfache Lösung: Anstatt alle Zeilen abzurufen, verringern Sie die Anzahl, indem Sie eine Gruppierung durchführen. Wenn Sie beispielsweise an einer Analyse der Verkäufe nach Kategorie und Unterkategorie interessiert sind, laden Sie nicht die Verkaufszahlen für jedes Produkt, sondern gruppieren die Daten nach Kategorie und Unterkategorie, was die Anzahl der Zeilen erheblich reduziert.

Wenn Sie die Verkaufstabelle mit ihren 12.000.000 Zeilen nach Hersteller, Marke, Kategorie und Unterkategorie gruppieren und die Angabe der Verkäufe pro Tag beibehalten, erhalten Sie 63.984 Zeilen, was sich in einer Excel-Arbeitsmappe gut handhaben lässt. Die richtige Abfrage zu schreiben, um eine solche Gruppierung durchzuführen, ist gewöhnlich eine Aufgabe für die IT-Abteilung (sofern Sie nicht selbst SQL gelernt haben). Wenn Sie den Code für die Abfrage haben, können Sie mit der Analyse der Zahlen beginnen. In Abbildung 1–1 sehen Sie die ersten Zeilen der Tabelle nach dem Import in Excel.

Wenn die Tabelle in Excel geladen ist, können Sie sich endlich zu Hause fühlen und eine Pivottabelle zur Analyse der Daten erstellen. In Abbildung 1–2 sehen Sie als Beispiel die Verkaufszahlen pro Hersteller für eine gegebene Kategorie. Dabei wurden eine gewöhnliche Pivottabelle und ein Datenschnitt verwendet.

FullDateLabel	Manufacturer	BrandName	ProductSubcategoryName	ProductCategoryName	SalesQuantity	SalesAmount	TotalCost
2007-03-31	Adventure Works	Adventure Works	Coffee Machines	Home Appliances	55	14332.268	7651.84
2008-10-22	Contoso, Ltd	Contoso	Cell phones Accessories	Cell phones	2040	23504.88	12648.94
2009-01-31	Adventure Works	Adventure Works	Televisions	TV and Video	194	51593.106	28146.4
2009-01-21	Fabrikam, Inc.	Fabrikam	Camcorders	Cameras and camcorders	282	163007.2	76709.45
2007-12-31	Adventure Works	Adventure Works	Laptops	Computers	29	14008.43	7944.32
2007-06-22	Contoso, Ltd	Contoso	Cell phones Accessories	Cell phones	680	6107.24	3420.44
2007-06-22	Proseware, Inc.	Proseware	Projectors & Screens	Computers	86	71417.6	30786.94
2007-08-23	Adventure Works	Adventure Works	Laptops	Computers	43	22672.2	9954.6
2009-03-30	The Phone Company	The Phone Company	Touch Screen Phones	Cell phones	198	48500.37	24164.56
2008-03-24	Contoso, Ltd	Contoso	Home & Office Phones	Cell phones	306	7353.594	3914.64
2007-09-30	Fabrikam, Inc.	Fabrikam	Microwaves	Home Appliances	44	4805.604	2824.24
2007-11-13	Adventure Works	Adventure Works	Desktops	Computers	153	47357.97	28256.02
2008-12-06	Contoso, Ltd	Contoso	Projectors & Screens	Computers	32	10790.4	6477.2
2007-11-14	Contoso, Ltd	Contoso	Digital SLR Cameras	Cameras and camcorders	146	55397.5	25876
2009-12-30	Adventure Works	Adventure Works	Desktops	Computers	32	15107.75	7952.97
2009-03-13	Wide World Importers	Wide World Importers	Recording Pen	Audio	42	7990.92	3607.26
2009-08-11	Wide World Importers	Wide World Importers	Recording Pen	Audio	9	1466.1	749.16
2009-09-28	Contoso, Ltd	Contoso	Microwaves	Home Appliances	78	9955.268	5189.27
2008-02-18	A. Datum Corporation	A. Datum	Digital Cameras	Cameras and camcorders	345	70989.93	32872.58
2007-08-15	Litware, Inc.	Litware	Washers & Dryers	Home Appliances	69	112603.8	56472.35

Abbildung 1–1 Durch die Gruppierung von Verkaufsdaten entsteht eine kleine und leicht zu analysierende Tabelle.

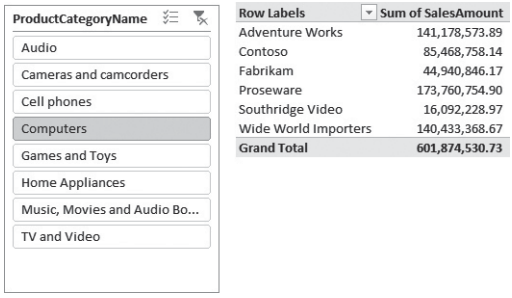


Abbildung 1–2 Aus einer Excel-Tabelle lässt sich leicht eine Pivottabelle erstellen.

Ob Sie es glauben oder nicht – damit haben Sie schon ein Datenmodell erstellt! Auch wenn es nur eine einzige Tabelle umfasst, ist es doch ein Datenmodell. Sie können nun seine analytischen Möglichkeiten erkunden und möglicherweise verbessern. Das Datenmodell in diesem Beispiel ist stark eingeschränkt, da es weniger Zeilen aufweist als die Quelltabelle.

Als Anfänger sind Sie vielleicht der Meinung, dass der Grenzwert von 1.000.000 Zeilen in einer Excel-Tabelle nur die Anzahl der Zeilen betrifft, die Sie zur Analyse abrufen können. Das stimmt zwar, aber diese Größeneinschränkung führt auch zu einer Einschränkung des Datenmodells und damit der analytischen Möglichkeiten Ihrer Berichte. Um die Anzahl der Zeilen zu verringern, mussten Sie die Daten schon in der Quelle gruppieren, sodass Sie nur die nach Spalten geordneten Verkäufe abrufen konnten, in diesem Beispiel nach Kategorie, Unterkategorie und einigen anderen Spalten.

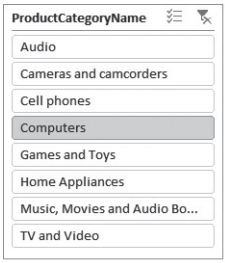
Dadurch beschränken Sie implizit Ihre Analysemöglichkeiten. Wenn Sie beispielsweise einen Datenschnitt nach Farbe durchführen wollen, ist die Tabelle schon nicht mehr als Quelle geeignet, da sie keine Spalte für die Produktfarbe enthält. Eine Spalte zu der Abfrage hinzuzufügen ist kein großes Problem; das wirkliche Problem besteht darin, dass die Tabelle mit jeder Spalte größer wird, und zwar nicht nur, was die Breite (also die Anzahl der Spalten) angeht, sondern auch die Länge (die Anzahl der Zeilen). Tatsächlich wird aus einer einzigen Zeile mit den Verkäufen für eine gegebene Kategorie – z. B. Audio – ein Satz mehrerer Zeilen, jede davon mit der Kategorie Audio, aber mit Werten für die verschiedenen Farben.

Wenn Sie im Extremfall nicht im Voraus entscheiden wollen, welche Spalten Sie für den Datenschnitt verwenden möchten, müssen Sie alle 12.000.000 Zeilen laden – und damit können Sie keine Excel-Tabelle mehr verwenden. Das ist es, was wir meinten, als wir schrieben, dass die Modellierungsmöglichkeiten von Excel eingeschränkt sind. Nicht in der Lage zu sein, viele Zeilen zu laden, bedeutet implizit, nicht in der Lage zu sein, eine fortgeschrittene Analyse an umfangreichen Datenvolumen vorzunehmen.

Hier kommt Power Pivot ins Spiel. Damit haben Sie nicht mehr mit der Beschränkung auf 1.000.000 Zeilen zu leben. Es gibt praktisch keinen Grenzwert für die Anzahl der Zeilen, die Sie in eine Power Pivot-Tabelle laden können. Mit Power Pivot können Sie die komplette Verkaufstabelle in das Modell laden und eine tiefschürfende Analyse der Daten durchführen.

Power Pivot ist seit Excel 2010 als externes Add-In in Excel verfügbar und seit Excel 2013 Teil des Produkts. Seit Excel 2016 verwendet Microsoft die neue Bezeichnung Excel-Datenmodell für ein Power Pivot-Modell, allerdings wird der Begriff Power Pivot ebenfalls noch verwendet.

Da Ihnen jetzt alle Verkaufsinformationen in einer einzigen Tabelle vorliegen, können Sie eine ausführlichere Analyse an den Daten ausführen. In Abbildung 1–3 sehen Sie beispielsweise eine Pivottabelle aus dem Datenmodell (d. h. aus Power Pivot), in der alle Spalten geladen sind. Jetzt können Sie Datenschnitte nach Kategorie, Farbe und Jahr durchführen, da all diese Informationen vorhanden sind. Mehr verfügbare Spalten in der Tabelle bedeuten mehr Analysemöglichkeiten.



Sum of SalesAmount	Column Labels			
Row Labels	2007	2008	2009	Grand Total
Black	$59,783,936.63	$63,073,257.64	$70,489,997.35	$193,347,191.62
Blue	$5,128,405.12	$6,516,691.17	$7,715,161.42	$19,360,257.71
Brown	$6,301,722.60	$7,183,128.75	$4,904,738.33	$18,389,589.68
Gold		$39,185.40	$122,755.63	$161,941.03
Green	$1,747,237.19	$1,566,822.01	$2,159,190.14	$5,473,249.35
Grey	$6,318,419.68	$5,924,762.67	$6,410,704.05	$18,653,886.41
Orange		$12,800.63	$84,766.80	$97,567.43
Pink	$20,078.44	$43,870.83	$228,526.86	$292,476.13
Red	$6,926,045.96	$7,872,382.02	$11,495,613.50	$26,294,041.47
Silver	$43,375,399.72	$39,082,679.67	$36,471,502.90	$118,929,582.29
White	$64,963,894.39	$64,142,854.42	$70,639,615.97	$199,746,364.78
Yellow	$260,512.33	$330,165.82	$537,704.67	$1,128,382.82
Grand Total	$194,825,652.07	$195,788,601.04	$211,260,277.62	$601,874,530.73

Abbildung 1–3 Wenn sämtliche Spalten verfügbar sind, können Sie aus Ihren Daten interessantere Pivottabellen erstellen.

Dieses einfache Beispiel vermittelt Ihnen schon einen ersten wichtigen Grundsatz der Datenmodellierung: *Größe ist wichtig, da sie mit der Granularität zusammenhängt.* Aber was ist *Granularität*? Da es sich um einen der wichtigsten Begriffe handelt, den Sie in diesem Buch kennenlernen werden, führen wir ihn hier möglichst früh ein. Im weiteren Verlauf werden wir die Erklärung noch vertiefen, aber zur Einführung wollen wir zunächst eine einfache Erklärung geben. In dem ersten Dataset haben Sie die Informationen nach Kategorie und Unterkategorie gruppiert und dabei auf einige Einzelheiten verzichtet, um die Größe zu verringern. Technisch ausgedrückt haben Sie eine Granularität auf der Ebene von Kategorien und Unterkategorien

gewählt. Sie können sich die Granularität als die Detailliertheit oder Feinheit Ihrer Tabellen vorstellen: je größer die Granularität, umso detaillierter die Informationen. Mit mehr Einzelheiten können Sie auch eine detailliertere Analyse durchführen. Beim letzten Dataset – also demjenigen, das wir in Power Pivot geladen haben – befindet sich die Granularität nicht mehr auf Kategorie- und Unterkategorie-, sondern auf Produktebene. (Tatsächlich ist sie noch feiner, nämlich auf der Ebene der einzelnen Verkäufe eines Produkts.) Ihre Möglichkeiten für Datenschnitte und zum Drehen (Slice and Dice) hängen von der Anzahl der Spalten in der Tabelle, also von deren Granularität ab. Wie Sie bereits wissen, haben Sie mit einer erhöhten Anzahl von Spalten auch mehr Zeilen.

Die richtige Granularität zu wählen, ist immer eine schwierige Aufgabe. Wenn Ihre Daten die falsche Granularität aufweisen, wird es fast unmöglich, Formeln zu schreiben, da die Informationen entweder verloren sind (wie in dem vorherigen Beispiel, wo es keine Farbinformationen mehr gab) oder über die Tabelle verstreut und falsch gegliedert. Es ist daher auch nicht richtig zu sagen, dass eine höhere Granularität immer gut ist. Die Daten müssen die *richtige* Granularität aufweisen, also diejenige, die sich am besten für den vorliegenden Zweck eignet.

Ein Beispiel für verlorene Informationen haben Sie bereits gesehen. Aber was bedeutet »verstreute« Informationen? Das lässt sich nicht ganz so einfach erkennen. Nehmen wir an, Sie wollen das durchschnittliche jährliche Einkommen der Kunden berechnen, die eine bestimmte Auswahl Ihrer Produkte kaufen. Diese Information ist vorhanden, denn in der Verkaufstabelle sind alle Informationen über die Kunden verfügbar. Das können Sie in Abbildung 1–4 erkennen, die einige Spalten der Tabelle zeigt, mit der wir arbeiten. (Um den Inhalt der Tabelle sehen zu können, müssen Sie das Power Pivot-Fenster öffnen.)

ProductCategoryName	ProductSubcategoryName	ProductName	SalesAmount	FirstName	LastName	YearlyIncome
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Katrina	Xie	€ 20,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Seth	Rodriguez	€ 80,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Evelyn	Arun	€ 10,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Christy	Beck	€ 40,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Alejandro	Nara	€ 40,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Leah	Lu	€ 30,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Robyn	Torres	€ 20,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Jimmy	Moreno	€ 30,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Rafael	Cai	€ 20,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Jenny	Ferrier	€ 110,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Levi	Arun	€ 70,000.00
Cameras and camcorders	Digital SLR Cameras	A. Datum SLR Camera X137 Grey	$627.00	Randall	Torres	€ 40,000.00

Abbildung 1–4 Die Produkt- und die Kundeninformationen stehen in derselben Tabelle.

In jeder Zeile der Tabelle *Sales* befindet sich auch eine Spalte mit dem Jahreseinkommen des Kunden, der das betreffende Produkt gekauft hat. Für einen einfachen Versuch, das durchschnittliche Jahreseinkommen der Kunden zu berechnen, können wir wie folgt ein berechnetes Feld (*Measure*) aus DAX verwenden:

```
AverageYearlyIncome := AVERAGE ( Sales[YearlyIncome] )
```

Das berechnete Feld funktioniert sehr gut. Sie können es in einer Pivottabelle wie in Abbildung 1–5 einsetzen, die das durchschnittliche Jahreseinkommen der Kunden zeigt, die Haushaltsgeräte verschiedener Marken kaufen.

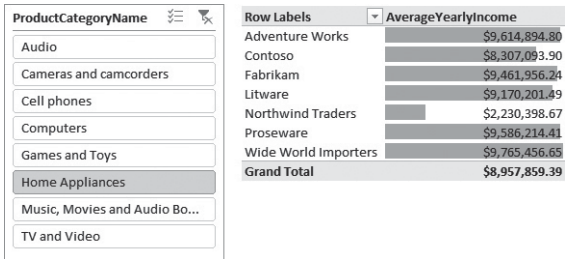


Abbildung 1–5 Analyse des durchschnittlichen Jahreseinkommens von Kunden, die Haushaltsgeräte kaufen.

Der Bericht sieht gut aus, aber leider ist die berechnete Zahl falsch, nämlich viel zu hoch. Was Sie hier berechnen, ist der Durchschnitt über die Verkaufstabelle, die eine Granularität auf der Ebene einzelner Verkäufe aufweist. Das heißt, die Tabelle enthält eine Zeile für jeden Verkauf, also möglicherweise mehrere Zeilen für denselben Kunden. Kauft ein Kunde also beispielsweise drei Produkte an drei verschiedenen Tagen, wird er bei der Bildung des Durchschnitts dreimal gezählt, was zu einem falschen Ergebnis führt.

Man könnte meinen, dadurch würde ein gewichteter Durchschnitt berechnet, aber auch das ist nicht ganz korrekt. Um einen gewichteten Durchschnitt zu berechnen, müssen Sie eine Gewichtung definieren, und dazu ziehen Sie nicht einfach die Anzahl der Kaufvorgänge heran, sondern die Anzahl der Produkte, den Gesamtbetrag oder irgendeinen anderen bedeutungsvollen Wert. Außerdem hatten wir in diesem Beispiel ja ohnehin vorgehabt, nur einen einfachen Durchschnitt zu berechnen, und das erledigt das berechnete Feld nun einmal nicht korrekt.

Es ist nicht so leicht zu erkennen, aber auch hier haben wir mit dem Problem einer falschen Granularität zu kämpfen. Die Informationen sind zwar verfügbar, aber nicht mit einem einzelnen Kunden verknüpft, sondern über die ganze Verkaufstabelle verstreut, weshalb es schwer ist, die Berechnungsformel zu schreiben. Um den richtigen Durchschnitt zu erhalten, müssen Sie für Granularität auf Kundenebene sorgen, indem Sie entweder die Tabelle neu laden oder eine kompliziertere DAX-Formel einsetzen.

Letzteres können Sie wie folgt tun, allerdings ist diese Formel nicht ganz einfach zu verstehen:

```
CorrectAverage :=
AVERAGEX (
   SUMMARIZE (
      Sales,
      Sales[CustomerKey],
      Sales[YearlyIncome]
   ),
   Sales[YearlyIncome]
)
```

Als Erstes müssen Sie die Verkäufe auf Kundenebene aggregieren (um eine Granularität auf Kundenebene zu bekommen). Erst dann können Sie eine `AVERAGE`-Operation auf der resultierenden Tabelle durchführen, in der jeder Kunde nur einmal vorkommt. In diesem Beispiel ver-

wenden wir SUMMARIZE, um die Vorabaggregation auf Kundenebene in einer temporären Tabelle durchzuführen, und ermitteln dann den Durchschnitt von *YearlyIncome* in dieser temporären Tabelle. Wie Sie in Abbildung 1–6 sehen, unterscheidet sich die korrekte Zahl erheblich von dem zuvor berechneten falschen Wert.

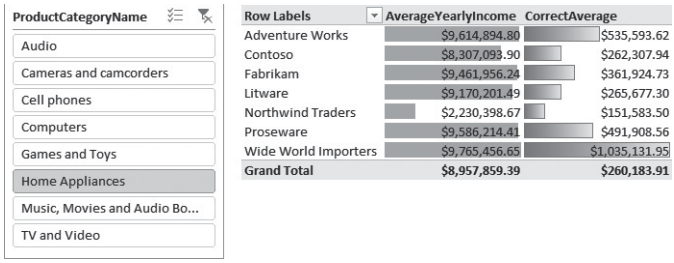


Abbildung 1–6 Die Gegenüberstellung der Daten für den richtigen und den falschen Durchschnitt zeigt, wie weit wir danebengelegen haben.

Es lohnt sich, etwas mehr Zeit zu investieren, um sich voll und ganz mit der folgenden einfachen Wahrheit vertraut zu machen: Das Jahreseinkommen ist eine Information, die auf der Ebene des einzelnen Kunden von Bedeutung ist. Auf der Ebene der einzelnen Verkäufe dagegen ist diese Zahl fehl am Platze. Anderes ausgedrückt, Sie können einen Wert, der eine Bedeutung auf Kundenebene hat, nicht mit derselben Bedeutung auf der Ebene der einzelnen Verkäufe verwenden. Um das richtige Ergebnis zu erhalten, mussten wir hier die Granularität reduzieren, wenn auch nur in einer temporären Tabelle.

Aus diesem Beispiel können Sie zwei wichtige Dinge lernen:

- Die richtige Formel ist weit komplizierter als ein einfaches AVERAGE. Sie müssen Werte vorübergehend aggregieren, um die korrekte Granularität der Tabelle zu erhalten, da die Daten in der Tabelle verstreut und nicht auf geeignete Weise geordnet sind.
- Es ist sehr wahrscheinlich, dass Sie einen solchen Fehler übersehen, wenn Sie mit den Daten nicht vertraut sind. Ein Blick auf den Bericht in Abbildung 1–6 lässt schon ahnen, dass das Jahreseinkommen viel zu hoch ist, um wahr zu sein – als ob keiner der Kunden weniger als 2.000.000 Dollar pro Jahr verdienen würde! Bei komplizierteren Berechnungen jedoch kann es weit schwieriger sein, den Fehler zu erkennen, und das kann zu Berichten mit falschen Zahlen führen.

Sie müssen die Granularität erhöhen, um Berichte mit dem gewünschten Detaillierungsgrad hervorzurufen, aber wenn Sie sie zu weit erhöhen, wird die Berechnung mancher Zahlen schwieriger. Wie wählen Sie daher die korrekte Granularität? Das ist eine knifflige Frage, deren Beantwortung wir uns für später aufheben. Wir hoffen, Ihnen die erforderlichen Kenntnisse vermitteln zu können, um die richtige Granularität der Daten in Ihren Modellen erkennen zu können. Allerdings ist die Auswahl der richtigen Granularität eine Fähigkeit, die sich selbst erfahrene Datenmodellierer nicht so leicht aneignen können. Vorläufig jedoch begnügen wir uns mit der Erkenntnis, was Granularität ist und wie wichtig es ist, die richtige Granularität für jede Tabelle in Ihrem Modell zu wählen.

Das Modell, an dem wir bis jetzt gearbeitet haben, leidet in Wirklichkeit unter einem viel größeren Problem, das jedoch auch in gewissem Sinne mit der Granularität zu tun hat. Das

größte Problem dieses Modells besteht darin, dass es nur über eine einzige Tabelle verfügt, die sämtliche Informationen enthält. In einem solchen Modell müssen Sie die Granularität der Tabelle wählen und dabei alle möglichen berechneten Felder und Analysen berücksichtigen, die Sie benutzen bzw. durchführen wollen. Wie sehr Sie sich auch anstrengen, die gewählte Granularität wird niemals ideal für alle berechneten Felder sein. In den nächsten Abschnitten führen wir die Verwendung mehrerer Tabellen ein, bei denen Sie unterschiedliche Granularitäten auswählen können.

Datenmodelle

Im vorherigen Abschnitt haben Sie gelernt, dass ein Modell mit einer einzigen Tabelle Probleme bei der Festlegung der richtigen Granularität hervorruft. Excel-Benutzer verwenden oft Einzeltabellenmodelle, da dies vor der Version 2013 die einzige Möglichkeit zum Erstellen von Pivottabellen war. In Excel 2013 hat Microsoft das Excel-Datenmodell eingeführt, mit dem Sie viele Tabellen laden und über Beziehungen verbinden können. Dadurch können Sie viel leistungsfähigere Datenmodelle erstellen.

Was aber ist ein *Datenmodell*? Dabei handelt es sich einfach um einen Satz von Tabellen, die durch Beziehungen verknüpft sind. Ein Einzeltabellenmodell ist bereits ein Datenmodell, aber kein sehr interessantes. Wenn Sie mehrere Tabellen haben, machen die Beziehungen dazwischen das Modell viel leistungsfähiger und interessanter zu analysieren.

Ein Datenmodell zu erstellen, ist ein ganz natürlicher Vorgang, sobald Sie mehr als eine Tabelle laden. Überdies laden Sie gewöhnlich Daten aus Datenbanken, die von Profis gepflegt werden und bereits über ein Datenmodell verfügen. Das bedeutet, dass Ihr Datenmodell sehr wahrscheinlich das in der Quelldatenbank bereits vorhandene Modell nachstellen wird. In gewissem Sinne vereinfacht das Ihre Arbeit.

Doch wie Sie in diesem Buch noch sehen werden, ist es leider sehr unwahrscheinlich, dass das Quelldatenmodell die ideale Struktur für die Art von Analyse aufweist, die Sie durchführen möchten. Wir werden Ihnen anhand von Beispielen mit zunehmender Komplexität zeigen, wie Sie von einer beliebigen Datenquelle ausgehend Ihr eigenes Modell aufbauen können. Um Ihnen das Lernen zu vereinfachen, beschreiben wir diese Techniken nach und nach im weiteren Verlauf dieses Buches. Zunächst einmal beginnen wir mit den Grundlagen.

Um das Prinzip eines Datenmodells kennenzulernen, laden Sie die Tabellen *Product* und *Sales* der Contoso-Datenbank in das Excel-Datenmodell. Anschließend sehen Sie die Diagrammansicht aus Abbildung 1–7, die die beiden Tabellen mit ihren Spalten zeigt.

Das Beziehungsdiagramm steht in Power Pivot zur Verfügung. Um darauf zuzugreifen, klicken Sie im Excel-Menüband auf die Registerkarte *Power Pivot* und dann auf *Verwalten*. Klicken Sie anschließend auf der Registerkarte *Start* des Power Pivot-Fensters in der Gruppe *Ansicht* auf *Diagrammansicht*.

Zwei unverbundene Tabellen wie in diesem Beispiel sind noch kein echtes Datenmodell, sondern nur zwei Tabellen. Um sie in ein sinnvolles Modell umzuwandeln, müssen Sie Beziehungen zwischen ihnen herstellen. In diesem Beispiel verfügt sowohl die Tabelle *Sales* als auch die

Tabelle *Product* über die Spalte *ProductKey*. In *Product* ist diese Spalte der *Primärschlüssel*. Das bedeutet, dass sie in jeder Zeile einen anderen Wert aufweist und daher verwendet werden kann, um ein Produkt eindeutig zu identifizieren. In der Tabelle *Sales* dagegen dient diese Spalte einem anderen Zweck, nämlich das verkaufte Produkt anzugeben.

Abbildung 1–7 Mit dem Excel-Datenmodell können Sie mehrere Tabellen laden.

Der Primärschlüssel einer Tabelle ist eine Spalte, die in jeder Zeile einen anderen Wert aufweist. Wenn Sie daher einen Wert aus dieser Spalte kennen, können Sie ihn eindeutig einer Zeile zuordnen. Es kann mehrere Spalten mit solchen eindeutigen Werten geben, wobei alle diese Spalten Schlüssel sind. Der Primärschlüssel ist nichts Besonderes. Technisch gesehen ist er lediglich die Spalte, die Sie zur eindeutigen Bezeichnung einer Zeile verwenden. In einer Kundentabelle kann dies beispielsweise die Kundennummer sein, auch wenn die Werte in der Namensspalte möglicherweise alle eindeutig sind.

Wenn Sie in einer Tabelle einen eindeutigen Bezeichner haben und eine Spalte in einer anderen Tabelle darauf verweist, können Sie eine Beziehung zwischen den beiden Tabellen herstellen. Für eine gültige Beziehung müssen beide Bedingungen gelten. Wenn der gewünschte Schlüssel für die Beziehung in keiner der beiden Tabellen Ihres Modells ein eindeutiger Bezeichner ist, müssen Sie das Modell mit einer der Techniken abwandeln, die Sie in diesem Buch kennenlernen werden. Zunächst einmal wollen wir aber anhand unseres Beispiels einige grundlegende Tatsachen über Beziehungen feststellen:

- **Die Tabelle *Sales* ist die *Quelltabelle*** Die Beziehung geht von *Sales* aus, denn um ein Produkt abzurufen, beginnen Sie stets bei *Sales*: Sie ermitteln den Wert der Produktnummer in *Sales* und suchen anschließend in *Product* danach. Danach kennen Sie das Produkt und all seine Attribute.

- **Die Tabelle *Product* ist das *Ziel* der Beziehung** Das liegt daran, dass Sie bei *Sales* anfangen und dann zu *Product* übergehen. Daher ist *Product* das Ziel Ihrer Suche.
- **Eine Beziehung verläuft von der Quelle zum Ziel** Anderes ausgedrückt: Eine Beziehung hat eine Richtung. Aus diesem Grund werden Beziehungen oft in Form von Pfeilen von der Quelle zum Ziel abgebildet. Allerdings werden in Softwareprodukten unterschiedliche grafische Darstellungen von Beziehungen verwendet.
- **Die Quelltabelle ist die *n-Seite* der Beziehung** Diese Bezeichnung rührt daher, dass es für ein Produkt sehr wahrscheinlich viele Verkaufsvorgänge gibt, für einen Verkaufsvorgang aber immer nur ein Produkt. Daher haben wir hier eine *n:1-Beziehung*, bei der die Zieltabelle die 1-Seite und die Quelltabelle die n-Seite ist.
- **Die Spalte *ProductKey* ist sowohl in *Sales* als auch in *Products* vorhanden** In *Product* ist *ProductKey* ein Schlüssel, aber nicht in *Sales*. Daher wird diese Spalte in *Product* als Primärschlüssel bezeichnet, in *Sales* dagegen als *Fremdschlüssel*, also als eine Spalte, die auf einen Primärschlüssel in einer anderen Tabelle verweist.

All dies sind gebräuchliche Begriffe in der Datenmodellierung, weshalb wir sie auch in diesem Buch verwenden. Machen Sie sich aber keine Sorgen, denn wir werden die Definitionen in den ersten Kapiteln noch einige Male wiederholen, damit Sie sich damit vertraut machen können.

Mit Excel und Power BI können Sie eine Beziehung zwischen zwei Tabellen herstellen, indem Sie den Fremdschlüssel (in unseren Beispielen *ProductKey* in *Sales*) auf den Primärschlüssel (*ProductKey* in *Product*) ziehen. Dabei werden Sie feststellen, dass weder Excel noch Power BI Pfeile zur Darstellung von Beziehungen verwenden. Stattdessen wird die Beziehung in der Diagrammansicht durch eine Zahl (1) für die 1-Seite und ein Sternchen für die n-Seite angezeigt, wie Sie in Abbildung 1–8 sehen. In der Mitte steht zwar auch ein Pfeil, aber er gibt nicht die Richtung der Beziehung an, sondern der Filterweiterleitung, was etwas völlig anderes ist. Damit werden wir uns weiter hinten in diesem Buch noch beschäftigen.

Sollte die Power Pivot-Registerkarte verschwinden, liegt das wahrscheinlich daran, dass Excel ein Problem festgestellt und das Add-In ausgeschaltet hat. Um es wieder zu aktivieren, klicken Sie auf die Registerkarte *Datei*, dann im linken Bereich auf *Optionen* und im linken Teil des Optionenfensters auf *Add-Ins*. Öffnen Sie das Listenfeld *Verwalten* am unteren Rand der Seite, wählen Sie *COM-Add-Ins* und klicken Sie auf *Los*. Wählen Sie anschließend im Fenster *COM-Add-Ins* das Add-In *Microsoft Power Pivot for Excel*. Sollte es bereits ausgewählt sein, wählen Sie es ab. Klicken Sie auf *OK*. Haben Sie die Auswahl von Power Pivot aufgehoben, müssen Sie das Fenster *COM-Add-Ins* erneut aufrufen und das Add-In wieder auswählen. Anschließend sollte die Power Pivot-Registerkarte wieder im Menüband erscheinen.

Wenn eine Beziehung vorhanden ist, können Sie die Werte in der Tabelle *Sales* summieren und mit einem Datenschnitt nach Spalten in der Tabelle *Products* filtern. Wie Sie in Abbildung 1–9 sehen, können Sie die Summe der Stückzahlen (Spalte *Quantity* in *Sales*; siehe Abbildung 1–8) nach der Farbe (Spalte *Color* in *Product*) filtern.

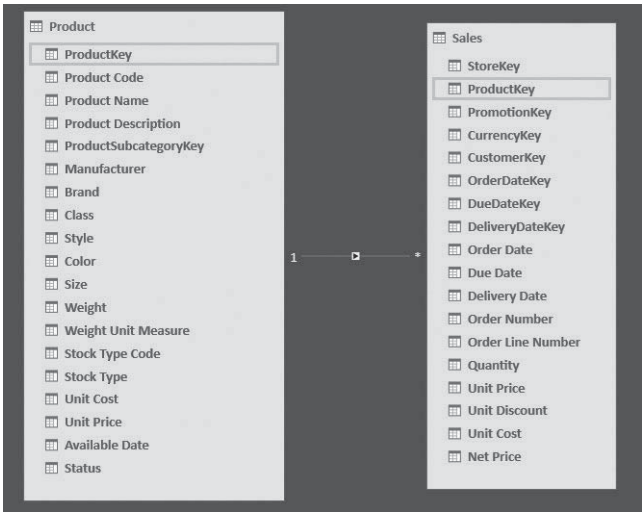


Abbildung 1–8 Eine Beziehung wird als eine Linie (hier zwischen den Tabellen *Product* und *Sales*) mit einer Bezeichnung der Seiten (*1* für die 1-Seite, * für die n-Seite) dargestellt.

Damit haben Sie ein erstes Beispiel für ein Datenmodell mit zwei Tabellen gesehen. Wie bereits erwähnt ist ein Datenmodell lediglich ein Satz von Tabellen (hier *Sales* und *Product*), die über Beziehungen verknüpft sind. Bevor wir zu weiteren Beispielen übergehen, wollen wir uns noch mit der Granularität in dem Fall beschäftigen, dass wir mehrere Tabellen haben.

Row Labels	Sum of Quantity
Azure	60
Black	4307
Blue	985
Brown	453
Gold	155
Green	374
Grey	1551
Orange	179
Pink	600
Purple	10
Red	896
Silver	3604
Silver Grey	143
Transparent	141
White	3746
Yellow	294
Grand Total	**17498**

Abbildung 1–9 Sobald eine Beziehung besteht, können Sie Werte aus einer Tabelle mithilfe eines Datenschnitts nach Spalten in einer anderen Tabelle filtern.

Im ersten Abschnitt dieses Kapitels haben Sie gelernt, wie wichtig – und wie schwierig – es ist, die richtige Granularität für eine einzelne Tabelle festzulegen. Wenn Sie eine falsche Wahl treffen, lassen sich die Berechnungen viel schwerer schreiben. Was aber machen Sie mit der Granularität in dem neuen Datenmodell mit zwei Tabellen? Das Problem ist hier etwas anders gelagert und lässt sich auch in gewissem Maße leichter lösen. Allerdings ist es etwas schwieriger zu verstehen.

Mit zwei Tabellen haben wir auch zwei verschiedene Granularitäten. Die von *Sales* liegt auf der Ebene der einzelnen Verkäufe, die von *Product* auf Produktebene. Die Granularität gilt immer nur für eine Tabelle, nicht für das gesamte Datenmodell. Wenn Sie in Ihrem Modell mehrere Tabellen haben, müssen Sie die Granularität für jede davon einstellen. Das scheint etwas aufwendiger zu sein als bei einer einzigen Tabelle, führt aber zu Modellen, die sich einfacher verwalten lassen und bei denen die Granularität kein Problem mehr darstellt.

Bei zwei Tabellen ist es nur natürlich, etwa die Granularität der Verkaufstabelle auf die Ebene der einfachen Verkäufe und die der Produkttabelle auf – richtig geraten – Produktebene einzustellen. Im ersten Beispiel hatten wir eine einzige Tabelle mit Verkäufen, aber eine Granularität auf der Ebene der Kategorien und Unterkategorien, da die Produktkategorien und -unterkategorien in der Verkaufstabelle gespeichert waren. Daher mussten Sie eine *Entscheidung über die Granularität* treffen, was insbesondere daran lag, dass *Informationen an der falschen Stelle gespeichert* waren. Wenn die einzelnen Informationen an der richtigen Stelle stehen, ist die Granularität kein so großes Problem mehr.

Die Produktkategorie ist ein Attribut des Produkts, nicht eines einzelnen Verkaufsvorgangs. In gewissem Sinne stellt sie auch ein Attribut eines Verkaufs dar, aber nur, da der Verkauf ein Produkt betrifft. Wenn Sie aber die Produktnummer in der Verkaufstabelle speichern, können Sie die Beziehung nutzen, um sämtliche Attribute des Produkts abzurufen, z. B. die Produktkategorie, die Farbe und alle anderen Angaben zum Produkt. Bei dieser Vorgehensweise ist es nicht mehr nötig, die Produktkategorie in der Verkaufstabelle zu speichern, und damit wird auch die Frage der Granularität weniger bedeutsam. Das gilt natürlich für sämtliche Produktattribute – die Farbe, den Stückpreis, den Produktnamen und ganz allgemein sämtliche Spalten in der Produkttabelle.

In einem sauber gestalteten Modell sind die Granularitäten aller Tabellen auf die richtige Ebene eingestellt. Das führt zu einer einfacheren und zugleich leistungsfähigeren Struktur. Das ist es, was Beziehungen so wertvoll macht. Diese Möglichkeiten können Sie nutzen, sobald Sie die klassische Einzeltabellenmethode von Excel aufgeben und mehrere Tabellen einsetzen.

Wenn Sie sich die Tabelle *Products* genauer ansehen, werden Sie feststellen, dass sie keine Kategorien und Unterkategorien mehr enthält. Stattdessen gibt es die Spalte *ProductSubcategoryKey*, deren Name schon andeutet, dass es sich dabei um einen Verweis (einen Fremdschlüssel) auf einen Schlüssel in einer anderen Tabelle handelt (in der diese Spalte der Primärschlüssel ist). Tatsächlich enthält die Datenbank zwei Tabellen für die Produktkategorie und die Produktunterkategorie. Wenn Sie beide in das Modell laden und die richtigen Beziehungen einrichten, erhalten Sie in der Diagrammansicht von Power Pivot die Struktur aus Abbildung 1–10.

Die Informationen über ein Produkt sind also in drei verschiedenen Tabellen gespeichert, nämlich *Product*, *Product Subcategory* und *Product Category*. Dadurch entsteht eine Kette von Beziehungen von *Product* über *Product Subcategory* zu *Product Category*.

Abbildung 1–10 Die Produktkategorien und -unterkategorien sind in separaten Tabellen gespeichert, die über Beziehungen zugänglich sind.

Was ist der Grund für diesen Aufbau? Auf den ersten Blick sieht das nach einer übermäßig komplizierten Vorgehensweise aus, um ganz einfache Informationen zu speichern. Allerdings hat diese Technik viele Vorteile, auch wenn sie nicht auf den ersten Blick offensichtlich sind. Durch die Speicherung der Produktkategorie in einer eigenen Tabelle erhalten Sie ein Datenmodell, in dem der Name der Kategorie nur in einer einzigen Zeile der Tabelle *Product Category* gespeichert ist, obwohl viele Produkte darauf verweisen. Das ist aus zwei Gründen eine gute Vorgehensweise für die Speicherung von Informationen: Erstens verringert es den erforderlichen Speicherplatz, wenn ein und derselbe Name nicht mehrmals gespeichert wird. Zweitens: Wenn Sie den Kategorienamen irgendwann einmal ändern müssen, ist das nur in dieser einzigen Zeile erforderlich. Über die Beziehung erhalten dann alle betroffenen Produkte automatisch den neuen Kategorienamen.

Diese Technik wird als *Normalisierung* bezeichnet. Man sagt, ein Attribut wie hier die Produktkategorie ist normalisiert, wenn es in einer eigenen Tabelle gespeichert und durch einen Schlüssel ersetzt wird, der auf diese Tabelle verweist. Das ist eine Standardtechnik, die Datenbankdesigner beim Entwerfen eines Datenmodells routiniert einsetzen. Die gegenteilige Vorgehensweise – die Speicherung von Attributen in der Tabelle, zu der sie gehören – wird als *Denormalisierung* bezeichnet. Wenn ein Modell denormalisiert ist, erscheinen die Attribute jeweils an mehreren Stellen, und bei einer Aktualisierung müssen Sie alle Zeilen anfassen, in denen sie vorkommen. In unserem Beispiel ist etwa die Produktfarbe denormalisiert, denn der String *Red* taucht bei allen roten Produkten auf.

Vielleicht fragen Sie sich, warum die Designer der Contoso-Datenbank die Kategorien und Unterkategorien in eigenen Tabellen gespeichert (also normalisiert haben), während Farbe, Hersteller und Marke in der Produkttabelle gespeichert (denormalisiert) sind. In diesem besonderen Fall ist die Antwort ganz einfach: weil Contoso eine Beispieldatenbank ist, deren Struktur verschiedene Designtechniken widerspiegeln soll. In der Praxis – also auch in den Datenbanken Ihrer Organisation – finden Sie mit hoher Wahrscheinlichkeit Datenstrukturen, die entweder stark normalisiert oder stark denormalisiert sind, wobei die Wahl vom Verwendungszweck der Datenbank abhängt. Es kann allerdings durchaus sein, dass einige Attribute normalisiert sind und andere nicht. Das ist nicht ungewöhnlich, denn bei der Datenmodellierung gibt es immer

viele verschiedene Möglichkeiten. Es kann durchaus sein, dass ein Designer im Laufe der Zeit unterschiedliche Entscheidungen treffen muss.

Stark normalisierte Strukturen sind typisch für OLTP-Systeme (Online Transactional Processing). Dabei handelt es sich um Datenbanken für alltägliche Aufgaben wie die Rechnungsstellung, Bestellannahme, Lieferung und Reklamationsbearbeitung. Diese Datenbanken sind normalisiert, um so wenig Speicherplatz wie möglich einzunehmen (wodurch sie gewöhnlich schneller laufen) und viele Einfüge- und Aktualisierungsoperationen zu erlauben. Wenn Sie bei der Routinearbeit in Ihrem Unternehmen Informationen ändern – etwa Angaben zu einem Kunden –, wollen Sie gewöhnlich, dass dabei alle Daten aktualisiert werden, die auf diesen Kunden verweisen. Das geht reibungslos, wenn die Kundeninformationen korrekt normalisiert sind. Alle Bestellungen eines Kunden verweisen dann auf einen Schlag auf die neuen, aktualisierten Informationen. Wären die Kundenangaben denormalisiert, würde der Server bei einer Änderung der Kundenadresse Hunderte von `UPDATE`-Anweisungen ausführen müssen, was die Leistung stark verringert.

OLTP-Systeme bestehen oft aus Hunderten von Tabellen, da fast jedes Attribut in einer eigenen Tabelle gespeichert ist. Bei Produkten gibt es dann beispielsweise eine Tabelle für den Hersteller, eine für die Marke, eine für die Farbe usw. Eine einfache Entität wie ein Produkt kann über zehn oder zwanzig verschiedene Tabellen hinweg gespeichert sein, die alle über Beziehungen verknüpft sind. Datenbankdesigner bezeichnen so etwas als »gut gestaltetes Datenmodell«. Auch wenn es seltsam aussieht, ist ein solches Modell jedoch etwas, worauf die Designer mit Recht stolz sein können. Für OLTP-Datenbanken ist die Normalisierung fast immer eine nützliche Technik.

Bei der Analyse von Daten führen Sie jedoch keine Einfügungen und Aktualisierungen durch, sondern wollen Informationen lesen. Dafür aber ist die Normalisierung fast nie eine geeignete Technik. Stellen Sie sich beispielsweise vor, Sie würden eine Pivottabelle aus dem vorherigen Datenmodell erstellen. Die Liste der Felder könnte so aussehen wie in Abbildung 1–11.

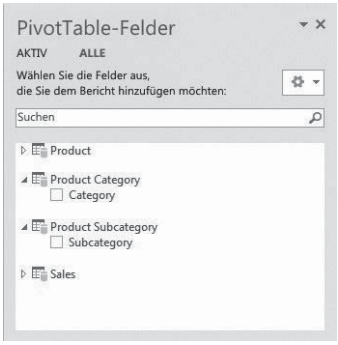


Abbildung 1–11 In einem normalisierten Modell enthält die Felderliste zu viele Tabellen und wird daher leicht unübersichtlich.

Das Produkt ist über drei Tabellen hinweg gespeichert, weshalb die Felderliste (im Dialogfeld *PivotTable-Felder*) drei Tabellen enthält. Was die Sache noch schlimmer macht, ist die Tatsache, dass *Product Category* und *Product Subcategory* nur jeweils eine einzige Spalte enthalten. Für OLTP-Systeme ist Normalisierung daher eine gute Sache, doch zur Analyse stellt sie gewöhnlich

eine schlechte Wahl dar. Wenn Sie versuchen, die Zahlen in einem Bericht zu filtern und zu drehen, sind Sie ja nicht an der technischen Darstellung des Produkts interessiert, sondern wollen die Kategorie und die Unterkategorie als Spalten in der Produkttabelle sehen, was schließlich die natürlichere Art und Weise zur Untersuchung der Daten darstellt.

In diesem Beispiel haben wir absichtlich einige für unsere Zwecke nutzlose Spalten wie die Primärschlüssel der Tabelle ausgeblendet, was immer eine gute Vorgehensweise ist. Sonst würden Sie einige Spalten mehrfach sehen, was es noch schwerer macht, das Modell zu untersuchen. Sie können sich leicht vorstellen, wie die Felderliste mit zehn Tabellen für das Produkt aussehen würde. Die brauchbaren Spalten für den Bericht zu finden, würde viel mehr Zeit kosten.

Unabhängig davon, wie die Originaldaten gespeichert sind, müssen Sie beim Aufbau eines Datenmodells für die Berichterstattung ein gewisses Maß an Denormalisierung durchführen. Wie Sie bereits gesehen haben, führt eine übermäßige Denormalisierung jedoch wieder zu Problemen mit der Granularität. Weiter hinten in diesem Buch werden Sie noch sehen, dass eine übermäßige Denormalisierung noch weitere negative Folgen hat. Was also ist das rechte Maß für die Denormalisierung?

Dafür gibt es keine feste Regel. Sie sollten die Denormalisierung so weit vorantreiben, bis die Tabelle Ihrem Gefühl nach eine eigenständige Struktur ist, die die darin gespeicherte Entität vollständig beschreibt. Bei dem Beispiel aus diesem Abschnitt sollten Sie die Spalten *Product Category* und *Product Subcategory* in die Tabelle *Product* verschieben, da es sich bei ihnen um Attribute des Produkts handelt, die Sie nicht in eigenen Tabellen haben wollen. Denormalisieren Sie aber nicht das Produkt in die Verkaufstabelle, da Produkte und Verkäufe zwei verschiedene Informationen sind. Ein Verkauf bezieht sich zwar auf ein Produkt, kann aber niemals vollständig mit einem Produkt gleichgesetzt werden.

Ein Modell mit nur einer einzigen Tabelle können Sie durchaus als übermäßig denormalisiert auffassen. Schließlich mussten wir uns in der Tabelle *Sales* schon Sorgen über die Granularität auf der Ebene der Produktattribute machen, was nicht richtig ist. Bei einem korrekt gestalteten Modell mit dem richtigen Maß an Denormalisierung ergibt sich die richtige Granularität auf natürliche Weise. Bei einem übermäßig denormalisierten Modell dagegen müssen Sie sich Gedanken über die Granularität machen und haben mit Problemen zu kämpfen.

Sternschemata

Wir haben uns bis jetzt nur sehr einfache Datenmodelle angesehen, in denen es nur um Produkte und Verkäufe ging. In der Praxis sind Datenmodelle jedoch nur sehr selten so einfach. In einem typischen Unternehmen wie Contoso gibt es verschiedene Arten von Informationsgütern wie Produkte, Läden, Mitarbeiter, Kunden und Zeit, die alle miteinander interagieren und Ereignisse auslösen. Beispielsweise wird ein Produkt an einem bestimmten Datum in einem Laden von einem Mitarbeiter an einen Kunden verkauft.

Unterschiedliche Unternehmen haben natürlich mit unterschiedlichen Informationsgütern zu tun, deren Interaktion unterschiedliche Ereignisse hervorruft. Ganz allgemein gesehen gibt es jedoch fast immer eine klare Unterscheidung zwischen den Gütern und den Ereignissen. Dieses Muster zeigt sich in allen Geschäftszweigen, auch bei sehr verschiedenen Informationsgütern. Im medizinischen Bereich etwa schließen die Güter Patienten, Krankheiten und Medikamente ein, wobei ein Ereignis darin besteht, dass ein Patient mit einer bestimmten Krankheit diagnostiziert wird und dafür Medikamente erhält. In einem Reklamationssystem dagegen kann es um Kunden, Beanstandungen und Zeit gehen, wobei die Ereignisse die verschiedenen Stadien sind, die die Beanstandung durchläuft. Nehmen Sie sich die Zeit, über Ihre Branche nachzudenken. Sehr wahrscheinlich können Sie auch dort klar zwischen den Informationsgütern und den Ereignissen trennen.

Diese Trennung führt zu einer Datenmodellierungstechnik, die als *Sternschema* bezeichnet wird. In einem solchen Schema gibt es zwei Kategorien von Entitäten (Tabellen):

- **Dimensionen** Eine *Dimension* ist ein Informationsgut, z. B. ein Produkt, ein Kunde, ein Angestellter oder ein Patient. Dimensionen weisen Attribute auf. Beispielsweise hat ein Produkt Attribute wie Farbe, Kategorie, Unterkategorie, Hersteller und Preis, ein Patient dagegen Attribute wie Name, Anschrift und Geburtsdatum.
- **Fakten** Ein *Faktum* ist ein Ereignis, an dem mehrere Dimensionen beteiligt sind. Bei Contoso ist der Verkauf eines Produkts ein Faktum. Daran beteiligt sind ein Produkt, ein Kunde, ein Datum und weitere Dimensionen. Fakten haben Maße (Metriken). Dies sind Zahlen, die Sie aggregieren können, um Erkenntnisse über Ihre Geschäfte zu gewinnen. Beispiele für solche Metriken sind die Zahl der verkauften Einheiten, der Gesamtbetrag der Verkäufe, Rabatte usw.

Wenn Sie Ihre Tabellen im Kopf in diese beiden Kategorien aufteilen, wird es deutlich, dass die Fakten mit den Dimensionen zusammenhängen. Für ein einzelnes Produkt gibt es viele Verkäufe. Mit anderen Worten, es gibt eine n:1-Beziehung zwischen den Tabellen *Sales* und *Products*, wobei *Sales* die n-Seite und *Products* die 1-Seite darstellt. Wenn Sie alle Dimensionen um eine einzige Faktentabelle anordnen, erhalten Sie die typische Form eines Sternschemas. Abbildung 1–12 zeigt ein solches Schema in der Diagrammansicht von Power Pivot.

Sternschemata lassen sich leicht lesen, verstehen und verwenden. Sie verwenden die Dimensionen, um die Daten zu filtern und zu drehen, und die Faktentabelle, um Zahlen zu aggregieren. Überdies weist die Felderliste für die Pivottabelle weniger Einträge auf.

Sternschemata sind im Bereich der Data Warehouses sehr beliebt. Heutzutage gelten Sie als eine Standardmöglichkeit zur Darstellung von analytischen Modellen.

Dimensionen sind naturgemäß eher kleine Tabellen mit weniger als 1.000.000 Zeilen, gewöhnlich in einer Größenordnung von wenigen Hundert oder Tausend Zeilen. Faktentabellen dagegen sind viel größer und sollen Dutzende, wenn nicht gar Hunderte von Millionen Zeilen aufnehmen. Abgesehen davon sind Sternschemata so weit verbreitet, dass die meisten Datenbanksysteme über besondere Optimierungen verfügen, die bei der Arbeit mit Sternschemata besonders wirkungsvoll sind.

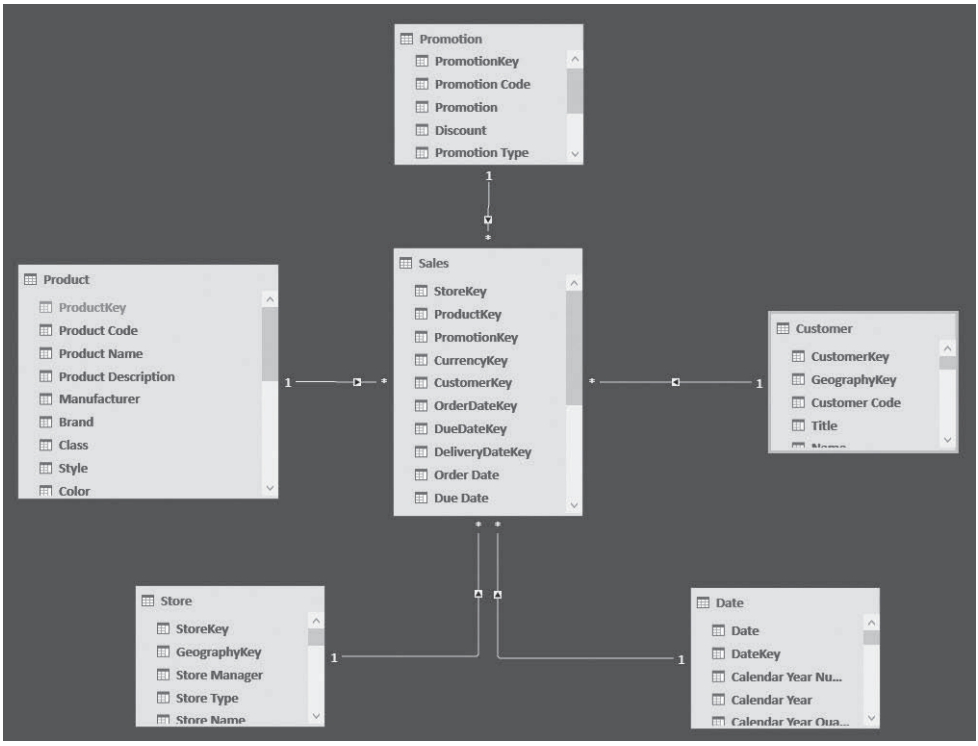


Abbildung 1–12 Wenn Sie die Faktentabelle in die Mitte stellen und alle Dimensionen rundherum anordnen, wird das Sternschema sichtbar.

Bevor Sie weiterlesen, sollten Sie etwas Zeit darauf verwenden, sich zu überlegen, wie sich Ihr Geschäftsmodell als Sternschema darstellen lässt. Sie müssen jetzt natürlich nicht das perfekte Sternschema gestalten. Diese Übung soll Ihnen helfen, Faktentabellen und Dimensionen besser konstruieren zu können.

Sie sollten sich unbedingt mit Sternschemata vertraut machen, da sie eine komfortable Möglichkeit zur Darstellung Ihrer Daten bieten. Außerdem wird im Zusammenhang mit Business Intelligence häufig Sternschemata-Terminologie verwendet, und dieses Buch bildet da keine Ausnahme. Wir verwenden häufig Begriffe wie Faktentabellen und Dimensionen, um zwischen größeren und kleineren Tabellen zu unterscheiden. Beispielsweise behandeln wir im nächsten Kapitel die Verwendung von Header/Detail-Tabellen, wobei es um das allgemeinere Problem geht, Beziehungen zwischen mehreren Faktentabellen herzustellen. An dieser Stelle setzen wir voraus, dass Sie die grundlegenden Unterschiede zwischen Faktentabellen und Dimensionen kennen.

Sternschemata weisen noch weitere wichtige Eigenschaften auf. So stehen zwar die Faktentabellen in Beziehungen zu den Dimensionen, doch sollte es keine Beziehungen zwischen den Dimensionen geben. Um Ihnen zu zeigen, warum diese Regel so wichtig ist und was geschieht, wenn Sie sie brechen, fügen wir unserem Beispiel als weitere Dimension *Geography* hinzu, die

Angaben zu geografischen Orten wie Stadt, Bundesstaat und Region enthält. Die Dimensionen *Store* (für die Läden) und *Customer* (für die Kunden) lassen sich beide in Beziehung zu *Geography* setzen. Damit könnten Sie ein Modell wie das aus Abbildung 1–13 aufbauen.

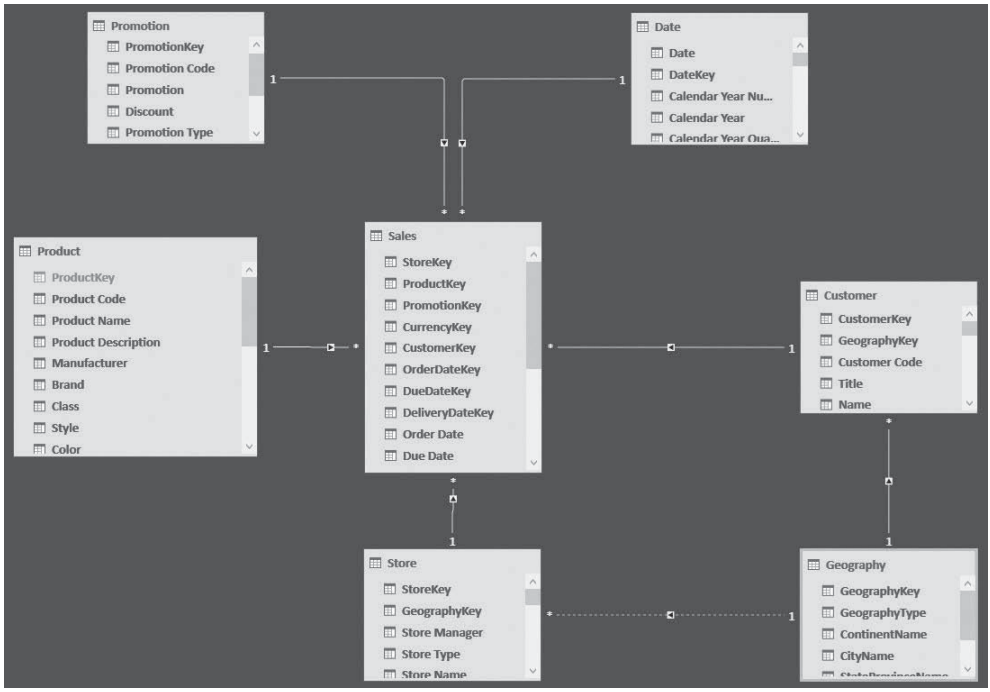


Abbildung 1–13 Die neue Dimension *Geography* hat Beziehungen zu den Dimensionen *Customer* und *Stores*.

Dieses Modell verletzt jedoch die Regel, dass Dimensionen keine Beziehungen untereinander haben sollten. Die drei Tabellen *Customer*, *Store* und *Geography* sind allesamt Dimensionen, aber trotzdem durch Beziehungen verbunden. Was ist so schlecht daran? Nun, hierdurch entsteht eine *Mehrdeutigkeit*.

Nehmen wir an, Sie möchten einen Datenschnitt nach der Stadt durchführen und dafür den Gesamtbetrag der Verkäufe berechnen. Das System kann dazu der Beziehung zwischen *Geography* und *Customer* folgen und den Verkaufsbetrag für die Stadt ermitteln, in der der Kunde wohnt. Genauso gut aber könnte es auch der Beziehung zwischen *Geography* und *Store* folgen und den Betrag für die Stadt zurückgeben, in der sich der Laden befindet. Als dritte Möglichkeit könnte es sogar beiden Beziehungen folgen und den Betrag der Verkäufe in Läden einer gegebenen Stadt an Kunden in dieser Stadt berechnen. Das Datenmodell ist mehrdeutig. Es gibt keine einfache Möglichkeit, um herauszufinden, welche Zahl zurückgegeben wird. Das ist nicht nur ein technisches, sondern auch ein logisches Problem. Ein Benutzer, der sich dieses Datenmodell ansieht, wäre verwirrt und wüsste nicht, was die Zahlen nun wirklich bedeuten. Aufgrund dieser Mehrdeutigkeit ist es weder in Excel noch in Power BI möglich, ein solches Modell zu bauen. In nachfolgenden Kapiteln werden wir diese Mehrdeutigkeit noch ausführlicher erläutern. Vorläufig aber ist es nur wichtig, sich zu merken, dass Excel (das Werkzeug, in dem

wir dieses Beispiel konstruiert haben) die Beziehung zwischen *Store* und *Geography* deaktiviert hat, um sicherzustellen, dass das Modell nicht mehrdeutig ist.

Als Datenmodellierer müssen Sie Mehrdeutigkeiten auf jeden Fall vermeiden. Wie könnten Sie nun die Mehrdeutigkeit in dem vorstehenden Beispiel auflösen? Ganz einfach: Sie müssen die entsprechenden Spalten der Tabelle *Geography* denormalisieren und sowohl in *Store* als auch in *Customer* übertragen, um *Geography* aus dem Modell entfernen zu können. Beispielsweise können Sie wie in Abbildung 1–14 die Spalten *ContinentName*, *CityName* usw. sowohl in *Store* als auch in *Customer* aufnehmen.

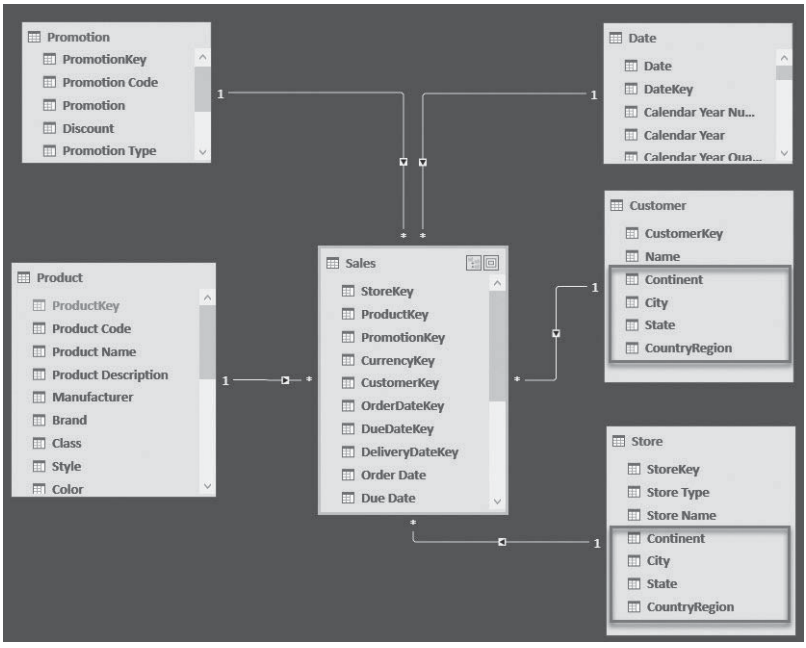


Abbildung 1–14 Die Spalten von *Geography* werden in *Customer* und *Store* denormalisiert, sodass die Tabelle *Geography* nicht mehr in dem Modell vorhanden ist.

Durch richtige Denormalisierung können Sie die Mehrdeutigkeit aufheben. Jetzt können die Benutzer eine Filterung nach geografischen Daten in der Tabelle *Customer* oder *Store* durchführen. Die geografische Verteilung ist zwar eine Dimension, aber um ein korrektes Sternschema zu erhalten, müssen wir sie denormalisieren.

Bevor wir uns einem anderen Thema zuwenden, müssen wir noch einen weiteren wichtigen Begriff einführen, nämlich den der *Schneeflocke*. Dabei handelt es sich um eine Variante des Sternschemas, bei dem eine Dimension nicht unmittelbar mit der Faktentabelle verbunden ist, sondern über eine andere Dimension. Beispiele dafür haben Sie bereits gesehen, und eines davon finden Sie noch einmal in Abbildung 1–15.

Verletzen Schneeflocken die Regel, dass Dimensionen nicht untereinander verbunden sein dürfen? In gewissem Sinne ist das der Fall, da die Beziehung zwischen *Product Subcategory* und *Product* tatsächlich zwei Dimensionen verbindet. Der Unterschied zwischen diesem Fall und dem vorherigen Beispiel besteht jedoch darin, dass dies die einzige Beziehung zwischen

Product Subcategory und den anderen Dimensionen ist, die mit der Faktentabelle (*Product*) verknüpft sind. Sie können sich *Product Subcategory* daher als eine Dimension vorstellen, die mehrere Produkte gruppiert, aber keine anderen Dimensionen oder Fakten. Das Gleiche gilt natürlich auch für *Product Category*. Auch wenn ein Schneeflockenmuster die Regel verletzt, führt es daher nicht zu Mehrdeutigkeiten. Daher stellt ein Schneeflocken-Datenmodell kein Problem dar.

Abbildung 1–15 *Product Category*, *Product Subcategory* und *Product* bilden eine Kette von Beziehungen und damit ein Schneeflockenmuster.

Um Schneeflocken zu vermeiden, können Sie die Spalten der Tabelle, die am weitesten von der Faktentabelle entfernt ist, in eine Tabelle denormalisieren, die näher daran liegt. Manchmal bilden Schneeflockenmuster jedoch eine gute Möglichkeit, um Daten darzustellen. Bis auf eine geringe Leistungsverringerung verursachen sie auch keine Probleme.

Wie Sie in diesem Buch noch sehen werden, sind Sternschemata fast immer die beste Möglichkeit zur Darstellung von Daten. Es gibt zwar einige Situationen, in denen sie nicht die *ideale* Vorgehensweise darstellen, doch wenn Sie an einem Datenmodell arbeiten, ist es immer richtig, es als Sternschema zu gestalten. Selbst wenn es nicht perfekt sein sollte, so ist es doch nahezu perfekt und damit immer eine gute Lösung.

Bei der weiteren Beschäftigung mit Datenmodellierung können Sie in Situationen geraten, in denen es so aussieht, als wäre es am besten, vom Sternschema abzuweichen. Tun Sie es trotzdem nicht. Sternschemata sind aus verschiedenen Gründen immer die nahezu beste Möglichkeit. Leider lassen sich die meisten dieser Gründe erst mit einer gewissen Erfahrung in Datenmodellierung richtig nachvollziehen. Solange Sie diese Erfahrung noch nicht haben, vertrauen Sie einfach den Zehntausenden von BI-Experten in aller Welt, die wissen, dass Sternschemata immer die nahezu beste Lösung darstellen.

Die Wichtigkeit von Namen

Wenn Sie ein Datenmodell aufbauen, laden Sie gewöhnlich Daten aus einer SQL Server-Datenbank oder einer anderen Quelle. Sehr wahrscheinlich haben die Entwickler dieser Datenquelle bereits eine Namenskonvention vorgesehen. Es gibt so viele verschiedene Namenskonventionen, dass man schon fast sagen kann, jeder Entwickler verwendet seine eigene.

Beim Aufbau von Data Warehouses verwenden manche Datenbankingenieure gern Präfixe wie *Dim* für Dimensionen und *Fact* für Faktentabellen. Daher finden Sie häufig Tabellen mit Namen wie *DimCustomer* und *FactSales*. Andere dagegen unterscheiden lieber zwischen Sichten und physischen Tabellen und verwenden *Vw* (»view«) für Sichten und *Tbl* für Tabellen. Wieder andere halten Namen für mehrdeutig und bevorzugen Zahlen, z. B. *Tbl_190_Sales*. Wir könnten noch viele weitere Möglichkeiten nennen, aber eines ist jetzt schon klar: Es gibt viele Normen, und jede weist ihre eigenen Vor- und Nachteile auf.

Wir können darüber diskutieren, ob diese Normen in der Datenbank irgendeinen Sinn haben, aber das würde den Rahmen dieses Buches sprengen. Stattdessen erklären wir, wie Sie mit solchen Normen in dem Datenmodell umgehen, das Sie in Power BI oder Excel untersuchen.

Sie müssen keiner technischen Norm folgen, sondern nur dem gesunden Menschenverstand und dabei die einfache Nutzung im Auge behalten. Beispielsweise wäre es ziemlich nervtötend, ein Datenmodell zu untersuchen, in dem die Tabellen umständliche Namen wie *VwDimCstmr* oder *Tbl_190_FactShpmnt* aufweisen. Solche Namen wirken unnatürlich und sind nicht aussagekräftig. Trotzdem finden wir sie sehr häufig in Datenmodellen. Und dabei geht es hier nur um die Tabellennamen! Bei den Spaltennamen wird der Mangel an Kreativität noch viel schlimmer. Wir können Ihnen nur raten, all diese Namen loszuwerden und aussagekräftige Namen zu verwenden, die deutlich machen, worum es sich bei der Dimension oder der Faktentabelle handelt.

Im Laufe der Jahre haben wir schon viele analytische Systeme konstruiert. Dabei haben wir einige einfache Regeln für Tabellen- und Spaltennamen aufgestellt:

- **Tabellennamen für Dimensionen sollten nur aus der Bezeichnung des Geschäftsguts in Singular- oder Pluralform bestehen** Kunden speichern sie daher in einer Tabelle namens *Kunde* oder *Kunden* (bzw. *Customer* oder *Customers*, wenn das Modell auch für Personen außerhalb des deutschen Sprachraums zugänglich sein soll), Produkte in *Produkt* oder *Produkte* (*Product* oder *Products*). Unserer Meinung nach ist die Singularform zu bevorzugen, da die Abfragen in natürlicher Sprache aus Power BI damit etwas besser funktionieren.
- **Besteht die Bezeichnung des Geschäftsguts aus mehreren Wörtern, so trennen Sie diese durch Groß- und Kleinschreibung** Das gilt natürlich vor allem für englische Tabellennamen. Produktkategorien speichern Sie nach dieser Regel in *ProductCategory*, das Lieferland in *CountryShip* oder *CountryShipment*. Statt der Binnenmajuskel können Sie auch Leerzeichen verwenden, also z. B. *Product Category*, allerdings wird es dadurch etwas schwieriger, DAX-Code zu schreiben. Letzten Endes ist das jedoch Geschmackssache.

- **Tabellennamen für Fakten bestehen aus dem Geschäftsnamen des Faktums, und zwar immer im Plural** Verkäufe werden daher in *Verkäufe* bzw. *Sales* gespeichert, Einkäufe in *Einkäufe* oder *Purchases*. Durch die Verwendung des Plurals verknüpfen Sie in Gedanken viel leichter einen Kunden (Tabelle *Kunde*) mit mehreren Verkäufen (Tabelle *Verkäufe*), was die Natur der 1:n-Beziehung schon auf den ersten Blick auf die Tabellen erkennen lässt.
- **Vermeiden Sie zu lange Namen** Namen wie *CountryOfShipmentOfGoodsWhenSoldBy-Reseller* sind verwirrend. Niemand will so lange Namen lesen. Kürzen Sie sie sinnvoll, indem Sie auf überflüssige Wörter verzichten.
- **Vermeiden Sie zu kurze Namen** Abkürzungen und Akronyme sind zwar im Alltag allgegenwärtig, doch bei der Verwendung in Berichten ist nicht unbedingt klar, was sie bedeuten. Beispielsweise könnten Sie statt *Country of shipment for resellers* einfach *CSR* schreiben, aber jemand, der nicht ständig mit Ihnen zusammenarbeitet, wird nicht wissen, was das bedeuten soll. Denken Sie immer daran, dass Berichte vielen verschiedenen Benutzern zur Verfügung gestellt werden, von denen die meisten Ihre internen Abkürzungen nicht kennen.
- **Der Schlüssel einer Dimension ist der Dimensionsname gefolgt von *Key*** Der Primärschlüssel von *Customer* ist *CustomerKey*. Das Gleiche gilt auch für Fremdschlüssel. Dass eine Spalte ein Fremdschlüssel ist, lässt sich daran erkennen, dass sie in einer Tabelle mit einem anderen Namen gespeichert ist. Beispielsweise ist die Spalte *CustomerKey* in *Sales* ein Fremdschlüssel, der auf die Tabelle *Customer* verweist, während sie in *Customer* der Primärschlüssel ist.

Das sind nur sehr wenige Regeln. Alles andere können Sie selbst entscheiden. Lassen Sie bei den Namen aller anderen Spalten den gesunden Menschenverstand walten. Ein Datenmodell mit sorgfältig gewählten Namen lässt sich einfacher an andere weitergeben. Außerdem fällt es leichter, Fehler oder Probleme in dem Datenmodell zu finden, wenn Sie den üblichen Benennungskonventionen folgen.

Wenn Sie Zweifel über einen Namen haben, fragen Sie sich: »Kann irgendjemand außer mir diesen Namen verstehen?« Glauben Sie nicht, dass Sie der einzige Benutzer Ihrer Berichte sind! Früher oder später müssen Sie den Bericht anderen zur Verfügung stellen, die womöglich einen ganz anderen Hintergrund haben als Sie. Wenn diese Personen Ihre Namen verstehen, sind Sie auf dem richtigen Weg. Wenn nicht, müssen Sie die Namen in Ihrem Modell noch einmal überdenken.

Zusammenfassung

In diesem Kapitel haben Sie die folgenden Grundlagen der Datenmodellierung gelernt:

- Eine einzelne Tabelle ist bereits ein Datenmodell, wenn auch in einfachster Form.
- Wenn Sie nur eine einzige Tabelle haben, müssen Sie die Granularität der Daten festlegen. Die Wahl der richtigen Granularität macht die Berechnungen viel einfacher.
- Der Unterschied zwischen der Arbeit mit einer einzigen und mit mehreren Tabellen besteht darin, dass mehrere Tabellen durch Beziehungen verknüpft sind.
- In einer Beziehung gibt es eine *1-Seite* und eine *n-Seite*, was angibt, auf wie viele Zeilen Sie wahrscheinlich stoßen, wenn Sie der Beziehung folgen. Da es jeweils nur ein Produkt für Verkäufe gibt, ist die Produkttabelle die 1-Seite und die Verkaufstabelle die n-Seite.
- Eine Tabelle, die das Ziel einer Beziehung ist, muss über einen Primärschlüssel verfügen. Das ist eine Spalte mit eindeutigen Werten, die zur Identifizierung einer einzelnen Zeile verwendet werden können. Ist kein Schlüssel vorhanden, kann die Beziehung nicht definiert werden.
- In einem normalisierten Modell sind die Daten in den Tabellen sehr kompakt gespeichert, da ein und derselbe Wert nicht in mehreren Zeilen wiederholt wird. Allerdings erhöht sich bei dieser Struktur die Anzahl der Tabellen.
- In einem denormalisierten Modell gibt es viele Wiederholungen (z. B. wird die Bezeichnung *Rot* bei jedem roten Produkt angegeben), dafür aber weniger Tabellen.
- Normalisierte Modelle werden für OLTP verwendet, denormalisierte Modelle dagegen für die Datenanalyse.
- In einem typischen analytischen Modell wird zwischen Informationsgütern (Dimensionen) und Ereignissen (Fakten) unterschieden. Durch die Klassifizierung aller Entitäten in dem Modell als entweder Fakten oder Dimensionen erhält das Modell die Form eines Sternschemas. Sternschemata sind die am häufigsten verwendete Architektur für analytische Modelle, und das aus gutem Grunde: Sie funktionieren fast immer.

KAPITEL 10

Segmentierungsmodelle

In Kapitel 1 haben Sie gelernt, wie Sie Ihre Daten mit Standardbeziehungen modellieren: Zwei Tabellen werden über eine einzelne Spalte miteinander verknüpft. Später haben wir sogar m:n-Beziehungen mithilfe von Standardbeziehungen eingerichtet. In diesem Kapitel erfahren Sie, wie Sie mithilfe von DAX mit noch anspruchsvolleren Beziehungen zwischen Tabellen umgehen. In Tabellenmodellen lassen sich zwischen den Tabellen einfache und bidirektionale Beziehungen einrichten, was jedoch eine Beschränkung darstellt. Mit DAX können Sie aber weit aufwendigere Modelle mit praktisch allen möglichen Arten von Beziehungen aufbauen, auch virtuellen. In sehr anspruchsvollen Situationen spielt DAX eine große Rolle bei der Definition des Datenmodells.

Als Beispiele zur Veranschaulichung dieser Art von Beziehungen verwenden wir Datenmodelle, bei denen es hauptsächlich um die Segmentierung der Daten geht. Die Segmentierung ist ein gängiges Modellierungsmuster, das immer dann auftritt, wenn Sie Ihre Daten auf der Grundlage einer Konfigurationstabelle schichten wollen. Nehmen wir an, Sie wollen Ihre Kunden nach dem Altersbereich oder nach dem erzielten Umsatz gruppieren oder Ihre Produkte nach dem Verkaufsbetrag.

Wir werden Ihnen in diesem Kapitel keine fertigen Muster an die Hand geben, die Sie auf Ihre eigenen Modelle anwenden können. Stattdessen wollen wir Ihnen ungewöhnliche Vorgehensweisen für den Aufbau komplexer Modelle mit DAX zeigen, Ihre Kenntnisse über Beziehungen vertiefen und Ihnen vorführen, was Sie mit DAX-Formeln alles erreichen können.

Mehrspaltige Beziehungen

Bei der ersten Art von Beziehungen, die wir Ihnen zeigen möchten, handelt es sich um berechnete physische Beziehungen. Der einzige Unterschied zu Standardbeziehungen besteht darin, dass sich der Schlüssel für die Beziehung in einer berechneten Spalte befindet. Wenn Ihnen für eine Beziehung ein Schlüssel fehlt oder wenn Sie ihn mit komplizierten Formeln berechnen müssen, können Sie auf berechnete Spalten zurückgreifen. Das Ergebnis ist immer noch eine physische Beziehung.

Die Tabular-Engine lässt nur Beziehungen auf der Grundlage einer einzigen Spalte zu. Allerdings können Beziehungen, die auf mehreren Spalten beruhen, sehr nützlich sein und treten in vielen Datenmodellen auf. Bei solchen Modellen haben Sie die beiden folgenden Möglichkeiten:

- Definieren Sie eine berechnete Spalte mit einer Kombination der gewünschten Schlüssel und verwenden Sie diese Spalte als neuen Schlüssel für die Beziehung.
- Denormalisieren Sie die Spalten der Zieltabelle (der 1-Seite in der 1:n-Beziehung) mithilfe der Funktion `LOOKUPVALUE`.

Nehmen wir an, Sie haben die Aktion »Produkt des Tages«, bei der einzelne Produkte an einem bestimmten Tag einen besonderen Rabatt erhalten (siehe Abbildung 10–1).

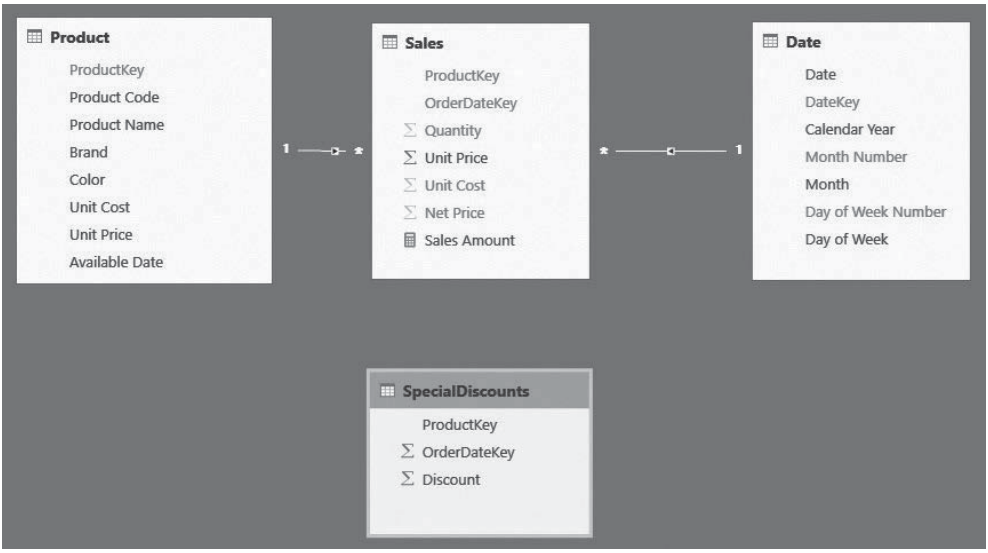


Abbildung 10–1 Die Tabelle *SpecialDiscounts* benötigt eine Beziehung auf der Grundlage von zwei Spalten in *Sales*.

Die Rabatttabelle *SpecialDiscounts* enthält die drei Spalten *ProductKey*, *OrderDateKey* und *Discount*. Bei der Berechnung eines Rabatts haben Sie das Problem, dass dieser Rabatt bei jedem Verkauf sowohl von *ProductKey* als auch von *OrderDateKey* abhängt. Die erforderliche Beziehung zwischen *Sales* und *SpecialDiscounts* können Sie jedoch nicht einrichten, da sie zwei Spalten umfasst, während Tabular nur Beziehungen über eine Spalte zulässt.

Um dieses Problem zu lösen, nutzen Sie die Tatsache, dass es nicht verboten ist, eine Beziehung auf eine berechnete Spalte zu stützen. Sie können also eine neue Spalte erstellen, die die beiden gewünschten Spalten kombiniert, und über diese neue Spalte die Beziehung einrichten. Diese neue Spalte legen Sie sowohl in *SpecialDiscount* als auch in *Sales* an. In *Sales* verwenden Sie dazu den folgenden Code:

```
Sales[SpecialDiscountKey] = Sales[ProductKey] & "-" & Sales[OrderDateKey]
```

In *SpecialDiscount* verwenden Sie einen ähnlichen Ausdruck. Anschließend können Sie die Beziehung zwischen den beiden Tabellen einrichten. Dadurch erhalten Sie das Modell aus Abbildung 10–2.

Diese Lösung ist einfach und funktioniert einwandfrei. Es gibt jedoch Situationen, in denen sie nicht die beste Lösung darstellt, da Sie dazu zwei berechnete Spalten mit vielen verschiedenen Werten erstellen müssen, was im Hinblick auf die Leistung nicht ratsam ist.

Abbildung 10–2 Als Grundlage für eine Beziehung kann auch eine berechnete Spalte dienen.

Eine andere Möglichkeit besteht darin, die Funktion `LOOKUPVALUE` zu verwenden. Damit können Sie den Rabatt unmittelbar in die Faktentabelle denormalisieren, indem Sie mit dem folgenden Code eine neue berechnete Spalte in *Sales* definieren:

```
Sales[SpecialDiscount] =
LOOKUPVALUE (
   SpecialDiscounts[Discount],
   SpecialDiscounts[ProductKey], Sales[ProductKey],
   SpecialDiscounts[OrderDateKey], Sales[OrderDateKey]
)
```

Bei dieser Vorgehensweise erstellen Sie keine Beziehung, sondern verschieben den Rabattwert in die Faktentabelle und führen einen Nachschlagevorgang durch. Technisch ausgedrückt denormalisieren Sie den Wert von *SpecialDiscount* von der Tabelle *SpecialDiscounts* in *Sales*.

Beide Vorgehensweisen sind gut, und für welche Sie sich entscheiden, hängt von mehreren Faktoren ab. Wenn *Discount* die einzige Spalte ist, die Sie von der Tabelle *SpecialDiscounts* benötigen, dann stellt die Denormalisierung die bessere Methode dar. Dabei wird nur eine einzige berechnete Spalte mit weniger einzigartigen Werten erstellt, und nicht zwei mit vielen einzigartigen Werten. Das verringert die Speichernutzung und macht den Code einfacher.

Enthält *SpecialDiscounts* dagegen viele Spalten, die Sie in Ihrem Code benötigen, würde es eine Speicherverschwendung und möglicherweise auch Leistungseinbußen bedeuten, wenn

Sie sie alle in die Faktentabelle denormalisierten. In diesem Fall ist die berechnete Spalte mit dem neuen zusammengesetzten Schlüssel die bessere Methode.

Dieses erste, einfache Beispiel ist vor allem deshalb wichtig, weil es eine wichtige Eigenschaft von DAX veranschaulicht, nämlich die Möglichkeit, Beziehungen auf der Grundlage von berechneten Spalten zu erstellen. Dadurch können Sie jede mögliche Art von Beziehung einrichten, sofern Sie sie nur berechnen und in einer berechneten Spalte materialisieren lassen können. Mit dem nächsten Beispiel zeigen wir Ihnen, wie Sie Beziehungen auf der Grundlage statischer Bereiche einrichten. Durch die Erweiterung dieses Prinzips lassen sich fast alle Arten von Beziehungen aufstellen.

Statische Segmentierung

Statische Segmentierung kommt sehr häufig vor, wenn Sie einen Wert in einer Tabelle haben und weniger an der Analyse des Wertes an sich interessiert sind (da es Hunderte und Tausende von möglichen Werten geben kann), sondern ihn in Segmente aufteilen wollen. Zwei gängige Beispiele dafür sind die Analyse von Verkäufen nach dem Alter der Kunden oder nach dem Listenpreis. Es wäre sinnlos, die Verkaufsbeträge auf alle möglichen Werte des Listenpreises aufzuteilen, da es zu viele davon gibt. Wenn Sie die verschiedenen Preise aber zu Bereichen gruppieren, können Sie aus der Untersuchung dieser Gruppen wertvolle Erkenntnisse gewinnen.

Betrachten Sie als Beispiel die Tabelle *PriceRanges* mit den Preisbereichen. Wie Sie in Abbildung 10–3 sehen, legen Sie dabei für jeden Bereich die Grenzen fest.

PriceRangeKey	PriceRange	MinPrice	MaxPrice
1	VERY LOW	*0*	*10*
2	LOW	*10*	*30*
3	MEDIUM	*30*	*80*
4	HIGH	*80*	*150*
5	VERY HIGH	*150*	*99999*

Abbildung 10–3 Die Konfigurationstabelle für die Preisbereiche

Wie im vorherigen Beispiel können Sie auch hier keine direkte Beziehung zwischen der Faktentabelle mit den Verkäufen und der Konfigurationstabelle *PriceRanges* aufstellen, da der Schlüssel in der Konfigurationstabelle auf einer Bereichsbeziehung beruht, diese Art von Beziehung in DAX aber nicht zulässig ist. Daher besteht die beste Lösung darin, den Preisbereich mithilfe einer berechneten Spalte direkt in die Faktentabelle zu denormalisieren. Das Codemuster dafür ähnelt dem vorherigen, wobei der Hauptunterschied in der folgenden Formel besteht:

```
Sales[PriceRange] =
CALCULATE (
   VALUES ( PriceRanges[PriceRange] ),
   FILTER (
      PriceRanges,
      AND (
         PriceRanges[MinPrice] <= Sales[Net Price],
```

```
            PriceRanges[MaxPrice] > Sales[Net Price]
        )
    )
)
```

Beachten Sie, dass `VALUES` in diesem Code zum Abruf eines einzelnen Werts verwendet wird, obwohl diese Funktion normalerweise eine Tabelle zurückgibt. Wenn eine Tabelle jedoch nur aus einer einzigen Zeile und einer einzigen Spalte besteht, wird sie automatisch in einen Skalar umgewandelt, wenn das in dem Ausdruck erforderlich ist.

Aufgrund der Art und Weise, wie die Funktion `FILTER` das Ergebnis berechnet, gibt sie immer eine einzelne Zeile aus der Konfigurationstabelle zurück. Daher gibt auch `VALUES` immer eine einzige Zeile zurück, und das Ergebnis von `CALCULATE` ist die Beschreibung des Preisbereichs mit dem aktuellen Nettopreis. Dieser Ausdruck funktioniert problemlos, wenn die Konfigurationstabelle ordnungsgemäß aufgebaut ist. Sollten die Preisbereiche aber aus irgendeinem Grund Lücken oder Überlappungen aufweisen, gibt `VALUES` mehrere Zeilen zurück, sodass der Ausdruck zu einem Fehler führt.

Um den vorstehenden Code zu verbessern, können Sie eine Fehlerbehandlungsfunktion nutzen, die eine unsaubere Konfiguration erkennt und eine entsprechende Meldung ausgibt:

```
Sales[PriceRange] =
VAR ResultValue =
    CALCULATE (
        IFERROR (
            VALUES ( PriceRanges[PriceRange] ),
            "Overlapping Configuration"
        ),
        FILTER (
            PriceRanges,
            AND (
                PriceRanges[MinPrice] <= Sales[Net Price],
                PriceRanges[MaxPrice] > Sales[Net Price]
            )
        )
    )
RETURN
    IF (
        ISEMPTY ( ResultValue ),
        "Wrong Configuration",
        ResultValue
    )
```

Dieser Code erkennt sowohl sich überlappende Werte (mit dem internen `IFERROR`) als auch Lücken in der Konfiguration (durch Überprüfung des Ergebnisses mit `ISEMPTY` vor dem Rücksprung zum Aufrufer). Da hiermit garantiert ist, dass immer ein guter Wert zurückgegeben wird, ist der Code jetzt viel sicherer als in der vorherigen Version.

Berechnete physische Beziehungen sind ein sehr vielseitiges Instrument für die Modellierung in Power BI und Excel, da Sie damit sehr anspruchsvolle Beziehungen einrichten können. Außerdem erfolgt die Berechnung der Beziehung während der Aktualisierung der Daten und nicht bei der Abfrage des Modells. Unabhängig von der Komplexität der Beziehungen ergibt sich dadurch eine sehr gute Leistung.

Dynamische Segmentierung

In vielen Situationen können Sie die logische Beziehung zwischen den Tabellen nicht statisch einrichten. In diesen Fällen ist es nicht möglich, berechnete statische Beziehungen zu verwenden. Stattdessen müssen Sie die Beziehung in dem berechneten Feld definieren, um sie dynamisch handhaben zu können. Da die Beziehungen dann nicht Bestandteile des Modells sind, sprechen wir hier im Gegensatz zu den bisher betrachteten physischen Beziehungen von *virtuellen Beziehungen*.

In unserem nächsten Beispiel führen wir eine virtuelle Beziehung als Variante der zuvor gezeigten statischen Segmentierung vor. Bei der statischen Segmentierung hatten wir jeden Verkauf mithilfe einer berechneten Spalte einem bestimmten Segment zugeordnet. Bei der dynamischen Segmentierung erfolgt diese Zuordnung dynamisch.

Nehmen wir an, Sie wollen Ihre Kunden nach dem Verkaufsvolumen gruppieren. Da das Verkaufsvolumen aber jeweils von den in dem Bericht verwendeten Datenschnitten abhängt, kann die Segmentierung nicht statisch sein. Beispielsweise kann ein Kunde bei der Filterung nach einem einzelnen Jahr zu der einen Gruppe gehören und bei der Filterung nach einem anderen Jahr zu einer anderen. Daher können Sie zur Lösung keine physische Beziehung heranziehen oder das Datenmodell abwandeln, um den DAX-Code zu vereinfachen. Die einzige Möglichkeit in einem solchen Fall besteht darin, die Ärmel hochzukrempeln und anspruchsvollen DAX-Code zur Berechnung des Wertes zu schreiben.

Als Erstes definieren Sie die Konfigurationstabelle *Segments*, die Sie in Abbildung 10–4 sehen.

SegmentCode	Segment	MinSale	MaxSale
1	Very Low	0	75
2	Low	75	100
3	Medium	100	500
4	High	500	1000
6	Very High	1000	99999999

Abbildung 10–4 Die Konfigurationstabelle für die dynamische Segmentierung

Was wir berechnen wollen, ist die Anzahl der Kunden, die zu einer bestimmten Gruppe gehören, wobei alle Filter im aktuellen Filterkontext berücksichtigt werden müssen. Die folgende Formel sieht zwar harmlos aus, erfordert aber doch Aufmerksamkeit, da sie einen Kontextübergang enthält:

```
CustInSegment :=
COUNTROWS (
   FILTER (
      Customer,
      AND (
         [Sales Amount] > MIN ( Segments[MinSale] ),
         [Sales Amount] <= MAX ( Segments[MaxSale] )
      )
   )
)
```

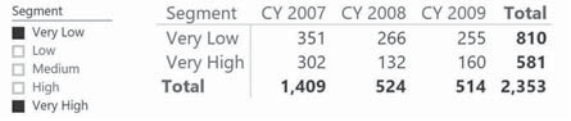
Um das Verhalten dieser Formel zu verstehen, sehen Sie sich den Bericht aus Abbildung 10–5 an, der in den Zeilen die Segmente und in den Spalten die Kalenderjahre anzeigt.

Segment	CY 2007	CY 2008	CY 2009	Total
Very Low	351	266	255	**810**
Low	141	14	12	**166**
Medium	365	76	52	**485**
High	250	36	35	**311**
Very High	302	132	160	**581**
Total	**1,409**	**524**	**514**	**2,353**

Abbildung 10–5 Diese Pivottabelle zeigt die dynamische Segmentierung in Aktion.

Schauen Sie sich die Zelle an, die für das Jahr 2008 in der Gruppe mit mittlerem Verkaufsvolumen (*Medium*) 76 Kunden ausgibt. Die Formel iteriert über *Customer* und prüft für jeden Kunden, ob der Wert von *Sales Amount* zwischen das `MIN` von *MinSale* und das `MAX` von *MaxSale* fällt. Der Wert von *Sales Amount* stellt aufgrund des Kontextübergangs die Verkäufe für den einzelnen Kunden dar. Wie zu erwarten, ist das resultierende berechnete Feld für die Segmente und Kunden additiv, für alle anderen Dimensionen aber nicht additiv.

Die Formel funktioniert nur, wenn Sie alle Segmente auswählen. Bei der Auswahl von beispielsweise nur *Very Low* und *Very High* (also bei Wegfall der drei Zwischensegmente) geben `MIN` und `MAX` nicht die korrekte Auswahl wieder, da sie alle Kunden einbeziehen. Das führt zu falschen Gesamtsummen, wie Abbildung 10–6 zeigt.



Segment	CY 2007	CY 2008	CY 2009	Total
Very Low	351	266	255	**810**
Very High	302	132	160	**581**
Total	**1,409**	**524**	**514**	**2,353**

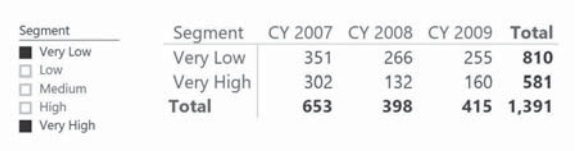
Abbildung 10–6 Falsche Werte bei der Verwendung eines Datenschnitts mit nicht zusammenhängender Auswahl

Wollen Sie den Benutzer einzelne Segmente auswählen lassen, müssen Sie die Formel daher wie folgt umschreiben:

```
CustInSegment :=
SUMX (
   Segments,
```

```
        COUNTROWS (
            FILTER (
                Customer,
                AND (
                    [Sales Amount] > Segments[MinSale],
                    [Sales Amount] <= Segments[MaxSale]
                )
            )
        )
    )
```

Diese Version der Formel leidet zwar nicht unter dem möglichen Problem einer Teilauswahl der Segmente, doch aufgrund der doppelten Iteration über die Tabellen kann die Leistung leiden. Wie Sie in Abbildung 10–7 sehen, sind die Ergebnisse jetzt aber korrekt.



Segment	CY 2007	CY 2008	CY 2009	Total
Very Low	351	266	255	**810**
Very High	302	132	160	**581**
Total	**653**	**398**	**415**	**1,391**

Abbildung 10–7 Das berechnete Feld zeigt jetzt andere Gesamtsummen, da die Teilauswahl der Segmente diesmal korrekt berücksichtigt wird.

Virtuelle Beziehungen bieten viele Möglichkeiten. Auch wenn sie den Benutzern als echte Beziehungen erscheinen, sind sie keine Bestandteile des Modells, sondern werden jedes Mal mit dem DAX-Code berechnet. Ist die dazu erforderliche Formel sehr aufwendig oder das Modell zu umfangreich, kann die Leistung beeinträchtigt werden. Für Modelle mittlerer Größe sind solche Beziehungen jedoch hervorragend geeignet.

Versuchen Sie, diese Prinzipien auf Ihr Geschäftsfeld zu übertragen, um zu sehen, ob dieses Muster auch für die von Ihnen benötigten Schichtungen geeignet ist.

ABC-Analyse

Berechnete Spalten werden in der Datenbank gespeichert. Für unsere Zwecke ist das von enormer Bedeutung, da es neue Möglichkeiten für die Modellierung eröffnet. In diesem Abschnitt sehen wir uns einige Aufgaben an, die sich mit berechneten Spalten sehr wirtschaftlich lösen lassen.

Als Beispiel dafür führen wir Ihnen die ABC-Analyse mit Power BI vor. Die ABC-Analyse basiert auf dem Pareto-Prinzip und ist eine weitverbreitete Technik, um das Kerngeschäft eines Unternehmens zu bestimmen, gewöhnlich in Form der besten Produkte oder besten Kunden. In diesem Beispiel wollen wir uns auf die Produkte konzentrieren.

Das Ziel der ABC-Analyse besteht darin, die Produkte herauszufinden, die eine erhebliche Auswirkung auf das Gesamtgeschäft haben, sodass die Manager ihre Anstrengungen darauf

konzentrieren können. Dazu wird jedes Produkt wie folgt einer der drei Kategorien A, B oder C zugewiesen:

- Produkte in Klasse A machen 70 % der Erträge aus.
- Produkte in Klasse B machen 20 % der Erträge aus.
- Produkte in Klasse C machen 10 % der Erträge aus.

Die Klasse eines Produkts muss in einer berechneten Spalte gespeichert werden, da Sie sie zur Analyse der Produkte verwenden und die Informationen danach filtern wollen. Ein Beispiel dafür sehen Sie in Abbildung 10–8. Sie zeigt eine einfache Pivottabelle, deren Zeilen für die einzelnen Klassen stehen.

ABC Class	NumOfProducts	Margin
A	215	$1,411,868.11
B	285	$404,299.99
C	2,017	$202,448.10
Total	**2,517**	**$2,018,616.20**

Abbildung 10–8 Dieser Bericht zeigt die Anzahl der Produkte und die Gewinnmargen nach Klasse an.

Wie so häufig bei der ABC-Analyse befinden sich auch hier nur wenige Produkte in Klasse A. Sie bilden das Kerngeschäft von Contoso. Produkte in Klasse B sind weniger bedeutend, für das Unternehmen aber immer noch unverzichtbar. Die Produkte in Klasse C dagegen bieten sich zur Herausnahme aus dem Angebot an, da sie im Vergleich zu den Kernprodukten nur sehr wenig zu den Einnahmen beitragen, obwohl sie so zahlreich sind.

Wie Sie in Abbildung 10–9 sehen, ist das Datenmodell für dieses Beispiel recht einfach. Sie brauchen dazu lediglich die Verkäufe und die Produkte.

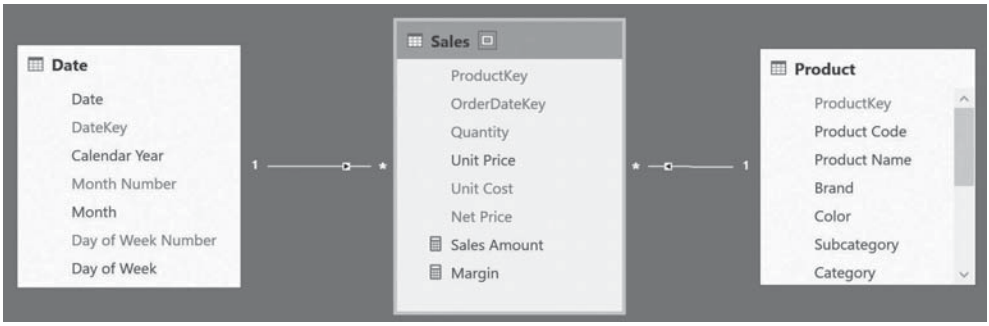


Abbildung 10–9 Das Datenmodell zur Berechnung der Produktklassen ist sehr einfach.

Wir wollen dieses Modell jetzt ändern, indem wir weitere Spalten hinzufügen. Neue Tabellen oder Beziehungen brauchen wir dagegen nicht. Um die Klasse der Produkte zu bestimmen, müssen Sie jeweils die Gesamtmarge für ein Produkt berechnen und mit dem Gesamtertrag vergleichen. Dadurch erhalten Sie den prozentualen Anteil des Produkts am Gesamtertrag. Anschließend sortieren Sie alle Produkte nach diesem prozentualen Anteil und ermitteln eine kumulierte Summe. Sobald diese kumulierte Summe 70 % beträgt, haben Sie die Produkte in Klasse A identifiziert. Die restlichen Produkte fallen in Klasse B, bis 90 % erreicht sind (70 + 20),

und alle weiteren Produkte gehören zu Klasse C. Die Berechnung erfolgt ausschließlich mithilfe berechneter Spalten.

Als Erstes benötigen Sie in der Tabelle *Product* eine berechnete Spalte mit der Marge für jedes Produkt. Das können Sie mit dem folgenden Ausdruck leicht erledigen:

```
Product[TotalMargin] =
SUMX (
   RELATEDTABLE( Sales ),
   Sales[Quantity] * ( Sales[Net Price] - Sales[Unit Cost] )
)
```

Abbildung 10–10 zeigt die Tabelle *Product* mit der neuen berechneten Spalte. Die Daten sind hier in absteigender Reihenfolge nach *TotalMargin* geordnet.

Product Name	TotalMargin
Adventure Works 26" 720p LCD HDTV M140 Silver	*$81,856.27*
Contoso Telephoto Conversion Lens X400 Silver	*$53,464.04*
Fabrikam Refrigerator 24.7CuFt X9800 White	*$51,574.26*
A. Datum SLR Camera X137 Grey	*$51,459.16*
Litware Refrigerator 24.7CuFt X980 Brown	*$29,756.33*
Litware Refrigerator 24.7CuFt X980 White	*$28,256.56*
NT Washer & Dryer 27in L2700 Blue	*$26,591.59*
Proseware Projector 1080p DLP86 Black	*$25,065.45*
NT Washer & Dryer 24in M2400 Green	*$24,472.50*
SV 16xDVD M360 Black	*$20,989.22*
Contoso Projector 1080p X980 White	*$19,648.68*

Abbildung 10–10 *TotalMargin* ist eine berechnete Spalte in der Tabelle *Product*.

Der nächste Schritt besteht darin, für die gesamte nach *TotalMargin* geordnete Tabelle *Product* die kumulierte Summe von *TotalMargin* zu berechnen. Die kumulierte Summe für ein Produkt ist dabei jeweils die Summe der Produktmargen aller Produkte, deren *TotalMargin*-Wert größer oder gleich dem des aktuellen Produkts ist. Dazu verwenden Sie die folgende Formel:

```
Product[MarginRT] =
VAR
   CurrentTotalMargin = 'Product'[TotalMargin]
RETURN
   SUMX (
      FILTER (
         'Product',
         'Product'[TotalMargin] >= CurrentTotalMargin
      ),
   'Product'[TotalMargin]
   )
```

In Abbildung 10–11 sehen Sie die Tabelle *Product* mit dieser neuen Spalte.

Product Name	TotalMargin ↑	MarginRT
Adventure Works 26" 720p LCD HDTV M140 Silver	$81,856.27	$81,856.27
Contoso Telephoto Conversion Lens X400 Silver	$53,464.04	$135,320.31
Fabrikam Refrigerator 24.7CuFt X9800 White	$51,574.26	$186,894.56
A. Datum SLR Camera X137 Grey	$51,459.16	$238,353.72
Litware Refrigerator 24.7CuFt X980 Brown	$29,756.33	$268,110.06
Litware Refrigerator 24.7CuFt X980 White	$28,256.56	$296,366.62
NT Washer & Dryer 27in L2700 Blue	$26,591.59	$322,958.20
Proseware Projector 1080p DLP86 Black	$25,065.45	$348,023.65
NT Washer & Dryer 24in M2400 Green	$24,472.50	$372,496.15
SV 16xDVD M360 Black	$20,989.22	$393,485.38
Contoso Projector 1080p X980 White	$19,648.68	$413,134.06

Abbildung 10–11 *MarginRT* enthält die kumulierte Summe (»running total«) für die nach *TotalMargin* sortierten Zeilen.

Nun müssen Sie die kumulierte Summe als prozentualen Anteil an der Gesamtmarge berechnen. Das lässt sich mit einer weiteren berechneten Spalte leicht erledigen. Fügen Sie mit der folgenden Formel die Spalte *RunningPct* hinzu:

```
Product[MarginPct] = DIVIDE ( 'Product'[MarginRT], SUM ( 'Product'[TotalMargin] )
)
```

Abbildung 10–12 zeigt die neue berechnete Spalte. Um das Ergebnis aussagekräftiger zu machen, wurde sie als Prozentangabe formatiert.

Product Name	TotalMargin ↑	MarginRT	MarginPct
Adventure Works 26" 720p LCD HDTV M140 Silver	$81,856.27	$81,856.27	4.06 %
Contoso Telephoto Conversion Lens X400 Silver	$53,464.04	$135,320.31	6.70 %
Fabrikam Refrigerator 24.7CuFt X9800 White	$51,574.26	$186,894.56	9.26 %
A. Datum SLR Camera X137 Grey	$51,459.16	$238,353.72	11.81 %
Litware Refrigerator 24.7CuFt X980 Brown	$29,756.33	$268,110.06	13.28 %
Litware Refrigerator 24.7CuFt X980 White	$28,256.56	$296,366.62	14.68 %
NT Washer & Dryer 27in L2700 Blue	$26,591.59	$322,958.20	16.00 %
Proseware Projector 1080p DLP86 Black	$25,065.45	$348,023.65	17.24 %
NT Washer & Dryer 24in M2400 Green	$24,472.50	$372,496.15	18.45 %
SV 16xDVD M360 Black	$20,989.22	$393,485.38	19.49 %
Contoso Projector 1080p X980 White	$19,648.68	$413,134.06	20.47 %

Abbildung 10–12 *MarginPct* berechnet den prozentualen Anteil der kumulierten Summe am Gesamtertrag.

Als Letztes müssen Sie jetzt noch die Prozentangaben in die Klassenzugehörigkeit übersetzen. Mit den zuvor genannten Werten 70, 20 und 10 ergibt sich die folgende einfache Formel:

```
Product[ABC Class] =
IF (
   'Product'[MarginPct] <= 0.7,
   "A",
   IF (
      'Product'[MarginPct] <= 0.9,
      "B",
      "C"
   )
)
```

Das Ergebnis sehen Sie in Abbildung 10–13.

Product Name	TotalMargin	MarginRT	MarginPct	ABC Class
Adventure Works 26" 720p LCD HDTV M140 Silver	$81,856.27	$81,856.27	4.06 %	A
Contoso Telephoto Conversion Lens X400 Silver	$53,464.04	$135,320.31	6.70 %	A
Fabrikam Refrigerator 24.7CuFt X9800 White	$51,574.26	$186,894.56	9.26 %	A
A. Datum SLR Camera X137 Grey	$51,459.16	$238,353.72	11.81 %	A
Litware Refrigerator 24.7CuFt X980 Brown	$29,756.33	$268,110.06	13.28 %	A
Litware Refrigerator 24.7CuFt X980 White	$28,256.56	$296,366.62	14.68 %	A
NT Washer & Dryer 27in L2700 Blue	$26,591.59	$322,958.20	16.00 %	A
Proseware Projector 1080p DLP86 Black	$25,065.45	$348,023.65	17.24 %	A
NT Washer & Dryer 24in M2400 Green	$24,472.50	$372,496.15	18.45 %	A
SV 16xDVD M360 Black	$20,989.22	$393,485.38	19.49 %	A
Contoso Projector 1080p X980 White	$19,648.68	$413,134.06	20.47 %	A

Abbildung 10–13 Das Ergebnis der Formel für die Klasseneinteilung ist die berechnete Spalte *ABC Class*.

Da *ABC Class* eine berechnete Spalte ist, wird sie in der Datenbank gespeichert und kann für Datenschnitte, Filter, Zeilen und Spalten verwendet werden, um interessante Berichte zu erstellen.

Wie dieses Beispiel zeigt, können Sie in dem Modell anspruchsvolle Berechnungen speichern, indem Sie berechnete Spalten verwenden und diese systematisch ausführen. Es braucht etwas Erfahrung, bis Sie erkennen können, ob für eine Berechnung eine berechnete Spalte oder ein berechnetes Feld besser geeignet ist, aber sobald Sie sich damit vertraut gemacht haben, können Sie die Möglichkeiten nutzen, die berechnete Spalten bieten.

Weitere Informationen zur ABC-Analyse finden Sie auf *https://de.wikipedia.org/wiki/ABC-Analyse*.

Zusammenfassung

In diesem Kapitel sind wir einen Schritt über Standardbeziehungen hinausgegangen, indem wir Segmentierungstechniken betrachtet haben, die erheblichen Gebrauch von DAX machen. Die wichtigsten Punkte sind:

- Mit berechneten Spalten können Sie berechnete Beziehungen erstellen. Das ermöglicht es Ihnen, die Beziehung auf eine beliebige Berechnung zu stützen, nicht nur auf Gleichheitsverknüpfungen wie bei den Beziehungen, die die Engine normalerweise anbietet.
- Wenn Sie eine benötigte Beziehung nicht einrichten können, da sie dynamisch ist und von den im Bericht verwendeten Filtern und Datenschnitten abhängt, können Sie virtuelle Beziehungen nutzen. Für den Benutzer sehen sie wie Standardbeziehungen aus, allerdings werden sie im laufenden Betrieb berechnet. Die Leistung kann dabei leiden, aber das wird durch die Flexibilität wettgemacht, die Sie dadurch gewinnen.
- Berechnete Spalten bilden eine hervorragende Ergänzung zu den Modellierungsmöglichkeiten von Tabular. Mit nur wenigen berechneten Spalten können Sie äußerst anspruchsvolle Segmentierungen erreichen. Berechnet werden diese Spalten außerdem bei der Aktualisierung des Modells. Dadurch gewinnen Sie sowohl Geschwindigkeit und Flexibilität, was es Ihnen erlaubt, äußerst leistungsfähige Modelle aufzubauen.

Wir hoffen, Ihnen mit diesen wenigen Beispielen neue Einsichten darin gegeben zu haben, wie Sie mit ein wenig Kreativität hervorragende Modelle gestalten können.