

Συναλλαγές

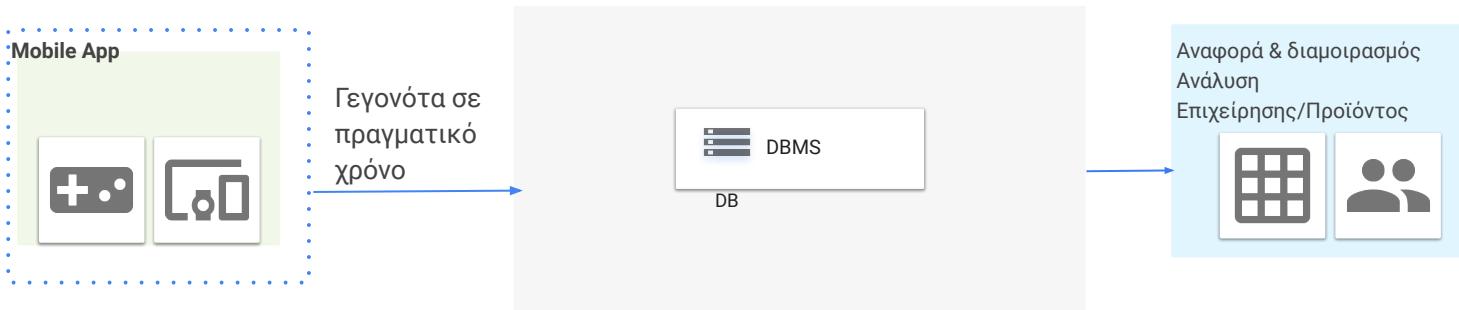
Βάσεις Δεδομένων

Πως;

Παράδειγμα Παιχνίδι (Game App)

DB v0

(Υπενθύμιση)



- Q1: 1000 χρήστες/sec;
- Q2: Εκτός σύνδεσης;
- Q3: Υποστήριξη v1, v1' εκδόσεων;

Σχεδιαστής εφαρμογής

- Q7: Πως να μοντελοποιήσουμε/εξελίξουμε τα δεδομένα του παιχνιδιού;
- Q8: Πως να το κλιμακώσουμε σε εκατομύρια χρήστες;
- Q9: Όταν πεθάνει η συσκευή, επανάφερε την κατάσταση του παιχνιδιού κομψά;

Σχεδιαστής συστημάτων

- Q4: Ποιός χρήστης συνοδεύει;
- Q5: Επόμενα χαρακτηριστικά; Πειράματα να τρέξουμε;
- Q6: Πρόβλεψη απαιτήσεων των διαφημίσεων;

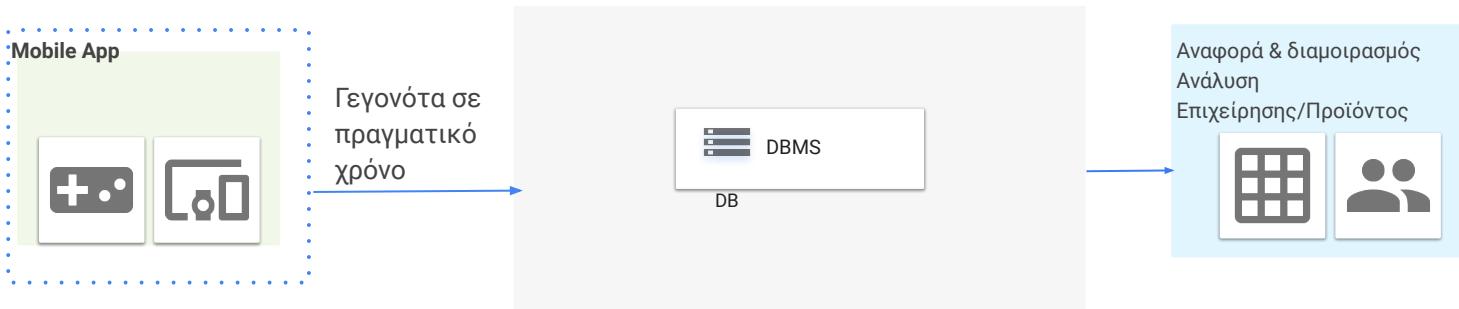
Σχεδιαστής προϊόντος/επιχείρησης

Πως;

Παράδειγμα Παιχνίδι (Game App)

DB v0

(Υπενθύμιση)



Q1: 1000 χρήστες/sec;

Q2: Εκτός σύνδεσης;

Q3: Υποστήριξη v1 εκδόσεων;

Θα δούμε σήμερα

Σχεδιαστής εφαρμογής

Q7: Πως να μοντελοποιήσουμε/εξελίξουμε τα δεδομένα του παιχνιδιού;

Q8: Πως να το κλιμακώσουμε σε εκατομμύρια χρήστες;

Q9: Όταν πεθάνει η συσκευή, επανάφερε την κατάσταση του παιχνιδιού κομψά;

Σχεδιαστής συστημάτων

Q4: Ποιός χρήστης συνοδεύει;

Q5: Επόμενα χαρακτηριστικά;
Πειράματα να τρέξουμε;

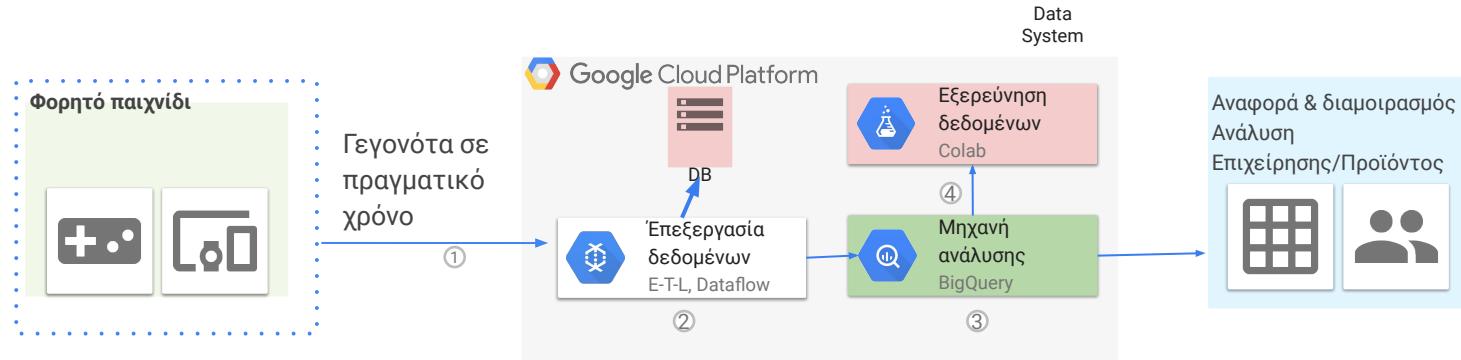
Q6: Πρόβλεψη απαιτήσεων των διαφημίσεων;

Σχεδιαστής προϊόντος/επιχεί

Πως;

Παράδειγμα Παιχνίδι

Σύστημα δεδομένων “v1” στο Cloud



1 Καταγραφή ενεργειών χρήστη

2 Αποθήκευση στη ΒΔ, ύστερα από
Εξαγωγή-Μετατροπή-Φόρτωση

3 Τρέξιμο ερωτημάτων σε ένα
σύστημα ανάλυσης ρετα κλίμακας

4 Οπτικοποίηση αποτελεσμάτων
ερωτήματος

Συναλλαγές

1ο Μέρος

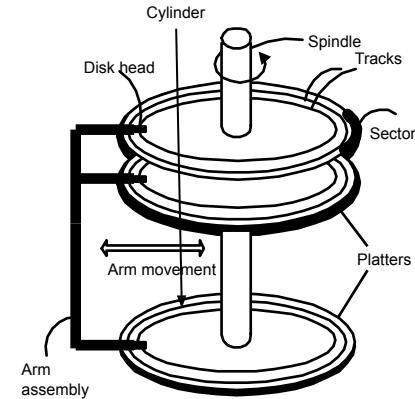
- ▶ **Επιδόσεις Βασικής Αποθήκευσης**
- ▶ Το μοντέλο του ΣΔΒΔ
- ▶ Συναλλαγές
 - ▷ Βασικά των Συναλλαγών
 - ▶ Κίνητρα για συναλλαγές
 - ▶ Ιδιότητες Συναλλαγών (ACID)

Επιδόσεις Βασικής Αποθήκευσης

Υψηλό επίπεδο: Δίσκος vs. Κύρια μνήμη

Δίσκος

- ▶ Αργή αναζήτηση (8-15 msec)
 - ▷ Περιστροφές: 5000-15000 rpm
- ▶ Ανθεκτικός
 - ▷ Υποθέτουμε πως αφού αποθηκευτούν στον δίσκο, τα δεδομένα είναι ασφαλή!
- ▶ Φθηνός



<https://www.youtube.com/watch?v=h-2eiLFkBFw>





Υψηλό επίπεδο: Δίσκος vs. Κύρια μνήμη

Μνήμη τυχαίας προσπέλασης (Random Access Memory -RAM-) ή Κύρια Μνήμη:

- ▶ Γρήγορη
- ▶ Τυχαία προσπέλαση, διευθυνσιοδότηση με byte
 - ▷ ~10x γρηγορότερη για σειριακή προσπέλαση
 - ▷ ~100,000x γρηγορότερη για τυχαία προσπέλαση!
- ▶ Πτητική
 - ▷ Δεδομένα χάνονται αν π.χ. συμβεί κάποιο σφάλμα, κοπεί η ρευματοδότηση κτλ!
- ▶ Ακριβή
 - ▷ Για €100, παίρνεις 16GB RAM vs. 2TB δίσκου! (~125x)



Αριθμοί
καθυστέρησης που
κάθε μηχανικός
πρέπει να ξέρει

Χονδρικές
μετρήσεις σε ένα
τυπικό PC:

Εκτέλεση τυπική εντολή	1/1,000,000,000 sec = 1 nanosec
Φέρε από την L1 κρυφή μνήμη	0.5 nanosec
Φέρε από την L2 κρυφή μνήμη	7 nanosec
Κλείδωμα/ξεκλείδωμα Mutex	25 nanosec
Φέρε από την κύρια μνήμη	100 nanosec
Στείλε 2K διφιοσυλλαβές πάνω από δίκτυο 1Gbps	20,000 nanosec
Διάβασε 1MB σειριακά από τη μνήμη	250,000 nanosec
Αναζήτησε τοποθεσία του δίσκου (αναζήτηση)	8,000,000 nanosec
Διάβασε 1MB σειριακά από τον δίσκο	20,000,000 nanosec
Στείλε ένα πακέτο από ΗΠΑ στην Ευρώπη και πίσω	150 milliseconds = 150,000,000 nanosec

Αριθμοί
καθυστέρησης που
κάθε μηχανικός
πρέπει να ξέρει

Οι προηγούμενες
τιμές αν τις δούμε
σε “ανθρώπινη
κλίμακα”

1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 µs	2-6 days
Rotational disk I/O	1-10 ms	1-12 months
Internet: SF to NYC	40 ms	4 years
Internet: SF to UK	81 ms	8 years
Internet: SF to Australia	183 ms	19 years
OS virtualization reboot	4 s	423 years
SCSI command time-out	30 s	3000 years
Hardware virtualization reboot	40 s	4000 years
Physical system reboot	5 m	32 millenia



Υψηλό επίπεδο: Δίσκος vs. Κύρια μνήμη

- ▶ Κρατήστε κατά νου τα εξής **tradeoffs** ως κίνητρα για τους μηχανισμούς που θα παρουσιάσουμε
- ▶ **Κύρια μνήμη:** Γρήγορη αλλά με μειωμένη χωρητικότητα και πτητική
- ▶ Vs. **Δίσκος:** αργός αλλά με μεγάλη χωρητικότητα και ανθεκτικός

Πως αξιοποιούμε αποτελεσματικά **και τα δύο** εξασφαλίζοντας συγκεκριμένες σημαντικές εγγυήσεις;

- ▶ Επιδόσεις Βασικής Αποθήκευσης
- ▶ **Το μοντέλο του ΣΔΒΔ**
- ▶ Συναλλαγές
 - ▷ Βασικά των Συναλλαγών
 - ▷ Κίνητρα για συναλλαγές
 - ▷ Ιδιότητες Συναλλαγών (ACID)

Το μοντέλο του ΣΔΒΔ

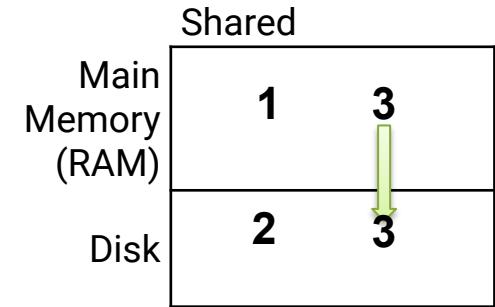


Το μοντέλο μας

Κοινόχρηστη: Κάθε διεργασία μπορεί να διαβάσει από / γράψει σε κοινόχρηστα δεδομένα στην κύρια μνήμη

Δίσκος: Η μνήμη μπορεί να διαβάσει από / γράψει στον δίσκο

Καταγραφή (Log): Υποθέτουμε σε σταθερή αποθήκευση σε δίσκο - εκινάει από την κύρια μνήμη και περνάει στον δίσκο...



Η Καταγραφή ενός Log γίνεται από την μνήμη→στον→ δίσκο

“Flush to disk” = γράψιμο στον δίσκο από την κύρια μνήμη

- ▶ Επιδόσεις Βασικής Αποθήκευσης
 - ▷ Το μοντέλο του ΣΔΒΔ
- ▶ Συναλλαγές
 - ▷ Βασικά των Συναλλαγών
 - ▷ Κίνητρα για συναλλαγές
 - ▷ Ιδιότητες Συναλλαγών (ACID)

Συναλλαγές

Βασικά των Συναλλαγών

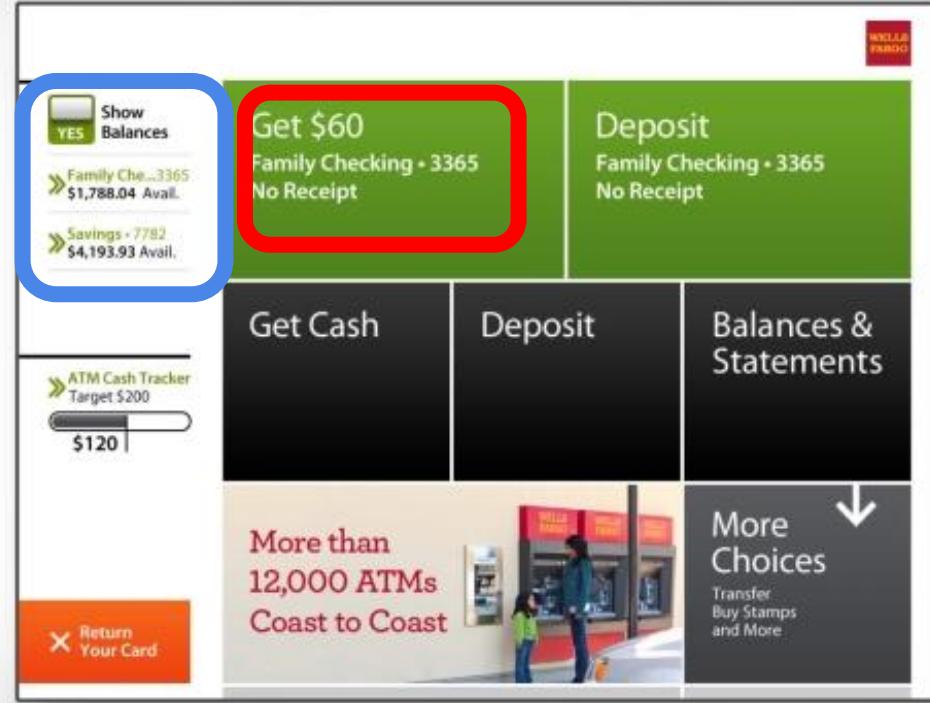
Παράδειγμα

Ανάλυση ΒΔ ΑΤΜ: Συναλλαγή

Διαβάζουμε υπόλοιπο
Δίνουμε λεφτά
Ενημερώνουμε υπόλοιπο

vs

Διαβάζουμε υπόλοιπο
Ενημερώνουμε υπόλοιπο
Δίνουμε λεφτά



WELLS
FARGO

Στόχοι

Οι **Συναλλαγές** είναι μια προγραμματιστική αφαίρεση που επιτρέπει στο ΣΔΒΔ να χειρίζεται την ανάκτηση και τον ταυτοχρονισμό για τους χρήστες.

Εφαρμογή: Οι συναλλαγές είναι κρίσιμες για τους χρήστες

Βασικές Αρχές: Τα βασικά του **πώς** δουλεύουν οι συναλλαγές (TXNs)

- ▶ Ο τρόπος διαχείρισης των συναλλαγών είναι θέμα για συζήτηση σχετικά με τα SQL, noSQL, NewSQL συστήματα.
- ▶ Πως δουλεύουν οι συναλλαγές (TXNs) ; Ποιά είναι τα tradeoffs?



Συναλλαγές: Βασικός Ορισμός

Συναλλαγή ("TXN")

Μια ακολουθία από μία ή περισσότερες λειτουργίες (διάβασμα ή γράψιμο) η οποία αντανακλά μια ενιαία πραγματική μετάβαση.

Στον πραγματικό κόσμο, μια συναλλαγή (TXN) είτε ολοκληρώθηκε είτε δεν συνέβη καθόλου

```
START TRANSACTION
```

```
Insert Into company Values  
( 'Panasonic', 25, 'USA' )
```

Λειτουργία 1

```
UPDATE Product  
SET Manufacturer = 'Panasonic'  
WHERE pname = 'Gizmo'
```

Λειτουργία 2

```
COMMIT
```



Συναλλαγές: Βασικός Ορισμός

Συναλλαγή (“ΤΧΝ”)

Μια ακολουθία από μία ή περισσότερες λειτουργίες (διάβασμα ή γράψιμο) η οποία αντανακλά μια ενιαία πραγματική μετάβαση.

Στον πραγματικό κόσμο, μια συναλλαγή (ΤΧΝ) είτε ολοκληρώθηκε είτε δεν συνέβη καθόλου

Παραδείγματα

“Μεταφορά χρημάτων μεταξύ λογαριασμών”

“Αγορά ενός συνόλου προϊόντων”

“Εγγραφή σε μια τάξη (είτε σε λίστα αναμονής είτε δεσμευμένη)”



Συναλλαγές στην SQL

Στην “ad-hoc” SQL:

- ▶ Προκαθορισμένο: κάθε εντολή = μια συναλλαγή
- ▶ Σε ένα πρόγραμμα, πολλαπλές εντολές μπορούν να ομαδοποιηθούν ως μια συναλλαγή:

```
START TRANSACTION
```

```
    UPDATE Bank SET amount = amount - 100  
    WHERE name = 'Bob'
```

Εντολή 1

```
    UPDATE Bank SET amount = amount + 100  
    WHERE name = 'Joe'
```

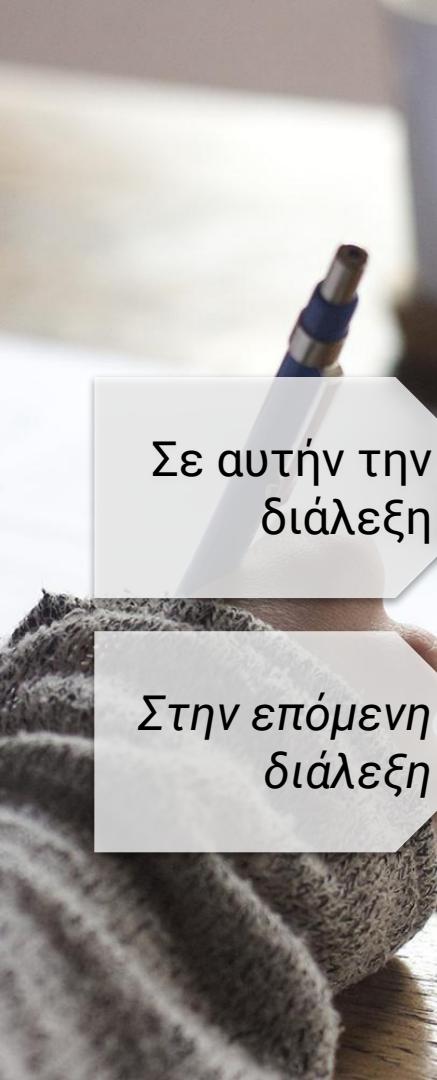
Εντολή 2

```
COMMIT
```

- ▶ Επιδόσεις Βασικής Αποθήκευσης
- ▶ Το μοντέλο του ΣΔΒΔ
- ▶ **Συναλλαγές**
 - ▷ Βασικά των Συναλλαγών
 - ▷ **Κίνητρα για συναλλαγές**
 - ▷ Ιδιότητες Συναλλαγών (ACID)

Συναλλαγές

Κίνητρα για συναλλαγές



Κίνητρα για τις Συναλλαγές

Η ομαδοποίηση των ενεργειών των χρηστών (διάβασμα & γράψιμο) σε συναλλαγές βοηθάει σε δύο σκοπούς:

- 1. Ανάνηψη & Μονιμότητα:** Διατηρούμε τα δεδομένα του ΣΔΒΔ σταθερά και ανθεκτικά σε αστοχίες, ματαιώσεις, απενεργοποιήσεις συστημάτων, κλπ.
- 2. Ταυτοχρονισμός:** Πετυχαίνουμε καλύτερη απόδοση κάνοντας τις συναλλαγές (TXNs) παράλληλα χωρίς να δημιουργούμε ανωμαλίες

Σε αυτήν την
διάλεξη

Στην επόμενη
διάλεξη



Κίνητρο: Ανάνηψη & Μονιμότητα

1.Η Ανάνηψη & Μονιμότητα των δεδομένων των χρηστών είναι απαραίτητη για την αξιόπιστη χρήση των ΣΔΒΔ

- ▶ Στο ΣΔΒΔ μπορεί να συμβούν ζημιές (crashes) (π.χ. Διακοπές ρεύματος, κλπ.)
- ▶ Μεμονωμένες συναλλαγές (TXNs) ενδεχομένως να εγκαταλειφθούν (π.χ. Από τον χρήστη)

Ιδέα: Να σιγουρευτούμε ότι οι συναλλαγές (TXNs) είτε αποθηκεύονται πάντοτε πλήρως ή καθόλου· κράτα καταγραφές (logs) για να μπορείς να “γυρίσεις πίσω” τις συναλλαγές (TXNs)

Προστασία εναντίον κόλλημάτων/ ματαιώσεων

Θέλω να φτιάξω
έναν πίνακα με
προϊόντα μικρού
μεγέθους

```
INSERT INTO      SmallProduct(name, price)
                SELECT  pname, price
                FROM    Product
                WHERE   price <= 0.99
```

κόλλημα / ματαιώση!

```
DELETE  Product
WHERE   price <= 0.99
```

Τι πάει στραβά;

⇒ Θα υπάρχει το ίδιο προϊόν και στους δύο
πίνακες

Προστασία εναντίον κολλημάτων/ ματαιώσεων

Θέλω να φτιάξω έναν πίνακα με προϊόντα μικρού μεγέθους

START TRANSACTION

```
INSERT INTO      SmallProduct(name, price)
                SELECT  pname, price
                FROM    Product
                WHERE   price <= 0.99
```

```
DELETE  Product
WHERE   price <=0.99
```

COMMIT OR ROLLBACK

Τώρα είναι εντάξει! Θα δούμε πως / γιατί σε αυτήν την διάλεξη



Κίνητρο: Ταυτόχρονη Εκτέλεση Προγραμμάτων

2. Η Ταυτόχρονη εκτέλεση των προγραμμάτων των χρηστών είναι απαραίτητη για την καλή απόδοση του ΣΔΒΔ.

- ▶ Οι προσπελάσεις στον δίσκο μπορεί να είναι συχνές και αργές-βελτιώνουμε τον ρυθμό διεκπεραίωσης αυξάνοντας τον # συναλλαγών αντισταθμίζοντας την καθυστέρηση (χρόνος για κάθε συναλλαγή)
- ▶ Οι χρήστες πρέπει να μπορούν να εκτελούν συναλλαγές σαν να είναι μόνοι τους. Έτσι, διατηρείται η συνέπεια.

Ιδέα: Το ΣΔΒΔ να τρέχει τις συναλλαγές πολλών χρηστών ταυτόχρονα, ώστε να έχει τους επεξεργαστές (CPUs) να δουλεύουν...



Πολλαπλοί χρήστες: μοναδικές εντολές

```
/*Client 1:*/ UPDATE Product
    SET Price = Price - 1.99
    WHERE pname = 'Gizmo'

/*Client 2:*/ UPDATE Product
    SET Price = Price*0.5
    WHERE pname='Gizmo'
```

Δύο διαχειριστές προσπαθούν να κάνουν έκπτωση στα προϊόντα ταυτόχρονα
- Τι μπορεί να πάει στραβά αν δεν έχω transactions;



Πολλαπλοί χρήστες: μοναδικές εντολές

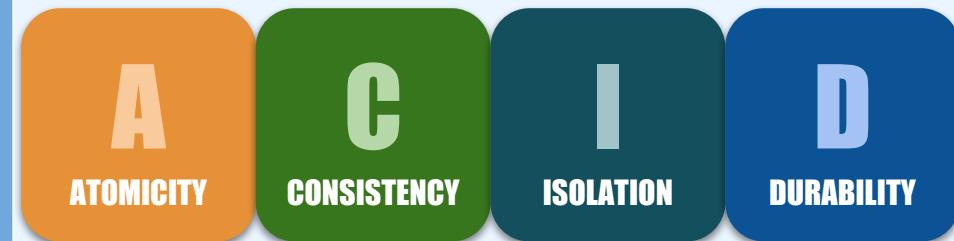
```
/*Client 1:*/ START TRANSACTION
    UPDATE Product
        SET Price = Price - 1.99
    WHERE pname = 'Gizmo'
    COMMIT

/*Client 2:*/ START TRANSACTION
    UPDATE Product
        SET Price = Price*0.5
    WHERE pname='Gizmo'
    COMMIT
```

Τώρα λειτουργεί μια χαρά- θα δούμε πως / γιατί στην επόμενη διάλεξη...

- ▶ Επιδόσεις Βασικής Αποθήκευσης
- ▶ Το μοντέλο του ΣΔΒΔ
- ▶ Συναλλαγές
 - ▷ Βασικά των Συναλλαγών
 - ▷ Κίνητρα για συναλλαγές
- ▶ **Ιδιότητες Συναλλαγών (ACID)**

Ιδιότητες Συναλλαγών





Ιδιότητες Συναλλαγών : ACID

Atomic: Η κατάσταση μιας βάσης περιέχει όλες τις αλλαγές μιας συναλλαγής, ή καμία απολύτως

Consistent: Η συναλλαγή μετακινείται από κατάσταση που υπάρχει ακεραιότητα σε μία άλλη που διατηρείται η ακεραιότητα

Isolated: Τα αποτελέσματα των συναλλαγών είναι τα ίδια με το να τρέξουν οι συναλλαγές η μία μετά την άλλη

Durable: Όταν τελειώσει μία συναλλαγή, τα αποτελέσματά της παραμένουν στην βάση

To ACID συνεχίζει να είναι θέμα μεγάλου διαλόγου!



ACID: (Atomicity) Ατομικότητα

- ▶ Οι TXN's δραστηριότητες είναι ατομικές: **όλα ή τίποτα**

Διαισθητικά: στον πραγματικό κόσμο, μια συναλλαγή είναι κάτι που είτε θα συμβεί ολοκληρωτικά ή καθόλου.
- ▶ Δύο είναι τα πιθανά αποτελέσματα για μια TXN
 - ▷ Τερματίζεται (commits): όλες οι αλλαγές γίνονται
 - ▷ Εγκαταλείπει (aborts): καμία αλλαγή δεν γίνεται



ACID: (Consistency) Συνέπεια

- ▶ Οι πίνακες πρέπει πάντα να ικανοποιούν τους **περιορισμούς ακεραιότητας** που έχουν δημιουργήσει οι χρήστες
 - “Το λογιστικό υπόλοιπο είναι μοναδικό”
 - “Το κεφάλαιο δεν μπορεί να είναι αρνητικό”
 - “Το άθροισμα των χρεώσεων και των πιστώσεων πρέπει να είναι 0”
- ▶ Πως επιτυγχάνεται η συνέπεια:
 - ▷ Ο χρήστης γράφει μία συναλλαγή η οποία μεταβαίνει από μία **συνεπή κατάσταση** της βάσης σε μία άλλη **συνεπή κατάσταση** της βάσης
 - ▷ Το σύστημα φροντίζει η συναλλαγή να είναι **συνεπής**



ACID: (Isolation) Απομόνωση

- ▶ Μια συναλλαγή εκτελείται ταυτόχρονα με άλλες συναλλαγές
- ▶ **Απομόνωση:** το αποτέλεσμα είναι σαν να εκτελείται κάθε συναλλαγή απομονωμένα και ανεξάρτητα των υπολοίπων.

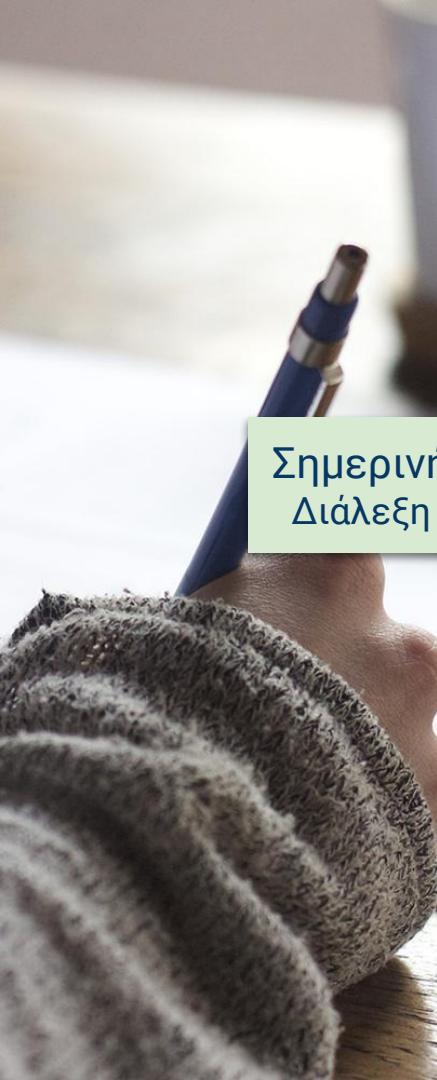
Π.χ Δε θα πρέπει να φαίνονται στον χρήστη οι αλλαγές από τις άλλες συναλλαγές κατά τη διάρκεια που αλληλεπιδρά με το σύστημα.



ACID: (Durability) Μονιμότητα

- ▶ Το αποτέλεσμα μιας συναλλαγής πρέπει να συνεχίσει να υπάρχει μετά τη συναλλαγή (TXN)
 - ▷ Και αφότου το όλο πρόγραμμα έχει τερματιστεί
 - ▷ Ακόμα και αν συμβούν διακοπές ενέργειας, λειτουργίας κλπ.
 - ▷ Και άλλα...
- ▶ Σημαίνει: Εγγραφή των δεδομένων στον δίσκο

Αλλαγές στον ορίζοντα;
Μη-Πτητική Ram (NVRam). Διευθυνσιοδότηση με byte.



Προκλήσεις για τις ιδιότητες ACID

Σημερινή
Διάλεξη

- ▶ Λειτουργία του ΣΔΒΔ παρά τις αποτυχίες:
Αποτυχίες ηλεκτροδότησης, αλλά όχι αποτυχίες μέσων
- ▶ Οι χρήστες ενδεχομένως να εγκαταλείψουν το πρόγραμμα: πρέπει να “επαναφέρουμε τις αλλαγές”
 - ▷ Πρέπει να καταγράψουμε (log) τι συνέβη
- ▶ Πολλοί χρήστες εκτελούν ταυτόχρονα
 - ▷ Μπορεί να επιλυθεί με κλείδωμα (Θα το δούμε σε επόμενη διάλεξη!)

Και όλα αυτά με... Υψηλές επιδόσεις!!

Μια σημείωση: το ACID είναι συνεχώς υπό συζήτηση!

- ▶ Πολλές συζητήσεις με θέμα το ACID, και Ιστορικά αλλά και Επίκαιρα
- ▶ Πολλές νέες “NoSQL” ΣΒΔ δεδομένων χαλαρώνουν το ACID
- ▶ Σε αντίθεση, τώρα η “NewSQL” ξαναεισάγει την συμμόρφωση με το ACID σε σχέση με τα NoSQL-τύπου ΣΒΔ...



To ACID είναι ένα εξαιρετικά σημαντικό & επιτυχημένο μοντέλο , αλλά ακόμα συζητείται!

Καταγραφή (Logging)

2ο Μέρος

Εισαγωγή σε Logging

Εισαγωγή

Write-Ahead Logging (WAL)

Παράδειγμα

Συσταδοποιημένο Μοντέλο



Καταγραφή (Log)

- ▶ Είναι μια λίστα ενεργειών
 - ▷ Δημιουργείται διπλότυπο καταγραφής και αρχειοθετείται σε **σταθερό μέσο αποθήκευσης**.
 - ▷ Μπορούμε να **εξαναγκάσουμε** την εγγραφή των καταγραφών στο δίσκο.
⇒ Μια σελίδα πηγαίνει στο δίσκο.
 - ▷ Όλες οι δραστηριότητες καταγραφής διαχειρίζονται με διαφάνεια από το ΣΔΒΔ.

Υποθέστε ότι
δεν το χάνουμε!



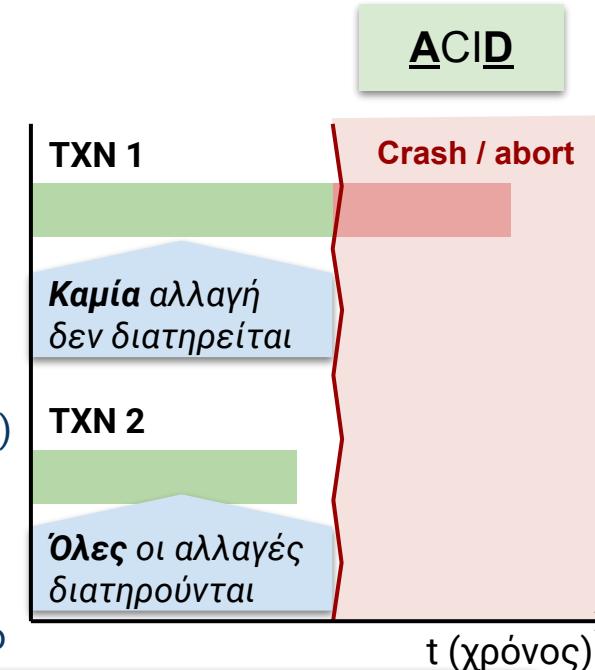
Στόχος σε αυτή τη διάλεξη: Εξασφάλιση Ατομικότητας & Μονιμότητας

Ατομικότητα

- ▶ Οι συναλλαγές πρέπει να ολοκληρωθούν ή να μην συμβούν καθόλου
- ▶ Εάν εγκαταλειφθούν / αποτυχία κατά την διάρκεια της TXN, δεν πρέπει να φανούν αλλαγές

Μονιμότητα

- ▶ Εάν ένα σύστημα Βάσεων δεδομένων (ΣΒΔ) σταματήσει να λειτουργεί, οι αλλαγές στις ολοκληρωμένες TXNs πρέπει να διατηρηθούν
- ▶ Απλά αποθηκεύουμε σε ένα σταθερό δίσκο



Θα εστιάσουμε στο πως θα επιτύχουμε ατομικότητα με καταγραφή (logging)



Βασική Ιδέα: (Φυσική) Καταγραφή

- ▶ Καταγραφή UNDO πληροφορίας για κάθε ενημέρωση!
 - ▷ Σειριακό γράψιμο στο log
 - ▷ Γράψιμο μόνο της πρόσθετης πληροφορίας στο log
- ▶ Το **log** αποτελείται από μία **διατεταγμένη λίστα ενεργειών**
 - ▷ Μία εγγραφή στο log περιέχει:
 $\langle \text{XID}, \text{location}, \text{old data}, \text{new data} \rangle$

Αυτό επαρκεί για να κάνουμε UNDO οποιαδήποτε συναλλαγή!



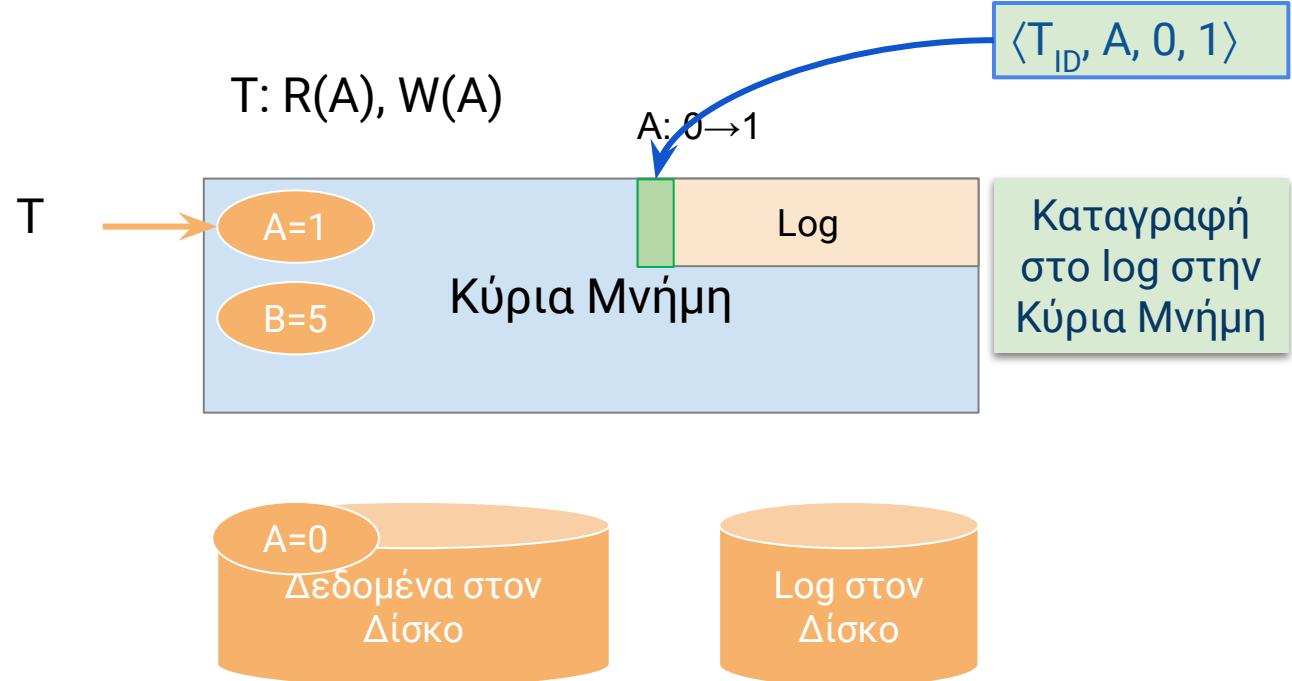
Logging

T: R(A), W(A)





Logging





Ποιος είναι ο σωστός τρόπος να γραφτεί στον δίσκο;;

- ▶ Θα δούμε το Write-Ahead Logging (WAL) πρωτόκολλο
- ▶ Θα δούμε γιατί δουλεύει συγκρίνοντας με άλλα πρωτόκολλα που είναι λανθασμένα!

Θυμηθείτε: Βασική ιδέα είναι να εξασφαλίσουμε Μονιμότητα διατηρώντας την δυνατότητα να κάνουμε “undo”!



Γιατί χρειαζόμαστε logging για την Ατομικότητα;

- ▶ Δεν μπορούσαμε απλώς να γράφουμε κάθε TXN στον δίσκο μόνο εφόσον ολοκληρώθηκε όλη;;
 - ▷ Επομένως, εάν μία TXN εγκαταλείψει/διακοπεί χωρίς να ολοκληρωθεί, δεν θα έχει αποτέλεσμα- ατομικότητα!
 - ▷ Κάτι τέτοιο θα λειτουργούσε με απεριόριστη μνήμη και χρόνο...
- ▶ Παρόλα αυτά, **χρειάζεται να κρατάμε logs για επιμέρους αποτελέσματα** των TXNs λόγω των:
 - ▷ Περιορισμών στη μνήμη (υπάρχει αρκετός χώρος για όλο το TXN;;)
 - ▷ Περιορισμοί χρόνου (τι γίνεται εάν μία TXN διαρκέσει πολλή ώρα;)

Άρα πρέπει να γράφουμε και τα ενδιάμεσα, μερικά αποτελέσματα στον δίσκο!
...Δηλαδή χρειαζόμαστε ένα log για να μπορούμε να κάνουμε undo αυτά τα ενδιάμεσα!

Εισαγωγή

Write-Ahead Logging (WAL)

Παράδειγμα

Συσταδοποιημένο Μοντέλο

Write-Ahead Logging



Διαδικασία για Commit (Επικύρωση)

1. Κάθε εγγραφή ενός commit στο log της μνήμης μεταφέρεται (flushed) στο log του δίσκου
2. Όλες καταγραφές μιας συναλλαγής στο log της μνήμης από το τελευταίο μεταφέρονται (flushed) στον δίσκο
3. Ολοκληρώνεται το **Commit()**

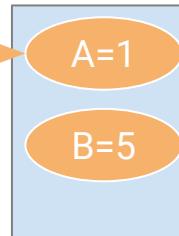
Η Συναλλαγή γίνεται commit μόνο όταν η log εγγραφή είναι αποθηκευμένη σε δίσκο

Λανθασμένο

Commit πρωτόκολλο #1

T: R(A), W(A)

T



A: 0→1 Commit

Κύρια Μνήμη



Ας δοκιμάσουμε να κάνουμε commit προτού αποθηκεύσουμε δεδομένα και log στον δίσκο...

OK, Commit!

Εάν υπάρξει αποτυχία, θα έχει η T μονιμότητα;



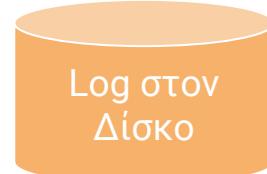
Λανθασμένο

Commit πρωτόκολλο #1

T: R(A), W(A)

T

Κύρια Μνήμη



Ας δοκιμάσουμε να κάνουμε commit προτού αποθηκεύσουμε δεδομένα και log στον δίσκο...

OK, Commit!

Εάν υπάρχει αποτυχία, θα έχει η T μονιμότητα;

Χάθηκε το update της T!

Λανθασμένο

Commit πρωτόκολλο #2

T: R(A), W(A)



Ας δοκιμάσουμε να κάνουμε commit **αφότου** γράψουμε τα δεδομένα στον δίσκο αλλά πριν καταγράψουμε log...

OK, Commit!

Εάν σκάσει τώρα θα έχει η T μονιμότητα; Ναι! Μόνο που....



Λανθασμένο

Commit πρωτόκολλο #2

T: R(A), W(A)

T

Κύρια Μνήμη

A=1

Δεδομένα στον
Δίσκο

Log στον
Δίσκο

Ας δοκιμάσουμε να κάνουμε commit **αφότου** γράψουμε τα δεδομένα στον δίσκο αλλά πριν καταγράψουμε log...

OK, Commit!

Εάν σκάσει τώρα θα έχει η T μονιμότητα; Ναι! Μόνο που....

**Πως ξέρουμε εάν η T έκανε commit;;
⇒ Λείπει η αντίστοιχη καταγραφή από το log του δίσκου**

Write-ahead Logging (WAL) Commit



Τώρα, ας δοκιμάσουμε να κάνουμε commit αφότου έχουμε καταγράψει log στον δίσκο αλλά προτού γράψουμε τα δεδομένα στον δίσκο... αυτό είναι WAL!

OK, Commit!

Εάν σκάσει τώρα, θα έχει η Τ μονιμότητα;

Write-ahead Logging (WAL) Commit

T: R(A), W(A)

T

Κύρια Μνήμη

A=0

Δεδομένα στον
Δίσκο

A: 0→1

Log στον
Δίσκο

Τώρα, ας δοκιμάσουμε να κάνουμε commit **αφότου** **έχουμε καταγράψει log στον δίσκο** αλλά **προτού γράψουμε τα δεδομένα στον δίσκο...** αυτό είναι WAL!

OK, Commit!

Εάν σκάσει τώρα, θα έχει η T μονιμότητα;

KOITAME TO LOG!

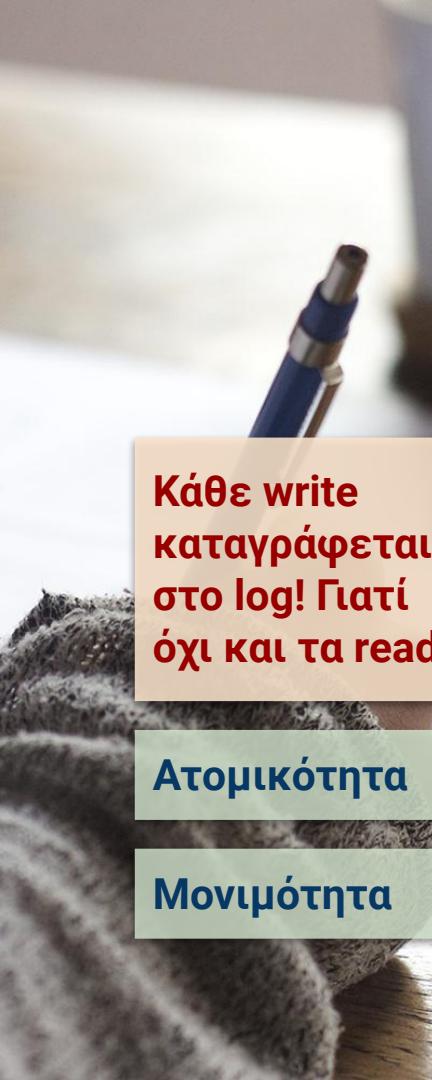
Write-Ahead Logging (WAL)

- Κάθε αλλαγή (π.χ. Update πλειάδας), καταγράφεται στην **Log της RAM**
- Πριν μεταφερθεί μία αλλαγμένη σελίδας από την ενδιάμεσης μνήμης στον δίσκο: Μετέφερε τις **καταγεγραμμένες αλλαγές** στο **Log του Δίσκου**
- Πριν το commit μιας συναλλαγής
 - a. Μεταφέρω όλες οι **καταγεγραμμένες αλλαγές** στο **Log του Δίσκου**
 - b. Καταγράφω το Commit της συναλλαγής στο **Log του Δίσκου**

Κάθε write
καταγράφεται
στο log! Γιατί όχι
και τα read?

⇒ Ατομικότητα

⇒ Μονιμότητα



Write-Ahead Logging (WAL)

Κάθε write
καταγράφεται
στο log! Γιατί
όχι και τα read?

Ατομικότητα

Μονιμότητα

- ▶ Κάθε αλλαγή (π.χ. Update πλειάδας), καταγράφεται στην **Log της RAM**
- ▶ Πριν μεταφερθεί μία αλλαγμένη σελίδας από την ενδιάμεσης μνήμης στον δίσκο
 - a. Μεταφέρονται οι **καταγεγραμμένες αλλαγές** για την σελίδα:
Log της RAM → Log του Δίσκου
- ▶ Πριν το commit μιας συναλλαγής
 - a. Μεταφέρονται όλες οι **καταγεγραμμένες αλλαγές** της:
Log της RAM → Log του Δίσκου
 - b. Μεταφέρεται το Commit της συναλλαγής στο **Log του Δίσκου**

Εισαγωγή

Write-Ahead Logging (WAL)

Παράδειγμα

Συσταδοποιημένο Μοντέλο

Παράδειγμα



Write-Ahead Logging (WAL)

Η Βάση Δεδομένων χρησιμοποιεί
Write-Ahead Logging (WAL) Πρωτόκολλο:

1. Πρέπει να κάνει καταγραφή της log εγγραφής ώστε να ενημερωθεί το log πριν αποθηκευτούν τα αντίστοιχα δεδομένα στο δίσκο.
2. Πρέπει να εισάγει όλες τις log εγγραφές για μια συναλλαγή (TX) πριν το commit.

Ατομικότητα

Μονιμότητα

Παράδειγμα: Μηνιαίος τόκος τραπεζικών συναλλαγών

Χρήματα (@4:29 πμ)

Account	Balance (\$)
3001		500
4001		100
5001		20
6001		60
3002		80
4002		-200
5002		320
...	...	
30108		-100
40008		100
50002		20

Χρήματα (@4:29 πμ ημέρα+1)

Account	Balance (\$)
3001		550
4001		110
5001		22
6001		66
3002		88
4002		-220
5002		352
...	...	
30108		-110
40008		110
50002		22



WAL (@4:29 πμ ημέρα+1)

T-Monthly-423	START TRANSACTION		
T-Monthly-423	3001	500	550
T-Monthly-423	4001	100	110
T-Monthly-423	5001	20	22
T-Monthly-423	6001	60	66
T-Monthly-423	3002	80	88
T-Monthly-423	4002	-200	-220
T-Monthly-423	5002	320	352
T-Monthly-423
T-Monthly-423	30108	-100	-110
T-Monthly-423	40008	100	110
T-Monthly-423	50002	20	22
T-Monthly-423	COMMIT		

'T-Monthly-423' (Μηνιαίο επιτόκιο 10%)

➤Στις 4:28 πμ ξεκινά να "τρέχει" σε 10M τραπεζικούς λογαριασμούς. ➤Χρειάζεται 24 ώρες να "τρέξει"

START TRANSACTION
UPDATE Money
SET Amt = Amt * 1.1
COMMIT

Παράδειγμα: Μηνιαίος τόκος τραπεζικών συναλλαγών (Διακοπή)

Χρήματα (@4:29 πμ)

Account	Balance (\$)
3001		500
4001		100
5001		20
6001		60
3002		80
4002		-200
5002		320
...	...	
30108		-100
40008		100
50002		20



Χρήματα (@10:45 πμ)

Account	Balance (\$)
3001		550
4001		110
5001		22
6001		66
3002		88
4002		-200
5002		320
...	...	
30108		-110
40008		110
50002		22

??

??

??

??

WAL (@10:29 πμ)

T-Monthly-423	START TRANSACTION		
T-Monthly-423	3001	500	550
T-Monthly-423	4001	100	110
T-Monthly-423	5001	20	22
T-Monthly-423	6001	60	66
T-Monthly-423	3002	80	88
T-Monthly-423
T-Monthly-423	30108	-100	-110
T-Monthly-423	40008	100	110
T-Monthly-423	50002	20	22
T-Monthly-423	4002	-200	-220
T-Monthly-423	5002	320	352

'T-Monthly-423' (Μηνιαίο επιτόκιο 10%)

➤Στις 4:28 πμ ξεκινά να "τρέχει" σε 10M τραπεζικούς λογαριασμούς. ➤Χρειάζεται 24 ώρες να "τρέξει"

Διακοπή στο δίκτυο στις 10:29 πμ,
Το σύστημα επανέρχεται στις 10:45 πμ

Τελικά η T-Monthly-423 ολοκληρώθηκε;
Ποιες πλειάδες είναι αλλοιωμένες;

Περίπτωση 1: T-Monthly-423 διακόπηκε.

Περίπτωση 2: T-Monthly-423 ολοκληρώθηκε. Στον 4002 κατατέθηκαν 20\$ στις 10:45 πμ

Παράδειγμα: Μηνιαίος τόκος τραπεζικών συναλλαγών (Ανάκτηση)

Χρήματα (@10:45 πμ)

Account	Balance (\$)
3001		550
4001		110
5001		22
6001		66
3002		88
4002		-200
5002		320
...		
30108		-110
40008		110
50002		22

Χρήματα (μετά την ανάκτηση)

Account	Balance (\$)
3001		500
4001		100
5001		20
6001		60
3002		80
4002		-200
5002		320
...		
30108		-100
40008		100
50002		20

WAL (@10:29 πμ)

T-Monthly-423	START TRANSACTION		
T-Monthly-423	3001	500	550
T-Monthly-423	4001	100	110
T-Monthly-423	5001	20	22
T-Monthly-423	6001	60	66
T-Monthly-423	3002	80	88
T-Monthly-423
T-Monthly-423	30108	-100	-110
T-Monthly-423	40008	100	110
T-Monthly-423	50002	20	22
T-Monthly-423	4002	-200	-220
T-Monthly-423	5002	320	352

Επαναφορά Συστήματος (μετά τις 10:45 πμ)

- Αναίρεση(Rollback) συναλλαγών που δεν έγιναν commit
- Επαναπραγματοποίηση(Redo) των Πρόσφατων συναλλαγών (w/ νέες τιμές)
- Ξανά σε λειτουργία
- Επαναπραγματοποίηση(Redo) εκκρεμών συναλλαγών

Ανάκτηση παλαιών τιμών από WALlog
(εάν υπάρχουν)
Ειδοποίηση των developers για την διακοπή

Παράδειγμα: Μηνιαίος τόκος τραπεζικών συναλλαγών (Επιδόσεις)

Χρήματα (@4:29 πμ)

Account	Balance (\$)
3001		500
4001		100
5001		20
6001		60
3002		80
4002		-200
5002		320
...	...	
30108		-100
40008		100
50002		20

Χρήματα (@4:29 πμ ημέρα+1)

Account	Balance (\$)
3001		550
4001		110
5001		22
6001		66
3002		88
4002		-220
5002		352
...	...	
30108		-110
40008		110
50002		22

WAL (@4:29 πμ ημέρα+1)

T-Monthly-423	START TRANSACTION		
T-Monthly-423	3001	500	550
T-Monthly-423	4001	100	110
T-Monthly-423	5001	20	22
T-Monthly-423	6001	60	66
T-Monthly-423	3002	80	88
T-Monthly-423	4002	-200	-220
T-Monthly-423	5002	320	352
T-Monthly-423
T-Monthly-423	30108	-100	-110
T-Monthly-423	40008	100	110
T-Monthly-423	50002	20	22
T-Monthly-423	COMMIT		

Κόστος ανανέωσης όλων των δεδομένων

10M τραπεζικοί λογαριασμοί → 10M αναζήτησεις;
(στη χειρότερη περίπτωση)
(*10 msec/αναζήτηση, που είναι 100,000 δευτ.)

Επιτάχυνση για τα commit

100,000 δευτ έναντι 1 δευτ!!!

Κόστος επέκτασης των log

+1 αναζήτηση για να βρούμε 'τέλος του log'
+ γράψιμο 10M log εγγραφών σειριακά (γρήγορο!)
< 1 δευτ)
["Αργή" ανανέωση των δεδομένων στο δίσκο αργότερα, όποτε είναι βιολικό.]

Ανασκόπηση Logging

- ▶ Εάν η βάση γράφει TXN **commits**, το αποτέλεσμα της TXN **παραμένει** και μετά από crash της βάσης.
- ▶ Η βάση μπορεί και κάνει **undo ενέργειες** και μας βοηθάει με την **Ατομικότητα**
- ▶ Μέχρι τώρα έχουμε πει τα μισά...

Εισαγωγή
Write-Ahead Logging (WAL)
Παράδειγμα
Συσταδοποιημένο Μοντέλο

Συσταδοποιημένο Μοντέλο

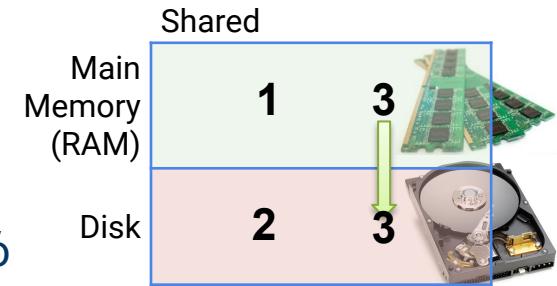
Το πρωταρχικό μας μοντέλο (ανασκόπηση)



Κοινόχρηστη: Κάθε διεργασία μπορεί να διαβάσει από / γράψει σε κοινόχρηστα δεδομένα στην κύρια μνήμη

Δίσκος: Η μνήμη μπορεί να διαβάσει από ή να γράψει στον δίσκο

Καταγραφή (Log): Υποθέτουμε σε σταθερή αποθήκευση σε δίσκο- ξεκινάει από την κύρια μνήμη και περνάει στον δίσκο...



Η Καταγραφή ενός Log γίνεται από την μνήμη→στον→δίσκο

"Flush to disk" = γράψιμο στον δίσκο από την κύρια μνήμη

Συσταδοποιημένο Μοντέλο

Μια δημοφιλής εναλλακτική (με tradeoffs)



Commit καταχωρώντας τα logs ή/και τα δεδομένα σε Αντίγραφα (Replicas) σε 'n' μηχανήματα (π.χ. $n=2$)

- ▶ [Είτε στο ίδιο rack, είτε σε διαφορετικό rack είτε και σε διαφορετικό datacenter]

(Παράδειγμα [datacenter](#) από το 2:50)

Συσταδοποιημένο Μοντέλο

Επιδόσεις

Αυξανόμενο κόστος για εγγραφή σε μηχάνημα

- ▶ **Η ταχύτητα του εσωτερικού δικτύου στα datacenters μπορεί να είναι 1-10 ms**
- ▶ [Αναβολή ενημέρωσης δεδομένων στο δίσκο για αργότερα, όποτε βολεύει]

Μοντέλο Αποτυχίας

Κυρίως Μοντέλο: η RAM μπορεί να αποτύχει, ο δίσκος είναι πιο ανθεκτικός

Συσταδοποιημένο μοντέλο: "Σχεδίαση" του hardware να είναι ανθεκτικό στα σφάλματα

- ▶ Η RAM διαφορετικών μηχανημάτων δεν αποτυγχάνει ταυτόχρονα.
- ▶ Η παροχή ενέργεια σε κάθε rack είναι ανεξάρτητη από τα υπόλοιπα



Παράδειγμα: ΒΔ Youtube

YouTube

funny cats|

About 12,100,000 results

How to Get Rid of Cat Pee Stains
BISELL • 2M views
Your cat had an accident on the carpet. BISSELL is here to help!

REMOVE CAT PEE STAINS

CATS make us LAUGH ALL THE TIME! - Ultra FUNNY CAT
Tiger FunnyWorks • 100K views • 3 days ago
Ultra funny cats and kitten that will make you cry with laughter! Cats are the best laugh all the time! This is ...
New

You will LAUGH SO HARD that YOU WILL FAINT - FUNNY compilation
Tiger FunnyWorks • 19M views • 8 months ago
Well well well, cats for you again. But this time, even better, even funnier, even you like these furries the ...
New

Have you EVER LAUGHED HARDER? - Ultra FUNNY CATS
Tiger FunnyWorks • 124K views • 1 week ago
Super funny cats and kitten that will make you scream with laughter! This is the LAUGH challenge ever!

Upload video
(o) Go live

cats funny

Baby Cats 🐱 Funny and Cute Baby Cat Videos Compilation (2018) Gatitos Bebes Video Recopilacion
809,337 views

4.7K 768 SHARE

Animal Planet Videos • Published on May 23, 2018

SUBSCRIBE 337K

Baby Cats - Funny and Cute Baby Cat Videos Compilation (2018) Gatitos Bebes Video Recopilación
Animal Planet Videos
Subscribe Here: <https://goo.gl/qor4XN>

Up next

Top Cats Vs. Cucumbers 🥒Funny Cat Videos Compilation...
Animal Planet Videos • 23M views

Funny Elias play with the wheels on the bus and another toys - ...
Elias Adventures • 291 watching
LIVE NOW

LIVE: Rescue kitten nursery! TinyKittens HQ • 1.3K watching
LIVE NOW

Puss in boots and the three diablos [HD]
Mathew Garcia • 7.6M views

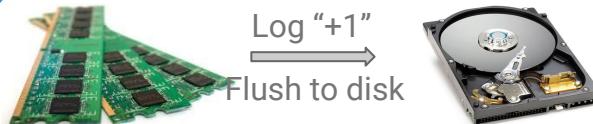
AUTOPLAY

Παράδειγμα: ΒΔ Youtube

Σχεδίαση 1

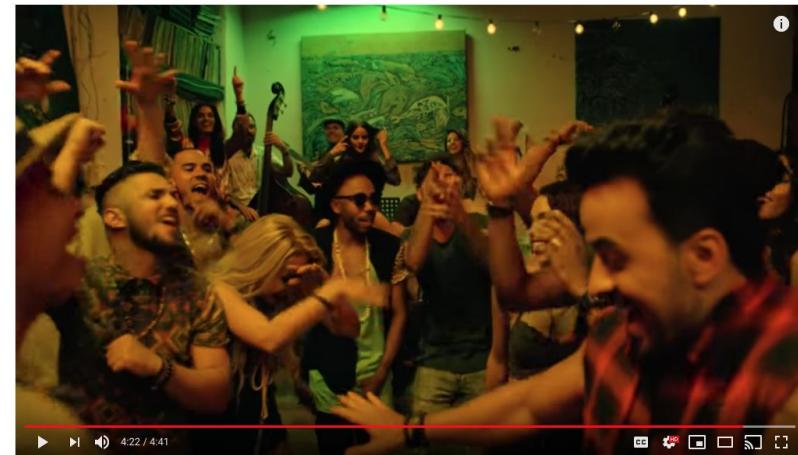
- ▶ WAL Log για προβολές Video
- ▶ ⟨video_id, old_#_views, new_#_views⟩

T-LIKE-4307	START TRANSACTION		
T-LIKE-4307	3001	537	538
T-LIKE-4307	COMMIT		
T-LIKE-4308	START TRANSACTION		
T-LIKE-4308	5309	100001	10002
T-LIKE-4308	COMMIT		
T-LIKE-4309	START TRANSACTION		
T-LIKE-4309	3001	538	539
T-LIKE-4309	COMMIT		
...
T-LIKE-4341	5309	100002	10003
T-LIKE-4351	5309	100003	10004
...
T-LIKE-4383	START TRANSACTION		



Κριτική:

- ▶ Είναι σωστό;
- ▶ Ταχύτητα εγγραφής; Κόστος; Αποθήκευση;
- ▶ Σημεία συμφόρησης (bottlenecks);



Log '+1'
5,611,744,868 views

ft. Daddy Yankee

1M 30M 3.5M SHARE SAVE ...

Παράδειγμα: ΒΔ Youtube

Σχεδίαση 2

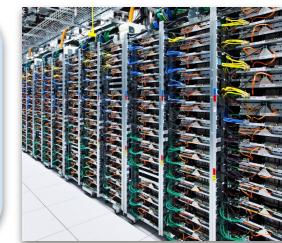
- ▶ Καταχώριση Πλήθους Προβολών βίντεο σε Αντίγραφα στο cluster
- ▶ ⟨unix_time, video_id, #_views⟩

1539893189	3001	'+1'
1539893195	5309	'+1'
1539893225	3001	'+1'
..		'+1'
	5309	'+1'
...		'+1'
	5309	'+1'
...		'+1'
1539893289	5309	'+1'

Κριτική:

- ▶ Είναι σωστό;
- ▶ Ταχύτητα εγγραφής; **Κόστος**; Αποθήκευση;
- ▶ Σημεία συμφόρησης;
- ▶ Επαναφορά Συστήματος;

- ▶ Εγγραφή στη RAM n=3 Μηχανήματα
- ▶ ⟨unix_time, video_id, #_views⟩

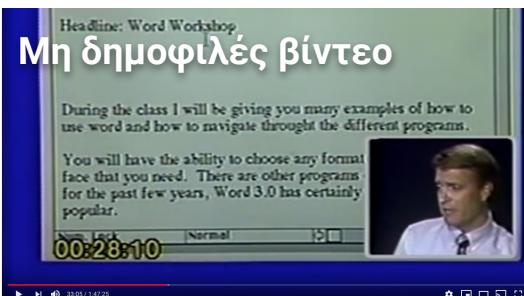


Παράδειγμα: ΒΔ YouTube

Επίδοση Έναντι Κόστους



Luis Fonsi - Despacito ft. Daddy Yankee



Σχεδίαση 3

- ▶ Για τα περισσότερα βίντεο, Σχεδίαση 1 (πλήρης WAL logs)
- ▶ Για τα δημοφιλή βίντεο, Σχεδίαση 2

Κριτική:

- ▶ Σωστό;
- ▶ Ταχύτητα εγγραφής; Κόστος; Αποθήκευση;
- ▶ Σημεία συμφόρησης;
- ▶ Ανάκτηση Συστήματος;

Ανακεφαλαίωση

Ερωτήσεις Σχεδιασμού

- ▶ **Ορθότητα:** Χρειαζόμαστε το ACID; ή ένα Ψευδο-ACID; Ποιες απώλειες είναι OK;
- ▶ **Σχεδιασμός παραμέτρων:**
Υπάρχουν ιδιότητες των δεδομένων που μπορούμε να εκμεταλλευτούμε; (π.χ., '+1', δημοφιλές ή όχι)
 - ▷ Πόση RAM, σκλ. δίσκοι και μηχανήματα;
 - ▷ Πόσες θα είναι τα writes ανά δευτερόλεπτο;
 - ▷ Πόσο γρήγορα θέλουμε το σύστημα να επανέλθει;
- ▶ **Επιλέγουμε:** WAL logs, Αντίγραφα σε n-μηχανήματα, Υβριδικά;

Ταντοχροισμός, Χρονοδιαγράμματα & Ανωμαλίες Ταντοχροισμού

Μέρος 30

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
- ▶ Ανωμαλίες σε Ταυτόχρονη Εκτέλεση
- ▶ Διενέξεις
 - ▶ Σειριοποιησιμότητα διενέξεων
 - ▶ Γράφος Διενέξεων
 - Dag & Τοπολογικές
- ▶ Ταξινομήσεις
- ▶ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Εισαγωγικά

Ηαράδειγμα



- ▶ "Η Visa πραγματοποιεί > 60,000 συναλλαγές/δευτερόλεπτο με τους χρήστες & τους εμπόρους"
- ▶ Θα θέλατε η συναλλαγή σας των 4\$ στα Starbucks να περιμένει για ένα στοίχημα των 10k\$ στο Las Vegas ?
- ▶ Οι συναλλαγές μπορούν (1) να είναι **γρήγορες** ή να **παίρνουν πολύ χρόνο**, (2) να μην σχετίζονται με εσάς

Παράδειγμα



Θα θέλατε η συναλλαγή σας να περιμένει;

Ή να γίνεται επεξεργασία πολλών; (όπως το unix χειρίζεται τις διεργασίες)

- ▶ **Ταυτοχρονισμός απομόνωση & συνέπεια**
 - ▷ Χρονοδιαγράμματα
- ▶ Ανωμαλίες σε Ταυτόχρονη Εκτέλεση
- ▶ Διενέξεις
 - ▷ Σειριοποιησιμότητα διενέξεων
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Ταυτοχρονισμός: Απομόνωση & Συνέπεια



Ταντοχρονισμός: Απομόνωση & Συνέπεια

Το ΣΔΒΔ πρέπει να χειριστεί την Συνέπεια έτσι ώστε...

ACID

1. **Η Απομόνωση** να διατηρείται: Οι χρήστες πρέπει να μπορούν να εκτελούν κάθε συναλλαγή **σαν να ήταν οι μοναδικοί χρήστες**
 - ▷ Το ΣΔΒΔ χειρίζεται τις λεπτομέρειες του σχεδιασμού χρονοδιαγραμμάτων διαφόρων συναλλαγών
2. **Η Συνέπεια** να διατηρείται: Οι συναλλαγές πρέπει να αφήνουν την ΒΔ σε μια **σταθερή κατάσταση**
 - ▷ Το ΣΔΒΔ χειρίζεται τις λεπτομέρειες της επιβολής περιορισμών ακεραιότητας

ACID

Σημείωση: Επιστρέφουμε στο απλό μοντέλο μιας μηχανής για μνήμη RAM / δίσκο



Σημειώστε ότι το δύσκολο κομμάτι...

...είναι ο σχεδιασμός **ταυτόχρονων** χρονοδιαγραμμάτων με
συναλλαγές και **αποτυχίες** (crashes).

Παράδειγμα- θεωρείστε δύο συναλλαγές

T1: START TRANSACTION

UPDATE Accounts

SET Amt = Amt + 100

WHERE Name = 'A'

UPDATE Accounts

SET Amt = Amt - 100

WHERE Name = 'B'

COMMIT

T1 μεταφέρει \$100 από τον λογαριασμό του B στον λογαριασμό του A

T2: START TRANSACTION

UPDATE Accounts

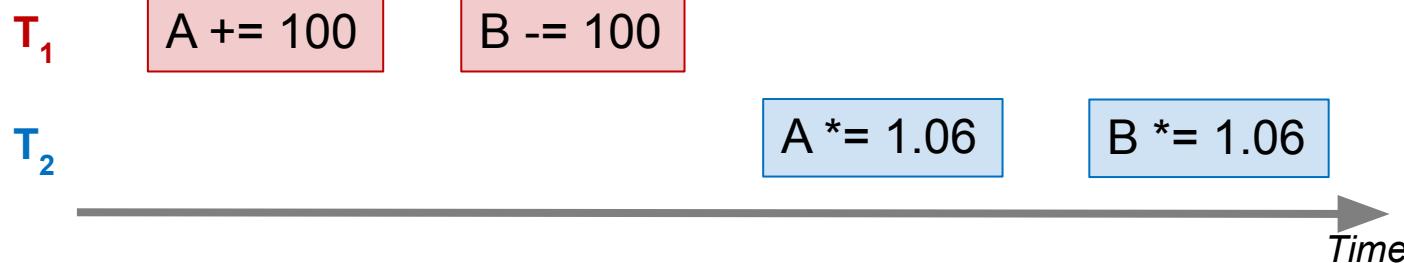
SET Amt = Amt * 1.06

COMMIT

T2 πιστώνει και τους δύο λογαριασμούς με 6% επιτόκιο

Παράδειγμα- θεωρείστε δύο συναλλαγές

Μπορούμε να δούμε τις συναλλαγές σε μια χρονική σειρά-
σειριακή εκτέλεση:

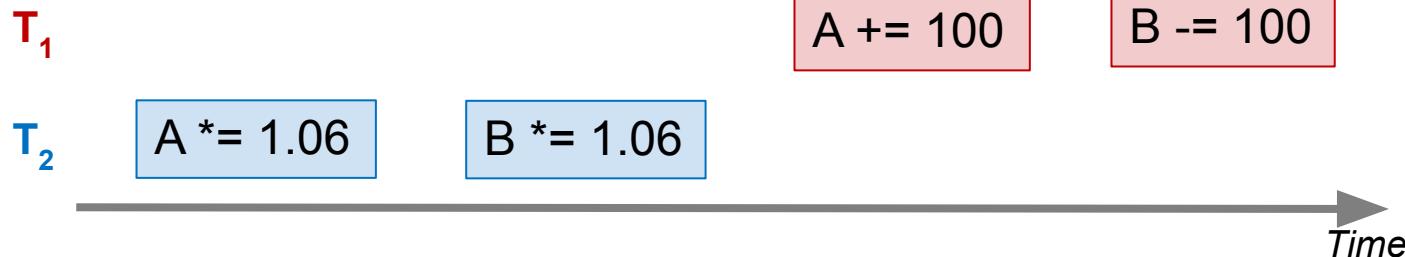


T_1 μεταφέρει \$100 από
τον λογαριασμό του B
στον λογαριασμό του A

T_2 πιστώνει και τους δύο
λογαριασμούς με 6%
επιτόκιο

Παράδειγμα- θεωρείστε δύο συναλλαγές

Οι συναλλαγές θα μπορούσαν να συμβούν και με την ανάποδη σειρά... το ΣΔΒΔ το επιτρέπει!

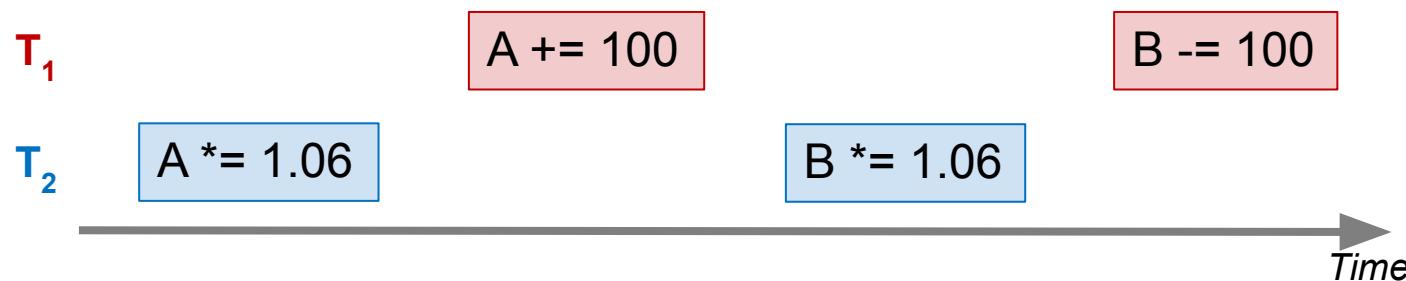


T_2 πιστώνει και τους δυο λογαριασμούς με 6% επιτόκιο

T_1 μεταφέρει \$100 από τον λογαριασμό του B στον λογαριασμό του A

Παράδειγμα- θεωρείστε δύο συναλλαγές

Το ΣΔΒΔ μπορεί να συνδυάσει τις συναλλαγές

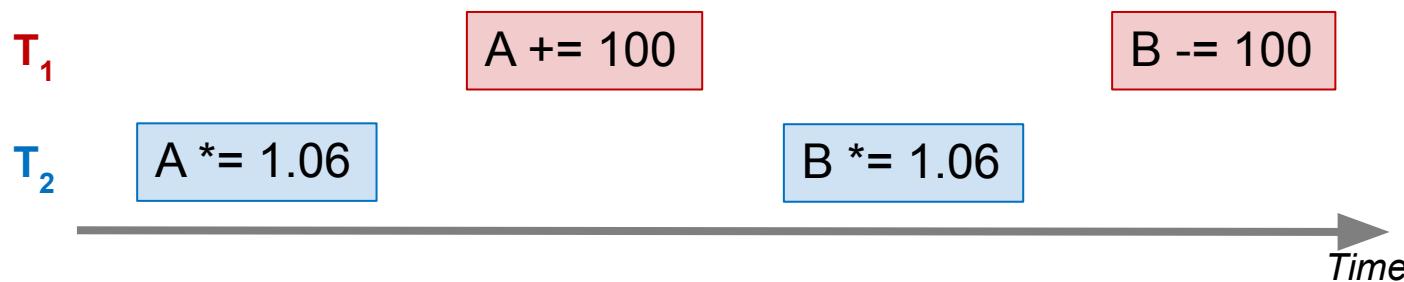


Η T_2 πιστώνει τον λογαριασμό του A με 6% επιτόκιο, μετά η T_1 μεταφέρει \$100 στον λογαριασμό του A ...

Η T_2 πιστώνει τον λογαριασμό του B με 6% επιτόκιο, μετά η T_1 μεταφέρει \$100 από τον λογαριασμό του B ...

Παράδειγμα- θεωρείστε δύο συναλλαγές

Το ΣΔΒΔ μπορεί να **συνδυάσει** τις συναλλαγές



Τι πάει λάθος εδώ?

Το επιτόκιο έχει μπει 2 φορές
στο “ίδιο ποσό”



Γιατί να συνδυάζουμε Συναλλαγές;

- ▶ Τα χρονοδιαγράμματα με ταυτόχρονη εκτέλεση συναλλαγών μπορεί να οδηγήσουν σε ανώμαλα αποτελέσματα... γιατί να ακολουθούμε αυτή την πρακτική;
- ▶ Πολλοί σημαντικοί λόγοι:
 - ▷ **Οι ατομικές συναλλαγές μπορεί να είναι αργές-δεν θέλουμε να εμποδίζουμε τους υπόλοιπους χρήστες!**
 - ▷ **Η πρόσβαση στον δίσκο μπορεί να είναι αργή-ας αφήσουμε μερικές συναλλαγές να χρησιμοποιούν τους επεξεργαστές όσο κάποιες άλλες έχουν πρόσβαση στο δίσκο!**

Όλα αφορούν μεγάλες διαφορές στην **επίδοση (performance)**

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ Ανωμαλίες σε Ταυτόχρονη Εκτέλεση
- ▶ Διενέξεις
 - ▷ Σειριοποιησιμότητα διενέξεων
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Χρονοδιαγράμματα



Παρεμβολές & Απομόνωση

- ▶ Το ΣΔΒΔ έχει την ελευθερία να παρεμβάλει συναλλαγές
- ▶ Παρόλα αυτά, πρέπει να φτιάξει ένα **χρονοδιάγραμμα** έτσι ώστε η **Απομόνωση** και η **Συνέπεια** να διατηρούνται

Πρέπει να είναι **σαν να εκτελούνται** οι συναλλαγές σειριακά!

Μεγάλη Δύναμη
↔
Μεγάλη Ευθύνη

ACID

Τα ΣΔΒΔ πρέπει να επιλέξουν ένα χρονοδιάγραμμα όπου να διατηρείται η **Απομόνωση** και η **Συνέπεια**

Παραδείγματα χρονοδιαγραμμάτων

Αρχικό υπόλοιπο	A	B
	\$50	\$200

Σειριακή εκτέλεση T_1, T_2 :

T_1 A += 100 B -= 100

T_2 A *= 1.06 B *= 1.06

A B
\$159 \$106

Ταυτόχρονη εκτέλεση A:

T_1 A += 100 B -= 100

T_2 A *= 1.06 B *= 1.06

A B
\$159 \$106

Ίδιο
αποτέλεσμα!

Παραδείγματα χρονοδιαγραμμάτων

Αρχικό υπόλοιπο	A	B
	\$50	\$200

Σειριακή εκτέλεση T_1, T_2 :

T_1 A += 100 B -= 100

T_2 A *= 1.06 B *= 1.06

A B
\$159 \$106

Ταυτόχρονη εκτέλεση B:

T_1 A += 100 B -= 100

T_2 A *= 1.06 B *= 1.06

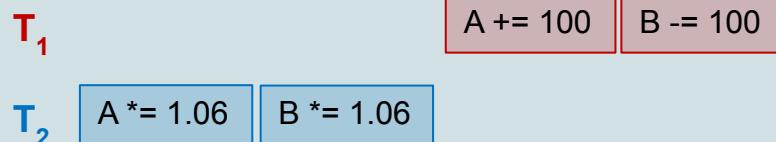
A B
\$159 \$112

Διαφορετικό
αποτέλεσμα από
την σειριακή
 T_1, T_2 !

Παραδείγματα χρονοδιαγραμμάτων

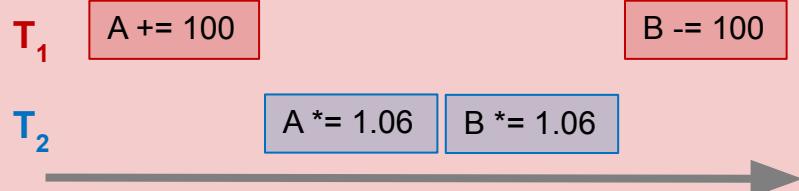
Αρχικό υπόλοιπο	A	B
	\$50	\$200

Σειριακή εκτέλεση T_2, T_1 :



A	B
\$153	\$112

Ταυτόχρονη εκτέλεση B:



A	B
\$159	\$112

Διαφορετικό αποτέλεσμα από την σειριακή T_2, T_1 ΕΠΙΣΗΣ!

Παραδείγματα χρονοδιαγραμμάτων

Ταυτόχρονη εκτέλεση Β:

T_1

$A += 100$

$B -= 100$

T_2

$A *= 1.06$

$B *= 1.06$

- Αυτό το χρονοδιάγραμμα έχει διαφορετικό αποτέλεσμα από οποιαδήποτε σειριακή σειρά εκτέλεσης!
- Λέμε ότι δεν είναι σειριοποιήσιμο



Ορισμοί χρονοδιαγραμάτων

- ▶ **Σειριακό χρονοδιάγραμμα** είναι αυτό που δεν παρεμβάλλει εντολές από διαφορετικές συναλλαγές.
- ▶ Τα Α και Β είναι **ισοδύναμα χρονοδιαγράμματα** εάν, για οποιοδήποτε κατάσταση της βάσης, η επίδραση της εκτέλεσης του Α στην ΒΔ **είναι πανομοιότυπη** με την επίδραση της εκτέλεσης του Β.
- ▶ **Σειριοποιήσιμο χρονοδιάγραμμα** είναι το χρονοδιάγραμμα που είναι ισοδύναμο με κάποιο σειριακό χρονοδιάγραμμα.

Η λέξη “κάποιο” κάνει αυτόν τον ορισμό ισχυρό & πολύπλοκο!

Παράδειγμα - δύο συναλλαγές T1 και T2

υποβάλλονται στην ΒΔ

T₁

A += 100

B -= 100

Η T1 μεταφέρει \$100 από τον λογαριασμό του B στον λογαριασμό του A

T₂

A *= 1.06

B *= 1.06

Η T2 πιστώνει και τους δύο λογαριασμούς με 6% επιτόκιο

Στόχος για χρονοδιαγράμματα συναλλαγών:

- Ταυτοχρονίζουμε συναλλαγές για καλύτερη επίδοση
- Τα δεδομένα μένουν σε **καλή κατάσταση** μετά τα commits και/ή τις εγκαταλείψεις (ACID)

Παράδειγμα

Χρονοδιαγράμματα Σειριακής Εκτέλεσης

T1	A += 100	B -= 100		
T2			A *= 1.06	B*= 1.06

S₁

T1			A += 100	B -= 100
T2		A *= 1.06	B *= 1.06	

S₂

Σειριακά Χρονοδιαγράμματα

S₁, S₂

Σειριοποιήσιμα Χρονοδιαγράμματα

S₃, S₄

Ισοδύναμα Χρονοδιαγράμματα

$\langle S_1, S_3 \rangle$
 $\langle S_2, S_4 \rangle$

Μη-Σειριοποιήσιμα (Κακά) Χρονοδιαγράμματα

S₅, S₆

Χρονοδιαγράμματα Ταυτόχρονης Εκτέλεσης

T1	A += 100			B -= 100
T2		A *= 1.06		B*= 1.06

S₃

T1		A += 100		B -= 100
T2	A *= 1.06		B *= 1.06	

S₄

T1			A += 100	B -= 100
T2	A *= 1.06			B*= 1.06

S₅

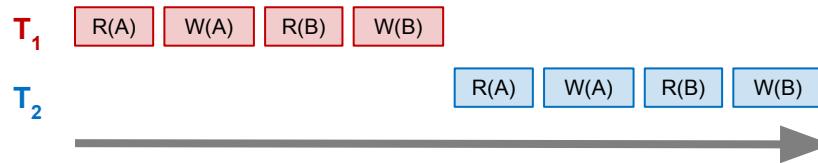
T1	A += 100			B -= 100
T2		A *= 1.06	B *= 1.06	

S₆



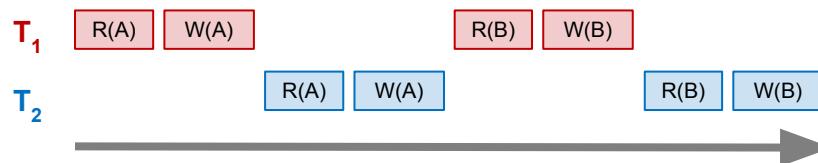
Γενικό μοντέλο ΣΔΒΔ: Ταυτοχρονισμός Παρεμβάλλοντας Συναλλαγές

Χρονοδιάγραμμα σειριακής εκτέλεσης



“Κάθε ενέργεια στις συναλλαγές διαβάζει μια τιμή από την μνήμη και έπειτα γράφει κάποια πίσω σε αυτήν”

Χρονοδιάγραμμα ταυτόχρονης εκτέλεσης



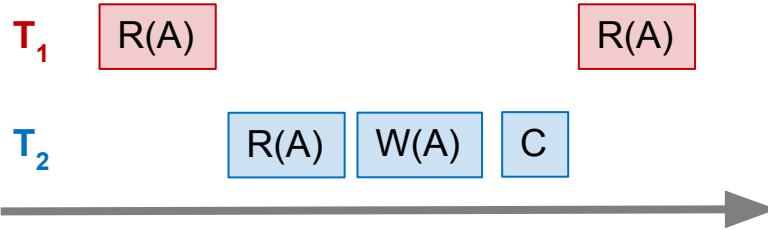
Το να έχουμε συναλλαγές που συμβαίνουν ταυτόχρονα σημαίνει να παρεμβάλλονται οι επιμέρους ενέργειές τους (διάβασμα -R-/γράψιμο-W-)

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ **Ανωμαλίες σε Ταυτόχρονη Εκτέλεση**
- ▶ Διενέξεις
 - ▷ Σειριοποιησιμότητα διενέξεων
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Ανωμαλίες σε Ταυτόχρονη Εκτέλεση

Κλασσικές Ανωμαλίες σε Ταυτόχρονη Εκτέλεση

“Μη-επαναλαμβανόμενο διάβασμα”

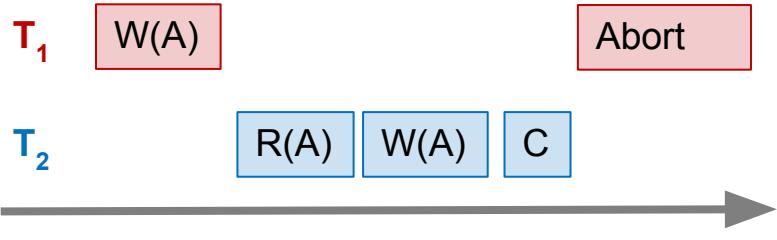


1. Η T_1 διαβάζει μερικά δεδομένα από A
2. Η T_2 γράφει στο A
3. Έπειτα, η T_1 διαβάζει από A ξανά και τώρα παίρνει μια διαφορετική / ασυνεπή τιμή

Παρουσιάζεται με / εξαιτίας μιας **διένεξης διαβάσματος-γραψίματος (RW)**

Κλασσικές Ανωμαλίες σε Ταυτόχρονη Εκτέλεση

“Βρώμικο διάβασμα” / “Διαβάζοντας αδέσμευτα(uncommitted)
δεδομένα”



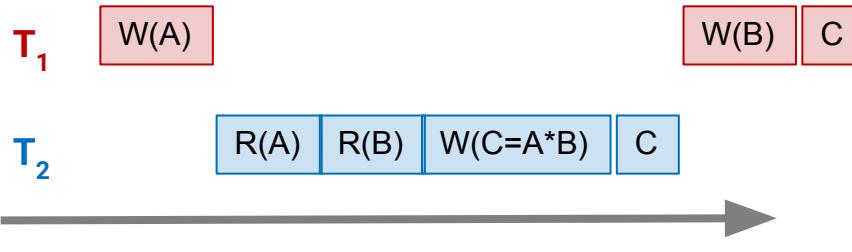
1. Η T_1 γράφει μερικά δεδομένα στο A
2. Η T_2 διαβάζει από A, έπειτα γράφει πίσω στο A & τερματίζει (commits)
3. Η T_1 έπειτα ακυρώνει- τώρα το αποτέλεσμα της T_2 βασίζεται σε μια παρωχημένη / ασυνεπή τιμή

Παρουσιάζεται με / εξαιτίας μιας διένεξης γραψίματος-διαβάσματος (WR)

Κλασσικές Ανωμαλίες σε Ταυτόχρονη Εκτέλεση

“Ασυνεπές διάβασμα” / “Διαβάζοντας αποσπασματικά commits”

Παράδειγμα: $A=1, B=1 \rightarrow A=2, B=2$

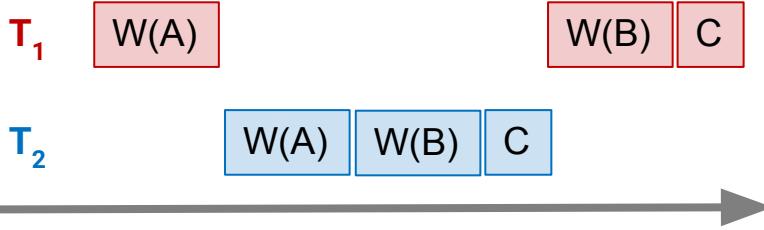


1. Η T_1 γράφει μερικά δεδομένα στο A
2. Η T_2 διαβάζει από A και B , και έπειτα γράφει μερικές τιμές που εξαρτώνται από τον A & τον B
3. Η T_1 έπειτα γράφει στον B - τώρα το αποτέλεσμα της T_2 βασίζεται σε ένα ατελές commit

Ξανά, συμβαίνει εξαιτίας μιας **διένεξης γραφίματος-διαβάσματος (WR)**

Κλασσικές Ανωμαλίες σε Ταυτόχρονη Εκτέλεση

“Μερικώς-χαμένη ενημέρωση”



1. Η T_1 γράφει τυφλά μερικά δεδομένα στον A
2. Η T_2 γράφει τυφλά στον A και στον B
3. Η T_1 έπειτα γράφει τυφλά στον B ; τώρα έχουμε την τιμή του T_2 για τον B και την τιμή του T_1 για τον A - **μη ισοδύναμο με οποιοδήποτε σειριακό χρονοδιάγραμμα!**

Συμβαίνει εξαιτίας μιας διένεξης **εγγραφής-εγγραφής (WW)**

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ Ανωμαλίες σε Ταυτόχρονη Εκτέλεση
- ▶ **Διενέξεις**
 - ▷ Σειριοποιησιμότητα διενέξεων
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Διενέξεις



Τύποι Διενέξεων

Δύο ενέργειες είναι σε **διένεξη** εάν είναι μέρος διαφορετικών συναλλαγών, περιλαμβάνουν την **ίδια μεταβλητή**, και τουλάχιστον μία από αυτές είναι **γράψιμο**.

Έτσι, υπάρχουν τρεις τύποι διενέξεων:

- ▶ Διενέξεις Διάβασμα-Γράψιμο (RW)
- ▶ Διενέξεις Γράψιμο-Διάβασμα (WR)
- ▶ Διενέξεις Γράψιμο-Γράψιμο (WW)

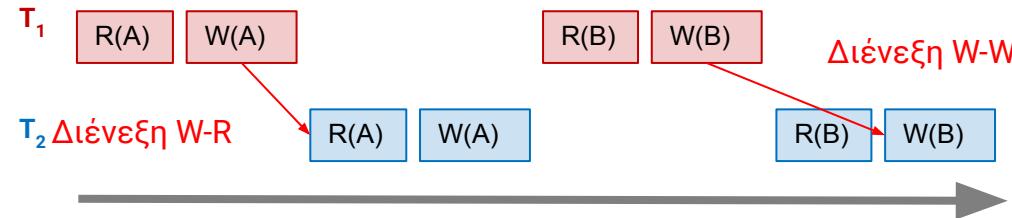
Γιατί να μην υπάρχει Διένεξη “**Ανάγνωση-Ανάγνωση (RR)**”;;

Σημείωση: Οι **διενέξεις** συμβαίνουν συχνά σε πολλές συναλλαγές στον πραγματικό κόσμο.
(π.χ., δύο άνθρωποι προσπαθούν να κλείσουν ένα αεροπορικό εισιτήριο)



Διενέξεις

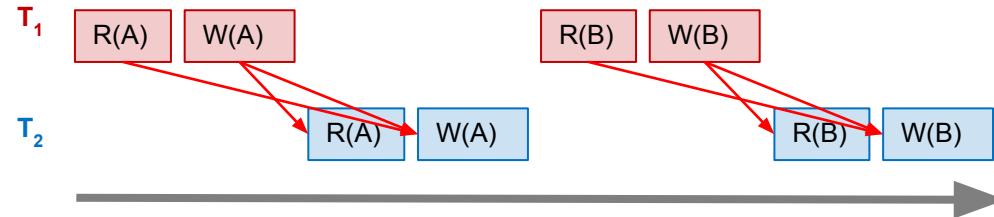
Δύο ενέργειες είναι σε **διένεξη** εάν είναι μέρος διαφορετικών συναλλαγών, περιλαμβάνουν την **ίδια μεταβλητή**, και τουλάχιστον μία από αυτές είναι **γράψιμο**.





Διενέξεις

Δύο ενέργειες είναι σε **διένεξη** εάν είναι μέρος διαφορετικών συναλλαγών, περιλαμβάνουν την **ίδια μεταβλητή**, και τουλάχιστον μία από αυτές είναι **γράψιμο**.



Όλες οι “διενέξεις”!



Σημείωση: Διενέξεις vs. Ανωμαλίες

Οι διενέξεις βρίσκονται σε “καλά” και “κακά” χρονοδιαγράμματα (είναι μια ιδιότητα των συναλλαγών)

Στόχος: Να αποφύγουμε τις Ανωμαλίες όσο ταυτοχρονίζουμε συναλλαγές με διενέξεις!

- Μην δημιουργείτε “κακά” χρονοδιαγράμματα όπου η Απομόνωση και/ή Συνέπεια δεν ισχύουν (δηλ, Ανωμαλίες)

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ **Ανωμαλίες σε Ταυτόχρονη Εκτέλεση**
- ▶ Διενέξεις
 - ▷ **Σειριοποιησιμότητα διενέξεων**
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Σειριοποιησιμότητα διενέξεων



Σειριοποιησιμότητα διενέξεων

2 χρονοδιαγράμματα είναι **ισοδύναμα ως προς τις διενέξεις (conflict equivalent)** αν:

- ▶ Περιλαμβάνουν τις ίδιες ενέργειες των ίδιων Συναλλαγών
- ▶ Κάθε ζεύγος συγκρουόμενων ενεργειών δύο Συναλλαγών έχει τις ενέργειες αυτές με την ίδια αλληλουχία

Το χρονοδιάγραμμα S είναι **σειριοποιήσιμο ως προς τις διενέξεις** αν το S είναι ισοδύναμο ως προς τις διενέξεις με κάποιο σειριακό χρονοδιάγραμμα.

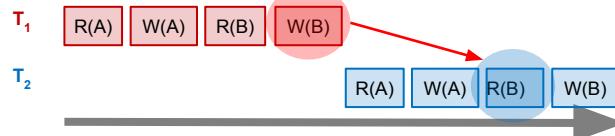
Σειριοποιήσιμο ως προς διενέξεις ⇒ σειριοποιήσιμο

Οπότε αν έχουμε σειριοποιήσιμο ως προς διενέξεις, έχουμε Συνέπεια & Απομόνωση

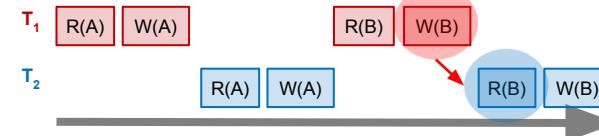
Παράδειγμα “Καλού” vs. “Κακού”

χρονοδιαγράμματος

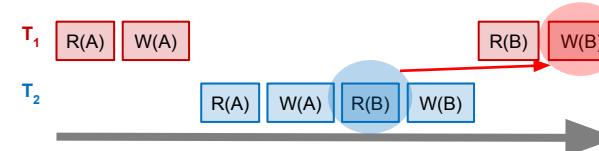
Σειριακό Χρονοδιάγραμμα



Ταυτόχρονο Χρονοδιάγραμμα



Σημειώστε ότι στο “κακό” χρονοδιάγραμμα, η αλληλουχία των συγκρουόμενων ενεργειών είναι διαφορετική από το άνω (ή οποιοδήποτε) σειριακό χρονοδιάγραμμα!



Η σειριοποιησιμότητα ως προς τις διενέξεις μας παρέχει μία λειτουργική αντίληψη των “καλών” vs. “κακών” χρονοδιαγραμμάτων! Τα “κακά” χρονοδιαγράμματα δημιουργούν στα δεδομένα Ανωμαλίες

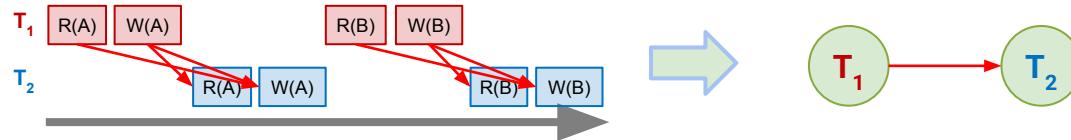
- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ Χρονοδιαγράμματα
- ▶ Ανωμαλίες σε Ταυτόχρονη Εκτέλεση
- ▶ Διενέξεις
 - ▷ Σειριοποιησιμότητα διενέξεων
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Γράφος Διενέξεων



Ο Γράφος Διενέξεων

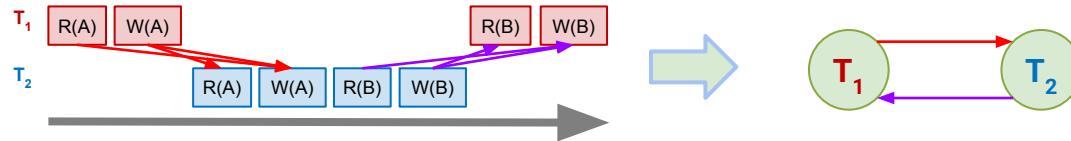
- ▶ Ας κοιτάξουμε τώρα σε διενέξεις **στο επίπεδο των συναλλαγών**
- ▶ Θεωρείστε γράφο όπου **οι κόμβοι είναι συναλλαγές**, και υπάρχει μία ακμή από $T_i \rightarrow T_j$ αν οποιεσδήποτε ενέργειες στο T_i προηγούνται και συγκρούονται με οποιεσδήποτε ενέργειες στο T_j





Ο Γράφος Διενέξεων

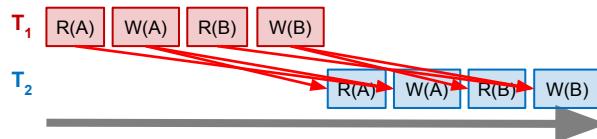
- ▶ Ας κοιτάξουμε τώρα σε διενέξεις **στο επίπεδο των συναλλαγών**
- ▶ Θεωρείστε γράφο όπου **οι κόμβοι είναι συναλλαγές**, και υπάρχει μία ακμή από $T_i \rightarrow T_j$ αν οποιεσδήποτε ενέργειες στο T_i προηγούνται και συγκρούονται με οποιεσδήποτε ενέργειες στο T_j





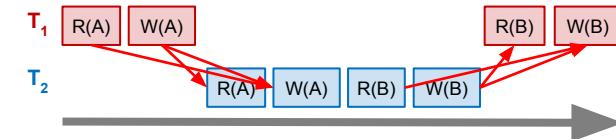
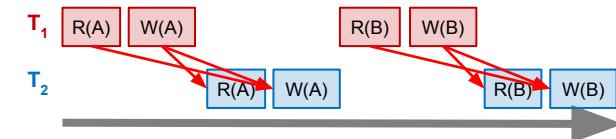
Τι μπορούμε να πούμε για τωνς “καλούς” vs. “κακούς” γράφους διενέξεων;

Σειριακό Χρονοδιάγραμμα:



Λίγο περίπλοκο...

Ταυτόχρονο Χρονοδιάγραμμα:





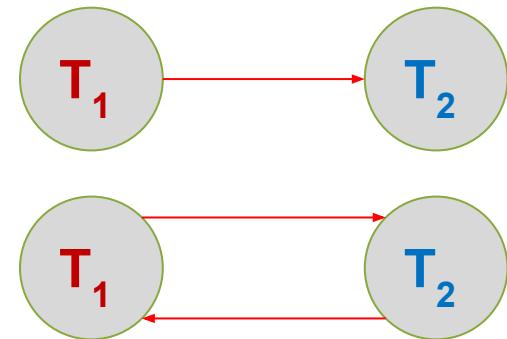
Τι μπορούμε να πουμε για τωνς “καλούς” vs. “κακούς” γράφους διενέξεων;

Σειριακό Χρονοδιάγραμμα:



Απλό!

Ταυτόχρονο Χρονοδιάγραμμα:



Θεώρημα: Ένα χρονοδιάγραμμα είναι **σειριοποιήσιμο** ως προς τις διενέξεις αν και μόνο αν ο γράφος διενέξεών του είναι ακυκλικός.

Παράδειγμα 1

Θεώρησε τις ακόλουθες συναλλαγές:

- T_1 : Μεταφορά 100€ μεταξύ των λογαριασμών A→B
- T_2 : Μεταφορά 100€ μεταξύ των λογαριασμών B→C
- T_3 : Μεταφορά 100€ μεταξύ των λογαριασμών C→A

Κάθε συναλλαγή θα πρέπει να διαβάσει και να γράψει στον λογαριασμό χρέωσης και στον λογαριασμό πίστωσης

“Είναι το ακόλουθο πλάνο σειριοποιήσιμο?”

T_1	R (A)	W (A)					R (B)	W (B)			
T_2			R (B)	W (B)					R (C)	W (C)	
T_3					R (C)	W (C)				R (A)	W (A)

T_1

T_2

T_3

“Φτιάχνω τον γράφο Διενέξεων”

Παράδειγμα 1

Θεώρησε τις ακόλουθες συναλλαγές:

- T_1 : Μεταφορά 100€ μεταξύ των λογαριασμών A→B
- T_2 : Μεταφορά 100€ μεταξύ των λογαριασμών B→C
- T_3 : Μεταφορά 100€ μεταξύ των λογαριασμών C→A

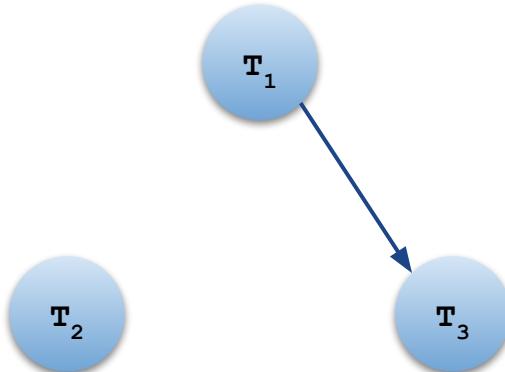
Κάθε συναλλαγή θα πρέπει να διαβάσει και να γράψει στον λογαριασμό χρέωσης και στον λογαριασμό πίστωσης

“Είναι το ακόλουθο πλάνο σειριοποιήσιμο?”

T_1	R (A)	W (A)						
T_2			R (B)	W (B)				
T_3					R (C)	W (C)		

Για την **εγγραφή A** έχουμε ενέργεια RW (και WR, WW) από την συναλλαγή T_1 στην T_3 :

► Προσθέτουμε την ακμή $T_1 \rightarrow T_3$



Παράδειγμα 1

Θεώρησε τις ακόλουθες συναλλαγές:

- T_1 : Μεταφορά 100€ μεταξύ των λογαριασμών A→B
- T_2 : Μεταφορά 100€ μεταξύ των λογαριασμών B→C
- T_3 : Μεταφορά 100€ μεταξύ των λογαριασμών C→A

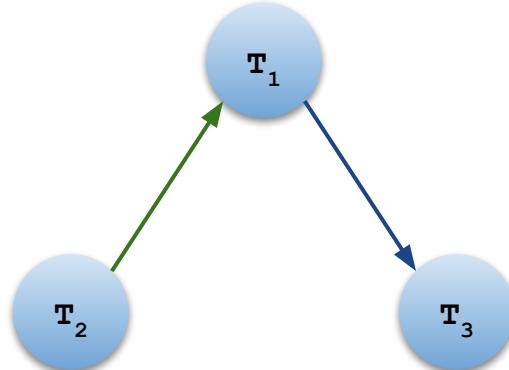
Κάθε συναλλαγή θα πρέπει να διαβάσει και να γράψει στον λογαριασμό χρέωσης και στον λογαριασμό πίστωσης

“Είναι το ακόλουθο πλάνο σειριοποιήσιμο?”

T_1	R (A)	W (A)									
T_2			R (B)	W (B)							
T_3					R (C)	W (C)			R (A)	W (A)	

Για την **εγγραφή B** έχουμε ενέργεια RW (και WR, WW) από την συναλλαγή T_2 στην T_1 :

► Προσθέτουμε την ακμή $T_2 \rightarrow T_1$



Παράδειγμα 1

Θεώρησε τις ακόλουθες συναλλαγές:

- T_1 : Μεταφορά 100€ μεταξύ των λογαριασμών A→B
 - T_2 : Μεταφορά 100€ μεταξύ των λογαριασμών B→C
 - T_3 : Μεταφορά 100€ μεταξύ των λογαριασμών C→A

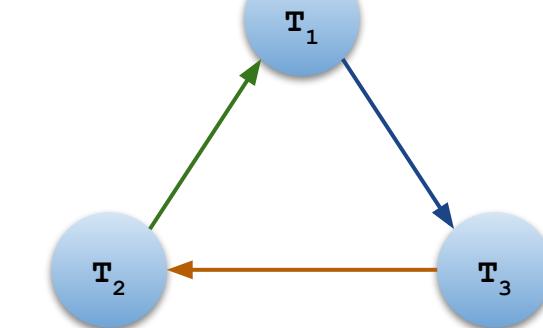
Κάθε συναλλαγή θα πρέπει να διαβάσει και να γράψει στον λογαριασμό χρέωσης και στον λογαριασμό πίστωσης

“Είναι το ακόλουθο πλάνο σειριοποιήσιμο?”

T₁	R (A)	W (A)					R (B)	W (B)		
T₂			R (B)	W (B)					R (C)	W (C)
T₃				R (C)	W (C)				R (A)	W (A)

Για την **εγγραφή C**
έχουμε ενέργεια RW (και
WR, WW) από την
συναλλαγή T_3 στην T_2 :
► Προσθέτουμε την ακμή
 $T_3 \rightarrow T_2$

“Κυκλικός Γράφος Διενέξεων ⇒ Μη σειριοποιέαντο πλάνο”



Παράδειγμα 2

Θεώρησε τις ακόλουθες συναλλαγές:

- T_1 : Μεταφορά 100€ μεταξύ των λογαριασμών A→B
- T_2 : Μεταφορά 100€ μεταξύ των λογαριασμών B→C
- T_3 : Μεταφορά 100€ μεταξύ των λογαριασμών C→A

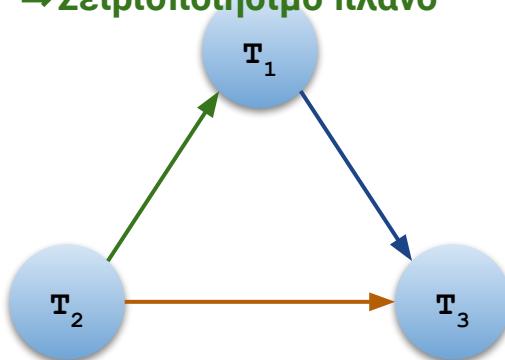
Κάθε συναλλαγή θα πρέπει να διαβάσει και να γράψει στον λογαριασμό χρέωσης και στον λογαριασμό πίστωσης

“Είναι το ακόλουθο πλάνο σειριοποιήσιμο?”

T_1	R (A) W (A)				
T_2		R (B) W (B)	R (C) W (C)		
T_3				R (C) W (C)	R (A) W (A)

“Φτιάχνω τον γράφο Διενέξεων”

**“Μη Κυκλικός Γράφος Διενέξεων
⇒Σειριοποιήσιμο πλάνο”**



- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ **Ανωμαλίες σε Ταυτόχρονη Εκτέλεση**
- ▶ Διενέξεις
 - ▷ **Σειριοποιησιμότητα διενέξεων**
 - ▷ Γράφος Διενέξεων
 - **Dag & Τοπολογικές Ταξινομήσεις**
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - Ανίχνευση Αδιεξόδων

Dag & Τοπολογικές Ταξινομήσεις



DAGs & Τοπολογικές Ταξινομήσεις

- ▶ Μία **τοπολογική ταξινόμηση** ενός κατευθυνόμενου γράφου είναι μία γραμμική διάταξη των κορυφών του που σέβεται όλες τις κατευθυνόμενες ακμές .
Αν υπάρχει η ακμή **a→b**, στον γράφο, στην ταξινόμηση το **a θα εμφανίζεται πριν το b**
- ▶ Ένας κατευθυνόμενος **ακυκλικός** γράφος (Directed Acyclic Graph -DAG-) έχει πάντα μία ή περισσότερες **τοπολογικές ταξινομήσεις**.
Και υπάρχει **μία τοπολογική ταξινόμηση** αν και μόνο αν δεν **υπάρχουν κατευθυνόμενοι κύκλοι**



Σύνδεση με σειριοποίηση ως προς τις διενέξεις

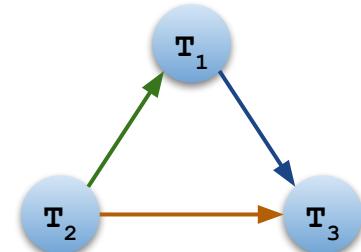
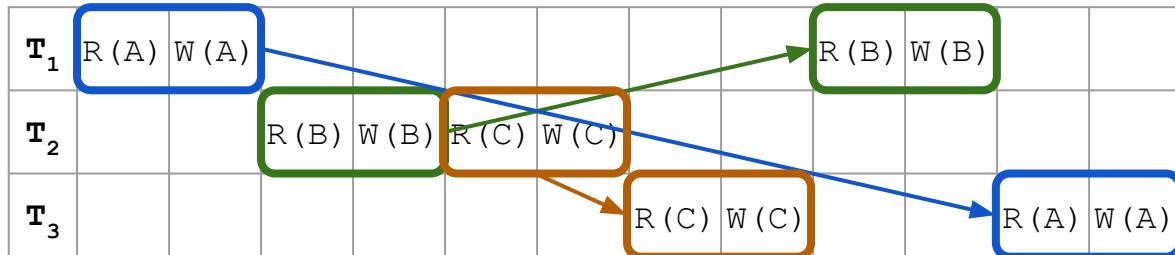
- ▶ Στον γράφο διενέξεων, μια τοπολογική ταξινόμηση των κόμβων αντιστοιχεί σε μια **σειριακή εκτέλεση των συναλλαγών (TXNs)**
- ▶ Για αυτό σε ένα ακυκλικό γράφο διενέξεων→ σειριοποίηση διενέξεων!

Θεώρημα: το χρονοδιάγραμμα είναι **σειριοποιήσιμο** ως προς διενέξεις εάν και μόνο αν ο γράφος διενέξεων είναι ακυκλικός.

Παράδειγμα

“Ποιό είναι το ισοδύναμο σειριακό πλάνο εκτέλεσης?”

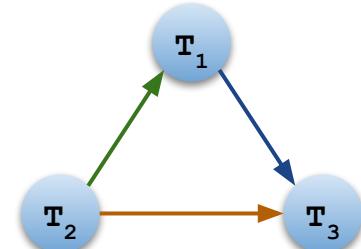
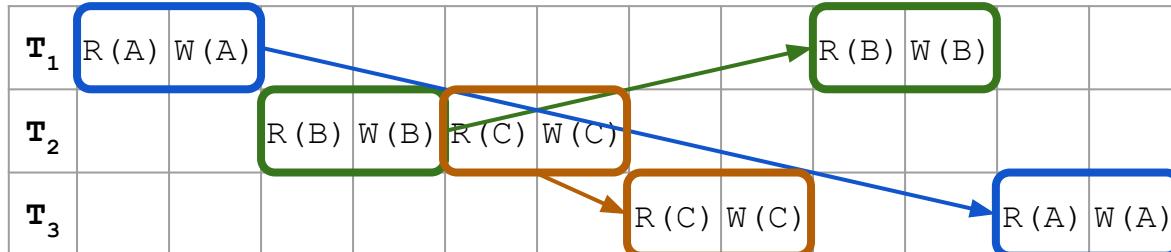
“Μη Κυκλικός Γράφος Διενέξεων”



Παράδειγμα

“Ποιό είναι το ισοδύναμο σειριακό πλάνο εκτέλεσης?”

“Μη Κυκλικός Γράφος Διενέξεων”



Από τον γράφο διενέξεων έχουμε:

- T_2 πριν από T_1 και T_3
- T_1 πριν το T_3

T_1						R (A)	W (A)	R (B)	W (B)				
T_2		R (B)	W (B)	R (C)	W (C)								
T_3										R (C)	W (C)	R (A)	W (A)

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ **Ανωμαλίες σε Ταυτόχρονη Εκτέλεση**
- ▶ Διενέξεις
 - ▷ **Σειριοποιησιμότητα διενέξεων**
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
- ▶ **Αυστηρό Κλείδωμα δύο Φάσεων**
 - Ανίχνευση Αδιεξόδων

Αυστηρό Κλείδωμα δύο Φάσεων



Αυστηρό κλείδωμα δύο-Φάσεων (2-Phase Locking -2PL-)

- ▶ Θεωρείστε Αυστηρό κλείδωμα δύο φάσεων
 - σαν ένα τρόπο να αντιμετωπίσουμε τον ταυτοχρονισμό
 - ▷ Εξασφαλίζει σειριοποιησιμότητα διενέξεων
 - ▷ (Εάν ολοκληρωθεί- Θα δούμε τη συνέχεια...)
- ▶ Επίσης (εννοιολογικά) άμεση εφαρμογή, ξεκάθαρη για το χρήστη!



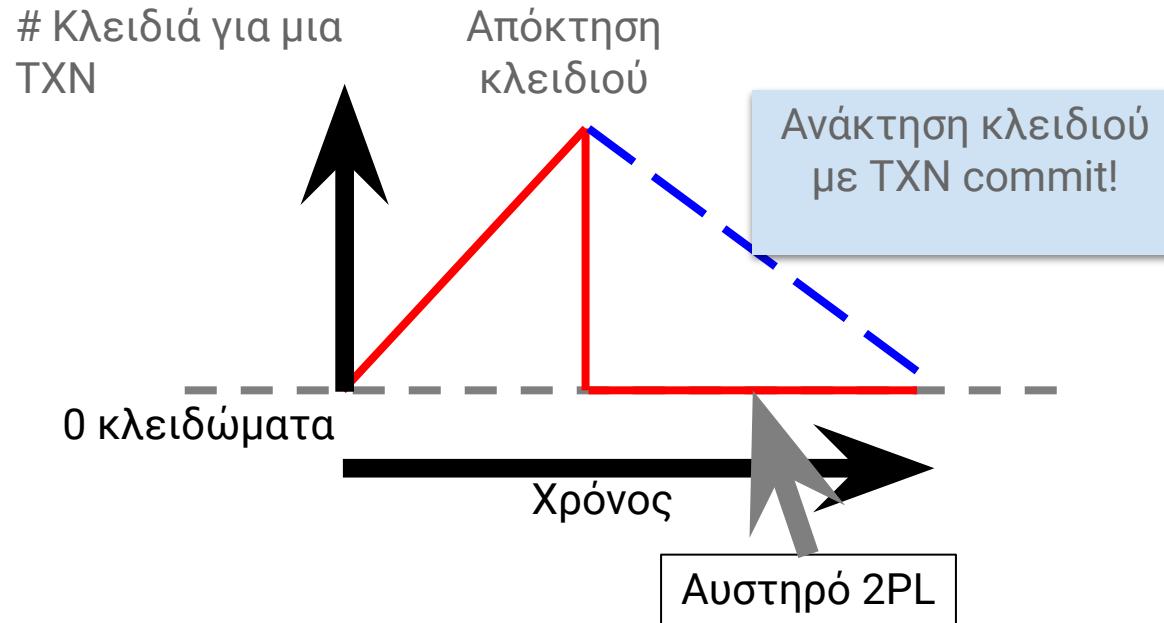
Αυστηρό κλείδωμα δύο-Φάσεων (2PL) Πρωτόκολλο

Οι συναλλαγές (TXNs) αποκτούν:

- ▶ 'Ένα **X (αποκλειστικό) κλείδωμα** στο αντικείμενο πριν την εγγραφή.
 - ▷ Εάν η συναλλαγή είναι ενεργή, καμία άλλη συναλλαγή δεν μπορεί να πάρει κλείδωμα (S ή X) στο ίδιο αντικείμενο.
- ▶ 'Ένα **S (κοινόχρηστο) κλείδωμα** στο αντικείμενο
 - ▷ Εάν η συναλλαγή είναι ενεργή, καμία άλλη συναλλαγή δεν μπορεί να πάρει **X κλείδωμα** σε αυτό το αντικείμενο
- ▶ 'Όλα τα κλειδώματα από μια συναλλαγή ελευθερώνονται όταν αυτή ολοκληρώνεται.

Σημείωση: Οι όροι- "**αποκλειστικό**", "**κοινόχρηστο**"- σημαίνουν ακριβώς αυτό που λένε!

Εικόνα ενός κλειδώματος 2-Φάσεων (2PL)





Αυστηρό Κλείδωμα Δύο-Φάσεων (2PL)

Θεώρημα: Το Αυστηρό κλείδωμα 2-Φάσεων (2PL) επιτρέπει μόνο χρονοδιαγράμματα στα οποία ο γράφος εξαρτήσεων είναι ακυκλικός

Απόδειξη διαισθητικά: Εάν υπάρχει μια ακμή $T_i \rightarrow T_j$ (δηλ οι T_i και T_j συγκρούονται) \Rightarrow η T_j πρέπει να περιμένει μέχρι η T_i να τελειώσει – ώστε να μην υπάρχει ακμή $T_j \rightarrow T_i$.

Για αυτό το αυστηρό κλείδωμα 2-φάσεων επιτρέπει μόνο σειριοποιήσιμα ως προς τις διενέξεις χρονοδιαγράμματα \Rightarrow σειριοποιήσιμα χρονοδιαγράμματα



Αυστηρό Κλείδωμα Δύο-Φάσεων (2PL)

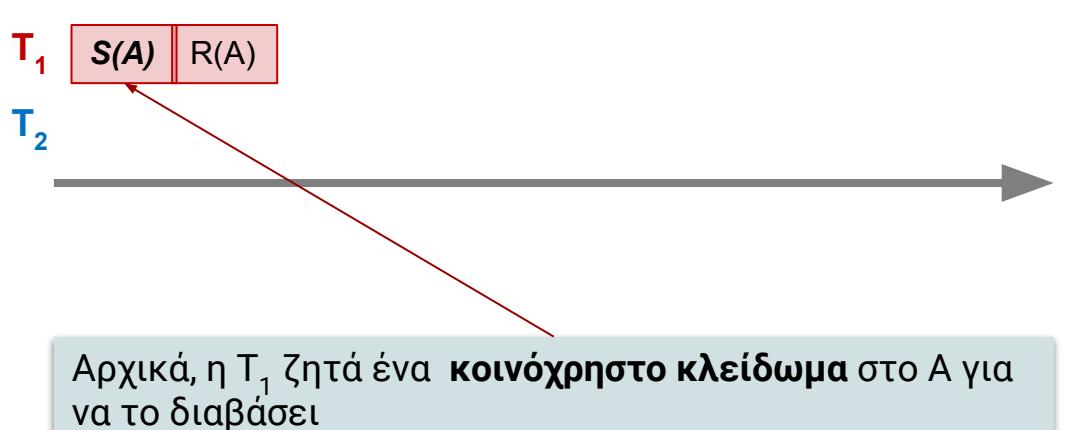
- ▶ Εάν ένα χρονοδιάγραμμα ακολουθεί το 2PL, είναι σειριοποιήσιμο ως προς διενέξεις...
 - ▷ ...και άρα σειριοποιήσιμο
 - ▷ ...και άρα έχουμε Απομόνωση & Συνέπεια!
- ▶ Δημοφιλής εφαρμογή
 - ▷ Απλά, παράγουμε υποσύνολο *όλων* των σειριοποιήσιμων κατά διενέξεις χρονοδιαγραμμάτων
 - ▷ 'Ενα ουσιαστικό πρόβλημα... (συνέχεια)

- ▶ Ταυτοχρονισμός απομόνωση & συνέπεια
 - ▷ **Χρονοδιαγράμματα**
- ▶ **Ανωμαλίες σε Ταυτόχρονη Εκτέλεση**
- ▶ Διενέξεις
 - ▷ **Σειριοποιησιμότητα διενέξεων**
 - ▷ Γράφος Διενέξεων
 - Dag & Τοπολογικές Ταξινομήσεις
 - ▷ Αυστηρό Κλείδωμα δύο Φάσεων
 - **Ανίχνευση Αδιεξόδων**

Ανίχνευση Αδιεξόδων

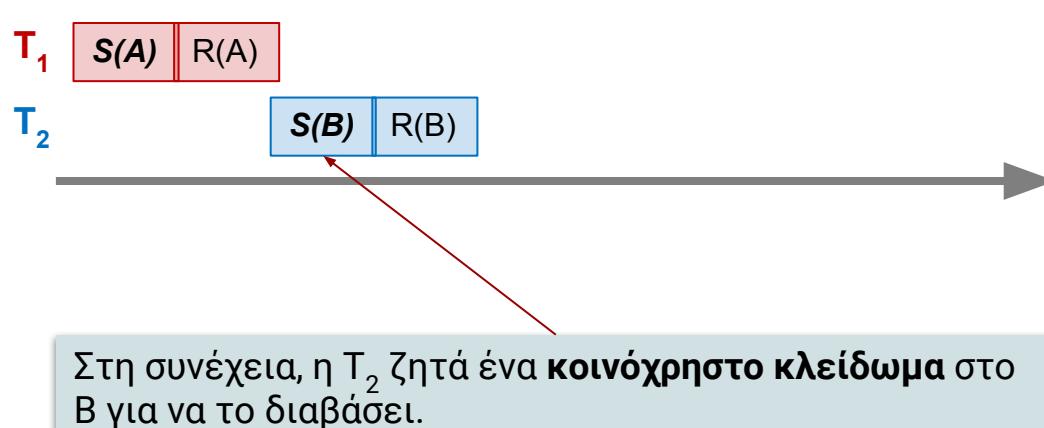
Παράδειγμα: Ανίχνευση Καταστάσεων Αδιεξόδων (Deadlocks)

Γράφος αναμονής
Ποιος-περιμένει-ποιον:



Παράδειγμα: Ανίχνευση Καταστάσεων Αδιεξόδων (Deadlocks)

Γράφος αναμονής
Ποιος-περιμένει-ποιον:



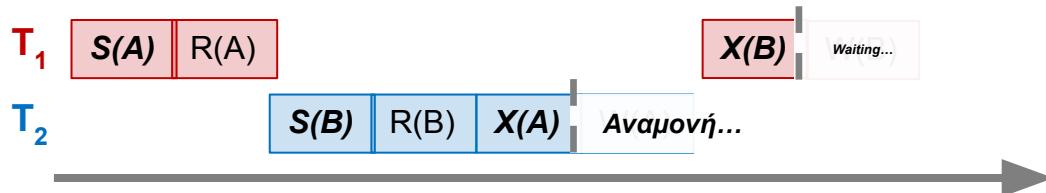
Παράδειγμα: Ανίχνευση Καταστάσεων Αδιεξόδων (Deadlocks)

Γράφος αναμονής
Ποιος-περιμένει-ποιον:

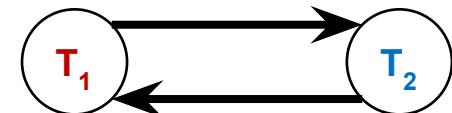


Η T_2 μετά ζητά ένα αποκλειστικό κλείδωμα στο A για να γράψει σε αυτό - τώρα η T_2 περιμένει την T_1 ...

Παράδειγμα: Ανίχνευση Καταστάσεων Αδιεξόδων (Deadlocks)



Γράφος αναμονής
Ποιος-περιμένει-ποιον:



Κύκλος = ΑΔΙΕΞΟΔΟ
(DEADLOCK)

Τελικά, η T_1 ζητά ένα μοναδικό κλείδωμα στο B για να γράψει σε αυτό- **τώρα η T_1 περιμένει την T_2 ...**
ΑΔΙΕΞΟΔΟ!



Αδιέξοδα

Κατάσταση Αδιεξόδου (Deadlock): Κύκλος συναλλαγών όπου τα κλειδώματα περιμένουν να ελευθερώσουν κάποιο αντικείμενο το ένα το άλλο...

Δύο είναι οι τρόποι να διαχειριστούμε τα αδιέξοδα:

1. Αποφυγή Αδιεξόδων
2. Ανίχνευση Αδιεξόδων



Ανίχνευση Αδιεξόδων

Φτιάχνουμε γράφο αναμονής ‘Ποιος-περιμένει-ποιον’:

- ▶ Οι κόμβοι είναι συναλλαγές
- ▶ Υπάρχει μια ακμή $T_i \rightarrow T_j$ εάν ο T_i περιμένει τον T_j να ελευθερώσει ένα κλείδωμα

Περιοδικά ελέγχουμε για (**και σπάμε**) κύκλους στον γράφο αναμονής



Σύνοψη

- ▶ Ο ταυτοχρονισμός επιτυγχάνεται με **εναλλαγή συναλλαγών** με τρόπο τέτοιο ώστε να διατηρούνται η **Απομόνωση & η Συνέπεια**

Ορίσαμε την έννοια της **σειριοποιησιμότητας** η οποία προσδίδει “καλό” τρόπο σχεδιασμού ταυτόχρονων χρονοδιαγραμμάτων

- ▶ Ορίσαμε τη **σειριοποιησιμότητα** ως **προς τις διενέξεις**
- ▶ Το **Κλείδωμα** επιτρέπει μόνο σειριοποιήσιμα ως προς τις διενέξεις χρονοδιαγράμματα

Αν το χρονοδιάγραμμα ολοκληρωθεί... (μπορεί να οδηγήσει σε αδιέξοδο!)

Συναλλαγές

Σύνοψη

Πιατί χρησιμοποιούμε Συναλλαγές;

Καλό προγραμματιστικό μοντέλο -- ACID -- για παράλληλες εφαρμογές σε κοινόχρηστα δεδομένα

- ▶ Atomicity Consistency Isolation Durability
- ▶ Ατομικότητα Συνέπεια Απομόνωση Μονιμότητα

Σχεδιαστικές επιλογές

- ▶ Πώς ενημερώνουμε τα Logs (π.χ., WAL logs)
- ▶ Πώς διαχειρίζόμαστε πολλές συναλλαγές; Σειριακά; Παράλληλα, παρεμβαλλόμενα και σειριοποιήσιμα;



Επιπλέον Υλικό

Κεφάλαια 16, 17, 18 από
Συστήματα Διαχείρισης Βάσεων Δεδομένων
Ramakrishnan, Gehrke/ Δημήτριος Δέρβος

Ευχαριστώ!