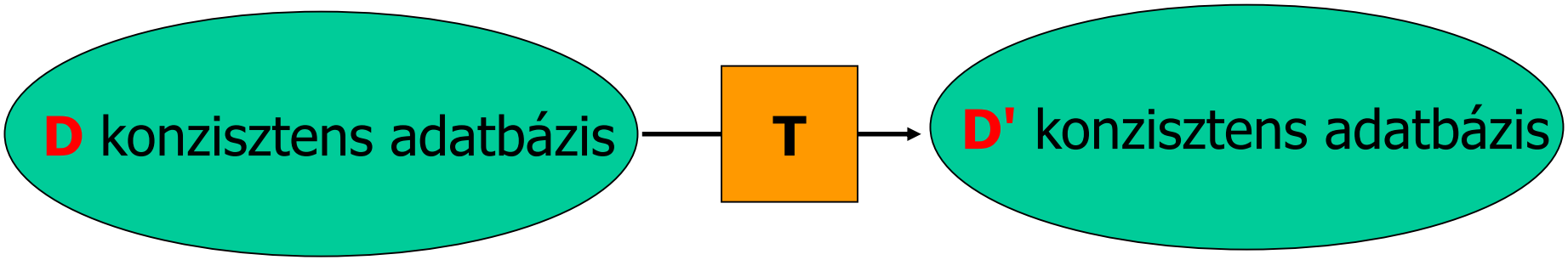
Tranzakciókezelés alapfogalmai

TRANZAKCIÓ:

Konzisztenciát megtartó adatkezelő műveletek sorozata



Ezek után mindig feltesszük:

Ha T tranzakció konzisztens állapotból indul

+ T tranzakció csak egyedül futna le

⇒ T konzisztens állapotban hagyja az adatbázis

Konzisztencia, megszorítások

- **Mit jelent a konzisztencia?**
- **Az adatok előre megadott feltételeket, predikátumokat elégítenek ki.**
- **Például:**
  - **X az R reláció kulcsa**
  - **X → Y függőség teljesül R-ben**
  - **megengedett értékek korlátozása:**
    - **Domain(X) = {piros, kék, zöld}**
  - **α indexfájl az R reláció X attribútumának érvényes indexe**
  - **Semelyik dolgozó nem keres többet, mint az átlagfizetés kétszerese**

Definíció:

- **Konzisztens állapot:**
  kielégíti az összes feltételt (megszorítást)
- **Konzisztens adatbázis:**
  konzisztens állapotú adatbázis

Általánosabb megszorítások

Tranzakciós megszorítások

- **Ha módosítjuk a fizetést, akkor az**

  új fizetés > régi fizetés

(Egy állapotból nem lehet ellenőrizni, mivel ez a változtatás módjára ad meg feltételt!)

- **A számla rekord törlése után legyen**

  az egyenleg = 0.

  (Ezt sem lehet egy állapotra ellenőrizni, mivel vagy törlés miatt lett az egyenleg 0, vagy eleve 0 értéket tároltunk.)

Megjegyzés: az utóbbi szimulálható közönséges megszorítással, ha felveszünk egy **törölve** oszlopot.

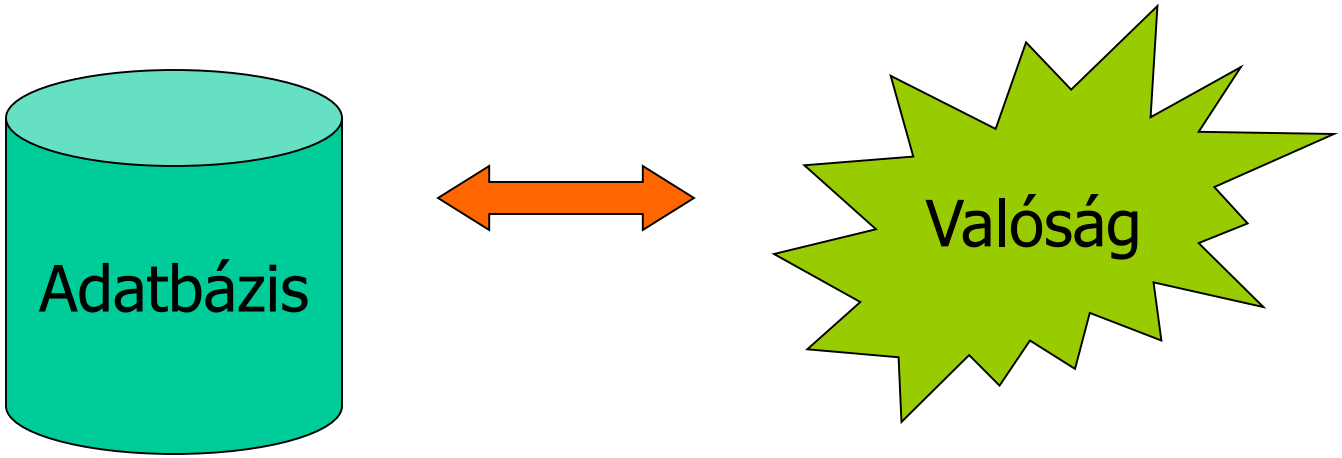
Ha törölve=igen, akkor egyenleg=0.

számla

AZON #	....	egyenleg	törölve

A megszorítások hiányossága

Az adatbázis a valóságot próbálja reprezentálni, de minden összefüggést (vagyis a valóság teljes szemantikáját) lehetetlen megadni.



Vegyük észre: Az adatbázis **nem lehet állandóan** konzisztens!

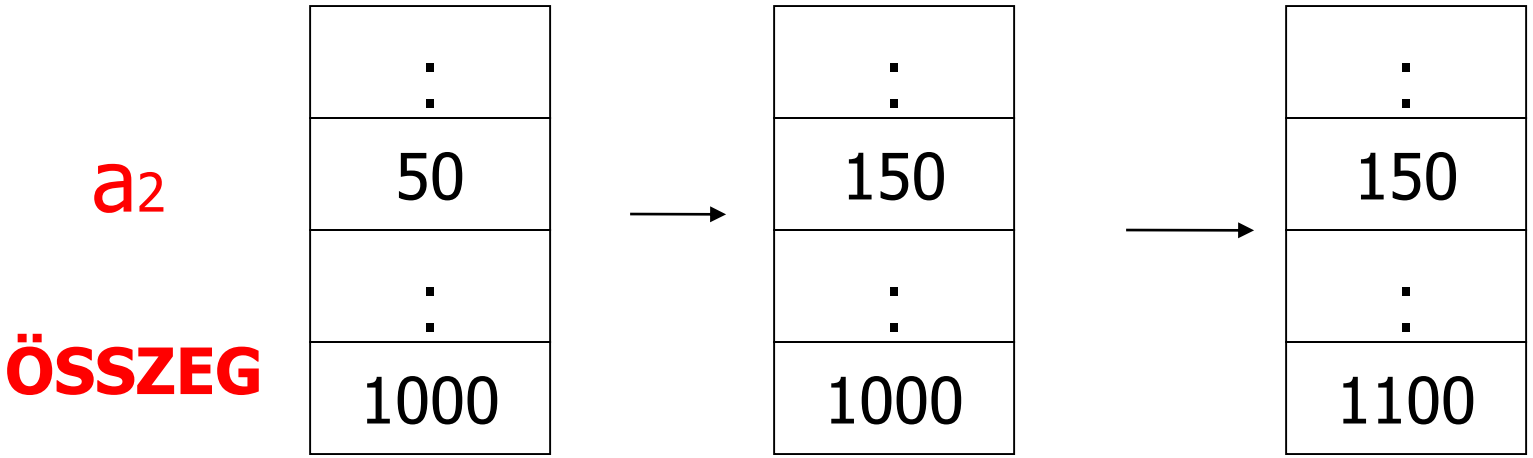
Például:

$a_1 + a_2 + \ldots a_n$ = **ÖSSZEG** (**megszorítás**)

Adjunk 100-at az a_2-höz:

$$\begin{cases} a_2 \leftarrow a_2 + 100 \\ \text{ÖSSZEG} \leftarrow \text{ÖSSZEG} + 100 \end{cases}$$

A két lépést nem tudjuk egyszerre végrehajtani, így egy pillanatra megsérül a konzisztencia.



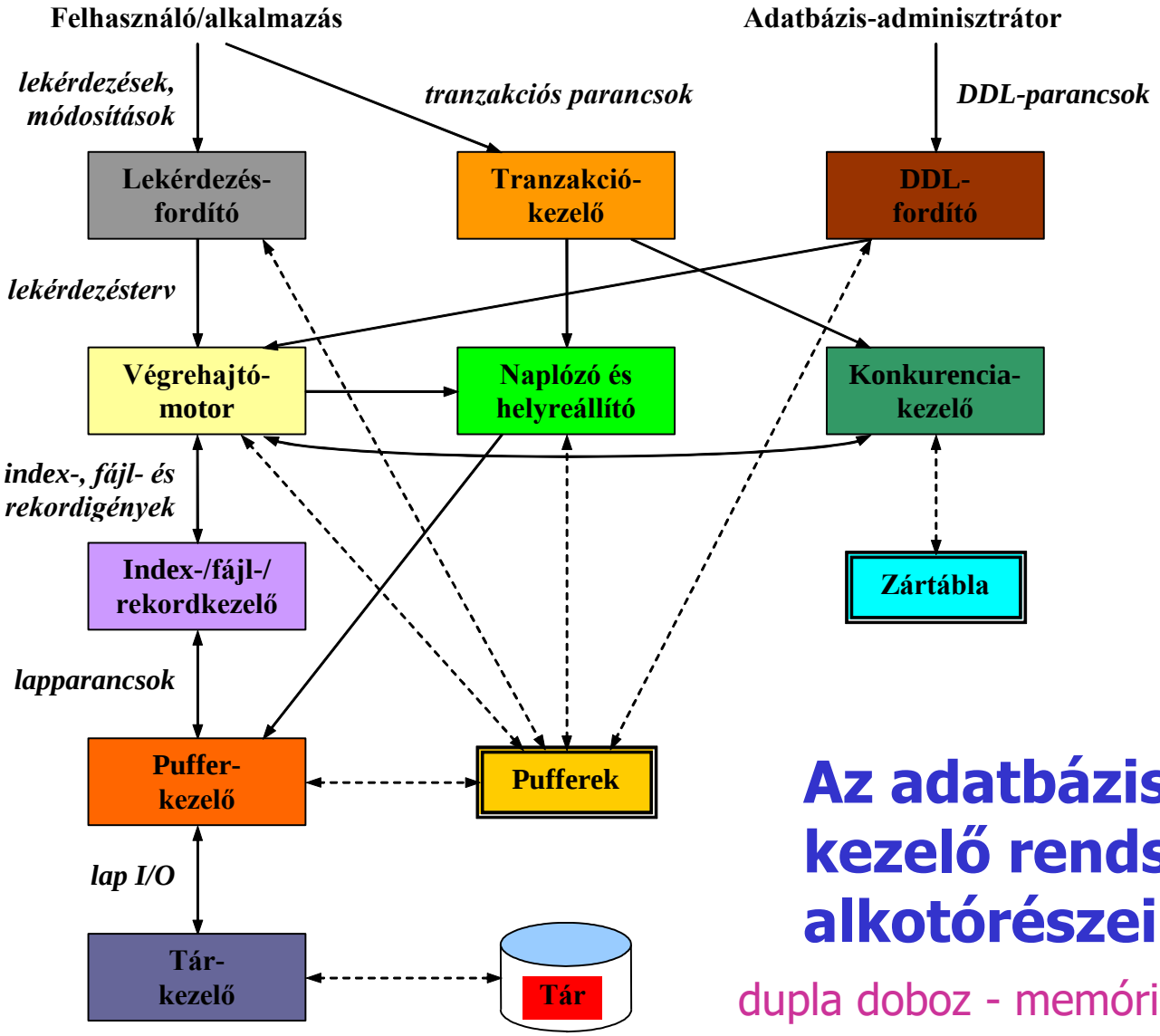
Konzisztens **NEM Konzisztens** **Konzisztens**

Helyesség feltétele

1. **Ha leáll valamelyik, vagy több tranzakció (abort, vagy hiba miatt), akkor is**

   konzisztens D adatbázist kell előállítanunk

2. **Mind egyes tranzakció induláskor konzisztens D adatbázist lát.**



Az adatbázis-kezelő rendszer alkotórészei

dupla doboz - memória

folytonos vonal - vezérlésátadás, adatáramlással

szaggatott vonal - csak adatmozgás

A lekérdezés megválaszolása

- A **lekérdezésfordító** elemzi és optimalizálja a lekérdezést.
- Az eredményül kapott lekérdezés-végrehajtási tervet (a megválaszolásához szükséges tevékenységek sorozatát) továbbítja a végrehajtómotornak.
- A **végrehajtómotor** kisebb adatdarabokra (tipikusan rekordokra) vonatkozó kérések sorozatát adja át az erőforrás-kezelőnek.
- Az **erőforrás-kezelő** ismeri a relációkat tartalmazó adatfájlokat, a fájlok rekordjainak formátumát, méretét és az indexfájlokat is. Az adatkéréseket az erőforrás-kezelő lefordítja lapokra (blokkokra), és ezeket a kéréseket továbbítja a pufferkezelőnek.
- A **pufferkezelő** feladata, hogy a másodlagos adattárolón (általában lemezen) tárolt adatok megfelelő részét hozza be a központi memória puffereibe. A pufferek és a lemez közti adatátvitel egysége általában egy lap vagy egy lemezblokk. A pufferkezelő információt cserél a tárkezelővel, hogy megkapja az adatokat a lemezről. Megtörténhet, hogy a tárkezelő az operációs rendszer parancsait is igénybe veszi, de tipikusabb, hogy az adatbázis-kezelő a parancsait közvetlenül a lemezvezérlőhöz intézi.

A tranzakció feldolgozása.

- A lekérdezéseket és más tevékenységeket tranzakciókba csoportosíthatjuk.
- A tranzakciók olyan munkaegységek, amelyeket atomosan és más tranzakcióktól látszólag elkülönítve kell végrehajtani. (Gyakran minden egyes lekérdezés vagy módosítás önmagában is egy tranzakció.)
- A tranzakció végrehajtásának tartósnak kell lennie, ami azt jelenti, hogy bármelyik befejezett tranzakció hatását még akkor is meg kell tudni őrizni, ha a rendszer összeomlik a tranzakció befejezése utáni pillanatban.
- A tranzakciófeldolgozót két fő részre osztjuk:
  - *Konkurenciavezérlés-kezelő* vagy *ütemező* (scheduler): a tranzakciók elkülönítésének és atomosságának biztosításáért felelős.
  - *Naplózás- és helyreállítás-kezelő*: a tranzakciók atomosságáért és tartósságáért felelős.

A tranzakció

- A *tranzakció* az adatbázis-műveletek végrehajtási egysége, amely DML-beli utasításokból áll, és a következő tulajdonságokkal rendelkezik:

A ***Atomosság*** **(atomicity):** a tranzakció „mindent vagy semmit” jellegű végrehajtása (vagy teljesen végrehajtjuk, vagy egyáltalán nem hajtjuk végre).

C ***Konzisztencia*** **(consistency):** az a feltétel, hogy a tranzakció megőrizze az adatbázis konzisztenciáját, azaz a tranzakció végrehajtása után is teljesüljenek az adatbázisban előírt konzisztenciamegszorítások (integritási megszorítások), azaz az adatelemekre és a közöttük lévő kapcsolatokra vonatkozó elvárások.

I ***Elkülönítés*** **(isolation):** az a tény, hogy minden tranzakciónak látszólag úgy kell lefutnia, mintha ez alatt az idő alatt semmilyen másik tranzakciót sem hajtanánk végre.

D ***Tartósság*** **(durability):** az a feltétel, hogy ha egyszer egy tranzakció befejeződött, akkor már soha többé nem veszhet el a tranzakciónak az adatbázison kifejtett hatása.

- Ezek a tranzakció *ACID-tulajdonságai* .

A tranzakció

Megjegyzések:

- **A konzisztenciát mindig adottnak tekintjük.**
- **A másik három tulajdonságot viszont az adatbázis-kezelő rendszernek kell biztosítania, de ettől időnként eltekintünk.**
- **Ha egy ad hoc utasítást adunk az SQL-rendszernek, akkor minden lekérdezés vagy adatbázis-módosító utasítás egy tranzakció.**
- **Amennyiben beágyazott SQL-interfészt használva a programozó készíti el a tranzakciót, akkor egy tranzakcióban több SQL-lekérdezés és -módosítás szerepelhet. A tranzakció ilyenkor általában egy DML-utasítással kezdődik, és egy COMMIT vagy ROLLBACK paranccsal végződik. Ha a tranzakció valamely utasítása egy triggert aktivizál, akkor a trigger is a tranzakció részének tekintendő, akárcsak a trigger által kiváltott további triggerek. (A trigger olyan programrész, amely bizonyos események bekövetkeztekor automatikusan lefut.)**

A tranzakció feldolgozása

A tranzakciófeldolgozó a következő 3 feladatot hajtja végre:

- naplózás
- konkurenciavezérlés
- holtpont feloldása

1. *Naplózás*:
   - Annak érdekében, hogy a tartósságot biztosítani lehessen, az adatbázis minden változását külön feljegyezzük (naplózzuk) lemezen.
   - A naplókezelő (log manager) többféle eljárásmód közül választja ki azt, amelyiket követni fog.
   - Ezek az eljárásmódok biztosítják azt, hogy teljesen mindegy, mikor történik a rendszerhiba vagy a rendszer összeomlása, a helyreállítás-kezelő meg fogja tudni vizsgálni a változások naplóját, és ez alapján vissza tudja állítani az adatbázist valamilyen konzisztens állapotába.
   - A naplókezelő először a pufferekbe írja a naplót, és egyeztet a pufferkezelővel, hogy a pufferek alkalmas időpillanatokban garantáltan íródjanak ki lemezre, ahol már az adatok túlélhetik a rendszer összeomlását.

A tranzakció feldolgozása

2. *Konkurenciavezérlés*:

- **A tranzakcióknak úgy kell látszódniuk, mintha egymástól függetlenül, elkülönítve végeznénk el őket.**

- **Az ütemező (konkurenciavezérlés-kezelő) feladata, hogy meghatározza az összetett tranzakciók résztevékenységeinek egy olyan sorrendjét, amely biztosítja azt, hogy ha ebben a sorrendben hajtjuk végre a tranzakciók elemi tevékenységeit, akkor az összhatás megegyezik azzal, mintha a tranzakciókat tulajdonképpen egyenként és egységes egészként hajtottuk volna végre.**

T1. tranzakció: $u1; u2; : : : ; u10$

T2. tranzakció: $v1; v2; : : : ; v103$

A két utasítássorozat nem elkülönülve jön, hanem összefésülődve:

$$u1; v1; v2; u2; u3; v3; : : : ; v103; u10$$

A saját sorrend megmarad mindkettőn belül.

Ekkor olyan állapot is kialakulhat, ami nem jött volna létre, ha egymás után futnak le a tranzakciók.

A tranzakció feldolgozása

2. *Konkurenciavezérlés*:

T1. READ A, A + +, WRITE A
T2. READ A, A + +, WRITE A

Ha ezek úgy fésülődnek össze, hogy

(READ A)$_1$, (READ A)$_2$, (A + +)$_1$,(A + +)$_2$, (WRITE A)$_1$, (WRITE A)$_2$

akkor a végén csak eggyel nő A értéke, holott kettővel kellett volna.

- **A tipikus ütemező ezt a munkát azáltal látja el, hogy az adatbázis bizonyos részeire elhelyezett *zárakat* (lock) karbantartja.**
- **Ezek a zárak megakadályoznak két tranzakciót abban, hogy rossz kölcsönhatással használják ugyanazt az adatrészt. A zárakat rendszerint a központi memória *zártáblájában* (lock table) tárolja a rendszer.**
- **Az ütemező azzal befolyásolja a lekérdezések és más adatbázis-műveletek végrehajtását, hogy megtiltja a végrehajtómotornak, hogy hozzányúljon az adatbázis zár alá helyezett részeihez.**

A tranzakció feldolgozása

3. ***Holtpont feloldása*:**

- **A tranzakciók az ütemező által engedélyezett zárak alapján versenyeznek az erőforrásokért. Így előfordulhat, hogy olyan helyzetbe kerülnek, amelyben egyiküket sem lehet folytatni, mert mindegyiknek szüksége lenne valamire, amit egy másik tranzakció birtokol.**
- **A tranzakciókezelő feladata, hogy ilyenkor közbeavatkozzon, és töröljön (abortáljon) egy vagy több tranzakciót úgy, hogy a többit már folytatni lehessen.**

Holtpont (deadlock):

***l1(A)* jelölje, hogy T1 tranzakció zár alá helyezte az A-t, stb.**

$$l1(A);\ l2(B);\ l3(C);\ l1(B);\ l2(C);\ l3(A)$$

sorrendben érkező zárkérések esetén egyik tranzakció se tud tovább futni.

Mitől sérülhet a konzisztencia?

- **Tranzakcióhiba (hibásan megírt, rosszul ütemezett, félbehagyott tranzakciók.)**
- **Adatbázis-kezelési hiba (az adatbázis-kezelő valamelyik komponense nem, vagy rosszul hajtja végre a feladatát.)**
- **Hardverhiba (elvész egy adat, vagy megváltozik az értéke.)**
- **Adatmegosztásból származó hiba**

  például:

  T1: 10% fizetésemelést ad minden programozónak
  T2: minden programozót átnevez rendszerfejlesztőnek

- **Feladat.** Tegyük fel, hogy az adatbázisra vonatkozó konzisztenciamegszorítás: $0 \leq A \leq B$. Állapítsuk meg, hogy a következő tranzakciók megőrzik-e az adatbázis konzisztenciáját!

```
T1: A := A + B; B := A + B;

T2: B := A + B; A := A + B;

T3: A := B + 1; B := A + 1;
```

Hogy lehet megakadályozni vagy kijavítani a hibák okozta konzisztenciasérülést?

Helyreállítás

- **Első lépés: meghibásodási modell definiálása**
- **Események osztályozása:**

Események → szabályos

→ kivételes → előreláthatató

→ nem várt

Szabályos esemény: "a felhasználói kézikönyv alapján kezelhető esemény"

Előreláthatató, kivételes esemény:

Rendszerösszeomlás:

- elszáll a memória
- leáll a cpu, újraindítás (reset)

Nem várt, kivételes esemény: MINDEN MÁS!

Előrelátható, kivételes események

A hibák fajtái

Az adatbázis lekérdezése vagy módosítása során számos dolog hibát okozhat a billentyűzeten történt adatbeviteli hibáktól kezdve az adatbázist tároló lemez elhelyezésére szolgáló helyiségben történő robbanásig.

- Hibás adatbevitel
- Készülékhibák
- Katasztrófális hibák
- Rendszerhibák

Előrelátható, kivételes események

- **Hibás adatbevitel:**
  - **tartalmi hiba: gyakran nem észrevehetők, például a felhasználó elüt egy számot egy telefonszámban.**
  - **formai hiba: kihagy egy számjegyet a telefonszámból. SQL-ben típus előírások, kulcsok, megszorítások (constraint) definiálásával megakadályozható a hibás adatbevitel.**
  - **A triggerek azok a programok, amelyek bizonyos típusú módosítások (például egy relációba történő beszúrás) esetén hajtódnak végre, annak ellenőrzésére, hogy a frissen bevitt adatok megfelelnek-e az adatbázis-tervező által megszabott előírásoknak.**

Előrelátható, kivételes események

- **Készülékhibák:**
  - **Kis hiba: A lemezegységek olyan helyi hibái, melyek egy vagy több bit megváltozását okozzák, a lemez szektoraihoz rendelt paritás-ellenőrzéssel megbízhatóan felismerhetők.**
  - **Nagy hiba: A lemezegységek jelentős sérülése, elsősorban az író-olvasó fejek katasztrófái, az egész lemez olvashatatlanná válását okozhatják. Az ilyen hibákat általában az alábbi megoldások segítségével kezelik:**

    1. **RAID**
    2. **Archiválás**
    3. **Osztott másolat**

Előreláthatók, kivételes események

1. A *RAID-módszerek* (Redundant Array of Independent Disks) valamelyikének használatával az elveszett lemez tartalma visszatölthető.

2. Az *archiválás* használatával az adatbázisról másolatot készítünk valamilyen eszközre (például szalagra vagy optikai lemezre). A mentést rendszeresen kell végezni vagy teljes, vagy növekményes (csak az előző mentés óta történt változásokat archiváljuk) mentést használva. A mentett anyagot az adatbázistól biztonságos távolságban kell tárolnunk.

3. Az adatbázisról fenntarthatunk *elosztott, on-line másolatokat*. Ebben az esetben biztosítanunk kell a másolatok konzisztenciáját.

Előrelátható, kivételes események

- **Katasztrofális hibák:**
  - **Ebbe a kategóriába soroljuk azokat a helyzeteket, amikor az adatbázist tartalmazó eszköz teljesen tönkremegy robbanás, tűz, vandalizmus vagy akár vírusok következtében.**
  - **A RAID ekkor nem segít, mert az összes lemez és a paritás-ellenőrző lemezeik is egyszerre használhatatlanná válnak.**
  - **A másik két biztonsági megoldás viszont alkalmazható katasztrofális hibák esetén is.**

Előrelátható, kivételes események

- **Rendszerhibák:**
  - **Minden tranzakciónak van *állapota*, mely azt képviseli, hogy mi történt eddig a tranzakcióban. Az állapot tartalmazza a tranzakció kódjában a végrehajtás pillanatnyi helyét és a tranzakció összes lokális változójának értékét.**
  - **A rendszerhibák azok a problémák, melyek a tranzakció állapotának elvesztését okozzák.**

Előrelátható, kivételes események

- Tipikus rendszerhibák az áramkimaradásból és a szoftverhibákból eredők, hiszen ezek a memória tartalmának felülírásával járhatnak.

- Ha egy rendszerhiba bekövetkezik, onnantól kezdve nem tudjuk, hogy a tranzakció mely részei kerültek már végrehajtásra, beleértve az adatbázis-módosításokat is. A tranzakció ismételt futtatásával nem biztos, hogy a problémát korrigálni tudjuk (például egy mezőnek eggyel való növelése esetén).

- Az ilyen jellegű problémák legfontosabb ellenszere minden adatbázis-változtatás naplózása egy elkülönült, nem illékony naplófájlban, lehetővé téve ezzel a visszaállítást, ha az szükséges. Ehhez hibavédett naplózási mechanizmusra van szükség.

Összefoglalás

- Áttekintett fogalmak kulcsszavakban
- Tranzakció, konzisztencia, adatbázis-kezelő tranzakciós moduljai, tranzakció ACID tulajdonságai, tranzakciók feldolgozása (naplózás, konkurenciakezelés, holtpontkezelés), konzisztencia sérülése, meghibásodási modellek