14/1/2016 Úvod do databáz, skúškový test, max 25 bodov, 90 min

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) Nájdite trojice [P, A1, A2] také, že pijan P ľúbi aspoň jeden z alkoholov A1 a A2; a zároveň A1 aj A2 sa niekde čapujú (nie nutne v rovnakej krčme); a zároveň P vypil A1 aj A2 v každej krčme, ktorú navštívil (nie nutne oba pri rovnakej návšteve). Sformulujte dotaz v Datalogu **(2)** a relačnom kalkule **(2)**.

Datalog:

```
answer(P, A1, A2) ←
        lubi(P, A1),
        capuje(_, A1),
        capuje(_, A2),
        not niekde_nevypil_oba(P, A1, A2).

answer(P, A1, A2) ←
        lubi(P, A2),
        capuje(_, A1),
        capuje(_, A2),
        not niekde_nevypil_oba(P, A1, A2).

niekde_nevypil_oba(P, A1, A2) ←
        capuje(_, A1),
        capuje(_, A2),
        navstivil(_, P, K),
        not vypil_oba(P, K, A1, A2).

vypil_oba(P, K, A1, A2) ←
        navstivil(I1, P, K),
        vypil(I1, A1, _),
        navstivil(I2, P, K),
        vypil(I2, A2, _).
```

Relačný kalkul:

```
{[P, A1, A2]:
        ∃K1 ∃K2
        capuje(K1, A1) ∧ capuje(K2, A2) ∧ (lubi(P, A1) ∨ lubi(P, A2)) ∧
        ¬
        ( /* niekde_nevypil_oba(P, A1, A2) */
                ∃I ∃K
                navstivil(I, P, K) ∧
                ¬
                ( /* vypil_oba(P, K, A1, A2) */
                        ∃I1 ∃M1 ∃I2 ∃M2
                        navstivil(I1, P, K) ∧ vypil(I1, A1, M1) ∧
                        navstivil(I2, P, K) ∧ vypil(I2, A2, M2)
                )
        )
}
```

b) Nájdite dvojice [K, A, T], kde T je celkové vypité množstvo takého alkoholu A, ktorý sa vypil len v krčme K. Sformulujte dotaz v Datalogu **(2)** a SQL **(2)**.

Datalog:

```
answer(K, A, T) ←
        subtotal(vypite_exkluzivne(K, I, A, M), [K, A], [T = sum(M)]).

vypite_exkluzivne(K, I, A, M) ←
        navstivil(I, _, K),
        vypil(I, A, M),
        not vypite_inde(K, A).

vypite_inde(K, A) ←
        navstivil(_, _, K),
        navstivil(I2, _, K2),
        vypil(I2, A, _),
        not K = K2.
```

SQL:

```
with vypite_exkluzivne as
(
select n.Krcma, v.Alkohol, v.Mnozstvo
from navstivil n, vypil v
where n.Idn = v.Idn and not exists
        ( /* vypite_inde */
        select *
        from navstivil n2, vypil v2
        where n2.Idn = v2.Idn and v2.Alkohol = v.Alkohol and n2.Krcma <> n.Krcma
        )
)
select ve.Krcma as K, ve.Alkohol as A, sum(ve.Mnozstvo) as T
from vypite_exkluzivne ve
group by ve.Krcma, ve.Alkohol
```

2. a) Uveďte definície tretej (3NF) a Boyce-Coddovej (BCNF) normálnej formy. **(2)**

Relačná schéma [r, F] je v tretej normálnej forme, ak F^+ neobsahuje funkčnú závislosť $X \rightarrow A$, kde A je jednoduchý atribút, $A \notin X$, X nie je nadkľúč r a A nepatrí do žiadneho kľúča r.

Relačná schéma [r, F] je v Boyce-Coddovej normálnej forme, ak F^+ neobsahuje funkčnú závislosť $X \rightarrow A$, kde A je jednoduchý atribút, $A \notin X$ a X nie je nadkľúč r.

b) Rozhodnite, či platí „Ak relačná schéma s atribútmi $A_1, A_2, \ldots, A_N$ nie je v 3NF, tak $\exists i \exists j\ 1 \leq i \leq N \wedge 1 \leq j \leq N \wedge i \neq j \wedge \{A_1, A_2, \ldots, A_N\} - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$. Dokážte alebo uveďte konkrétny kontrapríklad. **(2)**

Áno, platí.

Dôkaz (priamo): Predpokladáme, že relačná schéma nie je v 3NF. Ukážeme, ako nájdeme atribúty A_i a A_j, ktoré spĺňajú dôsledok tej implikácie.

Ak relačná schéma nie je v 3NF, tak v nej platí aspoň jedna funkčná závislosť $X \rightarrow A_j$ taká, že A_j je jednoduchý atribút, $A_j \notin X$, X nie je nadkľúč a A_j nepatrí do žiadneho kľúča. Nech K je ľubovoľný kľúč. Keďže K je kľúč, platí $K \rightarrow X$ aj $K \rightarrow A_j$. Sú len 2 prípady: 1. X nie je podmnožinou K; 2. X je striktnou podmnožinou K (striktnou preto, lebo K je nadkľúč a X nie je).
1. Ak X nie je podmnožinou K, tak X obsahuje aspoň jeden atribút A_i taký, že $A_i \notin K$. Keďže kľúč K neobsahuje A_i ani A_j, triviálne platí $K - \{A_i\} - \{A_j\} \rightarrow A_j$. Tým pádom platí tiež $\{A_1, A_2, \ldots, A_N\} - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$.
2. Ak X je striktnou podmnožinou K, tak K obsahuje aspoň jeden atribút A_i taký, že $A_i \notin X$. Keďže $A_j \notin K$ a $A_i \notin X$, triviálne platí $K - \{A_i\} - \{A_j\} \rightarrow X$ a tranzitívne tiež $K - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$. Tým pádom platí tiež $\{A_1, A_2, \ldots, A_N\} - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$.
QED

c) Rozhodnite, či platí „Ak $\exists i \exists j\ 1 \leq i \leq N \wedge 1 \leq j \leq N \wedge i \neq j \wedge \{A_1, A_2, \ldots, A_N\} - \{A_i\} - \{A_j\} \rightarrow \{A_j\}$, tak relačná schéma s atribútmi $A_1, A_2, \ldots, A_N$ nie je v 3NF. Dokážte alebo uveďte konkrétny kontrapríklad. **(2)**

Nie, toto neplatí.

Kontrapríklad: relačná schéma s atribútmi A_1, A_2, A_3 a množinou funkčných závislostí $\{A_1 \rightarrow A_2, A_1 \rightarrow A_3\}$. Táto relačná schéma je v 3NF a zároveň platí, že $A_1 \rightarrow A_3$, pričom $\{A_1\} = \{A_1, A_2, A_3\} - \{A_2\} - \{A_3\}$.

d) Rozhodnite, či nasledujúci algoritmus vypočíta kľúč pre ľubovoľnú danú relačnú schému $[r(A_1, A_2, \ldots, A_N), F]$. Dokážte alebo uveďte konkrétny kontrapríklad. **(2)**

```
K = {};
for i = 1 to N
    if A_i ∉ K+      /* K+ je uzáver K vzhľadom na F */
        K = K ∪ {A_i};
```

Nie, toto neplatí.

Kontrapríklad: relačná schéma s atribútmi A_1, A_2 a množinou funkčných závislostí $\{A_2 \rightarrow A_1\}$. Jediným kľúčom je $\{A_2\}$, lenže tento algoritmus vypočíta $K = \{A_1, A_2\}$.

3. Uvažujte nasledujúci SQL dotaz nad extenzionálnou databázou e(X, Y):
with recursive p as (select e.X from e where not exists (
select * from p where p.X = e.Y)) select distinct p.X from p;
a) Vyjadrite výpočet dotazu v relačnej algebre. **(2)**

$\varnothing($
$\quad p = \pi_X (e \triangleright_{p.X = e.Y} p);$
$);$
$\Delta p;$

Ekvivalentne, bez použitia antijoinu:
$\varnothing($
$\quad p = \pi_X (e \bowtie (\pi_Y (e) - \pi_X (p)));$
$);$
$\Delta p;$

Výpočet začína s p = {}. Potom sa iteruje priradenie $p = \pi_X (e \triangleright_{p.X = e.Y} p)$, až kým sa relácia p prestane meniť. Výsledkom dotazu je p z poslednej iterácie, s následným odstránením duplikátov.

b) Vypočítajte výsledok dotazu pre e(X, Y) = {[1, 2], [1, 3], [2, 3]}. **(2)**

Iterácia 0:
p = {}

Iterácia 1:
p = {1, 2} /* Keď sme puntičkári, p je po tejto iterácii multimnožina {1, 1, 2}. */

Iterácia 2:
p = {1, 2}

Iterácia 3:
p = {1, 2}

Výsledok dotazu je {1, 2}.

c) Rozhodnite, či výpočet z úlohy a) skončí pre ľubovoľnú konečnú populáciu relácie e(X, Y). Odpoveď ÁNO resp. NIE zdôvodnite. **(2)**

Nie. Výpočet neskončí napríklad pre reláciu e(X, Y) = {[1, 1]}.

Iterácia 0:
p = {}

Iterácia 1:
p = {1}

Iterácia 2:
p = {}

Iterácia 3:
p = {1}

…
Obsah relácie p sa po každej iterácii operátora ∅ zmení (osciluje medzi {} a {1}).

4. Uvažujte SQL dotaz select r.X from r, s where r(Y) > s(Y). Relácia r(X, Y) je uložená v 10000 diskových blokoch, relácia s(X, Y) je uložená v 1000 diskových blokoch. Bloky na disku a v operačnej pamäti sú rovnako veľké. Dotaz sa počíta metódou nested-loop join, k dispozícii je 1111 voľných blokov operačnej pamäte.
a) Popíšte organizáciu operačnej pamäte počas výpočtu dotazu. Vysvetlite akým spôsobom sa rezervované bloky použijú pri výpočte výsledku dotazu. **(1)**

1 výstupný blok RAM sa rezervuje na postupné ukladanie a zápis n-tíc, ktoré sú výsledkom joinu.
1 vstupný blok RAM sa rezervuje na postupné čítanie blokov relácie r (lebo je väčšia než s)
1000 vstupných blokov RAM sa rezervuje na čítanie blokov relácie s

Nested-loop join použije 1002 blokov:
Do rezervovaných 1000 vstupných blokov RAM postupne načítaj všetky bloky relácie s.
for i = 1 to 10000
    Do rezervovaného 1 vstupného bloku RAM načítaj blok i relácie r.
    Pre všetky dvojice záznamov r a s práve uložených v RAM vyhodnoť joinovaciu podmienku (where).
    Keď je podmienka splnená (t.j. daná dvojica sa joinuje), aplikuj projekciu (select) na túto dvojicu
    a výslednú n-ticu pridaj do rezervovaného výstupného bloku RAM. Keď je výstupný blok plný, zapíš ho
    na výstup.
Ak po ukončení predošlého cyklu výstupný blok nie je prázdny, tak ho zapíš na výstup.

b) Keď každá operácia čítania resp. zápisu diskového bloku trvá 0.1 sekundy, môže výpočet výsledku daného dotazu trvať kratšie než 1 hodinu? Odpoveď ÁNO resp. NIE zdôvodnite. **(2)**

Počas výpočtu nested-loop join sa postupne prečíta každý blok relácie r (1000 operácií), každý práve raz. Aj každý blok relácie s sa prečíta práve raz (10000 operácií). Ak sa relácie r a s nejoinujú ani v jednom zázname, tak je výstup prázdny a horeuvedené operácie trvajú dokopy **1100 sekúnd, čo je zhruba 18 minút**.

Áno, tento výpočet môže trvať kratšie než 1 hodinu. (Ako dlho skutočne potrvá, závisí od obsahu relácií r a s a od ich reprezentácie.)

Ak relácie chápeme ako množiny (bez duplikátov), tak počet záznamov výslednej relácie je nanajvýš taký, aký je počet záznamov relácie r (možno nie každá n-tica r sa joinuje s niektorou n-ticou s). Navyše, výsledná relácia má na rozdiel od r iba 1 atribút, takže v 1 výstupnom bloku bude uložených nanajvýš toľko záznamov, koľko záznamov relácie r je možné uložiť do 1 bloku. Počet blokov výslednej relácie je teda nanajvýš rovný počtu blokov relácie r, t.j. 10000. Každá z nich sa na výstup zapíše práve raz. Počet diskových operácií sa tým zvýši najviac o 10000 a celkový čas vykonávania o najviac 1000 sekúnd. Lenže výsledkom má byť opäť množina n-tíc a eliminácia duplikátov (ktoré vznikajú projekciou) môže vyžadovať ďalšie diskové operácie.

V SQL sa relácie chápu ako multimožiny (s duplikátmi n-tíc). Pri naivnej reprezentácii n-tíc bez kompresie (duplikáty explicitne vymenujeme) bude veľkosť výsledku dotazu 0 blokov až 10000000 blokov v závislosti od obsahu relácií r a s. V najhoršom prípade potrvá zápis blokov výsledku 1000000 sekúnd, čo je zhruba 280 hodín. Avšak môže trvať (pri inom naplnení relácií r a s) aj 0 sekúnd, nezávisle od reprezentácie relácií. Prirodzená reprezentácia multimnožiny je podobná reprezentácii množiny: každá n-tica relácie je uložená práve raz, spolu so svojou násobnosťou. To „práve raz“ však komplikuje (oproti naivnej reprezentácii) implementáciu relačných operátorov a pridáva diskové operácie. Kompromisom je „zmiešaná“ reprezentácia, v ktorej síce dovolíme opakované uloženie n-tice v relácii; ale v rámci jedného bloku n-ticu neopakujeme, len spolu s ňou ukladáme aj jej násobnosť. Toto môže zjednodušiť implementáciu operátorov a niekedy výrazne skomprimovať uloženie n-tíc v rámci jedného bloku.
Pri prirodzenej reprezentácii multimnožiny s ukladaním násobnosti n-tíc má zmysel využiť 110 voľných blokov operačnej pamäte ako výstupných (na oneskorenie zápisov). Obsah týchto 110 blokov sa udržiava utriedený podľa výstupných n-tíc (v tomto prípade podľa atribútu X) a so zápisom sa čaká tak dlho ako sa dá: keď pribúda duplikát, príslušná násobnosť sa inkrementuje; inak sa obsah 110 blokov reorganizuje tak, aby aj po vložení novej n-tice zostal utriedený. Keď sa nová n-tica už nezmestí do poľa 110 blokov, tak sa všetkých 110 blokov zapíše na výstup ako jeden utriedený beh. Po dočítaní relácií r a s sa utriedené behy zlúčia do jedného výstupného behu podobne ako v 2. fáze merge-sort. Ak je behov najviac 1110, tak sa zlúčia v jednom prechode a celé generovanie výstupu (bez duplikovaných n-tíc, s násobnosťami) stojí najviac 10000 vstupných a 10000 výstupných operácií, t.j. dokopy 2000 sekúnd, t.j. ca. 34 minút.