



FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Datenbanksysteme I

Datenbanken und Informationssysteme

Prof. Dr. Viktor Leis

WS 2019/2020

Professur für Datenbanken und Informationssysteme

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

- 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**
- Assoziiert 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)
```

- 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 $\subseteq \text{integer} \times \text{string} \times \text{date}$

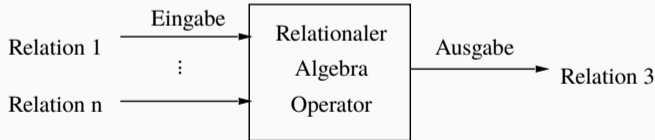
- Die Größe einer Relation (Anzahl Tupel) wird *Kardinalität* der Relation genannt
- Die minimale Menge von Attributen deren Werte ein Tupel eindeutig identifiziert heißt *Schlüssel*
- Der *Primärschlüssel* wird unterstrichen:

Student(MatrNr, Name, Geburtstag)

- 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

- 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

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"

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

<u>Name</u>
Schmidt, Hans
Müller, Anne
Klein, Birgit
Meier, Uwe

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

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"

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

Geburtstag

1980-10-12

1982-07-30

1981-03-24

- 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:
 $=, <, \leq, >, \geq$

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"

	MatrNr	Name	Geburtstag
$\sigma_{\text{MatrNr} \leq 3 \wedge \text{Geburtstag} > 1981-01-01}(\text{Student})$	2	Müller, Anne	1982-07-30
	3	Klein, Birgit	1981-03-24

- 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 × Professor

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: der Joinoperator
- $R_1 \bowtie_{R_1.A_j=R_2.A_j} R_2 = \sigma_{R_1.A_j=R_2.A_j}(R_1 \times R_2)$
- Joins sind effizienter implementierbar als Kreuzprodukte

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

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 :

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

$\bowtie_{Boss=PPersNr}$

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

- 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 "

$\rho_{PersNr \leftarrow ProfPersNr}(Vorlesung) \bowtie Professor$
(Joinprädikat wird nicht angegeben, ist implizit gegeben)

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

- 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

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

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

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

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

L			R			Resultat				
A	B	C	C	D	E	A	B	C	D	E
a_1	b_1	c_1	c_1	d_1	e_1	a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	c_3	d_2	e_2	a_2	b_2	c_2	-	-
						-	-	c_3	d_2	e_2

- 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	-	-

- 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

 \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

The diagram illustrates a semi-join operation between two relations, L and R. Relation L has columns A, B, and C, with rows (a₁, b₁, c₁) and (a₂, b₂, c₂). Relation R has columns C, D, and E, with rows (c₁, d₁, e₁) and (c₃, d₂, e₂). The join condition is C=L.C. The result table, labeled 'Resultat', contains only the row (a₁, b₁, c₁) from relation L, as it is the only tuple in L that has a matching value in R's column C.

- Hier noch die zweite Variante

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		
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

- 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
1	Moerkotte
2	Kemper

Prof2	
PersNr	Name
2	Kemper
3	Weikum

Anfrage: "Vereinige beide Listen"

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

PersNr	Name
1	Moerkotte
2	Kemper
3	Weikum

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

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 \ 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

- 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