



---

HY360

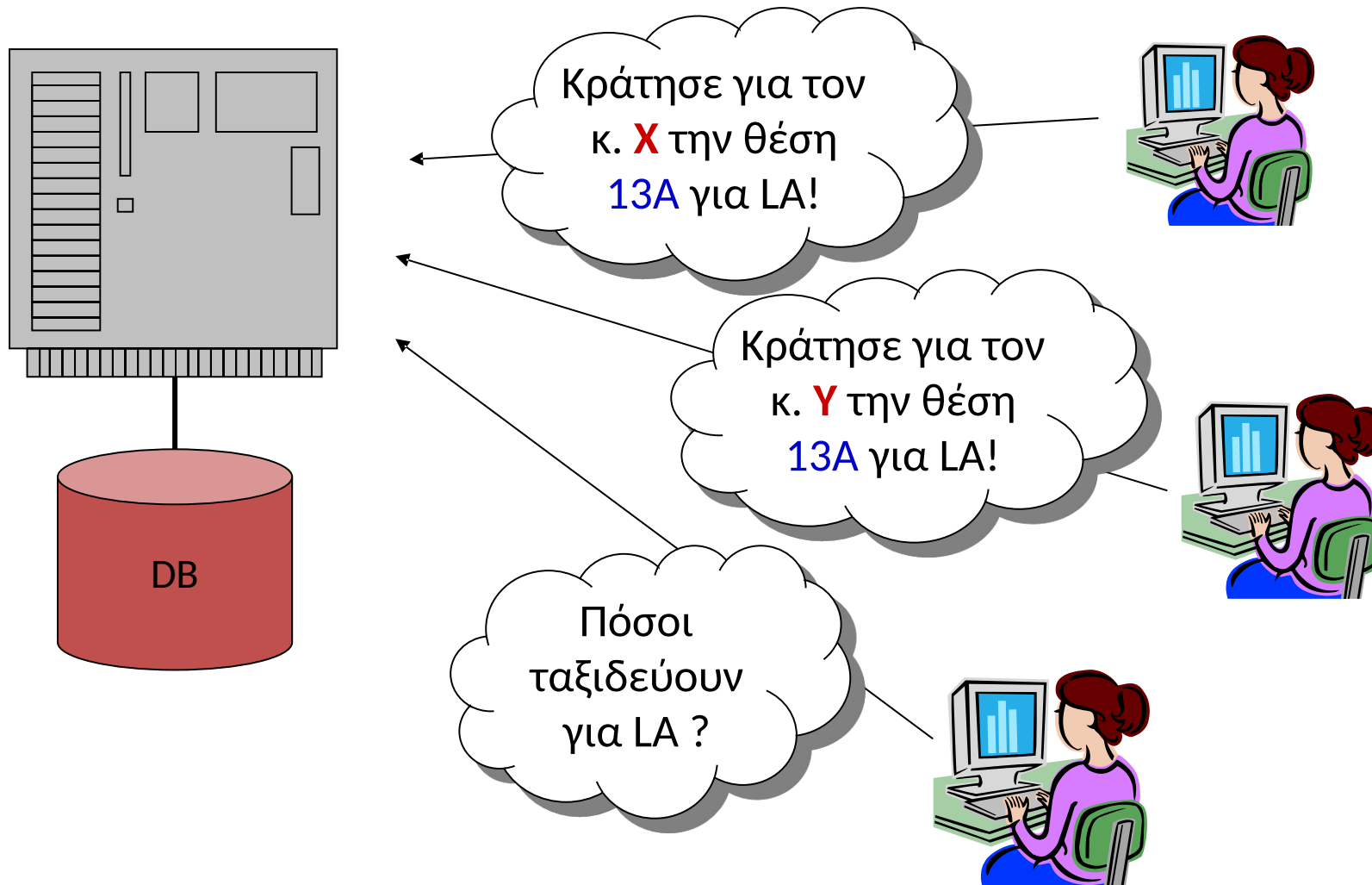
# Αρχεία και Βάσεις Δεδομένων

Διδάσκων : Δημήτρης Πλεξουσάκης

Συναλλαγές  
Διαχείριση Συναλλαγών



# Συναλλαγές





# Συναλλαγές

---

- ⇧ Οι ταυτόχρονες συναλλαγές (δοσοληψίες, **transactions**) σε μια βάση δεδομένων από πολλές διεργασίες / χρήστες είναι απαραίτητη.
- ⇧ Μια συναλλαγή (**T**) περιλαμβάνει μια σειρά από αναγνώσεις (**Reads**) και εγγραφές (**Writes**) σε αντικείμενα της ΒΔ



# Παράδειγμα Συναλλαγής

$T_0$ : μεταφορά 50€ από  
το λογαριασμό A στο  
λογαριασμό B

```
read (A) ;  
A := A - 50 ;  
write (A) ;  
read (B) ;  
B := B + 50 ;  
write (B) .
```

Σε ότι αφορά  
τη ΒΔ:

```
R (A) ; W (A) ; R (B) ; W (B) .
```



# Προβλήματα Συναλλαγών

---

- ⇧ Οι ενέργειες πολλών συναλλαγών πρέπει να εναλλάσσονται για λόγους απόδοσης
- ⇧ **Βασικά Προβλήματα:**
  - ⇧ Τι γίνεται αν κατά τη διάρκεια μιας συναλλαγής πέσει το σύστημα?
  - ⇧ Τι γίνεται αν δύο δοσοληψίες θέλουν να τροποποιήσουν το ίδιο αντικείμενο ταυτόχρονα?
- ⇧ Η «εικόνα» των δεδομένων της βάσης όπως αυτή παρουσιάζεται σε μια συναλλαγή είναι αναγκαίο να είναι ορθή κατά τη διάρκεια της συναλλαγής



# Ιδιότητες Συναλλαγών

---

Τί Ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS) πρέπει να εξασφαλίζει τις παρακάτω ιδιότητες **ACID**:

τ| **Atomicity** Ατομικότητα

Πρέπει να εκτελεστούν είτε όλες οι ενέργειες της συναλλαγής ή καμία

τ| **Consistency** – Συνέπεια

Η εκτέλεση μιας συναλλαγής θα πρέπει να διατηρεί τη συνέπεια της βάσης δεδομένων

τ| **Isolation** - Απομόνωση

Μια σειριακή εκτέλεση των συναλλαγών θα οδηγούσε στο ίδιο αποτέλεσμα, δηλαδή κάθε δοσοληψία πρέπει να νομίζει ότι τρέχει μόνη της

τ| **Durability** – Μονιμότητα ή Ανοχή

Αν μια συναλλαγή ολοκληρωθεί με επιτυχία, οι αλλαγές που έχει κάνει θα πρέπει να παραμένουν στη βάση δεδομένων, ακόμα και αν το σύστημα αποτύχει



# Συναλλαγές – Ορθή επαναδιατύπωση

---

Μια συναλλαγή (**T**) περιλαμβάνει μια σειρά από αναγνώσεις (**Reads**) και εγγραφές (**Writes**) σε αντικείμενα της ΒΔ και τελειώνει είτε με **Commit**, είτε με **Abort**



# Συμβολισμός Συναλλαγών 1/2

---

## ≡ Read

**R 1(A)**: Ανάγνωση του αντικειμένου A από τη ΒΔ από τη συναλλαγή T1

## ≡ Write

**W1(A)**: Εγγραφή του αντικειμένου A στη ΒΔ για τη συναλλαγή T1

## ≡ Commit

**C1**: Επιτυχής ολοκλήρωση της συναλλαγής T1

Οι ενέργειες της συναλλαγής πρέπει να φαίνονται στη ΒΔ!





# Συμβολισμός Συναλλαγών 2/2

---

## ≡ Abort

**A1:** Ανεπιτυχής προσπάθεια εκτέλεσης της συναλλαγής T1.  
Οι ενέργειες της συναλλαγής που έχουν ήδη εφαρμοστεί πρέπει να αναιρεθούν!

## ≡ Active

**A1:** Είναι ενεργή στο ξεκίνημα και κατά τη διάρκεια της

## ≡ Fail

Η ΒΔ έχει αντιληφθεί ότι η δοσοληψία δεν μπορεί να συνεχίσει και τη διακόπτει. Όταν η δοσοληψία χαρακτηριστεί ως Failed πρέπει να ακολουθήσει ένα Abort ώστε να αναιρεθούν όλες οι ενέργειες που έχουν γίνει



# Διαχείριση Συναλλαγών

≠ Ένα **χρονοπρόγραμμα (schedule)** περιλαμβάνει μια σειρά από ενέργειες (**R,W,C,A**) ενός σύνολου συναλλαγών (**T1,T2,..**) όπου οι ενέργειες αυτών των δοσοληψιών εμφανίζονται με την ίδια σειρά με την οποία εμφανίζονται σε κάθε δοσοληψία

≠ Παράδειγμα

Έστω οι συναλλαγές:

**T<sub>1</sub>: R(A);R(B);W(A);COMMIT**

**T<sub>2</sub>: R(A);R(B);W(B);COMMIT**

Πιθανά χρονοπρογράμματα είναι:

Schedule S<sub>1</sub>: **R<sub>1</sub>(A);R<sub>1</sub>(B);W<sub>1</sub>(A);C1;R<sub>2</sub>(A);R<sub>2</sub>(B);W<sub>2</sub>(B);C2.**

Schedule S<sub>2</sub>: **R<sub>2</sub>(A);R<sub>2</sub>(B);W<sub>2</sub>(B);C2;R<sub>1</sub>(A);R<sub>1</sub>(B);W<sub>1</sub>(A);C1.**

Schedule S<sub>3</sub>: **R<sub>1</sub>(A);R<sub>1</sub>(B);R<sub>2</sub>(A);W<sub>1</sub>(A);R<sub>2</sub>(B);C1;W<sub>2</sub>(B);C2.**



# Διαχείριση Συναλλαγών

---

Ιδιότητες χρονοπρογραμμάτων:

≡ Ένα **πλήρες** schedule είναι ένα χρονοπρόγραμμα που συμπεριλαμβάνει abort ή commit για κάθε δοσοληψία

Για παράδειγμα:

S : R2(A) W2(A) R1(A) R1(B) R2(B) W2(B) C1 C2

≡ Ένα **σειριακό** πλήρες schedule είναι ένα χρονοπρόγραμμα στο οποίο καμία συναλλαγή δεν αρχίζει πριν τελειώσει μια συναλλαγή η οποία ήδη τρέχει

Για παράδειγμα:

S : R1(A) R1(B) C1 R2(A) W2(A) R2(B) W2(B) C2

Η σειριακή εκτέλεση των δοσοληψιών εγγυάται τη συνέπεια της ΒΔ!



# Διαχείριση Συναλλαγών

---

- ⇧ Όταν δύο χρονοπρογράμματα οδηγούν στα ίδια αποτελέσματα, ανεξάρτητα από την αρχική κατάσταση λέγονται **ισοδύναμα**.
- ⇧ Ένα χρονοπρόγραμμα λέγεται **σειριακοποιήσιμο**, όταν είναι ισοδύναμο με ένα πλήρες σειριακό χρονοπρόγραμμα.
- ⇧ Κατά συνέπεια, αν μας δοθεί ένα χρονοπρόγραμμα, πρέπει να μπορέσουμε να αποφανθούμε αν είναι σειριοποιήσιμο ή όχι.



# Διαχείριση Συναλλαγών

---

⇨ **Αντικρουόμενες** ενέργειες είναι αυτές που:

- Ανήκουν σε διαφορετικές συναλλαγές **ΚΑΙ**
- Λειτουργούν στα ίδια δεδομένα **ΚΑΙ**
- Μία ενέργεια είναι Write.

⇨ **Παράδειγματα**

- αντικρουόμενες **R1(A) W2(A)**
- αντικρουόμενες **W1(A) W2(A)**
- αντικρουόμενες **W1(A) R2(A)**
- μη αντικρουόμενες **R2(A) W2(A)**
- μη αντικρουόμενες **R1(A) W2(B)**
- μη αντικρουόμενες **R1(A) R2(A)**



# Διαχείριση Συναλλαγών

---

⇐ Σε δύο ισοδύναμα χρονοπρογράμματα οι αντικρουόμενες ενέργειες εμφανίζονται με την ίδια σειρά.

– Παράδειγμα

Μη ισοδύναμα χρονοπρογράμματα:

S1 : R2(A) W2(A) R1(A) R1(B) R2(B) W2(B) C1 C2

S2 : R1(A) R1(B) C1 R2(A) W2(A) R2(B) W2(B) C2

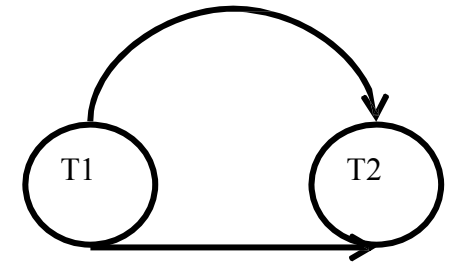
Εδώ:

- $W2(A) <_{\xi_1} R1(A)$
- $R1(A) <_{\xi_2} W2(A)$

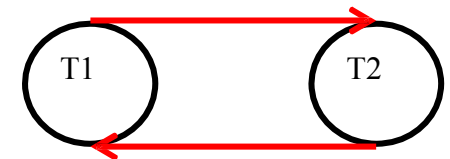


# Διαχείριση Συναλλαγών

- Για ένα χρονοπρόγραμμα μπορούμε να κατασκευάσουμε τον γραφό προτεραιότητας.
- Ένας **γραφός προτεραιότητας** είναι ένας κατευθυνόμενος γραφός όπου:
  - Κόμβοι είναι οι δοσοληψίες  $T$
  - Ακμές : μια ακμή  $T_i \rightarrow T_j$  , για κάθε ζεύγος αντικρουόμενων λειτουργιών για τις οποίες ισχύει ότι:  $op(X) < \xi op(Y)$
- **Αν περιέχει κύκλους δεν είναι σειριακοποιήσιμο.**
- Παράδειγμα
  - $S1 : R2(A) W2(A) R1(A) R1(B) R2(B) W2(B)$
  - Αντικρουόμενες
    - $W2(A) R1(A)$
    - $R1(B) W2(B)$



Σειριακοποιήσιμο



Μη Σειριακοποιήσιμο



# Διαχείριση Συναλλαγών

---

- ¶ Για τα παρακάτω προγράμματα εκτέλεσης, κατασκευάσετε τους γράφους προτεραιότητας
- a) R1(A) R2(A) W1(B) W2(B) R1(B) W2(C) W1(D)
  - b) R1(A) R2(A) R3(B) W1(A) R2(C) R2(B) W2(B) W1(C)
  - c) R1(A) W2(C) W1(B) R3(C) R2(B) W3(A)
  - d) W3(A)R1(A)W1(B)R2(B)W2(C)R3(C)R2(A)
  - e) R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)





# Διαχείριση Συναλλαγών

---

Παράδειγμα – γράφος προτεραιότητας

α) R1(A) R2(A) W1(B) W2(B) R1(B) W2(C) W1(D)



# Διαχείριση Συναλλαγών

Παράδειγμα – γράφος προτεραιότητας

α)  $R1(A)$   $R2(A)$   $W1(B)$   $W2(B)$   $R1(B)$   $W2(C)$   $W1(D)$

Αντικρουόμενες:

- $W1(B)$ ,  $W2(B)$
- $W2(B)$ ,  $R1(B)$

Υπάρχει κύκλος!  
Δεν είναι  
σειριακοποιήσιμο





# Διαχείριση Συναλλαγών

---

Παράδειγμα – γράφος προτεραιότητας

b) R1(A) R2(A) R3(B) W1(A) R2(C) R2(B) W2(B) W1(C)



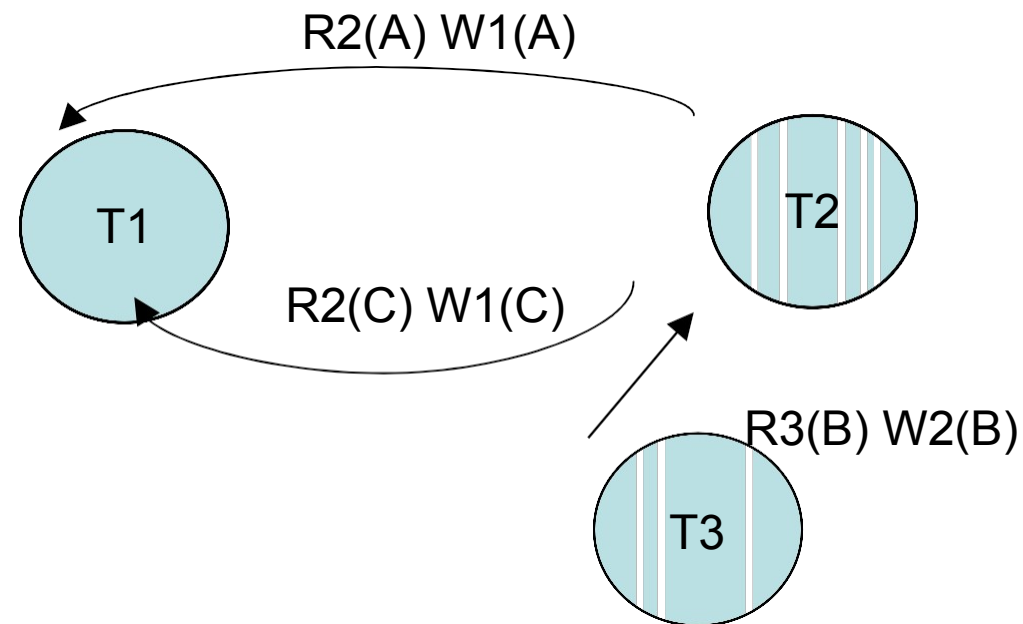
# Διαχείριση Συναλλαγών

Παράδειγμα – γράφος προτεραιότητας

b) R1(A) R2(A) R3(B) W1(A) R2(C) R2(B) W2(B) W1(C)

Αντικρουόμενες:

- R2(A), W1(A)
- R3(B), W2(B)
- R2(C), W1(C)





# Διαχείριση Συναλλαγών

---

Παράδειγμα – γράφος προτεραιότητας

C) R1(A) W2(C) W1(B) R3(C) R2(B) W3(A)



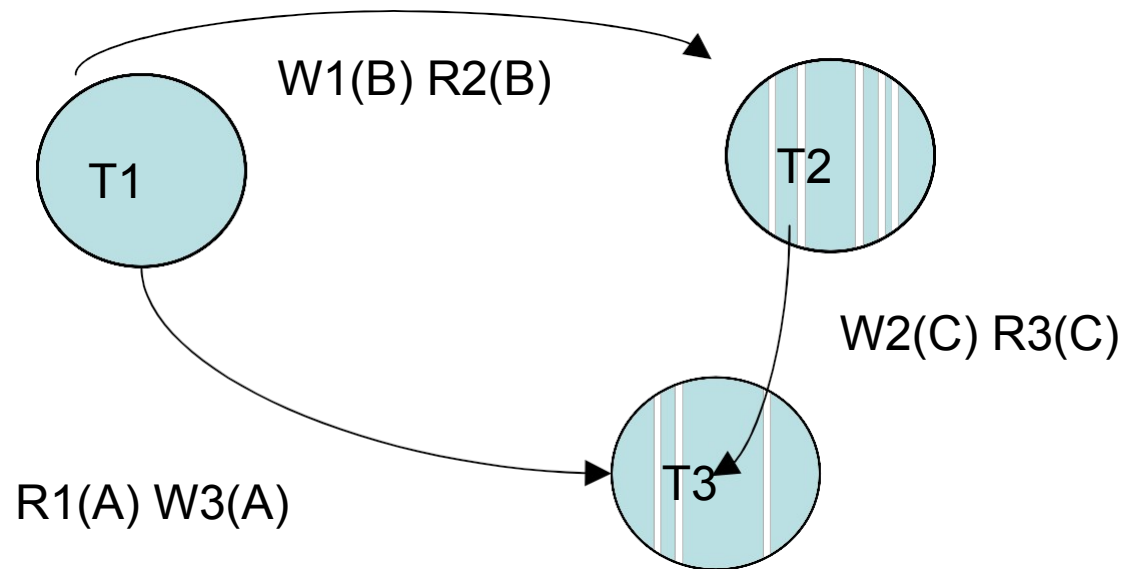
# Διαχείριση Συναλλαγών

Παράδειγμα – γράφος προτεραιότητας

C) R1(A) W2(C) W1(B) R3(C) R2(B) W3(A)

Αντικρουόμενες:

- R1(A), W3(A)
- W2(C), R3(C)
- W1(B), R2(B)





# Διαχείριση Συναλλαγών

---

Παράδειγμα – γράφος προτεραιότητας

D) W3(A)R1(A)W1(B)R2(B)W2(C)R3(C)R2(A)



# Διαχείριση Συναλλαγών

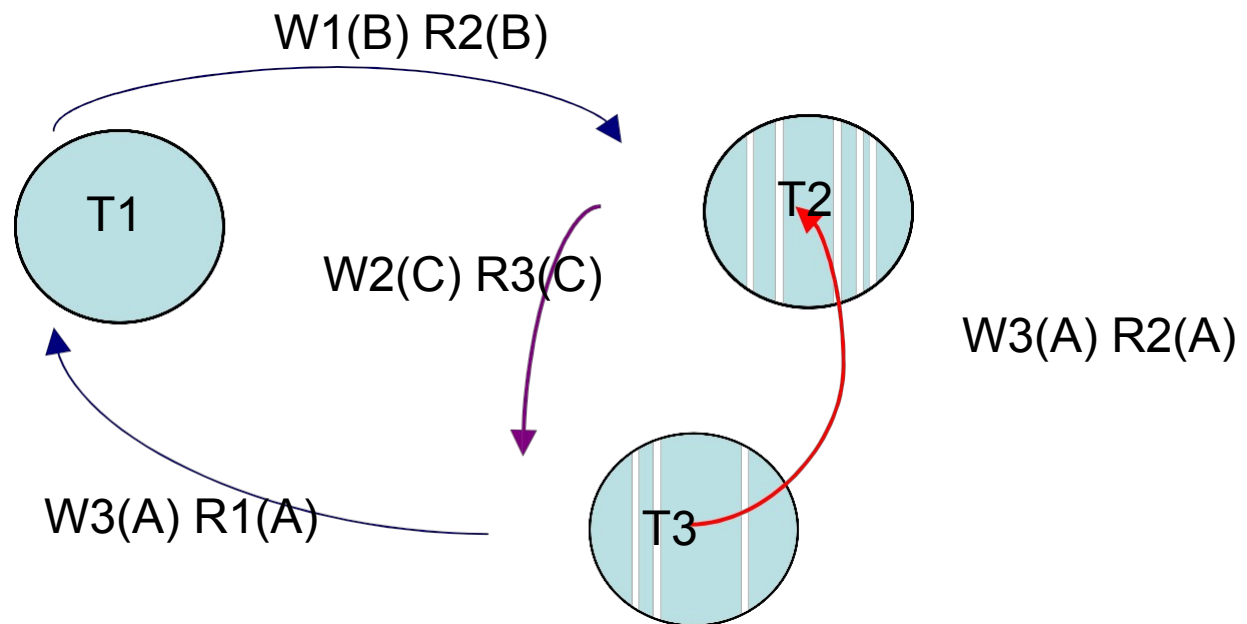
Παράδειγμα – γράφος προτεραιότητας

D)  $W3(A)R1(A)W1(B)R2(B)W2(C)R3(C)R2(A)$

Αντικρουόμενες:

- $W3(A), R1(A)$
- $W3(A), R2(A)$
- $W1(B), R2(B)$
- $W2(C), R3(C)$

Υπάρχουν κύκλοι!  
Δεν είναι  
σειριακοποιήσιμο







# Διαχείριση Συναλλαγών

---

Παράδειγμα – γράφος προτεραιότητας

E)  $R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)$



# Διαχείριση Συναλλαγών

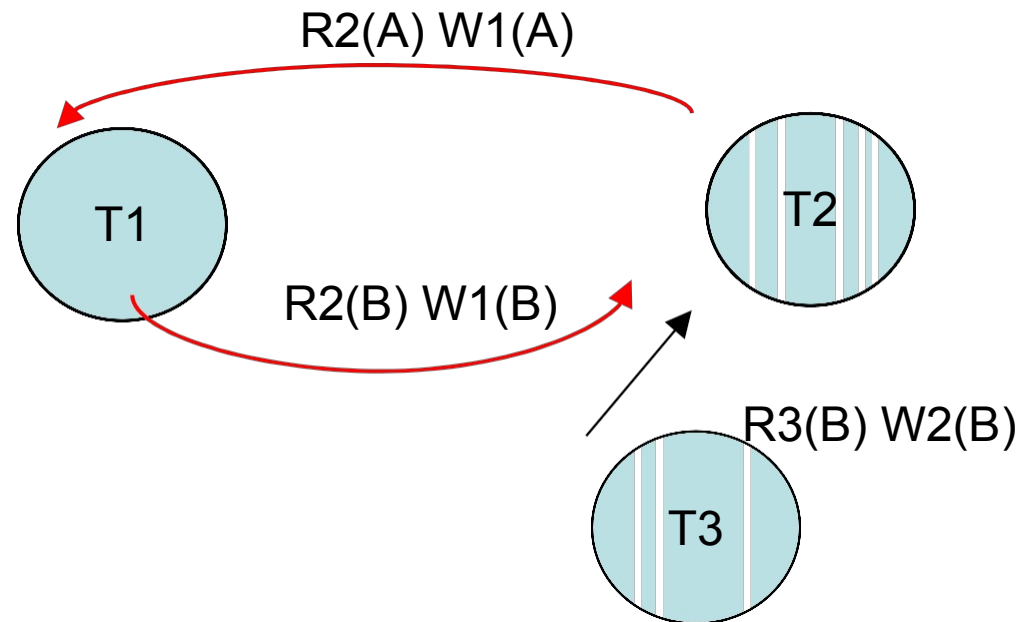
Παράδειγμα – γράφος προτεραιότητας

E)  $R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)$

Αντικρουόμενες:

- $R2(A), W1(A)$
- $R1(B), W2(B)$
- $R3(B), W2(B)$

Υπάρχει κύκλος!  
Δεν είναι  
σειριακοποιήσιμο





# Διαχείριση Συναλλαγών - locks

---

- ⇧ Για να επιτύχουμε σειριακοποιησιμότητα χρησιμοποιούμε **locks**, δικαιώματα πρόσβασης σε κάποιο αντικείμενο μιας βάσης δεδομένων
  - **Shared Lock**: όταν το έχει μια δοσοληψία, μπορούν να αποκτήσουν shared locks και οι υπόλοιπες (συνήθως χρησιμοποιείται για reads)
  - **Exclusive Lock**: όταν το έχει μια δοσοληψία, καμία άλλη δε μπορεί να διαβάσει ή να γράψει το συγκεκριμένο αντικείμενο (συνήθως χρησιμοποιείται για writes)
- ⇧ Κάθε δοσοληψία πρέπει να αποκτήσει lock για ένα αντικείμενο πριν το προσπελάσει
- ⇧ Όλα τα αντικείμενα που κλειδώνονται από μια δοσοληψία, πρέπει και να ξεκλειδώνονται από αυτή
- ⇧ Μια δοσοληψία πρέπει να περιμένει εάν ζητήσει lock σε ένα αντικείμενο που είναι κλειδωμένο από άλλη δοσοληψία



# Διαχείριση Συναλλαγών – locks 1/2

---

Με τη χρήση locks, αποφεύγονται τα προβλήματα που δημιουργούνται μεταξύ αντικρουόμενων λειτουργιών

– Εμφανίζονται όμως νέα προβλήματα

- **Livelock**: μια δοσοληψία προσπαθεί να αποκτήσει lock σε ένα αντικείμενο, στο οποίο διαδοχικά αποκτούν locks άλλες δοσοληψίες, έτσι περιμένει επ' αόριστον

Μπορεί να αποφευχθεί με τη χρήση της στρατηγικής first-come-first-served, ωστόσο ακόμα και τότε ένα deadlock μπορεί να προκύψει

- **Deadlock**: για ένα σύνολο δοσοληψιών, κάθε μία περιμένει μία από τις άλλες να κάνει unlock ένα αντικείμενο ώστε να αποκτήσει lock σε αυτό.



## Διαχείριση Συναλλαγών – locks 2/2

---

Για να αποφύγουμε ένα ενδεχόμενο deadlock μπορούμε να:

1. Απαιτήσουμε κάθε συναλλαγή να πραγματοποιεί όλα τα locks που χρειάζεται μαζεμένα
2. Να μην κάνουμε τίποτα! Απλά κάνουμε abort μια ή περισσότερες από τις συναλλαγές που συμμετέχουν στο deadlock και βλέπουμε αν έχει λυθεί το πρόβλημα

Ένα deadlock μπορεί να γίνει εύκολα αντιληπτό σε ένα γράφο προτεραιότητων ➡ Εμφανίζεται τουλάχιστον ένας κύκλος



# Wait-for Graph

- **Precedence Graph (Γράφος Προτεραιότητας):** Χρησιμοποιείται για να ανακαλύψουμε αν ένα χρονοπρόγραμμα είναι **σειριακοποιήσιμο**.

**Κύκλος → μη σειριακοποιήσιμο!**

- **Wait-for Graph (Γράφος Αναμονής):** Χρησιμοποιείται για να ανακαλύψουμε αν ένα χρονοπρόγραμμα περιέχει **Deadlock**.

**Κύκλος → Deadlock!**



# Wait-for Graph

- Για ένα χρονοπρόγραμμα συναλλαγών, ο **γράφος αναμονής** είναι ένας **κατευθυνόμενος γράφος** όπου:
  - οι **κορυφές** του είναι οι **συναλλαγές** που υπάρχουν στο χρονοπρόγραμμα
  - υπάρχει **ακμή (conflict)** από την **T<sub>i</sub> προς την T<sub>j</sub>**, εάν η **T<sub>i</sub>** περιμένει να **κλειδώσει** ένα στοιχείο το οποίο είναι **κλειδωμένο** από την **T<sub>j</sub>**.
- Περιπτώσεις **συγκρούσεων** μεταξύ κλειδώματων:
  - $SL_j(A) \ XLi(A)$
  - $XL_j(A) \ XLi(A)$
  - $XL_j(A) \ SL_i(A)$

Υπάρχει **DEADLOCK** στον γράφο αν υπάρχει **κύκλος!**

# Wait-for Graph

## Παράδειγμα:

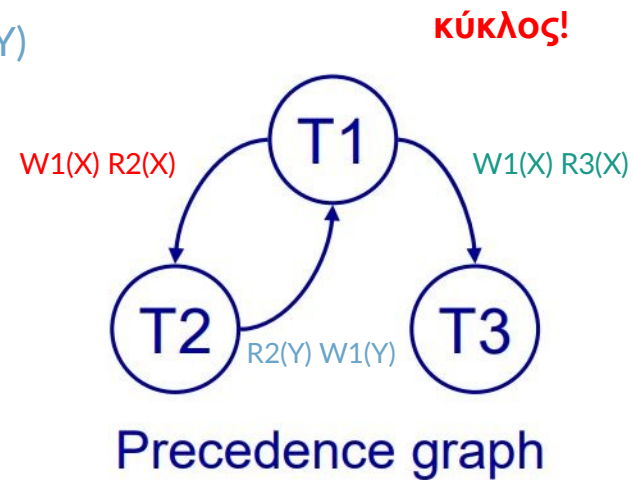
S: R1(X) R2(Y) W1(X) R2(X) R3(Z) W3(Z) R1(Y) R3(X) W1(Y)

## Γράφος Προτεραιότητας:

R2(Y) W1(Y)

W1(X) R2(X)

W1(X) R3(X)





# Wait-for Graph

## Παράδειγμα:

S: R1(X) R2(Y) W1(X) R2(X) R3(Z) W3(Z) R1(Y) R3(X) W1(Y)

- Για να κατασκευάσουμε τον γράφο αναμονής, μετατρέπουμε τα Reads και τα Writes στα κλειδιά που τους αντιστοιχούν (R→ SL, W→ XL).

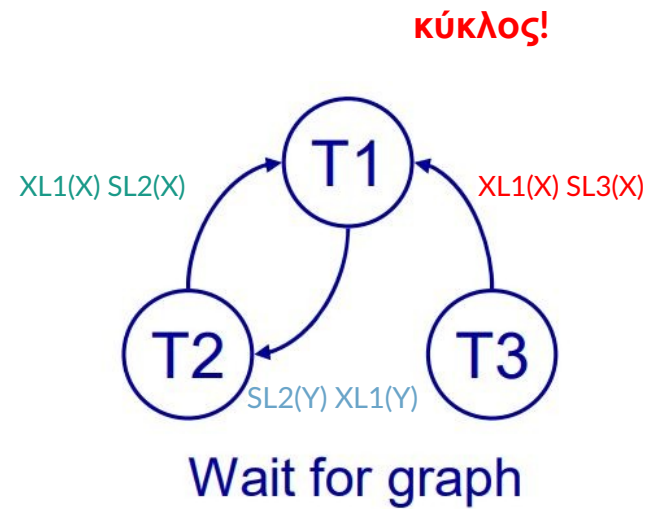
S: SL1(X) SL2(Y) XL1(X) SL2(X) SL3(Z) XL3(Z) SL1(Y) SL3(X) XL1(Y)

## Γράφος Προτεραιότητας:

SL2(Y) XL1(Y)

XL1(X) SL3(X)

XL1(X) SL2(X)





# Διαχείριση Συναλλαγών – 2PL

---

- ¶ Ένα πρόγραμμα εκτέλεσης ακολουθεί το πρωτόκολλο **Two Phase Locking (2PL)**, αν ισχύουν τα παρακάτω:
  - Όταν μια δοσοληψία θέλει να διαβάσει ή να γράψει κάποιο αντικείμενο, πρέπει να αποκτήσει ένα lock ( read RL ή write WL ) σε αυτό
  - Εάν ένα αντικείμενο είναι κλειδωμένο με write lock, κάθε δοσοληψία που θέλει να αποκτήσει πρόσβαση σε αυτό πρέπει να περιμένει, δηλαδή το lock είναι αποκλειστικό. (WL = XL)
  - Όλα τα locks που χρειάζεται μια δοσοληψία αποκτώνται κατά τη διάρκεια του **Growing phase (Φάση Ανάπτυξης)**.
  - Όλα τα locks ελευθερώνονται από τη δοσοληψία στο **Shrinking phase. Φάση Συρρίκνωσης**
  - Κανένα lock δε μπορεί να αποκτηθεί κατά τη διάρκεια του shrinking phase.  
*Συνεπώς, για κάθε δοσοληψία, όλα τα locks πρέπει να προηγούνται του πρώτου unlock.*
- ¶ Ένα πρόγραμμα εκτέλεσης που ακολουθεί το 2PL είναι πάντα σειριακοποιήσιμο



# Διαχείριση Συναλλαγών – 2PL

---

## Παράδειγμα

Για τα παρακάτω προγράμματα εκτέλεσης, δείξτε πως θα εκτελεστούν με βάση το πρωτόκολλο 2PL. Υποθέστε ότι

- υπάρχει μόνο αποκλειστικό lock
- κάθε lock αποκτάται αμέσως πριν από το αντίστοιχο read/write
- ελευθερώνονται όλα μαζί αμέσως μετά το τελευταίο read/write.

- R1(A) R2(A) W1(B) W2(B) R1(B) W2(C) W1(D)
- R1(A) R2(A) R3(B) W1(A) R2(C) R2(B) W2(B) W1(C)
- R1(A) W2(C) W1(B) R3(C) R2(B) W3(A)
- W3(A)R1(A)W1(B)R2(B)W2(C)R3(C)R2(A)
- R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)



# Διαχείριση Συναλλαγών - 2PL

- Παράδειγμα – 2PL

a) R1(A) R2(A) W1(B) W2(B) R1(B) W2(C) W1(D)

T1	T2
L1(A), R1(A)	
	L2(A) NO
L1(B), W1(B), R1(B)	...
L1(D), W1(D)	...
U1(A), U1(B), U1(D)	
	L2(A), R2(A)
	L2(B), W2(B)
	L2(C), W2(C)
	U2(A), U2(B), U2(C)



# Διαχείριση Συναλλαγών – 2PL

- Παράδειγμα – 2PL

b) R1(A) R2(A) R3(B) W1(A) R2(C) R2(B) W2(B) W1(C)

T1	T2	T3
L1(A), R1(A)		
	L2(A), NO	
		L3(B), R3(B), U3(B)
W1(A)		
L1(C), W1(C)		
U1(A), U1(C)		
	L2(A), R2(A)	
	L2(C), R2(C)	
	L2(B), R2(B), W2(B)	
	U2(A), U2(C), U2(B)	



# Διαχείριση Συναλλαγών - 2PL

- Παράδειγμα - 2PL  
c) R1(A) W2(C) W1(B) R3(C) R2(B) W3(A)

T1	T2	T3
L1(A), R1(A)		
	L2(C), W2(C)	
L1(B), W1(B)		
U1(A), U1(B)		
		L3(C), NO
	L2(B), R2(B)	
	U2(C), U2(B)	
		L3(C), R3(C)
		L3(A), W3(A)
		U3(C), U3(A)



# Διαχείριση Συναλλαγών - 2PL

- Παράδειγμα - 2PL  
d) W3(A)R1(A)W1(B)R2(B)W2(C)R3(C)R2(A)

T1	T2	T3
		L3(A), W3(A)
L1(A), NO		
	L2(B), R2(B)	
	L2(C), W2(C)	
		L3(C), NO
	L2(A), NO	





# Διαχείριση Συναλλαγών - 2PL

- Παράδειγμα - 2PL

e) R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)

T1	T2	T3
L1(A), R1(A)		
	L2(A), NO	
L1(B), R1(B)		
		L3(B), NO
W1(A)		
U1(A), U2(B)		
	L2(A), R2(A)	
	L2(B), R2(B)	
		L3(B), NO
	W2(B)	
	U2(A), U2(B)	
		L3(B), R3(B), U3(B)





## Διαχείριση Συναλλαγών – 2PL

---

- Πόσα νόμιμα σύγχρονα προγράμματα εκτέλεσης υπάρχουν για τις παρακάτω δύο δοσοληψίες;
  - T1 : L1(A) R1(A) W1(A) L1(B) R1(B) W1(B) U1(A) U1(B)
  - T2 : L2(B) R2(B) W2(B) L2(A) R2(A) W2(A) U2(B) U2(A)



# Διαχείριση Συναλλαγών - 2PL

- T1 : L1(A) R1(A) W1(A) L1(B) R1(B) W1(B) U1(A) U1(B)
- T2 : L2(B) R2(B) W2(B) L2(A) R2(A) W2(A) U2(B) U2(A)

T1	T2
L1(A), R1(A) W1(A)	
	L2(B) R2(B) W2(B)
	L2(A) NO
L1(B), NO	

T1	T2
	L2(B) R2(B) W2(B)
L1(A), R1(A) W1(A)	
	L2(A) NO
L1(B), NO	



Άρα μόνο σειριακές εκτελέσεις.

T1, T2

Ή

T2, T1



## Διαχείριση Συναλλαγών II - Tree Protocols

# Διαχείριση Συναλλαγών - Επανάληψη

- **Συναλλαγή (T)** είναι ένα σύνολο από ενέργειες σε κάποιο αντικείμενο της ΒΔ
  - Μπορεί να περιέχει ενέργειες **Read(R), Write (W), Commit (C), Abort (A)**
- Οι συναλλαγές πρέπει να εναλλάσσονται στην εκτέλεση τους για λόγους απόδοσης της ΒΔ εξασφαλίζοντας πάντα τις ιδιότητες **ACID**
- **Χρονοπρόγραμμα (schedule)** είναι μια σειρά από ενέργειες (**R, W, C, A**) μιας ομάδας δοσοληψιών όπου εμφανίζονται όλες οι ενέργειες αυτών των δοσοληψιών διατηρώντας τη σειρά με την οποία εμφανίζονται σε κάθε δοσοληψία
- **Σειριακό χρονοπρόγραμμα:** όταν οι συναλλαγές εκτελούνται εξ ολοκλήρου η μία μετά την άλλη

# Διαχείριση Συναλλαγών - Επανάληψη

- **Πλήρες χρονοπρόγραμμα:** ένα χρονοπρόγραμμα που συμπεριλαμβάνει abort ή commit στο τέλος της κάθε δοσοληψίας
- **Ισοδύναμα χρονοπρογράμματα:** όταν δύο χρονοπρογράμματα οδηγούν στα ίδια αποτελέσματα ανεξάρτητα από την αρχική κατάσταση
- **Σειριακοποιήσιμο χρονοπρόγραμμα:** όταν είναι ισοδύναμο με ένα πλήρες σειριακό χρονοπρόγραμμα
- Με το πρωτόκολλο 2PL εξασφαλίζουμε την σειριοποιησιμότητα, θεωρώντας ότι κάθε αντικείμενο είναι ανεξάρτητο από τα άλλα.

# Διαχείριση Συναλλαγών - Επανάληψη

## ▣ Άσκηση

Θεωρήστε το παρακάτω χρονοπρόγραμμα  $S = R_1(D) R_2(B) W_2(B) R_2(A) R_3(A) W_2(A) W_3(C) R_1(B)$ .

Δείξτε πως θα εκτελεστεί με βάση το πρωτόκολλο 2PL.

## ▣ Απάντηση

T1	T2	T3
SL1(D), R1(D)		
	SL2(B), R2(B)	
	XL2(B), W2(B)	
	SL2(A), R2(A)	
		SL3(A), R3(A)
	XL2(A), WAIT	
		XL3(C), W3(C)
		U3(A), U3(C)
SL1(B), WAIT		
	XL2(A), W2(A)	
	U2(B), U2(A)	
SL1(B), R1(B)		
U1(D), U1(B)		

# Διαχείριση Συναλλαγών – Tree Protocols

- ≡ Μια ΒΔ μπορεί να περιέχει και ομαδοποιημένες περιοχές δεδομένων με διαφορετικό βαθμό ομαδοποίησης ή να υπάρχει μια μερική διάταξη των δεδομένων.
- ≡ Για να εφαρμόσουμε πρωτόκολλα χρειάζεται να έχουμε γνώση αυτής της διάταξης ή ομαδοποίησης
- ≡ Ασχολούμαστε με ομαδοποιήσεις ή διατάξεις που αποτελούν ένα δέντρο
  - ≡ TP#1 όταν έχουμε μερική διάταξη των δεδομένων
  - ≡ TP#2 όταν έχουμε ομαδοποίηση των δεδομένων

# Διαχείριση Συναλλαγών – TP#1

- ▣ Εφαρμόζεται σε δεδομένα  $D = \{d_1, d_2, \dots, d_n\}$  τα οποία είναι μερικώς διατεταγμένα (άρα μπορούν να αναπαρασταθούν ως δέντρο). Αν  $d_i \rightarrow d_j$ , τότε κάθε συναλλαγή που απαιτεί πρόσβαση και στο  $d_i$  και στο  $d_j$ , θα πρέπει να επισκέπτεται πρώτα το  $d_i$  και μετά το  $d_j$
- ▣ Έχουμε **μόνο αποκλειστικά κλειδώματα XL**
- ▣ Το πρώτο κλείδωμα γίνεται σε οποιοδήποτε στοιχείο του δέντρου.
- ▣ Ένα **κλείδωμα** σε ένα στοιχείο γίνεται **μόνο αν** ο γονέας είναι ήδη κλειδωμένος από την ίδια συναλλαγή
- ▣ Κάθε στοιχείο κλειδώνεται **μόνο μία φορά** στην διάρκεια της συναλλαγής.
  - ▣ Καθοδική πορεία κλειδώματος προς τα φύλλα
- ▣ Το ξεκλείδωμα μπορεί να γίνει οποιαδήποτε στιγμή
  - ▣ Δεν υπάρχει φάση ανάπτυξης, συρρίκνωσης



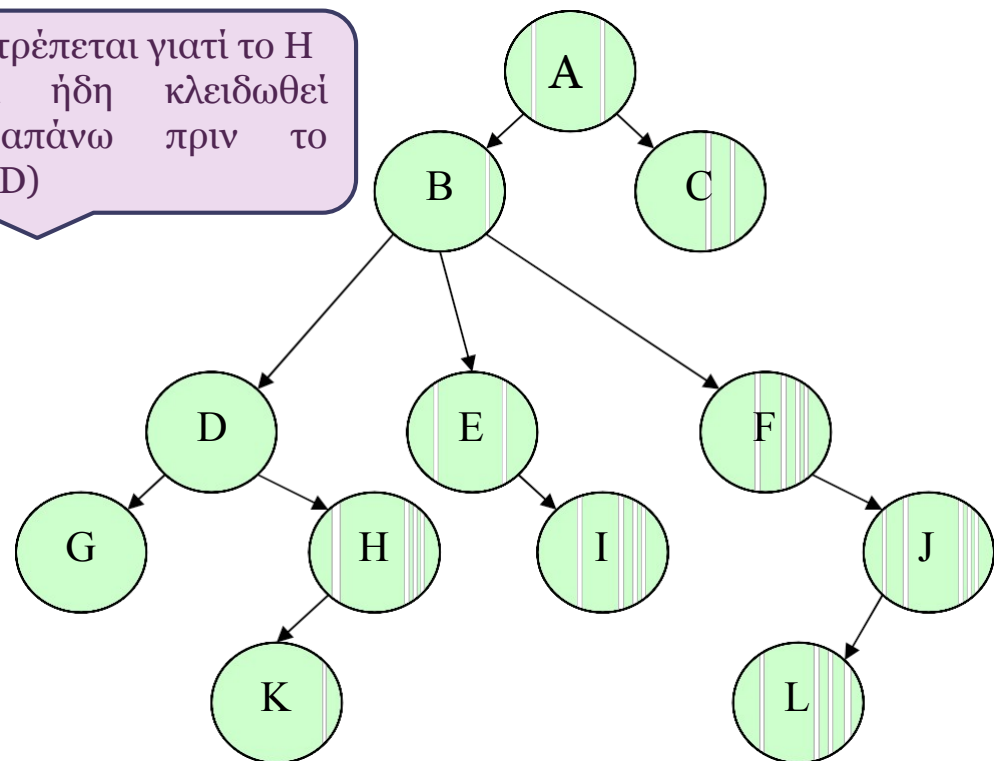
# Διαχείριση Συναλλαγών - TP#1

T1	T2
L1(B), R1(B)	
L1(E), R1(E)	
	L2(D) R2(D)
	L2(H), R2(H) W2(D), U2(D)
L1(D), R1(D)	
W1(B),U1(B)	
W1(E), U1(E)	
	W2(H)
L1(G), R1(G)	
	L2(K), R2(K), U2(H), U2(K)
U1(D)	
W1(G),U1(G)	

Παράδειγμα 1: Πως θα εκτελεστεί με το TP#1 ?

S: R1(B) R1(E) R2(D) R2(H) W2(D) R1(D)  
W1(B) W1(E) W2(H) R1(G) R2(K) W1(G)

Επιτρέπεται γιατί το H  
έχει ήδη κλειδωθεί  
παραπάνω πριν το  
U2(D)

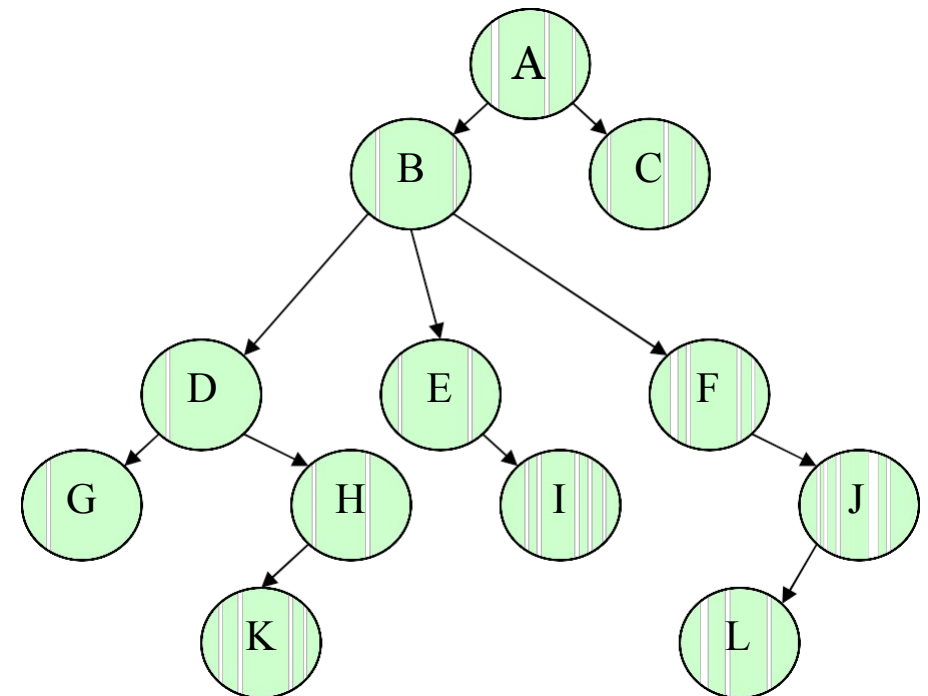


# Διαχείριση Συναλλαγών – TP#1

T1	T2	T3
		L3(A), W3(A)
L1(A), NO		
	L2(A), NO	
		L3(C), R3(C)
		U3(A), U3(C)
L1(A), R1(A)		
L1(B), R1(B), U1(A)		
	L2(A) R2(A)	
W1(B), U1(B)		
	L2(B), R2(B), U2(B), L2(C), W2(C), U2(C)	
	R2(A), U2(A)	

Παράδειγμα 2: Πως θα εκτελεστεί με TP#1 ?

S: W3(A) R1(A) R1(B) R2(A) W1(B)  
R2(B) W2(C) R3(C) R2(A)



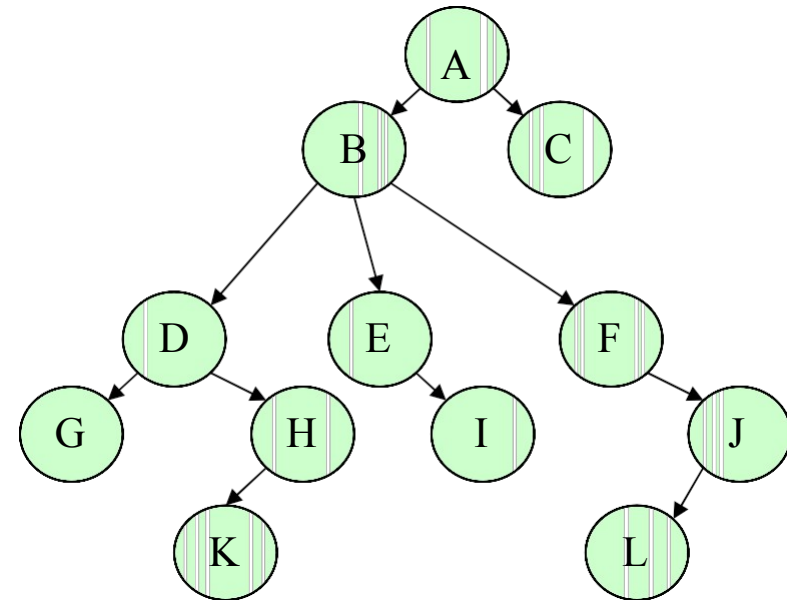
# Διαχείριση Συναλλαγών – TP#1

T1	T2
L1(B), R1(B)	
L1(F), R1(F), U1(F)	
	L2(A) R2(A)
	L2(B) NO
L1(E), R1(E), U1(B)	
	L2(B), U2(A), L2(D), U2(B)
W1(E), L1(I)	
	R2(D), U2(D)
R1(I), U1(E), U1(I)	

Παράδειγμα 3: Φτιάξτε χρονοπρόγραμμα S χρησιμοποιώντας TP#1

T1: L1(B), R1(B), L1(F), R1(F), U1(F), L1(E),  
R1(E), U1(B), W1(E), L1(I), R1(I), U1(E),  
U1(I)

T2: L2(A), R2(A), L2(B), U2(A), L2(D), U2(B),  
R2(D), U2(D)



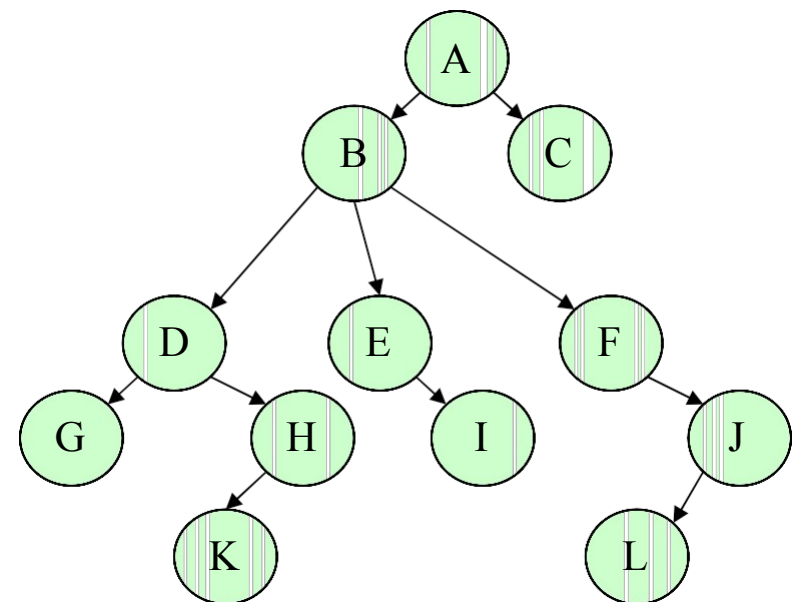
# Διαχείριση Συναλλαγών – TP#1

T1	T2
	L2(A) R2(A)
L1(B), R1(B)	
L1(F), R1(F), U1(F)	
	L2(B) NO
L1(E), R1(E), U1(B)	
	L2(B), U2(A), L2(D), U2(B)
W1(E), L1(I)	
	R2(D), U2(D)
R1(I), U1(E), U1(I)	

Παράδειγμα 4: Φτιάξτε χρονοπρόγραμμα χρησιμοποιώντας TP#1

T1: L1(B), R1(B), L1(F), R1(F), U1(F),  
L1(E), R1(E), U1(B), W1(E), L1(I),  
R1(I), U1(E), U1(I)

T2: L2(A), R2(A), L2(B), U2(A), L2(D),  
U2(B), R2(D), U2(D)



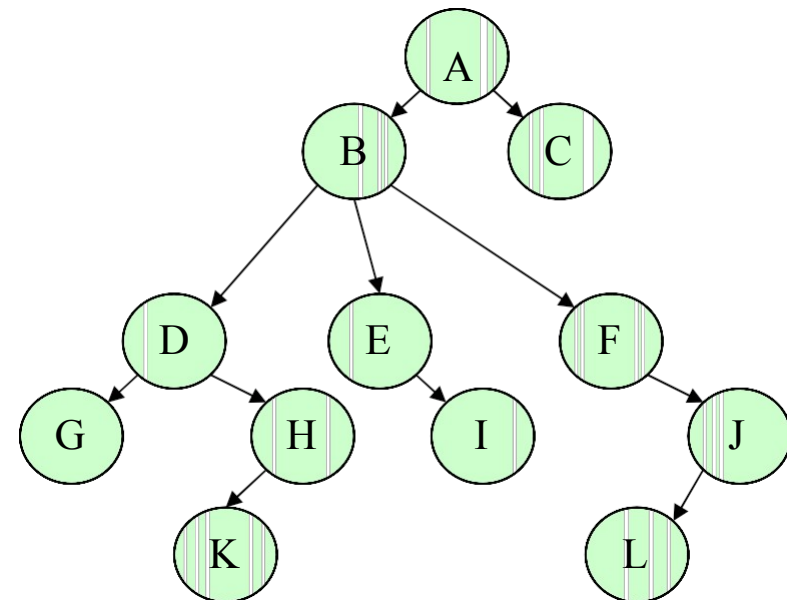
# Διαχείριση Συναλλαγών – TP#1

T1	T2
	L2(A) R2(A)
	L2(B), U2(A), L2(D),U2(B)
L1(B), R1(B)	
L1(F), R1(F),U1(F)	
L1(E), R1(E), U1(B)	
W1(E), L1(I)	
	R2(D), U2(D)
R1(I), U1(E), U1(I)	

Παράδειγμα 5: Φτιάξτε χρονοπρόγραμμα χρησιμοποιώντας TP#1

T1: L1(B), R1(B), L1(F), R1(F), U1(F),  
L1(E), R1(E), U1(B), W1(E),L1(I),  
R1(I),U1(E),U1(I)

T2: L2(A),R2(A), L2(B), U2(A) L2(D),  
U2(B), R2(D), U2(D)



# Διαχείριση Συναλλαγών – TP#2

- ≡ Έχουμε **διαμοιραζόμενα ή αποκλειστικά κλειδώματα** και **προειδοποιήσεις**
- ≡ Το κλείδωμα ενός κόμβου κλειδώνει και όλους τους απογόνους του στην ίδια κατάσταση
- ≡ Για να αποφύγω πολλαπλό κλείδωμα από συναλλαγές θέτω στους προγόνους μια προειδοποίηση(warning) πριν το κλείδωμα (**ξεκινάμε από την ρίζα προς τα φύλλα** )
- ≡ Κλειδώνω κόμβο μόνο αν δεν υπάρχει ήδη κλείδωμα ή προειδοποίηση
- ≡ Ξεκλειδώνω ή αφαιρώ προειδοποίηση αν δεν υπάρχει κλείδωμα ή προειδοποίηση σε απόγονο ( ξεκλείδωμα ή αφαίρεση προειδοποιήσεων από τα φύλλα προς την ρίζα)

# Διαχείριση Συναλλαγών – TP#2

## ≡ Συμβολισμοί:

- ⇧ Η τοποθέτηση στον κόμβο μιας προειδοποίησης διαμοιραζόμενου κλειδώματος: **IS**
- ⇧ Η τοποθέτηση στον κόμβο μιας προειδοποίησης αποκλειστικού κλειδώματος: **IX**
- ⇧ Το κλείδωμα ενός κόμβου από μια διεργασία:
  - ⇧ **X** για αποκλειστικό και **S** για διαμοιραζόμενο
- ⇧ Για να ξεκλειδώσω ένα κόμβο ή να αποσύρω μια προειδοποίηση: **U**

# Διαχείριση Συναλλαγών - TP#2

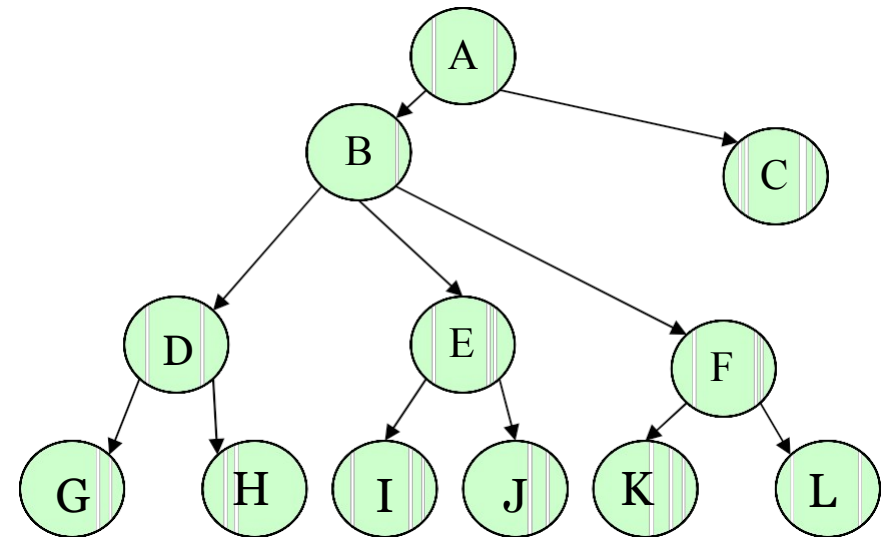
Παράδειγμα 1

T1: R1(D), W1(G), W1(H)

T2: R2(E), W2(I), W2(J)

Ένα χρονοπρόγραμμα:

S: R1(D), R2(E), W1(G), W2(I),  
W2(J), W1(H),



Δείξτε την εκτέλεση του με βάση το Tree Protocol #2

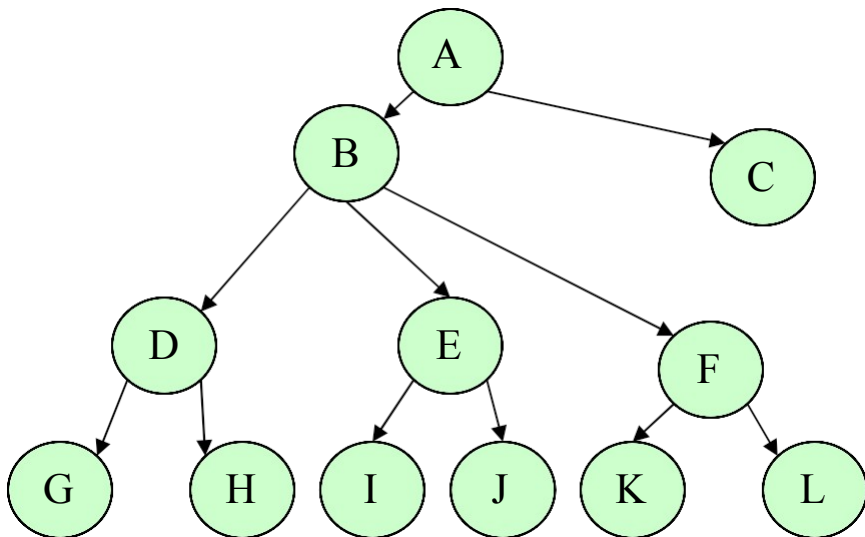


# Διαχείριση Συναλλαγών - TP#2

## Παράδειγμα 1

S: R1(D), R2(E), W1(G), W2(I), W2(J),  
W1(H),

Κάνουμε IX1(D), X1(D)  
γιατί έχουμε εγγραφή στα  
G, H



T1	T2
IX1(A), IX1(B), IX1(D), X1(D), R1(D)	
	IX2(A), IX2(B), IX2(E), X2(E), R2(E)
W1(G)	
	W2(I), W2(J), U2(I), U2(J), U2(E), U2(B), U2(A)
W1(H), U1(H), U1(G), U1(D), U1(B), U1(A)	

# Διαχείριση Συναλλαγών - TP#2

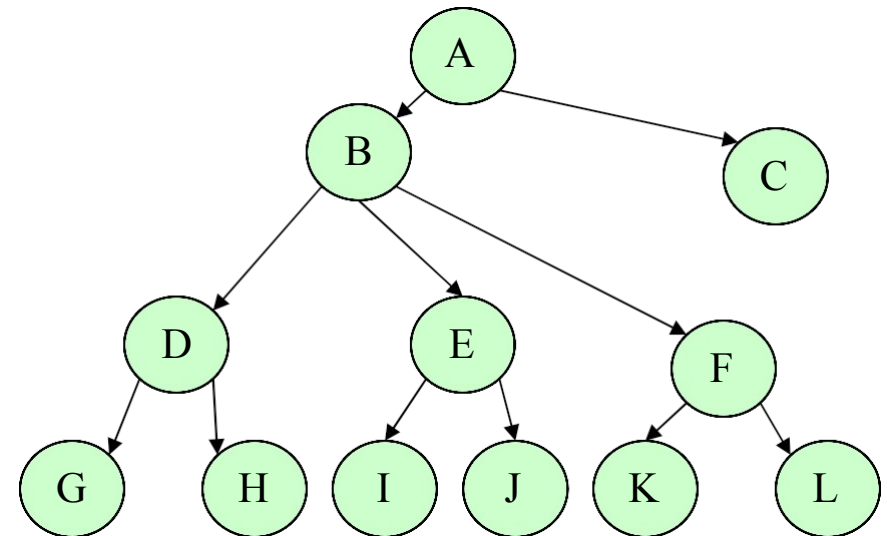
Παράδειγμα 2

T1: R1(B), R1(D), R1(G), R1(H), R1(E), R1(J),

T2: R2(D), W2(G), W2(H)

Ένα χρονοπρόγραμμα είναι:

S: R1(B), R2(D), W2(G), W2(H), R1(D),  
R1(G), R1(H), R1(E), R1(J)

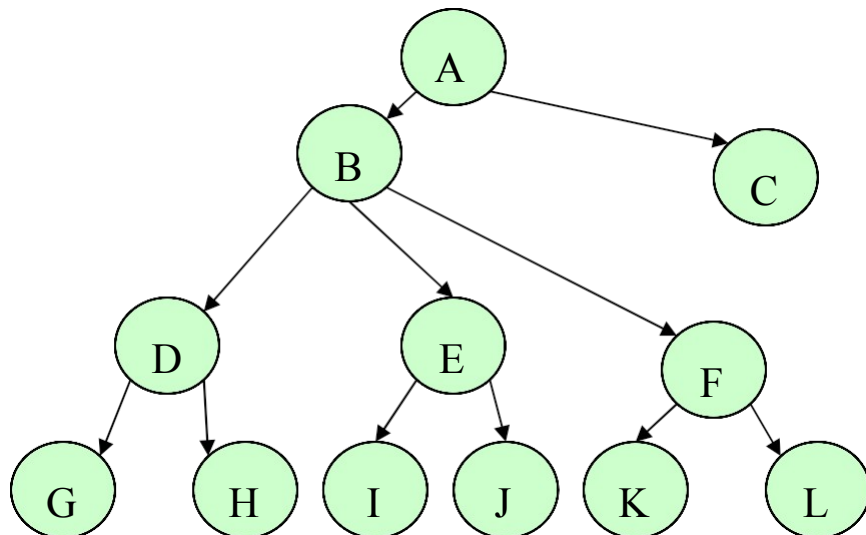


Δείξτε την εκτέλεση του με βάση το Tree Protocol #2

# Διαχείριση Συναλλαγών - TP#2

## Παράδειγμα 2

S: R1(B), R2(D) W2(G), W2(H), R1(D),  
R1(G) R1(H), R1(E), R1(J)



T1	T2
IS1(A), IS1(B), S1(B), R1(B), IS1(D)	
	IX2(A), IX2(B), IX2(D),
	X2(D), NO
S1(D), R1(D), R1(G),R1(H)	
U1(G), U1(H), U1(D)	
	X2(D), R2(D) W2(G), W2(H)
R1(E), R1(J)	
U1(J), U1(E), U1(B), U1(A)	
	U2(G), U2(H), U2(D),U2(B), U2(A)