Έλεγχος Ταυτοχρονισμού

Κεφάλαιο 17

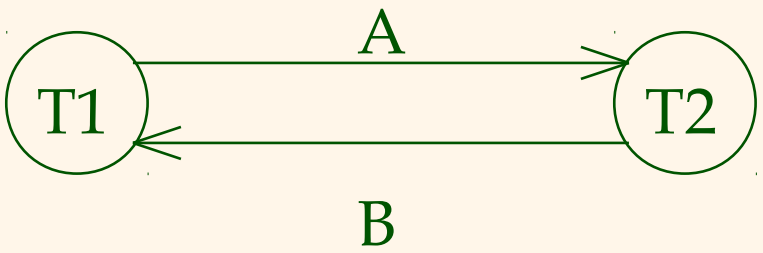
Συγκρουσιακώς Σειριοποιήσιμα Χρονοπρογράμματα

- Δυο χρονοπρογράμματα είναι συγκρουσιακώς ισοδύναμα αν:
  - Εμπλέκουν τις ίδιες ενέργειες των ίδιων συναλλαγών
  - Κάθε ζεύγος συγκρουόμενων ενεργειών είναι ταξινομημένο με τον ίδιο τρόπο
- Το χρονοπρόγραμμα S είναι συγκρουσιακώς σειριοποιήσιμο αν το S είναι συγκρουσιακώς ισοδύναμο με κάποιο σειριακό χρονοπρόγραμμα

Παράδειγμα

- Ένα χρονοπρόγραμμα που δεν είναι συγκρουσιακώς σειριοποιήσιμο:

T1:	R(A), W(A),	R(B), W(B)
T2:	R(A), W(A), R(B), W(B)	



Γράφημα Προτεραιότητας

- Το πρόβλημα φανερώνεται από τον κύκλο στο γράφημα. Το αποτέλεσμα της T1 εξαρτάται από την T2, και αντίστροφα.

Γράφημα Προτεραιότητας

- *Γράφημα Προτεραιότητας*: Ένας κόμβος ανά συναλλαγή. Ακμή από την *Ti προς της Tj* αν η *Tj διαβάζει/γράφει* ένα αντικείμενο που τελευταία έγραψε η *Ti*.
- Θεώρημα: Το χρονοπρόγραμμα είναι συγκρουσιακώς σειριοποιήσιμο αν και μόνο αν το γράφημα προτεραιότητάς του είναι ακυκλικό.

Αυστηρό 2PL

- *Αυστηρό Πρωτόκολλο Κλειδώματος δυο-φάσεων (Αυστηρό 2PL)*:
  - Κάθε συναλλαγή πρέπει να αποκτά μια S (*shared-διαμοιραζόμενη*) κλειδαριά σε ένα αντικείμενο πριν το διαβάσει, και μια X (*eXclusive-αποκλειστική*) κλειδαριά στο αντικείμενο πριν το γράψει.
  - Όλες οι κλειδαριές που διατηρεί μια συναλλαγή πρέπει να αποδεσμεύονται όταν η συναλλαγή ολοκληρώνει
  - Αν μια συναλλαγή διατηρεί μια κλειδαριά X σε ένα αντικείμενο, καμιά άλλη συναλλαγή δεν μπορεί να αποκτήσει κλειδαριά (S ή X) στο αντικείμενο.
- Το Αυστηρό 2PL επιτρέπει μόνο χρονοπρογράμματα με ακυκλικά γραφήματα προτεραιότητας

Κλείδωμα Δυο-Φάσεων (2PL)

- Πρωτόκολλο Κλειδώματος Δυο-Φάσεων
  - Κάθε συναλλαγή πρέπει να αποκτά μια S (*διαμοιραζόμενη*) κλειδαριά στο αντικείμενο πριν το διαβάσει, και μια X (*αποκλειστική*) κλειδαριά στο αντικείμενο πριν το γράψει.
  - Μια συναλλαγή δεν μπορεί να αιτηθεί απόκτηση επιπλέον κλειδαριών από τη στιγμή που αποδεσμεύσει κάποια κλειδαριά.
  - Αν μια συναλλαγή διατηρεί μια κλειδαριά X σε ένα αντικείμενο, καμιά άλλη συναλλαγή δεν μπορεί να αποκτήσει κλειδαριά (S ή X) στο αντικείμενο.

Διαχείριση Κλειδαριών

- Τις αιτήσεις απόκτησης και αποδέσμευσης κλειδαριών διαχειρίζεται ο διαχειριστής κλειδαριών
- Καταχώριση πίνακα κλειδαριών:
  - Πλήθος συναλλαγών που εκείνη τη στιγμή έχουν κλειδαριά
  - Τύπος κλειδαριάς (διαμοιραζόμενη ή αποκλειστική)
  - Δείκτης στην ουρά των αιτήσεων απόκτησης κλειδαριάς
- Το κλείδωμα και ξεκλείδωμα πρέπει να είναι ατομικές λειτουργίες
- Αναβάθμιση κλειδαριάς: μια συναλλαγή που διατηρεί μια διαμοιραζόμενη κλειδαριά μπορεί να την αναβαθμίσει σε αποκλειστική κλειδαριά

Αδιέξοδα

- Αδιέξοδο: Κύκλος συναλλαγών όπου η μια περιμένει την άλλη να αποδεσμεύσει κλειδαριές.
- Δυο τρόποι αντιμετώπισης αδιεξόδων:
  - Αποφυγή Αδιεξόδου
  - Ανίχνευση Αδιεξόδου

Αποφυγή Αδιεξόδου

- Ανάθεση προτεραιοτήτων με τη χρήση χρονόσημων (timestamps). Υποθέστε ότι η Ti θέλει μια κλειδαριά που διατηρεί η Tj. Δυο πιθανές στρατηγικές:
  - Wait-Die: Αν η Ti έχει υψηλότερη προτεραιότητα, η Ti περιμένει την Tj, αλλιώς η Ti εγκαταλείπει
  - Wound-wait: Αν η Ti έχει υψηλότερη προτεραιότητα, η Tj εγκαταλείπει, αλλιώς η Ti περιμένει
- Όταν μια συναλλαγή επανεκκινείται, πρέπει να έχει το αρχικό της χρονόσημο

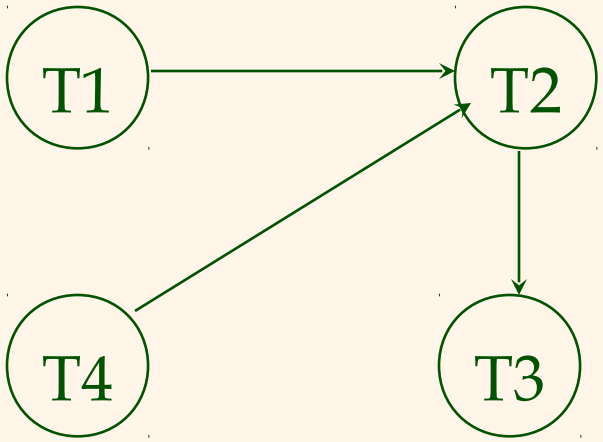
Ανίχνευση Αδιεξόδου

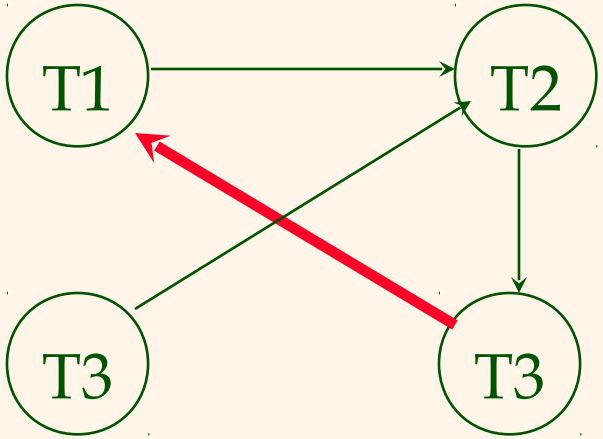
- Δημιουργούμε ένα γράφημα αναμονής (waits-for)
  - Οι κόμβοι είναι οι συναλλαγές
  - Υπάρχει μια ακμή από την Ti στην Tj αν η Ti περιμένει την Tj να αποδεσμεύσει μια κλειδαριά
- Περιοδικά, ελέγχουμε το γράφημα αναμονής για ύπαρξη κύκλων

Ανίχνευση Αδιεξόδου (συνέχεια)

Παράδειγμα:

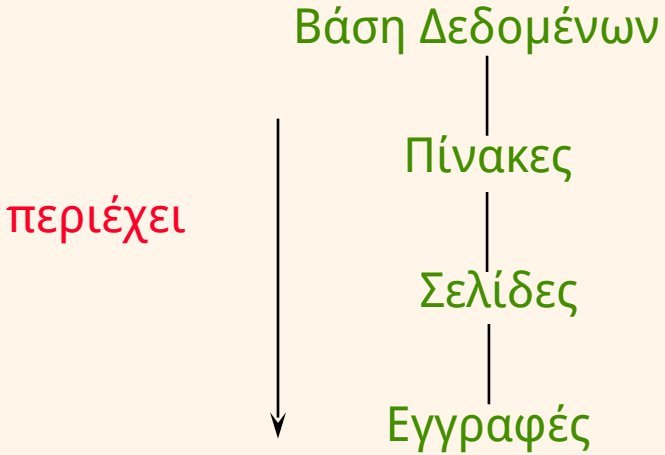
T1: S(A), R(A), S(B)
T2: X(B),W(B) X(C)
T3: S(C), R(C) X(A)
T4: X(B)





Κλειδαριές Πολλαπλής-Κλιμάκωσης

- Δεν είναι εύκολο να αποφασιστεί η κλίμακα στην οποία πρέπει να γίνει το κλείδωμα (εγγραφές vs. σελίδες vs. πίνακες).
- Δεν θα έπρεπε να αποφασίζουμε εμείς!
- Υπάρχει ιεραρχία (εμφώλευση) στις δομές αποθήκευσης των δεδομένων:



Λύση: νέοι τύποι κλειδώματος, πρωτόκολλο

- Επιτρέπουμε στις συναλλαγές να κλειδώνουν σε κάθε επίπεδο, αλλά με ένα ειδικό πρωτόκολλο και με τη χρήση νέων κλειδαριών "πρόθεσης":
- Πριν κλειδώσει ένα αντικείμενο, η συναλλαγή πρέπει να αποκτήσει "κλειδαριές πρόθεσης" σε όλους τους προγόνους του.
- Το ξεκλείδωμα γίνεται από την ειδική στη γενική κλειδαριά (δηλ., από κάτω προς τα επάνω).
- Τύπος SIX: Όπως S & IX ταυτόχρονα.

	--	IS	IX	S	X
--	√	√	√	√	√
IS	√	√	√	√	
IX	√	√	√		
S	√	√		√	
X	√				

Πρωτόκολλο Κλειδώματος Πολλαπλής Κλιμάκωσης

- Κάθε συναλλαγή ξεκινά από τη ρίζα της ιεραρχίας.
- Για να αποκτήσει κλειδαριά S ή IS σε ένα κόμβο, πρέπει να διατηρεί IS ή IX στο γονικό κόμβο.
  - Τί γίνεται αν διατηρεί SIX στο γονιό; S στο γονιό;
- Για να αποκτήσει X ή IX ή SIX σε ένα κόμβο, πρέπει να διατηρεί IX ή SIX στο γονικό κόμβο.
- Πρέπει να αποδεσμεύει τις κλειδαριές από κάτω προς τα επάνω.

Το Πρωτόκολλο είναι σωστό επειδή είναι ισοδύναμο με την απευθείας απόκτηση κλειδαριών στα φύλλα της ιεραρχίας.

Παραδείγματα

- Η T1 σαρώνει τον R, και ενημερώνει μερικές εγγραφές:
  - Η T1 αποκτά κλειδαριά SIX στον R, μετά λαμβάνει διαδοχικές κλειδαριές S σε εγγραφές του R, και περιστασιακά αναβαθμίζει κάποια από αυτές σε X.
- Η T2 χρησιμοποιεί ένα ευρετήριο για να διαβάσει μόνο τμήμα του R:
  - Η T2 αποκτά κλειδαριά IS στον R, και μετά διαδοχικές κλειδαριές S σε εγγραφές του R.
- Η T3 διαβάζει ολόκληρο τον R:
  - Η T3 λαμβάνει κλειδαριά S στον R.
  - ή, θα μπορούσε να συμπεριφερθεί όπως η T2 - μπορεί να χρησιμοποιήσει κλιμάκωση κλειδώματος για να αποφασίσει.

	--	IS	IX	S	X
--	√	√	√	√	√
IS	√	√	√	√	
IX	√	√	√		
S	√	√		√	
X	√				

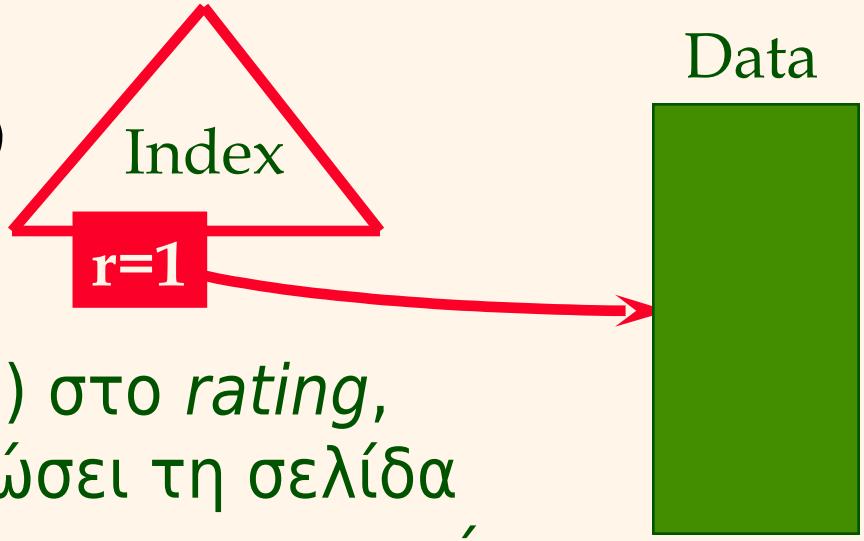
Δυναμικές Βάσεις Δεδομένων

- Αν χαλαρώσουμε την υπόθεση ότι η Βάση Δεδομένων είναι μια σταθερή συλλογή αντικειμένων, ακόμα και το Αυστηρό 2PL δεν θα εγγυηθεί τη σειριοποιησιμότητα:
  - Η T1 κλειδώνει όλες τις σελίδες που περιέχουν εγγραφές ναυτών με *rating* = 1, και βρίσκει τον γηραιότερο ναύτη (έστω, *age* = 71).
  - Μετά, η T2 εισάγει ένα νέο ναύτη με *rating* = 1, και *age* = 96.
  - Επίσης, η T2 διαγράφει τον γηραιότερο ναύτη με rating = 2 (και έστω, *age* = 80), και ολοκληρώνει.
  - Τώρα, η T1 κλειδώνει όλες τις σελίδες που περιέχουν εγγραφές ναυτών με *rating* = 2, και βρίσκει τον γηραιότερο (έστω, *age* = 63).
- Δεν υπάρχει συνεπής κατάσταση της Βάσης Δεδομένων όπου η T1 να είναι “ορθή”!

Το Πρόβλημα (του Φαντάσματος)

- Η T1 εμμέσως υποθέτει ότι έχει κλειδώσει το σύνολο των εγγραφών ναυτών με *rating* = 1.
  - Η υπόθεση ισχύει μόνο αν δεν εισαχθούν νέες εγγραφές ναυτών όσο εκτελείται η T1!
  - Χρειάζεται κάποιος μηχανισμός για την επιβολή αυτής της υπόθεσης. (Κλείδωμα ευρετηρίου και κλείδωμα κατηγορήματος.)
- Το παράδειγμα δείχνει ότι η συγκρουσιακή σειριοποιησιμότητα εγγυάται τη σειριοποιησιμότητα μόνο αν το σύνολο των αντικειμένων είναι σταθερό!

Κλείδωμα Ευρετηρίου



- Αν υπάρχει ένα πυκνό ευρετήριο Εναλλακτικής (2) στο *rating*, η T1 θα μπορούσε να κλειδώσει τη σελίδα του ευρετηρίου που περιέχει τις καταχωρίσεις δεδομένων με *rating* = 1.
  - Αν δεν υπάρχουν εγγραφές με *rating* = 1, η T1 πρέπει να κλειδώσει τη σελίδα ευρετηρίου που θα *βρισκόταν* μια τέτοια καταχώριση δεδομένων, αν υπήρχε!
- Αν δεν υπάρχει κατάλληλο ευρετήριο, η T1 πρέπει να κλειδώσει όλες τις σελίδες, καθώς και το αρχείο/πίνακα ώστε να αποτραπεί η προσθήκη νέων σελίδων, και να διασφαλιστεί ότι δεν θα εισαχθούν νέες εγγραφές με *rating* = 1.

Κλείδωμα Κατηγορήματος (Predicate)

- Χορήγηση κλειδαριών σε όλες τις εγγραφές που ικανοποιούν κάποιο λογικό κατηγόρημα, π.χ., *age > 2*salary*.
- Το κλείδωμα ευρετηρίου είναι μια ειδική περίπτωση κλειδώματος κατηγορήματος, όπου ένα ευρετήριο υποστηρίζει την αποδοτική υλοποίηση της κλειδαριάς κατηγορήματος.
  - Ποιο είναι το κατηγόρημα στο παράδειγμα με τους ναύτες;
- Γενικά, το κλείδωμα κατηγορήματος έχει υψηλό κόστος σε κλειδαριές.

Κλειδώνοντας B+Δέντρα

- Πώς μπορούμε να κλειδώσουμε αποτελεσματικά έναν κόμβο φύλλο;
  - Παρεμπιπτόντως, μην το συγχέετε με το κλείδωμα πολλαπλής κλιμάκωσης!
- Μια λύση: Αγνόησε τη δεντρική δομή, και απλά κλείδωνε τις σελίδες κατά τη διάσχιση του δέντρου χρησιμοποιώντας 2PL.
- Αυτό έχει απαράδεκτη απόδοση!
  - Η ρίζα (και πολλοί κόμβοι των υψηλότερων επιπέδων) γίνονται σημεία συμφόρησης επειδή κάθε πρόσβαση στο δέντρο ξεκινά από τη ρίζα.

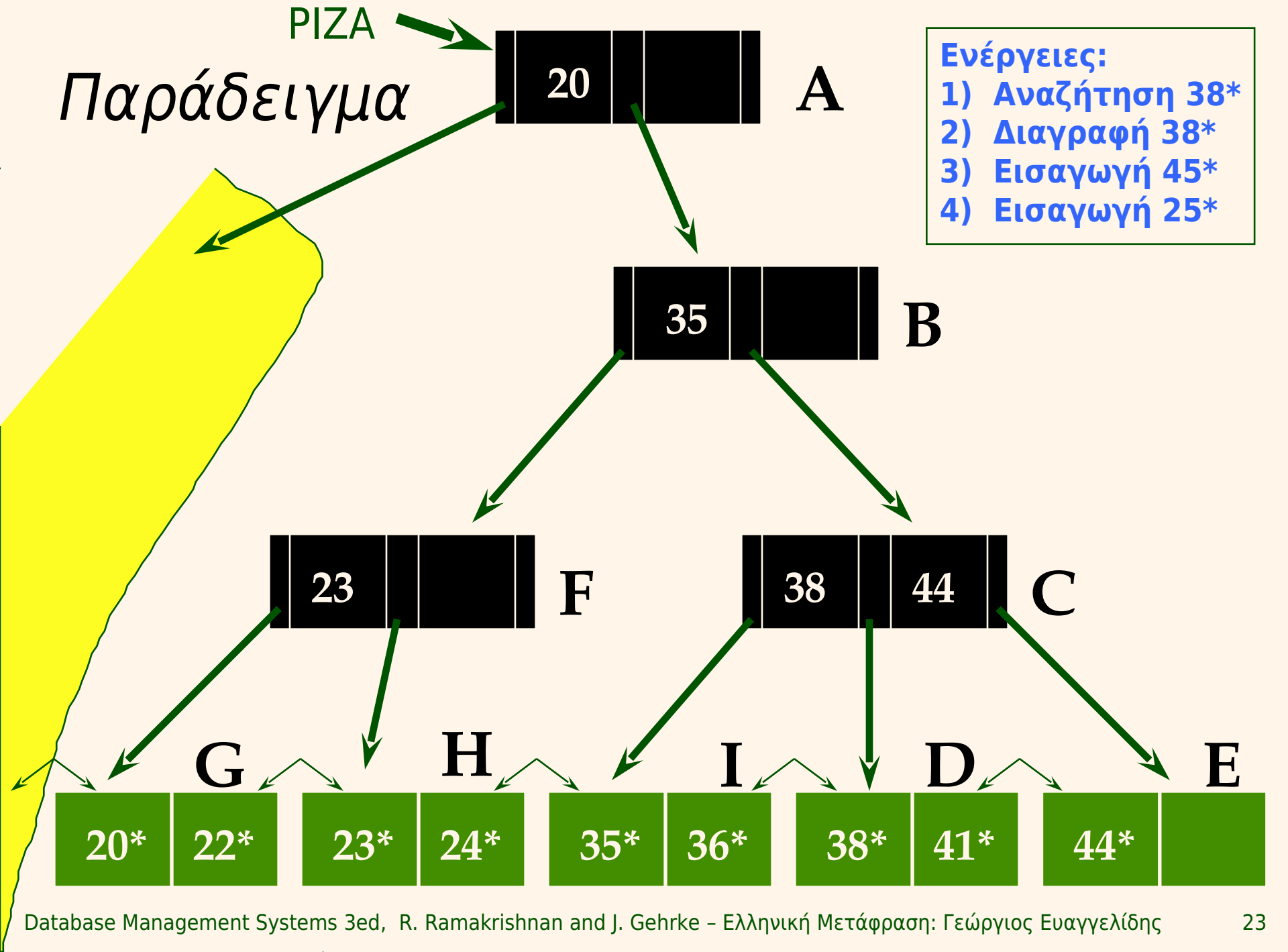
Δυο χρήσιμες παρατηρήσεις

- Τα υψηλότερα επίπεδα του δέντρου απλά κατευθύνουν τις αναζητήσεις προς τα φύλλα.
- Στις εισαγωγές, ένας κόμβος στη διαδρομή από τη ρίζα προς το τροποποιημένο φύλλο πρέπει να κλειδωθεί (φυσικά με κλειδαριά τύπου X), μόνο αν η διάσπαση μπορεί να διαδοθεί ως αυτόν από το τροποποιημένο φύλλο. (Το ίδιο ισχύει και στην περίπτωση των διαγραφών.)
- Μπορούμε να εκμεταλλευτούμε τις παρατηρήσεις αυτές για να σχεδιάσουμε αποτελεσματικά πρωτόκολλα κλειδώματος που εγγυώνται τη σειριοποιησιμότητα *ακόμα και αν παραβιάζουν το 2PL.*

Ένας απλός αλγόριθμος Κλειδώματος Δέντρου

- Αναζήτηση: Ξεκίνα από τη ρίζα και προχώρα προς τα κάτω – διαδοχικά κλείδωνε με S κλειδαριά ένα παιδί και μετά ξεκλείδωνε το γονιό του.
- Εισαγωγή/Διαγραφή: Ξεκίνα από τη ρίζα και προχώρα προς τα κάτω, αποκτώντας κλειδαριές X. Αφού κλειδωθεί ένα παιδί, έλεγξε αν είναι ασφαλές:
  - Αν ναι, αποδέσμευσε όλες τις κλειδαριές των προγόνων του.
- Ασφαλής κόμβος: Ένας κόμβος που αν τροποποιηθεί δεν προκαλεί μετάδοση των αλλαγών προς τα πάνω.
  - Εισαγωγές: Ο κόμβος δεν είναι γεμάτος.
  - Διαγραφές: Ο κόμβος δεν είναι ακριβώς 50% γεμάτος.

Παράδειγμα

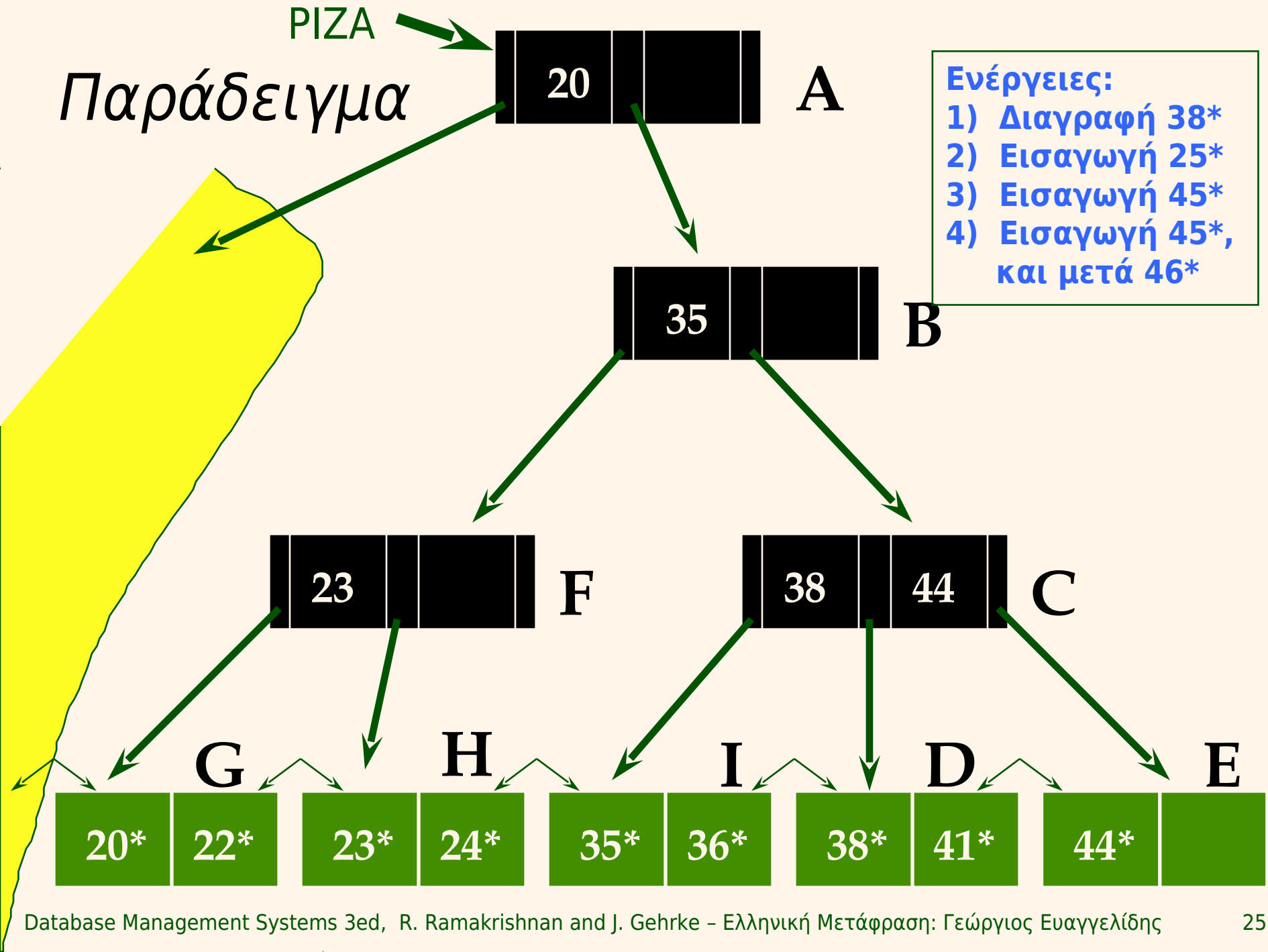


Ενέργειες:
1) **Αναζήτηση 38***
2) **Διαγραφή 38***
3) **Εισαγωγή 45***
4) **Εισαγωγή 25***

Ένας καλύτερος αλγόριθμος κλειδώματος δέντρου (Bayer-Schkolnick)

- Αναζήτηση: Όπως πριν.
- Εισαγωγή/Διαγραφή:
  - Απόκτησε κλειδαριές όπως στην αναζήτηση, και όταν φτάσεις στο φύλλο πάρε κλειδαριά X.
  - Αν το φύλλο δεν είναι ασφαλές, αποδέσμευσε όλες τις κλειδαριές, και επανεκκίνησε τη συναλλαγή χρησιμοποιώντας το προηγούμενο πρωτόκολλο.
- Ποντάρει στο ότι μόνο το φύλλο θα τροποποιηθεί. Αλλιώς, χαραμίζονται όλες οι κλειδαριές S του πρώτου περάσματος. Πρακτικά, είναι καλύτερος από τον προηγούμενο αλγόριθμο.

Παράδειγμα



Ενέργειες:
1) **Διαγραφή 38***
2) **Εισαγωγή 25***
3) **Εισαγωγή 45***
4) **Εισαγωγή 45*, και μετά 46***

Υποστήριξη Συναλλαγών στην SQL-92

- Κάθε συναλλαγή έχει μια μέθοδο πρόσβασης, ένα μέγεθος διαγνωστικών, και ένα επίπεδο απομόνωσης.

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Problem
Read Uncommitted	Maybe	Maybe	Maybe
Read Committed	No	Maybe	Maybe
Repeatable Reads	No	No	Maybe
Serializable	No	No	No