2 Das Relationale Modell

In diesem Kapitel wird das auf Arbeiten von E. F. Codd 1970 zurückgehende Relationale Modell vorgestellt. Es verzichtet auf die Verwendung von Pointern zur Darstellung von Beziehungen. Daten werden hier ausschließlich über inhaltliche Angaben aufeinander bezogen. Anders als im Hierarchischen Modell oder im Netzwerk-Modell stellen die relationalen Operationen weniger die Verarbeitung einzelner Objekte (Entities, Records) in den Vordergrund. Sie zielen vielmehr auf die Verarbeitung der Tabellen als Ganzes, sie sind mengenorientiert statt satzorientiert. Sowohl Hierarchisches als auch Netzwerk-Modell haben heute fast nur noch historische Bedeutung. Beginnend in den 70er Jahren des letzten Jahrhunderts sind beide Modelle die erfolgreichsten der Datenbankmodelle der ersten Generation.

2.1 Relationen

Im Relationalen Modell werden die Objekte einer Klasse in Form von Tabellen (Relationen) dargestellt. Die Beschreibung eines Objekts taucht dabei als eine Tabellen-Zeile auf. Aus faktischen und auch aus formalen Gründen spielt die Reihenfolge der Zeilen in den Tabellen keine Rolle.

Struktur

Die Struktur einer Tabelle wird durch die Spalten-Überschrift festgelegt. Die Eintragungen in den Spalten interpretieren wir dabei als Werte zu Attributen. Eine wichtige Regel des Relationalen Modells verlangt, daß die einzelnen Attributwerte elementar sind; mehrfache Ausprägungen wie z.B. zu Vornamen (sog. Wiederholungsgruppen) oder Zusammenfassungen von Attributen zu Gruppen wie z.B. von PLZ, Ort und Straße zu Adresse sind nicht erlaubt.

Abteilung

Abteilung	Bezeichnung
A1	Süßwaren
A2	Lebensmittel
A3	Bekleidung (Damen)
A4	Bekleidung (Herren)
...	

Mitarbeiter

Pers-Nr	Name	Vorname	Abteilung
12	K.	Wastl	A2
15	B.	Erna	A1
11	V.	Hugo	A1
18	L.	Liesel	A2
23	M.	Fredi	A4
21	S.	August	A2
26	Q.	Seppl	A1
...			

Artikel	
Artikel-Nr	Bezeichnung
1	Schokolade
12	Pralinen
3	Milch
5	Nudeln
9	Osterhasen
25	Kaugummi
...	

Angebot	
Artikel-Nr	Abteilung
1	A1
1	A2
3	A2
5	A2
25	A2
12	A1
9	A1
...	

Beziehungen

Obige Abbildung verdeutlicht mit der Relation Angebot, wie im Relationalen Modell Beziehungen zu dokumentieren sind. Welches Objekt mit wem in einer Beziehung steht, wird dort explizit in Tabellenform aufgezählt. Stellvertretend für die Objekte werden dabei identifizierende Merkmale (Schlüssel) verwendet. Die Relation Angebot beschreibt ausschließlich die Beziehungen zwischen Artikeln und Abteilungen. Als Erweiterung von Angebot kann zum Beispiel in der Relation Umsatz (Artikel-Nr, Abteilung, J-UMS) ein Attribut J-UMS verwendet werden, das pro Artikel den Jahresumsatz in den Abteilungen festhält. Beziehungen brauchen jedoch nicht in jedem Fall explizit in einer eigenen Relation dargestellt zu werden: Die Zugehörigkeit der Mitarbeiter zu ihren Abteilungen finden wir in Mitarbeiter aufgrund der Angabe zu Abteilung.

Bei der Vorstellung der Ausdrucksmittel bzw. der zentralen Grundkonzepte des Relationalen Modells werden wir im mathematischen Sinne präzise Schreibweisen benutzen, welche es uns dann ermöglichen, die Grundzüge der Theorie des Relationalen Datenmodells darzustellen. Unter diesen Gesichtspunkten entspricht eine Tabellenzeile einem Tupel. Ein Tupel ist eine endliche, geordnete Menge von Werten (Komponenten) mit einer Darstellung

$$(x_1, \ldots, x_n)$$

In dem hier betrachteten Modell ist jeder Komponente eines Tupels ein Attribut zugeordnet. Damit wird dem betreffenden Wert eine bestimmte Interpretation unterlegt. Das Tupel besteht dann aus Attributwerten (oder Ausprägungen zu den Attributen), deren Bedeutung wir stets miteinbeziehen. So können die Werte zu

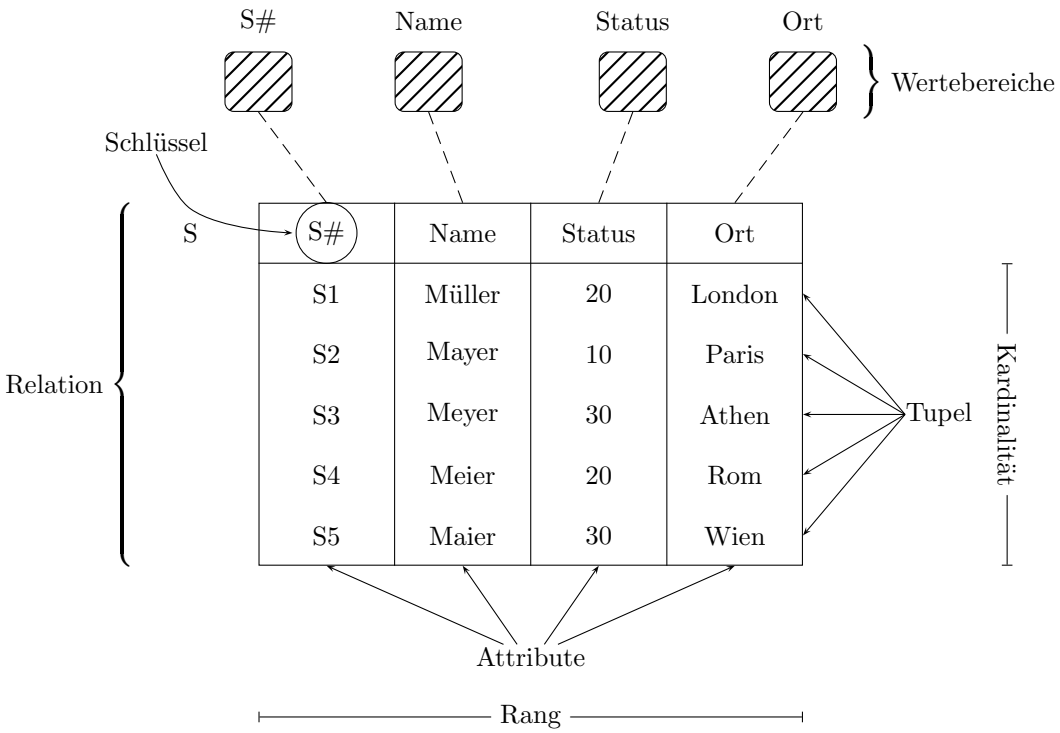
(Pers-Nr, Name, Vorname, Abteilung)

zur Beschreibung eines Mitarbeiters dienen.

Ein Attribut und ein entsprechender Wert liefern eine elementare Aussage über einen Gegenstand oder Sachverhalt. Ein Tupel kann dann als komprimierte Darstellung

zusammengehörender, elementarer Aussagen aufgefaßt werden. Z.B. beschreibt mit den obigen Attributen das Tupel (12, K., Wastl, A2) den Mitarbeiter Wastl K. mit der Personal-Nr. 12 aus der Abteilung A2.

2.1.1 Formalisierung



Tabellen können formal mit mathematischen Mitteln interpretiert werden. Dazu definieren wir zunächst den grundlegenden Begriff der Relation:

Definition 2.1 *Es seien zwei beliebige Mengen M_1 und M_2 gegeben.*

a) Das (kartesische) Produkt zweier Mengen M_1 und M_2 ist definiert durch

$$M_1 \times M_2 = \{(a, b) \mid a \in M_1 \wedge b \in M_2\}$$

b) Eine Relation R der Mengen M_1 und M_2 ist eine Teilmenge des Produkts:

$$R \subseteq M_1 \times M_2$$

Eine Relation setzt in den Tupeln (a, b) gewisse Elemente $a \in M_1$ und $b \in M_2$ in eine Beziehung zueinander.

Entsprechend wird das Produkt zwischen mehreren Mengen $M_1, \ldots, M_n$ definiert als die Menge aller n-Tupel $(x_1, \ldots, x_n)$ mit $x_i \in M_i$, $i = 1, 2, \ldots, n$. Eine Teilmenge aus $M_1 \times M_2 \times \cdots \times M_n$ wird dann als eine *n-stellige Relation* bezeichnet. Wir betrachten zwei Tupel $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$ als *gleich*, wenn sie in allen Komponenten übereinstimmen: $x_i = y_i$, für $i = 1, 2, \ldots, n$.

2.1.2 Relationen in der Informationsverarbeitung

Für unsere Zwecke stellen die Mengen $M_1, M_2, \ldots, M_n$ stets *Wertebereiche* (Domains) zu gewissen *Attributen* $A_1, A_2, \ldots, A_n$ dar. Statt M_i wählen wir deshalb die symbolischen Bezeichnungen $dom(A_i)$ oder D_i. Wir gehen grundsätzlich davon aus, daß einem Attribut stets implizit sein Wertebereich zugeordnet ist.

Mit der positionsgerechten Präsentation einer Relation erhalten wir eine Tabelle, welche aus einzelnen Zeilen besteht. Unter den Zeilen ist die Kopfzeile insofern ausgezeichnet, als sie zeitinvariante Eigenschaften enthält. Alle weiteren Zeilen zusammen beschreiben den zeitveränderlichen Inhalt der Tabelle. Die Reihenfolge der Tupel (Zeilen) ist dabei unbedeutend. Identische Tupel sollen grundsätzlich ausgeschlossen sein; sie beschreiben schließlich keine neuen Sachverhalte. Wir gehen immer davon aus, daß entweder das Datenbank-Managementsystem (DBMS) oder die Anwendung doppelte Tupel eliminiert.

Beispiel 2.2 *In einer Bibliothek möchten wir Informationen über die einzelnen Benutzer abspeichern. Es ergibt sich folgende Relation (Tabelle), die wir mit dem Namen* Benutzer *kennzeichnen:*

Ben-Nr	*Vorname*	*Name*	*PLZ*	*Wohnort*	*Alter*	*Geschlecht*
213	*Rosine*	*Sch.*	*54391*	*Salzburg*	*31*	*w*
215	*Abacus*	*B.*	*61142*	*Frau-Nauses*	*21*	*m*
220	*Rabiata*	*M.*	*69483*	*Corsika*	*44*	*w*
224	*Antiqua*	*W.*	*85634*	*St. Helena*	*32*	*w*
232	*Frustus*	*L.*	*64315*	*Machtlos*	*44*	*m*
240	*Mecki*	*K.*	*61147*	*Frau-Nauses*	*35*	*m*

Diese Relation kann als Teilmenge eines geeigneten Produkts interpretiert werden, das allerdings nicht eindeutig vorgegeben ist. Z.B. kommen als Wertebereiche für Ben-Nr die Menge $\{213, 215, 220, 224, 232, 240\}$*, für Vorname* {*Rosine, Abacus, Rabiata, Antiqua, Frustus, Mecki*}*, für Geschlecht* $\{m, w\}$ *etc. in Frage. Anstelle unserer Mengen dürfen wir aber auch für Ben-Nr die Werte* $1, \ldots, 100000$*, für Vorname die Menge aller möglichen Vornamen etc. verwenden. Das Produkt gibt lediglich einen strukturellen Rahmen ab, in dem sich die Relation bewegen muß.*

2.1.3 Relationen-Struktur (Relationenschema)

Eine Relation wird symbolisch in der Form

$$R(A_1, A_2, \ldots, A_n)$$

wiedergegeben, wobei A_i, $i = 1, \ldots, n$, die Attribute und R den Namen der Relation meint. Strenggenommen wird durch $(A_1, A_2, \ldots, A_n)$ die Bauart der Tupel oder eine *Relationen-Struktur* (*Relationenschema*) vermittelt, zu der unterschiedliche Konkretisierungen (Tupel-Mengen, Relationen) existieren können.

Eine Sammlung von mehreren Relationenschemata heißt auch *Relationales DB-Schema*.

2.1.4 Äquivalenz-Betrachtungen

Mit der Relationen-Struktur wird eine Anordnung der Attribute vorgegeben. Allerdings spielt die Reihenfolge für den Aussagegehalt der Relation keine Rolle. Manche Zugänge verzichten deshalb generell auf eine Attribut-Anordnung. Der Bezug zum mathematischen Tupelbegriff geht dann allerdings verloren. Wir halten an der konventionellen Interpretation fest und sprechen stattdessen von zwei äquivalenten Relationen-Strukturen, falls beide sich nur in der Attribut-Anordnung unterscheiden. Die folgende Betrachtung relativiert das Festhalten an der Anordnung.

Mit den Zahlen $i(1), i(2), \ldots, i(n)$ sei eine Permutation der Zahlen $1, 2, \ldots, n$ gegeben. $(A_1, A_2, \ldots, A_n)$ und $(A_{i(1)}, A_{i(2)}, \ldots, A_{i(n)})$ seien zwei äquivalente Relationen-Strukturen. Dann können die Tupel $t \in R_1(A_1, A_2, \ldots, A_n)$ nach der Vorschrift

$$t = (x_1, \ldots, x_n) \mapsto t' = (x_{i(1)}, x_{i(2)}, \ldots, x_{i(n)})$$

eindeutig in Tupel t' der Relation $R_2(A_{i(1)}, A_{i(2)}, \ldots, A_{i(n)})$ abgebildet werden. R_1 und R_2 bezeichnen wir als *äquivalent* oder auch als gleich bis auf Äquivalenz, in Zeichen: $R_1 \sim R_2$.

Die Äquivalenz-Betrachtung gestattet uns, stets eine beliebige (aber feste) Reihenfolge der Attribute anzunehmen.

2.2 Datenmanipulation im Relationalen Modell

In diesem Abschnitt beschreiben wir zunächst die grundlegenden Operationen auf den Relationen einer (Relationalen) Datenbank und ihre Rechenregeln. Diese Operationen werden es uns insbesondere ermöglichen, Anfragen an eine solche Datenbank zu stellen, wobei wesentlich ist, daß die Verarbeitung mengenorientiert erfolgt.

Dies bedeutet, daß relationale Operationen aus einer oder mehreren Tupelmengen (Relationen) eine neue (abgeleitete) Tupelmenge (Relation) erzeugen. Der Benutzer „navigiert“ also relationenweise und nicht satzweise.

Ein Benutzer stellt Anfragen an eine Relationale Datenbank im allgemeinen nur unter Verwendung von Schema-Informationen, insbesondere der vereinbarten Relationennamen. Eine Anfrage wird daher in Form eines formalen Ausdrucks (der Relationen-Algebra bzw. des Relationen-Kalküls) an das System übermittelt, welches ihn – insbesondere unter Ersetzung aller Namen durch ihren aktuellen „Inhalt“ – auswertet. Diese Unterscheidung entspricht dem Übergang von der konzeptionellen auf die interne Ebene, wobei der Benutzer jedoch keine Annahmen über deren Organisation zu kennen oder zu berücksichtigen braucht.

2.2.1 Klassifikation von Anfragesprachen

- Deskriptiv: Die auszuwählenden Objekte werden als Gesamtheit beschrieben (Was?); Relationen-Kalkül.
- Prozedural: Das Ergebnis wird durch eine Folge von Operationen konstruiert (Wie?); Relationen-Algebra.

2.2.2 Relationen-Algebra

Alle relationalen Operationen erzeugen wieder Relationen, wodurch ihre Schachtelung ermöglicht wird. Um den Aspekt der Abgeschlossenheit der Relationen-Welt mit ihren Operationen hervorzuheben, wird dieses Gebiet auch als Relationale Algebra bezeichnet.

Im folgenden seien endliche Relationen $R, S, \ldots$ mit den zugehörigen Schemata $R(A_1, \ldots, A_n)$, $S(B_1, \ldots, B_m), \ldots$ gegeben.

Grundoperationen

Vereinigung

Voraussetzung für die Vereinigung von R und S ist deren Verträglichkeit, d.h., $Rang(R) = Rang(S)$ und für alle $i = 1, 2, \ldots, n$ gilt $dom(A_i) = dom(B_i)$.

$R \cup S := R'(A_1, \ldots, A_n)$ ergibt sich dann aus der Mengenvereinigung von R und S:

$$R' = \{t \mid t \in R \vee t \in S\}$$

Differenz

Analog Vereinigung:

$R \setminus S := R'(A_1, \ldots, A_n)$ mit

$$R' = \{t \mid t \in R \wedge t \notin S\}$$

Kartesisches Produkt

Definition 2.3 *Zu $r = (r_1, \ldots, r_n) \in R$ und $s = (s_1, \ldots, s_m) \in S$ sei*

$$r \cdot s := (r_1, \ldots, r_n, s_1, \ldots, s_m)$$

Das kartesische Produkt $R \times S := R'(A_1, \ldots, A_n, B_1, \ldots, B_m)$ wird dann festgelegt durch

$$R' = \{r \cdot s \mid r \in R \wedge s \in S\}$$

Zu beachten ist ferner, daß bei Namensgleichheit von Attributen eine Umbenennung erfolgen muß.

Beispiel 2.4

R	A	B
	a	b
	e	f

S	C
	c
	d

$R \times S = R'$	A	B	C
	a	b	c
	a	b	d
	e	f	c
	e	f	d

Projektion

Definition 2.5 *Es sei $\bar{A} = (A_{i(1)}, A_{i(2)}, \ldots, A_{i(k)})$, $1 \leq i(1), \ldots, i(k) \leq Rang(R)$. Zu $r = (r_1, \ldots, r_n) \in R$ sei $\pi_{i(1),\ldots,i(k)}(r) = (r_{i(1)}, \ldots, r_{i(k)})$ festgelegt. Die Projektion $\pi_{\bar{A}}(R) := R'(A_{i(1)}, A_{i(2)}, \ldots, A_{i(k)})$ wird festgelegt durch*

$$R' = \{\pi_{i(1),\ldots,i(k)}(r) \mid r \in R\}$$

Beispiel 2.6

R	A	B	C	D
	a	b	c	d
	a	e	c	f
	g	b	h	f
	a	i	h	j

$\pi_{C,A}(R) = R'$	C	A
	c	a
	h	g
	h	a

Selektion

Definition 2.7 *Es sei φ eine atomare Formel $\Theta(u_1, \ldots, u_n)$, wobei $u_1, \ldots, u_n$ Konstanten, Attributnamen oder Terme aus Konstanten, Attributnamen und Datenoperationen (z.B. A+100.1) sind und Θ einen zum jeweiligen Wertebereich passenden logischen Operator ($\leq, <, =, \neq, >, \geq$) bezeichnet.*

Die Selektion $\sigma_\varphi(R) := R'(A_1, A_2, \ldots, A_n)$ wird festgelegt durch:

$R' = \{r \mid r \in R \wedge \varphi$ ergibt nach Ersetzung der Attributnamen A_i durch Komponenten r_i wahr $\}$.

Beispiel 2.8

R	A	B	C	D
	a	b	c	d
	a	e	c	f
	g	b	h	f
	a	i	h	j

$\sigma_{A='a'}(R) = R'$	A	B	C	D
	a	b	c	d
	a	e	c	f
	a	i	h	j

Umbenennung

Definition 2.9 *Es sei B ein Attributname, der nicht in $\{A_1, \ldots, A_n\}$ vorkommt. Dann ist $\delta_{A_i \leftarrow B}(R) := R'(A_1, \ldots, A_{i-1}, B, A_{i+1}, \ldots, A_n)$ mit $R' = R$.*

Abgeleitete Operationen

Durchschnitt

Ableitung: $R \cap S = R \setminus (R \setminus S)$

Verallgemeinerte Selektion

Die Verallgemeinerung der Selektion besteht darin, daß φ eine logische Verknüpfung von atomaren Formeln mittels $\wedge, \vee$ oder $\neg$ sein darf.

Ableitung: Komposition von Selektion, Vereinigung, Durchschnitt und Differenz.

Verbund

Definition 2.10 *Es sei Θ ein passender Vergleichsoperator (falls Θ die Gleichheit bezeichnet, sprechen wir auch von Equiverbund).*

Der Verbund $R \bowtie_{A_i \Theta B_j} S := R'(A_1, \ldots, A_n, B_1, \ldots, B_m)$ *wird festgelegt durch*

$$R' = \{r \cdot s \mid r \in R \wedge s \in S \wedge r_i \Theta s_j\}$$

Ableitung: $R \bowtie_{A_i \Theta B_j} S = \sigma_{R.A_i \Theta S.B_j}(R \times S)$

Beispiel 2.11

R	A	B	C
	1	2	3
	4	5	6
	7	8	9

S	D	E
	3	1
	6	2

$R \bowtie_{B<D} S = R'$	A	B	C	D	E
	1	2	3	3	1
	1	2	3	6	2
	4	5	6	6	2

Natürlicher Verbund

Definition 2.12 *Es seien $C_1, \ldots, C_k$ alle gemeinsamen Namen in den Schemata R und S, also $R(A_1, \ldots, A_{n-k}, C_1, \ldots, C_k)$ und $S(B_1, \ldots, B_{m-k}, C_1, \ldots, C_k)$. Der natürliche Verbund wird dann durch folgende Ableitung festgelegt:*

$R \bowtie S := \bar{\pi}(\sigma_{R.C_1 = S.C_1 \wedge \cdots \wedge R.C_k = S.C_k}(R \times S))$ *mit* $\bar{\pi} = \pi_{A_1, \ldots, A_{n-k}, C_1, \ldots, C_k, B_1, \ldots, B_{m-k}}$

Beispiel 2.13

R	A	B	C
	a	b	c
	e	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	c	f
	a	d	b

$R \bowtie S = R'$	A	B	C	D
	a	b	c	d
	a	b	c	f
	e	b	c	d
	e	b	c	f
	c	a	d	b

Weitere Verbund-Operatoren

Die bisher eingeführten Verbund-Operatoren werden auch *innere* Verbunde genannt. Es gehen diejenigen Tupel verloren, die keinen Verbundpartner gefunden haben. Bei den *äußeren Verbunden* werden auch solche Tupel in die Ergebnisrelation übernommen. Die unbekannten Attributwerte werden dann mit Hilfe sogenannter Nullwerte, hier durch '–' dargestellt, gekennzeichnet. Mit K wird die konstante Relation bezeichnet, die nur ein Attribut hat. Zudem ist nur ein Tupel vorhanden, dessen

Attributwert Null ist.

K	Q
	–

Linker äußerer Verbund

Die Tupel der linken Relation bleiben in jedem Fall erhalten.

Definition 2.14 *Es seien $C_1, \ldots, C_k$ alle gemeinsamen Namen in den Schemata R und S, also $R(A_1, \ldots, A_{n-k}, C_1, \ldots, C_k)$ und $S(B_1, \ldots, B_{m-k}, C_1, \ldots, C_k)$. Der linke äußere Verbund wird durch folgende Ableitung festgelegt:*

$R \bowtie_l S := (R \bowtie S) \cup ((R \bowtie (\bar{\pi}(R) \setminus \bar{\pi}(S))) \times K)$ *mit* $\bar{\pi} = \pi_{C_1,\ldots,C_k}$.

Beispiel 2.15

R	A	B	C
	a	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	c	f
	a	d	b

$R \bowtie_l S = R'$	A	B	C	D
	a	b	c	d
	a	b	c	f
	b	b	f	–
	c	a	d	b

Rechter äußerer Verbund

Die Tupel der rechten Relation bleiben in jedem Fall erhalten.

Beispiel 2.16

R	A	B	C
	a	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	d	f
	a	d	b

$R \bowtie_r S = R'$	A	B	C	D
	a	b	c	d
	–	b	d	f
	c	a	d	b

Äußerer Verbund

Die Tupel beider Relationen bleiben in jedem Fall erhalten.

Beispiel 2.17

R	A	B	C
	a	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	d	f
	a	d	b

$R \bowtie_{rl} S = R'$	A	B	C	D
	a	b	c	d
	–	b	d	f
	b	b	f	–
	c	a	d	b

Semiverbund von R mit S

Das Ergebnis enthält alle Tupel aus R in unveränderter Form, die einen Verbundpartner in S haben.

Beispiel 2.18

R	A	B	C
	a	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	d	f
	a	d	b

$R \ltimes S = R'$	A	B	C
	a	b	c
	c	a	d

Semiverbund von S mit R

Das Ergebnis enthält alle Tupel aus S in unveränderter Form, die einen Verbundpartner in R haben.

Beispiel 2.19

R	A	B	C
	a	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	d	f
	a	d	b

$R \rtimes S = R'$	B	C	D
	b	c	d
	a	d	b

Es gilt $S \rtimes R = R \ltimes S$.

Division

Definition 2.20 *Alle Attribute von S seien auch Attribute von R: $R(A_1, \ldots, A_{n-m}, B_1, \ldots, B_m)$ und $S(B_1, \ldots, B_m)$. Die Division wird dann folgendermaßen festgelegt:*

$$R \div S := \{\pi_{A_1,\ldots,A_{n-m}}(r) \mid r \in R \wedge \forall\, s \in S :\ \pi_{A_1,\ldots,A_{n-m}}(r) \cdot s \in R\}$$

Beispiel 2.21

R	A	B	C	D
	a	b	c	d
	a	b	e	f
	g	h	c	d
	i	j	k	l

S	C	D
	c	d
	e	f

$R \div S = R'$	A	B
	a	b

Relationale Operationen lassen sich zu Termen über Relationennamen und konstanten Relationen zusammensetzen. Solche relationalen Terme definieren eine prozedurale Anfragesprache.

Gegenstand folgender Beispiele ist eine Kochrezept-Datenbank:

$$\begin{aligned}&\textsc{Vorrat}(\underline{\text{Zutat}}, \text{VMenge})\\&\textsc{Rezept}(\underline{\text{RName}}, \underline{\text{Zutat}}, \text{BMenge})\\&\textsc{Herkunft}(\underline{\text{RName}}, \text{Land})\end{aligned}$$

Beispiel 2.22

a) Gesucht sind die Zutaten und Vorratsmengen von chinesischen Rezepten.

$$\pi_{Zutat}\Big(\pi_{RName}(\sigma_{Land='China'}(\textsc{Herkunft})) \bowtie \textsc{Rezept}\Big) \bowtie_l \textsc{Vorrat}$$

b) Gesucht sind alle Rezeptnamen, die keinen Knobi, aber Rindfleisch verwenden.

$$\pi_{RName}\Big(\sigma_{Zutat='Rindfleisch'}(\textsc{Rezept})\Big) \setminus \pi_{RName}\Big(\sigma_{Zutat='Knobi'}(\textsc{Rezept})\Big)$$

c) Gesucht sind alle Zutaten im Vorrat, die in keinem Rezept verwendet werden.

$$\pi_{Zutat}(\textsc{Vorrat}) \setminus \pi_{Zutat}(\textsc{Rezept})$$

d) Gesucht sind alle vorrätigen Zutaten des Rezeptes „Pizza“, deren Menge nicht ausreicht.

$$\pi_{Zutat}\Big(\sigma_{VMenge<BMenge}\Big(\sigma_{RName='Pizza'}(\textsc{Rezept}) \bowtie \textsc{Vorrat}\Big)\Big)$$

e) Gesucht sind die Namen der kochbaren Rezepte, d.h. Rezepte, für die alle Zutaten mindestens mit der benötigten Menge im Vorrat sind.

$$\begin{aligned}\pi_{RName}(\textsc{Rezept}) \setminus \Big(&\pi_{RName}\Big(\sigma_{BMenge>VMenge}(\textsc{Rezept} \bowtie \textsc{Vorrat})\Big)\\&\cup \pi_{RName}(\textsc{Rezept} \setminus (\textsc{Rezept} \ltimes \textsc{Vorrat}))\Big)\end{aligned}$$

Die Relationen-Algebra ist ein Maß für die Ausdrucksfähigkeit von Datenmanipulationssprachen.

Definition 2.23 *Eine Datenmanipulationssprache L heißt genau dann relational vollständig, wenn jeder Term der Relationen-Algebra in L simuliert werden kann.*

Die Sprache L heißt genau dann streng relational vollständig, wenn jeder Term der Relationen-Algebra durch eine einzige Anweisung in L simuliert werden kann.

Grenzen der Relationen-Algebra:

Satz 2.24 *Es gibt keinen Term der Relationen-Algebra, der zu jeder zweistelligen Relation R deren transitive Hülle*

$$R^* = \{(x, y) \mid \exists\, x_1, \ldots, x_k : (x, x_1) \in R \wedge (x_1, x_2) \in R \wedge \cdots \wedge (x_k, y) \in R\}$$

berechnet.

2.2.3 Relationen-Kalkül

Der Bereichs-Kalkül und der Tupel-Kalkül sind mit der Sprache der Prädikatenlogik erster Stufe verwandt.

Im folgenden sei ein Relationales DB-Schema **D** zugrundegelegt.

Bereichs-Kalkül

Der Bereichs-Kalkül bildet die Grundlage für z.B. die Anfragesprache QBE.

Die Syntax von Bereichs-Kalkül-Formeln wird nun schrittweise festgelegt.

Definition 2.25

a) Ein Term des Bereichs-Kalküls ist entweder eine Konstante oder eine Variable zu Datentypen aus $\boldsymbol{D}$*. Falls* $u_1, \ldots, u_n$ *Terme (bzgl. der Datentypen* $D_1, \ldots, D_n$*) sind und* f *eine Datentyp-Operation* $D_1 \times \cdots \times D_n \to D$ *ist, so ist* $f(u_1, \ldots, u_n)$ *ein Term (bzgl.* D*).*

b) Eine atomare Formel hat die Form $\Theta(u_1, \ldots, u_n)$*, wobei* Θ *eine Boolesche Operation ist und* $u_1, \ldots, u_n$ *Terme bezeichnen, oder* $R(u_1, \ldots, u_n)$*, wobei* R *ein Relationenname mit dem Schema* $R(A_1 : D_1, \ldots, A_n : D_n)$ *und* $u_1, \ldots, u_n$ *Terme bzgl.* $D_1, \ldots, D_n$ *sind.*

c) Die Menge der Formeln des Bereichs-Kalküls ist die kleinste Menge F *von Ausdrücken, welche die atomaren Formeln enthält und für die gilt: Sind* γ *und* δ *in* F*, so auch* $\gamma \wedge \delta$*,* $\gamma \vee \delta$*,* $\neg\gamma$*,* $\forall\, x(\gamma)$ *und* $\exists\, x(\gamma)$*, wobei* x *eine Variable bezeichnet.*

Beispiel 2.26 *Es seien die Schemata* $R(A, B)$ *und* $S(C, B)$ *gegeben. Alle Attribute seien vom Datentyp „Integer“.*

a) $R(4, 16) \wedge S(16, 18)$

b) $\exists\, x(R(a, x) \wedge S(x, b))$

c) $R(a, x) \wedge \forall\, y(R(a, y) \Rightarrow y \leq x)$

Definition 2.27

a) Jedes Vorkommen einer Variablen in einer atomaren Formel ist frei.

b) Kommt x *in* γ *frei vor, so wird* x *durch* $\forall\, x(\gamma)$ *bzw.* $\exists\, x(\gamma)$ *gebunden.*

c) Kommt x *in* γ *bzw.* δ *frei bzw. gebunden vor, so ist dieses Vorkommen auch in* $\gamma \wedge \delta$*,* $\gamma \vee \delta$ *und* $\neg\gamma$ *frei bzw. gebunden.*

Wann ist eine Formel wahr bzw. gültig?

Semantik:

Fest gegeben sind Wertebereiche $|D|$ zu Datentypen D, Datentyp-Operationen und Werte zu Konstanten.

Definition 2.28

a) *Eine Abbildung* σ : *Relationennamen* $\longrightarrow$ *Relation,* $R \mapsto \sigma(R)$ *beschreibt einen Zustand von* **D**.

b) *Eine Abbildung, die Variablen einen Wert aus ihrem jeweiligen Wertebereich zuordnet, heißt Belegung:* $\beta : Variablen \longrightarrow Werte$.

Sind ein Zustand σ der Datenbank (**D**) und eine Belegung von Variablen β gegeben, so läßt sich der Wert eines Terms u bestimmen. Dieser wird mit $[\sigma, \beta](u)$ bezeichnet.

Definition 2.29 *Die Gültigkeit einer Formel* φ *im Zustand* σ *unter* β *(in Zeichen* $[\sigma, \beta] \models \varphi$*) wird festgelegt durch:*

a) $[\sigma, \beta] \models \Theta(u_1, \ldots, u_n) \iff \Theta([\sigma, \beta](u_1), [\sigma, \beta](u_2), \ldots [\sigma, \beta](u_n)) = wahr$

b) $[\sigma, \beta] \models R(u_1, \ldots, u_n) \iff ([\sigma, \beta](u_1), \ldots, [\sigma, \beta](u_n)) \in \sigma(R)$

c) $[\sigma, \beta] \models \varphi_1 \wedge \varphi_2 \iff [\sigma, \beta] \models \varphi_1$ *und* $[\sigma, \beta] \models \varphi_2$

d) $[\sigma, \beta] \models \varphi_1 \vee \varphi_2 \iff [\sigma, \beta] \models \varphi_1$ *oder* $[\sigma, \beta] \models \varphi_2$

e) $[\sigma, \beta] \models \neg\varphi \iff \varphi$ *gilt nicht im Zustand* σ *unter der Belegung* β

f) $[\sigma, \beta] \models \exists\, x(\varphi) \iff$ *es gibt ein* d *aus dem Wertebereich von* x*, so daß* $[\sigma, \beta'] \models \varphi$*, wobei* $\beta'(y) = \beta(y)$ *und zusätzlich* $\beta'(x) = d$*, falls* x *in* φ *frei vorkommt*

g) $[\sigma, \beta] \models \forall\, x(\varphi) \iff$ *für alle* d *aus dem Wertebereich von* x*:* $[\sigma, \beta'] \models \varphi$*, wobei* $\beta'(y) = \beta(y)$ *und zusätzlich* $\beta'(x) = d$*, falls* x *in* φ *frei vorkommt*

Jetzt können Ausdrücke des Bereichs-Kalküls festgelegt werden, die (deskriptiv) eine (Ergebnis-)Relation liefern.

Definition 2.30

a) *Ein Ausdruck des Bereichs-Kalküls ist ein Ausdruck der Form* $\{x_1, \ldots, x_n \mid \varphi\}$*, wobei* φ *eine Formel mit den einzigen freien Variablen* $x_1 : D_1, \ldots, x_n : D_n$ *ist (Ergebnisvariablen).*

b) *In einem gegebenen DB-Zustand* σ *bestimmt* $\{x_1, \ldots, x_n|\varphi\}$ *die folgende Relation über* $|D_1| \times \cdots \times |D_n|$*:* $\{(d_1, \ldots, d_n) \mid$ *es gibt eine Belegung* β *mit* $\beta(x_i) = d_i$*,* $i = 1, \ldots, n$*, so daß* $[\sigma, \beta] \models \varphi\}$

Beispiel 2.31

VORRAT(*Zutat, VMenge*)
REZEPT(*RName, Zutat, BMenge*)
HERKUNFT(*RName, Land*)

a) Gesucht sind die Zutaten und Vorratsmengen von chinesischen Rezepten.

$$\{zutat, vmenge \mid \textsc{Vorrat}(zutat, vmenge) \wedge \exists\, rname\, \exists\, bmenge \Big(\textsc{Rezept}(rname, zutat, bmenge) \wedge \textsc{Herkunft}(rname, \text{'China'})\Big)\}$$

b) Gesucht sind alle Rezeptnamen, die keinen Knobi, aber Rindfleisch verwenden.

$$\{rname \mid \exists\, bmenge\Big(\textsc{Rezept}(rname, \text{'Rindfleisch'}, bmenge)\Big) \wedge \neg\exists\, bmenge'\Big(\textsc{Rezept}(rname, \text{'Knobi'}, bmenge')\Big)\}$$

c) Gesucht sind alle vorrätigen Zutaten des Rezeptes „Pizza“, deren Menge nicht ausreicht.

$$\{zutat \mid \exists\, vmenge\Big(\textsc{Vorrat}(zutat, vmenge) \wedge \Big(\exists\, bmenge(\textsc{Rezept}(\text{'Pizza'}, zutat, bmenge) \wedge vmenge < bmenge)\Big)\Big)\}$$

Definition 2.32 *Ein Ausdruck heißt sicher, wenn er in jedem Zustand eine endliche Relation bestimmt.*

Beispiel 2.33 *Der Ausdruck* $\{(x, y) \mid \neg R(x, y)\}$ *ist nicht sicher.*

Definition 2.34 *Zwei Ausdrücke heißen äquivalent, falls sie in jedem Zustand die gleiche Relation bestimmen.*

Satz 2.35 *Der Bereichs-Kalkül ist streng relational vollständig, d.h., zu jedem Term der Relationen-Algebra gibt es einen äquivalenten sicheren Ausdruck des Bereichs-Kalküls.*

Beweis Induktion über den Termaufbau der Relationen-Algebra.
Z.B. Darstellung von Relationen-Operationen im Bereichs-Kalkül.

Gegeben sei $R(A_1, \ldots, A_n)$, $S(B_1, \ldots, B_m)$.

$$
\begin{aligned}
R \cup S &= \{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n) \vee S(x_1, \ldots, x_n)\} \\
R \setminus S &= \{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n) \wedge \neg S(x_1, \ldots, x_n)\} \\
R \times S &= \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m} \mid R(x_1, \ldots, x_n) \wedge S(x_{n+1}, \ldots, x_{n+m})\} \\
\pi_{A_{i_1},\ldots,A_{i_k}}(R) &= \{y_1, \ldots, y_k \mid \exists\, x_1 \cdots \exists\, x_n (R(x_1, \ldots, x_n) \wedge y_1 = x_{i_1} \wedge \cdots \wedge y_k = x_{i_k})\} \\
\sigma_\varphi(R) &= \{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n) \wedge \varphi',\ \text{wobei } \varphi' \text{ aus } \varphi \text{ hervorgeht,} \\
&\qquad \text{indem die Symbole } A_i \text{ durch Variablennamen } x_i \text{ ersetzt werden}\}
\end{aligned}
$$

□

Satz 2.36 *Zu jedem sicheren Ausdruck des Bereichs-Kalküls gibt es einen äquivalenten Term der Relationen-Algebra.*

Tupel-Kalkül

Der Tupel-Kalkül bildet die Grundlage z.B. für die Anfragesprachen SQL und QUEL.

Definition 2.37

a) *Ein Term des Tupel-Kalküls ist entweder eine Konstante oder die Einschränkung einer Tupel-Variablen auf eine Stelle (Bildung von Komponenten). Falls den Termen $u_1, \ldots, u_n$ die Datentypen $D_1, \ldots, D_n$ zugeordnet sind und f eine Datentyp-Operation $D_1 \times \cdots \times D_n \to D$ ist, so ist $f(u_1, \ldots, u_n)$ ein Term (bzgl. D).*

b) *Eine atomare Formel hat die Form $\Theta(u_1, \ldots, u_n)$, wobei Θ eine Boolesche Operation ist und $u_1, \ldots, u_n$ Terme bezeichnen, oder $R(r)$, wobei R ein Relationenname und r eine passende Tupel-Variable ist, oder $r = s$, wobei r und s passende Tupel-Variablen sind.*

c) *(Wie Bereichs-Kalkül) Die Menge der Formeln des Tupel-Kalküls ist die kleinste Menge F von Ausdrücken, welche die atomaren Formeln enthält und für die gilt: Sind γ und δ in F, so auch $\gamma \wedge \delta$, $\gamma \vee \delta$, $\neg\gamma$, $\forall\, x(\gamma)$ und $\exists\, x(\gamma)$, wobei x eine Tupel-Variable bezeichnet.*

Semantik:

Abweichend vom Bereichs-Kalkül muß eine Belegung im Tupel-Kalkül jeder Komponente einen Wert zuordnen.

$$\beta : Variablen \longrightarrow Tupel, \qquad r \mapsto \beta(r) = (d_1, \ldots, d_n) \in |D_1| \times \cdots |D_n|$$

Entsprechend ergibt sich der Wert eines Terms $r.A_i$ zu

$$[\sigma, \beta](r.A_i) = (\beta(r))_i = d_i$$

Die Gültigkeit einer Formel $R(r)$ wird festgelegt durch:

$$[\sigma, \beta] \models R(r) \iff \beta(r) \in \sigma(R)$$

Definition 2.38

a) Ein Ausdruck des Tupel-Kalküls ist ein Ausdruck der Form $\{r \mid \varphi\}$, wobei r die einzige freie Variable in φ ist.

b) In einem gegebenen DB-Zustand σ wird durch den Ausdruck die Relation $\{\beta(r) \mid \beta$ ist Belegung, so daß $[\sigma, \beta] \models \varphi\}$ bestimmt.

Beispiel 2.39

VORRAT*(Zutat, VMenge)*
REZEPT*(RName, Zutat, BMenge)*
HERKUNFT*(RName, Land)*

a) Gesucht sind die Zutaten und Vorratsmengen von chinesischen Rezepten.

$$\{v \mid \text{VORRAT}(v) \wedge \exists\, r(\text{REZEPT}(r) \wedge \exists\, h(\text{HERKUNFT}(h) \wedge v.Zutat = r.Zutat \wedge r.RName = h.RName \wedge h.Land = \text{'China'}))\}$$

b) Gesucht sind alle Rezeptnamen, die keinen Knobi, aber Rindfleisch verwenden.

$$\{(r.RName) \mid \text{REZEPT}(r) \wedge r.Zutat = \text{'Rindfleisch'} \wedge \neg\exists\, r'(\text{REZEPT}(r') \wedge r'.Zutat = \text{'Knobi'} \wedge r.RName = r'.RName)\}$$

c) Gesucht sind alle Zutaten im Vorrat, die in keinem Rezept verwendet werden.

$$\{(v.Zutat) \mid \text{VORRAT}(v) \wedge \neg\exists\, r(\text{REZEPT}(r) \wedge r.Zutat = v.Zutat)\}$$

d) Gesucht sind alle vorrätigen Zutaten des Rezeptes „Pizza“, deren Menge nicht ausreicht.

$$\{(r.Zutat) \mid \text{REZEPT}(r) \wedge r.RName = \text{'Pizza'} \wedge \exists\, v(\text{VORRAT}(v) \wedge r.Zutat = v.Zutat \wedge r.BMenge > v.VMenge)\}$$

e) Gesucht sind die Namen der kochbaren Rezepte, d.h. Rezepte, für die alle Zutaten mindestens mit der benötigten Menge im Vorrat sind.

$$\{(r.RName) \mid \text{REZEPT}(r) \wedge \forall\, r'((\text{REZEPT}(r') \wedge r'.RName = r.RName) \Rightarrow \exists\, v(\text{VORRAT}(v) \wedge r'.Zutat = v.Zutat \wedge v.VMenge \geq r'.BMenge))\}$$

Satz 2.40 *Zu jedem (sicheren) Ausdruck des Bereichs-Kalküls gibt es einen äquivalenten (sicheren) Ausdruck des Tupel-Kalküls.*

Also sind Relationen-Kalkül (Bereichs- und Tupel-Kalkül) und Relationen-Algebra in ihrer Ausdrucksfähigkeit äquivalent.

2.2.4 Änderungen in Relationalen Datenbanken

Zum Abschluß dieses Kapitels wollen wir für das Relationale Modell noch kurz auf die Problematik von Datenbank-Änderungen eingehen, welche neben den Anfragen die zweite Klasse von DML-Operationen darstellen.

Da wir Relationen als Mengen verstehen, sind die Änderungs-Operationen „Einfügen“ und „Löschen“ formal durch die Mengenoperationen Vereinigung und Differenz beschreibbar. Die Operation „Ändern“ können wir zunächst als ein Löschen, gefolgt von einem Einfügen, auffassen. Für eine Relation $R(A_1, \ldots, A_n)$ und passende Tupel r und s können wir daher festlegen:

$$\begin{aligned} \mathit{insert}(R, r) &:= R \cup \{r\} \\ \mathit{delete}(R, r) &:= R \setminus \{r\} \\ \mathit{modify}(R, r, s) &:= \mathit{insert}(\mathit{delete}(R, r), s) \end{aligned}$$

Die Relationen-Algebra soll hier nicht formal um Änderungs-Ausdrücke erweitert werden, sondern wir wollen lediglich ein Phänomen (es gibt weitere) erwähnen, welches im Zusammenhang mit relationalen Änderungen zu berücksichtigen ist. Im folgenden Kapitel werden z.B. die konkreten Änderungs-Operationen von SQL behandelt.

Umfaßt eine relationale DML die Relationen-Algebra und ferner Änderungs-Operationen, so ist sie de facto „übervollständig“, da sich dieses Kriterium allein auf die Anfrage-Operationen bezieht.

Zur Definition der Änderungs-Operationen wird man im allgemeinen wenigstens für Löschen und Ändern zulassen, daß man Tupel, welche von der Operation betroffen sein sollen, durch eine (Selektions-) Bedingung beschreiben darf, so daß eine Lösch- bzw. Änderungsoperation eine Tupel-Menge betreffen kann.

Die Darstellung der Änderungs-Operation durch eine Lösch- gefolgt von einer Einfüge-Operation ist nicht stets möglich, wie das folgende Beispiel zeigen soll: Es sei $R(A, B)$ die folgende Relation:

R	A	B
	1	1
	2	2

Falls der „Änderungs-Auftrag“ eines Benutzers nun darin besteht, den A-Wert für jedes Tupel, dessen A-Wert 0 ist, auf 1 zu setzen, so ist hierfür eine Modify-Operation erforderlich, welche nicht durch Löschen und Einfügen „simuliert“ werden kann. Falls man etwa das „Programm“

$$\begin{aligned} R' &:= \mathit{delete}(R, (0, 0)) \\ R'' &:= \mathit{insert}(R', (1, 0)) \end{aligned}$$

auf r anwendet, so erhält man

R''	A	B
	1	1
	2	2
	1	0

während in R de facto kein Tupel r existiert, welches die Bedingung $r.A = 0$ erfüllt, so daß der Änderungs-Auftrag R eigentlich unverändert lassen müßte.