Grundlagen relationaler Datenbanken

1

In diesem Kapitel

- Informationen organisieren
- Den Begriff Datenbank definieren
- Den Begriff DBMS definieren
- Datenbankmodelle im Vergleich
- Relationale Datenbanken
- Der Datenbankentwurf – eine Herausforderung

SQL (*Structured Query Language*) ist eine standardisierte Sprache, (die *ess-kju-el* und nicht *si-quel* ausgesprochen wird und) die dazu dient, Datenbanken zu erstellen sowie Daten in Datenbanken zu speichern, zu verwalten und daraus auszulesen. Es gibt eine Reihe verschiedener Datenbankarten, die unterschiedliche konzeptionelle Modelle repräsentieren. SQL wurde ursprünglich zu dem Zweck entwickelt, Daten in Datenbanken zu verwalten, die nach dem *relationalen Modell* aufgebaut sind. Der internationale SQL-Standard wurde kürzlich um einen Teil des *Objektmodells* erweitert, was zu hybriden Strukturen führt, die als objektrelationale Datenbanken bezeichnet werden. In diesem Kapitel beschreibe ich verschiedene Formen der Datenspeicherung, vergleiche das relationale Modell mit anderen wichtigen Modellen und gehe dann auf die wichtigen Merkmale relationaler Datenbanken ein.

Bevor ich Sie jedoch mit SQL bekannt mache, möchte ich sicherstellen, dass Sie verstehen, was ich mit dem Begriff *Datenbank* meine. Die Bedeutung dieses Begriffs bekam eine völlig neue Bedeutung, als Computer die Art und Weise veränderten, in der wir Informationen speichern und verwalten.

Die Übersicht behalten

Heute werden Computer für viele Aufgaben benutzt, die früher mit anderen Werkzeugen erledigt wurden. Beispielsweise haben Computer Schreibmaschinen weitgehend ersetzt, um Dokumente zu erstellen und zu ändern. Sie haben elektromechanische Rechenmaschinen verdrängt. Sie haben Millionen von Papierseiten, Ordnern und Ablageschränken als Hauptmedium für die Speicherung wichtiger Informationen abgelöst. Verglichen mit diesen alten Werkzeugen können Computer natürlich sehr viel mehr Aufgaben viel schneller und genauer erledigen. Diese Vorteile haben jedoch einen Nachteil: Computerbenutzer können nicht mehr direkt physisch auf ihre Daten zugreifen.

Wenn Computer gelegentlich ausfallen, fragen sich die Benutzer manchmal, ob die Computerisierung tatsächlich einen Fortschritt gebracht hat. Früher konnte ein Ordner nicht

»abstürzen«, sondern höchstens zu Boden fallen. Dann knieten Sie sich einfach hin, sammelten die Blätter auf und hefteten diese wieder im Ordner ab. Außer bei Erdbeben können Ablageschränke nicht »abstürzen«. Sie melden Ihnen auch niemals einen Fehler. Ein Festplattenabsturz ist dagegen etwas ganz anderes: Sie können die verloren gegangenen Bits und Bytes nicht einfach wieder »aufheben«. Mechanische, elektrische und menschliche Fehlfunktionen können dazu führen, dass Ihre Daten im »Irgendwo« verschwinden und für immer verloren sind.

Wenn Sie die notwendigen Vorsichtsmaßnahmen treffen, können Sie sich vor einem zufälligen Verlust Ihrer Daten schützen. Wenn Sie sich auf diese Art abgesichert haben, sind Sie in der Lage, von dem großen Nutzen zu profitieren, den der Computer in Form höherer Geschwindigkeit und Genauigkeit bietet.

Wenn Sie wichtige Daten speichern, müssen Sie Ihr Augenmerk auf die folgenden vier Bereiche lenken:

- ✔ Die Daten müssen schnell und einfach gespeichert werden, weil dieser Vorgang sehr häufig notwendig ist.
- ✔ Das Speichermedium muss zuverlässig sein. Sie wollen später keine Überraschung erleben und feststellen, dass Ihre Daten ganz oder teilweise verschwunden sind.
- ✔ Die Daten müssen schnell und einfach abgerufen werden können, und zwar unabhängig von der Anzahl der gespeicherten Einträge.
- ✔ Sie benötigen eine einfache Methode, um genau die gewünschten Daten aus der Riesenmenge der insgesamt gespeicherten Daten herauszufiltern.

Computerdatenbanken, die dem Stand der Technik entsprechen, erfüllen diese vier Kriterien. Wenn Sie mehr als ein Dutzend Datenelemente speichern müssen, sollten Sie dafür eine Datenbank verwenden.

Was ist eine Datenbank?

In den letzten Jahren ist der Begriff *Datenbank* ziemlich frei benutzt worden und hat seine ursprüngliche Bedeutung weitgehend eingebüßt. Für einige Menschen ist eine Datenbank jede Sammlung von Datenelementen (Telefonbücher, Einkaufszettel, Schriftrollen usw.). Andere definieren den Begriff genauer.

In diesem Buch definiere ich eine *Datenbank* als eine selbstbeschreibende Sammlung integrierter Datensätze. Und diese Definition impliziert Computertechnologie zusammen mit Sprachen wie SQL.

Ein *Datensatz* ist eine Darstellung eines physischen oder konzeptionellen Objekts. Wenn Sie beispielsweise die Kunden einer Firma verwalten wollen, legen Sie für jeden Kunden einen Datensatz an. Jeder Datensatz besitzt ein oder mehrere *Attribute*, wie beispielsweise den Namen, die Adresse oder die Telefonnummer. Einzelne Namen, Adressen usw. bilden die eigentlichen Daten.

Eine Datenbank besteht sowohl aus Daten als auch aus Metadaten. *Metadaten* sind Daten, die die Struktur der Daten innerhalb einer Datenbank beschreiben. Wenn Sie wissen, wie Ihre Daten strukturiert sind, können Sie sie abrufen. Eine Datenbank ist *selbstbeschreibend*, weil sie eine Beschreibung ihrer eigenen Struktur enthält. Die Datenbank ist *integriert*, weil sie neben den Datenelementen auch Beziehungen zwischen den Datenelementen enthält.

Die Datenbanken speichern Metadaten in einem Bereich, der *Datenverzeichnis* (engl. *Data Dictionary*) genannt wird und der die Tabellen, Spalten, Indizes, Einschränkungen (Bedingungen) und andere Elemente beschreibt, aus denen die Datenbank besteht.

Weil ein flaches Dateisystem (das später in diesem Kapitel beschrieben wird) keine Metadaten enthält, müssen Anwendungen, die mit flachen Dateien arbeiten, das Äquivalent zu den Metadaten als Teil des Anwendungsprogramms enthalten.

Datenbankgröße und -komplexität

Datenbanken gibt es in allen möglichen Größen, angefangen bei einer einfachen Sammlung weniger Datensätze bis hin zu Millionen von Datensätzen.

Eine *persönliche Datenbank* ist für die Benutzung durch eine einzige Person auf einem einzigen Computer bestimmt. Eine solche Datenbank ist normalerweise ziemlich einfach strukturiert und nicht sehr groß. Eine *Abteilungs-* oder *Arbeitsgruppendatenbank* ist für die Benutzung durch die Mitglieder einer einzelnen Abteilung oder Arbeitsgruppe innerhalb eines Unternehmens vorgesehen. Diese Art von Datenbank ist im Allgemeinen größer als eine persönliche Datenbank und demgemäß auch komplexer, weil sie mehrere Benutzer verwalten muss, die gleichzeitig auf dieselben Daten zugreifen können. Eine *Unternehmensdatenbank* kann riesig sein. Unternehmensdatenbanken können den gesamten Informationsfluss großer Unternehmen bilden.

Was ist ein Datenbankverwaltungssystem?

Gut, dass Sie das fragen. Ein Datenbankverwaltungssystem (*DBMS*, im Englischen heißt das *Database Management System*) ist eine Zusammenstellung von Programmen, mit denen Sie Datenbanken und die dazugehörigen Anwendungen definieren, verwalten und ausführen können. Eine *verwaltete Datenbank* ist im Wesentlichen eine Struktur, um Daten zu speichern, die für Sie oder Ihr Unternehmen wichtig sind. Ein DBMS ist ein Werkzeug, mit dem Sie eine solche Struktur errichten können, um mit den darin enthaltenen Daten zu arbeiten.

Heute gibt es viele Datenbankverwaltungssysteme auf dem Markt. Einige laufen nur auf Mainframes und einige nur auf PCs. Die Entwicklung geht jedoch in die Richtung von Produkten, die auf mehreren Plattformen und in verschiedenen Netzwerken mit allen drei Arten von Rechnern ausgewertet werden können.

Ein DBMS, das auf mehreren verschiedenen großen und kleinen Plattformen läuft, wird als *skalierbar* bezeichnet.

Unabhängig von der Größe des Computers, auf dem die Datenbank läuft, und unabhängig davon, ob der Rechner in ein Netzwerk eingebunden ist, bleibt der Informationsfluss zwischen der Datenbank und dem Benutzer gleich. Abbildung 1.1 zeigt, wie der Benutzer über das DBMS mit der Datenbank kommuniziert. Das Datenbankverwaltungssystem verbirgt die physischen Details der Datenbankspeicherung, weshalb die Anwendung nur die logischen Eigenschaften der Daten, nicht aber deren physische Speicherform kennen muss.

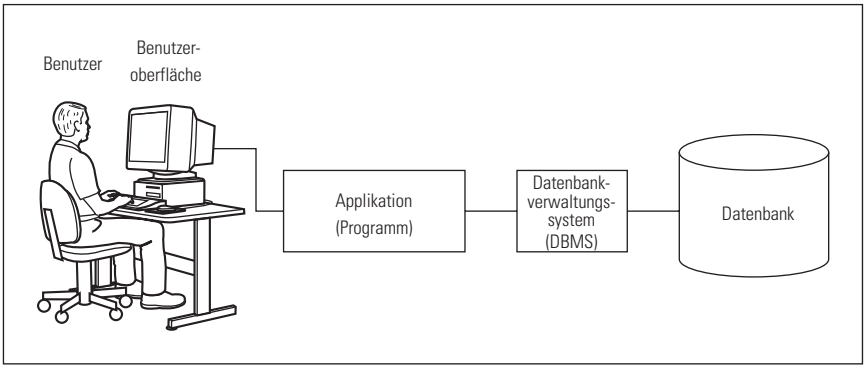

Abbildung 1.1: Blockdiagramm eines DBMS-Informationssystems

Der Wert liegt nicht in den Daten, sondern in der Struktur

Vor vielen Jahren hat eine schlaue Person berechnet, dass der Materialwert eines Menschen nur knapp einen Euro beträgt, wenn man ihn auf den Kohlenstoff, Wasserstoff, Sauerstoff und Stickstoff sowie die Spurenelemente reduziert, aus denen er zusammengesetzt ist. Diese Rechnung ist jedoch ebenso drollig wie irreführend. Menschen bestehen nicht nur aus einer Ansammlung von Atomen. Unsere Atome sind zu Enzymen, Eiweißen, Hormonen und vielen anderen Substanzen zusammengesetzt, die auf dem pharmazeutischen Markt teilweise Millionen von Euro pro Kilo kosten würden. Der Wert wird durch die spezifischen Strukturen geschaffen, zu denen die Atome zusammengesetzt sind. Analog dazu ermöglicht es die Datenbankstruktur, scheinbar bedeutungslose Daten zu interpretieren. Die Struktur bringt Muster, Trends und Tendenzen in den Daten zum Vorschein. Unstrukturierte Daten haben dagegen – wie die unverbundenen Atome – nur einen geringen Wert.

Flache Dateien

Wenn es um strukturierte Daten geht, ist die flache Datei das einfachste Element. Flache Dateien haben eine minimale Struktur. Wenn sie Gebäude wären, würden sie kaum über den Boden ragen. Eine flache Datei ist einfach eine Sammlung von Datensätzen, die nacheinander in einem bestimmten Format in einer Liste angeordnet sind – sie enthält die Daten, die ganzen Daten und nichts als die Daten. Bezogen auf Computer sind flache Dateien sehr einfach aufgebaut. Weil die Datei keine Informationen über ihre eigene Struktur enthält (Metadaten), ist der Overhead (Informationen in der Datei, die keine Daten sind) minimal.

Stellen Sie sich vor, dass Sie die Namen und Adressen der Kunden Ihrer Firma in einem flachen Dateisystem verwalten wollen. Das System soll ähnlich wie das folgende Beispiel strukturiert sein:

```
Klaus Klawuttke   Heimweg 15     12345Berlin
Jerry Cotton      Gasse 77       48787Hermannsburg
Adrian Hansen     Perlenallee 3  58789Oberstadt
Johann Bäcker     Terrenshof 128c98755Germsdorf
Michael Pens      Kalauerweg 1   57730Köln
Barbara Michimoto Sandufer 9     25252Kiel
Linda Schmidt     Vennweg 56     44134Düsseldorf
Robert Funnell    Kölner Ring 9  24240Lübeck
Boris Reckal      Am Hof 9       59554Siegen
Jeronimo KirenbergPferdeweg 7    35305Kassel
```

Wie Sie sehen, enthält die Datei nur Daten. Jedes Feld hat eine feste Länge (das Namensfeld ist beispielsweise immer genau 18 Zeichen lang) und ein Feld wird vom anderen nicht durch Strukturen getrennt. Die Person, die die Datenbank entworfen hat, hat die Feldpositionen und Längen festgelegt. Jedes Programm, das diese Datei benutzt, muss diese Felddefinitionen »kennen«, da diese Informationen nicht in der Datenbank selbst gespeichert sind.

Dieser geringe Overhead bewirkt, dass die Arbeit mit flachen Dateien sehr schnell gehen kann. Er hat jedoch den Nachteil, dass Anwendungsprogramme zusätzliche Logik enthalten müssen, um die Daten einer flachen Datei auf einer sehr niedrigen Ebene bearbeiten zu können. Die Anwendung muss genau wissen, wo und wie die Daten in der Datei gespeichert sind. Flache Dateien eignen sich für kleinere Systeme. Je größer jedoch ein System ist, desto umständlicher wird ein flaches Dateisystem.

Wenn Sie stattdessen eine Datenbank benutzen, können Sie den Mehraufwand verringern. Obwohl Datenbankdateien möglicherweise einen größeren Overhead aufweisen, können die Anwendungen leichter auf andere Hardware- und Betriebssystemplattformen übertragen werden. Außerdem erleichtert eine Datenbank das Schreiben von Anwendungsprogrammen, weil der Programmierer die Details über den Speicherort und die Speicherform der Daten nicht kennen muss.

Datenbanken sorgen also dafür, dass Ihre Anstrengungen nicht ins Uferlose gehen, weil das DBMS die komplette Datenbearbeitung übernimmt. Bei Anwendungen, die mit flachen Dateien arbeiten, müssen alle Einzelheiten im Anwendungscode programmiert werden. Wenn mehrere Anwendungen auf dieselben flachen Dateien zugreifen, muss jede Anwendung (redundanter-

weise) den gesamten Code zur Datenbearbeitung enthalten. Beim Einsatz eines DBMS braucht dieser Code überhaupt nicht in der Anwendung vorzukommen.

Offensichtlich ist es nicht einfach, eine Anwendung von einer Plattform auf eine andere zu portieren, wenn sie mit flachen Dateien arbeitet und plattformspezifischen Code zur Datenmanipulation enthält, weil Sie zunächst gezwungen sind, den gesamten spezifischen Hardware-Code zu ändern. Es ist sehr viel einfacher, eine DBMS-Anwendung auf eine andere Plattform zu portieren.

Datenbankmodelle

Unabhängig von ihrer Größe sind Datenbanken im Allgemeinen immer nach einem von drei Datenbankmodellen aufgebaut:

- ✔ **Relational:** Datenbankmanagementsysteme, die heute neu installiert werden, richten sich fast nur noch nach dem relationalen Modell. Unternehmen, die viel Geld in hierarchische oder Netzwerkdatenbanken investiert haben, bauen diese gelegentlich weiter aus, aber Unternehmen, die keine Kompatibilität zu »Altsystemen« aufrechterhalten müssen, wählen fast immer das relationale Modell für ihre Datenbanken.
- ✔ **Hierarchisch:** Hierarchische Datenbanken haben eine einfache hierarchische Struktur, die einen schnellen Datenzugriff ermöglicht. Sie leiden unter Redundanzproblemen und ihre Struktur ist unflexibel, wodurch Änderungen an der Datenbank schwierig sind.
- ✔ **Netzwerk:** Netzwerkdatenbanken sind nur minimal redundant, haben aber den Nachteil, dass ihre Struktur sehr komplex ist.

Die ersten Datenbanken, die weite Verbreitung fanden, waren große Unternehmensdatenbanken, die entweder nach dem hierarchischen Modell oder dem Netzwerkmodell strukturiert waren. Systeme, die nach dem relationalen Modell aufgebaut sind, wurden erst mehrere Jahre später eingeführt. SQL ist eine relativ moderne Sprache, die nur auf das relationale Modell und auf seinen Abkömmling, das objektrelationale Modell, anwendbar ist. Ich werde mich im Folgenden nicht mehr mit dem hierarchischen Modell und dem Netzwerkmodell beschäftigen.

Neuere Datenbankverwaltungssysteme, die nicht auf dem relationalen Modell basieren, stützen sich wahrscheinlich auf das neuere Objektmodell oder das objektrelationale Hybridmodell.

Das relationale Modell

E. F. Codd, der bei IBM arbeitete, formulierte das relationale Datenbankmodell erstmals im Jahre 1970. Es dauerte etwa ein Jahrzehnt, bis dieses Modell für Datenbankprodukte verwendet wurde. Es entbehrt nicht einer gewissen Ironie, dass nicht die Firma IBM das erste relationale DBMS lieferte. Diese Ehre kam einer kleinen, neu gegründeten Firma zu, die ihr Produkt *Oracle* nannte.

Relationale Datenbanken haben Datenbanken der früheren Modelle weitgehend verdrängt, weil das relationale Modell wertvolle Eigenschaften hat, die den anderen Datenbankmodellen fehlen. Wahrscheinlich ist das Wichtigste dieser Attribute die Tatsache, dass Sie in einer relationalen Datenbank die Datenbankstruktur ändern können, ohne die Anwendungen ändern zu müssen, die mit der alten Struktur gearbeitet haben. Nehmen Sie beispielsweise an, dass Sie eine oder mehrere neue Spalten in eine Datenbanktabelle einfügen möchten. Sie müssen vorher geschriebene Anwendungen, die mit dieser Tabelle arbeiten, nicht ändern, es sei denn, Sie ändern eine oder mehrere der Spalten, die in den Anwendungen verwendet werden.

Wenn Sie natürlich eine Spalte löschen, auf die eine vorhandene Anwendung Bezug nimmt (sprich: die von einer vorhandenen Anwendung referenziert wird), haben Sie Probleme, und zwar unabhängig von dem verwendeten Datenbankmodell. Eine der wirksamsten Methoden, einen Fehler in einer Datenbankanwendung auszulösen, besteht darin, Daten aus Spalten abzurufen, die in der Datenbank nicht enthalten sind.

Warum das relationale Modell besser ist

In Anwendungen, die mit einem DBMS geschrieben worden sind, das dem hierarchischen Modell oder dem Netzwerkmodell folgt, ist die Datenbankstruktur fest in der Anwendung kodiert. Dies bedeutet, dass die Anwendung von der spezifischen Implementierung der Datenbank abhängig ist. Wenn Sie ein neues Attribut in die Datenbank einfügen, müssen Sie Ihre Anwendung entsprechend anpassen, und zwar unabhängig davon, ob die Anwendung das neue Attribut benutzt oder nicht.

Relationale Datenbanken bieten Flexibilität in den Strukturen. Anwendungen, die für diese Datenbanken geschrieben werden, sind einfacher zu warten als Anwendungen für hierarchische oder Netzwerkdatenbanken. Diese strukturelle Flexibilität ermöglicht es Ihnen, Datenkombinationen abzurufen, von denen Sie zum Zeitpunkt des Datenbankentwurfs nicht gedacht hätten, dass Sie sie benötigen könnten.

Komponenten einer relationalen Datenbank

Die Flexibilität relationaler Datenbanken beruht darauf, dass ihre Daten in Tabellen gespeichert werden, die im Wesentlichen unabhängig voneinander sind. Sie können in einer Tabelle Daten einfügen, ändern oder löschen, ohne dadurch die Daten in den anderen Tabellen zu beeinflussen, vorausgesetzt, die betroffene Tabelle ist den anderen Tabellen nicht *übergeordnet*. (Über- und untergeordnete Beziehungen zwischen Tabellen werden in Kapitel 5 erklärt.) In diesem Abschnitt zeige ich, woraus diese Tabellen bestehen und in welcher Beziehung sie zu den Komponenten einer relationalen Datenbank stehen.

Wer kommt zum Abendessen?

In der Ferienzeit besuchen mich immer viele Menschen, zu denen ich engere Beziehungen pflege. Wir sitzen dann oft zusammen an meinem Tisch. Auch Datenbanken pflegen Beziehungen, doch jede ihrer Beziehungen besitzt einen eigenen Tisch. Eine relationale Datenbank besteht aus einer oder mehreren Beziehungen, auch Relationen genannt.

Eine *Relation* ist eine zweidimensionale, aus Zeilen und Spalten bestehende Anordnung von Feldern, auch Array genannt, die nur Einträge eines Typs von Datenwerten und keine identischen Zeilen enthalten kann. Jede Zelle des Arrays darf nur einen Wert enthalten. Keine zwei Zeilen dürfen identisch sein.

Die meisten Benutzer sind mit zweidimensionalen, aus Zeilen und Spalten bestehenden Arrays in Form von elektronischen Tabellenkalkulationen (wie zum Beispiel *Microsoft Excel*) vertraut. Die Offensivstatistik, die auf der Rückseite der Baseball-Karte jedes Baseball-Spielers der Major-League aufgedruckt ist, ist ein weiteres Beispiel für ein Datenfeld. Die Baseball-Karte enthält Spalten für das Jahr, die Mannschaft (Team), die geleisteten Spiele sowie für die verschiedenen Leistungskriterien (At Bat, Hits, Runs usw.), die in der Baseball-Welt eine Rolle spielen. Eine Zeile fasst die Daten eines Jahres zusammen, in dem der Spieler in der Major-League gespielt hat. Sie können diese Daten auch in einer Relation (einer Tabelle) speichern, die dieselbe Grundstruktur hat. Abbildung 1.2 zeigt eine relationale Datenbanktabelle, die die Offensivstatistik eines Spielers der Major-League enthält. In der Praxis würde eine solche Tabelle die statistischen Daten einer ganzen Mannschaft oder möglicherweise der gesamten Liga enthalten.

Spieler	Jahr	Team	Spiel	At Bat	Hits	Runs	RBI	2B	3B	HR	Walk	Steals	Bat. Avg.
Roberts	1988	Padres	5	9	3	1	0	0	0	0	1	0	.333
Roberts	1989	Padres	117	329	99	81	25	15	8	3	49	21	.301
					172	104							

Abbildung 1.2: Tabelle mit der Offensivstatistik eines Baseball-Spielers

Spalten in der Tabelle sind in dem Sinne *konsistent*, als eine Spalte in jeder Zeile dieselbe Bedeutung hat. Wenn eine Spalte in einer Zeile den Nachnamen eines Spielers enthält, muss die Spalte in allen Zeilen den Nachnamen eines Spielers enthalten. Die Reihenfolge, in der Zeilen und Spalten in einer Tabelle erscheinen, spielt dabei keine Rolle. Aus der Sicht des DBMS ist es deshalb gleichgültig, welche Spalte an erster, welche an zweiter und welche an letzter Stelle steht. Dasselbe gilt für die Zeilen. Das DBMS verarbeitet die Tabelle unabhängig davon, wie sie aufgebaut ist.

Jede Spalte einer Datenbanktabelle verkörpert ein einzelnes Attribut der Tabelle. Die Bedeutung einer Spalte ist in allen Zeilen der Tabelle gleich. Eine Tabelle kann beispielsweise die Namen, Adressen und Telefonnummern aller Kunden einer Firma enthalten. Jede Zeile in der Tabelle (auch *Datensatz* oder *Tupel* genannt) enthält die Daten eines einzelnen Kunden. Jede Spalte enthält ein einzelnes Attribut des Kunden, wie zum Beispiel die Kundennummer (`KundenNr`), die Firma (`Unternehmen`), die Straße (`Anschrift1`) usw. Abbildung 1.3 zeigt einige Zeilen und Spalten einer solchen Tabelle.

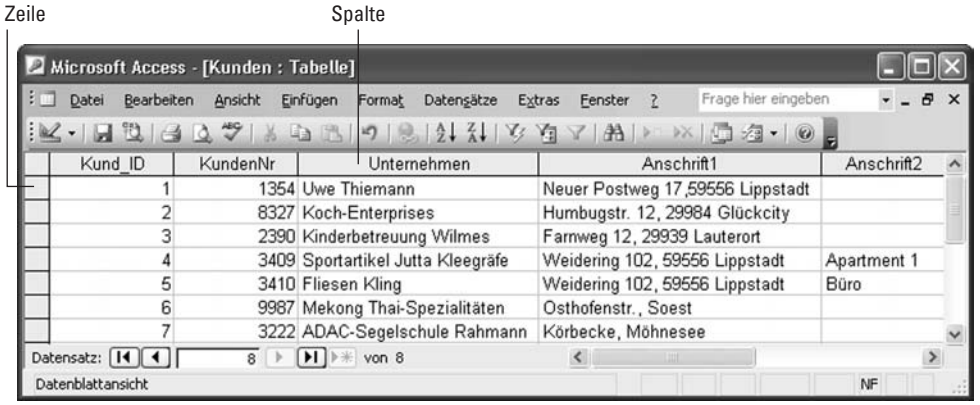

Abbildung 1.3: Jede Tabellenzeile enthält einen Datensatz; jede Spalte enthält ein einzelnes Attribut.

Die Beziehungen in einem Datenbankmodell entsprechen den Tabellen einer auf dem Modell basierenden Datenbank.

Sicht(weisen)

Tabellen können viele Spalten und Zeilen enthalten. Manchmal interessieren Sie sich für alle Daten und manchmal nicht. Möglicherweise möchten Sie nur einige der Spalten einer Tabelle sehen oder nur Zeilen, die eine bestimmte Bedingung erfüllen. Vielleicht möchten Sie einige Spalten einer Tabelle und einige andere Spalten einer mit ihr verknüpften Tabelle sehen. Um die Daten auszuschließen, die Sie in diesem Augenblick nicht interessieren, erstellen Sie eine so genannte *Sicht* (engl. *View*). Eine Sicht ist eine Untermenge einer Datenbank. Diese Untermenge kann von einer Anwendung verarbeitet werden. Eine Sicht kann Ausschnitte einer oder mehrerer Tabellen enthalten.

Sichten werden manchmal als *virtuelle Tabellen* bezeichnet. Für eine Anwendung oder die Benutzer verhalten sich Sichten wie Tabellen. Sichten existieren jedoch nicht unabhängig von Tabellen. Views sind eine bestimmte Methode, Daten zu betrachten, aber sie sind nicht die Daten selbst.

Angenommen, Sie arbeiten mit einer Datenbank, die die Tabellen `Kunde` und `Rechnung` enthält. Die Tabelle `Kunde` enthält die Spalten `KundenNr`, `Vorname`, `Nachname`, `Strasse`, `Ort`, `Land`, `Plz` und `Tel`. Die Tabelle `Rechnung` verfügt über die Spalten `RechnungNr`, `KundenNr`, `Datum`, `Umsatz`, `Bezahlt` und `Zahlungsart`.

Ein Verkaufsleiter möchte nur den Vornamen (`Vorname`), den Nachnamen (`Nachname`) und die Telefonnummer (`Tel`) eines Kunden auf dem Bildschirm sehen. Der Verkaufsleiter kann nun aus der Tabelle `Kunde` eine Sicht erstellen, die nur die drei gewünschten Spalten und nicht auch noch die im Moment überflüssigen Informationen aus anderen Spalten enthält. Abbildung 1.4 zeigt die vom Verkaufsleiter erstellte Sicht.

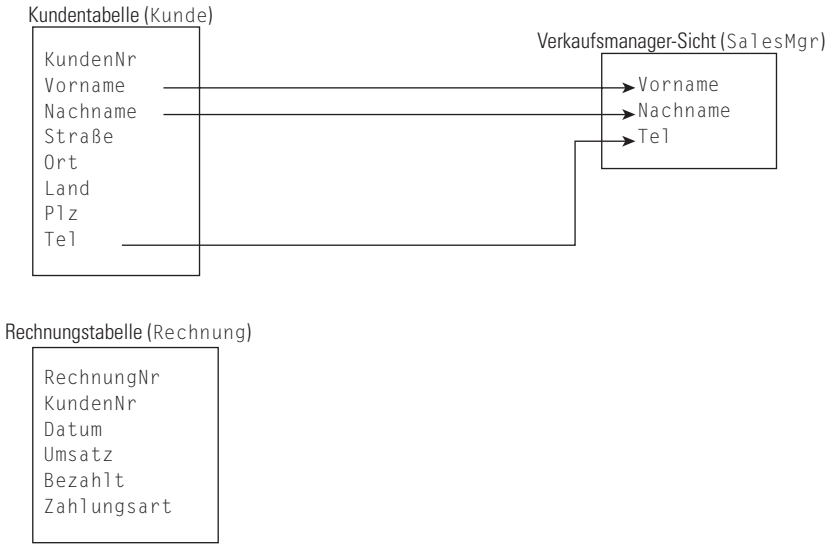

Abbildung 1.4: Die Sicht des Verkaufsmanagers wird von der Tabelle KUNDE abgeleitet.

Ein Zweigstellenleiter möchte die Namen und Telefonnummern aller Kunden sehen, deren Postleitzahl zwischen 40000 und 59999 (das entspricht ungefähr Nordrhein-Westfalen) liegt. Eine Sicht, die die von ihr abgerufenen Zeilen und angezeigten Spalten mit einer so genannten Einschränkung, einem Filter, versieht, führt diese Aufgabe aus. Abbildung 1.5 zeigt die Datenquellen für die Spalten der Sicht des Zweigstellenleiters.

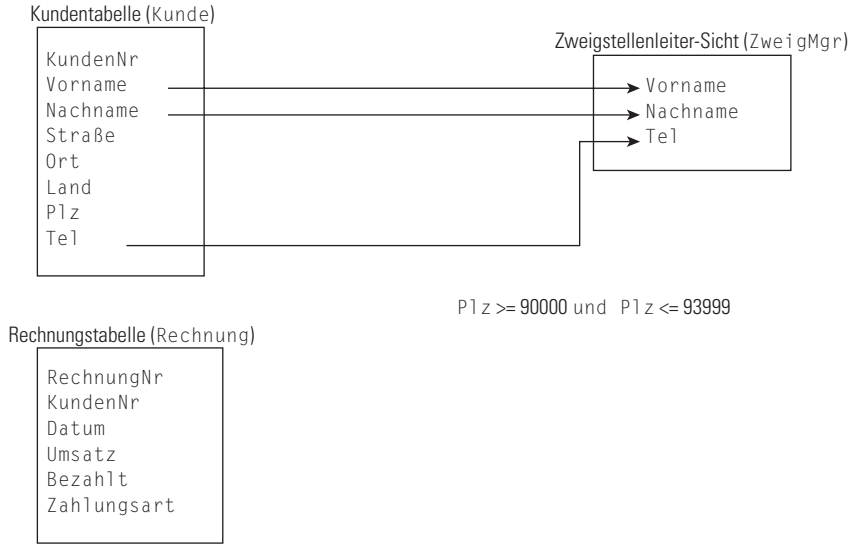

Abbildung 1.5: Die Sicht des Zweigstellenleiters enthält nur bestimmte Zeilen der Tabelle KUNDE.

Der Buchhaltungsleiter möchte die Kundennamen der Tabelle `Kunde` und die Spalten `Datum`, `Umsatz`, `Bezahlt` und `Zahlungsart` der Tabelle `Rechnung` sehen, bei denen `Bezahlt` kleiner als `Umsatz` ist. Die zuletzt genannte Bedingung ist erfüllt, wenn eine Rechnung noch nicht vollständig beglichen wurde. Für diese Sicht müssen Sie Daten aus beiden Tabellen kombinieren. Abbildung 1.6 zeigt, wie Daten der Tabellen `Kunde` und `Rechnung` in die Sicht des Buchhaltungsleiters einfließen.

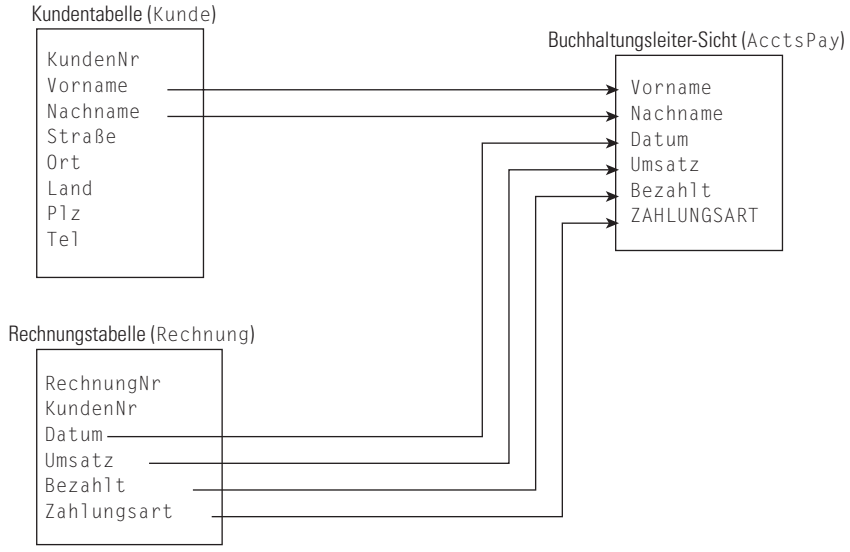

Abbildung 1.6: Die Sicht des Buchhaltungsleiters kombiniert Daten aus zwei Tabellen.

Sichten sind nützlich, weil sie es ermöglichen, Daten aus einer Datenbank zu extrahieren und zu formatieren, ohne die gespeicherten Daten wirklich zu ändern. In Kapitel 6 zeige ich, wie Sie mit SQL eine Sicht erstellen.

Schemata, Domänen und Einschränkungen

Eine Datenbank ist mehr als eine Sammlung von Tabellen. Zusätzliche Strukturen, die über mehrere Schichten verteilt sind, helfen, die Integrität der Daten zu bewahren. Das *Schema* einer Datenbank stellt eine übergreifende Organisation der Tabellen dar. Die *Domäne* einer Tabellenspalte sagt Ihnen, welche Werte Sie in der Spalte speichern dürfen. Sie können *Einschränkungen* (*engl. Constraints*) für eine Datenbanktabelle definieren, um zu verhindern, dass jemand (Sie eingeschlossen) ungültige Daten in der Tabelle speichert.

Schemata

Die Struktur einer ganzen Datenbank wird als ihr *Schema* oder als *konzeptionelle Sicht* und manchmal auch als *vollständige logische Sicht* der Datenbank bezeichnet. Das Schema gehört

zu den Metadaten und ist damit ein Teil der Datenbank. Die Metadaten selbst, die die Struktur der Datenbank beschreiben, werden wie normale Daten in Tabellen gespeichert. Damit sind sogar Metadaten nichts anderes als Daten, und das ist das Gute an ihnen.

Domänen

Ein *Attribut* einer Relation (das heißt die Spalte einer Tabelle) kann eine endliche Anzahl von Werten annehmen. Die Gesamtmenge dieser Werte wird als die *Domäne* des Attributs bezeichnet.

Angenommen, Sie sind ein Autohändler, der das neue Curarri GT 4000 Sportcoupé verkauft. Sie verwalten die Autos, die Sie am Lager haben, in einer Datenbanktabelle mit dem Namen `LAGER`. Eine Spalte dieser Tabelle hat den Namen `Farbe`, in der die Farbe jedes Autos gespeichert wird. Der GT 4000 wird nur in vier Farben geliefert: Lippenrot, Mitternachtsschwarz, Schneeweiß und Metallicgrau. Diese vier Farben bilden die Domäne des Attributs `Farbe`.

Einschränkungen

Einschränkungen sind eine wichtige, gleichwohl oft übersehene Komponente einer Datenbank. Einschränkungen sind Regeln, die festlegen, welche Werte die Attribute einer Tabelle annehmen dürfen.

Wenn Sie Einschränkungen für eine Spalte definieren, können Sie verhindern, dass die Benutzer in diese Spalte ungültige Daten eingeben. Natürlich muss jeder Wert, der in der Domäne der Spalte gültig wäre, auch alle ihre Einschränkungen berücksichtigen. Wie ich im vorangegangenen Abschnitt erläutert habe, bildet die Menge aller Werte, die die Spalte enthalten darf, ihre Domäne. Eine Einschränkung schränkt diese Werte ein. Die Eigenschaften einer Tabellenspalte definieren zusammen mit den Einschränkungen, die auf diese Spalte angewendet werden, die Spaltendomäne. Durch Einschränkungen können Sie verhindern, dass Daten in eine Spalte eingegeben werden, die außerhalb ihrer Domäne liegen.

Bei dem Beispiel des Autohändlers könnten Sie eine Einschränkung so definieren, dass die Datenbank in der Spalte `Farbe` nur die vier zulässigen Werte akzeptiert. Falls ein Benutzer dann versuchen sollte, eine andere Farbe, wie zum Beispiel `Waldgrün`, in die entsprechende Spalte einzugeben, weist das System die Eingabe zurück. Die Dateneingabe kann erst dann fortgesetzt werden, wenn der Benutzer eine gültige Farbe in das Feld `Farbe` eingibt.

Das Objektmodell fordert das relationale Modell heraus

Das relationale Modell hat in vielen Anwendungsbereichen einen fantastischen Erfolg gehabt. Es ist jedoch nicht frei von Problemen. Diese Probleme wurden mit zunehmender Beliebtheit objektorientierter Programmiersprachen, zum Beispiel C++, Java und C#, immer deutlicher. Solche Sprachen können komplexere Probleme leichter handhaben als traditionelle Sprachen, weil sie über Funktionen verfügen, wie beispielsweise durch Benutzer erweiterbare Datentypen,

Kapselung, Vererbung, dynamische Bindung von Methoden, komplexe und zusammengesetzte Objekte und Objektidentitäten.

Ich werde nicht alle diese Begriffe in diesem Buch erklären (obwohl Sie einigen später begegnen werden). Es reicht aus zu wissen, dass das klassische relationale Modell nicht mit allen diesen Funktionalitäten klarkommt. Deshalb werden Datenbankverwaltungssysteme entwickelt und angeboten, die auf dem Objektmodell basieren. Allerdings haben sie zurzeit nur einen relativ kleinen Marktanteil.

Das objektrelationale Modell

Datenbankentwickler streben wie fast jedermann danach, für ihre Aufgaben die beste aller Lösungen zu finden. Deshalb versuchten einige, die Vorteile eines objektorientierten Datenbanksystems mit denen des erfolgreichen relationalen Systems zu verbinden und dabei die Kompatibilität zu Letzterem zu bewahren. Das Ergebnis dieser Bemühungen ist das objektrelationale Hybridmodell. Objektrelationale Datenbankverwaltungssysteme erweitern das relationale Modell um die Fähigkeit, Daten objektorientiert zu modellieren. Der internationale SQL-Standard wurde um objektorientierte Funktionalitäten erweitert. Damit erhalten die Anbieter relationaler Datenbankverwaltungssysteme die Möglichkeit, ihre Produkte zu objektrelationalen Datenbankverwaltungssystemen auszubauen und dabei die Kompatibilität zum Standard zu wahren. Deshalb beschreibt SQL:1999 im Gegensatz zu dem SQL-92-Standard, der ein rein relationales Datenbankmodell beschreibt, ein objektrelationales Datenbankmodell. SQL:2003 weist sogar noch mehr objektorientierte Funktionen auf. Die Aktualisierung dieses Standards, SQL/XML:2005, konzentriert sich, wie es schon der Name vermuten lässt, mehr auf die Einbindung von XML als auf weitere objektorientierte Funktionalitäten.

Ich beschreibe in diesem Buch den internationalen ISO-/IEC-SQL-Standard. Dabei handelt es sich hauptsächlich um ein relationales Datenbankmodell. Außerdem beschreibe ich die objektorientierten Erweiterungen, die mit SQL:1999 eingeführt wurden, sowie die zusätzlichen Erweiterungen von SQL:2003. Die objektorientierten Funktionalitäten des neuen Standards geben Entwicklern die Möglichkeit, SQL-Datenbanken für die Lösung von Problemen einzusetzen, was mit dem früheren, rein relationalen Ansatz nicht gegeben war.

Überlegungen zum Datenbankentwurf

Eine Datenbank ist die Darstellung einer physischen oder konzeptionellen Struktur, wie eines Unternehmens, der Montage eines Autos oder der Erfolgsstatistik aller Mannschaften der Bundesliga. Die Genauigkeit dieser Darstellung hängt davon ab, wie detailliert Ihr Datenbankentwurf ist. Der Aufwand, den Sie in den Datenbankentwurf stecken, sollte von der Art der Informationen abhängen, die Ihnen die Datenbank liefern soll. Zu viele Details bedeuten eine Verschwendung von Arbeit, Zeit und Speicherplatz. Zu wenige Details können die Datenbank wertlos werden lassen.

Stellen Sie fest, wie viele Details Sie jetzt benötigen und wie viele möglicherweise in der Zukunft, und entwerfen Sie die Datenbank exakt für den ermittelten Bedarf – nicht für mehr und nicht für weniger. Seien Sie aber nicht überrascht, wenn Sie den Entwurf später ändern müssen, um ihn an die Anforderungen der Praxis anzupassen.

Die heutigen Datenbankverwaltungssysteme mit ihren attraktiven grafischen Benutzeroberflächen und intuitiven Entwicklungswerkzeugen können einen Möchtegern-Datenbankentwickler in falscher Sicherheit wiegen. Diese Systeme suggerieren, dass der Entwurf einer Datenbank mit der Konstruktion eines Arbeitsblatts in einer Tabellenkalkulation oder einer ähnlich unkomplizierten Aufgabe vergleichbar ist. Dem ist nicht so. Datenbanken zu entwerfen ist schwierig. Wenn Sie es nicht richtig machen, erzeugen Sie eine Datenbank, die nach und nach immer unbrauchbarer wird. Oft tritt das Problem erst zutage, wenn Sie bereits sehr viel Aufwand in die Datenerfassung gesteckt haben. Wenn Sie das Problem bemerken, ist es oft bereits zu spät. In vielen Fällen besteht die einzige Lösung darin, die Datenbank komplett neu zu entwerfen und alle Daten noch einmal einzugeben. Das hat allerdings den Vorteil, dass Sie dann etwas dazugelernt haben.