Κεφάλαιο 3

ΔΙΑΧΕΙΡΙΣΗ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ

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,

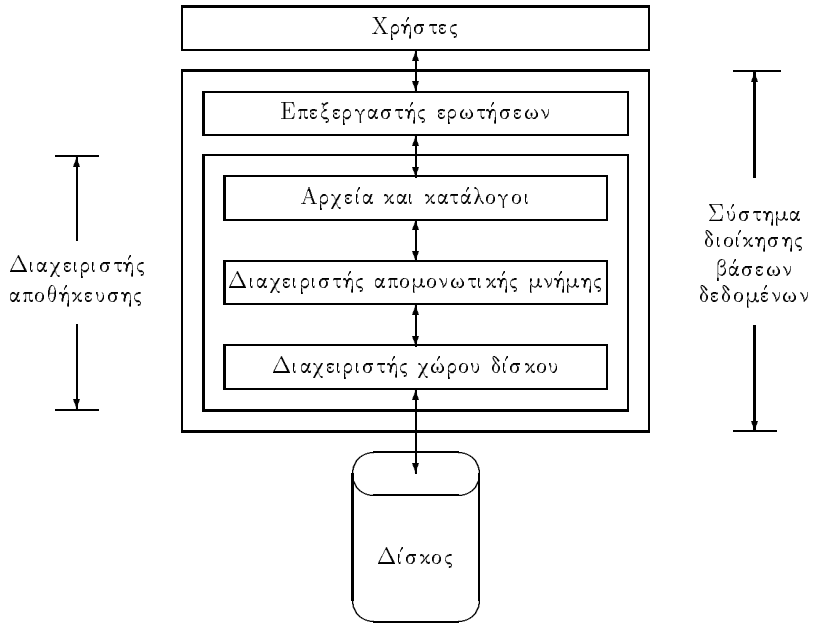
- οι κατασκευαστές συστημάτων διαχείρισης βάσεων δεδομένων επιθυμούν το προϊόν τους να τρέχει σε διάφορες πλατφόρμες λειτουργικών συστημάτων. Έτσι, είναι αναγκασμένοι να μην προσαρμόσουν το

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.

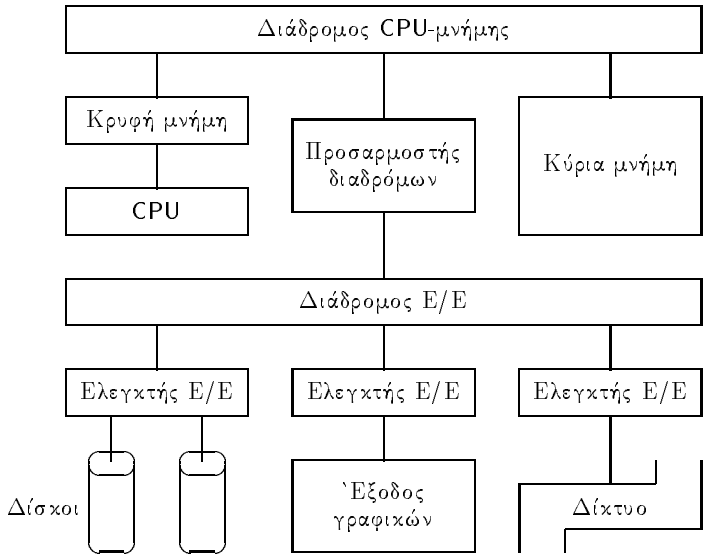
Ο κάθε ένας από τους προηγούμενους συντελεστές επικοινωνεί με το γειτονικό του στρώμα ζητώντας και δίνοντας δεδομένα. Για παράδειγμα, ο διαχειριστής του χώρου του δίσκου είναι το λογισμικό που είναι υπεύθυνο για την ανάγνωση και την αποθήκευση σελίδων στο δίσκο, ενώ ο διαχειριστής της απομονωτικής μνήμης 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).

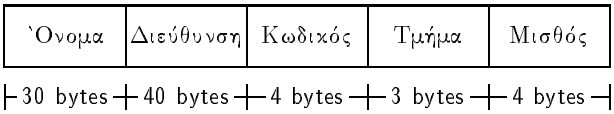
Σκοπός και των δύο είναι η εξομάλυνση του ρυθμού μεταφοράς δεδομένων. Ειδικότερα, αν και το κανάλι έχει δική του μικρή μνήμη και μία μικρή ομάδα εντολών, εντούτοις δεν είναι ισότιμος επεξεργαστής προς τον κεντρικό επεξεργαστή. Η βασική λειτουργία του καναλιού είναι η μετατροπή των λέξεων της κύριας μνήμης 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 **γραμμογράφηση** (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 ιδαίτερα 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:

- Ὑπαρξη **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	Ὄνομα	Διεύθυνση	Τμήμα	Ωρομίσθιο	Ὧρες
Τύπος=Ω	Ὄνομα	Διεύθυνση	Τμήμα	Μισθός	

Σχήμα 3.4: Εγγραφές μεταβλητού μήκους.

- Ὑπαρξη **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.

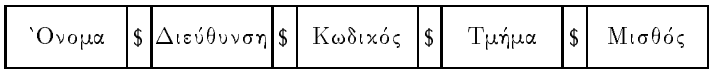
- Ὑπαρξη **επαναλαμβανόμενων ομάδων πεδίων** (repeating groups). Το φαινόμενο αυτό συμβαίνει όταν ένα ή περισσότερα χαρακτηριστικά επαναλαμβάνονται περισσότερο από δύο φορές. Για παράδειγμα, η οντότητα `υπάλληλος΄ μεταξύ των άλλων μπορεί να περιέχει τα χαρακτηριστικά `όνομα_προστατευόμενου΄ και `ημερομηνία_γέννησης΄ προστατευόμενου. Συνεπώς η εγγραφή ενός υπάλληλου με δύο ή τρία

Ὄνομα	Κωδικός	Ὄνομα Προστατευόμ.	Ημερομηνία Γέννησης	Ὄνομα Προστατευόμ.	...

Σχήμα 3.5: Εγγραφή με επαναλαμβανόμενες ομάδες.

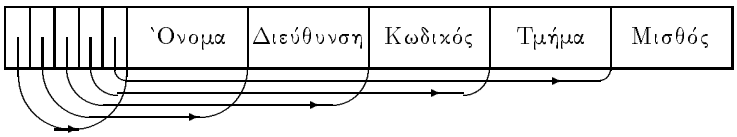
παιδιά θα περιέχει αντίστοιχα δύο ή τρία στιγμιότυπα των χαρακτηριστικών αυτών. Στο Σχήμα 3.5 παρουσιάζεται μία τέτοια περίπτωση. Μία τέτοια περίπτωση αντιμετωπίζονταν εύκολα από την Cobol με την εντολή Occurs. Ωστόσο, τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων δεν επιτρέπουν σε μία εγγραφή την 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:



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 αποθηκεύται η νέα διεύθυνση ώστε να μη χαθούν δεδομένα.

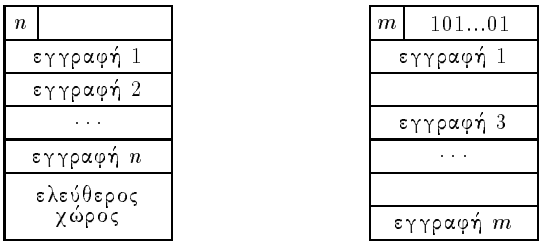
3.3 Τύποι σελίδων

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 χαρακτηρισθεί 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.



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 συνεχίζε 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 τακτοποίηση 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 υποτεθεί 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 'κουτσό'), που επιτυγχάνει καλύτερους χρόνους προσπέλασης αλλά είναι σχετικά περίπλοκη.

Η τεχνική της παρεμβολής είναι αναγκαία επειδή ο ελεγκτής 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 ατράκτου. Επίσης, ας υποτεθεί ότι ο ελεγκτής απαιτεί διπλάσιο

Φυσικός τομέας	1	2	3	4	5	6	7	8	9	10
Λογικός τομές	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 αρκούν τρεις περιστροφές για την ανάγνωση όλων των τομέων της ατράκτου, 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 ατράκτους 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.

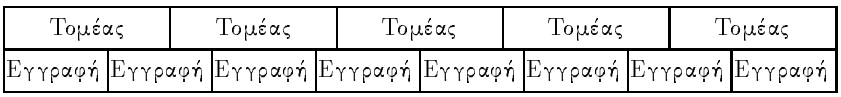
Η εκλογή του κατάλληλου μεγέθους κάδου με σκοπό τη βελτιστοποίηση της χρήσης του χώρου είναι υπόθεση του διαχειριστή του συστήματος και δεν υπολογίζεται δύσκολα. Ας υποτεθεί ότι το μέγεθος της εγγραφής είναι 160 bytes, ενώ το μέγεθος της σελίδας είναι 256 bytes. Οι επιλογές είναι οι εξής:

- 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.



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 κάδου, 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 συγκεχριμένο 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.

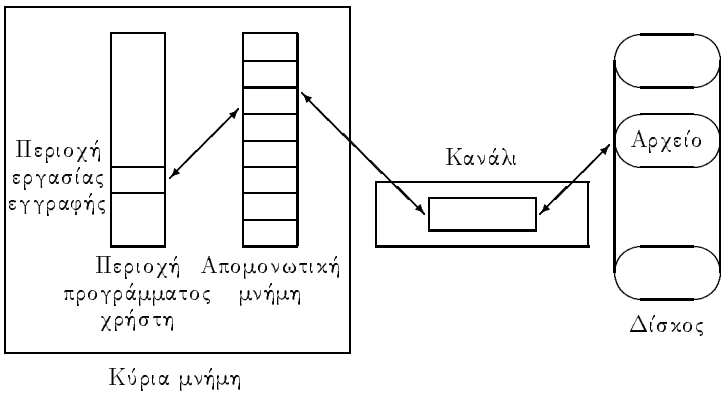
3.6 Διαχείριση απομονωτικής μνήμης

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 ελεγκτή 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 καρφωμάτων (pin_count) και του λεγόμενου βρώμικου bit, οι οποίοι αρχικοποιούνται με τιμές 0_{10} και 0_2.

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

pinning), ενώ ο μετρητής μειώνεται κατά ένα αν ερθεί μήνυμα ότι το περιεχόμενο κάποιου πλαίσιου δεν είναι πλέον χρήσιμο (διαδικασία `ξεκάρφωμα´ - 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. `Ομως 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). `Οπως 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 ακόμη αυτές ζητηθούν.

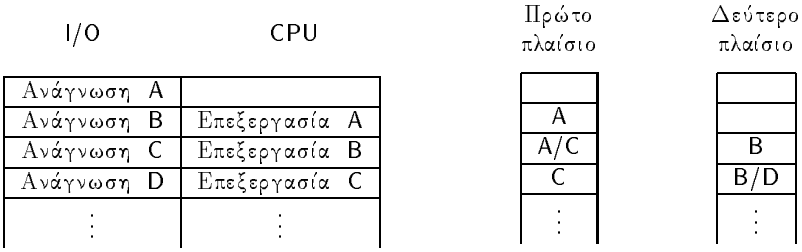
Σε εφαρμογές βάσεων δεδομένων το κόστος επεξεργασίας των σελίδων από τη στιγμή που θα έρθουν στις απομονωτικές μνήμες θεωρείται αμελητέο σε σχέση με το κόστος μεταφοράς δεδομένων από/προς τη δευτερεύουσα μνήμη. `Ετσι, η προ-προσπέλαση βελτιώνει σημαντικά την επίδοση του συστήματος όταν δεν αιτώνται όλες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

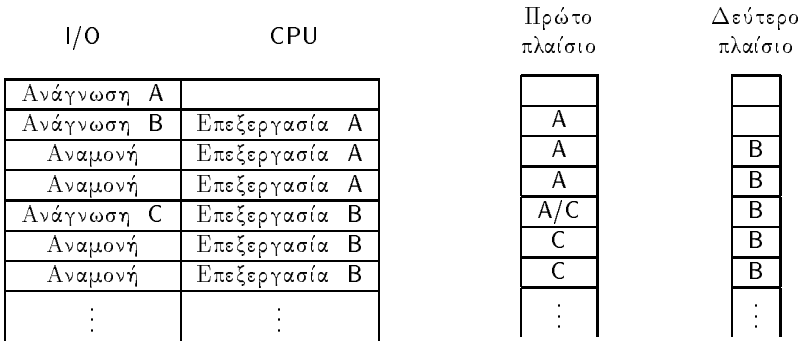


Σχήμα 3.12: Χρονική αλληλουχία εργασιών με χρήση δύο πλαισίων.

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:

$$\lceil t \rceil \times b \times ebt$$

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 τεχικές 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, fisrt 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 ατράκτου είναι 10.000 bytes και ότι απαιτούνται 10 ms για μία πλήρη περιστροφή. Επίσης, ας υποτεθεί ότι το μέγεθος των σελίδων είναι 1000 bytes, ενώ το μέγεθος των κενών λόγω φορμαρίσματος είναι 100 bytes. Να υπολογισθεί η μέγιστη (στιγμιαία) ταχύτα 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 ποιό 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. Ποιό 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**> `Εστω ότι μία συσκευή δίσκων CD πρόκειται με τη βοήθεια μίας διεπιφάνειας να συνδεθεί με ένα μικροϋπολογιστή για αποθήκευση αρχείων. `Ενας 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.