

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 κατάσταση του**, σε σχέση με πιθανές πράξεις που μπορούν να εφαρμοσθούν σε αυτό.

Read@T1

Written@T2

EMP

ID	Name	
12	XX	
13	YY	
14	ZZ	

2 Phase Locking - 2PL

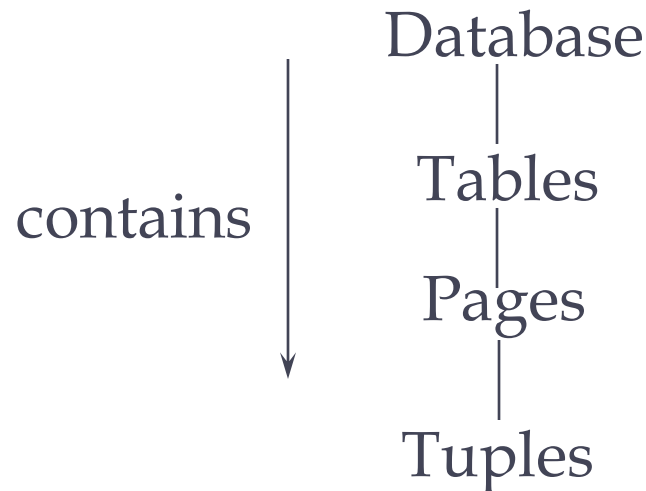
- Ο βασικός αλγόριθμος διαχείρισης του ταυτοχρονισμού δοσοληψιών είναι ο «**Αλγόριθμος Κλειδώματος Δύο Φάσεων**» -- στην αγγλική “2 Phase Locking” ή **2PL**

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**



Πρωτόλειος αλγόριθμος κλειδώματος ≠ 2PL (συνεχίζεται...)

- Κάθε ενέργεια διαβάσματος/γραφίσματος 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 \neq 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)

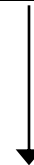
lock-S(A)

read(A)

display(A+B)

unlock(A).

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



X₇(B) ; $R_7(B)$; $W_7(B)$; **U₇(B)** ; **S₇(A)** ; $R_7(A)$ **U₇(A)**

Παράδειγμα

T_7
lock-X(B)
→ read(B)
→ $B := B - 50$
→ write(B)
unlock(B)

lock-S(A)
→ read(A)
 $\text{display}(A+B)$
unlock(A)

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 κλείδωμα για το ίδιο αντικείμενο.
- Ο πίνακας συγκρούσεων πρέπει να σας θυμίζει κάτι...
- Άρνηση \Rightarrow «περίμενε μέχρι να γίνει unlock»

		Lock held	
Lock requested	T1 \ T2	Read	Write
	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

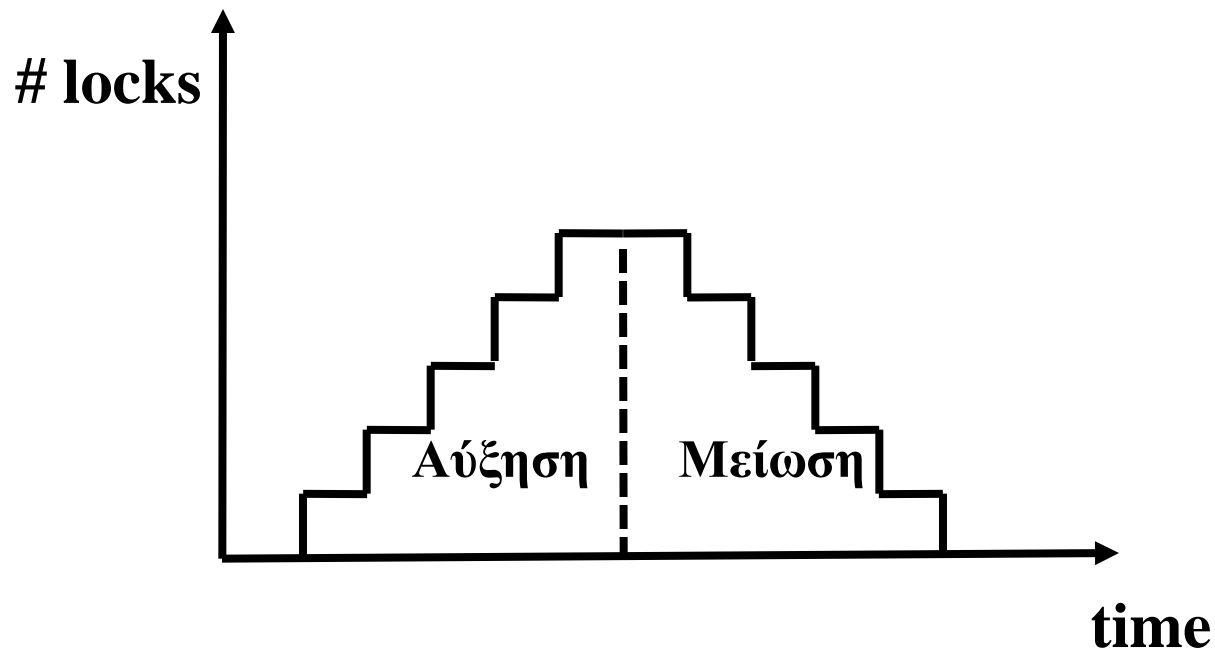
Η μία δοσοληψία έχει κλειδώσει το αντικείμενο που χρειάζεται η άλλη. Σταματούν και οι δύο, αναμένοντας η μία την άλλη ...

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

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

ΕΠΙΤΡΕΠΤΟ

 T_{10}

lock-S(A)
 read(A)
 lock-S(B)
 unlock(A)
 unlock(B)
 display(A + B)

ΕΠΙΤΡΕΠΤΟ

 T_7

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

ΠΑΡΑΒΑΣΗ

 T_8

lock-S(A)
 read(A)
 unlock(A)
 lock-S(B)
 unlock(B)
 display(A + B)

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:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

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:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

	x	y	z
T1		X	
T2	S		
T3			

$X_1[y]; R_1[y]; S_2[x]; R_2[x];$

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

Έχω ήδη το
lock...

$X_1[y]; R_1[y]; S_2[x]; R_2[x]; W_1[y];$

	x	y	z
T1		X	
T2	S		
T3			

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 T3 θέλει το y

OXI!!

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

	x	y	z
T1		X	
T2	S		
T3		W	

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

Και τι
κάνω?

	x	y	z
T1			X
T2	S		
T3		W	

$X_1[y]; R_1[y]; S_2[x]; R_2[x]; W_1[y]; X_1[z]; U_1[y];$

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

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

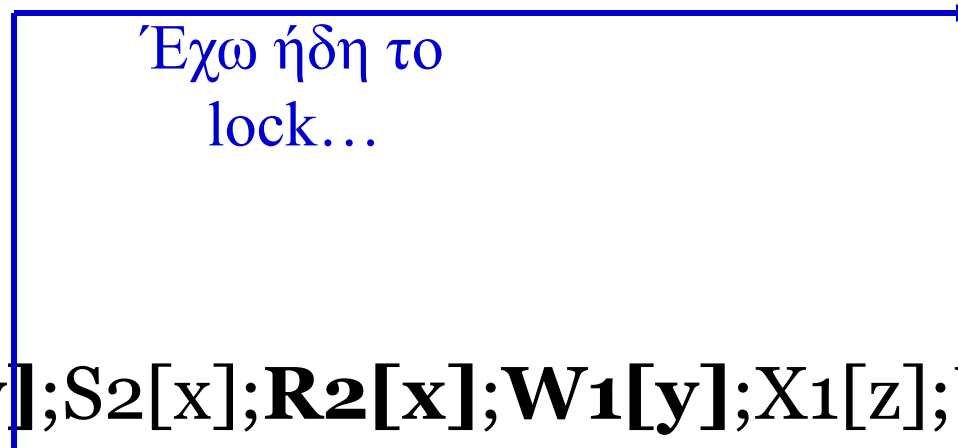
	x	y	z
T1			X
T2	S		
T3		X	

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	

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 AMEΣA

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

	x	y	z
T1			
T2	S		S
T3		X	

$X_1[y]; R_1[y]; S_2[x]; R_2[x]; W_1[y]; X_1[z]; U_1[y]; X_3[y]; W_3[y]; W_1[z]; U_1[z]; S_2[z]; R_2[z];$

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

S2:

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

ΔΕΝ υπάρχει
σύγκρουση για S-
locks!!

	x	y	z
T1			
T2	S		S
T3		X	S

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];

... val ...

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

$X_1[y]; R_1[y]; S_2[x]; R_2[x]; W_1[y]; X_1[z]; U_1[y]; X_3[y]; W_3[y]; W_1[z]; U_1[z]; S_2[z]; R_2[z]; S_3[z]; R_3[z]; U_{123}[all] \dots$

Μπορούσα να είχα κάνει ?

S2:

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

$U_3[y]$
 $U_2[x]$
 $X_1[y]; R_1[y]; S_2[x]; R_2[x]; W_1[y]; X_1[z]; U_1[y]; X_3[y]; W_3[y]; W_1[z]; U_1[z]; S_2[z]; R_2[z]; S_3[z]; R_3[z]; U_{123}[all] \dots$

Θεώρημα

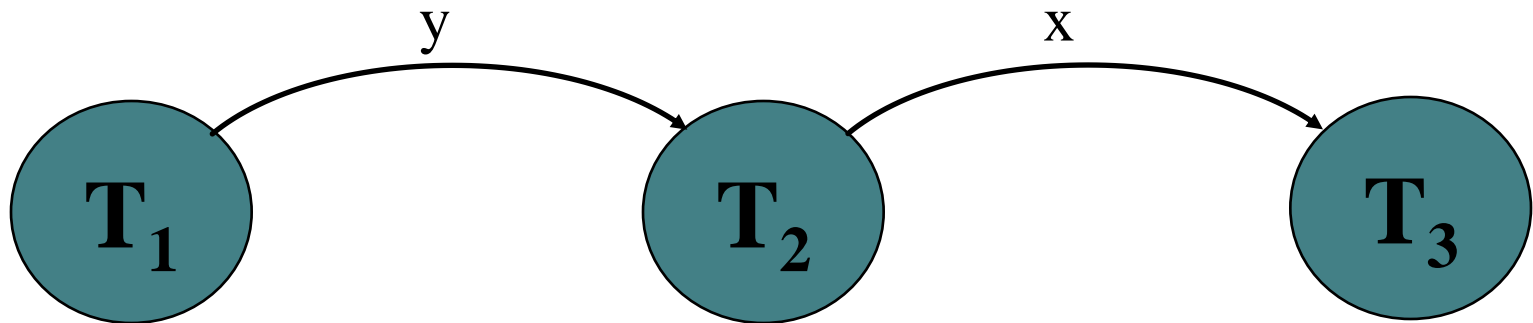
- Όλα τα χρονοπρογράμματα που προκύπτουν από ένα διαχειριστή κλειδαριών 2PL, έχουν **ακυκλικό γράφο σειριοποίησης**
- Το αντίστροφο ΔΕΝ ισχύει.
- Λήμμα: αν δεν έχει ακυκλικό γράφο σειριοποίησης \rightarrow δεν είναι 2PL.

Προκύπτει το παρακάτω
χρονοπρόγραμμα από αλγόριθμο 2PL ?

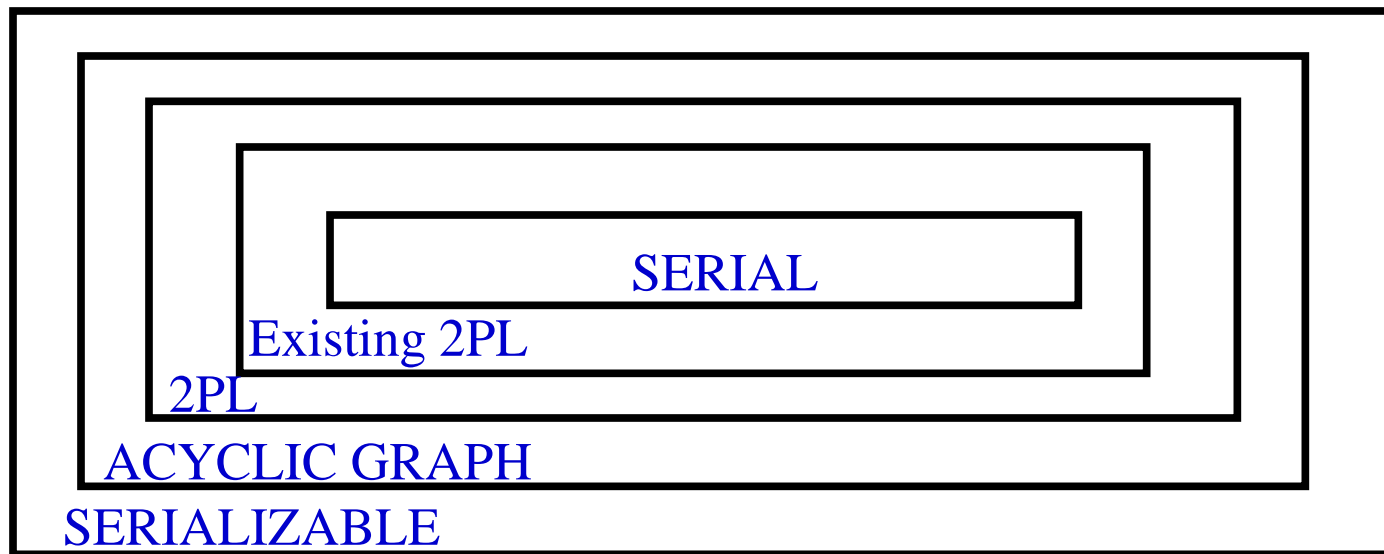
S₃: $W_2[x]; W_3[x]; W_1[y]; W_2[y]$.

Μπορώ να το σειριοποιήσω?

- **S₃**: $W_2[x]; W_3[x]; W_1[y]; W_2[y]$.

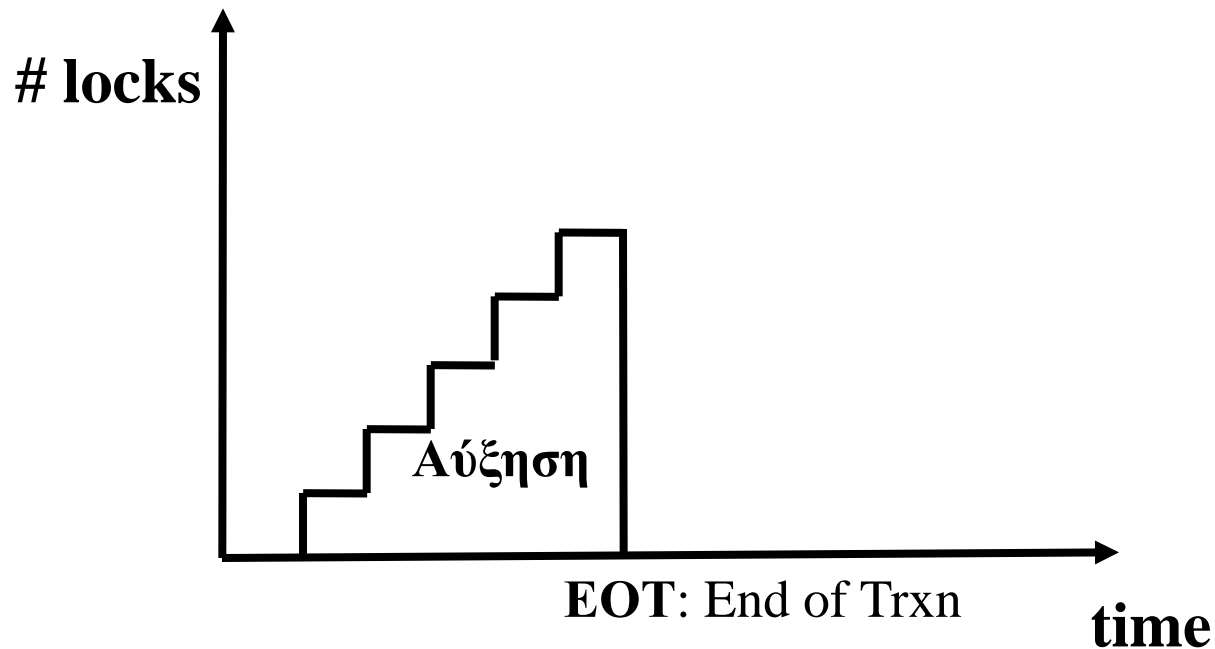


Venn διάγραμμα



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?

$R_1[y]; R_2[x]; W_1[y]; W_3[y]; W_1[z]; R_2[z]; R_3[z].$

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.

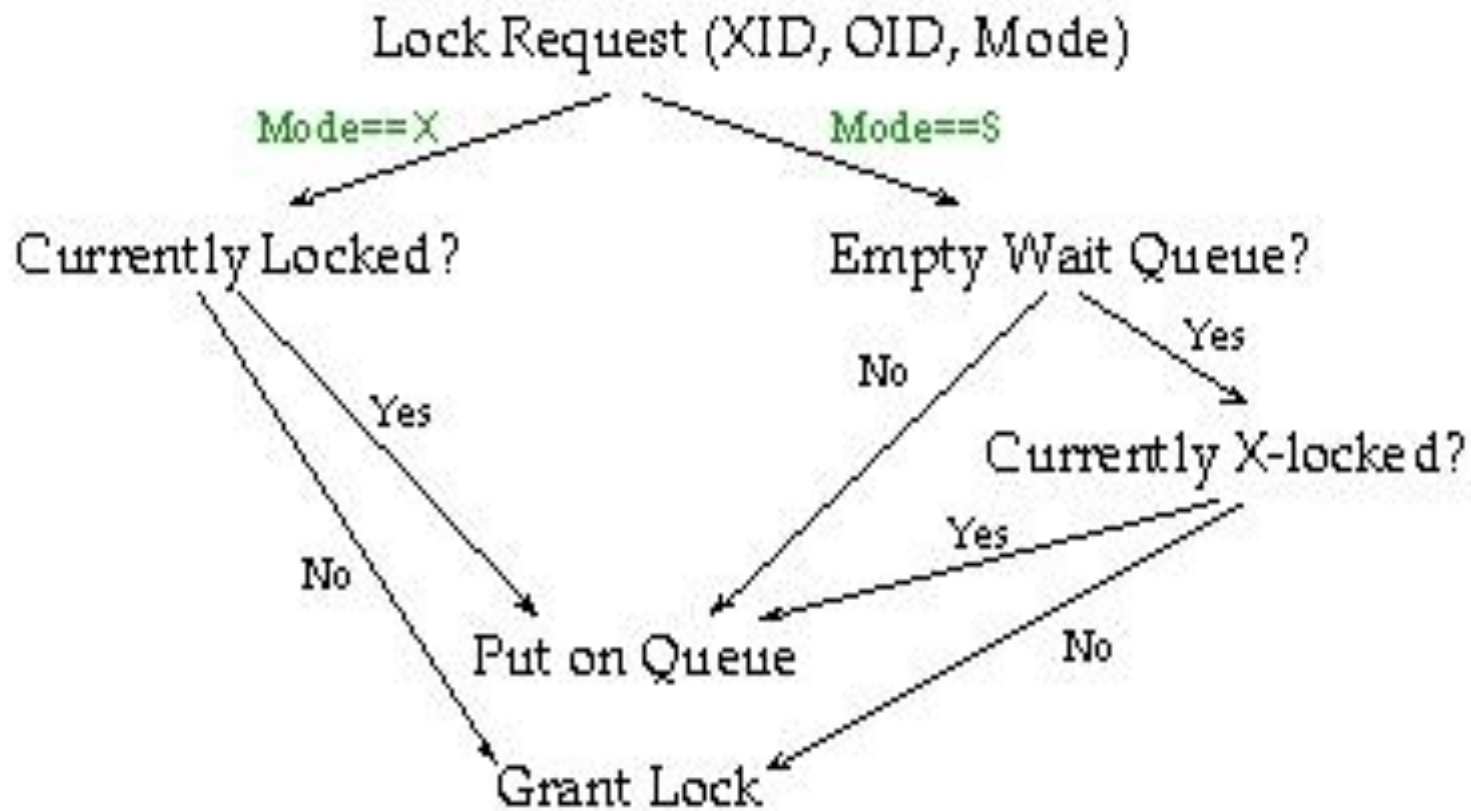
Πώς γίνεται στην πράξη?

- 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 $X3[x]$

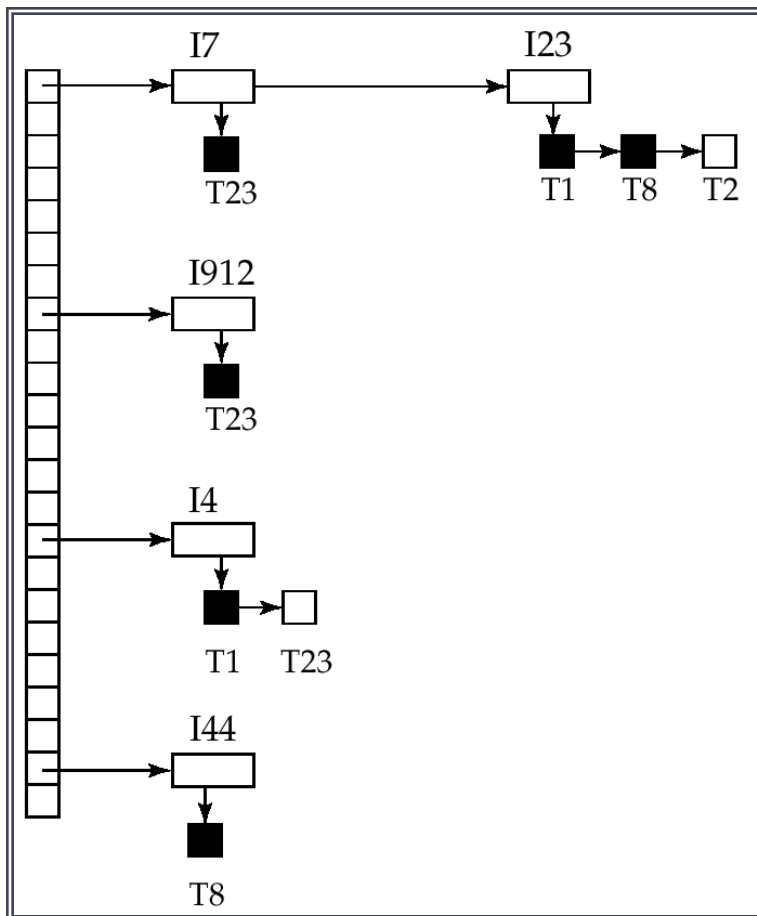
Υλοποίηση Κλειδώματος

- 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 έναν πίνακα κατακερματισμού στην κύρια μνήμη, ο οποίος δεικτοδοτείται από το όνομα του στοιχείου που έχει κλειδωθεί

Υλοποίηση Κλειδώματος



Πίνακας Κλειδωμάτων



- Τα μαύρα ορθογώνια 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

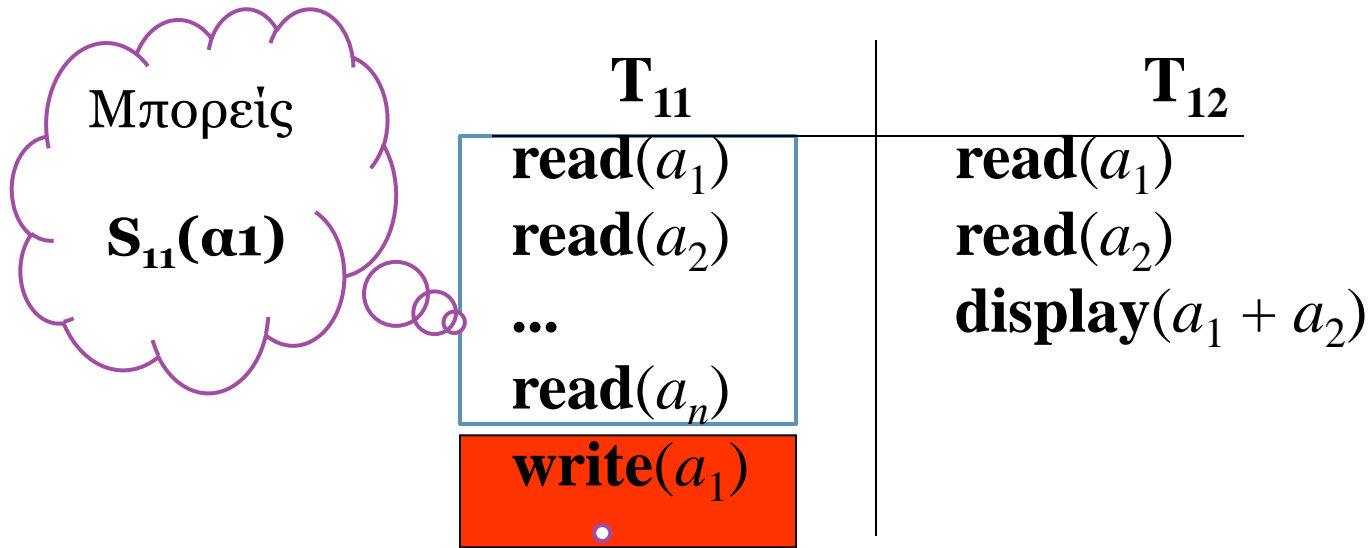
- Μέχρι τώρα υποθέταμε ότι κάθε δοσοληψία γνωρίζει εκ προοιμίου όλα τα κλειδώματα που θα ζητήσει.
- Π.χ. $T : \mathbf{R}(\mathbf{A}) \mathbf{R}(\mathbf{B}) \mathbf{W}(\mathbf{A})$ θα κλειδώσει με \mathbf{X} το A στην αρχή, λόγω του $\mathbf{W}(\mathbf{A}) \rightarrow$

$$T : \mathbf{X}_T(A) \mathbf{R}(\mathbf{A}) S_T(B) \mathbf{R}(\mathbf{B}) \mathbf{W}(\mathbf{A}) U(A) U(B) .$$
- 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



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

Περιεχόμενα

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

Αδιέξοδο

- Η κατάσταση κατά την οποία δύο δοσοληψίες T και T' 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 T1 έχει S-lock στο x
- Η T2 έχει X-lock στο y. Όταν ζητά X-lock στο x, υποχρεωτικά περιμένει την T1
- Η T1 πάει να ζητήσει X-lock στο y, οπότε υποχρεωτικά περιμένει την T2 ...

Γράφος αναμονής (blocking graph)

- Για κάθε δοσοληψία και ένας κόμβος
- Μία κατευθυνόμενη ακμή από την δοσοληψία T_1 στην δοσοληψία T_2 , αν η **T_1 περιμένει την T_2** 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),

T2: X-Lock(B), W(B)

T3:

T4:

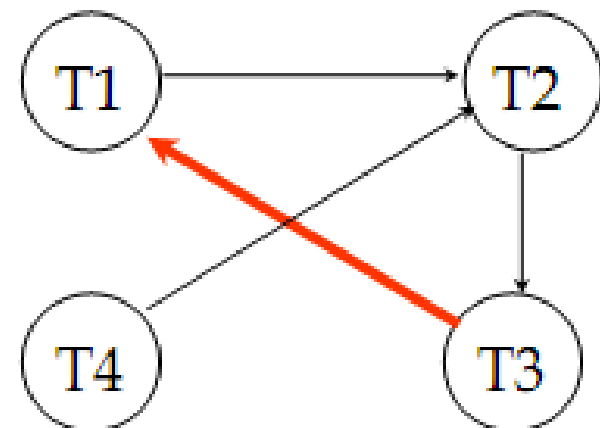
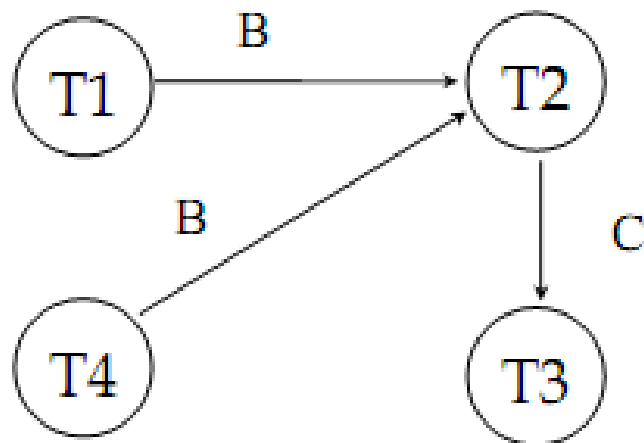
S-Lock(B)

X-Lock(C)

S-Lock(C), R(C)

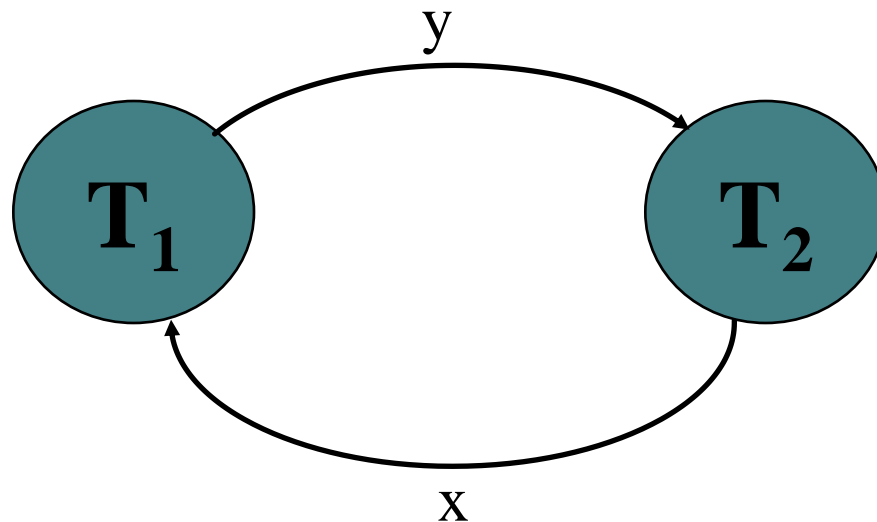
X-Lock(A)

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 κλειδιά που χρειάζεται στο ξεκίνημα της. Αν δεν μπορεί να τα πάρει **ΟΛΑ**, δεν ξεκινά, αλλά αναμένει!

Εν γένει, προτιμώνται τα αδιέξοδα...

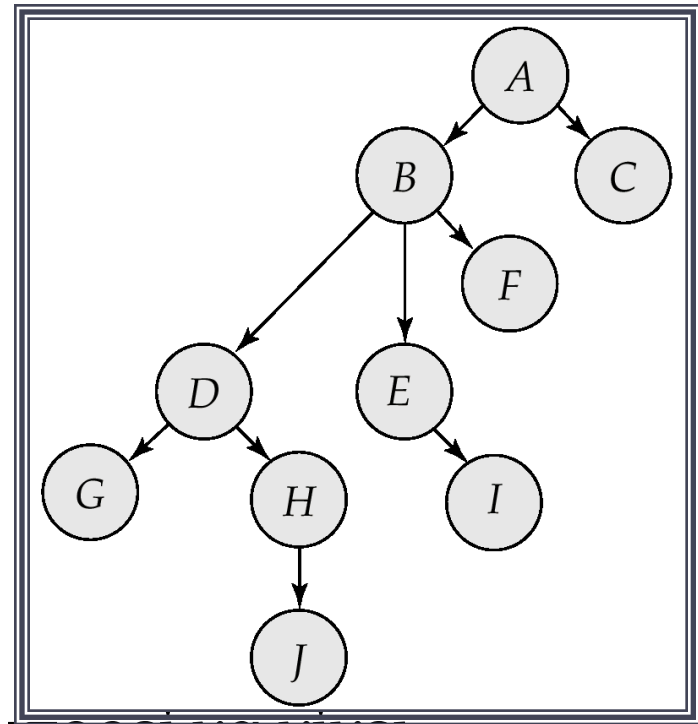
Χρονικά Όρια

- 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 προχωρήσει για απεριόριστο χρονικό διάστημα ενώ άλλες συνεχίζουν κανονικά

Πρωτόκολλα Βασισμένα σε Γράφους

- Εναλλακτικά του 2PL
- Θεωρούν μια μερική διάταξη \rightarrow στο σύνολο $\mathbf{D} = \{d_1, d_2, \dots, d_h\}$ 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 \mathbf{D} μπορεί να θεωρηθεί ως ένας κατευθυνόμενος άκυκλος γράφος (γράφημα βάσης δεδομένων)

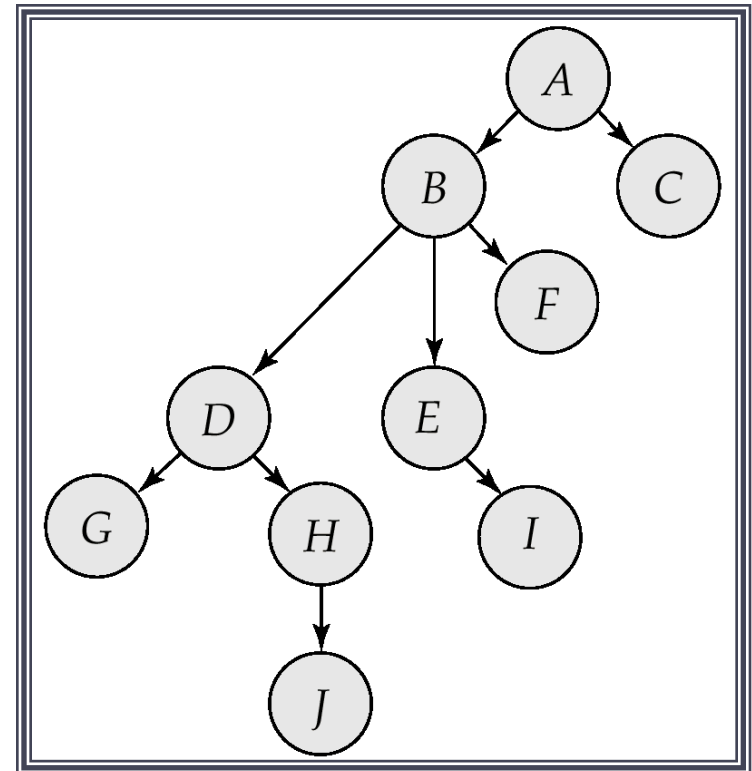
Πρωτόκολλο Δέντρου



- 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)	
lock-X(G) unlock(D)	unlock(H)		
		unlock(E) unlock(B)	
unlock (G)			lock-X(D) lock-X(H) unlock(D) unlock(H)



Πρωτόκολλα Βασισμένα σε Γράφους

- Το πρωτόκολλο δέντρου εξασφαλίζει σειριοποιησιμότητα αντιθέσεων 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)
- \Rightarrow άρα η σειρά προσπέλασης στα δεδομένα πρέπει να μη παραβιάζει τη σειριοποιησιμότητα

Διάταξη Χρονοσήμων

- 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 χρονόσημα.
- 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 Χρονοσήμων (1/2)

- Το πρωτόκολλο διάταξης χρονοσήμων εγγυάται ότι οι αντίθετες **read** και **write** πράξεις εκτελούνται με τη σειρά που ορίζεται από τα χρονόσημα.
- Υποθέστε ότι μία δοσοληψία T_i κάνει μία πράξη **read**(Q)
 1. Αν $TS(T_i) \leq \mathbf{W}\text{-timestamp}(Q)$, 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 $TS(T_i) \geq \mathbf{W}\text{-timestamp}(Q)$, τότε η πράξη **read** εκτελείται, και η $\mathbf{R}\text{-timestamp}(Q)$ γίνεται ίσο με το μέγιστο μεταξύ του $\mathbf{R}\text{-timestamp}(Q)$ και $TS(T_i)$.

Διάταξη Χρονοσήμων (2/2)

- Υποθέστε ότι μία δοσοληψία T_i κάνει μία πράξη **write**(Q).
 1. Αν $TS(T_i) < R\text{-timestamp}(Q)$, 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 $TS(T_i) < W\text{-timestamp}(Q)$, 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 $W\text{-timestamp}(Q)$ γίνεται $TS(T_i)$.

Παράδειγμα

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 2

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