## 9.2. Konfliktus sorbarendezhetőség

Most egy olyan elégséges feltételt adunk meg, mely biztosítja egy ütemezés sorbarendezhetőségét. A piaci rendszerek ütemezői a tranzakciók sorbarendezhetőségére általában ezt az erősebb feltételt biztosítják, amelyet „konfliktus sorbarendezhetőségnek” nevezünk. Ez a *konfliktus* (conflict) fogalmon alapul: amely olyan egymást követő műveletpár az ütemezésben, amelynek ha a sorrendjét felcseréljük, akkor legalább az egyik tranzakció viselkedése megváltozhat.

### 9.2.1. Konfliktusok

Azzal kezdjük, hogy vegyük észre a legtöbb műveletpár *nincs* konfliktusban a fenti értelemben. Ugyanis, tételezzük fel, hogy *Ti* és *Tj* különböző tranzakciók, vagyis *i*  *j*.

1. *ri*(*X*); *rj*(*Y*) sohasem konfliktus, még akkor sem, ha *X* = *Y*. Ennek az az oka, hogy egyik lépés sem változtatja meg az értékeket.

2. *ri*(*X*); *wj*(*Y*) nincs konfliktusban, feltéve, ha *X*  *Y*. Ennek az az oka, hogy a *Tj* írhatja az *Y*-t, mielőtt a *Ti* beolvasta az *X*-et, az *X* értéke ugyanis ettől nem változik. Annak sincs hatása a *Tj*-re, hogy a *Ti* olvassa az *X*-et, ugyanis ez nincs hatással arra, hogy milyen értéket ír be a *Tj* az *Y*-ba.

3. *wi*(*X*); *rj*(*Y*) nincs konfliktusban, ha *X*  *Y*, ugyanazért, mint a 2.

4. Szintén hasonlóan *wi*(*X*); *wj*(*Y*) sincs konfliktusban mindaddig, amíg *X*  *Y*.

Másrészt három esetben nem cserélhetjük fel a műveletek sorrendjét:

a) Ugyanannak a tranzakciónak két művelete, pl. *ri*(*X*); w*i*(*Y*) konfliktus. Ennek az az oka, hogy egyetlen tranzakción belül a műveletek sorrendje rögzített, és az adatbázis-kezelő rendszer ezt a sorrendet nem rendezheti át újra.

b) Különböző tranzakciók ugyanarra az adatbáziselemre való írása konfliktus. Vagyis *wi*(*X*); *wj*(*X*) konfliktus. Ennek az az oka, mint már írtuk, hogy az *X* értéke az marad, amelyet a *Tj* számolt ki. Ha felcseréljük a sorrendjüket, hogy *wj*(*X*); *wi*(*X*), akkor az *X*-nek a *Ti* által kiszámított értéke marad meg. Az a feltevésünk, hogy „nincs egybeesés”, azt adja, hogy a *Ti* és a *Tj* által írt értékek lehetnek különbözőek, és ezért az adatbázis valamelyik kezdeti állapotára különbözni fognak.

c) Különböző tranzakcióknak ugyanabból az adatbáziselemből való olvasása és írása is konfliktus. Vagyis *ri*(*X*); *wj*(*X*) konfliktus, és *wi*(*X*); *rj*(*X*) is konfliktus. Ha átvisszük *wj*(*X*)-et *ri*(*X*) elé, akkor a *Ti* által olvasott *X*-beli érték az lesz, amelyet a *Tj* írt, amiről pedig feltételeztük, hogy nem szükségképpen egyezik meg az *X* korábbi értékével. Tehát *ri*(*X*) és *wj*(*X*) sorrendjének cseréje befolyásolja, hogy *Ti* milyen értéket olvas *X*-ből, ez pedig befolyásolja a *Ti* működését.

Levonhatjuk a következtetést, hogy különböző tranzakciók bármely két műveletének sorrendje felcserélhető, hacsak nem

1. Ugyanarra az adatbáziselemre vonatkoznak, és

2. Legalább az egyik művelet írás.

Ezt az elvet kiterjesztve tetszőleges ütemezést véve annyi nem konfliktusos cserét készíthetünk, amennyit csak kívánunk, abból a célból, hogy az ütemezést soros ütemezéssé alakítsuk át. Ha ezt meg tudjuk tenni, akkor az eredeti ütemezés sorba rendezhető volt, ugyanis az adatbázis állapotára való hatása változatlan marad minden nem konfliktusos cserével.

Azt mondjuk, hogy két ütemezés *konfliktus ekvivalens* (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 konfliktus ekvivalens valamely soros ütemezéssel. Megjegyezzük, hogy a konfliktus sorbarendezhetőség elégséges feltétele a sorbarendezhetőségnek, vagyis egy konfliktus sorbarendezhető ütemezés sorba rendezhető ütemezés is egyben. Azonban a konfliktus-sorbarendezhetőség nem szükséges ahhoz, hogy egy ütemezés sorba rendezhető legyen, mégis általában ezt a feltételt ellenőrzik a piaci rendszerek ütemezői, amikor a sorbarendezhetőséget kell biztosítaniuk.

**9.6. példa:** Legyen az ütemezés

*r*1(*A*); *w*1(*A*); *r*2(*A*); *w*2(*A*); *r*1(*B*); *w*1(*B*); *r*2(*B*); *w*2(*B*);

a 9.5. példából. Azt állítjuk, hogy ez az ütemezés konfliktus sorbarendezhető. A 9.8. ábrán látható a cserék sorozata, amellyel 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. Aláhúztuk azokat a szomszédos műveletpárokat, amelyeket felcserélünk az egyes lépésekben. 

*r*1(*A*); *w*1(*A*); *r*2(*A*); *w*2(*A*); *r*1(*B*); *w*1(*B*); *r*2(*B*); *w*2(*B*);

*r*1(*A*); *w*1(*A*); *r*2(*A*); *r*1(*B*); *w*2(*A*); *w*1(*B*); *r*2(*B*); *w*2(*B*);

*r*1(*A*); *w*1(*A*); *r*1(*B*); *r*2(*A*); *w*2(*A*); *w*1(*B*); *r*2(*B*); *w*2(*B*);

*r*1(*A*); *w*1(*A*); *r*1(*B*); *r*2(*A*); *w*1(*B*); *w*2(*A*); *r*2(*B*); *w*2(*B*);

*r*1(*A*); *w*1(*A*); *r*1(*B*); *w*1(*B*); *r*2(*A*); *w*2(*A*); *r*2(*B*); *w*2(*B*);

**9.8. ábra.** Egy konfliktus sorbarendezhető ütemezés szomszédos műveletek felcserélésével való átalakítása soros ütemezéssé

### 9.2.2. Megelőzési gráfok és teszt a konfliktus sorbarendezhetőségre

Viszonylag könnyű megvizsgálnunk egy *S* ütemezést, és eldöntenünk, hogy konfliktus sorbarendezhető-e vagy nem. Az az alapötlet, hogy ha valahol konfliktusban álló műveletek szerepelnek az *S*-ben, ezeket a műveleteket végrehajtó tranzakcióknak ugyanabban a sorrendben kell előfordulniuk a konfliktus ekvivalens soros ütemezésekben, mint ahogyan az *S*-ben voltak. Tehát a konfliktusban álló műveletpárok megszorítást adnak a feltételezett konfliktus ekvivalens soros ütemezésben a tranzakciók sorrendjére. Ha ezek a megszorítások nem mondanak egymásnak ellent, akkor találhatunk konfliktus ekvivalens soros ütemezést. Ha pedig ellentmondanak egymásnak, akkor tudjuk, hogy nincs ilyen soros ütemezés.

Adott a *T*1 és *T*2 tranzakcióknak, esetleg további tranzakcióknak, egy *S* ütemezése. Azt mondjuk, hogy *T*1 *megelőzi T*2-t, és *T*1 <*S* *T*2-vel jelöljük, ha van a *T*1-ben olyan *A*1 művelet, és a *T*2-ben olyan *A*2, hogy

1. *A*1 megelőzi *A*2-t az *S*-ben,

2. *A*1 és *A*2 ugyanarra az adatbáziselemre vonatkoznak, és

3. *A*1 és *A*2 közül legalább az egyik írás művelet.

Megjegyezzük, hogy ezek pontosan azok a feltételek, amikor nem lehet felcserélni az *A*1 és *A*2 sorrendjét. Tehát, *A*1 az *A*2 előtt szerepel bármely az *S*-sel konfliktus ekvivalens ütemezésben. Ennek eredményeként, ha ezek közül az ütemezések közül az egyik soros ütemezés, akkor ebben *T*1-nek meg kell előznie *T*2-t.

Ezeket a megelőzéseket a *megelőzési gráfban* (precedence graph) összegezhetjük. A megelőzési gráf csomópontjai az *S* ütemezés tranzakciói. Ha a tranzakciókat *Ti*-vel jelöljük az *i* függvényében, akkor a *Ti*-nek megfelelő csomópontot az *i* egésszel jelöljük. Az *i* csomópontból a *j* csomópontba vezet irányított él, ha *Ti* <*S* *Tj*.

**9.7. példa:** A következő *S* ütemezés a *T*1, *T*2, *T*3 három tranzakciót tartalmazza:

*S*: *r*2(*A*); *r*1(*B*); *w*2(*A*); *r*3(*A*); *w*1(*B*); *w*3(*A*); *r*2(*B*); *w*2(*B*);

Ha az *A*-val kapcsolatos műveleteket nézzük meg, akkor több okot találunk, hogy miért igaz a *T*2 <*S* *T*3. Például *r*2(*A*) az *S*-ben *w*3(*A*) előtt áll, és *w*2(*A*) az *r*3(*A*) és a *w*3(*A*) előtt is áll. A három észrevételünk közül bármelyik elegendő, hogy igazoljuk, valóban vezet él a 2-ből 3-ba a 9.9. ábrán szereplő megelőzési gráfban.

**9.9. ábra.** A 9.7. példa S ütemezéséhez tartozó megelőzési gráf

Hasonló módon ha megnézzük a *B*-vel kapcsolatos műveleteket, akkor szintén több okot találunk, hogy miért igaz a *T*1 <*S* *T*2. Például *r*1(*B*) művelet a *w*2(*B*) művelet előtt áll. Tehát az *S* megelőzési gráfjában az 1-ből 2-be szintén vezet él. Tulajdonképpen ezek és csak ezek az élek azok, amelyeket az *S* ütemezésben szereplő műveletek sorrendjéből tudunk ellenőrizni. 

Van egy egyszerű szabály, amivel megmondhatjuk, hogy egy *S* ütemezés konfliktus sorbarendezhető-e:

• Rajzoljuk fel az *S* megelőzési gráfját, és nézzük meg tartalmaz-e kört!

Ha igen, akkor *S* nem konfliktus sorbarendezhető. Ha pedig a gráf körmentes, akkor *S* konfliktus sorbarendezhető, továbbá a csomópontok bármelyik topologikus sorrendje[[1]](#footnote-1) megadja a konfliktus ekvivalens soros sorrendet.

**9.8. példa:** A 9.9. ábra megelőzési gráfja körmentes, így a 9.7. példa *S* ütemezése konfliktus-sorbarendezhető. A csomópontoknak, azaz a tranzakcióknak csak egyetlen sorrendje van, amely konzisztens a gráf éleivel, és ez: (*T*1, *T*2, *T*3). Megjegyezzük, hogy az *S*-et valóban át lehet alakítani olyan ütemezéssé, amelyben a három tranzakció mindegyikének az összes művelete ugyanebben a sorrendben van, és ez a soros ütemezés:

**Miért nem szükséges konfliktus sorbarendezhetőség a sorbarendezhetőséghez?**

Egy példát már láttunk a 9.7. ábrán. Akkor megnéztük, hogy a *T*2 által végrehajtott speciális számítások miatt hogyan vált az ütemezés sorba rendezhetővé. Pedig a 9.7. ábra ütemezése nem konfliktus sorbarendezhető, ugyanis az *A*-t *T*1 írja előbb, a *B*-t pedig a *T*2. Mivel sem az *A* írását, sem a *B* írását nem lehet átrendezni, semmilyen módon nem kerülhet *T*1 összes művelete a *T*2 összes művelete elé, sem fordítva.

Azonban vannak olyan sorba rendezhető, de nem konfliktus sorbarendezhető ütemezések is, amelyek nem függnek a tranzakciók által végrehajtott számításoktól. Például tekintsük a *T*1, *T*2, és *T*3 három tranzakciót, amelyek mindegyike *X* értékét írja. A *T*1 és *T*2 az *Y*-nak is ír értéket, mielőtt az *X*-nek írnának értéket. Az egyik lehetséges ütemezés, amely itt éppen soros is, a következő:

*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 a *T*1 és a *T*2 milyen értéket ír az *X*-be, nincs hatása, ugyanis a *T*3 felülírja az értékeket. Emiatt *S*1 és *S*2 az *X*-nek is és az *Y*-nak is ugyanazt az értéket adja. Mivel az *S*1 soros, és az *S*2-nek bármely adatbázis-állapotra ugyanaz a hatása, mint az *S*1-nek, tehát *S*2 sorba rendezhető. Ugyanakkor mivel nem tudjuk felcserélni *w*1(*Y*)-t *w*2(*Y*)-nal, és nem tudjuk felcserélni *w*1(*X*)-et *w*2(*X*)-szel, így cseréken keresztül nem lehet az *S*2-t valamelyik soros ütemezéssé átalakítani. Tehát *S*2 sorba rendezhető, de nem konfliktus sorbarendezhető.

*S*: *r*1(*B*); *w*1(*B*); *r*2(*A*); *w*2(*A*); *r*2(*B*); *w*2(*B*); *r*3(*A*); *w*3(*A*);

Ahhoz, hogy belássuk, megkaphatjuk az *S*-ből az *S*-t szomszédos elemek cseréjével, az első észrevételünk, hogy az *r*1(*B*)-t konfliktus nélkül az *r*2(*A*) elé hozhatjuk. Ezután három cserével a *w*1(*B*)-t közvetlenül az *r*1(*B*) utánra tudjuk cserélni, ugyanis mindegyik közbeeső művelet az *A*-ra vonatkozik, és a *B*-re nem. Ezután az *r*2(*B*)-t és a *w*2(*B*)-t csak az *A*-ra vonatkozó műveleteken keresztül át tudjuk vinni pontosan a *w*2(*A*) utáni helyzetbe, amivel megkapjuk az *S*-t. 

**9.9. példa:** Tekintsük az alábbi ütemezést:

*S*1: *r*2(*A*); *r*1(*B*); *w*2(*A*); *r*2(*B*); *r*3(*A*); *w*1(*B*); *w*3(*A*); *w*2(*B*);

amely csak abban különbözik az *S*-től, hogy az *r*2(*B*) művelet három hellyel előbb szerepel. Az *A*-ra vonatkozó műveleteket megvizsgálva most is csak a *T*2 <*S*1 *T*3 megelőzési kapcsolathoz jutunk. De, ha a *B*-t vizsgáljuk, akkor nemcsak *T*1 < *S*1 *T*2 teljesül (ugyanis *r*1(*B*) és *w*1(*B*) a *w*2(*B*) előtt szerepel), hanem *T*2 < *S*1 *T*1 is (ugyanis *r*2(*B*) a *w*1(*B*) előtt fordul elő). Emiatt az *S*1 ütemezéshez tartozó megelőzési gráf az, amely a 9.10. ábrán látható.

**9.10. ábra.** A 9.9. példa S1 ütemezéséhez tartozó ciklikus megelőzési gráf,
ez az ütemezés nem konfliktus sorbarendezhető

Ez a gráf nyilvánvalóan tartalmaz kört. Arra következtethetünk, hogy *S*1 nem konfliktus sorbarendezhető, ugyanis intuíció alapján láthatjuk, hogy bármely konfliktus ekviva­lens soros ütemezésben a *T*1-nek *T*2 előtt is és után is kellene állnia, így emiatt nem létezik ilyen ütemezés. 

### 9.2.3. Miért működik a megelőzési gráfon alapuló tesztelés?

Láttuk, hogy a megelőzési gráfban a kör túl sok megszorítást jelent a feltételezett kon­flik­tusekvivalens soros ütemezésre nézve. Azaz, ha létezik *T*1  *T*2    *Tn*  *T*1 *n* darab tranzakcióból álló kör, akkor a feltételezett soros sorrendben *T*1 mű­ve­leteinek meg kell előznie a *T*2-ben szereplő műveleteket, amelyek megelőzik a *T*3-belieket és így tovább egészen *Tn*-ig. De a *Tn* műveletei emiatt a *T*1-beliek mögött vannak, ugyanakkor meg is kellene előzniük a *T*1-belieket a *Tn*  *T*1 él miatt. Ebből következik, hogyha a megelőzési gráf tartalmaz kört, akkor az ütemezés nem konfliktus­-sor­ba­rendezhető.

A másik irányt egy kicsit nehezebb belátnunk. Azt kell megmutatnunk, hogy amikor a megelőzési gráf körmentes, akkor az ütemezés műveletei átrendezhetők szomszédos műveletek szabályos cseréivel úgy, hogy az ütemezés egy soros ütemezéssé váljon. Ha ezt meg tudjuk tenni, akkor bebizonyítottuk, hogy minden körmentes megelőzési gráffal rendelkező ütemezés egyben konfliktus-sorbarendezhető. A bizonyítás az ütemezésben részt vevő tranzakciók száma szerinti indukcióval történik.

**Alapeset:** Ha *n* = 1, vagyis csak egyetlen tranzakcióból áll az ütemezés, akkor az ütemezés már önmagában soros, emiatt biztosan konfliktus-sorbarendezhető.

**Indukció:** Legyen *S*

*T*1, *T*2, …, *Tn*

*n* darab tranzakció műveleteiből álló ütemezés. Tételezzük fel, hogy *S*-nek körmentes megelőzési gráfja van. Ha a véges gráf körmentes, akkor van legalább egy olyan csomópontja, amelybe nem vezet él. Legyen a *Ti* tranzakciónak megfelelő *i* csomópont egy ilyen csomópont. Mivel nem vezet az *i* csomópontba él, nincs az *S*-ben olyan *A* művelet, hogy:

1. Valamelyik *Tj*, a *Ti*-től különböző tranzakcióra vonatkozzon,

2. *Ti* valamely műveletét megelőzi, és

3. Ezzel a művelettel konfliktusban van.

Ugyanis, ha lenne ilyen, akkor a megelőzési gráfban be kellene rajzolnunk egy élt a *j* csomópontból az *i* csomópontba.

Így lehetséges, hogy a *Ti* minden műveletét az *S* legelejére mozgatjuk át, miközben megtartjuk a sorrendjüket. Az ütemezés most a következő alakú:

(*Ti* műveletei) (a többi *n* – 1 tranzakció műveletei)

Most tekintsük az *S* második részét, vagyis a *Ti*-től különböző összes tranzakciónak a műveleteit. Mivel ezek a műveletek egymáshoz viszonyítva ugyanabban a sorrendben vannak, mint ahogyan az *S*-ben voltak, ennek a második résznek a megelőzési gráfja megegyezik az *S* olyan megelőzési gráfjával, amelyből elhagyjuk a *Ti* csomópontot és az ebből a csomópontból kimenő éleket.

Mivel az eredeti megelőzési gráf körmentes volt, és csomópontok, illetve élek törlésével nem válhatott ciklikussá, következik, hogy a második rész megelőzési gráfja is körmentes. Továbbá, mivel a második része *n* – 1 tranzakciót tartalmaz, alkalmazzuk rá az indukciós feltevést. Így tudjuk, hogy a második rész műveletei szomszédos műveletek szabályos cseréivel átrendezhetők soros ütemezéssé. Ily módon magát az *S*-et alakítottuk át olyan soros ütemezéssé, amelyben a *Ti* műveletei állnak legelöl, és a többi tranzakció műveletei ezután következnek valamilyen soros sorrendben. Az indukciót beláttuk, és így következik, hogy minden olyan ütemezés, amelynek körmentes a megelőzési gráfja, egyben konfliktus-sorbarendezhető.

1. Egy körmentes gráf *topologikus sorrendje* a csomópontok bármely olyan rendezése, amelyben minden *a*  *b* élre, az a csomópont megelőzi a *b* csomópontot a topologikus rendezésben. Bármely körmentes gráfnak találhatunk topologikus rendezettséget úgy, hogy többszörösen ismételve töröljük azokat a csomópontokat, amelyeknek nincs megelőzője a maradék csomópontok között. [↑](#footnote-ref-1)