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

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe24

Alice Rey, Maximilian Bandle, Michael Jungmair (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss24/impldb/

Blatt Nr. 04

Hinweise Die Datalogaufgaben können auf `https://souffle.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), $X = Y, X! = Y$ (gleich, ungleich), $!pred(X, Y)$ (existiert nicht pred(X,Y)).

Hausaufgabe 1

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

```
.decl segler(sid: number, sname: symbol, einstufung: number, alter: number)
.decl boot(bid: number, bname: symbol, farbe: symbol)
.decl reservierung(sid: number, bid: number, datum: number)
```

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

1. Geben Sie die Farben aller Boote, die von 'Lubber' reserviert wurden, aus.

   ```
   .decl lubber_farbe(farbe: symbol)
   ```

2. Geben Sie alle Segler aus, die eine Einstufung von mindestens 8 oder das Boot 103 reserviert haben.

   ```
   .decl a2(sid: number, name: symbol)
   ```

3. Geben Sie die Namen aller Segler aus, die mindestens zwei Boote reserviert haben.

   ```
   .decl doppelBoot(name: symbol)
   ```

4. Geben Sie alle Segler aus, die noch nie ein rotes Boot reserviert haben.
5. Geben Sie alle Segler aus, die mehr als 20 Jahre alt sind und kein rotes Boot reserviert haben.
6. Geben Sie die Ids der Segler aus, deren Einstufung besser als die eines Seglers mit Namen 'Horatio' ist.
7. Geben Sie die Ids der Segler aus, deren Einstufung besser als die aller Segler mit Namen 'Horatio' ist.
8. Geben Sie den Namen und Alter des ältesten Seglers aus.

Hausaufgabe 2

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
Poseidon	Amphitrite	Triton

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

a) Bestimmen Sie alle Geschwisterpaare.

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

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

d) Bestimmen Sie alle Nachfahren von Kronos. Formulieren Sie die Anfrage auch in SQL, so dass sie unter HyPer ausführbar ist (online testen unter: `http://hyper-db.de/interface.html`). 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'),
   ('Poseidon', 'Amphitrite', 'Triton')
), parent(eltern,kind) as (
  select vater, kind from kindEltern UNION select mutter, kind from kindEltern
) select * from parent where eltern='Zeus'
```

Hausaufgabe 3

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.

```
.decl pruefendeProfs(name: symbol)
```

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 die nachfolgende Datalog-Anfrage so, dass die Titel der Vorlesungen ausgegeben werden. Was bedeutet diese Anfrage?

```
.decl geschwisterVL(N1: number, N2: number)
.decl nahverwandtVL(N1: number, N2: number)
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).
```

Hausaufgabe 4

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 `Russel`s Prüfling, `Carnap`. Führen Sie die Anfrage im Datalog Tool aus! .decl fremdgeprueft(SN: symbol, PID: number, VPID: number)

Hausaufgabe 5

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.

```
.decl parent(child: symbol, parent: symbol)
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.

```
.decl sg(h1: symbol, h2: symbol)
```

b) Demonstrieren Sie die naive Ausführung des Prädikats.

c) Erläutern Sie das Vorgehen bei der seminaiven Auswertung.

Hausaufgabe (wird nicht in der Übung besprochen)

Nun fügen wir der EDB folgende Einträge hinzu[1]:

```
.decl modelParts(model: symbol, bauteil: symbol)
.decl part(teil: symbol, hersteller: symbol)
.decl consistsOf(komponente: symbol, bauteil: symbol)
```

`part` ist hierbei ein Bauteil eines Geräts, `marker` ist der Hersteller des Bauteils. `ModelParts` verbindet ein Modell aus den ursprünglichen Daten mit seinem/seinen Bauteil(en). `ConsistsOf` beschreibt die hierarchische Beziehung zwischen Bauteilen.

'kompaktes' Beispiel:

```
modelParts("workstation","mainboardh-l7").
modelParts("workstation","hdd30g").
part("mainboard-hl7","asuz").
part("gpu7700","nvidio").
part("hdd30g","sealgate").
part("transistor","foxcom").
part("motor","enginesUnited").
part("wire","theWireCompany").
part("magnet","theMagnetCompany").
consistsOf("hdd30g","transistor").
consistsOf("hdd30g","motor").
consistsOf("motor","wire").
consistsOf("motor","magnet").
...
```

Beantworten Sie in Datalog:

a) Find all models containing parts made by `sealgate`.

b) Find all models which contain two different parts by the same maker (regardless of where in the hierarchy).

Hausaufgabe (wird nicht in der Übung besprochen)

Gegeben das folgende Schema der EDB[2]:

```
.decl product(maker: symbol, model: symbol, type: symbol)
.decl pc(model: symbol, speed: float, ram: number, hd: number, price: number)
.decl laptop(model: symbol, speed: float, ram: number,
             hd: number, screen: symbol, price: number)
.decl printer(model: symbol, color: symbol, type: symbol, price: number)
```

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

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

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

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

```
.decl fast_pc(model: symbol, speed: float, price: number)
```

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

```
.decl manufacturers_100GB(maker: symbol)
```

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

```
.decl b_prod(model: symbol, price: number)
```

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

```
.decl color_laser_printers(model: symbol)
```

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

```
.decl laptop_manuf(maker: symbol)
```

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

```
.decl pop_sizes(hd: number)
```

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).

```
.decl sim_pc(m1: symbol, m2: symbol)
```