

ΙΩΑΝΝΗΣ ΜΑΝΩΛΟΠΟΥΛΟΣ
ΚΑΘΗΓΗΤΗΣ
ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ Α.Π.Θ.

ΔΟΜΕΣ ΑΡΧΕΙΩΝ

Εισαγωγή στη φυσική υλοποίηση
βάσεων δεδομένων

Εκδόσεις Art of Text

Ιωάννης Μανωλόπουλος
Τμήμα Πληροφορικής
Αριστοτέλειο Πανεπιστήμιο
Θεσσαλονίκη 54006
τηλ 31-996363
φαξ 31-996360
email manolopo@delab.csd.auth.gr

Εκδόσεις Art of Text
Κίτρους Νικολάου Επισκόπου 4
Θεσσαλονίκη 54635
τηλ 31-218546, 218569
φαξ 31-218569

ISBN 960-312-075-8
Έκδοση εξαντλημένη

Στους γονείς μου
στους δασκάλους μου
στους φοιτητές μου
στην οικογένειά μου

ΤΟ ΝΑΥΑΓΙΟ

Θα μείνω κι εγώ μαζί σας μες στη βάρκα
Ύστερα απ' το φριχτό ναυάγιο και το χαμό
Το πλοίο βουλιάζει τώρα μακριά
(Που πήγαν οι άλλες βάρκες; ποιοί γλυτώσαν;)
Εμείς θα βρούμε κάποτε μία ξέρα
Ένα νησί ερημικό όπως στα βιβλία
Εκεί θα χτίσουμε τα σπίτια μας
Γύρω γύρω απ' τη μεγάλη πλατεία
Και στη μέση μιά εκκλησιά
Θα κρεμάσουμε μέσα τη φωτογραφία
του καπετάνιου μας που χάθηκε - ψηλά ψηλά -
Λίγο πιο χαμηλά του δεύτερου, πιο χαμηλά του τρίτου
Θ' αλλάζουμε τις γυναίκες μας και θα κάνουμε πολλά παιδιά
Κι' ύστερα θα καλαφατίσουμε ένα μεγάλο καράβι
Καινούριο, ολοκαίνουριο και θα το ρίζουμε στη θάλασσα

Θα 'χουμε γεράσει μα θα μας γνωρίζουνε.

Μόνο τα παιδιά μας δε θα μοιάζουνε μ' εμάς.

Μανόλη Αναγνωστάκη
Η Συνέχεια 3 (1962)

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ	6
ΚΑΤΑΛΟΓΟΣ ΜΕΤΑΒΛΗΤΩΝ	10
ΚΕΦΑΛΑΙΟ 0: Πρόλογος	13
ΚΕΦΑΛΑΙΟ 1: Βασικές Έννοιες	21
1.1 Εισαγωγή	22
1.2 Είδη και τύποι αρχείων	24
1.3 Διεργασίες αρχείων	26
1.4 Ασκήσεις	30
ΚΕΦΑΛΑΙΟ 2: Μέσα Αποθήκευσης	31
2.1 Εισαγωγή	32
2.2 Μαγνητικές ταινίες	35
2.3 Μαγνητικοί δίσκοι	42
2.4 Οπτικοί δίσκοι	50
2.5 Άλλες συσκευές αποθήκευσης	54
2.6 Πίνακες δίσκων RAID	57
2.7 Ασκήσεις	60
ΚΕΦΑΛΑΙΟ 3: Διαχείριση Εισόδου/Εξόδου	63
3.1 Εισαγωγή	64
3.2 Τύποι εγγραφών	68
3.3 Τύποι σελίδων	71

Περιεχόμενα	7
3.4 Ομαδοποίηση εγγραφών	73
3.5 Διαχείριση χώρου δίσκου	76
3.6 Διαχείριση απομονωτικής μνήμης	80
3.7 Ασκήσεις	86
ΚΕΦΑΛΑΙΟ 4: Σειριακά Αρχεία	89
4.1 Εισαγωγή	90
4.2 Μη ταξινομημένα σειριακά αρχεία	91
4.3 Ταξινομημένα σειριακά αρχεία	97
4.4 Ασκήσεις	104
ΚΕΦΑΛΑΙΟ 5: Ταξινόμηση με Ταινίες	105
5.1 Εισαγωγή	106
5.2 Φυσική συγχώνευση	108
5.3 Ισοζυγισμένη ταξινόμηση	110
5.4 Πολυφασική ταξινόμηση	112
5.5 Ταξινόμηση καταρράκτη	116
5.6 Ασκήσεις	119
ΚΕΦΑΛΑΙΟ 6: Ταξινόμηση με Δίσκους	121
6.1 Εισαγωγή	122
6.2 Ταξινόμηση με μία συσκευή δίσκων	123
6.3 Ταξινόμηση με δύο συσκευές δίσκων	127
6.4 Συγχώνευση με μία συσκευή δίσκων	130
6.5 Συγχώνευση με δύο συσκευές δίσκων	133
6.6 Άλλες μέθοδοι εξωτερικής συγχώνευσης	138
6.7 Ασκήσεις	139
ΚΕΦΑΛΑΙΟ 7: Σειριακά Αρχεία με Δείκτη	141
7.1 Εισαγωγή	142
7.2 Μέθοδος των καταλόγων κυλίνδρων και επιφανειών	143
7.3 Μέθοδος των σελίδων καταλόγου και δεδομένων	154
7.4 Ασκήσεις	161
ΚΕΦΑΛΑΙΟ 8: Δενδρικές Δομές	163
8.1 Εισαγωγή	164
8.2 B-δένδρα	168
8.3 B*-δένδρα	179
8.4 B ⁺ -δένδρα	183

8.5 Άλλες παραλλαγές των B-δένδρων	196
8.6 Ασκήσεις	199
ΚΕΦΑΛΑΙΟ 9: Τυχαία Στατικά Αρχεία	203
9.1 Εισαγωγή	204
9.2 Συναρτήσεις κατακερματισμού	205
9.3 Διαχείριση συνωνύμων με ανοικτή διεύθυνση	210
9.4 Διαχείριση συνωνύμων με άμεσες αλυσίδες	214
9.5 Διαχείριση συνωνύμων με ψευδοαλυσίδες	221
9.6 Ασκήσεις	224
ΚΕΦΑΛΑΙΟ 10: Τυχαία Δυναμικά Αρχεία	227
10.1 Εισαγωγή	228
10.2 Δυναμικός κατακερματισμός	229
10.3 Επεκτάσιμος κατακερματισμός	233
10.4 Εκθετικός κατακερματισμός με περιορισμένο καταλόγο	239
10.5 Γραμμικός κατακερματισμός	242
10.6 Ασκήσεις	250
ΚΕΦΑΛΑΙΟ 11: Ανάκτηση με Δευτερεύον Κλειδί	253
11.1 Εισαγωγή	254
11.2 Αντεστραμμένα αρχεία	255
11.3 Πολλαπλές λίστες	258
11.4 Συνδυασμένοι κατάλογοι	262
11.5 Πολυδιάστατα δένδρα	266
11.6 Δικτυωτό αρχείο	272
11.7 Ασκήσεις	280
ΚΕΦΑΛΑΙΟ 12: Δομές με Δυαδική Αναπαράσταση	283
12.1 Εισαγωγή	284
12.2 Εξαγωγή υπογραφών	285
12.3 Δένδρα υπογραφών	289
12.4 Αναζήτηση με υπογραφές σελίδων	295
12.5 Κατακερματισμός με υπογραφές	298
12.6 Φίλτρο Bloome	304
12.7 Ασκήσεις	309
ΚΕΦΑΛΑΙΟ 13: Σύγκριση των Δομών	311
13.1 Εισαγωγή	312
13.2 Επιλογή της κατάλληλης οργάνωσης	314

13.3	Επίλογος	324
13.4	Ασκήσεις	324
ΚΕΦΑΛΑΙΟ 14: Νεότερες εξελίξεις		329
14.1	Εισαγωγή	330
14.2	R-δένδρα	333
14.3	Γραμμικά τετραδικά δένδρα περιοχών	341
14.4	Χρονικά διασπώμενο B-δένδρο	346
14.5	Χρονικός κατάλογος	352
14.6	RT-δένδρα	355
14.7	Επικαλυπτόμενα τετραδικά δένδρα περιοχών	359
14.8	Επίλογος	364
ΚΕΦΑΛΑΙΟ 15: Ειδικά Θέματα		367
15.1	Εισαγωγή	368
15.2	Προστασία δεδομένων	368
15.3	Συμπύεση δεδομένων	372
15.4	Συντονισμός ταυτόχρονων προσπελάσεων	379
15.5	Επίλογος	386
ΚΕΦΑΛΑΙΟ 16: Παράρτημα		391
16.1	Εισαγωγή	392
16.2	Εντολές φυσικού επιπέδου σε Pascal	392
16.3	Εντολές φυσικού επιπέδου σε C/C++	399
16.4	Εντολές φυσικού επιπέδου σε C++	403
ΚΕΦΑΛΑΙΟ 17: Βιβλιογραφία		409
17.1	Ξένη βιβλιογραφία	410
17.2	Ελληνική βιβλιογραφία	412
17.3	Αναφορές	413

ΚΑΤΑΛΟΓΟΣ ΜΕΤΑΒΛΗΤΩΝ

Μεταβλητή	Ορισμός	Κεφάλαιο
<i>a</i>	Αριθμός χαρακτηριστικών εγγραφής	11.4
<i>B</i>	Μέγεθος σελίδας (bytes)	2.2
<i>b</i>	Αριθμός σελίδων αρχείου	2.3
<i>b_m</i>	Αριθμός σελίδων στην κύρια περιοχή	4.3
<i>b_o</i>	Αριθμός σελίδων στην περιοχή υπερχείλισης	4.3
<i>bk</i>	Αριθμός κάρδων αρχείου	2.6
<i>bk_m</i>	Αριθμός κάρδων στην κύρια περιοχή	9.2
<i>Bfr</i>	Παράγοντας ομαδοποίησης	2.2
<i>Bkfr</i>	Παράγοντας καδοποίησης	8.4
<i>Buf</i>	Μέγεθος απομονωτικής μνήμης (bytes)	5.2
<i>btt</i>	Χρόνος μεταφοράς σελίδας (ms)	2.3
<i>c</i>	Χρόνος αναζήτησης σε κύριο δείκτη (ms)	2.6
<i>C</i>	Αριθμός σελίδων ανά κύλινδρο	2.6
<i>d</i>	Διανυόμενη απόσταση σε κυλίνδρους	2.3
<i>d</i>	Αριθμός μαρκαρισμένων εγγραφών	7.2
<i>d</i>	Βαθμός B-δένδρου	8.2
<i>d</i>	Βάθος καταλόγου	10.3
<i>d'</i>	Βάθος κάρδου	10.3
<i>D</i>	Πλήθος λέξεων λογικής ομάδας	12.2
<i>Den</i>	Πυκνότητα εγγραφής ταινίας (cpi)	2.2
<i>dt</i>	Χρόνος μεταφοράς κάρδου (ms)	2.6
<i>ebt</i>	Πραγματικός χρόνος μεταφοράς σελίδας (ms)	2.3
<i>F</i>	Μήκος υπογραφής	12.2
<i>h</i>	Ύψος δένδρου	8.2
<i>Ibg, Irg</i>	Μέγεθος κενού μεταξύ των block (inches)	2.2
<i>k</i>	Ελάχιστο πλήθος ζευγών σε κόμβο S-δένδρου	12.3
<i>K</i>	Μέγιστο πλήθος ζευγών σε κόμβο S-δένδρου	12.3
<i>K</i>	Μέγεθος κλειδιού (bytes)	7.2

Μεταβλητή	Ορισμός	Κεφάλαιο
l	Επίπεδο φάσης	4.4
Lf	Παράγοντας φόρτωσης (%)	9.2
m	Τάξη B-δένδρου	8.2
m	Πλήθος άσσεων σε υπογραφές λέξεων	12.2
m	Ελάχιστο πλήθος ζευγών σε κόμβο R-δένδρου	14.2
M	Μέγιστο πλήθος ζευγών σε κόμβο R-δένδρου	14.2
m	Ελάχιστο πλήθος ζευγών σε κόμβο RT-δένδρου	14.6
M	Μέγιστο πλήθος ζευγών σε κόμβο RT-δένδρου	14.6
n	Αριθμός εγγραφών αρχείου	2.2
nod	Αριθμός κόμβων δένδρου	8.2
N	Αριθμός κυλίνδρων δίσκου	2.6
nsg	Αριθμός αρχικών τμημάτων προς ταξινόμηση	6.4
o	Αριθμός εγγραφών υπερχείλισης	7.2
P	Αριθμός δρόμων συγχώνευσης	5.2
P	Μέγεθος δείκτη-pointer (bytes)	7.2
Ppb	Αριθμός σελίδων ανά κάδο	2.3
r	Μέσος χρόνος περιστροφής (ms)	2.3
R	Μέγεθος εγγραφής (bytes)	2.2
S	Μέγεθος τομέα (bytes)	2.3
s	Μέσος χρόνος εντοπισμού (ms)	2.3
seg	Αριθμός σελίδων σε τμήμα	6.4
SI	Μέγεθος καταλόγου (bytes)	7.2
Spd	Ταχύτητα μεταφοράς από συσκευή ταινίας (inches/sec)	2.2
t	Ταχύτητα μεταφοράς από συσκευή δίσκου (bytes/ms)	2.3
t'	Ταχύτητα μεταφοράς από φορμαρισμένο δίσκο (bytes/ms)	2.3
T	Μέγεθος ατράκτου (bytes)	2.3
$T_{s/s}$	Χρόνος εκκίνησης/στάσης ταινίας (ms)	2.2
$T_{προσ}$	Χρόνος προσπέλασης εγγραφής	4.1
$T_{επομ}$	Χρόνος προσπέλασης επόμενης εγγραφής	4.1
$T_{αναν}$	Χρόνος ανανέωσης εγγραφής	4.1
$T_{εισ}$	Χρόνος εισαγωγής εγγραφής	4.1
$T_{διαγ}$	Χρόνος διαγραφής εγγραφής	4.1
$T_{εξαν}$	Χρόνος εξαντλητικής ανάγνωσης αρχείου	4.1
$T_{αναδ}$	Χρόνος αναδιοργάνωσης αρχείου	4.1
U	Παράγοντας χρησιμοποίησης χώρου (%)	2.2
y	Λόγος διακλάδωσης	7.2

Κεφάλαιο 0

ΠΡΟΛΟΓΟΣ

Κεφάλαιο 0

ΠΡΟΛΟΓΟΣ

Ο στόχος του βιβλίου αυτού είναι διττός. Όπως δηλώνει ο υπέρτιτλος (δηλαδή: *Δομές Αρχείων*), ο πρώτος στόχος είναι να καλυφθεί η ύλη που παραδοσιακά ήταν γνωστή με τον όρο *Οργάνωση Αρχείων*. Στο παρελθόν, ο ίδιος ο προγραμματιστής εφαρμογών έπρεπε να σχεδιάσει και να κωδικοποιήσει τις δομές των αρχείων. Για το λόγο αυτό, η Cobol κατ'εξοχήν, και κατά δεύτερο λόγο η PL/1, ήταν οι χρησιμοποιούμενες γλώσσες προγραμματισμού για την ανάπτυξη εμπορικών εφαρμογών με αρχεία. Ωστόσο, από τις αρχές τις δεκαετίας του 80, με την εμπορική διάθεση ισχυρών συστημάτων διαχείρισης βάσεων δεδομένων, η αρμοδιότητα για τη φυσική υλοποίηση των αρχείων πέρασε από τον προγραμματιστή στο σύστημα. Έτσι, ο όρος *Οργάνωση Αρχείων* αποτελεί πλέον μία έννοια με ιστορικό παρελθόν. Από το σημείο αυτό, προκύπτει ο δεύτερος στόχος του βιβλίου, ο οποίος δηλώνεται από τον υπότιτλο (δηλαδή: *Εισαγωγή στη φυσική υλοποίηση βάσεων δεδομένων*). Από τη σκοπιά αυτή, λοιπόν, το βιβλίο περιέχει και υλικό που σχετίζεται με το μάθημα των Βάσεων Δεδομένων, όπως υλικό σχετικά με μεθόδους προσπέλασης (access methods) και τεχνικές που χρησιμοποιούνται στην επεξεργασία ερωτήσεων (query processing). Αναλυτικότερα, το βιβλίο αυτό περιέχει υλικό των μαθημάτων:

- Data Representation, Files and Database Systems, και
- Operating Systems and File Organizations,

όπως προσδιορίζονται στο Computing Curricula 1991 της Association for Computing Machinery (ACM), που είναι η μεγαλύτερη επιστημονική εταιρεία Πληροφορικής σε παγκόσμιο επίπεδο. Ο αναγνώστης παραπέμπεται στη διεύθυνση www.acm.org/education/curr91 για περισσότερες πληροφορίες.

Αλλά ας πάρουμε τα πράγματα από την αρχή. Τα αρχεία είναι δομές που αποθηκεύονται στη δευτερεύουσα μνήμη του υπολογιστή και χρησιμοποιούνται ευρύτατα από το λογισμικό του συστήματος και από το λογισμικό των εφαρμογών. Για παράδειγμα, στο λογισμικό του συστήματος τα αρχεία καλούνται από τους μεταφραστές, το λειτουργικό σύστημα, το συντάκτη κειμένου κλπ., ενώ τα προγράμματα εφαρμογών χρησιμοποιούν τα αρχεία για την αποθήκευση του πηγαίου και του αντικειμένου προγράμματος και των δεδομένων. Πρωταρχικός σκοπός, λοιπόν, του βιβλίου αυτού είναι η παρουσίαση βασικών στοιχείων της δομής, της επεξεργασίας και της επίδοσης των αρχείων δεδομένων που χρησιμοποιούνται από τα προγράμματα εφαρμογών. Όμως, ιδιαίτερη έμφαση θα δοθεί στο σημείο όπου οι δομές αρχείων αφορούν και στηρίζουν τα σπουδαιότερα σήμερα προγράμματα εφαρμογών, που πλέον, σχεδόν αποκλειστικά, χτίζονται με τη βοήθεια σύγχρονων συστημάτων διαχείρισης βάσεων δεδομένων.

Οι εξελίξεις σε σχέση με τα συστήματα διαχείρισης βάσεων δεδομένων δεν σημαίνουν ότι δεν υπάρχει λόγος για περαιτέρω έρευνα και διδασκαλία στην περιοχή αυτή. Ο λόγος είναι ότι καθώς νέες εφαρμογές έρχονται στο προσκήνιο, αποδεικνύεται ότι οι παλαιότερες οργανώσεις αρχείων είναι λιγότερο ή περισσότερο ανεπαρκείς. Για το σκοπό αυτό προτείνονται νέες δομές, που υλοποιούν στο φυσικό επίπεδο τους νέους τύπους δεδομένων, όπως τα χωροταξικά δεδομένα (δηλαδή σημείο, γραμμή, περιοχή, στερεό κλπ.) ή τα πολυμεσικά δεδομένα (δηλαδή κείμενο, ήχος, εικόνα, video κλπ.) ή καλύπτουν νέες ανάγκες, όπως για τη διαχρονική υποστήριξη δεδομένων (όπου δεν επιτρέπονται διαγραφές) καθώς και για την υποστήριξη στατιστικών βάσεων δεδομένων και εφαρμογών on-line αναλυτικής επεξεργασίας OLAP (με τα λεγόμενα πολυδιάστατα δεδομένα) κοκ. Επίσης, σημαντικό είναι να τονισθεί ότι οι νεότερες εκδόσεις των μεγάλων συστημάτων διαχείρισης βάσεων δεδομένων είναι ανοικτές και μπορεί ο προγραμματιστής/αναλυτής να τις επεκτείνει με δικά του modules, ώστε να αντιμετωπισθούν οι νέες εφαρμογές με ειδικές ανάγκες σε τύπους δεδομένων, οργανώσεις αρχείων και ερωτήσεις χρηστών. Για παράδειγμα, το DB2 της IBM δέχεται εξωτερικούς Extenders, το Informix δέχεται τα Datablades, η Oracle δέχεται τα Cartridges κοκ.

Είναι αληθές ότι στην Ελλάδα ο επαγγελματίας πληροφορικός δεν πρόκειται ο ίδιος να προγραμματίσει/κατασκευάσει ένα σύστημα διαχείρισης βάσεων δεδομένων, αλλά μάλλον θα προγραμματίσει μία εφαρμογή με τη βοήθεια ενός τέτοιου συστήματος. Ωστόσο, για να αποφευχθεί η αποσπασματική γνώση του αντικειμένου, ο επαγγελματίας πληροφορικός πρέπει να

μελετήσει συστηματικά τα μέρη ενός τέτοιου συστήματος. Εξ άλλου ένας απλός οδηγός αυτοκινήτου αρκείται να γνωρίζει πως βάζουν βενζίνη στο ρεζερβουάρ, αλλά ένας σωστός οδηγός γνωρίζει πολλά τεχνικά στοιχεία για το αυτοκίνητό του, άσχετα αν δεν πρόκειται να ασκήσει το επάγγελμα του μηχανικού.

Η προαπαιτούμενη υποδομή για την κατανόηση του υλικού του βιβλίου αυτού, είναι ο Προγραμματισμός και οι Δομές Δεδομένων. Εξ άλλου μπορεί να θεωρηθεί ότι το μάθημα της Οργάνωσης Αρχείων είναι μία συνέχεια του μαθήματος των Δομών Δεδομένων. Η προσέγγιση των διαφορών μεθόδων στο βιβλίο αυτό έχει επηρεασθεί από τις αντίστοιχες των βιβλίων του Wiederhold και της Salzberg, και έχει δύο όψεις. Η μία όψη είναι αλγοριθμική και η άλλη όψη είναι αναλυτική. Οι δύο αυτές οπτικές γωνίες αναλύονται στη συνέχεια.

Από μία οπτική γωνία, λοιπόν, οι δομές των αρχείων εξετάζονται αλγοριθμικά, όπως εξ άλλου εξετάζονται οι δομές δεδομένων με τους αντίστοιχους αλγορίθμους στο σχετικό μάθημα. Δηλαδή, όπως ο φοιτητής πρέπει να υλοποιήσει διάφορες δομές δεδομένων μέσα από προγραμματιστικές ασκήσεις, έτσι και εδώ ο αναγνώστης πρέπει να υλοποιήσει μερικές από τις δομές που αναφέρονται στο βιβλίο αυτό. Οι γλώσσες Cobol και PL/1, δεν δίνουν στο χρήστη την ευχαίρεια ούτε να αντιληφθεί τη δομή των αρχείων που χρησιμοποιεί, αλλά ούτε και να υλοποιήσει αποτελεσματικά τις νέες σύγχρονες τεχνικές οργάνωσης αρχείων. Για την καλύτερη κατανόηση της θεωρίας, λοιπόν, ο αναγνώστης παροτρύνεται να υλοποιήσει τις δομές αυτές σε μία γλώσσα δομημένου προγραμματισμού, όπως Pascal ή C/C++. Για το σκοπό αυτό, σε παράρτημα δίνονται οι κυριότερες εντολές διαχείρισης αρχείων στο χαμηλό επίπεδο για τις γλώσσες αυτές.

Από μία άλλη οπτική γωνία αφ' ετέρου, οι δομές των αρχείων εξετάζονται αναλυτικά στην προσπάθεια κοστολόγησης των διαφορών αλγορίθμων (ή διεργασιών επί των αρχείων) με την εύρεση εκφράσεων που να δίνουν τον αριθμό των προσπελάσεων στη δευτερεύουσα μνήμη ή/και τον αντίστοιχο χρόνο. Η μοντελοποίηση των φυσικών φαινομένων με αναλυτικές εκφράσεις είναι συχνά δύσκολη. Απαιτεί περισσότερο σωστή αντίληψη και κρίση παρά ιδιαίτερο μαθηματικό υπόβαθρο. Βασικά, στο βιβλίο αυτό θα χρησιμοποιηθούν μερικές απλές μαθηματικές έννοιες για την ανάλυση της επίδοσης των δομών, οι οποίες είναι οι εξής: ο λογάριθμος που κατά κανόνα είναι με βάση το 2, οι συναρτήσεις 'οροφή' και 'δάπεδο' ενός πραγματικού αριθμού, οι οποίες δίνουν τον αμέσως μεγαλύτερο και μικρότερο ακέραιο του

πραγματικού αριθμού αντίστοιχα, και οι ορισμένες πιθανότητες. Δεν είναι απαραίτητο ο φοιτητής να θυμάται όλες αυτές τις αναλυτικές εκφράσεις, αλλά είναι αναγκαίο να κατανοήσει το μηχανισμό εξαγωγής τους. Όμως σημειώνεται ότι η εξαγωγή τέτοιων αναλυτικών εκφράσεων δεν είναι απλά μία θεωρητική ενασχόληση αλλά μία *sine qua non* πρακτική υποστήριξη προς τη μηχανή επεξεργασίας ερωτήσεων κάθε συστήματος διαχείρισης βάσεων δεδομένων.

Το μάθημα Οργάνωση Αρχείων προηγείται του μαθήματος των Βάσεων Δεδομένων, που πρέπει να επακολουθήσει σε ένα πρόγραμμα σπουδών με έμφαση στην Πληροφορική. Ο σκοπός ενός συστήματος διαχείρισης βάσεων δεδομένων είναι διττός: να παρέχει ένα περιβάλλον φιλικό για την εξυπηρέτηση του χρήστη, το οποίο όμως πρέπει να είναι αποτελεσματικό όσον αφορά στη χρήση των πόρων του. Το υλικό του βιβλίου αυτού θα μπορούσε να θεωρηθεί ως προαπαιτούμενο του μαθήματος των Βάσεων Δεδομένων, και ταυτόχρονα μία πρώιμη εμβάθυνση στο φυσικό επίπεδό τους, το οποίο πρέπει να είναι οργανωμένο αποτελεσματικά. Με λίγα λόγια, σε όλα τα μεγάλα συνέδρια για βάσεις δεδομένων υπάρχουν ειδικές συνεδρίες για μεθόδους προσπέλασης και επεξεργασία ερωτήσεων, που δεν είναι τίποτε άλλο από οργανώσεις αρχείων που συνοδεύονται από τους αντίστοιχους αλγορίθμους επεξεργασίας τους. Έτσι, σκοπός της προσπάθειας αυτής είναι ο καλύτερος φωτισμός των θεμάτων που σχετίζονται με μεθόδους προσπέλασης και προκύπτουν κατά τη φυσική σχεδίαση και υλοποίηση ενός συστήματος διαχείρισης βάσεων δεδομένων. Κατά την ανάπτυξη δίνεται προσοχή ώστε να μη δημιουργούνται πρωθύστερα, και ταυτόχρονα να παρέχεται η άνεση στο μάθημα των Βάσεων Δεδομένων ώστε το ενδιαφέρον να εστιάζεται περισσότερο στο λογικό σχεδιασμό της βάσης, το οποίο είναι ένα από τα σημαντικότερα αντικείμενα για το μάχιμο επιστήμονα της Πληροφορικής στην Ελλάδα.

Τα μαθήματα των Λειτουργικών Συστημάτων και της Αρχιτεκτονικής Υπολογιστών δεν θεωρούνται προαπαιτούμενα για την κατανόηση του υλικού του βιβλίου αυτού, αν και υπάρχουν θέματα στα μαθήματα αυτά που σχετίζονται με τη δική μας οπτική γωνία. Στο βιβλίο αυτό περιέχεται ό,τι θεωρείται απαραίτητο από την ύλη των μαθημάτων αυτών, ώστε να μην είναι απαραίτητο ο αναγνώστης να ανατρέχει σε βιβλία άλλων περιοχών. Ωστόσο, ο αναγνώστης προτρέπεται να ανατρέξει σε ομότιτλα βιβλία, γιατί η παρούσα προσπάθεια δεν βασίζεται στη φιλοσοφία του ενός και μοναδικού συγγράμματος.

Το βιβλίο αυτό (με τις προηγούμενες μορφές του) εξυπηρέτησε διδακτικές ανάγκες του Τμήματος Πληροφορικής του ΑΠΘ, του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΑΠΘ, του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ, του Τμήματος Πληροφορικής του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών, του Τμήματος Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης, του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς, του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων και του Τμήματος Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας. Επίσης υπήρξε διδακτικό βοήθημα σε ιδιωτικές σχολές και σεμινάρια.

Η παρούσα έκδοση στηρίζεται βέβαια στον κορμό των προηγούμενων. Πιστεύεται ότι η έκδοση αυτή είναι πιο σφαιρική στο περιεχόμενο και πιο ολοκληρωμένη στη μορφή από ό,τι οι προηγούμενες. Βασικές προσθήκες έγιναν στο δεύτερο κεφάλαιο σχετικά με τα μέσα αποθήκευσης (που εξελίσσονται με δραματικούς ρυθμούς, όπως τα οπτικά μέσα και τα συστήματα RAID), στο δωδέκατο κεφάλαιο σχετικά με δομές που στηρίζονται σε δυαδική αναπαράσταση (όπως περί εξαγωγής υπογραφών και S-δένδρων) και στο δέκατο τέταρτο κεφάλαιο σχετικά με τις προχωρημένες δομές αρχείων για νέου είδους εφαρμογές βάσεων δεδομένων (όπως χωροταξικές, διαχρονικές, χωροχρονικές). Επίσης, γίνεται αναφορά σε νέου είδους προβλήματα, που ανακύπτουν στις αντικειμενοστραφείς-σχεσιακές και επεκτατές βάσεις δεδομένων, στην on-line αναλυτική επεξεργασία σε πολυδιάστατες βάσεις δεδομένων (OLAP), στην αναζήτηση ημιδομημένων δεδομένων από τον παγκόσμιο ιστό κλπ.

Πιστεύεται ότι το βιβλίο αυτό μπορεί να αναγνωσθεί κάτω από πολλές οπτικές γωνίες. Δηλαδή, μπορεί να αποτελέσει διδακτικό βοήθημα (textbook) σε προπτυχιακό ή/και μεταπτυχιακό επίπεδο, αλλά να αποτελέσει και βιβλίο αναφοράς (reference book) χάρη στην παρατιθέμενη βιβλιογραφία. Οι αναφορές αυτές προέρχονται από τις παγκόσμια εγχυρότερες επιστημονικές εκδόσεις που είναι σχετικές με την περιοχή, όπως τα περιοδικά και τα συνέδρια της ACM και της IEEE καθώς και ευρωπαϊκά περιοδικά όπως τα Information Systems, BIT, The Computer Journal κλπ. Καταβλήθηκε προσπάθεια ο αριθμός τους να μην είναι υπέρμετρος, όμως ο ενδιαφερόμενος αναγνώστης μπορεί να βρει εξαντλητική βιβλιογραφία στην επιτομή του Aoe. Εξάλλου περαιτέρω βιβλιογραφία μπορεί να ανιχνευθεί από το site <http://dblp.uni-trier.de> εκτελώντας search by author και search by title. Ελπίζεται ότι η προσπάθεια αυτή μπορεί να αποτελέσει οδηγό και κίνητρο

για το μεταπτυχιακό φοιτητή, ώστε να εμβαθύνει σε επιμέρους και εξειδικευμένα θέματα.

Οι φοιτητές του Τμήματος Πληροφορικής και του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Αριστοτελείου Πανεπιστήμιου Θεσσαλονίκης βοήθησαν με παρατηρήσεις τους σε βελτιώσεις και διορθώσεις. Επίσης είναι απαραίτητο να σημειωθεί η πολύτιμη βοήθεια και οι παρατηρήσεις των συνεργατών του Εργαστηρίου Τεχνολογίας και Επεξεργασίας Δεδομένων. Πιο συγκεκριμένα, εκφράζονται ευχαριστίες προς το Δρ. Μιχάλη Βασιλακόπουλο, το Δρ. Παναγιώτη Μποζάνη, τους υποψηφίους διδάκτορες Δημήτρη Κατσαρό, Αλέξη Νανόπουλο, Απόστολο Παπαδόπουλο, Αντώνη Σιδηρόπουλο, Θόδωρο Τζουραμάνη και Ελένη Τουσιδου, καθώς και τους προπτυχιακούς φοιτητές Δέσποινα Αναργυρίδου και Γρηγόρη Τσουμάχα. Προφανώς, κάθε λάθος αποτελεί ευθύνη του υπογράφοντος. Αναμένεται ότι σε μία μελλοντική προσπάθεια θα μειωθούν ακόμη περισσότεροι οι όποιες ατέλειες και αυτής της έκδοσης, αν ο αναγνώστης βοηθήσει με παρατηρήσεις του. Ευχαριστίες οφείλονται και στις αντιξοότητες των τελευταίων χρόνων, όπου ο όρος αντιξοότητες απεικονίζει ανθρώπους και καταστάσεις, γιατί μέσα από την υπερκέραση και την υπέρβασή τους επιβεβαιώνονται οι συνειδήσεις και τελεσφορούν οι προσπάθειες. Τέλος, στην οικογένειά μου οφείλεται ευγνωμοσύνη για την ανοχή και την υπομονή (απότοκα της αγάπης τους), που μου έδειξαν τα τελευταία χρόνια.

Η ιδέα της συγγραφής ενός δίτομου έργου για Δομές Δεδομένων, όπου ο πρώτος τόμος θα εξετάζει δομές κύριας μνήμης και ο δεύτερος τόμος δομές που αποθηκεύονται σε περιφερειακές συσκευές ανήκει κατά το ήμισυ στον αείμνηστο Γιάννη Κόλλια καθηγητή του Τομέα Πληροφορικής του Εθνικού Μετσοβείου Πολυτεχνείου. Η προσπάθεια αυτή, λοιπόν, αφιερώνεται πρώτιστα στο δάσκαλο που έφυγε πρόωρα. Κάθε σύστημα έχει τους μηχανισμούς αναπαραγωγής του, και ο δάσκαλος πρέπει να βγάζει δασκάλους. Έτσι, η προσπάθεια αφιερώνεται και στους φοιτητές με τους οποίους αρχίζει πλέον να δημιουργείται η απόσταση της ηλικίας (ή έχει δημιουργηθεί ήδη). Το γεγονός αυτό γεννά την αγωνία να θεωρήσουν δάσκαλο τον υπογράφοντα κάποιοι φοιτητές, από αυτούς που δεν μας μοιάζουν, όπως θα μας επέτρεπε να παραφράσουμε ο ποιητής.

Κεφάλαιο 1

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

1.1 Εισαγωγή

1.2 Είδη και τύποι αρχείων

1.3 Διεργασίες αρχείων

1.4 Ασκήσεις

Κεφάλαιο 1

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

1.1 Εισαγωγή

Στις επιστημονικές εφαρμογές τα δεδομένα είναι απλού τύπου, δηλαδή είναι διάφορα βαθμωτά μεγέθη. Όμως σε εμπορικές εφαρμογές είναι πιο σύνθετα και ονομάζονται **οντότητες** (entities). Για παράδειγμα, οντότητες είναι οι υπάλληλοι μίας εταιρείας, οι συναλλαγές σε ένα τραπεζικό σύστημα, οι φοιτητές στο σύστημα της γραμματείας κλπ. Απαιτεί, λοιπόν, ιδιαίτερη προσοχή ο σωστός καθορισμός των οντοτήτων ενός συστήματος και αποτελεί κομμάτι της ύλης του μαθήματος των Βάσεων Δεδομένων.

Κάθε οντότητα αποτελείται από **χαρακτηριστικά** (attributes) που διευκολύνουν την καλύτερη διάκριση μεταξύ των οντοτήτων. Για παράδειγμα, η οντότητα 'Υπάλληλος' έχει ως χαρακτηριστικά: το ονοματεπώνυμο, τον κωδικό, τη διεύθυνση, το τμήμα εργασίας, την ειδικότητα, το μισθό κλπ. Γενικά, τα χαρακτηριστικά μπορεί να είναι διαφορετικού τύπου (ακέραιος, πραγματικός, συμβολοσειρά) και συνεπώς η αποθήκευσή τους δεν μπορεί να γίνει με πίνακες απλών τύπων αλλά με εγγραφές.

Οι **εγγραφές** (records) είναι δομές που μπορούν να στεγάσουν διαφορετικών τύπων δεδομένα, που περιγράφουν ένα στιγμιότυπο κάποιας οντότητας. Κάθε εγγραφή αποτελείται από **πεδία** (fields). Τα πεδία της εγγραφής μπορεί να είναι αριθμητικά, αλφαβητικά, αλφαριθμητικά και κωδικοποιημένα/συμπιεσμένα. Η κωδικοποίηση και η συμπίεση των δεδομένων είναι ιδιαίτερα μεγάλη περιοχή και θίγεται στο τελευταίο κεφάλαιο των ειδικών θεμάτων. Στο σημείο αυτό απλώς σημειώνεται, ότι με σωστή κωδικοποίηση

επιτυγχάνεται οικονομία χώρου και χρόνου, ευκολία στην ταξινόμηση, τον προγραμματισμό κλπ.

Οι δομές αρχείων (file structures) ή οργανώσεις αρχείων (file organizations) ή φυσικές οργανώσεις (physical organizations) ή, τέλος απλούστερα, τα αρχεία ορίζονται ως μία συλλογή εγγραφών του ίδιου τύπου. Η διαφορά ενός αρχείου για ένα σύστημα διαχείρισης βάσεων δεδομένων ως προς ένα αρχείο ενός λειτουργικού συστήματος είναι ότι στην πρώτη περίπτωση το αρχείο θεωρείται ως μία συλλογή εγγραφών ή σελίδων (δες Κεφάλαια 2 και 3), ενώ στη δεύτερη περίπτωση θεωρείται ως μία συμβολοσειρά χαρακτήρων.

Σε λογικό επίπεδο τα αρχεία μπορεί να θεωρηθούν ως δισδιάστατες δομές. Η οριζόντια διάσταση παριστά τα πεδία μίας εγγραφής, ενώ η κατακόρυφη διάσταση παριστά τον αριθμό των εγγραφών. Ο ορισμός αυτός συμπεριλαμβάνει και τα αρχεία χειμένου. Στο Σχήμα 1.1 παρουσιάζεται το αρχείο ενός συντάκτη χειμένων, όπου κάθε γραμμή παριστά μία γραμμή του χειμένου και έχει δύο χαρακτηριστικά. Τα αρχεία πρέπει να θεωρούνται ως ένα ενιαίο αντικείμενο και όχι ως άθροισμα αντικειμένων, αφού άλλωστε περνούν μεταξύ των υποπρογραμμάτων ως απλές παράμετροι.

Γραμμή	Κείμενο
1	Τρία κόκκινα περιστέρια μέσα στο φώς
2	χαράζοντας τη μοίρα μας μέσα στο φώς με
3	χρώματα και χειρονομίες ανθρώπων που
4	αγαπήσαμε.

Σχήμα 1.1: Αρχείο συντάκτη χειμένου.

Τα πεδία που χρησιμοποιούνται για την αναζήτηση ή την ταξινόμηση των εγγραφών ονομάζονται **κλειδιά** (keys). Το σύνολο των δυνατών τιμών που μπορεί να λάβει ένα κλειδί λέγεται **πεδίο ορισμού** (domain). Το κλειδί που χαρακτηρίζει κατά μοναδικό και αμφομονοσήμαντο τρόπο την κάθε εγγραφή λέγεται **πρωτεύον** (primary) (πχ. ο κωδικός του υπαλλήλου), ενώ τα υπόλοιπα πεδία λέγονται **δευτερεύοντα** (secondary). Είναι δυνατόν να υπάρχουν περισσότερα του ενός κλειδιά που να μπορούν να χρησιμοποιηθούν ως πρωτεύοντα κλειδιά. Τα κλειδιά αυτά ονομάζονται **υποψήφια** (candidate), ενώ αυτά που τελικά δεν επιλέγονται για τέτοια χρήση λέγονται **εναλλακτικά** (alternate). Άλλοτε πάλι ένα πρωτεύον κλειδί είναι ένας συνδυασμός μερικών απλών κλειδιών, οπότε ονομάζεται **σύνθετο** (composite).

Μερικές φορές το πρωτεύον κλειδί δεν έχει πραγματικό νόημα αλλά είναι τεχνητό κατασκευάσμα. Το κλειδί αυτό ονομάζεται **εξωτερικό** (external). Για παράδειγμα, ο **αριθμός ταυτότητας αντικειμένου** (object identification number, oid) είναι ένας αριθμός που δίνεται από μία αντικειμενοστραφή βάση δεδομένων σε εγγραφές αντικειμένων.

Τα αρχεία αποθηκεύονται αποκλειστικά στη δευτερεύουσα μνήμη επειδή:

- το πλήθος των εγγραφών είναι πολύ μεγάλο, οπότε η κύρια μνήμη δεν επαρκεί,
- κάθε φορά μόνο ένα κομμάτι του αρχείου χρησιμοποιείται κατά την επεξεργασία του αρχείου, και τέλος έτσι
- πολλά προγράμματα εφαρμογών μπορούν να προσπελάσουν το ίδιο αρχείο ταυτόχρονα.

1.2 Είδη και τύποι αρχείων

Τα είδη των αρχείων είναι τα εξής:

σειριακά (sequential), όπου οι εγγραφές είναι αποθηκευμένες διαδοχικά στη μνήμη και προσπελάζονται κατά τον ίδιο τρόπο,

τυχαία (random) ή **άμεσα** (direct), όπου η διεύθυνση στη μνήμη προκύπτει με μετασχηματισμό του πρωτεύοντος κλειδιού. Υποπερίπτωση είναι τα **σχετικά** (relative) αρχεία, όπου η θέση κάθε εγγραφής προσδιορίζεται σε σχέση με τη θέση της πρώτης εγγραφής του αρχείου,

σειριακά με δείκτη (index sequential), που από άποψη χρόνου προσπέλασης είναι ένας συμβιβασμός μεταξύ του σειριακού και του τυχαίου αρχείου,

δενδρικά (tree-structured), που χρησιμοποιούνται κατά κόρο στα εμπορικά πακέτα, και τέλος

ανάκτησης με βάση δευτερεύον κλειδί (secondary key retrieval), που συνήθως στηρίζονται στη δομή της συνδεδεμένης λίστας ή στις δομές των δένδρων. Στην κατηγορία αυτή ανήκουν και οι λεγόμενες πολυδιάστατες δομές (όπως το k -d δένδρο, το διχτυωτό αρχείο κλπ), που αποτελούν σύγχρονες και αποτελεσματικές λύσεις.

Στο σημείο αυτό δεν γίνεται ευρύτερη αναφορά, επειδή κάθε είδος αρχείου που αναφέρθηκε θα εξετασθεί εκτενέστερα στα επόμενα κεφάλαια.

Τα δεδομένα μίας συγκεκριμένης εφαρμογής αποθηκεύονται σε πολλά αρχεία, όπου το κάθε αρχείο χαρακτηρίζεται από διαφορετική λειτουργικότητα. Με βάση, λοιπόν, τη λειτουργία τους τα αρχεία κατηγοριοποιούνται ως εξής.

κύριο ή βασικό αρχείο (master file). Περιέχει τα μόνιμα δεδομένα και πρέπει να είναι πάντα ενημερωμένο, όσον αφορά στις εισαγωγές, διαγραφές και ενημερώσεις των εγγραφών. Για παράδειγμα, το κύριο αρχείο σε μία εφαρμογή μισθοδοσίας μεταξύ των άλλων θα περιέχει τα εξής πεδία: κωδικός αριθμός υπαλλήλου, οικογενειακή κατάσταση, επιδόματα, τωρινός μισθός, έσοδα από την αρχή της χρονιάς κλπ.

αρχείο συναλλαγών ή κινήσεως (transaction file). Περιέχει τις εγγραφές του κυρίου αρχείου που έχουν εισαχθεί, διαγραφεί ή ενημερωθεί. Για παράδειγμα, ένα αρχείο συναλλαγών σε μία εφαρμογή μισθοδοσίας περιέχει τις ώρες εργασίας κάθε υπαλλήλου για μία περίοδο. Αυτές οι συναλλαγές στο τέλος της περιόδου εφαρμόζονται στο κύριο αρχείο με σκοπό την έκδοση των επιταγών πληρωμών και την ανανέωση των συνολικών εσόδων κάθε υπαλλήλου από την αρχή του έτους.

βοηθητικό αρχείο ή αρχείο πίνακας (auxiliary file) ή (table file). Περιέχει μία σειρά στατικών δεδομένων, όπου συχνά αναφέρονται τα άλλα αρχεία. Για παράδειγμα, ένα τέτοιο αρχείο μπορεί να περιέχει εγγραφές με τους κωδικούς των προϊόντων, την περιγραφή και τις τιμές τους. Αυτό το αρχείο θα συνοδεύει ένα κύριο αρχείο αποθήκης, ένα κύριο αρχείο παραγγελιών και ένα αρχείο παραγωγής. Έτσι γίνεται οικονομία στο χώρο αποθήκευσης, επειδή αποφεύγεται η επανάληψη πλεοναζόντων (redundant) δεδομένων.

αρχείο αναφορών (report file). Περιέχει πληροφορίες, που έχουν συλλεχθεί και προετοιμασθεί για το χρήστη. Ένα τέτοιο αρχείο είτε εμφανίζεται στην οθόνη, είτε εκτυπώνεται.

αρχείο ελέγχων (control file). Συνήθως είναι ένα μικρό αρχείο και περιέχει πληροφορίες σχετικές, για παράδειγμα, με τον αριθμό των εισαγωγών, διαγραφών και ενημερώσεων στο κύριο αρχείο, στο αρχείο συναλλαγών κλπ. Ο αριθμός των εγγραφών στο νέο κύριο αρχείο

προκύπτει από το άθροισμα του πλήθους των εγγραφών στο παλιό κύριο αρχείο και του πλήθους των εισαγωγών και διαγραφών στο αρχείο ελέγχων.

ιστορικό αρχείο (history file). Αποτελείται από όλα τα προηγούμενα κύρια αρχεία, αρχεία συναλλαγών και αρχεία ελέγχων. Τα αρχεία αυτά συνήθως δεν αποθηκεύονται σε μαγνητικό δίσκο αλλά σε μαγνητική ταινία.

εφεδρικό αρχείο (backup file). Είναι ένα αντίγραφο του κύριου αρχείου, το οποίο χρησιμοποιείται μόνο σε περίπτωση καταστροφής του κύριου αρχείου.

1.3 Διεργασίες αρχείων

Ο χρήστης ενός συστήματος αρχείων σε ένα μεγάλο υπολογιστικό σύστημα ανήκει σε μία από τρεις διακριτές κατηγορίες, δηλαδή μπορεί να είναι **τελικός χρήστης** (end user), **προγραμματιστής εφαρμογών** (application programmer) και **προγραμματιστής συστήματος** (systems programmer). Ο τελικός χρήστης θεωρείται ότι βρίσκεται στο υψηλότερο επίπεδο και του παρέχονται οι δυνατότητες μίας γλώσσας ερωτήσεων (query language), δηλαδή μία μικρή ομάδα εντολών πολύ υψηλού επιπέδου. Ο προγραμματιστής εφαρμογών έχει στη διάθεσή του μία γλώσσα υψηλού επιπέδου που λέγεται **φιλοξενούσα γλώσσα** (host language), επειδή είναι ενισχυμένη με μία ομάδα εντολών που αναφέρονται ως **υπογλώσσα δεδομένων** (data sublanguage). Η υπογλώσσα αυτή υποδιαιρείται στην υπογλώσσα **ορισμού δεδομένων** (data definition) και στην υπογλώσσα **χειρισμού δεδομένων** (data manipulation), όπου η κάθε συνιστώσα έχει προφανές περιεχόμενο. Τέλος, ο προγραμματιστής του συστήματος έχει στη διάθεσή του μία γλώσσα χαμηλού επιπέδου και αρμοδιότητά του είναι η διαχείριση του συστήματος αρχείων.

Ο τελικός χρήστης δεν έχει καμία άποψη για το μέσο αποθήκευσης, το είδος και τον τύπο του αρχείου, το μέγεθος του αρχείου και τη γραμμογράφηση της εγγραφής. Η μόνη απαίτηση του χρήστη αυτού είναι αξιόπιστη πληροφορία και γρήγορη απόκριση. Βέβαια, όλα αυτά τα θέματα είναι σε γνώση του προγραμματιστή εφαρμογών, που όμως με τη σειρά του δεν παρεμβαίνει στη διαχείριση του συστήματος αρχείων. Για το σκοπό αυτό

το πρόγραμμα του τελικού χρήστη μεταφράζεται από ένα πρόγραμμα κωδικοποιημένο στη φιλοξενούσα γλώσσα του προγραμματιστή εφαρμογών, το οποίο με τη σειρά του καλεί το σύστημα αρχείων μέσω των σχετικών μεθόδων προσπέλασης που είναι προγραμματισμένες σε γλώσσα χαμηλού επιπέδου. Το σύστημα επιστρέφει στο πρόγραμμα του προγραμματιστή εφαρμογών τις σχετικές εγγραφές, που στη συνέχεια μεταφέρονται στον τελικό χρήστη. Αυτή η διαδοχή μεταφράσεων προγραμμάτων έχει ως συνέπεια ότι αν τυχόν γίνουν κάποιες αλλαγές στο σύστημα των αρχείων, τότε δεν είναι απαραίτητο ο χρήστης να αλλάξει τα προγράμματά του. Δηλαδή, ως προς τους τελικούς χρήστες υπάρχει **ανεξαρτησία των δεδομένων από τα προγράμματα** (program/data independence), ενώ κάτι τέτοιο βέβαια δεν ισχύει για τις άλλες δύο κατηγορίες χρηστών. Στο επόμενο κεφάλαιο θα δοθούν περισσότερες λεπτομέρειες σχετικά με τα θέματα της αρμοδιότητας του προγραμματιστή του συστήματος.

Οι κυριότερες διεργασίες ή πράξεις επί των αρχείων είναι οι εξής: η ερώτηση, η διατήρηση, η ταξινόμηση και η συγχώνευση. Από τις διεργασίες αυτές μόνο οι δύο πρώτες είναι στη δικαιοδοσία του τελικού χρήστη και του προγραμματιστή εφαρμογών, ενώ οι άλλες δύο διεργασίες είναι της αρμοδιότητας του προγραμματιστή του συστήματος. Ας σημειωθεί ότι οι πράξεις που αναφέρονται στο λογικό επίπεδο των βάσεων δεδομένων, όπως η επιλογή (selection), η προβολή (projection), ο συνδυασμός (join), το καρτεσιανό γινόμενο κλπ. στο φυσικό επίπεδο στηρίζονται στις στοιχειώδεις πράξεις των αρχείων, οι οποίες αναφέρθηκαν.

Ερώτηση (query) στο αρχείο είναι η αναζήτηση εγγραφών, που έχουν συγκεκριμένες τιμές σε ένα ή περισσότερα συγκεκριμένα πεδία. Στη συνέχεια δίνονται οι (περισσότεροι) τύποι ερωτήσεων και αντίστοιχα παραδείγματα θεωρώντας ένα αρχείο μισθοδοσίας με τα πεδία: Κωδικός, Όνομα, Φύλλο, Μισθός, Έτος γέννησης, Τμήμα, Θέση.

απλή ερώτηση με βάση το πρωτεύον κλειδί (simple query based on primary key), ή **σημειακή ερώτηση** (point query) όπως: 'Ποιός υπάλληλος έχει κωδικό 999;',

απλή ερώτηση με βάση το δευτερεύον κλειδί (simple query based on secondary key), ή **πολυσημειακή ερώτηση** (multipoint query), όπως: 'Ποιοί είναι οι υπάλληλοι του Λογιστηρίου;',

ερώτηση διαστήματος (range query), όπως: 'Ποιοί υπάλληλοι γεννήθηκαν πριν το 1960;',

ερώτηση διάταξης (ordering query), όπως: `Να δοθούν τα ονόματα των υπαλλήλων διατεταγμένα κατά φθίνοντα μισθό`,

ερώτηση ταύτισης προθέματος (prefix match query), όπως: `Ποιοί είναι υπάλληλοι, των οποίων το όνομα αρχίζει από Μανω;`,

ακραία ερώτηση (extremal query), όπως: `Ποιοί υπάλληλοι παίρνουν το μεγαλύτερο μισθό;`,

συναρτησιακή ερώτηση (functional query), όπως: `Ποιοί υπάλληλοι παίρνουν μισθό μικρότερο από το μέσο μισθό όλων των υπαλλήλων;`,

ερώτηση ομαδοποίησης (grouping query), όπως: `Ποιός είναι ο μέσος μισθός των υπαλλήλων κάθε τμήματος;`,

ερώτηση επακριβούς ταύτισης (exact match query), όπως: `Ποιοί είναι οι κωδικοί των υπαλλήλων που λέγονται Ιωάννου, είναι άνδρες, παίρνουν μισθό περισσότερο από 300.000 δρχ, γεννήθηκαν πριν το 1950, ανήκουν στο τμήμα πωλήσεων και έχουν θέση προϊσταμένου;`,

ερώτηση μερική ταύτισης (partial match query), όπως: `Ποιοί υπάλληλοι είναι γυναίκες και έχουν θέση διευθυντή;`, είτε τέλος

λογική ερώτηση (boolean query), όπως: `Ποιοί υπάλληλοι είναι γυναίκες είτε/και έχουν θέση διευθυντή;`.

Σημειώνεται ότι η λογική ερώτηση αποτελεί ευρύτερο τύπο από την ερώτηση μερικής ταύτισης και την ερώτηση επακριβούς ταύτισης, επειδή δέχεται και τους τρεις λογικούς τελεστές (δηλαδή, and, or, not). Είναι προφανές ότι οι ερωτήσεις αυτές τίθενται σε σύστημα ενός μόνο αρχείου. Ωστόσο, είναι δυνατόν να τεθούν ερωτήσεις που απαιτούν συνδυασμό δεδομένων από δύο ή περισσότερα αρχεία. Έτσι προκύπτει ένας ακόμη τύπος:

ερώτηση σύνδεσης (join query), όπως: `Ποιοί υπάλληλοι παίρνουν μεγαλύτερο μισθό από τους διευθυντές τους;`.

Επίσης, σημειώνεται ότι η προηγούμενη ερώτηση χρησιμοποιεί ένα μόνο αρχείο αλλά συνδυάζει δύο αντίγραφα του και είναι ιδιαίτερα χρονοβόρα. Στο βιβλίο αυτό δεν θα εξετασθούν συστήματα πολλών αρχείων, όμως σε επόμενο κεφάλαιο θα γίνει αναφορά και σε άλλου τύπου ερωτήσεις. Όλες οι προηγούμενες ερωτήσεις υποστηρίζονται άνετα (δηλαδή, με φιλικό τρόπο) από γλώσσες ερωταπαντήσεων, που παρέχονται σε όλα τα συστήματα

διαχείρισης βάσεων δεδομένων (όπως, η γλώσσα SQL). Ο προγραμματιστής συστήματος κατά τη φάση του σχεδιασμού πρέπει να προνοήσει και να οργανώσει κατά τον καλύτερο τρόπο τα αρχεία, έτσι ώστε να απαντώνται αποτελεσματικά οι πιο συχνά υποβαλλόμενες ερωτήσεις.

Με τον όρο **διατήρηση** (maintenance) του αρχείου εννοείται η ανανέωση του λόγω **εισαγωγών** (inserts), **διαγραφών** (deletes) ή **ενημερώσεων** (updates) εγγραφών. Το πρόγραμμα διατήρησης του αρχείου εκτελεί σειριακά ή τυχαία τη διατήρηση ανάλογα με την οργάνωση του κύριου αρχείου. Επίσης, ο προγραμματιστής συστήματος δεν πρέπει να ξεχνά ότι πιθανώς σε μία σύνθετη εφαρμογή μία αλλαγή μπορεί να αναφέρεται σε περισσότερα από ένα αρχεία, επειδή υπάρχει **επανάληψη δεδομένων** (data duplication). Σε ένα σύστημα διαχείρισης βάσεων δεδομένων τα πράγματα είναι απλούστερα επειδή το σύστημα, που λειτουργεί ως μεσάζων μεταξύ αρχείων και χρήστη, γνωρίζει τη δομή των αρχείων. Έτσι ο χρήστης δεν ασχολείται με λεπτομέρειες, όπως σε ποιο αρχείο ανήκει ένα συγκεκριμένο πεδίο, τί τύπου είναι το κάθε πεδίο, πώς είναι οργανωμένο το αρχείο και σε ποιο μέρος του δίσκου είναι αποθηκευμένο κλπ.

Η **ταξινόμηση** (sorting) αρχείων με βάση κάποιο πεδίο είναι ένα σημαντικό πρόβλημα επειδή συναντάται συχνά στην πράξη και είναι ιδιαίτερα χρονοβόρο, είτε το αρχείο είναι αποθηκευμένο σε ταινία, είτε σε δίσκο. Επειδή το σύνολο του αρχείου δεν χωρά στην κύρια μνήμη, όσο μεγάλη και αν είναι αυτή, οι μέθοδοι ταξινόμησης αρχείων διαφοροποιούνται από τις γνωστές μεθόδους για την ταξινόμηση των δομών κύριας μνήμης (πχ. ταξινόμηση φουσαλίδας, ταξινόμηση εισαγωγής, ταξινόμηση σωρού κλπ.). Επίσης, σε αντίθεση με τις μεθόδους ταξινόμησης δομών κύριας μνήμης, το κριτήριο κόστους δεν είναι ο αριθμός των συγκρίσεων/ανταλλαγών των κλειδιών, αλλά ο αριθμός των προσπελάσεων στη δευτερεύουσα μνήμη. Αυτό ισχύει γιατί η τάξη μεγέθους του χρονικού κόστους των πράξεων αυτών είναι 1:1000.

Η **συγχώνευση** (merging) είναι μία διαδικασία που γίνεται συχνά σε εφαρμογές αρχείων, όπως για παράδειγμα όταν το αρχείο συναλλαγών συνδυάζεται με το κύριο αρχείο. Αν υποθεθεί ότι και τα δύο αρχεία είναι διατεταγμένα κατά αύξουσα τάξη του πρωτεύοντος κλειδιού, τότε η συγχώνευση επιτυγχάνεται με δύο σαρώσεις/αναγνώσεις των αρχείων και την αντίστοιχη αποθήκευση στο μαγνητικό μέσο.

Οι δομές αρχείων μαζί με τις αντίστοιχες διεργασίες τους αποτελούν αυτό που στην ορολογία των βάσεων δεδομένων αποκαλείται **μέθοδοι προ-**

σπέλασης (access methods). Οι μέθοδοι προσπέλασης και οι τεχνικές επεξεργασίας των ερωτήσεων αποτελούν αναπόσπαστο κομμάτι του φυσικού επιπέδου κάθε εμπορικού συστήματος διαχείρισης βάσεων δεδομένων. Από την άποψη αυτή, το υλικό του βιβλίου αυτού είναι μία πρώτη εισαγωγή στα θέματα φυσικής οργάνωσης των βάσεων δεδομένων, όπως δηλώνει και ο υπότιτλος του βιβλίου.

1.4 Ασκήσεις

<1> Μία εγγραφή αποτελείται από τα εξής πεδία με τα αντίστοιχα μήκη: Όνομα (30 χαρακτήρες), Κωδικός (6), Διεύθυνση οδού (26), Πόλη (16), Ταχυδρ. Κωδικός (5). Ποιό είναι το μέγεθος του αρχείου για ένα αρχείο 15.000 εγγραφών; Ποιά χαρακτηριστικά ή συνδυασμός χαρακτηριστικών θα μπορούσε να αποτελέσει μία καλή εκλογή για πρωτεύον κλειδί;

<2> Συχνά οι εταιρείες διατηρούν ταχυδρομικές λίστες πελατών και συλλέγουν τα στοιχεία τους από διάφορες πηγές. Αυτή η πρακτική οδηγεί σε διπλές και τριπλές αποθηκεύσεις του ίδιου προσώπου με αποτέλεσμα οικονομικό κόστος, αλλά και ενόσχληση του πελάτη. Να βρεθεί ένας αλγόριθμος που να εντοπίζει τις εγγραφές αυτές υιοθετώντας τις εξής παραδοχές:

- οι διπλοεγγραφές έχουν το ίδιο όνομα αλλά με παραλλαγές, όπως Γ.Μανωλόπουλος, Ιωάννης Μανωλόπουλος, Ι.Π.Μανωλόπουλος κλπ.,
- οι ταχυδρομικοί κωδικοί διαφέρουν,
- οι διευθύνσεις οδών μπορεί να είναι διαφορετικού τύπου, όπως Γ.Θεοτοκά 3, Θεοτοκά 3, Γεωργίου Θεοτοκά 3 κλπ., και τέλος,
- η πόλη και ο ταχυδ. κωδικός είναι ίδιοι για τις διπλοεγγραφές.

<3> Ποιά οργάνωση και γιατί είναι πιο ιδανική για κάθε μία από τις εξής εφαρμογές: Γενεαλογικό δένδρο, Τηλεφωνικός κατάλογος, Οργανόγραμμα εταιρείας, Ιατρικό δεδομένα, Σύστημα αεροπορικών κρατήσεων, Αρχεία ασφαλισμένων ΙΚΑ.

Κεφάλαιο 2

ΜΕΣΑ ΑΠΟΘΗΚΕΥΣΗΣ

2.1 Εισαγωγή

2.2 Μαγνητικές ταινίες

2.3 Μαγνητικοί δίσκοι

2.4 Οπτικοί δίσκοι

2.5 Άλλες συσκευές αποθήκευσης

2.6 Πίνακες δίσκων RAID

2.7 Ασκήσεις

Κεφάλαιο 2

ΜΕΣΑ ΑΠΟΘΗΚΕΥΣΗΣ

2.1 Εισαγωγή

Κάθε υπολογιστικό σύστημα και ιδιαίτερα ένα σύστημα διαχείρισης βάσεων δεδομένων χαρακτηρίζεται από μία **ιεραρχία μνήμης** (storage hierarchy) κατά επίπεδα, όπως φαίνεται στο Σχήμα 2.1. Τα επίπεδα αυτά μπορούν να κατηγοριοποιηθούν σε τρεις ομάδες: **πρωτεύουσα** (primary), **δευτερεύουσα** (secondary) και **τριτεύουσα** (tertiary) αποθήκευση (storage). Μία γενική παρατήρηση σχετικά με τις μνήμες της ιεραρχίας αυτής είναι ότι καθώς προχωρούμε από επάνω προς τα κάτω, τόσο οι μνήμες γίνονται αργότερες, μεγαλύτερες και φθηνότερες.



Σχήμα 2.1: Επίπεδα ιεραρχίας μνήμης.

Τα τρία ανώτερα επίπεδα της ιεραρχίας ανήκουν στην κατηγορία της πρωτεύουσας αποθήκευσης. Η **κρυφή μνήμη** (cache) είναι η πλέον γρήγορη και ακριβή μνήμη, αλλά είναι σχετικά μικρή. Το λειτουργικό σύστημα είναι υπεύθυνο για τη διαχείριση της μνήμης αυτής.

Η **κύρια μνήμη** (main memory) χρησιμεύει για την αποθήκευση προγραμμάτων, δεδομένων και πληροφοριών ελέγχου. Την τελευταία δεκαετία τα μεγέθη των κύριων μνημών έχουν μεγαλώσει κατά μερικές εκατοντάδες φορές, ενώ το κόστος τους κινείται αντιστρόφως ανάλογα. Μάλιστα, πριν μερικά χρόνια εθεωρείτο ότι θα ήταν δυνατόν οι βάσεις δεδομένων να είναι αποκλειστικά αποθηκευμένες στην κύρια μνήμη (main memory databases). Όμως αυτή η εικασία δεν επιβεβαιώθηκε γενικά, γιατί πάντοτε υπάρχουν εξαιρετικά μνημοβόρες εφαρμογές (για παράδειγμα, πολυμεσικές εφαρμογές), που δεν υλοποιούνται χρησιμοποιώντας μόνο τις έστω και υποθετικά πολύ μεγάλες κύριες μνήμες. Έτσι είναι απαραίτητο τα δεδομένα να αποθηκεύονται στη δευτερεύουσα ή τριτεύουσα αποθήκευση και να ανεβαίνουν στην κύρια μνήμη για επεξεργασία, όταν ζητηθούν. Ωστόσο, υπάρχουν πολλά εργαστηριακά πρωτότυπα αλλά και βιομηχανικά συστήματα διαχείρισης βάσεων δεδομένων κύριας μνήμης, που χρησιμοποιούνται σε ειδικές εφαρμογές (για παράδειγμα, σε εφαρμογές κινητής τηλεφωνίας).

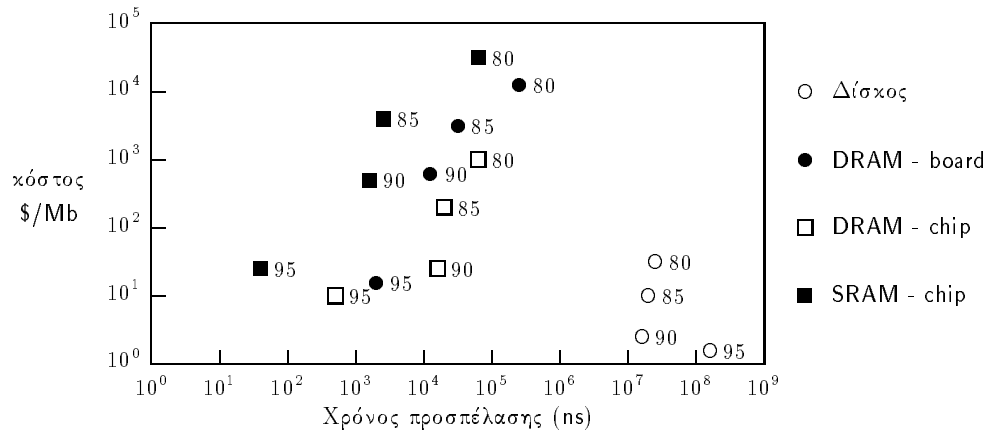
Ένα άλλο επίσης χαρακτηριστικό των κρυφών και των κύριων μνημών είναι η **μεταβλητότητα** (volatility), δηλαδή η ιδιότητά τους να χάνουν τα δεδομένα τους όταν διακοπεί η ηλεκτρική παροχή. Αυτή η ιδιότητα μπορεί να μην είναι δυσάρεστη για λίγα δεδομένα ενός προγράμματος, όμως σίγουρα είναι ανεπιθύμητη σε εφαρμογές όπου τα δεδομένα πρέπει να παραμείνουν μόνιμα αποθηκευμένα για μελλοντική χρήση και ενημέρωση. Σε αντίθεση, στις μνήμες που είναι γνωστές με το αγγλικό ακρωνύμιο EEPROM (από την έκφραση electrically erasable programmable read-only memory), τα δεδομένα δεν χάνονται αν διακοπεί η ηλεκτρική παροχή. Στις μνήμες αυτές η ανάγνωση είναι σχεδόν εξ ίσου γρήγορη όσο και η ανάγνωση από την κύρια μνήμη. Ωστόσο, η αποθήκευση γίνεται μόνο μία φορά ταχύτατα, ενώ στη συνέχεια για την αποθήκευση νέων δεδομένων απαιτείται η προηγούμενη διαγραφή των δεδομένων που ήδη είναι αποθηκευμένα. Ένα άλλο μειονέκτημα είναι ότι η διαδικασία διαγραφής/επανα-αποθήκευσης δεν μπορεί να επαναληφθεί πέρα από κάποιο όριο (για παράδειγμα, από 10.000 μέχρι 1.000.000 φορές). Συνήθως αυτή η μνήμη χρησιμοποιείται σε υπολογιστικά συστήματα που ενσωματώνονται σε άλλες συσκευές, και δεν θα μας αποσχολήσει στη συνέχεια.

Έτσι είναι απαραίτητο τα σύγχρονα συστήματα υπολογιστών και βάσεων δεδομένων να έχουν και δευτερεύουσα μνήμη (secondary memory, backing storage), που σε σχέση με τις προηγούμενες μνήμες μειονεκτεί σε ταχύτητα, αλλά είναι φθηνότερη και μεγαλύτερη. Επίσης στο σημείο αυτό αξίζει να σημειωθεί η δευτερεύουσα μνήμη είναι αναγκαία επειδή κάθε λειτουργικό σύστημα μπορεί να διαχειρισθεί κύριες μνήμες μέχρι ενός ορισμένου μεγέθους. Για παράδειγμα, στις πρώτες του εκδόσεις το Ms-Dos μπορούσε να χειρισθεί διευθύνσεις για προσπέλαση μέχρι 640 Kb, ενώ μόνο 2^{32} bytes μπορούν να προσπελασθούν άμεσα με τις σύγχρονες αρχιτεκτονικές των 32 bits. Τέλος ένα άλλο πλεονέκτημα της δευτερεύουσας μνήμης είναι η δυνατότητά της να χρησιμοποιηθεί ως υπερβατική μνήμη, και πάλι ακριβώς εξ αιτίας της ανεπάρκειας της κύριας μνήμης.

Επί του παρόντος ο μαγνητικός δίσκος (magnetic disc) είναι η προτιμότερη αποθηκευτική λύση και έτσι αποτελεί το κυρίαρχο μέσο αποθήκευσης. Στην πράξη χρησιμοποιείται και η μαγνητική ταινία (magnetic tape), που είναι σειριακό μέσο, καθώς μία εγγραφή προσπελάζεται μόνο αφού προσπελασθούν όλες οι προηγούμενες. Τέλος, στη σύγχρονη οπτική τεχνολογία στηρίζονται οι οπτικοί δίσκοι (optical disks), που διατίθενται εμπορικά σε πολλούς τύπους. Όλες οι προηγούμενες συσκευές/τεχνολογίες αποθήκευσης θα εξετασθούν στη συνέχεια του παρόντος κεφαλαίου.

Για την ιστορία αναφέρεται ότι το πρώτο υλικό που χρησιμοποιήθηκε για μόνιμη αποθήκευση ήταν η διάτρητη κάρτα (punched card) που διατηρεί μέχρι 80 bytes. Ένα αρχείο σε διάτρητες κάρτες μπορεί να αναγνωσθεί πολλές φορές (βέβαια μόνο σειριακά), αλλά κάθε ανανέωση γίνεται με το χέρι. Ένα άλλο μη μαγνητικό μέσο που χρησιμοποιούνταν στα πρώτα βήματα των υπολογιστικών συστημάτων ήταν η διάτρητη χαρτοταινία (paper tape). Τα δύο αυτά μέσα είναι πλέον απαρχαιωμένα και αποτελούν μουσειακά αντικείμενα. Από τα πρώτα επίσης υλικά που χρησιμοποιήθηκαν για δευτερεύουσα μνήμη ήταν οι μαγνητικές ταινίες, που ίσως δεν χρησιμοποιούνται σήμερα στο βαθμό που χρησιμοποιούνταν παλαιότερα, αλλά θεωρείται ότι θα αποκτήσουν μεγαλύτερο ειδικό βάρος στο μέλλον. Επίσης για ιστορικούς και μόνο λόγους πρέπει να αναφερθεί και το μαγνητικό τύμπανο (magnetic drum), που όμως εκτοπίστηκε πλήρως από το μαγνητικό δίσκο. Ο πρώτος μαγνητικός δίσκος που εμφανίστηκε εμπορικά ήταν το 1956 από την IBM με τη συσκευή Ramic (Random access method of accountinc and control), που είχε χωρητικότητα 5 Mb και χρόνο προσπέλασης 1 δευτερόλεπτο.

Στο Σχήμα 2.2 (δες βιβλίο των Hennessy-Patterson) παρουσιάζεται η



Σχήμα 2.2: Κόστος αποθήκευσης σε σχέση με το χρόνο προσπέλασης.

σχέση μεταξύ χρόνου και κόστους προσπέλασης για μαγνητικούς δίσκους και κύριες μνήμες. Παρατηρείται ότι με την εξέλιξη της τεχνολογίας όλα τα σύμβολα μετακινούνται προς την αρχή των αξόνων. Ωστόσο, πιο ενδιαφέροντα είναι η παρατήρηση αν και πάντοτε ο μαγνητικός δίσκος ήταν φθηνότερο μέσο αποθήκευσης από τις μνήμες (μέχρι και 100 φορές με στοιχεία 1995), εντούτοις η διαφορά στους χρόνους προσπέλασης πάντοτε ήταν συντριπτική υπέρ των μνημών (μέχρι και 10^7 φορές με στοιχεία 1995). Αν και έχει προταθεί η χρήση διαφόρων συσκευών (που στηρίζονται σε διάφορες τεχνολογίες) για να καλύψουν αυτό το κενό, το κόστος τους ήταν σχετικά ασύμφορο και γι' αυτό δεν γνώρισαν ευρεία εφαρμογή.

Συμπερασματικά, εξ αιτίας της υστέρησης της δευτερεύουσας μνήμης σε χρόνο προσπέλασης, επιβάλλεται η μελέτη των αρχείων με σκοπό την ελαχιστοποίηση του μειονεκτήματος αυτού.

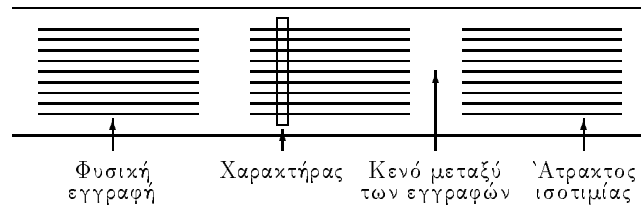
2.2 Μαγνητικές ταινίες

Αν και η τεχνολογία των μαγνητικών ταινιών έχει ήδη μία μακρόχρονη ιστορία, εντούτοις οι ταινίες χρησιμοποιούνται ακόμη και σήμερα στην πράξη, κυρίως για αποθήκευση ιστορικών και εφεδρικών αρχείων. Η ταινία έχει πλάτος 0,5 ίντσες, πάχος 1,5 χιλιοστά της ίντσας και μήκος από 300 μέχρι 3600 πόδια. Στην αρχή και στο τέλος της ταινίας υπάρχουν δύο μεταλλικές λωρίδες που ονομάζονται **σημάδι σημείου φόρτωσης** (load point mark) και **σημάδι τέλους ταινίας** (end of reel mark), αντίστοιχα. Τα καρούλια όπου

οι ταινίες τυλίγονται έχουν διάμετρο από 6,25 μέχρι 10,5 ίντσες. Συνήθως στα μεγάλα συστήματα υπολογιστών χρησιμοποιούνται ταινίες μήκους 3600 ποδών που είναι τυλιγμένες σε καρούλια διαμέτρου 10,5 ιντσών. Το υπόστρωμα της ταινίας είναι μία πλαστική βάση από Mylar, ενώ η επίστρωση είναι οξείδιο του σιδήρου ή οξείδιο του χρωμίου ή μείγμα των δύο οξειδίων.

Οι οδηγοί ταινιών για μεγάλα υπολογιστικά συστήματα έχουν τρεις κεφαλές (heads), δηλαδή μία κεφαλή διαγραφής, μία κεφαλή αποθήκευσης και μία κεφαλή ανάγνωσης. Τα σημάδια στην αρχή και το τέλος της ταινίας εντοπίζονται από φωτοκύτταρα, και έτσι οι κεφαλές ενεργοποιούνται μόνο στο ενδιαμέσο τμήμα της ταινίας. Με τη βοήθεια της κεφαλής αποθήκευσης και υπό την επίδραση ισχυρού εξωτερικού μαγνητικού πεδίου το υλικό της ταινίας μαγνητίζεται και διατηρεί την ιδιότητα αυτή μέχρις ότου ένα άλλο ισχυρό πεδίο να επιδράσει επάνω του. Η κεφαλή διαγραφής χρησιμεύει για να σβύνει κάθε αποθηκευμένη πληροφορία από την ταινία. Η κεφαλή ανάγνωσης αναγνωρίζει το μαγνητισμό της ταινίας σε επόμενα περάσματα και το μετατρέπει σε ηλεκτρικό παλμό. Στο παρελθόν, οι λεγόμενοι οικιακοί υπολογιστές ήταν συνδεδεμένοι με κοινά κασετόφωνα με μία κεφαλή και χρησιμοποιούσαν μία απλή κασέτα ως μέσο μόνιμης αποθήκευσης.

Τα δεδομένα συνήθως αποθηκεύονται σε εννέα παράλληλες ατράκτους (tracks), ενώ σπανιότερα συναντώνται ταινίες με επτά ατράκτους. Κάθε άτρακτος παριστά μία σειρά από δυαδικά ψηφία (bits) που καταχωρίζονται με πυκνότητα αποθήκευσης (recording density, *Den*) από 200 μέχρι 6250 bits ανά ίντσα (bpi). Μερικές συσκευές μπορούν να χειρισθούν ταινίες όλων των πυκνοτήτων αποθήκευσης, όμως μία μόνο πυκνότητα χρησιμοποιείται για κάθε ταινία. Κατά την ανάγνωση ή αποθήκευση δεδομένων η ταινία κινείται με καθορισμένη ταχύτητα. Συχνά η αναζήτηση γίνεται κατά τυχαίο τρόπο και συνεπώς πρέπει μία συγκεκριμένη εγγραφή να φθάσει εμπρός από την κεφαλή χωρίς απαραίτητα να γίνει επεξεργασία των προηγούμενων εγγραφών. Τότε η ταινία κινείται με πολύ μεγάλη ταχύτητα. Μάλιστα ειδικός σερβομηχανισμός ελέγχει ώστε να μην καταστραφεί η ταινία κατά τις στάσεις και τις εκκινήσεις, γιατί η αναπτυσσόμενη επιτάχυνση ή επιβράδυνση είναι της τάξης των 25g. Επίσης στην ταινία υπάρχουν διαστήματα χωρίς δεδομένα, που είναι απαραίτητα ώστε να μειωθεί η ταχύτητα μέχρι το επιθυμητό σημείο και να γίνει κατόπιν η ανάγνωση ή η αποθήκευση. Αυτά τα διαστήματα ονομάζονται κενά μεταξύ των εγγραφών (interrecord gaps, *Irg*) ή κενά μεταξύ των block (interblock gaps, *Ibg*) και παρουσιάζονται στο Σχήμα 2.3. Ο χρόνος που απαιτείται για τη σάρωση των κενών μεταξύ των εγγραφών στη συνέχεια θα συμβολίζεται με $T_{s/s}$ (δηλαδή Χρόνος εκχί-



Σχήμα 2.3: Δείγμα μαγνητικής ταινίας.

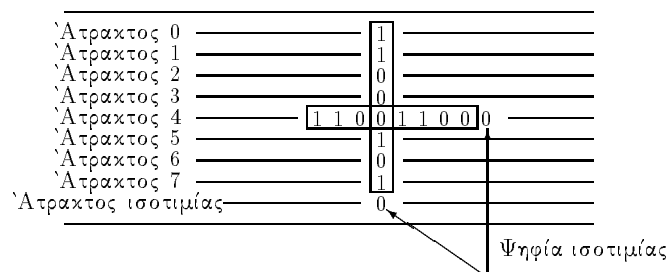
νησης/στάσης). Το μέγεθος των κενών είναι από 0,3 μέχρι 0,8 της ίντσας, ενώ συνήθως είναι μεγαλύτερο όσο μικρότερη είναι η πυκνότητα αποθήκευσης. Η τυπική τιμή του για ταινίες των 7 και των 9 ατράχτων είναι 0.75 και 0.6 ίντσες, αντίστοιχα. Ως **παράγοντας χρησιμοποίησης χώρου** (storage utilization factor, U) ορίζεται ο λόγος του μήκους της φυσικής εγγραφής προς το άθροισμα του μήκους της φυσικής εγγραφής και του κενού.

Η έννοια της εγγραφής, όπως είναι γνωστή από τον προγραμματισμό, στις μαγνητικές ταινίες αποκαλείται **λογική εγγραφή** (logical record) και χαρακτηρίζεται από το αντίστοιχο μήκος (logical record length, R). Οι λογικές εγγραφές ομαδοποιούνται σε μικρά σύνολα και αποτελούν μία **φυσική εγγραφή** ή **block** (physical record, B). Επομένως τώρα μπορεί να διασαφηνισθεί ότι ένα κενό μεσολαβεί μεταξύ δύο φυσικών εγγραφών. Το πλήθος των λογικών εγγραφών που περιέχεται σε ένα block λέγεται **παράγοντας ομαδοποίησης** (blocking factor, Bfr). Το block πρέπει να είναι όσο το δυνατό μεγαλύτερο, ώστε να ελαχιστοποιηθεί το συνολικό μέγεθος των κενών της ταινίας, αλλά ταυτόχρονα πρέπει να χωρά στις **απομονωτικές μνήμες** (buffers). Συνήθως μάλιστα το μέγεθος του block ισούται με το μέγεθος των απομονωτικών μνημών.

Τα δεδομένα ανταλλάσσονται μεταξύ κύριας μνήμης και ταινίας σε bytes των 8 bits. Στις οκτώ ατράχτους που βρίσκονται σε μία κατακόρυφη γραμμή της ταινίας αποθηκεύεται ένα byte. Ο αριθμός των άσπων σε ένα byte των οκτώ bits λέγεται **βάρος** (weight) Hamming. Η έννατη άτρακτος χρησιμεύει για την αποθήκευση του bit **ισοτιμίας** (parity). Στην **άρτια ισοτιμία** (even parity) το bit αυτό είναι άσπος ή μηδέν, έτσι ώστε το συνολικό βάρος Hamming να είναι άρτιο. Αντίθετα στην **περιττή ισοτιμία** (odd parity) το bit ισοτιμίας είναι άσπος ή μηδέν, ώστε το συνολικό βάρος Hamming να είναι περιττό.

Το bit ισοτιμίας χρησιμοποιείται για τον **κατακόρυφο** (vertical) έλεγχο λαθών που μπορεί να συμβούν στα bits ενός byte. Έτσι αν κατά την ανά-

γνώση της ταινίας διαπιστωθεί λάθος, τότε ενεργοποιείται μία διαδικασία διακοπής. Με άλλα λόγια επιχειρούνται πολλές προσπάθειες ανάγνωσης του συγκεκριμένου σημείου της ταινίας μέχρις ότου ή το λάθος να εντοπισθεί ή ο αριθμός των επαναλήψεων να φθάσει μία μέγιστη τιμή. Η αιτία των δυσλειτουργιών αυτών είναι είτε η ευαισθησία της ταινίας στη σκόνη, είτε η μη σωστή εφαρμογή της ταινίας στον οδηγό της. Επίσης περιοδικά τοποθετούνται bytes **ισοτιμίας** (parity) για τη διευκόλυνση των **κατά μήκος** (longitudinal) ελέγχων διαδοχικών bytes. Στο Σχήμα 2.4 δίνεται ένα παράδειγμα ελέγχων ισοτιμίας. Από το σχήμα φαίνεται ότι είναι εύκολο να προσδιορισθεί η θέση ενός λάθους, αν χρησιμοποιούνται και οι δύο τεχνικές.



Σχήμα 2.4: Έλεγχοι άρτιας ισοτιμίας.

Ο τρόπος επεξεργασίας μίας ταινίας είναι σχετικά απλός. Η επεξεργασία προχωρεί λαμβάνοντας όλα τα blocks σειριακά, γεγονός βέβαια που είναι και το μεγαλύτερο μειονέκτημά της. Συνεπώς, οι εντολές προς μία συσκευή ταινίας είναι 'ανάγνωση του επόμενου block', 'αποθήκευση στο επόμενο block' και 'επιστροφή στην αρχή της ταινίας'. Επειδή συνήθως είναι πολύ δύσκολο ένα block να αποθηκευθεί ακριβώς επάνω από ένα άλλο, γι' αυτό και συχνότερα η αποθήκευση γίνεται στο τέλος της ταινίας με προσάρτηση (append). Λόγω της σειριακής φύσης του μέσου, μία τυχαία αναζήτηση στο μέσο της ταινίας απαιτεί χρόνο της τάξης των μερικών λεπτών. Συνεπώς, η ταινία ενδείκνυται για εφαρμογές που αποκλειστικά απαιτούν σεριακή επεξεργασία. Ο απαιτούμενος χρόνος για την ανάγνωση των n λογικών εγγραφών ενός αρχείου είναι:

$$\frac{n}{Bfr} \times \left(T_{s/s} + \frac{Bfr \times R}{Spd \times Den} \right)$$

όπου Spd είναι η ταχύτητα μεταφοράς από/προς την ταινία και μετράται σε ίντσες/δευτερόλεπτο, ενώ ο απαιτούμενος χώρος για την αποθήκευση των

n λογικών εγγραφών του αρχείου είναι:

$$\frac{n}{Bfr} \times \left(Irg + \frac{Bfr \times R}{Den} \right)$$

Οι συσκευές που περιγράφηκαν προηγουμένως λέγονται **εκκίνησης/στάσης** (start/stop) και συνήθως δεν χρησιμοποιούνται πλέον στην πράξη. Υπάρχουν όμως σύγχρονες συσκευές μαγνητικών ταινιών που εργάζονται χωρίς να σταματούν στα κενά διαστήματα. Αυτές οι συσκευές λέγονται, στα αγγλικά, οδηγόι ταινιών streamer και χρησιμοποιούνται συνήθως στην αντιγραφή δίσκων για παραγωγή εφεδρικών αντιγράφων. Γενικά η ύπαρξη των κενών είναι απαραίτητη, γιατί η επεξεργασία του κάθε block μπορεί να είναι πιο αργή από την ανάγνωσή του. Στον Πίνακα 2.1 (δες βιβλίο του Harbron) παρουσιάζονται τα χαρακτηριστικά μερικών συσκευών οδηγών ταινιών. Οι πρώτες δύο συσκευές του πίνακα είναι οδηγόι εκκίνησης/στάσης, ενώ οι επόμενοι τρεις είναι οδηγόι τύπου streamer, που διακρίνονται για τις υψηλές τους χρονικές επιδόσεις σε σύγκριση με τους οδηγούς τύπου εκκίνησης/στάσης.

Μοντέλο	bits/ ίντσα/ άτρακτο	Ταχύτητα ανάγνωσης (in/sec)	Ταχύτητα περιστροφής (in/sec)	Ταχύτητα μεταφοράς (kb/sec)
HP 7970E	1600	45	160	88
DEC TU81	1600/6250	25	192	125
DEC TU81	1600/6250	75	192	367
HP 7980A	1600/6250	125	320	611
IBM 3480	16900	40	150	1160

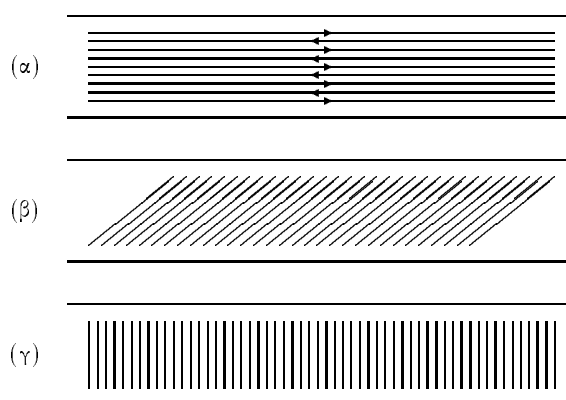
Πίνακας 2.1: Χαρακτηριστικά μερικών οδηγών ταινιών.

Όσα αναφέρθηκαν προηγουμένως αναφέρονται στον πρώτο χρονολογικά τύπο ταινίας που διατίθεται εμπορικά μέχρι σήμερα. Ο τύπος αυτός αποθήκευσης ονομάζεται **γραμμικός** (linear) ή **επιμήκης** (longitudinal). Όμως στις ημέρες μας διατίθενται εμπορικά και άλλοι τρεις τύποι ταινιών που διαφέρουν σε σχέση με τον τρόπο αποθήκευσης και επεξεργασίας των δεδομένων:

- ο τύπος **οφιοειδούς αποθήκευσης** (serpentine recording), που αποτελεί μία παραλλαγή του γραμμικού τύπου. Στις ταινίες αυτές ο αριθμός των ατράκτων είναι ιδιαίτερα μεγάλος, οπότε οι άτρακτοι ομαδοποιούνται σε αρκετά σύνολα. Για παράδειγμα, η ταινία Quantum

DLT4000 έχει 208 ατράκτους που ομαδοποιούνται σε 64 σύνολα. Ο τύπος, λοιπόν, αυτός έχει τη δυνατότητα ανάγνωσης/αποθήκευσης σε διαφορετικό σύνολο ατράκτων καθώς η ταινία κινείται και προς τις δύο κατευθύνσεις, όπως φαίνεται στο Σχήμα 2.5α. Έτσι, στις ταινίες αυτές ο χρόνος της εντολής 'επιστροφή στην αρχή της ταινίας' είναι μικρότερος σε σχέση με το γραμμικό τύπο ταινιών.

- ο τύπος **ελικοειδούς σάρωσης** (helical scan), που κατ' εξοχήν εφαρμόζεται στις βιντεοταινίες VCR. Στις ταινίες αυτές τα δεδομένα αποθηκεύονται σε ατράκτους που έχουν γωνία 10° ως 20° ως προς τον άξονα της ταινίας, όπως φαίνεται στο Σχήμα 2.5β, ενώ οι κεφαλές είναι στερεωμένες σε ένα κύλινδρο που επίσης περιστρέφεται υπό γωνία ως προς τον άξονα της ταινίας. Σε σχέση με τους προηγούμενους τύπους η ταινία αυτή κινείται με μικρή ταχύτητα κατά την ανάγνωση/αποθήκευση, ενώ σε περίπτωση εκτέλεσης εντολών rewind και forward κινείται με ταχύτητα από 50 μέχρι 75 φορές γρηγορότερα από την κανονική ταχύτητα. Μειονέκτημα της ταινίας είναι το γεγονός ότι οι κεφαλές εφάπτονται με την ταινία, το οποίο έχει επίπτωση στη διάρκεια ζωής της ταινίας. Πιο συγκεκριμένα, μετά τα 1500 περάσματα της ταινίας υπάρχει αυξημένη πιθανότητα απώλειας δεδομένων, ενώ οι οφιοειδείς ταινίες έχουν αυξημένη αξιοπιστία μέχρι και 500.000 περάσματα, που ισοδυναμεί 3,5 χρόνια συνεχούς λειτουργίας. Ωστόσο, το πλεονέκτημα της ταινίας αυτής είναι η μεγαλύτερη πυκνότητα αποθήκευσης από 20 μέχρι 50 φορές.



Σχήμα 2.5: Σύγχρονοι τύποι μαγνητικών ταινιών.

- ο τύπος εγκάρσιας αποθήκευσης (transverse recording), όπου τα δεδομένα αποθηκεύονται σε ατράκτους κατακόρυφες προς τον άξονα της ταινίας, όπως παρουσιάζεται στο Σχήμα 2.5γ. Οι ταινίες του τύπου αυτού διακρίνονται από υψηλή πυκνότητα αποθήκευσης αλλά και χαμηλή ταχύτητα μεταφοράς λόγω της μη σταθερής κίνησης της ταινίας. Ο τύπος αυτός χρησιμοποιείται σε ειδικές εφαρμογές, όπως για αποθήκευση δεδομένων από αισθητήρες που καταγράφονται με χαμηλούς ρυθμούς.

Ο Πίνακας 2.2 δίνει μερικά βασικά χαρακτηριστικά σύγχρονων ταινιών. Οι τρεις πρώτες ταινίες είναι οφιοειδούς τύπου, ενώ οι δύο τελευταίες ταινίες είναι ελικοειδούς τύπου.

Μοντέλο	Χωρητικότητα/ ταινία (Gb)	Ταχύτητα μεταφοράς (Mb/s)
Quantum DLT4000	20	1,5
Quantum DLT7000	35	5
IBM 3590	10	9
Exabyte 8505XL	7	0,5
Ampex DST	165	15

Πίνακας 2.2: Χαρακτηριστικά σύγχρονων ταινιών οφιοειδούς και ελικοειδούς τύπου.

Ήδη υπάρχουν διαθέσιμες εμπορικά συσκευές που στεγάζουν και χειρίζονται πολλές ταινίες. Με βάση το μέγεθος, οι συσκευές αυτές ανήκουν σε τέσσερις κατηγορίες:

- stacker libraries, που έχουν ένα (μόνο) οδηγό όπου ανάλογα με τη ζήτηση εναλλάσσονται μέχρι 10 ταινίες περίπου,
- carousel libraries, που διαθέτουν 1 ή 2 οδηγούς ταινιών και μπορούν να χειρισθούν από 50 μέχρι 100 ταινίες σε cartridge.
- silos, που χρησιμοποιούν μέχρι 4 οδηγούς για το χειρισμό μέχρι 6000 ταινιών.
- large libraries, που διαθέτουν 1 ή 2 ρομποτικούς μηχανισμούς για τον εντοπισμό της κατάλληλης ταινίας, από 1 μέχρι 5 οδηγούς και μπορούν να στεγάσουν από εκατοντάδες μέχρι μερικές χιλιάδες ταινίες.

Τύπος	Συσκευή	αριθμός οδηγών	αριθμός ταινιών	χωρητι- κότητα	χρόνος (sec) αλλαγής
stacker	Exabyte-10i	1	10	50 Gb	≤ 20
carousel	SpectraLogic	1-2	45	225 Gb	10
silo	StorageTek PowderHorn	1-4	6000	4800 Tb	10
large lib	IBM Magstar 3495/L50	4-64	18940	568 Tb	≤ 7

Πίνακας 2.3: Χαρακτηριστικά μερικών βιβλιοθηκών για ταινίες.

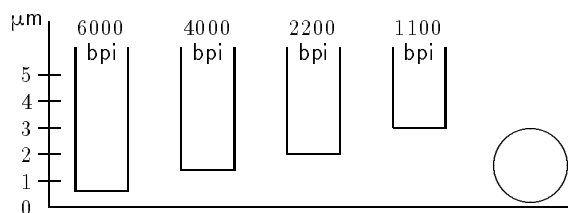
Ο Πίνακας 2.3 περιλαμβάνει μερικές χαρακτηριστικές συσκευές του εμπορίου για κάθε κατηγορία. Στις συσκευές αυτές, το χρονικό κόστος αλλαγής ταινίας και το χρονικό κόστος κίνησης της ταινίας (search + rewind delay) θεωρούνται οι πλέον σημαντικοί παράγοντες κόστους, και επιβαρύνουν τις χρονικές επιδόσεις ανάλογα με τις ιδιαιτερότητες της κάθε συσκευής. Έτσι υπάρχουν συσκευές, οι οποίες όταν πρέπει να αντικαταστήσουν μία ταινία στο πλατώ, πρέπει οπωσδήποτε να επαναφέρουν την ταινία στη φυσική της αρχή (πχ. συσκευές Exabyte). Αντίθετα, κάτι τέτοιο δεν είναι απαραίτητο σε άλλες συσκευές (πχ. συσκευές Ampex), όπου είναι δυνατόν πριν αντικατασταθεί μία ταινία, η επαναφορά να γίνει στο πλησιέστερο από ένα σύνολο καθορισμένων σημείων.

2.3 Μαγνητικοί δίσκοι

Η σημαντικότερη συσκευή αποθήκευσης δεδομένων είναι οι μαγνητικοί δίσκοι που ονομάζονται και αποθηκευτικές συσκευές άμεσης προσπέλασης (direct access storage device, DASD), γιατί οποιοδήποτε σημείο τους προσεγγίζεται σε κλάσμα του δευτερολέπτου κατά αντίθεση προς την ταινία, που αποτελεί ένα σειριακό μέσο αποθήκευσης. Αν και οι μαγνητικοί δίσκοι είναι συσκευές άμεσης προσπέλασης, χαρακτηριστικό που είναι κοινό και για την κύρια μνήμη, εντούτοις η διαφορά είναι ότι οποιαδήποτε διεύθυνση της κύριας μνήμης προσπελάζεται σε σταθερό χρόνο, ενώ δεν συμβαίνει το ίδιο και στην περίπτωση των μαγνητικών δίσκων.

Ο δίσκος είναι μία μεταλλική ή υάλινη σκληρή κυκλική πλάκα (platter) με επιφάνειες καλυμμένες με μαγνητικό υλικό, και περιστρέφεται περί τον άξονά του. Η διάμετρος του δίσκου ποικίλει από 1,3 μέχρι 14 ίντσες, με συννηθέστερα τα μεγέθη των 5,25 και 3,5 ιντσών. Σύστημα ή πακέτο δίσκων (disk pack) είναι μία συστοιχία από 1 μέχρι 20 μαγνητικούς δίσκους,

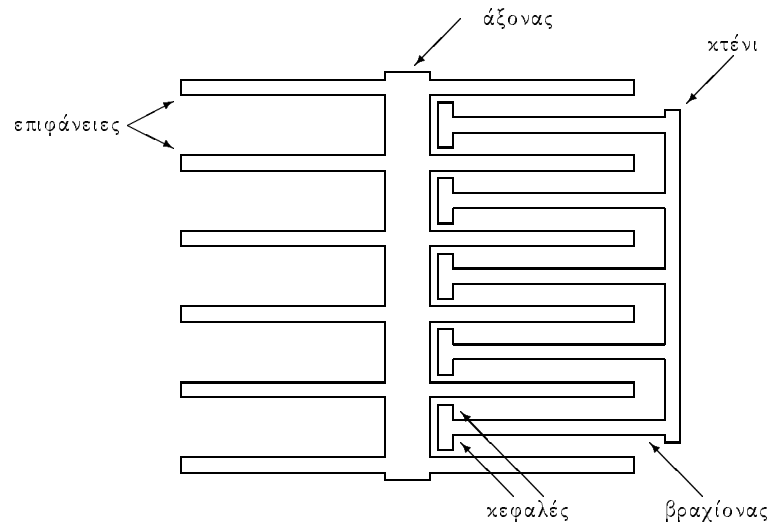
που απέχουν μεταξύ τους σταθερή απόσταση και έχουν κοινό άξονα, ενώ συνήθως στην πρώτη και στην τελευταία επιφάνεια δεν αποθηκεύονται δεδομένα. Το σύστημα αυτό είναι συνήθως κλεισμένο αεροστεγώς σε ένα σταθερό κουτί για να προστατεύεται από τη σκόνη, που μπορεί να προκαλέσει ζημιά τόσο στο δίσκο όσο και στην κεφαλή. Η συσκευή αυτή λέγεται οδηγός δίσκων Winchester κατ' αντίθεση προς τους φορητούς (removable) δίσκους, που χρησιμοποιούνται κυρίως σε μεγάλα υπολογιστικά κέντρα και μπορούν να εναλλάσσονται, αλλά δεν προστατεύονται από τη σκόνη κατά απόλυτα ασφαλή τρόπο. Στο Σχήμα 2.6 συγκρίνεται το μέγεθος του κόκου σκόνης επάνω στην επιφάνεια του δίσκου σε σχέση με την απόσταση των κεφαλών από την επιφάνεια του δίσκου για μερικές τιμές της πυκνότητας αποθήκευσης.



Σχήμα 2.6: Σύγκριση μεγέθους κόκου σκόνης και απόστασης κεφαλών από την επιφάνεια δίσκου.

Τα δεδομένα αποθηκεύονται σε ομόκεντρες περιφέρειες που ονομάζονται άτρακτοι. Ο αριθμός των ατράκτων ανά επιφάνεια ποικίλλει από 20 (στις παλαιότερες συσκευές) μέχρι 2000 (στις πλέον σύγχρονες). Οι άτρακτοι όλων των δίσκων που έχουν την ίδια ακτίνα αποτελούν ένα κύλινδρο (cylinder). Οι κεφαλές ανάγνωσης/αποθήκευσης (read/write heads) είναι ηλεκτρομαγνητικές διατάξεις σιδηρομαγνητικής κατασκευής, που ασκούν μαγνητικό πεδίο κατά μήκος μίας ατράκτου για να αποθηκεύσουν δεδομένα στο δίσκο, ενώ αντίστροφα κατά την ανάγνωση δεδομένων από κάποια άτρακτο παρουσιάζεται ρεύμα στην κεφαλή εξ αιτίας του μόνιμου μαγνητισμού του δίσκου. Κατά τη λειτουργία του δίσκου η απόσταση μεταξύ κεφαλής και επιφάνειας είναι της τάξης των μικροϊντσών, ενώ αν η κεφαλή αγγίσει την επιφάνεια, τότε και οι δύο καταστρέφονται. Οι κεφαλές είναι στερεωμένες σε βραχίονες (arm), ενώ οι βραχίονες είναι στερεωμένοι επάνω σε ένα στέλεχος παράλληλο προς τον άξονα του δίσκου, το οποίο ονομάζεται κτένι προσπέλασης (access comb). Το κτένι κινεί όλους τους βραχίονες ταυτόχρονα και συνεπώς όλες οι κεφαλές βρίσκονται σε κάθε στιγμή σε ίση απόστα-

ση από το κέντρο του δίσκου. Υπ' όψη ακόμη ότι σε κάθε βραχίονα είναι προσαρμοσμένες δύο κεφαλές, όπως φαίνεται στο Σχήμα 2.7.



Σχήμα 2.7: Οργάνωση δίσκου.

Σημειώνουμε ότι στο παρελθόν, υπήρχαν οδηγοί δίσκων με σταθερές (fixed) κεφαλές, όπου στον βραχίονα υπήρχαν προσαρμοσμένες τόσες κεφαλές όσες και οι άτρακτοι. Έτσι οι οδηγοί αυτοί διακρίνονταν από καλή επίδοση σε σχέση με το δεύτερο τύπο, όμως με αντιστάθμισμα το αυξημένο κόστος. Ακόμη στο σημείο αυτό υπενθυμίζεται η ομοιότητα των μαγνητικών τυμπάνων με τους δίσκους σταθερών κεφαλών. Η ομοιότητα έγκειται στο γεγονός ότι και το τύμπανο έχει σταθερές κεφαλές που σε αριθμό είναι τόσες όσες είναι οι άτρακτοι στην περιμετρική επιφάνειά του.

Ήδη από τις αρχές της δεκαετίας του 80 εμφανίσθηκαν στο εμπόριο συσκευές δίσκων με δύο κεφαλές ανά επιφάνεια (DEC RA81, DEC SA82 κλπ), όπου όλες οι κεφαλές στηρίζονται σε ένα μόνο κτένι προσπέλασης. Ακόμη οι δύο κεφαλές της κάθε επιφάνειας βρίσκονται σε σταθερή απόσταση μεταξύ τους, οπότε ο κύλινδρος θεωρείται ότι συμπεριλαμβάνει τις ατράκτους που αντιστοιχούν και στα δύο σύνολα κεφαλών. Επίσης σε πειραματικό στάδιο βρίσκονται συσκευές δίσκων με δύο κινούμενες κεφαλές σε κάθε επιφάνεια που στηρίζονται σε δύο ανεξάρτητους βραχίονες. Σε μία τέτοια συσκευή, πρακτικά το κάθε κτένι εξυπηρετεί τις μισές ατράκτους ενός κυλίνδρου, και επομένως η επίδοση είναι βελτιωμένη.

Όπως τονίστηκε, όλες οι κεφαλές κινούνται συγχρονισμένα, αλλά σε κάθε συγκεκριμένη χρονική στιγμή μόνο μία από τις κεφαλές μπορεί να μεταφέρει δεδομένα από/προς την κύρια μνήμη. Έτσι όταν οι κεφαλές τοποθετηθούν σε έναν κύλινδρο, τότε ανάγνωση ή αποθήκευση μπορεί να γίνει σε οποιαδήποτε άτρακτο. Για παράδειγμα, αν κατά την αποθήκευση πολλών δεδομένων στο δίσκο μία άτρακτος γεμίσει, τότε αχαιαία η αντίστοιχη κεφαλή σηκώνεται και πλησιάζει προς το δίσκο η κεφαλή της επόμενης άτρακτου του ίδιου κυλίνδρου.

Οι άτρακτοι χωρίζονται σε **σελίδες** (pages, blocks) ή σε **τομείς** (sectors). Ο αριθμός των τομέων ανά άτρακτο κυμαίνεται από 4 μέχρι 32, ενώ το μέγεθος τους κυμαίνεται από 32 (στις παλαιότερες συσκευές) μέχρι 8192 bytes (στις νεότερες). Η μορφοποίηση και συνεπώς η μεθοδευμένη προσέλαση του δίσκου γίνεται με δύο τρόπους: χρησιμοποιώντας **διευθύνσεις σελίδων** (block addressability) ή χρησιμοποιώντας **διευθύνσεις άτρακτων** (sector addressability). Ο πρώτος τρόπος συναντάται περισσότερο σε μεγάλα συστήματα, όπου από τον κατασκευαστή γίνεται η μορφοποίηση του δίσκου σε σελίδες που μπορεί να είναι διαφορετικού μεγέθους σε κάθε άτρακτο. Ο δεύτερος τρόπος συναντάται περισσότερο στα μικρά συστήματα, όπου τη μορφοποίηση σε τομείς σταθερού μεγέθους (σε bytes) εκτελεί ο χρήστης με ειδική εντολή/πρόγραμμα (format). Ας σημειωθεί ότι η μορφοποίηση/φορμάρισμα από τον κατασκευαστή είναι μόνιμη και λέγεται **σκληρή σελιδοποίηση** (hard sectoring), ενώ η μορφοποίηση από το χρήστη δεν είναι μόνιμη και λέγεται **μαλακή σελιδοποίηση** (soft sectoring). Από πρακτική άποψη οι δύο έννοιες της σελίδας και του τομέα έχουν το ίδιο περιεχόμενο: αποτελούν το ελάχιστο ποσό δεδομένων που μεταφέρεται μεταξύ δίσκου και κύριας μνήμης και γι' αυτό στη συνέχεια θα χρησιμοποιείται μόνο ο όρος σελίδα.

Και οι δύο μέθοδοι μορφοποίησης δεσμεύουν σε κάθε άτρακτο και σε κάθε σελίδα κάποιο χώρο για δεδομένα του συστήματος, όπως ποσό άγραφου χώρου, χαρακτήρες ελέγχου, διευθύνσεις κλπ. Η διεύθυνση κάθε σελίδας δίνεται από τον **αριθμό της** (physical block number) που εκφράζει την απόσταση από την πρώτη σελίδα του δίσκου. Από τον αριθμό αυτό, ο ελεγκτής του δίσκου υπολογίζει την επακριβή θέση της σελίδας (δηλαδή τον κύλινδρο, την άτρακτο και τη θέση μέσα στην άτρακτο) και ενεργοποιεί την κατάλληλη κεφαλή.

Οι δίσκοι που υποβάλλονται σε μορφοποίηση σκληρής σελιδοποίησης διακρίνονται σε δύο βασικούς τύπους:

- σε σταθερής γωνιακής ταχύτητας (constant angular velocity, CAV), και
- σε σταθερής γραμμικής ταχύτητας (constant linear velocity, CLV).

Ωστόσο σημειώνεται ότι υπάρχουν δύο παραλλαγές των δύο αυτών τύπων:

- η μορφοποίηση τροποποιημένης σταθερής γωνιακής ταχύτητας (modified constant angular velocity, MCAV), και
- η μορφοποίηση τροποποιημένης σταθερής γραμμικής ταχύτητας (modified constant linear velocity, MCLV).

Οι μαγνητικοί δίσκοι είναι βασικά τύπου CAV, ενώ οι πλέον σύγχρονοι είναι MCAV. Οι οπτικοί δίσκοι είναι κυρίως τύπου CLV, και γι' αυτό οι λεπτομέρειες της τεχνολογίας αυτής θα εξετασθούν στο επόμενο κεφάλαιο.

Το βασικό χαρακτηριστικό των δίσκων CAV είναι η σταθερή πυκνότητα αποθήκευσης (constant bit density) για κάθε άτρακτο και ο σταθερός χρόνος μεταφοράς σελίδας (που θα εξηγηθεί λεπτομερέστερα στη συνέχεια), οπότε η γραμμική ταχύτητα και το μήκος σελίδας είναι μεταβλητά μεγέθη. Πιο συγκεκριμένα, η γραμμική ταχύτητα είναι μικρότερη στις εσωτερές ατράκτους από ότι στις εξώτερες. Έτσι, οι σελίδες που βρίσκονται πλησιέστερα προς τον άξονα περιστροφής έχουν μικρότερο μέγεθος σε σχέση με τις σελίδες που βρίσκονται στους εξώτερους κυλίνδρους, αλλά διακρίνονται από μεγαλύτερη πυκνότητα αποθήκευσης σε bits/inch.

Η τεχνολογία των δίσκων MCAV στηρίζεται στη λεγόμενη τεχνική των ζωνών (zoning), που σημαίνει ότι οι διαδοχικές άτρακτοι ομαδοποιούνται σε ζώνες. Σε κάθε ζώνη τα λειτουργικά χαρακτηριστικά του δίσκου παραμένουν σταθερά, ενώ μεταβάλλονται από ζώνη σε ζώνη. Οι άτρακτοι των εξώτερων ζωνών περιέχουν περισσότερες σελίδες απ' ότι των εσωτερικών ζωνών. Συνεπώς, οι σελίδες των εξώτερων ζωνών έχουν μικρότερο μήκος και με μικρότερο χρόνο μεταφοράς (σε σχέση με τις σελίδες των εσωτερικών ζωνών), και επομένως ο ρυθμός μεταφοράς δεδομένων είναι μεγαλύτερος. Ο αριθμός των ζωνών ποικίλλει στα διάφορα προϊόντα. Έτσι, προς το παρόν υπάρχουν δίσκοι από 3 μέχρι 20 ζώνες, ενώ αναμένεται αυτός ο αριθμός να διπλασιασθεί μέχρι το τέλος της χιλιετίας. Για παράδειγμα, ο δίσκος HP C2240 έχει ρυθμούς μεταφοράς από 3,1 MB/sec για την εσωτερική ζώνη, και μέχρι 5,3 MB/sec για την εξώτερη.

Αναφέρονται πέντε βασικές παράμετροι κατά την κοστολόγηση του απαιτούμενου χρόνου για τη μεταφορά δεδομένων από το δίσκο με κινητές κεφαλές στην κύρια μνήμη και αντίστροφα.

- Η πρώτη παράμετρος κόστους, και μάλιστα η σημαντικότερη, είναι ο χρόνος που απαιτείται για να κινηθεί ο βραχίονας από έναν κύλινδρο σε κάποιον άλλο, και ονομάζεται **χρόνος αναζήτησης ή εντοπισμού** (seek time, s). Στο παρελθόν, ο χρόνος εντοπισμού είχε μοντελοποιηθεί σαν γραμμική συνάρτηση της διανυόμενης απόστασης, d . Όμως, αυτή η παλινδρομική κίνηση του βραχίονα υποστηρίζεται από πολύ εξελιγμένους μηχανισμούς που επιτυγχάνουν επιταχύνσεις της τάξης των 30-40g. Έτσι, στους σύγχρονους δίσκους κατά τη μετάβαση του βραχίονα από μία θέση κυλίνδρου σε μία άλλη, διακρίνεται η φάση της επιτάχυνσης του βραχίονα, η φάση της κίνησης με σταθερή ταχύτητα και η φάση της επιβράδυνσης. Με το σχεπτικό αυτό, ο χρόνος εντοπισμού μοντελοποιείται από τη σχέση:

$$s = \begin{cases} c_1 + c_2 \times \sqrt{d} & \text{αν } d < cutoff \\ c_3 + c_4 \times d & \text{αν } d \geq cutoff \end{cases}$$

Για παράδειγμα, οι αντίστοιχες τιμές για τη συσκευή HP 97560 είναι $c_1=3,24$ ms, $c_2=0,4$ ms, $c_3=8$ ms, $c_4=0,008$ ms, $cutoff=383$, ενώ ο δίσκος αυτός έχει χωρητικότητα 1,3 Mb και 1962 κυλίνδρους συνολικά. Για την ιστορία σημειώνεται ότι στους δίσκους σταθερών κεφαλών και στα τύμπανα δεν συναντάται αυτός ο παράγοντας.

- Ο δίσκος κινείται συνεχώς με σταθερή γωνιακή ταχύτητα (όπως εξηγήθηκε προηγουμένως). Συνεπώς αν εντοπισθεί ο ζητούμενος κύλινδρος και κάποια κεφαλή προσεγγίσει την κατάλληλη άτρακτο του συγκεκριμένου κυλίνδρου, τότε απαιτείται κάποιος χρόνος ώστε η κεφαλή να έρθει επάνω από την κατάλληλη σελίδα της άτρακτου. Ο χρόνος αυτός λέγεται **λανθάνων περιστροφικός χρόνος** (rotational latency time, r) και είναι η δεύτερη παράμετρος κόστους.
- Η τρίτη παράμετρος είναι ο χρόνος που απαιτείται για να περάσει η κεφαλή επάνω από τη σελίδα και να μεταφερθούν τα δεδομένα της στην κύρια μνήμη. Επομένως, εύλογα αυτός ο χρόνος λέγεται **χρόνος μεταφοράς της σελίδας** (block transfer time, btt).
- Η τέταρτη παράμετρος είναι ο χρόνος που απαιτείται για την ενεργοποίηση μίας κεφαλής μετά την απενεργοποίηση μίας άλλης. Η παράμε-

τρος αυτή είναι πολύ μικρότερης σημασίας από τις προηγούμενες και δεν θα εξετασθεί στη συνέχεια.

- Τέλος, η πέμπτη παράμετρος σχετίζεται με την γενικότερη αρχιτεκτονική του υπολογιστικού συστήματος και συμπεριλαμβάνει την **καθυστέρηση λόγω ουρών** (queueing delay), και τον απαιτούμενο **χρόνο από τον ελεγκτή του δίσκου** (controller time). Η παράμετρος αυτή θα γίνει καλύτερα αντιληπτή στο επόμενο κεφάλαιο.

Γενικά Χαρακτηριστικά	W.Digital WDAC 32500	IBM 0662 SXX	Quantum Prodrive 1800S	Seagate Barracuda 4LP
έτος κατασκευής	1995		1994	1995
διάμετρος δίσκου (in)	3.5	3.5	3.5	3.5
χωρητικότητα (Gb)	2.6	1.05	1.8	2.15
ταχύτητα περιστροφής (rpm)	5200	5400	4500	7200
αριθμός επιφανειών	3	3	7	
ρυθμός μεταφοράς (MB/s)	16.6	6	10	20
αναζητήσεις (ms)				
ελάχιστος χρόνος	3		3	0.9
μέσος χρόνος	13	9	10	9
εγγύηση εταιρείας (έτη)	3	5	5	5
μέσος χρόνος μεταξύ βλαβών MTBF (ώρες)	300000	800000	350000	1000000

Πίνακας 2.4: Χαρακτηριστικά μερικών οδηγών δίσκων.

Στον Πίνακα 2.4 παρουσιάζονται στοιχεία που δίνονται από κατασκευαστές δίσκων για την ταχύτητα περιστροφής, το ρυθμό μεταφοράς δεδομένων και τους χρόνους αναζήτησης. Μέχρι πρόσφατα εθεωρείτο ότι ο χρόνος αναζήτησης είναι σημαντικά μεγαλύτερος από το λανθάνοντα περιστροφικό χρόνο, που με τη σειρά του είναι μεγαλύτερος από το χρόνο μεταφοράς δεδομένων. Ωστόσο, τα νεότερα δεδομένα υπαγορεύουν ότι οι πρώτοι δύο παράγοντες πλέον αποτελούν συγκρίσιμα μεγέθη και επομένως κάθε προσπάθεια ελαχιστοποίησης του χρόνου απόκρισης θα έπρεπε να θεωρήσει και τους δύο παράγοντες.

Η τεχνολογία των μαγνητικών δίσκων έχει προοδεύσει ταχύτατα τα τελευταία χρόνια. Πιο συγκεκριμένα, έχουν επιτευχθεί μικρότερες αποστάσεις μεταξύ της κεφαλής και των επιφανειών, περισσότερο επακριβή ηλεκτρονικά συστήματα τοποθέτησης των κεφαλών στις ατράκτους, καθώς επίσης και

αυξημένη επιφανειακή πυκνότητα (areal density) αποθήκευσης, που υπολογίζεται από τον τύπο:

$$\text{Επιφανειακή πυκνότητα} = \frac{\text{άτρακτοι}}{\text{ίντσα}} \times \frac{\text{bits}}{\text{ίντσα}}$$

Μάλιστα, αυτή η αυξημένη επιφανειακή πυκνότητα έχει συντελέσει στη βελτίωση των επιδόσεων των δίσκων κατά δύο τρόπους:

- μειώνοντας το μέγεθος του δίσκου (για παράδειγμα, από 5,25 ίντσες το 1983 σε 1,3 ίντσες το 1993), οπότε μικραίνουν οι χρόνοι αναζήτησης, και
- αυξάνοντας την ταχύτητα μεταφοράς των δεδομένων (σε συνδυασμό με την αύξηση της περιστροφικής ταχύτητας από 3.600 το 1980 σε 5.400-7.200 rpm).

Ωστόσο, είναι χρήσιμο να παρατηρήσουμε ότι η πρόοδος αφορά περισσότερο στη χωρητικότητα παρά στις επιδόσεις των δίσκων. Για παράδειγμα, ο χρόνος αναζήτησης από 20 ms το 1980 έφθασε πρόσφατα μόλις τα 10 ms. Αυτό φαίνεται ανάγλυφα από τα στοιχεία του επόμενου πίνακα.

	Συνήθεις τιμές (1993)	Ετήσια πρόοδος (%)
Επιφανειακή πυκνότητα	50-150 Mbits/τετρ.ίντσα	27%
Γραμμική πυκνότητα	40.000-60.000 bpi	13%
Απόσταση ατράκτων	1.500-3.000 άτρακτοι/ίντσα	10%
Χωρητικότητα	100-2000 Mb	27%
Ρυθμός μεταφοράς	3-4 Mb/s	22%
Χρόνος αναζήτησης	7-20 ms	8%

Πίνακας 2.5: Εξελίξεις στην τεχνολογία των δίσκων.

Στα επόμενα κεφάλαια θα εξετασθεί, μεταξύ των άλλων, η επίδραση των διαφόρων οργανώσεων στο χρόνο εντοπισμού και στο χρόνο περιστροφής. Ο χρόνος που απαιτείται από την κεντρική μονάδα για επεξεργασία των πληροφοριών είναι τρεις τάξεις μεγέθους μικρότερος και γι' αυτό συνήθως αγνοείται.

2.4 Οπτικοί δίσκοι

Η οπτική τεχνολογία είναι η πιο σύγχρονη τεχνολογία αποθήκευσης δεδομένων και έχει πολλά περιθώρια ανάπτυξης στο μέλλον. Ο οπτικός δίσκος (optical disk) έχει ακτίνα από 3,5 μέχρι 12 ίντσες και η επιφάνειά του είναι από χράμα τελλουρίου (πάχους από 10 μέχρι 50 nm), που έχει ιδιαίτερες θερμοαγώγιμες ιδιότητες. Μία ακτίνα laser προσπίπτει στην επάνω επιφάνεια και δημιουργεί τρύπες (τεχνική ablation) ή στην κάτω επιφάνεια και δημιουργεί φυσαλίδες (τεχνική vesication), που ανιχνεύονται λόγω της διαφορετικής ανάκλασης σε σχέση με την παρθένα επιφάνεια. Ας σημειωθεί, επίσης, ότι οι δίσκοι που εγγράφονται στην κάτω επιφάνεια είναι πιο αξιόπιστοι, γιατί είναι λιγότερο ευαίσθητοι στη σκόνη.

Μέχρι πρόσφατα υπήρχαν εμπορικά διαθέσιμοι οδηγοί δίσκων και αντίστοιχοι δίσκοι σε δύο τύπους που διαφέρουν ως προς τη δυνατότητα του οδηγού για αποθήκευση ή όχι στο δίσκο. Οι τύποι αυτοί είναι:

- compact disk read only memory (CD-ROM). Οι δίσκοι αυτοί προσφέρονται μόνο για ανάγνωση, και έχουν ήδη προτυποποιηθεί σε σχέση με την κωδικοποίηση και την αποθήκευση των δεδομένων στην επιφάνεια, καθώς και σε σχέση με τους απαραίτητους κώδικες αναγνώρισης/διόρθωσης λαθών.
- compact disk recordable, (CD-R), που είναι αποθηκευτικοί δίσκοι, καθώς μάλιστα ονομάζονται και δίσκοι 'μίας αποθήκευσης, πολλών αναγνώσεων' (write once read many, WORM).

Οι δίσκοι WORM διακρίνονται σε δύο κατηγορίες ανάλογα με τον τρόπο ελέγχου των αποθηκευόμενων δεδομένων:

- τους δίσκους άμεσης ανάγνωσης μετά την αποθήκευση (direct read after write, DRAW), όπου ο έλεγχος της ορθότητας των αποθηκευόμενων δεδομένων γίνεται στην επόμενη περιστροφή του δίσκου, και
- τους δίσκους άμεσης ανάγνωσης κατά την αποθήκευση (direct read during write, DRDW), όπου ο σχετικός έλεγχος παραγματοποιείται τη στιγμή της αποθήκευσης συγκρίνοντας με τα δεδομένα της απομονωτικής μνήμης.

Προφανώς, στη δεύτερη μέθοδο οι αποθηκεύσεις γίνονται εξ ίσου αποτελεσματικά όσο οι αναγνώσεις, πράγμα που δεν συμβαίνει με την πρώτη μέθοδο.

Η χρήση των δίσκων WORM γίνεται με ιδιαίτερη προσοχή, γιατί οι εγγραφές δεν ανανεώνονται ούτε διαγράφονται, ενώ οι εισαγωγές αποθηκεύονται με προσάρτηση (append) σε απομακρυσμένο σημείο από το υπόλοιπο αρχείο. Για το λόγο αυτό είναι απαραίτητη η χρήση ειδικών τεχνικών οργάνωσης και διαχείρισης αρχείων που θα εξετασθούν στο τελευταίο κεφάλαιο των ειδικών θεμάτων.

Ήδη βέβαια στο εμπόριο διατίθενται οδηγοί και αντίστοιχοι δίσκοι που επιτρέπουν την επανα-αποθήκευση σε κάθε συγκεκριμένο σημείο του δίσκου, οι λεγόμενοι CD-erasable, (CD-E). Ανάλογα με την αρχή λειτουργίας, οι συσκευές αυτές χωρίζονται σε δύο κατηγορίες:

- στις συσκευές με **αλλαγή φάσης** (phase change), όπου η επιφάνεια των δίσκων (με επίστρωση από τελλούριο ή σεληνίο) έχει την ιδιότητα ανάλογα με τη θερμοκρασία να βρίσκεται σε άμορφη ή χρυσταλλική κατάσταση, και
- στις **μαγνητοοπτικές** (magneto-optic) συσκευές, όπου η επίστρωση των δίσκων είναι από ειδικά σιδηρομαγνητικά κράματα, που αλλάζουν πολικότητα όταν η θερμοκρασία ξεπερνά τους 150° C.

Και για τις δύο κατηγορίες είναι ευνόητο ότι στη μία κατάσταση γίνεται η αποθήκευση και στην άλλη γίνεται η ανάγνωση. Ο μηχανισμός των μαγνητοοπτικών συσκευών είναι ιδιαίτερα πολύπλοκος και διαφαίνεται ότι θα επικρατήσουν οι συσκευές αλλαγής φάσης. Μάλιστα δίσκοι που στηρίζονται στην αλλαγή φάσης μπορούν να αναγνωσθούν από οδηγό δίσκων WORM. Η ύπαρξη των δίσκων CD-E δεν απομακρύνει από την επικαιρότητα τους δίσκους WORM, που αποτελούν το ιδανικό μέσο για πλήθος εφαρμογών, όπως για αποθήκευση εγχειροπαιδιών, λεξικών και γενικά στατικών βάσεων δεδομένων.

Σε σύγκριση με τους μαγνητικούς δίσκους, οι οπτικοί δίσκοι διακρίνονται γενικά από:

- τεράστια χωρητικότητα (πχ. 650 Mb για δίσκους CD-ROM, και περισσότερο από 7 Gb για δίσκους WORM διαμέτρου 12 ιντσών),
- μεγαλύτερο χρόνο μεταφοράς δεδομένων (πχ. από 150 Kb μέχρι 6 Mb για ταχύτητες από 1X μέχρι 40X),
- μεγαλύτερο χρόνο προσπέλασης (πχ. από 150 μέχρι 350 ms),

- πολύ καλή αξιοπιστία επειδή η απόσταση μεταξύ της επιφάνειας του δίσκου και του συστήματος της κεφαλής είναι σχετικά μεγάλη,
- μεταφερσιμότητα, και τέλος,
- μεγάλη διάρκεια ζωής (τουλάχιστον 10 χρόνια, και στο μέλλον αναμένεται 100 χρόνια, σε σύγκριση με τη διάρκεια των 2-10 χρόνων για μαγνητικούς δίσκους).

Προς το παρόν γίνεται έρευνα για την επίτευξη ακόμη μεγαλύτερων χωρητικότητων σε οπτικούς δίσκους χρησιμοποιώντας μικρότερο μήκος κύματος laser ώστε να μειωθεί το μέγεθος των τρυπών/φυσσαλίδων. Επίσης, γίνεται έρευνα για την ελαχιστοποίηση της μάζας του κινούμενου μηχανισμού, ώστε να βελτιωθεί ο χρόνος αναζήτησης. Έτσι μέχρι στιγμής υπάρχει ανταγωνισμός δύο τύπων: του MultiMedia Compact Disk (MMCD) των Sony-Philips και του Super Density (SD) των Toshiba-Time/Warner, με αντίστοιχες χωρητικότητες 7,4 και 10 Gb σε δίσκο διαμέτρου 12 cm.

Όσον αφορά στη μορφοποίηση των οπτικών δίσκων επαναλαμβάνεται ότι βασικά είναι τύπου CLV. Το πλεονέκτημα των συσκευών CLV είναι η καλύτερη χρήση του χώρου λόγω ομοιόμορφης πυκνότητας αποθήκευσης σε όλο το δίσκο. Το μειονέκτημα είναι ότι η κεφαλή σε ένα τέτοιο δίσκο δεν μπορεί να μετρήσει τις ατράκτους που προσπερνά καθώς κινείται με μεγάλη ταχύτητα. Έτσι η γωνιακή ταχύτητα αυξάνεται καθώς οι κεφαλές κινούνται προς τον άξονα περιστροφής, ώστε όλες οι εγγραφές να περνούν από το μηχανισμό ανάγνωσης με τον ίδιο ρυθμό ανεξάρτητα της θέσης τους στην επιφάνεια του δίσκου. Όμως είναι δυνατόν να χρειασθούν περισσότερες της μίας προσπάθειες μέχρι να εντοπισθεί η αναζητούμενη άτρακτος. Το αποτέλεσμα είναι ότι η μετακίνηση του μηχανισμού και συνεπώς η αναζήτηση είναι σχετικά αργότερες σε συσκευές CLV από ότι σε συσκευές CAV. Για παράδειγμα, η συσκευή SONY WDD531 έχει μέσο χρόνο προσπέλασης 190 ms για δίσκους CAV και 800 ms για δίσκους CLV.

Είναι σημαντικό να αναφερθεί ότι στους δίσκους CLV δεν υπάρχουν οι γνωστοί ομόκεντροι άτρακτοι των δίσκων CAV, αλλά μία ή περισσότερες σπειροειδείς άτρακτοι. Η μορφοποίηση θεωρείται με βάση μία ακτίνα του δίσκου, οπότε μία νοερή άτρακτος βρίσκειται μεταξύ δύο διαδοχικών τομών της ακτίνας και μίας σπειροειδούς ατράκτου. Έτσι, το σύνολο των διαδοχικών νοερών ατράκτων ίδιας χωρητικότητας μπορεί να θεωρηθεί ότι αποτελούν μία ζώνη (δίσκοι MCLV).

Σε σχέση με τους τέσσερις παράγοντες κόστους που αναφέρθηκαν για τους μαγνητικούς δίσκους, υπάρχει μία διαφοροποίηση που αφορά στο χρόνο αναζήτησης κατά την προσπέλαση οπτικών δίσκων CLV. Πιο συγκεκριμένα, δεδομένης της θέσης της κεφαλής, υπάρχει μία μικρή γειτονιά ατράκτων που εντοπίζονται χωρίς κάποια χρονοβόρα παλινδρομική μετακίνηση του βραχίονα. Δηλαδή, με μία απλή περιστροφική κίνηση ενός φακού στο μηχανισμό της κεφαλής προσεγγίζονται μερικές άτρακτοι προς τα αριστερά και δεξιά της τρέχουσας θέσης της κεφαλής με αμελητέο κόστος. Το σύνολο των ατράκτων αυτών ονομάζεται: **εγγύς παράθυρο** (proximal window). Ωστόσο, η περιστροφή αυτή μπορεί να γίνει μέχρι μία μέγιστη γωνία, οπότε το μέγεθος του εγγύς παραθύρου είναι σχετικά περιορισμένο. Έτσι, η μοντελοποίηση και ο αναλυτικός υπολογισμός της επίδοσης, καθώς και η προσομοίωση ενός τέτοιου συστήματος καθίσταται πιο δύσκολη υπόθεση.

Κατ' αναλογία προς τις βιβλιοθήκες ταινιών που μπορούν να χειρισθούν πολλές ταινίες, οι **βιβλιοθήκες οπτικών δίσκων** (optical disk libraries, jukebox) μπορούν να στεγάσουν πολλούς οπτικούς δίσκους, που τοποθετούνται από μία ρομποτική διάταξη σύμφωνα με τη ζήτηση επάνω στο πλατώ, όπου προσπελάζονται από κατάλληλο μηχανισμό. Οι συσκευές αυτές διακρίνονται από τεράστιες αποθηκευτικές ικανότητες. Για παράδειγμα, η συσκευή Hitachi OL321 αποτελείται από 64 δίσκους WORM με συνολική χωρητικότητα 0,5 Tb, ενώ ο χρόνος αλλαγής δίσκου (robot delay) στο πλατώ είναι λιγότερο από 10 δευτερόλεπτα. Επίσης, μία συσκευή της Ascent αποτελείται από 200 δίσκους CD-ROM. Εξ άλλου υπάρχουν συσκευές όπου μπορούν να εγκατασταθούν ταυτόχρονα δίσκοι CD-ROM, CD-R και CD-E. Είναι προφανές ότι κατά τη μελέτη της επίδοσης των βιβλιοθηκών δίσκων η χρονική επιβάρυνση για την αλλαγή δίσκου πρέπει να ληφθεί υπ' όψη ως ο πλέον σημαντικός παράγοντας κόστους.

Ως αντικαταστάτης του δίσκου CD-ROM θεωρείται ο οπτικός δίσκος DVD-ROM, δηλαδή digital versatile disk read only memory. Οι τρύπες/φυσαλίδες στην επιφάνεια του δίσκου DVD είναι σημαντικά μικρότερες από τις αντίστοιχες του δίσκου CD, με αποτέλεσμα ο δίσκος DVD να έχει 7 φορές μεγαλύτερη χωρητικότητα. Έτσι, ένας τέτοιος δίσκος μπορεί να αποθηκεύσει 8 ώρες μουσική, 135 λεπτά ψηφιακής κινούμενης εικόνας (με κωδικοποίηση MPEG-2) ή 4,7 Gb. Όμως, επιπλέον προβλέπεται οι δίσκοι DVD της επόμενης γενιάς να διαθέτουν δύο στρώματα (dual layer), επομένως διπλάσια χωρητικότητα, ενώ αργότερα θα υπάρξουν δίσκοι DVD με δεδομένα και στις δύο επιφάνειες του δίσκου (dual side/dual layer), επομένως τετραπλάσια χωρητικότητα.

Σήμερα τα αρχικά DVD θεωρείται ότι προέρχονται και από τους αγγλικούς όρους Digital Video Disk, λόγω του γεγονότος ότι αναμένεται η καθιέρωσή του το χώρο της διασκέδασης και της ψυχαγωγίας. Τα πλεονεκτήματα του δίσκου DVD σε σχέση με τη βιντεοταινία είναι:

- πολύ μεγαλύτερη αντοχή,
- πολύ μικρότερο κόστος, και
- άμεση προσπέλαση (ενώ η ταινία είναι σειριακό μέσο).

Το μειονέκτημά του είναι ότι απαιτείται ειδική συσκευή για την ανάγνωσή του (DVD-ROM driver), που ωστόσο μπορεί να αναγνωρίσει και δίσκους CD.

2.5 Άλλες συσκευές αποθήκευσης

Οι προσωπικοί υπολογιστές έχουν μικρούς εύκαμπτους (floppy) δίσκους των 3,5 (και στο παρελθόν 5,25) ιντσών. Οι δίσκοι αυτοί έχουν βασικές διαφορές από τους μεγάλους σκληρούς δίσκους. Είναι πλαστικοί από υλικό Mylar, κλεισμένοι σε τετράγωνες προστατευτικές πλαστικές θήκες και δεν συσχευάζονται σε συστοιχίες. Οι σελίδες κάθε ατράχτου διακρίνονται μεταξύ τους με τη μέθοδο της μαλακής σελιδοποίησης. Από άποψη επίδοσης έχουν αυξημένους χρόνους αναζήτησης, περιστροφής και μεταφοράς δεδομένων κατά δέκα περίπου φορές. Επίσης από άποψη πυκνότητας και χωρητικότητας η διαφορά είναι πολύ μεγάλη. Οι κεφαλές ανάγνωσης/αποθήκευσης κατά τη λειτουργία τους εφάπτονται στους εύκαμπτους δίσκους (που σημαίνει μειωμένη αξιοπιστία), και έτσι ο μηχανισμός είναι πολύ πιο απλός, αργός και φθηνός.

Οι συσκευές που θα αναφερθούν στη συνέχεια δεν έχουν γνωρίσει πολύ ευρεία χρήση παρά μόνο σε ειδικές εφαρμογές και πολλές από αυτές βρίσκονται ακόμη στο στάδιο της ανάπτυξής τους. Οι συσκευές διεύθυνσης με βάση το περιεχόμενο (content addressable filestores) στηρίζονται σε ειδικού τύπου υλικό και λογισμικό, ώστε οι εγγραφές να αναζητούνται όχι με βάση την τιμή του πρωτεύοντος κλειδιού αλλά με λογικούς συνδυασμούς των τιμών πολλών κλειδιών. Το κύριο πεδίο χρήσης των συσκευών αυτών είναι για Ανάκτηση Πληροφοριών (Information Retrieval) σε βιβλιοθηκονομικά συστήματα. Τυπικό παράδειγμα είναι η συσκευή CAFS-ISP της ICL με χωρητικότητα 900 Gb σε πλήρη ανάπτυξη.

Οι συσκευές μαζικής αποθήκευσης (mass storage) χρησιμοποιούν πλήθος από κασέτες για μακροχρόνια αποθήκευση σε συνδυασμό με μαγνητικούς δίσκους για την αποθήκευση των δεδομένων κατά την επεξεργασία. Για παράδειγμα, οι συσκευές Control Data 3850 και IBM 3850 σε πλήρη ανάπτυξη έχουν συνολική χωρητικότητα 1.000 και 472 Gb, αντίστοιχα. Το μειονέκτημα βέβαια είναι ότι ο χρόνος προσπέλασης και αλλαγής των κασετών είναι της τάξης των μερικών δευτερολέπτων.

Οι μνήμες μαγνητικών φουσάλιδων (magnetic bubble memories) έχουν καλούς χρόνους προσπέλασης (από 4 μέχρι 7 ms), αλλά υστερούν σε χωρητικότητα (από 92.000 μέχρι 1310.720 bits) και χρόνο μεταφοράς (από 50.000 μέχρι 5000.000 bps) σε σχέση με τις συσκευές δίσκων. Η τεχνολογία των μνημών αυτών έχει αγγίσει τα όριά της και διαπιστώθηκε ότι δεν μπορεί να χρησιμοποιηθεί για εμπορικούς παρά μόνο για στρατιωτικούς και διαστημικούς σκοπούς, επειδή δεν έχουν κινητά και συνεπώς ευαίσθητα μέρη.

Οι συσκευές ημιαγωγικών δίσκων (semiconductor disks) εμφανίσθηκαν εμπορικά το 1978 και έκτοτε έχουν γνωρίσει σημαντική εξέλιξη και εφαρμογή σε περιπτώσεις αποθήκευσης βιβλιοθηκών προγραμμάτων, καταλόγων, προσωρινών αρχείων εργασίας, κλπ. Οι ημιαγωγά δίσκοι διακρίνονται από πάρα πολύ γρήγορη προσπέλαση (πχ. 0,5 ms), μεγάλη χωρητικότητα (πχ. 700 Mb), χαμηλό κόστος και κατανάλωση ρεύματος. Πολλές τέτοιες συσκευές μπορούν να μεταδώσουν ταυτόχρονα δεδομένα από πολλά κανάλια (από 2 μέχρι 4) επιτυγχάνοντας πολύ υψηλούς ρυθμούς μεταφοράς δεδομένων (πχ. 12 Mb/sec).

Ωστόσο, η οπτική τεχνολογία διαφαίνεται ότι είναι η τεχνολογία του μέλλοντος, καθώς ήδη υπάρχουν νέες πρωτοποριακές συσκευές δευτερεύουσας αποθήκευσης. Η οπτική ταινία (optical tape) διακρίνεται για τη δυνατότητα αποθήκευσης τεράστων χωρητικότητων σε πολύ μικρό όγκο. Για παράδειγμα, στο τέλος της χιλιετίας το υπόστρωμα της ταινίας θα φθάσει στο πάχος από 0,076 mm σε 0,013 mm. Επίσης, πραγματική αποθήκευση γίνεται μόνο στο 1/2 της επιφάνειας του οπτικού δίσκου, σε αντίθεση με την οπτική ταινία, όπου αποθήκευση γίνεται σε περισσότερο από τα 4/5 της επιφάνειάς της. Έτσι, τελικά για την ίδια επιφανειακή πυκνότητα αποθήκευσης, η πυκνότητα αποθήκευσης ως προς τον όγκο για μία οπτική ταινία είναι 25 φορές περισσότερο από ότι σε ένα οπτικό δίσκο, με προοπτική να γίνει 75 φορές περισσότερο. Ακόμη σε σχέση με τη μαγνητική ταινία, η οπτική ταινία διακρίνεται από:

- πολλαπλάσια αποθηκευτική ικανότητα (πχ. μαγνητική ταινία 3480 ποδών αποθηκεύει 10 GB, ενώ για την οπτική ταινία αναμένεται δυνατότητα για αποθήκευση 200 Gb στο τέλος της χιλιετίας),
- πολύ μεγάλες ταχύτητες μεταφοράς δεδομένων (πχ. 3 Mb/sec με προοπτική από 4,5 μέχρι 10 Mb/sec στο τέλος της χιλιετίας), και
- μεγαλύτερη διάρκεια ζωής.

Επί του παρόντος στα προϊόντα που διατίθενται στο εμπόριο χρησιμοποιείται ο γραμμικός τρόπος οργάνωσης των δεδομένων στην ταινία, υποστηρίζοντας μόνο τεχνολογία WORM.

Μία άλλη κατεύθυνση είναι η αξιοποίηση της τεχνικής της ολογραφίας (holography) για την ανάπτυξη ειδικών συσκευών, με σκοπό την εκμετάλλευση των τριών διαστάσεων του χώρου για την αποθήκευση δεδομένων. Για παράδειγμα, πειραματικά έχει επιτευχθεί η αποθήκευση 1000 εικόνων σε κρύσταλλο οξειδίου λιθίου-νιοβίου μήκους 1 cm, ενώ εκτιμάται ότι μπορεί να επιτευχθεί αποθήκευση δεδομένων 10 Tb σε 1 cm³. Βέβαια, παρά την υψηλή αποθηκευτική ικανότητα, πρόβλημα παραμένει η ταχύτητα μεταφοράς δεδομένων.

Τέλος, μία άλλη κατεύθυνση είναι προς την ανάπτυξη δίσκων που στηρίζονται στο φαινόμενο της παγίδευσης ηλεκτρονίων (electron trapping). Με απλά λόγια, κατά την αποθήκευση, ένα στρώμα υλικού τύπου φωσφόρου δέχεται ακτίνα φωτός και εκπέμπει ηλεκτρόνια, που απορροφώνται από μία αγωγίμη ζώνη. Για την ανάγνωση δεδομένων, ο δίσκος σαρώνεται από φως διαφορετικής συχνότητας που αναγκάζει τα ηλεκτρόνια να εξέλθουν από την αγωγίμη ζώνη και να εκπέμψουν φωτόνια. Ανάλογα με την ένταση της ακτίνας φωτός κατά την αποθήκευση, εξάγονται περισσότερα ή λιγότερα ηλεκτρόνια, και κατά συνέπεια φωτόνια κατά την τελική φάση. Αυτό ισοδυναμεί με την αποθήκευση περισσότερων από ένα bit σε κάθε θέση. Έτσι, εκτιμάται ότι με τη μέθοδο αυτή θα επιτευχθούν υψηλές αποθηκευτικές επιδόσεις και ταχύτητες μεταφοράς δεδομένων στο μέλλον. Μειονέκτημα είναι ότι μετά από κάθε ανάγνωση πρέπει να ακολουθήσει επανα-αποθήκευση με την ίδια αρχική διαδικασία. Ο Πίνακας 2.6 συνοψίζει τα μέχρι στιγμής δεδομένα για τις συσκευές που στηρίζονται στην οπτική τεχνολογία.

	Οπτικός δίσκος	Οπτική ταινία	Ολογραφία	Παγίδευση ηλεκτρονίου
Πλεονεκτήματα	Βιομηχανικό πρότυπο	Κόστος/Mb Διάρκεια ζωής	Πυκνότητα αποθήκευσης, Παραλληλισμός	Αποθηκευτική ικανότητα, Ταχύτητα μεταφοράς
Χωρητικότητα	5 Gb σε δίσκο 15 cm	200 Gb σε ταινία 3480 ft	1 Gb σε κάρτα 30x30 mm	10 Gb σε δίσκο 13 cm
Κύρια χρήση	Υπέρβαση εύκαμπτου δίσκου	Αρχειοθέτηση	Φορητά συστήματα	Βίντεο
Ανταγωνιστής	Φορητοί μαγνητικοί δίσκοι	Μαγνητική ταινία	Μαγνητικός δίσκος σε κάρτα	Πίνακες δίσκων RAID
Μειονεκτήματα	Χρόνος αναζήτησης	Σειριακή προσπέλαση	Καταστροφική ανάγνωση	Καταστροφική ανάγνωση

Πίνακας 2.6: Σύγκριση συσκευών που στηρίζονται στην οπτική τεχνολογία.

2.6 Πίνακες δίσκων RAID

Γενικά, οι χρήσεις των μαγνητικών δίσκων διακρίνονται σε δύο κατηγορίες:

- σε μεγάλες εγκαταστάσεις, υψηλού κόστους και επιδόσεων, που στηρίζονται σε αξιόπιστους μεγάλους δίσκους (πχ. μεγαλύτερους σε διάμετρο από 25 cm), και
- σε σταθμούς εργασίας και προσωπικούς υπολογιστές που χρησιμοποιούν φθηνότερους και μικρότερους δίσκους.

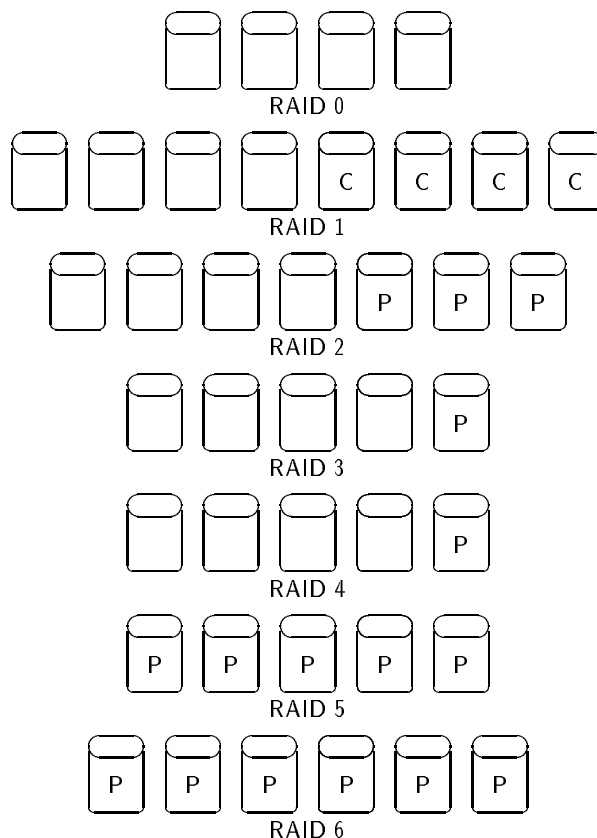
Στη δεκαετία του 80, το χάσμα μεταξύ αυτών των δύο κατηγοριών δίσκων ήταν σημαντικό. Από μία ομάδα ερευνητών, λοιπόν, τέθηκε το πρόβλημα: είναι δυνατόν να αντικατασταθεί ένας μεγάλος δίσκος από ένα σύνολο μικρών και φθηνών δίσκων; Έτσι, από τους Patterson, Katz και Gibson προτάθηκε (1988) η χρήση πινάκων δίσκων χαμηλού κόστους αντί των μεγάλων συστημάτων δίσκων. Οι διατάξεις αυτές ονομάστηκαν **Πλεονάζοντες Πίνακες μη Ακριβών Δίσκων** (Redundant Arrays of Inexpensive Disks, RAID).

Αν και οι μαγνητικοί δίσκοι είναι το κυρίαρχο μέσο αποθήκευσης, εντούτοις οι συσκευές αυτές δεν έχουν άπειρο χρόνο ζωής αλλά συχνά παθαίνουν βλάβες. Μία σημαντική παράμετρος, λοιπόν, που ενδιαφέρει κάθε

είδους χρήστη και κάθε είδους εφαρμογή είναι η αξιοπιστία του συστήματος. Όμως αν υποθεθεί ότι για ένα δίσκο ο μέσος χρόνος μεταξύ βλαβών (mean-time-between-failure, MTBF) είναι 200.000 ώρες, τότε ένα σύστημα 100 δίσκων θα έχει 2.000 ώρες ζωή πριν πάθει βλάβη. Το γεγονός ότι ένα σύστημα RAID αποτελείται από φθηνούς δίσκους, επιτρέπει τη λύση του προβλήματος της αξιοπιστίας με τον πλεονασμό (redundancy) των αποθηκευμένων δεδομένων, όπου τα δεδομένα αποθηκεύονται είτε αυτούσια δύο φορές, είτε συμπιεσμένα με τη βοήθεια ειδικών κωδικών διόρθωσης λαθών (error correcting codes). Έτσι σε περίπτωση βλάβης ενός δίσκου, είναι δυνατόν να γίνει πλήρης αποκατάσταση των δεδομένων χρησιμοποιώντας αυτούς τους κώδικες (πχ. Hamming κλπ.). Άρα, εκτός από το κόστος αποθήκευσης, ένα άλλο πλεονεκτήματα των συστημάτων RAID είναι η αυξημένη αξιοπιστία.

Τέλος, ένα τρίτο πλεονέκτημα των συστημάτων RAID είναι οι υψηλές επιδόσεις. Λόγω της ύπαρξης πολλών δίσκων, οι αποθηκεύσεις/αναγνώσεις από/προς τους δίσκους γίνονται παράλληλα χρησιμοποιώντας την τεχνική του stripping. Σύμφωνα με την τεχνική αυτή, μία λογική σελίδα τεμαχίζεται σε μικρότερα τμήματα και αποθηκεύεται στους δίσκους, έτσι ώστε εγγυημένα να υπάρχει ο μέγιστος παραλληλισμός (και όχι κάποιοι δίσκοι να εργάζονται, ενώ οι υπόλοιποι να παραμένουν αδρανείς).

Η αυξημένη επίδοση λόγω παραλληλισμού και η αυξημένη αξιοπιστία λόγω πλεονασμού είναι έννοιες αντιστρόφως ανάλογες. Δηλαδή, όσο περισσότεροι οι δίσκοι, τόσο καλύτερη η επίδοση, αλλά τόσο μικρότερη η αξιοπιστία (και το αντίστροφο). Έχουν προταθεί 7 αρχιτεκτονικές συστημάτων RAID που διαφέρουν στα σχήματα πλεονασμού και στις μεθόδους stripping. Κάθε μία από τις αρχιτεκτονικές αυτές έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Στο Σχήμα 2.8 παρουσιάζονται τα 7 επίπεδα RAID. Μπορεί να λεχθεί ότι το επίπεδο 0 δεν είναι RAID, αλλά παρατίθενται για λόγους σύγκρισης. Το επίπεδο RAID 1 παριστά την τεχνική του καθρεπτισμού (mirroring) ή σκίασης (shadowing), όπου για κάθε δίσκο υπάρχει ακόμη ένας πανομοιότυπος, ο οποίος συμβολίζεται με το γράμμα C (από την αγγλική λέξη copy). Τα υπόλοιπα επίπεδα χρησιμοποιούν ειδικούς κώδικες διόρθωσης λαθών, οπότε περιττεύουν περισσότερες επεξηγήσεις στα πλαίσια του βιβλίου αυτού. Απλώς, σημειώνεται ότι το γράμμα P (από την αγγλική λέξη parity) δηλώνει που τοποθετούνται τα δεδομένα ισοτιμίας. Επίσης, σε σχέση με το σχήμα σημειώνεται ότι σε κάθε περίπτωση, για το χρήστη οι χρήσιμοι δίσκοι είναι 4.



Σχήμα 2.8: Επίπεδα αρχιτεκτονικής RAID.

Σήμερα, υπάρχουν εμπορικά προϊόντα RAID από διάφορες εταιρείες, όπως το σύστημα IBM Ramac, που είναι μία υλοποίηση του 5ου επιπέδου, το σύστημα AutoRAID της Hewlett-Packard, που υλοποιεί μία προσαρμοστική τεχνική αυτόματης επιλογής του κατάλληλου επιπέδου (από το 1ο ως το 5ο), το σύστημα StorageTEK 9200 της Iceberg, που υλοποιεί την αρχιτεκτονική του 6ου επιπέδου κλπ. Κοινός παρονομαστής όλων αυτών των προϊόντων είναι ότι οι επιμέρους δίσκοι δεν είναι ούτε μικροί, ούτε φθηνοί. Απεναντίας, οι συσκευές αυτές σκοπό έχουν κυρίως την υψηλή αξιοπιστία και ανήκουν στην κλάση των προϊόντων υψηλών επιδόσεων (high performance). Έτσι, πολλές φορές γράφεται ότι RAID σημαίνει Redundant Arrays of Independent Disks, δηλαδή **Πλεονάζοντες Πίνακες Ανεξάρτητων Δίσκων** αντί 'μη Ακριβών'.

2.7 Ασκήσεις

<1> Μαγνητική ταινία αποτελείται από 2400 πόδια, η πυκνότητα αποθήκευσης είναι 1600 bpi και η ταχύτητα ανάγνωσης/αποθήκευσης είναι 50 ίντσες/sec. Το κενό μεταξύ των εγγραφών είναι 0,5 ίντσα, ενώ ο χρόνος εκκίνησης/στάσης είναι 0,01 sec. Αν το μήκος της λογικής εγγραφής είναι 200 bytes, ποιός είναι ο βέλτιστος παράγοντας ομαδοποίησης ώστε $N\%$ της ταινίας να καταλαμβάνεται από block. Πόσο μήκος ταινίας χρειάζεται για να αποθηκευθούν A block μεγέθους B bytes;

<2> Πακέτο δίσκων αποτελείται από 11 χρήσιμες επιφάνειες, κάθε επιφάνεια έχει 200 ατράκτους, κάθε ατράκτος έχει 20 σελίδες και κάθε σελίδα αποτελείται από 512 bytes. Πόση είναι η συνολική χωρητικότητα σε bytes; Αν υποθεθεί ότι δεν επιτρέπεται μία εγγραφή να αποθηκευθεί κατά το ήμισυ σε δύο σελίδες, τότε πόσες εγγραφές των 120 bytes χωρούν σε 10 κυλίνδρους αυτού του συστήματος;

<3> Το λειτουργικό σύστημα αντιγράφει αρχεία από το μαγνητικό δίσκο σε μαγνητική ταινία που βρίσκεται σε συσκευή εκκίνησης/στάσης. Το μέγεθος της σελίδας είναι 512 bytes, η πυκνότητα αποθήκευσης είναι 1600 bpi, το κενό μεταξύ των block της ταινίας είναι 0,5 ίντσα και τέλος σε κάθε block χωρούν 20 σελίδες. Πόσες σελίδες χωρούν σε μία ταινία μήκους 2400 ποδών;

<4> Τα αρχεία των χρηστών ενός υπολογιστικού συστήματος καταλαμβάνουν 150.000 σελίδες. Για λόγους ακεραιότητας και ασφάλειας τα αρχεία που ενημερώνονται, αποθηκεύονται σε αρχεία ταινίας στο τέλος της ημέρας. Έστω επίσης ότι καθημερινά κατά μέσο όρο 20% των σελίδων του δίσκου αποθηκεύονται στην ταινία. Πόσος χρόνος απαιτείται καθημερινά για την αντιγραφή στις ταινίες, αν είναι γνωστό ότι η γραμμική ταχύτητα της ταινίας κατά την ανάγνωση και την αποθήκευση είναι 50 ίντσες/sec και ότι ο χρόνος εκκίνησης/στάσης είναι 0,02 sec. Ό,τι άλλα δεδομένα χρειάζονται να ληφθούν από την προηγούμενη άσκηση.

<5> Ας υποθεθεί ότι για την προηγούμενη άσκηση η ανάγνωση των σελίδων από το δίσκο γίνεται κατά διαδοχικούς κυλίνδρους. Τα δεδομένα βρίσκονται σε 30.000 σελίδες που μοιράζονται εξίσου σε 500 κυλίνδρους. Το πακέτο των δίσκων αποτελείται από τέσσερις χρήσιμες επιφάνειες και οι σελίδες που αντιστοιχούν σε ένα κύλινδρο μοιράζονται εξίσου σε αυτές. Ο ελάχιστος χρόνος εντοπισμού είναι ίσος με 7 ms, ο μέσος χρόνος περιστροφής είναι

8,3 ms, ενώ ο χρόνος μεταφοράς και ο χρόνος ενεργοποίησης των κεφαλών για προσέγγιση στις επιφάνειες θεωρείται αμελητέος. Να υπολογισθεί και πάλι ο απαιτούμενος χρόνος για την αντιγραφή στις ταινίες.

<6> Να βρεθεί η μέση τιμή των κυλίνδρων που σαρώνονται σε μία τυχαία προσπέλαση. Το σύνολο των κυλίνδρων είναι C και θεωρείται ότι κάθε κύλινδρος προσπελάται ισοπίθανα. Αν υποτεθεί ότι οι κύλινδροι ενός συστήματος αρχείων δεν προσπελάζονται ισοπίθανα, τότε ποιός είναι ο καλύτερος τρόπος διάταξης του περιεχομένου των κυλίνδρων ώστε να ελαχιστοποιηθεί ο χρόνος εντοπισμού;

<7> Συνήθως οι κατασκευαστές μαγνητικών δίσκων δίνουν τον ελάχιστο, το μέγιστο και το μέσο χρόνο αναζήτησης. Η ελάχιστη και η μέγιστη τιμή αντιστοιχούν στον απαιτούμενο χρόνο για τη μετάβαση των κεφαλών σε γειτονικό κύλινδρο και από τον πρώτο ως τον τελευταίο κύλινδρο, αντίστοιχα. Τα μεγέθη αυτά είναι πραγματικά και ασφαλή. Ο μέσος χρόνος αναζήτησης υπολογίζεται ως το άθροισμα των χρόνων αναζήτησης για όλες τις δυνατές αποστάσεις δια του αριθμού των δυνατών αποστάσεων. Να υπολογισθεί ο μέσος χρόνος αναζήτησης για το δίσκο HP 97560. Γιατί στην πράξη ο μέσος χρόνος αναζήτησης είναι μικρότερος;

Κεφάλαιο 3

ΔΙΑΧΕΙΡΙΣΗ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ

3.1 Εισαγωγή

3.2 Τύποι εγγραφών

3.3 Τύποι σελίδων

3.4 Ομαδοποίηση εγγραφών

3.5 Διαχείριση χώρου δίσκου

3.6 Διαχείριση απομονωτικής μνήμης

3.7 Ασκήσεις

Κεφάλαιο 3

ΔΙΑΧΕΙΡΙΣΗ ΕΣΟΔΟΥ/ΕΞΟΔΟΥ

3.1 Εισαγωγή

Είναι γνωστό ότι ένα σύστημα διαχείρισης βάσεων δεδομένων τρέχει επάνω από το λειτουργικό σύστημα, που είναι υπεύθυνο για τη διαχείριση των αρχείων στο λεγόμενο χαμηλό φυσικό επίπεδο. Στο παρελθόν είχαν υπάρξει διχογνωμίες για τα διαχωριστικά όρια των δύο συστημάτων, δηλαδή για τις συγκεκριμένες υπηρεσίες που θα έπρεπε να προσφέρει το λειτουργικό σύστημα στο σύστημα διαχείρισης βάσεων δεδομένων. Τελικώς η ακαδημαϊκή και κατασκευαστική κοινότητα των ανθρώπων που διακονούσαν τις βάσεις δεδομένων κατέληξαν στο συμπέρασμα ότι μερικές υπηρεσίες των λειτουργικών συστημάτων είναι ανεπαρκείς σε σχέση με το δεύτερο σκοπό ενός συστήματος διαχείρισης βάσεων δεδομένων, που είναι η *αποτελεσματικότητα* (δες Κεφάλαιο 0). Οι λόγοι ήταν πρακτικοί και τεχνικοί. Για παράδειγμα:

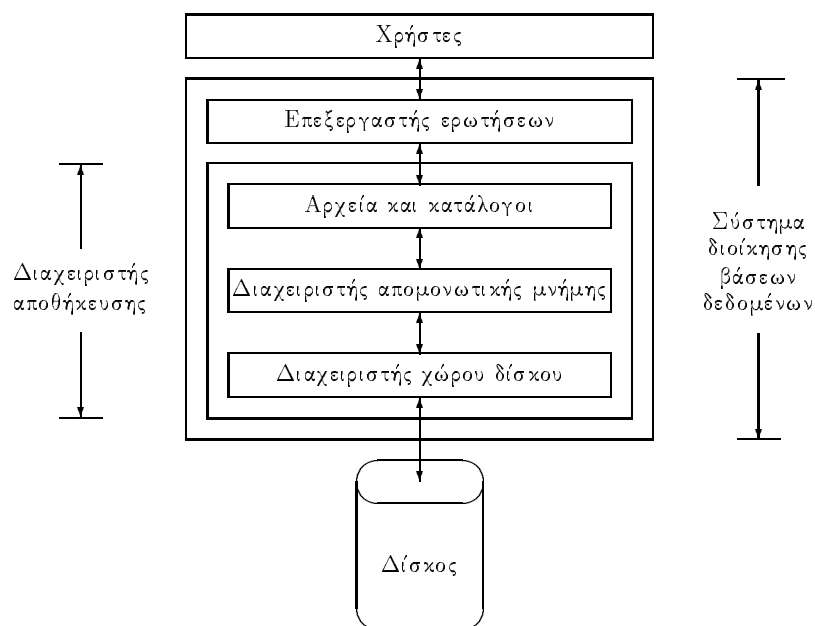
- σε συστήματα των 32 bit, το μεγαλύτερο μέγεθος αρχείου που μπορεί να διαχειρισθεί ένα λειτουργικό σύστημα είναι 4 Gb. Όμως συχνά υπάρχουν εμπορικές εφαρμογές που απαιτούν μεγαλύτερα αρχεία.
- τα αρχεία ενός λειτουργικού συστήματος δεν μπορούν να διαμοιρασθούν σε δύο ή περισσότερους δίσκους, πράγμα που είναι επιθυμητό σε πολλές περιπτώσεις. Τέλος,
- οι κατασκευαστές συστημάτων διαχείρισης βάσεων δεδομένων επιθυμούν το προϊόν τους να τρέχει σε διάφορες πλατφόρμες λειτουργικών συστημάτων. Έτσι, είναι αναγκασμένοι να μην προσαρμόσουν το

προϊόν τους προς τις ιδιαιτερότητες του κάθε λειτουργικού συστήματος, αλλά να το καταστήσουν όσο το δυνατόν περισσότερο αυτάρκες για λόγους μεταφερσιμότητας.

Περισσότεροι τεχνικοί λόγοι θα αναφερθούν στη συνέχεια του κεφαλαίου αυτού.

Έτσι, οι κατασκευαστές συστημάτων διαχείρισης βάσεων δεδομένων προχώρησαν προς την κατεύθυνση να συμπεριλάβουν στα προϊόντα τους συντελεστές που να εκτελούν λειτουργίες φυσικού επιπέδου με τρόπο ώστε να εξυπηρετούνται οι συγκεκριμένες ανάγκες των συγκεκριμένων συστημάτων με τον καλύτερο τρόπο.

Βέβαια η λεπτομερής αρχιτεκτονική ενός συστήματος διαχείρισης βάσεων δεδομένων είναι αντικείμενο που θα εξετασθεί στο αντίστοιχο μάθημα. Ωστόσο, στο σημείο αυτό αναφέρεται ελλειπτικά ότι οι βασικοί συντελεστές ενός τέτοιου συστήματος είναι ο επεξεργαστής ερωτήσεων (query processor) και ο διαχειριστής αποθήκευσης (storage manager), όπως φαίνεται στο Σχήμα 3.1. Ο πρώτος συντελεστής είναι πλησιέστερα προς το



Σχήμα 3.1: Αρχιτεκτονική συστήματος διαχείρισης βάσεων δεδομένων.

χρήστη, ενώ ο δεύτερος είναι πλησιέστερα προς το υλικό. Ειδικότερα, ο διαχειριστής αποθήκευσης διακρίνεται στα εξής βασικά τμήματα (με σειρά από κάτω προς τα επάνω):

- ο διαχειριστής του χώρου του δίσκου (disk space manager),
- ο διαχειριστής της απομονωτικής μνήμης (buffer manager), και
- τα αρχεία και τους καταλόγους.

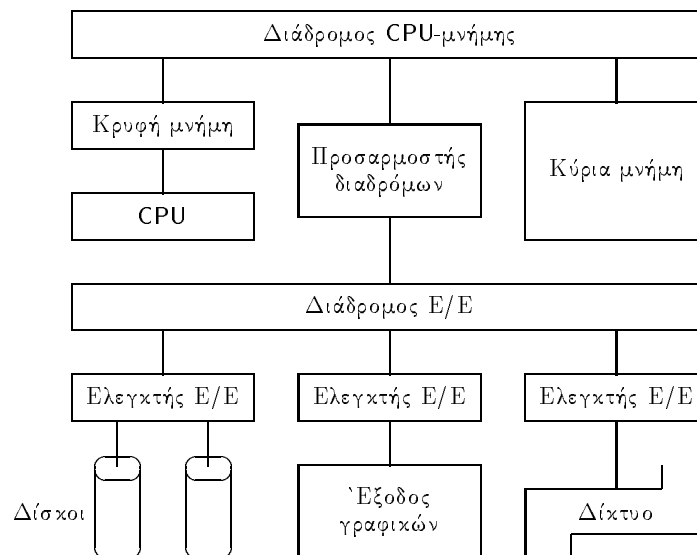
Ο κάθε ένας από τους προηγούμενους συντελεστές επικοινωνεί με το γειτονικό του στρώμα ζητώντας και δίνοντας δεδομένα. Για παράδειγμα, ο διαχειριστής του χώρου του δίσκου είναι το λογισμικό που είναι υπεύθυνο για την ανάγνωση και την αποθήκευση σελίδων στο δίσκο, ενώ ο διαχειριστής της απομονωτικής μνήμης είναι το λογισμικό που είναι υπεύθυνο για την κατάτμηση της κύριας μνήμης σε κατάλληλα τμήματα, ώστε εκεί να αποθηκεύονται προσωρινά τα περιεχόμενα των σελίδων που έρχονται από το δίσκο (μέσω του διαχειριστή του χώρου του δίσκου). Το ανώτερο επίπεδο περιλαμβάνει ένα σύνολο προγραμμάτων που υλοποιούν διάφορες δομές αρχείων. Μέσα από το συντελεστή αυτό επιτυγχάνεται και η ταχτοποίηση των εγγραφών σε σελίδες.

Το βιβλίο αυτό αναφέρεται στο μέγιστο βαθμό στον τρίτο συντελεστή της προηγούμενης λίστας. Στο παρόν κεφάλαιο θα αναπτυχθούν θέματα που αναφέρονται σε θέματα διαχείρισης εισόδου/εξόδου δεδομένων από τη δευτερεύουσα στην πρωτεύουσα μνήμη με τη βοήθεια των απομονωτικών μνημών. Όμως προηγουμένως πρέπει να αναφερθούν μερικά βασικά στοιχεία σχετικά με την αρχιτεκτονικού ενός τυπικού υπολογιστικού συστήματος.

Στα πρώτα υπολογιστικά συστήματα η κεντρική μονάδα επεξεργασίας (CPU) παρέμενε αδρανής όταν δεδομένα μεταφέρονταν από/προς τη δευτερεύουσα μνήμη. Η αδυναμία αυτή για την αποτελεσματική επικοινωνία της γρήγορης κύριας μνήμης και των βραδύτερων δευτερευουσών μνημών έχει ξεπερασθεί στους σύγχρονους υπολογιστές με τη δημιουργία μίας διεπιφάνειας (interface), που μπορεί να έχει δύο μορφές:

- τα μεγάλα συστήματα έχουν το λεγόμενο **κανάλι** (channel), που είναι ένας δεύτερος επεξεργαστής (I/O processor), ενώ
- τα μικρά συστήματα έχουν το λεγόμενο **διάδρομο δεδομένων** (data bus).

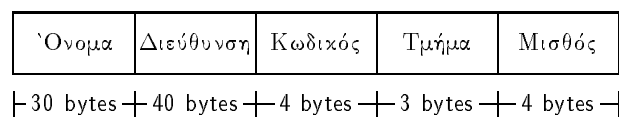
Σκοπός και των δύο είναι η εξομάλυνση του ρυθμού μεταφοράς δεδομένων. Ειδικότερα, αν και το κανάλι έχει δική του μικρή μνήμη και μία μικρή ομάδα εντολών, εντούτοις δεν είναι ισοτιμος επεξεργαστής προς τον κεντρικό επεξεργαστή. Η βασική λειτουργία του καναλιού είναι η μετατροπή των λέξεων της κύριας μνήμης σε bytes, ώστε να αποθηκευθούν στην περιφερειακή συσκευή και η μετατροπή των bytes σε λέξεις για την αντίστροφη μεταφορά. Οι λειτουργίες αυτές εκτελούνται από ένα πρόγραμμα που συνήθως είναι αποθηκευμένο στην κύρια μνήμη και στέλνεται εντολή προς εντολή από τον κεντρικό επεξεργαστή στο κανάλι όπου εκτελούνται. **Επιλεκτικό (selector)** λέγεται το κανάλι που μπορεί να ελέγχει πολλές δευτερεύουσες συσκευές, όμως πρέπει να εκτελέσει πλήρως το πρόγραμμα που αφορά σε μία συσκευή προτού αρχίσει να εκτελεί το πρόγραμμα μίας άλλης συσκευής. Το **πολυπλεκτικό κανάλι (multiplexor channel)** έχει τη δυνατότητα να εκτελεί ταυτόχρονα τα προγράμματα πολλών δευτερευουσών συσκευών. Βέβαια στην πραγματικότητα σε κάθε χρονική στιγμή δεδομένα μεταφέρονται από/προς μία μόνο συσκευή, όμως δίνεται η εντύπωση ότι δεδομένα από πολλές συσκευές μεταφέρονται ταυτόχρονα. Στο Σχήμα 3.2 παρουσιάζεται η αρχιτεκτονική ενός τυπικού συστήματος. Τα φθηνότερα συστήματα διαθέτουν μόνον ένα διάδρομο, όπου μπορεί να κυκλοφορούν εντολές της κεντρικής μονάδας επεξεργασίας αλλά και εντολές εισόδου/εξόδου.



Σχήμα 3.2: Αρχιτεκτονική υπολογιστικού συστήματος.

3.2 Τύποι εγγραφών

Το μέγεθος κάθε πεδίου μίας εγγραφής εξαρτάται από τον τύπο του. Η γραφική παράσταση της οργάνωσης των πεδίων μίας εγγραφής λέγεται **γραμμαμογράφηση** (layout). Εγγραφές με ίδια και ισομήκη πεδία αλλά διαφορετική διάταξη δεν έχουν ίδια γραμμογράφηση. Στο Σχήμα 3.3 παρουσιάζεται η γραμμογράφηση της εγγραφής ενός υπαλλήλου.



Σχήμα 3.3: Εγγραφή με σταθερό μήκος.

Οι εγγραφές μίας οντότητας συνήθως έχουν ίδια μορφή και λέγονται **εγγραφές σταθερού μήκους** (fixed length records). Κάτι τέτοιο συμβαίνει συνήθως στα λεγόμενα **σχεσιακά** (relational) συστήματα διαχείρισης βάσεων δεδομένων. Στην περίπτωση αυτή ο υπολογισμός μίας διεύθυνσης για την επεξεργασία του πεδίου μίας εγγραφής είναι εύκολη υπόθεση, αφού συμβαίνει όπως κατά τον υπολογισμό διευθύνσεων σε πίνακες (δες Κεφάλαιο 2 βιβλίου *Δομών Δεδομένων*).

Ωστόσο, μερικές φορές στα σχεσιακά αλλά και στα λεγόμενα **αντικειμενοστραφή** (object-oriented) συστήματα είναι βέβαιο ότι προκύπτουν **εγγραφές μεταβλητού μήκους** (variable length records). Μερικά λειτουργικά συστήματα δεν υποστηρίζουν εγγραφές μεταβλητού μήκους, αλλά τις χειρίζονται ως εγγραφές σταθερού μήκους. Τα πλεονεκτήματα των συστημάτων, που υποστηρίζουν τις εγγραφές μεταβλητού μήκους, είναι ιδιαίτερα σημαντικά όταν:

- υπάρχει μεγάλη απόκλιση των μήκων των εγγραφών από το μέσο μήκος εγγραφής,
- τα αρχεία είναι ογκώδη,
- είναι μεγάλη η συχνότητα χρήσης, και τέλος
- το υλικό είναι πολύ ακριβό.

Η μεταβλητότητα του μήκους των εγγραφών οφείλεται στους εξής τρεις λόγους:

- **Ύπαρξη πεδίων μεταβλητού μήκους (variable length fields).** Στα συστήματα που δεν υποστηρίζουν εγγραφές μεταβλητού μήκους, για κάθε εγγραφή δεσμεύεται χώρος ίσος με το μήκος του μεγαλύτερου στιγμιότυπου της εγγραφής. Από την άλλη πλευρά, στα συστήματα που υποστηρίζουν εγγραφές μεταβλητού μήκους γίνεται προσπάθεια εξοικονόμησης χώρου σε πεδία, που το μήκος τους ποικίλει κατά πολύ. Το φαινόμενο αυτό συναντάται συνήθως σε πεδία τύπου συμβολοσειράς όπως ονόματα, διευθύνσεις, περιγραφές κλπ.

Τύπος=Ω	Όνομα	Διεύθυνση	Τμήμα	Ωρομίσθιο	Όρες
Τύπος=Ω	Όνομα	Διεύθυνση	Τμήμα	Μισθός	

Σχήμα 3.4: Εγγραφές μεταβλητού μήκους.

- **Ύπαρξη εγγραφών διαφορετικής μορφής (variable format records).** Δηλαδή, έστω ότι σε μία εταιρεία υπάρχουν υπάλληλοι που πληρώνονται με την ώρα, την ημέρα, το μήνα ή και με το κομμάτι παραγωγής. Στο Σχήμα 3.4 φαίνεται ένα τέτοιο παράδειγμα. Πολλές φορές είναι πιθανό δύο εγγραφές να έχουν το ίδιο μήκος, αλλά να είναι διαφορετικής μορφής. Για την αποφυγή συγχύσεων, στις περιπτώσεις αυτές σίγουρη λύση είναι η χρήση ενός επιπλέον πεδίου που να δηλώνει τη μορφή της εγγραφής. Τα variant records της Pascal λύνουν αυτό το πρόβλημα.
- **Ύπαρξη επαναλαμβανόμενων ομάδων πεδίων (repeating groups).** Το φαινόμενο αυτό συμβαίνει όταν ένα ή περισσότερα χαρακτηριστικά επαναλαμβάνονται περισσότερο από δύο φορές. Για παράδειγμα, η οντότητα 'υπάλληλος' μεταξύ των άλλων μπορεί να περιέχει τα χαρακτηριστικά 'όνομα_προστατευόμενου' και 'ημερομηνία_γέννησης' προστατευόμενου. Συνεπώς η εγγραφή ενός υπάλληλου με δύο ή τρία

Όνομα	Κωδικός	Όνομα Προστατευόμ.	Ημερομηνία Γέννησης	Όνομα Προστατευόμ.	...
-------	---------	-----------------------	------------------------	-----------------------	-----

Σχήμα 3.5: Εγγραφή με επαναλαμβανόμενες ομάδες.

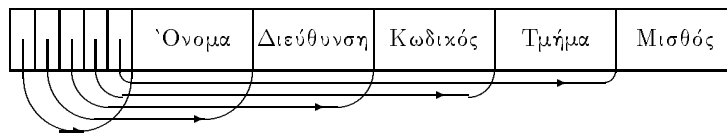
παιδιά θα περιέχει αντίστοιχα δύο ή τρία στιγμιότυπα των χαρακτηριστικών αυτών. Στο Σχήμα 3.5 παρουσιάζεται μία τέτοια περίπτωση. Μία τέτοια περίπτωση αντιμετωπίζονταν εύκολα από την Cobol με την εντολή Occurs. Ωστόσο, τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων δεν επιτρέπουν σε μία εγγραφή την ύπαρξη επαναλαμβανόμενων ομάδων και επιλύουν το πρόβλημα κατά διαφορετικό τρόπο.

Για τα σχεσιακά συστήματα ιδιαίτερο ενδιαφέρον παρουσιάζει μόνο η πρώτη περίπτωση της ύπαρξης πεδίων μεταβλητού μήκους. Έτσι, όταν ένας προγραμματιστής υλοποιεί μία δομή αρχείου πρέπει να είναι πολύ προσεκτικός στο χειρισμό των εγγραφών αυτού του είδους, επειδή απαιτείται αφ' ενός επιπλέον χώρος για την αποθήκευση δεδομένων ελέγχου και αφ' ετέρου πιο πολύπλοκο λογισμικό. Πιο συγκεκριμένα, η διαχείριση τέτοιων εγγραφών μπορεί να γίνει με δύο τρόπους:

Όνομα	\$	Διεύθυνση	\$	Κωδικός	\$	Τμήμα	\$	Μισθός
-------	----	-----------	----	---------	----	-------	----	--------

Σχήμα 3.6: Χρήση διαχωριστών σε σελίδες.

- με χρήση ειδικών διαχωριστών (delimiters) ή σημαδιών διαχωρισμού (separator marker) μεταξύ των επιμέρους πεδίων, όπως παρουσιάζεται στο Σχήμα 3.6. Η μέθοδος αυτή απαιτεί τη σειριακή σάρωση της εγγραφής για τον εντοπισμό του ζητούμενου πεδίου.



Σχήμα 3.7: Χρήση καταλόγων σε σελίδες.

- με χρήση ενός μικρού καταλόγου στην αρχή της εγγραφής, όπου για κάθε πεδίο αποθηκεύεται η απόστασή του (offset) από την αρχή της εγγραφής. Στο Σχήμα 3.7 παρουσιάζεται ένα τέτοιο παράδειγμα. Η προσέγγιση αυτή απαιτεί επιπλέον χώρο σε σχέση με την πρώτη προσέγγιση, αλλά είναι χρονικά αποτελεσματικότερη, γιατί δίνει τη δυνατότητα άμεσης προσπέλασης κάθε πεδίου.

Ένα πρόβλημα που συχνά συναντάται στις βάσεις δεδομένων είναι η ύπαρξη πεδίων με τιμή null, μία τιμή που δηλώνει ότι για κάποιο συγκεκριμένο πεδίο δεν είναι διαθέσιμη ή δεν μπορεί να υπάρξει πραγματική τιμή. Έτσι, σύμφωνα με αυτή την προσέγγιση δεν απαιτείται η αποθήκευση κάποιου συγκεκριμένου συμβόλου που να δηλώνει την τιμή null κάποιου πεδίου, αλλά απλώς η απόσταση του επόμενου πεδίου θα είναι ίση με την απόσταση του συγκεκριμένου πεδίου.

Το κυριότερο πρόβλημα που μπορεί να προκύψει λόγω της ύπαρξης πεδίων μεταβλητού μήκους είναι η αύξηση του μεγέθους της εγγραφής κατά την ενημέρωση ενός πεδίου. Ένα τέτοιο ενδεχόμενο μπορεί να προκαλέσει:

- την ανάγκη τακτοποίησης του περιεχομένου της σελίδας με την κατάλληλη μετακίνηση (shifting) μερικών εγγραφών, ή
- την ανάγκη αποθήκευσης της συγκεκριμένης εγγραφής σε άλλη σελίδα λόγω ανεπάρκειας χώρου. Αν όμως η εγγραφή αυτή είναι συνδεδεμένη με αλυσίδα από κάποια άλλη εγγραφή, τότε υπάρχει κίνδυνος να σπάσει η αλυσίδα. Για το λόγο αυτό, αν η εγγραφή μετακινηθεί, τότε στη θέση της αποθηκεύεται η νέα διεύθυνση ώστε να μη χαθούν δεδομένα.

3.3 Τύποι σελίδων

Κάθε σελίδα έχει μία επικεφαλίδα (page header), που περιέχει σημαντικές πληροφορίες, όπως αριθμός αποθηκευμένων εγγραφών στη σελίδα, διεύθυνση επόμενης σελίδας του αρχείου κλπ. Πέραν της επικεφαλίδας, λοιπόν, θα μπορούσαμε να φαντασθούμε μία σελίδα σαν ένα σύνολο θέσεων (slots) για την αποθήκευση των εγγραφών. Έτσι, κάθε εγγραφή μπορεί να χαρακτηριστεί από το σύστημα με ένα μοναδικό κωδικό εγγραφής (record identifier, rid), που αποτελείται από το ζεύγος <κωδικός σελίδας, αριθμός θέσης> (<page id, slot number>).

Αν οι εγγραφές είναι σταθερού μήκους, τότε ο εντοπισμός των εγγραφών μέσα στη σελίδα είναι εύκολος, καθώς μάλιστα υπάρχουν δύο πιθανές επιλογές:

- οι εγγραφές αποθηκεύονται στις πρώτες διαθέσιμες θέσεις της σελίδας, οπότε είναι εύκολος ο υπολογισμός των αποστάσεων (offsets).

Έτσι, σε περίπτωση διαγραφής η τελευταία εγγραφή καταλαμβάνει τη θέση της διαγραφείσας. Ωστόσο, η μέθοδος δεν είναι εφαρμόσιμη αν υπάρχουν αλυσίδες εγγραφών, γιατί αν μία εγγραφή εκτός της συγκεκριμένης σελίδας δείχνει προς τη θέση της τελευταίας εγγραφής, τότε αυτή δεν μπορεί να μετακινηθεί. Η οργάνωση αυτή παρουσιάζεται στα αριστερά του Σχήματος 3.8.

n		m	101...01
εγγραφή 1		εγγραφή 1	
εγγραφή 2			
...		εγγραφή 3	
εγγραφή n		...	
ελεύθερος χώρος			
		εγγραφή m	

Σχήμα 3.8: Οργάνωση σελίδων με εγγραφές σταθερού μήκους.

- Στα δεξιά του Σχήματος 3.8 παρουσιάζεται η εναλλακτική οργάνωση. Σύμφωνα με τη μέθοδο αυτή, στην επικεφαλίδα της σελίδας αποθηκεύεται ένας δυαδικός πίνακας με σημαίες που δηλώνουν αν μία θέση είναι κατειλημμένη ή όχι. Έτσι, κατά την αναζήτηση εξετάζονται μόνο οι θέσεις που αντιστοιχούν σε αληθείς σημαίες, ενώ κατά τη διαγραφή μίας εγγραφής, η αντίστοιχη σημαία καθίσταται ψευδής.

Αν οι εγγραφές είναι μεταβλητού μήκους, τότε οι σελίδες δεν περιέχουν σταθερό αριθμό θέσεων. Σε περίπτωση εισαγωγής, η νέα εγγραφή θα έπρεπε να καταλάβει κάποιο κενό χώρο κατάλληλου μεγέθους ώστε να μην αχρηστευθεί σημαντικό κομμάτι του, που θα συνέχιζε να παραμένει κενό. Επίσης, σε περίπτωση διαγραφής θα έπρεπε οι υπόλοιπες εγγραφές να μετακινούνται στη σελίδα, ώστε ο κενός χώρος να είναι ενοποιημένος.

Η καλύτερη τεχνική για το σκοπό αυτό είναι η ύπαρξη ενός μικρού καταλόγου θέσεων (directory of slots), ή πίνακα τοποθέτησης (position table) στην επικεφαλίδα της σελίδας, ο οποίος παρέχει όλες τις σχετικές πληροφορίες. Έτσι, ο κατάλογος αυτός περιέχει για κάθε εγγραφή το ζεύγος <απόσταση εγγραφής, μήκος εγγραφής> (<record offset, record length>). Επιπλέον, υπάρχει ένας δείκτης προς την αρχή της σελίδας από όπου αρχίζει η περιοχή αποθήκευσης των εγγραφών, καθώς και ένας δείκτης προς την αρχή της ενοποιημένης περιοχής που είναι ελεύθερη και διαθέσιμη για αποθήκευση νέων εγγραφών.

Ο κατάλογος αυτός εξυπηρετεί τις προηγούμενες προδιαγραφές αλλά θα έπρεπε να σημειωθεί ότι σε περίπτωση διαγραφής δεν μπορεί να γίνει ταχτοποίηση του ίδιου του καταλόγου με ελευθέρωση της αντίστοιχης θέσης. Ωστόσο, η θέση αυτή στον κατάλογο μπορεί να αποδοθεί προς χρήση σε επόμενη εισαγωγή εγγραφής. Έτσι, μία καινούργια θέση δημιουργείται στον κατάλογο μόνο αν όλες οι υπάρχουσες θέσεις δείχνουν σε πραγματικές εγγραφές.

Σύμφωνα με μία απλή παραλλαγή της τεχνικής αυτής, ο κατάλογος δεν περιέχει ζεύγη <απόσταση εγγραφής, μήκος εγγραφής> αλλά μόνο την <απόσταση εγγραφής>, ενώ η ένδειξη μήκους (length indicator) τοποθετείται στην αρχή της αντίστοιχης εγγραφής. Έτσι, σε περίπτωση αναζήτησης διευκολύνεται η υπέρβαση μίας εγγραφής.

3.4 Ομαδοποίηση εγγραφών

Για την ανάλυση του κεφαλαίου αυτού υποτίθεται ότι το μέσο αποθήκευσης είναι ο μαγνητικός δίσκος, αλλά η ανάλυση αυτή ισχύει εν μέρει και για την περίπτωση των μαγνητικών ταινιών. Για το χρήστη, λοιπόν, η εγγραφή είναι η ελάχιστη λογική μονάδα διακίνησης της πληροφορίας. Ωστόσο, όπως αναφέρθηκε, για το σύστημα η σελίδα είναι το ελάχιστο φυσικό ποσό δεδομένων που μεταφέρεται μεταξύ του δίσκου και της κύριας μνήμης. Ως παράγοντας ομαδοποίησης (blocking factor, Bfr) ορίζεται ο αριθμός των εγγραφών που χωρούν σε μία σελίδα. Αν υποθεθεί ότι το μέγεθος της επικεφαλίδας της σελίδας είναι αμελητέος και ότι οι εγγραφές είναι σταθερού μήκους, τότε ο παράγοντας ομαδοποίησης ισούται με:

$$Bfr = \left\lfloor \frac{B}{R} \right\rfloor$$

όπου B είναι το μέγεθος της σελίδας, R είναι το μέγεθος της εγγραφής, ενώ $B - Bfr \times R$ είναι ο κενός χώρος που παραμένει στο τέλος της σελίδας. Αν οι εγγραφές είναι μεταβλητού μήκους, τότε η μεταβλητή R συμβολίζει το μέσο μήκος των εγγραφών. Προσεγγιστικά γίνεται παραδεκτό ότι ο μέσος κενός χώρος ισούται με το μισό του μήκους της μέσης εγγραφής, $R/2$, οπότε ο παράγοντας ομαδοποίησης δίνεται από τη σχέση:

$$Bfr = \frac{B - R/2}{R}$$

Αν n είναι το πλήθος των εγγραφών, τότε ο αριθμός των σελίδων του αρχείου είναι:

$$b = \left\lceil \frac{n}{Bfr} \right\rceil$$

Αν οι κεφαλές είναι τοποθετημένες στην αρχή της σελίδας, τότε ο χρόνος μεταφοράς της σελίδας ισούται με:

$$btt = \frac{B}{t}$$

όπου t είναι ταχύτητα διακίνησης των δεδομένων μεταξύ δίσκου και μνήμης και μετράται με bytes/ms. Αν πάλι αντί της εγγραφής θεωρηθεί μία ολόκληρη άτρακτος χωρητικότητας T , τότε η ποσότητα αυτή μεταφέρεται σε μία πλήρη περιστροφή $2r$ (όπου r είναι ο μέσος χρόνος περιστροφής). Έτσι, προκύπτει η ισοδύναμη σχέση:

$$btt = \frac{T}{2r}$$

Όπως η μαγνητική ταινία είναι οργανωμένη σε φυσικές εγγραφές με κενά μεταξύ τους, έτσι και στο μαγνητικό δίσκο απαιτείται κάποια οργάνωση με σκοπό το συγχρονισμό κατά τη λειτουργία του και επομένως τη μείωση του λανθάνοντα περιστροφικού χρόνου. Δύο τρόποι για την οργάνωση ενός δίσκου συναντώνται στην πράξη:

- η ύπαρξη των κενών μεταξύ των σελίδων, και
- η χρήση της τεχνικής της παρεμβολής (interleaving) ή του hopscotching (σε ελεύθερη μετάφραση το παιχνίδι 'χουτσό'), που επιτυγχάνει καλύτερους χρόνους προσπέλασης αλλά είναι σχετικά περίπλοκη.

Η τεχνική της παρεμβολής είναι αναγκαία επειδή ο ελεγχτής του δίσκου απαιτεί κάποιο χρόνο επεξεργασίας των δεδομένων που δέχεται από το δίσκο, προτού μπορέσει να δεχθεί άλλα δεδομένα. Έτσι, αν η λογικά επόμενη σελίδα τοποθετούνταν στον φυσικά επόμενο τομέα, τότε θα έπρεπε να γίνει μία πλήρης περιστροφή του δίσκου ώστε να εντοπισθεί η αρχή της επόμενης σελίδας.

Το παράδειγμα του Σχήματος 3.9 αναφέρεται στην τεχνική της παρεμβολής. Έστω ότι 10 σελίδες ενός αρχείου πρέπει να αποθηκευθούν στους 10 τομείς μίας άτρακτου. Επίσης, ας υποθεθεί ότι ο ελεγχτής απαιτεί διπλάσιο

Φυσικός τομέας	1	2	3	4	5	6	7	8	9	10
Λογικός τομέας	1	8	5	2	9	6	3	10	7	4

Σχήμα 3.9: Παράδειγμα της τεχνικής της παρεμβολής.

χρόνο από το χρόνο που απαιτείται για την προσπέλαση του επόμενου τομέα. Άρα, όταν θα τελειώσει η επεξεργασία της φυσικής εγγραφής R1 που είναι αποθηκευμένη στον τομέα S1, η κεφαλή θα βρίσκεται επάνω από τον τομέα S4 όπου θα πρέπει να είναι αποθηκευμένη η δεύτερη φυσική εγγραφή R2. Είναι αντιληπτό ότι με την τεχνική αυτή μηδενίζεται η επιβάρυνση του χρόνου περιστροφής. Στο συγκεκριμένο παράδειγμα αρχούν τρεις περιστροφές για την ανάγνωση όλων των τομέων της τράχτου, ενώ θα χρειαζόταν δέκα περιστροφές αν δεν χρησιμοποιούνταν η τεχνική αυτή. Τα τελευταία χρόνια οι ταχύτητες των ελεγκτών έχουν βελτιωθεί σημαντικά. Έτσι, αναφέρεται από τους κατασκευαστές δίσκων ότι ο λόγος παρεμβολής είναι 1:1, που σημαίνει ότι η λογική σειρά των τομέων ταυτίζεται με τη φυσική σειρά τους.

Η συνέχεια της ανάλυσης αναφέρεται σε δίσκους που είναι φορμαρισμένοι με τη μέθοδο των κενών. Επειδή συχνά η ανάγνωση αφορά στην προσπέλαση πολλών (διαδοχικών ή μη) σελίδων από φορμαρισμένο δίσκο, εκτός από τη μεταβλητή *btt* θεωρείται μία καινούρια μεταβλητή, η μεταβλητή *ebt*, που ονομάζεται **πραγματικός χρόνος μεταφοράς σελίδας** (effective block transfer time) και ισούται με:

$$ebt = \frac{B}{t'}$$

όπου t' είναι η ταχύτητα διακίνησης δεδομένων από φορμαρισμένο δίσκο στη μνήμη και μετράται επίσης με bytes/ms. Στον Πίνακα 3.1 δίνεται μία λίστα των κυριότερων παραμέτρων και των αντίστοιχων τιμών τους σε μία συσκευή IBM 3380. Αυτά τα στοιχεία θα χρησιμοποιηθούν κατ' επανάληψη στα επόμενα κεφάλαια και στις ασκήσεις.

Αν, λοιπόν, ένα αρχείο καταλαμβάνει b σελίδες, τότε η σειριακή ανάγνωση του αρχείου απαιτεί χρόνο:

$$b \times ebt$$

ενώ η τυχαία ανάγνωση όλων των σελίδων απαιτεί χρόνο:

$$b \times (s + r + ebt)$$

Ορισμός - Παράμετρος	Τιμή
Μέγεθος σελίδας, B	2400 bytes
Χρόνος μεταφοράς σελίδας, btt	0,8 ms
Σελίδες ανά κύλινδρο, C	600
Πραγματικός χρόνος μεταφοράς σελίδας, ebt	0,84 ms
Αριθμός κυλίνδρων, N	885
Μέσος χρόνος περιστροφής, r	8,3 ms
Μέσος χρόνος εντοπισμού, s	16 ms
Ταχύτητα ανάγνωσης σε δίσκο, t	3000 bytes/ms
Ταχύτητα ανάγνωσης σε φορμαρισμένο δίσκο, t'	2857 bytes/ms

Πίνακας 3.1: Τιμές παραμέτρων για τη συσκευή IBM 3380.

όπου s είναι ο μέσος χρόνος εντοπισμού σε ms, r είναι ο μέσος χρόνος περιστροφής σε ms. Βέβαια θεωρείται αμελητέο το μέγεθος της επικεφαλίδας (header) ή σελίδας ελέγχου (control block) του αρχείου και υποτίθεται ότι οι εγγραφές έχουν σταθερό μήκος. Από τις σχέσεις αυτές φαίνεται ότι αν υπάρχει μεγάλο ποσοστό κενού χώρου, τότε η σειριακή ανάγνωση επιβραδύνεται.

3.5 Διαχείριση χώρου δίσκου

Ο διαχειριστής του χώρου του δίσκου είναι το πλησιέστερο τμήμα ενός συστήματος διαχείρισης βάσεων δεδομένων προς το υλικό. Ο διαχειριστής δίνει εντολές προς το δίσκο για να δεσμεύσει ή να αποδεσμεύσει μία σελίδα, όπως και για να αναγνωσθεί ή να αποθηκευθεί μία σελίδα. Συχνά είναι χρήσιμο ένα σύνολο σελίδων του αρχείου να αποθηκευθεί σε διαδοχικές απράκτους του δίσκου, ώστε η σειριακή ανάκτηση τους να γίνεται αποτελεσματικά. Το πλήθος των σελίδων που μπορούν να προσπελασθούν κατ' αυτόν τον τρόπο περιορίζεται από το μέγεθος της απομονωτικής μνήμης (η χρήση της οποίας θα αναλυθεί συνέχεια), και αποτελεί το λεγόμενο **κάδο** (bucket) που είναι η λογική μονάδα προσπέλασης στο δίσκο. Για παράδειγμα, στο μοντέλο EDS8 της ICL το λειτουργικό σύστημα μπορεί να διαχειρισθεί κάδους της 1 σελίδας, των 2, των 4 ή των 8 σελίδων, ενώ συνήθως το μέγεθος των κάδων είναι 8 σελίδες των 512 bytes. Ακόμη στα συστήματα Vax το μέγεθος του κάδου μπορεί να καθορισθεί από 1 μέχρι 65.535 σελίδες των 512 bytes, με προτεινόμενη default τιμή τους 3 τομείς. Παρόμοιες τεχνικές συναντώνται και σε άλλα συστήματα.

Η εκλογή του κατάλληλου μεγέθους κάδου με σκοπό τη βελτιστοποίηση της χρήσης του χώρου είναι υπόθεση του διαχειριστή του συστήματος και δεν υπολογίζεται δύσκολα. Ας υποτεθεί ότι το μέγεθος της εγγραφής είναι 160 bytes, ενώ το μέγεθος της σελίδας είναι 256 bytes. Οι επιλογές είναι οι εξής:

- οι εγγραφές να μπορούν να αποθηκεύονται εν μέρει σε δύο σελίδες με αυξημένο κόστος προσπέλασης, και
- κάθε εγγραφή να αποθηκεύεται σε μία μόνο σελίδα με κόστος τις αυξημένες απαιτήσεις σε χώρο λόγω της ύπαρξης κενού χώρου στο τέλος της σελίδας, φαινόμενο που ονομάζεται **εσωτερική τμηματοποίηση** (internal fragmentation).

Παράγοντας ομαδοποίησης	Μέγεθος εγγραφών	Σελίδες/ κάδο	Μέγεθος κάδου	Ποσοστό χρήσης
1	160	1	256	63%
2	320	2	512	63%
3	480	2	512	94%
4	640	3	768	83%
5	800	4	1024	78%
6	960	4	1024	94%
7	1120	5	1280	88%
8	1280	5	1280	100%

Πίνακας 3.2: Αντιστοιχία μεγεθών εγγραφής και τομέα.

Στον Πίνακα 3.2 δίνεται το ποσοστό χρησιμοποίησης χώρου, καθώς ο παράγοντας ομαδοποίησης αυξάνει από 1 ως 8. Παρατηρείται ότι το ποσοστό αυτό κυμαίνεται από 63% ως 100%, που προφανώς είναι η καλύτερη λύση. Στο Σχήμα 3.10 φαίνεται η αντιστοιχία του μεγέθους της εγγραφής προς το μέγεθος του τομέα για παράγοντα ομαδοποίησης 8.

Τομέας		Τομέας		Τομέας		Τομέας		Τομέας	
Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή	Εγγραφή

Σχήμα 3.10: Αντιστοιχία μεγεθών εγγραφής και τομέα.

Το ποσοστό χρησιμοποίησης του χώρου δίνεται από τη σχέση:

$$U = \frac{R \times Bfr}{S \times Ppb}$$

όπου Ppb είναι ο αριθμός των σελίδων ανά κάδο:

$$Ppb = \left\lfloor \frac{R \times Bfr}{S} \right\rfloor$$

ενώ S είναι το μέγεθος του τομέα σε bytes.

Αν η κλήση θεωρηθεί ότι γίνεται με βάση τον κάδο, τότε ο χρόνος σειριακής ανάγνωσης ενός αρχείου ισούται με:

$$bk \times dtt$$

όπου bk είναι ο αριθμός των κάδων του αρχείου και dtt είναι ο χρόνος μεταφοράς δεδομένων (data transfer data) ενός κάδου, ενώ ο χρόνος τυχαίας ανάγνωσης του αρχείου είναι:

$$bk \times (s + r + dtt)$$

Από τις προηγούμενες σχέσεις συνάγεται ότι η αλλαγή του μεγέθους του κάδου δεν επιδρά καθόλου στην ταχύτητα της σειριακής ανάγνωσης. Αντίθετα, η αύξηση του μεγέθους του κάδου αυξάνει την ταχύτητα της τυχαίας προσπέλασης του αρχείου με βάση τον κάδο, ενώ μειώνει την ταχύτητα αν η τυχαία προσπέλαση γίνεται με βάση την εγγραφή. Αν η κλήση γίνεται με βάση τις εγγραφές, τότε ο αντίστοιχος τύπος είναι:

$$n \times (s + r + dtt)$$

Δηλαδή, για την προσπέλαση μίας εγγραφής πρέπει να μεταφερθεί στην κύρια μνήμη ένας ολόκληρος κάδος αντί μίας ή δύο σελίδων, όπου πιθανώς μπορεί να είναι αποθηκευμένη η εγγραφή. Έτσι, προκύπτει ότι αν η επεξεργασία του αρχείου γίνεται κυρίως κατά τυχαίο τρόπο, τότε δεν συμφέρει να ομαδοποιούνται πολλές σελίδες σε έναν κάδο. Συνεπώς, τελικά, η εκλογή του κατάλληλου αριθμού σελίδων ανά κάδο είναι ένα πολυδιάστατο πρόβλημα.

Κατ' εξαίρεση όμως προς όσα αναφέρθηκαν προηγουμένως υπάρχουν μερικές περιπτώσεις, όπου δεν εκτελούνται τέτοιου είδους βελτιστοποιήσεις.

Για παράδειγμα, στα αρχεία ISAM της IBM (δες Κεφάλαιο 7) δεν επιτρέπεται μία εγγραφή να είναι αποθηκευμένη σε δύο διαφορετικές σελίδες. Η τεχνική αυτή δεν κάνει βελτιστοποίηση χώρου αλλά διευκολύνει σημαντικά τις εισαγωγές και τις ανανεώσεις των εγγραφών. Σε άλλα συστήματα επιτρέπεται μία εγγραφή να αποθηκευθεί κατά ένα μέρος σε ένα κάδο και κατά το υπόλοιπο σε άλλον. Έτσι το ποσοστό χρησιμοποίησης του χώρου είναι 100%, ωστόσο απαιτείται μία περισσότερο προχωρημένη μέθοδος διαχείρισης της απομονωτικής μνήμης.

Αν και συνήθως αρχικά ένα αρχείο καταλαμβάνει συνεχόμενες θέσεις στο δίσκο, το πιθανότερο είναι ότι λόγω εισαγωγών και διαγραφών οι σελίδες του αρχείου θα διασπαρθούν στις επιφάνειες του δίσκου, ενώ βέβαια το ίδιο συμβαίνει και για τον ελεύθερο χώρο που παύει να είναι ενιαίος αλλά διασκορπίζεται σε μικρά απομακρυσμένα τμήματα. Μία ακόμη υπευθυνότητα του διαχειριστή είναι να γνωρίζει ποιές σελίδες είναι κατειλημμένες και ποιές ελεύθερες. Για το σκοπό αυτό μπορούν να χρησιμοποιηθούν δύο τεχνικές:

- με χρήση μία λίστας για τις διευθύνσεις των ελεύθερων σελίδων, όπως γίνεται με τη συλλογή σκουπιδιών (garbage collection) από ένα λειτουργικό σύστημα, ή
- με χρήση ενός δυαδικού πίνακα, όπου τα 1 και 0 αντιστοιχούν στις κατειλημμένες και στις ελεύθερες σελίδες, αντίστοιχα.

Έτσι, παρ'όλο που μπορεί να υπάρχει γενικά ελεύθερος χώρος, εντούτοις αυτός ο χώρος μπορεί να μην είναι πρακτικά εκμεταλλεύσιμος, όπως για παράδειγμα για το άνοιγμα ενός νέου αρχείου. Αυτό το φαινόμενο λέγεται **τμηματοποίηση** (fragmentation) του δίσκου. Για το σκοπό αυτό τα αρχεία διαιρούνται σε εκτάσεις (extent), πέρα από την υποδιαίρεση τους σε κάδους, σελίδες, εγγραφές και πεδία. Έκταση είναι ένα τμήμα του αρχείου, το οποίο καταλαμβάνει κατ' αποκλειστικότητα ένα συγκεκριμένο μέρος του δίσκου. Συνήθως οι εκτάσεις είναι ισομεγέθεις και ποικίλουν σε πλήθος από μία μέχρι μερικές δεκάδες. Αν και η διαχείριση των εκτάσεων αφ' εαυτής είναι ένα επιπλέον πρόβλημα, εντούτοις με τον τεμαχισμό των αρχείων σε εκτάσεις είναι δυνατό να γίνει καλύτερη χρήση του δίσκου και για δύο ακόμη λόγους:

- σύμφωνα με τις τελευταίες εξελίξεις φέρεται ότι είναι πολύ αποτελεσματικό από άποψη παραλληλισμού τα μεγάλα αρχεία να επιμερίζονται

σε εκτάσεις που αποθηκεύονται σε διαφορετικούς δίσκους. Μάλιστα μερικές φορές κάτι τέτοιο γίνεται και για αρχεία μεσαίου μεγέθους. Η κατανομή αυτή του αρχείου λέγεται **ζεύξη ή σύνδεση** (spanning).

- μία εισαγόμενη εγγραφή μπορεί να μη χωρά σε κάποια έκταση (πιθανότερα την τελευταία). Τότε αυτόματα το σύστημα παραχωρεί στο αρχείο μία επιπλέον έκταση με τρόπο αδιαφανή για το χρήστη. Ακόμη, το σύστημα μπορεί μία άδεια έκταση να τη διαθέσει σε κάποιο άλλο αρχείο.

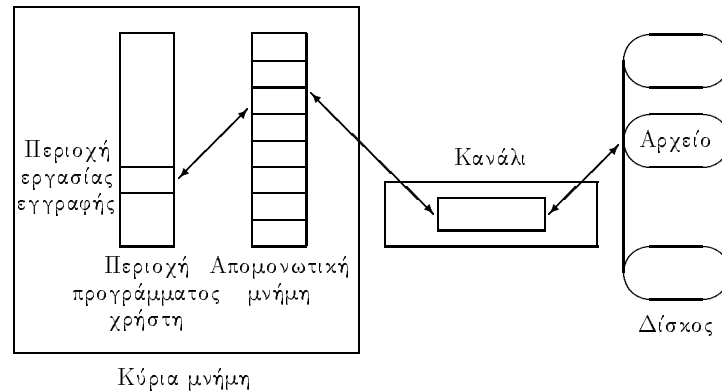
3.6 Διαχείριση απομονωτικής μνήμης

Ο σκοπός των **απομονωτικών μνημών** (buffer memories) είναι η ομαλοποίηση της ροής των δεδομένων μεταξύ της γρήγορης κύριας μνήμης και των βραδύτερων περιφερειακών συσκευών, και γενικά διακρίνονται σε δύο τύπους:

- τις **υλικές** (hardware buffers), που χρησιμοποιούνται ευρύτατα για τη διαχείριση των ιδιαίτερα αργών περιφερειακών συσκευών, και
- τις **λογισμικές** (software buffers), που είναι τμήματα της κύριας μνήμης.

Οι υλικές μνήμες στο παρελθόν χρησιμοποιούνταν στις αναγνωστικές μονάδες χαρτοταινιών και καρτών, ενώ το ίδιο συμβαίνει στους σύγχρονους εκτυπωτές, όπου η μνήμη αυτή βρίσκεται στον ελεγχτή τους. Ακόμη και οι οδηγοί ταινιών και δίσκων έχουν μνήμες που, όμως, είναι πολύ μικρές. Οι μνήμες αυτές δεν θα μας απασχολήσουν στη συνέχεια. Στο Σχήμα 3.11 παρουσιάζεται η ροή των δεδομένων μεταξύ κύριας και δευτερεύουσας μνήμης, όπου εμφανίζεται και η λογισμική απομονωτική μνήμη. Η ύπαρξη των λογισμικών μνημών είναι απαραίτητο συστατικό κάθε συστήματος διαχείρισης βάσεων δεδομένων.

Έστω ότι όλα τα δεδομένα σε ένα σύστημα διαχείρισης βάσεων δεδομένων περιέχονται σε ένα αρχείο και μόνο, το οποίο αποτελείται από 1.000.000 σελίδες (δηλαδή, 4 Gb περίπου θεωρώντας σελίδες των 4 Kb). Προφανώς η κύρια μνήμη είναι πολύ μικρότερη. Έτσι, μέσα στην κύρια μνήμη δεσμεύεται κάποιος χώρος για την απομονωτική μνήμη, ώστε εκεί να τοποθετούνται προσωρινά από το διαχειριστή της απομονωτικής μνήμης τα δεδομένα που



Σχήμα 3.11: Διάγραμμα ροής δεδομένων.

έρχονται από το δίσκο μέχρι να γίνει η επεξεργασία τους. Ο διαχειριστής, λοιπόν, αυτός είναι υπεύθυνος για τον τεμαχισμό της μνήμης σε ένα σύνολο **πλαισίων** (frames), όπου το μέγεθος κάθε πλαισίου είναι ίσο προς το μέγεθος της σελίδας, ενώ το σύνολο των πλαισίων αυτών ονομάζεται **δεξαμενή μνημών** (buffer pool).

Τα ανώτερα στρώματα του συστήματος επικοινωνούν με το διαχειριστή της απομονωτικής μνήμης και:

- του ζητούν κάποια σελίδα, που μπορεί ήδη να βρίσκεται σε κάποιο πλαίσιο, αλλιώς πρέπει να προσπελασθεί από το δίσκο, ή
- του περνούν ένα μήνυμα ότι δεν χρειάζονται πλέον κάποια σελίδα, οπότε το πλαίσιο μπορεί να ελευθερωθεί, ή τέλος
- του περνούν ένα μήνυμα για κάποια ενημέρωση που έχει επέλθει κάποια σελίδα, οπότε η ενημέρωση αυτή πρέπει να περάσει στην αντίστοιχη σελίδα του δίσκου.

Ο διαχειριστής για να ανταποκριθεί στα καθήκοντά του γνωρίζει για κάθε πλαίσιο πόσες φορές έχει ζητηθεί χωρίς να ελευθερωθεί από τότε που η αντίστοιχη σελίδα τοποθετήθηκε στην απομονωτική μνήμη, και αν το πλαίσιο είναι **βρώμικο** (dirty), δηλαδή αν έχει ενημερωθεί. Αυτό επιτυγχάνεται με τη βοήθεια του λεγόμενου μετρητή χαρφωμάτων (pin_count) και του λεγόμενου βρώμικου bit, οι οποίοι αρχικοποιούνται με τιμές 0₁₀ και 0₂.

Αν η σελίδα επόμενης ζήτησης βρίσκεται σε κάποιο πλαίσιο τότε ο μετρητής pin_count αυξάνεται κατά ένα (διαδικασία που λέγεται 'χάρφωμα' -

pinning), ενώ ο μετρητής μειώνεται κατά ένα αν ερθεί μήνυμα ότι το περιεχόμενο κάποιου πλαισίου δεν είναι πλέον χρήσιμο (διαδικασία 'ξεχάρωμα' - unpinning). Αν η αιτούμενη σελίδα δεν βρίσκεται στην απομονωτική μνήμη και δεν υπάρχει κάποιο ελεύθερο πλαίσιο, τότε επιλέγεται ένα πλαίσιο, το λεγόμενο θύμα (victim), όπου θα έρθει η αιτούμενη σελίδα από το δίσκο. Το θύμα επιλέγεται μεταξύ των πλαισίων που έχουν μηδενικό pin_count εφαρμόζοντας μία πολιτική **αντικατάστασης σελίδων** (page replacement) (που θα εξηγηθούν στη συνέχεια). Ωστόσο, πριν ελευθερωθεί ο χώρος του θύματος είναι απαραίτητο να ελεγχθεί το dirty bit. Αν το bit αυτό είναι 0, τότε πράγματι το πλαίσιο ελευθερώνεται χωρίς καμία άλλη ενέργεια, ενώ αν είναι 1 τότε το περιεχόμενο του πλαισίου αποθηκεύεται στην αντίστοιχη σελίδα του δίσκου. Όμως είναι πιθανό να μην υπάρχουν πλαίσια με μηδενικό pin_count. Σε μία τέτοια περίπτωση η νέα αίτηση πρέπει να περιμένει, και πιθανώς να απορριφθεί.

Η προηγούμενη ανάπτυξη του τρόπου λειτουργίας της απομονωτικής μνήμης προσομοιάζει στον τρόπο λειτουργίας της ιδεατής μνήμης (virtual memory) ενός λειτουργικού συστήματος. Ωστόσο, όπως σημειώθηκε στην αρχή του κεφαλαίου, ένα σύστημα διαχείρισης βάσεων δεδομένων δεν επαναπαύεται ούτε σε αυτές τις υπηρεσίες του λειτουργικού συστήματος γιατί το ίδιο μπορεί να προβλέψει τα **πρότυπα ζήτησης σελίδων** (page reference patterns). Όπως θα φανεί στο μάθημα των Βάσεων Δεδομένων, αυτό μπορεί να καταστεί δυνατόν όταν εκτελούνται συγκεκριμένες διεργασίες, όπως σειριακές σαρώσεις αρχείων ή συνδέσεις (join) αρχείων κλπ. Η δυνατότητα πρόβλεψης της μελλοντικής ζήτησης μπορεί να χρησιμοποιηθεί με σκοπό τη βελτίωση της αποτελεσματικότητάς του στα εξής σημεία:

- με την επιλογή του κατάλληλου πλαισίου για το ρόλο του θύματος,
- με την επίλογη της κατάλληλης στιγμής για την αποθήκευση των βρώμικων σελίδων στο δίσκο, και
- με την **προ-προσπέλαση** (prefetching) σελίδων από το δίσκο πριν αχόμη αυτές ζητηθούν.

Σε εφαρμογές βάσεων δεδομένων το κόστος επεξεργασίας των σελίδων από τη στιγμή που θα έρθουν στις απομονωτικές μνήμες θεωρείται αμελητέο σε σχέση με το κόστος μεταφοράς δεδομένων από/προς τη δευτερεύουσα μνήμη. Έτσι, η προ-προσπέλαση βελτιώνει σημαντικά την επίδοση του συστήματος όταν δεν αιτώνται όλες τις σελίδες του δίσκου με ίδια πιθανότητα,

αλλά οι αιτήσεις εστιάζονται κυρίως σε μερικές περιοχές δεδομένων. Το τελευταίο φαινόμενο ονομάζεται **τοπικότητα** (locality), και οδηγεί σε αύξηση του λεγόμενου λόγου επιτυχίας (hit ratio), που είναι το πηλίκο του αριθμού των αιτήσεων που απαντώνται χωρίς προσπέλαση στη δευτερεύουσα μνήμη προς τον αριθμό των συνολικών αιτήσεων. Με άλλα λόγια, η προ-προσπέλαση βελτιώνει την επίδοση όταν οι αιτήσεις αφορούν γειτονικές σελίδες σε περίπτωση σειριακής προσπέλασης, ενώ η επίδοση φθίνει σημαντικά σε περίπτωση τυχαίας προσπέλασης. Επιπλέον, πρέπει να τονισθεί ότι η προ-προσπέλαση συντελεί στη μείωση του μέσου χρόνου προσπέλασης στις σελίδες επειδή, όταν είναι γνωστές οι διευθύνσεις των σελίδων που θα προσπελασθούν, μπορεί να γίνει κάποια αναδιάταξη της σειράς των αιτήσεων με στόχο την ελαχιστοποίηση του χρόνου αναζήτησης και του λανθάνοντα περιστροφικού χρόνου.

Στη συνέχεια εξηγείται μία απλή τεχνική χρήσης της απομονωτικής μνήμης, που ονομάζεται single buffering, και μπορεί να υιοθετηθεί σε περιπτώσεις σειριακής προσπέλασης σειριακού αρχείου (δες Κεφάλαιο 4). Μία εναλλακτική λύση είναι να ακολουθηθεί η επαναλαμβανόμενη διαδικασία να γεμίζει η μνήμη και κατόπιν να ακολουθεί η επεξεργασία. Έτσι, θα μεταφέρεται η επόμενη σελίδα από την περιφερειακή συσκευή στην απομονωτική μνήμη, μόνο όταν τελειώσει η επεξεργασία όλων των εγγραφών της απομονωτικής μνήμης. Ωστόσο, σύμφωνα με μία προχωρημένη μέθοδο μπορεί να γίνει αποτελεσματικότερη χρήση της μνήμης μεταφέροντας δεδομένα ταυτόχρονα από την περιφερειακή συσκευή προς τη μνήμη και από τη μνήμη στην περιοχή του προγράμματος του χρήστη (user program area), όπως φαίνεται στο Σχήμα 3.11. Βέβαια η ίδια τεχνική μπορεί να εφαρμοσθεί για την ταυτόχρονη μεταφορά δεδομένων κατά την αντίστροφη σειρά. Η μέθοδος αυτή μπορεί να υλοποιηθεί με ένα μηχανισμό κυκλικής λίστας. Η διαχείριση της λίστας αυτής γίνεται με τη βοήθεια δύο δεικτών που δείχνουν την πρώτη διαθέσιμη εγγραφή για το πρόγραμμα εφαρμογής και την πρώτη διαθέσιμη θέση για αποθήκευση εγγραφών από τη δευτερεύουσα συσκευή. Έτσι, κατά τη διάρκεια της επεξεργασίας ο ένας δείκτης 'κυνηγά' τον άλλον. Όταν ο πρώτος φθάσει το δεύτερο, τότε η απομονωτική μνήμη είναι άδεια, ενώ στην αντίθετη περίπτωση η μνήμη είναι γεμάτη (όπως στην περίπτωση της κυκλικής ουράς, δες Κεφάλαιο 3.2.3 βιβλίου για Δομές Δεδομένων).

Σε περίπτωση σειριακής προσπέλασης σειριακού αρχείου μπορεί να χρησιμοποιηθεί μία ακόμη αποτελεσματικότερη τεχνική που ονομάζεται double buffering. Για την κατανόηση της τεχνικής, χωρίς βλάβη της γενικότητας ως υποθεθεί ότι η απομονωτική μνήμη αποτελείται από δύο πλαίσια μόνο.

Έστω ότι αρχικά αυτά τα δύο πλαίσια γεμίζουν με τις πρώτες δύο σελίδες του αρχείου. Μόλις τελειώσει η επεξεργασία των εγγραφών του πρώτου πλαισίου, τότε αυτό με νέα ανάγνωση ξαναγεμίζει με το περιεχόμενο της τρίτης σελίδας του αρχείου. Έτσι, η τρίτη σελίδα προσφέρεται για επεξεργασία αμέσως μόλις τελειώσει η επεξεργασία της δεύτερης σελίδας που βρίσκεται στο δεύτερο πλαίσιο. Με τον τρόπο αυτό, σε μία χρονική στιγμή το ένα πλαίσιο δέχεται δεδομένα από το δίσκο και το άλλο συμμετέχει στην επεξεργασία, ενώ στην επόμενη χρονική στιγμή οι ρόλοι των πλαισίων αντιστρέφονται. Η τεχνική αυτή αλλαγής των ρόλων των δύο μνημών ονομάζεται **ανταλλαγή απομονωτικών μνημών** (buffer swapping). Αν ο χρόνος επεξεργασίας των εγγραφών της σελίδας (χρόνος CPU) δεν είναι μεγαλύτερος από το χρόνο προσπέλασης (χρόνος I/O), τότε βάσιμα μπορεί να θεωρηθεί ότι ο συνολικός χρόνος ταυτίζεται με το χρόνο προσπέλασης. Στο Σχήμα 3.12 παρουσιάζεται αφ' ενός μία σειρά από λειτουργίες εισόδου/εξόδου με τις χρονικά αντίστοιχες λειτουργίες επεξεργασίας και αφ' ετέρου το περιεχόμενο των δύο πλαισίων. Κατά τον ίδιο τρόπο τα δύο πλαίσια μπορούν να συνεργαστούν για τη σειριακή αποθήκευση στο αρχείο.

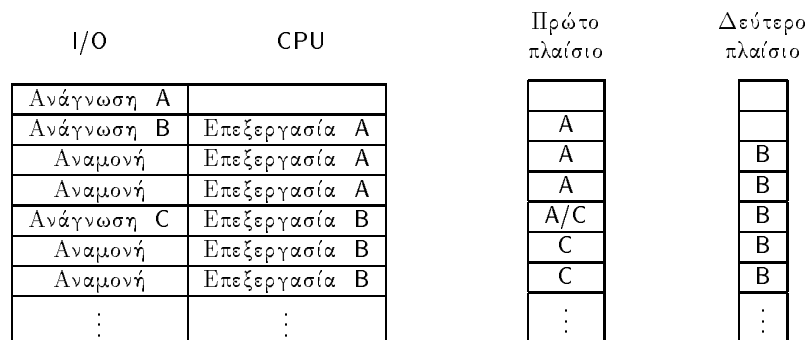
I/O	CPU	Πρώτο πλαίσιο	Δεύτερο πλαίσιο
Ανάγνωση A			
Ανάγνωση B	Επεξεργασία A	A	
Ανάγνωση C	Επεξεργασία B	A/C	B
Ανάγνωση D	Επεξεργασία C	C	B/D
⋮	⋮	⋮	⋮

Σχήμα 3.12: Χρονική αλληλουχία εργασιών με χρήση δύο πλαισίων.

Αν οι υπολογισμοί είναι πολύ σύνθετοι και χρονοβόροι, τότε η χρήση των δύο πλαισίων δεν έχει τόσο θεαματικά αποτελέσματα. Στο Σχήμα 3.13 παρουσιάζεται μία τέτοια περίπτωση, όπου ο χρόνος CPU είναι τριπλάσιος από το χρόνο εισόδου/εξόδου. Αν οι απομονωτικές μνήμες έχουν μέγεθος ίσο με μία άτρακτο, τότε ο χρόνος που απαιτείται για τη σειριακή επεξεργασία του αρχείου είναι:

$$[t] \times b \times ebt$$

όπου το t είναι ο λόγος του χρόνου CPU προς το χρόνο εισόδου/εξόδου. Γενικά μεγαλύτερος παραλληλισμός μπορεί να επιτευχθεί με τη χρήση πολλαπλών απομονωτικών μνημών (multiple buffering). Αλλά και πάλι αν ο



Σχήμα 3.13: Χρονική αλληλουχία εργασιών με χρήση δύο πλαισίων. Ο χρόνος CPU είναι τριπλάσιος από το χρόνο I/O.

χρόνος επεξεργασίας είναι μεγαλύτερος από το χρόνο εισόδου/εξόδου, τότε η τεχνική του double buffering είναι επαρκής.

Όπως αναφέρθηκε προηγουμένως, οι τεχνικές του single buffering και double buffering μπορούν να εφαρμοσθούν με επιτυχία σε περιπτώσεις σειριακής προσπέλασης σειριακού αρχείου. Ωστόσο, σε ένα σύστημα διαχείρισης βάσεων δεδομένων μπορεί ταυτόχρονα πολλοί χρήστες να προσπελαύνουν τη βάση, οπότε η απομονωτική μνήμη να περιέχει σελίδες από πολλά αρχεία. Μία τέτοια γενική περίπτωση δεν αντιμετωπίζεται με τις προηγούμενες τεχνικές, αλλά απαιτείται η εφαρμογή μίας συγκεκριμένης πολιτικής αντικατάστασης σελίδων για την επιλογή του θύματος.

Η περισσότερο γνωστή πολιτική αντικατάστασης σελίδων είναι η επιλογή της λιγότερο πρόσφατα χρησιμοποιημένης (least recently used, LRU). Η πολιτική αυτή υλοποιείται με τη βοήθεια μίας ουράς, στο τέλος της οποίας τοποθετούνται οι διευθύνσεις των πλαισίων που έχουν pin_count ίσο με 0. Έτσι, ως θύμα επιλέγεται το πλαίσιο που βρίσκεται στην κεφαλή της ουράς.

Μία παραλλαγή της προηγούμενης πολιτικής είναι η ωρολογιακή (clock) επιλογή, που δεν διαχειρίζεται κάποια ουρά αλλά στηρίζεται σε μία μεταβλητή (current), που θεωρεί κυκλικά όλες τα πλαίσια της απομονωτικής μνήμης. Έτσι, αν το πλαίσιο που υποδεικνύεται από την current έχει pin_count με τιμή μεγαλύτερη του 0, τότε η current αυξάνεται κατά ένα, μέχρι να βρεί πλαίσιο με τιμή pin_count ίση με 0.

Οι δύο αυτές πολιτικές δεν είναι αποτελεσματικές σε περίπτωση σειριακής επεξεργασίας του αρχείου. Για παράδειγμα, έστω ότι η απομονωτική

μνήμη έχει 10 πλαίσια, και ότι ένα αρχείο αποτελείται από 10 σελίδες. Αυτή η περίπτωση εξυπηρετείται από τις πολιτικές αυτές χωρίς κανένα πρόβλημα, γιατί όταν ξαναζητηθεί κάποια σελίδα είναι βέβαιο ότι δεν θα γίνει προσπέλαση στο δίσκο. Όμως αν το αρχείο αποτελείται από 11 σελίδες, τότε κατά το δεύτερο, τρίτο κλπ. πέρασμα του αρχείου θα προσπελαύνονται και πάλι όλες οι σελίδες, δηλαδή ο λόγος επιτυχίας θα είναι 0.

Άλλες χρησιμοποιούμενες πολιτικές αντικατάστασης σελίδων είναι η τυχαία (random), η περισσότερο πρόσφατα χρησιμοποιημένη (most recently used, MRU), πρώτη ερχόμενη, πρώτη αντικαθιστώμενη (first in, first out). Ο τρόπος λειτουργίας των πολιτικών αυτών είναι ευνόητος από τις ονομασίες τους και από την προηγούμενη ανάπτυξη. Στις περισσότερες περιπτώσεις χρησιμοποιείται κάποια παραλλαγή των δύο πρώτων πολιτικών ώστε να αντιμετωπίζεται αποτελεσματικά η σειριακή επεξεργασία.

3.7 Ασκήσεις

<1> Να βρεθεί ο βέλτιστος παράγοντας ομαδοποίησης αν το μέγεθος της σελίδας είναι 256 bytes, το μέγεθος του κάδου είναι από 2560-3840 bytes και το μήκος της εγγραφής είναι 64, 72, 91, 240 ή 400 bytes.

<2> Έστω ότι σε μία συσκευή δίσκου το μέγεθος μίας ατράχτου είναι 10.000 bytes και ότι απαιτούνται 10 ms για μία πλήρη περιστροφή. Επίσης, ας υποθεθεί ότι το μέγεθος των σελίδων είναι 1000 bytes, ενώ το μέγεθος των κενών λόγω φορμαρίσματος είναι 100 bytes. Να υπολογισθεί η μέγιστη (στιγμιαία) ταχύτητα μεταφοράς δεδομένων, καθώς και η πραγματική ταχύτητα μεταφοράς δεδομένων σε bytes/άτρακτο.

<3> Δίνεται αρχείο με 30.000 εγγραφές των 100 bytes. Κάθε μήνα γίνονται 1000 τυχαίες προσπελάσεις για μία εγγραφή και μία σειριακή προσπέλαση όλου του αρχείου κατά κάδους. Αν στην κύρια μνήμη υπάρχει ένας πίνακας με τις διευθύνσεις των κάδων που περιέχουν τις εγγραφές, τότε ποιο είναι το βέλτιστο μέγεθος του κάδου και πόσο είναι το αντίστοιχο κόστος για τις λειτουργίες αυτές; Να βρεθούν οι αντίστοιχες τιμές όταν σε μία σειριακή ανάγνωση αντιστοιχούν 10.000 τυχαίες προσπελάσεις.

<4> Έστω ότι το προηγούμενο αρχείο είναι αποθηκευμένο σε δίσκο IBM 3380. Ποιο είναι το κόστος για την ανάγνωση ολόκληρου του αρχείου κατά εγγραφές και κατά σελίδες όταν ο κάδος αποτελείται από 9600, 19.200 και 48.000 bytes.

<5> Να βρεθεί ο χρόνος που απαιτείται από μία συσκευή IBM 3380, για μία τροποποίηση αρχείου που απαρτίζεται από 10.000 σελίδες των 2.400 bytes, αν ο κάδος αποτελείται από 1, 2 ή 10 σελίδες και τέλος αν ο κάδος έχει το μέγεθος ενός τομέα.

<6> Έστω ότι μία συσκευή δίσκων CD πρόκειται με τη βοήθεια μίας διεπιφάνειας να συνδεθεί με ένα μικροϋπολογιστή για αποθήκευση αρχείων. Ένας δίσκος CD αποθηκεύει 60 λεπτά μουσικής, ενώ ο κατάλογος του μπορεί να περιέχει μέχρι 99 εισόδους. Η ταχύτητα περιστροφής είναι 500 στροφές/λεπτό και η ταχύτητα μεταφοράς δεδομένων είναι 100 bytes/ms. Η κεφαλή ανάγνωσης/αποθήκευσης κινείται με γραμμική ταχύτητα και απαιτούνται 3 δευτερόλεπτα για να σαρωθεί όλη η επιφάνεια. Να βρεθούν οι υπόλοιποι παράγοντες που χαρακτηρίζουν το δίσκο.

Κεφάλαιο 4

ΣΕΙΡΙΑΚΑ ΑΡΧΕΙΑ

4.1 Εισαγωγή

4.2 Μη ταξινομημένα σειριακά αρχεία

4.3 Ταξινομημένα σειριακά αρχεία

4.4 Ασκήσεις

Κεφάλαιο 4

ΣΕΙΡΙΑΚΑ ΑΡΧΕΙΑ

4.1 Εισαγωγή

Τα **σειριακά** (sequential) αρχεία είναι ο παλαιότερος τύπος οργάνωσης αρχείων. Αναπτύχθηκε γύρω στα 1950 σε συνδυασμό με την ύπαρξη των μαγνητικών ταινιών, που αποτελούσαν τότε την κυρίαρχη συσκευή δευτερεύουσας αποθήκευσης. Ακόμη και σήμερα τα σειριακά αρχεία χρησιμοποιούνται στην πράξη στα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων, όχι όμως σε ταινίες, αλλά σε μαγνητικούς δίσκους. Στα αρχεία αυτά η φυσική σειρά είναι ίδια με τη λογική σειρά, οπότε η επεξεργασία προχωρά από την αρχή προς το τέλος. Δηλαδή, η i -οστή εγγραφή είναι αναγνώσιμη μόνο αφού προηγηθεί η ανάγνωση των $i-1$ προηγούμενων. Οι εγγραφές σε ένα σειριακό αρχείο μπορεί να αποθηκεύονται είτε αταξινόμητες είτε ταξινομημένες κατά αύξουσα ή φθίνουσα τιμή του κλειδιού.

Οι στοιχειώδεις λειτουργίες που γίνονται στα σειριακά αρχεία, όπως εξ άλλου και σε όλα τα αρχεία που θα εξετασθούν στη συνέχεια, δίνονται στον επόμενο πίνακα. Έχοντας μία τιμή για το χρονικό κόστος αυτών των πράξεων για κάθε είδους υλοποίηση αρχείου, είναι δυνατό να γίνει εκτίμηση της επίδοσης μίας βάσης δεδομένων.

Κάθε αρχείο αρχίζει με την **επικεφαλίδα** (file header) που περιέχει κάποιες πληροφορίες, όπως αν το αρχείο είναι δεκαεξαδικό, κινητής υποδιαστολής ή ASCII, το μέγεθος του αρχείου, το μήκος και τον αριθμό των εγγραφών, τη διεύθυνση της τελευταίας εγγραφής του αρχείου, αν υπάρχουν κατάλογοι και τι είδους, τις διευθύνσεις των εκτάσεων κλπ. Όταν

Σύμβολο	Ορισμός
$T_{\text{προσ}}$	Χρόνος προσπέλασης εγγραφής
$T_{\text{επομ}}$	Χρόνος προσπέλασης επόμενης εγγραφής
$T_{\text{εισ}}$	Χρόνος εισαγωγής εγγραφής
$T_{\text{αναν}}$	Χρόνος ανανέωσης εγγραφής
$T_{\text{διαγ}}$	Χρόνος διαγραφής εγγραφής
$T_{\text{εξαν}}$	Χρόνος εξαντλητικής ανάγνωσης αρχείου
$T_{\text{αναδ}}$	Χρόνος αναδιοργάνωσης αρχείου

Πίνακας 4.1: Στοιχειώδεις λειτουργίες σε αρχεία.

ανοίγει ένα αρχείο, τότε όλες αυτές οι πληροφορίες έρχονται στην κύρια μνήμη και βοηθούν στο χειρισμό του. Κατά το κλείσιμο του αρχείου οι νέες πληροφορίες επανεγγράφονται στον αντίστοιχο χώρο στο δίσκο. Για πολύ μικρά αρχεία η επικεφαλίδα μπορεί να καταλαμβάνει συγκρίσιμο χώρο με τα κυρίως δεδομένα, ωστόσο αυτή η περίπτωση δεν θα εξετασθεί. Σημειώνεται ότι αυτές οι πληροφορίες (όπως και άλλες) σχετικά με το αρχείο βρίσκονται στο **λεξικό δεδομένων** (data dictionary) των σχεσιακών βάσεων δεδομένων. Στη συνέχεια υποτίθεται ότι τα αρχεία είναι μεγάλα, οπότε ο χώρος που καταλαμβάνει η επικεφαλίδα καθώς και ο χρόνος επεξεργασίας της θα αγνοηθούν, γιατί αυτές οι παράμετροι είναι κοινές για όλα τα αρχεία.

Στο σημείο αυτό δίνεται ορισμός του **καταλόγου** (index). Ο κατάλογος είναι μία δομή που συνδέει τις τιμές ενός πεδίου (ή πεδίων) με τη διεύθυνση των εγγραφών που περιέχουν τις τιμές. Στη συνέχεια υποτίθεται ότι τα σειριακά αρχεία δεν συνοδεύονται από κατάλογο και συνεπώς δεν αξιοποιείται η δυνατότητα του υλικού για άμεση προσπέλαση. Τα σειριακά αρχεία, όπως αναφέρθηκε, διακρίνονται σε αταξινομήτα και ταξινομημένα κατά αύξουσα ή φθίνουσα τάξη του κλειδιού.

4.2 Μη ταξινομημένα σειριακά αρχεία

Σε ένα **μη ταξινομημένο** σειριακό αρχείο ή αρχείο σωρού (pile file) οι εγγραφές τοποθετούνται η μία μετά την άλλη ανάλογα με τη σειρά άφιξης. Κάλλιστα, οι εγγραφές μπορεί να ποικίλουν σε μήκος. Για παράδειγμα, οι εγγραφές σε ένα νοσοκομειακό ή αστυνομικό πληροφοριακό σύστημα εγγράφονται ως ένα αρχείο χειμένου. Στη συνέχεια όμως θα υποθεθεί ότι οι εγγραφές είναι σταθερού μήκους για να απλοποιηθούν οι υπολογισμοί.

Ακόμη και στην περίπτωση που το αρχείο έχει κάποια δομή, ώστε να επιταχύνεται η άμεση αναζήτηση, αν η αναζήτηση γίνεται με βάση ένα δευτερεύον κλειδί, τότε και πάλι θα πρέπει να θεωρηθεί ως ένα αρχείο σωρού. Στο κεφάλαιο αυτό δίνεται έμφαση όχι γιατί τα αρχεία σωρού είναι τόσο σημαντικά αλλά γιατί θα ακολουθηθεί σταθερά η ίδια μεθοδολογία και στα επόμενα κεφάλαια.

4.2.1 Προσπέλαση εγγραφής

Αν η διεύθυνση της εγγραφής είναι γνωστή, τότε το χρονικό κόστος για την προσπέλαση της είναι:

$$T_{\text{προσ}} = s + r + btt$$

Όμως στο αρχείο σωρού δεν είναι γνωστή η διεύθυνση της εγγραφής. Άρα αν δοθεί η τιμή κάποιου ή κάποιων πεδίων, τότε η αναζήτηση πρέπει να γίνει σειριακά και ελέγχοντας αν η τιμή του κλειδιού της εγγραφής ταυτίζεται με τη δεδομένη τιμή. Προκύπτει, λοιπόν, ότι ο μέσος όρος των σελίδων που θα μεταφερθούν στην κύρια μνήμη μέχρι να βρεθεί η κατάλληλη εγγραφή είναι:

$$\frac{1}{b} \times \sum_{i=1}^b i = \frac{1}{b} \times \frac{b \times (b+1)}{2} \approx \frac{b}{2}$$

Άρα, ο αναμενόμενος χρόνος για την αναζήτηση μίας εγγραφής σε αρχείο σωρού είναι:

$$T_{\text{προσ}} = ebt \times \frac{b}{2}$$

Στον τύπο αυτό εισάγεται ο πραγματικός χρόνος μεταφοράς, γιατί τελικά πρόκειται για μία σειριακή αναζήτηση στο μισό ενός μεγάλου αρχείου, κατά μέσο όρο.

4.2.2 Προσπέλαση επόμενης εγγραφής

Εφ' όσον οι εγγραφές είναι αταξινόμητες, έπεται ότι η αναζήτηση της λογικά επόμενης εγγραφής (δηλαδή, με βάση την τιμή κάποιου κλειδιού) δεν είναι παρά αναζήτηση μίας τυχαίας εγγραφής. Αυτό συμβαίνει γιατί υπάρχει μικρή τοπικότητα, δηλαδή μικρή πιθανότητα δύο διαδοχικές λογικές εγγραφές να βρίσκονται σε γειτονικές περιοχές στο δίσκο. Συνεπώς ισχύει:

$$T_{\text{επομ}} = T_{\text{προσ}}$$

Δηλαδή, αν οι εγγραφές είναι αταξινομήτες και δοθεί για αναζήτηση μία λίστα τιμών (ίσως και ταξινομημένων) σε κάποιο πεδίο, τότε ο χρόνος αναζήτησης της επόμενης εγγραφής είναι ίδιος με το χρόνο αναζήτησης της πρώτης εγγραφής της λίστας.

4.2.3 Εξαντλητική ανάγνωση αρχείου

Αν αρκεί η ανάγνωση όλων των εγγραφών χωρίς απαραίτητα να δοθούν και ταξινομημένες, τότε ο απαιτούμενος χρόνος είναι:

$$T_{\text{εξαν}} = b \times ebt$$

Συνεπώς, ο απαιτούμενος χρόνος για την ανάγνωση ολόκληρου του αρχείου είναι διπλάσιος από το μέσο απαιτούμενο χρόνο για την προσπέλαση μίας μόνο εγγραφής. Επομένως, το αρχείο σωρού είναι πολύ καλή επιλογή για εφαρμογές όπου χρειάζεται απλά μία σάρωση του αρχείου, γιατί δεν απαιτείται επιπλέον δόμηση των δεδομένων. Σημειώνεται ότι αυτή η δόμηση εκτός του ότι καταναλώνει επιπλέον χώρο, αυξάνει και την πολυπλοκότητα. Για παράδειγμα, έστω ότι πρέπει να υπολογισθεί ο μέσος όρος των τιμών ενός πεδίου. Για τον υπολογισμό του τρέχοντος μέσου όρου κρατούνται δύο τιμές: ο αριθμός των εγγραφών (i) που έχουν αναγνωσθεί μέχρι τώρα και ο μέσος όρος των τιμών των πεδίων εγγραφών (A) που έχουν αναγνωσθεί μέχρι τώρα. Έτσι ισχύει:

$$New_mean_value = A \times \frac{i}{i+1} + \frac{new_term}{i+1}$$

Έστω ότι υπάρχει μία λίστα τιμών ενός πεδίου. Αν όλες οι εγγραφές πρέπει να δοθούν στο χρήστη ταξινομημένες με βάση τις τιμές αυτού του πεδίου, τότε κάθε εγγραφή θα πρέπει να αναζητηθεί ανεξάρτητα από τις άλλες. Δηλαδή ισχύει:

$$T_{\text{εξαν}} = n \times T_{\text{προσ}} = n \times ebt \times \frac{b}{2} = \frac{ebt \times n^2}{2 \times Bfr}$$

Από τον τύπο αυτό φαίνεται ότι η εξαντλητική ανάγνωση με βάση τις ταξινομημένες τιμές ενός πεδίου είναι μία διαδικασία με πολυπλοκότητα $O(n^2)$ και συνεπώς πολύ αναποτελεσματική, ιδιαίτερα μάλιστα όταν το n είναι πολύ μεγάλο.

4.2.4 Εισαγωγή εγγραφής

Αν υποθεθεί ότι δεν γίνεται έλεγχος για διπλοεγγραφές, τότε κάθε νέα εγγραφή προσαρτάται στο τέλος του αρχείου χωρίς κάποια άλλη επιπλέον ενέργεια. Η διεύθυνση της τελευταίας σελίδας βρίσκεται διαθέσιμη στην κύρια μνήμη τη στιγμή της εισαγωγής, επειδή είναι αποθηκευμένη στην επικεφαλίδα. Άρα, το χρονικό κόστος για την εισαγωγή μίας εγγραφής είναι:

$$T_{\text{εισ}} = s + 3r + btt$$

Το κόστος αυτό εξηγείται ως εξής. Ο χρόνος για τον εντοπισμό της τελευταίας σελίδας είναι $(s+r+btt)$, ο χρόνος για την απαραίτητη περιστροφή ώστε να επανέλθει η συγκεκριμένη σελίδα κάτω από την κεφαλή είναι $(2r-btt)$ και τέλος ο χρόνος για την επανα-αποθήκευση της σελίδας στο δίσκο είναι btt .

Είναι δυνατό με μία οριακή εισαγωγή να γεμίσει η συγκεκριμένη έκταση, οπότε το λειτουργικό σύστημα αποδίδει στο αρχείο μία νέα έκταση. Στην περίπτωση αυτή το κόστος αλλάζει.

4.2.5 Διαγραφή εγγραφής

Προφανώς για να γίνει η διαγραφή, πρέπει πρώτα να γίνει η αναζήτηση με κόστος $T_{\text{προσ}}$. Όταν η κατάλληλη σελίδα έρθει στην κύρια μνήμη, τότε η εγγραφή μαρκάρεται και θεωρείται πλέον διαγραμμένη. Η περιστροφή του δίσκου συνεχίζεται και συνεπώς απαιτείται χρόνος ίσος με $(2r-btt)$, ώστε να επανέλθει η συγκεκριμένη σελίδα κάτω από την κεφαλή. Επιπλέον απαιτείται χρόνος ίσος με btt ώστε να επανα-αποθηκευθεί η σελίδα στη θέση όπου βρισκόταν. Επομένως το χρονικό κόστος για μία διαγραφή είναι:

$$T_{\text{διαγ}} = T_{\text{προσ}} + 2r$$

Βέβαια, ο όρος $2r$ είναι πολύ μικρός σε σχέση με το χρονικό κόστος της αναζήτησης.

4.2.6 Ανανέωση εγγραφής

Για όλες τις μέχρι τώρα λειτουργίες το χρονικό κόστος είναι ίδιο είτε οι εγγραφές είναι σταθερού είτε μεταβλητού μήκους. Στην περίπτωση της ανανέωσης το μήκος της εγγραφής έχει πλέον σημασία, γιατί η νέα μορφή της

εγγραφής μπορεί να μην χωρά στη θέση, όπου ήταν αποθηκευμένη η παλιά μορφή της εγγραφής. Το χρονικό κόστος για την ανανέωση μίας εγγραφής σταθερού μήκους είναι:

$$T_{\text{αναν}} = T_{\text{προσ}} + 2r$$

Παρατηρείται ότι το κόστος είναι ίδιο με το κόστος της διαγραφής.

Αν το αρχείο περιέχει εγγραφές μεταβλητού μήκους, τότε η ανανέωση αντιμετωπίζεται ως μία διαγραφή και μία εισαγωγή. Αυτό οφείλεται στο γεγονός ότι είναι πολύ χρονοβόρο να μετακινηθούν όλες οι εγγραφές, ώστε να δημιουργηθεί χώρος για την νέα εκδοχή της εγγραφής. Έτσι ισχύει:

$$T_{\text{αναν}} = T_{\text{διαγ}} + T_{\text{εισ}}$$

Και στην περίπτωση αυτή το συνολικό κόστος κυριαρχείται από το κόστος της αναζήτησης.

4.2.7 Αναδιοργάνωση αρχείου

Το αρχείο αποδιοργανώνεται μετά από τις συνεχείς εισαγωγές και διαγραφές. Φυσική συνέπεια είναι η επίδοση του αρχείου να φθίνει για τους εξής λόγους:

- οι διαγραμμένες εγγραφές είναι μαρκαρισμένες αλλά καταλαμβάνουν το συγκεκριμένο χώρο. Έτσι, το μέγεθος του αρχείου (δηλαδή, ο αριθμός των σελίδων b) μεγαλώνει, ενώ αντίθετα η πραγματική χωρητικότητα των σελίδων μειώνεται, καθώς κενός χώρος διασκορπίζεται σε όλες τις σελίδες του αρχείου.
- οι εισαγωγές έχουν κατευθυνθεί προς το τέλος του αρχείου και κατά συνέπεια το αρχείο δυνητικά διαμοιράζεται σε διάφορες εκτάσεις. Έτσι, ο χρόνος εντοπισμού αυξάνεται.

Αναδιοργάνωση του αρχείου σημαίνει ότι πρέπει να αναγνωσθεί όλο το αρχείο και να επανα-αποθηκευθεί με τη νέα του μορφή. Αν το νέο αρχείο επανα-αποθηκευθεί επάνω από το παλιό, τότε υπάρχει ο κίνδυνος να χαθούν δεδομένα, αν το σύστημα πέσει κατά την διάρκεια της αναδιοργάνωσης. Για το λόγο αυτό το νέο αρχείο αποθηκεύεται σε νέα έκταση, ενώ η παλιά απελευθερώνεται και αποδίδεται ξανά στο σύστημα μετά το τέλος της αναδιοργάνωσης.

Επειδή πρέπει να αποφεύγεται η συνεχής μηχανική μετακίνηση του βραχίονα στις διάφορες εκτάσεις του νέου και του παλιού αρχείου, χρησιμοποιείται η τεχνική της διπλής απομονωτικής μνήμης. Δηλαδή, στην κύρια μνήμη μεταφέρονται οι σελίδες του παλιού αρχείου και ο χώρος των σελίδων ταχτοποιείται αγνοώντας τις διαγραμμένες εγγραφές. Κάθε φορά που η απομονωτική μνήμη γεμίζει, το περιεχόμενό της μεταφέρεται στο δίσκο. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να εξαντληθεί το παλιό αρχείο.

Το κόστος, λοιπόν, της αναδιοργάνωσης με μία συσκευή δίσκου είναι:

$$T_{\text{ανδ}} = b \times ebt + \left\lceil \frac{n}{Bfr} \right\rceil \times ebt$$

όπου b είναι οι σελίδες του παλιού αρχείου και n είναι οι εγγραφές του νέου αρχείου. Αν το σύστημα έχει δύο συσκευές δίσκου, τότε η αναδιοργάνωση γίνεται ταχύτερα, γιατί χρησιμοποιούνται τεχνικές παραλληλισμού.

Ανακεφαλαιώνοντας τα συμπεράσματα του κεφαλαίου αυτού, αναφέρεται και πάλι ότι το αρχείο σωρού είναι:

- οικονομικό από άποψη χώρου, αν δεν έχουν γίνει στο μεταξύ πολλές διαγραφές,
- βολικό στο χειρισμό των εισαγωγών,
- αποτελεσματικό για την εξαντλητική σάρωση του αρχείου όταν δεν χρειάζεται οι εγγραφές να δοθούν στο χρήστη ταξινομημένες με βάση τις τιμές κάποιου πεδίου,
- αργό για την αναζήτηση μίας εγγραφής ή της επόμενης με βάση την τιμή κάποιου κλειδιού, και τέλος
- τελείως αναποτελεσματικό αν πρόκειται να δοθούν οι εγγραφές στο χρήστη ταξινομημένες ως προς κάποιο πεδίο.

Τελικά, τα αρχεία σωρού χρησιμοποιούνται:

- σε στατιστικές επεξεργασίες, όπως για παράδειγμα για την εύρεση του μέσου όρου των τιμών κάποιου πεδίου,
- σε περίπτωση διαχωρισμού του αρχείου σε υποαρχεία με βάση κάποιο κριτήριο, όπως με βάση το φύλο, το μισθό κλπ. και με την προϋπόθεση ότι δεν υπάρχει κάποιος κατάλογος, και
- όταν το αρχείο δεν είναι ιδιαίτερα μεγάλο και ιδιαίτερα σε εφαρμογές με μικροϋπολογιστές.

4.3 Ταξινομημένα σειριακά αρχεία

Σε ένα ταξινομημένο αρχείο οι εγγραφές είναι διατεταγμένες ως προς τις τιμές ενός πεδίου ή συνδυασμού κάποιων πεδίων. Για παράδειγμα, συνήθως ένα αρχείο με διευθύνσεις είναι ταξινομημένο ως προς τον ταχυδρομικό κωδικό, ενώ ένα αρχείο πελατών είναι ταξινομημένο ως προς το ονοματεπώνυμο. Όμως αν το επίθετο και το όνομα αποτελούν διαφορετικά πεδία, τότε για το αρχείο πελατών η διάταξη των εγγραφών πρέπει να γίνει ως προς το συνδυασμό των δύο αυτών πεδίων.

Το ταξινομημένο σειριακό αρχείο μπορεί να παραμείνει ταξινομημένο κατά την εισαγωγή νέων εγγραφών, μόνο αν κάθε εισαγωγή συνοδεύεται και από μετατόπιση των εγγραφών με μεγαλύτερο κλειδί κατά μία θέση. Αυτή η διαδικασία είναι εξαιρετικά χρονοβόρα και για το λόγο αυτό δεν εφαρμόζεται ποτέ στην πράξη. Αντίθετα, το αρχείο αυτό παραμένει ανέπαφο, ενώ δημιουργείται ένα νέο αρχείο που περιέχει μόνο τις εισαγωγές. Το αρχείο αυτό ονομάζεται **περιοχή υπερχείλισης** (overflow area) και δεν είναι ταξινομημένο.

Όταν αναζητείται μία εγγραφή, τότε η επεξεργασία αρχίζει πρώτα από την **κύρια περιοχή** (main area) του ταξινομημένου αρχείου. Σε περίπτωση που η εγγραφή δεν είναι αποθηκευμένη στην κύρια περιοχή, τότε η αναζήτηση συνεχίζεται στην περιοχή υπερχείλισης. Αν η περιοχή υπερχείλισης περιέχει μόνο λίγες εγγραφές, τότε το κόστος αναζήτησης πρακτικά ισούται με το κόστος αναζήτησης στην κύρια περιοχή. Όσο όμως εισάγονται νέες εγγραφές, τόσο η περιοχή υπερχείλισης μεγαλώνει, καθώς και το αντίστοιχο κόστος αναζήτησης. Για το λόγο αυτό πρέπει περιοδικά το αρχείο να αναδιοργανώνεται, ώστε και πάλι να αποκτή τα πλεονεκτήματα του ταξινομημένου αρχείου.

Όπως και στο Κεφάλαιο 4.2, ακολουθεί η αναλυτική κοστολόγηση των στοιχειωδών πράξεων. Στη συνέχεια θεωρείται ότι οι εγγραφές είναι σταθερού μήκους.

4.3.1 Προσπέλαση εγγραφής

Έστω ότι η κύρια περιοχή και η περιοχή υπερχείλισης αποτελούνται αντίστοιχα από b_m και b_o σελίδες, επομένως το αρχείο αποτελείται συνολικά από $b = b_m + b_o$ σελίδες. Όταν δίνεται η τιμή ενός κλειδιού, τότε η αντίστοιχη εγγραφή μπορεί να εντοπισθεί με **δυαδική αναζήτηση** (binary search).

Δηλαδή, η πρώτη προσπέλαση γίνεται στη μεσαία σελίδα της κύριας περιοχής. Αν η τιμή του ζητούμενου κλειδιού είναι μικρότερη από τα κλειδιά της σελίδας, τότε στη συνέχεια προσπελάζεται η μεσαία σελίδα του πρώτου μισού αυτού του τμήματος του αρχείου, ενώ αντίστοιχα αν η τιμή του κλειδιού είναι μεγαλύτερη από τα κλειδιά της σελίδας, τότε προσπελάζεται η μεσαία σελίδα του δεύτερου μισού του τμήματος αυτού. Η διχοτόμηση της κύριας περιοχής συνεχίζεται μέχρι να εντοπισθεί το κλειδί σε κάποια σελίδα. Αν τελικά το κλειδί δεν εντοπισθεί, τότε η αναζήτηση συνεχίζεται κατά σειριακό τρόπο στην περιοχή υπερχείλισης. Παραδείγματα επιτυχούς και ανεπιτυχούς αναζήτησης παρουσιάζονται στο Σχήμα 4.1 και στο Σχήμα 4.2, αντίστοιχα, όπου ισχύει $b=b_m=8$.

4	12	56	100	146	193	220	250
5	19	71	113	150	202	225	255
10	52	90	142	175	215	232	278

\uparrow \uparrow
 2η 1η

Σχήμα 4.1: Προσπελάσεις σε επιτυχή δυαδική αναζήτηση του 19.

4	12	56	100	146	193	220	250
5	19	71	113	150	202	225	255
10	52	90	142	175	215	232	278

\uparrow \uparrow \uparrow \uparrow
 1η 2η 3η 4η

Σχήμα 4.2: Προσπελάσεις σε ανεπιτυχή δυαδική αναζήτηση του 280.

Αν η αναζητούμενη εγγραφή είναι αποθηκευμένη στην κύρια περιοχή, τότε εύκολα αποδεικνύεται ότι η μέση τιμή του αριθμού των τυχαίων προσπελάσεων για μία επιτυχή αναζήτηση είναι:

$$\frac{1}{b_m} \times 1 + \frac{2}{b_m} \times 2 + \frac{4}{b_m} \times 3 + \dots + \frac{1}{2} \times \log b_m = \log b_m - 1$$

Συνεπώς, ο απαιτούμενος χρόνος για τη δυαδική επιτυχή αναζήτηση μίας εγγραφής στην κύρια περιοχή είναι:

$$(\log b_m - 1) \times (s + r + btt)$$

Σημειώνεται ότι στην πρώτη προσπέλαση ο χρόνος εντοπισμού, s , θα βαρύνει περισσότερο, στη δεύτερη θα βαρύνει λιγότερο, γιατί ο εντοπισμός θα

γίνει στο μισό αρχείο κοχ. Από την άλλη πλευρά, ο χρόνος περιστροφής είναι σημαντικός και σταθερός κατά τις διαδοχικές προσπελάσεις. Τελικά, η τελευταία έκφραση είναι πεσιμιστική.

Αν η εγγραφή είναι αποθηκευμένη στην κύρια περιοχή, τότε πρέπει να γίνουν $\log b_m$ προσπελάσεις στη χειρότερη περίπτωση. Αν η εγγραφή είναι αποθηκευμένη στην περιοχή υπερχείλισης, τότε η αναζήτηση γίνεται κατά σειριακό τρόπο (όπως και στο αρχείο σωρού), και συνεπώς, πρέπει να αναγνωσθούν οι μισές σελίδες του αρχείου, κατά μέσο όρο. Ο απαιτούμενος χρόνος για την προσπέλαση της εγγραφής είναι:

$$\log b_m \times (s + r + btt) + \left(r + s + ebt \times \frac{b_o}{2} \right)$$

Αυτός ο τύπος δίνει το χρονικό κόστος της ανεπιτυχούς αναζήτησης στην ταξινομημένη περιοχή συν το χρόνο εντοπισμού και περιστροφής στην περιοχή υπερχείλισης συν το χρόνο για τη σειριακή αναζήτηση των μισών σελίδων της περιοχής υπερχείλισης. Βέβαια, πρέπει να σημειωθεί ότι το πρώτο άθροισμα της έκφρασης είναι πεσιμιστικό, επειδή ο όρος $\log b_m$ αφορά στη χειρότερη περίπτωση. Αυτό γίνεται αντιληπτό από το Σχήμα 4.2, όπου για παράδειγμα, η ανεπιτυχής αναζήτηση του κλειδιού 105 τερματίζει με μία μόνο προσπέλαση.

Συνδυάζοντας τους δύο προηγούμενους τύπους προκύπτει ότι ο χρόνος για την αναζήτηση μίας εγγραφής είναι:

$$T_{\pi\rho\sigma\sigma} = \frac{b_m}{b} \times (\log b_m - 1) \times (s + r + btt) + \frac{b_o}{b} \times \left(\log b_m \times (s + r + btt) + \left(r + s + ebt \times \frac{b_o}{2} \right) \right)$$

Αν το b_o είναι ιδιαίτερα μεγάλο, τότε κυριαρχεί το κόστος αναζήτησης στην περιοχή υπερχείλισης και πρακτικά ο προηγούμενος τύπος καταλήγει στον:

$$T_{\pi\rho\sigma\sigma} = \frac{ebt \times b_o^2}{2 \times b}$$

Στην αντίθετη περίπτωση (δηλαδή, το b_o είναι μικρό), κυριαρχεί το κόστος αναζήτησης στην ταξινομημένη περιοχή και ο τύπος μετασχηματίζεται στον:

$$T_{\pi\rho\sigma\sigma} = (\log b - 1) \times (s + r + btt)$$

Μία άλλη μέθοδος αναζήτησης με βάση την τιμή του πρωτεύοντος κλειδιού είναι η **αναζήτηση παρεμβολής** (interpolation search), που υπερτερεί της δυαδικής μεθόδου όταν τα κλειδιά υπακούουν σε ομοιόμορφη κατανομή. Σημειώνεται ότι στη βιβλιογραφία αναφέρονται επίσης η **αναζήτηση Fibonacci**, η **αναζήτηση άλματος** (jump search) και η **πολυωνυμική αναζήτηση** (polynomial search). Στο βιβλίο των Δομών Δεδομένων αναφέρονται περισσότερα στοιχεία για τις μεθόδους αυτές, που στηρίζονται στη στρατηγική ‘διαίρει και βασίλευε’ (divide and conquer).

4.3.2 Προσπέλαση επόμενης εγγραφής

Αν η περιοχή υπερχειλίσσης είναι μικρή, τότε το κόστος για την αναζήτηση της λογικά επόμενης εγγραφής είναι επίσης μικρό. Έστω, λοιπόν, ότι ο βραχίονας είναι τοποθετημένος επάνω από τη σωστή άτρακτο και ότι το περιεχόμενο της τελευταίας σελίδας βρίσκεται ακόμη στην κύρια μνήμη. Είναι πιθανό η επόμενη εγγραφή να είναι αποθηκευμένη στην ίδια σελίδα με την εγγραφή που αναζητήθηκε πιο πρόσφατα. Η πιθανότητα να μην είναι στην ίδια σελίδα είναι $1/Bfr$. Συνεπώς ο χρόνος για την αναζήτηση της επόμενης εγγραφής είναι:

$$T_{\text{επομ}} = \left(1 - \frac{1}{Bfr}\right) \times 0 + \frac{1}{Bfr} \times (r + btt) = \frac{r + btt}{Bfr}$$

Επίσης, σημειώνεται ότι ο όρος r της περιστροφής είναι απαραίτητος για να έλθει η κεφαλή επάνω από την κατάλληλη σελίδα.

Ωστόσο, υπάρχει κάποια πιθανότητα η επόμενη εγγραφή να μην βρίσκεται στον ίδιο κύλινδρο. Αυτή η πιθανότητα αγνοείται, επειδή για μεγάλα συστήματα είναι πραγματικά μηδαμινή και η επιβάρυνση στο κόστος αμελητέα. Η ανάλυση που προηγήθηκε ισχύει με την προϋπόθεση ότι δεν υπάρχει περιοχή υπερχειλίσσης. Αν αυτή η συνθήκη δεν ισχύει, τότε το αρχείο εκφυλίζεται σε αρχείο σωρού και απαιτείται νέα ανάλυση.

4.3.3 Διαγραφή ή ανανέωση εγγραφής

Ο απαιτούμενος χρόνος για τη διαγραφή ή την ανανέωση (αν η ανανέωση δεν αφορά στο κλειδί) μίας εγγραφής είναι:

$$T_{\text{διαγ}} = T_{\text{ανα}} = T_{\text{προσ}} + 2r$$

Ο όρος $2r$ είναι ο απαραίτητος για την περιστροφή του δίσκου και την επανατοποθέτηση της κεφαλής επάνω από την ίδια σελίδα. Αν πρόκειται για διαγραφή, τότε απλώς η εγγραφή μαρκάρεται και δεν γίνεται επαναχρησιμοποίηση του φυσικού χώρου.

Αν η ανανέωση αφορά στην τιμή του κλειδιού, τότε στην ουσία πρόκειται για μία διαγραφή και μία εισαγωγή στην περιοχή υπερχειλίσης. Επομένως ισχύει:

$$T_{\text{αναν}} = T_{\text{διαγ}} + T_{\text{εισ}}$$

Το κρίσιμο σημείο είναι αν η εγγραφή βρίσκεται αποθηκευμένη στην περιοχή υπερχειλίσης ή όχι.

4.3.4 Εισαγωγή εγγραφής

Ο χρόνος για την εισαγωγή μίας εγγραφής είναι ίδιος με το χρόνο εισαγωγής εγγραφής σε αρχείο σωρού. Έτσι το χρονικό κόστος ισούται με:

$$T_{\text{εισ}} = s + 3r + btt$$

Βέβαια, στην ποσότητα αυτή πρέπει να προστεθεί και ο χρόνος αναζήτησης ώστε να διασφαλισθεί ότι δεν υπάρχουν διπλές εγγραφές.

4.3.5 Εξαντλητική ανάγνωση αρχείου

Αν οι εγγραφές της περιοχής υπερχειλίσης είναι σχετικά λίγες, τότε ο απαιτούμενος χρόνος για την ανάγνωση του αρχείου είναι:

$$T_{\text{εξαν}} = b \times ebt$$

Συνεπώς, δεν ωφελεί να διατηρείται το αρχείο ταξινομημένο, αν τελικά πρόκειται να χρησιμοποιείται κυρίως για στατιστική επεξεργασία. Όμως, αν το αρχείο χρησιμοποιείται για εφαρμογές, όπως λόγου χάριν για έναν τηλεφωνικό κατάλογο ή μία ταχυδρομική λίστα, τότε η ταξινόμησή του είναι αναγκαία.

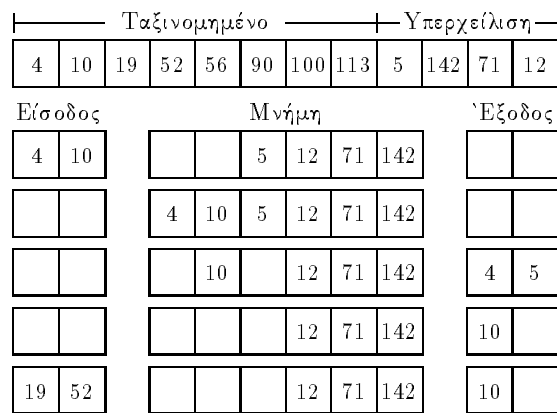
Αν υπάρχουν εγγραφές υπερχειλίσης, τότε ο χρόνος εξαντλητικής ανάγνωσης του αρχείου είναι το άθροισμα του χρόνου ανάγνωσης των εγγραφών υπερχειλίσης στην κύρια μνήμη (με την προϋπόθεση ότι χωρούν) συν το χρόνο ταξινόμησή τους συν το χρόνο συγχώνευσης των εγγραφών υπερχειλίσης με τις εγγραφές του ταξινομημένου αρχείου. Αν δεν χωρούν όλες οι εγγραφές υπερχειλίσης ταυτόχρονα στην κύρια μνήμη, τότε η επίδοση εχφυλίζεται και πρέπει να γίνει αναδιοργάνωση.

4.3.6 Αναδιοργάνωση αρχείου

Αν χρησιμοποιείται μόνο μία συσκευή δίσκου, τότε ο χρόνος αναδιοργάνωσης του αρχείου είναι:

$$T_{\text{αναδ}} = b_o \times ebt + (z \log z) \times 0.01 + b_m \times ebt + \frac{ebt \times n}{Bfr}$$

Στον τύπο αυτό ο πρώτος όρος δίνει το χρονικό κόστος για την ανάγνωση της περιοχής υπερχειλίσης (υποτίθεται ότι χωρά στην κύρια μνήμη). Ο δεύτερος όρος παριστά το χρόνο ταξινόμησης των z εγγραφών υπερχειλίσης χρησιμοποιώντας έναν αλγόριθμο τάξης $O(n \log n)$, με βάση την υπόθεση ότι κάθε κλήση της ρουτίνας απαιτεί 10 ms. Ο τρίτος όρος δίνει το χρόνο ανάγνωσης του ταξινομημένου αρχείου, ενώ ο τέταρτος όρος δίνει το χρόνο επανα-αποθήκευσης του αρχείου στο δίσκο υπό τη νέα του μορφή. Η διαδικασία αυτή παρουσιάζεται στο Σχήμα 4.3. Αν υπάρχουν διαθέσιμες δύο συσκευές δίσκων, τότε μπορεί μερικές διεργασίες να εκτελεστούν παράλληλα χρησιμοποιώντας την τεχνική της διπλής απομονωτικής μνήμης. Στο Σχήμα 4.4 εφαρμόζεται η τεχνική αυτή για το παράδειγμα του Σχήματος 4.3.



Σχήμα 4.3: Αναδιοργάνωση ταξινομημένου αρχείου με υπερχειλίση που χωρά στην κύρια μνήμη και μία συσκευή δίσκου.

Συμπερασματικά, για τα ταξινομημένα αρχεία σημειώνεται ότι:

- είναι πολύ χρήσιμα για σχετικά μικρά αρχεία και για στατικά δεδομένα, όπου δηλαδή δεν πρόκειται να υπάρξουν πολλές εισαγωγές στην

Ταξινομημένο							Υπερχείλιση				
4	10	19	52	56	90	100	113	5	142	71	12
Είσοδος		Μνήμη						Έξοδος			
4	10			5	12	71	142				
19	52	4	10	5	12	71	142				
56	90	52	10	19	12	71	142	4	5		
100	113	52	56	19	90	71	142	10	12		
		100	56	113	90	71	142	19	52		

Σχήμα 4.4: Αναδιοργάνωση ταξινομημένου αρχείου με υπερχείλιση που χωρά στην κύρια μνήμη και δύο συσκευές δίσκου.

περιοχή υπερχείλισης. Για παράδειγμα, τα δεδομένα ενός τηλεφωνικού καταλόγου ή μίας ταχυδρομικής λίστας είναι σχετικά στατικά.

- Η αναζήτηση μίας εγγραφής είναι γρήγορη, αφού απαιτεί $\log b$ προσπελάσεις σελίδων, αν γίνει με δυαδική αναζήτηση.
- Η αναζήτηση της επόμενης εγγραφής από την τρέχουσα είναι μία επίσης γρήγορη διεργασία, όπως επίσης και η εισαγωγή μίας νέας εγγραφής, αρκεί να μην υπάρξουν πολλές εισαγωγές, γιατί τότε θα χρειασθεί αναδιοργάνωση.
- Σε σχέση με το αρχείο σωρού, δεν υπάρχει κέρδος σε στατιστικές εφαρμογές, αλλά υπάρχουν σημαντικά πλεονεκτήματα σε εφαρμογές, όπως η εύρεση της τομής δύο αρχείων ή ο περιορισμός των διπλών εγγραφών.

Όταν η ανάκτηση μεμονωμένων εγγραφών είναι πολύ συχνή, τότε το ταξινομημένο αρχείο δεν ενδείκνυται. Όπως θα φανεί και σε επόμενα κεφάλαια, ένα αρχείο που χρησιμοποιεί κατάλογο είναι γενικά περισσότερο αποτελεσματικό.

4.4 Ασκήσεις

<1> Δίνεται αρχείο σωρού με 100.000 εγγραφές των 400 bytes. Για κάθε μία διαγραφή αντιστοιχούν τρεις εισαγωγές εγγραφών και έτσι τελικά το αρχείο αποκτά 150.000 εγγραφές. Πόσος χρόνος χρειάζεται για αναδιοργάνωση; Πόσος χρόνος απαιτείται για την εύρεση μίας εγγραφής πριν και μετά την αναδιοργάνωση;

<2> Αρχείο σωρού αποτελείται από 6.000.000 εγγραφές των 400 bytes. Έστω ότι υπάρχουν 500.000 διπλοεγγραφές. Να βρεθεί μία τεχνική για την απομάκρυνσή τους. Πόσο χρόνος θα χρειασθεί για την επεξεργασία αυτή, αν το μέγεθος της κύριας μνήμης είναι 10 Mb και ο δίσκος έχει τα χαρακτηριστικά του IBM 3380, αλλά μπορεί να αποθηκεύσει τόσο το αρχικό αρχείο, όσο και το παράγωγό του.

<3> Να λυθεί η προηγούμενη άσκηση για ταξινομημένο αρχείο.

<4> Το 10% των εγγραφών του αρχείου της Άσκησης 2 πρέπει να απομονωθούν σε ιδιαίτερο αρχείο σύμφωνα με μία συγκεκριμένη τιμή σε ένα δευτερεύον πεδίο. Πόσος χρόνος απαιτείται γι' αυτή την επεξεργασία; Αλλάζει το κόστος αν το αρχικό αρχείο είναι ταξινομημένο ως προς τις τιμές του δευτερεύοντος αυτού πεδίου;

<5> Έστω ότι σε ένα αταξινόμητο και σε ένα ταξινομημένο σειριακό αρχείο πρόκειται να εισαχθεί μία ομάδα εγγραφών ως σύνολο και όχι μία προς μία. Να υπολογισθεί το αντίστοιχο κόστος.

<6> Δίνεται ένα ταξινομημένο αρχείο με 100.000 εγγραφές των 400 bytes, το οποίο είναι αποθηκευμένο σε δίσκο IBM 3380. Ποιά από τις δύο επόμενες μεθόδους είναι προτιμότερη για την ανάκτηση 10 εγγραφών:

- να γίνει δυαδική αναζήτηση ανεξάρτητα για κάθε εγγραφή, ή
- να ταξινομηθούν τα κλειδιά των εγγραφών και να αναζητηθούν με μία σειριακή ανάγνωση του αρχείου;

Ποιά μέθοδος είναι προτιμότερη αν οι αναζητούμενες εγγραφές είναι 100, 1000 ή 10.000; Να προταθούν αποτελεσματικότεροι τρόποι ανάκτησης ενός συνόλου εγγραφών. Να θεωρηθεί ότι π χρόνος που απαιτείται για την ταξινόμηση z εγγραφών είναι $(z \log z) \times 0.01$ ms.

<7> Από το αρχείο της προηγούμενης άσκησης εκτελείται μία λογική διαγραφή 5000 εγγραφές με τη βοήθεια μίας σημαίας. Στη συνέχεια εισάγονται 5000 εγγραφές στην περιοχή υπερχείλισης. Να προταθεί και να κοστολογηθεί μία τεχνική για την αναδιοργάνωση του αρχείου.

Κεφάλαιο 5

ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΤΑΙΝΙΕΣ

- 5.1 Εισαγωγή
- 5.2 Φυσική συγχώνευση
- 5.3 Ισοζυγισμένη συγχώνευση
- 5.4 Πολυφασική συγχώνευση
- 5.5 Συγχώνευση καταρράκτη
- 5.6 Ασκήσεις

Κεφάλαιο 5

ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΤΑΙΝΙΕΣ

5.1 Εισαγωγή

Σχεδόν σε όλα τα υπολογιστικά συστήματα υπάρχουν έτοιμα πακέτα ταξινόμησης, και ο προγραμματιστής μπορεί να κωδικοποιήσει δικές του ρουτίνες ταξινόμησης αρχείων. Αν ένα αρχείο δεν είναι μεγάλο και χωρά στην κύρια μνήμη, τότε τα πράγματα είναι απλά. Εφαρμόζεται ένας οποιοσδήποτε αλγόριθμος εσωτερικής ταξινόμησης, όπως για παράδειγμα, η μέθοδος ταξινόμησης με **διαμερισμό και ανταλλαγή** (quicksort), και το αρχείο αποθηκεύεται και πάλι στη δευτερεύουσα μνήμη. Όμως, αν το αρχείο δεν χωρά στην κύρια μνήμη, τότε πρέπει να γίνει χρήση συμβατικής μνήμης, οπότε η ταχύτητα της μεθόδου καθορίζεται από το λειτουργικό σύστημα. Στο σημείο αυτό, θα υπενθυμισθεί σύντομα πώς λειτουργεί ο αλγόριθμος αυτός.

Σύμφωνα με τη μέθοδο της ταξινόμησης με διαμερισμό και ανταλλαγή, το αρχείο διαιρείται σε δύο υποαρχεία με βάση μία τιμή που λέγεται **‘άξονας’** (pivot). Το κάθε υποαρχείο περιέχει κλειδιά μικρότερα και μεγαλύτερα από τον άξονα, αντίστοιχα. Αυτό γίνεται με δύο σαρώσεις από την αρχή και το τέλος του αρχείου και την αντιμετάθεση των κλειδιών, που δεν βρίσκονται στο σωστό υποαρχείο. Αν γίνει σωστή εκλογή του άξονα, τότε ένα αρχείο του 1 Gb διαχωρίζεται σε δύο υποαρχεία των 500 Mb. Η διαδικασία αυτή συνεχίζεται για κάθε υποαρχείο επαναληπτικά χρησιμοποιώντας μία νέα τιμή άξονα για το καθένα. Αν η κύρια μνήμη είναι 4 Mb, τότε το αρχείο αυτό πρέπει να αναγνωσθεί και να αποθηκευθεί από μία φορά κατά το πρώτο πέρασμα. Κατά μέσο όρο αυτή η διαδικασία θα πρέπει να επαναληφθεί

8 φορές πριν τα υποαρχεία είναι τόσο μικρά, ώστε να χωρέσουν στην κύρια μνήμη. Προφανώς λοιπόν, για πολύ μεγάλα αρχεία η μέθοδος ταξινόμησης με διαμερισμό και ανταλλαγή δεν ενδείκνυται για εξωτερική ταξινόμηση. Πάντως, έχει παρατηρηθεί ότι αν τα αρχεία είναι λίγο μεγαλύτερα από τη διατιθέμενη κύρια μνήμη, τότε μπορεί να χρησιμοποιηθεί αυτή η μέθοδος καθώς και η μέθοδος επιλογής αντικατάστασης, που θα παρουσιασθεί στο επόμενο κεφάλαιο.

Συνήθως, λοιπόν, τα αρχεία δεν χωρούν στην κύρια μνήμη, οπότε οι γνωστές μέθοδοι εσωτερικής ταξινόμησης δεν μπορούν να εφαρμοσθούν. Ο όρος **εξωτερική ταξινόμηση** (external sorting) αναφέρεται σε μεθόδους ταξινόμησης αρχείων, που ελάχιστα σχετίζονται με τις γνωστές μεθόδους ταξινόμησης δομών κύριας μνήμης. Στο κεφάλαιο αυτό θα εξετασθούν μέθοδοι εξωτερικής ταξινόμησης, που παρουσιάστηκαν όταν κυρίαρχο μέσο για δευτερεύουσα αποθήκευση ήταν η μαγνητική ταινία, ενώ στο επόμενο θα παρουσιασθούν μέθοδοι σχεδιασμένες ειδικά για μαγνητικούς δίσκους. Όλες αυτές οι μέθοδοι εμφανίστηκαν στη βιβλιογραφία πριν το 1960 και αποτέλεσαν αντικείμενο ανάλυσης στη δεκαετία του 60, την εποχή που μοναδικός σκοπός των υπολογιστών ήταν η επεξεργασία αριθμών. Όπως θα φανεί στη συνέχεια, οι μέθοδοι που σχεδιάστηκαν για μαγνητικές ταινίες μπορούν να εφαρμοσθούν και σε μαγνητικούς δίσκους, αλλά οπωσδήποτε δεν είναι τόσο αποτελεσματικές όσο οι μέθοδοι που εφαρμόζονται ειδικά σε δίσκους. Επίσης στο παρόν κεφάλαιο θα γίνει ποιοτική ανάλυση της επίδοσης των μεθόδων αυτών. Ενώ στις μεθόδους εσωτερικής ταξινόμησης τα κριτήρια κόστους είναι ο αριθμός των συγκρίσεων και ο αριθμός των μετακινήσεων κλειδιών, στην περίπτωση της εξωτερικής ταξινόμησης το κύριο κριτήριο κόστους είναι ο αριθμός των προσπελάσεων σελίδων για είσοδο/έξοδο των δεδομένων.

Όλες οι μέθοδοι και των δύο κεφαλαίων έχουν την ίδια γενική δομή, δηλαδή υλοποιούνται σε δύο στάδια. Στο πρώτο στάδιο το αρχείο υποδιαιρείται σε τμήματα που χωρούν στην κύρια μνήμη και ταξινομούνται το καθένα με μία μέθοδο εσωτερικής ταξινόμησης. Η επιλογή της μεθόδου για την εσωτερική ταξινόμηση δεν αποτελεί πρόβλημα για το κεφάλαιο αυτό. Δηλαδή, θα μπορούσε να εφαρμοσθεί μία οποιαδήποτε μέθοδος από αυτές που περιγράφονται στο βιβλίο των Δομών Δεδομένων και χαρακτηρίζονται από πολυπλοκότητα $O(n \log n)$. Ωστόσο, στο επόμενο κεφάλαιο θα φανεί ότι έχει ιδιαίτερη σημασία για τη συνολική επίδοση της μεθόδου ποιός αλγόριθμος εσωτερικής ταξινόμησης χρησιμοποιείται για τη δημιουργία των ταξινομημένων τμημάτων.

Στο δεύτερο στάδιο, τα ταξινομημένα τμήματα συγχωνεύονται ώστε να προκύψει ένα ταξινομημένο αρχείο. Για αυτό το λόγο, οι μέθοδοι που ακολουθούν είναι γνωστές ως 'συγχωνεύσεις'. Ο αναγνώστης καλείται να ανατρέξει στο βιβλίο των Δομών Δεδομένων για λεπτομέρειες σχετικά με τον αλγόριθμο συγχώνευσης δομών κύριας μνήμης. Στο παρόν κεφάλαιο θα δοθεί έμφαση στις συγχωνεύσεις κάτω από ένα νέο πρίσμα κοστολόγησης, δηλαδή από την άποψη της εκτίμησης του μεγέθους των δεδομένων που μεταφέρονται από το δίσκο στη μνήμη, και το αντίστροφο.

5.2 Φυσική συγχώνευση

Βαθμός (degree) της συγχώνευσης είναι ο αριθμός των αρχείων που πρόκειται να συγχωνευθούν. Το ιδανικό θα ήταν να υπήρχαν τόσες διαθέσιμες συσχευές όσες και τα ταξινομημένα τμήματα. Έτσι με μία συγχώνευση ισάριθμων δρόμων η εξωτερική ταξινόμηση θα τελείωνε. Αυτή η περίπτωση είναι ιδεατή γιατί δεν είναι δυνατόν:

- να κρατούνται ανοικτά πάρα πολλά αρχεία, και
- να υπάρχουν διαθέσιμες τόσες πολλές συσχευές μαγνητικών ταινιών.

Σε μία φυσική συγχώνευση P δρόμων (P -way natural merge) υπάρχουν P συσχευές στην είσοδο και μία συσχευή στην έξοδο. Έτσι, συνολικά απαιτούνται $P+1$ συσχευές.

Έστω ότι αρχικά υπάρχουν nsg ταξινομημένα τμήματα συνολικού μεγέθους ίσου με το μέγεθος της κύριας μνήμης. Ας σημειωθεί ότι στην αγγλική βιβλιογραφία τα ταξινομημένα τμήματα αναφέρονται ως runs ή σπανιότερα ως strings. Επίσης, αρχικά κάθε ταξινομημένο τμήμα περιέχει ένα σταθερό αριθμό εγγραφών σταθερού μήκους. Τα ταξινομημένα αυτά τμήματα κατανέμονται με κυκλικό τρόπο στις P συσχευές κατά τη δημιουργία τους.

Η συγχώνευση αρχίζει λαμβάνοντας από κάθε μία από τις P συσχευές ένα ταξινομημένο τμήμα και δημιουργώντας ένα ταξινομημένο τμήμα μεγέθους P φορές μεγαλύτερο από το αρχικό. Το τμήμα αυτό αποθηκεύεται στην $(P+1)$ -οστή συσχευή. Η διαδικασία συνεχίζεται με τη διαδοχική συγχώνευση P τμημάτων, ανά ένα από τις P συσχευές, και τη δημιουργία νέων τμημάτων μεγαλύτερου μεγέθους. Όταν εξαντληθούν όλα τα τμήματα του αρχικού μεγέθους, το αποτέλεσμα είναι να υπάρχουν $\lceil nsg/P \rceil$ ταξινομημένα

τμήματα περίπου στην $(P+1)$ -οστή συσκευή. Τα τμήματα αυτά κατανέμονται και πάλι εξίσου στις P συσκευές κατά κυκλικό τρόπο. Στη συνέχεια οι φάσεις επαναλαμβάνονται κατά παρόμοιο τρόπο μέχρι να προκύψει ένα μόνο ταξινομημένο τμήμα στη συσκευή εξόδου.

Συσκευή 1	Συσκευή 2	Συσκευή 3
6×100	6×100	—
—	—	6×200
3×200	—	3×200
—	3×400	—
1×400	2×400	—
—	1×400	1×800
1×1200	—	—

Πίνακας 5.1: Φυσική συγχώνευση δύο δρόμων.

Το παράδειγμα του Πίνακα 5.1 παρουσιάζει τη μέθοδο φυσικής συγχώνευσης δύο δρόμων με σκοπό την εξωτερική ταξινόμηση 1200 εγγραφών που κατανέμονται σε 12 ταξινομημένα τμήματα των 100 εγγραφών. Το περιεχόμενο των συσκευών φαίνεται στις κατακόρυφες στήλες, ενώ οι διαδοχικές συγχωνεύσεις και ανακατανομές φαίνονται στην οριζόντια διάσταση. Πιο συγκεκριμένα, κάθε γινόμενο δηλώνει το πλήθος των ταξινομημένων τμημάτων και το μέγεθός τους σε εγγραφές. Στον Πίνακα 5.2 τα ίδια δεδομένα ταξινομούνται με τη μέθοδο της φυσικής συγχώνευσης τριών δρόμων. Η βελτίωση της επίδοσης είναι προφανής.

Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4
4×100	4×100	4×100	—
—	—	—	4×300
1×300	1×300	—	2×300
—	—	1×900	1×300
1×1200	—	—	—

Πίνακας 5.2: Φυσική συγχώνευση τριών δρόμων.

Με βάση όσα αναφέρθηκαν προηγουμένως, εύκολα προκύπτει ότι κάθε φάση διαρκεί λιγότερο από μία σειριακή ανάγνωση του αρχείου. Ωστόσο, επειδή η ταξινόμηση απαιτεί πολλές φάσεις, κάθε τμήμα υφίσταται επεξεργασία αρκετές φορές. Ως **πέρασμα** (pass) ορίζεται ο λόγος του συνολικού

αριθμού των προσπελάσεων των εγγραφών προς το συνολικό αριθμό των εγγραφών του αρχείου. Με άλλα λόγια ο αριθμός των περασμάτων δίνει το μέσο όρο αναγνώσεων μίας εγγραφής κατά τη συγχώνευση. Για τη φυσική συγχώνευση 2 δρόμων του πρώτου παραδείγματος απαιτούνται 4,5 περάσματα, ενώ για τη φυσική συγχώνευση 3 δρόμων του δεύτερου παραδείγματος απαιτούνται 3,25 περάσματα. Η εύρεση των αποτελεσμάτων αυτών αφήνεται ως άσκηση στον αναγνώστη.

Στη συνέχεια θα δοθεί αναλυτική έκφραση του απαιτούμενου χρόνου για μία φυσική συγχώρευση P δρόμων. Ο αριθμός των φάσεων όπου εκτελείται συγχώνευση είναι $\lceil \log_p nsg \rceil$. Βέβαια, κάθε φάση συγχώνευσης (εκτός από την τελευταία) ακολουθείται από μία φάση ανακατανομής. Συνεπώς, συνολικά απαιτούνται $2\lceil \log_p nsg \rceil - 1$ φάσεις. Στον αριθμό αυτό πρέπει να συνυπολογισθεί και άλλη μία φάση, αυτή της δημιουργίας των ταξινομημένων τμημάτων. Άρα, ο συνολικός απαιτούμενος χρόνος ισούται με:

$$2 \times 2\lceil \log_p nsg \rceil \times \left(\frac{Ibg}{Spd} + \frac{Buf}{(P+1) \times Spd \times Den} \right) \times \frac{n \times R}{\frac{Buf}{P+1}}$$

όπου Buf είναι το μέγεθος της απομονωτικής μνήμης. Ο τελευταίος όρος του γινομένου ερμηνεύεται ως εξής. Η διαθέσιμη απομονωτική μνήμη, Buf , επιμερίζεται σε $P+1$ τμήματα για την προσωρινή αποθήκευση των δεδομένων εισόδου/εξόδου. Συνεπώς, ο αριθμός των φορών που θα γεμίσει η μνήμη είναι $\frac{n \times R}{\frac{Buf}{P+1}}$. Ο χρόνος που απαιτείται κάθε φορά που γεμίζει ένα τμήμα της μνήμης δίνεται από τη σχετική παρένθεση. Επίσης, σημειώνεται ότι ο πρώτος συντελεστής 2 εισάγεται, ώστε να αποδοθεί ο χρόνος ανάγνωσης και ο χρόνος αποθήκευσης.

Σοβαρό μειονέκτημα της μεθόδου είναι ότι, όταν τελειώνει μία φάση συγχώνευσης, απαιτείται ανακατανομή μεγάλων ταξινομημένων τμημάτων σε άλλες συσκευές. Αυτή η ανακατανομή αποτελεί το μισό περίπου κόστος της μεθόδου και είναι μία μη παραγωγική διαδικασία, γιατί τα δεδομένα διαβάζονται και αποθηκεύονται χωρίς να υφίστανται κάποια επεξεργασία. Η επόμενη μέθοδος εξωτερικής ταξινόμησης δεν χαρακτηρίζεται από αυτό το μειονέκτημα.

5.3 Ισοζυγισμένη συγχώνευση

Η ισοζυγισμένη συγχώνευση P -δρόμων (P -way balanced merge) χρησιμοποιεί P συσκευές στην είσοδο και P συσκευές στην έξοδο, δηλαδή

συνολικά $2P$ συσκευές. Και πάλι τα αρχικά ταξινομημένα τμήματα κατανέμονται κυκλικά στις P συσκευές εισόδου. Η ύπαρξη όμως P συσκευών στην έξοδο συντελεί στην αποφυγή των διαδοχικών απαιτούμενων ανακατανομών, όπως στη φυσική συγχώνευση. Αυτό επιτυγχάνεται εναλλάσσοντας κάθε φορά το ρόλο των συσκευών εισόδου με το ρόλο των συσκευών εξόδου. Η μέθοδος είναι ισοζυγισμένη υπό την έννοια ότι σε κάθε συγχώνευση τα ταξινομημένα τμήματα κατανέμονται σε ίσο αριθμό στις συσκευές εξόδου. Στον Πίνακα 5.3 φαίνεται ένα παράδειγμα ισοζυγισμένης συγχώνευσης δύο δρόμων, όπου το αρχείο αποτελείται από 12 ταξινομημένα τμήματα των 100 εγγράφων. Στη παράδειγμα αυτό απαιτούνται 3,67 περάσματα.

Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4
6×100	6×100	—	—
—	—	3×200	3×200
2×400	1×400	—	—
1×400	—	1×800	—
—	1×1200	—	—

Πίνακας 5.3: Ισοζυγισμένη συγχώνευση δύο δρόμων.

Κατά την ισοζυγισμένη συγχώνευση απαιτούνται $\lceil \log_p nsg \rceil$ φάσεις συγχώνευσης συν μία φάση δημιουργίας των αρχικών ταξινομημένων τμημάτων. Έτσι ο απαιτούμενος χρόνος για είσοδο/έξοδο των δεδομένων είναι:

$$2 \times \lceil \log_p nsg \rceil \times \left(\frac{Ibg}{Spd} + \frac{Buf}{2P \times Spd \times Den} \right) \times \frac{n \times R}{\frac{Buf}{2P}}$$

Η έκφραση αυτή ερμηνεύεται ακολουθώντας το σκεπτικό της αντίστοιχης έκφρασης για τη φυσική συγχώνευση, και με βάση το γεγονός ότι δεν εκτελούνται αναδιανομές των ταξινομημένων τμημάτων (άρα, δεν χρειάζεται ο συντελεστής 2), ενώ η απομονωτική μνήμη χωρίζεται σε $2P$ τμήματα για την προσωρινή αποθήκευση των δεδομένων εισόδου/εξόδου.

Η μέθοδος αυτή είναι απλή στην υλοποίησή της, αλλά υπάρχουν σημαντικά περιθώρια βελτίωσης. Επίσης, γίνεται αντιληπτό ότι η ισοζυγισμένη συγχώνευση μπορεί να εφαρμοσθεί και στην περίπτωση όπου ο αριθμός των συσκευών εισόδου δεν ισούται με τον αριθμό των συσκευών εξόδου.

5.4 Πολυφασική συγχώνευση

Το μειονέκτημα της προηγούμενης μεθόδου είναι ότι, αν και απαιτεί $2P$ συσκευές, σε κάθε χρονική στιγμή μόνο $P+1$ χρησιμοποιούνται, ενώ οι υπόλοιπες παραμένουν αδρανείς. Έτσι προέκυψε μία νέα μέθοδος συγχώνευσης που δεν είναι ισοζυγισμένη, αλλά δεν διακρίνεται από περιττές ανακατανομές τμημάτων, όπως συμβαίνει στη φυσική συγχώνευση. Η μέθοδος αυτή ονομάζεται **πολυφασική συγχώνευση P δρόμων** (P -way polyphase merge) και χρησιμοποιεί P συσκευές για είσοδο και 1 για έξοδο. ‘Νονός’ της μεθόδου ήταν ο Gilstad που περιέγραψε το γενικό πρότυπο το 1960, αν και η μέθοδος είχε ήδη προταθεί από τον Betz το 1956 (αλλά μόνο για τρεις ταινίες). Η βασική διαφορά της συγχώνευσης αυτής σε σχέση με τις προηγούμενες είναι ότι τα ταξινομημένα τμήματα δεν κατανέμονται εξίσου στις συσκευές εισόδου, αλλά εφαρμόζεται ένας βέλτιστος τρόπος κατανομής των τμημάτων ώστε να διευκολύνονται οι κατοπινές φάσεις.

Έστω, λοιπόν, ότι υπάρχουν διαθέσιμες $P+1$ συσκευές δίσκων. Σε κάθε φάση η μέθοδος αυτή συγχωνεύει ταξινομημένα τμήματα από P συσκευές και τα αποθηκεύει σε αυτήν που απομένει. Έτσι, στο τέλος κάθε φάσης μία συσκευή αδειάζει από δεδομένα και καθίσταται η νέα έξοδος. Εν τέλει, το ιδανικό είναι κατά την τελευταία φάση σε κάθε συσκευή να υπάρχει ένα μοναδικό τμήμα που να υπεισέρχεται σε μία συγχώνευση P δρόμων. Για να επιτευχθεί ο στόχος αυτός (δηλαδή, στην τελευταία φάση σε κάθε συσκευή να υπάρχει ένα και μόνον ένα ταξινομημένο τμήμα), χρησιμοποιείται μία έξυπνη αρχική κατανομή των ταξινομημένων τμημάτων στις P συσκευές που στηρίζεται στους αριθμούς Fibonacci. Η ακολουθία των αριθμών Fibonacci τάξης P ορίζεται ως εξής:

$$F_i = \begin{cases} 0 & \text{αν } 0 \leq i < P-1 \\ 1 & \text{αν } i = P-1 \\ \sum_{j=i-P}^{i-1} F_j & \text{αν } i > P-1 \end{cases}$$

Σύμφωνα με την αρχική έξυπνη κατανομή ο αριθμός των ταξινομημένων τμημάτων της i -οστής συσκευής ($1 \leq i \leq P$) δίνεται από το άθροισμα:

$$F_{l+P-2} + F_{l+P-1} + \dots + F_{l+i-3} = \sum_{j=i}^P F_{l+j-2}$$

όπου l είναι το **επίπεδο** (level) της φάσης. Συνεπώς, ο συνολικός αριθμός των αρχικών ταξινομημένων τμημάτων είναι:

$$S_l = P \times F_{l+P-2} + (P-1) \times F_{l+P-3} + \dots + F_{l-1}$$

Προφανώς καθώς το l αυξάνει, αυξάνει και το S_l . Μάλιστα οι όροι της ακολουθίας S_l προκύπτουν, όπως ο γενικός όρος της ακολουθίας Fibonacci. Έτσι η ακολουθία αυτή αποκαλείται 'γενικευμένη ακολουθία Fibonacci'. Για παράδειγμα, έστω ότι διατίθενται $P+1=5$ ταινίες, άρα οι υπολογισμοί θα γίνουν με αριθμούς Fibonacci τέταρτης τάξης. Είναι εύκολο να υπολογισθεί η ακολουθία αυτών των αριθμών Fibonacci: 0, 0, 0, 1, 1, 2, 4, 8, 15, 29, 56 κλπ. Στον Πίνακα 5.4 φαίνεται η κατανομή των ταξινομημένων τμημάτων για επίπεδα $1 \leq l \leq 5$. Η αντίστοιχη γενικευμένη ακολουθία Fibonacci είναι 4, 7, 13, 25, 49, 94 κλπ.

l	Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4	S_l
1	$F_3 + F_2 + F_1 + F_0$	$F_3 + F_2 + F_1$	$F_3 + F_2$	F_3	4
2	$F_4 + F_3 + F_2 + F_1$	$F_4 + F_3 + F_2$	$F_4 + F_3$	F_4	7
3	$F_5 + F_4 + F_3 + F_2$	$F_5 + F_4 + F_3$	$F_5 + F_4$	F_5	13
4	$F_6 + F_5 + F_4 + F_3$	$F_6 + F_5 + F_4$	$F_6 + F_5$	F_6	25
5	$F_7 + F_6 + F_5 + F_4$	$F_7 + F_6 + F_5$	$F_7 + F_6$	F_7	49
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Πίνακας 5.4: Κατανομή ταξινομημένων τμημάτων.

Στη γενική περίπτωση, λοιπόν, δοθέντος ενός αριθμού S αρχικών ταξινομημένων τμημάτων πρέπει να βρεθεί ο μικρότερος δυνατός αριθμός S_l με τιμή μεγαλύτερη ή ίση με S . Αν οι αριθμοί S και S_l είναι άνισοι, τότε θεωρούνται εικονικά τμήματα για να γίνει η προβλεπόμενη έξυπνη κατανομή των ταξινομημένων τμημάτων. Σχετικά με τα εικονικά τμήματα θα γίνει αναφορά αργότερα και προς το παρόν υποθέτουμε ότι δίνεται ένας βολικός αριθμός. Σε μία τέλεια πολυφασική συγχώνευση P δρόμων ο αρχικός αριθμός ταξινομημένων τμημάτων πρέπει να ισούται με κάποιον γενικευμένο αριθμό Fibonacci τάξης P . Με άλλα λόγια ισχύει ο εξής γενικός κανόνας: 'Σε μία τέλεια πολυφασική συγχώνευση P δρόμων πρέπει ο αριθμός των ταξινομημένων τμημάτων να ισούται με το άθροισμα κάθε P , $P-1$, ..., 1 διαδοχικών αριθμών Fibonacci τάξης P '.

Στον Πίνακα 5.5 παρουσιάζεται ένας πρακτικός τρόπος εύρεσης της βέλτιστης κατανομής. Ας υποθεθεί ότι το αρχείο αποτελείται από 49 ταξινομημένα τμήματα των 100 εγγραφών και ότι οι διαθέσιμες συσκευές εισόδου είναι τέσσερις (συν μία για την έξοδο). Στην οριζόντια διάσταση του πίνακα δίνεται το περιεχόμενο της κάθε μίας συσκευής, ενώ στις κατακόρυφες στήλες παρουσιάζεται για κάθε κύκλο το περιεχόμενο κάθε συσκευής, καθώς

Συσκευή 1	0	1	1	3	7	15	...
Συσκευή 2	0	1	2	2	6	14	...
Συσκευή 3	0	1	2	4	4	12	...
Συσκευή 4	1	1	2	4	8	8	...
Άθροισμα	1	4	7	13	25	49	...

Πίνακας 5.5: Κατανομή ταξινομημένων τμημάτων.

και το σύνολο των ταξινομημένων τμημάτων σε όλες τις συσκευές. Έτσι φαίνεται ότι στον πρώτο κύκλο αποθηκεύεται από ένα τμήμα σε κάθε συσκευή. Στο δεύτερο κύκλο αποθηκεύονται στη δεύτερη, τρίτη και τέταρτη συσκευή τόσα τμήματα όσα είναι τα αποθηκευμένα τμήματα στην πρώτη συσκευή από τον προηγούμενο κύκλο. Έτσι με το πέρας του δεύτερου κύκλου το περιεχόμενο των τεσσάρων συσκευών είναι 1, 2, 2 και 2 τμήματα, αντίστοιχα. Στον τρίτο κύκλο αποθηκεύονται στην τρίτη, στην τέταρτη και στην πρώτη συσκευή τόσα τμήματα όσα είναι τα αποθηκευμένα τμήματα στη δεύτερη συσκευή από τον προηγούμενο κύκλο. Έτσι με το πέρας του τρίτου κύκλου το περιεχόμενο των τεσσάρων συσκευών είναι 3, 2, 4 και 4 τμήματα, αντίστοιχα. Προχωρώντας κατά παρόμοιο τρόπο προκύπτει ότι με το πέρας του τέταρτου κύκλου έχουν κατανεμηθεί 15, 14, 12 και 8 τμήματα στις τέσσερις συσκευές αντίστοιχα, δηλαδή συνολικά 49 ταξινομημένα τμήματα. Στον Πίνακα 5.6 φαίνεται το γενικό πρότυπο ανάπτυξης των γενικευμένων αριθμών Fibonacci για πολυφασική συγχώνευση τεσσάρων δρόμων.

l	Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4	S_l
0	1	0	0	0	1
1	1	1	1	1	4
2	2	2	2	1	7
3	4	4	3	2	13
4	8	7	6	4	25
5	15	14	12	8	49
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
n	a	b	c	d	t
$n+1$	$a+b$	$a+c$	$a+d$	a	$t+3a$

Πίνακας 5.6: Πρότυπο ανάπτυξης γενικευμένων αριθμών Fibonacci.

Στον Πίνακα 5.7 παρουσιάζεται η διαδικασία της πολυφασικής συγχώ-

Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4	Συσκευή 5
15 × 100	14 × 100	12 × 100	8 × 100	—
7 × 100	6 × 100	4 × 100	—	8 × 400
3 × 100	2 × 100	—	4 × 700	4 × 400
1 × 100	—	2 × 1300	2 × 700	2 × 400
—	1 × 2500	1 × 1300	1 × 700	1 × 400
1 × 4900	—	—	—	—

Πίνακας 5.7: Πολυφασική συγχώνευση τεσσάρων δρόμων.

νευσης τεσσάρων δρόμων για 49 αρχικά ταξινομημένα τμήματα, όπου κάθε γραμμή αντιστοιχεί σε μία φάση. Η σύμπτωση του αριθμού των ταξινομημένων τμημάτων ανά συσκευή σε κάθε φάση με τις αντίστοιχες τιμές του Πίνακα 5.6 δεν είναι τυχαία. Εύκολα προκύπτει ότι ο αριθμός των περασμάτων είναι $160/49 \approx 3,27$. Αν ο αναγνώστης δοκιμάσει να εφαρμόσει τους δύο προηγούμενους αλγορίθμους με τα ίδια δεδομένα, τότε θα διαπιστώσει σημαντική βελτίωση της επίδοσης. Ο Knuth έφθασε σε αναλυτικές εκφράσεις για την εύρεση του αριθμού των φάσεων και των περασμάτων σε συνάρτηση με το βαθμό της πολυφασικής συγχώνευσης. Τα αποτελέσμα-

P	Φάσεις	Περάσματα ($PN(P)$)	Περάσματα/ Φάσεις
2	$2,078 \times \ln nsg + 0,672$	$1,504 \times \ln nsg + 0,992$	72%
3	$1,641 \times \ln nsg + 0,374$	$1,015 \times \ln nsg + 0,965$	62%
4	$1,524 \times \ln nsg + 0,078$	$0,863 \times \ln nsg + 0,921$	57%
5	$1,479 \times \ln nsg + 0,185$	$0,795 \times \ln nsg + 0,864$	54%
6	$1,460 \times \ln nsg + 0,424$	$0,762 \times \ln nsg + 0,797$	52%
7	$1,451 \times \ln nsg + 0,642$	$0,744 \times \ln nsg + 0,723$	51%
8	$1,447 \times \ln nsg + 0,838$	$0,734 \times \ln nsg + 0,646$	51%
9	$1,445 \times \ln nsg + 1,017$	$0,728 \times \ln nsg + 0,568$	50%

Πίνακας 5.8: Αριθμός περασμάτων σε πολυφασική συγχώνευση.

τα αυτά συνοψίζονται στον Πίνακα 5.8, όπου φαίνεται επίσης το ποσοστό επεξεργασίας του αρχείου ως πηλίκο του αριθμού των περασμάτων δια του αριθμού των φάσεων. Συνεπώς, ο απαιτούμενος χρόνος για είσοδο/έξοδο των δεδομένων ισούται με:

$$2 \times PN(P) \times \left(\frac{Ibg}{Spd} + \frac{Buf}{(P+1) \times Spd \times Den} \right) \times \frac{n \times R}{P+1}$$

όπου ο όρος $PN(P)$ δίνει τον αριθμό των περασμάτων ως συνάρτηση του P .

Επισημαίνεται και πάλι ότι αν ο αρχικός αριθμός ταξινομημένων τμημάτων δεν ανήκει στην αντίστοιχη ακολουθία Fibonacci, τότε πρέπει να θεωρηθούν εικονικά ταξινομημένα τμήματα. Τα εικονικά αυτά τμήματα δεν είναι παρά κάποιοι μετρητές στον κώδικα της μεθόδου. Από τον αλγόριθμο επίσης φαίνεται ότι τα τελευταία τμήματα κάθε συσκευής υφίστανται λιγότερη επεξεργασία. Συνεπώς, η επίδοση βελτιώνεται αν τα εικονικά τμήματα καταναμεθούν εξίσου στις διαθέσιμες συσκευές εισόδου και μάλιστα αποθηκευθούν (θεωρητικά) εμπρός από τα πραγματικά τμήματα. Έτσι η προσθήκη των εικονικών αυτών τμημάτων δεν επιβαρύνει τη μέθοδο ούτε με περισσότερες φάσεις, ούτε με περισσότερη δραστηριότητα εισόδου/εξόδου.

5.5 Συγχώνευση καταρράκτη

Μία άλλη μη ισοζυγισμένη συγχώνευση είναι η **συγχώνευση καταρράκτη** (cascade merge), που προτάθηκε από τους Betz και Carter το 1959, δηλαδή πριν την πολυφασική συγχώνευση. Και σε αυτή τη μέθοδο, μία έξυπνη αρχική κατανομή των ταξινομημένων τμημάτων βελτιστοποιεί την επίδοσή της. Η συγχώνευση αυτή, λοιπόν, αρχίζει λαμβάνοντας ταξινομημένα τμήματα από τις P συσκευές και αποθηκεύοντας τα προκύπτοντα τμήματα στην $(P+1)$ -οστή συσκευή. Όταν μία από τις P συσκευές αδειάσει, τότε η διαδικασία συνεχίζεται λαμβάνοντας τμήματα από τις $P-1$ συσκευές και αποθηκεύοντας τα προκύπτοντα τμήματα στη συσκευή που μόλις άδειασε. Με τον τρόπο αυτό συνεχίζεται η διαδικασία μέχρις ότου από δύο συσκευές εισόδου τα αποτελέσματα να κατευθύνονται σε μία συσκευή εξόδου. Έτσι στο σημείο αυτό ταξινομημένα τμήματα υπάρχουν σε P συσκευές και η διαδικασία συνεχίζεται με τον ίδιο τρόπο. Δηλαδή, ο βαθμός της συγχώνευσης δεν διατηρείται σταθερός, αλλά μεταβάλλεται κυκλικά από P ως 2. Στον Πίνακα 5.9 φαίνεται η μεθοδολογία εύρεσης της βέλτιστης κατανομής ταξινομημένων τμημάτων για μία συγχώνευση καταρράκτη πέντε δρόμων.

Για παράδειγμα, έστωσαν 190 ταξινομημένα τμήματα των 1000 εγγράφων, που πρόκειται να ταξινομηθούν με τη μέθοδο αυτή. Στον Πίνακα 5.10 φαίνεται η διαδικασία των διαδοχικών συγχωνεύσεων. Και πάλι η αντιστοιχία των περιεχομένων των συσκευών με τη μέθοδο της εύρεσης της βέλτιστης κατανομής είναι προφανής. Είναι εύκολο για τον αναγνώστη να

l	Συσκ. 1	Συσκ. 2	Συσκ. 3	Συσκ. 4	Συσκ. 5	S_l
0	1	0	0	0	0	1
1	1	1	1	1	1	5
2	5	4	3	2	1	15
3	15	14	12	9	5	55
4	55	50	41	29	15	190
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
n	a	b	c	d	e	
$n+1$	$\left\{ \begin{array}{l} a+b+c \\ +d+e \end{array} \right\}$	$\left\{ \begin{array}{l} a+b+ \\ c+d \end{array} \right\}$	$a+b+c$	$a+b$	a	

Πίνακας 5.9: Ανάπτυξη βέλτιστης κατανομής για συγχώνευση καταρράκτη.

Συσκευή 1	Συσκευή 2	Συσκευή 3	Συσκευή 4	Συσκευή 5	Συσκευή 6
55×1	50×1	41×1	29×1	15×1	—
40×1	35×1	26×1	14×1	—	15×5
26×1	21×1	12×1	—	14×4	
14×1	9×1	—	12×3		
5×1	—	9×2			
—	5×15	4×2	7×3	9×4	10×5
4×14		—	3×3	5×4	6×5
		3×2	—	2×4	3×5
			2×9	—	1×5
3×14	4×15	2×12	1×9	1×55	—
2×14	3×15	1×12	—		1×50
1×14	2×15	—	1×41		
—	1×15	1×29			
1×190	—	—	—	—	—

Πίνακας 5.10: Συγχώνευση καταρράκτη πέντε δρόμων.

διαπιστώσει ότι ο αριθμός των περασμάτων είναι $735/190 \approx 3,87$. Αν ο αριθμός των αρχικών ταξινομημένων τμημάτων δεν είναι βολικός (όπως το 190), τότε θεωρούνται εικονικά τμήματα όπως και στην πολυφασική μέθοδο. Ο Knuth έφθασε σε αναλυτικές εκφράσεις για τον αριθμό των φάσεων και των περασμάτων ως συνάρτηση του βαθμού της συγχώνευσης. Ο Πίνακας 5.11 συνοψίζει τα αποτελέσματα αυτά.

P	Περάσματα
2	$1,504 \times \ln nsg + 0,992$
3	$1,102 \times \ln nsg + 0,820$
4	$0,897 \times \ln nsg + 0,800$
5	$0,773 \times \ln nsg + 0,808$
6	$0,691 \times \ln nsg + 0,882$
7	$0,632 \times \ln nsg + 0,834$
8	$0,587 \times \ln nsg + 0,845$
9	$0,552 \times \ln nsg + 0,854$

Πίνακας 5.11: Αριθμός περασμάτων σε συγχώνευση καταρράκτη.

Στο σημείο αυτό αναφέρεται ότι από τον Knuth προτάθηκε το 1963 μία υβριδική μέθοδος μεταξύ πολυφασικής και καταρράκτη. Πιο συγκεκριμένα, σε κάθε φάση της μεθόδου του καταρράκτη δεν εκτελούνται συγχωνεύσεις από P δρόμους μέχρι 2 δρόμους, αλλά από P μέχρι r δρόμους, όπου το r είναι παράμετρος. Στον Πίνακα 5.12 φαίνεται το βέλτιστο r για μερικές τιμές του P . Δηλαδή, για $P=2$ ή 3 η υβριδική αυτή μέθοδος ταυτίζεται με την πολυφασική, ενώ για $P \geq 14$ η μέθοδος ταυτίζεται με τον καταρράκτη.

P	2, 3	4, 5	6	10	14
r	2	3	4	8	14

Πίνακας 5.12: Βέλτιστο r υβριδικής μεθόδου.

Συμπερασματικά λοιπόν, θεωρώντας ως κριτήριο επίδοσης τον όγκο των δεδομένων που μεταφέρονται από/προς τη συσκευή της ταινίας, ισχύουν οι εξής παρατηρήσεις:

- όσο μεγαλύτερος είναι ο βαθμός της συγχώνευσης, τόσο μικρότερος είναι ο αριθμός των περασμάτων,
- σημασία έχει το μέγεθος του αρχείου και όχι το πλήθος των εγγράφων ή το μέγεθος της εγγραφής,
- όσο μεγαλύτερη είναι η κύρια μνήμη τόσο η επίδοση βελτιώνεται, γιατί δημιουργούνται λιγότερα και μεγαλύτερα ταξινομημένα τμήματα,
- γενικά, η λιγότερο αποτελεσματική μέθοδος είναι η φυσική συγχώνευση, η πολυφασική μέθοδος είναι καλύτερη από την ισοζυγισμένη

όταν οι διαθέσιμες συσκευές είναι λιγότερες από 9, ενώ η συγχώνευση καταρράκτη είναι καλύτερη από την πολυφασική για περισσότερες από πέντε συσκευές.

Στα πλαίσια του κεφαλαίου αυτού έγινε αναφορά των κυριότερων αλγορίθμων και δόθηκε προσοχή στην εύρεση του αριθμού των φάσεων και των περασμάτων. Ωστόσο, ένας άλλος σημαντικός παράγοντας κόστους είναι ο χρόνος περιστροφής της ταινίας μετά το πέρας κάθε φάσης. Αξίζει πάντως να σημειωθεί ότι η μέθοδος του καταρράκτη επιτρέπει κάποια επικάλυψη του χρόνου επαναφοράς της ταινίας με το χρόνο των παραγωγικών συγχωνεύσεων. Πρέπει επίσης να σημειωθεί ότι οι μέθοδοι αποχτούν νέο αλγοριθμικό και αναλυτικό ενδιαφέρον αν θεωρηθούν συσκευές με δυνατότητα ανάγνωσης/αποθήκευσης και κατά τις δύο φορές κίνησης. Μάλιστα για τις ταινίες αυτού του τύπου αναφέρεται μία ακόμη μέθοδος ταξινόμησης με πολύ καλή επίδοση, η *ταλαντευόμενη συγχώνευση* (oscillating merge), που προτάθηκε το 1962 από τον Sobel. Ένα άλλο ενδιαφέρον πρόβλημα είναι η εύρεση αλγορίθμων εξωτερικής ταξινόμησης, όταν είναι διαθέσιμες συνολικά δύο ή και μόνο μία περιφερειακή συσκευή. Για τις δύο αυτές περιπτώσεις, μάλιστα, έχει αποδειχθεί ότι δεν μπορεί να κατασκευασθεί αλγόριθμος μικρότερης πολυπλοκότητας από $O(n \log n)$ και $O(n^2)$, αντίστοιχα. Το θέμα της ταξινόμησης με ταινίες έχει πλούσια βιβλιογραφία αλλά δεν είναι πλέον καυτό θέμα για την Πληροφορική. Ο ενδιαφερόμενος αναγνώστης μπορεί να βρει πλούσιο υλικό στον τρίτο τόμο του έργου του Knuth 'The art of computer programming: sorting and searching'.

5.6 Ασκήσεις

<1> Δοθείσων $2P$ ταινιών, ισοζυγισμένη συγχώνευση μπορεί να γίνει θεωρώντας αρχικά X συσκευές εισόδου και $2P-X$ συσκευές εξόδου, ενώ στη συνέχεια ο ρόλος τους αντιστρέφεται. Να βρεθεί η επίδοση της μεθόδου ως συνάρτηση του X και η βέλτιστη τιμή του X .

<2> Έστωσαν 14 εγγραφές με κλειδιά 6, 29, 1, 10, 23, 48, 17, 13, 16, 12, 11, 7, 2 και 3. Να υποθεθεί ότι τα αρχικά ταξινομημένα τμήματα περιέχουν μόνο μία εγγραφή και να εφαρμοσθεί η:

- φυσική συγχώνευση δύο δρόμων,
- ισοζυγισμένη συγχώνευση δύο δρόμων,

- ισοζυγισμένη συγχώνευση τριών δρόμων, και
- πολυφασική ταξινόμηση τριών δρόμων.

Για κάθε περίπτωση να φανεί η δομή του αρχείου μετά από κάθε φάση.

<3> Να δημιουργηθούν πίνακες με το περιεχόμενο των συσχευών για 31 ταξινομημένα τμήματα και για τις περιπτώσεις:

- φυσική συγχώνευση δύο δρόμων,
- ισοζυγισμένη συγχώνευση δύο δρόμων,
- ισοζυγισμένη συγχώνευση τριών δρόμων, και
- πολυφασική ταξινόμηση δύο δρόμων.

Να υπολογισθεί για κάθε περίπτωση ο αριθμός των φάσεων και των περασμάτων.

<4> Να εφαρμοσθεί η πολυφασική συγχώνευση τεσσάρων δρόμων σε 100 ταξινομημένα τμήματα των 100 εγγραφών θεωρώντας εικονικά τμήματα, ώστε να βρεθεί ένας γενικευμένος αριθμός Fibonacci. Να δοκιμασθούν οι εξής περιπτώσεις:

- τα εικονικά τμήματα τίθενται εμπρός από τα πραγματικά,
- τα εικονικά τμήματα τίθενται μετά τα πραγματικά.

Τι παρατηρείτε όσον αφορά στην επίδοση της μεθόδου;

<5> Να δοκιμασθεί η πολυφασική συγχώνευση και η συγχώνευση καταρράκτη για μερικές τιμές του αριθμού των αρχικών τμημάτων (για παράδειγμα $n_{sg} = 100, 150, 200$) με σκοπό να βρεθεί για ποιόν αριθμό ταινιών ($P = 8, 9, 10, 11$) η μία μέθοδος είναι καλύτερη της άλλης.

<6> Ένα αρχείο αποτελείται από 2028 ταξινομημένα τμήματα των 4000 bytes. Το μέγεθος της εγγραφής είναι 200 bytes, ενώ το μέγεθος της απομονωτικής μνήμης είναι 16 Mb. Να υπολογισθεί ο απαιτούμενος χρόνος για είσοδο/έξοδο των δεδομένων κατά την εκτέλεση μίας ισοζυγισμένης συγχώνευσης P δρόμων ($P = 4, 8, 12, 16, 20, 24$). Δίνεται επίσης ότι το μήκος της ταινίας είναι 2400 πόδια, η πυκνότητα είναι 1600 bytes, το κενό έχει μήκος 0,6 ίντσες και η ταχύτητα είναι 10 πόδια/sec.

<7> Να θεωρηθούν τα δεδομένα της Άσκησης 6 και να υπολογισθεί ο απαιτούμενος χρόνος για πολυφασική συγχώνευση.

Κεφάλαιο 6

ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΔΙΣΚΟΥΣ

- 6.1 Εισαγωγή
- 6.2 Ταξινόμηση με μία συσκευή δίσκων
- 6.3 Ταξινόμηση με δύο συσκευές δίσκων
- 6.4 Συγχώνευση με μία συσκευή δίσκων
- 6.5 Συγχώνευση με δύο συσκευές δίσκων
- 6.6 Άλλες μέθοδοι εξωτερικής συγχώνευσης
- 6.7 Ασκήσεις

Κεφάλαιο 6

ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΔΙΣΚΟΥΣ

6.1 Εισαγωγή

Όπως είναι γνωστό, αν το αρχείο δεν χωρά στην κύρια μνήμη, τότε η εξωτερική ταξινόμηση γίνεται σε δύο στάδια. Αρχικά το αρχείο υποδιαιρείται σε τμήματα που χωρούν στην κύρια μνήμη και ταξινομούνται με μία μέθοδο εσωτερικής ταξινόμησης, οπότε συγχωνεύονται στη συνέχεια. Σε κάθε στάδιο απαιτείται ανάγνωση και αποθήκευση του αρχείου τουλάχιστο μία φορά. Με τη χρήση μεγάλης κυρίας μνήμης, δύο συσκευών δίσκων και υποστήριξη απομονωτικών μνημών για την επικάλυψη του χρόνου προσπέλασης κάποιας εγγραφής από το δίσκο με το χρόνο αποθήκευσης κάποιας άλλης στο δίσκο, είναι δυνατό να επιτευχθούν πολύ αποτελεσματικές μέθοδοι.

Έστω ότι δίνεται ένα αρχείο 6.000.000 εγγραφών των 400 bytes. Έτσι το αρχείο έχει μέγεθος 2,4 Gb, δηλαδή είναι ένα μεγάλο αρχείο που μπορεί, για παράδειγμα, να συναντηθεί συχνά στην πράξη σε εφαρμογές τραπεζών, δημοσίων οργανισμών ή αεροπορικών εταιρειών. Δεδομένου ότι δεν υπάρχει υπολογιστής με τόσο μεγάλη κύρια μνήμη, θα υποτεθεί ότι η διαθέσιμη κύρια μνήμη είναι 10 Mb.

Αν υπάρχει μόνο μία συσκευή δίσκου, τότε η καλύτερη επιλογή για το πρώτο στάδιο είναι η χρησιμοποίηση του αλγορίθμου ταξινόμησης σωρού (heapsort), που παράγει ταξινομημένα τμήματα μεγέθους ίσου με το μέγεθος της κύριας μνήμης. Ο αλγόριθμος αυτός, σε αντίθεση με όλους τους γνωστούς, είναι ο μόνος που με τη βοήθεια απομονωτικών μνημών επιτρέπει επικάλυψη της εισόδου με την έξοδο. Συνεπώς, σε τελική ανάλυση το

συνολικό χρονικό κόστος ισούται με το κόστος για είσοδο και έξοδο των δεδομένων, ενώ η καθ' εαυτή ταξινόμηση γίνεται παράλληλα και δεν επιβαρύνει χρονικά.

Αν υπάρχουν δύο διαθέσιμες συσκευές δίσκου, τότε χρησιμοποιείται η τεχνική ταξινόμησης που ονομάζεται **επιλογή αντικατάστασης** (replacement selection). Αυτή η μέθοδος παράγει τμήματα με διπλάσιο μέγεθος από το μέγεθος της διαθέσιμης κύριας μνήμης, κατά μέσο όρο. Επιπλέον, με τη βοήθεια απομονωτικών μνημών, η προσπέλαση από τον ένα δίσκο μπορεί να επικαλυφθεί χρονικά με την αποθήκευση στον άλλο. Έτσι τελικά, το πρώτο στάδιο της ταξινόμησης των τμημάτων απαιτεί χρόνο ίσο περίπου με την προσπέλαση του αρχείου.

Το δεύτερο στάδιο της συγχώνευσης είναι παρόμοιο προς τη συγχώνευση δομών κύριας μνήμης και συνεπώς αρκετά απλό. Αν υποθεθεί ότι δίνονται 100 ταξινομημένα τμήματα, τότε από κάθε τμήμα λαμβάνεται μία εγγραφή και εκτελείται μία συγχώνευση 100 δρόμων. Όταν μία εγγραφή κάποιου τμήματος κατευθυνθεί στην έξοδο, τότε αυτή αντικαθίσταται από την επόμενη του ίδιου τμήματος και έτσι η διαδικασία προχωρεί επαναληπτικά μέχρι να εξαντληθούν όλα τα τμήματα.

Η χρήση περισσότερων από δύο συσκευών δίσκου δεν εξυπηρετεί τόσο σημαντικά στη μείωση του χρόνου ταξινόμησης, όσο η χρήση των όλο και μεγαλύτερων και φθηνότερων κύριων μνημών. Επειδή στο παρελθόν δεν υπήρχαν μεγάλες κύριες μνήμες, χρησιμοποιούνταν άλλοι αλγόριθμοι εξωτερικής ταξινόμησης (πχ. η πολυφασική συγχώνευση). Στο τέλος του παρόντος κεφαλαίου οι μέθοδοι ταξινόμησης με ταινίες εξετάζονται και κοινοτολογούνται για την εφαρμογή τους σε δίσκους. Ωστόσο, σημειώνεται προκαταβολικά ότι μία μέθοδος που θα παρουσιασθεί στη συνέχεια, και η οποία απαιτεί δύο συσκευές δίσκων, ξεπερνά τη μέθοδο της πολυφασικής συγχώνευσης ακόμη και αν αυτή εφαρμοσθεί με 10 ή 20 συσκευές δίσκων.

6.2 Ταξινόμηση με μία συσκευή δίσκων

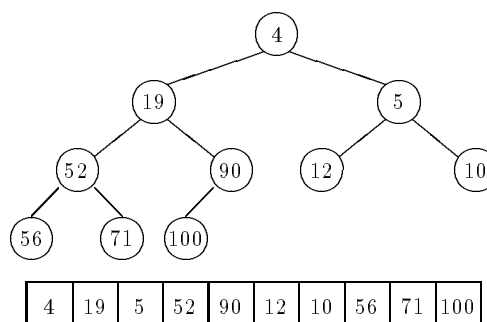
Όταν υπάρχει διαθέσιμη μόνο μία συσκευή δίσκων, τότε στο πρώτο στάδιο χρησιμοποιείται ο αλγόριθμος της ταξινόμησης σωρού με σκοπό τη δημιουργία των ταξινομημένων τμημάτων. Η γνωστή αυτή μέθοδος προτάθηκε από τον Williams το 1964, ενώ μία παραλλαγή γνωστή ως **ταξινόμηση δένδρου** (treesort) προτάθηκε από τον Floyd το 1967. Ο αλγόριθμος αυτός είναι ο

μοναδικός από τους αλγορίθμους εσωτερικής ταξινόμησης που επιτρέπει την επικάλυψη της εισόδου με την έξοδο.

Σχεδόν πλήρες δυαδικό δένδρο (almost complete binary tree) είναι εκείνο το δυαδικό δένδρο που:

1. όλα τα φύλλα του βρίσκονται σε δύο διαδοχικά επίπεδα, έστω το k -οστό και $(k+1)$ -οστό επίπεδο,
2. όλοι οι κόμβοι του επιπέδου $(k-2)$ έχουν βαθμό δύο, και
3. τα φύλλα του k -οστού επιπέδου καταλαμβάνουν τις αριστερότερες θέσεις του επιπέδου.

Σωρός μεγίστων (max heap) είναι το σχεδόν πλήρες δυαδικό δένδρο, όπου η τιμή του κλειδιού κάθε κόμβου είναι μικρότερη από τις τιμές των κλειδιών των παιδιών του συγκεκριμένου κόμβου (δες βιβλίο για Δομές Δεδομένων, Κεφάλαιο 6.2). Έτσι είναι βέβαιο ότι στη ρίζα είναι πάντα αποθηκευμένο το μικρότερο κλειδί. Στο Σχήμα 6.1 παρουσιάζεται ένας σωρός.



Σχήμα 6.1: Σωρός ως δυαδικό δέντρο και ως διάνυσμα.

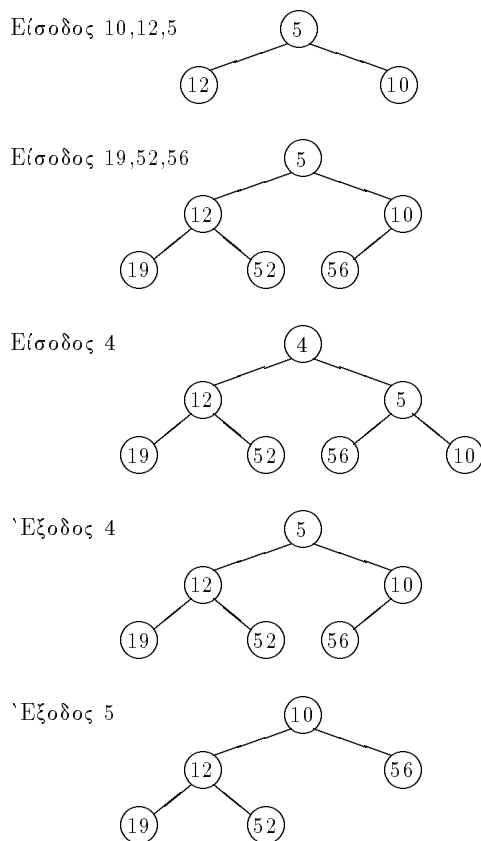
Ένα πλήρες δυαδικό δένδρο πολύ εύκολα μπορεί να παρασταθεί με έναν πίνακα, τοποθετώντας το περιεχόμενο των κόμβων στον πίνακα κατά επίπεδα και από αριστερά προς τα δεξιά. Δηλαδή, στην πρώτη θέση του πίνακα τοποθετείται η ρίζα, στη δεύτερη το αριστερό παιδί της ρίζας, στην τρίτη το δεξί παιδί της ρίζας, κοκ. Άρα, τα παιδιά ενός κόμβου που βρίσκεται στην i -οστή θέση του πίνακα είναι αποθηκευμένα στις θέσεις $2i$ και $2i+1$, ενώ ο πατέρας του κόμβου αυτού είναι αποθηκευμένος στη θέση $\lfloor i/2 \rfloor$.

Η βασική ιδέα του αλγορίθμου ταξινόμησης σωρού είναι η εξής. Η εισαγόμενη εγγραφή καταλαμβάνει την αριστερότερη θέση του τελευταίου επιπέδου του δένδρου (ή με άλλα λόγια την πρώτη κενή θέση του πίνακα). Το κλειδί της νέας εγγραφής συγκρίνεται προς το κλειδί του κόμβου πατέρα. Αν το νέο κλειδί είναι μικρότερο, τότε πρέπει οι δύο εγγραφές να ανταλλάχθούν, ώστε να ισχύει η βασική ιδιότητα των σωρών. Αυτή η διαδικασία σύγκρισης και αναρρίχησης μπορεί να επαναληφθεί, ίσως και μέχρι τη ρίζα. Η διαδοχική ανάγνωση εγγραφών και η τοποθέτησή τους στο σωρό συνεχίζεται μέχρι να γεμίσει ο διαθέσιμος χώρος στην κύρια μνήμη. Έτσι το περιεχόμενο της κύριας μνήμης αποτελεί ένα ταξινομημένο τμήμα.

Κατόπιν η ρίζα του σωρού μεταφέρεται στην έξοδο, ενώ η τελευταία εγγραφή του σωρού καταλαμβάνει τη θέση της ρίζας. Πλέον δεν ισχύει ο ορισμός του σωρού και πρέπει το κλειδί αυτό να συγκριθεί με τα κλειδιά των δύο παιδιών του. Το μικρότερο κλειδί ανταλλάσσεται με την εγγραφή της ρίζας που (δυνητικά) μπορεί να συνεχίσει την καθοδική πορεία μέχρι το τελευταίο επίπεδο. Μετά από όλη αυτή την επαναληπτική διαδικασία, ο ορισμός του σωρού αποκαθίσταται και συνεπώς η νέα ρίζα μπορεί να μεταφερθεί στην έξοδο. Κατ' αυτόν τον τρόπο ο αλγόριθμος συνεχίζει μέχρι τη συνολική μεταφορά όλου του σωρού στην έξοδο. Στη συνέχεια η μνήμη δέχεται νέες εγγραφές από το δίσκο για τη δημιουργία του νέου τμήματος.

Στο Σχήμα 6.2 δίνεται ένα παράδειγμα, υποθέτοντας ότι στη μνήμη χωρούν 7 εγγραφές. Έτσι στη μνήμη εισάγονται 7 εγγραφές με κλειδιά 10, 12, 5, 19, 52, 56 και 4, ενώ στη συνέχεια μεταφέρονται στην έξοδο οι εγγραφές με κλειδιά 4 και 5. Στο σχήμα παρουσιάζονται μερικά βήματα της διαδικασίας που υλοποιεί την ανάπτυξη αυτή. Κάθε δομή του σχήματος αντιστοιχεί στο σωρό μετά από την αποκατάσταση των ιδιοτήτων του (δηλαδή, μετά από τις αναγκαίες συγκρίσεις και ανταλλαγές).

Στη συνέχεια μελετάται η επίδοση του αλγορίθμου. Το ύψος ενός πλήρους δυαδικού δένδρου είναι $\log n$, όπου n είναι το πλήθος των κόμβων του δένδρου. Συνεπώς, ο αριθμός των απαραίτητων συγκρίσεων/ανταλλαγών για τη διατήρηση της βασικής ιδιότητας του σωρού είναι στη χειρότερη περίπτωση $\log n$, όπου πλέον n είναι οι μέχρι τώρα εγγραφές του σωρού. Δηλαδή, αν το κάθε τμήμα έχει μέγεθος 10 Mb και περιέχει 25.000 εγγραφές, τότε το ύψος του δένδρου είναι μεταξύ 14 και 15. Άρα, μία εισαγωγή εγγραφής στο σωρό μπορεί να προκαλέσει το μέγιστο 14 ανταλλαγές. Ακόμη αν υποθεθεί ότι η είσοδος συνίσταται στην προσπέλαση σελίδων μεγέθους 2400 bytes με συντελεστή ομαδοποίησης 6, τότε πρέπει να γίνουν το μέγιστο 84



Σχήμα 6.2: Ταξινόμηση σωρού με επικάλυψη I/O. Η κύρια μνήμη χωρά 7 εγγραφές.

ανταλλαγές μέχρι να εισαχθεί η επόμενη σελίδα. Με την προϋπόθεση ότι μία σύγκριση/ανταλλαγή απαιτεί 10 μ s έπεται ότι στη χειρότερη περίπτωση απαιτούνται 0.84 ms Αυτός ο χρόνος προσεγγίζει τον απαιτούμενο χρόνο για την προσπέλαση μίας σελίδας των 2400 bytes σε μία συσκευή IBM 3380.

Όταν η ρίζα μεταφερθεί στην απομονωτική μνήμη εξόδου για την αποθήκευση στο δίσκο, η τελευταία εγγραφή μεταφέρεται στη θέση της ρίζας, οπότε απαιτούνται νέες συγκρίσεις/ανταλλαγές. Ο αριθμός των ανταλλαγών αυτών είναι και πάλι της τάξης $\log n$, όπου n είναι οι εγγραφές που έχουν απομείνει στο δένδρο. Έτσι και στις δύο φάσεις μπορεί να υπάρξει πολύ μεγάλη επικάλυψη στους χρόνους εισόδου και εξόδου.

Στη συσκευή IBM 3380 ο μέσος χρόνος εντοπισμού και ο μέσος χρόνος περιστροφής είναι 24,3 ms, ενώ ο χρόνος για τη σειριακή προσπέλαση 10 Mb δεδομένων είναι 3500 ms. Έτσι, κατά τους υπολογισμούς αγνοείται ο χρόνος εντοπισμού, και τελικά ο χρόνος για τη δημιουργία των ταξινομημένων τμημάτων ισούται με:

$$2b \times ebt$$

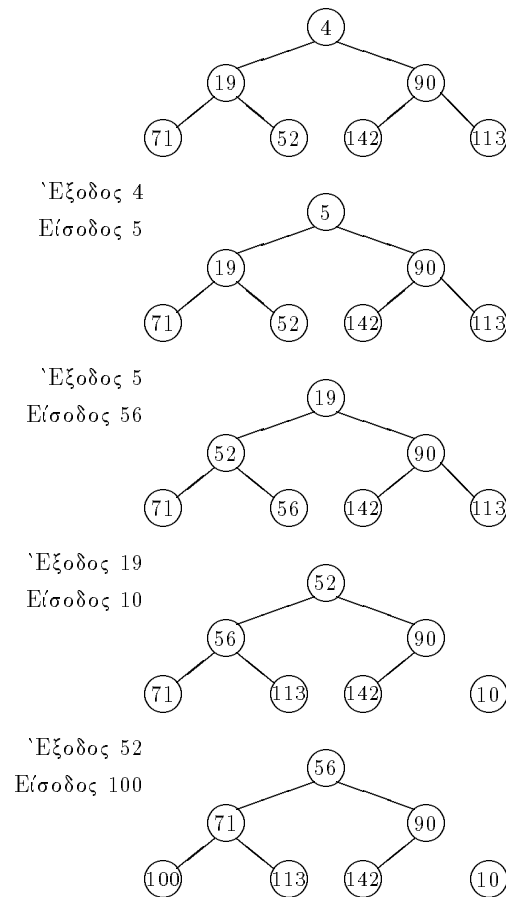
Ο χρόνος αυτός περιλαμβάνει το χρόνο για την προσπέλαση των τμημάτων και το χρόνο αποθήκευσης των ταξινομημένων τμημάτων και πάλι στο δίσκο.

6.3 Ταξινόμηση με δύο συσκευές δίσκων

Αν υπάρχουν δύο διαθέσιμες συσκευές δίσκων, τότε είναι δυνατό να προκύψουν ταξινομημένα τμήματα μεγαλύτερα από το μέγεθος της διαθέσιμης κύριας μνήμης. Μάλιστα έχει αποδειχθεί ότι το μέγεθος των τμημάτων είναι διπλάσιο της κύριας μνήμης κατά μέσο όρο. Άλλο σημαντικότερο πλεονέκτημα είναι ότι όλοι οι απαιτούμενοι χρόνοι για την προσπέλαση, την ταξινόμηση και την επαναποθήκευση στο δίσκο μπορεί να επικαλυφθούν.

Στην περίπτωση αυτή, για την εσωτερική ταξινόμηση χρησιμοποιείται ο αλγόριθμος της επιλογής αντικατάστασης, που έχει πολλά κοινά σημεία με τον αλγόριθμο ταξινόμησης σωρού. Όπως και με τον τελευταίο αλγόριθμο, αρχικά δημιουργείται ένας σωρός από εγγραφές που εισάγονται από τη μία συσκευή. Ο σωρός αυτός καταλαμβάνει όλο το χώρο της διαθέσιμης κύριας μνήμης. Στη συνέχεια, οι εγγραφές με το μικρότερο κλειδί που ομαδοποιούνται σε μία σελίδα αποθηκεύονται στον άλλο δίσκο και ταυτόχρονα μία νέα σελίδα μεταφέρεται από την πρώτη συσκευή στην κύρια μνήμη.

Η νέα σελίδα μπορεί να περιέχει μερικές εγγραφές με κλειδιά μεγαλύτερα από τα κλειδιά των εγγραφών, που έχουν ήδη αποθηκευθεί στο δίσκο. Αυτές οι εγγραφές μπορούν να ενσωματωθούν στον αρχικό σωρό. Όμως η νέα σελίδα μπορεί να περιέχει και εγγραφές με κλειδιά μικρότερα από αυτά που έχουν ήδη περάσει στην έξοδο. Αυτές οι εγγραφές δεν εισάγονται στον υπάρχοντα σωρό, αλλά σχηματίζουν ένα νέο σωρό. Έτσι, στην κύρια μνήμη δημιουργούνται δύο σωροί. Στο Σχήμα 6.3 παρουσιάζεται ένα παράδειγμα εφαρμογής της μεθόδου, θεωρώντας ότι η κύρια μνήμη χωρά 7 εγγραφές. Η αρχική δομή παρουσιάζει τον αρχικό σωρό των 7 εγγραφών, αλλά στη συνέχεια η έξοδος και η είσοδος γίνονται ταυτόχρονα ανά μία εγγραφή.



Σχήμα 6.3: Επιλογή αντικατάστασης με επικάλυψη I/O. Η κύρια μνήμη χωρά 7 εγγραφές.

Οι διαδικασίες εισόδου/εξόδου μπορούν να συντονισθούν. Σε αντίθεση με τη μέθοδο ταξινόμησης σωρού, στη μέθοδο αυτή όταν μία εγγραφή περνά στην απομονωτική μνήμη εξόδου, τότε τη θέση της στη ρίζα του σωρού δεν καταλαμβάνει η τελευταία εγγραφή του πίνακα αλλά μία νέα εγγραφή. Έτσι η είσοδος και η έξοδος γίνονται ταυτόχρονα σε ένα βήμα με επικάλυψη των σχετικών χρόνων. Συνεπώς είναι βέβαιο ότι ο νέος σωρός θα γίνει μεγαλύτερος από τον αρχικό, επειδή οι περισσότερες από τις εισαγόμενες εγγραφές θα έχουν μικρότερα κλειδιά από τις ήδη περασμένες στο δίσκο.

Συνήθως στο σημείο αυτό δεν υπάρχει μεγάλη επιχώληση μεταξύ εισόδου και εξόδου, επειδή οι περισσότερες νέες εγγραφές κατευθύνονται στο νέο σωρό. Σε κάποια χρονική στιγμή ο ένας σωρός θα αδειάσει, οπότε στο σημείο αυτό τελειώνει η δημιουργία ενός ταξινομημένου τμήματος. Έτσι η διαδικασία συνεχίζεται με το νέο σωρό και τη δημιουργία του επόμενου ταξινομημένου τμήματος. Η χρήση μεγαλύτερων σελίδων και απομονωτικών μνημών μετριάξει τη δυσμενή επίπτωση του φαινομένου αυτού.

Αν οι εγγραφές έρχονται περίπου ταξινομημένες, τότε τα προκύπτοντα τμήματα θα είναι πολύ μεγάλα. Αντίθετα, αν οι εγγραφές που έρχονται στην είσοδο είναι σχεδόν διατεταγμένες κατά φθίνουσα τάξη, τότε τα τμήματα θα είναι μικρά. Έχει αποδειχθεί ότι για τιμές κλειδιών που υπαχούν σε τυχαία κατανομή, το μέγεθος των ταξινομημένων τμημάτων είναι περίπου διπλάσιο από το μέγεθος της διαθέσιμης κύριας μνήμης.

Η επιλογή αντικατάστασης είναι μία μέθοδος εξαιρετικά αποτελεσματική όταν υπάρχουν δύο συσχευές δίσκων. Έτσι αμέσως μετά την αρχική φόρτωση της κύριας μνήμης με τις πρώτες σελίδες, η είσοδος με την έξοδο μπορούν να γίνουν παράλληλα στις δύο συσχευές. Αν υπάρχει μόνο μία συσχευή, τότε η μέθοδος δεν είναι αποτελεσματική, γιατί ο βραχίονας του δίσκου θα μετακινείται συνεχώς εμπρός και πίσω για να εκτελέσει τις διαδοχικές αναγνώσεις και αποθηκεύσεις. Ο απαιτούμενος χρόνος από τη μέθοδο αυτή είναι:

$$b \times ebt$$

χωρίς να λαμβάνεται υπ' όψη ο χρόνος για την προσπέλαση των πρώτων σελίδων που εκτελείται χωρίς παραλληλισμό στην έξοδο, επειδή θεωρείται μικρός σε σχέση με την προσπέλαση όλου του αρχείου.

Σε τελική ανάλυση, η μέθοδος επιλογής αντικατάστασης σε σχέση με τη μέθοδο ταξινόμησης σωρού υπερτερεί σε δύο σημεία.

- τα παραγόμενα τμήματα έχουν κατά μέσο όρο διπλάσιο μέγεθος, και
- το χρονικό κόστος υποδιπλασιάζεται.

Όμως από την άλλη πλευρά υπάρχει επιπρόσθετο κόστος επειδή απαιτούνται δύο συσχευές δίσκων.

6.4 Συγχώνευση με μία συσκευή δίσκων

Στο σημείο αυτό πλέον αρχίζει η δεύτερη φάση, όπου τα ταξινομημένα τμήματα πρέπει να συγχωνευθούν ώστε να προκύψει η τελική ταξινομημένη μορφή του αρχείου. Στη συνέχεια υποτίθεται ότι διαθέσιμη υπάρχει μία συσκευή δίσκων με χωρητικότητα τουλάχιστο διπλάσια από το μέγεθος του αρχείου των 2,4 Gb. Αρχικά, λοιπόν, χρησιμοποιείται ο αλγόριθμος ταξινόμησης σωρού, οπότε παράγονται 240 ταξινομημένα τμήματα μεγέθους 10 Mb. Ο απαιτούμενος χρόνος για τη σειριακή προσπέλαση αρχείου αυτού του μεγέθους είναι περίπου 14 λεπτά. Συνεπώς, ο απαιτούμενος χρόνος για τη δημιουργία των τμημάτων είναι 28 λεπτά.

	5 Mb					5 Mb				
A	4	10	19	52	71	100	146	150	193	202
B	5	12	56	90	113	142	175	215	220	225

Μνήμη A					Μνήμη B					Έξοδος
4	10	19	52	71	5	12	56	90	113	
	10	19	52	71	5	12	56	90	113	4
	10	19	52	71		12	56	90	113	5
		19	52	71		12	56	90	113	10
		19	52	71			56	90	113	12
			52	71			56	90	113	19
				71			56	90	113	52
				71				90	113	56
								90	113	71
100	146	150	193	202				90	113	
100	146	150	193	202					113	90
	146	150	193	202					113	100
	146	150	193	202						113
	146	150	193	202	142	175	215	220	225	
		146	150	193		175	215	220	225	142
			150	193		175	215	220	225	146
			⋮				⋮			⋮

Σχήμα 6.4: Συγχώνευση δύο ταξινομημένων τμημάτων των 10 Mb με κύρια μνήμη των 10 Mb.

Η πρώτη μέθοδος συγχώνευσης που θα μελετηθεί είναι η συγχώνευση δύο δρόμων. Η μέθοδος αυτή κάθε φορά συγχωνεύει δύο μόνο ταξινομημένα τμήματα. Επειδή η διαθέσιμη μνήμη είναι 10 Mb, δεν είναι δυνατό να μεταφερθούν στη μνήμη δύο τμήματα την ίδια στιγμή. Για το λόγο αυτό υποτίθεται ότι υπάρχουν απομονωτικές μνήμες των 5 Mb, όπου φορτώνονται τα πρώτα μισά των δύο τμημάτων. Ακόμη, επειδή διατίθεται μία μόνο συσκευή δίσκου δεν είναι δυνατόν τα τμήματα να προσπελασθούν με επικάλυψη των χρόνων. Μόνο κατά τη φάση της εξόδου είναι δυνατό να εφαρμοσθεί η τεχνική της χρήσης διπλής απομονωτικής μνήμης και να επιτευχθεί παραλληλισμός. Το Σχήμα 6.4 παρουσιάζει ένα παράδειγμα συγχώνευσης, όπου φαίνεται πως γίνεται διαδοχικά η είσοδος και η έξοδος.

Έστω, λοιπόν, ότι κατά τη διαδικασία της συγχώνευσης εξαντλείται το μισό ενός τμήματος. Τότε το δεύτερο μισό του ίδιου τμήματος έρχεται στη μνήμη και η συγχώνευση συνεχίζεται με την παράλληλη έξοδο. Τελικά, ο απαιτούμενος χρόνος για τη συγχώνευση δύο ταξινομημένων τμημάτων είναι:

$$(4 + 3) \times (r + s) + 2 \times 2 \times seg \times ebt$$

όπου *seg* είναι ο αριθμός των σελίδων κάθε ταξινομημένου τμήματος. Το πρώτο γινόμενο, προφανώς, δίνει το χρόνο εντοπισμού και περιστροφής. Ο συντελεστής 4 αντιστοιχεί στις τέσσερις προσπελάσεις στο δίσκο για τα ημίσεα των τμημάτων, ενώ ο συντελεστής 3 αντιστοιχεί στις φορές που γίνεται αποθήκευση στο δίσκο (οι δύο πρώτες αναγνώσεις γίνονται πριν την πρώτη αποθήκευση).

Αν στις παραμέτρους δοθούν οι γνωστές τιμές, τότε το πρώτο γινόμενο ισούται με 170 ms, ενώ το δεύτερο ισούται με 14000 ms. Αν και το κόστος που προκύπτει από το πρώτο ημιάθροισμα δεν είναι τόσο μεγάλο όσο το κόστος από το δεύτερο, εντούτοις διατηρείται στον τύπο. Ο λόγος είναι ότι, στη συνέχεια θα δοθούν μέθοδοι συγχώνευσης περισσότερων από δύο δρόμους εκεί θα υπάρξει διαφοροποίηση.

Ο απαιτούμενος χρόνος για το πρώτο πέρασμα συγχώνευσης των 120 ζευγών τμημάτων των 10 Mb είναι περίπου 28.4 λεπτά. Στα επόμενα περάσματα θα πρέπει να συγχωνευθούν 60 ζεύγη τμημάτων των 20 Mb, 30 ζεύγη των 40 Mb, κοκ. Δηλαδή, θα χρειασθούν συνολικά $\lceil \log 240 \rceil$ περάσματα. Στον Πίνακα 6.1 φαίνεται για κάθε πέρασμα πόσα τμήματα υπάρχουν και ποιο είναι το μέγεθός τους.

Μετά το πρώτο πέρασμα ο αριθμός των προσπελάσεων στο δίσκο για αποθήκευση προσεγγίζει σχετικά περισσότερο προς τον αριθμό των προσπε-

Πέρασμα	1	2	3	4	5	6	7	8
Μέγεθος τμήματος (Mb)	10	20	40	80	160	7×320 + 160	3×640 + 480	1280
Αριθμός	240	120	60	30	15	8	4	2

Πίνακας 6.1: Συγχώνευση δύο δρόμων για 240 τμήματα.

λάσεων για ανάγνωση. Έτσι, όταν κατά το τέταρτο πέρασμα το μέγεθος των τμημάτων είναι 80 Mb χρειάζονται 32 αναγνώσεις και 31 αποθηκεύσεις για τη συγχώνευση δύο τμημάτων. Για το λόγο αυτό, τελικά γίνεται δεκτό ότι για κάθε κομμάτι τμήματος που έρχεται στην απομονωτική μνήμη εισόδου αντιστοιχεί ακριβώς μία προσπέλαση στο δίσκο. Αν, λοιπόν, το αρχείο αποτελείται από nsg τμήματα, τότε κάθε πέρασμα απαιτεί χρόνο:

$$2 \times 2 \times nsg \times (r + s) + 2 \times b \times ebt$$

Το κόστος αυτό πρέπει να πολλαπλασιασθεί επί τον αριθμό των περασμάτων, για να δώσει το συνολικό χρονικό κόστος των περασμάτων. Επίσης στο ποσό αυτό πρέπει να προστεθεί και ο αρχικός χρόνος για την ταξινόμηση των τμημάτων. Για το συγκεκριμένο παράδειγμα, λοιπόν, προκύπτει ότι συνολικά απαιτούνται:

$$8 \times 28,4 + 28 = 255 \text{ λεπτά ή } 4,25 \text{ ώρες}$$

Στη συνέχεια θα μελετηθεί η συγχώνευση τεσσάρων δρόμων. Επειδή θα εξετασθεί μόνο η δεύτερη φάση της συγχώνευσης, υποτίθεται ότι τα τμήματα είναι ήδη ταξινομημένα. Αρχικά, λοιπόν, από τέσσερα ταξινομημένα τμήματα μεταφέρονται στην κύρια μνήμη τα πρώτα τέταρτά τους που συγχωνεύονται κατά το γνωστό τρόπο. Μετά το πρώτο πέρασμα θα προκύψουν 60 τμήματα των 40 Mb. Το δεύτερο πέρασμα θα δώσει 15 τμήματα των 160 Mb, ενώ το τρίτο πέρασμα θα δώσει 4 τμήματα. Συνεπώς, χρειάζονται συνολικά 4 περάσματα. Ωστόσο, επειδή κάθε φορά έρχονται στην κύρια μνήμη μικρά κομμάτια (τέταρτα) των τμημάτων, το κόστος του χρόνου εντοπισμού και περιστροφής είναι σχετικά μεγαλύτερο. Ο απαιτούμενος χρόνος για ένα πέρασμα της μεθόδου αυτής είναι:

$$2 \times 4nsg \times (r + s) + 2 \times b \times ebt$$

που για το συγκεκριμένο παράδειγμα δίνει 28,8 λεπτά. Άρα για τα τέσσερα περάσματα μαζί με το κόστος της αρχικής ταξινόμησης των τμημάτων

απαιτούνται:

$$4 \times 28,8 + 28 = 143 \text{ λεπτά}$$

δηλαδή 2,4 ώρες, που είναι σημαντικά μικρότερος χρόνος από τον απαιτούμενο χρόνο για τη συγχώνευση δύο δρόμων.

Στη γενική μορφή, μία συγχώνευση P δρόμων για κάθε πέρασμα απαιτεί (σε ms) χρόνο ίσο με:

$$2P \times nsg \times (r + s) + 2 \times b \times ebt$$

Ο αριθμός των περασμάτων είναι $\lceil \log_P nsg \rceil$, οπότε ο συνολικός χρόνος είναι:

$$2 \times b \times ebt + \lceil \log_P nsg \rceil \times (2 \times P \times nsg \times (r + s) + 2 \times b \times ebt)$$

Αν $P=nsg$, τότε απαιτείται ένα πέρασμα και ο σχετικός χρόνος είναι:

$$4b \times ebt + 2 \times nsg^2 \times (r + s)$$

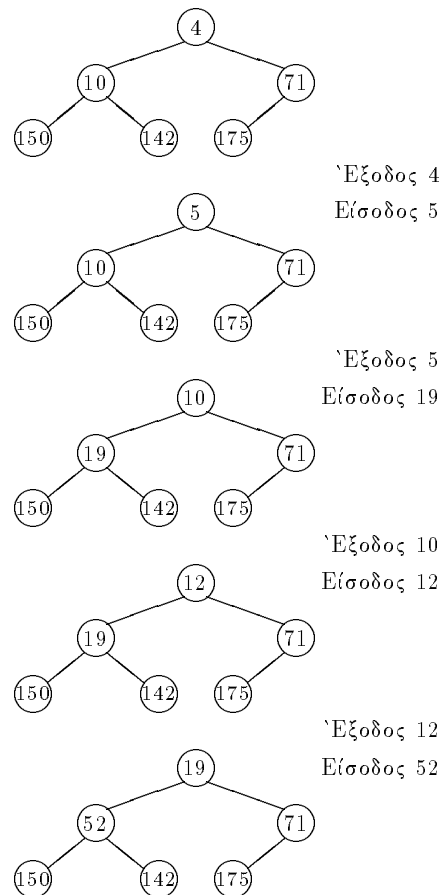
Αν χρησιμοποιηθεί ταξινόμηση 16 δρόμων, τότε το συγκεκριμένο αρχείο απαιτεί δύο περάσματα. Λαμβάνοντας υπ' όψη και τον αυξημένο χρόνο εντοπισμού και περιστροφής, τελικά προκύπτει ότι ο συνολικός χρόνος ταξινόμησης και συγχώνευσης είναι 62 λεπτά + 28 λεπτά = 1,5 ώρες.

Μέχρι στιγμής δεν αναφέρθηκε πως γίνεται η συγχώνευση. Η χρησιμοποιούμενη δομή είναι ένας σωρός, που αποτελείται από τα μικρότερα κλειδιά των τμημάτων. Ένα παράδειγμα της διαδικασίας αυτής παρουσιάζεται στο Σχήμα 6.5, όπου δίνονται 6 ταξινομημένα τμήματα των 4 εγγράφων, για να ενοποιηθούν με συγχώνευσης 6 δρόμων. Η διαδικασία στηρίζεται στη μέθοδο της επιλογής αντικατάστασης. Δηλαδή, όταν η ρίζα του σωρού μεταφέρεται στην έξοδο, τότε η εγγραφή με το μικρότερο κλειδί από το ίδιο τμήμα ενσωματώνεται στο σωρό και παίρνει τη θέση της ρίζας. Βέβαια στη συνέχεια γίνονται οι απαραίτητες συγκρίσεις/ανταλλαγές ώστε να αποκατασταθεί ο ορισμός του σωρού. Σημειώνεται ότι στο σχήμα κάθε φορά παρουσιάζεται η τελική μορφή του σωρού μετά τις απαραίτητες διεργασίες.

6.5 Συγχώνευση με δύο συσκευές δίσκων

Αν υπάρχουν διαθέσιμες δύο συσκευές δίσκων, τότε ελαττώνεται ο απαιτούμενος χρόνος, τόσο στη φάση της δημιουργίας των ταξινομημένων τμημάτων, όσο και στη φάση της συγχώνευσης. Στη δεύτερη, λοιπόν, αυτή φάση

A	4	5	19	90
B	71	100	113	225
Γ	10	12	52	56
Δ	150	193	215	232
E	142	146	250	278
ΣΤ	175	202	220	255



Σχήμα 6.5: Συγχώνευση έξι δρόμων.

χρησιμοποιείται η τεχνική της διπλής απομονωτικής μνήμης για τον παραλληλισμό της εισόδου με την έξοδο. Ωστόσο, ο χρόνος της φάσης αυτής δεν υποδιπλασιάζεται σε σχέση με το χρόνο συγχώνευσης όταν χρησιμοποιείται μία μόνο συσκευή δίσκου. Το επόμενο παράδειγμα είναι κατατοπιστικό.

Ας υποτεθεί ότι χρησιμοποιείται η συγχώνευση δύο δρόμων. Η διαθέσιμη κύρια μνήμη είναι 10 Mb, ενώ τα δύο τμήματα που θα συγχωνευθούν έχουν μέγεθος 20 Mb. Συνεπώς, η κύρια μνήμη μπορεί να χωρέσει τα πρώτα τέταρτα (5 Mb) των τμημάτων. Αν η στατιστική κατανομή των τιμών των κλειδιών στα δύο τμήματα είναι ίδια, τότε είναι πολύ πιθανό ότι τα δύο

τμήματα θα εξαντληθούν περίπου ταυτόχρονα. Έτσι, θα χρειασθούν δύο συνεχόμενες προσπελάσεις στο δίσκο για την πλήρωση των απομονωτικών μνημών με τα επόμενα τέταρτα των τμημάτων και θα δημιουργηθούν νεκροί χρόνοι αναμονής. Αυτό φαίνεται στο Σχήμα 6.6.

	— 5 Mb —				— 5 Mb —				— 5 Mb —				— 5 Mb —			
A	4	10	52	71	113	142	175	215	220	232	250	291	316	346	351	382
B	5	12	19	56	90	109	146	150	193	202	225	255	278	302	356	401

Είσοδος	Μνήμη A				Μνήμη B				Έξοδος
	10	52	71		5	12	19	56	4
	10	52	71		12	19	56		5
		52	71		12	19	56		10
		52	71			19	56		12
			52	71			56		19
				71			56		52
90				71					56
100							90		71
113						9	100		
142						9	10	113	
⋮				⋮			⋮		⋮

Σχήμα 6.6: Διπλή απομονωτική μνήμη των 5 Mb για κάθε ταξινομημένο τμήμα.

Καλύτερος συντονισμός της εισόδου με την έξοδο μπορεί να επιτευχθεί με διπλασιασμό του αριθμού των απομονωτικών μνημών για κάθε τμήμα και με ταυτόχρονο υποδιπλασιασμό του μεγέθους τους. Δηλαδή, πρέπει να χρησιμοποιηθούν μνήμες των 2,5 Mb. Έτσι, όταν έχουν συγχωνευθεί οι μισές εγγραφές από κάθε τμήμα, τότε ακολουθεί νέα είσοδος από εγγραφές του ίδιου τμήματος. Με τη μέθοδο αυτή δεν μηδενίζονται οι χρόνοι αναμονής, ωστόσο επιτυγχάνεται καλύτερη επικάλυψη. Από την άλλη πλευρά υπάρχει επιπλέον κόστος εξαιτίας της αύξησης των χρόνων εντοπισμού και περιστροφής. Η χρήση απομονωτικών μνημών των 2,5 Mb φαίνεται στο Σχήμα 6.7.

Ας θεωρηθεί και πάλι το αρχείο των 6.000.000 εγγραφών προς 400 bytes, που σημαίνει ότι το αρχείο καταλαμβάνει 2,4 Gb. Αν χρησιμοποιηθεί ο αλγόριθμος επιλογής αντικατάστασης, τότε προκύπτει ότι δημιουργούνται 120

Είσοδος	Μνήμη Α			Μνήμη Β				Έξοδος
	10	52	71	5	12	19	56	4
	10	52	71		12	19	56	5
		52	71		12	19	56	10
113		52	71			19	56	12
142	113		52				56	19
90	113	142					56	52
100	113	142		90				56
146	113	142		90	100			71
150	113	142		100		113		90
175	113	142				113	150	100
⋮		⋮			⋮			⋮

Σχήμα 6.7: Διπλή απομονωτική μνήμη των 2.5 Mb για κάθε ταξινομημένο τμήμα.

ταξινομημένα τμήματα των 20 Mb. Ο απαιτούμενος χρόνος για την παραγωγή των τμημάτων αυτών είναι 14 λεπτά περίπου.

Αν η συγχώνευση είναι 120 δρόμων, τότε θα γίνει μόνο ένα πέρασμα και κάθε τμήμα θα συνεισφέρει με το 1/120 του στο διαθέσιμο χώρο των 10 Mb. Αυτή η ποσότητα αντιστοιχεί στο 1/240 των 20 Mb των τμημάτων. Οι απομονωτικές μνήμες έχουν το μισό μέγεθος από την τελευταία ποσότητα, άρα κάθε ταξινομημένο τμήμα υποδιαιρείται σε 480 ανεξάρτητα κομμάτια που πρέπει να εισαχθούν στην κύρια μνήμη.

Ακόμη υποτίθεται ότι όλα τα τμήματα είναι αποθηκευμένα στη μία συσκευή, ενώ η αποθήκευση γίνεται στην άλλη συσκευή. Κάθε φορά που ένα κομμάτι κάποιου τμήματος εξαντλείται, τότε ένα άλλο κομμάτι του ίδιου τμήματος έρχεται από την πρώτη συσκευή στην κύρια μνήμη. Ο συνολικός αριθμός κομματιών είναι $120 \times 480 = 57.600$. Ο χρόνος εντοπισμού και περιστροφής κυριαρχεί σε σχέση με το χρόνο μεταφοράς δεδομένων. Προκύπτει ότι ο συνολικός απαιτούμενος χρόνος (σε ms) για τη συγχώνευση είναι:

$$120 \times 480 \times (s + r) + b \times ebt = 120 \times 480 \times 24,3 + 1000.000 \times 0,84$$

δηλαδή 37,3 λεπτά. Σε σχέση, λοιπόν, με τα 62 λεπτά της συγχώνευσης με μία συσκευή έχει επιτευχθεί σημαντική βελτίωση.

Για να ολοκληρωθεί η συγχώνευση στο συγκεκριμένο αρχείο με δύο περάσματα, πρέπει η συγχώνευση να είναι 11 δρόμων, ενώ κάθε ταξινομημένο τμήμα να αποτελείται από 44 κομμάτια. Με τη μέθοδο αυτή απαιτείται η παρουσία δύο κομματιών από κάθε τμήμα στην κύρια μνήμη, άρα οι απομονωτικές μνήμες πρέπει να έχουν μέγεθος $1/22$ της διαθέσιμης μνήμης. Γενικά, για μία συγχώνευση P δρόμων απαιτούνται $4P$ κομμάτια από κάθε τμήμα και συνεπώς ο συνολικός αριθμός προσπελάσεων στο δίσκο είναι $4P \times nsg$. Ο απαιτούμενος χρόνος για συγχώνευση 11 δρόμων είναι:

$$44 \times 120 \times 24,3 + 840.000 \text{ ms}$$

που ισούται με 968 sec. Επομένως, για δύο περάσματα απαιτούνται 1936 sec ή 32,3 λεπτά. Προσθέτοντας και το χρόνο των 14 λεπτών που απαιτείται για την αρχική ταξινόμηση των τμημάτων προκύπτει ένα συνολικό χρονικό κόστος της τάξης των 46 περίπου λεπτών. Αυτός ο χρόνος είναι περίπου μισός του χρόνου των 90 λεπτών που απαιτούνται για την ταξινόμηση και συγχώνευση δύο περασμάτων με μία συσκευή δίσκων.

Το χρονικό κόστος για την αρχική ταξινόμηση με επιλογή αντικατάστασης και στη συνέχεια τη συγχώνευση P δρόμων δίνεται από τη σχέση:

$$b \times ebt + [\log_P nsg] \times (4P \times nsg \times (r + s) + b \times ebt)$$

Σε κάθε πέρασμα αντιστοιχούν $4P \times nsg$ προσπελάσεις στο δίσκο για ανάγνωση και ισάριθμες για αποθήκευση. Οι προσπελάσεις για αποθήκευση στο δίσκο γίνονται παράλληλα, γι' αυτό το λόγο και δεν υπεισέρχεται στον υπολογισμό του κόστους. Αν η συγχώνευση γίνεται με ένα πέρασμα, δηλαδή $P=nsg$, τότε ο αντίστοιχος χρόνος είναι:

$$2b \times ebt + 4nsg^2 \times (r + s)$$

Αν υπάρχουν διαθέσιμες περισσότερες από δύο συσκευές δίσκων επιτυγχάνεται καλύτερος παραλληλισμός. Για παράδειγμα, χρησιμοποιώντας μία συσκευή για έξοδο και τις υπόλοιπες για την είσοδο δεδομένων μπορεί να γίνει καλύτερη επικάλυψη των χρόνων εντοπισμού και περιστροφής. Ακόμη και όταν αυτή η επικάλυψη γίνει κατά το θεωρητικά βέλτιστο τρόπο, τότε εύκολα προκύπτει ότι ο απαιτούμενος χρόνος για τη συγχώνευση ισούται με το χρόνο για μία σειριακή ανάγνωση ή αποθήκευση του αρχείου, δηλαδή 14 λεπτά. Προσθέτοντας και τον αρχικό χρόνο ταξινόμησης των τμημάτων προκύπτει συνολικός χρόνος 28 λεπτών. Αυτός ο χρόνος αποτελεί το θεωρητικό κάτω όριο και δεν μπορεί να βελτιωθεί περισσότερο.

6.6 Άλλες μέθοδοι εξωτερικής συγχώνευσης

Στο προηγούμενο κεφάλαιο παρουσιάστηκαν μέθοδοι ειδικά σχεδιασμένες για εξωτερική ταξινόμηση με ταινίες. Οι μέθοδοι αυτές μπορούν να εφαρμοστούν και σε δίσκους. Κατ' αναλογία προς την ανάλυση του προηγούμενου κεφαλαίου προκύπτει ότι ο αντίστοιχος συνολικός χρόνος για είσοδο και έξοδο των δεδομένων με τη μέθοδο της φυσικής συγχώνευσης είναι:

$$4 \lceil \log_P nsg \rceil \times \frac{n \times R}{\frac{Buf}{P+1}} \times \left(s + r + \frac{Buf}{(P+1) \times t'} \right)$$

για την ισοζυγισμένη συγχώνευση προκύπτει:

$$2 \lceil \log_P nsg \rceil \times \frac{n \times R}{\frac{Buf}{2P}} \times \left(s + r + \frac{Buf}{2P \times t'} \right)$$

ενώ για την πολυφασική συγχώνευση προκύπτει:

$$2 PN(P) \times \frac{n \times R}{\frac{Buf}{P+1}} \times \left(s + r + \frac{Buf}{(P+1) \times t'} \right)$$

Συγκρίνοντας τα ζεύγη των τύπων για κάθε μέθοδο και για διάφορες τιμές των παραμέτρων εύκολα φαίνεται η βελτίωση της χρονικής απόκρισης από τη χρήση των δίσκων σε σχέση με τη χρήση ταινιών. Μάλιστα στην πράξη η πολυφασική συγχώνευση έχει χρησιμοποιηθεί και με συσκευές μαγνητικών δίσκων. Ωστόσο, στη συνέχεια αποδείχθηκε η ακαταλληλότητα της μεθόδου για το περιβάλλον των μαγνητικών δίσκων. Για παράδειγμα, η ταξινόμηση του γνωστού αρχείου των 2,4 Gb με 120 τμήματα των 20 Mb απαιτεί 32 περίπου λεπτά όταν είναι διαθέσιμοι δύο δίσκοι. Σε αντίθεση, ο Knuth αναφέρει ότι όταν η πολυφασική συγχώνευση χρησιμοποιεί 10 συσκευές για να ταξινομήσει το ίδιο αρχείο, τότε ο απαιτούμενος χρόνος είναι τετραπλάσιος του χρόνου μίας σειριακής ανάγνωσης, δηλαδή 56 περίπου λεπτά. Δηλαδή, ακόμη και με 10 συσκευές η επίδοση της πολυφασικής συγχώνευσης υστερεί σε σχέση με την επίδοση της προηγούμενης μεθόδου.

Πολλές φορές παρατηρείται το φαινόμενο ότι πολλά εμπορικά πακέτα εξωτερικής ταξινόμησης είναι πολύ αργά. Ένας λόγος είναι ότι χρησιμοποιούν για την αρχική φάση της δημιουργίας των ταξινομημένων τμημάτων κάποιον αλγόριθμο τάξης $O(n^2)$. Ακόμη, η αρχιτεκτονική του συστήματος μπορεί να μην είναι η βέλτιστη για το συγκεκριμένο πακέτο. Επίσης, σε

ένα υπολογιστικό περιβάλλον πολλών χρηστών δημιουργούνται προβλήματα επάρκειας κύριας μνήμης, χώρου στο δίσκο, διάρκειας του χρόνου εξυπηρέτησης κάθε χρήστη κλπ. Ωστόσο, είναι βέβαιο ότι υπάρχουν και αξιόλογα διαθέσιμα εμπορικά πακέτα. Για παράδειγμα, αν ένα πακέτο χρειασθεί διπλάσιο ή τριπλάσιο χρόνο από αυτόν που έχει υπολογισθεί αναλυτικά, τότε θεωρείται ότι πρόκειται για μία καλή λύση.

6.7 Ασκήσεις

<1> Να εφαρμοσθεί ο αλγόριθμος επιλογής αντικατάστασης σε κύρια μνήμη χωρητικότητας 7 εγγραφών. Οι διαδοχικά εισαγόμενες εγγραφές έχουν κλειδιά: 12, 18, 25, 8, 20, 10, 7, 30, 32, 24, 9, 4, 28, 6, 29.

<2> Με τα δεδομένα της προηγούμενης άσκησης να εφαρμοσθεί ο αλγόριθμος ταξινόμησης σωρού.

<3> Σε μία συσκευή δίσκου αποθηκεύεται ένα αρχείο ασθενών νοσοκομείου με 100.000 εγγραφές των 400 bytes. Ποιά διαδικασία είναι ταχύτερη για την αναζήτηση μίας λίστας ονομάτων 500 ασθενών.

- να ταξινομηθεί το αρχείο και η λίστα, και να σαρωθεί το αρχείο μία φορά; ή
- να γίνουν 500 ανεξάρτητες αναζητήσεις σε αρχείο σωρού;

Αν η λίστα αποτελούνταν από 50 ονόματα τι θα συνέφερε;

<4> Δίνονται δύο αρχεία σωρού με 10.000.000 και 2.000.000 εγγραφές των 100 bytes αντίστοιχα. Τα δύο αρχεία έχουν 1.000.000 κοινές εγγραφές. Να προταθεί μία τουλάχιστον τεχνική για τη δημιουργία ενός μόνο αρχείου χωρίς διπλοεγγραφές και να κοστολογηθεί. Υποτίθεται ότι ο δίσκος έχει τα χαρακτηριστικά του IBM 3380 αλλά πολύ μεγαλύτερη χωρητικότητα.

<5> Δίνονται δύο ταξινομημένα αρχεία μεγέθους 500 και 200 Mb αντίστοιχα. Το περιεχόμενο των 100 Mb είναι κοινό. Πόσος χρόνος χρειάζεται για τη δημιουργία ενός μόνο αρχείου χωρίς διπλοεγγραφές. Υποτίθεται ότι ο δίσκος έχει τα χαρακτηριστικά του IBM 3380 αλλά πολύ μεγαλύτερη χωρητικότητα.

<6> Δίνεται αρχείο των 2000 Mb και σύστημα δύο συσκευών δίσκων και κύριας μνήμης των 20 Mb. Είναι προτιμότερη η συγχώνευση 100 δρόμων

ή η συγχώνευση 10 δρόμων; Είναι προτιμότερη η συγχώνευση 81 δρόμων ή η συγχώνευση 9 δρόμων; Για δεδομένες παραμέτρους του συστήματος να βρεθεί η βέλτιστη τιμή του P ;

<7> Ένα αρχείο αποτελείται από 2028 ταξινομημένα τμήματα των 4 Mb. Το μέγεθος της εγγραφής είναι 200 bytes, ενώ το μέγεθος της απομονωτικής μνήμης είναι 16 Mb. Να υπολογισθεί ο απαιτούμενος χρόνος για είσοδο και έξοδο των δεδομένων κατά την εκτέλεση μίας ισοζυγισμένης συγχώνευσης P δρόμων (όπου $P = 4, 8, 12, 16, 20, 24$). Δίνεται επίσης ότι: ο αριθμός των κυλίνδρων είναι 555, αριθμός ατράκτων/κύλινδρο 39, πυκνότητα εγγραφής 19,2 Mb/άτρακτο, μέσος χρόνος αναζήτησης 25 ms, μέσος λανθάνων χρόνος 9,3 ms, μέγεθος κενού αμελητέο.

<8> Να θεωρηθούν τα δεδομένα της Άσκησης 7 και να υπολογισθεί ο απαιτούμενος χρόνος για πολυφασική συγχώνευση.

<9> Να μελετηθούν οι τύποι του Κεφαλαίου 6.6 με σκοπό την εύρεση του βέλτιστου αριθμού δρόμων P για κάθε περίπτωση.

Κεφάλαιο 7

ΑΡΧΕΙΑ ΣΕΙΡΙΑΚΑ ΜΕ ΔΕΙΚΤΗ

7.1 Εισαγωγή

7.2 Μέθοδος των καταλόγων κυλίνδρων και επιφανειών

7.3 Μέθοδος των σελίδων καταλόγου και δεδομένων

7.4 Ασκήσεις

Κεφάλαιο 7

ΣΕΙΡΙΑΚΑ ΑΡΧΕΙΑ ΜΕ ΔΕΙΚΤΗ

7.1 Εισαγωγή

Τα σειριακά αρχεία (ταξινομημένα ή αταξινομήτα) δεν αποτελούν αποτελεσματικές δομές για την αναζήτηση μίας συγκεκριμένης εγγραφής. Η οργάνωση των **σειριακών αρχείων με δείκτη** (index sequential files) μοιάζει με τα ταξινομημένα σειριακά αρχεία κατά το ότι τα δεδομένα αποθηκεύονται με τον ίδιο τρόπο στο δίσκο, δηλαδή σειριακά και ταξινομημένα ως προς το πρωτεύον κλειδί. Συνεπώς, αν από τη δομή αυτή ζητηθεί η ανάκτηση όλων των εγγραφών, τότε δεν αλλάζει τίποτε ως προς την αντίστοιχη επεξεργασία των σειριακών αρχείων. Ωστόσο, η νέα αυτή δομή συνοδεύεται και από έναν **κατάλογο ή δείκτη** (index), που αποτελεί ένα μικρότερο αρχείο με σκοπό την ταχύτερη υποστήριξη της αναζήτησης μίας συγκεκριμένης εγγραφής.

Στο σημείο αυτό μία παρένθεση κρίνεται απαραίτητη για την αποφυγή παρεξήγησης. Στην αγγλική ορολογία υπάρχει ο όρος **pointer**, που εύλογα μεταφράζεται ως δείκτης, αλλά υπάρχει και ο όρος **index** που μεταφράζεται ως κατάλογος ή ως δείκτης κατά περίπτωση. Στο εξής, ο όρος **index** θα αποδίδεται ως κατάλογος, αλλά τα αντίστοιχα αρχεία θα ονομάζονται σειριακά με δείκτη, γιατί αυτός ο όρος έχει επικρατήσει στην ελληνική βιβλιογραφία (καθώς επίσης και ο όρος **σειριακά δεικτοδοτημένα αρχεία**).

Εδώ και πολλά χρόνια κυκλοφορούν εμπορικά πακέτα που υλοποιούν αυτήν την οργάνωση. Η ανάπτυξη νέων κομψότερων μεθόδων έχει μειώσει τη ζήτηση των πακέτων αυτών, που ωστόσο ακόμη και σήμερα χρησιμοποιούνται αρκετά πλατιά (λόγω αδράνειας). Τα κυριότερα πακέτα είναι τα αρχεία **ISAM** (Index Sequential Access Method), τα αρχεία **VSAM** (Virtual

Storage Access Method) και τα αρχεία SIS (Scope Indexed Sequential). Τα δύο πρώτα πακέτα παρουσιάστηκαν με το σύστημα 370 της IBM, ενώ το τρίτο χρησιμοποιείται σε μεγάλα συστήματα υπολογιστών της CDC. Τα αρχεία ISAM χρησιμοποιούν τη μέθοδο των καταλόγων κυλίνδρων και επιφανειών, ενώ τα αρχεία VSAM και SIS χρησιμοποιούν τη μέθοδο των σελίδων καταλόγου και δεδομένων.

7.2 Μέθοδος των καταλόγων κυλίνδρων και επιφανειών

Η μέθοδος των καταλόγων κυλίνδρων και επιφανειών (cylinder and surface indexing) χρησιμοποιείται κυρίως σε συστήματα της IBM και είναι ευρύτερα γνωστή ως οργάνωση ISAM. Η οργάνωση ISAM στο παρελθόν αποτέλεσε πρότυπο. Για το λόγο αυτό, σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων, όπως για παράδειγμα η CA-INGRES, παρ' ότι χρησιμοποιούν άλλες πιο αποτελεσματικές μεθόδους (όπως τα B⁺-δένδρα, που θα αναπτυχθούν στη συνέχεια), αποκαλούν τη δεικτοδοτημένη τους οργάνωση, μέθοδο ISAM. Η οργάνωση αυτή στηρίζεται στα φυσικά χαρακτηριστικά του δίσκου, δηλαδή είναι άμεσα εξαρτώμενη από το υλικό, σε αντίθεση με τις οργανώσεις VSAM και SIS. Στη συνέχεια δίνεται μία σύντομη περιγραφή της δομής αυτής και θα ακολουθήσει ένα παράδειγμα για την καλύτερη κατανόησή της. Τέλος, θα δοθούν αναλυτικοί τύποι εκτίμησης του κόστους των διαφόρων διεργασιών.

Κατά τη δημιουργία ενός αρχείου ISAM, οι εγγραφές ταξινομούνται κατά αύξουσα τάξη μεγέθους του κλειδιού τους και αποθηκεύονται σειριακά στις διαδοχικές ατράχτους ενός κυλίνδρου, αφήνοντας συνήθως κενές την πρώτη και μερικές από τις τελευταίες ατράχτους. Επίσης δεσμεύεται κενός χώρος σε κάθε άτρακτο με σκοπό την μελλοντική αποθήκευση νέων εγγραφών. Ακόμη, ο πρώτος και μερικοί τελευταίοι κύλινδροι αφήνονται κενοί. Όταν ο δεύτερος κύλινδρος γεμίσει, τότε η φόρτωση συνεχίζεται στους επόμενους κυλίνδρους. Ο μέχρι στιγμής κατειλημμένος χώρος ονομάζεται **κύρια περιοχή δεδομένων** (prime data area).

Όταν όλες οι εγγραφές έχουν φορτωθεί στους κυλίνδρους, τότε το σύστημα δημιουργεί τον κατάλογο, που είναι μία στατική δενδρική δομή. Η δομή αυτή συνήθως αποτελείται από τρία επίπεδα: τον κύριο κατάλογο

(main index), τον κατάλογο κυλίνδρων (cylinder index) και τον κατάλογο ατράχτων (track index).

Ο κατάλογος κυλίνδρων αποτελείται από έναν αριθμό ζευγών στοιχείων ίσο με τον αριθμό των κυλίνδρων που καταλαμβάνει το αρχείο. Κάθε ζεύγος αποτελείται από την τιμή του μεγαλύτερου κλειδιού για κάθε κύλινδρο και τη διεύθυνση του αντίστοιχου κυλίνδρου. Η θέση όπου θα αποθηκευθεί ο κατάλογος κυλίνδρων είναι θέμα βελτιστοποίησης σε συνάρτηση με τα χαρακτηριστικά της χρήσης του αρχείου. Αν δεν είναι γνωστή η πιθανότητα προσπελάσεων στους κυλίνδρους, τότε ο κατάλογος αυτός αποθηκεύεται στο μεσαίο κύλινδρο του αρχείου, ειδικά αποθηκεύεται μεταξύ των κυλίνδρων με τη μεγαλύτερη πιθανότητα προσπέλασης.

Αν ο κατάλογος κυλίνδρων είναι ογκώδης (και συνεπώς δαπανηρός κατά την επεξεργασία του), τότε δημιουργείται ο κύριος κατάλογος που αποτελείται από τόσα ζεύγη στοιχείων, όσες είναι οι άτρακτοι που καταλαμβάνει ο κατάλογος κυλίνδρων. Το λογισμικό της IBM δίνει τη δυνατότητα δημιουργίας μέχρι πέντε επιπέδων καταλόγου επάνω από το επίπεδο του καταλόγου κυλίνδρων για υπερβολικά μεγάλα αρχεία. Πολλές φορές αυτά τα επίπεδα καταλόγου αποθηκεύονται σε δίσκους ημιαγωγών (δες Κεφάλαιο 2.5) για την επιτάχυνση της διαδικασίας αναζήτησης. Κάθε ζεύγος του κύριου καταλόγου δίνει την τιμή του μεγαλύτερου κλειδιού για κάθε άτρακτο του καταλόγου κυλίνδρων και την αντίστοιχη διεύθυνση. Όσο το αρχείο είναι ανοικτό για επεξεργασία, ο κύριος κατάλογος είναι αποθηκευμένος στην κύρια μνήμη και συνεπώς προσπελάζεται με πρακτικά μηδαμινό κόστος.

Στο χαμηλότερο επίπεδο των καταλόγων βρίσκεται ο κατάλογος ατράχτων. Στην πρώτη άτρακτο κάθε κυλίνδρου αποθηκεύεται ο αντίστοιχος κατάλογος ατράχτων και αποτελείται από δύο ζεύγη στοιχείων. Το πρώτο ζεύγος ονομάζεται κανονική είσοδος (normal entry) και αποτελείται από την τιμή του μεγαλύτερου κλειδιού για κάθε άτρακτο και την αντίστοιχη διεύθυνση της άτρακτου. Το δεύτερο ζεύγος ονομάζεται είσοδος υπερχείλισης (overflow entry) και αποτελείται από την τιμή του μεγαλύτερου κλειδιού, που προέρχεται από την άτρακτο αυτή αλλά είναι αποθηκευμένο στην περιοχή υπερχείλισης (θα εξετασθεί στη συνέχεια) και την αντίστοιχη διεύθυνση. Αν N_{key_i} και O_{key_i} είναι αντίστοιχα η τιμή του κλειδιού της κανονικής εισόδου και της εισόδου υπερχείλισης της άτρακτου i , τότε ισχύει:

$$N_{key_1} \leq O_{key_1} < N_{key_2} \leq O_{key_2} < \dots < N_{key_n} \leq O_{key_n}$$

Αν $N_{key_i} = O_{key_i}$, τότε έπεται ότι δεν υπάρχουν εγγραφές υπερχείλισης.

Λόγω των εισαγωγών η κύρια περιοχή αργά ή γρήγορα εξαντλείται. Τότε αποδίδεται στο αρχείο η περιοχή υπερχείλισης (overflow area), που διακρίνεται στην τοπική ή κυλινδρική ή ενσωματωμένη (local, cylinder, embedded) και στην ολική ή ανεξάρτητη (global, independent) περιοχή υπερχείλισης. Η τοπική περιοχή υπερχείλισης αποτελείται από τις τελευταίες ατράκτους κάθε κυλίνδρου, ενώ η ολική περιοχή υπερχείλισης αποτελείται από κάποιους κυλίνδρους στο τέλος του αρχείου. Έτσι για την προσπέλαση στην τοπική περιοχή υπερχείλισης δεν υπάρχει επιπλέον επιβάρυνση με κόστος εντοπισμού κυλίνδρου.

Η αναζήτηση γίνεται με διαδοχικές προσπελάσεις στα επίπεδα του καταλόγου, στην κύρια περιοχή, και (ίσως) στην περιοχή υπερχείλισης. Μία νέα εγγραφή ανάλογα με την τιμή του κλειδιού της εισάγεται στην κύρια ή στην περιοχή υπερχείλισης εκτοπίζοντας πιθανώς άλλες εγγραφές. Στην περίπτωση διαγραφής, η εγγραφή δεν απομακρύνεται φυσικά από το αρχείο, αλλά ένα ιδιαίτερο πεδίο της, μία 'σημαία', 'ανεβαίνει' για να το δηλώσει λογικά. Έτσι ο χώρος δεν αποδίδεται και πάλι για χρήση αλλά μένει ανεκμεταλλεύτος προσωρινά. Είναι δυνατόν ο χώρος διαγραφείσας εγγραφής από την κύρια περιοχή να καταληφθεί και πάλι από μελλοντική εισαγωγή με ανάλογη διευθέτηση των εγγραφών της κύριας περιοχής. Αν κάποια εγγραφή διαγραφεί από την περιοχή υπερχείλισης, τότε ενημερώνεται σχετικά η συνδεδεμένη λίστα. Σε περίπτωση μελλοντικής εισαγωγής στην περιοχή υπερχείλισης, αυτή η λίστα κατευθύνει την αποθήκευση της νέας εγγραφής σε θέση όπου ήταν αποθηκευμένη προηγουμένως μία άλλη εγγραφή. Προφανώς οι διαγραφές δεν αντιστοιχούν μία προς μία προς τις εισαγωγές, άρα είναι βέβαιο ότι θα υπάρχει χώρος ανεκμεταλλεύτος. Με τη διαδικασία της **αναδιοργάνωσης** (reorganization) μπορεί να γίνει ταχτοποίηση του αρχείου και οι εγγραφές υπερχείλισης να μεταφερθούν στην κύρια περιοχή. Στη συνέχεια παρουσιάζεται ένα παράδειγμα για την καλύτερη κατανόηση των εννοιών και διαδικασιών που αναφέρθηκαν.

Στο Σχήμα 7.1 παρουσιάζεται δείγμα αρχείου που καταλαμβάνει δεκατέσσερις κυλίνδρους. Οι κύλινδροι αυτοί είναι αριθμημένοι από 100 ως 113. Ο πρώτος κύλινδρος (υπ' αριθμό 100) χρησιμεύει για την αποθήκευση του καταλόγου κυλίνδρων, ενώ ο τελευταίος (υπ' αριθμό 113) ως ολική περιοχή υπερχείλισης. Άρα, η κύρια περιοχή δεδομένων αποτελείται από δώδεκα κυλίνδρους. Οι κύλινδροι έχουν 6 ατράκτους χωρητικότητας τριών εγγραφών. Σε κάθε κύλινδρο της κύριας περιοχής, η πρώτη και η τελευταία άτρακτος χρησιμεύουν για την αποθήκευση του καταλόγου ατράκτων και της τοπικής περιοχής υπερχείλισης, αντίστοιχα.

Κύλινδρος C100	Κύλινδρος C101			...	Κύλινδρος C112			Κύλινδρος C113
Κατάλογος κυλίνδρων	Κατάλογος ατράκτων			...	Κατάλογος ατράκτων			Ολική περιοχή υπερχείλισης
	4	5	10		291	302	316	
	12	19	52		346	351	356	
	56	71	90		382	401	411	
	100	113	142		425	431	437	
	Τοπική περιοχή υπερχείλισης				Τοπική περιοχή υπερχείλισης			

Σχήμα 7.1: Σειριακό αρχείο με δείκτη σε 14 κυλίνδρους.

Μεγαλύτερο κλειδί	Διεύθυνση κυλίνδρου	Κανονική είσοδος	Είσοδος υπερχείλισης
142	C101	10	S1
⋮	⋮	52	S2
437	C112	90	S3
		142	S4

(α)
(β)

Σχήμα 7.2: Κατάλογος κυλίνδρων και κατάλογος ατράκτων κυλίνδρου 101.

Στο Σχήμα 7.2α παρουσιάζεται ο κατάλογος κυλίνδρων με τις τιμές των μεγαλύτερων κλειδιών, που είναι αποθηκευμένα στους δώδεκα κυλίνδρους. Στο Σχήμα 7.2β το ενδιαφέρον εστιάζεται στον κατάλογο ατράκτων του κυλίνδρου 101. Όταν γίνεται αναζήτηση μίας εγγραφής με βάση την τιμή του πρωτεύοντος κλειδιού, τότε τα κλειδιά αυτά σαρώνονται σειριακά μέχρι να βρεθεί ένα μεγαλύτερο κλειδί από εκείνο για το οποίο γίνεται η αναζήτηση.

Έστω ότι στο αρχείο εισάγεται μία εγγραφή με κλειδί 8. Είναι γνωστό βέβαια ότι σε όλους τους τύπους αρχείων κάθε εισαγωγή ενεργοποιεί τη ρουτίνα αναζήτησης της συγκεκριμένης εγγραφής για δύο λόγους. Πρώτον, για να αποφευχθεί η διπλή αποθήκευση, και δεύτερον, για να εντοπισθεί η θέση όπου τελικά θα γίνει η εισαγωγή. Σαρώνοντας τον κατάλογο του Σχήματος 7.2α διαπιστώνεται ότι η εγγραφή αυτή ανήκει στον κύλινδρο 101.

Με δεύτερη σάρωση του καταλόγου ατράκτων, όπως φαίνεται στο Σχήμα 7.2β, διαπιστώνεται ότι η εγγραφή 8 πρέπει να αποθηκευθεί στην άτρακτο 1. Ωστόσο, η άτρακτος αυτή είναι πλήρης αφού ήδη περιέχει τρεις εγγραφές. Για να διατηρηθεί η αύξουσα σειρά των κλειδιών παρεμβάλλεται η εγγραφή 8 στη θέση μεταξύ των εγγραφών 5 και 10. Έτσι η εγγραφή 10 εκτοπίζεται προς την τοπική περιοχή υπερχειλίσσης, δηλαδή στην άτρακτο 5, θέση 0. Ενημερώνεται, λοιπόν, ανάλογα η κανονική είσοδος και η είσοδος υπερχειλίσσης της ατράκτου 1 του κυλίνδρου 101. Το αποτέλεσμα της εισαγωγής παρουσιάζεται στο Σχήμα 7.3α.

Έστω ότι στη συνέχεια πρέπει να εισαχθεί η εγγραφή 14. Με παρόμοιο τρόπο διαπιστώνεται ότι η θέση της είναι στον κύλινδρο 101 και στην άτρακτο 2. Η εγγραφή 14 εισάγεται μεταξύ των εγγραφών 12 και 19 εκτοπίζοντας την εγγραφή 52 στην περιοχή υπερχειλίσσης. Πιο συγκεκριμένα, η εγγραφή 52 τοποθετείται στη θέση 1 της ατράκτου 5, οπότε ενημερώνεται ανάλογα η αντίστοιχη κανονική είσοδος του καταλόγου, όπως φαίνεται στο Σχήμα 7.3β.

Στη συνέχεια εισάγεται η εγγραφή 6, που αποθηκεύεται στην άτρακτο 1 του κυλίνδρου 101. Έτσι η εγγραφή 8 εκτοπίζεται στη θέση 2 της ατράκτου 5, και ενημερώνεται σχετικά ο κατάλογος. Η φυσική σειρά των εγγραφών υπερχειλίσσης δεν έχει καμία σημασία. Οι εγγραφές αυτές με τη βοήθεια των καταλόγων προσπελάζονται κατ' ευθείαν από τις ατράκτους της κύριας περιοχής. Μάλιστα, από κάθε άτρακτο ξεκινά μία συνδεδεμένη λίστα που συντηρεί τις εγγραφές της ατράκτου διατεταγμένες κατά αύξουσα τάξη. Το αποτέλεσμα της εισαγωγής της εγγραφής 6 παρουσιάζεται στο Σχήμα 7.3γ.

Από αυτή τη διαδικασία εισαγωγών που περιγράφηκε εξάγονται τα εξής συμπεράσματα. Η στήλη των μεγαλύτερων κλειδιών της κανονικής εισόδου είναι πιθανόν να αλλάξει με την είσοδο των νέων εγγραφών, οι στήλες της διεύθυνσης της κανονικής εισόδου και του μεγαλύτερου κλειδιού της εισόδου υπερχειλίσσης δεν αλλάζουν ποτέ, ενώ η στήλη της διεύθυνσης της εισόδου υπερχειλίσσης είναι βέβαιο ότι θα αλλάξει αν κάποια εγγραφή μεταφερθεί στην περιοχή υπερχειλίσσης.

Έστω, λοιπόν, ότι πρέπει να διαγραφεί η εγγραφή 14. Το αποτέλεσμα της διαγραφής φαίνεται στο Σχήμα 7.3δ, όπου ο χώρος της εγγραφής έχει γραμμοσκιασθεί. Στη συνέχεια ακολουθεί μαθηματική ανάλυση μερικών ποσοτικών στοιχείων της οργάνωσης αυτής, που αφορούν στο χρονικό κόστος των διαφόρων διεργασιών επί της δομής.

(α) εισαγωγή 8

Κανονική είσοδος		Είσοδος υπερχείλισης	
8	S1	10	S5R0
52	S2	52	S2
90	S3	90	S3
142	S4	142	S4

S1	4	5	8
S2	12	19	52
S3	56	71	90
S4	100	113	142
S5	10		

(β) εισαγωγή 14

Κανονική είσοδος		Είσοδος υπερχείλισης	
8	S1	10	S5R0
19	S2	52	S5R1
90	S3	90	S3
142	S4	142	S4

S1	4	5	8
S2	12	14	19
S3	56	71	90
S4	100	113	142
S5	10	52	

(γ) εισαγωγή 6

Κανονική είσοδος		Είσοδος υπερχείλισης	
6	S1	10	S5R2
19	S2	52	S5R1
90	S3	90	S3
142	S4	142	S4

S1	4	5	6
S2	12	14	19
S3	56	71	90
S4	100	113	142
S5	10	52	8

(δ) διαγραφή 14

Κανονική είσοδος		Είσοδος υπερχείλισης	
6	S1	10	S5R2
19	S2	52	S5R1
90	S3	90	S3
142	S4	142	S4

S1	4	5	6
S2	12	14	19
S3	56	71	90
S4	100	113	142
S5	10	52	8

Σχήμα 7.3: Κατάλογος ατράχτων και κύρια περιοχή κυλίνδρου 101 μετά την εισαγωγή των εγγραφών 8, 14, 6 και τη διαγραφή της εγγραφής 108.

7.2.1 Προσπέλαση εγγραφής

Για την προσπέλαση μίας εγγραφής σε ένα σειριακό αρχείο με δείκτη πρέπει, προφανώς, να προσπελασθεί πρώτα ο κατάλογος. Υποτίθεται ότι ο κύριος κατάλογος είναι αποθηκευμένος στην κύρια μνήμη, οπότε ο αντίστοιχος χρόνος αναζήτησης (c) είναι αμελητέος, ενώ τα άλλα επίπεδα του καταλόγου είναι αποθηκευμένα στο δίσκο. Αν δεν υπάρχουν εγγραφές υπερχειλίσης, τότε το χρονικό κόστος είναι:

$$T_{\text{προσ,κυρ}} = c + 2 \times (s + r + btt) + (r + btt)$$

όπου το γινόμενο $2 \times (r + btt)$ δίνει το χρόνο επεξεργασίας του καταλόγου κυλίνδρων και του καταλόγου ατράκτων, ενώ η ποσότητα $r + btt$ παριστά το χρόνο περιστροφής και μεταφοράς της κατάλληλης σελίδας από την κύρια περιοχή του αρχείου.

Ωστόσο, είναι πολύ πιθανό να υπάρχουν εγγραφές υπερχειλίσης στο αρχείο. Αν το αρχείο έχει n εγγραφές στην κύρια περιοχή και o στην περιοχή υπερχειλίσης, τότε η πιθανότητα μία εγγραφή να βρίσκεται στην περιοχή υπερχειλίσης είναι:

$$P_{\text{υπερ}} = \frac{o}{n + o}$$

Οι εγγραφές υπερχειλίσης συνδέονται με μία αλυσίδα από την κύρια σελίδα από όπου προήλθαν. Προσεγγιστικά το μήκος της αλυσίδας ισούται με:

$$L_{\text{αλυσ}} = P_{\text{υπερ}} \times Bfr$$

Αν οι εγγραφές υπερχειλίσης είναι αποθηκευμένες σε άλλον κύλινδρο, τότε υπεισέρχεται νέος χρόνος αναζήτησης s' , τουλάχιστο για την προσπέλαση της πρώτης εγγραφής της αλυσίδας. Ο χρόνος αυτός είναι:

$$T_{\text{προσ,υπερ}} = P_{\text{υπερ}} \times (s' + r + btt)$$

Είναι πολύ πιθανό ότι οι υπόλοιπες εγγραφές της αλυσίδας ανήκουν σε διαφορετικές σελίδες. Μία εκτίμηση του αριθμού των εγγραφών της αλυσίδας υπερχειλίσης (και συνεπώς των σελίδων) που θα προσπελασθούν είναι $\frac{(L_{\text{αλυσ}} - 1) + 1}{2}$. Πρέπει, βέβαια, να ληφθεί υπ' όψη ότι προσπέλαση στην αλυσίδα θα γίνει μόνο με την προϋπόθεση ότι αναζητείται εγγραφή υπερχειλίσης. Συνεπώς ισχύει:

$$T_{\text{προσ,αλυσ}} = P_{\text{υπερ}} \times \frac{L_{\text{αλυσ}}}{2} \times (r + btt)$$

Τελικά, ο χρόνος προσπέλασης μίας εγγραφής είναι:

$$T_{\text{προσ}} = T_{\text{προσ,κυρ}} + T_{\text{προσ,υπερ}} + T_{\text{προσ,αλυσ}}$$

7.2.2 Προσπέλαση επόμενης εγγραφής

Για την προσπέλαση της επόμενης εγγραφής δεν χρειάζεται και πάλι επεξεργασία του καταλόγου. Ωστόσο, η προσπέλαση της επόμενης εγγραφής είναι μία αρκετά σύνθετη διαδικασία και υποδιαιρείται σε πολλές περιπτώσεις:

- η τρέχουσα εγγραφή είναι αποθηκευμένη σε μία κύρια σελίδα και η επόμενη εγγραφή βρίσκεται στην ίδια σελίδα,
- η τρέχουσα εγγραφή είναι η τελευταία της κύριας σελίδας, δεν υπήρξαν εισαγωγές και η επόμενη εγγραφή είναι αποθηκευμένη στην επόμενη σελίδα του ίδιου κυλίνδρου,
- η τρέχουσα εγγραφή είναι η τελευταία της κύριας σελίδας, δεν υπήρξαν εισαγωγές, αλλά η επόμενη εγγραφή είναι αποθηκευμένη στην πρώτη σελίδα του επόμενου κυλίνδρου,
- η τρέχουσα εγγραφή είναι η τελευταία της κύριας σελίδας, έχουν υπάρξει εισαγωγές και η επόμενη εγγραφή είναι αποθηκευμένη στην περιοχή υπερχειλίσης,
- η τρέχουσα εγγραφή είναι μία περίπτωση εισαγωγής και η επόμενη εγγραφή είναι αποθηκευμένη σε άλλη σελίδα στον ίδιο κύλινδρο, και τέλος,
- η τρέχουσα εγγραφή είναι μία περίπτωση εισαγωγής και η επόμενη εγγραφή είναι αποθηκευμένη σε άλλη σελίδα.

Κάθε μία από τις περιπτώσεις αυτές πρέπει να αντιμετωπισθεί ιδιαίτερα. Έτσι εισάγονται νέες μεταβλητές:

$$\begin{aligned} P_{\text{κυρ}} (\text{τρέχουσα εγγραφή σε κύρια σελίδα}) &= 1 - P_{\text{υπερ}} \\ P_{\text{idia,σελ}} (\text{επόμενη εγγραφή στην ίδια σελίδα}) &= 1 - 1/Bfr \\ P_{\text{oχι,εισ}} (\text{μη εισαγωγές στην ίδια σελίδα}) &= 1 - P_{\text{υπερ}} \\ P_{\text{idio,κυλ}} (\text{επόμενη σελίδα στον ίδιο κύλινδρο}) &= 1 - 1/C \\ P_{\text{oχι,αλυσ}} (\text{τρέχουσα εγγραφή υπερχειλίσης δεν είναι} \\ &\quad \text{τελευταία της αλυσίδας}) &= 1 - 1/L_{\text{αλυσ}} \end{aligned}$$

Τελικά, ο χρόνος προσπέλασης της επόμενης εγγραφής είναι το άθροισμα που ακολουθεί, όπου κάθε όρος αντιστοιχεί στις έξι περιπτώσεις που αναφέρθηκαν:

$$T_{\text{επομ}} = \begin{cases} P_{\text{κυρ}} P_{\text{ιδια,σελ}} c + \\ P_{\text{κυρ}} (1 - P_{\text{ιδια,σελ}}) P_{\text{οχι,εισ}} P_{\text{ιδιο,κυλ}} (r + btt) + \\ P_{\text{κυρ}} (1 - P_{\text{ιδια,σελ}}) P_{\text{οχι,εισ}} (1 - P_{\text{ιδιο,κυλ}}) (s + r + btt) + \\ (1 - P_{\text{κυρ}}) P_{\text{οχι,αλυσ}} (r + btt) + \\ (1 - P_{\text{κυρ}}) (1 - P_{\text{οχι,αλυσ}}) (s' + r + btt) \end{cases}$$

Όταν οι περιοχές υπερχειλίσσης βρίσκονται στους ίδιους κυλίνδρους με τα δεδομένα, τότε ο όρος s' μηδενίζεται. Επίσης αν αγνοηθεί ο χρόνος επεξεργασίας του κύριου καταλόγου και αν αγνοηθούν οι χρόνοι αναζήτησης των κυλίνδρων, τότε ο προηγούμενος τύπος απλοποιείται ως εξής:

$$T_{\text{επομ}} = (r + btt) \times \frac{n + o \times Bfr}{(n + o) \times Bfr}$$

7.2.3 Εισαγωγή εγγραφής

Η εισαγωγή μίας εγγραφής πρέπει να αντιμετωπισθεί ως ένα σύνολο δύσκολα μοντελοποιήσιμων περιπτώσεων. Για απλοποίηση, εδώ υποτίθεται ότι η νέα εγγραφή είτε αποθηκεύεται στην περιοχή υπερχειλίσσης, είτε ωθεί μία άλλη εγγραφή στην περιοχή υπερχειλίσσης.

Η διαδικασία εισαγωγής αρχίζει με την ανάγνωση των προηγούμενων δεδομένων ή μίας σελίδας υπερχειλίσσης. Επομένως αρχικά υπάρχει ένα κόστος ίσο με $T_{\text{προσ}}$. Η πιθανότητα η νέα εγγραφή με τη λογικά προηγούμενή της να βρίσκονται στην ίδια σελίδα είναι πρακτικά πολύ μικρή για μεγάλα αρχεία. Ωστόσο, υποτίθεται ότι βρίσκονται στον ίδιο κύλινδρο, άρα χωρίς χρόνο εντοπισμού απαιτείται $r + btt$ για να προσπελασθεί η νέα σελίδα, ενώ $2r$ απαιτούνται για την επανα-αποθήκευση της σελίδας. Συνεπώς τελικά ισχύει:

$$T_{\text{εισ}} = T_{\text{προσ}} + 5r + btt$$

Ας σημειωθεί ότι η επίδραση του μήκους της αλυσίδας υπερχειλίσσης συμπεριλαμβάνεται στον όρο $T_{\text{προσ}}$.

7.2.4 Ανανέωση εγγραφής

Προκύπτει εύκολα ότι το χρονικό κόστος ανανέωσης μίας εγγραφής (χωρίς να αλλάζει το μήκος της ή η τιμή του κλειδιού της) ισούται με:

$$T_{\text{αναν}} = T_{\text{προσ}} + 2r$$

Στη γενική περίπτωση πρέπει να γίνει αποδεκτό ότι η εγγραφή μαρκάρεται ως διαγραμμένη και ότι εισάγεται μία νέα εγγραφή. (Εξήγηση της διαγραφής ακολουθεί.) Συνεπώς ισχύει:

$$T_{\text{αναν}} = T_{\text{προσ}} + 2r + T_{\text{εισ}} = 2T_{\text{προσ}} + 7r + btt$$

7.2.5 Διαγραφή εγγραφής

Το χρονικό κόστος για τη διαγραφή μίας εγγραφής ισούται με τον πρώτο όρο $T_{\text{αναν}}$, γιατί η περιοχή απλώς μαρκάρεται με μία σημαία χωρίς η εγγραφή πραγματικά να απομαχυνθεί. Άρα:

$$T_{\text{διαγ}} = T_{\text{προσ}} + 2r$$

7.2.6 Εξαντλητική ανάγνωση αρχείου

Η περίπτωση αυτή παρουσιάζεται όταν η αναζήτηση δεν γίνεται με βάση το πρωτεύον κλειδί ή με βάση το κλειδί ενός δευτερεύοντος καταλόγου. Η ανάγνωση του αρχείου μπορεί να γίνει είτε ακολουθώντας τη λογική σειρά των εγγραφών, είτε λαμβάνοντας τους κυλίνδρους με τη σειρά και αδιαφορώντας για το αν οι εγγραφές δίνονται ταξινομημένες στο χρήστη. Και στις δύο περιπτώσεις ο κατάλογος αγνοείται. Όλα σχεδόν τα συστήματα προσφέρουν την πρώτη δυνατότητα. Έτσι προκύπτει ότι:

$$\begin{aligned} T_{\text{εξαν}} &= T_{\text{προσ}} + (n + o - 1) \times T_{\text{επομ}} \approx (n + o) \times T_{\text{επομ}} \\ &= \frac{(n + o \times Bfr) \times (r + btt)}{Bfr} \end{aligned}$$

7.2.7 Αναδιοργάνωση αρχείου

Η αναδιοργάνωση είναι μία διαδικασία που συνίσταται στη σειριακή ανάγνωση του κύριου αρχείου αγνοώντας τον κατάλογο και τις διαγραμμένες εγγραφές. Στη συνέχεια, το αρχείο επανα-αποθηκεύεται σε άλλη έκταση του δίσκου χωρίς περιοχές υπερχείλισης. Στη φάση της νέας φόρτωσης δημιουργείται ο νέος κατάλογος με τη βοήθεια απομονωτικών μνημών. Κατόπιν ελευθερώνεται η έκταση που καταλαμβάνονταν από το παλιό αρχείο. Προκύπτει ότι το κόστος της αναδιοργάνωσης είναι:

$$T_{\text{αναδ}} = (n + o * Bfr) \times \frac{r + btt}{Bfr} + \frac{(n + o - d) \times R}{t'} + \frac{SI}{t'}$$

όπου d είναι ο αριθμός εγγραφών που έχουν μαρκαρισθεί ως διαγραμμένες, και SI είναι ο χώρος του καταλόγου. Προφανώς, ο πρώτος όρος δίνει το χρόνο ανάγνωσης του αρχείου, ενώ ο δεύτερος και ο τρίτος όρος δίνουν τον απαραίτητο χρόνο αποθήκευσης της κύριας περιοχής και του καταλόγου του αρχείου, αντίστοιχα.

Στη συνέχεια προχωρούμε σε μία εκτίμηση του χώρου που καταλαμβάνει ο κατάλογος, SI . Για κάθε σελίδα της κύριας περιοχής υπάρχει μία είσοδος στον κατάλογο ατράχτων. Άρα ο αριθμός αυτών των εισόδων είναι:

$$i_1 = \frac{n}{bfr}$$

Ο αριθμός των εισόδων που περιέχονται σε μία σελίδα του καταλόγου ατράχτων λέγεται λόγος διακλάδωσης (fanout ratio) ή παράγοντας διακλάδωσης (branching factor) και ισούται με:

$$y = \left\lfloor \frac{B}{K + P} \right\rfloor$$

όπου K είναι το μέγεθος του κλειδιού, και P είναι το μέγεθος του δείκτη σε bytes. Ο αριθμός των εισόδων στον κατάλογο κυλίνδρων που θα δεικτοδοτούν σελίδες του καταλόγου ατράχτων είναι:

$$i_2 = \frac{i_1}{y}$$

Θεωρητικά τα επίπεδα των καταλόγων είναι τόσα ώστε στο ανώτερο επίπεδο όλες οι εισοδοί να χωρούν σε μία σελίδα. Συνεπώς, ο αριθμός των σελίδων στο i -οστό επίπεδο ισούται με:

$$b_{i_l} = \left\lfloor \frac{i_l}{y} \right\rfloor = i_{l+1}$$

Άρα χρησιμοποιώντας την τιμή 1 για την ποσότητα b_{i_x} της σελίδας της ρίζας ισχύει:

$$SI = (b_{i_1} + b_{i_2} + \dots + 1) \times B = (i_2 + i_3 + \dots + 1) \times B$$

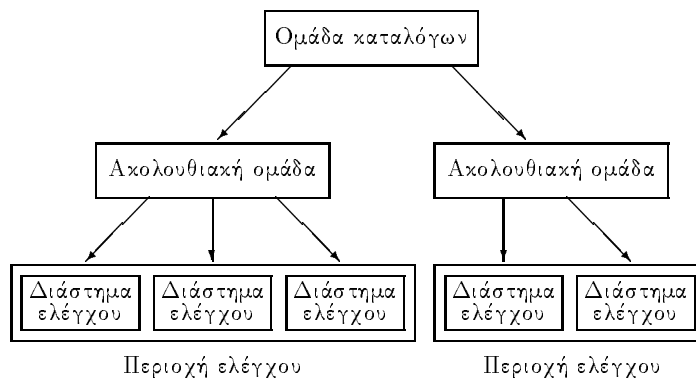
7.3 Μέθοδος των σελίδων καταλόγου και δεδομένων

Η μέθοδος των σελίδων καταλόγου και δεδομένων (index and data block) χρησιμοποιείται στα αρχεία VSAM της IBM και SIS της CDC. Οι δομές αυτές μοιάζουν με τα B⁺-δένδρα που θα αναπτυχθούν σε επόμενο κεφάλαιο. Στη συνέχεια θα παρουσιασθεί η μέθοδος VSAM που εμφανίστηκε το 1972 και εφαρμόστηκε σε μηχανές με υπερβατική μνήμη.

Η οργάνωση αυτή δεν στηρίζεται στα φυσικά χαρακτηριστικά του δίσκου (σε αντίθεση με την οργάνωση ISAM) και απαρτίζεται από τις εξής τέσσερις περιοχές:

- διαστήματα ελέγχου (control intervals), που περιέχουν τις εγγραφές με τα δεδομένα,
- περιοχές ελέγχου (control areas), που περιέχουν πολλά διαστήματα ελέγχου,
- ακολουθιακές ομάδες (sequence sets), που είναι κατάλογοι περιοχών ελέγχου, και
- ομάδα καταλόγων (index set), που αντιστοιχεί σε ένα διάστημα ελέγχου και είναι μία δενδρική δομή με το πολύ τρία επίπεδα. Οι σελίδες του κατωτέρου επιπέδου είναι κατάλογοι ακολουθιακών ομάδων.

Στο Σχήμα 7.4 παρουσιάζεται η δομή ενός VSAM αρχείου, ενώ το Σχήμα 7.5 παρουσιάζει τη δομή ενός διαστήματος ελέγχου. Κάθε διάστημα ελέγχου περιέχει στην αρχή μία ή περισσότερες εγγραφές, ελεύθερο χώρο για κατοπινές εισαγωγές εγγραφών, πεδία ορισμού εγγραφών (record definition fields, RDF) για κάθε εγγραφή του διαστήματος και ένα πεδίο ορισμού του διαστήματος ελέγχου (control interval definition fields, CIDF). Κάθε πεδίο ορισμού εγγραφών περιέχει τη σχετική διεύθυνση σε χαρακτήρες (relative byte address, RBA) της εγγραφής από την αρχή του αρχείου. Το



Σχήμα 7.4: Δομή αρχείου VSAM.

Εγγραφή 1	Εγγραφή 2	Εγγραφή 3	Εγγραφή 4
Εγγραφή 5	Εγγραφή 6	Ελεύθερος χώρος	
Ελεύθερος χώρος			RDF6
RDF4	RDF3	RDF2	RDF1
			CIDF

Σχήμα 7.5: Δομή διαστήματος ελέγχου.

πεδίο ορισμού του διαστήματος ελέγχου περιέχει δύο τιμές: το μέγεθος και τη σχετική θέση του ελεύθερου χώρου του διαστήματος.

Κατά τη δημιουργία του αρχείου VSAM οι εγγραφές ταξινομούνται και αποθηκεύονται στα διαστήματα ελέγχου. Η πρώτη εγγραφή αποθηκεύεται στην αρχή, στην πρώτη θέση του πρώτου διαστήματος και το αντίστοιχο πεδίο ορισμού εγγραφών αποθηκεύεται στα αριστερά του πεδίου ορισμού διαστήματος ελέγχου. Στη συνέχεια η δεύτερη εγγραφή αποθηκεύεται δεξιά της πρώτης εγγραφής και το αντίστοιχο πεδίο ορισμού της εγγραφής στα αριστερά του πεδίου ορισμού της πρώτης εγγραφής. Έτσι, οι εγγραφές και τα αντίστοιχα πεδία ορισμού τους αποθηκεύονται στα διαστήματα ως δύο στοίβες, που μεγαλώνουν προς αντίθετες κατευθύνσεις. Σε περιπτώσεις εισαγωγών και διαγραφών, οι επεμβάσεις γίνονται στα δύο άκρα του διαστήματος, και συνεπώς έτσι διασφαλίζεται ότι ο ελεύθερος χώρος είναι συνεχής και δεν διασπάται.

Η περιοχή ελέγχου και τα διαστήματα ελέγχου είναι έννοιες παρόμοιες προς τις φυσικές έννοιες του κυλίνδρου και των ατράκτων. Το μεγαλύτερο κλειδί κάθε διαστήματος μίας περιοχής ελέγχου αποθηκεύεται στο κατώτερο επίπεδο της ομάδας καταλόγων. Τα διαστήματα ελέγχου μίας περιοχής δεν είναι αναγκαίο να είναι αποθηκευμένα φυσικά κατά αύξουσα τάξη των κλειδιών. Αυτό συμβαίνει επειδή η ακολουθιακή ομάδα έχει μία διατεταγμένη σειρά από τα μεγαλύτερα κλειδιά των διαστημάτων ελέγχου μαζί με τις αντίστοιχες διευθύνσεις στο δίσκο. Η ακολουθιακή ομάδα περιέχει επίσης και δείκτες για τη σειριακή ανάκτηση όλων των διαστημάτων με αύξουσα τάξη των κλειδιών.

Η ομάδα καταλόγων είναι μία δομή δένδρου που περιέχει σελίδες με ζεύγη κλειδιών-δεικτών. Κάθε ζεύγος αποτελείται από το μεγαλύτερο κλειδί της σελίδας του καταλόγου του κατώτερου επιπέδου και τον αντίστοιχο δείκτη. Η σειρά των ζευγών είναι κατά αύξουσα τάξη των κλειδιών, έτσι ώστε να διευκολύνεται η σειριακή αναζήτηση στο αρχείο. Ας σημειωθεί ότι εφαρμόζοντας τεχνικές συμπίεσης είναι δυνατόν τα κλειδιά στην ομάδα των καταλόγων να αποθηκευθούν σε μικρότερο χώρο και έτσι το μέγεθος του καταλόγου να ελαττωθεί.

Η υλοποίηση του VSAM αρχείου που διατηρεί όλες τις εγγραφές των διαστημάτων ελέγχου, καθώς και όλα τα ζεύγη όλων των επιπέδων της ομάδας των καταλόγων, ταξινομημένες κατά αύξουσα τάξη των κλειδιών πρέπει να φορτωθεί έτσι από την αρχή. Η δομή αυτή λέγεται αρχείο VSAM με ακολουθία κλειδιών (key-sequenced). Κατά την αρχική φόρτωση του αρχείου αφήνεται κατανεμημένος ελεύθερος χώρος (distributed free space, embedded space) για μελλοντικές εισαγωγές, είτε στο τέλος των διαστημάτων ελέγχου, είτε διατηρώντας κάποια διαστήματα απολύτως ελεύθερα στο τέλος κάθε περιοχής ελέγχου. Υπάρχει, λοιπόν, μία στενή αναλογία με την περιοχή υπερχειλίσσης των αρχείων ISAM.

Αναζήτηση σε αρχείο VSAM μπορεί να γίνει κατά τρεις τρόπους:

- Η σειριακή αναζήτηση αρχίζει με διάσχιση των επιπέδων του καταλόγου, ώστε να εντοπισθεί η σχετική διεύθυνση σε bytes του πρώτου διαστήματος ελέγχου του αρχείου. Στη συνέχεια ακολουθώντας τη συνδεδεμένη λίστα της ακολουθιακής ομάδας διατρέχεται ολόκληρο το αρχείο. Η σειριακή αναζήτηση δεν είναι απαραίτητο να αρχίσει από το πρώτο διάστημα ελέγχου, αλλά μπορεί να αρχίσει από κάποια συγκεκριμένη εγγραφή και να συνεχίσει ως το τέλος του αρχείου. Αυτού

του είδους η αναζήτηση γίνεται σε περίπτωση ερώτησης διαστήματος με προσδιορισμένη την κάτω τιμή.

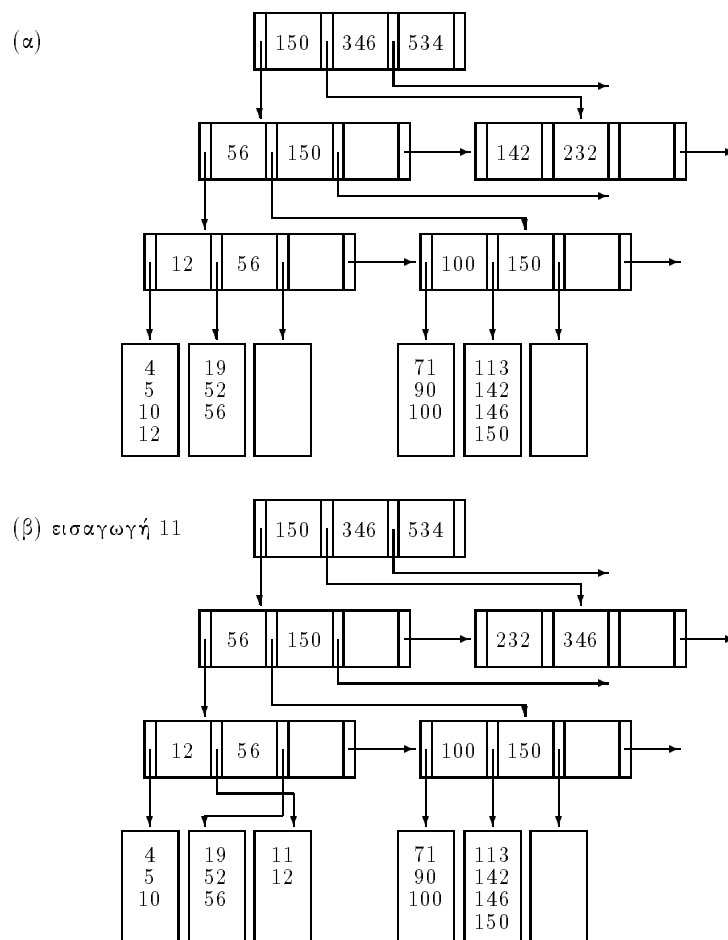
- Σειριακή αναζήτηση με άλμα εκτελείται όταν η ερώτηση διαστήματος έχει ορισμένες και τις δύο ακραίες τιμές.
- Τυχαία αναζήτηση γίνεται διασχίζοντας τη δενδρική δομή του καταλόγου μέχρι του σημείου να εντοπισθεί η σχετική διεύθυνση σε bytes της σελίδας που την περιέχει. Η διαδικασία αυτή είναι παρόμοια με τη διάσχιση ενός B⁺-δένδρου, όπως θα φανεί στο επόμενο κεφάλαιο.

Γενικά, πάντως, ο μηχανισμός προσπέλασης του αρχείου VSAM είναι πολύ αποτελεσματικός.

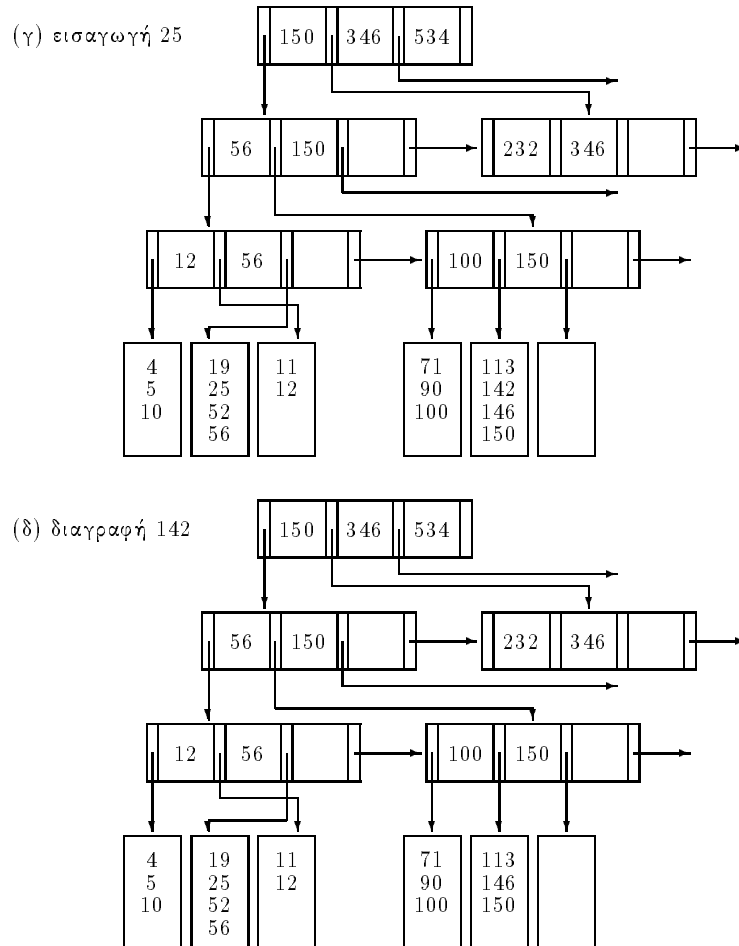
Κάθε νέα εγγραφή καταλαμβάνει μία θέση ανάλογα με την τιμή του κλειδιού της εξωθώντας τις υπόλοιπες. Το μέγιστο μέγεθος ενός διαστήματος ελέγχου είναι 32.768 bytes. Έτσι όταν λόγω μίας εισαγωγής γεμίσει ένα διάστημα ελέγχου, τότε συμβαίνει **διάσπαση** (splitting) του διαστήματος, οπότε οι μισές εγγραφές του μεταφέρονται σε ένα κενό διάστημα ελέγχου, που έχει δεσμευθεί στο τέλος της περιοχής ελέγχου. Με τον τρόπο αυτό οι εγγραφές δεν παραμένουν αποθηκευμένες σε αύξουσα τάξη. Εν τούτοις μπορεί να γίνει σειριακή ανάκτηση, επειδή η ομάδα των καταλόγων ενημερώνεται σχετικά. Αν μετά από διαδοχικές εισαγωγές γεμίσουν όλα τα διαστήματα μίας περιοχής ελέγχου, τότε η συγκεκριμένη περιοχή διασπάται και δημιουργείται μία νέα περιοχή στο τέλος του αρχείου, όπου μεταφέρονται τα μισά διαστήματα ελέγχου. Και πάλι η ομάδα των καταλόγων ενημερώνεται σχετικά. Η αυτόματη αυτή επέκταση ενός αρχείου VSAM μπορεί να φθάσει μέχρι το μέγιστο μέγεθος των 4,29 GB. Αυτή η διαδικασία είναι δαπανηρή και γι' αυτό ο σχεδιαστής του αρχείου πρέπει να είναι πολύ προσεκτικός στις τιμές που θα δώσει στις παραμέτρους του αρχείου.

Σε περίπτωση διαγραφής, ο χώρος ελευθερώνεται (σε αντίθεση με τα αρχεία ISAM) και διατίθεται προς χρήση. Οι εγγραφές του διαστήματος ελέγχου με κλειδιά μεγαλύτερα από αυτό που διαγράφεται ολισθαίνουν κατά μία θέση, ώστε ο διατιθέμενος χώρος να ενωθεί με τον ελεύθερο χώρο του διαστήματος. Αυτή η διαδικασία λέγεται **δυναμική ανάκτηση χώρου** (dynamic space reclamation). Αν κατά την ενημέρωση μίας εγγραφής το μήκος της δεν αλλάξει, τότε δεν υπάρχει πρόβλημα, αν όμως το μήκος της ελαττωθεί ή αυξηθεί, τότε το σύστημα λειτουργεί σαν να πρόκειται για διαγραφή ή εισαγωγή, αντίστοιχα.

Όλες οι έννοιες και οι διαδικασίες που αναπτύχθηκαν θα επεξηγηθούν με τη βοήθεια ενός παραδείγματος. Έστω ότι στο δείγμα αρχείου VSAM του Σχήματος 7.6α εισάγεται η εγγραφή με κλειδί 11. Η διαδικασία αρχίζει με διάσχιση της ομάδας των καταλόγων με στόχο το αμέσως μεγαλύτερο κλειδί από το 11. Το κλειδί αυτό είναι το 150 που συνοδεύεται από ένα δείκτη προς το κατώτερο επίπεδο του καταλόγου. Νέα σάρωση για το αμέσως μεγαλύτερο κλειδί από το 11 καταλήγει στον εντοπισμό του κλειδιού 56 με τον αντίστοιχο δείκτη προς την ακολουθιακή ομάδα.



Σχήμα 7.6: Εισαγωγές και διαγραφές σε αρχείο VSAM.



Σχήμα 7.6: Εισαγωγές και διαγραφές σε αρχείο VSAM (συνέχεια).

Και πάλι η σάρωση της ακολουθιακής ομάδας για το αμέσως μεγαλύτερο κλειδί από το 11 καταλήγει στο κλειδί 12 και τον αντίστοιχο δείκτη προς το κατάλληλο διάστημα ελέγχου, όπου δεν υπάρχει διαθέσιμος χώρος για την εγγραφή 11. Συνεπώς στην περίπτωση αυτή το διάστημα διασπάται. Οι εγγραφές 4, 5 και 10 παραμένουν στο ίδιο διάστημα ελέγχου, ενώ οι εγγραφές 11 και 12 αποθηκεύονται σε νέο διάστημα ελέγχου, που βρίσκεται στο τέλος της περιοχής ελέγχου. Η νέα μορφή του αρχείου φαίνεται στο Σχήμα 7.6β. Σχετικά έχει ενημερωθεί και η ακολουθιακή ομάδα.

Έστω ότι εισάγεται η εγγραφή 25. Αναζήτηση διά μέσου της ομάδας των καταλόγων καταλήγει στην ίδια περιοχή ελέγχου. Το διάστημα ελέγχου, όπου η εγγραφή 25 πρέπει να αποθηκευθεί, έχει διαθέσιμο χώρο. Στην περίπτωση αυτή οι εγγραφές 52 και 56 ολισθαίνουν μία θέση προς τα πίσω, ώστε η εγγραφή 25 να καταλάβει το δημιουργούμενο κενό χώρο χωρίς να διαταραχθεί η λογική σειρά των εγγραφών. Δεν χρειάζονται αλλαγές στην ακολουθιακή ομάδα. Η νέα μορφή του αρχείου φαίνεται στο Σχήμα 7.6γ.

Έστω ότι πρέπει να διαγραφεί η εγγραφή 142 του Σχήματος 7.6γ. Με την απομάκρυνση της εγγραφής 142, οι εγγραφές 146 και 150 ολισθαίνουν μία θέση προς τα εμπρός, ώστε να ενοποιηθεί ο ελεύθερος χώρος του διαστήματος (όπως φαίνεται στο Σχήμα 7.6δ). Αν στη συνέχεια διαγραφεί το κλειδί 150, τότε πρέπει να ενημερωθεί και η ακολουθιακή ομάδα και η ομάδα των καταλόγων, ώστε να καταγραφεί το γεγονός ότι πλέον το μεγαλύτερο κλειδί του διαστήματος είναι το 146.

Το αρχείο VSAM υποστηρίζει και δύο άλλες υλοποιήσεις εκτός από την υλοποίηση με ακολουθία κλειδιών.

- Η υλοποίηση του αρχείου με ακολουθία εισόδων (entry-sequenced file) μοιάζει με τα σειριακά αρχεία, επειδή κάθε νέα εγγραφή παρατίθεται στο τέλος του αρχείου. Το σύστημα δεν κτίζει αυτόματα την ομάδα καταλόγου, αλλά είναι στη διάθεση του προγραμματιστή να κτίσει έναν κατάλογο χρησιμοποιώντας τη σχετική διεύθυνση σε bytes που επιστρέφεται από το σύστημα και παραμένει σταθερή. Προφανώς, η εκδοχή αυτή του αρχείου VSAM είναι χρήσιμη σε περιπτώσεις, όπου δεν είναι απαραίτητη η ταξινόμηση των εγγραφών αλλά αντίθετα είναι απαραίτητη η αποθήκευση των εγγραφών με τη σειρά άφιξής τους (όπως για παράδειγμα σε αρχεία συναλλαγών).
- Η υλοποίηση του αρχείου σχετικών εγγραφών (relative record file) δεν έχει κατάλογο και μπορεί να διαχειρισθεί μόνο εγγραφές σταθερού μήκους. Το αρχείο αυτό μπορεί να αποθηκεύσει μέχρις ένα συγκεκριμένο αριθμό εγγραφών, οπότε δεσμεύεται ανάλογος χώρος. Η εκδοχή αυτή στηρίζεται στον κατακερματισμό και έχει αρκετές ομοιότητες με τα τυχαία αρχεία που θα εξετασθούν σε επόμενο κεφάλαιο.

Είναι προφανές ότι οι δύο αυτές εκδοχές δεν διακρίνονται από δυναμική ανάκτηση χώρου.

Συμπερασματικά, τα αρχεία VSAM είναι πιο αποτελεσματικά από τα αρχεία ISAM για τους εξής βασικούς λόγους:

- δεν κάνουν διάκριση μεταξύ κύριας περιοχής και περιοχής υπερχειλίσεως,
- συνδυάζουν αυτόματα το χώρο που προκύπτει από τις διαγραφές με τον ελεύθερο χώρο του διαστήματος,
- εκτελούν αυτόματα τις απαραίτητες διασπάσεις διαστημάτων και περιοχών ελέγχου καθώς και τη δυναμική ανάκτηση χώρου, και
- διαχειρίζονται εγγραφές μεταβλητού μήκους.

7.4 Ασκήσεις

<1> Αρχείο ISAM αποθηκεύεται σε συσκευές μαγνητικών δίσκων με τα εξής χαρακτηριστικά: 200 κύλινδροι, 19 άτρακτοι/κύλινδρο, 14000 bytes/άτρακτο, 2000 bytes/σελίδα. Το αρχείο αποτελείται από 1.000.000 εγγραφές, όπου το μέγεθος του κλειδιού είναι 14 bytes, ενώ του δείκτη είναι 4 bytes. Πόσες συσκευές δίσκων χρειάζονται;

<2> Δίνεται αρχείο ISAM με 10.000 εγγραφές των 160 bytes. Το μήκος του κλειδιού είναι 16 bytes και του δείκτη 4 bytes. Το αρχείο είναι αποθηκευμένο σε δίσκο HP7925 με χαρακτηριστικά: 256 bytes/σελίδα, 64 σελίδες/άτρακτο, 9 άτρακτοι/κύλινδρο, ενώ η συσκευή αποτελείται από 815 κυλίνδρους. Να βρεθεί ο βέλτιστος παράγοντας ομαδοποίησης, ώστε να ελαχιστοποιηθεί το βάθος του καταλόγου. Πόσο χώρο σε κυλίνδρους καταλαμβάνει συνολικά το αρχείο και ποιάς είναι ο συνολικός παράγοντας χρησιμοποίησης;

<3> Η προηγούμενη άσκηση να λυθεί και πάλι για τα εξής δεδομένα: 1000 bytes/σελίδα, 24 σελίδες/άτρακτο και 12 άτρακτοι/κύλινδρο.

<4> Δίνεται αρχείο VSAM με χαρακτηριστικά ίδια με το αρχείο της Άσκησης 2 που είναι αποθηκευμένο σε δίσκο HP7925. Να βρεθεί το βάθος του καταλόγου, ο χώρος σε σελίδες που καταλαμβάνει συνολικά το αρχείο και ο συνολικός παράγοντας χρησιμοποίησης; Τι θα συμβεί στο βάθος του καταλόγου αν το αρχείο αποτελείται από 100.000 εγγραφές; Να εξετασθεί η συνέπεια της αύξησης του μεγέθους της σελίδας.

<5> Η προηγούμενη άσκηση να λυθεί και πάλι για τα εξής δεδομένα: 1000 bytes/σελίδα, 24 σελίδες/άτρακτο και 12 άτρακτοι/κύλινδρο.

<6> Δίνεται αρχείο ISAM με τα εξής χαρακτηριστικά: μήκος εγγραφής 100 bytes, μέγεθος σελίδας 1000 bytes, μέγεθος κλειδιού 38 bytes, μέγεθος δείκτη 4 bytes, ενώ ο αριθμός των εγγραφών είναι 100.000. Το αρχείο είναι αποθηκευμένο σε HP7925. Πόσα επίπεδα καταλόγου απαιτούνται; Να υπολογισθεί ο απαιτούμενος χρόνος:

- για μία δυαδική αναζήτηση μόνο στα δεδομένα του αρχείου, και
- για μία αναζήτηση στο αρχείο ως σύνολο.

<7> Αν δεν επιτρέπεται οι εγγραφές να αποθηκεύονται σε 2 σελίδες, τότε να βρεθούν αναλυτικές εκφράσεις (ως συνάρτηση των παραμέτρων n , R , K και B) για:

- το χώρο που το αρχείο καταλαμβάνει σε σελίδες,
- πόσες σελίδες καταλαμβάνει το πρώτο επίπεδο ενός στατικού καταλόγου, αν ο δείκτης χρειάζεται 2 bytes,
- πόσα επίπεδα καταλόγου απαιτούνται;
- πόσος είναι ο συνολικός χώρος που απαιτείται από όλα τα επίπεδα του καταλόγου;

<8> Είναι δυνατόν ο κατάλογος να είναι αποθηκευμένος σε διαφορετική συσκευή από το κυρίως αρχείο. Έστω ότι ένα αρχείο αποτελείται από 10.000 εγγραφές των 80 bytes με κλειδί των 8 bytes. Το μέγεθος της σελίδας είναι 1024 bytes, ενώ το μέγεθος του δείκτη είναι 2 bytes. Χρησιμοποιώντας τους τύπους της προηγούμενης άσκησης να βρεθεί ο αριθμός των επιπέδων του καταλόγου και ο συνολικός χώρος (σε bytes) που απαιτείται για την αποθήκευση του καταλόγου, αν το μέγεθος της σελίδας μεταβάλλεται από 256 bytes ως 3072 bytes με βήματα των 256 bytes.

<9> Σειριακό αρχείο με δείκτη έχει τρία επίπεδα καταλόγων. Από το διαχειριστή του συστήματος αποφασίζεται ότι αναδιοργάνωση πρέπει να γίνεται όταν η περιοχή υπερχείλισης, που έχει μέγεθος 20% της κύριας περιοχής, είναι πλήρης κατά 80%. Να υπολογισθεί ο μέσος χρόνος προσπέλασης μίας εγγραφής θεωρώντας ότι $Bfr=10$.

Κεφάλαιο 8

ΔΕΝΔΡΙΚΕΣ ΔΟΜΕΣ

8.1 Εισαγωγή

8.2 B-δένδρα

8.3 B*-δένδρα

8.4 B⁺-δένδρα

8.5 Άλλες παραλλαγές των B-δένδρων

8.6 Ασκήσεις

Κεφάλαιο 8

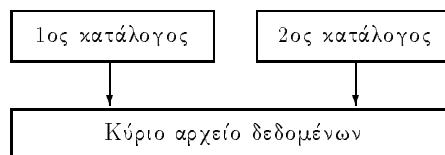
ΔΕΝΔΡΙΚΕΣ ΔΟΜΕΣ

8.1 Εισαγωγή

Ο κατάλογος είναι μία δομή που υλοποιεί μία συνάρτηση. Το όρισμα της συνάρτησης είναι η τιμή του κλειδιού και η τιμή της συνάρτησης είναι ο αριθμός της εγγραφής. Δηλαδή, είναι δυνατό με τη βοήθεια του καταλόγου να βρεθεί ο αριθμός (ή η θέση) μίας εγγραφής με βάση το κλειδί της. Ένας άλλος τρόπος για την επίτευξη αυτού του σκοπού είναι η μέθοδος του κατακερματισμού, που δεν απαιτεί μία συγκεκριμένη δομή (όπως ο κατάλογος), αλλά υποφέρει από το πρόβλημα των συνωνυμιών (δες Κεφάλαιο 9 βιβλίου για Δομές Δεδομένων).

Γενικά, ο κατάλογος είναι ανεξάρτητος από το αρχείο κατά δύο έννοιες, λογικά αλλά και φυσικά. Στην περίπτωση του αρχείου ISAM ο κατάλογος είναι μόνο λογικά ανεξάρτητος, αφού φυσικά διαχέεται μέσα στην κύρια περιοχή. Το κόστος αναζήτησης στον κατάλογο είναι σχετικά μικρό, και βέβαια πολύ πιο μικρό από το κόστος αναζήτησης ολόκληρης της εγγραφής κατ' ευθείαν μέσα στο αρχείο. Σε έναν κατάλογο το κλειδί μπορεί είτε να είναι ένα μόνο χαρακτηριστικό, είτε ένα μέρος χαρακτηριστικού ή συνδυασμός πολλών χαρακτηριστικών. Μπορεί να υπάρχει μόνο ένας κατάλογος (όπως στο αρχείο ISAM), αλλά μπορεί να υπάρχουν και περισσότεροι του ενός κατάλογοι. Στο Σχήμα 8.1 παρουσιάζεται μία περίπτωση αρχείου με δύο διαφορετικούς ανεξάρτητους καταλόγους.

Το μήκος μίας εγγραφής καταλόγου πρέπει να κρατηθεί όσο το δυνατό μικρότερο. Σε απλούς καταλόγους κάθε εγγραφή αποτελείται από την



Σχήμα 8.1: Αρχείο με δύο ανεξάρτητους καταλόγους.

τιμή του κλειδιού και τον αντίστοιχο δείκτη, ενώ πιο σύνθετοι κατάλογοι περιέχουν και άλλα πεδία. Μικρές εγγραφές στον κατάλογο επιτρέπουν μεγαλύτερο συντελεστή ομαδοποίησης, οπότε έτσι επιτυγχάνεται οικονομία στο χώρο αποθήκευσης και στο χρόνο εισόδου/εξόδου των δεδομένων.

Ο απλούστερος τύπος καταλόγου είναι ο γραμμικός κατάλογος (linear index), όπως για παράδειγμα ο κατάλογος ενός βιβλίου. Ένας γραμμικός κατάλογος περιέχει τις εγγραφές ταξινομημένες κατά αύξουσα τάξη χωρίς καμία άλλη δόμηση. Ένας τέτοιος κατάλογος μπορεί να προσπελασθεί σειριακά, αλλά είναι αποτελεσματικότερο να προσπελασθεί με δυαδική αναζήτηση ή αναζήτηση άλματος. Μάλιστα, η τελευταία μέθοδος χρησιμοποιείται για την ικανοποίηση ερωτήσεων διαστήματος. Ο κατάλογος αυτός διατηρεί όλα τα μειονεκτήματα των σειριακών αρχείων, δηλαδή μη αποτελεσματικές μεθόδους εισαγωγής, διαγραφής και αναδιοργάνωσης.

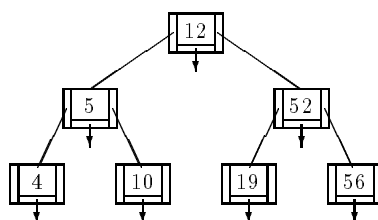
Δενδρικός κατάλογος (tree index) είναι ένας κατάλογος με συγκεκριμένη ιεραρχία επιπέδων. Η ρίζα του καταλόγου δεικτοδοτεί προς το δεύτερο επίπεδο του καταλόγου κοχ., ενώ το τελευταίο επίπεδο (δηλαδή τα φύλλα) περιέχουν δείκτες προς το αρχείο των δεδομένων. Προφανώς, ο κατάλογος των αρχείων ISAM, που εξετάσθηκαν στο προηγούμενο κεφάλαιο, ικανοποιεί τον προηγούμενο ορισμό. Οι δενδρικοί κατάλογοι διακρίνονται σε δύο είδη: τα **ετερογενή** (heterogeneous) και τα **ομογενή** (homogeneous) δένδρα. Ετερογενή είναι τα δένδρα, των οποίων οι κόμβοι περιέχουν μόνο ένα είδος δεικτών. Οι κόμβοι των μεσαίων επιπέδων περιέχουν μόνο δείκτες σε κόμβους, ενώ οι κόμβοι του τελευταίου επιπέδου περιέχουν μόνο δείκτες σε δεδομένα. Αντίθετα, στα ομογενή δένδρα υπάρχουν δύο είδη δεικτών, οι **δείκτες δεδομένων** (data pointers) και οι **δενδρικοί δείκτες** (tree pointers). Στα μεσαία επίπεδα και οι δύο τύποι είναι ενεργοί, ενώ στο τελευταίο επίπεδο είναι ενεργοί μόνο οι δείκτες δεδομένων.

Στο Σχήμα 8.2 παρουσιάζεται η εσωτερική δομή ενός κόμβου ομογενούς δένδρου, όπου οι δενδρικοί δείκτες βρίσκονται μεταξύ των κλειδιών,

Δενδρικός δείκτης	Κλειδί Δείκτης	Δενδρικός δείκτης	...	Δενδρικός δείκτης	Κλειδί Δείκτης	Δενδρικός δείκτης
----------------------	-------------------	----------------------	-----	----------------------	-------------------	----------------------

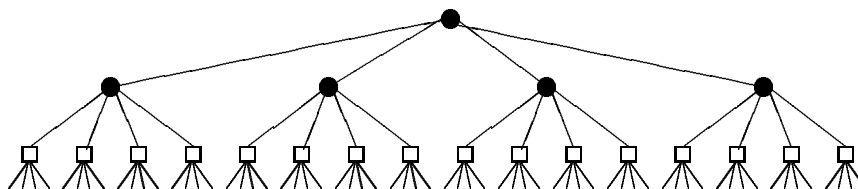
Σχήμα 8.2: Κόμβος ομογενούς δένδρου.

ενώ οι δείκτες δεδομένων βρίσκονται κάτω από τα κλειδιά. Όπως φαίνεται, για κάθε κλειδί υπάρχει ένας δενδρικός δείκτης, ένας δείκτης δεδομένων και επιπλέον ένας ακόμη δενδρικός δείκτης. Ο πρώτος δενδρικός δείκτης κατευθύνει προς κλειδιά που είναι μικρότερα από το πρώτο κλειδί του κόμβου, ενώ ο τελευταίος δενδρικός δείκτης κατευθύνει προς κλειδιά μεγαλύτερα από το τελευταίο κλειδί του κόμβου. Κάθε άλλος δενδρικός δείκτης κατευθύνει προς κάποιον κόμβο του δένδρου με τιμές κλειδιών μεταξύ των τιμών των κλειδιών που περιλαμβάνουν το δείκτη. Σε περίπτωση επίσκεψης του κόμβου, όπου πραγματικά βρίσκεται αποθηκευμένο το αναζητούμενο κλειδί, τότε ο δείκτης δεδομένων κατευθύνει προς την αντίστοιχη εγγραφή.



Σχήμα 8.3: Ομογενές δένδρο.

Στα Σχήμα 8.3 και 8.4 παρουσιάζονται δύο παραδείγματα ομογενούς και ετερογενούς δένδρου, αντίστοιχα. Μία βασική διαφορά μεταξύ ετερογενών



Σχήμα 8.4: Ετερογενές δένδρο.

και ομογενών δένδρων είναι το ύψος του δένδρου, που έχει ως αποτέλεσμα διαφορετική επίδοση στην αναζήτηση. Η μέση τιμή προσπελάσεων για αναζήτηση είναι μεγαλύτερη για τα ετερογενή δένδρα, γιατί η αναζήτηση φθάνει οπωσδήποτε μέχρι τα φύλλα του δένδρου. Το σχετικό πλεονέκτημα των ομογενών δένδρων ισοζυγίζεται από τον επιπλέον απαιτούμενο χώρο για δείκτες.

Στο δένδρο του Σχήματος 8.4 ο λόγος διακλάδωσης είναι $y=4$, οπότε το δένδρο στα 3 επίπεδα του μπορεί να χωρέσει 64 κλειδιά. Γενικά, προκύπτει ότι το ύψος, h , του καταλόγου που δεικτοδοτεί n εγγραφές είναι:

$$h \geq \log_y n$$

Ένα ομογενές δένδρο με ύψος 3 και λόγο διακλάδωσης 4 μπορεί να αποθηκεύσει 63 κλειδιά. Γενικά σε ένα ομογενές δένδρο με h επίπεδα θα υπάρχουν $(y-1)$ κλειδιά στη ρίζα, $y(y-1)$ κλειδιά στο δεύτερο επίπεδο, $y^2(y-1)$ κλειδιά στο τρίτο κοχ. Άρα, ο συνολικός αριθμός των εγγραφών που δεικτοδοτούνται θα είναι:

$$(y^0 + y^1 + y^2 + \dots + y^{h-1}) \times (y - 1) = y^h - 1$$

και συνεπώς:

$$n \leq y^h - 1 \iff h \geq \log_y(n + 1)$$

Το δυσκολότερο μέρος της διαχείρισης ενός καταλόγου είναι η εισαγωγή και η διαγραφή χωρίς να καταστεί αναγκαία η γενική αναδιοργάνωση του καταλόγου. Για κάθε είδος καταλόγου υπάρχουν οι σχετικοί αλγόριθμοι. Στη συνέχεια θα εξετασθεί μία οικογένεια δένδρων, τα λεγόμενα B-δένδρα, που είναι η βασική δομή για καταλόγους. Τα δένδρα της οικογένειας αυτής χρησιμοποιούνται σε όλα σχεδόν τα εμπορικά συστήματα διαχείρισης βάσεων δεδομένων, όπως για παράδειγμα στο σύστημα της Oracle, της Sybase, στο DB2 της IBM, στο Ingres της Computer Associates, στο SQL Server της Microsoft κλπ κλπ. Οι δομές θα παρουσιασθούν με παραδείγματα και θα γίνει ανάλυση των χρονικών κόστων για τις διάφορες διεργασίες. Στο κεφάλαιο των ειδικών θεμάτων θα γίνει και πάλι αναφορά στις δομές αυτές στο πλαίσιο των προβλημάτων που δημιουργούνται κατά την ταυτόχρονη προσπέλασή τους από πολλούς χρήστες.

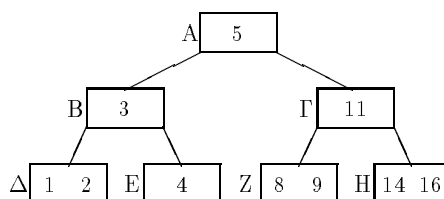
8.2 B-δένδρα

Η δομή αυτή προτάθηκε από τους Bayer και McCreight το 1972. Λέγεται ότι η ονομασία του δένδρου, το 'B', προέρχεται από το επώνυμο του Bayer, ή από την ονομασία της εταιρείας Boeing όπου εργαζόταν ο Bayer, ή ακόμη ότι προέρχεται από τη λέξη balanced (ισοζυγισμένο). Το βέβαιο είναι ότι δεν προέρχεται από τη λέξη binary (δυαδικό).

Το B-δένδρο είναι μία μερική περίπτωση του δένδρου αναζήτησης m δρόμων (m -way search tree), που έχει αναπτυχθεί στο βιβλίο των Δομών Δεδομένων. Σύμφωνα με έναν ορισμό, ως **B-δένδρο τάξης m** ορίζεται το ομογενές δένδρο με τα ακόλουθα χαρακτηριστικά:

- η ρίζα έχει το ελάχιστο δύο παιδιά,
- οι εσωτερικοί κόμβοι (εκτός της ρίζας) έχουν το ελάχιστο $\lceil m/2 \rceil$ παιδιά και το μέγιστο m παιδιά,
- ένας εσωτερικός κόμβος με k κλειδιά έχει $k+1$ παιδιά, και
- οι εξωτερικοί κόμβοι βρίσκονται στο ίδιο επίπεδο.

Στη βιβλιογραφία συναντάται ένας παρόμοιος ορισμός του **B-δένδρου βαθμού (degree) d** , που διαφέρει από τον προηγούμενο μόνο στο δεύτερο σχέλος. Δηλαδή, οι εσωτερικοί κόμβοι, εκτός της ρίζας, περιέχουν το ελάχιστο d κλειδιά και το μέγιστο $2d$ κλειδιά. Προφανώς ο δεύτερος ορισμός είναι μερική περίπτωση του πρώτου. Το B-δένδρο του Σχήματος 8.5 είναι τρίτης τάξης ή πρώτου βαθμού.



Σχήμα 8.5: B-δένδρο τάξης 3.

Ο παράγοντας **χρησιμοποίησης χώρου** (storage utilization factor), U , ορίζεται ως ο λόγος του αριθμού των αποθηκευμένων κλειδιών προς το

το σύνολο των διαθέσιμων θέσεων. Από τον ορισμό φαίνεται ότι η ελάχιστη και η μέγιστη τιμή του παράγοντα χρησιμοποίησης χώρου είναι περίπου $U_{min}=50\%$ και $U_{max}=100\%$, αντίστοιχα. Σχετικά με την εύρεση της μέσης τιμής του παράγοντα αυτού έχουν εμφανισθεί αρκετές δημοσιεύσεις στη βιβλιογραφία που είναι όμως από λίγο ως πολύ σύνθετες. Ακολουθεί μία απλοποιητική αλλά προσεγγιστική μέθοδος που δόθηκε από τον Leung (1984). Αν n ο αριθμός των κλειδιών και nod ο αριθμός των κόμβων του δένδρου, τότε από το δεύτερο ορισμό του B-δένδρου (που δεν θα χρησιμοποιηθεί άλλο στη συνέχεια) ισχύει:

$$U = \frac{n}{2d \times nod}$$

Για μία σταθερή τιμή του n ισχύει:

$$E[U] = \frac{n}{2d} \times E\left[\frac{1}{nod}\right]$$

ενώ για την ελάχιστη και τη μέγιστη τιμή του nod προκύπτει αντίστοιχα:

$$nod_{max} = \frac{n}{2d \times U_{min}} \quad nod_{min} = \frac{n}{2d}$$

Υποθέτοντας προσεγγιστικά μία συνεχή ομοιόμορφη κατανομή στο διάστημα $[nod_{min}, nod_{max}]$ προκύπτει:

$$E\left[\frac{1}{nod}\right] = \frac{1}{nod_{max} - nod_{min}} \int_{nod_{min}}^{nod_{max}} \frac{1}{nod} dnod = \frac{2d U_{min}}{n \times (1 - U_{min})} \times \ln \frac{1}{U_{min}}$$

Αντικαθιστώντας $U_{min}=50\%$ προκύπτει ότι $E[U] = \ln 2 \approx 69\%$. Σημειώνεται ότι στην ανάλυση αυτή θα γίνει αργότερα αναφορά σε σχέση με άλλες δομές. Για το λόγο αυτό, το U_{min} θεωρήθηκε ως παράμετρος.

Από τον ορισμό του B-δένδρου είναι εύκολο να υπολογισθεί η μέση τιμή των προσπελάσεων για επιτυχή αναζήτηση στη χειρότερη περίπτωση. Έστω, λοιπόν, ένα B-δένδρο τάξης m και ύψους h . Ο ελάχιστος αριθμός κλειδιών που μπορεί να περιέχει είναι:

$$n = 2 \times \left\lceil \frac{m}{2} \right\rceil^{h-1} - 1$$

ενώ ο μέγιστος αριθμός κλειδιών που μπορεί να περιέχει είναι:

$$n = m^h - 1$$

Συνεπώς η τιμή του ύψους ενός B-δένδρου περικλείεται μεταξύ των ορίων:

$$\log_m(n+1) \leq h \leq 1 + \log_{m/2} \frac{n+1}{2}$$

Η εύρεση της βέλτιστης τιμής του m είναι ένα ενδιαφέρον πρόβλημα. Σε μία πρακτική υλοποίηση ενός καταλόγου με τη δομή του B-δένδρου οι εγγραφές θα αποτελούνται από μία τριάδα πεδίων: το κλειδί, το δενδρικό δείκτη, και το δείκτη των δεδομένων. Επομένως συνολικά το μέγεθος ενός κόμβου είναι $m \times (K + 2P)$ bytes περίπου. Ο χρόνος για την ανάγνωση ενός τυχαίου κόμβου είναι:

$$s + r + \frac{m \times (K + 2P)}{t'}$$

Δοθέντος του κόμβου η εύρεση ενός κλειδιού μπορεί να γίνει με δυαδική αναζήτηση σε χρόνο:

$$c \times \log m + d$$

όπου τα c και d είναι σταθερές. Απλοποιώντας τον προηγούμενο τύπο για το ύψος, προκύπτει ότι στη χειρότερη περίπτωση το ύψος είναι:

$$e \times \frac{\log \frac{n+1}{2}}{\log m}$$

όπου το e σταθερά. Επομένως ο μέγιστος χρόνος αναζήτησης δίνεται από τον τύπο:

$$e \times \log \frac{n+1}{2} \times \left(\frac{s + r + d + \frac{m \times (K+2P)}{t'}}{\log m} + c \right)$$

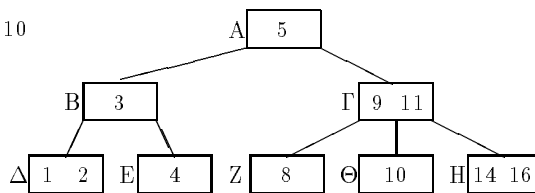
που ελαχιστοποιείται αν ελαχιστοποιηθεί το άθροισμα της αγγύλης. Έτσι σε κάθε περίπτωση με αντικατάσταση των παραμέτρων και χρησιμοποιώντας μία υπολογιστική μέθοδο λαμβάνεται το βέλτιστο m . Αν προκύψει μέγεθος κόμβου μεγαλύτερο από το μέγεθος των απομονωτικών μνημών, τότε θεωρείται η τιμή του m που δεν ξεπερνά το μέγεθος των μνημών.

8.2.1 Εισαγωγή σε B-δένδρο

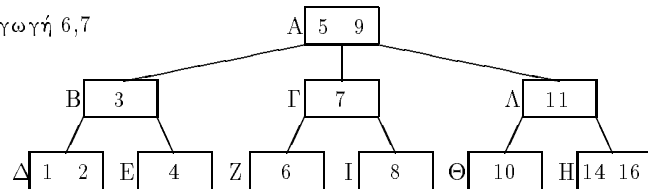
Για την εισαγωγή μίας εγγραφής σε ένα B-δένδρο, αρχικά με βάση το κλειδί της εγγραφής εντοπίζεται το φύλλο όπου πρέπει να γίνει η εισαγωγή. Αν

το φύλλο έχει ελεύθερο χώρο, τότε η εισαγωγή είναι εύκολη υπόθεση. Αλλιώς, απαιτείται ιδιαίτερη προσοχή καθώς πρέπει ο συγκεκριμένος κόμβος να διασπασθεί σε δύο κόμβους, ώστε να αποθηκευθούν όλες οι εγγραφές. Η διάσπαση ενός φύλλου μπορεί να προκαλέσει αλυσιδωτή αντίδραση και να χρειστούν περαιτέρω διασπάσεις κόμβων σε ανώτερα επίπεδα. Η διαδικασία των διασπάσεων θα διευκρινισθεί με το επόμενο παράδειγμα.

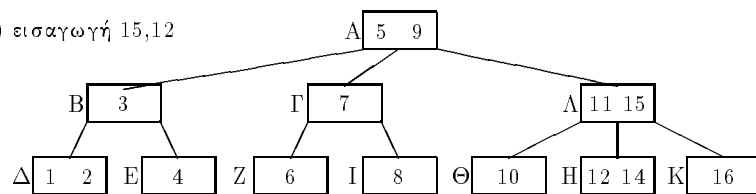
(α) εισαγωγή 10



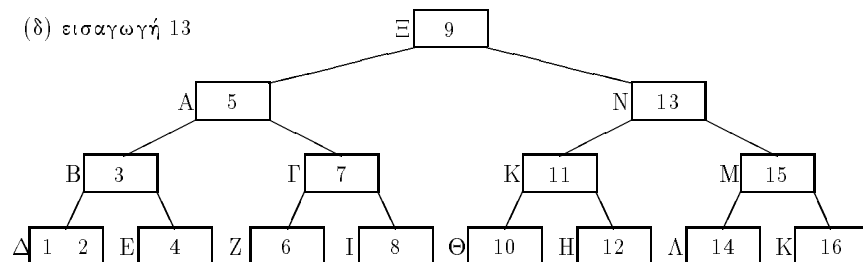
(β) εισαγωγή 6,7



(γ) εισαγωγή 15,12



(δ) εισαγωγή 13



Σχήμα 8.6: B-δένδρο με διαδοχικές εισαγωγές.

Έστω ότι στο δένδρο του Σχήματος 8.5 εισάγεται το κλειδί 10, που κατευθύνεται στον κόμβο Z. Ο ορισμός του δένδρου παραβιάζεται, συνεπώς πρέπει να δημιουργηθεί νέος κόμβος όπως φαίνεται στο Σχήμα 8.6α. Το

μεσαίο από τα κλειδιά 8, 9 και 10 (δηλαδή το 9) έχει ανέβει στον πατρικό κόμβο, ενώ το 8 επανα-αποθηκεύεται στον κόμβο Z και το 10 αποθηκεύεται στο νέο κόμβο Θ.

Η εισαγωγή του κλειδιού 6 στο δένδρο του Σχήματος 8.6α γίνεται εύκολα και γι' αυτό το νέο σχήμα παραλείπεται. Στη συνέχεια, έστω ότι εισάγεται το κλειδί 7, οπότε προκύπτει το Σχήμα 8.6β. Όπως και κατά την εισαγωγή του 10, έτσι και εδώ το μεσαίο κλειδί (δηλαδή το 7) προχωρεί ένα επίπεδο προς τη ρίζα. Όμως και στο νέο κόμβο παραβιάζεται ο ορισμός του B-δένδρου. Συνεπώς πρέπει αναδρομικά να εφαρμοσθεί η ίδια διαδικασία. Έτσι, το μεσαίο κλειδί (δηλαδή το 9) ανεβαίνει και πάλι ένα επίπεδο και αποθηκεύεται στη ρίζα. Κατά παρόμοιο τρόπο με την εισαγωγή των κλειδιών 15 και 12 προκύπτει το Σχήμα 8.6γ. Αν, όμως, στο σημείο αυτό εισαχθεί το κλειδί 13, τότε μετά από διαδοχικές διασπάσεις κόμβων θα διασπασθεί και η ρίζα. Έτσι προκύπτει το Σχήμα 8.6δ.

Η ερώτηση είναι πόσο συχνά γίνεται η διάσπαση των κόμβων. Είναι φανερό ότι η πρώτη διάσπαση (δηλαδή η διάσπαση της αρχικής ρίζας) δημιουργεί δύο νέους κόμβους, ενώ οι επόμενες δημιουργούν από ένα μόνο νέο κόμβο μέχρι και $h-1$ νέους κόμβους στην περίπτωση που η διάσπαση μεταφερθεί μέχρι τη ρίζα (όπου h είναι το ύψος του δένδρου). Έτσι, όταν το δένδρο έχει ήδη p κόμβους, έχουν συμβεί $p-2$ διασπάσεις. Κάθε κόμβος έχει το ελάχιστο $\lceil m/2 \rceil - 1$ κλειδιά, εκτός από τη ρίζα που έχει το ελάχιστο ένα κλειδί. Συνεπώς ένα δένδρο με p κόμβους περιέχει το ελάχιστο $1 + (p-1) \times (\lceil m/2 \rceil - 1)$ κλειδιά. Η πιθανότητα να συμβεί διάσπαση με την εισαγωγή ενός νέου κλειδιού είναι λιγότερο από:

$$\frac{p-2}{1 + (p-1) \times (\lceil m/2 \rceil - 1)}$$

που είναι λιγότερο από 1 διάσπαση ανά $\lceil m/2 \rceil - 1$ εισαγωγές κλειδιών. Ασυμπτωτικά (δηλαδή, μεγάλο p) για $m=10$ και $m=20$ η πιθανότητα διάσπασης είναι 0.25 και 0.204, αντίστοιχα. Δηλαδή, όσο μεγαλύτερη είναι η τάξη του δένδρου, τόσο αυτή η πιθανότητα μικραίνει.

Η προηγούμενη ανάλυση είναι προσεγγιστική. Επακριβής ανάλυση των χαρακτηριστικών του B-δένδρου είναι εξαιρετικά πολύπλοκη και μέχρι στιγμής περιορίζεται στα κατώτερα επίπεδα. Η ανάλυση αυτή λέγεται **ανάλυση κρασπέδου** (fringe analysis) και χρησιμοποιεί αλυσίδες Markov. Η σημαντικότερη δημοσίευση στη βιβλιογραφία εμφανίσθηκε το 1982 από τους

Eisenbarth et al., όπου έχει αποδειχθεί ότι η πιθανότητα να συμβεί μία ή περισσότερες διασπάσεις κατά την εισαγωγή του $n+1$ κλειδιού είναι:

$$\frac{1}{(m-1) \times \ln 2} + O(m^{-2})$$

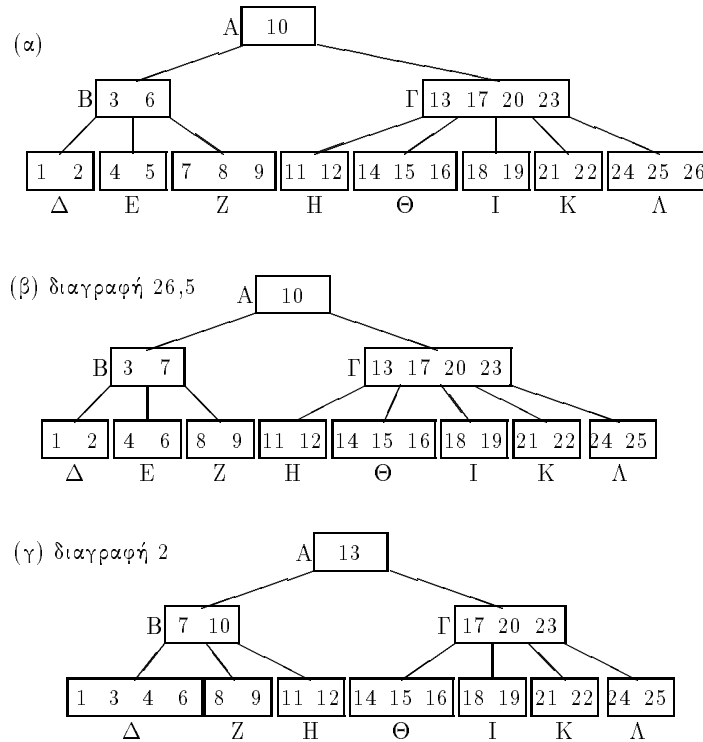
8.2.2 Διαγραφή σε B-δένδρο

Η διαγραφή σε B-δένδρο είναι σαφώς δυσκολότερη διεργασία από την εισαγωγή. Αυτό οφείλεται στο γεγονός ότι με τη διαγραφή ενός κλειδιού μπορεί να παραβιασθεί ο ορισμός του B-δένδρου και να χρειασθεί συγχώνευση δύο γειτονικών κόμβων. Μάλιστα, η συγχώνευση δύο κόμβων μπορεί να προκαλέσει άλλες συγχωνεύσεις κόμβων σε ανώτερα επίπεδα. Υπενθυμίζεται στο σημείο αυτό ότι το αρχείο VSAM είναι δυναμικό κατά τις εισαγωγές, όμως δεν διακρίνεται για δυναμικότητα κατά τις διαγραφές. Αυτή είναι η κυριότερη διαφορά μεταξύ αρχείου VSAM και B-δένδρου. Η διαδικασία των συγχωνεύσεων θα διευκρινισθεί με το επόμενο παράδειγμα.

Έστω, λοιπόν, ότι δίνεται ένα B-δένδρο τάξης 5, όπως στο Σχήμα 8.7α. Αρχικά παρουσιάζονται τρεις περιπτώσεις όπου το κλειδί που πρέπει να διαγραφεί ανήκει σε ένα φύλλο. Για παράδειγμα, έστω ότι πρέπει να διαγραφεί το κλειδί 26. Αυτή η πρώτη περίπτωση είναι η απλούστερη και εκτελείται χωρίς άλλες παρενέργειες, επειδή ο ορισμός δεν διαταράσσεται.

Έστω ότι στη συνέχεια πρέπει να διαγραφεί το κλειδί 5. Πιθανή διαγραφή του κλειδιού 5 θα διατάραζε τον ορισμό, επειδή τελικά στον κόμβο θα έμενε μόνο ένα κλειδί, δηλαδή λιγότερο από $\lceil m/2 \rceil - 1 = 2$. Στην περίπτωση αυτή ελέγχεται αν ο γειτονικός στα δεξιά κόμβος Z έχει περισσότερο από 2 κλειδιά. Αυτή η συνθήκη συμβαίνει, αφού ο κόμβος Z έχει τρία κλειδιά. Επίσης, φαίνεται ότι ο πατρικός κόμβος του κόμβου E (δηλαδή ο B) περιέχει το κλειδί 6 που αποτελεί το αμέσως μεγαλύτερο κλειδί από το 5. Άρα είναι δυνατό να διαγραφεί το κλειδί 5 με ταυτόχρονη άνοδο του κλειδιού 7 από τον κόμβο Z στον κόμβο B, και χάθοδο του κλειδιού 6 από τον κόμβο B στον κόμβο E. Έτσι προκύπτει το B-δένδρο του Σχήματος 8.7β. Αν ο δεξιός γειτονικός κόμβος δεν είχε περισσότερο από 2 κόμβους, τότε η προηγούμενη διαδικασία θα υλοποιούνταν με το γειτονικό αριστερό κόμβο και το αντίστοιχο κλειδί του πατρικού κόμβου. Αυτή ήταν η δεύτερη περίπτωση που μπορεί να παρουσιαθεί σε περίπτωση διαγραφής.

Αν δεν μπορεί να βρεθεί παρόμοια λύση ούτε από δεξιά ούτε από αριστερά, τότε πρέπει να γίνει συγχώνευση (η τρίτη περίπτωση). Η συγχώνευση

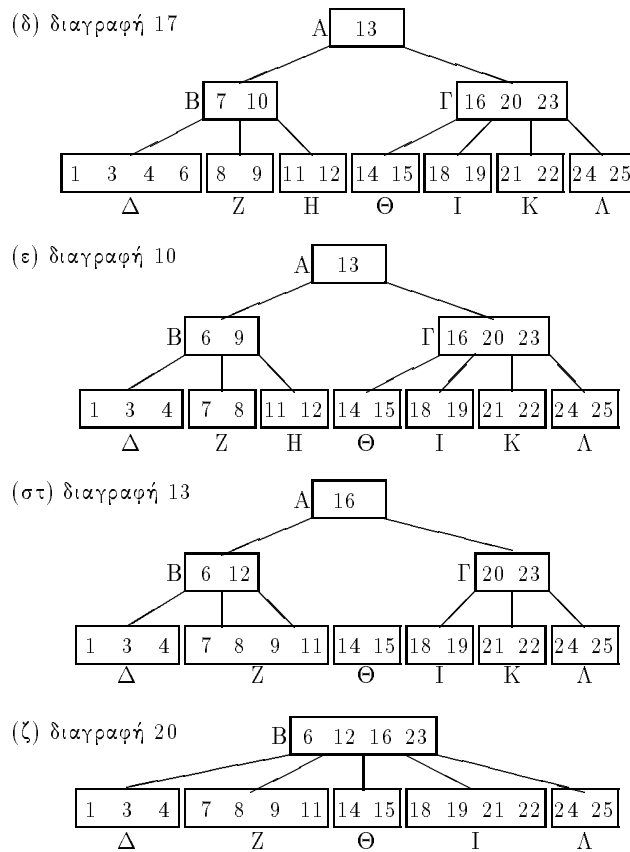


Σχήμα 8.7: B-δένδρο με διαδοχικές διαγραφές.

αφορά σε τρεις κόμβους, όπου οι δύο είναι από το κατώτερο επίπεδο και ο ένας από το ανώτερο. Για παράδειγμα, έστω ότι πρέπει να διαγραφεί το κλειδί 2. Η διαδικασία ακολουθεί τα εξής βήματα. Πρώτον, το κλειδί 1 που απομένει στον κόμβο Δ, το κλειδί 3 του κόμβου B και τα κλειδιά 4 και 6 του κόμβου E συγχωνεύονται και αποθηκεύονται στον κόμβο Δ. Ο κόμβος E δεν έχει πλέον κανένα κλειδί και αποδίδεται στο σύστημα, ενώ ο κόμβος B έχει μόνο ένα. Το δεύτερο βήμα είναι παρόμοιο προς τη διαδικασία που αναπτύχθηκε κατά τη διαγραφή του κλειδιού 26. Δηλαδή, ελέγχεται αν ο δεξιός γείτονας του κόμβου B (δηλαδή ο Γ) έχει περισσότερα από 2 κλειδιά. Αυτό πράγματι συμβαίνει, οπότε το κλειδί 13 του κόμβου Γ ανεβαίνει κατά ένα επίπεδο και φθάνει στη ρίζα, ενώ το κλειδί της ρίζας που είναι αμέσως μεγαλύτερο από το 7 (δηλαδή το 10) κατεβαίνει στον κόμβο B. Στο Σχήμα 8.7γ φαίνεται η τελική μορφή που παίρνει το B-δένδρο μετά τη διαγραφή του κλειδιού 2.

Μέχρι τώρα εξετάστηκαν περιπτώσεις διαγραφής που το κλειδί είναι αποθηκευμένο σε φύλλο. Η διαδικασία διαγραφής κλειδιού που είναι αποθηκευμένο σε εσωτερικό κόμβο είναι λίγο διαφορετική. Στην περίπτωση αυτή πρέπει να εντοπισθεί το λεξικογραφικά αμέσως προηγούμενο κλειδί και να καταλάβει τη θέση του διαγραφόμενου. Και πάλι διακρίνονται μερικές περιπτώσεις.

Για παράδειγμα, έστω ότι από το δένδρο του Σχήματος 8.7γ πρέπει να διαγραφεί το κλειδί 17. Ελέγχεται, λοιπόν, αν ο κόμβος Θ περιέχει περισσότερο από 2 κλειδιά. Η συνθήκη αυτή πράγματι συμβαίνει και συνεπώς το κλειδί 16 ανέρχεται κατά ένα επίπεδο στον κόμβο Γ, ώστε να μη διαταραχθεί ο ορισμός του B-δένδρου. Φαίνεται ότι αυτή η περίπτωση διαγραφής αντιμε-



Σχήμα 8.7: B-δένδρο με διαδοχικές διαγραφές (συνέχεια).

τωπίσθηκε εύκολα 'ενοχλώντας' μόνο έναν κόμβο παιδί του κόμβου, όπου ανήκε το κλειδί που διαγράφηκε. Η νέα μορφή του δένδρου παρουσιάζεται στο Σχήμα 8.7δ.

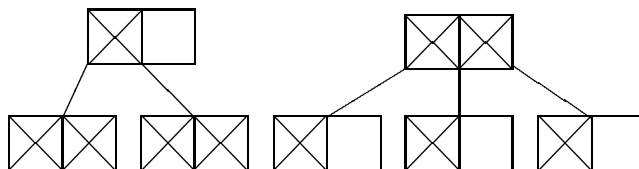
Έστω ότι στη συνέχεια πρέπει να διαγραφεί το κλειδί 10. Αν όμως ανέλθει το κλειδί 9 για να καλύψει το κενό, τότε δεν ισχύει πλέον ο ορισμός για τον κόμβο Z, γιατί περιέχει μόνο το κλειδί 8. Η συνέχεια ακολουθεί κατά τα γνωστά. Δηλαδή, ελέγχεται προς τα δεξιά ο κόμβος H, αλλά αυτός δεν μπορεί να συνεισφέρει. Ο προς τα αριστερά κόμβος Δ έχει περισσότερα από 2 κλειδιά. Άρα, με άνοδο του κλειδιού 6 από τον κόμβο Δ στον κόμβο B και κάθοδο του κλειδιού 7 από τον κόμβο B στον κόμβο Z το δένδρο καταλήγει στην τελική μορφή του Σχήματος 8.7ε. Έτσι, στην περίπτωση αυτή υπεισέλθαν τρεις κόμβοι στη διαδικασία αποκατάστασης της ισορροπίας.

Απομένει να εξετασθούν δύο άλλες περιπτώσεις διαγραφής. Έστω ότι πρέπει να διαγραφεί το κλειδί 13 της ρίζας. Στην περίπτωση αυτή το αμέσως μικρότερο κλειδί 12 παίρνει τη θέση του 13 στη ρίζα και συνεπώς το πρόβλημα μεταφέρεται στον κόμβο H, που απομένει μόνο με το κλειδί 11. Ο κόμβος H δεν έχει δεξιό αδελφό, ενώ ο αριστερός του αδελφός Z δεν μπορεί να συνεισφέρει, γιατί έχει μόλις δύο κλειδιά. Σε επόμενο βήμα, το περιεχόμενο των κόμβων Z και H μαζί με το κλειδί από τον κόμβο B, που ανήκει στο μέσο των διαστημάτων τιμών των δύο κόμβων (δηλαδή το 9) συγχωνεύονται σε ένα κόμβο και αποθηκεύονται στη θέση του κόμβου Z. Άρα, το πρόβλημα έχει μεταφερθεί στον κόμβο B που περιέχει μόνο το κλειδί 6. Η λύση πλέον μπορεί να δοθεί μόνο από τον αδελφό του κόμβου B, δηλαδή τον κόμβο Γ. Στον κόμβο B κατέρχεται το κλειδί 12 (που προηγουμένως είχε ανέλθει στη ρίζα), ενώ το κλειδί 16 του κόμβου Γ ανέρχεται στη ρίζα. Ταυτόχρονα όμως, επειδή αλλάζουν τα διαστήματα τιμών που δεικτοδοτούνται από τη ρίζα, ο κόμβος Θ δεν δεικτοδοτείται πλέον από τον κόμβο Γ αλλά από τον κόμβο B. Η τελική μορφή του δένδρου παρουσιάζεται στο Σχήμα 8.7στ.

Η τελευταία περίπτωση διαγραφής είναι εκείνη που προκαλεί την ελάττωση του ύψους του δένδρου. Έστω, λοιπόν, ότι πρέπει να διαγραφεί το κλειδί 20. Αρχικά το κλειδί 19 ανεβαίνει από τον κόμβο I στον κόμβο Γ για να καλύψει το κενό. Το πρόβλημα πλέον βρίσκεται στον κόμβο I που περιέχει μόνο το κλειδί 18. Όπως και πριν, το περιεχόμενο των κόμβων I και K μαζί με το κλειδί 19, που βρίσκεται μεταξύ των διαστημάτων τιμών των κόμβων αυτών, αποθηκεύονται στον κόμβο I, ενώ ο κόμβος K αποδίδεται και πάλι στο σύστημα. Τώρα πλέον το πρόβλημα βρίσκεται στον κόμβο Γ

που έχει μόνο το κλειδί 23. Ακόμη ο αριστερός αδελφός και η ρίζα έχουν τον ελάχιστο αριθμό κλειδιών. Άρα, θα γίνει νέα συγχώνευση. Έτσι, το περιεχόμενο των κόμβων A, B και Γ αποθηκεύεται στον κόμβο B και οι άλλοι αποδίδονται στο σύστημα. Η τελική μορφή του δένδρου φαίνεται στο Σχήμα 8.7ζ, όπου φαίνεται ότι το ύψος του δένδρου από τρία έγινε δύο.

Όσον αφορά στην επίδοση της διαγραφής αναφέρεται ότι στη χειρότερη περίπτωση (όπου μία διαγραφή θα προκαλέσει συγχώνευση της ρίζας) θα γίνουν $2h-1$ προσπελάσεις κόμβων για ανάγνωση και $h+1$ προσπελάσεις για αποθήκευση. Ωστόσο, αυτή η περίπτωση δεν είναι ιδιαίτερα πιθανή. Κατά μέσο όρο απαιτείται η ανάγνωση $h+1+1/k$ κόμβων και η επανα-αποθήκευση $4+2/k$ κόμβων, όπου $k = \lceil m/2 \rceil - 1$.

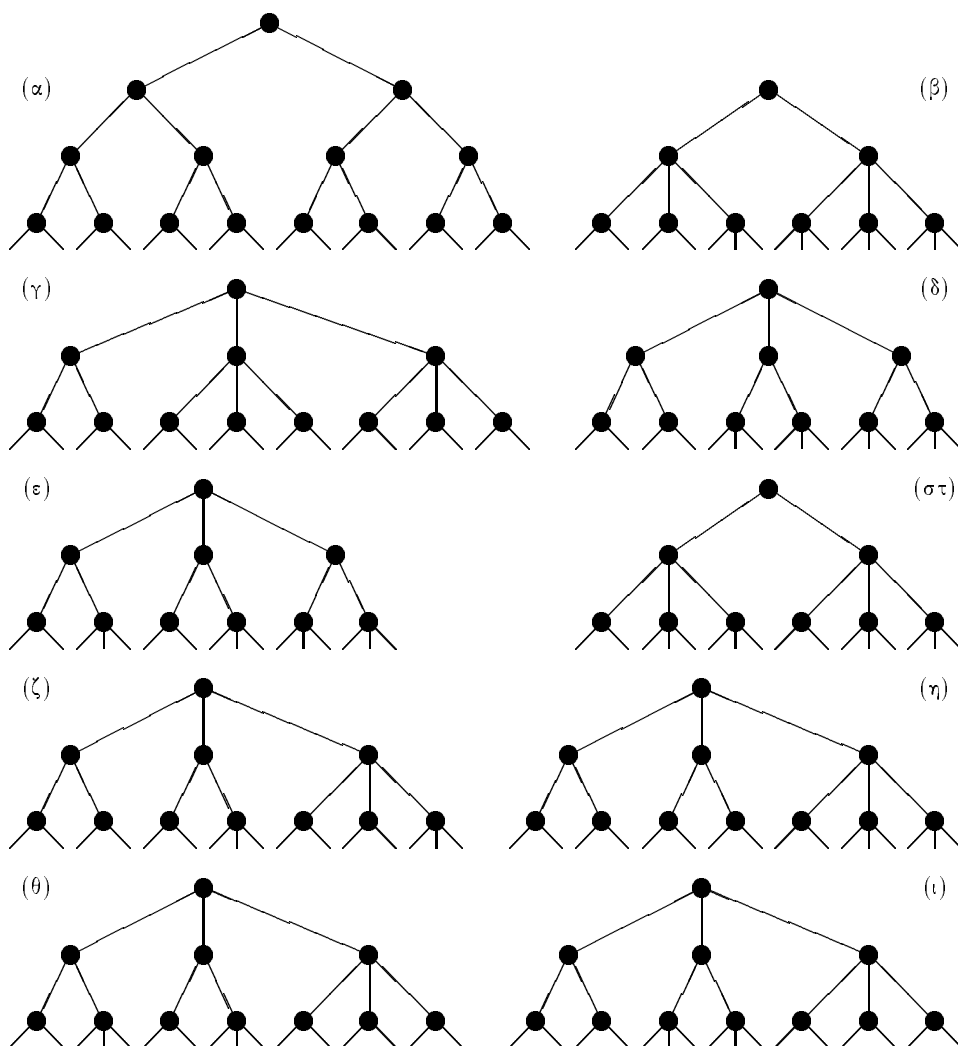


Σχήμα 8.8: Ομάδα BT(3,5).

Είναι ευνόητο ότι από τη μορφή του B-δένδρου εξαρτάται ο συνολικός καταλαμβανόμενος χώρος (και επομένως ο παράγοντας χρησιμοποίησης χώρου) και η επίδοση κατά την επιτυχή ή την ανεπιτυχή αναζήτηση. Στο Σχήμα 8.8 φαίνονται δύο διακριτές δομές B-δένδρων τάξης 3 που περιέχουν 5 κλειδιά. Οι δύο αυτές εκδοχές καταλαμβάνουν 3 και 4 κόμβους αντίστοιχα, έχουν παράγοντα χρησιμοποίησης χώρου $5/6$ και $5/8$ αντίστοιχα, ενώ η μέση τιμή προσπελάσεων για επιτυχή αναζήτηση είναι $9/5$ και $8/5$ αντίστοιχα. Ο αναγνώστης θα διαπιστώσει ότι εκτός από αυτά τα δύο δένδρα δεν μπορούν να υπάρξουν άλλα παρόμοια τάξης 3 με 5 κλειδιά. Οι δύο αυτές διακριτές δομές αποτελούν μία κλάση που συμβολίζεται με BT(3,5).

Στο Σχήμα 8.9 παρουσιάζεται η κλάση BT(3,15) που αποτελείται από 10 διακριτές δομές. Στον Πίνακα 8.1 για κάθε δομή δίνονται οι αντίστοιχα η μέση τιμή προσπελάσεων για επιτυχή αναζήτηση και ο συνολικός καταλαμβανόμενος χώρος. Από τον πίνακα αυτόν προκύπτει ότι:

τα B-δένδρα που είναι βέλτιστα από πλευράς χώρου είναι περίπου βέλτιστα και από πλευράς χρόνου, ενώ τα B-δένδρα που είναι βέλτιστα από πλευράς χρόνου είναι περίπου χειρότερα από πλευράς χώρου.



Σχήμα 8.9: Ομάδα $BT(3,15)$.

Δένδρο	Μέση τιμή προσπελάσεων	Καταλαμβάνο- μενος χώρος
α	3,27	15
β	2,60	9
γ	2,40	12
δ	2,53	10
ε	2,53	10
στ	2,60	9
ζ	2,47	11
η	2,47	11
θ	2,47	11
ι	2,27	11

Πίνακας 8.1: Σύγκριση δομών της κλάσης BT(3,15).

Η διαπίστωση αυτή δικαιολογείται από το γεγονός ότι οι δομές (β) και (στ), που καταλαμβάνουν τον ελάχιστο χώρο των 9 κόμβων, έχουν χρονικό κόστος 2,60 που είναι μόλις 8,3% μεγαλύτερο από την ελάχιστη τιμή των 2,40 προσπελάσεων. Αντίθετα, η δομή (γ), που είναι η καλύτερη από πλευράς χρόνου, καταλαμβάνει 12 κόμβους που είναι 33,3% περισσότερο από την ελάχιστη τιμή των 9 κόμβων. Η διαπίστωση αυτή ανήκει στους Rosenberg και Snyder (1981).

Βέβαια, η μορφή ενός B-δένδρου προσδιορίζεται από τη σειρά άφιξης των κλειδιών και δεν είναι δυνατόν να γίνει επιλογή εκ μέρους του χρήστη. Ωστόσο, το πρακτικό συμπέρασμα είναι ότι με μία άλλη πολιτική διαχείρισης των διασπάσεων/συγχωνεύσεων μπορεί να υπάρξουν διαφοροποιήσεις στην επίδοση της δομής. Στη συνέχεια παρουσιάζεται μία τέτοια μέθοδος.

8.3 B*-δένδρα

Η μέση τιμή του παράγοντα χρησιμοποίησης χώρου (δηλαδή $E[U]=69\%$) ενός B-δένδρου είναι δυνατό να αυξηθεί αν κατά την εισαγωγή εφαρμοσθεί η εξής τεχνική. Αν κάποια εισαγωγή προκαλέσει υπερχειλίση, τότε να μην γίνει διάσπαση (οπότε θα προκύψουν δύο κόμβοι με περιεκτικότητα 50%), αλλά κάποια κλειδιά να αποθηκευθούν σε ένα γειτονικό κόμβο που δεν είναι 100% πλήρης. Αν και οι δύο γειτονικοί κόμβοι είναι πλήρεις, τότε διασπώνται δύο κόμβοι (ο υπερχειλίζων και ένας γειτονικός) και προκύπτουν τρεις.

Με αυτό το τέχνασμα η ελάχιστη τιμή του παράγοντα χρησιμοποίησης χώρου καθίσταται $U_{min}=67\%$, ενώ με αντικατάσταση στον τύπο του Leung προκύπτει μέση τιμή $E[U]=81\%$. Άλλο πλεονέκτημα είναι ότι η αναζήτηση γίνεται ταχύτερα, ενώ το μειονέκτημα είναι το αυξημένο κόστος κατά τις εισαγωγές και τις διαγραφές. Τα B*-δένδρα, που προτάθηκαν επίσης από τους Bayer και McCreight το 1972, ορίζονται κατ' αναλογία προς τα B-δένδρα.

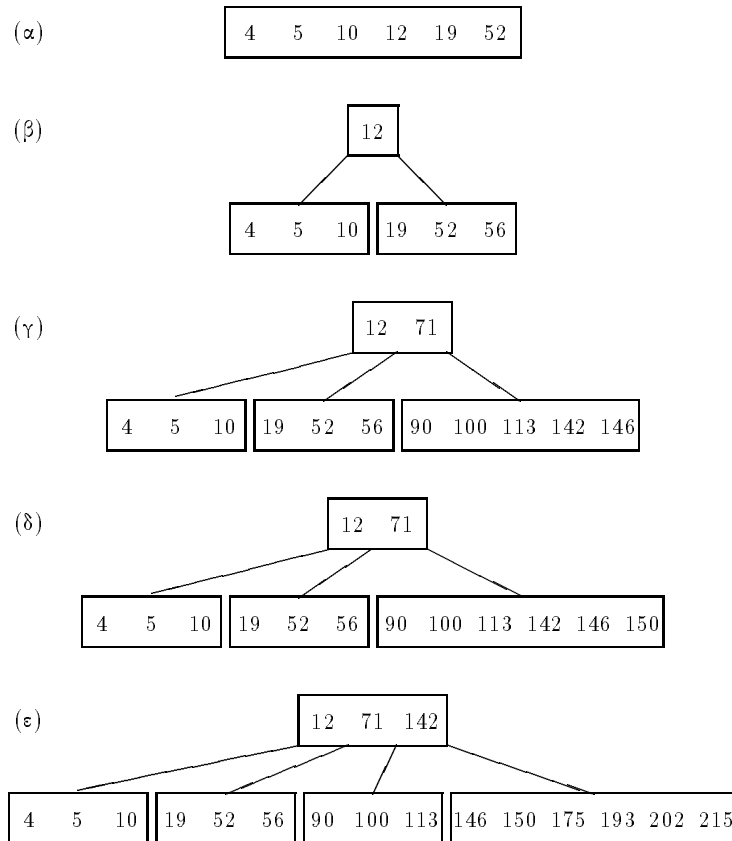
Σύμφωνα με έναν ορισμό το B*-δένδρο τάξης m είναι ένα δένδρο αναζήτησης m δρόμων όπου:

- η ρίζα έχει το ελάχιστο δύο παιδιά και το μέγιστο $2\lfloor \frac{2m-2}{3} \rfloor + 1$ παιδιά,
- οι εσωτερικοί κόμβοι, εκτός της ρίζας, έχουν το ελάχιστο $\lfloor \frac{2m-2}{3} \rfloor + 1$ παιδιά και το μέγιστο m παιδιά,
- ένας εσωτερικός κόμβος με k κλειδιά έχει $k+1$ παιδιά, και
- οι εξωτερικοί κόμβοι βρίσκονται στο ίδιο επίπεδο.

Τα πράγματα φωτίζονται καλύτερα με τα Σχήματα 8.10 και 8.11, όπου παρουσιάζεται αντίστοιχα η αλληλουχία διαδοχικών εισαγωγών σε ένα B-δένδρο και σε ένα B*-δένδρο τάξης 7. Σε ένα B-δένδρο τάξης 7 ο ελάχιστος και ο μέγιστος αριθμός παιδιών της ρίζας είναι 2 και 7 αντίστοιχα, ενώ ο ελάχιστος και ο μέγιστος αριθμός παιδιών των εσωτερικών κόμβων είναι 4 και 7 αντίστοιχα. Σε αντίθεση για ένα B*-δένδρο τάξης 7 ισχύουν τα εξής: ο ελάχιστος και ο μέγιστος αριθμός παιδιών της ρίζας είναι 2 και 9 αντίστοιχα, ενώ ο ελάχιστος και ο μέγιστος αριθμός παιδιών των εσωτερικών κόμβων είναι 5 και 7 αντίστοιχα.

Έστω ότι στο B-δένδρο έχουν ήδη εισαχθεί 6 κλειδιά που στεγάζονται στη ρίζα. Με την εισαγωγή ενός έβδομου κλειδιού η ρίζα διασπάται και προκύπτουν δύο κόμβοι-παιδιά. Η διαδικασία αυτή παρουσιάζεται στο Σχήμα 8.10β. Αντίστοιχα, έστω ότι στο B*-δένδρο έχουν εισαχθεί ήδη 8 κλειδιά και ακολουθεί εισαγωγή του ένατου. Στο Σχήμα 8.11β παρουσιάζεται η διάσπαση της ρίζας και η γένεση δύο κόμβων-παιδιών. Η ρίζα στο B*-δένδρο μπορεί να προσφέρει στέγη σε περισσότερα κλειδιά από την τάξη του δένδρου, ώστε με τη διάσπαση της ρίζας να προκύψουν κόμβοι πλήρεις κατά 67%.

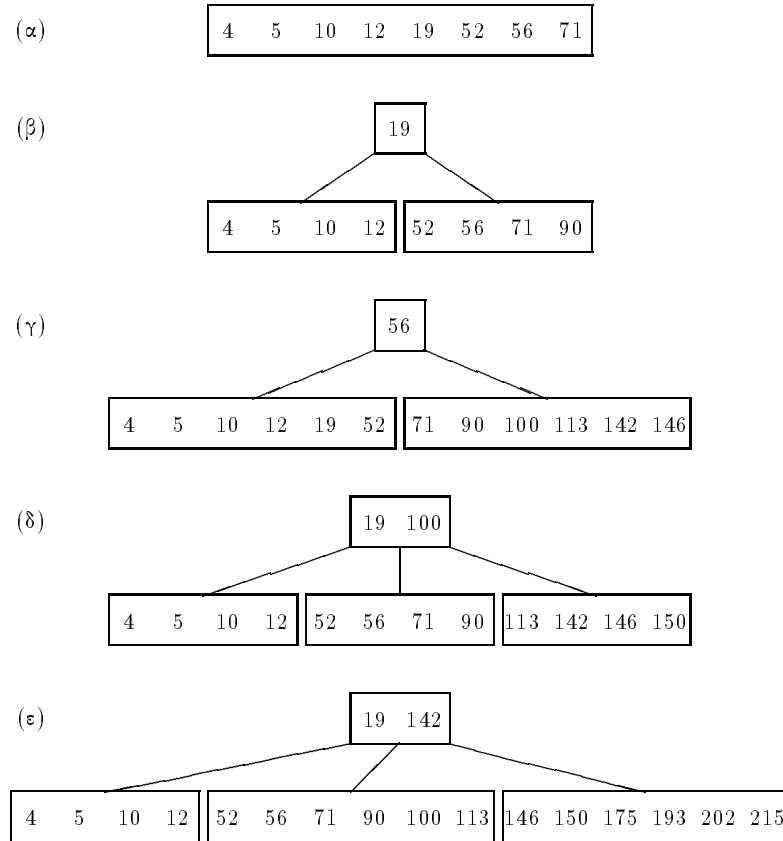
Η διαδικασία των εισαγωγών συνεχίζεται. Τα κλειδιά 100, 113 και 142 κατευθύνονται στο δεξιό παιδί της ρίζας του B*-δένδρου του Σχήματος 8.11β



Σχήμα 8.10: Διαδοχικές εισαγωγές σε B-δένδρο.

και προκαλούν την υπερχειλίση του, όχι όμως και τη διάσπασή του. Στο Σχήμα 8.11γ παρουσιάζεται το B*-δένδρο μετά την εισαγωγή του κλειδιού 146. Είναι φανερό ότι μερικά κλειδιά έχουν περάσει στον αριστερό γειτονικό κόμβο με ανάλογη ρύθμιση των κλειδιών της ρίζας. Το Σχήμα 8.10γ παρουσιάζει αντίστοιχο B-δένδρο που περιέχει τα ίδια κλειδιά. Στην περίπτωση αυτή καταλαμβάνονται 4 κόμβοι έναντι των 3 του B*-δένδρου.

Αν υπάρξει υπερχειλίση σε κόμβο ενός B*-δένδρου και δεν υπάρχει χώρος σε γειτονικό κόμβο, τότε δημιουργείται ένας νέος κόμβος. Ταυτόχρονα τα κλειδιά του κόμβου που υπερχειλίζει και ενός γειτονικού κόμβου (με προτεραιότητα στα αριστερά) αναδιανέμονται πλέον στους τρεις κόμβους. Για παράδειγμα, έστω ότι στα δένδρα των Σχημάτων 8.10γ και 8.11γ εισάγεται



Σχήμα 8.11: Διαδοχικές εισαγωγές σε B*-δένδρο.

το κλειδί 150. Η επιλογή των δύο μεσαίων κλειδιών που θα ανέλθουν στο ανώτερο επίπεδο προκύπτει με την εξής απλή μέθοδο. Τα κλειδιά που υπεισέρχονται στη διαδικασία της αναδιανομής είναι τα κλειδιά των δύο κόμβων, το κλειδί από το ανώτερο επίπεδο και το εισαγόμενο κλειδί, άρα ο συνολικός αριθμός κλειδιών είναι $2m$. Το πρώτο κλειδί που επιλέγεται είναι το $\lceil (2m+1)/3 \rceil$ -οστό (έστω ότι η ποσότητα αυτή συμβολίζεται με a), ενώ το δεύτερο είναι το $\lceil (2m+1-a)/2 \rceil$ -οστό από τα υπόλοιπα $2m-a$. Τα Σχήματα 8.10δ και 8.11δ δείχνουν τη νέα μορφή των δένδρων. Αν και τα δύο δένδρα έχουν από τέσσερις κόμβους, εντούτοις οι δύο κόμβοι του B-δένδρου είναι πλήρεις κατά 50% και ο ένας κατά 100%, ενώ όλοι οι κόμβοι του B*-δένδρου είναι πλήρεις κατά 50%. Η διαδικασία των αφίξεων νέων εγγραφών

συνεχίζεται με τα κλειδιά 15, 16, 17 και 18. Οι νέες μορφές των δένδρων φαίνονται στα Σχήματα 8.10ε και 8.11ε.

Όπως φαίνεται τα B^* -δένδρα είναι πιο χρονοβόρα από τα B -δένδρα κατά την εισαγωγή και κατά τη διαγραφή, όμως κάνουν καλύτερη χρήση του χώρου από τα B -δένδρα, επειδή διακρίνονται από μικρότερο ύψος και λιγότερους κόμβους για τον ίδιο αριθμό κλειδιών. Επιπλέον προσφέρουν καλύτερους χρόνους αναζήτησης, δηλαδή η μέση τιμή των προσπελάσεων για αναζήτηση ενός κλειδιού σε δένδρο τάξης m με n κλειδιά στη χειρότερη περίπτωση είναι:

$$T_{\text{προσ}} \leq 1 + \log_{(2m-2)/3} \frac{n+1}{2}$$

Από τους Eisenbarth et al. έχει αποδειχθεί ότι η πιθανότητα να γίνει μία ή περισσότερες διασπάσεις κατά την εισαγωγή του $(n+1)$ -κλειδιού είναι:

$$\frac{1}{m-1} + O(m^{-2})$$

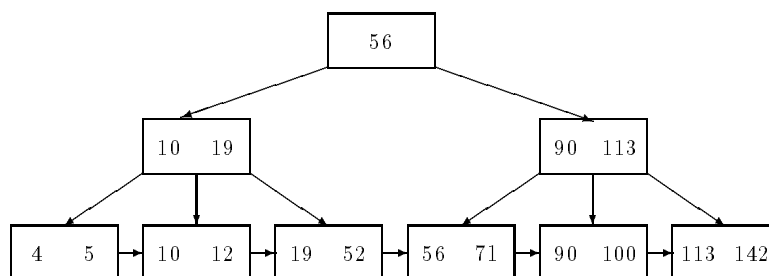
Συγκρίνοντας τον τύπο αυτό με τον αντίστοιχο τύπο από τα B -δένδρα προκύπτει αμέσως ότι στα B^* -δένδρα γίνεται σπανιότερα διάσπαση.

Από την προηγούμενη ανάπτυξη γίνεται φανερό ότι για την υλοποίηση της δομής αυτής πρέπει να θεωρηθεί διαφορετικός τύπος εγγραφής για τη ρίζα. Λιγότερο συχνά συναντάται στη βιβλιογραφία ένας παραλλαγμένος ορισμός του B^* -δένδρου, όπου ο κόμβος της ρίζας δέχεται το ίδιο μέγιστο περιεχόμενο με τους υπόλοιπους κόμβους, ενώ η πολιτική διαχείρισης της υπερχειλίσης είναι κοινή. Η τεχνική της διάσπασης δύο κόμβων σε τρεις μπορεί να γενικευθεί με διάσπαση τριών κόμβων σε τέσσερις, τεσσάρων κόμβων σε πέντε κοκ. Στις περιπτώσεις αυτές η ελάχιστη τιμή του παράγοντα χρησιμοποίησης χώρου θα ισούται αντίστοιχα με 75%, 80% κοκ., ενώ η μέση τιμή προκύπτει από τον τύπο του Leung. Βέβαια είναι ευνόητο ότι το τίμημα έναντι του κέρδους αυτού σε καταλαμβανόμενο χώρο είναι το αυξημένο κόστος στον απαιτούμενο χρόνο για εισαγωγές και διαγραφές.

8.4 B^+ -δένδρα

Η δομή που χρησιμοποιείται περισσότερο σε εμπορικά πακέτα για τη δημιουργία καταλόγων είναι τα B^+ -δένδρα. Η δομή αυτή προτάθηκε από τον Knuth και ορίζεται ως εξής. Σε ένα B^+ -δένδρο βαθμού d :

- η ρίζα έχει το ελάχιστο δύο παιδιά, εκτός αν είναι φύλλο,
- οι εσωτερικοί κόμβοι δεν έχουν περισσότερα από $2d$ κλειδιά και λιγότερο από d κλειδιά. Οι εσωτερικοί κόμβοι περιέχουν μόνο κλειδιά και δείκτες προς το κατώτερο επίπεδο,
- όλοι οι εξωτερικοί κόμβοι βρίσκονται στο ίδιο επίπεδο. Αν το B^+ -δένδρο χρησιμοποιείται ως κύριος κατάλογος, τότε οι εξωτερικοί κόμβοι περιέχουν εγγραφές, ενώ αν χρησιμοποιείται ως δευτερεύων κατάλογος, τότε οι εξωτερικοί κόμβοι περιέχουν κλειδιά και δείκτες προς εγγραφές, και
- ένας εσωτερικός κόμβος με k κλειδιά έχει $k+1$ παιδιά.



Σχήμα 8.12: Κατάλογος με δομή B^+ -δένδρου.

Από τον ορισμό φαίνεται ότι (όπως το B -δένδρο) το B^+ -δένδρο δεν είναι απαραίτητα δυαδικό δένδρο. Για παράδειγμα, είναι δυνατόν από κάθε εσωτερικό κόμβο του B^+ -δένδρου να δεικτοδοτούνται 100 κόμβοι παιδιά. Σε αντίθεση προς το B -δένδρο, το B^+ -δένδρο είναι ετερογενές. Στο Σχήμα 8.12 παρουσιάζεται ένα B^+ -δένδρο, όπου οι αριθμοί στα φύλλα δηλώνουν τα κλειδιά ολόκληρων εγγραφών, ενώ οι αριθμοί στους εσωτερικούς κόμβους δηλώνουν απλά κλειδιά. Επίσης τα φύλλα περιέχουν δείκτες προς το επόμενο φύλλο για διευκόλυνση της σειριακής επεξεργασίας. Φαίνεται, λοιπόν, ότι η δομή υποδιαιρείται σε δύο λογικά μέρη, όπου το ανώτερο εξυπηρετεί το κατώτερο ως καταλόγος. Πραγματικά σε μία υλοποίηση θα πρέπει να είναι διαφορετικός ο τύπος των εσωτερικών από τον τύπο των φύλλων. Στη συνέχεια θα φανεί ότι αυτός ο κατάλογος λειτουργεί ως ένα απλό B -δένδρο όπως αναπτύχθηκε προηγουμένως. Κάλλιστα, επίσης, θα μπορούσε ο κατάλογος αυτός να είναι ένα B^* -δένδρο.

Στο σημείο αυτό ανοίγει μία παρένθεση. Έχει χρησιμοποιηθεί ήδη ο όρος σελίδα για να δηλωθεί το ελάχιστο φυσικό μέγεθος δεδομένων που μεταφέρεται μεταξύ κύριας και δευτερεύουσας μνήμης. Ωστόσο, η μεταφορά δεδομένων γίνεται λογικά κατά κάδους. Στην πιο απλή μορφή μπορεί να θεωρηθεί ότι ένας κάδος ταυτίζεται με μία σελίδα. Στη συνέχεια λοιπόν, για λόγους ευκολίας ο όρος κάδος θα χρησιμοποιηθεί ισοδύναμα με την έννοια του κόμβου.

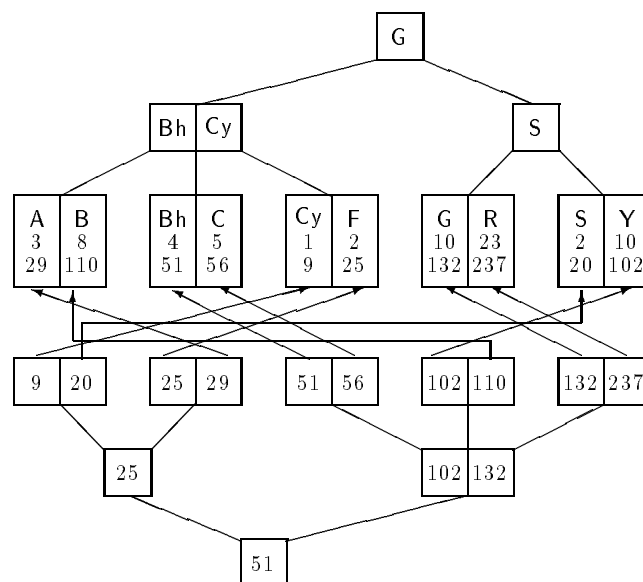
Η βασική διαφορά μεταξύ των B- και B*-δένδρων αφ' ενός και του B⁺-δένδρου αφ' ετέρου είναι ότι στο B⁺-δένδρο τα δεδομένα βρίσκονται μόνο στα φύλλα. Σύμφωνα με τον ορισμό, οι εσωτερικοί κόμβοι έχουν από d ως $2d$ κλειδιά, ενώ τα φύλλα (που ταυτίζονται με τους κάδους) περιέχουν έναν αριθμό εγγραφών που λέγεται **παράγοντας καδοποίησης** (bucket factor, $Bkfr$). Για παράδειγμα, σε ένα εσωτερικό και σε ένα εξωτερικό κόμβο μπορεί να έχουν αποθηκευθεί 200 κλειδιά και 10 εγγραφές, αντίστοιχα. Επειδή οι εσωτερικοί κόμβοι περιέχουν μόνο κλειδιά και διευθύνσεις, ο λόγος διακλάδωσης είναι πολύ μεγάλος. Έτσι αυτά τα δένδρα είναι ρηχά και εξαιρετικά πλατιά. Αυτός είναι ο βασικός λόγος που τα B⁺-δένδρα είναι τόσο αποτελεσματικά. Μάλιστα τα B⁺-δένδρα είναι η πιο κατάλληλη δομή για ερωτήσεις διαστήματος (ενώ όπως θα φανεί σε επόμενα κεφάλαια η πιο κατάλληλη δομή για απλές ερωτήσεις στηρίζεται στον κατακερματισμό).

Η δομή του B⁺-δένδρου που εξηγήθηκε προηγουμένως είναι μία περίπτωση πρωτεύοντος καταλόγου, δηλαδή καταλόγου με βάση το πρωτεύον κλειδί. Στην περίπτωση αυτή, ο πρωτεύων κατάλογος καθορίζει τον τρόπο αποθήκευσης των εγγραφών. Αν οι εγγραφές είναι αποθηκευμένες σε γειτονικούς κάδους κατά απόλυτη τάξη του κλειδιού τους, τότε ο κατάλογος λέγεται **συμπαγής** (clustered). Έτσι, στο ανώτερο επίπεδο από τα φύλλα ο κατάλογος περιέχει μόνο την τιμή του μικρότερου (ή μεγαλύτερου) κλειδιού κάθε κάδου. Για το λόγο αυτό, ο κατάλογος αυτός λέγεται και **αραιός** (sparse). Για παράδειγμα, ο κατάλογος ενός αρχείου ISAM είναι ένας αραιός κατάλογος.

Ωστόσο, το B⁺-δένδρο μπορεί να χρησιμοποιηθεί και ως δευτεύων κατάλογος. Στο Σχήμα 8.13 δίνεται ένα παράδειγμα, όπου οι εγγραφές αναφέρονται σε στοιχεία βαλχανικών κρατών (που δίνονται στον Πίνακα 8.2). Πιο συγκεκριμένα, πρωτεύον κλειδί είναι το όνομα του κράτους, ενώ δευτερεύοντα κλειδιά είναι ο πληθυσμός (σε εκατομμύρια κατοίκους) και η έκταση (σε χιλιάδες τετραγωνικών χιλιομέτρων). Έτσι στο επάνω μέρος του σχήματος φαίνεται η δομή του πρωτεύοντος καταλόγου, ενώ στο κάτω

Χώρα	Πληθυσμός	Έκταση
Αλβανία (A)	3	29
Βοσνία-Ερζεγοβίνη (Bh)	4	51
Βουλγαρία (B)	8	110
Γιουγκοσλαβία (Y)	10	102
Ελλάδα (G)	10	132
Κροατία (C)	5	56
Κύπρος (Cy)	1	9
ΠΓΔΜ (F)	2	25
Ρουμανία (R)	23	237
Σλοβενία (S)	2	20

Πίνακας 8.2: Πληθυσμός και έκταση των Βαλκανικών κρατών.

Σχήμα 8.13: B⁺-δένδρο ως πρωτεύων και δευτερεύων κατάλογος.

μέρος του σχήματος διακρίνεται ο δευτερεύων κατάλογος που είναι οργανωμένος με βάση το πεδίο της έκτασης. Στο τελευταίο επίπεδο του καταλόγου αυτού υπάρχουν αποθηκευμένα ζεύγη κλειδιών-δεικτών, όπου οι δείκτες σηματοδοτούν προς τις αντίστοιχες εγγραφές του πρωτεύοντος καταλόγου. Ο κατάλογος αυτός δεν είναι αραιός, αλλά αντίθετα λέγεται πυκνός (dense).

Συχνά για το ίδιο αρχείο υπάρχουν πολλοί δευτερεύοντες κατάλογοι, που αντιστοιχούν σε διαφορετικά πεδία της εγγραφής. Έτσι διευκολύνεται η αναζήτηση με βάση είτε το πρωτεύον, είτε το δευτερεύον κλειδί. Όμως αν μία εγγραφή έχει πολλά πεδία, δεν είναι απαραίτητο να δημιουργηθεί ένας κατάλογος για κάθε πεδίο, γιατί έτσι:

- η ίδια πληροφορία θα επαναλαμβάνόταν δύο φορές και θα χρειαζόταν πολλαπλάσιος χώρος από το χώρο των κυρίως δεδομένων, και
- το κόστος ενημέρωσης των καταλόγων θα ήταν ιδιαίτερα μεγάλο ακόμη και για μία απλή εισαγωγή νέας εγγραφής.

Ευθύνη του σχεδιαστή των αρχείων είναι να αποφασίσει το πόσους και ποιούς καταλόγους πρέπει να δημιουργήσει, γιατί αν γίνει ερώτηση με βάση δευτερεύον κλειδί για το οποίο δεν υπάρχει αντίστοιχος κατάλογος, τότε η αναζήτηση ταυτίζεται με την αναζήτηση σε αρχείο σωρού. Ας σημειωθεί, επίσης, ότι τα αρχεία ISAM δεν μπορούν να χρησιμεύσουν ως δευτερεύοντες κατάλογοι.

Η αναζήτηση σε B^+ -δένδρα γίνεται με την εξής μέθοδο. Ξεκινώντας από τη ρίζα, οι δείκτες προς το κατώτερο επίπεδο ονομάζονται από 0 ως k και τα κλειδιά από 1 ως k . Αν το κλειδί που αναζητείται είναι μικρότερο από το πρώτο κλειδί της ρίζας, τότε λαμβάνεται ο δείκτης υπ' αριθμό 0. Γενικά, αν το αναζητούμενο κλειδί είναι μεγαλύτερο από το i -οστό κλειδί και μικρότερο από το $(i+1)$ -οστό κλειδί, τότε ακολουθείται ο i -οστός δείκτης. Η διαδικασία αυτή εκτελείται για όλα τα επίπεδα του δένδρου. Έτσι η αναζήτηση φθάνει στα φύλλα, όπου εντοπίζεται η σχετική εγγραφή αν το B^+ -δένδρο είναι πρωτεύων κατάλογος, ενώ αν το B^+ -δένδρο είναι δευτερεύων κατάλογος τότε ακολουθείται ο σχετικός δείκτης προς την αντίστοιχη εγγραφή.

8.4.1 Προσπέλαση εγγραφής

Έστω ότι το B^+ -δένδρο χρησιμοποιείται ως πρωτεύων κατάλογος. Μπορεί εύκολα να αποδειχθεί, ότι όταν ανοίγει το αρχείο, τότε όλοι οι κόμβοι εκτός των κόμβων των δύο τελευταίων επιπέδων αποθηκεύονται στην κύρια μνήμη. Συνεπώς, το κόστος διάσχισης των επιπέδων αυτών είναι αμελητέο. Απαιτείται μία προσπέλαση για τον κόμβο που βρίσκεται επάνω από το κατάλληλο φύλλο και άλλη μία προσπέλαση για το φύλλο. Άρα το σχετικό κόστος είναι:

$$T_{\text{πρσ}} = (s + r + btt) + (s + r + dtt)$$

Έστω, τώρα, ότι το B^+ -δένδρο χρησιμοποιείται ως δευτερεύων κατάλογος. Είναι εύλογο να υποτεθεί ότι στην κύρια μνήμη χωρούν όλα τα επίπεδα του δένδρου πλην του τελευταίου που περιέχει τους δείκτες προς τους κάδους. Έτσι, και στην περίπτωση αυτή το χρονικό κόστος δίνεται από την προηγούμενη έκφραση. Σε κάθε περίπτωση, το σημαντικό συμπέρασμα είναι ότι η αναζήτηση σε B^+ -δένδρο απαιτεί δύο προσπελάσεις στο δίσκο.

8.4.2 Εξαντλητική ανάγνωση αρχείου

Κάθε κόμβος περιέχει το ελάχιστο και το μέγιστο d και $2d$ κλειδιά αντίστοιχα, όμως έχει αποδειχθεί ότι κατά μέσο όρο περιέχει $2d \times \ln 2$ κλειδιά. Συνεπώς, ο αριθμός των κάδων που απαιτούνται για την αποθήκευση του B^+ -δένδρου είναι:

$$\frac{n}{Bkfr \times \ln 2} = bk$$

όπου n είναι ο αριθμός των εγγραφών του αρχείου. Ο απαιτούμενος χρόνος για την εξαντλητική ανάγνωση ενός B^+ -δένδρου που χρησιμεύει ως πρωτεύων κατάλογος είναι:

$$T_{\text{εξαν}} (\text{πρωτεύων}) = bk \times (s + r + dtt) = \frac{n \times (s + r + dtt)}{Bkfr \times \ln 2}$$

Αυτός ο τύπος προϋποθέτει ότι σε κάθε φύλλο υπάρχει ένας δείκτης προς το επόμενο φύλλο. Αν αυτή η συνθήκη δεν ισχύει, τότε εναλλακτικά μπορεί να εφαρμοσθεί μία ενδοδιατεταγμένη διάσχιση με αμελητέο επιπλέον χρονικό κόστος. Για παράδειγμα, αν υποτεθεί ότι ο βαθμός του δένδρου είναι 100, τότε ο λόγος διακλάδωσης είναι 140. Άρα, για κάθε 140 φύλλα πρέπει να γίνει μία ακόμη προσπέλαση στον κόμβο πατέρα.

Πρέπει να σημειωθεί ότι το χρονικό κόστος που προκύπτει από τη σχέση αυτή είναι μεγαλύτερο από το κόστος εξαντλητικής ανάγνωσης ενός σειριακού ταξινομημένου αρχείου, επειδή οι διαδοχικές εισαγωγές και διαγραφές συντελούν στη φυσική απομάκρυνση λογικά διπλανών κόμβων. Έτσι, απαιτούνται συνεχώς νέες προσπελάσεις με κόστος εντοπισμού του κατάλληλου κυλίνδρου.

Ο απαιτούμενος χρόνος για την εξαντλητική ανάγνωση ενός B^+ -δένδρου που χρησιμεύει ως δευτερεύων κατάλογος είναι:

$$T_{\text{εξαν}} (\text{δευτερεύων}) = n \times (s + r + dtt) + n \times \frac{s + r + btt}{2d \times \ln 2}$$

Ο πρώτος όρος αναφέρεται στο κόστος ανάγνωσης των εγγραφών μία προς μία, ενώ ο δεύτερος αναφέρεται στο κόστος ανάγνωσης του τελευταίου επιπέδου του B^+ -δένδρου. Ο δεύτερος όρος είναι σημαντικά μικρότερος και μπορεί να παραλειφθεί.

8.4.3 Προσπέλαση επόμενης εγγραφής

Έστω και πάλι ότι το B^+ -δένδρο χρησιμοποιείται ως πρωτεύων κατάλογος και ότι υπάρχουν δείκτες από το ένα φύλλο προς το άλλο. Το χρονικό κόστος για την ανάκτηση της επόμενης εγγραφής είναι:

$$T_{\text{επομ}} (\text{πρωτεύων}) = \frac{s + r + dtt}{Bkfr \times \ln 2}$$

Η σχέση αυτή εξηγείται με βάση το γεγονός ότι $1/(Bkfr \times \ln 2)$ είναι η πιθανότητα η επόμενη εγγραφή να βρίσκεται στον ίδιο κόμβο με την προηγούμενη.

Αν το B^+ -δένδρο χρησιμοποιείται ως δευτερεύων κατάλογος, τότε απαιτείται τουλάχιστο μία προσπέλαση κόμβου για την εύρεση της επόμενης εγγραφής. Επιπλέον μπορεί να απαιτηθεί μία προσπέλαση στο επόμενο φύλλο για την εύρεση της διεύθυνσης αυτού του κόμβου. Άρα, τελικά, το χρονικό κόστος για την ανάκτηση της επόμενης εγγραφής είναι:

$$T_{\text{επομ}} (\text{δευτερεύων}) = (s + r + dtt) + \frac{s + r + btt}{2d \times \ln 2}$$

Ο πρώτος όρος δίνει το κόστος για την εύρεση του επόμενου κλάδου όταν η διεύθυνσή του είναι γνωστή. Ο δεύτερος όρος προκύπτει από το γεγονός ότι είναι πιθανό αυτή η διεύθυνση να μην είναι γνωστή. Στην περίπτωση αυτή, γνωστή είναι μόνο η διεύθυνση της σελίδας, όπου είναι αποθηκευμένη η διεύθυνση της επόμενης εγγραφής. Και πάλι, ο δεύτερος όρος είναι λιγότερο σημαντικός από τον πρώτο και η σχέση μπορεί να απλοποιηθεί.

8.4.4 Εισαγωγή εγγραφής

Οι εισαγωγές στο B^+ -δένδρο γίνονται στο τελευταίο επίπεδο. Μία εισαγωγή πάντα αρχίζει με αναζήτηση του κλειδιού που πρόκειται να εισαχθεί, ώστε να εντοπισθεί ο κατάλληλος κόμβος. Αν το κλειδί βρίσκεται ήδη στο

αρχείο αποφεύγεται ένα λάθος. Αν υπάρχει διαθέσιμος χώρος μέσα στον κόμβο, τότε το κλειδί (ή η εγγραφή) αποθηκεύεται. Στην απλούστερη, λοιπόν, περίπτωση το κόστος μίας εισαγωγής είναι:

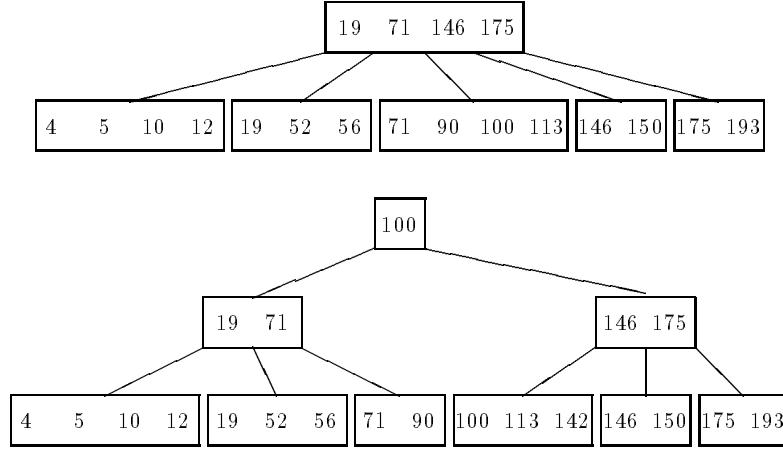
$$T_{\text{εισ}} = T_{\text{προσ}} + 2r$$

Όμως είναι δυνατόν να υπάρξει διάσπαση κόμβου λόγω υπερχειλίσης, και αυτή η διαδικασία να συνεχισθεί μέχρι την κορυφή. Για παράδειγμα, έστω ότι το B^+ -δένδρο χρησιμοποιείται ως πρωτεύων κατάλογος και ότι σε έναν κόμβο συγκεντρώνονται $Bkfr+1$ εγγραφές. Οι πρώτες μισές και οι δεύτερες μισές εγγραφές αυτού του συνόλου αποθηκεύονται στον παλιό και σε νέο κόμβο, αντίστοιχα. Δηλαδή, αν $Bkfr=5$ τότε κάθε κόμβος δέχεται από τρεις εγγραφές. Αν $Bkfr=10$, τότε ο παλιός και ο νέος κόμβος δέχονται 5 και 6 εγγραφές, αντίστοιχα. Στην περίπτωση αυτή είναι φανερό ότι οι δύο κόμβοι είναι πλήρεις κατά 50%. Αυτός είναι ένας λόγος που το αρχείο αυτό καταλαμβάνει περισσότερο χώρο από ένα αρχείο σωρού.

Αν το B^+ -δένδρο χρησιμοποιείται ως δευτερεύων κατάλογος, τότε κάθε φύλλο πρέπει να περιέχει από d ως $2d$ κλειδιά. Έτσι, στον παλιό και στο νέο κόμβο αποθηκεύονται d και $d+1$ κλειδιά, αντίστοιχα. Και στις δύο περιπτώσεις, η τιμή του μικρότερου κλειδιού του νέου κόμβου καθώς και η διεύθυνση του νέου κόμβου πρέπει να ανέλθουν προς τον κόμβο πατέρα. Αν αυτό το ζεύγος που αντλείται προς τα επάνω δεν χωρά στον πατέρα κόμβο, τότε πρέπει να γίνει νέα διάσπαση. Ωστόσο, στο σημείο αυτό υπάρχει η εξής μικρή διαφορά. Όταν στους εσωτερικούς κόμβους υπάρξει διάσπαση, τότε τα πρώτα d κλειδιά αποθηκεύονται στον παλιό κόμβο, τα τελευταία d (και όχι $d+1$) κλειδιά αποθηκεύονται στο νέο κόμβο, ενώ το μεσαίο κλειδί αντλείται προς το ανώτερο επίπεδο. Δηλαδή, η διαφορά έγκειται στο γεγονός ότι το ανερχόμενο κλειδί αποθηκεύεται και σε ένα νέο κόμβο του τελευταίου επιπέδου αλλά δεν συμβαίνει το ίδιο και στα ανώτερα επίπεδα. Η διαδικασία αυτή μπορεί να συνεχισθεί μέχρι τη ρίζα. Στο Σχήμα 8.14 δίνεται ένα παράδειγμα εισαγωγής του κλειδιού 142 στο B^+ -δένδρο που είναι βαθμού 2.

Το κόστος εισαγωγής είναι συνάρτηση των διασπάσεων που θα συμβούν. Ο Baeza-Yates (1989) επεκτείνοντας την ανάλυση κρασπέδου των Eisenbarth et al. κατέληξε ότι η πιθανότητα να συμβούν περισσότερες από μία διασπάσεις κατά την εισαγωγή σε ένα B^+ -δένδρο είναι:

$$\frac{1}{Bkfr \times \ln 2} + O(Bkfr^{-2})$$



Σχήμα 8.14: Εισαγωγή σε B^+ -δένδρο με $d=2$ και $Bkfr=4$.

Στην περίπτωση μας θεωρείται προσεγγιστικά ότι είναι ισοπίθανο σε ένα φύλλο να είναι αποθηκευμένες $Bkfr/2, \dots, Bkfr$ εγγραφές. Η πιθανότητα μία νέα εγγραφή να μην χωρά σε κάποιο φύλλο είναι $\frac{1}{Bkfr/2} = \frac{2}{Bkfr}$. Άρα, η πιθανότητα μία νέα εγγραφή να χωρά στο φύλλο είναι $1 - \frac{2}{Bkfr}$. Με το ίδιο σκεπτικό η πιθανότητα να διασπασθεί ο πατρικός κόμβος είναι $1/d$. Συνεπώς, θεωρώντας ότι όλα τα επίπεδα του δένδρου πλην των δύο τελευταίων είναι αποθηκευμένα στην κύρια μνήμη, τελικά προκύπτει ότι το κόστος εισαγωγής σε B^+ -δένδρο είναι:

$$T_{\text{εισ}} = T_{\text{προσ}} + 2r + \frac{2}{Bkfr} \times \left((s + r + dtt) + (s + r + btt) + \frac{s + r + btt}{d} \right)$$

Επειδή ο βαθμός του δένδρου είναι μεγάλος (πχ. $d=100$), μπορεί η σχέση να απλοποιηθεί:

$$T_{\text{εισ}} = \left(1 + \frac{2}{Bkfr} \right) \times T_{\text{προσ}} + 2r$$

Αν $Bkfr=10$ και το μέγεθος της σελίδας είναι 2400 bytes, τότε προκύπτει ότι ο απαιτούμενος χρόνος για μία εισαγωγή είναι:

$$T_{\text{εισ}} = \left(1 + \frac{2}{5} \right) \times 25,1 + 16,6 = 77 \text{ ms}$$

Από αυτά τα 77 ms, τα 10 περίπου οφείλονται στη διάσπαση.

Αν το B^+ -δένδρο χρησιμοποιείται ως δευτερεύων κατάλογος, τότε πρέπει ο όρος $2/Bkfr$ να αντικατασταθεί με τον όρο $1/d$. Άρα:

$$T_{\epsilon\sigma}(\text{δευτερεύων}) = \left(1 + \frac{1}{d}\right) \times (s + r + btt) + 2r$$

Επειδή ο λόγος διακλάδωσης ενός B^+ -δένδρου, που χρησιμοποιείται ως δευτερεύων κατάλογος, είναι μεγάλος και η πιθανότητα διάσπασης είναι μικρή ο τύπος αυτός απλοποιείται σε:

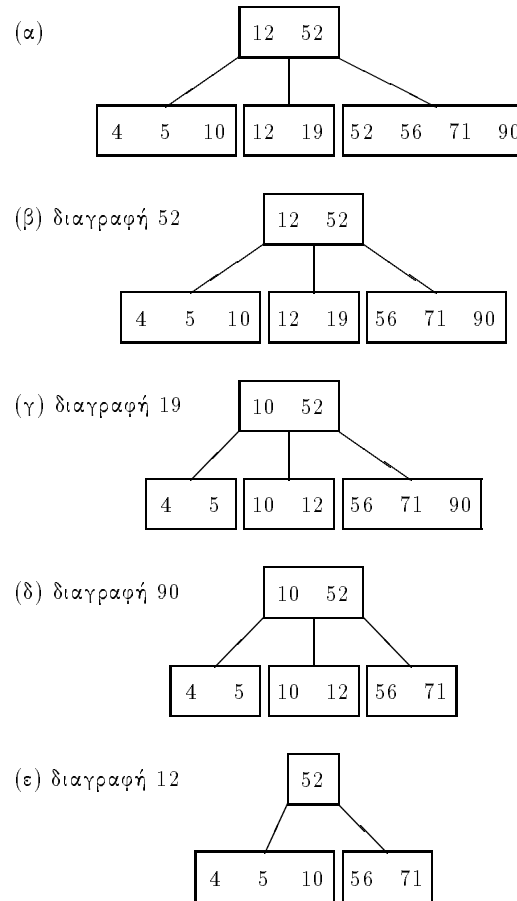
$$T_{\epsilon\sigma}(\text{δευτερεύων}) = s + 3r + btt$$

Όταν γίνεται μία εισαγωγή στον πρωτεύοντα κατάλογο, πρέπει να ενημερωθούν όλοι οι δευτερεύοντες κατάλογοι. Η ενημέρωση των δευτερεύοντων καταλόγων είναι προβληματική, επειδή κατά τη διάσπαση ενός κόμβου πολλές εγγραφές αλλάζουν διεύθυνση. Για το λόγο αυτό, πολλά συστήματα διαχείρισης αρχείων δεν κρατούν μαζί με το δευτερεύον κλειδί την αντίστοιχη διεύθυνση αλλά την τιμή του πρωτεύοντος κλειδιού. Το πλεονέκτημα των συστημάτων αυτών είναι ότι δεν ενημερώνονται για κάθε εισαγωγή όλοι οι δευτερεύοντες κατάλογοι. Από την άλλη πλευρά, το μειονέκτημα είναι ότι για την εύρεση μίας εγγραφής με βάση το δευτερεύον κλειδί πρέπει να γίνει προσπέλαση και στους δύο καταλόγους.

8.4.5 Διαγραφή εγγραφής

Η διαχείριση των διαγραφών σε ένα B^+ -δένδρο είναι δύσκολο πρόβλημα. Σύμφωνα με τον ορισμό δεν πρέπει η περιεκτικότητα των κόμβων του B^+ -δένδρου να είναι κατώτερη του 50%. Άρα, όταν λόγω διαγραφής ένας κόμβος μείνει με λιγότερο από d κλειδιά ή $Bkfr/2$ εγγραφές, τότε πρέπει αυτός ο κόμβος να συγχωνευθεί. Υπάρχουν πολλές περιπτώσεις διαγραφών.

Η πρώτη και απλούστερη περίπτωση είναι όταν μετά τη διαγραφή κάποιος κόμβος δεν έχει λιγότερες εγγραφές από $Bkfr/2$ εγγραφές ή d κλειδιά. Αν από τα φύλλα διαγραφεί κάποια εγγραφή με κλειδί που είναι αποθηκευμένο σε κάποιο ανώτερο επίπεδο ως τιμή σύγκρισης, τότε δεν δημιουργείται πρόβλημα, γιατί και πάλι η αναζήτηση διαμέσου των επιπέδων του δένδρου είναι σωστή. Στο Σχήμα 8.15β αυτή η περίπτωση αντιστοιχεί στη διαγραφή της εγγραφής με κλειδί 52.

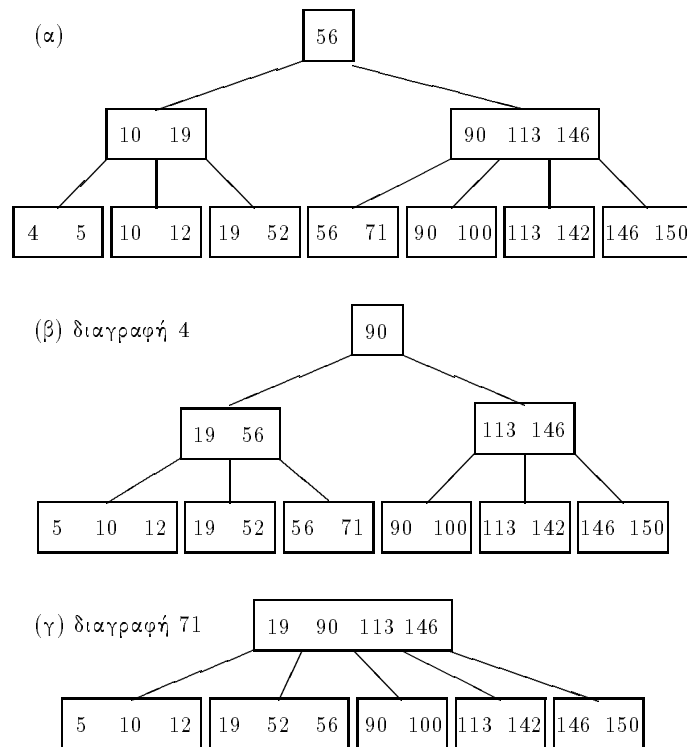


Σχήμα 8.15: Διαγραφές σε B^+ -δένδρο με $d=2$ και $Bkfr=4$.

Αν ένα φύλλο απομείνει με λιγότερες από $Bkfr/2$ εγγραφές, τότε ελέγχεται αν ένας γειτονικός κόμβος (με προτεραιότητα εξετάζεται ο αριστερός) έχουν περισσότερο από αυτό το όριο. Αν κάποιος, λοιπόν, αδελφός κόμβος έχει περισσότερο από το όριο, τότε οι εγγραφές των δύο κόμβων μοιράζονται εξίσου μεταξύ τους και ταυτόχρονα ενημερώνεται ο πατέρας κόμβος. Αυτή η διαδικασία παρουσιάζεται στο Σχήμα 8.15γ με τη διαγραφή της εγγραφής 19. Κατόπιν έστω ότι διαγράφεται η εγγραφή 90. Αυτή η περίπτωση είναι παρόμοια με την περίπτωση της διαγραφής της εγγραφής 52. Η νέα μορφή του B^+ -δένδρου φαίνεται και πάλι στο Σχήμα 8.15δ.

Έστω τώρα ότι πρέπει να διαγραφεί η εγγραφή 12. Είναι φανερό ότι κανείς από τους γειτονικούς αδελφούς κόμβους δεν μπορεί να συνεισφέρει. Επιλέγεται ο αριστερός κόμβος για συγχώνευση. Οι εγγραφές των δύο κόμβων επανα-αποθηκεύονται στον έναν κόμβο, ενώ ο άλλος αποδίδεται στο σύστημα. Ταυτόχρονα, ενημερώνεται και ο πατέρας κόμβος, όπου διαγράφεται ένα κλειδί και ο αντίστοιχος δείκτης. Αυτή η διαδικασία παρουσιάζεται στο Σχήμα 8.15δ.

Μέχρι στιγμής δεν εξετάστηκαν οι εσωτερικοί κόμβοι. Πράγματι, είναι απίθανο κάποιος εσωτερικός κόμβος να καταλήξει με λιγότερο από d κλειδιά, αφού όπως είναι γνωστό κατά μέσο όρο οι εσωτερικοί κόμβοι έχουν $2d \times \ln 2$ κλειδιά. Αν $d=100$, τότε $2d \times \ln 2 = 140$ και συνεπώς ο κόμβος αυτός πρέπει να χάσει 40 κλειδιά. Ωστόσο, για λόγους πληρότητας πρέπει να αναφερθούν τα εξής.



Σχήμα 8.16: Διαγραφές σε B^+ -δένδρο με $d=2$ και $Bkfr=4$.

Έστω ότι ένας εσωτερικός κόμβος μένει με λιγότερο από d κλειδιά. Ελέγχονται οι αδελφοί κόμβοι (με προτεραιότητα στον αριστερό) αν περιέχουν περισσότερα κλειδιά από το όριο. Αν κάποιος αδελφός έχει περισσότερα κλειδιά από το όριο, τότε τα κλειδιά του κόμβου αυτού, το κλειδί του πατέρα κόμβου που χωρίζει τους δύο αδελφούς και τα κλειδιά του υπ' όψη κόμβου αναδιανέμονται στους δύο κόμβους. Το μεσαίο κλειδί από αυτό το σύνολο κλειδιών ανέρχεται στον πατέρα κόμβο. Αν και οι δύο αδελφοί κόμβοι έχουν ακριβώς d κλειδιά, τότε πρέπει να γίνει συγχώνευση (κατά προτεραιότητα με τον αριστερό). Τα κλειδιά αυτά επανα-αποθηκεύονται στον ένα κόμβο, ο άλλος κόμβος αποδίδεται στο σύστημα, ενώ το αντίστοιχο κλειδί και ο σχετικός δείκτης διαγράφονται από τον πατέρα κόμβο. Αν και ο πατέρας κόμβος με τη σειρά του απομείνει με λιγότερο από d κλειδιά, τότε το πρόβλημα μεταφέρεται κατά ένα επίπεδο προς τη ρίζα. Στο Σχήμα 8.16 παρουσιάζεται ένα παράδειγμα διαγραφής σε B^+ -δένδρο που καταλήγει σε συγχώνευση εσωτερικών κόμβων.

Ακολουθεί η ανάλυση για τη διαγραφή σε B^+ -δένδρο που χρησιμοποιείται ως πρωτεύων κατάλογος. Το κόστος διαγραφής της απλής (και πιο συνηθισμένης περίπτωσης) είναι:

$$T_{\text{διαγ}} (\text{πρωτεύων}) = T_{\text{προσ}} + 2r$$

Στη δεύτερη περίπτωση γίνεται αναδιανομή των εγγραφών των γειτονικών κόμβων και του πατέρα κόμβου. Στην περίπτωση αυτή απαιτείται επιπλέον προσπέλαση των δύο αδελφών, ενώ ο πατέρας κόμβος έχει ήδη προσπελασθεί, και επανεγγραφή των τριών κόμβων συνολικά. Η πιθανότητα το φύλλο να είναι γεμάτο κατά το ήμισυ είναι $2/Bkfr$. Εύκολα προκύπτει ότι το σχετικό χρονικό κόστος είναι:

$$T_{\text{διαγ}} (\text{πρωτεύων}) = T_{\text{προσ}} + 4 \times (s + r + dtt) + (s + r + btt)$$

Αν και οι δύο αδελφοί κόμβοι περιέχουν ακριβώς d κλειδιά, τότε το σχετικό κόστος είναι:

$$T_{\text{διαγ}} (\text{πρωτεύων}) = \left(1 + \frac{4}{Bkfr}\right) \times T_{\text{προσ}} + (s + r + btt)$$

Αν το B^+ -δένδρο χρησιμοποιείται ως δευτερεύων κατάλογος, τότε ο όρος $2/Bkfr$ αντικαθίσταται από τον όρο $1/d$. Το κόστος διαγραφής είναι:

$$T_{\text{διαγ}} (\text{δευτερεύων}) = \left(1 + \frac{3}{d}\right) \times (s + r + btt) + 2r$$

Επειδή το d είναι αρκετά μεγάλο, ο τύπος μπορεί να απλοποιηθεί:

$$T_{\delta\alpha\gamma} \text{ (δευτερεύων)} = s + 3r + btt$$

8.5 Άλλες παραλλαγές των B-δένδρων

Από τα B-δένδρα έχει προέλθει μία μεγάλη ομάδα ενδιαφέρουσων παραλλαγών. Μεταξύ των άλλων στη βιβλιογραφία αναφέρονται οι εξής παραλλαγές: τα **B-δένδρα με κλειδιά μεταβλητού μήκους** (B-trees with variable length entries), τα B-δένδρα μικρής τάξης, τα **Δυαδικά B-δένδρα** (Binary B-trees), τα **Συμμετρικά Δυαδικά B-δένδρα** (Symmetric Binary B-trees) και τα **Προθεματικά B⁺-δένδρα** (Prefix B⁺-trees). Αναφορά-κλειδί για την οικογένεια αυτή των παραλλαγών είναι η επισκόπηση του Comer (1979), όπου το B-δένδρο χαρακτηρίζεται ως 'πανταχού παρόν'. Στη συνέχεια θα γίνει εκτενέστερη αναφορά στα Προθεματικά B⁺-δένδρα, ενώ για τις υπόλοιπες δομές δίνονται μόνο λίγα κατατοπιστικά στοιχεία.

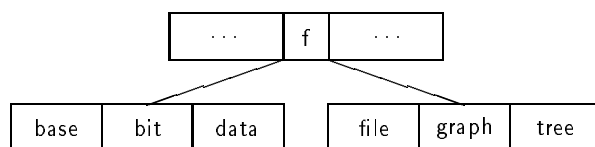
Αν τα κλειδιά είναι μεταβλητού μήκους (πχ. συμβολοσειρές), τότε είναι δυνατόν για δεδομένη ομάδα κλειδιών να κατασκευασθεί ένα B-δένδρο με περιεκτικότητα κόμβων τουλάχιστον 50% (McCreigh 1977). Δηλαδή, στο δένδρο αυτό δεν έχει σημασία ο αριθμός των κλειδιών που περιέχει (δεν χαρακτηρίζεται από κάποια τάξη), αλλά το συνολικό μήκος των κλειδιών. Έχουν προταθεί αρκετοί αλγόριθμοι κατασκευής τέτοιων δένδρων που αποσκοπούν στην ελαχιστοποίηση του ύψους του δένδρου και του συνολικού αριθμού κόμβων και συνεπώς στην ελαχιστοποίηση του χρόνου αναζήτησης. Γενικά, για να επιτευχθεί αυτός ο σκοπός θα πρέπει τα μικρού μεγέθους κλειδιά να αποθηκευθούν προς τη ρίζα, ώστε ο λόγος διακλάδωσης στα υψηλά επίπεδα να είναι μεγάλος. Το χαρακτηριστικό της δομής αυτής και των σχετικών αλγορίθμων είναι ότι εφαρμόζονται μόνο σε στατικά και γνωστά από την αρχή δεδομένα.

Στην κατηγορία των B-δένδρων μικρής τάξης εμπίπτουν πολλές παραλλαγές. Τα **2-3 δένδρα** και τα **1-2 δένδρα** είναι B-δένδρα τάξης 3 και 2 αντίστοιχα, δηλαδή περιέχουν 1 ως 2 και 0 ως 1 εγγραφές αντίστοιχα. Τα **2-3 δένδρα αδελφών** (2-3 Brother trees) είναι 2-3 δένδρα με τον επιπρόσθετο περιορισμό ότι ένας κόμβος με μία εγγραφή πρέπει να έχει οπωσδήποτε αδελφό κόμβο με δύο εγγραφές. Παρόμοιος περιορισμός ισχύει και στα **1-2 δένδρα αδελφών**. Τα **2-3 δένδρα υιών** (2-3 Son trees) είναι 2-3 δένδρα με τον επιπρόσθετο περιορισμό ότι δεν μπορεί να υπάρξει ένα ζεύγος

κόμβων με σχέση πατέρα-παιδιού, όπου και οι δύο κόμβοι να έχουν μία μόνο εγγραφή. Παρόμοιος περιορισμός ισχύει για τα 1-2 δένδρα υιών (1-2 Son trees). Είναι ευκολονόητο ότι οι περιορισμοί αυτοί καθιστούν δύσκολο καθήκον την ανάπτυξη των σχετικών αλγορίθμων διαχείρισης των δομών. Όλες αυτές οι παραλλαγές αποτελούν περισσότερο θεωρητικές αναζητήσεις παρά οργανώσεις με πρακτική εφαρμογή. Ο ενδιαφερόμενος αναγνώστης για τις δομές αυτές μπορεί να ανατρέξει στο βιβλίο του Gonnet, όπου θα βρει περισσότερα στοιχεία και αναφορές.

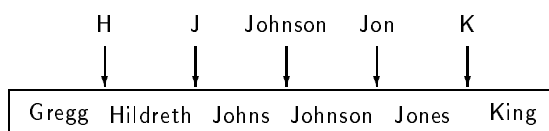
Τα Δυαδικά B-δένδρα είναι μία συγκεκριμένη υλοποίηση των 2-3 δένδρων με χρήση συνδεδεμένων λιστών μέσα σε κάθε κόμβο (Bayer 1972). Έτσι στη δομή συνυπάρχουν οι δενδρικοί δείκτες και οι δείκτες των κόμβων και τελικά το δένδρο μοιάζει με ένα απλό δυαδικό δένδρο. Στην εκδοχή αυτή ο δεξιός δενδρικός κόμβος μπορεί να καταστεί οριζόντιος προς τον αδελφό κόμβο, επομένως σε κάθε εγγραφή χρειάζεται και ένα επιπλέον πεδίο που δηλώνει αν ο δεξιός δείκτης είναι κατακόρυφος ή οριζόντιος. Τα Συμμετρικά Δυαδικά B-δένδρα επιτρέπουν και τον αριστερό δείκτη να είναι οριζόντιος (Bayer 1973). Μάλιστα σημειώνεται ότι τα δένδρα AVL είναι υποπερίπτωση των Συμμετρικών Δυαδικών B-δένδρων. Στο βιβλίο του Wirth υπάρχουν περισσότερα στοιχεία σχετικά με τους αλγορίθμους διαχείρισης των δένδρων αυτών.

Όλες αυτές οι παραλλαγές είναι δομές που λόγω επίδοσης αναφέρονται κυρίως στην κύρια μνήμη. Αντίθετα, τα Προθεματικά B⁺-δένδρα, που προτάθηκαν από τους Bayer και Unteraurer (1977), έχουν παρόμοια δομή με τα B⁺-δένδρα και είναι μία άλλη παραλλαγή για υλοποίηση κυρίως σε δευτερεύουσα μνήμη. Η βασική ιδέα είναι ότι αν το κλειδί είναι συμβολοσειρά, τότε είναι προτιμότερο στους εσωτερικούς κόμβους του καταλόγου να αποθηκεύεται μόνο εκείνο το πρόθεμα του κλειδιού, το οποίο είναι απαραίτητο για τη διαδικασία αναζήτησης. Έτσι ο κατάλογος περιέχει τους λεγόμενους διαχωριστές (separators) που είναι μεταβλητού μήκους. Το μήκος των διαχωριστών ποικίλει από ένα μόνο byte μέχρι και ένα πλήρες κλειδί



Σχήμα 8.17: Ελάχιστος διαχωριστής.

και εξαρτάται από τις τιμές των γειτονικών κλειδιών των εγγραφών. Ένα παράδειγμα φαίνεται στο Σχήμα 8.17, όπου ο διαχωριστής είναι ένα μόνο byte.



Σχήμα 8.18: Διαχωριστές σε Προθεματικό B^+ -δένδρο.

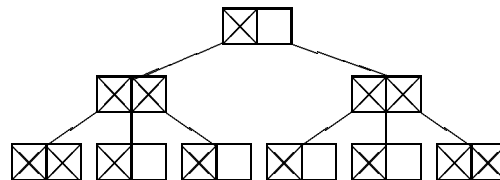
Στο Σχήμα 8.18 παρουσιάζεται ένας πλήρης κόμβος ενός Προθεματικού B^+ -δένδρου με έξι κλειδιά και με τους διαχωριστές μεταξύ των κλειδιών. Έστω ότι στον κόμβο αυτόν εισάγεται ένα νέο κλειδί, το όνομα JUNG, οπότε προκαλείται υπερχειλίση και διάσπαση του κόμβου. Αν τα κλειδιά αναδιανεμηθούν εξίσου, τότε πρέπει το κλειδί JOHNSON να αποθηκευθεί σε ένα νέο κόμβο προς τα δεξιά και ο ελάχιστος διαχωριστής, το JOHNSON, να ανέλθει κατά ένα επίπεδο. Απεναντίας αν τα κλειδιά δεν διανεμόνταν εξίσου στους δύο κόμβους, αλλά στον αριστερό κόμβο αποθηκεύονταν τα GREGG, HILDRETH και στο δεξιό τα JOHNS, JOHNSON κλπ., τότε ο διαχωριστής θα είναι το J. Συνεπώς η αναδιανομή των κλειδιών σε περίπτωση διάσπασης είναι θέμα βελτιστοποίησης. Εδώ υπεισέρχεται η έννοια του **διαστήματος διάσπασης** (split interval) των φύλλων, σ_{ex} , και των εσωτερικών κόμβων, σ_{in} , που ορίζεται ως ο αριθμός των bytes ή των κλειδιών προς οποιαδήποτε πλευρά του μέσου του κόμβου που θα μπορούσε να θεωρηθεί ως **σημείο διάσπασης** (split point). Το σημείο διάσπασης εκλέγεται μεταξύ του διαστήματος διάσπασης, ώστε να ελαχιστοποιηθεί ο σχετικός διαχωριστής. Αύξηση των σ_{in} και σ_{ex} επιτρέπει καλύτερη εκλογή του διαχωριστή, αλλά μπορεί να οδηγήσει σε κακή εκμετάλλευση του χώρου των φύλλων ως και λιγότερο από 50%.

Με την τεχνική αυτή όχι μόνο ελαχιστοποιείται ο χώρος που καταλαμβάνουν οι εσωτερικοί κόμβοι αλλά και η αναζήτηση γίνεται ταχύτερα. Αυτό φαίνεται εύκολα και διαισθητικά, γιατί το δένδρο έχει πλέον λιγότερα επίπεδα. Δεν αλλάζει, όμως, τίποτε σε σχέση με το B^+ -δένδρο σε ότι αφορά στην αναζήτηση της επόμενης εγγραφής και της εξαντλητικής ανάγνωσης του αρχείου. Η τεχνική αυτή χρησιμοποιείται πρακτικά σε πολλά συστήματα διαχείρισης βάσεων δεδομένων.

8.6 Ασκήσεις

<1> Να δοθούν αναλυτικές εκφράσεις για τον ελάχιστο και μέγιστο αριθμό εγγραφών σε B-δένδρο τάξης M που αποτελείται από L επίπεδα.

<2> B-δένδρο τάξης $m=6$ περιέχει 17 εγγραφές με κλειδιά 8, 13, 26, 28, 41, 49, 54, 67, 83, 84, 96, 107, 132, 146, 151, 154, 170 και αποτελείται από τρία επίπεδα. Να σχεδιασθεί το δένδρο και να φανεί η δομή του μετά τις διαγραφές των εγγραφών με κλειδιά 83, 67 και 26.



<3> Στο B-δένδρο τάξης $m=3$ του προηγούμενου σχήματος, οι κατειλημμένες θέσεις από εγγραφών σημειώνονται με διαγράμμιση. Στις θέσεις αυτές να τοποθετηθούν εγγραφές με κλειδιά $P, Y, A, Q, F, W, J, T, B, L, M, D$ και U . Να σχεδιασθεί το δένδρο και να φανεί η δομή του μετά από εισαγωγή εγγραφής με κλειδί Q , διαγραφή εγγραφής με κλειδί Q , διαγραφή εγγραφής με κλειδί M και εισαγωγή εγγραφής με κλειδί C .

<4> Να σχεδιασθεί το B*-δένδρο τάξης $m=6$ που δημιουργείται μετά από διαδοχικές εισαγωγές των εγγραφών με κλειδιά 25, 26, 24, 39, 32, 9, 28, 45, 13, 41, 5, 23, 19, 27, 6, 14, 34, 21, 31, 11 και 28. Στη συνέχεια να σχεδιασθεί η δομή του δένδρου μετά από διαγραφές των εγγραφών με κλειδιά 26, 21, 11, 9, 45, 13, 39, 6 και 14.

<5> Να σχεδιασθεί η κλάση των BT(4,15) και BT(4,16). Να επιβεβαιωθεί ο κανόνας των βέλτιστων δένδρων.

<6> Δίνεται B⁺-δένδρο που χρησιμεύει ως πρωτεύων κατάλογος. Ο αριθμός των χάδων του κατώτερου επιπέδου είναι bk . Στο δίσκο είναι αποθηκευμένα τα δύο κατώτερα επίπεδα του δένδρου, ενώ τα υπόλοιπα επίπεδα είναι αποθηκευμένα στην κύρια μνήμη που έχει μέγεθος 10 Mb. Αν υποθεθεί ότι ο μέσος λόγος διακλάδωσης είναι 140, τότε πόσο μεγάλο μπορεί να είναι το αρχείο ώστε ποτέ να μην υπάρξει κόστος μεγαλύτερο από 2 προσπελάσεις I/O.

<7> Δίνεται B^+ -δένδρο που χρησιμεύει ως δευτερεύων κατάλογος. Να θεωρηθεί ότι το πλήθος των ζευγών κλειδί-δείκτης είναι 40.000.000, το μέγεθος του κλειδιού είναι 4 bytes και το μέγεθος του δείκτη είναι 6 bytes. Να υπολογισθεί ο αριθμός των επιπέδων και ο αριθμός των κόμβων ανά επίπεδο, θεωρώντας 100% και 69% χρήση του χώρου. Η άσκηση να επαναληφθεί θεωρώντας ότι το μέγεθος του κλειδιού είναι 94 bytes (και επομένως το B^+ -δένδρο χρησιμεύει ως πρωτεύων κατάλογος).

<8> Δίνεται ένα B^+ -δένδρο με 6.000.000 εγγραφές των 400 bytes. Πόσος χώρος απαιτείται για το τελευταίο επίπεδο; Πόσος είναι ο απαιτούμενος χώρος για το επίπεδο επάνω από τα φύλλα, αν το ζεύγος κλειδί-δείκτης αποτελείται από 12 bytes, ενώ κάθε κάδος αποτελείται από 1 ή από 4 σελίδες; Για κάθε περίπτωση πόσος είναι ο χρόνος για την εξαντλητική του ανάγνωση;

<9> Έστω ότι στο αρχείο της προηγούμενης άσκησης εκτελείται μία εξαντλητική ανάγνωση κάθε 5000 απλές αναζητήσεις. Ο εσωτερικός κάδος έχει σταθερό μέγεθος μία σελίδα των 2400 bytes. Ποιό είναι το βέλτιστο μέγεθος του κάδου δεδομένων;

<10> Τα δεδομένα των προηγούμενων ασκήσεων είναι οργανωμένα ως πρωτεύον B^+ -δένδρο και ως δευτερεύον B^+ -δένδρο. Για κάθε περίπτωση ποιό είναι το βέλτιστο μέγεθος κόμβου ώστε μία ερώτηση διαστήματος που αναχτά το 5% των εγγραφών να είναι πιο οικονομική από τη σειριακή αναζήτηση ενός αρχείου σωρού;

<11> Για αρχείο που αποτελείται από 100.000 εγγραφές των 400 bytes χτίζεται ως δευτερεύων κατάλογος B^+ -δένδρο με μέγεθος φύλλων 2400 bytes. Έστω ότι πρέπει να δοθεί λίστα εγγραφών με τιμή στο δευτερεύον πεδίο μεγαλύτερη από μία συγκεκριμένη τιμή. Αν οι κατάλληλες εγγραφές στην έξοδο είναι 25% (ή 10% ή 5%) του συνόλου των εγγραφών, μήπως είναι προτιμότερο να γίνει εξαντλητική ανάγνωση του αρχείου και κατόπιν να ταξινομηθούν οι εγγραφές πριν δοθούν στο χρήστη.

<12> Σε B^+ -δένδρο τάξης $d=1$ και $Bkfr=2$ να εφαρμοσθεί διαδοχικά ο αλγόριθμος εισαγωγής για τις εγγραφές με κλειδιά 4, 12, 6, 1, 3, 15, 5, 9, 7, 14, 16, 10 και 11. Κατόπιν να γίνουν διαδοχικές διαγραφές των εγγραφών με κλειδιά 1, 3 και 4. Για τα ίδια δεδομένα να εξετασθεί ένα B^+ -δένδρο τάξης $d=3$ και $Bkfr=2$.

<13> Τα δεδομένα της προηγούμενης άσκησης μπορεί να δομηθούν είτε ως αρχείο σωρού, είτε ως B^+ -δένδρο. Ποιό είναι προτιμότερο αν σε κάθε 10 προσπελάσεις αντιστοιχούν 200 εισαγωγές. Να θεωρηθεί ότι $Bkfr=6$.

<14> Το αρχείο των 6.000.000 εγγραφών των 400 bytes είναι προτιμότερο να χτισθεί ως B^+ -δένδρο με 6.000.000 εισαγωγές ή με ταξινόμηση των δεδομένων και μετά αποθήκευση;

<15> Ένα αρχείο αποτελείται από 20.000 εγγραφές με κλειδί 9 bytes. Το μέγεθος της σελίδας είναι 2000 εγγραφές, ενώ ο δείκτης αποτελείται από 4 bytes. Επίσης έστωσαν δύο περιπτώσεις μήκους εγγραφής: 15 ή 180 bytes. Να θεωρηθούν οι εξής περιπτώσεις: στατικό δενδρικό αρχείο, στατικός πυκνός κατάλογος και κυρίως αρχείο, πρωτεύων και δευτερεύων κατάλογος B-δένδρου και κυρίως αρχείο. Για κάθε περίπτωση να βρεθεί το συνολικό μέγεθος σε σελίδες και το μέσο κόστος προσπέλασης.

Κεφάλαιο 9

ΣΤΑΤΙΚΑ ΤΥΧΑΙΑ ΑΡΧΕΙΑ

9.1 Εισαγωγή

9.2 Συναρτήσεις κατακερματισμού

9.3 Διαχείριση συνωνύμων με ανοικτή διεύθυνση

9.4 Διαχείριση συνωνύμων με άμεσες αλυσίδες

9.5 Διαχείριση συνωνύμων με ψευδοαλυσίδες

9.6 Ασκήσεις

Κεφάλαιο 9

ΣΤΑΤΙΚΑ ΤΥΧΑΙΑ ΑΡΧΕΙΑ

9.1 Εισαγωγή

Τα **τυχαία** ή **άμεσα** (random, direct) αρχεία είναι μία πολύ σπουδαία δομή. Αν τα αρχεία αυτά είναι σωστά σχεδιασμένα, τότε αποτελούν την ιδανική οργάνωση για on-line εφαρμογές, γιατί εντοπίζουν τη ζητούμενη εγγραφή με πολύ λίγες προσπελάσεις (ίσως και μόνο μία). Το βασικό μειονέκτημα της δομής αυτής είναι ότι δεν προσφέρεται για σειριακή ανάκτηση ταξινομημένων εγγραφών κατά οποιοδήποτε κλειδί.

Η βασική ιδέα της μεθόδου είναι η εφαρμογή κάποιου απλού αλγεβρικού μετασχηματισμού στην τιμή του κλειδιού, ώστε να προκύψει η απόλυτη διεύθυνση της εγγραφής. Η απόλυτη διεύθυνση εξαρτάται από το υλικό και αποτελείται από την τριάδα: αριθμός κυλίνδρου, αριθμός ατράχτου και αριθμός εγγραφής. Ωστόσο, είναι δυνατόν η μέθοδος να εφαρμοσθεί και κατά τρόπο που να μην εξαρτάται από το υλικό.

Τα κλειδιά λέγονται **πυκνά** (dense) όταν οι τιμές τους είναι συνεχόμενες. Στην απλούστερη περίπτωση πυκνών κλειδιών οι τιμές τους αρχίζουν από τη μονάδα. Για παράδειγμα, έστω ότι οι τιμές των κλειδιών είναι από 1 ως 5000 και ότι η χωρητικότητα των σελίδων είναι μία εγγραφή. Στην περίπτωση αυτή δημιουργείται ένα αρχείο των 5000 σελίδων, οπότε η i -οστή εγγραφή θα αποθηκευθεί στην $(i-1)$ -οστή σελίδα. Είναι, επίσης, δυνατόν οι τιμές των πυκνών κλειδιών να μην αρχίζουν από τη μονάδα αλλά, για παράδειγμα, να κυμαίνονται από το 10.001 ως το 15.000. Τότε αφαιρώντας μία σταθερή τιμή από την τιμή του κλειδιού (στη συγκεκριμένη περίπτωση το

10.000), και πάλι βρίσκεται εύκολα η κατάλληλη σελίδα του αρχείου. Τα αρχεία αυτά λέγονται **αυτοδεικτοδοτημένα** (self-indexing) και υλοποιούνται απλούστατα σε όλες τις γλώσσες προγραμματισμού.

Αν τα κλειδιά δεν είναι πυκνά, τότε με την τεχνική αυτή το μεγαλύτερο μέρος του αρχείου θα παραμείνει κενό και θα σπαταληθεί δευτερεύουσα μνήμη. Η περίπτωση αυτή αντιμετωπίζεται μετασχηματίζοντας την τιμή του κλειδιού, έτσι ώστε το διάστημα των κλειδιών να αντιστοιχεί σε ένα πολύ μικρότερο διάστημα τιμών διευθύνσεων. Η μέθοδος αυτή ονομάστηκε αρχικά **μετασχηματισμός του κλειδιού σε διεύθυνση** (key-to-address transformation), αλλά τελικά επικράτησε ο όρος **κατακερματισμός** (hashing), και γι' αυτό τα τυχαία αρχεία λέγονται και **αρχεία κατακερματισμού** (hashed files).

Κατά τη φόρτωση του αρχείου δεσμεύεται κάποιος χώρος του δίσκου, που ονομάζεται κύρια περιοχή. Σε μερικές παραλλαγές της μεθόδου προβλέπεται η μελλοντική χρήση περιοχής υπερχειλίσσης, που συνήθως βρίσκεται στον ίδιο κύλινδρο με την κύρια περιοχή, ώστε να μην υπάρξει χρονικό κόστος εντοπισμού. Όμως σε άλλες παραλλαγές δεν προβλέπεται ξεχωριστή περιοχή για τις εγγραφές υπερχειλίσσης. Παράγοντας διευθύνσεων (address factor) είναι το κλάσμα του μεγέθους της κύριας περιοχής προς το συνολικό μέγεθος του αρχείου. Αν ο χώρος που έχει κρατηθεί από το αρχείο δεν αυξομειώνεται με την πάροδο του χρόνου, τότε το αρχείο είναι στατικό. Όμως η έρευνα από το 1980 και εντεύθεν κατέληξε σε μία σειρά δομών τυχαίων αρχείων που μεγεθύνονται και συρρικνώνονται, ανάλογα με τον αριθμό των εισαγωγών και διαγραφών εγγραφών αντίστοιχα. Δηλαδή τα αρχεία αυτά είναι δυναμικά και θα εξετασθούν στο επόμενο κεφάλαιο.

9.2 Συναρτήσεις κατακερματισμού

Ο μετασχηματισμός της τιμής του κλειδιού σε τιμή διεύθυνσης επιτυγχάνεται με τη βοήθεια της **συνάρτησης κατακερματισμού** (hashing function) που σκοπό έχει την αντιστοίχιση του πεδίου τιμών των κλειδιών στο πεδίο των τιμών διευθύνσεων κατά τυχαίο τρόπο. Καλή συνάρτηση είναι εκείνη που διασπείρει τις εγγραφές σε όλη την έκταση του αρχείου, ακόμη και αν οι τιμές των κλειδιών συγκεντρώνονται σε μερικές περιοχές του πεδίου τιμών των κλειδιών. Για το λόγο αυτό η μέθοδος του κατακερματισμού ονομάζεται επίσης και **τεχνική διασκορπισμού αποθήκευσης** (scatter storage

technique). Αν τα κλειδιά είναι αριθμητικά, τότε μπορεί να εφαρμοσθεί μία απλή αλγεβρική έκφραση. Αν τα κλειδιά είναι αλφαβητικά ή αλφαριθμητικά, τότε πρέπει να μετατραπούν πρώτα σε αριθμητικά. Η μετατροπή αυτή μπορεί να γίνει χρησιμοποιώντας κάποια εσωτερική αναπαράσταση των χαρακτήρων των κλειδιών, όπως τους κώδικες ASCII, EBCDIC κλπ. Έτσι για παράδειγμα, το κλειδί C1 μπορεί να μετατραπεί σε 6749 (67 είναι ο κώδικας ASCII για το C και 49 είναι ο κώδικας για το 1).

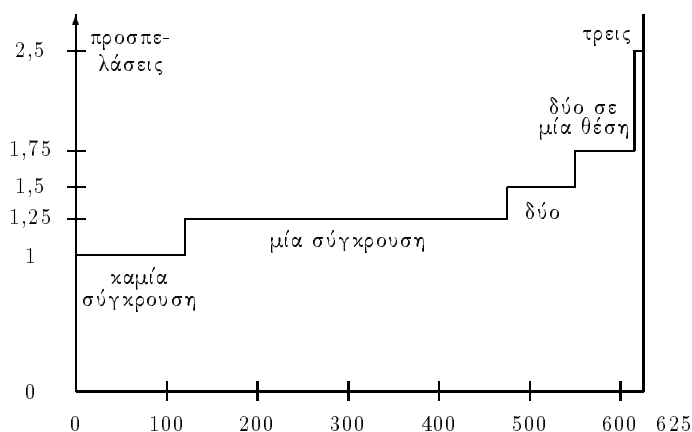
Το πρόβλημα των συναρτήσεων κατακερματισμού είναι ότι δύο ή περισσότερες διαφορετικές τιμές κλειδιών μπορεί να δώσουν την ίδια διεύθυνση. Αυτό το φαινόμενο καλείται **σύγκρουση** (collision), ενώ οι εγγραφές που συγκρούονται λέγονται **συνώνυμα** (synonyms). Αν από μία συνάρτηση προκύπτει σημαντικός αριθμός εγγραφών που συνωστίζονται στην ίδια περιοχή του αρχείου, τότε λέγεται ότι παρουσιάζεται **πρωτεύουσα συγκέντρωση** (primary clustering). Έτσι, συνήθως, δεσμεύεται για το αρχείο περισσότερος χώρος από την αρχική πρόβλεψη σχετικά με το τελικό πλήθος των εγγραφών. Στη βιβλιογραφία αναφέρονται πολλά χαρακτηριστικά παραδείγματα που δείχνουν ανάγλυφα ότι το πρόβλημα των συγκρούσεων εμφανίζεται με βεβαιότητα. Στη συνέχεια αναφέρονται δύο τέτοια παραδείγματα.

Έστω ότι οι 31 πιο συνηθισμένες αγγλικές λέξεις (and, or, not, the, κλπ.) αποθηκεύονται σε ένα πίνακα 42 θέσεων. Το σύνολο των δυνατών τοποθετήσεων είναι 10^{50} , ενώ οι βολικές τοποθετήσεις (δηλαδή χωρίς σύγκρουση) είναι μόνο 10^{43} . Άρα αντιστοιχεί μία βολική τοποθέτηση στις δέκα εκατομμύρια περιπτώσεις. Το δεύτερο παράδειγμα είναι το λεγόμενο 'παράδοξο των γενεθλίων'. Έστω, ότι επιλέγονται τυχαία από το πλήθος 23 άτομα. Η πιθανότητα τουλάχιστο δύο άτομα από το σύνολο των 23 ατόμων να έχουν την ίδια μέρα γενέθλια είναι 50.83%.

Το φαινόμενο των συγκρούσεων θα μοντελοποιηθεί μαθηματικά. Αν n εγγραφές πρέπει να αποθηκευθούν σε m θέσεις, τότε ο αριθμός των δυνατών τοποθετήσεων είναι m^n . Για παράδειγμα, αν $n=4$ και $m=5$ τότε ο αριθμός των τοποθετήσεων είναι 625. Στις βολικές τοποθετήσεις, που είναι $\frac{m!}{(m-n)!} = 120$, κάθε εγγραφή αναχτάται με μία προσπέλαση. Στη χειρότερη περίπτωση σε μία θέση αντιστοιχούν και οι τέσσερις εγγραφές, οπότε κάθε εγγραφή αναχτάται με $(n+1)/2=2,5$ προσπελάσεις κατά μέσο όρο. Ο αριθμός αυτών των δυσμενών περιπτώσεων είναι $m=5$. Οι υπόλοιπες 500 τοποθετήσεις διακρίνονται σε τρία είδη: μία σύγκρουση δύο εγγραφών, δύο συγκρούσεις δύο εγγραφών και μία σύγκρουση τριών εγγραφών.

Η πιθανότητα να εμφανισθεί μία βολική τοποθέτηση είναι $p_0 = \frac{\frac{m!}{(m-n)!}}{m^n} =$

0.192, οπότε η μέση τιμή των προσπελάσεων για εγγραφές υπερχειλίσης είναι $o_0=0$. Η πιθανότητα να εμφανισθεί η πιο δυσμενής τοποθέτηση είναι $p_{max} = \frac{m}{m^n} = 0.008$, οπότε ο αριθμός των συγκρούσεων και των προσπελάσεων για εγγραφή υπερχειλίσης είναι $c=n-1=3$ και $o_{max}=1+2+3$ αντίστοιχα. Στο Σχήμα 9.1. παρουσιάζονται τα στοιχεία των υπολοίπων τριών περιπτώσεων. Κατά μέσο όρο απαιτούνται $p = \sum_c p_c o_c = 0,3$ επιπρόσθετες προσπελάσεις για την ανάκτηση μίας εγγραφής. Η μέση τιμή των εγγραφών που συγκρούστηκαν και δεν αποθηκεύθηκαν στην κατάλληλη θέση είναι $o = \sum_c p_c c = 1,05$ εγγραφές, άρα κατά μέσο όρο $n-o=2,95$ εγγραφές αποθηκεύθηκαν στο κύριο αρχείο. Η πιθανότητα να συγκρουσθεί η επόμενη εγγραφή που θα εισαχθεί στο αρχείο είναι $1-2,95/5=0,41$. Για άλλες τιμές των m και n το σχήμα θα είναι παρόμοιο. Ο αριθμός των βημάτων αυξάνει πολύ γρήγορα για μεγαλύτερα m και n και η συνάρτηση ομαλοποιείται.



Σχήμα 9.1: Κατανομή συγκρούσεων και προσπελάσεων.

Στη βιβλιογραφία αναφέρονται πολλές μέθοδοι για την εύρεση συναρτήσεων κατακερματισμού που δεν παράγουν συνώνυμα. Οι συναρτήσεις αυτές λέγονται *τέλειες* (perfect) και συνήθως χρησιμοποιούνται στην πράξη σε μικρούς πίνακες της κύριας μνήμης για ειδικές εφαρμογές όπως, για παράδειγμα, σε μεταφραστές για αποθήκευση δεσμευμένων λέξεων, σε επεξεργασία φυσικής γλώσσας για φιλτράρισμα λέξεων υψηλής συχνότητας κλπ. Αν μία τέλεια συνάρτηση δεσμεύει τον ελάχιστο δυνατό χώρο, τότε λέγεται *ελάχιστη* (minimal). Μία αναγκαία προϋπόθεση των τέλειων μεθόδων κατακερματισμού είναι ότι συνήθως απαιτούν την εκ των προτέρων γνώση

των τιμών των κλειδιών. Ο αναγνώστης μπορεί να βρει περισσότερα στοιχεία για τις τέλειες μεθόδους κατακερματισμού στην κύρια μνήμη στο βιβλίο για Δομές Δεδομένων. Σχετικά πρόσφατα έχουν προταθεί τέλειες μέθοδοι κατακερματισμού για περιπτώσεις ογκωδών στατικών αλλά και δυναμικών αρχείων. Οι μέθοδοι αυτές αποτελούν μία κλάση μεθόδων που ονομάζονται **εξωτερικός τέλειος κατακερματισμός** (external perfect hashing) και δεν θα εξετασθούν στη συνέχεια λόγω της εξαιρετικής πολυπλοκότητάς τους.

Ο λόγος των κατειλημμένων θέσεων προς το σύνολο των θέσεων του αρχείου ονομάζεται **παράγοντας φόρτισης** (load factor, Lf), και ισούται με:

$$Lf = \frac{n}{bk_m \times Bkfr}$$

όπου bk_m είναι ο αριθμός των κάδων στην κύρια περιοχή του αρχείου. Υπάρχει άμεση σχέση της τιμής του παράγοντα φόρτισης και της συχνότητας των συγχρούσεων. Όσο μικρότερος είναι ο παράγοντας φόρτισης, τόσο μικρότερη είναι η πιθανότητα σύγχρουσης, και το αντίστροφο. Αρμοδιότητα του σχεδιαστή των αρχείων είναι να βρει την ισορροπία μεταξύ του πλήθους των συγχρούσεων και του ποσοστού αχρησιμοποίητου χώρου του αρχείου.

Υπερχείλιση συμβαίνει όταν μία εγγραφή πρέπει να αποθηκευθεί σε ένα πλήρη κάδο. Τότε η εγγραφή κατευθύνεται για αποθήκευση σε άλλον κάδο. Όσο μεγαλύτερος είναι ο κάδος, τόσο μικρότερη είναι η πιθανότητα να υπέρχει υπερχειλίση. Ωστόσο, όσο μεγαλύτερο είναι το μέγεθος του κάδου, τόσο πιο χρονοβόρα είναι η προσπέλαση στο δίσκο. Άρα, και σ' αυτό το σημείο πρέπει να βρεθεί μία ισορροπία.

Η στατιστική κατανομή που χαρακτηρίζει τη διανομή των εγγραφών στους κάδους είναι η διωνυμική. Αν οι εγγραφές μοιράζονται τυχαία στους κάδους, τότε η πιθανότητα να έχουν κατευθυνθεί σε ένα κάδο i από τις n εγγραφές είναι:

$$P(i) = \frac{n!}{i! \times (n-i)!} \times \left(\frac{1}{bk}\right)^i \times \left(1 - \frac{1}{bk}\right)^{n-i}$$

Για μεγάλες τιμές των n και bk η κατανομή αυτή μπορεί να προσεγγισθεί από την υπολογιστικά βολικότερη κατανομή Poisson:

$$P(i) = \frac{e^{-n/bk} \times \left(\frac{n}{bk}\right)^i}{i!}$$

Η μέση τιμή των εγγραφών υπερχείλισης σε ένα χάδο είναι:

$$\sum_{j=1}^{n-Bkfr} j \times P(Bkfr + j)$$

και συνεπώς το ποσοστό των εγγραφών υπερχείλισης είναι:

$$100 \times \frac{bk}{n} \times \sum_{j=1}^{n-Bkfr} j \times P(Bkfr + j)$$

Στον Πίνακα 9.1 δίνεται η μέση τιμή του ποσοστού των εγγραφών που υπερχειλίζουν ως συνάρτηση του μεγέθους του χάδου και του παράγοντα φόρτισης. Το μέγεθος του αρχείου έχει ληφθεί ίσο με 200 χάδους.

Μέγεθος χάδου	Παράγοντας φόρτισης									
	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
2	10,3	11,9	13,6	15,3	16,9	18,7	20,5	22,0	23,7	25,4
3	5,9	7,3	8,8	10,3	11,9	13,6	15,3	17,0	18,8	20,6
5	2,4	3,4	4,5	5,7	7,1	8,6	10,2	11,9	13,7	15,6
8	0,8	1,3	2,0	2,9	3,9	5,2	6,7	8,3	10,0	11,9
12	0,2	0,5	0,8	1,4	2,1	3,1	4,3	5,8	7,5	9,4
17	0,1	0,2	0,3	0,6	1,1	1,9	2,8	4,1	5,7	7,5
23	0,0	0,0	0,1	0,3	0,6	1,1	1,9	3,0	3,0	6,2

Πίνακας 9.1: Μέσο ποσοστό υπερχείλισης αρχείου.

Κάθε χρήστης μπορεί να εφεύρει μία νέα συνάρτηση κατακερματισμού με σκοπό να τυχαιοποιήσει τα δεδομένα του. Στη συνέχεια παρουσιάζονται μερικές συναρτήσεις που αναφέρονται κατά κόρο στη βιβλιογραφία. Έστω ότι δίνεται μία εγγραφή με κλειδί 172.148 για να αποθηκευθεί σε αρχείο που αποτελείται από 7000 χάδους.

Διαίρεση με πρώτο αριθμό (prime number division)

Η τιμή της διεύθυνσης, όπου θα αποθηκευθεί η εγγραφή, ισούται με το υπόλοιπο της διαίρεσης της τιμής του κλειδιού δια του μεγέθους του αρχείου. Σύμφωνα με μία πρακτική παρατήρηση οι συγκρούσεις ελαχιστοποιούνται αν διαιρέτης είναι ο μεγαλύτερος πρώτος αριθμός που είναι μικρότερος από το μέγεθος του αρχείου. Στην προκειμένη περίπτωση ισχύει: $172148 \bmod 6997 = 4220$. Βέβαια, διαιρέτης μπορεί να είναι οποιοσδήποτε αριθμός αρχεί να

είναι μικρότερος από το πλήθος των διευθύνσεων του αρχείου. Αναφέρεται επίσης από πρακτικές παρατηρήσεις ότι οι συγκρούσεις ελαχιστοποιούνται αν ο διαιρέτης δεν περιέχει πρώτους αριθμούς μικρότερους από 20.

Μετατροπή της ρίζας (radix conversion)

Θεωρείται ότι η τιμή του κλειδιού δεν είναι αριθμός του δεκαδικού συστήματος και επομένως πρέπει να μετατραπεί σε αριθμό του συστήματος αυτού. Έτσι αν υποθεθεί ότι ο συγκεκριμένος αριθμός έχει ως βάση το 11, τότε με τη μετατροπή προκύπτει:

$$1 \times 11^5 + 7 \times 11^4 + 2 \times 11^3 + 1 \times 11^2 + 4 \times 11 + 8 = 266373$$

Το αποτέλεσμα κανονικοποιείται διαιρώντας δια 6997. Η διεύθυνση του κάδου του αρχείου ισούται με το τελικό αποτέλεσμα που είναι 487.

Μέση του τετραγώνου (mid square)

Σύμφωνα με τη μέθοδο αυτή λαμβάνονται τα μεσαία ψηφία του τετραγώνου της τιμής του κλειδιού. Έτσι, αν η συγκεκριμένη τιμή του κλειδιού υψωθεί στο τετράγωνο, προκύπτει 029634933904. Λαμβάνονται τα 4 μεσαία ψηφία, δηλαδή 3493. Ακολουθεί κανονικοποίηση ως προς το 7000 και προκύπτει το αποτέλεσμα 2445.

Μετακίνηση ή δίπλωση (move, folding)

Πρόκειται για δύο παρόμοιες μεθόδους. Η τιμή του κλειδιού χωρίζεται σε δύο ή περισσότερα τμήματα που προστίθενται. Έστω, ότι δίνεται ο αριθμός 17207359 που χωρίζεται σε δύο τετραψήφιους αριθμούς, τους 1720 και 7359. Σύμφωνα με την πρώτη μέθοδο οι τιμές των τετραψηφίων αριθμών προστίθενται και προκύπτει 9079 (shift folding). Σύμφωνα με τη δεύτερη μέθοδο προστίθενται οι τετραψήφιοι αριθμοί, αφού πρώτα η σειρά των ψηφίων του δεύτερου αριθμού αντιστραφεί (boundary folding). Δηλαδή προστίθεται ο τετραψήφιος 1720 και ο 9537, που προέρχεται από τον 7359. Προκύπτει 11257. Στη συνέχεια ακολουθεί κανονικοποίηση ως προς το μέγεθος του αρχείου. Έτσι στην πρώτη περίπτωση θα προκύψει η διεύθυνση $9079 \times 7000 / 19.998 = 3178$, ενώ στη δεύτερη περίπτωση θα προκύψει $11.257 \times 7000 / 19.998 = 3940$.

9.3 Διαχείριση συνωνύμων με ανοικτή διεύθυνση

Η διαχείριση των συνωνύμων είναι το πιο κρίσιμο τμήμα κατά τη σχεδίαση ενός αρχείου κατακερματισμού. Διακρίνονται τρεις γενικές κατηγορίες

μεθόδων διαχείρισης της υπερχείλισης: οι μέθοδοι της ανοικτής διεύθυνσης (open addressing), οι μέθοδοι με άμεσες αλυσίδες (direct chaining) και οι μέθοδοι με ψευδοαλυσίδες (pseudo-chaining). Η βασική διαφορά των μεθόδων της πρώτης κατηγορίας από τις άλλες μεθόδους είναι η μη χρήση δεικτών ως ιδιαίτερων πεδίων στον τύπο της εγγραφής. Στο υποκεφάλαιο αυτό θα παρουσιασθούν παραλλαγές της πρώτης κατηγορίας, ενώ τα επόμενα δύο υποκεφάλαια είναι αφιερωμένα στην εξέταση των δύο άλλων κατηγοριών.

Στην πρώτη, λοιπόν, κατηγορία ανήκουν παραλλαγές όπως η γραμμική αναζήτηση (linear search), η μη γραμμική αναζήτηση (nonlinear search), ο επανακατακερματισμός (rehashing) και ο διπλός κατακερματισμός (double hashing). Η γραμμική αναζήτηση παρουσιάστηκε χρονικά πριν από κάθε άλλη μέθοδο κατακερματισμού. Ήδη από το 1953 γίνεται αναφορά γι' αυτήν από ερευνητές της IBM, ο πρώτος όμως που τη συστηματοποίησε σε δημοσίευση ήταν ένας Ρώσος επιστήμονας, ο Ershov το 1957. Ακολουθεί παρουσίαση των παραλλαγών αυτών που έχουν κοινό σημείο τη μη υπάρξη ξεχωριστής περιοχής υπερχείλισης.

Σύμφωνα με τη γραμμική αναζήτηση η εγγραφή που υπερχειλίζει αποθηκεύεται στον πρώτο επόμενο διαθέσιμο κάδο. Δηλαδή, η αναζήτηση προχωρεί γραμμικά στους επόμενους κάδους μέχρις ότου ή να βρεθεί κενός χώρος ή να προσπελασθεί και πάλι ο αρχικός κάδος (αν όλοι οι κάδοι είναι πλήρεις). Αυτός ο τύπος αναζήτησης λέγεται και μέθοδος ανοικτής υπερχείλισης (open overflow) ή γραμμικής εξέτασης (linear probing) ή προοδευτικής υπερχείλισης (progressive overflow). Ο αριθμός των προσπελάσεων κάδων που μεσολαβούν πριν προσπελασθεί ο κάδος, όπου πράγματι η εγγραφή είναι αποθηκευμένη, λέγεται μετατόπιση (displacement) της εγγραφής. Όταν οι τιμές των κλειδιών δεν είναι πραγματικά τυχαίες, αλλά συσσωρεύονται γύρω από ορισμένες τιμές, τότε η μετατόπιση στη χειρότερη περίπτωση είναι αρκετά μεγαλύτερη από τη μέση τιμή. Επίσης, όταν ο παράγοντας φόρτισης τείνει προς τη μονάδα, τότε η μέση τιμή της μετατόπισης αυξάνει σημαντικά.

Στο Σχήμα 9.2 φαίνεται ένα παράδειγμα εφαρμογής της μεθόδου αυτής σε ένα αρχείο αποτελούμενο από 10 διευθύνσεις αριθμημένες από 0 ως 9. Κάθε διεύθυνση αντιστοιχεί σε 1 κάδο των 2 σελίδων, με χωρητικότητα 2 εγγραφών ανά σελίδα. Ήδη στο αρχείο έχει εισαχθεί ένα σύνολο εγγραφών, όταν έρχεται για εισαγωγή η εγγραφή με κλειδί 220. Σύμφωνα με τη συνάρτηση μετασχηματισμού της διαίρεσης ($220 \bmod 10$), η εγγραφή αυτή πρέπει να αποθηκευθεί στη διεύθυνση 0, που όμως είναι πλήρης. Έτσι η

0	10	90	100	150
1	71	220		
2	12	52	142	202
3	13	193		
4	4			
5	5	175	215	
6	56	146		
7				
8				
9	19			

Εγγραφή
 ┌──Σελίδα──┐
 └──────────┘ Κάδος

Σχήμα 9.2: Μέθοδος γραμμικής εξέτασης.

εγγραφή 220 αποθηκεύεται στην αμέσως επόμενη διεύθυνση με διαθέσιμο χώρο, δηλαδή τη διεύθυνση 1. Έτσι, η μετατόπιση είναι ένας κάδος.

Για να αποφευχθεί το πρόβλημα που δημιουργείται από τη συσσώρευση των κλειδιών γύρω από μία περιοχή διευθύνσεων, συχνά χρησιμοποιούνται μη γραμμικές μέθοδοι αναζήτησης του επόμενου διαθέσιμου κάδου. Τέτοια μέθοδος είναι η **τετραγωνική** (quadratic) αναζήτηση, που σκοπό έχει την απομάκρυνση των εγγραφών από την αρχική περιοχή σύγκρουσης. Σύμφωνα με τη μέθοδο αυτή οι θέσεις των επόμενων κάδων που θα εξετασθούν βρίσκονται από τη σχέση:

$$(A \times i^2 + B \times i + key \bmod bk) \bmod bk$$

όπου i είναι ο αύξων αριθμός της αναζήτησης ($i=0,1,2,\dots$), A και B είναι αυθαίρετες σταθερές, ενώ key είναι η τιμή του κλειδιού. Έστω και πάλι η περίπτωση σύγκρουσης της εγγραφής 220 κατά την εισαγωγή στη θέση 0 του Σχήματος 9.2. Θεωρώντας την προηγούμενη τετραγωνική σχέση με $A=1$ και $B=1$, για $i=1$ προκύπτει ότι η εγγραφή 220 πρέπει να αποθηκευθεί στην διεύθυνση 2, που είναι επίσης κατειλημμένη. Τελικά θέτοντας $i=2$ προκύπτει ότι η εγγραφή 220 αποθηκεύεται στη θέση 6. Στην περίπτωση αυτή η μετατόπιση είναι δύο κάδοι.

Μία άλλη τεχνική είναι η μέθοδος του επανακατακερματισμού, όπου χρησιμοποιείται μία άλλη συνάρτηση μετασχηματισμού για να εντοπίσει τον επόμενο διαθέσιμο κάδο. Για παράδειγμα, έστω ότι χρησιμοποιείται η συ-

νάρτηση της διαίρεσης

$$h_1(key) = key \bmod bk$$

για την εύρεση της αρχικής θέσης αποθήκευσης της εγγραφής. Σε περίπτωση σύγκρουσης χρησιμοποιείται μία άλλη συνάρτηση για να δώσει την απόσταση σε κάδους από την αρχική θέση. Σε περίπτωση νέας σύγκρουσης γίνεται δοκιμή σε θέση που απέχει την ίδια απόσταση από τη θέση της δεύτερης σύγκρουσης. Ένα παράδειγμα συνάρτησης επανακατακερματισμού είναι:

$$dis = (h_1(key) + p) \bmod bk$$

όπου p είναι μία σταθερά που επιλέγεται έτσι ώστε να είναι πρώτος αριθμός προς το bk . Στο γνωστό παράδειγμα μετά τη σύγκρουση της εγγραφής 220 στη θέση 0, υπολογίζεται η απόσταση $dis = (0 + 3) \bmod 10 = 3$ και η εγγραφή αποθηκεύεται στη θέση 3. (Το 3 και το 10 είναι πρώτοι αριθμοί μεταξύ τους.) Επίσης, σημειώνεται ότι αν προκύψει $dis=0$, τότε τίθεται $dis=1$.

Όπως αναφέρθηκε καλή είναι εκείνη η συνάρτηση κατακερματισμού που διασπείρει ομοιόμορφα τις εγγραφές σε όλο το μέγεθος του αρχείου. Στην αντίθετη περίπτωση υπάρχει πρωτεύουσα συγχέντρωση, δηλαδή πολλές εγγραφές συνωστίζονται σε μία ορισμένη περιοχή του αρχείου. Είναι δυνατόν να παρουσιασθούν πολλές συγχρούσεις λόγω συνωνυμιών, οπότε υπάρχει **δευτερεύουσα συγχέντρωση** (secondary clustering). Η επίδραση του φαινομένου αυτού μειώνεται με τη μέθοδο του διπλού κατακερματισμού που είναι μία πιο κομψή εκδοχή του επανακατακερματισμού. Στο διπλό κατακερματισμό η παράμετρος p δεν είναι σταθερά αλλά υπολογίζεται κάθε φορά με μία άλλη συνάρτηση. Συνεπώς με τη μέθοδο αυτή απαιτούνται συνολικά τρεις συναρτήσεις. Η δεύτερη συνάρτηση είναι της μορφής:

$$h_2(key) = key \bmod (bk - q) + 1$$

όπου το q είναι σταθερά. Το αποτέλεσμα της συνάρτησης h_2 εισάγεται στην:

$$dis = (h_1(key) + h_2(key)) \bmod bk$$

Στο γνωστό παράδειγμα μετά τη σύγκρουση της εγγραφής 220 στη θέση 0, υπολογίζεται η απόσταση $dis = (0 + 220 \bmod 9 + 1) \bmod 10 = 6$ και η εγγραφή αποθηκεύεται στη θέση 4 (θεωρήθηκε $q=1$). Σε περίπτωση νέων συγχρούσεων η εγγραφή 220 θα κατευθύνονταν διαδοχικά στις θέσεις 8, 2, 6 κλπ., ενώ η εγγραφή 230 θα κατευθύνονταν στις θέσεις 7, 4, 1, 8 κλπ.

Μέχρι στιγμής δεν αναφέρθηκε τίποτε σε σχέση με τη διαγραφή. Η πιο συνήθης και εύκολη τεχνική είναι το μαρκάρισμα των διαγραμμένων εγγραφών. Ειδικά, αν είναι αναγκαίο να ελευθερωθεί ο χώρος για κατοπινές εισαγωγές, τότε απαιτούνται χρονοβόροι αλγόριθμοι διαγραφής. Πρέπει να σημειωθεί επίσης ότι στη βιβλιογραφία αναφέρονται μέθοδοι που δεν αποθηκεύουν μόνιμα κάποια εγγραφή υπερχειλίστη στην αρχική της θέση αλλά είναι δυνατόν να την εκτοπίσουν προκειμένου η θέση αυτή να καταληφθεί από εγγραφή που έρχεται φυσιολογικά στη θέση αυτή με βάση την πρώτη συνάρτηση κατακερματισμού. Οι ενδιαφέρουσες αυτές τεχνικές, που βελτιστοποιούν την επίδοση της αναζήτησης με τίμημα το αυξημένο κόστος εισαγωγής, είναι εκτός των ορίων του τόμου αυτού. Εξ άλλου, όπως θα τονισθεί στη συνέχεια, οι μέθοδοι της ανοικτής διεύθυνσης δεν χρησιμοποιούνται στην πράξη για δομές αρχείων, αλλά κυρίως για δομές κύριας μνήμης. Στο βιβλίο των Δομών Δεδομένων υπάρχουν ποσοτικά στοιχεία σχετικά με την επίδοση των μεθόδων αυτών.

9.4 Διαχείριση συνωνύμων με άμεσες αλυσίδες

Σημαντικό μειονέκτημα των προηγούμενων μεθόδων είναι η κακή επίδοση σε περίπτωση αναζήτησης όταν ο παράγοντας φόρτισης πλησιάζει τη μονάδα. Οι επιδόσεις όλων των διεργασιών βελτιώνονται αν θεωρηθεί ένα επιπλέον πεδίο στον τύπο της εγγραφής, για να εξυπηρετεί ως δείκτης προς τις επόμενες λογικά εγγραφές και έτσι να τις συνδέει σε μία αλυσίδα. Η πρώτη αναφορά σχετικά με τη μέθοδο των αλυσίδων έγινε από τον Dumey (1956) και από τότε έχουν προταθεί πολλές παραλλαγές που ανήκουν σε δύο οικογένειες: των σύμφυτων αλυσίδων και των ξεχωριστών αλυσίδων. Και οι δύο μπορούν να υλοποιηθούν είτε με ιδιαίτερη περιοχή υπερχειλίστη είτε χωρίς τέτοια περιοχή.

Η μέθοδος των σύμφυτων αλυσίδων προτάθηκε από τον Williams (1959), αναλύθηκε όμως σε μεγάλο βάθος και έχταση από τον Vitter στη δεκαετία του 80. Γενικά η μέθοδος διατηρεί μία κυκλική συνδεδεμένη λίστα από τις διευθύνσεις των διαθέσιμων χώρων, ώστε να χρησιμοποιηθούν για την αποθήκευση των εγγραφών υπερχειλίστη. Όταν μία εγγραφή υπερχειλίστει από τον χώρο A, τότε αποθηκεύεται στον χώρο B, που είναι ο πρώτος χώρος της συνδεδεμένης λίστας. Όσες εγγραφές υπερχειλίζουν αποθηκεύονται στον χώρο B μέχρι να γεμίσει και να διαγραφεί από τη λίστα των διαθέσιμων.

Κάθε εγγραφή που στο μέλλον θα υπερχειλίσει από τον κάδο Α θα κατευθυνθεί προς τον Γ, όπου όμως θα κατευθυνθούν και οποιεσδήποτε εγγραφές υπερχειλίσουν από τον Β. Έτσι, στον κάδο Γ συμφύονται οι αλυσίδες των συνώνυμων που προέρχονται από τους κάδους Α και Β.

0	10	90	100	150
1	71	220		
2	12	52	142	202
3	113	193		
4	4			
5	5	175	215	
6	56	146		
7				
8				
9	19			

Κεφαλή
ελεύθερου
χώρου

Σχήμα 9.3: Μέθοδος σύμφυτων αλυσίδων.

Στο Σχήμα 9.3 παρουσιάζεται η δομή με τα δεδομένα των προηγούμενων παραδειγμάτων. Και πάλι έχει ήδη αποθηκευθεί κατά τον ίδιο τρόπο η ίδια ακολουθία εγγραφών. Η διεύθυνση 0 είναι ήδη γεμάτη όταν έρχεται η εγγραφή 220. Η κορυφή της λίστας των διαθέσιμων κάδων δείχνει στον κάδο 1. Συνεπώς, η εγγραφή 220 κατευθύνεται στον κάδο αυτό.

Το προηγούμενο σχήμα είναι αρκετά αφαιρετικό και δεν δείχνει την ακριβή γραμμογράφηση ούτε του κάδου ούτε της εγγραφής. Είναι όμως απαραίτητη η σωστότερη δόμησή τους. Πιο συγκεκριμένα, κάθε εγγραφή πρέπει να έχει δύο πεδία δεικτών που να δείχνουν τον κάδο, όπου είναι αποθηκευμένη η λογικά επόμενη και η λογικά προηγούμενη εγγραφή. Είναι δυνατόν επίσης μία εγγραφή να εισαχθεί σε κάδο που είναι κατειλημμένος από εγγραφές υπερχειλίσης. Επομένως απαιτείται ένα επιπλέον πεδίο-δείκτης στο επίπεδο του κάδου, που να δείχνει τη διεύθυνση του κάδου, όπου είναι αποθηκευμένη η πρώτη λογικά εγγραφή που θα έπρεπε να είχε αποθηκευθεί στο συγκεκριμένο κάδο.

Στη συνέχεια θα εξετασθεί η μέθοδος των ξεχωριστών αλυσίδων που, όπως είναι γνωστό, μπορεί να διατηρεί ιδιαίτερη λογικά περιοχή υπερχειλίσης. Η περιοχή αυτή φυσικά μπορεί να είναι είτε τμήμα του κύριου αρχείου είτε ανεξάρτητο αρχείο. Εδώ θα εξετασθεί η δεύτερη περίπτωση και θα δοθούν αναλυτικές εκφράσεις για το χρονικό κόστος των διαφόρων διεργασιών,

γιατί αυτή η μέθοδος εφαρμόζεται περισσότερο στα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων.

Προσπέλαση μίας εγγραφής σημαίνει μία σειρά ενεργειών. Πρώτον, το κλειδί μετασχηματίζεται, δεύτερον, μεταφέρεται ο κάδος στην κύρια μνήμη, τρίτον, οι εγγραφές του κάδου διαβάζονται σειριακά και, τέλος, σε περίπτωση αποτυχίας η αναζήτηση συνεχίζεται στην περιοχή υπερχειλίσης. Αρμοδιότητα του σχεδιαστή του αρχείου είναι η απόφαση της συγκεκριμένης υλοποίησης της περιοχής αυτής που μπορεί να γίνει κατά πολλούς τρόπους. Το κύριο ερώτημα είναι τί μεγέθους θα είναι οι κάδοι της περιοχής αυτής. Αν η περιοχή υπερχειλίσης είναι τμήμα του κύριου αρχείου (οπότε λέγεται *κελλάρι* - *cellar*), τότε το μέγεθος του κάδου προφανώς είναι ενιαίο.

	Κύρια περιοχή				Υπερχειλίση
0	10	90	100	150	220
1	71				
2	12	52	142	202	
3	113	193			
4	4				
5	5	175	215		
6	56	146			
7					
8					
9	19				

Σχήμα 9.4: Μέθοδος ξεχωριστών αλυσίδων.

Στο Σχήμα 9.4 δίνεται ένα παράδειγμα της τεχνικής αυτής παρόμοιο με τα προηγούμενα. Στην περιοχή υπερχειλίσης για κάθε διεύθυνση της κύριας περιοχής δεσμεύεται ένας κάδος, που αποτελείται από μία σελίδα με χωρητικότητα μίας εγγραφής. Έχει αποθηκευθεί η ίδια ακολουθία εγγραφών, οπότε εισάγεται η εγγραφή 220 στη διεύθυνση 0 που είναι ήδη γεμάτη. Συνεπώς, η εγγραφή 220 κατευθύνεται στην περιοχή υπερχειλίσης και αποθηκεύεται στον αντίστοιχο κάδο.

9.4.1 Αναζήτηση εγγραφής

Η μέθοδος είναι πολύ αποτελεσματική αν η κύρια περιοχή είναι αρκετά μεγάλη και η κατανομή των εγγραφών στους κάδους είναι περίπου ομοιόμορφη.

Με την προϋπόθεση ότι δεν υπάρχει υπερχειλίση, το χρονικό κόστος για την προσπέλαση μίας εγγραφής είναι:

$$T_{\text{προσ}} = s + r + dtt$$

Με την πάροδο του χρόνου και την αποθήκευση πολλών εγγραφών είναι βέβαιο ότι θα υπάρξει υπερχειλίση με επιμήκεις αλυσίδες. Στην περίπτωση αυτή το χρονικό κόστος είναι:

$$T_{\text{προσ}}(\text{επιτ}) = (s + r + dtt) + \frac{x}{2} \times (s + r + dtt)$$

Ο τύπος εξηγείται ως εξής. Η πρώτη παρένθεση δηλώνει την προσπέλαση του κάδου της κύριας περιοχής, ενώ το γινόμενο δηλώνει τη διάσχιση της αλυσίδας x κάδων υπερχειλίσης για τον εντοπισμό της ζητούμενης εγγραφής. Κατά μέσο όρο η μισή αλυσίδα θα πρέπει να διασχισθεί.

Μία αναζήτηση είναι ανεπιτυχής, όταν όλες οι τιμές των κλειδιών των εγγραφών που εξετάζονται, είτε στην κύρια είτε στην περιοχή υπερχειλίσης, δεν ταυτίζονται προς το ζητούμενο κλειδί. Έτσι η αναζήτηση τερματίζει όταν φθάσει σε εγγραφή με δείκτη NIL. Η ανεπιτυχής αναζήτηση είναι πιο χρονοβόρα από την επιτυχή, γιατί πρέπει να διασχισθεί ολόκληρη η αλυσίδα υπερχειλίσης (ενώ στο B^+ -δένδρο η επιτυχής και η ανεπιτυχής αναζήτηση έχουν το ίδιο κόστος). Άρα, το σχετικό κόστος είναι:

$$T_{\text{προσ}}(\text{ανεπ}) = (s + r + dtt) + x \times (s + r + dtt)$$

Αυτός ο τύπος ισχύει όταν το ποσοστό υπερχειλίσης είναι μεγάλο. Αν το ποσοστό αυτό είναι μικρό, τότε ισχύει η πρώτη σχέση και για τους δύο τύπους αναζητήσεων. Φαίνεται, λοιπόν, ότι κατά τη φάση σχεδίασης του αρχείου πρέπει να προσεχθούν μερικές παράμετροι, ώστε πράγματι η επίδοση του αρχείου να είναι υψηλή. Αν το αρχείο αποκτήσει μεγάλη περιοχή υπερχειλίσης πρέπει να αναδιοργανωθεί και να επανα-αποθηκευθεί σε μεγαλύτερη κύρια περιοχή.

Πρέπει να τονισθεί ότι οι προηγούμενοι τύποι είναι προσεγγιστικοί, γιατί δεν λαμβάνουν υπ' όψη τον παράγοντα φόρτισης και τη χωρητικότητα του κάδου. Οι Πίνακες 9.2 και 9.3 δίνουν τον αριθμό των προσπελάσεων σε αρχείο κατακερματισμού για επιτυχή και ανεπιτυχή αναζήτηση αντίστοιχα, ως συνάρτηση του παράγοντα φόρτισης και της χωρητικότητας του κάδου. Τα στοιχεία αυτά έχουν προκύψει από μαθηματική ανάλυση που είναι εκτός των ορίων του βιβλίου, οι πίνακες όμως παρουσιάζονται γιατί από αυτούς εξάγονται ποιοτικά πρακτικά συμπεράσματα. Απαραίτητες προϋποθέσεις εργασίας είναι ότι:

Μέγεθος κάδου	Παράγοντας φόρτισης							
	50%	70%	90%	100%	150%	200%	300%	500%
1	1,3	1,4	1,5	1,5	1,8	2,0	2,5	3,5
2	1,1	1,2	1,4	1,4	1,8	2,3	3,2	5,1
3	1,1	1,2	1,3	1,4	1,9	2,5	3,8	6,7
4	1,1	1,1	1,3	1,4	2,0	2,8	4,5	8,3
5	1,0	1,1	1,3	1,4	2,1	3,0	5,2	9,9
10	1,0	1,1	1,2	1,3	2,5	4,3	8,5	17,9
20	1,0	1,0	1,2	1,3	3,3	6,8	15,2	33,9
50	1,0	1,0	1,1	1,3	5,8	14,3	35,2	81,9

Πίνακας 9.2: Μέση τιμή προσπελάσεων για επιτυχή αναζήτηση σε τυχαίο αρχείο με ξεχωριστές αλυσίδες.

Μέγεθος κάδου	Παράγοντας φόρτισης								
	20%	30%	40%	50%	60%	70%	80%	90%	95%
1	1,0	1,0	1,1	1,1	1,1	1,2	1,2	1,3	1,3
2	1,0	1,0	1,0	1,1	1,2	1,2	1,3	1,4	1,5
3	1,0	1,0	1,0	1,1	1,2	1,3	1,4	1,5	1,6
4	1,0	1,0	1,0	1,1	1,1	1,3	1,4	1,6	1,6
5	1,0	1,0	1,0	1,1	1,1	1,2	1,4	1,6	1,6
10	1,0	1,0	1,0	1,0	1,1	1,2	1,4	1,8	2,0
20	1,0	1,0	1,0	1,0	1,0	1,1	1,4	1,9	2,3
50	1,0	1,0	1,0	1,0	1,0	1,0	1,2	1,9	2,7

Πίνακας 9.3: Μέση τιμή προσπελάσεων για ανεπιτυχή αναζήτηση σε τυχαίο αρχείο με ξεχωριστές αλυσίδες.

- οι εγγραφές κατανέμονται στους κάδους με δυωνυμική κατανομή,
- οι εγγραφές αναζητώνται ισοπίθανα, και
- η χωρητικότητα των κάδων στην κύρια περιοχή και στην περιοχή υπερχείλισης είναι $Bkfr$ και 1, αντίστοιχα.

Όπως αναμένεται, και στις δύο περιπτώσεις η απόδοση του αρχείου εκφυλίζεται καθώς αυξάνει ο παράγοντας φόρτισης. Ακόμη, αν ο παράγοντας φόρτισης είναι μικρότερος του 100% και του 50%, τότε η αύξηση της χωρητικότητας του κάδου έχει ως αποτέλεσμα σταθερή βελτίωση της απόδοσης

της επιτυχούς και της ανεπιτυχούς αναζήτησης, αντίστοιχα, επειδή μικραίνει το μήκος της αλυσίδας των κάδων. Αυτό όμως το πλεονέκτημα, που ισχύει για μεγάλες χωρητικότητες, αντισταθμίζεται από το γεγονός ότι μεγαλώνει ο χρόνος μεταφοράς του κάδου, dtt . Πάντως, το πρακτικό συμπέρασμα είναι ότι ο παράγοντας φόρτισης δεν πρέπει να υπερβαίνει το 70%-80%, γιατί στην περιοχή αυτή βελτιστοποιείται η σχέση κενός χώρος προς επίδοση.

9.4.2 Διαγραφή εγγραφής

Υπάρχουν δύο τρόποι διαγραφής των εγγραφών. Ο ένας είναι παρόμοιος με τον τρόπο που χρησιμοποιείται στα αρχεία σωρού και στα αρχεία ISAM. Δηλαδή, ανεβαίνει μία σημαία που δηλώνει ότι η εγγραφή είναι ουσιαστικά διαγραμμένη χωρίς να απομακρυνθεί φυσικά, ώστε να αποδοθεί ο χώρος στο σύστημα. Κατά την περιοδική αναδιοργάνωση οι εγγραφές ταχτοποιούνται και πάλι χωρίς να γίνεται σπατάλη χώρου.

Σύμφωνα με το δεύτερο τρόπο, η εγγραφή διαγράφεται φυσικά και αντικαθίσταται με την τελευταία εγγραφή της αλυσίδας που ξεκινά από τον κάδο αυτόν. Επιπλέον, αν ο τελευταίος κάδος αδειάσει από εγγραφές, τότε αποδίδεται και πάλι στο σύστημα. Αυτή η τεχνική είναι περισσότερο αποτελεσματική από την άποψη του καταλαμβανόμενου χώρου και του χρονικού κόστους για αναζήτηση. Ωστόσο η διαγραφή καθίσταται πιο χρονοβόρα, γιατί πρέπει να γίνει διάσχιση της αλυσίδας ώστε να μεταφερθεί η τελευταία εγγραφή στον κενό χώρο. Έτσι στην ουσία εκτελείται μία ανεπιτυχής αναζήτηση. Ύστερα πρέπει να επαναποθηκευθούν δύο κάδοι, αυτός όπου γίνεται η διαγραφή και ο τελευταίος της αλυσίδας. Στην απλή περίπτωση όπου αυτοί οι δύο κάδοι ταυτίζονται, το χρονικό κόστος είναι:

$$T_{\text{διαγ}} = T_{\text{προσ(ανεπ)}} + 2r$$

Οι περισσότερες υλοποιήσεις στατικών άμεσων αρχείων χρησιμοποιούν την πρώτη μέθοδο. Οι νέες μέθοδοι οργάνωσης δυναμικών αρχείων κατακερματισμού δεν χρειάζονται αναδιοργάνωση, όπως απαιτούν περιοδικά οι στατικές. Φαίνεται ότι στο μέλλον οι στατικές οργάνώσεις θα χρησιμοποιούν κυρίως τη δεύτερη μέθοδο διαγραφών.

9.4.3 Εισαγωγή εγγραφής

Έστω ότι μία εισαγόμενη εγγραφή αποθηκεύεται στον τελευταίο κάδο της αλυσίδας. Άρα, και αυτή η περίπτωση μοιάζει με μία ανεπιτυχής αναζήτηση.

Στη συνέχεια πρέπει ο τελευταίος κάδος να επανα-αποθηκευθεί συμπεριλαμβανόντας την τελευταία εγγραφή. Συνεπώς το χρονικό κόστος είναι:

$$T_{\text{εισ}} = T_{\text{προσ}}(\text{ανεπ}) + 2r$$

Ο τύπος αυτός δεν λαμβάνει το κόστος που απαιτείται για την προσάρτηση στην αλυσίδα ενός νέου κάδου, όταν ο τελευταίος είναι πλήρης και δεν διαθέτει χώρο για την εγγραφή. Φαίνεται, λοιπόν, ότι και η εισαγωγή δεν είναι δαπανηρή διεργασία.

9.4.4 Εξαντλητική ανάγνωση αρχείου

Η μέθοδος του κατακερματισμού καταστρέφει τη λογική σειρά των εγγραφών, και είναι προβληματική όταν τίθενται ερωτήσεις διαστήματος με βάση το πρωτεύον ή κάποιο δευτερεύον κλειδί. Για παράδειγμα, έστω ότι δίνεται ένα αρχείο χιλίων υπαλλήλων μίας εταιρείας και ότι απαιτείται να προσπελασθούν οι εγγραφές των υπαλλήλων με κωδικούς από 100 ως 200. Αυτή η περίπτωση θα ικανοποιηθεί με μία χρονοβόρα διαδικασία γιατί θα εκτελεσθούν εκατό ανεξάρτητες προσπελάσεις. Όμως αν ζητηθεί να προσπελασθούν οι εγγραφές των υπαλλήλων με βάση το επίθετο από το Κόλλιας ως το Μανωλόπουλος, τότε η απάντηση είναι υπερβολικά χρονοβόρα, επειδή πρέπει να προσπελασθούν όλες οι εγγραφές και να γίνει επιλογή των κατάλληλων. Έτσι προκύπτει ότι το κόστος για την ανάκτηση όλων των εγγραφών ενός αρχείου είναι:

$$T_{\text{εξαν}} = n \times T_{\text{προσ}}$$

Επιπλέον απαιτείται χρόνος για την ταξινόμηση των εγγραφών πριν δοθούν στο χρήστη.

9.4.5 Αναδιοργάνωση αρχείου

Αναδιοργάνωση του αρχείου συμβαίνει όταν οι αποθηκευμένες εγγραφές ξεπεράσουν τα πλαίσια, που είχαν τεθεί κατά τη φάση του σχεδιασμού, και η επίδοση εκφυλίζεται. Η αναδιοργάνωση είναι εξαιρετικά χρονοβόρα διαδικασία, ενώ συχνά η επιλογή του χρόνου εκτέλεσης της είναι δύσκολη, επειδή προσωρινά το αρχείο δεν είναι διαθέσιμο στο χρήστη. Το κόστος της αναδιοργάνωσης ισούται με το κόστος εξαντλητικής ανάγνωσης του αρχείου

συν το κόστος της διαδοχικής εισαγωγής όλων των εγγραφών σε μία νέα έκταση στο δίσκο. Δηλαδή ισχύει:

$$T_{\text{ανδ}} = T_{\text{εξαν}} + n \times T_{\text{εισ}}$$

9.5 Διαχείριση συνωνύμων με ψευδοαλυσίδες

Σύμφωνα με τη μέθοδο αυτή, που προτάθηκε από τους Χαλάτση και Φιλοχύπρου (1978), ο τύπος της εγγραφής περιέχει ένα 'ψευδοδείκτη', δηλαδή ένα επιπλέον πεδίο που σκοπό έχει τη λογική σύνδεση της κάθε εγγραφής με την επόμενη της. Με άλλα λόγια το πεδίο αυτό δεν είναι ένας πραγματικός δείκτης μεγέθους μερικών bytes που δείχνει επακριβώς τη θέση της επόμενης εγγραφής, αλλά αποτελείται από μερικά μόνο bits που με κατάλληλους υπολογισμούς δίνουν τη θέση της επόμενης εγγραφής. Η μέθοδος ονομάζεται και μέθοδος των υπολογιζόμενων αλυσίδων (computed chaining), επειδή γίνονται κάποιοι υπολογισμοί. Εξ άλλου υπενθυμίζεται ότι βασικός κανόνας της οργάνωσης δεδομένων είναι ότι προτιμάται ο υπολογισμός από την αποθήκευση, γιατί το κόστος υπολογισμών είναι μικρότερο από το κόστος αποθήκευσης συν το κόστος εισόδου/εξόδου δεδομένων.

Σε σχέση με τις μεθόδους που ανήκουν στην οικογένεια της ανοικτής υπερχειλίσσης υπάρχει μία επιπλέον βασική διαφορά. Έστω ότι η συνάρτηση κατακερματισμού κατευθύνει την εισαγόμενη εγγραφή σε μία θέση κατειλημμένη. Ελέγχεται αν η εγγραφή που είναι αποθηκευμένη στη θέση αυτή είναι συνώνυμη ή είναι εγγραφή υπερχειλίσσης. Αν είναι συνώνυμη, τότε η εισαγόμενη εγγραφή κατευθύνεται σε μία άλλη θέση που είναι συνάρτηση όχι του κλειδιού της εγγραφής που εισάγεται, αλλά του κλειδιού της εγγραφής που είναι ήδη αποθηκευμένη. Πιο συγκεκριμένα, από την τιμή του κλειδιού προκύπτει με μία συνάρτηση κατακερματισμού μία τιμή i , που δηλώνει ότι η εισαγόμενη εγγραφή θα πρέπει να απομακρυνθεί κατά i θέσεις. Αν η νέα αυτή θέση δεν είναι κατειλημμένη, τότε η εισαγόμενη εγγραφή αποθηκεύεται στη θέση αυτή και ο ψευδοδείκτης παίρνει την τιμή 1, που δηλώνει ότι συνώνυμη εγγραφή υπάρχει μετά $1 \times i$ θέσεις. Αν η νέα θέση είναι κατειλημμένη, τότε η εισαγόμενη εγγραφή απομακρύνεται κατά άλλες i θέσεις, δηλαδή συνολικά $2 \times i$ θέσεις από την αρχική θέση. Γίνεται αντιληπτό ότι αν τελικά αποθηκευθεί η εγγραφή στην τελευταία αυτή θέση τότε ο ψευδοδείκτης θα πάρει την τιμή 2. Αν η εισαγόμενη εγγραφή κατευθυνθεί σε μία θέση που είναι κατειλημμένη από εγγραφή υπερχειλίσσης, τότε η τελευταία παραχωρεί

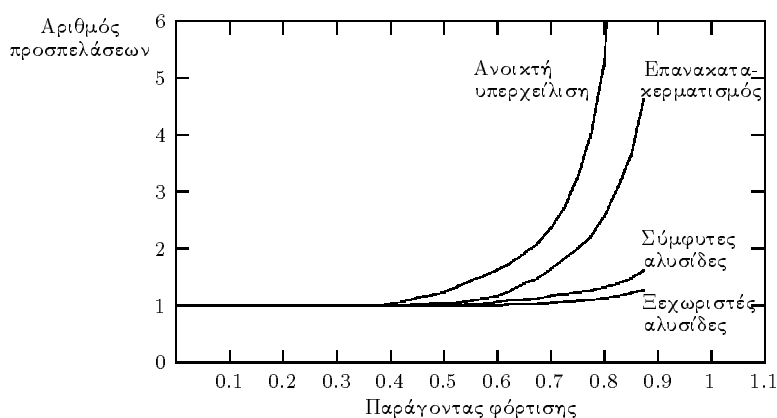
τη θέση στην εισαγόμενη και επανα-εισάγεται μαζί με όλες τις εγγραφές που έπονταν στην αλυσίδα. Στο Σχήμα 9.5 παρουσιάζεται ένα παράδειγμα διαδοχικών εισαγωγών εγγραφών με κλειδιά: 4, 5, 10, 12, 19, 52, 56, 71, 90, και 113, σε αρχείο 10 θέσεων. Τα τρία στιγμιότυπα του σχήματος δίνουν τη δομή μετά την εισαγωγή των εγγραφών 52, 90 και 113 αντίστοιχα. Ως πρώτη συνάρτηση κατακερματισμού χρησιμοποιείται η $key \bmod 10$, ενώ ως δεύτερη χρησιμοποιήθηκε η συνάρτηση $key \div 10$. Σημειώνεται ότι αν από τη συνάρτηση αυτή προκύψει αποτέλεσμα 0, τότε ως αποτέλεσμα θεωρείται το 1.

0	10		10	7	10	7
1			71		71	
2	12	1	12	1	12	6
3	52		52		113	
4	4		4		4	
5	5		5		5	
6			56		56	
7			90		90	
8					52	
9	19		19		19	
	(α)		(β)		(γ)	

Σχήμα 9.5: Μέθοδος ψευδοαλυσίδων.

Η μέθοδος αυτή είναι ένας συμβιβασμός μεταξύ των μεθόδων ανοικτής υπερχειλίσσης και των μεθόδων με χρήση αλυσίδας, τόσο από άποψη χώρου όσο και από άποψη χρονικών επιδόσεων. Σημαντικό πάντως είναι να υπολογισθεί εκ των προτέρων το μέγεθος του ψευδοδείκτη σε bits. Για παράδειγμα για αρχείο 10 θέσεων στη χειρότερη περίπτωση αρκούν ψευδοδείκτες των 4 bits, για αρχείο 100 θέσεων αρκούν 7 bits κοχ. Είναι δυνατόν όμως ο διαθέσιμος αριθμός των bits να είναι σχετικά μικρός. Έτσι αν πρέπει να αποθηκευθεί αριθμός αλμάτων, i , μεγαλύτερος από το διαθέσιμο, τότε απλώς αποθηκεύεται ο διαθέσιμος. Κατά τον τρόπο αυτό επιτυγχάνεται η σύνδεση των εγγραφών, αλλά προφανώς αυξάνει το κόστος αναζήτησης.

Ανακεφαλαιώνοντας για όλες τις μεθόδους, έστω ότι ένας χάδος αποτελείται από μία σελίδα χωρητικότητας μίας εγγραφής, οπότε το χρονικό κόστος αναζήτησης μετράται με προσπελάσεις εγγραφών. Στο Σχήμα 9.6 παρουσιάζεται συγκριτικά η επίδοση των τεσσάρων μεθόδων διαχείρισης των συνωνύμων υπερχειλίσσης. Φαίνεται ότι όταν $Lf > 90\%$, τότε οι μέθοδοι που



Σχήμα 9.6: Επίδοση τεχνικών διαχείρισης συνωνύμων.

χρησιμοποιούν αλυσίδα είναι σημαντικά καλύτερες. Επίσης, αν $Lf > 50\%$, τότε η μη γραμμική αναζήτηση υπερτερεί της ανοιχτής υπερχειλίσης.

Εκτός από το κόστος εισόδου/εξόδου δεδομένων, υπάρχουν και άλλα κόστη που πρέπει να ληφθούν υπ' όψη. Κατ' αρχήν, οι μέθοδοι της ανοιχτής διεύθυνσης έχουν πολύ απλό λογισμικό, γιατί ούτε αποθηκεύουν ούτε διαχειρίζονται δείκτες. Αυτό το γεγονός τείνει να μειώσει το χρόνο υπολογισμών, που ίσως θα μπορούσε να συνεισφέρει σημαντικά στο συνολικό χρόνο. Επίσης, το γεγονός ότι αυτές οι μέθοδοι δεν χρησιμοποιούν δείκτες δεν σημαίνει ότι γίνεται ιδιαίτερα μεγάλη οικονομία από άποψη χώρου. Αυτό οφείλεται στο ότι με τη χρήση των αλυσίδων αντιστοιχεί ένας δείκτης ανά κάδο και συνεπώς το ποσό είναι ίσως αμελητέο. Η κρίσιμη και βασική παράμετρος είναι ο παράγοντας φόρτισης, που πρέπει να διατηρηθεί σε χαμηλά επίπεδα για να μείνει σε λογικά πλαίσια και η επίδοση.

Άρα τελικά, για την κατηγορία των τεχνικών της ανοιχτής διεύθυνσης τα πλεονεκτήματα είναι απλότητα λογισμικού και οικονομία σε χρόνο υπολογισμών, ενώ τα μειονεκτήματα είναι αυξημένοι χρόνοι για είσοδο και έξοδο δεδομένων καθώς και χώρος αποθήκευσης. Αν μπορεί να προβλεφθεί η ανάπτυξη του αρχείου τότε η χρήση ξεχωριστών αλυσίδων είναι μια εξαιρετική τεχνική για αναζήτηση, εισαγωγή ή διαγραφή μίας εγγραφής. Απεναντίας είναι μία πολύ άσχημη επιλογή για σειριακές διεργασίες και ερωτήσεις διαστήματος, όπως συμβαίνει εξ άλλου για όλες τις παραλλαγές του στατικού κατακερματισμού. Συνήθως αμέσως μετά την αρχική φόρτωση, η αναζήτηση τερματίζεται με μία προσπέλαση. Αν είναι γνωστό εκ των

προτέρων ότι οι ερωτήσεις διαστήματος θα γίνονται συχνότερα, τότε ίσως να προτιμηθεί η δομή του B^+ -δένδρου. Πρέπει να σημειωθεί βέβαια ότι η σχετική σπουδαιότητα όλων των προηγούμενων παραγόντων εξαρτάται από το φόρτο εργασίας, τις απαιτήσεις του συστήματος και γενικότερα το υπολογιστικό περιβάλλον.

9.6 Ασκήσεις

<1> Σε στατικό αρχείο κατακερματισμού 15 σελίδων χωρητικότητας μίας εγγραφής να εισαχθούν οι εγγραφές με τις επόμενες τιμές κλειδιών: 42, 57, 16, 52, 66, 77, 12, 25, 21, 33, 32 και 14. Να χρησιμοποιηθούν όλες οι τεχνικές της οικογένειας της ανοικτής διεύθυνσης.

<2> Σε στατικό αρχείο κατακερματισμού 15 σελίδων χωρητικότητας μίας εγγραφής να εισαχθούν οι εγγραφές με τις επόμενες τιμές κλειδιών: 42, 57, 16, 52, 66, 77, 12, 25, 21, 33, 32 και 14. Να χρησιμοποιηθεί η τεχνική των ψευδοαλυσίδων. Πόσα bits είναι απαραίτητα για το μέγεθος του ψευδοδείκτη;

<3> Έστω ότι στον ορισμό του τύπου της εγγραφής περιέχονται δύο ψευδοδείκτες, συνεπώς μπορούν να δημιουργηθούν δύο υπολογιζόμενες αλυσίδες. Με κατακερματισμό στο κλειδί της υπερχειλίζουσας εγγραφής επιλέγεται ποιά αλυσίδα θα ακολουθηθεί ($key \bmod 2$). Να επαναληφθεί η προηγούμενη άσκηση για την παραλλαγή αυτή και να γίνει σύγκριση της επίδοσης κατά την αναζήτηση για την ίδια ομάδα δεδομένων.

<4> Δίνεται στατικό αρχείο κατακερματισμού 5 σελίδων χωρητικότητας 3 εγγραφών. Χρησιμοποιώντας τη συνάρτηση $h(key) = key \bmod 5$ και τη μέθοδο της αλυσίδας να εισαχθούν οι εγγραφές με τις επόμενες τιμές κλειδιών: 42, 57, 16, 52, 66, 77, 12, 25, 21, 33, 32 και 14.

<5> Στατικό αρχείο κατακερματισμού με χρήση αλυσίδας διακρίνεται από τα εξής χαρακτηριστικά: $Bkfr=50$, επιθυμητό $Lf=70\%$, $n=100.000$, $R=400$ bytes. Μετά τη φόρτωση κάθε εγγραφή ανακτάται με μία προσπάθεια στο δίσκο, γιατί το μήκος της αλυσίδας είναι 1. Μετά από καιρό στο αρχείο υπάρχουν συνολικά 400.000 εγγραφές. Αν το μέσο μήκος της αλυσίδας έχει γίνει 3, ποιά είναι η νέα τιμή του παράγοντα φόρτισης. Με πόσες προσπάθειες γίνεται πλέον η αναζήτηση μίας εγγραφής. Συμφέρει ή όχι η δομή αυτή σε σχέση με το B^+ -δένδρο;

<6> Σε αρχείο με 100.000 εγγραφές των 400 bytes γίνονται 10.000 προσπελάσεις και 10 αναζητήσεις διαστήματος που αφορούν στο 3% των εγγραφών. Ποιά δομή πρέπει να προτιμηθεί: ένα αρχείο κατακερματισμού με σειριακό διάβασμα, ένα δευτερεύον B^+ -δένδρο ή ένα πρωτεύον B^+ -δένδρο σχεδιασμένο κατάλληλα για το συγκεκριμένο τύπο ερωτήσεων.

<7> Για το αρχείο της προηγούμενης άσκησης να υπολογισθεί τι θα ήταν προτιμότερο αν σε κάθε 10.000 προσπελάσεις αντιστοιχούσαν 100 ερωτήσεις διαστήματος που αφορούν το 50% των εγγραφών;

<8> Σε αρχείο με 1.000.000 εγγραφές των 400 bytes πρέπει να γίνουν 10.000 προσπελάσεις και 100 ερωτήσεις διαστήματος που απαιτούν το 3% των εγγραφών. Ποιά μέθοδος είναι αποτελεσματικότερη: ένα αρχείο κατακερματισμού με σειριακή ανάγνωση για τις ερωτήσεις διαστήματος ή ένα πρωτεύον B^+ -δένδρο σχεδιασμένο κατάλληλα;

<9> Σε στατικό αρχείο κατακερματισμού με χρήση αλυσίδας το 10% των εγγραφών αναζητάται με πιθανότητα 50% και το υπόλοιπο 90% των εγγραφών επίσης με 50%. Να περιγραφεί μία διαδικασία τοποθέτησης των εγγραφών που να βελτιστοποιεί την επίδοση και να βρεθεί το αντίστοιχο κόστος προσπέλασης.

Κεφάλαιο 10

ΔΥΝΑΜΙΚΑ ΤΥΧΑΙΑ ΑΡΧΕΙΑ

10.1 Εισαγωγή

10.2 Δυναμικός κατακερματισμός

10.3 Επεκτατός κατακερματισμός

10.4 Εκθετικός κατακερματισμός με περιορισμένο
κατάλογο

10.5 Γραμμικός κατακερματισμός

10.6 Ασκήσεις

Κεφάλαιο 10

ΔΥΝΑΜΙΚΑ ΤΥΧΑΙΑ ΑΡΧΕΙΑ

10.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εξετάσθηκαν τα τυχαία στατικά αρχεία. Ο αναλυτής/προγραμματιστής κατά τη φάση σχεδιασμού και φόρτωσης των αρχείων αυτών πρέπει να αποφασίσει πόσος χώρος θα καταληφθεί συνολικά στην κύρια περιοχή και στην περιοχή υπερχείλισης. Σε πολλές εφαρμογές είναι δυνατόν ο αριθμός των εγγραφών να ποικίλει κατά πολύ. Απεναντίας, ο χώρος του αρχείου μένει αμετάβλητος ανεξάρτητα αν το αρχείο έχει πολύ χαμηλή ή πολύ υψηλή πληρότητα, για παράδειγμα 5% ή 95%. Στην πρώτη περίπτωση υπάρχει πρόβλημα αχρησιμοποίητου χώρου, ενώ στη δεύτερη πρόβλημα φτωχής επίδοσης κατά την αναζήτηση. Τα στατικά αρχεία διακρίνονται για αυτές τις αδυναμίες μέχρι του σημείου που θα γίνει αναδιοργάνωση. Κατά την αναδιοργάνωση όλο το αρχείο σχεδιασμένο από την αρχή θα ξαναγραφεί σε μικρότερο ή σε μεγαλύτερο χώρο και μάλιστα σε άλλο μέρος του δίσκου. Ωστόσο, η διαδικασία αυτή είναι ιδιαίτερα χρονοβόρα. Σε μερικές εφαρμογές αυτό μπορεί να είναι απαράδεκτο επειδή το αρχείο δεν είναι διαθέσιμο στο χρήστη κατά το συγκεκριμένο χρονικό διάστημα.

Στο παρελθόν έχουν προταθεί πολλές οργανώσεις τυχαίων αρχείων, που δεν παραμένουν παθητικές στις εισαγωγές και διαγραφές εγγραφών αλλά προσαρμόζονται, ώστε και να μη γίνεται σπατάλη χώρου αλλά και η επίδοση να μην εκφυλίζεται. Στη βιβλιογραφία αναφέρονται περί τις δέκα παραλλαγές τέτοιων τυχαίων αρχείων που λέγονται δυναμικές (dynamic). Στη συνέχεια θα εξετασθούν από αλγοριθμική ή/και αναλυτική άποψη οι εξής

οργανώσεις: ο δυναμικός κατακερματισμός (dynamic hashing), ο επεκτατός κατακερματισμός (extendible hashing), ο εκθετικός κατακερματισμός με περιορισμένο κατάλογο (bounded index exponential hashing), και τέλος ο γραμμικός κατακερματισμός (linear hashing). Κάθε μία από τις προηγούμενες τεχνικές με τα προτερήματα και τα μειονεκτήματα σε σχέση με τις υπόλοιπες απαντά με σύνθετους αλγορίθμους στα προβλήματα:

- πως και πότε διασπάται ένας κάδος,
- πως διανέμονται οι εγγραφές από τον παλιό κάδο στους νέους,
- πως και πότε συγχωνεύονται δύο κάδοι, και τέλος
- πως οι εγγραφές από τους δύο παλιούς κάδους αποδίδονται στο νέο κάδο.

10.2 Δυναμικός κατακερματισμός

Ο δυναμικός κατακερματισμός ήταν η πρώτη χρονικά δομή δυναμικών τυχαίων αρχείων που εμφανίσθηκε στη βιβλιογραφία. Ο ερευνητής που την παρουσίασε, ο Larson (1978), θεωρείται από τους θεμελιωτές της περιοχής αυτής. Ο δυναμικός κατακερματισμός αποτελείται από δύο φυσικά ανεξάρτητες δομές: έναν κατάλογο και ένα κυρίως αρχείο. Ο κατάλογος είναι ένα δάσος δυαδικών δένδρων που υλοποιούνται ως συνδεδεμένες δομές και είναι αποθηκευμένα στην κύρια μνήμη. Στο τελευταίο επίπεδο των δυαδικών δένδρων περιέχονται δείκτες προς τις σελίδες του κυρίως αρχείου που βέβαια είναι αποθηκευμένο στη δευτερεύουσα μνήμη. Ο αριθμός των δυαδικών δένδρων που αποτελούν το δάσος ισούται με τον αριθμό των κάδων, bk , του αρχείου όπως έχει σχεδιασθεί αρχικά. Μία συνάρτηση κατακερματισμού $h_1(key) = key \bmod bk$ από την τιμή του κλειδιού δίνει το δυαδικό δένδρο, όπου είναι αποθηκευμένη η διεύθυνση του κάδου όπου η εγγραφή είναι πραγματικά αποθηκευμένη. Αυτή είναι η γενική φιλοσοφία λειτουργίας της δομής που στηρίζεται στη στρατηγική 'διαίρει και βασίλευε'. Στη συνέχεια αναλύονται δύο ειδικότερα θέματα. Πρώτον, πως μεγεθύνεται ένα αρχείο και, δεύτερον, ποιά είναι η δομή και η επεξεργασία των δυαδικών δένδρων.

Κάθε κάδος από τους bk έχει χωρητικότητα $Bkfr$ εγγραφές, οπότε αν κάποιος από αυτούς δεχθεί την $(Bkfr+1)$ -οστή εγγραφή τότε πρέπει να διασπασθεί. Αυτό σημαίνει ότι ένας νέος κάδος παραχωρείται από το σύστημα

στο αρχείο και οι $(Bkfr+1)$ εγγραφές πρέπει να αναδιανεμηθούν μεταξύ των δύο χάδων. Αυτό επιτυγχάνεται με τη βοήθεια μίας δεύτερης συνάρτησης κατακερματισμού $h_2(key)$ που παίρνοντας ως σπόρο το κλειδί (ή κάποιο μετασχηματισμό του κλειδιού) μπορεί να παράγει μία ψευδοτυχαία δυαδική συμβολοσειρά οποιουδήποτε μήκους, όπου τα 0 και 1 εμφανίζονται ισοπίθानα. Βέβαια αρχικά απαιτούνται μόνο μερικά bits, όμως με τις διαδοχικές διασπάσεις των χάδων χρειάζονται όλο και περισσότερα. Αυτός είναι ο λόγος που αυτή η δεύτερη συνάρτηση κατακερματισμού δεν μετατρέπει απλά το κλειδί στο δυαδικό αντίστοιχο. Έτσι, σύμφωνα με μία σύμβαση, αν το πρώτο bit είναι 0 (αντίστοιχα, 1), τότε η αντίστοιχη εγγραφή κατευθύνεται στον υπάρχοντα (αντίστοιχα, στο νέο) χάδο. Στη γενική περίπτωση, χρησιμοποιούνται τόσα bits της δυαδικής συμβολοσειράς όσα είναι απαραίτητα ώστε οι $Bkfr+1$ εγγραφές να διαμοιρασθούν σε δύο χάδους.

Αρχικά καθένα από τα bk δυαδικά δένδρα του δάσους αποτελείται μόνο από μία ρίζα, οπότε θεωρείται ότι ο κατάλογος έχει μόνο ένα επίπεδο. Όταν γίνει κάποια διάσπαση χάδου, τότε πρέπει το αντίστοιχο δένδρο να επεκταθεί κατά ένα επίπεδο. Προφανώς, δεν συμβαίνει το ίδιο σε όλα τα δένδρα, λέγεται όμως ότι ο κατάλογος επεκτείνεται κατά ένα επίπεδο. Έτσι από τη ρίζα του αντίστοιχου δένδρου δημιουργούνται δύο φύλλα με τους αντίστοιχους δείκτες προς τους δύο χάδους. Κατά σύμβαση, ο αριστερός κόμβος δείχνει στον υπάρχοντα χάδο, ενώ ο δεξιός κόμβος δείχνει στο νέο χάδο. Αυτή η διαδικασία επέκτασης των δυαδικών δένδρων γίνεται ανάλογα με τις εισαγωγές των νέων εγγραφών. Αν η συνάρτηση h_1 διασπείρει τυχαία τις εγγραφές στα δυαδικά δένδρα, τότε τα δένδρα έχουν περίπου το ίδιο μέγεθος, ενώ εκτιμάται ότι τα δένδρα είναι ισοζυγισμένα επειδή η ψευδοτυχαία συνάρτηση h_2 παράγει ισοπίθानα τα 0 και 1.

Είναι ευνόητο ότι οι τύποι των εσωτερικών και των εξωτερικών κόμβων είναι διαφορετικοί. Πιο συγκεκριμένα:

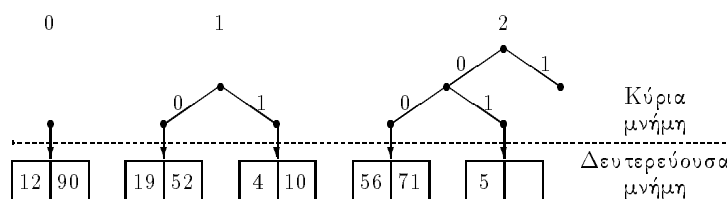
- κάθε εσωτερικός κόμβος αποτελείται από τα εξής τέσσερα πεδία: σημαία, δεικνικός δείκτης προς τον πατέρα κόμβο, δεικνικός δείκτης προς τον αριστερό απόγονο και δεικνικός δείκτης προς τον δεξιό απόγονο,
- κάθε εξωτερικός κόμβος αποτελείται από τα εξής τέσσερα πεδία: σημαία, δεικνικός δείκτης προς τον πατέρα κόμβο, δείκτης προς τον χάδο του αρχείου και μετρητής εγγραφών που είναι αποθηκευμένες στον αντίστοιχο χάδο.

Η σημαία των εσωτερικών και των εξωτερικών κόμβων παίρνει την τιμή 0 και 1, αντίστοιχα.

$R(0)$	$R(1)$	$R(2)$	$R(3)$	$R(4)$	$R(5)$	$R(6)$	$R(7)$	$R(8)$	$R(9)$
1011	0000	0100	0110	1111	0101	0001	1110	1001	0011

Πίνακας 10.1: Γεννήτρια ψευδοτυχαίων δυαδικών αριθμών.

Η προηγούμενη διαδικασία φαίνεται στο παράδειγμα που ακολουθεί. Έστω ότι σε κενό αρχείο δυναμικού κατακερματισμού με κάδους μεγέθους δύο εγγραφών πρόκειται να εισαχθούν διαδοχικά εγγραφές με τιμές κλειδιών 4, 5, 10, 12, 19, 52, 56, 71 και 90. Αρχικά το αρχείο σχεδιάζεται ώστε να έχει τρεις κάδους, άρα πρέπει ως πρώτη συνάρτηση κατακερματισμού να χρησιμοποιηθεί η συνάρτηση $h_1 = \text{key} \bmod 3$ που χωρίζει το δάσος του καταλόγου σε τρία διακριτά δυαδικά δένδρα. Πρέπει επίσης να χρησιμοποιηθεί και μία δεύτερη συνάρτηση για το χτίσιμο των δυαδικών δένδρων. Έστω, λοιπόν, ότι επιλέγεται η συνάρτηση $\text{key} \bmod 10$, ώστε από το κλειδί να προκύψει υπόλοιπο από 0 ως 9. Κάθε τιμή του υπολοίπου μπορεί να θεωρηθεί ως σπόρος σε μία ψευδοτυχαία γεννήτρια, $R()$, που να παράγει δυαδικά ψηφία 0 και 1 με την ίδια πιθανότητα (δηλαδή 50%). Το αποτέλεσμα μίας τέτοιας γεννήτριας για κάθε σπόρο φαίνεται στον Πίνακα 10.1. Στο Σχήμα 10.1 φαίνεται το τελικό αποτέλεσμα της εισαγωγής των εγγραφών με τη δημιουργία του καταλόγου των τριών δένδρων και του αρχείου των πέντε κάδων.



Σχήμα 10.1: Αρχείο δυναμικού κατακερματισμού.

Με το παράδειγμα αυτό φαίνεται ότι η επιτυχής αναζήτηση κοστίζει μία μόνο προσπέλαση στο δίσκο. Επίσης κατά την αναζήτηση μη υπάρχοντος κλειδιού μπορεί να βρεθεί φύλλο του δυαδικού δένδρου με μετρητή εγγραφών

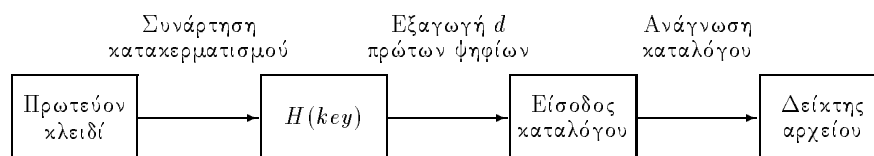
ίσο με μηδέν. Άρα στην περίπτωση αυτή το αντίστοιχο χρονικό κόστος ανεπιτυχούς αναζήτησης θα είναι μηδέν από την άποψη των προσπελάσεων στο δίσκο. Για παράδειγμα, έστω ότι γίνεται αναζήτηση της εγγραφής με τιμή κλειδιού 68. Για την αναζήτηση αυτή θα εξετασθεί το τελευταίο δυαδικό δένδρο (επειδή: $68 \bmod 3 = 2$), και ειδικότερα το δεξιό παιδί της ρίζας (επειδή: $60 \bmod 10 = 8$ και $R(8) = 1001$, το οποίο αρχίζει από 1). Παρατηρούμε, λοιπόν, ότι από το συγκεκριμένο φύλλο δεν υπάρχει δείκτης προς το δίσκο, άρα το κόστος αυτής της ανεπιτυχούς αναζήτησης είναι 0.

Ωστόσο, γίνεται αντιληπτό ότι λόγω των εισαγωγών ο κατάλογος αυξάνει βαθμιαία, οπότε μπορεί το μέγεθός του να ξεπεράσει το μέγεθος της διαθέσιμης κύριας μνήμης. Έτσι σε κάποια χρονική στιγμή η επιτυχής αναζήτηση θα κοστίζει δύο προσπελάσεις στο δίσκο γιατί τμήμα του καταλόγου θα πρέπει να αποθηκευθεί στη δευτερεύουσα μνήμη. Σε σχέση με τον παράγοντα χρησιμοποίησης χώρου έχει αποδειχθεί αναλυτικά ότι η μέση τιμή του είναι 69%. Η τιμή αυτή ισούται με την αντίστοιχη για τις δομές της οικογένειας των B-δένδρων και θα συναντηθεί και πάλι στη συνέχεια του κεφαλαίου αυτού.

Από τον Scholl (1981) έχουν προταθεί δύο παραλλαγές του δυναμικού κατακερματισμού για τη βελτίωση της επίδοσης της αναζήτησης. Σύμφωνα με μία παραλλαγή από αυτές, δεν γίνεται διάσπαση του κάδου όταν σ' αυτόν κατευθυνθεί η $(Bkfr+1)$ -οστή εγγραφή αλλά δημιουργείται αλυσίδα υπερχειλίσσης με ένα δεύτερο κάδο. Το ζεύγος αυτό των κάδων θα διασπασθεί και ο κατάλογος θα ενημερωθεί αντίστοιχα όταν θα έχουν εισαχθεί $\beta \times Bkfr$ εγγραφές, όπου $1 \leq \beta \leq 2$. Η τεχνική αυτή λέγεται **αναβολή διάσπασης** (deferred splitting). Έτσι βέβαια για μερικές περιπτώσεις θα χρειάζεται μία προσπέλαση στο δίσκο, ενώ για άλλες περιπτώσεις θα χρειάζονται δύο προσπελάσεις. Το πλεονέκτημα όμως είναι ότι ο κατάλογος θα είναι μικρότερος κατά β φορές και θα χωρά στην κύρια μνήμη. Αν το αρχείο μεγαλώσει υπερβολικά τότε μπορεί δίνοντας στο β μεγαλύτερες τιμές να επιτραπουν αλυσίδες υπερχειλίσσης μεγαλύτερου μήκους, ώστε ο κατάλογος να χωρά οπωσδήποτε στην κύρια μνήμη. Εύκολα συμπεραίνεται ότι η μέση τιμή προσπελάσεων στο δίσκο στη χειρότερη περίπτωση θα είναι $\lceil \beta \rceil$. Ο ενδιαφερόμενος αναγνώστης μπορεί από την αναφορά να βρει περισσότερες λεπτομέρειες για τις παραλλαγές του δυναμικού κατακερματισμού.

10.3 Επεκτατός κατακερματισμός

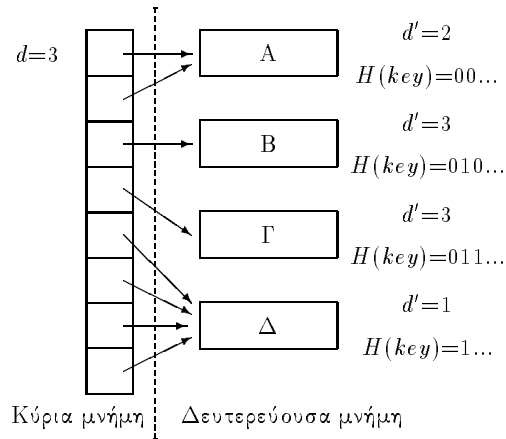
Η δομή αυτή προτάθηκε από τους Fagin et al. (1979) και ήταν η χρονικά δεύτερη υλοποίηση δυναμικού τυχαίου αρχείου. Η αλληλουχία των μετασχηματισμών του κλειδιού που εκτελούνται στον επεκτατό κατακερματισμό φαίνεται στο Σχήμα 10.2. Ο πρώτος μετασχηματισμός είναι μία συνάρτηση κατακερματισμού που απεικονίζει κατά τυχαίο τρόπο ένα διάστημα κλειδιών σε ένα σταθερό διάστημα διευθύνσεων. Ο μόνος περιορισμός είναι ότι το διάστημα διευθύνσεων πρέπει να είναι δύναμη του δύο. Επειδή όμως προτιμάται το διάστημα διευθύνσεων να είναι πρώτος αριθμός, γι' αυτό τελικά επιλέγεται ο μεγαλύτερος ακέραιος που είναι αμέσως μικρότερος από τη δυαδική δύναμη. Για παράδειγμα, ο μεγαλύτερος πρώτος αριθμός που είναι μικρότερος από το 2^{16} είναι ο 65521.



Σχήμα 10.2: Μετασχηματισμοί κλειδιού στον επεκτατό κατακερματισμό.

Στη συνέχεια, η τιμή του κλειδιού μετατρέπεται στον ισοδύναμο δυαδικό αριθμό και λαμβάνονται τα πρώτα ψηφία του, δηλαδή κάποια bits. Είναι δυνατόν επίσης να μη ληφθούν τα πρώτα bits αλλά τα τελευταία ή κάποια μεσαία, όπως συμβαίνει σε πολλές παραλλαγές της μεθόδου που συναντώνται στη βιβλιογραφία. Τα bits αυτά λαμβάνονται ως είσοδος σε κατάλογο, που περιέχει δείκτες προς τους κάδους του αρχείου. Ο κατάλογος αυτός είναι ένας μονοδιάστατος πίνακας με 2^d στοιχεία, όπου d είναι ο αριθμός των ψηφίων που επιλέγονται από το αποτέλεσμα της συνάρτησης κατακερματισμού και λέγεται βάθος (depth) ή επίπεδο (level) του καταλόγου.

Ο αριθμός των ψηφίων που εξάγονται από το αποτέλεσμα της συνάρτησης κατακερματισμού μεταβάλλεται χρονικά ανάλογα με τη μεταβολή του μεγέθους του αρχείου. Για το λόγο αυτό, το επίπεδο του καταλόγου είναι μία απαραίτητη παράμετρος. Ένα παράδειγμα παρουσιάζεται στο Σχήμα 10.3, όπου έχουν χρησιμοποιηθεί τα πρώτα τρία ψηφία του μετασχηματισμού, και συνεπώς ο πίνακας αποτελείται από οκτώ δείκτες. Αυτοί οι οκτώ δείκτες μπορούν να αναφερθούν σε οκτώ το πολύ κάδους. Έστω, λοιπόν, ότι η δυαδική μορφή του μετασχηματισμού ενός κλειδιού είναι



Σχήμα 10.3: Κατάλογος επιπέδου 3 με τέσσερις κάδους.

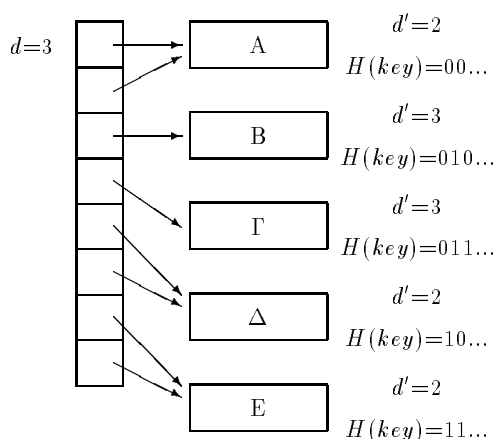
0110100101100101. Από αυτόν τον αριθμό απομονώνονται τα τρία πρώτα bits που ισοδυναμούν με το δεκαδικό αριθμό τρία. Άρα, με προσπέλαση στην υπ' αριθμό 3 είσοδο του πίνακα (που ουσιαστικά είναι η τέταρτη είσοδος του πίνακα) βρίσκεται ο κατάλληλος δείκτης που αναφέρεται στον κάδο Γ. Επίσης αξ προσεχθεί ότι κάθε κάδος συνοδεύεται από μία παράμετρο, την d' , που λέγεται βάθος (depth) ή επίπεδο (level) του κάδου. Η παράμετρος αυτή δηλώνει τον αριθμό των bits που είναι κοινός για τα κλειδιά όλων των εγγραφών του κάδου και είναι πάντα μικρότερη ή ίση από το βάθος του καταλόγου d . Συνεπώς, ο αριθμός των δεικτών που αναφέρονται σε ένα δεδομένο κάδο είναι $2^{d-d'}$.

Ανακεφαλαιώνοντας, η αναζήτηση στον επεκτατό κατακερματισμό γίνεται με πέντε βήματα. Πρώτον, εφαρμόζεται μία συνάρτηση κατακερματισμού στο κλειδί. Δεύτερον, εξάγονται τα πρώτα d bits. Τρίτο, χρησιμοποιείται ο κατάλογος για να εντοπισθεί ο κατάλληλος δείκτης. Τέταρτο, με βάση το δείκτη γίνεται προσπέλαση στον κάδο. Πέμπτο, γίνεται αναζήτηση μέσα στον κάδο για τον εντοπισμό του κλειδιού.

Η επεξεργασία των εισαγωγών και των διαγραφών εγγραφών είναι αρκετά πολύπλοκη. Ένας βασικός κανόνας είναι ότι η περιεκτικότητα ενός κάδου δεν μπορεί να είναι μεγαλύτερη από 100%, και δεν πρέπει να είναι μικρότερη από ένα ποσοστό που ορίζεται από το χρήστη, για παράδειγμα 50%. Όταν αυτός ο κανόνας παραβιάζεται τότε αλλάζει η δομή του αρχείου.

Έστω ότι στον κάδο Δ του αρχείου του Σχήματος 10.3 πρόκειται να

εισαχθεί μία εγγραφή. Αν ο κάδος Δ είναι ήδη πλήρης, τότε η νέα εγγραφή δεν μπορεί να αποθηκευθεί εκεί και προκαλείται επέκταση του αρχείου. Έτσι, ένας νέος κάδος E προστίθεται στο αρχείο. Οι μισοί από τους δείκτες που πριν αναφέρονταν στον κάδο Δ , τώρα αλλάζουν και αναφέρονται στον κάδο E . Οι αντίστοιχες εγγραφές μεταφέρονται από τον κάδο Δ στον κάδο E . Έτσι τελικά, οι δύο κάδοι Δ και E έχουν περιεκτικότητα 50% και υπάρχει αρκετός χώρος για μελλοντικές εισαγωγές. Η τελική μορφή του αρχείου φαίνεται στο Σχήμα 10.4. Ας προσεχθεί ότι πριν τη διάσπαση στον κάδο Δ βρίσκονταν όλες οι εγγραφές των οποίων ο μετασχηματισμός του κλειδιού άρχιζε από 1... Μετά τη διάσπαση στον κάδο Δ και στον κάδο E αποθηκεύονται όλες οι εγγραφές των οποίων ο μετασχηματισμός του κλειδιού αρχίζει από 10... και 11... αντίστοιχα.

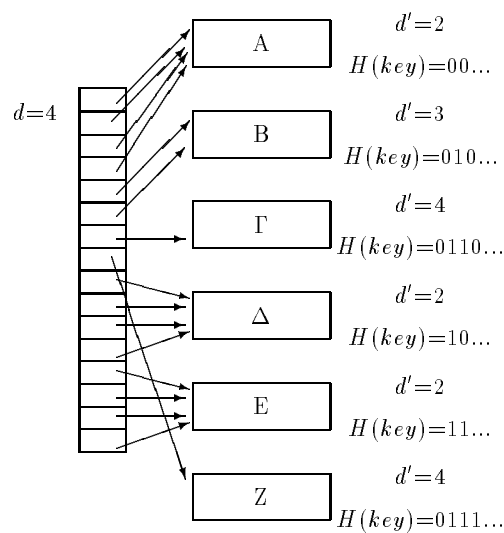


Σχήμα 10.4: Αναδιανομή κλειδιών λόγω διάσπασης κάδου.

Η διάσπαση των κάδων μπορεί να συνεχισθεί με τις διαδοχικές εισαγωγές νέων εγγραφών μέχρι ενός ορισμένου σημείου. Όταν ο κάδος που πρόκειται να διασπασθεί αναφέρεται από ένα μόνο δείκτη, δηλαδή $d = d'$, τότε ο κατάλογος πρέπει να επεκταθεί. Έστω ότι στον ήδη πλήρη κάδο Γ του αρχείου του Σχήματος 10.4 εισάγεται μία νέα εγγραφή. Τότε ένας νέος κάδος Z αποδίδεται στο αρχείο. Ωστόσο, ο κατάλογος δεν διαθέτει χώρο για να αποθηκευθούν οι σχετικοί δείκτες. Άρα, στο σημείο αυτό πρέπει να μεγεθυνθεί ο κατάλογος.

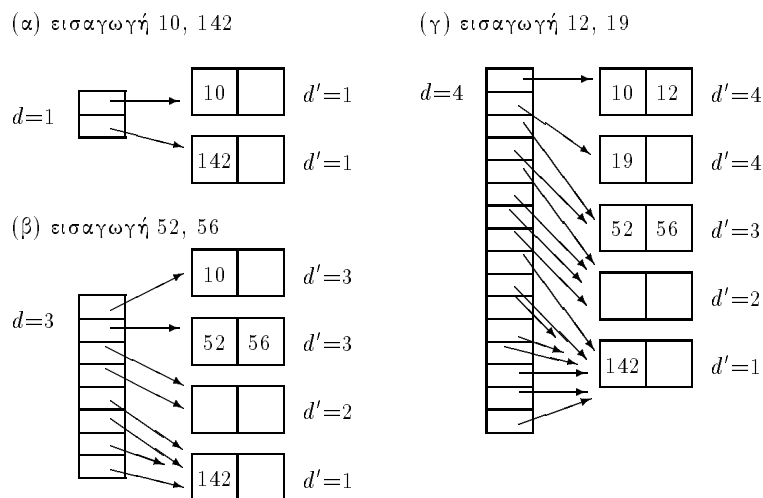
Σε κάθε διαδοχική μεγέθυνση ο κατάλογος διπλασιάζεται με αύξηση της παραμέτρου d κατά μία μονάδα. Κάθε δείκτης του αρχικού καταλόγου καταλαμβάνει δύο θέσεις του νέου καταλόγου. Για παράδειγμα, ο κατάλογος του

Σχήματος 10.4 περιέχει δείκτες που αναφέρονται κατά σειρά στους κάδους A, A, B, Γ, Δ, Δ, E και E. Μετά την επέκταση του καταλόγου οι δείκτες αναφέρονται κατά σειρά στους κάδους A, A, A, A, B, B, Γ, Γ, Δ, Δ, Δ, Δ, E, E, E και E. Με το διπλασιασμό του καταλόγου ο κάδος Γ αναφέρεται πλέον από δύο δείκτες και συνεπώς μπορεί να προχωρήσει η διάσπαση. Έτσι, ο ένας από αυτούς τους δύο δείκτες στο εξής αναφέρεται στον κάδο Z, ενώ παράλληλα οι εγγραφές μοιράζονται στους δύο κάδους ανάλογα με τα πρώτα τέσσερα ψηφία από το αποτέλεσμα της συνάρτησης κατακερματισμού. Η νέα μορφή του αρχείου παρουσιάζεται στο Σχήμα 10.5.



Σχήμα 10.5: Διπλασιασμός καταλόγου.

Ο κατάλογος διπλασιάζεται κάθε φορά που διασπάται ένας κάδος, για τον οποίο ισχύει η σχέση $d = d'$, ανεξάρτητα από κατάσταση την άλλων κάδων. Αυτή η τεχνική μπορεί να οδηγήσει σε παθολογικές καταστάσεις. Για παράδειγμα, στο Σχήμα 10.6 παρουσιάζεται ένα αρχείο με δύο κάδους χωρητικότητας δύο εγγραφών, όπου αρχικά εισάγονται δύο εγγραφές με κλειδιά 10 ($= 00001010_2$) και 142 ($= 10001110_2$). Στη συνέχεια εισάγονται οι εγγραφές με κλειδιά 52 ($= 00110100_2$) και 56 ($= 00111000_2$), οπότε γίνονται δύο διαδοχικοί διπλασιασμοί του μεγέθους του καταλόγου, επειδή τα κλειδιά 10, 52 και 56 έχουν κοινά τα πρώτα δύο bits. Διπλασιασμός του καταλόγου γίνεται και με την εισαγωγή των εγγραφών με κλειδιά 12 ($= 00001100_2$) και 19 ($= 00010011_2$). Έτσι ο κατάλογος έχει $2^4 = 16$



Σχήμα 10.6: Παθολογική περίπτωση διπλασιασμού καταλόγων.

εισόδους, οπότε υπάρχουν αρκετοί διακριτοί δείκτες που δείχνουν είτε σε ένα κενό κάδο είτε σε ένα κοινό κάδο.

Εξ αιτίας διαδοχικών διαγραφών εγγραφών μπορεί να προκληθεί η αντίστροφη δυαδική διαδικασία της συρρίκνωσης του αρχείου. Δύο κάδοι συσσωματώνονται σε έναν κάδο κάτω από τρεις προϋποθέσεις:

- η μέση περιεκτικότητα των δύο κάδων δεν ξεπερνά το 50%, γιατί σε αντίθετη περίπτωση, δεν θα υπήρχε χώρος σε έναν κάδο για όλες τις εγγραφές,
- οι κάδοι που πρόκειται να συνδυασθούν χαρακτηρίζονται από την ίδια τιμή της παραμέτρου d' , και
- τα κλειδιά των εγγραφών των δύο κάδων έχουν κοινά τα πρώτα $(d'-1)$ ψηφία του αποτελέσματος του μετασχηματισμού κατακερματισμού.

Οι δύο τελευταίες προϋποθέσεις είναι απαραίτητες, έτσι ώστε όλες οι εγγραφές του νέου κάδου να έχουν κοινά τα πρώτα d' ψηφία. Έστω για παράδειγμα, ότι η πρώτη συνθήκη συντρέχει για τους κάδους Α και Β του Σχήματος 10.5. Οι κάδοι αυτοί, όμως, δεν μπορούν να συγχωνευθούν γιατί τα κλειδιά των εγγραφών δεν έχουν τον ίδιο αριθμό κοινών ψηφίων. Απεναντίας, οι κάδοι Δ και Ε μπορούν να συγχωνευθούν γιατί ικανοποιούνται όλες οι συνθήκες.

Κατά τον ίδιο τρόπο, όταν όλοι οι δείκτες είναι κατά ζεύγη μπορεί να υποδιπλασιασθεί και το μέγεθος του καταλόγου. Για παράδειγμα, αν υποτεθεί ότι οι δείκτες του καταλόγου αναφέρονται κατά σειρά στους κάδους A, A, A, A, B, B, Γ, Γ, Δ, Δ, Δ, Δ, E, E, Z και Z, τότε με τη συρρίκνωση του καταλόγου οι δείκτες θα αναφέρονται στους κάδους A, A, B, Γ, Δ, Δ, E και Z. Αντίθετα, αν οι δείκτες αφορούσαν στους κάδους A, A, A, B, B, B, Γ, Γ, Δ, Δ, Δ, Δ, E, E, Z και Z δεν θα μπορούσε να γίνει υποδιπλασιασμός του καταλόγου.

Ας σημειωθεί ότι με μία άλλη απλή υλοποίηση του καταλόγου μπορεί να μην υπάρχουν περιττοί δείκτες. Δηλαδή, αν για δεδομένο επίπεδο d του καταλόγου και για κάποιο συνδυασμό d ψηφίων δεν υπάρχουν αντίστοιχες τιμές κλειδιών, τότε ο αντίστοιχος δείκτης έχει τιμή NIL. Η παραλλαγή αυτή έχει ως αποτέλεσμα ταχύτερη ανεπιτυχή αναζήτηση και απλούστερη διαδικασία διάσπασης κάδων. Ταυτόχρονα όμως έχει μεγαλύτερες απαιτήσεις χώρου γιατί θα χρειάζονται πολλοί κάδοι με λίγες εγγραφές ο καθένας και επομένως όσο μεγαλύτερος ο κάδος τόσο περισσότερος και ο μη χρησιμοποιημένος χώρος.

Θεωρητικά, είναι πιθανό όταν η περιεκτικότητα ενός κάδου πέσει κάτω από 50%, να μην υπάρχει κάποιος άλλος κάδος κατάλληλος ώστε οι δύο τους να συγχωνευθούν σε ένα νέο κάδο. Δηλαδή, δεν υπάρχει εγγύηση ότι η περιεκτικότητα κάθε κάδου και του αρχείου συνολικά είναι τουλάχιστον 50%. Αν

- ο αριθμός των εγγραφών, n , είναι μεγάλος,
- η χωρητικότητα των κάδων είναι επίσης μεγάλη, και
- τα κλειδιά διαμοιράζονται ομοιόμορφα μεταξύ των κάδων,

τότε αποδεικνύεται ότι είναι μηδαμινή η πιθανότητα να πέσει κάτω από 50% η περιεκτικότητα ενός κάδου. Η τελευταία υπόθεση σημαίνει ότι η χωρητικότητα των κάδων είναι περίπου η ίδια, άρα όλοι οι κάδοι διασπώνται περίπου ταυτόχρονα. Σε κάποια χρονική στιγμή, λοιπόν, κάποιοι κάδοι θα έχουν περιεκτικότητα περί το 100%, ενώ άλλοι θα έχουν περιεκτικότητα περί το 50%.

Η μέση περιεκτικότητα των κάδων είναι περίπου 69%, άρα η μέση τιμή του αριθμού των κάδων του αρχείου είναι $\frac{n}{Bkfrx \ln 2}$. Η μέση τιμή του αριθμού των εισόδων του καταλόγου είναι $\frac{n}{Bkfrx (\ln 2)^2}$. Από την τελευταία σχέση

φαίνεται ότι όταν το αρχείο διογκωθεί αρκετά, αντίστοιχα διογκώνεται και ο κατάλογος. Σε εξαιρετικές περιπτώσεις ο κατάλογος μπορεί να γίνει τόσο μεγάλος, ώστε να μην χωρά στην κύρια μνήμη, οπότε θα πρέπει να αποθηκευθεί στη δευτερεύουσα μνήμη και θα απαιτούνται επιπλέον προσπελάσεις στο δίσκο για επεξεργασία του πίνακα. Αυτό αποτελεί ένα σοβαρό μειονέκτημα της μεθόδου. Αυτό το πρόβλημα επιλύεται σε μία παραλλαγή της μεθόδου που λέγεται εκθετικός κατακερματισμός με περιορισμένο κατάλογο και παρουσιάζεται στη συνέχεια. Ένα άλλο μειονέκτημα της μεθόδου είναι ότι δεν προσφέρεται για ερωτήσεις διαστήματος, όπως για παράδειγμα η δομή του B^+ -δένδρου.

10.4 Εκθετικός κατακερματισμός με περιορισμένο κατάλογο

Βασικό μειονέκτημα της προηγούμενης μεθόδου είναι ο ανεξέλεγκτος διπλασιασμός του μεγέθους του καταλόγου. Ένα άλλο μειονέκτημα της μεθόδου είναι ότι οι κάδοι διασπώνται περίπου ταυτόχρονα, επειδή οι εγγραφές κατανέμονται ισοπίθانا στους κάδους. Επομένως ενώ το κόστος της εισαγωγής εγγραφών διατηρείται περίπου σταθερό, περιοδικά το κόστος αυτό έχει εξάρσεις που οφείλονται στις πολλές και ταυτόχρονες διασπάσεις των κάδων του αρχείου και του καταλόγου. Η οργάνωση του εκθετικού κατακερματισμού με περιορισμένο κατάλογο, που προτάθηκε από το Lomet (1983), επιλύει και τα δύο προβλήματα.

Όσον αφορά στο πρώτο μειονέκτημα, ο κατάλογος δεν μπορεί να ξεπεράσει ένα μέγιστο μέγεθος που καθορίζεται από τη διαθέσιμη κύρια μνήμη. Δηλαδή, ο αριθμός των σελίδων που συμπεριλαμβάνονται σε έναν κατάλογο είναι δύναμη του δύο και άρα διαδοχικά διπλασιάζεται μέχρι του σημείου να φθάσει στο όριο των 2^{max} σελίδων. Για παράδειγμα, για μεγάλα αρχεία το μέγεθος αυτό θα μπορούσε να είναι 64, 128 κοχ. σελίδες. Έτσι ο κατάλογος διπλασιάζεται μέχρι να φθάσει αυτό το σταθερό οριακό μέγεθος και επομένως παραμένει σταθερός και ο αριθμός των κάδων που δεικτοδοτούνται από τον κατάλογο. Έτσι, μετά από αυτό το σημείο το αρχείο μεγαλώνει αυξάνοντας όχι τον αριθμό των κάδων αλλά το μέγεθος τους.

Όσον αφορά στο δεύτερο μειονέκτημα, ο μετασχηματισμός του κλειδιού σε τέτοια αρχεία ακολουθεί έναν αριθμό βημάτων. Αρχικά, τα κλειδιά

μετασχηματίζονται με βάση μία συνάρτηση ώστε να προκύψει μία ομοιόμορφη κατανομή. Ύστερα, το αποτέλεσμα της συνάρτησης ανακατανέμεται εκθετικά με βάση τον τύπο:

$$exhash(k) = 2^{h(k)} - 1$$

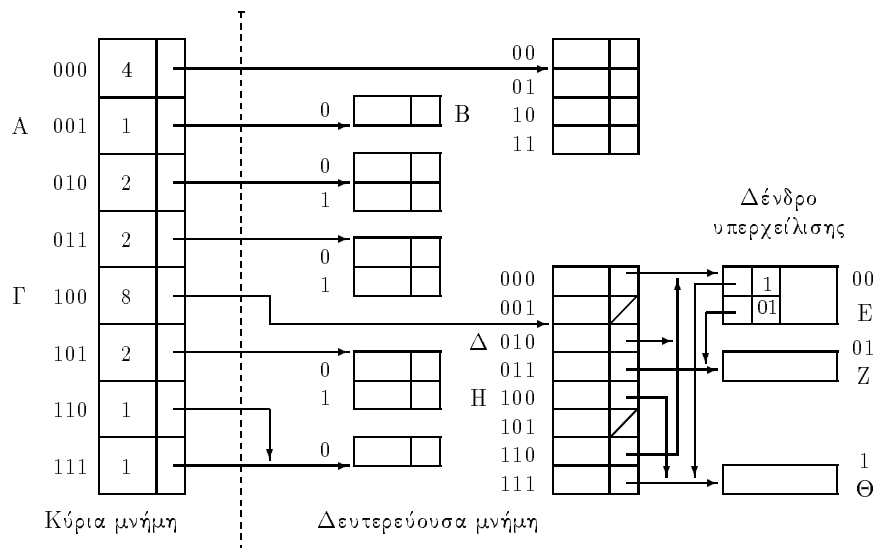
όπου k είναι το κλειδί, $h(k)$ είναι το αποτέλεσμα του πρώτου μετασχηματισμού, και $exhash(k)$ είναι το αποτέλεσμα του δεύτερου. Τελικό αποτέλεσμα είναι η ομοιόμορφη κατανομή να αντικατασταθεί με μία άλλη που διακρίνεται για τη συσσώρευση των κλειδιών προς το κάτω άκρο του διαστήματος των διευθύνσεων. Έτσι οι κάδοι δεν δέχονται τον ίδιο αριθμό εγγγραφών και η επίδοση της εισαγωγής είναι περίπου σταθερή σε όλη τη διάρκεια της ζωής του αρχείου. Στον Πίνακα 10.2 παρατηρούμε ότι η απόσταση μεταξύ διαδοχικών τιμών της $exhash(k)$ μεγαλώνει καθώς οι τιμές της $h(k)$ αυξάνονται. Για παράδειγμα, αν και η απόσταση μεταξύ διαδοχικών τιμών της $h(k)$ είναι 0,067, εν τούτοις η απόσταση των πρώτων δύο τιμών της $exhash(k)$ είναι 0,47, ενώ η απόσταση των τελευταίων τιμών της $exhash(k)$ είναι 0,90.

$h(k)$	$exhash(k)$	$h(k)$	$exhash(k)$	$h(k)$	$exhash(k)$
0,000	0,000	0,400	0,320	0,800	0,741
0,067	0,047	0,467	0,382	0,867	0,823
0,133	0,097	0,533	0,447	0,933	0,910
0,200	0,149	0,600	0,516	1,000	1,000
0,267	0,203	0,667	0,662		
0,333	0,260	0,733	0,662		

Πίνακας 10.2: Εφαρμογή εκθετικού μετασχηματισμού.

Τα πρώτα bits του $exhash(k)$ καθορίζουν τη σελίδα του καταλόγου, όπου είναι αποθηκευμένος ο δείκτης προς τον κατάλληλο κάδο. Η επιλογή της σελίδας (από το σύνολο των σελίδων του κάδου), όπου θα πρέπει να συνεχισθεί η αναζήτηση, γίνεται με βάση μερικά από τα επόμενα bits από το $exhash(k)$. Ακόμη σε κάθε κάδο αντιστοιχεί μία περιοχή υπερχείλισης, που είναι οργανωμένη ως δυαδικό δένδρο που ονομάζεται ο-δένδρο (από το overflow). Ο κατάλογος για αυτό το δένδρο αποθηκεύεται στην πρώτη σελίδα του αντίστοιχου κάδου. Κατά παρόμοιο τρόπο, η διαχείριση της περιοχής υπερχείλισης γίνεται με τα επόμενα bits του $exhash(k)$.

Οι προηγούμενες έννοιες διασαφηνίζονται με το παράδειγμα του Σχήματος 10.7. Αριστερά φαίνεται ο κατάλογος που έχει φθάσει στα όρια της



Σχήμα 10.7: Εκθετικός κατακερματισμός με περιορισμένο κατάλογο.

ανάπτυξής του, επειδή $max=3$. Ο κατάλογος περιέχει δείκτες προς τους κύριους κάδους. Οι δύο τελευταίοι δείκτες αναφέρονται στον ίδιο κάδο, γεγονός που άλλωστε είναι δυνατό και στη μέθοδο του απλού επεκτατού κατακερματισμού.

Ας υποθεθεί ότι πρέπει να αναζητηθεί το κλειδί με τιμή 204. Έστω ότι $exhash(204) = (.0011001100_2)$. Επειδή ο κατάλογος αποτελείται από οκτώ σελίδες, θεωρούνται τα τρία πρώτα bits για την προσπέλαση του καταλόγου, δηλαδή τα 001. Άρα η αναζήτηση κατευθύνεται στη δεύτερη σελίδα του καταλόγου, τη σελίδα A. Επειδή η είσοδος του μεγέθους στον κατάλογο είναι 1, ο αντίστοιχος δείκτης αναφέρεται σε κάδο με μία μόνο σελίδα. Άρα τελικά, η αναζήτηση τερματίζεται στον κάδο B του Σχήματος 10.7.

Στη συνέχεια, ας υποθεθεί ότι πρέπει να αναζητηθεί το κλειδί 641. Κατά τον ίδιο τρόπο, έστω ότι ισχύει $exhash(641) = (.1000101100_2)$. Τα τρία πρώτα bits είναι τα 100, άρα η αναζήτηση κατευθύνεται στην πέμπτη σελίδα του καταλόγου, τη σελίδα Γ. Επειδή η είσοδος του μεγέθους στον κατάλογο είναι 3, ο αντίστοιχος δείκτης αναφέρεται σε κάδο με οκτώ σελίδες και συνεπώς πρέπει να χρησιμοποιηθούν τα τρία επόμενα bits για τη διευκρίνιση της συγκεκριμένης σελίδας, όπου πρέπει να συνεχισθεί η αναζήτηση. Αυτά τα bits είναι 010 και συνεπώς η αναζήτηση συνεχίζεται στην τρίτη σελίδα,

τη σελίδα Δ. Αν η σελίδα Δ δεν περιέχει το κλειδί, τότε η αναζήτηση πρέπει να συνεχισθεί στο ο-δένδρο. Στη σελίδα Δ υπάρχει δείκτης που αναφέρεται στην πρώτη σελίδα του δένδρου, τη σελίδα Ε του καταλόγου του δένδρου. Αν το τέταρτο και το πέμπτο bit είναι 00 ή 01, τότε το κλειδί περιέχεται στη σελίδα Ε ή στη σελίδα Ζ, αντίστοιχα. Ειδικά, αν το τέταρτο bit είναι 1, τότε η αναζήτηση συνεχίζεται στη σελίδα Θ. Έτσι, στην περίπτωση αναζήτησης του κλειδιού 641 απαιτούνται τρεις προσπελάσεις στο δίσκο, δηλαδή διαδοχικά στις σελίδες Δ, Ε και Θ. Αν παρουσιασθεί το φαινόμενο να χρειασθούν περισσότερο από δύο προσπελάσεις, τότε οι δείκτες αλλάζουν ώστε να συντομευθεί η διαδικασία. Έτσι ο δείκτης της σελίδας Δ δεν θα δείχνει πλέον στη σελίδα Ε, αλλά στη σελίδα Ζ.

Τέλος, ας υποθεθεί ότι πρέπει να αναζητηθεί το κλειδί 670. Κατά τον ίδιο τρόπο, ισχύει $exhash(670) = (.1001001011_2)$. Τα τρία πρώτα bits είναι τα 100, άρα η αναζήτηση και πάλι κατευθύνεται στη σελίδα Γ. Τα τρία επόμενα bits είναι 100 και συνεπώς η αναζήτηση συνεχίζεται στην πέμπτη σελίδα που ονομάζεται Η. Αν το κλειδί δεν είναι αποθηκευμένο στη σελίδα Η, τότε η αναζήτηση πρέπει να συνεχισθεί στο ο-δένδρο. Στη σελίδα Η υπάρχει δείκτης που αναφέρεται στη σελίδα Θ, που είναι ένα φύλλο του ο-δένδρου. Δηλαδή, τελικά το κλειδί 670 βρίσκεται ή στη σελίδα Η ή στη σελίδα Θ.

Όπως αναφέρθηκε όταν ένας κάδος γεμίσει, τότε το μέγεθος του διπλασιάζεται και ενημερώνεται κατάλληλα η αντίστοιχη είσοδος του καταλόγου. Ο κάδος μεταφέρεται από το συγκεκριμένο σημείο του δίσκου και ξαναγράφεται σε κάποια άλλη περιοχή του δίσκου ώστε να μην χρειάζονται δύο προσπελάσεις. Μία άλλη συνηθισμένη τεχνική είναι να αφήνονται εγγραφές στην υπερχειλίση παρά να διπλασιάζεται συνεχώς το μέγεθος του κάδου, οπότε με αυξημένο κόστος αναζήτησης επιτυγχάνεται καλύτερη χρήση του χώρου. Σε σχέση με την απλή μέθοδο του επεκτατού κατακερματισμού υπάρχουν τα εξής πλεονεκτήματα: δεν χρειάζεται προσπέλαση στο δίσκο για τον κατάλογο, ενώ απαιτείται μόνο μία προσπέλαση στο δίσκο αν δεν υπάρχει υπερχειλίση.

10.5 Γραμμικός κατακερματισμός

Η μέθοδος αυτή προτάθηκε από τον Litwin (1980) και γνώρισε πολλές βελτιώσεις με παραλλαγές. Αν και είναι διαφορετική από τις προηγούμενες γιατί δεν έχει κατάλογο, έχει ωστόσο πολλές ομοιότητες με αυτές. Στο

κλειδί μίας εγγραφής εφαρμόζεται μία συνάρτηση κατακερματισμού (κατά τα γνωστά) και απομονώνονται τα τελευταία (αντί τα πρώτα) k δυαδικά ψηφία του αποτελέσματος. Για παράδειγμα, αν αρχικά το αρχείο έχει οκτώ κάδους, τότε απομονώνονται τα τρία τελευταία bits του αποτελέσματος της συνάρτησης του κατακερματισμού για τον εντοπισμό του κατάλληλου κάδου. Στο Σχήμα 10.8 παρουσιάζεται αυτή η περίπτωση, όπου κάθε κάδος έχει χωρητικότητα τρεις εγγραφές.

000	56			4 = 0000 0100
001	113	193		5 = 0000 0101
010	10	146		10 = 0000 1010
011	19			19 = 0001 0011
100	4			56 = 0011 1000
101	5			113 = 0111 0001
110				146 = 1001 0010
111				193 = 1100 0001

Σχήμα 10.8: Γραμμικός κατακερματισμός με χρήση τριών bits.

Το αρχείο αυξάνει με διαδοχικές διασπάσεις των κάδων, όπου οι εγγραφές ενός κάδου που έχουν κοινά τα τελευταία k bits, μοιράζονται σε δύο κάδους ανάλογα με την τιμή των τελευταίων $k+1$ bits. Στο Σχήμα 10.8 οι εγγραφές με κλειδιά 113 και 193 είναι αποθηκευμένες στον κάδο, που χαρακτηρίζεται από τα τρία ψηφία 001. Η διανομή αυτών των εγγραφών σε δύο κάδους γίνεται ανάλογα με την τιμή των τεσσάρων τελευταίων bits. Κατά σύμπτωση και οι δύο εγγραφές θα αποθηκευθούν και πάλι στον ίδιο κάδο. Δεν συμβαίνει όμως το ίδιο στην περίπτωση των εγγραφών με κλειδιά 10 και 146, που τελικά αποθηκεύονται σε διαφορετικούς κάδους. Αυτό φαίνεται στο Σχήμα 10.8 όπου έχουν διασπασθεί οι τρεις πρώτοι κάδοι. Έτσι, όμως μερικοί κάδοι περιέχουν εγγραφές με κλειδιά που έχουν k κοινά bits, ενώ άλλοι περιέχουν εγγραφές με κλειδιά που έχουν $k+1$ κοινά bits. **Οριακή τιμή** (boundary value) ονομάζεται η τιμή πέρα από την οποία οι κάδοι διακρίνονται από τα τελευταία k bits. Στην επικεφαλίδα κάθε αρχείου γραμμικού κατακερματισμού αποθηκεύεται η οριακή τιμή και η αντίστοιχη τιμή του k . Στο Σχήμα 10.9 οι τιμές αυτές είναι 011 και 3 αντίστοιχα.

0000				
0001	113	193		
0010	90	146		
* 011	19			
100	4	12	52	→ 100
101	5			
110				
111	71			
1000	56			
1001				
1010	10			

$12 = 0000\ 1100$
 $52 = 0011\ 0100$
 $71 = 0100\ 0111$
 $90 = 0101\ 1010$
 $100 = 0110\ 0100$

Σχήμα 10.9: Γραμμικός κατακερματισμός με χρήση τριών και τεσσάρων bits.

10.5.1 Προσπέλαση εγγραφής

Κατά την αναζήτηση εγγραφής απομονώνονται τα τελευταία k bits από το αποτέλεσμα της συνάρτησης κατακερματισμού. Αν η τιμή που προκύπτει με χρήση των k bits είναι μικρότερη από την οριακή τιμή, τότε χρησιμοποιούνται $(k+1)$ bits. Έτσι, αν στο αρχείο του παραδείγματος του Σχήματος 10.11 αναζητείται μία εγγραφή με τελευταία bits τα 1000 ή τα 1101, τότε χρησιμοποιούνται 4 ή 3 bits από αυτά, αντίστοιχα.

Η επίδοση της αναζήτησης σε γραμμικό αρχείο είναι πολύ καλή και προσεγγίζει τη μία προσπέλαση στο δίσκο. Αν η χωρητικότητα του κάδου, $Bkfr$, είναι 50, ενώ ο παράγοντας φόρτισης, Lf , είναι 75%, τότε η επιτυχής και η ανεπιτυχής αναζήτηση απαιτούν 1.05 και 1.27 προσπελάσεις, αντίστοιχα. Κάτω από αυτές τις συνθήκες το χρονικό κόστος για την αναζήτηση είναι:

$$\begin{aligned}
 T_{\text{προσ}}(\text{επιτ}) &= 1.05 \times (s + r + dtt) \\
 T_{\text{προσ}}(\text{ανεπ}) &= 1.27 \times (s + r + dtt)
 \end{aligned}$$

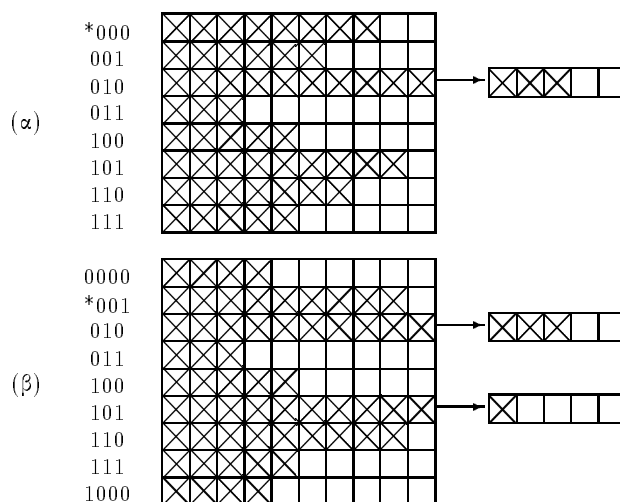
Ο αριθμός των προσπελάσεων είναι συνήθως μεγαλύτερος από τον αντίστοιχο αριθμό του στατικού αρχείου κατακερματισμού με χρήση αλυσίδων,

όταν ο παράγοντας φόρτισης είναι ίδιος. Αυτό οφείλεται στο ότι οι εγγραφές δεν κατανέμονται ομοιόμορφα μεταξύ των κάδων στο γραμμικό κατακερματισμό. Οι κάδοι που αναφέρονται με k bits δέχονται συνήθως περισσότερες εγγραφές απ' ότι οι κάδοι που αναφέρονται με $k+1$ bits. Ας προσεχθεί, όμως, ότι ο αριθμός των προσπελάσεων δεν εξαρτάται από τον αριθμό των εγγραφών στο αρχείο. Δηλαδή, ακόμη και αν στο αρχείο εισαχθούν πολλαπλάσιες εγγραφές η επίδοση θα παραμείνει σταθερή. Αυτό είναι το μεγάλο πλεονέκτημα της μεθόδου αυτής έναντι των άλλων μεθόδων οργάνωσης αρχείων και ιδιαίτερα έναντι της απλής τεχνικής κατακερματισμού με χρήση αλυσίδων.

10.5.2 Εισαγωγή εγγραφής

Η διαδικασία εισαγωγής σε αρχείο γραμμικού κατακερματισμού θα δοθεί με τη βοήθεια ενός παραδείγματος. Ας υποθεθεί ότι ένα αρχείο αποτελείται από οκτώ κύριους κάδους με χωρητικότητα δέκα εγγραφών. Έστω, επίσης, ότι ο παράγοντας φόρτισης δεν πρέπει να ξεπεράσει το 70%. Εκτός από την κύρια περιοχή υπάρχει και η περιοχή υπερχειλίσσης, όπου χρησιμοποιείται η τεχνική της αλυσίδας όπως ακριβώς και στην απλή στατική μέθοδο.

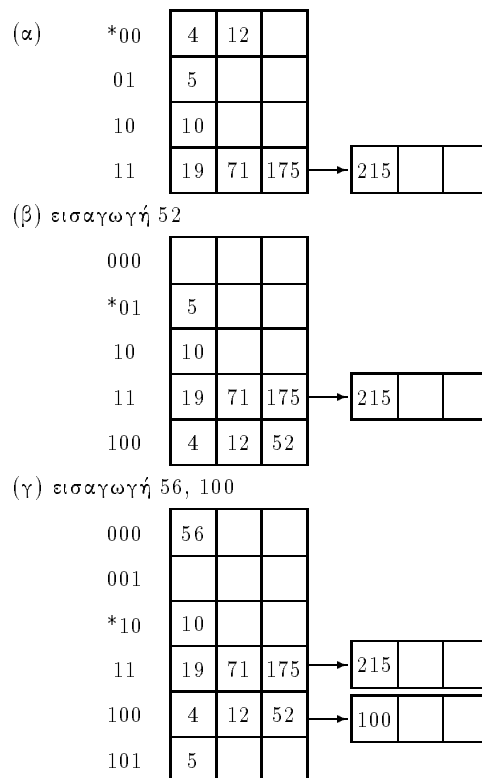
Στο Σχήμα 10.10α παρουσιάζεται αρχείο γραμμικού κατακερματισμού



Σχήμα 10.10: Επέκταση αρχείου γραμμικού κατακερματισμού.

που σε μία δεδομένη χρονική στιγμή περιέχει 56 εγγραφές. Επομένως, ο παράγοντας φόρτισης είναι ακριβώς 70%. Αν προστεθούν άλλες 7 εγγραφές τότε χρειάζεται ένας ακόμη κάδος, ώστε ο παράγοντας φόρτισης να παραμείνει στο 70%. Δηλαδή, γενικά όταν εισάγονται $Lf \times Bkfr$ εγγραφές, τότε το αρχείο αποκτά ένα κάδο ακόμη. Έτσι, ο πρώτος κάδος που χαρακτηρίζεται από το string 000 διασπάται και προκύπτουν δύο κάδοι με αντίστοιχα strings τα 0000 και 1000. Η διαχείριση των εισαγωγών γίνεται με βάση τις πληροφορίες που υπάρχουν αποθηκευμένες στην επικεφαλίδα του αρχείου, την οριακή τιμή και την τιμή του k . Το αποτέλεσμα της εισαγωγής παρουσιάζεται στο Σχήμα 10.10β.

Η διάσπαση των κάδων και η τακτοποίηση της υπερχειλίσης είναι ανεξάρτητα γεγονότα. Δηλαδή, ούτε η υπερχειλίση προκαλεί απαραίτητα διάσπαση,



Σχήμα 10.11: Γραμμική διάσπαση κάδων.

αλλά ούτε και η διάσπαση μειώνει απαραίτητα την υπερχειλίση. Πιο συγκεκριμένα, οι διασπάσεις εκτελούνται γραμμικά από την αρχή προς το τέλος του αρχείου, άσχετα από τον κάδο που υπερχειλίσει. Για παράδειγμα, η δομή του Σχήματος 10.11 αποτελείται από τέσσερις κάδους χωρητικότητας τριών εγγραφών. Αν ο παράγοντας φόρτισης δεν πρέπει να υπερβεί το 67%, τότε το πολύ οκτώ εγγραφές μπορεί να αποθηκευθούν στο αρχείο υπό αυτή τη μορφή. Αυτή η κατάσταση παρουσιάζεται στο Σχήμα 10.11α, όπου ο τέταρτος κάδος συνοδεύεται από αλυσίδα υπερχειλίσης. Με την εισαγωγή της ένατης εγγραφής πρέπει να διασπασθεί ο πρώτος κάδος, όπως φαίνεται στο Σχήμα 10.11β. Με την εισαγωγή τριών ακόμη εγγραφών πρέπει να διασπασθεί ο δεύτερος κάδος. Η νέα κατάσταση παρουσιάζεται στο Σχήμα 10.11γ, όπου πλέον οι δύο κάδοι έχουν αλυσίδα υπερχειλίσης. Ο τέταρτος κάδος θα διασπασθεί μετά την εισαγωγή τεσσάρων ακόμη εγγραφών.

Πειραματικά αποτελέσματα έχουν δείξει ότι σε αρχείο με $Bkfr=50$ και $Lf=75\%$ απαιτούνται κατά μέσο όρο 2,62 προσπελάσεις στο δίσκο για να εισαχθεί μία νέα εγγραφή. Αυτή η τιμή εξηγείται ως εξής. Κατ' αρχήν μία προσπέλαση απαιτείται για να έρθει στην κύρια μνήμη ο κύριος κάδος. Με πιθανότητα $1/Bkfr$ (για την ακρίβεια με ακόμη μεγαλύτερη πιθανότητα) θα δημιουργηθεί ένας νέος κάδος. Τότε θα πρέπει οι εγγραφές του αρχικού κάδου να μοιρασθούν στους δύο νέους κάδους. Αυτό σημαίνει ότι θα γίνουν άλλες δύο προσπελάσεις στο δίσκο για την επανεγγραφή τους. Επιπλέον, αν ο αρχικός κάδος διαθέτει και αλυσίδα υπερχειλίσης, τότε θα είναι απαραίτητο να προσπελασθεί και αυτή. Τελικά, προκύπτει ότι το χρονικό κόστος για μία εισαγωγή είναι:

$$T_{\text{εισ}} = T_{\text{προσ}}(\text{επιτ}) + 2r + \frac{s+r+dt}{Bkfr} + \frac{s+r+dt+2r+s+r+dt}{Lf \times Bkfr}$$

Αν $Bkfr=50$ και $Lf=0.75$, τότε το κόστος αυτό ισούται με:

$$\begin{aligned} T_{\text{εισ}} &= \left(1.27 + \frac{1}{50} + \frac{2}{37}\right) \times (s+r+dt) + \left(1 + \frac{1}{37}\right) \times 2r \\ &= 74\text{ms} = 2.34 \times 31.8\text{ms} = 2.34 \times (s+r+dt) \end{aligned}$$

που προσεγγίζει πολύ την πειραματική τιμή. Το αξιοπρόσεχτο είναι ότι το κόστος εισαγωγής μίας εγγραφής είναι λίγο περισσότερο από δύο προσπελάσεις στο δίσκο. Δηλαδή, φαίνεται ότι το κόστος αυτό είναι σχετικά μεγάλο, ιδιαίτερα αν συγκριθεί με το αντίστοιχο κόστος εισαγωγής σε στατικό αρχείο κατακερματισμού με αλυσίδες. Ωστόσο, τα φαινόμενα απατούν. Το

κόστος εισαγωγής εγγραφής σε αρχείο γραμμικού κατακερματισμού συμπεριλαμβάνει και το κόστος τοπικής αναδιοργάνωσης. Αντίθετα, στο στατικό αρχείο η αναδιοργάνωση γίνεται περιοδικά. Σε τελευταία ανάλυση, η εισαγωγή δεν είναι χρονοβόρα πράξη.

10.5.3 Διαγραφή εγγραφής

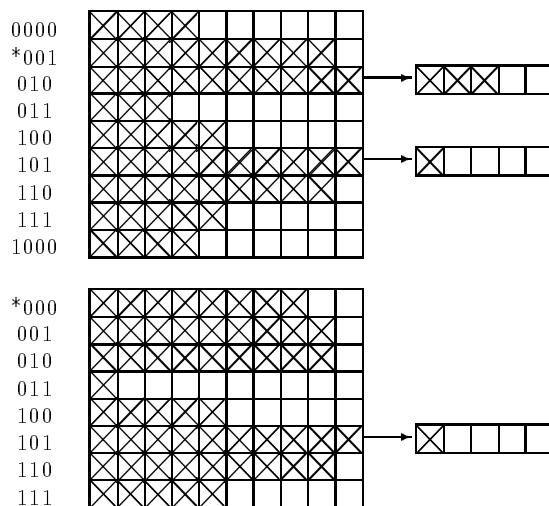
Η διαγραφή εγγραφής μπορεί να γίνει κατά πολλούς τρόπους. Μία λύση είναι να έρθουν στην κύρια μνήμη όλες οι εγγραφές του κάδου και της αλυσίδας υπερχειλίσης και να μεταφερθεί η τελευταία της αλυσίδας στη θέση της εγγραφής που διαγράφεται. Αν η εγγραφή που πρόκειται να διαγραφεί είναι τελευταία στην αλυσίδα, τότε απλώς ελευθερώνεται ο χώρος. Αν ο αριθμός των εγγραφών του αρχείου είναι κατά $Bkfr \times Lf$ μικρότερος από αυτόν που υπογορεύεται από το δεδομένο παράγοντα φόρτισης, τότε το αρχείο συρρικνώνεται κατά ένα κάδο.

Κάτω από αυτές τις συνθήκες το χρονικό κόστος για τη διαγραφή εγγραφής είναι:

$$T_{\text{διαγ}} = T_{\text{προσ}}(\text{ανεπ}) + 2r$$

Αυτός ο τύπος δεν συμπεριλαμβάνει το κόστος συρρίκνωσης του αρχείου. Η πιθανότητα για το γεγονός αυτό είναι πρακτικά μικρή αν υποθεθεί ότι ο ρυθμός άφιξης εγγραφών είναι πιο μεγάλος από το ρυθμό διαγραφών.

Από αλγοριθμική άποψη η διαδικασία συρρίκνωσης είναι η αντίστροφη της διαδικασίας διάσπασης λόγω εισαγωγής. Παρακολουθώντας το συνολικό αριθμό εγγραφών του αρχείου διαπιστώνεται τότε υπάρχουν $Bkfr \times Lf$ εγγραφές λιγότερο από το συγκεκριμένο παράγοντα φόρτισης. Τότε ο τελευταίος κάδος συγχωνεύεται με τον κάδο που έχει κοινά τα k τελευταία bits. Ένα παράδειγμα συρρίκνωσης λόγω διαγραφών φαίνεται στο Σχήμα 10.12, όπου το γραμμικό αρχείο αποτελείται από 9 κάδους με χωρητικότητα δέκα εγγραφές ανά κάδο. Ας υποθεθεί ότι διαγράφονται τέσσερις εγγραφές από τον κάδο 010, δύο εγγραφές από τον κάδο 011 και μία από τον κάδο 1000, οπότε απαιτείται συρρίκνωση του αρχείου. Από τους δύο κάδους 0000 και 1000 δημιουργείται ένας, ο κάδος 000 με τις εγγραφές των δύο προηγούμενων.



Σχήμα 10.12: Συρρίκνωση αρχείου γραμμικού κατακερματισμού.

10.5.4 Εξαντλητική ανάγνωση αρχείου

Όπως στη στατική έκδοση του αρχείου κατακερματισμού, έτσι και ο γραμμικός κατακερματισμός δεν διατηρεί την τάξη των κλειδιών των εγγραφών. Άρα, αν υπάρχει μία λίστα όλων των κλειδιών των εγγραφών, τότε ο χρόνος για την εξαντλητική ανάγνωση του αρχείου ισούται με:

$$T_{\text{εξαν}} = n \times 1.05 \times (s + r + dtt)$$

Αν δεν ενδιαφέρει να δοθούν στο χρήστη οι εγγραφές ταξινομημένες με βάση την τιμή κάποιου κλειδιού, τότε η διαδικασία επιταχύνεται και απλά σαρώνεται το αρχείο από την αρχή ως το τέλος. Ωστόσο, εξαιτίας του κενού χώρου στο τέλος των κάδων, η εξαντλητική αυτή ανάγνωση είναι πιο αργή από την αντίστοιχη σε ένα αρχείο σωρού.

Λόγω της μη ύπαρξης καταλόγου, ο γραμμικός κατακερματισμός είναι η πιο ενδιαφέρουσα τεχνική δυναμικών τυχαίων αρχείων και έχει γνωρίσει πολλές επεκτάσεις. Στη συνέχεια αναφέρονται επιγραμματικά μόνο δύο από αυτές. Σύμφωνα με το γραμμικό κατακερματισμό με μερικές επεκτάσεις (with partial expansions), που προτάθηκε από τον Larson (1980), όταν ο παράγοντας χρησιμοποίησης χώρου υπερβεί την προκαθορισμένη τιμή, τότε δεν διασπάται μόνο ένας κάδος αλλά οι εγγραφές ενός ζεύγους αναδιανέμονται μεταξύ τριών κάδων. Οι κάδοι ενός ζεύγους απέχουν $Bkfr/2$ κάδους

και επιλέγονται γραμμικά μέχρι να τελειώσουν, οπότε έχει συμβεί μία μερική διάσπαση. Στη συνέχεια με παρόμοια τεχνική επιλέγονται τριάδες κάρδων μέχρι το πέρας της δεύτερης μερικής διάσπασης και την ολοκλήρωση μίας πλήρους επέκτασης. Η μέθοδος αυτή βελτιώνει την επίδοση της αναζήτησης και τη χρήση του χώρου με τίμημα το αυξημένο κόστος εισαγωγής σε σχέση με την αρχική μέθοδο. Μία άλλη επέκταση της αρχικής μεθόδου έγινε από τον Tharp (1987) και λέγεται γραμμικός κατακερματισμός με **προτεραιότητα διάσπασης** (priority splitting). Αν γίνει υπέρβαση της καθορισμένης τιμής του παράγοντα χρησιμοποίησης χώρου, τότε δίνεται προτεραιότητα στη διάσπαση του κάρδου με τη μεγαλύτερη αλυσίδα υπερχείλισης. Βέβαια κάθε κάρδος διασπάται μία μόνο φορά κατά τη διάρκεια μίας επέκτασης-διπλασιασμού του αρχείου ακόμη και αν συνεχώς δέχεται εγγραφές στην υπερχείλισή του. Για την υλοποίηση αυτής της μεθόδου απαιτούνται επιπλέον δομές που όμως είναι σχετικά μικρές και αποθηκεύονται στην κύρια μνήμη. Για όλες τις παραλλαγές του γραμμικού κατακερματισμού ισχύει η γενική παρατήρηση ότι η αναζήτηση γίνεται με μία περίπου προσπάθεια στο δίσκο. Όμως, η δομή αυτή δεν προσφέρεται για ερωτήσεις διαστήματος που απαντώνται πολύ αποτελεσματικά από τα B^+ -δένδρα. Σε επόμενο κεφάλαιο θα περιγραφεί μέθοδος που εγγυάται την επιτυχή αναζήτηση με μία προσπάθεια.

10.6 Ασκήσεις

<1> Ποιός είναι ο ελάχιστος και ο μέγιστος αριθμός κάρδων που απαιτούνται από αρχείο δυναμικού κατακερματισμού με $Bkfr=2$ για την αποθήκευση 10 εγγραφών, αν όλες κατευθύνονται στο ίδιο δυαδικό δένδρο του καταλόγου;

<2> Σε αρχείο δυναμικού κατακερματισμού με $Bkfr=2$ να εισαχθούν διαδοχικά οι εγγραφές με τιμές κλειδιών 42, 57, 16, 52, 66, 77, 12, 25, 21 και 33. Να θεωρηθούν οι συναρτήσεις $h_1(key)$, $h_2(key)$ καθώς και η ψευδοτυχαία γεννήτρια του Κεφαλαίου 10.2. Η άσκηση να επαναληφθεί εφαρμόζοντας την παραλλαγή της αναβολής διάσπασης με $\beta=1,5$.

<3> Να σχεδιασθεί η μορφή ενός αρχικά κενού αρχείου επεκτατού κατακερματισμού κατά την εισαγωγή διαδοχικά των κλειδιών 27, 18, 29, 28, 42, 13 και 16. Ως συνάρτηση κατακερματισμού να ληφθεί η $h(key) = key \bmod 11$, ώστε από τα κλειδιά αυτά να προκύψουν τετραψήφιοι δυαδικοί αριθμοί και

να θεωρηθεί κάδος χωρητικότητας τριών εγγραφών. Στη συνέχεια να διαγραφούν τα κλειδιά 16, 29 και 13. Η άσκηση να επιλυθεί θεωρώντας αρχικά τα πρώτα bits και κατόπιν τα τελευταία bits.

<4> Να επιλυθεί η πρώτη άσκηση θεωρώντας ότι οι δείκτες του καταλόγου μπορούν να πάρουν τιμές NIL.

<5> Ποιός είναι ο ελάχιστος αριθμός bits που απαιτείται σε αρχείο επεχτατού κατακερματισμού με

- 10.000 εγγραφές και χωρητικότητα 25 εγγραφές ανά κάδο,
- 100.000 εγγραφές και χωρητικότητα 10 εγγραφές ανά κάδο, και
- 1.000.000 εγγραφές και χωρητικότητα 8 εγγραφές ανά κάδο.

Ποιά είναι η πιθανότητα να μην υπάρχουν συνώνυμα σε αρχείο με 10.000 εγγραφές, όταν η συνάρτηση κατακερματισμού δίνει τιμές στο διάστημα 216, 224 και 232.

<6> Σε εκθετικό αρχείο με περιορισμένο κατάλογο ο κατάλογος αποτελείται από τέσσερις σελίδες με τέσσερα ζεύγη κλειδιών-δεικτών ανά σελίδα, ενώ κάθε σελίδα δεδομένων μπορεί να αποθηκεύσει δύο εγγραφές. Η συνάρτηση κατακερματισμού δίνει οκταψήφιους δυαδικούς αριθμούς. Τα πρώτα δύο ψηφία προσδιορίζουν τη σελίδα του καταλόγου, ενώ τα δύο επόμενα προσδιορίζουν τη διεύθυνση της σελίδας δεδομένων. Να σχεδιασθεί το αποτέλεσμα της εισαγωγής των εγγραφών με κλειδιά 255, 200, 56, 64, 155, 67, 43, 12, 205, 132, 128, 144 και 79.

<7> Σε αρχείο γραμμικού κατακερματισμού με $Bkfr=2$ και $Lf=0.5$ να εισαχθούν διαδοχικά οι εγγραφές με τιμές κλειδιών 42, 57, 16, 52, 66, 77, 12, 25, 21 και 33. Αρχικά το αρχείο αποτελείται από τέσσερις κάδους. Οι τιμές των κλειδιών να μετατραπούν σε επταψήφιους δυαδικούς αριθμούς και να απομονωθούν τα τελευταία ψηφία. Η άσκηση να επαναληφθεί για $Bkfr=3$ και $Lf=0.67$.

<8> Στο αρχείο της προηγούμενης άσκησης να εισαχθούν δέκα εγγραφές, ώστε να προκληθεί υπερχειλίση με αλυσίδα δύο κάδων.

<9> Σε αρχείο γραμμικού κατακερματισμού με $Bkfr=3$, $Lf=0.67$ και τέσσερις αρχικούς κάδους εισάγονται διαδοχικά οι επόμενες 24 εγγραφές με τιμές κλειδιών 42, 57, 16, 52, 66, 77, 12, 25, 21, 33, 32, 14, 50, 49, 47, 7, 56, 13, 62, 27, 31, 41, 30, 4, 10 και 6. Ποιό είναι τα αρχικό μέγεθος του

αρχείου; Πόσα ψηφία θα χρησιμοποιηθούν διαδοχικά; Ποιά θα είναι η δομή του αρχείου με τις διαδοχικές επεκτάσεις;

<10> Να επαναληφθεί η προηγούμενη άσκηση εφαρμόζοντας την παραλλαγή του γραμμικού κατακερματισμού με μερικές επεκτάσεις.

<11> Να επαναληφθεί η προηγούμενη άσκηση εφαρμόζοντας την παραλλαγή του γραμμικού κατακερματισμού με προτεραιότητα διάσπασης.

Κεφάλαιο 11

ΑΝΑΚΤΗΣΗ ΜΕ ΔΕΥΤΕΡΕΥΟΝ ΚΛΕΙΔΙ

- 11.1 Εισαγωγή
- 11.2 Αντεστραμμένα αρχεία
- 11.3 Πολλαπλές λίστες
- 11.4 Συνδυασμένοι κατάλογοι
- 11.5 Πολυδιάστατα δένδρα
- 11.6 Δικτυωτό αρχείο
- 11.7 Ασκήσεις

Κεφάλαιο 11

ΑΝΑΚΤΗΣΗ ΜΕ ΔΕΥΤΕΡΕΥΟΝ ΚΛΕΙΔΙ

11.1 Εισαγωγή

Όλες οι οργανώσεις αρχείων που αναπτύχθηκαν στα προηγούμενα κεφάλαια είναι αποτελεσματικές, όταν οι υποβαλλόμενες ερωτήσεις γίνονται με βάση τιμές του πρωτεύοντος κλειδιού. Όμως, πολύ συχνά στην πράξη τίθενται ερωτήσεις με βάση την τιμή κάποιου δευτερεύοντος κλειδιού. Στην περίπτωση αυτή λέγεται ότι γίνεται **ανάκτηση με βάση δευτερεύον κλειδί** (secondary key retrieval). Ο δευτερεύων κατάλογος, με οποιαδήποτε δομή και αν είναι οργανωμένος (πχ. B^+ -δένδρο κλπ.), έχει ως βασικό χαρακτηριστικό ότι κάθε τιμή ενός δευτερεύοντος κλειδιού συνοδεύεται από μία λίστα δεικτών προς τις αντίστοιχες εγγραφές. Έτσι επιτυγχάνεται ταχύτατη προσπέλαση, βέβαια με τίμημα αυξημένες απαιτήσεις σε μνήμη και επιπρόσθετο χρονικό κόστος για την ενημέρωση των καταλόγων κάθε φορά που ενημερώνονται οι εγγραφές. Όσο περισσότεροι είναι οι δευτερεύοντες κατάλογοι για ένα αρχείο, τόσο μεγαλώνουν τα προβλήματα του αυξημένου χώρου μνήμης και των χρονοβόρων ενημερώσεων. Ένα μέτρο για τον προσδιορισμό του πόσοι και ποιοί δευτερεύοντες κατάλογοι πρέπει να δημιουργηθούν είναι ο λόγος του αριθμού των ενημερώσεων προς τον αριθμό των προσπελάσεων.

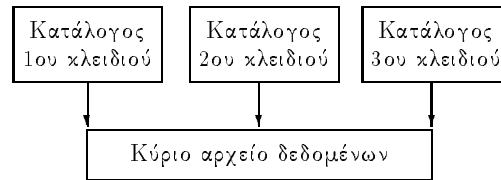
Στο παρόν κεφάλαιο αρχικά πρόκειται να μελετηθούν δύο δομές καταλόγων που στηρίζονται στις συνδεδεμένες λίστες, τα **αντεστραμμένα αρχεία** (inverted files) και οι **πολλαπλές λίστες** (multilists). Οι δομές αυτές μπορούν να χρησιμοποιηθούν για την εξυπηρέτηση λογικών ερωτήσεων που

σχηματίζονται από διαζεύξεις ή/και συζεύξεις απλών ερωτήσεων. Μία άλλη ενδιαφέρουσα οργάνωση καταλόγων είναι η δομή των **συνδυασμένων καταλόγων** (combined indexes), που αποσκοπεί στην αποτελεσματική απάντηση λογικών ερωτήσεων με βάση συζευκτικούς μόνο συνδυασμούς δευτερευόντων κλειδιών. Πιο συγκεκριμένα, σύμφωνα με τη μέθοδο αυτή δημιουργούνται κατάλογοι, όπου κάθε δείκτης προς το κύριο αρχείο αντιστοιχεί σε ζεύγη, τριάδες κλπ. υπαρχόντων τιμών των δευτερευόντων κλειδιών από το αρχείο αυτό. Ωστόσο, πρέπει να τονισθεί ότι αυτές οι οργανώσεις είναι επιπρόσθετες δομές σε σχέση με το κυρίως αρχείο, που μπορεί να είναι οποιουδήποτε τύπου από όσους αναφέρθηκαν στα προηγούμενα κεφάλαια. Όλες αυτές οι δομές εξυπηρετούν αποτελεσματικά απλές ερωτήσεις με βάση την τιμή κάποιου δευτερεύοντος κλειδιού.

Μία δενδρική δομή ειδική για ανάκτηση με δευτερεύον κλειδί είναι το **πολυδιάστατο δένδρο** (k -dimensional tree) και, η γενίκευσή του, το **πολυδιάστατο B-δένδρο** (k -dimensional B-tree). Ιδιαίτερο χαρακτηριστικό των δομών αυτών είναι ότι δεν γίνεται διάκριση μεταξύ πρωτεύοντος και δευτερευόντων κλειδιών, αλλά όλα τα κλειδιά αντιμετωπίζονται ισότιμα. Επίσης συχνά στην πράξη μπορεί να τεθούν από το χρήστη ερωτήσεις διαστήματος. Τα πολυδιάστατα δένδρα και τα πολυδιάστατα B-δένδρα μπορούν να απαντήσουν τέτοιου είδους ερωτήσεις. Το παρόν κεφάλαιο κλείνει την παρουσίαση των κυριότερων δομών αρχείων για ανάκτηση με δευτερεύοντα κλειδιά με το **δικτυωτό αρχείο** (grid file), που εξυπηρετεί το σκοπό αυτό αποτελεσματικότερα από κάθε άλλη οργάνωση.

11.2 Αντεστραμμένα αρχεία

Ένα αντεστραμμένο αρχείο για κάποιο δευτερεύον κλειδί περιέχει όλες τις διακριτές τιμές που λαμβάνει το δευτερεύον κλειδί από το πεδίο ορισμού του μαζί με δείκτες προς τις εγγραφές του κύριου αρχείου, οι οποίες περιέχουν αυτήν την τιμή του δευτερεύοντος κλειδιού. Τότε το αρχείο θεωρείται ότι είναι αντεστραμμένο ως προς το συγκεκριμένο κλειδί. **Πλήρως αντεστραμμένο** (fully inverted) ονομάζεται ένα αρχείο αν διατηρεί ένα δευτερεύοντα κατάλογο για κάθε δευτερεύον κλειδί, ενώ στην αντίθετη περίπτωση ονομάζεται **μερικώς αντεστραμμένο** (partially inverted). Το Σχήμα 11.1 παριστά τη δομή ενός κύριου αρχείου με τρία αντεστραμμένα αρχεία με σκοπό την ανάκτηση με βάση τα αντίστοιχα 3 δευτερεύοντα κλειδιά.



Σχήμα 11.1: Κύριο αρχείο με τρία αντεστραμμένα αρχεία.

Αν η εγγραφή δεν πρόκειται να μετακινηθεί φυσικά, τότε ως δείκτης χρησιμοποιείται η διεύθυνση της εγγραφής στο δίσκο. Ωστόσο, είναι δυνατόν οι θέσεις των εγγραφών να αλλάζουν εξαιτίας εισαγωγών και διαγραφών. Σε μία τέτοια περίπτωση, οι διακριτές τιμές του δευτερεύοντος κλειδιού πρέπει να συνοδεύονται από τις αντίστοιχες τιμές του πρωτεύοντος κλειδιού, ώστε να συνεχισθεί η αναζήτηση προς τις κατάλληλες εγγραφές. Η μέθοδος αυτή μειονεχτεί επειδή είναι πιο αργή, όμως ταυτόχρονα είναι πιο αξιόπιστη.

Κράτος	Πόλη
Βοσνία-Ερζεγοβίνη	Σαράγιεβο
Βουλγαρία	Σόφια, Φιλιπούπολη
Γιουγκοσλαβία	Βελιγράδι
Ελλάδα	Αθήνα, Θεσσαλονίκη
Κροατία	Ζάγκρεμπ
ΠΓΔΜ	Σκόπια
Ρουμανία	Βουκουρέστι, Κωνσταντζα

Λιμάνι	Πόλη
Ναι	Αθήνα, Θεσσαλονίκη, Κωνσταντζα
Όχι	Βελιγράδι, Βουκουρέστι, Σαράγιεβο, Σκόπια, Σόφια, Φιλιπούπολη, Ζάγκρεμπ

Σχήμα 11.2: Αντεστραμμένα αρχεία.

Προφανώς, οι λίστες των δεικτών που προσαρτώνται στις διάφορες διακριτές τιμές των δευτερευόντων κλειδιών δεν είναι ίδιου μήκους, αλλά μεταβλητού. Για παράδειγμα, στο Σχήμα 11.2 δίνονται δύο κατάλογοι ενός αρχείου που αφορά σε πόλεις των βαλκανικών κρατών με πληθυσμό περισσότερο από 340.000 κατοίκους. Ο ένας κατάλογος δίνει τις πόλεις για κάθε διακριτή τιμή του δευτερεύοντος πεδίου 'Κράτος', ενώ ο άλλος κατάλογος

επιμερίζει τις πόλεις σε λιμάνια και μη λιμάνια. Έτσι δημιουργείται πρόβλημα από την ύπαρξη εγγραφών μεταβλητού μήκους στο αντεστραμμένο αρχείο. Στην πράξη ο κατάλογος δεν περιέχει εγγραφές μεταβλητού μήκους, αλλά για όλες τις λίστες δεσμεύεται σταθερός χώρος με μέγεθος που μπορεί να ποικίλει και αποτελεί μία παράμετρο κατά το σχεδιασμό των αρχείων. Αν ο δεσμευμένος χώρος εξαντληθεί, τότε οι υπερχειλίζοντες δείκτες αποθηκεύονται σε νέο σταθερό χώρο που αποδίδεται από το σύστημα. Κατόπιν, η αλληλουχία των σταθερών χώρων για κάθε διακριτή τιμή του δευτερεύοντος κλειδιού συνδέεται, ώστε να σχηματισθεί ένα σειριακό αρχείο ή ένα σειριακό αρχείο με δείκτη. Η μέθοδος αυτή ονομάζεται **κυτταρικό αντεστραμμένο αρχείο** (cellular inverted file).

Οι κατάλογοι δημιουργούνται κατά τη φόρτωση του αρχείου. Με την εισαγωγή κάθε εγγραφής ενημερώνεται και ο αντίστοιχος κατάλογος για κάθε δευτερεύον πεδίο. Δηλαδή, με την εισαγωγή μίας εγγραφής ελέγχεται αν η τιμή ενός συγκεκριμένου πεδίου υπάρχει στον αντίστοιχο κατάλογο. Αν όχι, τότε δημιουργείται η σχετική λίστα με ένα μόνο δείκτη, αλλιώς η υπάρχουσα λίστα ενημερώνεται και επιμηκύνεται κατά άλλον ένα δείκτη. Για τη διαγραφή μίας εγγραφής πρέπει να προσπελασθούν όλοι οι σχετικοί κατάλογοι και να διαγραφούν οι δείκτες (ή οι διευθύνσεις). Αν η λίστα των δεικτών αποτελείται από ένα μόνο δείκτη, τότε διαγράφεται ολόκληρη η λίστα. Ανάλογη είναι και η διαδικασία σε περίπτωση αλλαγής της τιμής κάποιου δευτερεύοντος πεδίου.

Η αναζήτηση με βάση την τιμή ενός δευτερεύοντος κλειδιού είναι πλέον εύκολη. Προσπελάζεται ο σχετικός κατάλογος και επιλέγονται οι αντίστοιχοι δείκτες. Αν η υποβαλλόμενη ερώτηση είναι μία λογική σύζευξη (με and, or και not) πολλών απλών ερωτήσεων με βάση δευτερεύοντα κλειδιά, τότε πρέπει να προσπελασθούν όλοι οι αντίστοιχοι κατάλογοι και να ληφθεί η τομή των σχετικών λιστών πριν γίνει η προσπέλαση στο κύριο αρχείο.

Η μέθοδος των αντεστραμμένων καταλόγων χρησιμοποιείται από όλα σχεδόν τα εμπορικά πακέτα Ανάκτησης Πληροφοριών (Information Retrieval), που κυρίως αφορούν βιβλιοθηκονομικά συστήματα. Το γνωστότερο εμπορικό σύστημα είναι της IBM, το λεγόμενο STAIRS (STorage And Information Retrieval System), και η επεξεργασία του στηρίζεται στην άλγεβρα Boole. Άλλα εμπορικά συστήματα είναι τα BRS, DIALOG, MEDLARS και ORBIT.

11.3 Πολλαπλές λίστες

Σε ένα δευτερεύοντα κατάλογο κάθε διακριτή τιμή του δευτερεύοντος κλειδιού συνοδεύεται από μία σειρά δεικτών, επειδή η ίδια τιμή του κλειδιού μπορεί να εμφανισθεί σε πολλές εγγραφές. Είναι δυνατό να μειωθεί το μέγεθος του δευτερεύοντος καταλόγου, αν δεν αποθηκεύονται όλοι οι δείκτες προς τις αντίστοιχες εγγραφές, αλλά μόνο ένας προς την πρώτη εγγραφή. Βέβαια, για να μην υπάρξει απώλεια δεδομένων, πρέπει όλες οι εγγραφές του κύριου αρχείου με ίδιες τιμές στο δευτερεύον κλειδί να είναι συνδεδεμένες σε μία λίστα. Η συνδεδεμένη λίστα μπορεί να είναι απλή, διπλή ή κυκλική (simple, double, circular linked list). Το ίδιο μπορεί να γίνει και για τα υπόλοιπα δευτερεύοντα κλειδιά με αντίστοιχες λίστες. Όταν ένα αρχείο συνοδεύεται από τέτοιες οργανώσεις λέγεται αρχείο πολλαπλών λιστών.

Πόλη	Κράτος	Δείκτης	Λιμάνι	Δείκτης
Αθήνα	Ελλάδα	Θεσσαλονίκη	Ναι	Θεσσαλονίκη
Βελιγράδι	Γιουγκοσλαβία	—	Όχι	Βουκουρέστι
Βουκουρέστι	Ρουμανία	Κωνσταντζα	Όχι	Ζάγκρεμπ
Ζάγκρεμπ	Κροατία	—	Όχι	Σαράγιεβο
Θεσσαλονίκη	Ελλάδα	—	Ναι	Κωνσταντζα
Κωνσταντζα	Ρουμανία	—	Ναι	—
Σαράγιεβο	Βοσνία-Ερζεγοβίνη	—	Όχι	Σκόπια
Σκόπια	ΠΓΔΜ	—	Όχι	Σόφια
Σόφια	Βουλγαρία	Φιλιπούπολη	Όχι	Φιλιπούπολη
Φιλιπούπολη	Βουλγαρία	—	Όχι	—

Λιμάνι	Πόλη	Μήκος
Ναι	Αθήνα	3
Όχι	Βελιγράδι	7

Κράτος	Πόλη	Μήκος
Βοσνία-Ερζεγοβίνη	Σαράγιεβο	1
Βουλγαρία	Σόφια	2
Γιουγκοσλαβία	Βελιγράδι	1
Ελλάδα	Αθήνα	2
Κροατία	Ζάγκρεμπ	1
ΠΓΔΜ	Σκόπια	1
Ρουμανία	Βουκουρέστι	2

Σχήμα 11.3: Αρχείο πολλαπλών λιστών με απλή λίστα.

Τα δεδομένα του Σχήματος 11.2 παρουσιάζονται στο Σχήμα 11.3 με υλοποίηση απλής συνδεδεμένης λίστας. Το κύριο αρχείο έχει ως πρωτεύον κλειδί το όνομα της πόλης και ως δευτερεύοντα κλειδιά τα χαρακτηριστικά

‘Κράτος’ και ‘Λιμάνι’. Το αρχείο συνοδεύεται από δύο δευτερεύοντες καταλόγους, ένα για κάθε δευτερεύον κλειδί. Σε κάθε κατάλογο οι είσοδοι μίας διακριτής τιμής του δευτερεύοντος κλειδιού έχουν άλλα δύο πεδία: το δείκτη (πρωτεύον κλειδί) της πρώτης εγγραφής του κυρίως αρχείου και το μήκος της αλυσίδας. Επίσης, το κύριο αρχείο έχει δύο επιπλέον πεδία, που είναι οι σύνδεσμοι των λιστών.

Πόλη	Κράτος	Δ.εμπρός	Δ.πίσω	Λιμάνι	Δ.εμπρός	Δ.πίσω
Αθήνα	Ελλάδα	Θεσσαλ.	–	Ναι	Θεσσαλ.	–
Βελιγράδι	Γιουγκοσλ.	–	–	Όχι	Βουκ.	–
Βουκουρέστι	Ρουμανία	Κωνστ.	–	Όχι	Ζάγκρ.	Βελιγρ.
Ζάγκρεμπ	Κροατία	–	–	Όχι	Σαράγ.	Βουκ.
Θεσσαλονίκη	Ελλάδα	–	Αθήνα	Ναι	Κωνστ.	Αθήνα
Κωνσταντζα	Ρουμανία	–	Βουκ.	Ναι	–	Θεσσαλ.
Σαράγιεβο	Βοσνία	–	–	Όχι	Σκόπια	Ζάγκρ.
Σκόπια	ΠΓΔΜ	–	–	Όχι	Σόφια	Σαράγ.
Σόφια	Βουλγαρία	Φιλιπ.	–	Όχι	Φιλιπ.	Σκόπια
Φιλιπούπολη	Βουλγαρία	–	Σόφια	Όχι	–	Σόφια

Σχήμα 11.4: Αρχείο πολλαπλών λιστών (διπλή λίστα).

Το Σχήμα 11.4 είναι ένα παράδειγμα υλοποίησης με διπλά συνδεδεμένη λίστα, όπου για κάθε δευτερεύον πεδίο υπάρχουν δύο δείκτες, ένας προς τα εμπρός και ένας προς τα πίσω. Η υλοποίηση αυτή πλεονεκτεί σε σχέση με την προηγούμενη, γιατί οι εισαγωγές και οι διαγραφές εγγραφών γίνονται αποτελεσματικότερα, όμως απαιτείται επιπλέον χώρος. Σε ένα αρχείο πολλαπλών λιστών υπάρχουν πολλοί τρόποι για να απαντηθεί μία ερώτηση γιατί προσφέρεται η δυνατότητα πλεύσης (navigation) μέσα στο αρχείο. Για παράδειγμα, τίθεται η ερώτηση: ‘Ποιά είναι τα λιμάνια της Βουλγαρίας;’. Η ερώτηση αυτή μπορεί να απαντηθεί κατά τρεις τρόπους (ποιούς;).

Ένα αρχείο πολλαπλών λιστών έχει μία σειρά πλεονεκτημάτων σε σχέση με τη μέθοδο των αντεστραμμένων αρχείων:

- εύκολος προγραμματισμός του λογισμικού σε υλοποιήσεις με απλά συνδεδεμένες λίστες,
- καλή επίδοση κατά την αναζήτηση για μικρές λίστες και για μικρό αριθμό δευτερεύοντων κλειδιών,
- πολύ καλή επίδοση κατά την ανανέωση, ιδιαίτερα για διπλά συνδεδεμένες λίστες,

- δεν απαιτείται χώρος στην κύρια μνήμη για την επεξεργασία των λογικών ερωτήσεων,
- οι κατάλογοι απαιτούν λιγότερο χώρο και μπορεί κατά την επεξεργασία να αποθηκευθούν στην κύρια μνήμη. Αυτό έχει ως συνέπεια την ταχύτερη επεξεργασία τους, γιατί δεν απαιτείται μεταφορά δεδομένων από/προς το δίσκο.

Όμως η οργάνωση των πολλαπλών λιστών σε σχέση με τη μέθοδο των αντεστραμμένων αρχείων έχει και μία σειρά μειονεκτημάτων, όπως:

- σχετικά πολύπλοκος προγραμματισμός του λογισμικού εισαγωγών και διαγραφών σε υλοποιήσεις με διπλά συνδεδεμένες λίστες,
- μέτρια επίδοση κατά την αναζήτηση για μεγάλες λίστες,
- απαιτείται περισσότερος χώρος μέσα στο κύριο αρχείο για την αποθήκευση των δεικτών. Σε υλοποιήσεις με διπλά συνδεδεμένες λίστες για πολλά δευτερεύοντα κλειδιά ο επιπρόσθετος χώρος μπορεί να υπερβαίνει το μέγεθος του χώρου, που εξοικονομείται στον κατάλογο από τη μη αποθήκευση όλων των δεικτών.

Έχει προταθεί μία παραλλαγή του αρχείου πολλαπλών λιστών που υπερβαίνει το πρόβλημα της μέτριας επίδοσης κατά την αναζήτηση σε μεγάλες λίστες. Η οργάνωση αυτή ονομάζεται **κυτταρικό αρχείο πολλαπλών λιστών** (cellular multilist). Το μήκος κάθε λίστας περιορίζεται έτσι ώστε οι αντίστοιχες εγγραφές να χωρούν σε ένα 'κύτταρο', που προσδιορίζεται από το υλικό, όπως για παράδειγμα σε μία σελίδα ή μία άτρακτο του δίσκου. Έτσι κατά την ανάκτηση των εγγραφών μίας λίστας μειώνεται ο χρόνος αναζήτησης λόγω της παλινδρομικής κίνησης του βραχίονα του δίσκου. Το τίμημα για την καλύτερη απόκριση κατά την αναζήτηση είναι ο καταλαμβανόμενος χώρος. Κάθε είσοδος του αντίστοιχου καταλόγου δεν περιέχει μόνο ένα ζεύγος (πρώτη διεύθυνση, μήκος λίστας), αλλά τόσες τριάδες όσες είναι τα κύτταρα από όπου περνά μία λίστα. Επομένως απαιτείται επιπρόσθετος χώρος στον κατάλογο σε σχέση με μία απλή υλοποίηση πολλαπλών λιστών. Σύμφωνα με μία υλοποίηση δεν επιτρέπεται από κάθε κύτταρο να περνούν πάρα πολλές λίστες, αλλά ο αριθμός των λιστών ανά κύτταρο είναι μία παράμετρος που ορίζεται κατά το σχεδιασμό των αρχείων. Έτσι επιτυγχάνεται καλύτερη συγχέντρωση και τοπιχότητα των δεδομένων, αλλά απαιτείται επιπλέον χώρος στο κύριο αρχείο, γιατί διατηρείται κενός χώρος

σε κάθε κύτταρο για μελλοντική εισαγωγή εγγραφών από τις ήδη υπάρχουσες λίστες στο κύτταρο αυτό. Στο Σχήμα 11.5 παρουσιάζεται ο δευτερεύων κατάλογος για τα δεδομένα του Σχήματος 11.2 όταν είναι οργανωμένα με τη μέθοδο αυτή. Υποτίθεται ότι το κύριο αρχείο αποτελείται από τρεις σελίδες με χωρητικότητα τεσσάρων εγγραφών.

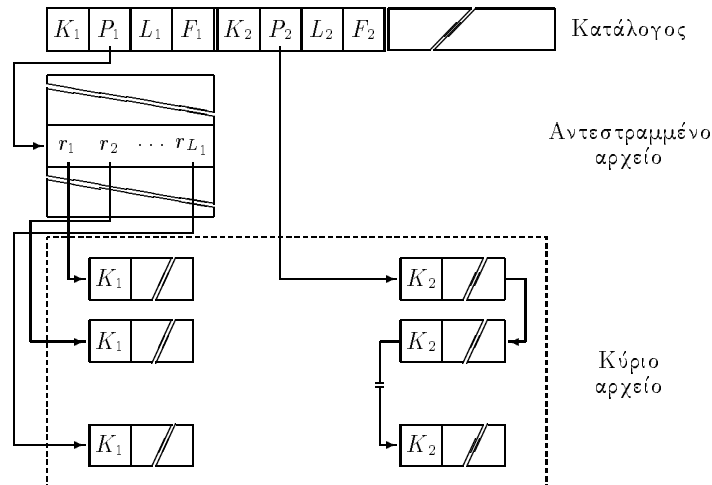
Πόλη	Κράτος	Δείκτης	Λιμάνι	Δείκτης
Αθήνα	Ελλάδα	Θεσσαλονίκη	Ναι	Θεσσαλονίκη
Βελιγράδι	Γιουγκοσλαβία	–	Όχι	Βουκουρέστι
Βουκουρέστι	Ρουμανία	Κωνσταντζα	Όχι	Ζάγκρεμπ
Ζάγκρεμπ	Κροατία	–	Όχι	Σαράγιεβο
Θεσσαλονίκη	Ελλάδα	–	Ναι	Κωνσταντζα
Κωνσταντζα	Ρουμανία	–	Ναι	–
Σαράγιεβο	Βοσνία-Ερζεγοβίνη	–	Όχι	Σκόπια
Σκόπια	ΠΓΔΜ	–	Όχι	Σόφια
Σόφια	Βουλγαρία	Φιλιπούπολη	Όχι	Φιλιπούπολη
Φιλιπούπολη	Βουλγαρία	–	Όχι	–

Λιμάνι	Πόλη	Κύτταρο	Μήκος
Ναι	Αθήνα	1	1
	Θεσσαλονίκη	2	2
Όχι	Βελιγράδι	1	3
	Σαράγιεβο	2	2
	Σόφια	3	2

Σχήμα 11.5: Κυτταρικό αρχείο πολλαπλών λιστών.

Μία άλλη οργάνωση που ανήκει σε αυτήν την κατηγορία είναι το υβριδικό αρχείο λιστών (hybrid list file), που προτάθηκε από τον Yang (1978) και αποτελεί μία λύση που συνδυάζει τα πλεονεκτήματα των δομών των αντεστραμμένων αρχείων και των πολλαπλών λιστών. Αν ο αριθμός των εγγραφών που έχουν μία συγκεκριμένη τιμή σε ένα δευτερεύον κλειδί είναι μεγαλύτερος από μία παράμετρο L , που ορίζεται κατά το σχεδιασμό του αρχείου, τότε δημιουργείται μία αντίστοιχη είσοδος σε ένα αντεστραμμένο αρχείο. Στην αντίθετη περίπτωση η σύνδεση των εγγραφών αυτών γίνεται με τη μέθοδο των πολλαπλών λιστών. Αν $L=0$ τότε η υβριδική αυτή οργάνωση μετατρέπεται σε καθαρή οργάνωση αντεστραμμένων αρχείων, ενώ για πολύ μεγάλες τιμές του L μετατρέπεται σε καθαρή οργάνωση πολλαπλών λιστών. Η επίδοση της μεθόδου αυτής κατά την αναζήτηση υπερτερεί των

επιδόσεων των δύο απλούστερων μεθόδων με αντίστοιχα αυξημένη πολυπλοκότητα λογισμικού.



Σχήμα 11.6: Υβριδικό αρχείο λιστών.

Στο Σχήμα 11.6 παρουσιάζεται ένα υβριδικό αρχείο λιστών. Η εγγραφή του καταλόγου αποτελείται από τέσσερα πεδία: την τιμή του δευτερεύοντος κλειδιού (K_i), το δείκτη προς το αντεστραμμένο αρχείο ή το κύριο αρχείο (P_i), το σύνολο των εγγραφών που έχουν τη συγκεκριμένη τιμή (L_i), και μία σημαία (F_i) που δηλώνει ποιά μέθοδος χρησιμοποιείται για κάθε τιμή του δευτερεύοντος κλειδιού.

Οργανώσεις πολλαπλών λιστών χρησιμοποιούνται στο πακέτο IMS (Information Management System) της IBM που είναι μία ιεραρχική (hierarchical) βάση δεδομένων, καθώς και παλαιότερα στο πακέτο IDMS της Cullinet που είναι μία δικτυωτή (network ή Codasyl) βάση δεδομένων. Οι σχεσιακές (relational) βάσεις δεδομένων δεν χρησιμοποιούν δομές με δείκτες. Σε αντίθεση οι αντικειμενοστραφείς (object-oriented) βάσεις δεδομένων χρησιμοποιούν δομές με δείκτες.

11.4 Συνδυασμένοι κατάλογοι

Οι συνδυασμένοι κατάλογοι είναι μία ομάδα δομών που εξυπηρετούν αποτελεσματικά ερωτήσεις μερικής ταύτισης (partial match queries). Ο

τύπος αυτός ερωτήσεων αντιδιαστέλλεται προς τον τύπο των ερωτήσεων επακριβούς ταύτισης (exact match queries). Με τον όρο 'μερική ταύτιση' εννοούνται οι λογικές ερωτήσεις που αποτελούνται από τη σύζευξη (με and) μερικών χαρακτηριστικών μίας εγγραφής, ενώ με τον όρο 'επακριβής ταύτιση' προσδιορίζονται τιμές για όλα τα χαρακτηριστικά της εγγραφής. Για παράδειγμα, έστω ότι τίθεται μία ερώτηση μερικής ταύτισης, που προσδιορίζει την τιμή a χαρακτηριστικών και ότι σε κάθε χαρακτηριστικό αντιστοιχεί ένας κατάλογος με δομή B^+ -δένδρου. Το κόστος ανάκτησης r εγγραφών που ικανοποιούν την ερώτηση είναι:

$$a \times 2 \times (s + r + btt) + c + r \times (s + r + btt)$$

όπου ο πρώτος όρος δίνει το κόστος προσπέλασης των δύο τελευταίων επιπέδων των B^+ -δένδρων, ο τρίτος όρος δίνει το κόστος προσπέλασης του κύριου αρχείου για την ανάκτηση των εγγραφών, ενώ c είναι το συνήθως αμελητέο κόστος συγχώνευσης στην κύρια μνήμη των λιστών των δεικτών προς το κύριο αρχείο.

Ο χρόνος απόκρισης για την ικανοποίηση μίας τέτοιας ερώτησης θα ήταν μικρότερος αν για κάθε πιθανό συζευχτικό συνδυασμό των a χαρακτηριστικών σε μία ερώτηση μερικής ταύτισης υπήρχε ένας αντίστοιχος κατάλογος. Στον κατάλογο αυτό για κάθε υπαρκτό συνδυασμό των τιμών των υπ' όψη χαρακτηριστικών θα έπρεπε να δίνεται μία λίστα από δείκτες προς τις σχετικές εγγραφές του κύριου αρχείου. Έτσι ο πρώτος όρος του κόστους θα ελαττώνονταν σημαντικά.

Το τίμημα στην περίπτωση αυτή θα ήταν διπλό:

- αντί ενός κλειδιού θα πρέπει να αποθηκεύονται περισσότερα. Έτσι κάθε κατάλογος θα ήταν ογκωδέστερος και επομένως σε σχέση με έναν απλό κατάλογο θα μειώνονταν ο λόγος διακλάδωσης (y), και το σχετικό κόστος θα ήταν μικρότερο κατά λιγότερο από a φορές.
- ο αριθμός των απαιτούμενων καταλόγων θα ήταν υπερβολικός.

Ας υποτεθεί ότι μία εγγραφή έχει τρία χαρακτηριστικά που ονομάζονται A , B και Γ . Εύκολα προκύπτει ότι οι δυνατές ερωτήσεις που μπορούν να απαντηθούν είναι πάρα πολλές, για παράδειγμα μπορεί να είναι ερωτήσεις με βάση μόνο ένα χαρακτηριστικό (A , B , ή Γ), ερωτήσεις με βάση δύο χαρακτηριστικά (AB , BA , $A\Gamma$, ΓA , $B\Gamma$, ΓB), και ερωτήσεις με βάση και τα τρία χαρακτηριστικά ($AB\Gamma$, $A\Gamma B$, $BA\Gamma$, $B\Gamma A$, ΓAB , ΓBA). Δηλαδή, συνολικά

μπορεί να τεθούν 15 διαφορετικές ερωτήσεις μερικής ταύτισης. Εύκολα αποδεικνύεται ότι ο συνολικός αριθμός δυνατών ερωτήσεων που μπορεί να προκύψουν είναι:

$$\sum_{i=1}^a i! \binom{a}{i}$$

Αν για κάθε πιθανή ερώτηση υπήρχε και αντίστοιχος κατάλογος, τότε ευνόητο είναι ότι, ακόμη και για μικρές τιμές του a , το κόστος διατήρησης και επεξεργασίας των καταλόγων θα ήταν απαγορευτικό.

Ο Lum (1970) παρατήρησε ότι δεν έχει σημασία η διάταξη των χαρακτηριστικών στους συνδυασμένους καταλόγους και επομένως ο αριθμός των συνδυασμένων καταλόγων μπορεί να ελαττωθεί δραστικά. Πιο συγκεκριμένα απαιτούνται μόνο κατάλογοι με εγγραφές που προκύπτουν από το συνδυασμό όλων των χαρακτηριστικών. Έτσι αν $a=3$ τότε αρχούν τρεις κατάλογοι, που ονομάζονται ABΓ, ΒΓΑ, και ΓΑΒ. Όσες ερωτήσεις αφορούν στο χαρακτηριστικό Α απαντώνται από τον πρώτο κατάλογο (ABΓ), όσες αφορούν στο Β από το δεύτερο κατάλογο (ΒΓΑ), όσες αφορούν το Γ από τον τρίτο κατάλογο (ΓΒΑ), όσες αφορούν στα Α και Β ή Β και Α από τον πρώτο, όσες αφορούν στα Β και Γ ή Γ και Β από το δεύτερο, ερωτήσεις που αφορούν στα Α και Γ ή Γ και Α από τον τρίτο, ενώ οι ερωτήσεις που αφορούν και στα τρία χαρακτηριστικά μπορούν να απαντηθούν από οποιοδήποτε κατάλογο. Ο συνολικός αριθμός συνδυασμένων καταλόγων για a χαρακτηριστικά είναι:

$$\binom{a}{\lfloor a/2 \rfloor} = \frac{a!}{\lceil a/2 \rceil! \lfloor a/2 \rfloor!}$$

Έτσι αν $a=4$ τότε αρχούν έξι κατάλογοι (οι ABΓΔ, ΒΓΔΑ, ΒΔΑΓ, ΓΑΔΒ, ΓΔΑΒ και ΔΑΒΓ), αν και ο αριθμός των πιθανών ερωτήσεων μερικής ταύτισης που μπορεί να τεθούν είναι εξήντα ένα.

Από τον Shneiderman (1977) προτάθηκε η μέθοδος των μειωμένων συνδυασμένων καταλόγων (reduced combined indices), που στηρίζεται στην παρατήρηση ότι εκτός από έναν κατάλογο όλοι οι υπόλοιποι κατάλογοι που προκύπτουν σύμφωνα με τη μέθοδο του Lum μπορούν να απλοποιηθούν. Δηλαδή αν $a=3$, τότε ο αριθμός των απαραίτητων καταλόγων είναι και πάλι τρεις, αλλά απλουστεύοντας τους προκύπτουν οι κατάλογοι ABΓ, ΒΓ και ΓΑ. Αν $a=4$ τότε ο αριθμός των απαραίτητων καταλόγων είναι και πάλι έξι, αλλά απλουστεύοντας τους προκύπτουν οι κατάλογοι ABΓΔ, ΒΔΑ, ΓΑΔ,

$\Delta\Gamma\text{B}$, $\text{A}\Delta$ και $\text{B}\Gamma$. Για παράδειγμα, χρησιμοποιώντας τον κατάλογο $\text{AB}\Gamma\Delta$ απαντώνται οι ερωτήσεις μερικής ταύτισης που αφορούν στους εξής τέσσερις συνδυασμούς χαρακτηριστικών: $\text{AB}\Gamma\Delta$, $\text{AB}\Gamma$, AB και A . Κατά τον ίδιο τρόπο και οι υπόλοιποι κατάλογοι εξυπηρετούν περισσότερες από μία ερωτήσεις μερικής ταύτισης, ενώ διασφαλίζεται ότι κάθε πιθανή ερώτηση θα ικανοποιηθεί από αυτήν την ομάδα καταλόγων. Είναι προφανές ότι η μέθοδος των μειωμένων συνδυασμένων καταλόγων υπερτερεί σε σχέση με την αρχική μέθοδο τόσο από πλευράς χώρου όσο και πλευράς χρόνου. Πιο συγκεκριμένα προκύπτει ότι ο αριθμός των καταλόγων που δεικτοδοτούν i χαρακτηριστικά είναι:

$$\binom{a}{i} - \binom{a}{i+1} \quad \text{για } \lceil a/2 \rceil \leq i < a$$

ενώ απαιτείται και ένας κατάλογος που δεικτοδοτεί όλα τα χαρακτηριστικά.

Από τον Shneiderman επίσης προτάθηκε η εναλλακτική μέθοδος των τροποποιημένων συνδυασμένων καταλόγων (modified combined indices), που σε σχέση με τις προηγούμενες μεθόδους διατηρεί περισσότερους καταλόγους, αλλά μικρότερου μεγέθους. Ο συνολικός αριθμός τροποποιημένων συνδυασμένων καταλόγων για a χαρακτηριστικά είναι:

$$2^{a-1}$$

Αν $a=3$ τότε ο αριθμός των απαραίτητων καταλόγων είναι τέσσερις ($\text{AB}\Gamma$, $\text{A}\Gamma$, $\text{B}\Gamma$ και Γ), ενώ αν $a=4$ τότε ο αριθμός των απαραίτητων καταλόγων είναι οκτώ ($\text{AB}\Gamma\Delta$, $\text{AB}\Delta$, $\text{A}\Gamma\Delta$, $\text{A}\Delta$, $\text{B}\Gamma\Delta$, $\text{B}\Delta$, $\Gamma\Delta$ και Δ). Πιο συγκεκριμένα προκύπτει ότι ο αριθμός των καταλόγων που δεικτοδοτούν i χαρακτηριστικά είναι:

$$\binom{a-1}{i-1}$$

Επομένως όταν ο χρόνος είναι πιο κρίσιμος παράγοντας από τον πρώτο η μέθοδος των τροποποιημένων καταλόγων πρέπει να προτιμηθεί σε σχέση με τη μέθοδο των μειωμένων, ενώ αν ο χώρος είναι πιο κρίσιμος παράγοντας η προτίμηση θα πρέπει να αντιστραφεί. Βέβαια, εκτός από το χρόνο ανάκτησης, πρέπει να θεωρήσουμε και το χρόνο εισαγωγής και διαγραφής. Ως προς τις παραμέτρους αυτές, η μέθοδος των τροποποιημένων καταλόγων υστερεί επειδή για κάθε εισαγωγή ή διαγραφή απαιτείται η ενημέρωση περισσότερων καταλόγων.

Παρά το σημαντικό περιορισμό του αριθμού των καταλόγων όλες οι προηγούμενες παραλλαγές είναι απαγορευτικές για λίγο μεγαλύτερο αριθμό χαρακτηριστικών. Από τον Stonebraker (1974), το θεμελιωτή των Συστημάτων Διαχείρισης Βάσεων Δεδομένων Ingres, Postgres και Illustra μελετήθηκαν τρόποι εύρεσης του βέλτιστου αριθμού συνδυασμένων καταλόγων και του βέλτιστου τρόπου ομαδοποίησης των χαρακτηριστικών σε κάθε ένα συνδυασμένο κατάλογο. Ας θεωρηθεί μία απλή περίπτωση: για παράδειγμα, αν $a=9$ τότε προκύπτουν 126 συνδυασμένοι κατάλογοι. Αν μερικά χαρακτηριστικά δεν συνδυάζονται ποτέ μεταξύ τους σε μία ερώτηση μερικής ταύτισης, τότε τα πράγματα διευκολύνονται σημαντικά. Ας υποθεθεί ότι $a=9$ και ότι τα χαρακτηριστικά αυτά υποδιαιρούνται σε τρεις διακριτές τριάδες, ενώ στοιχεία από διαφορετικές τριάδες δεν συνδυάζονται μεταξύ τους. Στην περίπτωση αυτή προκύπτουν εννέα κατάλογοι, δηλαδή αριθμός που είναι ίσος με τον αριθμό των χαρακτηριστικών και συνεπώς ίσος με τον αριθμό των απλών καταλόγων. Όμως, σύμφωνα με τη μέθοδο αυτή, οι κατάλογοι έχουν μεγαλύτερο ειδικό βάρος.

11.5 Πολυδιάστατα δένδρα

Το πολυδιάστατο δένδρο (που απλούστερα λέγεται και k -d δένδρο) είναι μία πολύ αποτελεσματική δομή για ανάκτηση με δευτερεύον κλειδί, καθώς και για ερωτήσεις μερικής ταύτισης ή ερωτήσεις διαστήματος. Η δομή αυτή δεν κάνει διάκριση μεταξύ των k κλειδιών σε πρωτεύον και δευτερεύοντα, αλλά τα αντιμετωπίζει όλα ισότιμα. Έτσι η δομή παρουσιάζει ιδιαίτερα πλεονεκτήματα.

Τα πολυδιάστατα δένδρα προτάθηκαν από τον Bentley (1975). Έκτοτε έχουν μελετηθεί από πολλούς ερευνητές και έχουν προκύψει πολλές βελτιωμένες παραλλαγές. Στην αρχική εκδοχή η δομή χρησιμοποιούνταν για την αποθήκευση των δεδομένων στην κύρια μνήμη αλλά την ίδια στιγμή εξυπηρετεί και ως κατάλογος (όπως άλλωστε συμβαίνει και στα Β-δένδρα, που εξετάστηκαν στο βιβλίο των Δομών Δεδομένων).

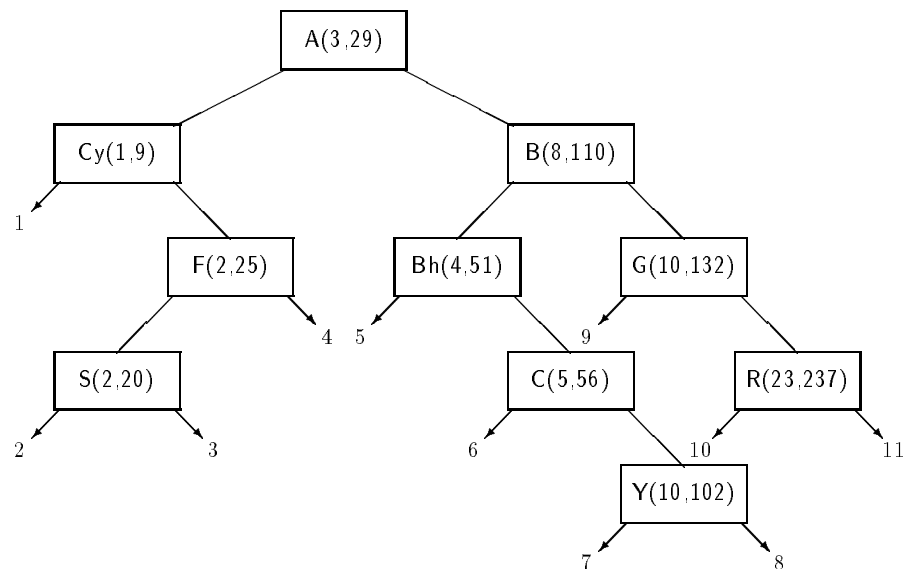
Τα απλά δυαδικά δένδρα αναζήτησης μπορούν να θεωρηθούν ως μονοδιάστατα (1 -d) δένδρα. Η ομοιότητα των δύο τύπων δένδρων έγκειται στο γεγονός ότι ο βαθμός τους είναι δύο, οπότε σε κάθε κόμβο αποθηκεύονται δύο μόνο δείκτες. Ωστόσο, η βασική διαφορά των k -d δένδρων από τα απλά δυαδικά δένδρα αναζήτησης είναι ότι σε κάθε επίπεδο του δένδρου χρησιμοποιείται ένα διαφορετικό χαρακτηριστικό της εγγραφής, ώστε οι εγγραφές

να υποδιαιρεθούν σε δύο ανεξάρτητα μεταξύ τους υποσύνολα σύμφωνα με τη στρατηγική 'διαίρει και βασίλευε'. Για παράδειγμα, στο επίπεδο της ρίζας οι εγγραφές υποδιαιρούνται σε δύο υποσύνολα με βάση την τιμή του πρώτου χαρακτηριστικού, οπότε οι εγγραφές με τις μικρότερες και τις μεγαλύτερες τιμές στο πρώτο χαρακτηριστικό κατευθύνονται στο αριστερό και στο δεξιό υποδένδρο, αντίστοιχα. Δηλαδή, στο δεύτερο, τρίτο κοχ. επίπεδο η διάκριση των εγγραφών γίνεται με βάση τις τιμές του δεύτερου, του τρίτου κοχ. χαρακτηριστικού, αντίστοιχα. Στο $(k+1)$ -οστό επίπεδο η διάκριση γίνεται και πάλι με βάση το πρώτο χαρακτηριστικό. Έτσι, η διαδικασία προχωρεί κατά κυκλικό τρόπο.

Έχει αποδειχθεί αναλυτικά ότι για ένα τυχαίο πολυδιάστατο δένδρο η πολυπλοκότητα της εισαγωγής και της τυχαίας διαγραφής είναι $O(\log n)$, ενώ η πολυπλοκότητα μίας λογικής ερώτησης που αφορά στα t από τα k χαρακτηριστικά είναι $O(n^{(k-t)/k})$. Οι επιδόσεις αυτές είναι πολύ καλές, αλλά πρακτικά συνιστάται να χρησιμοποιείται αυτή η δομή μόνο αν ο αριθμός των εγγραφών είναι μεγαλύτερος από 2^{2k} . Ο όρος αυτός τίθεται ώστε να είναι βέβαιο ότι το δένδρο θα έχει περισσότερο από $2k$ επίπεδα, οπότε κάθε χαρακτηριστικό θα χρησιμοποιείται τουλάχιστον δύο φορές για τη διάκριση των εγγραφών σε δύο υποσύνολα. Νεότερη παραλλαγή είναι τα **ισοζυγισμένα πολυδιάστατα δένδρα** (multidimensional balanced binary trees) κατά αναλογία προς τα δένδρα AVL, που εξετάστηκαν στο βιβλίο των Δομών Δεδομένων. Έχει αποδειχθεί από τον Vaishnavi (1989) ότι η πολυπλοκότητα για την αναζήτηση, εισαγωγή και διαγραφή μίας εγγραφής είναι της τάξης $O(\log n + k)$, ενώ σε κάθε εισαγωγή ή διαγραφή απαιτούνται περιστροφές της τάξης $O(k)$.

Ωστόσο, όλες οι προηγούμενες παραλλαγές είναι ομογενείς δομές και σχεδιάστηκαν για αποθήκευση στην κύρια μνήμη. Βέβαια, μπορούν να αποθηκευθούν και στη δευτερεύουσα μνήμη αλλά η επίδοσή τους θα εκφυλισθεί (όπως το απλό δυαδικό δένδρο αναζήτησης). Κάτω από αυτό το πρίσμα μία ενδιαφέρουσα παραλλαγή της οργάνωσης αυτής ονομάζεται **εκτεταμένο πολυδιάστατο δένδρο** (extended k -d tree) και προτάθηκε από την Chang (1981). Η ετερογενής αυτή δομή μπορεί να θεωρηθεί ότι αποτελείται από δύο μέρη: ένα απλό k -d δένδρο που αποθηκεύεται στην κύρια μνήμη και λειτουργεί ως μηχανισμός καταλόγου για τη δεικτοδότηση των δεδομένων του κυρίου αρχείου, που προφανώς είναι αποθηκευμένο στη δευτερεύουσα μνήμη. Στη συνέχεια εξετάζεται ένα παράδειγμα οργάνωσης δεδομένων σύμφωνα με τη μέθοδο αυτή.

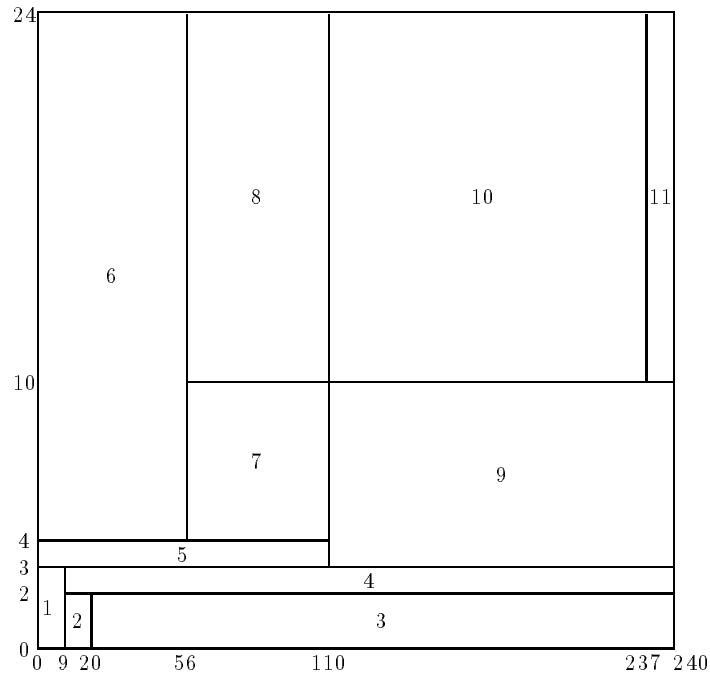
Έστωσαν τα δεδομένα του Πίνακα 7.2 που αφορούν σε δεδομένα (πληθυσμό και έκταση) των Βαλκανικών κρατών. Δηλαδή, τα δευτερεύοντα χαρακτηριστικά είναι δύο ($k=2$) και επομένως η αποθήκευση των δεδομένων θα γίνει σε ένα δισδιάστατο δένδρο. Ας υποτεθεί ότι τα δεδομένα εισάγονται στο δένδρο με αλφαβητική σειρά: A, B, Bh, C, Cy, F, G, R, S, Y. Η ρίζα του 2- d δένδρου θα είναι η εγγραφή A(3,29). Η επόμενη εισαγωγή της εγγραφής B(8,110) θα γίνει η ρίζα του αριστερού δένδρου γιατί $8 > 3$, ενώ η εγγραφή Bh(4,51) θα κατευθυνθεί στα αριστερά της εγγραφής B(8,110) γιατί $51 < 110$. Στο Σχήμα 11.7 παρουσιάζεται το δένδρο στην τελική του μορφή. Ας σημειωθεί ότι αν σε περίπτωση σύγκρισης προκύψει ισότητα, τότε τα εισαγόμενα δεδομένα κατευθύνονται στο αριστερό υποδένδρο.



Σχήμα 11.7: Δισδιάστατο δένδρο με τα δεδομένα του Πίνακα 7.2.

Στους κόμβους του δένδρου του Σχήματος 11.7 αποθηκεύονται και οι δύο τιμές των σχετικών εγγραφών. Όμως σε μία πραγματική υλοποίηση δεν είναι απαραίτητο να αποθηκεύονται και οι δύο τιμές, αλλά μόνον εκείνη που στο συγκεκριμένο επίπεδο χρησιμεύει για τη σύγκριση. Έτσι ολόκληρες οι εγγραφές αποθηκεύονται στο αρχείο, που είναι αποθηκευμένο στο δίσκο. Από τα φύλλα του δένδρου δεικτοδοτούνται 11 χάντοι του αρχείου, με διευθύνσεις που φαίνονται στο Σχήμα 11.7. Είναι ευνόητο ότι αν n είναι οι

εγγραφές του καταλόγου, τότε οι δεικτοδοτούμενοι κάδοι του αρχείου είναι $n+1$.



Σχήμα 11.8: Υποδιαίρεση χώρου σύμφωνα με τα δεδομένα του Πίνακα 7.2.

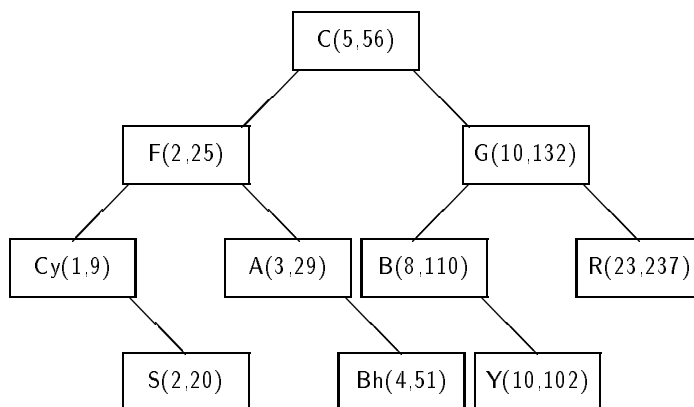
Κάθε κόμβος του 2- d δένδρου διαιρεί το χώρο σε δύο υποχώρους ανάλογα με τις συνθήκες \leq και $>$. Σε κάθε κατώτερο επίπεδο γίνονται διαδοχικές υποδιαίρεσεις των υποχώρων. Στο Σχήμα 11.8 φαίνεται η διαίρεση του χώρου σε υποχώρους, ενώ σε κάθε υποχώρο παρουσιάζεται η διεύθυνση του αντίστοιχου κάδου, όπως προκύπτει από το Σχήμα 11.7. Στον οριζόντιο άξονα φαίνονται οι τιμές του χαρακτηριστικού έκτασης (9,20,56,110,237) που χρησιμοποιούνται για σύγκριση στα επίπεδα άρτιας τάξης. Αντίστοιχα, στον κατακόρυφο άξονα φαίνονται οι τιμές του χαρακτηριστικού πληθυσμός (2,3,4,10), που χρησιμοποιούνται για σύγκριση στα επίπεδα περιττής τάξης.

Το κύριο αρχείο, λοιπόν, έχει 11 κάδους. Η αποθήκευση των εγγραφών φαίνεται στον Πίνακα 11.1. Παρατηρείται ότι ένας κάδος περιέχει δύο εγγραφές, ενώ δύο κάδοι παραμένουν κενοί. Δηλαδή, σε μία τέτοια υλοποίηση ο κατάλογος διαμορφώνεται πριν τη φόρτωση του αρχείου και είναι

Χώρα	Κάδος
Αλβανία (A)	4
Βοσνία-Ερζεγοβίνη (Bh)	5
Βουλγαρία (B)	7
Γιουγκοσλαβία (Y)	7
Ελλάδα (G)	9
Κροατία (C)	6
Κύπρος (Cy)	1
ΠΓΔΜ (F)	3
Ρουμανία (R)	10
Σλοβενία (S)	2

Πίνακας 11.1: Αποθήκευση εγγραφών στους κάδους του αρχείου.

στατικός. Σωστά σχεδιασμένος είναι ο κατάλογος που είναι περίπου ισοζυγισμένος. Για το σκοπό αυτό πρέπει κατά τη δημιουργία του καταλόγου σε κάθε διαδοχικό επίπεδο του δένδρου ως ρίζα να επιλέγεται η εγγραφή εκείνη, που στο κατάλληλο πεδίο έχει τη μεσαία τιμή σε σχέση με τις τιμές του ίδιου πεδίου όλων των εγγραφών, που ανήκουν στο συγκεκριμένο υποδένδρο. Στο Σχήμα 11.9 παρουσιάζεται ένα ισοζυγισμένο 2-d δένδρο που προκύπτει εφαρμόζοντας αυτή τη διαδικασία. Αν είναι γνωστό το πλήθος των εγγραφών και η χωρητικότητα των κάδων, τότε εύκολα προκύπτει ο αριθμός των κλειδιών, που χρειάζονται για τη δημιουργία του καταλόγου.



Σχήμα 11.9: Ισοζυγισμένο 2-d δένδρο με τα δεδομένα του Πίνακα 7.2.

Αν οι κάδοι του αρχείου αυτού γεμίσουν, τότε δημιουργούνται κάδοι υπερχειλίσης. Αν το φαινόμενο της υπερχειλίσης γίνει έντονο και η επίδοση κατά την αναζήτηση εκφυλισθεί, τότε η αναδιοργάνωση του αρχείου και η δημιουργία ενός νέου καταλόγου είναι αναγκαία.

Στη συνέχεια δίνονται παραδείγματα αναζήτησης για ερώτηση μερικής ταύτισης και ερώτηση διαστήματος με βάση τα Σχήματα 11.7 και 11.8. Ας υποθεθεί ότι τίθεται η ερώτηση: 'Ποιά βαλκανική χώρα έχει πληθυσμό 10.000.000 κατοίκους και έκταση 102.000 τετραγωνικά χιλιόμετρα;'. Η αναζήτηση ξεκινά από τη ρίζα, όπου η σύγκριση γίνεται με την τιμή 3, και συνεχίζει στο δεξιό υποδένδρο. Στο νέο κόμβο η σύγκριση γίνεται με βάση την τιμή 110, οπότε η αναζήτηση συνεχίζει στο αριστερό υποδένδρο. Στο τρίτο πλέον επίπεδο μετά από σύγκριση με την τιμή 4, ακολουθείται ο δεξιός δενδρικός δείκτης που παραπέμπει στο τέταρτο επίπεδο. Εκεί η σύγκριση γίνεται με βάση το κλειδί 56, οπότε η αναζήτηση η αναζήτηση συνεχίζει στο δεξιό παιδί (που είναι φύλλο). Από τον κόμβο αυτό λαμβάνεται ο αριστερός δείκτης προς τον κάδο υπ' αριθμόν 7. Όλη αυτή η διαδικασία γίνεται στην κύρια μνήμη και θεωρείται αμελητέου κόστους. Από το δίσκο προσπελάζεται ο κάδος υπ' αριθμόν 7, που περιέχει δύο εγγραφές: της Βουλγαρίας και της Γιουγκοσλαβίας. Οι εγγραφές ελέγχονται στην κύρια μνήμη και τελικώς επιστρέφεται στο χρήστη η σωστή απάντηση. Αυτός ο έλεγχος των εγγραφών του προσπελασθέντος κάδου είναι απαραίτητος, αφού έτσι άλλωστε διαπιστώνεται και η ανεπιτυχής αναζήτηση.

Ας θεωρηθεί τώρα η ερώτηση διαστήματος: 'Ποιές βαλκανικές χώρες έχουν πληθυσμό περισσότερο από 6.000.000;'. Και πάλι η αναζήτηση ξεκινά από τη ρίζα και προφανώς συνεχίζει στο δεξιό υποδένδρο. Στο νέο κόμβο δεν μπορεί να γίνει σύγκριση, οπότε η αναζήτηση συνεχίζει και στο αριστερό και στο δεξιό υποδένδρο. Στο τρίτο πλέον επίπεδο μετά από σύγκριση με τις τιμές 4 και 10, η αναζήτηση συνεχίζει σε τρεις κατευθύνσεις. Η τελική κατάληξη είναι να προσπελασθούν οι κάδοι υπ' αριθμόν 6, 7, 8, 9, 10, και 11. Οι εγγραφές ελέγχονται στην κύρια μνήμη και τελικώς από το περιεχόμενο των κάδων αυτών απορρίπτεται μόνο η εγγραφή της Κροατίας. Αν και η επεξεργασία του καταλόγου θεωρείται αμελητέου κόστους, στην περίπτωση αυτή πρέπει να τονισθεί ότι η αναζήτηση είναι σχετικά πολύπλοκη διαδικασία, γιατί πρέπει να διατηρούνται ενεργά πολλά μονοπάτια.

Τα πολυδιάστατα B-δένδρα (*k-d B-trees*), που προτάθηκαν από τον Robinson (1981), είναι μία γενίκευση των πολυδιάστατων δένδρων (όπως τα B-δένδρα είναι μία γενίκευση των απλών δυαδικών δένδρων). Η δομή

αυτή είναι σχεδιασμένη για υλοποίηση στο δίσκο, οπότε σε κάθε κόμβο των δένδρων αυτών δεν περιέχεται μόνο μία εγγραφή αλλά περισσότερες. Οι αλγόριθμοι εισαγωγής και διαγραφής είναι ιδιαίτερα πολύπλοκοι στον προγραμματισμό και την κατανόηση τους, γιατί η δομή πρέπει να παραμένει ισοζυγισμένη. Έτσι για να απλοποιηθεί η δομή δεν ισχύει το ελάχιστο ποσοστό χρήσης που ισχύει στα B-δένδρα (δηλαδή, το $U_{min}=50\%$), οπότε η μέση χρήση του χώρου είναι μόνο 60% περίπου, όταν τα δεδομένα υπακούουν σε ομοιόμορφη κατανομή.

11.6 Δικτυωτό αρχείο

Έστω ότι από δορυφορικές φωτογραφίες έχει δημιουργηθεί ένα αρχείο-χάρτης με τις πόλεις της Ρωσίας. Κάθε πόλη διακρίνεται από το γεωγραφικό πλάτος και το γεωγραφικό μήκος, που αποτελούν δύο ισότιμα κριτήρια. Αν το αρχείο οργανωθεί με βάση το γεωγραφικό πλάτος, τότε πόλεις με ίδιο γεωγραφικό πλάτος αλλά διαφορετικό γεωγραφικό μήκος θα αποθηκευθούν μαζί. Έτσι, για παράδειγμα, το Μαγκαντάν ($59^0 57'$ βόρεια και $150^0 43'$ ανατολικά, κοντά στα σύνορα με την Ιαπωνία) θα αποθηκευθεί μαζί με το Λένινγκραντ ($59^0 57'$ βόρεια και $30^0 20'$ ανατολικά, κοντά στα σύνορα με τη Φιλανδία). Αυτές οι πόλεις απέχουν μεταξύ τους 4000 μίλια και όμως πρέπει σύμφωνα με αυτήν την οργάνωση να αποθηκευθούν μαζί. Συνεπώς, χρειάζεται μία δομή με χαρακτηριστικά διαφορετικά από όσες εξετάστηκαν μέχρι εδώ.

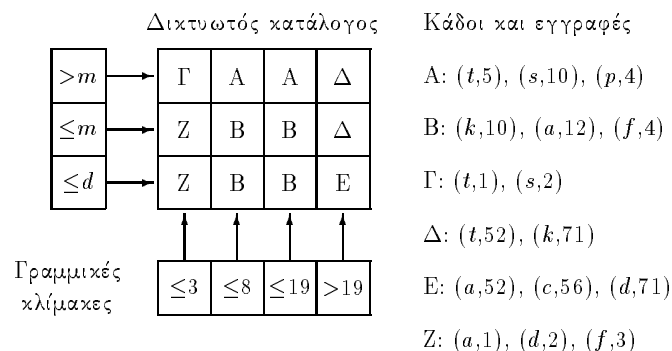
Έστω ότι μία εγγραφή έχει τέσσερα κλειδιά. Θεωρείται ο τετραδιάστατος χώρος που ορίζεται από τα πεδία ορισμού των χαρακτηριστικών. Αυτός ο χώρος είναι δυνατόν να θεωρηθεί ως ένας δυαδικός πίνακας. Στον πίνακα αυτόν η διάσταση i θα έχει τόσα στοιχεία όσες είναι οι διακριτές τιμές του χαρακτηριστικού i . Ένα στοιχείο του τετραδιάστατου πίνακα θα ισούται με 1, αν υπάρχει εγγραφή με τις αντίστοιχες τιμές για τα τέσσερα χαρακτηριστικά. Ένα στοιχείο του πίνακα θα είναι μηδέν, αν δεν υφίσταται η σχετική εγγραφή. Έτσι ο πίνακας των bits θα είναι μία πλήρης αναπαράσταση του συνόλου των εγγραφών.

Αρχικά φαίνεται ότι η δομή αυτή μπορεί να ικανοποιήσει όλες τις απαιτήσεις από ένα σύστημα αρχείων. Επεξεργασία των σημειακών ερωτήσεων γίνεται με εξέταση ενός μόνο στοιχείου, ενώ οι ερωτήσεις διαστήματος απαντώνται με επεξεργασία όλων των στοιχείων του j -διάστατου χώρου, όπου

$j \leq k$. Οι εισαγωγές και οι διαγραφές επιτυγχάνονται θέτοντας τα αντίστοιχα bits ίσα με 1 ή με 0. Τέλος, όλα τα κλειδιά αντιμετωπίζονται κατά τον ίδιο τρόπο. Ωστόσο, αυτή η οργάνωση δεν μπορεί να ανταπεξέλθει σε πρακτικές καταστάσεις. Για παράδειγμα, αν υπήρχαν τέσσερα πεδία με 100 διακριτές τιμές το καθένα, τότε ο πίνακας θα αποτελούνταν από 1.000.000 στοιχεία. Η προηγούμενη οργάνωση, λοιπόν, είναι μία ουτοπική λύση.

Το δικτυωτό αρχείο είναι μία νέα οργάνωση που προτάθηκε από το Nievergelt (1984). Η δομή αυτή διαχειρίζεται το ίδιο αποτελεσματικά τόσο στατικά όσο και δυναμικά δεδομένα, επειδή σχεδιάστηκε με βάση τις εξής απαιτήσεις:

- αποτελεσματική ικανοποίηση σημειακών ερωτήσεων (point queries), που γίνονται με βάση συγκεκριμένες τιμές για όλα τα κλειδιά,
- αποτελεσματική ικανοποίηση των ερωτήσεων διαστήματος και ερωτήσεων μερικής ταύτισης,
- δυναμική προσαρμοστικότητα στις εισαγωγές και διαγραφές εγγραφών, και τέλος
- ισότιμη αντιμετώπιση όλων των κλειδιών, πρωτεύοντος και δευτερεύοντων.



Σχήμα 11.10: Δομή δικτυωτού αρχείου.

Στο Σχήμα 11.10 παρουσιάζεται η δομή του δικτυωτού αρχείου που αποτελείται από τους κάδους A,B,...,Z, που περιέχουν τα δεδομένα, και τον κατάλογο που απαρτίζεται από δύο μέρη: το δικτυωτό κατάλογο ή πίνακα

(grid directory, array) και τις γραμμικές κλίμακες (linear scales). Για κάθε χαρακτηριστικό υπάρχει και μία αντίστοιχη γραμμική κλίμακα, που περιέχει ζεύγη τιμών κλειδιών και αντίστοιχων διευθύνσεων προς τον κατάλογο. Οι κλίμακες αποθηκεύονται στην κύρια μνήμη και δείχνουν ποιό $(k-1)$ -διάστατο τμήμα του καταλόγου περιέχει τους δείκτες προς τους κάδους των εγγραφών.

Ο δικτυωτός κατάλογος είναι ένας πίνακας με διάσταση k (όπου k είναι ο αριθμός των χαρακτηριστικών), που αν είναι μικρός, τότε αποθηκεύεται στην κύρια μνήμη. Το μέγεθος του καταλόγου περιορίζεται τεμαχίζοντας το σύνολο ορισμού κάθε χαρακτηριστικού σε ανάλογα υποσύνολα. Κάθε είσοδος του καταλόγου περιέχει τη διεύθυνση του αντίστοιχου κάδου των εγγραφών. Είναι δυνατόν πολλές εισόδους να περιέχουν την ίδια διεύθυνση. Συνήθως, ο κατάλογος είναι αρκετά μεγάλος και δεν χωρά στην κύρια μνήμη. Υπάρχει ένας περιορισμός όσον αφορά στις εισόδους του καταλόγου, που μπορεί να περιέχουν την ίδια διεύθυνση. Αν οι εισόδους (a_1, a_2, \dots, a_k) περιέχουν τις ίδιες διευθύνσεις με τις εισόδους (b_1, b_2, \dots, b_k) , τότε τις ίδιες διευθύνσεις πρέπει να περιέχουν και οι εισόδους (x_1, x_2, \dots, x_k) , για κάθε x_i όπου $1 \leq i \leq k$ και $\min(a_i, b_i) \leq x_i \leq \max(a_i, b_i)$. Αυτός ο περιορισμός σημαίνει ότι ένας κάδος περιέχει εγγραφές που ανήκουν σε μία τετράγωνη ή κοίλη (rectangular, convex) περιοχή. Στο Σχήμα 11.11 παρουσιάζεται μία περίπτωση που παραβιάζει τον περιορισμό.

A	A	A
B	B	A

Σχήμα 11.11: Λανθασμένη οργάνωση καταλόγου δικτυωτού αρχείου.

Αν ο κατάλογος είναι σχετικά μικρός και μπορεί να αποθηκευθεί στην κύρια μνήμη, τότε η ικανοποίηση μίας απλής ερώτησης είναι εύκολη υπόθεση. Πρώτα, χρησιμοποιείται η γραμμική κλίμακα για να εντοπισθεί η είσοδος του καταλόγου που περιέχει το δείκτη προς το δίσκο. Η είσοδος του καταλόγου μπορεί να βρεθεί εύκολα χρησιμοποιώντας τις γνωστές τεχνικές από το βιβλίο των Δομών Δεδομένων για την επεξεργασία πινάκων. Με την εύρεση του δείκτη και την επεξεργασία του κάδου στην κύρια μνήμη ελέγχεται αν πράγματι η ζητούμενη εγγραφή είναι αποθηκευμένη στο αρχείο.

Αν η υποβαλλόμενη ερώτηση είναι ερώτηση διαστήματος, τότε και πάλι η επεξεργασία δεν είναι δύσκολη. Έστω, για παράδειγμα, ότι ζητείται να

βρεθούν όλες οι πόλεις που απέχουν από τη Μόσχα ± 2 μοίρες ως προς το γεωγραφικό πλάτος και το γεωγραφικό μήκος. Άρα η ερώτηση μπορεί να τεθεί ως:

$$53^{\circ} 45' \text{ βόρεια} \leq \text{μήκος} \leq 57^{\circ} 45' \text{ βόρεια}$$

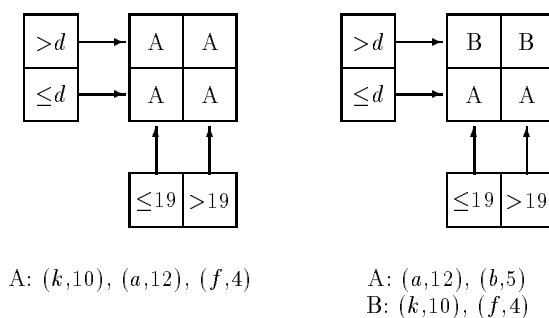
και

$$35^{\circ} 37' \text{ ανατολικά} \leq \text{πλάτος} \leq 39^{\circ} 37' \text{ ανατολικά}$$

Η επεξεργασία αρχίζει από τις γραμμικές κλίμακες για να εντοπισθεί ποιές είσοδοι του καταλόγου περιέχουν αυτές τις τιμές. Το αποτέλεσμα της επεξεργασίας είναι ένα σύνολο εισόδων. Από τον κατάλογο βρίσκονται οι δείκτες προς τη δευτερεύουσα μνήμη. Μάλιστα, είναι πολύ πιθανό ότι μερικοί δείκτες θα περιέχονται περισσότερο από μία φορά στον κατάλογο, αλλά προφανώς θα γίνει μόνο μία προσπέλαση στο δίσκο.

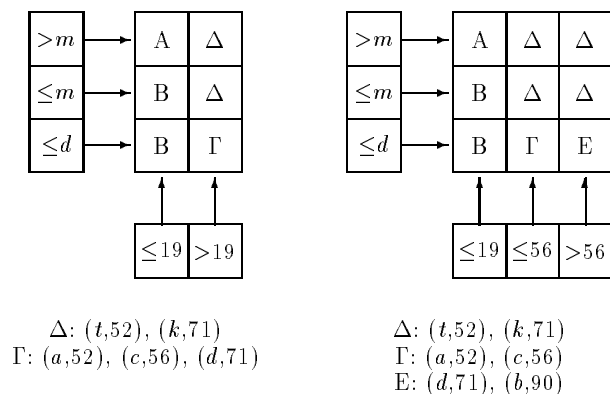
Η διαδικασία της εισαγωγής χρειάζεται ιδιαίτερη προσοχή. Βέβαια, αν ο κάδος όπου θα γίνει η εισαγωγή διαθέτει ακόμη κενό χώρο δεν είναι δύσκολη. Όμως αν ο κάδος υπερχειλίζει, τότε διακρίνονται δύο περιπτώσεις, μία απλή και μία σύνθετη. Και στις δύο περιπτώσεις στο αρχείο παραχωρείται ένας ακόμη κάδος για να στεγάσει μερικές από τις εγγραφές του κάδου που υπερχειλίζει.

Αν υπάρχουν περισσότερες είσοδοι του καταλόγου που αναφέρονται στον κάδο που υπερχειλίζει, τότε οι εγγραφές διανέμονται εκατέρωθεν μίας οριακής τιμής μεταξύ των τιμών της γραμμικής κλίμακας. Επίσης, πρέπει να ενημερωθούν σχετικά και οι δείκτες του καταλόγου, επειδή μερικοί θα αναφέρονται προς το νέο κάδο. Για παράδειγμα, στο Σχήμα 11.12 η χωρητικότητα των κάδων είναι τρεις εγγραφές. Με την εισαγωγή της εγγραφής (b,5) προκύπτει η δεξιά δομή του Σχήματος 11.12.



Σχήμα 11.12: Εισαγωγή με απλή ενημέρωση καταλόγου.

Η πιο σύνθετη περίπτωση παρουσιάζεται όταν ο κάρδος που υπερχειλίζει αναφέρεται από μία μόνο είσοδο του καταλόγου ή όταν οι εγγραφές του κάρδου που υπερχειλίζει ανήκουν σε ένα μόνο διάστημα τιμών της γραμμικής κλίμακας. Στην περίπτωση αυτή εκτός από την επέκταση του αρχείου κατά ένα νέο κάρδο επεκτείνεται και ο κατάλογος. Αυτή η διαδικασία φαίνεται στο Σχήμα 11.13, όπου εισάγεται η εγγραφή $(b,90)$ και ο κατάλογος επεκτείνεται κατά ένα μονοδιάστατο πίνακα. Στη γενική περίπτωση, αν ο κατάλογος είναι k -διάστατος, τότε η επέκταση γίνεται με ένα $(k-1)$ -διάστατο πίνακα. Αξιίζει να σημειωθεί ότι στο παράδειγμα αυτό η διάσπαση του καταλόγου έγινε κατά ένα νοερό κατακόρυφο άξονα. Έπεται ότι σε περίπτωση μελλοντικής διάσπασης του καταλόγου ο νοερός αυτός άξονας θα είναι οριζόντιος. Γενικά, αν υπάρχουν k χαρακτηριστικά, τότε οι διασπάσεις θα γίνονται κατά κυκλικό τρόπο, δηλαδή κάθε φορά ως προς ένα διαφορετικό χαρακτηριστικό της γραμμικής κλίμακας.

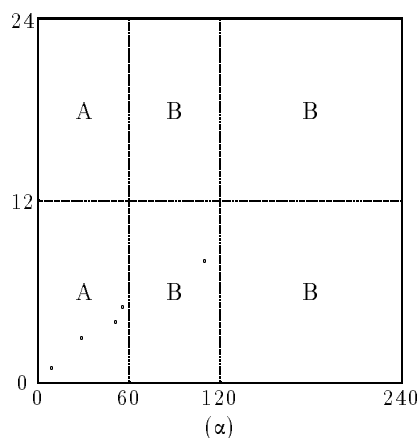


Σχήμα 11.13: Εισαγωγή με επέκταση καταλόγου.

Στο σημείο αυτό προκύπτει και το κυριότερο πρόβλημα του δικτυωτού αρχείου. Έστω ότι η εγγραφή αποτελείται από τρία κλειδιά, ενώ κάθε γραμμική κλίμακα αποτελείται από 100 τιμές. Όταν ο κατάλογος επεκταθεί κατά ένα διδιάστατο πίνακα, στην ουσία προστίθενται 10.000 νέες είσοδοι στον κατάλογο. Αν κάθε είσοδος είναι τέσσερις χαρακτήρες, τότε φαίνεται ότι για μία εισαγωγή εγγραφής απαιτείται επέκταση κατά 40 Kb περίπου που ισοδυναμεί με μία ατράχτο σε ένα σύστημα δίσκων IBM 3380. Αν τα δεδομένα δεν υπαχθούν σε μία ομοιόμορφη κατανομή, τότε αυτή η διαδικασία μπορεί να συμβαίνει σχετικά συχνά. Έτσι ο κατάλογος μπορεί να χρειάζεται σημαντικά περισσότερο χώρο από ότι καταλαμβάνουν οι εγγραφές.

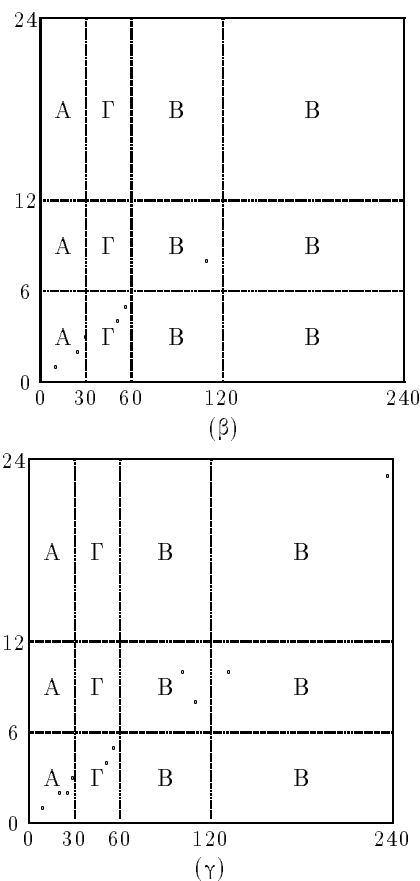
Ακόμη, αν υποβληθεί μία ερώτηση διαστήματος μπορεί να χρειασθεί μία χρονοβόρα επεξεργασία του καταλόγου για να προσπελασθούν τελικά λίγοι μόνο κάδοι του αρχείου.

Πάντως, πρέπει να σημειωθεί ότι (πέρα από το πρόβλημα του καταλόγου) για μία απλή σημειακή ερώτηση που βασίζεται σε οποιοδήποτε κλειδί, απαιτούνται το μέγιστο δύο προσπελάσεις στο δίσκο. Επιπλέον μετά την προσπέλαση των κάδων απαιτείται και επεξεργασία των εγγραφών στην κύρια μνήμη για να διευκρινισθεί πόσες και ποιές εγγραφές ανταποκρίνονται πράγματι την ερώτηση. Από τα προηγούμενα είναι προφανές ότι η ανεπιτυχής αναζήτηση μίας απλής ερώτησης έχει το ίδιο κόστος με την επιτυχή αναζήτηση, δηλαδή δύο προσπελάσεις στο δίσκο. Επίσης είναι ευνόητο ότι οι διαδικασίες ικανοποίησης μίας ερώτησης διαστήματος ή μίας ερώτησης μερικής ταύτισης δεν διαφέρουν σημαντικά από την αντίστοιχη διαδικασία μίας απλής σημειακής ερώτησης.



Σχήμα 11.14: Εισαγωγή δεδομένων Πίνακα 7.2 σε διχτυωτό αρχείο.

Στη συνέχεια τα δεδομένα του Πίνακα 7.2 εισάγονται σε ένα διχτυωτό αρχείο με κάδους χωρητικότητας τεσσάρων εγγραφών. Έστω ότι και πάλι τα δεδομένα εισάγονται στη δομή με αλφαβητική σειρά: A, B, Bh, C, Cy, F, G, R, S, Y. Στο Σχήμα 11.14α παρουσιάζεται η μορφή του διχτυωτού καταλόγου και των αντίστοιχων γραμμικών κλιμάκων μετά την εισαγωγή της πέμπτης εγγραφής (Cy) και τη σχετική επεξεργασία, δηλαδή διάσπαση γραμμικών κλιμάκων, διχτυωτού καταλόγου και αρχικού κάδου σε δύο κάδους A και B, ενώ με σημεία παρουσιάζονται οι πέντε εγγραφές. Στο Σχήμα



Σχήμα 11.14: Εισαγωγή δεδομένων Πίνακα 7.2 σε δικτυωτό αρχείο (συνέχεια).

11.14β παρουσιάζεται η κατάσταση της δομής μετά την εισαγωγή της έκτης εγγραφής (F) και τη σχετική επεξεργασία που καταλήγει στη διάσπαση του κάδου A και τη δημιουργία του κάδου Γ. Στο Σχήμα 11.14γ παρουσιάζεται η τελική κατάσταση, όπου κάθε κάδος περιέχει εγγραφές που ανήκουν σε μία τετράγωνη περιοχή.

Η διαδικασία μετάβασης από την αρχικά κενή δομή του Σχήματος 11.14α στο 11.14β είναι ανάγλυφα παραδείγματα της χρονοβόρας διαδικασίας, που αναφέρθηκε στις προηγούμενες παραγράφους. Αξίζει στο σημείο αυτό να αναφερθεί η συγγένεια αυτής της διαδικασίας των διαδοχικών διαιρέσεων του

χώρου του δικτυωτού καταλόγου με την παθολογική περίπτωση του διπλασιασμού του επεκτατού κατακερματισμού. Σύμφωνα με μία άλλη υλοποίηση ο δικτυωτός κατάλογος μπορεί να μοιάζει με τη μέθοδο του δυναμικού κατακερματισμού. Από τη Regnier (1985) μάλιστα αποδείχθηκε αναλυτικά ότι αν η χωρητικότητα των κάδων είναι λιγότερο από είκοσι εγγραφές ή αν οι τιμές των χαρακτηριστικών δεν υπακούουν σε ομοιόμορφες κατανομές, τότε πρέπει να προτιμηθεί η τελευταία μέθοδος.

Ας υποθεθεί, λοιπόν, ότι με βάση το Σχήμα 11.14 τίθεται η ερώτηση: 'Ποιά βαλκανική χώρα έχει πληθυσμό 10.000.000 κατοίκους και έκταση 132.000 τετραγωνικά χιλιόμετρα;'. Η αναζήτηση ξεκινά από τη γραμμική κλίμακα, όπου ο δείκτης κατευθύνει στην αντίστοιχη είσοδο του δικτυωτού καταλόγου. Η είσοδος αυτή περιέχει ένα δείκτη προς τον κάδο B, που προσπελάζεται από το δίσκο. Ο κάδος B περιέχει τέσσερις εγγραφές, που ελέγχονται στην κύρια μνήμη και τελικώς επιστρέφεται στο χρήστη η σωστή απάντηση. Ας θεωρηθεί τώρα η ερώτηση διαστήματος: 'Ποιές βαλκανικές χώρες έχουν έκταση λιγότερο από 50.000 τετραγωνικά χιλιόμετρα;'. Και πάλι η αναζήτηση ξεκινά από τη γραμμική κλίμακα και το δικτυωτό κατάλογο και στη συνέχεια προσπελάζονται οι κάδοι A και Γ, που περιέχουν τις κατάλληλες εγγραφές.

Η διαγραφή μίας εγγραφής είναι μία πολύπλοκη διαδικασία, αντίστροφη της εισαγωγής, και μπορεί να συνοδεύεται με ελάττωση του αριθμού των κάδων. Όπως και στις οργανώσεις του δυναμικού κατακερματισμού τίθεται από το σχεδιαστή των αρχείων ένα ελάχιστο όριο για τον παράγοντα χρησιμοποίησης του χώρου των κάδων. Αν η τιμή του παράγοντα χρησιμοποίησης γίνει μικρότερη από την προκαθορισμένη τιμή, τότε εξετάζεται αν μπορεί να γίνει συγχώνευση. Συγχώνευση μπορεί να γίνει με κάποιον κάδο που δημιουργήθηκε από την ίδια διάσπαση με τον υπ' όψη κάδο (buddy bucket) σε οποιαδήποτε από τις k διαστάσεις και με την προϋπόθεση ότι το περιεχόμενο και των δύο κάδων μπορεί να αποθηκευθεί σε ένα μόνο κάδο. Σύμφωνα με μία άλλη πιο πολύπλοκη τεχνική, συγχώνευση μπορεί να γίνει με οποιοδήποτε από τους δύο γειτονικούς κάδους στις k διαστάσεις (neighbor bucket) αρκεί το αποτέλεσμα να μην παραβιάζει την απαίτηση που υπαγορεύει κάθε κάδος να αντιστοιχεί σε μία τετράγωνη περιοχή του δικτυωτού καταλόγου. Βέβαια, εκτός από τη συγχώνευση των κάδων πρέπει να γίνει και συρρίκνωση του καταλόγου. Όμως σε πρακτικές περιπτώσεις η συρρίκνωση αποφεύγεται, ώστε να αποφευχθεί το διπλό κόστος σε μία μελλοντική επέκταση του καταλόγου.

Το δικτυωτό αρχείο σε σύγκριση με το πολυδιάστατο δένδρο έχει μία σειρά πλεονεκτημάτων:

- είναι περισσότερο ισοζυγισμένη δομή,
- γενικότερα σε δυναμικά δεδομένα έχει καλύτερες επιδόσεις σε κάθε είδους αναζήτηση,
- επιτυγχάνει κατά μέσο όρο χρήση του χώρου 69% περίπου,
- έχει καλύτερη επίδοση σε περιβάλλον πολλών χρηστών (για λόγους που θα γίνουν αντιληπτοί στο τελευταίο κεφάλαιο του τόμου αυτού).

Όμως υπάρχουν και πλεονεκτήματα του k -d δένδρου όπως:

- η απλότητα του λογισμικού του,
- έχει καλύτερη επίδοση κατά την εισαγωγή και τη διαγραφή,
- έχει την ίδια πολυπλοκότητα και επίδοση ανεξάρτητα από το πλήθος k των χαρακτηριστικών,
- αν το αρχείο είναι σωστά σχεδιασμένο για στατικά δεδομένα, τότε έχει πολύ καλές επιδόσεις σε όλες τις λειτουργίες, καθώς και πολύ καλή χρήση του χώρου.

Το κεφάλαιο αυτό ήταν αφιερωμένο σε οργανώσεις κατάλληλες για μη απλές ερωτήσεις. Από το πλήθος των οργανώσεων αυτών που είναι μεγάλο εξετάστηκαν τα αντεστραμμένα αρχεία, οι πολλαπλές λίστες, οι συνδυασμένοι κατάλογοι, τα πολυδιάστατα δένδρα και το δικτυωτό αρχείο. Οι προηγούμενες δομές είναι σύνθετες και απαιτούν ιδιαίτερη προσοχή για την υλοποίησή τους. Εξ άλλου, και η επακριβής μαθηματική έκφραση της επίδοσής τους είναι πολύπλοκη.

11.7 Ασκήσεις

<1> Δανειστική βιβλιοθήκη διατηρεί πληροφορίες για βιβλία και δανειζόμενους, ώστε να απαντώνται πολύ γρήγορα ερωτήσεις του τύπου:

- 'Ποιά βιβλία που έχει χρεωθεί ο κ.Τάδε',

- 'Είναι το βιβλίο τάδε δανεισμένο, και αν ναι σε ποιόν;'

Να σχεδιασθούν τα αρχεία που θα υποστήριζαν αποτελεσματικά αυτές τις ερωτήσεις.

<2> Δίνεται αρχείο με 100.000 εγγραφές ασθενών των 400 bytes, 200 εγγραφές γιατρών των 400 bytes και 500.000 εγγραφές εργαστηριακών εξετάσεων των 100 bytes. Υπάρχουν 100 είδη εργαστηριακών εξετάσεων. Να σχεδιασθεί μία δομή πολλαπλών λιστών και να κοστολογηθούν οι απαντήσεις των εξής ερωτήσεων:

- 'Ποιοί είναι οι ασθενείς του γιατρού κ.ΑΤάδε',
- 'Ποιές εργαστηριακές εξετάσεις έκανε ο κ.ΒΤάδε',
- 'Ποιές εργαστηριακές εξετάσεις έκαναν οι ασθενείς του κ.ΑΤάδε', και
- 'Ποιά τα αποτελέσματα των εργαστηριακών εξετάσεων χοληστερίνης'.

<3> Για έναν αριθμό a χαρακτηριστικών να βρεθούν οι αλγόριθμοι δημιουργίας:

- των συνδυασμένων καταλόγων,
- των μειωμένων συνδυασμένων καταλόγων, και
- των τροποποιημένων συνδυασμένων καταλόγων.

<4> Να δημιουργηθεί ένα δισδιάστατο δένδρο εισάγοντας τα δεδομένα του Πίνακα 7.2 κατά την αντίστροφη σειρά. Να σχεδιασθούν οι υποδιαίρεσεις του χώρου και να γίνει η κατανομή των εγγραφών στους κατάλληλους κάδους.

<5> Πόσες αντιπροσωπευτικές εγγραφές από ένα αρχείο 1000 εγγραφών χρειάζονται για να δημιουργηθεί ένα τρισδιάστατο δένδρο και ποιο είναι το βάθος του; Ας υποθεθεί ότι η χωρητικότητα των κάδων είναι 10 εγγραφές και ότι το δένδρο είναι περίπου ζυγισμένο.

<6> Να δημιουργηθεί ένα διχτυωτό αρχείο με τα δεδομένα του Πίνακα 7.2 θεωρώντας ότι η χωρητικότητα κάδων είναι τρεις εγγραφές. Να σχεδιασθούν οι υποδιαίρεσεις του διχτυωτού καταλόγου και να γίνει η κατανομή των εγγραφών στους κατάλληλους κάδους.

<7> Να δημιουργηθεί ένα διχτυωτό αρχείο εισάγοντας τα δεδομένα του Πίνακα 7.2 κατά την αντίστροφη σειρά. Να σχεδιασθούν οι υποδιαίρεσεις του διχτυωτού καταλόγου και να γίνει η κατανομή των εγγραφών στους κατάλληλους κάδους.

Κεφάλαιο 12

ΔΟΜΕΣ ΜΕ ΔΥΑΔΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

12.1 Εισαγωγή

12.2 Εξαγωγή υπογραφών

12.3 Δένδρα υπογραφών

12.4 Αναζήτηση με υπογραφές σελίδων

12.5 Κατακερματισμός με υπογραφές

12.6 Φίλτρο Bloome

12.7 Ασκήσεις

Κεφάλαιο 12

ΔΟΜΕΣ ΜΕ ΔΥΑΔΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

12.1 Εισαγωγή

Η εξέλιξη των υπολογιστικών συστημάτων βοήθησε ώστε να δημιουργηθούν διεπιφάνειες χρήστη σε υψηλότερο επίπεδο, δηλαδή με λιγότερη σχέση προς το υλικό της μηχανής. Έτσι η επικοινωνία έγινε φιλικότερη και αυξήθηκε η παραγωγικότητα. Ωστόσο, στο κεφάλαιο αυτό σε μία αντίθετη κατεύθυνση από την εξέλιξη αυτή, θα εξετασθούν οργανώσεις αρχείων που έχουν το δυαδικό ψηφίο (bit) ως δομικό στοιχείο για την αναπαράστασή τους. Το κέρδος από μία τέτοια προσέγγιση είναι διπλό: οικονομία στο χώρο και ταχύτητα στην επεξεργασία. Κατά την υλοποίηση των δομών αυτών ο προγραμματιστής των εφαρμογών μπορεί είτε να χρησιμοποιήσει μία γλώσσα χαμηλού επιπέδου για την αποθήκευση και την επεξεργασία των δεδομένων είτε να χρησιμοποιήσει μία γλώσσα υψηλού επιπέδου και να μετατρέψει τα bits σε bytes. Ας σημειωθεί επίσης ότι πολλοί υπολογιστές έχουν ενσωματωμένες ταχύτατες διαδικασίες αναζήτησης σε δυαδικές συμβολοσειρές (bitstring) μεγέθους byte, λέξης κλπ.

Η οργάνωση και επεξεργασία βάσεων κειμένων (textual databases) είναι ένα ευρύ αντικείμενο που εξετάζεται κυρίως από τον κλάδο της Ανάκτησης Πληροφοριών. Βέβαια, στον αναγνώστη είναι ήδη γνωστές δύο κατηγορίες μεθόδων:

- οι μέθοδοι πλήρους σάρωσης κειμένου που εξετάστηκαν στο βιβλίο των Δομών Δεδομένων (αλγόριθμοι Boyer-Moore, Knuth-Morris-Pratt κλπ.), και

- η μέθοδος των αντεστραμμένων καταλόγων που εξετάστηκε στο προηγούμενο κεφάλαιο.

Η κατηγορία των αρχείων υπογραφών (signature files) αποτελεί μία ακόμη μη οικογένεια μεθόδων για την οργάνωση και την επεξεργασία χειμένων, με ιδιαίτερο γνώρισμα ότι είναι δομές με δυαδική αναπαράσταση. Η μέθοδος αυτή δεν είναι νέα. Χρησιμοποιήθηκε για πρώτη φορά με σκοπό την ανάκτηση δεδομένων το 1949 από τον Mooers και αναφέρεται ήδη από το 1963 σε διδακτικά βιβλία (δες βιβλίο Bourne).

Στη συνέχεια του κεφαλαίου θα εξετασθούν δενδρικές και τυχαίες δομές αρχείων που χρησιμεύουν για την οργάνωση και επεξεργασία δομημένων δεδομένων (όπως οι κλασικές εγγραφές που θεωρήθηκαν μέχρι τώρα) καθώς και μη δομημένων δεδομένων (όπως πχ. το ελεύθερο κείμενο ή πολυμεσικά δεδομένα, όπως ήχος, εικόνα, κινούμενη εικόνα κλπ). Τα θέματα της υλοποίησης των σύνθετων αυτών δομών δεν θα εξετασθούν σε βάθος, αλλά περισσότερη έμφαση θα δοθεί στην παρουσίαση των εννοιών.

12.2 Εξαγωγή υπογραφών

Πριν εξετασθεί οποιαδήποτε δομή καταλόγου που στηρίζεται σε υπογραφές (signatures), πρέπει πρώτα να εξετασθεί η έννοια της υπογραφής των δεδομένων και οι τρόποι εξαγωγής της υπογραφής (signature extraction) από τα δεδομένα, είτε δομημένα είτε μη δομημένα.

Αρχικά, λοιπόν, ας θεωρήσουμε την περίπτωση του ελεύθερου (μη δομημένου) κειμένου. Η μέθοδος των υπογραφών μπορεί να χρησιμοποιηθεί σε πληθώρα εφαρμογών οργάνωσης και επεξεργασίας βάσεων χειμένων, όπως για παράδειγμα σε εγχυκλοπαίδειες, νομικά κείμενα, γραφεία πατεντών κλπ. Επίσης, μπορεί να χρησιμοποιηθεί και σε πολυμεσικά συστήματα (multimedia systems) εξάγοντας υπογραφές από τα διάφορα πολυμεσικά χαρακτηριστικά. Εν πάσει περιπτώσει, στο περιβάλλον της οργάνωσης και επεξεργασίας κειμένου η μέθοδος εφαρμόζεται ως εξής.

Το πλήρες κείμενο σαρώνεται ώστε να εξαλειφθούν οι κοινές λέξεις (για παράδειγμα, άρθρα, προθέσεις κλπ.) και να απομείνουν οι λεγόμενες `χρήσιμες` λέξεις. Οι χρήσιμες λέξεις μετατρέπονται στη ρίζα τους (δηλαδή ενικός αριθμός, ονομαστική πτώση κλπ). Οι λέξεις που χωρούν σε μία φυσική σελίδα του δίσκου (physical block) θεωρείται ότι αποτελούν μία λογική

ομάδα (logical block). Σε κάθε λέξη μίας λογικής ομάδας εφαρμόζεται κάποια τεχνική βασισμένη στον κατακερματισμό, οπότε η λέξη μετατρέπεται σε μία δυαδική συμβολοσειρά σταθερού μήκους, η οποία περιέχει ένα μέγιστο σταθερό αριθμό άσσων. Η συμβολοσειρά αυτή είναι η υπογραφή της συγκεκριμένης λέξης και αποτελεί μία αφαιρετική 'περίληψή' του.

Μία συνηθισμένη πρακτική για την απεικόνιση λέξεων σε δυαδικές συμβολοσειρές είναι η εξής. Θεωρούμε τα γράμματα της λέξης κατά κυλιόμενες τριάδες. Δηλαδή, από τη λέξη signature λαμβάνονται οι τριάδες sig, ign, gna, nat, atu, tur και ure. Κατόπιν, σε κάθε τριάδα εφαρμόζεται μία συνάρτηση κατακερματισμού. Για παράδειγμα, λαμβάνοντας το άθροισμα των κωδικών ASCII των γραμμάτων κάθε τριάδας, για τις προηγούμενες τριάδες προκύπτει το σύνολο των αριθμών (83+73+71), (73+71+78), (71+78+65), (78+65+84), (65+84+85), (84+85+82) και (85+82+69), δηλαδή: 227, 232, 214, 227, 234, 251 και 236. Παρατηρούμε ότι η πρώτη και η τέταρτη τριάδα (δηλαδή, sig και nat), παρ' ότι είναι διαφορετικές φθάνουν στο ίδιο αποτέλεσμα. Ωστόσο, μία τέτοια κατάσταση μπορεί να προκύψει με οποιαδήποτε συνάρτηση κατακερματισμού.

Κατόπιν, οι αχέραιοι της προηγούμενης ομάδας εισάγονται σε μία νέα συνάρτηση κατακερματισμού ώστε να προκύψει η υπογραφή της λέξης. Για παράδειγμα, αν το μήκος της υπογραφής είναι 50 bits, τότε θεωρώντας τη συνάρτηση $\text{mod } 50$ καταλήγουμε στο σύνολο των αριθμών: 27, 32, 14, 27, 34, 1, 36. Έτσι, από τα 50 bits της υπογραφής, τα οποία είναι αριθμημένα από 0 μέχρι 49 και αρχικοποιημένα με 0, θα μετατρέψουμε σε άσσους το 1ο, το 14ο, 27ο, το 32ο, το 34ο και το 36ο. Επομένως, τελικά η υπογραφή της λέξης signature είναι η συμβολοσειρά

0100000000000010000000000001000010101000000000000

δηλαδή, ο αριθμός των άσσων είναι έξι (και όχι επτά). Ο αριθμός αυτός ονομάζεται βάρος (weight) της υπογραφής. Επίσης, είναι προφανές ότι λόγω της φύσης του κατακερματισμού, είναι δυνατόν δύο διαφορετικές λέξεις να έχουν την ίδια υπογραφή, ενώ επίσης δεν είναι δυνατόν από μία υπογραφή να βγει συμπέρασμα για την αρχική λέξη εφαρμόζοντας μία αντίστροφη διαδικασία.

Στη συνέχεια, οι επιμέρους υπογραφές των λέξεων μίας συγκεκριμένης λογικής ομάδας συνδυάζονται με τη λογική πράξη oring, οπότε δημιουργείται μία μόνο υπογραφή σε επίπεδο λογικής ομάδας χειμένου. Η πράξη αυτή

Αρχείο	001	100	001	010
Εγγραφή	000	101	011	000
Πεδίο	001	000	001	110
Υπογραφή	001	101	011	110

Σχήμα 12.1: Κωδικοποίηση με υπέρθεση.

λέγεται επίσης **κωδικοποίηση με υπέρθεση** (superimposed coding). Ένα παράδειγμα της κωδικοποίησης αυτής παρουσιάζεται στο Σχήμα 12.1.

Για να είναι αποτελεσματική η αναζήτηση σε υπογραφές πρέπει οι υπογραφές να έχουν εξαχθεί κατά το βέλτιστο τρόπο. Έχει αποδειχθεί αναλυτικά ότι μία υπογραφή φέρει τη μέγιστη ποσότητα πληροφορίας αν ο αριθμός των άσσων ισούται με τον αριθμό των μηδενικών. Αυτό επιτυγχάνεται αν ισχύει η σχέση:

$$F \times \ln 2 = m \times D$$

όπου F είναι το μήκος της υπογραφής σε bits, m το βάρος σε κάθε υπογραφή λέξης, ενώ D είναι το πλήθος των λέξεων που αποτελούν μία λογική ομάδα. Επομένως, το βάρος της υπογραφής της κάθε λέξης πρέπει να είναι αρκετά μικρότερο από το βάρος της υπογραφής της λογικής ομάδας. Στην αντίθετη περίπτωση, η υπογραφή που θα προκύψει μετά την υπέρθεση θα είναι γεμάτη από άσσους και δεν θα έχει διακριτική ικανότητα. Στο παράδειγμα του Σχήματος 12.1 οι τιμές των παραμέτρων είναι: μήκος υπογραφής $F=12$ bits, βάρος υπογραφής λέξεων $m=4$ και πληθικός αριθμός λογικής ομάδας $D=3$. Έτσι, μετά την υπέρθεση το βάρος της υπογραφής στο επίπεδο των τριών λέξεων είναι έξι (δηλαδή, δώδεκα δια δύο).

Πλέον, η υπογραφή αυτή αποτελεί τη μονάδα σύγκρισης κατά τις αναζητήσεις. Για τη διαπίστωση αν μία λέξη ανήκει σε μία λογική ομάδα ακολουθείται η εξής διαδικασία. Εξάγεται η υπογραφή της συγκεκριμένης λέξης και συγκρίνεται με την υπογραφή της λογικής ομάδας, bit προς bit. Αν κάποιος άσος της υπογραφής της λέξης αντιστοιχεί σε μηδέν της υπογραφής της ομάδας, τότε είναι σαφές ότι η λέξη αυτή δεν συμπεριλαμβάνεται μεταξύ των λέξεων της λογικής ομάδας. Όμως αν όλοι οι άσσοι της υπογραφής της λέξης αντιστοιχούν κάποιους από τους άσσους της υπογραφής της λογικής ομάδας, τότε συμπεραίνεται ότι πιθανώς η λέξη αυτή να ανήκει στη λογική ομάδα. Στην περίπτωση αυτή είναι αναγκαίο να εξετασθούν όλες οι λέξεις της λογικής ομάδας, ώστε να υπάρξει θετική ή αρνητική απάντηση με

βεβαιότητα. Αν τελικά η συγκεκριμένη λέξη δεν ανήκει στη λογική ομάδα, τότε λέγεται ότι συνέβη μία **λανθασμένη πτώση** (false drop).

Ωστόσο, η μέθοδος των υπογραφών μπορεί να εφαρμοσθεί και σε δομημένα δεδομένα, δηλαδή σε εγγραφές που διακρίνονται σε διάφορα επιμέρους πεδία. Για παράδειγμα, έστω η εγγραφή ενός υπαλλήλου που περιλαμβάνει τα εξής τρία πεδία: Όνομα, Φύλλο και Μισθός. Με τη βοήθεια μίας συνάρτησης κατακερματισμού η τιμή κάθε πεδίου μίας εγγραφής μετατρέπεται σε μία επιμέρους υπογραφή δεδομένου σταθερού μήκους. Βέβαια, δεν είναι αναγκαίο οι υπογραφές αυτές να έχουν το ίδιο μήκος, ούτε οι συναρτήσεις κατακερματισμού να είναι ίδιες για όλα τα πεδία. Έτσι, το πεδίο Όνομα μπορεί να παρασταθεί με τρία bits, το πεδίο Φύλλο μπορεί να παρασταθεί με ένα bit, ενώ το πεδίο Μισθός με δύο bits, εφαρμόζοντας κάθε φορά μία διαφορετική συνάρτηση. Η συνολική υπογραφή της εγγραφής σχηματίζεται από τις τρεις επιμέρους υπογραφές με **παράθεση** (concatenation), οπότε προκύπτει μία υπογραφή μήκους 6 bits, όπως παρουσιάζεται στο Σχήμα 12.2. Από το σημείο αυτό μπορεί να ακολουθηθεί η διαδικασία που περιγράφηκε προηγουμένως για την περίπτωση του ελεύθερου χειμένου. Δηλαδή, οι υπογραφές των εγγραφών που ανήκουν σε μία σελίδα του δίσκου μπορούν να συνδυασθούν με υπέρθεση, ώστε να εξαχθεί η υπογραφή της σελίδας.



Σχήμα 12.2: Εξαγωγή υπογραφής από εγγραφή.

Ο Roberts (1975) χρησιμοποίησε ένα απλό σειριακό αρχείο για την οργάνωση των δεδομένων ενός τηλεφωνικού καταλόγου. Μία χρήσιμη παρατήρηση του ήταν ότι το αρχείο των υπογραφών θα μπορούσε να αποθηκευθεί κατά «φέτες» (bit-slices) σε ξεχωριστές δομές, δηλαδή πρώτα όλα τα πρώτα bits όλων των υπογραφών, ύστερα τα δεύτερα bits όλων των υπογραφών κ.ο.κ. Το αποτέλεσμα αυτής της μεθόδου ήταν πολύ καλή επίδοση κατά την αναζήτηση με αντιστάθμισμα μέτρια επίδοση κατά την ανανέωση. Στο πρόβλημα της φυσικής αποθήκευσης ενός αρχείου υπογραφών έχουν δοθεί αρκετές λύσεις, που όμως δεν θα εξετασθούν στα πλαίσια του βιβλίου αυτού. Όμως, αναφέρεται ότι οι υπογραφές των λέξεων είναι αραιές, με την έννοια

ότι περιέχουν πολλά μηδενικά, και μπορεί να εφαρμοσθεί κάποια μέθοδος συμπίεσης από αυτές που θα εξετασθούν σε επόμενο κεφάλαιο. Ο αναγνώστης μπορεί να ανατρέξει στις αναφορές για περισσότερα ενδιαφέροντα στοιχεία (Faloutsos 1985).

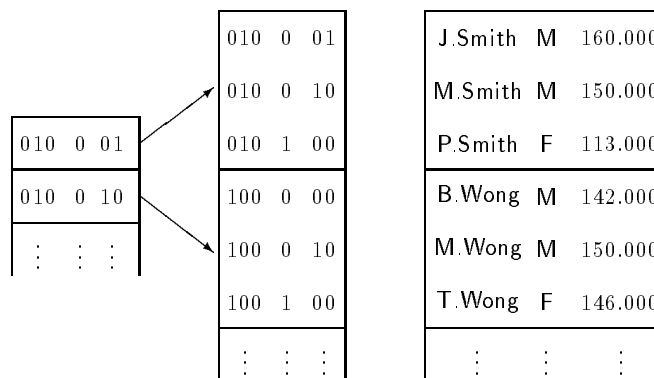
12.3 Δένδρα υπογραφών

Ερωτήσεις μερικής ταύτισης μπορεί να ικανοποιηθούν με αρκετές από τις δομές που εξετάστηκαν στο προηγούμενο κεφάλαιο. Στη συνέχεια θα θεωρηθεί το παράδειγμα της τελευταίας παραγράφου, δηλαδή του αρχείου με εγγραφές που αποτελούνται από τα τρία γνωστά πεδία, και από τις οποίες με παράθεση και υπέρθεση σχηματίζονται οι αντίστοιχες υπογραφές. Στο αρχείο αυτό θα εξετασθεί η περίπτωση ερωτήσεων μερικής ταύτισης με τη βοήθεια των υπογραφών.

Έστω, λοιπόν, ότι κατ' αρχήν οι εγγραφές αποθηκεύονται σε ένα σειριακό αρχείο. Οι υπογραφές τους σχηματίζουν ένα δεύτερο σειριακό αρχείο που είναι μία συμπυκνωμένη έκφραση του κύριου αρχείου και ονομάζεται **αρχείο υπογραφών** (signature file). Επίσης, έστω ότι τίθεται μία απλή ερώτηση ως προς οποιοδήποτε από τα τρία πεδία ή μία ερώτηση μερικής ταύτισης ως προς οποιονδήποτε συνδυασμό των τριών πεδίων λαμβανομένων ανά δύο (ή και μία ερώτηση επακριβούς ταύτισης). Αρχικά οι τιμές των υπ' όψη πεδίων που αναζητώνται με την ερώτηση μετατρέπονται στις αντίστοιχες υπογραφές και παρατίθενται για το σχηματισμό της συνολικής υπογραφής της ερώτησης. Αν δεν υπάρχει ενδιαφέρον ως προς κάποιο πεδίο (ή κάποια πεδία) από τα τρία, τότε οι τιμές των bits στις αντίστοιχες θέσεις θεωρούνται αδιάφορες (don't care bits) και συμβολίζονται με ερωτηματικό (?). Έτσι, η αναζήτηση αρχίζει με την προσπέλαση του αρχείου υπογραφών, όπου η προσπέλαση είναι σειριακή. Αν η υπογραφή της αναζητούμενης εγγραφής ταυτισθεί με κάποια από τις υπογραφές του αρχείου, τότε πρέπει να προσπελασθεί το κύριο αρχείο για να διαπιστωθεί αν πραγματικά η αντίστοιχη εγγραφή ικανοποιεί το χρήστη ή αποτελεί μία λανθασμένη πτώση. Η αναζήτηση τελειώνει με την εξάντληση των υπογραφών του αντίστοιχου αρχείου.

Η επίδοση μπορεί να βελτιωθεί αν κατασκευασθεί ένας κατάλογος, όπως συμβαίνει με την κατασκευή ενός καταλόγου στα απλά σειριακά αρχεία. Η δομή αυτή αναλογικά προς τη μέθοδο ISAM ονομάζεται μέθοδος IDAM

(Indexed Descriptor Access Method). Ο κατάλογος στην περίπτωση αυτή σχηματίζεται εξάγοντας από τις υπογραφές ενός σταθερού αριθμού διαδοχικών αριθμών μία νέα υπερ-υπογραφή με τη μέθοδο της κωδικοποίησης με υπέρθεση. Έτσι, από το σύνολο των υπογραφών προκύπτει ένα νέο σύνολο υπερυπογραφών υποπολλαπλάσιου μεγέθους. Στο Σχήμα 12.3 παρουσιάζεται ένα δένδρο υπογραφών (signature tree) με δύο επίπεδα, όπου από κάθε τρεις υπογραφές εξάγεται μία υπερ-υπογραφή. Γενικά η διαδικασία αυτή μπορεί να επαναληφθεί για όσα επίπεδα χρειάζεται, ώστε εφαρμόζοντας την κωδικοποίηση με υπέρθεση να προκύψει ένα σύνολο υπερ-...-υπερ-υπογραφών που να χωρά για αποθήκευση στην κύρια μνήμη. Σε πρακτικές υλοποιήσεις το μήκος των υπογραφών μπορεί να είναι από 100 ως 200 bytes, ενώ ο παράγοντας ομαδοποίησης είναι της τάξης του 100. Με άλλα λόγια το μέγεθος του δένδρου των υπογραφών είναι μία επιβάρυνση της τάξης του 10% περίπου σε σχέση με το μέγεθος του κύριου αρχείου.



Σχήμα 12.3: Δένδρο υπογραφών.

Ο τρόπος αναζήτησης στο δένδρο αυτό για ερωτήσεις με βάση δευτερεύον κλειδί αλλά και γενικότερα ερωτήσεις μερικής ταύτισης είναι πλέον προφανής. Ας σημειωθεί επίσης ότι μία ανεπιτυχής αναζήτηση μπορεί να τερματισθεί στα ανώτερα επίπεδα του δένδρου χωρίς να καταστεί αναγκαία η προσπέλαση στο κύριο αρχείο.

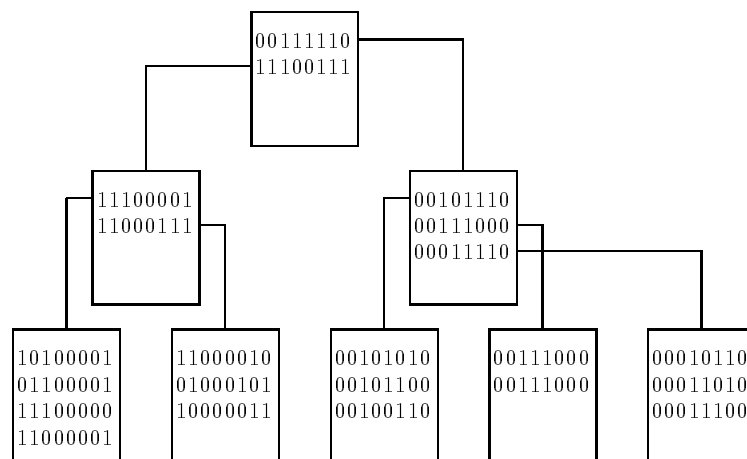
Ωστόσο, η προσέγγιση της δομής IDAM είναι στατική, όπως εξ άλλου στατική είναι και η δομή των αρχείων ISAM. Στη συνέχεια θα εξετασθεί μία ακόμη δεινρική δομή για υπογραφές, η οποία έχει τα χαρακτηριστικά του B-δένδρου, δηλαδή είναι ισοζυγισμένη και διακρίνεται από δυναμικότητα που

εκφράζεται με διασπάσεις κόμβων μετά από εισαγωγή και υπερχειλίση, και επαναεισαγωγές εγγραφών μετά από διαγραφή και υποχειλίση. Η δομή αυτή ονομάζεται S-δένδρο (S-tree) και προτάθηκε από τον Deppisch το 1986.

S-δένδρο τάξης (k, K) είναι το ετερογενές δένδρο με τα ακόλουθα χαρακτηριστικά:

- η ρίζα περιέχει (εκτός αν είναι φύλλο) τουλάχιστο δύο ζεύγη και το μέγιστο K ζεύγη του τύπου (p, s) , όπου p είναι ένας δείκτης προς ένα παιδί, s είναι η υπογραφή του συγκεκριμένου κόμβου, ενώ η υπογραφή αυτή παράγεται με υπέρθεση των υπογραφών όλων των παιδιών,
- οι εσωτερικοί κόμβοι (εκτός της ρίζας) περιέχουν το ελάχιστο k ζεύγη και το μέγιστο K ζεύγη του τύπου (p, s) , όπου $1 \leq k \leq K/2$,
- ένας εσωτερικός κόμβος με l ζεύγη έχει l παιδιά, και
- τα φύλλα βρίσκονται στο ίδιο επίπεδο και περιέχουν ζεύγη του τύπου (p', s) , όπου p' είναι ένας δείκτης προς το αντίστοιχο αντικείμενο στο κυρίως αρχείο και s είναι η υπογραφή του σχετικού αντικειμένου.

Επειδή οι υπογραφές παράγονται με κατακερματισμό, μία υπογραφή μπορεί να είναι αποθηκευμένη περισσότερο από μία φορά μέσα στο S-δένδρο. Επίσης, σημειώνεται ότι δεν υπάρχει καμία διάταξη για τα ζεύγη των κόμβων. Ένα παράδειγμα S-δένδρου παρουσιάζεται στο Σχήμα 12.4, όπου οι υπογραφές αποτελούνται από οκτώ bits, ενώ το βάρος είναι τρία bits.



Σχήμα 12.4: Παράδειγμα S-δένδρου $(k=2, K=4)$.

Τα ποσοτικά χαρακτηριστικά του S-δένδρου είναι διαφορετικά από τα αντίστοιχα του B-δένδρου, αλλά εξάγονται με παρόμοιο τρόπο. Είναι προφανές ότι:

$$U_{min} = k/K$$

και όχι 50% όπως στην περίπτωση του B-δένδρου. Βέβαια, αυτό σημαίνει ότι δεν ισχύει ούτε ότι $E[U]=69\%$. Η τιμή τους ύψους, h , περιορίζεται από τη σχέση:

$$h \leq \lceil \log_k n \rceil - 1$$

όπου n είναι ο αριθμός των υπογραφών, ενώ ο ελάχιστος και ο μέγιστος αριθμός κόμβων δίνεται από τις σχέσεις:

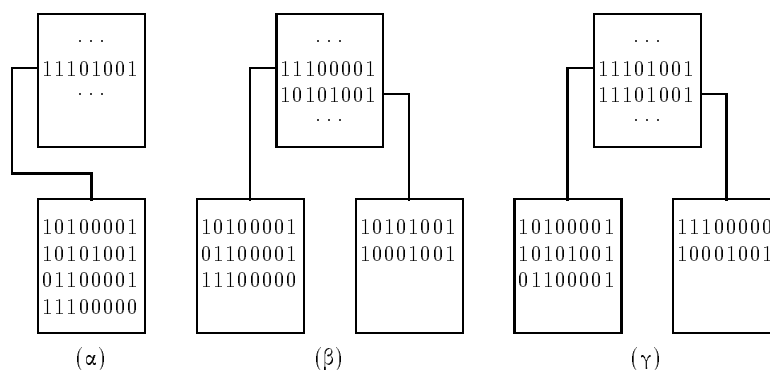
$$nod_{min} = 1 + \sum_{i=0}^{h-2} k^i \quad \quad \quad nod_{max} = 1 + 2 \frac{k^{h-1} - 1}{k - 1}$$

Η διαδικασία αναζήτησης μίας υπογραφής σε ένα S-δένδρο είναι σχεδόν προφανής για τον αναγνώστη. Έστω ότι η αναζήτηση αφορά στην υπογραφή 00100110. Η διαδικασία αρχίζει από τη ρίζα. Εκ των δύο υπογραφών της ρίζας ταιριάζει η 00111110, γιατί έχει άσσους εκεί όπου έχει άσσους και η αναζητούμενη. Έτσι, η διαδικασία συνεχίζει στο δεξιό κόμβο του δευτέρου επιπέδου, ο οποίος περιέχει τρεις υπογραφές. Εξ αυτών των υπογραφών ταιριάζει η 00101110, για τον ίδιο λόγο όπως προηγουμένως. Με τον τρόπο αυτό η διαδικασία καταλήγει στο αντίστοιχο φύλλο, όπου διαπιστώνεται ότι η αναζητούμενη υπογραφή πράγματι υπάρχει. Η διαδικασία που περιγράφηκε αφορά σε μία ερώτηση επακριβούς ταύτισης και ήταν επιτυχής. Αν η αναζητούμενη υπογραφή ήταν η 00001110, τότε θα επρόκειτο για μία ανεπιτυχή αναζήτηση που θα κατέληγε στο δεξιότερο φύλλο.

Με τη δομή των S-δένδρων μπορούν να απαντηθούν και ερωτήσεις μερικής ταύτισης. Κατά τη διάσχιση του δένδρου σε μία τέτοια περίπτωση, το μονοπάτι από τη ρίζα προς τα φύλλα δεν είναι απαραίτητα μοναδικό, αλλά μπορεί να ακολουθηθούν περισσότερα του ενός μονοπάτια. Για παράδειγμα, έστω ότι αναζητώνται οι υπογραφές ??1???1?. Έτσι, από τη ρίζα θα πρέπει να προσπελασθούν και οι δύο κόμβοι του δευτέρου επιπέδου. Εξετάζοντας το περιεχόμενο των κόμβων αυτών, αρχικά διαπιστώνεται ότι είναι αδύνατο από τον αριστερό κόμβο να βρεθεί κάποια υπογραφή με τις ζητούμενες προδιαγραφές, ενώ από το δεξιό κόμβο υπάρχει μόνο μία υπογραφή με άσσους στην τρίτη, την έκτη και την έβδομη θέση. Επομένως, πρακτικά η διαδικασία συνεχίζεται στο αριστερότερο παιδί του δεξιού κόμβου, όπου

διαπιστώνεται και η τελική απάντηση που αποτελείται από τις υπογραφές 00101010 και 00100110. Γενικώς, όσο λιγότεροι άσσοι προσδιορίζονται σε μία ερώτηση μερικής ταύτισης, τόσο περισσότερα είναι τα μονοπάτια που πρέπει να ακολουθηθούν.

Η διαδικασία της αναζήτησης σε ένα S-δένδρο δεν είναι ιδιαίτερα δύσκολη. Περισσότερο δύσκολη είναι η διαδικασία των εισαγωγών, οπότε διασπώνται οι υπερχειλίζοντες κόμβοι. Η δυσκολία έγκειται στο γεγονός ότι αν υπογραφές δεν κατανεμηθούν στους δύο κόμβους με ένα έξυπνο τρόπο, τότε οι υπογραφές των κόμβων αυτών που θα ανέλθουν στο ανώτερο επίπεδο θα έχουν πολλούς άσσους. Έτσι, σε επόμενες αναζητήσεις θα ακολουθούνται πολλά μονοπάτια του δένδρου ακόμη και αν η υπογραφή της ερώτησης δεν περιέχει λίγους άσσους. Στη συνέχεια δίνεται ένα παράδειγμα καλής και κακής διάσπασης. Έστω ότι στον κόμβο του Σχήματος 12.5α, ο οποίος έχει μέγιστη χωρητικότητα τεσσάρων υπογραφών, εισάγεται η υπογραφή 10001001 και πρέπει να γίνει διάσπαση λόγω υπερχειλίσης. Στο Σχήμα 12.5β και στο Σχήμα 12.5γ παρουσιάζονται δύο εναλλακτικά σενάρια κατανομής των υπογραφών σε δύο κόμβους. Προφανώς, το σενάριο του Σχήματος 12.5γ δεν πρέπει να προτιμηθεί γιατί οι υπογραφές που ανέρχονται στον πατέρα (α) έχουν βάρος έξι άσσων, και (β) είναι ταυτόσημες. Αντίθετα, οι υπογραφές που ανέρχονται στον πατέρα των δύο κόμβων του Σχήματος 12.5β έχουν βάρος τεσσάρων άσσων.



Σχήμα 12.5: Παράδειγμα διάσπασης σε S-δένδρο.

Το πρόβλημα, λοιπόν, της εισαγωγής αντιμετωπίζεται ως εξής. Δεδομένης της υπογραφής προς εισαγωγή, γίνεται μία διάσχιση από τη ρίζα προς τα φύλλα έτσι ώστε σε κάθε επίπεδο να επιλέγεται ως επόμενος κόμβος, ο

κόμβος του οποίου η υπογραφή αν υπερτεθεί με την εισαγόμενη υπογραφή να προκύψει η μικρότερη αύξηση βάρους. Προφανώς, η στρατηγική αυτή αποσκοπεί στην ελαχιστοποίηση των διαφορετικών μονοπατιών που θα πρέπει να διασχισθούν σε μελλοντικές αναζητήσεις. Τελικώς, λοιπόν, προσπελάζεται ένα φύλλο. Αν το φύλλο έχει διαθέσιμο χώρο, τότε η υπογραφή εισάγεται στο φύλλο αυτό και ταυτόχρονα ελέγχεται αν πρέπει να ενημερωθεί η υπογραφή του πατέρα, καθώς και οι υπογραφές των κόμβων των ανωτέρων επιπέδων προς τη ρίζα του δένδρου. Αν το φύλλο όπου καταλήγει η εισαγωγή είναι πλήρες, τότε γίνεται διάσπαση σύμφωνα με την εξής ευριστική τεχνική. Κατ' αρχήν επιλέγονται δύο υπογραφές s_1 και s_2 , που ονομάζονται σπόροι (seed) των δύο σελίδων. Ως σπόρος s_1 επιλέγεται η υπογραφή με το μεγαλύτερο βάρος, ενώ ως σπόρος s_2 επιλέγεται η υπογραφή με το μεγαλύτερο αριθμό άσπων στις θέσεις όπου ο σπόρος s_2 έχει μηδέν. Έτσι, δεδομένης της υπογραφής s_1 , η υπογραφή s_2 είναι εκείνη που διαφέρει περισσότερο προς την s_1 . Ως μέτρο διαφοράς (dissimilarity) επιλέγεται η απόσταση Hamming, δηλαδή ο αριθμός των θέσεων όπου η μία υπογραφή έχει άσσο και η άλλη έχει μηδέν. Αν με $\delta(s_1, s_2)$ συμβολίζεται η απόσταση Hamming δύο υπογραφών s_1 και s_2 , ενώ με γ συμβολίζεται το βάρος μίας υπογραφής, τότε ισχύει η σχέση:

$$\delta(s_1, s_2) = \gamma(s_1 \vee s_2) - \gamma(s_1 \wedge s_2)$$

Κατόπιν, οι υπόλοιπες υπογραφές λαμβάνονται μία-μία με τυχαίο τρόπο και συγκρίνονται με τους δύο σπόρους. Έτσι, κάθε υπογραφή καταλήγει στη σελίδα, όπου με το σπόρο της δίνει ελάχιστη αύξηση βάρους σε περίπτωση υπέρθεσης. Όμως υπάρχει περίπτωση, καθώς γίνεται η ανάθεση των υπογραφών στις δύο σελίδες, κάποια σελίδα να γεμίσει από K υπογραφές. Τότε όλες οι υπόλοιπες αναθέτονται στην άλλη σελίδα χωρίς περαιτέρω εξέταση. Βέβαια, στη συνέχεια πρέπει να εξαχθούν οι υπογραφές των δύο φύλλων και να ανέλθουν στο ανώτερο επίπεδο. Αυτό σημαίνει ότι αν ο πατέρας είναι πλήρης, τότε πρέπει να γίνει νέα διάσπαση σε ανώτερο επίπεδο κοχ. μέχρι πιθανώς να διασπασθεί η ρίζα και να αυξηθεί το ύψος του δένδρου.

Σημειώνεται ότι αυτή η μέθοδος διάσπασης σελίδων δεν είναι η βέλτιστη από την άποψη της δημιουργίας σελίδων (α) με περίπου τον ίδιο αριθμό υπογραφών, οπότε θα αναβάλονταν χρονικά η μελλοντική διάσπαση των ιδίων, και (β) με το χωρισμό τους σε δύο υποσύνολα έτσι ώστε οι υπογραφές των δύο υποσυνόλων να έχουν το ελάχιστο βάρος. Το δεύτερο χαρακτηριστικό θα μπορούσε να ικανοποιηθεί με έναν αλγόριθμο εκθετικής πολυπλοκότητας, ενώ ο αλγόριθμος που περιγράφηκε προηγουμένως είναι γραμμικός.

Οι διαγραφές εκτελούνται επίσης με δυναμικό τρόπο. Δηλαδή, σε περίπτωση διαγραφής είναι δυνατόν να απαιτείται ενημέρωση της υπογραφής του ανωτέρου ή των ανωτέρων επιπέδων μέχρι το επίπεδο της ρίζας. Πλέον σύνθετη είναι η περίπτωση όπου ένας κόμβος μείνει με $k-1$ ζεύγη. Στην περίπτωση αυτή ο κόμβος αποδίδεται στο σύστημα, ενημερώνονται οι υπογραφές του πατέρα (και ίσως αναδρομικά οι κόμβοι στο μονοπάτι προς τη ρίζα), ενώ οι $k-1$ υπογραφές επανα-εισάγονται ώστε να τοποθετηθούν σε νέα φύλλα.

Τα στατικά (IDAM) και τα δυναμικά (S-trees) δένδρα υπογραφών σε σχέση με τα αντεστραμμένα αρχεία και τις πολλαπλές λίστες έχουν συγκεκριμένα πλεονεκτήματα:

- στο δένδρο υπογραφών το κόστος προσπέλασης παραμένει σχεδόν σταθερό ανεξάρτητα από τον αριθμό των πεδίων που καθορίζονται από την ερώτηση μερικής ταύτισης, σε αντίθεση με τις άλλες μεθόδους, όπου αυξάνοντας τον αριθμό των πεδίων αυξάνει και το κόστος απάντησης,
- η μέθοδος των αντεστραμμένων αρχείων και των πολλαπλών λιστών ή υλοποιείται ή δεν υλοποιείται για κάποιο συγκεκριμένο πεδίο. Στα δένδρα υπογραφών αν κάποιο πεδίο είναι πιο σημαντικό από τα υπόλοιπα και αναζητάται περισσότερο συχνά, τότε το μήκος της υπογραφής για το πεδίο αυτό ρυθμίζεται ώστε να είναι μεγαλύτερο και να υπάρχει μεγαλύτερη αξιοπιστία (δηλαδή, λιγότερες λανθασμένες πτώσεις), και τέλος
- ο επιπρόσθετος χώρος για τα δένδρα υπογραφών ποικίλει από 5% ως 40%, ενώ στις άλλες μεθόδους μπορεί να φθάσει το 100% του όγκου του κύριου αρχείου. Βέβαια, υπάρχει και το κόστος ενημέρωσης, που στα δένδρα υπογραφών είναι μεγαλύτερο απ' ό,τι στις άλλες οργανώσεις αν η ενημερώνονται λίγα πεδία (για παράδειγμα ένα ή δύο πεδία), αλλά είναι μικρότερο στην αντίθετη περίπτωση.

12.4 Ανάκτηση με υπογραφές σελίδων

Τα δένδρα υπογραφών της προηγούμενης μεθόδου λειτουργούν σαν ένας πολυεπίπεδος μηχανισμός φίλτρου, ώστε να μειωθεί ο όγκος των υπογραφών που πρέπει να εξετασθούν. Όμως είναι δυνατό χρησιμοποιώντας ένα διαφορετικό τρόπο απεικόνισης στη δευτερεύουσα μνήμη, οι εγγραφές να μην

αποθηκεύονται τυχαία στις σελίδες του αρχείου, αλλά οι εγγραφές με ίδιες τιμές στα διάφορα χαρακτηριστικά να κατευθύνονται στην ίδια σελίδα. Έτσι, σε μία ερώτηση μερικής ταύτισης οι εγγραφές θα είναι συγκεντρωμένες (clustered) και θα προσπελάζονται μόνο οι σχετικές σελίδες αντί να γίνεται σάρωση ολόκληρου του αρχείου. Η ανάκτηση μερικής ταύτισης με υπογραφές σελίδων (partial match retrieval with page signatures), που προτάθηκε από τον Ramamohanarao (1983), εφαρμόζει μία τέτοια τεχνική.

Η μέθοδος βασίζεται σε δύο μηχανισμούς φίλτρων προσπέλασης που βοηθούν, ώστε να γίνει ταχύτερα η αναζήτηση επειδή μειώνεται ο αντίστοιχος χώρος αναζήτησης (φιλοσοφία 'διαίρει και βασίλευε'). Κατ' αρχήν με βάση τις εγγραφές κάθε σελίδας κατασκευάζεται ένα μονοεπίπεδο (σειρακό) αρχείο υπογραφών. Σε δεύτερο στάδιο, από κάθε εγγραφή εξάγεται μία συμβολοσειρά πολύ περιορισμένου μήκους σε σχέση με την προηγούμενη υπογραφή, η οποία δίνει και τη διεύθυνση της σελίδας του αρχείου όπου θα γίνει η αποθήκευση. Η δυαδική αυτή συμβολοσειρά εξάγεται χρησιμοποιώντας νέες συναρτήσεις κατακερματισμού με μικρότερο μήκος συμβολοσειράς για κάθε χαρακτηριστικό. Στη συνέχεια αυτές οι επιμέρους συμβολοσειρές των χαρακτηριστικών παρατίθενται για να σχηματισθεί κατά τα γνωστά μία νέα συμπυκνωμένη υπογραφή της εγγραφής.

Αρχείο υπογραφών	Βασικό αρχείο	Υπογραφή σελίδας	Αριθμός σελίδας
010 0 01	J.Smith M 146.000	000	0
		001	1
010 0 10	M.Smith M 150.000	010	2
010 1 00	P.Smith F 113.000	011	3
100 0 00	B.Wong M 142.000	100	4
100 0 10	M.Wong M 150.000	101	5
		110	6
100 1 00	T.Wong F 146.000	111	7
⋮ ⋮ ⋮	⋮ ⋮ ⋮	⋮	⋮

Σχήμα 12.6: Υπογραφές και διευθύνσεις σελίδων.

Στη συνέχεια θα εξετασθεί και πάλι το προηγούμενο παράδειγμα της εγγραφής που περιλαμβάνει τα πεδία: Όνομα, Φύλλο και Μισθός. Και πάλι με τη βοήθεια μίας συνάρτησης κατακερματισμού η τιμή κάθε πεδίου μίας εγγραφής μετατρέπεται σε μία δυαδική συμβολοσειρά σταθερού μήκους. Έστω, λοιπόν, ότι το κάθε πεδίο παρίσταται με ένα bit, συνεπώς η περιγραφή της σελίδας έχει μήκος τρία bits. Οι έξι γνωστές εγγραφές του Σχήματος 12.3 τοποθετούνται σε οκτώ σελίδες με χωρητικότητα μίας εγγραφής όπως παρουσιάζεται στο Σχήμα 12.6.

Ο τρόπος αναζήτησης στη δομή αυτή για ερωτήσεις με βάση δευτερεύον κλειδί, αλλά και ερωτήσεις μερικής ταύτισης είναι παρόμοια με την προηγούμενη μέθοδο. Πρώτα, λοιπόν, εξάγονται οι δύο συμβολοσειρές σύμφωνα με τα δύο σύνολα των συναρτήσεων κατακερματισμού και διαπιστώνεται σε ποιές πιθανές σελίδες του αρχείου μπορεί να είναι αποθηκευμένη η αναζητούμενη εγγραφή. Στη συνέχεια προσπελάζονται οι κατάλληλες σελίδες από το αρχείο υπογραφών και ελέγχονται οι υπογραφές των αποθηκευμένων εγγραφών του κυρίου αρχείου σε σχέση με την υπογραφή της αναζητούμενης εγγραφής. Σε περίπτωση μη ταύτισης, γίνεται αντιληπτό ότι πρόκειται για ανεπιτυχή αναζήτηση και η διαδικασία τερματίζει. Σε περίπτωση ταύτισης, η διαδικασία συνεχίζεται με προσπέλαση στο κύριο αρχείο για τον τελικό έλεγχο. Έστω, λοιπόν για παράδειγμα, ότι δίνεται μία ερώτηση με βάση το πεδίο Φύλλο που πρέπει να έχει τιμή 'M'. Έτσι, εξάγεται ότι οι διευθύνσεις των σελίδων είναι ?0?, οπότε τελικά προσπελάζονται οι σελίδες 0, 1, 4 και 5. Είναι προφανές ότι όσα περισσότερα χαρακτηριστικά καθορίζονται σε μία ερώτηση μερικής ταύτισης, τόσο ταχύτερα γίνεται ο εντοπισμός των σχετικών σελίδων. Έστω, τώρα, ότι τίθεται μία ερώτηση μερικής ταύτισης, όπου πρέπει να ισχύει: 'Φύλλο=F' και 'Μισθός>145.000'. Στην περίπτωση αυτή εξάγεται ότι οι διευθύνσεις των σελίδων είναι ?11, οπότε προσπελάζονται οι σελίδες 3 και 7. Η συνέχεια της επεξεργασίας είναι πλέον ευνόητη.

Η μέθοδος αυτή παρουσιάζει σημαντικά πλεονεκτήματα σε σχέση με τη μέθοδο του δένδρου υπογραφών:

- η δομή με τις υπογραφές των σελίδων δεν είναι απαραίτητο να βρίσκεται μαζί με το κυρίως αρχείο, οπότε αν κατά την επεξεργασία είναι αποθηκευμένο στην κύρια μνήμη, τότε δεν υπάρχει επιβάρυνση για είσοδο/έξοδο των δεδομένων,
- όταν μία εγγραφή εισάγεται, διαγράφεται ή ανανεώνεται αλλάζει μόνο η υπογραφή της σελίδας, χωρίς να προκαλούνται αλυσιδωτές αλλαγές στις υπογραφές των ανώτερων επιπέδων του δένδρου,

- αν και αρχικά σχεδιάστηκε για στατικά δεδομένα και για ερωτήσεις ανάκτησης με δευτερεύον κλειδί, η ιδέα της υλοποίησης ενός αρχείου περιγραφών μπορεί να συνδυασθεί με τα δυναμικά τυχαία αρχεία, οπότε μπορούν να εξυπηρετηθούν και ερωτήσεις με βάση το πρωτεύον κλειδί.

12.5 Κατακερματισμός με υπογραφές

Από τον Larson (1984) προτάθηκε ένα στατικό αρχείο κατακερματισμού με υπογραφές (signature hashing), που επιτυγχάνει την επιτυχή και ανεπιτυχή αναζήτηση με μία και μόνο μία προσπέλαση στο δίσκο. Είναι αντιληπτό ότι η μέθοδος αυτή είναι πολύ ενδιαφέρουσα γιατί καμία δομή απ' όσες εξετάστηκαν μέχρι τώρα δεν μπορεί να εγγυηθεί την επίδοση αυτή. Για παράδειγμα, πολλές από τις μεθόδους των δυναμικών τυχαίων αρχείων διακρίνονται για αυτήν την επίδοση με την προϋπόθεση ότι ο κατάλογος είναι μικρός και χωρά στην κύρια μνήμη. Βέβαια, στην προκειμένη περίπτωση τίποτε δεν είναι δωρεάν. Τα μειονεκτήματα της μεθόδους είναι:

- η πολυπλοκότητα του λογισμικού της,
- η εισαγωγή μπορεί να κοστίζει αρκετά περισσότερο από μία προσπέλαση στο δίσκο, και
- δεσμεύεται χώρος στην κύρια μνήμη για την αποθήκευση του αρχείου των υπογραφών που έχει το ίδιο μήκος (δηλαδή, αριθμός διευθύνσεων) με το κύριο αρχείο.

Για την κατανόηση της μεθόδου θα εξετασθεί αμέσως ένα παράδειγμα που διασαφηνίζει τις διαδικασίες εισαγωγής και αναζήτησης. Για την ευκολία του παραδείγματος υποτίθεται ότι το κύριο αρχείο αποτελείται από 11 σελίδες με χωρητικότητα μία μόνο εγγραφή ($b=11$). Το ίδιο μήκος έχει και το αρχείο των υπογραφών που ονομάζονται *διαχωριστές* (separators). Υποτίθεται επίσης ότι το μέγεθος των υπογραφών είναι τέσσερα bits, ενώ αρχικά σε κάθε θέση του αρχείου υπογραφών αποθηκεύεται η τιμή 1111. Κατά την εισαγωγή χρησιμοποιείται ως συνάρτηση κατακερματισμού η μέθοδος της διαίρεσης:

$$f(key) = key \bmod b$$

ενώ σε περίπτωση σύγκρουσης ως δεύτερη συνάρτηση χρησιμοποιείται η σχέση:

$$i(key) = \lfloor \frac{key}{b} \rfloor \bmod b$$

ώστε να προκύψει η απόσταση της νέας θέσης του αρχείου που θα πρέπει να εξετασθεί.

Έστω ότι στο αρχείο πρόκειται να εισαχθούν εγγραφές με κλειδιά 52, 19, 71, 56, 68, 5 και 12. Σύμφωνα με τη μέθοδο απαιτείται για κάθε κλειδί να υπολογισθεί μία αντίστοιχη ακολουθία υπογραφών. Μάλιστα κάθε υπογραφή της ακολουθίας αντιστοιχεί σε κάθε διαδοχική εξέταση που μπορεί να γίνει για ένα κλειδί εξαιτίας των συγκρούσεων. Δηλαδή, η μέθοδος αυτή μοιάζει με τη μέθοδο της ανοικτής διεύθυνσης γιατί δεν χρησιμοποιεί δείκτες ή ψευδοδείκτες. Οι υπογραφές συνήθως εξάγονται από το κλειδί με τη βοήθεια μίας ψευδοτυχαίας γεννήτριας αριθμών που είναι μοναδική για κάθε κλειδί. Ωστόσο, για την ευκολία του παραδείγματος στη συνέχεια θα χρησιμοποιηθεί και πάλι μία συνάρτηση κατακερματισμού, όπως για παράδειγμα η σχέση:

$$s_1(key) = key \bmod 15$$

όπου το 15 χρησιμοποιείται ως διαιρέτης γιατί το μήκος των υπογραφών είναι τέσσερα bits, ενώ γενικά για κάθε επόμενη προσπάθεια εξέτασης θα χρησιμοποιηθεί η σχέση:

$$s_i(key) = ((s_{i-1}(key) + 1) \times key) \bmod 15$$

Έτσι, στον Πίνακα 12.1 φαίνεται η ακολουθία των υπογραφών των προη-

Κλειδί	Εξέταση			
	1	2	3	4
52	7 (0111)	11 (1001)	9 (1001)	10 (1010)
19	4 (0100)	5 (0101)	9 (1001)	10 (1010)
71	11 (1011)	12 (1100)	8 (1000)	9 (1001)
56	11 (1011)	12 (1100)	8 (1000)	9 (1001)
68	8 (1000)	12 (1100)	14 (1110)	0 (0000)
5	10 (1010)	10 (1010)	10 (1010)	10 (1010)
12	12 (1100)	6 (0110)	9 (1001)	0 (0000)

Πίνακας 12.1: Ακολουθία υπογραφών κλειδιών.

γούμενων κλειδιών. Οι υπογραφές αυτές ονομάζονται **υπογραφές θέσης-κλειδιού** (position-key signatures). Είναι φανερό ότι αν προκύψει μηδέν, τότε στη συνέχεια προκύπτει και πάλι η ίδια ακολουθία τιμών. Ο λόγος που χρησιμοποιείται μία ολόκληρη ακολουθία τιμών υπογραφών είναι ο διαφορισμός δύο συνωνύμων σε μία επόμενη εξέταση.

Όταν πρόκειται να εισαχθεί μία εγγραφή, τότε κατ' αρχήν εξάγεται η υπογραφή της και με τη βοήθεια της συνάρτησης κατακερματισμού εντοπίζεται η κατάλληλη θέση του αρχείου υπογραφών. Αν η υπογραφή της εγγραφής είναι μεγαλύτερη ή ίση προς την αντίστοιχη υπογραφή του αρχείου, τότε με τη βοήθεια της δεύτερης συνάρτησης κατακερματισμού εντοπίζεται η νέα θέση του αρχείου. Έτσι, τη φορά αυτή η δεύτερη υπογραφή του κλειδιού από τη γνωστή ακολουθία υπογραφών συγκρίνεται με την αντίστοιχη υπογραφή της νέας θέσης του αρχείου. Η διαδικασία αυτή συνεχίζεται μέχρις ότου η υπογραφή του κλειδιού είναι μικρότερη από την αντίστοιχη του αρχείου, οπότε προσπελάζεται το κύριο αρχείο. Αν η θέση του αρχείου αυτού είναι κενή, τότε η εγγραφή αποθηκεύεται και η διαδικασία τελειώνει. Αν κάποια άλλη εγγραφή είναι αποθηκευμένη στη θέση αυτή, τότε τελικά τη θέση καταλαμβάνει εκείνη η εγγραφή που έχει τη μικρότερη υπογραφή, ενώ η αντίστοιχη θέση του αρχείου υπογραφών ενημερώνεται με την τιμή της μεγαλύτερης από τις δύο υπογραφές των υπ' όψη εγγραφών. Έτσι, η διαδικασία συνεχίζεται για την εισαγωγή της άλλης εγγραφής σε μία νέα θέση.

Η διαδικασία που αναπτύχθηκε θεωρητικά φαίνεται καλύτερα στο Σχήμα 12.7 που δείχνει την εξέλιξη της δομής με τις διαδοχικές εισαγωγές των εγγραφών με τα προηγούμενα κλειδιά. Η εγγραφή 52 κατευθύνεται στη θέση 8. Η υπογραφή της 0111 είναι μικρότερη από την τιμή 1111 που είναι αποθηκευμένη στην αντίστοιχη θέση του αρχείου υπογραφών. Άρα γίνεται προσπέλαση στη θέση 8 του αρχείου και η εγγραφή αποθηκεύεται γιατί η θέση είναι κενή. Το αποτέλεσμα παρουσιάζεται στο Σχήμα 12.7α. Στη συνέχεια εισάγεται η εγγραφή 19 που επίσης κατευθύνεται στη θέση 8. Η υπογραφή της θέσης είναι 1111, άρα προσπελάζεται η θέση 8 και διαπιστώνεται ότι είναι κατειλημμένη. Μία από τις δύο εγγραφές πρέπει να μείνει στη θέση και η άλλη να φύγει. Μένει η εγγραφή 19 γιατί έχει μικρότερη υπογραφή (δηλαδή 0100 σε σχέση με το 0111), ενώ η εγγραφή 52 απομακρύνεται κατά $\lfloor \frac{52}{11} \rfloor \bmod 11 = 4$ θέσεις, και αποθηκεύεται στη θέση $(8+4) \bmod 11 = 1$. Ταυτόχρονα ενημερώνεται η υπογραφή της θέσης 8 από 1111 σε 0111, όπως φαίνεται στο Σχήμα 12.7β. Η εγγραφή 71 εισάγεται εύκολα στη θέση 5. Κατόπιν η εγγραφή 56 κατευθύνεται στη θέση 1, όπου η υπογραφή

0	1111		1111		1111		1111	71	1111	71
1	1111		1111	52	1011	52	1011	52	1011	52
2	1111		1111		1111		1111	68	0100	12
3	1111		1111		1111		1111		1111	68
4	1111		1111		1111		1111		1111	
5	1111		1111		1111	71	1011	5	1011	5
6	1111		1111		1111	56	1111	56	1111	56
7	1111		1111		1111		1111		1111	
8	1111	52	0111	19	0111	19	0111	19	0111	19
9	1111		1111		1111		1111		1111	
10	1111		1111		1111		1111		1111	
	(α)		(β)		(γ)		(δ)		(ε)	

Σχήμα 12.7: Εισαγωγές σε αρχείο κατακερματισμού με υπογραφές.

είναι 1111, δηλαδή μεγαλύτερη από την υπογραφή 1011 της συγχεκριμένης εγγραφής. Άρα προσπελάζεται η θέση 1, που είναι κατειλημμένη από την εγγραφή 52. Η εγγραφή 52 παραμένει στη θέση αυτή γιατί έχει μικρότερη υπογραφή (δηλαδή 0111 έναντι 1011). Η εγγραφή 56 αφήνει την υπογραφή της στη θέση 1 και αποθηκεύεται μετά από $\lfloor \frac{56}{11} \rfloor \bmod 11 = 5$ θέσεις, δηλαδή στη θέση 6 όπως φαίνεται στο Σχήμα 12.7γ. Η εγγραφή 68 αποθηκεύεται χωρίς πρόβλημα στη θέση 2. Όμως πρόβλημα παρουσιάζεται στην εισαγωγή της εγγραφής 5, που κατευθύνεται στη θέση 5 που είναι κατειλημμένη από την εγγραφή 71. Εξ των δύο αυτών εγγραφών, η εγγραφή 5 αποθηκεύεται στη θέση 5 γιατί έχει μικρότερη υπογραφή από την εγγραφή 71 (1010 έναντι 1011), και η εγγραφή 71 αποθηκεύεται στη θέση 0 εφαρμόζοντας τη δεύτερη συνάρτηση κατακερματισμού, ενώ ταυτόχρονα ενημερώνεται και η υπογραφή της θέσης 5. Στο Σχήμα 12.7δ φαίνεται το αποτέλεσμα μετά την εισαγωγή της εγγραφής 5. Τέλος η εγγραφή 12 κατευθύνεται στη θέση 1 που είναι κατειλημμένη από την εγγραφή 52. Η εγγραφή αυτή παραμένει στη θέση της γιατί έχει μικρότερη υπογραφή, ενώ η εγγραφή 12 κατευθύνεται στην επόμενη θέση που είναι κατειλημμένη από την εγγραφή 68. Στο σημείο αυτό συγκρίνεται η πρώτη υπογραφή της εγγραφής 68 (δηλαδή 1000) με τη δεύτερη υπογραφή της εγγραφής 12 (δηλαδή 0110). Έτσι, η εγγραφή 12 καταλαμβάνει τη θέση 2, ενώ η εγγραφή 68 αφήνει την υπογραφή της στη θέση 2 και κατευθύνεται στη θέση 8. Η υπογραφή της θέσης αυτής είναι 0111, που είναι μικρότερη από την τιμή της δεύτερης υπογραφής (δηλαδή

1100) της εγγραφής 68. Έτσι, τελικά η εγγραφή 68 αποθηκεύεται στη θέση 3 μετά από δύο προσπάθειες. Ο αναγνώστης μπορεί πλέον να συμπεράνει πως προκύπτει η τελική μορφή της δομής στο Σχήμα 12.7ε. Η αναζήτηση ακολουθεί μία ανάλογη λογική και γι' αυτό το λόγο δεν δίνεται περισσότερη έμφαση στη διαδικασία αυτή.

(α)

1000	1011	1111	1000	Διαχωριστές
Κλειδί Υπογραφή	Κλειδί Υπογραφή	Κλειδί Υπογραφή	Κλειδί Υπογραφή	
cd 0100	ef 0100	kl 0101	op 0010	Σελίδες
	gh 1000	mn 1001		
	ij 1000			
10	46	95	116	Διεύθυνση

(β) εισαγωγή ab

1000	1000	1111	1000	Διαχωριστές
Κλειδί Υπογραφή	Κλειδί Υπογραφή	Κλειδί Υπογραφή	Κλειδί Υπογραφή	
cd 0100	ef 0100	kl 0101	op 0010	Σελίδες
	ab 0101	mn 1001	ij 0101	
		gh 1011		
10	46	95	116	Διεύθυνση

Σχήμα 12.8: Εισαγωγές σε αρχείο με μεγάλη χωρητικότητα.

Αν η χωρητικότητα των σελίδων είναι μεγαλύτερη της μίας εγγραφής, τότε η διαδικασία δεν διαφέρει σημαντικά. Έστω ένα αρχείο με σελίδες χωρητικότητας 3 εγγραφών. Στο Σχήμα 12.8α παρουσιάζονται μερικές σελίδες του αρχείου, όπου πρέπει να εισαχθεί η εγγραφή *ab*. Δίνεται ότι η ακολουθία των θέσεων όπου το κλειδί πρέπει να εισαχθεί και οι αντίστοιχες υπογραφές είναι:

$$f(ab) = (10, 46, \dots) \quad \text{και} \quad s(ab) = (1011, 0101, \dots)$$

Έτσι, λοιπόν η εγγραφή *ab* με υπογραφή 1011 δεν μπορεί να αποθηκευθεί στη σελίδα 10 που χαρακτηρίζεται από μικρότερο διαχωριστή (1000). Επομένως η διαδικασία συνεχίζεται στη σελίδα 46, που έχει τον ίδιο διαχωριστή, αλλά είναι πλήρης. Οι εγγραφές *gh* και *ij* εξάγονται από τη σελίδα αυτή,

ώστε να εισαχθεί το κλειδί ab . Ταυτόχρονα ενημερώνεται ο διαχωριστής της σελίδας σε 1000, όπως φαίνεται στο Σχήμα 12.8β. Τώρα οι εγγραφές gh και ij πρέπει να επανα-εισαχθούν. Δίνεται ότι

$$f(gh) = (46, 95, \dots) \quad \text{και} \quad s(gh) = (1000, 1011, \dots)$$

$$f(ij) = (\dots, 46, 116, \dots) \quad \text{και} \quad s(gh) = (\dots, 1000, 0101, \dots)$$

Είναι ευνόητο πλέον πως προκύπτει το Σχήμα 12.8β. Έτσι, για μία εισαγωγή έπρεπε να προσπελασθούν τρεις σελίδες και να αλλάξει ένας διαχωριστής.

Ανακεφαλαιώνοντας μερικά πρακτικά συμπεράσματα από υλοποιήσεις της μεθόδου αυτής παρατηρείται ότι:

- η επιτυχής και η ανεπιτυχής αναζήτηση εκτελούνται εγγυημένα με μία προσπέλαση στο δίσκο,
- η διαγραφή γίνεται πολύ εύκολα με τη μέθοδο της σημαίας, που δηλώνει τη λογική διαγραφή της αντίστοιχης εγγραφής,
- η φυσική διαγραφή είναι χρονοβόρα διαδικασία, γιατί ίσως θα έπρεπε να ενημερωθεί ο πίνακας υπογραφών και να μετακινηθούν μερικές εγγραφές του κύριου αρχείου,
- η εισαγωγή είναι ακριβή αν ο παράγοντας φόρτισης πλησιάζει τη μονάδα, γιατί μπορεί να προκαλέσει διαδοχικές επαναεισαγωγές ήδη αποθηκευμένων εγγραφών,
- ο παράγοντας φόρτισης ωστόσο μπορεί να είναι υψηλός (για παράδειγμα, να ισούται με 80%), το μέγεθος της σελίδας είναι 10 εγγραφές και το μήκος του διαχωριστή είναι 8 bits, τότε μία εισαγωγή απαιτεί προσπέλαση 1,5 σελίδων,
- υπάρχει απειροστή πιθανότητα κατά την εισαγωγή η διαδικασία των διαδοχικών δοκιμών να βρεθεί εκτός ελέγχου ή και να είναι εντελώς αδύνατη, οπότε δημιουργείται υπερχειλίση με ταυτόχρονη αύξηση του κόστους προσπέλασης,
- ο απαιτούμενος χώρος στην κύρια μνήμη είναι ιδιαίτερα μικρός, όσο μεγαλύτερη είναι η χωρητικότητα των σελίδων, για παράδειγμα όπως προκύπτει και αναλυτικά (Gonnet 1988) λιγότερο από ένα byte ανά εγγραφή του κύριου αρχείου, και τέλος

- υπάρχει το ανάλογο τίμημα της πολυπλοκότητας του λογισμικού.

Η μέθοδος που παρουσιάσθηκε είναι στατική, δηλαδή το μέγεθος του αρχείου δεν μεταβάλλεται ανάλογα με το πλήθος των εισαγωγών και των διαγραφών. Όπως είναι γνωστό η μέθοδος του γραμμικού κατακερματισμού είναι δυναμική, διακρίνεται όμως από υπερχειλίση και δεν εγγυάται ότι η επιτυχής ή η ανεπιτυχής αναζήτηση μπορεί να εκτελεσθεί με μία προσπάθεια. Από τον Larson (1988) έχει προταθεί μία δυναμική εκδοχή της μεθόδου χρήσης υπογραφών (διαχωριστών), η οποία βασίζεται στο γραμμικό κατακερματισμό. Η μελέτη από τη βιβλιογραφία της σύνθετης αυτής δομής αφήνεται στον αναγνώστη.

Στο σημείο αυτό αναφέρεται επιγραμματικά ότι από τον Larson (1985) προτάθηκε μία άλλη ενδιαφέρουσα δομή που ονομάζεται **εξωτερικός τέλειος κατακερματισμός** (external perfect hashing), γιατί δεν διακρίνεται από συγχρούσεις και υπερχειλίσεις. Η δομή αυτή είναι εξαιρετικά σύνθετη, αφού συνδυάζει τον επανακατακερματισμό, τον τέλειο κατακερματισμό και τα B-δένδρα. Στον αναγνώστη επαφίεται η μελέτη από τη βιβλιογραφία των αλγορίθμων διαχείρισης και αυτής της σύνθετης αυτής δομής. Ωστόσο, φαίνεται ότι παρά το γεγονός ότι τα τελευταία χρόνια έχουν προταθεί τόσο πανίσχυρες και αποτελεσματικές δομές, εντούτοις στα εμπορικά συστήματα διαχείρισης βάσεων δεδομένων το 'πανταχού παρόν' B-δένδρο (και η παραλλαγή του B⁺-δένδρου) είναι η πιο δημοφιλής και αναντικατάστατη δομή.

12.6 Φίλτρο Bloome

Σε πολλές εφαρμογές είναι δυνατόν η μεγάλη πλειοψηφία των ερωτήσεων να αφορά σε εγγραφές που είναι ανύπαρκτες. Όπως είναι γνωστό, συνήθως η ανεπιτυχής αναζήτηση κοστίζει ακριβότερα από την επιτυχή αναζήτηση επειδή η διαδικασία διαπίστωσης μη ύπαρξης ενός κλειδιού είναι σχετικά πιο χρονοβόρα. Το φίλτρο Bloome (1970) είναι μία μέθοδος που χρησιμεύει στην ταχύτερη διαπίστωση μίας ανεπιτυχούς αναζήτησης. Σύμφωνα με τη μέθοδο αυτή για κάθε κλειδί εγγραφής που είναι αποθηκευμένη στη δευτερεύουσα μνήμη προκύπτει μία συμπληρωματική δυαδική δομή εφαρμόζοντας μερικές συναρτήσεις κατακερματισμού. Κάθε συνάρτηση κατακερματισμού εξυπηρετεί στο να επιλέξει τυχαία κάποιο bit του φίλτρου και να το μετατρέψει από μηδέν σε άσσο. Η δομή αυτή είναι αρκετά μικρή ώστε να χωρά στην κύρια μνήμη. Επομένως η επεξεργασία της είναι πολύ γρήγορη και

θεωρείται ότι δεν συνεισφέρει στο συνολικό κόστος. Οι πιθανές αποχρίσεις σε μία ερώτηση είναι δύο: 'το κλειδί δεν υπάρχει' ή 'το κλειδί ίσως υπάρχει'. Στην πρώτη περίπτωση δεν απαιτείται προσπέλαση στο κύριο αρχείο, ενώ στη δεύτερη η προσπέλαση εκτελείται αν και μπορεί να είναι περιττή (περίπτωση λανθασμένης πτώσης). Έτσι, η δομή αυτή μπορεί να θεωρηθεί ως μία υπογραφή όλου του κύριου αρχείου.

Ακολουθεί ένα μικρό παράδειγμα για την καλύτερη κατανόηση της μεθόδου. Δίνεται στατικό αρχείο κατακερματισμού 5 θέσεων με χωρητικότητα μίας εγγραφής, όπου τα δεδομένα εισάγονται με εφαρμογή της συνάρτησης: $f(key) = key \bmod 5$, ενώ σε περίπτωση σύγκρουσης εφαρμόζεται η τεχνική της γραμμικής εξέτασης. Συγχρόνως κατασκευάζεται ένα φίλτρο Bloome μεγέθους 16 ψηφίων. Για κάθε εισαγόμενη εγγραφή εφαρμόζονται άλλες δύο συναρτήσεις κατακερματισμού: η συνάρτηση $'key \bmod 16'$ και η συνάρτηση $'key \bmod 14 + 2'$. Στο Σχήμα 12.9 παρουσιάζεται το περιεχόμενο του κύριου αρχείου και του αντίστοιχου φίλτρου μετά την εισαγωγή των εγγραφών 52, 19, 71 και 56. Έστω τώρα, ότι αγνοείται ο μηχανισμός του φίλτρου και ότι αναζητούνται δύο ανύπαρκτες εγγραφές 12 και 68. Η ανεπιτυχής αναζήτηση θα τερματίσει μετά από τέσσερις και τρεις προσπελάσεις στο κύριο αρχείο, αντίστοιχα. Ο αναγνώστης μπορεί να διαπιστώσει ότι με τη χρήση του φίλτρου οι προσπελάσεις αυτές θα αποφεύγονταν. Σε περίπτωση αναζήτησης της εγγραφής 44, εύκολα προκύπτει ότι το φίλτρο θα οδηγούσε σε λανθασμένη πτώση.

0																	
1	71	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	52	0	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0
3	56																
4	19																

Σχήμα 12.9: Αρχείο και αντίστοιχο φίλτρο Bloome.

Μία τέτοια μέθοδος μπορεί να φανεί πολύ χρήσιμη σε πληθώρα εφαρμογών, όπου είναι πολύ πιθανό ότι η ερώτηση θα είναι ανεπιτυχής, όπως για παράδειγμα,

- κατά την αναζήτηση από έναν καταστηματάρχη σε μία λίστα κλεμμένων ή χαμένων πιστωτικών καρτών,

- κατά την αναζήτηση από κάποιον τελωνειακό υπάλληλο σε μία λίστα κλεμμένων διαβατηρίων,
- κατά την αναζήτηση από ένα πρόγραμμα επεξεργασίας κειμένου για τις λανθασμένες λέξεις (spelling check), ή
- κατά τη χρήση της επιλογής DISTINCT για την απάλειψη των διπλοεγγραφών, που μπορεί να προκύψουν από μία ερώτηση με SELECT της γλώσσας ερωτήσεων SQL.

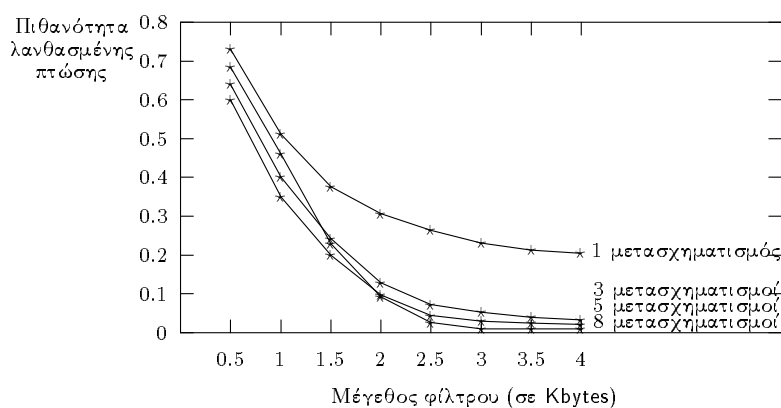
Ωστόσο, ιδιαίτερα αναφέρεται ότι ένα φίλτρο Bloome μπορεί κάλλιστα να εφαρμοσθεί και στη μέθοδο του κατακερματισμού με υπογραφές. Όπως φάνηκε με τη μέθοδο αυτή είναι δυνατό μία εισαγωγή να είναι αρκετά χρονοβόρα αν προκαλέσει τη διαδοχική διαγραφή και επανα-εισαγωγή άλλων εγγραφών. Στη χειρότερη περίπτωση μπορεί θεωρητικά η εκτέλεση μίας εισαγωγής να μην τερματισθεί ποτέ. Με σκοπό την πρόληψη του εκφυλισμού της επίδοσης κατά την εισαγωγή μπορεί να δημιουργηθεί ένα αρχείο υπερχειλίσης, όπου να αποθηκεύονται οι εγγραφές που δεν έγινε κατορθωτό να αποθηκευθούν στο κύριο αρχείο μετά από ένα συγκεκριμένο μικρό αριθμό προσπελάσεων. Είναι προφανές ότι το αρχείο αυτό με τις εγγραφές υπερχειλίσης θα είναι σχετικά μικρό, οπότε και το αντίστοιχο φίλτρο θα είναι επίσης σχετικά μικρό και επομένως αποτελεσματικό. Η επεξεργασία του φίλτρου αυτού στην κύρια μνήμη θα είναι λιγότερο χρονοβόρα από την επεξεργασία του αρχείου των υπογραφών και τις διαδοχικές συγκρίσεις με τους διάφορους διαχωριστές, που εκτελούνται επίσης στην κύρια μνήμη. Στη συνέχεια εξετάζεται αναλυτικότερα μία σπουδαία πρακτική εφαρμογή του φίλτρου Bloome.

Αν τα δεδομένα ενός αρχείου χρησιμοποιούνται κυρίως για αναγνώσεις, ενώ οι εισαγωγές και διαγραφές εγγραφών είναι λιγότερο συχνές, τότε μπορεί να χρησιμοποιηθεί μία μέθοδος που προτάθηκε από τον Severance (1976). Σύμφωνα με τη μέθοδο αυτή το κύριο αρχείο παραμένει στατικό, ενώ σε ένα δευτερεύον αρχείο, που ονομάζεται **αρχείο διαφορών** (differential file), αποθηκεύονται όλες οι ενημερώσεις των εγγραφών. Το αρχείο διαφορών μπορεί να είναι ένα σειριακό αρχείο με την προϋπόθεση ότι δεν θα μεγαλώσει υπέρμετρα. Έτσι, τα δεδομένα είναι αποθηκευμένα σε δύο αρχεία, που μπορεί να συγχωνευθούν σε περιόδους χαμηλού φόρτου επεξεργασίας. Ανάλογη διαδικασία συμβαίνει στο κύριο αρχείο και στο αρχείο συναλλαγών που εξετάστηκαν στο πρώτο κεφάλαιο, ωστόσο στην προκειμένη περίπτωση τα δεδομένα θα πρέπει να είναι διαθέσιμα κάθε στιγμή στους χρήστες.

Μία απλοϊκή μέθοδος αναζήτησης προτείνει την εξέταση κατ' αρχήν του αρχείου διαφορών. Αν η εγγραφή δεν υπάρχει στο αρχείο αυτό, τότε στη συνέχεια εξετάζεται το κύριο στατικό αρχείο. Μία περισσότερο εξελιγμένη μέθοδος αναζήτησης ανήκει στον Gremillion (1982), που συνιστά τη δημιουργία ενός φίλτρου Bloome για την ταχύτερη επεξεργασία του αρχείου διαφορών με σκοπό την πιθανή πρόβλεψη ανεπιτυχούς αναζήτησης. Έτσι, αρχικά η εξέταση του φίλτρου μπορεί να δώσει δύο πιθανές απαντήσεις:

- η αναζητούμενη εγγραφή δεν είναι αποθηκευμένη στο αρχείο διαφορών, οπότε αμέσως εξετάζεται το κύριο αρχείο, ή
- η αναζητούμενη εγγραφή μπορεί να είναι αποθηκευμένη στο αρχείο διαφορών, οπότε η εξέταση αρχίζει από αυτό. Αν η εγγραφή δεν βρεθεί στο αρχείο αυτό, τότε και πάλι η διαδικασία συνεχίζεται στο κύριο αρχείο.

Στο Σχήμα 12.10 παρουσιάζονται μερικά πειραματικά αποτελέσματα του Gremillion που δείχνουν μεγάλη αποτελεσματικότητα της μεθόδου, αν το μέγεθος του φίλτρου είναι της τάξης 3 ως 4 Kb και γίνεται χρήση περίπου 5 συναρτήσεων κατακερματισμού. Ενδιαφέρουσα είναι η παρατήρηση ότι για μικρά μεγέθη του φίλτρου η πιθανότητα λανθασμένης πτώσης αυξάνει καθώς αυξάνεται ο αριθμός των συναρτήσεων. Το φαινόμενο αυτό εξηγείται από το γεγονός ότι στις περιπτώσεις αυτές είναι πολλοί οι σχετικοί άσοι.



Σχήμα 12.10: Πιθανότητα λανθασμένης πτώσης σε αρχείο διαφορών.

Στη συνέχεια ακολουθεί μία απλή αναλυτική εξέταση της μεθόδου που αναφέρεται από τον Mullin (1983). Έστω ότι το φίλτρο Bloome έχει μέγεθος m δυαδικά ψηφία, όπου το m είναι της τάξης των μερικών δεκάδων χιλιάδων. Τα ψηφία αυτά αρχικά ισούνται όλα με μηδέν. Σε κάθε πρωτεύον κλειδί εισαγόμενης εγγραφής στο κύριο αρχείο εφαρμόζονται k συναρτήσεις κατακερματισμού (όπου το k μικρός ακέραιος), οπότε προκύπτουν οι θέσεις των ψηφίων του φίλτρου που θα πάρουν την τιμή του άσσου. Δηλαδή, ο μέγιστος αριθμός των ψηφίων του φίλτρου που θα γίνουν άσσοι για κάθε κλειδί θα είναι k . Επομένως κατά την αναζήτηση του κλειδιού μίας εγγραφής πρώτα εφαρμόζονται οι συναρτήσεις κατακερματισμού και ελέγχονται τα αντίστοιχα ψηφία με τη σειρά. Αν από κάποια συνάρτηση προκύψει η θέση ενός ψηφίου που είναι μηδέν στο φίλτρο, τότε η αναζήτηση τερματίζεται ως ανεπιτυχής. Στην αντίθετη περίπτωση μπορεί να συμβεί μία λανθασμένη πτώση με μία συγκεκριμένη πιθανότητα. Ποιά είναι όμως αυτή η πιθανότητα;

Έστω ότι στο κύριο αρχείο έχουν εισαχθεί n εγγραφές. Η πιθανότητα κάποιο ψηφίο να γίνει άσσος από μία συνάρτηση κατακερματισμού ισούται με $1/m$, ενώ η πιθανότητα να μη γίνει άσσος είναι $1-1/m$. Η πιθανότητα αυτό το ψηφίο να μη γίνει άσσος μετά από την εφαρμογή nk συναρτήσεων μετασχηματισμού είναι $(1-1/m)^{nk}$. Άρα η πιθανότητα το ψηφίο αυτό να γίνει άσσος μετά την εφαρμογή των συναρτήσεων αυτών είναι $1-(1-1/m)^{nk}$. Έστω τώρα, ότι αναζητάται μία εγγραφή. Από την εγγραφή αυτή με τους κατάλληλους μετασχηματισμούς προκύπτουν k άσσοι. Η πιθανότητα οι k αυτοί άσσοι να υπάρχουν στο φίλτρο είναι $\left(1-(1-1/m)^{nk}\right)^k$. Αν a είναι η πιθανότητα η αναζήτηση να είναι ανεπιτυχής, τότε η πιθανότητα να συμβεί μία λανθασμένη πτώση είναι $a \times \left(1-(1-1/m)^{nk}\right)^k$. Η ανάλυση αυτή οδηγεί σε τιμές που βρίσκονται κοντά στα αποτελέσματα του Gremillion και μπορούν να χρησιμοποιηθούν κατά το σχεδιασμό του αρχείου διαφορών και του αντίστοιχου φίλτρου.

Τα πλεονεκτήματα μίας τέτοιας οργάνωσης των δεδομένων με ένα αρχείο διαφορών είναι πολλαπλά, όπως για παράδειγμα:

- καλή επίδοση κατά την αναζήτηση, γιατί το κύριο αρχείο μπορεί να αποθηκευθεί σε ειδική ταχύτερη δευτερεύουσα συσκευή,
- απλοποίηση του λογισμικού επεξεργασίας του κύριου αρχείου,
- δυνατότητα ταυτόχρονης χρήσης του κύριου αρχείου από πολλούς χρήστες για λόγους που θα εξηγηθούν σε επόμενο κεφάλαιο, και τέλος

- μεγάλη αξιοπιστία των δεδομένων του κύριου αρχείου, αφού μάλιστα ο κάθε προγραμματιστής εφαρμογών μπορεί να έχει το δικό του αρχείο διαφορών.

Η ίδια φιλοσοφία διατρέχει και τη δομή του **B-δένδρου με περιοχή αλλαγών** (change area B-tree) που προτάθηκε από τον Mullin (1981). Σύμφωνα με αυτή τη δομή και το κύριο αρχείο αλλά και το αρχείο διαφορών είναι δύο απλά B-δένδρα.

12.7 Ασκήσεις

<1> Να κατασκευασθεί ένα δένδρο υπογραφών με εγγραφές που να αποτελούνται από τα πεδία Ειδικότητα, Διεύθυνση, Διευθυντής, και Μισθός. Το συνολικό μήκος της υπογραφής είναι 16 bits και επιμερίζεται σε 3, 4, 4 και 5 bits αντίστοιχα. Οι επόμενες συναρτήσεις κατακερματισμού 'σηκώνουν' ένα bit σε άσσο ως εξής:

$$\begin{aligned}
 H_1 &= \begin{cases} \text{το 1o bit αν η ονομασία της ειδικότητας αρχίζει από A - Θ} \\ 2o & \text{από I - Π} \\ 3o & \text{από P - Ω} \end{cases} \\
 H_2 &= \begin{cases} \text{το 1o bit αν η ονομασία της διεύθυνσης αρχίζει από A - Z} \\ 2o & \text{από H - M} \\ 3o & \text{από N - Σ} \\ 4o & \text{από T - Ω} \end{cases} \\
 H_3 &= \begin{cases} \text{το 1o bit αν η ονομασία του διευθυντή αρχίζει από A - Z} \\ 2o & \text{από H - M} \\ 3o & \text{από N - Σ} \\ 4o & \text{από T - Ω} \end{cases} \\
 H_4 &= \begin{cases} \text{το 1o bit αν ο μισθός είναι μέχρι 150.000 δραχμές} \\ 2o & \text{από 150.001 μέχρι 250.000 δραχμές} \\ 3o & \text{από 250.001 μέχρι 350.000 δραχμές} \\ 4o & \text{από 350.001 μέχρι 450.000 δραχμές} \\ 5o & \text{περισσότερο από 450.000 δραχμές} \end{cases}
 \end{aligned}$$

Κατά το σχηματισμό των ανωτέρων επιπέδων του δένδρου να θεωρηθεί ότι ο παράγοντας ομαδοποίησης ισούται με τρία. Να γίνουν παραδείγματα αναζήτησης.

<2> Να κατασκευασθεί παράδειγμα μίας δομής με υπογραφές σελίδων με βάση τα δεδομένα της προηγούμενης άσκησης. Για τις υπογραφές των εγγραφών να χρησιμοποιηθούν οι ίδιες συναρτήσεις κατακερματισμού για όλα τα πεδία. Η περιγραφή της σελίδας έχει μήκος οκτώ bits, δύο bits ανά χαρακτηριστικό, και χρησιμοποιούνται οι επόμενες συναρτήσεις που 'σηκώνουν' τα αντίστοιχα bits:

$$T_1 = \begin{cases} \text{το 1ο bit αν η ονομασία της ειδικότητας αρχίζει από A - M} \\ \text{2ο} & \text{από N - Ω} \end{cases}$$

$$T_2 = \begin{cases} \text{το 1ο bit αν η ονομασία της διεύθυνσης αρχίζει από A - M} \\ \text{2ο} & \text{από N - Ω} \end{cases}$$

$$T_3 = \begin{cases} \text{το 1ο bit αν η ονομασία του διευθυντή αρχίζει από A - M} \\ \text{2ο} & \text{από N - Ω} \end{cases}$$

$$T_4 = \begin{cases} \text{το 1ο bit αν ο μισθός είναι μέχρι και 350.000 δραχμές} \\ \text{2ο} & \text{περισσότερο από 150.001 δραχμές} \end{cases}$$

Να γίνουν παραδείγματα αναζήτησης. Να γίνει ποιοτική σύγκριση με την επίδοση της αναζήτησης της δομής της προηγούμενης άσκησης.

<3> Στο παράδειγμα του Σχήματος 12.7 να εισαχθεί η εγγραφή 38 ή η εγγραφή 49. Τι αλλαγές προκαλούνται στο κύριο αρχείο και στο αρχείο των υπογραφών αντίστοιχα;

<4> Τα κλειδιά του παραδείγματος του Σχήματος 12.7 να εισαχθούν σε αντίστροφη σειρά. Να σχεδιασθεί διαδοχικά η διαδικασία.

<5> Δίνεται αρχείο κατακερματισμού 5 θέσεων με χωρητικότητα μίας εγγραφής, όπου εφαρμόζεται η συνάρτηση $key \bmod 5$ και η μέθοδος της γραμμικής εξέτασης για την επίλυση των συγκρούσεων. Φίλτρο Bloome των 16 bits κατασκευάζεται με τη βοήθεια των συναρτήσεων ' $key \bmod 16$ ' και ' $key \bmod 14 + 2$ '. Να εισαχθούν οι εγγραφές 22, 51, 26 και 85 στο αρχείο και να ενημερωθεί το φίλτρο. Σε ποιές περιπτώσεις κατά τις αναζητήσεις των εγγραφών 81, 27, 46, 23, 51 και 39 συμβαίνει λανθασμένη πτώση; Πόσες προσπελάσεις στο αρχείο εξοικονομούνται λόγω της ύπαρξης του φίλτρου κατά τις αναζητήσεις των κλειδιών αυτών;

Κεφάλαιο 13

ΣΥΓΚΡΙΣΗ ΤΩΝ ΔΟΜΩΝ

13.1 Εισαγωγή

13.2 Κριτήρια επιλογής της κατάλληλης οργάνωσης

13.3 Επίλογος

13.4 Ασκήσεις

Κεφάλαιο 13

ΣΥΓΚΡΙΣΗ ΤΩΝ ΔΟΜΩΝ

13.1 Εισαγωγή

Στα προηγούμενα κεφάλαια παρουσιάστηκε μία σειρά μεθόδων προσπέλασης (access methods), δηλαδή οργανώσεων αρχείων και αλγορίθμων διαχείρισης, οι οποίες είναι προσανατολισμένες στην επεξεργασία με βάση το πρωτεύον ή/και το δευτερεύον κλειδί. Δηλαδή οι οργανώσεις είναι οι δομές, ενώ οι αλγόριθμοι διαχείρισης είναι οι τρόποι επεξεργασίας της δομής και εκτελούνται από ένα σύνολο ρουτινών. Το κύριο πρόβλημα, λοιπόν, στη φυσική σχεδίαση ενός συστήματος διαχείρισης βάσεων δεδομένων είναι η διάταξη των εγγραφών, ώστε η προσπέλασή τους από τα προγράμματα εφαρμογής να γίνεται κατά αποτελεσματικό τρόπο.

Η επεξεργασία με βάση το πρωτεύον κλειδί μπορεί να είναι μία από τις παρακάτω βασικές ερωτήσεις:

- απλή ερώτηση με βάση μία συγκεκριμένη τιμή,
- μαζική (batched) ερώτηση μερικών ταξινομημένων εγγραφών που θα εξετασθεί ειδικότερα στη συνέχεια,
- ερώτηση διαστήματος, ή τέλος
- εξαντλητική ερώτηση όλων των εγγραφών.

Στον Πίνακα 13.1 γίνεται μία ανακεφαλαίωση της επίδοσης των ερωτήσεων αυτών στα αρχεία πρωτεύοντος κλειδιού. Όπου Α σημαίνει ότι η επίδοση

Δομή αρχείου	Ερώτηση			
	Απλή	Μαζική	Διαστήματος	Εξαντλητική
Σωρός	E	Γ	E	B
Σειριακό ταξινομημένο	Δ	Γ	Δ	A
Σειριακό με δείκτη	B	Γ	B	B
Δενδρικό	A	B	A	Γ
Τυχαίο στατικό	B	E	E	Δ
Τυχαίο δυναμικό	A	Δ	Δ	Γ

Πίνακας 13.1: Επίδοση δομών σε ερώτηση με βάση πρωτεύον κλειδί.

είναι πολύ καλή, το B σημαίνει καλή επίδοση, το Γ δηλώνει ότι απαιτείται μικρή προσπάθεια, το Δ ότι απαιτείται πολλή προσπάθεια, ενώ το E ότι δεν είναι λογική η χρήση της οργάνωσης αυτής για τέτοιου είδους ερώτηση. Ο πίνακας αυτός είναι αρκετά εύγλωττος όσον αφορά στην επιλογή μίας δομής για την αποτελεσματική υλοποίηση μίας συγκεκριμένης εφαρμογής.

Δομή αρχείου	Ερώτηση			
	Σημειακή	Μαζική	Μερικής Ταύτισης	Διαστήματος
Αντεστραμμένα αρχεία	B	B	B	Γ
Πολλαπλές λίστες	B	Γ	Γ	Γ
Συνδυασμένοι κατάλογοι	B	Γ	A	Γ
Πολυδιάστατα δένδρα	B	B	B	B
Δικτυωτό αρχείο	A	B	A	A

Πίνακας 13.2: Επίδοση δομών σε ερώτηση με βάση δευτερεύον κλειδί.

Αν οι εγγραφές πρόκειται να ανακτηθούν με βάση περισσότερο από ένα κλειδιά, τότε οι οργανώσεις του Πίνακα 13.1 δεν είναι οι πλέον κατάλληλες, γιατί μεροληπτούν σε βάρος μερικών κλειδιών. Στον Πίνακα 13.2 ανακεφαλαιώνεται η επίδοση των δομών που εξετάστηκαν στο ενδέκατο κεφάλαιο σε σχέση με:

- τη σημειακή ερώτηση,
- τη μαζική ερώτηση,
- την ερώτηση μερικής ταύτισης, και τέλος

- την ερώτηση διαστήματος.

Υπενθυμίζεται ότι τα δικτυωτά αρχεία είναι μία από τις πιο σύγχρονες και αποτελεσματικές οργανώσεις όταν οι τιμές των δεδομένων κάθε πεδίου υπακούουν σε ομοιόμορφη στατιστική κατανομή, οπότε με τη δομή αυτή απαιτούνται δύο μόνο προσπελάσεις στο δίσκο για τη σημειακή ερώτηση. Αν οι κατανομές των τιμών των πεδίων δεν είναι ομοιόμορφες, τότε η δομή αυτή είναι προβληματική από άποψη χώρου. Τα πολυδιάστατα δένδρα και τα πολυδιάστατα B-δένδρα είναι επίσης αξιόλογες δομές για την ικανοποίηση σημειακών ερωτήσεων και ερωτήσεων διαστήματος. Συχνότερα όμως στην πράξη συναντώνται οι παραδοσιακές οργανώσεις των αντεστραμμένων αρχείων. Στο επόμενο κεφάλαιο θα αναπτυχθούν και άλλες δομές που έχουν προταθεί στην πρόσφατη βιβλιογραφία και μπορούν να ανταποκριθούν στις απαιτήσεις της αναζήτησης με βάση το πρωτεύον και το δευτερεύον κλειδί σε εξειδικευμένες εφαρμογές (για παράδειγμα τα R-δένδρα, δες Κεφάλαιο 14.2). Τέλος, αξ σημειωθεί ότι οι δύο προηγούμενοι πίνακες δεν δίνουν την επίδοση των δομών κατά την εισαγωγή και τη διαγραφή, λειτουργίες που οπωσδήποτε χρειάζονται ιδιαίτερη προσοχή.

13.2 Κριτήρια επιλογής της κατάλληλης οργάνωσης

Η επιλογή της κατάλληλης οργάνωσης γίνεται με βάση ορισμένα κριτήρια. Καμία δομή δεν μπορεί να χαρακτηριστεί γενικά ως η καλύτερη για κάθε λειτουργία. Πολύ περισσότερο η επιλογή μίας δομής για ένα πρόβλημα δεν είναι εύκολη υπόθεση, γιατί μπορεί δύο ή περισσότερες οργανώσεις να λειτουργούν περίπου το ίδιο ικανοποιητικά. Μία όσο το δυνατόν πιο εμπειριστατωμένη απάντηση πρέπει να εξετάσει τους εξής πέντε παράγοντες:

- τύπος, όγκος και χαρακτηριστικά των δεδομένων,
- πρότυπα χρήσης αρχείου (κυρίως ερώτηση, εισαγωγή ή διαγραφή),
- δομή, οργάνωση και χαρακτηριστικά (ταχύτητα, χωρητικότητα κλπ.) του υλικού,
- απαιτήσεις, όπως χρόνος προσπέλασης, ποσοστό χρήσης των υπολογιστικών πόρων κλπ., και τέλος,
- άλλα κριτήρια, όπως πχ. διαθεσιμότητα, ανακτησιμότητα, τρωσιμότητα, αξιοπιστία, εξελισιμότητα, ασφάλεια, ακεραιότητα κλπ.

Στη συνέχεια σχολιάζονται μερικές ποιοτικές παράμετροι που επηρεάζουν την επιλογή του κατάλληλου αρχείου. Ποσοτικά συμπεράσματα μπορούν να εξαχθούν από τα στοιχεία αυτά είτε

- με αναλυτικές μεθόδους, οπότε για να υλοποιηθεί ένα μοντέλο θα πρέπει να γίνουν αρκετές απλοποιήσεις, είτε
- με προσομοίωση, που είναι μία σύνθετη και πιο χρονοβόρα διαδικασία.

Το μέγεθος του αρχείου (file size) μετράται σε χαρακτήρες και ισούται με το γινόμενο του πλήθους των εγγραφών επί το μέγεθος μίας εγγραφής σε bytes. Το μέγεθος μίας εγγραφής ισούται με το άθροισμα των μηκών των πεδίων της εγγραφής σε bytes. Αν είναι γνωστό το τρέχον μέγεθος του αρχείου και γίνει σωστή πρόβλεψη για την εξέλιξη του μεγέθους του αρχείου στο μέλλον, τότε μπορεί να επιλεγθεί επαρκές υλικό δευτερεύουσας μνήμης. Αν η εγγραφή είναι μεταβλητού μήκους, τότε η κατάσταση είναι συνθετότερη και πρέπει να γίνει λεπτομερέστερη ανάλυση για την εύρεση του τύπου των εγγραφών.

Δομή αρχείου	Αιτία επιπρόσθετου απαιτούμενου χώρου
Σειριακά με δείκτη	Χώρος για δείκτες (συνήθως τρία επίπεδα ή και περισσότερα) και περιοχή υπερχείλισης.
B-δένδρα	Δείκτες προς τους κόμβους. Μέση χρησιμοποίηση του χώρου περίπου 69%.
Δικτυωτά αρχεία	Γραμμικές κλίμακες, δικτυωτός κατάλογος. Στους κάδους μέση χρησιμοποίηση του χώρου 69%.
Στατικά Τυχαία	Μη χρησιμοποιημένος χώρος στους κάδους απαιτεί δείκτες για τις νέες εγγραφές υπερχείλισης.
Δυναμικά Τυχαία	Δενδρικοί κατάλογοι. Μέση χρησιμοποίηση του χώρου περίπου 69%.

Πίνακας 13.3: Επιπλέον απαιτούμενος χώρος.

Στον Πίνακα 13.3 δίνεται ένας κατάλογος που δίνει για τις κυριότερες οργανώσεις αρχείων το χώρο που δεσμεύεται πλέον του χώρου που καταλαμβάνεται μόνο από τις εγγραφές. Παρατηρείται γενικά ότι οι δομές που

είναι ισοζυγισμένες και δυναμικές θεωρητικά επιτυγχάνουν μέση χρησιμοποίηση του χώρου περίπου 69% (για παράδειγμα, η οικογένεια B-δένδρων, το δικτυωτό αρχείο, τα δυναμικά τυχαία αρχεία κλπ). Αλλά και στις στατικές δομές υπάρχει ελεύθερος χώρος (τόσο στο κύριο αρχείο, όσο και στην περιοχή υπερχειλίσσης, όπως στη δομή ISAM) με μέγεθος που πρέπει να εκτιμηθεί από το σχεδιαστή της φυσικής οργάνωσης.

Πρέπει ωστόσο να σημειωθεί παρεπιπτόντως ότι στην πράξη οι κατασκευαστές λογισμικού δεν υλοποιούν τις δομές όπως ακριβώς περιγράφονται θεωρητικά. Για παράδειγμα, στο B⁺-δένδρο, που είναι η βασική δομή στο φυσικό επίπεδο κάθε συστήματος διαχείρισης βάσεων δεδομένων, δεν γίνονται συγχωνεύσεις κόμβων όταν η περιεκτικότητά τους είναι λιγότερο από 50%. Έτσι, προφανώς η μέση τιμή του παράγοντα χρησιμοποίησης χώρου γίνεται λιγότερο από 69%. Ο πρώτος λόγος για την επιλογή αυτή είναι ότι συνήθως οι διαγραφές εκτελούνται σπανιότερα από τις εισαγωγές και επομένως το κόστος σε χώρο δεν είναι σημαντικό. Δεύτερον, με την εξέλιξη των μέσων αποθήκευσης, το κόστος του απαιτούμενου χώρου καθίσταται συνεχώς μικρότερο. Ο τρίτος λόγος για την επιλογή αυτή είναι, ότι έτσι η δομή μπορεί να χρησιμοποιηθεί ταυτόχρονα από περισσότερους χρήστες, καθώς εκτελούνται λιγότερα κλειδώματα κόμβων (δες Κεφάλαιο 15.4). Από τον Shasha (1989) έχει εξεταστεί ένα B-δένδρο με n εγγραφές, όπου γίνονται $m \gg n$ ενημερώσεις. Αν, λοιπόν, κάθε ενημέρωση αντιμετωπισθεί ως μία διαγραφή και μία εισαγωγή, τότε αποδεικνύεται αναλυτικά ότι αν οι κόμβοι συγχωνεύονται μόνο όταν είναι τελείως κενοί, τότε η μέση χρησιμοποίηση του χώρου είναι μόλις 39%.

Κατά τη φάση του φυσικού σχεδιασμού, λοιπόν, ο αναλυτής πρέπει να εκτιμήσει το διατιθέμενο χώρο για την κύρια και τη δευτερεύουσα μνήμη. Για παράδειγμα, αν ο συνολικός καταλαμβανόμενος χώρος αποτελεί έναν περιοριστικό παράγοντα, τότε ίσως απαιτείται ιδιαίτερη προσοχή (για παράδειγμα, παθολογική περίπτωση επεκτατού κατακερματισμού). Επίσης, μερικές οργανώσεις έχουν καλή απόδοση μόνο αν ο κατάλογος τους βρίσκεται στην κύρια μνήμη. Επομένως, ακόμη και αν ο δίσκος είναι μεγάλης χωρητικότητας, η μη ύπαρξη της απαιτούμενης κύριας μνήμης καθιστά τις δομές του δικτυωτού αρχείου και των τυχαίων δυναμικών οργανώσεων (εκτός του γραμμικού κατακερματισμού) απρόσφορες.

Σε αυτό το σημείο υπενθυμίζεται ότι για την οικονομικότερη αποθήκευση των δεδομένων και την ταχύτερη απόκριση χρησιμοποιείται πολύ συχνά στη πράξη η τεχνική της συμπίεσης (compression). Στη βιβλιογραφία συνα-

ντάται πληθώρα τεχνικών συμπίεσης που είτε εξαρτώνται από τη συγκεκριμένη εφαρμογή είτε όχι. Στο Κεφάλαιο 15.2 γίνεται μία επιλεγμένη παρουσίαση μερικών μεθόδων συμπίεσης. Ακόμη κατά το λογικό σχεδιασμό μίας βάσης δεδομένων πρέπει να προσεχθεί ποιά είναι τα κοινά χαρακτηριστικά μεταξύ των διαφόρων αρχείων (δηλαδή, πινάκων κατά το σχεσιακό μοντέλο). Με σκοπό τον περιορισμό των πλεοναζόντων δεδομένων (redundant data) επιδιώκεται αυτά τα κοινά χαρακτηριστικά να αποθηκευθούν μόνο μία φορά. Το αντικείμενο αυτό θα εξετασθεί λεπτομερώς στα πλαίσια του μαθήματος των Βάσεων Δεδομένων.

Με τον όρο **χρόνος ανάπτυξης** (development time) εννοούνται οι εξής παράγοντες:

- υπονοείται ο απαιτούμενος χρόνος για την ανάπτυξη και συντήρησης του λογισμικού. Μία πιο σύνθετη δομή αρχείου απαιτεί, προφανώς, περισσότερο χώρο για να δημιουργηθεί.
- υπάρχουν και άλλοι παράγοντες που πρέπει να συνυπολογισθούν στο χρόνο ανάπτυξης του αρχείου, όπως ο χρόνος ανάπτυξης του λογισμικού για τα βοηθητικά αρχεία, όπως το αρχείο συναλλαγών κλπ.
- η **δοκιμασία** (testing) του λογισμικού του αρχείου είναι ένας τρίτος παράγοντας.

Εναλλακτικά, αντί να γίνει πλήρης ανάπτυξη του λογισμικού, μπορεί να χρησιμοποιηθούν έτοιμες βιβλιοθήκες για μεθόδους προσπέλασης, που διατίθενται είτε εμπορικά είτε είναι διαθέσιμες μέσω του διαδικτύου. Σε μία τέτοια περίπτωση, απαιτείται προσπάθεια για την ενοποίηση και την ομογενοποίηση των συνιστωσών.

Ο απαιτούμενος **χρόνος απόκρισης** (responce time) είναι ένας άλλος σημαντικός παράγοντας που λαμβάνεται σοβαρά υπ' όψη κατά το φυσικό σχεδιασμό του συστήματος μίας εφαρμογής. Για παράδειγμα, αν απαιτείται απόκριση σε χρόνο της τάξης του δευτερολέπτου στη χειρότερη περίπτωση και όχι της τάξης λεπτών ή περισσότερο, τότε το τυχαίο αρχείο είναι η μοναδική επιλογή. Αν οι απαιτήσεις των χρηστών για προσπέλαση σε κάποιο αρχείο είναι πολύ συχνές, τότε είναι επιθυμητό ο χρόνος απόκρισης να είναι μικρός ιδιαίτερα σε αυτή τη δομή. Ωστόσο, ο χρόνος απόκρισης εξαρτάται από πολλούς σύνθετους παράγοντες. Για παράδειγμα, αποτελεί αντικείμενο της περιοχής της Αξιολόγησης Επίδοσης (Perfromance Evaluation) η μελέτη ιδιαιτέρων παραγόντων, όπως ο ρυθμός άφιξης των απαιτήσεων προς το

σύστημα, τα ίδια χαρακτηριστικά του συστήματος (όπως αλγόριθμοι χρονοδρομολόγησης, χρόνος εντοπισμού, ταχύτητα επεξεργασίας, διαθέσιμη απομονωτική μνήμη, αλγόριθμος αντικατάστασης σελίδων) κλπ. Σε κρίσιμες εφαρμογές όπως σε συστήματα πραγματικού χρόνου (όπως σε συστήματα αεροπορικών εταιρειών για κράτησεις θέσεων, σε τραπεζικά ή χρηματιστηριακά συστήματα κλπ.), πρέπει να χρησιμοποιηθούν τεχνικές παράλληλης αναζήτησης, ενώ επίσης πρέπει η δομή του αρχείου να λαμβάνει υπ' όψη την ελαχιστοποίηση του χρόνου εντοπισμού.

Σε ένα σύστημα πολλών χρηστών είναι οι δυνατόν οι ερωτήσεις να ομαδοποιηθούν και να απαντηθούν ως σύνολο. Δηλαδή, αγνοώντας τη σειρά άφιξης των ερωτήσεων και διατάσσοντας τις κατά κάποιο διαφορετικό τρόπο μπορεί και:

- οι πόροι του συστήματος να χρησιμοποιηθούν πιο αποδοτικά (εργασίες στην κύρια μνήμη και στο δίσκο), αλλά και
- η μέση χρονική απόκριση προς τους χρήστες να ελαττωθεί.

Αυτή η μέθοδος, που είναι γνωστή ως **μαζικοποίηση** (batching), αναφέρθηκε για πρώτη φορά από τον Shneiderman (1976) και μπορεί να εφαρμοσθεί τόσο κατά την επεξεργασία δομών της δευτερεύουσας ή της κύριας μνήμης όσο και σε χρονοδρομολόγηση δίσκων. Για παράδειγμα, αντί οι αναζητήσεις σε ένα σειριακό ή σε ένα δενδρικό αρχείο να εκτελεσθούν ανεξάρτητα η μία από την άλλη ανάλογα με τη σειρά άφιξης, είναι δυνατόν τα κλειδιά των αναζητούμενων εγγραφών να συγκεντρωθούν και να διαταχθούν κατά την τάξη του αρχείου ώστε να απαντηθούν ομαδικά. Έτσι η εξοικονόμηση προσπελάσεων στο δίσκο μπορεί να είναι σημαντική, και ιδιαίτερα με τη βοήθεια μεγάλων απομονωτικών μνημών. Όπως γίνεται όμως αντιληπτό η τεχνική αυτή κάνει διάκριση σε βάρος μερικών ερωτήσεων που καθυστερεί η εξυπηρέτησή τους, ώστε να εξυπηρετηθεί γρηγορότερα το σύνολο των ερωτήσεων. Έτσι δεν ενδείκνυται για χρήση σε συστήματα πραγματικού χρόνου.

Η **δραστηριότητα** (activity) ενός αρχείου είναι ένα μέτρο που δείχνει το ποσοστό των εγγραφών του κύριου αρχείου που ενημερώνεται σε κάθε τρέξιμο ενός προγράμματος εφαρμογής. Ένα αρχείο με πολύ μεγάλο δείκτη δραστηριότητας κατά προτίμηση οργανώνεται κατά σειριακό τρόπο, επειδή σχεδόν κάθε εγγραφή θα πρέπει να προσπελασθεί και να ενημερωθεί κατά τη διάρκεια του τρεξίματος της εφαρμογής. Ένα αρχείο με χαμηλό δείκτη δραστηριότητας θα είναι περισσότερο αποτελεσματικό αν οργανωθεί κατά

τυχαίο τρόπο έτσι ώστε να προσπελάζονται κάθε φορά μόνο οι εγγραφές που θα ενημερωθούν. Όταν μερικά πεδία μίας εγγραφής δεν ενημερώνονται τόσο συχνά όσο άλλα, τότε επιτυγχάνεται καλύτερη επίδοση αν το αρχείο χωριστεί σε δύο μικρότερα, το ένα αρχείο σταθερό και το άλλο με μεγαλύτερη δραστηριότητα. Το πρόβλημα αυτό είναι και αντικείμενο του λογικού σχεδιασμού των βάσεων δεδομένων και θα μελετηθεί στα πλαίσια του αντίστοιχου μαθήματος.

Η μεταβλητότητα (volatility) ενός αρχείου είναι το μέτρο που δείχνει το ποσοστό των εγγραφών που εισάγονται ή διαγράφονται από το κύριο αρχείο. Ένα πολύ μεταβαλλόμενο σειριακό κύριο αρχείο ανανεώνεται με μία διαδικασία συγχώνευσης των νέων εγγραφών με τις εγγραφές του αρχείου συναλλαγών για να προκύψει το νέο κύριο αρχείο. Η επίδοση των σειριακών αρχείων φθίνει όσο αυξάνει η μεταβλητότητα του αρχείου επειδή είναι απαραίτητη μία εκτεταμένη αναδιοργάνωση ακόμη και για περιορισμένο αριθμό εισαγωγών ή/και διαγραφών. Γενικά, οι στατικές δομές δεν προσφέρονται σε εφαρμογές που διακρίνονται από υψηλή μεταβλητότητα.

Οι δύο αυτές έννοιες της δραστηριότητας και της μεταβλητότητας δίνουν το βαθμό προσπέλασης του αρχείου για αναζήτηση ή για ενημέρωση, αντίστοιχα, σε μία συγκεκριμένη χρονική περίοδο. Ποσοτικά μάλιστα εκφράζονται με το λόγο των εγγραφών που αναζητώνται ή ενημερώνονται, αντίστοιχα, προς το σύνολο των εγγραφών. Όσο ο πρώτος λόγος τείνει προς τη μονάδα (το μηδέν), τόσο πιο ισχυρό είναι το επιχείρημα υπέρ της σειριακής (αντίστοιχα, της τυχαίας) οργάνωσης. Η επιλογή είναι δύσκολη όταν αυτός ο λόγος είναι, για παράδειγμα, 30% ή 50%. Μία τέτοια επιλογή θα μπορούσε να υποστηριχθεί μόνο με βάση τα αποτελέσματα μίας προσομοίωσης. Ακόμη πρέπει να διασαφηνισθεί αν ο παράγοντας μεταβλητότητας οφείλεται σε απλές ενημερώσεις ή σε εισαγωγές και διαγραφές. Μία δομή αρχείου ISAM δεν δημιουργεί προβλήματα στην πρώτη περίπτωση, δημιουργεί όμως στη δεύτερη. Αντίθετα η οικογένεια των B-δένδρων συμπεριφέρεται πολύ ομαλά στις εισαγωγές και διαγραφές.

Στις λεγόμενες Στατιστικές βάσεις δεδομένων κάθε εγγραφή έχει πάρα πολλά πεδία και συνεπώς έχει μεγάλο μήκος, ενώ ταυτόχρονα ο αριθμός των εγγραφών είναι επίσης τεράστιος. Ένα κλασικό παράδειγμα της κατηγορίας αυτής είναι τα αρχεία που προκύπτουν κατά τις απογραφές πληθυσμού. Σε εφαρμογές αυτού του είδους απαιτείται η εύρεση του μέγιστου, του ελάχιστου, του μέσου όρου ή της απόκλισης των τιμών ενός συγκεκριμένου πεδίου. Βέβαια σε μία τέτοια περίπτωση η δραστηριότητα ισούται με τη

μονάδα, όμως η οργάνωση των δεδομένων στο περιβάλλον αυτό αντιμετωπίζεται διαφορετικά. Επειδή απαιτείται η σάρωση όλης της δομής του αρχείου εκτελώντας μία εξαντλητική αναζήτηση με βάση ένα μόνο συγκεκριμένο πεδίο του αρχείου για κάθε ερώτηση κάθε φορά, χρησιμοποιείται μία διαφορετική οργάνωση: το **αρχείο αντιμετάθεσης** (transposed file), που προτάθηκε από τον Easton (1969). Η δομή αυτή αποτελείται από τόσες εγγραφές όσα είναι τα πεδία του αρχικού αρχείου, ενώ κάθε εγγραφή του νέου αρχείου αποτελείται από τις τιμές ενός πεδίου από όλες τις εγγραφές του αρχικού αρχείου. Η τάξη των πεδίων στις εγγραφές του αρχείου αντιμετάθεσης είναι η ίδια με την τάξη των εγγραφών στο αρχικό αρχείο και δεν περιέχει καθόλου δείκτες προς κάποιο άλλο αρχείο (είναι κύριο αρχείο αυτό το ίδιο). Είναι προφανές ότι το μέγεθος του αρχείου αντιμετάθεσης είναι ίσο με το μέγεθος του αρχικής δομής. Ωστόσο στις διαδοχικές τιμές μίας εγγραφής του αρχείου αντιμετάθεσης μπορεί να εφαρμοσθεί μία τεχνική συμπίεσης με σκοπό τη μεγαλύτερη οικονομία σε χώρο και χρόνο. Εν πάσει περιπτώσει, για τη στατιστική επεξεργασία ενός πεδίου του αρχικού αρχείου δεν προσπελάζονται από το δίσκο άσχετα δεδομένα αλλά μόνο η σχετική εγγραφή του αρχείου αντιμετάθεσης. Βέβαια, πρέπει να σημειωθεί ότι το κόστος για τη δημιουργία και την ενημέρωση του αρχείου αντιμετάθεσης λόγω των εισαγωγών και διαγραφών είναι υψηλό.

Στενά συνδεδεμένη με τις βασικές έννοιες της δραστηριότητας και της μεταβλητότητας είναι η έννοια της **συχνότητας χρήσης** (frequency of use) του αρχείου. Ο παράγοντας αυτός είναι επίσης πολύ σπουδαίος για τη σωστή φυσική σχεδίαση. Κατά κανόνα όσο πιο συχνά χρησιμοποιείται μία δομή, τόσο πιο ισχυροί είναι οι λόγοι που επιβάλλουν την υλοποίηση του με μία τυχαία οργάνωση.

Έχει διαπιστωθεί επίσης ότι η **συχνότητα αναφοράς των εγγραφών** μίας δομής δεν είναι σταθερή αλλά μεταβάλλεται. Δηλαδή, πρακτικά έχει διαπιστωθεί ότι το 80% (το 20%) των εγγραφών αναζητώνται με πιθανότητα 20% (αντίστοιχα, με πιθανότητα 80%). Αυτή η παρατήρηση λέγεται κανόνας του Heising και συνετέλεσε στον ορισμό της λεγόμενης στατιστικής κατανομής 80-20 (1963). Συνήθως μάλιστα οι πιο συχνά αναζητούμενες εγγραφές είναι εκείνες που έχουν εισαχθεί στη δομή τελευταίες. Έτσι αν κάποια ανταλλαχτικά μίας αποθήκης αναζητώνται πιο συχνά τότε θα έπρεπε να τοποθετηθούν στη δομή, έτσι ώστε να ελαχιστοποιείται ο χρόνος απόκρισης (για παράδειγμα, στην αρχή μίας αλυσίδας και όχι στο τέλος της). Οι δευτερεύοντες κατάλογοι ενός αρχείου θα έπρεπε να αντιμετωπισθούν κατά τον ίδιο τρόπο, θα έπρεπε να τοποθετηθούν στον κεντρικό κύλινδρο

του δίσκου, ώστε να ελαχιστοποιηθεί ο χρόνος εντοπισμού. Για λόγους ευκολίας πολύ συχνά χρησιμοποιείται σε μελέτες με αναλυτικό υπόβαθρο η υπόθεση εργασίας ότι όλες οι εγγραφές αναζητώνται ισοπίθانا. Ωστόσο, έχει αποδειχθεί ωστόσο από το Χριστοδουλάκη ότι η παραδοχή οδηγεί σε πεσσιμιστικές εκτιμήσεις των διαφόρων χρονικών απαιτήσεων (1984).

Με τον όρο **ομαδοποίηση, συγκέντρωση ή συνάθροιση** (clustering, aggregation) των δεδομένων εννοείται η τεχνική τοποθέτησης των εγγραφών που σχετίζονται και θα αναζητηθούν ταυτόχρονα στην ίδια σελίδα ή σε γειτονικές σελίδες στο δίσκο. Κατά τον τρόπο αυτό επιτυγχάνεται καλύτερη επίδοση κατά την αναζήτηση. Για παράδειγμα, δίνεται ένα αρχείο ανταλλακτικών με πρωτεύον κλειδί τον κωδικό του ανταλλακτικού που είναι ένα αλφαριθμητικό πεδίο. Έστω, επίσης ότι τα κομμάτια με κωδικούς από 217AK300 ως 217AK700 αποθηκεύονται στην ίδια περιοχή του δίσκου. Ωστόσο, διαδοχικές εισαγωγές εκατό κομματιών από 217AK400 ως το 217AK499 μπορεί να προκαλέσουν πρόβλημα στη φυσική οργάνωση. Μερικές δομές μπορούν να απορροφήσουν τη συγκέντρωση των δεδομένων χωρίς ιδιαίτερα προβλήματα (οικογένεια Β-δένδρων). Σε άλλες δομές όμως δημιουργείται τοπική υπερχειλίση με τη συνακόλουθη πτώση της επίδοσης, ενώ σε άλλα σημεία της δομής το ποσοστό χρησιμοποίησης του χώρου είναι σχετικά χαμηλό. Σωστή φυσική σχεδίαση της συνάθροισης των δεδομένων απαιτεί σύνθετα μαθηματικά εργαλεία, όπως για παράδειγμα μαθηματικό προγραμματισμό. Η επισκόπηση από τον March (1983) είναι ιδιαίτερα κατατοπιστική.

Σε πολλές εφαρμογές πραγματικού χρόνου είναι πολύ κρίσιμο θέμα να υπάρχει υψηλή **διαθεσιμότητα** (availability) του αρχείου, δηλαδή να είναι ανά πάσα στιγμή διαθέσιμο και πλήρως ενημερωμένο. Στις περιπτώσεις αυτές δεν ισχύει η μέθοδος της διατήρησης κύριου αρχείου και αρχείου συναλλαγών. Όλες οι εισαγωγές και διαγραφές πρέπει να γίνουν στο κύριο αρχείο που πρέπει να είναι στην εντέλεια ενημερωμένο. Αυτό έχει ως συνέπεια την αποφυγή σειριακών αρχείων. Ακόμη μερικές φορές οι συσκευές αποθήκευσης παθαίνουν βλάβες. Για να αποφευχθεί αυτή η καταστροφική περίπτωση, τα δεδομένα αποθηκεύονται δύο ή και περισσότερες φορές σύμφωνα με την πρακτική των συστημάτων RAID (δες Κεφάλαιο 2.6). Έτσι, οι εφαρμογές είναι πάντοτε on-line, ακόμη και αν κάποια δεδομένα χαθούν από βλάβη του λογισμικού ή του δίσκου. Μία δεύτερη εναλλακτική λύση είναι η χρήση φορητών off-line δίσκων και γενικά η τακτική λήψη φεδρικών αντιγράφων (backups).

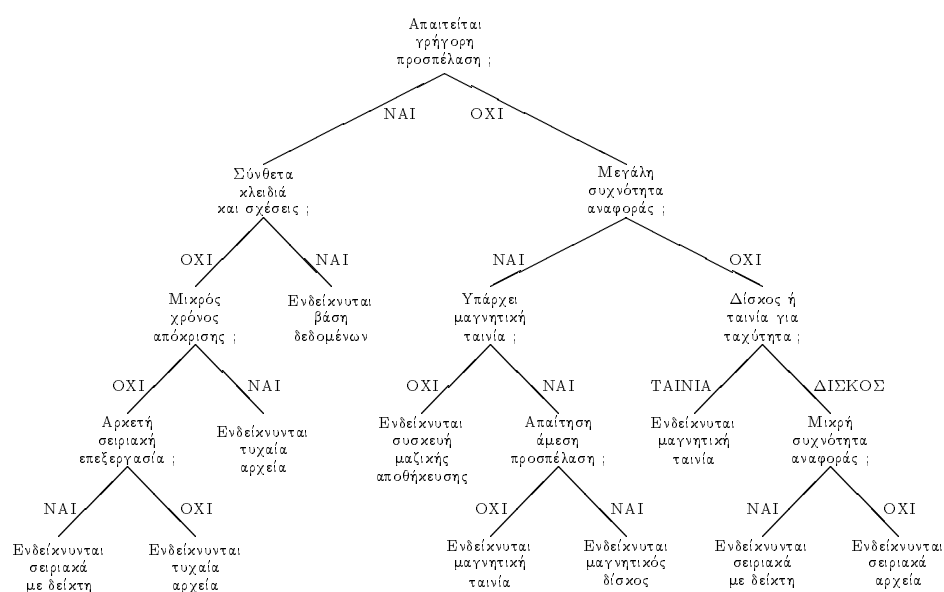
Άλλοτε μπορεί να μην καταστραφεί ο ολόκληρος ο δίσκος, αλλά μόνο ένα μικρό τμήμα του. Παλαιότερα ήταν απαραίτητο να υπάρχουν ρουτίνες ανάκτησης ή αναδημιουργίας των αρχείων από τα εφεδρικά αρχεία και τα υπόλοιπα βοηθητικά ή ιστορικά αρχεία. Στα συστήματα RAID η περίπτωση αυτή αντιμετωπίζεται συστηματικά, και η αποκατάσταση των δεδομένων γίνεται με τη βοήθεια ειδικών κωδικών διόρθωσης λαθών (error correcting codes). Η **ανάκτηση** (recovery), λοιπόν, είναι μία ακόμη σημαντική παράμετρος.

Η διαδικασία της **αναδιοργάνωσης**, όπως εξετάσθηκε, σε δυναμικές δομές γίνεται κατά τη διάρκεια της επεξεργασίας της δομής. Αντίθετα σε στατικές δομές (όπως σειριακά, τυχαία αρχεία, ISAM κλπ.), όταν περιοδικά εκτελείται αυτή η αναγκαία διαδικασία, τότε το αρχείο τίθεται off-line, δηλαδή δεν είναι διαθέσιμο στους χρήστες. Το πρόβλημα της βέλτιστης επιλογής του χρόνου αναδιοργάνωσης έχει προσεγγισθεί αναλυτικά κατά πολλούς τρόπους που αναφέρονται στην επισκόπηση του Sockut (1979). Στην πράξη ίσως χρησιμοποιούνται μάλλον πρακτικές συνταγές για τη σωστή επιλογή του χρόνου αναδιοργάνωσης. Για παράδειγμα, αν γίνει δεκτό ότι σε αρχείο ISAM οι εγγραφές υπερχειλίσας καταλαμβάνουν το 80% του συνολικού χώρου που έχει διατεθεί για την υπερχείλιση, τότε μία τέτοια πρακτική συνταγή υπαγορεύει ότι: *‘Αναδιοργάνωση του αρχείου πρέπει να γίνει την πρώτη νύκτα που η περιοχή υπερχειλίσας υπερβαίνει το 75% του συνολικού χρησιμοποιημένου χώρου και όταν η μέση ημερήσια αύξηση της περιοχής υπερχειλίσας είναι 10%’*.

Η **τρωσιμότητα** (vulnerability) σχετίζεται με την ικανότητα των δομών να είναι ανθεκτικές σε πιθανά λάθη. Σε μερικές δομές είναι πιο εύκολο από ότι σε άλλες να χαθούν δεδομένα από λάθος προγραμματιστικό. Έστω ότι υπάρχει μία αλυσίδα από δείκτες, όπου η εγγραφή Α έχει ένα δείκτη προς την εγγραφή Β, η εγγραφή Β έχει ένα δείκτη προς την εγγραφή Γ κοκ. Αν από λάθος καταστραφεί ένα δείκτης της ακολουθίας των δεικτών, τότε μερικά δεδομένα θα χαθούν. Όμως, με χρήση, για παράδειγμα, διπλής αλυσίδας μπορεί να αποφευχθεί αυτή η δυνατότητα. Μία ενδιαφέρουσα ερευνητική περιοχή είναι οι **εύρωστες δομές** (robust structures) που μελετούν όλες τις γνωστές μεθόδους οργάνωσης (για παράδειγμα Β-δένδρα, λίστες κλπ.) ώστε χρησιμοποιώντας πλεονάζοντες (redundant) δείκτες να επιτευχθεί αυξημένη αξιοπιστία, έστω και σε περίπτωση που συμβεί κάποιο λάθος.

Ανακεφαλαιώνοντας αναφέρεται ότι μερικοί από τους προηγούμενους παράγοντες είναι συγχρουόμενοι. Για το λόγο αυτό, στη φάση της φυσικής σχεδίασης δίνεται προτεραιότητα του ενός παράγοντα έναντι του άλλου. Τα πιο συνηθισμένα ζευγάρια αντιτιθέμενων παραγόντων είναι:

- η πυκνότητα συνάθροισης έναντι της ταχύτητας ανάκτησης,
- η ταχύτητα ανάκτησης έναντι του κόστους του υλικού,
- η ευελιξία στην αναζήτηση έναντι της ταχύτητας ανάκτησης,
- η ευκολία στη διατήρηση έναντι της πυκνότητας συνάθροισης,
- η χαμηλή πολυπλοκότητα έναντι της αποτελεσματικότητας, και τέλος
- η ανακτησιμότητα έναντι της υψηλής πολυπλοκότητας.



Σχήμα 13.1: Δένδρο αποφάσεων για επιλογή οργάνωσης και μέσου.

Τις περισσότερες φορές η επιλογή είναι σχετικά εύκολη, ωστόσο συχνά υπάρχουν δυσκολίες επιλογής αφ' ενός μεταξύ σειριακών αρχείων και σειριακών αρχείων με δείκτη αφ' ετέρου μεταξύ σειριακών αρχείων με δείκτη και τυχαίων αρχείων. Στα φύλλα του δένδρου αποφάσεων του Σχήματος 13.1 παρουσιάζονται οι κατάλληλες οργανώσεις για μερικές περιπτώσεις

που μπορεί να εμφανισθούν συχνά στην πράξη. Ο αναγνώστης καλείται να συμπληρώσει και να ολοκληρώσει τον πίνακα με τις δομές που δεν αναφέρονται στον πίνακα αυτό, αλλά έχουν ήδη παρουσιασθεί στα προηγούμενα κεφάλαια.

13.3 Επίλογος

Τα αρχεία πρέπει να θεωρηθούν ως φυσικές επεκτάσεις των δομών δεδομένων της κύριας μνήμης. Οι δύο βασικές διαφορές τους είναι το μέγεθος και η επίδοσή τους. Παρ' ότι διατίθενται εμπορικά ολόένα και περισσότερο μεγάλες και φθηνές κύριες μνήμες, εν τούτοις δεν μπορούν να ικανοποιήσουν μεγάλες εφαρμογές. Στην πράξη δεν είναι ασυνήθιστο να συναντώνται αρχεία πολλών εκατομμυρίων εγγραφών και το μέγεθός τους να προσεγγίζει το διατιθέμενο χώρο στο δίσκο.

Επίσης, είναι γνωστό ότι η κύρια μνήμη δεν μπορεί να διατηρήσει τα δεδομένα μετά το τέλος του προγράμματος που τα χρησιμοποίησε, γιατί άλλα δεδομένα έρχονται με τη σειρά τους στην κύρια μνήμη. Εξ άλλου τα δεδομένα της κύριας μνήμης χάνονται μόλις σταματήσει η παροχή του ηλεκτρικού ρεύματος. Έτσι με τη βοήθεια της δευτερεύουσας μνήμης τα αρχεία μπορούν να διατηρηθούν για πολύ καιρό.

13.4 Ασκήσεις

<1> Η εταιρεία πιστωτικών καρτών 'Πλαστικό Χρήμα ΑΕ' διατηρεί τα δεδομένα των πελατών της σε έξι αρχεία με τα εξής χαρακτηριστικά.

- Αρχείο τρεχόντων λογαριασμών με 100.000 εγγραφές των 50 bytes. Καθημερινά γίνονται προσπελάσεις στο αρχείο αυτό για ερωτήσεις που αφορούν στο υπόλοιπο και το πιστωτικό όριο της χάρτας.
- Κύριο αρχείο πελατών με 100.000 εγγραφές των 200 bytes. Η προσπελάση μπορεί να είναι είτε άμεση είτε σειριακή. Το αρχείο ενημερώνεται κατά τη διάρκεια της νύχτας.
- Αρχείο συναλλαγών με 3.000 εγγραφές των 100 bytes. Δημιουργείται καθημερινά και οι προσπελάσεις γίνονται σειριακά με κλειδί τον αριθμό της χάρτας. Ποτέ δεν γίνονται ενημερώσεις.

- Ιστορικό αρχείο συναλλαγών με 20.000 εγγραφές των 100 bytes. Ενημερώνεται καθημερινά και περιέχει τις συναλλαγές της εβδομάδας. Οι προσπελάσεις είναι πολύ λίγες και γίνονται σειριακά με κλειδί τον αριθμό της κάρτας.
- Εφεδρικό αρχείο πελατών με 100.000 εγγραφές των 200 bytes. Ενημερώνεται κάθε εβδομάδα ενώ προσπελάσεις γίνονται μόνο σε περιπτώσεις ανάγκης.
- Εφεδρικό ιστορικό αρχείο συναλλαγών με 80.000 εγγραφές των 100 bytes. Περιέχει τις συναλλαγές των τεσσάρων τελευταίων εβδομάδων. Ενημερώνεται κάθε εβδομάδα, ενώ προσπελάσεις γίνονται σε περίπτωση ανάγκης.

Για κάθε ένα από αυτά τα αρχεία να επιλεγθεί μία μέθοδος προσπέλασης που ελαχιστοποιεί το κόστος διατήρησης του αρχείου και παρέχει ταυτόχρονα όλες τις ζητούμενες απαιτήσεις. Οι διατιθέμενες μέθοδοι προσπέλασης είναι η σειριακή, η άμεση και η σειριακή με δείκτη. Τα δευτερεύοντα μέσα αποθήκευσης είναι η ταινία, που κοστίζει 5\$ το μήνα και 1\$ για κάθε χρήση της, και ο δίσκος που κοστίζει 0.8\$ ανά άτρακτο. Δίνεται ότι ο δίσκος αποτελείται από 40.000 Mb/άτρακτο και 10 ατράκτους/κύλινδρο. Επίσης, να θεωρηθεί ότι η άμεση μέθοδος προσπέλασης απαιτεί 40% επιπλέον χώρο, ενώ η σειριακή με δείκτη απαιτεί 25% επιπλέον.

Η απάντηση πρέπει να περιέχει υπολογισμούς του κόστους για κάθε μέθοδο προσπέλασης σε συνδυασμό με το μέσο αποθήκευσης, έτσι βέβαια ώστε να ικανοποιούνται οι απαιτήσεις του χρήστη.

<2> Να σχεδιασθούν οι κατάλληλες δομές αρχείων για μία βάση δεδομένων αεροπορικής εταιρείας. Οι πληροφορίες συνηθέστατα αναζητώνται με τα εξής κλειδιά:

- Ονομα Επιβάτη,
- Αριθμός πτήσης και Ημερομηνία,
- Τόπος Αναχώρησης και Προορισμού, και
- Χρονική Διάρκεια.

Να θεωρηθούν εναλλακτικές δομές πρωτευόντων κλειδιών με γνώμονα ότι οι δομές πρέπει να ενοποιηθούν σε ενιαίο σύστημα. Τα δεδομένα που κρατούνται είναι τα εξής.

- Πρόγραμμα πτήσεων. Δοθέντος ενός ζευγαριού πόλεων και ημέρας της εβδομάδος και των σχετικών περιορισμών σε αναχωρήσεις και/ή αφίξεις το σύστημα να δίνει τις εναλλακτικές διαδρομές. Επίσης, πρέπει να διατηρούνται στοιχεία για τη διαθεσιμότητα των κλάσεων σε θέσεις, ώστε να εμφανίζονται μόνο οι πτήσεις που διαθέτουν θέσεις.
- Πτήσεις. Για κάθε στιγμιότυπο πτήσης (συνδυασμός αριθμού πτήσης και ημερομηνίας) πρέπει να διατηρούνται πληροφορίες για τον τύπο του αεροσκάφους, τις ώρες αναχώρησης και άφιξης για κάθε στάση. Επίσης θα πρέπει για κάθε τμήμα πτήσης να διατηρούνται πληροφορίες για τη διαθεσιμότητα κάθε τάξης σε θέσεις, για τη διαθεσιμότητα σε όγκο και βάρος φορτίου, για φορτώσεις γευμάτων, για τους επιβάτες και για τη λίστα αναμονής. Ακόμη, για κάθε στάση θα πρέπει να διατηρείται ο αριθμός των επιβατών που μεταβιβάζονται σε άλλη πτήση και ο αριθμός των επιβατών που επιβιβάζονται στην πτήση αυτή.
- Επιβάτες. Για κάθε επιβάτη, είτε έχει κρατήσει θέση είτε είναι σε λίστα αναμονής, θα πρέπει να διατηρούνται τα εξής στοιχεία: όνομα και τίτλος, διεύθυνση, δρομολόγιο και κωδικός εισητηρίου, τύπος γεύματος, απαιτήσεις για ξενοδοχείο και ενοικίαση αυτοκινήτου και ονόματα συνεπιβατών.

Οι διαδικασίες που θα πρέπει να υποστηρίζει το σύστημα ομαδοποιημένες ανάλογα με τη συχνότητα που υποβάλλονται ως εξής.

- Πολλές φορές ανά δευτερόλεπτο. Κράτηση θέσης, ακύρωση κράτησης, ερώτηση πιθανών δρομολογίων, έλεγχος διαθεσιμότητας θέσεων, έκδοση εισητηρίου.
- Πολλές φορές ανά ώρα. Δημιουργία στιγμιότυπου πτήσης (συνήθως μερικούς μήνες πριν την αναχώρηση της πτήσης), δημιουργία πληροφοριών αναχώρησης (ώρα αναχώρησης, πληροφορίες ακυρώσεων, απώλειες συνδέσεων κλπ.) και διαγραφή στιγμιότυπων πτήσεων (συνήθως ένα μήνα μετά την πτήση κρατούνται στατιστικά στοιχεία και τα περισσότερα δεδομένα διαγράφονται).
- Μερικές φορές το χρόνο. Αναθεώρηση στοιχείων πτήσεων (τύπος αεροσκάφους, δρομολόγια, ώρες αναχώρησης και άφιξης κλπ.)

- Οργάνωση μνήμης. Το σύστημα δίσκων μπορεί να αποθηκεύσει όλα τα αρχεία, ενώ στην κύρια μνήμη υπάρχει απομονωτική μνήμη που μπορεί να χωρέσει τους καταλόγους και μερικές σελίδες δεδομένων από τη δευτερεύουσα αποθήκευση.

Κύριος σκοπός του φυσικού σχεδιασμού είναι η γρήγορη απόκριση των πολύ συχνών ερωτήσεων (μερικές αεροπορικές εταιρείες δέχονται ένα εκατομμύριο συναλλαγές την ημέρα). Δευτερεύων σκοπός είναι η αποφυγή περιττών επαναλήψεων στοιχείων. Για παράδειγμα, τα στοιχεία του επιβάτη να κρατούνται μόνο μία φορά αν και ο επιβάτης μπορεί να επιβιβασθεί σε πολλές διαδοχικές πτήσεις.

Υποτίθεται ότι εκ μέρους ενός κατασκευαστικού οίκου λογισμικού πρόκειται να υποβάλλετε σε μία μεγάλη αεροπορική εταιρεία (και να αμοιφθείτε γενναιόδωρα) μία έκθεση μερικών σελίδων (χωρίς μαθηματικές αποδείξεις αλλά μόνο με ποιοτικά στοιχεία) με τις πιθανές επιλογές και την προτιμότερη τελική λύση για τις οργανώσεις των αρχείων που θα χρησιμοποιήσετε.

Κεφάλαιο 14

ΝΕΟΤΕΡΕΣ ΕΞΕΛΙΞΕΙΣ

- 14.1 Εισαγωγή
- 14.2 R-δένδρα
- 14.3 Γραμμικά τετραδικά δένδρα περιοχών
- 14.4 Χρονικά διασπώμενο B-δένδρο
- 14.5 Χρονικός κατάλογος
- 14.6 RT-δένδρα
- 14.7 Επικαλυπτόμενα τετραδικά δένδρα περιοχών
- 14.8 Επίλογος

Κεφάλαιο 14

ΝΕΟΤΕΡΕΣ ΕΞΕΛΙΞΕΙΣ

14.1 Εισαγωγή

Τα συμβατικά συστήματα διαχείρισης βάσεων δεδομένων μπορούν να αναπαριστούν και να διαχειρίζονται με μεγάλη ευκολία απλά βαθμωτά δεδομένα (όπως ακεραίους ή πραγματικούς αριθμούς, ημερομηνίες, συμβολοσειρές κλπ.). Ωστόσο, υπάρχει μία πληθώρα νέων σύγχρονων εφαρμογών, όπου οι δομές που εξετάστηκαν στα προηγούμενα κεφάλαια αποδεικνύονται ανεπαρκείς. Ο λόγος είναι ότι στις εφαρμογές αυτές υπεισέρχονται νέοι τύποι δεδομένων (για παράδειγμα, μη βαθμωτά μεγέθη), οπότε απαιτούνται εντελώς νέες δομικές προσεγγίσεις σε σχέση με την αποτελεσματική αποθήκευση και διαχείριση των δεδομένων αυτών. Στο κεφάλαιο αυτό θα εξετασθούν δομές που χρησιμοποιούνται σε τρεις περιπτώσεις εφαρμογών: σε χωροταξικές, σε διαχρονικές και σε χωροχρονικές βάσεις δεδομένων. Στη συνέχεια, για κάθε μία από τις περιπτώσεις αυτές δίνεται το πλαίσιο αναφοράς τους.

Οι **χωροταξικές βάσεις δεδομένων** (spatial databases) καλύπτουν ένα ευρύ φάσμα εφαρμογών που διαχειρίζονται γεωμετρικά δεδομένα, όπως σημεία, γραμμές, πολύγωνα, επιφάνειες ή όγκους σε χώρους δύο, τριών ή και περισσότερων διαστάσεων. Προφανώς όσο μεγαλώνει ο αριθμός των διαστάσεων, τόσο το πρόβλημα της διαχείρισης καθίσταται δυσκολότερο. Η αναπαράσταση και η επεξεργασία πολυδιάστατων (multi-dimensional) αντικειμένων απαιτούνται σε πολλές εφαρμογές, όπως:

Χαρτογραφία. Υπάρχει η ανάγκη για αναπαράσταση χαρτών, έτσι ώστε να υποβάλλουμε ερωτήσεις γεωμετρικού τύπου, όπως: 'Ποιό είναι

το πλησιέστερο σχολείο προς δοθείσα κατοικία;’, ‘Ποιά νοσοκομεία βρίσκονται εντός της συγκεκριμένης περιοχής;’, ‘Ποιά δάση διασχίζονται από ποτάμια;’, κοκ.

Συστήματα CAD. Σε συστήματα σχεδιασμού κυκλωμάτων VLSI πρέπει να αποθηκεύουμε ένα μεγάλο αριθμό αντικειμένων (όπως transistors, πύλες, κλπ.), που αποτελούν το ολοκληρωμένο κύκλωμα, καθώς και τις αντίστοιχες συνδέσεις.

Ρομποτική. Σε βιομηχανικά συστήματα είναι απαραίτητο να υπάρχουν δυνατότητες τεχνητής όρασης (computer vision). Στις περιπτώσεις αυτές, οι αντίστοιχοι αισθητήρες (sensors), πρέπει να διακρίνουν τους όγκους, τις επιφάνειες και τις αποστάσεις των διαφόρων αντικειμένων, και να αλληλεπιδρούν με αντίστοιχα αποθηκευμένα δεδομένα.

Έμπειρα Συστήματα. Σε έμπειρα συστήματα βάσεων δεδομένων χρησιμοποιούνται κανόνες (rules) για την εξαγωγή γνώσης. Η αναζήτηση του κατάλληλου κανόνα μέσα από ένα σύνολο κανόνων διευκολύνεται αν υπάρχει ένας κατάλογος (rule indexing). Έτσι, οι κανόνες θεωρούνται ως γεωμετρικά αντικείμενα (για παράδειγμα, σημεία ή γραμμές), οπότε η αναζήτηση κανόνων ανάγεται σε γεωμετρικό πρόβλημα.

Η αναγκαιότητα για χωροταξικές βάσεις δεδομένων έχει γίνει πλέον αντιληπτή από τους βιομηχανικούς κατασκευάστες βάσεων δεδομένων, που έχουν ήδη εμπλουτίσει τα προϊόντα τους με αντίστοιχα εργαλεία (για παράδειγμα, το Spatial Data Cartridge της Oracle, χωροταξικά Datablades τρίτων κατασκευαστών για την Informix κλπ.).

Οι παραδοσιακές βάσεις δεδομένων αναπαριστούν την παρούσα κατάσταση κάθε φορά. Έτσι, όταν συμβαίνει κάποια διαγραφή ή κάποια ανανέωση χάνονται τα προηγούμενα δεδομένα. Ωστόσο, υπάρχουν πολλές εφαρμογές όπου απαιτείται η ικανοποίηση ερωτήσεων που σχετίζονται με το παρελθόν ή το μέλλον. Για παράδειγμα, ερωτήσεις που αφορούν στο παρελθόν θα μπορούσαν να είναι: ‘Ποιά ήταν η εξέλιξη του τιμάριθμου από 1/1/1990 μέχρι 31/12/1999;’, ή ‘Πότε ο μισθός του υπαλλήλου Α ήταν μεγαλύτερος από το μισθό του υπαλλήλου Β κατά το χρονικό διάστημα από 1/1/1990 μέχρι 31/12/1999;’. Αντίστοιχα, μία ερώτηση σχετικά με μέλλον θα μπορούσε να είναι: ‘Ποιός θα είναι ο πληθυσμός της Ελλάδας το 2.010;’ (θεωρώντας το μέσο ρυθμό αύξησης της τελευταίας δεκαετίας). Οι ερωτήσεις αυτές είναι κλασικές περιπτώσεις που δεν μπορούν να ικανοποιηθούν από παραδοσιακές αλλά από διαχρονικές βάσεις δεδομένων (temporal databases).

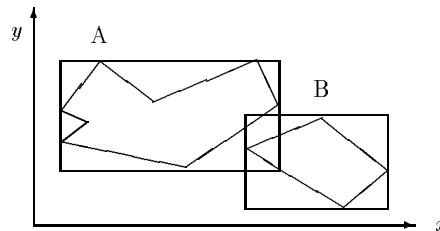
Στις βάσεις αυτές, οι διαγραφές και οι ανανεώσεις δεν εκτελούνται με φυσικό τρόπο αλλά με λογικό, δηλαδή τα αντίστοιχα δεδομένα συνεχίζουν να υπάρχουν στη βάση αλλά 'σημαδεμένα' με κατάλληλες χρονοσφραγίδες (timestamps). Είναι προφανές ότι σε τέτοιες εφαρμογές υπάρχουν κάποια ιδιαίτερα χαρακτηριστικά. Το πρώτο χαρακτηριστικό είναι ότι οι αντίστοιχες δομές καταλόγων απαιτούν τεράστιο αποθηκευτικό χώρο καθώς συνεχώς θα δέχονται συνεχώς νέα δεδομένα. Η απαίτηση αυτή ήταν πράγματι ένα πρόβλημα στο παρελθόν, όχι όμως πλέον σήμερα καθώς οι χωρητικότητες των δίσκων είναι τεράστιες και θα συνεχίσουν να αυξάνουν στο μέλλον με ραγδαίο τρόπο (δες Κεφάλαιο 2). Η δεύτερη απαίτηση είναι ότι οι υποβαλλόμενες ερωτήσεις δεν αφορούν μόνο τις τιμές διαφόρων πεδίων, αλλά επίσης προσδιορίζουν ή ερωτούν για τιμές της διάστασης του χρόνου. Σε σχέση με την απαίτηση αυτή έχουν προταθεί πολλές εναλλακτικές λύσεις που θα εξετασθούν σε επόμενα υποκεφάλαια και που προσπαθούν να συμβιβάσουν την επίδοση των διαφόρων ερωτήσεων κατά τον αποτελεσματικότερο τρόπο. Επί του παρόντος δεν υπάρχουν εμπορικές διαχρονικές βάσεις δεδομένων. Ωστόσο, δεν θα αργήσει αυτή η στιγμή καθώς έχει διαπιστωθεί η χρησιμότητα της διάστασης του χρόνου. Χαρακτηριστικό είναι το γεγονός ότι η νέα έκδοση της γλώσσας SQL διαθέτει εντολές που σχετίζονται με τη διαχείριση του χρόνου.

Η περιοχή των χωροχρονικών βάσεων δεδομένων (spatio-temporal databases) είναι μία νέα περιοχή που προέκυψε από το συνδυασμό των δύο προηγούμενων. Για παράδειγμα, συχνά τίθενται ερωτήσεις που σχετίζονται με αλλαγές γεωμετρικών αντικειμένων στο χρόνο. Ο όρος 'αλλαγή' υποδηλώνει πολλές περιπτώσεις, όπως: την κίνηση σημείων, επιφανειών ή στερεών στο χώρο (δηλαδή προβλήματα σχετικά με τροχιές), την επέκταση επιφανειών, τη διόγκωση στερεών σωμάτων κλπ. Για παράδειγμα, χωροχρονικές ερωτήσεις θα μπορούσαν να είναι: 'Ποιά ελικόπτερα μπορούν να βρισκονται εντός της συγκεκριμένης περιοχής μέσα σε 5 λεπτά;', 'Ποιά ήταν η μικρότερη απόσταση μεταξύ των τροχιών δύο αεροπλάνων;', 'Ποιά ήταν η έκταση της λίμνης Λαγκαδά από 1/1/1990 μέχρι 31/12/1999;' ή 'Ποιές ήταν οι 10 ευρωπαϊκές πόλεις με το μεγαλύτερο ποσοστό ηλιοφάνειας κατά το 1999;'. Είναι προφανές ότι οι ερωτήσεις αυτές είναι συνθετότερες από αντίστοιχες χωροταξικές ή διαχρονικές ερωτήσεις. Δομές που υποστηρίζουν τέτοιου τύπου ερωτήσεις θα εξετασθούν στα τελευταία υποκεφάλαια του παρόντος κεφαλαίου.

14.2 R-δένδρα

Τα R-δένδρα (R-trees) προτάθηκαν από τον Guttman (1984), και το όνομα της προέρχεται από το αρχικό γράμμα της αγγλικής λέξης rectangle. Τα R-δένδρα χρησιμεύουν για την αποθήκευση και διαχείριση οποιωνδήποτε γεωμετρικών αντικειμένων σε οποιονδήποτε αριθμό διαστάσεων. Για λόγους ευκολίας, στη συνέχεια η περιγραφή της δομής αφορά χώρο δύο διαστάσεων.

Το σημείο αφετηρίας είναι ότι κάθε αντικείμενο που πρέπει να αποθηκευθεί προσδιορίζεται από το ελάχιστο περιγεγραμμένο ορθογώνιο (minimum bounding rectangle, MBR) με πλευρές παράλληλες προς τους άξονες, όπως φαίνεται στο Σχήμα 14.1. Δηλαδή, στη δομή δεν αποθηκεύονται οι λεπτομέρειες του αντικειμένου, αλλά μόνο το ορθογώνιό του. Αυτό απλουστεύει τη δομή καθώς όσο περίπλοκο και αν είναι ένα αντικείμενο, τελικά αντιπροσωπεύεται από ένα απλό ορθογώνιο. Βέβαια αν υποβληθεί μία ερώτηση στη δομή, τότε τα ορθογώνια που θα προκύψουν στην έξοδο πρέπει να εξεταστούν λεπτομερέστερα, διότι με την αναπαράσταση αυτή χάνεται η επακριβής μορφή του αντικειμένου. Για παράδειγμα, έστω η ερώτηση: 'Ποιά αντικείμενα τέμνονται από το αντικείμενο A;'. Στο σχήμα φαίνεται ότι αν και το ορθογώνιο του αντικειμένου A τέμνεται από το ορθογώνιο του αντικειμένου B, εντούτοις τα αντικείμενα A και B δεν τέμνονται.



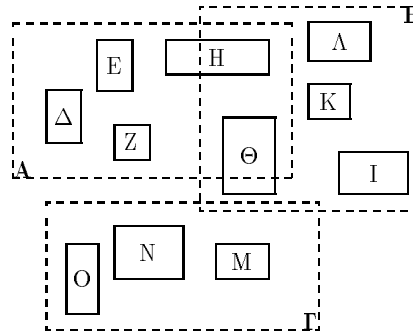
Σχήμα 14.1: Τομή δύο ορθογωνίων χωρίς να τέμνονται τα αντικείμενα.

R-δένδρο τάξης (m, M) είναι το ετερογενές δένδρο με τα εξής χαρακτηριστικά:

- κάθε φύλλο περιέχει μεταξύ M και $m \leq M/2$ στοιχείων του τύπου (R, O) , εκτός αν είναι ρίζα. Για κάθε ζεύγος (R, O) , R είναι το ορθογώνιο του αντικειμένου που προσδιορίζεται από τον κωδικό O ,

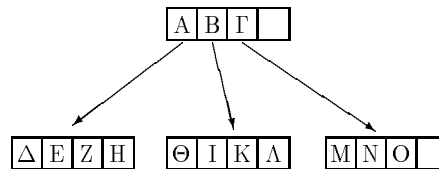
- κάθε εσωτερικός κόμβος περιέχει μεταξύ M και $m \leq M/2$ στοιχείων τύπου (R,P) , εκτός αν είναι ρίζα. Για κάθε ζεύγος (R,P) , P είναι ένας δείκτης προς κόμβο κατωτέρου επίπεδου, ενώ R είναι το ορθογώνιο που περιλαμβάνει όλα τα ορθογώνια του αντίστοιχου κόμβου.
- η ρίζα έχει τουλάχιστον δύο παιδιά, εκτός αν είναι φύλλο, και τέλος
- όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο.

Από το ορισμό φαίνεται ότι το R-δένδρο είναι ένα ισοζυγισμένο δένδρο και αποτελεί γενίκευση του B-δένδρου σε περισσότερες διαστάσεις. Επίσης, σημειώνεται ότι οι κόμβοι του δένδρου αντιστοιχούν σε σελίδες δίσκου, ενώ η δομή είναι δυναμική και δεν απαιτείται περιοδική αναδιοργάνωση λόγω εισαγωγών ή διαγραφών.



Σχήμα 14.2: Ορθογώνια στο επίπεδο.

Στο Σχήμα 14.2 φαίνεται ένα σύνολο ορθογωνίων που αποτελούν τα ορθογώνια κάποιων δισδιάστατων αντικειμένων. Τα βασικά ορθογώνια που μας ενδιαφέρουν είναι τα Δ , E , Z , H , Θ , I , K , Λ , M , N και O , που αποθηκεύονται στα φύλλα του R-δένδρου, όπως φαίνεται στο Σχήμα 14.3. Τα



Σχήμα 14.3: R-δένδρο για τα ορθογώνια του Σχήματος 14.2.

ορθογώνια A , B και Γ αποτελούν τα ορθογώνια που βρίσκονται αποθηκευμένα στο επίπεδο επάνω από τα φύλλα, δηλαδή στη ρίζα. Από το σχήμα αυτό γίνεται φανερό ότι για ένα σύνολο ορθογωνίων (ή στη γενικότερη περίπτωση, οποιωνδήποτε γεωμετρικών αντικειμένων) δεν υπάρχει ένα και μοναδικό R -δένδρο. Αυτό οφείλεται στο γεγονός ότι η διαμόρφωση ενός R -δένδρου εξαρτάται από τη σειρά εισαγωγής και διαγραφής των στοιχείων στη/από τη δομή. Έτσι είναι δυνατό για τα ίδια ακριβώς ορθογώνια να υπάρχουν πολλές διαφορετικές αναπαραστάσεις R -δένδρων.

Έστω ένα R -δένδρο που διαχειρίζεται N ορθογώνια. Στην περίπτωση αυτή, η μέγιστη τιμή του ύψους είναι:

$$h_{max} = \lceil \log_m N \rceil - 1$$

Ο μέγιστος αριθμός κόμβων του δένδρου προκύπτει αθροίζοντας το μέγιστο αριθμό κόμβων ανά επίπεδο. Ο αριθμός αυτός προκύπτει όταν οι κόμβοι περιέχουν τον ελάχιστο αριθμό στοιχείων, δηλαδή m . Έτσι, συνάγεται ότι ο μέγιστος αριθμός κόμβων σε ένα R -δένδρο είναι

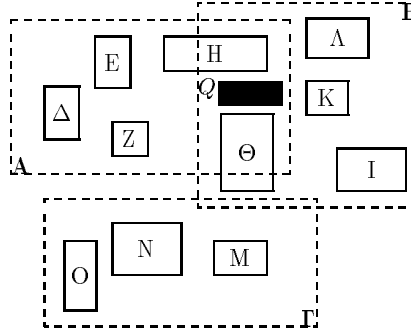
$$\sum_{i=1}^{h_{max}} \lceil N/m^i \rceil = \lceil N/m \rceil + \lceil N/m^2 \rceil + \dots + 1$$

Συνεπώς, στη χειρότερη περίπτωση ο παράγοντας χρησιμοποίησης χώρου για όλους τους κόμβους του δένδρου, εκτός της ρίζας, είναι:

$$U = m/M$$

ενώ για τη ρίζα είναι $2/M$, γιατί ρίζα έχει τουλάχιστον 2 παιδιά.

Έστω ότι δίνεται μία ορθογώνια περιοχή Q του επιπέδου (το λεγόμενο παράθυρο) και τίθεται μία ερώτηση διαστήματος: 'Ποιά αντικείμενα τέμνονται από το παράθυρο Q ;'. Από το Σχήμα 14.4 φαίνεται ότι το παράθυρο Q βρίσκεται στο εσωτερικό των ορθογωνίων A και B . Αυτό σημαίνει ότι για να βρεθούν τα αντικείμενα που τέμνει το Q , θα πρέπει να ακολουθηθούν δύο μονοπάτια του R -δένδρου από τη ρίζα προς τα φύλλα, ένα μονοπάτι για το A και ένα για το B . Το μειονέκτημα αυτό είναι αναπόφευκτο επειδή η δομή επιτρέπει την επικάλυψη των ορθογωνίων, ακόμη και στα φύλλα. Οι αλγόριθμοι εισαγωγής και διαγραφής φροντίζουν ώστε να μειωθεί το φαινόμενο της επικάλυψης στα ανώτερα επίπεδα του δένδρου και να προκύψει μία αποτελεσματική δομή.



Σχήμα 14.4: Το Q είναι το ορθογώνιο ερώτησης.

Στη συνέχεια δίνεται ο αλγόριθμος αναζήτησης σε R-δένδρο. Για κάποιο ζεύγος E ενός κόμβου συμβολίζουμε με $E.I$ το αντίστοιχο ορθογώνιο, ενώ με $E.p$ συμβολίζουμε το πεδίο O ή P , ανάλογα αν ο κόμβος είναι φύλλο ή εσωτερικός.

Αλγόριθμος Search. Αναζητούνται όλα τα ορθογώνια που τέμνουν ένα παράθυρο Q σε ένα R-δένδρο με ρίζα T .

Βήμα 1: διάσχιση υποδένδρων. Αν ο κόμβος T δεν είναι φύλλο, τότε εξετάζεται κάθε ζεύγος E του κόμβου T , ώστε να διαπιστωθεί αν το $E.I$ επικαλύπτει το Q . Για τα ζεύγη που καλύπτουν το Q , καλείται αναδρομικά ο αλγόριθμος Search για κάθε υποδένδρο με ρίζα $E.p$.

Βήμα 2: αναζήτηση στα φύλλα. Αν ο κόμβος T είναι φύλλο, τότε εξετάζονται όλα τα ζεύγη E που είναι αποθηκευμένα στον κόμβο, ώστε να διαπιστωθεί αν το $E.I$ επικαλύπτει το Q . Αν υπάρχει επικάλυψη, τότε το αντίστοιχο ζεύγος επιστρέφεται στην έξοδο.

Σημειώνεται ότι ο προηγούμενος αλγόριθμος περιγράφει την πρώτη φάση κατά τη διαδικασία αναζήτησης, η οποία ονομάζεται **βήμα φιλτραρίσματος** (filter step). Ο αλγόριθμος αυτός στην έξοδο επιστρέφει ορθογώνια, που ως γνωστό είναι προσεγγίσεις των αρχικών αντικειμένων. Επομένως, απαιτείται λεπτομερής έλεγχος της γεωμετρίας των αντίστοιχων αντικειμένων για να διαπιστωθεί αν πράγματι το παράθυρο Q τα τέμνει. Η δεύτερη αυτή φάση ονομάζεται **βήμα διύλισης** (refinement step), και απαιτεί εξειδικευμένους αλγορίθμους υπολογιστικής γεωμετρίας (computational geometry). Για το λόγο αυτό, στα πλαίσια του βιβλίου αυτού δεν εξετάζονται οι λεπτομέρειες της δεύτερης φάσης.

Η διαδικασία εισαγωγής σε R-δένδρο είναι παρόμοια με την αντίστοιχη του B-δένδρου. Με λίγα λόγια, το δένδρο διασχίζεται ακολουθώντας μία πορεία από επάνω προς τα κάτω ώστε να εντοπισθεί το κατάλληλο φύλλο. Εκτελείται η εισαγωγή στο φύλλο και στη συνέχεια ενημερώνονται οι κόμβοι του μονοπατιού προς τη ρίζα ακολουθώντας την αντίστροφη πορεία. Ταυτόχρονα, όλοι οι κόμβοι (φύλλα ή εσωτερικοί) με M ζεύγη, οι οποίοι υπερχειλίζουν διασπώνται σε δύο κόμβους. Ωστόσο, η διάσπαση των κόμβων του R-δένδρου χρειάζεται ιδιαίτερη προσοχή.

Αλγόριθμος Insert. Εισαγωγή νέου ζεύγους E σε R-δένδρο με ρίζα T .

Βήμα 1: εύρεση κατάλληλης θέσης νέου ζεύγους. Διασχίζεται το δένδρο από τη ρίζα T προς το κατάλληλο φύλλο. Το μονοπάτι σχηματίζεται επιλέγοντας σε κάθε επίπεδο τον κόμβο L του οποίου το ορθογώνιο $L.I$ χρειάζεται την ελάχιστη επαύξηση για να καλύψει το ορθογώνιο $E.I$. Αν υπάρχουν πολλές υποψηφιότητες, τότε επιλέγεται το ορθογώνιο με τη μικρότερη επιφάνεια.

Βήμα 2: εισαγωγή ζεύγους E στο L . Αν το L έχει διαθέσιμο χώρο, τότε το ζεύγος E εισάγεται στο L και ενημερώνονται τα ορθογώνια του μονοπατιού προς τη ρίζα, ώστε να καλύπτουν το ορθογώνιο του E .

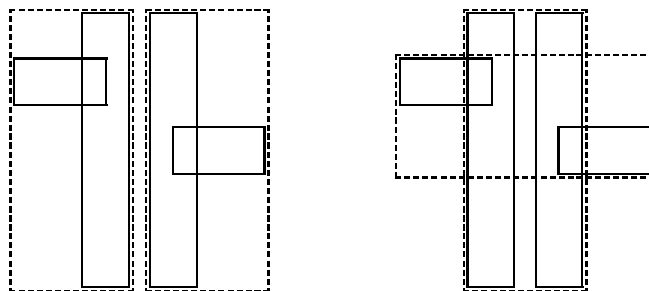
Βήμα 3: διάσπαση L και εισαγωγή ζεύγους E . Αν το L δεν έχει διαθέσιμο χώρο, τότε

- θεωρείται το σύνολο S που αποτελείται από τα ζεύγη του κόμβου L συν το ζεύγος E . Ως σπόροι δύο υποσυνόλων $S1$ και $S2$ επιλέγονται δύο μέλη του S με ορθογώνια που απέχουν περισσότερο μεταξύ τους.
- Τα υπόλοιπα μέλη του S ανατίθενται στο σύνολο $S1$ ή το $S2$, όπου το αντίστοιχο ορθογώνιο επαυξάνεται το λιγότερο δυνατό. Σε περίπτωση ισοπαλίας (δηλαδή, η επαύξηση των δύο ορθογώνιων είναι ίση), τότε το στοιχείο ανατίθεται στο σύνολο με το ορθογώνιο μικρότερου εμβαδού. Σε περίπτωση νέας ισοπαλίας (δηλαδή, τα δύο ορθογώνια έχουν ίσο εμβαδό), τότε το στοιχείο ανατίθεται στο σύνολο που περιέχει λιγότερα ζεύγη. Σε περίπτωση και νέας ισοπαλίας (δηλαδή, τα δύο ορθογώνια έχουν ίσο αριθμό ζευγών), τότε γίνεται τυχαία ανάθεση. Αν κατά την ανάθεση στοιχείων απομείνουν k ζεύγη και το ένα σύνολο έχει $m-k$ ζεύγη, τότε όλα τα εναπομείναντα ζεύγη τοποθετούνται στο συγκεχυμένο σύνολο.

- Τα σύνολα $S1$ και $S2$ αποθηκεύονται στους κόμβους L και LL αντίστοιχα.

Βήμα 4: μετάδοση αλλαγών προς τη ρίζα. Ενημερώνονται τα ορθογώνια του μονοπατιού προς τη ρίζα, ώστε να καλύπτουν τα ορθογώνια των L και LL (πιθανώς εκτελώντας νέες διασπάσεις). Αν η ρίζα υπερχειλίσει, τότε αυτή διασπάται σε δύο κόμβους και δημιουργείται νέα ρίζα με παιδιά τους κόμβους αυτούς.

Η διαμέριση των στοιχείων στους δύο νέους κόμβους πρέπει να γίνει έτσι ώστε να ελαχιστοποιηθεί η πιθανότητα εξέτασης και των δύο υποδένδρων των νέων κόμβων σε επόμενες αναζητήσεις. Το κριτήριο που χρησιμοποιείται στον προηγούμενο αλγόριθμο είναι τα ορθογώνια να απέχουν περισσότερο μεταξύ τους. Το κριτήριο αυτό ουσιαστικά σημαίνει ότι η συνολική επιφάνεια των δύο νέων κόμβων πρέπει να ελαχιστοποιηθεί. Έτσι, επιλέγονται τα στοιχεία που θα σπαταλήσουν περισσότερο χώρο αν τοποθετηθούν στην ίδια ομάδα. Η σπατάλη χώρου μετράται με τη βοήθεια του λεγόμενου **νεκρού χώρου** (dead space), δηλαδή της επιφάνειας που καλύπτουν τα ορθογώνια μείον την επιφάνεια που καλύπτουν τα περιχλειόμενα ορθογώνια. Στο Σχήμα 14.7 φαίνεται μία ‘κακή’ και μία ‘καλή’ διάσπαση, με την έννοια ότι η συνολική επιφάνεια των δύο ορθογωνίων (καθώς και ο νεκρός χώρος) στο αριστερό σχήμα είναι πολύ μεγαλύτερη από αυτήν που καλύπτουν στο δεξιό σχήμα. Προφανώς, το ίδιο κριτήριο χρησιμοποιείται στον αλγόριθμο Insert για να επιλεγεί το υποδένδρο κατά την κάθοδο από τη ρίζα προς τα φύλλα.



Σχήμα 14.5: ‘Κακή’ και ‘καλή’ διάσπαση.

Ο αλγόριθμος διάσπασης κόμβων που περιγράφηκε έχει τετραγωνική πολυπλοκότητα. Σύμφωνα με ένα εναλλακτικό αλγόριθμο, δημιουργούνται

όλες οι δυνατές διαμερίσεις των $M+1$ ζευγών και επιλέγεται αυτή που δίνει τα δύο ορθογώνια (για τους δύο κόμβους) με την ελάχιστη συνολική επιφάνεια. Όλες οι δυνατές διαμερίσεις των $M+1$ στοιχείων είναι περίπου 2^{M-1} . Επομένως, αν και στην έξοδο θα δώσει τελικά τη βέλτιστη διαμέριση, εντούτοις ο αλγόριθμος αυτός είναι χρονικά ιδιαίτερα δαπανηρός καθώς είναι εκθετικής πολυπλοκότητας. Εκτός των δύο αυτών αλγορίθμων, υπάρχει και ένας γραμμικός αλγόριθμος, ωστόσο ο τετραγωνικός είναι η μέθοδος που συμβιβάζει ταχύτητα και αποτελεσματικότητα κατά την αναζήτηση.

Αλγόριθμος Delete. Διαγραφή ζεύγους E από R -δένδρο με ρίζα T .

Βήμα 1: εύρεση φύλλου που περιέχει το ζεύγος. Αν ο κόμβος T είναι φύλλο, τότε ελέγχεται κάθε ζεύγος F του T για να διαπιστωθεί αν τα $F.I$ και $E.I$ τέμνονται. Αλλιώς θεωρούνται (αναδρομικά) όλα τα υποδένδρα της ρίζας T , μέχρι να βρεθεί ένα φύλλο L που περιέχει ένα ζεύγος F που τέμνει το $E.I$, ή μέχρι να εξετασθούν όλα τα υποδένδρα (αν το E δεν υπάρχει στο δένδρο).

Βήμα 2: διαγραφή ζεύγους. Διαγράφεται το E από το φύλλο L .

Βήμα 3: διάδοση αλλαγών. Καλείται ο αλγόριθμος CondenseTree με παράμετρο το L .

Βήμα 4: σμίχρυνση δένδρου. Αν η ρίζα έχει μόνο ένα παιδί, τότε η ρίζα διαγράφεται και τίθεται ως ρίζα το μοναδικό παιδί της.

Αλγόριθμος CondenseTree. Δίνεται ένα φύλλο L από όπου έχει διαγραφεί ένα ζεύγος E . Αν μετά τη διαγραφή το φύλλο έχει λιγότερα από m ζεύγη, τότε καταστρέφεται ο κόμβος-φύλλο και τα ζεύγη του επανα-εισάγονται. Η αλλαγή διαδίδεται προς τα επάνω και τα ορθογώνια του μονοπατιού προς τη ρίζα διευθετούνται (μικραίνοντάς τα, αν χρειασθεί).

Βήμα 1: αρχικοποίηση. Θέτουμε $N=L$. Έστω Q το σύνολο των κόμβων που θα καταστραφούν (αρχικά το κενό σύνολο).

Βήμα 2: εύρεση ζεύγους πατέρα. Αν N είναι η ρίζα, τότε πηγαίνουμε στο Βήμα 6. Αλλιώς, έστω P ο πατέρας του N και EN το ζεύγος του P , που αντιστοιχεί στον κόμβο N .

Βήμα 3: αφαίρεση κόμβου. Αν ο κόμβος N περιέχει λιγότερα από m ζεύγη, τότε διαγράφουμε το ζεύγος EN από τον κόμβο P και εισάγουμε το κόμβο N στο σύνολο Q .

Βήμα 4: διευθέτηση ορθογωνίου. Αν το N δεν έχει εξουδετερωθεί, τότε ενημερώνουμε (πιθανώς) το ορθογώνιο $EN.I$ έτσι ώστε να περιχλείει ακριβώς όλα τα ορθογώνια του κόμβου N .

Βήμα 5: άνοδος προς ρίζα. Θέτουμε $N=P$ και πηγαίνουμε στο Βήμα 2.

Βήμα 6: επανα-εισαγωγή στοιχείων. Επανα-εισάγονται όλα τα στοιχεία που ανήκουν στο σύνολο Q . Τα ζεύγη που ήταν αποθηκευμένα σε φύλλα επανα-εισάγονται καλώντας τον αλγόριθμο *Insert*. Τα ζεύγη που ήταν αποθηκευμένα σε εσωτερικούς κόμβους πρέπει να τοποθετηθούν πιο ψηλά στο δένδρο, γιατί τα φύλλα των υποδένδρων τους πρέπει να βρίσκονται στο ίδιο επίπεδο με τα υπόλοιπα φύλλα του δένδρου.

Η διαχείριση των κόμβων που περιέχουν στοιχεία λιγότερα από το κατώτατο επιτρεπτό όριο διαφέρει από αυτή που πραγματοποιείται σε ένα B -δένδρο. Δηλαδή, στην περίπτωση του B -δένδρου γίνεται συγχώνευση γειτονικών κόμβων, ενώ στο R -δένδρο τα στοιχεία επανα-εισάγονται. Βέβαια, αν και στην περίπτωση των R -δένδρων η έννοια της γειτονικότητας των κόμβων δεν έχει νόημα, θα μπορούσε να γίνει συγχώνευση κόμβων του ίδιου επιπέδου. Όμως προτιμάται η επανα-εισαγωγή των στοιχείων για τους εξής λόγους:

- Με την επανα-εισαγωγή επιτυγχάνεται το ίδιο αποτέλεσμα όπως και στην περίπτωση της συγχώνευσης. Επιπλέον, η διαδικασία *Insert* είναι ήδη διαθέσιμη για χρήση. Επίσης, επειδή ο αριθμός των προσπελάσιμων σελίδων του δίσκου είναι κρίσιμο μέγεθος, οι σελίδες που θα χρειασθούν για την επανα-εισαγωγή θα βρίσκονται ήδη στη μνήμη από τη διαδικασία της εύρεσης του συγκεκριμένου ζεύγους (τουλάχιστον οι περισσότερες από αυτές).
- Όπως έχει περιγραφεί στη διαδικασία *Insert*, η εισαγωγή ενός στοιχείου φροντίζει η δομή να χτίζεται έτσι ώστε οι ερωτήσεις να ικανοποιούνται με αποτελεσματικό τρόπο. Άρα είναι φρόνημο να χρησιμοποιηθεί και στην περίπτωση αυτή καθώς η απόδοση της δομής μπορεί να χειροτερεύσει αν κάποιο στοιχείο βρίσκεται συνεχώς κάτω από τον ίδιο πατέρα.

Το R -δένδρο χρησιμεύει για την αποθήκευση οποιωνδήποτε γεωμετρικών αντικειμένων και επομένως με τη δομή αυτή μπορούν να ικανοποιηθεί μία πληθώρα ερωτήσεων με βάση τον αλγόριθμο αναζήτησης που εξετάστηκε προηγουμένως:

- **ερώτηση τομής ορθογωνίων** (rectangle intersection query): ‘Δεδομένου ενός ορθογωνίου S , για ποιά ορθογώνια R ισχύει $S \cap R \neq \emptyset$;’.
- **σημειακή ερώτηση** (point query): ‘Ποιά ορθογώνια περιέχουν το σημείο P ;’.
- **ερώτηση περιγεγραμμένου ορθογωνίου** (rectangle enclosure query): ‘Δεδομένου ενός ορθογωνίου S , για ποιά ορθογώνια R $S \subseteq R$;’.

Ωστόσο, είναι δυνατόν να τεθούν και άλλες ερωτήσεις, των οποίων η ικανοποίηση απαιτεί τροποποίηση του αλγορίθμου Search. Παραδείγματα τέτοιων ερωτήσεων είναι:

- **ερώτηση πλησιέστερου γείτονα** (nearest neighbor query): ‘Ποιά είναι τα $k \leq 1$ πλησιέστερα σημεία προς το σημείο P ;’.
- **τοπολογική ερώτηση** (topological query): ‘Ποιά ορθογώνια βρίσκονται προς τα δυτικά του ορθογωνίου S ;’.

Από τα προηγούμενα φαίνεται ότι η δομή των R-δένδρων είναι ιδιαίτερα σημαντική για την περιοχή των χωροταξικών βάσεων δεδομένων. Για το λόγο αυτό έχει υλοποιηθεί στα περισσότερα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων, για την υποστήριξη ερωτήσεων όπως οι προηγούμενες, που δεν θα μπορούσαν να υποστηριχθούν με μία δομή B⁺-δένδρου ή με ένα αρχείο κατακερματισμού. Στη βιβλιογραφία αναφέρονται πολλές παραλλαγές του R-δένδρου. Οι σπουδαιότερες είναι τα **συμπιεσμένα** (packed) R-δένδρα που προτάθηκαν από τους Ρουσσόπουλο και Leifker, τα R⁺-δένδρα που προτάθηκαν από τους Σελλή, Ρουσσόπουλο και Φαλούτσο, τα R^{*}-δένδρα που προτάθηκαν από τον Beckman και άλλους, και τα Hilbert R-δένδρα που προτάθηκαν από τους Kamel και Φαλούτσο. Λεπτομερέστερη περιγραφή των δομών αυτών γίνεται στην πρόσφατη επισκόπηση των Gaede και Guenther.

14.3 Γραμμικά τετραδικά δένδρα περιοχών

Ο όρος **Τετραδικό δένδρο** (Quadtree) δηλώνει μία πολυμελή οικογένεια δομών για την αποθήκευση κάθε είδους γεωμετρικού αντικειμένου (δηλαδή, σημεία, γραμμές, πολύγωνα, περιοχές, όγκους κλπ.), ενώ στο επίπεδο της υλοποίησης μπορούν να αποθηκευθούν είτε στην κύρια μνήμη (δες Κεφάλαιο 4.5.2 βιβλίου για Δομές Δεδομένων) είτε στη δευτερεύουσα μνήμη. Ο

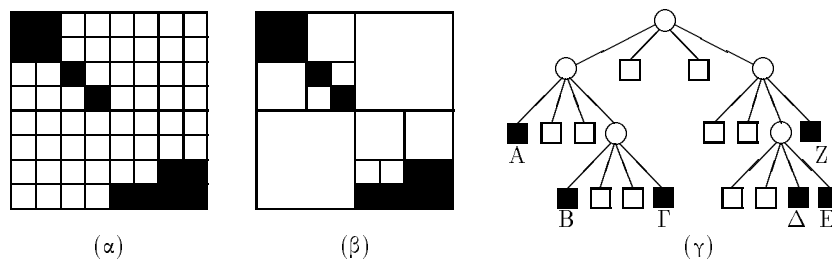
ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στα βιβλία του Samet για πληρέστερη μελέτη των δομών αυτών. Στα πλαίσια του κεφαλαίου αυτού θα εξετασθούν τα Γραμμικά τετραδικά δένδρα περιοχών που χρησιμεύουν για την αποθήκευση και επεξεργασία εικόνων τύπου ψηφιδωτού (raster (από τη γερμανική λέξη που σημαίνει οθόνη), όπως για παράδειγμα είναι οι δορυφορικές εικόνες, οι αεροφωτογραφίες κλπ. Βέβαια, η δομή αυτή είναι δημοφιλής και σε άλλες περιοχές της Πληροφορικής όπως τα Γραφικά και η Επεξεργασία Εικόνων. Πριν παρουσιασθεί η συγκεκριμένη δομή, θα παρουσιασθεί η δομή του τετραδικού δένδρου περιοχών, που είναι δομή για την κύρια μνήμη.

Το Τετραδικό δένδρο περιοχών (Region quadtree) ονομάσθηκε έτσι από τον Hunter το 1978, αν και είχε προταθεί νωρίτερα από τον Klinger ως Q-tree το 1971. Το Τετραδικό δένδρο περιοχών τάξης n είναι μία δομή κατ'άλληλη για την αναπαράσταση δισδιάστατων δεδομένων περιοχών, δηλαδή εικόνων, διαστάσεων $2^n \times 2^n$. Πρόκειται για ένα δένδρο βαθμού τέσσερα με ύψος το πολύ n . Το επίπεδο όπου βρίσκεται η ρίζα ορίζεται ως *επίπεδο n* , το επίπεδο όπου βρίσκονται τα παιδιά της ρίζας ως *επίπεδο $n-1$* , κοκ. Προφανώς, το κατώτατο επίπεδο όπου βρίσκονται οι κόμβοι που αντιστοιχούν σε μεμονωμένα εικονοκύτταρα (pixels) είναι το επίπεδο 0. Ένας κόμβος στο επίπεδο i , όπου $0 \leq i \leq n$, αντιπροσωπεύει ένα τμήμα εικόνας με $2^i \times 2^i$ ($=4^i$) pixels, ενώ υπάρχουν το πολύ 4^{n-i} κόμβοι στο επίπεδο αυτό.

Για τη συνέχεια, θα θεωρηθεί ότι η εικόνα είναι ασπρόμαυρη, αλλά αυτό δεν αποτελεί περιορισμό για τη γενίκευση των τεχνικών σε περισσότερα χρώματα. Εξ άλλου, η έννοια της ασπρόμαυρης εικόνας καλύπτει τις περιπτώσεις θεματικών χαρτών, δηλαδή χαρτών που παρουσιάζουν την ύπαρξη ενός φαινομένου, μίας κατάστασης κλπ, όπως για παράδειγμα την ύπαρξη νεφώσεων ή βροχοπτώσεων, την καλλιέργεια σίτου, τις περιοχές που επηρεάζονται από την τρύπα του όζοντος κλπ.

Η δομή βασίζεται στην αναδρομική διάσπαση της εικόνας (δηλαδή, ενός δισδιάστατου πίνακα με μαύρα και άσπρα pixels) σε τέσσερα ισομεγέθη τεταρτημόρια μέχρι να προκύψουν τμήματα που να είναι αμιγή (δηλαδή, να αποτελούνται από ένα μόνο χρώμα). Πιο συγκεκριμένα, κάθε κόμβος αντιστοιχεί σε έναν τετράγωνο πίνακα από pixels, ενώ η ρίζα αντιστοιχεί σε ολόκληρη την εικόνα. Αν όλα τα pixels που αντιστοιχούν σε ένα κόμβο έχουν το ίδιο χρώμα (άσπρο ή μαύρο), τότε ο κόμβος είναι φύλλο του ίδιου χρώματος. Διαφορετικά, ο κόμβος είναι εσωτερικός (που λέγεται ότι έχει γκρι χρώμα) και έχει τέσσερα παιδιά-υποδένδρα. Κάθε ένα από τα παιδιά

αυτά αντιστοιχεί σε ένα τεταρτημόριο του πίνακα pixels του γονέα του. Γίνεται η σύμβαση ότι το πρώτο (δηλαδή, το αριστερότερο) παιδί αντιστοιχεί στο βορειοδυτικό τεταρτημόριο, το δεύτερο στο βορειοανατολικό τεταρτημόριο, το τρίτο στο νοτιοδυτικό τεταρτημόριο και το τέταρτο (δηλαδή, το δεξιότερο) παιδί στο νοτιοανατολικό τεταρτημόριο. Για παράδειγμα, τα ομογενή τμήματα της εικόνας του Σχήματος 14.6α φαίνονται στο Σχήμα 14.6β, ενώ το Σχήμα 14.6γ παριστά το αντίστοιχο τετραδικό δένδρο.



Σχήμα 14.6: Παράδειγμα (α) εικόνας, (β) χωρισμού της σε τμήματα, και (γ) αντίστοιχου τετραδικού δένδρου.

Η προηγούμενη ανάπτυξη σχετικά με τα τετραδικά δένδρα περιοχών θεωρεί ότι η δομή είναι αποθηκευμένη στη μνήμη και επομένως μπορεί να υλοποιηθεί με τη βοήθεια δεικτών. Ωστόσο, μία τέτοια υλοποίηση δεν μπορεί να χρησιμοποιηθεί σε συστήματα βάσεων δεδομένων, όπου το μέγεθος και το πλήθος των εικόνων απαιτούν τη χρήση δευτερεύουσας αποθήκευσης. Έτσι προτάθηκε η δομή των Γραμμικών τετραδικών δένδρων περιοχών (Linear region quadtrees) από την Gargantini (1982). Ο όρος 'γραμμικό' προέκυψε από το γεγονός ότι ένα σύνολο γεωμετρικών δεδομένων του δισδιάστατου χώρου (ή και χώρου περισσότερων διαστάσεων) μετατρέπεται σε μία λίστα ακεραίων τιμών, δηλαδή τιμών που μπορούν να παρασταθούν επί μίας ευθείας γραμμής.

Η γραμμική αναπαράσταση, λοιπόν, για το τετραδικό δένδρο συνίσταται σε ένα σύνολο ακεραίων τιμών, όπου υπάρχει μία τιμή για κάθε μαύρο κόμβο της υλοποίησης του αντίστοιχου δένδρου που βασίζεται στους δείκτες. Η τιμή για έναν κόμβο είναι μία διεύθυνση που περιγράφει τη θέση και το μέγεθος του σχετιζόμενου τμήματος της εικόνας. Οι πιο δημοφιλείς γραμμικές υλοποιήσεις είναι τρεις: οι FL (Fixed Length), VL (Variable Length) και FD (Fixed length – Depth) γραμμικές υλοποιήσεις. Στις επόμενες παραγράφους περιγράφουμε σύντομα την κάθε μία.

Στην FL γραμμική υλοποίηση, η διεύθυνση ενός μαύρου κόμβου είναι μία κωδικολέξη (codeword), που αποτελείται από n ψηφία βάσης 5. Οι πρώτοι 4 διαφορετικοί κώδικες εκφράζουν μία από τις διαφορετικές κατευθύνσεις που μπορούν να ακολουθηθούν από έναν κόμβο σε κάποιο από τα παιδιά του. Ο τελευταίος κώδικας εκφράζει την 'αδιάφορη' (don't care) κατεύθυνση. Για ένα μαύρο κόμβο που βρίσκεται στο επίπεδο i τα πρώτα $n-i$ ψηφία της κωδικολέξης εκφράζουν τις κατευθύνσεις που απαρτίζουν το μονοπάτι από τη ρίζα μέχρι αυτό τον κόμβο, ενώ τα τελευταία i ψηφία είναι όλα ίσα με την αδιάφορη κατεύθυνση. Έτσι, από μία κωδικολέξη ενός μαύρου κόμβου συνάγεται και το επίπεδο αλλά και η θέση του στο αντίστοιχο δένδρο.

Στην VL γραμμική υλοποίηση, η διεύθυνση ενός μαύρου κόμβου είναι μία κωδικολέξη που αποτελείται από το πολύ n ψηφία βάσης 5. Ο κώδικας 0 δε χρησιμοποιείται στις κωδικολέξεις, ενώ οι υπόλοιποι 4 διαφορετικοί κώδικες εκφράζουν από μία τις διαφορετικές κατευθύνσεις που μπορούν να ακολουθηθούν από έναν κόμβο προς κάποιο από τα παιδιά του. Για ένα μαύρο κόμβο που βρίσκεται στο επίπεδο i μία κωδικολέξη έχει $n-i$ ψηφία που εκφράζουν τις κατευθύνσεις που απαρτίζουν το μονοπάτι από τη ρίζα μέχρι αυτό τον κόμβο. Το επίπεδο του κόμβου μπορεί να το βρεθεί ανακαλύπτοντας ποιά είναι η μικρότερη δύναμη του 5 που όταν διαιρέσουμε (με αχέραια διαίρεση) την κωδικολέξη με αυτή μας δίνει πηλίκο 0.

Οι FL και VL κωδικολέξεις είναι αριθμοί στο πενταδικό σύστημα, ενώ θα ήταν αποδοτικότερο να ήταν αριθμοί στο τετραδικό σύστημα αρίθμησης, ώστε η αποκωδικοποίησή τους να χρησιμοποιεί πράξεις ολίσθησης αντί για πράξεις αχέραιας διαίρεσης. Για το σκοπό αυτό χρησιμοποιείται περισσότερο η FD γραμμική υλοποίηση. Σύμφωνα με τη μέθοδο αυτή, η διεύθυνση ενός μαύρου κόμβου έχει δύο μέρη. Το πρώτο μέρος είναι μία κωδικολέξη που αποτελείται από n ψηφία βάσης 4. Οι 4 διαφορετικοί κώδικες εκφράζουν από μία τις διαφορετικές κατευθύνσεις που μπορούν να ακολουθηθούν από έναν κόμβο προς κάποιο από τα παιδιά του. Για ένα μαύρο κόμβο που βρίσκεται στο επίπεδο i τα πρώτα $n-i$ ψηφία μίας κωδικολέξης εκφράζουν τις κατευθύνσεις που απαρτίζουν το μονοπάτι από τη ρίζα μέχρι αυτό τον κόμβο, ενώ τα τελευταία i ψηφία είναι όλα ίσα με κάποια αυθαίρετη τιμή (για παράδειγμα, 0). Το δεύτερο μέρος της διεύθυνσης δηλώνει το επίπεδο του μαύρου κόμβου. Με άλλα λόγια, δείχνει τον αριθμό των ψηφίων του πρώτου μέρους που εκφράζουν το μονοπάτι από τη ρίζα μέχρι τον κόμβο αυτό. Έτσι, από το δεύτερο μέρος μίας κωδικολέξης γίνεται γνωστό το επίπεδο του μαύρου κόμβου που αυτή εκφράζει, και από το πρώτο μέρος η θέση αυτού του κόμβου στο δένδρο.

	A	B	Γ	Δ	E	Z
Υλοποίηση FL	004 ₅ 4 ₁₀	030 ₅ 15 ₁₀	033 ₅ 18 ₁₀	322 ₅ 87 ₁₀	323 ₅ 88 ₁₀	334 ₅ 94 ₁₀
Υλοποίηση VL	11 ₅ 6 ₁₀	141 ₅ 46 ₁₀	441 ₅ 121 ₁₀	334 ₅ 94 ₁₀	434 ₅ 119 ₁₀	44 ₅ 24 ₁₀
Υλοποίηση FD	0001 ₄ 1 ₁₀	0301 ₄ 48 ₁₀	0330 ₄ 60 ₁₀	3220 ₄ 232 ₁₀	3230 ₄ 236 ₁₀	3301 ₄ 241 ₁₀

Πίνακας 14.1: Κωδικολέξεις μαύρων κόμβων Σχήματος 14.6.

Ο Πίνακας 14.1 δίνει τις τιμές των κωδικολέξεων για τους μαύρους κόμβους του δένδρου του Σχήματος 14.6 σύμφωνα και με τις τρεις γραμμικές υλοποιήσεις. Οι τιμές αυτές, είτε βάσης 4 είτε βάσης 5, ισοδυναμούν με τιμές βάσης 10, οπότε η διαχείρισή τους αποτελεί πλέον εύκολη υπόθεση καθώς μπορούν να αποθηκευθούν σε μία αποδοτική δομή δευτερεύουσας μνήμης (για παράδειγμα, σε μία δομή της οικογένειας των B-δένδρων). Έτσι, στην ουσία ένα Γραμμικό τετραδικό δένδρο είναι ένα B⁺-δένδρο με όλα τα θετικά επακόλουθα της δομής αυτής. Το σημαντικότερο γεγονός είναι ότι η δομή του B⁺-δένδρου στηρίζει κάθε σύστημα διαχείρισης βάσεων δεδομένων, οπότε ένα σύστημα εύκολα μπορεί να διαχειριστεί εικόνες χωρίς ιδιαίτερες επεκτάσεις. Έτσι, τα Γραμμικά τετραδικά δένδρα είναι περισσότερο δημοφιλή σε σχέση με τα R-δένδρα για υλοποίηση στα εμπορικά συστήματα διαχείρισης βάσεων δεδομένων. Εξ άλλου, οι υλοποιήσεις του R-δένδρου στα διάφορα εμπορικά συστήματα δεν ταυτίζονται με την ανάπτυξη του προηγούμενου υποκεφαλαίου, αλλά στην ουσία τα ορθογώνια του R-δένδρου εισάγονται στους συνήθεις πίνακες του σχεσιακού μοντέλου και αποθηκεύονται πάλι ως B⁺-δένδρα. Ωστόσο, αυτή η απεικόνιση είναι μη αποτελεσματική γιατί κατά την αποθήκευση στους πίνακες χάνονται τα χωροταξικά χαρακτηριστικά των δεδομένων και απαιτούνται ιδιαίτεροι αλγόριθμοι χειρισμού τους.

Οι απαιτήσεις χώρου για το Γραμμικό τετραδικό δένδρο εξαρτώνται από τον αριθμό των εξωτερικών μαύρων κόμβων και από το μήκος που έχουν οι τιμές-διευθύνσεις. Γενικά, η γραμμική αναπαράσταση είναι οικονομική ως προς τον απαιτούμενο χώρο, όμως δεν είναι κατάλληλη για πολλούς χρήσιμους αλγόριθμους που είναι αποτελεσματικοί σε υλοποιήσεις κύριας μνήμης (δηλαδή, με δείκτες). Με λίγα λόγια, σε μία ερώτηση διαστήματος της μορφής: 'Είναι η περιοχή τάδε μαύρη;', πρέπει το συγκεκριμένο παράθυρο να σπάσει σε υποπαράθυρα ακολουθώντας τη λογική της διάσπασης σε τεταρ-

τημόρια, για κάθε τεταρτημόριο να βρεθεί η αντίστοιχη κωδικολέξη και να γίνει ανεξάρτητη αναζήτηση στο B^+ -δένδρο. Εναλλακτικά, μπορεί να βρεθεί η μικρότερη και η μεγαλύτερη κωδικολέξη και να εκτελεσθεί μία ερώτηση διαστήματος στο B^+ -δένδρο. Ωστόσο, στη γραμμική υλοποίηση δεν εύκολο να απαντηθούν ερωτήσεις πλησιέστερου γείτονα του τύπου: 'Ποιά είναι η πλησιέστερη περιοχή προς την περιοχή τάδε;', ή τοπολογικές ερωτήσεις του τύπου: 'Ποιές περιοχές βρίσκονται προς τα δυτικά της περιοχής τάδε;', οι οποίες εύκολα ικανοποιούνται σε τετραδικά δένδρα χύριας μνήμης.

Για στερεά σώματα τριών διαστάσεων μπορεί να χρησιμοποιηθεί μία συγγενής δομή που ονομάζεται **Γραμμικό Οκταδικό Δέντρο Περιοχών** (Linear Region Octree). Η δομή αυτή βασίζεται στην αναδρομική υποδιαίρεση του τρισδιάστατου πίνακα σε ίσα οκτημόρια μέχρι να προκύψουν ομογενείς υπο-κύβοι, που ονομάζονται voxel. Περισσότερες λεπτομέρειες για τη δομή αυτή αλλά και για πλήθος άλλων παραλλαγών του τετραδικού δένδρου, ο αναγνώστης μπορεί να βρει στα βιβλία του Samet.

14.4 Χρονικά διασπώμενο B-δένδρο

Στις διαχρονικές βάσεις δεδομένων τα δεδομένα δεν διαγράφονται φυσικά αλλά λογικά, ώστε να είναι δυνατή η υποστήριξη ερωτήσεων που αναφέρονται σε δεδομένα του παρελθόντος αλλά και του μέλλοντος. Σε μία εγγραφή μπορεί να υπάρχουν ένα ή περισσότερα πεδία με τιμές που μεταβάλλονται με το χρόνο, και μάλιστα μεταβάλλονται σε χρονικές στιγμές ανεξάρτητες μεταξύ τους. Έτσι για μία εγγραφή με δεδομένο κύριο κλειδί μπορεί να υπάρχουν πολλές **εκδοχές** (versions), και επομένως θα πρέπει η αντίστοιχη εγγραφή ή το πεδίο να συνοδεύεται από αντίστοιχες πληροφορίες για τη χρονική ισχύ των δεδομένων. Αν οι χρονικές αυτές πληροφορίες αποθηκεύονται σε επίπεδο εγγραφής, τότε λέγεται ότι εφαρμόζεται η τεχνική της **εκδοχής πλειάδων** (tuple versioning), ενώ αν οι πληροφορίες αυτές αποθηκεύονται σε επίπεδο πεδίου, τότε λέγεται ότι εφαρμόζεται η τεχνική της **εκδοχής χαρακτηριστικών** (attribute versioning). Στη συνέχεια θα θεωρηθεί η μέθοδος της εκδοχής πλειάδων, που είναι απλούστερη ως προς τη φυσική υλοποίησή της. Με απλά λόγια, σε κάθε εγγραφή υπάρχουν δύο επιπλέον πεδία που δηλώνουν την αρχή και το τέλος έναρξης της ισχύος της εγγραφής. Ο όρος **ενεργό** (active) ή **ζωντανό** (alive) κλειδί ή εγγραφή δηλώνουν το κλειδί ή την εγγραφή που ισχύουν την παρούσα χρονική στιγμή.

Ένα δεύτερο χαρακτηριστικό των διαχρονικών βάσεων δεδομένων είναι οι τεράστιες απαιτήσεις τους σε αποθηκευτικό χώρο. Μέχρι πρόσφατα οι μαγνητικοί δίσκοι δεν διακρίνονταν για τις αποθηκευτικές τους δυνατότητες σε σχέση με τους οπτικούς δίσκους. Μάλιστα, ο οπτικός δίσκος με τις μεγαλύτερες δυνατότητες ήταν ο δίσκος WORM, δηλαδή «μίας αποθήκευσης, πολλών αναγνώσεων» (δες Κεφάλαιο 2.4). Λόγω της ιδιαιτερότητάς του, το μέσο αυτό δεν προσφέρεται για υλοποιήσεις των γνωστών δομών, γιατί λόγω της δυναμικότητας των δομών μία εισαγωγή ή μία διαγραφή θα προκαλέσει αλυσιδωτές αλλαγές σε ένα ολόκληρο μονοπάτι της δομής, που θα είναι απαραίτητο να αντιγραφεί σε νέα περιοχή του δίσκου. Έτσι, πολύ γρήγορα η δομή θα δεσμεύσει τεράστιες περιοχές του δίσκου.

Οι πρώτες, λοιπόν, δομές που παρουσιάστηκαν για χρήση σε διαχρονικές βάσεις δεδομένων βασίζονταν στην ύπαρξη οπτικών δίσκων WORM. Τώρα πλέον οι μαγνητικοί δίσκοι έχουν μεγάλες δυνατότητες και θα έχουν στο μέλλον ακόμη μεγαλύτερες. Επομένως, οι προηγούμενοι περιορισμοί δεν ισχύουν πλέον. Συνεπώς, οι δομές που θα εξετασθούν στη συνέχεια (το Χρονικά Διασπώμενο Β-δένδρο και ο πρόγονός του, το Β-δένδρο Μίας Αποθήκευσης), μπορεί να είναι σχεδιασμένες κάτω από άλλες συνθήκες, όμως δεν παύουν να είναι χρήσιμες ακόμη και σήμερα.

Μεταξύ του Β-δένδρου Μίας Αποθήκευσης (WOBT, Write-once B-tree), που προτάθηκε από τον Easton της IBM (1986) και του κλασικού Β⁺-δένδρου υπάρχουν ομοιότητες όπως:

- οι εγγραφές αποθηκεύονται στα φύλλα,
- το μικρότερο κλειδί κάθε φύλλου, που λέγεται **επικεφαλίδα** (header), ανέρχεται στο ανώτερο επίπεδο με ένα δείκτη προς τη συγκεκριμένη σελίδα. Έτσι, τα επίπεδα της δομής επάνω από τα φύλλα αποτελούν τον κατάλογο.

Η σημαντική διαφορά είναι ότι (λόγω της ιδιαιτερότητας του δίσκου WORM) οι εγγραφές και τα κλειδιά μπορούν να αποθηκευθούν στους κόμβους αταξινόμητες.

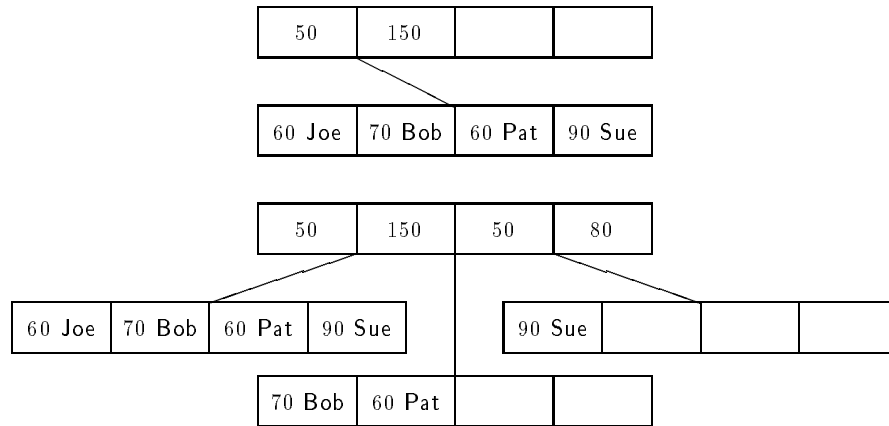
Η αναζήτηση μίας εγγραφής αρχίζει από τη ρίζα. Αναζητείται το μεγαλύτερο ενεργό κλειδί που δεν ξεπερνά τη συγκεκριμένη τιμή του αναζητούμενου κλειδιού και ακολουθείται ο δείκτης αυτού του κλειδιού προς το αντίστοιχο παιδί. Η διαδικασία επαναλαμβάνεται μέχρι τη σελίδα δεδομένων, όπου αναζητείται η ενεργός εγγραφή με το κλειδί αυτό. Αν δεν βρεθεί, τότε η αναζήτηση είναι ανεπιτυχής.

Για τη διαγραφή μίας εγγραφής, προσπελάζεται η κατάλληλη σελίδα δεδομένων με παρόμοιο τρόπο. Αν η εγγραφή υπάρχει, τότε μία νέα εγγραφή αποθηκεύεται στη σελίδα αυτή με το ίδιο κύριο κλειδί, αλλά με μία ένδειξη ότι η εγγραφή δεν είναι ενεργή. Προφανώς, στην περίπτωση της διαγραφής δεν υπάρχει δυνατότητα συγχώνευσης, όπως στην οικογένεια των κλασικών B-δένδρων.

Κατά την εισαγωγή εγγραφής αν υπάρχει ελεύθερος χώρος στη σελίδα, τότε η διαδικασία είναι εύκολη. Αν όχι, τότε γίνεται 'διάσπαση', οπότε (α) ο πλήρης κόμβος καθίσταται ιστορικός, (β) οι εγγραφές ταξινομούνται αγνοώντας τις ανενεργές εγγραφές και (γ) επανα-αποθηκεύονται σε μία ή δύο νέες σελίδες. Ο αριθμός των νέων σελίδων που προκύπτουν εξαρτάται από δύο παραμέτρους $TD > 1$ και $TI > 1$ που αναφέρονται στις σελίδες δεδομένων και στις σελίδες καταλόγου, αντίστοιχα. Αν η διασπώμενη σελίδα περιέχει (όχι) λιγότερο από TD ενεργές εγγραφές ή TI ενεργά κλειδιά, τότε δημιουργείται μία νέα σελίδα (αντίστοιχα, δύο νέες σελίδες). Κάθε σελίδα δέχεται ίσο αριθμό ταξινομημένων εγγραφών ή κλειδιών. Ένα νέο ζεύγος επικεφαλίδα-δείκτης για κάθε μία από τις νέες σελίδες ανέρχεται στο ανώτερο επίπεδο. Η διάσπαση της ρίζας γίνεται κατά τον ίδιο τρόπο. Αν από τη ρίζα προκύψει μία μόνο νέα σελίδα, τότε αυτή γίνεται η νέα ρίζα, ενώ αν προκύψουν δύο νέες σελίδες, τότε το δένδρο πρέπει να μεγαλώσει κατά ένα επίπεδο. Κατά τη διάσπαση γενικά δεν αντιγράφονται οι ανενεργές εγγραφές εκτός των περιπτώσεων που (α) ανενεργός εγγραφή είναι η επικεφαλίδα, και (β) ανενεργός εγγραφή είναι η τελευταία εγγραφή της σελίδας. Με τη διαδικασία αυτή περιορίζονται οι άχρηστες εγγραφές από το μονοπάτι αναζήτησης και ελαχιστοποιείται το βάθος του δένδρου.

Στο Σχήμα 14.7 παρουσιάζεται η διαδικασία της εισαγωγής και της διάσπασης σε δομή WOBT. Αρχικά η δομή αποτελείται από ένα πλήρες φύλλο με τέσσερις εγγραφές, ενώ η μία από αυτές είναι ανενεργή ('60 Joe'). Λόγω εισαγωγής της εγγραφής '80 Jim' πρέπει να γίνει διάσπαση, οπότε ο κόμβος μετατρέπεται σε ιστορικό. Οι τέσσερις ενεργές εγγραφές ταξινομούνται ως προς το κλειδί τους και έτσι προκύπτουν δύο σελίδες με ίσο αριθμό εγγραφών (από δύο). Από τις δύο σελίδες ανέρχονται οι κατάλληλες επικεφαλίδες στο ανώτερο επίπεδο. Ας σημειωθεί ότι στο σχήμα αυτό δεν παρουσιάζονται οι **χρονοσφραγίδες** (timestamps), που αποτελούν τη βάση για αναζήτηση δεδομένων παρελθόντων χρονικών στιγμών.

Το **Χρονικά Διασπώμενο B-δένδρο** (TSBT, Time-split B-tree), που προτάθηκε από το Lomet (της DEC) και τη Salzberg (1989), είναι μία βελ-



Σχήμα 14.7: Εισαγωγή εγγραφής '80 Jim' και διάσπαση σε δομή WOBT.

τιωμένη παραλλαγή της δομής WOBT. Σύμφωνα με τη νέα δομή, προτείνεται η αποθήκευση των τρεχόντων και ιστορικών δεδομένων σε μαγνητικό και σε δίσκο WORM αντίστοιχα. Δηλαδή, προτιμάται για τα τρέχοντα δεδομένα μία γρηγορότερη και ακριβότερη συσκευή, ενώ για τα ιστορικά δεδομένα προτιμάται μία αργότερη και φθηνότερη συσκευή. Ωστόσο, με την εξέλιξη του υλικού δεν συντρέχουν πλέον οι λόγοι για τη θεώρηση αυτή, και επομένως ολόκληρη η δομή μπορεί να αποθηκευθεί σε μαγνητικό δίσκο.

Η αναζήτηση και η διαγραφή είναι ακριβώς η ίδια όπως στη δομή WOBT, όμως η διάσπαση είναι διαφορετική και επιτυγχάνει τη μείωση των περιττών επαναλαμβανόμενων δεδομένων. Κατ' αρχήν, η πρώτη διαφορά είναι ότι η διάσπαση αφορά μόνο στους κόμβους που βρίσκονται στο μαγνητικό δίσκο και μπορεί να έχει δύο μορφές, δηλαδή:

- **διάσπαση με βάση το κλειδί**, η οποία είναι παρόμοια με τη διάσπαση που περιγράφηκε στο κλασικό B-δένδρο (δες Κεφάλαιο 8.2), ή
- **διάσπαση με βάση το χρόνο**, όπου το χρονικό κατώφλι (time threshold) δεν είναι απαραίτητα η τρέχουσα χρονική στιγμή.

Ειδικότερα, κατά τη διάσπαση με βάση το χρόνο, κάθε εκδοχή μίας εγγραφής που ισχύει πριν (κατά και μετά) το χρονικό κατώφλι πρέπει να βρίσκεται στον ιστορικό (αντίστοιχα, στο νέο) κόμβο. Έτσι με τη διάσπαση οι νεότερες παραμένουν στο μαγνητικό δίσκο, ενώ οι παλαιότερες εγγραφές μεταφέρονται στον οπτικό δίσκο και τοποθετούνται στο τέλος του κατειλημμένου χώρου επιτυγχάνοντας υψηλή τιμή του παράγοντα χρησιμοποίησης χώρου.

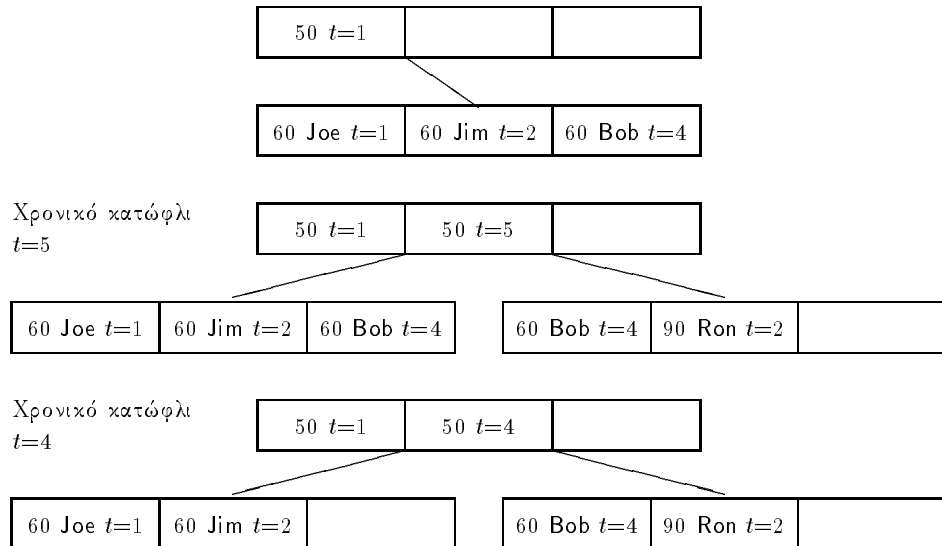
Οι κόμβοι του καταλόγου διασπώνται μόνο με βάση το κλειδί. Η τιμή διάσπασης και ένα αντίγραφο του χρόνου που χρησιμοποιήθηκε για προηγούμενες αναφορές στον υπό διάσπαση κόμβο ανέρχονται στον κόμβο-πατέρα. Αναφορές σε πεδία κλειδιών όπου το άνω όριο είναι μικρότερο ή ίσο της τιμής διάσπασης μεταφέρονται στο νέο αριστερό κόμβο, ενώ αναφορές σε πεδία κλειδιών όπου το κάτω όριο είναι μεγαλύτερο ή ίσο της τιμής διάσπασης μεταφέρονται στο νέο δεξιό κόμβο. Τέλος, όλα τα άλλα δεδομένα που αναφέρονται στο ιστορικό μέρος της δομής αντιγράφονται και στους δύο κόμβους.

Αν προέχει η ελαχιστοποίηση του συνολικού χώρου, τότε επιλέγεται η διάσπαση με βάση το κλειδί, ενώ αν προέχει η ελαχιστοποίηση του χώρου του ακριβότερου μαγνητικού δίσκου, τότε επιλέγεται η διάσπαση με βάση το χρόνο. Το είδος της διάσπασης εξαρτάται επίσης και από τα περιεχόμενα του κόμβου. Αν ένας πλήρης κόμβος περιέχει μόνο ενεργές εγγραφές, τότε δεν υπάρχει λόγος να γίνει διάσπαση με βάση το χρόνο, επειδή όλα τα δεδομένα θα πρέπει να παραμείνουν στον τρέχοντα κόμβο. Αντίθετα, αν στον πλήρη κόμβο έχουν συμβεί μόνον ενημερώσεις της ίδιας εγγραφής δεν υπάρχει λόγος διάσπασης με βάση το κλειδί. Η επιλογή του κατωφλίου διάσπασης έχει άμεση σχέση με την ποσότητα των επαναλαμβανόμενων δεδομένων. Αν μία εγγραφή ισχύει πριν και μετά το χρονικό αυτό σημείο, τότε αποθηκεύεται και στα δύο μέσα. Το πλεονέκτημα της επανάληψης μερικών δεδομένων είναι η βελτιωμένη επίδοση κατά την αναζήτηση.

Στο Σχήμα 14.8 παρουσιάζονται οι δύο διαφορετικοί τρόποι διάσπασης σε μία δομή TSBT. Στην αρχική μορφή του δένδρου εισάγεται η εγγραφή '90 Ron' κατά τη χρονική στιγμή $t=5$. Η διάσπαση μπορεί να γίνει είτε με βάση την τρέχουσα χρονική στιγμή ($t=5$) και επανάληψη δεδομένων ('60 Bob') όπως στο δεύτερο υποσχήμα, είτε με βάση προηγούμενη χρονική στιγμή ($t=4$) και χωρίς επανάληψη δεδομένων όπως συμβαίνει στο τρίτο υποσχήμα.

Οι ερωτήσεις που υποβάλλονται σε μία διαχρονική βάση δεδομένων είναι τριών ειδών:

- **ερωτήσεις χρονικού διαστήματος (time-interval queries):** 'Ποιοί υπάλληλοι εργάζονταν στην επιχείρηση από 1/1/90 μέχρι 31/12/99;'. Υποπερίπτωση είναι η **ερώτηση χρονικής στιγμής (time-slice query):** 'Ποιοί υπάλληλοι εργάζονταν στην επιχείρηση την 1/1/2000;'.



Σχήμα 14.8: Εισαγωγή εγγραφής '90 Ron' και διάσπαση με βάση το χρόνο.

- **ερωτήσεις διαστήματος σε χρονικό διάστημα** (range time-interval queries): 'Ποιοί υπάλληλοι με ονόματα από Κόλλιας μέχρι Μανωλόπουλος εργάζονταν στην επιχείρηση από 1/1/90 μέχρι 31/12/99;'. Υποπερίπτωση είναι η ερώτηση διαστήματος σε χρονική στιγμή (range time-slice query): 'Ποιοί υπάλληλοι με ονόματα από Κόλλιας μέχρι Μανωλόπουλος εργάζονταν στην επιχείρηση την 1/1/2000;'.
- **αγνές ερωτήσεις κλειδιών** (pure key queries): 'Σε ποιά τμήματα εργάστηκε ο υπάλληλος τάδε;'.

Από την προηγούμενη περιγραφή προκύπτει ότι οι δομές WOBT και TSBT δεν μπορούν να απαντήσουν μία αγνή ερώτηση κλειδιού (παρά μόνο σαρώνοντας όλη τη δομή), ενώ για τις άλλες δύο κατηγορίες οι δομές αυτές έχουν προβληματική επίδοση, καθώς τα δεδομένα των παρελθουσών χρονικών στιγμών είναι αποθηκευμένα στις ίδιες σελίδες. Έτσι, μονοπάτια από τη ρίζα προς τα φύλλα είναι μεγαλύτερα σε σχέση με την περίπτωση των δομών που περιέχουν μόνο δεδομένα μίας χρονικής στιγμής. Ερωτήσεις για ενεργά δεδομένα που είναι απομονωμένα από τα μη ενεργά απαντώνται με ικανοποιητική επίδοση.

14.5 Χρονικός κατάλογος

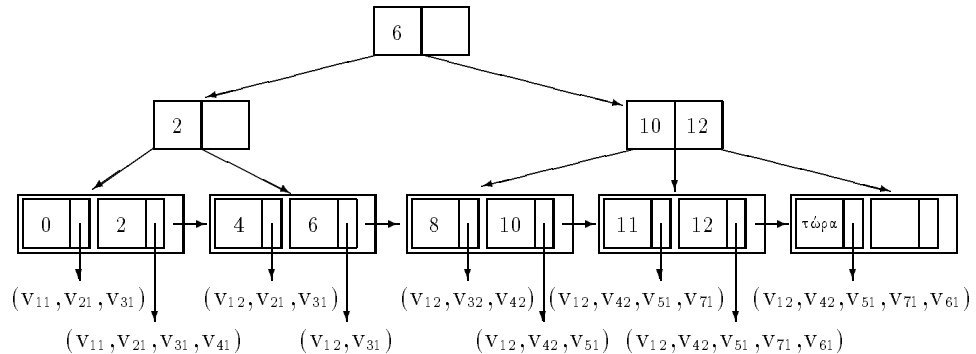
Ο Χρονικός κατάλογος (Time index) προτάθηκε από τον Elmasri και άλλους το 1990 και είναι μία δομή που στηρίζεται στο B^+ -δένδρο. Πιο συγκεκριμένα, ο κατάλογος δεικτοδοτεί όλες τις τιμές του χρόνου όπου έχει συμβεί μία εισαγωγή, μία διαγραφή ή μία ενημέρωση εγγραφής, δηλαδή ο κατάλογος καταγράφει κάθε χρονική στιγμή όπου υπήρξε κάποια αλλαγή στη βάση δεδομένων.

Υπάλληλος	Τμήμα	Χρονικό διάστημα	σύμβολο εκδοχής
Υπαλ.1	A	[0,3]	V ₁₁
Υπαλ.1	B	[4,τώρα]	V ₁₂
Υπαλ.2	B	[0,5]	V ₂₁
Υπαλ.3	Γ	[0,7]	V ₃₁
Υπαλ.3	A	[8,9]	V ₃₂
Υπαλ.4	Γ	[2,3]	V ₄₁
Υπαλ.4	A	[8,τώρα]	V ₄₂
Υπαλ.5	B	[10,τώρα]	V ₅₁
Υπαλ.6	Γ	[12,τώρα]	V ₆₁
Υπαλ.7	Γ	[11,τώρα]	V ₇₁

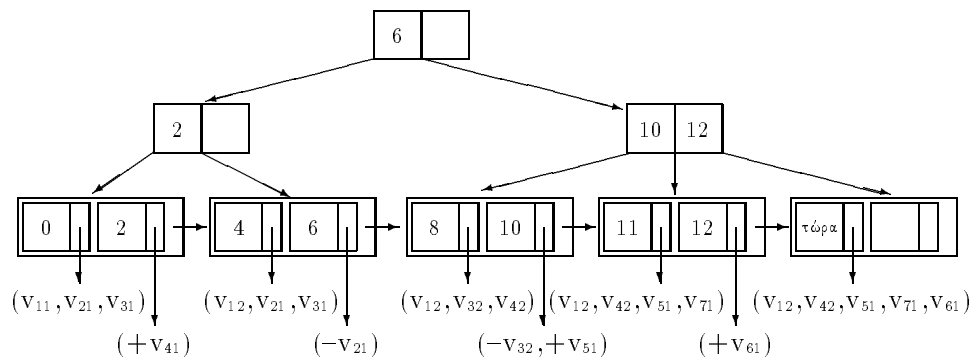
Πίνακας 14.2: Απασχόληση υπαλλήλων σε τμήματα επιχείρησης.

Για παράδειγμα, έστωσαν τα δεδομένα του Πίνακα 14.2 όπου φαίνονται τα χρονικά διαστήματα απασχόλησης κάποιων υπαλλήλων στα τμήματα μίας επιχείρησης (οι τιμές του χρόνου είναι ακέραιες τιμές, και τα διαστήματα είναι κλειστά). Οι χρονικές στιγμές που συνέβησαν κάποιες αλλαγές είναι 0, 2, 4, 6, 8, 10, 11 και 12, αλλά επίσης υπάρχει και η χρονική στιγμή 'τώρα'. Στο Σχήμα 14.9 παρουσιάζεται ένα B^+ -δένδρο τάξης 3 με τα δεδομένα του Πίνακα 14.2. Στα φύλλα του δένδρου οι εγγραφές είναι της μορφής (t,P), όπου t είναι μία χρονική στιγμή, ενώ P είναι ένας δείκτης προς μία σελίδα που περιέχει επίσης δείκτες προς το αρχείο με τα πραγματικά δεδομένα (δηλαδή, τις εκδοχές v_{11} μέχρι v_{71}).

Ωστόσο, στο Σχήμα 14.9 φαίνεται ότι υπάρχουν περιττές επαναλήψεις δεικτών. Για παράδειγμα, για την εκδοχή v_{12} υπάρχουν επτά διαφορετικοί δείκτες. Για το λόγο αυτό, ο Χρονικός κατάλογος ακολουθεί μία άλλη τεχνική οργάνωσης των δεικτών, ώστε να αποφευχθεί η σπατάλη του χώρου.

Σχήμα 14.9: B⁺-δένδρο με τα δεδομένα του Πίνακα 14.2.

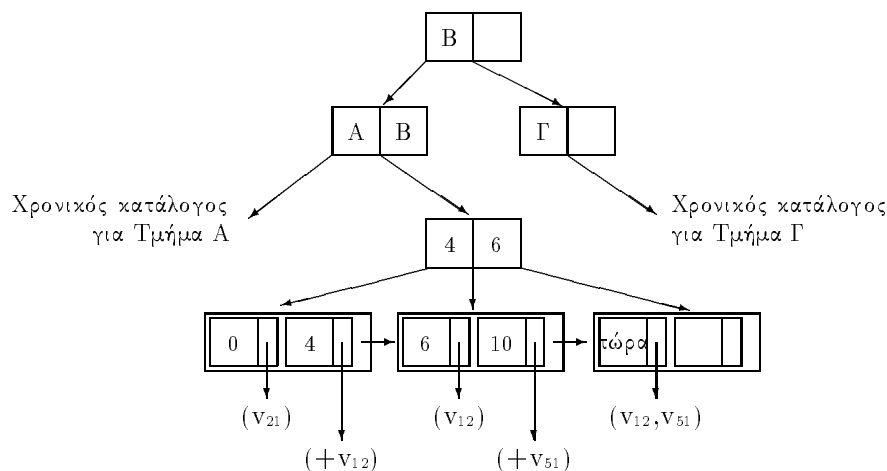
Πιο συγκεκριμένα, το πρώτο ζεύγος κάθε φύλλου, (t, P) , που ονομάζεται **οδηγούσα είσοδος** (leading entry), δείχνει προς μία σελίδα με δείκτες προς όλες τις ενεργές εγγραφές για τη χρονική στιγμή t , ενώ οι δείκτες των υπόλοιπων ζευγών κάθε φύλλου δείχνουν προς σελίδες όπου κρατούνται οι αλλαγές σε σχέση με τους δείκτες του πρώτου ζεύγους του φύλλου. Η οργάνωση του Χρονικού καταλόγου παρουσιάζεται στο Σχήμα 14.10. Όμως και πάλι θα υπάρχει σπατάλη χώρου, αν οι εγγραφές ισχύουν για μεγάλες χρονικές περιόδους. Ωστόσο, πλεονέκτημα της μεθόδου είναι ότι καθώς οι τιμές του χρόνου t αυξάνονται μονοτονικά, οι εισαγωγές στο Χρονικό κατάλογο εντοπίζονται στο τελευταίο φύλλο του.



Σχήμα 14.10: Χρονικός κατάλογος με τα δεδομένα του Πίνακα 14.2.

Ο Χρονικός κατάλογος είναι ιδιαίτερα χρήσιμος για την ικανοποίηση ερωτήσεων διαστήματος ως προς το χρόνο. Ένα τέτοιο παράδειγμα είναι

η ερώτηση: 'Ποιοί υπάλληλοι εργάζονταν στην επιχείρηση κατά το χρονικό διάστημα $[3,6]$;' . Στην περίπτωση αυτή πρέπει να εκτελεσθεί μία ερώτηση διαστήματος στο Χρονικό κατάλογο με βάση ένα διάστημα, που να αρχίζει από τη χρονική στιγμή $t=3$ ή τη χρονική στιγμή που είναι αμέσως μικρότερη του 3, μέχρι τη χρονική στιγμή $t=6$ ή τη χρονική στιγμή που είναι αμέσως μικρότερη του 6. Με τον τρόπο αυτό θα απομονωθούν οι σελίδες με τους κατάλληλους δείκτες προς το κύριο αρχείο των δεδομένων



Σχήμα 14.11: Χρονικός κατάλογος δύο επιπέδων.

Ωστόσο, αν τεθεί μία ερώτηση διαστήματος σε χρονικό διάστημα, όπως: 'Ποιοί ήταν οι υπάλληλοι του Τμήματος B κατά το χρονικό διάστημα $[3,6]$;', τότε θα πρέπει να εκτελεσθεί η ίδια διαδικασία και να γίνει ένας χρονοβόρος τελικός έλεγχος αν οι υπάλληλοι πράγματι ανήκαν στο Τμήμα B. Αυτό σημαίνει περιττή επιβάρυνση στο σύστημα. Για το λόγο αυτό, ο Χρονικός κατάλογος μπορεί να οργανωθεί σε επίπεδα. Στο Σχήμα 14.11 επάνω από το γνωστό Χρονικό κατάλογο τίθεται ένα άλλο B^+ -δένδρο, ώστε να απομονωθούν τα δεδομένα των τμημάτων. Είναι ευνόητο ότι ο αριθμός των επιπέδων δεν συμφέρει να είναι μεγαλύτερος από δύο, επειδή αν πιθανώς υπήρχε μία αλληλουχία καταλόγων επάνω από το Χρονικό κατάλογο, τότε θα είχε ιδιαίτερη σημασία για την επίδοση το ποιο θα ήταν το χαρακτηριστικό που δεικτοδοτούνταν θα σε κάθε επίπεδο.

Η επίδοση του Χρονικού καταλόγου πάσχει επίσης στην περίπτωση της αγνής ερώτησης κλειδιού, γιατί δεν συνδέει τις διάφορες εκδοχές των εγγγραφών. Στο Κεφάλαιο 14.6 θα περιγραφεί μία δομή που υποστηρίζει τέτοιου

τύπου ερωτήσεις. Από τον Elmasri και άλλους προτάθηκε μία παραλλαγή του Χρονικού καταλόγου (1994), που ονομάζεται Time Index⁺, και έχει βελτιωμένη επίδοση τόσο σε σχέση με τον απαιτούμενο χώρο όσο και σε σχέση με τις χρονικές επιδόσεις.

Οι δομές WOBT, TSBT και ο Χρονικός κατάλογος είναι δομές για την υποστήριξη διαχρονικών βάσεων δεδομένων. Και οι τρεις αυτές δομές δανείζονται από τη φιλοσοφία της οικογένειας των B-δένδρων. Θα μπορούσε και η δομή των R-δένδρων να χρησιμοποιηθεί σε διαχρονικές βάσεις δεδομένων με βάση το εξής σκεπτικό. Αν ο άξονας x παριστά το χρόνο και ο άξονας y τις τιμές του κλειδιού, τότε κάθε εκδοχή μπορεί να θεωρηθεί ως ένα ευθύγραμμο τμήμα παράλληλο προς τον άξονα x . Η ιδέα θεωρητικά είναι κατ' αρχήν σωστή και υλοποιήσιμη (αν λυθεί το πρόβλημα της απ' άπειρο αύξησης των τιμών του χρόνου, ενώ ο χώρος σε ένα R-δένδρο είναι γνωστός και σταθερός), αλλά η επίδοση της δομής θα πάσχει αν οι εκδοχές έχουν μεγάλη διάρκεια ζωής. Ο λόγος είναι ότι στην περίπτωση αυτή τα ορθογώνια του R-δένδρου θα είναι μεγάλα και επικαλυπτόμενα, οπότε κατά την αναζήτηση θα ακολουθούνται πολλά μονοπάτια από τη ρίζα προς τα φύλλα. Εν πάσει περιπτώσει στη βιβλιογραφία έχουν προταθεί δομές βασισμένες στα R-δένδρα που λύνουν τα προβλήματα αυτά. Στην επισκόπηση των Salzberg και Τσότρα ο αναγνώστης θα βρεί μία συγκριτική παρουσίαση όλων των δομών που χρησιμοποιούνται σε διαχρονικές βάσεις δεδομένων.

14.6 RT-δένδρα

Προηγουμένως παρουσιάστηκαν δομές για την υποστήριξη χωροταξικών και διαχρονικών βάσεων δεδομένων. Πρόσφατα, προτάθηκαν δομές για την υποστήριξη χωροχρονικών εφαρμογών, δηλαδή εφαρμογών όπου γεωμετρικά αντικείμενα κινούνται, μεταβάλλονται ή γενικά αλλάζουν χαρακτηριστικά. Ο σχεδιασμός χωροχρονικών δομών στηρίζεται σε τρεις συντελεστές: το θέμα, τη θέση και το χρόνο. Συνήθως για να μετρηθεί ο ένας συντελεστής, ο δεύτερος μεταβάλλεται ελεγχόμενα και ο τρίτος παραμένει σταθερός. Για παράδειγμα, έστω ότι πρέπει να μελετήσουμε την αστική ανάπτυξη της Ελλάδας. Επομένως, η θέση είναι σταθερή (δηλαδή, ο γεωγραφικός χώρος της Ελλάδας), ο χρόνος ελέγχεται (λόγου χάριν, με τις χρονικές στιγμές των απογραφών) και μετράται ο πληθυσμός των διαφόρων δήμων και κοινοτήτων. Για τέτοιου είδους εφαρμογές, λοιπόν, έχουν προταθεί δομές που

στηρίζονται στην οικογένεια των B-δένδρων αλλά και στην οικογένεια των R-δένδρων. Η δομή των RT-δένδρων ανήκει στην τελευταία κατηγορία.

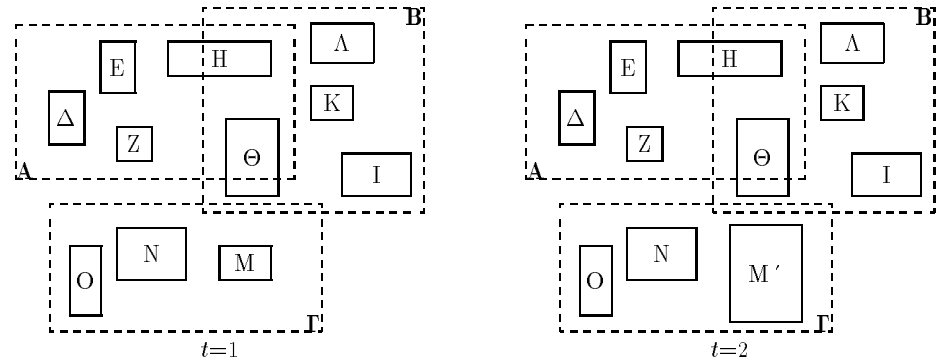
RT-δένδρο τάξης (m, M) είναι το ετερογενές δένδρο με τα εξής χαρακτηριστικά:

- κάθε εξωτερικός κόμβος περιέχει μεταξύ M και $m \leq M/2$ στοιχείων του τύπου (R, T, P) , εκτός αν είναι ρίζα. Για κάθε ζεύγος (R, T, P) , R είναι το ορθογώνιο του αντικειμένου που δεικτοδοτείται από το δείκτη P , ενώ $T = [t_{start}, t_{end}]$ είναι το χρονικό διάστημα ισχύος του.
- κάθε εσωτερικός κόμβος περιέχει μεταξύ M και $m \leq M/2$ στοιχείων τύπου (R, T, P) , εκτός αν είναι ρίζα. Για κάθε ζεύγος (R, T, P) , P είναι ένας δείκτης προς κόμβο κατωτέρου επίπεδου που περιέχει ζεύγη (R_i, T_i, P_i) , R είναι το ορθογώνιο που περιλαμβάνει όλα τα ορθογώνια R_i , ενώ T είναι το χρονικό διάστημα ισχύος του αντίστοιχου κόμβου,
- η ρίζα έχει τουλάχιστον δύο παιδιά, εκτός αν είναι φύλλο, και τέλος
- όλοι οι εξωτερικοί κόμβοι βρίσκονται στο ίδιο επίπεδο και συνδέονται με δείκτες.

Οι ομοιότητες με το R-δένδρο είναι προφανείς. Με λίγα λόγια, η ουσιαστική διαφορά είναι ότι χρονική πληροφορία συνοδεύει κάθε ορθογώνιο εσωτερικού κόμβου ή φύλλου. Ωστόσο, αυτή η 'μικρή' διαφορά καθιστά αρκετά σύνθετους τους αλγορίθμους χειρισμού των RT-δένδρου.

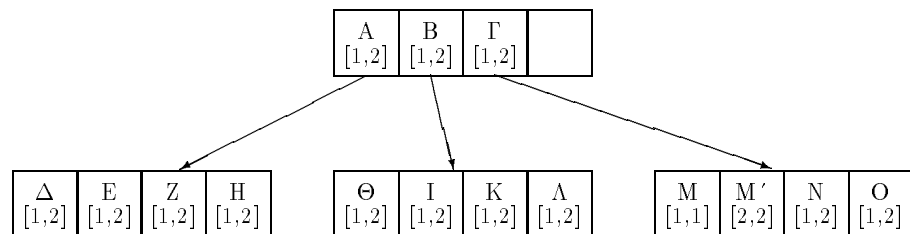
Έστω ότι κατά τη χρονική στιγμή $t=1$ δίνεται η πρώτη εικόνα που περιέχει έναν αριθμό από ορθογώνια. Το αρχικό RT-δένδρο χτίζεται χρησιμοποιώντας τους γνωστούς αλγορίθμους που ισχύουν για τα R-δένδρα. Στη συνέχεια, λοιπόν, κατά τη χρονική στιγμή $t=2$ εισάγεται η δεύτερη εικόνα, που επίσης περιέχει έναν αριθμό από ορθογώνια. Για κάθε ένα από τα νέα ορθογώνια γίνεται αναζήτηση, ώστε να εντοπισθεί ένα αντικείμενο με το ίδιο ορθογώνιο και τα ίδια χαρακτηριστικά. Αν υπάρχει ένα τέτοιο αντικείμενο, τότε απλώς επεκτείνεται το χρονικό διάστημα ισχύος του. Αν τέτοιο αντικείμενο δεν υπάρχει, τότε το νέο αντικείμενο εισάγεται στο αντίστοιχο φύλλο. Σε κάθε περίπτωση ανανεώνονται οι κόμβοι που βρίσκονται στο μονοπάτι προς τη ρίζα.

Για παράδειγμα, ας θεωρηθούν και πάλι τα δεδομένα του Σχήματος 14.3, τα οποία παρουσιάζονται στο αριστερό μέρος του Σχήματος 14.12, ως μία



Σχήμα 14.12: Δύο διαδοχικές εικόνες για $t=1$ και $t=2$.

εικόνα που εισάγεται τη χρονική στιγμή $t=1$. Στα δεξιά του σχήματος παρουσιάζεται μία εικόνα που εισάγεται τη χρονική στιγμή $t=2$. Η νέα εικόνα διαφέρει από την προηγούμενη μόνο κατά το ότι το αντικείμενο Μ έχει αλλάξει θέση και σχήμα και απεικονίζεται ως Μ', ενώ δεν έχει επέλθει καμία αλλαγή στα άλλα αντικείμενα. Στο Σχήμα 14.13 παρουσιάζεται η δομή του RT-δένδρου που θα προκύψει κατά τη χρονική στιγμή $t=2$.



Σχήμα 14.13: Δομή RT-δένδρου τη χρονική στιγμή $t=2$.

Η προηγούμενη ανάπτυξη για την εισαγωγή μίας νέας εικόνας είναι απλώς περιγραφική. Ωστόσο, απαιτείται ιδιαίτερη προσοχή κατά τη διάσχιση του δένδρου για τον εντοπισμό του κατάλληλου φύλλου. Σε σχέση με τον αλγόριθμο του R-δένδρου, η διαφορά είναι ότι κάθε φορά η επιλογή του κόμβου που πρέπει να προσπελασθεί στο κατώτερο επίπεδο εξαρτάται όχι μόνο από το υπερ-ορθογώνιο του αλλά και από το χρονικό διάστημα ισχύος του. Δηλαδή, δεν αρκεί να εξετάζεται μόνο η επικάλυψη των ορθογωνίων αλλά και η επικάλυψη των χρονικών διαστημάτων.

Ένα άλλο σημαντικό ζήτημα, που δεν θίχθηκε προηγουμένως, είναι οι διασπάσεις των κόμβων σε περίπτωση υπερχειλίσης λόγω εισαγωγής. Στην περίπτωση των Χρονικά Διασπώμενων Β-δένδρων εξετάστηκαν δύο τρόποι διάσπασης: (α) με βάση το χρόνο, και (β) με βάση το κλειδί. Για τη δομή των RT-δένδρων έχουν προταθεί τρεις μέθοδοι διάσπασης:

- **διάσπαση με βάση το χώρο**, όπου τα αντικείμενα διαχωρίζονται στους δύο νέους κόμβους έτσι ώστε τα ορθογώνια των κόμβων αυτών να έχουν την ελάχιστη δυνατή επικάλυψη (για παράδειγμα, υιοθετώντας τον αλγόριθμο διάσπασης των R-δένδρων),
- **διάσπαση με βάση το χρόνο**, όπου τα αντικείμενα διαχωρίζονται στους δύο νέους κόμβους έτσι ώστε οι χρονικές περίοδοι ισχύος των κόμβων αυτών να έχουν την ελάχιστη δυνατή χρονική επικάλυψη, θεωρώντας ότι κάποια αντικείμενα μπορούν να ανήκουν και στους δύο νέους κόμβους (όπως άλλωστε συμβαίνει και στα Χρονικά Διασπώμενα Β-δένδρα),
- **διάσπαση με βάση τα σημασιόμενα (semantics)**, όπου τα αντικείμενα διαχωρίζονται με βάση χωροταξικά αλλά και χρονικά κριτήρια, που μπορούν να προσδιορισθούν από την εφαρμογή.

Είναι προφανές ότι η πρώτη μέθοδος ενδείκνυται αν οι υποβαλλόμενες ερωτήσεις έχουν κυρίως χωροταξικά κατηγορήματα, ενώ η δεύτερη μέθοδος ενδείκνυται κυρίως σε περιπτώσεις ερωτήσεων με χρονικά κατηγορήματα. Τέλος, η τρίτη μέθοδος είναι ένας συμβιβασμός μεταξύ των δύο προηγούμενων.

Ως προς την επίδοση χώρου, η δομή αυτή είναι αποτελεσματική αν οι διαδοχικές εικόνες δεν διαφέρουν σημαντικά. Σε αντίθετη περίπτωση, διαπιστώνεται ιδιαίτερη επιβάρυνση ως προς τον καταλαμβανόμενο χώρο γιατί το χρονικό διάστημα ισχύος κάθε αντικειμένου (και κατ'επέχαση, κάθε κόμβου) θα είναι μικρό. Έτσι, καθώς θα εισάγονται νέες εικόνες, το δένδρο θα μεγεθύνεται συνεχώς και η αποτελεσματικότητα της δομής στις διάφορες χωροχρονικές ερωτήσεις θα φθίνει. Στη συνέχεια θα παρουσιασθεί μία ακόμη χωροχρονική δομή, που έχει παρόμοια συμπεριφορά ως προς τις απαιτήσεις χώρου, αλλά δεν πάσχει αντίστοιχα σε σχέση με τις χρονικές επιδόσεις.

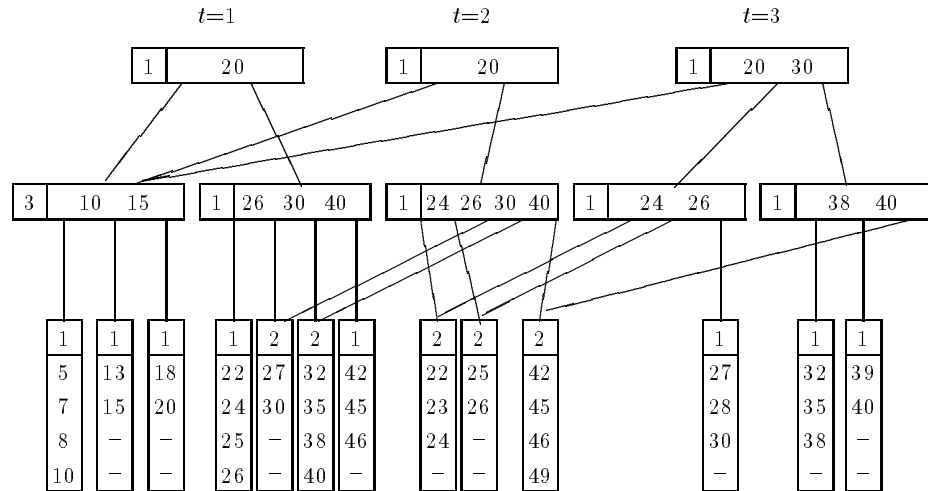
14.7 Επικαλυπτόμενα τετραδικά δένδρα περιοχών

Η τεχνική της επικάλυψης οφείλεται στον Κόλλια και μπορεί να εφαρμοσθεί σε οποιαδήποτε δενδρική δομή. Εφαρμόζοντας την τεχνική αυτή τα παρελθόντα δεδομένα διατηρούνται χωρίς αλλαγές, ενώ αντιγράφονται μόνο όσοι κόμβοι επηρεάζονται από τις εισαγωγές, διαγραφές και ενημερώσεις. Στη συνέχεια εξετάζεται η εφαρμογή της τεχνικής αυτής στο **Επικαλυπτόμενο B^+ -δένδρο** (Overlapping B^+ -tree), που αρχικά σχεδιάστηκε για αποθήκευση διαχρονικών δεδομένων σε μαγνητικό ή σε οπτικό δίσκο WORM.

Οι διαδοχικές εκδοχές ενός αρχείου θα μπορούσαν να παρασταθούν ως ανεξάρτητα B^+ -δένδρα. Οι εκδοχές όμως αυτές μπορεί να διαφέρουν ελάχιστα η μία από την άλλη. Είναι μάλιστα δυνατό πολλά υποδένδρα δύο διαδοχικών εκδοχών να είναι ταυτόσημα. Έτσι οι μεμονωμένες εκδοχές των Επικαλυπτόμενων B^+ -δένδρων συνδέονται με κατάλληλη διαχείριση των δεικτών. Αν η δομή είναι αποθηκευμένη σε μαγνητικό δίσκο, τότε ένα επιπλέον πεδίο, ο **μετρητής αναφορών** (reference counter), μπορεί να χρησιμοποιηθεί για να ελέγχεται αν ένα υπόδενδρο διανέμεται μεταξύ πολλών εκδοχών. Έτσι όλοι οι κόμβοι με μετρητή αναφορών μεγαλύτερο του 1, μαζί με τα παιδιά τους συνιστούν δεδομένα κοινά για περισσότερα του ενός υποδένδρα. Αν η δομή αποθηκεύεται σε δίσκο WORM, τότε δεν συμπεριλαμβάνεται ένα τέτοιο πεδίο στον ορισμό του τύπου της εγγραφής. Ας σημειωθεί επίσης στο σημείο αυτό ότι για λόγους που συνδέονται με την αξιοπιστία του υλικού, σε δίσκους WORM συνιστάται η χρήση των δεικτών προς τα πίσω (backward pointers) και η αποφυγή των δεικτών προς τα εμπρός (forward pointers).

Η συγκεκριμένη δομή γίνεται κατανοητή με τη βοήθεια του Σχήματος 14.14. Το δένδρο έχει τρεις εκδοχές που αποτελούνται από δύο επίπεδα καταλόγου και ένα επίπεδο δεδομένων η κάθε μία. Η πρώτη εκδοχή για $t=1$ περιέχει 21 εγγραφές στο τελευταίο επίπεδο και αρχικά όλοι οι μετρητές αναφορών ισούνται με 1. Η δεύτερη εκδοχή για $t=2$ προέρχεται από την πρώτη με εισαγωγή δύο εγγραφών με κλειδιά 23 και 49. Η τρίτη εκδοχή για $t=3$ προκύπτει με την εισαγωγή δύο ακόμη εγγραφών με κλειδιά 28 και 39 στη δεύτερη εκδοχή. Οι μετρητές αναφορών στο σχήμα αντικατοπτρίζουν την τελική κατάσταση του δένδρου. Έτσι η τιμή αυτών των πεδίων είναι 1 για τη ρίζα, ενώ στα άλλα επίπεδα ποικίλει από 1 μέχρι 3.

Ένα βασικό χαρακτηριστικό των Επικαλυπτόμενων δένδρων είναι ότι έχουν πολλές ρίζες που αντιστοιχούν η κάθε μία σε κάποια χρονική στιγμή,

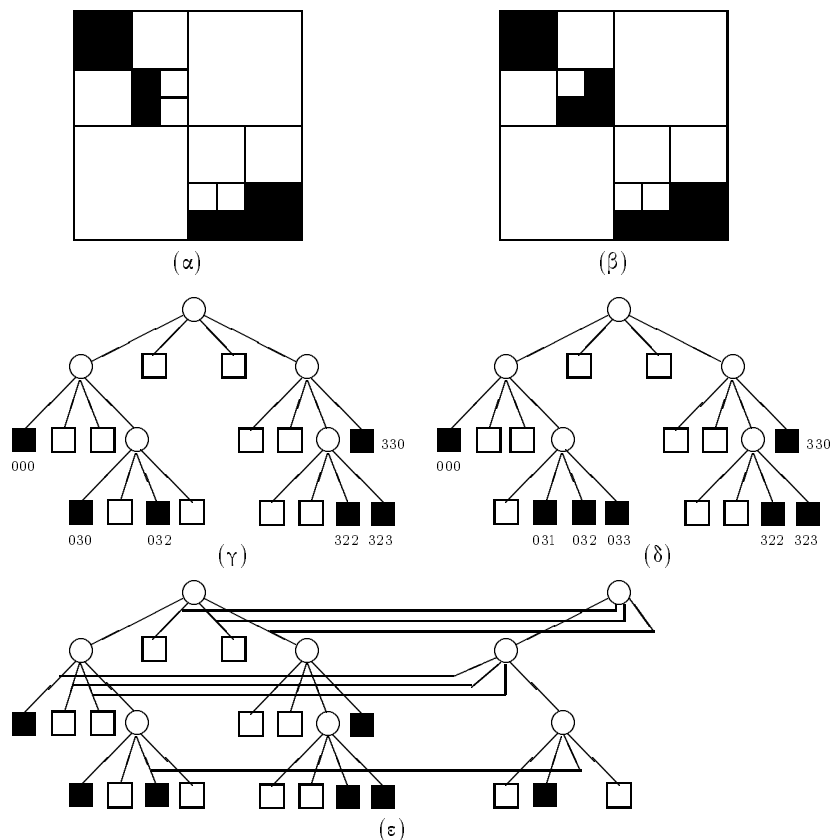
Σχήμα 14.14: Επικαλυπτόμενο B^+ -δένδρο.

σε αντίθεση με τις δομές WOBT και TSBT, όπου από μία ρίζα γίνεται προσπέλαση σε οποιαδήποτε εγγραφή οποιασδήποτε χρονικής στιγμής. Έτσι συμπεραίνεται αμέσως ότι για απλή ερώτηση με βάση οποιαδήποτε χρονική στιγμή τα Επικαλυπτόμενα B^+ -δένδρα έχουν καλύτερη επίδοση γιατί κάτω από μία ρίζα δεν αναμειγνύονται εγγραφές διαφορετικών χρονικών περιόδων. Ένας επιπρόσθετος λόγος είναι ότι στις δομές WOBT και TSBT πολλές εγγραφές επαναλαμβάνονται αυτούσιες σε δύο ή περισσότερους διαφορετικούς κόμβους. Για τον ίδιο λόγο τα Επικαλυπτόμενα δένδρα είναι πιο αποτελεσματικά για μία μαζική ερώτηση που αφορά σε όλες τις εγγραφές μίας χρονικής στιγμής.

Το πλεονέκτημα των Επικαλυπτόμενων B^+ -δένδρων, δηλαδή η μη ανάμειξη στον ίδιο κόμβο εγγραφών από διαφορετικές στιγμές, από μία άλλη σκοπιά είναι και το μειονέκτημά τους. Έτσι τα δένδρα αυτά έχουν τη χειρότερη επίδοση σε ερωτήσεις, που αφορούν στην ανάκτηση εγγραφών ενός χρονικού διαστήματος. Ο λόγος είναι ότι πρέπει να γίνουν σχετικά περισσότερες διασχίσεις των επιμέρους δένδρων ξεκινώντας από διαφορετικές ρίζες.

Η δομή που μόλις περιγράφηκε είναι διαχρονική και όχι χωροχρονική. Ωστόσο, στη δομή αυτή στηρίζεται η δομή των Επικαλυπτόμενων γραμμικών τετραδικών δένδρων περιοχών για την αποθήκευση εικόνων τύπου

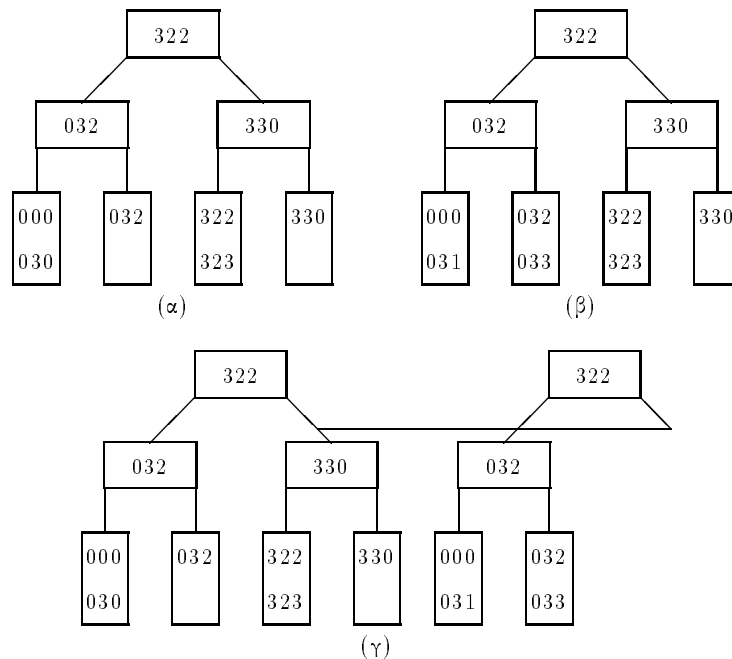
ψηφιδωτού, οι οποίες μεταβάλλονται με το χρόνο. Η νέα δομή στηρίζεται στην παρατήρηση ότι τα Επικαλυπτόμενα B^+ -δένδρα μπορούν να χρησιμοποιηθούν για την αποθήκευση όχι μόνο αριθμών αλλά και εικόνων αν αυτές κωδικοποιηθούν με τη βοήθεια μίας από τις μεθόδους που εξετάστηκαν στο Κεφάλαιο 14.3.



Σχήμα 14.15: Επικαλυπτόμενα τετραδικά δένδρα περιοχών.

Στα Σχήματα 14.15α και 14.15β παρουσιάζονται δύο διαδοχικές εικόνες, που σε μεγάλο βαθμό είναι παρόμοιες. Στα Σχήματα 14.15γ και 14.15δ δίνονται τα Τετραδικά δένδρα περιοχών που παριστούν τις αντίστοιχες εικόνες, ενώ σημειώνονται κάτω από τα μαύρα φύλλα οι κωδικολέξεις σύμφωνα με την υλοποίηση FL. Τέλος, στο Σχήμα 14.15ε υλοποιείται η τεχνική της επικάλυψης για την αποθήκευση των δύο τετραδικών δένδρων χωρίς σπατάλη χώρου για τα κοινά υποδένδρα. Όμως είναι προφανές, ότι η υλοποίηση

αυτή αφορά στην κύρια μνήμη, δηλαδή δεν είναι γραμμική. Στα Σχήματα 14.16α και 14.16β παρουσιάζονται οι γραμμικές υλοποιήσεις που αντιστοιχούν στις προηγούμενες εικόνες. Πιο συγκεκριμένα, θεωρείται μία δομή B^+ -δένδρου με $d=1$ και $Bkfr=2$. Τέλος, στο Σχήμα 14.16γ παρουσιάζεται το ζητούμενο, δηλαδή η δομή των Επικαλυπτόμενων γραμμικών τετραδικών δένδρων περιοχών, όπου και πάλι επιτυγχάνεται οικονομία καθώς τα κοινά υποδένδρα αποθηκεύονται μία μόνο φορά.

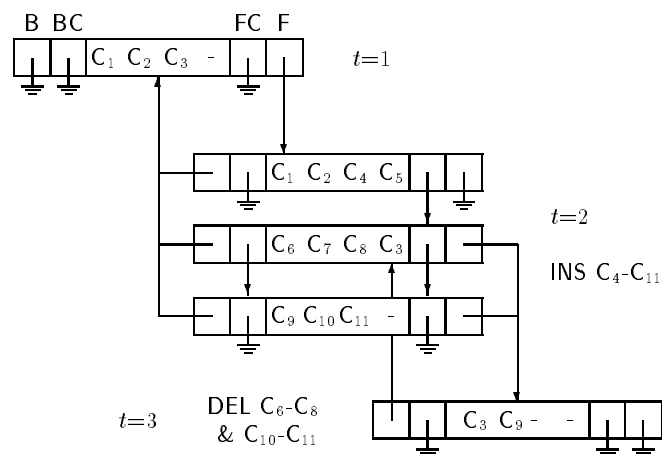


Σχήμα 14.16: Επικαλυπτόμενα τετραδικά δένδρα περιοχών.

Η ιδιαιτερότητα της δομής αυτής είναι ότι κάθε διαδοχική εκδοχή της δομής παρουσιάζει μία νέα εικόνα, ενώ κάθε εικόνα αποτελείται από πολλές κωδικολέξεις FD. Επομένως, όταν εισάγεται μία νέα εικόνα, είναι σαν στην τρέχουσα εκδοχή της δομής να εκτελείται ένα σύνολο διαγραφών και ένα σύνολο εισαγωγών. Αν οι πράξεις αυτές γίνουν μία-προς-μία, τότε είναι εξαιρετικά απίθανο να προκύψουν κοινά υποδένδρα μεταξύ των διαδοχικών εκδοχών της δομής. Για το λόγο αυτό, οι διαγραφές και οι εισαγωγές γίνονται μαζί, οπότε από ένα φύλλο της τρέχουσας εκδοχής μπορεί να διαγραφούν ταυτόχρονα περισσότερες από μία κωδικολέξεις και ταυτόχρονα

να εισαχθούν περισσότερες από μία κωδικολέξεις. Έτσι, γίνεται αντιληπτό ότι αν ένα φύλλο διασπασθεί, τότε μπορεί να προκύψουν περισσότεροι από δύο κόμβοι, ενώ κατά τη συγχώνευση μπορεί περισσότεροι από δύο κόμβοι να δώσουν ένα νέο κόμβο.

Όπως είναι γνωστό, τα B^+ -δένδρα διαθέτουν οριζόντιους δείκτες στο επίπεδο των φύλλων, ώστε να διευκολύνεται η σειριακή προσπέλασή τους. Ωστόσο, λόγω της επικάλυψης δεν είναι δυνατή η χρήση αυτών των δεικτών. Όμως, αν υποβάλλονται ερωτήσεις χρονικού διαστήματος είναι απαραίτητο να γίνει σύνδεση των διαδοχικών εικόνων. Για το σκοπό, σε κάθε φύλλο των Επικαλυπτόμενων γραμμικών τετραδικών δένδρων περιοχών τίθενται τέσσερις ιδιαίτεροι δείκτες, που πρέπει να θεωρηθούν ως δύο ζεύγη. Το ένα ζεύγος χρησιμεύει για να δείχνει από την τρέχουσα εκδοχή προς την επόμενη χρονικά, ενώ το άλλο για να δείχνει προς την προηγούμενη χρονικά εκδοχή. Ειδικότερα, δεδομένου ενός φύλλου της τρέχουσας εκδοχής, ο δείκτης F και ο δείκτης B χρειάζονται για να δείχνουν προς έναν κόμβο της επόμενης εκδοχής, ο οποίος προήλθε από το δεδομένο φύλλο, ή ένα κόμβο της προηγούμενης εκδοχής, από όπου προήλθε το δεδομένο φύλλο, αντίστοιχα. Επιπλέον, σε κάθε εκδοχή ο δείκτης FC και ο δείκτης BC χρησιμεύουν για τη σύνδεση των φύλλων που προήλθαν από μία διάσπαση κατά την προηγούμενη χρονική στιγμή, ή των φύλλων που σε επόμενη χρονική στιγμή θα συγχωνευθούν, αντίστοιχα.



Σχήμα 14.17: Σύνδεση φύλλων με τη βοήθεια τεσσάρων δεικτών.

Το Σχήμα 14.17 δείχνει ένα παράδειγμα όπου ένα φύλλο με $Bkfr=4$ περιέχει 3 εγγραφές κατά τη χρονική στιγμή $t=1$. Οι εγγραφές αυτές έχουν για κλειδιά τους FD κωδικούς: $C_1=002/3$, $C_2=003/3$ και $C_3=302/3$. Έστω, ότι στη συνέχεια και κατά τη χρονική στιγμή $t=2$, εισάγονται οκτώ νέες εγγραφές με κλειδιά τους FD κωδικούς: $C_4=030/3$, $C_5=031/3$, $C_6=032/3$, $C_7=200/2$, $C_8=211/3$, $C_9=330/3$, $C_{10}=331/3$ και $C_{11}=332/3$. Στην περίπτωση αυτή, εκτελείται μία διάσπαση κόμβου αλλά προκύπτουν τρεις νέοι κόμβοι. Το φύλλο της πρώτης χρονικής στιγμής συνδέεται με το πρώτο φύλλο της επόμενης χρονικής στιγμής με τη βοήθεια του δείκτη F, ενώ ο δείκτης FC συνδέει τους τρεις κόμβους μεταξύ τους. Τη χρονική στιγμή $t=3$, διαγράφονται πέντε εγγραφές με κλειδιά τους FD κωδικούς: C_6 , C_7 , C_8 , C_{10} και C_{11} . Έτσι, δύο κόμβοι της δεύτερης χρονικής στιγμής συγχωνεύονται, ώστε να προκύψει ένας κόμβος της τρίτης χρονικής στιγμής, ενώ ο δείκτης B και ο δείκτης BC ενημερώνονται ανάλογα.

Η βασική διαφορά των Επιχαλυπτόμενων γραμμικών τετραδικών δένδρων περιοχών από τη δομή των RT-δένδρων είναι ότι η πρώτη δομή χρησιμοποιείται για την αποθήκευση εικόνων τύπου ψηφιδωτού (raster), ενώ η δεύτερη δομή χρησιμοποιείται για την αποθήκευση εικόνων διανυσματικού τύπου (vector). Και οι δύο δομές έχουν μεγάλες απαιτήσεις σε χώρο, και αυτό αποτελεί μία ομοιότητα στην επίδοσή τους. Ωστόσο, η χρονική επίδοσή τους είναι σημαντικά διαφορετική γιατί η δομή των Επιχαλυπτόμενων γραμμικών τετραδικών δένδρων έχει σταθερή απόδοση ανεξάρτητα από το πλήθος των εικόνων που έχουν εισαχθεί στη βάση, σε αντίθεση με τη δομή των RT-δένδρων, όπου η επίδοση φθίνει καθώς εισάγονται νέες εικόνες. Ο λόγος είναι ότι, επειδή εφαρμόζεται η τεχνική της επικάλυψης, κάθε εικόνα μπορεί να θεωρηθεί ότι αποθηκεύεται ως μία ανεξάρτητη δομή όπου δεν εμπλέχονται δεδομένα των άλλων εικόνων. Βέβαια, κάτι τέτοιο δεν συμβαίνει στην περίπτωση των RT-δένδρων όπου ένας κόμβος της δομής μπορεί να περιέχει ορθογώνια από διάφορες χρονικές στιγμές.

14.8 Επίλογος

Στο κεφάλαιο αυτό εξετάστηκαν μερικές αντιπροσωπευτικές δομές που σχεδιάστηκαν για χρήση στο φυσικό επίπεδο χωροταξικών, διαχρονικών και χωροχρονικών βάσεων δεδομένων. Από τις δομές αυτές, ήδη χρησιμοποιούνται πρακτικά σε εμπορικά συστήματα οι δομές των Γραμμικών τετραδικών

δένδρων και των R-δένδρων. Η έρευνα σε σχέση με τις ειδικές αυτές βάσεις δεδομένων συνεχίζεται. Οι απαιτήσεις των χρηστών που πιέζουν για περαιτέρω ειδικά εργαλεία και οι απαιτήσεις των εταιρειών που αδρανούν σε δραματικά νέες αλλαγές, θέτουν τα όρια της έρευνας και των εφαρμογών της.

Κεφάλαιο 15

ΕΙΔΙΚΑ ΘΕΜΑΤΑ

15.1 Προστασία δεδομένων

15.2 Συμπίεση δεδομένων

15.3 Συντονισμός ταυτόχρονων προσπελάσεων

15.4 Επίλογος

Κεφάλαιο 15

ΕΙΔΙΚΑ ΘΕΜΑΤΑ

15.1 Εισαγωγή

Κατά τη φάση του φυσικού σχεδιασμού μία βάσης δεδομένων, ο διαχειριστής πρέπει να μεριμνήσει και για μία σειρά θεμάτων που σχετίζονται με την εύρυθμη και αποτελεσματική λειτουργία του συστήματος. Επίσης, πέραν του διαχειριστή το ίδιο το σύστημα απαρτίζεται από συντελεστές που έχουν τους ίδιους στόχους αναφορικά με τη λειτουργία του. Αντιθέμενο, λοιπόν, αυτού του κεφαλαίου είναι η παρουσίαση ειδικών προβλημάτων που περισσότερο σχετίζονται με τη γενικότερη λειτουργία του συστήματος και λιγότερο με κάποια ειδική δομή. Τα προβλήματα αυτά αφορούν την προστασία και συμπίεση των δεδομένων και τον έλεγχο των ταυτόχρονων προσπελάσεων χρηστών.

15.2 Προστασία δεδομένων

Χωρίς ιδιαίτερα μέτρα προστασίας (security) από πιθανές βλάβες ή καταστροφές μπορεί να συμβούν μεγάλες ζημιές σε ένα υπολογιστικό σύστημα. Στο κεφάλαιο αυτό δεν τίγονται θέματα προστασίας των εγκαταστάσεων και του προσωπικού αλλά μέθοδοι προστασίας των δεδομένων από καταστροφές ή απώλειες, εκτός αν αυτό προκύψει σύμφωνα με την προδιαγεγραμμένη διαδικασία. Ειδικότερα θα μελετηθεί η **ακρίβεια** (accuracy) και η **ακεραιότητα** (integrity) των αρχείων. Τα δεδομένα ενός αρχείου θεωρούνται ακριβή αν

κατά την εισαγωγή τους δεν περιέχουν λάθη. Ο όρος ακεραιότητα των δεδομένων ισχύει για δεδομένα που έχουν ήδη αποθηκευθεί στη δευτερεύουσα μνήμη.

Προστασία των δεδομένων επιτυγχάνεται με διάφορους ελέγχους κατά την είσοδο και την επεξεργασία των δεδομένων χρησιμοποιώντας μία πληθώρα τεχνικών. Είναι γνωστή η δυνατότητα των λειτουργικών συστημάτων να πιστοποιούν την *αυθεντικότητα* (authentication) της ταυτότητας των χρηστών με τη μέθοδο των *κλειδαρίθμων* (passwords). Μάλιστα, το λειτουργικό σύστημα UNIX έχει τη δυνατότητα να δίνει ή να αρνείται στον κάθε χρήστη οποιαδήποτε δυνατότητα επέμβασης σε ένα συγκεκριμένο αρχείο με σκοπό την ανάγνωση, την ενημέρωση ή την εκτέλεση. Ανάλογη δυνατότητα υπάρχει και στα συστήματα διαχείρισης βάσεων δεδομένων.

Πρωταρχικός έλεγχος είναι η εξουσιοδότηση των χρηστών με συγκεκριμένα δικαιώματα χρήσης των αρχείων ενός συστήματος και επιτυγχάνεται με τη βοήθεια του *πίνακα ελέγχου προσπέλασης* (access control matrix). Μία γραμμή του πίνακα αυτού αντιστοιχεί σε κάθε χρήστη, οι στήλες του πίνακα αντιστοιχούν στα αρχεία του συστήματος, ενώ σε κάθε θέση του πίνακα εισάγονται τα δικαιώματα του χρήστη (δηλαδή, ανάγνωση, αποθήκευση, διαγραφή, αντιγραφή). Για παράδειγμα, η γλώσσα ερωταπαντήσεων SQL, που είναι μία πρότυπη γλώσσα για όλα τα σχεσιακά συστήματα, δίνει στο διαχειριστή του συστήματος τη δυνατότητα παραχώρησης και ανάκλησης δικαιωμάτων στο χρήστη με τη βοήθεια των εντολών ορισμού δεδομένων GRANT και REVOKE <privilege list>. Έτσι, λοιπόν, οι απαιτήσεις του χρήστη μέσα από το σύστημα διαχείρισης της βάσης δεδομένων μεταφράζονται σε απαιτήσεις προς το σύστημα αρχείων του λειτουργικού συστήματος. Στη συνέχεια οι ρουτίνες προσπέλασης συμβουλευονται τον πίνακα ελέγχου προσπέλασης και επιστρέφουν στο σύστημα διαχείρισης της βάσης τα σχετικά δεδομένα.

Μία άλλη μέθοδος προστασίας των δεδομένων είναι ο μηχανισμός της *όψης* (view), που ορίζεται ως ένα αρχείο (πίνακας του σχεσιακού μοντέλου) αποτελούμενο από μερικά πεδία ενός ή περισσότερων αρχείων. Ας σημειωθεί ότι η όψη είναι ιδεατό αρχείο, δηλαδή δεν είναι στην πραγματικότητα αποθηκευμένη στη δευτερεύουσα μνήμη. Στη γλώσσα SQL η όψη υλοποιείται με την εντολή VIEW, ενώ και πάλι με τις εντολές GRANT και REVOKE δίνεται στο χρήστη η δυνατότητα επεξεργασίας μίας όψης. Η προστασία των δεδομένων προέρχεται από γεγονός ότι η όψη αποκρύπτει από το χρήστη μερικά πεδία, που μπορεί να είναι κρίσιμα σύμφωνα με τη

γνώμη του διαχειριστή του συστήματος.

Η ακρίβεια επιτυγχάνεται κατά την είσοδο των στοιχείων με έλεγχο των χαρακτήρων, έλεγχο των πεδίων και έλεγχο των εγγραφών που εισάγονται ομαδικά στο αρχείο. Οι χαρακτήρες μίας εγγραφής ελέγχονται αν είναι κενά, γράμματα, αριθμοί, πρόσημο ή τέλος αν είναι κάποιοι ειδικοί χαρακτήρες. Για παράδειγμα, η παλαιότερη DBase IV αναγνωρίζει τον αλφαριθμητικό τύπο, τον αριθμητικό, τους αριθμούς κινητής υποδιαστολής, τις ημερομηνίες, το λογικό τύπο και τον τύπο των σημειώσεων. Κάτι αντίστοιχο ισχύει σε όλα τα σύγχρονα εμπορικά συστήματα διαχείρισης βάσεων δεδομένων.

Ο έλεγχος των πεδίων είναι πιο ενδιαφέρων για τον προγραμματιστή και μπορεί να επισημάνει και λάθη λογικής εκτός από λάθη πληκτρολόγησης. Ο έλεγχος των πεδίων μπορεί να γίνει κατά πολλούς τρόπους.

Αρίθμηση συναλλαγών (transaction numbering). Μία τεχνική είναι να εισάγεται ένα σύνολο από εγγραφές ταξινομημένες κατά αύξουσα (ή φθίνουσα) τάξη του πρωτεύοντος κλειδιού της. Αν, λοιπόν, το κλειδί μίας εγγραφής δεν υπακούει στην αύξουσα ακολουθία, τότε απορρίπτεται. Επίσης απορρίπτονται οι εγγραφές με ίδιο κλειδί. Ακόμη πολλές φορές οι εγγραφές αριθμούνται και το σύνολό τους πρέπει να είναι σταθερό. Επομένως, αν το πλήθος των εισαγόμενων εγγραφών δεν ισούται με τον προκαθορισμένο αριθμό, τότε η συναλλαγή δεν συνεχίζεται.

Έλεγχος περιορισμών ορθότητας (integrity constrained checking). Μία άλλη τεχνική είναι να ελέγχεται το εύρος των τιμών των πεδίων. Για παράδειγμα, δεν είναι δυνατόν ένας υπάλληλος να έχει εργασθεί περισσότερο από 365 μέρες το χρόνο ή ο μισθός του να υπερβαίνει το 1.000.000 δραχμές το μήνα. Για το σκοπό αυτό, μάλιστα οι γλώσσες ερωταπαντήσεων των συστημάτων διαχείρισης βάσεων δεδομένων έχουν ειδικές εντολές που θέτουν περιορισμούς σε πεδία εγγραφής ενός μόνο αρχείου ή και σε πεδία εγγραφών περισσότερων αρχείων. Ένα παράδειγμα είναι η εντολή `DEFINE INTEGRITY` της `INGRES`:
`DEFINE INTEGRITY ON Student IS Student.mark < 11.`

Έλεγχος πληρότητας . Σε εγγραφές με μεταβλητό αριθμό πεδίων πρέπει να γίνουν ιδιαίτεροι έλεγχοι για τη διαπίστωση ύπαρξης των εγγραφών αυτών.

Έλεγχος ημερομηνιών (date checking). Η ημερομηνία δίνεται υπό τον τύπο MMDDYY ή DDMMYY. Αυτό είναι ένας πρώτος έλεγχος. Ένας δεύτερος έλεγχος εξετάζει τις τιμές που λαμβάνουν οι χαρακτήρες που αντιστοιχούν στις ημέρες, τους μήνες και τα έτη. Για παράδειγμα, έτσι μπορούν να απορριφθούν χρονολογίες μεγαλύτερες από το 1990.

Έλεγχος συνέπειας (consistency checking). Πολλές φορές οι τιμές δύο πεδίων είναι σε άμεση συνάρτηση. Για παράδειγμα, έστωσαν τα πεδία 'Κέρδη' και 'Ζημίες'. Αν στο ένα πεδίο υπάρχει συγκεκριμένη τιμή, τότε στο άλλο πεδίο πρέπει να υπάρχει κενό.

Έλεγχος κωδικών (code checking). Οι κωδικοί αριθμοί συγκρίνονται με πίνακες κωδικών για τη διαπίστωση των ανύπαρκτων. Αν ο πίνακας των κωδικών είναι μεγάλος και η διαδικασία χρονοβόρα, τότε εξετάζονται μόνο τμήματα του κωδικού.

Έλεγχος ψηφίων (digit checking). Τα κυριότερα λάθη που γίνονται κατά την πληκτρολόγηση αριθμών είναι η παραδρομή, πχ. 1237 αντί 1234, η αντιμετάθεση, πχ. 1324 αντί 1234, η διπλή αντιμετάθεση, πχ. 4231 αντί 1234, και τέλος τα τυχαία λάθη, δηλαδή ένας συνδυασμός των προηγούμενων λαθών, πχ. 0214 ή 2124. Τα λάθη που ανήκουν στις τρεις πρώτες κατηγορίες λαθών μπορούν να ανιχνευθούν αν χρησιμοποιηθεί ένα επιπλέον ψηφίο ελέγχου με μία μέθοδο που αναλύεται στη συνέχεια.

Κάθε ψηφίο ενός αριθμού από τα δεξιά προς τα αριστερά πολλαπλασιάζεται με 2,3,...,11,2,3... αντίστοιχα και τα γινόμενα αθροίζονται. Υπ' όψη ότι πρέπει ο μεγαλύτερος αριθμός που χρησιμοποιείται να είναι ένας πρώτος αριθμός, όπως το 11. Έτσι, για τον αριθμό 1234 προκύπτει $1*5+2*4+3*3+4*2 = 5+8+9+8 = 30$. Το 30 διαιρούμενο δια 11 δίνει ακέραιο υπόλοιπο 8. Το 8 αφαιρούμενο από το 11 δίνει υπόλοιπο 3. Το ψηφίο αυτό θεωρείται ψηφίο ελέγχου και τελικά ο αριθμός αποθηκεύεται ως 12343. Αν και πάλι εκτελεσθούν οι προηγούμενες πράξεις και το ψηφίο ελέγχου πολλαπλασιασθεί με τη μονάδα και προστεθεί στο προηγούμενο άθροισμα, δηλαδή το 30, προκύπτει 33 που διαιρούμενο δια 11 δίνει υπόλοιπο 0. Ας σημειωθεί ότι εναλλακτικά, αντί της αλληλουχίας 2,3,..., μπορούν να χρησιμοποιηθούν ως βάρη είτε μία γεωμετρική ακολουθία 1,2,4,8 κλπ. είτε μία ακολουθία πρώτων αριθμών 1,3,5,7,11 κλπ.

Κατά την επεξεργασία γίνεται ο επανέλεγχος και μπορούν να ανιχνευθούν λάθη ενός ή δύο ψηφίων αν ο διαιρέτης είναι πρώτος αριθμός και

μάλιστα μεγαλύτερος από το πλήθος των ψηφίων του αριθμού. Έτσι, ο αριθμός που προκύπτει από το 12373 διαιρούμενος με το 11 δίνει υπόλοιπο 6, αυτός που προκύπτει από το 13243 δίνει υπόλοιπο 1, ενώ αυτός που προκύπτει από το 42313 δίνει υπόλοιπο 9.

Τα τυχαία λάθη δεν ανιχνεύονται πάντοτε. Η πιθανότητα μη ανίχνευσης τυχαίου λάθους είναι ίση προς τον αντίστροφο του διαιρέτη. Αυτό μπορεί να φανεί από τα επόμενα παραδείγματα. Από τον αριθμό 02143 προκύπτει $0*5+2*4+1*3+4*2+3*1 = 22$, που διαιρείται ακριβώς με το 11, αλλά από τον αριθμό 21243 προκύπτει $2*5+1*4+2*3+4*2+3*1 = 31$ που δεν διαιρείται ακριβώς με το 11.

Ας σημειωθεί ότι για την ανίχνευση και τη διόρθωση λαθών υπάρχει μία πληθώρα μεθόδων που ονομάζονται **κώδικες διόρθωσης λαθών** (error correcting codes) και αποτελούν αντικείμενο της Θεωρίας Πληροφοριών. Η πιο γνωστή μέθοδος είναι ο κώδικας Hamming, που αναφέρθηκε στο Κεφάλαιο 2 και χρησιμοποιείται στις μαγνητικές ταινίες και στα συστήματα RAID. Η λεπτομερής εξέταση των κωδίκων διόρθωσης λαθών είναι εκτός των ορίων του βιβλίου αυτού.

15.3 Συμπίεση δεδομένων

Όπως αναφέρθηκε στο Κεφάλαιο 1 εφαρμόζοντας μεθόδους **κωδικοποίησης** (coding) και **συμπίεσης** (compression) των δεδομένων επιτυγχάνεται οικονομία χώρου αποθήκευσης, χρόνου προσπέλασης και ταξινόμησης, καθώς και χρόνου ανάπτυξης των προγραμμάτων εφαρμογών. Στη συνέχεια δίνονται μερικές από τις περισσότερο χρησιμοποιούμενες τεχνικές στην πράξη.

Ένα πολύ χαρακτηριστικό παράδειγμα είναι η συμπίεση των ημερομηνιών. Έτσι, η αποθήκευση της ημερομηνίας '28 Οκτωβρίου 1940' απαιτεί 17 χαρακτήρες ως συμβολοσειρά, ενώ η αποθήκευσή της υπό τη μορφή '281040' απαιτεί 4 χαρακτήρες ως πραγματικός. Αν όμως θεωρηθεί ως αρχή η 1η Ιανουαρίου 1940, τότε η ημερομηνία αυτή μπορεί να αποθηκευθεί ως '301', δηλαδή 2 χαρακτήρες ως αχέραιος. Βέβαια, όπως είναι γνωστό, οι τεχνικές αυτές οδήγησαν στην εμφάνιση του **προβλήματος του 2000** (Y2K problem). Το πρόβλημα αυτό πρόκυψε όχι από λάθος αλλά από ανάγκη. Αρχεί κάποιος να σκεφθεί ότι οι χωρητικότητες των μνημών και των δίσκων πριν, για παράδειγμα, από 20 χρόνια δεν είχαν τα τωρινά μεγέθη, και η εξοικονόμηση έστω και μερικών χαρακτήρων ήταν σημαντική.

Αρχικά δεδομένα:	0,14,0,0,0,15,0,0,0,3,3,0,0,0,0,0,423,0,*
Αναπαράσταση με συντεταγμένες:	2,14,6,15,10,3,11,3,18,423,20,*
Δυαδική αναπαράσταση:	01000100011000000101 και 14,15,3,3,423,*
Δεκαδική αναπαράσταση:	280068 και 14,15,3,3,423,*
Τρεις διακριτές περιπτώσεις:	0110001100011110000011010 και 14,15,3,3,423
Τέσσερις διακριτές περιπτώσεις:	0011000000110000001101000000000000110010 και 14,15,3,3,423

Πίνακας 15.1: Συμπύεση αραιών δεδομένων.

Στη συνέχεια εξετάζονται μερικές περιπτώσεις συμπίεσης αριθμητικών δεδομένων. Λέγεται ότι τα δεδομένα είναι **αραιά** (sparse) όταν σε συντριπτικό ποσοστό έχουν μηδενικές τιμές. Η περίπτωση των αραιών πινάκων έχει αναπτυχθεί και στο βιβλίο των Δομών Δεδομένων, αλλά το θέμα λόγω της σπουδαιότητας του θίγεται και εδώ. Στην πρώτη γραμμή του Πίνακα 15.1 δίνεται ένα παράδειγμα αραιών δεδομένων, όπου το σύμβολο '*' δηλώνει μία μη καθορισμένη (don't care) τιμή. Στη δεύτερη γραμμή του πίνακα δίνεται η αντίστοιχη παράσταση χρησιμοποιώντας τη γνωστή μέθοδο, όπου μόνο οι μη μηδενικές τιμές μαζί με τις συντεταγμένες τους στο αρχείο. Στην τρίτη γραμμή του πίνακα δίνεται η επίσης γνωστή αναπαράσταση αραιών δεδομένων με τη βοήθεια δυαδικών συμβολοσειρών (bitmap). Υπ' όψη ότι αυτή η δυαδική συμβολοσειρά μπορεί να γραφεί σε δεκαδική μορφή ως ο αχέραιος 280068, όπως φαίνεται στην τέταρτη σειρά του πίνακα. Αν υποθεθεί ότι οι μη καθορισμένες τιμές είναι σχετικά πολλές και ότι ενδιαφέρει η επακριβής θέση τους στον πίνακα με δυαδική αναπαράσταση, τότε πρέπει να γίνει διάκριση σε τρεις διακριτές περιπτώσεις. Έτσι, στην πέμπτη σειρά του πίνακα, οι μη καθορισμένες τιμές παριστώνται με 10, οι μη μηδενικές τιμές παριστώνται με 11, ενώ φυσικά με 0 παριστώνται οι μηδενικές τιμές. Τέλος, αν υπάρχουν πολλές επαναλαμβανόμενες τιμές τότε μπορούν να χρησιμοποιηθούν τέσσερις διακριτές τιμές: το 00 για τις μηδενικές τιμές, το 01 για τις επαναλαμβανόμενες τιμές, το 10 για τις μη καθορισμένες και το 11 για τις καθορισμένες μη μηδενικές τιμές, όπως παρουσιάζεται στην έκτη σειρά του πίνακα. Έχει αναφερθεί ότι αν η τελευταία μέθοδος εφαρμοσθεί σε μεγάλα αρχεία με στατιστικά δεδομένα, τότε ο απαιτούμενος χώρος μειώνεται κατά 45% ως 80%.

Όταν τιμές των δεδομένων είναι μικρές τότε μπορούν να παρασταθούν με λίγα bits αντί για ολόκληρες λέξεις. Ο αριθμός των απαραίτητων bits βρίσκει εύκολα με μία σάρωση των δεδομένων. Για παράδειγμα, ο αριθμός

423 του Πίνακα 15.1 απαιτεί 9 bits. Συνεπώς, ενώ με την τελευταία μέθοδο του πίνακα απαιτούνται εκτός του δυαδικού ανύσματος άλλες τέσσερις λέξεις, τώρα απαιτούνται μία λέξη για το δείκτη μεγέθους και 36 bits για τα τέσσερα σημαντικά στοιχεία.

Αρχικά δεδομένα:	854, 855, 857, 856, 859, 860, 863
Συμπιεσμένη μορφή:	854, 4, 1, 2, -1, 3, 1, 3

Πίνακας 15.2: Κωδικοποίηση Δέλτα.

Αν οι τιμές των δεδομένων μεταβάλλονται αργά και βαθμιαία, τότε μπορεί να εφαρμοσθεί η **κωδικοποίηση Δέλτα**. Σύμφωνα με τη μέθοδο αυτή κατ' αρχήν αποθηκεύεται η πρώτη τιμή, κατόπιν ο αριθμός των απαραίτητων bits για την αποθήκευση των διαφορών και στη συνέχεια οι διαφορές. Ένα τέτοιο παράδειγμα φαίνεται στον Πίνακα 15.2. Σύμφωνα με μία παραλλαγή της μεθόδου αυτής παρεμβάλλεται περιοδικά ένας ολόκληρος αριθμός μετά από ένα συγκεκριμένο αριθμό διαφορών. Ειδική περίπτωση είναι η λεγόμενη **κωδικοποίηση των Αζτέκων**, όπου χρειάζεται μόνο ένα bit για την αύξηση ή τη μείωση των τιμών των δεδομένων. Η τελευταία μέθοδος είναι αποτελεσματική μόνο αν η περίοδος δειγματοληψίας είναι πολύ μικρή.

Αρχικοί ακέραιοι:	v_1, v_2, v_3, v_4
Τελικός ακέραιος:	$v_1 + m \times v_2 + m^2 \times v_3 + m^3 \times v_4 \quad (m > v_i)$

Πίνακας 15.3: Συνδυασμός πολλών μικρών ακεραίων σε ένα ακέραιο.

Πολλές φορές είναι δυνατόν ένας αριθμός μικρών ακεραίων να συνδυασθεί σε ένα μόνο ακέραιο με τη μέθοδο που φαίνεται στον Πίνακα 15.3. Αν μάλιστα η παράμετρος m του παραδείγματος είναι δύναμη του δύο, τότε η κωδικοποίηση και η αποκωδικοποίηση διευκολύνονται με την πράξη της διολίσθησης (shift).

Η συμπίεση χειμένου είναι επίσης ένα μεγάλο κεφάλαιο με ευρεία πρακτική εφαρμογή. Ήδη στο Κεφάλαιο 12.2 έγινε εκτενής αναφορά στη μέθοδο της κωδικοποίησης με υπέρθεση και δεν πρόκειται να εξετασθεί πάλι στο παρόν κεφάλαιο. Ένα πολύ συχνό παράδειγμα είναι η **σύντμηση** ή **συντομία** (abbreviation) ταξινομημένων κλειδιών καταλόγων και λέγεται **εσωτερική**

Edgar R.B. Hagstrom 155 Proteus Park Chicago, Ill 60282	60282HGS5155POT31
---	-------------------

Πίνακας 15.4: Ταχυδρομική διεύθυνση και αντίστοιχη κωδικοποιημένη.

(internal) ή εξωτερική (external), αν γίνεται πριν ή μετά την εισαγωγή στο σύστημα αντίστοιχα. Παράδειγμα εξωτερικής σύντμησης είναι μία μέθοδος κωδικοποίησης που εφαρμόζεται κυρίως σε στοιχεία συνδρομητών. Στον Πίνακα 15.4 δίνεται μία πλήρης ταχυδρομική διεύθυνση και η αντίστοιχη κωδικοποιημένη. Η κωδικοποιημένη διεύθυνση προκύπτει με πρόταξη του ταχυδρομικού κώδικα και παράθεση όλων των υπογραμμισμένων γραμμάτων, ενώ οι αριθμοί αντιστοιχούν στο πλήθος των πλαγίων γραμμάτων. Ακολουθεί ένας μετρητής, το 1, που αυξάνει στην περίπτωση που προκύψουν δύο ή περισσότερες όμοιες συμπιεσμένες συμβολοσειρές. Ας σημειωθεί ότι η πρόταξη του ταχυδρομικού κώδικα εξυπηρετεί στην ταξινόμηση.

1.	Αγνοούνται όλοι οι μη αλφαβητικοί χαρακτήρες		
2.	Οι πεζοί χαρακτήρες μετατρέπονται σε κεφαλαία		
3.	Λαμβάνεται ο πρώτος χαρακτήρας του ονόματος στο αποτέλεσμα		
4.	Αγνοούνται τα φωνήεντα A,E,I,O,U,Y και τα σύμφωνα H,W		
5.	Γίνονται οι εξής μετατροπές:		
	τα χειλεόφωνα	B,F,P,V	σε 1
	τα λαρυγγικά και συριστικά	C,G,J,K,Q,S,X,Z	σε 2
	τα οδοντικά	D,T	σε 3
	τα μακρά υγρά	L	σε 4
	τα ένρινα	M,N	σε 5
	τα βραχεία υγρά	R	σε 6
6.	Δύο ή περισσότερα ίδια όμοια ψηφία αντικαθιστώνται με ένα		
7.	Τα πρώτα τρία ψηφία λαμβάνονται στο αποτέλεσμα		

Πίνακας 15.5: Διαδικασία συμπίεσης με μέθοδο SOUNDEx.

Η SOUNDEx είναι μία σπουδαία μέθοδος εξωτερικής σύντμησης, η οποία χρησιμεύει για τη συμπίεση κυρίων αγγλικών ονομάτων και μπορεί να θεωρηθεί σαν ένας φωνητικός κατακερματισμός. Ο λόγος αυτής της θεώρησης είναι το γεγονός ότι ονόματα που μοιάζουν φωνητικά μεταξύ τους συγχεντρώνονται σε παραπλήσιες θέσεις του αρχείου, όπως συμβαίνει με

τις συγχρονούμενες εγγραφές ενός αρχείου κατακερματισμού. Το πλεονέκτημα της μεθόδου είναι ότι με τη μέθοδο αυτή μειώνονται οι συνέπειες που μπορεί να συμβούν από ένα λάθος κατά την επεξεργασία ενός ονόματος. Οι κανόνες συμπίεσης ενός ονόματος με τη μέθοδο SOUNDEX παρουσιάζονται στον Πίνακα 15.5. Όπως εύκολα φαίνεται, το αποτέλεσμα της διαδικασίας είναι μία συμβολοσειρά από ένα μέχρι τέσσερις χαρακτήρες. Το πλήθος των διαφορετικών συμβολοσειρών SOUNDEX είναι 26×7^3 , δηλαδή 8918 δυνατότητες. Άρα είναι πολύ πιθανό δύο ή περισσότερα ονόματα να δίνουν τον ίδιο κώδικα SOUNDEX, όπως εξάλλου συμβαίνει και στον απλό κατακερματισμό. Για παράδειγμα, τα ονόματα MacCloud και McLeod ή τα ονόματα Smith και Schmidt δίνουν αποτέλεσμα M243 και Σ53 αντίστοιχα. Επομένως είναι πιθανό κατά την αναζήτηση ενός ονόματος να επιστραφεί στο χρήστη ένα σύνολο ονομάτων, οπότε η επιλογή θα γίνει με ευθύνη του σε συνδυασμό με τα υπόλοιπα πεδία.

Οι μέθοδοι εσωτερικής σύντμησης είναι βασικά δύο, αλλά υπάρχει και ο υβριδικός συνδυασμός τους. Σύμφωνα με τη **σύντμηση κλειδιού χαμηλής τάξης** (low order key abbreviation) ή **οπίσθια περικοπή** (rear truncation) λαμβάνονται από τα διαδοχικά κλειδιά τόσοι αρχικοί χαρακτήρες, ώστε να υπάρξει διαφοροποίηση μεταξύ τους και αγνοούνται οι τελικοί χαρακτήρες τους. Η τεχνική αυτή είναι γενικά γνωστή από τους τόμους των πολύτομων εγκυκλοπαιδειών και λεξικών, αλλά σε ότι αφορά στα αρχεία χρησιμοποιείται στα Προθεματικά B⁺-δένδρα επιτυγχάνοντας μεγαλύτερο λόγο διακλάδωσης και επομένως μεγαλύτερο κέρδος σε χώρο και χρόνο. Πιο συγκεκριμένα, στις σελίδες των κατωτέρων επιπέδων του δένδρου δεν αποθηκεύονται ολόκληρα τα κλειδιά αν όλα έχουν κοινό πρόθεμα. Το πρόθεμα αυτό μπορεί να ανιχνευθεί στα ανώτερα επίπεδα του δένδρου. Ακόμη στα Προθεματικά B⁺-δένδρα τα συμπιεσμένα κλειδιά μίας σελίδας δεν αποθηκεύονται ολόκληρα αλλά μόνο οι διαφορές τους ως προς το προηγούμενο κλειδί. Η τεχνική αυτή χρησιμοποιείται και από την οργάνωση VSAM στην ομάδα καταλόγων ως εξής: κάθε συμβολοσειρά αποθηκεύεται με ένα ψηφίο που δείχνει ποσά κοινά γράμματα έχει το κλειδί αυτό με το προηγούμενό του και έπονται τα υπόλοιπα γράμματα της συμβολοσειράς. Αν οι διαδοχικές συμβολοσειρές διαφέρουν κατά πολύ, τότε το κέρδος σε χώρο είναι σημαντικό.

Σύμφωνα με τη **σύντμηση κλειδιού υψηλής τάξης** (high order key abbreviation) ή **εμπρόσθια περικοπή** (front truncation), αν οι αρχικοί χαρακτήρες διαδοχικών συμβολοσειρών είναι ίδιοι τότε μπορούν να αγνοηθούν ή να συμπιεσθούν. Παρ' ότι η τεχνική αυτή θεωρητικά έχει πλεονεκτήματα

καθώς επιτυγχάνει οικονομία χώρου, εντούτοις έχει σοβαρά μειονεκτήματα καθώς:

- απαιτεί αυξημένο χρόνο επεξεργασίας, και
- χαθιστά επαχθέστερο το συντονισμό των ταυτόχρονων προσπελάσεων (δες επόμενο υποκεφάλαιο).

Για παράδειγμα, στην έκδοση Oracle 5 χρησιμοποιήθηκε η τεχνική αυτή, που όμως καταργήθηκε στην επόμενη έκδοση, καθώς διαπιστώθηκε ότι η επίδραση της στην επίδοση ήταν αρνητική. Πιο σύνθετη είναι βέβαια η περίπτωση που χρησιμοποιείται συνδυασμός των συντμήσεων υψηλής και χαμηλής τάξης. Προφανώς τα δεδομένα που είναι αποθηκευμένα κατά αυτόν τον τρόπο μπορούν να αναγνωσθούν μόνο σειριακά. Στον Πίνακα 15.6 φαίνονται τα κλειδιά που περιέχονται σε μία σελίδα αρχικά χωρίς σύντμηση, με σύντμηση υψηλής τάξης, με σύντμηση χαμηλής τάξης και κατόπιν με συνδυασμό των δύο συντμήσεων.

Κλειδί	Υψηλής τάξης	Χαμηλής τάξης	Συνδυασμός
JEFCOATE IF	JEFCOATE IF	JEFCO	JEFCO
JEFCUT R	4UT R	JEFCU	3CU
JEFF CJ	3F CJ	JEFF C	3F C
JEFF D	5D	JEFF D	5D
JEFF KD	5KD	JEFF K	5K
JEFFAIR DC	4AIR DC	JEFFAI	4AI
JEFFARES AS	5RES AS	JEFFARES A	5REA A
JEFFARES K	9K	JEFFARES K	9K
JEFFARES KM	9KM	JEFFARES KM	9KM
JEFFCOATE DV	4COATE DV	JEFFCOATE DV	4COATE DV

Πίνακας 15.6: Μέθοδοι εσωτερικής σύντμησης.

Εκτός από τη σύντμηση υπάρχουν και πολλές άλλες μέθοδοι κωδικοποίησης συμβολοσειρών. Έτσι, αν κάθε χαρακτήρας παρασταθεί με 7 ή με 6 bits γίνεται οικονομία κατά 12.5% ή 25% αντίστοιχα. Με 6 bits μπορούν να παρασταθούν 64 διαφορετικοί χαρακτήρες που συνήθως είναι τα κεφαλαία και τα πεζά αγγλικά γράμματα, τα δέκα ψηφία και δύο σύμβολα (το κενό και η παύλα). Αν δεν υπάρχει ανάγκη χρήσης πεζών γραμμάτων, τότε αρκούν 5 bits.

Συχνά στις συμβολοσειρές συναντώνται ακολουθίες κενών, όπως στον Πίνακα 15.7. Μία τέτοια ακολουθία μπορεί να παρασταθεί με ένα μόνο

Giovannini	123.89	Giovannini	1123.89
Jean	87.50	Jean	887.50
Johann	103.76	Johann	5103.76
John	154.66	John	7154.66

Πίνακας 15.7: Διαγραφή κενών.

κενό ακολουθούμενο από ένα αριθμό που δηλώνει το πλήθος των κενών. Επίσης αντί ενός πλήθους συνεχόμενων ομοίων χαρακτήρων μπορεί να τεθεί ένα ειδικό σύμβολο, ένας αριθμός που δηλώνει το πλήθος των χαρακτήρων και στη συνέχεια ο χαρακτήρας. Για παράδειγμα, αντί `'111111'` μπορεί να γραφεί `'*61'`. Προφανώς, όταν το πλήθος των ομοίων χαρακτήρων είναι μεταξύ του τέσσερα και του εννέα, τότε η μέθοδος είναι αποδοτική. Αν το πλήθος των ομοίων συνεχόμενων χαρακτήρων είναι περισσότερο του δέκα, τότε μπορεί να γίνει χρήση διψήφιου μετρητή. Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί για τη συμπίεση χαρτών που παριστώνται με πίνακες τύπου raster. Συνηθισμένη τεχνική επίσης είναι η αντικατάσταση των συχνών λέξεων με απλούς χαρακτήρες, όπως για παράδειγμα οι αγγλικές λέξεις the, of, to, and, in, that, is, it, for, one.

Γράμμα	Συχνότητα	Κωδικός	Γράμμα	Συχνότητα	Κωδικός
E	133	111	U	28	01010
T	93	000	M	27	00110
O	85	1111	P	22	00100
A	81	1110	Y	15	011111
N	75	1100	W	15	011110
I	71	1011	G	14	001111
R	70	1010	B	13	001110
S	65	0110	V	10	001010
H	61	0100	K	5	0010110
D	43	11011	X	3	00101110
L	38	11010	J	2	001011110
C	31	01110	Q	2	001011111
F	29	01011	Z	1	0010111110

Πίνακας 15.8: Συχνότητα εμφάνισης και κωδικοί Huffman.

Είναι γνωστό ότι τα γράμματα του αλφαβήτου δεν εμφανίζονται ισόπιθανα σε ένα κείμενο. Στον Πίνακα 15.8 δίνεται η σχετική συχνότητα

εμφάνισης των γραμμάτων του αγγλικού αλφαβήτου και παρατηρείται ότι η απόκλιση στη συχνότητα εμφάνισης μεταξύ των γραμμάτων E και Z είναι τεράστια. Μπορεί να υπάρξει οικονομία στο χώρο αποθήκευσης αν κάθε χαρακτήρας δεν αντιστοιχεί στον ίδιο αριθμό bits. Στην παρατήρηση αυτή στηρίζεται η μέθοδος κωδικοποίησης με σύμβολα μεταβλητού μήκους (variable length symbol encoding), γνωστή και ως μέθοδος Huffman, που αναπτύχθηκε αλγοριθμικά στο βιβλίο των Δομών Δεδομένων. Σύμφωνα με τη μέθοδο Huffman αντιστοιχίζεται μικρότερος ή μεγαλύτερος κωδικός στα συχνότερα και στα σπανιότερα γράμματα, αντίστοιχα. Στον Πίνακα 15.8 φαίνεται επίσης για κάθε γράμμα ο αντίστοιχος κωδικός. Λαμβάνοντας υπ' όψη τις συχνότητες προκύπτει ότι το μέσο μήκος των κωδικών είναι 4,1754 bits, ενώ αν κάθε γράμμα παριστάνονταν με ισομήχεις κωδικούς τότε θα χρειαζόταν $\log_2 26 = 4,70$ bits. Όσο περισσότερο η κατανομή πιθανότητας εμφάνισης των γραμμάτων ενός αλφαβήτου αποκλίνει από την ομοιόμορφη κατανομή, τόσο η μέθοδος αυτή δίνει μεγαλύτερα κέρδη σε χώρο. Επίσης, σημειώνεται ότι αν αλλάξει η κατανομή πιθανοτήτων, τότε το κέρδος μπορεί να μειωθεί αλλά μπορεί και να αυξηθεί. Η προσπέλαση αρχείου κειμένου που έχει συμπιεσθεί με τη μέθοδο αυτή μπορεί να γίνει μόνο με σειριακό τρόπο.

Αν στα δεδομένα εφαρμόζεται κάποια μέθοδος συμπίεσης, τότε προκύπτουν εγγραφές μεταβλητού μήκους που απαιτούν ιδιαίτερη προσοχή στο χειρισμό τους. Σημειώνεται ότι δεν προσφέρουν όλα τα συστήματα αρχείων τη δυνατότητα χρήσης τέτοιων εγγραφών και απομένει στον προγραμματιστή η υλοποίηση των σχετικών διαδικασιών.

15.4 Συντονισμός ταυτόχρονων προσπελάσεων

Όταν ένα αρχείο προσπελάζεται από πολλούς χρήστες ταυτόχρονα, τότε μπορεί να παρουσιασθούν τα λεγόμενα προβλήματα ταυτοχρονισμού (concurrency) και μπορεί να συμβούν μερικές απροσδόκητες παρενέργειες. Οι παρενέργειες αυτές φαίνονται ανάγλυφα στο επόμενο παράδειγμα.

Δύο άτομα έχουν κοινό λογαριασμό ταμειευτηρίου, ο οποίος προς στιγμή έχει υπόλοιπο 500.000 δραχμές. Ταυτόχρονα και οι δύο πηγαίνουν σε διαφορετικά υποκαταστήματα και οι ταμίες κάνουν προσπέλαση για την ανάγνωση της σχετικής εγγραφής. Έστω ότι ο πρώτος ζητά ανάληψη 100.000 δραχμών, ενώ ο δεύτερος 200.000 δραχμών. Συνεπώς από τον πρώτο ταμιά

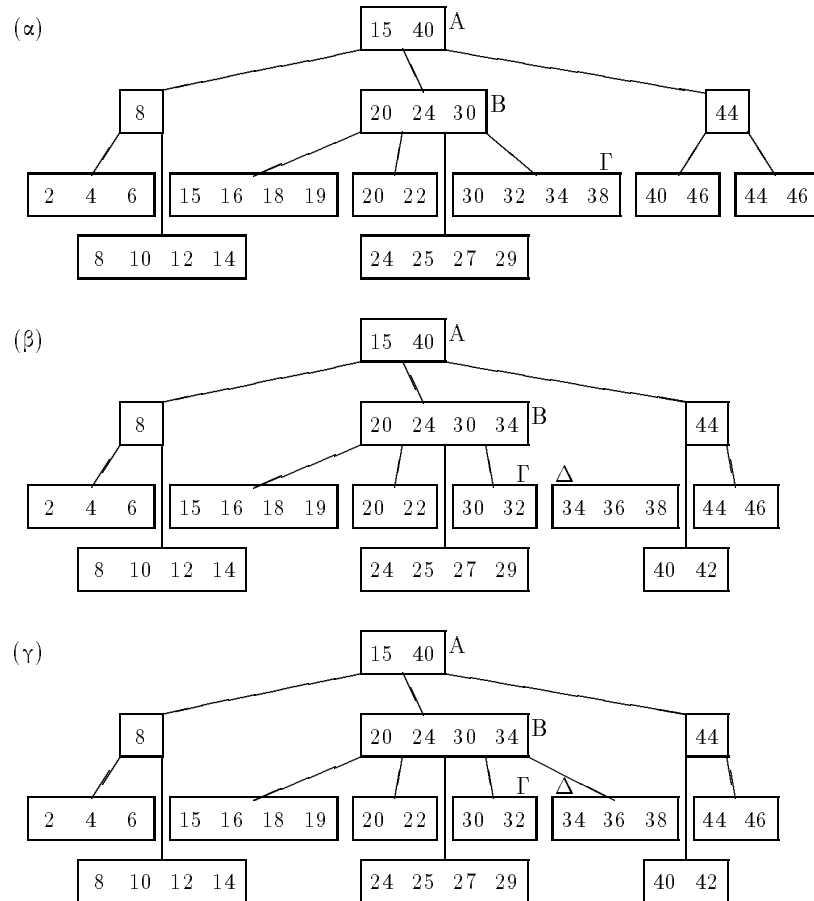
γίνεται αποθήκευση της τιμής 400.000 στο υπόλοιπο του λογαριασμού, ενώ από το δεύτερο γίνεται αποθήκευση της τιμής 300.000. Προφανώς, η τελική αυτή τιμή δεν είναι η ορθή.

Κάθε συναλλαγή ή δοσοληψία (transaction) αποτελείται από επιμέρους βήματα-διεργασίες σε εγγραφές ή σελίδες ενός αρχείου ή περισσότερων αρχείων. Η αλληλουχία των επιμέρους βημάτων λέγεται **χρονικό** (schedule) των συναλλαγών. Ένα χρονικό λέγεται **σειριακό** (serial), αν τα βήματα μίας συναλλαγής δεν παρεμβάλλονται στα βήματα μίας άλλης συναλλαγής, δηλαδή μία συναλλαγή αρχίζει να εκτελείται όταν τελειώσει η προηγούμενή της. Όμως ο κανόνας σε συστήματα πολλών χρηστών και καταμερισμού χρόνου είναι οι επιμέρους διεργασίες ταυτόχρονων συναλλαγών να παρεμβάλλονται μεταξύ τους. Σκοπός του **ελέγχου του ταυτοχρονισμού** (concurrency control) είναι τα βήματα ενός χρονικού να διαταχθούν, ώστε να προκύψει το ίδιο ορθό αποτέλεσμα σαν το χρονικό να ήταν σειριακό. Τότε το χρονικό λέγεται **σειριοποιήσιμο** (serializable). Μία απλή τεχνική που εγγυάται τη σειριοποιεσιμότητα είναι το **κλείδωμα** (locking) των κρίσιμων δεδομένων, δηλαδή η στέρηση της δυνατότητας χρήσης τους από άλλους χρήστες.

Τα προβλήματα ταυτοχρονισμού δεν εμφανίζονται μόνο όταν δύο ή περισσότεροι χρήστες επιθυμούν να προσπελάσουν την ίδια εγγραφή του αρχείου, αλλά ακόμη και διαφορετικές εγγραφές. Για το λόγο αυτό, διακρίνεται το **κλείδωμα εγγραφής** (record locking) και το **κλείδωμα σελίδας** (page locking). Τα κλειδωμάτα σελίδων χρησιμοποιούνται ιδιαίτερα σε ισοζυγισμένες δομές, όπως σε AVL, B-δένδρα κλπ., αλλά και σε τυχαία δυναμικά αρχεία, που διακρίνονται από διασπάσεις και συγχωνεύσεις κόμβων. Στη συνέχεια θα δοθεί ένα ακόμη παράδειγμα ταυτόχρονων προσπελάσεων σε B⁺-δένδρο (με $d=2$ και $Bkfr=4$), που χρησιμοποιείται ως δευτερεύων κατάλογος.

Έστω ότι στο δένδρο του Σχήματος 15.1 πρέπει ταυτόχρονα να εισαχθεί το κλειδί 36 και να αναζητηθεί το κλειδί 38. Στον Πίνακα 15.9 φαίνεται ένα πιθανό χρονικό των επιμέρους βημάτων των δύο διεργασιών. Είναι φανερό ότι η αναζήτηση του κλειδιού 38 καταλήγει σε λανθασμένο συμπέρασμα. Λανθασμένο συμπέρασμα μπορεί να προκύψει σε ταυτόχρονη εισαγωγή και διαγραφή κλειδιών, όπως για παράδειγμα κατά την διαγραφή του κλειδιού 32 και την εισαγωγή του κλειδιού 31 στο δένδρο του Σχήματος 15.1γ.

Για τον έλεγχο του ταυτοχρονισμού σε δομές της οικογένειας των B-δένδρων έχει δοθεί πληθώρα λύσεων. Γενικά είναι παραδεκτή η επόμενη προ-



Σχήμα 15.1: Μορφή B^+ -δένδρου σύμφωνα με τον Πίνακα 15.9.

βληματική. Παρενέργειες δεν δημιουργούνται αν οι απαιτήσεις των χρηστών είναι μόνο αναγνώσεις. Στην αντίθετη περίπτωση πρέπει μερικοί κόμβοι να κλειδωθούν εκ μέρους μίας διεργασίας μέχρι την ολοκλήρωσή της, ώστε να μην είναι προσπελάσιμοι από τις άλλες διεργασίες. Οι αλγόριθμοι κλειδώματος-ξεκλειδώματος κόμβων χωρίζονται σε δύο κατηγορίες: από την κορυφή προς τη βάση (top-down) και από τη βάση προς την κορυφή (bottom-up). Οι αλγόριθμοι της δεύτερης κατηγορίας είναι πιο αποτελεσματικοί γιατί διατηρούν μικρότερο αριθμό κλειδωμένων κόμβων και επιτρέπουν περισσότερο ταυτοχρονισμό.

Βήμα	Εισαγωγή 36	Αναζήτηση 38
1	Βρες το παιδί του κόμβου A (είναι ο κόμβος B)	
2	Βρες το παιδί του κόμβου B (είναι ο κόμβος Γ)	
3		Βρες το παιδί του κόμβου A (είναι ο κόμβος B)
4		Βρες το παιδί του κόμβου B (είναι ο κόμβος Γ)
5	Ο κόμβος Γ είναι φύλλο, πρόσθεσε το 36 στον κόμβο Γ,	
6	ο κόμβος Γ είναι πλήρης Διάσπασε τον κόμβο Γ σε Γ και Δ	
7	Ανέβασε το κλειδί 34 στον κόμβο B	
8		Ο κόμβος Γ είναι φύλλο, αναζήτησε το 38 στον κόμβο Γ, ανεπιτυχής αναζήτηση !

Πίνακας 15.9: Χρονικό βημάτων δύο διεργασιών.

Ιδιαίτερη σημασία για την επίδοση των σχετικών αλγορίθμων έχει το επίπεδο, όπου βρίσκεται ο **ασφαλής κόμβος** (safe node), δηλαδή ο ανώτερος κόμβος του μονοπατιού που μπορεί να απορροφήσει οποιαδήποτε αλλαγή των κατωτέρων επιπέδων της δομής. Αν αυτός ο κόμβος περιέχει λιγότερο από το μέγιστο αριθμό επιτρεπόμενων κλειδιών λέγεται **ασφαλής κατά την εισαγωγή** (insertion safe), ενώ αν περιέχει περισσότερο από τον ελάχιστο αριθμό λέγεται **ασφαλής κατά τη διαγραφή** (deletion safe). Το παιδί του πλησιέστερου προς τα φύλλα ασφαλούς κόμβου ονομάζεται **υψηλότερος μη ασφαλής κόμβος** (highest unsafe node). Είναι φανερό ότι όσο πλησιέστερα προς τη ρίζα είναι ο υψηλότερος μη ασφαλής κόμβος, τόσο δυσκολότερη είναι η ταυτόχρονη χρήση του δένδρου από πολλούς χρήστες λόγω των κλειδωμάτων. Διακρίνονται τεσσάρων ειδών **κλειδώματα**: **ανάγνωσης, εισαγωγής, διαγραφής και αποκλειστικότητας**. Τα κλειδώματα εισαγωγής και διαγραφής είναι ταυτόχρονα και κλειδώματα ανάγνωσης.

Έστω ότι το μέγιστο επιτρεπόμενο περιεχόμενο ενός κόμβου είναι $2d$ κλειδιά, ενώ ένας συγκεκριμένος κόμβος περιέχει s κλειδιά. Ο κόμβος αυτός μπορεί να δεχθεί $(2d-s)$ κλειδώματα εισαγωγής και $(s-d)$ κλειδώματα διαγραφής. Τα κλειδώματα κάθε κόμβου τοποθετούνται σε μία αντιστοιχη

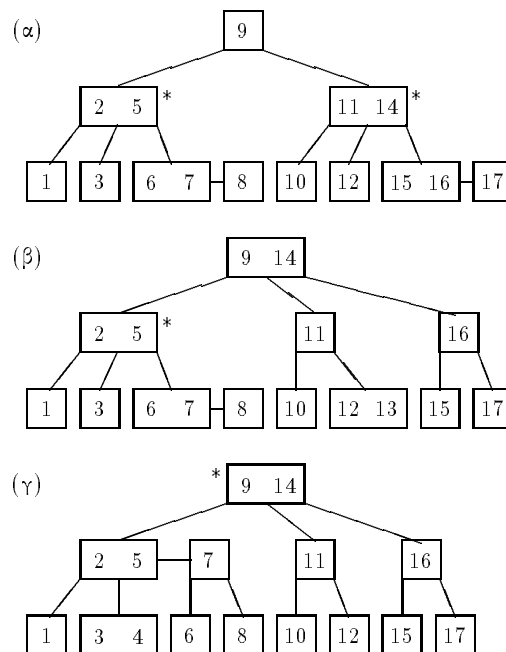
ουρά και παίζουν το ρόλο των κρατήσεων των ελεύθερων θέσεων του κόμβου. Αν μία συναλλαγή απαιτήσει να επιφέρει αλλαγή στο περιεχόμενο του κόμβου, τότε πρώτα το αντίστοιχο κλείδωμα μετατρέπεται σε αποχλειστικό κλείδωμα και διαγράφεται από τη λίστα, και ύστερα εκτελείται η αλλαγή που απαιτείται.

Είναι προφανές ότι ο έλεγχος ταυτοχρονισμού, εκτός του ότι είναι μία πολύπλοκη και χρονοβόρα διαδικασία, απαιτεί την ύπαρξη πινάκων στην κύρια μνήμη για τη διαχείριση των ουρών. Το πρόβλημα του συντονισμού των ταυτόχρονων προσπελάσεων έχει ερευνηθεί εκτεταμένα στο παρελθόν και συνεχίζει να αποτελεί επίκαιρο θέμα, τόσο από πλευράς αλγοριθμικής όσο και από αναλυτικής. Το αντικείμενο εξετάζεται κυρίως στο πλαίσιο του μαθήματος των Βάσεων Δεδομένων, και συνεπώς μεγαλύτερη ανάπτυξη των σχετικών τεχνικών συντονισμού των ταυτόχρονων προσπελάσεων υπερβαίνει τα πλαίσια του βιβλίου αυτού. Ωστόσο, έχει γίνει σημαντική έρευνα, και συνεχίζει να γίνεται, με σκοπό την εύρεση δομών που επιτρέπουν από τη φύση τους μεγαλύτερη ταυτόχρονη χρήση τους από πολλούς χρήστες. Στη συνέχεια θα παρουσιασθεί μία παραλλαγή του B-δένδρου που χαρακτηρίζεται από καλή επίδοση κατά την ταυτόχρονη χρήση της από πολλούς χρήστες.

Είναι γνωστό ότι αν ένα φύλλο B-δένδρου υπερχειλίζει λόγω εισαγωγής μίας εγγραφής, τότε γίνεται διάσπαση του κόμβου με ταυτόχρονη άνοδο του μεσαίου κλειδιού, που μπορεί να προκαλέσει νέα διάσπαση. Από τον Keller προτάθηκε η εξής ενδιαφέρουσα τεχνική (1988). Αν σε περίπτωση υπερχειλίστη ο πατρικός κόμβος δεν είναι πλήρης, τότε ακολουθείται η γνωστή διαδικασία. Όμως αν ο πατρικός κόμβος είναι πλήρης (οπότε με το ανερχόμενο κλειδί θα πρέπει να διασπασθεί) τότε: (α) δημιουργείται ένας νέος κόμβος-φύλλο και οι εγγραφές διαμοιράζονται στους δύο αδελφούς (sibling) κόμβους, (β) το μεσαίο κλειδί αποθηκεύεται στον πρώτο από τους δύο αδελφούς, (γ) οι δύο αδελφοί κόμβοι συνδέονται με έναν οριζόντιο δείκτη, και (δ) ο πατρικός κόμβος σημαδεύεται ότι ο συγκεκριμένος κόμβος δείχνει σε έναν αδελφό.

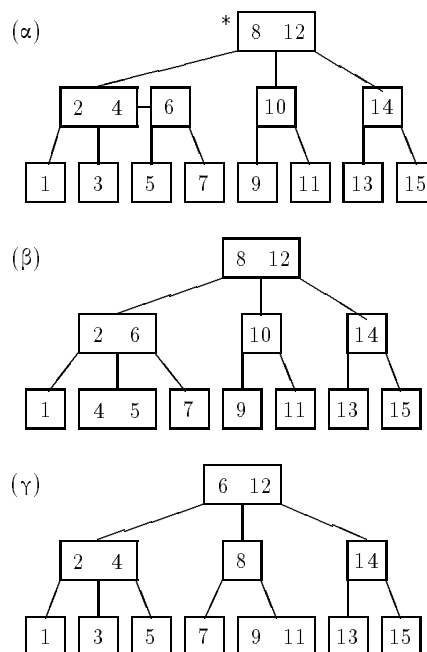
Για να υλοποιηθεί η παραλλαγή αυτή απαιτεί ένα νέο ορισμό του τύπου της εγγραφής της δομής. Πιο συγκεκριμένα, η εγγραφή περιλαμβάνει ένα επιπλέον πεδίο με τιμές από 0 ως m . Αν σε ένα κόμβο η τιμή του πεδίου αυτού είναι 0, τότε κανείς κόμβος που δεικτοδοτείται από το συγκεκριμένο κόμβο δεν δείχνει σε αδελφό του με οριζόντιο δείκτη. Αν η τιμή του πεδίου είναι από 1 ως m , τότε αυτό σημαίνει ότι ο αντίστοιχος δεικτοδοτούμενος κόμβος δείχνει με οριζόντιο δείκτη σε αδελφό κόμβο.

Η καταστρατήγηση του ορισμού του κλασικού B-δένδρου έχει το πλεονέκτημα ότι δεν επιτρέπει στο ανερχόμενο κλειδί να προκαλέσει τη διάσπαση ενός ή περισσότερων κόμβων του μονοπατιού προς τη ρίζα με επακόλουθο την αποφυγή των κλειδωμάτων στα υψηλότερα επίπεδα. Η καταστρατήγηση αυτή είναι προσωρινή, επειδή μπορεί να τακτοποιηθεί σε μία κατοπινή αναζήτηση, εισαγωγή ή διαγραφή. Για παράδειγμα, αν κατά την εκτέλεση μίας αναζήτησης ή μίας εισαγωγής προσπελασθεί ο σημαδεμένος κόμβος, τότε πριν την ολοκλήρωση της ακολουθείται πρώτα η εξής διαδικασία: (α) το φύλλο παύει να δείχνει στον αδελφό του με οριζόντιο δείκτη, (β) το μεσαίο κλειδί ανέρχεται κατά ένα επίπεδο, (γ) ο πατρικός κόμβος διασπάται και προκύπτουν δύο κόμβοι, (δ) το νέο μεσαίο κλειδί ανέρχεται κατά ένα επίπεδο μόνο υπάρχει διαθέσιμος χώρος, αλλιώς οι δύο κόμβοι συνδέονται με οριζόντιο δείκτη και σημαδεύεται ο νέος πατρικός κόμβος. Έτσι, το φαινόμενο αυτό μπορεί να επαναληφθεί σταδιακά στα ανώτερα επίπεδα μέχρι τη διάσπαση της ρίζας. Για το λόγο αυτό η παραλλαγή αυτή ονομάζεται B-δένδρο με τεμπέλικη διάσπαση ρίζας (lazy root splitting B-tree).



Σχήμα 15.2: Εισαγωγές με τεμπέλικη διάσπαση ρίζας.

Η περίπτωση της διαγραφής μίας εγγραφής που ανήκει σε κόμβο που δεικτοδοτείται από σημαδεμένο κόμβο είναι διαφορετική. Δηλαδή, δεν εκτελείται πρώτα η τακτοποίηση και κατόπιν η διαγραφή, αλλά γίνονται ταυτόχρονα. Για παράδειγμα, είναι πιθανό η διαγραφή να αφορά σε μία εγγραφή που είναι αποθηκευμένη σε έναν από τους δύο αδελφούς κόμβους που συνδέονται με οριζόντιο δείκτη. Είναι προφανές ότι μετά τη διαγραφή το περιεχόμενο των δύο κόμβων μπορεί να αποθηκευθεί σε ένα μόνο κόμβο, οπότε θα πρέπει ο σημαδεμένος κόμβος να ενημερωθεί ανάλογα.



Σχήμα 15.3: Διαγραφές με τεμπέλικο διάσπαση ρίζας.

Πρέπει να τονισθεί ότι οι διαδικασίες αναζήτησης, εισαγωγής και διαγραφής είναι αρκετά πιο πολύπλοκες σε αυτή τη δομή απ' ό,τι στο κλασικό B-δένδρο και οι απαραίτητες λεπτομέρειες βρίσκονται στην αναφορά. Όμως δίνονται μερικά παραδείγματα με σχήματα. Στο αρχικό B-δένδρο τάξης $m=3$ του Σχήματος 15.2α εισάγεται μία εγγραφή με κλειδί 13 και κατόπιν μία εγγραφή με κλειδί 4. Η διαδοχική μορφή της δομής παρουσιάζεται στα Σχήματα 15.2β και 15.2γ. Στο Σχήμα 15.3 παρουσιάζεται η μορφή του δένδρου κατά τη διαγραφή. Πιο συγκεκριμένα, το Σχήμα 15.3β προέρχεται από

το Σχήμα 15.3α με διαγραφή της εγγραφής με κλειδί 3, ενώ μετά τη διαγραφή της εγγραφής με κλειδί 10 προκύπτει το Σχήμα 15.3γ. Ο αστερίσχος δηλώνει ότι υπάρχει αδελφός με οριζόντιο δείκτη.

Επακριβή ποσοτικά στοιχεία για τη σύγκριση των δύο δομών δεν υπάρχουν. Ωστόσο η μελέτη των Eisenbarth και άλλων, η οποία αναφέρθηκε στο αντίστοιχο κεφάλαιο δίνει μερικά ενδιαφέροντα στοιχεία για τα τυχαία B-δένδρα τάξης 3. Στη δομή, λοιπόν, αυτή η πιθανότητα να βρίσκεται ο ασφαλής κόμβος κατά την εισαγωγή στο πρώτο, δεύτερο, τρίτο ή ακόμη υψηλότερο επίπεδο από το επίπεδο των φύλλων είναι 0,57, 0,25, 0,1 και 0,08 αντίστοιχα. Βέβαια, οι τιμές αυτές δηλώνουν την ασυμπτωτική πιθανότητα να γίνει διάσπαση κατά την εισαγωγή της $(n+1)$ -οστής εγγραφής. Επομένως, με τις αντίστοιχες επίσης πιθανότητες θα γίνουν αποκλειστικά κλειδώματα στα αντίστοιχα επίπεδα. Αντίθετα, με την εφαρμογή της τεχνικής της τεμπέλικης διάσπασης της ρίζας κλειδώνεται μόνο ο πατρικός κόμβος του κόμβου που υπερχειλίζει. Έτσι είναι προφανές ότι κατά την επεξεργασία αυτής της δομής διατηρούνται λιγότεροι κόμβοι κλειδωμένοι απ' ό,τι στο απλό B-δένδρο. Εξ άλλου με την τακτοποίηση της δομής, η οποία γίνεται τοπικά μετά από μία διαγραφή αποφεύγεται το ενδεχόμενο να γίνουν διαδοχικές αλληπάλληλες συγχωνεύσεις λόγω διαγραφής μετά από διαδοχικές αλληπάλληλες διασπάσεις λόγω εισαγωγής. Βέβαια, ευνόητο είναι ότι όσο μεγαλύτερη είναι η τάξη του δένδρου, τόσο μειώνεται η ευεργετική επίδραση της τεχνικής αυτής στην επίδοση της ταυτόχρονης χρήσης του δένδρου.

15.5 Επίλογος

Ίσως στον αναγνώστη υπάρχει η αντίληψη ότι η σημασία της μελέτης των δομών των αρχείων είναι ανεπίκαιρη, καθώς φαίνεται ότι τα σύγχρονα εμπορικά συστήματα διαχείρισης βάσεων δεδομένων είναι πανίσχυρα. Ωστόσο, η άποψη αυτή είναι λανθασμένη, επειδή στο φυσικό επίπεδο οι βάσεις δεδομένων δεν είναι παρά αλληλοσυνδεδεμένα αρχεία και κατάλογοι και επομένως δεν μπορεί ένα σύστημα διαχείρισης βάσεων δεδομένων να επιτύχει αυξημένες επιδόσεις αν δεν στηρίζεται σε σωστά σχεδιασμένα αρχεία και καταλόγους. Η έρευνα των συστημάτων αρχείων δεν έχει σταματήσει λόγω των εξελίξεων στους παρακάτω, μεταξύ των άλλων, πέντε τομείς.

Πρώτον, η πρόταση νέων δομών αρχείων προκύπτει εν μέρει και από την εξέλιξη της τεχνολογίας υλικού. Όποτε εμφανίζονται νέες συσχευές αποθήκευσης, ταυτόχρονα αναπτύσσονται και νέες οργανώσεις. Ακόμη συμβαίνει

και το αντίστροφο, δηλαδή μία τεχνολογία εγκαταλείπεται η περιορίζεται η εφαρμογή της, αν δεν γίνει η απαραίτητη επένδυση στην ανάπτυξη λογισμικού. Έτσι η εξέλιξη της τεχνολογίας των laser και η εμπορική διάθεση οπτικών δίσκων (όπως VD-ROM, DVD κλπ.) έδωσε έδαφος για την πρόταση νέων δομών που αξιοποιούνται σε πλήθος νέων εφαρμογών, όπως για παράδειγμα στις διαχρονικές βάσεις δεδομένων, στις πολυμεσικές βάσεις δεδομένων (multimedia database systems) κλπ. Άλλη μία εξελισσόμενη τεχνολογία είναι, βέβαια, και η τεχνολογία των ημιαγωγών. Όλο και μεγαλύτερο τμήμα μίας βάσης μπορεί να αποθηκευθεί στην κύρια μνήμη και δεν χρειάζεται να προσπελάζονται τα ίδια δεδομένα συνεχώς στο δίσκο. Οι οργανώσεις αυτές ονομάζονται βάσεις δεδομένων κύριας μνήμης (Main Memory Databases) και έχουν ήδη αποκτήσει εμπορική υπόσταση με σκοπό την υποστήριξη εξειδικευμένων εφαρμογών, όπου ο χρήστης ή η διεργασία απαιτεί άμεση ικανοποίηση. Τέτοιες εφαρμογές εμφανίζονται στο ηλεκτρονικό εμπόριο, στις μηχανές αναζήτησης για τον παγκόσμιο ιστό, σε συστήματα κινητής τηλεφωνίας, καθώς και σε βιομηχανικά ευέλικτα συστήματα συστήματα παραγωγής (flexible manufacturing systems, FMS). Τέτοια εμπορικά συστήματα είναι το TimesTen και το DataBlitz.

Δεύτερον, εμφανίζονται αυξημένες ανάγκες για γεωγραφικά κατανεμημένα επεξεργασία. Αυτό έχει ως αποτέλεσμα να εφαρμόζονται οι τεχνικές της κατανομής (distribution) και της ολικής ή μερικής αντιγραφής (replication) των αρχείων σε περισσότερο από έναν κόμβους του δικτύου, οι οποίοι μπορεί να χρησιμοποιούν διαφορετικά υλικά και ετερογενή λειτουργικά συστήματα. Ο τελικός χρήστης δεν αντιλαμβάνεται που βρίσκονται τα δεδομένα και για το λόγο αυτό λέγεται ότι τα δεδομένα είναι διαφανή (transparent). Βέβαια, το μόνο που μπορεί να αντιληφθεί ο χρήστης είναι η χρονική επίδοση, που φθίνει αν η κατανομή και η αντιγραφή των δεδομένων δεν γίνει με τον καλύτερο τρόπο. Η κατανομή μίας οργάνωσης σε ένα σύνολο ανεξάρτητων δίσκων είναι ένα θέμα που έχει εξετασθεί σε βάθος για πλήθος μεθόδων προσπέλασης. Η κατανομή θα πρέπει να γίνει με τέτοιο έξυπνο τρόπο έτσι ώστε να εξυπηρετείται η αναζήτηση, δηλαδή κατά την αναζήτηση θα πρέπει να μεγιστοποιείται ο παραλληλισμός των προσπελάσεων. Βέβαια, η απαίτηση για μεγιστοποίηση του παραλληλισμού κατά την αναζήτηση ικανοποιείται αν υπάρχουν αντίγραφα των δεδομένων σε όλους τους κόμβους. Ωστόσο, μία τέτοια λύση δεν συνιστάται επειδή, εκτός του απαιτούμενου τεράστιου αποθηκευτικού χώρου, θα υπάρχει και επιπλέον κόστος ενημέρωσης των αντιγράφων σε κάθε περίπτωση εισαγωγής, διαγραφής ή ενημέρωσης. Τέλος, σε ένα κατανεμημένο περιβάλλον πρέπει να υπάρχουν

τεχνικές ανάκτησης (recovery) των δεδομένων των αρχείων σε περίπτωση βλάβης ενός κόμβου. Επίσης, σημειώνεται ότι παρόμοια προβλήματα συναντώνται και στα συστήματα (RAID), που εξετάσθηκαν στο Κεφάλαιο 2.6.

Τρίτον, η ανάπτυξη της μεθοδολογίας του αντικειμενοστραφούς προγραμματισμού (object oriented programming) άνοιξε ένα νέο πεδίο. Στις πρώτες γλώσσες προγραμματισμού ο μοναδικός τύπος ήταν οι πίνακες (μονοδιάστατοι και πολυδιάστατοι), ενώ τώρα όλες οι γλώσσες προσφέρουν και άλλους τύπους, όπως λίστες, εγγραφές, αρχεία, σύνολα κλπ. Οι αντικειμενοστραφείς γλώσσες δίνουν τη δυνατότητα ορισμού σύνθετων αντικειμένων και με βάση αυτά τη δυνατότητα ορισμού υπερ-αντικειμένων. Ο συγχερασμός της τεχνολογίας των βάσεων δεδομένων και της τεχνολογίας του αντικειμενοστραφούς προγραμματισμού έδωσε τη δυνατότητα να ανοίξει το πεδίο των αντικειμενοστραφών βάσεων δεδομένων. Στα συστήματα αυτά δημιουργούνται νέα προβλήματα σε σχέση με τη δημιουργία μεθόδων προσπέλασης, γιατί ο αριθμός των πεδίων του κάθε αντικειμένου είναι μεταβλητός και επομένως το μήκος της εγγραφής του κάθε αντικειμένου είναι επίσης μεταβλητό. Έτσι δεν είναι εύκολο να επιτευχθεί ο διαμερισμός ενός αρχείου αντικειμένων από δύο διαφορετικά προγράμματα. Επομένως, το 'πανταχού παρόν' B-δένδρο ή κάποιο άλλο δένδρο της ίδιας οικογένειας δεν είναι πρακτικά χρήσιμο αν εφαρμοσθεί με την κλασική του υλοποίηση. Ωστόσο, και στο σημείο αυτό έχουν προταθεί παραλλαγές του B-δένδρου για την ικανοποίηση των απαιτήσεων των συστημάτων αυτών.

Στο σημείο αυτό, αξίζει να σημειωθεί η ύπαρξη των λεγόμενων επεκτατών (extensible) συστημάτων βάσεων δεδομένων, που μελετήθηκαν στα τέλη της δεκαετίας του '80. Ο σκοπός των συστημάτων αυτών ήταν να καλύψουν νέες εφαρμογές, όπως στατιστικές εφαρμογές, επεξεργασία εικόνας, πολυμεσικά συστήματα και εφαρμογές γραφείου. Οι εφαρμογές αυτές απαιτούν την υποστήριξη νέων τύπων (όπως κείμενο, ψηφιακή φωνή, εικόνα και βίντεο), νέες συναρτήσεις, σύνθετα αντικείμενα, κανόνες ενεργοποίησης, αποθηκευτικές τεχνικές και μεθόδους προσπέλασης με μεγαλύτερες δυνατότητες μοντελοποίησης και επίδοσης. Κατά δεύτερο λόγο, τα εμπορικά συστήματα είναι πάντα μονολιθικά και δεν μπορούν εύκολα να παρακολουθήσουν τις εξελίξεις. Επομένως, μία ανοιχτή αρχιτεκτονική σε επίπεδο συστήματος διαχείρισης βάσεων δεδομένων θα μπορούσε να καταστήσει το σύστημα περισσότερο εύελιχτο και προσαρμόσιμο στην τεχνολογία και τις ανάγκες των χρηστών. Έτσι, στο παρελθόν υπήρξαν πολλά πρωτότυπα επεκτατά συστήματα, όπως Adt-Ingres, Rad, Dasdbs, Postgres, Probe, Genesis, Exodus, Starburst, R²D², Sabrina κλπ. Εκτός των άλλων, στα

συστήματα αυτά υπήρχε η δυνατότητα ορισμού νέων μεθόδων προσπέλασης από τον (έμπειρο) προγραμματιστή/αναλυτή μέσω καταλλήλων διεπιφανειών. Έτσι, οι νέες αυτές μέθοδοι θα μπορούσαν να τρέχουν επάνω από τα υπάρχοντα συστήματα. Τέλος, η νεότερη αντίληψη στην αρχιτεκτονική των συστημάτων διαχείρισης βάσεων δεδομένων είναι τα **αντικειμενοστραφή-σχεσιακά** (object-relational) συστήματα που συνδυάζουν τα πλεονεκτήματα των αντικειμενοστραφών συστημάτων (δηλαδή, χρήση νέων τύπων) με τα πλεονεκτήματα των σχεσιακών συστημάτων (δηλαδή, ισχυρές μηχανές επεξεργασίας ερωτήσεων). Τα συστήματα αυτά, ακολουθώντας τη φιλοσοφία των επεκτατών συστημάτων, παρέχουν δυνατότητες εμπλουτισμού με νέα υποσυστήματα, σχεδιασμένα για ειδικές εφαρμογές. Μία κλασική περίπτωση είναι η ανοικτή αρχιτεκτονική του Informix με τις επεκτάσεις των Datablades, τα οποία διατίθενται από τρίτους βιομηχανικούς κατασκευαστές ή και ακαδημαϊκά-ερευνητικά ιδρύματα (με ελεύθερη πρόσβαση από το διαδίκτυο). Από την ίδια φιλοσοφία της ανοικτής αρχιτεκτονικής διέπεται η Oracle που δέχεται τα Cartridges ως επεκτάσεις, και η DB2 της IBM που δέχεται τους λεγόμενους Extenders. Στο πλαίσιο αυτό αναφέρεται και η μέθοδος του γενικευμένου δένδρου αναζήτησης (generalized search tree, GiST), που προτάθηκε από τον Hellerstein (1995). Η δομή GiST είναι σημαντική γιατί κάτω από ένα ενιαίο πλαίσιο έχουν υλοποιηθεί το B⁺-δένδρο, το R-δένδρο, ενώ ακόμη μπορεί να υλοποιηθεί κάθε άλλη δομή.

Τέταρτον, σπουδαίο παρόν αλλά και μέλλον στη θεωρία και την πρακτική των βάσεων δεδομένων έχουν οι λεγόμενες **αποθήκες δεδομένων** (data warehouses). Η έννοια αυτή δηλώνει συστήματα που αποτελούν συλλογές δεδομένων και μετα-δεδομένων (δηλαδή, δεδομένων σχετικά με δεδομένα). Πρακτικά, οι αποθήκες αυτές χρησιμοποιούνται ως/σε **Συστήματα Λήψης Αποφάσεων** (Decision Support Systems) με τη βοήθεια τεχνικών on-line αναλυτικής επεξεργασίας (on-line analytical processing, OLAP). Ο τελευταίος όρος δηλώνει κάτι διαφορετικό από τον όρο on-line επεξεργασία συναλλαγών (on-line transaction processing, OLTP), που αναφέρεται σε παραδοσιακές βάσεις δεδομένων, όπου η αποτελεσματικότητα του συστήματος αποτιμάται με βάση των αριθμό των συναλλαγών στη μονάδα του χρόνου. Αντίθετα, ο όρος OLAP δεν αποσκοπεί στο να καταδείξει την αποδοτικότητα ενός συστήματος, καθώς η λήψη μίας απόφασης απαιτεί ιδιαίτερη προσοχή και σχέση. Ένα σύστημα με δυνατότητες OLAP μπορεί να είναι μία καλή επέκταση ενός σχεσιακού συστήματος ή μπορεί να είναι ένας ειδικά κατασκευασμένος OLAP server. Οι OLAP servers σε επίπεδο καταλόγων χρησιμοποιούν τη μέθοδο των κύβων δεδομένων (data cubes).

Οι κύβοι αυτοί είναι πολυδιάστατες προβολές μίας σχέσης, όπου υπολογίζονται όλοι οι δυνατοί συνδυασμοί τελεστών `groupby` (κατά SQL) και τα αποτελέσματα τοποθετούνται σε μία πολυδιάστατη δομή, για παράδειγμα σε ένα R-δένδρο. Μία άλλη οικογένεια δομών που έχουν προταθεί για χρήση σε συστήματα OLAP είναι οι **κατάλογοι με δυαδική αναπαράσταση** (*bitmap indices*), που μοιάζουν με τα κλασικά B⁺-δένδρα, αλλά με τη διαφορά ότι αποθηκεύουν τα κλειδιά στα φύλλα με δυαδική αναπαράσταση. Οι κατάλογοι αυτοί, υπό προϋποθέσεις, μπορεί να καταλάβουν τεράστιο χώρο, και για το λόγο αυτό έχουν προταθεί αρκετές δομές στην προσπάθεια για την εύρεση της πιο κομψής και αποτελεσματικής λύσης. Σημειώνεται ότι στο εμπόριο διατίθενται ήδη μηχανές OLAP, όπως ο Express Server της Oracle.

Πέμπτον, νέα ώθηση στη θεωρία και την τεχνολογία των βάσεων δεδομένων έδωσε η επέκταση του παγκόσμιου ιστού (*web*). Ο παγκόσμιος ιστός μπορεί να θεωρηθεί ως μία τεράστια βάση δεδομένων, όπου ο καθένας μπορεί να κάνει αναζητήσεις (καθώς και άλλες πράξεις που συναντώνται στις βάσεις δεδομένων, όπως συνδέσεις κλπ). Ωστόσο, το πρόβλημα σε ένα τέτοιο περιβάλλον είναι η παντελής έλλειψη πειθαρχίας ως προς τους τύπους των δεδομένων. Για παράδειγμα, ένας αριθμός μπορεί να είναι τύπου ακεραίου αλλά και τύπου συμβολοσειράς. Για το λόγο αυτό, λέγεται ότι τα δεδομένα είναι **ημι-δομημένα** (*semistructured*). Η έρευνα για τη δημιουργία μεθόδων προσπέλασης για ημι-δομημένα δεδομένα είναι σε πρώιμα στάδια. Βασικά, έχουν προταθεί διαφόρων ειδών κατάλογοι για τους γνωστούς τύπους δεδομένων αλλά και κατάλογοι για την οργάνωση των συνδέσεων (*link*), που συνδέουν αντικείμενα, σελίδες και τοποθεσίες (*cites*) μεταξύ τους. Ήδη διατίθενται μέσω του ιστού αρκετά πρωτότυπα συστήματα όπως τα Araneus, Lore, Strudel, Tsimmis, UnQL, W3QS, WebLog, WebSQL κλπ. Ο σκοπός σε τέτοια συστήματα είναι η αναζήτηση δεδομένων μέσα σε αρχεία HTML, XML, POSTSCRIPT, BIBTEX, ASCII κλπ. σε συνδυασμό με την ανάκτηση εικόνων τύπου BMP, GIF, JPG, MPEG κλπ. Προς το παρόν δεν υπάρχουν αντίστοιχα βιομηχανικά εργαλεία στο εμπόριο.

Πιστεύεται ότι ο τόμος αυτός θα αποτελέσει ένα βοήθημα στην περιοχή της φυσικής σχεδίασης βάσεων δεδομένων και θα συντελέσει στην περαιτέρω εξέταση πολλών επίκαιρων προβλημάτων. Η μελέτη των μεθόδων προσπέλασης θα συνεχίσει να αποτελεί μία ενδιαφέρουσα περιοχή για μελέτη, σχεδίαση και ανάπτυξη.

Κεφάλαιο 16

ΠΑΡΑΡΤΗΜΑ

16.1 Εισαγωγή

16.2 Εντολές φυσικού επιπέδου σε Pascal

16.3 Εντολές φυσικού επιπέδου σε C/C++

16.4 Εντολές φυσικού επιπέδου σε C++

Κεφάλαιο 16

ΠΑΡΑΡΤΗΜΑ

16.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μία σύντομη περιγραφή των εντολών χαμηλού επιπέδου, οι οποίες παρέχονται από τις γλώσσες προγραμματισμού Pascal και C/C++. Το βοήθημα αυτό δίνεται ώστε ο αναγνώστης να έχει συγκεντρωμένο το υλικό, που απαιτείται για την υλοποίηση των μεθόδων που εξετάστηκαν στα προηγούμενα κεφάλαια. Θα ήταν χρήσιμο ο αναγνώστης να ανατρέξει σε σχετικά λεπτομερέστερα βιβλία και εγχειρίδια για να εμβαθύνει στα θέματα αυτά.

16.2 Εντολές φυσικού επιπέδου σε Pascal

Τύποι και δηλώσεις μεταβλητών αρχείων

Υπάρχουν 3 τύποι αρχείων:

- `var FileVar: Text;`
Δηλώνει *αρχείο κειμένου* και περιέχει γραμμές χαρακτήρων ASCII.
- `var FileVar: file of DataRecord;`
Δηλώνει *αρχείο καθορισμένου τύπου* και περιέχει δεδομένα ενός μόνο συγχεκτιμένου τύπου.
- `var FileVar: file;`
Δηλώνει *αρχείο μη καθορισμένου τύπου* και περιέχει οποιαδήποτε δεδομένα με οποιοδήποτε μήκος.

Ορισμός φυσικού αρχείου

Η αντιστοίχιση μίας μεταβλητής τύπου αρχείου με ένα πραγματικό αρχείο γίνεται με την εντολή:

- `Assign(FileVar, 'file_name');`

Το αλφαριθμητικό `'file_name'` μπορεί να περιέχει και το μονοπάτι (path) του αρχείου. Η εντολή `Assign` καλείται μόνο μία φορά στο πρόγραμμα.

Άνοιγμα αρχείου

Το άνοιγμα του αρχείου μπορεί να γίνει με 3 τρόπους:

- `Reset(FileVar);`
Ανοίγει υπάρχον αρχείο για ανάγνωση, αρχίζοντας από την αρχή. Αν το αρχείο δεν υπάρχει, τότε προκαλείται λάθος.
- `Rewrite(FileVar);`
Ανοίγει αρχείο για αποθήκευση, αρχίζοντας από την αρχή. Αν το αρχείο υπάρχει ήδη, τότε το σβήνει, αλλιώς το δημιουργεί.
- `Append(FileVar);`
Ανοίγει αρχείο για αποθήκευση με προσάρτηση, δηλαδή αρχίζοντας από το τέλος του. Αν δεν υπάρχει, τότε προκαλείται λάθος.

Ειδικότερα για τα αρχεία καθορισμένου και μη καθορισμένου τύπου, το άνοιγμα με τη συνάρτηση `Rewrite` επιτρέπει και την ανάγνωση από το αρχείο.

Κλείσιμο αρχείου

Ένα αρχείο κλείνει με την εντολή:

- `Close(FileVar);`

Ωστόσο, ακόμα και αν δεν υπάρχει ρητή εντολή κλεισίματος, κάθε αρχείο κλείνει αυτόματα στο τέλος του προγράμματος.

Ανάγνωση και αποθήκευση σε αρχεία κειμένου

Η ανάγνωση τιμών μπορεί να γίνει με 2 εντολές:

- `Read(FileVar, Var1, Var2, ...);`
Διαβάζει όλες μεταβλητές `Var1, Var2, ...` από την τρέχουσα γραμμή του αρχείου.
- `Readln(FileVar, Var1, Var2, ...);`
Διαβάζει όλες τις μεταβλητές `Var1, Var2, ...` από την τρέχουσα γραμμή του αρχείου. Μετά την ανάγνωση τοποθετεί το δείκτη αρχείου στην αρχή της επόμενης γραμμής.

Η `Readln` αγνοεί τυχόντα υπάρχοντα δεδομένα, που μεσολαβούν μέχρι την αρχή της επόμενης γραμμής.

Και οι δύο διαδικασίες ανάγνωσης από αρχείο κειμένου σταματούν την ανάγνωση από την τρέχουσα γραμμή, όταν αναγνωσθεί ο χαρακτήρας αλλαγής γραμμής (`Eoln`). Επίσης, σταματούν την ανάγνωση από το αρχείο, όταν αναγνωσθεί ο χαρακτήρας τέλους αρχείου (`Eof`). Οι δύο αντίστοιχες συναρτήσεις για την ανίχνευση αυτών των χαρακτήρων είναι:

- `Eoln(FileVar);`
Αν ο χαρακτήρας που πρόκειται να αναγνωσθεί είναι ο χαρακτήρας αλλαγής γραμμής (`Eoln`), τότε επιστρέφει `True`,
- `Eof(FileVar);`
Αν ο χαρακτήρας που πρόκειται να αναγνωσθεί είναι ο χαρακτήρας τέλους αρχείου (`Eof`), τότε επιστρέφει `True`.

Η αποθήκευση τιμών μπορεί να γίνει με δύο εντολές:

- `Write(FileVar, Var1, Var2, ...);`
Γράφει τις μεταβλητές `Var1, Var2,`
- `Writeln(FileVar, Var1, Var2, ...);`
Γράφει τις μεταβλητές `Var1, Var2, ...` και προσθέτει και το χαρακτήρα αλλαγής γραμμής (`Eoln`).

Η αποθήκευση με τις εντολές `Write` και `Writeln` δεν γίνεται αυτομάτως στο αρχείο αλλά σε απομονωτικές μνήμες, τα περιεχόμενα των οποίων γράφονται στο αρχείο με ευθύνη του λειτουργικού συστήματος. Για τη ρητή αποθήκευση των περιεχομένων των απομονωτικών μνημών στο αρχείο, χρησιμοποιείται η εντολή:

- Flush(FileVar);

Η εντολή Flush χρησιμοποιείται με αρχεία κάθε τύπου για την εξασφάλιση της αποθήκευσης υπολογισμών, έτσι ώστε σε περίπτωση λάθους σε επόμενο στάδιο, να έχει γίνει τουλάχιστον μερική καταχώρηση των αποτελεσμάτων στο αρχείο.

Ανάγνωση και αποθήκευση σε αρχεία καθορισμένου τύπου

Χρησιμοποιούνται οι διαδικασίες Read και Write με όμοιο τρόπο όπως στα αρχεία κειμένου. Όμως, στην περίπτωση αυτή δεν χρησιμοποιούνται οι διαδικασίες Readln και Writeln. Επίσης, μπορεί να χρησιμοποιηθεί η συνάρτηση Eof, αλλά όχι η Eoln.

Ανάγνωση και αποθήκευση σε αρχεία μη καθορισμένου τύπου

Η ανάγνωση και αποθήκευση στα αρχεία μη καθορισμένου τύπου γίνεται όχι κατά εγγραφές αλλά κατά τμήματα (blocks). Οι αντίστοιχες εντολές είναι:

- BlockRead(FileVar, Buffer, Count, NumRead);
- BlockWrite(FileVar, Buffer, Count, NumCount);

Η μεταβλητή Buffer περιέχει τα δεδομένα που θα αναγνωσθούν/αποθηκευθούν από/προς το δίσκο. Η μεταβλητή Count δηλώνει τον αριθμό των αντικειμένων, ενώ οι μεταβλητές NumRead και NumCount περιέχουν τον επακριβή αριθμό αντικειμένων που αναγνώσθηκαν και αποθηκεύθηκαν, αντίστοιχα. Σε περίπτωση κανονικής ανάγνωσης ή αποθήκευσης, οι τιμές αυτών των μεταβλητών θα είναι ίσες με την τιμή της Count, ενώ σε περίπτωση λάθους θα είναι διαφορετικές.

Το πλήθος των αντικειμένων που δηλώνεται με τη μεταβλητή Count ορίζεται με βάση το μέγεθος του block, που εξ ορισμού είναι 128 bytes. Για να γίνεται πιο κατανοητή η χρήση της ανάγνωσης από αρχεία μη καθορισμένου τύπου συνηθίζεται η αλλαγή του μεγέθους του block σε 1 byte. Αυτό γίνεται κατά το άνοιγμα του αρχείου με την προσθήκη μίας ακόμη παραμέτρου:

- Reset(FileVar, 1);
- Rewrite(FileVar, 1);

Τυχαία αναζήτηση σε αρχεία

Σε σειριακό αρχείο οι αναγνώσεις και οι αποθηκεύσεις γίνονται σε ακολουθία, από την αρχή ως το τέλος. Σε αρχείο τυχαίας προσπέλασης, η επόμενη ανάγνωση ή αποθήκευση μπορεί να γίνει σε οποιοδήποτε σημείο του αρχείου. Αυτό δεν μπορεί να γίνει για τα αρχεία τύπου χειμένου. Η διαδικασία Seek χρησιμοποιείται για την τοποθέτηση του δείκτη αρχείου στην επιθυμητή θέση. Η τρέχουσα τιμή του δείκτη αρχείου ανακτάται με τη συνάρτηση FilePos:

- Seek(FileVar, Offset);
- offset=FilePos(FileVar);

Αν το αρχείο έχει ανοιχθεί με την εντολή Append, τότε δεν μπορεί να γίνει τυχαία αναζήτηση. Επομένως, αν το αρχείο ανοιχθεί με την εντολή Rewrite και χρειάζεται προσθήκη δεδομένων στο τέλος του, μετακινείται πρώτα ο δείκτης στην θέση τέλους και μετά η αποθήκευση εκτελείται κανονικά. Η μετακίνηση γίνεται με την εντολή:

- Seek(FileVar, FileSize(FileVar));

Παράδειγμα

Program Example;

type

```
RecordExm = Record
    id : word;
    value : real;
End;
```

var

```
TextFile : Text;
TypedFile : File of RecordExm;
NonTypedFile1, NonTypedFile2 : File;

Str80 : string[80];
Rec : RecordExm;
Buffer : array[ 1..512] of char;
NumRead, NumWrite : word;
```

Begin

```
(* ----- Αρχείο κειμένου ----- *)
(* Ανάθεση αρχείου κειμένου *)
Assign(TextFile, 'example.txt');
(* Άνοιγμα αρχείου κειμένου για ανάγνωση *)
Reset(TextFile);
(* Ανάγνωση από το αρχείο κειμένου και κλείσιμό του *)
Readln(TextFile, Str80);
Close(TextFile);
(* Άνοιγμα του ίδιου αρχείου κειμένου για αποθήκευση. Δεν μεσολαβεί
δεύτερη κλήση της Assign *)
Rewrite(TextFile);
(* Αποθήκευση στο αρχείο κειμένου και κλείσιμό του *)
Writeln(TextFile, Str80); Close(TextFile);
(* ----- Αρχείο καθορισμένου τύπου ----- *)
(* Ανάθεση αρχείου καθορισμένου τύπου *)
Assign(TypedFile, 'example.dat');
(* Άνοιγμα αρχείου καθορισμένου τύπου για ανάγνωση και αποθήκευση *)
Rewrite(TypedFile);
(* Αποθήκευση μίας εγγραφής στο τέλος του αρχείου *)
rec.id = 100;
rec.val = 123.45;
Seek(TypedFile, FileSize(TypedFile));
Write(TypedFile, rec);
(* κλείσιμο αρχείου καθορισμένου τύπου *)
Close(TypedFile);
(* ----- Αρχεία μη καθορισμένου τύπου ----- *)
(* Άνοιγμα δύο αρχείων μη καθορισμένου τύπου για αντιγραφή των περιε-
χομένων του ενός στο άλλο *)
Assign(NonTypedFile1, 'source');
Assign(NonTypedFile2, 'destination');
```

Reset(NonTypedFile1); (* για ανάγνωση *)
 Rewrite(NonTypedFile2); (* για αποθήκευση *)

Repeat

BlockRead(NonTypedFile1, Buffer, SizeOf(Buffer), NumRead);
 BlockWrite(NonTypedFile1, Buffer, SizeOf(Buffer), NumWrite);

Until (NumRead=0) or (NumRead <> NumWrite);

(* Χρησιμοποιούνται οι τιμές των πραγματικών αριθμών bytes που αναγνώσθηκαν και αποθηκεύθηκαν για να γίνει έλεγχος λάθους. Ο βρόχος τερματίζει όταν δεν υπάρχουν άλλα δεδομένα εισόδου (NumRead=0) ή όταν συμβεί λάθος κατά την αντιγραφή (NumRead <> NumWrite) *)

(* κλείσιμο αρχείων *)

Close(NonTypedFile1);

Close(NonTypedFile1);

End.

Σύνοψη εντολών αρχείων στην Pascal

Ο επόμενος πίνακας συνοψίζει τις διαθέσιμες συναρτήσεις για τις διάφορες λειτουργίες επί των διαφόρων τύπων αρχείων.

Λειτουργία	Υπορουτίνα	Κειμένου	Καθ. τύπου	Μη καθ. τύπου
Άνοιγμα	Append	Ναι	Όχι	Όχι
	Assign	Ναι	Ναι	Ναι
	Reset	Ναι	Ναι	Ναι
	Rewrite	Ναι	Ναι	Ναι
	Close	Ναι	Ναι	Ναι
Είσοδος	BlockRead	Όχι	Όχι	Ναι
	Read	Ναι	Ναι	Όχι
	Readln	Ναι	Ναι	Όχι
Έξοδος	BlockWrite	Όχι	Όχι	Ναι
	Write	Ναι	Ναι	Όχι
	Writeln	Ναι	Ναι	Όχι
Έλεγχος θέσης	Eof	Ναι	Ναι	Ναι
	Eoln	Ναι	Όχι	Όχι
Τυχαίας Προσπέλασης	FilePos	Όχι	Ναι	Ναι
	FileSize	Όχι	Ναι	Ναι
	Seek	Όχι	Ναι	Ναι

16.3 Εντολές φυσικού επιπέδου σε C/C++

Καθώς υπάρχει συμβατότητα μεταξύ των γλωσσών C και C++, οι εντολές της C μπορούν κάλλιστα να χρησιμοποιηθούν και στη C++. Στο Κεφάλαιο 16.4, θα δοθούν οι εντολές που αφορούν αποκλειστικά στη C++.

Τύποι και δηλώσεις μεταβλητών αρχείων

Η δήλωση μίας μεταβλητής αρχείου στη γλώσσα C προϋποθέτει τη χρήση του τύπου δεδομένων FILE. Πιο συγκεκριμένα, απαιτείται η δήλωση μίας μεταβλητής, που είναι δείκτης σε έναν τύπο FILE. Η δήλωση έχει ως εξής:

- FILE * f;

Η μεταβλητή f χρησιμοποιείται για να πραγματοποιηθούν προσπελάσεις στο αρχείο.

Άνοιγμα αρχείου

Πριν από οποιαδήποτε λειτουργία με το αρχείο, πρέπει πρώτα να γίνουν οι απαραίτητες αρχικοποιήσεις, όπως για παράδειγμα ο ορισμός του ονόματος του αρχείου. Οι αρχικοποιήσεις πραγματοποιούνται με τη βοήθεια της συνάρτησης fopen. Η επακριβής σύνταξη της συνάρτησης έχει ως εξής:

- FILE *fopen(const char *fname, const char *access_mode);

όπου fname είναι το όνομα του φυσικού αρχείου και access_mode είναι τα χαρακτηριστικά προσπέλασης που θα έχει το αρχείο (για παράδειγμα, μόνο για ανάγνωση, ή για ανάγνωση/αποθήκευση, κλπ.). Οι διαφορετικές τιμές της μεταβλητής access_mode συνοψίζονται στον ακόλουθο πίνακα:

Τρόπος προσπέλασης	Λειτουργία
r	Ανοίγει το αρχείο μόνο για ανάγνωση.
w	Ανοίγει το αρχείο για αποθήκευση. Αν το αρχείο υπάρχει, τότε τα περιεχόμενά του καταστρέφονται.
a	Ανοίγει το αρχείο για προσάρτηση. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται νέο αρχείο.
r+	Ανοίγει ένα υπάρχον αρχείο για ανάγνωση και αποθήκευση. Αν το αρχείο δεν υπάρχει, τότε επιστρέφεται κωδικός λάθους.
w+	Δημιουργεί ένα αρχείο και το ανοίγει για ανάγνωση και αποθήκευση. Αν το αρχείο υπάρχει, τότε τα τρέχοντα δεδομένα καταστρέφονται.
a+	Ανοίγει ένα αρχείο για ανάγνωση και προσάρτηση. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται ένα νέο.

Οι τιμές αυτές χρησιμοποιούνται για αρχεία κειμένου (δηλαδή ASCII). Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε δυαδικά (δηλαδή binary) αρχεία, τότε πρέπει να προσθέσουμε το χαρακτήρα `b` στις προηγούμενες τιμές (για παράδειγμα, `rb`, `w+b`, κλπ.).

Κλείσιμο αρχείου

Όταν έχει ολοκληρωθεί η εργασία με το αρχείο, πρέπει αυτό να κλείσει. Για το λόγο αυτό χρησιμοποιείται η συνάρτηση `fclose`.

- `int fclose(FILE *f);`

Αν το αρχείο έχει κλείσει με επιτυχία, τότε η `fclose` επιστρέφει μηδέν. Σε περίπτωση σφάλματος, η τιμή επιστροφής είναι ίση με τη σταθερά `EOF`, που ορίζεται στο αρχείο `stdio.h`.

Ανάγνωση και αποθήκευση αρχείων

Δύο είναι οι βασικές λειτουργίες σε ένα αρχείο, η ανάγνωση και η αποθήκευση. Οι βασικές συναρτήσεις είναι οι `fread` και `fwrite`, η σύνταξη των οποίων είναι ως εξής:

- `size_t fread(void *buffer, size_t size, size_t count, FILE *f);`

όπου τα ορίσματα δηλώνουν τα εξής. Η `*buffer` είναι ένας δείκτης στη μνήμη, όπου η `fread` αποθηκεύει τα δεδομένα που διαβάζει από το αρχείο, `size` είναι το μέγεθος του κάθε στοιχείου σε bytes, `count` είναι ο μέγιστος αριθμός στοιχείων που θα διαβασθούν από το αρχείο, και `*f` είναι δείκτης στο αρχείο.

- `size_t fwrite(const void *buffer, size_t size, size_t count, FILE *f);`

όπου και πάλι τα ορίσματα δηλώνουν τα εξής. Η `*buffer` είναι ένας δείκτης στη μνήμη όπου η `fwrite` διαβάζει τα δεδομένα που θα αποθηκευθούν στο αρχείο, `size` είναι το μέγεθος του κάθε στοιχείου σε bytes, `count` είναι ο μέγιστος αριθμός στοιχείων που θα αποθηκευθούν στο αρχείο, και `*f` είναι δείκτης στο αρχείο `FILE`.

Ανάγνωση και αλλαγή θέσης αρχείου

Μερικές φορές είναι χρήσιμο να γνωρίζουμε σε ποιά θέση βρισκόμαστε μέσα στο αρχείο. Η θέση αυτή γίνεται γνωστή με τη βοήθεια της συνάρτησης `ftell`:

- `long ftell(FILE *f);`

Αν η εκτέλεση είναι επιτυχής, τότε η `ftell` επιστρέφει έναν ακέραιο τύπου `long`, ο οποίος περιέχει τον αριθμό των bytes που η τρέχουσα θέση απέχει από την αρχή του αρχείου. Σε περίπτωση λάθους η `ftell` επιστρέφει `-1`.

Για να μετακινήσουμε την τρέχουσα θέση μέσα στο αρχείο χρησιμοποιούμε τη συνάρτηση `fseek`:

- `int fseek(FILE *f, long offset, int origin);`

όπου `offset` είναι η μετατόπιση της νέας θέσης σε bytes από την αρχική θέση, και `origin` είναι μία σταθερά που δείχνει τη θέση από την οποία υπολογίζεται η μετατόπιση.

Στο αρχείο `stdio.h` ορίζονται οι ακόλουθες σταθερές που μπορούν να χρησιμοποιηθούν στη θέση της μεταβλητής `origin`:

- `SEEK_SET` για την αρχή του αρχείου,
- `SEEK_CUR` για την τρέχουσα θέση στο αρχείο, και
- `SEEK_END` για το τέλος αρχείου.

Άλλες συναρτήσεις

Η συνάρτηση `fprintf` χρησιμοποιείται για φορμαρισμένη αποθήκευση σε αρχείο χειμένου, όπως ακριβώς χρησιμοποιείται και η `printf`. Στην περίπτωση της `fprintf` πρέπει να ορίσουμε σε ποιο αρχείο θα εργασθούμε.

- `int fprintf(FILE *f, const char *format_string);`

Η συνάρτηση `fscanf` χρησιμοποιείται για φορμαρισμένη ανάγνωση από αρχείο χειμένου, όπως ακριβώς χρησιμοποιείται και η `scanf`. Στην περίπτωση της `fscanf` πρέπει να ορίσουμε σε ποιο αρχείο θα εργασθούμε.

- `int fscanf(FILE *f, const char *format_string, ...);`

Η συνάρτηση `fputs` χρησιμοποιείται για την αποθήκευση μίας σειράς χαρακτήρων σε ένα αρχείο χειμένου.

- `int fputs(const char *string, FILE *f);`

Η συνάρτηση `fgets` χρησιμοποιείται για την ανάγνωση μίας σειράς χαρακτήρων από ένα αρχείο χειμένου.

- `char *fgets(char *string, int maxchar, FILE *f);`

Η μακροεντολή `feof` χρησιμοποιείται για να προσδιορίσουμε αν βρισκόμαστε στο τέλος του αρχείου ή όχι. Αν έχουμε φθάσει στο τέλος αρχείου, τότε η `feof` επιστρέφει μία μηδενική τιμή, διαφορετικά επιστρέφει μηδέν.

- `int feof(FILE *f);`

Παράδειγμα

```
include <stdio.h>
```

```
main()  
{
```

```
    FILE *f1;  
    FILE *f2;  
    size_t size;  
    size_t count;  
    char string[100];  
    struct { int x; int y;} mystruct;
```

```
    /* άνοιγμα αρχείου χειμένου myfile.txt για ανάγνωση */  
    f1 = fopen("myfile.txt", "r");
```

```
    /* άνοιγμα δυαδικού αρχείου myfile.bin για ανάγνωση και αποθήκευση */  
    f2 = fopen("myfile.bin", "r+b");
```

```
    /* αρχικοποίηση */  
    size = 1;  
    count = 100;
```

```
    /* ανάγνωση 100 χαρακτήρων από το f1 */  
    fread((void *) string, size, count, f1);
```

```
/* αρχικοποίηση */  
mystruct.x = 100;  
mystruct.y = 200;  
size = sizeof(mystruct);  
count = 1;  
  
/* εγγραφή δομής mystruct στο f2 */  
fwrite((void *) &mystruct, size, count, f2);  
  
/* κλείσιμο αρχείων */  
fclose(f1);  
fclose(f2);  
}
```

16.4 Εντολές φυσικού επιπέδου σε C++

Η είσοδος/έξοδος στη C++ μπορεί να γίνει με χρήση ρευμάτων (streams). Με τον τρόπο αυτό γίνεται μετατροπή των αντικειμένων σε σύνολα bytes. Η είσοδος/έξοδος σε αρχείο με streams γίνεται με συναρτήσεις της βιβλιοθήκης `fstream.h`. Στη συνέχεια, αντί του όρου `stream` χρησιμοποιείται ισοδύναμα ο όρος 'αρχείο' για λόγους συμβατότητας με τα προηγούμενα.

Άνοιγμα αρχείου

Ένα αρχείο ανοίγει για είσοδο με την εντολή:

- `ifstream inFile("file_name");`

και για έξοδο με την εντολή:

- `ofstream outFile("file_name");`

Οι κλάσεις `ifstream` και `ofstream` είναι απόγονοι της κλάσης `fstream`. Για χρήση περισσότερων επιλογών κατά το άνοιγμα του αρχείου, δηλώνονται αντικείμενα της κλάσης `fstream` με τον ακόλουθο constructor:

- `fstream file("file_name", mode);`

όπου ο τρόπος ανοίγματος εξαρτάται από την τιμή της παραμέτρου `mode`, η οποία μπορεί να είναι συνδυασμός με `|` (με λογικό `H`) των εξής επιλογών:

- `ios::app`
Άνοιγμα για προσάρτηση, δηλαδή για προσθήκη στο τέλος του αρχείου.
- `ios::in`
Άνοιγμα για είσοδο. Αν το αρχείο υπάρχει, τότε δεν καταστρέφονται τα περιεχόμενά του.
- `ios::out`
Άνοιγμα για έξοδο. Αν το αρχείο υπάρχει, τότε καταστρέφονται τα περιεχόμενά του και γράφονται τα νέα.
- `ios::trunc`
Άνοιγμα με καταστροφή των περιεχομένων του αρχείου. Μπορεί να συνδυασθεί, για παράδειγμα με το `ios::in` (`ios::in | ios::trunc`) για άνοιγμα για είσοδο με ταυτόχρονη καταστροφή των περιεχομένων του αρχείου.
- `ios::nocreate`
Δηλώνει ρητά άνοιγμα χωρίς δημιουργία αρχείου σε περίπτωση που αυτό δεν υπάρχει (ταυτόχρονα προκαλεί και λάθος).

Τέλος, μία βασική επιλογή κατά το άνοιγμα του αρχείου αφορά στο αν αυτό είναι αρχείο κειμένου ή δυαδικό αρχείο. Αυτό γίνεται με την τιμή της παραμέτρου `mode`:

- `ios::binary`

η οποία δηλώνει δυαδικό αρχείο, ενώ αν δεν υπάρχει η τιμή, τότε πρόκειται για αρχείο κειμένου.

Κλείσιμο αρχείου

Το κλείσιμο ενός αρχείου γίνεται με την εντολή:

- `file.close();`

Ακόμη και αν δεν υπάρχει η ρητή εντολή κλεισίματος, το αρχείο κλείνει από το destructor του.

Ανάγνωση και αποθήκευση σε αρχεία

Για την ανάγνωση ενός χαρακτήρα από αρχείο χρησιμοποιείται η εντολή:

- `file.get(ch);`

όπου `ch` είναι μεταβλητή τύπου `char`. Αν συμβεί λάθος κατά την ανάγνωση, τότε επιστρέφει `NULL`. Για την ανάγνωση συγκεκριμένου αριθμού `bytes` υπάρχει η εντολή:

- `file.read(pch, nCount);`

όπου `pch` είναι μεταβλητή τύπου `char*`, ενώ η μεταβλητή `nCount` είναι τύπου `int` και δηλώνει τον αριθμό των `bytes`. Η εντολή `read` χρησιμοποιείται κυρίως για δυαδικά αρχεία.

Για την αποθήκευση ενός χαρακτήρα σε αρχείο υπάρχει η εντολή:

- `file.put(ch);`

όπου `ch` είναι μεταβλητή τύπου `char`. Αν συμβεί λάθος κατά την αντιγραφή, τότε επιστρέφει `NULL`. Για την αποθήκευση συγκεκριμένου αριθμού `bytes` υπάρχει η εντολή:

- `file.write(pch, nCount);`

όπου `pch` είναι μεταβλητή τύπου `char*` και η μεταβλητή `nCount` είναι τύπου `int` και δηλώνει τον αριθμό των `bytes`. Και η εντολή `write` χρησιμοποιείται κυρίως για δυαδικά αρχεία.

Για τον έλεγχο αν ο δείκτης του αρχείου βρίσκεται στο τέλος του αρχείου, χρησιμοποιείται η εντολή:

- `file.eof();`

η οποία επιστρέφει μία μη μηδενική τιμή, αν ο δείκτης είναι στο τέλος του αρχείου.

Τυχαία αναζήτηση σε αρχεία

Για τη μετακίνηση του δείκτη σε αρχείο, που είναι αντικείμενο της κλάσης `ifstream` (δηλαδή, αρχείο για είσοδο), υπάρχει η εντολή:

- `seekg(offset, dir);`

Αν το αρχείο είναι αντικείμενο της κλάσης `ofstream` (δηλαδή, αρχείο για έξοδο), τότε υπάρχει η εντολή:

- `seekp(offset, dir);`

Η παράμετρος `offset` είναι τύπου `streamoff`, που είναι ισοδύναμος με τον τύπο `long`. Η παράμετρος `dir` και στις δύο συναρτήσεις δηλώνει την κατεύθυνση προς την οποία γίνεται η αναζήτηση και μπορεί να πάρει τις εξής τιμές:

- `ios::beg`
Εκτελεί αναζήτηση από την αρχή του αρχείου.
- `ios::cur`
Εκτελεί αναζήτηση από την τρέχουσα θέση στο αρχείο.
- `ios::end`
Εκτελεί αναζήτηση από το τέλος του αρχείου.

Η τιμή του δείκτη αρχείου γίνεται για τα αρχεία κλάσης `ifstream` με την εντολή:

- `filePos = tellg();`

ενώ για τα αρχεία κλάσης `ofstream` με την εντολή:

- `filePos = tellp();`

Η μεταβλητή `filePos` είναι τύπου `long`.

Παράδειγμα

```
#include <fstream.h>
#include <iostream.h>           // για μηνύματα στην οθόνη

void main()
{
    /* Αντιγραφή των περιεχομένων ενός αρχείου σε άλλο */
    ifstream source("inputFileName");
    ofstream destination("outputFileName");

    char ch;
    while (source.get(ch))
        destination.put(ch);

    if (!source.eof())
        cout << "Λάθος κατά την αντιγραφή";

    /* Ο βρόχος while εκτελείται όσο η συνάρτηση get δεν επιστρέφει
    τιμή NULL. Μετά το τέλος του βρόχου γίνεται έλεγχος αν το αρχείο
    έχει φτάσει στο τέλος με τη συνάρτηση eof. Αν δεν συμβαίνει
    αυτό τότε προκαλείται λάθος, γιατί αυτό σημαίνει ότι δεν αντιγρά-
    φηκαν όλα τα περιεχόμενά του */

    /* Ανάγνωση περιεχομένων αρχείου και προβολή των θέσεων στις
    οποίες υπάρχουν κενά */
    ifstream tfile("text");

    while ( !tfile.eof() )
    {
        long here = tfile.tellg();
        tfile.get(ch);
        if (ch == " ")
            cout << "\nPosition " << here << " is a space";
    }
}
```


Κεφάλαιο 17

ΒΙΒΛΙΟΓΡΑΦΙΑ

17.1 Ξένη βιβλιογραφία

17.2 Ελληνική βιβλιογραφία

17.3 Αναφορές

Κεφάλαιο 17

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ξένη βιβλιογραφία

1. Aoe J.I. (editor): "Computer algorithms - key search strategies", IEEE Computer Science Press, 1991.
2. Bernstein P.A., Hadzilakos V., Goodman N.: "Concurrency control and recovery in database systems", Addison-Wesley, 1987.
3. Bourne C.P.: "Methods of information handling", Wiley, 1963.
4. Bradley J.: "File and database techniques", Holt, Rinehart and Winston, 1982.
5. Bouros M.: "Getting into VSAM - an introduction and technical reference", John Wiley, 1985.
6. Claybrook B.G.: "File management techniques", John Wiley, 1983.
7. Folk M.J., Zoelick B., Riccardi G.: "File structures - an object-oriented approach with C++", Addison-Wesley, 1998.
8. Fuller S.H.: "Analysis of drum and disk storage units", Springer Verlag, Lecture Notes in Computer Science, Vol.31, 1975.
9. Gonnet G.H.: "Handbook of algorithms and data structures", Addison-Wesley, 1984.
10. Gray J., Reuter A.: "Transaction processing - concepts and techniques", Morgan Kaufmann, 1993.
11. Grosshans D.: "File systems - design and implementation", Prentice Hall, 1986.
12. Guenther O.: "Efficient Structures for Geometric Data Management", Springer Verlag, Lecture Notes in Computer Science, Vol.337, 1988.
13. Cunningham M.: "File structures and design", Chartwell-Bratt, 1985.
14. Hanson O.: "Design of computer data files", Pitman, 2nd edition, 1988.
15. Harborn T.R.: "File systems - structures and algorithms", Prentice Hall, 1988.

16. Held G.: "Data compression - techniques and applications, hardware and software considerations", Wiley, 1988.
17. Hennessy J., Paterson D.: "Computer architecture - a quantitative approach", Morgan Kaufmann, 1996.
18. Hill E.Jr.: "A comparative study of very large databases", Springer Verlag, Lecture Notes in Computer Science, Vol.59, 1978.
19. Hunter G.M.: "Efficient computation and data structures for graphics", Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Princeton University, NJ, 1978.
20. Johnson L.F., Cooper R.H.: "File techniques for data base organization in COBOL", Prentice Hall, 1981.
21. Knuth D.E.: "The art of computer programming, Vol.3, Sorting and searching", Addison-Wesley, 2nd edition, 1973.
22. Lefkovits D.: "File structures for on-line systems", Spartan Books, 1969.
23. Livadas P.: "File structures - theory and practice", Prentice Hall, 1990.
24. Loomis M.E.S.: "Data management and file processing", Prentice Hall, 1983.
25. Lorin H.: "Sorting and sort systems", Addison-Wesley, 1975.
26. Manolopoulos Y., Theodoridis Y., Tsotras V.: "Advanced database indexing", Kluwer Academic Publishers, 1999.
27. Martin J.: "Computer database organization", Prentice Hall, 2nd edition, 1977.
28. Miller N.E.: "File structures using Pascal", Benjamin Cummings, 1987.
29. Ramakrishnan R.: "Database management systems", McGraw Hill, 1998.
30. Salzberg B.: "File structures", Prentice Hall, 1988.
31. Samet H.: "The design and analysis of spatial data structures", Addison-Wesley, 1990.
32. Shasha D.: "Database tuning - a principled approach", Prentice Hall, 1992.
33. Silberschatz A., Korth H.F., Sudarshan S.: "Database system concepts", 3rd edition, McGraw Hill, 1998.
34. Swan T.: "Mastering Turbo Pascal files", H.W. Sams, 1987.
35. Smith P.D., Barnes G.M.: "Files and databases - an introduction", Addison-Wesley, 1987.
36. Tansel A., Clifford J., Gadia S., Jajodia S., Segev A., Snodgrass R. (eds): "Temporal databases: theory, design and implementation", Benjamin/Cummings, 1993.
37. Teorey T.J., Fry J.P.: "Design of database structures", Prentice Hall, 1982.
38. Tharp A.L.: "File organization and processing", John Wiley, 1988.
39. Vitter J.S., Chen C.W.: "Design and analysis of coalesced hashing", Oxford University Press, 1987.
40. Wiederhold G.: "Database design", McGraw-Hill, 2nd edition, 1983.
41. Wiederhold G.: "File organizations for database design", McGraw-Hill, 1987.
42. Wong C.K.: "Algorithmic studies in mass storage systems", Springer Verlag, 1983.

Ελληνική βιβλιογραφία

1. Βατικιώτης Α.: 'Μηχανογραφικά αρχεία', Αθήνα, 1984.
2. Βλέτσας Γ.: 'Διαχείριση δεδομένων', Αθήνα, 1987.
3. Γιαννακουδάκης Ε.: 'Σχεδιασμός και διαχείριση βάσεων δεδομένων', Εκδόσεις Μπέ-νου, Αθήνα, 1999.
4. Δέρβος Δ.: 'Μαθήματα βάσεων δεδομένων', Εκδόσεις Τζιόλα, Θεσσαλονίκη, 1995.
5. Ζάρκος Σ.: 'Δομές δεδομένων και αρχεία στην C', Αθήνα, 1993.
6. Κοίλιας Χ.: 'Τα Αρχεία της Basic και οι εφαρμογές της', Αθήνα, 1985.
7. Κοίλιας Χ.: 'Αρχεία και βάσεις δεδομένων', Αθήνα, 1995.
8. Κόλλιας Γ.: 'Βάσεις δεδομένων', Τόμος I, Αθήνα, 1987.
9. Κόλλιας Γ.: 'Βάσεις δεδομένων', Τόμος II, Αθήνα, 1987.
10. Λάζος Κ.: 'Οργάνωση αρχείων', Θεσσαλονίκη, 1990.
11. Μανωλόπουλος Ι.: 'Δομές δεδομένων - μία προσέγγιση με Pascal', Art of Text, Θεσσαλονίκη, 1998.
12. Μαρινάκης Κ.Ι., Τασσόπουλος Α.Κ.: 'Αρχεία και βάσεις δεδομένων', Αθήνα, 1989.
13. Παπακωνσταντίνου Γ.Κ., Μπιλάλης Ν.Α., Τσανάκας Π.Δ.: 'Λειτουργικά συστήμα-τα', Τόμος I, Αθήνα, 1986.
14. Παπακωνσταντίνου Γ.Κ., Τσανάκας Π.Δ., Φραγκάκης Γ.Π.: 'Αρχιτεκτονική υπολο-γιστών', Αθήνα, 1987.
15. Wirth N.: 'Αλγόριθμοι και δομές δεδομένων', Αθήνα, 1990.
16. Χρυσουλίδη Δ.: 'Εισαγωγή στη θεωρία πληροφοριών', Θεσσαλονίκη, 1991.

Αναφορές

1. Abel D.J.: "A B⁺-tree for structure for large quadtrees", Computer Vision, Graphics and Image Processing, Vol.27, No.1, pp.19-31, 1984. (14)
2. Anderson H.D., Berra P.B.: "Minimum cost selection of secondary indexes for formatted files", ACM Transactions on Database Systems, Vol.2, No.1, pp.68-90, 1977. (13)
3. Anderson D.P., Osawa Y., Govidan R.: "A file system for continuous media", ACM Transactions on Computer Systems, Vol.10, No.4, pp.311-337, 1992. (2)
4. Aghili H., Severance D.G.: "A practical guide to the design of differential files for recovery of on-line databases", ACM Transactions on Database Systems, Vol.7, No.4, pp.540-565, 1982. (12)
5. Ammon G.J., Galabria J.A., Thomas D.T.: "A high-speed large capacity jukebox optical disk system", IEEE Computer, Vol.18, No.7, pp.36-45, 1985. (2)
6. Arnow D., Tenenbaum A.M.: "An empirical comparison of B-trees, Compact B-trees and Multiway trees", Proceedings ACM SIGMOD 84 Conference, pp.33-46, 1984. (8)
7. Asthana P., Finkelstein B.: "Superdense optical storage", IEEE Spectrum, Vol.32, No.8, pp.25-31, 1995. (2)
8. Babad J.M.: "A record and file partitioning model", Communications of the ACM, Vol.20, No.1, pp.22-31, 1977. (13)
9. Baeza-Yates R.A.: "Modeling splits in file structures", Acta Informatica, Vol.26, pp. 349-362, 1989. (8)
10. Baeza-Yates R.A.: "Expected behavior of B⁺-trees under random insertions", Acta Informatica, Vol.26, pp.439-471, 1989. (8)
11. Baeza-Yates R.A., Larson P.A.: "Performance of B⁺-trees with partial expansions", IEEE Transactions on Knowledge and Data Engineering, Vol.1, No.2, pp.248-257, 1989. (8)
12. Batory D.S.: "On searching transposed files", ACM Transactions on Database Systems, Vol.7, No.4, pp.531-544, 1979. (13)
13. Batory D.S.: "B⁺-trees and indexed sequential files - a performance comparison", Proceedings ACM SIGMOD 81 Conference, pp. 30-39, 1981. (7,8)
14. Bayer R., McCreight E.M.: "Organization and maintenance of large ordered indices", Acta Informatica, Vol.1, No.3, pp.173-189, 1972. (8)
15. Bayer R., McCreight E.M.: "Symmetric binary B-trees - data structures and maintenance algorithms", Acta Informatica, Vol.1, No.4, pp.209-306, 1972. (8)
16. Bayer R., Schkolnick M.: "Concurrency of operations on B-trees", Acta Informatica, Vol.9, No.1, pp.1-21, 1977. (15)
17. Bayer R., Unterauer K.: "Prefix B-trees", ACM Transactions on Database Systems, Vol.2, No.1, pp.11-26, 1977. (8,15)
18. Beausoleil W.F., Brown D.T., Phelps B.E.: "Magnetic bubble memory organization", IBM Journal of Research and Development, 1972. (2)

19. Becker B., Gschwind S., Ohler T., Seeger B., Widmayer P.: "An asymptotically optimal multiversion B-tree", *The VLDB Journal*, Vol.5, No.4, pp.264-275, 1996. (14)
20. Beckley D.A., Evans M.W., Raman V.K.: "Multikey retrieval from k-d trees and Quadrees", *Proceedings ACM SIGMOD 85 Conference*, pp.291-301, 1985. (11,14)
21. Beckman N., Kriegel H.P., Schneider R., Seeger B.: "The R*-tree - an efficient and robust access method for points and rectangles", *Proceedings ACM SIGMOD 90 Conference*, pp.322-331, 1990. (14)
22. Bell J.R., Kaman C.H.: "The linear quotient hash code", *Communications of the ACM*, Vol.13, No.11, pp.675-677, 1970. (9)
23. Bentley J.L.: "Multidimensional binary trees used for associative searching", *Communications of the ACM*, Vol.18, No.9, 509-517, 1975. (11)
24. Bentley J.L.: "How to sort", *Communications of the ACM*, Vol.22, No.4, pp.287-291, 1979. (5,6)
25. Bentley J.L., Friedman J.H.: "Data structures for range searching", *ACM Computing Surveys*, Vol.11, No.4, pp.397-409, 1979. (11)
26. Berchtold S., Keim D.A., Kriegel H.P.Q.: "The X-tree: an index structure for high-dimensional data", *Proceedings 22nd Conference on Very Large Data Bases*, pp.28-39, 1996. (14)
27. Bertino E., Kim W.: "Indexing techniques for queries on nested objects", *IEEE Transactions on Knowledge and Data Engineering*, Vol.1, No.2, pp.196-214, 1989. (15)
28. Betz B.K.: "Unpublished memorandum", Minneapolis, Honeywell Regulator Co., 1956. (4)
29. Betz B.K., Carter W.C.: "New merge sorting techniques", *Proceedings 14th ACM National Conference*, 1959. (4)
30. Blanken H., Ijbema A., Meek P., Akker B.V.D.: "The generalized grid file - description and performance aspects", *Proceedings 6th IEEE Data Engineering Conference*, pp.380-388, 1990. (11)
31. Biliris A.: "Operation-specific locking in balanced structures", *Information Sciences*, Vol.48, No.27-51, 1989. (15)
32. Bitton D., DeWitt D.J.: "Duplicate record elimination in large data files", *ACM Transactions on Database Systems*, Vol.8, No.2, pp.255-265, 1983. (1)
33. Bitton D., Gray J.: "Disk shadowing", *Proceedings 14th Conference on Very Large Data Bases*, pp.331-338, 1988. (2)
34. Bloome B.H.: "Space-time trade-offs in hash coding with allowable errors", *Communications of the ACM*, Vol.13, No.7, pp.422-426, 1970. (12)
35. Bourne C.P., Ford D.F.: "A study of methods for systematically abbreviating English words and names", *Journal of ACM*, Vol.8, No.4, 1961. (15)
36. Burge W.H.: "An analysis of the compromise merge sorting techniques", *Proceedings Information Processing Conference*, 1971. (4)

37. Burkhard W.A.: "Hashing and trie algorithms for partial match retrieval", *ACM Transactions on Database Systems*, Vol.1, No.2, pp.175-187, 1976. (9)
38. Burkhard W.A.: "Partial match hash coding - benefits of redundancy", *ACM Transactions on Database Systems*, Vol.4, No.2, pp.228-239, 1979. (9)
39. Burton W.F., Lewis G.N.: "A robust variation of interpolation search", *Information Processing Letters*, Vol.10, No.4, pp.198-201, 1980. (4)
40. Burton F.W., Huntbach M.W., Kollias J.: "Multiple generation text files using overlapping tree structures", *The Computer Journal*, Vol.28, No.4, pp.414-416, 1985. (14)
41. Burton W.F., Kollias G.J., Matsakis G.D.: "Extending the change area B-tree to cover multiple time records", *Proceedings Eurinfo Conference*, pp.769-774, 1988. (14)
42. Burton F.W., Kollias J.G., Kollias V.G., Matsakis D.G.: "Implementation of overlapping B-trees for time and space efficient representation of collection of similar files", *The Computer Journal*, Vol.33, No.3, pp.279-280, 1990. (14)
43. Cardenas A.F.: "Evaluation and selection of file organizations - a model and a system", *Communications of the ACM*, Vol.16, No.9, pp.540-548, 1973. (13)
44. Cardenas A.F.: "Analysis and performance of inverted database structures", *Communications of the ACM*, Vol.18, No.5, pp.253-263, 1975. (11)
45. Carey M., DeWitt D., Richardson J., Shekita E.: "Object and file management in the Exodus extensible database system", *Proceedings 12th Conference on Very Large Data Bases*, pp.91-100, 1986. (15)
46. Carey M., Haas L., Linvy M.: "Tapes hold data too - challenges of tuples on tertiary storage", *Proceedings ACM SIGMOD 93 Conference*, pp.413-417, 1993. (2)
47. Carg A.K., Gotlieb C.C.: "Order preserving key transformation", *ACM Transactions on Database Systems*, Vol.11, No.2, pp.213-234, 1986. (9)
48. Catania V., Puliafito A., Riccobene S., Vita L.: "Design and performance analysis of a disk array system", *IEEE Transactions on Computers*, Vol.44, No.10, pp.1236-1247, 1995. (2)
49. Cesarini F., Soda G.: "A dynamic hash method with signature", *ACM Transactions on Database Systems*, Vol.16, No.2, pp.309-337, 1991. (12)
50. Chang C.C.: "The study of an ordered minimal perfect hashing scheme", *Communications of the ACM*, Vol.27, No.4, pp.384-387, 1984. (9)
51. Chang J.M., Fu K.S.: "Extended k-d tree database organization - a dynamic multiattribute clustering method", *IEEE Transactions on Software Engineering*, Vol.7, No.3, pp.284-290, 1981. (11)
52. Chang J.W., Lee J.H., Lee Y.L.: "Multikey access method based on term discrimination and signature clustering", *Proceedings ACM SIGIR 89 Conference*, pp.176-185, 1989. (12)
53. Chauhundri S., Narasayya V.: "An efficient cost-driven index selection tool for Microsoft SQL server", *Proceedings 23rd Conference on Very Large Data Bases*, pp.146-155, 1997. (13)
54. Chen P.M., Lee E.K., Gibson G.A., Katz R.H., Patterson D.A.: "RAID - high-performance, reliable secondary storage", *ACM Computing Surveys*, Vol.26, No.2, pp.145-185, 1994. (2)

55. Chen W.C., Vitter J.S.: "Analysis of new variants of coalesced hashing", ACM Transactions on Database Systems, Vol.9, No.4, pp. 616-645, 1984. (8)
56. Chi C.S.: "Advances in computer mass storage technology", IEEE Computer, Vol.15, No.5, pp.60-74, 1982. (2)
57. Christodoulakis S.: "Implications of certain assumptions in database performance evaluation", ACM Transactions on Database Systems, Vol.9, No.2, pp.163-186, 1984. (13)
58. Christodoulakis S.: "Analysis of retrieval performance for records and objects using optical disk technology", ACM Transactions on Database Systems, Vol.12, No.2, pp.137-169, 1987. (2)
59. Christodoulakis S., Ford D.A.: "File organizations and access methods for CLV optical disks", Proceedings ACM SIGIR 89 Conference, pp.152-159, 1989. (14)
60. Christodoulakis S., Ford D.A.: "Retrieval performance vs. disc space utilization on WORM optical disks", Proceedings ACM SIGMOD 89 Conference, pp.306-314, 1989. (14)
61. Christodoulakis S., Manolopoulos Y., Larson P.A.: "Analysis of overflow handling for variable length records", Information Systems, Vol.14, No.2, pp.151-162, 1989. (8)
62. Christodoulakis S., Triantafyllou P., Zioga F.: "Principles of optimally placing data in tertiary storage libraries", Proceedings 23rd Conference on Very Large Data Bases, pp.236-245, 1997. (2)
63. Chou H.T., DeWitt D.: "An evaluation of buffer management strategies for relational database systems", Proceedings 11th Conference on Very Large Data Bases, pp.127-141, 1985. (3)
64. Cichelli R.J.: "Minimal perfect hash functions made simple", Communications of the ACM, Vol.23, No.1, pp.17-19, 1980. (8)
65. Claybrook B.G., Yang C.S.: "Efficient algorithms for answering queries with unsorted multilists", Information Systems, Vol.3, No.2, pp.93-97, 1978. (11)
66. Comer D.: "The difficulty of optimum index selection", ACM Transactions on Database Systems, Vol.3, No.4, pp.440-445, 1978. (12)
67. Comer D.: "The ubiquitous B-tree", ACM Computing Surveys, Vol. 11, No.2, pp.121-137, 1979. (8)
68. Davis I.J.: "Local correction of helix(k) lists", IEEE Transaction on Computers, Vol.38, No.5, pp.718-724, 1989. (13)
69. Denning D.E., Denning P.J.: "Data security", ACM Computing Surveys, Vol.1, No.3, pp.227-249, 1969. (15)
70. Deppisch U.: "S-tree - a dynamic balanced signature index for office retrieval", Proceedings ACM SIGIR 86 Conference, pp.77-87, 1986. (12)
71. Dervos D., Manolopoulos Y., Linardis P.: "Ranking the validity of block candidacies in signature files", Information Sciences, Vol.79, pp.89-108, 1994. (12)
72. Dervos D., Linardis P., Manolopoulos Y.: "Binary ranking for the signature file method", Information and Software Technology, Vol.36, No.3, pp.131-139, 1994. (12)

73. DeWitt D., Chou H.T., Katz R., Klug A.: "Design and implementation of the Wisconsin storage system", *Software - Practice and Experience*, Vol.15, No.10, pp.943-962, 1985. (2,3)
74. Diehr G., Faaland B.: "Optimal pagination of B-trees with variable length items", *Communications of the ACM*, Vol.27, No.3, pp. 241-247, 1984. (8)
75. Dobosiewicz W.: "Replacement selection in 3-level memories", *The Computer Journal*, Vol.27, No.4, pp.334-339, 1984. (6)
76. Dodd G.D.: "Elements of data management systems", *ACM Computing Surveys*, Vol.1, No.2, pp.117-133, 1969. (13)
77. Dodds D.I.: "Reducing directory size by using a hashing technique", *Communications of the ACM*, Vol.25, No.6, pp.368-370, 1982. (15)
78. Drapeau A.L., Katz R.H.: "Stripped tape arrays", *Proceedings 12th IEEE Symposium on Mass Storage Systems*, pp.257-265, 1993. (2)
79. Driscoll J.R., Lang S.D., Franklin L.A.: "Modeling B-trees insertion activity", *Information Processing Letters*, Vol.26, No.1, pp.5-18, 1987. (8)
80. Dumey A.I.: "Indexing for rapid random access memory", *Computers and Automation*, Vol.5, No.12, pp.6-9, 1956. (8)
81. Dwer B.: "One more time - how to update a master file", *Communications of the ACM*, Vol.24, No.1, 1981. (1)
82. Easton M.: "A streamlined statistical for a medical computer center", *Proceedings 24th ACM National Conference*, pp.494-475, 1969. (13)
83. Easton M.: "Key sequence data sets on indelible storage", *IBM Journal on Research and Development*, Vol.30, No.3, pp.230-241, 1986. (14)
84. Effelsberger W., Haerder T.: "Principles of database buffer management", *ACM Transactions on Database Systems*, Vol.9, No.4, pp.560-595, 1984. (3)
85. Eisenbarth B., Ziviani N., Gonnet G.H., Mehlhorn K., Wood D.: "The theory of fringe analysis and its applications to 2-3 trees and B-trees", *Information and Control*, Vol.55, pp.125-174, 1982. (8,15)
86. Ellis C.: "Concurrency in linear hashing", *ACM Transactions on Database Systems*, Vol.12, No.2, pp.195-207, 1987. (15)
87. Elmasri R., Jaseemuddin M., Kouramajian V.: "Partition of Time index for optical disks", *Proceedings 8th IEEE Conference on Data Engineering*, pp.574-583, 1992. (14)
88. Elmasri R., Kim Y.J., Wu G.T.J.: "Efficient implementation techniques for the Time Index", *Proceedings 7th IEEE Conference on Data Engineering*, pp.102-111, 1991. (14)
89. Elmasri G., Wu G.T.J., Kim Y.J.: "The Time Index - an access structure for temporal data", *Proceedings 16th Conference on Very Large Data Bases*, pp.1-12, 1990. (14)
90. Enbody R.J., Du H.C.: "Dynamic hashing schemes", *ACM Computing Surveys*, Vol.20, No.2, pp.85-113, 1988. (10)
91. Ershov A.P.: "On programming arithmetic operations", *Communications of the ACM*, Vol.1, No.8, pp.3-6, 1958. (9)

92. Fagin R., Nievergelt J., Pippenger N., Strong H.R.: "Extendible hashing - a fast method for dynamic files", *ACM Transaction on Database Systems*, Vol.4, No.3, pp.315-344, 1979. (10)
93. Faloutsos C., Christodoulakis S.: "Signature files - an access method for documents and its analytical performance evaluation", *ACM Transactions on Office Information Systems*, Vol.2, No.4, pp. 267-288, 1984. (12)
94. Faloutsos C.: "Access methods for text", *ACM Computing Surveys*, Vol.17, No.3, pp.49-74, 1985. (12)
95. Faloutsos C.: "Signature files - design and performance comparison of some signature extraction methods", *Proceedings ACM SIGMOD 85 Conference*, pp.63-82, 1985. (12)
96. Faloutsos C., Jagadish H.: "On B-tree indices for skew distributions", *Proceedings 18th Conference on Very Large Data Bases*, pp.363-374, 1992. (8)
97. Faloutsos C., Kamel I.: "Beyond uniformity and independence - analysis of R-trees using the concept of fractal dimension", *Proceedings ACM PODS 94 Conference*, pp.4-13, 1994. (14)
98. Faloutsos C., Jagadish H.V., Manolopoulos Y.: "Analysis of n-dimensional quadtree decomposition of arbitrary rectangles", *IEEE Transactions on Knowledge and Data Engineering*, Vol.9, No.3, pp.373-383, 1997. (14)
99. Finkel R.A., Bentley J.L.: "Quadrees - a data structure for retrieval on composite keys", *Acta Informatica*, Vol.4, No.1, pp.1-9, 1974 (14)
100. Flajolet P.: "On the performance evaluation of extendible hashing and trie hashing", *Acta Informatica*, Vol.20, pp.345-369, 1983. (10)
101. Flores I., Madpis G.: "Average binary search length for dense ordered lists", *Communications of the ACM*, Vol.14, No.9, pp.602-603, 1971. (4)
102. Floyd R.W.: "Algorithm 245", *Communications of the ACM*, Vol.7, No.12, pp.701, 1964. (6)
103. Ford D.A., Christodoulakis S.: "Optimal placement of high-probability randomly retrieved blocks on CLV optical disks", *ACM Transactions on Information Systems*, Vol.19, No.1, pp.1-30, 1991. (2,13)
104. Freese R.: "Optical disks become erasable", *IEEE Spectrum*, Vol.25, No.2, pp.41-45, 1988. (2)
105. Freeston M.: "The bang file - a new kind of grid file", *Proceedings ACM SIGMOD 87 Conference*, pp.260-269, 1987. (11)
106. French J.C.: "IDAM file organizations", *UMI Research Press*, 1985. (12)
107. Friedman M.B.: "RAID keeps going and going and ...", *IEEE Spectrum*, Vol.33, No.4, pp.73-79, 1996. (2)
108. Fujitani L.: "Laser optical disk - the coming revolution in on-line storage", *Communications of the ACM*, Vol.27, No.6, pp.546-554, 1984. (2)
109. Gaede V., Guenther O.: "Multidimensional Access Methods", *ACM Computer Surveys*, Vol.30, No.2, pp.170-231, 1998. (14)

110. Gairola B.K., Rajaraman V.: "A distributed index sequential access method", *Information Processing Letters*, Vol.5, pp.1-5, 1976. (7)
111. Gait J.: "The optical file cabinet - a random-access file system for write-once optical disks", *IEEE Computer*, Vol.21, No.6, pp.11-22, 1988. (2)
112. Garg A., Gotlieb C.: "Order preserving key transformations", *ACM Transactions on Database Systems*, Vol.11, No.2, pp.213-234, 1986. (8)
113. Gargantini I.: "An Effective Way to Represent Quadrees", *Communications of the ACM*, Vol.25, No.12, pp.905-910, 1982. (14)
114. Gassner B.J.: "Sorting by replacement selecting", *Communications of the ACM*, Vol.10, No.2, pp.89-93, 1967. (6)
115. Ghosh S.P., Senko M.E.: "File organization - on the selection of random access index points for sequential files", *Journal of the ACM*, Vol.16, No.4, pp.569-579, 1969. (13)
116. Gilstad R.L.: "Polyphase merge sorting - an advanced technique", *Proceedings AFIPS Eastern Joint Computer Conference*, Vol.18, pp.143-148, 1960. (5)
117. Gilstad R.L.: "Read-backwards polyphase sorting", *Communications of the ACM*, Vol.6, No.5, pp.220-223, 1963. (5)
118. Goetz M.A.: "Internal and tape sorting using the replacement selection technique", *Communications of the ACM*, Vol.6, No.5, pp.201-206, 1963. (5)
119. Goetz M.A., Toth G.S.: "A comparison between the polyphase and oscillating sort techniques", *Communications of the ACM*, Vol. 6, No.5, pp.223-225, 1963. (5)
120. Gonnet G.H., Rogers L.D., George A.: "An algorithmic and complexity analysis of interpolation search", *Acta Informatica*, Vol. 13, No.1, pp.39-46, 1980. (4)
121. Gonnet G.H., Larson P.A.: "External hashing with limited internal storage", *Journal of the ACM*, Vol.35, No.1, pp.161-184, 1988. (12)
122. Grazzini E., Pippolini F.: "Performance evaluation of shared and separate inverted files", *BIT*, Vol.29, pp.561-565, 1989. (11)
123. Greene D.: "An implementation and performance analysis on spatial data access methods", *Proceedings 5th IEEE Data Engineering Conference*, pp.606-615, 1989. (14)
124. Gremillion L.L.: "Designing a Bloome filter for differential file access", *Communications of the ACM*, Vol.25, No.9, pp.600-604, 1982. (12)
125. Griffiths P.P., Wade B.W.: "An authorization mechanism for relational database systems", *ACM Computing Surveys*, Vol.1, No. 3, pp.242-255, 1976. (15)
126. Gueting R.H.: "An introduction to spatial database systems", *The VLDB Journal*, Vol.3, No.4, pp.357-399, 1994. (14)
127. Guibas L.J.: "The analysis of hashing techniques that exhibit a k-ary clustering", *Journal of the ACM*, Vol.25, No.4, pp.544-555, 1978. (9)
128. Guibas L.J., Sedgewick R.: "A dichromatic framework for balanced trees", *Proceedings 19th IEEE Symposium on Foundations of Computer Science*, pp.8-21, 1978. (8,15)
129. Gupta G.K., Srinivasan B.: "Approximate storage utilization of B-trees", *Information Processing Letters*, Vol.22, pp.243-246, 1986. (8)

130. Guttman A.: "R-trees - a dynamic index structure for spatial searching", Proceedings ACM SIGMOD 84 Conference, pp.47-57, 1984. (14)
131. Hahn B.A.: "A new technique for compression and storage of data", Communications of the ACM, Vol.17, No.8, pp.434-436, 1974. (15)
132. Hakola J., Heiskanen A.: "On the distribution of the wasted space at the end of file blocks", BIT, Vol.20, No.2, pp.145-156, 1980. (3)
133. Halatsis C., Philokyprou G.: "Pseudochaining in hash tables", Communications of the ACM, Vol.21, No.7, pp.554-557, 1978. (9)
134. Hamming R.W.: "Error detecting and correcting codes", Bell System Technical Journal, Vol.29, pp.147-160, 1950. (2,15)
135. Hansen W.J.: "A cost model for the internal organization of B⁺-tree nodes", ACM Transactions on Programming Languages and Systems, Vol.3, No.4, pp.508-532, 1981. (8)
136. Hatzopoulos M., Kollias J.G.: "Some rules for introducing indexing paths in a primary file", The Computer Journal, Vol.23, pp.207-211, 1980. (13)
137. Hatzopoulos M., Kollias J.G.: "The determination of the optimum database maintenance points", The Computer Journal, Vol.25, No.1, pp.126-129, 1982. (13)
138. Hatzopoulos M., Kollias J.G.: "On the optimal selection of multilist database structures", IEEE Transactions on Database Engineering, Vol.10, No.6, pp.681-687, 1984. (11)
139. Hatzopoulos M., Kollias J.G.: "On the selection of a reduced set of indexes", The Computer Journal, Vol.28, No.4, pp.406-408, 1985. (13)
140. Heising W.P.: "Note on random addressing techniques", IBM Systems Journal, Vol.2, No.2, pp.112-116, 1963. (8)
141. Held G.D., Stonebraker M.R.: "B-trees re-examined", Communications of the ACM, Vol.21, No.2, pp.139-143, 1978. (7)
142. Hellerstein J., Naughton J., Pfeffer A.: "Generalized search trees for database systems", Proceedings 21th Conference on Very Large Data Bases, pp.562-573, 1995. (15)
143. Henrich A., Six H.W., Widmayer P.: "The LSD-tree: spatial access to multidimensional point and non-point objects", Proceedings 15th Conference on Very Large Data Bases, pp.45-53, 1989. (14)
144. Hester J.H., Hirschberg D.S.: "Self-organizing linear search", ACM Computing Surveys, Vol.17, No.3, pp.295-311, 1985. (3)
145. Hillyer B.K., Silberchatz A.: "On the modeling and performance characteristics of a serpentine tape drive", Proceedings ACM SIGMETRICS 96 Conference, pp.170-179, 1996. (2)
146. Hinrichs K.: "Implementation of the grid file - design concepts and experience", BIT, pp.569-592, 1985. (11)
147. Hoare C.A.R.: "Quicksort", The Computer Journal, Vol.5, No.4, pp. 10-15, 1962. (5)

148. Hoffman L.J.: "Computers and privacy - a survey", ACM Computing Surveys, Vol.1, No.2, pp.85-103, 1969. (15)
149. Hsiao D.K., Harary F.D.: "A formal system for information retrieval from files", Communications of the ACM, Vol.13, No.2, pp.67-73, 1970. (13)
150. Hu T.C., Wachs M.L.: "Binary search on a tape", SIAM Journal on Computing, Vol.16, No.3, pp.573-590, 1987. (3)
151. Huang B.C., Langston M.A.: "Stable duplicate extraction with optimal time and space bounds", Acta Informatica, Vol.26, pp.473-484, 1989. (1)
152. Huang S.H.S.: "Height-balanced trees of order(β, γ, δ)", ACM Transactions on Database Systems, Vol.10, No.2, pp.261-284, 1985. (8)
153. Huffman D.A.: "A method for the construction of minimum redundancy codes", Proceedings IRE, Vol.40, pp.1098-1101, 1952. (15)
154. Hutflesz A., Six H.W., Widmayer P.: "Twin grid files - space optimizing access schemes", Proceedings ACM SIGMOD 88 Conference, pp.183-190, 1988. (11)
155. Inglis J.: "Updating a master file - yet one more time", Communications of the ACM, Vol.24, No.5, p.299, 1981. (1)
156. Janko W.: "Variable jump search - the algorithm and its efficiency", Angewandte Informatik, Vol.23, No.1, pp.6-11, 1981. (4)
157. Johnson T.: "An analytical performance model of robotic storage libraries", Performance Evaluation, Vol.27/28, No.4, pp.231-251, 1996. (2)
158. Jaeschke G.: "Reciprocal hashing - a method for generating minimal perfect hashing functions", Communications of the ACM, Vol.24, No.12, pp.829-833, 1981. (9)
159. Jensen C.S., Clifford J., Elmasri R., Gadia S., Hayes P., Jajodia S., Dyreson S., Grandi F., Kafer W., Kline N., Lorentzos N., Mitsopoulos Y., Montanari A., Nonen D., Peressi E., Pernici B., Roddick J., Sarda N., Scalas M., Segev A., Snodgrass R., Soo M., Tansel A., Tiberio P., Wiederhold G.: A Consensus Glossary of Temporal Database Concepts, ACM SIGMOD Record, Vol.23, No.1, pp.52-64, 1994. (14)
160. Johnson T., Shasha D.: "Utilization of B-trees with inserts, deletes and modifies", Proceedings ACM PODS 89 Conference, pp.235-246, 1989. (8,13)
161. Kamel I., Faloutsos C.: "On packing R-trees", Proceedings 2nd Conference on Information and Knowledge Management, pp.490-499, 1993. (14)
162. Kamel I., Faloutsos C.: "Hilbert R-tree: an improved R-tree using fractals", Proceedings 20th Conference on Very Large Data Bases, pp.500-509, 1996. (14)
163. Kawagoe K.: "Modified dynamic hashing", Proceedings ACM SIGMOD 85 Conference, pp.201-213, 1985. (10)
164. Keen D., Lacy J.: "VSAM data and design parameters", IBM Systems Journal, Vol.5, pp.186-212, 1974. (7)
165. Kjelleberg P., Zahle T.U.: "Cascade hashing", Proceedings 10th Conference on Very Large Data Bases, pp.481-492, 1984. (10)
166. Klinger A., Rhodes M.L.: "Organization and access of image data by areas", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.1, No.1, pp.50-60, 1979. (14)

167. Knott G.D.: "Hashing functions", *The Computer Journal*, Vol.18, No.3, pp.265-278, 1975. (9)
168. Kollias J.G.: "The selection of secondary file organizations", *Management Datamatics*, Vol.5, No.6, pp.241-250, 1976. (11)
169. Kollias J.G.: "A heuristic approach for determining the optimal degree of file inversion", *Information Systems*, Vol.4, No.3, pp.307-318, 1979. (11)
170. Kollias J.G.: "File organizations and their reorganization", *Information Systems*, Vol.4, No.1, pp.49-54, 1979. (13)
171. Kollias V.G., Hatzopoulos M., Kollias J.G.: "Database maintenance efficiency using differential files", *Information Systems*, Vol.5, No.3, pp.319-321, 1980. (13)
172. Kolovson C. and Stonebraker M.: "Indexing techniques for historical databases", *Proceedings 5th IEEE Data Engineering Conference*, pp.127-137, 1989. (14)
173. Kouramajian V., Kamel I., Elmasri E., Waheed S.: "The Time Index⁺ - an incremental access structure for temporal databases", *Proceedings 3rd Conference on Information and Knowledge Management*, pp.296-303, 1994. (14)
174. Kouvatso D.D., Wong S.K.: "On optimal blocking of sequential files", *The Computer Journal*, Vol.27, No.4, pp.321-327, 1984. (3,4)
175. Kumar A., Tsotras V.J., Faloutsos C.: "Designing access methods for bi-temporal databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol.10, No.1, pp.1-20, 1998. (14)
176. Kung H., Lehman P.: "Concurrent manipulation of binary search trees", *ACM Transactions on Database Systems*, Vol.5, No.3, pp.354-382, 1980. (15)
177. Kwong Y.S., Wood D.: "Method for concurrency in B⁺-trees", *IEEE Transactions on Software Engineering*, Vol.6, No.2, pp.211-223, 1982. (15)
178. Landau G.M., Schmidt J.P., Tsotras V.J.: "On historical queries along multiple lines of time evolution", *The VLDB Journal*, Vol.4, No.4, pp.703-726, 1995. (14)
179. Lang S.D., Driscoll J.R., Jou J.H.: "Batch insertions for tree structured file organizations - improving differential database representation", *Information Systems*, Vol.11, No.2, pp.167-175, 1986. (13)
180. Lang S.D., Driscoll J.R., Jou J.H.: "A unified analysis of batched searching of sequential and tree-structured files", *ACM Transactions on Database Systems*, Vol.14, No.4, pp.604-618, 1989. (8,13)
181. Lang S.D., Manolopoulos Y.: "Efficient expressions for partly and completely unsuccessful batched search of tree-structured files", *IEEE Transactions on Software Engineering*, Vol.16, No.12, pp.1433-1435, 1990. (8,13)
182. Lanka S., Mays E.: "Fully persistent B⁺-trees", *Proceedings ACM SIGMOD 91 Conference*, pp.436-445, 1991. (8,14)
183. Larmore L.L., Hirschberg D.S.: "Efficient optimal pagination of scrolls", *Communications of the ACM*, Vol.28, No.8, pp.854-856, 1985. (8)
184. Larson P.A.: "Dynamic hashing", *BIT*, Vol.18, pp.184-201, 1978. (10)
185. Larson P.A.: "Linear hashing with partial expansions", *Proceedings 6th Conference on Very Large Data Bases*, pp.224-232, 1980. (10)

186. Larson P.A.: "Analysis of index sequential files with overflow chaining", *ACM Transactions on Database Systems*, Vol.6, No.4, pp.671-680, 1981. (7)
187. Larson P.A.: "Performance analysis of linear hashing with partial expansions", *ACM Transactions on Database Systems*, Vol.7, No.4, pp.566-587, 1982. (10)
188. Larson P.A.: "Analysis of uniform hashing", *Journal of the ACM*, Vol.30, No.4, pp.805-819, 1983. (9)
189. Larson P.A.: "Analysis of hashing with chaining in the prime area", *Journal of Algorithms*, Vol.5, pp.36-47, 1984. (9)
190. Larson P.A., Kajla A.: "File organization - implementation of a method guaranteeing retrieval in one access", *Communications of the ACM*, Vol.27, No.7, pp.670-677, 1984. (12)
191. Larson P.A., Ramakrishna M.V.: "External perfect hashing", *Proceedings ACM SIGMOD 85 Conference*, pp.190-200, 1985. (12)
192. Larson P.A.: "Linear hashing with overflow handling by linear probing", *ACM Transactions on Database Systems*, Vol.10, No.1, pp.75-89, 1985. (10)
193. Larson P.A.: "Linear hashing with separators - a dynamic hashing scheme with one access retrieval", *ACM Transactions on Database Systems*, Vol.13, No.3, pp.366-388, 1988. (12)
194. Larson P.A.: "Dynamic hash tables", *Communications of the ACM*, Vol.31, No.4, pp.446-457, 1988. (10)
195. Lee D.T., Wong C.K.: "Quintary trees - a file structure of multidimensional database systems", *ACM Transactions on Database Systems*, Vol.5, No.3, pp.339-353, 1980. (11)
196. Lehman P.L., Yao S.B.: "Efficient locking for concurrent operations on B-trees", *ACM Transactions on Database Systems*, Vol.6, No.4, pp.650-670, 1981. (15)
197. Leipala: "On the design of one-level indexed sequential files", *International Journal of Computer and Information Sciences*, Vol.10, No. 3, pp.177-186, 1981. (7)
198. Leipala: "On optimal multilevel indexed sequential files", *Information Processing Letters*, Vol.15, No.5, pp.191-195, 1982. (7)
199. Lempel A.: "Cryptology in transition", *ACM Computing Surveys*, Vol.11, No.4, pp.285-303, 1979. (15)
200. Lessuire R.: "Some lessons drawn from the history of binary search algorithm", *The Computer Journal*, Vol.26, No.2, pp.154-163, 1983. (4)
201. Leung C.H.C.: "Approximate storage utilization of B-trees - a simple derivation and generalizations", *Information Processing Letters*, Vol. 19, pp.199-201, 1984. (4)
202. Leung C.H.C., Wolfenden K.: "Mathematical models of file growth", *The Computer Journal*, Vol.28, No.2, pp.179-184, 1985. (13)
203. Levy M.R.: "Modularity and the sequential file update problem", *Communications of the ACM*, Vol.25, No.6, pp.362-367, 1982. (1)
204. Linden T.A.: "Operating system structures to support security and reliable software", *ACM Computing Surveys*, Vol.8, No.4, pp.409-455, 1976. (15)

205. Litwin W.: "Virtual hashing - a dynamically changing hashing", Proceedings 4th Conference on Very Large Data Bases, pp.517-523, 1978. (10)
206. Litwin W.: "Linear hashing - a new tool for file and table addressing", Proceedings 6th Conference on Very Large Data Bases, pp.212-223, 1980. (10)
207. Litwin W.: "Trie hashing", Proceedings ACM SIGMOD 81 Conference, pp.19-29, 1981. (10)
208. Lomet D.: "Digital B-trees", Proceedings 7th Conference on Very Large Data Bases, pp.333-344, 1981. (10)
209. Litwin W., Lomet D.: "The bounded disorder access method", Proceedings 2nd IEEE Data Engineering Conference, pp.38-48, 1986. (10)
210. Litwin W., Lomet D.: "A new method for fast data searches with keys", IEEE Software, Vol.4, No.2, pp.16-24, 1987. (10)
211. Lomet D.: "Bounded index exponential hashing", ACM Transactions on Database Systems, Vol.8, No.1, pp.136-165, 1983. (10)
212. Lomet D.: "Partial expansions for file organizations with an index", ACM Transactions on Database Systems, Vol.12, No.1, pp.65-84, 1987. (10)
213. Lomet D.: "A simple bounded disorder file organization with good performance", ACM Transactions on Database Systems, Vol.13, No.4, pp.525-551, 1988. (10)
214. Lomet D., Salzberg B.: "Access methods for multi-version data", Proceedings ACM SIGMOD 89 Conference, pp.315-324, 1989. (14)
215. Lomet D., Salzberg B.: "The performance of a multi-version access method", Proceedings ACM SIGMOD 90 Conference, pp.353-363, 1990. (14)
216. Lomet D., Salzberg D.: "The hB-tree - a multiattribute indexing method with good guaranteed performance", ACM Transactions on Database Systems, Vol.15, No.4, 1990. (11,14)
217. Lubel P.: "The gathering storm of high-density compact disks", IEEE Spectrum, Vol.32, No.8, pp.32-37, 1995. (2)
218. Lum V.Y.: "Multiattribute retrieval with combined indexes", Communications of the ACM, Vol.13, No.11, pp., 1970. (11)
219. Lum V.Y., Yuen P.S.T., Dodd M.: "Key-to-address transformation techniques - a fundamental performance study on large existing formatted files", Communications of the ACM, Vol.14, No.4, pp.228-239, 1971. (9)
220. Lum V.Y., Yuen P.S.T.: "Additional results on key-to-address transform techniques - a fundamental performance study on large existing formatted files", Communications of the ACM, Vol.15, No. 11, pp.996-997, 1972. (9)
221. Lum V.Y.: "General performance analysis of key-to-address transformation models", Communications of the ACM, Vol.16, No.10, pp. 603-612, 1973. (9)
222. McCreight E.M.: "Pagination of B*trees with variable length records", Communications of the ACM, Vol.20, No., pp.670-674, 1977. (8)
223. Manber U., Ladner R.E.: "Concurrency control in a dynamic search structure", ACM Transactions on Database Systems, Vol.9, No.3, pp.439-455, 1984. (15)

224. Manolopoulos Y., Kapetanakis G.: "Overlapping B⁺-trees for temporal data", Proceedings 5th Jerusalem Conference on Information Technology, pp.491-499, 1990. (14)
225. Manolopoulos Y.: "Batched search of index sequential files", Information Processing Letters, Vol.22, pp.267-272, 1986. (8,13)
226. Manolopoulos Y., Kollias J.G., Hatzopoulos M.: "Binary vs. sequential batched search", The Computer Journal, Vol.29, No.4, pp.368-372, 1986. (4)
227. Manolopoulos Y., Kleftouris D., Petrou L.: "A model for an ISAM file with multiple overflow chains", The Computer Journal, Vol.30, No.6, pp.529-534, 1987. (7)
228. Manolopoulos Y., Kollias J.G., Burton F.W.: "Batched interpolation search", The Computer Journal, Vol.30, No.6, pp.565-568, 1987. (4)
229. Manolopoulos Y., Kollias J.G.: "Estimating disk head movements in batched searching", BIT, Vol.28, pp.27-36, 1988. (2)
230. Manolopoulos Y., Kollias J.G.: "Expressions for partly and completely unsuccessful batched search of sequential and tree-structured files", IEEE Transactions on Software Engineering, Vol.15, No.6, pp.794-799, 1989. (8,13)
231. Manolopoulos Y., Poulakas G.: "An adaptation of a root finding method to searching ordered disk files revisited", BIT, Vol.29, 364-368, 1989. (4)
232. Manolopoulos Y., Kollias J.G.: "Performance of a two-headed disk system when serving database queries under the scan policy", ACM Transactions on Database Systems, Vol.14, No.3, pp.425-442, 1989. (2)
233. Manolopoulos Y., Faloutsos C.: "Analysis for the end of block wasted space", BIT, Vol.30, pp.620-630, 1990. (3)
234. Manolopoulos Y., Vakali A.: "Seek distances in disk systems with two independent heads per surface", Information Processing Letters, Vol.37, No.1, pp.37-42, 1991. (2)
235. Manolopoulos Y.: "Probability distributions for seek time evaluation", Information Sciences, Vol.60, No.12, pp.29-40, 1992. (2)
236. Manolopoulos Y., Fistas N.: "Algorithms for hash based files with variable length records", Information Sciences, Vol.63, No.3, pp.229-243, 1992. (9)
237. Manolopoulos Y., Christodoulakis S.: "File organizations with shared overflow blocks for variable length objects", Information Systems, Vol.17, No.6, pp.491-509, 1992. (9)
238. Manolopoulos Y.: "B-trees with lazy parent split", Information Sciences, Vol.79, pp.73-88, 1994. (8,15)
239. Manolopoulos Y., Lorentzos N.: "Performance of linear hashing schemes for primary key retrieval", Information Systems, Vol.19, No.5, pp.433-446, 1994. (10)
240. Manolopoulos Y.: "Seek time evaluation", Lemma in Encyclopedia of Microcomputers, Marcel Dekker Inc., 1995. (2)
241. Manolopoulos Y.: "Tree structures", Lemma in Encyclopedia of Computer Science and Technology, Marcel Dekker Inc., 1996. (8)
242. Manolopoulos Y., Nardelli E., Proietti G., Vassilakopoulos M.: "On creating random quadtrees by using a branching process", Image and Vision Computing, Vol.14, No.2, pp.159-164, 1996. (14)

243. Manolopoulos Y., Nardelli E., Papadopoulos A., Proietti G.: "MOF-tree: a spatial data structure to manipulate multiple overlapping features", *Information Systems*, Vol.22, No.8, pp.465-481, 1997. (14)
244. March S.T.: "Techniques for structuring database records", *ACM Computing Surveys*, Vol.15, No.1, pp.45-79, 1983. (13)
245. March S.T., Scudder G.D.: "On the selection of efficient record and backup strategies for large shared databases", *ACM Transactions on Database Systems*, Vol.9, No.3, pp.409-438, 1984. (13)
246. March S.T., Severance D.: "The determination of efficient record segmentations and blocking factors for shared files", *ACM Transactions on Database Systems*, Vol.2, No.3, pp.279-296, 1977. (13)
247. Martin W.A.: "Sorting", *ACM Computing Surveys*, Vol.3, No.12, pp. 147-174, 1971. (5,6)
248. Maruyama K., Smith S.E.: "Optimal reorganization of distributed space disk files", *Communications of the ACM*, Vol.19, No.11, pp.634-642, 1976. (13)
249. Maurer W.D., Lewis T.G.: "Hash table methods", *ACM Computing Surveys*, Vol.7, No.1, pp.5-20, 1975. (9)
250. Mendelson H.: "Analysis of extendible hashing", *IEEE Transactions on Software Engineering*, Vol.8, No.6, pp.611-619, 1982. (10)
251. Michaels P.C., Richards W.J.: "Magnetic bubble mass memory", *IEEE Transactions on Magnetics*, Vol.11, No.1, pp.21-25, 1975. (2)
252. Mitchell R.W.: "Content addressable filestore", *Proceedings Online Database Technology Conference*, England, 1976. (2)
253. Mohan C., Narang I.: "Algorithms for creating indexes for very large tables without quiescing updates", *Proceedings ACM SIGMOD 97 Conference*, pp.361-370, 1992. (8,13)
254. Mooers C.: "Applications of random codes to the gathering of statistical information", *Bulletin 31*, Zator Co., Cambridge, Ma, 1949. (12)
255. Morris R.: "Scatter storage techniques", *Communications of the ACM*, Vol.11, No.1, pp.38-44, 1968. (8)
256. Mullin J.K.: "Retrieval-update speed trade-offs using combined indexes", *Communications of the ACM*, Vol.14, No.12, pp.775-776, 1971. (11)
257. Mullin J.K.: "An improved index sequential access method using hashed overflow", *Communications of the ACM*, Vol.15, No.5, pp. 301-307, 1972. (7)
258. Mullin J.K.: "Unified dynamic hashing", *Proceedings 10th Conference on Very Large Data Bases*, pp.473-480, 1980. (10)
259. Mullin J.K.: "Change area B-trees - a technique to aid error recovery", *The Computer Journal*, Vol.24, No.4, pp.367-373, 1981. (12,13)
260. Mullin J.K.: "Tightly controlled linear hashing without separate overflow storage", *BIT*, Vol.21, pp.390-400, 1981. (10)
261. Mullin J.K.: "A second look at Bloome filters", *Communications of the ACM*, Vol.26, No.8, pp.570-571, 1983. (12)

262. Murayama K., Smith S.E.: "Analysis of design alternatives for virtual memory indexes", *Communications of the ACM*, Vol.20, No.4, pp.245-254, 1977. (7,8)
263. Nascimento M.A., Silva J.R.O.: "Towards historical R-trees", *Proceedings ACM Symposium on Applied Computing*, pp.235-240, 1998. (14)
264. Nievergelt J.: "Binary search trees and file organization", *ACM Computing Surveys*, Vol.6, No.3, pp.195-207, 1974. (8)
265. Nievergelt J., Hinterberger H., Sevcik K.C.: "The grid file - an adaptable symmetric, multikey file structure", *ACM Transactions on Database Systems*, Vol.9, No.1, pp.38-71, 1984. (11)
266. Nishihara S., Ikeda K.: "Reducing the retrieval time of hashing method using predictors", *Communications of the ACM*, Vol.26, No. 12, pp.1082-1088, 1983. (9)
267. Orenstein J.A.: "Multidimensional tries used for associative searching", *Information Processing Letters*, Vol.14, No.4, pp.150-157, 1982. (11)
268. Ouksel M., Scheuermann P.: "Multidimensional B-trees - analysis of dynamic behavior", *BIT*, Vol.21, No.4, pp.401-418, 1981. (11)
269. Palvia P.: "Expressions for batched searching of sequential and hierarchical files", *ACM Transactions on Database Systems*, Vol.10, No.1, pp.97-106, 1985. (8,13)
270. Papadimitriou C.H., Bernstein P.A.: "On the performance of balanced hashing functions when keys are not equiprobable", *ACM Transactions on Programming Languages and Systems*, Vol.2, No.1, pp.77-89, 1980. (9)
271. Papadopoulos A.N., Manolopoulos Y.: "Performance of nearest neighbor queries in R-trees", *Proceedings 6th International Conference on Database Theory*, pp.394-408, 1997. (14)
272. Patterson D.A., Gibson G.A., Katz R.H.: "A case for redundant arrays of inexpensive disks (RAID)", *Proceedings ACM SIGMOD 88 Conference*, pp.109-125, 1988. (2)
273. Pechura M.A., Schoeffler J.D.: "Estimating file access time floppy disks", *Communications of the ACM*, Vol.26, No.10, pp.754-763, 1983. (2)
274. Perl Y., Itai A., Avni H.: "Interpolation search - a $\log \log N$ search", *Communications of the ACM*, Vol.21, No.7, pp.550-553, 1978. (4)
275. Peterson W.W.: "Addressing for random-access storage", *IBM Journal on Research and Development*, Vol.1, No.4, pp.130-146, 1957. (9)
276. Pfaltz J.L., Berman W.J., Cagley E.M.: "Partial match retrieval using indexed descriptor files", *Communications of the ACM*, Vol.23, No.9, pp.522-528, 1980. (12)
277. Pike J.: "Text compression using a 4-bit coding scheme", *The Computer Journal*, Vol.24, No.4, 1981. (15)
278. Piwowarski M.: "Comments on batched searching of sequential and tree structured files", *ACM Transactions on Database Systems*, Vol.10, No.2, pp.285-287, 1985. (8,13)
279. Price C.E.: "Table lookup techniques", *ACM Computing Surveys*, Vol.3, No.2, pp.49-65, 1971. (8)
280. Quitzow K.H., Klopprogge M.R.: "Space utilization and access path length in B-trees", *Information Systems*, Vol.5, pp.7-16, 1980. (8)

281. Rathman P.: "Dynamic data structure on optical disks", Proceedings 1st IEEE Data Engineering Conference, pp.175-180, 1984. (14)
282. Ramakrishna M.V.: "An exact probability model for finite hash tables", Proceedings 4th IEEE Data Engineering Conference, pp.362-368, 1988. (9)
283. Ramakrishna M.V., Larson P.A.: "File organizations using composite perfect hashing", ACM Transactions on Database Systems, Vol.14, No.2, pp.231-263, 1989. (10,12)
284. Ramamohanarao K., Loyd J.W.: "Dynamic hashing schemes", The Computer Journal, Vol.25, No.4, 1982. (10)
285. Ramamohanarao K., Loyd J.W., Thom J.A.: "Partial match retrieval using hashing and descriptors", ACM Transactions on Database Systems, Vol.8, No.4, pp.552-675, 1983. (12)
286. Ramamohanarao K., Sacks-Davis R.: "Recursive linear hashing", ACM Transactions on Database Systems, Vol.9, No.3, pp.369-391, 1984. (10)
287. Ramamohanarao K., James A.T.: "Partial match retrieval using recursive linear hashing", BIT, Vol.25, No.3, pp.477-484, 1985. (10)
288. Ramamohanarao K., Shepherd J., Sacks-Davis R.: "Partial match retrieval for dynamic files using linear hashing with partial expansions", Proceedings Conference on Foundations of Data Organizations and Algorithms, pp.202-206, 1989. (10)
289. Regnier M.: "Analysis of the grid file algorithms", BIT, pp.335-357, 1985. (11)
290. Reuter A.: "Performance analysis of recovery techniques", ACM Transactions on Database Systems, Vol.9, No.4, pp.526-559, 1984. (15)
291. Roberts C.S.: "Partial match retrieval via the method of superimposed coding", Proceedings of the IEEE, Vol.67, No.12, pp.1624-1642, 1979. (112)
292. Robinson J.T.: "The k-d B-tree - a search structure for large multidimensional dynamic indexes", Proceedings ACM SIGMOD 81 Conference, pp.10-18, 1981. (11)
293. Rosenberg A.L., Snyder L.: "Time and space optimality in B-trees", ACM Transactions in Database Systems, Vol.6, No.1, pp.174-183, 1981. (8)
294. Roussopoulos N. and Leifker D.: "Direct spatial search on pictorial databases using packed R-trees", Proceedings ACM SIGMOD 85 Conference, pp.17-31, 1985. (14)
295. Roussopoulos N., Kelley S., Vincent F.: "Nearest neighbor queries", Proceedings ACM SIGMOD 95 Conference, pp.71-79, 1995. (14)
296. Ruchte W.D., Tharp A.L.: "Linear hashing with priority Splitting", Proceedings 3rd IEEE Data Engineering Conference, pp.2-9, 1987. (10)
297. Ruemmler C., Wilkes J.: "An introduction to disk drive modeling", IEEE Computer, Vol.27, No.4, pp.17-28, 1994. (2)
298. Sacks-Davis R., Kent A., Ramamohanarao K.: "Multikey access methods based on superimposed coding techniques", ACM Transactions on Database Systems, Vol.12, No.4, pp.655-696, 1987. (12)
299. Sagiv Y.: "Concurrent operations on B*trees with overtaking", Journal of Computer and Systems Sciences, Vol.33, pp.275-296, 1986. (15)

300. Saltzer J.H., Schroeder M.D.: "The protection of information in computer systems", Proceedings of the IEEE, Vol.63, No.9, pp.1278-1308, 1975. (15)
301. Salzberg B., Tsotras V.: "A comparison of access methods for time evolving data", ACM Computing Surveys, 1999. (14)
302. Samadi B.: "B-trees in a system with multiple views", Information Processing Letters, Vol.5, No.4, pp.107-112, 1976. (15)
303. Samet H.: "The quadtree and related hierarchical data structures", ACM Computing Surveys, Vol.16, No.2, pp.187-260, 1984. (14)
304. Sager T.J.: "A polynomial time generator for minimal perfect hash functions", Communications of the ACM, Vol.28, No.5, pp.523-532, 1985. (9)
305. Schkolnick M.: "The optimal selection of secondary indices for files", Information Systems, Vol.1, pp.141-146, 1975. (13)
306. Scholl M.: "New file organizations based on dynamic hashing", ACM Transactions on Database Systems, Vol.6, No.1, pp.194-211, 1981. (10)
307. Sellis T., Roussopoulos N., Faloutsos C.: "The R⁺-tree - a dynamic index for multidimensional objects", Proceedings 13th Conference on Very Large Data Bases, pp.507-518, 1987. (14)
308. Severance D.: "Identifier search mechanisms - a survey and generalized model", ACM Computing Surveys, Vol.6, No.3, pp.174-194, 1974. (13)
309. Severance D., Duhne R.: "A practitioner's guide to addressing algorithms", Communications of the ACM, Vol.19, No.6, pp.314-326, 1976. (9)
310. Severance D., Lohman G.M.: "Differential files - their applications to the maintenance of large databases", ACM Transactions on Database Systems, Vol.1, No.3, pp.256-267, 1976. (12,13)
311. Severance D., Carlis J.V.: "A practical approach to selecting record access paths", ACM Computing Surveys, Vol.9, No.4, pp.259-272, 1977. (13)
312. Shadri S., Rotem D., Segev A.: "Optimal arrangements of cartridges in carousel type mass storage systems", The Computer Journal, Vol.36, No.10, pp.873-887, 1994. (2)
313. Shasha D., Goodman N.: "Concurrent search structure algorithms", ACM Transactions on Database Systems, Vol.13, No.1, pp.53-90, 1988. (15)
314. Sharma K.D., Rani R.: "Choosing optimal branching factors for k-d B-trees", Information Systems, Vol.10, No.1, pp.127-134, 1985. (11)
315. Shell D.S.: "Optimizing the polyphase sort", Communications of the ACM, Vol.14, No.11, pp.713-719, 1971. (5)
316. Shlomo M.: "On the complexity of designing optimal partial match retrieval systems", ACM Transactions on Database Systems, Vol.8, No.4, pp.543-551, 1983. (13)
317. Shneiderman B.: "Polynomial search", Software - Practice and Experience, Vol.3, No.1, pp.5-8, 1973. (4)
318. Shneiderman B.: "Optimum database reorganization points", Communications of the ACM, Vol.16, No.6, pp.362-365, 1973. (13)

319. Shneiderman B.: "A model for optimizing indexed file structures", International Journal of Computer and Information Sciences, Vol.3, No.1, pp.93-103, 1974. (7)
320. Shneiderman B., Goodman V.: "Batched searching of sequential and tree-structured files", ACM Transactions on Database Systems, Vol.1, No.3, pp.268-275, 1976. (8,13)
321. Shneiderman B.: "Reduced combined indexes for efficient multiple attribute retrieval", Information Systems, Vol.1, No.4, pp.149-154, 1977. (11)
322. Shneiderman B.: "Jump searching - a fast sequential search technique", Communications of the ACM, Vol.21, No.10, pp.831-834, 1978. (4)
323. Six H.W., Wegner L.: "Sorting a random access file in situ", The Computer Journal, Vol.27, No.3, pp.270-275, 1984. (6)
324. Smith A.J.: "Cache memories", ACM Computing Surveys, Vol.14, No.3, pp.473-530, 1982. (3)
325. Snyder L.: "On B-trees re-examined", Communications of the ACM, Vol.21, No.7, pp.594, 1978. (8)
326. Sobel S.: "Oscillating sort - a new sort merging technique", Journal of the ACM, Vol.9, pp.372-374, 1962. (5)
327. Sockut G.H., Goldberg R.P.: "Database reorganization - principles and practice", ACM Computing Surveys, Vol.11, No.4, pp., 1979. (13)
328. Solomon M.K., Bickel R.W.: "A self-assessment procedure dealing with file processing", Communications of the ACM, Vol.29, No.8, pp.745-750, 1986. (13)
329. Sprugnoli R.: "Prefect hashing functions - a simple probe retrieving method for static sets", Communications of the ACM, Vol.20, No.11, pp.841-850, 1977. (9)
330. Srinivasan V., Carey M.J.: "Performance of B-tree concurrency control algorithms", Proceedings ACM SIGMOD 91 Conference, pp.416-425, 1991. (15)
331. Stonebraker M.: "The choice of partial inversions and combined indices", Journal of Computer and Information Sciences, pp.167-188, 1974. (11)
332. Stonebraker M.: "Operating system support for database management", Communications of the ACM, Vol.24 No.7 pp.412-418, 1981. (13)
333. Stonebraker M., Sellis T., Hanson E.: "Rule indexing implementations in database systems", Proceedings 1st Conference on Expert Database Systems, pp.465-476, 1986. (15)
334. Sussenguth E.H.: "Use of tree structures for processing files", Communications of the ACM, Vol.6, No.5, pp. 272-279, 1963. (8)
335. Tai K.C., Tharp A.L.: "Computed chaining - a hybrid of direct chaining and open addressing", Information Systems, Vol.6, No.2, pp.111-116, 1981. (9)
336. Tan K.P., Hsu L.S.: "Block sorting of a large file in external storage by a 2-component key", The Computer Journal, Vol.25, No.3, pp.327-329, 1985. (6)
337. Taylor D., Morgan D., Black J.: "Redundancy in data structures - improving software fault tolerance", IEEE Transactions on Software Engineering, Vol.6, No.6, pp.585-594, 1980. (13)

338. Taylor D., Black J.: "A locally correctable B-tree implementation", *The Computer Journal*, Vol.29, pp.269-276, 1986. (8,13)
339. Teuhola J., Wegner L.: "Minimal space - average linear time duplicate deletion", *Communications of the ACM*, Vol.34, No.3, pp. 62-73, 1989. (1)
340. Ting T.C., Wang Y.W.: "Multiway replacement selection sort with dynamic reservoir", *The Computer Journal*, Vol.20, No.4, 1977. (5)
341. Tomasic A., Garcia-Molina H., Shoens K.: "Incremental updates of inverted lists for text document retrieval", *Proceedings ACM SIGMOD 94 Conference*, pp.289-300, 1994. (11,15)
342. Toobs D.: "CCD and bubble memories", *IEEE Spectrum*, Vol.15, No.4, pp.22-30 and No.5, pp.36-39, 1978. (2)
343. Tsotras V., Kangelaris N.: "The snapshot index - an I/O optimal access method for timeslice queries", *Information Systems*, Vol.20, No.3, pp.237-260, 1995. (14)
344. Tzouramanis T., Vassilakopoulos M., Manolopoulos Y.: "Overlapping linear quadtrees - a spatiotemporal access method", *Proceedings 6th ACM Workshop on Advances in Geographical Information Systems*, pp.1-7, 1998. (14)
345. Tuel W.G.jr.: "Optimum reorganization points for linearly growing files", *ACM Transactions on Database Systems*, Vol.3, No.1, pp.32-40, 1978. (13)
346. Vaishnavi V.K.: "On the height of multidimensional height-balanced trees", *IEEE Transactions on Computers*, Vol.35, No.9, pp.773-780, 1986. (11)
347. Vaishnavi V.K.: "Multidimensional balanced binary trees", *IEEE Transactions on Computers*, Vol.38, No.7, pp.968-985, 1989. (11)
348. Vakali A., Manolopoulos Y.: "Parallel data paths in two-headed disk systems", *Information and Software Technology*, Vol.39, No.1, pp.125-135, 1997. (2)
349. Vakali A., Manolopoulos Y.: "An exact analysis on expected seeks in shadowed disks", *Information Processing Letters*, Vol.61, No.6, pp.323-329, 1997. (2)
350. Varman P.J., Verma R.M.: "An efficient multi-version access structure", *IEEE Transactions on Knowledge and Data Engineering*, Vol.9, No.3, pp.391-409, 1997. (14)
351. Vassilakopoulos M., Manolopoulos Y., Economou K.: "Overlapping quadtrees for the representation of similar images", *Image and Vision Computing*, Vol.11, No.5, pp.257-262, 1993. (14)
352. Vassilakopoulos M., Manolopoulos Y.: "Analytical comparison of two spatial data structures", *Information Systems*, Vol.19, No.7, pp.569-582, 1994. (14)
353. Vassilakopoulos M., Manolopoulos Y., Kröll B.: "Efficiency analysis of overlapped quadtrees", *Nordic Journal on Computing*, Vol.2, No.1, pp.70-84, 1995. (14)
354. Vassilakopoulos M., Manolopoulos Y.: "Dynamic inverted quadtrees - a structure for pictorial databases", *Information Systems, Special Issue on Multimedia Information Systems*, Vol.20, No.6, pp.483-500, 1995. (14)
355. Vassilakopoulos M., Manolopoulos Y.: "On random models for analyzing region quadtrees", *Pattern Recognition Letters*, Vol.16, pp.1137-1145, 1995. (14)
356. Veclerov E.: "Analysis of dynamic hashing with deferred splitting", *ACM Transactions on Database Systems*, Vol.10, No.1, pp.90-196, 1985. (10)

357. Vitter J.S.: "Analysis of the search performance of coalesced hashing", *Journal of the ACM*, Vol.30, No.2, pp.231-258, 1983. (9)
358. Vitter J.S.: "An efficient I/O interface for optical disks", *ACM Transactions on Database Systems*, Vol.10, No.2, pp.129-162, 1985. (2,13)
359. Wagner R.: "Indexing design considerations", *IBM Systems Journal*, Vol.4, pp.351-367, 1974. (8)
360. Waters S.J.: "Blocking sequentially processed magnetic files", *The Computer Journal*, Vol.14, No.2, 1971. (3)
361. Wegner L., Teuhola J.: "The external heapsort", *IEEE Transactions on Software Engineering*, Vol.15, No.7, pp.917-925, 1989. (6)
362. Wilkes J., Golding R., Staelin C., Sullivan T.: "The HP AutoRAID hierarchical storage system", *ACM Transactions on Computer Systems*, Vol.14, No.1, pp.108-136, 1996. (2)
363. Willard D.: "Maintaining dense sequential files in a dynamic environment", *Proceedings 14th ACM SIGACT Conference*, pp. 114-121, 1982. (3)
364. Willard D.: "Searching unindexed and nonuniformly generated files in loglogN time", *SIAM Journal on Computing*, Vol.14, No.4, pp. 1013-1029, 1985. (4)
365. Willard D.: "Good worst-case algorithms for inserting and deleting records in dense sequential files", *Proceedings ACM SIGMOD 86 Conference*, pp.251-260, 1986. (4)
366. Williams F.A.: "Handling identifiers as internal symbols in language processors", *Communications of the ACM*, Vol.2, No.6, pp.21-24, 1959. (9)
367. Williams J.W.J.: "Algorithm 232", *Communications of the ACM*, Vol.7, No.6, pp.347-348, 1964. (6)
368. Wong C.K.: "Minimizing expected head movement in one-dimensional and two-dimensional mass storage systems", *ACM Computing Surveys* Vol.12, No.2, pp.167-178, 1980. (2)
369. Wong H., Li J.Z.: "Transposition algorithms for very large compressed databases", *Proceedings 12th Conference on Very Large Data Bases*, pp.304-311, 1986. (13)
370. Wright W.E.: "A note on external sorting using almost single input buffering", *Information Processing Letters*, Vol.24, pp.403-405, 1987. (6)
371. Yao A.: "On random 2-3 trees", *Acta Informatica*, Vol.2, No.9, pp. 159-170, 1978. (8)
372. Yang C.S.: "Avoiding redundant record accesses in unsorted multilist file organizations", *Information Systems*, Vol.2, pp.155-158, 1977. (11)
373. Yang C.S.: "A class of hybrid list file organizations", *Information Systems*, Vol.3, No.1, pp.49-58, 1978. (11)
374. Yao S.B., Das K.S., Teorey T.J.: "A dynamic reorganization algorithm", *ACM Transactions on Database Systems*, Vol.1, No.2, pp.159-174, 1976. (13)
375. Yu C.T., Suen C.M., Lam K., Siu M.K.: "Adaptive record clustering", *ACM Transactions on Database Systems*, Vol.10, No.2, pp.180-204, 1985. (13)
376. Xu X., Han J., Lu W.: "RT-tree: an improved R-tree index structure for spatiotemporal databases", *Proceedings 4th Symposium on Spatial Data Handling*, pp.1040-1049, 1990. (14)