## 8.3. Helyrehozó naplózás (redo logging)

A semmisségi naplózás (undo logging) természetes és egyszerű stratégiát valósít meg a napló kezelésére és a rendszerhibák esetén való visszaállításra, de a probléma megoldásának nem ez az egyetlen lehetséges megközelítése. A semmisségi naplózás potenciális problémája az, hogy csak azután tudjuk befejezni a tranzakciót, ha az összes adatbázis-módosításai már lemezre íródtak. Olykor a lemezműveletekkel tudnánk takarékoskodni, ha megengednénk, hogy az adatbázis-módosításokat csak a memóriában végezzék a tranzakciók, miközben a napló az eseményeket rögzíti, azért, hogy katasztrófa esetében is biztonságban legyen az adatbázis.

Az adatbáziselemek lemezre való azonnali visszaírásának kényszerét elkerülhetjük, ha a *helyrehozó naplózás* (redo logging) módszerét választjuk. Az alapvető különbségek a semmisségi és a helyrehozó naplózás között az alábbiak:

1. Amíg a semmisségi naplózás a helyreállítás során a be nem fejezett tranzakciók hatásait semmissé teszi, a befejezett tranzakciók hatásait pedig nem módosítja, addig a helyrehozó naplózás figyelmen kívül hagyja a be nem fejezett tranzakciókat, és megismétli a normálisan befejezettek által végrehajtott változtatásokat.

2. A semmisségi naplózás megkívánja az adatbáziselemek lemezen való módosítását a COMMIT naplóbejegyzés lemezre írása előtt, addig a helyrehozó naplózás a COMMIT naplóbejegyzés lemezre írását várja el, mielőtt bármit is változtatna a lemezen lévő adatbázisban.

3. A semmisségi naplózás *U*1 és *U*2 szabályainak betartása mellett csak a módosított adatbáziselemek régi tartalmát kell megőriznünk az esetleges visszaállítás biztosításához, a helyrehozó naplózással történő helyreállításhoz a módosított elemek új értékére van szükség. Emiatt a helyrehozó naplózás naplóbejegyzései ugyanolyan formájúak, de más a jelentésük, mint a semmisségi naplózásnál alkalmazottaké.

### 8.3.1. A helyrehozó naplózás szabályai

A helyrehozó naplózás az adatbáziselemek módosítását a naplóbejegyzésben az új értékkel képviseli (nem pedig a régivel, mint a semmisségi naplózásnál). Ez a bejegyzés ugyanúgy néz ki, mint a semmisségi naplózásnál használt: <*T*,*X*,*v*>, a jelentése azonban más. E bejegyzés jelentése: „a *T* tranzakció az *X* adatbáziselemnek a *v* értéket adta”. E bejegyzésben az *X* régi értékét nem jelzi semmi. Mindig, ha a *T* tranzakció módosítja az *X* adatbáziselem értékét, akkor egy <*T*,*X*,*v*> bejegyzést kell a naplóba írni.

Annak sorrendjét, hogy az adat- és naplóbejegyzések hogyan kell hogy lemezre kerüljenek, az alábbi egyszerű „helyrehozó naplózási szabály”, az úgynevezett *írj korábban naplózási szabály* írja le.

*R*1: Mielőtt az adatbázis bármely *X* elemét a lemezen módosítanánk, szükséges, hogy az *X* ezen módosítására vonatkozó összes naplóbejegyzése, azaz <*T*,*X*,*v*> és <COMMIT *T*>, a lemezre kerüljenek.

Minthogy a COMMIT bejegyzést csak akkor írhatjuk a naplóba, ha a tranzakció teljesen és hibamentesen befejeződött, így a COMMIT bejegyzés csak a módosításokat leíró bejegyzések után állhat, ezért úgy is összegezhetjük az *R*1 szabálya hatását, hogy: ha helyrehozó naplózást használunk, akkor az egy tranzakcióra vonatkozó lemezre írásoknak a következő sorrendben kell megtörténniük:

1. Az adatbáziselemek módosítását leíró naplóbejegyzések lemezre írása.

2. A COMMIT naplóbejegyzés lemezre írása.

3. Az adatbáziselemek értékének tényleges cseréje a lemezen.

**8.7. példa:** Tanulmányozzuk ugyanazt a tranzakciót, amelyiket a 8.3. példában is elemeztünk. A 8.7. ábrán látható ezen tranzakcióra vonatkozó események lehetséges sorrendje.

A főbb különbségek a 8.7. és 8.3. ábrák között a következők: először nézzük a 8.7. ábra 4) és 7) sorait, ezekben a módosítást leíró naplóbejegyzésben az *A* és *B* adatbáziselemek új értéke szerepel (s nem a régi, mint a 8.3. ábrán). A másik különbség, hogy a COMMIT bejegyzés korábbra került, a 8) lépésbe. Ezt követően a napló lemezre írását kiváltó FLUSH LOG következik, s így a *T* tranzakció által végrehajtott módosításokat leíró összes naplóbejegyzés lemezre íródik. Csak ezt követően kerül lemezre az *A* és *B* új, módosított értéke. Az ábrán ezen új értékek kiírását a közvetlenül következő 10) és 11) sorokban láthatjuk, bár a gyakorlatban ezekre esetleg csak később kerül sor. 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lépés | Tevékenység | t | M-A | M-B | D-A | D-B | Napló |
| 11) |  |  |  |  |  |  | <START *T*> |
| 12) | READ(A,t) | 18 | 18 |  | 18 | 18 |  |
| 13) | t := t\*2 | 16 | 18 |  | 18 | 18 |  |
| 14) | WRITE(A,t) | 16 | 16 |  | 18 | 18 | <*T*,*A*,16> |
| 15) | READ(B,t) | 18 | 16 | 18 | 18 | 18 |  |
| 16) | t := t\*2 | 16 | 16 | 18 | 18 | 18 |  |
| 17) | WRITE(B,t) | 16 | 16 | 16 | 18 | 18 | <*T*,*B*,16> |
| 18) |  |  |  |  |  |  | <COMMIT *T*> |
| 19) | FLUSH LOG |  |  |  |  |  |  |
| 10) | OUTPUT(A) | 16 | 16 | 16 | 16 | 18 |  |
| 11) | OUTPUT(B) | 16 | 16 | 16 | 16 | 16 |  |

**8.7. ábra.** Tevékenységek és naplóbejegyzéseik helyrehozó naplózás használatakor

### 8.3.2. Helyreállítás a helyrehozó naplózás használatával

A helyrehozó naplózás *R*1 szabályának fontos következménye, hogy ha a naplóban nincs <COMMIT *T*> bejegyzés, akkor tudjuk, hogy a *T* tranzakció nem hajtott végre az adatbázisban módosítást a lemezen. Így a be nem fejezett (nem teljes) tranzakciók a helyreállítás során úgy tekinthetők, mintha meg sem történtek volna. Problémát a befejezett (COMMIT) tranzakciók jelenthetnek, mert nem tudjuk, hogy az általuk elvégzett adatbázis-változtatások közül melyik íródott már lemezre. Szerencsére a helyrehozó naplózás naplója éppen azon információkat – az új értékeket – tartalmazza, melyekre szükségünk van a helyreállításhoz. Ezen új értékeket kell lemezre írnunk, attól függetlenül, hogy esetleg már korábban is kiíródtak. A rendszerkatasztrófa bekövetkezése után a helyrehozó naplózással történő helyreállításhoz a következőket kell tennünk:

1. Meghatározni a befejezett (COMMIT) tranzakciókat.

2. Elemezni a naplót az elejétől kezdve. Minden <*T*,*X*,*v*> naplóbejegyzés megtalálásakor:

a) Ha *T* nem befejezett tranzakció, akkor nem kell tenni semmit.

b) Ha *T* befejezett tranzakció, akkor *v* értéket kell az *X* adatbáziselembe írni.

3. Minden *T* be nem fejezett tranzakcióra vonatkozóan <ABORT *T*> naplóbejegyzést kell a naplóba írni, és a naplót ki kell írni lemezre (FLUSH LOG).

**8.8. példa:** Tegyük fel, hogy a napló a 8.7. ábrának megfelelő, nézzük meg hogyan lehet a helyreállítást elvégezni a különböző pillanatokban bekövetkező katasztrófák esetében.

1. Ha a katasztrófa a 9) lépés után bármikor következik be, akkor a <COMMIT *T*> bejegyzés már lemezen van. A helyreállító rendszer *T*-t befejezett tranzakcióként azonosítja. Amikor a naplót az elejétől kezdve elemzi, a <*T*,*A*,16> és a <*T*,*B*,16> bejegyzések hatására a helyreállítás-kezelő az *A* és *B* adatbáziselemekbe a 16 értéket írja. Megjegyezzük, hogy ha a katasztrófa a 10) és 11) lépések között következett be, akkor *A* újraírása redundáns ugyan, de *B* írása (korában nem történt meg) lényeges lépés az adatbázis konzisztens állapotának eléréséhez. Amennyiben a hiba a 11) lépést követően keletkezett, akkor mindkét adatbáziselem új értékének lemezre írása redundáns ugyan, de semmi gondot nem okoz.

2. Ha a hiba a 8) és 9) lépések között jelentkezik, akkor bár a <COMMIT *T*> bejegyzés már a naplóba került, de nem biztos, hogy lemezre íródott (ez attól függ, hogy esetleg valami más okból sor került-e a napló lemezre írására). Ha lemezre került, akkor a helyreállítási eljárás az 1) esetnek megfelelően történik. Ha pedig a napló még nem került lemezre, akkor a helyreállítás a következő, 3) esettel megegyező.

3. Ha a katasztrófa a 8) lépést megelőzően keletkezik, akkor <COMMIT *T*> naplóbejegyzés még biztosan nem került lemezre, így *T* be nem fejezett tranzakciónak tekintendő. Ennek megfelelően *A* és *B* értékeit a lemezen még nem változtatta meg a *T* tranzakció, nincs mit helyreállítani, s végül egy <ABORT *T*> bejegyzést írunk a naplóba.

### 8.3.3. Helyrehozó naplózás ellenőrzőpont-képzés használatával

A semmisségi naplózásnál látottakhoz hasonlóan a helyrehozó naplózás naplójába is illeszthetünk ellenőrzőpontokat. A helyrehozó naplózásnál azonban új probléma jelentkezik: Minthogy a befejeződött tranzakciók módosításainak lemezre írása a befejeződés után sokkal később is történhet, így az e vonatkozásban ugyanazon pillanatban aktív tranzakciók számát nem tudjuk korlátozni, azon pillanatban sem, amikor az ellenőrzőpont létrehozásáról döntünk. Tekintet nélkül arra, hogy az ellenőrzőpont-képzés alatt tranzakciók indulását megengedjük vagy sem, a kulcsfeladat – amit meg kell tennünk az ellenőrzőpont-készítés kezdete és befejezése közötti időben – azon összes adatbáziselem lemezre való kiírása, melyeket befejezett tranzakciók módosítottak, és még nem voltak lemezre kiírva. Ennek megvalósításához a pufferkezelőnek nyilván kell tartania a *piszkos* puffereket, melyekben már végrehajtott, de lemezre még ki nem írt módosításokat tárol. Azt is tudnunk kell, mely tranzakciók mely puffereket módosították.

Más oldalról viszont, be tudjuk fejezni az ellenőrzőpont-képzést az aktív tranzakciók (normális vagy abnormális) befejezésének kivárása nélkül, mert ők ekkor még amúgy sem engedélyezik lapjaik lemezre írását. A helyrehozó naplózásban a működés közbeni ellenőrzőpont-képzés a következőkből áll:

1. <START CKPT (*T*1, …, *Tk*)> naplóbejegyzés elkészítése és kiírása lemezre, ahol
*T*1, …, *Tk* az összes éppen aktív (még be nem fejezett) tranzakció.

2. Az összes olyan adatbáziselem kiírása lemezre, melyeket olyan tranzakciók írtak pufferekbe, melyek a START CKPT naplóba írásakor már befejeződtek, de puffereik lemezre még nem kerültek.

3. <END CKPT> bejegyzés naplóba írása és a napló lemezre írása (FLUSH LOG).

<START *T*1>

<*T*1,*A*,5>

<START *T*2>

<COMMIT *T*1>

<*T*2,*B*,10>

<START CKPT (*T*2)>

<*T*2,*C*,15>

<START *T*3>

<*T*3,*D*,20>

<END CKPT>

<COMMIT *T*2>

<COMMIT *T*3>

**8.8. ábra.** A helyrehozó naplózás naplója

**8.9. példa:** A 8.8. ábra egy lehetséges naplót mutat, melynek közepén ellenőrzőpont található. Amikor az ellenőrzőpont-képzés elkezdődött, csak *T*2 volt aktív, de a *T*1 által *A*-ba írott érték még csak esetleg került lemezre. Ha még nem, akkor *A*-t lemezre kell másolnunk, mielőtt az ellenőrzőpont-képzést befejezhetnénk. A napló érzékelteti, hogy az ellenőrzőpont-képzés befejezéséig más események is bekövetkezhetnek: *T*2 a *C* adatbáziselem tartalmát módosítja, elindul *T*3 új tranzakció, és módosítja *D* értékét. Az ellenőrzőpont-képzés befejezése után már csak *T*2 és *T*3 tranzakciók befejeződése történt meg. 

### 8.3.4. Visszaállítás az ellenőrzőponttal kiegészített helyrehozó típusú naplózással

Mint a semmisségi naplózásnál, most is, az ellenőrzőpontok naplóba illesztése segít a naplóátvizsgálás korlátozásában, amikor adatbázis-helyreállítás szükséges. Szintén a semmisségi naplózáshoz hasonlóan két eset fordulhat elő, attól függően, hogy az utolsó ellenőrzőpont-bejegyzés a START vagy az END.

• Tegyük fel először, hogy a katasztrófa előtt a naplóba feljegyzett utolsó ellenőrzőpont-bejegyzés <END CKPT>. Ekkor tudjuk, hogy az olyan értékek, melyeket olyan tranzakciók írtak, melyek a <START CKPT (*T*1, …, *Tk*)> naplóbejegyzés megtétele előtt befejeződtek, már biztosan lemezre kerültek, s így nem kell velük foglalkoznunk helyreállítandó ezen tranzakciók hatását. Foglalkoznunk kell viszont a *Ti*-k közé tartozó, valamint az ellenőrzőpont kialakításának megkezdése után induló tranzakciókkal, ezeknek lehetnek olyan adatbázis-módosításaik, melyek még nem kerültek lemezre, pedig a tranzakció már befejeződött. Ekkor olyan visszaállítást kell tennünk, amilyenről a 8.3.2. részben már szó volt, azzal a különbséggel, hogy figyelmünket azon tranzakciókra korlátozhatjuk, melyek az utolsó <START CKPT (*T*1, …, *Tk*)> naplóbejegyzésben a *Ti*-k között szerepelnek, vagy ezen naplóbejegyzést követően indultak el. A naplóban való keresés során a legkorábbi <START *Ti*> naplóbejegyzésig kell visszamennünk, annál korábbra már nem. Megjegyezzük, hogy ezek a START naplóbejegyzések akárhány korábbi ellenőrzőpontnál korábban is felbukkanhatnak. Ahogy a semmisségi naplózásnál is láttuk, az adott tranzakcióra vonatkozó naplóbejegyzések visszafelé keresése segít megtalálni a számunkra éppen fontos bejegyzéseket.

• Tegyük fel most, hogy a naplóba feljegyzett utolsó ellenőrzőpont-bejegyzés a <START CKPT (*T*1,…, *Tk*)> naplóbejegyzés. Nem lehetünk abban biztosak, hogy az ezt megelőzően befejezett tranzakciók által módosított adatbáziselemek már lemezre íródtak. Ezért az előző <END CKPT> bejegyzéshez tartozó <START CKPT (*S*1, …, *Sm*)> naplóbejegyzésig[[1]](#footnote-1) vissza kell keresnünk, és helyre kell állítanunk az olyan befejeződött tranzakciók tevékenységének eredményeit, melyek ez utóbbi <START CKPT (*S*1, …, *Sm*)> naplóbejegyzés után indultak, vagy az *Si*-k közül valók.

**8.10. példa:** Tekintsük ismét a 8.8. ábrán bemutatott naplót. Ha a katasztrófa a végén lép fel, akkor az <END CKPT> bejegyzésig kell visszakeresnünk. Ekkor tudjuk, hogy a helyreállítás szempontjából elegendő csak azon tranzakciókat figyelembe venni, melyek egyrészt a <START CKPT (*T*2)> bejegyzés felírását követően indultak, vagy szerepelnek e bejegyzés listájában (most csak *T*2). Így a vizsgálandó tranzakcióhalmazunk {*T*2, *T*3}. <COMMIT *T*2> és <COMMIT *T*3> bejegyzéseket találunk, s ebből tudjuk, hogy mindkettő tranzakció hatását helyre kell állítanunk. A naplóban visszafelé meg kell keresnünk a <START *T*2> bejegyzést, s innen már időrendben haladva a naplóban a következő – *T*2, *T*3 befejezett tranzakciókra vonatkozó – módosítást leíró bejegyzéseket találjuk: <*T*2,*B*,10>, <*T*2,*C*,15> és <*T*3,*D*,20>. Mivel azt nem tudjuk, hogy ezen változtatások a lemezen már megtörténtek-e, ezért most a lemezre újraírjuk a *B*, *C* és *D* tartalmát, megfelelően 10, 15 és 20 értékeket adva nekik.

Tegyük fel most, hogy a katasztrófa a <COMMIT *T*2> és <COMMIT *T*3> bejegyzések között történt. A helyreállítás az előbbi esethez hasonló, azzal a különbséggel, hogy *T*3 nem befejezett tranzakció, ennek megfelelően a <*T*3,*D*,20> helyreállítást *nem* kell végrehajtani. *D* értékét a helyreállítás során nem változtatjuk meg, hacsak a vizsgált naplórészben található, más tranzakció bejegyzése miatt meg nem kell változtatnunk. A helyreállítást követően egy <ABORT *T*3> bejegyzést írunk a naplóba.

Végül, ha a hiba az <END CKPT> bejegyzést megelőzően lépett fel, akkor az utolsó előtti START CKPT bejegyzést kell megkeresnünk (melynek már van <END CKPT> párja), és annak listájából tudjuk meg, melyek az aktív tranzakciók. Ha nem találunk korábbi ellenőrzőpont-bejegyzést, akkor mindenképpen a napló elejére kell mennünk. Így esetünkben az egyedüli befejezett tranzakciónak *T*1-et fogjuk találni, s ezért a <*T*1,*A*,5> tevékenységét helyreállítjuk. A helyreállítást követően <ABORT *T*2> és <ABORT *T*3> bejegyzéseket írunk a naplóba. 

Minthogy a tranzakciók több ellenőrzőpont készítésekor is aktívak lehetnek, célszerű lehet, hogy a <START CKPT (*T*1, …, *Tk*)> naplóbejegyzésbe nem csak az aktív tranzakciók neveit, hanem olyan mutatókat is elhelyezzünk, melyek az aktív tranzakciók indulását leíró bejegyzések naplóbeli helyét adják meg. Így eljárva, biztonsággal meg tudjuk állapítani, hogy a napló mely korábbi részeit törölhetjük. Amikor <END CKPT> bejegyzést írunk a naplóba, akkor tudjuk, hogy a naplóban már sosem kell korábbra visszatekintenünk, mint ahol a *Ti* aktív tranzakcióra vonatkozó, legkorábbi <START *Ti*> bejegyzést találjuk. Következésképpen az ezen START bejegyzést megelőző bejegyzések mindegyike törölhető.

1. Előzetes hiba miatt előfordulhat, hogy a START CKPT bejegyzésnek nincs <END CKPT> párja. Ezért kell úgy eljárnunk, hogy nem csak a korábbi START CKPT bejegyzést keressük, hanem előbb egy <END CKPT>-t, majd az ezt megelőző START CKPT-t. [↑](#footnote-ref-1)