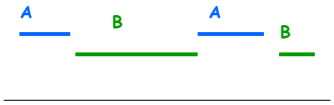
Επεξεργασία Δοσοληψιών

- Η ταυτόχρονη εκτέλεση προγραμμάτων χρηστών είναι απαραίτητη για την καλή απόδοση ενός ΣΔΒΔ
  - Επειδή οι προσπελάσεις στο δίσκο είναι συχνές και σχετικά αργές, είναι σημαντικό να κρατείται η cpu απασχολημένη με πολλά προγράμματα χρηστών

⇒ Πολυχρηστικά ΣΔΒΔ

Διαπλεγμένο μοντέλο ταυτόχρονης εκτέλεσης



Δοσοληψία (transaction)

εκτέλεση ενός προγράμματος που προσπελαύνει ή τροποποιεί το περιεχόμενο της βάσης δεδομένων

το πώς βλέπει το ΣΔΒΔ τα προγράμματα των χρηστών

Δοσοληψία (transaction)

Ένα πρόγραμμα χρήστη μπορεί να εκτελεί πολλές λειτουργίες στα δεδομένα που ανακτά από τη ΒΔ, αλλά το ΣΔΒΔ ενδιαφέρεται μόνο για τα δεδομένα που *διαβάζονται/γράφονται* στη ΒΔ

- Ανάγνωση(X) - R(X)
- Εγγραφή(X) - W(X)

- Οι χρήστες υποβάλουν δοσοληψίες και πρέπει να μπορούν να θεωρούν ότι κάθε δοσοληψία *εκτελείται μόνη της.*
  - Η συνδρομικότητα (concurrency) *επιτυγχάνεται από το ΣΔΒΔ* που διαπλέκει τις πράξεις (αναγνώσεις/εγγραφές) των διαφόρων συναλλαγών

- *Όταν μια δοσοληψία υποβάλλεται στο ΣΔΒΔ το σύστημα πρέπει να εξασφαλίσει ότι (α) είτε όλες οι πράξεις της θα ολοκληρωθούν είτε (β) καμία δε θα εκτελεστεί - δηλαδή δε θα έχει καμία επίδραση στη ΒΔ -- ακόμα και αν συμβούν αποτυχίες*
- *Αυτή είναι μια σημαντική ιδιότητα που πρέπει να εξασφαλίσει το ΣΔΒΔ - Ο χρήστης πρέπει να μπορεί να θεωρεί ότι όλο το πρόγραμμα (πράξεις) εκτελούνται σε ένα βήμα είτε καμία πράξη δεν εκτελείται (ατομικότητα των δοσοληψιών)*

Είδη Αποτυχιών

Δυο κατηγορίες: καταστροφή ή όχι της μόνιμης αποθήκευσης (δίσκου)

Παραδείγματα αποτυχιών ...

Πιο αναλυτικά ...

⌘ Θεωρείστε δύο συναλλαγές (*Xacts*):

T1:	BEGIN R(X), X=X-N, W(X), R(Y), Y=Y+N, W(Y), END
T2:	BEGIN R(X) X=X+M, W(X) END

- Διαισθητικά, η T1 μεταφέρει N κρατήσεις θέσεων από μια πτήση (X) και τις μεταφέρει σε μία άλλη (Y). Η T2 απλώς κρατά M θέσεις στην πρώτη πτήση (τη X)
- Δεν υπάρχει καμία εγγύηση ότι η T1 θα εκτελεστεί πριν την T2 η το ανάποδο, αν και η δύο υποβληθούν ταυτόχρονα. Ωστόσο, το συνολικό αποτέλεσμα πρέπει να είναι ισοδύναμο με τη μία ή την άλλη περίπτωση (δηλαδή, με κάποια *σειριακή εκτέλεση* των δύο δοσοληψιών)

X = 100 κρατήσεις
Y = 90 κρατήσεις
Μεταφορά N = 30 κρατήσεων
Νέα κράτηση M = 5 θέσεις

T1	T2
BEGIN	
R(X) read 100	
X=X-N X= 70	
W(X) **write X = 70**	
R(Y) read 90	
Y=Y+N Y = 120	
W(Y) **write Y = 120**	
END	
	BEGIN
	R(X) read 70
	X=X+M X = 75
	W(X) **write X = 75**
	END

σειριακή εκτέλεση
T1 → T2

T1	T2
	BEGIN
	R(X) read 100
	X=X+M X = 105
	W(X) **write 105**
	END
BEGIN	
R(X) read 105	
X=X-N X = 75	
W(X) **write X = 75**	
R(Y) read 90	
Y=Y+N Y = 120	
W(Y) **write Y = 120**	
END	

σειριακή εκτέλεση
T2 → T1

X = 100 κρατήσεις
Y = 90 κρατήσεις
Μεταφορά N = 30 κρατήσεων
Νέα κράτηση M = 5 θέσεις

T1	T2
BEGIN	
R(X) read 100	
X=X-N X = 70	
	BEGIN
	R(X) read 100
	X=X+M X = 105
W(X) **write X =70**	
R(Y) Y = 90	
	W(X) **write X = 105**
	END
Y=Y+N Y = 110	
W(Y) **write Y = 110**	
END	

Η τιμή του X είναι λανθασμένη

Απώλεια Ενημερώσεων

X = 100 κρατήσεις
Y = 90 κρατήσεις
Μεταφορά N = 30 κρατήσεων
Νέα κράτηση M = 5 θέσεις

T1	T2
BEGIN	
R(X) read 100	
X=X-N X = 70	
W(X) write X = 70	
	BEGIN
	R(X) read 70
	X=X+M X = 75
	W(X) write X =75
	END
Y=Y+N	

Dirty Read

Προσωρινή Ενημέρωση

Τι θα συμβεί αν η T1 αποτύχει - η T2 θα έχει διαβάσει «ανύπαρκτη τιμή»

X = 100 κρατήσεις
Y = 90 κρατήσεις
Μεταφορά N = 30 κρατήσεων
Νέα κράτηση M = 5 θέσεις

Προβλήματα Λόγω Συνδρομικότητας

T1	T2
BEGIN	
R(X) read 100	
	BEGIN
	R(X) read 100
	X=X+M X = 105
	W(X) write X = 105
	END
R(X) ←	**Η τιμή του X που διαβάζει η Τι είναι διαφορετική!!**

Μη Επαναλήψιμη Ανάγνωση

Πράξεις μιας Δοσοληψίας

- **Ας προσπαθήσουμε να ορίσουμε το πρόβλημα στη γενική του μορφή**

Πράξεις μιας Δοσοληψίας

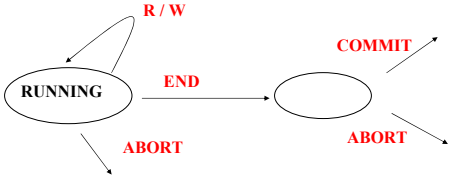
Πράξεις Δοσοληψιών

- **BEGIN**
- **R(X)**
- **W(X)**
- **END**
- **COMMIT (επικύρωση) – επιτυχία – όλες οι τροποποιήσεις επικυρώνονται και δεν μπορούν να αναιρεθούν**
- **ABORT (ακύρωση ή ανάκληση) – αποτυχία – όλες οι τροποποιήσεις πρέπει να αναιρεθούν**

Πράξεις μιας Δοσοληψίας

- Μια δοσοληψία μπορεί να να επικυρωθεί (commit) αφού ολοκληρώσει *όλες* τις πράξεις της ενώ μπορεί να ακυρωθεί (abort) αφού εκτελέσει *κάποιες* από τις πράξεις της
  - Το ΣΔΒΔ *logs* όλες τις πράξεις έτσι ώστε να μπορεί να αναιρέσει (undo) τις πράξεις μιας ακυρωμένης (aborted) δοσοληψίας.

Πράξεις μιας Δοσοληψίας



Συνδρομικότητα σε ΣΔΒΔ

- Κάθε δοσοληψία πρέπει να αφήνει τη ΒΔ σε μια συνεπή κατάσταση αν η ΒΔ ήταν σε συνεπή κατάσταση όταν άρχισε η δοσοληψία
  - Το ΣΔΒΔ επιβάλει κάποιους ΠΑ (Περιορισμούς Ακεραιότητας) με βάση τους ΠΑ που έχουν δηλωθεί στις εντολές CREATE TABLE
  - Πέρα από αυτό, το ΣΔΒΔ δεν καταλαβαίνει τη σημασιολογία των δεδομένων (π.χ., δεν καταλαβαίνει πώς να υπολογίσει το επιτόκιο)

Θέμα: Αποτελέσματα της διαπλεγμένης εκτέλεσης δοσοληψιών (έλεγχος συνδρομικότητας) και των αποτυχιών (ανάκαμψη)

Ιδιότητες Δοσοληψιών

- **A**tomicity (ατομικότητα) - είτε όλες οι πράξεις είτε καμία
- **C**onsistency (συνέπεια) - διατήρηση συνέπειας της ΒΔ
- **I**solation (απομόνωση) - δεν αποκαλύπτει ενδιάμεσα αποτελέσματα
- **D**urability (μονιμότητα ή διάρκεια) - μετά την επικύρωση μιας δοσοληψίας οι αλλαγές δεν είναι δυνατόν να χαθούν

- **A**tomicity (ατομικότητα) → ΤΕΧΝΙΚΕΣ ΑΝΑΚΑΜΨΕΙΣ
- **C**onsistency (συνέπεια) → ΥΠΕΥΘΥΝΟΤΗΤΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΤΗ
- **I**solation (απομόνωση) → ΕΛΕΓΧΟΣ ΣΥΝΔΡΟΜΙΚΟΤΗΤΑΣ
- **D**urability (μονιμότητα ή διάρκεια) → ΤΕΧΝΙΚΕΣ ΑΝΑΚΑΜΨΕΙΣ

- **I**solation (απομόνωση) → ΕΛΕΓΧΟΣ ΣΥΝΔΡΟΜΙΚΟΤΗΤΑΣ

Βαθμός απομόνωσης 0 :: **δεν επικαλύπτει ασταθείς αναγνώσεις (dirty reads) δοσοληψιών με μεγαλύτερο βαθμό**

Βαθμός απομόνωσης 1 :: **δεν παρουσιάζει απώλειες ενημέρωσης**

Βαθμός απομόνωσης 2 :: **ούτε απώλειες ενημέρωσης, ούτε ασταθείς αναγνώσεις**

Βαθμός απομόνωσης 3 :: **επίπεδο 2 + επαναλήψιμες αναγνώσεις**

Τώρα που καταλάβαμε το πρόβλημα::

μπορούμε να βρούμε ένα θεωρητικό μοντέλο που να το εκφράζει;

Ορισμοί

1. δοσοληψία
2. διαπεπλεγμένη εκτέλεση δοσοληψιών (χρονοπρόγραμμα)
3. σωστό -- αποδεκτό χρονοπρόγραμμα

Μια δοσοληψία είναι μια ακολουθία από πράξεις εγγραφής και ανάγνωσης που τελειώνει με μια πράξη επικύρωσης (commit) ή με μια πράξη ακύρωσης (abort)

⌘ Θεωρείστε τις δύο συναλλαγές (*Xacts*) του παραδείγματος:

T1:	BEGIN R(X), X=X-N, W(X), R(Y), Y=Y+N, W(Y), END
T2:	BEGIN R(X) X=X+M, W(X) END

- T1: R(X) W(X) R(Y) W(Y) *C*
- T2: R(X) W(X) *C*

• *Εκφράζει μια συγκεκριμένη εκτέλεση ενός συνόλου δοσοληψιών*

• Οι πράξεις των δοσοληψιών εμφανίζονται στο χρονοπρόγραμμα με τη σειρά που εκτελούνται

Συγκεκριμένα

Ένα χρονοπρόγραμμα (schedule) S των δοσοληψιών T_1, T_2, .., T_n είναι μια διάταξη των πράξεων τους με τον περιορισμό ότι για κάθε δοσοληψία T_i που συμμετέχει στο S οι πράξεις της T_i στο S πρέπει να εμφανίζονται με την ίδια σειρά που εμφανίζονται στην T_i

Θα χρησιμοποιούμε δείκτη στις πράξεις που να δείχνει σε ποια δοσοληψία αναφέρονται

T1	T2
$R_1(X)$	
$W_1(X)$	
$R_1(Y)$	
$W_1(Y)$	
C_1	
	$R_2(X)$
	$W_2(X)$
	C_2

S: $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1 $R_2(X)$ $W_2(X)$ C_2

T1	T2
$R_1(X)$	
	$R_2(X)$
$W_1(X)$	
$R_1(Y)$	
	$W_2(X)$
	C_2
$W_1(Y)$	
C_1	

Τόσα διαφορετικά χρονοπρογράμματα όσες και πιθανές εκτελέσεις

S: $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

T1	T2
$R_1(X)$	
$W_1(X)$	
	$R_2(X)$
	$W_2(X)$
	C_2
A_1	

S: $R_1(X)$ $W_1(X)$ $R_2(X)$ $W_2(X)$ C_2 A_1

Διάταξη πράξεων

S1: $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

S2: $R_2(X)$ $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

Ποια είναι η σχέση των χρονοπρογραμμάτων S1 και S2;

Διαισθητικά «δεν διαφέρουν»

Σύγκρουση πράξεων σε χρονοπρόγραμμα

Δύο πράξεις σε ένα χρονοπρόγραμμα συγκρούονται αν (α) ανήκουν σε διαφορετικές δοσοληψίες, (β) προσπελαύνουν το ίδιο στοιχείο, και (γ) μια από αυτές είναι πράξη εγγραφής (W)

σημασία έχει η σχετική θέση (διάταξη) των πράξεων που συγκρούονται

S1: $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

S2: $R_2(X)$ $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

Τα S1 και S2 ισοδύναμα (διαφέρουν μόνο στη διάταξη πράξεων που δε συγκρούονται) ⇒ μερική διάταξη

Επίσης, ζητάμε να μην περιέχει ενεργές δοσοληψίες (πλήρες)

Ένα πλήρες χρονοπρόγραμμα (schedule) S των δοσοληψιών T_1, T_2, .., T_n είναι ένα σύνολο από πράξεις και μια *μερική* διάταξη των πράξεων αυτών με τους ακόλουθους περιορισμούς:

(i) οι πράξεις του S είναι ακριβώς οι πράξεις των T_1, T_2, .., T_n συμπεριλαμβανομένης μιας πράξης ακύρωσης ή επικύρωσης ως τελευταίας πράξης σε κάθε δοσοληψία στο χρονοπρόγραμμα

(ii) για κάθε δοσοληψία T_i που συμμετέχει στο S οι πράξεις της T_i στο S πρέπει να εμφανίζονται με την ίδια σειρά που εμφανίζονται στην T_i

(iii) Για κάθε ζεύγος *συγκρουόμενων* πράξεων, μια από τις δύο πρέπει να προηγείται της άλλης στο χρονοπρόγραμμα

Επικυρωμένη προβολή C(S) ενός χρονοπρογράμματος S η οποία περιλαμβάνει μόνο τις πράξεις του S που ανήκουν σε επικυρωμένες δοσοληψίες

Ορισμοί

√ 1. δοσοληψία

√ 2. διαπεπλεγμένη εκτέλεση δοσοληψιών (χρονοπρόγραμμα)

⇒ 3. σωστό -- αποδεκτό χρονοπρόγραμμα

Ισοδυναμία με σειριακό χρονοπρόγραμμα

- Σειριακά Χρονοπρογράμματα: χρονοπρογράμματα που δεν διαπλέκουν πράξεις διαφορετικών δοσοληψιών (οι πράξεις κάθε δοσοληψίας εκτελούνται διαδοχικά, χωρίς παρεμβολή πράξεων από άλλη δοσοληψία)

Παρατήρηση: Αν κάθε δοσοληψία διατηρεί τη συνέπεια, τότε κάθε σειριακό χρονοπρόγραμμα διατηρεί τη συνέπεια

Ένα σειριακό χρονοπρόγραμμα είναι σωστό

S: $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1 $R_2(X)$ $W_2(X)$ C_2

• Ισοδύναμα Χρονοπρογράμματα :

Για κάθε κατάσταση της ΒΔ, το αποτέλεσμα της εκτέλεσης του πρώτου χρονοπρογράμματος είναι το ίδιο με το αποτέλεσμα του δεύτερου χρονοπρογράμματος

Ένα χρονοπρόγραμμα ισοδύναμο με ένα σειριακό είναι σωστό

• Ισοδύναμα Χρονοπρογράμματα : Για κάθε κατάσταση της ΒΔ, το αποτέλεσμα της εκτέλεσης του πρώτου χρονοπρογράμματος είναι *το ίδιο* με το αποτέλεσμα του δεύτερου χρονοπρογράμματος

Αρκεί;

Είναι δυνατόν να ελεγχθεί;

• Σειριοποιήσιμο Χρονοπρόγραμμα :

Ένα χρονοπρόγραμμα που είναι *ισοδύναμο* με κάποιο σειριακό

Τι σημαίνει ισοδύναμο;

Τι σημαίνει ισοδύναμο;

- Ισοδυναμία βάσει συγκρούσεων
- Ισοδυναμία όψεων

• Ισοδύναμα Χρονοπρογράμματα βάσει Συγκρούσεων:

Δυο χρονοπρογράμματα είναι ισοδύναμα βάσει συγκρούσεων αν η διάταξη κάθε ζεύγους συγκρουόμενων πράξεων είναι ίδια και στα δυο χρονοπρογράμματα.

S1: $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

Παραδείγματα

S2: $R_2(X)$ $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

S3: $R_2(X)$ $W_2(X)$ C_2 $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1

S4: $R_2(X)$ $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1 $W_2(X)$ C_2

- Σειριοποιησιμότητα βάσει Συγκρούσεων:

Ένα χρονοπρόγραμμα S είναι σειριοποιήσιμο βάσει συγκρούσεων αν είναι ισοδύναμο βάσει συγκρούσεων με κάποιο σειριακό χρονοπρόγραμμα S'.

•Σε αυτήν την περίπτωση μπορούμε να αναδιατάξουμε τις μη συγκρουόμενες πράξεις στο S μέχρι να σχηματίσουμε ένα ισοδύναμο σειριακό χρονοπρόγραμμα.

S1: $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

S2: $R_2(X)$ $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_2(X)$ C_2 $W_1(Y)$ C_1

Σειριοποιήσιμα;

Sα: $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1 $R_2(X)$ $W_2(X)$ C_2

Sβ: $R_2(X)$ $W_2(X)$ C_2 $R_1(X)$ $W_1(X)$ $R_1(Y)$ $W_1(Y)$ C_1

⌘ Θεωρείστε δύο συναλλαγές (*Xacts*):

T1:	BEGIN A=A+100, B=B-100 END
T2:	BEGIN A=1.06*A, B=1.06*B END

- Διαισθητικά, η πρώτη μεταφέρει $100 από το λογαριασμό B στο λογαριασμό A. Η δεύτερη καταθέτει και στους δύο τόκο 6%.
- Δεν υπάρχει καμία εγγύηση ότι η T1 θα εκτελεστεί πριν την T2 η το ανάποδο, αν και η δύο υποβληθούν ταυτόχρονα. Ωστόσο, το συνολικό αποτέλεσμα πρέπει να είναι ισοδύναμο με τη μία ή την άλλη περίπτωση (δηλαδή, με κάποια *σειριακή εκτέλεση* των δύο δοσοληψιών)

⌘ Θεωρείστε ένα πιθανό χρονοπρόγραμμα:

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

- Αυτό είναι OK. Αλλά:

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

- Το δεύτερο χρονοπρόγραμμα:

T1:	$R_1(A)$ $W_1(A)$,	$R_1(B)$, $W_2(B)$
T2:	$R_2(A)$, $W_2(A)$ $R_2(B)$ $W_2(B)$	

⌘ Ανάγνωση Uncommitted δεδομένων (WR Συγκρούσεις "dirty reads", προσωρινή ενημέρωση):

T1:	$R_1(A)$ $W_1(A)$	$R_1(B)W_1(B)$, Abort
T2:	$R_2(A)$ $W_2(A)$, C	

⌘ Μη επαναλήψιμες αναγνώσεις (RW Συγκρούσεις):

T1:	$R_1(A)$	$R_1(A)$ $W_1(A)$ C
T2:	$R_2(A)$ $W_2(A)$ C	

⌘ Απώλειες Ενημερώσεων (WW Συγκρούσεις):

T1:	$W_1(A)$	$W_1(B)$ C
T2:	$W_2(A)$ $W_2(B)$ C	

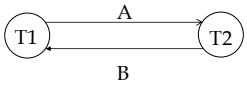
Υπάρχει τρόπος να ελέγξουμε *αποδοτικά* αν ένα χρονοπρόγραμμα είναι σωστό, δηλαδή σειριοποιήσιμο βάσει συγκρούσεων;

Γράφος προήγησης (precedence graph) ή γράφος σειριοποιησιμότητας (serialization graph)

Κόμβος :: Δοσοληψία

Ακμή $T_i \rightarrow T_j$ αν μια πράξη της T_i προηγείται μιας συγκρουόμενης πράξης της T_j

T1: $R_1(A)\ W_1(A),$ $R_1(B)\ W_1(B)$
T2: $R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$



Η ετικέτα στην ακμή δείχνει σε πιο δεδομένο συγκρούονται

S1: $R_1(X)\ R_2(X)\ W_1(X)\ R_1(Y)\ W_2(X)\ C_2\ W_1(Y)\ C_1$

S2: $R_1(X)\ R_2(X)\ W_2(X)\ C_2\ W_1(X)\ R_1(Y)\ W_1(Y)\ C_1$

Γράφοι:

Θεώρημα

Ένα χρονοπρόγραμμα είναι σειριοποιήσιμο (βάσει συγκρούσεων) αν και μόνο αν ο γράφος προήγησής του είναι ακυκλικός.

• ζητούμενο: όταν μια δοσοληψία επικυρωθεί δεν θα χρειαστεί ποτέ να ανακληθεί

• Χρονοπρογράμματα με δυνατότητα ανάκαμψης

αν καμιά δοσοληψία T στο S δεν επικυρώνεται έως ότου επικυρωθούν όλες οι δοσοληψίες οι οποίες τροποποίησαν ένα δεδομένο που διαβάζει η T

• Χρονοπρογράμματα με δυνατότητα ανάκαμψης: καμιά δοσοληψία T στο S δεν επικυρώνεται έως ότου επικυρωθούν όλες οι δοσοληψίες οι οποίες τροποποίησαν ένα δεδομένο που διαβάζει η T

$R_1(X)\ W_1(X)\ R_2(X)\ R_1(Y)\ W_2(X)\ C_2\ A_1$

$R_1(X)\ W_1(X)\ R_2(X)\ R_1(Y)\ W_2(X)\ W_1(Y)\ C_1\ C2$

• πρόβλημα: διαδιδόμενη ανάκληση (όταν μια δοσοληψία πρέπει να ανακληθεί γιατί διάβασε κάποιο στοιχείο από μια δοσοληψία που απέτυχε)

• Χρονοπρογράμματα που αποφεύγουν τη διάδοση ανακλήσεων

αν κάθε δοσοληψία T στο S διαβάζει μόνο στοιχεία που έχουν γραφεί από επικυρωμένες δοσοληψίες

• Αυστηρά Χρονοπρογράμματα

οι δοσοληψίες δεν μπορούν ούτε να διαβάσουν ούτε να γράψουν ένα στοιχείο X έως ότου επικυρωθεί η δοσοληψία που έγραψε το X

Δυνατότητα επιστροφής στην before image

$W_1(X, 5)\ W_2(X, 9)\ A_1$

Αρχικά θα υποθέσουμε ότι δεν υπάρχουν αντίγραφα.

Τοπικό χρονοπρόγραμμα σε ένα κόμβο K: Η προβολή του χρονοπρογράμματος στον κόμβο K, δηλαδή μόνο οι πράξεις του χρονοπρογράμματος που περιλαμβάνουν δεδομένα στον κόμβο K

Ολικό χρονοπρόγραμμα

***Βασικό ερώτημα**: Αν τα τοπικά χρονοπρογράμματα είναι σειριοποιήσιμα, είναι και το ολικό σειριοποιήσιμο;*

Ναι αν, η σειρά σειριοποιησιμότητας τους είναι συμβατή

Παράδειγμα

(Ολικό) χρονοπρόγραμμα

$R_1(X)\ W_2(Y)\ W_2(X)\ R_1(Y)\ C_2\ C_1$

Τοπικό χρονοπρόγραμμα στον Κόμβο 1:

$R_1(X)\ W_2(X)$

Τοπικό χρονοπρόγραμμα στον Κόμβο 2:

$W_2(Y)\ R_1(Y)$

• Ο χρήστης δεν ασχολείται με τη συνδρομικότητα

• Το ΣΔΒΔ εξασφαλίζει «σωστή συνδρομικότητα», γενικά δρομολογεί τις πράξεις των δοσοληψιών ώστε να προκύπτουν χρονοπρογράμματα σειριοποιήσιμα βάσει συγκρούσεων

• μέσω τεχνικών ελέγχου συνδρομικότητας

Τεχνικές

1. Κλειδώματος (locking) για να αποτρέψουν τη συνδρομική (ταυτόχρονη) προσπέλαση των δεδομένων από πολλές δοσοληψίες

2. Διάταξης χρονοσημάτων (timestamps)

3. Πιστοποίησης (validation) μιας δοσοληψίας (αισιόδοξα πρωτόκολλα)

- ένα κλειδί ανά δεδομένο
- μια δοσοληψία πριν προσπελάσει ένα δεδομένο X ζητά ένα κλειδί -- αίτηση **lock(X)**
- μπορεί να προσπελάσει το δεδομένο, μόνο αφού της δοθεί το κλειδί -- *πότε παίρνει το κλειδί;*
- μια δοσοληψία μπορεί να άρει το κλειδί στο δεδομένο -- αίτηση **unlock (X)**

Αρκεί αυτό για να δώσει σειριοποιήσιμα χρονοπρογράμματα;

Γενικά όχι. Χρειάζεται (όπως θα δούμε) διάταξη των πράξεων lock-unlock κάθε δοσοληψίας

ένα απλό κλείδωμα:

- στην πιο απλή περίπτωση, ένα μόνο είδος κλειδιού
- **lock(X)** :: πραγματοποιείται αν το δεδομένο δεν είναι ήδη κλειδωμένο, αλλιώς η δοσοληψία περιμένει μέχρι να ελευθερωθεί το δεδομένο
- μια δομή (πίνακας) (δεδομένο, μια ένδειξη (κλειδωμένο - μη-κλειδωμένο), ουρά με δοσοληψίες που περιμένουν)
- **unlock (X)**

Ιδέα: πολλές δοσοληψίες να μπορούν να διαβάσουν ένα δεδομένο ταυτόχρονα

Δύο ειδών κλειδιά:

- διαμοιραζόμενο (shared) κλειδί ή κλειδί ανάγνωσης
- αποκλειστικό (exclusive) κλειδί ή κλειδί εγγραφής

- μια δοσοληψία πριν διαβάσει ένα δεδομένο X ζητά ένα διαμοιραζόμενο κλειδί -- αίτηση **S-lock(X)**
- μια δοσοληψία πριν γράψει ένα δεδομένο X ζητά ένα αποκλειστικό κλειδί -- αίτηση **X-lock(X)**
- η αίτηση για κλειδί δίνεται αν δεν υπάρχει «συγκρούμενο κλειδί»
- (πάλι) μια δοσοληψία μπορεί να άρει το κλειδί στο δεδομένο -- αίτηση **unlock (X)**

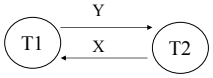
Πίνακας συμβατότητας κλειδιών

	S-Lock(X)	X-Lock(X)
S-Lock(X)	√	
X-Lock(X)		

T1	T2
S-Lock(Y)	
$R_1(Y)$	
Unlock(Y)	
	S-Lock(X)
	$R_2(X)$
	Unlock(X)
	X-Lock(Y)
	$W_2(Y)$
	Unlock(Y)
	C_2
S-Lock(X)	
$W_1(X)$	
Unlock(X)	
C_1	

Δεν αρκεί για σειριοποιησιμότητα

S: $R_1(Y)$ $R_2(X)$ $W_2(Y)$ C_2 $W_1(X)$ C_1



Λύση; Κλείδωμα Δύο Φάσεων

- Πρωτόκολλο κλειδώματος δυο φάσεων (Two-Phase Locking 2PL)

Όλες οι πράξεις (αιτήσεις) κλειδώματος μιας δοσοληψίας προηγούνται της πρώτης πράξης (αίτησης) άρσης κλειδώματος της διαδικασίας

Δηλαδή, μόλις μια δοσοληψία αφήσει (unlock) ένα κλειδί δεν μπορεί να ζητήσει ξανά κλειδί

Κάθε δοσοληψία δυο φάσεις

- μια φάση επέκτασης ή εξάπλωσης
- μια φάση συρρίκνωσης

Παρατηρήσεις

- Οι αιτήσεις lock και unlock πρέπει να είναι ατομικές πράξεις
- Αναβάθμιση κλειδιού: μια δοσοληψία που κατέχει ένα διαμοιραζόμενο κλειδί μπορεί να αναβαθμιστεί ώστε να κατέχει ένα αποκλειστικό κλειδί
- Η σειρά σειριοποιησιμότητας ακολουθεί τη σειρά με την οποία οι δοσοληψίες περνούν από τη φάση επέκτασης στη φάση συρίκνωσης

- **Κεντρικοποιημένο κλείδωμα**: ένας κόμβος του συστήματος αναλαμβάνει να διαχειρίζεται τα κλειδιά για όλα τα δεδομένα – οι πίνακες κλειδιών αποθηκεύονται σε αυτόν τον κόμβο που καλείται διαχειριστής κλειδιών
- **Κλείδωμα πρωτεύοντος κόμβου**: οι αρμοδιότητες του διαχειριστή κλειδιών μοιράζονται ανάμεσα στους κόμβους, δηλαδή κάθε κόμβους αναλαμβάνει τη διαχείριση κλειδιών κάποιων από τα δεδομένα

  Θέμα: για κάθε δεδομένο θα πρέπει να ξέρουμε ποιος κόμβος διαχειρίζεται τα κλειδιά του
- **Αποκεντρικοποιημένο κλείδωμα**: Κάθε κόμβος αναλαμβάνει τη διαχείριση των κλειδιών για τα δεδομένα που είναι αποθηκευμένα σε αυτόν

Οι τεχνικές κλειδώματος μπορεί να προκαλέσουν αδιέξοδα (deadlocks)

T1	T2
S-Lock(Y)	
$R_1(Y)$	
	S-Lock(X)
	$R_2(X)$
X-Lock(X)	
	X-Lock(Y)

Η T1 περιμένει την T2 να ελευθερώσει το X, και η T2 περιμένει την T1 να ελευθερώσει το Y

Δυο τεχνικές:

• Πρωτόκολλα Πρόληψης Αδιεξόδων (Deadlock Prevention): Αποφυγή δημιουργίας αδιεξόδου

• Πρωτόκολλα Ανίχνευσης Αδιεξόδου (Deadlock Detection): Ελέγχουμε περιοδικά αν το σύστημα βρίσκεται σε κατάσταση αδιεξόδου

• Κάθε δοσοληψία πρέπει να κλειδώσει *όλα* τα δεδομένα που χρειάζεται πριν ξεκινήσει

• Αν δε μπορεί να κλειδώσει έστω και ένα, δε κλειδώνει κανένα και προσπαθεί ξανά

«σπάσιμο» του κύκλου ⇒ κάποια διάταξη μεταξύ των δοσοληψιών

Κάθε δοσοληψία T έχει ένα χρονόσημο TS(T):

Μια διαδικασία παίρνει ένα χρονόσημο κατά την εκκίνησή της.

TS(T1) < TS(T2), σημαίνει ότι η T1 ξεκίνησε πριν την T2

ιδέα: μια δοσοληψία περιμένει μόνο αν το κλειδί το έχει μια δοσοληψία με *μικρότερο* (μεγαλύτερο) χρονόσημο, αλλιώς ακυρώνεται

Δύο σχήματα

• αναμονής-θανάτωσης

• τραυματισμού-αναμονής

Έστω ότι η T_i ζητά να κλειδώσει το X που είναι κλειδωμένο από την T_j

• αναμονή-θανάτωση

Αν $TS(T_i) < TS(T_j)$ **ευνοείται η παλιότερη**

Ti περιμένει

αλλιώς

ακυρώνεται η T_i και επανεκκινείται με *το ίδιο* χρονόσημα

• τραυματισμός -αναμονή

Αν $TS(T_i) > TS(T_j)$ **ευνοείται η νεότερη**

Ti περιμένει

αλλιώς

ακυρώνεται η T_j και επανεκκινείται με *το ίδιο* χρονόσημα

Κατασκευή γράφου αναμονής (wait-for graph)

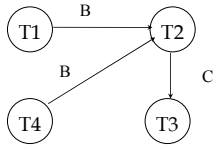
κόμβοι : δοσοληψίες

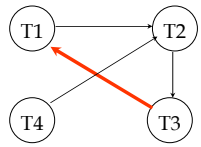
ακμή από τον κόμβο T_i στον T_j, αν η T_i περιμένει την T_j να αφήσει ένα κλειδί

• Περιοδικά έλεγχος για κύκλους στο γράφο αναμονής

Παράδειγμα:

T1: S-Lock(A), R(A), S-Lock(B)
T2: X-Lock(B),W(B) X-Lock(C)
T3: S-Lock(C), R(C) X-Lock(A)
T4: X-Lock(B)



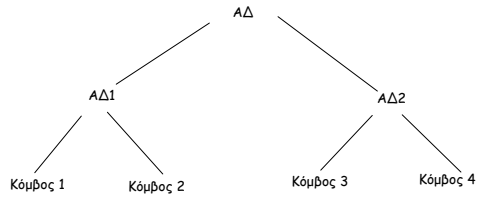


Κάθε διαχειριστής κλειδιών έχει έναν τοπικό γράφο αναμονής

Κεντρικοποιημένη προσέγγιση: Ένας κόμβος παίζει το ρόλο του ανιχνευτή αδιεξόδων. Οι διαχειριστές κλειδιών μεταφέρουν περιοδικά το τοπικό γράφο αναμονής στον ανιχνευτή αδιεξόδων, ο οποίος τους συνδυάζει.

- Περίοδος αποστολής;
- Κατάλληλη στην περίπτωση του κεντρικοποιημένου κλειδώματος

Ιεραρχική προσέγγιση: ιεραρχία από ανιχνευτές αδιεξόδων



Αποκεντρικοποιημένη προσέγγιση

Κάθε κόμβος στέλνει τον τοπικό κύκλο αναμονής σε όλους του άλλους κόμβους

Κάθε κόμβος ανιχνεύει το αδιέξοδο που μπορεί να είναι είτε τοπικό είτε να περιλαμβάνει παραπάνω από έναν κόμβους

Τεχνικές

√ 1. Κλειδώματος (locking)

→ 2. Διάταξης χρονοσημάτων (timestamps)

3. Πιστοποίησης (validation)

Το χρονόσημα δημιουργείται από το ΣΔΒΔ και προσδιορίζει μοναδικά μια δοσοληψία

Ιδέα: *διάταξη των δοσοληψιών με βάση το χρονόσημα τους* (δηλαδή, χρονοπρόγραμμα ισοδύναμο με σειριακό στο οποίο οι δοσοληψίες εμφανίζονται διατεταγμένες με βάση τις τιμές των χρονοσημάτων)

⇒ άρα η σειρά προσπέλασης στα δεδομένα πρέπει να μη παραβιάζει τη σειριοποιησιμότητα

Δηλαδή: αν μια πράξη a_i μιας δοσοληψίας T_i συγκρούεται με μια πράξη a_j μιας δοσοληψίας T_j και $TS(T_i) < TS(T_j)$, τότε η a_i πρέπει να προηγείται της a_j. Αλλιώς, restart τη δοσοληψία.

Κάθε δεδομένο X έχει δύο τιμές χρονοσημάτων:

ΧΣΑ(Χ) (χρονόσημα ανάγνωσης) το μεγαλύτερο μεταξύ όλων των χρονοσημάτων των δοσοληψιών που διάβασαν το Χ

ΧΣΕ(Χ) (χρονόσημα εγγραφής) το μεγαλύτερο μεταξύ όλων των χρονοσημάτων των δοσοληψιών που έγραψαν το Χ

Η δοσοληψία Τ με ΧΣ(Τ) εκτελεί μια πράξη ανάγνωσης R(X)

Αν ΧΣ(Τ) < ΧΣΕ(Χ) (αυτό παραβιάζει τη διάταξη)

η Τ ακυρώνεται,

μπορεί να ξαναρχίσει αλλά με μεγαλύτερο χρονόσημα (γιατί;)

Αν ΧΣ(Τ) > ΧΣΕ(Χ)

η ανάγνωση είναι επιτρεπτή

Θέσε το ΧΣΑ(Χ) = max{ΧΣΑ(Τ), ΧΣ(ΤΑ)}

• Οι αλλαγές στο ΧΣΑ(Χ) πρέπει να γράφονται στο δίσκο! Αυτό και το ότι η δοσοληψίες ξαναρχίζουν προκαλεί overhead

Η δοσοληψία Τ με ΧΣ(Τ) εκτελεί μια πράξη εγγραφής W(X)

Αν ΧΣΑ(Χ) > ΧΣ(Τ) ή ΧΣΕ(Χ) > ΧΣ(Τ)

η Τ ακυρώνεται (γιατί;)

Βελτιστοποίηση: τι σημαίνει ΧΣΕ(Χ) > ΧΣ(Τ)

Ο κανόνας του Thomas για εγγραφές (Thomas Write Rule) Μπορούμε να αγνοήσουμε μερικές «ξεπερασμένες» εγγραφές, δε χρειάζεται επανεκκίνηση της Τ (η εγγραφή της Τ ακολουθείται από άλλη εγγραφή, χωρίς ενδιάμεση ανάγνωση)

T1	T2
R(A)	
	W(A)
	Commit
W(A)	
Commit	

Επιτρέπει σειριοποίησιμα - αλλά όχι σειριοποιήσιμα βάσει συγκρούσεων

- Δυστυχώς, παράγει και χρονοπρογράμματα χωρίς δυνατότητα ανάκαμψης

T1	T2
W(A)	
	R(A)
	W(B)
	Commit

⌘ Τροποποίηση
- Buffer all writes μέχρι την επικύρωση του writer (αλλά το ΧΣΕ(Χ) τροποποιείται κανονικά
- Block readers T (όπου ΧΣ(Τ) > ΧΣΕ(Χ)) μέχρι να επικυρωθεί ο writer του Χ

⌘ Όμοια με το να κρατούν οι writers X-Locks μέχρι την επικύρωση τους αλλά όχι ακριβώς 2PL

Τεχνικές

√ 1. Κλειδώματος (locking)

√ 2. Διάταξης χρονοσημάτων (timestamps)

→ 3. Πιστοποίησης (validation)

Οι τεχνικές κλειδώματος είναι συντηρητικές (αποφεύγονται οι συγκρούσεις)

Μειονεκτήματα

- επιβάρυνση (overhead) χειρισμού κλειδώματος
- αποφυγή/ανίχνευση αδιεξόδων
- lock contention για τα δεδομένα που χρησιμοποιούνται συχνά

Αν οι συγκρούσεις είναι σπάνιες, μεγαλύτερη συγχρονικότητα, αν αντί για κλείδωμα, έλεγχος για συγκρούσεις όταν μια δοσοληψία επικυρώνεται (commits)

Κάθε δοσοληψία έχει τρεις φάσεις

- **ΑΝΑΓΝΩΣΗ**: η δοσοληψία διαβάζει από τη βδ, αλλά τροποποιεί προσωπικά αντίγραφα των δεδομένων
- **ΠΙΣΤΟΠΟΙΗΣΗ**: έλεγχος για συγκρούσεις
- **ΕΓΓΡΑΦΗ**: γράφει τα τοπικά αντίγραφα στη βδ

Έλεγχος συνθηκών που είναι **ικανές** για να εξασφαλίσουν ότι δεν υπήρχαν συγκρούσεις

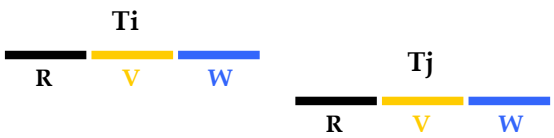
- Κάθε δοσοληψία Τ έχει ένα μοναδικό αριθμό TID (χρονόσημα)
- Το TID ανατίθεται στο τέλος της φάσης ΑΝΑΓΝΩΣΗΣ (ακριβώς πριν αρχίσει η πιστοποίηση)
- Με κάθε δοσοληψία

ReadSet(T): το σύνολο των δεδομένων που διάβασε η Τ

WriteSet(T): το σύνολο των δεδομένων που διάβασε η Τ

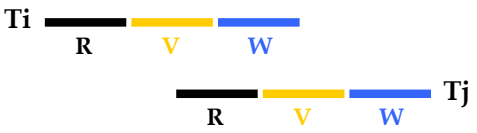
ΕΛΕΓΧΟΣ 1

⌘ Για όλα τα i και j τέτοια ώστε Ti < Tj, η Ti τελειώνει πριν αρχίσει η Tj.



ΕΛΕΓΧΟΣ 2

⌘ Για όλα τα i και j τέτοια ώστε Ti < Tj:
- η Ti τελειώνει πριν αρχίσει η φάση εγγραφής της Tj
- WriteSet(Ti) ∩ ReadSet(Tj) = ∅



ΕΛΕΓΧΟΣ 3

⌘ Για όλα τα i και j τέτοια ώστε Ti < Tj:
- η Ti τελειώνει τη φάση ανάγνωσης πριν αρχίσει η φάση ανάγνωσης της Tj
- WriteSet(Ti) ∩ ReadSet(Tj) = ∅
- WriteSet(Ti) ∩ WriteSet(Tj) = ∅

⌘ Πιστοποίηση της T (έλεγχος 1 & 2):

```
valid = true;
// S = set of Xacts that committed after Begin(T)
< foreach  Ts in S do {
  if ReadSet(Ts) does  intersect WriteSet(T)
        then valid = false;
  }
  if valid then { install updates; // Write phase
                  Commit T } >
           else Restart T
```

τέλος κρίσιμης περιοχής