



Abb. 10.3: Beispiel eines E/R-Diagramms

Ein Beispiel eines E/R-Diagramms zeigt Abb. 10.3. Es beschreibt die „Miniwelt“ eines Industrieunternehmens, in dem als Entitätstypen Angestellte, Abteilung, Produkt und Kunde auftreten. Es gibt zwei 1:n-Beziehungstypen (*arbeitet in*, *stellt her*) und einen n:m-Beziehungstyp (*bestellt*). Da die Beziehung *bestellt* eine n:m-Beziehung ist, kann ein Kunde verschiedene Produkte bestellen und ein Produkt von verschiedenen Kunden bestellt werden. Bei der n:1-Beziehung *arbeitet in* ist zu erkennen, dass ein Angestellter zu höchstens einer Abteilung gehören darf. Der Entitätstyp Kunde besitzt insgesamt vier Attribute: *KId* ist eine eindeutige Kundennummer und wird als Primärschlüssel verwendet, *KName* bezeichnet den Namen, *Wohnsitz* ist der Hauptwohnsitz des Kunden. Da ein Kunde im Allgemeinen noch nicht alle Rechnungen bezahlt hat, wird die Summe seiner Fälligkeiten in dem Attribut *Kto* vermerkt. Wir werden in diesem Kapitel immer wieder auf dieses Beispiel zurückgreifen. Es sei hier ausdrücklich darauf hingewiesen, dass das E/R-Diagramm nur einen Auszug des gesamten E/R-Modells visualisiert und dass deshalb zusätzlich zu dem Diagramm das eigentliche E/R-Modell in einer Dokumentation noch vollständig beschrieben werden muss.

Insgesamt stellt das E/R-Datenmodell die Grundlage für einen guten Datenbankentwurf dar. Die E/R-Modellierung wird auch in anderen Bereichen wie z.B. bei der Entwicklung von Software verwendet. Es wurden vielfältige Erweiterungen des E/R-Datenmodells vorgestellt, z.B. in dem Buch von Vossen, die sich auch in der Praxis gegenüber dem ursprünglichen Modell bewährt haben.

10.2.2 Das Relationale Datenbankmodell

Das *relationale Datenmodell* wurde in seiner ursprünglichen Form 1970 von Codd vorgestellt. Die heute marktbeherrschenden DBVS wie z.B. Oracle, SQL Server, Sybase, Informix und DB2 orientieren sich im Wesentlichen an diesem relationalen Modell. Der folgende Abschnitt beschreibt die Grundlagen, die für das Verständnis der erwähnten Datenbanksysteme von großer Wichtigkeit sind.

10.2.3 Relationen

Der Begriff der *Relation* ist von zentraler Bedeutung innerhalb des relationalen Modells. Zu einer Relation R gibt es ein *Relationenschema*, das aus dem Namen der Relation und einer Folge von Attributen besteht. Zu jedem *Attribut* A_i gehört ein *Wertebereich* D_i , für den im Allgemeinen nur atomare Werte wie ganze Zahlen, Fließkommazahlen und Zeichenketten zugelassen sind, nicht aber strukturierte und mengenwertige Daten.

Der Inhalt einer in der Datenbank gespeicherten Relation ist zeitlichen Veränderungen unterworfen. Zu jedem Zeitpunkt hat man daher eine bestimmte *Instanz* I_R von R vorliegen. Unter einer Instanz verstehen wir hier eine *Teilmenge* des kartesischen Produkts der zugehörigen Wertebereiche, also das, was man ansonsten in der Mathematik unter einer Relation versteht: $I_R \subseteq D_1 \times \dots \times D_n$. Ein Element einer Instanz wird *Tupel* oder *Datensatz* genannt. Eine *Relation* ist daher genau genommen die Menge aller erlaubten Instanzen eines vorgegebenen Relationenschemas. Der Begriff der Relation wird oft aber auch synonym für die (aktuelle) Instanz benutzt.

Der bisherige Begriff der Relation ist für praktische Zwecke zu weit gefasst, da Datensätze zugelassen werden, die in der realen Welt nicht auftreten können. Um die Möglichkeiten der Relationsinstanzen in diesem Sinne einzuschränken, führt man *Integritätsbedingungen* ein, die für alle Instanzen einer Relation erfüllt sein müssen. Ein Beispiel für eine Integritätsbedingung ist der Primärschlüssel einer Relation. Entsprechend dem Begriff des Primärschlüssels im E/R-Datenmodell bezeichnet der *Primärschlüssel* im relationalen Modell eine minimale Menge von Attributen, welche die Datensätze der Relation eindeutig identifiziert. Es darf also keine Instanz einer Relation geben, in der zwei Tupel existieren, die bezüglich ihres Primärschlüssels gleich sind. Primärschlüssel werden im relationalen Modell auch als „Zeiger“ benutzt, die insbesondere zur Modellierung von Beziehungen verwendet werden.

Relationen lassen sich übersichtlich als Tabellen veranschaulichen. Abbildung 10.4 stellt z.B. eine Relation *Kunde* als Tabelle dar. Die Attribute der Relation werden dabei in einer Kopfzeile aufgeführt. Jede weitere Zeile der Tabelle enthält genau einen Datensatz. Die Repräsentation von Relationen durch Tabellen ist so intuitiv, dass viele DBVS eine entsprechende grafische Benutzerschnittstelle anbieten.

KId	KName	Wohnsitz	Kto
2	Lutz	Darmstadt	0
3	Feldmann	Bremen	-2000
5	Seidl	München	-1000
7	Breitenbach	Darmstadt	0

Abb. 10.4: Darstellung der Relation *Kunde* als Tabelle

10.2.4 Die relationale Algebra

Die *relationale Algebra* stellt eine Menge von ein- und zweistelligen Operatoren zur Verfügung. Diese Operatoren berechnen zu einer bzw. zwei Relationen eine neue Relation. Sei im Folgenden Rel die Menge aller möglichen Relationen. Im Wesentlichen umfasst die relationale Algebra folgende fünf Operatoren:

- *Vereinigung* $\cup : (Rel, Rel) \rightarrow Rel$: Zu zwei Relationen mit gleichem Relationenschema wird die Vereinigung ihrer Instanzen berechnet.
- *Differenz* $- : (Rel, Rel) \rightarrow Rel$: Zu zwei Relationen mit gleichem Relationenschema wird die Mengendifferenz ihrer Instanzen berechnet.
- *Kartesisches Produkt* $\times : (Rel, Rel) \rightarrow Rel$: Zu zwei Relationen wird das kartesische Produkt ihrer Instanzen gebildet.
- *Projektion* $\pi_T : Rel \rightarrow Rel$: Zu einer Relation und einer Teilmenge T ihres Relationenschemas wird eine Relation mit Schema T erzeugt, deren Instanz alle Tupel aus der Eingabere relation eingeschränkt auf T enthält.
- *Selektion* $\sigma_P : Rel \rightarrow Rel$: Zu einer Relation und einem Prädikat P werden alle Tupel der Relation berechnet, die das Prädikat P erfüllen.

Als sechster Operator wird die *Umbenennung* $\rho : Rel \rightarrow Rel$ benötigt, die es ermöglicht, Attribute und Relationen umzubenennen (ohne dabei die Instanz der Relation zu verändern). Man beachte bei der Projektion und bei der Vereinigung, dass eine Instanz einer Relation eine Menge ist und deshalb keine Duplikate existieren dürfen.

Bei den zweistelligen Operatoren wird die Infixnotation und bei den einstelligen Operatoren die funktionale Schreibweise benutzt. Betrachten wir z.B. die in Abb. 10.4 aufgeführte Relation $Kunde$. Durch die Operation $\sigma_{Kto < 0}(Kunde)$ wird eine Relation erzeugt, deren Instanz aus zwei Datensätzen besteht. Die Operation $\pi_{Wohnort, Kto}(Kunde)$ erzeugt eine Relation mit den Attributen $Wohnort$ und Kto . Die Instanz enthält insgesamt drei Datensätze, da die Datensätze mit KId 2 und 7 gleich sind. Durch Verschachtelung von mehreren Operatoren der relationalen Algebra können auch komplexere Anfragen formuliert werden. Sollen z.B. nur die Namen der Kunden mit negativem Kontostand ausgegeben werden, so kann das Resultat durch den Ausdruck $\pi_{KName}(\rho_{Kto < 0}(Kunde))$ berechnet werden.

Auf Basis dieser wenigen Operatoren lassen sich noch weitere Operatoren ableiten. Beispielsweise kann der Schnitt zweier Relationen (mit gleichen Relationenschema) durch eine Kombination von Vereinigung und Differenz ausgedrückt werden. Eine der wichtigsten Operatoren im relationalen Datenmodell ist der *Verbund* (engl. *join*). Der so genannte θ -*Verbund* verknüpft zwei Relationen bezüglich des Prädikats θ miteinander, indem alle Tupel des kartesischen Produkts betrachtet werden, die das Prädikat θ erfüllen. Der θ -Verbund lässt sich also als eine Kombination von kartesischem Produkt und Selektion ausdrücken.

10.2.5 Erweiterungen des relationalen Datenmodells

Es gibt einige wichtige Unterschiede zwischen den „relationalen“ Datenmodellen, die in einem kommerziellen DBVS verwendet werden, und dem relationalen Datenmodell in seiner ursprünglichen Form. In kommerziellen DBVS gibt es nicht nur die Relationen (Menge von Tupeln), sondern auch Relationen, deren Instanzen Duplikate enthalten und somit Multimengen darstellen (*M-Relation*). Des Weiteren kann auf Relationen auch eine Ordnung definiert werden (*O-Relation*). Daraus erwächst die Notwendigkeit, weitere Operationen einzuführen, die verschiedene Typen von Relationen ineinander überführen. Um zu einer Relation eine entsprechende O-Relation zu erzeugen, wird z.B. ein *Sortieroperator* benötigt. Zu einer M-Relation kann durch *Eliminierung aller Duplikate* eine gewöhnliche Relation erzeugt werden.

Eine weitere zentrale Erweiterung des relationalen Modells stellt die Berechnung von Aggregaten (wie z.B. Summen- und Durchschnitten) dar. Ein *Aggregationsoperator* berechnet zu einer Relation eine neue Relation, die genau ein Attribut und genau ein Tupel besitzt. Soll z.B. die Summe aller Kontostände der Relation `Kunde` berechnet werden, könnte dies durch den Operator $\text{sum}_{\text{Kto}}(\text{Kunde})$ erfolgen. Dieses Konzept kann noch durch die Einführung eines Gruppierungsoperators verfeinert werden. Zu einer vorgegebenen Teilmenge des Relationenschemas (den so genannten Gruppierungsattributen) unterteilt der *Gruppierungsoperator* die Tupel der Instanz in disjunkte Teilmengen, so dass in jeder Teilmenge die Tupel bezüglich den Gruppierungsattributen gleich sind. Für jede dieser Teilmengen wird genau ein Aggregat berechnet. Das Schema der Ergebnisrelation besteht dabei aus den Gruppierungsattributen und dem Aggregationsattribut.

In den letzten Jahren haben sich relationale DBVS zu objekt-relationalen Systemen gewandelt. Insbesondere das relationale Datenmodell wurde durch Berücksichtigung objekt-orientierter Aspekte erheblich erweitert. So ist es inzwischen z.B. erlaubt, dass Attribute strukturierte Daten besitzen können. Diese neuen Entwicklungen sind aber noch nicht vollständig ausgereift, so dass wir in dieser kurzen Einführung nicht weiter darauf eingehen werden.

10.2.6 Vom E/R-Datenmodell zu einem relationalen Modell

In der Einführung dieses Kapitels wurde bereits die Vorgehensweise bei der Erstellung einer Datenbank erläutert. Nachdem auf Basis einer Anforderungsanalyse ein E/R-Diagramm erstellt wurde, muss dieses in ein relationales Modell überführt werden. Es wird dabei folgendermaßen vorgegangen:

- Für jeden Entitätstyp E des E/R-Modells wird eine Relation R_E angelegt, die alle Attribute des Entitätstyps E enthält.
- Für jede 1:n-Beziehung zwischen einem Entitätstyp F und dem Typ E wird der Primärschlüssel von F als Attribut in R_E importiert. Der Primärschlüssel der Relation R_E ergibt sich direkt aus dem des Entitätstyps E .
- Eine n:m-Beziehung B wird als eigenständige Relation R_B umgesetzt, die alle Attribute des Beziehungstyps enthält. Zusätzlich werden die Primärschlüssel der an der Beziehung

beteiligten Entitätstypen als Attribute in der Relation R_B aufgenommen. Diese importierten Primärschlüssel bilden den Primärschlüssel der Relation R_B .

Betrachten wir wieder unser Beispiel aus Abbildung 10.3. Dieses E/R-Datenmodell wird in ein relationales Modell mit fünf Relationen umgesetzt. Jeder der vier Entitätstypen wird in eine eigene Relation abgebildet. Man beachte, dass bei den Entitätstypen *Angestellte* und *Produkt* als Fremdschlüssel der Primärschlüssel des Entitätstypen *Abteilung* (*Abt-Name*) aufgenommen wird. Die fünfte Relation ergibt sich aus dem n:m-Beziehungstyp *bestellt*. Das Relationenschema von *bestellt* besteht aus drei Attributen *PId*, *KID* und *Anzahl*, wobei sich der Primärschlüssel aus *PId* und *KID* zusammensetzt.

10.3 Die Anfragesprache SQL

Mitte der 70er Jahre wurde ein erster Prototyp eines relationalen DBVS (System R) im Forschungslabor Almaden von IBM entwickelt. Die dabei konzipierte Anfragesprache SQL ist heute in leicht modifizierter Form die Sprache der relationalen DBVS. Der gegenwärtige Standard wird auch als SQL2 bzw. SQL92 bezeichnet. Derzeit befindet sich SQL3 bzw. SQL99, das erhebliche Erweiterungen zu SQL2 vorsieht, (immer noch) in der Entwicklung. Trotz des SQL2-Standards bietet jeder Anbieter relationaler DBVS einen erweiterten Sprachumfang an. Möchte man eine Datenbank und die entsprechenden Anwendungsprogramme unabhängig vom zugrundeliegenden DBVS entwickeln, so ist zu empfehlen, sich auf den standardisierten Sprachumfang zu beschränken. Auch sei hier darauf verwiesen, dass SQL nicht eine Programmiersprache im herkömmlichen Sinn ist, sondern im Vergleich zu Java nur relativ eingeschränkte Möglichkeiten bietet. Kontrollstrukturen wie Schleifen und bedingte Anweisungen gibt es in SQL nicht. Das ursprüngliche Ziel der Entwickler von SQL war es, eine einfache *deskriptive* Anfragesprache zu entwickeln. Der Anwender sollte bei einer Anfrage an das DBS nur beschreiben, *was* als Ergebnis verlangt ist, aber nicht, *wie* das Ergebnis berechnet werden soll. Das DBVS leistet letztendlich die Umsetzung der Anfrage in einen entsprechenden Ausführungsplan.

SQL bietet dem Benutzer die Möglichkeit, einerseits Schemata anzulegen (*Datendefinition*) und andererseits die Daten der Datenbank anzusprechen (*Datenmanipulation*). Beide dieser Möglichkeiten werden im Folgenden diskutiert. Unser Ziel ist es dabei, dem Leser an einigen Beispielen wichtige Prinzipien von SQL zu erläutern, ohne auf die Details der Sprache einzugehen.

10.3.1 Datendefinition

Wir haben bereits zu Beginn dieses Kapitels gesehen, dass eine Datenbank in drei Ebenen unterteilt ist. Auf allen diesen kann durch SQL ein Schema angelegt bzw. modifiziert werden.

Auf der konzeptuellen Ebene werden die Relationen der Datenbank angelegt. Dies geschieht durch den Befehl **create table** und der Angabe eines Relationennamens. Jedes Attribut der Relation muss mit Namen und Datentyp aufgeführt werden. Weiterhin gibt es eine Vielzahl von so genannten Integritätsregeln, die bei der Definition einer Relation angegeben wer-