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

Übung zur Vorlesung
Einsatz und Realisierung von Datenbanksystemen im SoSe16

Moritz Kaufmann (moritz.kaufmann@tum.de)
http://db.in.tum.de/teaching/ss16/impldb/

Blatt Nr. 04

Hinweise Die Datelogaufgaben können auf `http://datalog.db.in.tum.de/` getestet werden. Auf der Seite könnt Ihr dann unter *examples* einen entsprechenden Datensatz laden. An das Ende der EDB könnt ihr dann neue IDB Regeln definieren und diese dann in dem Query Eingabefeld abfragen. Zusätzlich zu dem in der Vorlesung vorgestellten Syntax hier noch eine kurz Übersicht der Vergleichsoperatoren: $X < Y, Y > X$ (kleiner, größer), $X =< Y, X >= Y$ (kleiner gleich, größer gleich), $X = Y, X\backslash = Y$ (gleich, ungleich), $not(pred(X, Y))$ (existiert nicht pred(X,Y)).

Hausaufgabe 1

Gegeben das folgende Schema der EDB[1]:

```
Product(maker, model, type).
PC(model, speed, ram, hd, price).
Laptop(model, speed, ram, hd, screen, price).
Printer(model, color, type, price).
```

Beantworten Sie in Datalog und testen Sie unter (`http://datalog.db.in.tum.de/`):

a) What PC models have a speed of at least 3.00 GHz?

```
fast_pc(X,Y,P) :- pc(X,Y,_,_,P), Y>3.0.
```

b) Which manufacturers make laptops with a hard disk (hd) of at least 100 GB?

```
manuf(M) :- laptop(P,_,_,D,_,_), D > 100, product(M,P,laptop).
```

c) Find the model number and price of products (of any type) made by manufacturer B.

```
b_prod(M,P) :- product(b, M, pc), pc(M,_,_,_,P).
b_prod(M,P) :- product(b, M, laptop), laptop(M,_,_,_,_,P).
b_prod(M,P) :- product(b, M, printer), printer(M,_,_,P).
```

d) Find the model numbers of all color laser printers.

```
printer(M,color,laser,_)
```

e) Find those manufacturers that sell Laptops, but not PC's.

```
laptop_manuf(M) :- product(M,_,laptop), not(product(M,_,pc)).
```

f) Find those hard-disk sizes that occur in two or more PC's.

```
pop_sizes(D) :- pc(M1,_,_,D,_), pc(M2,_,_,D,_), M1\=M2.
```

[1] Inspiriert von `http://people.inf.elte.hu/sila/DB1English/exercise06_products.pdf`.

g) Find those pairs of PC models that have both the same cpu speed and RAM. A pair should be listed only once, e.g., list (i,j) but not (j,i).

```
sim_pc(M1,M2) :- pc(M1,S,R,_,_), pc(M2,S,R,_,_), M1<M2.
```

Hausaufgabe 2 Gegeben sei die folgende Segler-Boots-Reservierung Datenbank:

```
%segler(SID,SNAME,EINSTUFUNG,ALTER)
%boot(BID,BNAME,FARBE)
%reservierung(SID,BID,DATUM)
```

Beantworten Sie die folgenden Anfragen in Datalog und testen Sie unter (`http://datalog.db.in.tum.de/`, Examples => Segler-Boots-Reservierung):

1. Geben Sie die Farben aller Boote, die von 'Lubber' reserviert wurden aus.
   ```
   lubber_farbe(F) :- segler(SID,'Lubber',_,_), reservierung(SID,BID,_),
                      boot(BID, _, F).
   ```
2. Geben Sie alle Segler aus, die eine Einstufung von mindestens 8 oder das Boot 103 reserviert haben.
   ```
   a2(SID,N) :- segler(SID,N,R,_), R>=8.
   a2(SID,N) :- segler(SID,N,_,_), reservierung(SID,103,_).
   ```
3. Geben Sie die Namen aller Segler aus, die mindestens zwei Boote reserviert haben.
   ```
   doppelBoot(S) :- segler(SID,S,_,_), reservierung(SID,BIDA,_),
                    reservierung(SID,BIDB,_), BIDA\=BIDB .
   ```
4. Geben Sie alle Segler aus, die noch nie ein rotes Boot reserviert haben.
   ```
   rotReserviert(SID) :- segler(SID,_,_,_), reservierung(SID,BID,_),
                         boot(BID,_,red).
   nichtRot(SID,S) :- segler(SID,S,_,_), not(rotReserviert(SID)).
   ```
5. Geben Sie alle Segler aus, die mehr als 20 Jahre alt sind und kein rotes Boot reserviert haben.
   ```
   rotReserviert(SID) :- segler(SID,_,_,_), reservierung(SID,BID,_),
                         boot(BID,_,red).
   nichtRotAlt(SID,S,A) :- segler(SID,S,_,A), A>20, not(rotReserviert(SID)).
   ```
6. Geben Sie die Ids der Segler aus, deren Einstufung besser als die eines Seglers mit Namen 'Horatio' ist.
   ```
   nichtSchlecht(SID) :- segler(SID,_,R,_), segler(_,'Horatio',RH,_), R > RH.
   ```
7. Geben Sie die Ids der Segler aus, deren Einstufung besser als die aller Segler mit Namen 'Horatio' ist.
   ```
   dochSchlecht(SID) :- segler(SID,_,R,_), segler(_,'Horatio',RH,_), R=<RH.
   nochBesser(SID) :- segler(SID,_,_,_), not(dochSchlecht(SID)).
   ```
8. Geben Sie den Namen und Alter des ältesten Seglers aus.
   ```
   junger(SID) :- segler(SID,_,_,A), segler(_,_,_,AO), A<AO.
   alter(S,A) :- segler(SID,S,_,A), not(junger(SID)).
   ```

Hausaufgabe 3

Gegeben sei die nachfolgende *KindEltern*-Ausprägung für den Stammbaum-Ausschnitt der griechischen Götter und Helden:

KindEltern		
Vater	Mutter	Kind
Zeus	Leto	Apollon
Zeus	Leto	Artemis
Kronos	Rheia	Hades
Zeus	Maia	Hermes
Koios	Phoebe	Leto
Atlas	Pleione	Maia
Kronos	Rheia	Poseidon
Kronos	Rheia	Zeus

```
ke(zeus,leto,apollon).
ke(zeus,leto,artemis).
ke(kronos,rheia,hades).
ke(zeus,maia,hermes).
ke(koios,phoebe,leto).
ke(atlas,pleione,maia).
ke(kronos,rheia,poseidon).
ke(kronos,rheia,zeus).
```

Formulieren Sie folgende Anfragen in Datalog und testen Sie unter (`http://datalog.db.in.tum.de/`):

a) Bestimmen Sie alle Geschwisterpaare.

```
parent(P,K) :- kindEltern(P,_,K).
parent(P,K) :- kindEltern(_,P,K).

sibling(A,B) :- parent(P,A),parent(P,B),A\=B.
```

b) Ermitteln Sie Paare von Cousins und Cousinen beliebigen Grades. Die Definition finden Sie auf Wikipedia.

```
cousin(A,B) :- parent(PA,A), parent(PB,B), sibling(PA,PB).
cousin(A,B) :- parent(PA,A), parent(PB,B), cousin(PA,PB).
```

c) Geben Sie alle Verwandtschaftspaare an. Überlegen Sie sich eine geeignete Definition von Verwandtschaft und setzen Sie diese in Datalog um.

```
related(A,B) :- sibling(A,B).
related(A,B) :- related(A,C),parent(C,B).
related(A,B) :- related(C,B),parent(C,A).
```

d) Bestimmen Sie alle Nachfahren von Kronos. Formulieren Sie die Anfrage auch in SQL, so dass sie unter PostgreSQL ausführbar ist (online testen unter: `http://sqlfiddle.com` mit der Datenbank PostgreSQL statt MySQL, das Schema Textfeld können sie leer lassen, müssen aber trotzdem auf 'Build Schema' drücken). Sie können die Daten als Common Table Expression definieren und dann nutzen:

```
WITH RECURSIVE
kindEltern(vater,mutter,kind) as (
  VALUES
```

```
   ('Zeus', 'Leto', 'Apollon'),
   ('Zeus', 'Leto', 'Artemis'),
   ('Kronos', 'Rheia', 'Hades'),
   ('Zeus', 'Maia', 'Hermes'),
   ('Koios', 'Phoebe', 'Leto'),
   ('Atlas', 'Pleione', 'Maia'),
   ('Kronos', 'Rheia', 'Poseidon'),
   ('Kronos', 'Rheia', 'Zeus')
),
parent(eltern,kind) as (
  select vater, kind from kindEltern UNION
  select mutter, kind from kindEltern
)
select * from parent where eltern='Zeus'
```

Datalog

```
nachfahr(P,N) :- parent(P,N).
nachfahr(P,N) :- nachfahr(P,X),nachfahr(X,N).
```

Alternativ

```
nachfahr(P,N) :- parent(P,N).
nachfahr(P,N) :- nachfahr(P,X),parent(X,N).
```

Anfrage für die Nachfahren von Kronos

```
nachfahr(kronos,X).
```

SQL

```
WITH RECURSIVE
kindEltern(vater,mutter,kind) as (
  VALUES
   ('Zeus', 'Leto', 'Apollon'),
   ('Zeus', 'Leto', 'Artemis'),
   ('Kronos', 'Rheia', 'Hades'),
   ('Zeus', 'Maia', 'Hermes'),
   ('Koios', 'Phoebe', 'Leto'),
   ('Atlas', 'Pleione', 'Maia'),
   ('Kronos', 'Rheia', 'Poseidon'),
   ('Kronos', 'Rheia', 'Zeus')
),
parent(eltern, kind) as(
  select vater, kind from kindEltern UNION
  select mutter, kind from kindEltern
),
nachfahren(person, nachfahre) AS (
  SELECT * from parent
UNION ALL
  SELECT n.person, p.kind FROM nachfahren n, parent p
  WHERE p.eltern = n.nachfahre
)
select * from nachfahren WHERE person='Kronos'
```

Gruppenaufgabe 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?

Siehe Übungsbuch