19/1/2017 Úvod do databáz, skúškový test, max **60** bodov

1. Uvažujte databázu bez duplikátov a null hodnôt:
lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Idn → Pijan, Krcma; Idn, Alkohol → Mnozstvo; Mnozstvo > 0.
a) Sformulujte v relačnom kalkule **(6)** a SQL **(6)** bezpečný dotaz na dvojice [P, A] také, že pijan P ľúbi alkohol A; a zároveň P aspoň raz vypil A v každej krčme, ktorú navštívil.

Relačný kalkul:

```
{
        [P, A]:
        lubi(P, A) ∧ ¬
        ( /* navstivil_nevypil(P, A) */
                ∃I ∃K
                navstivil(I, P, K) ∧ ¬
                (
                        /* niekedy_vypil(P, K, A) */
                        ∃I ∃M
                        navstivil(I, P, K) ∧ vypil(I, A, M)
                )
        )
}
```

SQL:

```
with navstivil_nevypil as
        (
        select n.Pijan, l.Alkohol
        from navstivil n, lubi l
        where n.Pijan = l.Pijan and not exists
                (
                select *
                from navstivil nv, vypil v
                where n.Pijan = nv.Pijan and n.Krcma = nv.Krcma and nv.Idn = v.Idn and l.Alkohol = v.Alkohol
                )
        )
select l.Pijan, l.Alkohol
from lubi l
where not exists
        (
        select *
        from navstivil_nevypil nn
        where l.Pijan = nn.Pijan and l.Alkohol = nn.Alkohol
        )
}
```

b) Sformulujte dotaz v Datalogu **(6)** a SQL **(6)** na pijanov, ktorí držia rekord v miešaní alkoholov počas jednej návštevy, t.j. spomedzi všetkých pijanov vypili maximálny počet druhov alkoholu počas niektorej svojej návštevy. (Rekord môže držať niekoľko pijanov.)

Datalog:

```
answer(P) ←
        pocet(P, C),
        not iny_viac(P, C).

vpn(I, P, A) ←
        navstivil(I, P, _),
        vypil(I, A, _).

pocet(P, C) ←
        subtotal(vpn(I, P, A), [I, P], [C = count(A)]).

iny_viac(P, C) ←
        pocet(P, C),
        pocet(P2, C2),
        not P2 = P,
        C2 > C.
```

Keď chceme byť veľmi pedantní, môžeme sa špeciálne venovať prípadu, keď žiaden pijan nič nevypil. V takom prípade je každý pijan rekordérom. To robia nasledujúce pravidlá.

```
answer(P) ←
        lubi(P, _),
        not niekto_nieco_vypil.

answer(P) ←
        navstivil(_, P, _),
        not niekto_nieco_vypil.

niekto_nieco_vypil ←
        vypil(_, _, _).
```

SQL (nebudeme riešiť prípad, keď nikto nič nevypil):

```
with pocet as
        (
        select n.Pijan, count(distinct v.Alkohol) as C
        from navstivil n, vypil v
        where n.Idn = v.Idn
        group by n.Idn, n.Pijan
        )
select p.Pijan
from pocet p
where not exists
(
        select *
        from pocet p2
        where p.Pijan <> p2.Pijan and p.C < p2.C
)
```

2. Uvažujte nasledujúci SQL dotaz nad reláciou r(A, B, C):

```
select r1.C, sum(distinct r1.B) as S
from r r1, r r2
where not exists (select * from r r3 where r1.B = r3.C)
group by r1.C
having r1.C < 6;
```

a) Zapíšte dotaz v relačnom kalkule. **(6)**

$$
\{
\quad [C1, S]:
\quad \blacktriangledown\ S = sum(B1)
\quad (
\quad\quad \exists A1\ \exists A2\ \exists B2\ \exists C2
\quad\quad r(A1, B1, C1) \wedge\ r(A2, B2, C2) \wedge \neg
\quad\quad (
\quad\quad\quad \exists A3\ \exists B3
\quad\quad\quad r(A3, B3, B1)
\quad\quad)
\quad)
\quad \wedge C < 6
\}
$$

Alebo, ekvivalentne (od premenných A2, B2, C2 sa vyžaduje len splnenie r(A2, B2, C2), stačí dosadiť A2 = A1, B2 = B1, C2 = C1):

$$
\{
\quad [C1, S]:
\quad \blacktriangledown\ S = sum(B1)
\quad (
\quad\quad \exists A1
\quad\quad r(A1, B1, C1) \wedge \neg
\quad\quad (
\quad\quad\quad \exists A3\ \exists B3
\quad\quad\quad r(A3, B3, B1)
\quad\quad)
\quad)
\quad \wedge C < 6
\}
$$

b) Vypočítajte výsledok dotazu pre reláciu
r(A, B, C) = {[2, 1, 2], [2, 1, 3], [2, 2, 2], [3, 4, 3], [3, 2, 5], [3, 4, 5], [4, 4, 5], [5, 4, 3], [5, 6, 3], [5, 1, 6], [5, 3, 6]}. Vysvetlite priebeh výpočtu. **(6)**

V SQL dotaze sa najskôr spracujú klauzy FROM a WHERE. Relácia r2 je irelevantná pre zvyšok výpočtu. Vypočíta sa

$t1 := P_{r1}(r) \triangleright_{r1.B=r3.C} P_{r3}(r) =$

A	B	C
2	1	2
2	1	3
3	4	3
3	4	5
4	4	5
5	4	3
5	1	6

Potom sa spracuje klauza GROUP BY a vypočíta sa agregácia v klauze SELECT:

$t2 := \Gamma_{C,\ sum(distinct\ B)}\ (t1) =$

C	sum(distinct B)
2	1
3	5
5	4
6	1

Potom sa urobí selekcia v klauze HAVING:

$t3 := \sigma_{sum(distinct\ B) < 6}\ (t2) =$

C	sum(distinct B)
2	1
3	5
5	4

Nakoniec sa urobí premenovanie v klauze SELECT:

$P_{S\ =\ sum(distinct\ B)}\ (t3)$

C	S
2	1
3	5
5	4

Výsledok dotazu je relácia (C, S) = {[2, 1], [3, 5], [5, 4]}.

3. a) Napíšte algoritmus (pseudokód), ktorý bezstratovo dekomponuje ľubovoľnú relačnú schému [r, F] do Boyce-Coddovej normálnej formy (BCNF); a zároveň nedekomponuje [r, F], ktorá je v BCNF. Zdôvodnite prečo je výsledná dekompozícia bezstratová (prečo sa spája bezstratovo). **(6)**

Nasledujúca rekurzívna procedúra dekomponuje relačnú schému len ak nie je v BCNF. V podmienke if sa hľadá konkrétny kontrapríklad, ktorý priamo porušuje definíciu BCNF. Podľa definície [r, F] je v BCNF, ak v nej neexistuje platná netriviálna funkčná závislosť (v F^+), ktorej ľavá strana nie je nadkľúč r.

```
decompose(r, F)
{
        if (F+ obsahuje funkčnú závislosť X → Y, kde X ∩ Y = ∅ a X+ ≠ r)
        {
                decompose(r – Y, F);
                decompose (X ∪ Y, F);
        }
        else
                output(r);        /* r je v BCNF */
}
```

Relácie r – Y a X ∪ Y, ktoré vznikajú dekompozíciou v procedúre decompose, sa spájajú bezstratovo do relácie r, lebo obe obsahujú všetky atribúty X a zároveň X je nadkľúč v X ∪ Y.

b) Dekomponujte relačnú schému [r(A, B, C, D, E, F, G),
{ABCD → EF, ABE → FG, ABDG → CF, G → BD}] do BCNF, bezstratovo. Snažte sa o zachovanie funkčných závislostí, vyhnite sa zbytočnej dekompozícii. **(6)**

Použijeme algoritmus z úlohy a).

(A, B, C, D, E, F, G) nie je v BCNF kvôli platnej funkčnej závislosti G → BD (G nie je nadkľúč).
Dekomponujeme ju do relácií (A, C, E, F, G) a (BDG).
(B, D, G) je v BCNF, lebo okrem G → B a G → D v nej neplatia žiadne iné netriviálne funkčné závislosti z F^+ s 1 atribútom na ľavej aj pravej strane. G je nadkľúč v (B, D, G).
(A, C, E, F, G) nie je v BCNF, lebo v nej neplatia žiadne netriviálne funkčné závislosti z F^+.

Dekompozícia (B, D, G), (A, C, E, F, G) je v BCNF.
(Všetky funkčné závislosti sa v žiadnej BCNF dekompozícii nepodarí zachovať.)

4. Rozvrh je podľa definície konflikt-sériovateľný, ak jeho projekcia na commitované transakcie je konflikt-ekvivalentná niektorému sériovému rozvrhu tých commitovaných transakcií.
a) Uveďte definíciu konflikt-ekvivalencie dvoch rozvrhov. **(6)**

Nech i, j sú identifikátory transakcií v rozvrhu H1 aj H2. Rozvrhy H1 a H2 sú konflikt-ekvivalentné, ak (obsahujú operácie od tých istých transakcií a zároveň) pre každú dvojicu konfliktných operácií O_i, O_j z H1, kde $O_i <_{H1} O_j$ (t.j. O_i je v H1 pred O_j) sú tieto dve operácie v rozvrhu H2 v rovnakom poradí, t.j. $O_i(X) <_{H2} O_j(X)$. Dve read/write operácie sú konfliktné, ak sú od rôznych transakcií, týkajú sa rovnakého dátového objektu a aspoň jedna z nich je write (teda pre $i \neq j$ a ľubovoľný objekt X, konfliktné dvojice sú $[r_i(X), w_j(X)]$, $[w_i(X), r_j(X)]$, $[w_i(X), w_j(X)]$).

b) Rozhodnite či platí: ak rozvrh H je konflikt-sériovateľný, tak projekcia *každého jeho prefixu* na commitované transakcie je konflikt-ekvivalentná niektorému sériovému rozvrhu tých commitovaných transakcií. Ak áno, dokážte. Ak nie, uveďte konkrétny kontrapríklad. **(6)**

Toto tvrdenie sa dá interpretovať dvomi spôsobmi (ten druhý je asi prirodzenejší):
1. Existuje sériový rozvrh, ktorému je konflikt-ekvivalentná commitovaná projekcia každého prefixu.
2. Pre každý prefix existuje sériový rozvrh, ktorému je konflikt-ekvivalentná commitovaná projekcia daného prefixu.

Interpretácia 1. NIE. Toto neplatí.
Uvažujme konflikt-sériovateľný rozvrh H = s1, r1(X), c1, s2, w2(X), c2. Tento rozvrh je dokonca sériový, takže 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 (jediná iná možnosť je śeriový rozvrh, v ktorom sú T1 a T2 v opačnom poradí, ten však nie je konflikt-ekvivalentný rozvrhu H).
Uvažujme takýto prefix H: H' = s1, r1(X), c1. Commitovaná projekcia H' je identická s H'. Ale H' nie je konflikt-ekvivalentný s rozvrhom H, lebo v H' chýba transakcia T2.

Interpretácia 2. 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.