Konzisztencia és többszörözés

10. kurzus

Konzisztencia és többszörözés

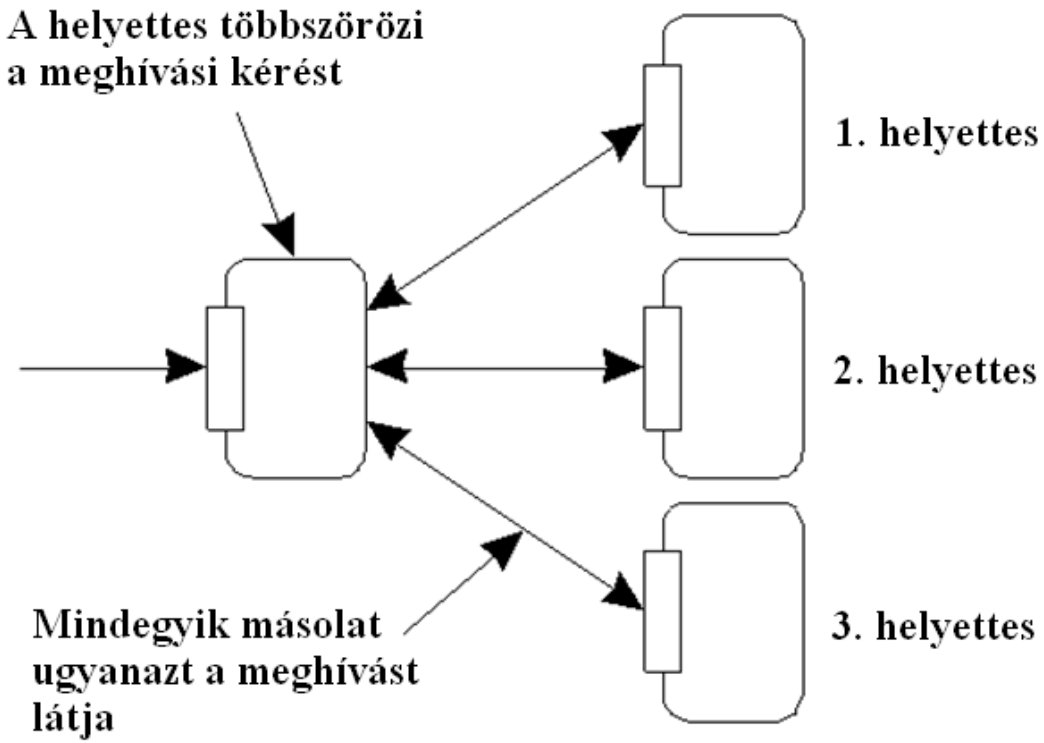
- Megbízhatóság növekedése
- Teljesítmény növekedése

- Többszörözés ára – konzisztencia

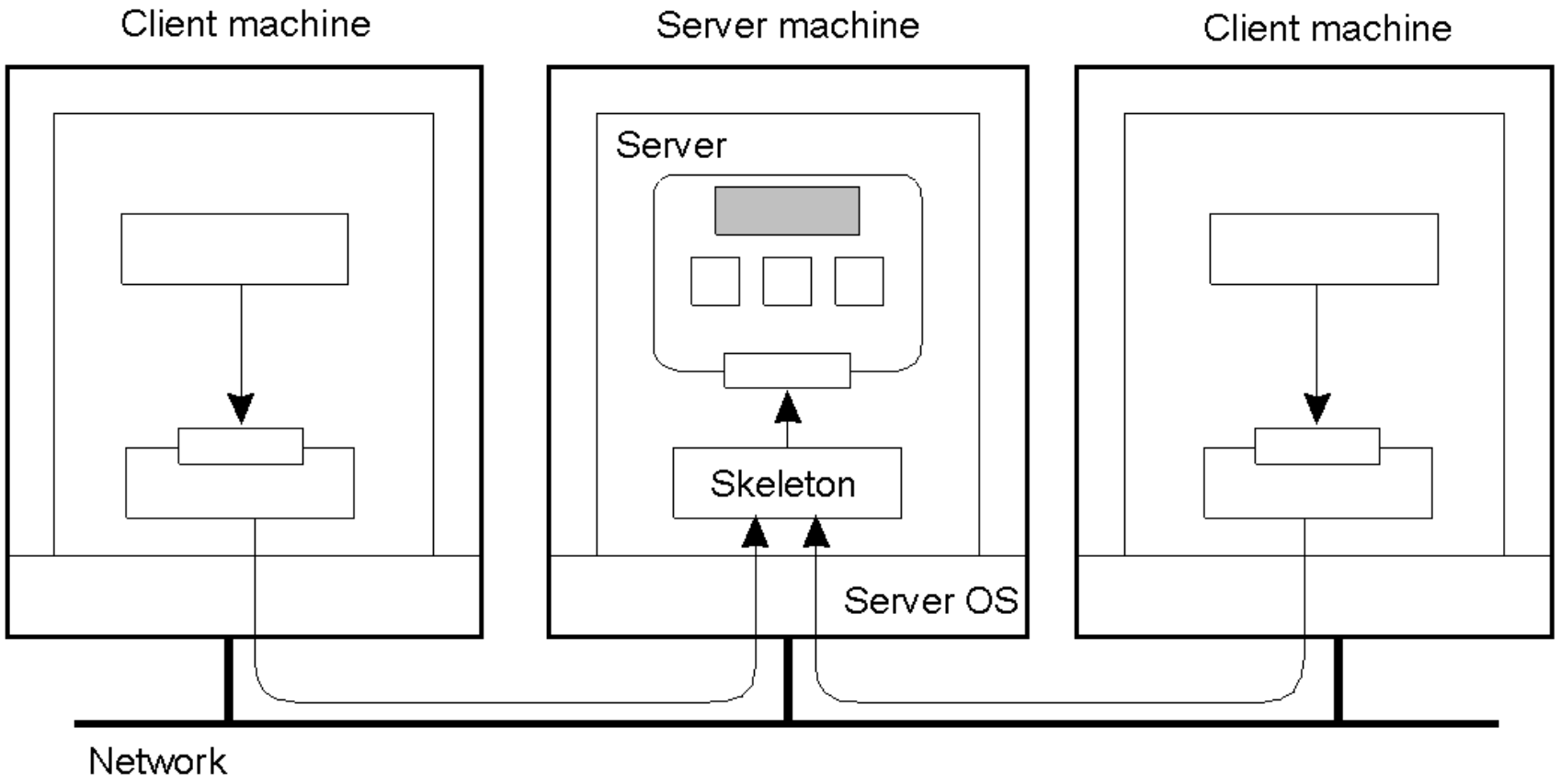
- Átméretezési technika
  - frissítések gyakorisága
  - Konzisztencia

- Gyorsítótár (cache)

Többszörözhetőségi átlátszóság megvalósítása kliens-oldalon

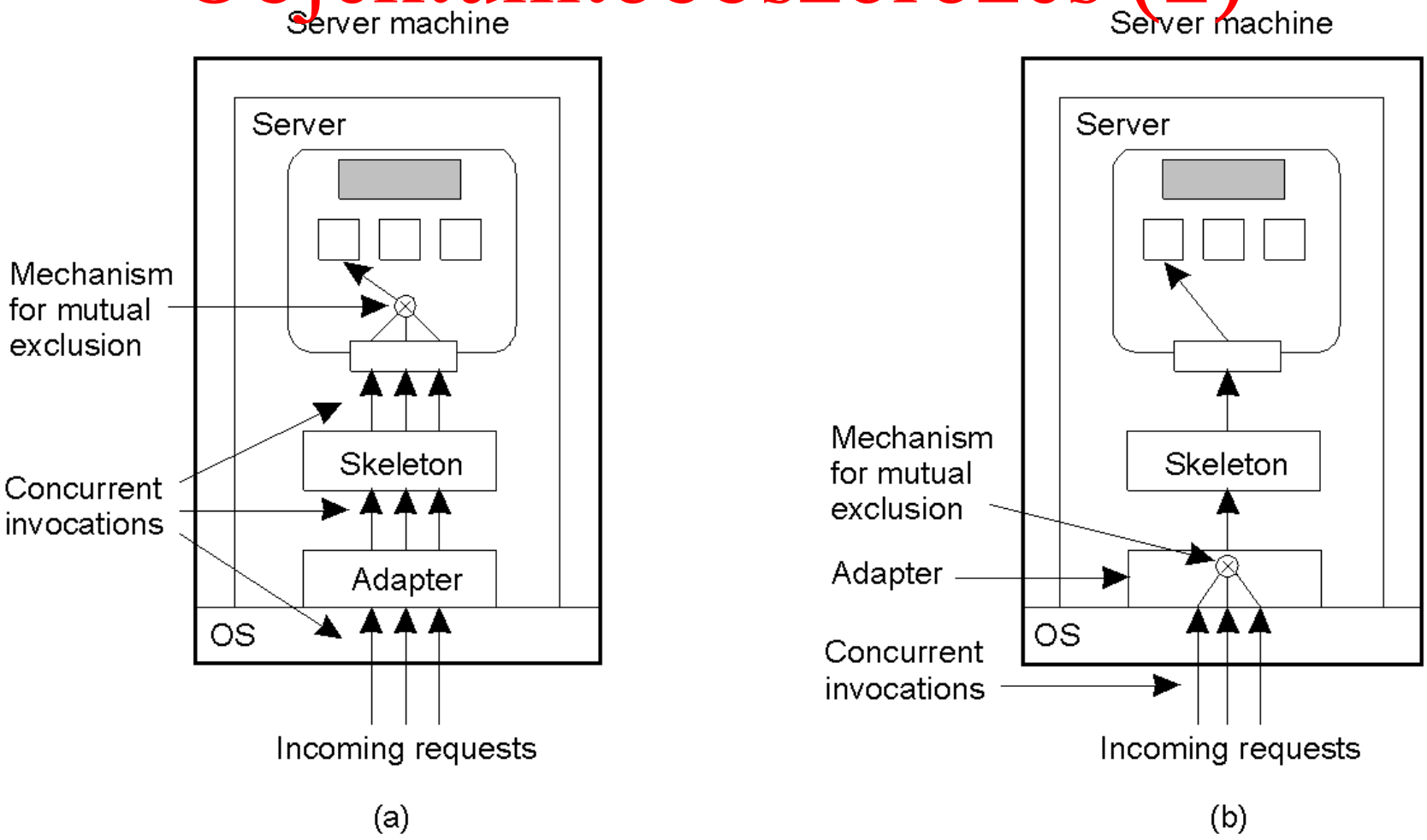


Objektumtöbbszörözés (1)



- Két különböző kliens által közösen használt elosztott távoli objektum szerkezete.

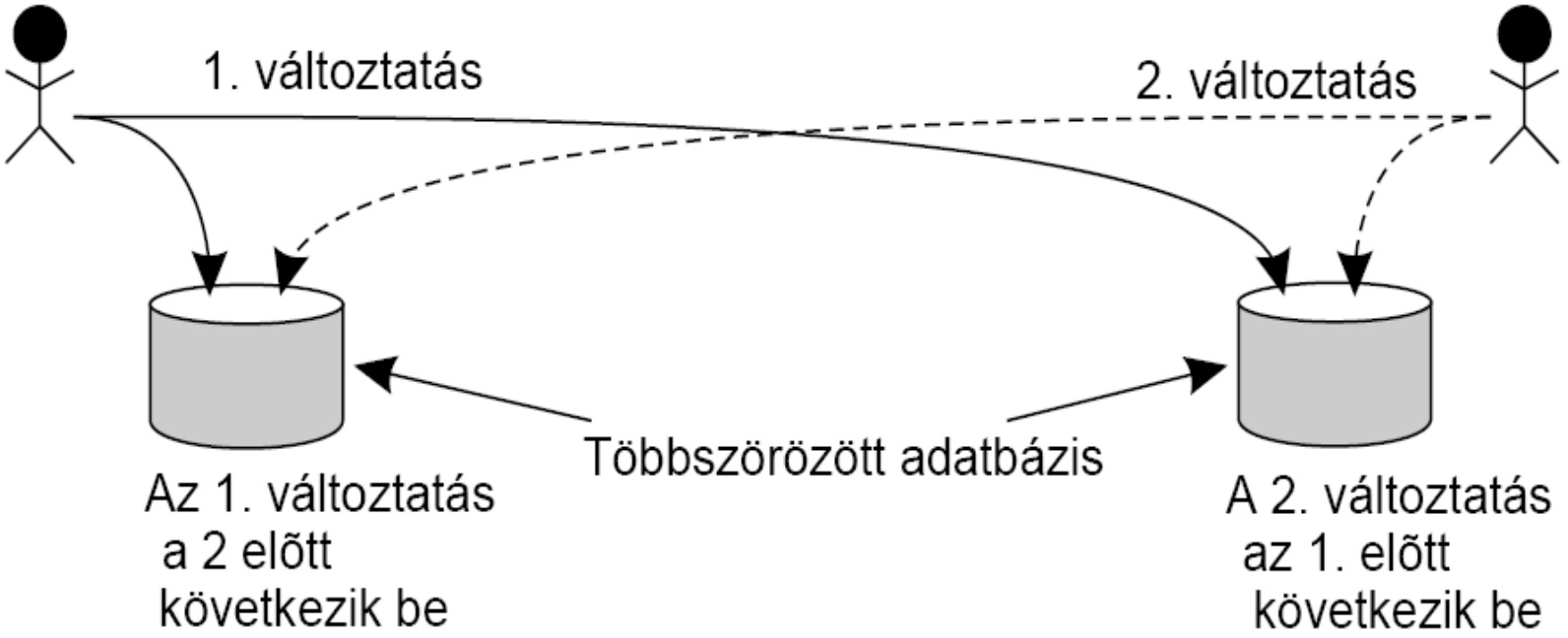
Objektumtöbbszörözés (2)



a) A konkurens hívások kezelésére képes távoli objektum.

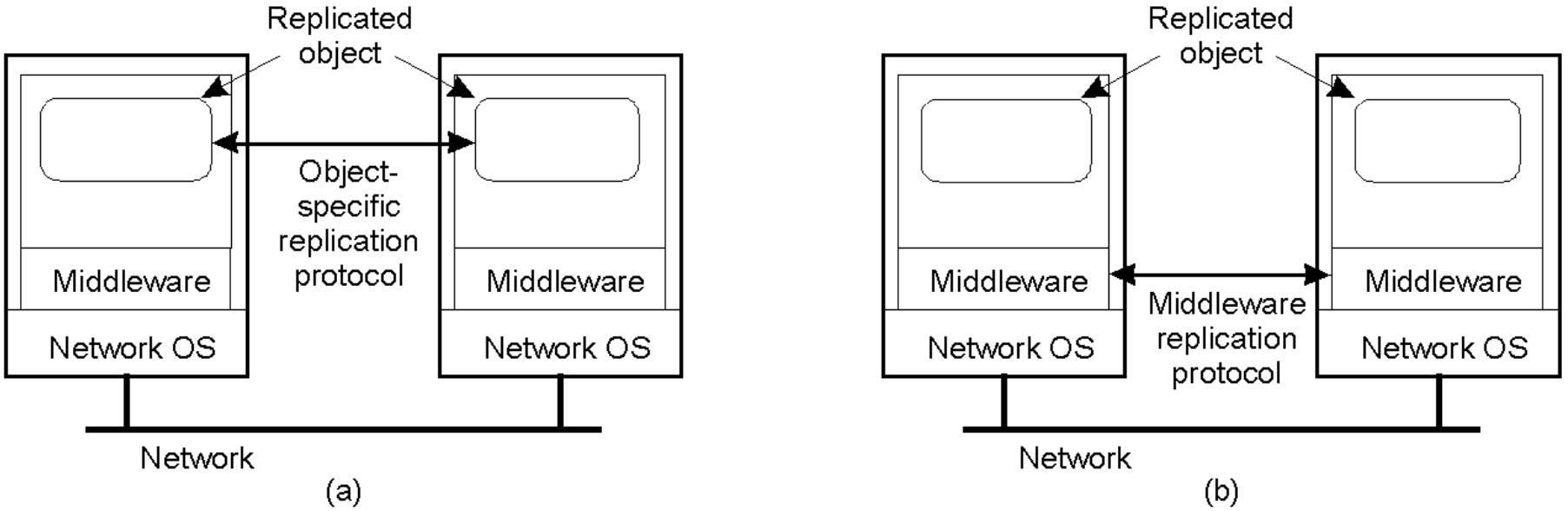
b) A konkurens hívások kezelésének felelősségét az objektumadapterre átruházó távoli objektum.

Példa – pontosan sorbarendezett csoportcímzés



A többszörözött adatbázis frissítése
és az inkonzisztens állapot
kialakulása.

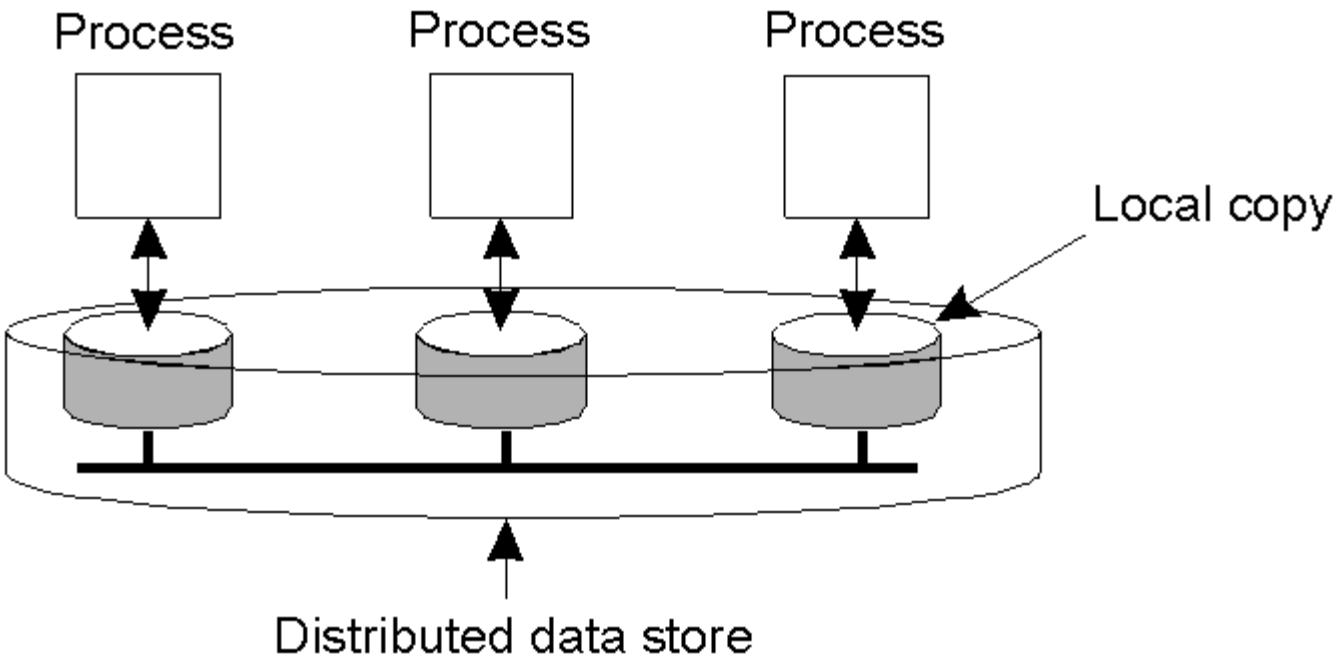
Objektumtöbbszörözés (3)



a) Többszörözést támogató objektumok az elosztott rendszerben.

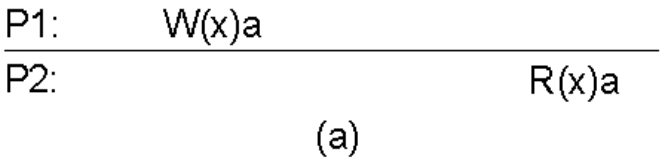
b) Az elosztott rendszer felel a többszörözés kezeléséért.

Adatközpontú konzisztenciamodellek

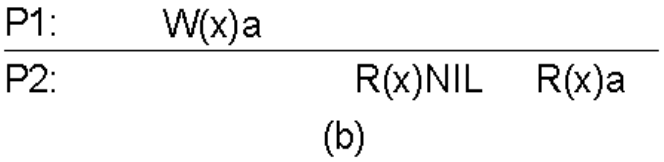


- A fizikailag több folyamat között elosztott és többszörözött logikai adattár általános felépítése.

Szigorú konzisztencia



(a)



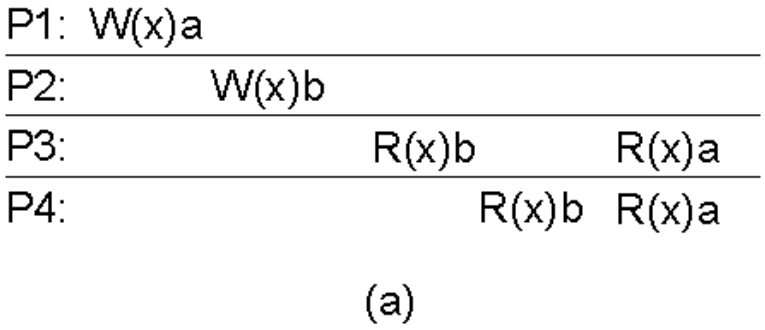
(b)

Azonos adatelemen műveletet végző két folyamat viselkedése:

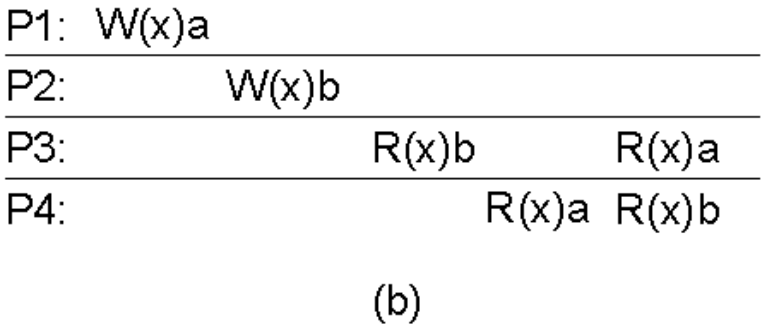
a) szigorúan konzisztens adattár

b) nem szigorúan konzisztens adattár

Lineáris és soros konzisztencia (1)



(a)



(b)

a) A sorosan konzisztens adattár.

b) A sorosan konzisztens kritériumot nem teljesítő adattár.

Lineáris és soros konzisztencia (2)

Process P1	**Process P2**	**Process P3**
x = 1; print (y, z);	y = 1; print (x, z);	z = 1; print (x, y);

- Három egyidejűleg végrehajtott folyamat.

Lineáris és soros konzisztencia (3)

```
x = 1;
print ((y, z);
y = 1;
print (x, z);
z = 1;
print (x, y);
```

Prints: 001011

Signature: 001011

(a)

```
x = 1;
y = 1;
print (x,z);
print(y, z);
z = 1;
print (x, y);
```

Prints: 101011

Signature: 101011

(b)

```
y = 1;
z = 1;
print (x, y);
print (x, z);
x = 1;
print (y, z);
```

Prints: 010111

Signature: 110101

(c)

```
y = 1;
x = 1;
z = 1;
print (x, z);
print (y, z);
print (x, y);
```

Prints: 111111

Signature: 111111

(d)

- Ez előző ábrán bemutatott folyamatok 4 lehetséges végrehajtási sorrendje.

Okozati konzisztencia (1)

- Szükséges feltétel:

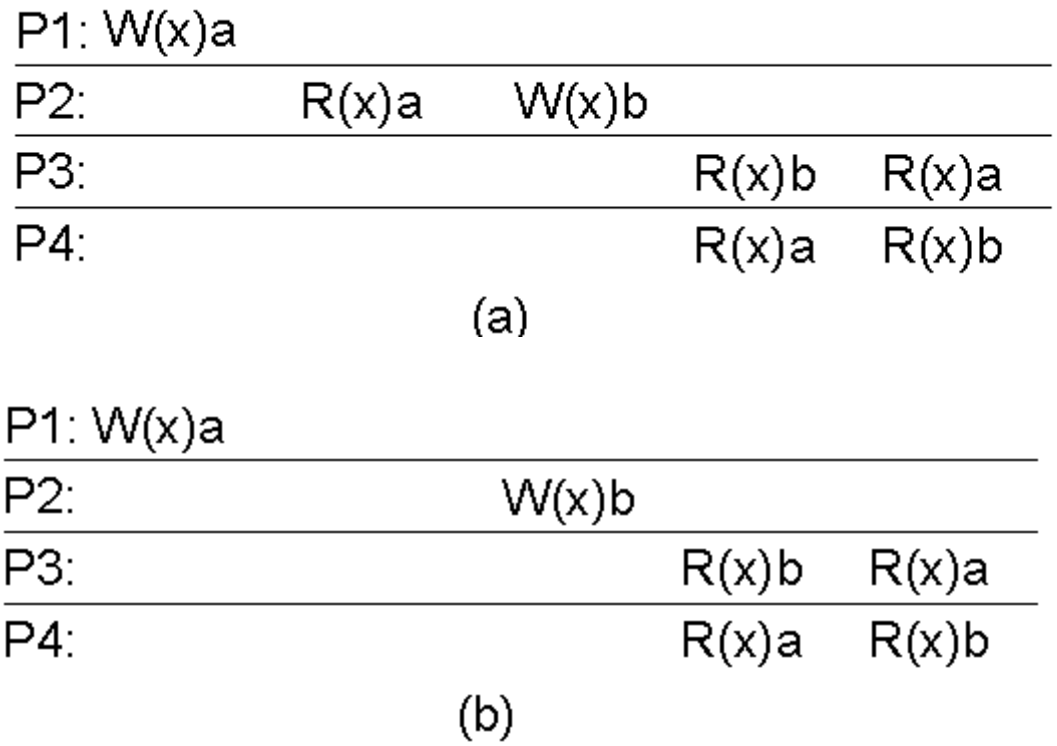
A potenciálisan okozatilag összefüggő írási műveleteket valamennyi folyamatnak ugyanabban a sorrendben kell látnia.

Okozati konzisztencia (2)

P1:	W(x)a			W(x)c	
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

- Ez a sorrend megengedett az okozatilag konzisztens adattárban, de tiltott a sorosan vagy a szigorúan konzisztens rendszerben.

Okozati konzisztencia (3)



a) Az okozati konzisztenciát megsértő műveleti sorrend.

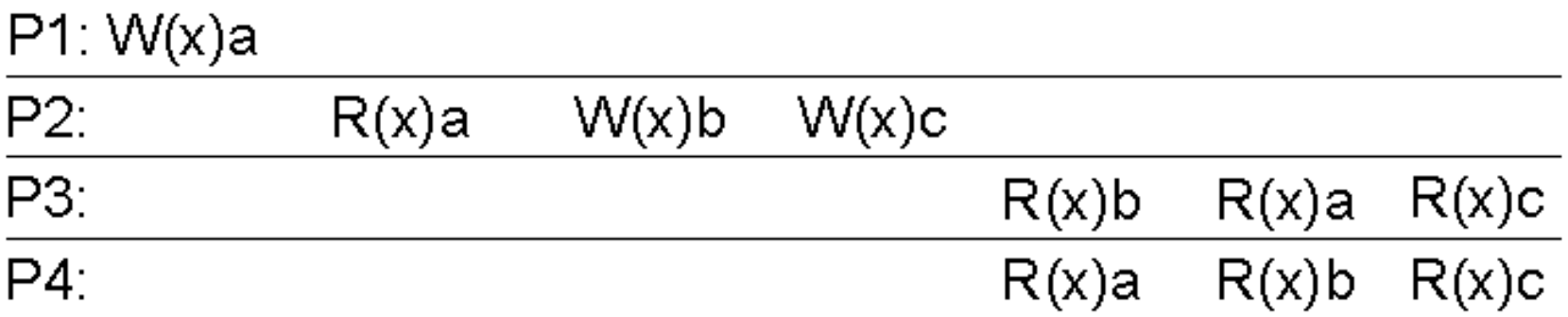
b) Az események konzisztens adattárban érvényes lehetséges sorrendje.

FIFO-konzisztencia (1)

- Szükséges feltétel:

Egy adott folyamat által végrehajtott valamennyi írási művelet eredményét az összes többi folyamatnak az írást végző folyamat által meghatározott sorrendben kell látnia.

FIFO-konzisztencia (2)



- Az események lehetséges sorrendje a FIFOkonzisztenciamodell szerint.

FIFO-konzisztencia (3)

```
x = 1;
print (y, z);
y = 1;
print(x, z);
z = 1;
print (x, y);
```

Prints: 00

(a)

```
x = 1;
y = 1;
print(x, z);
print ( y, z);
z = 1;
print (x, y);
```

Prints: 10

(b)

```
y = 1;
print (x, z);
z = 1;
print (x, y);
x = 1;
print (y, z);
```

Prints: 01

(c)

- A korábbi ábrán látható 3 folyamat utasításainak különböző végrehajtási sorrendje.

FIFO-konzisztencia (4)

Process P1	**Process P2**
x = 1;	y = 1;
if (y == 0) kill (P2);	if (x == 0) kill (P1);

- Két konkurens folyamat..

Gyenge konzisztencia (1)

Tulajdonságok:

- Az adattárhoz tartozó szinkronizáló változók elérése sorosan konzisztens.
- A szinkronizáló változóval végzett újabb művelet mindaddig nem engedélyezett, amíg az összes korábbi írási művelet mindenütt be nem fejeződött.
- Az adatelemeken végzett egyetlen írási vagy olvasási művelet sem engedélyezett, amíg az összes szinkronizáló változó értelmezett művelet mindenütt be nem fejeződött.

Gyenge konzisztencia (2)

```
int a, b, c, d, e, x, y;          /* variables */
int *p, *q;                       /* pointers */
int f( int *p, int *q);           /* function prototype */

a = x * x;                        /* a stored in register */
b = y * y;                        /* b as well */
c = a*a*a + b*b + a * b;          /* used later */
d = a * a * c;                    /* used later */
p = &a;                           /* p gets address of a */
q = &b                            /* q gets address of b */
e = f(p, q)                       /* function call */
```

- A program fragment in which some variables may be kept in registers.

Gyenge konzisztencia (3)

P1: W(x)a	W(x)b	S			
P2:			R(x)a	R(x)b	S
P3:			R(x)b	R(x)a	S

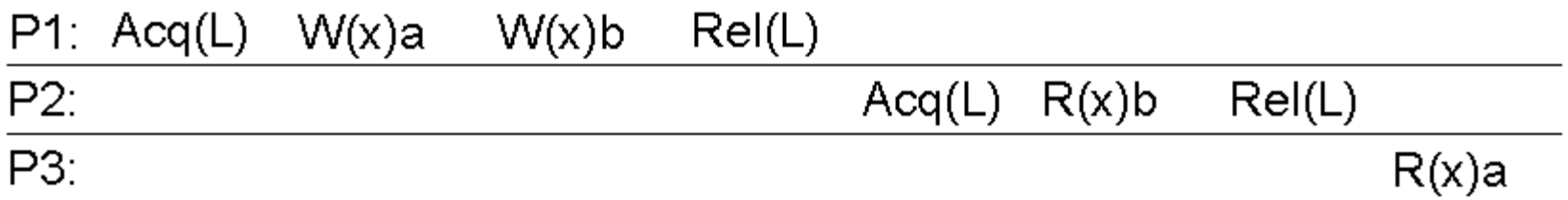
(a)

P1: W(x)a	W(x)b	S		
P2:			S	R(x)a

(b)

a) Gyenge konzisztencia eseten megengedett eseménysorrend.
b) Gyenge konzisztencia esetén tiltott eseménysorrend.

Feloldó konzisztencia (1)



- Az eseményeknek a feloldó konzisztenciamodellben megengedett lehetséges sorrendje..

Feloldó konzisztencia (2)

Szabályok:

- Megosztott adaton bármely olvasási/írási művelet előtt minden korábbi igénylésnek be kell fejeződnie.
- Feloldási művelet előtt a folyamatnak az összes korábbi írási/olvasási műveletét be kell fejeznie.
- A szinkronizáló változók elérése FIFOkonzisztens.

Belépő konzisztencia (1)

Feltételek:

- Szinkronizáló változó adott folyamat általi igénylése addig nem történhet, amíg az általa védett megosztott adat valamennyi frissítőművelete be nem fejeződött az adott folyamat számára.
- A folyamat csak akkor szerezheti meg a kizárólagos ellenőrzést a szinkronizáló változó felett, ha egyetlen más folyamat sem birtokolja.
- Miután egy folyamat megszerezte a kizárólagos ellenőrzést egy szinkronizáló változó felett, bármely más folyamat nem kizárólagos ellenőrzési kérelme addig nem teljesíthető, amíg a kizárólagos jogot az azt birtokló folyamat fel nem adja.

Belépő konzisztencia (2)

P1:	Acq(Lx)	W(x)a	Acq(Ly)	W(y)b	Rel(Lx)	Rel(Ly)						
P2:							Acq(Lx)	R(x)a				R(y)NIL
P3:									Acq(Ly)	R(y)b		

- A belépő konzisztenciában érvényes eseménysorrend.

Adatközpontú konziszteniamodellek összehasonlítása

Konzisztencia	Leírás
Szigorú	Valamennyi megosztott hozzáférés abszolút sorrendje.
Lineáris	A megosztott hozzáféréseket valamennyi folyamat pontosan ugyanabban a sorrendben látja. A hozzáférések sorrendjét (nem egyedi) globális időbélyeg határozza meg.
Soros	A megosztott hozzáféréseket valamennyi folyamat pontosan ugyanabban a sorrendben látja. A hozzáférések nincsenek idő szerint sorba rendezve.
Okozati	Az okozatilag összefüggő megosztott hozzáféréseket valamennyi folyamat pontosan ugyanabban a sorrendben látja.
FIFO	Valamennyi folyamat a többi írási műveleteit pontosan a kiadás sorrendjében fogja látni. A különböző folyamatok által végrehajtott írási műveletek sorrendje bármi lehet.

(a)

Konzisztencia	Leírás
Gyenge	A megosztott adat konzisztens voltára csak a szinkronizálás végrehajtása után lehet számítani.
Feloldó	A megosztott adat konzisztensé válik a kritikus terület elhagyásakor.
Belépő	A kritikus területhez tartozó megosztott adat konzisztensé válik a kritikus területre való belépéskor.

(b)

a) Szinkronizáló műveleteket nem igénylő konzisztenciamodellek.

b) Szinkronizáló műveleteken alapuló konzisztenciamodellek.

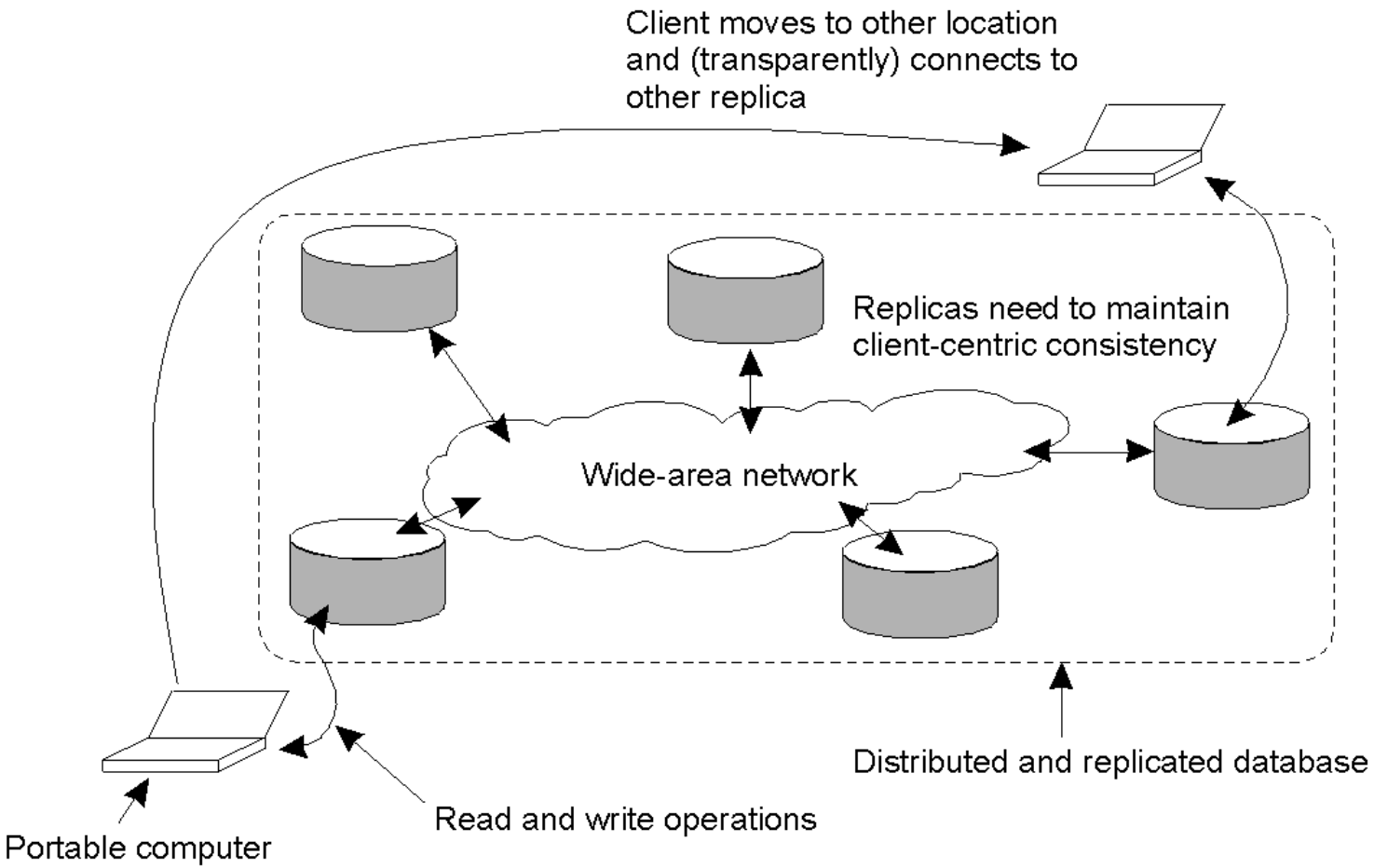
Kliensközpontú konzisztenciamodellek

- Adattárak különleges csoportja:
  - nincs párhuzamos módosítás
  - a legtöbb művelet csak olvassa az adattárat

Fokozatos konzisztencia (1)

- Frissítés hiányában valamennyi másolat lassan az összes többivel egyformává válik (a frissítéseket előbb-utóbb valamennyi másolathoz elküldjük)

Fokozatos konzisztencia (2)



- Az elosztott adatbázis különböző másolatait elérő
- mozgó felhasználó elve.

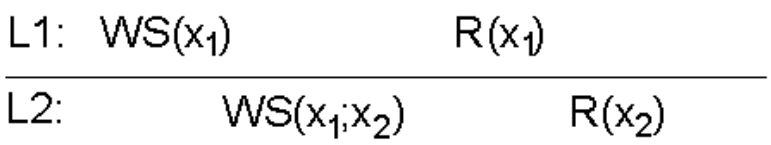
Fokozatos konzisztencia (3)

- **Probléma:** a felhasználó nem mindig ugyanahhoz a másolathoz csatlakozik
- **Megoldás:** kliensközpontú konzisztencia – egy adott felhasználó számára nyújt biztosítékot arra nézve, hogy az általa látott adattár konzisztens lesz

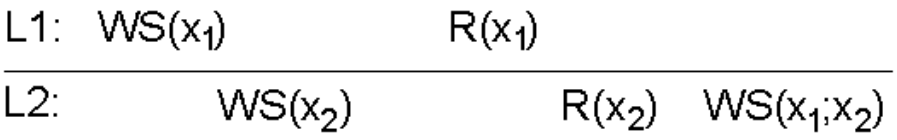
Monoton olvasás (1)

- Ha a folyamat beolvassa *x adatelemet*, *akkor x* minden további olvasásának ugyanazt vagy frissebb értéket kell szolgáltatnia.

Monoton olvasás (2)



(a)



(b)

- Adott P folyamat által végzett olvasási műveletek az adattár két különböző helyi másolatán futnak.

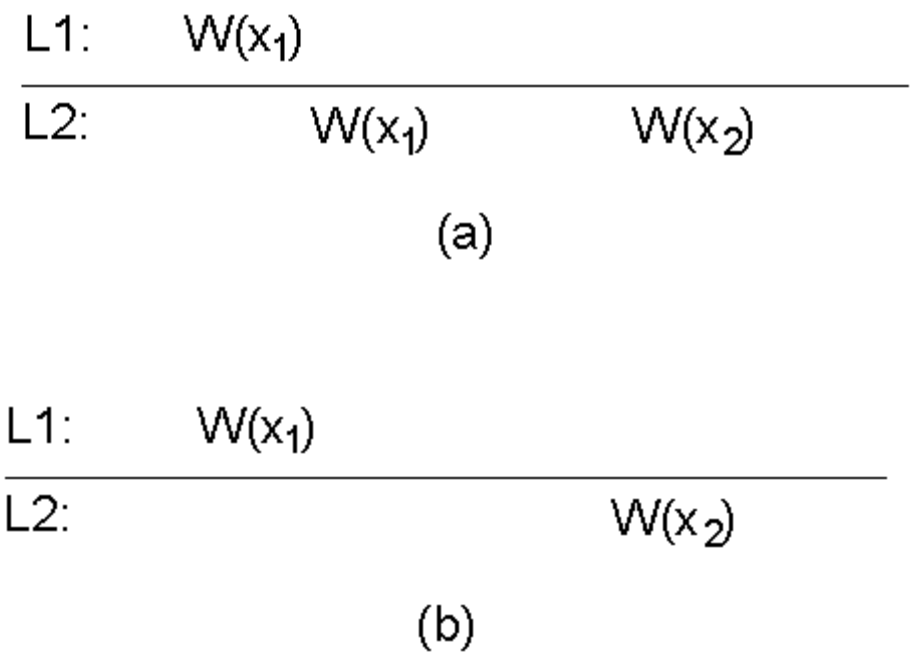
a) Monoton olvasási konzisztens adattár

b) Monoton olvasás követelményét ki nem elégítő adattár

Monoton írás (1)

- Adott folyamat által végrehajtott *x adatelemet* módosító írási műveletnek be kell fejeződnie, mielőtt ugyanez a folyamat újabb írási műveletet hajtana végre ugyanezen az adatelemen.

Monoton írás (2)



- Egyetlen P folyamat által ugyanazon adattár két különböző másolatán végrehajtott írási műveletek sorrendje.

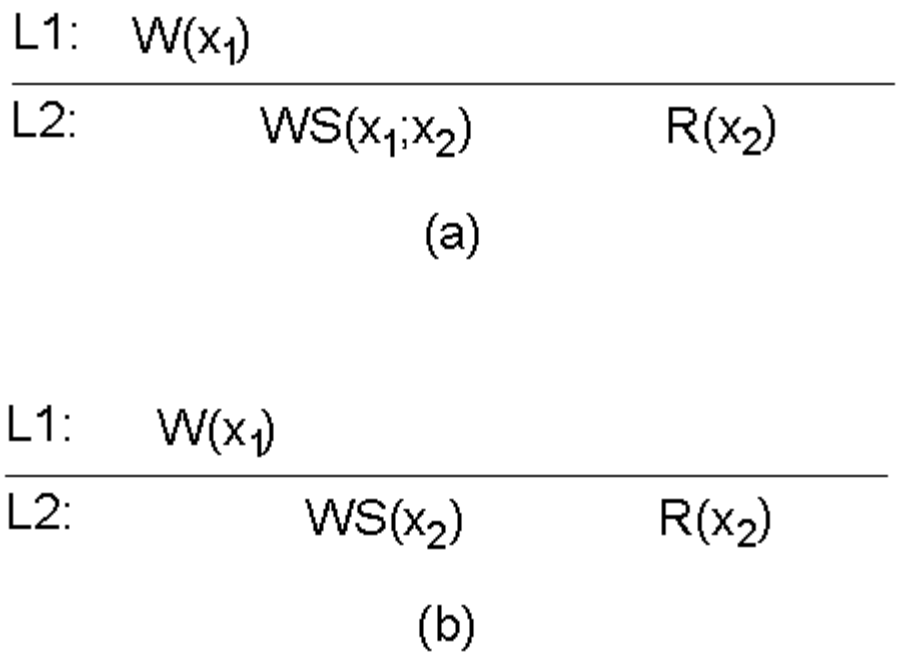
a) Monoton írási konzisztenciával rendelkező adattár esete

b) Monoton írási konzisztencia követelményét nem teljesítő adattár esete

Olvasd az írásod (1)

- Adott folyamat által *x adatelemen végrehajtott* írási művelet eredményének mindig láthatónak kell lennie a folyamat által *x adatelemen* végrehajtott későbbi olvasási művelet számára.

Olvasd az írásod (2)



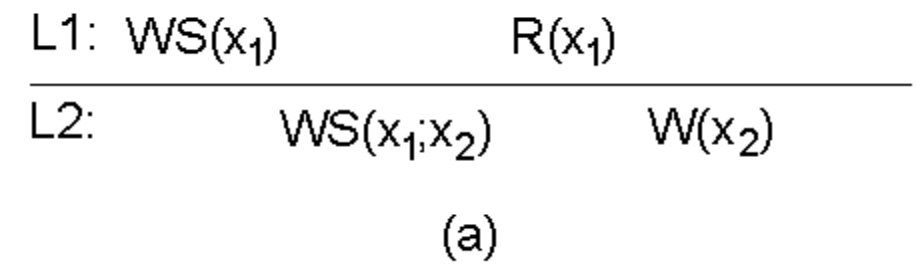
a) Az „olvasd az írásod” konzisztencia követelményét kielégítő adattár

b) „olvasd az írásod” konzisztenciának nem megfelelő adattár

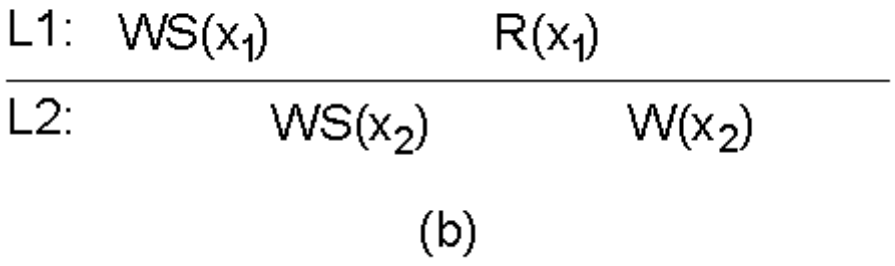
Írás olvasás után (1)

- A folyamat által *x adatelemen végrehajtott írási* művelet, amely *x adatelemnek a folyamat által* korábban történt olvasását követi, feltétlenül ugyanazt az értéket vagy annál frissebbet állít be, mint amit korábban olvasott.

Írás olvasás után (2)



(a)



(b)

a) Az „írás olvasás után” konzisztens adattár

b) Az „írás olvasás után” konzisztencia követelményét nem teljesítő adattár

Kliensközpontú konzisztencia implementálása (1)

- Minden írási művelethez egyedi azonosítót rendelünk.

- Minden kliens tárol:
  - Olvasáskészlet
  - Íráskészlet

- **Monoton olvasás: olvasás előtt olvasáskészlet ellenőrzése**
  - Olvasáskészletben az írások kiadási helyének és idejének tárolása

- **Monoton írás:** írás előtt íráskészlet vizsgálata
- **Olvasd az írásod:** olvasás előtt íráskészlet vizsgálata
- **Írás olvasás után:** írás előtt olvasáskészlet vizsgálata

Kliensközpontú konzisztencia implementálása (2)

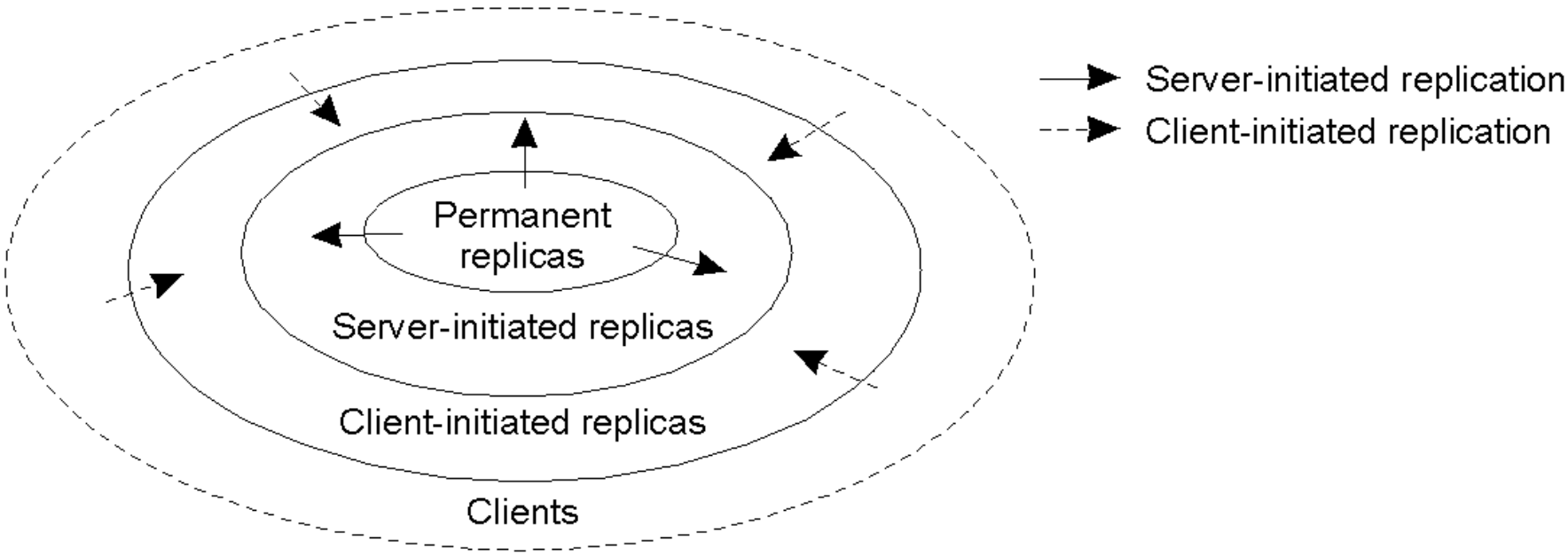
- **Probléma:** olvasás- és íráskészletek mérete
- **Megoldás:** szakaszokra bontás

- **Probléma:** teljes készlet átküldése a hálózaton
- **Megoldás:** időbélyeg vektorok

Elosztott protokollok

- Implementációs kérdések konzisztenciamodelltől függetlenül.
- **Másolat elhelyezése**
- **Frissítés terjesztése**
- **Járványprotokollok**

Másolat elhelyezése (1)

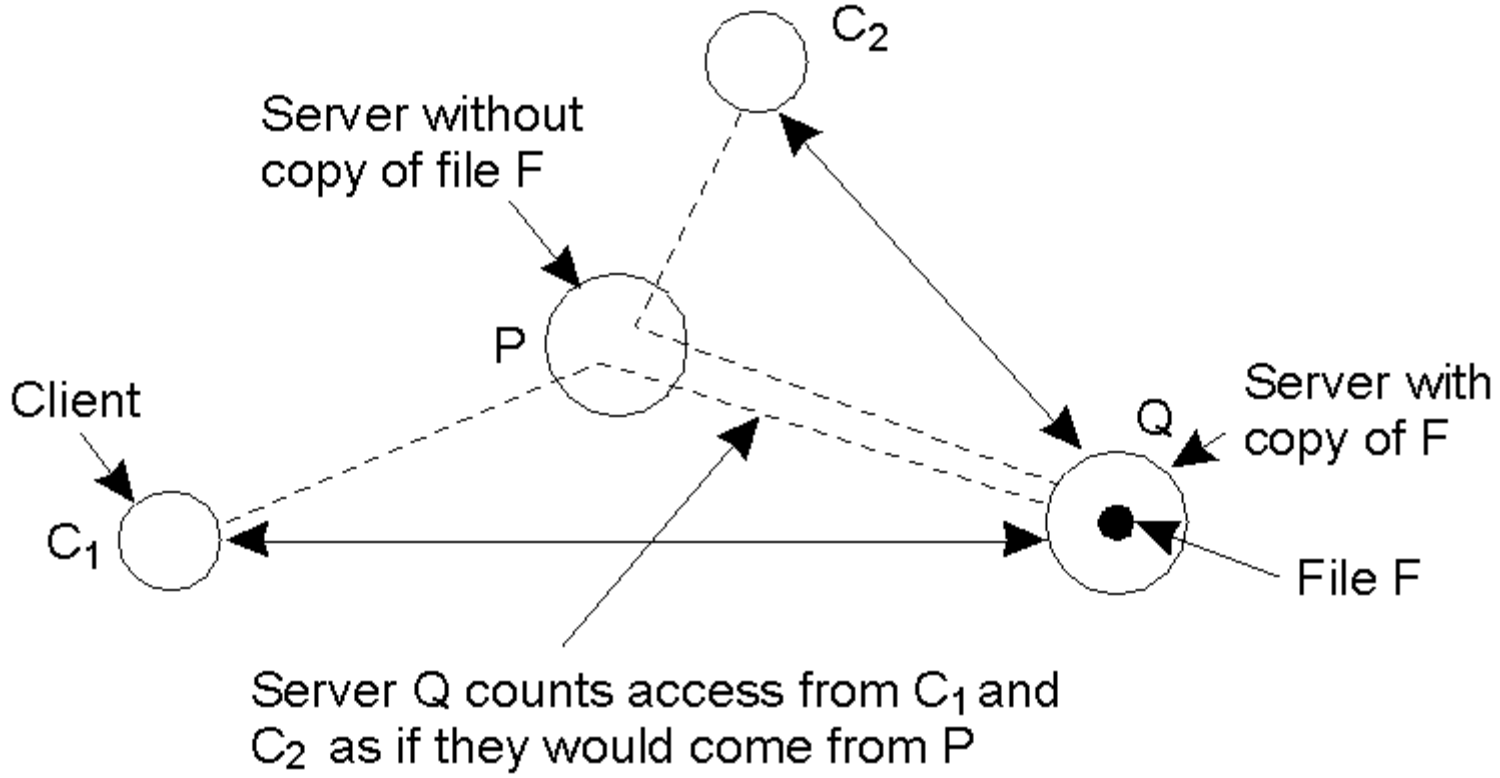


- Az adattár különböző típusú másolatainak három koncentrikus körként megjelenített logikai szervezése..

Másolat elhelyezése (2)

- **Másolatok típusa:**
  - **Állandó másolatok:** másolatok kezdeti készlete, tükrözések, biztonsági másolatok
  - **Szerver által kezdeményezett másolatok:** szükségmásolatok (dinamikus)
  - **Kliens által kezdeményezett másolatok:** kliensoldali gyorsítótár (cache)

Szerver által kezdeményezett másolatok



- Különböző kliensektől érkező kérések számlálása.

Frissítés terjesztése

- **Mit kell elterjeszteni?**
  - frissítésről szóló értesítés – érvénytelenítő protokollok
  - adatok – a módosított adat átküldése
  - frissítőművelet – mit kell a másolatoknak lefuttatnia
- **Ki kezdeményezzen?**
  - küldés alapú módszer (szerver alapú protokoll)
  - rendelés alapú módszer (kliens alapú protokoll)
  - vegyes módszer: haszonbérlet (dinamikus: kor alapú, megújítási frekvencia alapú, szerverterhelés alapú)
- **Hogyan küldjük?**
  - egycímű átvitel
  - többcímű átvitel

Rendelő és küldő protokollok

Témakör	Küldés alapú	Rendelés alapú
Szerverállapot	A kliensmásolatok és -gyorsítótárak listája	Nincs
Küldött üzenetek	Frissítés (esetleg letöltő frissítés később)	Lekérdezés és frissítés
Válaszidő a kliensnél	Azonnali (vagy letöltő frissítés ideje)	Letöltő frissítés ideje

- A küldés és a rendelés alapú protokollok összehasonlítása az egyszerveres, többkliensű rendszerekben.

Járványprotokollok (1)

- Fokozatos konzisztencia implementálásához

- **fertőző:** terjesztendő frissítéssel rendelkező
- **fertőzhető:** még nem frissített
- **elkülönített:** frissített, de terjesztésre nem képes

- **Frissítésterjesztő modellek:**
  - küldési módszer
  - rendelési módszer
  - küldés-rendelés módszer

- az eddigi módszerekkel a fertőzés előbb-utóbb elterjed

Járványprotokollok (2)

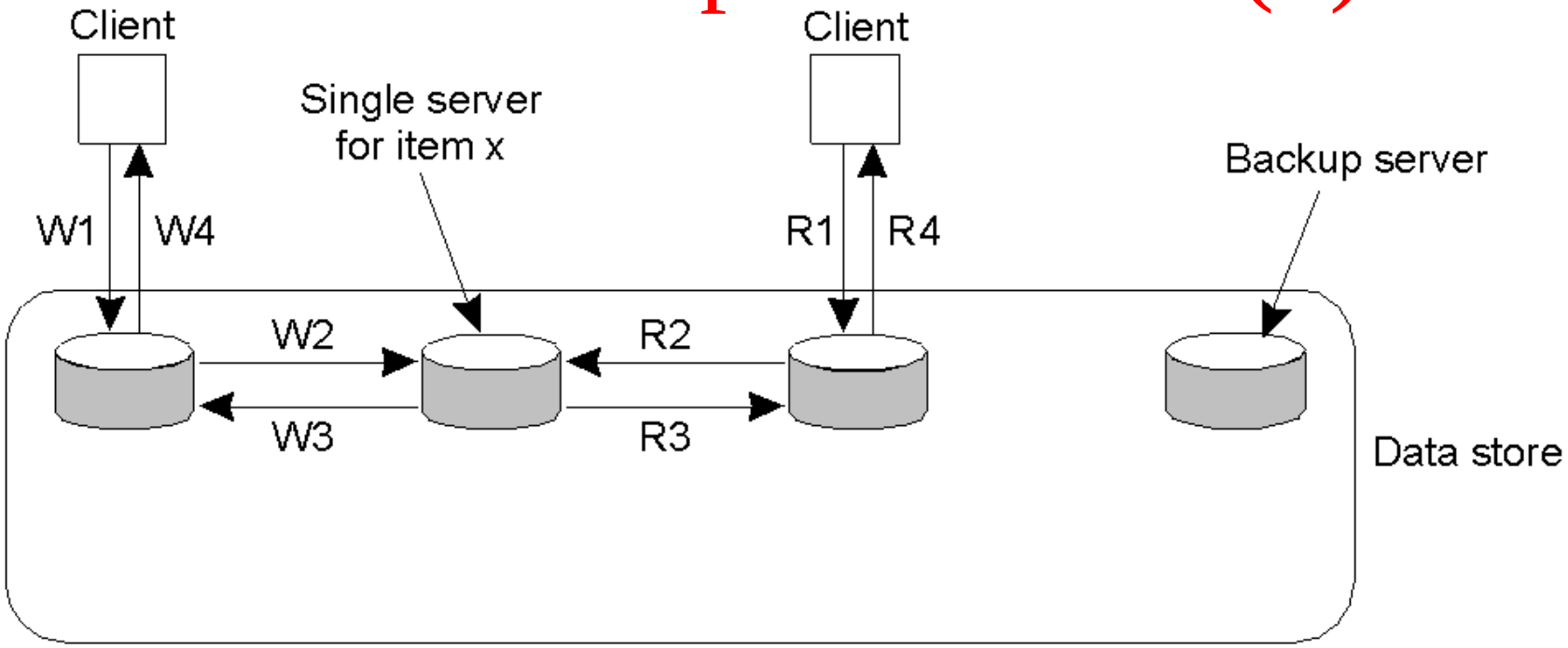
- másik módszer: **rémhírterjesztés / pletykálkodás**
  - nincs garancia a teljes elterjedésre

- **Adatok eltávolítása:**
  - törlést nehéz elterjeszteni
  - halotti bizonyítványok – törlés, mint frissítés
  - halotti bizonyítvánnyal rendelkezők mikor törölhetők?

Elsődleges másolaton alapuló protokollok

- Az adattár minden elemének van elsődleges másolata, aki felel az írási műveletek koordinálásáért.

Távoli írás protokollok (1)

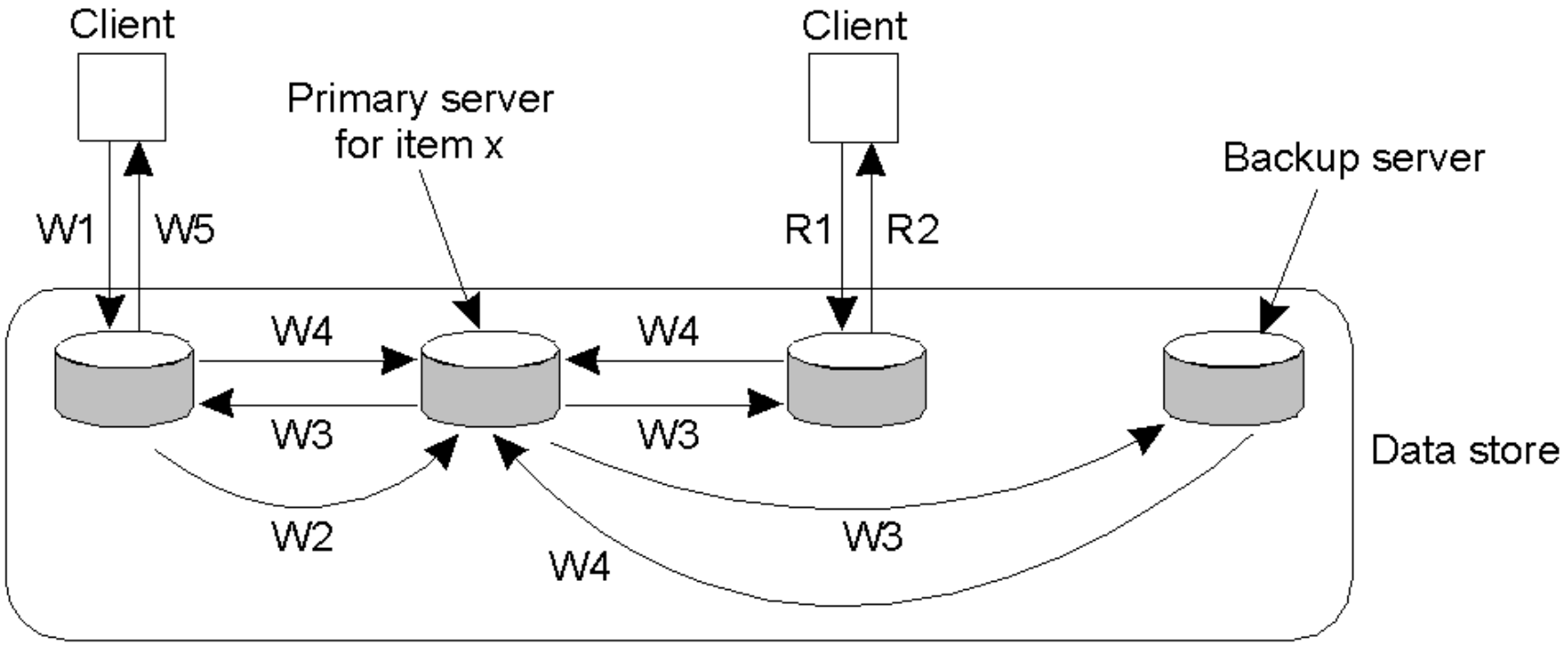


W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

R1. Read request
R2. Forward request to server for x
R3. Return response
R4. Return response

- Elsődleges távoli írás-protokoll, az olvasási és írási műveletek
- elvégzésére kizárólagosan hivatott rögzített szerverrel..

Távoli írás protokollok (2)

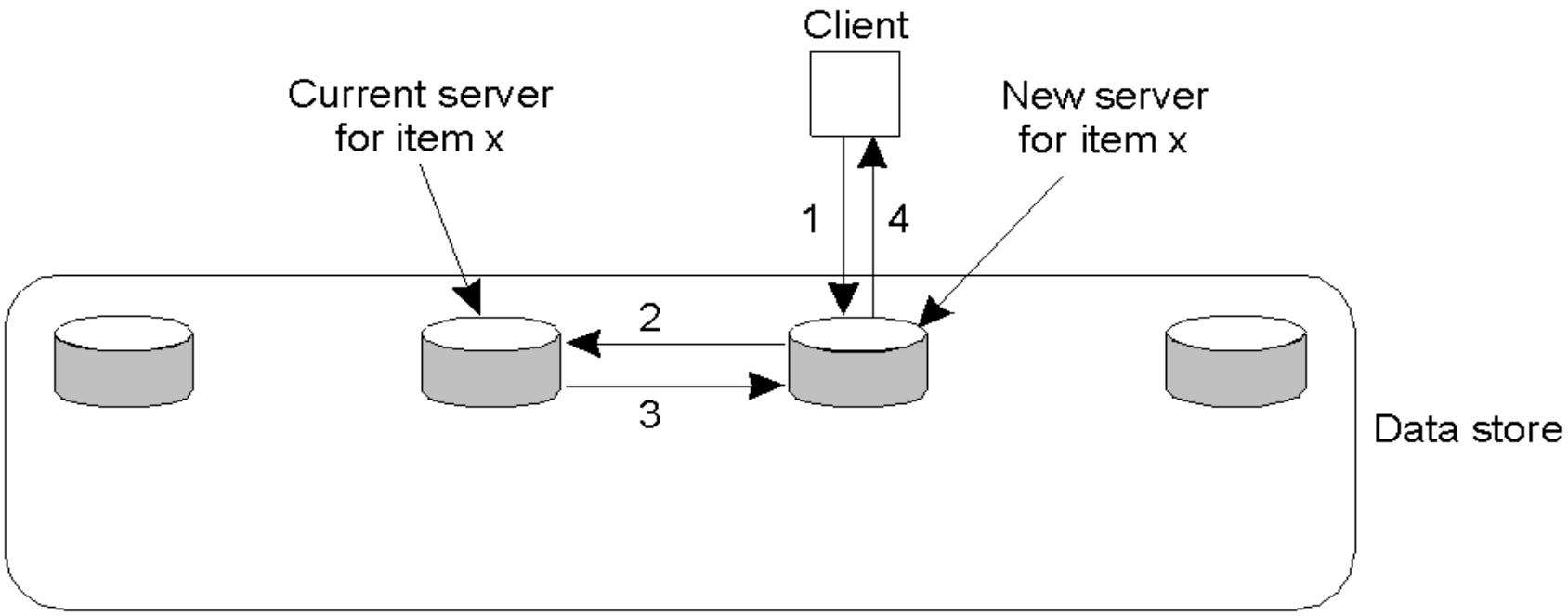


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

- Az elsődleges másolatprotokoll alapelve.

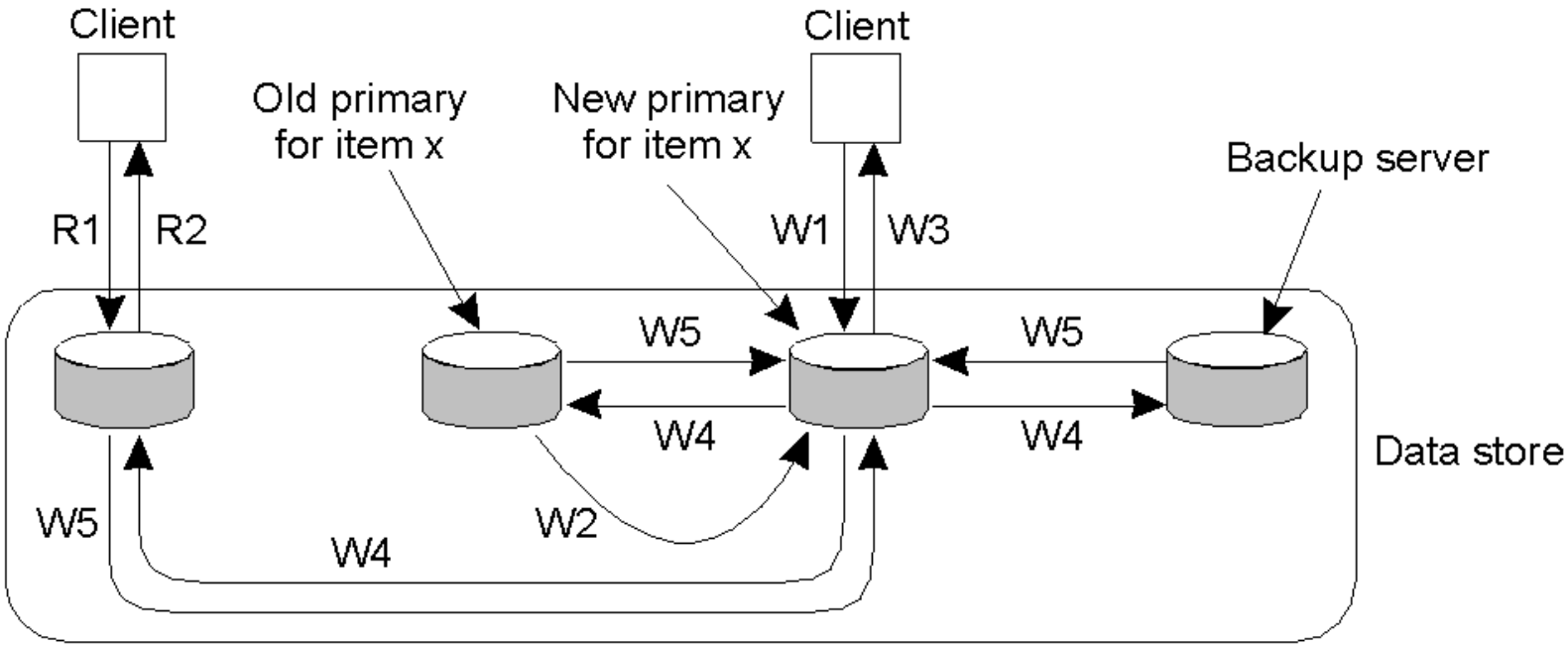
Helyileg író protokollok (1)



1. Read or write request
2. Forward request to current server for x
3. Move item x to client's server
4. Return result of operation on client's server

- Elsődleges másolat alapú helyileg író protokoll, amelyben az adatelem egyetlen példánya vándorol.

Helvileg író protokollok (2)



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

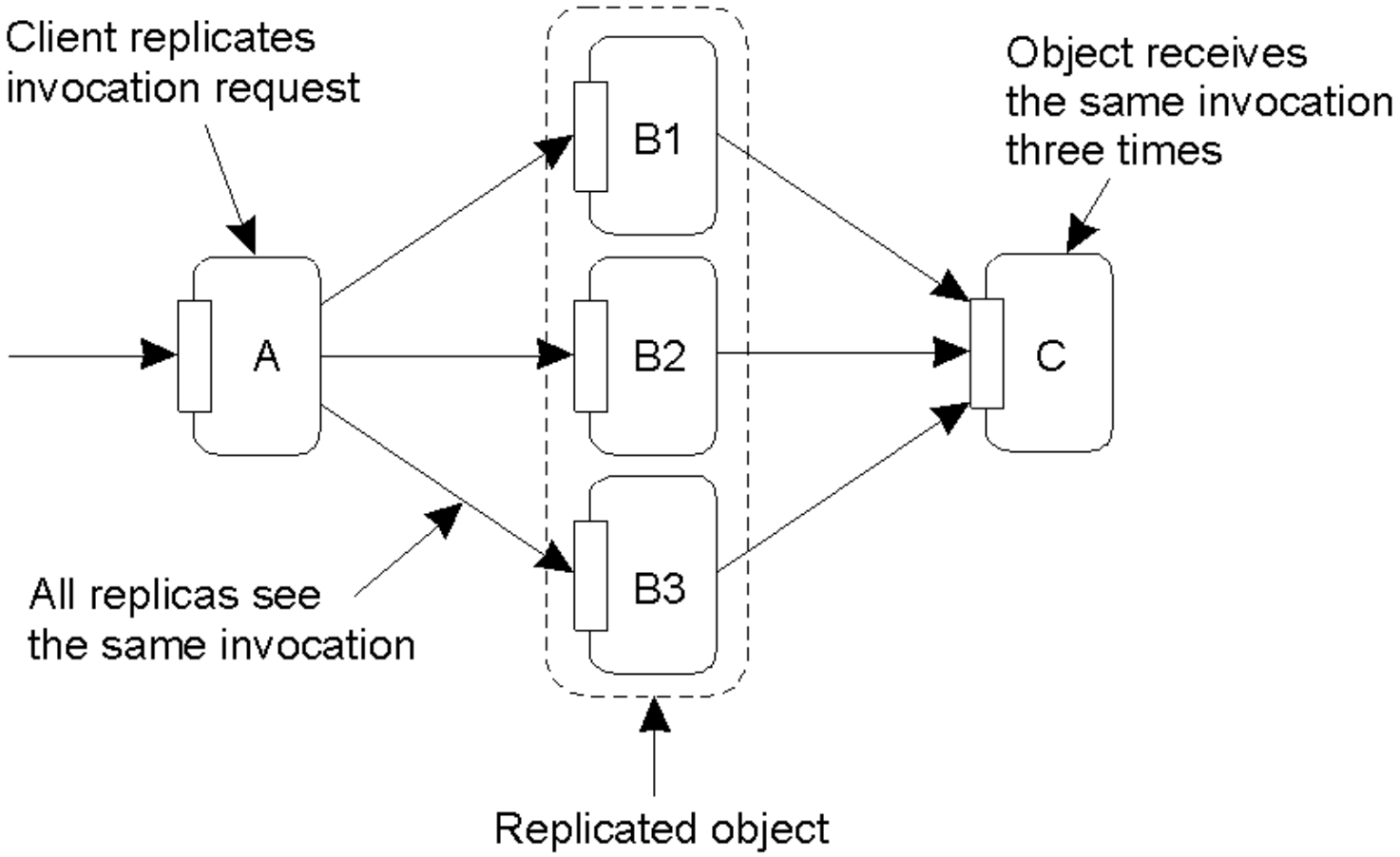
R1. Read request
R2. Response to read

- A frissítést végrehajtani kívánó folyamatok között vándorló elsődleges másolat protokollja.

Többszálú írás-protokollok

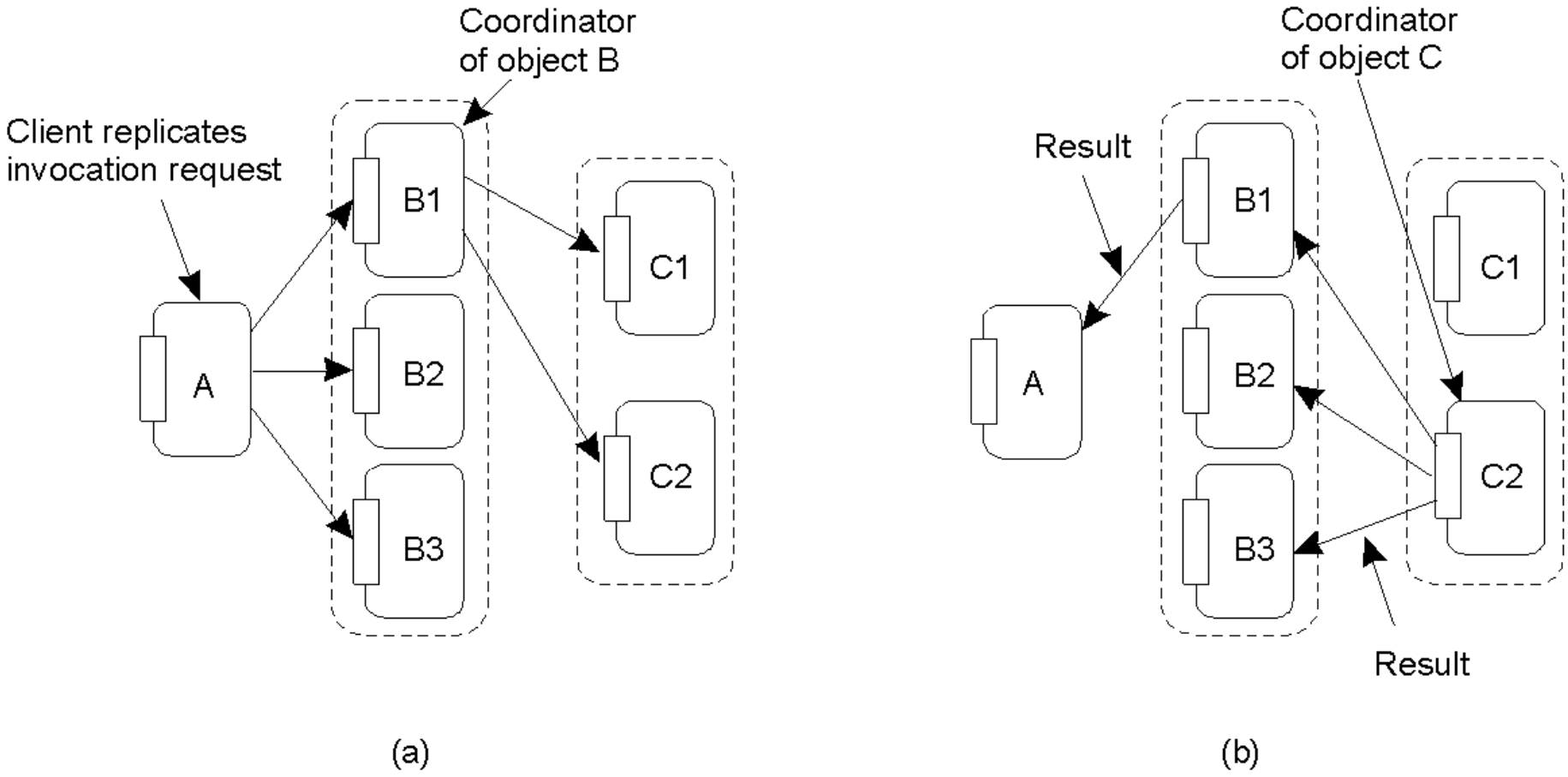
- Az írási művelet több másolaton is végrehajtható

Aktív többszörözés (1)



- A többszörözött hívás problémája

Aktív többszörözés (2)



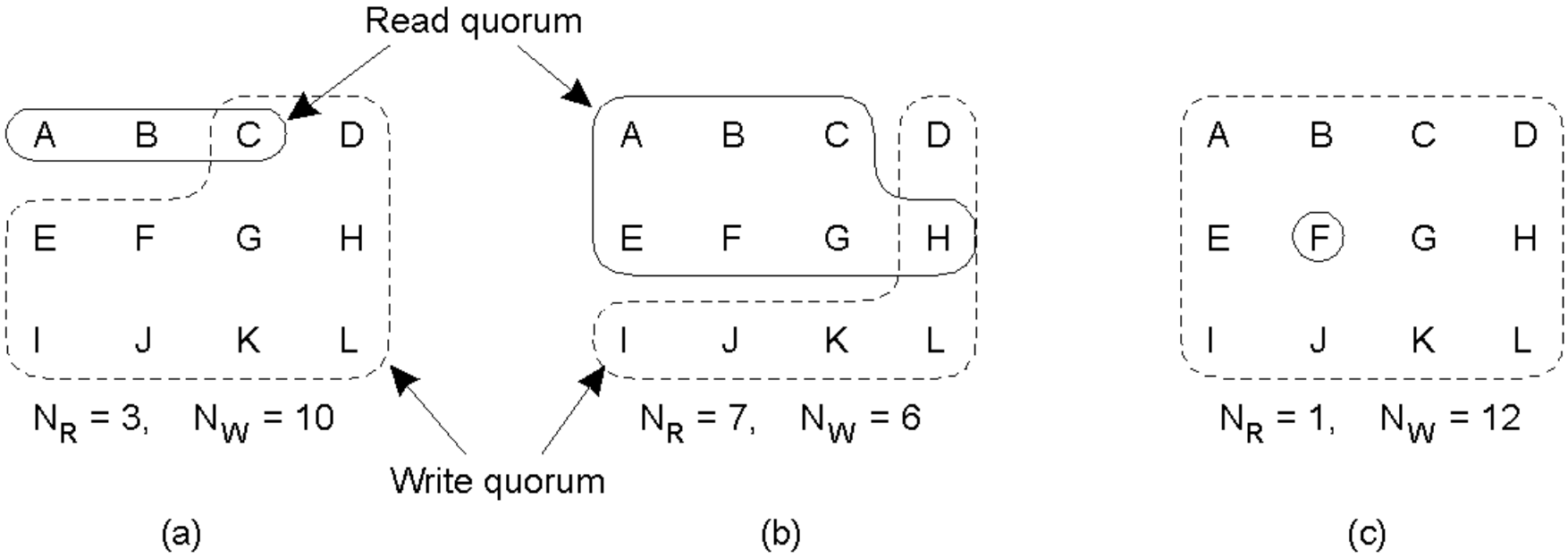
a) Továbbítja a többszörözött objektumtól érkező hívási kérést egy másik többszörözött objektumnak.

b) A válasz visszaküldése az egyik többszörözött objektumtól a másiknak..

Testület alapú protokollok (1)

- Többszörözött írás támogatása szavazás alapján
- Olvasás és írás előtt a többiektől engedélyt kell kérni
- Olvasási testület: NR
- Írási testület: NW
- NR + NW > N
- NW > N/2

Testület alapú protokollok (2)



A szavazó algoritmus három példája:
a) Az írási és olvasási testület taglétszámának helyes megválasztása
b) Lehetséges írás-írás ütközéshez vezető taglétszám
c) A ROWA (read one, write all) néven ismert helyes taglétszámválasztás

Gyorsítótár-egyezőségi protokollok (1)

- Egyöntetűség-észlelő stratégia – mikor veszik észre a tár eltérő voltát
  - Statikus
  - Dinamikus

- Mikor történik az észlelés?
  - tranzakció előtt
  - tranzakció közben
  - tranzakció lezárásakor

Gyorsítótár-egyezőségi protokollok (2)

- Egyöntetűség-biztosítási stratégia – hogyan maradnak konzisztensek az adatok
  - megosztott adat csak a szerveren
  - változáskor érvényesítő üzenet
  - változáskor frissítés terjesztése
  - keresztülíró gyorsítótár
  - visszaíró gyorsítótár