AB2 zh2 2021. május 10. 10:15 **Név:**
Neptun kód:

„Papíron” megoldandó feladatok

A megoldásokat ebbe az állományba írják bele, és küldjék el a nikovits@inf.elte.hu címre!

1. feladat (5 + 5 pont)

a) A következő **UNDO** típusú naplóbejegyzés-sorozat a T, U és V tranzakciókra vonatkozik:
`<start V> <V,A,15> <start T> <T,B,20> <V,A,30> <start U> <T,D,80> <V,C,45> <T,B,40> <U,E,50> <T,D,50> <COMMIT T> <abort U> <V,A,50>` **`<V,C,60>`**
(`<COMMIT V>` még nincs a lemezen)
Adjuk meg a helyreállításhoz szükséges tevékenységeket, ha az utolsó lemezre került naplóbejegyzés: **`<V, C, 60>`**. `WRITE, OUTPUT, <napló>, FLUSH LOG` műveletek megfelelő sorrendjét kell megadni.

A megoldást kérem ide írják:

WRITE(C,60) OUTPUT(C)
WRITE(A,50) OUTPUT(A)
WRITE(C,45) OUTPUT(C)
WRITE(A,30) OUTPUT(A)
WRITE(A,15) OUTPUT(A)
<ABORT,V>
FLUSH LOG

---

b) A következő **REDO** típusú naplóbejegyzés-sorozat a T, U és V tranzakciókra vonatkozik:
`<start U> <U,A,10> <start V> <V,B,22> <U,A,30> <start T> <V,D,75> <U,C,50> <V,B,47> <T,E,50> <V,D,48> <COMMIT T> <COMMIT V> <U,A,50> <END T>` **`<U,C,70>`**
(`<COMMIT U>` még nincs a lemezen)
Adjuk meg a helyreállításhoz szükséges tevékenységeket, ha az utolsó lemezre került naplóbejegyzés: **`<U, C, 70>`**. `WRITE, OUTPUT, <napló>, FLUSH LOG` műveletek megfelelő sorrendjét kell megadni.

A megoldást kérem ide írják:

WRITE(B,22) OUTPUT(B)
WRITE(D,75) OUTPUT(D)
WRITE(B,47) OUTPUT(B)
WRITE(D,48) OUTPUT(D)
<END, V>
FLUSH LOG

---

2. feladat (10 pont)

Adjuk meg a konfliktus-sorbarendezhető ütemezések számát az alábbi tranzakció párokra.
Indokoljuk is meg az eredményt!

a) **T1**: W1(A); R1(B); W1(B); R1(C); **T2**: R2(B); W2(B); R2(A);
b) **T1**: R1(A); W1(A); W1(C); R1(B); **T2**: R2(A); W2(B); W2(A);
c) **T1**: W1(A); R1(B); W1(B); **T2**: R2(C); W2(C); W2(D);

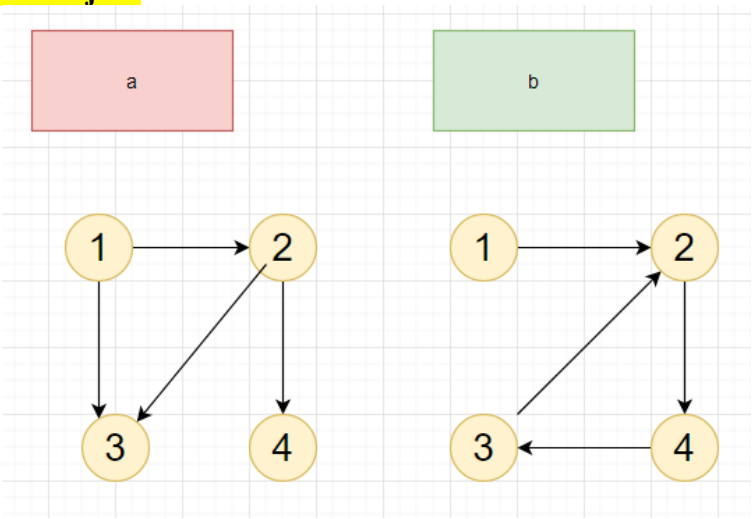
A megoldást kérem ide írják:

3. feladat (8 pont)
Adja meg az alábbi ütemezések megelőzési gráfját. (A gráf megrajzolásához használhatja a https://app.diagrams.net/ programot és a honlapon szereplő **graf_rajzolas.svg** állományt.) Adja meg az összes soros ütemezést (elég tranzakciók megadásával, nem kell műveletenként), amelyek konfliktus-elvivalensek a megadott ütemezéssel.

a) `R1(A); W2(B); R1(C); R2(A); R4(B); W2(A); R3(B); W1(C); R3(C); W4(D);`

b) `R1(A); R3(B); R4(C); R1(B); R2(A); W2(A); W2(B); W3(C); R4(A); W4(D);`

A megoldást kérem ide írják:

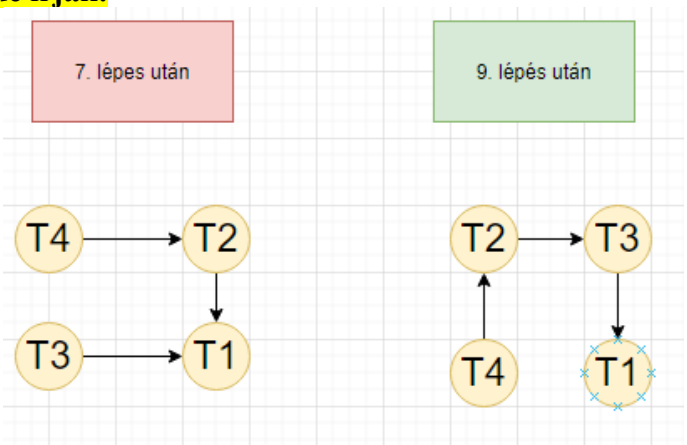


a, 1,2,3,4
b, Nincs konfliktus ekvivalens ütemezés (irányított kör)

---

4. feladat (4 + 4 pont)
Rajzoljuk fel a következő ütemezéshez tartozó várakozási gráfot a 7., és 9. lépés után.
(tegyük fel, hogy egy felszabaduló zárat az a várakozó fog megkapni, aki a legrégebben vár)
l1(B); l2(C); l3(D); l1(A); l2(B); l3(A); l4(C); u1(B); l2(D);

A megoldást kérem ide írják:



Számítógépes feladatok

5. feladat (8 pont, ez a kötelezően megoldandó)
Küldjék el egy külön szöveges állományban a zh2_kotelezo_feladat.txt állományban leírt feladat megoldását. A forráskódot és az alábbi futtatás eredményét kell elküldeni.

```
CALL print_histogram('nikovits','test2','yr');
```

A további lekérdezésekhez az alábbi, NIKOVITS felhasználó tulajdonában levő táblákat kell használni.

```
CIKK(ckod, cnev, szin, suly)
SZALLITO(szkod, sznev, statusz, telephely)
PROJEKT(pkod, pnev, helyszin)
SZALLIT(szkod, ckod, pkod, mennyiseg, datum)
```

6. feladat (4 + 4 pont)
Adjuk meg a szegedi telephelyű szállítók (telephely='Szeged') által szállított piros színű cikkek összmennyiségét. (A SZALLIT tábla `mennyiseg` oszlopát kell összegezni.)

a) Adjuk meg úgy a lekérdezést hintek segítségével, hogy a végrehajtási tervben minden join művelet NESTED LOOPS legyen, és indexet ne használjon a rendszer.

b) Adjuk meg úgy a lekérdezést hintek segítségével, hogy a végrehajtási tervben minden join művelet SORT-MERGE JOIN legyen, és a végrehajtás egy indexet használjon.

A **lekérdezést**, annak **végeredményét** és a **végrehajtási tervet** is el kell küldeni.
A megoldást kérem ide másolják be:

a)

```
SELECT /*+ full(p) full(c) full(sz) use_nl(p,c,sz) */
SUM(sz.mennyiseg) AS DB
FROM nikovits.projekt p, nikovits.szallit sz, nikovits.cikk c
WHERE p.helyszin = 'Szeged' AND c.szin='piros'
AND sz.pkod = p.pkod AND sz.ckod = c.ckod;
```

EREDMENY:

```
DB  |
----+
8796|
```

TERV:

Operation	Object	Optimizer	Cost	Cardinality	Bytes
SELECT STATEMENT		ALL_ROWS	2,561	1	30
SORT (AGGREGATE)			2,561	1	30
NESTED LOOPS			2,561	219	6,570
NESTED LOOPS			1,189	1,011	20,220
TABLE ACCESS (FULL)	CIKK	ANALYZED	3	101	1,010
TABLE ACCESS (FULL)	SZALLIT	ANALYZED	12	10	100
TABLE ACCESS (FULL)	PROJEKT	ANALYZED	1	1	10

b)

```
SELECT /*+ USE_MERGE(p,sz,c) index(p) */
SUM(sz.mennyiseg) AS DB
FROM nikovits.projekt p, nikovits.szallit sz, nikovits.cikk c
WHERE p.helyszin = 'Szeged' AND c.szin='piros'
AND sz.pkod = p.pkod AND sz.ckod = c.ckod;
```

EREDMENY:

```
DB  |
----+
8796|
```

TERV:

Operation	Object	Optimizer	Cost	Cardinality	Bytes
SELECT STATEMENT		ALL_ROWS	21	1	30
SORT (AGGREGATE)			21	1	30
MERGE JOIN			21	219	6,570
SORT (JOIN)			17	2,167	43,340
MERGE JOIN			16	2,167	43,340
TABLE ACCESS (BY INDEX ROWID)	PROJEKT	ANALYZED	2	7	70
INDEX (FULL SCAN)	P_PKOD	ANALYZED	1	30	0
SORT (JOIN)			14	10,000	100,000
TABLE ACCESS (FULL)	SZALLIT	ANALYZED	13	10,000	100,000
SORT (JOIN)			4	101	1,010
TABLE ACCESS (FULL)	CIKK	ANALYZED	3	101	1,010

A táblára nem kellett létrehoznom indexet, mert már volt rajta.

---

7. feladat (8 pont)
Adjunk meg egy olyan lekérdezést (hintekkel együtt, ha szükséges), aminek az alábbi lesz a végrehajtási terve:

```
SELECT STATEMENT
  SORT + GROUP BY
    HASH JOIN
      TABLE ACCESS + FULL + PROJEKT
      HASH JOIN
        TABLE ACCESS + BY INDEX ROWID BATCHED + CIKK
          INDEX + RANGE SCAN + C_SZIN
        TABLE ACCESS + FULL + SZALLIT
```

Adjuk meg a **lekérdezést**, és a **végrehajtási tervet** is, még akkor is, ha nem teljesen egyezik meg a végrehajtási terv az elvárttal. A feladatra részpontszám is szerezhető.
A megoldást kérem ide másolják be:

```
SELECT /*+ INDEX_ASC(c c_szin) USE_HASH(sz c p) */
helyszin, SUM(sz.mennyiseg) AS DB
FROM nikovits.projekt p, nikovits.szallit sz, nikovits.cikk c
WHERE c.szin like'piros'
AND sz.pkod = p.pkod AND sz.ckod = c.ckod
GROUP BY helyszin
ORDER BY helyszin ASC;
```

TERV

Operation	Object	Optimizer	Cost	Cardinality	Bytes
SELECT STATEMENT		ALL_ROWS	23	5	150
SORT (GROUP BY)			23	5	150
HASH JOIN			22	1,011	30,330
TABLE ACCESS (FULL)	PROJEKT	ANALYZED	3	30	300
HASH JOIN			19	1,011	20,220
TABLE ACCESS (BY INDEX ROWID BATCHED)	CIKK	ANALYZED	6	101	1,010
INDEX (RANGE SCAN)	C_SZIN	ANALYZED	1	101	0
TABLE ACCESS (FULL)	SZALLIT	ANALYZED	13	10,000	100,000