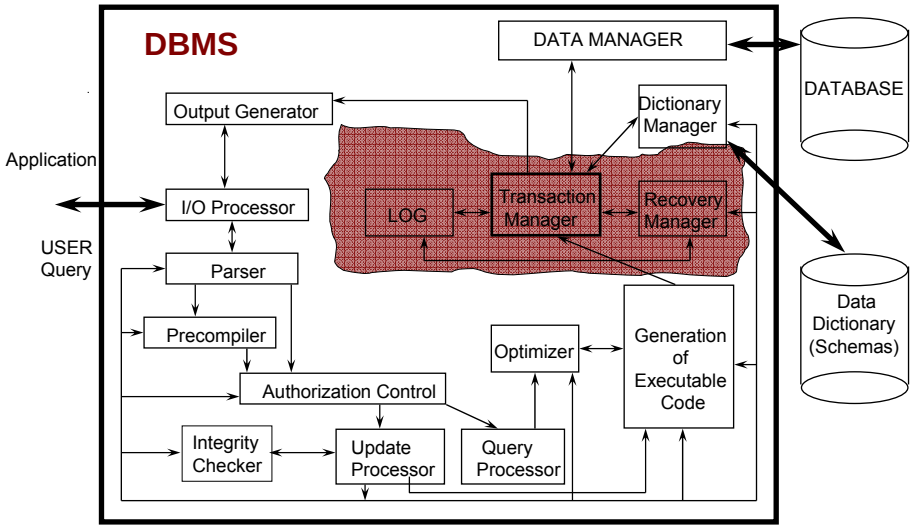
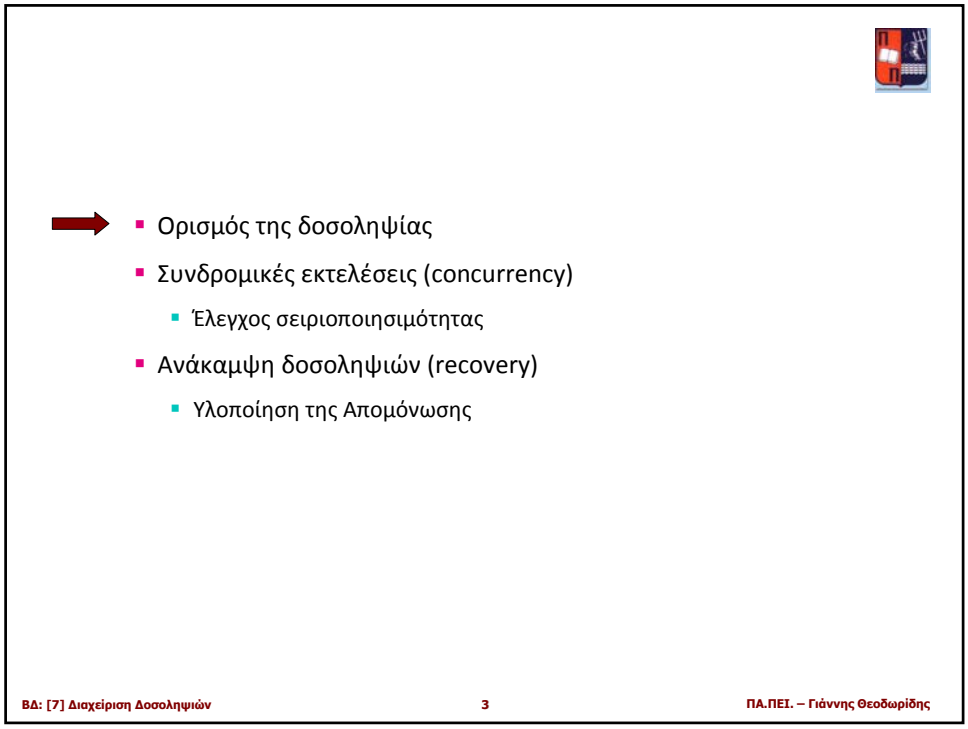
Διαχείριση Δοσοληψιών

- Ορισμός της δοσοληψίας
- Συνδρομικές εκτελέσεις (concurrency)
  - Έλεγχος σειριοποιησιμότητας
- Ανάκαμψη δοσοληψιών (recovery)
  - Υλοποίηση της Απομόνωσης

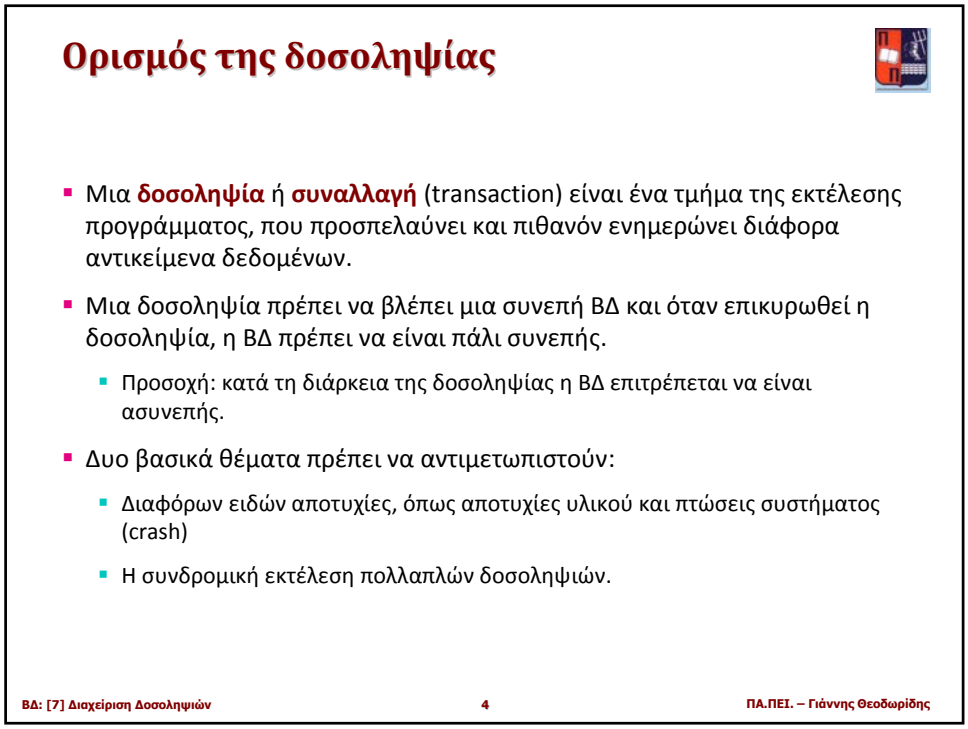
Βασική πηγή διαφανειών: Silberschatz et al., "Database System Concepts", 4/e
Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς (http://infolab.cs.unipi.gr/)
έκδοση: Φεβ. 2010

Αρχιτεκτονική ενός ΣΔΒΔ





- Ορισμός της δοσοληψίας
- Συνδρομικές εκτελέσεις (concurrency)
  - Έλεγχος σειριοποιησιμότητας
- Ανάκαμψη δοσοληψιών (recovery)
  - Υλοποίηση της Απομόνωσης



Ορισμός της δοσοληψίας

- Μια **δοσοληψία** ή **συναλλαγή** (transaction) είναι ένα τμήμα της εκτέλεσης προγράμματος, που προσπελαύνει και πιθανόν ενημερώνει διάφορα αντικείμενα δεδομένων.
- Μια δοσοληψία πρέπει να βλέπει μια συνεπή ΒΔ και όταν επικυρωθεί η δοσοληψία, η ΒΔ πρέπει να είναι πάλι συνεπής.
  - Προσοχή: κατά τη διάρκεια της δοσοληψίας η ΒΔ επιτρέπεται να είναι ασυνεπής.
- Δυο βασικά θέματα πρέπει να αντιμετωπιστούν:
  - Διαφόρων ειδών αποτυχίες, όπως αποτυχίες υλικού και πτώσεις συστήματος (crash)
  - Η συνδρομική εκτέλεση πολλαπλών δοσοληψιών.

Ιδιότητες ACID

- Για να προστατέψουμε την ακεραιότητα των δεδομένων, το σύστημα της ΒΔ πρέπει να εγγυάται:
  - **Ατομικότητα (Atomicity)**. Είτε όλες οι πράξεις της δοσοληψίας ανακλώνται στη ΒΔ είτε καμία.
  - **Συνέπεια (Consistency)**. Η εκτέλεση της δοσοληψίας σε απομόνωση προστατεύει τη συνέπεια της ΒΔ.
  - **Απομόνωση (Isolation)**. Μολονότι πολλαπλές δοσοληψίες μπορεί να εκτελούνται συνδρομικά, κάθε δοσοληψία πρέπει να αγνοεί τις υπόλοιπες συνδρομικές δοσοληψίες. Τα ενδιάμεσα αποτελέσματα μιας δοσοληψίας πρέπει να «κρύβονται» από άλλες δοσοληψίες που εκτελούνται συνδρομικά.
  - **Ανθεκτικότητα (Durability)**. Αφού μια δοσοληψία ολοκληρωθεί επιτυχώς, οι όποιες τροποποιήσεις έχει επιφέρει στη ΒΔ παραμένουν, ακόμα και αν εκ των υστέρων προκύψουν αποτυχίες του συστήματος.

Παράδειγμα μεταφοράς κεφαλαίου (1)

- Δοσοληψία για τη μεταφορά 50€ από το λογαριασμό Α στο λογαριασμό Β:

```
1. read (A);
2. A := A – 50;
3. write (A);
4. read (B);
5. B := B + 50;
6. write (B);
```

  - **Απαίτηση συνέπειας (consistency)**: το άθροισμα των Α και Β δεν αλλάζει με την εκτέλεση της δοσοληψίας.
  - **Απαίτηση ατομικότητας (atomicity)**: εάν η δοσοληψία αποτύχει μετά το βήμα 3 και πριν το βήμα 6, το σύστημα πρέπει να διαβεβαιώνει ότι οι αλλαγές που έγιναν δεν αντανακλώνται στη ΒΔ, αλλιώς θα οδηγηθούμε σε ασυνεπή ΒΔ.
  - **Απαίτηση ανθεκτικότητας (durability)**: μόλις ο χρήστης ενημερωθεί ότι η δοσοληψία έχει ολοκληρωθεί (δηλ., έχει γίνει η μεταφορά των 50€), οι αλλαγές στη ΒΔ πρέπει να παραμείνουν παρά τις οποιεσδήποτε αποτυχίες.
  - ...

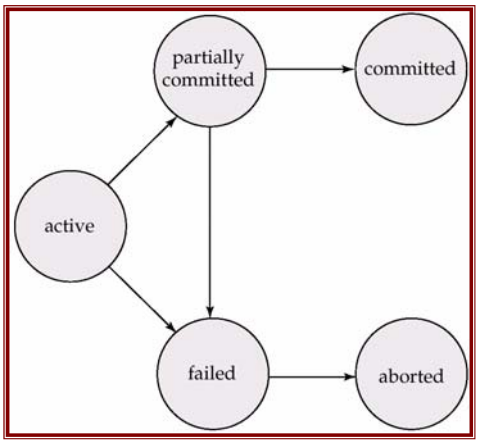
Παράδειγμα μεταφοράς κεφαλαίου (2)

- (συν.)
  - **Απαίτηση απομόνωσης (isolation)**: εάν μεταξύ των βημάτων 3 και 6, επιτραπεί σε μια άλλη δοσοληψία να προσπελάσει τη μερικώς ενημερωμένη ΒΔ, εκείνη θα δει μια ασυνεπή ΒΔ (το άθροισμα A + B θα είναι μικρότερο από όσο θα έπρεπε να είναι).
    - Μπορούμε να εγγυηθούμε την απομόνωση, αν εκτελέσουμε τις δοσοληψίες σειριακά, δηλαδή τη μια μετά την άλλη. Παρ' όλα αυτά, η συνδρομική εκτέλεση πολλαπλών δοσοληψιών έχει σημαντικά πλεονεκτήματα, όπως θα δούμε.

1. read (A);
2. A := A – 50;
3. write (A);
4. read (B);
5. B := B + 50;
6. write (B);

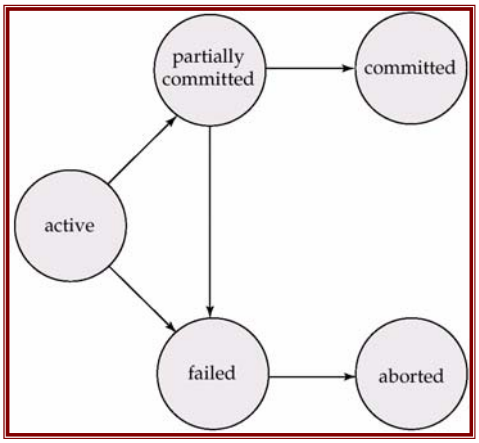
Καταστάσεις μιας δοσοληψίας (1)

- **Ενεργή (Active)**, η αρχική κατάσταση. Η δοσοληψία παραμένει σε αυτήν την κατάσταση όσο εκτελείται.
- **Μερικώς επικυρωμένη (Partially committed)**, μετά την εκτέλεση και της τελευταίας εντολής.
- **Αποτυχημένη (Failed)**, μετά τη διαπίστωση ότι δε μπορεί να προχωρήσει η κανονική εκτέλεση.
- ...



Καταστάσεις μιας δοσοληψίας (2)

- ...
- **Ακυρωμένη (Aborted)**, αφού έχει υποχωρήσει η δοσοληψία (rollback) και η ΒΔ έχει επιστρέψει στην κατάσταση που ήταν πριν την έναρξη της δοσοληψίας. Υπάρχουν τώρα δυο δυνατότητες:
  - Επανέναρξη της δοσοληψίας – μόνο αν δεν υπάρχει κάποιο εσωτερικό λογικό λάθος
  - «Θανάτωση» της δοσοληψίας.
- **Επικυρωμένη (Committed)**, μετά την επιτυχή ολοκλήρωση.



- Ορισμός της δοσοληψίας
- Συνδρομικές εκτελέσεις (concurrency)
  - Έλεγχος σειριοποιησιμότητας
- Ανάκαμψη δοσοληψιών (recovery)
  - Υλοποίηση της Απομόνωσης

Συνδρομικές εκτελέσεις

- Το ΣΔΒΔ πρέπει να επιτρέπει τη συνδρομική (concurrent) εκτέλεση πολλαπλών δοσοληψιών. Τα πλεονεκτήματα είναι:
  - Καλύτερη εκμετάλλευση των πόρων του Η/Υ (➔ καλύτερη ρυθμαπόδοση - throughput): μια δοσοληψία μπορεί να χρησιμοποιεί τη CPU την ώρα που μια άλλη διαβάζει από ή γράφει στο δίσκο.
  - Μείωση του μέσου χρόνου απόκρισης για δοσοληψίες: οι μικρές δοσοληψίες δε χρειάζεται να περιμένουν να ολοκληρωθούν οι μεγάλες.
- **Σχήματα ελέγχου συνδρομικότητας**: μηχανισμοί για την επίτευξη απομόνωσης, δηλ, μηχανισμοί ελέγχου της αλληλεπίδρασης μεταξύ των συνδρομικών δοσοληψιών με στόχο την αποφυγή της καταστροφής της συνέπειας της ΒΔ.

Χρονοπρογράμματα

- **Χρονοπρογράμματα (Schedules)**: ακολουθίες που δηλώνουν τη χρονολογική σειρά με την οποία εκτελούνται εντολές από συνδρομικές δοσοληψίες
- Ένα χρονοπρόγραμμα για ένα σύνολο από δοσοληψίες πρέπει:
  - να αποτελείται από το σύνολο των εντολών αυτών των δοσοληψιών
  - να διατηρεί τη σειρά με την οποία εμφανίζονται οι εντολές σε κάθε δοσοληψία ξεχωριστά.

Παράδειγμα χρονοπρογράμματος (1)

- Έστω η T_1 μεταφέρει 50€ από το A στο B και η T_2 αφαιρεί το 10% του A και το μεταφέρει στο B.
- Το χρονοπρόγραμμα S_1 είναι **σειριακό**, με την T_1 να προηγείται της T_2.

S_1

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

Παράδειγμα χρονοπρογράμματος (2)

- Έστω T_1 και T_2 οι δοσοληψίες που ορίστηκαν προηγουμένως.
- Το χρονοπρόγραμμα S_2 **δεν** είναι σειριακό, αλλά παρόλα αυτά είναι ισοδύναμο με το (σειριακό) S_1.
  - Τόσο στο S_1 όσο και στο S_2, διατηρείται το άθροισμα A + B

S_2

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

Παράδειγμα χρονοπρογράμματος (3)

- Αντίθετα, το χρονοπρόγραμμα S_3 **δεν** είναι ισοδύναμο με το S_1
  - Αλλοιώνεται η τιμή του αθροίσματος A + B.

S_3

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Σειριοποιησιμότητα

- Βασική παραδοχή: κάθε δοσοληψία (από μόνη της) διατηρεί τη συνέπεια της ΒΔ.
- Συνεπώς, η σειριακή εκτέλεση ενός συνόλου δοσοληψιών διατηρεί τη συνέπεια της ΒΔ.
- Ένα χρονοπρόγραμμα είναι **σειριοποιήσιμο (serializable)** εάν είναι ισοδύναμο με ένα σειριακό χρονοπρόγραμμα.
- Άρα, στόχος είναι να επιτευχθούν σειριοποιήσιμα χρονοπρογράμματα, ώστε να διατηρείται τη συνέπεια της ΒΔ.
  - Πρόβλημα: να αποφανθούμε εάν ένα χρονοπρόγραμμα είναι σειριοποιήσιμο
- Στη συνέχεια θα μελετήσουμε ένα είδος ισοδυναμίας χρονοπρογραμμάτων, τη **σειριοποιησιμότητα βάσει συγκρούσεων** (conflict serializability)

Σειριοποιησιμότητα βάσει συγκρούσεων (1)

- Οι εντολές I και J των δοσοληψιών T_i και T_j αντιστοίχως, **συγκρούονται** εάν και μόνο αν υπάρχει κάποιο αντικείμενο Q που προσπελαύνεται τόσο από την I όσο και από την J, και τουλάχιστον μια από αυτές τις εντολές γράφει στο Q.
  - I = read(Q), J = read(Q). I και J δεν συγκρούονται.
  - I = read(Q), J = write(Q). Συγκρούονται.
  - I = write(Q), J = read(Q). Συγκρούονται.
  - I = write(Q), J = write(Q). Συγκρούονται.
- Διαισθητικά, μια σύγκρουση μεταξύ των I και J αναγκάζει μια (λογική) προσωρινή αναδιάταξη μεταξύ τους. Εάν οι I και J είναι συνεχόμενες σε ένα χρονοπρόγραμμα και δεν συγκρούονται, τα αποτελέσματά τους θα παρέμεναν τα ίδια ακόμα κι αν είχαν ανταλλάξει θέσεις στο χρονοπρόγραμμα.

Σειριοποιησιμότητα βάσει συγκρούσεων (2)

- Εάν ένα χρονοπρόγραμμα S μπορεί να μετασχηματιστεί σε ένα χρονοπρόγραμμα S΄ με μια σειρά από αντιμεταθέσεις μη συγκρουόμενων εντολών, λέμε ότι τα S και S΄ είναι **ισοδύναμα βάσει συγκρούσεων (conflict equivalent)**.
- Λέμε ότι ένα χρονοπρόγραμμα S είναι **σειριοποιήσιμο βάσει συγκρούσεων (conflict serializable)**, εάν είναι ισοδύναμο βάσει συγκρούσεων με ένα σειριακό χρονοπρόγραμμα.
  - Παράδειγμα χρονοπρογράμματος που **δεν** είναι σειριοποιήσιμο βάσει συγκρούσεων:

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

Σειριοποιησιμότητα βάσει συγκρούσεων (3)

- Το (μη σειριακό) χρονοπρόγραμμα S_2 μπορεί να μετασχηματιστεί στο (σειριακό) S_1, με μια σειρά από αντιμεταθέσεις μη συγκρουόμενων εντολών.
- Επομένως, το S_2 είναι σειριοποιήσιμο βάσει συγκρούσεων.

S_1

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

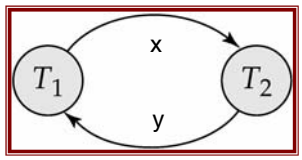
S_2

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

Έλεγχος σειριοποιησιμότητας (1)

- Έστω χρονοπρόγραμμα S ενός συνόλου δοσοληψιών T_1, T_2, ..., T_n
- **Γράφος προβαδίσματος (precedence graph)** — ένας κατευθυνόμενος γράφος όπου οι κορυφές είναι οι δοσοληψίες και οι ακμές φανερώνουν συγκρούσεις μεταξύ δοσοληψιών.
- Αλγόριθμος κατασκευής του γράφου προβαδίσματος:
  1. Για κάθε δοσοληψία T_i που συμμετέχει στο χρονοπρόγραμμα S, δημιούργησε έναν κόμβο T_i στο γράφο προτεραιοτήτων.
  2. Για κάθε περίπτωση στο S όπου η T_j εκτελεί μια πράξη
     - read(X) μετά από μια εντολή write(X)
     - write(X) μετά από μια εντολή read(X)
     - write(X) μετά από μια εντολή write(X)

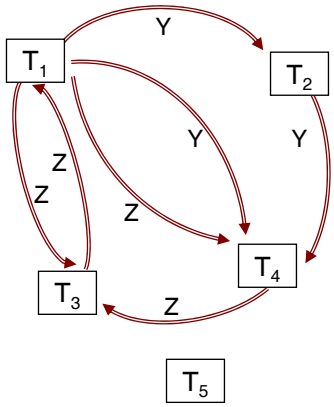
     ... που εκτελείται από την T_i δημιούργησε μια ακμή (T_i → T_j) στο γράφο προτεραιοτήτων.



Έλεγχος σειριοποιησιμότητας (2)

T_1	T_2	T_3	T_4	T_5
read(Y) read(Z)	read(X)			read(V) read(W) write(W)
	read(Y) write(Y)	read(Z)		
read(U)			read(Y) write(Y) read(Z)	
		write(Z)		
write(Z) write(U)				

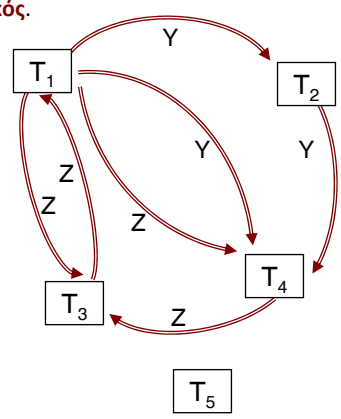
Γράφος προβαδίσματος για το χρονοπρόγραμμα S



Έλεγχος σειριοποιησιμότητας (3)

- **Σειριοποιησιμότητα βάσει συγκρούσεων**
  - Ένα χρονοπρόγραμμα είναι σειριοποιήσιμο βάσει συγκρούσεων αν και μόνο αν ο αντίστοιχος γράφος προβαδίσματος είναι **ακυκλικός**.
    - Στο διπλανό παράδειγμα, υπάρχει κύκλος: $T_1 \rightarrow T_3 \rightarrow T_1$
  - Οι συνήθεις αλγόριθμοι εντοπισμού κύκλων απαιτούν χρόνο $O(n^2)$, όπου n είναι ο αριθμός των κορυφών του γράφου.
  - Εάν ο γράφος προβαδίσματος είναι ακυκλικός, η σειρά σειριοποιησιμότητας μπορεί να βρεθεί με μια τοπολογική ταξινόμηση του γράφου.
  - Στο διπλανό παράδειγμα (αν αφαιρέσουμε την ακμή $T_3 \rightarrow T_1$ που δημιουργεί τον κύκλο):

    $T_5 \rightarrow T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3$



Έλεγχος σειριοποιησιμότητας (4)

- Σχόλιο πάνω στη σειριοποιησιμότητα βάσει συγκρούσεων
  - Για να γίνει ο παραπάνω έλεγχος σειριοποιησιμότητας πρέπει να γνωρίζουμε το χρονοπρόγραμμα (ποιες δοσοληψίες θα εμπλακούν σε συνδρομική εκτέλεση και πώς).
    - Πρόβλημα: αυτό δεν είναι εφικτό γιατί η διαχείριση της ΒΔ γίνεται δυναμικά. Άρα δεν γνωρίζουμε το χρονοπρόγραμμα που θα προκύψει κάθε στιγμή
- Πρακτικά, στόχος δεν είναι ο έλεγχος της σειριοποιησιμότητας αλλά η **αποφυγή της μη σειριοποιησιμότητας**.
  - Πώς το επιτυγχάνουμε αυτό; με χρήση κατάλληλων **κλειδωμάτων**.

Συνδρομικότητα βασισμένη σε κλειδιά (1)

- **Κλείδωμα** είναι ένας μηχανισμός για τον έλεγχο της συνδρομικής προσπέλασης σε ένα αντικείμενο δεδομένων.
- Τα αντικείμενα μιας ΒΔ μπορούν να κλειδωθούν με δυο τρόπους:
  - **Αποκλειστικός - exclusive (X) τρόπος**. Το αντικείμενο μπορεί και να διαβαστεί και να γραφεί. Το κλειδί γραφής X-lock ζητείται με χρήση της lock-X αίτησης.
  - **Διαμοιραζόμενος - shared (S) τρόπος**. Το αντικείμενο μπορεί μόνο να διαβαστεί. Το κλειδί ανάγνωσης S-lock ζητείται με χρήση της lock-S αίτησης.
- Οι αιτήσεις κλειδώματος γίνονται από το διαχειριστή ελέγχου συνδρομικότητας. Μια δοσοληψία μπορεί να προχωρήσει μόνο αφού ικανοποιηθεί η αίτηση κλειδώματος που έχει κάνει.

Συνδρομικότητα βασισμένη σε κλειδιά (2)

- Πίνακας συμβατότητας κλειδωμάτων

	S	X
S	true	false
X	false	false

- Σε μια δοσοληψία μπορεί να παραχωρηθεί ένα κλειδί, εάν αυτό είναι συμβατό με κλειδιά που έχουν ήδη παραχωρηθεί (σε άλλες δοσοληψίες).
- Πολλές δοσοληψίες μπορούν να διατηρούν κλειδιά S για ένα αντικείμενο, αλλά αν μια δοσοληψία πάρει κλειδί X, σε καμιά άλλη δοσοληψία δεν μπορεί να παραχωρηθεί κλειδί (είτε S είτε X).
- Εάν δεν μπορεί να παραχωρηθεί κλειδί σε μια δοσοληψία, αυτή περιμένει έως ότου ελευθερωθούν όλα τα μη συμβατά κλειδιά που διατηρούνται από άλλες δοσοληψίες. Στη συνέχεια τής παραχωρείται το κλειδί που έχει ζητήσει.

Συνδρομικότητα βασισμένη σε κλειδιά (3)

- Παράδειγμα εκτέλεσης κλειδώματος από μια δοσοληψία:

- Ένα τέτοιο σχήμα απόδοσης κλειδιών **δεν** εγγυάται σειριοποιησιμότητα
  - εάν το A και το B ενημερωθούν στο χρονικό διάστημα ανάμεσα στις εντολές unlock() και display(), το άθροισμα A+B που θα απεικονιστεί θα είναι λάθος !!
- Άρα τα κλειδώματα / ξεκλειδώματα πρέπει να γίνουν με κάποιο έξυπνο τρόπο

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

Πρωτόκολλο κλειδώματος 2 φάσεων (1)

- Εγγυάται χρονοπρογράμματα σειριοποιήσιμα βάσει συγκρούσεων και αποτελείται από 2 φάσεις (**2 phase locking – 2PL**)
- **Φάση 1 (εξάπλωσης)**
  - Η δοσοληψία μπορεί να δεσμεύει κλειδιά αλλά δεν μπορεί να ελευθερώσει κλειδιά.
- **Φάση 2 (συρρίκνωσης)**
  - Η δοσοληψία μπορεί να ελευθερώνει κλειδιά αλλά δεν μπορεί να δεσμεύσει κλειδιά.
- Το πρωτόκολλο 2PL **εγγυάται** σειριοποιησιμότητα. Μπορεί να αποδειχθεί ότι οι δοσοληψίες μπορούν να σειριοποιηθούν με τη σειρά των σημείων κλειδώματός τους - lock points
  - Σημείο κλειδώματος λέγεται το χρονικό σημείο στο οποίο η δοσοληψία απαίτησε το τελευταίο της κλειδί.

Πρωτόκολλο κλειδώματος 2 φάσεων (2)

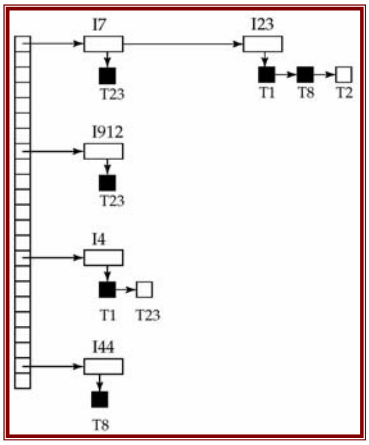
- Το πρωτόκολλο 2PL **δεν** εγγυάται την αποτροπή αδιεξόδων.
- Είναι πιθανό να οδηγηθούμε σε διαδοχικές υποχωρήσεις (cascading roll-back). Για την αποφυγή αυτού του ενδεχομένου, υπάρχουν δύο παραλλαγές του πρωτοκόλλου 2PL:
  - **«αυστηρό» πρωτόκολλο δυο φάσεων (strict 2PL)**: μια δοσοληψία πρέπει να διατηρήσει όλα τα κλειδιά X που έχει δεσμεύσει έως το τέλος της (που έρχεται με την επικύρωση ή ακύρωσή της).
  - **«άκαμπτο» (δηλ. «ακόμη πιο αυστηρό») πρωτόκολλο δυο φάσεων (rigorous 2PL)**: μια δοσοληψία πρέπει να διατηρήσει όλα τα κλειδιά (S ή X) που έχει δεσμεύσει έως το τέλος της.
    - Σε αυτό το πρωτόκολλο οι δοσοληψίες μπορούν να σειριοποιηθούν με τη σειρά που επικυρώνονται.

Υλοποίηση των κλειδωμάτων (1)

- Ο **Διαχειριστής κλειδωμάτων** (lock administrator) αποτελεί μια ξεχωριστή διαδικασία στην οποία οι δοσοληψίες στέλνουν αιτήσεις κλειδώματος και ξεκλειδώματος.
  - Ο διαχειριστής κλειδωμάτων απαντά σε μια αίτηση κλειδώματος στέλνοντας ένα μήνυμα παραχώρησης κλειδιού (ή ένα μήνυμα ζητώντας από τη δοσοληψία να κάνει rollback, στην περίπτωση αδιεξόδου).
  - Η αιτούσα δοσοληψία περιμένει έως ότου απαντηθεί η αίτησή της.
- Ο διαχειριστής κλειδωμάτων συντηρεί μια δομή δεδομένων, που λέγεται **πίνακας κλειδιών (lock table)**, για την καταγραφή των παραχωρημένων κλειδιών και των εκκρεμών αιτήσεων.
  - Ο πίνακας κλειδωμάτων υλοποιείται συνήθως ως ένας πίνακας κατακερματισμού κύριας μνήμης με δεικτοδότηση στο όνομα του αντικειμένου δεδομένων που κλειδώνεται.

Υλοποίηση των κλειδωμάτων (2)

- **Πίνακας Κλειδωμάτων**
  - Τα μαύρα τετράγωνα δηλώνουν παραχωρημένα κλειδιά, τα άσπρα τετράγωνα δηλώνουν αιτήσεις που περιμένουν.
  - Κάθε νέα αίτηση κλειδώματος προστίθεται στο τέλος της ουράς των αιτήσεων και το ζητούμενο κλειδί παραχωρείται αν είναι συμβατό με όλα τα προηγούμενα κλειδιά.
  - Μια αίτηση ξεκλειδώματος επιδρά στην αντίστοιχη αίτηση κλειδώματος (διαγράφεται) και στις εκκρεμείς αιτήσεις (ελέγχεται το εάν μπορούν να ικανοποιηθούν).



Προβλήματα λόγω των κλειδωμάτων (1)

- Θεωρήστε το ακόλουθο «παράθυρο» ενός χρονοπρογράμματος
  - Ούτε η T_3 ούτε η T_4 μπορούν να προχωρήσουν
  - η εκτέλεση του lock-S(B) αναγκάζει την T_4 να περιμένει την T_3 να ελευθερώσει το κλειδί της στο B
  - η εκτέλεση του lock-X(A) αναγκάζει την T_3 να περιμένει την T_4 να ελευθερώσει το κλειδί της στο A.

S_4

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

- Μια τέτοια κατάσταση ονομάζεται **αδιέξοδο (deadlock)**.
  - Για την αντιμετώπιση του αδιεξόδου, μια από τις T_3 και T_4 πρέπει να υποχωρήσει (**rollback**), ώστε τα κλειδιά που κατέχει να ελευθερωθούν.

Προβλήματα λόγω των κλειδωμάτων (2)

- Το ενδεχόμενο για αδιέξοδα δεν μπορεί να αποκλειστεί. Τα αδιέξοδα είναι ένα αναγκαίο κακό.
- **Λιμοκτονία (starvation)** μπορεί επίσης να προκύψει εάν ο διαχειριστής ελέγχου συνδρομικότητας είναι κακοσχεδιασμένος. Για παράδειγμα:
  - Μια δοσοληψία μπορεί να περιμένει για ένα X-lock σε ένα αντικείμενο, ενώ μια ακολουθία από άλλες δοσοληψίες αιτούν και αποκτούν ένα S-lock στο ίδιο αντικείμενο.
  - Η ίδια δοσοληψία ακυρώνεται (rollback) κατ' επανάληψη εξαιτίας αδιεξόδων.
- Ο διαχειριστής ελέγχου συνδρομικότητας μπορεί να σχεδιαστεί έτσι ώστε να αποτρέπει τις καταστάσεις λιμοκτονίας.

- Ορισμός της δοσοληψίας
- Συνδρομικές εκτελέσεις (concurrency)
  - Έλεγχος σειριοποιησιμότητας
- Ανάκαμψη δοσοληψιών (recovery)
  - Υλοποίηση της Απομόνωσης

Ανάκαμψη δοσοληψιών (1)

- Ανάγκη αντιμετώπισης της επίδρασης των αποτυχιών των δοσοληψιών στην περίπτωση συνδρομικών δοσοληψιών.
- **Ανακάμψιμο χρονοπρόγραμμα** — εάν μια δοσοληψία T_j διαβάζει ένα αντικείμενο που έχει προηγουμένως γραφτεί από μια δοσοληψία T_i , η λειτουργία επικύρωσης (**commit**) της T_i πρέπει να προηγείται της λειτουργίας επικύρωσης της T_j.
  - Το χρονοπρόγραμμα S_5 είναι μη ανακάμψιμο εάν η T_9 επικυρωθεί (commit) αμέσως μετά την εντολή read(A).
  - Εάν στη συνέχεια ακυρωθεί η T_8, η T_9 θα έχει διαβάσει μια ασυνεπή κατάσταση της ΒΔ.

S_5

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

- Τα ΣΔΒΔ πρέπει να εξασφαλίζουν ότι τα χρονοπρογράμματα είναι ανακάμψιμα.

Ανάκαμψη δοσοληψιών (2)

- **Διαδοχικές υποχωρήσεις (Cascading rollbacks)** – η αποτυχία μίας δοσοληψίας οδηγεί σε μια σειρά από υποχωρήσεις (rollbacks) και άλλων δοσοληψιών.
  - Ας θεωρήσουμε το ακόλουθο χρονοπρόγραμμα όπου καμιά δοσοληψία δεν έχει ακόμα επικυρωθεί (συνεπώς το χρονοπρόγραμμα είναι ανακάμψιμο).
  - Εάν η T_{10} αποτύχει, τότε οι T_{11} και T_{12} πρέπει επίσης να κάνουν rollback.
  - Τέτοιου είδους σενάρια μπορεί να οδηγήσουν στην ακύρωση αρκετής δουλειάς που έχει ήδη γίνει
- Ένα επιθυμητό χαρακτηριστικό είναι να έχουμε **χρονοπρογράμματα χωρίς διαδοχικές υποχωρήσεις (Cascadeless)**

S_6

T_{10}	T_{11}	T_{12}
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)

Αλγόριθμοι ανάκαμψης

- Οι **αλγόριθμοι ανάκαμψης (recovery)** είναι τεχνικές που εγγυώνται τη συνέπεια της ΒΔ και την ατομικότητα και την ανθεκτικότητα μιας δοσοληψίας παρά τις οποιεσδήποτε αποτυχίες.
- Οι αλγόριθμοι ανάκαμψης έχουν δυο μέρη:
  - Εργασίες που εκτελούνται **κατά τη διάρκεια της εκτέλεσης των δοσοληψιών** για να εγγυηθούν την ύπαρξη επαρκούς πληροφορίας για την ανάκαμψη από αποτυχίες.
  - Εργασίες που εκτελούνται **μετά από μια αποτυχία για την ανάκαμψη των περιεχομένων της ΒΔ** σε μια κατάσταση που εγγυάται την ατομικότητα, συνέπεια και ανθεκτικότητα.

Δομή αποθήκευσης

- **Προσωρινή (volatile) αποθήκευση**:
  - Δεν επιβιώνει μετά από πτώσεις συστήματος.
  - Παραδείγματα: κύρια μνήμη, μνήμη cache.
- **Μόνιμη (non-volatile) αποθήκευση**:
  - Επιβιώνει μετά από πτώσεις συστήματος (αλλά όχι από άλλες αποτυχίες π.χ. δίσκου).
  - Παραδείγματα: δίσκοι, ταινίες, μνήμη flash.
- **Ακλόνητη (stable) αποθήκευση**:
  - Μια "ιδανική" μορφή αποθήκευσης που επιβιώνει μετά από όλες τις αποτυχίες.
  - Προσεγγίζεται μέσω πολλαπλών αντιγράφων σε διακριτά μόνιμα μέσα.

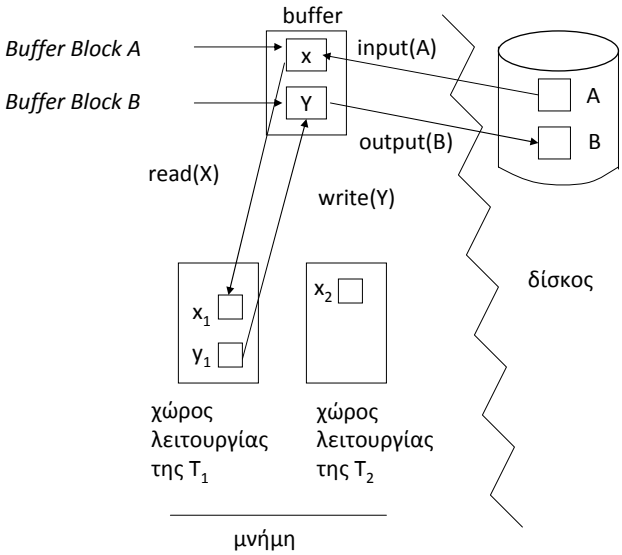
Προσπέλαση δεδομένων (1)

- Τα φυσικά blocks είναι εκείνα τα blocks που βρίσκονται στο δίσκο.
- Τα **blocks ενδιάμεσης μνήμης (buffer blocks)** είναι εκείνα τα blocks που βρίσκονται προσωρινά στην κύρια μνήμη.
  - Οι μετακινήσεις των blocks μεταξύ της κυρίας μνήμης και του δίσκου ξεκινούν μέσω των ακόλουθων δυο λειτουργιών:
  - input(B): μεταφέρει το φυσικό block B στην κύρια μνήμη.
  - output(B): μεταφέρει το block ενδιάμεσης μνήμης B στο δίσκο και αντικαθιστά εκεί το κατάλληλο φυσικό block.
- Κάθε δοσοληψία T_i έχει τη δική της ιδιωτική περιοχή λειτουργίας, στην οποία φυλάσσονται τοπικά αντίγραφα από όλα τα αντικείμενα δεδομένων που προσπελαύνονται και ενημερώνονται από αυτήν.

Προσπέλαση δεδομένων (2)

- Η δοσοληψία μεταφέρει αντικείμενα μεταξύ των blocks ενδιάμεσης μνήμης του συστήματος και της ιδιωτικής περιοχής λειτουργίας της (στη μνήμη) με χρήση των ακόλουθων λειτουργιών:
  - read(X): αναθέτει την τιμή του αντικειμένου δεδομένων X στην τοπική μεταβλητή x.
  - write(X): αναθέτει την τιμή της τοπικής μεταβλητής x στο αντικείμενο X στο block ενδιάμεσης μνήμης.
- Οι δοσοληψίες εκτελούν την read(X) όταν προσπελαύνουν το X για πρώτη φορά.
  - Όλες οι επακόλουθες προσπελάσεις γίνονται στο τοπικό αντίγραφο.
- Μετά την τελευταία προσπέλαση, η δοσοληψία εκτελεί την write(X).

Παράδειγμα προσπέλασης δεδομένων



Ανάκαμψη και ατομικότητα (1)

- Η τροποποίηση της ΒΔ χωρίς εγγύηση ότι η δοσοληψία θα επικυρωθεί μπορεί να αφήσει τη ΒΔ σε μια ασυνεπή κατάσταση.
- Παράδειγμα δοσοληψίας: μεταφορά 50€ από το λογαριασμό Α στο λογαριασμό Β.
  - Ο στόχος είναι είτε να πραγματοποιηθούν είτε όλες οι απαιτούμενες τροποποιήσεις στη ΒΔ ή καμιά από αυτές.
  - Ασυνέπεια προκύπτει όταν έχει γίνει τουλάχιστον μια από αυτές τις τροποποιήσεις στη ΒΔ, αλλά όμως δεν έχουν γίνει όλες.

Ανάκαμψη και ατομικότητα (2)

- Για να εγγυηθούμε για την ατομικότητα παρά τις πιθανές αποτυχίες, εξάγουμε πρώτα πληροφορία που περιγράφει τις τροποποιήσεις στην ακλόνητη αποθήκευση χωρίς να τροποποιείται η ίδια η ΒΔ.
- Μελετάμε την εξής προσέγγιση:
  - Ανάκαμψη βασισμένη στο ημερολόγιο του συστήματος (Log)
- Υποθέτουμε ότι οι δοσοληψίες εκτελούνται σειριακά, δηλαδή η μια μετά την άλλη.

Ανάκαμψη βασισμένη στο ημερολόγιο συστήματος (1)

- Ένα **ημερολόγιο συστήματος (log)** φυλάσσεται στην ακλόνητη αποθήκευση.
  - Το ημερολόγιο συστήματος είναι μια ακολουθία από εγγραφές ημερολογίου και συντηρεί μια καταγραφή των ενημερώσεων που έγιναν στη ΒΔ.
- Διαδικασία:
  - Όταν μια δοσοληψία T_i ξεκινά, καταχωρεί τον εαυτό της στο ημερολόγιο γράφοντας μια <T_i start> εγγραφή ημερολογίου.
  - Πριν η T_i εκτελέσει την λειτουργία write(X), προστίθεται μια εγγραφή στο ημερολόγιο <T_i, X, V_1, V_2>, όπου V_1 είναι η προηγούμενη τιμή του X (πριν εκτελεστεί η write) και V_2 είναι η τιμή που πρόκειται να γραφεί στο X.
  - Όταν η T_i ολοκληρώσει και την τελευταία της εντολή, γράφεται στο ημερολόγιο η εγγραφή <T_i commit>.

Ανάκαμψη βασισμένη στο ημερολόγιο συστήματος (2)

- Υποθέτουμε (για την ώρα) ότι οι εγγραφές ημερολογίου γράφονται απευθείας στην ακλόνητη αποθήκευση (δηλαδή δεν τοποθετούνται σε έναν buffer).
- Δυο προσεγγίσεις με χρήση ημερολογίου:
  - Ετεροχρονισμένη τροποποίηση της ΒΔ.
  - Άμεση τροποποίηση της ΒΔ.

Ετεροχρονισμένη τροποποίηση της ΒΔ (1)

- Το σχήμα ετεροχρονισμένης τροποποίησης μιας ΒΔ καταγράφει όλες τις τροποποιήσεις στο ημερολόγιο, αλλά καθυστερεί όλες τις γραφές στη ΒΔ έως ότου η δοσοληψία επικυρωθεί μερικώς.
  - Υποθέτει ότι οι δοσοληψίες εκτελούνται σειριακά.
  - Η δοσοληψία ξεκινά γράφοντας την εγγραφή <T_i start> στο ημερολόγιο.
  - Μια λειτουργία write(X) προκαλεί την προσθήκη στο ημερολόγιο μιας εγγραφής <T_i, X, V>, όπου V είναι η νέα τιμή για το X.
    - Σημείωση: η παλιά τιμή δε χρειάζεται σ' αυτό το σχήμα.
  - Η γραφή στο X δε γίνεται αυτή τη στιγμή, αλλά αναβάλλεται.
  - Όταν η T_i επικυρώνεται μερικώς, η εγγραφή <T_i commit> γράφεται στο ημερολόγιο.
  - Τέλος, οι εγγραφές του ημερολογίου διαβάζονται και χρησιμοποιούνται για την πραγματική εκτέλεση των προηγούμενων γραφών που έχουν αναβληθεί.

Ετεροχρονισμένη τροποποίηση της ΒΔ (2)

- Κατά τη διάρκεια της ανάκαμψης μετά από πτώση συστήματος, μια δοσοληψία χρειάζεται να εκτελεστεί ξανά αν και μόνο αν και η <T_i start> και η <T_i commit> υπάρχουν στο ημερολόγιο.
- Η εκ νέου εκτέλεση μιας δοσοληψίας T_i (redo T_i) θέτει την τιμή όλων των αντικειμένων δεδομένων που ενημερώνονται από τη δοσοληψία στις νέες τιμές.
- Αποτυχίες συστήματος μπορούν να συμβούν ενώ:
  - Η δοσοληψία εκτελεί τις αρχικές ενημερώσεις, ή
  - Εκτελείται η διαδικασία ανάκαμψης (λόγω προηγούμενης αποτυχίας συστήματος).
- Παράδειγμα (η T_1 εκτελείται πριν την T_2):

T_1	T_2
read(A) A := A – 50 write(A) read(B) B := B + 50 write(B)	
	read(C) C := C – 100 write(C)

Ετεροχρονισμένη τροποποίηση της ΒΔ (3)

- Παράδειγμα: το ημερολόγιο συστήματος σε τρεις χρονικές στιγμές:

(a)	(b)	(c)
$<T_0$ start>	$<T_0$ start>	$<T_0$ start>
$<T_0, A, 950>$	$<T_0, A, 950>$	$<T_0, A, 950>$
$<T_0, B, 2050>$	$<T_0, B, 2050>$	$<T_0, B, 2050>$
	$<T_0$ commit>	$<T_0$ commit>
	$<T_1$ start>	$<T_1$ start>
	$<T_1, C, 600>$	$<T_1, C, 600>$
		$<T_1$ commit>

- Εάν συμβεί αποτυχία στην περίπτωση:
  - (a) δε χρειάζεται να γίνει καμία επανεκτέλεση, απλά διαγράφονται οι 3 εγγραφές.
  - (b) η redo(T_0) πρέπει να εκτελεστεί αφού υπάρχει η $<T_0$ commit> (όχι όμως και η T_1 , απλά διαγράφονται οι 2 τελευταίες εγγραφές).
  - (c) τότε η redo(T_0) πρέπει να εκτελεστεί ακολουθούμενη από την redo(T_1) αφού υπάρχουν οι $<T_0$ commit> και $<T_1$ commit>.

Άμεση τροποποίηση της ΒΔ (1)

- Το σχήμα άμεσης τροποποίησης μιας ΒΔ επιτρέπει οι ενημερώσεις από μια μη επικυρωμένη δοσοληψία να γίνονται αμέσως μόλις εκτελεστούν οι αντίστοιχες εντολές γραφής.
  - Αφού μπορεί να χρειαστεί να γίνει κάποια ακύρωση, τα ενημερωμένα ημερολόγια πρέπει να έχουν **τόσο την παλιά όσο και τη νέα τιμή**.
- Η ενημέρωση της εγγραφής του ημερολογίου πρέπει να γίνει **πριν** ενημερωθεί το αντικείμενο της ΒΔ.
  - Υποθέτουμε ότι η εγγραφή του ημερολογίου εξάγεται αμέσως στην ακλόνητη αποθήκευση.
- Η έξοδος των ενημερωμένων blocks στο δίσκο (εντολή output) μπορεί να γίνει οποιαδήποτε στιγμή (πριν/μετά την επικύρωση μιας δοσοληψίας).
- Η σειρά με την οποία εξάγονται τα blocks στο δίσκο (εντολή output) μπορεί να είναι διαφορετική από τη σειρά με την οποία ενημερώνονται στην ενδιάμεση μνήμη (εντολή write).

Άμεση τροποποίηση της ΒΔ (2)

Ημερολόγιο	Ενδιάμεση μνήμη	Δίσκος
$<T_0$ start>		
$<T_0$, A, 1000, 950>		
$<T_0$, B, 2000, 2050>		
	write (A) // 950	
	write (B) // 2050	
$<T_0$ commit>		
$<T_1$ start>		
$<T_1$, C, 700, 600>		
	write (C) // 600	
		output (B_B)
		output (B_C)
$<T_1$ commit>		
		output (B_A)

Σημείωση:

το B_x δηλώνει το block που περιέχει το X.

Άμεση τροποποίηση της ΒΔ (3)

- Η διαδικασία ανάκαμψης έχει δυο λειτουργίες:
  - undo(T_i): επαναφέρει όλα τα αντικείμενα που ενημερώθηκαν από την T_i στις παλιές τους τιμές, πηγαίνοντας προς τα πίσω από την τελευταία προς την πρώτη εγγραφή του ημερολογίου για την T_i
  - redo(Ti): θέτει όλα τα αντικείμενα που ενημερώθηκαν από την T_i στις νέες τιμές, πηγαίνοντας προς τα εμπρός από την πρώτη προς την τελευταία εγγραφή του ημερολογίου για την T_i
- Κατά την ανάκαμψη μετά από αποτυχία:
  - η δοσοληψία T_i χρειάζεται να ακυρωθεί (**εντολή undo**), εάν το ημερολόγιο περιέχει την εγγραφή <T_i start>, αλλά δεν περιέχει την εγγραφή <T_i commit>.
  - η δοσοληψία T_i χρειάζεται να επανεκτελεστεί (**εντολή redo**), εάν το ημερολόγιο περιέχει και την εγγραφή <T_i start> και την εγγραφή <T_i commit>.
- Κατά τη διαδικασία ανάκαμψης, οι ακυρώσεις δοσοληψιών προηγούνται των επανεκτελέσεων

Άμεση τροποποίηση της ΒΔ (4)

- Παράδειγμα: το ημερολόγιο συστήματος σε τρεις χρονικές στιγμές:

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 700, 600>	<T_1, C, 700, 600>
		<T_1 commit>

(a) undo (T_0): Το Β επαναφέρεται σε 2000 και το Α σε 1000.

(b) undo (T_1) και redo (T_0): Το C επαναφέρεται στο 700 και μετά τα Α και Β γίνονται 950 και 2050, αντιστοίχως.

(c) redo (T_0) και redo (T_1): Τα Α και Β γίνονται 950 και 2050, αντιστοίχως. Στη συνέχεια, το C γίνεται 600.

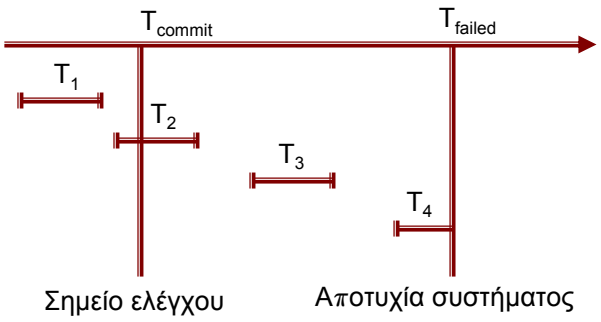
Σημεία ελέγχου (1)

- Η διαδικασία ανάκαμψης που συζητήσαμε προηγουμένως έχει τα εξής προβλήματα:
  - Η σάρωση όλου του ημερολογίου είναι χρονοβόρα.
  - Μπορεί χωρίς λόγο να εκτελέσουμε εκ νέου δοσοληψίες οι οποίες ήδη έχουν εξάγει τα αποτελέσματά τους στη ΒΔ.
- Η διαδικασία ανάκαμψης μπορεί να απλοποιηθεί εκτελώντας περιοδικά ελέγχους σε προκαθορισμένα **σημεία ελέγχου (check points)**
  - Έξοδος όλων των εγγραφών του ημερολογίου που βρίσκονται επί του παρόντος στην κύρια μνήμη στην ακλόνητη αποθήκευση.
  - Έξοδος όλων των τροποποιημένων blocks ενδιάμεσης μνήμης στα αντίστοιχα φυσικά blocks στο δίσκο.
  - Γράψιμο στο ημερολόγιο, στην ακλόνητη αποθήκευση, μιας εγγραφής <checkpoint>.

Σημεία ελέγχου (2)

- Κατά τη διάρκεια της ανάκαμψης πρέπει να θεωρήσουμε μόνο την πιο πρόσφατη δοσοληψία T_i που ξεκίνησε πριν το σημείο ελέγχου και τις δοσοληψίες που ξεκίνησαν μετά την T_i.
  - (κρατάμε την παραδοχή ότι οι δοσοληψίες εκτελούνται σειριακά)
  - Εξετάζουμε προς τα πίσω από το τέλος του ημερολογίου για να βρούμε την πιο πρόσφατη εγγραφή <checkpoint>.
  - Συνεχίζουμε ψάχνοντας προς τα πίσω μέχρι να βρεθεί μια εγγραφή <T_i start>
  - Θεωρούμε μόνο το τμήμα του ημερολογίου που ακολουθεί την παραπάνω start εγγραφή. Το προηγούμενο τμήμα του ημερολογίου μπορεί να αγνοηθεί (και μπορεί να διαγραφεί όποτε το επιθυμούμε).
  - Για όλες τις δοσοληψίες (από την T_i και έπειτα) χωρίς <T_i commit>, εκτελούμε undo(T_i). (μόνο στην περίπτωση άμεσης τροποποίησης.)
  - Εξετάζοντας προς τα εμπρός το ημερολόγιο, για όλες τις δοσοληψίες (από την T_i και έπειτα) με <T_i commit>, εκτελούμε redo(T_i).

Παράδειγμα σημείων ελέγχου



- Η δοσοληψία T_1 μπορεί να αγνοηθεί (οι ενημερώσεις έχουν ήδη εξαχθεί στο δίσκο εξαιτίας του σημείου ελέγχου T_c).
- Οι δοσοληψίες T_2 και T_3 επανεκτελούνται (επειδή υπάρχει commit).
- Η δοσοληψία T_4 ακυρώνεται (επειδή δεν υπάρχει commit).

Ανάκαμψη μετά από απώλεια του μέσου αποθήκευσης (1)

- Έως τώρα υποθέταμε ότι το μέσο μόνιμης αποθήκευσης (π.χ. δίσκος) στο οποίο βρίσκεται η ΒΔ δεν αποτυγχάνει. Όμως μπορεί να συμβεί κι αυτό !!
- Για να το αντιμετωπίσουμε, αποθηκεύουμε περιοδικά το περιεχόμενο της ΒΔ σε μέσο ακλόνητης αποθήκευσης (διαδικασία dump). Κατά τη διαδικασία αυτή, καμία δοσοληψία δεν είναι ενεργή
- Συγκεκριμένα: ...

Ανάκαμψη μετά από απώλεια του μέσου αποθήκευσης (2)

- **Διαδικασία dump**:
  - Εξάγουμε στην ακλόνητη αποθήκευση όλες τις εγγραφές ημερολογίου που βρίσκονται προσωρινά στην κύρια μνήμη.
  - Εξάγουμε όλα τα blocks ενδιάμεσης μνήμης στο δίσκο.
  - Αντιγράφουμε το περιεχόμενο της ΒΔ στην ακλόνητη αποθήκευση.
  - Γράφουμε μια εγγραφή <dump> στο ημερολόγιο, στην ακλόνητη αποθήκευση.
- Για ανάκαμψη μετά από απώλεια του δίσκου κάνουμε τα εξής:
  - Αποκαθιστούμε τη ΒΔ από την ακλόνητη αποθήκευση στο δίσκο (με βάση το τελευταίο dump)
  - Επανεκτελούμε όλες τις δοσοληψίες που έγιναν commit μετά την εγγραφή <dump> (με βάση το ημερολόγιο)