Adatbázisok II.

2. ZH elméleti része

A jegyzetet Csonka Szilvia írta. A jegyzet első számú forrása a gyakorlat.

Információk

- pótZH: vizsgaidőszak 1. hetében.
- A ZH $\frac{1}{4}$ 9-kor kezdődik.

1. Rész

Források

- A feladatsor és további elméleti részek Nikovits Tibor honlapján: `http://people.inf.elte.hu/nikovits/AB2/ab2_feladat9.txt`.
- További kapcsolódó források a gyakorlatvezető honlapjáról:
  - UW_redo_naplo.doc
  - UW_undo_naplo.doc
  - UW_undo_redo_naplo.doc
- Az előadás honlapján található (`http://people.inf.elte.hu/kiss/15ab2/13ab2osz.htm`) *naplo.ppt* egyes diái (a 8., 9., 10. előadás anyagai között megtalálható) szintén segítségül szolgálnak az alábbi feladatok megoldásának megértéséhez.

8.1.1 Feladat (könyv 463. old.)

Tegyük fel, hogy az adatbázisra vonatkozó konzisztenciamegszorítás: 0 <= A <= B.
Állapítsuk meg, hogy a következő tranzakció megőrzik-e az adatbázis konzisztenciáját.
T1: $A := A + B; B := A + B;$

- *Megjegyzés:* itt valójában $B := A + 2B$ fog végrehajtódni, ugyanis az értékadásokat szekvenciálisan, egymás után fogjuk végrehajtani.
- *Megoldás:*

```
INPUT(A)     //A-t beolvassuk a memóriapufferbe a lemezről
INPUT(B)     //B-t is
READ(A, t)   //a memóriából a t lokális változóba másoljuk A-t
READ(B, s)   //B-t pedig s-be
t := t + s   //végrehajtjuk a kért értékadások közül az elsőt
WRITE(A, t)  //az eredményt a memóriapufferbe írjuk
s := s + t   //végrehajtjuk a második értékadást is
WRITE(B, s)  //ezt is a memóriába írjuk
OUTPUT(A)    //a memóriából a lemezre mentjük A eredményét
OUTPUT(B)    //B-ét is
```

- *Mejegyzés:* itt nem kérdezték a naplózást, így azt ebben a feladatban nem kell részleteznünk.

8.2.4 Feladat(könyv 476. old.)

A következő naplóbejegyzés-sorozat a T és U két tranzakcióra vonatkozik:

```
<start T>
<T, A, 10>
<start U>
<U, B, 20>
<T, C, 30>
<U, D, 40>
<T, A, 11>
<U, B, 21>
<COMMIT U>
<T, E, 50>
<COMMIT T>
```

Adjuk meg a helyreállítás-kezelő tevékenységeit, ha az utolsó lemezre került naplóbejegyzés:

a) <start U>

b) <COMMIT U>

c) <T, E, 50>

d) <COMMIT T>

- ***Megoldás UNDO helyreállítás (lásd. naplo.ppt 54. dia) esetén:***

  a) Az adatvesztéssel járó hiba bekövetkeztéig a következők futottak le:

  ```
  <start T>
  <T, A, 10>
  <start U>
  ```

  Megoldás:

  ```
  WRITE(A, 10) //Visszaállítjuk A T tranzakció lefutása előtti állapotát.
  OUTPUT(A)    //Lemezre írjuk A helyreállított változatát.
  <T, ABORT>   //Naplózzuk, hogy T tranzakció eredményét visszaállítottuk.
               //Azaz helyreállítottuk, a nem teljesen lefutott tranzakció
               // okozta nem konzisztesn állapotot, hogy ismét konzisztens
               // legyen az adatbázis.
  FLUSH LOG   //Naplózást is lementjük.
  ```

  b) Az adatvesztéssel járó hiba bekövetkeztéig a következők futottak le:

  ```
  <start T>
  <T, A, 10>
  <start U>
  <U, B, 20>
  <T, C, 30>
  <U, D, 40>
  <T, A, 11>
  <U, B, 21>
  <COMMIT U>
  ```

  Megoldás:

  ```
  WRITE(A, 11)
  OUTPUT(A)
  WRITE(C, 30)
  OUTPUT(C)
  WRITE(A, 10)
  OUTPUT(A)
  <T, ABORT> //Naplóbejegyzés: helyreállítottuk A állapotát a T előttire
  FLUSH LOG  //Lementjük a fenti naplóbejegyzést is.
  ```

Megjegyzés: Visszaállítjuk a dolgokat a napló végétől az eleje felé haladva. Az *U* tranzakció által végrehajtott módosításokat nem kell helyreállítanunk, ugyanis a hiba előtt ez már sikeresen lefutott és le is lett mentve (*COMMIT*) az eredménye, tehát nem zavar bele az adatbázis konzisztenciájába.

c) (Gyakorlaton nem volt.)

d) Minden le lett mentve az adatvesztéssel járó hiba előtt, tehát nem szükséges semmit sem helyreállítani.:)

– ***Megoldás REDO helyreállítás (lásd. naplo.ppt 78. dia) esetén:***
Megjegyzés: a dián a végéről lemaradt a "FLUSH LOG".

a) (Nem volt gyakorlaton.)

b) Az adatvesztéssel járó hiba bekövetkeztéig a következők futottak le:

```
<start T>
<T, A, 10>
<start U>
<U, B, 20>
<T, C, 30>
<U, D, 40>
<T, A, 11>
<U, B, 21>
<COMMIT U>
```

Megoldás:

```
WRITE(B, 20)
OUTPUT(B)
WRITE(D, 40)
OUTPUT(D)
WRITE(B, 21)
OUTPUT(B)
FLUSH LOG
```

Megjegyzés:
REDO esetén azokat az adatokat állítjuk helyre a tranzakció lefutása utáni állapotukra, amelyekre volt *COMMIT*, azaz lefutottak, csak eredményük nem került egészében mentésre (mivel eközben következett be a hiba), azaz nem volt *END*. Így mivel *U* tranzakcióra volt *COMMIT*, de *END* nem, ezért a fenti példában ennek az eredményét állítottuk helyre. *T*-vel nem foglalkozunk, hiszen arra *COMMIT* sem volt, azaz be sem fejeződött, így nincs lenaplózva és nem is tudhatjuk az összes adatváltoztatásának következményét (azaz ha helyreállítanánk a naplóban szereplő eddigi adatváltoztatásait, nem konzisztens állapotú adatbázist kapnánk).

Az Oracle naplózása

– Az Oracle valójában *REDO* naplózást használ. Így elég csak az utasítást lementenie egy update-ről.

– Az *UNDO* naplózásra külön *UNDO táblateret* használ, mivel ez nagyon memóriaigényes, hiszen ekkor az összes olyan táblaelem korábbi értékét le kell menteni, amelyet az adott update megváltoztatott, azaz előfordulhat, hogy komplett táblákat kell eltárolni. Ezen mentések felhasználásával tudja megoldani az Oracle, hogy amennyiben egyszerre többen használnak egy táblát, akkor a használat kezdeti pillanatbeli állapotát bocsájtja a felhasználó rendelkezésére a táblának.

2. Rész

Források

- Az előadás diái közül a *konkurencia.ppt*, mely a 10., 11., 12. előadás anyagánál megtalálható az előadó honlapján: `http://people.inf.elte.hu/kiss/15ab2/13ab2osz.htm`
- A tankönyv ütemezésekre vonatkozó része a gyakorlat honlapján: `http://people.inf.elte.hu/nikovits/AB2/` *UW_ utemezesek.doc*.
- A feladat és további elméleti részek Nikovits Tibor honlapján: `http://people.inf.elte.hu/nikovits/AB2/ab2_feladat10.txt`