21/1/2022 Úvod do databáz, skúškový test, max **60** bodov

1. Uvažujte databázu bez duplikátov a null hodnôt: capuje(Krcma, Alkohol),
lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Idn → Pijan, Krcma; Idn, Alkohol → Mnozstvo; Mnozstvo > 0.
a) Sformulujte bezpečný dotaz v Datalogu **(6)** a relačnej algebre **(6)** na také dvojice [P, K], že pijan P vypil v krčme K len také alkoholy, ktoré neľúbi (pričom P aspoň jeden neobľúbený alkohol v K vypil); a v ostatných krčmách vypil P len také alkoholy, ktoré ľúbi (nie nutne v každej navštívenej krčme musel niečo vypiť).

Datalog:

```
answer(P, K) ←
        navstivil(I, P, K),
        vypil(I, A, _),
        not lubi(P, A),
        not vypil_oblubeny(P, K),
        not inde_vypil_neoblubeny(P, K).

vypil_oblubeny(P, K) ←
        navstivil(I, P, K),
        vypil(I, A, _),
        lubi(P, A),

inde_vypil_neoblubeny(P, K) ←
        navstivil(_, P, K),
        navstivil(I, P, K2),
        vypil(I, A, _),
        not lubi(P, A),
        not K = K2.
```

Relačná algebra:

$$\text{inde\_vypil\_neoblubeny} := \pi_{\text{Pijan, Krcma}} ((\pi_{\text{n1.Idn, n1.Pijan, n1.Krcma}} (\rho_{\text{n1}} (\text{navstivil}) \bowtie_{\text{n1.Pijan = n2.Pijan} \wedge \text{n1.Krcma} \neq \text{n2.Krcma}} \rho_{\text{n2}} (\text{navstivil})) \bowtie \text{vypil}) \triangleright \text{lubi})$$

$$\text{vypil\_oblubeny} := \pi_{\text{Pijan, Krcma}} (\text{navstivil} \bowtie \text{vypil} \bowtie \text{lubi})$$

/* answer */

$$((\pi_{\text{Pijan, Krcma}} (\text{navstivil} \bowtie \text{vypil}) \triangleright \text{lubi}) \triangleright \text{vypil\_oblubeny}) \triangleright \text{inde\_vypil\_neoblubeny}$$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na také dvojice [P, K], že P navštívil krčmu K aspoň 100-krát; a každý alkohol, ktorý K čapuje, vypil P v K v celkovom množstve (počas všetkých návštev K) aspoň 50.

Datalog:

```
answer(P, K) ←
        subtotal(navstivil(I, P, K), [P, K], [N = count(I)]),
        N >= 100,
        not nieco_pil_malo(P, K).

nieco_pil_malo(P, K) ←
        subtotal(ucet(_, P, K, A, M), [P, K, A], [T = sum(M)]),
        T < 50.

nieco_pil_malo(P, K) ←
        navstivil(_, P, K),
        capuje(K, A),
        not niekedy_vypil(P, K, A).

niekedy_vypil(P, K, A) ←
        navstivil(I, P, K),
        vypil(I, A, _).

ucet(I, P, K, A, M) ←
        navstivil(I, P, K),
        vypil(I, A, M).
```

SQL:

```
with nieco_pil_malo as
(
(
select n.Pijan, n.Krcma
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Pijan, n.Krcma, v.Alkohol
having sum(v.Mnozstvo) < 50
)
union
(
select n.Pijan, n.Krcma
from capuje c, navstivil n
where c.Krcma = n.Krcma and not exists (
        select *
        from navstivil n, vypil v
        where n.Idn = v.Idn and c.Alkohol = v.Alkohol)
)
)
select n.Pijan, n.Krcma
from navstivil n
group by n.Pijan, n.Krcma
having count(n.Idn) >= 100 and not exists (
        select *
        from nieco_pil_malo npm
        where n.Pijan = npm.Pijan and n.Krcma = npm.Krcma)
```

2. a) Rozhodnite, či v ľubovoľnej relačnej schéme s atribútmi A, B, C, D, E platí implikácia $((A \rightarrow B) \wedge (BD \rightarrow C)) \Rightarrow (AD \rightarrow C)$.

Ak ÁNO, dokážte s použitím Armstrongovych axióm (uveďte tie axiómy, okomentujte jednotlivé kroky dôkazu). Ak NIE, uveďte kontrapríklad. **(6)**

ÁNO. Uzáver $\{AD\}^*$ (vzhľadom na tie dve funkčné závislosti v predpoklade tej implikácie) obsahuje atribút C. Poďme to tvrdenie dokázať s použitím Armstrongovych axióm.

Armstrongove axiómy:
(A1) $\mathbf{X} \subseteq \mathbf{Y} \Rightarrow \mathbf{Y} \rightarrow \mathbf{X}$ (reflexívnosť)
(A2) $\mathbf{X} \rightarrow \mathbf{Y} \Rightarrow \mathbf{XZ} \rightarrow \mathbf{YZ}$ (rozšírenie)
(A3) $\mathbf{X} \rightarrow \mathbf{Y} \wedge \mathbf{Y} \rightarrow \mathbf{Z} \Rightarrow \mathbf{X} \rightarrow \mathbf{Z}$ (tranzitívnosť)

Dôkaz (priamy):
1. $A \rightarrow B$ (predpoklad)
2. $AD \rightarrow BD$ (aplikácia A2 na 1.)
3. $BD \rightarrow C$ (predpoklad)
4. $AD \rightarrow C$ (aplikácia A3 na 2. a 3.)
QED

b) Uveďte algoritmus (v zrozumiteľnom pseudokóde), ktorý bezstratovo dekomponuje relačnú schému [r, F] do Boyce-Coddovej normálnej formy. Napíšte definície bezstratovej dekompozície (bezstratového spojenia dekompozície) a BCNF. Zdôvodnite prečo dekompozícia získaná uvedeným algoritmom má obe požadované vlastnosti. **(6)**

Definícia: Relačná schéma [r, F] je v Boyce-Coddovej normálnej forme (BCNF), ak pre každú netriviálnu funkčnú závislosť $\mathbf{X} \rightarrow Y$ z F^+ platí $X^+ = r$.

Definícia: Dekompozícia r do r1, …, rN je bezstratová (spája sa bezstratovo), keď pre každú populáciu r, ktorá spĺňa integritné obmedzenia schémy (v tomto prípade dané funkčné závislosti z F^+) platí
$r = \pi_{r1}(r) \bowtie \ldots \bowtie \pi_{rN}(r)$.

Algoritmus dekompozície (rekurzívny):

```
dekomponuj(r, F)
{
        if (v F+ existuje netriviálna funkčná závislosť závislosť X → Y taká, že X+ ≠ r)
        {
                dekomponuj(X ∪ Y, F);
                dekomponuj(r – Y, F);
        }
        else
                zapíš r na výstup;
}
```

Volanie funkcie dekomponuj skončí pre každú dvojicu [r, F], lebo $X \cup Y$ aj $r - Y$ sú menšie relácie ako r. Výsledné relácie sú v BCNF, lebo na výstup sa zapisujú len také, ktoré spĺňajú definíciu BCNF.
V každom kroku dekompozície sa nejaká relácia s(**X**, Y) dekomponuje do dvoch relácií, s1(**X** ∪ Y) a s2(s – Y). Relácie s1 a s2 majú spoločné atribúty **X**, pričom **X** je nadkľúč v s1. Preto s1 a s2 sa spájajú bezstratovo, t.j. $s = s1 \bowtie s2$ pre ľubovoľnú populáciu databázy (ktorá rešpektuje platné funkčné závislosti). Keďže s1 a s2 sa dekomponujú rovnakým spôsobom, platí pre ne rovnaké tvrdenie. Výsledná dekompozícia sa spája bezstratovo do pôvodnej relácie r, s ktorou tá rekurzia začína.

3. Uvažujte Datalogový program s extenzionálnou databázou c(., .):
p(X, Y) ← c(X, _), c(_, Y), not q(X, Y).
p(X, Y) ← c(X, Y), not q(X, Y).
q(X, Y) ← c(X, _), c(_, Y), not c(X, Y).
q(X, Y) ← c(X, Y), q(Y, X).

a) Zapíšte výpočet dotazu ?- p(X, Y) pre daný program v relačnej algebre. **(6)**

Naivná evaluácia:
p := {}
q := {}
φ (
    p := $((\pi_X(c) \times \pi_Y(c)) - q) \cup (c - q)$
    q := $((\pi_X(c) \times \pi_Y(c)) - c) \cup (c - \rho_{q(Y, X)}(q))$
)

/* answer */
p

b) Vypočítajte výsledok dotazu ?- p(X, Y) pre daný program s extenzionálnou databázou c(., .) ={[1, 1], [1, 2], [2, 2], [2, 3], [3, 1], [4, 1], [5, 6]}. **(6)**

p := {}
q := {}

Po 1. iterácii φ:
p = {[1, 1], [1, 2], [2, 2], [2, 3], [3, 1], [4, 1], [5, 6]}
q = {[1, 3], [1, 6], [2, 1], [2, 6], [3, 2], [3, 3], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

Po 2. iterácii φ:
p = {[1, 1], [1, 2], [2, 2], [2, 3], [3, 1], [4, 1], [5, 6]}
q = {[1, 2], [1, 3], [1, 6], [2, 1], [2, 3], [2, 6], [3, 1], [3, 2], [3, 3], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

Po 3. iterácii φ:
p = {[1, 1], [2, 2], [4, 1], [5, 6]}
q = {[1, 2], [1, 3], [1, 6], [2, 1], [2, 3], [2, 6], [3, 1], [3, 2], [3, 3], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

Po 4. iterácii φ:
p = {[1, 1], [2, 2], [4, 1], [5, 6]}
q = {[1, 2], [1, 3], [1, 6], [2, 1], [2, 3], [2, 6], [3, 1], [3, 2], [3, 3], [3, 6], [4, 2], [4, 3], [4, 6], [5, 1], [5, 2], [5, 3]}

Výsledkom dotazu ?- p(X, Y) je **{[1, 1], [2, 2], [4, 1], [5, 6]}**.

4. a) Uveďte metódu (zrozumiteľný pseudokód) wound-or-wait na riešenie deadlockov v transakčnom systéme. Pre každú vstupnú operáciu popíšte akcie, ktoré scheduler používajúci dvojfázové zamykanie s metódou wound-or-wait urobí pri čítaní rozvrhu (predpokladajte, že v systéme bežia len tieto dve transakcie):
start1, rl1(X), start2, r1(X), wl2(Y), w2(X), rl1(Y), wl2(X). **(6)**

Wound-or-wait:
Keď transakcia T1 žiada o zámok, ktorý je konfliktný so zámkom, ktorý v tej chvíli drží T2, sa vykoná nasledujúci kód:

```
if (T1 začala skôr ako T2)
        abort T2;
else
        T1 čaká na pridelenie zámku;
```

Pre daný rozvrh

start1	systém pridelí T1 časovú pečiatku TS(T1)
rl1(X)	systém pridelí T1 rl(X), lebo žiadna iná transakcia nedrží zámok na X
start2	systém pridelí T2 časovú pečiatku TS(T2)
r1(X)	systém vykoná r1(X)
wl2(Y)	systém pridelí T2 wl(Y), lebo žiadna iná transakcia nedrží zámok na Y
w2(X)	systém abortuje T2, lebo sa pokúša zapisovať X bez príslušného zámku
rl1(Y)	systém pridelí T1 rl(Y), lebo žiadna iná transakcia nedrží zámok na X
wl2(X)	systém ignoruje (zahodí) túto operáciu, lebo T2 nie je aktívna transakcia

b) Vysvetlite ako sa dá metóda wound-or-wait použiť v transakčnom systéme, ktorý na izoláciu transakcií používa validačnú metódu. **(6)**

V systéme, ktorý používa validačnú metódu, nevznikajú deadlocky, lebo transakcie na seba navzájom nečakajú (každá transakčná operácia sa po prečítaní buď vykoná alebo sa abortuje transakcia, ktorej patrí). **Použitie wound-or-wait v takomto systéme je nezmyselné.**