10/2/2015 Úvod do databáz, skúškový test, max 25 bodov, 90 min

1. Daná je databáza (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) Nájdite pijanov, ktorí ľúbia aspoň jeden taký alkohol, ktorý čapuje každá krčma, v ktorej ten pijan niečo vypil.
Sformulujte dotaz v relačnom kalkule **(2)**, relačnej algebre **(2)** a Datalogu **(2)**.

Relačný kalkul:

```
{P:
        ∃A
        lubi(P, A) ∧
        ¬
        (       /* vypil_necapuje(P, A) */
                ∃I ∃K ∃A2 ∃M
                lubi(P, A) ∧    /* safety */
                navstivil(I, P, K) ∧
                vypil(I, A2, M) ∧
                ¬
                capuje(K, A)
        )
}
```

Datalog:

```
answer(P) ←
        lubi(P, A),
        not vypil_necapuje(P, A).

vypil_necapuje(P, A) ←
        lubi(P, A),     /* safety */
        navstivil(I, P, K),
        vypil(I, _, _),
        not capuje(K, A).
```

Relačná algebra:

$vypil\_necapuje := \pi_{Pijan,\ Alkohol}$ (

$(\pi_{Krcma,\ Alkohol}$ (lubi ⋈ navstivil ⋈ $P_{v(Idn,\ Alkohol2,\ Mnozstvo)}$ (vypil)) − capuje) ⋈ lubi)

$answer := \pi_{Pijan}$ (lubi − vypil_necapuje)

S použitím operátora antijoin, ◁▷:

$vypil\_necapuje := \pi_{Krcma,\ Alkohol}$ ((lubi ⋈ navstivil ⋈ $P_{v(Idn,\ Alkohol2,\ Mnozstvo)}$ (vypil)) ◁▷ capuje)

$answer := \pi_{Pijan}$ (lubi ◁▷ vypil_necapuje)

b) Nájdite dvojice [K, N], kde N je počet pijanov, ktorí počas niektorej svojej návštevy krčmy K vypili viacej než 7 druhov alkoholov. Sformulujte dotaz v Datalogu **(2)** a SQL **(2)**. Vo výsledku nemajú byť dvojice s N=0.

Datalog:

```
answer(K, N) ←
        subtotal(vypil7(P, K), [K], [N = count(P)]).

vypil7(P, K) ←
        subtotal(v(P, K, I, A), [P, K, I], [C = count(A)]),
        C > 7.

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

SQL:

```
create temporary table vypil7 as
select n.Pijan, n.Krcma
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Pijan, n.Krcma, n.Idn
having count(v.Alkohol) > 7

/* answer */
select v7.Krcma, count(distinct v7.Pijan) as N
from vypil7 v7
group by v7.Krcma
```

2. Uvažujte reláciu r(A, B, C, D, E) s funkčnými závislosťami
A→CE, ACD→BE, BC→D, BE→AC.
a) Uveďte príklad konkrétneho naplnenia relácií r, r1 a r2, ktoré demonštruje, že dekompozícia r do r1(A, C, D, E), r2(A, B, E) sa nespája bezstratovo (t.j. nie je bezstratová). Vysvetlite. **(2)**

Simulácia algoritmu chase:

	A	B	C	D	E
ACDE	a1		a3	a4	a5
ABE	a1	a2			a5

A→CE

	A	B	C	D	E
ACDE	a1		a3	a4	a5
ABE	a1	a2	a3		a5

Tu algoritmus chase končí. Žiaden riadok tabuľky neobsahuje všetky symboly a*, takže táto dekompozícia skutočne nie je bezstratová.

Použijeme nasledujúce naplnenia relácií podľa finálnej tabuľky chase:
r(A, B, C, D, E) = {[a1, b12, a3, a4, a5], [a1, a2, b23, b24, a5]}
r1(A, C, D, E) = {[a1, a3, a4, a5], [a1, b23, b24, a5]}
r2(A, B, E) = {[a1, b12, a5], [a1, a2, a5]}

r1 ⋈ r2 = {[a1, b12, a3, a4, a5], [a1, a2, a3, a4, a5], [a1, a2, b23, b24, a5], [a1, a2, b23, b24, a5]} =
/* po odstránení duplikátov */
{[a1, b12, a3, a4, a5], [a1, a2, a3, a4, a5], [a1, a2, b23, b24, a5]}.

Pre toto konkrétne naplnenie relácií r1 ⋈ r2 ≠ r.

b) Uveďte bezstratovú dekompozíciu r, ktorá je v tretej normálnej forme a zároveň zachováva všetky funkčné závislosti. **(2)**

Po minimalizácii ľavých strán kánonických funkčných závislostí:
A→C, A→E, AD→B, A→E, BC→D, BE→A, BE→C.

Po odstránení redundantných funkčných závislostí dostávame (nejaké) minimálne pokrytie danej množiny funkčných závislostí:
A→C, A→E, AD→B, BC→D, BE→A.

3NF dekompozícia (relácie syntetizované z funkčných závislosti s rovnakou ľavou stranou môžeme spojiť do jednej):
(A, C)
(A, B, D) /* {A, B, D} je nadkľúč r */
(B, C, D)
(A, B, E)

Alebo (atribúty A, B, D, E sú kľúčové v r):
(A, B, D, E), (A, C), (B, C, D)

c) Nájdite všetky kľúče r. **(1)**

ABCDE

–A: BCDE

–B: CDE

+B: BCDE

–E: BCD

+E: **BE**

+A: ABD

–B: AD

–D: A

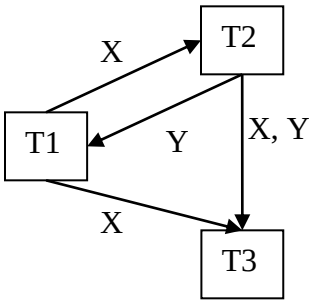
+D: **AD**

+B: **AB**

Kľúče relácie r sú AB, AD, BE. Iné kľúče nie sú.

3. Uvažujte rozvrh r1(X), w1(X), r2(Y), w2(Y), r3(Y), r2(X), r1(Y), w3(X).
Vložte do rozvrhu commity všetkých troch transakcií (použite operácie c1, c2, c3 na ukončenie transakcií) tak, aby výsledný rozvrh bol
a) nie konflikt-sériovateľný a zároveň nie obnoviteľný; **(1)**

Graf konfliktov:



Graf konfliktov obsahuje cyklus T1→T2→T1. To znamená, že po ľubovoľnom doplnení commitov T1, T2 a T3 výsledný rozvrh bude nie konflikt-sériovateľný.

Pre splnenie požiadavky, aby výsledný rozvrh bol nie obnoviteľný, stačí sa zamerať na nejaký dirty-read, povedzme w2(Y)→r3(Y) a doplniť príslušné dva commity neskôr, v poradí c3→c2. Toto znemožní obnovu, lebo ak sa hneď po commitovaní T3 systém reštartuje, nie je vo všeobecnosti možné urobiť UNDO T2 a zároveň vrátiť následky toho dirty-read (ktoré môžu spočívať v zápise w3(X) do databázy, kde zapisovaná hodnota X môže závisieť od prečítanej hodnoty Y).

Rozvrh s požadovanými vlastnosťami:
r1(X), w1(X), r2(Y), w2(Y), r3(Y), r2(X), r1(Y), w3(X), c1, c3, c2

b) konflikt-sériovateľný, ale nie obnoviteľný; **(1)**

Toto sa nedá urobiť. Z riešenia úlohy a) vyplýva, že daný rozvrh akokoľvek doplnený o c1, c2, c3 bude nie konflikt-sériovateľný.

c) konflikt-sériovateľný, obnoviteľný, ale nie vyhýbajúci sa kaskádovým abortom; **(1)**

Toto sa nedá urobiť. Z riešenia úlohy a) vyplýva, že daný rozvrh akokoľvek doplnený o c1, c2, c3 bude nie konflikt-sériovateľný.

d) konflikt-sériovateľný, vyhýbajúci sa kaskádovým abortom, ale nie striktný. **(1)**
V riešeniach zdôvodnite, prečo má rozvrh doplnený o commity požadovanú vlastnosť, resp. prečo sa commity nedajú doplniť tak, aby požadovanú vlastnosť mal.

Toto sa nedá urobiť. Z riešenia úlohy a) vyplýva, že daný rozvrh akokoľvek doplnený o c1, c2, c3 bude nie konflikt-sériovateľný.

4. Relácia zoznam(Priezvisko, Meno, Cislo, ...) má 1 000 000 záznamov. Relácia je uložená v diskovom poli s blokmi veľkosti 4kB. V každom bloku je uložených 10 záznamov. K dispozícii nie je žiaden index. Stredná doba prenosu diskového bloku (do RAM aj opačne) je 0.1 s. Pre vykonanie SQL dotazu
select z.Priezvisko, z.Meno, z.Cislo from zoznam z order by z.Priezvisko, z.Meno
je rezervovaných 200 blokov v RAM (bloky v RAM a na disku sú rovnako veľké).
a) Popíšte organizáciu pamäte a jednotlivé fázy (na úrovni vstupov a výstupov) algoritmu fyzického operátora merge-sort pri vykonávaní daného dotazu. **(2)**

Nech M je počet blokov rezervovaných v RAM. V prvej fáze sa použije M blokov v RAM pre čítanie relácie r. Po utriedení záznamov v RAM sa všetky bloky RAM zapíšu na disk do behu (utriedeného súboru). Toto sa opakuje, kým sa nedočíta celá relácia r.

V druhej fáze sa behy zlučujú do jedného výsledného behu. M–1 blokov v RAM sa použije pre vstup behov (1 blok pre 1 beh), 1 blok sa použije ako výstupný. Výstupný blok sa po naplnení pripája k výslednému behu. Po zlúčení vstupných (krátkych) behov do jedného výsledného (dlhého) behu tie vstupné behy zaniknú.

Druhá fáza sa opakuje, až kým neostane jediný beh. Ten jediný beh obsahuje utriedenú reláciu r.

(Zanedbávame optimalizácie, ktoré využívajú obsah RAM z predošlej fázy.)

b) Odhadnite čas vykonania daného dotazu. Svoj odhad zdôvodnite. **(2)**

Relácia r je uložená v 100 000 blokoch.

V prvej fáze sa všetky bloky prečítajú a rovnaký počet blokov sa zapíše do utriedených behov:
500 behov (= 100 000 / 200) dĺžky 200 blokov, 200 000 I/O operácií.

V druhej fáze sa postupne prečíta 199 behov dĺžky 200 blokov, ktoré sa zlučujú do jedného behu. Prečíta sa 39 800 blokov (= 199 * 200), rovnaký počet sa zapíše na disk:
301 behov dĺžky 200 blokov, 1 beh dĺžky 39800 blokov, 79600 I/O operácií (= 2 * 39800).

Druhá fáza sa zopakuje. Prečíta sa 199 behov dĺžky 200 blokov, ktoré sa zlučujú do jedného behu. Prečíta sa 39 800 blokov (= 199 * 200), rovnaký počet sa zapíše na disk:
102 behov dĺžky 200 blokov, 2 behy dĺžky 39800 blokov, 79600 I/O operácií (= 2 * 39800).

Druhá fáza sa zopakuje. Počet behov je už menší než 200, takže sa všetky zlúčia do jedného behu. Prečíta sa 100 000 blokov (= 102 * 200 blokov + 2 * 39800), rovnaký počet sa zapíše na disk.
1 beh dĺžky 100 000 blokov, 200 000 operácií.

Dokopy sa urobí zhruba 479 600 I/O operácií (= 2 * 200 000 + 2 * 79 600), čo trvá **55 920 sekúnd**, čo je **zhruba 15.5 hodín.** (Zanedbávame výpočty v RAM.)

c) Ak sa namiesto 200 blokov rezervuje v RAM 2000 blokov (teda desaťkrát viacej), koľkokrát sa tým urýchli čas vykonávania daného dotazu? Zdôvodnite. **(2)**

V prvej fáze sa všetky bloky prečítajú a rovnaký počet blokov sa zapíše do utriedených behov:
50 behov (= 100 000 / 2000) dĺžky 200 blokov, 200 000 I/O operácií.

V druhej fáze sa všetky bloky zlúčia do jedného behu. Prečíta sa 100 000 blokov, rovnaký počet sa zapíše na disk:
1 beh dĺžky 100 000 blokov, 200 000 operácií.

Dokopy sa urobí zhruba 400 000 I/O operácií (= 2 * 200 000), čo trvá **40 000 sekúnd**, čo je **zhruba 11 hodín**.
Výpočet sa urýchli zhruba 1.4 krát. (Zanedbávame výpočty v RAM.)