4/1/2018 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 pijanov, ktorí ľúbia aspoň jeden taký alkohol, ktorý hľadaný pijan vypil v každej krčme, ktorá ten alkohol čapuje.

Datalog:

```
answer(P) ←
        lubi(P, A),
        not capuje_nevypil(P, A).

capuje_nevypil(P, A) ←
        lubi(P, A),
        capuje(K, A),
        not niekedy_vypil(P, A, K).

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

Relačná algebra:

$niekedy\_vypil = \Pi_{Pijan,\ Alkohol,\ Krcma}\ (navstivil \bowtie vypil)$

$capuje\_nevypil = \Pi_{Pijan,\ Alkohol}\ ((lubi \bowtie capuje) - niekedy\_vypil)$

$\Pi_{Pijan}\ (lubi - capuje\_nevypil)$ /* answer */

b) Sformulujte bezpečný dotaz v relačnom kalkule **(6)** a SQL **(6)** na dvojice [P, K] také, že pijan P vypil počas niektorej svojej návštevy krčmy K viacej druhov alkoholu než ktorýkoľvek iný pijan počas všetkých svojich návštev krčmy K.

Relačný kalkul:

$$
\begin{aligned}
&\{[P, K]: \\
&\quad \exists I\, \exists T \\
&\quad \blacktriangledown\ T = count(A) \\
&\quad (\\
&\qquad \exists M \\
&\qquad navstivil(I, P, K) \wedge vypil(I, A, M) \\
&\quad) \wedge \\
&\quad \neg \\
&\quad (\\
&\qquad \exists P2\, \exists T2 \\
&\qquad \blacktriangledown\ T2 = count(A) \\
&\qquad (\\
&\qquad\quad \exists I\, \exists M \\
&\qquad\quad navstivil(I, P2, K) \wedge vypil(I, A, M) \\
&\qquad) \wedge \\
&\qquad P2 \neq P \wedge \\
&\qquad T2 \geq T \\
&\quad) \\
&\}
\end{aligned}
$$

SQL:

```
with
	vypil_1n as (
	select n.Krcma, n.Pijan, count(distinct v.Alkohol) as Total
	from navstivil n, vypil v
	where n.Idn = v.Idn
	group by n.Krcma, n.Pijan, n.Idn),

	vypil_vsn as (
	select n.Krcma, n.Pijan, count(distinct v.Alkohol) as Total
	from navstivil n, vypil v
	where n.Idn = v.Idn
	group by n.Krcma, n.Pijan)

select v1n.Pijan, v1n.Krcma
from vypil_1n v1n
where not exists (
	select *
	from vypil_vsn vsn
	where v1n.Krcma = vsn.Krcma and v1n.Pijan <> vsn.Pijan and v1n.Total <= vsn.Total)
```

2. Uvažujte relačnú schému r(A, B, C, D, E, F, G, H) s funkčnými závislosťami
A→G, AD→B, AC→BDE, BCD→GH, BDG→A, BG→CF, BH→E, E→BD.
Uvažujte dekompozíciu tejto relačnej schémy do r1(B, F, G), r2(A, B, C, D, E, G, H).

a) Rozhodnite či daná dekompozícia je v tretej normálnej forme. Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Nájsť kontrapríklad na 3NF je ťažké bez znalosti kľúčov relačnej schémy. Bez znalosti kľúčov je tiež ťažké dokázať, že schéma je v 3NF. Nájdime teda všetky kľúče r1 a r2. Kľúče r2 nájdeme prehľadaním všetkých podmnožín {A, B, C, D, E, G, H} (s priebežným počítaním uzáverov množín atribútov), pričom niektoré vetvy prehľadávania vieme orezať.

ABCDEGH
- -A: BCDEGH
  - -B: CDEGH
    - -C: DEGH
      - -D: EGH
        - -E: GH
        - +E: EGH
          - -G: EH
          - +G: **EG**
      - +D: DEGH
        - -E: DGH
        - +E: DEH
    - +C: CDEGH
      - -D: CEGH
        - -E: CGH
        - +E: **CE**
      - +D: CDGH
  - +B: BCDEGH
    - -C: BDEGH
      - -D: BEGH
        - -E: BGH
          - -G: BH
          - +G: BGH
            - -H: BG
            - +H: **BGH**
        - +E: BEH
      - +D: BDEGH
        - -E: BDGH
          - -G: BDH
          - +G: **BDG**
        - +E: BDEH
    - +C: BCDGH
      - -D: BCGH
        - -G: BCH
          - -H: BC
          - +H: **BCH**
        - +G: BCG
      - +D: **BCD**
- +A: ABCDEGH
  - -B: ABCDEGH
    - -C: ADEGH
      - -D: AEGH
        - -E: AGH
        - +E: **AE**
      - +D: **AD**
    - +C: **AC**
  - +B: **AB**

Kľúče r2 sú: AB, AC, AD, AE, BCD, BCH, BDG, BGH, CE, EG (iné nie sú).
Podobne zistíme, že jediným kľúčom v r1 je BG.

Podľa definície, relačná schéma (r, **F**) je v 3NF, ak pre každú netriviálnu funkčnú závislosť **X**→Y z $\mathbf{F}^+$ platí, že buď **X** je nadkľúč r alebo Y patrí do niektorého kľúča r.

Každý atribút z {A, B, C, D, E, G, H} patrí do niektorého kľúča r2, preto r2 je v 3NF.

V r1(B, F, G) je F jediným nekľúčovým atribútom. Kandidátmi na porušenie 3NF sú iba funkčné závislosti B→F a G→F. Lenže žiadna z nich nepatrí do $\mathbf{F}^+$ (t.j. neplatí v danej relačnej schéme), lebo $F \notin \{B\}^+$ a $F \notin \{G\}^+$. Teda aj r1 je v 3NF.

Áno, daná dekompozícia je v tretej normálnej forme.

b) Rozhodnite či daná dekompozícia je v Boyce-Coddovej normálnej forme. Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Podľa definície, relačná schéma (r, **F**) je v BCNF, ak pre každú netriviálnu funkčnú závislosť **X**→Y z $\mathbf{F}^+$ platí, že **X** je nadkľúč r.

V r2 platí funkčná závislosť A→G, pričom A nie je nadkľúč r2 (lebo napr. $B \notin \{A\}^+$). Teda r2 nie je v BCNF.

Daná dekompozícia nie je v Boyce-Coddovej normálnej forme.

3. a) Vysvetlite metódu validácie pre izoláciu transakcií. Validačnú podmienku sformulujte za predpokladu, že validáciu a finalizáciu transakcie scheduler vykonáva v jednom atomickom kroku. **(6)**

Scheduler používajúci validačnú metódu udržiava pre každú aktívnu transakciu T množinu identifikátorov objektov RS(T) (objekty, ktoré T čítala), množinu identifikátorov objektov WS(T) (objekty, ktoré T aktualizovala) a niekoľko časových pečiatok (všetky nepridelené časové pečiatky majú hodnotu ∞).
Operácie $r_T(D)$ a $w_T(D)$ (read, write) vykonáva v takom poradí, ako prichádzajú (hneď ako ich prečíta); avšak zápisy robí do privátnej kópie objektu D prislúchajúcom transakcii T, nie do databázy.
Po prečítaní operácie s_T (start) pridelí transakcii T časovú pečiatku TS(T).
Po prečítaní operácie a_T (abort) zahodí privátnu kópiu všetkých objektov prislúchajúcich transakcii T a zahodí tiež všetky ostatné údaje o T.
Po prečítaní operácie c_T (commit) pridelí T časovú pečiatku TVAL(T) a otestuje validačnú podmienku. Ak validačná podmienka nie je splnená, tak T abortuje (takým spôsobom ako po prečítaní a_T). Inak začne finalizačnú fázu T, v ktorej zapíše privátnu kópiu všetkých aktualizovaných objektov T do databázy, následne vykoná (zapíše do logu) c_T a zahodí privátnu kópiu všetkých objektov prislúchajúcich transakcii T.

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

b) Uvažujte scheduler s validačnou podmienkou z úlohy a). Uveďte vstupný rozvrh, ktorý scheduler transformuje do nesériovateľného výstupného rozvrhu, ak validáciu a finalizáciu transakcie nerobí v jednom atomickom kroku. Vysvetlite (po každej prečítanej operácii uveďte akcie, ktoré scheduler vykoná). **(6)**

Uvažujme (sériový) vstupný rozvrh: **s1, w1(X), c1, s2, r2(X), w2(X), c2**
Scheduler môže urobiť pri čítaní týchto operácií nasledujúce akcie:

Vstupný rozvrh	*Výstupný rozvrh*	*Akcia schedulera*
s1	s1	pridelí TS(T1)
w1(X)		zapíše hodnotu X do privátnej kópie T1
c1		úspešne validuje T1 a začne finalizovať T1
s2	s2	pridelí TS(T2)
r2(X)	r2(X)	prečíta (pôvodnú) hodnotu X z databázy
	w1(X)	zapíše hodnotu X z privátnej kópie T1 do databázy
	c1	ukončí finalizáciu T1 a commituje T1
w2(X)		zapíše hodnotu X do privátnej kópie T2
c2		úspešne validuje T2 a začne finalizovať T2
	w2(X)	zapíše hodnotu X z privátnej kópie T2 do databázy
	c2	ukončí finalizáciu T2 a commituje T2

Výstupný rozvrh nie je konflikt-sériovateľný, kvôli konfliktom r2(X)→w1(X)→w2(X).

4. Uvažujte SQL dotaz *select r.X from r, s where r(Y) <> s(Y)*. Relácia r(X, Y) je uložená v 10000 diskových blokoch, relácia s(X, Y) je uložená v 2000 diskových blokoch. Bloky na disku a v operačnej pamäti sú rovnako veľké. Dotaz sa počíta metódou nested-loop join, k dispozícii je 1024 voľných blokov operačnej pamäte.
a) Popíšte organizáciu operačnej pamäte počas výpočtu dotazu. Vysvetlite ako sa rezervované bloky operačnej pamäte použijú pri výpočte výsledku dotazu. **(6)**

1 výstupný blok RAM sa rezervuje na postupné ukladanie a zápis n-tíc, ktoré sú výsledkom joinu.
1 vstupný blok RAM sa rezervuje na postupné čítanie blokov relácie r (lebo je väčšia než s)
1022 vstupných blokov RAM sa rezervuje na čítanie blokov relácie s

Priebeh nested-loop join:
Do RAM postupne načítaj prvých 1022 blokov relácie s.

for i = 1 to 10000
    Do RAM načítaj blok i relácie r (blok v RAM rezervovaný pre čítanie relácie r sa „recykluje“).
    Pre všetky dvojice záznamov r a s práve uložených v RAM vyhodnoť joinovaciu podmienku (where).
    Keď je podmienka splnená (t.j. daná dvojica sa joinuje), aplikuj projekciu (select) na túto dvojicu
    a výslednú n-ticu pridaj do rezervovaného výstupného bloku RAM. Keď je výstupný blok plný, zapíš ho
    na výstup (a „recykluj“ ho pre prípadné ďalšie zápisy).

Do RAM načítaj zvyšných 978 blokov relácie s (bloky v RAM rezervované pre čítanie relácie s sa „recyklujú“)
a zopakuj procedúru z predošlého odstavca.

Ak po ukončení predošlého cyklu výstupný blok nie je prázdny, tak ho zapíš na výstup.

b) Keď každá operácia čítania resp. zápisu diskového bloku trvá 0.1 sekundy, môže výpočet výsledku daného dotazu trvať kratšie než 1 hodinu? Odpoveď ÁNO resp. NIE zdôvodnite. **(6)**

Počas výpočtu nested-loop join každý blok relácie r prečíta dvakrát (20000 operácií).
Každý blok relácie s sa prečíta raz (2000 operácií). Ak sa relácie r a s nejoinujú ani v jednom zázname, tak je výstup prázdny a horeuvedené operácie trvajú dokopy 2200 sekúnd, čo je zhruba 36 minút.

Áno, tento výpočet môže trvať kratšie než 1 hodinu. Ako dlho skutočne potrvá, závisí najmä od veľkosti výstupu (t.j. obsahu relácií r a s), a tiež od reprezentácie relácií, od rýchlosti CPU a RAM atď.