Datenbanksysteme im Klartext
ISBN 3-8273-7066-3

3 Relationale Datenbanken 1 – die wesentlichen Ideen

Relationale Datenbanken dominieren zur Zeit den Markt. Sie stellen bemerkenswert einfache Mittel zur Darstellung und Manipulation von Daten zur Verfügung. Daneben besitzen sie auch fundierte theoretische Grundlagen. In diesem Kapitel werden wir die wichtigsten Aspekte des relationalen Modells mit Hilfe einfacher Definitionen und Beispiele umreißen.

Wer dieses Kapitel durchgearbeitet hat, sollte Folgendes können:

1. Die wichtigsten Eigenschaften relationaler Datenbanken beschreiben
2. Die wichtigsten Bestandteile der relationalen Theorie erläutern, als da sind: relationale Datenstrukturen, relationale Datenmanipulation und relationale Datenintegrität
3. Die Bedeutung der Ausdrücke „minimal relational“, „relational vollständig“ und „voll relational“ verstehen

Die relationale Theorie hat drei wesentliche Komponenten: Datenstrukturen, Datenmanipulation und Datenintegrität. Diese werden wir nacheinander untersuchen.

3.1 Relationale Datenstrukturen

Relationale Datenbanken speichern alle Daten in einfachen zweidimensionalen Tabellen, die man Relationen nennt. Tabelle 3.1 zeigt ein Beispiel einer Relation mit Daten über Angestellte einer Firma.

ANG

ANGNR	ANGNAME	ABTNAME	STUFE
1	F JONES	VERKAUF	6
2	P SMITH	BUCHHALTUNG	6
3	K CHAN	VERKAUF	4
4	J PETERS	VERKAUF	5
9	S ABDUL	BUCHHALTUNG	3

Tabelle 3.1: Die Relation Ang

Alle Tabellen in relationalen Datenbanken sehen so aus. Wie gesagt bezeichnet man Tabellen korrekterweise als Relationen; wir verwenden jedoch beide Begriffe. In einem sehr vagen Sinn entspricht eine Relation ungefähr (aber wirklich nicht genau)

einer Entität in einem E/R-Diagramm. Für jeden relevanten Gegenstand in einer Datenbank erstellen wir eine Relation. Alle Gegenstände mit den gleichen Eigenschaften werden in derselben Relation abgelegt. Die Eigenschaften von Gegenständen sind in den Überschriften der Spalten festgehalten (ANGNR, ANGNAME, ABTNAME, STUFE). Die konkreten einzelnen Vorkommen der Relation ANG sind durch die Zeilen von Daten in der Tabelle repräsentiert. Die Bedeutung jeder Zeile erschließt sich leicht: Zum Beispiel beschreibt die erste Zeile ein ANG mit der ANGNR 1, dem ANGNAME F JONES, dem ABTNAME VERKAUF und der STUFE 6. Wir sehen bei dieser Tabelle eine Reihe von Eigenschaften, die allen Tabellen in relationalen Datenbanken zu Eigen sind.

3.1.1 ÜBERSCHRIFTEN UND EINTRÄGE

1. Die Überschrift. Alle Relationen haben eine. Sie besteht aus dem Namen der Relation (ANG) und den Namen der Spalten, aus denen die Relation besteht. Der korrekte Begriff für die Spalten einer Relation ist „Attribute“. Die Anzahl Attribute bestimmt den *Grad* der Relation. Diese Relation hat den Grad vier. In der relationalen Theorie ist die Anzahl Attribute einer Relation nicht beschränkt, doch in der Praxis setzen die meisten relationalen DBMS eine obere Schranke.
2. Die Einträge. Diese sind durch die Zeilen der Relation gegeben. Korrekt bezeichnet man sie als Tupel. Ein Tupel ist eine geordnete Liste von Werten. Die Bedeutung jedes Wertes ist durch seine Position im Tupel bestimmt. So ist im ersten Tupel der erste Wert (1) die ANGNR, der zweite (F JONES) der ANGNAME und so weiter. Die Kardinalität einer Relation ist die Anzahl Tupel, die sie enthält. Hier haben wir somit eine Relation der Kardinalität fünf.

Jede Relation verfügt über eine Menge von Tupeln. Es ist wichtig, dass es sich hierbei tatsächlich um eine Menge im mathematischen Sinn handelt. Mengen sind ungeordnete Ansammlungen verschiedener Gegenstände. Oben haben wir die Tupelmenge in der Reihenfolge ihrer ANGNR angegeben. Diese Reihenfolge ist an sich ohne Bedeutung; es gibt keine implizite Information der Art, dass die ANGNR 1 „größer“ wäre als ANGNR 6. Tupel einer Relation können in beliebiger Reihenfolge gespeichert und ausgegeben werden. Die meisten relationalen Systeme verwenden einfach die Reihenfolge, in der die Tupel der Tabelle hinzugefügt wurden. Falls wir einen neuen Angestellten mit der ANGNR 5 hinzufügen, würden wir dann die Tabelle 3.2 erhalten.

ANG

ANGNR	ANGNAME	ABTNAME	STUFE
1	F JONES	VERKAUF	6
2	P SMITH	BUCHHALTUNG	6
3	K CHAN	VERKAUF	4
4	J PETERS	VERKAUF	5
9	S ABDUL	BUCHHALTUNG	3
5	J LEWIS	FORSCHUNG	5

Tabelle 3.2: Nach dem Einfügen eines neuen Tupels

3.1.2 Domänen

Für jedes Attribut gibt es offensichtliche Einschränkungen hinsichtlich der Daten, die es enthalten kann. Für ANGNR haben wir ganze Zahlen, ANGNAME ist eine Zeichenkette und so weiter. Der Wertebereich, den ein Attribut annehmen kann, heißt seine Domäne. Auf einer gewissen Ebene ist eine Domäne etwas sehr Ähnliches wie ein Datentyp bei der Programmierung. Wie ein Datentyp legt eine Domäne nicht nur den Wertebereich eines Attributes fest, sie bestimmt auch, welche Operationen auf diese Werte anwendbar sind: Addition oder Subtraktion von Zahlen, Aufteilen oder Aneinanderreihen von Zeichenketten und so fort. Auf einer höheren Ebene haben Domänen jedoch darüber hinaus eine semantische Nebenbedeutung. Zum Beispiel könnten wir eine Domäne „Kilogramm“ verwenden, um Gewichte darzustellen, und eine weitere Domäne „Geld“ für Geldbeträge. Beide Domänen würden einen Bereich von Dezimalzahlen umfassen, aber ihre Werte wären nicht miteinander kompatibel. Diese höhere Ebene der Domänenunterstützung trifft man bei relationalen Modellen meist nicht an.

Wenn wir ein Attribut einführen, müssen wir ihm einen Namen und einen Wertebereich zuweisen. Von da an muss jeder Wert des Attributes zu der Domäne passen. Domänen können sehr allgemein gewählt sein, zum Beispiel „alle positiven ganzen Zahlen zwischen 00000 und 99999“ oder „alle Zeichenketten mit höchstens 20 Buchstaben“; sie können aber auch sehr speziell sein, zum Beispiel „eines aus VERKAUF, BUCHHALTUNG, FORSCHUNG“. Die meisten relationalen Datenbankprodukte stellen allgemeine Domänen in der Form elementarer Datentypen zur Verfügung. Nur wenige erlauben den Datenbankbenutzern mehr, als sehr rudimentäre eigene Domänen zu definieren.

Eine wichtige Eigenschaft aller relationalen Domänen ist Primitivität; dies bedeutet, dass Domänen nur aus einzelnen Werten bestehen können. Jedes Tupel in ANG besitzt genau einen Wert für jedes der Attribute ANGNR, ANGNAME, ABTNAME und STUFE. Mehrwertige Domänen gibt es nicht. Finden wir eine Entität mit einem mehrwertigen Attribut, so müssen wir weitere Relationen erzeugen, um dieses Attribut relational abzubilden.

Als Beispiel stellen wir uns vor, wir wollten die Kenntnisse jedes Angestellten in einer Datenbank ablegen. Dabei kann ein Angestellter mehrere Kenntnisse haben. Eine Darstellung wie in Tabelle 3.3 wäre nicht erlaubt.

ANGNR	ANGNAME	ABTNAME	STUFE	QUALIFIKATION
1	F JONES	VERKAUF	6	(DEUTSCH)
2	P SMITH	BUCHHALTUNG	6	(SCHREIBMASCHINE, STENO, FRANZÖSISCH)
3	K CHAN	VERKAUF	4	(DEUTSCH, FRANZÖSISCH)
4	J PETERS	VERKAUF	5	(FRANZÖSISCH, SCHREIBMASCHINE)
9	S ABDUL	BUCHHALTUNG	3	(FRANZÖSISCH, DEUTSCH, COBOL)
5	J LEWIS	FORSCHUNG	5	(KLAVIER)

Tabelle 3.3: Eine verbotene (nicht atomare) Domäne

Hier haben wir eine Domäne QUALIFIKATION eingeführt, die *nicht atomar* ist. Damit ist gemeint, dass Werte vorkommen können, die nicht *atomar*, also unteilbar, sind. In relationalen Datenbanken müssen alle Domänen atomar sein. In dieser Situation sind wir gezwungen, unsere Datenbank umzustrukturieren, so dass wir zwei Tabellen erhalten: neben der alten Relation ANG die neue QUALIFIKATIONEN, die wir in Tabelle 3.4 zeigen.

QUALIFIKATIONEN

ANGNR	QUALIFIKATION
1	DEUTSCH
2	SCHREIBMASCHINE
2	STENO
2	FRANZÖSISCH
3	DEUTSCH
3	FRANZÖSISCH
6	FRANZÖSISCH
6	SCHREIBMASCHINE
9	FRANZÖSISCH
9	DEUTSCH
9	COBOL
5	KLAVIER

Tabelle 3.4: Eine Relation mit atomaren Werten

Die neue Tabelle QUALIFIKATIONEN verwendet nur atomare Werte: Jede Zeile beschreibt eine Qualifikation eines Angestellten. Kennt ein Angestellter sich mit drei Dingen aus, so gibt es dafür drei Tupel in der Tabelle QUALIFIKATIONEN.

3.1.3 NULL-WERTE

Angenommen wir wollten unserer Datenbank einen Angestellten hinzufügen, der keiner bestimmten Abteilung angehört (und angenommen, die Semantik unserer Datenbank gestatte dies). Wir würden dann ein neues Tupel einfügen; Tabelle 3.5 zeigt das Ergebnis.

ANG

ANGNR	ANGNAME	ABTNAME	STUFE
1	F JONES	VERKAUF	6
2	P SMITH	BUCHHALTUNG	6
3	K CHAN	VERKAUF	4
4	J PETERS	VERKAUF	5
9	S ABDUL	BUCHHALTUNG	3
5	J LEWIS	FORSCHUNG	5
10	J MAJOR		1

Tabelle 3.5: Ein Tupel mit Null-wertigem Attribut

In diesem Fall sagen wir, das Tupel von ANG mit der ANGNR 10 habe den Wert NULL für ABTNAME. Das kann bedeuten, dass der Angestellte keiner Abteilung angehört oder dass wir nicht wissen, welcher Abteilung er angehört. NULL ist ein spezieller Wert, der keinem anderen Wert gleicht (nicht einmal sich selbst). Wir können nicht behaupten, zwei Angestellte mit NULL-Einträgen für ABTNAME hätten für jenes Attribut den gleichen Wert. Wir können jedoch sagen, dass für alle anderen Angestellten ABTNAME nicht NULL ist (beachten Sie, dass wir sagen „nicht NULL“ und nicht „ungleich NULL“).

Wenn wir ein Attribut definieren, müssen wir nicht nur seine Domäne angeben, sondern auch festlegen, ob seine Domäne den Wert NULL enthält. Falls nicht, müssen alle Tupel der betroffenen Relation für dieses Attribut einen Wert haben. Offenbar sollten solche Attribute, die einem Tupel seine eindeutige Identität verleihen, nicht null sein dürfen; ein Beispiel dafür ist ANGNR in der Relation ANG. Wir werden hierauf in Abschnitt 3.2.2 über Entitätenintegrität noch näher eingehen. Ob andere Attribute NULL-Werte annehmen dürfen, hängt von den semantischen Anforderungen der Datenbank ab.

3.1.4 BASISRELATIONEN UND SICHTEN

Bis jetzt haben wir nur Beispiele einer besonderen Art von Relation gesehen, die man auch *Basisrelation* nennt. Basisrelationen bilden die unterste Ebene der Datenrepräsentation, die dem Benutzer relationaler Datenbanken zugänglich ist. Alle Daten in relationalen Datenbanken werden letztendlich in Basisrelationen gespeichert. Die Daten können jedoch auch über *Sichten* abgefragt und verändert werden. Eine Sicht ist eine logische Relation, die ihre Daten direkt oder indirekt aus Basisrelationen bezieht.

Eine Sicht kann einfach eine Teilmenge einer Basisrelation sein. Tabelle 3.6 zeigt eine Sicht mit dem Namen VERKAUF, die eine Teilmenge der Relation ANG ist.

VERKAUF

ANGNR	ANGNAME	ABTNAME	STUFE
1	F JONES	VERKAUF	6
3	K CHAN	VERKAUF	4
6	J PETERS	VERKAUF	5

Tabelle 3.6: Eine einfache Sicht

In dieser Sicht haben wir eine Relation aufgestellt, die alle ANGS enthält, deren ABTNAME gleich VERKAUF ist. Offenbar gibt es hier ein gewisses Maß an Redundanz. Dass alle Tupel den gleichen Wert für ABTNAME haben, brauchen wir nicht zu zeigen. Sichten können auch auf Teilmengen von Attributen aufsetzen (Tabelle 3.7).

VERKAUF_1

ANGNR	ANGNAME	STUFE
1	F JONES	6
3	K CHAN	4
6	J PETERS	5

Tabelle 3.7: Eine etwas komplexere Sicht

Sichten können nicht nur Teilmengen von Tupeln und Attributen zeigen, sondern auch Daten aus mehreren Tabellen zusammenfassen. Tabelle 3.8 zeigt eine Sicht, die das Attribut ANGNAME aus ANG mit denjenigen Tupeln aus QUALIFIKATIONEN verbindet, die zu deutsch sprechenden Angestellten gehören.

DEUTSCHSPRECHEND

ANGNR	ANGNAME
1	F JONES
3	K CHAN
6	J PETERS

Tabelle 3.8: Eine Sicht, die auf zwei Basistabellen aufbaut

Nun können wir die Sichten VERKAUF_1 und DEUTSCHSPRECHEND kombinieren, um eine Sicht wie die in Tabelle 3.9 zu erhalten, die alle in der Abteilung VERKAUF beschäftigten Angestellten zeigt, welche deutsch sprechen.

DEUTSCHSPRECHEND_VERKAUF

ANGNR	ANGNAME
1	F JONES
3	K CHAN

Tabelle 3.9: Eine Sicht, die auf zwei Sichten aufbaut

Bis zu einem gewissen Grad kann man Sichten in relationalen Datenbanken wie Basistabellen behandeln: Man kann sie verwenden, um Daten abzufragen und (mit Einschränkungen) auch um Daten einzufügen, zu löschen und zu ändern. Diese Einschränkungen werden in einem späteren Abschnitt behandelt. Wenn man eine Sicht anlegt, sind die Domänen schon durch die Basistabellen gegeben, in denen die Daten der Sicht liegen. Dies betrifft auch Regeln bezüglich NULL-Werten. Für die Definition einer Sicht muss man die Sichten und/oder Basistabellen angeben, die an der Sicht beteiligt sind. Wird eine Basistabelle aus einer relationalen Datenbank entfernt, so müssen alle Sichten, die diese Tabelle brauchen, aufhören zu existieren – egal, ob sie sie ganz oder teilweise, direkt oder indirekt brauchen.

3.2 Relationale Datenintegrität

3.2.1 Schlüssel

Datenintegrität in relationalen Datenbanken basiert auf dem Konzept des Schlüssels. Es gibt drei Arten von Schlüsseln in relationalen Datenbanken: Kandidatenschlüssel, Primärschlüssel und Fremdschlüssel.

Ein Kandidatenschlüssel ist ein Attribut oder eine Menge von Attributen, die jedes Tupel einer Relation eindeutig identifizieren. Zum Beispiel ist ANGNR ganz offensichtlich ein Schlüssel für die Relation ANG. Jede Zeile in ANG hat einen anderen Wert für ANGNR. Gelegentlich kann man Attribute zusammenfassen, um Tupel zu identifizieren. In der QUALIFIKATIONEN-Tabelle, sind weder die Werte für ANGNR noch die für QUALIFIKATION für sich genommen einzigartig. Da jedoch kein Angestellter die gleiche Qualifikation zweimal haben kann, wird jede Zeile eine einzigartige Kombination von ANGNR und QUALIFIKATION enthalten. Dies nennt man einen zusammengesetzten Kandidatenschlüssel.

Ein Primärschlüssel ist ein Spezialfall eines Kandidatenschlüssels. Es kann für eine Relation mehr als einen Kandidatenschlüssel geben. Zum Beispiel könnten wir in unserer Tabelle ANG ein Attribut STEUERNUMMER einführen, um die Angestellten mit anderswo verwendeten eindeutigen Steuernummern zu verknüpfen. Falls die STEUERNUMMER für jeden Angestellten eindeutig ist, hätten wir hier zwei Kandidatenschlüssel. In einer solchen Situation müssen wir einen der beiden als Primärschlüssel auswählen. Wenn man ein Attribut oder eine Menge von Attributen als Primärschlüssel wählt, ergeben sich daraus einige Konsequenzen für die betroffenen Attribute (siehe unten). Jede Tabelle kann beliebig viele Kandidatenschlüssel haben, aber es muss genau einen Primärschlüssel geben. Falls es nur einen Kandidatenschlüssel gibt, ist dieser automatisch der Primärschlüssel.

Ein Fremdschlüssel ist ein Attribut (oder eine Menge von Attributen), das in mehr als einer Tabelle vorkommt und in einer dieser Tabellen den Primärschlüssel bildet. ANGNR kommt sowohl in der Tabelle ANG als auch in der Tabelle QUALIFIKATIONEN vor; da es der Primärschlüssel für ANG ist, tritt es folglich in QUALIFIKATIONEN als Fremdschlüssel auf. Fremdschlüssel sind in relationalen Datenbanken außerordentlich wichtig. Sie sind die wichtigste Methode, um Daten in verschiedenen Tabellen zu verknüpfen. Wir können zwischen den Zeilen von ANG und den entsprechenden Zeilen von QUALIFIKATIONEN eine Verbindung herstellen, indem wir den Fremdschlüssel ANGNR in QUALIFIKATIONEN benutzen. Man spricht dann davon, dass zwischen den beiden Tabellen eine Beziehung aufgebaut wird. Damit solche Beziehungen gültig sind, gibt es eigene Regeln für den Gebrauch von Fremdschlüsseln (siehe Abschnitt 3.2.3 unten).

3.2.2 Entitätenintegrität

Eine Entität ist definiert als ein Gegenstand, der zu unabhängiger Existenz in der Lage ist. Eine Entitätenmenge ist eine Menge von Gegenständen mit den gleichen Eigenschaften. Damit die Gegenstände in einer Entitätenmenge eine unabhängige Existenz besitzen können, muss es möglich sein, sie zu unterscheiden. In relationa-

len Datenbanken setzt man Basisrelationen ein, um Entitätenmengen zu modellieren. Eine Basisrelation besteht aus einer Menge von Tupeln, die alle die gleichen Attribute haben. Damit die Tupel einer Relation zu einer echten Menge unterschiedlicher Gegenstände gehören und damit so eine Menge einer Entitätenmenge entspricht, muss jedes Tupel eine unterscheidbare Identität besitzen.

Aus diesem Grund braucht jede Basisrelation in relationalen Datenbanken einen Primärschlüssel. Darüber hinaus müssen wir sicherstellen, dass jede Zeile einen eigenen Wert für ihren Primärschlüssel hat, denn sonst hat sie keine eindeutige Identität. Daher lassen wir nicht zu, dass ein Attribut, das Teil des Primärschlüssels ist, NULL-Werte annimmt. In der Tabelle ANG müssen deshalb alle Zeilen einen Wert für ANGNR enthalten. In der Tabelle QUALIFIKATIONEN, die einen zusammengesetzten Primärschlüssel hat, muss jede Zeile sowohl einen Wert für ANGNR als auch einen für QUALIFIKATION enthalten. Wenn wir für eines der beiden Attribute NULL-Werte zuließen, könnte es vorkommen, dass zwei Zeilen die gleichen Werte für QUALIFIKATION enthielten und keine ANGNR; solche Zeilen könnte man nicht unterscheiden. Entitätenintegrität verbietet deswegen NULL-Werte für Attribute, die Teile des Primärschlüssels sind.

Entitätenintegrität betrifft nur den Primärschlüssel. Hat eine Tabelle noch weitere Kandidatenschlüssel, so dürfen die nichtprimären Kandidatenschlüssel ganz oder in Teilen NULL-Werte annehmen. Je nach den semantischen Anforderungen des Systems kann der Datenbankentwickler sich jedoch entscheiden, NULL-Werte auch für solche Schlüssel oder sogar für Nichtschlüsselattribute zu verbieten. Solche Entscheidungen sind dann jedoch spezielle Eigenschaften dieses konkreten Systems, während Entitätenintegrität eine theoretische Forderung an alle relationalen Systeme ist. („Theoretisch“ nennen wir sie deshalb, weil es viele relationale Datenbankprodukte gibt, die Tabellen ohne Primärschlüssel zulassen.)

Entitätenintegrität bedeutet, dass Entitäten unterscheidbar sein müssen. Insbesondere müssen sie sich in den Werten des Primärschlüsselattributes unterscheiden; daraus folgt, dass kein Attribut, das Teil des Primärschlüssels ist, den Wert NULL annehmen darf.

3.2.3 Referentielle Integrität

Referentielle Integrität betrifft den Gebrauch von Fremdschlüsseln. In unserer Tabelle QUALIFIKATIONEN kommt ANGNR als Fremdschlüssel für ANG vor. Wie stellen wir sicher, dass in dieser Tabelle nur gültige Verweise auf ANG stehen? Indem wir die Regel der referentiellen Integrität anwenden.

Referentielle Integrität bedeutet, dass jeder von NULL abweichende Wert in einem Fremdschlüsselattribut auch in der Relation vorkommen muss, in der dieses Attribut als Primärschlüssel auftritt. Wir können also keine ANGNR in QUALIFIKATIONEN haben, die nicht auch in ANG vorkäme. Somit sind alle Verweise von QUALIFIKATIONEN auf ANG gültig. Wir können nur für solche Angestellten Qualifikationen eintragen, die auch tatsächlich existieren.

Referentielle Integrität bringt hinsichtlich der Änderung oder des Löschens von Zeilen in Tabellen besondere Probleme mit sich. Was passiert, wenn wir einen Wert von ANGNR in ANG ändern und damit alle Verweise auf den alten Wert ungültig machen? Drei Verfahren sind möglich:

1. Einschränkung. Dieses Verfahren unterbindet jegliche Änderungen eines Primärschlüssels, solange Fremdschlüsselreferenzen darauf existieren. Es wäre dann verboten, ANGNR 1 aus der Tabelle ANG zu entfernen oder den Wert von ANGNR 1 in ANGNR 7 zu ändern. Wir könnten nur die ANGNR-Werte von solchen Angestellten ändern oder löschen, für die keine Einträge in QUALIFIKATIONEN existierten.
2. Kaskadieren. Hierbei setzt man die Änderung an der ursprünglichen Zeile auf alle Zeilen in allen referenzierenden Tabellen fort. Wollten wir ANGNR 1 aus ANG löschen, so müssten wir damit auch alle Zeilen mit ANGNR 1 aus QUALIFIKATIONEN löschen. Wenn wir einen Wert von ANGNR in ANG ändern, müssen wir auch die entsprechenden Werte von ANGNR in QUALIFIKATIONEN ändern.
3. Auf NULL setzen. Hiermit gestattet man die Änderung oder Löschung in der ursprünglichen Tabelle. Um die referentielle Integrität zu erhalten, werden danach alle entsprechenden Werte von Fremdschlüsseln auf NULL gesetzt. Im Beispiel oben würde dies dazu führen, dass Zeilen aus der Tabelle QUALIFIKATIONEN gelöscht werden, weil ANGNR Teil des Primärschlüssels für QUALIFIKATIONEN ist. Wir erhalten ein besseres Beispiel, wenn wir eine Tabelle ABT wie folgt einführen:

   ABT

ABTNAME	LEITERANGNR	BUDGET
VERKAUF	3	200.000
BUCHHALTUNG	2	5.000.000
FORSCHUNG	5	100

   In dieser Tabelle ist ABTNAME der Primärschlüssel und somit in ANG ein Fremdschlüssel. Wenn wir also den Namen der Verkaufsabteilung ändern, würden bei der Auf-NULL-setzen-Strategie alle Zeilen aus ANG, in denen VERKAUF als ABTNAME eingetragen ist, einen NULL-Wert erhalten. Die Entitätenintegrität dieser Zeilen wäre davon unberührt, da ABTNAME nicht Teil des Primärschlüssels ist.

Ihnen ist vielleicht das Attribut LEITERANGNR in der Tabelle ABT aufgefallen. Dieses ist ein Fremdschlüssel, der auf die Tabelle ANG verweist und die ANGNR des Abteilungsleiters enthält. Fremdschlüssel müssen nicht in allen Tabellen, in denen sie vorkommen, den gleichen Namen haben. Referentielle Integrität gilt trotzdem. Wir müssen entscheiden, was mit Zeilen aus ABT passiert, wenn ANGNR-Werte in ANG geändert oder gelöscht werden. Was passiert zum Beispiel, wenn wir versuchen, ANGNR 2 zu löschen – verbieten wir die Operation, weil der Wert noch in ABT vorkommt (Strategie „Einschränkung“) oder entfernen wir die entsprechende Zeile aus ABT (Strategie „Kaskadieren“) oder setzen wir den Wert von LEITERANGNR in der Zeile für BUCHHALTUNG auf NULL? Diese Entscheidung müssen wir für jeden Fremdschlüssel in der Datenbank treffen.

Beachten Sie, dass referentielle Integrität nur in eine Richtung wirkt. Wir können die Werte von ANGNR in der Tabelle QUALIFIKATIONEN, die Werte von ABTNAME in der Tabelle ABT oder die Werte von LEITERANGNR in der Tabelle ABT nach Belieben ändern, so lange sie weiterhin eine Referenz auf den Wert eines tatsächlich existierenden Primärschlüssels enthalten. Wenn eine solche Referenz nicht mehr hergestellt werden kann, muss der entsprechende Wert des Fremdschlüssels auf NULL gesetzt werden.

Zusammenfassend ist referentielle Integrität die Forderung, dass jeder Wert eines Fremdschlüssels eine Referenz auf einen Wert eines Primärschlüssels sein muss, der wirklich existiert. Falls kein entsprechender Wert eines Primärschlüssels existiert, muss der Wert des Fremdschlüssels auf NULL gesetzt werden.

3.3 RELATIONALE DATENMANIPULATION

Einer der großen Vorteile relationaler Datenbanken besteht darin, dass Daten aus beliebigen Ansammlungen relationaler Tabellen mittels Kombinationen von nur acht intuitiv einfachen relationalen Operationen ausgelesen werden können. Diese Operationen bilden etwas, das man die *relationale Algebra* nennt. Die relationale Algebra enthält fünf Basisoperationen, RESTRICT, PROJECT, TIMES, UNION und MINUS. Daneben gibt es drei abgeleitete relationale Operationen JOIN, INTERSECT und DIVIDE. Sie sind insofern abgeleitet, als man sie aus den Basisoperationen aufbauen kann; es ist jedoch bequem, sie wie Basisoperationen zu behandeln. Insbesondere der Operator JOIN bezeichnet eine der am meisten eingesetzten relationalen Operationen und verdient erhebliche Aufmerksamkeit.[1]

Alle relationalen Systeme setzen die Algebra ein, um Daten aus Tabellen abzufragen. Es gibt jedoch nur sehr wenige Systeme, die die Algebra direkt unterstützen. Meist wird eine eigene Schnittstelle, häufig in Form der Sprache SQL, zur Verfügung gestellt. SQL werden wir in Kapitel 5 behandeln; es ist eine deklarative Datenbanksprache, die Benutzern erlaubt, Ausdrücke zu schreiben, die Datenmengen „beschreiben“. Die Aufgabe des SQL-Interpretierers besteht darin, SQL-Ausdrücke in Folgen algebraischer Operationen umzuformen, die die verlangte Datenmenge liefern.

Ein wichtiges Prinzip der relationalen Algebra besagt, dass alle ihre Operationen ausschließlich auf der Ebene der Relationen arbeiten. Als Argument akzeptieren sie nur eine Relation oder eine Menge von Relationen, und sie geben immer nur eine einzige Relation zurück, welche die durch die Operation spezifizierten Bedingungen erfüllt. Definitionsgemäß muss jede als Argument übergebene Relation Teil der Datenbank sein, sei es als Basisrelation oder als Sicht. Das Ergebnis einer relationalen Operation ist eine temporäre Relation, die durch Anwendung der Operation auf die Argumentrelationen entsteht.

Wir beschreiben nun die Operationen, aus denen die relationale Algebra aufgebaut ist.

[1] Anm. d. Übers.: Üblicherweise repräsentiert man diese Operationen mit folgenden Symbolen: σ für RESTRICT, π für PROJECT, $\times$ für TIMES, $\cup$ für UNION, für MINUS, \Join für JOIN, $\cap$ für INTERSECT und $\div$ für DIVIDE.