TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe22

Alice Rey, Maximilian {Bandle, Schüle}, Michael Jungmair (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss22/impldb/

Blatt Nr. 05

Hinweise Die Datalogaufgaben können auf `https://datalog.db.in.tum.de/` getestet werden. Auf der Seite kann unter *examples* ein entsprechender Datensatz geladen werden. Die neuen IDB Regeln sollten am Ende der EDB definiert und dann im Query-Eingabefeld abgefragt werden.

Zusätzlich zu der in der Vorlesung vorgestellten Syntax hier noch eine Kurzübersicht der Vergleichsoperatoren: $X < Y, Y > X$ (kleiner, größer), $X =< Y, X >= Y$ (kleiner oder gleich, größer oder gleich), $X = Y, X \backslash = Y$ (gleich, ungleich), $not(pred(X, Y))$ (existiert nicht pred(X,Y)).

Hausaufgabe 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.

```
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).
```

a) Definieren Sie das Prädikat in Datalog.

```
sg(X,Y) = :- parent(Z,X),X=Y. % X als Elternteil
sg(X,Y) = :- parent(X,Z),X=Y. % X als Kind
% X,Y Kind von U und V, U und V gleiche Generation
sg(X,Y) = :- sg(U,V),parent(X,U),parent(Y,V).
```

b) Demonstrieren Sie die naive Ausführung des Prädikats.

c) Erläutern Sie das Vorgehen bei der seminaiven Auswertung.

Siehe Übungsbuch

Gruppenaufgabe 2

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?

Loesung: Vgl. Übungsbuch. Das Programm ist **nicht stratifizert**, aber **sicher**. Es ist nicht stratifiziert, weil q_2 von p abhängt, aber negiert in p vorkommt. Es ist sicher, weil alle Variablen in IDB- oder EDB-Prädikaten gebunden sind.

Zur Wiederholung: Eine Regel ist **sicher** gdw. alle Variablen **eingeschränkt** sind. Variable X ist in einer Regel **eingeschränkt**, falls sie im Rumpf enthalten ist und sie:

- in einem positiven Prädikat vorkommt (nicht Vergleichsprädikat),
- $X = c$ (Konstante) oder
- $X = Y$, wenn Y bereits nachgewiesen ist.

Jede Variable eines negierten Prädikates muss bereits eingeschränkt sein.[a][b]

Ein Datalog-Programm ist **stratifiziert**, wenn in einer Regel p alle negierten Prädikate $not(q_i)$ nicht von p abhängen (kein Zyklus) und alle positiven Prädikate vorher oder unabhängig von der Reihenfolge (wie bei Rekursion) ausgewertet werden. Formal ausgedrückt können wir jedem Prädikat eine sogenannte Stratifikationsnummer zuordnen. Negierte Prädikate müssen eine echt kleinere Stratifikationsnummer besitzen, positive Prädikate eine maximal so große Stratifikationsnummer wie das davon abhängige Prädikat.

[a] `https://www.dbis.informatik.uni-goettingen.de/Teaching/DB/db-datalog.pdf`
[b] `http://pages.cs.wisc.edu/~paris/cs784-s17/lectures/lecture9.pdf`

Hausaufgabe 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 `pfad(V,N,L)` an, dass alle möglichen Pfade von `V` nach `N` mit Länge `L` ausgibt.

   ```
   pfad(V,N,L) :- kante(V,N), L=1.
   pfad(V,N,L) :- pfad(V,X,LX), kante(X,N), L=LX+1.
   ```

2. Geben Sie nun das Prädikat `kuerzestePfade(V,N,L)` an, das pro Beginn `V` und Ziel `N` nur den kürzesten Pfad ausgibt.

```
langepfade(V,N,L2) :- pfad(V,N,L), pfad(V,N,L2), L<L2.
kuerzestePfade(V,N,L) :- pfad(V,N,L), not(langepfade(V,N,L)).
```

3. Bestimmen Sie nun den längsten kürzesten Pfad `laengsterkuerzesterPfad(L)`.

```
kurzekuerzestePfade(L) :- kuerzestePfade(_,_,L), kuerzestePfade(_,_,L2), L<L2.
laengsterkuerzesterPfad(L) :- kuerzestePfade(_,_,L), not(kurzekuerzestePfade(L)).
```

4. Erstellen Sie in SQL eine rekursive CTE `pfad(V,N,L)`, die die Länge aller Pfade im DAG ausgibt.

```
with recursive kante(V,N) as (values (1,2), (2,3), (3,4),(2,5),(5,3)),
  pfad(V,N,L) as ( select k.V, k.N, 1 from kante k union
   select k.V, p.N, p.L + 1 from kante k, pfad p where k.N = p.V)
```

5. Basierend auf `pfad(V,N,L)`, geben Sie die Länge des längsten kürzesten Pfades aus.

```
select max(min) from (select v,n,min(l) from pfad group by v,n) tmp;
```

Hausaufgabe 4

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

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

Abbildung 1: Beispielausprägung der um drei Attribute erweiterten Relation *Professoren*

1. Zuerst erfolgt eine vertikale Fragmentierung in

$$\begin{aligned} \text{ProfVerw} &:= \Pi_{PersNr,Name,Gehalt,Steuerklasse}(\text{Professoren}) \\ \text{Profs} &:= \Pi_{PersNr,Name,Rang,Raum,Fakultt}(\text{Professoren}) \end{aligned}$$

2. Das Fragment Profs wird weiter horizontal fragmentiert in

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

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

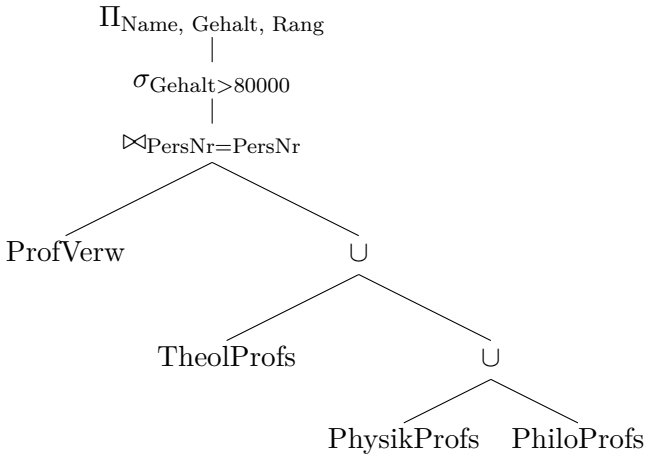
```
select Name, Gehalt Rang
from Professoren
where Gehalt > 80000;
```

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

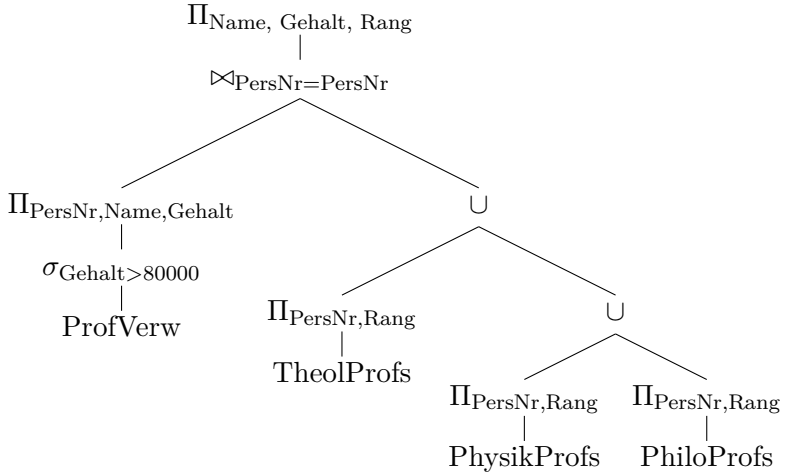
Vgl. Übungsbuch.

$$\Pi_{\text{Name, Gehalt, Rang}}(\Pi_{\text{PersNr,Name, Gehalt}}(\sigma_{\text{Gehalt} > 80000}(\text{ProfVerw}))\bowtie_{\text{PersNr=PersNr}} (\Pi_{\text{PersNr, Rang}}(\text{TheolProfs}) \cup \Pi_{\text{PersNr, Rang}}(\text{PhysikProfs}) \cup \Pi_{\text{PersNr, Rang}}(\text{PhiloProfs})))$$

Der Baum zum kanonischen Auswertungsplan sieht wie folgt aus:



In einem ersten Schritt verschiebt man die Selektion näher an die Datenquellen, die sich nur auf *ProfVerw* bezieht. Anschließend versuchen wir, die Zwischenergebnisse so klein wie möglich zu halten, indem wir zusätzliche Projektionen einfügen. Das Attribut *Name* ist redundant in beiden vertikalen Fragmenten enthalten und wird nicht benötigt.



Gruppenaufgabe (wird nicht in der Übung besprochen)

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 Stationen, die ohne Umstieg erreichbar sind, auflistet.

   ```
   bidirekt(A,B,L) :- direkt(A,B,L), A\=B.
   bidirekt(A,B,L) :- direkt(B,A,L), A\=B.
   bidirekt(A,B,L) :- bidirekt(A,X,L), direkt(X,B,L), A\=B.

   bidirekt(froettmanning,B,_)
   ```

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.

Vereinfachte Lösung (betrachten nur in Fahrtrichtung)

```
% Erreichbar naechster Stop
aufwand(A,B,L,S,U) :- direkt(A,B,L), S=0, U=0.
% Erreichbar auf gleicher Linie
aufwand(A,B,L,S,U) :- aufwand(A,C,L,SX,UX), direkt(C,B,L), S=SX+1, U=UX.
% Erreichbar durch umsteigen
aufwand(A,B,L,S,U) :- aufwand(A,C,LA,SX,UX), direkt(C,B,LB),
   S=SX+1, LA\=LB, L=LB, U=UX+1.
```

Lösung mit Richtungs- und Linienwechsel.

```
% Merke Richtung in die gefahren wird (R=vorwaerts oder rueckwaerts)
bdirekt(A,B,L,R) :- direkt(A,B,L), R=v.
bdirekt(A,B,L,R) :- direkt(B,A,L), R=r.

% Maximale Anzahl der Stops, ist noetig falls die Rekursion
% in einem Kreis im Graph festhaengt.
% Ohne Aggregation einfach 49 statt SMAX bei aufwand(...) einsetzen
smax(MAX):- count(direkt(_,_,_), MAX).

% Erreichbar naechster Stop
aufwand(A,B,L,R,S,U) :- bdirekt(A,B,L,R), S=1, U=0, A\=B.
% Erreichbar auf gleicher Linie
aufwand(A,B,L,R,S,U) :- aufwand(A,C,L,R,SX,UX), bdirekt(C,B,L,R),
   S=SX+1, U=UX, A\=B, smax(SMAX), S<SMAX.
% Erreichbar durch Umsteigen auf andere Linies.
% Richtungswechsel erlaubt.
aufwand(A,B,L,R,S,U) :- aufwand(A,C,LA,_,SX,UX), bdirekt(C,B,LB,R),
   S=SX+1, LA\=LB, L=LB, U=UX+1, A\=B, smax(SMAX), S<SMAX.
```

Im *aufwand* Prädikat ist ein Richtungswechsel ohne gleichzeitigen Linienwechsel nicht berücksichtigt. Dies ist auch nicht notwendig, da am Startpunkt durch *bdirekt* in beide Richtungen gestartet werden kann, und bei einem Linienwechsel dann auch jedesmal die Möglichkeit besteht, die Richtung frei zu wählen.
Wegen der Struktur der Linien im Beispielgraph (sie treffen sich nur am Sendlinger Tor) ist die Loesung des *aufwand* Prädikats schon jeweils die kürzeste Strecke. Bei einem Netz mit zwei Treffpunkten wären durch die Richtungswechsel Kreise möglich und das Minimum für jede Strecke müsste mit folgendem Prädikat gefunden werden.

```
minaufwand(A,B,S,U) :- aufwand(A,B,_,_,S,U),
   min(aufwand(A,B,_,_,ST,U), ST,S),
   min(aufwand(A,B,_,_,S,UM),UM,U).

minaufwand(garching_forschungszentrum,B,S,U)
```