a)

UNDO naplózás

Legyen T, T2 és T3 egy tranzakció.

<T START> Elindult a tranzakció.

<T adat régi_érték> feljegyezzük a módosított adat régi értékét.

<T COMMIT> Lezártuk a tranzakciót, nem kell vele többet foglalkoznunk

<T ABORT> Visszaállítottuk a T tranzakciót a leállás előtti állapotába és nem kell vele többet foglalkoznunk.

<START CKPT(T2,T3)> Minden olyan tranzakció, amely még ezelőtt az ellenőrző pont előtt megkezdődött és nem szerepel az ellenőrző pont listájában lezárult vagy COMMITtal vagy ABORTtal, tehát nem kell velük foglalkoznunk. Az ellenőrzőpont listájában szerepelnek az akutális időpillanatban létező aktív tranzakciók.

<END CKPT> Az ellenőrző pont listájában szereplő tranzakciók is lezárultak vagy COMMITtal vagy ABORTtal, tehát velük sem kell már foglalkoznunk.

REDO naplózás

Legyen T, T2 és T3 egy tranzakció.

<T START> Elindult a tranzakció.

<T adat új_érték> Feljegyezzük, hogy milyen új értékre módosítjuk az adott adatot.

<T COMMIT> A T tranzakció nem fog több adatot módosítani.

<T END> Lezártuk a tranzakciót, nem kell már vele foglalkoznunk.

<T ABORT> Ha T tranzakciónak nincs COMMITja, akkor önmagában meghívjuk, és nem kell vele többet foglalkoznunk.

<START CKPT(T2,T3)> Minden olyan tranzakció, amely még ezelőtt az ellenőrző pont előtt megkezdődött, rendelkezik COMMITtal. Ha a rendszer ezen ellenőrzőpont vége (<END CKPT>) előtt állna le, akkor minden, az ellenőrzőpont előtt megkezdett, annak listájában nem szereplő és ENDdel nem rendelkező tranzakciót REDO-zni kell. Az ellenőrzőpont listájában szerepelnek az akutális időpillanatban létező aktív tranzakciók.

<END CKPT> Minden, az ellenőrző pont előtt megkezdődött, és annak listájában nem szereplő tranzakció befejeződött és nem kell már velük foglalkoznunk.

UNDO és REDO közti különbségek

Az UNDO a COMMITtal lezár egy tranzakciót, a REDO csak jelzi, hogy az adott tranzakció már nem fog módosításokat végezni.

Az UNDO naplózásban nincs szükség ENDre, az csak a REDOban kell a már COMMITtal rendelkező tranzakciók lezárására.

Az UNDO naplózás a legújabban naplózott tranzakciótól visszafele haladva végzi a visszaállítást, a REDO pedig a legrégebbi tranzakciótól előrefele halad.

A visszaállítást az UNDO egy ABORTtal, a REDO egy ENDdel zárja le. A REDO akkor használja az ABORTot, amikor olyan tranzakciót lát, amelynek nincs COMMITja. Ez esetben nem állítottunk vissza semmit.

UNDO példa:

<T START>

<T,F,10>

<START U>

<T,G,12>

<U,H,17>

<U,I,19>

<T COMMIT>

A T-t lezártuk, az U-t vissza kell állítani.

<U,I,19>

<U,H,17>

<U ABORT>

REDO példa:

<T START>

<T,F,10>

<START U>

<T,G,12>

<U,H,17>

<X START>

<X,J,22>

<U,I,19>

<T COMMIT>

<U COMMIT>

<T END>

T-nek van ENDje, ezért vele semmit sem kell csinálni, X-nek nincs COMMITja, ezért őt abortálni kell és U-nak van COMMITja, de nincs ENDje, ezért REDO-zni kell

<U,H,17>

<U,I,19>

<U END>

<X ABORT>

UNDO: A legújabb naplóbejegyzéstől visszafelé haladva a legrégebbi, még le nem zárt naplóbejegyzásig történik a lemezre írás.

REDO: A legrégebbi, még le nem zárt naplóbejegyzéstől előrefelé haladva történik a lemezre írás.

Módosított REDO: A legrégebbi, még le nem zárt naplóbejegyzéstől előrefelé haladva történik a lemezre írás.

b)

U1: Ha a *T* tranzakció módosítja az *X* adatbáziselemet, akkor a (*T*, *X*, régi érték) naplóbejegyzést azelőtt kell a lemezre írni, mielőtt az *X* új értékét a lemezre írná a rendszer.

U2: Ha a tranzakció hibamentesen befejeződött, akkor a COMMIT naplóbejegyzést cask azután szabad a lemezre írni, ha a tranzakció által módosított összes adatbáziselem már a lemezre íródott, de ezután rögtön.

R1: Mielőtt az adatbázis bármely X elemét a lemezen módosítanánk, az X módosítására vonatkozó összes naplóbejegyzésnek, azaz <T,X,v>-nek és <T, COMMIT>-nak a lemezre kell kerülnie.

c)

Az UNDO a COMMITtal befejez egy tranzakciót, a REDO csak lezárja azt. Ilyenkor az UNDO esetén az adott tranzakcióval már semmit sem kell csinálnunk, a REDO esetén még helyre kell azt hoznunk és a végére END-et tenni.

d)

UNDO: Összegyűjtjük azokat a tranzakciókat, amelyeknek nincs sem commitjuk, sem abortjuk. Ekkor a legújabbtól a legrégebbi naplóbejegyzésig egyesével visszaírjuk a módosított változókba a régi értéket és a végén mindegyik tranzakciót abortáljuk egymás után.

REDO: Összegyűjtjük azokat a tranzakciókat, amelyeknek van commitja, de nincs endje. Végig megyünk ezen tranzakciók módosításain a legrégebbitől a legújabbig és módosítjuk az adott változó értékét az új értékre. Ha kész vagyunk, akkor mindegyik tranzakcióra meghívjuk az endet.

Módosított REDO: A be nem fejezett tranzakciókat abortoljuk. A committal rendelkező, de enddel nem rendelkező tranzakciók módosításait a legrégebbitől a legújabbig újra elvégezzük, azaz az adott X adatbáziselembe beleírjuk az új értéket. A végén a naplót kiírjuk a lemezre (FLUST LOG).

e)

UNDO

1. Az ellenőrző pont létrehozásakor bele írjuk annak listájába azon megkezdett tranzakciók neveit, amelyek nem fejeződtek be. <START CKPT(T1,…,Tk)> Ezt a lemezre is kiírjuk (FLUSH LOG).

2. Megvárjuk, amíg a T1,…,Tk tranzakciók mindegyike befejeződik akár committal, akár aborttal. Eközben megengedjük, hogy új tranzakciók is induljanak.

3. <END CKPT>-vel jelezzük, ha mindegyik tranzakció (T1,…,Tk) befejeződött és ezt lemezre is írjuk (FLUSH LOG).

REDO

1. A <START CKPT(T1,...,Tk)> naplóbejegyzés elkészítésével, majd lemezre írásával feljegyezzük az összes aktív tranzakciót (T1,...,Tk).

2. Összegyűjtjük az összes nem aktív, be nem fejezett tranzakció módosításait, amelyek már befejeződtek, de még nincsenek a lemezen a puffereik. Ezeknek a tranzakcióknak van commitjuk, de nincs endjük.

3. <END CKPT>-t írunk a naplóba, illetve a lemezre, ha a 2-es pontban foglaltak megtörténtek.

f)

UNDO

1. <T0 START>
2. <T1 START>
3. <T0,A,31>
4. <T0 COMMIT>
5. <T2 START>
6. <T1,B,42>
7. <T2,C,77>
8. <T3 START>
9. <T3,D,54>
10. <START CKPT(T2,T3)>
11. <T2,E,22>
12. <T4 START>
13. <T3,F,67>
14. <END CKPT>
15. <T4,G,69>
16. <T4,H,94>

I) Például ha a hiba a 9. és 10. sor között történik, akkor a 4. sorból tudjuk, hogy a T0 már befejeződött, UNDO-zni kell a T1, T2 és T3 tranzakciókat, mert ők nem fejeződtek be sem committal, sem aborttal.

<T3,D,54>

<T2,C,77>

<T1,B,42>

<T3 ABORT>

<T2 ABORT>

<T1 ABORT>

II) Például ha a hiba a 16. sor után történik, akkor a 10. sor miatt tudjuk, hogy T0 és T1 befejeződött vagy committal vagy aborttal. A 14. sor miatt tudjuk, hogy T2 és T3 is befejeződött vagy committal vagy aborttal. A T4-es tranzakciót UNDO-zni kell.

<T4,H,94>

<T4,G,69>

<T4 ABORT>

III) Például ha a hiba a 13. és a 14. sor között történik, akkor a 9. sor miatt tudjuk, hogy T0 és T1 befejeződött vagy committal vagy aborttal. A T2 és T3-as tranzakciót UNDO-zni kell.

<T3,F,67>

<T2,E,22>

<T3,D,54>

<T2,C,77>

<T2 ABORT>

<T3 ABORT>

REDO

1. <T0 START>
2. <T1 START>
3. <T0,A,31>
4. <T0,X,11>
5. <T0 COMMIT>
6. <T2 START>
7. <T1,B,42>
8. <T2,C,77>
9. <T3 START>
10. <T3,D,54>
11. <START CKPT(T2,T3)>
12. <T2,E,22>
13. <T2 COMMIT>
14. <T4 START>

15. <T3,F,67>

16. <END CKPT>

17. <T4,G,69>

18. <T4,H,94>

I) Például ha a hiba a 10. és 11. sor között történik, akkor a 4. sor miatt tudjuk, hogy T0-t REDO-zni kell, T1-et, T2-t és T3-at pedig abortálni.

<T0,A,31>

<T0,X,11>

<T0 END>

<T1 ABORT>

<T2 ABORT>

<T3 ABORT>

II) Például ha a hiba a 18. sor után történik, akkor a 15. sor miatt tudjuk, hogy T0 és T1 befejeződött. A 13. sor miatt tudjuk, hogy T2-t REDO-zni kell. T3-at és T4-et abortálni kell, mert nincs sem commitjuk, sem abortjuk.

<T2,C,77>

<T2,E,22>

<T2 END>

<T3 ABORT>

<T4 ABORT>

III) Például ha a hiba a 15. és 16. sor között történik, akkor a 11. sor miatt az összes, az ellenőrzőpont listájában már nem szereplő tranzakciót (T0, T1) REDO-zni kell mivel azoknak már van commitjuk. A 13. sor miatt T2-t is REDO-zni kell, T3-at és T4-et pedig abortálni kell.

<T0,A,31>

<T0,X,11>

<T1,B,42>

<T2,C,77>

<T2,E,22>

<T0 END>

<T1 END>

<T2 END>

<T3 ABORT>

<T4 ABORT>