Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

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

Bleiben wir bei dem bekannten Universitätsschema:

```
Assistenten(PersNr, Name, Fachgebiet, Boss)
hoeren(MatrNr, VorlNr)
pruefen(MatrNr,VorlNr, PersNr, Note)
Vorlesungen(VorlNr, Titel, SWS, gelesenVon)
Professoren(PersNr,Name,Rang, Raum)
voraussetzen(Vorg,Nachf)
Studenten(MatrNr, Name, Semester)
```

Formulieren Sie folgende Anfragen in Datalog und testen Sie sie:

a) Geben Sie alle *Professoren* an, die mindestens eine Prüfung abgehalten haben.

b) Übersetzen Sie folgenden Ausdruck des Domänenkalküls in Datalog. Machen Sie sich der Bedeutung des Ausdrucks bewusst.

$$\{[t] \mid \exists\, v,s,g([v,t,s,g] \in \text{Vorlesungen} \wedge \exists\, v2([v,v2] \in \text{voraussetzen} \wedge \exists\, s2,g2([v2,\text{`Wissenschaftstheorie'},s2,g2] \in \text{Vorlesungen})))\}$$

c) Joinen Sie nachfolgende Datalog-Anfrage so, dass Titel ausgegeben werden. Was bedeutet diese Anfrage?

```
geschwisterVL(N1,N2):-voraussetzen(V,N1),voraussetzen(V,N2), N1<N2.
nahverwandtVL(N1,N2):-geschwisterVL(N1,N2).
nahverwandtVL(N1,N2):-geschwisterVL(M1,M2),voraussetzen(M1,N1),
                                    voraussetzen(M2,N2).
```

Aufgabe 2

Geben Sie Datalog Regeln an, die Studenten (Namen angeben) finden, die von einem Prüfer geprüft worden, der selbst nicht die geprüfte Vorlesung gehalten hat. Das korrekte Ergebnis für diese Anfrage ist `Russels` Prüfling, `Carnap`. Führen Sie die Anfrage im Datalog Tool aus!

Aufgabe 3

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.

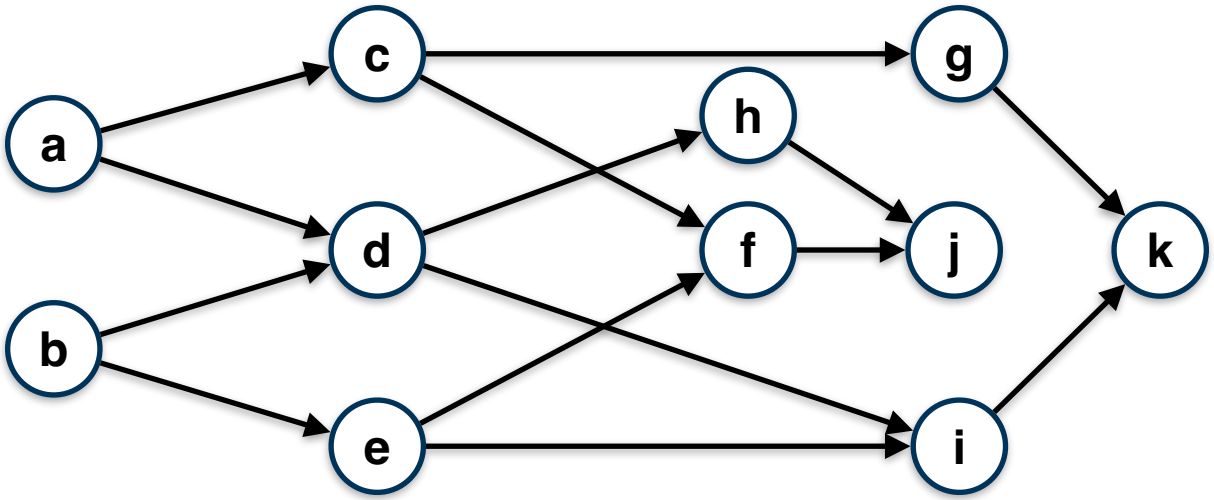
Verwenden Sie beispielsweise die folgende Ausprägung einer ElternKind Relation. Das erste Element ist hier das Kind, das Zweite ein Elternteil.

a) Definieren Sie das Prädikat in Datalog.

b) Demonstrieren Sie die naive Ausführung des Prädikats.

c) 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 3

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 3

Semi-naive Auswertung

$S := \{\}; \Delta S := \{\};$
$\Delta S := \Pi_{X,Y}(P(Z,X) \bowtie_{X=Y} P(Z,Y));$
$\Delta S := \Delta S(X,Y) \cup \Pi_{X,Y}(P(X,Z) \bowtie_{X=Y} P(Y,Z));$
$\Delta S := \Delta S(X,Y) \cup \Pi_{X,Y}(P(X,U) \bowtie (S(U,V) \bowtie P(Y,V)));$
$S := \Delta S;$
repeat
  $\Delta S' := \Delta S;$
  $\Delta S := \Pi_{X,Y}(P(Z,X) \bowtie_{X=Y} \Delta P(Z,Y))$ /*erste und*/;
    $\cup\, \Pi_{X,Y}(\Delta P(Z,X) \bowtie_{X=Y} P(Z,Y))$
    $\cup\, \Pi_{X,Y}(P(X,Z) \bowtie_{X=Y} \Delta P(Y,Z))$ /*zweite Regel*/
    $\cup\, \Pi_{X,Y}(\Delta P(X,Z) \bowtie_{X=Y} P(Y,Z));$ /*liefern $\emptyset$*/
  $\Delta S := \Delta S$
    $\cup\, \Pi_{X,Y}(\Delta P(X,U) \bowtie (S(U,V) \bowtie P(Y,V)))$ /*liefert $\emptyset$*/
    $\cup\, \Pi_{X,Y}(P(X,U) \bowtie (\Delta S'(U,V) \bowtie P(Y,V)))$ /*kann $\neq \emptyset$ sein*/
    $\cup\, \Pi_{X,Y}(P(X,U) \bowtie (S(U,V) \bowtie \Delta P(Y,V)));$ /*liefert $\emptyset$*/
  $\Delta S := \Delta S - S;$ /*entferne Tupel, die schon vorhanden waren*/
  $S := S \cup \Delta S;$
until $\Delta S = \emptyset$

Deduktive Datenbanken
Naive Auswertung der Rekursion

parent <= (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 <= (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121), (K12,K122)

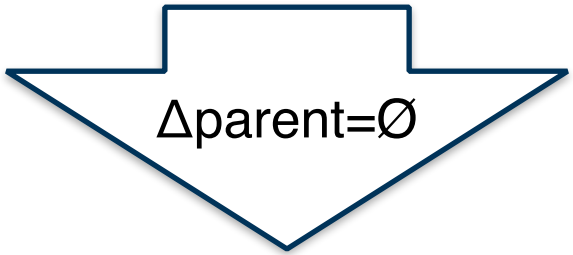
verwandte(Vor, Nach) :- parent(Vor, Nach)
verwandte(Vor, Nach) :- verwandte(Vor, Mitte), parent(Mitte, Nach)

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)

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



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

Deduktive Datenbanken
Semi-Naive Auswertung der Rekursion

parent <= (K1,K11), (K1,K12), (K11,K111), (K11,K112), (K12,K121), (K12,K122)

verwandte(Vor, Nach) :- Δverwandte(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: Ø

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

$$p(...) :- q_1(...), ..., \neg q_i(...), \ldots, q_n(...).$$

- Regel p mit **negiertem** Prädikat q_i nur sinnvoll auswertbar, wenn q_i bereits materialisiert ist

- zuerst alle Regeln mit q_i auswerten => 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 4

Ist folgendes Datalog-Programm stratifiziert?

$$
\begin{aligned}
p(X,Y) &:- q_1(Y,Z), \neg q_2(Z,X), q_3(X,P).\\
q_2(Z,X) &:- q_4(Z,Y), q_3(Y,X).\\
q_4(Z,Y) &:- p(Z,X), q_3(X,Y).
\end{aligned}
$$

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

Gruppenaufgabe 5

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 `pfad(V,N,L)` an, dass alle möglichen Pfade von `V` nach `N` mit Länge `L` ausgibt.
2. Geben Sie nun das Prädikat `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 `laengsterkuerzesterPfad(L)`.
4. Erstellen Sie in SQL eine rekursive Tabelle `pfad(V,N,L)`, die die Länge aller Pfade im DAG ausgibt.
5. Basierend auf der Tabelle `pfad(V,N,L)`, geben Sie die Länge des längsten kürzesten Pfades aus.

Fragen?