ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Βάσεις Δεδομένων ΙΙ

Ταυτοχρονισμός

Μανώλης Μαραγκουδάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Πανεπιστήμιο Αιγαίου-
Τμήμα Μηχανικών
Πληροφοριακών και
Επικοινωνιακών Συστημάτων

Βάσεων Δεδομένων ΙΙ

Ενότητα 2:Ταυτοχρονισμός

Μανώλης Μαραγκουδάκης

www.icsd.aegean.gr/mmarag

Περιεχόμενα

- Κλειδώματα – 2 Phase Locking
- Πώς γίνεται στην πράξη?
- Αδιέξοδα

Πώς ελέγχουμε σειριοποιησιμότητα στην πράξη?

- Η σειριοποιησιμότητα όψεων είναι πολύ ακριβή για να ελεγχθεί, ούτως ή άλλως…
- Οι αλγόριθμοι για τον έλεγχο σειριοποιησιμότητας συγκρούσεων παίρνουν n^2[ενίοτε και καλύτερα] για n δοσοληψίες.
- Μόνο που ο έλεγχος αυτός γίνεται αφού φτιαχτεί το χρονοπρόγραμμα, ήτοι γίνεται λίγο αργά …

- Γι' αυτό και έχουμε αναπτύξει on-line πρωτόκολλα ελέγχου του ταυτοχρονισμού των δοσοληψιών.

Κλειδώματα και πρωτόκολλα

- **Πρωτόκολλο κλειδώματος** είναι ένα συγκεκριμένο είδος πρωτοκόλλων, που διασφαλίζουν την σειριοποιησιμότητα μιας σειράς ενεργειών του DBMS, στη βάση κανόνων για το τι μπορεί[ή δεν μπορεί] να προσπελάσει μια δοσοληψία.
- Η βασική έννοια γύρω από την οποία οργανώνεται ένα τέτοιο πρωτόκολλο, είναι η έννοια του «**κλειδώματος**» (η «κλειδιού» ή «κλειδαριάς» ...)

Στην αγγλική lock[ενίοτε, και key].

Κλείδωμα

- **Κλείδωμα** είναι μια μεταβλητή που σχετίζεται με ένα στοιχειώδες δεδομένο και περιγράφει την κατάστασή του, σε σχέση με πιθανές πράξεις που μπορούν να εφαρμοσθούν σε αυτό.

EMP

	ID	Name
Read@T1	12	XX
Written@T2	13	YY
	14	ZZ

Elmasri & Navathe

2 Phase Locking – 2PL

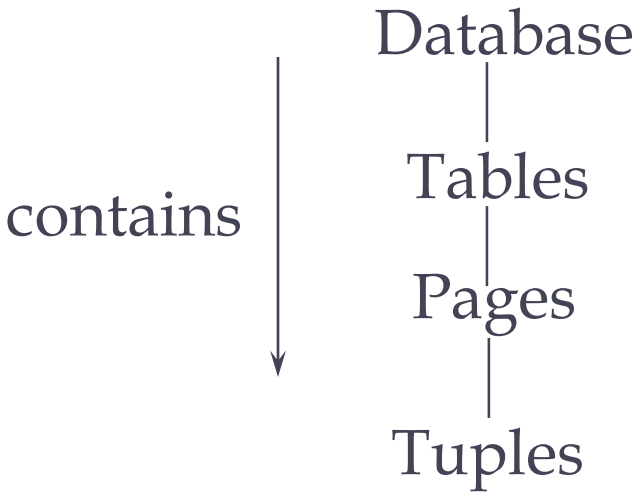
- Ο βασικός αλγόριθμος διαχείρισης του ταυτοχρονισμού δοσοληψιών είναι ο «Αλγόριθμος Κλειδώματος Δύο Φάσεων» -- στην αγγλική “2 Phase Locking” ή **2PL**

2PL

- Εξασφαλίζει τη σειριοποιησιμότητα επιτρέποντας στις δοσοληψίες να προσπελάσουν αντικείμενα, αν καταφέρουν να αποκτήσουν κάποιο κλείδωμα γι' αυτά.
- Δύο είδη κλειδώματος:
  - **Ανάγνωσης**: λαμβάνεις κλείδωμα για να διαβάσεις ένα αντικείμενο
  - **Εγγραφής**: λαμβάνεις κλείδωμα για να γράψεις ή/και για να διαβάσεις ένα αντικείμενο

Τι είναι «αντικείμενο»?

- Μπορούμε να κλειδώσουμε τη βάση σε πολλά «επίπεδα»
- Θεωρήστε προς το παρόν εγγραφές
- Στην πράξη, συνήθως μιλάμε για σελίδες



Ramakrishnan & Gehrke

Θα επανέλθουμε ...

Πρωτόλειος αλγόριθμος κλειδώματος ≠ 2PL (συνεχίζεται…)

- Κάθε ενέργεια διαβάσματος/γραψίματος απαιτεί να (περιμένεις μέχρι να) έχεις πάρει το σωστό κλείδωμα
- Για να διαβάσεις, πρέπει να πάρεις ένα read lock – το οποίο και θα αποκαλούμε **Shared lock** (**S-lock**). Είναι δυνατόν να υπάρχουν πολλές δοσοληψίες με S-lock για το ίδιο αντικείμενο

Πρωτόλειος αλγόριθμος κλειδώματος ≠ 2PL (συνέχεια)

- Για να γράψεις, πρέπει να πάρεις ένα write lock – το οποίο και θα αποκαλούμε **eXclusive lock** (X-lock). Αν μια δοσοληψία έχει X-lock για ένα αντικείμενο, απαγορεύεται οποιαδήποτε άλλη να έχει κάποιο κλείδωμα για το αντικείμενο αυτό
- Μπορείς να ξεκλειδώνεις (**Unlock**) αντικείμενα, αν δεν τα χρειάζεσαι.

Συμβολισμός

- $R_T(A)$: η δοσοληψία T διαβάζει το αντικείμενο A
- $W_T(A)$: η δοσοληψία T γράφει το αντικείμενο A
- **$S_T(A)$** : η δοσοληψία T παίρνει S-lock για το αντικείμενο A
- **$X_T(A)$** : η δοσοληψία T παίρνει X-lock για το αντικείμενο A
- **$U_T(A)$** : η δοσοληψία T ξεκλειδώνει το αντικείμενο A
- $COMMIT_T$: η δοσοληψία T τερματίζει επιτυχώς
- $ABORT_T$: η δοσοληψία T αποτυγχάνει

Παράδειγμα

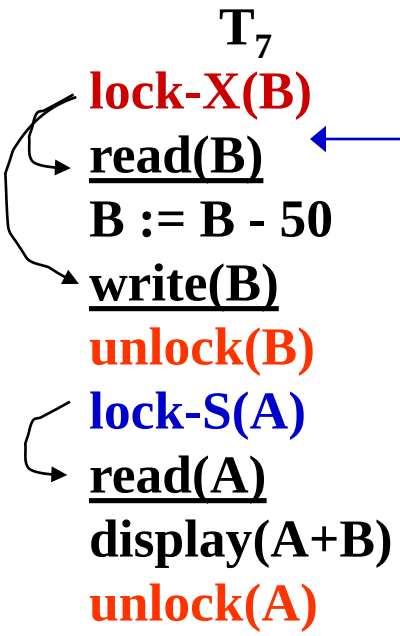
T_7

lock-X(B)
read(B)
B := B - 50
write(B)
unlock(B)
lock-S(A)
read(A)
display(A+B)
unlock(A).

$R_7(B)$; $W_7(B)$; $R_7(A)$.

$X_7(B)$; $R_7(B)$; $W_7(B)$; $U_7(B)$; $S_7(A)$; $R_7(A)$ $U_7(A)$

Παράδειγμα



Βασική υπόθεση: ξέρουμε από πριν όλες τις ενέργειες της δοσοληψίας ...

Συγκρούσεις

- Έστω δύο δοσοληψίες, T1 και T2, εκ των οποίων, η T1 ζητά ένα κλείδωμα και η T2 έχει ήδη ένα κλειδωμα για το ίδιο αντικείμενο.
- Ο πίνακας συγκρούσεων πρέπει να σας θυμίζει κάτι...
- **Άρνηση ⇒ «περίμενε μέχρι να γίνει unlock»**

		Lock held	
	T1\T2	Read	Write
Lock requested	Read	☑	☒
	Write	☒	☒

2 ειδών προβλήματα

- Αν προσπαθήσεις να αυξήσεις τις ταυτόχρονες δοσοληψίες, ξεκλειδώνοντας αντικείμενα πολύ νωρίς, μπορεί να έχεις προβλήματα ασυνέπειας
- Αν προσπαθήσεις να καθυστερήσεις το ξεκλείδωμα,
  - μπορεί να προκύψουν **αδιέξοδα** (όπου δύο δοσοληψίες περιμένουν –για πάντα– η μία την άλλη για το ξεκλείδωμα διαφορετικών αντικειμένων)
  - καθυστερείς (ενώ έχουμε πιο έξυπνους τρόπους)…

Στη συνέχεια, παραδείγματα από Korth, Silberschatz & Sudarshan…

Ξεκλειδώνοντας νωρίς...

Επειδή η T7 ξεκλειδώνει το B νωρίς, το αποτέλεσμα της T8 είναι λάθος...

T_7	T_8
lock-X(B)	
read(B)	
B := B - 50	
write(B)	
unlock(B)	
	lock-S(A)
	read(A)
	unlock(A)
	lock-S(B)
	read(B)
	unlock(B)
	display(A + B)
lock-X(A)	
read(A)	
A := A + 50	
write(A)	
unlock(A)	

Deadlock

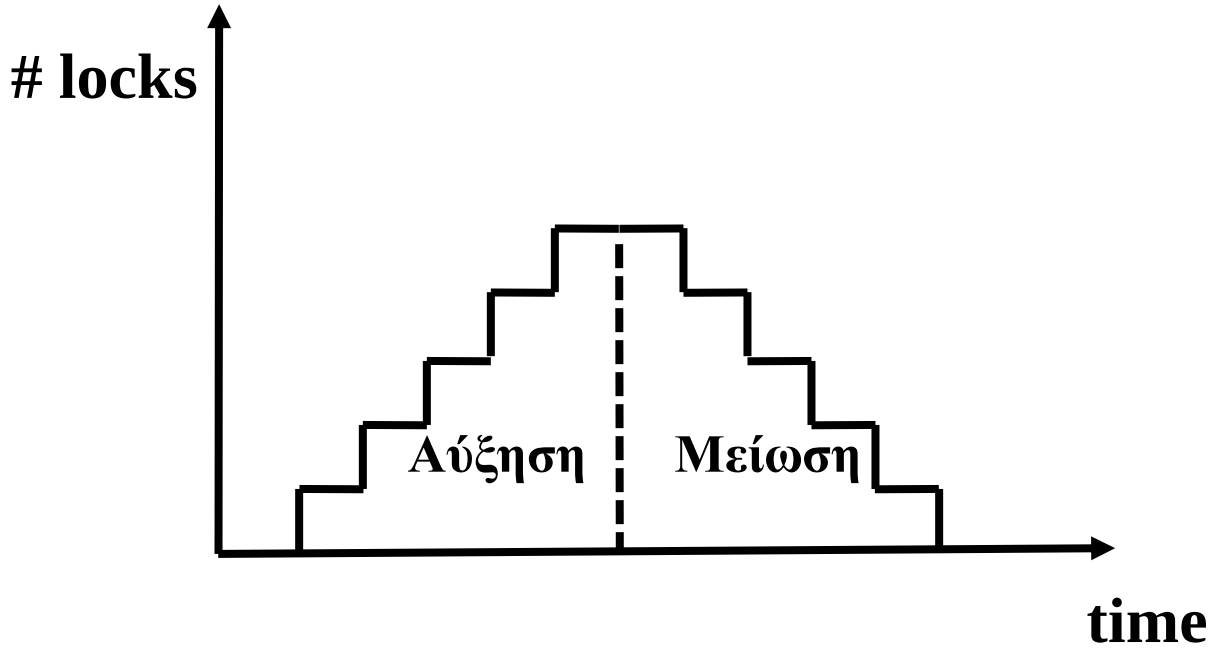
Η μία δοσοληψία έχει κλειδώσει το αντικείμενο που χρειάζεται η άλλη. Σταματούν και οι δύο, αναμένοντας η μία την άλλη ...

T_9	T_{10}
lock-X(B)	
read(B)	
B := B - 50	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	
read(A)	
A := A + 50	
write(A)	
unlock(B)	
unlock(A)	
	unlock(A)
	unlock(B)
	display(A + B)

2PL

- **Ζήτα κλείδωμα πριν πράξεις:**
  - Για να διαβάσεις, πρέπει να πάρεις ένα **Shared lock** (**S-lock**). Είναι δυνατόν να υπάρχουν πολλές δοσοληψίες με S-lock για το ίδιο αντικείμενο
  - Για να γράψεις, πρέπει να πάρεις ένα **eXclusive lock** (X-lock). Αν μια δοσοληψία έχει X-lock για ένα αντικείμενο, απαγορεύεται οποιαδήποτε άλλη να έχει κάποιο κλείδωμα για το αντικείμενο αυτό
- **Δεν μπορείς να ξεκλειδώσεις αντικείμενο μέχρι να έχεις πάρει και το τελευταίο κλείδωμα που χρειάζεσαι**

2 Phase Locking



Ξανά: έτσι και ξεκλειδώσεις αντικείμενο, δεν μπορείς να πάρεις άλλο κλείδωμα!!!

2PL?

T_9	T_{10}	T_7	T_8
lock-X(B)	lock-S(A)	lock-X(B)	lock-S(A)
read(B)	read(A)	read(B)	read(A)
B := B - 50	lock-S(B)	B := B - 50	unlock(A)
write(B)	unlock(A)	write(B)	lock-S(B)
lock-X(A)	unlock(B)	unlock(B)	unlock(B)
read(A)	display(A + B)	lock-X(A)	display(A + B)
A := A + 50		read(A)	
write(A)		A := A + 50	
unlock(B)		write(A)	
unlock(A)		unlock(A)	
ΕΠΙΤΡΕΠΤΟ	ΕΠΙΤΡΕΠΤΟ	ΠΑΡΑΒΑΣΗ	ΠΑΡΑΒΑΣΗ

ΥΠΟΜΝΗΜΑ
ΑΥΞΗΣΗ
ΜΕΙΩΣΗ
ΠΑΡΑΒΑΣΗ

Απλοϊκή περίπτωση που κάθε schedule έχει μόνο μία δοσοληψία

Διαχειριστής κλειδαριών

- Το τμήμα του DBMS που διαχειρίζεται τα κλειδώματα
- Αν το DBMS υποστηρίζει 2PL → **2PL scheduler**
- **Πίνακας κλειδαριών**:
  - ObjectID
    - TransactionID, LockType
    - Queue με δοσοληψίες εν αναμονή

	x	y	z
T1		S	
T2		S	
T3		W	

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S1: R1[x];W2[x];R1[x].

Βοήθεια:

- *Ας δοκιμάσω να βάλω locks & unlocks…*
- *Έχω μη επιτρεπτές συγκρούσεις?*
- *Έχουν όλες οι δοσοληψίες τις 2 φάσεις?*

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	y	z
T1		X	
T2			
T3			

X1[y];**R1[y]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	y	z
T1		X	
T2	S		
T3			

X1[y];**R1[y]**;S2[x];**R2[x]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Έχω ήδη το lock…

	x	y	z
T1		X	
T2	S		
T3			

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Unlock κάνω?
Το Τ3 θέλει το y

ΟΧΙ!!

	x	y	z
T1		X	
T2	S		
T3		W	

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Και τι κάνω?

	x	y	z
T1			X
T2	S		
T3		W	

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	**y**	**z**
T1			X
T2	S		
T3		X	

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];
W3[y];

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	y	z
T1			X
T2	S		
T3		X	

Έχω ήδη το lock…

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	y	z
T1			
T2	S		
T3		X	

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;U1[z]; ← Unlock ΑΜΕΣΑ

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	**y**	**z**
T1			
T2	S		S
T3		X	

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;U1[z];S2[z];**R2[z]**;

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

	x	y	z
T1			
T2	S		S
T3		X	S

ΔΕΝ υπάρχει σύγκρουση για S-locks!!

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;U1[z];S2[z];**R2[z]**;S3[z];**R3[z]**;

… vai …

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;U1[z];S2[z];**R2[z]**;S3[z];**R3[z]**;U_{123}[all] …

Μπορούσα να είχα κάνει ?

S2:

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

X1[y];**R1[y]**;S2[x];**R2[x]**;**W1[y]**;X1[z];U1[y];X3[y];**W3[y]**;**W1[z]**;U1[z];S2[z];**R2[z]**;S3[z];**R3[z]**;U_{123}[all] …

Άσκηση 2PL

- Είναι το παρακάτω χρονοπρόγραμμα συμβατό με το 2PL?
- R1(V) W2(X) R3(Y) W1(X) R1(V) R2(V) R3(Z) R2(Y) W3(V)

S1(V) R1(V) X2(X) W2(X)S3(Y) R3(Y)S2(V) S2(Y) U2(X)
X1(X) W1(X) R1(V) R2(V)S3(Z) R3(Z)
R2(Y)U1(V) U2(V) X3(V) W3(V)U123(all) C123

Θεώρημα

- **Όλα** τα χρονοπρογράμματα που προκύπτουν από ένα διαχειριστή κλειδαριών 2PL, έχουν **ακυκλικό γράφο σειριοποίησης**
- Το αντίστροφο ΔΕΝ ισχύει.

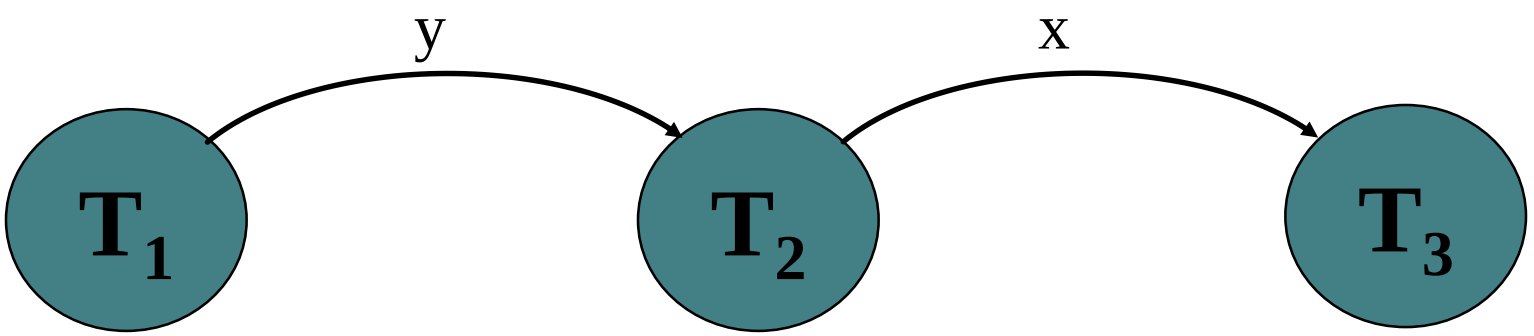
- Λήμμα: αν δεν έχει ακυκλικό γράφο σειριοποήσης ➔ δεν είναι 2PL.

Προκύπτει το παρακάτω χρονοπρόγραμμα από αλγόριθμο 2PL ?

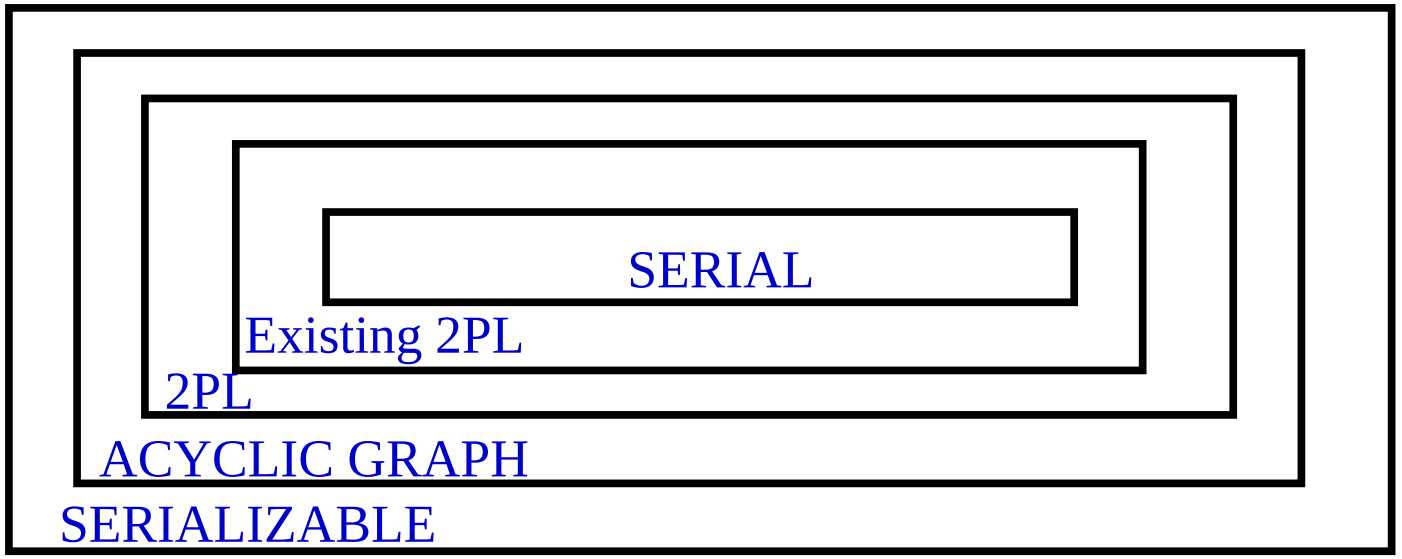
S3: W2[x];W3[x];W1[y];W2[y].

Μπορώ να το σειριοποιήσω?

- **S3**: W2[x];W3[x];W1[y];W2[y].

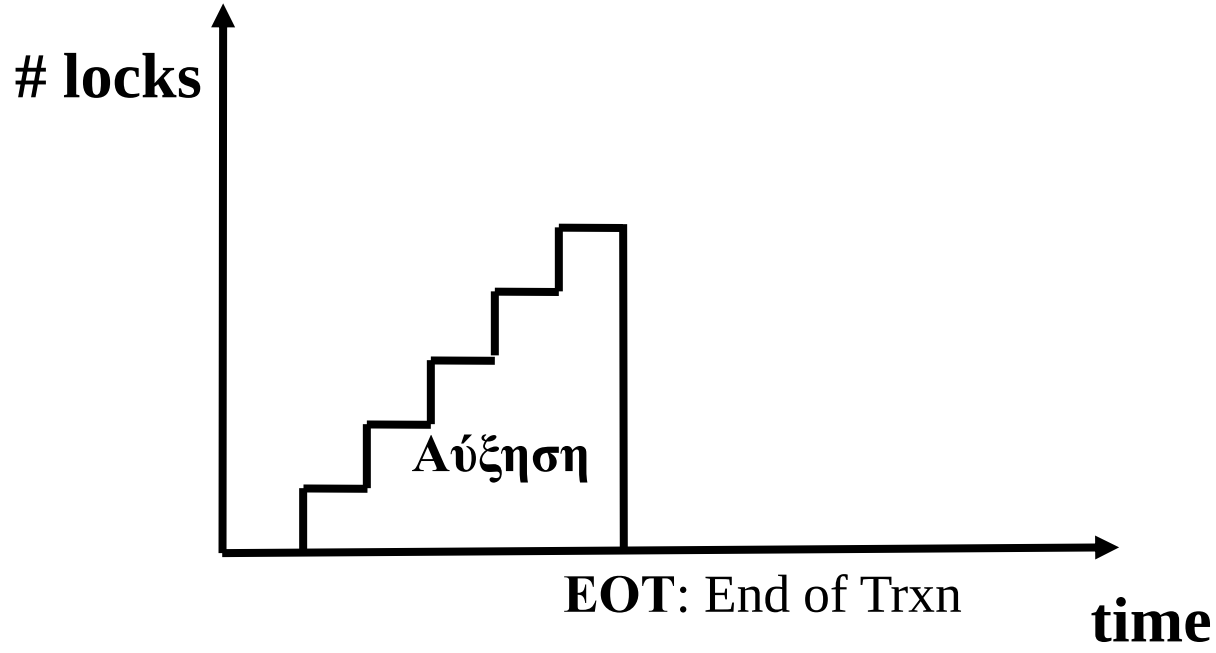


Venn διάγραμμα



Παραλλαγές

- **Αυστηρός 2PL**: όλα τα κλειδώματα αφήνονται στο τέλος της δοσοληψίας, μαζί.



Αυστηρός 2PL

- Στην πράξη όλα τα συστήματα χρησιμοποιούν αυστηρό 2PL…

Προκύπτει από Strict 2PL?

R1[y];R2[x];W1[y];W3[y];W1[z];R2[z];R3[z].

Περιεχόμενα

- Κλειδώματα – 2 Phase Locking
- Πώς γίνεται στην πράξη?
- Αδιέξοδα

Πώς γίνεται στην πράξη?

- Κάθε δοσοληψία ζητά από το διαχειριστή κλειδωμάτων το αντίστοιχο κλειδί
- Αυτός ελέγχει τον πίνακα κλειδωμάτων και αν μπορεί, παραχωρεί το κλειδί και ενημερώνει τον πίνακα
- Αν δεν μπορεί, χρησιμοποιεί μια ουρά αναμονής ανά αντικείμενο, όπου και καταχωρείται η εν λόγω δοσοληψία. Η δοσοληψία περιμένει.

Πώς γίνεται στην πράξη?

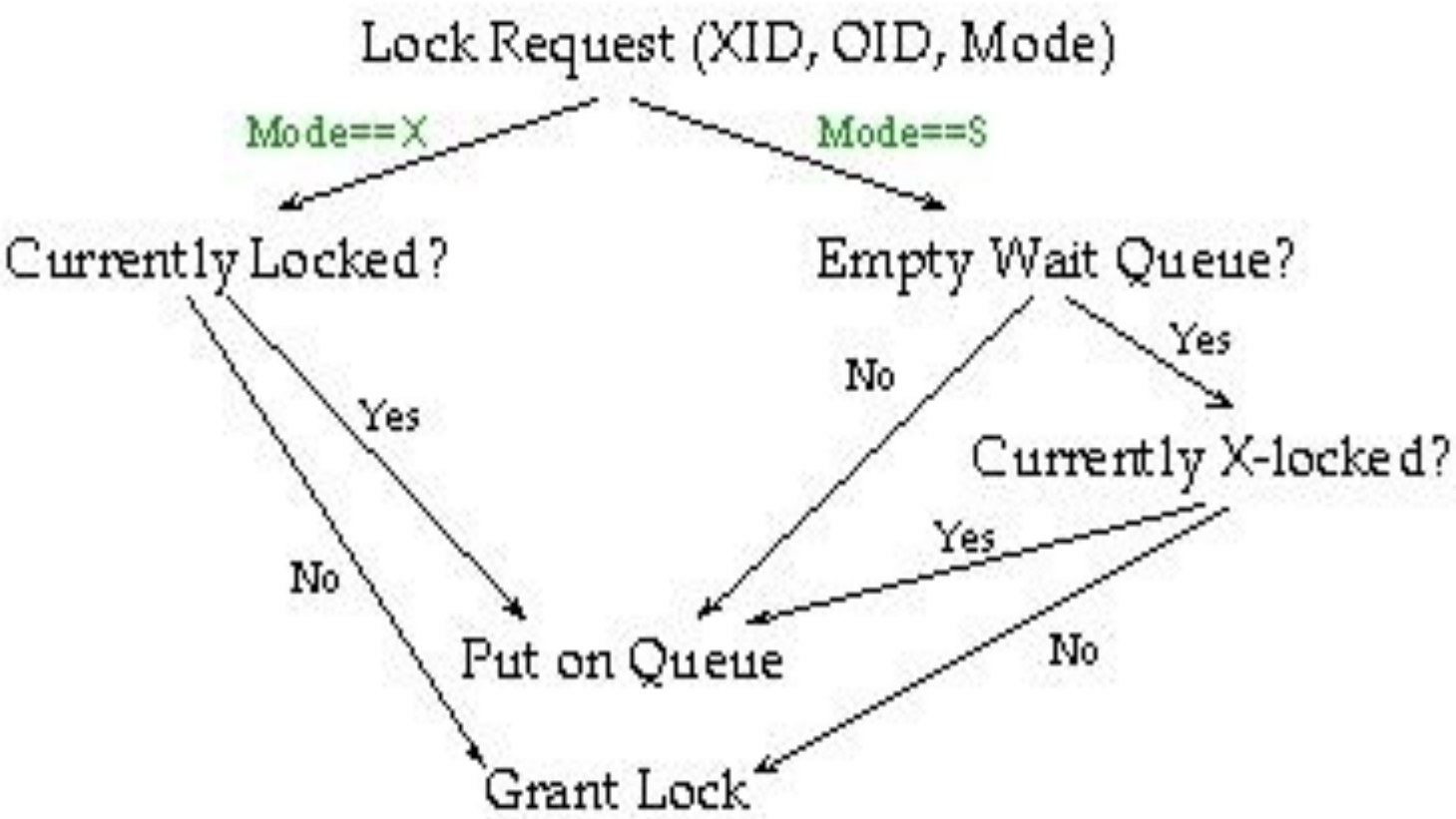
- Όταν μια δοσοληψία τελειώνει [επιτυχώς, ή μη], όλα τα κλειδώματα αποδεσμεύονται.
- Όσες δοσοληψίες περιμένουν στην ουρά αναμονής για κάθε κλείδωμα, ενεργοποιούνται και λαμβάνουν τα κλειδώματα με τη σειρά

- Π.χ., αν η ουρά είναι `S1[x]S2[x]X3[x]` για το x, όταν θα ενεργοποιηθεί η ουρά, η T1 και T2 θα πάρουν το S-lock και η ουρά θα γίνει `X3[x]`

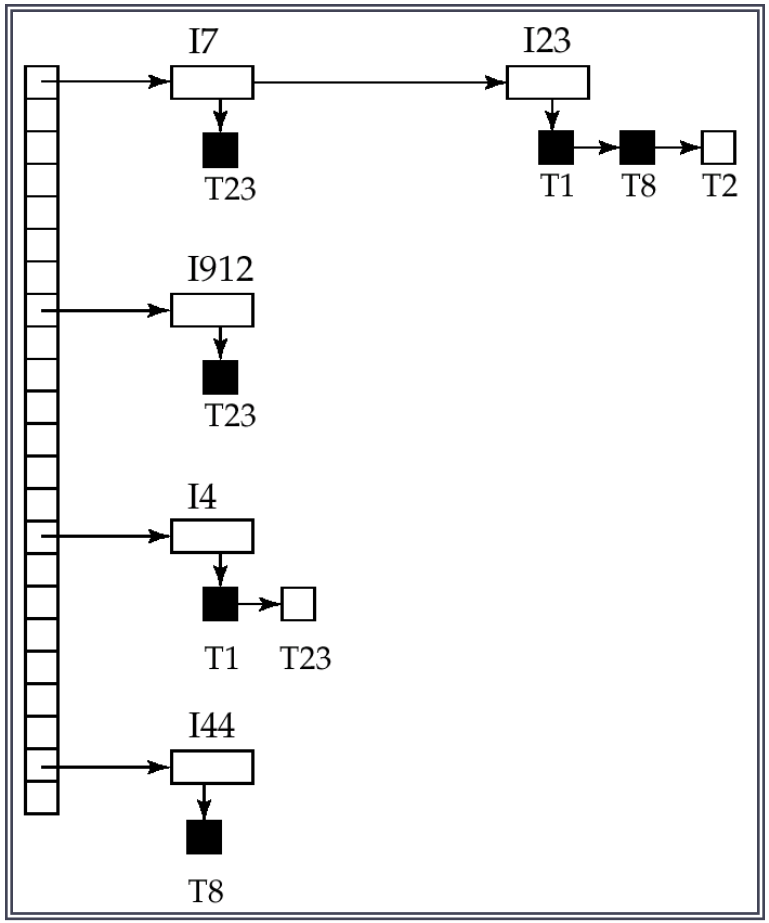
Υλοποίηση Κλειδώματος

- Ένας *διαχειριστής κλειδωμάτων* υλοποιείται σαν μία ξεχωριστή διαδικασία που επεξεργάζεται lock και unlock αιτήσεις
- Ο διαχειριστής απαντά σε μία αίτηση κλειδώματος στέλνοντας ένα μήνυμα αποδοχής κλειδώματος (ή ένα μήνυμα ανάκλησης σε περίπτωση αδιεξόδου)
- Ο διαχειριστής κλειδωμάτων διατηρεί ένα *πίνακα κλειδωμάτων* για να αποτυπώνει όλη την απαραίτητη πληροφορία
- Συνήθως ο πίνακας κλειδωμάτων υλοποιείται με ένα πίνακα κατακερματισμού στην κύρια μνήμη, ο οποίος δεικτοδοτείται από το όνομα του στοιχείου που έχει κλειδωθεί

Υλοποίηση Κλειδώματος



Πίνακας Κλειδωμάτων



- Τα μαύρα τετράγωνα είναι κλειδώματα που έχουν δοθεί ενώ τα άσπρα περιμένουν για μήνυμα αποδοχής.
- Μία νέα αίτηση εισάγεται στο τέλος της ουράς που ενδεχομένως να υπάρχει για το συγκεκριμένο στοιχείο και γίνεται αποδεκτή αν είναι συμβατή με όλα τα προηγούμενα κλειδώματα.
- Unlock αιτήσεις έχουν ως αποτέλεσμα την διαγραφή της αίτησης ενώ ελέγχονται οι επόμενες στην ουρά.
- Αν η δοσοληψία αποτύχει, όλες οι αιτήσεις της διαγράφονται

Λιμοκτονία -- starvation

- Έστω ότι η T1 έχει S-lock στο x
- Έστω ότι η T2 ζητά X-lock στο x ➔ θα μπει στην ουρά αναμονής `Q:X2[x]`
- Έστω ότι, ακολούθως, έρχεται η T3 και ζητά S-lock στο x. Θα το λάβει?

- ΟΧΙ! Αν ακολουθήσουμε αυτή την πολιτική, είναι πιθανό η T2 να περιμένει για πάντα...
- Η T3 θα μπει στην ουρά `Q:X2[x]S3[x]`

Αναβάθμιση κλειδώματος

- Μέχρι τώρα υποθέταμε ότι κάθε δοσοληψία γνωρίζει εκ προοιμίου όλα τα κλειδώματα που θα ζητήσει.
- Π.χ. `T:R(A)R(B)W(A)` θα κλειδώσει με **X** το A στην αρχή, λόγω του **W(A)** →

  $T:X_T(A)R(A)S_T(B)R(B)W(A)U(A)U(B)$.
- Τι γίνεται όμως αν δεν τα γνωρίζει?
- Η λύση έγκειται στην **αναβάθμιση κλειδώματος (αντίστοιχα, υποβάθμιση)**

Κανόνες αναβάθμισης

- Αναβαθμίσεις επιτρέπονται μόνο στη φάση αύξησης
- Υποβαθμίσεις επιτρέπονται μόνο στη φάση μείωσης
- Αν υπάρχει και άλλη δοσοληψία που έχει S-lock ένα αντικείμενο, και θες να αναβαθμίσεις από S- σε X-lock, πρέπει να περιμένεις…

Αναβάθμιση

Μπορείς

$\mathbf{S_{11}(\alpha 1)}$

T_{11}	T_{12}
read(a_1)	**read**(a_1)
read(a_2)	**read**(a_2)
...	**display**($a_1 + a_2$)
read(a_n)	
write(a_1)	

Upgrade από $\mathbf{S_{11}(\alpha 1)}$ σε $\mathbf{X_{11}(\alpha 1)}$ προϋποθέτει να κάνει UNLOCK η T_{12} το α1

Περιεχόμενα

- Κλειδώματα – 2 Phase Locking
- Πώς γίνεται στην πράξη?
- Αδιέξοδα

Αδιέξοδο

- Η κατάσταση κατά την οποία δύο δοσοληψίες Τ και Τ’ αναμένουν η μία την άλλη για την απελευθέρωση κάποιων κλειδωμάτων
- Εναλλακτικοί όροι: «Λειτουργική παύση» ή “deadlock”
- Προφανώς, ο παραπάνω ορισμός γενικεύεται για περισσότερες από δύο δοσοληψίες

Αδιέξοδο

- Θεωρήστε το παρακάτω χρονοπρόγραμμα

S1[x],R1[x],X2[y],W2[y],X2[x],W2[x],X1[y],W1[y]

- Η Τ1 έχει S-lock στο x
- Η Τ2 έχει X-lock στο y. Όταν ζητά X-lock στο x, υποχρεωτικά περιμένει την Τ1
- Η Τ1 πάει να ζητήσει X-lock στο y, οπότε υποχρεωτικά περιμένει την Τ2 …

Γράφος αναμονής (blocking graph)

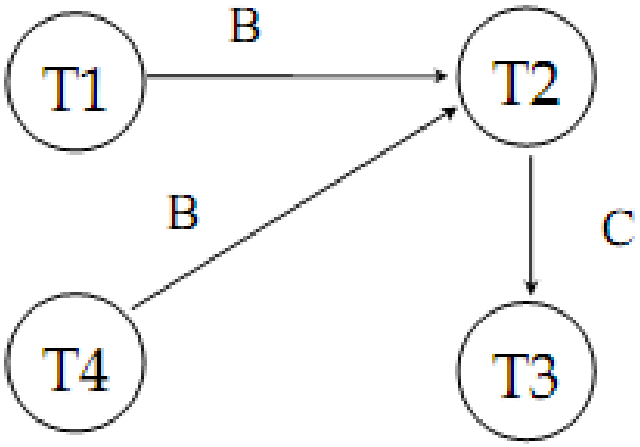
- Για κάθε δοσοληψία και ένας κόμβος
- Μία κατευθυνόμενη ακμή από την δοσοληψία T1 στην δοσοληψία T2, αν η T1 **περιμένει** την T2 να απελευθερώσει κάποιο κλείδωμα

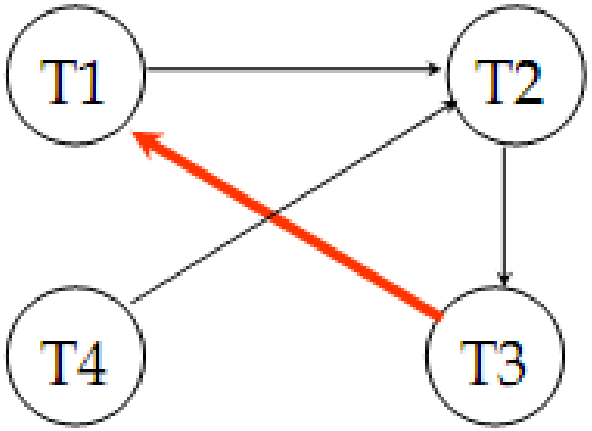
- Επίσης γνωστός και ως γράφος **waits-for**

ΠΡΟΣΟΧΗ: Δεν είναι ίδιος με το γράφο σειριοποιησιμότητας!!!

Παράδειγμα

T1: S-Lock(A), R(A), S-Lock(B)
T2: X-Lock(B),W(B) X-Lock(C)
T3: S-Lock(C), R(C) X-Lock(A)
T4: X-Lock(B)

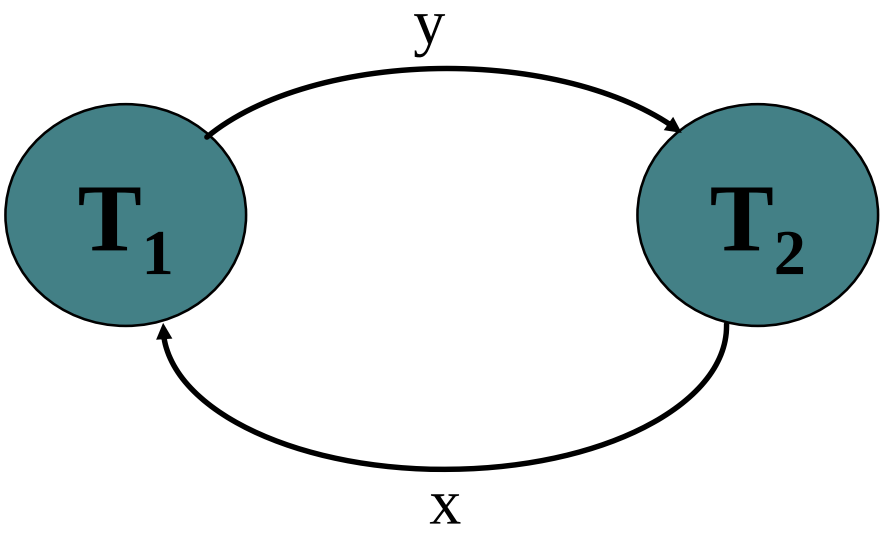




Αδιέξοδο

- Θεωρήστε το παρακάτω χρονοπρόγραμμα

S1[x],R1[x],X2[y],W2[y],X2[x],W2[x],X1[y],W1[y]



Θεώρημα

- **Θεώρημα**: Υπάρχει αδιέξοδο αν και μόνο αν ο γράφος αναμονής έχει κύκλο

- Στην πράξη το σύστημα ελέγχει τον γράφο αναμονής περιοδικά
- Εναλλακτικά, αντί για γράφο, μπορεί να μετρά πόση ώρα μια δοσοληψία αναμένει για ένα κλείδωμα και να την τερματίζει αν περάσει κάποιο όριο...

Ανίχνευση ή αποτροπή?

- Αντί να περιμένουμε να συμβεί το αδιέξοδο, μπορούμε να το αποτρέψουμε προληπτικά

- **Συντηρητικός 2PL**: μια δοσοληψία αποκτά όλα τα κλειδώματα που χρειάζεται στο ξεκίνημα της. Αν δεν μπορεί να τα πάρει ΟΛΑ, δεν ξεκινά, αλλά αναμένει!

Εν γένει, προτιμώνται τα αδιέξοδα…

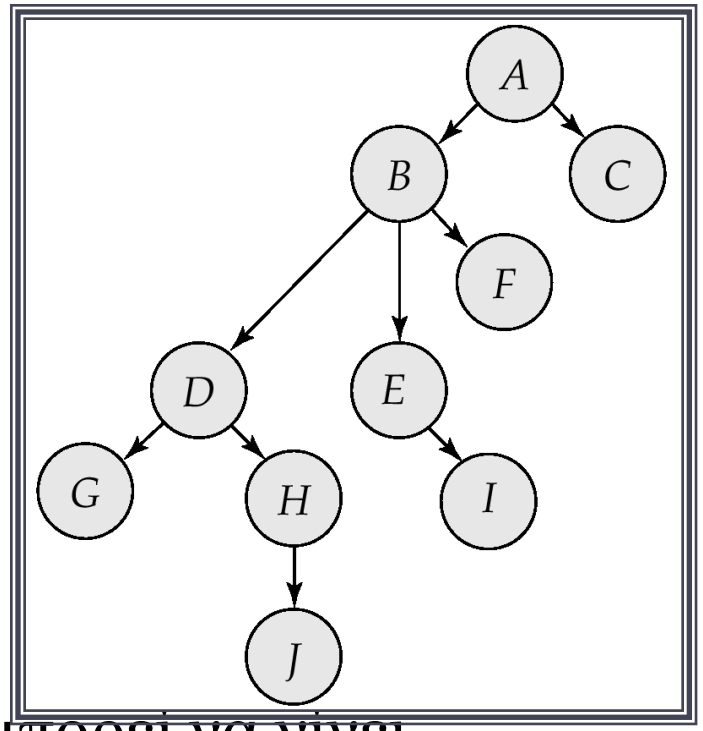
Χρονικά Όρια

- Επιλογή Θύματος (victim selection)
  - Επιλογή δοσοληψιών που θα ακυρωθούν
  - Αποφεύγει δοσοληψίες που έχουν κάνει πολλές αλλαγές
- Χρονικά Όρια (timeouts)
  - Αν μια δοσοληψία εκτελείται για μεγάλο χρονικό διάστημα τότε την ακυρώνει
  - Ασχέτως με αν στη πραγματικότητα υπάρχει αδιέξοδο
- Λιμοκτονία (starvation)
  - Μια δοσοληψία δεν μπορεί να περιμένει για απεριόριστο χρονικό διάστημα ενώ άλλες συνεχίζουν κανονικά

Πρωτόκολλα Βασισμένα σε Γράφους

- Εναλλακτικά του 2PL
- Θεωρούν μια μερική διάταξη→ στο σύνολο**D** = $\{d_1, d_2, \ldots, d_h\}$ όλων των δεδομένων.
  - Αν $d_i \rightarrow d_j$ τότε όποια συναλλαγή που έχει πρόσβαση στα d_i και d_j πρέπει να προσπελάσει το d_i πριν από το d_j.
  - Υπονοεί ότι το **D** μπορεί να θεωρηθεί ως ένας κατευθυνόμενος άκυκλος γράφος (γράφημα βάσης δεδομένων)

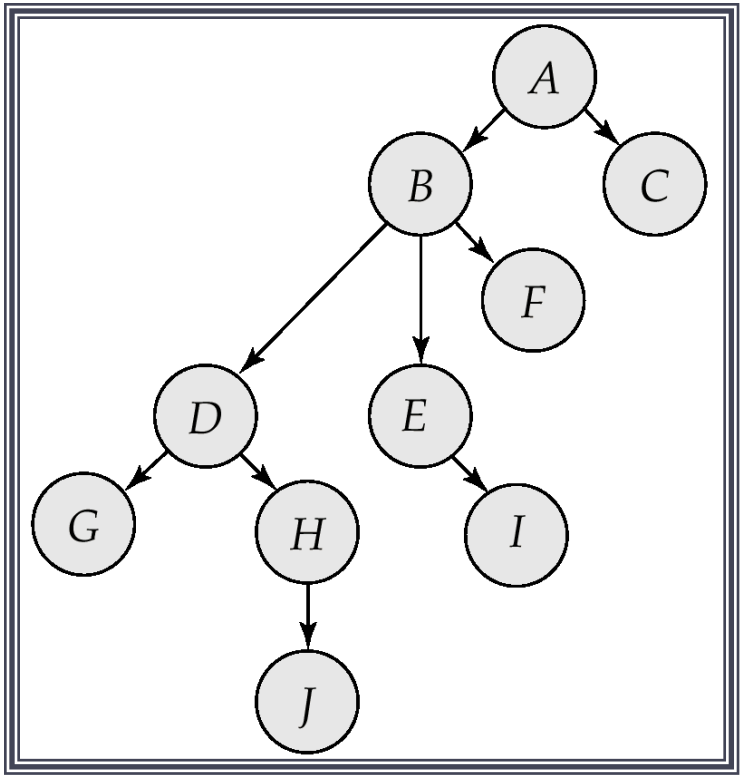
Πρωτόκολλο Δέντρου



- Επιτρέπονται μόνο X-locks.
- Το πρώτο κλείδωμα από την T_i μπορεί να γίνει οπουδήποτε.
  - Στη συνέχεια, ένα δεδομένο Q μπορεί να κλειδωθεί από την T_i μόνο αν ο γονέας του Q είναι ήδη κλειδωμένος από την T_i.
- Τα δεδομένα μπορούν να ξεκλειδωθούν οποιαδήποτε ώρα

Σειριοποιήσιμο Χρονοπρόγραμμα με Πρωτόκολλο Δέντρου

T_{10}	T_{11}	T_{12}	T_{13}
lock-X(B)			
	lock-X(D)		
	lock-X(H)		
	unlock(D)		
lock-X(E)			
lock-X(D)			
unlock(B)			
unlock(E)			
		lock-X(B)	
		lock-X(E)	
	unlock(H)		
lock-X(G)			
unlock(D)			
			lock-X(D)
			lock-X(H)
			unlock(D)
			unlock(H)
		unlock(E)	
		unlock(B)	
unlock (G)			



Πρωτόκολλα Βασισμένα σε Γράφους

- Το πρωτόκολλο δέντρου εξασφαλίζει σειριοποιησιμότητα συγκρούσεων και απαλλαγή από αδιέξοδα
- Το ξεκλείδωμα μπορεί να συμβεί νωρίτερα από το 2PL
  - Μικρότεροι χρόνοι αναμονής, αύξηση της συνδρομικότητας
  - Η ακύρωση μιας συναλλαγής μπορεί να οδηγήσει σε διαδιδόμενες ανακλήσεις.)
- Όμως, μια συναλλαγή πρέπει να κλειδώσει δεδομένα που δεν χρειάζεται
- Τα χρονοπρογράμματα που δεν υλοποιούνται με 2PL γίνονται με πρωτόκολλα δέντρου και το αντίστροφο.

Διάταξη Χρονοσημάτων (TimeStamps)

- Το χρονόσημα δημιουργείται από το ΣΔΒΔ και προσδιορίζει μοναδικά μια δοσοληψία
- Ιδέα: διάταξη των δοσοληψιών με βάση το χρονόσημα τους (δηλαδή, χρονοπρόγραμμα ισοδύναμο με σειριακό στο οποίο οι δοσοληψίες εμφανίζονται διατεταγμένες με βάση τις τιμές των χρονοσημάτων)
- ⇒ άρα η σειρά προσπέλασης στα δεδομένα πρέπει να μη παραβιάζει τη σειριοποιησιμότητα

Διάταξη Χρονοσημάτων

- Κάθε δοσοληψία αποκτά ένα χρονόσημα όταν εισέρχεται στο σύστημα. Αν μία παλιά δοσοληψία T_i έχει χρονόσημα TS(T_i), μία νέα δοσοληψία T_j αποκτά το χρονόσημα TS(T_j) έτσι ώστε TS(T_i) <TS(T_j).
- Το χρονοπρόγραμμα που προκύπτει είναι σειριοποιήσιμο. Η σειρά καθορίζεται από τα χρονοσήματα.
- Για κάθε στοιχείο Q στη βάση διατηρούνται δύο τιμές:
  - **W-timestamp**(Q) είναι το μεγαλύτερο χρονόσημα μίας δοσοληψίας που εκτέλεσε επιτυχώς μία πράξη εγγραφής στο Q.
  - **R-timestamp**(Q) είναι το μεγαλύτερο χρονόσημα μίας δοσοληψίας που εκτέλεσε επιτυχώς μία πράξη ανάγνωσης στο Q.

Διάταξη Χρονοσημάτων (1/2)

- Το πρωτόκολλο διάταξης χρονοσημάτων εγγυάται ότι οι αντίθετες **read** και **write** πράξεις εκτελούνται με τη σειρά που ορίζεται από τα χρονοσήματα.
- Υποθέστε ότι μία δοσοληψία T_i κάνει μία πράξη **read**(Q)
  1. Αν TS(T_i) < **W**-timestamp(Q), τότε η T_i θέλει να διαβάσει μία τιμή του Q που έχει ήδη γραφτεί.
     - Η πράξη **read** ακυρώνεται, και η T_i αναιρείται.
  2. Αν TS(T_i)≥ **W**-timestamp(Q), τότε η πράξη **read** εκτελείται, και η R-timestamp(Q) γίνεται ίσο με το μέγιστο μεταξύ του R-timestamp(Q) και TS(T_i).

Διάταξη Χρονοσήμων (2/2)

- Υποθέστε ότι μία δοσοληψία T_i κάνει μία πράξη **write**(Q).
  1. Αν TS(T_i) < R-timestamp(Q), τότε η τιμή του Q που η T_i δημιουργεί ήταν αναγκαία πριν (κάποια προηγούμενη συναλλαγή βασίζεται στην παλιά τιμή του Q).
     - Η πράξη **write** ακυρώνεται, και η T_i αναιρείται.
  2. Αν TS(T_i) < W-timestamp(Q), τότε η T_i προσπαθεί να γράψει μία ξεπερασμένη χρονικά τιμή.
     - Η πράξη **write** ακυρώνεται, και η T_i αναιρείται.
  3. Διαφορετικά, η πράξη **write** εκτελείται, και W-timestamp(Q) γίνεται TS(T_i).

Παράδειγμα 1

T1(10)

RTS(X): 10
WTS(X): 10
RTS(Y): 0
WTS(Y): 0

1. A1 = Read(X)
2. A1 = A1 – k
3. Write(X, A1)

T2(20)

1. A1 = Read(X)
2. A1 = A1* 1.01
3. Write(X, A1)

RTS(X): 20
WTS(X): 20
RTS(Y): 0
WTS(Y): 0

RTS(X): 20
WTS(X): 20
RTS(Y): 10
WTS(Y): 10

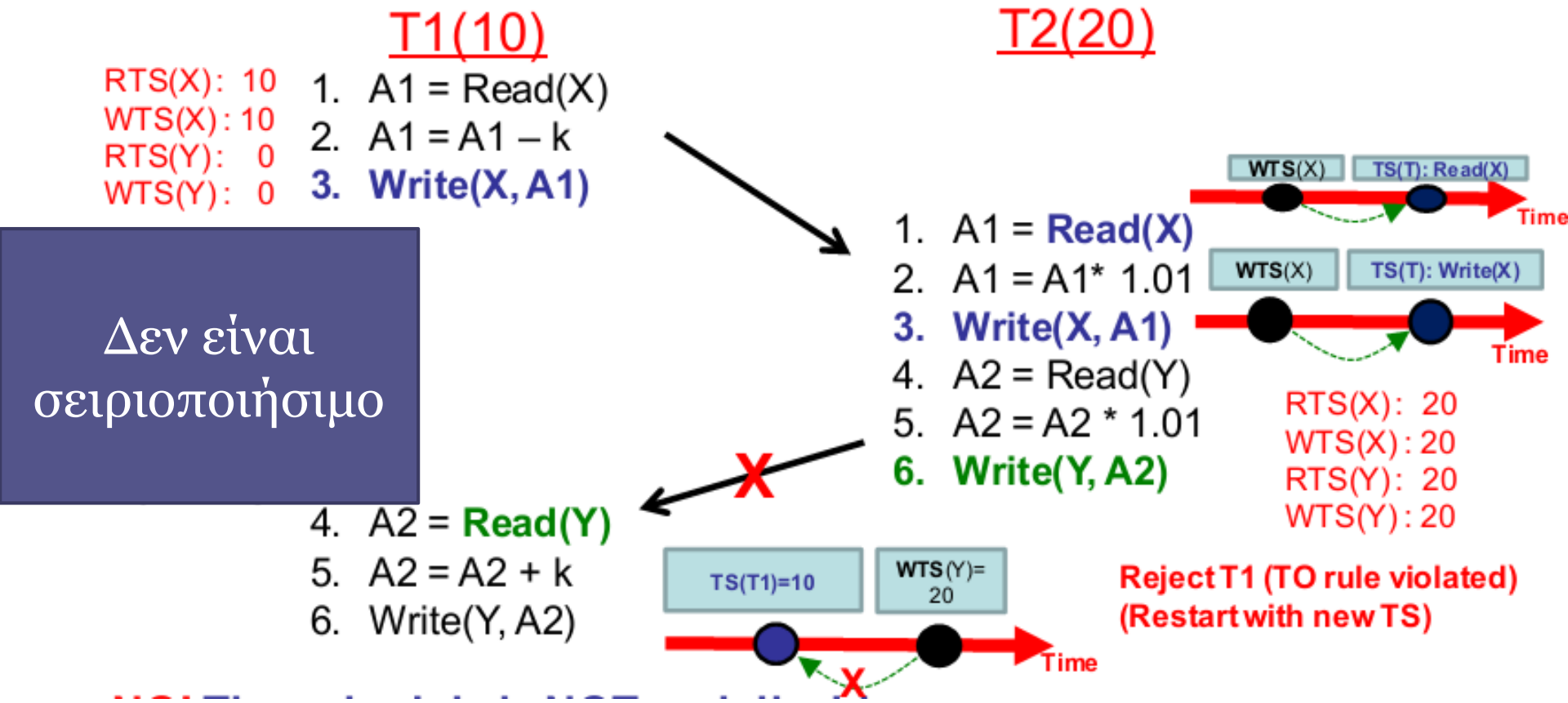
4. A2 = Read(Y)
5. A2 = A2 + k
6. Write(Y, A2)

4. A2 = Read(Y)
5. A2 = A2 * 1.01
6. Write(Y, A2)

RTS(X): 20
WTS(X): 20
RTS(Y): 20
WTS(Y): 20

Είναι Σειριοποιήσιμο

Παράδειγμα 2



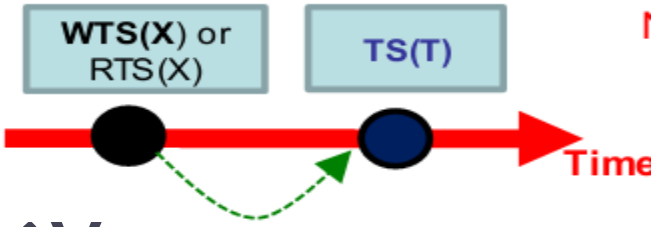
Δεν είναι σειριοποιήσιμο

Παράδειγμα 3

T_{14}	T_{15}
read (B)	
	read (B)
	$B := B - 50$
	write (B)
read (A)	
	read (A)
display $(A + B)$	
	$A := A + 50$
	write (A)
	display $(A + B)$

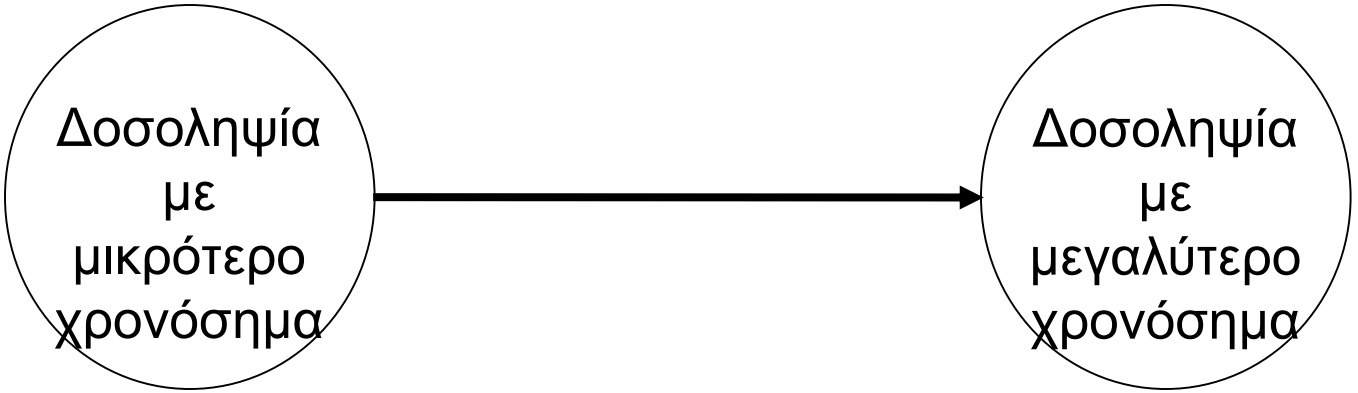
Παράδειγμα 4

T_1	T_2	T_3	T_4	T_5
				read(X)
read(Y)	read(Y)			
		write(Y)		
		write(Z)		
				read(Z)
	write(X)			
	abort			
read(X)				
		write(Z)		
		abort		
				write(Y)
				write(Z)



Ορθότητα Χρονοσήμων

- Το πρωτόκολλο εγγυάται σειριοποιησιμότητα αφού όλες οι ακμές στο γράφο προήγησης είναι τύπου:



  Επομένως, δεν υπάρχει κύκλος στο γράφημα
- Το πρωτόκολλο δεν επιτρέπει αδιέξοδα.
- Όμως, το χρονοπρόγραμμα μπορεί να έχει διαδιδόμενες ανακλήσεις, και μάλιστα μπορεί να μην είναι ανακάμψιμο.

Ανάκαμψη και Αποφυγή Διαδιδόμενων Ανακλήσεων

- Πρόβλημα με τα χρονοσημματα:
  - Υποθέστε ότι η T_i ακυρώνεται, αλλά η T_j έχει διαβάσει τιμή που γράφτηκε από T_i
  - Η T_j πρέπει να ακυρωθεί; Αν η T_j είχε επικυρωθεί νωρίτερα τότε το χρονοπρόγραμμα θα ήταν μη ανακάμψιμο.
  - Οδηγούμαστε σε διαδιδόμενες ανακλήσεις
- Λύση 1:
  - Μία δοσοληψία δομείται με τρόπο ώστε οι εγγραφές να γίνονται στο τέλος της
  - Το σύνολο των εγγραφών εκτελείται ατομικά
  - Μία δοσοληψία που αναιρείται, εκκινεί με νέο χρονόσημο
- Λύση 2: αναμονή για επικύρωση δεδομένων πριν την ανάγνωσή τους