10/1/2024 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 relačnej algebre **(6)** na dvojice [P, K] také, že pijan P vypil pri každej návšteve krčmy K rovnakú množinu alkoholov. (Predpokladajte, že P navštívil K aspoň raz.)

Datalog:

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

odlisne_navstevy(P, K) ←
        navstivil(I1, P, K),
        v(I1, A),
        navstivil(I2, P, K),
        not v(I2, A).

v(I, A) ←
        vypil(I, A, _).
```

Relačná algebra:

$$odlisne\_navstevy = \pi_{Pijan,\ Krcma}\ (\ (\rho_{n1(I1,\ Pijan,\ Krcma)}\ (navstivil) \bowtie \rho_{v(I1,\ A,\ M1)}\ (vypil) \bowtie \rho_{n1(I2,\ Pijan,\ Krcma)}\ (navstivil)) \triangleright \rho_{v(I2,\ A,\ M2)}\ (vypil))$$

/* answer */

$$\pi_{Pijan,\ Krcma}\ (navstivil) \triangleright odlisne\_navstevy$$

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

Datalog:

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

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

niecoho_malo(P, K) ←
        subtotal(nv(_, P, K, A, M), [P, K, A], [S1 = sum(M)]),
        subtotal(nv_inde(_, P, K, A, M), [P, K, A], [S2 = sum(M)]),
        not S1 > S2.

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

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

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

SQL:

```
with
snv as (
	select n.Pijan, n.Krcma, v.Alkohol, sum(v.Mnozstvo) as Suma
	from navstivil n, vypil v
	where n.Idn = v.Idn
	group by n.Pijan, n.Krcma, v.Alkohol
),
snv_inde as (
	select n1.Pijan, n1.Krcma, v2.Alkohol, sum(v2.Mnozstvo) as Suma
	from navstivil n1, navstivil n2, vypil v2
	where n1.Pijan = n2.Pijan and n1.Krcma <> n2.Krcma and n2.Idn = v2.Idn
	group by n1.Pijan, n1.Krcma, v2.Alkohol
),
niecoho_malo as (
	(
	select n.Pijan, c.Krcma
	from capuje c, navstivil n
	where c.Krcma = n.Krcma and not exists (
		select *
		from navstivil n2, vypil v2
		where n2.Krcma = c.Krcma and n2.Pijan = n.Pijan and v2.Alkohol = c.Alkohol and
			n2.Idn = v2.Idn
		)
	)
	union
	(
	select s.Pijan, s.Krcma
	from snv s, snv_inde s_inde
	where s.Pijan = s_inde.Pijan and s.Krcma = s_inde.Krcma and s.Alkohol = s_inde.Alkohol and
		s.Suma <= s_inde.Suma
	)
)
select n.Pijan, n.Krcma
from navstivil n
where not exists (
	select *
	from niecoho_malo nm
	where n.Pijan = nm.Pijan and n.Krcma = nm.Krcma
	)
```

2. Uvažujte relačnú schému r(A, B, C, D) s funkčnými závislosťami
F = {{A, B} → {A, C, D}, {A, C, D} → {A}, {C, D} → {B}, {A, C} → {B},
{A, D} → {B, C}, {} → {C}}.

a) Napíšte v Datalogu bezpečný program s dotazom, ktorý vráti *true* práve vtedy, keď sú v r (súčasne) splnené všetky funkčné závislosti z F^+. **(6)**

Nájdime nejaké minimálne pokrytie F, tým sa ten program trošku zjednoduší.
Kánonické funkčné závislosti:
AB → A, AB → C, AB → D, ACD → A, CD → B, AC → B, AD → B, AD → C, {} → {C}
Po minimalizácii ľavých strán:
A → A, {} → C, A → D, A → A, D → B, A → B, A → B, {} → C, {} → {C}
Po vynechaní redundantných funkčných závislostí:
{} → C, A → D, D → B je minimálne pokrytie F.

```
answer ←
        not zero_c_broken,
        not a_d_broken,
        not d_b_broken.

zero_c_broken ←
        r(_, _, C1, _),
        r(_, _, C2, _),
        not C1 = C2.

a_d_broken ←
        r(A, _, _, D1),
        r(A, _, _, D2),
        not D1 = D2.

d_b_broken ←
        r(_, B1, _, D),
        r(_, B2, _, D),
        not B1 = B2.

? answer.
```

b) Rozhodnite či existuje dekompozícia r, ktorá je v Boyce-Coddovej normálnej forme, je bezstratová (spája sa bezstratovo) a zachováva všetky funkčné závislosti. Odpoveď ÁNO resp. NIE zdôvodnite; t.j. buď napíšte tú dekompozíciu a vysvetlite prečo má požadované vlastnosti, alebo vysvetlite prečo taká neexistuje. **(6)**

ÁNO. Hľadaná dekompozícia je **r1 = {C}, r2 = {A, D}, r3 = {B, D}**.
C je konštantný atribút. Spoločný atribút r2 a r3 je D, čo je (nad)kľúč v r2. **Táto dekompozícia je bezstratová**. (Bezstratová je tiež preto, že bola skonštruovaná z minimálneho pokrytia, pričom r2 je nadkľúč v r.)
r1 je v BCNF. r2 aj r3 sú v BCNF, lebo žiaden atribút v r2 ani v r3 sa nedá určiť z prázdnej množiny, a funkčné závislosti s 1 a viac atribútmi na ľavej strane neohrozujú BCNF relácií r2 a r3. Teda **táto dekompozícia je v BCNF.**
Funkčná závislosť {} → C je zachovaná v r1. Funkčná závislosť A → D je zachovaná v r2. Funkčná závislosť D → B je zachovaná v r3. **Táto dekompozícia zachováva všetky platné funkčné závislosti** (uzáver minimálneho pokrytia je F^+).

3. Uvažujte SQL dotaz nad reláciou r(X, Y, Z, W) bez duplikátov a null hodnôt:
select distinct r1.X, sum(r1.Z) as S, sum(distinct r1.Z) as D
from r r1 group by r1.X, r1.Y having sum(r1.Z) < sum(distinct r1.Z)

a) Uveďte príklad naplnenia r, pre ktoré výsledok dotazu obsahuje dve trojice, uveďte výsledok dotazu. **(6)**

Pre **r = {[0, 0, -1, 0], [0, 0, -1, 1], [1, 1, -1, 0], [1, 1, -1, 1]}** je výsledkom dotazu **{[0, -2, -1], [1, -2, -1]}**.

b) Sformulujte daný dotaz ekvivalentne v Datalogu. **(6)**

```
r3(X, Y, Z) ←
        r(X, Y, Z, _).

answer(X, S, D) ←
        subtotal(r(X, Y, Z, _), [X, Y], [S = sum(Z)]),
        subtotal(r3(X, Y, Z), [X, Y], [D = sum(Z)]),
        S < D.

? answer(X, S, D).
```

4. Rozhodnite platnosť nasledujúcich tvrdení. Odpovede ÁNO resp. NIE zdôvodnite; t.j. buď dokážte tvrdenie, alebo uveďte kontrapríklad a vysvetlite ho.

a) Ak rozvrh H je konflikt-sériovateľný, tak pre každý prefix H existuje sériový rozvrh, ktorému je konflikt-ekvivalentná commitovaná projekcia toho prefixu H. **(6)**

ÁNO, toto platí.
Ak H je konflikt-sériovateľný, tak z definície existuje sériový rozvrh S, ktorému je konflikt-ekvivalentná commitovaná projekcia H. Odoberme z H poslednú operáciu, ten skrátený rozvrh nazvime H'. Odobratie poslednej operácie nemení relatívne poradie konfliktov v rozvrhu, a nepridáva žiadne nové konflikty. Sú len dve možnosti:
1. Odobratá operácia je iná ako c_T. V tomto prípade je commitovaná projekcia H' identická s commitovanou projekciou H. To znamená, že aj H' je konflikt-ekvivalentný sériovému rozvrhu S.
2. Odobratá operácia je c_T. V tomto prípade je commitovaná projekcia H' skrátená oproti H o transakciu T (všetky operácie transakcie T sa z commitovanej projekcie H vynechajú). Relatívne poradie konfliktov ostatných transakcií ostane nezmenené, takže H' bude konflikt-ekvivalentný sériovému rozvrhu S', v ktorom je tiež oproti S vynechaná celá transakcia T. T.j. aj H' je konflikt-sériovateľný rozvrh.
Táto úvaha sa dá aplikovať aj na skrátený rozvrh H', teda uvedené tvrdenie platí pre každý prefix H.

b) Ak rozvrh H je konflikt-sériovateľný, tak existuje sériový rozvrh, ktorému je konflikt-ekvivalentná commitovaná projekcia každého prefixu H. **(6)**

NIE, toto neplatí.
Uvažujme konflikt-sériovateľný rozvrh H = s1, r1(X), c1, s2, w2(X), c2. Tento rozvrh je dokonca sériový, jeho commitovaná projekcia je konflikt-ekvivalentná rozvrhu H. Rozvrh H je jediný sériový rozvrh transakcií T1 a T2, ktorému je konflikt-ekvivalentná commitovaná projekcia H (sériový rozvrh, v ktorom sú T1 a T2 v opačnom poradí, nie je konflikt-ekvivalentný rozvrhu H).
Uvažujme prefix H' = s1, r1(X), c1. Commitovaná projekcia H' (identická s H') nie je konflikt-ekvivalentná rozvrhu H, lebo neobsahuje operácie transakcie T2, zatiaľ čo H ich obsahuje.