



Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

Alice Rey, Maximilian {Bandle, Schüle}, Michael Jungmair

i3erdb@in.tum.de

Folien erstellt von Maximilian Bandle & Alexander Beischl



Organisatorisches

Disclaimer

Die Folien werden von der Übungsleitung allen Tutoren zur Verfügung gestellt.

Sollte es Unstimmigkeiten zu den Vorlesungsfolien von Prof. Kemper geben, so sind die Folien aus der Vorlesung ausschlaggebend.

Falls Ihr einen Fehler oder eine Unstimmigkeit findet, schreibt an i3erdb@in.tum.de mit Angabe der Foliennummer.

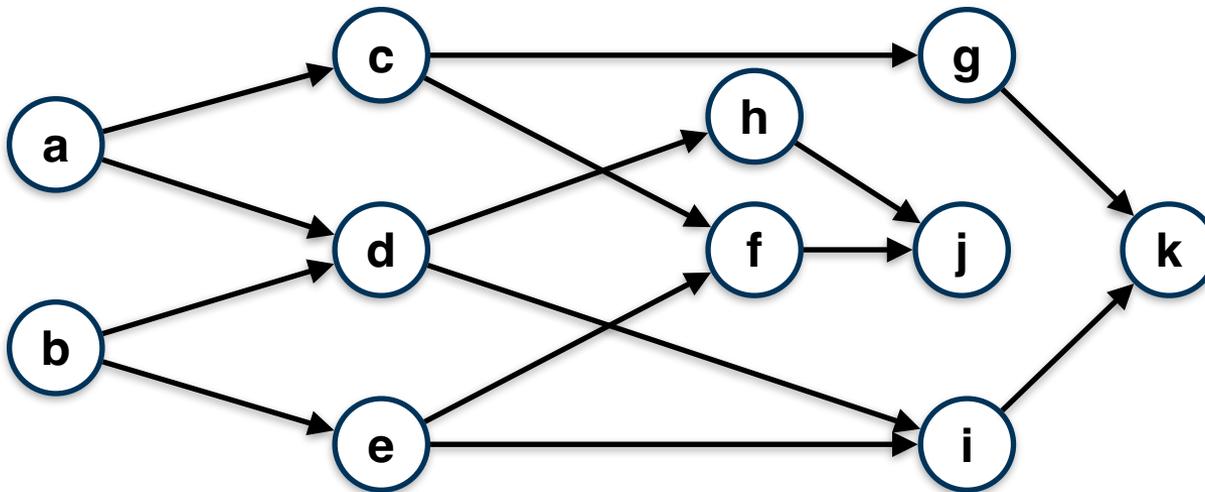
Aufgabe 1

Definieren Sie das Prädikat $sg(X,Y)$ das für “same generation” steht. Zwei Personen gehören zur selben Generation, wenn Sie mindestens je ein Elternteil haben, das derselben Generation angehört.

Verwenden Sie beispielsweise die folgende Ausprägung einer ElternKind Relation. Das erste Element ist hier das Kind, das Zweite ein Elternteil.

- Definieren Sie das Prädikat in Datalog.
- Demonstrieren Sie die naive Ausführung des Prädikats.
- Erläutern Sie das Vorgehen bei der seminaiven Auswertung.

```
parent(c,a).
parent(d,a).
parent(d,b).
parent(e,b).
parent(f,c).
parent(g,c).
parent(h,d).
parent(i,d).
parent(i,e).
parent(f,e).
parent(j,f).
parent(j,h).
parent(k,g).
parent(k,i).
```





Aufgabe 1

Naive Auswertung

$S := \{\};$

repeat

$S' := S;$

$S := \Pi_{X,Y} (P(Z, X) \bowtie_{X=Y} P(Z, Y));$

$S := S(X, Y) \cup \Pi_{X,Y} (P(X, Z) \bowtie_{X=Y} P(Y, Z));$

$S := S(X, Y) \cup \Pi_{X,Y} (P(X, U) \bowtie (S'(U, V) \bowtie P(Y, V)));$

until $S' = S$

output $S;$



Aufgabe 1

Semi-naive Auswertung

```
S := {}; ΔS := {};  
ΔS := ΠX,Y (P(Z, X) ⋈X=Y P(Z, Y));  
ΔS := ΔS(X, Y) ∪ ΠX,Y (P(X, Z) ⋈X=Y P(Y, Z));  
ΔS := ΔS(X, Y) ∪ ΠX,Y (P(X, U) ⋈ (S(U, V) ⋈ P(Y, V)));  
S := ΔS;  
repeat  
  ΔS' := ΔS;  
  ΔS := ΠX,Y (P(Z, X) ⋈X=Y ΔP(Z, Y)) !erste und*!  
    ∪ ΠX,Y (ΔP(Z, X) ⋈X=Y P(Z, Y))  
    ∪ ΠX,Y (P(X, Z) ⋈X=Y ΔP(Y, Z)) !zweite Regel*!  
    ∪ ΠX,Y (ΔP(X, Z) ⋈X=Y P(Y, Z)); !liefern ∅*!  
  ΔS := ΔS  
    ∪ ΠX,Y (ΔP(X, U) ⋈ (S(U, V) ⋈ P(Y, V))) !liefert ∅*!  
    ∪ ΠX,Y (P(X, U) ⋈ (ΔS'(U, V) ⋈ P(Y, V))) !kann ≠ ∅ sein*!  
    ∪ ΠX,Y (P(X, U) ⋈ (S(U, V) ⋈ ΔP(Y, V))); !liefert ∅*!  
  ΔS := ΔS - S; !entferne Tupel, die schon vorhanden waren*!  
  S := S ∪ ΔS;  
until ΔS = ∅
```



Deduktive Datenbanken

Naive Auswertung der Rekursion

parent \leq (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

verwandte(Vor, Nach) :- parent(Vor, Nach)

verwandte(Vor, Nach) :- verwandte(Vor, Mitte), parent(Mitte, Nach)

Schritt 0: (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

Schritt 1: (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122), (K1, K111), (K1, K112), (K1, K121), (K1, K122)

Schritt 2: (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122), (K1, K111), (K1, K112), (K1, K121), (K1, K122)



Deduktive Datenbanken

Semi-Naive Auswertung der Rekursion

parent \leq (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

verwandte(Vor, Nach) :- parent(Vor, Nach)

verwandte(Vor, Nach) :- verwandte(Vor, Mitte), parent(Mitte, Nach)

Schritt 0: (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

Deduktive Datenbanken

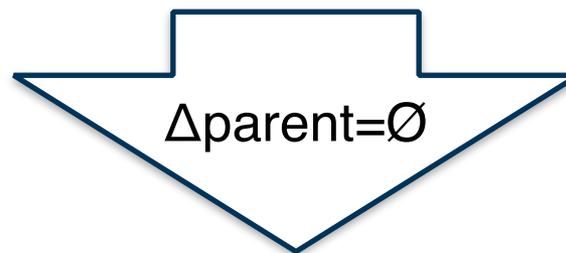
Semi-Naive Auswertung der Rekursion

Nur weiteres Auswerten der neu hinzugefügten Knoten (Δ der Relation)

$\text{verwandte}(\text{Vor}, \text{Nach}) :- \Delta\text{parent}(\text{Vor}, \text{Nach})$

$\text{verwandte}(\text{Vor}, \text{Nach}) :- \Delta\text{verwandte}(\text{Vor}, \text{Mitte}), \text{parent}(\text{Mitte}, \text{Nach})$

$\text{verwandte}(\text{Vor}, \text{Nach}) :- \text{verwandte}(\text{Vor}, \text{Mitte}), \Delta\text{parent}(\text{Mitte}, \text{Nach})$



$\text{verwandte}(\text{Vor}, \text{Nach}) :- \Delta\text{verwandte}(\text{Vor}, \text{Mitte}), \text{parent}(\text{Mitte}, \text{Nach})$



Deduktive Datenbanken

Semi-Naive Auswertung der Rekursion

parent \leq (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

verwandte(Vor, Nach) :- Δ verwandte(Vor, Mitte), parent(Mitte, Nach)

Schritt 0: (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121),
(K12,K122)

Schritt 1: (K1, K111), (K1, K112), (K1, K121), (K1, K122)

Schritt 2: \emptyset



Deduktive Datenbanken

Sichere Programme

- Durch Datalog-Regeln erzeugte Relationen müssen endlich sein
- Jede Variable, die in einer Regel vorkommt, muss eingeschränkt sein:
 - die Variable im Rumpf kommt in mindestens einem normalen Prädikat vor (nicht nur in Vergleichsprädikaten) oder
 - $X = c$ mit Konstante c existiert oder
 - ein Prädikat $X = Y$ vorkommt, und Y bereits eingeschränkt ist



Deduktive Datenbanken

Stratifizierte Programme

$$p(\dots) \text{ :- } q_1(\dots), \dots, \neg q_i(\dots), \dots, q_n(\dots).$$

- Regel p mit **negiertem** Prädikat q_i nur sinnvoll auswertbar, wenn q_i bereits materialisiert ist
- zuerst alle Regeln mit q_i auswerten $\Rightarrow q_i$ materialisiert
- nur möglich, wenn q_i nicht von p abhängig ist
- ➔ Abhängigkeitsgraph darf keine Pfade von p nach q_i enthalten
- muss für alle Regeln gelten



Aufgabe 2

Ist folgendes Datalog-Programm stratifiziert?

$$p(X, Y) \text{ :- } q_1(Y, Z), \neg q_2(Z, X), q_3(X, P).$$

$$q_2(Z, X) \text{ :- } q_4(Z, Y), q_3(Y, X).$$

$$q_4(Z, Y) \text{ :- } p(Z, X), q_3(X, Y).$$

Ist das Programm sicher – unter der Annahme, dass p, q_1, q_2, q_3, q_4 IDB- oder EDB-Prädikate sind?



Aufgabe 3

Gegeben sei folgende Faktenbasis, die einen direkten azyklischen Graphen (DAG) darstellt.

```
kante(1,2).  
kante(2,3).  
kante(3,4).  
kante(2,5).  
kante(5,3).
```

1. Geben Sie in Datalog ein Prädikat $\text{pfad}(V,N,L)$ an, das alle möglichen Pfade von V nach N mit Länge L ausgibt.
2. Geben Sie nun das Prädikat $\text{kuerzestePfade}(V,N,L)$ an, das pro Beginn V und Ziel N nur den kürzesten Pfad ausgibt.
3. Bestimmen Sie nun den längsten kürzesten Pfad $\text{laengsterkuerzesterPfad}(L)$.
4. Erstellen Sie in SQL eine rekursive Tabelle $\text{pfad}(V,N,L)$, die die Länge aller Pfade im DAG ausgibt.
5. Basierend auf der Tabelle $\text{pfad}(V,N,L)$, geben Sie die Länge des längsten kürzesten Pfades aus.

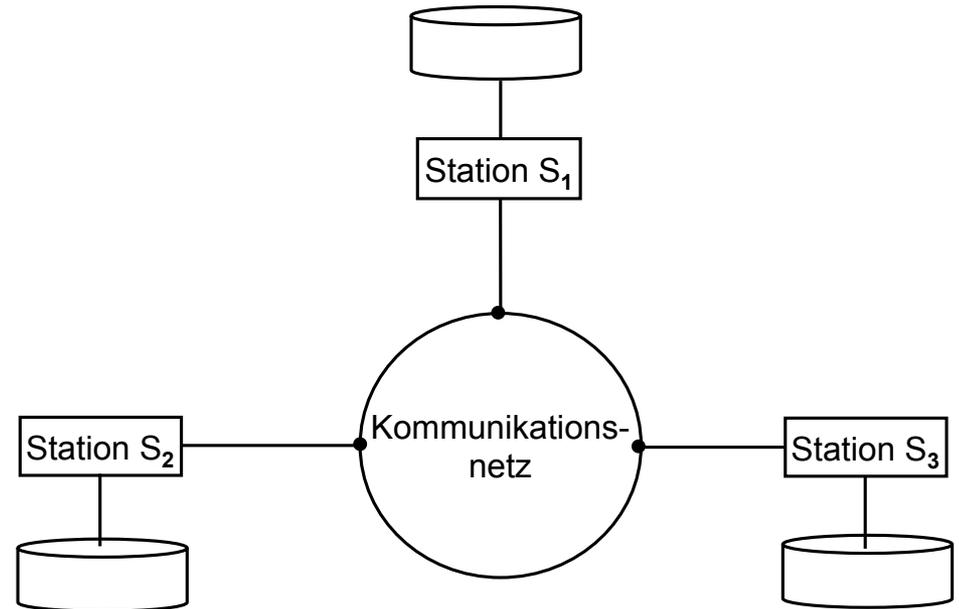


Verteilte Datenbanken

Verteilte Datenbanksysteme

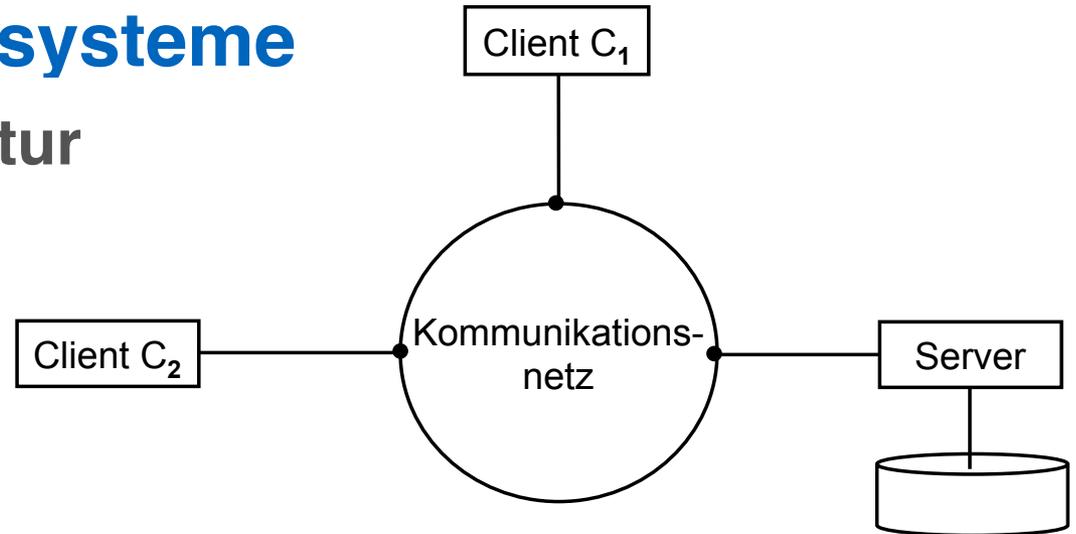
Motivation

- Globale Gesamtinformation auf Stationen (Sites) verteilt
- Daten werden von verteilten Datenbankverwaltungssystemen (VDBMS) verwaltet
- Stationen dürfen lokale Daten bearbeiten
- Kommunikationsverbindung (LAN, WAN, Telefonverbindungen)



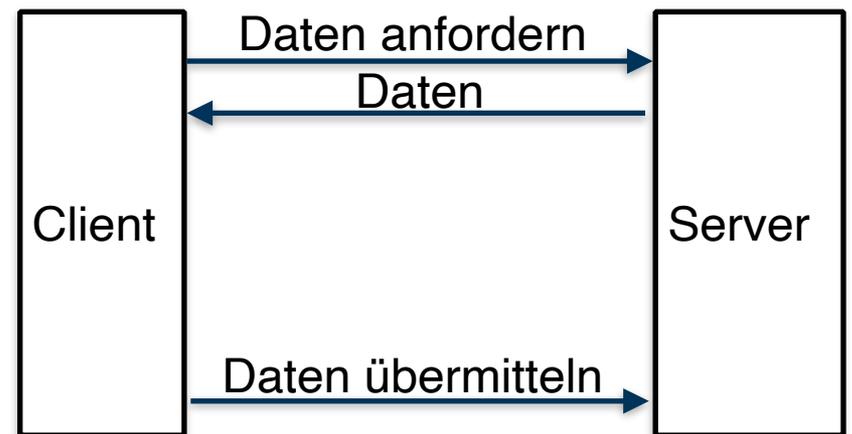
Verteilte Datenbanksysteme

Client-/Server-Architektur



- Degradiertes verteiltes Datenbanksystem
- Nur Server darf Daten abspeichern

Daten lokal
bearbeiten





Verteilte Datenbanksysteme

Fragmentierung

Horizontale Fragmentierung:

- Relation wird in disjunkte Tupelmengen geteilt

Vertikale Fragmentierung:

- Relation wird nach Attributen geteilt (durch Projektionen)
- Kombinierte Fragmentierung möglich



Aufgabe 4

Professoren						
PersNr	Name	Rang	Raum	Fakultät	Gehalt	Steuerklasse
2125	Sokrates	C4	226	Philosophie	85000	1
2126	Russel	C4	232	Philosophie	80000	3
2127	Kopernikus	C3	310	Physik	65000	5
2133	Popper	C3	52	Philosophie	68000	1
2134	Augustinus	C3	309	Theologie	55000	5
2136	Curie	C4	36	Physik	95000	3
2137	Kant	C4	7	Philosophie	98000	1

Gehen Sie von folgender kombinierter Fragmentierung der in Abbildung 1 dargestellten Relation *Professoren* aus:

1. Zuerst erfolgt eine vertikale Fragmentierung in

$$\text{ProfVerw} := \Pi_{\text{PersNr, Name, Gehalt, Steuerklasse}}(\text{Professoren})$$
$$\text{Profs} := \Pi_{\text{PersNr, Name, Rang, Raum, Fakultät}}(\text{Professoren})$$

2. Das Fragment Profs wird weiter horizontal fragmentiert in

$$\text{TheolProfs} := \sigma_{\text{Fakultät} = \text{'Theologie'}}(\text{Profs})$$
$$\text{PhysikProfs} := \sigma_{\text{Fakultät} = \text{'Physik'}}(\text{Profs})$$
$$\text{PhiloProfs} := \sigma_{\text{Fakultät} = \text{'Philosophie'}}(\text{Profs})$$

Übersetzen Sie aufbauend auf dieser Fragmentierung die folgende SQL-Anfrage in die kanonische Form.

```
select Name, Gehalt, Rang
from Professoren
where Gehalt > 80000;
```

Optimieren Sie diesen kanonischen Auswertungsplan durch Anwendung algebraischer Transformationsregeln (Äquivalenzen).



Verteilte Datenbanksysteme

Fragmentierung

Professoren						
PersNr	Name	Rang	Raum	Fakultät	Gehalt	Steuerklass
2125	Sokrates	C4	226	Philosophie	85000	1
2126	Russel	C4	232	Philosophie	80000	3
2127	Kopernikus	C3	310	Physik	65000	5
2133	Popper	C3	52	Philosophie	68000	1
2134	Augustinus	C3	309	Theologie	55000	5
2136	Curie	C4	36	Physik	95000	3
2137	Kant	C4	7	Philosophie	98000	1

1. Zuerst erfolgt eine vertikale Fragmentierung in

ProfVerw := $\Pi_{\text{PersNr, Name, Gehalt, Steuerklasse}}(\text{Professoren})$

Profs := $\Pi_{\text{PersNr, Name, Rang, Raum, Fakultät}}(\text{Professoren})$



Verteilte Datenbanksysteme

Fragmentierung Vertikal



ProfVerw				Profs				
PersNr	Name	Gehalt	Steuerklasse	PersNr	Name	Rang	Raum	Fakultät
2125	Sokrates	85000	1	2125	Sokrates	C4	226	Philosophie
2126	Russel	80000	3	2126	Russel	C4	232	Philosophie
2127	Kopernikus	65000	5	2127	Kopernikus	C3	310	Physik
2133	Popper	68000	1	2133	Popper	C3	52	Philosophie
2134	Augustinus	55000	5	2134	Augustinus	C3	309	Theologie
2136	Curie	95000	3	2136	Curie	C4	36	Physik
2137	Kant	98000	1	2137	Kant	C4	7	Philosophie

2. Das Fragment Profs wird weiter horizontal fragmentiert in

TheolProfs := $\sigma_{\text{Fakultät} = \text{'Theologie'}}(\text{Profs})$

PhysikProfs := $\sigma_{\text{Fakultät} = \text{'Physik'}}(\text{Profs})$

PhiloProfs := $\sigma_{\text{Fakultät} = \text{'Philosophie'}}(\text{Profs})$



Verteilte Datenbanksysteme

Fragmentierung Horizontal

Profs				
PersNr	Name	Rang	Raum	Fakultät
2125	Sokrates	C4	226	Philosophie
2126	Russel	C4	232	Philosophie
2127	Kopernikus	C3	310	Physik
2133	Popper	C3	52	Philosophie
2134	Augustinus	C3	309	Theologie
2136	Curie	C4	36	Physik
2137	Kant	C4	7	Philosophie

2. Das Fragment Profs wird weiter horizontal fragmentiert in

TheolProfs := $\sigma_{\text{Fakultät} = \text{'Theologie'}}(\text{Profs})$

PhysikProfs := $\sigma_{\text{Fakultät} = \text{'Physik'}}(\text{Profs})$

PhiloProfs := $\sigma_{\text{Fakultät} = \text{'Philosophie'}}(\text{Profs})$



Verteilte Datenbanksysteme

Fragmentierung Horizontal

TheolProfs				
PersNr	Name	Rang	Raum	Fakultät
2134	Augustinus	C3	309	Theologie

PhysikProfs				
PersNr	Name	Rang	Raum	Fakultät
2127	Kopernikus	C3	310	Physik
2136	Curie	C4	36	Physik

PhiloProfs				
PersNr	Name	Rang	Raum	Fakultät
2125	Sokrates	C4	226	Philosophie
2126	Russel	C4	232	Philosophie
2133	Popper	C3	52	Philosophie
2137	Kant	C4	7	Philosophie

Zerlegung
in disjunkte
Tupelmengen



Verteilte Datenbanksysteme

Fragmentierung Vertikal & Horizontal

ProfVerw			
PersNr	Name	Gehalt	Steuerklasse
2125	Sokrates	85000	1
2126	Russel	80000	3
2127	Kopernikus	65000	5
2133	Popper	68000	1
2134	Augustinus	55000	5
2136	Curie	95000	3
2137	Kant	98000	1

TheolProfs				
PersNr	Name	Rang	Raum	Fakultät
2134	Augustinus	C3	309	Theologie

PhysikProfs				
PersNr	Name	Rang	Raum	Fakultät
2127	Kopernikus	C3	310	Physik
2136	Curie	C4	36	Physik

PhiloProfs				
PersNr	Name	Rang	Raum	Fakultät
2125	Sokrates	C4	226	Philosophie
2126	Russel	C4	232	Philosophie
2133	Popper	C3	52	Philosophie
2137	Kant	C4	7	Philosophie

Übersetzen Sie aufbauend auf dieser Fragmentierung die folgende SQL-Anfrage in die kanonische Form.

```
select Name, Gehalt, Rang
from Professoren
where Gehalt > 80000;
```



Gruppenaufgabe 5

Schreiben Sie zu dem U-Bahn-Netz-Beispiel auf der Datalog Seite (unter Examples) folgende Anfragen in Datalog:

1. Erstellen Sie den Stationsplan für den U-Bahnhof Fröttmanning, der alle Station, die ohne umsteigen erreichbar sind, auflistet.
2. Erstellen Sie für Garching-Forschungszentrum einen Plan, der alle erreichbaren Stationen, die minimale Anzahl an Umstiegen und Stops auflistet. Beschreiben Sie Ihren Ansatz ausführlich.



Fragen?