

Kapitel 2

Das Relationale Modell

Generelle Anmerkungen

- 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

Definitionen

- 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

Beispiel einer Instanz

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

MatrNr	Name	Geburtstag
1	Schmidt, Hans	1980-10-12
2	Müller, Anne	1982-07-30
3	Klein, Birgit	1981-03-24
...

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

Schema der Relation Student

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

```
Student(MatrnR:integer,  
        Name:string,  
        Geburtstag:date)
```

Formale Definition

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

Formale Definition(2)

- 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:

Student(MatrNr, Name, Geburtstag)

Anfragesprachen

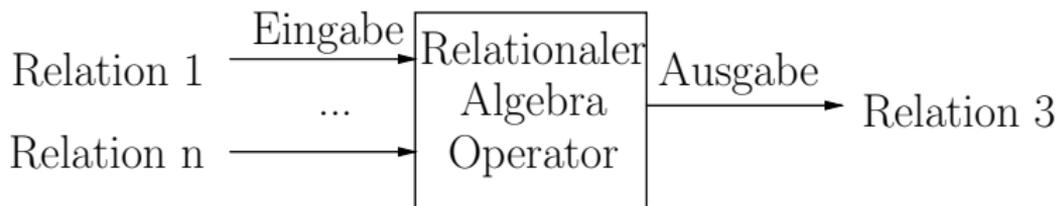
- Bisher wurde nur die Struktur der Daten beschrieben, um Informationen aus der Datenbank abzufragen benötigt man eine Anfragesprache
- Wir behandeln zunächst
 - ▶ die relationale Algebra
 - ▶ den Relationenkalkül

Anfragesprachen(2)

- 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)

Relationale Algebra

- 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



Projektion

- 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

Beispiel

Student		
MatrNr	Name	Geburtstag
1	Schmidt, Hans	1980-10-12
2	Müller, Anne	1982-07-30
3	Klein, Birgit	1981-03-24
4	Meier, Uwe	1982-07-30

Anfrage: "Gib die Namen aller Studenten aus"

	<u>Name</u>
$\pi_{Name}(\text{Student})$	Schmidt, Hans
	Müller, Anne
	Klein, Birgit
	Meier, Uwe

Projektion(2)

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

Beispiel

Student		
MatrNr	Name	Geburtstag
1	Schmidt, Hans	1980-10-12
2	Müller, Anne	1982-07-30
3	Klein, Birgit	1981-03-24
4	Meier, Uwe	1982-07-30

Anfrage: "Gib die Geburtsdaten aller Studenten aus"

	<u>Geburtstag</u>
$\pi_{\text{Geburtstag}}(\text{Student})$	1980-10-12
	1982-07-30
	1981-03-24

Selektion

- 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

Beispiel

Student		
MatrNr	Name	Geburtstag
1	Schmidt, Hans	1980-10-12
2	Müller, Anne	1982-07-30
3	Klein, Birgit	1981-03-24
4	Meier, Uwe	1982-07-30

Anfrage: "Gib alle Studenten mit einer MatrNr kleiner gleich 3 und einem Geburtstag nach 1.1.1981 aus"

$\sigma_{MatrNr \leq 3 \wedge Geburtstag > 1981-01-01}(\text{Student})$

MatrNr	Name	Geburtstag
2	Müller, Anne	1982-07-30
3	Klein, Birgit	1981-03-24

Kreuzprodukt

- 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

Beispiel

Vorlesung

Nr	...	ProfPersNr
1	...	1
2	...	1
3	...	2
4	...	2

Professor

PersNr	Name	...
1	Moerkotte	...
2	Kemper	...

Anfrage: "Gib alle Vorlesungen und Professoren aus"
Vorlesung \times Professor

Ergebnis

Nr	...	ProfPersNr	PersNr	Name	...
1	...	1	1	Moerkotte	...
1	...	1	2	Kemper	...
2	...	1	1	Moerkotte	...
2	...	1	2	Kemper	...
3	...	2	1	Moerkotte	...
3	...	2	2	Kemper	...
4	...	2	1	Moerkotte	...
4	...	2	2	Kemper	...

Viele dieser Kombinationen machen keinen Sinn!

Joins (Verbundoperatoren)

- Unsinnige Kombinationen des Kreuzprodukts kann man mit einer nachgeschalteten Selektion ausfiltern
- Für obige Anfrage:
 $\sigma_{ProfPersNr=PersNr}(Vorlesung \times Professor)$
- 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

Beispiel

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

Vorlesung $\bowtie_{ProfPersNr=PersNr}$ Professor

Nr	...	ProfPersNr	PersNr	Name	...
1	...	1	1	Moerkotte	...
2	...	1	1	Moerkotte	...
3	...	2	2	Kemper	...
4	...	2	2	Kemper	...

Joins(2)

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

- Beispiel:

Assistent(PersNr, Name, Fachgebiet, Boss)

Professor(PersNr, Name, ZimmerNr)

Assistent $\bowtie_{Boss=PersNr}$ Professor

- Hier benutzt man Umbenennungsoperator ρ :

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

$\bowtie_{Boss=PPersNr}$

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

Joins(3)

- 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)

Natürlicher Join

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

$\rho_{PersNr \leftarrow ProfPersNr}(Vorlesung) \bowtie Professor$

(Joinprädikat wird nicht angegeben, ist implizit gegeben)

Nr	...	PersNr	Name	...
1	...	1	Moerkotte	...
2	...	1	Moerkotte	...
3	...	2	Kemper	...
4	...	2	Kemper	...

Joins(4)

- 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)

Beispiel

- Studenten mit besuchten Vorlesungen verbinden

Student		
MatrNr	Name	...
1	Schmidt	...
2	Müller	...
...

Vorlesung		
Nr	Titel	...
1	Datenbanken	...
2	Netzwerke	...
...

besucht	
MatrNr	Nr
1	1
1	2
2	1
...	...

Weitere Joinarten

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

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

=

Resultat				
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

Linker Äußerer Join

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

L		
A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

 \bowtie

R		
C	D	E
c ₁	d ₁	e ₁
c ₃	d ₂	e ₂

 $=$

Resultat				
A	B	C	D	E
a ₁	b ₁	c ₁	d ₁	e ₁
a ₂	b ₂	c ₂	-	-

Rechter Äußerer Join

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

L		
A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

 \bowtie

R		
C	D	E
c ₁	d ₁	e ₁
c ₃	d ₂	e ₂

 $=$

Resultat				
A	B	C	D	E
a ₁	b ₁	c ₁	d ₁	e ₁
-	-	c ₃	d ₂	e ₂

Semi-Join

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

L		
A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

 \bowtie

R		
C	D	E
c ₁	d ₁	e ₁
c ₃	d ₂	e ₂

 =

Resultat		
A	B	C
a ₁	b ₁	c ₁

Semi-Join(2)

- Hier noch die zweite Variante

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

 \times

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

 $=$

Resultat		
C	D	E
c_1	d_1	e_1

Mengenoperationen

- 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

Vereinigung

PersNr	Name
1	Moerkotte
2	Kemper

PersNr	Name
2	Kemper
3	Weikum

Anfrage: "Vereinige beide Listen"

$\text{Prof1} \cup \text{Prof2}$

PersNr	Name
1	Moerkotte
2	Kemper
3	Weikum

Schnitt

PersNr	Name
1	Moerkotte
2	Kemper

PersNr	Name
2	Kemper
3	Weikum

Anfrage: "Welche Professoren sind auf beiden Listen?"

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

PersNr	Name
2	Kemper

Mengendifferenz

PersNr	Name
1	Moerkotte
2	Kemper

PersNr	Name
2	Kemper
3	Weikum

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

Prof1 \ Prof2

PersNr	Name
1	Moerkotte

Relationale Division

- Wird für allquantifizierte Anfragen eingesetzt

- Formale Definition:

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

- ... oder:

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

$$u.\mathcal{R}_2 = v.\mathcal{R}_2$$

$$u.(\mathcal{R}_1 \setminus \mathcal{R}_2) = t.(\mathcal{R}_1 \setminus \mathcal{R}_2)$$

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

Beispiel

besucht	
MatrNr	Nr
1	1
1	2
2	1
2	3
3	1
3	2
3	3

Vor1
Nr
1
2

Vor2
Nr
1
3

Vor3
Nr
1
2
3

Anfrage: "Liste die MatrNr *aller* Studenten die *alle* Vorlesungen in Liste 1,2,3 besuchen"

Ergebnis

besucht \div Vorl1

<u>MatrNr</u>
1
3

besucht \div Vorl2

<u>MatrNr</u>
2
3

besucht \div Vorl3

<u>MatrNr</u>
3

Relationenkalkül

- 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

Relationaler Tupelkalkül

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

$$\begin{array}{l} \text{Alle C4 Professoren} \\ \{p \mid p \in \text{Professoren} \wedge p.\text{Rang} = \text{'C4'}\} \end{array}$$

$$\begin{array}{l} \text{Studenten mit mind. einer Pythagoras-Vorlesung} \\ \{s \mid s \in \text{Studenten} \\ \wedge \exists h \in \text{hören}(s.\text{MatrNr} = h.\text{MatrNr} \\ \wedge \exists v \in \text{Vorlesungen}(h.\text{VorlNr} = v.\text{VorlNr} \\ \wedge \exists p \in \text{Professoren}(p.\text{PersNr} = v.\text{gelesenVon} \\ \wedge p.\text{Name} = \text{'Pythagoras'}))\} \end{array}$$

Formale Definition

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

Formale Definition(2)

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

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

Sicherheit

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

Relationaler Domänenkalkül

- 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

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

Formale Definition

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

Formale Definition(2)

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 $\forall v(P(v))$

Sicherheit

- $\{[p, n, r, o] \mid \neg([p, n, r, o] \in Professoren)\}$ 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.

Ausdruckskraft

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

Zusammenfassung

- 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