18/1/2023 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 [P, K] také, že pijan P pri každej svojej návšteve krčmy K vypil rovnakú množinu alkoholov, pričom P navštívil K aspoň dvakrát.

Datalog:

```
answer(P, K) ←
        navstivil(I1, P, K),
        navstivil(I2, P, K),
        not I1 = I2,
        not niekedy_vypil_rozne(P, K).

niekedy_vypil_rozne(P, K) ←
        navstivil(I1, P, K),
        vypil(I1, A, _),
        navstivil(I2, P, K),
        not v(I2, A).

v(I, A) ←
        vypil(I, A, _).
```

Relačná algebra:

$$\text{niekedy\_vypil\_rozne} := \pi_{Pijan,\ Krcma}\ ((navstivil \bowtie vypil \bowtie_{Pijan = P2\ \wedge\ Krcma = K2}\ \rho_{I2,\ P2,\ K2}\ (navstivil)) \triangleright_{I2 = IV\ \wedge\ Krcma = K\ \wedge\ Alkohol = AV}\ \rho_{IV,\ AV,\ MV}\ (vypil));$$

/* answer */

$$\pi_{Pijan,\ Krcma}\ ((navstivil \bowtie_{Pijan = P2\ \wedge\ Krcma = K2 \wedge Idn \neq I2}\ \rho_{I2,\ P2,\ K2}\ (navstivil)) \triangleright_{Pijan = PV\ \wedge\ Krcma = KV}\ \rho_{PV,\ MV}\ (\text{niekedy\_vypil\_rozne}))$$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na také dvojice [P, K], že krčma K čapuje aspoň 7 alkoholov, ktorý pijan P ľúbi; a zároveň každý alkohol, ktorý P ľúbi, vypil P pri najviac 9 návštevách krčmy K.

Datalog:

```
answer(P, K) ←
        subtotal(capuje_lubi(P, K, A), [P, K], [C = count(A)]),
        C >= 7,
        not niecoho_pricasto(P, K).

capuje_lubi(P, K, A) ←
        capuje(K, A),
        lubi(P, A).

niecoho_pricasto(P, K) ←
        subtotal(v(I, P, K, A), [P, K, A], [C = count(I)]),
        I >= 9.

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

SQL:

```
with
pocty_capuje_lubi as
(
select l.Pijan, c.Krcma, C as count(Alkohol)
from capuje c, lubi l
where l.Alkohol = c.Alkohol
group by l.Pijan, c.Krcma
),
pocty_pricasto as
(
select n.Pijan, n.Krcma, C as count(n.Idn)
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Pijan, n.Krcma, v.Alkohol
)
/* main */
select pcl.Pijan, pcl.Krcma
from pocty_capuje_lubi pcl
where pcl.C >= 7 and not exists (
        select *
        from pocty_pricasto pp
        where pcl.Pijan = pp.Pijan and pcl.Krcma = pp.Krcma and pp.C >= 9)
```

2.
a) Uvažujte reláciu r(A, B, C, D, E) s funkčnými závislosťami
A→CE, ADE→BCDE, BC→D, BE→AC. Rozhodnite, či sa dekompozícia r1(A, C, D, E), r2(A, B, E) spája bezstratovo. Ak áno, zdôvodnite. Ak nie, uveďte príklad konkrétneho naplnenia relácií r, r1 a r2, ktoré je v rozpore s definíciou. Vysvetlite. **(6)**

Definícia: Dekompozícia relačnej schémy [r, F] do $[r_1, F_1]$, ..., $[r_N, F_N]$ je bezstratová (spája sa bezstratovo), keď pre všetky naplnenia r (spĺňajúce integritné obmedzenia dané funkčnými závislosťami F^+) platí
$r = r_1 \bowtie r_2 \bowtie ... \bowtie r_N$.

Použijeme test z prednášky pre dekompozíciu do dvoch relácií. Spoločné atribúty r1 a r2 sú {A, E}. Uzáver $\{A, E\}^* = \{A, C, E\}$. Keďže tento uzáver neobsahuje všetky atribúty r1 ani r2, {A, E} nie je nadkľúčom v r1 ani v r2. Takže **NIE**: **uvedená dekompozícia sa nespája bezstratovo**.

Konkrétny kontrapríklad:
Uvažujme populáciu r(A, B, C, D, E) = **{[0, 0, 0, 0, 0], [0, 1, 0, 1, 0]}**, ktorá spĺňa dané funkčné závislosti.
Potom r1(A, C, D, E) = {[0, 0, 0, 0], [0, 0, 1, 0]}, r2(A, B, E) = {[0, 0, 0], [0, 1, 0}.
(r1 ⋈ r2) (A, B, C, D, E) = {[0, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 0, 1, 0], [0, 1, 0, 1, 0]}}.
Napríklad táto populácia je v rozpore s vyššieuvedenou definíciou, lebo neplatí r1 ⋈ r2 = r.

b) Rozhodnite, či existuje dekompozícia relačnej schémy r z úlohy a), ktorá sa spája bezstratovo, je v tretej normálnej forme a zároveň zachováva všetky funkčné závislosti. Ak ÁNO, uveďte príklad takej dekompozície. Ak NIE, zdôvodnite. **(6)**

ÁNO. Dekompozícia s požadovanými vlastnosťami existuje pre každú relačnú schému, teda aj pre túto. Na nájdenie požadovanej dekompozície použijeme algoritmus z prednášky, ktorý vyžaduje nájdenie (nejakého) minimálneho pokrytia danej množiny funkčných závislostí.

Po kanonizácii funkčných závislostí:
A→C, A→E, ADE→B, ADE→C, ADE→D, ADE→E, BC→D, BE→A, BE→C

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

Po vynechaní redundantných funkčných závislostí:
A→C, A→E, AD→B, BC→D, BE→A
Toto je minimálne pokrytie danej množiny funkčných závislostí.

3NF dekompozícia z minimálnej pokrytia, ktorá má všetky požadované vlastnosti:
(A, C), (A, B, E), (A, B, D), (B, C, D)

3.
a) Popíšte čo najpresnejšie všeobecný algoritmus obnovy, ktorý sa používa pri opätovnom štarte databázového systému. **(6)**

Všeobecný algoritmus obnovy prechádza log-file najskôr zostupne (od konca log-file k začiatku). Počas tohto prechodu aktualizuje zoznamy redo_list a undo_list (oba sú na začiatku prázdne), ktoré sú na začiatku prázdne, a zároveň robí UNDO operácie pre transakcie z undo_list.
Keď príde na začiatok log-file, začne vzostupný prechod, pri ktorom vykonáva REDO operácie pre transakcie z redo_list, a aktualizuje redo_list. Keď je redo_list prázdny, systém normálnu prevádzku.

Aktualizácia redo_list a undo_list počas zostupného prechodu:
Po prečítaní <T, commit> sa T pridá do redo_list.
Ak transakcia T nie je v redo_list ani v undo_list, tak po prečítaní záznamu o zápise alebo o štarte transakcie T sa transakcia T pridá do undo_list.

Aktualizácia redo_list počas vzostupného prechodu:
Ak transakcia T je v redo_list, tak po prečítaní <T commit> sa T odstráni z redo_list.

b) Nech pri opätovnom štarte databázového systému obsahuje log-file nasledujúce záznamy (zoradené od najstaršieho po najnovší):
<T0, start>, <T0, A, 20, 100>, <T1, start>, <T0, commit>, <T2, start>,
<T2, A, 100, 50>, <T2, B, 200, 250>, <T1, A, 50, 70>, <T1, commit>.
Uveďte sekvenciu zápisov do databázy, ktoré sa vykonajú počas obnovy v tomto konkrétnom prípade. **(6)**

/* Zostupný prechod */
B := 200 (undo T2)
A := 100 (undo T2)
/* Vzostupný prechod */
A := 100 (redo T0)
A := 70 (redo T1)

4. Relácia r(X, Y, Z, ...) obsahuje 5 000 000 záznamov. Relácia je uložená v diskovom poli s blokmi veľkosti 4kB, v každom bloku je uložených 10 záznamov. Stredná doba prenosu diskového bloku (do RAM aj opačne) je 100 ms (=0,1 s). Pre vykonanie SQL dotazu *select r.X, r.Y, r.Z from r order by r.X, r.Y*
je rezervovaných 500 blokov v RAM, bloky v RAM a na disku sú rovnako veľké.
a) Popíšte organizáciu pamäte a priebeh výpočtu (stačí na úrovni vstupov a výstupov) fyzického operátora merge-sort pri vykonávaní daného dotazu. **(6)**

Relácia r je uložená v 500 000 blokoch.

V prvej fáze merge-sort sa všetkých 500 blokov RAM naplní prvými 500 blokmi r. Záznamy v RAM sa utriedia a na disk sa zapíše prvý utriedený beh. To isté sa urobí s nasledujúcimi 500 blokmi r. Toto sa opakuje, až kým sa neprečítajú všetky bloky r.
Výsledkom prvej fázy je 500 000 / 500 = 1 000 (utriedených) behov dĺžky 500 blokov.
Keďže každý blok r bol raz prečítaný do RAM a raz zapísaný na disk, dokopy sa urobilo
*2 * 500 000 = 1 000 000 I/O operácií.*

V druhej fáze sa 1 blok RAM rezervuje pre výstup. Každý zo zvyšných 499 blokov sa použije pre vstup jedného z prvých 499 behov. Vstupné behy sa zlúčia do jedného výstupného behu, ktorý sa priebežne ukladá na disk. Po ukončení zlučovania sa vstupné behy zahodia.
*Výsledkom je jeden beh dĺžky 499 * 500 blokov = 249 500 blokov.*
*Každý z týchto blokov bol raz prečítaný do RAM a raz zapísaný na disk, teda dokopy sa urobilo 2 * 249 500 = 499 500 I/O operácií.*
Toto sa zopakuje s nasledujúcimi 499 behmi.
*Výsledkom je jeden beh dĺžky 499 * 500 blokov = 249 500 blokov.*
*Každý z týchto blokov bol raz prečítaný do RAM a raz zapísaný na disk, teda dokopy sa urobilo 2 * 249 500 = 499 500 I/O operácií.*
Na disku sú v tomto momente uložené 2 behy dĺžky 249 500 blokov a 2 behy z prvej fázy dĺžky 500 blokov. Tieto 4 behy sa zlúčia do jedného výstupného behu, ktorý sa priebežne ukladá na disk. Po ukončení zlučovania sa vstupné behy zahodia.
Na disku ostal uložený jediný beh, ktorý je výsledkom dotazu. Tým výpočet končí.
*Počas tohto posledného zlučovania sa každý blok každého vstupného behu raz prečíta do RAM a raz zapíše na disk, t.j. urobí sa 2 * 500 000 = 1 000 000 I/O operácií.*

b) Odhadnite čas vykonania daného dotazu. Svoj odhad zdôvodnite. **(6)**

Celkový počet I/O operácií je (viď riešenie úlohy a), časti sádzané kurzívou)
1 000 000 + 2 * 499 500 + 1 000 000 = 2 999 000, čo pri danej rýchlosti vykonávania I/O operácií trvá 299 900 sekúnd, t.j. **približne 83 hodín**.