Das Relationale Modell

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 `Studenten`:

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

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

Schema der Relation Studenten

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

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

Formale Definition

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

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:

Studenten(<u>MatrNr</u>, Name, Semester)

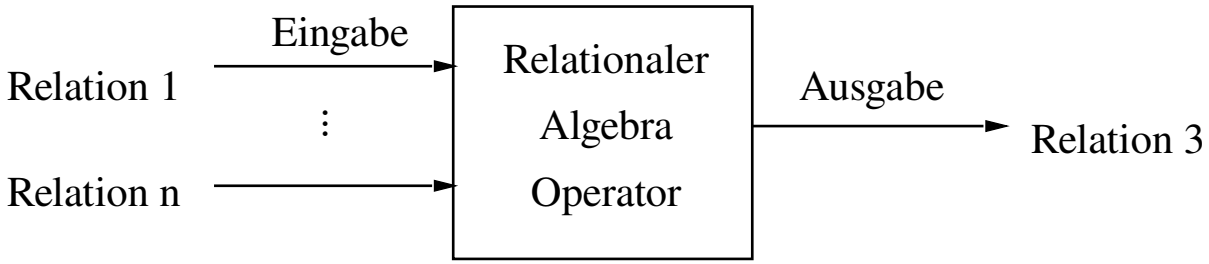
Anfragesprachen

- 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)

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

Beispiel

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 die Namen aller Studenten aus”

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

Name
Xenokrates
Jonas
Fichte
Aristoxenos
Schopenhauer
Carnap
Theophrastos
Feuerbach

Projektion(2)

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

Beispiel

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 die Semester aller Studenten aus”

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

Semester
18
12
10
8
6
3
2

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

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

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

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

Ergebnis

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!

Joins (Verbundoperatoren)

- 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 “

$$\text{Vorlesungen} \bowtie_{gelesenVon=PersNr} \text{Professoren}$$

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

Joins(2)

- 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)
  Assistenten $\bowtie_{Boss=PersNr}$ Professoren
- Hier benutzt man Umbenennungsoperator ρ:
  $\rho_{APersNr \leftarrow PersNr, AName \leftarrow Name}$(Assistenten)
  $\bowtie_{Boss=PPersNr}$
  $\rho_{PPersNr \leftarrow PersNr, PName \leftarrow Name}$(Professoren)

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)

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	...

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

Studenten

MatrNr	Name	...
26120	Fichte	...
27550	Schopenhauer	...
...	...	...

Vorlesungen

VorlNr	Titel	...
4052	Logik	...
5001	Grundzüge	...
...	...	...

hören

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

Weitere Joinarten

- Falls ein Tupel keinen Joinpartner in der 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

$\bowtie$

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	–	–
–	–	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

- 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

$\ltimes$

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

$\rtimes$

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

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

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\triangleright$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

$=$

Resultat		
A	B	C
a_2	b_2	c_2

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

Prof1

PersNr	Name
2125	Sokrates
2126	Russel

Prof2

PersNr	Name
2126	Russel
2137	Kant

Anfrage: “Vereinige beide Listen”

Prof1 $\cup$ Prof2

PersNr	Name
2125	Sokrates
2126	Russel
2137	Kant

Schnitt

Prof1

PersNr	Name
2125	Sokrates
2126	Russel

Prof2

PersNr	Name
2126	Russel
2137	Kant

Anfrage: “Welche Professoren sind auf beiden Listen?”

Prof1 ∩ Prof2

PersNr	Name
2126	Russel

Mengendifferenz

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

Relationale Division

- Wird für allquantifizierte Anfragen eingesetzt
- Formale Definition:
  $R_1 \div R_2 = \pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}(R_1) \setminus \pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}\Big(\big(\pi_{(\mathcal{R}_1 \setminus \mathcal{R}_2)}(R_1) \times R_2\big) \setminus R_1\Big)$
- ...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

hören

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

Vorl1

VorlNr
5001

Vorl2

VorlNr
5001
5041

Vorl3

VorlNr
5001
4052

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

Ergebnis

hören ÷ Vorl1

MatrNr
26120
27550
29120
29555

hören ÷ Vorl2

MatrNr
29120

hören ÷ Vorl3

MatrNr
27550

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

- 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

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

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 ($=, \neq, \leq, ...$)
- $s.A\phi c$ 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

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

- $\{n \mid \neg(n \in \text{Professoren})\}$ 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:
  $\{[v_1, v_2, ..., v_n] \mid P(v_1, ..., v_n)\}$
  wobei $v_1, ..., v_n$ Domänenvariablen sind und P Formel ist
- Beispiel:

MatrNr und Namen der Prüflinge von Sokrates

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

Formale Definition

Atome

- $[w_1, w_2, ..., w_m] \in R$, mit m-stelliger Relation R und Domänenvariablen $w_1, ..., w_m$
- $x \phi y$, 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 $\forall v(P(v))$

- $\{[p, n, r, o] \mid \neg([p, n, r, o] \in \text{Professoren})\}$ unsicher
- $\{[x_1, x_2, ..., x_n] \mid P(x_1, x_2, ..., x_n)\}$ sicher, falls
  - Falls Tupel $[c_1, c_2, ..., c_n]$ 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 ist – 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 Ausdrücke

  sind 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