5/1/2021 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 dvojice [P, K] také, že krčma K čapuje aspoň dva rôzne alkoholy, ktoré pijan P ľúbi a ktoré P oba vypil v každej krčme, ktorú navštívil (nie nutne oba pri rovnakej návšteve).

Datalog:

```
answer(P, K) ←
        capuje(K, A1),
        capuje(K, A2),
        lubi(P, A1),
        lubi(P, A2),
        not A1 = A2,
        not niekde_nevypil_oba(P, A1, A2).

niekde_nevypil_oba(P, A1, A2) ←
        lubi(P, A1),
        lubi(P, A2),
        navstivil(_, P, K),
        not vypil_oba(P, K, A1, A2).

vypil_oba(P, K, A1, A2) ←
        navstivil(I1, P, K),
        vypil(I1, A1, _),
        navstivil(I2, P, K),
        vypil(I2, A2, _).
```

Relačná algebra:

$vypil\_oba = \pi_{P, K, A1}\ (navstivil \bowtie \rho_{I1, A1, M1}\ (vypil)) \bowtie \pi_{P, K, A2}\ (navstivil \bowtie \rho_{I2, A2, M2}\ (vypil))$

$niekde\_nevypil\_oba = \pi_{P, A1, A2}\ ((\rho_{P, A1}\ (lubi) \bowtie \rho_{P, A2}(lubi) \bowtie \rho_{P, K}\ (navstivil)) \triangleright vypil\_oba)$

$answer = \pi_{P, K}\ ((\rho_{K, A1}\ (capuje) \bowtie_{A1 \neq A2} \rho_{K, A2}\ (capuje) \bowtie \rho_{P, A1}\ (lubi) \bowtie \rho_{P, A1}\ (lubi)) \triangleright niekde\_nevypil\_oba)$

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na trojice [K, A1, A2] také, že krčma K čapuje A1 aj A2; a zároveň alkohol A1 vypilo v krčme K viacej pijanov než alkohol A2; a zároveň alkohol A1 sa vypil v krčme K v menšom celkovom množstve než A2.

Datalog:

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

nvkap(K, A, P) ←
        nv(_, K, P, A, _).

nvikam(I, K, A, M) ←
        nv(I, K, _, A, M).

pocet_mnozstvo(K, A, C, S) ←
        subtotal(nvkap(K, A, P), [K, A], [C = count(P)]),
        subtotal(nvikam(_, K, A, M), [K, A], [S = sum(M)]).

answer(K, A1, A2) ←
        capuje(K, A1),
        capuje(K, A2),
        pocet_mnozstvo(K, A1, C1, S1),
        pocet_mnozstvo(K, A2, C2, S2),
        C1 > C2,
        S1 < S2.
```

SQL:

```
with pocet_mnozstvo as
(
select n.Krcma, v.Alkohol, count(distinct n.Pijan) as C, sum(v.Mnozstvo) as S
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Krcma, v.Alkohol
)
select c1.Krcma, c1.Alkohol as A1, c2.Alkohol as A2
from capuje c1, capuje c2, pocet_mnozstvo pm1, pocet_mnozstvo pm2
where c1.Krcma = c2.Krcma and c1.Krcma = pm1.Krcma and c1.Krcma = pm2.Krcma and c1.Alkohol =
pm1.Alkohol and c1.Alkohol = pm2.Alkohol and pm1.C > pm2.C and pm1.S < pm2.S
```

2.
a) Existuje algoritmus, ktorý ľubovoľnú relačnú schému dekomponuje do Boyce-Coddovej normálnej formy tak, že výsledná dekompozícia je bezstratová (spája sa bezstratovo) a zachováva všetky funkčné závislosti? Odpoveď ÁNO resp. NIE zdôvodnite.

NIE. Taký algoritmus neexistuje, lebo v niektorých prípadoch neexistuje ani dekompozícia s požiadovanými vlastnosťami. Napríklad, relačná schéma [r(A, B, C), {AB→C; C→A}] nie je v BCNF kvôli funkčnej závislosti C→A (C nie je nadkľúč r). Akákoľvek dekompozícia zlomí funkčnú závislosť AB→C.

b) Aký negatívny dôsledok má v databázovej aplikácii použitie dekompozície, ktorá láme (t.j. nezachováva) niektoré funkčné závislosti?

Zlomenie funkčnej závislosti komplikuje automatickú kontrolu konzistencie databázy pri aktualizácii dát a znižuje jej efektivitu. Integritnú podmienku (constraint) zodpovedajúcu zlomenej funkčnej závislosti nie je možné vyjadriť lokálne v rámci jednej relácie (pri create table).
Ak sa zlomená funkčná závislosť vyjadrí ako constraint, ktorý sa týka viacerých relácií, tak pri aktualizácii dát je potrebné počítať ich join atď.

3. Uvažujte relácie r(A1, …, Am, C1, …, Cn), s(B1, …, Bk, C1, …, Cn)
(bez duplikátov a NULL hodnôt; C1, …, Cn sú spoločné atribúty r a s, atribúty A1, …, Am sú len v relácii r, atribúty B1, …, Bk sú len v relácii s) a operátor relačnej algebry α s definíciou:
r α s = {[A1, …, Am]: $\exists$X1, …, $\exists$Xn r(A1, …, Am, X1, …, Xn) $\wedge$
$\forall$B1, …, $\forall$Bk, $\forall$C1, …, $\forall$Cn
(s(B1, …, Bk, C1, …, Cn) $\Rightarrow$ r(A1, …, Am, C1, …, Cn))}.
a) Definujte v Datalogu charakteristický predikát alpha(A1, …, Am) pre r α s. **(6)**

alpha(A_1, …, A_m) $\leftarrow$
    r(A_1, …, A_m, X_1, …, X_n),
    not counterex(A_1, …, A_m).

counterex(A_1, …, A_m) $\leftarrow$
    r(A_1, …, A_m, X_1, …, X_n),
    s(B_1, …, B_k, C_1, …, C_n),
    not r(A_1, …, A_m, C_1, …, C_n).

b) Uveďte výpočet r α s v relačnej algebre bez použitia operátora α (s použitím operátorov z prednášky). **(6)**

counterex = $\pi_{A1, \ldots, Am}$ (($\pi_{A1, \ldots, Am}$ (r) $\times$ $\pi_{C1, \ldots, Cn}$ (s)) - r)
answer = $\pi_{A1, \ldots, Am}$ (r) - counterx

4. Uvažujte vstupný rozvrh s1, r1(X), s2, s3, w3(Y), w2(X), w1(X), c1, r2(Z), w3(Z), c2, c3.
Ku každej z týchto operácií uveďte postupnosť akcií, ktoré urobí po prečítaní danej operácie (dosiaľ nepoužitý) systém, ktorý na izoláciu transakcií používa
a) Metódu časových pečiatok. **(6)**

(časové pečiatky dátových objektov sú inicializované na -∞)

s1 autentifikuje transakciu T1 a pridelí jej časovú pečiatku TS1
r1(X) otestuje TSR(X) < TS1 (= TRUE), priradí TSR(X) := TS1, vykoná r1(X)
s2 autentifikuje transakciu T2 a pridelí jej časovú pečiatku TS2
s3 autentifikuje transakciu T3 a pridelí jej časovú pečiatku TS3
w3(Y) otestuje TSR(Y) < TS3 AND TSW(Y) < TS3 (= TRUE), priradí TSW(Y) := TS3, vykoná w3(Y)
w2(X) otestuje TSR(X) < TS2 AND TSW(X) < TS2 (= TRUE), priradí TSW(X) := TS2, vykoná w2(X)
w1(X) otestuje TSR(X) < TS1 AND TSW(X) < TS1 (= FALSE), abortuje T1
c1 ignoruje
r2(Z) otestuje TSR(Z) < TS2 (= TRUE), priradí TSR(Z) := TS2, vykoná r2(Z)
w3(Z) otestuje TSR(Z) < TS3 AND TSW(Z) < TS3 (= TRUE), priradí TSW(Z) := TS3, vykoná w3(Z)
c2 vykoná commit T2
c3 vykoná commit T3

Výstupný rozvrh:
s1, r1(X), s2, s3, w3(Y), w2(X), a1, r2(Z), w3(Z), c2, c3

b) Validačnú metódu. **(6)**

s1 autentifikuje transakciu T1, pridelí jej časovú pečiatku TS1, inicializuje RS1 = WS1 = {}
r1(X) pridá X do RS1, vykoná r1(X)
s2 autentifikuje transakciu T2, pridelí jej časovú pečiatku TS2, inicializuje RS2 = WS2 = {}
s3 autentifikuje transakciu T3, pridelí jej časovú pečiatku TS3, inicializuje RS3 = WS3 = {}
w3(Y) pridá Y do WS3, vykoná w3(Y) so zápisom do lokálnej kópie Y3
w2(X) pridá X do WS2, vykoná w2(X) so zápisom do lokálnej kópie X2
w1(X) pridá X do WS1, vykoná w1(X) so zápisom do lokálnej kópie X1
c1 pridelí časovú pečiatku TVAL1, otestuje validačnú podmienku pre T1 (úspešne validuje T1), začne kopírovať zápisy T1 do databázy (skopíruje X1 do X), po skopírovaní pridelí časovú pečiatku TF1, commituje T1
r2(Z) pridá Z do RS2, vykoná r2(Z)
w3(Z) pridá Z do WS3, vykoná w3(Z) so zápisom do lokálnej kópie Z3
c2 pridelí časovú pečiatku TVAL2, otestuje validačnú podmienku pre T2 (úspešne validuje T2, lebo hoci WS2 ∩ WS1 = {X}, zápis X od T1 už skončil), začne kopírovať zápisy T2 do databázy (skopíruje X2 do X), po skopírovaní pridelí časovú pečiatku TF2, commituje T2
c3 pridelí časovú pečiatku TVAL3, otestuje validačnú podmienku pre T3 (úspešne validuje T3, lebo hoci WS3 ∩ RS2 = {Z}, T1 už skončila), začne kopírovať zápisy T3 do databázy (skopíruje Y3 do Y a Z3 do Z), po skopírovaní pridelí časovú pečiatku TF3, commituje T3

Výstupný rozvrh:
s1, r1(X), s2, s3, w1(X), c1, r2(Z), w2(X), c2, w3(Y), w3(Z), c3

V uvedenej simulácii predpokladáme, že validáciu, kopírovanie lokálnych dát do databázy a commit vykoná systém v jednom kroku (vzhľadom na neveľký počet kopírovaných dátových objektov to je realistický predpoklad). Oddelená validácia a kopírovanie môže viesť k inému výstupnému rozvrhu.

V oboch prípadoch uveďte tiež výstupný rozvrh, t.j. poradie operácií vykonaných schedulerom systému.