5/1/2017 Úvod do databáz, skúškový test, max **60** bodov

1. Uvažujte databázu bez duplikátov a null hodnôt: capuje(Krcma, Alkohol, Cena), lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Idn → Pijan, Krcma; Idn, Alkohol → Mnozstvo; Krcma, Alkohol → Cena; Mnozstvo > 0.
a) Sformulujte bezpečný dotaz v Datalogu **(6)** a relačnom kalkule **(6)** na dvojice [P, K] také, že pijan P vypil v krčme K pivo aj rum; pri každej návšteve K vypil buď pivo alebo rum (prípadne ďalšie alkoholy); ale nikdy nevypil pivo aj rum počas jednej návštevy K.

Relačný kalkul:

```
{[P, K]:
        ∃I1 ∃M1 ∃I2 ∃M2
        navstivil(I1, P, K) ∧ vypil(I1, pivo, M1) ∧ navstivil(I2, P, K) ∧ vypil(I2, rum, M2) ∧
        ¬ /* niekedy_nevypil(P, K) */
        (
                ∃I navstivil(I, P, K) ∧
                ¬
                (
                        ∃M vypil(I, pivo, M)
                ) ∧
                ¬
                (
                        ∃M vypil(I, rum, M)
                ) ∧

        ) ∧
        ¬ /* vypil_oba(P, K) */
        (
                ∃I ∃M1 ∃M2 navstivil(I, P, K) ∧ vypil(I, pivo, M1) ∧ vypil(I, rum, M2)
        )
}
```

Datalog:

```
answer(P, K) ←
        navstivil(I1, P, K),
        vypil(I1, pivo, _),
        navstivil(I2, P, K),
        vypil(I2, rum, _),
        not niekedy_nevypil(P, K),
        not vypil_oba(P, K).

niekedy_nevypil(P, K) ←
        navstivil(I, P, K),
        not v(I, pivo),
        not v(I, rum).

v(I, A) ←
        vypil(I, A, _).

vypil_oba(P, K) ←
        navstivil(I, P, K),
        vypil(I, pivo, _),
        vypil(I, rum, _).
```

b) Sformulujte dotaz v Datalogu **(6)** a SQL **(6)** na dvojice [P, SL], kde SL je celková suma, ktorú P utratil za pitie alkoholov, ktoré ľúbi. Vo výsledku majú byť všetci pijani (pijan je ten, kto ľúbi nejaký alkohol alebo navštívil nejakú krčmu), t.j. aj dvojice s SL=0.

Datalog:

```
answer(P, SL) ←
        subtotal(vypil_lubi(P, _, _, U), [P], [SL = sum(U)]).

answer(P, 0) ←
        pijan(P),
        not vl(P).

vypil_lubi(P, I, A, U) ←
        lubi(P, A),
        navstivil(I, P, K),
        vypil(I, A, M),
        capuje(K, A, C),
        U is M * C. /* U = M * C */

vl(P) ←
        vypil_lubi(P, _, _, _).

pijan(P) ←
        lubi(P, _).

pijan(P) ←
        navstivil(_, P, _).
```

SQL:

```
(
select l.Pijan as P, sum(v.Mnozstvo * c.Cena) as SL
from lubi l, navstivil n, vypil v, capuje c
where l.Pijan = n.Pijan and l.Alkohol = v.Alkohol and l.Alkohol = c.Alkohol and n.Idn = v.Idn and
n.Krcma = c.Krcma
group by l.Pijan
)
union
(
select l.Pijan as P, 0 as SL
from lubi l
where not exists (
        select *
        from lubi l2, navstivil n, vypil v
        where l.Pijan = l2.Pijan and l2.Pijan = n.Pijan and l2.Alkohol = v.Alkohol and n.Idn = v.Idn)
)
union
(
select l.Pijan as P, 0 as SL
from navstivil n
where not exists (
        select *
        from lubi l, navstivil n n2, vypil v
        where n.Pijan = n2.Pijan and l.Pijan = n2.Pijan and l.Alkohol = v.Alkohol and n2.Idn = v.Idn)
)
```

2. Uvažujte relačnú schému r(A, B, C, D, E) s funkčnými závislosťami
A→CE, ACD→BE, AD→B, BC→D, BE→AC.
a) Rozhodnite či dekompozícia r do r1(A, B, E), r2(A, C, E), r3(A, D, E) je bezstratová (t.j. spája sa bezstratovo). Ak nie, uveďte konkrétny príklad naplnenia relácie r, ktoré to demonštruje.
Vysvetlite. **(6)**

Podľa definície dekompozícia r do r1, r2, r3 je bezstratová, ak pre každú populáciu r platí
$r = \Delta(\Pi_{r1}(r) \bowtie \Pi_{r2}(r) \bowtie \Pi_{r3}(r))$.

Na testovanie bezstratovosti použijeme algoritmus chase:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| ABE | a1 | a2 | b13 | b14 | a5 |
| ACE | a1 | b22 | a3 | b24 | a5 |
| ADE | a1 | b32 | b33 | a4 | a5 |

A→CE

| | A | B | C | D | E |
|---|---|---|---|---|---|
| ABE | a1 | a2 | a3 | b14 | a5 |
| ACE | a1 | b22 | a3 | b24 | a5 |
| ADE | a1 | b32 | a3 | a4 | a5 |

Tu chase končí. Žiaden riadok neobsahuje len symboly a*. **Teda daná dekompozícia je stratová (t.j. nespája sa bezstratovo).**

Stratovosť danej dekompozície demonštruje naplnenie relácie r z poslednej iterácie chase:

**r(A, B, C, D, E)**

| **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|
| **a1** | **a2** | **a3** | **b14** | **a5** |
| **a1** | **b22** | **a3** | **b24** | **a5** |
| **a1** | **b32** | **a3** | **a4** | **a5** |

r1(A, B, E)

| A | B | E |
|---|---|---|
| a1 | a2 | a5 |
| a1 | b22 | a5 |
| a1 | b32 | a5 |

r2(A, C, E)

| A | C | E |
|---|---|---|
| a1 | a3 | a5 |

r3(A, D, E):

| A | D | E |
|---|---|---|
| a1 | b14 | a5 |
| a1 | b24 | a5 |
| a1 | a4 | a5 |

$[a1, a2, a3, a4, a5] \in \Delta(\Pi_{r1}(r) \bowtie \Pi_{r2}(r) \bowtie \Pi_{r3}(r))$, ale $[a1, a2, a3, a4, a5] \notin r$.
Teda $r \neq \Delta(\Pi_{r1}(r) \bowtie \Pi_{r2}(r) \bowtie \Pi_{r3}(r))$ pre dané naplnenie relácie r.

b) Nájdite bezstratovú dekompozíciu r do dvoch relácií r1 a r2, ktorá je v tretej normálnej forme. Zdôvodnite prečo má uvedená dekompozícia požadované vlastnosti. **(6)**

Definícia. Relačná schéma [r, F] je v tretej normálnej forme, ak pre každú platnú netriviálnu funkčnú závislosť **X** → Y (kde X je množina atribútov a Y je jeden atribút) platí:
**X** je nadkľúč v r; alebo Y patrí do niektorého kľúča r.

Nájdime všetky kľúče r:
ABCDE
- -A: BCDE
  - -B: CDE
  - +B: BCDE
    - -C: BDE
      - -D: **BE**
      - +D: BD
    - +C: BCD
- +A: ABD
  - -B: AD
    - -D: A
    - +D: **AD**
  - +B: **AB**

AB, AD a BE sú kľúče r, iné kľúče nie sú. Jediný atribút, ktorý sa nenachádza v žiadnom kľúči, je C. To znamená, že r1(A, B, D, E) je v 3NF (lebo každý z týchto atribútov patrí do niektorého kľúča r). Potrebujeme vyrobiť r2 tak, aby bola v 3NF a aby r1 a r2 bola bezstratová dekompozícia r. Napríklad r2(A, C) sa spája bezstratovo s r1, lebo r1 ∩ r2 = {A} je nadkľúč v r2. r2 je v 3NF, lebo v nej platí jediná netriviálna funkčná závislosť A→C, pričom {A} je nadkľúč v r2.

**Dekompozícia r1(A, B, D, E), r2(A, C) je bezstratová a r1 aj r2 sú v 3NF.**

3. Uvažujte SQL dotaz
with recursive q as
(select r1.X, r2.Y from r r1, r r2 where not exists
  (select * from r r3 where r3.X = r1.X and r3.Y = r2.Y))
union
(select r1.X, r1.Y from r r1, q q1 where r1.X = q1.Y and r1.Y = q1.X)
select r1.X, r1.Y
from r r1 where not exists (select * from q q1 where r1.X = q1.X and r1.Y = q1.Y);
a) Zapíšte výpočet daného dotazu 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)**

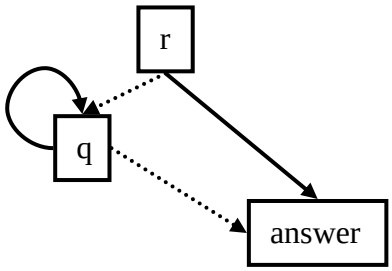
**q := {};**
**Φ(**
**q := ($\Pi_{r1.X, r2.Y}$ ($P_{r1}$(r) × $P_{r2}$(r)) - r) ∪ $\Pi_{r.X, r.Y}$ (r ⋈$_{r.X = q.Y \wedge r.Y = q.X}$ q);**
**);**
**answer := r - q;**

Keďže r sa v priradení q := … nemení, výsledkom výrazu $\Pi_{r1.X, r2.Y}$ ($P_{r1}$(r) × $P_{r2}$(r)) - r je rovnaká relácia vo všetkých iteráciách Φ. Tento výraz stačí teda vypočítať raz. Optimalizovaný plán výpočtu:
q1 := ($\Pi_{r1.X, r2.Y}$ ($P_{r1}$(r) × $P_{r2}$(r)) - r);
q := {};
Φ(
q := q1 ∪ $\Pi_{r.X, r.Y}$ (r ⋈$_{r.X = q.Y \wedge r.Y = q.X}$ q);
);
answer := r - q;

q1(X, Y) = {[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]}
q(X, Y) = {}
Po 1.iterácii Φ:
q(X, Y) = {[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 2.iterácii Φ:
q(X, Y) = {[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]}
Vypočítal sa pevný bod Φ, iterácie končia.
answer(X, Y) = **{[0, 0], [1, 1], [2, 2], [3, 0], [4, 5]}**
Toto je výsledok daného dotazu.

b) Rozhodnite či výpočet daného dotazu semi-naivnou evaluáciou skončí pre ľubovoľné naplnenie relácie r. Zdôvodnite. **(6)**

Graf závislostí relácií (prerušovaná hrana znamená závislosť s použitím negácie) pre daný dotaz:




Graf závislostí relácií neobsahuje cyklus s negovanou (prerušovanou) hranou, takže relácie sa dajú stratifikovať: stratum(r) = 0, stratum(q) = 1, stratum(answer) = 2. Tým pádom **výpočet semi-naivnou evaluáciou skončí pre ľubovoľné naplnenie relácie r.**

4. a) Vysvetlite metódu časových pečiatok pre izoláciu transakcií (uveďte kedy a aké akcie robí scheduler, akcie zapíšte v pseudokóde). **(6)**

Každá transakcia T dostane pri štarte časovú pečiatku TS(T).
Každý objekt X má dve časové pečiatky, TSR(X) a TSW(X). Obe sú inicializované na $-\infty$.

Pred vykonaním $r_T(X)$ vykoná scheduler toto:

```
if (TS(T) < TSW(X))
        abort T;
else
        TSR(X) := TS(T);
```

Pred vykonaním $w_T(X)$ vykoná scheduler toto:

```
if ((TS(T) < TSR(X)) ∨ (TS(T) < TSW(X)))
        abort T;
else
        TSW(X) := TS(T);
```

b) Uveďte príklad sériovateľného a obnoviteľného rozvrhu, ktorý sa nedá generovať schedulerom, ktorý používa metódu časových pečiatok. Vysvetlite. **(6)**

**s1, s2, r2(X), w1(X), c1, c2**

Tento rozvrh je konflikt-sériovateľný (lebo obsahuje len jednu dvojicu konfliktných operácií) a obnoviteľný (lebo neobsahuje dirty read).

Scheduler urobí pre tento vstupný rozvrh nasledujúce akcie:

| | |
|---|---|
| s1 | pridelí časovú pečiatku TS(T1) |
| s2 | pridelí časovú pečiatku TS(T2) |
| r2(X) | TSR(X) := TS(T2) |
| w1(X) | abort(T1), lebo TS(T1) < TSR(X) |