ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Βάσεις Δεδομένων ΙΙ

Τεχνικές Ανάνηψης

Μανώλης Μαραγκουδάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

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

Χρηματοδότηση

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

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Πανεπιστήμιο Αιγαίου-
Τμήμα Μηχανικών
Πληροφοριακών και
Επικοινωνιακών Συστημάτων

Βάσεις Δεδομένων ΙΙ

Ενότητα 3-Τεχνικές Ανάνηψης

Μανώλης Μαραγκουδάκης

www.icsd.aegean.gr/mmarag

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Ανάνηψη τη παρουσία WAL

Επίπεδα αποθήκευσης

- Κυρίως μνήμη
  - RAM, cache
  - Ταχύτητα στην προσπέλαση
  - Τα δεδομένα χάνονται σε περίπτωση αποτυχίας
- Δευτερεύουσα μνήμη
  - Σκληρός Δίσκος, Ταινίες
  - Πιο αργά, λόγω ηλεκτρομηχανικής κίνησης
  - Πιο αξιόπιστα σε επίπεδο αστοχίας
  - Υπόκεινται και αυτά σε αστοχίες όμως

Στη συνέχεια, Siberschatz, Korth & Sudarsan

Επιπλέον επίπεδα αποθήκευσης

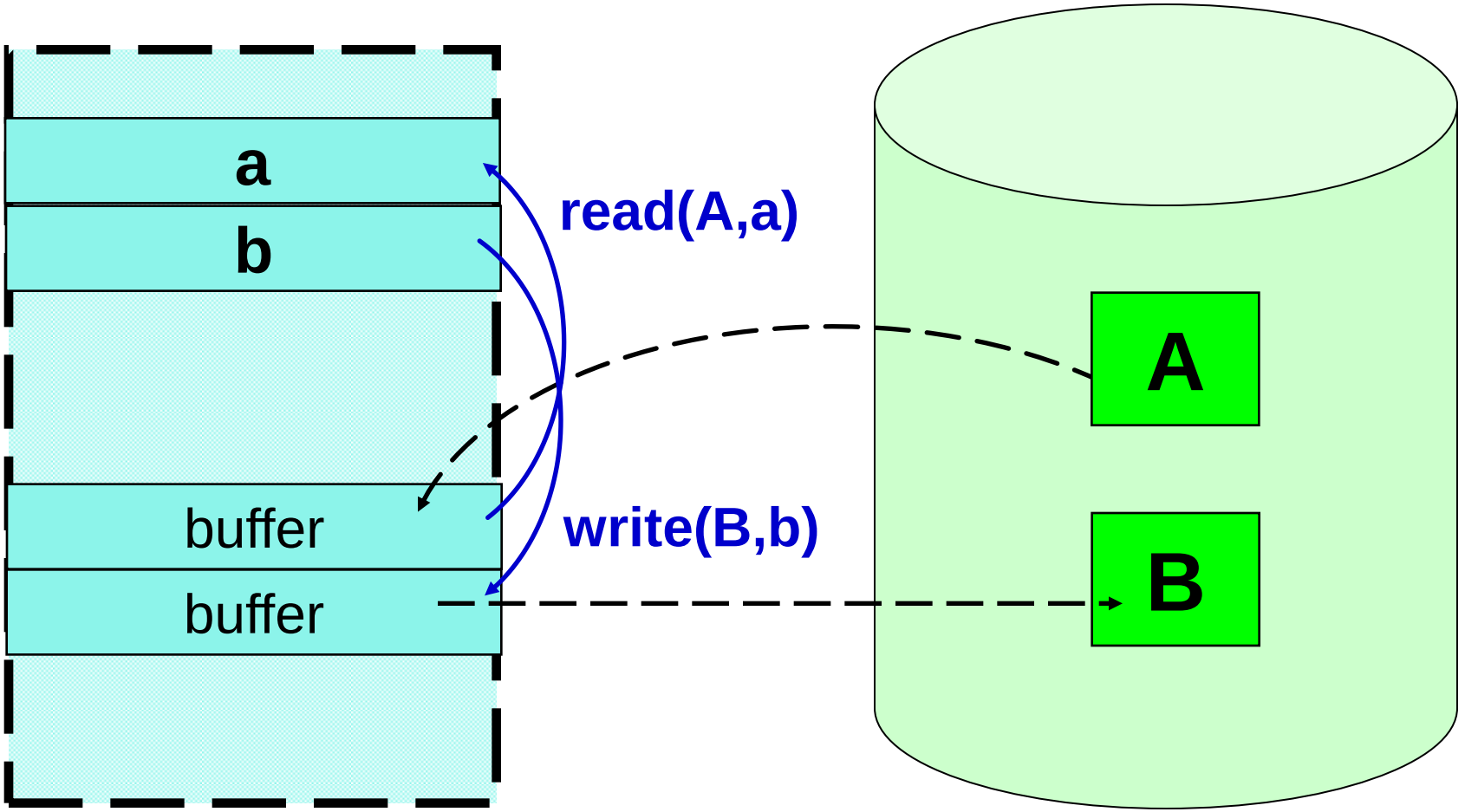
- Σταθερή αποθήκευση
  - Π.χ., συστήματα RAID [k αντίγραφα του ιδίου αποθηκευτικού μέσου]
- Hard-copies ☺

Αστοχίες του συστήματος

- Κακό πρόγραμμα (με λάθη, δηλ.)
- Αστοχία δοσοληψίας (αδιέξοδο, abort από τον χρήστη, κλπ)
- Αστοχία του συστήματος (πτώση ρεύματος, αδιέξοδο λειτουργικού συστήματος)
- Αστοχία υλικού (καταστροφή σκληρού δίσκου)

Failure: αστοχία ή αποτυχία

Read & Write

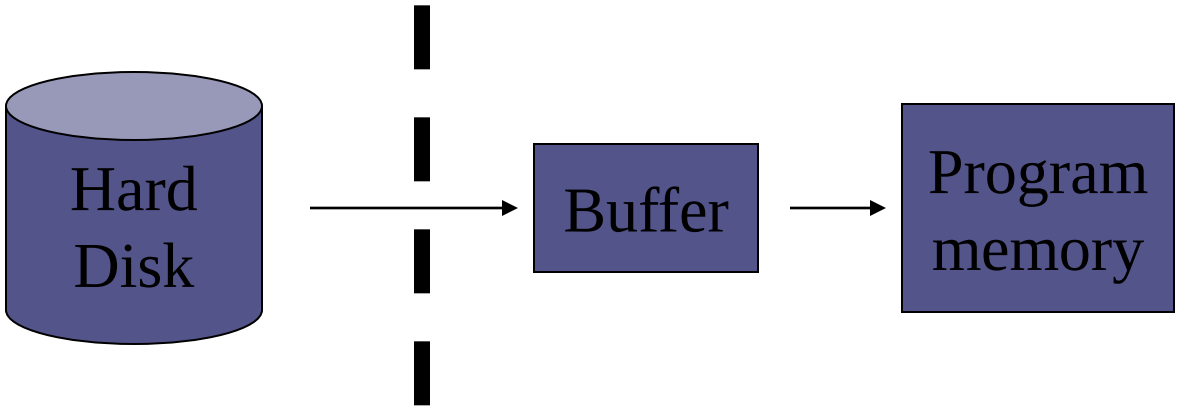


Οι διακεκομμένες αναπαριστούν λειτουργίες που μπορεί να καθυστερήσουν/ αγνοηθούν

Επανάληψη:

- Τι πάει να πει `read(a)` ?
- `a` είναι μια μεταβλητή του προγράμματος
- `read(a)` σημαίνει:
  - Διάβασε από το δίσκο την αντίστοιχη με το **a** εγγραφή στη βάση,
  - Φέρε την σε κάποιο buffer
  - Αντίγραψέ την στην περιοχή μνήμης του προγράμματος

Επανάληψη …



- Το αντίστοιχο συμβαίνει και με τη `write`
- Όπως έχουμε πει, το `a`, εν γένει, δεν είναι εγγραφή, αλλά σελίδα στο δίσκο …

Εμείς θα ασχοληθούμε με …

- Αστοχίες δοσοληψίας (αδιέξοδο, abort από τον χρήστη, κλπ)
- Αστοχίες του συστήματος (πτώση ρεύματος, αδιέξοδο λειτουργικού συστήματος)

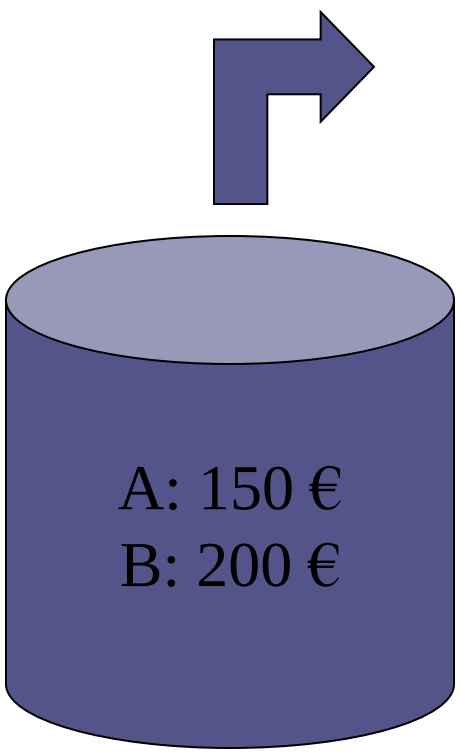
Τα αποτελέσματα τροποποιούνται για να αντιμετωπίσουμε και αστοχίες υλικού…

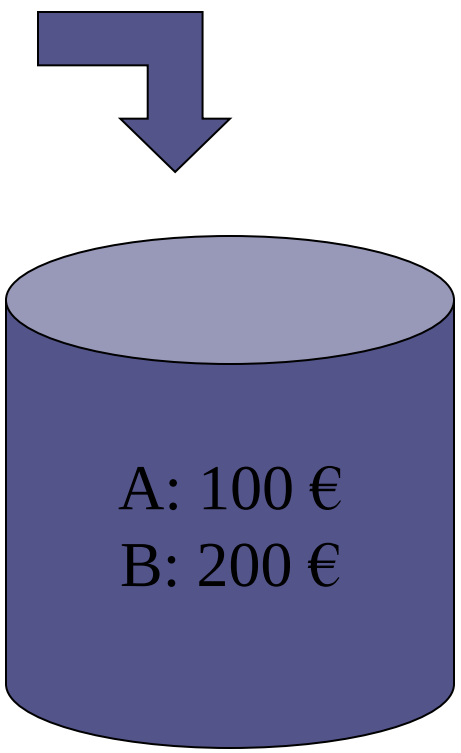
Αυτό μεταφράζεται πρακτικά ως …

- Τα δεδομένα από την κυρίως μνήμη χάνονται οριστικά
- Τα δεδομένα που έχουν αποθηκευθεί στο σκληρό δίσκο είναι ασφαλή
- Η ΒΔ πιθανώς βρίσκεται σε ασυνεπή μορφή.

«ασυνεπή»? Κάτι μου θυμίζει ...

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





A+B=350 ⟷ A+B=300

Σε περίπτωση αποτυχίας …

- Σκοπός είναι να διατηρήσουμε τη συνέπεια του συστήματος.
- Πρέπει να επαναλάβουμε (**REDO**) όλες τις δοσοληψίες που έκαναν commit
- Πρέπει να αναιρέσουμε (**UNDO**) όσες παρενέργειες επέφεραν οι δοσοληψίες που δεν πρόλαβαν να κάνουν commit

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Ανάνηψη παρουσία WAL

Log (Ιστορικό)

- **Log** (Ιστορικό/Ίχνος/Ημερολόγιο): ένα αρχείο στο σκληρό δίσκο που καταγράφει όλη την ιστορία των ενεργειών που εκτελέστηκαν από το DBMS
- Ενέργειες:
  - BOT/EOT (Begin/End Of Transaction)
  - INS/UPD/DEL (δηλ, write) ένα record
  - COMMIT/ABORT μια δοσοληψία
  - UNDO/REDO μια ενέργεια εγγραφής (write)

Βασικές έννοιες για το Log

- Το log ως αρχείο είναι ένα σύνολο από εγγραφές (**log records**)
- Κάθε log record χαρακτηρίζεται μονοσήμαντα από ένα **Log Sequence Number** (**LSN**).
- Το σύστημα εκδίδει αυτόματα το max(LSN)+1 για κάθε νέο log record

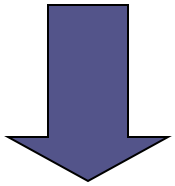
Log Record τύπου “Write”

- LSN
- Transaction ID (TID)
- Σελίδα που κάνουμε update
- Offset στην εν λόγω σελίδα
- Μήκος σε bytes που αλλάζουμε
- Παλιά τιμή
- Νέα τιμή

T1: UPDATE EMP

SET ID = 30

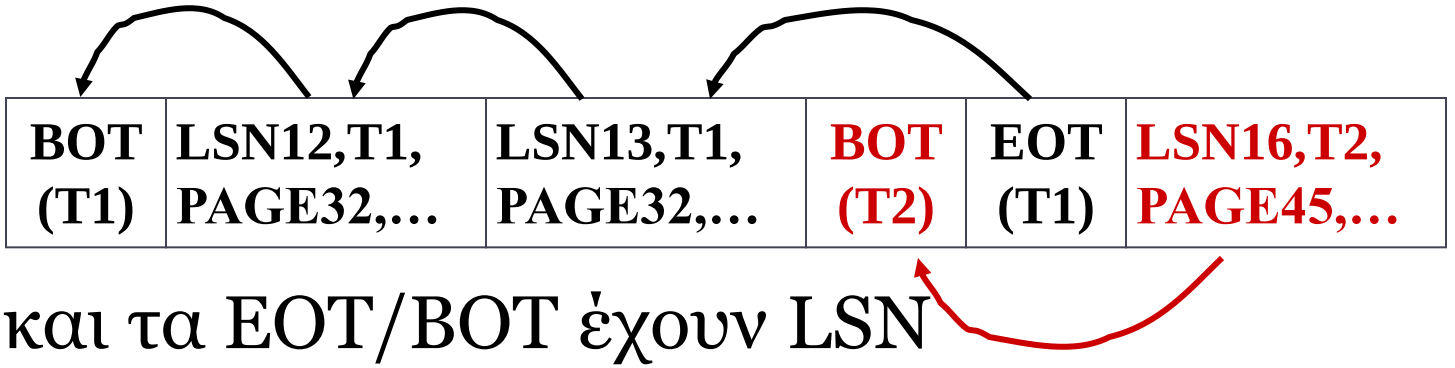
WHERE ID = 3



Log Entry: LSN12,T1,PAGE32,0xFFF32,8,3,30

Δεν φτάνουν αυτά …

- PrevLSN: Το ακριβώς προηγούμενο LSN της ίδιας δοσοληψίας



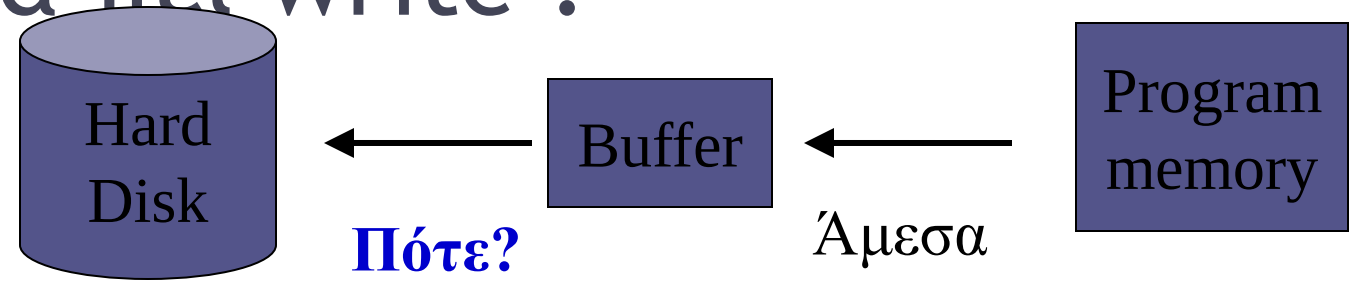
- …και τα EOT/BOT έχουν LSN

Βασική Αρχή Write Ahead Log

Προτού γράψεις οτιδήποτε στη ΒΔ, καταχώρησε την αντίστοιχη εγγραφή στο log

Φυσικά υπάρχουν τεχνικές λεπτομέρειες …

Τι θα πει write ?



- Σε κάθε commit?
- Σε κάθε μια ενέργεια write (ασχέτως commit)?
- Σε τακτά χρονικά διαστήματα?
- Κάθε όποτε δεν έχει δουλειά το μηχάνημα ?
- …

Dirty page: *σελίδα που έχει αλλαχθεί στον buffer, αλλά όχι στο σκληρό δίσκο*

Δύο βασικοί τρόποι εγγραφής

- **Steal**: επιτρέπουμε μια σελίδα να γραφτεί στο δίσκο, χωρίς να έχει κάνει commit η δοσοληψία που την άλλαξε [No-steal, αν γράφω μόνο committed σελίδες]
- **Force**: επιβάλλουμε σε όλες τις σελίδες μιας δοσοληψίας να γραφτούν στο δίσκο, αμέσως μετά το commit [No-Force, αν κάποιες μπορεί και να μη γραφτούν]

Στην πράξη: Steal

- Γράφω μια μη commited σελίδα στο δίσκο, είτε γιατί γέμισαν οι buffers, είτε γιατί αν κάνει commit η δοσοληψία χωρίς να αλλάξει ξανά τη σελίδα κέρδισα σε χρόνο
- Κι αν αποτύχει η δοσοληψία? Τότε πρέπει να κάνω UNDO στην αλλαγή της σελίδας

- Dirty bit: κρατάω ένα bit που λέει αν η σελίδα είναι dirty ή όχι

Στην πράξη: No-Force

- Δεν γράφω μια commited σελίδα στο δίσκο, για να αποφύγω το κόστος εγγραφής (π.χ., λόγω φόρτου του συστήματος εκείνη τη στιγμή)
- Κι αν αποτύχει το σύστημα? Τότε πρέπει να κάνω REDO στην αλλαγή της σελίδας στο δίσκο

Write Ahead Log revisited

Προτού γράψεις οτιδήποτε στη ΒΔ, καταχώρησε την αντίστοιχη εγγραφή στο log

- Για να γράψεις μια updated σελίδα από το buffer πίσω στο δίσκο, πρέπει στο log (στο δίσκο) να έχουν περαστεί οι παλιές τιμές για τα records της
- Για να κάνεις commit μια δοσοληψία πρέπει στο log (στο δίσκο) να έχουν γραφτεί όλες οι σχετικές log records

Με άλλα λόγια …

- ΠΡΙΝ γράψω μια σελίδα στη ΒΔ, γράφω όλα τα log records που την αφορούν στο δίσκο
- ΠΡΙΝ κάνω commit μια δοσοληψία, γράφω όλα τα log records που την αφορούν στο δίσκο

- Προσοχή: τα παραπάνω είναι περιορισμοί ορθότητας και όχι αλγόριθμος ☺

Σχόλια

- Steal: και να πας να κλέψεις στη ΒΔ, ΔΕΝ μπορείς να κλέψεις στο log
- No-Force: και να μην εξαναγκάσεις τις εγγραφές της ΒΔ να γραφούν στο δίσκο, πρέπει να γραφούν όλες οι log records

- Μην ξεχνάτε ότι και το log περνά από buffering!

Και τι κέρδισα?

- Για να γράψεις μια updated σελίδα από το buffer πίσω στο δίσκο, πρέπει στο log (στο δίσκο) να έχουν περαστεί οι παλιές τιμές για τα records της
  - Σε περίπτωση αποτυχίας της δοσοληψίας, μπορώ να κάνω UNDO
- Για να κάνεις commit μια δοσοληψία πρέπει στο log (στο δίσκο) να έχουν γραφτεί όλες οι σχετικές log records
  - Σε περίπτωση αποτυχίας του συστήματος, μπορώ να κάνω REDO

Ανάνηψη με καθυστερημένη ενημέρωση

- Οι ενημερώσεις δεν γράφονται στη βάση μέχρι να γίνει commit η συναλλαγή Τ.
- Τ ξεκινά:
  - Εγγραφή start στο log
- Τ κάνει commit:
  - Εγγραφή commit στο log
  - Όλες οι εγγραφές του log γράφονται στο δίσκο
  - Υλοποιούνται οι πραγματικές ενημερώσεις στη ΒΔ
- Τ κάνει abort:
  - Εγγραφή abort στο log
  - Αγνοούνται όλες οι εγγραφές Log της Τ
- Ανάνηψη
  - Συναλλαγές με Start και Commit στο Log ξαναγίνονται (redo)
  - Οι άλλες αγνοούνται.

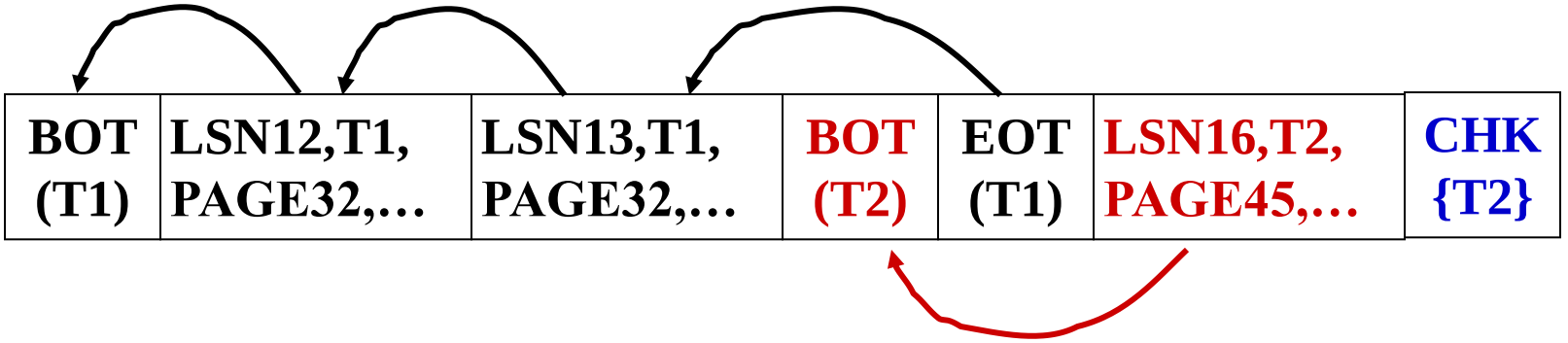
Ανάνηψη με άμεση ενημέρωση

- Οι ενημερώσεις γράφονται στη βάση όταν συμβαίνουν
- Τ ξεκινά:
  - Εγγραφή start στο log
- Τ κάνει write:
  - Εγγραφή update στο log
  - Η ενημέρωση γράφεται στο Buffer και μετά στη ΒΔ
- Τ κάνει commit:
  - Εγγραφή commit στο log
- Τ κάνει abort:
  - Οι ενημερώσεις αναιρούνται (undo) με αντίστροφη σειρά
- Ανάνηψη
  - Συναλλαγές με Start και Commit στο Log ξαναγίνονται (redo)
  - Συναλλαγές με Start και **ΌΧΙ** Commit στο Log αναιρούνται (undo)

Checkpoints – Σημεία ελέγχου

- Περιοδικά, το σύστημα κάνει τις εξής ενέργειες:
  - Σταματά κάθε άλλη ενέργεια
  - Καταγράφει το σύνολο των ενεργών δοσοληψιών
  - Γράφει (**flush**) όλους τους buffers με log records, στο δίσκο
  - Γράφει (**flush**) όλους τους buffers με records της ΒΔ, στο δίσκο
  - Γράφει στο log μια εγγραφή CHK (checkpoint)

Οπότε το log θα δείχνει κάπως έτσι …



{T2} είναι το σύνολο των ενεργών δοσοληψιών

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Ανάνηψη τη παρουσία WAL

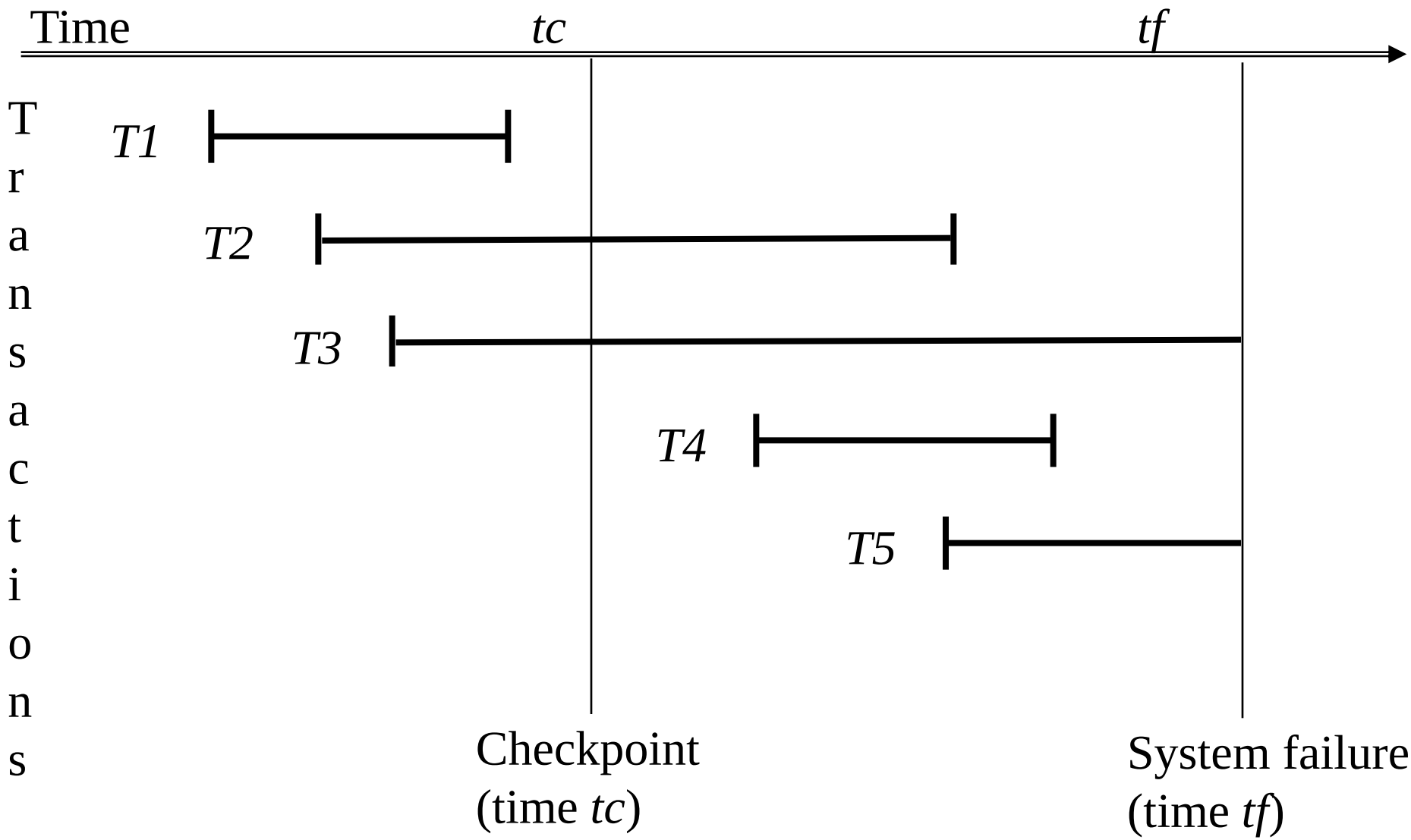
Ανάνηψη αν έχουμε WAL

- Έστω ότι το σύστημα αποτυγχάνει και πρέπει να το επαναφέρουμε (δηλ. να επαναφέρουμε τη ΒΔ σε συνεπή μορφή).
- Η διαδικασία αυτή ονομάζεται **ανάνηψη** (**recovery**) ή ανάκαμψη ή επαναφορά
- Αν έχουμε χρησιμοποιήσει WAL κατά την κανονική λειτουργία του συστήματος, η ανάνηψη έχει **3 φάσεις:**
  1. Ανάλυση
  2. UNDO
  3. REDO

Φάση Ανάλυσης

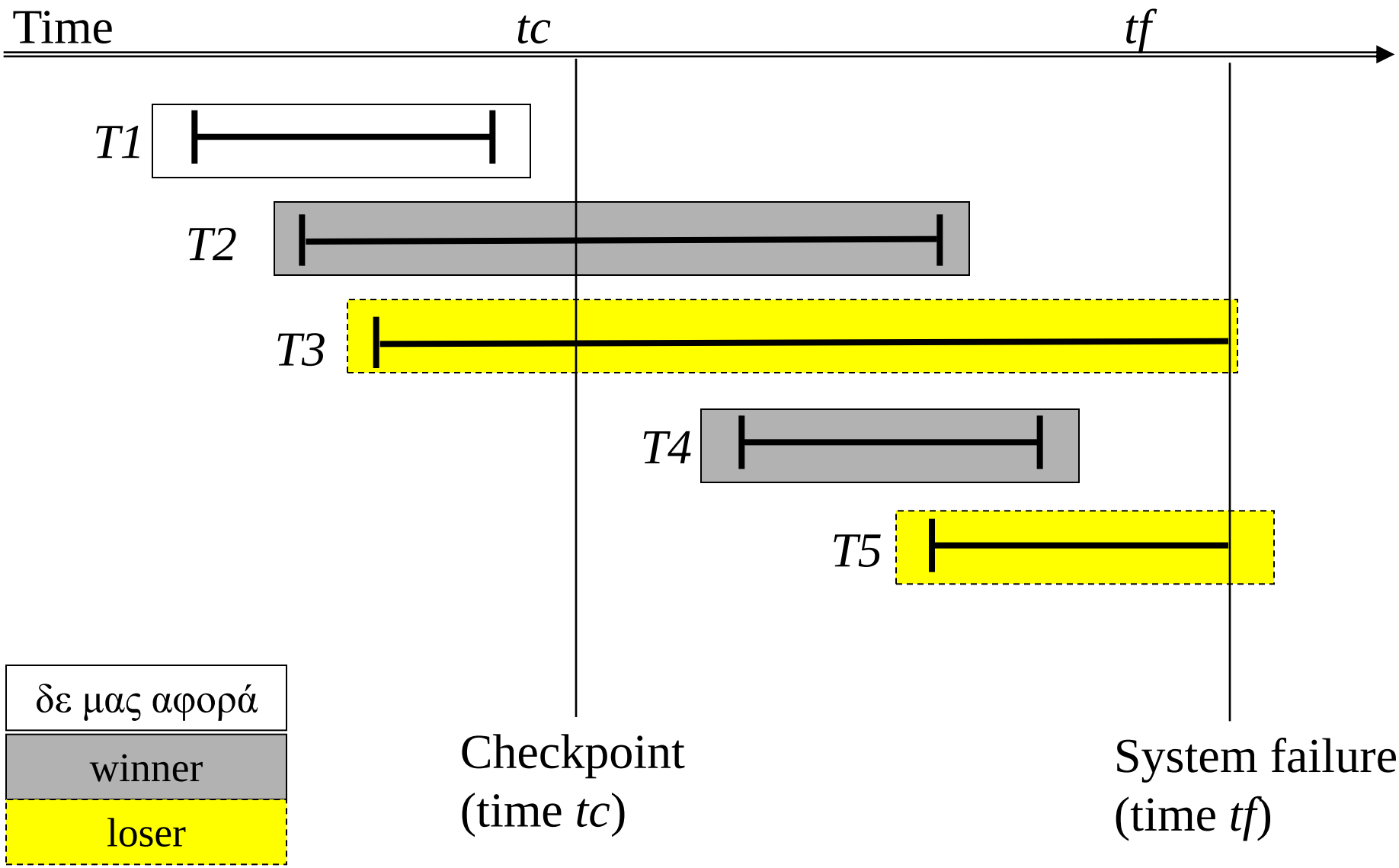
- Διαβάζουμε το log από το τελευταίο CHK ως το τέλος
- Ανακαλύπτουμε
  - **νικητές (winners)**, ήτοι, δοσοληψίες που πρόλαβαν και έκαναν commit μέσα σε αυτό το διάστημα
  - **ηττημένους (losers)**, ήτοι δοσοληψίες που είτε δεν πρόλαβαν να κάνουν commit, είτε οι χρήστες τους τις έκαναν abort
- Εντοπίζουμε τις dirty pages τη στιγμή της αποτυχίας (βλ. στη συνέχεια)

Winners & losers



Source: An Introduction to Database Systems, C.J. Date, p. 381

Winners & losers



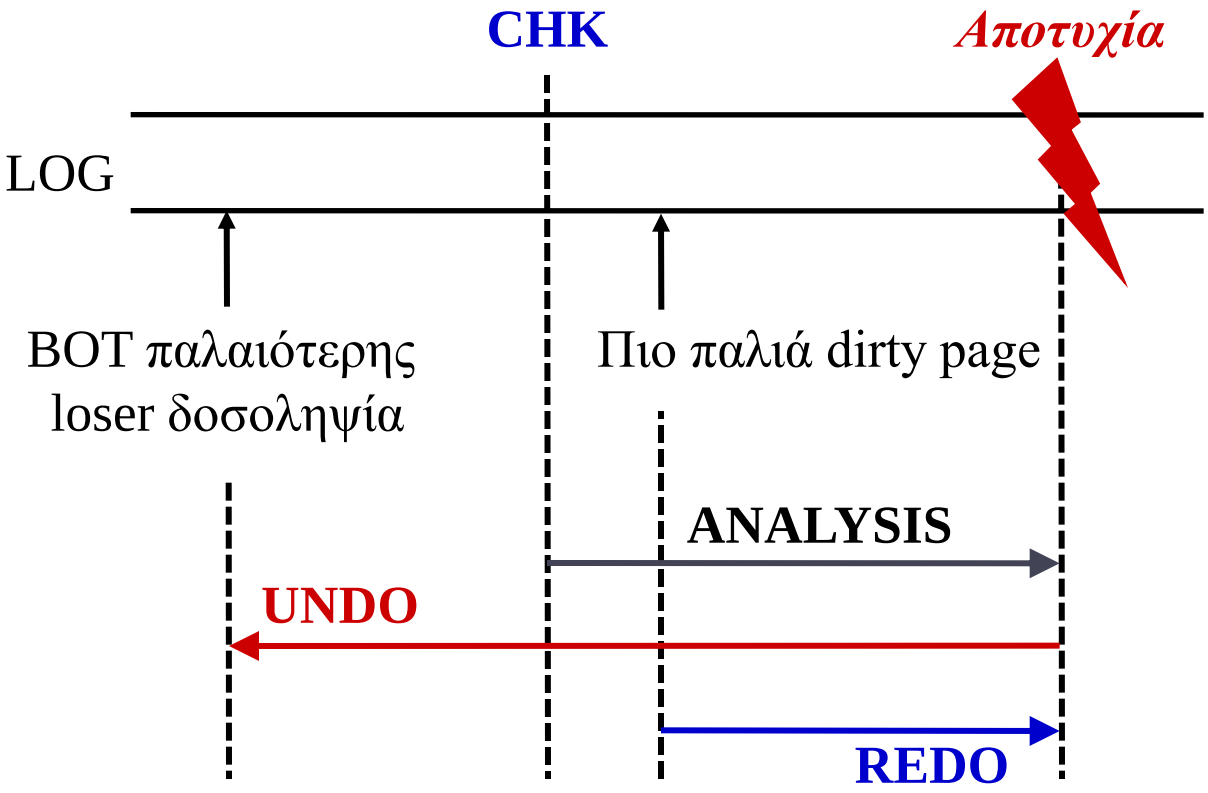
Φάση UNDO

- **UNDO losers!**
- Διαβάζουμε ανάποδα το log, από το τέλος προς την αρχή
- Κάθε πράξη που ανήκει σε δοσοληψία loser γίνεται UNDO
- Προσοχή: μπορεί μια loser να έχει ξεκινήσει πριν το checkpoint!

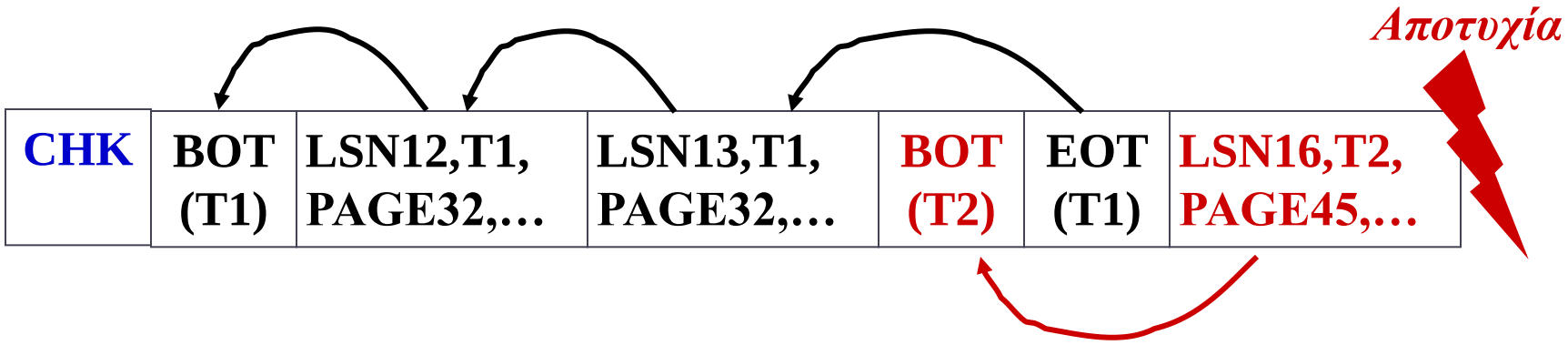
Φάση REDO

- **REDO winners!**
- Διαβάζουμε κανονικά το log, από το σημείο που κάναμε update την πιο παλιά buffer page ως το τέλος
- Κάθε πράξη που ανήκει σε δοσοληψία winner γίνεται REDO

Ανάνηψη αν έχουμε WAL



Εδώ τι θα κάναμε ?



Step	T1	T2	Log
1	Read(b1)		Start(T1)
2	b1:=b1-a		
3	Write(b1)		(T1, b1, v1, v1-a)
4		Read(b1)	Start(T2)
5		b1:=b1-a'	
6		Write(b1)	(T2, b1, v1-a, v1-a-a')
7	Read(b2)		
8	b2:=b2+a		
9	Write(b2)		(T1, b2, v2, v2+a)
10	commit		Commit(T1)
11		Read(b2)	
12		b2:=b2+a'	
13		Write(b2)	(T2, b2, v2+a, v2+a+a')
14		commit	Commit(T2)

Πώς εντοπίζω τις dirty pages?

- Έστω ότι σε κάθε σελίδα κρατάω ένα πεδίο **pageLSN** που καταγράφει το LSN της τελευταίας ενέργειας που έκανε write κάποιο record στη σελίδα
- Πώς μπορώ να βρω τις dirty pages όταν ανανήψει το σύστημα?
- Τι θα άλλαζε αν δεν κρατούσα αυτή την πληροφορία?