5/2/2025 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)** a SQL **(6)** na dvojice [K, P] také, že pijan P počas žiadnej návštevy krčmy K nevypil všetky svoje obľúbené alkoholy, ale počas ľubovoľných dvoch rôznych návštev K dokopy áno. Zaujímajú nás len dvojice, kde P navštívil K aspoň dvakrát.

Datalog:

```
answer(K, P) ←
        navstivil(I1, P, K),
        navstivil(I2, P, K),
        not I1 = I2,
        not niekedy_vypil_vsetky(P, K),
        not pri_dvoch_nieco_nevypil(P, K).

niekedy_vypil_vsetky(P, K) ←
        navstivil(I, P, K),
        not lubi_nevypil(I).

lubi_nevypil(I) ←
        navstivil(I, P, _),
        lubi(P, A),
        not nv(I, A).

nv(I, A) ←
        navstivil(I, _, _),
        vypil(I, A, _).

pri_dvoch_nieco_nevypil(P, K) ←
        navstivil(I1, P, K),
        navstivil(I2, P, K),
        lubi(P, A),
        not nv(I1, A),
        not nv(I2, A).
```

SQL:

```
with
	lubi_nevypil as (
		select n.Idn
		from navstivil n, lubi l
		where n.Pijan = l.Pijan and not exists (
			select *
			from navstivil n2, vypil v
			where n2.Idn = v.Idn and n2.Idn = n.Idn and
			v.Alkohol = l.Alkohol
		)
	)
	niekedy_vypil_vsetky as (
		select n.Pijan, n.Krcma
		from navstivil n, lubi_nevypil nv
		where n.Idn = nv.Idn
	),
	pri_dvoch_nieco_nevypil nn2 as (
		select n1.Pijan, n1.Krcma
		from navstivil n1, navstivil n2, lubi l
		where n1.Pijan = n2.Pijan and n1.Pijan = l.Pijan and
		n1.Krcma = n2.Krcma and not exists (
			select *
			from navstivil n3, vypil v3
			where n3.Idn = v4.Idn and n3.Idn = n1.Idn and
			v3.Alkohol = l.Alkohol
				) and not exists (
			select *
			from navstivil n4, vypil v4
			where n4.Idn = v4.Idn and n4.Idn = n1.Idn and
			v4.Alkohol = l.Alkohol
		)
	)
select
from navstivil n1, navstivil n2
where n1.Idn <> n2.Idn and n1.Pijan = n2.Pijan and n1.Krcma = n2.Krcma and not exists (
	select *
	from niekedy_vypil_vsetky nvv
	where nvv.Pijan = n1.Pijan and nvv.Krcma = n1.Krcma
	) and not exists (
	select *
	from pri_dvoch_nieco_nevypil pdnn
	where where pdnn.Pijan = n1.Pijan and pdnn.Krcma = n1.Krcma
	)
```

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na dvojice krčiem
[K1, K2] také, že K1 navštívilo menej pijanov než K2; a každý alkohol čapovaný v oboch krčmách sa vypil v K2 vo väčšom celkovom množstve než v K1.

Datalog:

```
answer(K1, K2) ←
        subtotal(navstevy(P, K1), [K1], [C1 = count(P)]),
        subtotal(navstevy(P, K2), [K2], [C2 = count(P)]),
        C1 < C2,
        not niecoho_malo(K1, K2).

niecoho_malo(K1, K2) ←
        capuje(K1, A),
        capuje(K2, A),
        subtotal(alkoholy(_, K1, A, M), [K1, A], [S1 = sum(M)]),
        subtotal(alkoholy(_, K2, A, M), [K2, A], [S2 = sum(M)]),
        S1 >= S2.

niecoho_malo(K1, K2) ←
        capuje(K1, A),
        capuje(K2, A),
        not nv(K2, A).

nv(K, A) ←
        navstivil(I, _, K),
        vypil(I, A, _).

navstevy(P, K) ←
        navstivil(_, P, K).

alkoholy(I, K, A, M) ←
        navstivil(I, _, K),
        vypil(I, A, M).
```

SQL:

```
with
	pocet as (
		select n.Krcma, count(distinct n.Pijan) as C
		from navstivil n
		group by n.Krcma
	),
	mnozstvo as (
		select n.Krcma, v.Alkohol, sum(v.Mnozstvo) as S
		from navstivil n, vypil v
		where n.Idn = v.Idn
		group by n.Krcma, v.Alkohol
	),
	niecoho_malo as (
		(
		select c1.Krcma as K1, c2.Krcma as K2
		from capuje c1, capuje c2, mnozstvo m1, mnozstvo m2
		where c1.Alkohol = c2.Alkohol and c1.Alkohol = m1.Alkohol
		and c1.Alkohol = m2.Alkohol and m1.S >= m2.S
		)
		union
		(
		select c1.Krcma as K1, c2.Krcma as K2
		from capuje c1, capuje c2
		where c1.Alkohol = c2.Alkohol and not exists (
			select *
			from navstivil n, vypil v
			where n.Idn = v.Idn and n.Krcma = c2.Krcma and
			v.Alkohol = c2.Alkohol
		)
		)
	)
select p1.Krcma as K1, p2.Krcma as K2
from pocet p1, pocet p2
where p1.C < p2.C and not exists (
	select *
	from niecoho_malo nm
	where nm.K1 = p1.Krcma and nm.K2 = p2.Krcma
	)
```

2. a) Uvažujte relačnú schému r(A, B, C, D, E, F, G) s funkčnými závislosťami ABCD → EF, ABE → FG, ABDG → CF, G → BD;
a jej dekompozíciu r1(A, B, C, D, E, G), r2(A, C, F, G).
a) Definujte pojem zachovania funkčných závislostí pri dekompozícii relačnej schémy. Uveďte minimálne pokrytie množiny funkčných závislostí, ktoré nie sú zachované v danej dekompozícii. **(6)**

Definícia. Dekompozícia relačnej schémy [r, F] do $[r_1, F_1]$, ..., $[r_n, F_n]$ zachováva všetky funkčné závislosti, ak $F^+ = \cup_{i=1, ..., n} F_i^+$.

Začnime s minimálnym pokrytím množiny funkčných závislostí.
Po minimalizácii ľavých strán kánonických funkčných závislostí:
ABCD → E, ABCD → F, ABE → F, ABE → G, AG → C, AG → F, G → B, G → D
Po vynechaní redundantných funkčných závislostí dostávame nejaké minimálne pokrytie:
ABCD → E, ABE → G, AG → C, AG → F, G → B, G → D

V r1 platí ABCD → E, ABE → G, AG → C, G → B, G → D.
V r2 platí AG → F.

Daná dekompozícia zachováva všetky funkčné závislosti pôvodnej relačnej schémy. Tým pádom minimálne pokrytie množiny nezachovaných funkčných závislostí je prázdna množina.

b) Rozhodnite, či daná dekompozícia je v tretej normálnej forme. Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Nájdime všetky kľúče r:
ABCDEFG
- -B: ACDEFG
  - -C: ADEFG
    - -D: AEFG
      - -E: AFG
        - -F: **AG**
        - +F: AF
      - +E: AEF
    - +D: ADEF
  - +C: ACDEF
- +B: ABCDEF
  - -C: ABDEF
    - -D: ABEF
      - -E: ABF
      - +E: **ABE**
    - +D: ABDF
  - +C: ABCDF
    - -D: ABCF
    - +D: **ABCD**

Všetky kľúče v r sú: ABCD, ABE, AG.

Každý atribút okrem F patrí do nejakého kľúča. Teda r1 je v 3NF.
V r2 neplatí žiadna funkčná závislosť s najviac jedným atribútom na ľavej strane (a s najmenej jedným na pravej). Jediné platné funkčné závislosti s dvomi atribútmi na ľavej strane (a s najmenej jedným na pravej) majú na ľavej strane kľúč AG. Teda r2 je v 3NF.

ÁNO, daná dekompozícia je v 3NF.

c) Dekomponujte r do Boyce-Coddovej normálnej formy, bezstratovo. Vyhnite sa zbytočnému zlomeniu funkčných závislostí. **(6)**

Začnime s 3NF dekompozíciou z minimálneho pokrytia:
r1(A, B, C, D, E),
r2(A, B, E, G),
r3(A, C, G),
r4(A, F, G),
r5(B, D, G).

V r1 platia ABCD → E a ABE → G, ale tie majú na ľavej strane nadkľúč r1. Iné netriviálne funkčné závislosti v r1 neplatia, teda r1 je v BCNF.
V r2 platí G → B, pričom G nie je nadkľúč. Dekomponujeme r2 do (A, E, G) a (B, G), ktoré sú v BCNF. Keďže (B, G) je podmnožinou r5, nepridáme ju do dekompozície.
V r3 neplatí žiadna netriviálna funkčná závislosť s najviac dvomi atribútmi na ľavej strane, teda r3 je v BCNF.
V r4 neplatí žiadna netriviálna funkčná závislosť s najviac dvomi atribútmi na ľavej strane, teda r4 je v BCNF.
V r5 platia G → B a G → D, ale G je nadkľúč r5. Teda r5 je v BCNF.

BCNF dekompozícia, bezstratová (nezachováva ABE → G):
(A, B, C, D, E),
(A, E, G),
(A, C, G),
(A, F, G),
(B, D, G).

3. Uvažujte Datalogový program s extenzionálnou databázou r(X, Y):
p(X, Y) ← r(X, Y), not q(X), not q(Y).
q(X) ← r(X, Y), s(Y).
s(X) ← r(_, X), q(X).
a) Zapíšte výpočet dotazu ? p(X, Y) naivnou evaluáciou. **(6)**

p := {};
q := {};
s := {};
ϕ (
    p := r ▷ $\rho_{q(Z)}$ (q) ▷ $\rho_{q(Y)}$ (q);
    q := π_X (r ⋈ $\rho_{s(Y)}$ (s));
    s := π_X ($\rho_{r(Y, X)}$ (r) ⋈ $\rho_{q(X)}$ (q));
);
/* answer */
p

b) Uveďte výsledok dotazu pre r(X, Y) = {[0, 0], [0, 1], [1, 0], [1, 2]}. **(6)**

Naivná evaluácia, po inicializácii:
p = {},
q = {},
s = {}.

Po 1. iterácii ϕ:
p = {[0, 0], [0, 1], [1, 0], [1, 2]},
q = {},
s = {}.

Po 2. iterácii ϕ:
p = {[0, 0], [0, 1], [1, 0], [1, 2]},
q = {},
s = {}.

Tu výpočet ϕ končí. Výsledok dotazu:
{[0, 0], [0, 1], [1, 0], [1, 2]}.

c) Rozhodnite, či výpočet dotazu ? p(X, Y) naivnou evaluáciou skončí pre ľubovoľné naplnenie extenzionálnej databázy r(X, Y). Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

ÁNO.

Daný program je stratifikovateľný: stratum(q) = stratum(s) = 1, stratum(p) = 2.
Naivná evaluácia skončí pre každý stratifikovateľný program, teda aj pre tento.