Kapitel 6

Das Relationale Modell

6.1 Definition

Gegeben sind n nicht notwendigerweise unterschiedliche *Wertebereiche* (auch *Domänen* genannt) $D_1, \ldots, D_n$, welche nur *atomare* Werte enthalten, die nicht strukturiert sind, z.B. Zahlen oder Strings.

Eine Relation R ist definiert als Teilmenge des kartesischen Produkts der n Domänen:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n$$

Es wird unterschieden zwischen dem *Schema* einer Relation, gegeben durch die n Domänen und der aktuellen *Ausprägung* (Instanz). Ein Element der Menge R wird als Tupel bezeichnet, dessen $Stelligkeit$ sich aus dem Relationenschema ergibt. Wir bezeichnen mit **sch**(R) oder mit $\mathcal{R} = A_1, \ldots, A_n$ die Menge der Attribute und mit R die aktuelle Ausprägung. Mit **dom**(A) bezeichnen wird die Domäne eines Attributs A. Also gilt

$$R \subseteq \mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \ldots \times \mathrm{dom}(A_n)$$

Im Datenbankbereich müssen die Domänen außer einem Typ noch einen Namen haben. Wir werden Relationenschemata daher durch eine Folge von Bezeichner/Wertebereich - Tupeln spezifizieren, z.B.

Telefonbuch : { [Name : string, Adresse: string, <u>TelefonNr : integer</u>] }

Hierbei wird in den eckigen Klammern [...] angegeben, wie die Tupel aufgebaut sind, d.h. welche Attribute vorhanden sind und welchen Wertebereich sie haben. Ein Schlüsselkandidat wird unterstrichen. Die geschweiften Klammern { ... } sollen ausdrücken, daß es sich bei einer Relationenausprägung um eine Menge von Tupeln handelt. Zur Vereinfachung wird der Wertebereich auch manchmal weggelassen:

Telefonbuch : { [Name, Adresse, <u>TelefonNr</u>] }

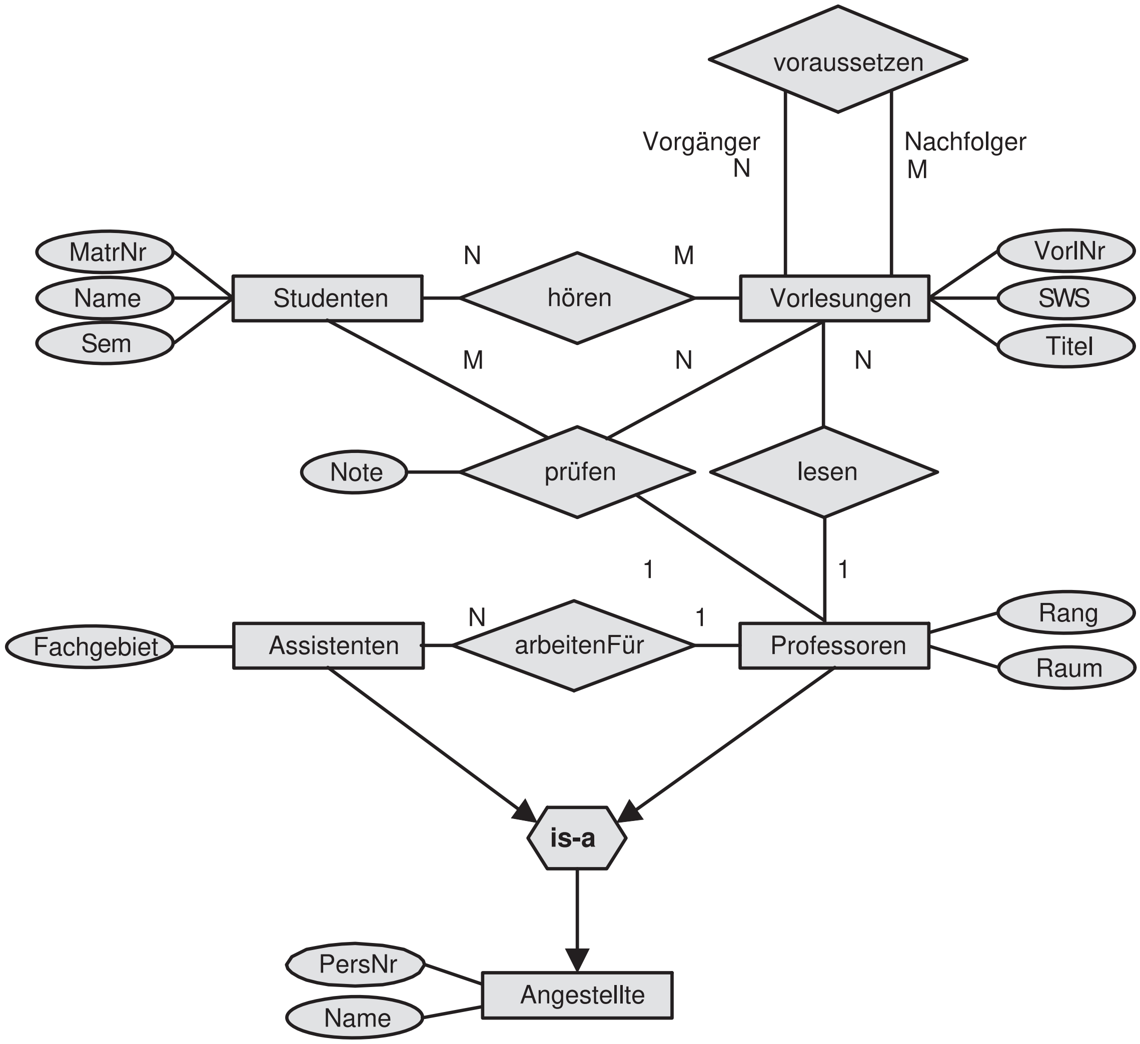

Abbildung 6.1: Konzeptuelles Schema der Universität

6.2 Umsetzung in ein relationales Schema

Das ER-Modell besitzt zwei grundlegende Strukturierungskonzepte:

- Entity-Typen
- Relationship-Typen

Abbildung 6.1 zeigt ein ER-Diagramm zum Universitätsbetrieb. Zunächst wollen wir die Generalisierung ignorieren, da es im relationalen Modell keine unmittelbare Umsetzung gibt. Dann verbleiben vier Entity-Typen, die auf folgende Schemata abgebildet werden:

Studenten	:	{[$\underline{\text{MatrNr : integer}}$, Name : string, Semester : integer] }
Vorlesungen	:	{[$\underline{\text{VorlNr : integer}}$, Titel : string, SWS : integer] }
Professoren	:	{[$\underline{\text{PersNr : integer}}$, Name : string, Rang : string, Raum : integer] }
Assistenten	:	{[$\underline{\text{PersNr : integer}}$, Name : string, Fachgebiet : string] }

Bei der relationalen Darstellung von Beziehungen richten wir im *Initial*-Entwurf für jeden Beziehungstyp eine eigene Relation ein. Später kann davon ein Teil wieder eliminiert werden. Grundsätzlich entsteht das Relationenschema durch die Folge aller Schlüssel, die an der Beziehung beteiligt sind sowie ggf. weitere Attribute der Beziehung. Dabei kann es notwendig sein, einige der Attribute umzubenennen. Die Schlüsselattribute für die referierten Entity-Typen nennt man *Fremdschlüssel*.

Für das Universitätsschema entstehen aus den Relationships die folgenden Schemata:

hören	:	{[$\underline{\text{MatrNr : integer}}$, $\underline{\text{VorlNr : integer}}$] }
lesen	:	{[PersNr : integer, $\underline{\text{VorlNr : integer}}$] }
arbeitenFür	:	{[$\underline{\text{AssiPersNr : integer}}$, ProfPersNr : integer] }
voraussetzen	:	{[$\underline{\text{Vorgänger : integer}}$, $\underline{\text{Nachfolger : integer}}$] }
prüfen	:	{[$\underline{\text{MatrNr : integer}}$, $\underline{\text{VorlNr : integer}}$, PersNr : integer, Note : decimal] }

Unterstrichen sind jeweils die Schlüssel der Relation, eine *minimale* Menge von Attributen, deren Werte die Tupel eindeutig identifizieren.

Da die Relation *hören* eine $N : M$-Beziehung darstellt, sind sowohl die Vorlesungsnummern als auch die Matrikelnummern alleine keine Schlüssel, wohl aber ihre Kombination.

Bei der Relation *lesen* liegt eine 1:N-Beziehung vor, da jeder Vorlesung genau ein Dozent zugeordnet ist mit der partiellen Funktion

$$lesen : Vorlesungen \to Professoren$$

Also ist für die Relation *lesen* bereits das Attribut *VorlNr* ein Schlüsselkandidat, für die Relation *arbeitenFür* bildet die *AssiPersNr* einen Schlüssel.

Bei der Relation *prüfen* liegt wiederum eine partielle Abbildung vor:

$$prüfen : Studenten \times Vorlesungen \to Professoren$$

Sie verlangt, daß *MatrNr* und *VorlNr* zusammen den Schlüssel bilden.

6.3 Verfeinerung des relationalen Schemas

Das im Initialentwurf erzeugte relationale Schema läßt sich verfeinern, indem einige der $1 : 1$-, $1 : N$- oder $N : 1$-Beziehungen eliminiert werden. Dabei dürfen nur Relationen mit gleichem Schlüssel zusammengefaßt werden.

Nach dieser Regel können von den drei Relationen

Vorlesungen	:	{[$\underline{\text{VorlNr : integer}}$, Titel : string, SWS : integer] }
Professoren	:	{[$\underline{\text{PersNr : integer}}$, Name : string, Rang : string, Raum : integer] }
lesen	:	{[PersNr : integer, $\underline{\text{VorlNr : integer}}$] }

die Relationen *Vorlesungen* und *lesen* zusammengefaßt werden. Somit verbleiben im Schema

Vorlesungen : {[VorlNr : integer, Titel : string, SWS : integer, gelesenVon : integer] }
Professoren : {[PersNr : integer, Name : string, Rang : string, Raum : integer] }

Das Zusammenlegen von Relationen mit unterschiedlichen Schlüsseln erzeugt eine Redundanz von Teilen der gespeicherten Information. Beispielsweise speichert die (unsinnige) Relation

Professoren' : {[PersNr, liestVorl, Name, Rang, Raum] }

zu jeder von einem Professor gehaltenen Vorlesung seinen Namen, seinen Rang und sein Dienstzimmer:

Professoren´				
PersNrN	liestVorl	Name	Rang	Raum
2125	5041	Sokrates	C4	226
2125	5049	Sokrates	C4	226
2125	4052	Sokrates	C4	226

Bei 1 : 1-Beziehungen gibt es zwei Möglichkeiten, die ursprünglich entworfene Relation mit den beteiligten Entity-Typen zusammenzufassen.

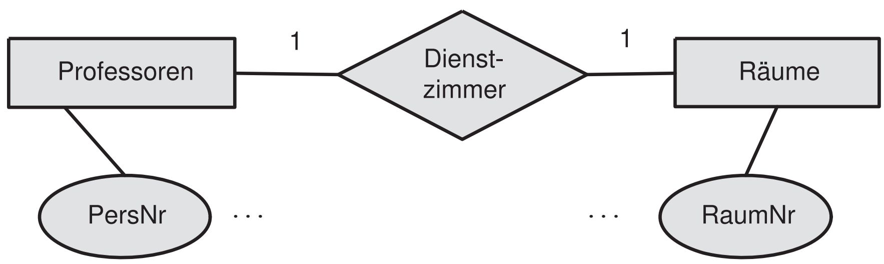

Abbildung 6.2: Beispiel einer 1:1-Beziehung

Abbildung 6.2 zeigt eine mögliche Modellierung für die Unterbringung von Professoren in Räumen als 1 : 1-Beziehung. Die hierzu gehörenden Relationen heißen

Professoren : {[PersNr, Name, Rang] }
Räume : {[RaumNr, Größe, Lage] }
Dienstzimmer : {[PersNr, RaumNr] }

Da *Professoren* und *Dienstzimmer* denselben Schlüssel haben, kann zusammengefaßt werden zu

Professoren : {[PersNr, Name, Rang, Raum] }
Räume : {[RaumNr, Größe, Lage] }

Da das Attribut *RaumNr* innerhalb der Relation *Dienstzimmer* ebenfalls einen Schlüssel bildet, könnten als Alternative auch die Relationen *Dienstzimmer* und *Räume* zusammengefaßt werden:

Professoren : {[PersNr, Name, Rang] }
Räume : {[RaumNr, Größe, Lage, ProfPersNr] }

Diese Modellierung hat allerdings den Nachteil, daß viele Tupel einen sogenannten Nullwert für das Attribut *ProfPersNr* haben, da nur wenige Räume als Dienstzimmer von Professoren genutzt werden.

Die in Abbildung 6.1 gezeigte Generalisierung von *Assistenten* und *Professoren* zu *Angestellte* könnte wie folgt durch drei Relationen dargestellt werden:

Angestellte : {[PersNr, Name] }
Professoren : {[PersNr, Rang, Raum] }
Assistenten : {[PersNr, Fachgebiet] }

Hierdurch wird allerdings die Information zu einem Professor, wie z.B.

[2125, Sokrates, C4, 226]

auf zwei Tupel aufgeteilt:

[2125, Sokrates] und [2125, C4, 226]

Um die vollständige Information zu erhalten, müssen diese beiden Relationen verbunden werden (Join).

Abbildung 6.3 zeigt eine Beispiel-Ausprägung der Universitäts-Datenbasis. Das zugrundeliegende Schema enthält folgende Relationen:

Studenten : {[MatrNr : integer, Name : string, Semester : integer] }
Vorlesungen : {[VorlNr : integer, Titel : string, SWS : integer, gelesenVon : integer] }
Professoren : {[PersNr : integer, Name : string, Rang : string, Raum : integer] }
Assistenten : {[PersNr : integer, Name : string, Fachgebiet : string, Boss : integer] }
hören : {[MatNr : integer, VorlNr : integer] }
voraussetzen : {[Vorgänger : integer, Nachfolger : integer] }
prüfen : {[MatrNr : integer, VorlNr : integer, PersNr : integer, Note :decimal] }

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Abbildung 6.3: Beispielausprägung der Universitäts-Datenbank

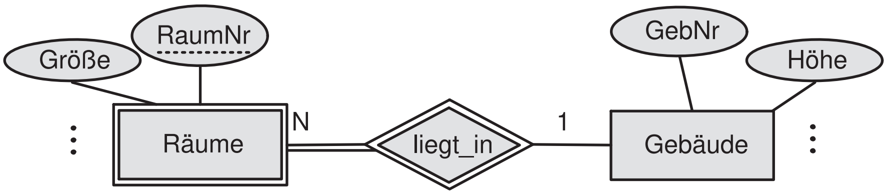

Abbildung 6.4: Schwacher Entity-Typ

Zur Modellierung von schwachen Entity-Typen betrachten wir Abbildung 6.4, in der mittels der Relation *liegt_in* der schwache Entitity-Typ *Räume* dem Entity-Typ *Gebäude* untergeordnet wurde.

Wegen der $1 : N$-Beziehung zwischen *Gebäude* und *Räume* kann die Beziehung *liegt_in* verlagert werden in die Relation Räume:

$$\text{Räume} : \{[\ \underline{\text{GebNr, RaumNr}}, \text{Größe}]\ \}$$

Ein Beziehung *bewohnt* zwischen *Professoren* und *Räume* benötigt als Fremdschlüssel zum einen die Personalnummer des Professors und zum anderen die Kombination von Gebäude-Nummer und Raum-Nummer:

$$\text{bewohnt} : \{[\ \underline{\text{PersNr, GebNr, RaumNr}}]\ \}$$

Da *bewohnt* eine $1 : 1$-Beziehung darstellt, kann sie durch einen Fremdschlüssel beim Professor realisiert werden. Ist die beim *Gebäude* hinterlegte Information eher gering, käme auch, wie im Universitätsschema in Abbildung 6.1 gezeigt, ein Attribut *Raum* bei den *Professoren* infrage.

6.4 Abfragesprachen

Es gibt verschiedene Konzepte für formale Sprachen zur Formulierung einer Anfrage (Query) an ein relationales Datenbanksystem:

- Relationenalgebra (prozedural):
  Verknüpft konstruktiv die vorhandenen Relationen durch Operatoren wie $\cup, \cap, \ldots$:

- Relationenkalkül (deklarativ):
  Beschreibt Eigenschaften des gewünschten Ergebnisses mit Hilfe einer Formel der Prädikatenlogik 1. Stufe unter Verwendung von $\wedge, \vee, \neg, \exists, \forall$.

- SQL (kommerziell):
  Stellt eine in Umgangssprache gegossene Mischung aus Relationenalgebra und Relationenkalkül dar (siehe Kapitel 7).

- Query by Example (für Analphabeten):
  Verlangt vom Anwender das Ausfüllen eines Gerüstes mit Beispiel-Einträgen.

6.5 Relationenalgebra

Die Operanden der Sprache sind Relationen. Als unabhängige Operatoren gibt es *Selektion, Projektion, Vereinigung, Mengendifferenz, Kartesisches Produkt, Umbenennung*; daraus lassen sich weitere Operatoren *Verbund, Durchschnitt, Division* ableiten.

Selektion :

Es werden diejenigen Tupel ausgewählt, die das *Selektionsprädikat* erfüllen. Die Selektion wird mit σ bezeichnet und hat das Selektionsprädikat als Subskript.

Die Projektion

$$\sigma_{Semester>10}(Studenten)$$

liefert als Ergebnis

$\sigma_{Semester>10}(Studenten)$		
MatNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12

Das Selektionsprädikat wird beschrieben durch eine Formel F mit folgenden Bestandteilen:

- Attributnamen der Argumentrelation R oder Konstanten als Operanden
- arithmetische Vergleichsoperatoren $< = > \leq \neq \geq$
- logische Operatoren: $\wedge$ $\vee$ $\neg$ (und oder nicht)

Projektion :

Bei der Projektion werden Spalten der Argumentrelation extrahiert. Das Operatorsymbol lautet Π, die gewünschten Attribute werden im Subskript aufgelistet:

$$\Pi_{Rang}(Professoren)$$

liefert als Ergebnis

$\Pi_{Rang}(Professoren)$
Rang
C4
C3

Die Attributmenge wird üblicherweise nicht unter Verwendung von Mengenklammern sondern als durch Kommata getrennte Sequenz gegeben. Achtung: Da das Ergebnis wiederum eine Relation ist, existieren per definitionem keine Duplikate ! In der Praxis müssen sie dennoch algorithmisch entfernt werden.

Vereinigung :

Zwei Relationen mit gleichem Schema können durch die Vereinigung, symbolisiert durch $\cup$, zusammengefaßt werden. Beispiel:

$$\Pi_{PersNr,Name}(Assistenten) \cup \Pi_{PersNr,Name}(Professoren)$$

Mengendifferenz :

Für zwei Relationen R und S mit gleichem Schema ist die Mengendifferenz $R - S$ definiert als die Menge der Tupel, die in R aber nicht in S vorkommen. Beispiel

$$\Pi_{MatrNr}(Studenten) - \Pi_{MatrNr}(prüfen)$$

liefert die Matrikelnummern derjenigen Studenten, die noch nicht geprüft wurden.

Kartesisches Produkt :

Das kartesische Produkt (Kreuzprodukt) zweier Relationen R und S wird mit $R \times S$ bezeichnet und enthält alle möglichen Paare von Tupeln aus R und S. Das Schema der Ergebnisrelation, also **sch**$(R \times S)$, ist die Vereinigung der Attribute aus **sch**(R) und **sch**(S).

Das Kreuzprodukt von *Professoren* und *hören* hat 6 Attribute und enthält 91 $(= 7 \cdot 13)$ Tupel.

Professoren × hören					
Professoren				hören	
PersNr	name	Rang	Raum	MatNr	VorlNr
2125	Sokrates	C4	226	26120	5001
...	...	...	...	...	...
2125	Sokrates	C4	226	29555	5001
...	...	...	...	...	...
2137	Kant	C4	7	29555	5001

Haben beide Argumentrelationen ein gleichnamiges Attribut A, so kann dies durch Voranstellung des Relationennamen R in der Form $R.A$ identifiziert werden.

Umbenennung von Relationen und Attributen :

Zum Umbenennen von Relationen und Attributen wird der Operator ρ verwendet, wobei im Subskript entweder der neue Relationenname steht oder die Kombination von neuen und altem Attributnamen durch einen Linkspfeil getrennt. Beispiele:

$$\rho_{Dozenten}(Professoren)$$

$$\rho_{Zimmer \leftarrow Raum}(Professoren)$$

Eine Umbenennung kann dann erforderlich werden, wenn durch das kartesische Produkt Relationen mit identischen Attributnamen kombiniert werden sollen.

Als Beispiel betrachten wir das Problem, die Vorgänger der Vorgänger der Vorlesung mit der Nummer 5216 herausfinden. Hierzu ist ein kartesisches Produkt der Tabelle mit sich selbst erforderlich, nachdem zuvor die Spalten umbenannt worden sind:

$$\Pi_{V1.Vorgänger}(\sigma_{V2.Nachfolger=5216 \wedge V1.Nachfolger=V2.Vorgänger}$$

$$(\rho_{V1}(voraussetzen) \times \rho_{V2}(voraussetzen)))$$

Die konstruierte Tabelle hat vier Spalten und enthält das Lösungstupel mit dem Wert 5001 als Vorgänger von 5041, welches wiederum der Vorgänger von 5216 ist:

V1		V2	
Vorgänger	Nachfolger	Vorgänger	Nachfolger
5001	5041	5001	5041
...	...	...	...
5001	5041	5041	5216
...	...	...	...
5052	5259	5052	5259

Natürlicher Verbund (Join) :

Der sogenannte *natürliche Verbund* zweier Relationen R und S wird mit $R \bowtie S$ gebildet. Wenn R insgesamt $m + k$ Attribute $A_1, \ldots, A_m, B_1, \ldots, B_k$ und S insgesamt $n + k$ Attribute $B_1, \ldots, B_k, C_1, \ldots, C_n$ hat, dann hat $R \bowtie S$ die Stelligkeit $m+k+n$. Hierbei wird vorausgesetzt, daß die Attribute A_i und C_j jeweils paarweise verschieden sind. Das Ergebnis von $R \bowtie S$ ist definiert als

$$R \bowtie S \ := \ \Pi_{A_1,\ldots,A_m,R.B_1,\ldots,R.B_k,C_1,\ldots,C_n}(\sigma_{R.B_1=S.B_1 \wedge \ldots \wedge R.B_k=S.B_k}(R \times S))$$

Es wird also das kartesische Produkt gebildet, aus dem nur diejenigen Tupel selektiert werden, deren Attributwerte für gleichbenannte Attribute der beiden Argumentrelationen gleich sind. Diese gleichbenannten Attribute werden in das Ergebnis nur einmal übernommen.

Die Verknüpfung der *Studenten* mit ihren *Vorlesungen* geschieht durch

$$(Studenten \bowtie hören) \bowtie Vorlesungen$$

Das Ergebnis ist eine 7-stellige Relation:

$(Studenten \bowtie hören) \bowtie Vorlesungen$						
MatrNr	Name	Semester	VorlNr	Titel	SWS	gelesenVon
26120	Fichte	10	5001	Grundzüge	4	2137
25403	Jonas	12	5022	Glaube und Wissen	2	2137
28106	Carnap	3	4052	Wissenschaftstheorie	3	2126
...	...	...	...	...	...	...

Da der Join-Operator assoziativ ist, können wir auch auf die Klammerung verzichten und einfach schreiben

$$Studenten \bowtie hören \bowtie Vorlesungen$$

Wenn zwei Relationen verbunden werden sollen bzgl. zweier Attribute, die zwar die gleiche Bedeutung aber unterschiedliche Benennungen haben, so müssen diese vor dem Join mit dem ρ-Operator umbenannt werden. Zum Beispiel liefert

$$Vorlesungen \bowtie \rho_{gelesenVon \leftarrow PersNr}(Professoren)$$

die Relation $\{[VorlNr, Titel, SWS, gelesenVon, Name, Rang, Raum]\}$

Allgemeiner Join :

Beim natürlichen Verbund müssen die Werte der gleichbenannten Attribute übereinstimmen. Der allgemeine Join-Operator, auch *Theta-Join* genannt, erlaubt die Spezifikation eines beliebigen Join-Prädikats θ. Ein Theta-Join $R \bowtie_\theta S$ über der Relation R mit den Attributen $A_1, A_2, \ldots, A_n$ und der Relation S mit den Attributen $B_1, B_2, \ldots, B_n$ verlangt die Einhaltung des Prädikats θ, beispielsweise in der Form

$$R \bowtie_{A_1 < B_1 \wedge A_2 = B_2 \wedge A_3 < B_5} S$$

Das Ergebnis ist eine $n+m$-stellige Relation und läßt sich auch als Kombination von Kreuzprodukt und Selektion schreiben:

$$R \bowtie_\theta S := \sigma_\theta(R \times S)$$

Wenn in der Universitätsdatenbank die *Professoren* und die *Assistenten* um das Attribut *Gehalt* erweitert würden, so könnten wir diejenigen Professoren ermitteln, deren zugeordnete Assistenten mehr als sie selbst verdienen:

$$Professoren \bowtie_{Professoren.Gehalt < Assistenten.Gehalt \wedge Boss = Professoren.PersNr} Assistenten$$

Die bisher genannten Join-Operatoren werden auch innere Joins genannt (*inner join*). Bei ihnen gehen diejenigen Tupel der Argumentrelationen verloren, die keinen Join-Partner gefunden haben. Bei den äußeren Join-Operatoren (*outer joins*) werden - je nach Typ des Joins - auch partnerlose Tupel gerettet:

- left outer join: Die Tupel der linken Argumentrelation bleiben erhalten
- right outer join: Die Tupel der rechten Argumentrelation bleiben erhalten
- full outer join: Die Tupel beider Argumentrelationen bleiben erhalten

Somit lassen sich zu zwei Relationen L und R insgesamt vier verschiedene Joins konstruieren:

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

inner Join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

left outer join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

right outer Join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
-	-	c_3	d_2	e_2

outer Join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

Mengendurchschnitt :

Der Mengendurchschnitt (Operatorsymbol $\cap$) ist anwendbar auf zwei Argumentrelationen mit gleichem Schema. Im Ergebnis liegen alle Tupel, die in beiden Argumentrelationen liegen. Beispiel:

$$\Pi_{PersNr}(\rho_{PersNr \leftarrow gelesenVon}(Vorlesungen)) \cap \Pi_{PersNr}(\sigma_{Rang=C4}(Professoren))$$

liefert die Personalnummer aller C4-Professoren, die mindestens eine Vorlesung halten.

Der Mengendurchschnitt läßt sich mithilfe der Mengendifferenz bilden:

$$R \cap S \; = \; R \setminus (R \setminus S)$$

Division :

Sei R eine r-stellige Relation, sei S eine s-stellige Relation, deren Attributmenge in R enthalten ist.

Mit der Division

$$Q \; := \; R \div S \; := \; \{t = t_1, t_2, \ldots, t_{r-s} \mid \forall u \in S \; : \; tu \in R\}$$

sind alle die Anfangsstücke von R gemeint, zu denen sämtliche Verlängerungen mit Tupeln aus S in der Relation R liegen. Beispiel:

$$\begin{array}{|c|c|}\hline \multicolumn{2}{|c|}{R} \\ \hline M & V \\ \hline m_1 & v_1 \\ m_1 & v_2 \\ m_1 & v_3 \\ m_2 & v_2 \\ m_2 & v_3 \\ \hline \end{array} \;\div\; \begin{array}{|c|}\hline S \\ \hline V \\ \hline v_1 \\ v_2 \\ \hline \end{array} \;=\; \begin{array}{|c|}\hline R \div S \\ \hline M \\ \hline m_1 \\ \hline \end{array}$$

Die Division von R durch S läßt sich schrittweise mithilfe der unabhängigen Operatoren nachvollziehen (zur Vereinfachung werden hier die Attribute statt über ihre Namen über ihren Index projiziert):

$T := \pi_{1,\ldots,r-s}(R)$	alle Anfangsstücke
$T \times S$	diese kombiniert mit allen Verlängerungen aus S
$(T \times S) \backslash R$	davon nur solche, die nicht in R sind
$V := \pi_{1,\ldots,r-s}((T \times S) \backslash R)$	davon die Anfangsstücke
$T \backslash V$	davon das Komplement

6.6 Relationenkalkül

Ausdrücke in der Relationenalgebra spezifizieren konstruktiv, wie das Ergebnis der Anfrage zu berechnen ist. Demgegenüber ist der *Relationenkalkül* stärker *deklarativ* orientiert. Er beruht auf dem mathematischen Prädikatenkalkül erster Stufe, der quantifizierte Variablen zuläßt. Es gibt zwei unterschiedliche, aber gleichmächtige Ausprägungen des Relationenkalküls:

- Der relationale Tupelkalkül
- Der relationale Domänenkalkül

6.7 Der relationale Tupelkalkül

Im *relationalen Tupelkalkül* hat eine Anfrage die generische Form

$$\{t \mid P(t)\}$$

wobei t eine sogenannte Tupelvariable ist und P ist ein Prädikat, das erfüllt sein muß, damit t in das Ergebnis aufgenommen werden kann. Das Prädikat P wird formuliert unter Verwendung von $\vee, \wedge, \neg, \forall, \exists, \Rightarrow$.

Alle C4-Professoren:

$$\{p \mid p \in Professoren \wedge p.Rang = 'C4'\}$$

Alle Professorennamen zusammen mit den Personalnummern ihrer Assistenten:

$$\{[p.Name, a.PersNr] \mid p \in Professoren \wedge a \in Assistenten \wedge p.PersNr = a.Boss\}$$

Alle diejenigen Studenten, die sämtliche vierstündigen Vorlesungen gehört haben:

$$\{s \mid s \in Studenten \wedge \forall v \in Vorlesungen(v.SWS = 4 \Rightarrow$$

$$\exists h \in hören(h.VorlNr = v.VorlNr \wedge h.MatrNr = s.MatrNr))\}$$

Für die Äquivalenz des Tupelkalküls mit der Relationenalgebra ist es wichtig, sich auf sogenannte *sichere* Ausdrücke zu beschränken, d.h. Ausdrücke, deren Ergebnis wiederum eine Teilmenge der Domäne ist. Zum Beispiel ist der Ausdruck

$$\{n \mid \neg(n \in Professoren)\}$$

nicht sicher, da er unendlich viele Tupel enthält, die nicht in der Relation *Professoren* enthalten sind.

6.8 Der relationale Domänenkalkül

Im *relationalen Domänenkalkül* werden Variable nicht an Tupel, sondern an Domänen, d.h. Wertemengen von Attributen, gebunden. Eine Anfrage hat folgende generische Struktur:

$$\{[v_1, v_2, \ldots, v_n] \mid P(v_1, v_2, \ldots, v_n)\}$$

Hierbei sind die v_i Domänenvariablen, die einen Attributwert repräsentieren. P ist eine Formel der Prädikatenlogik 1. Stufe mit den freien Variablen $v_1, v_2, \ldots, v_n$.

Join-Bedingungen können implizit durch die Verwendung derselben Domänenvariable spezifiziert werden. Beispiel:

Alle Professorennamen zusammen mit den Personalnummern ihrer Assistenten:

$$\{[n, a] \mid \exists p, r, t([p, n, r, t] \in Professoren$$
$$\wedge \exists v, w([a, v, w, p] \in Assistenten))\}$$

Wegen des Existenz- und Allquantors ist die Definition des *sicheren Ausdruckes* etwas aufwendiger als beim Tupelkalkül. Da sich diese Quantoren beim Tupelkalkül immer auf Tupel einer vorhandenen Relation bezogen, war automatisch sichergestellt, daß das Ergebnis eine endliche Menge war.

6.9 Query by Example

Query-by-Example (QBE) beruht auf dem relationalen Domänenkalkül und erwartet vom Benutzer das beispielhafte Ausfüllen eines Tabellenskeletts.

Liste alle Vorlesungen von Sokrates mit mehr als 3 SWS:

Vorlesungen	VorlNr	Titel	SWS	gelesenVon
		p._t	> 3	Sokrates

Die Spalten eines Formulars enthalten Variablen, Konstanten, Bedingungen und Kommandos. Variablen beginnen mit einem Unterstrich (_), Konstanten haben keinen Präfix. Der Druckbefehl **p._t** veranlaßt die Ausgabe von _t.

Im Domänenkalkül lautet diese Anfrage

$$\{[t] | \exists v, s, r([v, t, s, r] \in \ Vorlesungen \wedge s > 3)\}$$

Ein Join wird durch die Bindung einer Variablen an mehrere Spalten möglich:

Liste alle Professoren, die Logik lesen:

Vorlesungen	VorlNr	Titel	SWS	gelesenVon
		Logik		_x

Professoren	PersNr	Name	Rang	Raum
	_x	**p.**_n		

Über eine *condition box* wird das Einhalten von Bedingungen erzwungen:

Liste alle Studenten, die in einem höheren Semester sind als Feuerbach:

Studenten	MatrNr	Name	Semester
		p._s	_a
		Feuerbach	_b

conditions
_a > _b

Das Kommando zur Gruppierung lautet **g.** , hierdurch werden alle Tupel einer Relation zusammengefaßt (gruppiert), die bezüglich eines Attributes gleiche Werte haben. Innerhalb einer Gruppe kann dann über Aggregatfunktionen summiert, minimiert, maximiert, der Durchschnitt gebildet oder einfach nur gezählt werden. Die Schlüsselworte dafür heißen **sum.**, **avg.**, **min.**, **max.** und **cnt.**. Standardmäßig werden innerhalb einer Gruppe die Duplikate eliminiert. Die Duplikateliminierung wird durch **all.** erreicht:

Liste die Summe der SWS der Professoren, die überwiegend lange Vorlesungen halten:

Vorlesungen	VorlNr	Titel	SWS	gelesenVon
			p.sum.all._x	p.g.

conditions
avg.all._x>2

Einfügen, Ändern und Löschen geschieht mit den Kommandos **i.**, **u.**, **d.**.

Füge neuen Studenten ein:

Studenten	MatrNr	Name	Semester
i.	4711	`Wacker`	5

Setze die Semesterzahlen von Feuerbach auf 3:

Studenten	MatrNr	Name	Semester
u.		`Feuerbach`	**u.3**

Entferne Sokrates und alle seine Vorlesungen:

Professoren	PersNr	Name	Rang	Raum
d.	_x	`Sokrates`		

Vorlesungen	VorlNr	Titel	SWS	gelesenVon
d.	_y			_x

hören	VorlNr	MatrNr
d.	_y	