29/1/2024 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 [P, K] také, že pijan P pri každej návšteve krčmy K vypil niektorý zo svojich obľúbených alkoholov. Predpokladajte, že P navštívil K aspoň raz.

Datalog:

```
answer(P, K) ←
        navstivil(_, P, K),
        not niekedy_nevypil(P, K).

niekedy_nevypil(P, K) ←
        navstivil(I, P, K),
        not vypil_oblubeny(I).

vypil_oblubeny(I) ←
        vypil(I, A, _),
        lubi(P, A).
```

Relačná algebra:

vypil_oblubeny = π_{Idn} (vypil ⋈ lubi);

niekedy_nevypil = $\pi_{\text{Pijan, Krcma}}$ (navstivil ▷ vypil_oblubeny);

/* answer */

$\pi_{\text{Pijan, Krcma}}$ (navstivil) ▷ niekedy_nevypil;

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na štvorice [P, K, C, T], kde C je počet druhov alkoholov a T je celkové množstvo alkoholov, ktoré pijan P ľúbi a vypil ich v krčme K. Výsledok nemá obsahovať štvorice s C = T = 0.

Datalog:

```
answer(P, K, C, T) ←
        subtotal(pocet(P, K, A), [P, K], [C = count(A)]),
        subtotal(mnozstvo(_, P, K, M), [P, K], [T = sum(M)]).

pocet(P, K, A) ←
        lubi(P, A),
        navstivil(I, P, K),
        vypil(I, A, _).

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

SQL:

```
select n.Pijan, n.Krcma, count(distinct v.Alkohol) as C, sum(v.Mnozstvo) as T
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
```

2. Uvažujte relačnú schému r(A, B, C, D, E, F) s funkčnými závislosťami
AB → CDF, ACF → B, AE → B, CD → ABF, CE → D, CEF → ABD.

a) Dekomponujte r do tretej normálnej formy, bezstratovo a so zachovaním všetkých funkčných závislostí. **(6)**

Nájdime nejaké minimálne pokrytie vstupnej množiny funkčných závislostí.
Kánonické funkčné závislosti:
AB → C, AB → D, AB → F, ACF → B, AE → B, CD → A, CD → B, CD → F, CE → D, CEF → A, CEF → B, CEF → D

Po minimalizácii ľavých strán funkčných závislostí:
AB → C, AB → D, AB → F, ACF → B, AE → B, CD → A, CD → B, CD → F, CE → D, CE → A, CE →B, CE → D
Po vynechaní redundantných funkčných závislostí:
AB → C, AB → D, ACF → B, AE → B, CD → A, CD → F, CE → D
Toto je nejaké minimálne pokrytie.

Syntéza 3NF z tohto minimálneho pokrytia:
r1(A, B, C, D),
r2(A, B, C, F),
r3(A, B, E) /* ABE je nadkľúč v r */
r4(A, C, D, F),
r5(C, D, E)
Táto dekompozícia je v 3NF, je bezstratová a zachováva všetky funkčné závislosti.

b) Dekomponujte r do Boyce-Coddovej normálnej formy, bezstratovo. Vyhnite sa zbytočnému rozbitiu funkčných závislostí. **(6)**

Začnime s 3NF dekompozíciou z úlohy a).
Všetky platné netriviálne funkčné závislosti v r majú aspoň 2 atribúty na ľavej strane.
Tým pádom r3 a r5 sú v BCNF,
V r1 žiadna platná netriviálna funkčná závislosť s atribútmi AB na ľavej strane neohrozuje BCNF, lebo AB je nadkľúč v r2. Iné netriviálne funkčné závislosti v r1 neplatia.
V r2 je AB nadkľúč, a tiež ACF. Netriviálne funkčné závislosti s inými ľavými stranami v r2 neplatia.
V r4 je CD nadkľúč, a tiež ACF. Netriviálne funkčné závislosti s inými ľavými stranami v r2 neplatia.

3NF dekompozícia z úlohy a) je zároveň v BCNF.

Alternatívna BCNF dekompozícia, bezstratová a zachovávajúca všetky funkčné závislosti:
r1(A, B, E)
r2(A, B, C, D, F) /* AB je kľúč v r2 */

3. Uvažujte Datalogový program nad extenzionálnou reláciou r(X, Y):
answer(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) Rozhodnite či výpočet dotazu ?- answer(1, Y) naivnou evaluáciou skončí pre ľubovoľné naplnenie relácie r. **(6)**

Graf závislostí IDB predikátov (predikátov, ktoré sú definované v programe):

$$\begin{tikzcd} \text{answer} & q \arrow[l, "\neg"'] \arrow[loop right] \end{tikzcd}$$

Graf závislostí IDB predikátov neobsahuje cyklus s negovanou hranou. Tým pádom daný program je stratifikovateľný. Naivná evaluácia skončí pre akékoľvek naplnenie relácie r. (Stratum q je 1, stratum answer je 2.)

b) Zapíšte výpočet dotazu ?- answer(1, Y) naivnou evaluáciou v relačnej algebre, a vypočítajte výsledok dotazu pre reláciu
r(X, Y) = {[0, 0], [0, 1], [1, 1], [2, 2], [2, 0], [3, 0], [4, 5]}. **(6)**

answer = {};
q = {};
ϕ (

$q = (\pi_{r1.X,\, r2.Y}\, (\rho_{r1}\, (r) \times \rho_{r2}\, (r)) - r) \cup (r \bowtie \rho_{q(Y,\, X)}\, (q))$;

answer = $r \triangleright q$;

);
/* query */
$\pi_Y\, (\sigma_{X\,=\,1}\, (answer))$;

Tento výpočet sa dá optimalizovať tak, že pred cyklus sa predradí konštantná časť q; a answer sa vypočíta raz, mimo cyklu:
$q = (\pi_{r1.X,\, r2.Y}\, (\rho_{r1}\, (r) \times \rho_{r2}\, (r)) - r)$;
ϕ (

$q = q \cup (r \bowtie \rho_{q(Y,\, X)}\, (q))$;

);
answer = $r \triangleright q$;
/ query */*
$\pi_Y\, (\sigma_{X\,=\,1}\, (answer))$;

Simulácia optimalizovaného výpočtu pre dané naplnenie r:
Po inicializácii:
q = {[0, 2], [0, 5], [1, 0], [1, 2], [1, 5], [2, 1], [2, 5], [3, 1], [3, 2], [3, 5], [4, 0], [4, 1], [4, 2]}
Po prvej iterácii ϕ:
q = {*[0, 1]*, [0, 2], [0, 5], [1, 0], [1, 2], [1, 5], *[2, 0]*, [2, 1], [2, 5], [3, 1], [3, 2], [3, 5], [4, 0], [4, 1], [4, 2]}
Po druhej iterácii ϕ:
q = {[0, 1], [0, 2], [0, 5], [1, 0], [1, 2], [1, 5], [2, 0], [2, 1], [2, 5], [3, 1], [3, 2], [3, 5], [4, 0], [4, 1], [4, 2]}
Po priradení do answer:
answer = {[0, 0], [1, 1], [2, 2], [3, 0], [4, 5]}
Po finálnej selekcii a projekcii dostávame **odpoveď na dotaz: {[1]}.**

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 externý merge-sort, aby celkový počet vstupno-výstupných diskových operácií nepresiahol 2B. Vysvetlite. **(6)**

Keď sa v RAM rezervuje B blokov, tak sa do RAM zmestí celá relácia r. Každý blok r sa raz prečíta, a po utriedení v RAM (prvá fáza merge-sort) sa rovnaký počet blokov zapíše do výsledného behu. To je 2B vstupno-výstupných operácií.
S menšou RAM sa to v danom limite urobiť nedá, lebo po prvej fáze (s 2B vstupno-výstupnými operáciami) vzniknú aspoň dva behy, ktoré treba spojiť v druhej fáze. To znamená dodatočné vstupno-výstupné operácie.
Minimálny požadovaný počet blokov RAM je B.

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

So 4B vstupno-výstupnými diskovými operáciami si môžeme dovoliť okrem prvej fázy merge-sort aj druhú fázu. Druhá fáza smie bežať nanajvýš raz, lebo každé jej opakovanie znamená aspoň 2B operácií (každý beh treba načítať, a zapísať rovnaký počet blokov do výstupného behu resp. behov).

Ak R je veľkosť rezervovanej RAM, po prvej fáze sa vytvorí $\lceil B / R \rceil$ behov. Tento počet behov smie byť nanajvýš $R - 1$, lebo ich treba zlúčiť v jednej iterácii druhej fázy:
$\lceil B / R \rceil \leq R - 1$
Zanedbáme to zaokrúhlenie nahor:
$B / R \leq R - 1$
$R (R - 1) - B \geq 0$
Toto je kvadratická nerovnica s neznámou R. Jej prípustné riešenie je $R \geq 1/2 + \mathrm{sqrt}(1 + 4B) / 2$.
Zaujíma nás minimálne R, pri ktorom nastáva rovnosť. Keďže R má byť celé číslo, a keďže predtým sme optimisticky zanedbali zaokrúhlenie, zaokrúhlime R konzervatívne nahor:
$R = 1 + \lceil \mathrm{sqrt}(1 + 4B) / 2 \rceil$
Toto je veľkosť RAM, s ktorou externý merge-sort urobí nanajvýš 4B vstupno-výstupných operácií. Kvôli tým zaokrúhľovaniam môže byť možno o 1 blok väčšia než nutné minimum.