ΒΑΣΕΙΣ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

ΓΙΑΝΝΗΣ ΘΕΟΔΩΡΙΔΗΣ

7.1 ΕΙΣΑΓΩΓΗ

Οι *βάσεις χωρικών δεδομένων* αποτελούν σημείο αιχμής της έρευνας στον τομέα των βάσεων δεδομένων τα τελευταία 15 χρόνια. Η βασική (και περισσότερο προφανής) εφαρμογή τους είναι η υποστήριξη των Γεωγραφικών Πληροφοριακών Συστημάτων - ΓΠΣ (Geographical Information Systems - GIS) [AG97]. Τα γεωγραφικά συστήματα πληροφοριών είναι ιδιαίτερα χρήσιμα στη χαρτογράφηση τόσο περιοχών όσο και διαφόρων δικτύων όπως οδικών, τηλεφωνικών, υπολογιστικών κ.α. Χωρικά δεδομένα, όμως, συναντούνται και σε άλλες εφαρμογές, όπως ΒΔ Εικόνων, Πολυμέσων κλπ. Είναι, λοιπόν, από πρακτικής άποψης, μεγάλο το ενδιαφέρον για την ανάπτυξη τέτοιων συστημάτων.

Αρχικά θα παρουσιάσουμε σε συντομία το τι είναι μια βάση χωρικών δεδομένων και ποια είναι τα κύρια χαρακτηριστικά της. Στη συνέχεια, θα επιχειρήσουμε μια πιο αναλυτική παρουσίαση του θέματος της οργάνωσης αυτών των τύπων δεδομένων με χρήση των πολυδιάστατων μεθόδων προσπέλασης (multidimensional access methods). Ως γνωστόν, οι μέθοδοι προσπέλασης (ή δεικτοδότησης) παίζουν πρωτεύοντα ρόλο γενικότερα ως προς την επίδοση των βάσεων δεδομένων.

7.2 ΑΝΤΙΚΕΙΜΕΝΟ

7.2.1 Ορισμός

Το πρώτο πράγμα που πρέπει να πούμε είναι το τι είναι μια βάση χωρικών δεδομένων και κυρίως τι είναι αυτό που την κάνει να διαφέρει από μια "απλή" βάση δεδομένων. Καταρχήν μια βάση χωρικών δεδομένων προφανώς παρέχει όλα όσα παρέχονται και από τις "απλές" βάσεις.

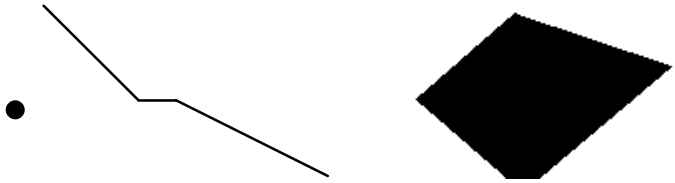
Όμως προσφέρει επιπλέον την δυνατότητα να παρασταθούν και να αποθηκευτούν *τύποι χωρικών δεδομένων* όπως π.χ. ένα σημείο στο χώρο, μια ευθεία ή και ένα πολύπλοκο γεωμετρικό σχήμα ενός κ-διάστατου χώρου (πρακτικά, δε γεωγραφικές εφαρμογές, ενδιαφερόμαστε πιο πολύ για την περίπτωση όπου κ = 2 ή 3). Δεν αρκεί βέβαια η παράσταση των παραπάνω τύπων. Χρειάζεται και υποστήριξη των σχέσεων μεταξύ τους (όπως του αν ένα ευθύγραμμο τμήμα τέμνεται η όχι με ένα άλλο), των ιδιοτήτων τους (όπως του αν ένα τετράγωνο έχει εμβαδό μεγαλύτερο από κάποια τιμή), καθώς και διαφόρων πράξεων με αυτά (όπως το να βρεθεί η τομή δύο παραλληλογράμμων). Τέλος, είναι απαραίτητο να υπάρχει ένας αποδοτικός τρόπος αναζήτησης και προσπέλασης των τύπων χωρικών δεδομένων όπως ακριβώς και στις "απλές" βάσεις. Στην περίπτωση μας όμως οι μέθοδοι αναζήτησης και προσπέλασης είναι εκ των πραγμάτων διαφορετικές και πιο ισχυρές, αφού θα χρειαστούμε λειτουργίες που δεν συναντώνται στις "απλές" βάσεις, όπως, για παράδειγμα, το να βρεθούν εκείνα τα διδιάστατα σχήματα που βρίσκονται σε κάποιο υποχώρο του διδιάστατου χώρου. Σαν πρακτικό παράδειγμα μπορούμε να δώσουμε το να βρεθούν τα κράτη εκείνα που ανήκουν γεωγραφικά σε ένα παραλληλόγραμμο που ορίζει ο χρήστης ενός πακέτου ΓΠΣ.

7.2.2 Τύποι χωρικών δεδομένων

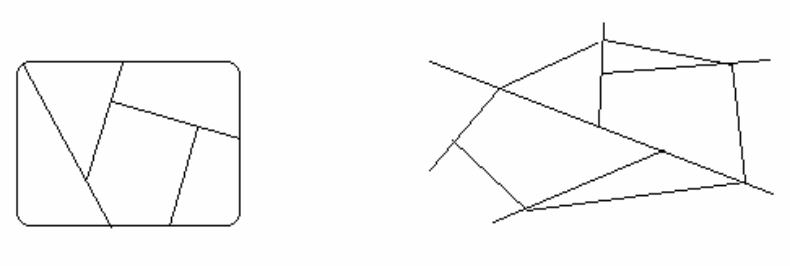
Αξίζει να αναφερθούμε λίγο στο τι ακριβώς είναι οι τύποι χωρικών δεδομένων. Πριν από αυτό πρέπει κανείς να σκεφτεί τι χρειάζεται να μοντελοποιηθεί και άρα πως θα θέλαμε να ορίσουμε τους παραπάνω τύπους έτσι ώστε να ικανοποιούν τις ανάγκες μας.

Υπάρχουν, γενικά, δύο πράγματα που θέλουμε κάπως να μοντελοποιήσουμε. Το πρώτο είναι ο ίδιος ο χώρος. Επί παραδείγματι, θέλουμε σε ένα χάρτη να χωρίσουμε το χώρο σε κράτη. Το δεύτερο είναι διάφορα αντικείμενα στο χώρο όπως πόλεις, ποτάμια, δρόμους, δίκτυα μεταφοράς κ.τ.λ. καθώς και πιθανές συνδέσεις ανάμεσα τους όπως μια σιδηροδρομική διασταύρωση.

Για την μοντελοποίηση αντικειμένων στο χώρο χρειαζόμαστε σημεία, γραμμές και περιοχές (σχήματα με εμβαδό). Τα σημεία αντιπροσωπεύουν την περίπτωση όπου μόνο η θέση στο χώρο παίζει ρόλο και όχι και η έκταση. Οι γραμμές αποτελούν το τρόπο σύνδεσης αντικειμένων και άρα είναι κατάλληλες για την αναπαράσταση δρόμων, ποταμών κ.τ.λ. Τέλος, οι περιοχές αναπαριστούν αντικείμενα όπου έχει σημασία και η έκταση τους στο χώρο εκτός της θέσης τους. Όσον αφορά την μοντελοποίηση του ίδιου του χώρου χρειαζόμαστε ένα τρόπο να τον χωρίζουμε σε τεμάχια και ένα τρόπο να φτιάχνουμε δίκτυα μέσα του (προφανώς από γραμμές). Τα παραπάνω φαίνονται στα σχήματα 7.1 και 7.2 [Gu94].



Σχήμα 7.1: Σημείο, γραμμή, περιοχή

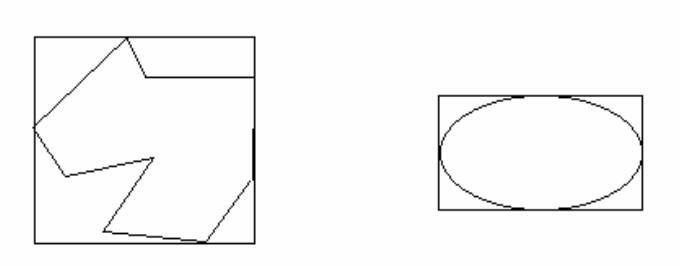


Σχήμα 7.2: Τεμαχισμός, δικτυώση

Από τα παραπάνω είναι φανερό πως τα χωρικά δεδομένα είναι πολύπλοκες δομές. Μπορούν να αποτελούνται είτε από ένα απλό σημείο είτε από πάρα πολλά σημεία αυθαίρετα τοποθετημένα στο χώρο. Κατά συνέπεια είναι κάποιες φορές αδύνατο να αποθηκεύσουμε τέτοια δεδομένα σε κλασσικούς σχεσιακούς πίνακες. Επίσης, τα δεδομένα (άρα και οι αντίστοιχες ΒΔ) τείνουν να καταλαμβάνουν πολύ μεγάλο χώρο στο δίσκο (έως αρκετά gigabytes). Επομένως είναι απαραίτητη η αποθήκευση τους σε δευτερεύουσα μνήμη. Ακόμα, δεν υπάρχει κοινώς αποδεκτή (πρότυπη) άλγεβρα για τα χωρικά δεδομένα. Η αλγεβρα που χρησιμοποιείται κάθε φορά εξαρτάται από την εκάστοτε εφαρμογή. Τέλος, οι χωρικοί τελεστές είναι εν γένει πιο ακριβοί από τους απλούς.

7.2.3 Τελεστές

Προκειμένου να γίνουμε πιο σαφείς ως προς το τι συζητάμε, θα δώσουμε τους ορισμούς των πιο συχνά συναντώμενων τελεστών χωρικών δεδομένων. Πριν από αυτό, όμως, πρέπει να τονίσουμε ότι πολύ συχνά, λόγω της πολυπλοκότητας των γεωμετριών των αντικειμένων σε μια βάση χωρικών δεδομένων, οι τελεστές αρχικά επιχειρούν να βρουν υποψήφιες λύσεις χρησιμοποιώντας περιβάλλοντα κουτιά (bounding box) τα οποία περικλείουν τη σύνθετη γεωμετρία (σχήμα 7.3). Τα περιβάλλοντα κουτιά έχουν προφανώς το ελάχιστο εμβαδό και για αυτό τα καλούμε *ελάχιστα περιβάλλοντα κουτιά.* Στη συνέχεια, οι τελεστές ελέγχουν μία προς μια τις υποψήφιες λύσεις χρησιμοποιώντας την ακριβή γεωμετρία των αντικειμένων.



Σχήμα 7.3: Ελάχιστα περιβάλλοντα κουτιά

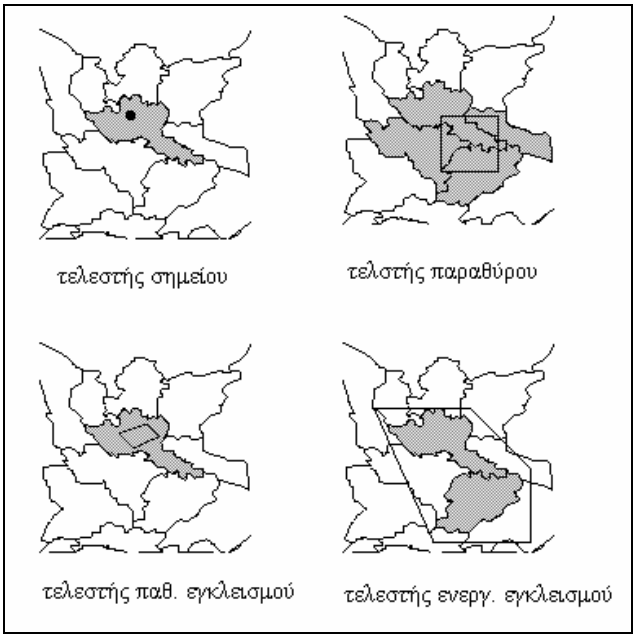
Ας δούμε τώρα αναλυτικά τους πιο δημοφιλείς τελεστές βάσεων χωρικών δεδομένων:

- *Ισότητας* (exact match query): Δοθέντος ενός αντικειμένου με μια γεωμετρία να βρεθούν όλα τα αντικείμενα με την ίδια γεωμετρία.
- *Σημείου* (point query): Δοθέντος ενός σημείου του κ-διάστατου χώρου, να βρεθούν όλα τα αντικείμενα που το περιέχουν.
- *Παραθύρου* (window query): Δοθέντος ενός κ-διάστατου παραθύρου, να βρεθούν όλα τα αντικείμενα που έχουν ένα τουλάχιστον κοινό σημείο με το παράθυρο.

- *Τομής* (intersection query): Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα με τα οποία έχει κοινά εσωτερικά σημεία.
- *Παθητικού Εγκλεισμού* (enclosure query): Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα που το περιέχουν.
- *Ενεργητικού Εγκλεισμού* (containment query): Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα που αυτό περιέχει.
- *Γειτνίασης* (adjacent query): Δοθέντος ενός αντικειμένου να βρεθούν όλα τα αντικείμενα με τα οποία δεν έχει κανένα κοινό εσωτερικό σημείο αλλά έχει κάποια κοινά εξωτερικά (τα αντικείμενα, δηλαδή, με τα οποία εφάπτεται κατά κάποιο τρόπο).

Τέλος αξίζει να σημειωθεί και η ανάγκη για σύνδεση (join) με βάση κάποιο χωρικό χαρακτηριστικό των δεδομένων μιας βάσης. (spatial join).

Στο παρακάτω σχήμα δίνουμε σχηματικά κάποιους από τους παραπάνω τελεστές [GG98]:



Σχήμα 7.4: Χωρικοί Τελεστές

7.2.4 Μέθοδοι προσπέλασης χωρικών δεδομένων

Μια σημαντική κλάση χωρικών τελεστών οι οποίοι χρειάζονται ειδική αντιμετώπιση είναι οι τελεστές *επιλογής* χωρικών δεδομένων γιατί η ανεύρεση και ενημέρωση τέτοιων δεδομένων εξαρτάται και από την θέση τους στο χώρο. Οι μέθοδοι που χρησιμοποιούν αυτοί οι τελεστές διακρίνονται σε δύο μεγάλες κατηγορίες: στις *μεθόδους προσπέλασης σημείων* (point access methods) και στις *μεθόδους προσπέλασης περιοχών* (spatial access methods). Το βασικό πρόβλημα στη σχεδίαση τέτοιων μεθόδων είναι ότι δεν υπάρχει απόλυτος τρόπος αντιστοίχισης του διδιάστατου χώρου (κ-διάστατου στη γενική περίπτωση) στο μονοδιάστατο χώρο ο οποίος να εγγυάται πως δύο αντικείμενα που βρίσκονται κοντά στο διδιάστατο χώρο θα βρίσκονται κοντά και στον αντίστοιχο μονοδιάστατο. Άρα προκύπτει ότι δεν μπορούν να χρησιμοποιηθούν οι 'κλασικές' μέθοδοι προσπέλασης, όπως τα B-δέντρα ή τα αρχεία κατακερματισμού, αλλά απαιτούνται ειδικές μέθοδοι. Τις παραπάνω μεθόδους, λόγω ακριβώς του γεγονότος ότι αφορούν κ-διάστατους χώρους, τις ονομάζουμε *πολυδιάστατες μεθόδους προσπέλασης*

(multidimensional access methods) και η μελέτη τους θα μας απασχολήσει στη συνέχεια. Επίσης, οι ζητούμενες μέθοδοι οφείλουν να ικανοποιούν πολλές απαιτήσεις όπως δυνατότητα αποδοτικής χρήσης δευτερεύουσας μνήμης, ανεξαρτησία από την κατανομή των δεδομένων, απλότητα (simplicity), ανεξαρτησία από την διόγκωση της βάσης (scalability) και βέβαια αποδοτικότητα (efficiency).

7.3 ΧΩΡΙΚΕΣ ΜΕΘΟΔΟΙ ΔΕΙΚΤΟΔΟΤΗΣΗΣ

7.3.1 Μονοδιάστατες μέθοδοι προσπέλασης

Είναι χρήσιμο να αναφερθούμε εν συντομία στις βασικές μονοδιάστατες μεθόδους προσπέλασης που χρησιμοποιούνται σήμερα.

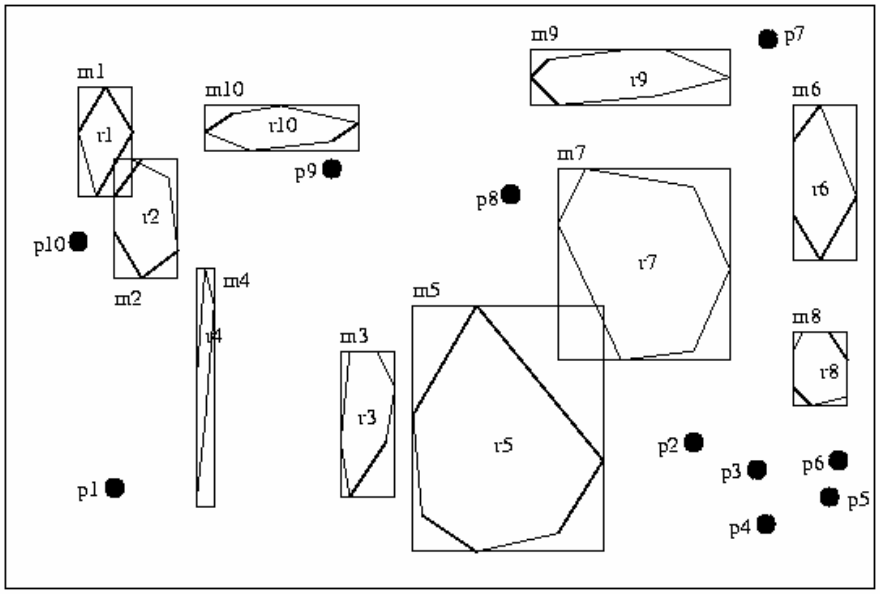
Αναφέρουμε αρχικά το *γραμμικό κατακερματισμό* (linear hashing) [Li80]. Σύμφωνα με τη μέθοδο αυτή, ένα αρχικό διάστημα [A,B) διαιρείται σε διαστήματα (intervals) μεγέθους $(B-A)/2^k$ και $(B-A)/2^{k+1}$. Τα δεύτερα εξ αυτών, που είναι και τα πιο μικρά, ξεκινάνε από το A και φτάνουν μέχρι έναν δείκτη t. Από τον t και μετά έχουμε τα πρώτα διαστήματα που έχουν διπλάσιο μήκος από τα άλλα. Κάθε διάστημα αντιστοιχεί σε ένα κάδο (bucket) που είναι ένα σύνολο από εγγραφές (records) αποθηκευμένα σε μια σελίδα του δίσκου. Σύμφωνα με μια συνάρτηση, κάθε νέα εγγραφή μπαίνει σε ένα από τα διαστήματα. Όταν ένας κάδος γεμίσει, τότε ο δείκτης t προχωράει μία θέση κατόπιν σπασίματος στη μέση του διαστήματος μήκους $(B-A)/2^k$ που βρισκόταν στα δεξιά του δείκτη. Όταν ένα διάστημα περιέχει περισσότερα δεδομένα από τα επιτρεπτά στον αντίστοιχο κάδο, τα παραπάνω δεδομένα αποθηκεύονται σε μια ειδική σελίδα υπερχείλισης.

Σε αντίθεση με το παραπάνω, ο *επεκτατός κατακερματισμός* (extendible hashing) [FNPS79] δεν χρησιμοποιεί σελίδες υπερχείλισης παρά ένα κεντρικό κατάλογο. Εδώ τα διαστήματα καλούνται κελιά (cells). Κάθε κελί έχει ένα δείκτη στον κεντρικό κατάλογο και αρχικά αντιστοιχεί σε ένα κάδο. Αν κάποιος κάδος γεμίσει τότε όλα τα κελιά σπάνε στα δύο και οι δείκτες επαναπροσδιορίζονται.

Μια άλλη, ιδιαίτερα δημοφιλής, οργάνωση είναι το *B^+-δέντρο* (B^+-tree) [Kn73], το οποίο οργανώνει τα δεδομένα σε μία ιεραρχική - δεντρική - δομή. Το B^+-δέντρο είναι ένα ισοσταθμισμένο δέντρο, τα φύλλα του οποίου αποτελούνται από κλειδιά και δείκτες προς τα δεδομένα. Οι εσωτερικοί κόμβοι του δέντρου αποτελούνται από κλειδιά και δείκτες προς απογόνους κόμβους. Κάθε κόμβος έχει ένα κατώτερο και ένα ανώτερο αριθμό απογόνων κόμβων. Αν οι απόγονοι πέσουν κάτω από το κάτω όριο έχουμε συνένωση (merge) δύο κόμβων σε έναν ενώ αν ανέβουν πάνω από το άνω όριο έχουμε διάσπαση (split) του κόμβου που υπερχειλίζει σε δύο.

Το B^+-δέντρο πλεονεκτεί επί των τεχνικών κατακερματισμού όταν έχουμε ακανόνιστα δεδομένα. Μολαταύτα, στην περίπτωση όπου τα δεδομένα μας συμπεριφέρονται με καλό τρόπο (κατανομή αρκετά κοντά στην ομοιόμορφη), οι τεχνικές κατακερματισμού υπερτερούν.

Θα αναφερθούμε στη συνέχεια σε πολυδιάστατες μεθόδους προσπέλασης, οι οποίες βασίζονται είτε στα αρχεία κατακερματισμού είτε στα B^+-δέντρα. Θα παρουσιαστούν μέσω ενός κοινού παραδείγματος το οποίο απεικονίζεται στο Σχήμα 7.5 και περιλαμβάνει δέκα σημεία p_i και δέκα πολύγωνα r_i με κέντρα c_i και αντίστοιχα ελάχιστα περιβάλλοντα κουτιά m_i, τοποθετημένα σε ένα διδιάστατο χώρο.



Σχήμα 7.5: Γενικό παράδειγμα

Στη βιβλιογραφία μπορούν να βρεθούν αρκετές μέθοδοι οι οποίες λειτουργούν στην κύρια μνήμη. Η περίπτωση αυτή δεν θα μας απασχολήσει στη συνέχεια. Οι πολυδιάστατες μέθοδοι που θα αναφερθούν είναι σχεδιασμένες με τέτοιο τρόπο, ώστε να λαμβάνουν υπόψιν τους και την αποθήκευση σε δευτερεύουσα μνήμη. Ο αναγνώστης που ενδιαφέρεται περισσότερο παραπέμπεται στην εκτεταμένη επισκόπηση [GG98].

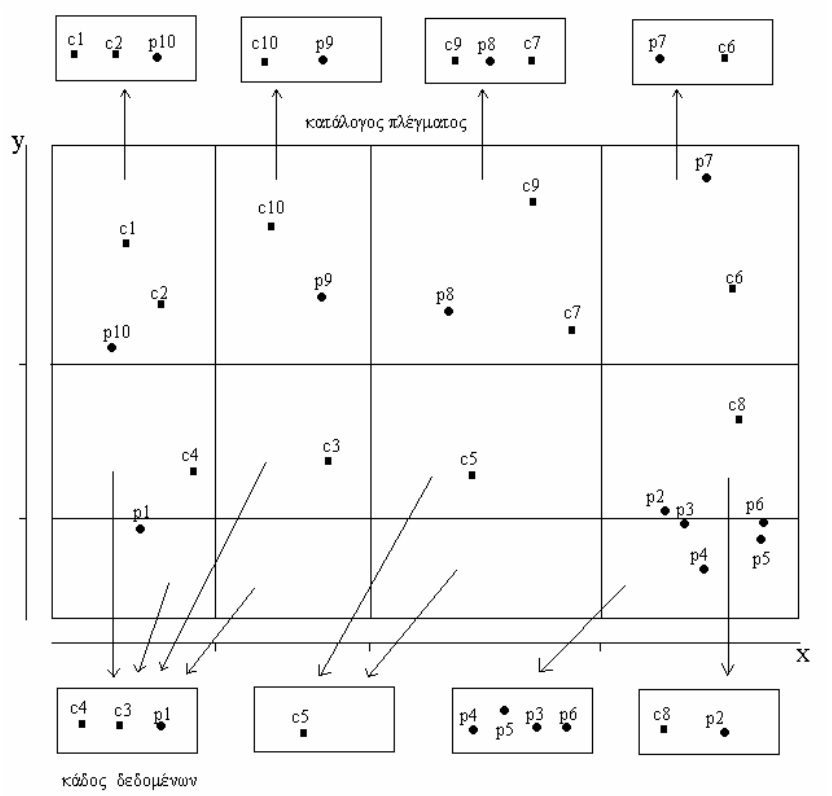
7.3.2 Μέθοδοι προσπέλασης σημείων

Αρχικά θα αναφερθούμε στις μεθόδους που στηρίζονται στο κατακερματισμό και μετά στις ιεραρχικές μεθόδους (βασισμένες σε δομές δέντρων).

Το αρχείο πλέγματος (Grid file) και το αρχείο πλέγματος διπλού σταδίου (two level grid file)

Το *αρχείο πλέγματος* [NHS84] υπερθέτει ένα κ-διάστατο ορθογωνικό πλέγμα στον κ-διάστατο χώρο. Τα κελιά του πλέγματος μπορούν να έχουν διαφορετικό μέγεθος και σχήμα. Ένας κατάλογος συσχετίζει ένα ή περισσότερα από τα κελιά με κάδους δεδομένων (data buckets) οι οποίοι αποθηκεύονται ο καθένας σε μια σελίδα δίσκου. Ο κατάλογος, λόγω μεγάλου εν γένει μεγέθους, αποθηκεύεται σε δευτερεύουσα μνήμη. Πάντως, το πλέγμα αποθηκεύεται στην κύρια μνήμη για λόγους απόδοσης (εξασφάλιση ότι θα έχουμε το πολύ δύο προσπελάσεις στο δίσκο για να βρούμε ένα δεδομένο). Ο μέσος χρησιμοποιούμενος χώρος του αρχείου πλέγματος έχει βρεθεί ότι ισούται με 69%.

Υποθέτουμε για το παράδειγμα μας ότι κάθε κάδος μπορεί να αποθηκεύσει μέχρι και τέσσερα σημεία. Στο σχήμα 7.6 φαίνεται το αρχείο πλέγματος που προκύπτει από το γενικό παράδειγμα.

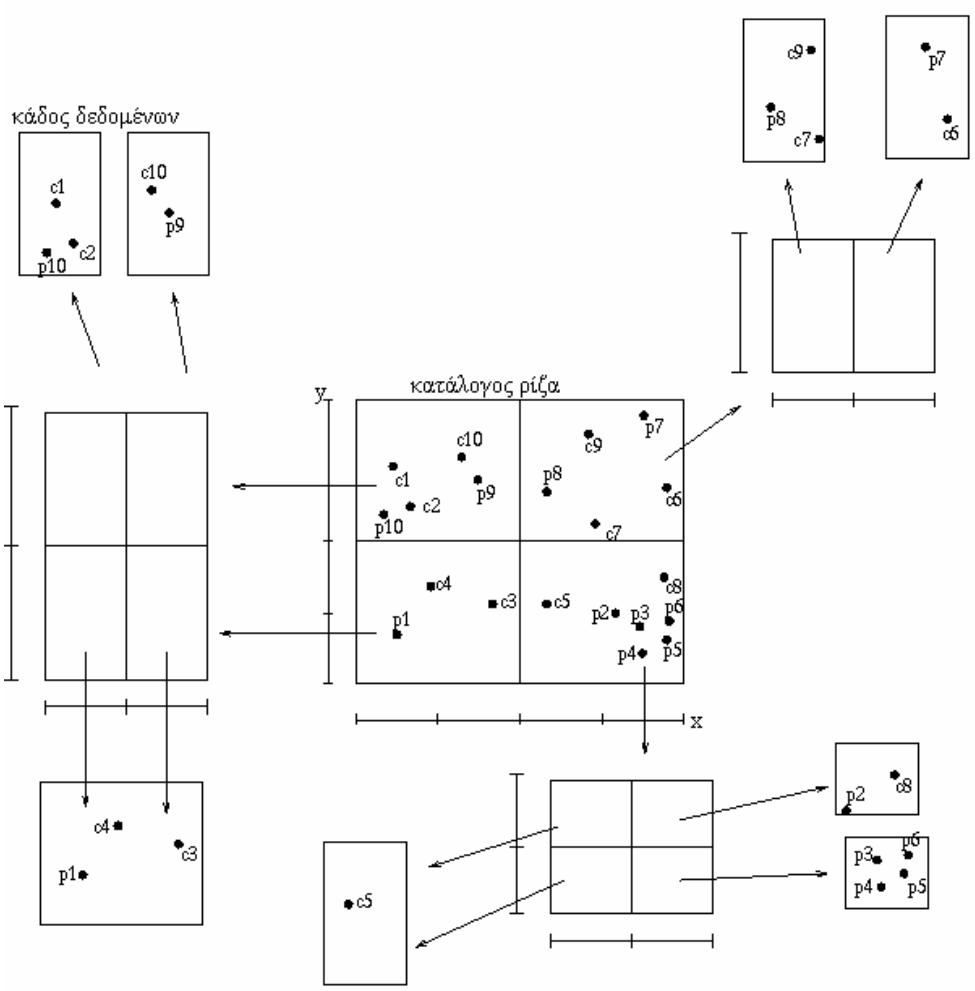


Σχήμα 7.6: Αρχείο πλέγματος

Για να απαντηθεί μια ερώτηση ισότητας, πρώτα βρίσκουμε το κελί του τρέχοντος σημείου (τα αντίστοιχα x και y). Αν το κελί δεν είναι στη κύρια μνήμη τότε χρειαζόμαστε μια προσπέλαση στο δίσκο. Στη συνέχεια, με χρήση του δείκτη που περιέχει το κελί, βρίσκουμε με νέα προσπέλαση στο δίσκο το τρέχον δεδομένο και ελέγχουμε για τυχόν ισότητα.

Για να εισάγουμε ένα σημείο πρώτα εκτελούμε μια ερώτηση ισότητας για να βρούμε σε πιο κελί πρέπει να εισαχθεί το σημείο. Αν υπάρχει χώρος στην σελίδα, έστω v_i, τότε γίνεται εισαγωγή. Αλλιώτικα, με χρήση μιας γραμμής η οποία ενημερώνει και τον αντίστοιχο άξονα x ή y, κάνουμε διαχωρισμό και φτιάχνουμε μια νέα σελίδα v_j και ισομοιράζουμε το καινούργιο δεδομένο και τα δεδομένα της v_i στις v_i και v_j. Ο διαχωρισμός προκαλεί προφανώς και σε άλλα κελιά αλλαγές πράγμα που σημαίνει ότι η εισαγωγή δεν είναι τοπική διαδικασία υπό την έννοια ότι ενδέχεται να έχει επιπτώσεις και σε άλλα κελιά. Το ίδιο ισχύει και για τη διαγραφή. Το γεγονός αυτό μπορεί να οδηγήσει σε εκθετική αύξηση του καταλόγου.

Συγγενική μέθοδος είναι το *αρχείο πλέγματος διπλού σταδίου* (two level grid file) [Hi85], το οποίο χρησιμοποιεί ένα δεύτερο πλέγμα για να διαχειριστεί τον κατάλογο πλέγματος. Στο πρώτο στάδιο έχουμε τον κατάλογο ρίζα ο οποίος είναι μια χοντρική αναπαράσταση του πραγματικού πλέγματος. Ο κατάλογος ρίζα περιέχει δείκτες στους καταλόγους του δευτέρου σταδίου οι οποίοι με τη σειρά τους περιέχουν δείκτες στις σελίδες δεδομένων. Κατά τα άλλα, η λογική που ακολουθείται είναι ταυτόσημη με αυτή του αρχείου πλέγματος, όπως φαίνεται και από το σχήμα 7.7.

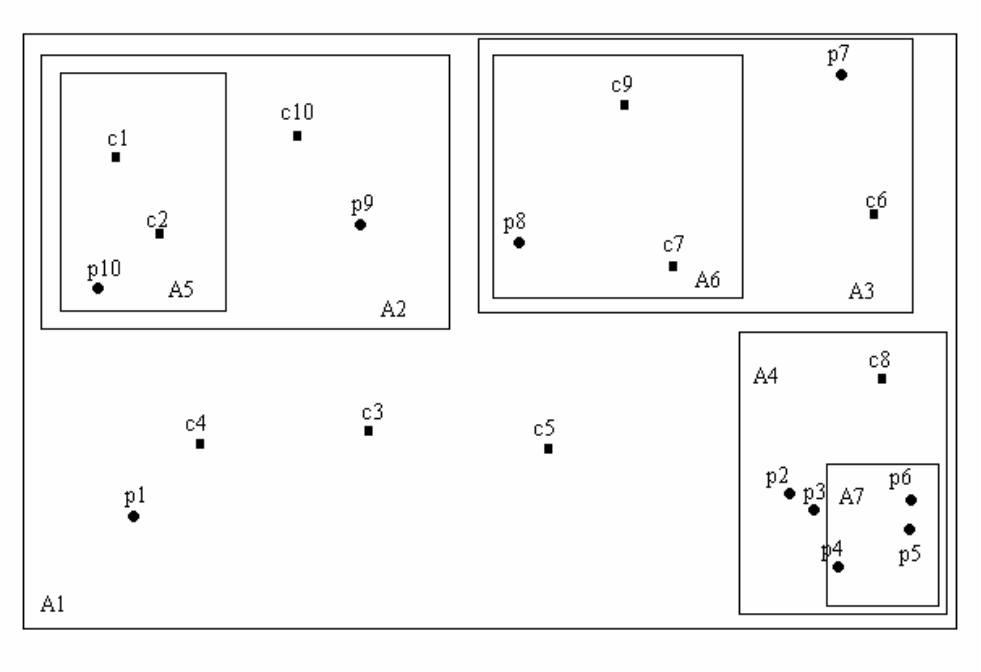


Σχήμα 7.7: Αρχείο πλέγματος διπλού σταδίου

Το αρχείο ισοσταθμισμένου και φωλιασμένου πλέγματος (BANG file)

Το *αρχείο ισορροπημένου και φωλιασμένου πλέγματος* (**b**alanced **a**nd **n**ested **g**rid file - BANG file) [Fr87] επιχειρεί να δώσει λύση στο πρόβλημα της εκθετικής αύξησης του μεγέθους του καταλόγου του αρχείου πλέγματος. Το BANG αρχείο χωρίζει και αυτό το χώρο σε υποχώρους. Η διαφορά του έγκειται στο ότι επιτρέπει οι υποχώροι αυτοί να τέμνονται. Μάλιστα είναι δυνατόν να δημιουργηθούν μη παραλληλόγραμμοι υποχώροι με τη γεωμετρική διαφορά ανάμεσα σε δύο παραλληλόγραμμους.

Για τη διαχείριση του προκύπτοντος καταλόγου, το BANG αρχείο χρησιμοποιεί ισοσταθμισμένες δεντρικές δομές αναζήτησης. Το BANG αρχείο για το παράδειγμα μας φαίνεται στο σχήμα 7.8. Παρατηρούμε ότι μέσα στον αρχικό μας χώρο A1 έχουμε τους υποχώρους A2, A3, A4 και στη συνέχεια τα παραλληλόγραμμα A5, A6, A7 είναι φωλιασμένα μέσα στα A2, A3, A4 αντίστοιχα.

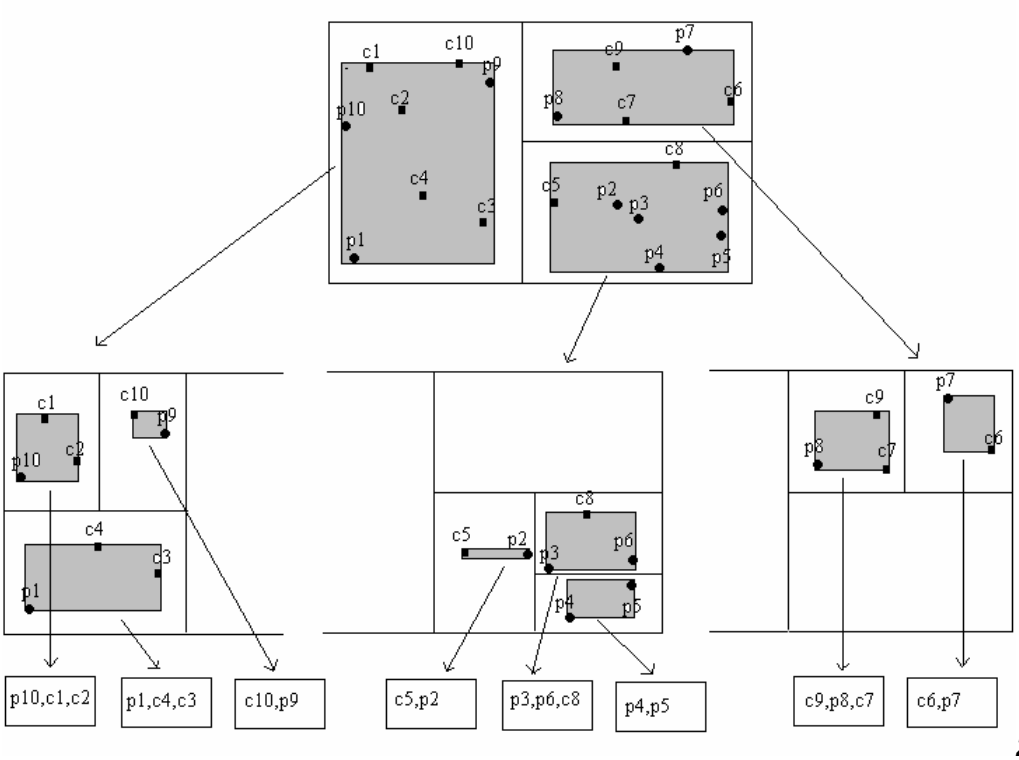


Σχήμα 7.8: Το BANG αρχείο

Το δέντρο φίλος (The buddy Tree)

Το *δέντρο φίλος* [SK90] στηρίζεται σε δυναμικό κατακερματισμό -ο δε κατάλογος του είναι δομημένος ως δέντρο. Με άλλα λόγια πρόκειται για μια υβριδική μέθοδο. Το δέντρο κατασκευάζεται με συνεχόμενες υποδιαιρέσεις του χώρου σε δύο μέρη ίσων διαστάσεων με χρήση γραμμών παράλληλων στους άξονες του αρχικού χώρου. Κάθε εσωτερικός κόμβος αντιστοιχεί σε ένα διδιάστατο μέρος Μ του αρχικού χώρου και σε ένα διάστημα Ι το οποίο αποτελεί το ελάχιστο περιβάλλον κουτί του παραπάνω μέρους. Τα μέρη που αντιστοιχούν σε κόμβους του ίδιου επιπέδου δεν έχουν κοινά σημεία και το σύνολο τους αποτελεί ολόκληρο το χώρο του συγκεκριμένου επιπέδου. Στα φύλλα του δέντρου έχουμε δείκτες προς τα δεδομένα. Αξίζει να σημειωθούν και τα εξής: (α) κάθε κόμβος περιέχει τουλάχιστον δύο δεδομένα, (β) τα Ι επαναπροσδιορίζονται όποτε έχουμε διαχωρισμό ενός κόμβου (προφανώς λόγω υπερχείλισης) και τέλος, (γ) υπάρχει μόνο ένας δείκτης για κάθε σελίδα του καταλόγου εκτός από την περίπτωση της ρίζας του καταλόγου.

Λόγω των παραπάνω, το δέντρο φίλος δεν είναι απαραίτητα ισοσταθμισμένο και επίσης εξασφαλίζεται ότι ο κατάλογος θα μεγαλώνει γραμμικά. Στο σχήμα 7.9 φαίνεται το δέντρο φίλος για το παράδειγμα μας. Τέλος, το δέντρο φίλος έχει -κατόπιν προσομοιώσεων- αποδειχτεί μία από τις καλύτερες μεθόδους προσπέλασης σημείων.



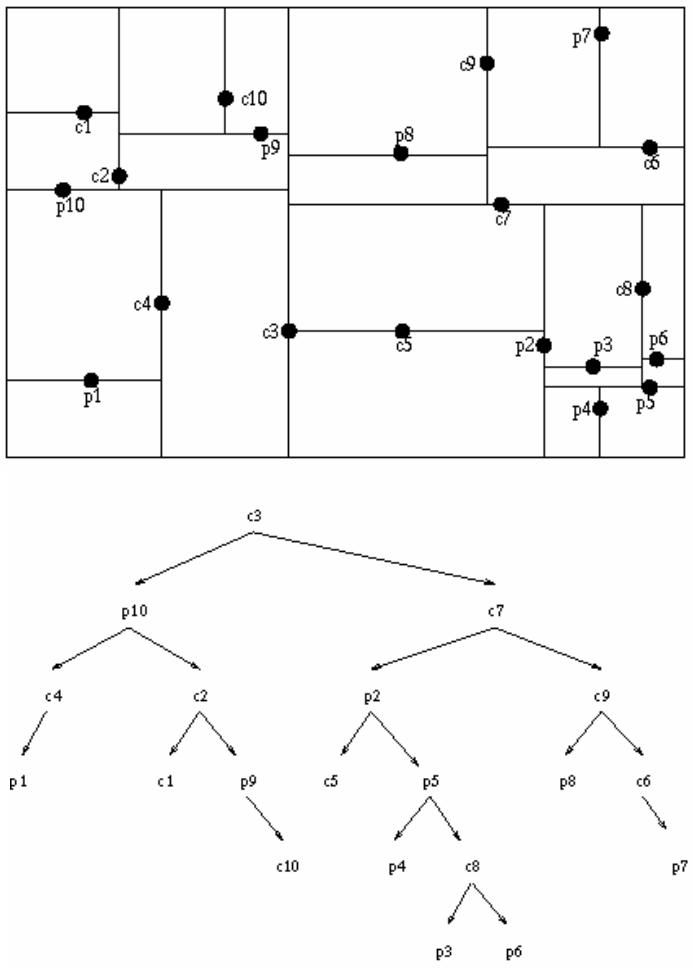
Σχήμα 7.9:

Το δέντρο φίλος

Μέχρι τώρα μιλήσαμε μόνο για μεθόδους βασισμένες στον κατακερματισμό, είτε εξολοκλήρου, είτε κατά κύριο λόγο. Τώρα θα αναφερθούμε σε ιεραρχικές μεθόδους προσπέλασης οι οποίες δεν κάνουν υπολογισμό κάποιας διεύθυνσης. Όπως και πριν, έχουμε οργάνωση των σημείων - δεδομένων σε κάδους η οποίοι συνήθως αντιστοιχούν σε φύλλα δέντρου και σε μια σελίδα δίσκου. Οι εσωτερικοί κόμβοι χρησιμοποιούνται στην αναζήτηση και αντιστοιχούν σε υποχώρους του αρχικού χώρου.

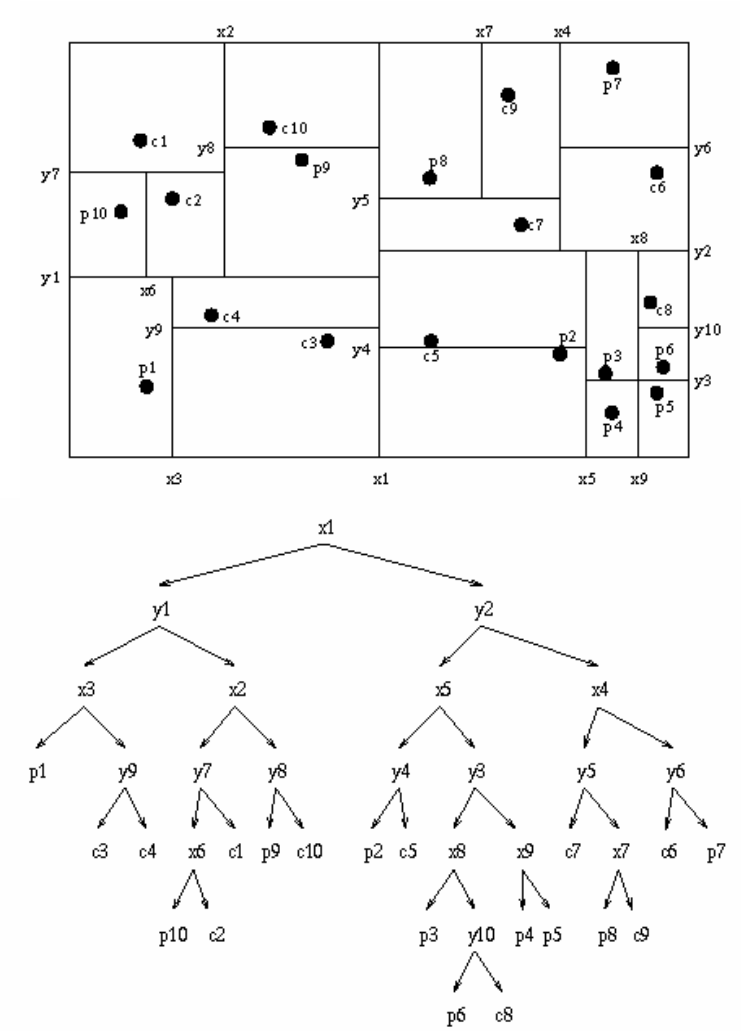
Το K-D-B-δέντρο

Αρχικά θα αναφερθούμε στο *K-D-δέντρο* [Be75], το οποίο είναι μία δομή κύριας μνήμης. Το K-D-δέντρο είναι ένα δυαδικό δέντρο το οποίο παριστάνει τη διαδοχική υποδιαίρεση του χώρου σε υποχώρους μέσω υπερεπιπέδων (γενική περίπτωση) διάστασης D-1. Στη διδιάστατη περίπτωση έχουμε γραμμές που χωρίζουν το επίπεδο και οι οποίες είναι παράλληλες στους άξονες του επιπέδου. Κάθε γραμμή οφείλει να έχει τουλάχιστον ένα από το σημεία-δεδομένα πάνω της. Στο παράδειγμα μας τα πολύγωνα τα παριστάνουμε με τα κέντρα τους μιας και το δέντρο μας χειρίζεται μόνο σημεία. Το σχήμα 7.10 δείχνει το K-D-δέντρο για το παράδειγμα μας. Ξεκινάμε από το c3. Φέρνουμε γραμμή κάθετη που περνάει από αυτό και μετά κάνουμε το ίδιο στους δύο χώρους που προκύπτουν με οριζόντιες γραμμές αυτή τη φορά στα p10 και c7 κ.ο.κ. Είναι υποχρεωτικό να έχουμε εναλλάξ οριζόντιες και κάθετες γραμμές.



Σχήμα 7.10: K-D-δέντρο

Το βασικό πρόβλημα του παραπάνω δέντρου είναι ότι εξαρτάται έντονα από τη σειρά με την οποία θα έρθουν τα σημεία, μιας και, αν αυτή αλλάξει, αλλάζει και το δέντρο. Το *προσαρμοζόμενο K-D-δέντρο* [Be79] είναι μια παραλλαγή του που λύνει αυτό ακριβώς το πρόβλημα. Το δέντρο αυτό χωρίζει κάθε φορά τους υποχώρους με τέτοιο τρόπο ώστε να βρίσκει κανείς σχεδόν τον ίδιο αριθμό σημείων σε κάθε προκύπτον υποχώρο. Οι γραμμές είναι και πάλι παράλληλες στους άξονες του επιπέδου όμως τώρα δεν χρειάζεται να περιλαμβάνουν ένα τουλάχιστον σημείο. Οι εσωτερικοί κόμβοι κρατάνε την τετμημένη ή τεταγμένη των γραμμών και τα σημεία-δεδομένα αποθηκεύονται στα φύλλα του δέντρου. Στο σχήμα 7.11 φαίνεται το προσαρμοσμένο K-D-δέντρο για το υπό εξέταση παράδειγμα.



Σχήμα 7.11 Προσαρμοσμένο K-D-δέντρο

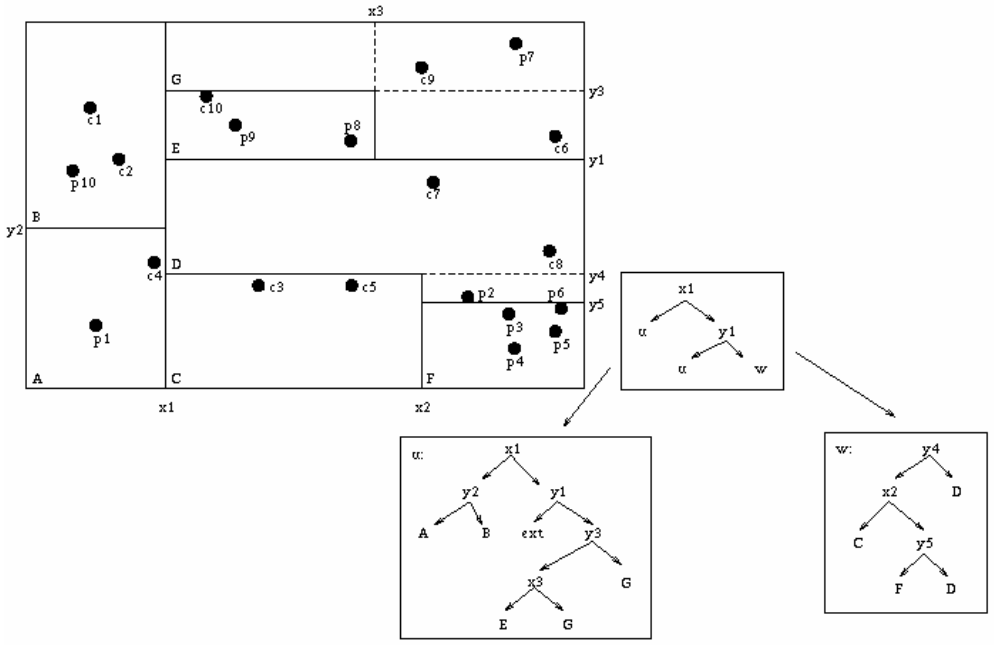
Το *K-D-B-δέντρο* [Ro81] συνδυάζει τις ιδιότητες του προσαρμοσμένου K-D-δέντρου με αυτές του B-δέντρου. Χωρίζει το χώρο όπως το πρώτο σε υποχώρους τους οποίους αντιστοιχεί στους εσωτερικούς κόμβους του και είναι πλήρως ισοσταθμισμένο όπως το δεύτερο. Τόσο η εισαγωγή νέου στοιχείου όσο και η διαγραφή γίνεται σύμφωνα με τους αλγορίθμους του K-D-δέντρου και είναι άμεσες διαδικασίες. Επι παραδείγματι, όσον αφορά την εισαγωγή, αρχικά γίνεται ψάξιμο για να βρεθεί η κατάλληλη θέση και μετά το στοιχείο εισάγεται εφόσον χωράει στον κάδο που επιλέχθηκε. Αν δεν χωράει, έχουμε σπάσιμο και δημιουργία νέου κόμβου (θυμίζουμε ότι οι κόμβοι αντιστοιχούν σε κάδους), ο οποίος μοιράζεται με τον παλιό τα δεδομένα κατά το ήμισυ. Βέβαια, πρέπει και ο πρόγονος κόμβος να χωράει ένα νέο απόγονο κόμβο για να γίνει το παραπάνω. Αν δεν τον χωράει, έχουμε την ίδια διαδικασία με πριν να μεταδίδεται προς τη ρίζα του δέντρου.

Το hB-δέντρο

Το *hB-δέντρο* (**h**oley **b**rick tree) [LS90] έχει μια σημαντική διαφορά από τα προηγούμενα. Το σπάσιμο ενός κόμβου σε δύο στηρίζεται σε πολλά χαρακτηριστικά (ενώ μέχρι τώρα έχουμε

δει μόνο το ενδεχόμενο να γεμίσει ο κόμβος). Οι κόμβοι δεν αντιστοιχούν σε παραλληλόγραμμα αλλά σε περιοχές που προκύπτουν από την αφαίρεση μικρότερων παραλληλογράμμων από μεγαλύτερα, όπως περίπου συμβαίνει και με το BANG αρχείο. Η μέθοδος παίρνει το όνομα της από αυτό ακριβώς το χαρακτηριστικό, το οποίο έχει ως αποτέλεσμα να αποφεύγεται η κλιμάκωση των διαχωρισμών των κόμβων.

Οι εσωτερικοί κόμβοι του δέντρου χρησιμοποιούν K-D-δέντρα για να απεικονίσουν τον χώρο στον οποίο αντιστοιχούν. Επιτρέπεται σε αυτούς να έχουν περισσότερα από ένα φύλλα που δείχνουν στον ίδιο απόγονο κόμβο με αποτέλεσμα να μην έχουμε τώρα πια, ένα δέντρο με την αυστηρή έννοια, παρά μάλλον ένα προσανατολισμένο ακυκλικό γράφο. Στο σχήμα 7.12 φαίνεται το hB-δέντρο για το παράδειγμα μας. Είναι φανερό πως οι υποχώροι δεν είναι αναγκαστικά παραλληλόγραμμα. Η ρίζα έχει δύο δείκτες σε απογόνους-κόμβους αντίστοιχους στους υποχώρους u και w. Στο εσωτερικό του αριστερού απόγονου-κόμβου παρατηρούμε ότι γράφει ext πράγμα που δηλώνει ότι κάτω από το y1 υπάρχει μια περιοχή που δεν ανήκει στον υποχώρο u.



Σχήμα 7.12: hB-δέντρο

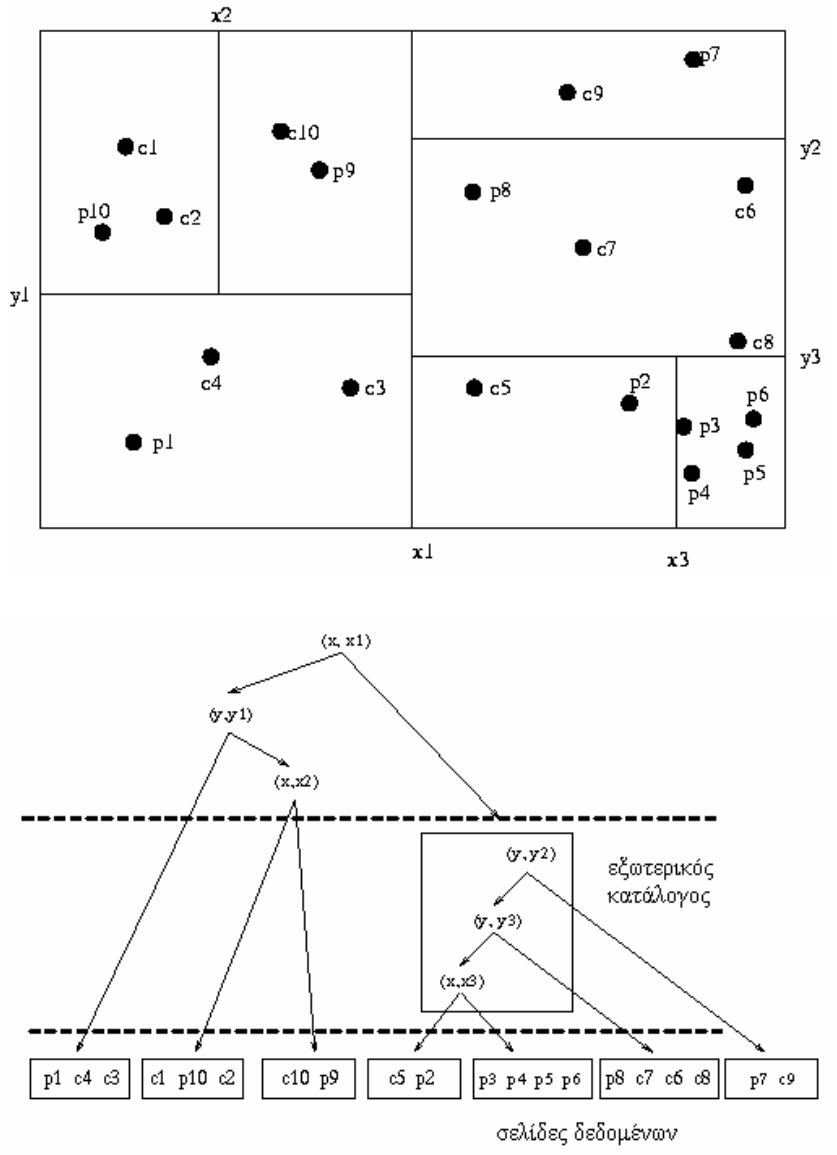
Όπως προκύπτει και από το σχήμα, τα φύλλα των κόμβων του δέντρου μπορούν να δείχνουν είτε σε εγγραφές με δεδομένα (π.χ. στο A), είτε σε άλλους κόμβους του δέντρου (π.χ. στο u), είτε τέλος να υποδηλώνουν ένα αποκομμένο μέρος με χρήση του ext. Το παραπάνω δέντρο έχει πολλά πλεονεκτήματα σε σχέση με τα άλλα όμως αυτή η μη κανονική κατασκευή του μπορεί να προκαλέσει προβλήματα σε ορισμένες εφαρμογές.

Το LSD δέντρο

Το *δέντρο τοπικής απόφασης διάσπασης* (**l**ocal **s**plit decision **t**ree - LSD tree) [HSW89] έχει ένα κατάλογο οποίος οργανώνεται με χρήση του προσαρμοζόμενου K-D-δέντρου και χωρίζει το χώρο σε κελιά (υποχώρους) διαφόρων διαστάσεων. Αυτό έχει ως αποτέλεσμα την καλύτερη προσαρμογή της μεθόδου στις διάφορες πιθανές κατανομές των δεδομένων. Το δέντρο αυτό

είναι ισοσταθμισμένο, πράγμα που επιτυγχάνεται με χρήση ενός ειδικού αλγορίθμου, σύμφωνα με τον οποίο, αν έχουμε απαγορευτικά μεγάλο μέγεθος του καταλόγου για να διατηρηθεί στην κύρια μνήμη, επιλέγονται κάποια υποδέντρα τα οποία αποθηκεύονται σε εξωτερική σελίδα έτσι ώστε να έχουμε ισοσταθμισμένο δέντρο. Αυτό φαίνεται καλύτερα στο σχήμα που ακολουθεί.

Η ισχύς της παραπάνω δομής έγκειται στο ότι όχι μόνο δεν θεωρεί ότι τα δεδομένα ακολουθούν ομοιόμορφη κατανομή, αλλά και λαμβάνει υπόψιν της τυχόν ακανόνιστα δεδομένα. Η επιλογή της θέσης όπου θα γίνει ο διαχωρισμός του χώρου SP γίνεται με χρήση του τύπου: $SP = a \cdot SP1 + (1-a) \cdot SP2$, όπου a μια σταθερά εξαρτώμενη από τη φύση των δεδομένων, SP1 η θέση που θα επιλεγόταν αν κάθε φορά επιχειρούσαμε ισομοίρασμα των δεδομένων και SP2 η θέση που θα επιλεγόταν αν ξέραμε εκ των προτέρων την κατανομή που ακολουθείται.



Σχήμα 7.13: LSD δέντρο

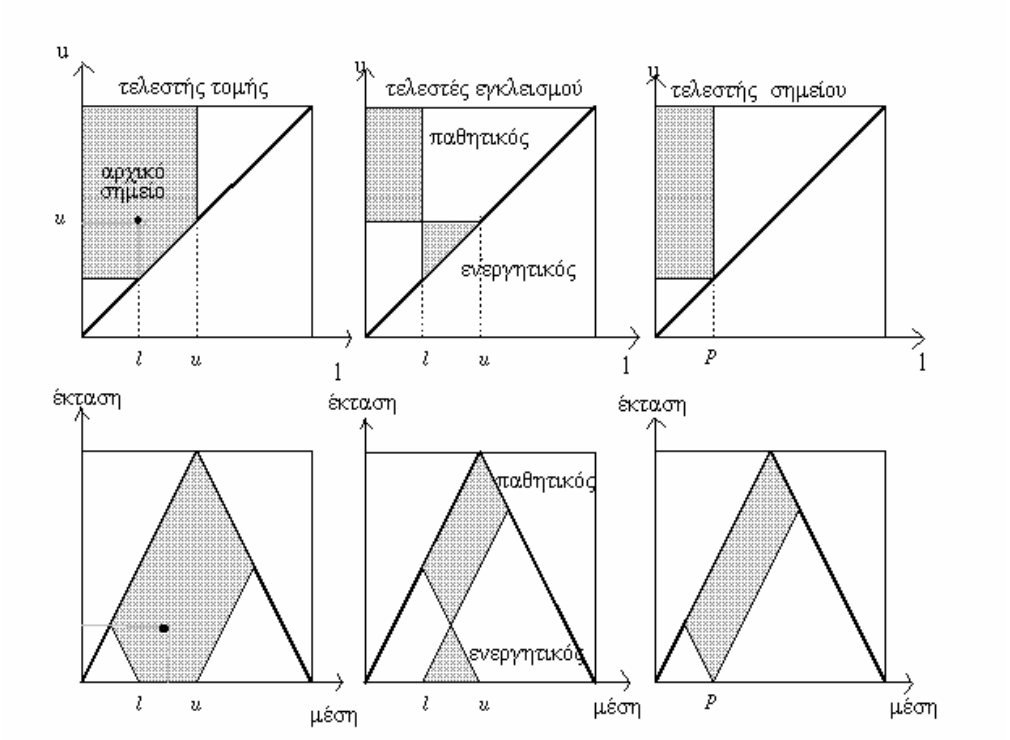
7.3.3 Μέθοδοι προσπέλασης περιοχών

Ο χειρισμός μη-σημειακών χωρικών δεδομένων με έκταση απαιτεί ιδιαίτερες τεχνικές λόγω της ιδιαίτερης φύσης των δεδομένων. Τρεις είναι οι βασικές τεχνικές που χρησιμοποιούνται [SK88] και εμείς θα αναφερθούμε σύντομα και στις τρεις.

Η τεχνική *μετασχηματισμού*.

Σύμφωνα με την τεχνική αυτή, ένα αντικείμενο με έκταση στο διδιάστατο χώρο (η τεχνική γενικεύεται και στον κ-διάστατο χώρο) μπορεί να καθοριστεί πλήρως με στοιχεία του μονοδιάστατου χώρου. Για να είναι αποδοτική μια τέτοια διαδικασία πρέπει τα αντικείμενο να έχει σχετικά απλό σχήμα. Επί παραδείγματι, ένα ορθογώνιο παραλληλόγραμμο μπορεί να παρασταθεί από δύο απέναντι κορυφές (άρα από ένα τετραδιάστατο σημείο) και άρα μπορούμε να χρησιμοποιήσουμε τις προηγούμενες μεθόδους που αφορούν σημεία.

Σαν αναλυτικό παράδειγμα θα δώσουμε ένα διάστημα του μονοδιάστατου χώρου, το οποίο μετασχηματίζεται σε ένα σημείο του διδιάστατου. Το σημείο μπορεί να έχει ως συνιστώσες, είτε την αρχή και το τέλος του διαστήματος, είτε τη μέση και την έκταση του. Στο παρακάτω σχήμα δείχνουμε πως μετασχηματίζεται ο τελεστής τομής καθώς και οι τελεστές εγκλεισμού κατόπιν των παραπάνω μετασχηματισμών. Αν το αρχικό μας διάστημα είναι το $[l,u]$, αν δηλαδή αρχίζει στο l και τελειώνει στο u, τότε η μέση ισούται με $(l+u)/2$ και η έκταση με $u-l$. Το γραμμοσκιασμένο τμήμα στην πρώτη περίπτωση (τελεστής τομής) δείχνει το χώρο μέσα στον οποίο αν βρίσκεται ένα σημείο, έχει τομή με το αρχικό. Όμοια για τους άλλους δύο τελεστές.



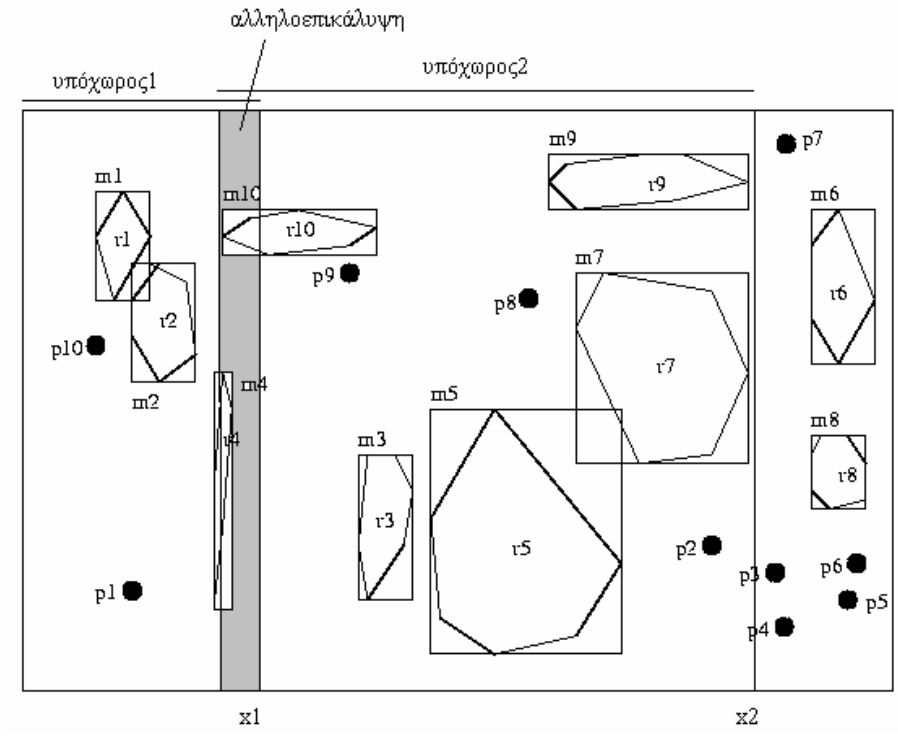
Σχήμα 7.14: Μετασχηματισμοί

Αξίζει πάντως να σημειωθεί ότι αυτός ο τρόπος αντιμετώπισης του προβλήματος έχει αρκετά μειονεκτήματα όπως μεγαλύτερη πολυπλοκότητα, αλλαγή της κατανομής που

ακολουθείται από τα δεδομένα και αλλαγή της απόστασης ανάμεσα σε δύο δεδομένα τα οποία ενώ πριν τον μετασχηματισμό ήταν κοντά μπορεί μετά από αυτόν να βρίσκονται αυθαίρετα μακριά.

Η τεχνική *αλληλοεπικαλυπτόμενων περιοχών*.

Η δεύτερη τεχνική στηρίζεται στην ιδέα του να αφήνουμε διαφορετικούς κάδους δεδομένων να αντιστοιχούν σε *αλληλοεπικαλυπτόμενες περιοχές* (overlapping regions) του χώρου. Ας το δούμε με ένα παράδειγμα. Έστω το πολύγωνο r10 του σχήματος 7.15. Παρατηρούμε ότι το πολύγωνο αυτό ανήκει και στην δεξιά και στην αριστερή περιοχή από την διαχωριστική γραμμή x1. Προκειμένου να αποθηκεύσουμε ολόκληρο το πολύγωνο σε έναν από τους δύο αντίστοιχους κάδους (έστω τον αριστερό) θα πρέπει να μεγαλώσουμε τον αντίστοιχο υπόχωρο, τότε όμως θα έχουμε αλληλοεπικάλυψη δηλαδή θα υπάρχει μια περιοχή που θα αντιστοιχεί και στους δύο κάδους. Σε μια τέτοια περίπτωση, οι μέθοδοι που αναπτύξαμε μέχρι τώρα είναι άμεσα εφαρμόσιμες. Μολαταύτα, ενδέχεται να έχουμε πρόβλημα στην απόδοση, αφού, τώρα, κατά την προσπέλαση ενός στοιχείου, ίσως να χρειαστεί να ακολουθήσουμε παραπάνω από έναν δρόμους, λόγω ακριβώς της αλληλοεπικάλυψης (διότι μπορεί π.χ. ένα σημείο να περιέχεται σε πολλούς υποχώρους).



Σχήμα 7.15: Αλληλοεπικάλυψη χώρων

Τεχνική *αποκοπής*.

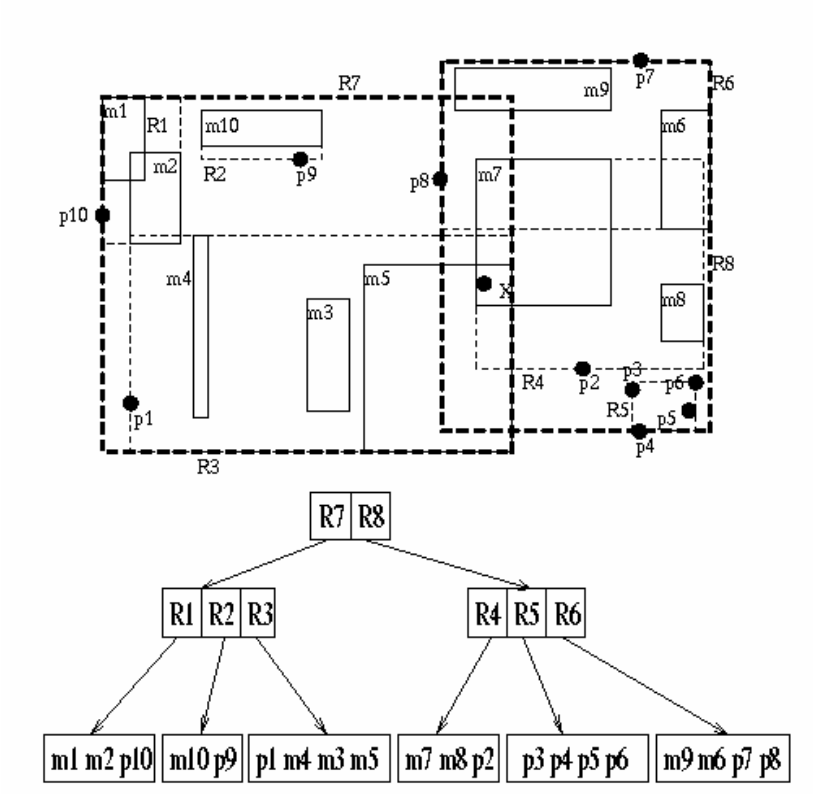
Τέλος, έχουμε την τεχνική *clipping* (τεχνική αποκόματος). Σύμφωνα με την τεχνική αυτή δεν έχουμε ποτέ αλληλοεπικάλυψη. Σε περίπτωση όπου ένα αντικείμενο εκτείνεται σε περισσότερες από μία περιοχές, είτε το σπάμε και αποθηκεύουμε κάθε κομμάτι του σε διαφορετικό κάδο, είτε το αποθηκεύουμε ολόκληρο σε κάθε έναν από τους κάδους.

Το βασικό πρόβλημα αυτής της μεθόδου είναι ακριβώς ότι τα δεδομένα διασκορπίζονται σε διαφορετικές σελίδες δεδομένων με αποτέλεσμα και η απόδοση να μειώνεται και να έχουμε πιο συχνές υπερχειλίσεις των κάδων. Ένα δεύτερο πρόβλημα οφείλεται στα γεγονός ότι κάθε φορά που εισάγουμε ένα αντικείμενο που δεν χωράει ολόκληρο σε έναν υπόχωρο πρέπει να μεγαλώσουμε τον υπόχωρο. Αν αυτό οδηγεί σε αλληλοεπικάλυψη πρέπει να σπάσουμε το αντικείμενο και να εισάγουμε τα κομμάτια του. Όμως, και αυτά με τη σειρά τους πρέπει να εισαχθούν σε κάδους όπου αν δεν χωράνε θα χρειαστεί να χωριστούν κ.ο.κ.

Συνολικά, πάνω από εκατό μέθοδοι προσπέλασης περιοχών έχουν προταθεί τα τελευταία 15 χρόνια. Στη συνέχεια θα αναφερθούμε στις πιο σημαντικές από αυτές.

Το R-δέντρο

Το *R-δέντρο* [Gu84] αντιστοιχεί σε μια δομή φωλιασμένων διδιάστατων διαστημάτων. Κάθε κόμβος ν του δέντρου αντιστοιχεί σε μια σελίδα δίσκου και σε ένα διδιάστατο διάστημα I(ν). Τα διαστήματα που βρίσκονται στο ίδιο επίπεδο μπορούν να αλληλοεπικαλύπτονται. Οι απόγονοι του εσωτερικού κόμβου ν αντιστοιχούν σε περιοχές που ανήκουν στο I(ν). Αν ο κόμβος ν είναι φύλλο, το I(ν) είναι το ελάχιστο περιβάλλον κουτί των αντικειμένων που είναι αποθηκευμένα στον κόμβο με τη μορφή των ελάχιστων περιβαλλόντων κουτιών τους. Επίσης, αποθηκευμένος είναι και ένας δείκτης στο πλήρες αντικείμενο για κάθε αντικείμενο.



Σχήμα 7.16: R-δεντρο

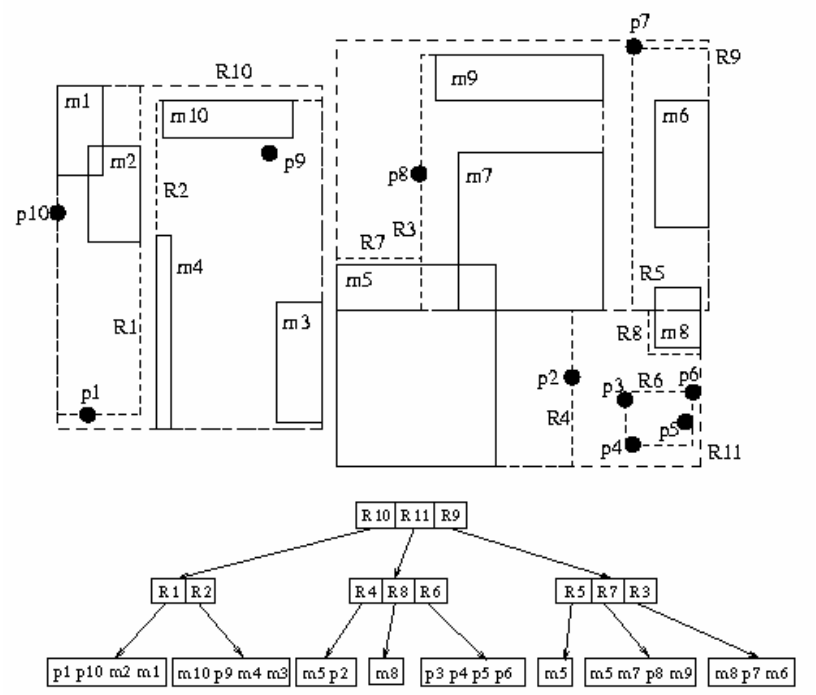
Κάθε κόμβος εκτός της ρίζας περιέχει από m μέχρι M στοιχεία, η δε ρίζα έχει τουλάχιστον δύο στοιχεία εκτός και αν είναι φύλλο. Το δέντρο είναι ισοσταθμισμένο. Από τα παραπάνω προκύπτει ότι η αναζήτηση στο R-δέντρο είναι παρόμοια με αυτή στο B-δέντρο. Λόγω όμως

της αλληλοεπικάλυψης ενδέχεται να αναγκαστούμε να ακολουθήσουμε περισσότερα από ένα μονοπάτια στο δέντρο. Αυτό φαίνεται και στο σχήμα 7.16 όπου έχουμε το R-δέντρο για το παράδειγμα μας. Στην περίπτωση του τελεστή αναζήτησης σημείου με σημείο προς αναζήτηση το x του σχήματος, η αναζήτηση θα γίνει κατά μήκος των εξής δύο μονοπατιών: R8 → R4 → m7 και R7 → R3 → m5.

Για την εισαγωγή ενός στοιχείου, εισάγουμε το ελάχιστο περιβάλλον κουτί του και ένα δείκτη στο πλήρες αντικείμενο. Ξεκινάμε από τη ρίζα και πάμε προς τα φύλλα επιλέγοντας κάθε φορά εκείνο τον κόμβο του οποίο το I(ν) χρειάζεται την ελάχιστη αύξηση. Όταν φτάσουμε σε φύλλο εισάγουμε το αντικείμενο. Αν δεν χωράει δημιουργούμε μια νέα σελίδα και μοιράζουμε τα αντικείμενα (αυτά που ήταν στην παλιά σελίδα και το νέο) στις δύο σελίδες. Κατά την διαγραφή ίσως χρειαστεί να ακολουθήσουμε πολλά μονοπάτια με αποτέλεσμα να χρειαστεί να μεταβάλλουμε πολλά διαστήματα -πράγμα που σημαίνει χαμηλή απόδοση.

Οι παραλλαγές R^+-δέντρο και R-δέντρο*

Προκειμένου να λυθούν τα προβλήματα που προκαλούνται από την δυνατότητα αλληλοεπικάλυψης στην περίπτωση του R-δέντρου, προτάθηκε το *R^+-δέντρο* [SRF87] το οποίο χρησιμοποιεί τεχνική αποκοπής (clipping). Έτσι ένα αντικείμενο ενδέχεται να διαχωριστεί και να αποθηκευτεί σε πολλούς διαφορετικούς κόμβους. Κατά την εισαγωγή ενός αντικειμένου ενδέχεται να χρειαστεί να ακολουθήσουμε πολλά μονοπάτια ανάλογα με το με πόσα διαστήματα τέμνεται το ελάχιστο περιβάλλον κουτί του αντικειμένου. Κατά την διαγραφή, αρχικά βρίσκουμε όλα τα κομμάτια του αντικειμένου και μετά τα αφαιρούμε. Αν τα στοιχεία κάποιου κόμβου μειωθούν πολύ, λόγω κάποιων διαγραφών, επιχειρούμε να τον ενώσουμε με τους γειτονικούς του. Το R^+-δέντρο για το παράδειγμα μας, δίνεται στο σχήμα 7.17.

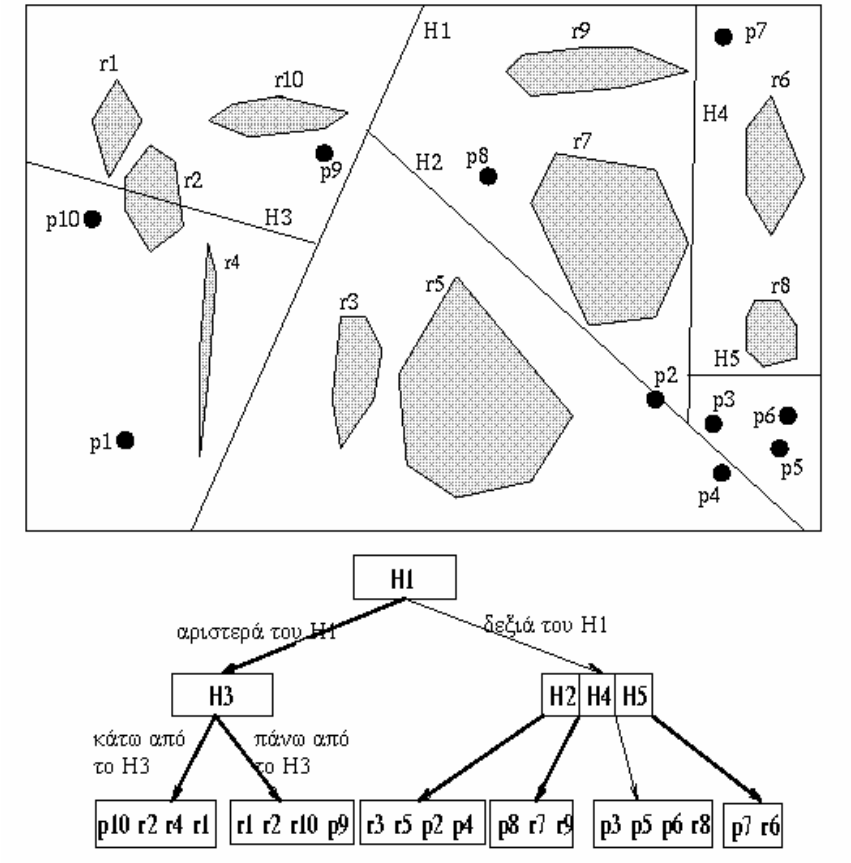


Σχήμα 7.17: Το R^+-δέντρο

Τελειώνοντας την αναφορά μας στα R-δέντρα αξίζει να σημειωθεί και η ύπαρξη του *R*-δέντρου* [BKSS90], στο οποίο έχουμε τη λεγόμενη επιβεβλημένη επανεισαγωγή. Σύμφωνα με αυτή, αν κάποιος κόμβος υπερχειλίσει, προτού τον σπάσουμε του αφαιρούμε p στοιχεία τα οποία και επανεισάγουμε με την ελπίδα να μην χρειαστεί τελικά το σπάσιμο. Τυπική τιμή για το p είναι 30% επί του συνολικού αριθμού στοιχείων του κόμβου.

Το κυτταρικό δέντρο (cell tree)

Το *κυτταρικό δέντρο* [Gu89] σχεδιάστηκε για να διευκολυνθεί ο χειρισμός αντικειμένων με αυθαίρετο σχήμα, χρησιμοποιώντας την τεχνική του clipping. Το δέντρο αντιστοιχεί στον διαμελισμό του αρχικού χώρου σε ασύνδετους (χωρίς κοινά σημεία) κυρτούς υποχώρους. Στους εσωτερικούς κόμβους του δέντρου αποθηκεύονται οι γραμμές που χωρίζουν το χώρο και στα φύλλα του δέντρου, τα οποία αντιστοιχούν στους υποχώρους, όλες οι απαραίτητες πληροφορίες για κάθε αντικείμενο που ανήκει στον αντίστοιχο υπόχωρο. Η τελευταία αυτή ιδιότητα είναι ένα πλεονέκτημα σε σχέση με το R-δέντρο, στο οποίο όπως είδαμε αποθηκεύονται τα ελάχιστα περιβάλλοντα κουτιά και όχι όλη η πληροφορία για κάθε αντικείμενο.



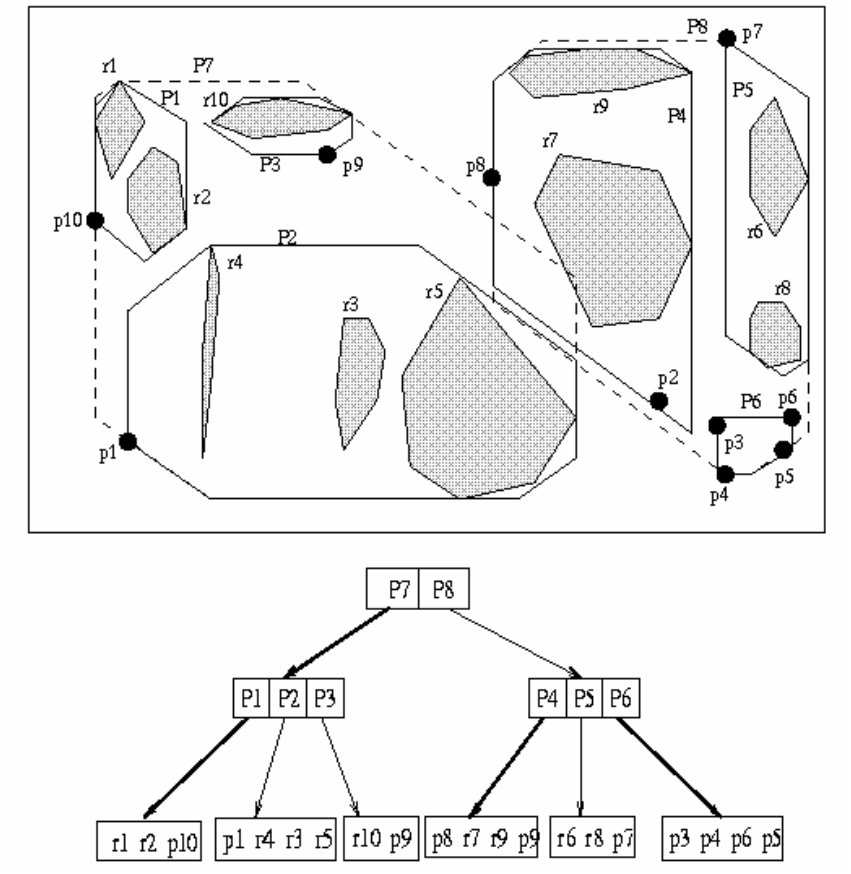
Σχήμα 7.18: Το κυτταρικό δέντρο

Προτού εισάγουμε ένα νέο αντικείμενο, το διαμελίζουμε σε κομμάτια, η ένωση των οποίων δίνει το αρχικό αντικείμενο. Τα κομμάτια εισάγονται στο δέντρο ένα - ένα. Δεδομένου του clipping είναι πιθανό να χρειαστεί περαιτέρω χωρισμός κάθε κομματιού. Αν μια σελίδα δίσκου

υπερχειλίσει, τότε πρέπει να χωρίσουμε τον αντίστοιχο υποχώρο. Στο σχήμα 7.18 δίνουμε το κυτταρικό δέντρο για το παράδειγμα μας, θεωρώντας ότι έχουμε 5 γραμμές που χωρίζουν το χώρο σε 6 υποχώρους. Βλέπουμε στο σχήμα ότι, παρόλο που τώρα οι υπόχωροι μας δεν είναι απαραίτητα ορθογώνια παραλληλόγραμμα, και πάλι κάποιες φορές δεν γίνεται να αποφύγουμε τον χωρισμό ενός αντικειμένου όπως π.χ. του r2.

Το P-δέντρο

Το *P-δέντρο* [Ja90] εισάγει έναν αριθμό από διευθύνσεις, έστω m, στον κ-διάστατο χώρο όπου m > κ. Στην περίπτωση των δύο διαστάσεων θα μπορούσαμε να έχουμε π.χ. m = 4 με δύο εκ των διευθύνσεων παράλληλες στους άξονες και άλλες δύο παράλληλες στις διαγωνίους του χώρου. Τα αντικείμενα τώρα προσεγγίζονται με ελάχιστα περιβάλλοντα κουτιά των οποίων οι πλευρές είναι παράλληλες στις m διευθύνσεις. Οι εσωτερικοί κόμβοι του δέντρου αντιστοιχούν σε μια ιεραρχία από φωλιασμένα τετράπλευρα τα οποία επιτρέπεται να αλληλοεπικαλύπτονται εφόσον βρίσκονται στο ίδιο επίπεδο. Στην πράξη, προκειμένου για διδιάστατο χώρο, μια καλή τιμή για το m (με την έννοια ότι συμβιβάζει την ανάγκη για ακρίβεια με την ανάγκη για μικρή πολυπλοκότητα) είναι το 10. Στο παρακάτω σχήμα δίνουμε το P-δέντρο που αντιστοιχεί στο παράδειγμα μας.

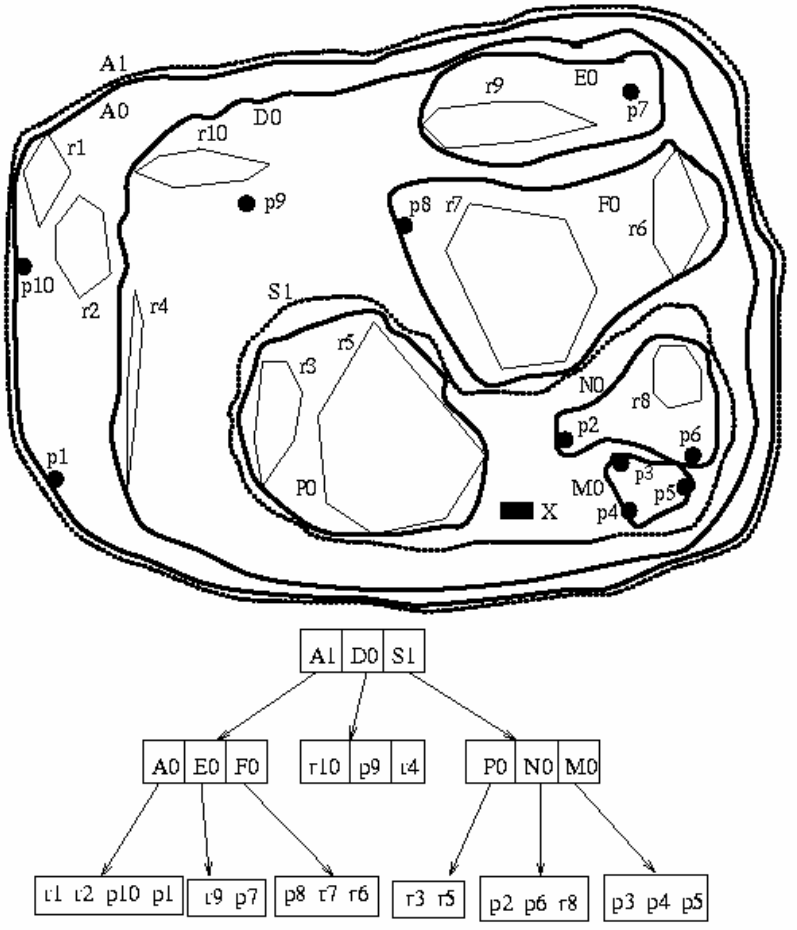


Σχήμα 7.19: Το P-δέντρο

Το BV-δέντρο

Η προσπάθεια να γενικευτεί το B-δέντρο και στη περίπτωση των πολλών διαστάσεων οδήγησε στο *BV-δέντρο* [Fr95]. Το δέντρο αυτό επιχειρεί να έχει λογαριθμική απόδοση κατά την αναζήτηση δεδομένων οποιασδήποτε κατανομής και να ικανοποιεί τουλάχιστον κάποιο ελάχιστο όριο χρησιμοποίησης της μνήμης.

Όσον αφορά τη χρησιμοποίηση, το ζητούμενο επιτυγχάνεται με το να μην έχουμε ένα αυστηρά ισοσταθμισμένο δέντρο. Μελέτες έδειξαν ότι στη χειρότερη περίπτωση η χρησιμοποίηση είναι κοντά στο 33%, που είναι ότι καλύτερο μπορούμε να έχουμε στη περίπτωση περισσότερων διαστάσεων από μία (θυμίζουμε ότι το B-δέντρο έχει στη χειρότερη περίπτωση χρησιμοποίηση 50% αλλά αυτό αφορά μονοδιάστατο χώρο). Όσον αφορά την απόδοση, το δέντρο αυτό χρησιμοποιεί μια τεχνική (η οποία ονομάζεται *posting*) κατά την οποία διαστήματα (υποχώροι) κατωτέρων επιπέδων μετακινούνται πιο κοντά στη ρίζα και η μετακίνηση αυτή αποθηκεύεται ως ένας αριθμός που ονομάζεται *φρουρός* και ο οποίος ισούται με το αρχικό επίπεδο στο οποίο βρισκόταν το διάστημα αρχικά (πριν τη μετακίνηση). Κατά την αναζήτηση δεδομένων οι φρουροί χρησιμοποιούνται για να γυρίζουμε πίσω εφόσον δεν βρούμε το προς αναζήτηση αντικείμενο και να ψάξουμε και τους υποχώρους που έχουν μετακινηθεί προς τη ρίζα. Για το λόγο αυτό οι φρουροί τοποθετούνται σε ένα σύνολο φρουρών.



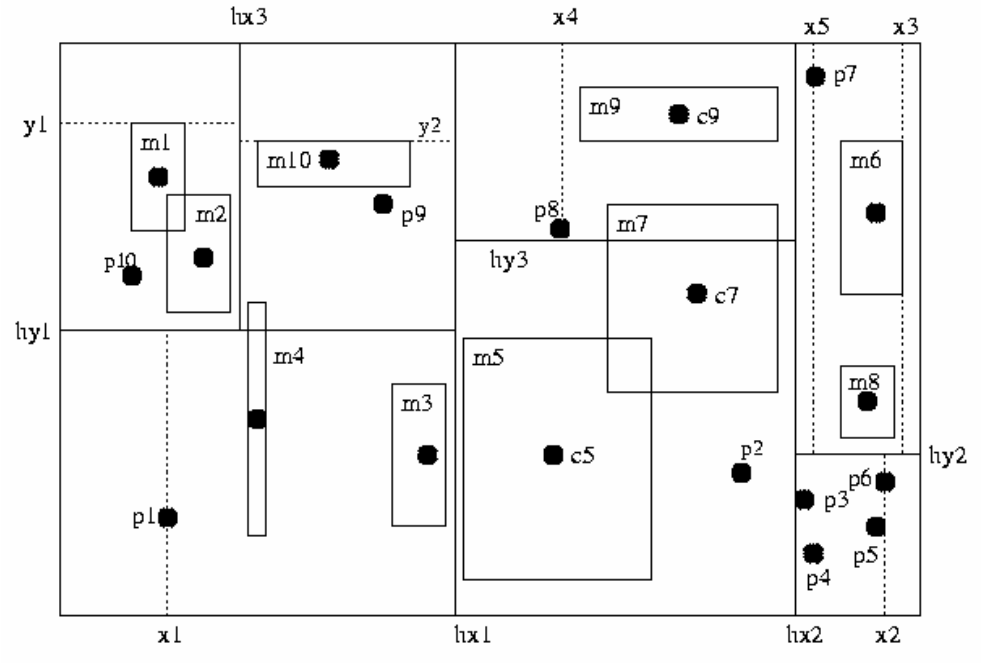
Σχήμα 7.20: Το BV-δέντρο

Έστω το παράδειγμα του σχήματος 7.20 (BV-δέντρο για το παράδειγμα μας) και έστω ότι αναζητούμε τα αντικείμενα με μη μηδενικές τομές με το ορθογώνιο x. Στο σχήμα, στο πάνω επίπεδο (επίπεδο ρίζας) αντιστοιχούν τα A1 και S1 (διακεκομμένες γραμμές) ενώ στο κάτω τα A0, D0, E0, F0, P0, N0 και M0 (συνεχείς γραμμές). Το D0 έχει μετακινηθεί προς τη ρίζα και άρα δρα σαν φρουρός. Κατά τη διαδικασία εύρεσης των τομών, βάζουμε το D0 στο σύνολο των φρουρών και μετά πάμε στην S1 και μετά στις P0, N0, M0 χωρίς κανένα αποτέλεσμα. Στη συνέχεια, με χρήση του φρουρού ξαναγυρνάμε στο πάνω επίπεδο και εξετάζουμε τα δεδομένα του D0. Στη συγκεκριμένη περίπτωση δεν βρίσκουμε τομή με κανένα αντικείμενο.

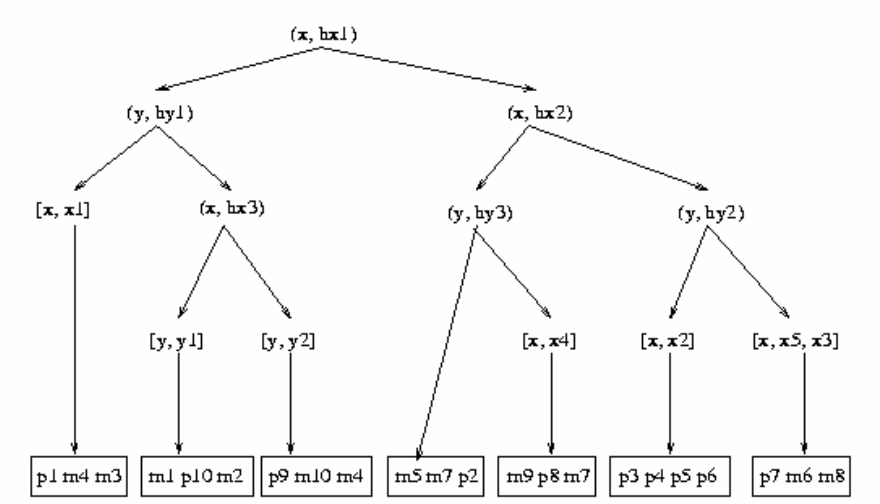
Το εκτεταμένο K-D-δέντρο (extended K-D-tree)

Το *εκτεταμένο K-D-δέντρο* [MHN84] είναι μια από τις πρώτες επεκτάσεις του προσαρμοζόμενου K-D-δέντρου το οποίο έχει την δυνατότητα να χειριστεί ιδιαίτερα μεγάλα και εκτεταμένα αντικείμενα. Κάθε εσωτερικός κόμβος συμβολίζεται με τη διάσταση (x ή y) επί της οποίας γίνεται το χώρισμα του χώρου και με τη θέση της γραμμής που προκαλεί το χώρισμα. Τα φύλλα αντιστοιχούν σε ορθογώνιους υποχώρους και περιέχουν τις διευθύνσεις των σελίδων δεδομένων που τους αντιστοιχούν. Το δέντρο αυτό χρησιμοποιεί clipping για να χειριστεί τα αντικείμενα.

Για να εισάγουμε ένα αντικείμενο ξεκινάμε από τη ρίζα. Σε κάθε ενδιάμεσο κόμβο είτε προχωράμε προς κάποιο απόγονο του είτε σπάμε το αντικείμενο -εφόσον αυτό ανήκει σε περισσότερους από έναν υποχώρους- και ακολουθούμε και τα δύο μονοπάτια που προκύπτουν. Αν μια σελίδα δεδομένων δεν χωράει το αντικείμενο, τη σπάμε με μια νέα γραμμή η οποία επιλέγεται να είναι κάθετη στη διάσταση με το μεγαλύτερο μήκος του αντίστοιχου υποχώρου. Αυτό ενδέχεται να προκαλέσει και κάποια παλιά αντικείμενα του δέντρου να σπάσουν. Στο σχήμα 7.21 δίνουμε το δέντρο αυτό για το παράδειγμα μας.



Σχήμα 7.21α: Τα αντικείμενα για το εκτεταμένο K-D-δέντρο

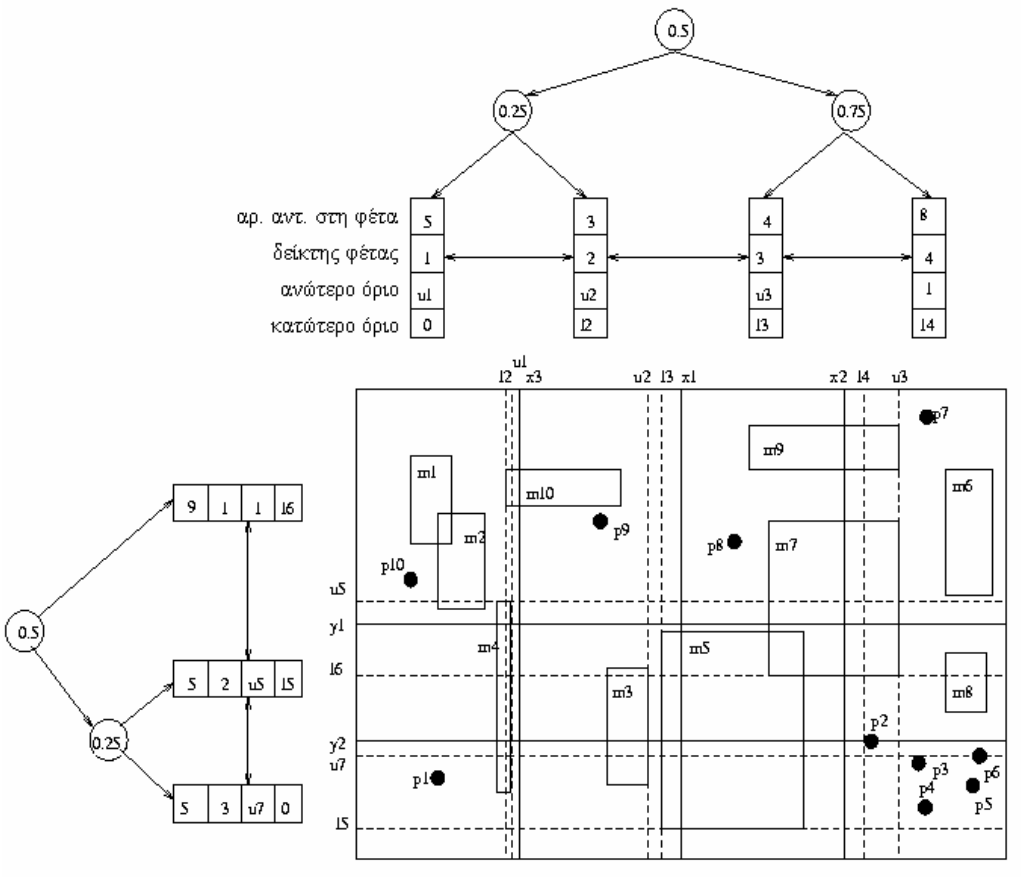


Σχήμα 7.21β: Το εκτεταμένο K-D-δέντρο

PLOP-κατακερματισμός

Ο *PLOP-κατακερματισμός* (**p**iecewise **l**inear **o**rder **p**reserving hashing) [SK88] είναι μια παραλλαγή του κατακερματισμού που χειρίζεται εκτεταμένα αντικείμενα χωρίς να εκτελεί πρώτα μετασχηματισμό τους σε σημεία. Ο PLOP-κατακερματισμός διαμερίζει το χώρο παρόμοια με το αρχείο πλέγματος (grid file) χρησιμοποιώντας γραμμές παράλληλες στους άξονες, τις οποίες και οργανώνει με χρήση δυαδικών δέντρων για κάθε διάσταση (άρα με δύο δέντρα στη περίπτωση διδιάστατου χώρου). Κάθε εσωτερικός κόμβος των δυαδικών δέντρων αντιστοιχεί σε μια γραμμή και κάθε φύλλο σε έναν διδιάστατο υπόχωρο. Τα αντικείμενα μπορεί και να ανήκουν σε περισσότερα από ένα κελιά.

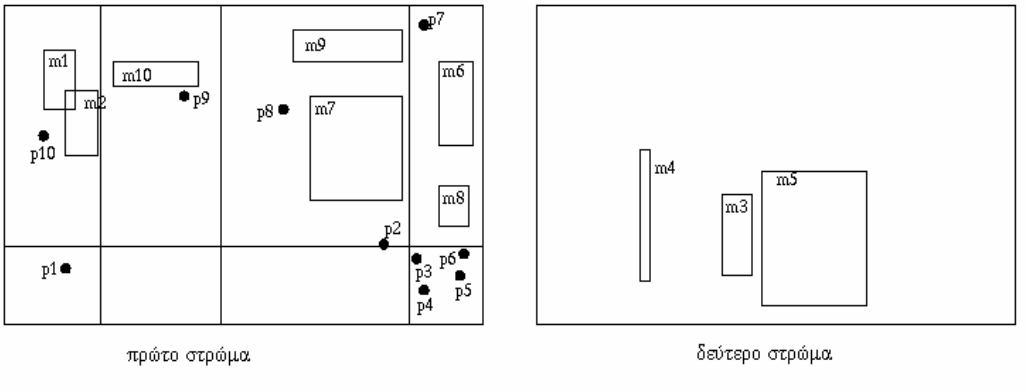
Τα φύλλα περιέχουν δείκτες στα αντίστοιχα αντικείμενα. Για να έχουμε γρήγορη προσπέλαση τα δέντρα τα αποθηκεύουμε στη κύρια μνήμη. Όπως φαίνεται και στο σχήμα 7.22, κάθε φέτα του χώρου (που προκύπτει από τον διαχωρισμό του χώρου λόγω μόνο του ενός από τα δύο δέντρα) έχει ένα κατώτερο και ένα ανώτερο όριο (διακεκομμένες γραμμές), τα οποία δείχνουν μέχρι που εκτείνονται επί της φέτας τα αντικείμενα που βρίσκονται μέσα της. Η εισαγωγή και διαγραφή αντικειμένων γίνεται άμεσα και παρόμοια με το αρχείο πλέγματος. Αναλυτικές μελέτες έχουν δείξει ότι η μέθοδος αυτή δίνει καλύτερα αποτελέσματα από το R-δέντρο και τις παραλλαγές του όταν έχουμε ομοιόμορφα κατανεμημένα δεδομένα.



Σχήμα 7.22: PLOP-κατακερματισμός

Το πολυστρωματικό αρχείο πλέγματος (multi-layer grid file)

Το πολυστρωματικό αρχείο πλέγματος [SW88] αποτελείται από μια σειρά από στρώματα πλεγμάτων. Κάθε στρώμα αντιστοιχεί και σε ένα αρχείο πλέγματος το οποίο καλύπτει όλο τα χώρο και τον χωρίζει ανεξάρτητα από τα άλλα αρχεία πλέγματος. Ένα αντικείμενο εισάγεται στο πρώτο αρχείο στο οποίο θα χωρέσει ολόκληρο (χωρίς να χρειαστεί clipping). Το μέγεθος των κάδων σε κάθε αρχείο μεγαλώνει καθώς προχωράμε από το αρχικό προς το τελικό αρχείο και άρα μεγάλα αντικείμενα είναι πιο πιθανό να τοποθετηθούν σε ένα από τα τελευταία αρχεία της σειράς. Αν κάποιο αντικείμενο δεν μπορεί να εισαχθεί ούτε και στο τελευταίο αρχείο χωρίς να υποστεί σπάσιμο, φτιάχνουμε ένα νέο αρχείο που να το χωράει ολόκληρο. Μια άλλη λύση είναι να επιτρέψουμε clipping μόνο στο τελευταίο αρχείο. Στο σχήμα 7.23 δίνουμε ένα πολυστρωματικό αρχείο που μπορεί να προκύψει για το παράδειγμα μας. Πιθανό μειονέκτημα της μεθόδου είναι η χαμηλή χρησιμοποίηση της μνήμης. Προκειμένου να αποφευχθεί το πρόβλημα αυτό, προτάθηκε μια εναλλακτική δομή η οποία καλείται R-αρχείο [HSW90], η απόδοση της οποίας είναι ανταγωνιστική με αυτήν του R-δέντρου.



Σχήμα 7.23: Πολυστρωματικό αρχείο πλέγματος

7.4 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Οι βάσεις χωρικών δεδομένων αποτελούν ένα πολύ ενδιαφέρον τομέα της έρευνας όπως ήδη αναφέραμε. Είναι σαφές από τα όσα αναπτύχθηκαν παραπάνω, ότι η έρευνα σε αυτόν τον τομέα είχε ως αποτέλεσμα μια πληθώρα πολυδιάστατων μεθόδων προσπέλασης. Μολαταύτα, δεν είναι δυνατόν να ξεχωρίσει κανείς μια από τις παραπάνω μεθόδους ως την καλύτερη, κάτω από οποιεσδήποτε συνθήκες. Ο βασικός λόγος για αυτό είναι η μεγάλη εξάρτηση της απόδοσης των μεθόδων αυτών από την κατανομή, το μέγεθος και το σχήμα των δεδομένων.

Κάποιες συγκριτικές μελέτες έχουν γίνει και έχουν καταλήξει σε περιορισμένου εύρους συμπεράσματα (ενδεικτικά αναφέρονται οι μελέτες [KSSS89, GB91]). Πιο συγκεκριμένα, όπως έχουμε ήδη πει, από την μία οι τεχνικές κατακερματισμού υπερτερούν όταν έχουμε ομοιόμορφη κατανομή ενώ από την άλλη, σε περίπτωση ακανόνιστων δεδομένων, οι δενδρικές δομές δίνουν καλύτερα αποτελέσματα. Ειδικά όσον αφορά το R-δέντρο και τις παραλλαγές του, το R*-δέντρο δείχνει να υπερτερεί έναντι όλων των άλλων [BKSS90].

Ο μόνος τρόπος που ίσως δώσει συγκεκριμένα αποτελέσματα κοινώς αποδεκτά από όλους, είναι ο καθορισμός κάποιων τυπικών διαδικασιών και δοκιμών (benchmarks) με τα οποία θα συγκριθούν όλες οι μέθοδοι, κάτω από πολλές διαφορετικές συνθήκες όσον αφορά την υφή των δεδομένων και θα μπορούν να βγουν έτσι κάποια γενικά συμπεράσματα.

7.5 ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

[AG97] N. R. Adam, A. Gangopadhyay, *Database Issues on Geographic Information Systems*, Kluwer Academic Press, 1997.

[Be75] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, vol. 18(9), pp. 509-517, 1975.

[Be79] J. L. Bentley, "Multidimensional binary search in database applications", *IEEE Transactions on Software Engineering*, vol. 4(5), pp. 333-340, 1979.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles", In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1990.

[Fre87] M. Freeston, "The BANG file: a new kind of grid file", In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1987.

[Fr95] M. Freeston, "A general solution of the n-dimensional B-tree problem", In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1995.

[GB91] O. Gunther, J. Bilmes, "Tree-based access methods for spatial databases:

Implementation and performance evaluation", *IEEE Transactions on Knowledge and Data Engineering*, vol. 3(3), pp. 342-356, 1991.

[GG98] V. Gaede, O. Gunther, "Multidimensional Access Methods", *ACM Computing Surveys*, in press, 1998.

[Gut84] A. Guttman, "R-trees: a dynamic index structure for spatial searching", In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1984.

[Gu89] O. Gunther, "The cell tree: An object-oriented index structure for geometric databases", In *Proceedings of the 5th IEEE Conference on Data Engineering*, 1989.

[Gu94] R. H. Guting, "An Introduction to Spatial Database Systems", *The VLDB Journal*, vol. 3(4), October 1994.

[Hi85] K. Hinrichs, "Implementation of the grid file: Design concepts and experiences", *BIT*, vol. 25, 1985.

[HSW89] A. Henrich, H.-W. Six, P. Widmayer, "The LSD tree: spatial access to multidimensional point and non point objects", In *Proceedings of the 15th International Conference on Very Large Data Bases* (VLDB), 1989.

[HSW90] A. Hutflesz, H.-W. Six, P. Widmayer, "The R-file: An efficient access structure for proximity queries", In *Proceedings of the 6th IEEE Conference on Data Engineering*, 1990.

[Ja90] H. V. Jagadish, "Spatial search with polyhedra", In *Proceedings of the 6th IEEE Conference on Data Engineering*, 1990.

[Knu73] D. Knuth, *The Art of Computer Programming, vol. 3: Sorting and Searching*, Addison-Wesley, 1973.

[KSSS89] H.-P. Kriegel, M. Schiwietz, R. Schneider, B. Seeger, "Performance comparison of point and spatial access methods", In *Proceedings of the 1st International Symposium on Large Spatial Databases* (SSD), 1989.

[Li80] W. Litwin, "Linear hashing: A new tool for file and table addressing", In *Proceedings of the 6th International Conference on Very Large Data Bases* (VLDB), 1980.

[LS90] D. Lomet, B. Salzberg, "The hB-tree: A multiattribute indexing method with good guaranteed performance", *ACM Transactions on Database Systems*, vol.15(4), pp. 625-658, 1990.

[MHN84] T. Matsuyama, L. V. Hao, M. Nagao, "A file organization for geographic information systems based on spatial proximity", *International Journal of Computer Vision, Graphics and Image Processing*, vol.26(3), pp. 303-318, 1984.

[NHS84] J. Nievergelt, H. Hinterberger, K.C. Sevcik, "The Grid File: An Adaptable, Symmetric Multikey file Structure", *ACM Transactions on Database Systems*, vol.9(1), pp. 38-71, March 1984.

[Ro81] J. T. Robinson, "The K-D-B-tree: A search structure for large multidimensional dynamic indexes", In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1981.

[SK88] B. Seeger, H.-P. Kriegel, "Techniques for design and implementation of spatial access methods", In *Proceedings of the 14th International Conference on Very Large Data Bases* (VLDB), 1988.

[SK90] B. Seeger, H.-P. Kriegel, "The buddy-tree: An efficient and robust access method for spatial data base systems", In *Proceedings of the 16th International Conference*

on Very Large Data Bases (VLDB), 1990.

[SRF87] T. Sellis, N. Roussopoulos, C. Faloutsos, "The R^+-tree: a dynamic index for multidimensional objects", In *Proceedings of the 13th International Conference on Very Large Data Bases* (VLDB), 1987.

[SW88] H.-W. Six, P. Widmayer, "Spatial searching in geometric databases", In *Proceedings of the 4th IEEE Conference on Data Engineering*, 1988.