7/1/2026 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 SQL **(6)** na dvojice [K, A] také, že alkohol A sa čapuje v krčme K; a zároveň A ľúbi každý pijan, ktorý vypil A pri každej návšteve krčmy K.

Datalog:

```
answer(K, A) ←
        capuje(K, A),
        not pije_nelubi(K, A).

pije_nelubi(K, A) ←
        capuje(K, A),
        navstivil(_, P, K),
        not niekedy_nevypil(P, K, A),
        not lubi(P, A).

niekedy_nevypil(P, K, A) ←
        capuje(_, A),
        navstivil(I, P, K),
        not v(I, A).

v(I, A) ←
        vypil(I, A, _).
```

SQL:

```
with
niekedy_nevypil as
(
select n.Pijan, n.Krcma, c.Alkohol
from capuje c, navstivil n
where not exists (
        select *
        from vypil v
        where v.Idn = n.Idn and v.Alkohol = c.Alkohol)
),
pije_nelubi as
(
select c.Krcma, c.Alkohol
from capuje c, navstivil n
where c.Krcma = n.Krcma and not exists (
        select *
        from niekedy_nevypil nn
        where nn.Pijan = n.Pijan and nn.Krcma = n.Krcma and nn.Alkohol = c.Alkohol)
        and not exists (
        select *
        from lubi l
        where l.Pijan = n.Pijan and l.Alkohol = c.Alkohol)
)
```

```
select c.Krcma, c.Alkohol
from capuje c
where not exists (
        select *
        from pije_nelubi pn
        where pn.Krcma = c.Krcma and pn.Alkohol = c.Alkohol)
```

b) Sformulujte bezpečný dotaz v Datalogu **(6)** a SQL **(6)** na dvojice [P, A] také, že pijan P pri každej návšteve krčmy vypil alkohol A vo väčšom množstve než ktorýkoľvek iný pijan dokopy počas všetkých návštev tej krčmy.

Datalog:

```
answer(P, A) ←
        navstivil(I, P, _),
        alkohol(A),
        not niekedy_malo(P, A).

niekedy_malo(P, A) ←
        navstivil(I, P, K),
        vypil(I, A, M),
        subtotal(mnozstva(_, P2, K, A, M), [P2, K, A], [S = sum(M)]),
        S >= M,
        not P = P2.

niekedy_malo(P, A) ←
        navstivil(I, P, _),
        alkohol(A),
        not v(I, A).

v(I, A) ←
        vypil(I, A, _).

alkohol(A) ←
        capuje(_, A).

alkohol(A) ←
        lubi(_, A).

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

SQL:

```
with
alkohol as
((
select c.Alkohol
from capuje c
)
union
(
select l.Alkohol
from lubi l
)),
total as
(
select n.Pijan, n.Krcma, v.Alkohol, sum(v.Mnozstvo) as S
from navstivil n, vypil v
where n.Idn = v.Idn
group by n.Pijan, n.Krcma, v.Alkohol
),
niekedy_malo as
((
select n.Pijan, v.Alkohol
from navstivil n, vypil v, total t
where n.Idn = v.Idn and t.Pijan <> n.Pijan, t.Krcma = n.Krcma and t.Alkohol = v.Alkohol and
        t.S >= v.Mnozstvo)
)
union
(
select n.Pijan, a.Alkohol
from navstivil n, alkohol a
where not exists (
        select *
        from vypil v
        where n.Idn = v.Idn and a.Alkohol = v.Alkohol)
))
select n.Pijan, v.Alkohol
from navstivil n, alkohol a
where not exists (
        select *
        from niekedy_malo nm
        where nm.Pijan = n.Pijan and nm.Alkohol = a.Alkohol)
```

2. Uvažujte Datalogový program s EDB databázou a(., .):
p1(X, Y) ⟵ a(X, Z), a(Z, Y), not q1(X, Y).
p2(X, Y) ⟵ a(X, Z), a(Z, Y), not q2(X, Y).
q1(X, Y) ⟵ a(X, Z), a(Z, Y), q1(Y, X).
q2(X, Y) ← a(X, Z), a(Z, Y), not q1(Y, X).
a) Rozhodnite, či dotazy ?- p1(X, Y) a ?- p2(X, Y) dávajú rovnaký výsledok pre ľubovoľnú inštanciu EDB. Zdôvodnite. **(6)**

Predikát q1 je definovaný jediným pravidlom. V tele toho pravidla sa vyžaduje splnenie q1 pre nejakú dvojicu argumentov. Podľa princípu „platí len to, čo platiť musí“, q1(., .) neplatí pre žiadnu dvojicu argumentov. Tým pádom môžeme predpokladať platnosť not q1(., .) pre akúkoľvek dvojicu argumentov v pravidlách pre p1(., .) a q2(., .).

Program sa dá ekvivalentne prepísať takto:
p1(X, Y) ← a(X, Z), a(Z, Y).
p2(X, Y) ← a(X, Z), a(Z, Y), not q2(X, Y).
q2(X, Y) ← a(X, Z), a(Z, Y).

NIE. Dotazy ?- p1(X, Y) a ?- p2(X, Y) nedávajú rovnaký výsledok pre každú inštanciu EDB.
Napríklad pre a(., .) = {[0, 0]} dotaz ?- p1(X, Y) vráti {[0, 0]}, zatiaľ čo dotaz ?- p2(X, Y) vráti prázdnu množinu.

b) Zapíšte výpočty dotazov ?- p1(1, 1) a ?- p2(1, 1) naivnou evaluáciou. **(6)**

Výpočet dotazu ?- p1(1, 1) naivnou evaluáciou:
p1 := {}; p2 := {}; q1 := {}; q2 := {};
ϕ (
$p1 := \pi_{a.X,\, a2.Y}\ (a \bowtie_{a.Y = a2.X} \rho_{a2}(a)) - q1;$
$p2 := \pi_{a.X,\, a2.Y}\ (a \bowtie_{a.Y = a2.X} \rho_{a2}(a)) - q2;$
$q1 := \pi_{a.X,\, a2.Y}\ ((a \bowtie_{a.Y = a2.X} \rho_{a2}(a)) \bowtie_{a2.Y = q.X \wedge a.X = q1.X} q1);$
$q2 := \pi_{a.X,\, a2.Y}\ (a \bowtie_{a.Y = a2.X} \rho_{a2}(a)) - \rho_{q1(Y,\, X)}\ (q1);$
);
$\pi\ \ \sigma_{X = 1 \wedge Y = 1}\ (p1);$

Výpočet dotazu ?- p2(1, 1) je rovnaký, až na posledný krok:
$\pi\ \ \sigma_{X = 1 \wedge Y = 1}\ (p2);$

Optimalizovaný výpočet ?- p1(1, 1):
$\pi\ \ (\pi_Y(\sigma_{X = 1}(a)) \bowtie_{Y = X} \pi_X\ \ (\sigma_{Y = 1}(a)))$

Optimalizovaný výpočet ?- p2(1, 1):
{}

3. a) Napíšte algoritmus (v zrozumiteľnom pseudokóde), ktorý bezstratovo dekomponuje relačnú schému [r, F] do Boyce-Coddovej normálnej formy, a ktorý sa vyhýba zbytočnej dekompozícii. Uveďte definíciu bezstratovej dekompozície a Boyce-Coddovej normálnej formy. Zdôvodnite prečo má výsledná dekompozícia požadované vlastnosti. **(6)**

Definícia. Dekompozícia [r, F] do $[r_1, F_1]$, ..., $[r_n, F_n]$ je bezstratová, ak $r = \pi_{r1}(r) \bowtie \pi_{r2}(r) \bowtie ... \bowtie \pi_{rn}(r)$ pre každú populáciu relácie r, ktorá spĺňa funkčné závislosti platné v r.

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

Algoritmus bezstratovej dekompozície [r, F] do Boyce-Coddovej normálnej formy:

1. Nájdi netriviálnu funkčnú závislosť $\mathbf{X} \rightarrow Y$ z F^+ takú, že **X** nie je nadkľúč v r (tá porušuje BCNF).
2. Ak taká funkčná závislosť existuje, dekomponuj r do r - {Y} a $\mathbf{X} \cup \{Y\}$ a s obomi novými reláciami opakuj tento algoritmus od kroku 1.

Tento algoritmus dekomponuje len relačnú schému, ktorá nie je v BCNF. Zastaví sa až keď celá dekompozícia je v BCNF (funkčná závislosť $\mathbf{X} \rightarrow Y$ neporušuje BCNF v žiadnej z tých nových relácií).

Tento algoritmus v kroku 2 buď neurobí nič, alebo dekomponuje r do dvoch relácií, r - {Y} a $\mathbf{X} \cup \{Y\}$. Pre ten druhý prípad dokážeme, že tie nové relácie sa spájajú bezstratovo do r. Keďže ide o dekompozíciu do dvoch relácií, stačí keď ich spoločné atribúty sú nadkľúčom v aspoň jednej z nich. To je splnené, lebo spoločné atribúty sú **X**, a **X** je nadkľúčom v $\mathbf{X} \cup \{Y\}$. Schémy r1 a r2 môžu byť ďalej dekomponované, ale tento argument sa dá aplikovať na každý dekompozičný krok.

b) Garantuje algoritmus z úlohy a) zachovanie všetkých funkčných závislostí vo výslednej dekompozícii? Odpoveď ÁNO resp. NIE zdôvodnite. Uveďte tiež definíciu zachovania všetkých funkčných závislostí v dekompozícii relačnej schémy. **(6)**

Definícia. Dekompozícia relačnej schémy [r, F] do $[r_1, F_1]$, ..., $[r_n, F_n]$ zachováva všetky funkčné závislosti, ak $F^+ = (\cup_{i=1, ..., n} F_i)^+$.

NIE. Dokonca nie vždy existuje BCNF dekompozícia, ktorá zachováva všetky funkčné závislosti. Uvažujme napríklad relačnú schému r(A, B, C) s funkčnými závislosťami $A \rightarrow B$, $BC \rightarrow A$. Kvôli tej prvej funkčnej závislosti r nie je v BCNF. Avšak po akejkoľvek dekompozícii nebude zachovaná tá druhá funkčná závislosť.

4. a) Vysvetlite metódu validácie pre izoláciu transakcií, v ktorej sa validácia a finalizácia transakcie vykonáva v jednom atomickom kroku. Uveďte validačnú podmienku. **(6)**

Transakcie zapisujú zmeny do databázy až po úspešnej validácii. Scheduler si pre každú začatú transakciu T pamätá časovú pečiatku TS(T), privátnu kópiu objektov ktoré T aktualizovala, množinu identifikátorov objektov RS(T) (objekty, ktoré T čítala), a množinu identifikátorov objektov WS(T) (objekty, ktoré T aktualizovala).
Po prečítaní operácie s_T pridelí transakcii T časovú pečiatku TS(T).
Operácie $r_T(D)$ a $w_T(D)$ vykonáva v takom poradí, ako prichádzajú (hneď ako ich prečíta). Zápisy robí do privátnej kópie objektu D prislúchajúcom transakcii T, nie do databázy. Pri čítaní D preferuje privátnu kópiu objektu D, ak existuje; inak číta hodnotu z databázy.
Po prečítaní operácie a_i zahodí všetky údaje o transakcii T.
Po prečítaní operácie c_T pridelí T časovú pečiatku TVAL(T) a otestuje validačnú podmienku. Ak validačná podmienka nie je splnená, tak abortuje T (tak, ako po prečítaní operácie a_T). Inak zapíše (v atomickom kroku) privátnu kópiu všetkých objektov modifikovaných transakciou T do databázy, a vykoná commit T.
Transakcie sú sériované v poradí, v ktorom sa commitujú. Ak sa validácia a finalizácia vykonávajú v jednom atomickom kroku, následnosť všetkých write-write konfliktov je rovnaká ako následnosť commitov. Stačí sa zamerať na read-write konflikty. Mohlo sa stať toto: s1, s2, r1(D), w2(D), c2. Od tohto momentu systém nesmie dovoliť c1, lebo výstupný rozvrh by nebol konflikt-ekvivalentný sériovému rozvrhu T2 → T1. Validačná podmienka pre transakciu T ošetruje tento prípad (všetky časové pečiatky, ktoré neboli pridelené, majú hodnotu ∞): $\neg (\exists T2\ TS(T) < TVAL(T2) < TVAL(T) \wedge WS(T2) \cap RS(T) \neq \emptyset)$.

b) Uveďte triedu obnoviteľnosti, do ktorej patria rozvrhy generované systémom, ktorý používa metódu izolácie z úlohy a). Zdôvodnite. **(6)**

Preskúmajme, či výstupný rozvrh môže obsahovať dirty read. Predpokladajme, že áno. Nech je vo výstupnom rozvrhu sekvencia operácií w1(D), ..., r2(D), kde r2(D) je prvý dirty read vo výstupnom rozvrhu. Keďže transakcie zapisujú do databázy len súčasne so svojím commitom, musí táto sekvencia vyzerať takto: w1(D), ..., c1, ..., r2(D). To je spor s predpokladom, že r2(D) je dirty read. Tým pádom sú všetky generované rozvrhy obnoviteľné, a vyhýbajúce sa kaskádovým abortom.

Výstupný rozvrh nemôže obsahovať dirty write, lebo hneď po zápisoch všetkých zmien do databázy nasleduje commit transakcie, ktorá tie zápisy urobila.

Rozvrhy generované takýmto systémom sú striktné.