IN DIESEM KAPITEL

Kapitel 1
Grundlagen relationaler Datenbanken

SQL (*Structured Query Language*) ist eine Abfragesprache, die *ess-ku-el* und nicht *si-quel* ausgesprochen wird und die speziell für das Arbeiten mit Datenbanken entwickelt wurde. Mit ihr kann man Datenbanken erstellen, neue Daten in Datenbanken einfügen und ausgewählte Teilmengen der Daten abrufen. SQL wurde 1970 eingeführt und im Laufe der Jahre weiterentwickelt. Heute ist sie längst zu einem wichtigen Branchenstandard geworden. SQL wird durch einen formellen Standard definiert, der von der International Standards Organization (ISO) betreut wird.

Es gibt verschiedene Datenbankarten, die unterschiedliche konzeptionelle Modelle für die Organisation von Daten widerspiegeln.

SQL wurde ursprünglich zu dem Zweck entwickelt, Daten zu verwalten, die in Datenbanken enthalten sind, deren Aufbau dem *relationalen Modell* entspricht. Der internationale SQL-Standard wurde vor einigen Jahren 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 mit anderen wichtigen Datenbankmodellen und gehe dann auf die wichtigen Merkmale relationaler Datenbanken ein.

Doch bevor ich näher auf SQL eingehe, will ich den Begriff der *Datenbank* klar definieren. Seine Bedeutung änderte sich mit der Art und Weise, in der Computer Informationen speicherten und verwalteten.

Die Übersicht über die Dinge bewahren

Heute werden Computer für viele Aufgaben benutzt, für die früher andere Werkzeuge benutzt wurden. Mechanische und elektrische Schreibmaschinen wurden von Computern beispielsweise längst weitestgehend verdrängt, wenn es darum geht, Dokumente zu erstellen und zu bearbeiten. Sie haben elektronische und mechanische Rechenmaschinen verdrängt. Millionen auf Papier ausgedruckte Seiten, Ordner und Archivschränke wurden als Hauptmedium zur Aufbewahrung wichtiger Daten abgelöst. Computer sind normalerweise sehr viel schneller und effizienter als diese alten Werkzeuge. Diese Vorteile haben aber auch einen Nachteil, denn Computerbenutzer haben keinen direkten physischen Zugang zu ihren Daten mehr.

Bei Computerausfällen fragen sich Benutzer zuweilen, ob die Computerisierung wirklich Fortschritte gebracht hat. Früher konnten Ordner nicht »abstürzen«, sondern höchstens mit lautem Krach auf dem Boden landen. Dann wurden die Blätter einfach wieder aufgesammelt und wieder in den Aktenordner geheftet. Außer bei Erdbeben können Aktenschränke kaum »abstürzen«. Fehlermeldungen zeigen sie Ihnen auch nie an. Festplattenabstürze sind hingegen etwas ganz anderes: Verloren gegangene Bits und Bytes lassen sich nicht einfach vom Boden »aufheben«. Mechanische, elektronische und menschliche Fehlfunktionen können dazu führen, dass Ihre Daten ins »Nirwana« verschwinden und für immer verloren sind.

Wenn Sie sich aber mit den gebotenen Vorsichtsmaßnahmen gegen zufällige Datenverluste schützen, können Sie die Vorteile nutzen, die Ihnen Computer mit ihrer höheren Geschwindigkeit und Präzision bieten.

Im modernen Arbeitsumfeld müssen Sie gerade bei wichtigen Datenbanken darauf aufpassen, dass sie möglichst gut geschützt werden. Externe Zugriffe von außen sollten nur möglich sein, wenn es wirklich erforderlich ist.

Wenn Sie wichtige Daten speichern, müssen Sie Ihr Augenmerk folgenden vier Bereichen widmen:

- ✔ Die Daten müssen schnell und einfach gespeichert werden, weil dieser Vorgang sehr häufig notwendig ist.
- ✔ Die Speichermedien müssen zuverlässig arbeiten. Sie wollen schließlich später keine böse 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 Menge der gespeicherten Daten.
- ✔ Sie benötigen einfache Möglichkeiten, um genau die gewünschten Daten aus der Masse der gespeicherten Daten herauszufiltern.

Moderne Computer-Datenbanken, die sich auf dem Stand moderner Technik befinden, erfüllen diese vier Kriterien. Wenn Sie mehr als einige Dutzend Datenelemente speichern müssen, sollten Sie dafür Datenbanken verwenden.

Was ist eine Datenbank?

Mit der Entwicklung der Computer hat der Begriff *Datenbank* seine ursprüngliche Bedeutung recht weitgehend verloren. Einige betrachten jede Sammlung von Datenelementen (Telefonbücher, Einkaufszettel, Schriftrollen und so weiter) als Datenbank. Andere definieren den Begriff präziser.

In diesem Buch wird eine *Datenbank* als selbstbeschreibende Sammlung integrierter Datensätze beschrieben. Und diese Definition setzt neben Computertechnologien Programmiersprachen wie SQL voraus.

Bei einem *Datensatz* handelt es sich um die Repräsentation eines physischen oder konzeptionellen Objekts. Wenn Sie beispielsweise Kunden einer Firma verwalten wollen, legen Sie für jeden Kunden einen Datensatz an. Jeder Datensatz enthält ein oder mehrere *Attribute*, wie beispielsweise den Namen, die Adresse oder die Telefonnummer. Einzelne Namen, Adressen und so weiter bilden die eigentlichen Daten oder *Datenelemente*.

Eine Datenbank besteht aus Daten und Metadaten. *Metadaten* beschreiben dabei die Struktur der Daten innerhalb der Datenbank. 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* (englisch *Data Dictionary*) genannt wird und der die Tabellen, Spalten, Indizes, Einschränkungen (Bedingungen) und andere Elemente beschreibt, aus denen die Datenbank besteht.

Weil flache Dateisysteme (die später in diesem Kapitel beschrieben werden) keine Metadaten enthalten, müssen mit flachen Dateien arbeitende Anwendungen die den Metadaten entsprechenden Angaben im Programm enthalten. Anders ausgedrückt, muss dann das Programm selbst die Struktur der Daten kennen.

Datenbankgröße und -komplexität

Datenbanken gibt es mit allen möglichen Datenmengen, angefangen bei einfachen Sammlungen mit wenigen Datensätzen bis hin zu Millionen Datensätzen und mehr. Die meisten Datenbanken lassen sich in eine von drei Kategorien einordnen, die von der Größe der Datenbank selbst, der relativen Kapazität der Computer, auf denen sie laufen, und der Größe der Organisationen abhängen, von denen sie betrieben werden:

- ✔ Eine *persönliche Datenbank* ist für die Benutzung durch eine einzige Person auf einem einzigen Computer bestimmt. Derartige Datenbanken sind meist recht einfach strukturiert und relativ klein.
- ✔ Eine *Abteilungs-* oder *Arbeitsgruppendatenbank* ist für die Benutzung durch die Mitglieder einer einzelnen Abteilung oder Arbeitsgruppe innerhalb eines Unternehmens

vorgesehen. Diese Art Datenbank ist meist umfangreicher als eine persönliche Datenbank und demgemäß auch komplexer, weil sie mehrere Benutzer verwalten muss, die gemeinsam auf dieselben Daten zugreifen.

- ✔ Eine *Unternehmensdatenbank* kann riesig sein. Unternehmensdatenbanken können den gesamten geschäftskritischen Informationsfluss großer Unternehmen abbilden.

Was ist ein Datenbankverwaltungssystem?

Gut, dass Sie das fragen. Ein *Datenbankverwaltungssystem* (*DBMS – Database Management System*) ist ein Satz von Programmen, mit denen Sie Datenbanken und die dazugehörigen Anwendungen definieren, verwalten und ausführen können. Eine *verwaltete Datenbank* ist im Wesentlichen eine Struktur zum Speichern von Daten, die für Sie oder Ihr Unternehmen wichtig sind. Ein DBMS ist ein Werkzeug, mit dem Sie eine derartige Struktur erstellen können, um die darin enthaltenen Daten bearbeiten zu können.

Heute werden viele Datenbankverwaltungssysteme angeboten. Einige laufen nur auf Servern, andere nur lokal auf Arbeitsrechnern, wie PCs, Laptops oder Tablets. Die Entwicklung geht jedoch deutlich in Richtung von Produkten, die von mehreren Plattformen unterstützt werden und in unterschiedlichen Netzwerken mit Rechnern verschiedener Leistungsklassen arbeiten können. Sehr viele Anwendungen besitzen Bedienoberflächen für Internet-Browser. Noch ausgeprägter ist der Trend, Daten in der *Cloud* zu speichern. Dieser Speicher kann heute von privaten Unternehmen oder auch einzelnen Personen, aber auch von großen Unternehmen wie etwa Amazon, Google oder Microsoft bereitgestellt werden. Dabei kann man zwischen privater und öffentlicher Cloud unterscheiden. Bereitgestellt werden die Daten irgendwo »in den Wolken« über das Internet. Lediglich der Zugang zu den Daten wird in einer privaten Cloud eingeschränkt. Im Grunde handelt es sich dann um eine Erweiterung des firmeninternen Intranets bis hinein ins Internet.

Der Begriff *Cloud* ist heute zwar einigermaßen unscharf definiert, aber in aller Munde. Eigentlich besteht eine Cloud lediglich aus einer Ansammlung von Computer-Ressourcen, auf die irgendwie mit einem Browser oder einer Anwendung über das Internet oder im privaten Intranet zugegriffen werden kann. Der Nebel der »Wolken« verhüllt lediglich, wo sich die benutzten Computer-Ressourcen in der Cloud befinden, also ob sich diese in physischen Rechenzentren mit bekanntem Standort oder vielleicht bei Google oder Amazon irgendwo auf dem Planeten auf angemieteten Geräten befinden. Der Datenzugriff muss dabei nicht zwangsläufig mit einem Browser erfolgen, da sich die gleichen Funktionen selbstverständlich auch mit geeignet programmierten und vorkonfigurierten Apps realisieren lassen.

Ein DBMS, das auf mehreren verschiedenen großen und kleinen Plattformen läuft, nennt man *skalierbar*.

Unabhängig von der Leistung des Computers, auf dem die Datenbank läuft, und davon, ob der Rechner in ein Netzwerk eingebunden ist, bleibt der Informationsfluss zwischen der

Datenbank und dem Benutzer gleich. Abbildung 1.1 zeigt, wie Benutzer über das DBMS mit der Datenbank kommunizieren. Das Datenbankverwaltungssystem verbirgt die physischen Details der Datenbankspeicherung, weshalb die Anwendung nur die logischen Eigenschaften der Daten, nicht aber deren physische Speicherform kennen muss.

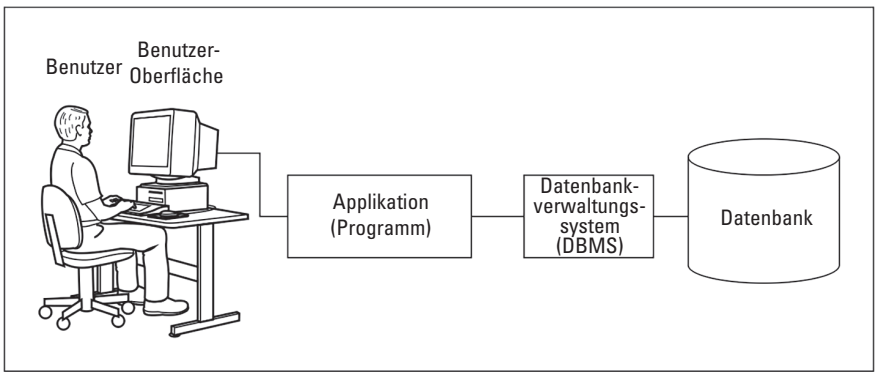

Abbildung 1.1: Blockdiagramm eines DBMS-Informationssystems

Nicht die Daten, sondern die Struktur stellen den Wert dar

Vor vielen Jahren hat eine kluge Person berechnet, dass der Materialwert eines Menschen nur knapp einen Euro beträgt, wenn man ihn auf Kohlenstoff, Wasserstoff, Sauerstoff und Stickstoff sowie die Spurenelemente reduziert, aus denen er besteht. Diese Rechnung ist jedoch nicht nur skurril, sondern auch irreführend. Menschen bestehen nicht nur aus einem Haufen Atome. Unsere Atome bilden Enzyme, Eiweiße, Hormone und viele andere Substanzen, für die in der Pharmaindustrie auf dem Markt teilweise Millionen Euro pro Kilo bezahlt werden würden. Für den Wert sorgen die spezifischen Strukturen, die von den Atomen gebildet werden. Ähnlich ermöglicht die Datenbankstruktur eine Interpretation scheinbar bedeutungsloser Daten. Die Struktur enthüllt Muster, Trends und Tendenzen in den Daten. Unstrukturierte und isolierte Daten besitzen dagegen – wie die unverbundenen Atome – nur einen geringen Wert.

Flache Dateien

Die einfachste Sammlung strukturierter Daten stellt eine sogenannte *flache Datei* dar. Flache Dateien besitzen nur eine minimale Struktur. Sie bestehen einfach aus einer 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. Für Computer sind flache Dateien sehr einfach aufgebaut. Weil die Dateien keinerlei Informationen über ihre eigene Struktur enthalten (keine Metadaten), ist der enthaltene Overhead (Informationen in der Datei, die selbst keine Daten sind, aber Speicherplatz belegen) minimal.

Angenommen, Sie wollten die Namen und Adressen der Kunden Ihrer Firma in einer flachen Datei verwalten. Diese soll ähnlich wie das folgende Beispiel strukturiert sein:

```
Klaus Klawuttke    Heimweg 15      12345 Berlin
Jerry Cotton       Gasse 77        48787 Hermannsburg
Adrian Hansen      Perlenallee 3   58789 Oberstadt
Johann Bäcker      Terrenshof 128  98755 Germsdorf
Michael Pens       Kalauerweg 1    57730 Köln
Barbara Michimoto  Sandufer 9      25252 Kiel
Linda Schmidt      Vennweg 56      44134 Düsseldorf
Robert Funnell     Kölner Ring 9   24240 Lübeck
Boris Reckal       Am Hof 9        59554 Siegen
Kevin Kirenberg    Pferdeweg 7     35305 Kassel
```

Diese Datei enthält 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 vorgegeben. Alle Programme, die diese Datei benutzen, müssen diese Felddefinitionen »kennen«, da diese Angaben nicht in der Datenbank selbst gespeichert sind.

Wegen des geringen Overheads kann man mit flachen Dateien sehr schnell arbeiten. Nachteilig ist jedoch, dass Anwendungsprogramme zusätzliche Logik enthalten müssen, um die Daten einer flachen Datei auf Detailebene 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 umfangreicher das System jedoch wird, desto umständlicher wird die Verarbeitung flacher Dateisysteme.

Wenn Sie stattdessen eine Datenbank benutzen, können Sie den Doppelaufwand verringern. Obwohl Datenbankdateien möglicherweise einen größeren Overhead haben, können die Anwendungen leichter auf andere Hardware- und Betriebssystemplattformen übertragen werden. Außerdem erleichtern Datenbanken das Schreiben von Anwendungsprogrammen, weil Programmierer keine Details über den Speicherort und die Speicherform der Daten wissen müssen.

Datenbanken vermeiden Doppelarbeit, weil das DBMS die Details der 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 (redundant) den gesamten Code für die Datenbearbeitung enthalten. Bei einem DBMS ist dieser Code in der Anwendung überflüssig.

Verständlicherweise 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 den gesamten spezifischen Hardware-Code ändern müssen. DBMS-basierte Anwendungen lassen sich sehr viel leichter auf andere Plattformen portieren.

Datenbankmodelle

Die ersten Datenbanken, die in den 1950er-Jahren eingeführt wurden, waren nach dem hierarchischen Modell strukturiert. Sie litten unter Redundanzproblemen, und ihre inflexible Struktur erschwerte Datenbankänderungen. Bald darauf folgten Datenbanken, die nach dem Netzwerkmodell strukturiert waren und die Hauptnachteile der hierarchischen Datenbanken beseitigen sollten. Netzwerkdatenbanken verfügen über eine minimale Redundanz auf Kosten einer hohen strukturellen Komplexität.

Einige Jahre später entwickelt Dr. E. F. Codd bei IBM das relationale Modell, das minimale Redundanz und leicht verständliche Struktur miteinander kombiniert. Die Sprache SQL wurde für das Arbeiten mit relationalen Datenbanken entwickelt. Relationale Datenbanken haben hierarchische Datenbanken und Netzwerkdatenbanken fast vollständig verdrängt.

Seit etlichen Jahren werden auch sogenannte *NoSQL-Datenbanken* auf den Markt gebracht. Diese sind nicht wie relationale Datenbanken strukturiert und arbeiten nicht mit SQL als Abfragesprache. NoSQL-Datenbanken werden in diesem Buch nicht behandelt.

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 wurde einer kleinen, neu gegründeten Firma zuteil, die ihr Produkt *Oracle* nannte.

Relationale Datenbanken haben Datenbanken der früher verwendeten Modelle weitgehend verdrängt, weil man die Struktur relationaler Datenbanken ändern kann, ohne die Anwendungen ändern zu müssen, die noch mit der alten Struktur gearbeitet haben. Angenommen, Sie wollten eine oder mehrere neue Spalten in eine Datenbanktabelle einfügen. Dann müssen Sie vorher erstellte Anwendungen, die mit dieser Tabelle arbeiten, nur dann ändern, wenn Sie eine oder mehrere der in den Anwendungen verwendeten Spalten ändern.

Wenn Sie eine Spalte löschen, auf die eine vorhandene Anwendung Bezug nimmt, die also von einer vorhandenen Anwendung referenziert wird, bekommen Sie natürlich Probleme, und zwar unabhängig vom verwendeten Datenbankmodell. Eine der wirksamsten Methoden, einen Fehler in einer Datenbankanwendung zu verursachen, besteht darin, Daten aus Spalten abzurufen, die in der Datenbank nicht vorhanden sind.

Komponenten relationaler Datenbanken

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, sofern die betroffene Tabelle den anderen Tabellen nicht *übergeordnet* ist.

(Die Beziehungen der Über- und Unterordnung 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.

Was sind Relationen?

Datenbanken bestehen aus miteinander verknüpften Tabellen. In der Datenbankterminologie werden diese Tabellen auch *Relationen* genannt. Eine relationale Datenbank besteht also aus einer oder mehreren Relationen.

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.

Ein Beispiel: Die meisten Benutzer kennen zweidimensionale Arrays, die aus Zeilen und Spalten bestehen, in Form von Arbeitsblättern elektronischer Tabellenkalkulationen (wie zum Beispiel *Microsoft Excel*). Die auf der Rückseite der Baseball-Karten von Baseball-Spielern der Major-League aufgedruckte Offensivstatistik stellt ein weiteres Beispiel für ein Datenfeld dar. Die Baseball-Karte enthält Spalten für das Jahr, die Mannschaft (Team), die geleisteten Spiele sowie für die verschiedenen Leistungskriterien (At Bat, Hits, Runs und so weiter), 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. Im praktischen Einsatz 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
			117	329	99	81	25						
Roberts	1989	Padres			172	104		15	8	3	49	21	.301

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. Das DBMS verarbeitet die Tabelle unabhängig von deren Aufbau.

Jede Spalte einer Datenbanktabelle stellt ein einzelnes Attribut der Tabelle dar. Die Bedeutung einer Spalte ist für alle Zeilen der Tabelle gleich. Eine Tabelle kann beispielsweise die Namen, Adressen und Telefonnummern aller Kunden einer Firma enthalten. Zeilen in der Tabelle, die auch *Datensatz* oder *Tupel* genannt werden, enthalten jeweils 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`) und so weiter. Abbildung 1.3 zeigt einige Zeilen und Spalten einer solchen Tabelle.

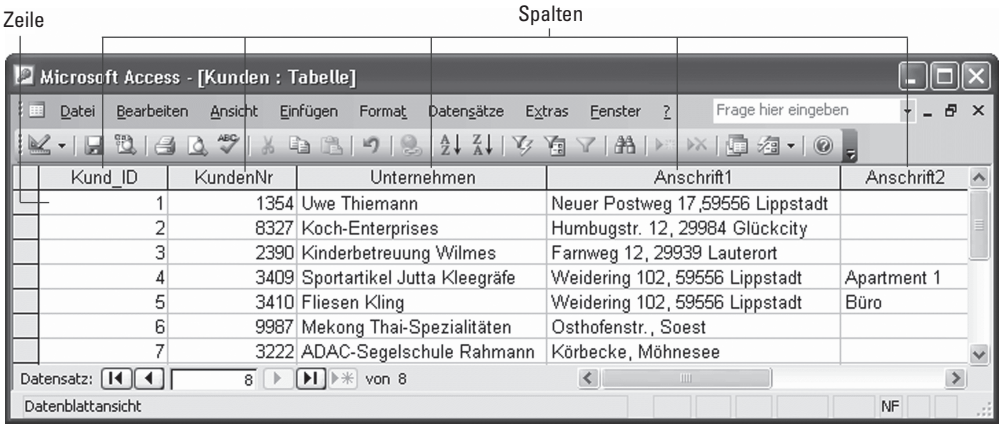

Abbildung 1.3: Jede Tabellenzeile enthält einen Datensatz; jede Spalte enthält ein einzelnes Attribut.

Die Relationen in diesem Datenbankmodell entsprechen den *Tabellen* einer auf dem Modell basierenden Datenbank.

Views oder Sichten

Tabellen können viele Spalten und Zeilen enthalten. Manchmal interessieren Sie sich für alle Daten, bei umfangreichen Tabellen aber oft auch nicht. Möglicherweise möchten Sie nur einige der Spalten einer Tabelle sehen oder nur Zeilen, die einer bestimmten Bedingung genügen. Vielleicht möchten Sie einige Spalten einer Tabelle und einige andere Spalten einer mit ihr verknüpften Tabelle sehen. Um die Daten auszuschließen, die Sie augenblicklich nicht interessieren, erstellen Sie eine sogenannte *Sicht* (englisch *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 auch *virtuelle Tabellen* genannt. Aus der Perspektive von Anwendungen oder Benutzern verhalten sich Sichten wie Tabellen. Sichten existieren jedoch nicht unabhängig von Tabellen. Mit Views können Sie Daten betrachten, wobei die Views selbst kein Bestandteil der Daten sind.

Angenommen, Sie arbeiten mit einer Datenbank, in der die beiden Tabellen `Kunde` und `Rechnung` enthalten sind. 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 will nur den Vornamen (`Vorname`), den Nachnamen (`Nachname`) und die Telefonnummer (`Tel`) eines jeweiligen Kunden auf dem Bildschirm sehen. Der Verkaufsleiter kann nun für die Tabelle `Kunde` eine Sicht erstellen, die nur die drei benötigten Spalten und nicht auch noch die momentan überflüssigen Informationen aus anderen Spalten enthält. Abbildung 1.4 zeigt die vom Verkaufsleiter erstellte Sicht.

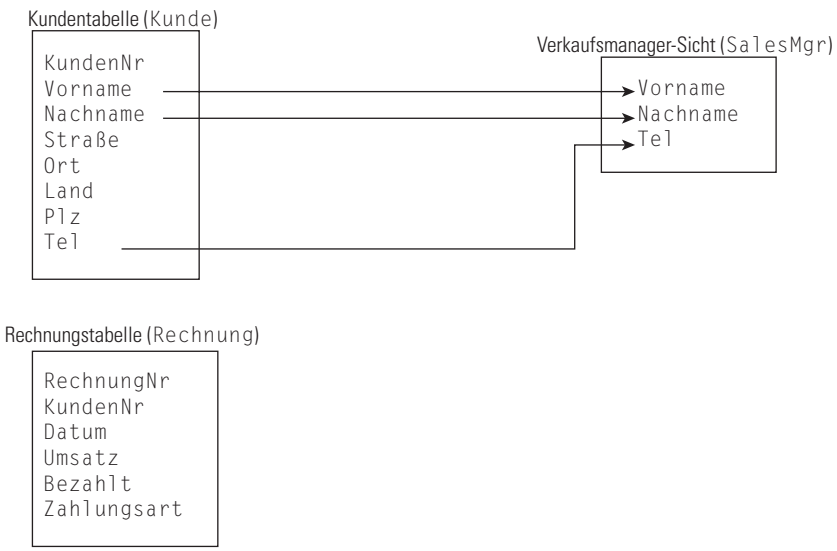

Abbildung 1.4: Die für den Verkaufsmanager definierte View wird von der Tabelle KUNDE abgeleitet.

Ein Zweigstellenleiter benötigt die Namen und Telefonnummern aller Kunden, deren Postleitzahl zwischen 90000 und 93999 liegt. Eine Sicht, die von ihr abgerufene Zeilen und angezeigte Spalten mit einer sogenannten Einschränkung versieht und sie damit filtert, führt diese Aufgabe aus. Abbildung 1.5 zeigt die Datenquellen für die Spalten der Sicht des Zweigstellenleiters.

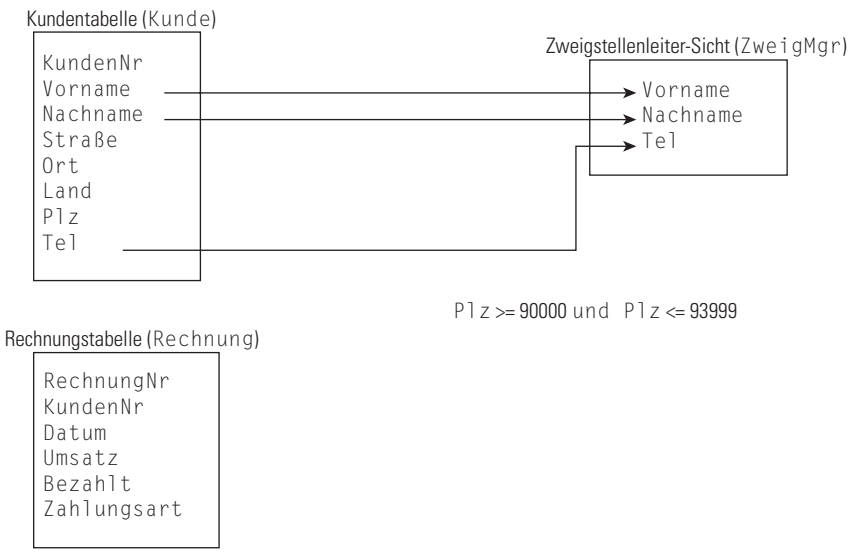

Abbildung 1.5: Die Sicht des Zweigstellenleiters enthält nur bestimmte Zeilen der Tabelle KUNDE.

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.

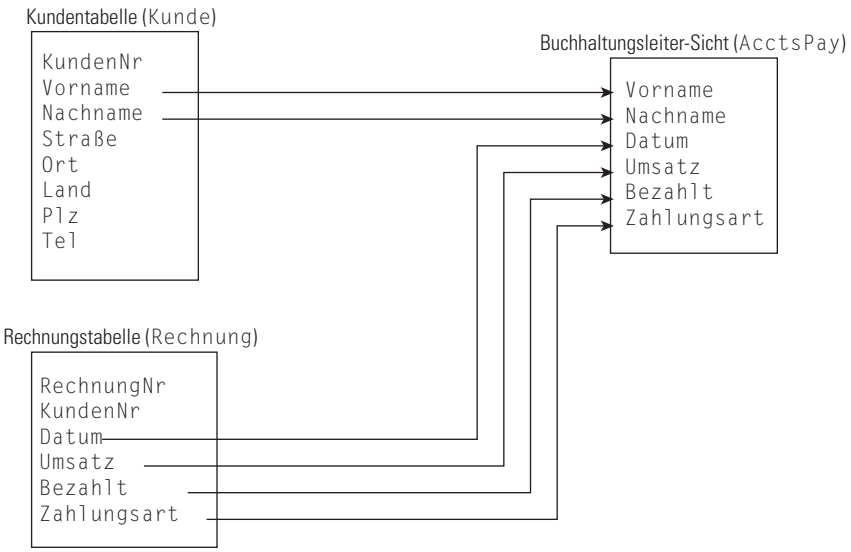

Abbildung 1.6: In der View für den Buchhaltungsleiter werden Daten aus zwei Tabellen kombiniert.

Sichten sind nützlich, weil sie es ermöglichen, Daten aus Datenbanken zu extrahieren und geeignet zu formatieren, ohne dabei an den gespeicherten Daten wirklich etwas zu ändern. Zugleich bieten sie auch für jene Daten einen gewissen Schutz, die nicht angezeigt werden sollen, weil sie in den Sichten nicht enthalten sind. In Kapitel 6 erfahren Sie, wie mit SQL Sichten erstellt werden.

Schemata, Domänen und Einschränkungen

Eine Datenbank ist mehr als eine Sammlung von Tabellen. Zusätzliche Strukturen werden über mehrere Schichten verteilt und helfen, die Integrität der Daten zu wahren. Das *Schema* einer Datenbank sorgt für die übergreifende Organisation der Tabellen. Die *Domäne* einer Tabellenspalte sagt Ihnen, welche Werte Sie in der Spalte speichern dürfen. Sie können *Einschränkungen* (*Constraints*) für eine Datenbanktabelle definieren, um zu verhindern, dass unzulässige Daten in der Tabelle gespeichert werden können.

Schemata

Die Struktur einer ganzen Datenbank wird als ihr *Schema* oder als *konzeptionelle Sicht* und manchmal auch *vollständige logische Sicht* der Datenbank genannt. 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, also die Wertemenge, wird im Relationenmodell *Domäne* des Attributs genannt.

Angenommen, Sie verkaufen als Autohändler das neue Curarri GT 4000 Sportcoupé. 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 (*Constraints*) 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 Benutzer ungültige Daten in diese Spalte 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 zulässige Wertemenge der Spalte 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.

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 nur dann fortgesetzt werden, wenn der Benutzer eine gültige Farbe in das Feld `Farbe` eingibt.

Vielleicht fragen Sie sich, was passiert, wenn der Hersteller Curarri eine waldgrüne Version des GT 4000 als Sommeroption anbietet. Die Antwort impliziert eine Arbeitsplatzgarantie für Datenbankprogrammierer, denn derartige Dinge passieren immer wieder und erfordern Anpassungen der Datenbankstruktur. Dann können nur Fachleute (zu denen auch Sie gehören), die die Strukturen zu ändern wissen, größere Katastrophen verhindern.

Das Objektmodell fordert das relationale Modell heraus

Das relationale Modell hat in vielen Anwendungsbereichen fantastische Erfolge gefeiert. 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 möglichst die besten 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. Im Gegensatz zu dem SQL-92-Standard, der ein rein relationales Datenbankmodell definiert, beschreibt SQL:1999 deshalb ein objektrelationales Datenbankmodell. SQL:2003 enthält mehr objektorientierte Funktionen; und die nachfolgenden SQL-Versionen wurden in dieser Richtung weiterentwickelt.

Ich beschreibe in diesem Buch den internationalen ISO-/IEC-SQL-Standard. (IEC steht für *International Electrotechnical Commission*, aber das spielt eigentlich keine Rolle. Wie viele Menschen wissen, wofür die Buchstaben der Abkürzung *LASER* stehen?) 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 aus späteren SQL-Versionen. Die objektorientierten Funktionalitäten des neuen Standards geben Entwicklern die Möglichkeit, SQL-Datenbanken für die Lösung von Problemen einzusetzen, die für den früheren, rein relationalen Ansatz zu komplex waren. Anbieter von DBMS-Systemen wie zum Beispiel Oracle bauten die objektorientierten Funktionen aus dem ISO-Standard in ihre Produkte ein. Einige dieser Funktionen werden schon seit Jahren angeboten, während andere noch nicht in die Sprache integriert wurden.

Überlegungen zum Datenbankentwurf

Eine Datenbank ist eine Repräsentation einer physischen oder konzeptionellen Struktur, wie eines Unternehmens, der Montage eines Autos oder der Erfolgsstatistik aller Mannschaften der Bundesliga. Die Genauigkeit dieser Repräsentation hängt davon ab, wie detailliert Ihr Datenbankentwurf ist. Der Aufwand, den Sie in den Datenbankentwurf investieren, sollte von der Art der Informationen abhängen, die Ihnen die Datenbank liefern soll. Mit zu vielen Details wird nur Arbeit, Zeit und Speicherplatz verschwendet. Doch zu wenig Details können Datenbanken wiederum wertlos werden lassen. Näheres zum Datenbankentwurf finden Sie beispielsweise in »Datenbanksysteme für Dummies«.

Ermitteln Sie, wie viele Details aktuell benötigt werden und welche möglicherweise zukünftig noch erforderlich werden könnten, und entwerfen Sie die Datenbank exakt für den ermittelten Bedarf – nicht für mehr und nicht für weniger. Sie sollten aber nicht überrascht sein, wenn Sie den Entwurf später ändern müssen, um ihn an geänderte Anforderungen anzupassen.

Die heutigen Datenbankverwaltungssysteme mit ihren optisch attraktiven grafischen Benutzeroberflächen und intuitiven Entwicklungswerkzeugen können Möchtegern-Datenbankentwickler in falscher Sicherheit wiegen. Derartige Systeme suggerieren, dass der Entwurf einer Datenbank mit dem Aufbau des Arbeitsblatts in einer Tabellenkalkulation oder einer relativ 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 sehr schnell unbrauchbar werden kann. Oft fällt das Problem erst auf, wenn in die Datenerfassung bereits sehr viel Aufwand gesteckt wurde. Wenn Sie das Problem bemerken, ist es oft bereits zu spät. In vielen Fällen besteht die einzig praktikable Lösung dann darin, die Datenbank komplett neu zu entwerfen und alle Daten noch einmal einzugeben. Daraus sollten Sie dann allerdings etwas dazugelernt haben.