19/1/2012 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) Nájdite trojice [K, P1, P2] také, že pijan P1 navštívil krčmu K viackrát než pijan P2 (P2 nemusel krčmu K vôbec navštíviť); a zároveň pijan P1 minul v krčme K viacej peňazí než pijan P2 (P2 nemusel v krčme K minúť žiadne peniaze). Sformulujte tento dotaz v relačnom kalkule **(2)**, Datalogu **(2)**, SQL **(2)** a relačnej algebre **(2)**.

Relačný kalkul:

$$
\begin{aligned}
&\{[K, P1, P2]: \text{/* P1 aj P2 su pijani, K je krcma */}\\
&\quad ((\exists I\ navstivil(I, P1, K)) \vee (\exists A\ lubi(P1, A))) \wedge\\
&\quad ((\exists I\ navstivil(I, P2, K)) \vee (\exists A\ lubi(P2, A))) \wedge (\exists A\ capuje(K, A)) \wedge\\
&\quad (\exists C1\ \exists C2\ C1 > C2 \wedge\\
&\qquad \text{(/* navstevy\_total(K, P1, C1) */}\\
&\qquad\quad (\heartsuit\ C1 = count(I)\ navstivil(I, P1, K)) \vee\\
&\qquad\quad (C1 = 0 \wedge \neg\ (\exists I\ navstivil(I, P1, K))\\
&\qquad) \wedge\\
&\qquad \text{(/* navstevy\_total(K, P2, C2) */}\\
&\qquad\quad (\heartsuit\ C2 = count(I)\ navstivil(I, P2, K)) \vee\\
&\qquad\quad (C2 = 0 \wedge \neg\ (\exists I\ navstivil(I, P2, K))\\
&\qquad)\\
&\quad) \wedge\\
&\quad (\exists T1\ \exists T2\ T1 > T2 \wedge\\
&\qquad (\text{ /* prepil\_total(K, P1, T1) */}\\
&\qquad\quad (\heartsuit\ I, A, T1 = sum(M * C)\\
&\qquad\quad (navstivil(I, P1, K) \wedge vypil(I, A, M) \wedge capuje(K, A, C))) \vee\\
&\qquad\quad (T1 = 0 \wedge \neg\ (\exists I\ \exists A\ \exists M\ navstivil(I, P1, K) \wedge vypil(I, A, M)))\\
&\qquad) \wedge\\
&\qquad (\text{ /* prepil\_total(K, P2, T2) */}\\
&\qquad\quad (\heartsuit\ I, A, T2 = sum(M * C)\\
&\qquad\quad (navstivil(I, P2, K) \wedge vypil(I, A, M) \wedge capuje(K, A, C))) \vee\\
&\qquad\quad (T2 = 0 \wedge \neg\ (\exists I\ \exists A\ \exists M\ navstivil(I, P2, K) \wedge vypil(I, A, M)))\\
&\qquad)\\
&\quad)\\
&\}
\end{aligned}
$$

Datalog:

```
answer(K, P1, P2) ←
        navstevy_total(K, P1, C1),
        navstevy_total(K, P2, C2),
        C1 > C2,
        prepil_total(K, P1, T1),
        prepil_total(K, P2, T2),
        T1 > T2.

pijan(P) ←
        navstivil(_, P, _).

pijan(P) ←
        lubi(P, _).

navstevy_total(K, P, C) ←
        subtotal(navstivil(I, P, K), [K, P], [C = count(I)]).

navstevy_total(K, P, 0) ←
        capuje(K, _, _),  /* safety */
        pijan(P),         /* safety */
        not navstivil(_, P, K).

prepil_total(K, P, T) ←
        subtotal(prepil(_, K, P, _, U), [P, K], [T = sum(U)]).

prepil_total(K, P, 0) ←
        capuje(K, _, _),
        pijan(P),
        not prepil2(K, P).

prepil(I, K, P, A, U) ←
        navstivil(I, P, K),
        vypil(I, A, M),
        capuje(K, A, C),
        U = M * C.        /* U is M * C */

prepil2(K, P) ← prepil(K, P).
```

SQL:

```
create temporary table pijan as
select n.Pijan
from navstivil n
union
select l.Pijan
from lubi l

create temporary table navstevy_total as
select n.Pijan, n.Krcma, count(n.Idn) as C
from navstivil n
group by n.Pijan, n.Krcma
union
select p.Pijan, c.Krcma, 0 as C
from pijan p, capuje c
where not exists (
        select *
        from navstivil n
        where n.Pijan = p.Pijan and n.Krcma = c.Krcma)

create temporary table prepil_total as
select n.Pijan, n.Krcma, sum(v.Mnozstvo * c.Cena) as T
from navstivil n, vypil v, capuje c
where n.Idn = v.Idn and n.Krcma = c.Krcma
group by n.Pijan, n.Krcma
union
select p.Pijan, c.Krcma, 0 as T
from pijan p, capuje c
where not exists (
        select *
        from navstivil n, vypil v
        where n.Idn = v.Idn and n.Pijan = p.Pijan and n.Krcma = c.Krcma)

/* main */
select nt1.Pijan, nt2.Pijan, nt1.Krcma
from navstevy_total nt1, navstevy_total nt2, prepil_total pt1, prepil_total pt2
where nt1.Pijan = pt1.Pijan and nt2.Pijan = pt2.Pijan and
        nt1.Krcma = pt1.Krcma and nt2.Krcma = pt2.Krcma and
        nt1.C > nt2.C and pt1.T > pt2.T
```

Relačná algebra:

$$pijan = \Pi_{Pijan}(navstivil) \cup \Pi_{Pijan}(lubi)$$

$$navstevy\_total = \Gamma_{Pijan,\ Krcma,\ C = count(Idn)}(navstivil) \cup$$
$$\Pi_{Pijan,\ Krcma,\ C = 0}(pijan \times \Pi_{Krcma}(capuje) - \Pi_{Pijan,\ Krcma}(navstivil))$$

$$prepil\_total = \Gamma_{Pijan,\ Krcma,\ T = sum(Mnozstvo * Cena)}(navstivil \bowtie vypil \bowtie capuje) \cup$$
$$\Pi_{Pijan,\ Krcma,\ T = 0}(pijan \times \Pi_{Krcma}(capuje) - \Pi_{Pijan,\ Krcma}(navstivil \bowtie vypil))$$

/* main */

$$answer = \Pi_{Krcma,\ np1.Pijan,\ np2.Pijan}($$
$$P_{np1}(navstevy\_total \bowtie prepil\_total)$$
$$\bowtie_{np1.Pijan = np2.Pijan \wedge np1.Krcma = np2.Krcma \wedge np1.C > np2.C \wedge np1.T > np2.T}$$
$$P_{np2}(navstevy\_total \bowtie prepil\_total))$$

b) Traduje sa, že pri konzumácii jednotkového množstva alkoholu Dryer vypije pijan pri rovnakej návšteve aspoň dve pivá. Nájdite dvojice [P, K], také, že pijan P aspoň pri jednej návšteve krčmy K túto tradíciu porušil. Sformulujte tento dotaz v Datalogu **(2)** a v relačnom kalkule **(2)**.

Formulácia v prirodzenom jazyku je možno nie celkom jednoznačná. Predpokladáme, že tá tradícia hovorí, že spolu s každým vypitým Dryerom musí pijan pri tej istej návšteve vypiť aspoň dve (rôzne) pivá. Inak povedané, množstvo piva vypitého pri návšteve, kde sa pije Dryer, musí byť aspoň dvojnásobkom množstva vypitého Dryera.

Datalog:

```
answer(P, K) ←
        navstivil(I, P, K),
        vypil(I, dryer, MD),
        not vypil_dost_piv(I, MD).

vypil_dost_piv(I, MD) ←
        vypil(I, dryer, MD), /* safety */
        vypil(I, pivo, MP),
        MD2 = 2 * MD, /* MD2 is 2 * MD */
        MP >= MD2.
```

Relačný kalkul:

```
{[P, K]:
        (∃I ∃MD
                navstivil(I, P, K) ∧ vypil(I, dryer, MD) ∧
                ¬
                (∃MP /* vypil_dost_piv */
                        vypil(I, pivo, MP) ∧ MP >= 2 * MD
                )
        )
}
```

2. Predpokladajte, že relácie r(A, B), s(X, Y, Z) neobsahujú duplikáty a NULL hodnoty.
a) Napíšte po slovensky (čo najpresnejšie), čo je výsledkom dotazu
select r.A, r.B, s.Z from r left outer join s on (r.A = s.X and r.B = s.Y) **(2)**

Výsledkom je množina
$\{[A, B, Z]: r(A, B) \wedge s(A, B, Z)\} \cup \{[A, B, null]: r(A, B) \wedge \neg (\exists Z\ s(A, B, Z))\}$

Po slovensky:
Výsledkom je projekcia spojenia (podľa podmienky on) relácií r, s na atribúty A, B, Z; zjednotená s trojicami [A, B, null] takými, že dvojica [A, B] patrí do r, ale nespája sa podľa podmienky on so žiadnym záznamom v s.

b) Zapíšte dotaz z úlohy a) v SQL bez použitia kľúčového slova „join". **(2)**

```
select r.A, r.B, s.Z
from r, s
where r.A = s.X and r.B = s.Y
union
select r.A, r.B, null as Z
from r
where not exists (
        select *
        from s
        where r.A = s.X and r.B = s.Y)
```

3. a) Definujte bezpečnosť Datalogových programov. **(2)**

Pravidlo programu je bezpečné keď každá premenná vyskytujúca sa kdekoľvek v pravidle platí, že tá premenná sa vyskytuje aj v niektorom nie negovanom relačnom podcieli v tele toho pravidla (alebo je viazaná reláciou rovnosti s niektorou premennou, ktorá sa vyskytuje v niektorom nie negovanom relačnom podcieli v tele toho pravidla).

Program je bezpečný keď každé jeho pravidlo je bezpečné. *(Program je bezpečný aj vtedy, keď je ekvivalentný niektorému bezpečnému programu v zmysle predošlého odstavca. Lenže ekvivalencia programov je vo všeobecnosti algoritmicky nerozhodnuteľný problém. Preto ak konkrétny program nie je bezpečný v zmysle toho prvého odstavca, je rozumné ho nepovažovať za bezpečný, kým nie je známy dôkaz ekvivalencie s niektorým programom, ktorý je bezpečný v zmysle toho prvého odstavca.)*

b) Prečo je rozumné vyžadovať od Datalogových programov, aby boli bezpečné? **(1)**

Dôvodom je, že chceme, aby sa výsledok akéhokoľvek dotazu na Datalogový program dal algoritmicky vypočítať (za predpokladu konečnosti extenzionálnej databázy). Toto je garantované pre bezpečné nerekurzívne programy. *Túto garanciu je pri dôslednom výbere a aplikácii matematickej teórie databáz možné rozšíriť aj na rekurzívne programy.*

(Veľmi pragmatickým dôvodom je, že programy, ktoré nie sú bezpečné, sa nedajú vyjadriť v SQL.)

c) Rozhodnite o každom z programov P1, P2, P3 (relácia a je extenzionálna databáza), či je bezpečný v zmysle Vašej definície z úlohy a). V prípade, že nie, zdôvodnite prečo nie. **(3)**

P1: $q(X) \leftarrow a(X, Y), \text{not } p(Y, Y).\ p(Y, Y) \leftarrow a(Y, Y), \text{not } a(Y, 3).$

Toto je bezpečný program. X aj Y sú v prvom pravidle v relačnom podcieli a(X, Y). Premenná Y je v druhom pravidle relačnom podcieli a(Y, Y).

P2: $q(X) \leftarrow a(\_, Y), \text{not } p(X, Y).\ p(Y, Y) \leftarrow a(Y, 3), \text{not } a(Y, Y).$

Toto nie je bezpečný program, lebo v prvom pravidle sa premenná X nachádza len v hlave pravidla a v negovanom podcieli not p(X, Y).

P3: $q(X) \leftarrow a(X, Y), \text{not } p(X, Y).\ p(X, Y) \leftarrow X = Y, Y > 3, Y < 7.$

Toto nie je bezpečný program, lebo v druhom pravidle sa premenné X aj Y nevyskytujú v žiadnom pozitívnom relačnom podcieli (sú si rovné a vyskytujú sa len v aritmetických podcieľoch).

Dotaz ?- p(X, Y) nie je algoritmicky vypočítateľný. Vo výsledku je zrejme p(4, 4), p(5, 5), p(6, 6), ale napríklad aj p(4.8752726, 4.8752726), p(π, π) a podobne.

Za zmienku stojí, že výsledok dotazu ?- p(X, Y) je napriek tomu algoritmicky vypočítateľný (predpokladáme, že extenzionálna databáza je konečná). Vo výsledku sú dvojice [X, Y], pre ktoré platí $a(X, Y) \wedge \neg (X = Y \wedge X > 3 \wedge X < 7)$.
Kandidátmi na výsledok sú teda len dvojice [X, Y], pre ktoré platí a(X, Y). Tých je konečne veľa. Pre ***konkrétne*** *X a Y je jednoduché otestovať, či platí*
$X = Y \wedge X > 3 \wedge X < 7$ *alebo nie (do výsledku sa dostanú len tie kandidátske dvojice [X, Y], pre ktoré táto podmienka neplatí).*

4. Vysvetlite ako funguje merge-sort (operátor fyzickej algebry, ktorý sa používa na triedenie relácií). Zamerajte sa najmä na popis organizácie operačnej pamäte. **(2)**

Úlohou externého triedenia, akým je merge-sort, je utriediť reláciu, ktorá je uložená na trvácnom médiu (napr. disk alebo páska), povedzme v sekvenčnom súbore. Merge-sort dostane k dispozícii M blokov v RAM. Výsledkom je sekvenčný súbor, v ktorom sú záznamy usporiadané podľa daného triediaceho kritéria. Pre jednoduchosť predpokladajme, že bloky v súbore a bloky v RAM sú rovnako veľké. Predpokladajme, že súbor treba utriediť vzostupne.

V prvej fáze merge-sort sa všetkých M blokov v RAM naplní blokmi vstupného súboru. Záznamy v RAM sa utriedia (napr. quicksortom) a zapíšu do sekvenčného súboru, tzv. behu, o veľkosti M blokov. Potom sa do všetkých M blokov v RAM prečíta nasledujúcich M blokov vstupného súboru. Záznamy v RAM sa utriedia a zapíšu do druhého behu. Atď.

V druhej fáze sa behy zlučujú do výsledného súboru (resp. do dlhšieho behu, v zmysle nasledujúceho odstavca). Pritom 1 blok RAM sa rezervuje pre výstup, zvyšných M-1 blokov sa rezervuje pre bloky z behov (1 blok pre 1 beh). Z každého behu sa prečíta 1 blok. Spomedzi všetkých prvých záznamov vo vstupných M-1 blokoch sa nájde najmenší, uloží sa do výstupného bloku a označí sa ako spracovaný (vo vstupnom aj vo výstupnom bloku). Opäť sa hľadá minimum zo všetkých prvých záznamov vstupných blokov atď. Keď sa výstupný blok zaplní, zapíše sa do výsledného súboru. Keď sú všetky záznamy v niektorom vstupnom bloku spracované, prečíta sa do toho bloku nasledujúci blok z príslušného behu. Druhá fáza končí, keď sú všetky záznamy všetkých vstupných behov spracované a výstupný blok je zapísaný do súboru. Vtedy sa všetky spracované vstupné behy odstránia z disku (resp. z pásky).

Ak je počet behov vytvorených v prvej fáze väčší ako M-1, treba druhú fázu opakovať (možno niekoľkokrát). Produktom každej iterácie druhej fázy sú dlhšie behy než v predošlej fáze. Tieto dlhšie behy sú vstupom do nasledujúcej iterácie druhej fázy. Keď ostane už len jeden beh, opakovanie druhej fázy skončí. *(Pri súčasných veľkostiach databáz a pamäťových médií stačí obvykle jedna iterácia druhej fázy.)*

Čo je cieľom optimalizácie pri implementácii tohto algoritmu? **(1)**

Cieľom je minimalizovať počet diskových operácií. Diskové operácie sú čítanie bloku zo súboru do RAM a zápis bloku z RAM do súboru. (Tieto operácie sú oveľa drahšie než operácia porovnávania záznamov.)