27/1/2015 Úvod do databáz, skúškový test, max 25 bodov, 90 min
1. Daná je databáza (bez duplikátov a null hodnôt): capuje(Krcma, Alkohol, Cena),
lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Krcma, Alkohol→Cena; Idn→Pijan, Krcma; Idn, Alkohol→Mnozstvo; Mnozstvo > 0.
a) Nájdite dvojice [P, A] také, že pijan P ľúbi alkohol A; a zároveň každá krčma, v ktorej P vypil A, čapuje alkohol A lacnejšie než ktorákoľvek iná krčma, ktorá čapuje A. Sformulujte tento dotaz v relačnom kalkule **(2)**, Datalogu **(2)** a relačnej algebre **(2)**.

*[P, A]: P ľúbi A a zároveň **neplatí**, že pre **niektorú** krčmu K v ktorej P vypil A **neplatí**, že K čapuje alkohol A lacnejšie než ktorákoľvek iná krčma.*
*[P, A]: P ľúbi A a zároveň neplatí, že pre niektorú krčmu K v ktorej P vypil A **platí**, že **niektorá iná krčma K2 čapuje A aspoň tak lacno ako K**.*

Relačný kalkul:
{[P, A]:
lubi(P, A) ∧
¬ /* zla_krcma(P, A) */
(
∃I ∃K ∃M ∃C ∃C2
navstivil(I, P, K) ∧
vypil(I, A, M) ∧
capuje(K, A, C) ∧
capuje(K2, A, C2) ∧
K ≠ K2 ∧
C2 <= C
)
}

Datalog:
answer(P, A) ←
lubi(P, A),
not zla_krcma(P, A).

zla_krcma(P, A) ←
navstivil(I, P, K),
vypil(I, A, _),
capuje(K, A, C),
capuje(K2, A, C2),
not K = K2,
C2 <= C.

Relačná algebra:
zla_krcma := $\pi_{Pijan,\ Alkohol}$ (((navstivil ⋈ vypil) ⋈ capuje) ⋈ $_{Krcma \neq K2 \wedge Alkohol = A2 \wedge C2 <= Cena}$ $P_{c(K2,\ A2,\ C2)}$ (capuje))
/* answer */
lubi – zla_krcma

b) Nájdite pijanov P, ktorí spĺňajú aspoň jednu z nasledujúcich podmienok: P navštívil viacej (rôznych) krčiem než ostatní pijani dokopy; každý alkohol, ktorý P ľúbi, vypil P vo väčšom celkovom množstve než ostatní pijani dokopy. Sformulujte v Datalogu **(2)** a SQL **(2)**.

Datalog:

```
answer(P) ←   /* pocet krciem, ktore navstivil P, je vacsi nez pocet krciem ktore navstivili ostatni */
        subtotal(n(P, K), [P], [C = count(K)]),
        subtotal(n_ostatni(P, K), [P], [C = count(K)]),
        C > C2.

answer(P) ←   /* P nejaku krcmu navstivil, ostatni nie */
        n(P, _),
        not n2_ostatni(P).

answer(P) ←   /* P prekonal ostatnych pijanov v piti kazdeho oblubeneho alkoholu */
        l(P, _), /* povedzme, ze hladany pijan musi lubit nejaky alkohol, hoci by sa dal definovat aj inak */
        not lubi_neprekonal(P).

n(P, K) ←
        navstivil(_, P, K).

n_ostatni(P, K) ←
        navstivil(_, P, _),
        navstivil(_, P2, K),
        not P = P2.

n2_ostatni(P) ←
        n_ostatni(P, _).

lubi_neprekonal(P) ←/* P neprekonal ostatnych pijanov v piti oblubeneho alkoholu A, hoci A vypil */
        subtotal(nvl(I, P, A, M), [P, A], [T = sum(M)]),
        subtotal(nvl_ostatni(I, P, A, M), [P, A], [T = sum(M)]),
        T2 >= T.

lubi_neprekonal(P) ←/* P neprekonal ostatnych pijanov v piti oblubeneho alkoholu A, lebo A nevypil */
        lubi(P, A),
        not nvl2(P, A).

nvl(I, P, A, M) ←
        lubi(P, A),
        navstivil(I, P, _),
        vypil(I, A, M).

nvl2(P, A) ←
        nvl(_, P, A, _).

nvl_ostatni(I, P, A, M) ←
        lubi(P, A),
        navstivil(I, P2, _),
        vypil(I, A, M),
        not P = P2.
```

SQL:

```
create temporary table pocet_krciem as
select n.Pijan, count(distinct n.Krcma) as C
from navstivil n
group by n.Pijan

create temporary table pocet_krciem_ostatni as
(
	select n.Pijan, count(distinct no.Krcma) as C
	from navstivil n, navstivil no
	where n.Pijan <> no.Pijan
	group by n.Pijan
)
union
(
	select n.Pijan, 0 as C
	from navstivil n
	where not exists (
		select *
		from navstivil no n.Pijan <> no.Pijan
	)
)

create temporary table vypite_oblubene as
(
	select l.Pijan, l.Alkohol, sum(v.Mnozstvo) as S
	from lubi l, navstivil n, vypil v
	where l.Pijan = n.Pijan and l.Alkohol = v.Alkohol and n.Idn = v.Idn
	group by l.Pijan, l.Alkohol
)
union
(
	select l.Pijan, l.Alkohol, 0 as S
	from lubi l
	where not exists (
		select *
		from navstivil n, vypil v
		where n.Pijan = l.Pijan and l.Alkohol = v.Alkohol and n.Idn = v.Idn
	)
)

create temporary table vypite_oblubene_ostatni as
select l.Pijan, l.Alkohol, sum(vo.Mnozstvo) as S
from lubi l, navstivil no, vypil vo
where l.Pijan <> no.Pijan and l.Alkohol = vo.Alkohol and no.Idn = vo.Idn
group by l.Pijan, l.Alkohol
```

```
/* answer */
(
	select pk.Pijan
	from pocet_krciem pk, pocet_krciem_ostatni pko
	where pk.Pijan = pko.Pijan and pk.C > pko.C
)
union
(
	select l.Pijan
	from lubi l
	where not exists (
		select *
		from vypite_oblubene v, vypite_oblubene vo
		where v.Pijan = l.Pijan and vo.Pijan = l.Pijan and v.Alkohol = vo.Alkohol and v.S <= vo.S
	)
)
```

Formulácia „... P navštívil viacej (rôznych) krčiem než ostatní pijani dokopy“ sa interpretuje tak, že mohutnosť množiny krčiem, ktoré navštívil P, je väčšia ako mohutnosť množiny krčiem, ktorí navštívili dokopy ostatní pijani. Napríklad, ak P navštívil krčmy K1 a K2, obaja pijani P1 a P2 navštívili krčmu K1 a iné návštevy nie sú, tak P bude vo výsledku dotazu.

Logika dotazu:
Pijan P je vo výsledku, keď počet krčiem navštívených P je väčší než počet krčiem navštívených ostatnými pijanmi. Toto zahŕňa aj prípad, keď pijan P navštívil aspoň jednu krčmu a ostatní pijani nie.
Pijan P je vo výsledku aj vtedy, keď nie je pravda, že existuje taký alkohol A, ktorý P ľúbi a v pití A neprekonal ostatných pijanov. Časť pod negáciou zahŕňa aj prípad, keď P ľúbi nejaký alkohol, ktorý nikdy nevypil.

2. Daná je relácia r(A, B, C, D, E, F, G, H) s funkčnými závislosťami
B, E → G, H; B, E, G → F; A, D → C; F → B; B, F → A
a) Rozhodnite, či je dekompozícia r1(A, B, F), r2(B, E, F, G, H), r3(C, D), r4(D, E, F) v tretej normálnej forme (3NF). Zdôvodnite. **(2)**

r1 je v 3NF, lebo v nej platí iba jedna netriviálna závislosť o 2 atribútoch, F → B, pričom F je kľúč v r1.
r3 je v 3NF, lebo každá binárna relácia je v 3NF.
r4 je v 3NF, lebo v nej neplatí žiadna netriviálna funkčná závislosť.
V relácii r2 platí netriviálna funkčná závislosti F → B, pričom F nie je kľúčom v r2. Lenže {B, E} je kľúč v r2, takže r2 je v 3NF. Keďže {B, E} je kľúč v r2, funkčné závislosti s ľavou stranou obsahujúcou {B, E} neohrozujú 3NF.
Daná dekompozícia je v tretej normálnej forme.

b) Rozhodnite, či je dekompozícia z úlohy a) bezstratová. Zdôvodnite. **(2)**

Použijeme chase algoritmus.

	A	B	C	D	E	F	G	H
ABF	a1	a2				a6		
BEFGH		a2			a5	a6	a7	a8
CD			a3	a4				
DEF				a4	a5	a6		

F → B:

	A	B	C	D	E	F	G	H
ABF	a1	a2				a6		
BEFGH		a2			a5	a6	a7	a8
CD			a3	a4				
DEF		a2		a4	a5	a6		

BF → A:

	A	B	C	D	E	F	G	H
ABF	a1	a2				a6		
BEFGH	a1	a2			a5	a6	a7	a8
CD			a3	a4				
DEF	a1	a2		a4	a5	a6		

BE → GH:

	A	B	C	D	E	F	G	H
ABF	a1	a2				a6		
BEFGH	a1	a2			a5	a6	a7	a8
CD			a3	a4				
DEF	a1	a2		a4	a5	a6	a7	a8

Daná dekompozícia nie je bezstratová (nespája sa bezstratovo).

c) Nájdite niektoré minimálne pokrytie danej množiny funkčných závislostí. **(2)**

Po minimalizácii ľavých strán kánonických funkčných závislostí:
BE → G
BE → H
BE → F
AD → C
F → B
F → A

Toto je minimálne pokrytie.

d) Dekomponujte reláciu r do Boyce-Coddovej normálnej formy, bezstratovo. Vyhnite sa zbytočnému rozbitiu funkčných závislostí. **(2)**

Začnime s 3NF dekompozíciou z minimálneho pokrytia (BDE je kľúč relácie r), relácie s rovnakou ľavou stranou môžeme (ale nemusíme) spojiť do jednej:
r1(B, E, F, G, H)
r2(A, C, D)
r3(A, F)
r4(B, D, E)

V r1 platí F → B, ktorá porušuje BCNF. r1 rozbijeme do (E, F, G, H) a (B, F). Tieto relácie sú už v BCNF.
V r2 a r4 neplatí žiadna netriviálna funkčná závislosť okrem AD → C. Teda r2 a r4 sú v BCNF.
Binárna relácia r3 je v BCNF.

BCNF dekompozícia:
(A, C, D)
(A, F)
(B, D, E)
(B, F)
(E, F, G, H)

3. Daný je Datalogový program
f(X, Y) ← a(X, _), a(Y, _), not g(X, Y).
g(X, Y) ← a(X, Z), a(Y, _), not a(Y, Z).
g(Y, X) ← a(X, Z), a(Y, _), not a(Y, Z).

a) Nech a je EDB relácia s atribútmi X, Y. Preložte dotaz „? f(X, 1)“ do SQL. **(2)**

```
create temporary table g as
(
	select a1.X, a2.X as Y
	from a a1, a a2
	where not exists (
		select *
		from a a3
		where a3.X = a2.X and a3.Y = a1.Y
	)
)
union
(
	select a2.X, a1.X as Y
	from a a1, a a2
	where not exists (
		select *
		from a a3
		where a3.X = a2.X and a3.Y = a1.Y
	)
)

/* f(X, 1) */
select a1.X, a2.X as Y
from a a1, a a2
where a2.X = 1 and not exists (
	select *
	from g g1
	where g1.X = a1.X and g1.Y = a2.X)
```

b) Vypočítajte výsledok dotazu „? f(X, Y)“ pre reláciu
a(X, Y) = {[1, 1], [1, 4], [2, 1], [2, 3], [2, 4], [3, 1], [4, 1], [4, 4]}. **(2)**

Najskôr vypočítame reláciu g(., .):
g(., .) = {[1, 2], [2, 1], [1, 3], [3, 1], [2, 3], [3, 2], [2, 4], [4, 2], [3, 4], [4, 3]}
V f(., .) sú všetky dvojice prvých atribútov relácie a(., .), ktoré netvoria usporiadanú dvojicu v g.
f(., .) = **{[1, 1], [1, 4], [2, 2], [3, 3], [4, 1], [4, 4]}**.
Tá druhá množina je výsledkom dotazu ? f(X, Y).

4. a) Uvažujte (čiastočný) vstupný rozvrh start1, rl1(X), start2, wl2(Y), rl1(Y), wl2(X). Uveďte akciu resp. akcie, ktoré systém používajúci dvojfázové zamykanie a metódu wait-or-die vykoná po prečítaní každej operácie tohto rozvrhu. **(2)**

start1	Systém pridelí T1 časovú pečiatku
rl1(X)	Systém pridelí T1 read-lock na X
start2	Systém pridelí T2 časovú pečiatku
wl2(Y)	Systém pridelí T2 write-lock na X
rl1(Y)	Systém porovná časové pečiatky T1 a T2. Keďže T1 je staršia, čaká na read-lock na Y
wl2(X)	Systém porovná časové pečiatky T1 a T2. Keďže T2 je mladšia, systém abortuje T2

b) Aký spôsob riešenia deadlocku by ste navrhli použiť v systéme, ktorý používa na izoláciu transakcií validačnú metódu? Vysvetlite. **(1)**

Žiaden. Pri validačnej metóde deadlocky nevznikajú.