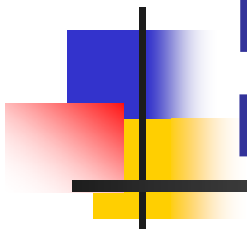
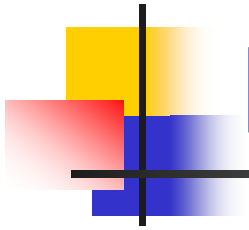


ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΤΜΗΥΠ
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ

Β. Μεγαλοοικονόμου

Επεξεργασία Δοσοληψιών
(Transaction Processing)

(παρουσίαση βασισμένη εν μέρη σε σημειώσεις των Silberchatz, Korth και Sudarshan και του C. Faloutsos)



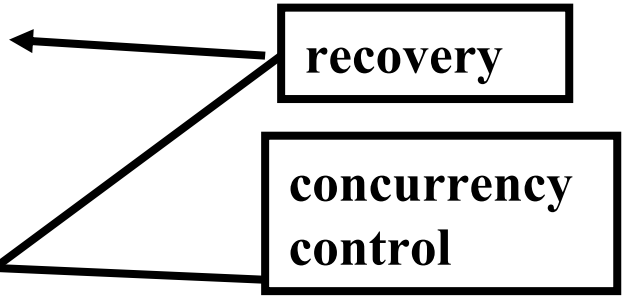
Γενική Επισκόπηση

- Σχεσιακό Μοντέλο - SQL
- Συναρτησιακές Εξαρτήσεις & Κανονικοποίηση
- Φυσικός σχεδιασμός & Δεικτοδότηση
- Βελτιστοποίηση ερωτήσεων
- **Επεξεργασία δοσοληψιών**
  - Έλεγχος συνδρομικότητας (concurrency control)
  - Ανάκαμψη (recovery)

Ορισμός & Επιθυμητές Ιδιότητες των Δοσοληψιών

- Δοσοληψία: Μια λογική μονάδα εργασίας της ΒΔ που περιλαμβάνει μία ή περισσότερες πράξεις προσπέλασης στη ΒΔ, π.χ., μετακίνησε 10€ από το απόθεμα στην πληρωμή

- **Atomicity (Ατομικότητα)**
  - είτε όλες οι πράξεις είτε καμία
- **Consistency (Συνέπεια)**
  - διατήρηση συνέπειας της ΒΔ
- **Isolation (Απομόνωση)**
  - δεν εμπλέκεται με άλλη δοσοληψία που εκτελείται ταυτόχρονα
- **Durability (Μονιμότητα ή διάρκεια)**
  - μετά την επικύρωση μιας δοσοληψίας οι αλλαγές δεν είναι δυνατόν να χαθούν



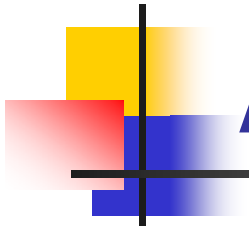
ACID

Δοσοληψία (transaction)

- Το ΣΔΒΔ ενδιαφέρεται για τα δεδομένα που γράφονται ή διαβάζονται από τη ΒΔ
- Πράξεις:
  - ***Ανάγνωση(X)*** – read(x): Διαβάζει ένα στοιχείο X από το δίσκο στην μνήμη
  - ***Εγγραφή(X)*** – write(x): Γράφει το X στο δίσκο (κάποια στιγμή αργότερα).

Διακοπή ρεύματος: Προβλήματα!

Μπορεί να οδηγήσει σε ασυνέπειες ...



Διάρκεια (Durability)

Οι δοσοληψίες θα πρέπει να επιβιώνουν στις αποτυχίες

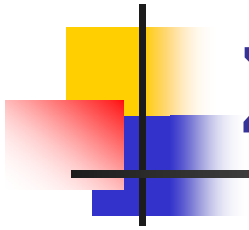
(αφού ολοκληρωθεί επιτυχώς μια δοσοληψία οι αλλαγές παραμένουν στη ΒΔ)

Ατομικότητα (Atomicity)

Απλά:

Λογ. κίνησης = Λογ. κίνησης + 10
Απόθεμα = Απόθεμα - 10

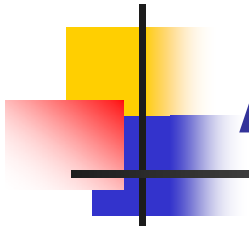
Checking = Checking + 10
Savings = Savings - 10



Συνέπεια (Consistency)

π.χ., το συνολικό άθροισμα των € να είναι ίδιο πριν και μετά

(αλλά όχι απαραίτητα κατά την διάρκεια)



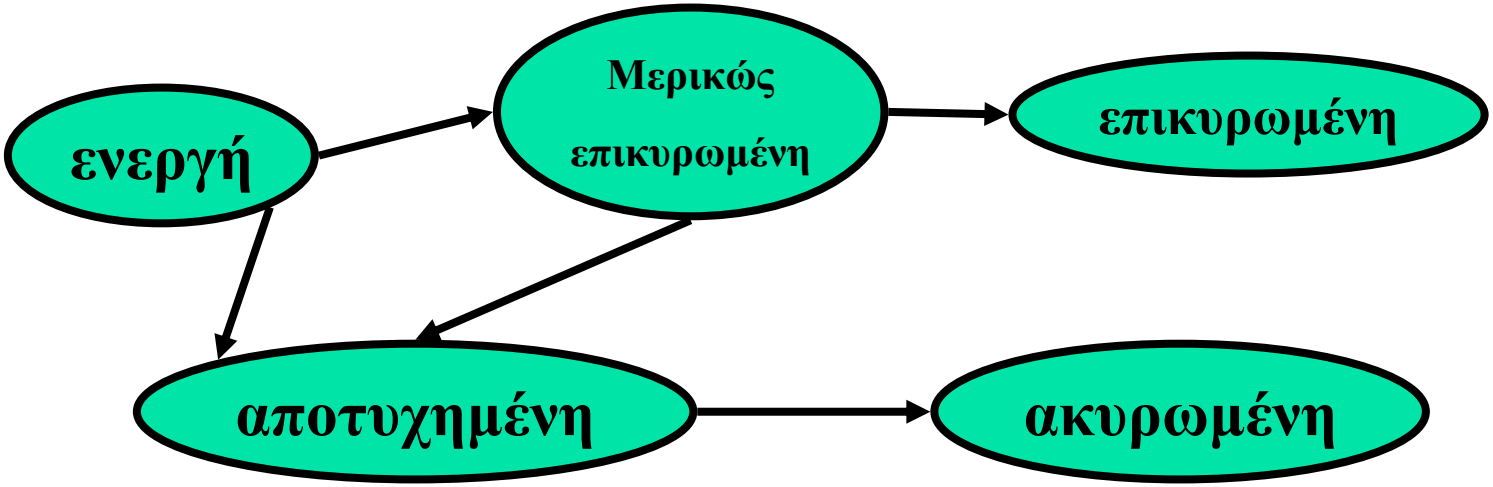
Απομόνωση (Isolation)

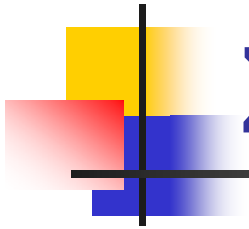
Άλλες δοσοληψίες δεν θα πρέπει να μας επηρεάζουν

Αντιπαράδειγμα: πρόβλημα χαμένης ενημέρωσης (lost update):

read(N)	
	read(N)
N = N - 1	
	N = N - 1
write(N)	
	write(N)

Καταστάσεις Δοσοληψίας



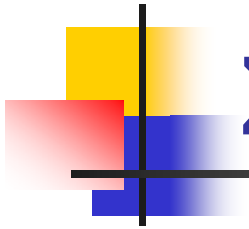


Συνοπτικά

Έλεγχος συνδρομικότητας (concurrency control) (→ απομόνωση (isolation))

- ‘σωστές’ παρεμβολές (correct interleavings)
- πως μπορούν να επιτευχθούν

Αποκατάσταση (recovery) (→ διάρκεια (durability), ατομικότητα (atomicity))



Συνδρομικότητα (Concurrency)

Για ποιο λόγο την χρειαζόμαστε?

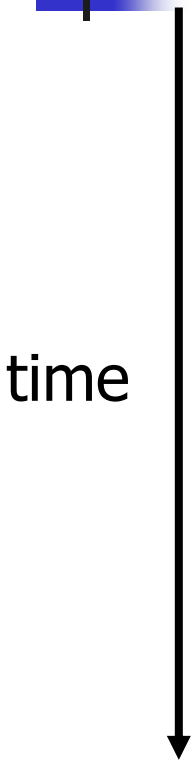
- Αυξημένος αριθμός εκτελέσιμων δοσοληψιών σε συγκεκριμένο χρόνο (increased throughput)
- Αυξημένη χρησιμοποίηση (utilization) (η ΚΜΕ και ο δίσκος ξοδεύουν λιγότερο χρόνο ανενεργά)
- Μειωμένος χρόνος αναμονής (μέσος χρόνος απόκρισης: μέσος χρόνος ολοκλήρωσης μίας δοσοληψίας)

Παράδειγμα παρεμβολής (Interleaving):

T1: μεταφέρει 10€ από τις αποταμιεύσεις (X) στην πληρωμή (Y)

T2: προσθέτει 10% τόκο σε όλα

Εκτέλεση με παρεμβολή (Interleaved execution)



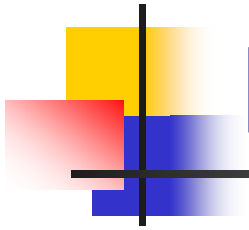
Read(X)

Read(X)

X=X-10
Write(X)
Read(Y)
Y=Y+10
Write(Y)

‘σωστή’;

X = X * 1.1
Write(X)
Read(Y)
Y=Y*1.1
Write(Y)



Πως ορίζεται η ορθότητα?

Πίσω στις βασικές έννοιες…

… ας ξεκινήσουμε από κάτι απόλυτα σωστό:

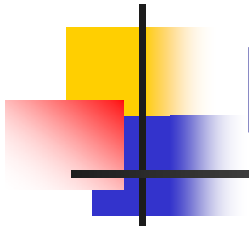
→ Σειριακές εκτελέσεις

Serial execution

T1	T2
Read(X)	
X=X-10	
Write(X)	
Read(Y)	
Y=Y+10	
Write(Y)	
	Read(X)
	X = X * 1.1
	Write(X)
	Read(Y)
	Y=Y*1.1
	Write(Y)

‘σωστό’

Εξ ορισμού



Πως ορίζεται η ορθότητα?

Α: Σειριοποιησιμότητα (serializability):

Ένα χρονοπρόγραμμα (=παρεμβολή) είναι ‘σωστό’ εφόσον είναι σειριοποιήσιμο,

δηλαδή, ισοδύναμο με μία σειριακή παρεμβολή

(ανεξάρτητα από την ακριβή φύση των ενημερώσεων)

Παραδείγματα και αντιπαραδείγματα:

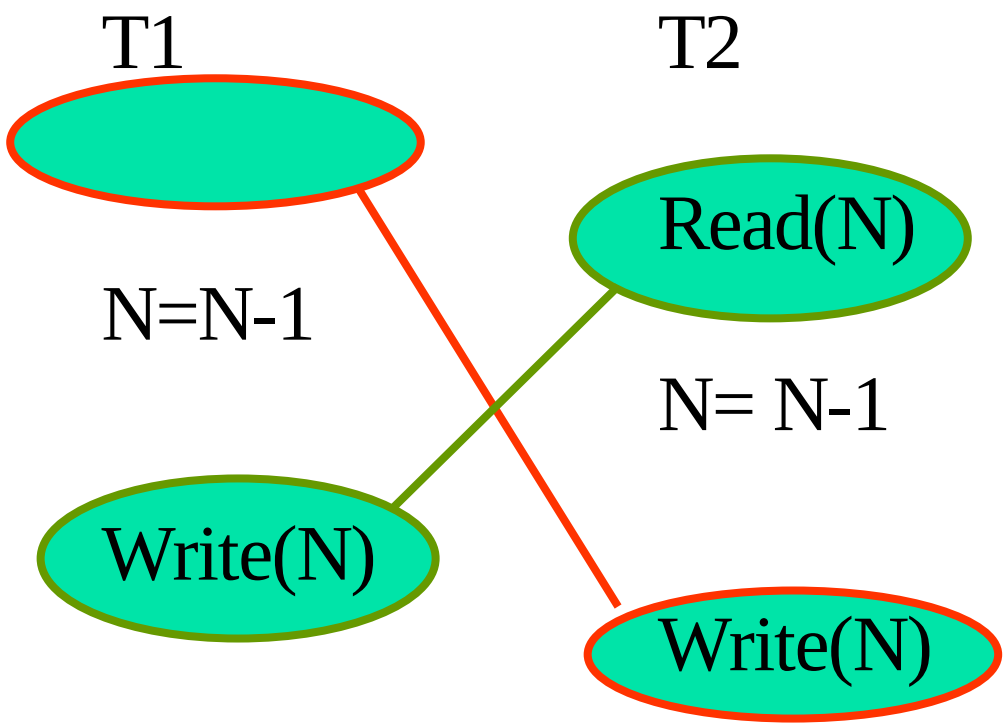
Παράδειγμα: Πρόβλημα 'Χαμένης- ενημέρωσης'

T1	T2
Read(N)	
	Read(N)
N=N-1	
	N= N-1
Write(N)	
	Write(N)

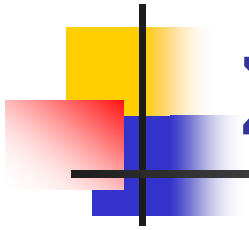
Δεν είναι ισοδύναμη με καμία σειριακή εκτέλεση (γιατί όχι?) →

ΛΑΘΟΣ!

Περισσότερες λεπτομέρειες: ‘Σειριοποιησιμότητα συγκρούσεων’



‘Conflict serializability’



Σειριοποιησιμότητα συγκρούσεων

r/w: π.χ., το αντικείμενο X διαβάζεται από το Ti και εγγράφεται από το Tj

w/w: .....εγγράφεται από Ti και εγγράφεται από το Tj

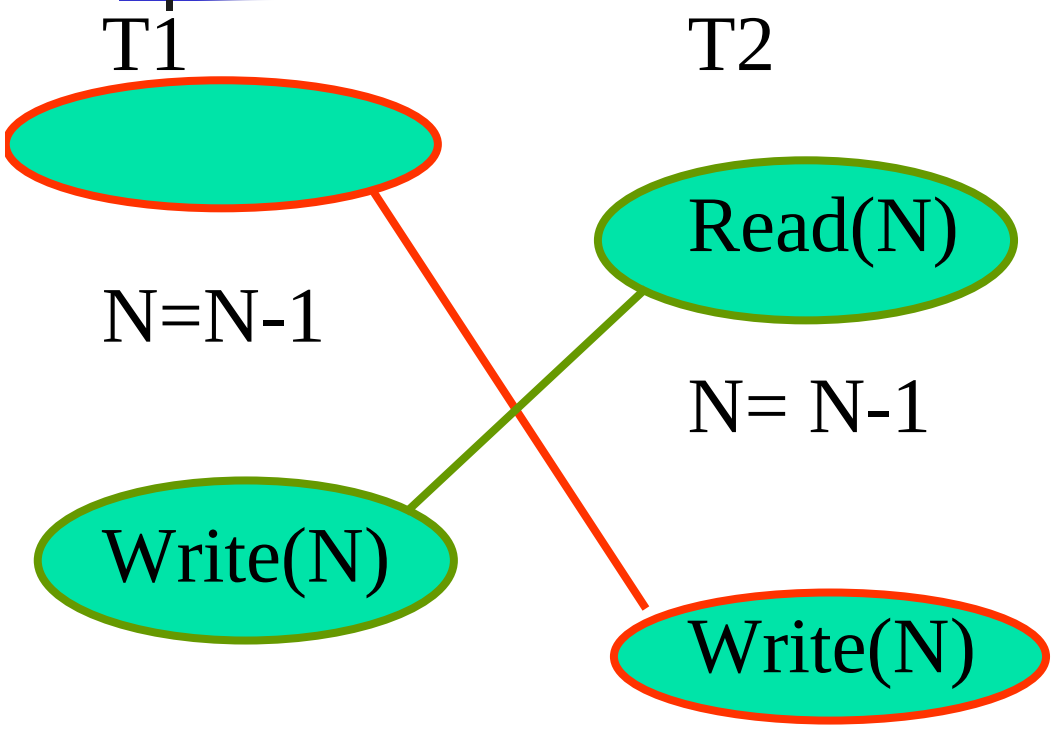
- η σειρά παίζει ρόλο και στις δύο περιπτώσεις ...

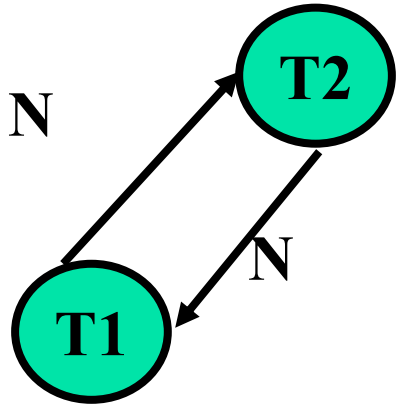
ΓΡΑΦΟΣ ΠΡΟΤΕΡΑΙΟΤΗΤΑΣ (precedence graph):

Κόμβοι: δοσοληψείες (transactions)

Τόξα: r/w, w/r ή w/w συγκρούσεις

Γράφος Προτεραιότητας (Precedence graph)





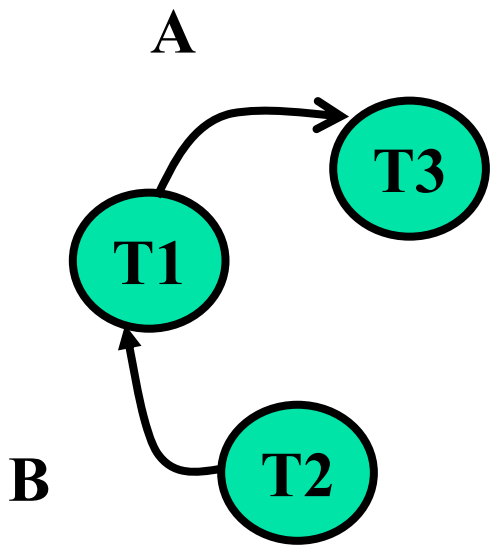
Κύκλος -> μη σειριοποιήσιμος

Παράδειγμα

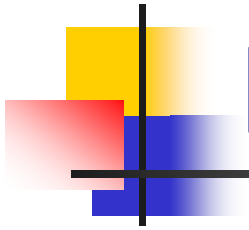
T1	T2	T3
Read(A)		
…		
write(A)		
		Read(A)
		…
		Write(A)
	Read(B)	
	…	
	Write(B)	
Read(B)		
…		
Write(B)		

Παράδειγμα

T1	T2	T3
Read(A)		
…		
write(A)		
		Read(A)
		…
		Write(A)
	Read(B)	
	…	
	Write(B)	
Read(B)		
…		
Write(B)		



σειριακή εκτέλεση;

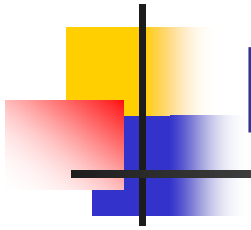


Παράδειγμα

Α: Τ2, Τ1, Τ3

(Σημειώνεται ότι το Τ3 θα πρέπει να πάει μετά το Τ2 αν και ξεκινά πριν από αυτό!)

Ε:Υπάρχει αλγόριθμος για την παραγωγή σειριακής εκτέλεσης από έναν άκυκλο κατευθυνόμενο γράφο (DAG) προτεραιότητας;



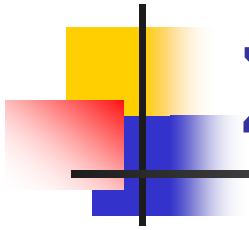
Παράδειγμα

Α: Ναι, χρησιμοποιώντας την Τοπολογική Ταξινόμηση (Topological Sorting)

Μία τοπολογική ταξινόμηση του DAG=(V,E) είναι μία γραμμική διάταξη όλων των κόμβων έτσι ώστε εάν το G περιέχει μία ακμή (u,v), τότε το u εμφανίζεται πριν το v στη διάταξη

…είναι η διάταξη των κόμβων του κατά μήκος μίας οριζόντιας ακμής έτσι ώστε όλες οι κατευθυνόμενες ακμές να έχουν κατεύθυνση από τα αριστερά προς τα δεξιά

…οι τοπολογικά ταξινομημένοι κόμβοι εμφανίζονται σε αντίστροφη σειρά από τον χρόνο ολοκλήρωσής τους σύμφωνα με την αναζήτηση κατά βάθος (DFS)



Σειριοποιησιμότητα

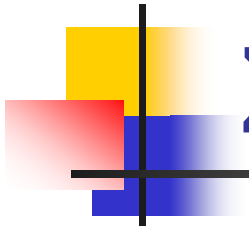
-Αγνοείστε την ‘view serializability’

-Υποθέτουμε ότι δεν υπάρχουν ‘τυφλές εγγραφές’, δηλ., ‘διάβασε πριν την εγγραφή’

Αντιπαράδειγμα: 'Ανάλυση Ασυνέπειας'

T1	T2
Read(A)	
A=A-10	
Write(A)	
	Read(A)
	Sum = A
	Read(B)
	Sum += B
Read(B)	
B=B+10	
Write(B)	

Γράφος Προτεραιότητας?



Συμπεράσματα

- ‘ACID’ ιδιότητες των δοσοληψιών
- Αποκατάσταση για τις ‘A’, ‘D’ (ατομικότητα και μονιμότητα)
- Έλεγχος συνδρομικότητας για την ‘I’ (απομόνωση)
- Σωστό χρονοπρόγραμμα -> σειριοποιήσιμο
- Γράφος προτεραιότητας

  ακυκλικό -> σειριοποιήσιμο