7/1/2016 Úvod do databáz, skúškový test, max 25 bodov, 90 min
1. Uvažujte databázu bez duplikátov a null hodnôt: capuje(Krcma, Alkohol, Cena),
lubi(Pijan, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo). Platí:
Idn → Pijan, Krcma; Idn, Alkohol → Mnozstvo; Krcma, Alkohol → Cena; Mnozstvo > 0.
a) Nájdite trojice [K, A1, A2] také, že krčma K čapuje alkoholy A1 aj A2; a zároveň každý pijan, ktorý ľúbi A1 aj A2, pri každej návšteve krčmy K vypil aspoň jeden z týchto alkoholov. Sformulujte dotaz v Datalogu **(2)**, SQL **(2)** a relačnom kalkule **(2)**.

Datalog:

```
answer(K, A1, A2) ←
        capuje(K, A1, _),
        capuje(K, A2, _),
        not lubi_oba_niekedy_nevypil_ziaden(K, A1, A2).

lubi_oba_niekedy_nevypil_ziaden(K, A1, A2) ←
        lubi(P, A1),
        lubi(P, A2),
        navstivil(I, P, K),
        not v(I, A1),
        not v(I, A2).

v(I, A) ←
        vypil(I, A, _).
```

SQL:

```
with lubi_oba_niekedy_nevypil_ziaden as
(
select n.Krcma, l1.Alkohol as A1, l2.Alkohol as A2
from lubi l1, lubi l2, navstivil n
where l1.Pijan = l2.Pijan and l1.Pijan = n.Pijan and not exists (
        select *
        from vypil v
        where n.Idn = v.Idn and l1.Alkohol = v.Alkohol) and not exists (
        select *
        from vypil v
        where n.Idn = v.Idn and l2.Alkohol = v.Alkohol)
)
select c1.Krcma, c1.Alkohol, c2.Alkohol
from capuje c1, capuje c2
where c1.Krcma = c2.Krcma and not exists (
        select *
        from lubi_oba_niekedy_nevypil_ziaden ln
        where ln.Krcma = c1.Krcma and ln.A1 = c1.Alkohol and ln.A2 = c2.Alkohol)
```

Relačný kalkul:

$$
\begin{aligned}
&\{[K, A1, A2]: \\
&\quad \exists C1\ \exists C2 \\
&\quad capuje(K, A1, C1) \wedge capuje(K, A2, C2) \wedge \\
&\quad \neg (/* \text{ existuje P, ktory lubi A1 a A2 a navsteva I, pri ktorej P v K nepil A1 ani A2 } */ \\
&\qquad \exists P\ \exists I \\
&\qquad lubi(P, A1) \wedge lubi(P, A2) \wedge navstivil(I, P, K) \wedge \\
&\qquad \neg (\\
&\qquad\quad \exists M \\
&\qquad\quad vypil(I, A1, M) \\
&\qquad) \wedge \\
&\qquad \neg (\\
&\qquad\quad \exists M \\
&\qquad\quad vypil(I, A2, M) \\
&\qquad) \\
&\quad) \\
&\}
\end{aligned}
$$

b) Sformulujte dotaz v Datalogu **(2)** a relačnej algebre **(2)**, ktorý nájde trojice [P, K, S], kde S je priemerná suma, ktorú pijan P prepil počas jednej návštevy krčmy K. Do priemeru sa počítajú aj návštevy K, počas ktorých P nič nevypil. Trojice s S=0 nemajú byť vo výsledku.

Datalog:

```
answer(P, K, S) ←
        subtotal(ucet(_, P, K, U), [P, K], [S = avg(U)]),
        S > 0.

ucet(I, P, K, U) ←
        subtotal(polozka(I, _, X), [I], [U = sum(X)]),
        navstivil(I, P, K).

ucet(I, P, K, 0) ←
        navstivil(I, P, K),
        not p(I).

polozka(I, A, X) ←
        navstivil(I, _, K),
        vypil(I, A, M),
        capuje(K, A, C),
        X = M * C.              /* X is M * C */

p(I) ←
        polozka(I, _, _).
```

Relačná algebra:

$polozka = \pi_{Idn,\ Alkohol,\ X = Mnozstvo * Cena}\ (navstivil \bowtie vypil \bowtie capuje)$

$ucet = \pi_{Idn,\ Pijan,\ Krcma,\ U}\ (\Gamma_{Idn,\ U = sum(X)}\ (polozka) \bowtie navstivil) \cup \pi_{Idn,\ Pijan,\ Krcma,\ U = 0}\ (navstivil \triangleright polozka)$

/* main */

$\sigma_{S>0}\ (\Gamma_{Pijan,\ Krcma,\ S = avg(U)}\ (ucet))$

2. Uvažujte reláciu r(A, B, C, D, E, F, G, H) s funkčnými závislosťami
A→DG, ADE→B, ACG→BDE, BG→CF, BE→D, BH→E, BCD→GH, BDG→A, E→B.
a) Nájdite nejaké minimálne pokrytie danej množiny funkčných závislostí. **(2)**

Po minimalizácii ľavých strán kánonických funkčných závislostí:
A→D, A→G, E→B, AC→B, A→D, AC→E, BG→C, BG→F, E→D, BH→E, BCD→G, BCD→H, BDG→A
Minimálne pokrytie:
A→G, A→D, AC→E, BCD→G, BCD→H, BDG→A, BG→C, BG→F, BH→E, E→D, E→B

b) Dekomponujte r do tretej normálnej formy, bezstratovo a so zachovaním všetkých funkčných závislostí. **(1)**

3NF dekompozícia z minimálneho pokrytia:
(A, B, D, G) /* {A, B, D, G} je nadkľúč v r */
(A, C, E)
(B, C, F, G)
(B, C, D, G, H)
(B, E, H)
(D, E)

c) Dekomponujte r do Boyce-Coddovej normálnej formy, bezstratovo. Snažte sa vyhnúť zbytočnému rozbitiu funkčných závislostí. **(2)**

Začnime s 3NF dekompozíciou z úlohy b).
(A, B, D, G) nie je v BCNF kvôli A→G; dekomponujeme do (A, G) a (A, B, D).
(A, B, D) nie je v BCNF kvôli A→D; dekomponujeme do (A, D) a (A, B).
(B, C, D, G, H) nie je v BCNF kvôli BG→C; dekomponujeme do (B, C, G) a (B, D, G, H).
(B, D, G, H) nie je v BCNF kvôli BH→D; dekomponujeme do (B, D, H) a (B, G, H).
(B, E, H) nie je v BCNF kvôli E→B; dekomponujeme do (B, E) a (E, H).

Výsledná BCNF dekompozícia:
(A, B)
(A, D)
(A, G)
(A, C, E),
(B, C, F, G)
(B, E)
(B, D, H)
(B, G, H)
(D, E)
(E, H)

d) Existuje algoritmus, ktorý pre ľubovoľnú danú relačnú schému nájde aspoň jeden jej kľúč v polynomiálnom čase? Ak nie, vysvetlite prečo. Ak áno, uveďte jeho pseudokód. **(2)**

Áno, existuje. Nájde jeden kľúč. Pre relačnú schému s atribútmi $A_1, ..., A_N$ začína s (platnou) funkčnou závislosťou $A_1, ..., A_N \rightarrow A_1, ..., A_N$ a minimalizuje jej ľavú stranu:
L := {A_1, ..., A_N};
while (existuje atribút $A_i \in L$ taký, že $\{L - A_i\}^* = \{A_1, ..., A_N\}$)
L := L – A_i;
Keď tento cyklus skončí, L obsahuje nejaký kľúč tej relačnej schémy. Výpočet uzáveru atribútov má polynomiálnu časovú zložitosť. V podmienke cyklu sa uzáver počíta nanajvýš N krát, cyklus sa opakuje nanajvýš N krát.

3. Predpokladajte, že relácie r(A, B), s(X, Y, Z) neobsahujú duplikáty a NULL hodnoty. Uvažujte SQL dotaz select r.A, s.Z from r full outer join s on (r.A = s.X and r.B = s.Y).
a) Vyjadrite daný dotaz ekvivalentne v relačnom kalkule. **(2)**

$$
\begin{aligned}
&\{[A, Z]: \\
&(\exists B\ r(A, B) \wedge s(A, B, Z)) \\
&\vee \\
&(Z = null \wedge \exists B\ r(A, B) \wedge \neg\ (\exists Z2\ s(A, B, Z2))) \\
&\vee \\
&(A = null \wedge \exists X\ \exists Y\ s(X, Y, Z) \wedge \neg\ r(X, Y)) \\
&\}
\end{aligned}
$$

b) Vyjadrite daný dotaz ekvivalentne v SQL bez použitia kľúčového slova „join". **(2)**

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

4. a) Vysvetlite metódu validácie pre izoláciu transakcií. Validačnú podmienku sformulujte za predpokladu, že validáciu a finalizáciu transakcie systém vykonáva v jednom atomickom kroku. **(2)**

Scheduler používajúci validačnú metódu udržiava pre každú začatú transakciu T množinu identifikátorov objektov RS(T) (objekty, ktoré T čítala), množinu identifikátorov objektov WS(T) (objekty, ktoré T aktualizovala) a niekoľko časových pečiatok.
Operácie $r_i(D)$ a $w_i(D)$ vykonáva v takom poradí, ako prichádzajú (hneď ako ich prečíta); avšak zápisy robí do privátnej kópie objektu D prislúchajúcom transakcii T_i, nie do databázy.
Po prečítaní operácie s_i pridelí transakcii T_i časovú pečiatku $TS(T_i)$.
Po prečítaní operácie a_i zahodí privátnu kópiu všetkých objektov a prislúchajúcich transakcii T_i (tiež zahodí všetky ostatné údaje o T_i).
Po prečítaní operácie c_i pridelí T_i časovú pečiatku $TVAL(T_i)$ a otestuje validačnú podmienku. Ak validačná podmienka nie je splnená, tak T_i abortuje. Inak spustí finalizačnú fázu T_i, v ktorej zapíše privátnu kópiu všetkých aktualizovaných objektov T_i do databázy, vykoná (zapíše do logu) commit T_i a zahodí privátnu kópiu všetkých objektov prislúchajúcich transakcii T_i.

Ak sa validácia a finalizácia vykonáva v jednom atomickom kroku, validačná podmienka pre transakciu T je jednoduchá (všetky časové pečiatky, ktoré neboli pridelené, majú hodnotu ∞):
$\neg (\exists T2\ TS(T) < TVAL(T2) < TVAL(T) \wedge WS(T2) \cap RS(T) \neq \emptyset)$.
Commitované transakcie sú sériované v poradí, v ktorom sa validujú. Vďaka atomickosti validácie a finalizácie sú write-write konflikty vylúčené. Pre úspešnú validáciu stačí zaručiť, aby v prípade write-read konfliktov validovaná transakcia čítala hodnoty objektov zapísané naposledy commitovanou transakciou (teda nie staré hodnoty).

b) Uvažujte validačnú metódu z úlohy a). V systéme bežia dve transakcie, scheduler číta rozvrh s2, r2(Z), s1, r1(X), w2(Y), r1(Y), r2(X), w1(X), c1, w2(Z), c2.
Ku každej prečítanej operácii popíšte akciu, ktorú scheduler vykoná. Uveďte výstupný rozvrh, ktorý scheduler vygeneruje. **(2)**

s2	vyrobí časovú pečiatku TS(T2)
r2(Z)	prečíta hodnotu Z z databázy pre T2 a pridá Z do RS(T2)
s1	vyrobí časovú pečiatku TS(T1)
r1(X)	prečíta hodnotu X z databázy pre T1 a pridá X do RS(T1)
w2(Y)	zapíše novú hodnotu Y do privátnej kópie T2 a pridá Y do WS(T2)
r1(Y)	prečíta hodnotu Y z databázy pre T1 a pridá Y do RS(T1)
r2(X)	prečíta hodnotu X z databázy pre T2 a pridá X do RS(T2)
w1(X)	zapíše novú hodnotu X do privátnej kópie T1 a pridá X do WS(T1)
c1	otestuje validačnú podmienku pre T1; keďže je splnená, zapíše objekty z WS(T1) z privátnej kópie T1 do databázy, zruší privátnu kópiu objektov T1 a commituje T1
w2(Z)	zapíše novú hodnotu Z do privátnej kópie T2 a pridá Z do WS(T2)
c2	otestuje validačnú podmienku pre T2; keďže nie je splnená, abortuje T2

Výstupný rozvrh:
s2, r2(Z), s1, r1(X), r1(Y), r2(X), w1(X), c1, a2

Operácie write, ktorá aktualizujú privátne kópie dát, nie sú vo výstupnom rozvrhu uvedené, lebo sa netýkajú obsahu databázy ani logu. Súvisia len s „interným účtovníctvom“ (podobne sa vo výstupnom rozvrhu neobjaví čítanie a aktualizácia interných katalógových relácií, prideľovanie časových pečiatok atď.). Až bezprostredne pred commitom transakcie, keď sa dáta zapisujú z privátnej kópie do databázy (v ľubovoľnom poradí), sa vo výstupnom rozvrhu objavia príslušné write operácie od tej transakcie.