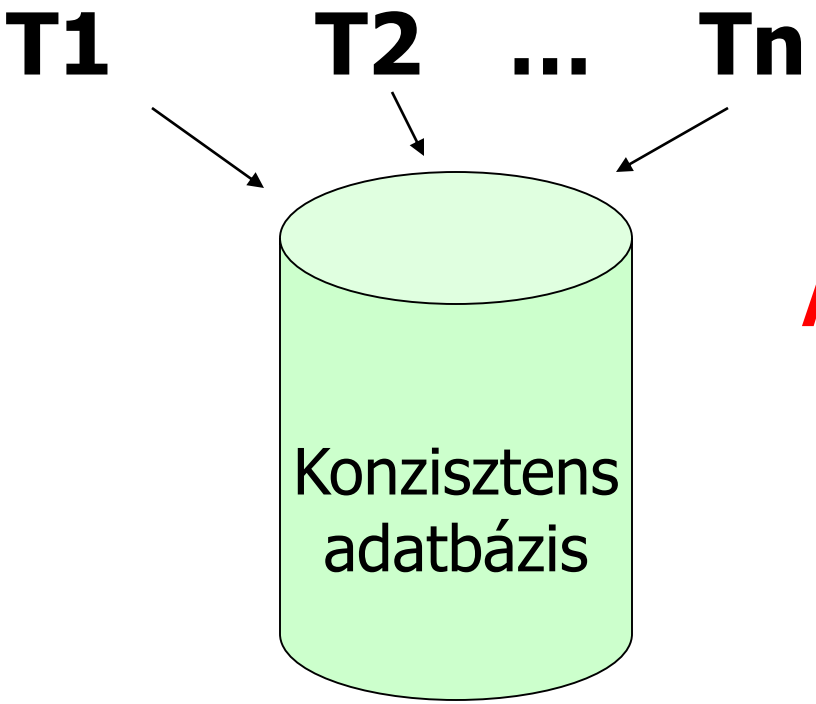
# Konkurenciavezérlés, sorbarendezhetőség, konfliktus-sorbarendezhetőség

# Egyszerre több tranzakció is ugyanazt az adatbázist használja.




**Az adatbázisnak konzisztensnek kell maradnia!**

**A tranzakciók közötti egymásra hatás az adatbázis-állapot inkonzisztenssé válását okozhatja, még akkor is, amikor a tranzakciók külön-külön megőrzik a konzisztenciát, és rendszerhiba sem történt.**

# A konkurenciavezérlés

- **A tranzakciós lépések szabályozásának feladatát az adatbázis-kezelő rendszer *ütemező* (scheduler) része végzi.**

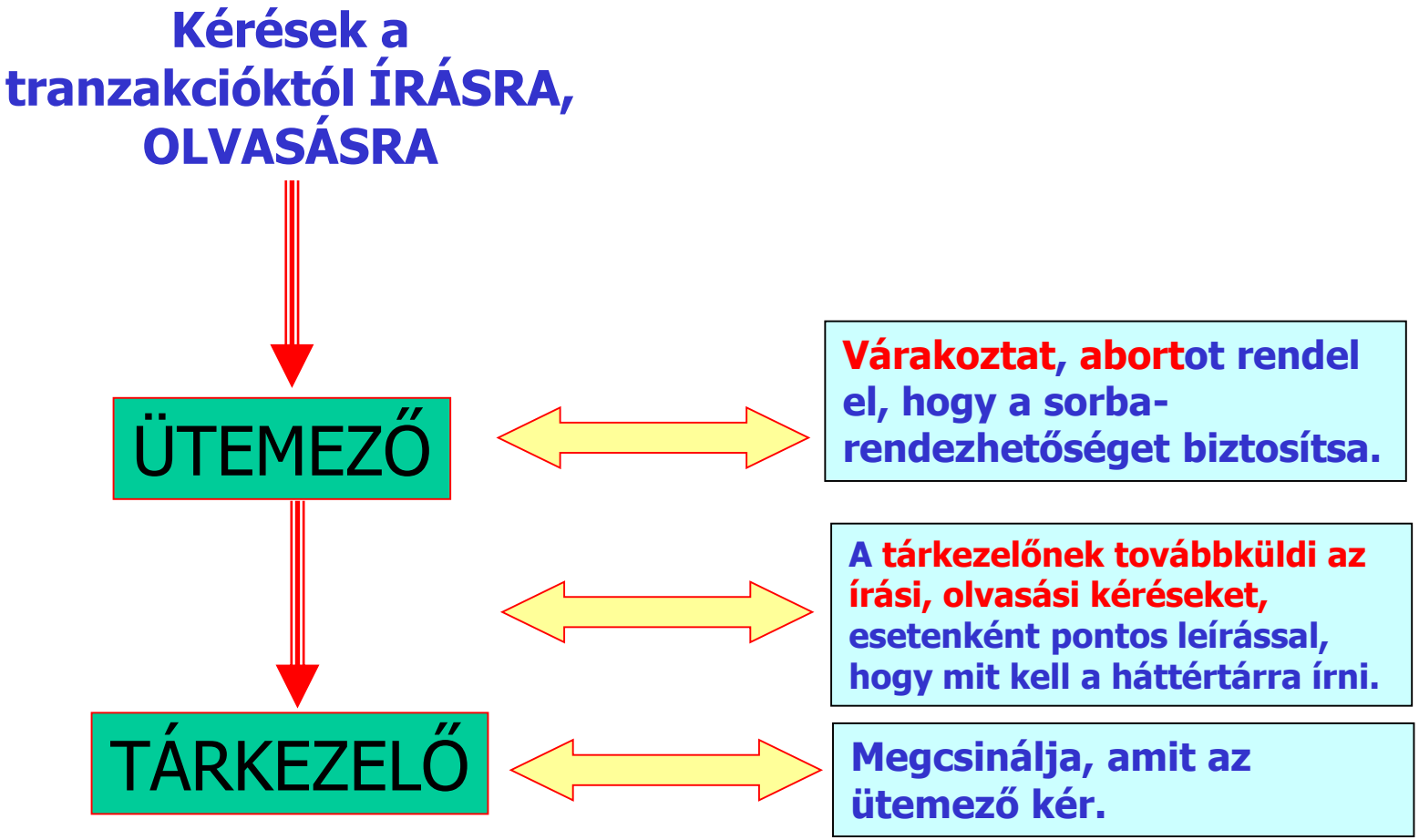
- **Azt az általános folyamatot, amely biztosítja, hogy a tranzakciók egyidejű végrehajtása során megőrizzék a konzisztenciát, *konkurenciavezérlésnek* (concurrency control) nevezzük.**

- **Amint a tranzakciók az adatbáziselemek olvasását és írását kérik, ezek a kérések az ütemezőhöz kerülnek, amely legtöbbször közvetlenül végrehajtja azokat. Amennyiben a szükséges adatbáziselem nincs a pufferben, először a pufferkezelőt hívja meg.**

- **Bizonyos esetekben azonban nem biztonságos azonnal végrehajtani a kéréseket. Az ütemezőnek ekkor késleltetnie kell a kérést, sőt bizonyos esetben abortálnia kell a kérést kiadó tranzakciót.**

- **Az *ütemezés* (schedule) egy vagy több tranzakció által végrehajtott lényeges műveletek időrendben vett sorozata, amelyben az egy tranzakcióhoz tartozó műveletek sorrendje megegyezik a tranzakcióban megadott sorrenddel.**

# Az ütemező és a tárkezelő együttműködése



• A konkurenciakezelés szempontjából a lényeges olvasási és írási műveletek a központi memória puffereiben történnek, nem pedig a lemezen. Tehát csak a READ és WRITE műveletek sorrendje számít, amikor a konkurenciával foglalkozunk, az INPUT és OUTPUT műveleteket figyelmen kívül hagyjuk.

| $T_1$ | $T_2$ |
|---|---|
| READ(A,t) | READ(A,s) |
| t := t+100 | s := s*2 |
| WRITE(A,t) | WRITE(A,s) |
| READ(B,t) | READ(B,s) |
| t := t+100 | s := s*2 |
| WRITE(B,t) | WRITE(B,s) |

**Konzisztencia:**

**A=B**

**Egymás után futtatva, megőrzik a konzisztenciát.**

# Soros ütemezések

**A két tranzakciónak két soros ütemezése van, az egyikben T1 megelőzi T2-t, a másikban T2 előzi meg T1-et**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| READ(B,t) | | | 25 |
| t := t+100 | | | |
| WRITE(B,t) | | | 125 |
| | READ(A,s) | 125 | |
| | s := s*2 | | |
| | WRITE(A,s) | 250 | |
| | READ(B,s) | | 125 |
| | s := s*2 | | |
| | WRITE(B,s) | | 250 |

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| | READ(A,s) | 25 | |
| | s := s*2 | | |
| | WRITE(A,s) | 50 | |
| | READ(B,s) | | 25 |
| | s := s*2 | | |
| | WRITE(B,s) | | 50 |
| READ(A,t) | | 50 | |
| t := t+100 | | | |
| WRITE(A,t) | | 150 | |
| READ(B,t) | | | 50 |
| t := t+100 | | | |
| WRITE(B,t) | | | 150 |

**Mindkét soros ütemezés konzisztens (A=B) adatbázist eredményez, bár a két ütemezésben a végeredmények különböznek (250, illetve 150).**

# Sorbarendezhető ütemezések

**Az első ütemezés egy sorbarendezhető, de nem soros. A hatása megegyezik a (T1, T2) soros ütemezés hatásával: tetszőleges konzisztens kiindulási állapotra: A = B = c-ből kiindulva A-nak is és B-nek is 2(c + 100) lesz az értéke, tehát a konzisztenciát mindig megőrizzük.**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*2 | | |
| | WRITE(A,s) | 250 | |
| READ(B,t) | | | 25 |
| t := t+100 | | | |
| WRITE(B,t) | | | 125 |
| | READ(B,s) | | 125 |
| | s := s*2 | | |
| | WRITE(B,s) | | 250 |

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*2 | | |
| | WRITE(A,s) | 250 | |
| | READ(B,s) | | 25 |
| | s := s*2 | | |
| | WRITE(B,s) | | 50 |
| READ(B,t) | | | 50 |
| t := t+100 | | | |
| WRITE(B,t) | | | 150 |

# Sorbarendezhető ütemezések

**A második példában szereplő ütemezés nem sorbarendezhető. A végeredmény sosem konzisztens A := 2(A + 100), B := 2B + 100, így nem lehet a hatása soros ütemezéssel megegyező. Az ilyen viselkedést a különböző konkurenciavezérlési technikákkal el kell kerülnünk.**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*2 | | |
| | WRITE(A,s) | 250 | |
| READ(B,t) | | | 25 |
| t := t+100 | | | |
| WRITE(B,t) | | | 125 |
| | READ(B,s) | | 125 |
| | s := s*2 | | |
| | WRITE(B,s) | | 250 |

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*2 | | |
| | WRITE(A,s) | 250 | |
| | READ(B,s) | | 25 |
| | s := s*2 | | |
| | WRITE(B,s) | | 50 |
| READ(B,t) | | | 50 |
| t := t+100 | | | |
| WRITE(B,t) | | | 150 |

# A tranzakció szemantikájának hatása

**Ez az ütemezés elvileg sorbarendezhető, de csak T2 speciális aritmetikai művelete (1-gyel szorzás) miatt. Az, hogy mivel szorzunk, lehet, hogy egy bonyolult függvény eredményeképpen dől el. Az ütemező csak az írási, olvasási műveleteket figyelve pesszimista alapon dönt a sorbarendezhetőségről:**

- **Ha T tudna A-ra olyan hatással lenni, hogy az adatbázis-állapot inkonzisztenssé váljon, akkor T ezt meg is teszi.**

**A feltevés miatt az ütemező szerint ez az ütemezés nem lesz sorbarendezhető.**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*1 | | |
| | WRITE(A,s) | 125 | |
| | READ(B,s) | | 25 |
| | s := s*1 | | |
| | WRITE(B,s) | | 25 |
| READ(B,t) | | | 25 |
| t := t+100 | | | |
| WRITE(B,t) | | | 125 |

# A tranzakciók és az ütemezések jelölése

- **Csak a tranzakciók által végrehajtott olvasások és írások számítanak!**

$T_1: r_1(A); w_1(A); r_1(B); w_1(B);$

$T_2: r_2(A); w_2(A); r_2(B); w_2(B);$

- **Az ütemezés jelölése:**

$S: r_1(A); w_1(A); r_2(A); w_2(A); r_1(B); w_1(B); r_2(B); w_2(B);$

1. Az $r_i(X)$ vagy $w_i(X)$ azt jelenti, hogy a $T_i$ tranzakció olvassa, illetve írja az $x$ adatbáziselemet.
2. Egy $T_i$ *tranzakció* az $i$ indexű műveletekből álló sorozat.
3. Egy $S$ *ütemezés* olyan műveletek sorozata, amelyben minden $T_i$ tranzakcióra teljesül, hogy $T_i$ műveletei ugyanabban a sorrendben fordulnak elő $S$-ben, mint a $T_i$ -ben. $S$ a tranzakciók műveleteinek *átlapolása* (interleaving).

# Konfliktus-sorbarendezhetőség

- **Elégséges feltétel: biztosítja egy ütemezés sorba-rendezhetőségét.**

- **A forgalomban lévő rendszerek ütemezői a tranzakciók sorbarendezhetőségére általában ezt az erősebb feltételt biztosítják, amelyet *konfliktus-sorbarendezhetőségnek* nevezünk.**

- **A *konfliktus* (conflict) vagy *konfliktuspár* olyan egymást követő műveletpár az ütemezésben, amelynek ha a sorrendjét felcseréljük, akkor legalább az egyik tranzakció viselkedése megváltozhat.**

# Nincs konfliktus

- Legyen $T_i$ és $T_j$ két különböző tranzakció ($i \neq j$).

1. $r_i(X); r_j(Y)$ sohasem konfliktus, még akkor sem, ha $X = Y$,
   - mivel egyik lépés sem változtatja meg az értékeket.

2. $r_i(X); w_j(Y)$ nincs konfliktusban, ha $X \neq Y$,
   - mivel $T_j$ írhatja $Y$-t, mielőtt $T_i$ beolvasta $X$-et, $X$ értéke ettől ugyanis nem változik. Annak sincs hatása $T_j$-re, hogy $T_i$ olvassa $X$-et, ugyanis ez nincs hatással arra, hogy milyen értéket ír $T_j$ $Y$-ba.

3. $w_i(X); r_j(Y)$ nincs konfliktusban, ha $X \neq Y$,
   - ugyanazért, amiért a 2. pontban.

4. $w_i(X); w_j(Y)$ sincs konfliktusban, ha $X \neq Y$.

# Konfliktus

- **Három esetben nem cserélhetjük fel a műveletek sorrendjét:**

**a)** $r_i(X); w_i(Y)$ **konfliktus,**

- mivel egyetlen tranzakción belül a műveletek sorrendje rögzített, és az adatbázis-kezelő ezt a sorrendet nem rendezheti át.

**b)** $w_i(X); w_j(X)$ **konfliktus,**

- mivel X értéke az marad, amit $T_j$ számolt ki. Ha felcseréljük a sorrendjüket, akkor pedig X-nek a $T_i$ által kiszámolt értéke marad meg. A pesszimista feltevés miatt a $T_i$ és a $T_j$ által kiírt értékek lehetnek különbözőek, és ezért az adatbázis valamelyik kezdeti állapotára különbözni fognak.

**c)** $r_i(X); w_j(X)$ **és** $w_i(X); r_j(X)$ **is konfliktus.**

- Ha átvisszük $w_j(X)$-et $r_i(X)$ elé, akkor a $T_i$ által olvasott X-beli érték az lesz, amit a $T_j$ kiírt, amiről pedig feltételeztük, hogy nem szükségképpen egyezik meg X korábbi értékével. Tehát $r_i(X)$ és $w_j(X)$ sorrendjének cseréje befolyásolja, hogy $T_i$ milyen értéket olvas X-ből, ez pedig befolyásolja $T_i$ működését.

# Konfliktus-sorbarendezhetőség

- **ELV: nem konfliktusos cserékkel az ütemezést megpróbáljuk soros ütemezéssé átalakítani. Ha ezt meg tudjuk tenni, akkor az eredeti ütemezés sorbarendezhető volt, ugyanis az adatbázis állapotára való hatása változatlan marad minden nem konfliktusos cserével.**

- **Azt mondjuk, hogy két ütemezés *konfliktusekvivalens* (conflict-equivalent), ha szomszédos műveletek nem konfliktusos cseréinek sorozatával az egyiket átalakíthatjuk a másikká.**

- **Azt mondjuk, hogy egy ütemezés *konfliktus-sorbarendezhető* (conflict-serializable schedule), ha konfliktusekvivalens valamely soros ütemezéssel.**

- **A konfliktus-sorbarendezhetőség elégséges feltétel a sorbarendezhetőségre, vagyis egy konfliktus-sorbarendezhető ütemezés sorbarendezhető ütemezés is egyben.**

- **A konfliktus-sorbarendezhetőség nem szükséges ahhoz, hogy egy ütemezés sorbarendezhető legyen, mégis általában ezt a feltételt ellenőrzik a forgalomban lévő rendszerek ütemezői, amikor a sorbarendezhetőséget kell biztosítaniuk.**

- ***Példa.*** **Legyen az ütemezés a következő:**

  $r_1(A); w_1(A); r_2(A); w_2(A); r_1(B); w_1(B); r_2(B); w_2(B);$

- **Azt állítjuk, hogy ez az ütemezés konfliktus-sorbarendezhető. A következő cserékkel ez az ütemezés átalakítható a $(T_1, T_2)$ soros ütemezéssé, ahol az összes $T_1$-beli művelet megelőzi az összes $T_2$-beli műveletet:**

1. $r_1(A); w_1(A); r_2(A); \underline{w_2(A); r_1(B);} w_1(B); r_2(B); w_2(B);$
2. $r_1(A); w_1(A); \underline{r_2(A); r_1(B);} w_2(A); w_1(B); r_2(B); w_2(B);$
3. $r_1(A); w_1(A); r_1(B); r_2(A); \underline{w_2(A); w_1(B);} r_2(B); w_2(B);$
4. $r_1(A); w_1(A); r_1(B); \underline{r_2(A); w_1(B);} w_2(A); r_2(B); w_2(B);$
5. $r_1(A); w_1(A); r_1(B); w_1(B); r_2(A); w_2(A); r_2(B); w_2(B);$

# Nem konfliktus-sorbarendezhető ütemezés

S : $r_1(A); w_1(A); r_2(A); w_2(A); r_2(B); w_2(B); r_1(B); w_1(B);$

**Ez az ütemezés nem konfliktus-sorbarendezhető, ugyanis A-t T1 írja előbb, B-t pedig T2. Mivel sem A írását, sem B írását nem lehet átrendezni, semmilyen módon nem kerülhet T1 összes művelete T2 összes művelete elé, sem fordítva.**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| READ(A,t) | | 25 | |
| t := t+100 | | | |
| WRITE(A,t) | | 125 | |
| | READ(A,s) | 125 | |
| | s := s*1 | | |
| | WRITE(A,s) | 125 | |
| | READ(B,s) | | 25 |
| | s := s*1 | | |
| | WRITE(B,s) | | 25 |
| READ(B,t) | | | 25 |
| t := t+100 | | | |
| WRITE(B,t) | | | 125 |

# Sorbarendezhető, de nem konfliktus-sorbarendezhető ütemezés

- Tekintsük a $T_1$, $T_2$ és $T_3$ tranzakciókat és egy **soros** ütemezésüket:

  $S_1$: $w_1(Y)$; $w_1(X)$; $w_2(Y)$; $w_2(X)$; $w_3(X)$;

- Az $S_1$ ütemezés X értékének a $T_3$ által írt értéket, Y értékének pedig a $T_2$ által írt értéket adja. Ugyanezt végzi a következő ütemezés is:

  $S_2$: $w_1(Y)$; $w_2(Y)$; $w_2(X)$; $w_1(X)$; $w_3(X)$;

- Intuíció alapján átgondolva annak, hogy $T_1$ és $T_2$ milyen értéket ír X-be, nincs hatása, ugyanis $T_3$ felülírja X értékét. **Emiatt $S_1$ és $S_2$ X-nek is és Y-nak is ugyanazt az értéket adja.** Mivel $S_1$ soros ütemezés, és $S_2$-nek bármely adatbázis-állapotra ugyanaz a hatása, mint $S_1$-nek, ezért **$S_2$ sorbarendezhető**.

- Ugyanakkor mivel nem tudjuk felcserélni $w_1(X)$-et $w_2(X)$-szel, így cseréken keresztül nem lehet $S_2$-t valamelyik soros ütemezéssé átalakítani. Tehát $S_2$ sorbarendezhető, de **nem konfliktus-sorbarendezhető.**

## Összefoglalás

- Konkurenciakezelés
  - Ütemezés, ütemező feladatai
  - Konzisztens adatbázis eredményező ütemezések
    - Sorbarendezhető (hatása megegyezik a tanzakciók valamelyien sorrendű végrehajtásával)
    - Konfliktus-sorbarendezhető (nem-konfliktusos párok cseréjével el lehet jutni sorbarendezett ütemezésig)
  - A konfliktus-sorbarendezhetőség elégséges, de nem szükséges feltétel a sorbarendezhetőség igazolásához.