13.Realizarea serializabilitatii prin blocari

Pentru a putea asigura serializabilitatea tranzactiilor sistemele de gestiune pun la dispozitie posibilitatea de blocare a articolelor. Daca o tranzactie blocheaza un articol, celelalte tranzactii care vor si ele sa aiba acces la acel articol pot fi puse in asteptare pana la deblocarea acestuia.
Exista mai multe modele de blocare, prezentate in continuare.

6.4.1. Modelul LOCK/UNLOCK

In cadrul acestui model exista o singura primitiva de blocare, LOCK, ea ducand la obtinerea unui acces exclusiv la articol pentru tranzactia care il blocheaza (celelalte tranzactii nu pot nici scrie nici citi articolul). Deblocarea se face cu UNLOCK.
Vom presupune in continuare ca o tranzactie

- nu blocheaza un articol deja blocat de ea
- nu deblocheaza un articol pe care nu l-a blocat.

In acest caz se poate realiza **testul de serializabilitate** astfel:

Se construieste **graful de precedenta** G astfel

- Nodurile sunt tranzactiile planificarii
- Daca pentru vreun articol (notat simbolic A) avem in S secventa
  Ti: UNLOCK A
  Tj: LOCK A
  atunci vom avea un arc in graf de la nodul Ti la nodul Tj
- Daca graful are cicluri atunci S nu e serializabila.
- Daca nu are cicluri e serializabila si planificarea seriala echivalenta se obtine prin sortarea topologica a grafului G

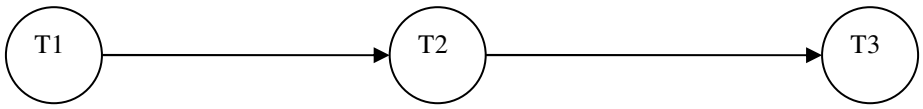
Sortarea topologica se face astfel:

- Se alege un nod care nu are arce care intra (neexistand cicluri exista cel putin un astfel de nod)
- Se listeaza tranzactia asociata nodului dupa care acesta este sters dion graf impreuna cu toate arcele care ies din el
- Procesul se reia

Exemplu:

T1	T2	T3
	LOCK A	
	UNLOCK A	
		LOCK A
		UNLOCK A
LOCK B		
UNLOCK B		
	LOCK B	
	UNLOCK B	

Graful este:



Nu are cicluri deci planificarea e serializabila. Planificarea seriala echivalenta este:

T1; T2; T3

Protocolul de blocare in doua faze

Definitie: O tranzactie respecta protocolul de blocare in doua faze daca toate blocarile preced toate deblocarile

Acest protocol ne garanteaza serializabilitatea: daca toate tranzactiile respecta cerintele protocolului se poate demonstra ca orice planificare a lor e serializabila.
De asemenea se poate demonstra ca daca o tranzactie nu respecta protocolul pot exista executii neserializabile ale acelei tranzactii in conjunctie cu alte tranzactii:

Pentru o tranzactie care contine secventa:

```
UNLOCK A
LOCK B
```

Putem avea o planificare care contine:

```
   T1                  T2
UNLOCK A
                    LOCK A
                    LOCK B
                    UNLOCK A
                    UNLOCK B
LOCK B
```

Care are un graf de precedenta care contine un ciclu.

Protocolul de blocare in 2 faze implica insa uneori operatii de roll-back in cascada:

T1	T2
LOCK A **LOCK B**	
READ A **WRITE A**	
UNLOCK A	
	LOCK A **READ A** **WRITE A** **UNLOCK A**
READ B **WRITE B**	
ROLLBACK	

In momentul Rollback pentru T1 este necesar Rollback si pentru T2 deoarece T2 a citit date scrise de T1, date care prin operatia de Rollback se pierd. O astfel de planificare se numeste planificare cu rollback in cascada (eng.: cascading aborts)

Exista pentru a evita si astfel de cazuri varianta **protocolului de blocare stricta in 2 faze** care implica eliberarea toturor articolelor blocate la sfarsitul tranzactiei. In acest caz tranzactia T2 din exemplul anterior porneste abia dupa terminarea complete a tranzactiei T1.

6.4.2. Modelul RLOCK/WLOCK/UNLOCK

In cadrul acestui model exista o doua primitive de blocare:

- RLOCK (blocare pentru citire). Oricate tranzactii pot bloca acelasi articol pentru citire dar o tranzactie nu poate bloca pentru scriere un articol blocat cu RLOCK
- WLOCK (blocare pentru citire). Duce la obtinerea unui acces exclusiv la articol pentru tranzactia care il blocheaza. Celelalte tranzactii nu mai pot sa blocheze cu RLOCK sau WLOCK acel articol.
- Deblocarea pentru ambele tipuri se face cu UNLOCK.

Vom presupune ca si anterior ca o tranzactie

- nu blocheaza un articol deja blocat de ea
- nu deblocheaza un articol pe care nu l-a blocat.

Si in acest caz se poate construi (altfel decat in paragraful anterior) un **graf de precedenta** din care se poate deduce daca planificarea e serializabila sau nu.

Observatie importanta: Si in acest caz protocolul de blocare in doua faze este valabil: Daca toate blocarile (de orice fel, la citire sau la scriere) preced toate deblocarile, planificarea este serializabila.

Test de serializabilitate pentru modelul cu blocari pentru citire si scriere

Intrare: o planificare P a mulţimii de tranzacţii T1, T2, ..., Tk.
Ieşirea: raspunsul daca planificarea P este serializabilă, si daca da planificarea serială echivalentă.
Metoda: construirea grafului de precedenta G, similar cu modelul anterior: fiecarei tranzactii ii corespunde un nod al grafului iar arcele se traseaza astfel:

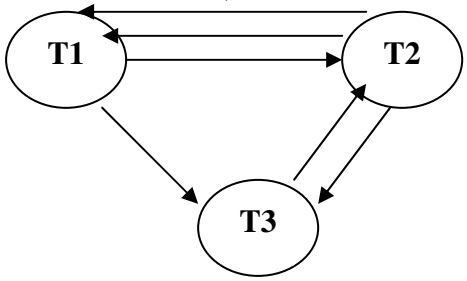
1. Fie Ti tranzactia care executa RLOCK A iar Tj urmatoarea tranzactie (diferita de Ti) care face WLOCK A. Trasam atunci un arc de la Ti la Tj.
2. Fie Ti tranzactia care face WLOCK A si Tj urmatoarea tranzactie (daca exista) care face WLOCK A. Se traseaza un arc de la Ti la Tj. De asemenea, in acest ultim caz fie Tm tranzactia care face RLOCK A dupa ce Ti elibereaza pe A dar inainte de blocarea acestuia de catre Tj. Se traseaza un arc de la Ti la Tm. Nota: Daca in cazul 2 Tj nu exista atunci Tm este urmatoarea tranzactie care face RLOCK A dupa eliberarea lui A de catre Ti.

Rezultat: Daca graful obtinut contine cicluri, P nu este serializabila.
Daca insa nu contine cicluri atunci P este serializabila iar planificarea seriala echivalenta se obtine prin sortarea topologica a grafului (ca in cazul modelului anterior, cu blocare simpla).

Exemplu:

T1	T2	T3
	WLOCK A	
		RLOCK B
1	UNLOCK A	
	5	UNLOCK B
	WLOCK B	
RLOCK A		6
	UNLOCK B	
UNLOCK A	4	
2		WLOCK A
WLOCK B		
3		UNLOCK A
UNLOCK B		
	RLOCK B	
	UNLOCK B	

- Graful este cel de mai jos. Cum are cicluri, planificarea nu este serializabila.



Arcele s-au trasat pe baza regulilor de mai sus :

- Prima regula a generat arcele de tip R-W: 4, 5.
- Prima parte a celei de-a doua reguli a generat arcele de tip W-W: 2, 6.

- A doua parte a celei de-a doua reguli a generat arcele de tip W-R: 1, 3. Arcul 3 e generat luand in considerare nota de la a doua regula.

6.4.3. Blocarea articolelor structurate ierarhic

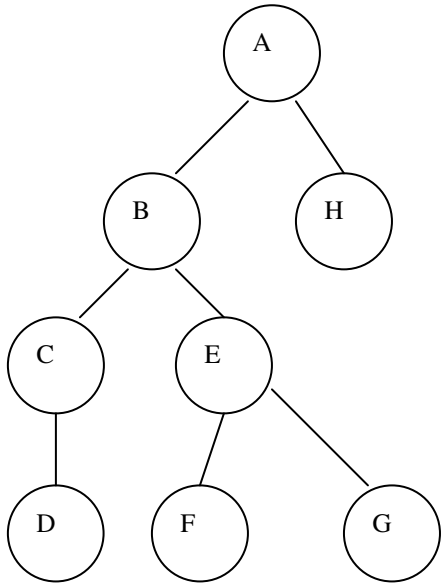
Exista cazuri in care articolele unei baze de date se pot reprezenta ca nodurile unui arbore.
In acest caz exista un protocol care garanteaza serializabilitatea numit protocol de arbore. Acesta este urmatorul:

- Cu exceptia primului articol blocat, nici un articol nu poate fi blocat decat daca tatal sau este deja blocat
- Nici un articol nu este blocat de doua ori de aceeasi tranzactie.

Exemplu:

Urmatoarea tranzactie respecta acest protocol, pentru structurarea ierarhica prezentata in continuare:

LOCK B – primul articol blocat
LOCK C – blocarea lui C este necesara pentru a realiza blocarea lui D
LOCK D
UNLOCK C – C poate fi acum deblocat
LOCK E – blocarea lui E este necesara pentru a realiza blocarea lui F
LOCK F
UNLOCK E – E poate fi acum deblocat
UNLOCK D
UNLOCK F
UNLOCK B



6.5. Controlul concurentei prin etichete timp

Se poate efectua un control al corectitudinii executiei concurente a mai multor tranzactii

- Fara mecanisme de blocare.
- Utilizand etichete-timp (timestamps)

In acest caz:

1. Fiecare articol are asociate doua etichete-timp: una de citire iar cealalta de scriere.
2. Fiecare tranzactie are asociata o eticheta timp (de exemplu: momentul lansarii tranzactiei)
3. Cand o tranzactie citeste sau scrie un articol, se actualizeaza eticheta-timp corespunzatoare a articolului respectiv la valoarea etichetei-timp a tranzactiei.

O tranzactie sesizeaza incalcarea ordinii seriale (in care caz ea trebuie abortata si relansata ulterior) in urmatoarele doua cazuri:

1. O tranzactie incearca sa citeasca un articol scris in viitor
2. O tranzactie vrea sa scrie un articol citit in viitor.

Urmatoarele operatii sunt insa valide:

1. O tranzactie incearca sa citeasca un articol citit in viitor
2. O tranzactie incearca sa scrie un articol scris in viitor (in acest caz ea nu scrie articolul dar isi continua executia).