Kompatibilitási mátrixok használata

1. A mátrix alapján dönti el az ütemező, hogy egy ütemezés/zárkérés legális-e, illetve ez alapján várakoztatja a tranzakciókat. Minél több az Igen a mátrixban, annál kevesebb lesz a várakoztatás.

2. A mátrix alapján keletkező várakozásokhoz elkészített várakozási gráf segítségével az ütemező kezeli a holtpontot.

3. A mátrix alapján készíti el az ütemező a megelőzési gráfot egy zárkérés-sorozathoz:

- a megelőzési gráf csúcsai a tranzakciók és akkor van él T_i-ből T_j-be, ha van olyan A adategység, amelyre az ütemezés során Z_k zárat kért és kapott T_i , ezt elengedte, majd
- ezután A-ra legközelebb T_j kért és kapott olyan Z_l zárat, hogy a mátrixban a Z_k sor Z_l oszlopában Nem áll.

Vagyis olyankor lesz él, ha a két zár nem kompatibilis egymással, nem mindegy a két művelet sorrendje.

..... $T_i:Z_k(A)$ .... $T_i:UZ_k(A)$ .... $T_j:Z_l(A)$...

nem kompatibilis

Kompatibilitási mátrixok használata

- **A sorbarendezhetőséget a megelőzési gráf segítségével lehet eldönteni.**

- **Tétel.** ***Egy csak zárkéréseket és zárelengedéseket tartalmazó jogszerű ütemezés sorbarendezhető akkor és csak akkor, ha a kompatibilitási mátrix alapján felrajzolt megelőzési gráf nem tartalmaz irányított kört.***

Bizonyítás:

Ha van irányított kör, akkor ekvivalens soros ütemezésben is ugyanebben a sorrendben következnének a körben szereplő tranzakciók, ami ellentmondás.

A másik irány indukcióval. A topológikus sorrendben első tranzakció lépéseit előre mozgathatjuk, a hatás nem változik, és a maradék tranzakciók indukció miatt ekvivalensek egy soros ütemezéssel. Q.E.D.

A zárakra vonatkozó megelőzési gráf körmentességi feltétel erőssége

Tekintsük az

$$l_1(A);\ r_1(A);\ u_1(A);\ l_2(A);\ r_2(A);\ u_2(A);\ l_1(A);\ w_1(A);\ u_1(A);\ l_2(B);\ r_2(B);\ u_2(B)$$

ütemezést.

Ha megnézzük az írás/olvasás műveleteket ($r_1(A)$; $r_2(A)$; $w_1(A)$; $r_2(B)$), akkor látszik, hogy az ütemezés hatása azonos a T_2T_1 soros ütemezés hatásával, vagyis ez egy sorbarendezhető ütemezés zárak nélkül.

De ha felrajzoljuk a zárakra vonatkozó megelőzési gráfot (és ilyenkor persze nem nézzük, hogy milyen írások/olvasások vannak, hanem a legrosszabb esetre készülünk), akkor

lesz a gráf, és mivel ez irányított kört tartalmaz, akkor ezt elvetnénk, mert nem lesz sorbarendezhető az az ütemezés, amiben már csak a zárak vannak benne.

A sorbarendezhetőség biztosítása tetszőleges zármodellben

Az ütemező egyik lehetősége a sorbarendezhetőség elérésére, hogy folyamatosan figyeli a megelőzési gráfot és ha irányított kör keletkezne, akkor ABORT-ot rendel el.

Másik lehetőség a protokollal való megelőzés. Tetszőleges zármodellben értelmes a 2PL és igaz az alábbi tétel:

Tétel. ***Ha valamilyen zármodellben egy jogszerű ütemezésben minden tranzakció követi a 2PL-t, akkor az ütemezéshez tartozó megelőzési gráf nem tartalmaz irányított kört, azaz az ütemezés sorbarendezhető.***

Bizonyítás: Az előzőekhez hasonlóan.

Megjegyzés: Minél gazdagabb a zármodell, minél több az IGEN a kompatibilitási mátrixban, annál valószínűbb, hogy a megelőzési gráfban nincs irányított kör, minden külön protokoll nélkül. Ez azt jelenti, ilyenkor egyre jobb lesz az ABORT-os módszer (azaz ritkán kell visszagörgetni egy tranzakciót).

Zárak felminősítése

- **L2 *erősebb* L1-nél, ha a kompatibilitási mátrixban L2 sorában /oszlopában minden olyan pozícióban „NEM” áll, amelyben L1 sorában /oszlopában „NEM” áll.**
- **Például az SX zárolási séma esetén X erősebb S-nél (X minden zármódnál erősebb, hiszen X sorában és oszlopában is minden pozíción „NEM” szerepel).**

	S	X
S	IGEN	NEM
X	NEM	NEM

- **Azt mondjuk, hogy a T tranzakció *felminősíti* (upgrade) az L1 zárját az L1-nél erősebb L2 zárra az A adatbáziselemen, ha L2 zárat kér (és kap) A-ra, amelyen már birtokol egy L1 zárat (azaz még nem oldotta fel L1-et). Például: $sl_i(A)$.....$Xl_i(A)$**

Zárak felminősítése

T_1: $sl_1(A); r_1(A); sl_1(B); r_1(B); xl_1(B); w_1(B); u_1(A); u_1(B);$

T_2: $sl_2(A); r_2(A); sl_2(B); r_2(B); u_2(A); u_2(B);$

T_1	T_2
$sl_1(A);\ r_1(A);$	
	$sl_2(A);\ r_2(A);$
	$sl_2(B);\ r_2(B);$
$xl_1(B);$ elutasítva	
	$u_2(A);\ u_2(B);$
$xl_1(B);\ r_1(B);$	
$w_1(B);$	
$u_1(A);\ u_1(B);$	

Felminősítés nélkül

T_1	T_2
$sl_1(A);\ r_1(A);$	
	$sl_2(A);\ r_2(A);$
	$sl_2(B);\ r_2(B);$
$sl_1(B);\ r_1(B);$	
$xl_1(B);$ elutasítva	
	$u_2(A);\ u_2(B);$
$xl_1(B);\ w_1(B);$	
$u_1(A);\ u_1(B);$	

Felminősítéssel

A T_1 tranzakció T_2-vel konkurensen tudja végrehajtani az írás előtti, esetleg hosszadalmas számításait, amely nem lenne lehetséges, ha T_1 kezdetben kizárólagosan zárolta volna B-t.

Új típusú holtpontok felminősítés esetén

T_1	T_2
$sl_1(A)$;	
	$sl_2(A)$;
$xl_1(A)$; elutasítva	
	$xl_2(A)$; elutasítva

Egyik tranzakció végrehajtása sem folytatódhat:

- **vagy mindkettőnek örökösen kell várakoznia,**
- **vagy addig kell várakozniuk, amíg a rendszer fel nem fedezi, hogy holtpont alakult ki, abortálja valamelyik tranzakciót, és a másiknak engedélyezi `A`-ra a kizáró-lagos zárat.**
- **Megoldás: módosítási zárak**

Módosítási zárak

- **Az $ul_i(X)$ módosítási zár a T_i tranzakciónak csak X olvasására ad jogot, X írására nem. Később azonban csak a módosítási zárat lehet felminősíteni írásira, az olvasási zárat nem (azt csak módosításira).**
- **A módosítási zár tehát nem csak a holtpontproblémát oldja meg, hanem a kiéheztetés problémáját is.**

	S	X	U
S	igen	nem	igen
X	nem	nem	nem
U	nem	nem	nem

NEM SZIMMETRIKUS!

Az U módosítási zár úgy néz ki, mintha osztott zár lenne, amikor kérjük, és úgy néz ki, mintha kizárólagos zár lenne, amikor már megvan.

Módosítási zárak

T_1: $ul_1(A); r_1(A); xl_1(A); w_1(A); u_1(A);$

T_2: $ul_2(A); r_2(A); xl_2(A); w_2(A); u_2(A);$

T_1	T_2
$ul_1(A);\ r_1(A);$	
	$ul_2(A);$ elutasítva
$xl_1(A);\ w_1(A);\ u_1(A);$	
	$ul_2(A);\ r_2(A);$
	$xl_2(A);\ w_2(A);\ u_2(A);$

Nincs holtpont!

Növelési zárak

- **Az `INC(A,c)` művelet:**
  `READ(A,t); t := t+c; WRITE(A,t);`
  műveletek atomi végrehajtása, ahol c egy konstans.

- **Tetszőleges sorrendben kiszámíthatók.**
- **A növelés nem cserélhető fel sem az olvasással, sem az írással.**
  - **Ha azelőtt vagy azután olvassuk be A-t, hogy valaki növelte, különböző értékeket kapunk,**
  - **és ha azelőtt vagy azután növeljük A-t, hogy más tranzakció új értéket írt be A-ba, akkor is különböző értékei lesznek A-nak az adatbázisban.**

- **Az $inc_i(X)$ művelet:**
  a T_i tranzakció megnöveli az X adatbáziselemet valamely konstanssal.
  (Annak, hogy pontosan mennyi ez a konstans, nincs jelentősége.)

- **A műveletnek megfelelő *növelési zárat* (increment lock) $il_i(X)$-szel jelöljük.**

Növelési zárak

	S	X	I
S	igen	nem	nem
X	nem	nem	nem
I	nem	nem	igen

a) Egy konzisztens tranzakció csak akkor végezheti el x-en a növelési műveletet, ha egyidejűleg növelési zárat tart fenn rajta. A növelési zár viszont nem teszi lehetővé sem az olvasási, sem az írási műveleteket. $il_i(X) \ldots inc_i(X)$

b) Az $inc_i(X)$ művelet konfliktusban áll $r_j(X)$-szel és $w_j(X)$-szel is $j \neq i$-re, de nem áll konfliktusban $inc_j(X)$-szel.

c) Egy jogszerű ütemezésben bármennyi tranzakció bármikor fenntarthat x-en növelési zárat. Ha viszont egy tranzakció növelési zárat tart fenn x-en, akkor egyidejűleg semelyik más tranzakció sem tarthat fenn sem osztott, sem kizárólagos zárat x-en.

Növelési zárak

T_1: $sl_1(A); r_1(A); il_1(B); inc_1(B); u_1(A); u_1(B);$

T_2: $sl_2(A); r_2(A); il_2(B); inc_2(B); u_2(A); u_2(B);$

T_1	T_2
$sl_1(A);\ r_1(A);$	
	$sl_2(A);\ r_2(A);$
	$il_2(B);\ inc_2(B);$
$il_1(B);\ inc_1(B);$	
	$u_2(A);\ u_2(B);$
$u_1(A);\ u_1(B);$	

Az ütemezőnek egyik kérést sem kell késleltetnie!

Zárak nélkül: S: $r_1(A); r_2(A); inc_2(B); inc_1(B);$

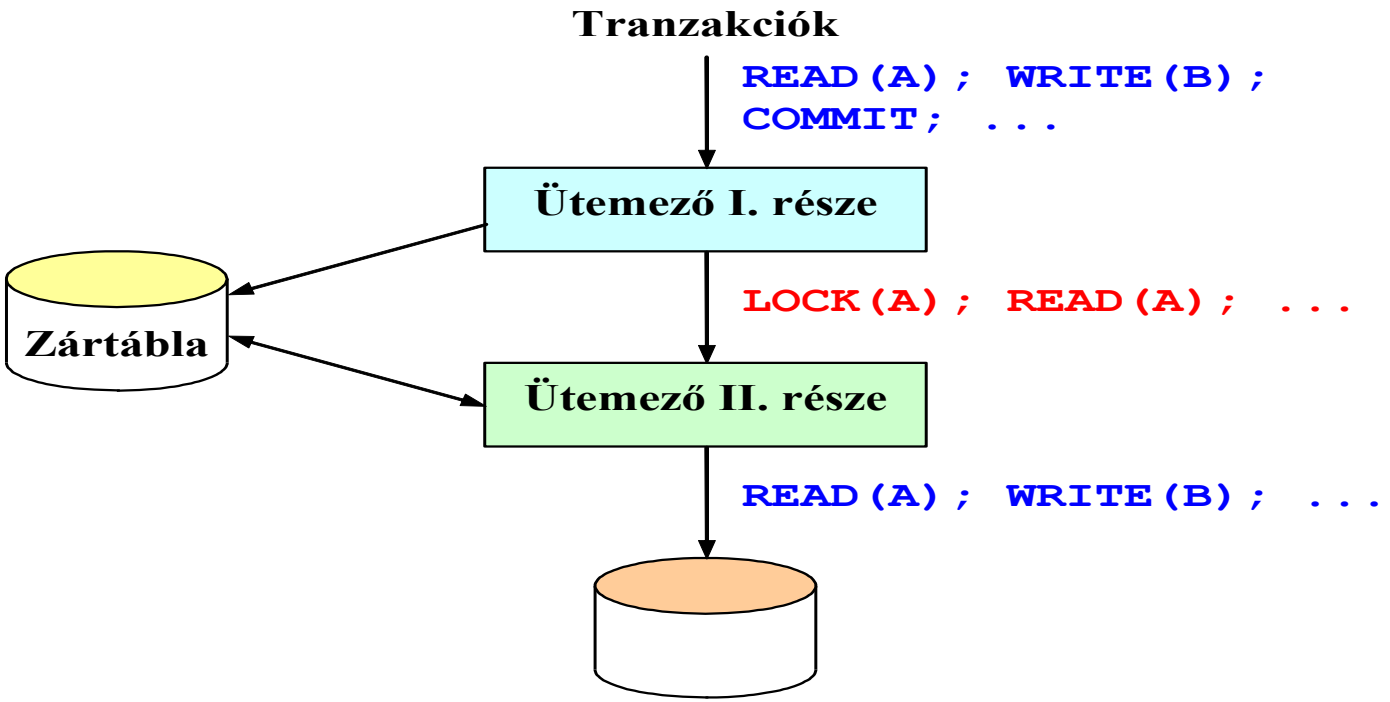
Az utolsó művelet nincs konfliktusban az előzőekkel, így előre hozható a második helyre, így a T_1,T_2 soros ütemezést kapjuk:

S1: $r_1(A); inc_1(B); r_2(A); inc_2(B);$

A zárolási ütemező felépítése

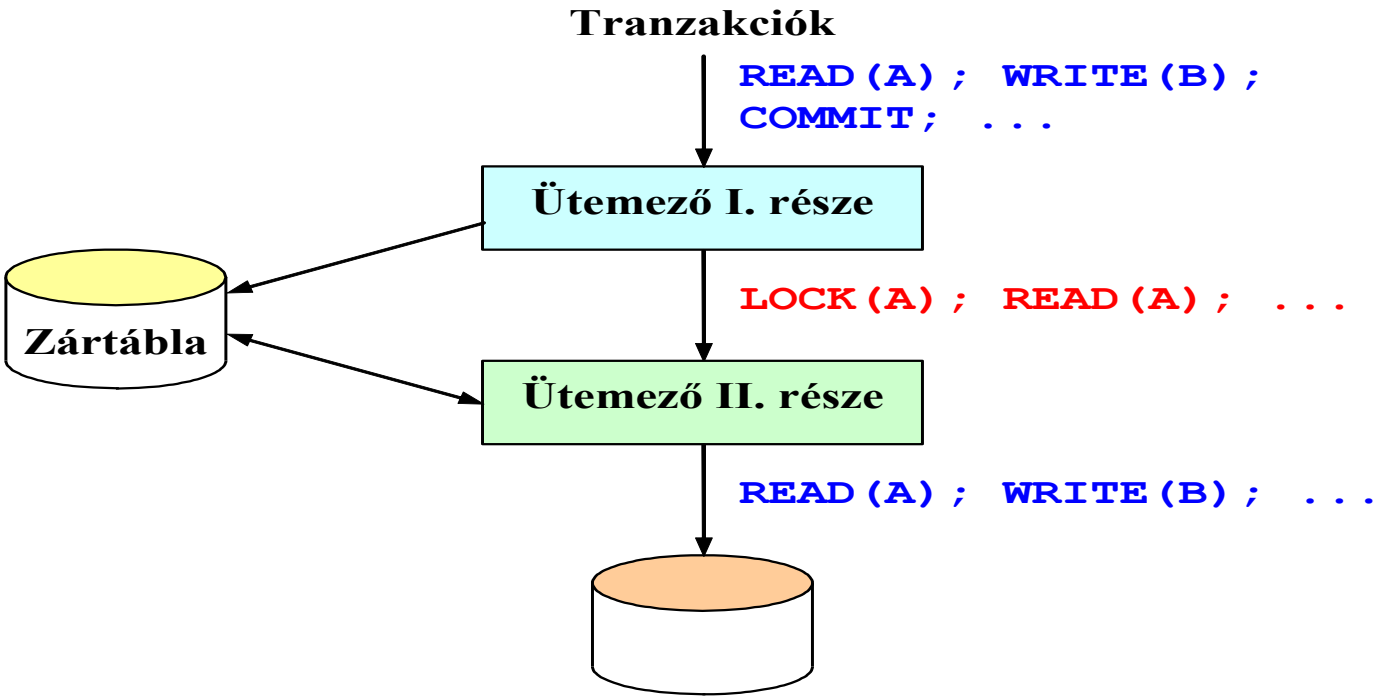
1. Maguk a tranzakciók nem kérnek zárakat, vagy figyelmen kívül hagyjuk, hogy ezt teszik. Az ütemező szúrja be a zárolási műveleteket az adatokhoz hozzáférő olvasási, írási, illetve egyéb műveletek sorába.
2. Nem a tranzakciók, hanem az ütemező oldja fel a zárakat, mégpedig akkor, amikor a tranzakciókezelő a tranzakció véglegesítésére vagy abortálására készül.

Zárolási műveleteket beszúró ütemező



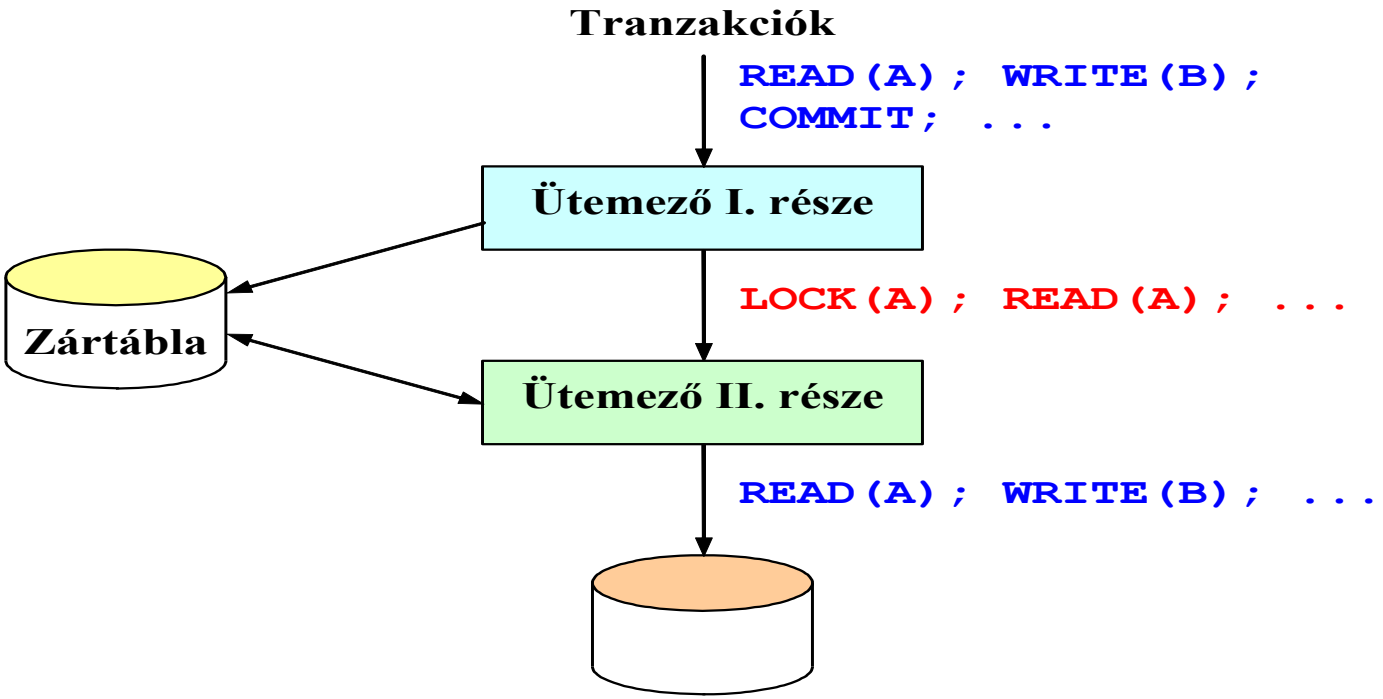
1. Az I. rész fogadja a tranzakciók által generált kérések sorát, és minden adatbázis-hozzáférési művelet elé beszúrja a megfelelő zárolási műveletet. Az adatbázis-hozzáférési és zárolási műveleteket ezután átküldi a II. részhez (a COMMIT és ABORT műveleteket nem).

Zárolási műveleteket beszúró ütemező



2. A II. rész fogadja az I. részen keresztül érkező zárolási és adatbázis-hozzáférési műveletek sorozatát. Eldönti, hogy a T tranzakció késleltetett-e (mivel zárolásra vár). Ha igen, akkor magát a műveletet késlelteti, azaz hozzáadja azoknak a műveleteknek a listájához, amelyeket a T tranzakciónak még végre kell hajtania. Ha T nem késleltetett, vagyis az összes előzőleg kért zár már engedélyezve van, akkor megnézi, hogy milyen műveletet kell végrehajtania.

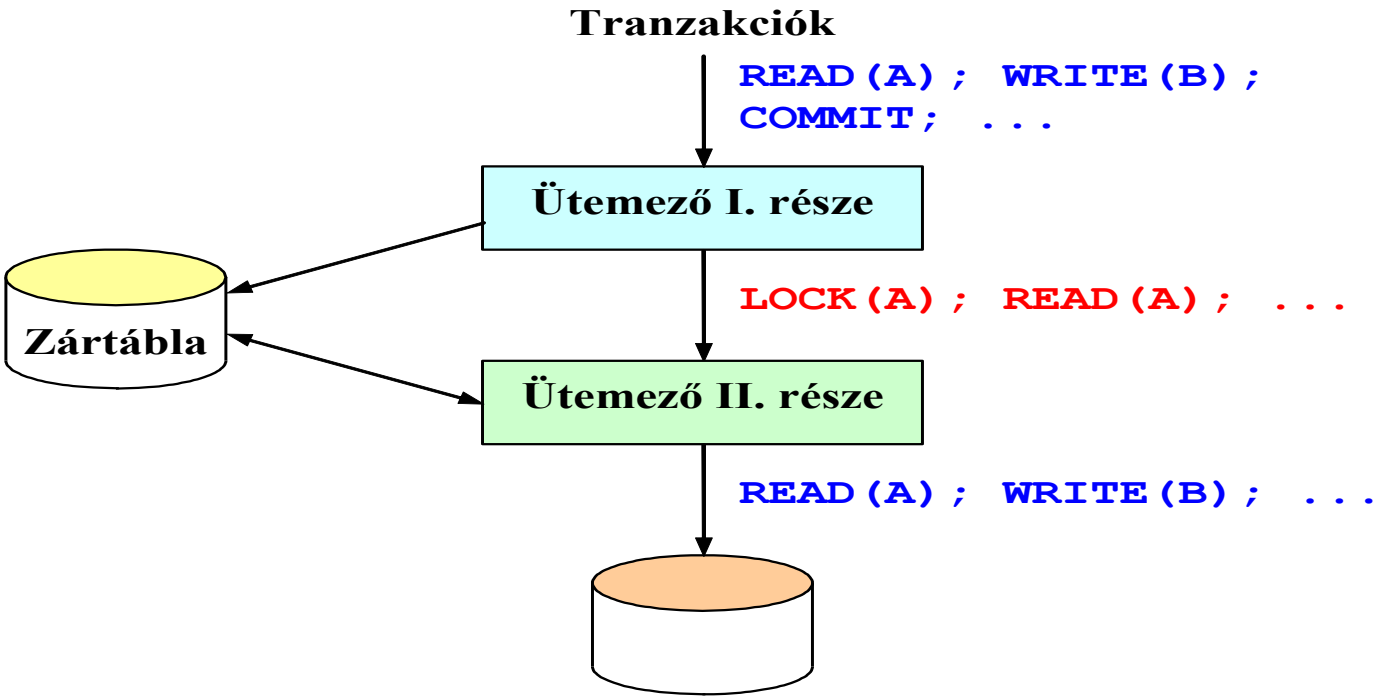
Zárolási műveleteket beszúró ütemező



a) Ha a művelet adatbázis-hozzáférés, akkor továbbítja az adatbázishoz, és végrehajtja.

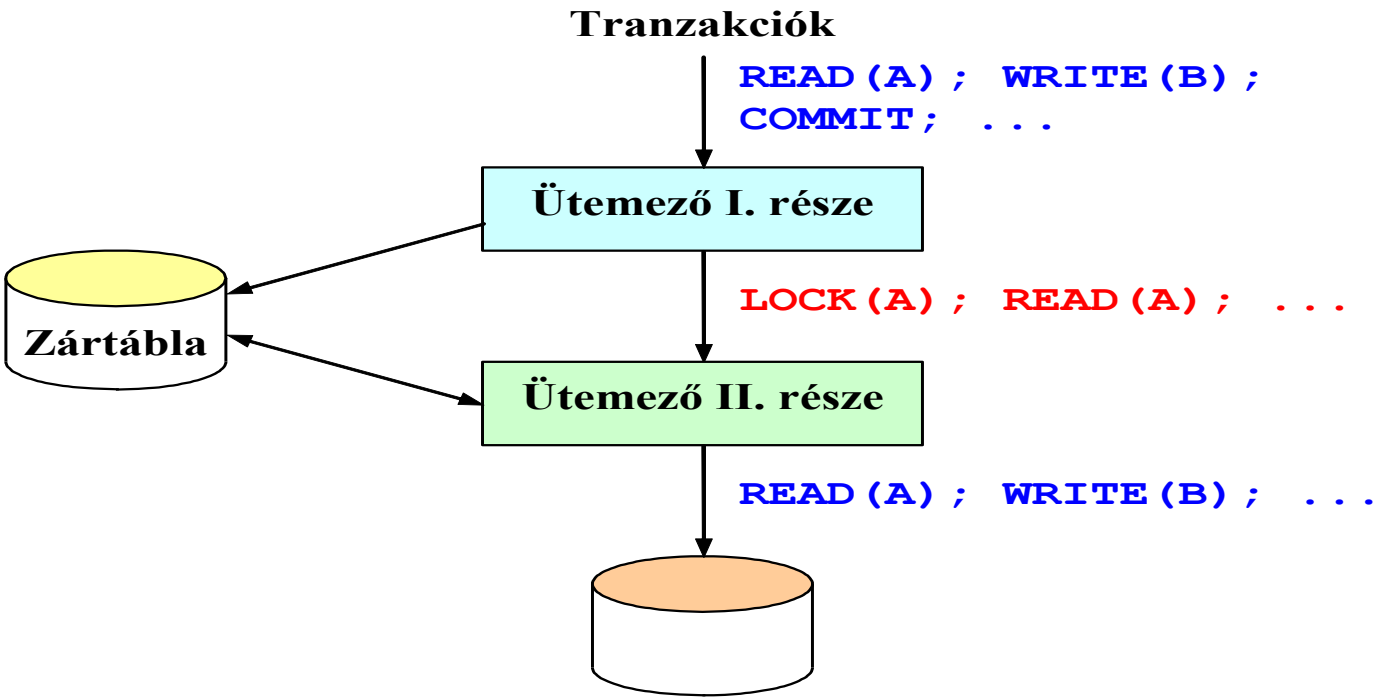
b) Ha zárolási művelet érkezik, akkor megvizsgálja a zártáblát, hogy a zár engedélyezhető-e. Ha igen, akkor úgy módosítja a zártáblát, hogy az az éppen engedélyezett zárat is tartalmazza. Ha nem, akkor egy olyan bejegyzést készít a zártáblában, amely jelzi a zárolási kérést. Az ütemező II. része ezután késlelteti a T tranzakció további műveleteit mindaddig, amíg nem tudja engedélyezni a zárat.

Zárolási műveleteket beszúró ütemező



3. Amikor a T tranzakciót véglegesítjük vagy abortáljuk, akkor a tranzakciókezelő COMMIT, illetve ABORT műveletek küldésével értesíti az I. részt, hogy oldja fel az összes T által fenntartott zárat. Ha bármelyik tranzakció várakozik ezen zárfeloldások valamelyikére, akkor az I. rész értesíti a II. részt.

Zárolási műveleteket beszúró ütemező



4. **Amikor a II. rész értesül, hogy egy X adatbáziselemen felszabadult egy zár, akkor eldönti, hogy melyik az a tranzakció, vagy melyek azok a tranzakciók, amelyek megkapják a zárat X-re. A tranzakciók, amelyek megkapták a zárat, a késleltetett műveleteik közül annyit végrehajtanak, amennyit csak végre tudnak hajtani mindaddig, amíg vagy befejeződnek, vagy egy másik olyan zárolási kéréshez érkeznek el, amely nem engedélyezhető.**

Zárolási műveleteket beszúró ütemező

- **Ha csak egymódú zárak vannak, akkor az ütemező I. részének a feladata egyszerű. Ha bármilyen műveletet lát az A adatbáziselemen, és még nem szúrt be zárolási kérést A-ra az adott tranzakcióhoz, akkor beszúrja a kérést. Amikor véglegesítjük vagy abortáljuk a tranzakciót, az I. rész törölheti ezt a tranzakciót, miután feloldotta a zárakat, így az I. részhez igényelt memória nem nő korlátlanul.**

Zárolási műveleteket beszúró ütemező

- **Amikor többmódú zárak vannak, az ütemezőnek szüksége lehet arra (például felminősítésnél), hogy azonnal értesüljön, milyen későbbi műveletek fognak előfordulni ugyanazon az adatbáziselemen.**

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

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

Felminősítés módosítási zárral:

T_1	T_2
$sl_1(A);\ r_1(A);$	
	$sl_2(A);\ r_2(A);$
	$sl_2(B);\ r_2(B);$
$ul_1(B);\ r_1(B);$	
$xl_1(B);$ elutasítva	
	$u_2(A);\ u_2(B);$
$xl_1(B);\ w_1(B);$	
$u_1(A);\ u_1(B);$	

Az ütemező I. része a következő műveletsorozatot adja ki:

$sl_1(A); r_1(A);$

$sl_2(A); r_2(A); sl_2(B); r_2(B);$

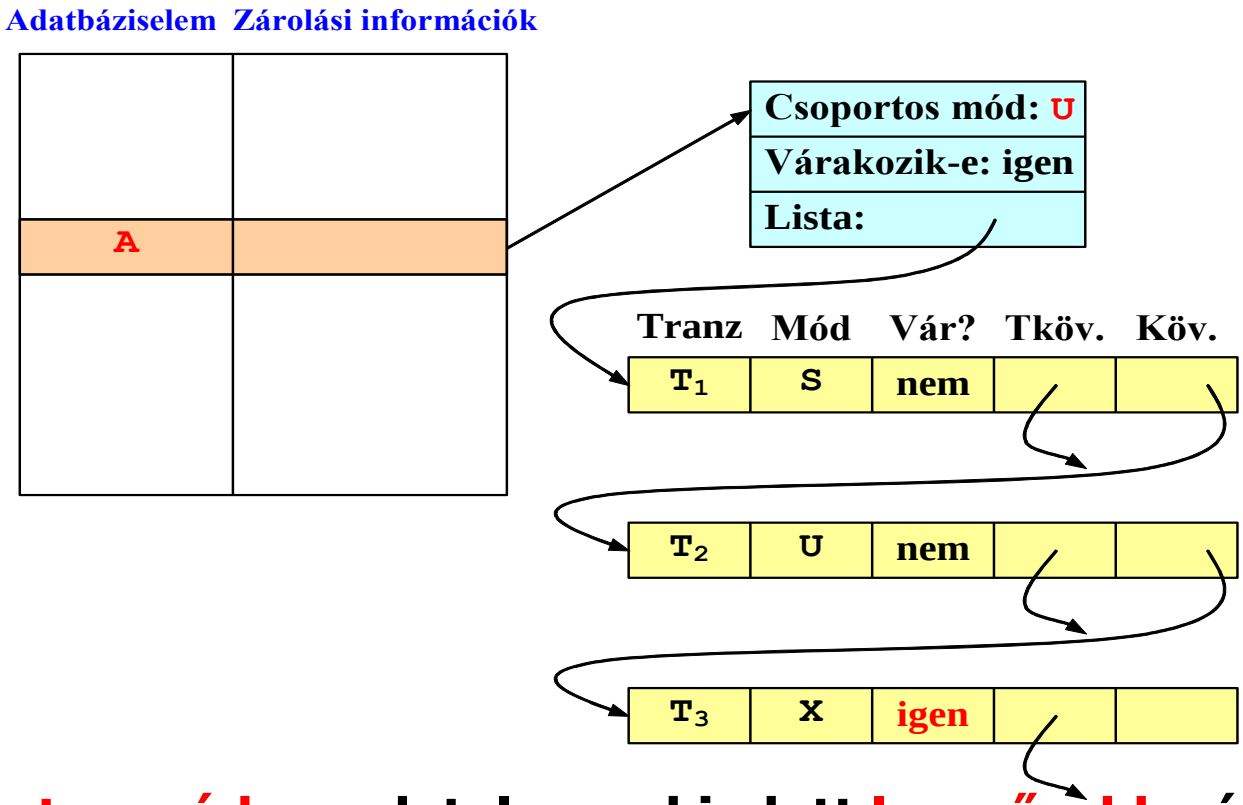
$ul_1(B);\ r_1(B)$

$xl_1(B); w_1(B)$

A II. rész viszont nem teljesítheti az $xl_1(B)$-t, várakoztatja T_1-et.

Végül T_2 végrehajtja a véglegesítést, és az I. rész feloldja a zárakat A-n és B-n. Ugyanekkor felfedezi, hogy T_1 várakozik B zárolására. Értesíti a II. részt, amely az $xl_1(B)$ zárolást most már végrehajthatónak találja. Beviszi ezt a zárat a zártáblába, és folytatja T_1 tárolt műveleteinek a végrehajtását mindaddig, ameddig tudja. Esetünkben T_1 befejeződik.

A zártábla



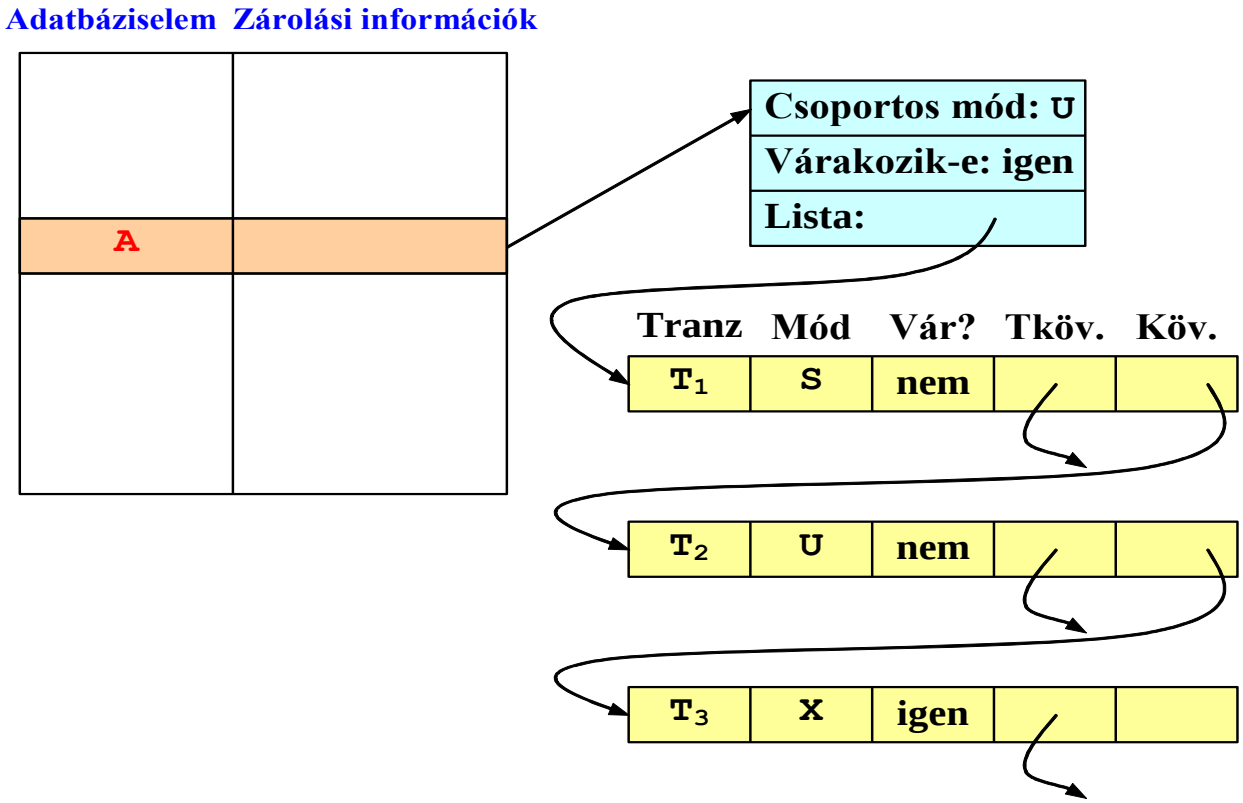
Csoportos mód az adatelemre kiadott legerősebb zár:

a) **S azt jelenti, hogy csak osztott zárak vannak;**

b) **U azt jelenti, hogy egy módosítási zár van, és lehet még egy vagy több osztott zár is;**

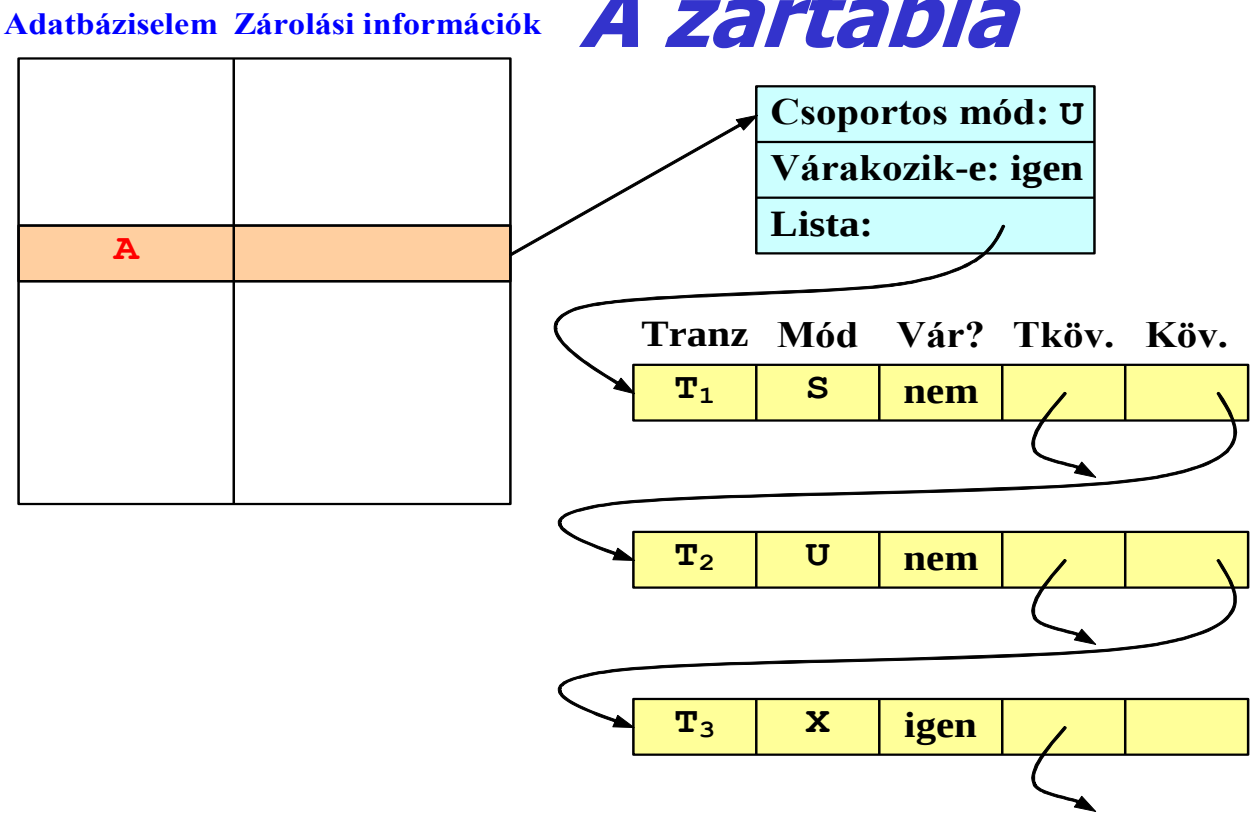
c) **X azt jelenti, hogy csak egy kizárólagos zár van, és semmilyen más zár nincs.**

A zártábla



A *várakozási bit* (waiting bit) azt adja meg, hogy van-e legalább egy tranzakció, amely az A zárolására várakozik.

A zártábla



Az összes olyan tranzakciót leíró lista, amelyek vagy jelenleg zárolják A-t, vagy A zárolására várakoznak.

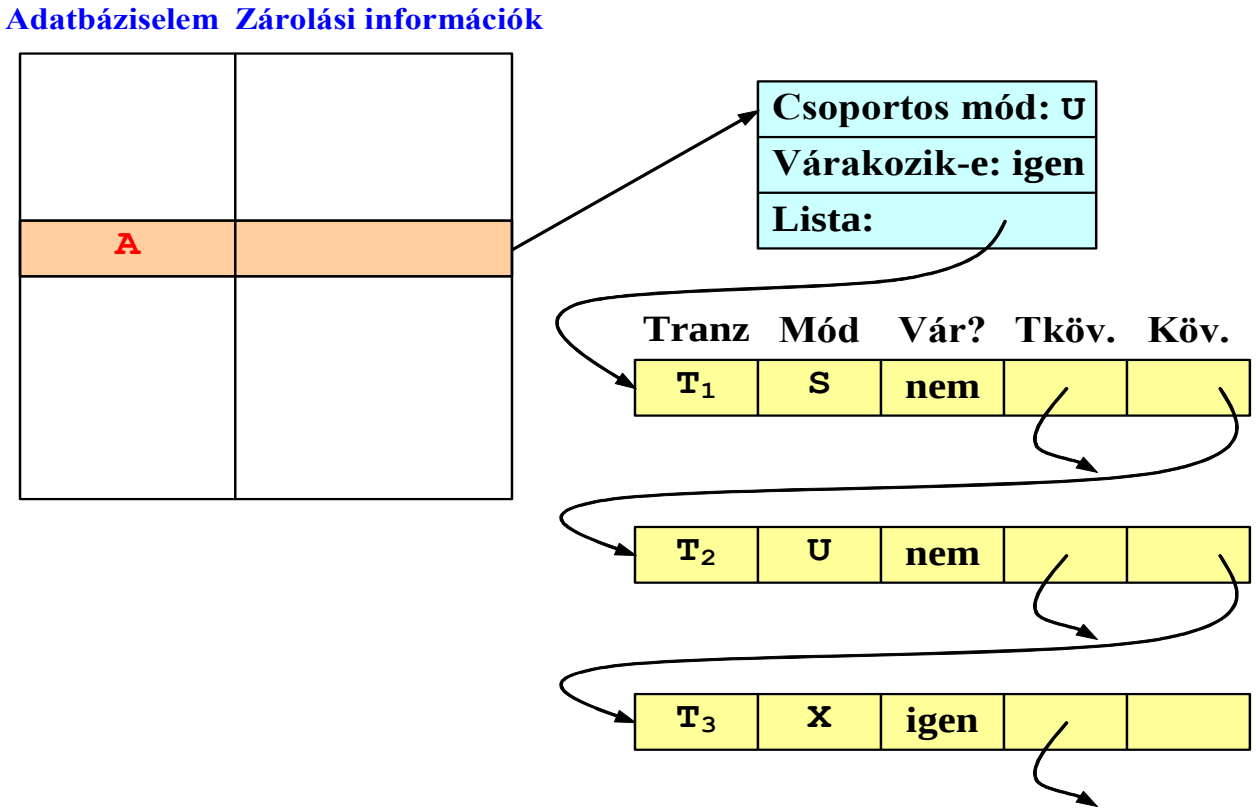
a) a zárolást fenntartó vagy a zárolásra váró **tranzakció neve**;

b) ennek a zárnak a **módja**;

c) a tranzakció **fenntartja-e a zárat**, vagy **várakozik-e a zárra**;

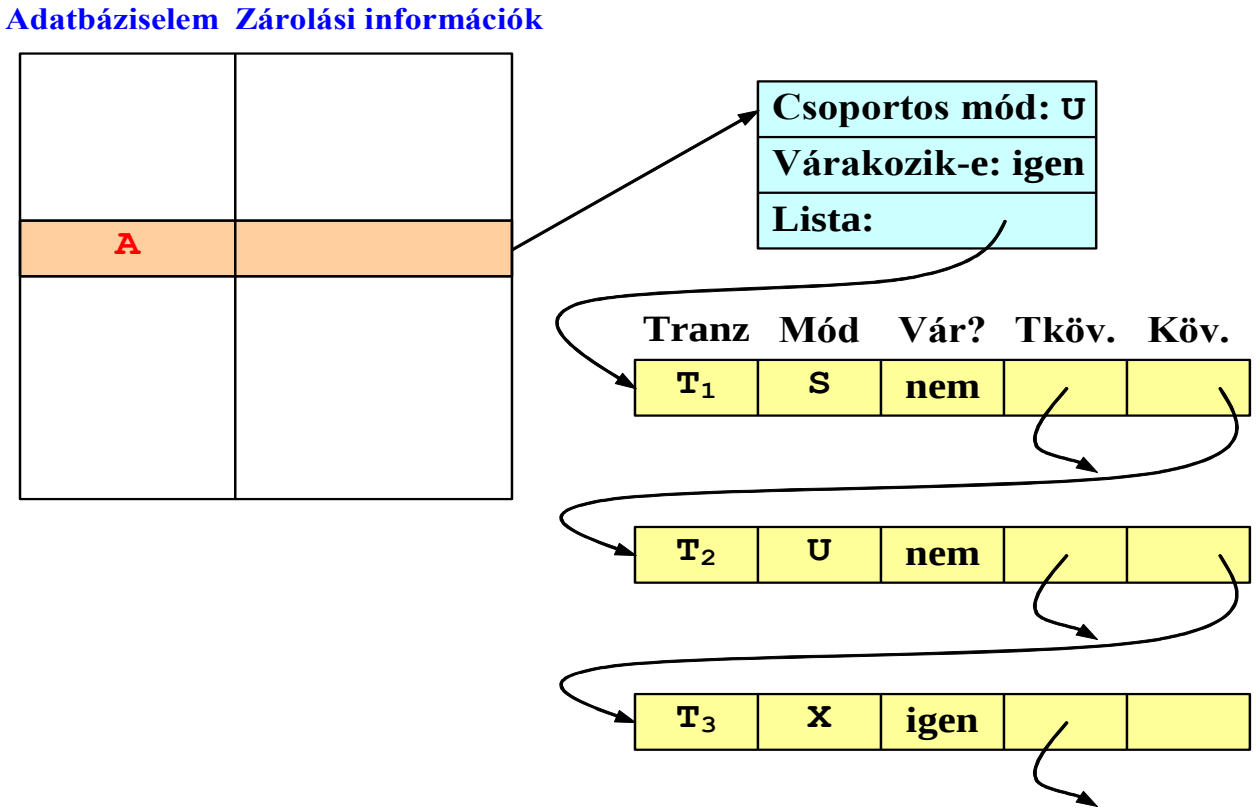
d) az adott tranzakció következő bejegyzése **Tköv**.

A zárolási kérések kezelése



Ha a T tranzakció zárat kér A-ra. Ha nincs A-ra bejegyzés a zártáblában, akkor biztos, hogy zárak sincsenek A-n, így létrehozhatjuk a bejegyzést, és engedélyezhetjük a kérést. Ha a zártáblában létezik bejegyzés A-ra, akkor megkeressük a csoportos módot, és ez alapján várakoztatunk, beírjuk a várakozási listába, vagy megengedjük a zárat (például ha T_2 X-et kér A-ra.)

A zárfeloldások kezelése



Ha T tranzakció feloldja az A-n lévő zárakat. Ekkor T bejegyzését A-ra a listában töröljük, és ha kell a csoportos módot is megváltoztatjuk. Ha maradnak várakozó tranzakciók, akkor engedélyeznünk kell egy vagy több zárat a kért zárak listájáról.

A zárfeloldások kezelése

Több különböző megközelítés lehetséges, mindegyiknek megvan a saját előnye:

1. Első beérkezett első kiszolgálása (first-come-first-served): Azt a zárolási kérést engedélyezzük, amelyik a legrégebb óta várakozik. Ez a stratégia azt biztosítja, hogy ne legyen kiéheztetés, vagyis a tranzakció ne várjon örökké egy zárra.

2. Elsőbbségadás az osztott záraknak (priority to shared locks): Először az összes várakozó osztott zárat engedélyezzük. Ezután egy módosítási zárolást engedélyezünk, ha várakozik ilyen. A kizárólagos zárolást csak akkor engedélyezzük, ha semmilyen más igény nem várakozik. Ez a stratégia csak akkor engedi a kiéheztetést, ha a tranzakció U vagy X zárolásra vár.

3. Elsőbbségadás a felminősítésnek (priority to upgrading): Ha van olyan U zárral rendelkező tranzakció, amely X zárrá való felminősítésre vár, akkor ezt engedélyezzük előbb. Máskülönben a fent említett stratégiák valamelyikét követjük.

Adatbáziselemekből álló hierarchiák kezelése

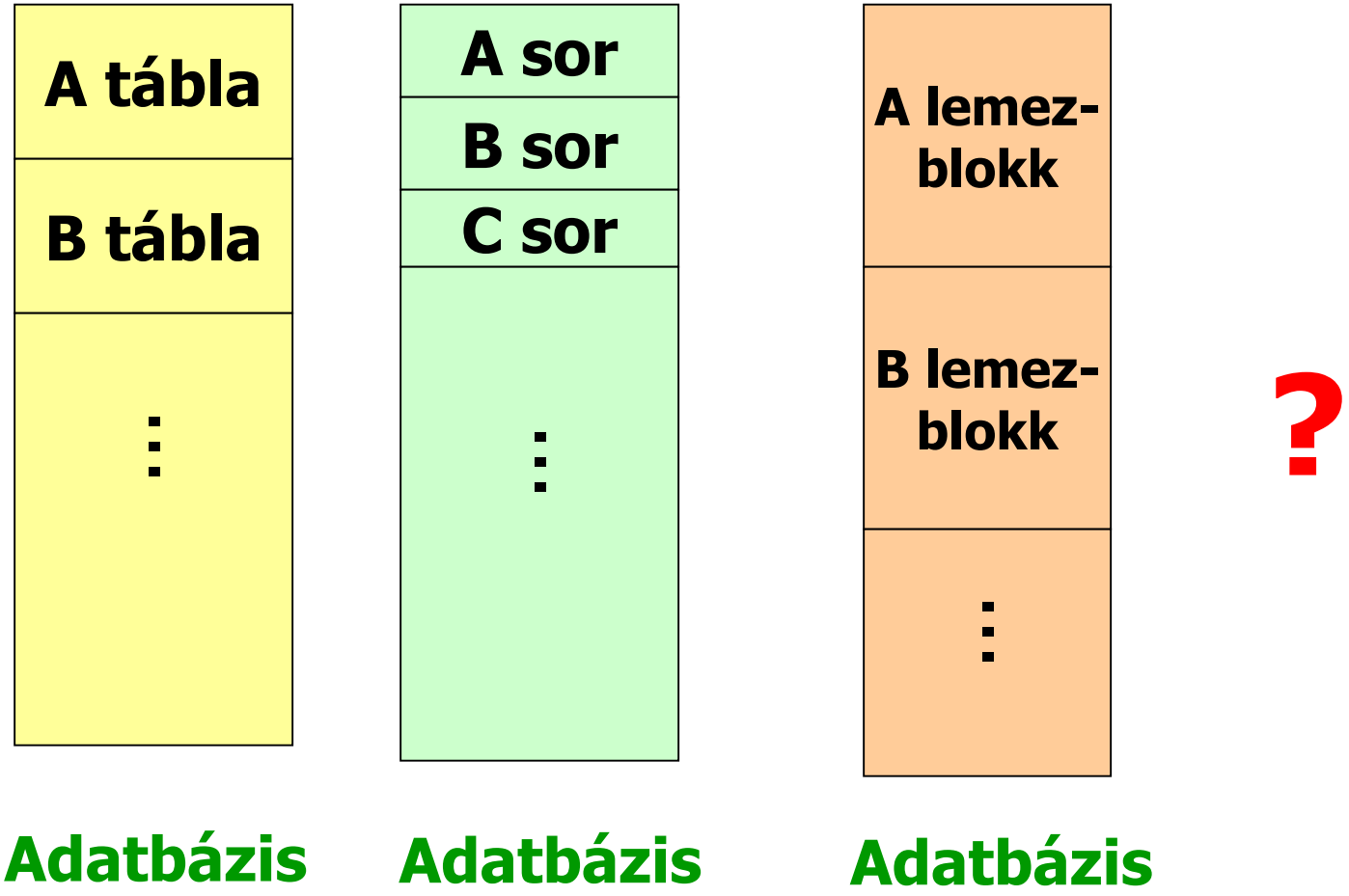
Kétféle fastruktúrával fogunk foglalkozni:

1. Az első fajta fastruktúra, amelyet figyelembe veszünk, a zárolható elemek (zárolási egységek) hierarchiája. Megvizsgáljuk, hogyan engedélyezünk zárolást mind a nagy elemekre, mint például a relációkra, mind a kisebb elemekre, mint például a reláció néhány sorát tartalmazó blokkokra vagy egyedi sorokra.

2. A másik lényeges hierarchiafajtát képezik a konkurenciavezérlési rendszerekben azok az adatok, amelyek önmagukban faszervezésűek. Ilyenek például a B-fa-indexek. A B-fák csomópontjait adatbáziselemeknek tekinthetjük, így viszont az eddig tanult zárolási sémákat szegényesen használhatjuk, emiatt egy új megközelítésre van szükségünk.

Többszörös szemcsézettségű zárak

Milyen objektumokat zároljunk?



Többszörös szemcsézettségű zárak

- **Az alapkérdés, hogy kicsi vagy nagy objektumokat zároljunk?**

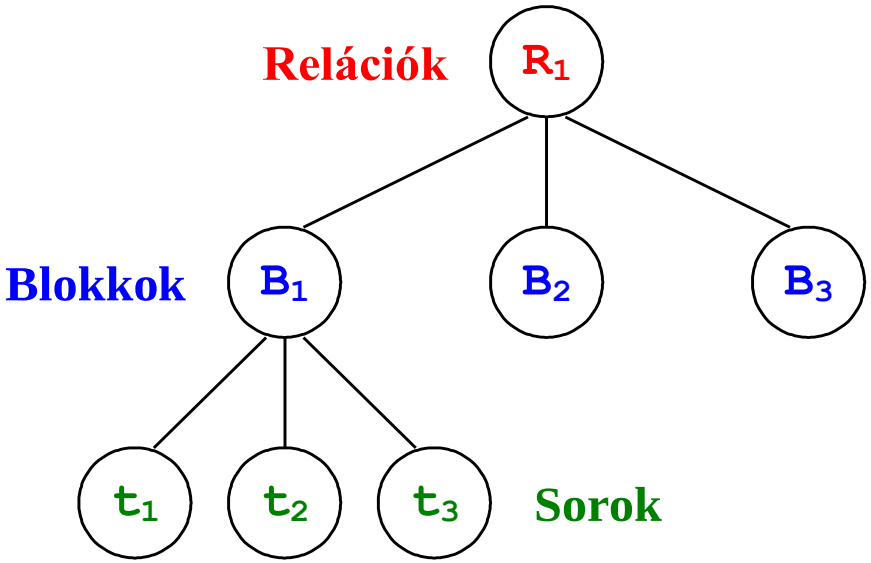
- **Ha nagy objektumokat (például dokumentumokat, táblákat) zárolunk, akkor**
  - **kevesebb zárra lesz szükség, de**
  - **csökken a konkurens működés lehetősége.**
- **Ha kicsi objektumokat, például számlákat, sorokat vagy mezőket) zárolunk, akkor**
  - **több zárra lesz szükség, de**
  - **nő a konkurens működés lehetősége.**

Megoldás: Kicsi ÉS nagy objektumokat is zárolhassunk!

Figyelmeztető zárak

Az adatbáziselemek több (például:három) szintjét különböztetjük meg:

1. a relációk a legnagyobb zárolható elemek;
2. minden reláció egy vagy több blokkból vagy lapból épül fel, amelyekben a soraik vannak;
3. minden blokk egy vagy több sort tartalmaz.



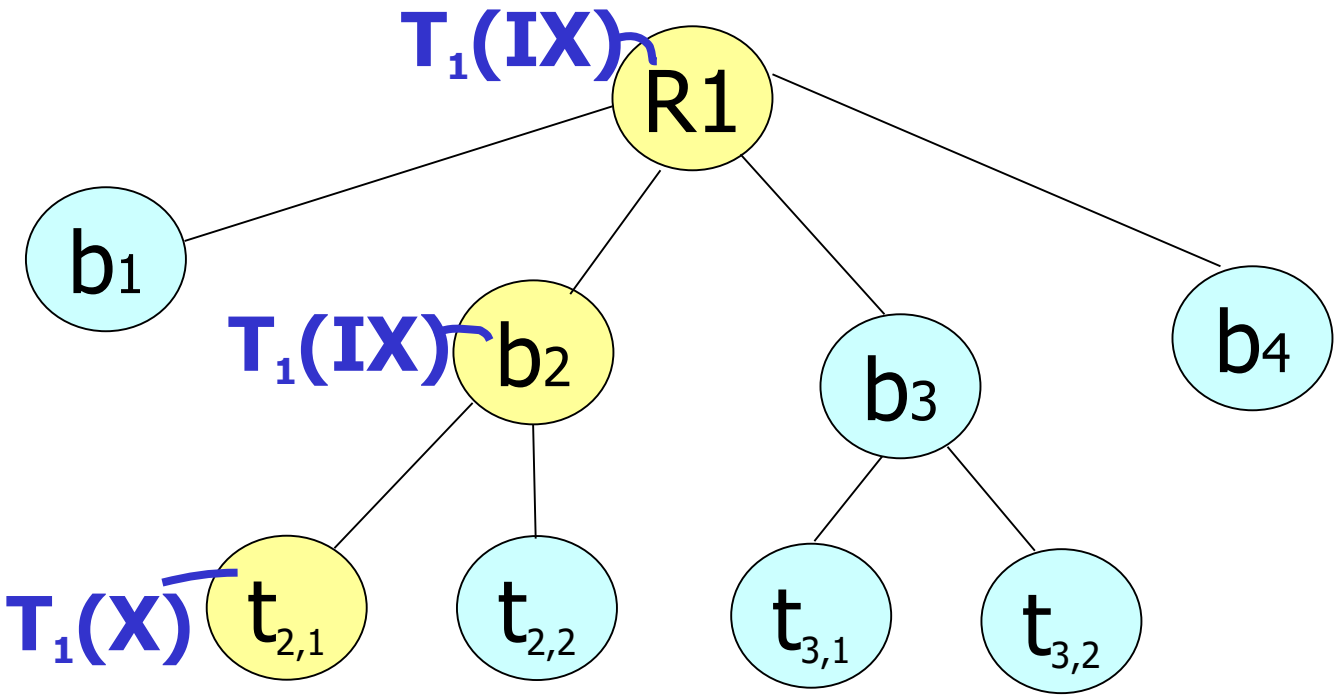
A *figyelmeztető protokoll* zárjai (warning protocol):

1. a közönséges zárak: S és X (osztott és kizárólagos),
2. figyelmeztető zárak: IS, IX (I=intention)

- Például IS azt jelenti, hogy szándékunkban áll osztott zárat kapni egy részelemen.

Figyelmeztető zárak

- **A kért zárnak megfelelő figyelmeztető zárakat kérünk az útvonal mentén a gyökérből kiindulva az adatelemig.**
- **Addig nem megyünk lejjebb, amíg a figyelmeztető zárat meg nem kapjuk.**
- **Így a konfliktusos helyzetek alsóbb szintekre kerülnek a fában.**

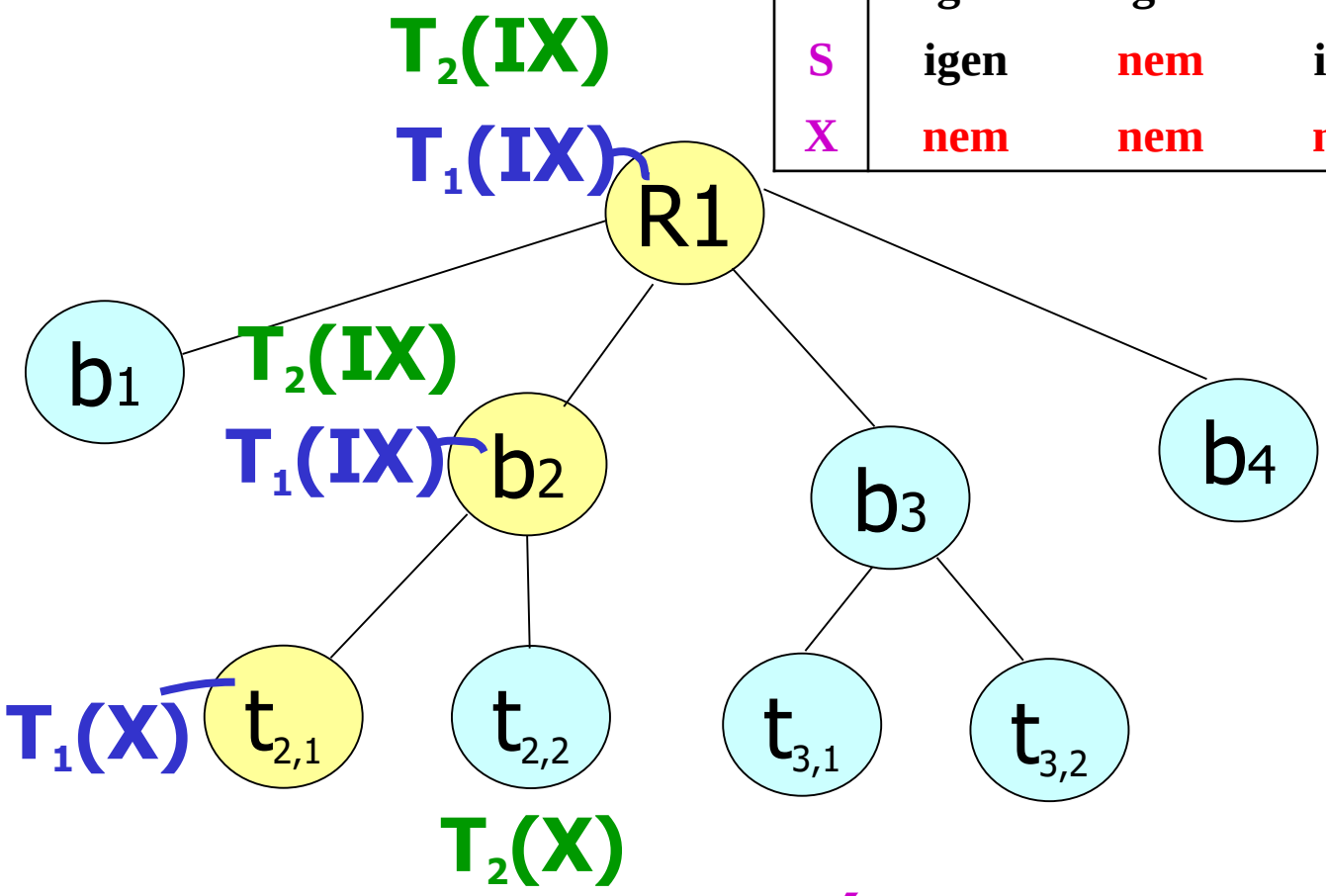


- **Megkaphatja-e T_2 az X zárat a $t_{2,2}$ sorra?**

Figyelmeztető zárak

Kompatibilitási mátrix:

	IS	IX	S	X
IS	igen	igen	igen	nem
IX	igen	igen	nem	nem
S	igen	nem	igen	nem
X	nem	nem	nem	nem

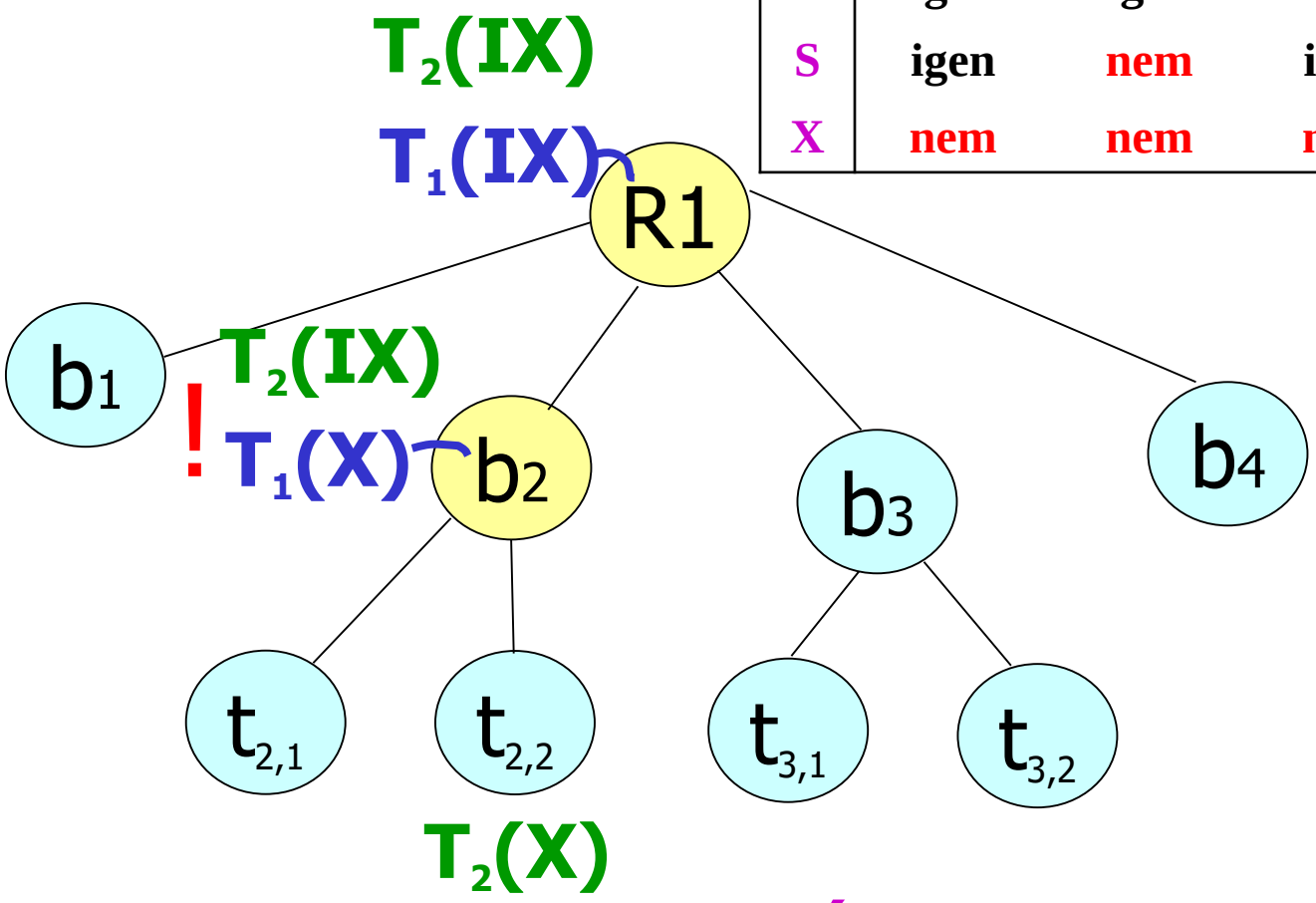


- Megkaphatja-e T_2 az X zárat a $t_{2,2}$ sorra? IGEN

Figyelmeztető zárak

Kompatibilitási mátrix:

	IS	IX	S	X
IS	igen	igen	igen	nem
IX	igen	igen	nem	nem
S	igen	nem	igen	nem
X	nem	nem	nem	nem

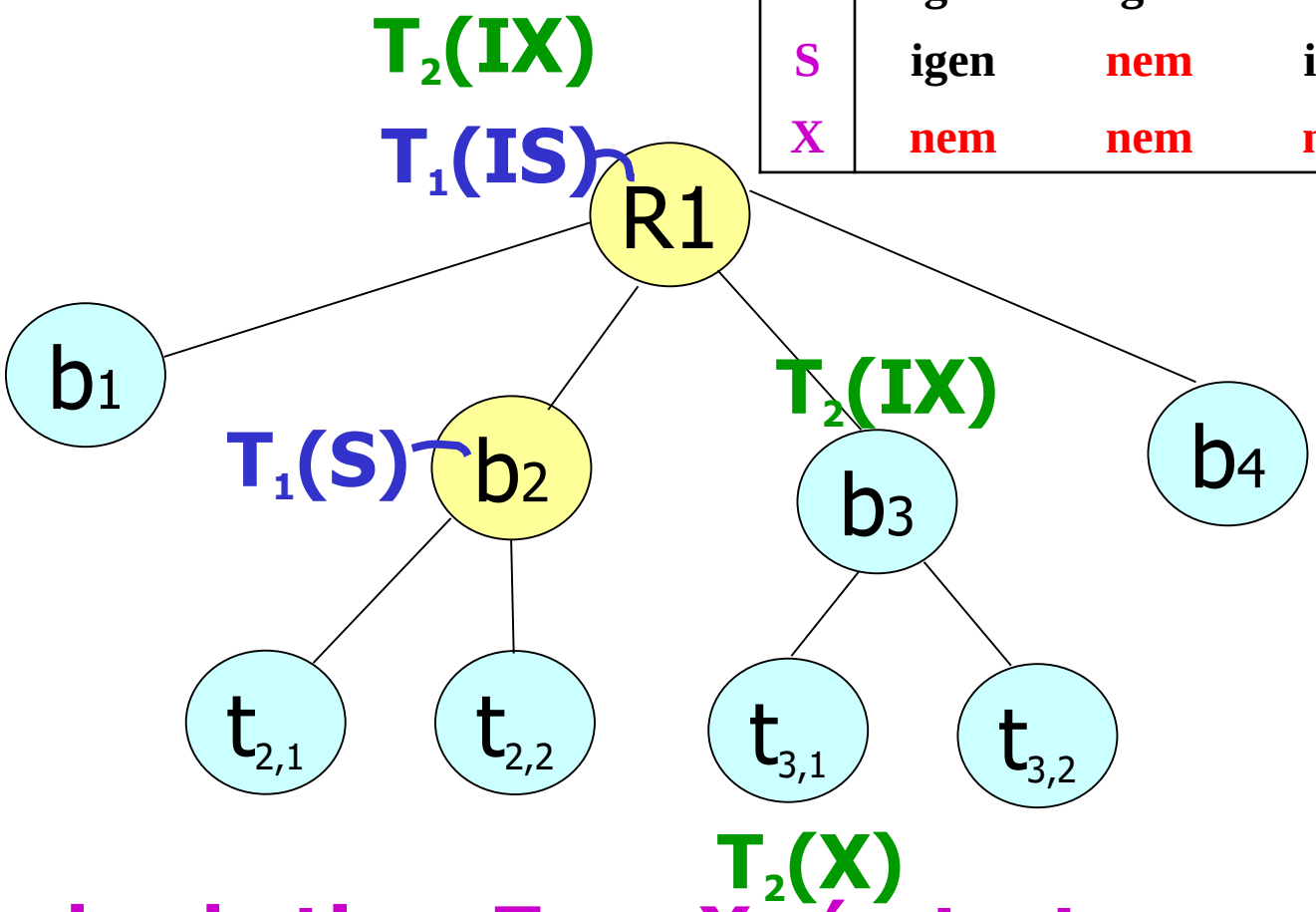


- Megkaphatja-e T_2 az X zárat a $t_{2,2}$ sorra? **NEM**

Figyelmeztető zárak

Kompatibilitási mátrix:

	IS	IX	S	X
IS	igen	igen	igen	nem
IX	igen	igen	nem	nem
S	igen	nem	igen	nem
X	nem	nem	nem	nem



- Megkaphatja-e T_2 az X zárat a $t_{3,1}$ sorra? IGEN

Figyelmeztető zárak

SOR: Ha ilyen zár van már kiadva

	IS	IX	S	X
IS	igen	igen	igen	nem
IX	igen	igen	nem	nem
S	igen	nem	igen	nem
X	nem	nem	nem	nem

Oszlop: Megkaphatjuk-e ezt a típusú zárat?

- Ha IS zárat kérünk egy N csomópontban, az N egy leszármazottját szándékozzuk olvasni. Ez csak abban az esetben okozhat problémát, ha egy másik tranzakció már jogosulttá vált arra, hogy az N által reprezentált teljes adatbáziselemet felülírja (X). Ha más tranzakció azt tervezi, hogy N-nek csak egy részelemét írja (ezért az N csomóponton egy IX zárat helyezett el), akkor lehetőségünk van arra, hogy engedélyezzük az IS zárat N-en, és a konfliktust alsóbb szinten oldhatjuk meg, ha az írási és olvasási szándék valóban egy közös elemre vonatkozik.

- Ha az N csomópont egy részelemét szándékozzuk írni (IX), akkor meg kell akadályoznunk az N által képviselt teljes elem olvasását vagy írását (S vagy X). Azonban más tranzakció, amely egy részelemet olvas vagy ír, a potenciális konfliktusokat az adott szinten kezeli le, így az IX nincs konfliktusban egy másik IX-szel vagy IS-sel N-en.

Figyelmeztető zárak

SOR: Ha ilyen zár van már kiadva

	IS	IX	S	X
IS	igen	igen	igen	nem
IX	igen	igen	nem	nem
S	igen	nem	igen	nem
X	nem	nem	nem	nem

Oszlop: Megkaphatjuk-e ezt a típusú zárat?

- Az N csomópontnak megfeleltetett elem olvasása (S) nincs konfliktusban sem egy másik olvasási zárral N-en, sem egy olvasási zárral N egy részelemén, amelyet N-en egy IS reprezentál. Azonban egy X vagy egy IX azt jelenti, hogy más tranzakció írni fogja legalább egy részét az N által reprezentált elemnek. Ezért nem tudjuk engedélyezni N teljes olvasását.

- Nem tudjuk megengedni az N csomópont írását sem (X), ha más tranzakciónak már joga van arra, hogy olvassa vagy írja N-et (S,X), vagy arra, hogy megszerezze ezt a jogot N egy részelemére (IS,IX).

Összefoglalás

- Kompatibilitási mátrix alapján definiált zártípusok, sorbarendezhetőség feltétele
- Felminősítés, módosítási zár
- Inkrementális (növekményes) zár
- Zárolási ütemező felépítése, 2PL és konzisztencia tranzakciók generálása, jogszerű ütemezés biztosítása, zártábla szerkezete
- Tartalmazási viszonyban álló adatelemek zárolása, figyelmeztető (szándék) zárak kompatibilitási mátrixa