17/1/2013 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) Sformulujte nasledujúci SQL dotaz v relačnej algebre **(1)**
select distinct c.Krcma from capuje c, lubi l where c.Alkohol <> l.Alkohol

$\Delta\ \Pi_{Krcma}\ (P_c(capuje) \bowtie_{c.Alkohol <> l.Alkohol} P_l(lubi))$

b) Rozhodnite, či veta „Krčmy, ktoré čapujú nejaký alkohol, ktorý neľúbi aspoň jeden pijan.“ zodpovedá dotazu z úlohy a). Zdôvodnite. **(1)**

Nie. Napríklad, ak krčma Rumba čapuje len rum a jediný pijan Fero ľúbi rum aj slivovicu, tak podľa tej slovenskej vety Rumba nebude vo výsledku. Lenže vo výsledku dotazu z úlohy a) Rumba bude, lebo joinovacia podmienka je splnená pre slivovicový záznam z relácie ľúbi.

c) Zapíšte nasledujúci SQL dotaz v relačnom kalkule **(2)**:
select distinct c1.Krcma from capuje c1, lubi l where not exists (
  select * from capuje c2 where c2.Cena < 0 and not exists (
    select * from capuje c3 where c1.Alkohol <> l.Alkohol))

$$
\begin{aligned}
&\{K1: \\
&\quad \exists A1\ \exists C1\ \exists P\ \exists A \\
&\quad capuje(K1, A1, C1) \wedge lubi(P, A) \wedge \\
&\quad \neg\ (\\
&\qquad \exists K2\ \exists A2\ \exists C2 \\
&\qquad capuje(K2, A2, C2) \wedge C2 < 0 \wedge \\
&\qquad \neg\ (\\
&\qquad\quad \exists K3\ \exists A3\ \exists C3 \\
&\qquad\quad capuje(K3, A3, C3) \wedge A1 \neq A \\
&\qquad) \\
&\quad) \\
&\}
\end{aligned}
$$

d) Rozhodnite, či dotazy z úloh a) a c) sú ekvivalentné. Odpoveď ÁNO resp. NIE zdôvodnite. **(2)**

Zapíšme dotaz z úlohy a) ako formulu:

$$
\begin{aligned}
&\{K1: \\
&\quad \exists A1\ \exists C1\ \exists P\ \exists A \\
&\quad capuje(K1, A1, C1) \wedge lubi(P, A) \wedge A1 \neq A \\
&\}
\end{aligned}
$$

Poďme teraz zjednodušiť formulu z úlohy c):

$$
\begin{aligned}
&\{K1: \\
&\quad \exists A1\ \exists C1\ \exists P\ \exists A \\
&\quad capuje(K1, A1, C1) \wedge lubi(P, A) \wedge \\
&\quad \neg (\\
&\qquad \mathbf{\exists K2\ \exists A2\ \exists C2} \\
&\qquad \mathbf{capuje(K2, A2, C2) \wedge C2 < 0} \wedge \\
&\qquad \neg (\\
&\qquad\quad \exists K3\ \exists A3\ \exists C3 \\
&\qquad\quad capuje(K3, A3, C3) \wedge A1 \neq A \\
&\qquad) \\
&\quad) \\
&\}
\end{aligned}
$$

Zvýraznená časť formuly je FALSE, lebo (prinajmenšom v tejto databáze) neplatí, že niektorá krčma čapuje nejaký alkohol za zápornú cenu.

Keďže za zvýraznenou časťou formuly nasleduje konjunkcia, celá tá konjunkcia je FALSE:

$$
\begin{aligned}
&\{K1: \\
&\quad \exists A1\ \exists C1\ \exists P\ \exists A \\
&\quad capuje(K1, A1, C1) \wedge lubi(P, A) \wedge \\
&\quad \neg (\\
&\qquad \mathbf{FALSE} \\
&\quad) \\
&\}
\end{aligned}
$$

Negácia FALSE je TRUE, formula sa teda zjednoduší na

$$
\begin{aligned}
&\{K1: \\
&\quad \exists A1\ \exists C1\ \exists P\ \exists A \\
&\quad capuje(K1, A1, C1) \wedge lubi(P, A) \\
&\}
\end{aligned}
$$

Oproti dotazu z úlohy a) chýba joinovacia podmienka A1 ≠ A (tá podmienka je len zdanlivo prítomná vo vnorenom selecte). Tie dva dotazy **nie sú ekvivalentné**. Napríklad, ak krčma Rumba čapuje len rum a jediný pijan Fero ľúbi len rum, výsledkom a) je prázdna množina, zatiaľ čo výsledkom c) je {Rumba}.

2. Ak krčmu navštívi skupina viacerých pijanov, ktorí do krčmy idú spolu, tak sa to dá považovať za jednu hromadnú návštevu krčmy. Avšak aj pri hromadnej návšteve pijani pijú (a platia) každý sám za seba. Databáza z úlohy 1 neumožňuje evidovať hromadné návštevy.
a) Navrhnite vhodnú organizáciu relačnej databázy, ktorá umožňuje pracovať s hromadnými návštevami. (Databázu popíšte spôsobom, ktorý je použitý vo formulácii úlohy 1.) **(2)**

Návštevy sa týkajú len skupín (bánd). V relácii bandy sa pamätá, ktorými pijanmi je ktorá banda tvorená. Pitie sa týka jednotlivých pijanov, teda reláciu vypil treba rozšíriť o atribút Pijan:
capuje(Krcma, Alkohol, Cena)
lubi(Pijan, Alkohol)
navstivil(Idn, Banda, Krcma)
vypil(Idn, Pijan, Alkohol, Mnozstvo)
bandy(Banda, Pijan)

Platia nasledujúce relevantné funkčné závislosti súvisiace so zmenou oproti pôvodnej databáze (je zrejmé, ktoré zanikli):
Idn → Banda, Krčma
Idn, Pijan, Alkohol → Mnozstvo

Alternatívne riešenie je konzervatívne rozšíriť pôvodnú databázu (z úlohy 1). V relácii bandy sa pamätá, ktoré Idn tvoria rovnakú hromadnú návštevu:
capuje(Krcma, Alkohol, Cena)
lubi(Pijan, Alkohol)
navstivil(Idn, Pijan, Krcma)
vypil(Idn, Alkohol, Mnozstvo)
bandy(Banda, Idn)

K pôvodne platným funkčným závislostiam pribudnú Idn → Banda, Banda → Krcma.

Pri návrhu databázy je vhodné minimalizovať redundanciu dát. Pokiaľ nie sú žiadne dodatočné predpoklady na dáta, sú horeuvedené dve riešenia sú zhruba rovnocenné. Ale ak je napríklad časté, že krčmy navštevujú často tie isté skupiny pijanov, tak je rozumnejšie použiť to prvé riešenie, lebo prirodzenejšie reprezentuje štruktúru dát. Rovnaké zloženie bandy je v prvom riešení vyjadrené iba raz, zatiaľ čo v druhom riešení sa opakuje spolu s každou návštevou tej bandy.

b) Sformulujte nad databázou z úlohy a) nasledujúci dotaz v Datalogu **(2)**: Nájdite pijanov, ktorí ľúbia borovičku, ale borovičku pijú len keď do krčmy idú so spoločnosťou (t.j. nie sami).

```
answer(P) ←
        lubi(P, borovicka),
        not vypil_b_sam(P).

vypil_b_sam(P) ←
        vypil(I, P, borovicka, _),
        not spolocnik(I, P).

spolocnik(I, P) ←
        lubi(P, borovicka), /* kvoli formálnej bezpecnosti */
        navstivil(I, B, _),
        bandy(B, P2),
        not P2 = P.
```

c) Sformulujte nad databázou z úlohy a) nasledujúci dotaz v SQL **(2)**: Nájdite trojice [K, C, S] , pre ktoré platí, že C je počet hromadných návštev krčmy K s aspoň 10 pijanmi a S je suma celkových účtov za tieto návštevy.

```
create temporary table velka_banda as
select b.Banda
from bandy b
group by b.Banda
having count(b.Pijan) >= 10

select n.Krcma, count(distinct n.Idn) as C, sum(c.Mnozstvo * c.Cena) as S    /* answer */
from navstivil n, vypil v, capuje c, velka_banda b
where n.Idn = v.Idn and n.Banda = b.Banda and n.Krcma = c.Krcma and v.Alkohol = c.Alkohol
group by n.Krcma
```

3. a) Definujte pojem bezpečnosti Datalogového programu. **(2)**

Datalogový program je bezpečný, ak každé jeho pravidlo je bezpečné.

Pravidlo je bezpečné, ak každá premenná použitá kdekoľvek v tom pravidle je použitá v nejakom nie negovanom relačnom podcieli toho pravidla. *Aritmetické podciele, ktoré narábajú s nekonečnými reláciami, sa v tomto kontexte nepovažujú za relačné podciele.*

b) Uveďte konkrétny príklad Datalogového programu, ktorý nie je bezpečný **(1)** a vysvetlite na tomto príklade negatívne dôsledky, ktoré z toho vyplývajú. **(1)**

universe(X) ← r(Y).

kde r je nejaká extenzionálna databáza (konečná relácia).

Toto pravidlo nie je bezpečné, lebo premenná X je použitá v hlave pravidla, ale nikde inde. Dôsledkom je napríklad, že výsledok dotazu ?- universe(X). nie je možné vypočítať (ak je relácia r neprázdna).

c) Uveďte konkrétny príklad Datalogového programu, ktorý síce nie je formálne bezpečný v zmysle definície z úlohy a), no napriek tomu ho neformálne možno za bezpečný považovať. Zdôvodnite. **(2)**

Formálne bezpečný nie je napríklad nasledujúci program:
answer(K) ← capuje(K1, _), K = K1.

Premenná K sa vyskytuje len v aritmetickom podcieli (relácia rovnosti je nekonečná binárna relácia). Avšak tento program je ekvivalentný programu
answer(K) ← capuje(K, _).
Preto ho môžeme (neformálne) považovať za bezpečný.

4. Uveďte príklad rozvrhu dvoch transakcií (alebo vysvetlite, prečo taký neexistuje), ktorý je:
a) konflikt-sériovateľný, ale nie obnoviteľný **(1)**

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

b) konflikt-sériovateľný, obnoviteľný, ale nie vyhýbajúci sa kaskádovým abortom **(1)**

s1, s2, w1(X), r2(X), w2(Y), c1

c) striktný, ale nie konflikt-sériovateľný **(1)**

s1, s2, r1(X), w2(X), r2(Y), w1(Y), c1, c2

d) nie konflikt-sériovateľný, ale dá sa generovať dvojfázovým zamykaním **(1)**

Dvojfázové zamykanie generuje len konflikt-sériovateľné rozvrhy. Teda požadovaný rozvrh neexistuje.

e) konflikt-sériovateľný, ale nedá sa generovať striktným dvojfázovým zamykaním **(1)**

s1, s2, r1(X), w2(X), c1, c2

5. Pri štarte databázového servera obsahuje log file nasledujúce záznamy, pričom v zázname [I, T, O, X, Y] I je sériové číslo log záznamu, T je identifikátor transakcie, O je identifikátor objektu, X je stará hodnota objektu, Y je nová hodnota objektu: <[4, T1, A, 0, 1], [5, T2, B, 2, 4], [6, T1, COMMIT], [7, T3, A, 1, 8], [8, T4, B, 4, 5], [9, T4, COMMIT]>.
Popíšte sekvenciu akcií, ktoré systém vykoná počas obnovy. **(2)**

Keďže nie sú uvedené žiadne špeciálne predpoklady na systém, budeme uvažovať všeobecný algoritmus obnovy (základnú verziu, bez optimalizácií).

Log file sa prechádza najskôr od konca k začiatku (sprava doľava), vytvárajú sa UNDO a REDO zoznamy, a vykonávajú sa UNDO akcie pre necommitované transakcie:
A := 1 (undo T3)
B := 2 (undo T2)

Po nájdení začiatku log file sa začne opačný prechod, v ktorom sa vykonávajú sa REDO akcie pre commitované transakcie:
A := 1 (redo T1)
B := 5 (redo T4)