5/1/2012 Úvod do databáz, **skúškový** test, max 25 bodov, 90 min

1. Daná je databáza: capuje(Krcma, Alkohol, Cena), lubi(Pijan, Alkohol)
navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo).
Platí: Idn → Pijan, Krcma; Krcma, Alkohol → Cena; Idn, Alkohol → Mnozstvo;
Mnozstvo > 0, Cena > 0.
a) Sformulujte nasledujúci dotaz v relačnom kalkule **(2)**, Datalogu **(2)**, SQL **(2)**
a relačnej algebre **(2)**: Nájdite dvojice [P, K], o ktorých platí, že pijan P prepil v krčme K celkovo viacej peňazí než prepili v krčme K ľubovoľní iní dvaja pijani dohromady; pritom o každom alkohole, ktorý pijan P v krčme K vypil platí, že P ho vypil v krčme K v menšom množstve než ľubovoľní iní dvaja pijani v krčme K dohromady.

Relačný kalkul:

$$
\begin{aligned}
&\{[P, K]: \\
&\quad /* \text{ definicia vypil\_pri\_navsteve}(I, P, K, A, M, T) */ \\
&\quad (\forall I\ \forall P\ \forall K\ \forall A\ \forall M\ \forall T\ \forall C \\
&\qquad (\\
&\qquad\quad (\text{navstivil}(I, P, K) \wedge \text{vypil}(I, A, M) \wedge \text{capuje}(K, A, C) \wedge T = M * C) \\
&\qquad\quad \Rightarrow \text{vypil\_pri\_navsteve}(I, P, K, A, M, T) \\
&\qquad) \\
&\quad) \wedge \\
&\quad (\exists I\ \exists A\ \exists M\ \exists T\ \text{vypil\_pri\_navsteve}(I, P, K, A, M, T)) \wedge \\
&\quad \neg\ /* \text{ niektori\_dvaja\_aspon\_tolko\_penazi}(P, K) */ \\
&\quad (\exists P1\ \exists P2\ \exists S\ \exists S1\ \exists S2 \\
&\qquad (\heartsuit\ I, A, M, S = \text{sum}(T)\ \text{vypil\_pri\_navsteve}(I, P, K, A, M, T)) \wedge \\
&\qquad (\heartsuit\ I, A, M, S1 = \text{sum}(T)\ \text{vypil\_pri\_navsteve}(I, P1, K, A, M, T)) \wedge \\
&\qquad (\heartsuit\ I, A, M, S2 = \text{sum}(T)\ \text{vypil\_pri\_navsteve}(I, P2, K, A, M, T)) \wedge \\
&\qquad S1 + S2 >= S \wedge P \neq P1 \wedge P \neq P2 \wedge P1 \neq P2 \\
&\quad) \wedge \\
&\quad \neg\ /* \text{ aspon\_tolko\_niektoreho\_alkoholu\_co\_niektori\_dvaja}(P, K) */ \\
&\quad (\exists A\ \exists P1\ \exists P2\ \exists S\ \exists S1\ \exists S2 \\
&\qquad (\heartsuit\ I, T, S = \text{sum}(M)\ \text{vypil\_pri\_navsteve}(I, P, K, A, M, T)) \wedge \\
&\qquad (\heartsuit\ I, T, S1 = \text{sum}(M)\ \text{vypil\_pri\_navsteve}(I, P1, K, A, M, T)) \wedge \\
&\qquad (\heartsuit\ I, T, S2 = \text{sum}(M)\ \text{vypil\_pri\_navsteve}(I, P2, K, A, M, T)) \wedge \\
&\qquad S1 + S2 <= S \wedge P \neq P1 \wedge P \neq P2 \wedge P1 \neq P2 \\
&\quad) \\
&\}
\end{aligned}
$$

Datalog:

```
answer(P, K) ←
        vypil_pri_navsteve(_, P, K, _, _, _),
        not niektori_dvaja_aspon_tolko_penazi(P, K),
        not aspon_tolko_niektoreho_alkoholu_co_niektori_dvaja(P, K).

vypil_pri_navsteve(I, P, K, A, M, T) ←
        navstivil(I, P, K),
        vypil(I, A, M),
        capuje(K, A, C),
        T = M * C.              /* T is M * C */

niektori_dvaja_aspon_tolko_penazi(P, K) ←
        subtotal(vypil_pri_navsteve(_, P, K, _, _, T), [P, K], [S = sum(T)]),
        subtotal(vypil_pri_navsteve(_, P1, K, _, _, T), [P1, K], [S1 = sum(T)]),
        subtotal(vypil_pri_navsteve(_, P2, K, _, _, T), [P1, K], [S2 = sum(T)]),
        S1 + S2 >= S,           /* ST is S1 + S2, ST >=S */
        not P = P1,
        not P = P2,
        not P1 = P2.

aspon_tolko_niektoreho_alkoholu_co_niektori_dvaja(P, K) ←
        subtotal(vypil_pri_navsteve(_, P, K, A, M, _), [P, K, A], [S = sum(M)]),
        subtotal(vypil_pri_navsteve(_, P1, K, A, M, _), [P1, K, A], [S1 = sum(M)]),
        subtotal(vypil_pri_navsteve(_, P2, K, A, M, _), [P1, K, A], [S2 = sum(M)]),
        S1 + S2 <= S,           /* ST is S1 + S2, ST <=S */
        not P = P1,
        not P = P2,
        not P1 = P2.
```

SQL:

```
create temporary table vypil_pri_navsteve as
select n.Idn, n.Pijan, n.Krcma, v.Alkohol, v.Mnozstvo, v.Mnozstvo * c.Cena as T
from navstivil n, vypil v, capuje c
where n.Idn = v.Idn and n.Krcma = c.Krcma and v.Alkohol = c.Alkohol

create temporary table niektori_dvaja_aspon_tolko_penazi as
select p.Pijan, p.Krcma
from
        (select vpn.Pijan, vpn.Krcma, sum(vpn.T as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma) p,
        (select vpn.Pijan, vpn.Krcma, sum(vpn.T as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma) p1,
        (select vpn.Pijan, vpn.Krcma, sum(vpn.T as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma) p2
where p.Krcma = p1.Krcma and p.Krcma = p2.Krcma and p1.S + p2.S >= p.S and
        p.Pijan <> p1.Pijan and p.Pijan <> p2.Pijan and p1.Pijan <> p2.Pijan

create temporary table aspon_tolko_niektoreho_alkoholu_co_niektori_dvaja as
select p.Pijan, p.Krcma
from
        (select vpn.Pijan, vpn.Krcma, sum(vpn.Mnozstvo as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma, vpn.Alkohol) p,
        (select vpn.Pijan, vpn.Krcma, sum(vpn.Mnozstvo as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma, vpn.Alkohol) p1,
        (select vpn.Pijan, vpn.Krcma, sum(vpn.Mnozstvo as S)
        from vypil_pri_navsteve vpn group by vpn.Pijan, vpn.Krcma, vpn.Alkohol) p2
where p.Krcma = p1.Krcma and p.Krcma = p2.Krcma and
        p.Alkohol = p1.Alkohol and p.Alkohol = p2.Alkohol and p1.S + p2.S <= p.S and
        p.Pijan <> p1.Pijan and p.Pijan <> p2.Pijan and p1.Pijan <> p2.Pijan

select vpn.Pijan, vpn.Krcma /* main */
from vypil_pri_navsteve vpn
where not exists (select * from niektori_dvaja_aspon_tolko_penazi n2atp
        where vpn.Pijan = n2atp.Pijan and vpn.Pijan = n2atp.Krcma)
        and not exists (select * from
        aspon_tolko_niektoreho_alkoholu_co_niektori_dvaja atancn2
        where vpn.Pijan = atancn2.Pijan and vpn.Pijan = atancn2.Krcma)
```

Relačná algebra:

$vpn = \Pi_{Idn,\ Pijan,\ Krcma,\ Alkohol,\ Mnozstvo,\ T = Cena * Mnozstvo}\ (navstivil \bowtie vypil \bowtie capuje)$

$n2atp = \Pi_{Pijan,\ Krcma}\ (P_p(\Gamma_{Pijan,\ Krcma,\ S = sum(T)}\ vpn) \bowtie_{p.Pijan <> p1.Pijan \wedge p.Krcma = p1.Krcma} P_{p1}(\Gamma_{Pijan,\ Krcma,\ S = sum(T)}\ vpn) \bowtie_{p.Pijan <> p2.Pijan \wedge p1.Pijan <> p2.Pijan \wedge p.Krcma = p2.Krcma \wedge p1.S + p2.S >= p.S} P_{p2}(\Gamma_{Pijan,\ Krcma,\ S = sum(T)}\ vpn))$

$atancn2 = \Pi_{Pijan,\ Krcma}\ (P_p(\Gamma_{Pijan,\ Krcma,\ Alkohol,\ S = sum(Mnozstvo)}\ vpn) \bowtie_{p.Pijan <> p1.Pijan \wedge p.Krcma = p1.Krcma} P_{p1}(\Gamma_{Pijan,\ Krcma,\ Alkohol,\ S = sum(Mnozstvo)}\ vpn) \bowtie_{p.Pijan <> p2.Pijan \wedge p1.Pijan <> p2.Pijan \wedge p.Krcma = p2.Krcma \wedge p1.S + p2.S <= p.S} P_{p2}(\Gamma_{Pijan,\ Krcma,\ Alkohol,\ S = sum(Mnozstvo)}\ vpn))$

/* main */

$\Delta\ \Pi_{Pijan,\ Krcma}\ (vpn) - \Delta\ n2atp - \Delta\ atancn2$

b) Sformulujte v Datalogu **(2)** dotaz, ktorý nájde dvojice [P, A] také, že pijan P ľúbi alkohol A a preferuje ho pri každej návšteve krčmy, ktorá alkohol A čapuje. (V takej krčme P vypije iný alkohol len vtedy, keď pri tej istej návšteve vypije aj alkohol A.)

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

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

v(I, A) ←
        vypil(I, A, _).
```

2. V relácii d(From, To) sú uložené priame lety, ktore ponúka letecká spoločnosť. Slovenských klientov zaujímajú destinácie priamych aj nepriamych (kombinovaných) letov, ktoré začínaju v Bratislave (bts).
a) Sformulujte tento dotaz v Datalogu. **(2)**

answer(F, T) ← d(F, T).
answer(F, T) ← d(F, V), answer(V, T).
?- answer(bts, T).

b) Sformulujte tento dotaz v SQL za predpokladu, že použitý databázový systém implementuje SQL3 štandard (t.j. rekurziu). **(2)**

```
with recursive answer as
(
d /* alebo ešte lepšie: select d.From, select d.To from d where d.From = ‘bts’ */
union
select d.From, answer.To
from d, answer
where d.To = answer.From
)
select distinct answer.To /* dá sa pridať aj answer.From (=bts) */
from answer
where answer.From = ‚bts‘
```

c) Sformulujte tento dotaz v SQL bez použitia *recursive with* a *recursive view*. Predpokladajte pritom, že pri kombinovanom lete sú možné maximálne 2 prestupy. **(2)**

```
(select d.To /* 0 prestupov */
from d
where d.From = ‚bts‘)
union
(select d2.To /* 1 prestup */
from d d1, d d2
where d1.From = ‚bts‘ and d1.To = d2.From)
union
(select d3.To /* 2 prestupy */
from d d1, d d2, d d3
where d1.From = ‚bts‘ and d1.To = d2.From and d2.To = d3.From)
```

3. Dané sú relácie r(X, Y, Z) a s(A, B, C). Vyjadrite nasledujúce dotazy ekvivalentne v SQL bez použitia *distinct*:
a) select distinct r.X from r, s where r.Y = s.A **(2)**

```
select r.X
from r, s
where r.Y = s.A
group by r.X
```

b) select r.X from r, s where r.Y = s.A group by r.X having count(distinct s.B) < 85 **(2)**

```
create temporary table proj_cnt as
select r.X, s.B
from r, s
where r.Y = s.A
group by r.X, s.B

/* main */
select p.X
from proj_cnt p
group by p.X
having count(p.B) < 85
```

4. a) Môže deadlock vzniknúť v systéme, ktorý používa na implementáciu izolácie transakcií validačnú metódu? Zdôvodnite svoju odpoveď áno, resp. nie. **(1)**

Nie, nemôže. Pri validačnej metóde sa všetky vstupné operácie vykonávajú hneď, t.j. bez čakania na čokoľvek.

b) Popíšte metódu wound-or-wait na riešenie deadlockov. **(2)**

Systém si pamätá časové pečiatky aktívnych transakcií (časová pečiatka hovorí, kedy transakcia začala). Vždy keď nejaká transakcia T1 žiada o konfliktný zamok, ktorý v tom momente drží iná transakcia T2, systém spustí nasledujúci program:
if (TS(T1) < TS(T2))
wound T2;
else
let T1 wait for the lock;
Inak povedané, mladšia transakcia čaká na uvoľnenie zámku držaného staršou transakciou. Staršia transakcia si "kliesni cestu" zraňovaním mladších transakcií, ktoré držia konfliktné zámky. Zranenie sa rovná abortu (až na prípad, keď mladšia transakcia už predtým požiadala o commit a systém jej práve odoberá zámky—vtedy nechá system zranenú transakciu commitovať).

c) Popíšte čo najpresnejšie akcie, ktoré scheduler používajúci wound-or-wait vykoná pri čítaní nasledujúcej postupnosti operácií (v systéme bežia len tieto transakcie): start2, start1, rl1(X), wl2(Y), rl1(Y), wl2(X). **(2)**

start2	Pridelí T2 časovú pečiatku.
start1	Pridelí T1 časovú pečiatku.
rl1(X)	Pridelí T1 read-lock na X.
wl2(Y)	Pridelí T2 write-lock na Y.
rl1(Y)	Toto je prvý konflikt. Keďže T1 je mladšia než T2, systém nechá T1 čakať na uvoľnenie zámku na Y.
wl2(X)	Toto je ďalší konflikt. T2 je staršia než T1, takže system zraní T1. Toto zranenie je "smrteľné", t.j. rovná sa abortu T1. Systém abortuje T1, odoberie T1 všetky zámky a pridelí T2 write-lock na X.