Kapitel 3

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`
- Assoziert mit jedem Attribut ist eine Domäne (Wertebereich)
  - D_{MatrNr} = integer
  - D_{Name} = string
  - $D_{Geburtstag}$ = 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
- $R \subseteq D_1 \times D_2 \times \cdots \times D_n$
- Beispiel:
  Student $\subseteq D_{MatrNr} \times D_{Name} \times D_{Geburtstag} =$
  Student $\subseteq$ integer $\times$ string $\times$ 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(<u>MatrNr</u>, Name, Geburtstag)

Anfragesprachen

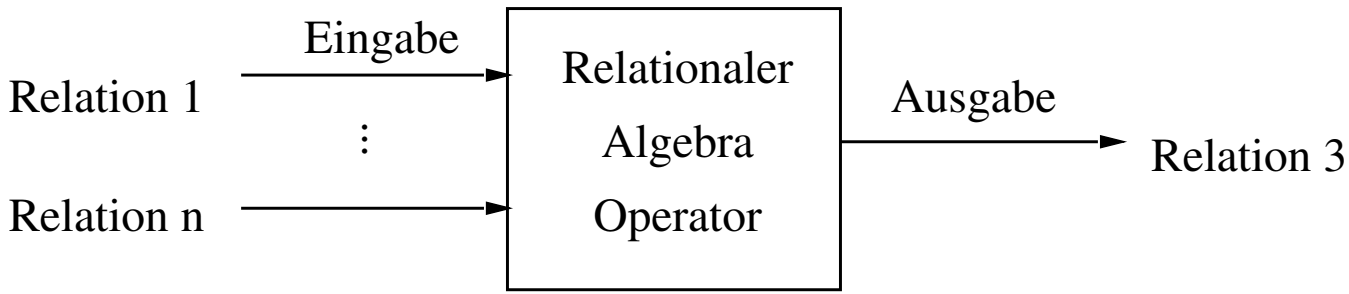
- Bisher wurde nur die Struktur der Daten beschrieben, um Informationen aus der Datenbank abzufragen benötigt man eine Anfragesprache
- Wir behandeln zunächst
  - die relationale Algebra
  - den Relationenkalkül

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: $\pi_{A_1,\ldots,A_n}(R)$
- Wählt die Attribute (Spalten) $A_1, \ldots, A_n$ 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"

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

Name
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"

$\pi_{Geburtstag}(\text{Student})$

Geburtstag
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:
  $\sigma_{ProfPersNr=PersNr}(\text{Vorlesung} \times \text{Professor})$
- 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 "

$$\text{Vorlesung} \bowtie_{ProfPersNr=PersNr} \text{Professor}$$

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:
  $\rho_{APersNr \leftarrow PersNr, AName \leftarrow Name}$(Assistent)
  $\bowtie_{Boss=PPersNr}$
  $\rho_{PPersNr \leftarrow PersNr, PName \leftarrow Name}$(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 "

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

(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 ($\mathbb{X}$) bleiben diese Tupel erhalten

L

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\mathbb{X}$

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

$\ltimes$

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

$\rtimes$

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

$\ltimes$

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

Anti-Join

- 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
1	Moerkotte
2	Kemper

Prof2

PersNr	Name
2	Kemper
3	Weikum

Anfrage: "Vereinige beide Listen"

Prof1 $\cup$ Prof2

PersNr	Name
1	Moerkotte
2	Kemper
3	Weikum

Schnitt

Prof1

PersNr	Name
1	Moerkotte
2	Kemper

Prof2

PersNr	Name
2	Kemper
3	Weikum

Anfrage: "Welche Professoren sind auf beiden Listen?"

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

PersNr	Name
2	Kemper

Mengendifferenz

Prof1

PersNr	Name
1	Moerkotte
2	Kemper

Prof2

PersNr	Name
2	Kemper
3	Weikum

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

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

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:

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

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

Beispiel

besucht

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

Vorl1

Nr
1
2

Vorl2

Nr
1
3

Vorl3

Nr
1
2
3

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

Ergebnis

besucht ÷ Vorl1

MatrNr
1
3

besucht ÷ Vorl2

MatrNr
2
3

besucht ÷ Vorl3

MatrNr
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:

Alle C4 Professoren

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

Studenten mit mind. einer Pythagoras-Vorlesung

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

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

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

Sicherheit

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

MatrNr und *Namen* der Prüflinge von Sokrates

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

Formale Definition

Atome

- $[w_1, w_2, \ldots, w_m] \in R$, mit m-stelliger Relation R und Domänenvariablen $w_1, \ldots, w_m$
- $x \, \phi \, y$, 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

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