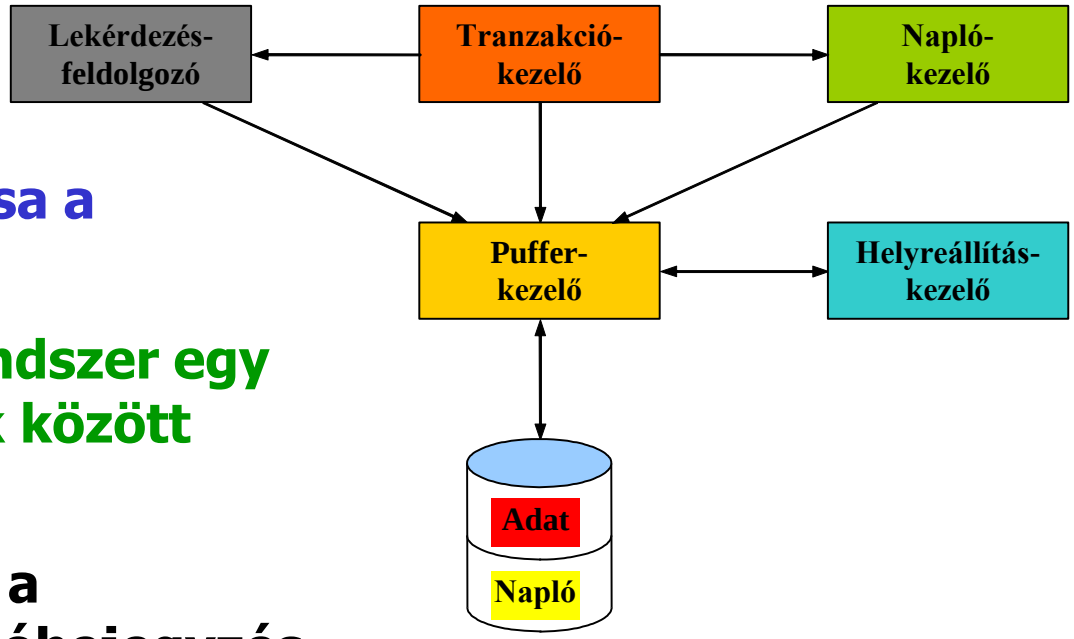
A naplókezelő és a tranzakciókezelő



A tranzakciók korrekt végrehajtásának biztosítása a tranzakciókezelő feladata.

A tranzakciókezelő részrendszer egy sor feladatot lát el, többek között

- **jelzéseket ad át a naplókezelőnek úgy, hogy a szükséges információ naplóbejegyzés formában a naplóban tárolható legyen;**
- **biztosítja, hogy a párhuzamosan végrehajtott tranzakciók ne zavarhassák egymás működését (ütemezés).**

A naplókezelő és a tranzakciókezelő

A tranzakciókezelő

1. a tranzakció tevékenységeiről üzeneteket küld a naplókezelőnek,
2. üzen a pufferkezelőnek arra vonatkozóan, hogy a pufferek tartalmát szabad-e vagy kell-e lemezre másolni,
3. és üzen a lekérdezésfeldolgozónak arról, hogy a tranzakcióban előírt lekérdezéseket vagy más adatbázis-műveleteket kell végrehajtania.

A naplókezelő és a tranzakciókezelő

A naplókezelő

1. **a naplót tartja karban,**
2. **együtt kell működnie a pufferkezelővel, hiszen a naplózandó információ elsődlegesen a memóriapufferekben jelenik meg, és bizonyos időnként a pufferek tartalmát lemezre kell másolni.**
3. **A napló (adat lévén) a lemezen területet foglal el.**

- **Ha baj van, akkor a helyreállítás-kezelő aktivizálódik. Megvizsgálja a naplót, és ha szükséges, a naplót használva helyreállítja az adatokat. A lemez elérése most is a pufferkezelőn át történik.**

A naplókezelő és a tranzakciókezelő

Azt mindig feltesszük, hogy a háttértár nem sérül, azaz csak a memória, illetve a puffer egy része száll el.

Az ilyen belső társérülés elleni védekezés két részből áll:

1. Felkészülés a hibára: naplózás

2. Hiba után helyreállítás: a napló segítségével egy konzisztens állapot helyreállítása

Természetesen a naplózás és a hiba utáni helyreállítás összhangban vannak, de van több különböző naplózási protokoll (és ennek megfelelő helyreállítás).

Adategység (adatbáziselem)

Feltesszük, hogy az adatbázis adategységekből, elemekből áll.

Az *adatbáziselem* (database element) a fizikai adatbázisban tártolt adatok egyfajta funkcionális egysége, amelynek értékét tranzakciókkal lehet elérni (kiolvasni) vagy módosítani (kiírni).

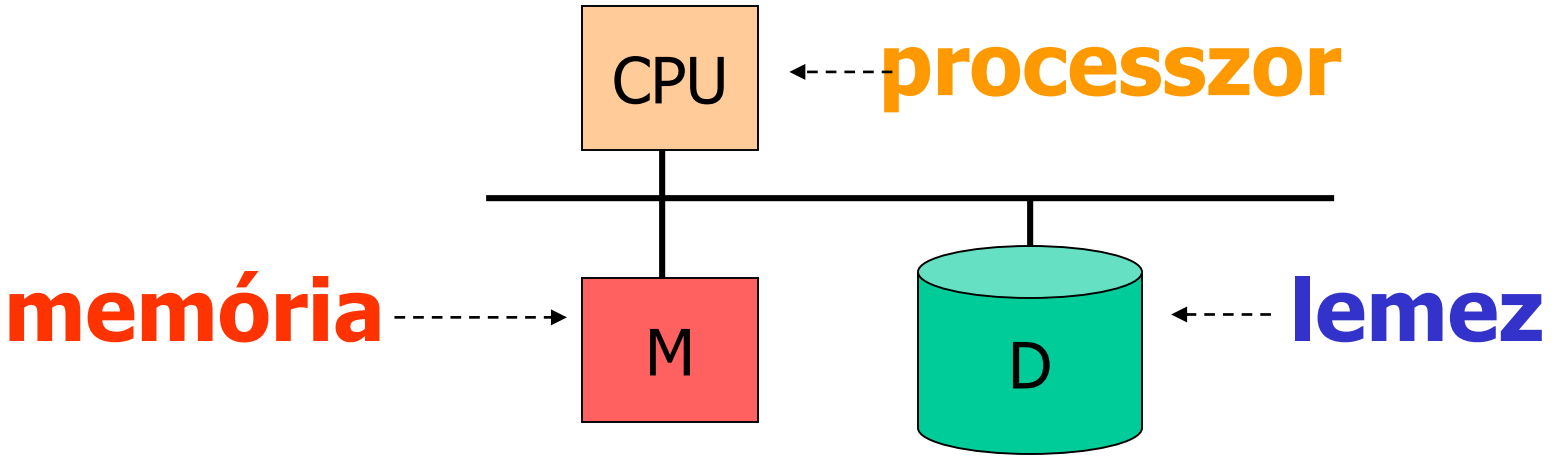
Az adatbáziselem lehet:

- **reláció (vagy OO megfelelője, az osztálykiterjedés),**
- **relációsor (vagy OO megfelelője, az objektum)**
- **lemezblokk**
- **lap**

Ez utóbbi a legjobb választás a naplózás szempontjából, mivel ekkor a puffer egyszerű elemekből fog állni, és ezzel elkerülhető néhány súlyos probléma, például amikor az adatbázis valamely elemének egy része van csak a nem illékony memóriában (a lemezen).

A vizsgált meghibásodási modell

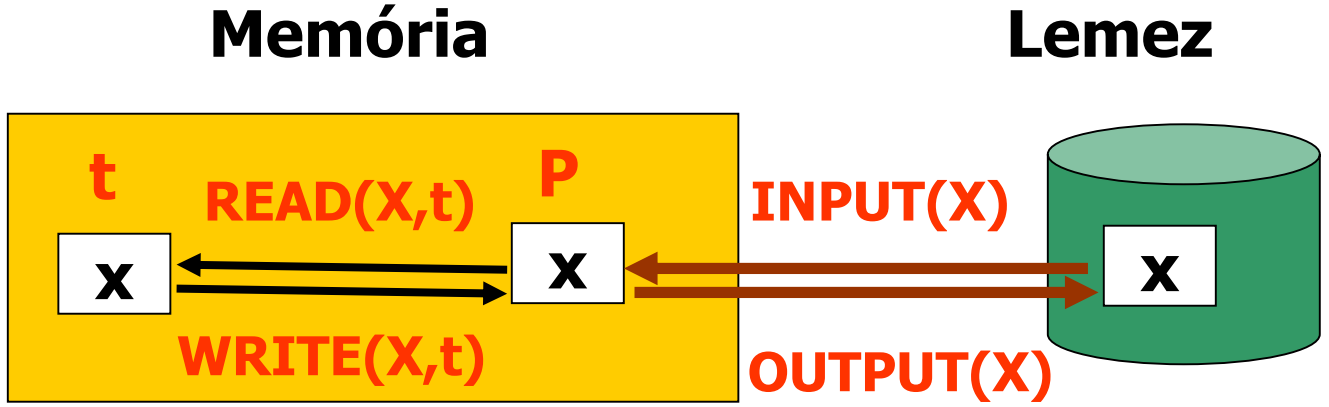
A tranzakció és az adatbázis kölcsönhatásának három fontos helyszíne van:



1. az adatbázis elemeit tartalmazó lemezblokkok területe; (D)
2. a pufferkezelő által használt virtuális vagy valós memóriaterület; (M)
3. a tranzakció memóriaterülete. (M)

OLVASÁS:

- **Ahhoz, hogy a tranzakció egy X adatbáziselemet beolvashasson, azt előbb memóriapuffer(ek)be (P) kell behozni, ha még nincs ott.**
- **Ezt követően tudja a puffer(ek) tartalmát a tranzakció a saját memóriaterületére (t) beolvasni.**



ÍRÁS:

- **Az adatbáziselem új értékének kiírása fordított sorrendben történik: az új értéket a tranzakció alakítja ki a saját memóriaterületén, majd ez az új érték másolódik át a megfelelő puffer(ek)be.**

Fontos, hogy egy tranzakció sohasem módosíthatja egy adatbáziselem értékét közvetlenül a lemezen!

Az adatmozgások alapműveletei:

1. **INPUT(X): Az X adatbáziselemet tartalmazó lemezblokk másolása a memóriapufferbe.**
2. **READ(X,t): Az X adatbáziselem bemásolása a tranzakció t lokális változójába. Részletesebben: ha az X adatbáziselemet tartalmazó blokk nincs a memóriapufferben, akkor előbb végrehajtódik INPUT(X). Ezután kapja meg a t lokális változó X értékét.**
3. **WRITE(X,t): A t lokális változó tartalma az X adatbáziselem memóriapufferbeli tartalmába másolódik. Részletesebben: ha az X adatbáziselemet tartalmazó blokk nincs a memóriapufferben, akkor előbb végrehajtódik INPUT(X). Ezután másolódik át a t lokális változó értéke a pufferbeli X-be.**
4. **OUTPUT(X): Az X adatbáziselemet tartalmazó puffer kimásolása lemezre.**

Az adatbáziselem mérete

- **FELTEVÉS: az adatbáziselemek elférnek egy-egy lemezblokkban és így egy-egy pufferben is, azaz feltételezhetjük, hogy az adatbáziselemek pontosan a blokkok.**
- **Ha az adatbáziselem valójában több blokkot foglalna el, akkor úgy is tekinthetjük, hogy az adatbáziselem minden blokkméretű része önmagában egy adatbáziselem.**
- **A naplózási mechanizmus atomos, azaz vagy lemezre írja X összes blokkját, vagy semmit sem ír ki.**
- **A READ és a WRITE műveleteket a tranzakciók használják, az INPUT és OUTPUT műveleteket a pufferkezelő alkalmazza, illetve bizonyos feltételek mellett az OUTPUT műveletet a naplózási rendszer is használja.**

Főprobléma: A befejezetlen tranzakciók

Például: **Konzisztencia feltétel: A=B**

T_1: $A \leftarrow A \times 2$

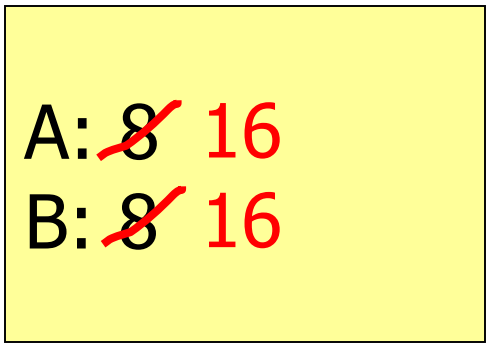
$B \leftarrow B \times 2$

Pontosabban 8 lépésből áll:

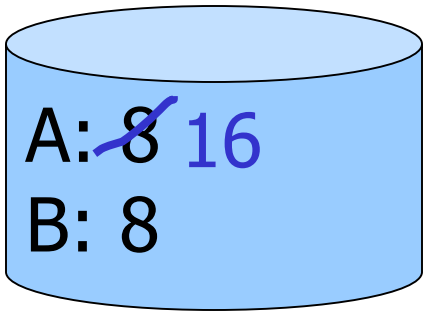
READ(A,t); t := t*2; WRITE(A,t);
READ(B,t); t := t*2; WRITE(B,t);
OUTPUT(A); OUTPUT(B);

T_1: Read (A,t); $t \leftarrow t \times 2$
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);

Rendszerhiba!
Inkonzisztens maradna!



MEMÓRIA



LEMEZ

Az értékek változása a memóriában és a lemezen

	Tevékenység	*t*	*M*-A	*M*-B	*D*-A	*D*-B
1.	READ(A,t)	8	8		8	8
2.	t := t*2	16	8		8	8
3.	WRITE(A,t)	16	16		8	8
4.	READ(B,t)	8	16	8	8	8
5.	t := t*2	16	16	8	8	8
6.	WRITE(B,t)	16	16	16	8	8
7.	OUTPUT(A)	16	16	16	16	8
8.	OUTPUT(B)	16	16	16	16	16

- **Az atomosság miatt:**
  - nem maradhat így, vagy minden lépést végre kell hajtani, vagy egyet sem:
  - vagy A=B=8 vagy A=B=16 lenne jó
- **MEGOLDÁS: naplózással**

• A *napló* (log): *naplóbejegyzések* (log records) sorozata, melyek mindegyike arról tartalmaz valami információt, hogy mit tett egy tranzakció.

• Ha rendszerhiba fordul elő, akkor a napló segítségével rekonstruálható, hogy a tranzakció mit tett a hiba fellépéséig.

• A naplót (az archívmentéssel együtt) használhatjuk akkor is, amikor eszközhiba keletkezik a naplót nem tároló lemezen.

Naplóbejegyzések

A tranzakciók legfontosabb történéseit írjuk ide, például:

- ***Ti kezdődik:*** **(*Ti*, START)**
- ***Ti írja A-t:*** **(*Ti*, *A*, régi érték, új érték)**

(néha elég csak a régi vagy csak az új érték, a naplózási protokolltól függően)

- ***Ti rendben befejeződött*****: (*Ti*, COMMIT)**
- ***Ti a normálisnál korábban fejeződött be*****: (*Ti*, ABORT)**

A napló időrendben tartalmazza a történéseket és tipikusan a háttértáron tartjuk, amiről feltesszük, hogy nem sérült meg.

Fontos, hogy a naplóbejegyzéseket mikor írjuk át a pufferből a lemezre (például a naplókezelő kényszeríti ki, hogy COMMIT esetén a változások a lemezen is megtörténtek.

KÉTFÉLE NAPLÓZÁS:

- Egyes tranzakciók hatását viszont vissza kívánjuk vonni, azaz kérjük az adatbázis visszaállítását olyan állapotba, mintha a tekintett tranzakció nem is működött volna. (SEMMISSÉGI – UNDO)

- A katasztrófák hatásának kijavítását követően a tranzakciók hatását meg kell ismételni, és az általuk adatbázisba írt új értékeket ismételten ki kell írni. (HELYREÁLLÍTÓ – REDO)

Összefoglalás

Tranzakciókezelés és naplózás feladatai, folyamatai, adategységek választása, meghibásodás modell (részei és műveletei), tranzakciók táblázatos reprezentálása, lehetséges naplóbejegyzések, a naplózás két fajtája (Undo, Redo)