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

Übung zur Vorlesung
***Einsatz und Realisierung von Datenbanksystemen* im SoSe14**
Moritz Kaufmann (moritz.kaufmann@tum.de)
http://www-db.in.tum.de/teaching/ss14/impldb/

Blatt Nr. Ausnahme

Aufgabe 1

Sie fangen die folgende, mit RSA verschlüsselte Nachricht ab: 13. Sie kennen den öffentlichen Schlüssel (3,15). Wie lautet die Nachricht im Klartext? Geben Sie die komplette Herleitung an.

Alles laut Wikipedia `https://de.wikipedia.org/wiki/RSA-Kryptosystem`:

- Öffentlicher Schlüssel (e,N)
- Privater Schlüssel: (d,N)

$$N = p * q$$

mit p und q sehr großen Primzahlen. e, der sog. Verschlüsselungsexponent wird als Teilerfremde zahl zu $\phi(N)$ gewählt, wobei gilt $1 < e < \phi(N)$. $\phi(N)$ ist hierbei definiert als $\phi(N) = (p-1) * (q-1)$. d, der sog. Entschlüsselungsexponent, ist gerade das multiplikative inverse von e bezüglich des Moduls $\phi(N)$. Die Berechnung erfolgt mittels erweiterten euklidschen Algorithmus.
Die Entschlüsselung einer verschlüsselten Nachricht C zu ihrem Klartext K erfolgt mittels der Formal

$$K = C^d \mod N.$$

Aus der Angabe wissen wir:

- $N = 15$
- $e = 3$
- $C = 13$

Wir müssen also zunächst d berechnen. Dies wäre einfach, wenn wir $\phi(N)$ wüssten. Hierzu ist die Primfaktorzerlegung von N nötig. Dies ist für die Zahl 15 äußerst einfach, es gilt $N = 5 * 3$. Damit ist $\phi(N) = 4 * 2 = 8$. Die Lösung der Kongruenz

$$e * d \equiv 1 \ mod \phi(N)$$

bzw. im konkreten Fall

$$3 * d \equiv 1 \ mod 8$$

können wir raten, indem wir alle im Bezug auf 8 teilerfremden Zahlen z betrachten, für die gilt: $1 < z < 8$.
Für $z = 3$ gilt $3 * 3 \equiv 1 \ mod 8$, womit $d = 3$ ist.
Wir entschlüsseln nun die Nachricht:

$$K = 13^3 \mod 15 = 7$$

.
Der Klartext K ist also 7.

Aufgabe 2

Gegeben das folgende Schema der EDB[1]:

```
Product(maker, model, type).
```

[1] Inspired by and mostly taken from `http://people.inf.elte.hu/sila/DB1English/exercise06_products.pdf`.

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

Benatworten Sie in Datalog:

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

```
fastcpu(MODEL) :- pc(MODEL,S,R,H,P), S>=3.0.
```

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

```
makerlh(MAKER) :- laptop(MODEL,_,_,HDD,_,_),
        product(MAKER,MODEL,laptop),
        HDD>=100.
```

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

```
allmodels(MAKER,MODEL,TYPE,PRICE) :-
        product(MAKER,M,TYPE),
        pc(M,_,_,_,_,PRICE),
        MODEL=M.
allmodels(MAKER,MODEL,TYPE,PRICE) :-
        product(MAKER,M,TYPE),
        laptop(M,_,_,_,_,_,PRICE),
        MODEL=M.
allmodels(MAKER,MODEL,TYPE,PRICE) :-
        product(MAKER,M,TYPE),
        printer(M,_,_,_,PRICE),
        MODEL=M.

bproducts(M,T,P) :- allmodels(b,M,T,P).
```

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

```
cprinters(MODEL) :- printer(MODEL,color,laser,_).
```

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

```
pcmakers(MAKER) :- allmodels(MAKER,_,pc,_).
laptopmakers(MAKER) :- allmodels(MAKER,_,laptop,_).

makerslnp(MAKER) :- laptopmakers(MAKER),not(pcmakers(MAKER)).
```

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

```
hdds(HDDSIZE) :- pc(MODELA,_,_,HDDSIZE,_), pc(MODELB,_,_,HDDSIZE,_),
                 MODELA\=MODELB.
```

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).

```
pcpair(MODELA,MODELB) :-
        pc(MODELA,SPEEDA,RAMA,_,_),
        pc(MODELB,SPEEDB,RAMB,_,_),
        SPEEDA=SPEEDB,
        RAMA=RAMB,
        MODELA<MODELB.
```

h) Find those manufacturers of at least two different computers (PC's or laptops) with speeds of at least 2.80 GHz.

```
computer(MAKER,MODEL,SPEED,RAM,HDD,PRICE) :-
        pc(MODEL,SPEED,RAM,HDD,PRICE), product(MAKER, MODEL, _).
computer(MAKER,MODEL,SPEED,RAM,HDD,PRICE) :-
        laptop(MODEL,SPEED,RAM,HDD,_,PRICE), product(MAKER, MODEL, _).

manyfast(MAKER) :-
        group_by((computer(MAKER,MODEL,SPEED,RAM,HDD,PRICE), SPEED>=2.8),
                 [MAKER], COUNT=count),
        COUNT >= 2.
```

i) Find the manufacturers of the computer (PC or laptop) with the highest available speed.

```
fastestmaker(MAKER) :-
        max(computer(_,_,COMPSPEED,_,_,_),COMPSPEED, MAXSPEED),
        computer(MAKER,_,SPEED,_,_,_),
        SPEED=MAXSPEED.
```

j) Find the manufacturers of PC's with at least three different cpu speeds.

```
% pc(maker,model, speed, ram, hdd, price)
pc(MAKER,MODEL,SPEED,RAM,HDD,PRICE) :-
        pc(MODEL,SPEED,RAM,HDD,PRICE),
        product(MAKER,MODEL,_).

pcMakerAndSpeed(MAKER,SPEED) :- pc(MAKER,_,SPEED,_,_,_).

threecpumaker(MAKER) :-
        group_by(pcMakerAndSpeed(MAKER,SPEED), [MAKER], COUNT=count),
        COUNT>=3.
```

k) Find the manufacturers who sell exactly three different models of PC.

```
exactthreemaker(MAKER) :-
        group_by(product(MAKER,_,pc),[MAKER], COUNT=count),
        COUNT=3.
```

Nun fügen wir der EDB folgende Enträge hinzu:

```
ModelParts(model,partname)
Part(partname,maker)
ConsistsOf(partname,partname)
```

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

'kompaktes' Beispiel:

```
ModelParts(workstation,mainboard-hl7).
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:

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

```
consistsOfRec(PARTNAME1,PARTNAME2) :-
        consistsOf(PARTNAME1,PARTNAME2).
consistsOfRec(PARTNAME1,PARTNAME2) :-
        consistsOf(PARTNAME1,P),consistsOfRec(P,PARTNAME2).

modelPartsRec(MODEL,PARTNAME,PARTMAKER) :-
        modelParts(MODEL,PARTNAME),part(PARTNAME,PARTMAKER).
modelPartsRec(MODEL,PARTNAME,PARTMAKER) :-
        modelParts(MODEL,P),consistsOfRec(P,PARTNAME),
        part(PARTNAME,PARTMAKER).

sealgatemodels(MODEL) :- modelPartsRec(MODEL,_,sealgate).
```

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

```
twoparts(MODEL) :-
        modelPartsRec(MODEL,PARTNAME1,PARTMAKER),
        modelPartsRec(MODEL,PARTNAME2,PARTMAKER),
        PARTNAME1<PARTNAME2.
```

Aufgabe 3

Wann genau können die Sperren gemäß dem strengen 2PL-Protokoll freigegeben werden? Denken Sie an die Recovery-Komponente.

Die Schreibsperren müssen so lange gehalten werden, bis ein lokales Zurücksetzen der Transaktion nicht mehr nötig ist. Dies bedeutet, dass die Sperren als letzte Aktion der **commit**-Behandlung freigegeben werden. In der **commit**-Behandlung werden eventuell noch Konsistenzchecks durchgeführt. Falls etwa eine Verletzung einer referentiellen Integrität festgestellt wird, so muss die entsprechende Transaktion lokal zurückgesetzt werden.

Aufgabe 4

Berechnen Sie, wie groß ein Data Warehouse für ein Handelsunternehmen wie Quelle oder Amazon.com wäre, wenn die Bestelldaten der letzten 3 Jahre enthalten sind. Verwenden Sie den Jahresumsatz von Amazon als Ausgangspunkt Ihrer Abschätzung.