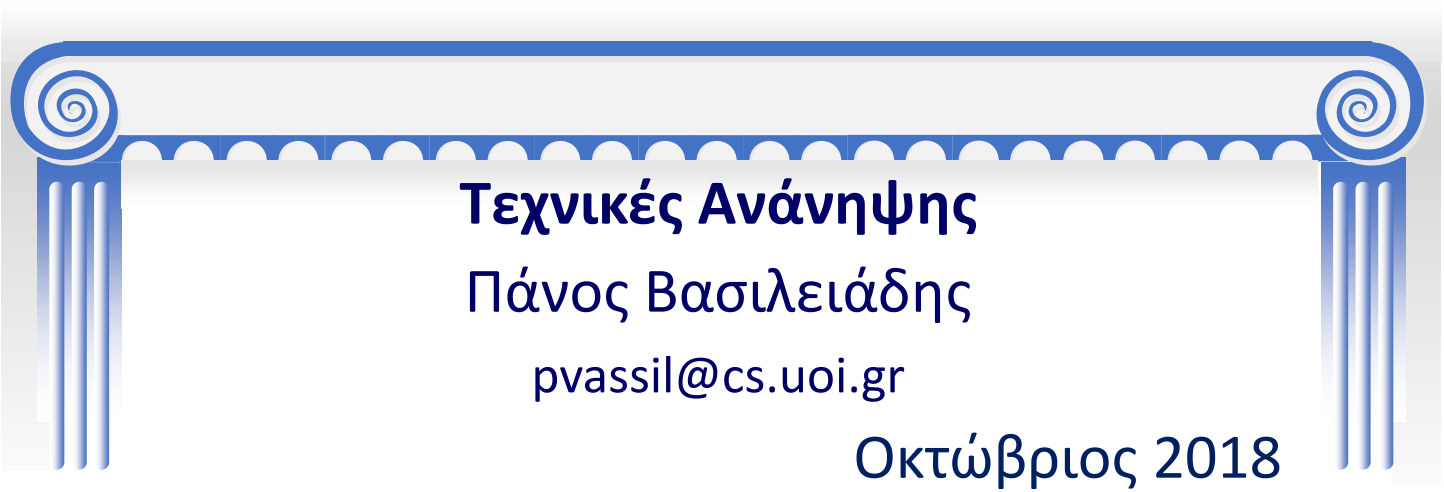
Προχωρημένα Θέματα Τεχνολογίας και Εφαρμογών Βάσεων Δεδομένων



Τεχνικές Ανάνηψης

Πάνος Βασιλειάδης

pvassil@cs.uoi.gr

Οκτώβριος 2018

www.cs.uoi.gr/~pvassil/courses/db_III/

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Checkpoints
- Commit & Abort σε κανονική λειτουργία
- Ανάνηψη τη παρουσία WAL

Περιεχόμενα

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 συναλλαγής (αδιέξοδο, abort από τον χρήστη, κλπ)
- Αστοχία του συστήματος (πτώση ρεύματος, αδιέξοδο λειτουργικού συστήματος)
- Αστοχία υλικού (καταστροφή σκληρού δίσκου)

Failure: αστοχία ή αποτυχία

Εμείς θα ασχοληθούμε με ...

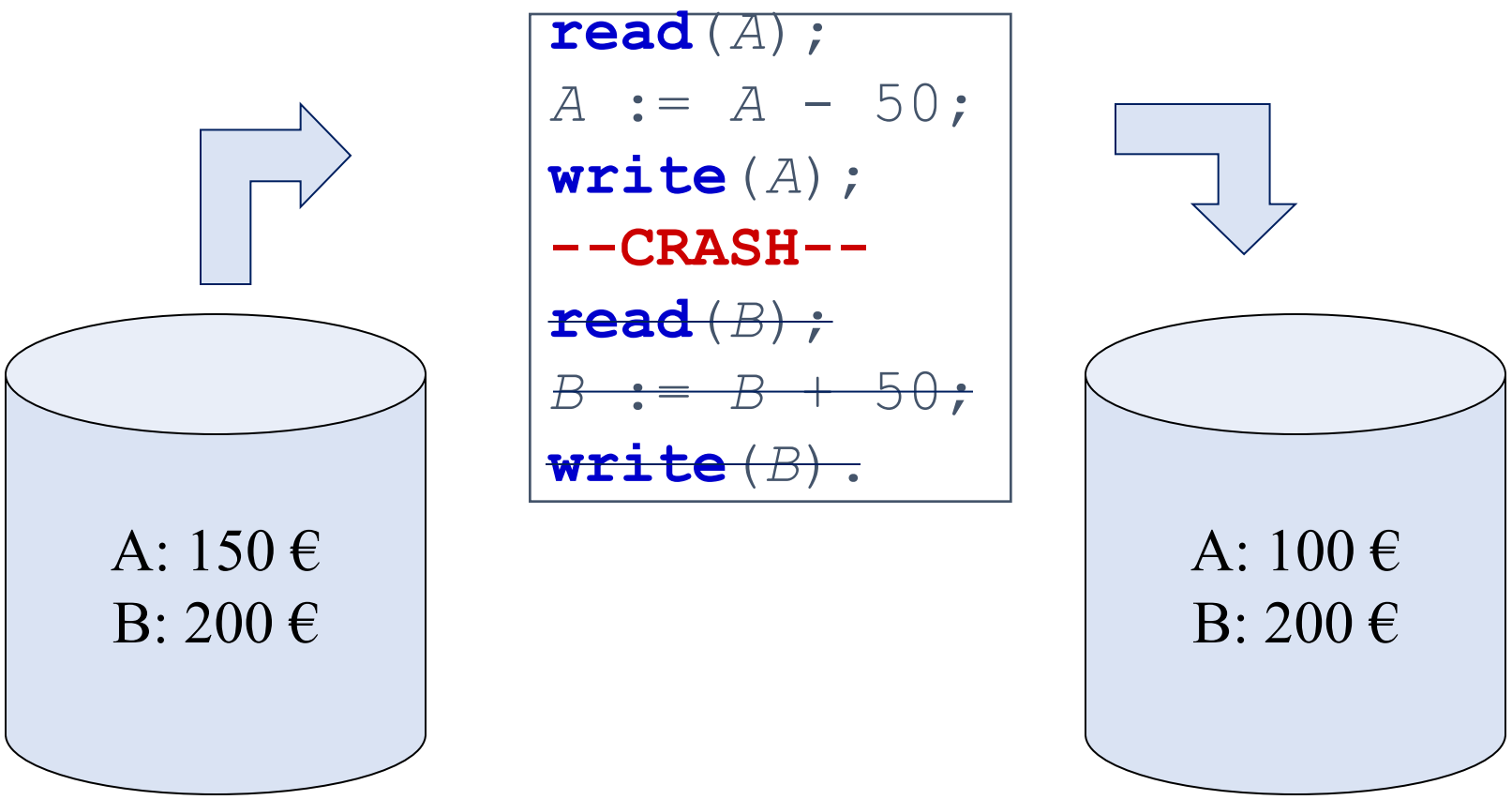
- Αστοχίες συναλλαγής (αδιέξοδο, abort από τον χρήστη, κλπ)
- Αστοχίες του συστήματος (πτώση ρεύματος, αδιέξοδο λειτουργικού συστήματος)

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»? Θυμηθείτε τη συνέπεια από το ACID test



Σε περίπτωση αποτυχίας ...

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

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Checkpoints
- Commit & Abort σε κανονική λειτουργία
- Ανάνηψη τη 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 (ήτοι, write) ένα record
  - COMMIT/ABORT μια συναλλαγή
  - UNDO/REDO μια ενέργεια εγγραφής (write)
  - CLR: Compensation Log Records για τα UNDO's

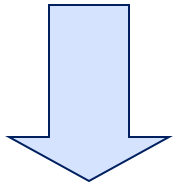
Βασικές έννοιες για το Log

- 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 νέα 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

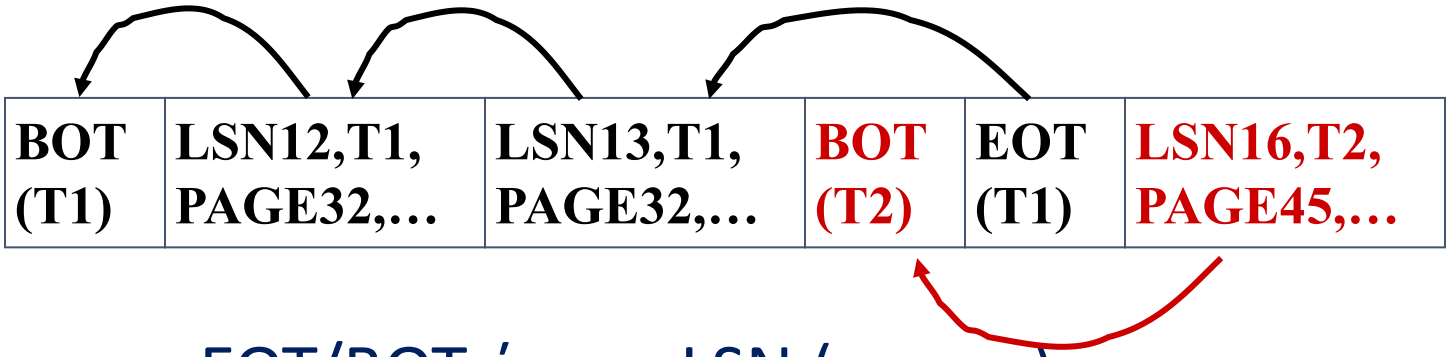
```
T1: UPDATE EMP
SET ID = 30
WHERE ID = 3
```



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

Δεν φτάνουν αυτά …

- PrevLSN: Το ακριβώς προηγούμενο LSN της ίδιας συναλλαγής
- Τι γίνεται αν αλλάξει σελίδα η εγγραφή?



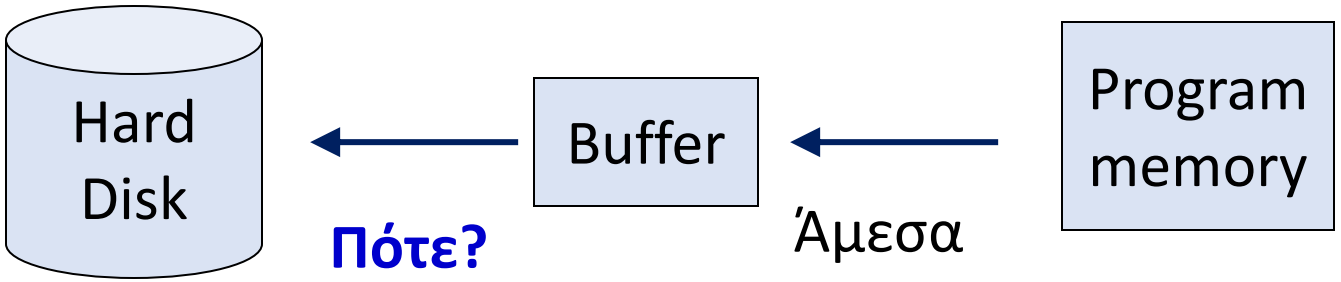
- …και τα EOT/BOT έχουν LSN (oops…)

Βασική Αρχή Write Ahead Logging

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

Φυσικά υπάρχουν τεχνικές λεπτομέρειες …

Τι θα πει write ?



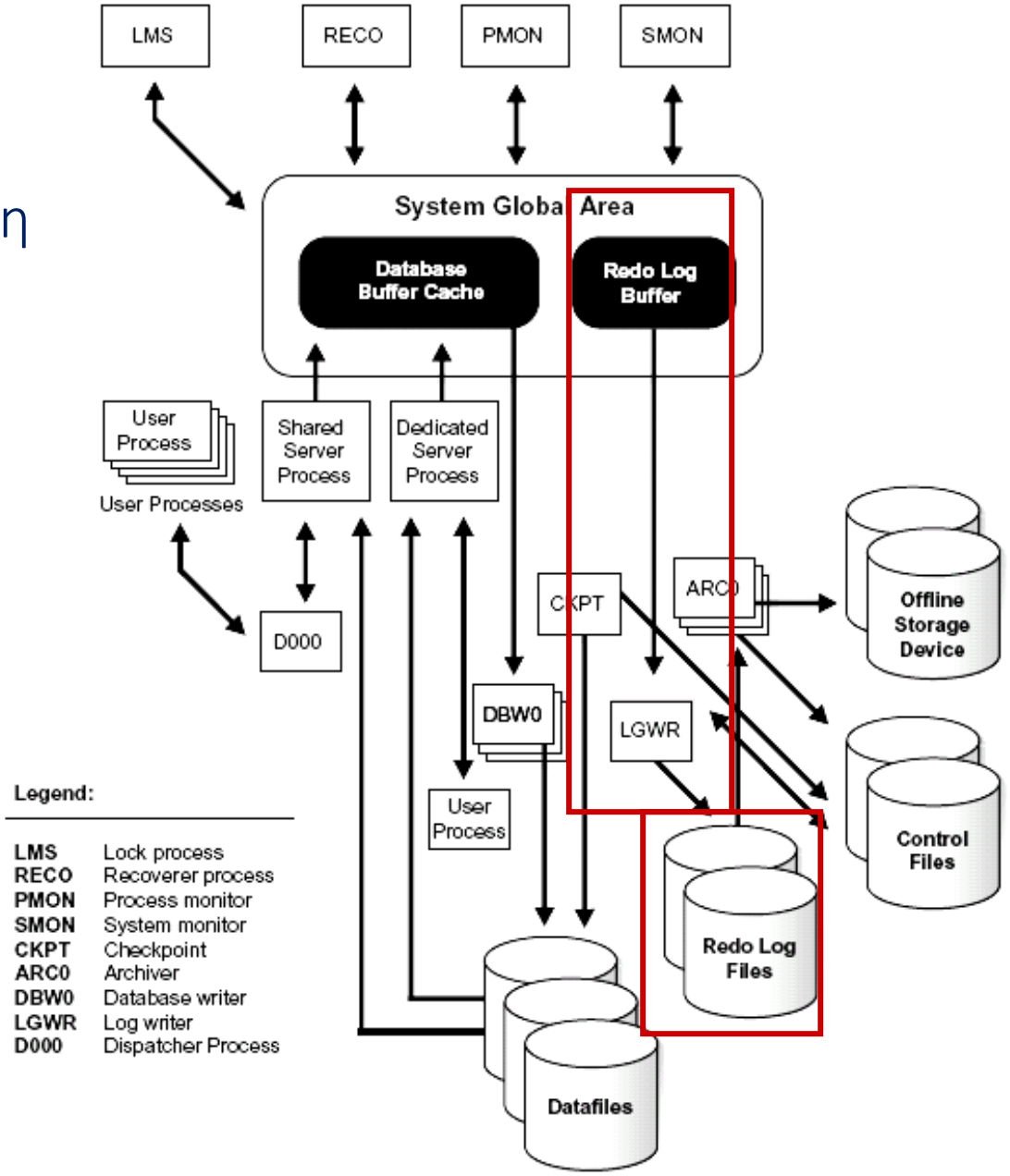
- Σε κάθε commit?
- Σε κάθε μια ενέργεια write (ασχέτως commit)?
- Σε τακτά χρονικά διαστήματα?
- Κάθε όποτε δεν έχει δουλειά το μηχάνημα ?
- ...

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

Log entries στο σκληρό δίσκο και την κύρια μνήμη

Έχω buffers στην κύρια μνήμη για το log.

Log writer:
Υπεύθυνος για να μεταφέρει σελίδες από τους log buffers στο (append only) log file



Δύο βασικοί τρόποι εγγραφής

- **Steal**: επιτρέπουμε μια σελίδα να γραφτεί στο δίσκο, χωρίς να έχει κάνει commit η συναλλαγή που την άλλαξε [No-steal, αν γράφω μόνο committed σελίδες]
- **Force**: επιβάλλουμε σε όλες τις σελίδες μιας συναλλαγής 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 συναλλαγή χωρίς να αλλάξει ξανά τη σελίδα κέρδισα σε χρόνο
- Κι αν αποτύχει η συναλλαγή? 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 συναλλαγή 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 συναλλαγή, 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 συναλλαγής, μπορώ να κάνω UNDO
- Για να κάνεις commit μια συναλλαγή πρέπει στο log (στο δίσκο) να έχουν γραφτεί όλες οι σχετικές log records
  - Σε περίπτωση αποτυχίας του συστήματος, μπορώ να κάνω REDO

Περιεχόμενα

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

Επιτάχυνση της ανάνηψης

- Αν έχω μόνο το log file, όταν συμβεί μια αποτυχία, πρέπει να επαναληφθεί όλη η ιστορία της βάσης δεδομένων, από την αρχή της κατασκευής της
- Δύο μηχανισμοί επιταχύνουν την ανάνηψη και ταυτόχρονα αυξάνουν την αξιοπιστία της διαθεσιμότητας των δεδομένων σε περίπτωση αποτυχίας:
  - Backup
  - Checkpoints

Backup

- Με τα backup το ζήτημα είναι λίγο πολύ προφανές: αν επαναφέρουμε το backup, το μόνο που χρειάζεται είναι να επαναλάβουμε την ιστορία από τη στιγμή του backup και μετά

από το backup
ως τώρα

Checkpoints – Σημεία ελέγχου

- Περιοδικά, το σύστημα κάνει τις εξής ενέργειες:
  - Σταματά κάθε άλλη ενέργεια
  - Καταγράφει το σύνολο των **ενεργών συναλλαγών**
  - 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)

Checkpoints

- Sharp checkpoint: το προαναφερθέν είδος checkpoint
- Fuzzy checkpoint: αντί να σταματήσει το σύστημα, γράφει μόνο ποιες είναι οι dirty pages + οι ενεργές συναλλαγές και στέλνει την εγγραφή των σελίδων αυτών στο background. Δικαιούμαστε να ξαναπάρουμε checkpoint μόνο όταν η παρασκηνιακή αυτή διεργασία τελειώσει.

Fuzzy Checkpoints

- Αν έχουμε fuzzy checkpoint:
  - Οι συναλλαγές συνεχίζουν να τρέχουν, οπότε η παραπάνω καταγραφή αφορά την κατάσταση του συστήματος στην αρχή του checkpoint.
  - ΔΕΝ κάνουμε force τις dirty pages στο δίσκο στη διάρκεια του checkpoint
- Έτσι, αν γίνει μια αποτυχία, δεν έχουμε καμία εγγύηση ότι οι dirty pages γράφτηκαν στο δίσκο και η ανάνηψη καθυστερεί παραπάνω
- Από την άλλη, η διαθεσιμότητα του συστήματος αυξάνει

Checkpoints & εγγραφές στο log

- Στο log γράφεται:
  - begin_checkpoint record: πότε ξεκίνησε το checkpoint
  - end_checkpoint record: περιέχει τον πίνακα ενεργών συναλλαγών και των πίνακα με τις *dirty pages table*.
  - Πάντα κρατάμε το LSN των checkpoint records **αλλού** (master record) για να χρησιμοποιηθεί κατά την ανάνηψη

Εκτός κι αν ειπωθεί ξεκάθαρα, στη συνέχεια θα θεωρούμε sharp checkpoints και ένα CHKP record χάριν απλότητας

Βαριάντες fuzzy checkpoints

- Δεν παίρνουμε νέο checkpoint αν δεν έχει ολοκληρωθεί η καταγραφή όλων των dirty pages του προηγούμενου το δίσκο – πρακτικά, η ύπαρξη του επόμενου checkpoint υπονοεί ότι το προηγούμενο ολοκληρώθηκε
- Απλά στο checkpoint καταγράφουμε τον πίνακα από dirty pages, στον οποίο, για κάθε dirty σελίδα, καταγράφουμε το LSN της πιο παλιάς LSN που την τροποποιεί, από τότε που τη φέραμε στη μνήμη (**recLSN**). Στην ανάνηψη, θα ξεκινήσουμε το REDO από το πιο παλιό recLSN που καταγράφηκε στο τελευταίο checkpoint

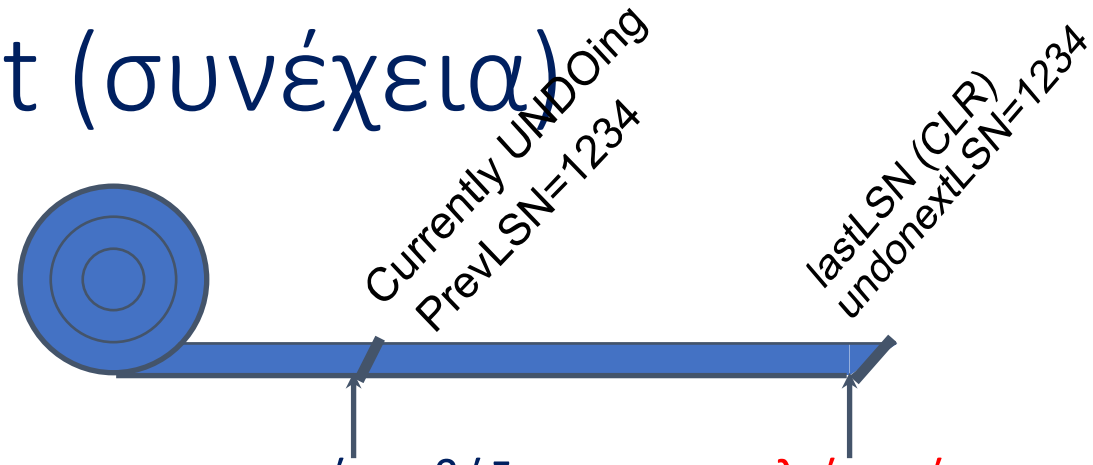
Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Checkpoints
- Commit & Abort σε κανονική λειτουργία
- Ανάνηψη τη παρουσία WAL

Abort συναλλαγής σε κανονική λειτουργία

- Θεωρήστε ένα απλό abort μιας συναλλαγής, με το σύστημα να λειτουργεί κανονικά.
- Θα αναιρέσουμε τις επιμέρους πράξεις της συναλλαγής με την ανάποδη σειρά κάνοντας ενέργειες UNDO.
  - Βρίσκουμε το lastLSN της συναλλαγής (το έχει ο πίνακας συναλλαγών που κρατάμε στην μνήμη.
  - Ακολουθούμε αντίστροφα τις ενέργειες της συναλλαγής μέσω του prevLSN τους.
  - Πριν κάνουμε τα UNDO, γράφουμε ένα *Abort* log record.

Abort (συνέχεια)



- Για κάθε ενέργεια που αναιρούμε, βάζουμε την παλιά τιμή στη σχετική σελίδα (την οποία έχουμε φέρει στην κύρια μνήμη, όπου και την ενημερώνουμε).
- Πριν το κάνουμε αυτό, γράφουμε στο log ένα **Compensation Log Records** (CLR). CLR's = log entries που καταγράφουν τις πράξεις του συστήματος από την ανάνηψη μέχρι την επαναφορά σε κανονική λειτουργία
- Κάθε CLR έχει ένα επιπλέον πεδίο: undonextLSN που δείχνει το LSN που πρέπει να γίνει undo αμέσως μετά (δλδ. Το prevLSN της log record την οποία κάνουμε τώρα undo).
  - Τα CLRs ΠΟΤΕ δεν υφίστανται Undo στην ανάνηψη
- Στο τέλος του UNDO, γράφουμε ένα "Transaction end" log record.

Commit συναλλαγής σε κανονική λειτουργία

- Γράφουμε ένα commit record στο log.
- Όλα τα log records μέχρι το τελευταίο LSN της συναλλαγής (το lastLSN από τον πίνακα συναλλαγών) γίνονται flush στο Log file από τους log buffers.
  - Εγγυούμεθα ότι flushedLSN ≥ lastLSN.
- Η συναλλαγή ενημερώνεται ότι το Commit() ολοκληρώθηκε.
- Καταγράφεται ένα "Transaction end" log record.

Περιεχόμενα

- Εισαγωγή & υποθέσεις εργασίας
- Αλγόριθμος Write-Ahead Log (WAL)
- Checkpoints
- Commit & Abort σε κανονική λειτουργία
- Ανάνηψη τη παρουσία WAL

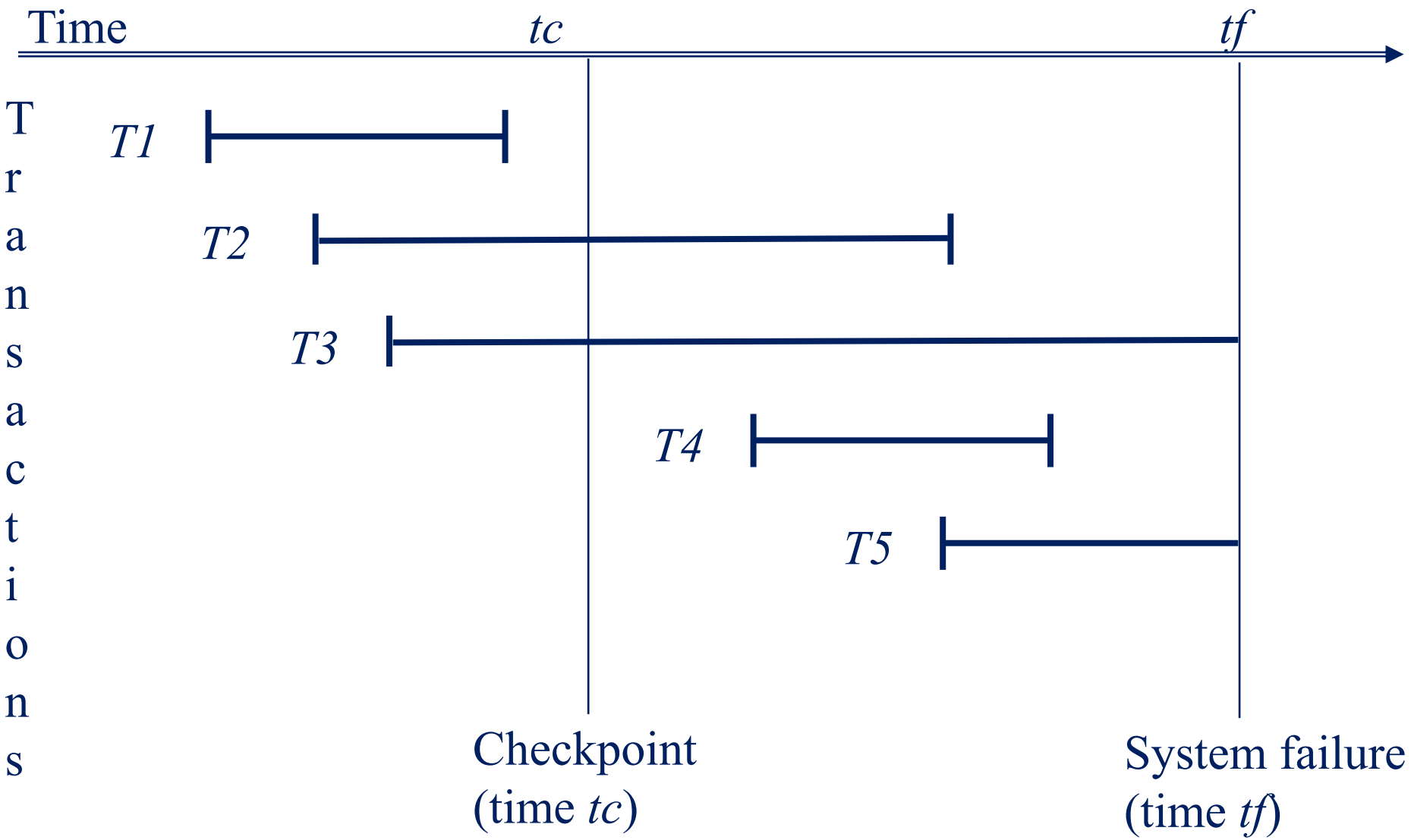
Ανάνηψη αν έχουμε WAL

- Έστω ότι το σύστημα αποτυγχάνει και πρέπει να το επαναφέρουμε (ήτοι, να επαναφέρουμε τη ΒΔ σε συνεπή μορφή).
- Έστω επίσης ότι το σύστημα υλοποιεί ένα **Strict 2PL πρωτόκολλο**
- 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 (DO)
    2. REDO
    3. UNDO

Φάση Ανάλυσης

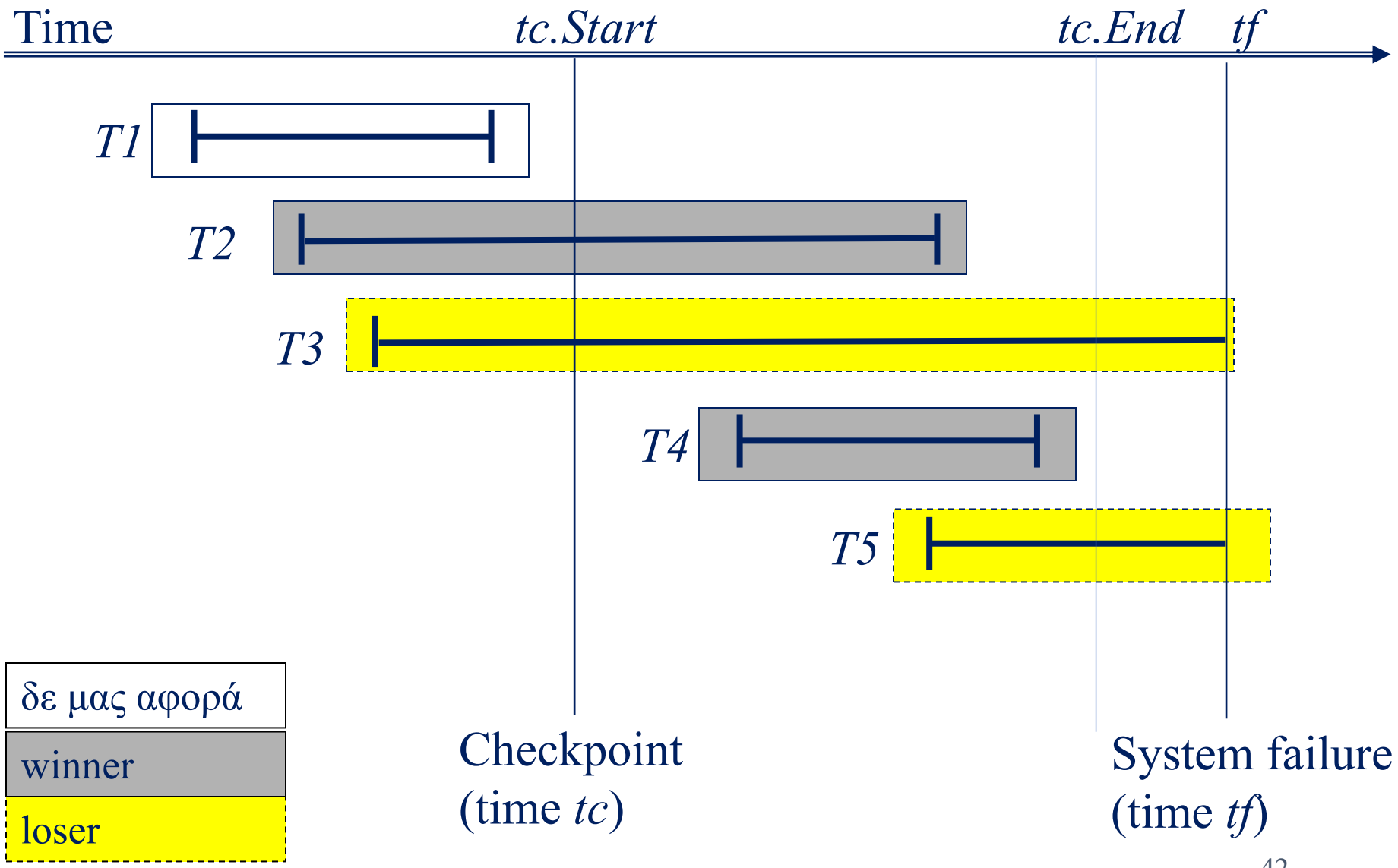
- Διαβάζουμε το log από το τελευταίο CHK ως το τέλος
- Ανακαλύπτουμε
  - **νικητές (winners)**, ήτοι, συναλλαγές που πρόλαβαν και έκαναν commit μέσα σε αυτό το διάστημα
  - **ηττημένους (losers)**, ήτοι συναλλαγές που είτε δεν πρόλαβαν να κάνουν commit, είτε οι χρήστες τους τις έκαναν abort
  - Όλοι οι άλλοι ονομάζονται “don’t care’s”
- 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 της ανάλυσης στην απλοϊκή εκδοχή

- Από το τέλος του log file ως το τελευταίο checkpoint:
  - Όποιο log record <T, COMMIT> λέει ότι συναλλαγή έχει κάνει commit, μπαίνει στη winner list
  - Όποιο log record <T, ABORT> λέει ότι συναλλαγή έχει κάνει abort, μπαίνει στην abort list
- Όποια συναλλαγή ήταν στις ενεργές συναλλαγές του checkpoint και δεν έχει κάνει ούτε commit, ούτε abort, μπαίνει στην abort list

Φάση της ανάλυσης στην ARIES εκδοχή

- Από το τέλος του log file ως το τελευταίο checkpoint:
  - Όποιο log record <T, END> ασχέτως commit ή abort βρεθεί, λέει ότι συναλλαγή T δε θα γίνει undo
  - Κάθε άλλη συναλλαγή (δλδ οι ενεργές τη στιγμή της αποτυχίας), μπαίνει στην undo list
- Δομές δεδομένων που χρειαζόμαστε
  - Μαζί με τις ενεργές συναλλαγές που υπήρχαν στο checkpoint, όποια άλλη συναλλαγή συναντήσουμε μπαίνει σε ένα πίνακα συναλλαγών, το Xact table, μαζί με το πεδίο lastLSN που λέει ποιο ήταν το τελευταίο LSN της κάθε συναλλαγής
  - Για κάθε σελίδα που αλλάζει, κρατάμε ένα πίνακα Dirty Page Table, οποίος για κάθε σελίδα κρατά και το recLSN = το πιο παλιό LSN που την αλλάζει

Φάση REDO

- **Απλοϊκή εκδοχή**
  - **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 συναλλαγή winner γίνεται REDO

- **ARIES εκδοχή**
  - **REDO all (winners, losers, CLR's)!** Επαναφέρουμε τη βάση στην κατάσταση που ήταν τη στιγμή της κατάρρευσης
  - Αποφεύγουμε άχρηστες αλλαγές (βλ. παρακάτω)
  - Θέτουμε το pageLSN στο τρέχον LSN.
  - ΔΕΝ γράφουμε τίποτα στο log file

Φάση UNDO

- **UNDO losers!** Διαβάζουμε ανάποδα το log, από το τέλος προς την αρχή
- Κάθε πράξη που ανήκει σε συναλλαγή loser γίνεται UNDO
- ARIES:

```
While there exist losers not fully undone{
        Choose largest loser LSN not undone.
        If this LSN is a CLR and undonextLSN==NULL
                Write an End record for this Xact.
        If this LSN is a CLR, and undonextLSN != NULL
                Add undonextLSN to ToUndo
        Else this LSN is an update => Undo the update & write a CLR
}
```

- Προσοχή: μπορεί μια loser να έχει ξεκινήσει πριν το checkpoint!

Στο τέλος…

- Καταγράφεται ένα Recovery complete log record
- Κανονικά, το σύστημα παίρνει ένα checkpoint ώστε να εκκινήσει με
  - όλες τις πειραγμένες σελίδες flushed at disk
  - all committed changes flushed at disk
  - all changes of aborted transactions undone from disk
- … δλδ, στο δίσκο δεν υπάρχουν uncommitted values, και στην κύρια μνήμη δεν υπάρχουν dirty pages

Πώς εντοπίζω τις dirty pages?

- Έστω ότι σε κάθε σελίδα κρατάω ένα πεδίο **pageLSN** που καταγράφει το LSN της τελευταίας ενέργειας που έκανε update κάποιο record στη σελίδα
- Avoid REDOing a page when:
  - Affected page is not in the Dirty Page Table, or
  - Affected page is in D.P.T., but has recLSN > LSN, or
  - pageLSN (in DB) ≥ LSN.

Compensation Log Records

- Και τι κάνουμε αν το σύστημα αποτύχει ξανά κατά τη διαδικασία της ανάνηψης?
- Η κεντρική ιδέα είναι ότι συνεχίζουμε από κει που είχαμε μείνει (όσο αυτό είναι δυνατό), αξιοποιώντας
  - όπου υπάρχει checkpoint μέσα στη διαδικασία ανάνηψης, και,
  - το γεγονός ότι κάθε πράξη που κάνουμε κατά τη διαδικασία ανάνηψης καταγράφεται επίσης στο log file με **Compensation Log Records** (CLR's)

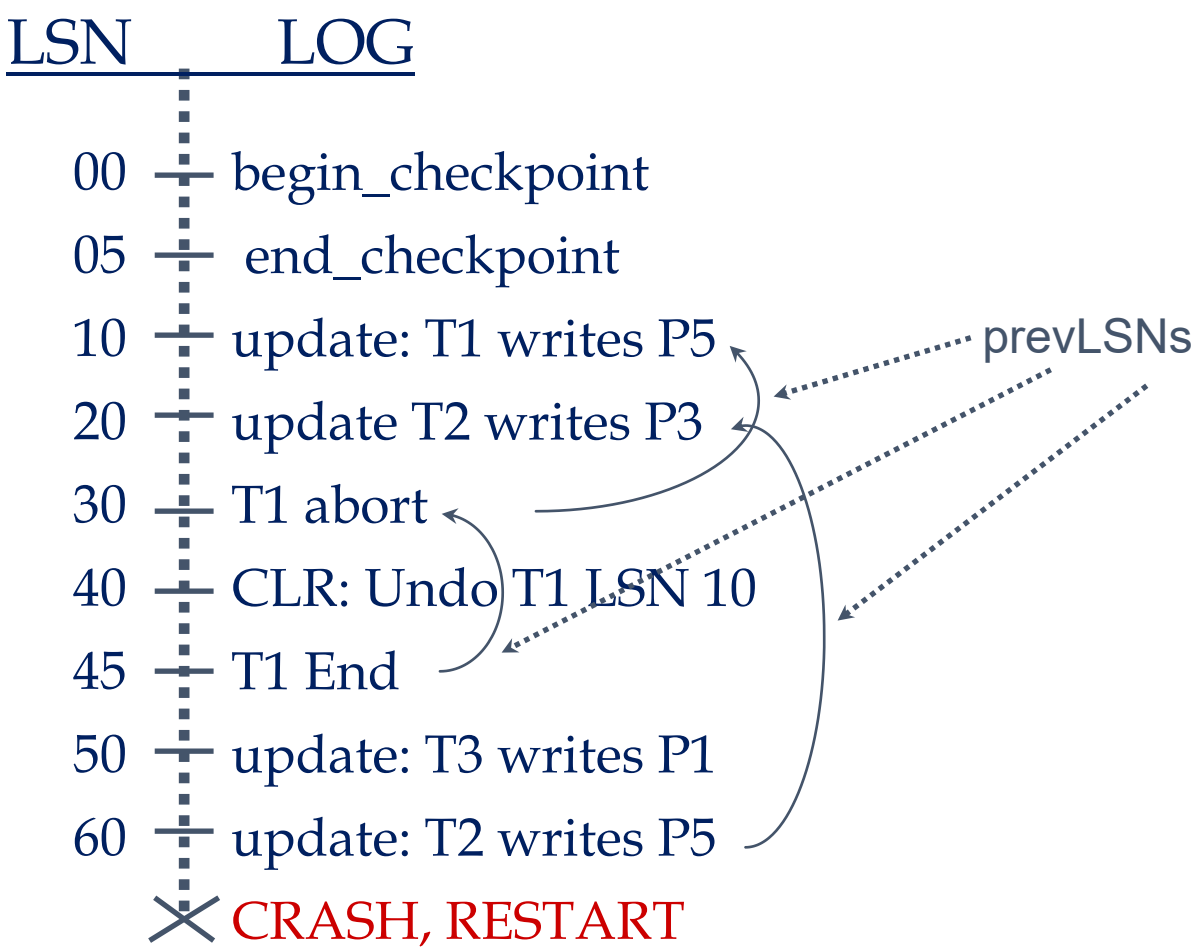
Τι γίνεται σε περίπτωση αποτυχίας?

Xact Table
- lastLSN
- status

Dirty Page Table
- recLSN

flushedLSN

ToUndo



Αποτυχία κατά την ανάνηψη

Xact Table
lastLSN
status
Dirty Page Table
recLSN
flushedLSN

ToUndo

LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART
90	CLR: Undo T2 LSN 20, T2 end

undonextLSN

Διαφάνειες από
Database Management Systems, R. Ramakrishnan and J. Gehrke

Ερωτήσεις κρίσεως ...

- Πώς μπορώ να προφυλαχθώ από αποτυχίες του υλικού [με βάση όλα τα προηγούμενα]?
- Τι θα κέρδιζα/έχανα/άλλαζε αν πέρναγα τα updates στο δίσκο, μόνο στο commit [και όχι πιο πριν] ?
- Τι θα κέρδιζα/έχανα/άλλαζε αν έκανα υποχρεωτικώς flush τις dirty pages στο commit ?

Ερωτήσεις κρίσεως

- Τι θα γίνει αν κατά τη διάρκεια της ανάνηψης το σύστημα αποτύχει ξανά?
- Τι θα άλλαζε αν αντί για `[παλιά τιμή,νέα τιμή]` στο log record έγραφα `[παλιά τιμή,διαφορά]`?
- Έχει σημασία η σειρά των UNDO και REDO? Μέσα σε κάθε μια από αυτές τις φάσεις, έχει σημασία η σειρά?

Μην ξεχνάτε:

από το backup
ως τώρα

Offline && On-line backups

- Offline backup
  - Η βάση δεν είναι διαθέσιμη σε άλλους χρήστες πλην του DBA
  - Δεν υπάρχουν uncommitted or active transactions
  - Μια καθαρή εκδοχή της βάσης καταγράφεται σε ένα μέσο αποθήκευσης
- Recovery with Offline backup
  - Restore the database από το backup
  - Redo post-backup winner transactions from the log

Offline && On-line backups

- On-line backup
  - Η βάση λειτουργεί κανονικά
  - Λειτουργούμε κατ' αντιστοιχία με ένα fuzzy checkpoint
  - Όλα τα περιεχόμενα της βάσης καταγράφονται σε ένα μέσο αποθήκευσης, χωρίς να περάσουμε από τον lock manager (άρα αντιγράφονται και uncommitted records)
- Recovery with On-line backup
  - Restore the database από το backup
  - Process the log and act as if it was a fuzzy checkpoint (Do-Redo-Undo)