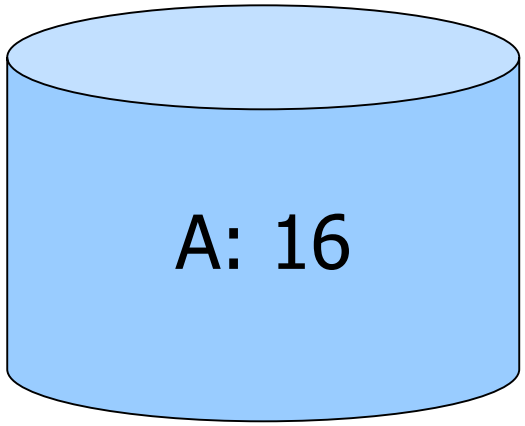
Az eszközök meghibásodásának kezelése

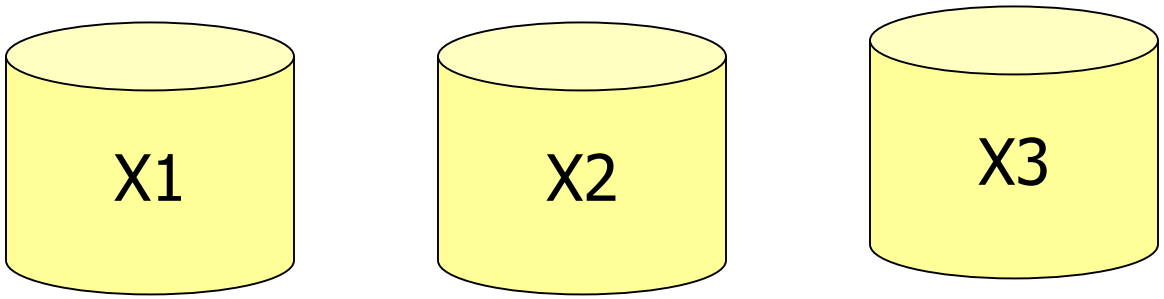


Megoldás: **Készítsünk másolatokat (backup - mentés) az adatokról!**

Védelmi módszerek a lemezhibák ellen:

1. **Háromszoros redundancia**

- **3 másolat különböző lemezeken**
- **Output(X) --> 3 kiírás**
- **Input(X) --> 3 beolvasás + szavazás**



Védelmi módszerek a lemezhibák ellen:

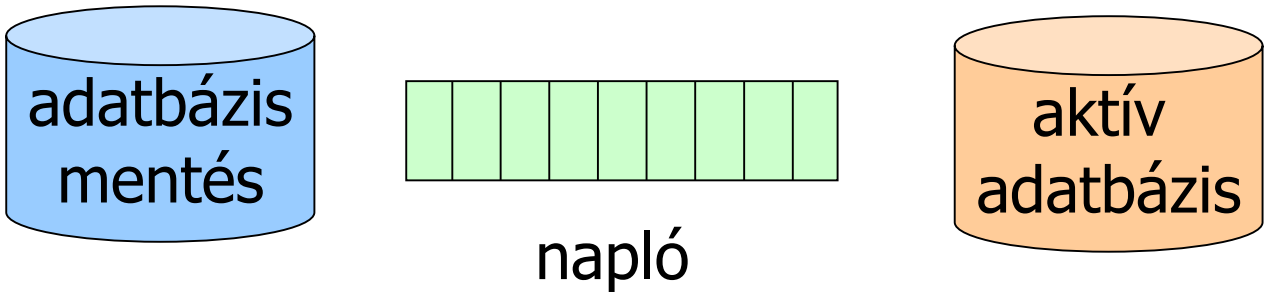
2. Többszörös írás, egyszeres olvasás

- N másolatot tartunk különböző lemezeken
- Output(X) --> N kiírás
- Input(X) --> 1 másolat beolvasása
  - ha ok, kész
  - különben egy másik másolat beolvasása

⇔ Feltevés: észrevesszük, ha rossz egy adat

Védelmi módszerek a lemezhibák ellen:

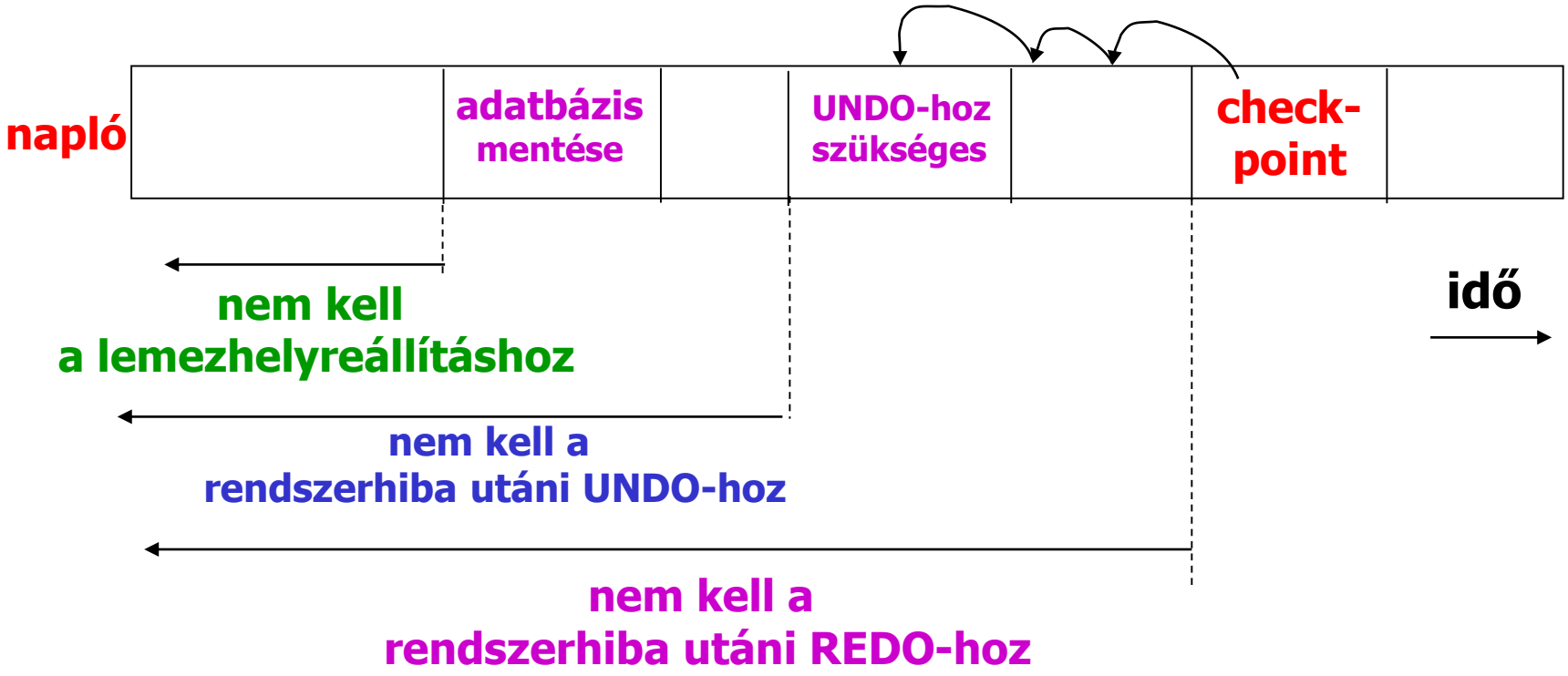
3: Adatbázis mentés + napló



Ha az aktív adatbázis megsérül,

1. **az adatbázis visszatöltése a mentésből**
2. **a napló redo bejegyzéseiből naprakész állapot visszaállítása**

A napló melyik részét lehet eldobni?



Helyreállítás mentésekből és naplóból

- A napló használatával sokkal frissebb állapotot tudunk rekonstruálni.
- **Feltétel:** A biztonsági másolat készítése után történt adatbázis-változásokról keletkező napló túlélte az eszköz meghibásodását
- Visszaállítjuk a biztonsági másolatot, majd a napló felhasználásával a mentés óta történt adatbázis-változásokat át tudjuk vezetni az adatbázison.

Az adatbázist a naplóból akkor tudjuk rekonstruálni, ha:

1. a naplót tároló lemez különbözik az adatbázist tartalmazó lemez(ek)től;
2. a naplót sosem dobjuk el az ellenőrzőpont-képzést követően;
3. a napló helyrehozó vagy semmisségi/helyrehozó típusú, így az új értékeket (is) tárolja.

Probléma: A napló esetleg az adatbázisnál is gyorsabban növekedhet, így nem praktikus a naplót örökre megőrizni.

A mentések szintjei

A mentésnek két szintjét különböztetjük meg:

- ***teljes mentés*** **(full dump), amikor az egész adatbázisról másolat készül;**
- ***növekményes mentés*** **(incremental dump), amikor az adatbázisnak csak azon elemeiről készítünk másolatot, melyek az utolsó teljes vagy növekményes mentés óta megváltoztak.**

Helyreállítás: a teljes mentésből és a megfelelő növekményes mentésekből

- **A helyrehozó vagy a semmisségi/helyrehozó naplózás rendszerhiba utáni visszaállítási folyamatához hasonló módszerrel.**
- **Visszamásoljuk a teljes mentést, majd az ezt követő legkorábbi növekményes mentéstől kezdve végrehajtjuk a növekményes mentésekben tárolt változtatásokat.**

Mentés működés közben

Ha leállítjuk a rendszert, akkor nyugodtan lehet menteni.

Probléma:

- **sokáig tarthat a leállítás, újraindítás**
- **nem biztos, hogy egyáltalán le szabad állítani a rendszert**

Megoldás: működés közben mentünk

Példa:

- **A, B, C és D értéke az archiválás kezdetekor rendre 1, 2, 3, 4.**
- **A mentés közben A értéke 5-re, C értéke 6-ra, B értéke 7-re módosul.**
- **Az adatbáziselemeket a mentéskor sorban másoljuk az archívumba.**

- **A mentés végére pedig 5, 7, 6, 4 az adatbázis állapota, a mentett archívumba 1, 2, 6, 4 került, jóllehet ilyen adatbázis-állapot a mentés ideje alatt nem is fordult elő.**

Lemez	Mentés
	A
A := 5	
	B
C := 6	
	C
B := 7	
	D

Mentés működés közben

1. A <START DUMP> bejegyzés naplóba írása.
2. A REDO vagy UNDO/REDO naplózási módnak megfelelő ellenőrzőpont kialakítása.
3. Az adatlemez(ek) teljes vagy növekményes mentése.
4. A napló mentése. A mentett naplórész tartalmazza legalább a 2. pontbeli ellenőrzőpont-képzés közben keletkezett naplóbejegyzéseket, melyeknek túl kell élniük az adatbázist hordozó eszköz meghibásodását.
5. <END DUMP> bejegyzés naplóba írása.

Megjegyzés: A mentés befejezésekor eldobhatjuk a naplónak azt a részét, amelyre nincs szükség a 2. pontban végrehajtott ellenőrzőpont-képzéshez tartozó helyreállítási folyamat szabályai szerint.

UNDO napló nem használható: Mivel UNDO naplózás esetén az OUTPUT műveletek a módosítási bejegyzés naplóba írását követően bármikor lefuthatnak, ezért előfordulhat, olyan eredményt kapunk, mintha egy tranzakció nem atomosan hajtódott volna végre.

Mentés működés közben

Tegyük fel, hogy a fenti adatbázis mentés közbeni módosításait két tranzakció, T1 (mely A-t és B-t módosította) és T2 (mely C-t módosította) végezte, melyek a mentés kezdetekor aktívak voltak. UNDO/REDO naplózási módszert alkalmazva a mentés alatti események lehetséges naplóbejegyzései a következők:

<START DUMP>
<START CKPT(T1,T2)>
<T1,A,1,5>
<T2,C,3,6>
<T2, COMMIT>
<T1,B,2,7>
<END CKPT>
a mentés befejezése
<END DUMP>

Lemez	Mentés
	A
A := 5	
	B
C := 6	
	C
B := 7	
	D

Helyreállítás mentésből és naplóból

Tegyük fel, hogy a biztonsági mentés elkészítését követően történik katasztrófa, és a napló ezt túlélte. Az érdekesség kedvéért tegyük fel, hogy a napló katasztrófát túlélt részében nincs <T1,COMMIT> bejegyzés, van viszont <T2,COMMIT>. Az adatbázist először a biztonsági mentésből visszatöltjük, így A, B, C, D értékei rendre 1, 2, 6, 4 lesznek.

<START DUMP>
<START CKPT(T1,T2)>
<T1,A,1,5>
<T2,C,3,6>
<T2, COMMIT>
<T1,B,2,7>
<END CKPT>
a mentés befejezése
<END DUMP>

- T2 befejezett tranzakció, helyreállítjuk azon lépés hatását, amely C értékét 6-ra módosította.
- T1 hatásait semmissé kell tennünk. A értékét 1-re, B értékét 2-re kell visszaállítanunk.

Lemez	Mentés
	A
A := 5	
	B
C := 6	
	C
B := 7	
	D

Az Oracle naplózási és archiválási rendszere

- **Rendszerhiba esetén a helyreállítás-kezelő automatikusan aktivizálódik, amikor az Oracle újraindul.**
- **A helyreállítás a *napló* (redo log) alapján történik. A napló olyan állományok halmaza, amelyek az adatbázis változásait tartalmazzák, akár lemezre kerültek, akár nem. Két részből áll: az online és az archivált naplóból.**
- **Az *online napló* kettő vagy több online naplófájlból áll.**
- **A naplóbejegyzések ideiglenesen az SGA (System Global Area) memóriapuffereiben tárolódnak, amelyeket a Log Writer (LGWR) háttérfolyamat folyamatosan ír ki lemezre. (Az SGA tartalmazza az adatbáziselemeket tároló puffereket is, amelyeket pedig a Database Writer háttérfolyamat ír lemezre.)**
- **Ha egy felhasználói folyamat befejezte egy tranzakció végrehajtását, akkor a LGWR egy COMMIT bejegyzést is kiír a naplóba.**

Az Oracle naplózási és archiválási rendszere

- **Az online naplófájlok ciklikusan töltődnek föl. Például ha a naplót két fájl alkotja, akkor először az elsőt írja tele a LGWR, aztán a másodikat, majd újraírja az elsőt stb. Amikor egy naplófájl megtelt, kap egy sorszámot (log sequence number), ami azonosítja a fájlt.**

- **A biztonság növelése érdekében az Oracle lehetővé teszi, hogy a naplófájlokat több példányban letároljuk. A *multiplexelt online naplóban* ugyanazon naplófájlok több különböző lemezen is tárolódnak, és ezek egyszerre módosulnak. Ha az egyik lemez megsérül, akkor a napló többi másolata még mindig rendelkezésre áll a helyreállításhoz.**

- **Lehetőség van arra, hogy a megtelt online naplófájlokat archiváljuk, mielőtt újra felhasználnánk őket. Az *archivált (offline) napló* az ilyen archivált naplófájlokból tevődik össze.**

Az Oracle naplózási és archiválási rendszere

- A naplókezelő két módban működhet: ARCHIVELOG módban a rendszer minden megtelt naplófájlt archivál, mielőtt újra felhasználná, NOARCHIVELOG módban viszont a legrégebbi megtelt naplófájl mentés nélkül felülíródik, ha az utolsó szabad naplófájl is megtelt.
- ARCHIVELOG módban az adatbázis teljesen visszaállítható rendszerhiba és eszközhiba után is, valamint az adatbázist működés közben is lehet archiválni. Hátránya, hogy az archivált napló kezeléséhez külön adminisztrációs műveletek szükségesek.
- NOARCHIVELOG módban az adatbázis csak rendszerhiba után állítható vissza, eszközhiba esetén nem, és az adatbázist archiválni csak zárt állapotában lehet, működés közben nem. Előnye, hogy a DBA-nak nincs külön munkája, mivel nem jön létre archivált napló.
- A naplót a LogMiner naplóelemző eszköz segítségével analizálhatjuk, amelyet SQL alapú utasításokkal vezérelhetünk.

Az Oracle naplózási és archiválási rendszere

- **A helyreállításhoz szükség van még egy *vezérlőfájlra* (control file) is, amely többek között az adatbázis fájlszerkezetéről és a LGWR által éppen írt naplófájl sorszámáról tartalmaz információkat. Az automatikus helyreállítási folyamatot a rendszer ezen vezérlőfájl alapján irányítja. Hasonlóan a naplófájlokhoz, a vezérlőfájlt is tárolhatjuk több példányban, amelyek egyszerre módosulnak. Ez a *multiplexelt vezérlőfájl*.**

A rollback szegmensek és a helyreállítás folyamata

- Az Oracle az UNDO és a REDO naplózás egy speciális keverékét valósítja meg.
- A tranzakciók hatásainak semmissé tételéhez szükséges információkat a *rollback szegmensek* tartalmazzák. Minden adatbázisban van egy vagy több rollback szegmens, amely a tranzakciók által módosított adatok régi értékeit tárolja attól függetlenül, hogy ezek a módosítások lemezre íródtak vagy sem. A rollback szegmenseket használjuk az olvasási konzisztencia biztosítására, a tranzakciók visszagörgetésére és az adatbázis helyreállítására is.
- A rollback szegmens *rollback bejegyzésekből* áll. Egy rollback bejegyzés többek között a megváltozott blokk azonosítóját (fájlsorszám és a fájlon belüli blokkazonosító) és a blokk régi értékét tárolja. A rollback bejegyzés mindig előbb kerül a rollback szegmensbe, mint ahogy az adatbázisban megtörténik a módosítás. Az ugyanazon tranzakcióhoz tartozó bejegyzések össze vannak láncolva, így könnyen visszakereshetők, ha az adott tranzakciót vissza kell görgetni.

A rollback szegmensek és a helyreállítás folyamata

- **A rollback szegmenseket sem a felhasználók, sem az adatbázis-adminisztrátorok nem olvashatják. Mindig a SYS felhasználó a tulajdonosuk, attól függetlenül, ki hozta őket létre.**

- **Minden rollback szegmenshez tartozik egy *tranzakciós tábla*, amely azon tranzakciók listáját tartalmazza, amelyek által végrehajtott módosításokhoz tartozó rollback bejegyzések az adott rollback szegmensben tárolódnak. Minden rollback szegmens fix számú tranzakciót tud kezelni. Ez a szám az adatblokk méretétől függ, amit viszont az operációs rendszer határoz meg. Ha explicit módon másképp nem rendelkezünk, az Oracle egyenletesen elosztja a tranzakciókat a rollback szegmensek között.**

A rollback szegmensek és a helyreállítás folyamata

- **Ha egy tranzakció befejeződött, akkor a rá vonatkozó rollback bejegyzések még nem törölhetők, mert elképzelhető, hogy még a tranzakció befejeződése előtt elindult egy olyan lekérdezés, amelyhez szükség van a módosított adatok régi értékeire. Hogy a rollback adatok minél tovább elérhetők maradjanak, a rollback szegmensbe a bejegyzések sorban egymás után kerülnek be. Amikor megtelik a szegmens, akkor az Oracle az elejéről kezdi újra feltölteni. Előfordulhat, hogy egy sokáig futó tranzakció miatt nem írható felül a szegmens eleje, ilyenkor a szegmenst ki kell bővíteni.**

- **Amikor létrehozunk egy adatbázist, automatikusan létrejön egy SYSTEM nevű rollback szegmens is a SYSTEM táblaterületen. Ez nem törölhető. Erre a szegmensre mindig szükség van, akár létrehozunk további rollback szegmenseket, akár nem. Ha több rollback szegmensünk van, akkor a SYSTEM nevűt az Oracle csak speciális rendszertranzakciókra próbálja használni, a felhasználói tranzakciókat pedig szétosztja a többi rollback szegmens között. Ha viszont túl sok felhasználói tranzakció fut egyszerre, akkor a SYSTEM szegmenst is használni fogja erre a célra.**

A rollback szegmensek és a helyreállítás folyamata

- **Naplózás naplózása: Amikor egy rollback bejegyzés a rollback szegmensbe kerül, a naplóban erről is készül egy naplóbejegyzés, hiszen a rollback szegmensek (más szegmensekhez hasonlóan) az adatbázis részét képezik.**
- **A helyreállítás szempontjából nagyon fontos a módosításoknak ez a kétszeres feljegyzése.**

1. **Ha rendszerhiba történik, először a napló alapján visszaállításra kerül az adatbázisnak a rendszerhiba bekövetkezése előtti állapota, amely inkonzisztens is lehet. Ez a folyamat a *rolling forward*.**
2. **A helyreállítás folyamán a rollback szegmens is visszaállítódik, amiben az aktív tranzakciók által végrehajtott tevékenységek semmissé tételéhez szükséges információk találhatók. Ezek alapján ezután minden aktív tranzakcióra végrehajtódik egy ROLLBACK utasítás. Ez a *rolling back* folyamat.**

Az archiválás folyamata

- Az eszközhibák okozta problémák megoldására az Oracle is használja az *archiválást*.
- A *teljes mentés* az adatbázishoz tartozó adatfájlok, az online naplófájlok és az adatbázis vezérlőfájljának operációsrendszer-szintű mentését jelenti.
- *Részleges mentés* esetén az adatbázisnak csak egy részét mentjük, például az egy táblaterülethez tartozó adatfájlokat vagy csak a vezérlőfájlt. A részleges mentésnek csak akkor van értelme, ha a naplózás ARCHIVELOG módban történik. Ilyenkor a naplót felesleges újra archiválni.
- A mentés lehet teljes és növekményes is.
- Ha az adatbázist konzisztens állapotában archiváltuk, akkor *konzisztens mentésről* beszélünk. A konzisztens mentésből az adatbázist egyszerűen visszamásolhatjuk, nincs szükség a napló alapján történő helyreállításra.
- Lehetőség van az adatbázist egy régebbi mentésből visszaállítani, majd csak néhány naplóbejegyzést figyelembe véve az adatbázist egy meghatározott időpontbeli állapotába visszavinni. Ezt *nem teljes helyreállításnak* (incomplete recovery) nevezzük.

Összefoglalás

- Eszközök meghibásodása
  - Másolatok
  - Mentés leállással, működés közben
- Oracle naplózása és archiválása
  - Log fájl (REDO napló)
  - Rollback szegmens (UNDO napló)
  - Helyreállítás (rolling forward, rollback)
  - Teljes vagy részleges mentés