6/2/2013 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: Idn → Pijan, Krcma; Krcma, Alkohol → Cena; Idn, Alkohol → Mnozstvo;
Mnozstvo > 0, Cena > 0.
a) Nájdite trojice [A, K1, K2] také, že alkohol A sa čapuje v krčme K1 aj K2; a zároveň každý pijan, ktorý ľúbi alkohol A, vypil A v K1 aj v K2. Sformulujte tento dotaz v relačnom kalkule **(2)**, Datalogu **(2)** a SQL **(2)**.

Relačný kalkul:

$$
\begin{aligned}
&\{[A, K1, K2]: \\
&\quad \exists C1\ \exists C2 \\
&\quad capuje(K1, A, C1) \wedge capuje(K2, A, C2) \wedge \neg \\
&\quad (\ /* \text{ nejaky P lubi A, ale bud v K1 alebo v K2 alkohol A nikdy nevypil } */ \\
&\qquad \exists P \\
&\qquad lubi(P, A) \wedge \neg \\
&\qquad (\ /* \text{ P niekedy vypil A v K1 aj v K2 } */ \\
&\qquad\quad \exists I1\ \exists M1\ \exists I2\ \exists M2 \\
&\qquad\quad navstivil(I1, P, K1) \wedge vypil(I1, A, M1) \wedge \\
&\qquad\quad navstivil(I2, P, K2) \wedge vypil(I2, A, M2) \\
&\qquad) \\
&\quad) \\
&\}
\end{aligned}
$$

Datalog:

```
answer(A, K1, K2) ←
        capuje(K1, A, _),
        capuje(K2, A, _),
        not niekto_lubi_nikdy_nevypil(A, K1),
        not niekto_lubi_nikdy_nevypil(A, K2).

niekto_lubi_nikdy_nevypil(A, K) ←
        lubi(P, A),
        capuje(K, A, _),
        not niekedy_vypil(P, A, K).

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

Alebo:

```
answer(A, K1, K2) ←
        capuje(K1, A, _),
        capuje(K2, A, _),
        not niekto_lubi_nevypil_v_oboch(A, K1, K2).

niekto_lubi_nevypil_v_oboch(A, K1, K2) ←
        lubi(P, A),
        capuje(K1, A, _), /* safety */
        capuje(K2, A, _), /* safety */
        not vypil_v_oboch(P, A, K1, K2).

vypil_v_oboch(P, A, K1, K2) ←
        navstivil(I1, P, K1),
        vypil(I1, A, _),
        navstivil(I2, P, K2),
        vypil(I2, A, _).
```

SQL:

```
create temporary table vypil_v_oboch as
select n1.Pijan, v1.Alkohol, n1.Krcma as K1, n2.Krcma as K2
from navstivil n1, vypil v1, navstivil n2, vypil v2
where n1.Idn = v1.Idn and n1.Pijan = n2.Pijan and v1.Alkohol = v2.Alkohol and n2.Idn = v2.Idn

create temporary table niekto_lubi_nevypil_v_oboch as
select l.Alkohol, c1.Krcma as K1, c2.Krcma as K2
from lubi l, capuje c1, capuje c2
where l.Alkohol = c1.Alkohol and l.Alkohol = c2.Alkohol and not exists (
        select *
        from vypil_v_oboch v
        where v.Pijan = l.Pijan and v.Alkohol = c1.Alkohol and
        v.K1 = c1.Krcma and v.K2 = c2.Krcma
)

select c1.Alkohol, c1.Krcma as K1, c2.Krcma as K2 /* answer */
from capuje c1, capuje c2
where c1.Alkohol = c2.Alkohol and not exists (
        select *
        from niekto_lubi_nevypil_v_oboch n
        where n.Alkohol = c1.Alkohol and n.K1 = c1.Krcma and n.K2 = c2.Krcma
)
```

b) Nájdite dvojice [P, A] také, že pijan P buď vypil alkohol A pri aspoň 10 návštevách, alebo navštívil aspoň 20 (navzájom rôznych) krčiem, ktoré A čapujú. Sformulujte tento dotaz v relačnom kalkule **(2)**, Datalogu **(2)** a SQL **(2)**.

Relačný kalkul:

$$\{[P, A]: (\exists N1 \heartsuit N1 = count(I)\ (\exists K\ \exists M\ navstivil(I, P, K) \wedge vypil(I, A, M))) \wedge N1 >= 10) \vee (\exists N2 \heartsuit N2 = count(K)\ (\exists I\ \exists C\ navstivil(I, P, K) \wedge capuje(K, A, C))) \wedge N2 >= 20)\}$$

Datalog:

```
answer(P, A) ←
        subtotal(nv(P, A, I), [P, A], [N1 = count(I)]),
        N1 >= 10.

answer(P, A) ←
        subtotal(nc(P, A, K), [P, A], [N2 = count(K)]),
        N2 >= 20.

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

nc(P, A, K) ←
        navstivil(_, P, K),
        capuje(K, A, _).
```

SQL:

```
select n.Pijan, v.Alkohol
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Pijan, v.Alkohol
having count(distinct n.Idn) >= 10
union
select n.Pijan, c.Alkohol
from navstivil n, capuje c
where n.Krcma = c.Krcma
group by n.Pijan, v.Alkohol
having count(distinct n.Krcma) >= 20
```

2. Uvažujte relácie r(X, Y, U), s(X, Y, V) (bez duplikátov a null hodnôt) a SQL dotaz
select distinct r.X, r.U from r where not exists
(select * from s where s.X = r.X or s.Y = r.Y).
a) Zapíšte ten dotaz ekvivalentne v SQL, bez použitia *or* v klauze *where*. **(2)**

```
select distinct r.X, r.U
from r
where not exists
        (select * from s where s.X = r.X)
        and not exists
        (select * from s where s.Y = r.Y)
```

b) Preložte ten dotaz do relačnej algebry. Dajte si záležať na presnosti. **(2)**

$\Delta\,(\Pi_{X,\,U}\,((r \bowtie (\Pi_X\,(r) - \Pi_X\,(s))) \bowtie (\Pi_Y\,(r) - \Pi_Y\,(s))))$

Alebo:

$\Delta\,(\Pi_{X,\,U}\,(r - \Pi_{r.X,\,r.Y,\,r.U}\,((r \bowtie_{r.X\,=\,s.X} s) \cup (r \bowtie_{r.Y\,=\,s.Y} s))))$

3. Daná je relácia r(A, B, C, D, E, F) s funkčnými závislosťami
AB→CDF, ACF→B, AE→B, CD→ABF, CE→D, CEF→ABD.
a) Nájdite všetky kľúče relácie r. **(1)**

E musí byť v každom kľúči, ale E nie je kľúč. Takže CE je kľúč. AE je tiež kľúč. Iné dvojatribútové kľúče nie sú. Dlhšie kľúče nesmú obsahovať A a C. BDEF nie je nadkľúč, takže ďalšie kľúče nie sú. **Všetky kľúče: AE, CE.**

b) Nájdite minimálne pokrytie funkčných závislostí. **(1)**

Po minimalizácii ľavých strán:
AB→C, AB→D, AB→F, AE→B, CD→A, CD→B, CD→F, CE→D, ACF→B, CE→A, CE→B, CE→D

Po vynechaní redundantných funkčných závislostí dostávame nejaké minimálne pokrytie:
AB→C, AB→D, AE→B, CD→A, CD→F, ACF→B, CE→D

c) Dekomponujte r do tretej normálnej formy, bezstratovo a so zachovaním všetkých funkčných závislostí. **(1)**

3NF z minimálneho pokrytia:
ABD, ABE, ACD, CDF, ABCF, CDE

Alebo:
ABE, ABCDF, CDE

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

Začnime s 3NF z minimálneho pokrytia. Všetky funkčné závislosti v minimálnom pokrytí majú na ľavej strane aspoň 2 atribúty. Takže stačí otestovať, či 4-atribútová ABCF je v BCNF. Je. Platí totiž AB→F a AB→F, ale AB je kľúč v ABCF. Taktiež platí ACF→B, ale ACF je kľúč v ABCF. Iné netriviálne závislosti v ABCF neplatia.

BCNF:
ABD, ABE, ACD, CDF, ABCF, CDE

Alebo:
ABE, ABCDF, CDE

4. Niektoré transakčné systémy používajú okrem read- a write-locks aj tzv. upgrade-lock. Upgrade-lock dáva transakcii, ktorá ho vlastní, iba právo na čítanie zamknutých dát. Avšak transakcia, ktorý vlastní upgrade-lock na nejaký objekt, smie požiadať o write-lock na ten objekt (presnejšie, o zmenu svojho upgrade-lock na write-lock). Okrem bežných pravidiel platia nasledujúce pravidlá pre prideľovanie zámkov. Keď transakcia T1 vlastní write-lock na objekt X, nesmie systém prideliť inej transakcii T2 upgrade-lock na X, a naopak. Keď transakcia T1 vlastní read-lock na objekt X, tak systém smie prideliť inej transakcii T2 upgrade-lock na X (ale nie naopak—t.j. keď T1 vlastní upgrade-lock na X, nesmie systém prideliť inej transakcii T2 read-lock na X).
a) Uveďte príklad rozvrhu dvoch commitovaných transakcií, ktorý je možné generovať striktným dvojfázovým zamykaním s upgrade-locks, ale nie je možné ho generovať striktným dvojfázovým zamykaním bez upgrade-locks (ten rozvrh nemá obsahovať operácie, ktoré sa týkajú zamykania). Vysvetlite. **(2)**

s1, s2, r1(X), r2(X), c2, w1(X), c1

Tento rozvrh sa dá generovať striktným 2PL s upgrade-locks. Tu je jeho rozšírenie o operácie získavania zámkov (zámky sa uvoľňujú automaticky, spolu s commitom):
s1, s2, rl2(X), upgl1(X), r1(X), r2(X), c2, wl1(X), w1(X), c1

Tento rozvrh sa nedá generovať striktným 2PL bez upgrade-locks, lebo transakcia T1 by musela získať write-lock na X už pred r1(X). Lenže v tom prípade by operácia r2(X) mohla byť vykonaná až niekedy po c1.

b) Ako zareaguje scheduler systému, keď od transakcie T2 dostane žiadosť o read-lock na X, pričom iná transakcia T1 v tom momente drží upgrade-lock na X? Vysvetlite. **(2)**

Ak scheduler nepoužíva žiadnu metódu prevencie proti deadlocku, tak odloží vykonanie operácie rl2(X) na neskôr, t.j. žiadosť transakcie T2 o read-lock na X nechá čakať.
(Ak scheduler používa nejakú metódu prevencie proti deadlocku, tak najskôr spustí tú metódu a až potom sa venuje tej žiadosti o read-lock.)