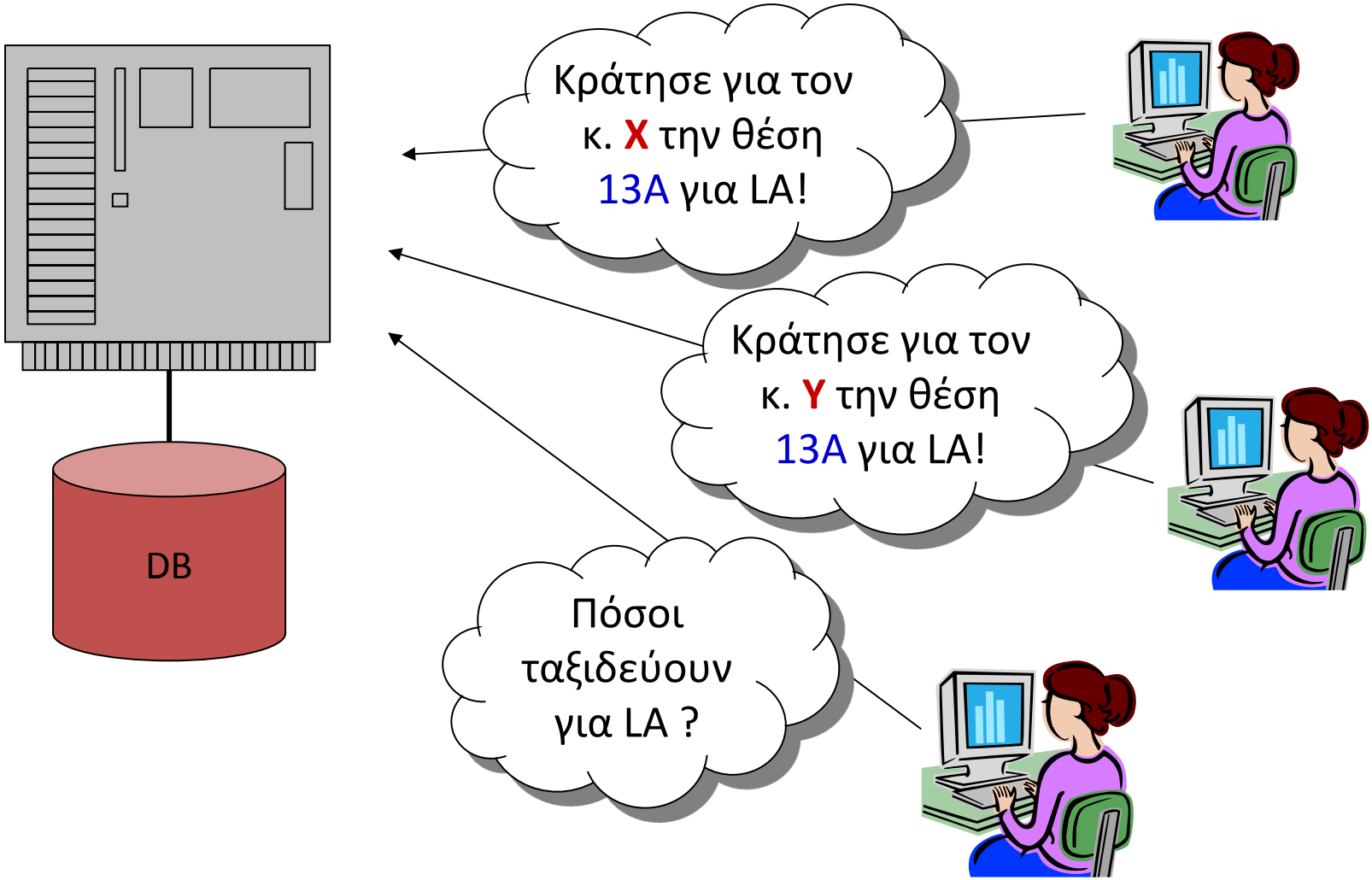
HY360
Αρχεία και Βάσεις Δεδομένων

Διδάσκων: Δημήτρης Πλεξουσάκης

Συναλλαγές
Διαχείριση Συναλλαγών

Συναλλαγές


Κράτησε για τον κ. X την θέση 13A για LA!
Κράτησε για τον κ. Y την θέση 13A για LA!
Πόσοι ταξιδεύουν για LA ?
DB

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 Α στο λογαριασμό Β

```
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

- Ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS) πρέπει να εξασφαλίζει τις παρακάτω ιδιότητες ACID:
  - Atomicity – Ατομικότητα
    Πρέπει να εκτελεστούν είτε όλες οι ενέργειες της συναλλαγής ή καμία
  - Consistency – Συνέπεια
    Η εκτέλεση μιας συναλλαγής θα πρέπει να διατηρεί τη συνέπεια της βάσης δεδομένων
  - Isolation - Απομόνωση
    Μια σειριακή εκτέλεση των συναλλαγών θα οδηγούσε στο ίδιο αποτέλεσμα, δηλαδή κάθε δοσοληψία πρέπει να νομίζει ότι τρέχει μόνη της
  - 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

  R1(A): Ανάγνωση του αντικειμένου Α από τη ΒΔ από τη συναλλαγή Τ1
- Write

  W1(A): Εγγραφή του αντικειμένου Α στη ΒΔ για τη συναλλαγή Τ1
- Commit

  C1: Επιτυχής ολοκλήρωση της συναλλαγής Τ1

  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_1: R(A);R(B);W(A);COMMIT T_2: R(A);R(B);W(B);COMMIT

Πιθανά χρονοπρογράμματα είναι:

Schedule S_1: $R_1(A);R_1(B);W_1(A);C1;R_2(A);R_2(B);W_2(B);C2.$

Schedule S_2: $R_2(A);R_2(B);W_2(B);C2;R_1(A);R_1(B);W_1(A);C1.$

Schedule S_3: $R_1(A);R_1(B);R_2(A);W_1(A);R_2(B);C1;W_2(B);C2.$

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

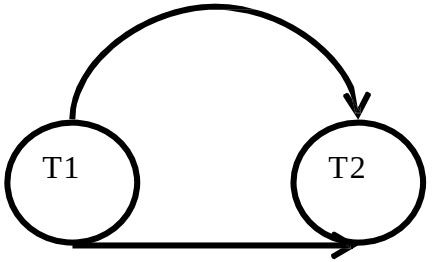
    S2 : R1(A) R1(B) C1 R2(A) W2(A) R2(B) W2(B) C2

    Εδώ:

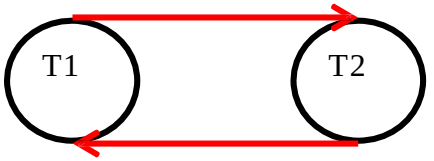
    - $W2(A) <<_{S1} R1(A)$
    - $R1(A) <<_{S2} W2(A)$

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 Ti →Tj , για κάθε ζεύγος αντικρουόμενων λειτουργιών για τις οποίες ισχύει ότι: $op_i(X) <<_S op_j(X)$
- 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)

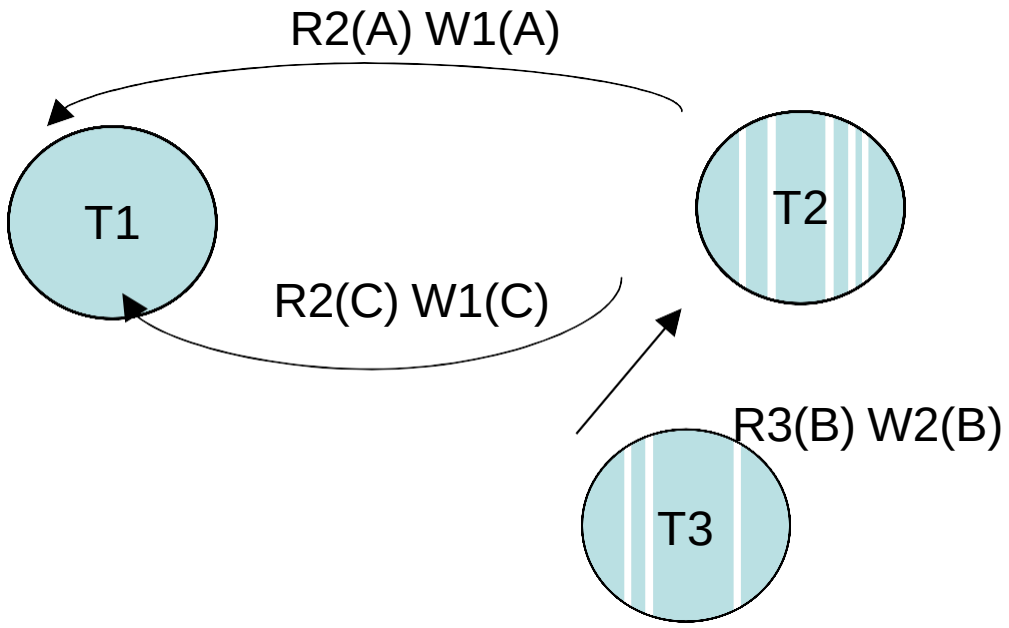
Διαχείριση Συναλλαγών

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)

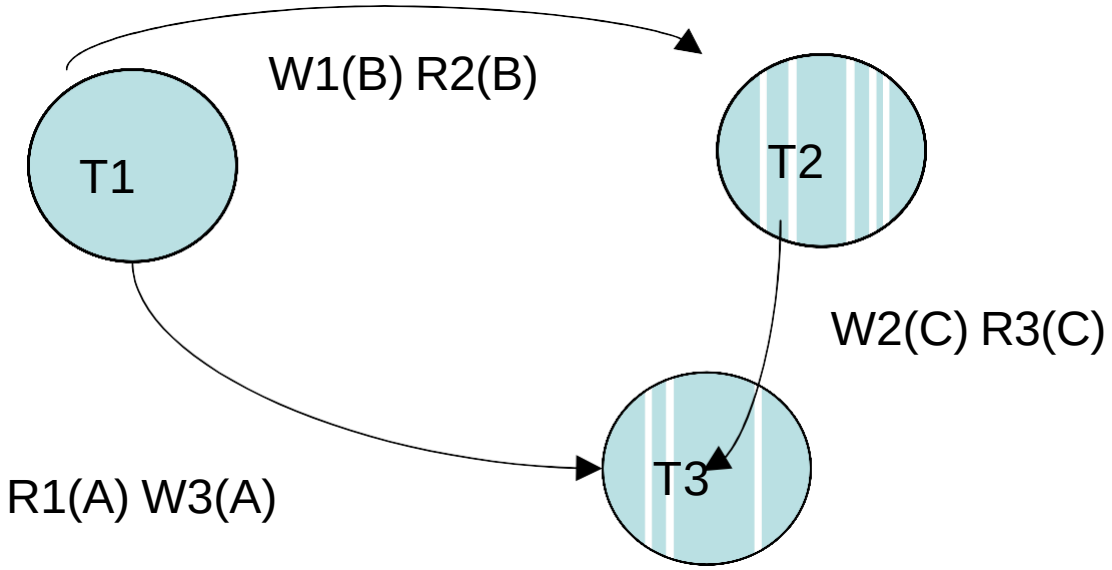
Διαχείριση Συναλλαγών

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

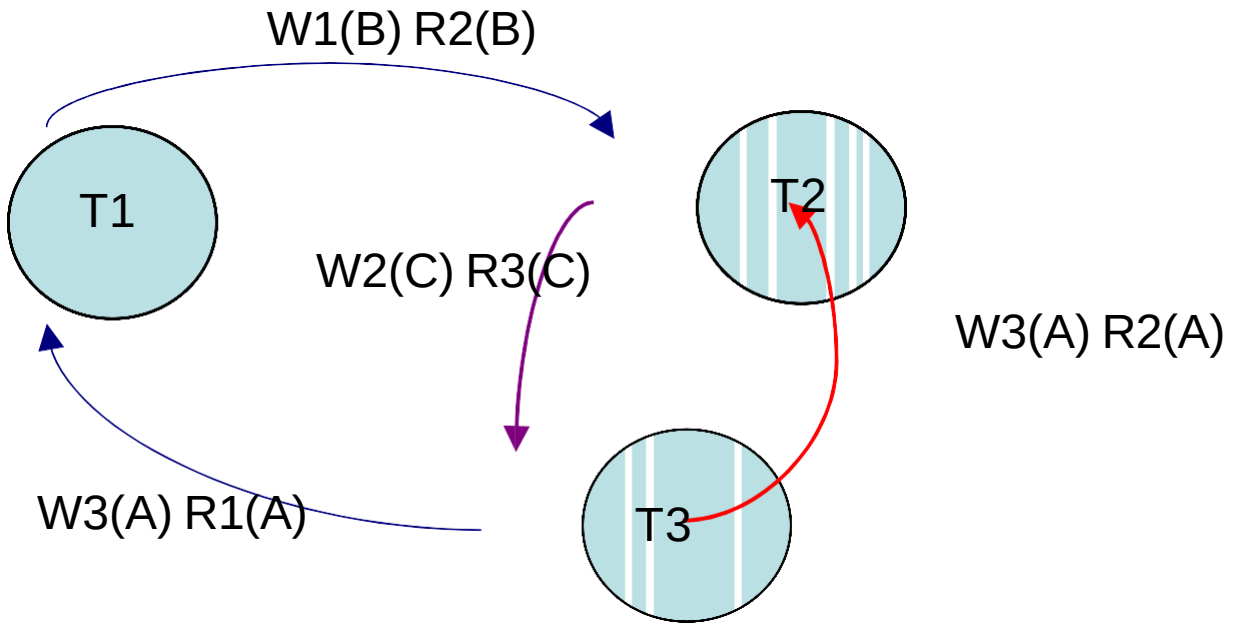
Παράδειγμα – γράφος προτεραιότητας

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



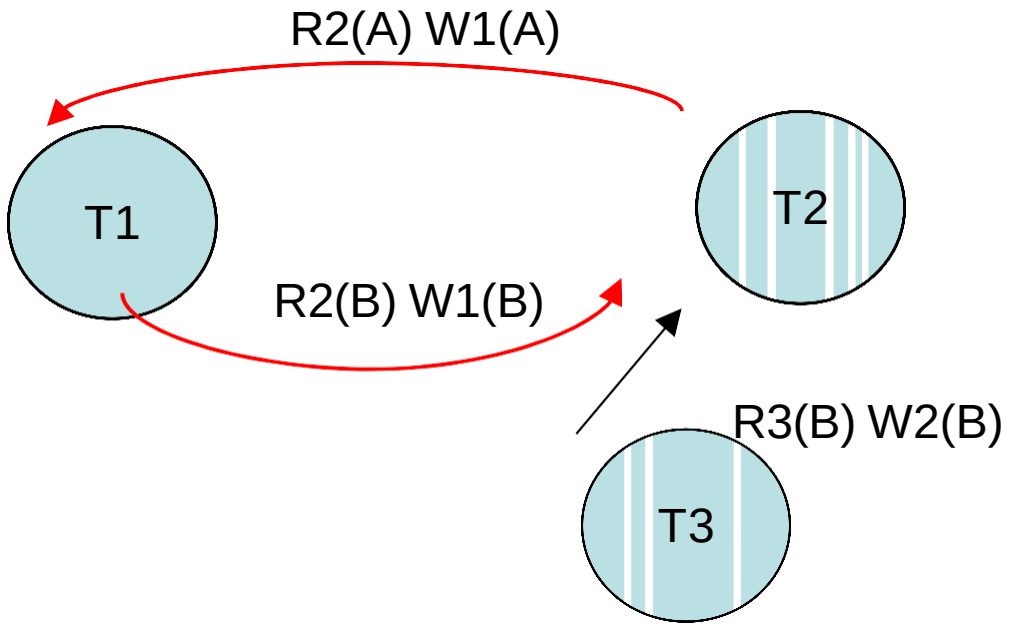
Διαχείριση Συναλλαγών

Παράδειγμα – γράφος προτεραιότητας
E) R1(A)R2(A)R1(B)R2(B)R3(B)W1(A)W2(B)

Διαχείριση Συναλλαγών

Παράδειγμα – γράφος προτεραιότητας
Ε) 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 προτεραιοτήτων ⇨ 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 **Ti προς την Tj**, εάν η **Ti** περιμένει να **κλειδώσει** ένα στοιχείο το οποίο είναι **κλειδωμένο** από την **Tj**.
- Περιπτώσεις **συγκρούσεων** μεταξύ κλειδωμάτων:
  - SLj(A) XLi(A)
  - XLj(A) XLi(A)
  - XLj(A) SLi(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)

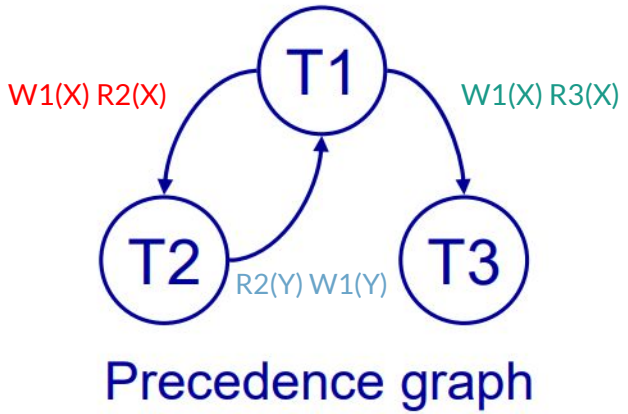
Γράφος Προτεραιότητας:

R2(Y) W1(Y)

W1(X) R2(X)

W1(X) R3(X)

κύκλος!



Precedence graph

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)

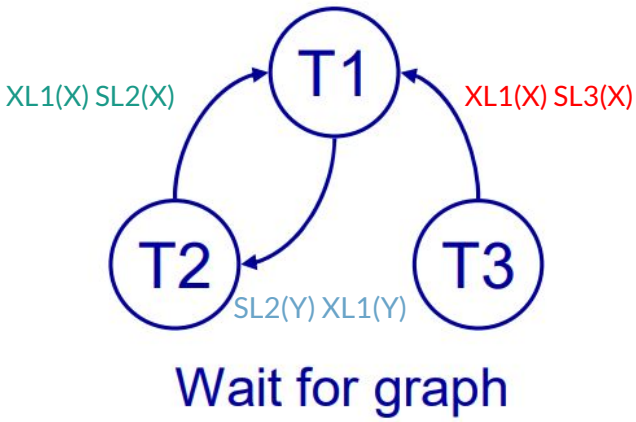
Γράφος Προτεραιότητας:

SL2(Y) XL1(Y)

XL1(X) SL3(X)

XL1(X) SL2(X)

κύκλος!



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.

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 – 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	

DEADLOCK

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

DEADLOCK

Άρα μόνο σειριακές εκτελέσεις.

T1,T2

Ή

T2, T1