13/2/2019 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 $\rightarrow$ Pijan, Krcma; Idn, Alkohol $\rightarrow$ Mnozstvo; Mnozstvo $>$ 0.
Hodnoty Idn sú časové pečiatky návštev, t.j. rastú spolu s časom (presnejšie, s každou návštevou krčmy).
a) Nájdite dvojice [P, K] také, že pijan P vypil (nie nutne v krčme K) všetky alkoholy čapované v krčme K; a aspoň raz navštívil krčmu K dvakrát za sebou. Sformulujte tento dotaz v relačnom kalkule **(6)** a Datalogu **(6)**.

Relačný kalkul:

```
{[P, K]:
        ∃I1 ∃I2
        navstivil(I1, P, K) ∧ navstivil(I2, P, K) ∧ I1 < I2 ∧
        ¬
        (
                ∃I3 ∃K3
                navstivil(I3, P, K3) ∧ I1 < I3 ∧ I3 < I2
        )
        ∧
        ¬
        (
                ∃A
                capuje(K, A) ∧
                ¬
                (
                        ∃I ∃M ∃K2
                        navstivil(I, P, K2) ∧ vypil(I, A, M)
                )
        )
}
```

Datalog:

```
answer(P, K) ←
        navstivil(I1, P, K),
        navstivil(I2, P, K),
        I1 < I2,
        not navsteva_medzitym(I1, I2),
        not capuje_nevypil(P, K).

navsteva_medzitym(I1, I2) ←
        navstivil(I1, P, K),
        navstivil(I2, P, K),
        navstivil(I3, P, K3),
        I1 < I3,
        I3 < I2.

capuje_nevypil(P, K) ←
        capuje(K, A),
        navstivil(_, P, K),
        not niekedy_vypil(P, A).

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

b) Sformulujte v Datalogu **(6)** a SQL **(6)** dotaz na trojice [P, K, A] také, že pijan P ľúbi alkohol A, navštívil krčmu K, a alkohol A v krčme K pije s neklesajúcou tendenciou. Neklesajúca tendencia znamená, že pijan P pri každej návšteve krčmy K vypil alkohol A v aspoň takom množstve ako pri svojej predošlej návšteve krčmy K (z čoho napríklad vyplýva, že ak P pri niektorej návšteve K vypil A, tak potom vypil A aj pri každej svojej neskoršej návšteve K).

Datalog:

```
answer(P, K, A) ←
        lubi(P, A),
        navstivil(_, P, K),
        not klesol(P, K, A).

klesol(P, K, A) ←
        navstivil(I1, P, K),
        vypil(I1, A, M1),
        navstivil(I2, P, K),
        vypil(I2, A, M2),
        I1 < I2,
        M2 < M1.

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

v2(I, A) ←
        vypil(I, A, _).
```

SQL:

```
with klesol as (
        (
                select n1.Pijan, n1.Krcma, v1.Alkohol
                from navstivil n1, vypil v1, navstivil n2, vypil v2
                where n1.Idn = v1.Idn and n1.Pijan = n2.Pijan and n1.Krcma = n2.Krcma and
                v1.Alkohol = v2.Alkohol and n1.Idn < n2.Idn and v2.Mnozstvo < v1.Mnozstvo
        )
        union
        (
                select n1.Pijan, n1.Krcma, v1.Alkohol
                from navstivil n1, vypil v1, navstivil n2
                where n1.Idn = n2.Idn and n1.Pijan = n2.Pijan and n1.Krcma = n2.Krcma and n1.Idn < n2.Idn
                and not exists (
                        select *
                        from vypil v2
                        where v2.Idn = n2.Idn and v2.Alkohol = v1.Alkohol
                )
        )
),
select n.Pijan, n.Krcma, l.Alkohol
from lubi l, navstivil n
where n.Idn = v.Idn and not exists (
        select *
        from klesol k
        where k.Pijan = n.Pijan and k.Krcma = n.Krcma and k.Alkohol = v.Alkohol)
```

2. Uvažujte SQL dotaz nad reláciou r(X, Y), ktorá neobsahuje duplikáty ani NULL hodnoty:
select r1.Y as X, r2.Y as Y, sum(distinct r2.X) as S from r r1, r r2
where r1.X < r2.Y group by r1.Y, r2.Y having count(r1.X) > r1.Y
a) Vyjadrite daný dotaz v Datalogu. **(6)**

```
?- answer(R1Y, R2Y, S).

answer(R1Y, R2Y, S) ←
        subtotal(sjoin(R1Y, R2Y, R2X), [R1Y, R2Y], [S = sum(R2X)]),
        subtotal(cjoin(R1Y, R2Y, R1X, _), [R1Y, R2Y], [C = count(R1X)]),
        C > R1Y.

cjoin(R1Y, R2Y, R1X, R2X) ←
        r(R1X, R1Y),
        r(R2X, R2Y),
        R1X < R2Y.

sjoin(R1Y, R2Y, R2X) ←
        cjoin(R1Y, R2Y, _, R2X).
```

b) Vyjadrite daný dotaz v relačnej algebre. **(6)**

$\boldsymbol{\pi_{X \leftarrow r1.Y,\, Y \leftarrow r2.Y,\, S} (\sigma_{C > r1.Y} (\Gamma_{r1.Y,\, r2.Y,\, C = count(r1.X),\, S = sum(distinct\ r2.X)} (P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r))))}$

Alebo (bez použitia distinct v agregácii, s použitím operátora δ pre elimináciu duplikátov v relácii):

$cjoin = P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r);$

$sjoin = \delta(\pi_{r1.Y,\, r2.Y,\, r2.X} (cjoin));$

$\pi_{X \leftarrow r1.Y,\, Y \leftarrow r2.Y,\, S} ((\sigma_{C > r1.Y} (\Gamma_{r1.Y,\, r2.Y,\, C = count(r1.X)} (cjoin)) \bowtie \Gamma_{r1.Y,\, r2.Y,\, S = sum(r2.X)} (sjoin));$ */* answer */*

c) Vypočítajte výsledok dotazu pre reláciu
r(X, Y) = {[1, 1], [1, 4], [2, 1], [2, 3], [2, 4], [3, 1], [4, 2], [4, 3], [4, 4]}. **(6)**

$P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r)$ = .(r1.Y, r2.Y, r1.X, r2.X) ={
[1,2,1,4],
[1,3,1,2], [1,3,1,4], [1,3,2,2], [1,3,2,4],
[1,4,1,1], [1,4,1,2], [1,4,1,4], [1,4,2,1], [1,4,2,2], [1,4,2,4], [1,4,3,1], [1,4,3,2], [1,4,3,4],
[3,3,2,2], [3,3,2,4],
[3,4,2,1], [3,4,2,2], [3,4,2,4],
[4,2,1,4],
[4,3,1,2], [4,3,1,4], [4,3,2,2], [4,3,2,4],
[4,4,1,1], [4,4,1,2], [4,4,1,4], [4,4,2,1], [4,4,2,2], [4,4,2,4]
}

Každý riadok z predošlého medzivýsledku zodpovedá jednej grupe podľa r1.Y, r2.Y.
Po grupovaní a agregácii:
$\Gamma_{r1.Y,\, r2.Y,\, C = count(r1.X),\, S = sum(distinct\ r2.X)} (P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r))$ = .(r1.Y, r2.Y, C, S) =
{[1,2,1,4], [1,3,4,6], [1,4,9,7], [3,3,2,6], [3,4,3,7], [4,2,1,6], [4,3,4,6], [4,4,6,7]}
Po selekcii (having):
$\sigma_{C > r1.Y} (\Gamma_{r1.Y,\, r2.Y,\, C = count(r1.X),\, S = sum(distinct\ r2.X)} (P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r)))$ = .(r1.Y, r2.Y, C, S) =
{[1,3,4,6], [1,4,9,7], [4,4,6,7]}
Po finálnej projekcii:
$\pi_{X \leftarrow r1.Y,\, Y \leftarrow r2.Y,\, S} (\sigma_{C > r1.Y} (\Gamma_{r1.Y,\, r2.Y,\, C = count(r1.X),\, S = sum(distinct\ r2.X)} (P_{r1} (r) \bowtie_{r1.X < r2.Y} P_{r2} (r))))$ = .(X, Y, S) =
{[1,3,6], [1,4,7], [4,4,7]}
Táto množina je výsledkom dotazu pre danú populáciu r.

3. Uvažujte reláciu r(A, B, C, D, E, F), v ktorej platia funkčné závislosti
AB→CDF, ACF→B, AE→B, CD→ABF, CF→E, CEF→ABD, AD → {}, {}→B
({} označuje prázdnu množinu).
a) Vyjadrite posledné dve funkčné závislosti v relačnom kalkule (reláciu r interpretujte ako predikát). **(6)**

Funkčná závislosť AD→{} vyjadruje integritné obmedzenie
∀A ∀D ∀B1 ∀C1 ∀E1 ∀F1 ∀B2 ∀C2 ∀E2 ∀F2
((r(A, B1, C1, D, E1, F1) ∧ r(A, B2, C2, D, E2, F2)) ⇒ true),
t.j. **true** (táto funkčná závislosť v skutočnosti neznamená žiadne obmedzenie).

Funkčná závislosť {}→B vyjadruje integritné obmedzenie
∀A1 ∀B1 ∀C1 ∀D1 ∀E1 ∀F1 ∀A2 ∀B2 ∀C2 ∀D2 ∀E2 ∀F2
((r(A1, B1, C1, D1, E1, F1) ∧ r(A2, B2, C2, D2, E2, F2)) ⇒ B1 = B2),
t.j. **hodnota atribútu B v relácii r je konštantná**.

b) Nájdite všetky kľúče relácie r. **(6)**

B nie je v žiadnom kľúči, lebo {}→B. {} nie je kľúč.

ACDEF
- -A: CDEF
  - -C: DEF
  - +C: CDEF
    - -D: CEF
      - -E: CF
        - -F: C
        - +F: **CF**
      - +E: CE
    - +D: **CD**
- +A: **A**

Kľúče v r sú A, CD, CF. Iné kľúče v r nie sú.

c) Dekomponujte reláciu r do Boyce-Coddovej normálnej formy, bezstratovo. Snažte sa vyhnúť zbytočnému rozbitiu funkčných závislostí. **(6)**

Nájdime nejaké minimálne pokrytie množiny funkčných závislostí platných v r. Po minimalizácii ľavých strán kánonických funkčných závislostí:
A→C, A→D, A→F, {}→B, {}→B, CD→A, {}→B, CD→F, CF→E, CF→A, {}→B, CF→D, {}→{}, {}→B.
Po vynechaní redundantných funkčných závislostí:
{}→B, A→C, A→D, A→F, CD→A, CF→D, CF→E
Toto je minimálne pokrytie.

r(A, B, C, D, E, F) nie je v BCNF kvôli {}→B (prázdna množina nie je nadkľúč v r). Keďže B je konštantný atribút, patrí do samostatnej relácie. Uvažujme dekompozíciu r1(A, C, D, E, F), r2(B). Relácia r2 je v BCNF (prázdna množina je v nej kľúčom). Aj relácia r1 je v BCNF, lebo funkčné závislosti tohto minimálneho pokrytia okrem {}→B (ktorá je pre r1 irelevantná) sa atribútu B netýkajú; a na ľavej strane každej funkčnej závislosti je niektorý z kľúčov r1. (Teda všetky netriviálne funkčné závislosti platné v r1 majú na ľavej strane nadkľúč.)

r1(A, C, D, E, F), r2(B) je BCNF dekompozícia, ktorá sa spája bezstratovo a zachováva všetky funkčné závislosti danej relačnej schémy.