

Das Relationale Modell

Das Relationale Modell

- Wurde in den Siebzigern von E.F.Codd entwickelt (er bekam den Turing Award dafür)
- Im Moment das am weitesten verbreitete Datenmodell
- Hat die hierarchischen und Netzwerk-Modelle abgelöst
- Das relationale Modell wird nach und nach um objekt-orientierte Konzepte erweitert

- Eine relationale Datenbank enthält eine Menge von Relationen
- Eine Relation R besteht aus zwei Bestandteilen:
 - Einer Instanz R : eine Tabelle mit Zeilen und Spalten; der aktuelle Inhalt der Relation
 - Einem Schema $\mathcal{R}$: spezifiziert den Namen der Relation und die Namen und Datentypen der Spalten; legt die Struktur der Relation fest

- Angenommen wir speichern die Daten von Studenten in der Relation Studenten:

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
...	...	...

- Jede Zeile wird *Tupel* genannt
- Jede Spalte wird *Attribut* genannt

- Studenten hat drei Attribute: MatrNr, Name, and Semester
- Assoziiert mit jedem Attribut ist eine Domäne (Wertebereich)
 - $D_{MatrNr} = \text{integer}$
 - $D_{Name} = \text{string}$
 - $D_{Semester} = \text{integer}$
- Komplettes Schema sieht so aus:

```
Studenten(MatrNr:integer,  
          Name:string,  
          Semester:integer)
```

- Eine Relation (Instanz) ist eine Untermenge des kartesischen Produkts der Domänen
- $R \subseteq D_1 \times D_2 \times \dots \times D_n$
- Beispiel:
Studenten $\subseteq D_{MatrNr} \times D_{Name} \times D_{Semester} =$
Studenten $\subseteq \text{integer} \times \text{string} \times \text{integer}$

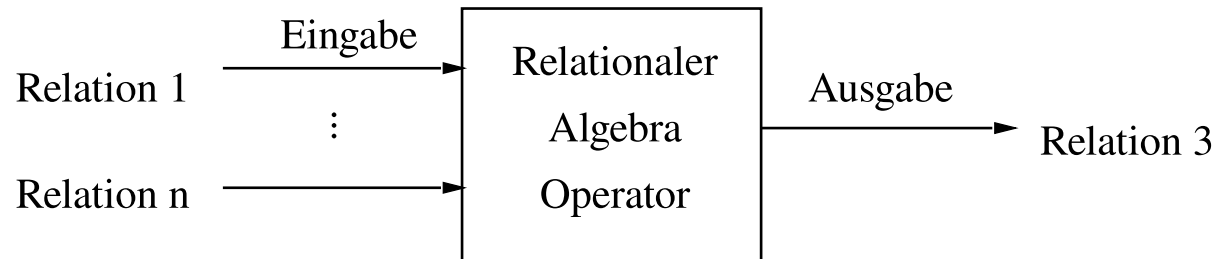
- Die Größe einer Relation (Anzahl Tupel) wird *Kardinalität* der Relation genannt
- Die minimale Menge von Attributen deren Werte ein Tupel eindeutig identifiziert heißt *Schlüssel*
- Der *Primärschlüssel* wird unterstrichen:

Studenten(MatrNr, Name, Semester)

- Bisher wurde nur die Struktur der Daten beschrieben, um Informationen aus der Datenbank abzufragen benötigt man eine Anfragesprache
- Wir behandeln zunächst
 - die relationale Algebra
 - den Relationenkalkül

- Der Relationenkalkül ist
 - eine rein deklarative Sprache
 - ursprünglich mit dem relationalen Modell entwickelt worden, bildet die Grundlage für SQL
- Die relationale Algebra
 - ist stärker prozedural orientiert
 - Eine Variante davon wird auf der physischen Ebene eingesetzt, um Anfragepläne zu bauen (DBMS selbst arbeitet nicht deklarativ)

- Alle Operatoren der relationalen Algebra sind mengenorientiert und abgeschlossen:
 - Jeder Operator bekommt als Eingabe eine (oder mehrere) Mengen von Tupeln ...
 - ...und gibt eine Menge von Tupeln aus



- Hat folgende Syntax: $\pi_{A_1, \dots, A_n}(R)$
- Wählt die Attribute (Spalten) $A_1, \dots, A_n$ aus der Relation R aus
- Filtert alle anderen Attribute heraus

Anfrage: “Gib die Namen aller Studenten aus”

$\pi_{Name}(\text{Studenten})$

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

Name
Xenokrates
Jonas
Fichte
Aristoxenos
Schopenhauer
Carnap
Theophrastos
Feuerbach

- Was passiert mit Duplikaten?
- Das relationale Modell ist mengenorientiert (Relation ist eine Menge von Tupeln)
- Duplikate werden eliminiert

Anfrage: “Gib die Semester aller Studenten aus”

$\pi_{Semester}(Studenten)$

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

Semester

18

12

10

8

6

3

2

- Syntax: $\sigma_p(R)$
- Wählt alle Tupel aus R aus, die das Prädikat p erfüllen
- Prädikate können mit logischen Operatoren kombiniert werden: $\wedge, \vee, \neg$
- Als Vergleichsoperatoren sind die üblichen Operatoren zugelassen: $=, <, \leq, >, \geq$

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

Anfrage: “Gib alle Studenten mit einer MatrNr kleiner als 26000 und einem Semester größer als 8 aus”

$\sigma_{MatrNr < 26000 \wedge Semester > 8}(\text{Studenten})$

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12

- Bisher arbeiteten alle Operatoren auf nur einer Relation, was ist wenn Information aus mehreren Relationen benötigt wird?
- Zwei Relationen können mit dem Kreuzprodukt verbunden werden: $R_1 \times R_2$
- Wird auch kartesisches Produkt genannt

Vorlesungen		
VorlNr	...	gelesenVon
5041	...	2125
5049	...	2125
5001	...	2137
4630	...	2137

Professoren		
PersNr	Name	...
2125	Sokrates	...
2137	Kant	...

Anfrage: “Gib alle Vorlesungen und Professoren aus”
Vorlesungen $\times$ Professoren

VorlNr	...	gelesenVon	PersNr	Name	...
5041	...	2125	2125	Sokrates	...
5041	...	2125	2137	Kant	...
5049	...	2125	2125	Sokrates	...
5049	...	2125	2137	Kant	...
5001	...	2137	2125	Sokrates	...
5001	...	2137	2137	Kant	...
4630	...	2137	2125	Sokrates	...
4630	...	2137	2137	Kant	...

Viele dieser Kombinationen machen keinen Sinn!

- Unsinnige Kombinationen des Kreuzprodukts kann man mit einer nachgeschalteten Selektion ausfiltern
- Für obige Anfrage:
 $\sigma_{gelesenVon=PersNr}(\text{Vorlesungen} \times \text{Professoren})$
- Da dies sehr häufig vorkommt, existiert ein eigener Operator dafür: ein Joinoperator
- $R_1 \bowtie_{R_1.A_i=R_2.A_j} R_2 = \sigma_{R_1.A_i=R_2.A_j}(R_1 \times R_2)$
- Joins sind effizienter implementierbar als Kreuzprodukte

Anfrage: “Gib alle Vorlesungen zusammen mit den zuständigen Professoren aus “
Vorlesungen $\bowtie_{gelesenVon=PersNr}$ Professoren

VorlNr	...	gelesenVon	PersNr	Name	...
5041	...	2125	2125	Sokrates	...
5049	...	2125	2125	Sokrates	...
5001	...	2137	2137	Kant	...
4630	...	2137	2137	Kant	...

- Alle Attributnamen müssen eindeutig sein, deswegen muß man auf gleichnamige Attribute in verschiedenen Relationen aufpassen

- Beispiel:

Assistenten(PersNr, Name, Fachgebiet, Boss)

Professoren(PersNr, Name, Rang, Raum)

$\text{Assistenten} \bowtie_{\text{Boss}=\text{PersNr}} \text{Professoren}$

- Hier benutzt man Umbenennungsoperator ρ :

$\rho_{APersNr \leftarrow PersNr, AName \leftarrow Name}(\text{Assistenten})$

$\bowtie_{\text{Boss}=PPersNr}$

$\rho_{PPersNr \leftarrow PersNr, PName \leftarrow Name}(\text{Professoren})$

- Joinoperatoren werden nach ihrem Joinprädikat klassifiziert
 - Theta-Join (θ -Join): die allgemeinste Art, das Joinprädikat darf beliebige Vergleichsoperatoren enthalten: $=, \neq, <, \leq, >, \geq$
 - Equi-Join: das Joinprädikat darf nur auf Gleichheit ($=$) prüfen
 - Natürlicher Join: eine spezielle Art des Equi-Joins, die nur Attribute mit gleichen Namen vergleicht (und redundante Spalten wegprojiziert)

Anfrage: “Gib alle Vorlesungen zusammen mit den zuständigen Professoren aus “

$\rho_{PersNr \leftarrow gelesenVon}(\text{Vorlesungen}) \bowtie \text{Professoren}$
 (Joinprädikat wird nicht angegeben, ist implizit gegeben)

VorlNr	...	PersNr	Name	...
5041	...	2125	Sokrates	...
5049	...	2125	Sokrates	...
5001	...	2137	Kant	...
4630	...	2137	Kant	...

- In einem Algebraausdruck können beliebig viele Relationen gejoint werden
- Die Reihenfolge in der dies gemacht wird spielt für die Korrektheit des Ergebnisses keine Rolle (Joins sind kommutativ und assoziativ)

- Studenten mit besuchten Vorlesungen verbinden

Studenten			Vorlesungen		
MatrNr	Name	...	VorlNr	Titel	...
26120	Fichte	...	4052	Logik	...
27550	Schopenhauer	...	5001	Grundzüge	...
...	...	...	...	...	...

hören	
MatrNr	VorlNr
26120	5001
27550	4052
27550	5001
...	...

- Falls ein Tupel keinen Joinpartner in anderen Relation findet, geht es verloren
- Im äußeren Join ($\bowtie$) bleiben diese Tupel erhalten

L			$\bowtie$	R			=	Resultat				
<i>A</i>	<i>B</i>	<i>C</i>		<i>C</i>	<i>D</i>	<i>E</i>		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁		<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁
<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂		<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂		<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂	–	–
								–	–	<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂

- Für den linken äußeren Join gilt dies nur für die Tupel aus der linken Relation

L			$\bowtie$	R			=	Resultat				
<i>A</i>	<i>B</i>	<i>C</i>		<i>C</i>	<i>D</i>	<i>E</i>		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁		<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁
<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂		<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂		<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂	–	–

- Für den rechten äußeren Join gilt dies nur für diejenigen aus der rechten Relation

L			$\bowtie$	R			$=$	Resultat				
<i>A</i>	<i>B</i>	<i>C</i>		<i>C</i>	<i>D</i>	<i>E</i>		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁		<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁
<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂		<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂		–	–	<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂

- Ein Semi-Join prüft die Joinbedingung, behält aber nur Tupel aus einer der beiden Relationen (die die Bedingung erfüllen)

L			$\bowtie$	R			=	Resultat		
<i>A</i>	<i>B</i>	<i>C</i>		<i>C</i>	<i>D</i>	<i>E</i>		<i>A</i>	<i>B</i>	<i>C</i>
<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁		<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁
<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂		<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂				

- Hier noch die zweite Variante

L			$\bowtie$	R			=	Resultat		
<i>A</i>	<i>B</i>	<i>C</i>		<i>C</i>	<i>D</i>	<i>E</i>		<i>C</i>	<i>D</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	<i>c</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁		<i>c</i> ₁	<i>d</i> ₁	<i>e</i> ₁
<i>a</i> ₂	<i>b</i> ₂	<i>c</i> ₂		<i>c</i> ₃	<i>d</i> ₂	<i>e</i> ₂				

- Ein Anti-Join prüft die Joinbedingung, behält aber nur Tupel aus einer der beiden Relationen (die die Bedingung **nicht** erfüllen)

L				R				Resultat		
A	B	C		C	D	E		A	B	C
a ₁	b ₁	c ₁	▷	c ₁	d ₁	e ₁	=	a ₂	b ₂	c ₂
a ₂	b ₂	c ₂		c ₃	d ₂	e ₂				

- Die üblichen Mengenoperationen wie Vereinigung, Schnitt und Differenz können auf Relationen angewendet werden
- Setzt allerdings voraus, daß beide Relationen das gleiche Schema haben:
 - Gleiche Anzahl von Attributen
 - Sich entsprechende Attribute haben gleichen Typ

Prof1	
PersNr	Name
2125	Sokrates
2126	Russel

Prof2	
PersNr	Name
2126	Russel
2137	Kant

Anfrage: “Vereinige beide Listen”
 $\text{Prof1} \cup \text{Prof2}$

PersNr	Name
2125	Sokrates
2126	Russel
2137	Kant

Prof1		Prof2	
PersNr	Name	PersNr	Name
2125	Sokrates	2126	Russel
2126	Russel	2137	Kant

Anfrage: “Welche Professoren sind auf beiden Listen?”

$\text{Prof1} \cap \text{Prof2}$

PersNr	Name
2126	Russel

Prof1	
PersNr	Name
2125	Sokrates
2126	Russel

Prof2	
PersNr	Name
2126	Russel
2137	Kant

Anfrage: “Welche Professoren sind auf der ersten aber nicht auf der zweiten Liste?”

$\text{Prof1} \setminus \text{Prof2}$

PersNr	Name
2125	Sokrates

- Wird für allquantifizierte Anfragen eingesetzt

- Formale Definition:

$$R_1 \div R_2 = \pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}(R_1) \setminus \pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}\left(\left(\pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}(R_1) \times R_2\right) \setminus R_1\right)$$

- ...oder:

$t \in R_1 \div R_2$, falls für jedes $v \in R_2$ ein $u \in R_1$ existiert, so dass gilt:

$$\begin{aligned} u.\mathcal{R}_2 &= v.\mathcal{R}_2 \\ u.(\mathcal{R}_1 \setminus \mathcal{R}_2) &= t.(\mathcal{R}_1 \setminus \mathcal{R}_2) \end{aligned}$$

- Voraussetzung $\mathcal{R}_2 \subseteq \mathcal{R}_1$

hören				
MatrNr	VorlNr			
25403	5022			
26120	5001			
27550	4052			
27550	5001			
28106	5041			
28106	5052			
28106	5216			
28106	5259			
29120	5001			
29120	5041		Vorl2	Vorl3
29120	5049	Vorl1	VorlNr	VorlNr
29555	5001	VorlNr	5001	5001
29555	5022	5001	5041	4052

Anfrage: “Liste die MatrNr *aller* Studenten die *alle* Vorlesungen in Liste 1,2,3 hören”

hören ÷ Vorl1

MatrNr
26120
27550
29120
29555

hören ÷ Vorl2

MatrNr
29120

hören ÷ Vorl3

MatrNr
27550

- Der Relationenkalkül ist stärker deklarativ orientiert, d.h. es werden Ergebnistupel ohne Herleitungsvorschrift beschrieben
- Basiert auf dem mathematischen Prädikatenkalkül erster Stufe
- Es gibt zwei verschiedene Varianten:
 - Relationaler Tupelkalkül
 - Relationaler Domänenkalkül

- Eine Anfrage im Relationenkalkül hat die Form $\{t \mid P(t)\}$
wobei t Tupelvariable und P Formel ist
- Beispiele:

Alle C4 Professoren

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

Studenten mit mind. einer Pythagoras-Vorlesung

$$\begin{aligned} \{s \mid & s \in \text{Studenten} \\ & \wedge \exists h \in \text{hören}(s.\text{MatrNr} = h.\text{MatrNr}) \\ & \wedge \exists v \in \text{Vorlesungen}(h.\text{VorlNr} = v.\text{VorlNr}) \\ & \wedge \exists p \in \text{Professoren}(p.\text{PersNr} = v.\text{gelesenVon}) \\ & \wedge p.\text{Name} = \text{'Pythagoras'})\} \end{aligned}$$

Atome

- $s \in R$, mit s Tupelvariable und R Relationenname
- $s.A \phi t.B$, mit s und t Tupelvariablen, A und B Attributnamen und ϕ Vergleichsoperator ($=, \neq, \leq, \dots$)
- $s.A \phi c$ mit c Konstante

Formeln

- Alle Atome sind Formeln
- Ist P Formel, so auch $\neg P$ und (P)
- Sind P_1 und P_2 Formeln, so auch $P_1 \wedge P_2$, $P_1 \vee P_2$ und $P_1 \Rightarrow P_2$
- Ist $P(t)$ Formel mit freier Variable t , so auch

$$\forall t \in R(P(t)) \quad \text{und} \quad \exists t \in R(P(t))$$

- $\{n \mid \neg(n \in \text{Professoren})\}$ z.B. ist nicht sicher, da das Ergebnis unendlich ist
- Bedingung für Sicherheit: Ergebnis des Ausdrucks muss Teilmenge der *Domäne* der Formel sein.
- Die Domäne einer Formel enthält
 - alle in der Formel vorkommenden Konstanten
 - alle Attributwerte von Relationen, die in der Formel referenziert werden

- Anfrage im Domänenkalkül hat die Form:
 $\{[v_1, v_2, \dots, v_n] \mid P(v_1, \dots, v_n)\}$
wobei $v_1, \dots, v_n$ Domänenvariablen sind und P Formel ist
- Beispiel:

MatrNr und Namen der Prüflinge von Sokrates

$$\begin{aligned} \{[m, n] \mid & \exists s([m, n, s] \in \text{Studenten}) \\ & \wedge \exists p, v, g([m, p, v, g] \in \text{prüfen}) \\ & \wedge \exists a, r, b([p, a, r, b] \in \text{Professoren}) \\ & \wedge a = \text{'Sokrates'})\} \end{aligned}$$

Atome

- $[w_1, w_2, \dots, w_m] \in R$, mit m -stelliger Relation R und Domänenvariablen $w_1, \dots, w_m$
- $x\phi y$, mit x und y Domänenvariablen, ϕ Vergleichsoperator
- $x\phi c$, mit Konstante c

Formeln

- Alle Atome sind Formeln
- Ist P Formel, so auch $\neg P$ und (P)
- Sind P_1 und P_2 Formeln, so auch $P_1 \vee P_2$, $P_1 \wedge P_2$ und $P_1 \Rightarrow P_2$
- Ist $P(v)$ Formel mit freier Variable v , so auch $\exists v(P(v))$ und $\forall v(P(v))$

- $\{[p, n, r, o] \mid \neg([p, n, r, o] \in \text{Professoren})\}$ unsicher
- $\{[x_1, x_2, \dots, x_n] \mid P(x_1, x_2, \dots, x_n)\}$ sicher, falls
 - Falls Tupel $[c_1, c_2, \dots, c_n]$ im Ergebnis, so muss jedes c_i ($1 \leq i \leq n$) in der Domäne von P enthalten sein.
 - Für jede Teilformel $\exists x(P_1(x))$ muss gelten, dass P_1 nur für Elemente aus der Domäne von P_1 erfüllbar – oder evtl. für gar keine.
 - Für jede Teilformel $\forall x(P_1(x))$ muss gelten, dass sie erfüllt ist, gdw. $P_1(x)$ für alle Werte der Domäne von P_1 erfüllt ist. Für alle anderen Werte muss sie sowieso erfüllt sein.

- Die drei Sprachen
 - relationale Algebra
 - relationaler Tupelkalkül, eingeschränkt auf sichere Ausdrücke
 - relationaler Domänenkalkül, eingeschränkt auf sichere Ausdrücke
- sind gleich mächtig

- Das relationale Modell ist das am weitesten verbreitete Datenmodell in heutigen DBMS
- In diesem Kapitel wurden die theoretischen Grundlagen des Modells erläutert:
 - Definition des Modells
 - Anfragesprachen