16/1/2019 Úvod do databáz, skúškový test, max **60** bodov

1. Uvažujte databázu bez duplikátov a null hodnôt: capuje(Krcma, Alkohol),
lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Idn → Pijan, Krcma; Idn, Alkohol → Mnozstvo; Mnozstvo > 0.
a) Sformulujte bezpečný dotaz v relačnom kalkule **(6)** a relačnej algebre **(6)** na
pijanov, ktorí ľúbia práve jeden z alkoholov, ktoré ľúbi pijan Juro; a v krčmách, ktoré Juro navštívil, nevypili žiaden Jurov obľúbený alkohol.

Relačný kalkul:

$$
\begin{aligned}
&\{P: \\
&\quad \exists A \ /^* \text{ A je alkohol, ktorý ľúbi P aj juro } ^*/ \\
&\quad lubi(P, A) \wedge lubi(juro, A) \wedge \\
&\quad \neg \ /^* \text{ neplatí, že aj nejaký iný alkohol ľúbi P aj juro } ^*/ \\
&\quad (\\
&\qquad \exists A2 \\
&\qquad lubi(P, A2) \wedge lubi(juro, A2) \wedge A2 \neq A \\
&\quad) \wedge \\
&\quad \neg \ /^* \text{ neplatí, že P a juro navštívili nejakú krčmu K, v ktorej P vypil nejaký jurov obľúbený alkohol A2 } ^*/ \\
&\quad (\\
&\qquad \exists J \ \exists K \ \exists A2 \ \exists I \ \exists M \\
&\qquad navstivil(J, juro, K) \wedge lubi(juro, A2) \wedge navstivil(I, P, K) \wedge vypil(I, A2, M) \\
&\quad) \\
&\}
\end{aligned}
$$

Relačná algebra:

$$
\begin{aligned}
&/^* \ jurove\_oblubene(Alkohol) \ ^*/ \\
&jurove\_oblubene = \pi_{Alkohol} (\sigma_{Pijan = \text{‘juro’}} (lubi)); \\
&\\
&/^* \ lubi\_co\_juro(Pijan, Alkohol) \ ^*/ \\
&lubi\_co\_juro = lubi \bowtie jurove\_oblubene; \\
&\\
&/^* \ jurom\_navstivene(Krcma) \ ^*/ \\
&jurom\_navstivene = \pi_{Krcma} (\sigma_{Pijan = \text{‘juro’}} (navstivil)); \\
&\\
&/^* \ answer \ ^*/ \\
&\pi_{lj1.Pijan} \\
&(\\
&\quad P_{lj1} (lubi\_co\_juro) \triangleright_{lj1.Pijan = lj2.Pijan \wedge ll1.Alkohol \neq lj2.Alkohol} P_{lj2} (lubi\_co\_juro) \\
&) \\
&- \\
&\pi_{Pijan} \\
&(\\
&\quad \pi_{Pijan, Krcma, Alkohol} (navstivil \bowtie vypil) \bowtie jurove\_oblubene \bowtie jurom\_navstivene \\
&);
\end{aligned}
$$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na dvojice krčiem
[K1, K2] také, že každý alkohol vypitý v K1 bol v K1 vypitý vo väčšom celkovom množstve než v K2.

Datalog:

```
answer(K1, K2) ←
        mnozstva(K1, _),
        capuje(K2, _), /* predpokladáme, že každá krčma niečo čapuje */
        not nerekordny_alkohol(K1, K2).

nerekordny_alkohol(K1, K2) ←
        mnozstva(K1, A, T1),
        mnozstva(K2, A, T2),
        T2 >= T1,
        not K1 = K2.

mnozstva(K, A, T) ←
        subtotal(nv(_, K, A, M), [K, A], [T = sum(M)]).

nv(I, K, A, M) ←
        navstivil(I, _, K),
        vypil(I, A, M).
```

SQL:

```
with
mnozstva as
(
select n.Krcma, v.Alkohol, sum(v.Mnozstvo) as T
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Krcma, v.Alkohol
),

nerekordny_alkohol as
(
select m1.Krcma as K1, m2.Krcma as K2
from mnozstva m1, mnozstva m2
where m1.Alkohol = m2.Alkohol and m1.T <= m2.T and m1.Krcma <> m2.Krcma
)

select m.Krcma as K1, c.Krcma as K2
from mnozstva m, capuje c
where not exists (
        select *
        from nerekordny_alkohol na
        where m.Krcma = na.K1 and c.Krcma = na.K2)
```

2. Uvažujte relačnú schému r(A, B, C, D, E, F, G, H) s funkčnými závislosťami
BE → GH, BEG → F, AD → C, F → B, BF → A.
a) Rozhodnite či dekompozícia r1(A, B, F), r2(B, E, F, G, H), r3(C, D), r4(D, E, F) je v tretej normálnej forme. Zdôvodnite. **(6)**

Definícia. Relačná schéma s je v 3NF, ak pre každú v s platnú netriviálnu funkčnú závislosť **X** → Y platí, že buď **X** je nadkľúč v s alebo Y je časťou nejakého kľúča.

Definícia. Daná dekompozícia je v 3NF, ak všetky r1, r2, r3, r4 sú v 3NF.

r1(A, B, F):
Netriviálne funkčné závislosti s 1 atribútom na ľavej aj pravej strane sú F → B, F → A, iné nie sú. Keďže F je nadkľúč v r1, r1 je v 3NF.

r2(B, E, F, G, H):
Netriviálne funkčné závislosti s 1 atribútom na pravej strane a max. 3 atribútmi na ľavej sú BE → G, BE → H, BE → F, EF → G, EF → H, iné nie sú. Keďže BE aj EF sú nadkľúče v r2, r2 je v 3NF.

r3(C, D) je v 3NF, lebo obsahuje len 2 atribúty a neplatí žiadna funkčná závislosť s prázdnou množinou na ľavej strane.

r4(D, E, F) je v 3NF, lebo v nej neplatí žiadna funkčná závislosť s 1 atribútom na ľavej aj pravej strane.

Daná dekompozícia je v 3NF.

b) Uveďte príklad konkrétneho naplnenia relácií r, r1, r2, r3, r4 ktoré demonštruje, že dekompozícia z úlohy a) nie je bezstratová (nespája sa bezstratovo). **(6)**

Overme algoritmom chase, že tá dekompozícia sa skutočne nespája bezstratovo, t.j. pre nejakú populáciu r
$r \neq \pi_{r1}(r) \bowtie \pi_{r2}(r) \bowtie \pi_{r3}(r) \bowtie \pi_{r4}(r)$.

Inicializácia chase (prázdne symboly označujú nejaké navzájom rôzne hodnoty, rôzne od a*):

	A	B	C	D	E	F	G	H
r1(A, B, F)	a1	a2				a6		
r2(B, E, F, G, H)		a2			a5	a6	a7	a8
r3(C, D)			a3	a4				
r4(D, E, F)				a4	a5	a6		

Výsledok chase:

	A	B	C	D	E	F	G	H
r1(A, B, F)	a1	a2				a6		
r2(B, E, F, G, H)	a1	a2			a5	a6	a7	a8
r3(C, D)			a3	a4				
r4(D, E, F)	a1	a2		a4	a5	a6	a7	a8

Tu algoritmus chase končí, daná dekompozícia sa nespája bezstratovo. Demonštruje to naplnenie relácií z výsledku chase:
r(A, B, C, D, E, F, G, H) = {[a1, a2, b13, b14, b15, a6, b17, b18], [a1, a2, b23, b24, a5, a6, a7, a8], [b31, b32, a3, a4, b35, b36, b37, b38], [a1, a2, b43, a4, a5, a6, a7, a8]}.
r1(A, B, F) = {[a1, a2, a6], [b31, b32, b36]}.
r2(B, E, F, G, H) = {[a2, a5, a6, a7, a8], [a2, b15, a6, b17, b18], [b32, b35, b36, b37, b38]}.
r3(C, D) = {[a3, a4], [b13, b14], [b23, b24], [b43, a4]}.
r4(D, E, F) = {[a4, a5, a6], [a4, b35, b36], [b14, b15, a6], [b24, a5, a6]}.

[a1, a2, a3, a4, a5, a6, a7, a8] $\notin$ r, ale [a1, a2, a3, a4, a5, a6, a7, a8] $\in \pi_{r1}(r) \bowtie \pi_{r2}(r) \bowtie \pi_{r3}(r) \bowtie \pi_{r4}(r)$.

c) SQL príkaz *create assertion* definuje integritné obmedzenie na databázu. Systém dovolí aktualizáciu databázy len keď by sa podmienka v klauze *check* vyhodnotila po aktualizácii ako true (vyhodnocuje sa rovnako ako podmienka v klauze *where*
príkazu *select*); inak abortuje transakciu, ktorá tú aktualizáciu robí. Napríklad,
integritné obmedzenie Mnozstvo > 0 pre databázu z úlohy 1 sa vyjadrí takto:
create assertion mnozstvo_positive check (
not exists (select * from vypil v where v.Mnozstvo <= 0)).
Vyjadrite funkčnú závislosť BE → GH ako integritnú podmienku *cond* v relácii r2 z úlohy a) príkazom create assertion fd_be_gh check *cond*. **(6)**

```
create assertion fd_be_gh check not exists (
        select *
        from r2 t1, r2 t2
        where t1.B = t2.B and t1.E = t2.E and not (t1.G = t2.G and t1.H = t2.H))
```

3. Daný je datalogový program s extenzionálnou databázou r(., .):
p(X, Y) ← r(X, _), r(_, Y), not q(X, Y).
p(X, Y) ← r(X, Y), not q(X, Y).
q(X, Y) ← r(X, _), r(_, Y), not r(X, Y).
q(X, Y) ← r(X, Y), q(Y, X).
a) Zapíšte výpočet dotazu ?- p(X, Y) pre daný program v relačnej algebre. **(6)**

Univerzálnou metódou výpočtu datalogového programu je (semi-) naivná evaluácia. Ale čiastočné pochopenie programu môže výpočet uľahčiť. Napríklad, druhé pravidlo pre s nevypočíta oproti prvému pravidlu žiadnu ďalšiu dvojicu [X, Y]. To prvé pravidlo je všeobecnejšie, o čom sa dá presvedčiť tak, že za prvú anonymnú premennú dosadíme Y a za druhú dosadíme X. Teda druhé pravidlo môžeme ignorovať a uvažovať program
r1: p(X, Y) ← r(X, _), r(_, Y), not q(X, Y).
r2: q(X, Y) ← r(X, _), r(_, Y), not r(X, Y).
r3: q(X, Y) ← r(X, Y), q(Y, X).

Idea plánu výpočtu: raz aplikovať pravidlo r2 (telo pravidla r2 neobsahuje žiadne intenzionálne predikáty); potom iterovať pravidlo r3 kým sa relácia q prestane meniť (keďže r3 neobsahuje negáciu, v iterácii môžu do q len pribúdať nové dvojice); potom raz aplikovať pravidlo r1.

V relačnej algebre (postupnosť priradení):
q2 = (π_X (r) × π_Y (r)) – r;
q = q2;
∅(q = r ⋈ $P_{q(Y, X)}$ (q);) /* iteruj, kým sa q mení */
q = q ∪ q2;
s = (π_X (r) × π_Y (r)) – q /* relácia s obsahuje výsledok dotazu */

Pre daný program stačí jedna iterácia ∅ pre výpočet výslednej relácie (to platí pre ľubovoľné naplnenie extenzionálnej relácie r), čo umožňuje ďalšiu optimalizáciu výpočtu. Taktiež výsledok spoločného podvýrazu π_X (r) × π_Y (r) stačí vypočítať len raz. Optimalizovaný plán výpočtu:
rr = π_X (r) × π_Y (r);
q2 = rr – r;
q = q2 ∪ (r ⋈ $P_{q2(Y, X)}$ (q2));
s = rr – q; /* relácia s obsahuje výsledok dotazu */

b) Odsimulujte výpočet dotazu ?- p(X, Y) z úlohy a) pre extenzionálnu databázu
r = {[1, 1], [1, 2], [2, 2], [2, 3], [3, 1], [3, 3], [4, 1], [5, 6]}. **(6)**

V tele pravidla r2 nie sú žiadne intenzionálne predikáty, preto ho stačí vypočítať len raz:
q2 = (π_X (r) × π_Y (r)) – r;
q2(X, Y) = {[1, 3], [1, 6], [2, 1], [2, 6], [3, 2], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

q = q2;
q(X, Y) = {[1, 3], [1, 6], [2, 1], [2, 6], [3, 2], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

∅(q = r ⋈ $P_{q(Y, X)}$ (q);) /* iteruj, kým sa q mení */
q(X, Y) = {[1, 2], [2, 3], [3, 1]}
(v druhej iterácii sa q už nezmení)

q = q ∪ q2;
q(X, Y) = {[1, 3], [1, 2], [1, 6], [2, 1], [2, 3], [2, 6], [3, 1], [3, 2], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

s := (π_X (r) × π_Y (r)) – q /* relácia s obsahuje výsledok dotazu */
s(X, Y) = {[1, 1], [2, 2], [3, 3], [4, 1], [5, 6]}

4. Pri opätovnom štarte databázového systému obsahuje log-file nasledujúce záznamy (zoradené od najstaršieho po najnovší): [T0, start], [T0, A, 2, 1],
[T1, start], [T0, commit], [T2, start], [T2, A, 1, 5], [T2, B, 2, 3], [T1, A, 5, 7],
[T1, commit], [T2, A, 7, 5]. Odsimulujte algoritmus obnovy (bez predpokladov na cache) pre tento log-file, uveďte postupnosť krokov algoritmu pri čítaní log-file. **(6)**

Log záznam	Akcia
	undo_list = {}; redo_list = {};
	/* zostupný prechod cez log */
[T2, A, 7, 5]	undo_list = {T2}; write(A, 7);
[T1, commit]	redo_list = {T1};
[T1, A, 5, 7]	
[T2, B, 2, 3]	write(B, 2);
[T2, A, 1, 5]	write(A, 1);
[T2, start]	undo_list = {};
[T0, commit]	redo_list = {T1, T0};
[T1, start]	
[T0, A, 2, 1]	
[T0, start]	
[BOF]	
	/* vzostupný prechod cez log */
[T0, start]	
[T0, A, 2, 1]	write(A, 1);
[T1, start]	
[T0, commit]	redo_list = {T1};
[T2, start]	
[T2, A, 1, 5]	
[T2, B, 2, 3]	
[T1, A, 5, 7]	write(A, 7);
[T1, commit]	redo_list = {}; terminate;