22/1/2020 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 $\rightarrow$ Pijan, Krcma; Idn, Alkohol $\rightarrow$ Mnozstvo; Mnozstvo > 0.
a) Sformulujte bezpečný dotaz v Datalogu **(6)** a relačnej algebre **(6)** na pijanov, ktorí neľúbia žiaden alkohol ktorý ľúbi Móric, ale pri každej svojej návšteve krčmy vypili niektorý z Móricových obľúbených alkoholov (nie nutne stále ten istý).

Datalog:

```
answer(P) ←
        pijan(P),
        not lubi_moricov(P),
        not niekedy_nevypil_nic_moricove(P).

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

lubi_moricov(P) ←
        lubi(P, A),
        lubi(moric, A).

niekedy_nevypil_nic_moricove(P) ←
        navstivil(I, P, _),
        not vypil_nieco_moricove(I).

vypil_nieco_moricove(I) ←
        vypil(I, A, _),
        lubi(moric, A).
```

pijan = π_{Pijan} (navstivil) $\cup$ π_{Pijan} (vypil)

lubi_moricov = $\pi_{\text{l1.Pijan}}$ (P_{l1} (lubi) $\bowtie_{\text{l1.Alkohol = l2.Alkohol and l2.Pijan = 'moric'}}$ P_{l2} (lubi))

vypil_nieco_moricove = π_{Idn} (vypil $\bowtie$ $\sigma_{\text{Pijan= 'moric'}}$ lubi)

niekedy_nevypil_nic_moricove = π_{Pijan} (navstivil $\triangleright$ vypil_nieco_moricove)

/* answer */

(pijan – lubi_moricov) – niekedy_nevypil_nic_moricove

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na dvojice [K, S], ktoré hovoria koľko peňazí celkovo prepili v krčme K pijani, ktorí navštívili K aspoň stokrát. Výsledok nemá obsahovať dvojice s S=0.

Datalog:

```
answer(K, S) ←
        subtotal(data(_, _, K, U), [K], [S = sum(U)]).

data(I, A, K, U) ←
        navstivil(I, P, K),
        subtotal(navstivil(I, P, K), [P, K], [N = count(I)]),
        N >= 100,
        vypil(I, A, M),
        capuje(K, A, C),
        U = M * C.
```

SQL:

```
with
pocet_navstev as
(
        select n.Pijan, n.Krcma, N = count(n.Idn)
        from navstivil n
        group by n.Pijan, n.Krcma
)
select n.Krcma, sum(v.Mnozstvo * c.Cena) as S
from navstivil n, pocet_navstev p, vypil v, capuje c
where n.Idn = v.Idn and n.Pijan = p.Pijan and n.Krcma = p.Krcma and n.Krcma = c.Krcma and p.N >=100 and
v.Alkohol = c.Alkohol
group by n.Krcma
```

2. Uvažujte reláciu r(A, B, C, D, E, F, G, H) s funkčnými závislosťami
A→E, AD→B, AC→BD, E→B, BG→F, BE→D, BH→E, BCD→GH, BCDF→A.
a) Dekomponujte r do tretej normálnej formy, bezstratovo a so zachovaním všetkých funkčných závislostí. **(6)**

Nájdime nejaké minimálne pokrytie množiny funkčných závislostí.
Po kánonizácii:
A→E, AD→B, AC→B, AC→D, E→B, BG→F, BE→D, BH→E, BCD→G, BCD→H, BCDF→A.
Po odstránení redundantných atribútov z ľavých strán:
A→E, A→B, A→B, A→D, E→B, BG→F, E→D, BH→E, BCD→G, BCD→H, BCD→A.
Po eliminácii redundantných funkčných závislostí:
A→E, A→B, A→D, E→B, BG→F, E→D, BH→E, BCD→G, BCD→H, BCD→A.
Toto je minimálne pokrytie.

Požadovanú 3NF dekompozíciu urobíme podľa tohto minimálneho pokrytia:
(A, B, D, E), **/* A→E, A→B, A→D, E→B, E→D */**
(B, F, G), **/* BG→F */**
(B, E, H), **/* BH→E */**
(A, B, C, D, G, H) **/* BCD→G, BCD→H, BCD→A */**
(Atribúty v poslednej relácii sú nadkľúčom v r.)

b) Rozhodnite či dekompozícia danej relačnej schémy do
(A, B, C, D, E, H), (A, C, F, G)
je v tretej normálnej forme. Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Nájdime všetky kľúče r:
ABCDEFGH
-A: BCDEFGH
-B: CDEFGH
-D: CEFGH
-E: CFGH
+E: **CE**
+D: CDFGH
+B: BCDFGH
-D: BCFGH
-F: BCGH
-G: **BCH**
+G: BCG
+F: BCFG
+D: **BCD**
+A: **AC**

Kľúče v r sú AC, BCD, BCH, CE. Iné kľúče nie sú.

V relácii (A, B, C, D, E, H) každý atribút patrí do niektorého kľúča, teda táto relácia je v 3NF.
V relácii (A, C, F, G) atribúty A a C patria do kľúča AC a neplatí v nej žiadna netriviálna funkčná závislosť s atribútom F alebo G na pravej strane. Aj táto relácia je teda v 3NF.
Áno, daná dekompozícia je v tretej normálnej forme.

c) Definujte pojem zachovania funkčných závislostí pri dekompozícii relačnej schémy. Na dekompozícii z úlohy b) vysvetlite negatívne dôsledky nezachovania funkčných závislostí pri použití tejto konkrétnej dekompozície v databázovej aplikácii. **(6)**

Definícia. Dekompozícia $(r_1, F_1), \ldots, (r_n, F_n)$ zachováva funkčné závislosti schémy (r, F), ak každá platná funkčná závislosť $\mathbf{X} \rightarrow \mathbf{Y}$ z F je v uzávere $(\cup F_i)^+$.

Dekompozícia r1(A, B, C, D, E, H), r2(A, C, F, G) nezachováva funkčné závislosti, lebo platná funkčná závislosť BG→F neplatí lokálne v r1 ani v r2 (a nie je v uzávere funkčných závislostí zachovaných v r1 a r2, lebo žiadna z nich neobsahuje oba atribúty B a G). Podobne BCD→G nie je zachovaná v tejto dekompozícii.

Každá platná funkčná závislosť v r je v uzávere minimálneho pokrytia, ktoré sme našli v úlohe a). Keď vytvárame r v SQL systéme, každú funkčnú závislosť z minimálneho pokrytia vieme použiť na definíciu integritného obmedzenia v r (v klauze *check* príkazu *create table*). Pri vytváraní relácií r1 a r2 sa integritné obmedzenia BG→F a BCD→G nedajú vyjadriť lokálne v r1 a v r2. Ak použitý systém umožňuje definíciu globálnych integritných obmedzení (pre celú databázu, napr. v PostgreSQL *create assertion*), vieme síce BG→F a BCD→G vyjadriť, ale pre reláciu r1 ⋈ r2. **Toto vedie k drahšej (menej efektívnej) kontrole integrity pri vkladaní a modifikácii dát v r1 a r2** (pre zachované funkčné závislosti postačuje lokálna kontrola v r1 resp. v r2).

3. Uvažujte nasledujúci SQL dotaz nad reláciami r(A, B) a s(C) bez duplikátov a NULL hodnôt:
select s1.C from s s1, s s2 where s1.C > s2.C and not exists
(select * from r r1, r r2 where r1.B = s1.C or (r2.A = s1.C and r1.B = s2.C))
a) Zapíšte daný dotaz ekvivalentne v Datalogu (ako bezpečný Datalogový program s dotazom). **(6)**

?- answer(C).

answer(C1) ←
s(C1),
s(C2),
C1 > C2
not sub(C1),
not sub(C1, C2).

sub(C1) ←
r(_, C1).

sub(C1, C2) ←
r(C1, _),
r(_, C2).

~~sub(C1) ←~~
~~r(A1, C1),~~
~~r(A2, B2).~~

~~sub(C1, C2) ←~~
~~r(A1, C2),~~
~~r(C1, B2).~~

b) Zapíšte výpočet daného dotazu v relačnej algebre. **(6)**

$\pi_{s1.C}\ (((P_{s1}\ (s) \bowtie_{s1.C\ >\ s2.C} P_{s2}\ (s)) \rhd_{s1.C\ =\ B}\ \pi_B\ (r)) \rhd_{s1.C\ =\ B\ \wedge\ s2.C\ =\ A}\ (\pi_B\ (r) \times \pi_A\ (r)))$

c) Vypočítajte výsledok daného dotazu pre
s(C) = {0, 1, 2, 3, 4, 5}, r(A, B) = {[0, 1], [2, 1], [2, 4], [3, 4]}. **(6)**

V záujme jednoduchosti budeme počítať bez duplikátov:
$P_{s1}\ (s) \bowtie_{s1.C\ >\ s2.C} P_{s2}\ (s) =$
{[1, 0], [2, 0], [2, 1], [3, 0], [3, 1], [3, 2], [4, 0], [4, 1], [4, 2], [4, 3], [5, 0], [5, 1], [5, 2], [5, 3], [5, 4]}

$\pi_B\ (r) = \{1, 4\}$

$(P_{s1}\ (s) \bowtie_{s1.C\ >\ s2.C} P_{s2}\ (s)) \rhd_{s1.C\ =\ B}\ \pi_B\ (r) =$
{[2, 0], [2, 1], [3, 0], [3, 1], [3, 2], [5, 0], [5, 1], [5, 2], [5, 3], [5, 4]}

$\pi_B\ (r) \times \pi_A\ (r) =$
{[1, 0], [1, 2], [1, 3], [4, 0], [4, 2], [4, 3]}

$((P_{s1}\ (s) \bowtie_{s1.C\ >\ s2.C} P_{s2}\ (s)) \rhd_{s1.C\ =\ B}\ \pi_B\ (r)) \rhd_{s1.C\ =\ B\ \wedge\ s2.C\ =\ A}\ (\pi_B\ (r) \times \pi_A\ (r)) =$
{[2, 0], [2, 1], [3, 0], [3, 1], [3, 2], [5, 0], [5, 1], [5, 2], [5, 3], [5, 4]}

$\pi_{s1.C}\ (((P_{s1}\ (s) \bowtie_{s1.C\ >\ s2.C} P_{s2}\ (s)) \rhd_{s1.C\ =\ B}\ \pi_B\ (r)) \rhd_{s1.C\ =\ B\ \wedge\ s2.C\ =\ A}\ (\pi_B\ (r) \times \pi_A\ (r))) = \{2, 3, 5\}$
Výsledkom dotazu je {2, 3, 5}.