Επεξεργασία Δοσοληψιών

Ανακεφαλαίωση

Πρόβλημα

«Σωστή» εκτέλεση προγραμμάτων προσπέλασης μίας βδ, στην περίπτωση:

- *ταυτοχρονισμού*
- *αποτυχιών*

Βασικές έννοιες:

- Δοσοληψία (transaction)
- Χρονοπρόγραμμα (schedule/history)

Ιδιότητες δοσοληψιών

«Σωστά» χρονοπρογράμματα (από δύο πλευρές:

- συνδρομικότητα
- ανάρρωση από αποτυχίες

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,

Από την πλευρά του ΣΔΒΔ δυο βασικές πράξεις (δεδομένων)

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

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**

- 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

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

- Παράδειγμα:: Θεωρείστε τις 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

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

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

Βασική υπόθεση: εκτέλεση μίας δοσοληψίας μόνη της και χωρίς αποτυχίες αφήνει τη βάση δεδομένων σε συνεπή κατάσταση

- Η συνδρομικότητα (concurrency) *επιτυγχάνεται από το ΣΔΒΔ* που διαπλέκει τις πράξεις (αναγνώσεις/εγγραφές) των διαφόρων δοσοληψιών

Διαπλοκές που είναι «σωστές», συνεπή κατάσταση + απομόνωση

Αποτυχίες, ατομικότητα και διάρκεια

- Ένα χρονοπρόγραμμα 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

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

• 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

Επίσης, απομόνωση

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: σειριοποιησιμότητα (serializability)

- Δυο χρονοπρογράμματα S1 και S2 είναι **ισοδύναμα βάσει όψεων** αν:
  - Αν στο S1, η T_i διαβάζει την αρχική τιμή του A, τότε η T_i επίσης διαβάζει την αρχική τιμή του A στο S2
  - Αν στο S1, η T_i *διαβάζει* την τιμή του A που έγραψε η T_j, τότε η T_i διαβάζει την τιμή του A που έγραψε η T_j και στο S2
  - Αν στο S1, η T_i *γράφει την τελική τιμή* του A, τότε η T_i γράφει την τελική τιμή του A και στο S2

Σειριοποιησιμότητα βάσει Όψεων: Ένα χρονοπρόγραμμα S είναι σειριοποιήσιμο βάσει όψεων αν είναι ισοδύναμο βάσει όψεων με κάποιο σειριακό χρονοπρόγραμμα S'.

T1: R(A)		W(A)	
T2:	W(A)		
T3:			W(A)

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)

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

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

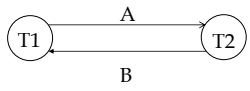
Σε αυτήν την περίπτωση μπορούμε να αναδιατάξουμε τις μη συγκρουόμενες πράξεις στο S μέχρι να σχηματίσουμε ένα ισοδύναμο σειριακό χρονοπρόγραμμα.

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 (απλώς διακοσμητική!)

Θεώρημα

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

Σειριακά $\subset$

Σειριοποιήσιμα βάσει συγκρούσεων $\subset$

Σειριοποιήσιμα βάσει όψεων $\subset$

Όλα τα χρονοπρογράμματα

T1: R(A)		W(A)
T2:	W(A)	
T3:		W(A)

χρονοπρογράμματα ισοδύναμα με σειριακά αφήνουν τη βάση δεδομένων σε συνεπή κατάσταση

Αρκεί;

Όχι, στην περίπτωση αποτυχιών

Θα δούμε κάποιες ιδιότητες που εξασφαλίζουν «εύκολη» ανάρρωση

- Χρονοπρογράμματα με δυνατότητα ανάκαμψης (recoverable)

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

- Χρονοπρογράμματα χωρίς διάδοση ανακλήσεων (avoids cascading aborts)

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

- Αυστηρά Χρονοπρογράμματα (strict)

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

Παράδειγμα

Χαρακτηρίστε καθένα από τα παρακάτω χρονοπρογράμματα:

- σειριοποιήσιμα βάσει συγκρούσεων, σειριοποιήσιμα βάσει όψεων
- με δυνατότητα ανάκαμψης, χωρίς διάδοση ανακλήσεων, αυστηρά

$R_1(X)\ W_2(X)\ W_1(X)\ A_2\ C_1$

$R_1(X)\ W_2(X)\ C_2\ W_1(X)\ C_1\ R_3(X)\ C_3$

$R_1(X)\ W_2(X)\ W_1(X)\ C_1\ R_3(X)\ C_3$ - γιατί θα μπορούσε να γίνει abort C_2

$R_2(X)\ W_3(X)\ C_3\ W_1(Y)\ C_1\ R_2(Y)\ W_2(Z)\ C_2$

Υπόδειξη:

(1) σειριοποιήσιμα βάσει συγκρούσεων (με χρήση του θεωρήματος)

(2) σειριοποιήσιμα βάσει όψεων (έλεγχος με όλα τα πιθανά σειριακά)

Επίσης, (1) $\Rightarrow$ (2)

Παράδειγμα

Έστω το χρονοπρόγραμμα

$R_1(X)\ R_1(Y)\ W_1(X)\ R_2(Y)\ W_3(Y)\ W_1(X)\ R_2(Y)$

(α) Υποθέτοντας ότι και οι τρεις δοσοληψίες τελικά επικυρώνονται, δώστε το γράφο σειριοποιημότητας

(β) Τροποιείστε το χρονοπρόγραμμα ώστε ένα πλήρες χρονοπρόγραμμα με τις παρακάτω ιδιότητες

i. δυνατότητα ανάκαμψης, αλλά όχι χωρίς διάδοση ανακλήσεων

ii. χωρίς διάδοση ανακλήσεων

iii. σειριοποιήσιμο βάσει συγκρούσεων

Τεχνικές Ελέγχου Συνδρομικότητας

- 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

Δηλαδή, ψάχνουμε αλγόριθμους (πρωτόκολλα) που θα δρομολογούν τις πράξεις των δοσοληψιών

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)

Η πιο απλή εκδοχή: 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)**

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

  Υλοποίηση: μια δομή (πίνακας) (δεδομένο, μια ένδειξη (κλειδωμένο - μη-κλειδωμένο), ουρά με δοσοληψίες που περιμένουν)

- **unlock (X)**

Δυο προβλήματα:

1. Δεν επιτρέπει ταυτόχρονες αναγνώσεις
2. Δε δουλεύει! (δηλαδή, δεν αρκεί για να δώσει σειριοποιήσιμα χρονοπρογράμματα)

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

Ας διορθώσουμε το πρώτο

Στόχος: 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)**

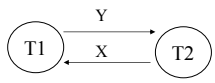
Πίνακας συμβατότητας κλειδιών

	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

Αποδεικνύεται ότι είναι σωστό

Συγκεκριμένα, σειριοποιούνται με βάση τη σειρά που εισέρχονται στη φάση συρρίκνωσης (εύκολη απόδειξη με χρήση του γράφου σειριοοποιησιμότητας και απαγωγή σε άτοπο)

Τροποποίηση ώστε να είναι αυστηρό;

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

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

Ονομάζεται συντηρητικό ή στατικό κλείδωμα δυο φάσεων

«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

Και στα δύο σχήματα οι παλιές δοσοληψίες «εκτοπίζουν» τις νεώτερες

Πρόληψη Αδιεξόδων

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

• αναμονή-θανάτωση

Αν $TS(T_i) < TS(T_j)$

T_i περιμένει (* περιμένουμε αν το κλειδί το έχει νεώτερη *)

Αλλιώς (* το κλειδί το έχει παλαιότερη *)

ακυρώνεται (πεθαίνει!) η T_i και επανεκκινείται με το ίδιο (γιατί;) χρονόσημα

• τραυματισμός-αναμονή

Αν $TS(T_i) > TS(T_j)$

T_i περιμένει (* περιμένουμε αν το κλειδί το έχει παλαιότερη *)

Αλλιώς (* το κλειδί το έχει νεώτερη *)

ακυρώνεται (τραυματίζεται) η T_j και επανεκκινείται με το ίδιο χρονόσημα

Ανίχνευση Αδιεξόδων

Κατασκευή γράφου αναμονής (wait-for graph)

κόμβοι : δοσοληψίες

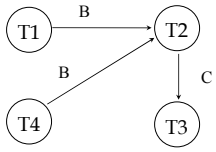
ακμή από τον κόμβο T_i στον T_j, αν η T_i περιμένει την T_j να αφήσει ένα κλειδί

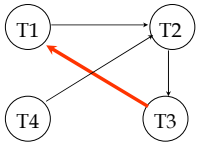
• Περιοδικά έλεγχος για κύκλους στο γράφο αναμονής

Ανίχνευση Αδιεξόδων

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)





Τεχνικές Κλειδώματος: Ανακεφαλαίωση

Δυο τεχνικές:

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

  • όλα τα κλειδιά δίνονται μαζί

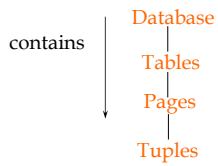
  • διάταξη των δοσοληψιών (με χρήση χρονοσήμων)

• Πρωτόκολλα Ανίχνευσης Αδιεξόδου (Deadlock Detection): Ελέγχουμε περιοδικά αν το σύστημα βρίσκεται σε κατάσταση αδιεξόδου (π.χ με χρήση γράφου αναμονής)

Περισσότερες παραλλαγές στο βιβλίο

Διακριτότητα

⌘ Δύσκολο να αποφασίσουμε τη διακριτότητα (granularity) για το κλείδωμα, δηλαδή, τι κλειδώνουμε;



Τεχνικές Κλειδώματος: Ανακεφαλαίωση

Δύο ειδών κλειδιά:

• διαμοιραζόμενο (shared) κλειδί ή κλειδί ανάγνωσης

• αποκλειστικό (exclusive) κλειδί ή κλειδί εγγραφής

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

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

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

▪ Δυνατή η «αναβάθμιση» κλειδιών

Τεχνικές Κλειδώματος: Ανακεφαλαίωση

- Κάθε δοσοληψία δυο φάσεις
  - μια φάση επέκτασης ή εξάπλωσης και μια φάση συρρίκνωσης
- Αποδεικνύεται ότι είναι σωστό: παράγει χρονοπρογράμματα σειριοποιήσιμα με συγκρούσεις.
- Υπάρχουν σειριοποιήσιμα με συγκρούσεις χρονοπρογράμματα που δεν παράγονται με κλείδωμα δύο φάσεων; ΝΑΙ
- Παράγεται αυστηρό χρονοπρόγραμμα; Όχι.

Παραλλαγές:

Αυστηρό Κλείδωμα Δύο Φάσεων: μια δοσοληψία δεν απελευθερώνει κανένα *αποκλειστικό κλειδί* πριν επικυρωθεί ή ακυρωθεί

Υπερ-αυστηρό Κλείδωμα Δύο Φάσεων: μια δοσοληψία δεν απελευθερώνει κανένα (αποκλειστικό και διαμοιραζόμενο) κλειδί πριν επικυρωθεί ή ακυρωθεί

Τα πιο δημοφιλή πρωτόκολλα – ευρεία χρήση σε εμπορικά συστήματα

Τεχνικές Κλειδώματος: Ανακεφαλαίωση

- Πρόβλημα με τη δημιουργία αδιεξόδων
- Κόστος διαχείρισης κλειδιών
- Λιμοκτονία
  - δίκαια διαχείριση της ουράς με τα κλειδιά
  - αν γράφος για τα αδιέξοδα, προσοχή στην επιλογή του «θύματος»

Τεχνικές Ελέγχου Συνδρομικότητας

Τεχνικές

√ 1. Κλειδώματος (locking)

→ 2. Διάταξης χρονοσημάτων (timestamps)

3. Πιστοποίησης (validation)

Διάταξη Χρονοσημάτων

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

Διάταξη Χρονοσημάτων

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.

Διάταξη Χρονοσημάτων

Κάθε δεδομένο X έχει δύο τιμές χρονοσημάτων:

ΧΣΑ(Χ) (χρονόσημα ανάγνωσης) το μεγαλύτερο μεταξύ όλων των χρονοσημάτων των δοσοληψιών που διάβασαν το Χ (διαισθητικά, η πιο πρόσφατη που το διάβασε)

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

Κάθε πράξη ανάγνωσης και εγγραφής μιας δοσοληψίας συνοδεύεται από κατάλληλους ελέγχους του χρονοσήματος της δοσοληψίας και των χρονοσημάτων του αντίστοιχου δεδομένου.

Αν ο έλεγχος αποτύχει, η δοσοληψία απορρίπτεται και αρχίζει πάλι

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 επιβάρυνση (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» ή τυφλές 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 Τ (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

- Παράγει σωστά χρονοπρογράμματα
- Δε δημιουργούνται αδιέξοδα (αφού καμιά δοσοληψία δεν περιμένει)
- Λιμοκτονία (κυκλική επανεκκίνηση) είναι πιθανή
- Μπορεί να παραχθούν χρονοπρογράμματα που δεν έχουν δυνατότητα ανάκαμψης (not recoverable)

Παραλλαγή για χρονοπρόγραμμα χωρίς διάδοση ανακλήσεων (ιδέα: commit bit με κάθε δεδομένο – καθυστέρηση αναγνώσεων)

Παραλλαγή για αυστηρά (αναμονή αναγνώσεων και εγγραφών)

Άλλη ιδέα: R(X) ή W(X) από Τ με ΧΣ(Τ) > ΧΣΕ(Χ) (αν μικρότερο ακυρώνεται) περιμένει μέχρι να επικυρωθεί ή να ακυρωθεί η δοσοληψία που έγραψε το Χ (όχι αδιέξοδα)

Επιβάρυνση διαχείρισης χρονοσημάτων

Απόρριψη και επανεκκίνηση δοσοληψιών

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)

Τεχνικές

√ 1. Κλειδώματος (locking)

√ 2. Διάταξης χρονοσημάτων (timestamps)

→ 3. Πιστοποίησης (validation)

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)
- Για κάθε δοσοληψία διατηρούμε:

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

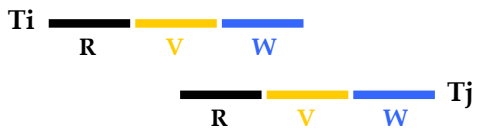
WriteSet(T): το σύνολο των δεδομένων που έγγραψε η Τ

ΕΛΕΓΧΟΣ Περίπτωση 1

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

ΕΛΕΓΧΟΣ Περίπτωση 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 Περίπτωση 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

- Υπάρχουν πολλά πρωτόκολλα ελέγχου συνδρομικότητας (concurrency control protocols)
- Τρεις βασικές κατηγορίες: κλείδωμα, χρονοσήματα, αισιόδοξα

Επόμενο Μάθημα:

Μερικά προχωρημένα θέματα ελέγχου συνδρομικότητας

Τι γίνεται με την SQL (μπορεί ο προγραμματιστής να καθορίσει τι θέλει);

Ανάκαμψη από σφάλματα (ποιες ιδιότητες των δοσοληψιών;)