Επαναφορά του Συστήματος (Μέρος Α')

Κεφάλαιο 18

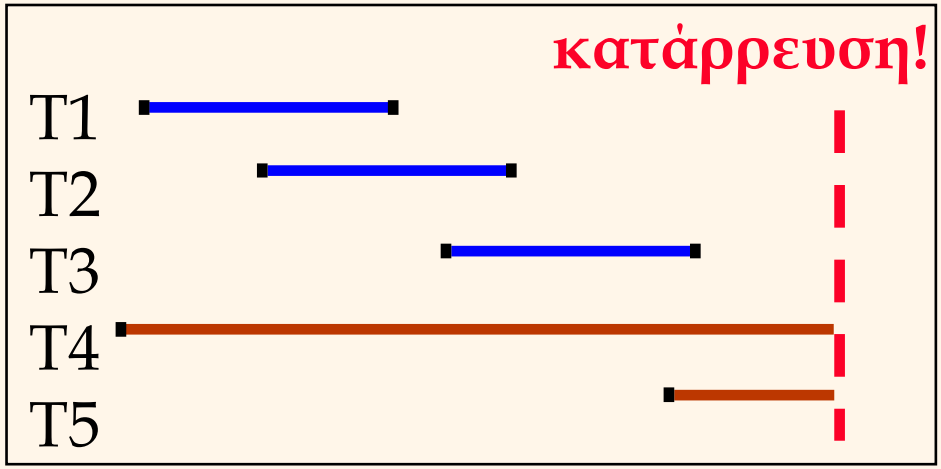
Επανάληψη: Οι ιδιότητες ACID

- A tomicity (Ατομικότητα): Όλες οι ενέργειες μιας συναλλαγής εκτελούνται ή καμία δεν εκτελείται.
- C onsistency (Συνέπεια): Αν κάθε συναλλαγή είναι συνεπής και ξεκινήσουμε με μια συνεπή βάση δεδομένων τότε καταλήγουμε με μια συνεπή βάση δεδομένων.
- I solation (Απομόνωση): Η εκτέλεση μιας συναλλαγής είναι απομονωμένη από τις εκτελέσεις των υπολοίπων.
- D urability (Μονιμότητα): Αν μια συναλλαγή ολοκληρώσει, τα αποτελέσματά της είναι μόνιμα.

- Ο **Διαχειριστής Επαναφοράς** εγγυάται τα A και D.

Κίνητρο

- Atomicity:
  - Οι συναλλαγές μπορεί να εγκαταλείψουν (“Rollback”).
- Durability:
  - Τί γίνεται αν καταρρεύσει το DBMS; (Αιτίες;)
- Επιθυμητή συμπεριφορά μετά την επανεκκίνηση του Συστήματος:
  - Οι T1, T2 & T3 θα πρέπει να είναι μόνιμες.
  - Οι T4 & T5 θα πρέπει να εγκαταλείψουν (οι αλλαγές τους να μην είναι ορατές).



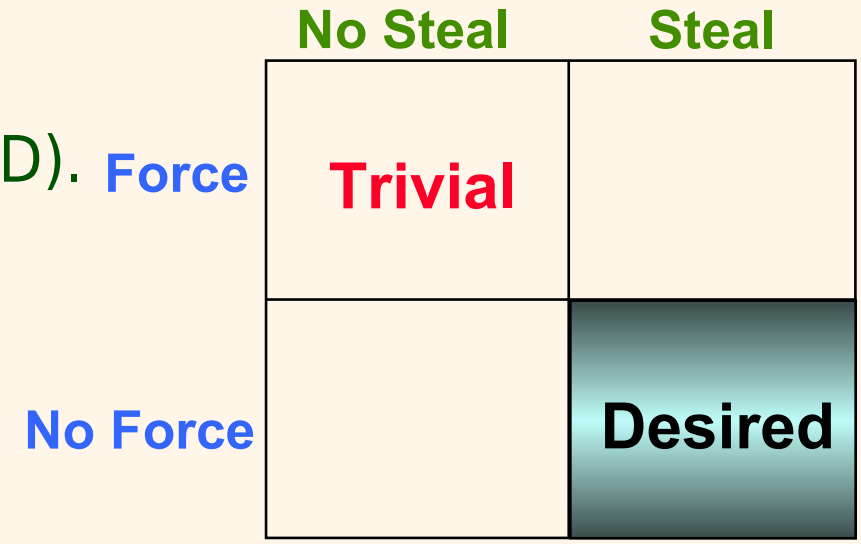
Υποθέσεις

- Υπάρχει Έλεγχος Ταυτοχρονισμού.
  - Πιο συγκεκριμένα το Αυστηρό 2PL.
- Οι Ενημερώσεις (Updates) συμβαίνουν “επί τόπου”.
  - δηλ., τα δεδομένα επανεγράφονται στον ίδιο χώρο στο δίσκο (τα παλιά διαγράφονται).

- Σκεφτείτε ένα απλό σχήμα που εγγυάται τις Atomicity & Durability.

Διαχείριση της Ενδιάμεσης Μνήμης

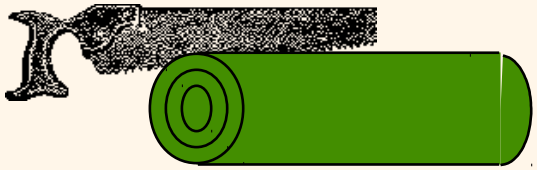
- Force (να εξαναγκάσουμε) κάθε γράψιμο στο δίσκο;
  - Κακός χρόνος απόκρισης.
  - Αλλά έχουμε μονιμότητα (D).
- Steal (να κλέψουμε) πλαίσια της ενδιάμεσης μνήμης από μη ολοκληρωμένες συναλλαγές;
  - Αν όχι, κακή απόδοση, αλλά έχουμε ατομικότητα (A).
  - Αν ναι, πώς εξασφαλίζουμε την ατομικότητα (A);



Περισσότερα για τα Steal και Force

- **STEAL** (γιατί η επιβολή της Ατομικότητας είναι δύσκολη)
  - *Για να "κλέψουμε" (steal) είναι πλαίσιο F:* Η τρέχουσα σελίδα στο F (έστω P) γράφεται στο δίσκο, αλλά κάποια συναλλαγή έχει κλειδαριά στην P.
    - Τί γίνεται αν η συναλλαγή με την κλειδαριά εγκαταλείψει;
    - Πρέπει να θυμόμαστε την παλιά τιμή της P κατά τη στιγμή του κλεψίματος (ώστε να υποστηρίξουμε την αναίρεση [UNDO] των τροποποιήσεων στη σελίδα P).
- **NO FORCE** (γιατί η επιβολή της Μονιμότητας είναι δύσκολη)
  - Τί γίνεται αν το σύστημα καταρρεύσει πριν γραφεί μια τροποποιημένη σελίδα στο δίσκο;
  - Για να υποστηρίξουμε την επανεκτέλεση [REDO] των τροποποιήσεων στη σελίδα, πρέπει να γράφουμε σε κατάλληλο χώρο κάποια ελάχιστη πληροφορία τη στιγμή της ολοκλήρωσης.

Βασική Ιδέα: Διατήρηση Ιστορικού



- Καταγράφουμε REDO και UNDO πληροφορία για κάθε ενημέρωση (update) σε ένα ιστορικό *(log)*.
  - Οι εγγραφές στο ιστορικό είναι σειριακές (το βάζουμε σε ένα χωριστό δίσκο).
  - Στο ιστορικό καταγράφεται η ελάχιστη πληροφορία (διαφορική/diff), ώστε σε μια σελίδα ιστορικού να χωράνε πολλές ενημερώσεις.
- Ιστορικό: Μια ταξινομημένη λίστα ενεργειών REDO/UNDO
  - Μια εγγραφή ιστορικού περιέχει:

    <XID, pageID, offset, length, old data, new data>
  - και επιπλέον πληροφορίες ελέγχου (θα δούμε σε λίγο).

Πρωτόκολλο Write-Ahead Logging(WAL)

- Το Πρωτόκολλο Write-Ahead Logging:
  - #1 Πρέπει να εξαναγκάζεται (force) η εγγραφή ιστορικού για μια ενημέρωση, *πριν* πάει στο δίσκο η αντίστοιχη σελίδα δεδομένων.
  - #2 Πρέπει να γράφονται όλες οι εγγραφές ιστορικού μιας συναλλαγής *πριν την ολοκλήρωσή της*.
- Το #1 εγγυάται την Ατομικότητα (A).
- Το #2 εγγυάται την Μονιμότητα (D).
- Πώς ακριβώς γίνεται η καταγραφή του ιστορικού (και η επαναφορά!) του συστήματος;
  - Θα μελετήσουμε τον αλγόριθμο ARIES.