11/1/2012 Úvod do databáz, **skúškový** test, max 25 bodov, 90 min
1. Daná je databáza: 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 dotaz v relačnom kalkule **(2)**, Datalogu **(2)**, SQL **(2)**
a relačnej algebre **(2)**: Nájdite dvojice [K, A], o ktorých platí, že krčma K čapuje alkohol A a zároveň v krčme K sa alkohol A vypil v menšom celkovom množstve než v ktorejkoľvek inej krčme, ktorá čapuje alkohol A. (Pozor, v krčme K sa alkohol A možno nikdy nepil.)

Relačný kalkul:

$$
\begin{aligned}
&\{[K, A]: \exists K2\ \exists T2\ \exists T \\
&\quad (\exists C\ capuje(K, A, C)) \wedge \\
&\quad \neg\ /* niekde\_inde\_najviac\_tolko(K, A) */ \\
&\quad (\\
&\qquad /* alkohol\ A\ sa\ v\ K\ vypil\ v\ celkovom\ mnozstve\ T */ \\
&\qquad (\\
&\qquad\quad \blacktriangledown\ I, T = sum(M)\ (\exists P\ navstivil(I, P, K) \wedge vypil(I, A, M)) \\
&\qquad\quad \vee\ /* T = 0, ak\ A\ sa\ v\ K\ nikdy\ nevypil */ \\
&\qquad\quad (T = 0 \wedge (\exists C\ capuje(K, A, C)) \wedge \\
&\qquad\quad \neg\ (\exists I\ \exists P\ \exists M\ navstivil(I, P, K) \wedge vypil(I, A, M))) \\
&\qquad) \\
&\qquad \wedge \\
&\qquad /* alkohol\ A\ sa\ v\ K2\ vypil\ v\ celkovom\ mnozstve\ T2 */ \\
&\qquad (\\
&\qquad\quad \blacktriangledown\ I, T2 = sum(M)\ (\exists P\ navstivil(I, P, K2) \wedge vypil(I, A, M)) \\
&\qquad\quad \vee\ /* T2 = 0, ak\ A\ sa\ v\ K2\ nikdy\ nevypil */ \\
&\qquad\quad (T2 = 0 \wedge (\exists C\ capuje(K2, A, C)) \wedge \\
&\qquad\quad \neg\ (\exists I\ \exists P\ \exists M\ navstivil(I, P, K2) \wedge vypil(I, A, M))) \\
&\qquad) \\
&\qquad \wedge (K2 \neq K) \wedge (T2 <= T) \\
&\quad) \\
&\}
\end{aligned}
$$

Datalog:

```
answer(K, A) ←
        capuje(K, A, _),
        not niekde_inde_najviac_tolko(K, A).

niekde_inde_najviac_tolko(K, A) ←
        total(K, A, T),
        total(K2, A, T2),
        not K2 = K,
        T2 <= T.

total(K, A, T) ←
        subtotal(nv(_, K, A, M), [K, A], [T = sum(M)]).

total(K, A, 0) ←
        capuje(K, A, _),
        not nv2(K, A).

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

nv2(K, A) ←
        nv(_, K, A, _).
```

SQL:

```
create temporary table total as
select n.Krcma, v.Alkohol, sum(v.Mnozstvo) as T
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Krcma, v.Alkohol
union
select c.Krcma, c.Alkohol, 0 as T
from capuje c
where not exists (
        select *
        from navstivil n, vypil v
        where n.Idn = v.Idn and n.Krcma = c.Krcma and v.Alkohol = c.Alkohol)

/* main */
select c.Krcma, c.Alkohol
from capuje c
where not exists (
        select *
        from total t1, total t2
        where c.Krcma = t1.Krcma and c.Alkohol = t1.Alkohol and
        t1.Krcma <> t2.Krcma and t1.Alkohol = t2.Alkohol and t2.T <= t1.T)
```

Relačná algebra:

$$total = \Gamma_{Krcma,\ Alkohol,\ T\ =\ sum(Mnozstvo)}\ (navstivil \bowtie vypil) \cup$$

$$\Pi_{Krcma,\ Alkohol,\ T=0}\ (\Pi_{Krcma,\ Alkohol}\ (capuje) - \Pi_{Krcma,\ Alkohol}\ (navstivil \bowtie vypil))$$

$$answer = \Pi_{Krcma,\ Alkohol}\ (capuje) - \Pi_{Krcma,\ Alkohol}$$

$$(P_{t1}\ (total) \bowtie_{t1.Krcma \neq t2.Krcma \wedge t1.Alkohol = t2.Alkohol \text{ and } t2.T <= t1.T}\ P_{t2}\ (total))$$

b) Sformulujte nasledujúci dotaz v Datalogu **(2)** a v relačnom kalkule **(2)**: Nájdite krčmy, pre ktoré platí: hľadanú krčmu nenavštívil žiaden pijan, ktorý ľúbi všetky alkoholy, ktore tá krčma čapuje. (Každá krčma čapuje nejaký alkohol.)

Datalog:

```
answer(K) ←
        capuje(K, _, _),
        not navstivil_lubi_vsetko(K).

navstivil_lubi_vsetko(K) ←
        navstivil(_, P, K),
        not nelubi_nieco(P, K).

nelubi_nieco(P, K) ←
        capuje(K, A, _),
        navstivil(_, P, K), /* safety */
        not lubi(P, A).
```

Relačný kalkul:

$$
\begin{aligned}
&\{K: \\
&\quad (\exists A\ \exists C\ \text{capuje}(K, A, C)) \wedge \\
&\quad \neg \\
&\quad /* \text{ navstivil\_lubi\_vsetko}(K) */ \\
&\quad (\\
&\qquad \exists P \\
&\qquad (\exists I\ \text{navstivil}(I, P, K)) \wedge \\
&\qquad \neg \\
&\qquad /* \text{ nelubi\_nieco}(P, K) */ \\
&\qquad (\\
&\qquad\quad \exists A\ \exists C \\
&\qquad\quad \text{capuje}(K, A, C) \wedge \neg\ \text{lubi}(P, A) \\
&\qquad) \\
&\quad) \\
&\}
\end{aligned}
$$

2. V SQL databáze sú relácie r(X, Y), s(X, Y), t(X, Y).
Povedzte po slovensky čo najpresnejšie, čo vypočíta nasledujúci SQL dotaz:
select r.X, r.Y from r, s, t where r.X = s.X and r.Y = s.Y group by r.X, r.Y
Zaujímajú nás aj okrajové prípady, t.j. akokoľvek „patologické“ naplnenia databázy. Uveďte konkrétne príklady zaujímavých naplnení relácií *r*, *s*, *t* a vypočítajte pre ne výsledok dotazu. **(2)**

Ak je relácia t neprázdna, tak dotaz vyberie množinu záznamov z r, ktoré sa nachádzajú aj v s. Pritom sa ignorujú (t.j. do výsledku nejdú) záznamy s hodnotou NULL v X alebo v Y.
Ak je relácia t prázdna, tak je výsledkom dotazu prázdna množina.

Nasledujúci príklad zahŕňa snáď všetky zaujímavé prípady. Nech t je neprázdna relácia, napríklad
t = {[0, null]}.
Nech
r = {[0, 0], [0, 0], [0, null], [null, 0], [1, 1], [1, null], [null, 1]},
s = {[0, 0], [0, null], [0, 1]}.
Výsledkom dotazu je množina {[0, 0]}.

3. a) Rozhodnite, či pre všetky relácie typu r(X, Y, W) a s(X, Y, Z) bez duplikátov a NULL hodnôt platí v relačnej algebre rovnosť

$\Delta(\Pi_Y(\sigma_{X=5 \wedge Y=2}(r \bowtie s))) = \Delta(\Pi_Y(\sigma_{X=5}(\Pi_{X,Y}(r)) \bowtie \sigma_{Y=2}(\Pi_{X,Y}(s))))$.
Ak platí, dokážte. Ak nie, uveďte príklad konkrétnych relácií, pre ktoré tá rovnosť neplatí a uveďte výsledky výrazov na ľavej a pravej strane rovnosti. **(3)**

Rovnosť platí. Vyplýva to z riešenia úlohy b).

b) Preložte výrazy na ľavej a pravej strane rovnosti do Datalogu. **(4)**

Ľavá strana:

```
natural_join(RSX, RSY, RW, SZ) ←
        r(RX, RY, RW),
        s(SX, SY, SZ),
        RX = SX,
        RY = SY,
        RSX = RX,
        RSY = RY.

left(Y) ←
        natural_join(RSX, RSY, RW, SZ),
        /* selection */
        RSX = 5,
        RSY = 2,
        /* projection */
        Y = RSY.
```

Dotaz: ?- left(Y).

Tento program je ekvivalentný programu (natural_join nepotrebujeme definovať, stačí dosadiť definíciu do left a následne aplikovať rovnosti premenných):

```
left(2) ←
        r(5, 2, _),
        s(5, 2, _).
```

Pravá strana:

```
proj_r(X, Y) ←
        r(X, Y, _).

proj_s(X, Y) ←
        s(X, Y, _).

sel_r(X, Y) ←
        proj_r(X, Y),
        X = 5.

sel_s(X, Y) ←
        proj_s(X, Y),
        Y = 2.

natural_join(X, Y) ←
        sel_r(X, Y),
        sel_s(X, Y).

right(Y) ←
        natural_join(_, Y).
```

Dotaz: ?- right(Y).

Po postupnom dosadení definícií do pravidla pre right(Y) a aplikácii rovnosti premenných sa program zjednoduší na:

```
right(2) ←
        r(5, 2, _),
        s(5, 2, _).
```

Teda dotazy ?- left(Y) a ?- right(Y) vrátia rovnaký výsledok.

4. Databázový systém spadol. Pred opätovným spustením obsahuje log-file nasledujúce záznamy: <T1, start>, <T1, X, 1, 2>, <T2 start>, <T3, start>, <T3, Y, 3, 2>, <T1, Y, 2, 3>, <T2, Z, 0, 1>, <T2, commit>, <T3, Z, 1, 0>. Záznamy týkajúce sa operácií write obsahujú na prvom mieste starú hodnotu, na druhom mieste novú hodnotu. Popíšte čo najpresnejšie všeobecný algoritmus obnovy (bez checkpointov) **(2)**

Všeobecný algoritmus obnovy prechádza log-file najskôr zostupne. Počas tohto prechodu aktualizuje zoznamy redo_list a undo_list (undo_list aj redo_list sú na začiatku prázdne) a zároveň robí UNDO operácie pre transakcie z undo_list. Keď príde na začiatok log-file, začne vzostupný prechod, pri ktorom vykonáva REDO operácie pre transakcie z redo_list. Keď príde na koniec logfile, začne systém normálnu prevádzku.

a uveďte sekvenciu priradení, ktoré sa vykonajú počas obnovy v tomto konkrétnom prípade. **(2)**

/* Zostupný prechod */
Z := 1 (undo T3)
Y := 2 (undo T1)
Y := 3 (undo T3)
X := 1 (undo T1)
/* Vzostupný prechod */
Z := 1 (redo T2)