FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Datenbanksysteme I

Prof. Dr. Viktor Leis

Professur für Datenbanken und Informationssysteme

Das Relationale Modell

Generelle Anmerkungen

- Wurde von E. F. Codd bei IBM entwickelt und 1970 publiziert (er bekam den Turing Award dafür)
- Das am weitesten verbreitete Datenmodell
- Hat die hierarchischen und Netzwerk-Modelle abgelöst

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` und `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

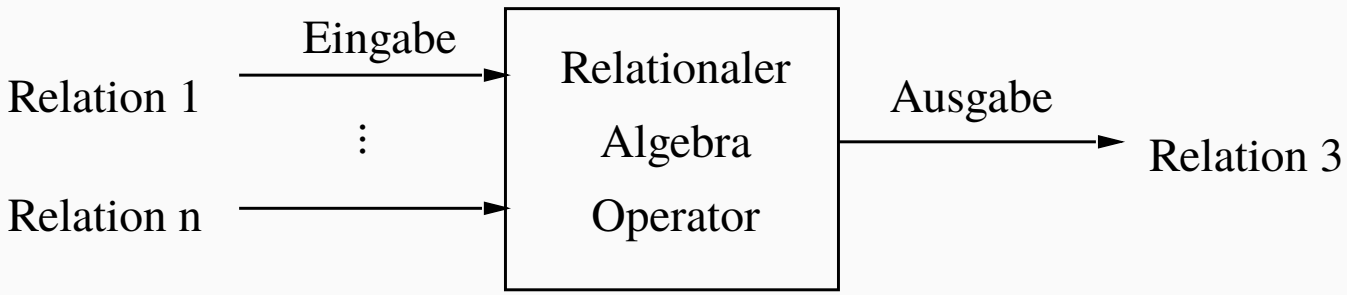
- Bisher wurde nur die Struktur der Daten beschrieben, um Informationen aus der Datenbank abzufragen, benötigt man eine Anfragesprache
- Wir fokussieren uns auf die relationale Algebra
- Es gibt auch noch den Relationenkalkül

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”

π_{Name}(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: der 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 ($\ltimes$) bleiben diese Tupel erhalten

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	–	–
–	–	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

$\ltimes$

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?”

Prof1 ∩ 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?”

Prof1 $\setminus$ Prof2

PersNr	Name
1	Moerkotte

Gruppierung und Aggregation

- Gruppierung von Tupeln mit gleichen Attributwerten
- Auf jede Gruppe können Aggregatfunktionen (z.B. count, sum, avg, min, max) angewendet werden

Anfrage: ”Anzahl Studis nach Semestern”

$\Gamma_{Semester;count(*)}(\text{Studenten})$

Semester	count(*)
7	4
12	1
5	12
3	5
1	1

Zusammenfassung

- Das relationale Modell ist das am weitesten verbreitete Datenmodell in heutigen DBMS
- Bildet die theoretische “Inspiration” hinter SQL
- In diesem Kapitel wurden die theoretischen Grundlagen des Modells erläutert:
  - Definition des Modells
  - Relationale Algebra