11/1/2013 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 SQL **(2)**, relačnej algebre **(2)** a Datalogu **(2)**: Nájdite dvojice [A, M] také, že M je mediánom ceny alkoholu A cez všetky krčmy, ktoré alkohol A čapujú.

Datalog:

```
answer(A, M) ←
        subtotal(kandidat(A, C), [A], [M = avg(C)]).

kandidat(A, C) ←
        subtotal(capuje(K, A, _), [A], [T = count(K)]),
        subtotal(mensi_rovny(A, C, K2), [A, C], [MR = count(K2)]),
        subtotal(vacsi_rovny(A, C, K2). [A, C], [VR = count(K2)]),
        2 * MR >= T,
        2 * VR >= T.

mensi_rovny(A, C, K2) ←
        capuje(_, A, C),
        capuje(K2, A, C2),
        C2 <= C.

vacsi_rovny(A, C, K2) ←
        capuje(_, A, C),
        capuje(K2, A, C2),
        C2 >= C.
```

Poznámka. Nie je nutné priemerovať hodnoty kandidátov. Pre daný alkohol existujú najviac 2 kandidátske hodnoty C, stačí spriemerovať tie. Ak je kandidát len jeden, tak je mediánom:

```
answer(A, M) ←
        kandidat(A, C1),
        kandidat(A, C2),
        not C1 = C2,
        M = (C1 + C2) / 2.

answer(A, M) ←
        kandidat(A, M),
        not  iny_kandidat(A, M).

iny_kandidat(A, C) ←
        kandidat(A, C),
        kandidat(A, C2),
        not C = C2.
```

Alternatívne riešenie.
Uvažujme neusporiadanú postupnosť čísiel C_i, i=1..N, z ktorej vyberáme medián. Pre jednoduchosť najskôr predpokladajme, že čísla v postupnosti sú navzájom rôzne. Každému prvku postupnosti priradíme rank. Rank je číslo 1..N, ktoré hovorí, na ktorej pozícii sa prvok nachádza v usporiadanej postupnosti. Inak povedané, rank prvku je počet prvkov, ktoré sú pred tým rankovaným prvkom v danom usporiadaní (presnejšie, sú menšie alebo rovné). Ak N je nepárne, tak medián je prvok s rankom (N-1)/2. Ak N je párne, tak medián je priemer prvkov s rankami N/2 a N/2+1.

Uvažujme teraz prípad, že čísla v postupnosti (v tomto konkrétnom prípade hodnoty atribútu Cena) nemusia byť navzájom rôzne. Aby sme rovnakým číslam priradili rôzne ranky, tak umelo rozšírime čísla o nejakú jednoznačnú (kľúčovú) hodnotu, na ktorej je definované úplné usporiadanie. V tomto konkrétnom prípade sa ponúka hodnota atribútu Krcma. Pôvodné usporiadanie podľa hodnoty atribútu Cena zmeníme na lexikografické usporiadanie hodnôt [Cena, Krcma]. Keďže všetky dvojice [Cena, Krcma] sú navzájom rôzne, môžeme použiť postup z predošlého odstavca.

```
answer(A, C) ←
    subtotal(capuje(K, _, _), [], [T = count(K)]),
    1 is T mod 2, /* odd(T) */
    R is (T + 1) / 2,
    subtotal(mensi_rovny(K1, A, C, K2), [K1, A, C], [R = count(K2)]).

answer(A, M) ←
    subtotal(capuje(K, _, _), [], [T = count(K)]),
    0 is T mod 2, /* not odd(T) */
    R1 is T / 2,
    R2 is T / 2 + 1,
    subtotal(mensi_rovny(K1, A, C1, K3), [K1, A, C1], [R1 = count(K3)]),
    subtotal(mensi_rovny(K2, A, C2, K3), [K2, A, C2], [R2 = count(K3)]),
    M is (C1 + C2) / 2.

mensi_rovny(K1, A, C, K2) ←
    capuje(K1, A, C),
    capuje(K2, A, C2),
    C2 < C.

mensi_rovny(K1, A, C, K2) ←
    capuje(K1, A, C),
    capuje(K2, A, C),
    K2 =< K1.
```

SQL (preklad pôvodného Datalogového programu):

```
create temporary table mensi_rovny as
select c1.Alkohol, c1.Cena, c2.Krcma
from capuje c1, capuje c2
where c1.Alkohol = c2.Alkohol and c2.Cena <= c1.Cena

create temporary table vacsi_rovny as
select c1.Alkohol, c1.Cena, c2.Krcma
from capuje c1, capuje c2
where c1.Alkohol = c2.Alkohol and c2.Cena >= c1.Cena

create temporary table kandidat as
select sub1.Alkohol, sub2.Cena
from
        (
                select c.Alkohol, count(c.Krcma) as T
                from capuje c
                group by c.Alkohol, c.Cena
        ) sub1,
        (
                select mr.Alkohol, count(mr.Krcma) as MR
                from mensi_rovny mr
                group by mr.Alkohol, mr.Cena
        ) sub2,
        (
                select vr.Alkohol, count(vr.Krcma) as VR
                from vacsi_rovny vr
                group by vr.Alkohol, vr.Cena
        ) sub3
where sub1.Alkohol = sub2.Alkohol and sub1.Alkohol = sub3.Alkohol and
  sub2.Cena = sub3.Cena and 2 * sub2.MR >= sub1.T and 2 * sub3.VR >= sub1.T

select k.Alkohol, avg(k.Cena) as Median /* answer */
from kandidat k
group by k.Alkohol
```

Relačná algebra (preklad pôvodného Datalogového programu):

$$\text{mensi\_rovny} := \Pi_{c1.Alkohol,\ c1.Cena,\ c2.Krcma} \left(P_{c1}(\text{capuje}) \bowtie_{c1.Alkohol = c2.Alkohol \text{ and } c2.Cena <= c1.Cena} P_{c2}(\text{capuje})\right)$$

$$\text{vacsi\_rovny} := \Pi_{c1.Alkohol,\ c1.Cena,\ c2.Krcma} \left(P_{c1}(\text{capuje}) \bowtie_{c1.Alkohol = c2.Alkohol \text{ and } c2.Cena >= c1.Cena} P_{c2}(\text{capuje})\right)$$

$$\text{kandidat} := \Pi_{sub1.Alkohol,\ sub2.Cena} \left(\sigma_{sub2.Cena = sub3.Cena \text{ and } 2 * sub2.MR >= sub1.T \text{ and } 2 * sub3.VR >= sub1.T} \right.$$
$$\left(P_{sub1} \left(\Gamma_{Alkohol,\ T = count(Krcma)} (\text{capuje})\right) \bowtie \right.$$
$$P_{sub2} \left(\Gamma_{Alkohol,\ Cena,\ MR = count(Krcma)} (\text{mensi\_rovny})\right) \bowtie$$
$$\left.\left. P_{sub3} \left(\Gamma_{Alkohol,\ Cena,\ VR = count(Krcma)} (\text{vacsi\_rovny})\right)\right)\right)$$

$$\text{answer} = \Gamma_{Alkohol,\ Median = avg(Cena)} (\text{kandidat})$$

b) Sformulujte nasledujúci dotaz v Datalogu **(2)**, v relačnom kalkule **(2)** a SQL **(2)**: Nájdite dvojice [K, A] také, že krčma K čapuje alkohol A; a zároveň žiaden pijan, ktorý vypil alkohol A v krčme K, nenavštívil inú krčmu, ktorá alkohol A čapuje lacnejšie.

Relačný kalkul:

$$
\begin{aligned}
&\{[K, A]: (\exists C\ capuje(K, A, C)) \wedge \neg \\
&\quad (/* vypil\_navstivil\_lacnejsiu(K, A) */ \\
&\qquad \exists I\ \exists P\ \exists C\ \exists M\ \exists I2\ \exists K2\ \exists C2 \\
&\qquad navstivil(I, P, K) \wedge \\
&\qquad capuje(K, A, C) \wedge \\
&\qquad vypil(I, A, M) \wedge \\
&\qquad navstivil(I2, P, K2) \wedge \\
&\qquad capuje(K2, A, C2) \wedge \\
&\qquad C2 < C \\
&\quad) \\
&\}
\end{aligned}
$$

Datalog:

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

vypil_navstivil_lacnejsiu(K, A) ←
        navstivil(I, P, K),
        capuje(K, A, C),
        vypil(I, A, _),
        navstivil(_, P, K2),
        capuje(K2, A, C2),
        C2 < C.
```

SQL:

```
create temporary table vypil_navstivil_lacnejsiu as
select n1.Krcma, c1.Alkohol
from navstivil n1, capuje c1, vypil v1, navstivil n2, capuje c2
where n1.Idn = v1.Idn and n1.Pijan = n2.Pijan and n1.Krcma = c1.Krcma and
  c1.Alkohol = v1.Alkohol and c1.Alkohol = c2.Alkohol and c1.Cena > c2.Cena and
  n2.Krcma = c2.Krcma

select c.Krcma, c.Alkohol /* answer */
from capuje c
where not exists (
        select *
        from vypil_navstivil_lacnejsiu vnl
        where c.Krcma = vnl.Krcma and c.Alkohol = vnl.Alkohol)
```

2. a) Definujte bezstratovosť spojenia dekompozície relačnej schémy. **(1)**

Dekompozícia relácie r do relácií $r_1, r_2, ..., r_N$ je bezstratová (spája sa bezstratovo), ak pre každé naplnenie (populáciu) relácie r platí

$r = \Pi_{r1} (r) \bowtie \Pi_{r2} (r) \bowtie ... \bowtie \Pi_{rN} (r)$

b) Uvažujte reláciu r(A, B, C, D, E) s funkčnými závislosťami A→CE, AD→B, BC→D, BE→A. Rozhodnite, či sa dekompozícia r1(A, C, D, E), r2(A, B, E) spája bezstratovo. Ak áno, zdôvodnite. Ak nie, uveďte príklad konkrétnych naplnení relácií r, r1 a r2, ktoré je v rozpore s definíciou z úlohy a). **(2)**

Keďže ide o dekompozíciu do dvoch relácií, stačí použiť jednoduchý test. Spoločné atribúty r1 a r2 sú A a E. Uvedená dekompozícia sa spája bezstratovo, ak AE je nadkľúčom buď v r1 alebo v r2.

$\{AE\}^* = \{ACE\}$, teda AE nie je nadkľúčom ani v r1 ani v r2. Teda táto dekompozícia sa nespája bezstratovo.

Poďme skonštruovať konkrétny kontrapríklad. ABE je zjavne nadkľúč v r2. Keby atribút B bol aj v relácii r1, dekompozícia by bola bezstratová. Skúsme dať atribútom A, E rovnakú hodnotu a atribútu B rôzne hodnoty (samozrejme tak, aby boli splnené všetky funkčné závislosti):

r:

A	**B**	**C**	**D**	**E**
0	0	1	0	0
0	1	1	1	0

$\Pi_{r1} (r)$:

A	**C**	**D**	**E**
0	1	0	0
0	1	1	0

$\Pi_{r2} (r)$:

A	**B**	**E**
0	0	0
0	1	0

$\Pi_{r1} (r) \bowtie \Pi_{r2} (r)$:

A	**B**	**C**	**D**	**E**
0	0	1	0	0
0	1	1	0	0
0	0	1	1	0
0	1	1	1	0

$\Pi_{r1} (r) \bowtie \Pi_{r2} (r) \neq r$ pre toto naplnenie r.

c) Uveďte príklad dekompozície relačnej schémy r(A, B, C, D, E) s funkčnými závislosťami A→CE, AD→B, BC→D, BE→A, ktorá sa spája bezstratovo, je v tretej normálnej forme a zároveň zachováva všetky funkčné závislosti. **(2)**

Minimálne pokrytie množiny funkčných závislostí:
A→C, A→E, AD→B, BC→D, BE→A

Dekompozíciu s požadovanými vlastnosťami skonštruujeme z tohto minimálneho pokrytia. Treba ešte overiť, že niektorá z generovaných relácií je nadkľúčom v r (ak nie, treba do dekompozície pridať reláciu s nejakým kľúčom): **r1(A, C), r2(A, B, D), r3(B, C, D), r4(A, B, E)**.
({ABE} je nadkľúčom v r.)

3. V systéme bežia nasledujúce dve transakcie (a žiadne iné):
T1: start, r(X), r(Y), w(X), commit. T2: start, r(X), r(Y), w(Y), commit.
a) Rozšírte transakcie o operácie rl (read-lock), wl (write-lock) a ul (unlock) tak, aby boli konformné s dvojfázovým zamykacím protokolom. Maximalizujte stupeň paralelizácie. **(1)**

Zámky treba získať čím neskôr a uvoľniť čím skôr.
T1: start, wl(X), r(X), rl(Y), r(Y), ul(Y), w(X), ul(X), commit.
T2: start, rl(X), r(X), wl(Y), ul(X), r(Y), w(Y), ul(Y), commit.

b) Rozhodnite, či existuje rozvrh, v ktorom T1 a T2, rozšírené o operácie z úlohy a), končia v deadlocku. Svoju odpoveď ÁNO resp. NIE zdôvodnite. **(1)** (Buď uveďte príklad takého rozvrhu alebo dokážte, že taký rozvrh neexistuje.)

Ak T1 získa zámok na X ako prvá, tak rozvrh bude sériový. Deadlock vzniknúť nemôže.

Ak zámok na X získa najskôr T2, tak sa vykoná časť T2 až po operáciu ul(X). Neskôr však už deadlock vzniknúť nemôže, lebo transakcia, ktorá ako prvá získa zámok na Y, dobehne až do konca.

Teda **neexistuje rozvrh, v ktorom T1 a T2 končia v deadlocku.**

c) Vysvetlite, ako funguje metóda wait-or-die na prevenciu deadlocku. **(2)**

Systém pridelí každej transakcii časovú pečiatku v čase vykonávania jej operácie start.
Ak nejaká transakcia T žiada o zámok, ktorý v tom momente drží transakcia T', tak systém najskôr zistí, ktorá z tých transakcií je staršia (t.j. má menšiu časovú pečiatku). Ak je T staršia než T', tak systém nechá T čakať na pridelenie zámku. Inak systém abortuje T.

d) Uveďte príklad rozvrhu horeuvedených transakcií, v ktorom transakcia T1 bude abortovaná pri použití metódy wait-or-die. **(1)**

start2, rl(X), start1, ~~wl1(X),~~ a1

4. Uvažujte SQL dotaz select r.X, s.X from r, s where r.X = s.X and r.Y > 2 * s.Y. Relácia r je uložená v 500 diskových blokoch, relácia s je uložená v 1000 diskových blokoch. Bloky na disku a v operačnej pamäti sú rovnako veľké. Dotaz sa počíta metódou nested-loop-join, k dispozícii je 900 voľných blokov v operačnej pamäti.
a) Popíšte spôsob organizácie operačnej pamäte. **(2)**

Pre výpočet joinu sa z voľných 900 blokov rezervuje celkovo 502 blokov:
500 blokov pre čítanie relácie r (lebo r je menšia než relácia s), 1 blok pre postupné čítanie blokov relácie s, 1 blok pre zápis joinovaných n-tíc.

b) Uveďte počet vstupných diskových operácií. **(1)**

Každý blok relácie r aj relácie s sa prečíta práve raz. To je celkovo 1500 **diskových operácií**.