6/2/2017 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 v relačnom kalkule **(6)** a Datalogu **(6)** bezpečný dotaz na trojice
[K, A1, A2] také, že krčma K čapuje alkoholy A1 aj A2; a zároveň každý pijan, ktorý v K niekedy vypil A1, vypil v K niekedy aj A2 (nie nutne oba pri rovnakej návšteve).

Relačný kalkul:

```
{[K, A1, A2]:
        capuje(K, A1) ∧ capuje(K, A2) ∧ ¬
        (       /* niekto_porusil(K, A1, A2) */
                ∃P ∃I ∃M
                navstivil(I, P, K) ∧ vypil(I, A1, M) ∧ ¬
                (       /* niekedy_vypil(P, K, A2) */
                        ∃I ∃M
                        navstivil(I, P, K) ∧ vypil(I, A2, M)
                )
        )
}
```

Datalog:

```
answer(K, A1, A2) ←
        capuje(K, A1),
        capuje(K, A2),
        not niekto_porusil(K, A1, A2).

niekto_porusil(K, A1, A2) ←
        niekedy_vypil(P, K, A1),
        capuje(K, A2),          /* safety */
        not niekedy_vypil(P, K, A2).

niekedy_vypil(P, K, A) ←
        navstivil(I, P, K),
        vypil(I, A, _).
```

b) Sformulujte v Datalogu **(6)** a v SQL **(6)** dotaz na trojice [K, A, P], kde alkohol A sa čapuje v krčme K a P je počet návštev krčmy K, pri ktorých sa alkohol A vypil v množstve menšom ako 2. (Počet P zahŕňa aj tie návštevy K, pri ktorých sa alkohol A nevypil.)

Datalog:

```
answer(K, A, P) ←
        capuje(K, A),
        subtotal(navstevy(K, A, I), [K, A], [P = count(I)]).

navstevy(K, A, I) ←
        navstivil(I, _, K),
        vypil(I, A, M),
        M < 2.

navstevy(K, A, I) ←
        capuje(K, A),  /* safety */
        navstivil(I, _, K),
        not v(I, A).

v(I, A) ←
        vypil(I, A, _).
```

SQL:

```
with navstevy as
(
        (
                select n.Krcma, n.Idn
                from navstivil n, vypil v
                where n.Idn = v.Idn and v.Mnozstvo < 2
        )
        union
        (
                select n.Krcma, n.Idn
                from navstivil n, capuje c
                where c.Krcma = n.Krcma and not exists
                (
                        select *
                        from vypil v
                        where n.Idn = v.Idn and c.Alkohol = v.Alkohol
                )
        )
)
select count(n.Idn)
from navstevy n
group by n.Krcma
```

2. Uvažujte nasledujúce dotazy v relačnom kalkule:
Q1: {X: ∃C ∃Y C>3 ∧ ♥Z, C=count(V) (∃W r(X, Y, Z, V, W))}
Q2: {X: ∃C ∃Y C>3 ∧ ♥C=count(V) (∃Z ∃W r(X, Y, Z, V, W))}
a) Zapíšte Q1 a Q2 v relačnej algebre. **(6)**

Q1:
$r1 = \Pi_{X, Y, Z, V}\ (r);$
$r2 = \Delta(r1);$
$r3 = \Gamma_{X, Y, C = count(V)}\ (r2);$
$r4 = \sigma_{C>3}\ (r3);$
$r5 = \Pi_{X}\ (r4);$
$q1 = \Delta(r4);$

Q2:
$s1 = \Pi_{X, Y, V}\ (r);$
$s2 = \Delta(s1);$
$s3 = \Gamma_{X, Y, C = count(V)}\ (s2);$
$s4 = \sigma_{C>3}\ (s3);$
$s5 = \Pi_{X}\ (s4);$
$q2 = \Delta(s5);$

b) Uveďte príklad konkrétneho naplnenia relácie (resp. predikátu) r(., ., ., ., .), pre ktoré Q1 a Q2 dávajú rôzne výsledky. Vysvetlite kde sa výpočty Q1 a Q2 líšia a uveďte ich výsledky pre zvolené naplnenie r. **(6)**

Výpočty Q1 a Q2 sa pre niektoré naplnenia r líšia hneď po tej prvej projekcii a následnom odstránení duplikátov. To odstránenie duplikátov spôsobí, že C bude po agregácii rôzne v Q1 a Q2.

Napríklad, pre r(X, Y, Z, V, W) = {[0, 0, 1, 0, 0], [0, 0, 2, 0, 0], [0, 0, 3, 0, 0], [0 ,0, 4, 0, 0]} sú medzivýsledky takéto (q1 a q2 sú výsledky dotazov Q1 resp. Q2):
r1(X, Y, Z, V) = {[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0 ,0, 0, 0]},
r2(X, Y, Z, V) = {[0, 0, 1, 0], [0, 0, 2, 0], [0, 0, 3, 0], [0 ,0, 4, 0]},
r3(X, Y, C) = {[0, 0, 4]},
r4(X, Y, C) = {[0, 0, 4]},
r5(X) = {[0]},
q1 = {[0]}.

s1(X, Y, V) = {[0, 0, 0], [0, 0, 0], [0, 0, 0], [0 ,0, 0]},
s2(X, Y, V) = {[0, 0, 0]},
s3(X, Y, C) = {[0, 0, 1]},
s4(X, Y, C) = {},
r5(X) = {},
q2 = {}.

3.a) Čo znamená, že dekompozícia relačnej schémy [r, F] zachováva (všetky) funkčné závislosti [r, F]? Aký praktický zmysel má snaha o zachovanie funkčných závislostí pri dekompozícii relačnej schémy? **(6)**

Podľa definície, dekompozícia [r, F] do $[r_1, F_1], \ldots, [r_N, F_N]$ zachováva (všetky) funkčné závislosti, keď $F^+ = (F_1 \cup \ldots \cup F_N)^+$.

Dekompozíciou nevznikajú žiadne nové funkčné závislosti, t.j. pre ľubovoľnú dekompozíciu platí $F^+ \supseteq (F_1 \cup \ldots \cup F_N)^+$.

Ak dekompozícia zachováva funkčné závislosti, tak je možné zabezpečiť automatickú ochranu integrity dát definíciou **lokálneho** obmedzenia (constraintu) resp. obmedzení pre každú reláciu dekompozície. V SQL systémoch je možné každú funkčnú závislosť z F_i preložiť do klauzy CONSTRAINT v CREATE TABLE relácie r_i. Systém potom automaticky abortuje každú transakciu, ktorá pri aktualizácii r_i porušuje niektorý constraint v r_i.

Automatická ochrana integrity dát je komplikovanejšia a drahšia v prípade dekompozície, ktorá niektoré funkčné závislosti láme (platnosť zlomených funkčných závislostí je potrebné overovať na joine viacerých relácií dekompozície).

b) Uveďte algoritmus (pseudokód) s polynomiálnou časovou zložitosťou, ktorý pre ľubovoľnú relačnú schému [r, F] a jej dekompozíciu $[r_1, F_1], \ldots, [r_N, F_N]$ overí či daná dekompozícia zachováva (všetky) funkčné závislosti [r, F]. Zdôvodnite. **(6)**

Budeme predpokladať, že dekompozícia je korektná, t.j. že $F^+ \supseteq (F_1 \cup \ldots \cup F_N)^+$ (hoci aj toto sa dá overiť v polynomiálnom čase). Podľa definície stačí overiť či $F^+ \subseteq (F_1 \cup \ldots \cup F_N)^+$. K tomu nie je potrebné počítať uzáver množiny funkčných závislostí.
Stačí pre každú funkčnú závislosť $X \rightarrow Y$ z F overiť, že je v $(F_1 \cup \ldots \cup F_N)^+$. To znamená vypočítať uzáver množiny atribútov X s použitím funkčných závislostí z F a otestovať či ten uzáver je nadmnožinou Y.

```
ok = TRUE;
foreach (X→Y ∈ F)
{
        C = X+, kde X+ sa počíta vzhľadom na F1 ∪ … ∪ FN;
        if (not C ⊇ Y)
        {
                print(„dekompozícia NEzachováva funkčné závislosti“);
                ok = FALSE;
                break;
        }
}
if (ok)
{
        print(„dekompozícia zachováva (všetky) funkčné závislosti“);
};
```

Uzáver množiny atribútov sa počíta v polynomiálnom čase vzhľadom na počet atribútov a počet funkčných závislostí v F.

4. Neutriedená relácia r je uložená na disku v B blokoch, všetky bloky sú maximálne naplnené. Bloky na disku sú rovnako veľké ako bloky v RAM. Na sekvenčné médium treba uložiť súbor s utriedenou reláciou r.
a) Vyjadrite ako funkciu B minimálny počet blokov, ktoré treba v RAM rezervovať pre externé triedenie merge-sort, aby celkový počet vstupno-výstupných diskových operácií nepresiahol 2B. Vysvetlite. **(6)**

Ak je v RAM voľných B blokov, tak sa v prvej fáze merge-sort všetkých B blokov relácie r načíta do RAM. Relácia r sa v RAM utriedi a následne sa B blokov zapíše do výstupného súboru. Celkovo sa urobí 2B diskových operácií.

Ak je v RAM menej než B voľných blokov, tak k horeuvedeným 2B diskovým operáciám pribudne réžia zápisu a opätovného čítania utriedených behov.

Minimálna veľkosť RAM, s ktorou je možné danú požiadavku splniť, je B blokov.

b) Vyjadrite ako funkciu B minimálny počet blokov, ktoré treba v RAM rezervovať pre externé triedenie merge-sort, aby celkový počet vstupno-výstupných diskových operácií nepresiahol 4B. Vysvetlite. **(6)**

Nech M označuje veľkosť rezervovanej RAM v blokoch.

V prvej fáze merge-sort sa prečíta B blokov a zapíše B blokov. Výstupom je $\lceil B / M \rceil$ utriedených behov.

Aj pri každom opakovaní druhej fázy sa prečíta B blokov a zapíše B blokov. Keďže počet diskových operácií má byť najviac 4B, druhá fáza (spájanie utriedených behov) sa môže vykonať maximálne raz. V každej iterácii druhej fázy sa v RAM rezervuje 1 blok pre výstup a 1 blok pre čítanie každého behu). To znamená, že behov po prvej fáze môže byť nanajvýš M – 1, aby sa v jednej iterácii druhej fázy mohli všetky spojiť do jedného behu. Musí teda platiť $B / M <= M - 1$. Keďže M chceme minimalizovať, tak musí platiť $B / M = M - 1$. To je kvadratická rovnica s neznámou M. Riešením s $M > 0$ je $M = (1 + \sqrt{1 + 4B}) / 2$. Toto M treba ešte zaokrúhliť nahor, lebo počet blokov je celé číslo.

Minimálna veľkosť RAM, s ktorou je možné danú požiadavku splniť, je $\lceil (1 + \sqrt{1 + 4B}) / 2 \rceil$ blokov.