1/2/2023 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 Datalogu **(6)** a relačnej algebre **(6)** na dvojice [K, A] také, že krčma K čapuje alkohol A, a zároveň A bol pri každej návšteve krčmy K vypitý v množstve aspoň 5.

Datalog:

```
answer(K, A) ←
        capuje(K, A),
        not niekedy_malo(K, A).

niekedy_malo(K, A) ←
        capuje(K, A),
        navstivil(I, _, K),
        not aspon5(I, A).

aspon5(I, A) ←
        navstivil(I, _, _),
        vypil(I, A, M),
        M >= 5.
```

Relačná algebra:

$aspon5 = \pi_{Idn,\ Alkohol}(\sigma_{Mnozstvo >= 5}(navstivil \bowtie vypil))$

$niekedy\_malo = \pi_{Krcma,\ Alkohol}((capuje \bowtie navstivil) \triangleleft aspon5)$

/* answer */

$capuje - niekedy\_malo$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na trojice [K, A, C], kde C je počet pijanov, ktorí alkohol A ľúbia a vypili ho v krčme K v celkovom množstve aspoň 100. (Vo výsledku majú byť len trojice s C>0.)

Datalog:

```
answer(K, A, C) ←
        subtotal(lubi100(P, K, A), [K, A], [C = count(P)]).

lubi100(P, K, A) ←
        lubi(P, A),
        subtotal(v(_, P, K, A, M), [P, K, A], [S = sum(M)]),
        S >= 100.

v(I, P, K, A, M) ←
        navstivil(I, P, K),
        vypil(I, A, M).
```

SQL:

```
with lubi100 as
(
select l.Pijan, n.Krcma, v.Alkohol
from lubi l, navstivil n, vypil v
where l.Pijan = n.Pijan and l.Alkohol = v.Alkohol and n.Idn = v.Idn
group by n.Pijan, n.Krcma, v.Alkohol
having sum(v.Mnozstvo) >= 100
)
select l100.Krcma as K, l100.Alkohol as A, count(distinct l100.Pijan) as P
from lubi100 l100
group by l100.Krcma, l100.Alkohol
```

2. Uvažujte relačnú schému s atribútmi $A_1, ..., A_N$.
a) Rozhodnite či platí
„Ak $\exists i \exists j\ 1 \leq i \leq N \wedge 1 \leq j \leq N \wedge i \neq j \wedge \{A_1, ..., A_N\} - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$, tak daná relačná schéma nie je v Boyce-Coddovej normálnej forme."
Ak ÁNO, dokážte. Ak NIE, uveďte definíciu BCNF a vysvetlite rozpor na konkrétnom príklade. **(6)**

NIE. Toto neplatí.

Uvažujme relačnú schému s atribútmi A_1, A_2, A_3, s množinou funkčných závislostí $A_1 \rightarrow A_2$, $A_1 \rightarrow A_3$. Platí (pre i=2, j=3), že $\{A_1, A_2, A_3\} - \{A_2\} - \{A_3\} \rightarrow \{A_3\}$ (keďže $A_1 \rightarrow A_3$). Zároveň táto schéma je v BCNF (jediné platné netriviálne funkčné závislosti sú tie uvedené dve, A_1 je kľúč).

b) Uvažujte množiny funkčných závislostí **C**, **S** nad atribútmi $A_1, ..., A_N$. Definujte čo znamená, že **C** pokrýva **S**; a rozhodnite či existuje algoritmus s polynomiálnou časovou zložitosťou, ktorý rozhodne, či **C** pokrýva **S**. Ak NIE, zdôvodnite. Ak ÁNO, popíšte ten algoritmus (napíšte pseudokód a zdôvodnite jeho časovú zložitosť). **(6)**

Definícia. Množina funkčných závislostí **C** pokrýva množinu funkčných závislostí **S**, ak $\mathbf{C}^+ \supseteq \mathbf{S}^+$ (kde $\mathbf{M}^+$ označuje uzáver množiny funkčných závislostí **M**).

Množiny $\mathbf{C}^+$ a $\mathbf{S}^+$ v tejo definícii môžu byť exponenciálne veľké vzhľadom na veľkosť C a S. Napriek tomu je možné testovať pokrytie v polynomiálnom čase, t.j. odpoveď je **ÁNO**.

Algoritmus:
Vstup: $A_1, ..., A_N$, **C**, **S**.
for (funkčná závislosť z **S** tvaru $\mathbf{X} \rightarrow \mathbf{Y}$)
    if (not $\mathbf{Y} \subseteq \mathbf{X}^{C+}$)
        výstup „**C** nepokrýva **S**";
výstup „**C** pokrýva **S**";

V tomto algoritme $\mathbf{X}^{C+}$ označuje uzáver množiny atribútov **X** vzhľadom na množinu funkčných závislostí **C**. Uzáver množiny atribútov je možné vypočítať v polynomiálnom čase. Pre každú funkčnú závislosť z **S** stačí raz vypočítať uzáver ľavej strany a porovnať s pravou stranou. Tým pádom má uvedený algoritmus polynomiálnu časovú zložitosť (vzhľadom na veľkosť C a **S**).

3. Uvažujte Datalogový program nad extenzionálnou databázou {e(., .)}:
$p(X, Y) \leftarrow e(X, Y)$.
$p(X, Y) \leftarrow p(X, Z), p(Z, Y)$.
$q(X, Y, Z) \leftarrow p(X, Z), p(Z, Z), p(Z, Y)$.
a) Zapíšte výpočet dotazu ? q(X, Y, 2) v relačnej algebre. **(6)**

Naivná evaluácia:
/* inicializácia */
$p(X, Y) = \{\}$;
$q(X, Y, Z) = \{\}$;
/* fixpoint */
$\phi($
$p = e \cup \pi_{p1.X,\, p2.Y} (\rho_{p1} (p) \bowtie_{p1.Y = p2.X} \rho_{p2} (p))$;
$q = \pi_{p1.X,\, p3.Y,\, p1.Y} ((\rho_{p1} (p) \bowtie_{p1.Y = p2.X \wedge p2.X = p2.Y} \rho_{p2} (p)) \bowtie_{p2.X = p3.X} \rho_{p3} (p))$;
);
/* answer */
$\pi_{X,\, Y} (\sigma_{Z=2} (q))$

b) Vypočítajte výsledok dotazu z úlohy a) pre reláciu
e(., .) = {[0, 1], [1, 3], [2, 4], [3, 5], [4, 2], [5, 4]}. **(6)**

Výpočet uvedený v a) budeme trošku optimalizovať. Keďže p závisí len od p, a tiež q závisí len od p, môžeme najskôr vypočítať celú reláciu p a následne celú reláciu q:
/* inicializácia */
p(X, Y) = {};
q(X, Y, Z) = {};
/* fixpoint */
ϕ(
$p = e \cup \pi_{p1.X,\, p2.Y} (\rho_{p1} (p) \bowtie_{p1.Y = p2.X} \rho_{p2} (p))$;
);
$q = \pi_{p1.X,\, p3.Y,\, p1.Y} ((\rho_{p1} (p) \bowtie_{p1.Y = p2.X \wedge p2.X = p2.Y} \rho_{p2} (p)) \bowtie_{p2.X = p3.X} \rho_{p3} (p))$;
/* answer */
$\pi_{X,\, Y} (\sigma_{Z=2} (q))$

Po inicializácii:
p(X, Y) = {};
q(X, Y, Z) = {};

Po 1. iterácii ϕ:
p = {**[0, 1]**, **[1, 3]**, **[2, 4]**, **[3, 5]**, **[4, 2]**, **[5, 4]**}

Po 2. iterácii ϕ:
p = {[0, 1], **[0, 3]**, [1, 3], **[1, 5]**, **[2, 2]**, [2, 4], **[3, 4]**, [3, 5], [4, 2], **[4, 4]**, **[5, 2]**, [5, 4]}

Po 3. iterácii ϕ:
p = {[0, 1], [0, 3], **[0, 4]**, **[0, 5]**, **[1, 2]**, [1, 3], **[1, 4]**, [1, 5], [2, 2], [2, 4], **[3, 2]**, [3, 4], [3, 5], [4, 2], [4, 4], [5, 2], [5, 4]}

Po 4. iterácii ϕ:
p = {[0, 1], **[0, 2]**, [0, 3], [0, 4], [0, 5], [1, 2], [1, 3], [1, 4], [1, 5], [2, 2], [2, 4], [3, 2], [3, 4], [3, 5], [4, 2], [4, 4], [5, 2], [5, 4]}

Po 5. iterácii ϕ:
p = {[0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [1, 2], [1, 3], [1, 4], [1, 5], [2, 2], [2, 4], [3, 2], [3, 4], [3, 5], [4, 2], [4, 4], [5, 2], [5, 4]}

Po výpočte q(X, Y, Z):
q = {[0, 2, 2], [0, 2, 4], [0, 4, 2], [0, 4, 4], [1, 2, 2], [1, 2, 4], [1, 4, 2], [1, 4, 4],
[2, 2, 2], [2, 2, 4], [2, 4, 2], [2, 4, 4], [3, 2, 2], [3, 2, 4], [3, 4, 2], [3, 4, 4],
[4, 2, 2], [4, 2, 4], [4, 4, 2], [4, 4, 4], [5, 2, 2], [5, 2, 4], [5, 4, 2], [5, 4, 4]}

Po finálnej selekcii a projekcii:
/* answer */
.(X, Y) = {[0, 2], [0, 4], [1, 2], [1, 4], [2, 2], [2, 4], [3, 2], [3, 4], [4, 2], [4, 4], [5, 2], [5, 4]}

4.
a) Vysvetlite validačnú metódu pre izoláciu transakcií, ktorá validáciu aj zápisy transakcie robí súčasne, v jednom atomickom kroku. **(6)**

Scheduler používajúci validačnú metódu, kde validácia prebehne súčasne so všetkými zápismi transakcie, udržiava pre každú začatú transakciu T množinu identifikátorov objektov RS(T) (objekty, ktoré T čítala), množinu identifikátorov objektov WS(T) (objekty, ktoré T aktualizovala), privátnu kópiu objektov, ktoré T aktualizuje, a niekoľko časových pečiatok.
Po prečítaní operácie s_T pridelí transakcii T časovú pečiatku TS(T).
Operácie $r_T(D)$ a $w_T(D)$ vykonáva v takom poradí, ako prichádzajú (hneď ako ich prečíta). Zápisy robí do privátnej kópie objektu D prislúchajúcom transakcii T, nie do databázy. Pri čítaní preferuje privátnu kópiu objektu D, ak existuje; inak číta hodnotu z databázy.
Po prečítaní operácie a_i zahodí všetky údaje o transakcii T.
Po prečítaní operácie c_T pridelí T časovú pečiatku TVAL(T) a otestuje validačnú podmienku. Ak validačná podmienka nie je splnená, tak abortuje T (tak, ako po prečítaní operácie a_T). Inak zapíše privátnu kópiu všetkých objektov modifikovaných transakciou T do databázy, zapíše do logu commit T a zahodí všetky údaje o T.
Validačná podmienka pre transakciu T je (všetky časové pečiatky, ktoré neboli pridelené, majú hodnotu ∞):
$\neg (\exists T2\ TS(T) < TVAL(T2) < TVAL(T) \wedge WS(T2) \cap RS(T) \neq \emptyset)$.
Transakcie sú commitované v poradí, v ktorom sa validujú. Vďaka atomickosti validácie a finalizácie sa write-write konfliktami netreba zapodievať. Uvedená validačná podmienka (možno sa dá zoslabiť) zaručuje, aby v prípade write-read konfliktov validovaná transakcia čítala len hodnoty objektov zapísané (sebou, alebo) poslednou commitovanou transakciou.

b) Uvažujte systém s validačnou metódou z úlohy a). Do systému vstupuje rozvrh (v systéme sú len tieto dve transakcie)
s2, r2(Z), s1, r1(X), w2(Y), r1(Y), r2(X), w1(X), c1, w2(Z), c2.
Po každej prečítanej operácii popíšte akciu, ktorú scheduler vykoná. Uveďte výstupný rozvrh. **(6)**

s2 pridelí časovú pečiatku TS(T2)
r2(Z) prečíta hodnotu Z z databázy pre T2, pridá Z do RS(T2)
s1 pridelí časovú pečiatku TS(T1)
r1(X) prečíta hodnotu X z databázy pre T1, pridá X do RS(T1)
w2(Y) zapíše novú hodnotu Y do privátnej kópie T2, pridá Y do WS(T2)
r1(Y) prečíta hodnotu Y z databázy pre T1, pridá Y do RS(T1)
r2(X) prečíta hodnotu X z databázy pre T2, pridá X do RS(T2)
w1(X) zapíše novú hodnotu X do privátnej kópie T1, pridá X do WS(T1)
c1 otestuje validačnú podmienku pre T1; keďže je splnená, zapíše objekty z WS(T1) z privátnej kópie T1 do databázy, commituje T1 a zruší všetky údaje o T1
w2(Z) zapíše novú hodnotu Z do privátnej kópie T2, pridá Z do WS(T2)
c2 otestuje validačnú podmienku pre T2; keďže nie je splnená, abortuje T2 a zruší všetky údaje o T2

Výstupný rozvrh:
s2, r2(Z), s1, r1(X), r1(Y), r2(X), w1(X), c1, a2