17/1/2025 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)**, SQL **(6)** a relačnej algebre **(6)** na dvojice [P, K] také, že pijan P navštívil krčmu K; a P niekde vypil každý alkohol, ktorý niekto vypil v krčme K.

Datalog:

```
answer(P, K) ←
        navstivil(_, P, K),
        not nevypil_niektory(P, K).

nevypil_niektory(P, K) ←
        navstivil(_, P, K),
        navstivil(I, _, K),
        vypil(I, A, _),
        not v(P, A).

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

SQL:

```
with
nevypil_niektory as (
        select np.Pijan, np.Krcma
        from navstivil np, navstivil n, vypil v
        where np.Krcma = n.Krcma and n.Idn = v.Idn and not exists (
                select *
                from vypil v2
                where v2.Pijan = np.Pijan and v2.Alkohol = v.Alkohol)
        )
)
select n.Pijan, n.Krcma
from navstivil n
where not exists (
        select *
        from nevypil_niektory nn
        where n.Pijan = nn.Pijan and n.Krcma = nn.Krcma
)
```

Relačná algebra:

$v := \pi_{Pijan,\ Alkohol}\ (navstivil \bowtie vypil)$

$nevypil\_niektory := \pi_{Pijan,\ Krcma}\ ($

$(\pi_{np.Pijan,\ n.Krcma,\ Alkohol}\ (\rho_{np}\ (navstivil) \bowtie_{np.Krcma\ =\ n.Krcma} \rho_{n}\ (navstivil) \bowtie_{n.Idn\ =\ vypil.Idn}\ vypil) \triangleright v))$

/* answer */

$\pi_{Pijan,\ Krcma}\ (navstivil) \triangleright nevypil\_niektory$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** na dvojice [P, K] také, že P ľúbi aspoň jeden alkohol; a každý alkohol, ktorý pijan P ľúbi, vypil P v krčme K vo väčšom celkovom množstve než vo všetkých ostatných krčmách dokopy.

```
answer(P, K) ←
        lubi(P, _),
        niektory_malo(P, K).

niektory_malo(P, K) ←
        lubi(P, A),
        suma(P, A, K, S1),
        suma_inde(P, A, K, S2),
        S2 >= S1.

suma(P, A, K, S) ←
        subtotal(vypite(_, P, K, A, M), [P, K, A], [S = sum(M)]).

suma_inde(P, A, K, S) ←
        subtotal(vypite_inde(_, P, K, A, M), [P, K, A], [S = sum(M)]).

vypite(I, P, K, A, M) ←
        navstivil(I, P, K),
        vypil(I, A, M).

vypite_inde(I, P, K, A, M) ←
        navstivil(_, _, K),
        navstivil(I, P, K2),
        vypil(I, A, M),
        not K = K2.
```

2. a) Definujte čo znamená, že relačná schéma [r, F] je v tretej normálnej forme. **(6)**

Relačná schéma [r, F] je v tretej normálnej forme, ak pre každú kánonickú netriviálnu funkčnú závislosť **X→Y** z F^+ platí, že **X** je nadkľúč r, alebo **Y** je časťou niektorého kľúča r.

b) Uvažujte reláciu r(A, B, C, D, E, F, G, H), v ktorej platia funkčné závislosti
ABCG→D, BH→AD, DE→CF, AH→DG, G→BE, DF→G, AC→B.
Rozhodnite, či dekompozícia r do r1(A, B, D, E, F, G, H), r2(A, C, H) je v tretej normálnej forme. Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Nájdime všetky kľúče r:
ABCDEFGH
-A: BCDEFGH
-B: CDEFGH
-C: DEFGH
-D: EFGH
-E: FGH
-F: **GH**
+F: FH
+E: EFH
+D: DEFH
-E: **DFH**
+E: **DEH**
+C: CDEFH
-D: CEFH
+D: CDH
+B: **BH**
+A: **AH**

Všetky kľúče r sú: AH, BH, DEH, DFH, GH.
Každý atribút r1 patrí do niektorého kľúča, takže r1 je v tretej normálnej forme.
V r2 iba atribút C nepatrí do žiadneho kľúča. Tretiu normálnu formu r2 by mohli porušiť len funkčné závislosti A→C alebo H→C. Lenže žiadna z nich neplatí. Preto aj r2 je v tretej normálnej forme.
ÁNO, daná dekompozícia je v tretej normálnej forme.

c) Nech v úlohe b) všetky atribúty r1 sú typu *int*. Napíšte SQL príkaz, ktorý vytvorí prázdnu reláciu r1, a ktorý obsahuje klauzu *check* aplikovanú na r1. Do klauzy *check* napíšte booleovský výraz (podobný klauze *where* v *select*), ktorý garantuje splnenie funkčnej závislosti AH→DG v r1 (t.j. ten výraz je true práve vtedy, keď pre naplnenie r1 platí táto funkčná závislosť). **(6)**

```
create table r1 (
        A int,
        B int,
        D int,
        E int,
        F int,
        G int,
        H int,
        check (
                not exists (
                        select *
                        from r1 r1x, r1 r1y
                        where r1x.A = r1y.A and r2x.H = r1y.H and (r1x.D <> r1y.D or r1x.G <> r1y.G)
        )
)
```

3. Uvažujte relácie r(X, Y) a s(X, Y) bez duplikátov a NULL hodnôt, všetky atribúty sú typu *int*.
a) Zapíšte nasledujúci dotaz v relačnom kalkule: **(6)**
select distinct r.X from r, s where r.Y = 3 and r.Y < s.Y and s.Y > 7.

{RX: ∃SX∃SY r(RX, 3) ∧s(SX, SY) ∧3 < SY ∧SY > 7}

b) Rozhodnite, či nasledujúci dotaz a dotaz úlohy a) vypočítajú rovnaký výsledok pre ľubovoľné naplnenie relácií r a s:
select distinct r.X from r where r.Y = 3.
Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

NIE. Napríklad pre r = {[1, 3]}, s = {[1, 1]} je výsledkom vyššieuvedeného dotazu {3}, zatiaľ čo výsledkom dotazu z úlohy a) je prázdna množina.

4. Popíšte metódu riešenia deadlockov wait-or-die pri dvojfázovom zamykaní. Uveďte konkrétny rozvrh, pre ktorý nastane deadlock, ak sa použije dvojfázové zamykanie bez wait-or-die; a popíšte čo najpresnejšie akcie, ktoré urobí na tom rozvrhu scheduler používajúci dvojfázové zamykanie s wait-or-die. **(6)**

Keď systém vykonáva operáciu $start_T$, vytvorí časovú pečiatku TS(T).

Keď transakcia T1 žiada o zámok, ktorý je konfliktný so zámkom, ktorý v tej chvíli drží transakcia T2, scheduler vykoná nasledujúci program:

```
if (TS(T1) < TS(T2))
        T1 čaká na pridelenie zámku;
else
        abort T1;
```

Uvažujme vstupný rozvrh s_1, $r_1(X)$, s_2, $r_2(Y)$, $w_1(Y)$, $w_2(X)$, c_1, c_2.

Akcie schedulera bez wait-or-die na tomto rozvrhu doplnenom o získavanie a uvoľnovanie zámkov:

s_1 vytvorí TS(T1)
$rl_1(X)$ pridelí T1 read-lock na X
$r_1(X)$ vykoná $r_1(X)$
s_2 vytvorí TS(T2)
$rl_2(Y)$ pridelí T2 read-lock na Y
$r_2(Y)$ vykoná $r_2(Y)$
$wl_1(Y)$ nechá T1 čakať na write-lock na Y
$w_1(Y)$ odloží na neskôr
$wl_2(X)$ nechá T2 čakať na write-lock na X
$w_2(X)$ odloží na neskôr, T1 a T2 sú v deadlocku

Akcie schedulera s wait-or-die na tomto rozvrhu doplnenom o získavanie a uvoľnovanie zámkov:

s_1 vytvorí TS(T1)
$rl_1(X)$ pridelí T1 read-lock na X
$r_1(X)$ vykoná $r_1(X)$
s_2 vytvorí TS(T2)
$rl_2(Y)$ pridelí T2 read-lock na Y
$r_2(Y)$ vykoná $r_2(Y)$
$wl_1(Y)$ nechá T1 čakať na write-lock na Y, lebo TS(T1) < TS(T2)
$w_1(Y)$ odloží na neskôr
$wl_2(X)$ vykoná abort T2, lebo TS(T1) < TS(T2); pridelí T1 write-lock na Y a vykoná $w_1(Y)$
c1 vykoná commit T1