Semmisségi/helyrehozó (undo/redo) naplózás

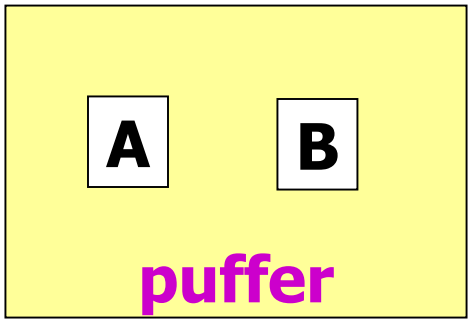
• A semmisségi naplózás esetén az adatokat a tranzakció befejezésekor nyomban lemezre kell írni, nő a végrehajtandó lemezműveletek száma.

• A helyrehozó naplózás minden módosított adatbázisblokk pufferben tartását igényli egészen a tranzakció rendes és teljes befejezéséig, így a napló kezelésével nő a tranzakciók átlagos pufferigénye.

Semmisségi/helyrehozó undo/redo) naplózás

További probléma (blokknál kisebb adatbáziselemek esetén): ellentétes pufferírási igények

(T1,COMMIT) lemezen van



(T2,COMMIT) nincs a lemezen

Például:

T1 az A-t módosította és befejeződött rendben

T2 a B-t módosította, de a COMMIT nincs a lemezen

Ekkor az R1 szabály miatt:

- **a puffert lemezre kell írni A miatt,**
- **a puffert nem szabad lemezre írni B miatt.**

Megoldás

- **A naplóbejegyzés négykomponensű:**
  - **a <T,X,v,w> naplóbejegyzés azt jelenti, hogy a T tranzakció az adatbázis X elemének korábbi v értékét w-re módosította.**

- **UR1: Mielőtt az adatbázis bármely X elemének értékét – valamely T tranzakció által végzett módosítás miatt – a lemezen módosítanánk, ezt megelőzően a <T,X,v,w> naplóbejegyzésnek lemezre kell kerülnie.**

- **WAL – Write After Log elv: előbb naplózunk, utána módosítunk**

- **NAGYOBB SZABADSÁG: A <T, COMMIT> bejegyzés megelőzheti, de követheti is az adatbáziselemek lemezen történő bármilyen megváltoztatását.**

- **NAGYOBB MÉRETŰ NAPLÓ: - régi és új értéket is tároljuk**

UNDO/REDO naplózás

Lépés	Tevékenység	t	M-A	M-B	D-A	D-B	Napló
1)							<T,START>
2)	READ(A,t)	8	8		8	8	
3)	t := t*2	16	8		8	8	
4)	WRITE(A,t)	16	16		8	8	<T,A,8,16>
5)	READ(B,t)	8	16	8	8	8	
6)	t := t*2	16	16	8	8	8	
7)	WRITE(B,t)	16	16	16	8	8	<T,B,8,16>
8)	FLUSH LOG						
9)	OUTPUT(A)	16	16	16	16	8	
10)							<T,COMMIT>
11)	OUTPUT(B)	16	16	16	16	16	

Megjegyzés: A <T,COMMIT> naplóbejegyzés kiírása kerülhetett volna a 9) lépés elé vagy a 11) lépés mögé is.

Helyreállítás UNDO/REDO naplózás esetén

A semmisségi/helyrehozó módszer alapelvei a következők:

1. **(REDO): A legkorábbitól kezdve állítsuk helyre minden befejezett tranzakció hatását.**
2. **(UNDO): A legutolsótól kezdve tegyük semmissé minden be nem fejezett tranzakció tevékenységeit.**

Megjegyzés: A COMMIT kötetlen helye miatt előfordulhat, hogy egy befejezett tranzakció néhány vagy összes változtatása még nem került lemezre, és az is, hogy egy be nem fejezett tranzakció néhány vagy összes változtatása már lemezen is megtörtént.

Helyreállítás UNDO/REDO naplózás esetén

Lépés	Tevékenység	t	M-A	M-B	D-A	D-B	Napló
1)							<T,START>
2)	READ(A,t)	8	8		8	8	
3)	t := t*2	16	8		8	8	
4)	WRITE(A,t)	16	16		8	8	<T,A,8,16>
5)	READ(B,t)	8	16	8	8	8	
6)	t := t*2	16	16	8	8	8	
7)	WRITE(B,t)	16	16	16	8	8	<T,B,8,16>
8)	FLUSH LOG						
9)	OUTPUT(A)	16	16	16	16	8	
10)							<T,COMMIT>
11)	OUTPUT(B)	16	16	16	16	16	

- Ha a katasztrófa a <T,COMMIT> naplóbejegyzés lemezre írása után történik:
- T-t befejezett tranzakciónak tekintjük. 16-ot írunk mind A-ba, mind B-be.
- A-nak már 16 a tartalma, de B-nek lehet, hogy nem, aszerint, hogy a hiba a 11) lépés előtt vagy után következett be.

Helyreállítás UNDO/REDO naplózás esetén

Lépés	Tevékenység	t	M-A	M-B	D-A	D-B	Napló
1)							<T,START>
2)	READ(A,t)	8	8		8	8	
3)	t := t*2	16	8		8	8	
4)	WRITE(A,t)	16	16		8	8	<T,A,8,16>
5)	READ(B,t)	8	16	8	8	8	
6)	t := t*2	16	16	8	8	8	
7)	WRITE(B,t)	16	16	16	8	8	<T,B,8,16>
8)	FLUSH LOG						
9)	OUTPUT(A)	16	16	16	16	8	
10)							<T,COMMIT>
11)	OUTPUT(B)	16	16	16	16	16	

- Ha a katasztrófa a <T,COMMIT> naplóbejegyzés lemezre írását megelőzően, a 9) és 10) lépések között következett be:
- T befejezetlen tranzakció: Ekkor A és B korábbi értéke, 8 íródik lemezre. Az A értéke már 16 volt a lemezen, és emiatt a 8-ra való visszaállítás feltétlenül szükséges. A B értéke nem igényelne visszaállítást, de nem lehetünk biztosak benne, így végrehajtjuk.

Helyreállítás UNDO/REDO naplózás esetén

Lépés	Tevékenység	t	M-A	M-B	D-A	D-B	Napló
1)							<T,START>
2)	READ(A,t)	8	8		8	8	
3)	t := t*2	16	8		8	8	
4)	WRITE(A,t)	16	16		8	8	<T,A,8,16>
5)	READ(B,t)	8	16	8	8	8	
6)	t := t*2	16	16	8	8	8	
7)	WRITE(B,t)	16	16	16	8	8	<T,B,8,16>
8)	FLUSH LOG						
9)	OUTPUT(A)	16	16	16	16	8	
10)							<T,COMMIT>
11)	OUTPUT(B)	16	16	16	16	16	

- Ha a katasztrófa a <T,COMMIT> naplóbejegyzés lemezre írását megelőzően, a 9) lépés előtt következett be:
- T befejezetlen tranzakció: Az A és B korábbi értéke, 8 íródik lemezre. (Most A és B sem igényelné a visszaállítást, de mivel nem lehetünk biztosak abban, hogy a visszaállítás szükséges-e vagy sem, így azt (a biztonság kedvéért) mindig végre kell hajtanunk.)

Helyreállítás UNDO/REDO naplózás esetén

- **Probléma (befejezett változtatást is megsemmisítünk): Az UNDO naplózáshoz hasonlóan most is előfordulhat, hogy a tranzakció a felhasználó számára korrekten befejezettnek tűnik, de még a <T,COMMIT> naplóbejegyzés lemezre kerülése előtt fellépett hiba utáni helyreállítás során a rendszer a tranzakció hatásait semmissé teszi ahelyett, hogy helyreállította volna.**
- **Amennyiben ez a lehetőség problémát jelent, akkor a semmisségi/helyrehozó naplózás során egy további szabályt célszerű bevezetni:**
- ***UR2*: A <T,COMMIT> naplóbejegyzést nyomban lemezre kell írni, amint megjelenik a naplóban.**
- **Ennek teljesítéséért a fenti példában a 10) lépés (<T,COMMIT>) után egy FLUSH LOG lépést kell beiktatnunk.**

Helyreállítás UNDO/REDO naplózás esetén

- **Konkurencia problémája:**

  Előfordulhat, hogy a T tranzakció rendben és teljesen befejeződött, és emiatt helyreállítása során egy X adatbáziselem T által kialakított új értékét rekonstruáljuk, melyet viszont egy be nem fejezett, és ezért visszaállítandó U tranzakció korábban módosított, ezért vissza kellene állítani az X régi értékét.

  A probléma nem az, hogy először helyreállítjuk X értékét, és aztán állítjuk vissza U előttire, vagy fordítva. Egyik sorrend sem helyes, mert a végső adatbázisállapot nagy valószínűséggel így is, úgy is inkonzisztens lesz.

  A konkurenciakezelésnél fogjuk megoldani, hogyan biztosítható T és U elkülönítése, amivel az ugyanazon X adatbáziselemen való kölcsönhatásuk elkerülhető.

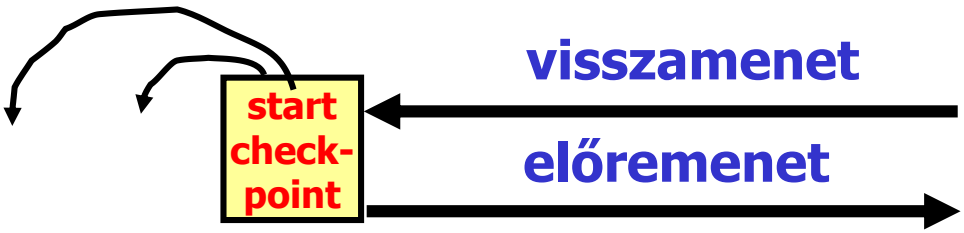
Semmisségi/helyrehozó naplózás ellenőrzőpont-képzéssel

- Egyszerűbb, mint a másik két naplózás esetén.

1. Írjunk a naplóba <START CKPT(T1,…,Tk)> naplóbejegyzést, ahol T1,…,Tk az aktív tranzakciók, majd írjuk a naplót lemezre.
2. Írjuk lemezre az összes piszkos puffert, tehát azokat, melyek egy vagy több módosított adatbáziselemet tartalmaznak. A helyrehozó naplózástól eltérően itt az összes piszkos puffert lemezre írjuk, nem csak a már befejezett tranzakciók által módosítottakat.
3. Írjunk <END CKPT> naplóbejegyzést a naplóba, majd írjuk a naplót lemezre.

Helyreállítás:

- **Visszamenet** (napló végétől ➲ az utolsó érvényes checkpoint kezdetéig)
  - meghatározzuk a befejezett tranzakciók S halmazát
  - megsemmisítjük azoknak a tranzakciók hatását, amelyek nincsenek S-ben
- **Megsemmisítjük a függő tranzakciókat**
  - visszamegyünk azokon a tranzakciókon, amelyek (checkpoint aktív tranzakciói) – S halmazban vannak
- **Előremenet** (az utolsó checkpoint kezdetétől ➲ a napló végéig)
  - helyrehozzuk az S tranzakcióinak hatását



Semmisségi/helyrehozó naplózás ellenőrzőpont-képzéssel

1. <T1, START>
2. <T1,A,4,5>
3. <T2, START>
4. <T1,COMMIT>
5. <T2,B,9,10>
6. <START CKPT(T2)>
7. <T2,C,14,15>
8. <T3,START>
9. <T3,D,19,20>
10. <END CKPT>
11. <T2,COMMIT>
12. <T3,COMMIT>

- Lehetséges, hogy a T2 által B-nek adott új érték (10) lemezre íródik az <END CKPT> előtt, ami nem volt megengedve a helyrehozó naplózásban.
- Most lényegtelen, hogy ez a lemezre írás mikor történik meg. Az ellenőrzőpont képzése alatt biztosan lemezre írjuk B-t (ha még nem került oda), mivel minden piszkos (változásban érintett) puffert kiírunk lemezre.
- Hasonlóan A-t – melyet a befejezett T1 tranzakció alakított ki – is lemezre fogjuk írni, ha még nem került oda.

Semmisségi/helyrehozó naplózás ellenőrzőpont-képzéssel

1. <T1, START>
2. <T1,A,4,5>
3. <T2, START>
4. <T1,COMMIT>
5. <T2,B,9,10>
6. <START CKPT(T2)>
7. <T2,C,14,15>
8. <T3,START>
9. <T3,D,19,20>
10. <END CKPT>
11. <T2,COMMIT>
12. <T3,COMMIT>

Amikor olyan tranzakció hatásait állítjuk helyre, mint amilyen a T2 is, akkor a naplóban nem kell a <START CKPT(T2)> bejegyzésnél korábbra visszatekinteni, mert tudjuk, hogy a T2 által az ellenőrzőpont-képzést megelőzően elvégzett módosítások az ellenőrzőpont képzése alatt lemezre íródtak.

→ **KATASZTRÓFA**

- T2-t és T3-at teljesen és rendesen befejezett tranzakciónak tekintjük.
- A T1 tranzakció az ellenőrzőpontnál korábbi. Minthogy <END CKPT> bejegyzést találunk a naplóban, így T1-ről biztosan tudjuk, hogy teljesen és rendesen befejeződött, valamint az általa elvégzett módosítások lemezre íródtak. Ezért a T2 és T3 által végrehajtott módosítások helyreállítandók, T1 pedig figyelmen kívül hagyható.

Semmisségi/helyrehozó naplózás ellenőrzőpont-képzéssel

1. <T1, START>
2. <T1,A,4,5>
3. <T2, START>
4. <T1,COMMIT>
5. <T2,B,9,10>
6. <START CKPT(T2)>
7. <T2,C,14,15>
8. <T3,START>
9. <T3,D,19,20>
10. <END CKPT>
11. ~~<T2,COMMIT>~~
12. <T3,COMMIT>

KATASZTRÓFA

Ha T3 az ellenőrzőpont-képzés előtt már aktív tranzakció lett volna, akkor a naplóban a START CKPT bejegyzésben szereplő befejezetlen tranzakciók közül a legkorábban elindult Ti tranzakció <Ti,START> bejegyzéséig kellene visszakeresnünk, hogy megtaláljuk a Ti (most T2 vagy T3) semmissé teendő tevékenységeit leíró naplóbejegyzéseket. A helyrehozó lépést viszont most is elég a START CKPT bejegyzéstől végrehajtani.

- Ekkor T2-t befejezett, T3-at pedig befejezetlen tranzakciónak kell tekintenünk.
- T2 tevékenységét helyreállítandó C értékét a lemezen 15-re írjuk; B-t már nem kell 10-re írnunk a lemezen, mert tudjuk, hogy ez már lemezre került az <END CKPT> előtt.
- A helyreállító naplózástól eltérően T3 hatásait semmissé tesszük, azaz a lemezen D tartalmát 19-re írjuk.

Semmisségi/helyrehozó naplózás ellenőrzőpont-képzéssel

1. <T1, START>
2. <T1,A,4,5>
3. <T2, START>
4. <T1,COMMIT>
5. <T2,B,9,10>
6. <START CKPT(T2)>
7. <T2,C,14,15>
8. <T3,START>
9. <T3,D,19,20>
10. <END CKPT>
11. <T2,COMMIT>
12. <T3,COMMIT>

KATASZTRÓFA

- Ha a katasztrófa az <END CKPT> bejegyzés előtt lép fel, akkor figyelmen kívül hagyjuk az utolsó START CKPT bejegyzést, és az előzőek szerint járunk el.

Undo/Redo naplózás összefoglalása

1. Undo/Redo naplózás, UR1 szabály, WAL, (Commit bárhol lehet)
2. Helyreállítás algoritmusa (naplóban két irányú mozgás)
3. UR2 szabály
4. Ellenőrző pont képzése (összes piszkos puffer kiírása)