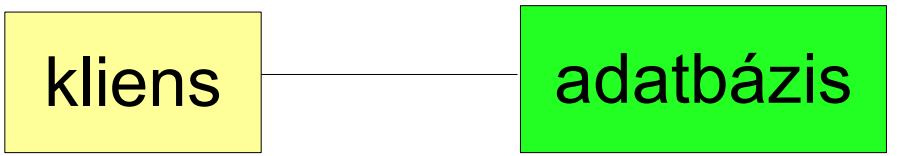
# Többfelhasználós adatbázis környezetek, tranzakciók, internetes megoldások

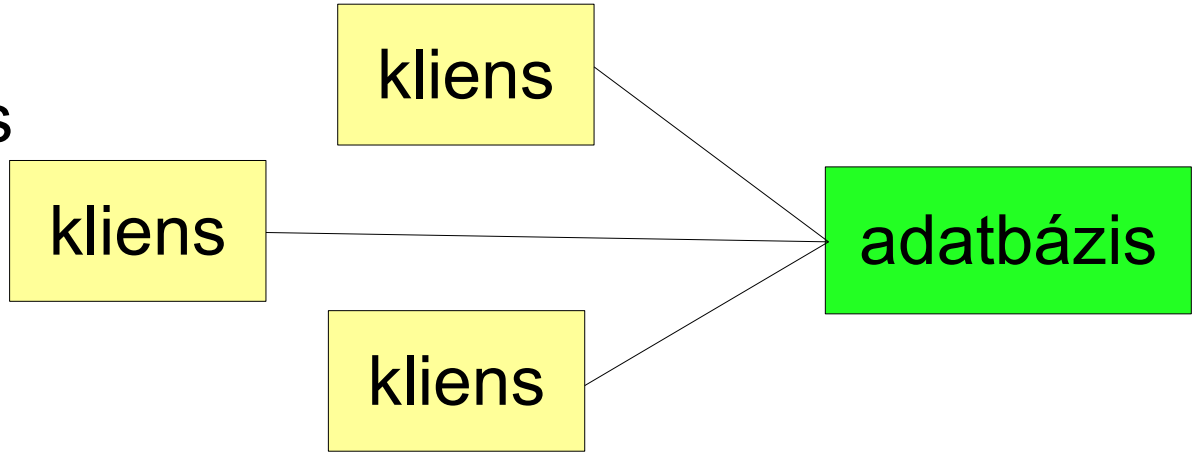
# Alkalmazás modellek

Egy felhasználós környezet

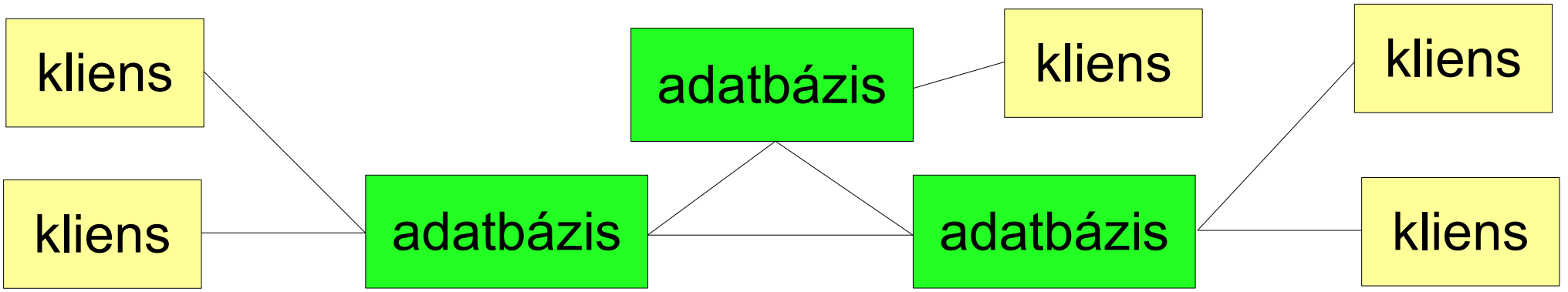



Több felhasználós környezet

Központi adatbázis




Osztott adatbázis



# Tranzakciók

A tranzakció egy egység, mely során különböző adatokat olvasunk az adatbázisból illetve módosítunk

A tranzakciónak konzisztens adatbázis képet kell látnia

A tranzakció végrehajtása során az adatbázis időlegesen inkonzisztens lehet

A tranzakció befejezésével (Commit) az adatbázisnak konzisztens állapotba kell kerülnie

A tranzakció véglegesítése után a módosításokat rendszerhiba esetén is meg kell őriznie az adatbázis-kezelőnek

Több tranzakció hajtható végre párhuzamosan

Két kezelendő probléma:

- Hardver hiba, rendszer leállás
- Több tranzakció konkurens végrehajtása

# Tranzakció példa

1. **read**(*A*)
2. *A* := *A* – 50
3. **write**(*A*)
4. **read**(*B*)
5. *B* := *B* + 50
6. **write**(*B*)

Az **A** számláról 50 forintot utalunk át a **B** számlára

**Atomikus művelet** szükséges – ha a tranzakció a 3. és 6. lépés között megszakad a módosítás nem kerülhet be az adatbázisba, mert inkonzisztenciához vezet (mindent vagy semmit)

**Konzisztencia** feltétel – a tranzakció végrehajtása után nem változik A+B értéke (konzisztens állapot fenntartása)

**Elszigeteltség** feltétel – ha a tranzakció 3. és 6. lépése között egy másik tranzakció is dolgozik a részlegesen aktualizált adatbázison, akkor inkonzisztens képet fog kapni

- Az elszigeteltség megvalósítható a tranzakció egymás utáni végrehajtásával
- A tranzakciók konkurens végrehajtása előnyösebb, de ehhez további feltételek szükségesek

**Állandóság** feltétel – ha a tranzakció sikeresen befejeződött, akkor a tranzakció eredményét az adatbázis-kezelő összeomlása esetén is meg kell őrizni

**ACID** (Atomicity, Cosistency, Isolation, Durability)

# Tranzakció állapotok

Aktív – a tranzakció végrehajtása során
Részlegesen véglegesített – a tranzakció utolsó utasításának végrehajtása után
Sikertelen – a normál végrehajtás nem folytatható
Megszakított – az adatbázis tartalma visszaáll a tranzakció megkezdése előtti konzisztens állapotba (Rollback)
Véglegesített – sikeres befejezés (Commit)

Konkurens tranzakciók – tábla és rekord szintű zárolások
A tranzakció által zárolt adatokat más tranzakciók nem módosíthatják
Más tranzakciók a zárolt rekordok tranzakció megkezdése előtti konzisztens állapotát látják (read committed)

# Tranzakció kezdés és lezárás

Minden DDL utasítás előtt a folyamatban lévő tranzakció automatikusan véglegesítésre kerül (Commit)

Minden DDL utasítás automatikusan véglegesítésre kerül DDL utasítás nem vonható vissza (Rollback)

Minden DML utasítás egy tranzakciót indít, ha nincs már folyamatban egy tranzakció

A tranzakció tulajdonosa a tranzakció közbeni pillanatnyi, esetleg inkonzisztens adatokat látja

A többi felhasználó mindig konzisztens képet lásson (pl. SELECT)

# Holtpont kezelés

Két tranzakciót hajtunk végre párhuzamosan
T1. **A** számláról 50 forintot utalunk **B** számlára
T2. **B** számláról 100 forintot utalunk **A** számlára

idő

T1 - **A** számla zárolás
T1 - **A** := **A** - 50
T2 - **B** számla zárolás
T2 - **B** := **B** -100
T1 - **B** számla zárolására várakozás
T2 – **A** számla zárolására várakozás
**Holtpont**

Holtpont megelőzés, kezelés
Az egyik vagy mindkét tranzakció megszakításával (Rollback)

# Hibalehetőségek

**Logikai hibák**: a tranzakció nem fejezhető be valamilyen belső hiba miatt
**Rendszer hibák**: az adatbázis-kezelő megszakítja a tranzakciót, pl. holtpont miatt
**Rendszer összeomlás**: áramszünet, hardver vagy szoftver hiba következtében
Feltételezés: az összeomlás következtében a lemez tartalom nem sérül, de a memória tartalom (bufferek) elveszik

**Lemez összeomlás**: pl. lemez fej meghibásodás, hiba detektálható, a lemezek ellenőrző összegeket használnak, hibatűrő rendszerek pl. RAID 1

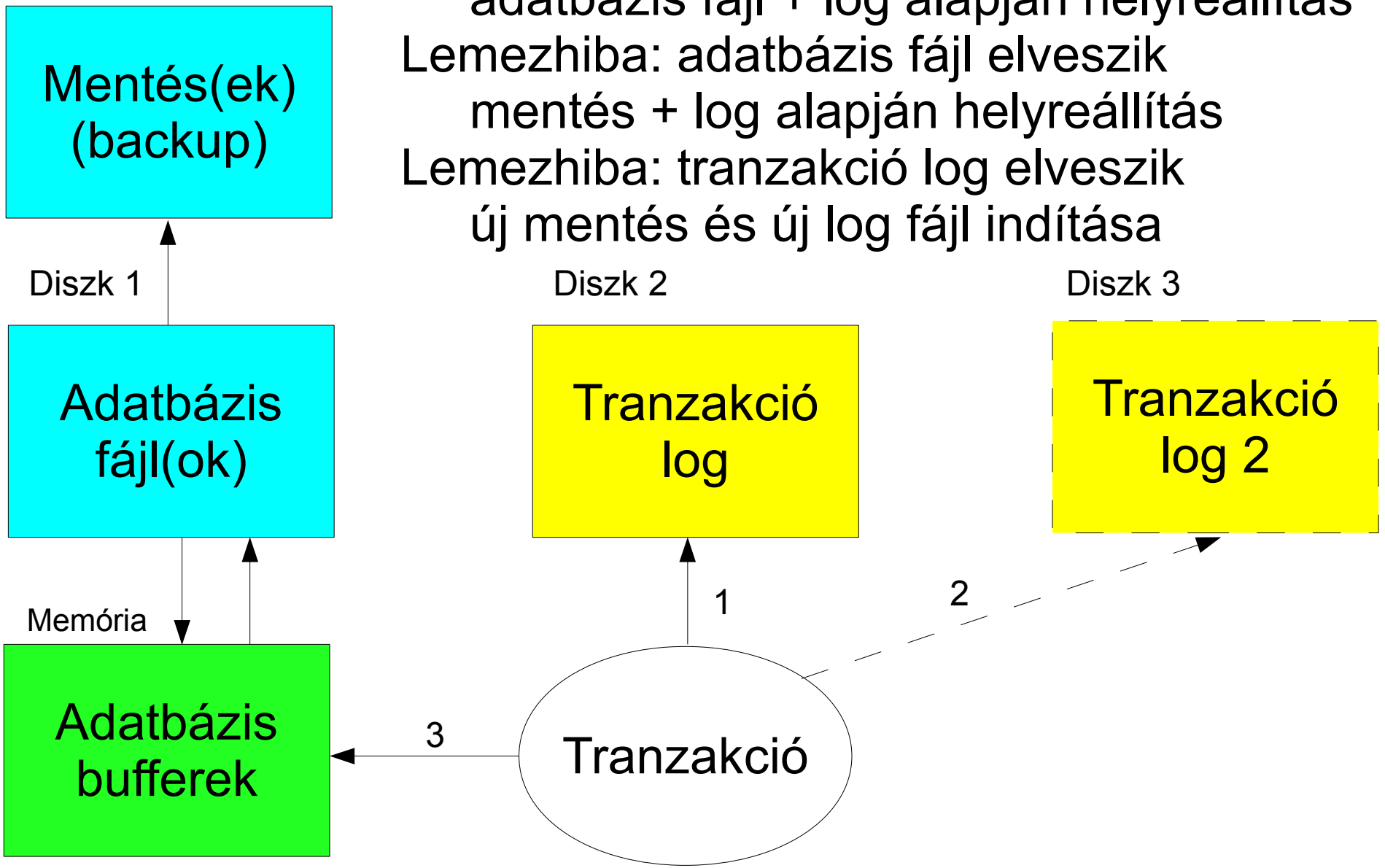
# Helyreállítási algoritmusok

A helyreállítási algoritmusok célja, hogy rendszerösszeomlás után biztosítsák, visszaállítsák az adatbázis konzisztens állapotát

Két részből tevődik össze
1. A normál tranzakciók során végrehajtott műveletek, hogy elég információnk legyen az összeomlás utáni helyreállításhoz, tranzakció log fájl
2. Az összeomlás után végrehajtott műveletek az adatbázis konzisztens állapotának helyreállításához
Összeomlás után a tranzakció log fájl elemzése:
Megszakadt tranzakciók visszavonása
Véglegesített, de az adatbázisban nem megjelenő módosítások újbóli végrehajtása

# Tranzakciók és fájlok

Hardver/szoftver hiba: bufferek elvesznek
adatbázis fájl + log alapján helyreállítás
Lemezhiba: adatbázis fájl elveszik
mentés + log alapján helyreállítás
Lemezhiba: tranzakció log elveszik
új mentés és új log fájl indítása



# Kétfázisú commit

Osztott adatbázisok esetén

Egy vezérgép és több alárendelt  (időtúllépés)

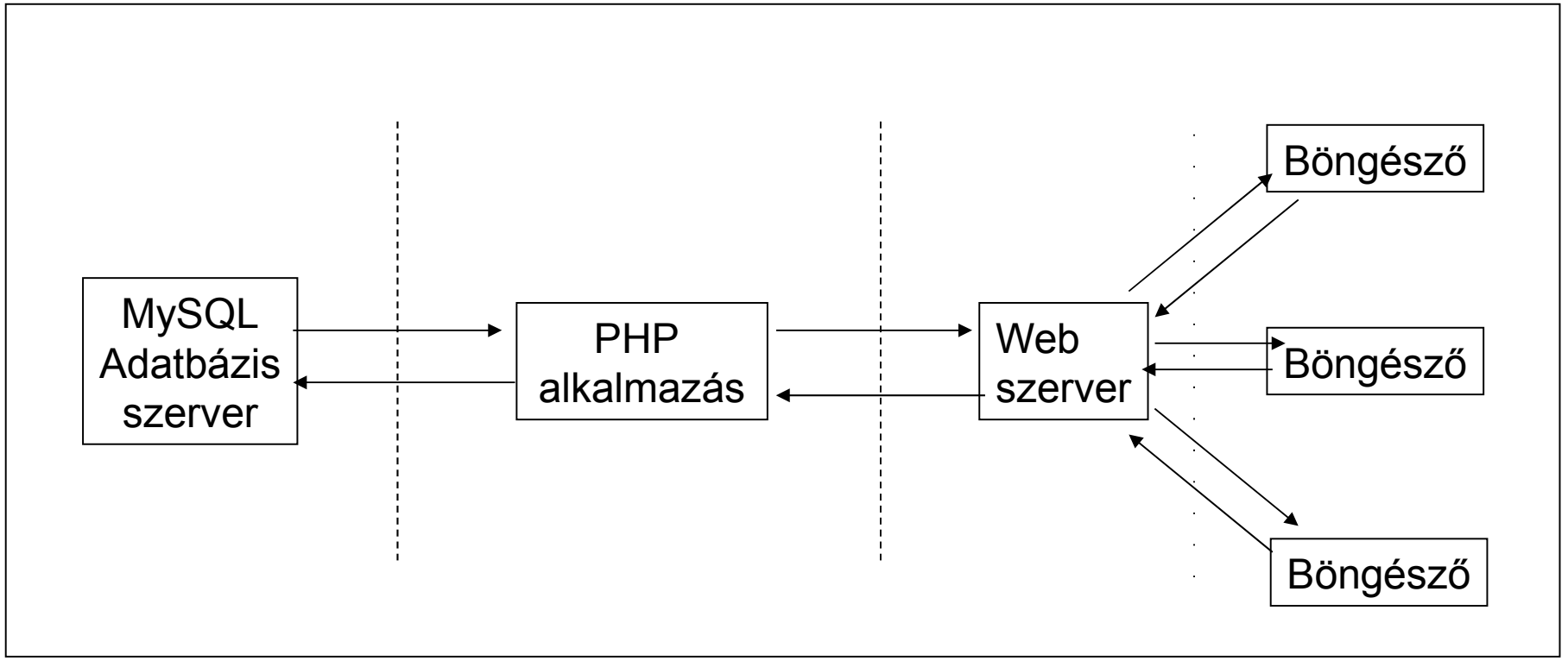
1. fázis
Vezérgépről végrehajtandó parancsok az alárendelteknek
Alárendeltek végrehajtják (nincs még commit)
Alárendeltek visszajeleznek a vezérgépnek sikeres/sikertelen

2. fázis (sikeres első fázis)
Vezérgépről  commit üzenet az alárendeltekhez
Alárendeltek commit és zárolások feloldása
Alárendeltek visszajeleznek a vezérgépnek sikeres/sikertelen
Vezérgép is commit

2. fázis (sikertelen első fázis)
Vezérgépről rollback üzenet a kliensekhez
Alárendeltek rollback
Alárendeltek visszajeleznek a vezérgépnek sikeres/sikertelen
Vezérgép is rollback

# Internetes adatbázisok

Három rétegű megoldás (kliens - szerver)




Adat réteg

Alkalmazás réteg (üzleti logika)

Prezentációs réteg (felhasználói interfész)

# Kliens – szerver megoldás

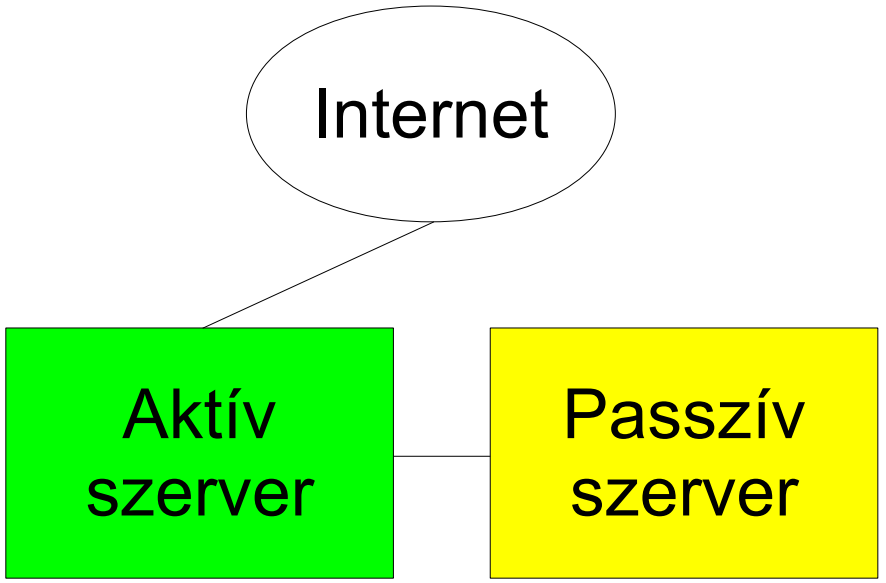
**Előnyök:**
Minden adat a szerveren található: nagyobb biztonság
Szoftver és adatkarbantartás egyszerűbb

**Hátrányok:**
Szerver kiesés blokkolja a szolgáltatást
Szerver túlterhelhető

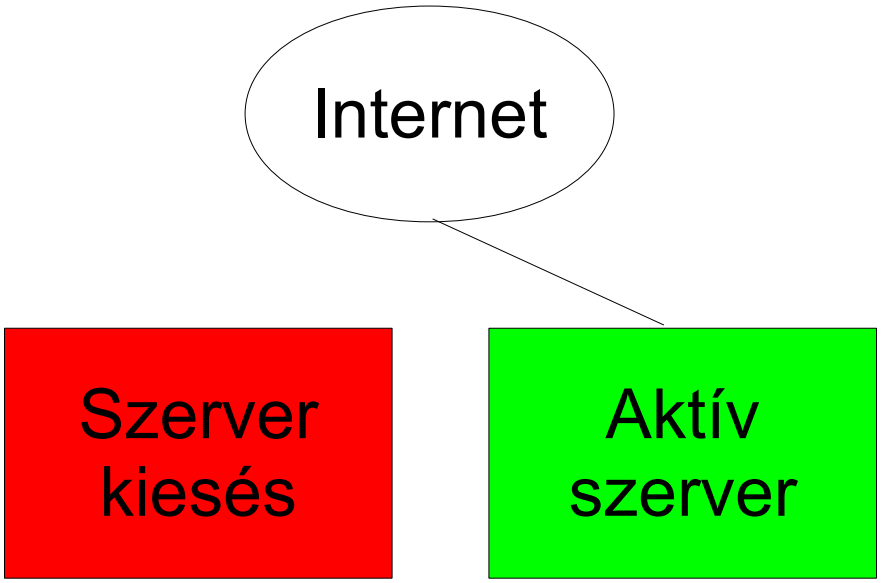
**Hátrányok kiküszöbölése**
Szerver kiesés – magas rendelkezésre állás
Redundáns  szerverek (aktív/passzív)
Túlterhelés - terhelés szétosztás (load balancing)
Szerver klaszter, több szerver között osztjuk meg a feladatot

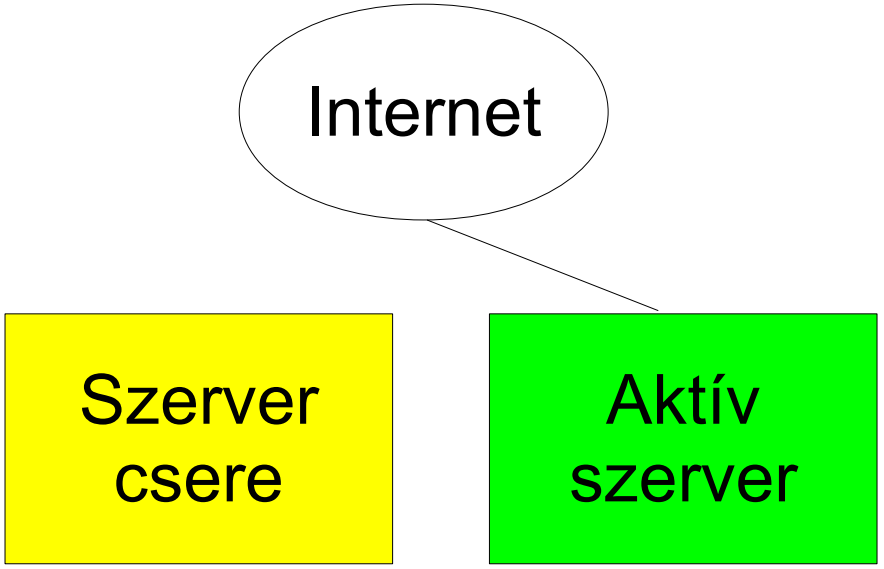
Vékony és vastag kliens megoldások

# Redundáns szerver

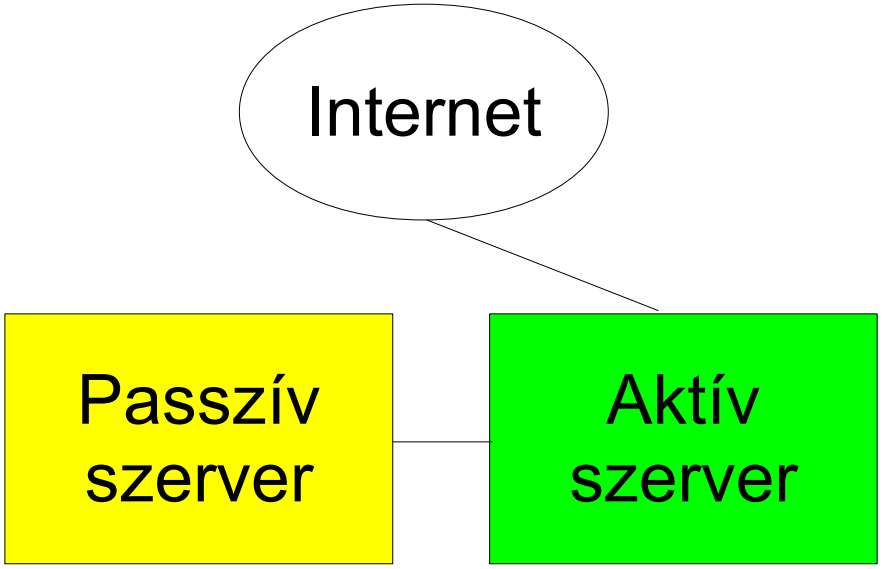



szinkronizálás




adatbázis tükrözés




szinkronizálás

# Terhelés szétosztás