


Αποθήκευση Δεδομένων

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά

1



Εισαγωγή


Στο μέρος Α' του μαθήματος, είδαμε το σχεδιασμό και υλοποίηση μιας βάσης δεδομένων χρησιμοποιώντας ένα ΣΔΒΔ

ΜΕΡΟΣ Β':
Το «εσωτερικό» ενός ΣΔΒΔ

- αποθήκευση δεδομένων
- ευρετήρια
- υπολογισμός ερωτήσεων

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά

2



Αποθηκευτικές Μονάδες

Η βάση δεδομένων θα πρέπει να αποθηκευτεί σε κάποιο αποθηκευτικό μέσο

Ιεραρχία αποθήκευσης


πρωτεύουσα αποθήκευση (primary storage)

κύρια μνήμη (main memory) - κρυφή μνήμη (cache)

- άμεση προσπέλαση από την κύρια ΚΜΕ (CPU)
- γρήγορη προσπέλαση
- περιορισμένη χωρητικότητα αποθήκευσης

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά

3




Αποθηκευτικές Μονάδες

Δευτερεύουσα αποθήκευση
(μαγνητικοί δίσκοι, ταινίες, δισκέτες, κλπ)

- για την επεξεργασία των δεδομένων απαιτείται η μεταφορά των δεδομένων στην πρωτεύουσα αποθήκευση
- πιο αργή προσπέλαση
- μεγάλη χωρητικότητα
- μικρότερο κόστος (για την ίδια ποσότητα χώρου η κύρια μνήμη 100 φορές ακριβότερη από τη δευτερεύουσα)

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά

4



Αποθηκευτικές Μονάδες

Οι περισσότερες βάσεις δεδομένων αποθηκεύονται σε δευτερεύουσες αποθηκευτικές μονάδες κυρίως σε δίσκους


- πολύ μεγάλες => μεγάλο κόστος
- μόνιμη αποθήκευση (nonvolatile storage)

Μαγνητικές ταινίες για

- τήρηση εφεδρικών αντιγράφων
- αρχειοθέτηση (archiving) (δεδομένα που θέλουμε να κρατήσουμε για πολύ καιρό αλλά η προσπέλαση τους είναι σπάνια)

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά


5



Μαγνητικοί Δίσκοι

Μαγνητικοί Δίσκοι

- Μαγνητισμός μιας περιοχής του δίσκου κατά ορισμένο τρόπο ώστε 1 ή 0
- Χωρητικότητα (capacity) σε Kbyte - Mbyte - Gbyte
- Μαγνητικό υλικό σε σχήμα κυκλικού δίσκου



- Απλής και διπλής όψης

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιο Πλουρά

6

Μαγνητικοί Δίσκοι

Σε πακέτα δίσκων

Οι πληροφορίες σε ομόκεντρους κύκλους διαφορετικής διαμέτρου: **άτρακτοι track** (συνήθως κάθε άτρακτος την ίδια ποσότητα πληροφορίας)

Ομόκεντροι κύκλοι σε διαφορετικές επιφάνειες: **κύλινδρος (cylinder)**

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 7

Μαγνητικοί Δίσκοι

Block (μονάδα μεταφοράς)

Τομέας (sector)

Κάθε άτρακτος χωρίζεται σε τόξα που ονομάζονται **τομέες (sectors)** και είναι χαρακτηριστικό του κάθε δίσκου και δε μπορεί να τροποιηθεί

Το μέγεθος ενός block τίθεται κατά την αρχικοποίηση του δίσκου και είναι κάποιο πολλαπλάσιο του τομέα

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 8

Μαγνητικοί Δίσκοι

Κεφαλή ανάγνωσης/εγγραφής

Άξονας περιστροφής

ΚΤΕΝΙ

βραχίονας

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 9

Μαγνητικοί Δίσκοι

χρόνος εντοπισμού (seek time) Τοποθέτηση κεφαλής στη σωστή άτρακτο

χρόνος περιστροφής (rotational delay ή latency) Ώσπου η αρχή του σωστού block να βρεθεί κάτω από την κεφαλή

χρόνος μεταφοράς block (block transfer time) χρόνος μεταφοράς δεδομένων από το δίσκο στη μνήμη

Χρόνος προσπέλασης = χρόνος εντοπισμού + χρόνος περιστροφής + χρόνος μεταφοράς

Μεταφορά αρκετών γειτονικών block

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 10

Μαγνητικοί Δίσκοι

Παράδειγμα IBM Deskstar 14GFX

Χωρητικότητα: 14.4 GB

(μέσος) Χρόνος Εντοπισμού: 9.1 msec (2.2 για γειτονικά - 15.5 μέγιστο)

(μέσος) Χρόνος Περιστροφής: 4.17 msec

5 διπλής όψης κυκλικούς δίσκους - 7,200 περιστροφές το λεπτό

Χρόνος Μεταφοράς 13MB ανά sec

Χρόνος προσπέλασης από το δίσκο ~ 10 msec ενώ για θέσης μνήμης 60 nanosecond

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 11

Μαγνητικοί Δίσκοι

Συνήθως μόνο μία κεφαλή τη φορά

Disk controller

- λειτουργίες εγγραφής/ανάγνωσης
- υπολογισμός αθροίσματος ελέγχου (checksum)

Δίαυλος Συστήματος

SCSI

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλουράς 12

Συμπεράσματα

1. Τα δεδομένα πρέπει να βρίσκονται στη μνήμη
 2. Η μονάδα μεταφοράς από το δίσκο στη μνήμη είναι ένα block . Το διάβασμα ή γράψιμο ενός block ονομάζεται λειτουργία Εισόδου/Εξόδου (Input/Output - I/O)
 3. Ο χρόνος προσπέλασης (εγγραφής ή ανάγνωσης) ενός block διαφέρει και εξαρτάται από τη θέση του block
- χρόνος προσπέλασης = χρόνος εντοπισμού + χρόνου περιστροφής + χρόνος μεταφοράς

Μαγνητικές Ταινίες

- Δίσκοι *τυχαίας προσπέλασης* (random access)
- Ταινίες *σειριακής προσπέλασης* (serial access) για να διαβάσουμε το *n*-οστό block πρέπει να ξεκινήσουμε από την αρχή και να διαβάσουμε και τα *n-1* blocks

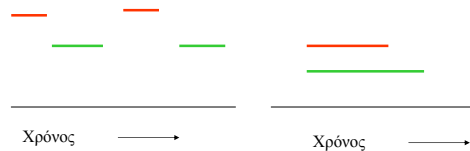
Μεταφορά block σε ενδιάμεση μνήμη

Ενώ γίνεται η μεταφορά των δεδομένων από την δευτερεύουσα στην κύρια μνήμη - παράλληλα και ανεξάρτητα η ΚΜΕ μπορεί να επεξεργάζεται δεδομένα

Ένας ανεξάρτητος επεξεργαστής Εισόδου/Εξόδου ή πολλαπλοί επεξεργαστές

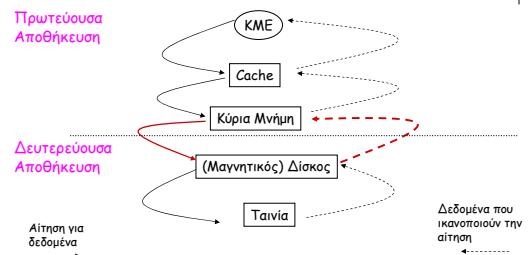
Συνδρομικά (concurrently) και ταυτόχρονα (simultaneously)

Συνδρομικά και εναλλασσόμενα (interleaved)



Χρήση διπλής ενδιάμεσης μνήμης

RAID: πλεονάζουσες συστοιχίες ανεξάρτητων δίσκων (καταμερισμός δεδομένων και πλεονασμός)



Οργάνωση Αρχείων

- Τα δεδομένα συνήθως αποθηκεύονται σε αρχεία
- Η μεταφορά δεδομένων από το δίσκο στη μνήμη και από τη μνήμη στο δίσκο γίνεται σε *μονάδες blocks*

Βασικός στόχος η ελαχιστοποίηση της επικοινωνίας με το δίσκο:
ελαχιστοποίηση του αριθμού των blocks που μεταφέρονται μεταξύ της πρωτεύουσας (κύριας μνήμης, cache - ενδιάμεση μνήμη - buffers, καταχωρητές) και της δευτερεύουσας αποθήκευσης (δίσκος)

Τα δεδομένα συνήθως αποθηκεύονται με τη μορφή *εγγραφών*

Οι εγγραφές συνήθως περιγράφουν οντότητες (σχέσεις) και τα γνωρίσματά τους

Ένα αρχείο είναι λογικά οργανωμένο σε μια ακολουθία από εγγραφές που μπορεί να βρίσκονται αποθηκευμένες σε πολλές σελίδες (pages) - θα θεωρούμε page = block

Blobs

Πώς οργανώνονται τα πεδία μέσα σε μία εγγραφή

Εγγραφές σταθερού και μεταβλητού μήκους

type film = record

branch-name: char(22);

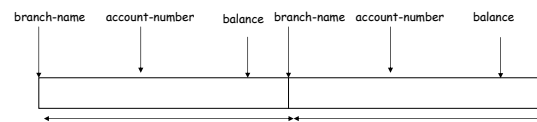
account-number: char(20);

balance: real;

end

Έστω κάθε char 1 byte - real 8 bytes

Κάθε εγγραφή 50 bytes



Γιατί είναι προτιμότερες οι εγγραφές σταθερού μήκους: *εύκολος ο εντοπισμός ενός πεδίου και η διατήρηση πληροφορίας για «άδειες» θέσεις*

Πώς προκύπτουν οι εγγραφές μεταβλητού τύπου.

Στο σχεσιακό μοντέλο κάθε εγγραφή (πλειάδα) μιας σχέσης περιέχει το ίδιο πλήθος πεδίων (αριθμό γνωρισμάτων). Άρα

- Εγγραφές του ίδιου τύπου αλλά έχουν *ένα ή περισσότερα πεδία μεταβλητού μεγέθους*
- *Ανάμεικτο (mixed) αρχείο:* εγγραφές διαφορετικού τύπου



- Αποθήκευση των πεδίων συνεχόμενα, χωρισμένα με διαχωριστές (ειδικούς χαρακτήρες που δεν εμφανίζονται ως δεδομένα)
- Χώρο στην αρχή κάθε εγγραφής - πίνακας ακεραίων $I[j]$ όπου j η μετατόπιση (offset) της j -οστής εγγραφής που κρατά την αρχή του j -οστού πεδίου + τη μετατόπιση του τέλους της εγγραφής
 - απευθείας πρόσβαση σε οποιαδήποτε πεδίο
 - καλό χειρισμό της τιμής null
- Ως εγγραφές σταθερού μήκους, θεωρώντας το μέγιστο μέγεθος για κάθε εγγραφή



Η μονάδα μεταφοράς μεταξύ δίσκου και μνήμης είναι ένα block δίσκου

Έστω εγγραφές σταθερού μήκους

Όταν $B \geq R$ περισσότερες από μια εγγραφή ανά block - κάθε εγγραφή σε ένα μόνο block

Παράγοντας ομαδοποίησης (blocking factor), όταν $B \geq R$
 $bfr = \lfloor (B / R) \rfloor$, όπου B μέγεθος block σε byte
 και R μέγεθος εγγραφής σε bytes

Δηλαδή, πόσες «ολόκληρες» εγγραφές χωρούν σε ένα block



Εκτεινόμενη και μη εκτεινόμενη καταχώρηση εγγραφών

- **Μη εκτεινόμενη** (unspanned) οργάνωση: οι εγγραφές δεν επιτρέπεται να διασχίζουν τα όρια ενός block
 - Αχρησιμοποίητος χώρος: $B - bfr * R$ bytes ανά block
 - Πιο εύκολη η προσπέλαση
- **Εκτεινόμενη** (spanned) οργάνωση: αποθήκευση μέρους μιας εγγραφής σε ένα block και το υπόλοιπο σε ένα άλλο block - δείκτης στο τέλος του πρώτου τμήματος δείχνει στο block που περιέχει το υπόλοιπο



b : Αριθμός blocks για την αποθήκευση ενός αρχείου r εγγραφών:

$$b = \lceil (r/bfr) \rceil$$



Τοποθέτηση block αρχείου στο δίσκο

συνεχόμενη τοποθέτηση (contiguous allocation) τα block του αρχείου τοποθετούνται σε διαδοχικά blocks του δίσκου

συνδεδεμένη τοποθέτηση (linked allocation) κάθε block του αρχείου περιλαμβάνει ένα δείκτη προς το επόμενο block του αρχείου

Εύκολη επέκταση - πιο αργή ανάγνωση όλου του αρχείου

συστάδες διαδοχικών blocks δίσκου (τμήματα (segments) ή επεκτάματα (extents))

ευρετηριοποιημένη τοποθέτηση (indexed allocation)



Επικεφαλίδες αρχείων

Μια επικεφαλίδα ή περιγραφές αρχείου (file header ή file descriptor) περιέχει πληροφορίες σχετικά με ένα αρχείο που είναι απαραίτητες στα προγράμματα που προσπελαίνουν τις εγγραφές του αρχείου

Πληροφορίες για προσδιορισμό διεύθυνσης των blocks αρχείου στο δίσκο

περιγραφές μορφοποίησης εγγραφών

Αποθηκεύεται στο αρχείο, μπορούμε να θεωρούμε ότι «ξέρουμε» σε ποιο block είναι αποθηκευμένη η i-οστή σελίδα του αρχείου



Οργάνωση αρχείων: πως είναι τοποθετημένες οι εγγραφές ενός αρχείου όταν αποθηκεύονται στο δίσκο

- Αρχεία Σωρού
- Ταξινομημένα Αρχεία
- Κατακερματισμένα Αρχεία

B blocks - R εγγραφές ανά block - D εγγραφή/ανάγνωση - C χρόνος επεξεργασίας ανά εγγραφή

D = 15 milliseconds -- C = 100 nanoseconds



Αρχεία Σωρού

Αρχείο Σωρού (heap file ή pile file): Οι εγγραφές τοποθετούνται στο αρχείο με τη σειρά που εισάγονται

Μη ταξινομημένο αρχείο

1. Εισαγωγή

$$2 * D + C$$

2. Αναζήτηση

$$0.5 * B * (D + R * C)$$

B blocks
R εγγραφές ανά block
D χρόνος μεταφοράς block
C χρόνος επεξεργασίας ανά εγγραφή



3. Διαγραφή εγγραφής

Σημάδι διαγραφής

Περιοδική αναδιοργάνωση

Χρόνος Αναζήτησης + (C + D)



4. Τροποποίηση εγγραφής

- εγγραφή μεταβλητού μήκους

5. Σάρωση (scan) Ανάγνωση όλων των εγγραφών

$$B(D+R*C)$$

6. Ανάγνωση όλων των εγγραφών σε διάταξη

Εξωτερική ταξινόμηση συνήθως μια παραλλαγή της ταξινόμησης με συγχώνευση



Ταξινομημένα Αρχεία

Φυσική διάταξη των εγγραφών ενός αρχείου με βάση την τιμή ενός από τα πεδία του το οποίο λέγεται **πεδίο διάταξης** (ordering field)

Διατεταγμένο ή φυσικό αρχείο

- Αν το πεδίο διάταξης είναι και κλειδί τότε λέγεται και **κλειδί διάταξης**



1. Εισαγωγή

- i. Εύρεση της σωστής θέσης της εγγραφής στο αρχείο
 - ii. Μετακίνηση εγγραφών για να κάνουμε χώρο για την εισαγωγή της
- Κατά μέσο όρο μετακίνηση των μισών εγγραφών

Χρόνος αναζήτησης + $2 * (0.5 * B * (D + R * C))$

B blocks
R εγγραφές ανά block
D χρόνος μεταφοράς block
C χρόνος επεξεργασίας ανά εγγραφή



1. Εισαγωγή (συνέχεια)

- Διατήρηση κάποιου αχρησιμοποίητου χώρου ανά block
- Δημιουργία ενός προσωρινού μη διατεταγμένου αρχείου (αρχείο υπερχειλίσισης) - κυρίως αρχείο



2. Αναζήτηση εγγραφής (με επιλογή ισότητας)

αποδοτική αν η συνθήκη αναζήτησης είναι στο πεδίο ταξινόμησης

Έστω B blocks, αναζήτηση της εγγραφής με τιμή K στο πεδίο διάταξης

Σημείωση: Υποθέτουμε ότι οι διευθύνσεις των blocks του αρχείου είναι αποθηκευμένες στην επικεφαλίδα του αρχείου



2. Αναζήτηση εγγραφής (συνέχεια)

lower := 1; upper := B;

while (upper ≥ lower)

 i := (lower + upper) div 2;

 read block i

 if ($K <$ τιμή διάταξης της πρώτης εγγραφής)

 upper := i - 1;

 else if ($K >$ τιμή διάταξης της τελευταίας εγγραφής)

 lower := i + 1;

 else ...

Χρόνος: $\log B * (D + \log R * C)$

Συνθήκη πχ., $<=$

B blocks
R εγγραφές ανά block
D χρόνος μεταφοράς block
C χρόνος επεξεργασίας ανά εγγραφή



3. Διαγραφή εγγραφής

Μετακίνηση εγγραφών

Χρήση σημάδιου διαγραφής

4. Τροποποίηση εγγραφής



5. Ανάγνωση όλων των εγγραφών σε διάταξη

Αποθήκευση (επανάληψη)

1. Τα δεδομένα αποθηκεύονται σε αρχεία στο δίσκο
2. Για να γίνει η επεξεργασία τους πρέπει να μεταφερθούν στη μνήμη
3. Η μονάδα μεταφοράς από το δίσκο στη μνήμη είναι ένα block
4. Ο χρόνος προσπέλασης (εγγραφής ή ανάγνωσης) ενός block διαφέρει και εξαρτάται από τη θέση του block - δε θα το εξετάσουμε στο μάθημα

Οργάνωση Αρχείων (επανάληψη)

Ένα αρχείο είναι λογικά οργανωμένο σε μια ακολουθία από **εγγραφές**
Συνήθως ένα αρχείο ανά (σχήμα) σχέσης και μια εγγραφή αντιστοιχεί σε μια πλειάδα

Μη εκτεινόμενη (unspanned) οργάνωση:

- οι εγγραφές δεν επιτρέπεται να διασχίζουν τα όρια ενός block
- (-) Αχρησιμοποίητος χώρος
- (+) Πιο εύκολη η προσπέλαση

Οργάνωση Αρχείων (επανάληψη)

Έστω B μέγεθος block σε byte και R μέγεθος εγγραφής σε bytes

Παράγοντας μαδδοποίησης (blocking factor), όταν $B \geq R$

$$\text{bfr} = \lfloor (B / R) \rfloor$$

Πόσες εγγραφές χωρούν σε ένα block

b: Αριθμός blocks για την αποθήκευση ενός αρχείου r εγγραφών:

$$b = \lceil (r/\text{bfr}) \rceil$$

Οργάνωση Αρχείων (επανάληψη)

Βασικός στόχος η ελαχιστοποίηση του αριθμού των blocks που μεταφέρονται

Θεωρούμε ότι η πληροφορία για τη θέση στο δίσκο ενός block υπάρχει (π.χ., στην επικεφαλίδα του αρχείου)

Οργάνωση Αρχείων (επανάληψη)

Οργάνωση αρχείων: πως είναι τοποθετημένες οι εγγραφές ενός αρχείου όταν αποθηκεύονται στο δίσκο

Βασικές πράξεις: διάβασμα όλου του αρχείου (scan), εισαγωγή εγγραφής, διαγραφή εγγραφής, αναζήτηση με συνθήκη ισότητας, αναζήτηση με συνθήκη διαστήματος τιμών

- Αρχεία Σωρού (δεν υπάρχει διάταξη)
- Ταξινομημένα Αρχεία (διάταξη με βάση κάποιο πεδίο διάταξης)

→ • Αρχεία Κατακερματισμού

Αρχεία Κατακερματισμού

Πεδίο ή κλειδί κατακερματισμού

Στόχος

h: συνάρτηση κατακερματισμού

$$h(k) = i$$

Διεύθυνση block του δίσκου που είναι αποθηκευμένη

Τιμή του πεδίου κατακερματισμού

Εσωτερικός Κατακερματισμός

Εσωτερικός Κατακερματισμός (τα δεδομένα είναι στη μνήμη, όπως στις δομές δεδομένων)

Πίνακας κατακερματισμού με M θέσεις - κάδους (buckets)

h : συνάρτηση κατακερματισμού

$$h(k) = i \leftarrow \text{Σε ποιο κάδο - τιμή από 0 έως } M-1$$

Πεδίο αναζήτησης -
Πεδίο κατακερματισμού

• Συνήθισμένη συνάρτηση κατακερματισμού:

$$h(k) = k \bmod M$$

Εσωτερικός Κατακερματισμός

• **Σύγκρουση (collision)**: όταν μια νέα εγγραφή κατακερματίζεται σε μια ήδη γεμάτη θέση

• **Καλή συνάρτηση κατακερματισμού**: κατανέμει τις εγγραφές ομοιόμορφα στο χώρο των διευθύνσεων (ελαχιστοποίηση συγκρούσεων και λίγες αχρησιμοποίητες θέσεις)

• **Ευριστικά**:

-- αν r εγγραφές, πρέπει να επιλέξουμε το M ώστε το r/M να είναι μεταξύ του 0.7 και 0.9

-- όταν χρησιμοποιείται η mod τότε είναι καλύτερα το M να είναι πρώτος

Εσωτερικός Κατακερματισμός

Επίλυση Συγκρούσεων

1. **Ανοιχτή Διευθυνσιοδότηση** (open addressing): χρησιμοποίησε την επόμενη κενή θέση
2. **Αλυσιδωτή Σύνδεση** (chaining): για κάθε θέση μια συνδεδεμένη λίστα με εγγραφές υπερχειλίσις
3. **Πολλαπλός Κατακερματισμός** (multiple hashing): εφαρμογή μιας δεύτερης συνάρτησης κατακερματισμού

Εξωτερικός Κατακερματισμός

Εξωτερικός Κατακερματισμός (εφαρμογή σε δεδομένα αποθηκευμένα σε αρχεία)

Κάδος: μια συστάδα από συνεχόμενα blocks του αρχείου

$$h(k) = i \leftarrow \text{Σχετική διεύθυνση του κάδου (ποιος κάδος του αρχείου)}$$

Τιμή του πεδίου κατακερματισμού

π.χ., η εγγραφή με τιμή k στο πεδίο κατακερματισμού βρίσκεται στον i -οστό κάδο

Εξωτερικός Κατακερματισμός

Ένας πίνακας που αποθηκεύεται στην επικεφαλίδα του αρχείου μετατρέπει τον αριθμό κάδου στην αντίστοιχη διεύθυνση block

0	διεύθυνση 1ου block του κάδου στο δίσκο
1	διεύθυνση 1ου block του κάδου στο δίσκο
2	διεύθυνση 1ου block του κάδου στο δίσκο
...	...
$M-1$	διεύθυνση 1ου block του κάδου στο δίσκο

Εξωτερικός Κατακερματισμός

Συγκρούσεις - αλυσιδωτή σύνδεση - εγγραφές υπερχειλίσις ανά κάδο

1. Ανάγνωση όλου του αρχείου (scan)

Έστω ότι διατηρούμε κάθε κάδο γεμάτο κατά 80% άρα ένα αρχείο με μέγεθος B blocks χρειάζεται $1.25 B$ blocks δίσκου

$$1.25 * B * (D + R * C)$$

2. Αναζήτηση

Συνθήκη **ισότητας** και μόνο ένα block ανά κάδο: $D + R * C$

Αν συνθήκη περιοχής: scan!

Κόστος: μεταφορά blocks (I/O)

	Σωρός	Ταξινομημένο	Κατακερματισμένο
Ανάγνωση του αρχείου	B	B	1,25B
Αναζήτηση με συνθήκη ισότητας	0.5 B	logB	1
Αναζήτηση με συνθήκη περιοχής	B	logB + ταιριάσματα	1,25 B
Εισαγωγή	2	αναζήτηση + B	2
Διαγραφή	αναζήτηση + 1	αναζήτηση + B	αναζήτηση + 1

Πρόβλημα:

Έστω M κάδους και r εγγραφές ανά κάδο - το πολύ $M * r$ εγγραφές (αλλιώς μεγάλες αλυσίδες υπερχειλίσις)

Δυναμικός κατακερματισμός

Δυναμικός Εξωτερικός Κατακερματισμός

- Διαδική αναπαράσταση του αποτελέσματος της συνάρτησης κατακερματισμού, δηλαδή ως μια ακολουθία δυαδικών ψηφίων
- Κατανομή εγγραφών με βάση την τιμή των αρχικών ψηφίων

- Το αρχείο ξεκινά με **ένα** μόνο κάδο
- Μόλις γεμίσει ένας κάδος διασπάται σε δύο κάδους με βάση **την τιμή του 1ου δυαδικού ψηφίου** των τιμών κατακερματισμού - δηλαδή οι εγγραφές που το πρώτο ψηφίο της τιμής κατακερματισμού τους είναι 1 τοποθετούνται σε ένα κάδο και οι άλλες (με 0) στον άλλο
- Νέα υπερχειλίση ενός κάδου οδηγεί σε διάσπαση του με βάση το **αμέσως επόμενο δυαδικό ψηφίο** κοκ

Έτσι δημιουργείται μια δυαδική δενδρική δομή που λέγεται **κατάλογος** (dirrectory) ή **ευρετήριο** (index) με δύο ειδών κόμβους

- εσωτερικούς: που καθοδηγούν την αναζήτηση
- εξωτερικούς: που δείχνουν σε ένα κάδο

Αλγόριθμος αναζήτησης

```

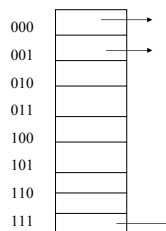
h := τιμή κατακερματισμού
t := ρίζα του δέντρου
i := 1
while (t εσωτερικός κόμβος)
    if (i-οστό bit του h είναι 0)
        t := αριστερά του t
    else t := δεξιά του t
    i := i + 1
    
```

Δυναμικός Εξωτερικός Κατακερματισμός

- Του αποθηκεύεται ο κατάλογος στη μνήμη, εκτός αν είναι πολύ μεγάλος αν στο δίσκο απαιτούνται επιπρόσθετες προσπελάσεις
- Δυναμική επέκταση αλλά **μέγιστος αριθμός επιπέδων** (το πλήθος των δυαδικών ψηφίων της συνάρτησης κατακερματισμού)
- Ισοζύγισι
- Συνένωση κώδων (δυναμική συρρίκνωση)

Επεκτάσιμος Εξωτερικός Κατακερματισμός

Ο **κατάλογος** είναι ένας πίνακας με 2^d διευθύνσεις κώδων (**d: ολικό βάθος του καταλόγου**)

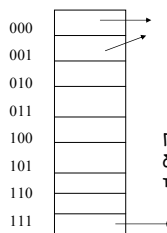


Κώδος για τις εγγραφές με τιμές κατακερματισμού που αρχίζουν από (ή τελειώνουν σε) 000

Τα πρώτα (ή τα τελευταία) **d** ψηφία της τιμής κατακερματισμού χρησιμοποιούνται ως δείκτης στον πίνακα

Επεκτάσιμος Εξωτερικός Κατακερματισμός

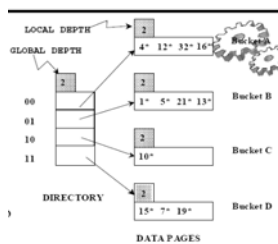
Δε χρειάζεται ένας διαφορετικός κώδος για κάθε μία από τις 2^d θέσεις - μπορεί η θέση του πίνακα να δείχνει στη διεύθυνση του ίδιου κώδου αν αυτές χωράνε σε ένα κώδο



Κώδος για τις εγγραφές με τιμές κατακερματισμού που αρχίζουν από 00

Για κάθε κώδο, **τοπικό βάθος d'** ο αριθμός των δυαδικών ψηφίων στα οποία βασίζεται η χρήση του κώδου

Επεκτάσιμος Εξωτερικός Κατακερματισμός (Παράδειγμα)



Χρήση των τελευταίων bits της δυαδικής αναπαράστασης

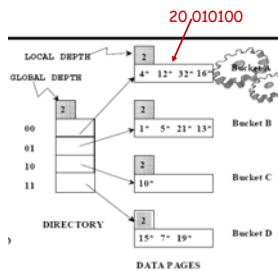
1	000001
4	000100
5	000101
7	000111
10	000110
12	001100
15	001111
16	010000
19	010011
21	010101
32	100000
13	001101

Επεκτάσιμος Εξωτερικός Κατακερματισμός

Η τιμή του **d** μπορεί να αυξάνεται (μέχρι 2^k , κ: αριθμός δυαδικών ψηφίων της τιμής κατακερματισμού) ή να μειώνεται

- **Αύξηση της τιμής του d**
Όταν ένας κώδος με τιμή $d' = d$ υπερχειλίζει
Διπλασιασμός του πίνακα *Δε χρειάζεται rehash (ξανακερματισμό), διασπάμε κάθε κώδο*
- **Μείωση της τιμής του d**
Όταν για όλους τους κώδους $d' < d$
Μείωση του μεγέθους του πίνακα στο μισό

Επεκτάσιμος Εξωτερικός Κατακερματισμός (Παράδειγμα)



1	000001
4	000100
5	000101
7	000111
10	000110
12	001100
15	001111
16	010000
19	010011
21	010101
32	100000
13	001101

Διάσπαση
-> Ολικό βάθος 3

Επεκτάσιμος Εξωτερικός Κατακερματισμός

4 12 32 16 20 -> διάσπαση

1 000001
4 000100
5 000101
7 000111
10 000110
12 001100
15 001111
16 010000
19 010011
21 010101
32 100000
13 001101
20 010100

LOCAL DEPTH GLOBAL DEPTH Bucket A Bucket B Bucket C Bucket D Bucket A2 (split image of Bucket A)

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 67

Γραμμικός Εξωτερικός Κατακερματισμός

Γραμμικός Κατακερματισμός

Θέλουμε να αποφύγουμε τη χρήση καταλόγου

Σημείωση: διατηρούμε λίστες υπερχειλίσης

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 68

Γραμμικός Εξωτερικός Κατακερματισμός

Χρησιμοποιεί μια οικογένεια από συναρτήσεις κατακερματισμού $h_0(k), h_1(k), \dots, h_n(k)$

Κάθε συνάρτηση *διπλάσιους κάδους* από την προηγούμενη:

$h_0(k) = k \bmod M, h_1(k) = k \bmod 2M, h_2(k) = k \bmod 4M, \dots,$
 $h_j(k) = k \bmod 2^j M$

Όταν συμβαίνει η πρώτη υπερχειλίση ενός κάδου, πάμε στην επόμενη συνάρτηση μέχρι να διασπαστούν όλοι οι κάδοι με αυτήν τη συνάρτηση

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 69

Γραμμικός Εξωτερικός Κατακερματισμός

Αρχικά:

Βήμα Διάσπασης (ποια συνάρτηση χρησιμοποιούμε) αρχικά $j = 0$:
Πλήθος Διασπάσεων (στο τρέχον βήμα) αρχικά $n = 0$,

Έστω αρχικά M κάδους αριθμημένους από 0 έως $M - 1$ και *αρχική συνάρτηση κατακερματισμού*

$h_0(k) = k \bmod M$

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 70

Γραμμικός Εξωτερικός Κατακερματισμός

Όταν συμβεί μια υπερχειλίση σε έναν οποιοδήποτε κάδο, **ο κάδος 0** χωρίζεται σε δύο κάδους: τον αρχικό κάδο 0 και ένα νέο κάδο M στο τέλος του αρχείου με βάση την συνάρτηση $h_1(k) = k \bmod 2M$

Βήμα Διάσπασης (ποια συνάρτηση χρησιμοποιούμε) $j = 1$
Πλήθος Διασπάσεων $n = 1$

Συνεχίζουμε γραμμικά, διασπώντας με τη σειρά τους κάδους 1, 2, 3, ... μέχρι να διασπαστούν όλοι οι «παλιοί» κάδοι

για μεταβλητή n («Πλήθος Διασπάσεων») κρατάει ποιος κάδος έχει σειρά για διάσπαση

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 71

Γραμμικός Εξωτερικός Κατακερματισμός

Βήμα διάσπασης (ποια συνάρτηση χρησιμοποιούμε) $j = 1$:
Πλήθος Διασπάσεων $n = m - 1$:

Όταν συμβεί μια υπερχειλίση σε έναν οποιοδήποτε κάδο, **ο κάδος $m - 1$** χωρίζεται σε δύο κάδους: τον αρχικό κάδο $m - 1$ και ένα νέο κάδο $m + k - 1$ στο τέλος του αρχείου με βάση την συνάρτηση $h_1(k) = k \bmod 2M$

Δηλαδή, σε κάθε υπερχειλίση χωρίζουμε όλους τους κάδους με τη σειρά ξεκινώντας από τον πρώτο κάδο

Βάσεις Δεδομένων 2005-2006 Ευαγγέλιος Πλιουρά 72

Αναζήτηση Εγγραφής

Δύο περιπτώσεις ο κώδος στον οποίο είναι (1) έχει ή (2) δεν έχει διασπαστεί

Κρατάμε μια μεταβλητή το πλήθος n των διασπάσεων

Έστω n ο αριθμός διασπάσεων και ότι αναζητούμε το k ,
 βρίσκεται στον κώδο $h_0(k)$
 τότε αν $n \leq h_0(k)$ ο κώδος δεν έχει διασπαστεί
 ενώ αν $n > h_0(k)$ ο κώδος έχει διασπαστεί και εφαρμόζουμε την $h_1(k)$

Συνεχίζουμε ...

Όλοι οι κώδοι έχουν διασπαστεί όταν: $n = M$

Τότε έχουμε $2M$ κώδους

Όταν $n = M$,
 μηδενίζουμε το n , $n = 0$
 και για οποιαδήποτε νέα διάσπαση εφαρμόζουμε την

$$h_2(k) = k \bmod 4M$$

Διασπώντας πάλι τον κώδο $0, 1, \dots$ κ.τ.λ

Γενικά βήμα διάσπασης j ($j = 0, 1, 2, \dots$)

$$h_j(k) = k \bmod 2^j M,$$

και την $h_{j+1}(k)$ για διασπάσεις

Αναζήτηση Εγγραφής (γενικά)

Τι χρειάζεται να ξέρουμε για να βρεθεί ο κώδος της εγγραφής k που ψάχνουμε;

- ποια συνάρτηση χρησιμοποιούμε (δηλαδή, το j)
- σε ποια διάσπαση βρισκόμαστε (δηλαδή το n)

Έστω ότι είμαστε στο βήμα j ,

Τότε θα πρέπει να κοιτάζουμε είτε το $h_j(k)$ αν ο κώδος δεν έχει διασπαστεί

ή το

$h_{j-1}(k)$ αν έχει διασπαστεί

Πως θα ελέγξουμε αν ο κώδος έχει διασπαστεί ή όχι

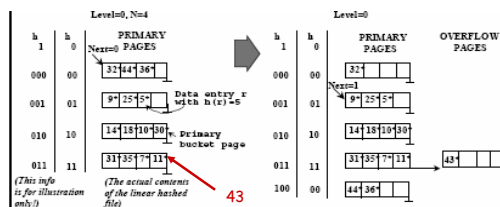
Αλγόριθμος Αναζήτησης

j : βήμα διάσπασης n : πλήθος διασπάσεων στο βήμα j

```

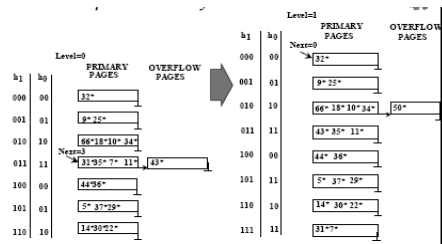
if (n = 0)
    then m := hj(k);
else {
    m := hj(k);
    if (m < n) then m := hj+1(k)
}
    
```

σημαίνει ότι ο κώδος έχει διασπαστεί



βήμα διάσπασης 0 (χρήση h_0) Διασπάμε τον πρώτο κώδο
 Πλήθος διασπάσεων = 0

Γραμμικός Εξωτερικός Κατακερματισμός (παράδειγμα)



Βήμα διάσπασης 0 (χρήση h₀)

Πλήθος διασπάσεων = 0