2/2/2012 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 pijanov, ktorí sú v niektorej krčme lokálnymi šampiónmi v pití rumu (t.j. hľadaný pijan v aspoň jednej krčme vypil viacej rumu než ľubovoľný iný pijan).
Sformulujte v relačnom kalkule **(2)**, Datalogu **(2)**, SQL **(2)** a relačnej algebre **(2)**.

Relačný kalkul:

```
{P:
 (∃K ∃I
        navstivil(I, P, K) ∧
        ¬ /*iny sampion */
        (∃T ∃P2 ∃T2 /*iny sampion */
                ♥ I, T = sum(M) (navstivil(I, P, K) ∧ vypil(I, rum, M)) ∧
                ♥ I, T2 = sum(M) (navstivil(I, P2, K) ∧ vypil(I, rum, M))
                ∧ T <= T2 ∧ P ≠ P2
        )
)
}
```

Datalog:

```
answer(P) ←
        navstivil(_, P, K),
        not iny_sampion(P, K).

iny_sampion(P, K) ←
        subtotal(nv(_, P, K, M), [P, K], [T = sum(M)]),
        subtotal(nv(_, P2, K, M), [P2, K], [T2 = sum(M)]),
        T <= T2,
        not P = P2.

nv(I, P, K, M) ←
        navstivil(I, P, K),
        vypil(I, rum, M).
```

SQL:

```
create temporary table skore as
select n.Pijan, n.Krcma, sum(v.Mnozstvo) as T
from navstivil n, vypil v
where n.Idn = v.Idn and v.Alkohol = ‚rum‘
group by n.Pijan, n.Krcma

/* main */
select n.Pijan
from skore s
where not exists (
        select *
        from skore s2
        where s.Krcma = s2.Krcma and s.T <= s2.T and s.Pijan <> s2.Pijan)
```

Relačná algebra:

skore = $\Gamma_{\text{Pijan, Krcma, T = sum(Mnozstvo)}}$ (navstivil $\bowtie$ $\sigma_{\text{Alkohol = ‚rum‘}}$ (vypil))

/* main */

Π_{Pijan} (skore) -

$\Pi_{\text{s1.Pijan}}$ (P_{s1} (skore) $\bowtie_{\text{s1.Krcma = s2.Krcma and s1.T <= s2.T and s1.Pijan <> s2.Pijan}}$ P_{s2} (skore))

b) Zapíšte v slovenskom jazyku (presne a čo najjednoduchšie) **(2)** a Datalogu **(2)**:
{[K1, K2]: (∃A ∃C1 capuje(K1, A, C1)) ∧ (∃A ∃C2 capuje(K2, A, C2)) ∧
∀A ∀C1 ∃C2
(capuje(K1, A, C1) ∧ capuje(K2, A, C2) ∧ (C1 <= C2)) ∨ (¬ capuje(K1, A, C1))}

Slovenský jazyk:
Usporiadané dvojice krčiem [K1, K2] pre ktoré platí: každý alkohol, ktorý sa čapuje v K1, sa v K2 čapuje tiež, ale nie lacnejšie než v K1.

Datalog:

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

nieco_necapuje_alebo_lacnejsie(K1, K2) ←
        capuje(K1, A, _),
        not c(K2, A).

nieco_necapuje_alebo_lacnejsie(K1, K2) ←
        capuje(K1, A, C1),
        capuje(K2, A, C2),
        C2 < C1.

c(K, A) ←
        capuje(K, A, _).
```

2. Uvažujte reláciu r(X, Y, Z, W) a množinu F funkčných závislostí platných v r.
a) Vysvetlite čo najpresnejšie (definujte), čo znamená, že F obsahuje funkčnú závislosť XY → ZW. **(1)**

V r platí funkčná závislosť XY→ ZW, keď
$\forall Z1\ \forall W1\ \forall Z2\ \forall W2$
$((r(X, Y, Z1, W1) \wedge r(X, Y, Z2, W2)) \Rightarrow (Z1 = Z2 \wedge W1 = W2))$
pre všetky inštancie relácie r.

b) Vysvetlite čo najpresnejšie (definujte), čo znamená, že XYZ je nadkľúčom v r. **(1)**

XYZ je nadkľúčom v r práve vtedy, keď $W \in \{X, Y, Z\}^+$.

c) Vysvetlite čo najpresnejšie (definujte), čo znamená, že XYZ je kľúčom v r. **(1)**

XYZ je kľúčom v r práve vtedy, keď XYZ je nadkľúčom v r a zároveň XYZ je minimálnym nadkľúčom v zmysle množinovej inklúzie (t.j. žiadna vlastná podmnožina {X, Y, Z} nie je nadkľúčom v r).

d) Vysvetlite čo najpresnejšie (definujte) čo znamená, že rozklad r do relácií r1(X, Y, Z), r2(X, W) je bezstratový (t.j. že r1(X, Y, Z) a r2(X, W) sa spájajú bezstratovo). **(1)**

Rozklad r(X, Y, Z, W) do r1(X, Y, Z) a r2(X, W) sa spája bezstratovo práve vtedy, keď
$r(X, Y, Z, W) = \Pi_{X, Y, Z}(r) \bowtie \Pi_{X, W}(r)$
pre všetky inštancie relácie r.

e) Napíšte algoritmus na overenie bezstratovosti rozkladu r do relácií r1(X, Y, Z) a r2(X, W). **(1)**

X je jediným spoločným atribútom r1(X, Y, Z) a r2(X, W). Ak je X nadkľúčom buď v r1 alebo v r2, tak rozklad r do r1 a r2 je bezstratový, inak je stratový:

if (X → XYZ) or (X → XW) then
    rozklad r(X, Y, Z, W) do r1(X, Y, Z) a r2(X, W) sa spája bezstratovo
else
    rozklad r(X, Y, Z, W) do r1(X, Y, Z) a r2(X, W) sa nespája bezstratovo

3. Dané sú relácie r(X, Y, Z), s(Z) bez duplikátov a NULL hodnôt. Relácia d je definovaná v relačnej algebre takto: $d = \Delta(\Pi_{X,Y}(r)) - \Pi_{X,Y}(\Delta(\Pi_{X,Y}(r) \times s) - r)$.
a) Zapíšte d v relačnom kalkule (relácie interpretujte ako predikáty). **(2)**

$$
\begin{aligned}
&\{[X, Y]: \\
&(\exists Z1 \\
&\quad r(X, Y, Z1) \wedge \\
&\quad \neg \\
&\quad (\exists Z2\ \exists Z3 \\
&\qquad r(X, Y, Z2) \wedge s(Z3) \wedge \neg\, r(X, Y, Z3) \\
&\quad) \\
&) \\
&\}
\end{aligned}
$$

b) Zapíšte d v Datalogu (relácie interpretujte ako predikáty). **(2)**

```
d(X, Y) ←
        r(X, Y, _),
        not second(X, Y).

second(X, Y) ←
        r(X, Y, _),
        s(Z),
        not r(X, Y, Z).
```

c) Vypočítajte reláciu d pre databázu
r(X, Y, Z) = {[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 0, 1], [1, 1, 1]},
s(Z) = {[0], [1]}. **(1)**

Vo výsledku, teda v relácii d, sú také [X, Y], ktoré sa v r vyskytujú vo všetkých kombináciách s hodnotami atribútu Z relácie s. Pre danú databázu je výsledkom **{[0, 0], [1, 0]}**.

Relácia d je operátor, v literatúre nazývaný division (÷). Niektorí autori ho radia medzi „štandardné“ operátory relačnej algebry, hoci sa dá definovať pomocou jednoduchších operátorov (ako je vidieť z tejto úlohy).

4. Relácia z obsahuje 4 000 000 záznamov. Jeden záznam zaberá 200 Bytov. Veľkosť bloku (na disku aj v RAM) je 4kB. Jedna I/O operácia trvá 10 ms. Odhadnite čas potrebný na vykonanie dotazu *select Meno from z where Telefon='0042126029501'* pre nasledujúce spôsoby indexovania relácie *z* na atribúte *Telefon*:

Počet stránok z je $\lceil 4\ 000\ 000 / \lfloor (4096 / 200) \rfloor \rceil = 200\ 000$. Budeme predpokladať, že daný dotaz vracia 1 záznam (čas výstupnej operácie zanedbáme).

a) žiadny index;

Treba prečítať všetky bloky: 200 000 * 10 ms = 2000 000 ms = 2 000 s. To je zhruba **33 minút**.

b) lineárny hashovaný index bez blokov preplnenia (overflow pages);

Vypočítame hash telefónneho čísla 0042126029501. Pre nájdenie výsledného záznamu stačí 1 disková operácia: **10 ms**. V prípade hashovania s adresárom by sme čítali z disku dvakrát, ale na tom príliš nezáleží, dotaz pobeží **rádovo desiatky ms**.

c) B^+ tree index.

Nevieme, koľko miesta zaberajú smerníky uložené v uzloch B^+ stromu, koľko miesta v nich zaberajú telefónne čísla, ktovieaké flagy atď (a navyše sme leniví počítať presnú hĺbku B^+ stromu). Ale z prednášky si pamätáme, že hĺbka B^+ stromov nebýva väčšia než 4, takže dotaz pobeží zhruba 40 ms.
Keď nie sme leniví, môžeme skúsiť odhadnúť skutočnú hĺbku B^+ stromu (budeme predpokladať vyvážený strom, tým pádom je irelevantné, či nás zaujíma priemerný alebo najhorší prípad). Otázkou je, koľko kľúčových hodnôt (telefónnych čísiel) je vo vnútornom uzle toho stromu. Povedzme, že ich je 30 (nejaký priestor zaberú smerníky na synovské uzly, ktorých je o 1 viac než záznamov; smerníky aj telefónne čísla nech sú 64-bitové). Ďalšou otázkou je faktor zaplnenia B^+ stromu. Nech je 0.5, t.j. nech v každom vnútornom uzle je len 15 kľúčových hodnôt. Hĺbka stromu je teda $\lceil \log_{15} 4\ 000\ 000 \rceil = \lceil \ln 4\ 000\ 000 / \ln 15 \rceil = 6$. (Pri odhadoch neznámych parametrov sme boli dosť konzervatívni, takže skutočná hĺbka bude možno menšia.) To zodpovedá „rule of thumb“, dotaz pobeží **rádovo desiatky ms**.

Bonus za empirické overenie na bežnom PC s Postgresql (identifikácia konštánt, vytvorenie databázy, „donútenie“ stroja k zvoleniu daných plánov, merania časov).

Svoje odhady zdôvodnite. **(3)**