ΤΕΙ ΣΕΡΡΩΝ
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

Θέμα

Μελέτη των αλγορίθμων ταξινόμησης Binary Search - Tree, Quad - Tree και R – Tree και εφαρμογή των Binary Search – Tree και Quad – Tree σε σχεσιακές βάσεις δεδομένων.

Πτυχιακή Εργασία

των

Καρατζά Μαργαρίτη - Συμιακάκη Μιχάλη

Επιβλέπων καθηγητής

Αλκιβιάδης Τσιμπίρης

Σέρρες, Ιούνιος 2006

Περίληψη

Η μελέτη των δομών δεδομένων και των αλγορίθμων είναι καθοριστικής σημασίας για την επιστήμη και τη τεχνολογία των υπολογιστών. Οι αλγόριθμοι Binary Search – Tree, Quad – Tree και R – Tree αποτελούν τρεις από τους καλύτερους αλγόριθμους ταξινόμησης με βάση την δενδρική δομή δεδομένων. Το Binary Search – Tree για δεδομένα μίας διάστασης, το Quad – Tree δύο διαστάσεων ενώ το R – Tree για δεδομένα δύο ή περισσότερων διαστάσεων. Οι αλγόριθμοι που αναφέραμε εφαρμόζονται συχνά σε γεωγραφικά συστήματα πληροφορίας (GIS), εφαρμογές σχεδίασης τύπου CAD, εφαρμογές εικόνας και ήχου (multimedia), την ρομποτική και αλλού για την ταχύτερη ταξινόμηση των δεδομένων και την βελτιστοποίηση της αναζήτησης.

Σκοπός της πτυχιακής εργασίας είναι καταρχήν η κατασκευή μιας βάσης δεδομένων στον SQL Server και η επικοινωνία της βάσης αυτής με το MatLab και τον C++ Builder. Επιπλέον, η κατασκευή μίας εφαρμογής σε προγραμματιστικό περιβάλλον (C++), η οποία θα παράγει – εισάγει δεδομένα στη βάση δεδομένων και θα τα ταξινομεί χρησιμοποιώντας τις μεθόδους Binary Search – Tree και Quad - Tree. Η εφαρμογή θα αντλεί τα δεδομένα από τη βάση δεδομένων και θα κατασκευάζει δένδρα ταξινόμησης βασισμένα στις σχετικές θέσεις των σημειακών δεδομένων, εφαρμόζοντας σε αυτά διάφορες τεχνικές αναζήτησης.

Abstract

The research of data structures and algorithms is significant for the science and the technology of computers. The algorithms Binary Search –Tree, Quad –Tree and R – Tree represent three of the best algorithms for classification with tree data structures. The Binary Search –Tree for one dimension data, the Quad –Tree for two dimensions data and the R –Tree for multiple dimensions data. The algorithms often used in geographic information systems (GIS), design applications like CAD, multimedia applications, robotics and elsewhere for the quickest classification of data and the optimisation of search.

The thesis aims to create a database in SQL Server and to establish a connection with Matlab and C++ Builder, also the design and programming of an application using the C++ language. The application has to generate and classify the data in order to create an index. The application will use the data of the database and create Binary Search – Tree and Quad - Tree data structures and apply in them various search techniques.

Ευχαριστίες...

Θα θέλαμε να ευχαριστήσουμε πρώτον απ' όλους τον επιβλέποντα καθηγητή μας Αλκιβιάδη Τσιμπίρη για τη συνεχή υποστήριξη που μας παρείχε και την εμπιστοσύνη που έδειξε σε εμάς καθ' όλη την διάρκεια της εργασίας αυτής. Οι γνώσεις του και η συνεχής παρουσία του δίπλα μας ήταν πολύτιμα στοιχεία για την επιτυχή ολοκλήρωση αυτής της πτυχιακής εργασίας. Ευχαριστούμε ακόμη όλους εκείνους τους ανώνυμους και μη χρήστες του διαδικτύου για την απλόχερη και αφιλοκερδή βοήθεια τους καθώς και τους φίλους και τα μέλη των οικογενειών μας που επέδειξαν μεγάλη υπομονή, κατανόηση και ηθική συμπαράσταση καθ' όλη τη διάρκεια της εκπόνησης αυτής της πτυχιακής εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο, στην παράγραφο 1.1 αναλύεται το αντικείμενο της παρούσας πτυχιακής εργασίας και στην παράγραφο 1.2 γίνεται μια επισκόπηση της οργάνωσης του τόμου που κρατάτε στα χέρια σας.

1.1 Αντικείμενο της Πτυχιακής

Η παρούσα Πτυχιακή εργασία εντάσσεται στο γενικότερο γνωστικό αντικείμενο των βάσεων δεδομένων (Database Theory) και συγκεκριμένα στον τομέα των αλγόριθμων ταξινόμησης (Data Classification Algorithms) με δένδρα. Η αποτελεσματική και αποδοτική διαχείριση μεγάλων όγκων δεδομένων είναι απαραίτητη σχεδόν σε όλες τις εφαρμογές υπολογιστών. Η ανάγκη ύπαρξης δομών δεδομένων που θα μειώνουν σημαντικά το κόστος αναζήτησης και θα προσφέρουν αποδοτικές λειτουργίες προσπέλασης, προσθήκης ή διαγραφής των δεδομένων είναι επιτακτική. Μια τέτοια ιεραρχική δομή δεδομένων είναι το δέντρο, το οποίο βασίζεται στις αρχές τις περιοδικά επαναλαμβανόμενης ανάλυσης – διάσπασης (recursive decomposition) οι οποίες είναι παρόμοιες με τις μεθόδους διαίρει κα βασίλευε (*divide and conquer* methods) [Aho et al. 1974]).

Στην πτυχιακή αυτή εργασία θα μελετήσουμε ορισμένους από τους σημαντικότερους αλγορίθμους που χρησιμοποιούνται σήμερα στους υπολογιστές. Οι αλγόριθμοι αυτοί είναι το Binary Search - Tree, Quad - Tree και το R - Tree. Στόχος μας είναι, να δώσουμε στους αναγνώστες την δυνατότητα να κατανοήσουν τις βασικές ιδιότητες των αλγορίθμων που μελετήσαμε. Ιδιαίτερα ενδιαφέρον θα βρουν το υλικό της πτυχιακής, όσοι έχουν ήδη επαφή με το αντικείμενο των αλγορίθμων ταξινόμησης με δενδρικές δομές δεδομένων.

Στα πλαίσια της πτυχιακής εργασίας θα υλοποιήσουμε τους αλγόριθμους Binary Search - Tree και Quad - Tree και θα εξάγουμε συμπεράσματα τόσο από την υλοποίησή τους όσο και από την εφαρμογή σε αυτούς διάφορων τεχνικών αναζήτησης. Οι αλγόριθμοι που αναφέραμε και οι παραλλαγές τους έχουν πολλές εφαρμογές. Ειδικότερα, οι πιο συνηθισμένοι τομείς στους οποίους βρίσκουν εφαρμογή είναι: εφαρμογές εικόνας και ήχου (multimedia), εφαρμογές σχεδίασης τύπου CAD, γεωγραφικά συστήματα πληροφορίας (GIS), στην σχεδίαση ολοκληρωμένων κυκλωμάτων (VLSI), εφαρμογές internet όπως Web servers, στον έλεγχο της μεθόδου παραγωγής (real-time process control), σε επιστημονικούς υπολογισμούς και αλλού.

1.2 Οργάνωση της Πτυχιακής

Στη συγγραφή του κειμένου προσπαθήσαμε να λάβουμε υπόψη και το νεοεισερχόμενο στο πεδίο αναγνώστη. Στην αρχή κάθε κεφαλαίου υπάρχει μια απλή περιγραφή των θεμάτων που θα αναλυθούν.

Η πτυχιακή αποτελείται από 7 κεφάλαια :

Στο 1ο κεφάλαιο αναφέρεται με λίγα λόγια το αντικείμενο της πτυχιακής εργασίας και η οργάνωση του τόμου.

Στο 2ο κεφάλαιο της πτυχιακής θα ορίσουμε και θα αναπτύξουμε ορισμένες βασικές έννοιες της πληροφορικής, απαραίτητες για την κατανόηση των επόμενων κεφαλαίων.

Στο 3ο κεφάλαιο αναπτύσσουμε τους αλγόριθμους ταξινόμησης Binary Search - Tree, Quad - Tree και R - Tree.

Στο 4ο κεφάλαιο περιγράφουμε τις λεπτομερείς υλοποίησης της εφαρμογής που δημιουργήσαμε και παρουσιάζουμε τον έλεγχο της.

Στο 5ο κεφάλαιο εξηγούμε το πως λειτουργεί η εφαρμογή (Εγχειρίδιο χρήσης - manual).

Στο 6ο κεφάλαιο συνοψίζουμε τα αποτελέσματα της πτυχιακής εργασίας και παραθέτουμε μελλοντικές επεκτάσεις αυτής.

Στο 7ο κεφάλαιο δίνεται η βιβλιογραφία και γενικότερα οι πηγές, οι οποίες προμήθευσαν τις απαραίτητες πληροφορίες για τη συγγραφή της πτυχιακής.

ΚΕΦΑΛΑΙΟ 2

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Στο παρόν κεφάλαιο, γίνεται μια εισαγωγή στις έννοιες των δεδομένων, των δομών δεδομένων, των δένδρων, των αλγόριθμων, των βάσεων δεδομένων και του προγραμματισμού. Κύριος στόχος μας είναι να προλειάνουμε το έδαφος για την ανάπτυξη των επόμενων κεφαλαίων της πτυχιακής εργασίας.

2.1 Δεδομένα

Η οργάνωση των δεδομένων για επεξεργασία αποτελεί ένα θεμελιώδες βήμα για την ανάπτυξη ενός προγράμματος υπολογιστή. Με τον όρο δεδομένα αναφερόμαστε συνήθως σε μη κατάλληλα επεξεργασμένα και μη ταξινομημένα σύνολα πληροφοριών. Ως σύνολο πληροφορίας εννοούμε ειδήσεις, γεγονότα και έννοιες που θεωρούμε ως αποκτηθείσα γνώση. Ένας αυστηρός ορισμός για το τι είναι δεδομένα και τι είναι πληροφορία, είναι ο ακόλουθος:

Δεδομένα (data): Δεδομένα είναι μια παράσταση, όπως γράμματα, αριθμοί, σύμβολα κ.ά. στα οποία μπορούμε να δώσουμε κάποια σημασία (έννοια).

Πληροφορία (information): Πληροφορία είναι η σημασία (έννοια) που δίνουμε σ' ένα σύνολο από δεδομένα, τα οποία μπορούμε να επεξεργαστούμε βάσει προκαθορισμένων κανόνων και να βγάλουμε έτσι κάποια χρήσιμα συμπεράσματα.

Τα δεδομένα μπορούν να θεωρηθούν ως τρόποι αναπαράστασης εννοιών και γεγονότων που μπορούν να υποστούν διαχείριση και επεξεργασία. Η συλλογή και αποθήκευση ενός τεράστιου όγκου δεδομένων όπως απαιτούν οι κοινωνικές συνθήκες σήμερα, δημιουργεί την ανάγκη της σωστής οργάνωσης και ταξινόμησης των δεδομένων. Τα δεδομένα θα πρέπει να οργανωθούν με τέτοιο τρόπο έτσι ώστε να μπορούμε να τα εντοπίζουμε και να τα αξιοποιούμε εύκολα και γρήγορα τη στιγμή που τα χρειαζόμαστε.

Ένα κλασικό παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν για παράδειγμα ο τηλεφωνικός κατάλογος της πόλης των Σερρών, όπου οι συνδρομητές δεν θα ήταν καταχωρημένοι αλφαβητικά σύμφωνα με το επώνυμο και το όνομά τους, αλλά εντελώς τυχαία. Ένας τέτοιος τηλεφωνικός κατάλογος θα περιείχε μια τεράστια ποσότητα δεδομένων αλλά θα ήταν ουσιαστικά άχρηστος.

Για να είναι χρήσιμα τα δεδομένα θα πρέπει να έχουν ορισμένα χαρακτηριστικά τα οποία καθορίζουν την ποιότητα τους. Συγκεκριμένα τα δεδομένα πρέπει να είναι:

- Ακριβή – δηλαδή να μην περιέχουν σφάλματα. Για να συμβεί αυτό πρέπει η διαδικασία συλλογής των δεδομένων να ελέγχει, στο μέτρο του δυνατού, την ακρίβεια τω δεδομένων που συλλέγονται και αποθηκεύονται.

- Πλήρη – δηλαδή όλα τα δεδομένα που απαιτούνται για την λύση ενός προβλήματος ή για την λήψη μιας απόφασης πρέπει να υπάρχουν και είναι διαθέσιμα σε αυτούς που τα χρειάζονται.
- Σχετικά – δηλαδή να είναι απαραίτητα – χρήσιμα για την λύση ενός συγκεκριμένου προβλήματος που αντιμετωπίζει κάποιος.
- Έγκαιρα – δηλαδή να είναι διαθέσιμα την χρονική στιγμή που τα χρειάζεται κάποιος που έχει να λύσει ένα πρόβλημα.

2.1.2 Το προβλήματα οργάνωσης των δεδομένων

Στα αρχικά στάδια της οργάνωσης δεδομένων, μια συνηθισμένη πρακτική ήταν η δημιουργία ξεχωριστών εφαρμογών (προγραμμάτων), όπως για παράδειγμα η δημιουργία ενός αρχείου πελατών και ενός άλλου ανεξάρτητου αρχείου για τις παραγγελίες των πελατών. Τα προβλήματα που προέκυψαν από την πρακτική αυτή είναι τα εξής :

- Πλεονασμός των δεδομένων (data redundancy). Υπάρχει η περίπτωση να έχουμε επανάληψη των ίδιων δεδομένων σε αρχεία διαφορετικών εφαρμογών. Για παράδειγμα, αν έχουμε ένα αρχείο πελατών και ένα αρχείο παραγγελιών αυτών των πελατών, είναι σχεδόν σίγουρο ότι θα υπάρχουν κάποια στοιχεία των πελατών που θα υπάρχουν και στα δύο αρχεία.
- Ασυνέπεια των δεδομένων (data inconsistency). Αυτό μπορεί να συμβεί όταν υπάρχουν τα ίδια στοιχεία των πελατών (πλεονασμός) και στο αρχείο πελατών και στο αρχείο παραγγελιών και χρειασθεί να γίνει κάποια αλλαγή στη διεύθυνση ή στα τηλέφωνα κάποιου πελάτη, οπότε είναι πολύ πιθανό να γίνει η διόρθωση μόνο στο ένα αρχείο και όχι και στο άλλο.
- Αδυναμία μερισμού δεδομένων (data sharing). Με τον όρο μερισμό δεδομένων εννοούμε τη δυνατότητα για κοινή χρήση των στοιχείων κάποιων αρχείων. Για παράδειγμα, ο μερισμός δεδομένων θα ήταν χρήσιμος αν με την παραγγελία ενός πελάτη μπορούμε να έχουμε πρόσβαση την ίδια στιγμή στο αρχείο πελατών για να δούμε το υπόλοιπο του πελάτη και μετά στο αρχείο της αποθήκης για να δούμε αν είναι διαθέσιμα τα προϊόντα που παρήγγειλε ο συγκεκριμένος πελάτης. Η

αδυναμία μερισμού δεδομένων δημιουργεί καθυστέρηση στη λήψη αποφάσεων και στην εξυπηρέτηση των χρηστών.

- Αδυναμία προτυποποίησης. Έχει να κάνει με την ανομοιομορφία και με την διαφορετική αναπαράσταση και οργάνωση των δεδομένων στα αρχεία των εφαρμογών. Η αδυναμία αυτή δημιουργεί προβλήματα προσαρμογής των χρηστών καθώς και προβλήματα στην ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων.

2.1.2 Δομές Δεδομένων

Τα δεδομένα που χρησιμοποιεί ένα πρόγραμμα οργανώνονται με συγκεκριμένους τρόπους ώστε να διευκολύνεται η χρήση τους και η αποδοτικότητα των αλγορίθμων που τα χρησιμοποιούν. Οι τρόποι αυτοί ονομάζονται δομές δεδομένων (*data structures*).

Οι βασικές δομές δεδομένων είναι οι εξής:

- Στίβα (*stack*)
- Ουρά (*queue*)
- Ουρά προτεραιότητας (*priority queue*)
- Λίστα (*list*)
- Δέντρο (*tree*)
- Σύνολο (*set*)
- Σάκος (*bag, multiset*)
- Απεικόνιση (*map*)
- Γράφος (*graph*)

Οι δομές δεδομένων (data structures) είναι τύποι αντικειμένων που μπορούν να χρησιμοποιηθούν σε αλγόριθμους για την επίλυση προβλημάτων. Συγκεκριμένα, στις δομές δεδομένων τοποθετούνται τα δεδομένα του προβλήματος. Επίσης, για κάθε δομή δεδομένων ορίζεται ένα σύνολο πράξεων που μπορούν να εφαρμοστούν. Έτσι, η επίλυση του προβλήματος επιτυγχάνεται με την τοποθέτηση των δεδομένων στην κατάλληλη δομή δεδομένων και την εφαρμογή των αντίστοιχων πράξεων σε αυτή.

Οι δομές δεδομένων χρησιμοποιούνται επειδή επιτρέπουν την προσπέλαση και επεξεργασία δεδομένων με εύκολο τρόπο. Κάθε δομή δεδομένων και οι πράξεις της παρουσιάζουν ιδιαίτερα χαρακτηριστικά. Η καταλληλότητα μιας δομής δεδομένων για την επίλυση ενός προβλήματος εξαρτάται από τη φύση του προβλήματος και τον χρησιμοποιούμενο αλγόριθμο επίλυσης.

Οι βασικές πράξεις που εφαρμόζονται στις δομές δεδομένων είναι οι ακόλουθες:

- Εισαγωγή
- Διαγραφή
- Αναζήτηση
- Ταξινόμηση
- Ενημέρωση (συνήθως περιλαμβάνει διαδοχικά αναζήτηση-διαγραφή-εισαγωγή)

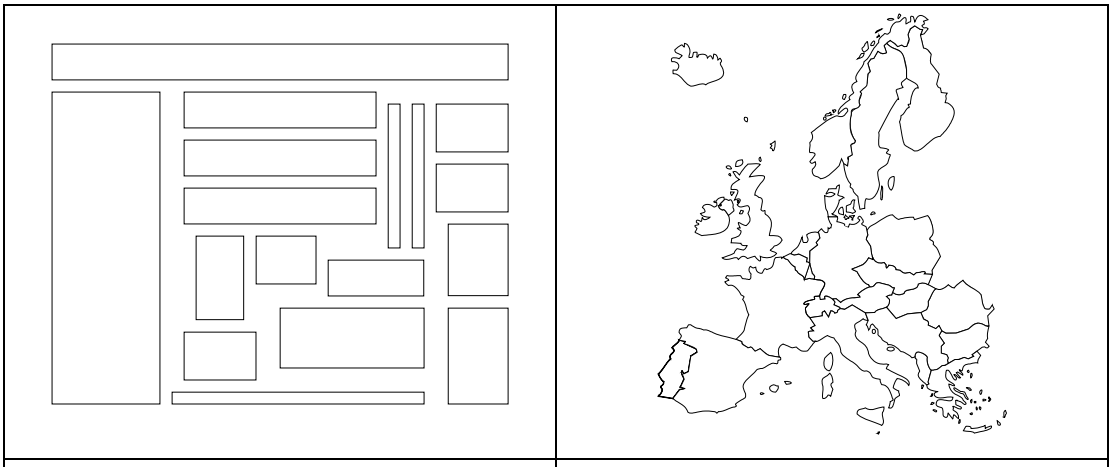
Ένας κύριος διαχωρισμός των δομών δεδομένων είναι σε γραμμικές και μη γραμμικές:

- **Γραμμικές:** Γραμμικές είναι οι δομές εκείνες για τις οποίες μπορεί να οριστεί διάταξη για δυο στοιχεία τους, δηλαδή κάποιο στοιχείο είναι πρώτο και κάποιο τελευταίο, ενώ οποιοδήποτε από τα υπόλοιπα θα έχει ένα προηγούμενο, από το οποίο έπεται και ένα επόμενο, από το οποίο προηγείται. Χαρακτηριστικά είδη γραμμικών δομών είναι οι πίνακες, *οι λίστες*, *οι στοίβες* και *οι ουρές*

- **Μη γραμμικές:** Στις μη γραμμικές δομές, οι σχέσεις των δεδομένων είναι πιο περίπλοκες και όχι μονοδιάστατες. Κάθε στοιχείο μιας τέτοιας δομής μπορεί να έχει πολλά επόμενα στοιχεία. Χαρακτηριστικό είδος μη γραμμικής δομής είναι τα *δέντρα*.

2.1.3 Χωρικά δεδομένα (Spatial Data)

Ονομάζουμε τα πολυδιάστατα δεδομένα με το γενικό όρο *χωρικά δεδομένα* και τις αντίστοιχες βάσεις δεδομένων, που αποθηκεύουν χωρικά δεδομένα, με τον όρο *χωρικές βάσεις δεδομένων* (ΧΒΔ).



(α) VLSI δεδομένα	(β) Χαρτογραφικά δεδομένα

Εικόνα 1: Παραδείγματα χωρικών βάσεων δεδομένων

Στα χωρικά δεδομένα ένα στοιχείο μπορεί να είναι σημείο, γραμμή, πολύγωνο, ή να απαρτίζεται από πλήθος γραμμών ή πολύγωνων. **Γεωμετρία** ονομάζεται ένα σύνολο από ένα ή περισσότερα στοιχεία. Τα στοιχεία της γεωμετρίας δεν είναι απαραίτητο να είναι του ίδιου τύπου, για παράδειγμα μπορείς να έχεις μια γεωμετρία που αποτελείται από σημεία και πολύγωνα.

Οι ερωτήσεις στις χωρικές βάσεις δεδομένων συνήθως έχουν να κάνουν με τα χωρικά χαρακτηριστικά των δεδομένων π.χ. τη θέση τους στο χώρο. Τυπικές χωρικές ερωτήσεις είναι οι ακόλουθες:

- *ερώτηση σημείου* (point query): δοθέντος ενός σημείου *p*, βρες όλα τα αντικείμενα που περιέχουν το *p*.
- *ερώτηση περιοχής* (range query): δοθέντος μιας ακτίνας *r*, βρες όλα τα αντικείμενα με όλα τα αντικείμενα, που περικλείονται από την ακτίνα r.
- *ερώτηση κατεύθυνσης* (direction query): δοθέντος ενός αντικειμένου *o* και μιας κατευθυντήριας σχέσης *R* (π.χ. βόρεια, αριστερά), βρες όλα τα αντικείμενα που βρίσκονται στην κατεύθυνση *R* σχετικά με το *o*.

- *ερώτηση κοντινότερου γείτονα* (Nearest Neighbor Query): δοθέντος ενός αντικειμένου *o*, βρες όλα τα αντικείμενα που απέχουν ελάχιστη απόσταση από το *o*.

Για την αποδοτική υποστήριξη των χωρικών δεδομένων και των χωρικών σχέσεων έχουν προταθεί την τελευταία δεκαετία διάφορες επεκτάσεις στα μοντέλα δεδομένων, τις γλώσσες ερωταποκρίσεων και τις δομές δεδομένων.

2.2 Δένδρα

Τα δένδρα (trees) είναι δομές που εκφράζουν κόμβους δεδομένων με ιεραρχική διάταξη. Είναι δομές δεδομένων στις οποίες κάθε στοιχείο, εκτός από ένα στοιχείο, τη ρίζα, έχει έναν ακριβώς πρόδρομο. Επομένως ένα χαρακτηριστικό γνώρισμα των δένδρων είναι ότι υπάρχει μία και μόνο διαδρομή από τη ρίζα προς ένα οποιαδήποτε στοιχείο.

Τα δένδρα ονομάζονται και **ιεραρχικές δομές δεδομένων**, επειδή καθορίζουν σχέσεις υπαγωγής των κατώτερων στα ανώτερα στοιχεία. Τα δένδρα είναι μια μαθηματική αφαίρεση η οποία παίζει κεντρικό ρόλο στη σχεδίαση και την ανάλυση των αλγορίθμων επειδή:

- Χρησιμοποιούμε δένδρα για να περιγράψουμε τις δυναμικές ιδιότητες των αλγορίθμων.
- Κατασκευάζουμε και χρησιμοποιούμε ρητές δομές δεδομένων που αποτελούν συμπαγείς υλοποιήσεις δένδρων.

Τα δένδρα απαντώνται συχνά τόσο στην πληροφορική όσο και στην καθημερινή ζωή. Δομή δένδρου έχει ο πίνακας περιεχομένων ενός βιβλίου , αλλά και το οργανόγραμμα μιας εταιρίας ή δημόσιας υπηρεσίας, όταν κάθε υπάλληλος έχει ακριβώς ένα προϊστάμενο. Αλλά και όταν αναλύουμε με το νου μας ένα αντικείμενο στα συστατικά του μέρη, παράγουμε νοητικά ένα δένδρο.

2.2.1 Ορισμοί

Δένδρρο (Tree): Ένα δένδρο είναι ένα σύνολο από κόμβους (*nodes*) που ενώνονται με ακμές (*edges*) και εκπληρεί την ακόλουθη ιδιότητα: Αν ορίσουμε ως μονοπάτι (*path*) ένα σύνολο κόμβων ενωμένων με ακμές και ένα συγκεκριμένο κόμβο ως ρίζα (*root*) του δέδρου τότε κάθε άλλος κόμβος ενώνεται με τη ρίζα με ένα και μόνο ένα μονοπάτι.

Στο πρώτο επίπεδο υπάρχει μόνο ένας κόμβος, η **ρίζα**. Κάθε κόμβος μπορεί να συνδέεται προς έναν ή περισσότερους κόμβους, τους κόμβους παιδιά του. Κάθε κόμβος έχει μόνο έναν κόμβο πατέρα, εκτός από τη ρίζα, η οποία είναι ο μόνος κόμβος χωρίς κόμβο πατέρα. Οι κόμβοι που δεν έχουν κόμβους παιδιά ονομάζονται ονομάζονται τερματικοί κόμβοι (*terminal nodes*) ή εξωτερικοί κόμβοι (*external nodes*) ή **φύλλα (*leaves*)**. Κόμβοι με παιδιά ονομάζονται μη τερματικοί κόμβοι (*non-terminal nodes*) ή εσωτερικοί κόμβοι (*internal nodes*) ή **κλαδιά (*branches*)**.

Επίπεδο (*Level*): Επίπεδο ενός κόμβου ορίζεται ως ο αριθμός των κόμβων στο μονοπάτι από τον κόμβο μέχρι τη ρίζα.

Μήκος μονοπατιού ενός κόμβου είναι ο αριθμός των κόμβων που πρέπει να περάσουμε για να φθάσουμε από τη ρίζα του δένδρου στο κόμβο .

Εσωτερικό μήκος μονοπατιού ενός δένδρου είναι το άθροισμα του μήκους των μονοπατιών όλων των κόμβων του δένδρου.

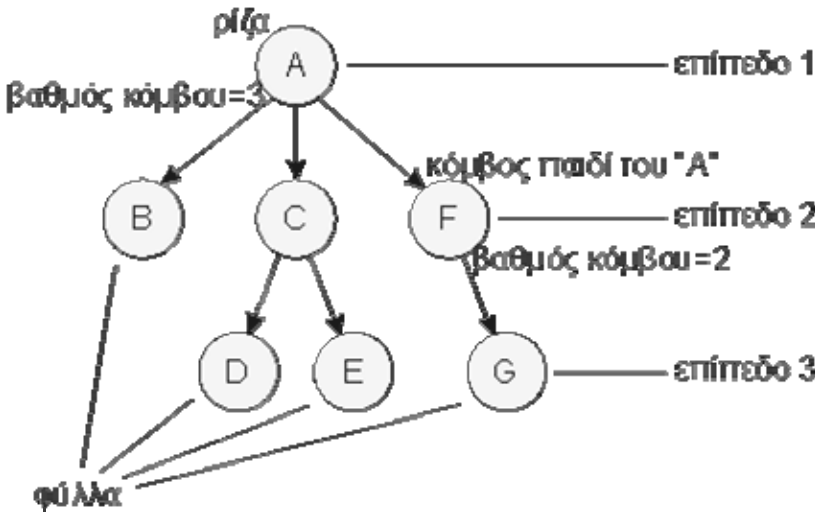
Μέσο μήκος μονοπατιού είναι το πηλίκο του εσωτερικού μήκους μονοπατιού δια το πλήθος των κόμβων.

Βάθος (*depth*) ή **Υψος** (*height*): Το μεγαλύτερο επίπεδο στο οποίο μπορεί να βρίσκεται ένας κόμβος του δέντρου είναι το **βάθος** ή **ύψος** του δέντρου.

Βαθμός (*Degree*) **ενός κόμβου** του δέντρου ονομάζεται το πλήθος των κόμβων παιδιών του.

Βαθμός (*Degree*) **ενός δένδρου:** Ο μέγιστος από τους βαθμούς όλων των κόμβων του δένδρου ονομάζεται βαθμός του δένδρου.

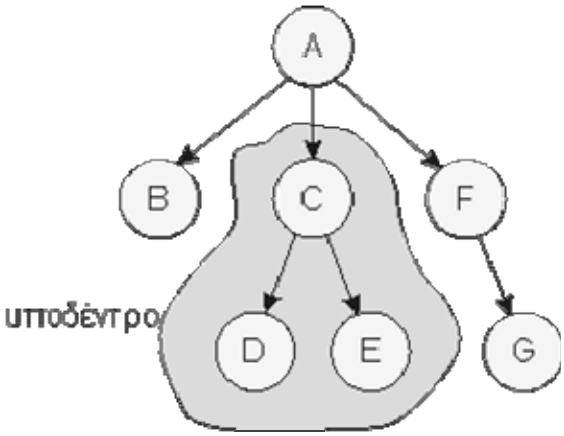
Στην **εικόνα 2** δίνεται ένα παράδειγμα δένδρου. Επιπλέον, μπορείτε να ξεχωρίσετε τα στοιχεία που το αποτελούν και που ορίσαμε παραπάνω.



Εικόνα 2: Παράδειγμα δένδρου

Στο παραπάνω παράδειγμα το "A" είναι η ρίζα του δένδρου. Έχει τρεις κόμβους παιδιά, το "B", το "C" και το "F", οπότε ο βαθμός της ρίζας είναι 3. Ο κόμβος "C" έχει δύο κόμβους παιδιά (βαθμός κόμβου 2), το "D" και το "E". Ο κόμβος "F" έχει ένα κόμβο παιδί, το "G". Οι κόμβοι "B", "D", "E" και "G" είναι κόμβοι φύλλα (δεν έχουν κόμβους παιδιά). Ο βαθμός του δένδρου είναι 3 και το βάθος επίσης 3.

Υποδένδρο (*Subtree*): Κάθε κομμάτι του δένδρου από έναν κόμβο και κάτω ονομάζεται υποδένδρο του δένδρου. Στην **εικόνα 3** ορίζεται ένα υποδένδρο στο δένδρο του παραπάνω παραδείγματος, της εικόνας 2.

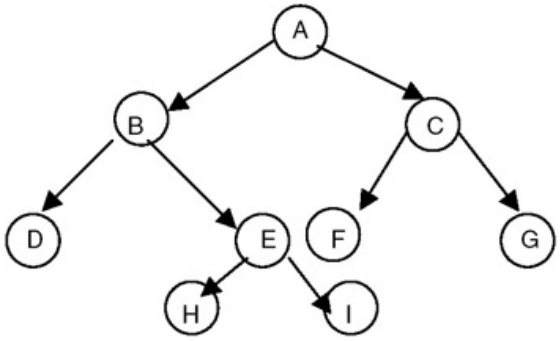


Εικόνα 3: Παράδειγμα υποδένδρου

Δάσος (Forest): Δάσος ονομάζεται ένα σύνολο από δένδρα και σχηματίζεται αν κόψουμε ένα δένδρο σε έναν κόμβο με παιδιά.

2.2.2 Το Δυαδικό δένδρο - Binary Tree

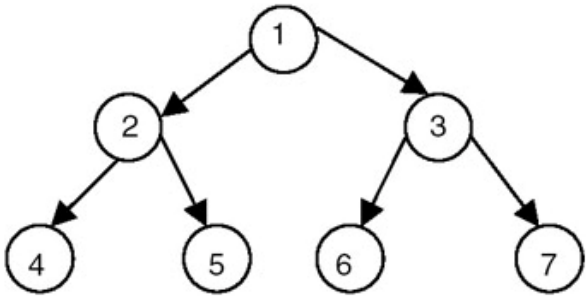
Ένα δένδρο ονομάζεται δυαδικό αν όλοι οι κόμβοι του έχουν βαθμό ≤ 2.
Ως δυαδικό δένδρο ορίζεται ένα δένδρο το οποίο είτε είναι κενό, είτε αποτελείται από μια ρίζα και δύο δυαδικά υπόδενδρα, το καθένα διακριτό από το άλλο και από τη ρίζα. Αναφερόμαστε στα δύο υπόδενδρα ως το αριστερό και το δεξιό υπόδενδρο.



Εικόνα 4: Παράδειγμα δυαδικού δένδρου

Το ύψος ενός δυαδικού δένδρου με n κόμβους μπορεί να είναι το πολύ $n - 1$ και το λιγότερο $\log_2(n)$

Γεμάτο (full): Ένα δυαδικό δένδρο είναι γεμάτο, αν κάθε εσωτερικός του κόμβος έχει δύο παιδιά. Δηλαδή αν έχει βάθος k θα πρέπει να έχει $2^k - 1$ κόμβους. Για παράδειγμα, για k = 3, ο αριθμός των κόμβων θα είναι = $2^k - 1 = 2^3 - 1 = 8 - 1 = 7$. Ένα γεμάτο δυαδικό δένδρο βάθους k = 3 φαίνεται στην **εικόνα 5**.



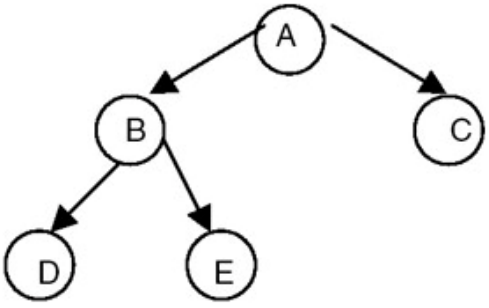
Εικόνα 5: Γεμάτο δυαδικό δένδρο

Χρησιμοποιούμε του αριθμούς από το κ ως το $2^k - 1$ ως ετικέτες για τους κόμβους του δένδρου.

Τέλειο (perfect): Ένα δυαδικό δένδρο είναι **τέλειο** (perfect), αν είναι γεμάτο και όλα τα φύλλα έχουν το ίδιο βάθος.

Πλήρες (complete): Ένα δυαδικό δένδρο είναι **πλήρες** (complete) αν
1. έχει βάθος 0 και ένα κόμβο,
2. έχει βάθος 1 και η ρίζα του έχει είτε δύο παιδιά είτε ένα αριστερό παιδί.
3. έχει βάθος κ και η ρίζα του έχει ένα τέλειο αριστερό υπόδενδρο ύψους κ-1
και ένα πλήρες δεξιό υπόδενδρο ύψους κ-1
4. έχει βάθος κ και η ρίζα του έχει ένα πλήρες αριστερό υπόδενδρο ύψους κ-1 και ένα τέλειο δεξιό υπόδενδρο ύψους κ-2.

Ένα **πλήρες** (complete) δυαδικό δένδρο βάθους k = 3 φαίνεται στην **εικόνα 6**.

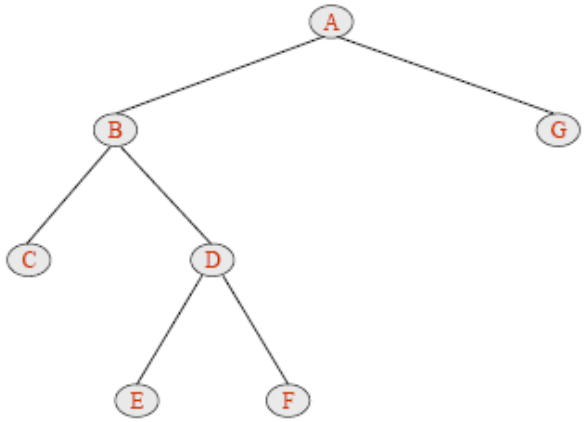


Εικόνα 6: Πλήρες δυαδικό δένδρο

Ισοζυγισμένο ή Ισορροπημένο (Balanced): Ισοζυγισμένο λέγεται το δυαδικά δένδρο που το δεξί και το αριστερά υποδένδρο έχουν διαφορά 1 το πολύ κόμβο. Συνήθως επιδιώκεται η δημιουργία ισοζυγισμένων δένδρων γιατί μειώνεται κατά πολύ ο χρόνος αναζήτησης ενός συγκεκριμένου κόμβου (μειώνεται το εσωτερικά μήκος μονοπατιού και το ύψος). Αποδεικνύεται ότι οι απαιτούμενες συγκρίσεις σε ένα δυαδικά δένδρο που δημιουργείται με κόμβους, οι οποίοι φθάνουν με τυχαία σειρά, είναι περίπου 39% περισσότερες από αυτές που απαιτούνται σε ένα ισοζυγισμένο δένδρο.

2.2.3 Διάσχιση δυαδικών δένδρων

Υπάρχουν τέσσερα είδη διάσχισης δυαδικών δέντρων. Για την παρουσίαση τους θα χρησιμοποιήσουμε το παράδειγμα της ακόλουθης εικόνας.



Εικόνα 7: Παράδειγμα δένδρου που θα χρησιμοποιηθεί στην εξήγηση των μεθόδων διάσχισης

Προδιατεταγμένη διάσχιση (preorder traversal):
Κατά την προδιατεταγμένη διάσχιση (preorder traversal) πραγματοποιούνται τα ακόλουθα:

1. Επίσκεψη της ρίζας
2. Επίσκεψη αριστερού υποδένδρου
3. Επίσκεψη δεξιού υποδένδρου

Το αποτέλεσμα της λειτουργίας της προδιατεταγμένης διάσχισης στο παράδειγμα του παραπάνω σχήματος, είναι: A, B, C, D, E, F, G.

Μεταδιατεταγμένη διάσχιση (Postorder traversal):

Κατά την μεταδιατεταγμένη διάσχιση πραγματοποιούνται τα ακόλουθα:

1. Επίσκεψη αριστερού υποδένδρου
2. Επίσκεψη δεξιού υποδένδρου
3. Επίσκεψη της ρίζας

Το αποτέλεσμα της λειτουργίας της μεταδιατεταγμένης διάσχισης στο παράδειγμα του παραπάνω σχήματος, είναι: C, E, F, D, B, G, A.

Ενδοδιατεταγμένη διάσχιση (Inorder traversal)

Κατά την ενδοδιατεταγμένη διάσχιση πραγματοποιούνται τα ακόλουθα:

1. Επίσκεψη αριστερού υποδένδρου
2. Επίσκεψη της ρίζας
3. Επίσκεψη δεξιού υποδένδρου

Τα αποτέλεσμα της λειτουργίας της ενδοδιατεταγμένη διάσχισης, στο παράδειγμα του παραπάνω σχήματος, είναι: C, B, D, E, F, A, G.

Κατά σειρά επιπέδων

Κατά την σειρά επιπέδων διάσχιση επισκεπτόμαστε τα στοιχεία κατά επίπεδα από πάνω προς τα κάτω. Σε κάθε επίπεδο επισκεπτόμαστε τα στοιχεία από τα αριστερά προς τα δεξιά.

Τα αποτέλεσμα της λειτουργίας της κατά σειρά επιπέδων διάσχισης, στο παράδειγμα του παραπάνω σχήματος, είναι A, B, G, C, D, E, F.

2.2.3 Δυαδική Αναζήτηση

Οι όροι δυαδική αναζήτηση και δυαδικό δένδρο αναζήτησης (Binary Search - Tree) συχνά συγχέονται. Λόγω του ότι στο επόμενο κεφάλαιο θα μελετήσουμε τα δυαδικά δένδρα αναζήτησης στο σημείο αυτό θα εξηγήσουμε τον όρο δυαδική αναζήτηση.

Η δυαδική αναζήτηση βασίζεται στην ιδέα ότι, αν οι αριθμοί του πίνακα είναι *ταξινομημένοι*, μπορούμε να παραλείπουμε από τη εξέταση τους μισούς από αυτούς, συγκρίνοντας κάθε φορά τον αριθμό που αναζητάμε με εκείνον τον αριθμό που βρίσκεται στη μεσαία θέση του πίνακα. Αν είναι ίσος, έχουμε μια επιτυχή αναζήτηση. Αν είναι μικρότερος, εφαρμόζουμε την ίδια μέθοδο στο αριστερό μισό του πίνακα. Αν είναι μεγαλύτερος, εφαρμόζουμε την ίδια μέθοδο στο δεξιό μισό του πίνακα. Ακολουθεί ένα παράδειγμα της λειτουργίας αυτής της μεθόδου σε ένα δεδομένο σύνολο αριθμών. Για να εξετάσουμε κατά πόσο περιλαμβάνεται ο αριθμός 5025 στον παρακάτω πίνακα αριθμών, αρχικά τον συγκρίνουμε με τον αριθμό 6504.Αυτό μας οδηγεί στην εξέταση του πρώτου μισού του πίνακα. Στη συνέχεια, τον συγκρίνουμε με τον αριθμό 4548 (τον μεσαίο του πρώτου μισού του πίνακα), κάτι που μας οδηγεί στο δεύτερο μισό του πρώτου μισού. Συνεχίζουμε με αυτόν τον τρόπο, εργαζόμενοι πάντοτε σε έναν υποπίνακα ο οποίος θα πρέπει να περιέχει τον αριθμό που αναζητάμε, εφόσον βέβαια ο αριθμός αυτός υπάρχει στον πίνακα. Τελικά, καταλήγουμε σε έναν πίνακα με ένα μόνο στοιχείο, το οποίο δεν είναι ίσο με τον αριθμό 5025, οπότε αντιλαμβανόμαστε ότι ο αριθμός 5025 δεν περιλαμβάνεται στον πίνακα.

Ο ταξινομημένος πίνακας για το παράδειγμα της δυαδικής αναζήτησης:

1488 1488
1578 1578
1973 1973
3665 3665
4426 4426
4548 4548
5435 5435 5435 5435 5435
5446 5446 5446 5446
6333 6333 6333
6385 6385 6385
6455 6455 6455
6504
6937
6965
7104
7230
8340
8950
9210
9363
9550
6945
6989

2.3 Αλγόριθμοι

Η λέξη αλγόριθμος (algorithm) προέρχεται από το όνομα ενός Πέρση μαθηματικού, του Abu Ja'far Mohammed ibn Musa al Khowarizmi, ο οποίος έζησε τον 9ο αιώνα μ.Χ. Η λέξη "al Khowarizmi" σημαίνει "από το Khowarazm" που είναι η σημερινή πόλη Khiva του Ουζμπεκιστάν. Η λέξη, λοιπόν, αλγόριθμος προέρχεται από το Khowarizmi, που ήταν η πατρίδα του μαθηματικού αυτού.

Μολονότι η έννοια και η λέξη αλγόριθμος δεν ανάγονται στο απώτατο παρελθόν, εντούτοις ουσιαστικά είχαν συλληφθεί στην Αρχαία Ελλάδα. Αρκεί να θυμηθούμε το γνωστό μας κόσκινο του Ερατοσθένη για την εύρεση πρώτων αριθμών ή τη μέθοδο του Ευκλείδη για την εύρεση του μέγιστου κοινού διαιρέτη δύο αριθμών. Αλλά μήπως κάθε ενέργεια στην καθημερινή μας ζωή δεν μπορεί να χαρακτηρισθεί ως αλγόριθμος; Για παράδειγμα, τι είναι μία συνταγή μαγειρικής ή η διαδικασία ανάληψης χρημάτων από μία αυτόματη ταμειακή μηχανή σε μία τράπεζα.

Ο όρος αλγόριθμος χρησιμοποιείται για να δηλώσει μεθόδους που εφαρμόζονται σε προγράμματα για την επίλυση προβλημάτων. Ωστόσο, ένας αναλυτικότερος ορισμός της έννοιας αυτής είναι ο εξής:

Αλγόριθμος: Αλγόριθμος είναι ένα πεπερασμένο σύνολο εντολών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, οι οποίες όταν ακολουθηθούν επιτυγχάνεται ένα επιθυμητό αποτέλεσμα ή επιλύεται ένα συγκεκριμένο πρόβλημα.

Επιπροσθέτως, μία ακολουθία εντολών πρέπει να ικανοποιεί τα ακόλουθα κριτήρια ώστε να θεωρείται αλγόριθμος:

1. **Είσοδος (input).** Καμία, μία ή περισσότερες ποσότητες να δίνονται ως είσοδοι στον αλγόριθμο.
2. **Εξοδος (output).** Ο αλγόριθμος να δημιουργεί τουλάχιστον μία ποσότητα ως αποτέλεσμα.
3. **Καθοριστικότητα (definiteness).** Κάθε εντολή να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της.
4. **Περατότητα (finiteness).** Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα πεπερασμένο αριθμό βημάτων λέγεται απλώς υπολογιστική διαδικασία (computational procedure).

5. **Αποτελεσματικότητα (effectiveness).** Κάθε μεμονωμένη εντολή του αλγορίθμου να επαρκώς απλή έτσι ώστε να μπορεί να εκτελεστεί από ένα άτομο με χρήση χαρτιού και μολυβιού. Δεν αρκεί δηλαδή να είναι ορισμένη (κριτήριο 3) αλλά πρέπει να είναι και εκτελέσιμη.

Στην πράξη διακρίνει κανείς ανάμεσα σε αλγορίθμους και προγράμματα. Τα δεύτερα είναι υλοποιήσεις των πρώτων με τις εντολές και τα εργαλεία ενός προγραμματιστικού περιβάλλοντος. Η διαφορά είναι ότι ένα πρόγραμμα δεν ικανοποιεί αναγκαστικά το 4ο κριτήριο της περατότητας (σκεφθείτε π.χ. το λειτουργικό σύστημα, το οποίο εκτελεί διαρκώς έναν ατέρμονα βρόχο περιμένοντας εντολές να εισέλθουν στο σύστημα). Στην πτυχιακή εργασία θα ασχοληθούμε με αλγορίθμους που πληρούν το κριτήριο αυτό (πάντα τερματίζουν).

Λόγω της σημαντικότητας του αντικειμένου και της κεντρικότητας τους στην επιστήμη, η Πληροφορική συχνά ορίζεται ως η επιστήμη που μελετά τους αλγορίθμους από τη σκοπιά:

1. **Του υλικού:** Η ταχύτητα εκτέλεσης του αλγορίθμου εξαρτάται από την αρχιτεκτονική του Η/Υ. Οι διάφορες τεχνολογίες υλικού επηρεάζουν την αποδοτικότητα ενός αλγορίθμου (όπως π.χ. στον τομέα της σχεδίασης ολοκληρωμένων κυκλωμάτων VLSI).
2. **Των γλωσσών προγραμματισμού**: Η δομή του αλγορίθμου θα είναι ανάλογη με την γλώσσα προγραμματισμού με την οποία τελικά θα εκτελεστεί. Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (χαμηλού ή υψηλού επιπέδου) αλλάζει τη μορφή και τον αριθμό των εντολών ενός αλγορίθμου (σε γλώσσα assembly π.χ. η υλοποίηση των δομών επανάληψης γίνεται με χρήση της εντολής διακλάδωσης – jump).
3. **Τη θεωρητική σκοπιά:** Πριν ο αλγόριθμος εφαρμοστεί ελέγχεται αν πράγματι είναι αποδοτικός για την λύση του συγκεκριμένου προβλήματος.
4. **Την αναλυτική σκοπιά:** Όταν αναπτύσσεται ο αλγόριθμος λαμβάνονται υπόψη οι υπολογιστικοί πόροι (computer resources) που θα απαιτηθούν για την εκτέλεσή του όπως το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για CPU και για I/O κλπ.

2.3.1 Ανάλυση της απόδοσης αλγορίθμων

Συνήθως, σε κάποιο δεδομένο πρόβλημα μπορούμε να εφεύρουμε πολλές εναλλακτικές λύσεις. Είναι αναγκαία επομένως η αξιολογική κατάταξη των εναλλακτικών λύσεων ώστε να επιλεγεί η προσφορότερη. Μιλώντας με τεχνικούς όρους, σκοπός μας είναι να προσδιορίσουμε την επίδοση (performance) ή την αποδοτικότητα (efficiency) του κάθε αλγορίθμου. Τα βασικά κριτήρια για την επιλογή ενός αλγορίθμου είναι οι απαιτήσεις του σε υπολογιστικούς πόρους, δηλαδή:

- απαιτούμενος χρόνος εκτέλεσης

και

- απαιτούμενος χώρος αποθήκευσης.

Καθώς πλέον η μνήμες (κύριες και δευτερεύουσες) γίνονται συνεχώς μεγαλύτερες και φθηνότερες, ο απαιτούμενος χρόνος εκτέλεσης είναι τελικά το σημαντικότερο κριτήριο. Άλλα κριτήρια, όπως η δικτυακή χωρητικότητα ή ο αριθμός των πυλών, είναι πολύ μικρότερης σημασίας.

Υπάρχουν δύο τρόποι για να εξετάσουμε την επίδοση ενός αλγορίθμου:

- πρακτικός ή εκ των υστέρων (a posteriori)

και

- θεωρητικός ή εκ των προτέρων (a priori).

Σύμφωνα με τον πρώτο τρόπο, κάθε αλγόριθμος εξετάζεται σε έναν υπολογιστή και χρονομετρείται καθώς η είσοδός του τροφοδοτείται με διάφορα δεδομένα. Αυτός ο τρόπος δεν είναι σωστός αν θεωρήσουμε ότι δεν λαμβάνει υπόψη του μία σειρά παραμέτρων, όπως:

- Το χρησιμοποιούμενο υλικό, δηλαδή, τα συγκεκριμένα χαρακτηριστικά του υπολογιστή, όπως ταχύτητα του επεξεργαστή, το μέγεθος της κύριας ή της δευτερεύουσας μνήμης, τη χρήση κρυφής μνήμης (cache) κοκ.
- Τη χρησιμοποιούμενη γλώσσα προγραμματισμού, καθώς είναι γνωστό ότι υπάρχουν ταχύτερες και βραδύτερες γλώσσες με περισσότερη ή λιγότερη πρόσβαση στο φυσικό επίπεδο (όπως, για παράδειγμα οι παλαιότερες γλώσσες σε αντίθεση με τις νεότερες C/C++).
- Το χρησιμοποιούμενο μεταγλωττιστή/ερμηνευτή, καθώς είναι δυνατόν να υπάρχουν διαθέσιμοι εμπορικά (ή και δωρεάν μέσα από το διαδίκτυο)

περισσότεροι του ενός μεταγλωττιστές που μάλιστα μπορεί να τρέχουν σε διαφορετικά λειτουργικά συστήματα (όπως, για παράδειγμα, Unix, Linux, Windows κοκ).

- Τον προγραμματιστή που χρησιμοποιεί όλα τα προηγούμενα, καθώς ο ανθρώπινος παράγοντας είναι σημαντικότατος σε κάθε περίπτωση.
- Τα χρησιμοποιούμενα δεδομένα, γιατί κάθε αλγόριθμος ταξινόμησης συμπεριφέρεται διαφορετικά, ανάλογα με τη φύση των δεδομένων στην είσοδο.

Καταλήγουμε, λοιπόν στο συμπέρασμα, ότι όταν χρησιμοποιούμε τον πρακτικό τρόπο ανάλυσης αλγορίθμων θα πρέπει λαμβάνουμε υπόψη όλες τις παραπάνω παραμέτρους.

2.4 Βάσεις Δεδομένων

Βάση δεδομένων (database) είναι μία συλλογή από σχετιζόμενα δεδομένα. Με τον όρο δεδομένα εννοούμε γνωστά γεγονότα που μπορούν να καταγραφούν και που έχουν κάποια υπονοούμενη σημασία. Για παράδειγμα μπορούμε να θεωρήσουμε τα ονόματα, τους αριθμούς τηλεφώνων και τις διευθύνσεις των ατόμων που γνωρίζουμε.

Η τεχνολογία των βάσεων δεδομένων είναι από τις πλέον κρίσιμες, καθώς η πλειονότητα των πληροφοριακών συστημάτων αφορούν την αποθήκευση και διαχείριση μεγάλων όγκων δεδομένων, σε ένα ευρύ πεδίο εφαρμογών:

- εμπορικές (αποθήκες, μισθοδοσία, τιμολόγια κ.τ.λ.)
- δημόσια διοίκηση (εφορία, ΔΕΗ, ΟΤΕ κ.τ.λ.)
- υπηρεσίες (αεροπορικές εταιρείες)

Με την τεχνολογία των βάσεων δεδομένων η διαχείριση των δεδομένων γίνεται ανεξάρτητα από την εφαρμογή που τα χρησιμοποιεί. Έτσι, ικανοποιείται η απαίτηση για:

- έλεγχο πλεονασμών
- επιβολή περιορισμών ορθότητας
- παράσταση πολύπλοκων συσχετίσεων μεταξύ των δεδομένων
- υποστήριξη πολλαπλών όψεων των δεδομένων
- περιορισμό της μη εξουσιοδοτημένης προσπέλασης
- αποδοτικότητα στη διαχείριση των δεδομένων
- διαχείριση των δεδομένων με δοσοληψίες

Η ίδια η βάση δεδομένων μπορεί να θεωρηθεί ένα είδος ηλεκτρονικής αρχειοθήκης, ένα χώρος για την αποθήκευση μιας συλλογής ηλεκτρονικών αρχείων δεδομένων.

Ο χρήστης του συστήματος έχει στη διάθεση του ορισμένα βοηθήματα για να εκτελεί στα αρχεία βάσεων δεδομένων διάφορες εργασίες, στις οποίες συγκαταλέγονται, ανάμεσα σε άλλες, και οι εξής:

- Η προσθήκη νέων κενών αρχείων στη βάση δεδομένων .
- Η εισαγωγή νέων δεδομένων σε υπάρχοντα αρχεία.
- Η ανάκληση δεδομένων από υπάρχοντα αρχεία.
- Η ενημέρωση δεδομένων σε υπάρχοντα αρχεία.
- Η διαγραφή δεδομένων από υπάρχοντα αρχεία
- Η αφαίρεση υπαρχόντων αρχείων, κενών ή όχι, από τη βάση δεδομένων.

Ένα σύστημα βάσης δεδομένων απαρτίζεται από τέσσερα βασικά στοιχεία: τα δεδομένα, το υλικό, το λογισμικό και τους χρήστες.

2.4.1 Πλεονεκτήματα των Βάσεων Δεδομένων

Τα πλεονεκτήματα ενός συστήματος βάσης δεδομένων, σε σύγκριση με τις παραδοσιακές μεθόδους παρακολούθησης (χαρτί και μολύβι), αρχικά για μια μικρή βάση δεδομένων είναι πολλά:

- Οικονομία χώρου (Καταργούνται τα ογκώδη παραδοσιακά αρχεία με φακέλους και έγγραφα).

- Ταχύτητα (το σύστημα μπορεί να ανακαλεί και να αλλάζει τα δεδομένα γρηγορότερα από τον άνθρωπο)
- Λιγότερος κόπος
- Άμεση πληροφόρηση (Ακριβείς και ενημερωμένες πληροφορίες είναι διαθέσιμες κάθε στιγμή).

Επιπλέον, σε ένα περιβάλλον πολλών χρηστών, το σύστημα βάσης δεδομένων παρέχει στην επιχείρηση κεντρικό έλεγχο των δεδομένων της ώστε να προκύπτουν τα εξής πλεονεκτήματα:

- Ο πλεονασμός μειώνεται στο ελάχιστο. Στα συμβατικά συστήματα (εκείνα που δεν είναι συστήματα βάσεων δεδομένων), η κάθε εφαρμογή έχει τα δικά της αρχεία. Αυτό το γεγονός οδηγεί πολύ συχνά σε υψηλό βαθμό πλεονασμού (επανάληψης) για τα αποθηκευμένα δεδομένα, με αποτέλεσμα τη σπατάλη αποθηκευτικού χώρου. Θα πρέπει εδώ να ξεκαθαρίσουμε ότι αυτό δε σημαίνει πως είναι πάντα δυνατό να εξαλειφθούν όλοι οι πλεονασμοί, ούτε πως είναι πάντα επιθυμητό. Μερικές φορές υπάρχουν σοβαροί επιχειρηματικοί ή τεχνικοί λόγοι που επιβάλλουν να τηρούνται ξεχωριστά αντίγραφα των ίδιων αποθηκευμένων δεδομένων.
- Η ασυνέπεια μπορεί να αποφευχθεί (ως ένα βαθμό).Το DBMS θα μπορεί να εγγυηθεί ότι η βάση δεδομένων δε θα είναι ποτέ ασυνεπής στα μάτια του χρηστή ,εξασφαλίζοντας ότι κάθε αλλαγή που θα γίνεται σε οποιαδήποτε από δυο όμοιες καταχωρίσεις θα γίνεται αυτόματα και στην άλλη. Αυτή η διαδικασία είναι γνωστή με το όνομα διάδοση ενημερώσεων.
- Τα δεδομένα μπορούν να είναι κοινόχρηστα. Ο μερισμός δε σημαίνει μόνο ότι οι υπάρχουσες εφαρμογές μπορούν να μοιράζονται τα δεδομένα της βάσης δεδομένων αλλά και ότι είναι δυνατή η ανάπτυξη νέων εφαρμογών που θα μπορούν να χρησιμοποιούν τα ίδια αποθηκευμένα δεδομένα.
- Μπορούν να επιβάλλονται πρότυπα. Η τυποποίηση της αναπαράστασης των δεδομένων διευκολύνει ιδιαίτερα την ανταλλαγή δεδομένων. Τα πρότυπα ονομασίας και τεκμηρίωσης των δεδομένων είναι επίσης πολύ

επιθυμητά για να διευκολύνεται ο μερισμός και η καλύτερη κατανόηση των δεδομένων.

- Μπορούν να εφαρμόζονται περιορισμοί ασφαλείας .Έχοντας πλήρη δικαιοδοσία πάνω στη βάση δεδομένων, ο διαχειριστής της (α) μπορεί να εξασφαλίσει ότι η πρόσβαση στη βάση δεδομένων θα μπορεί να γίνεται μόνο μέσω των κατάλληλων καναλιών και, κατά συνέπεια, (β) μπορεί να ορίσει κανόνες ασφαλείας με βάση τους οποίους θα γίνεται έλεγχος κάθε φορά που θα υπάρχει απόπειρα προσπέλασης εμπιστευτικών δεδομένων. Βέβαια ένα συστήματα βάσης δεδομένων απαιτεί την ύπαρξη ενός καλού συστήματος ασφαλείας.
- Μπορεί να διατηρείται η ακεραιότητα .Το πρόβλημα της ακεραιότητας είναι να εξασφαλίζεται ότι τα δεδομένα της βάσης δεδομένων είναι ακριβή. Η ασυμφωνία μεταξύ δυο καταχωρίσεων που υποτίθεται ότι αντιπροσωπεύουν το ίδιο "γεγονός" είναι ένα παράδειγμα έλλειψης ακεραιότητας φυσικά, αυτό το συγκεκριμένο πρόβλημα μπορεί να παρουσιαστεί μόνο αν υπάρχει πλεονασμός στα αποθηκευμένα δεδομένα. Ακόμη και αν δεν υπάρχει πλεονασμός όμως, πάλι υπάρχει περίπτωση να περιέχει η βάση δεδομένων λανθασμένες πληροφορίες. Αξίζει να επισημάνουμε ότι η ακεραιότητα των δεδομένων έχει πολύ μεγαλύτερη σημασία σε ένα σύστημα βάσης δεδομένων πολλών χρηστών από ότι σε ένα περιβάλλον "ιδιωτικών αρχείων ", ακριβώς επειδή η βάση δεδομένων είναι μεριζόμενη. Αυτό συμβαίνει γιατί, χωρίς τους κατάλληλους ελέγχους, μπορεί ένας χρήστης να ενημερώσει τη βάση δεδομένων με εσφαλμένο τρόπο, δημιουργώντας με αυτό τον τρόπο λανθασμένα δεδομένα και " μολύνοντας " τους υπόλοιπους "
- Οι αντικρουόμενες απαιτήσεις μπορούν να εξισορροπούνται. Γνωρίζοντας τις συνολικές απαιτήσεις της επιχείρησης, σε αντιδιαστολή με τις απαιτήσεις των μεμονωμένων χρηστών, ο διαχειριστής της βάσης δεδομένων (πάντα με τις οδηγίες του υπεύθυνου διαχείρισης δεδομένων) μπορεί να δομήσει το σύστημα με τέτοιον τρόπο ώστε να παρέχει γενικές υπηρεσίες που να είναι " βέλτιστες για την επιχείρηση". Για παράδειγμα μπορεί να επιλεχθεί μια αναπαράσταση των αποθηκευμένων δεδομένων που να παρέχει γρήγορη πρόσβαση στις σημαντικές εφαρμογές ίσως σε βάρος της απόστασης άλλων εφαρμογών.

2.4.2 Σύστημα Βάσεων Δεδομένων (ΣΒΔ)

Ένα Σύστημα Βάσης Δεδομένων (ΣΒΔ) ή Data Base System (DBS) αποτελείται από το υλικό, το λογισμικό, τη βάση δεδομένων και τους χρήστες. Είναι δηλαδή ένα σύστημα με το οποίο μπορούμε να αποθηκεύσουμε και να αξιοποιήσουμε δεδομένα με τη βοήθεια ηλεκτρονικού υπολογιστή. Αναλυτικά :

- Το υλικό (hardware) αποτελείται όπως είναι γνωστό από τους ηλεκτρονικούς υπολογιστές, τα περιφερειακά, τους σκληρούς δίσκους, τις μαγνητικές ταινίες κ.ά., όπου είναι αποθηκευμένα τα αρχεία της βάσης δεδομένων αλλά και τα προγράμματα που χρησιμοποιούνται για την επεξεργασία τους.
- Το λογισμικό (software) είναι τα προγράμματα που χρησιμοποιούνται για την επεξεργασία των δεδομένων (στοιχείων) της βάσης δεδομένων.
- Η βάση δεδομένων (database) αποτελείται από το σύνολο των αρχείων όπου είναι αποθηκευμένα τα δεδομένα του συστήματος. Τα στοιχεία αυτά μπορεί να βρίσκονται αποθηκευμένα σ' έναν φυσικό υπολογιστή αλλά και σε περισσότερους. Όμως, στον χρήστη δίνεται η εντύπωση ότι βρίσκονται συγκεντρωμένα στον ίδιο υπολογιστή. Τα δεδομένα των αρχείων αυτών είναι ενοποιημένα (data integration), δηλ. δεν υπάρχει πλεονασμός (άσκοπη επανάληψη) δεδομένων και μερισμένα (data sharing), δηλ. υπάρχει δυνατότητα ταυτόχρονης προσπέλασης των δεδομένων από πολλούς χρήστες. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και βλέπει διαφορετικό κομμάτι της βάσης δεδομένων, ανάλογα με τον σκοπό για τον οποίο συνδέεται.
- Οι χρήστες (users) μιας βάσης δεδομένων χωρίζονται στις εξής κατηγορίες :
- Τελικοί χρήστες (end users). Χρησιμοποιούν κάποια εφαρμογή για να παίρνουν στοιχεία από μια βάση δεδομένων, έχουν τις λιγότερες δυνατότητες επέμβασης στα στοιχεία της βάσης δεδομένων, χρησιμοποιούν ειδικούς κωδικούς πρόσβασης και το σύστημα τούς επιτρέπει ανάλογα πρόσβαση σε συγκεκριμένο κομμάτι της βάσης δεδομένων.
- Προγραμματιστές εφαρμογών (application programmers). Αναπτύσσουν τις εφαρμογές του ΣΒΔ σε κάποια από τις γνωστές γλώσσες προγραμματισμού.

- Διαχειριστής δεδομένων (data administrator – DA). Έχει τη διοικητική αρμοδιότητα και ευθύνη για την οργάνωση της βάσης δεδομένων και την απόδοση δικαιωμάτων πρόσβασης στους χρήστες.
- Διαχειριστής βάσης δεδομένων (database administrator – DBA). Λαμβάνει οδηγίες από τον διαχειριστή δεδομένων και είναι αυτός που διαθέτει τις τεχνικές γνώσεις και αρμοδιότητες για τη σωστή και αποδοτική λειτουργία του ΣΔΒΔ.

2.4.3 Διαχειριστές Βάσεων Δεδομένων (Database Administrators)

Οι διαχειριστές (Database Administrators) είναι υπεύθυνοι για την διαχείριση της Βάσης δεδομένων παρέχοντας την απαιτούμενη τεχνική υποστήριξη για την υλοποίηση αυτών των αποφάσεων. Οι διαχειριστές βάσεων δεδομένων είναι λοιπόν υπεύθυνοι για το συνολικό έλεγχο του συστήματος σε τεχνικό επίπεδο. Γενικά, αυτές οι λειτουργίες είναι οι εξής:

- Ορισμός του εννοιολογικού σχήματος. Αφού ο υπεύθυνος διαχείρισης δεδομένων καθορίσει τη λογική ή μερικές φορές εννοιολογική σχεδίαση της βάσης δεδομένων εντοπίζοντας τις σημαντικές οντότητες και άρα και τις σχετικές με αυτές πληροφορίες που θα τηρούνται, ο διαχειριστής της βάσης δεδομένων δημιουργεί το αντίστοιχο εννοιολογικό σχήμα, χρησιμοποιώντας την εννοιολογική γλώσσα ορισμού δεδομένων (Data Definition Language - DDL).
- Ορισμός του εσωτερικού σχήματος. Αποφασίζει για τη φυσική σχεδίαση της βάσης δεδομένων, και δημιουργεί τον αντίστοιχο ορισμό αποθηκευτικής δομής, χρησιμοποιώντας την εσωτερική DDL. Ο διαχειριστής της βάσης δεδομένων πρέπει επίσης να ορίσει και τη σχετική απεικόνιση μεταξύ του εσωτερικού και του εννοιολογικού σχήματος. Οι αντίστοιχες απεικονίσεις θα υπάρχουν και σε μορφή πηγαίου κώδικα και σε μορφή αντικειμένου κώδικα.
- Επαφή με τους χρήστες. Εξασφαλίζει ότι τα δεδομένα που χρειάζονται είναι διαθέσιμα. Γράφει τα απαραίτητα εξωτερικά σχήματα, χρησιμοποιώντας την κατάλληλη εξωτερική DDL ώστε να οριστεί η απεικόνιση ανάμεσα σε

οποιοδήποτε δεδομένο εξωτερικό σχήμα και στο εννοιολογικό σχήμα. Συγχρόνως παρέχει συμβουλές για την σχεδίαση των εφαρμογών ,τεχνική εκπαίδευση, εντόπιση και επίλυση προβλημάτων.

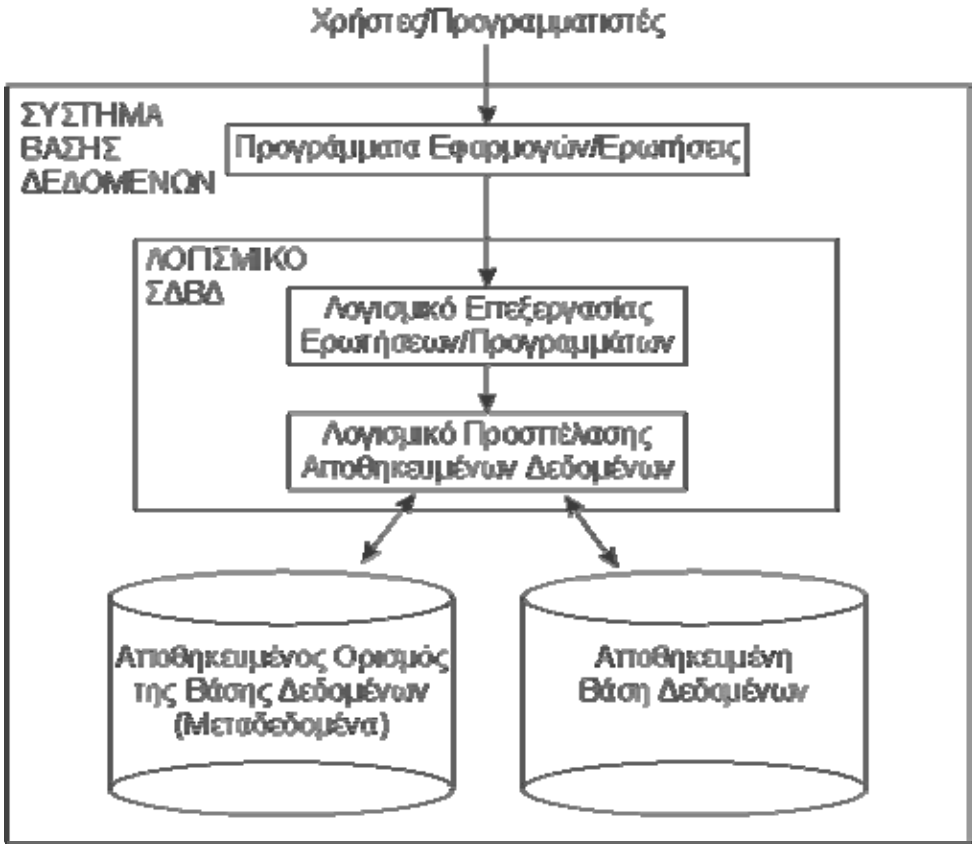
- Ορισμός κανόνων ασφάλειας και ακεραιότητας. Οι κανόνες ασφάλειας και ακεραιότητας μπορούν να θεωρηθούν μέρος του εννοιολογικού σχήματος. Η εννοιολογική DDL θα πρέπει να διαθέτει λειτουργίες για τον καθορισμό τέτοιων κανόνων.
- Ορισμός διαδικασιών για τη λήψη εφεδρικών αντιγράφων και την ανάκαμψη

Από τη στιγμή που μια επιχείρηση υιοθετεί ένα σύστημα βάσης δεδομένων, εξαρτάται σε κρίσιμο βαθμό από την επιτυχημένη λειτουργία αυτού του συστήματος. Σε περίπτωση που παρουσιαστεί μια αστοχία σε οποιοδήποτε μέρος της βάσης δεδομένων που μπορεί να οφείλεται είτε σε ανθρώπινο σφάλμα είτε σε κάποια αστοχία του υλικού ή του λειτουργικού συστήματος είναι απαραίτητο να μπορούν να αποκατασταθούν τα δεδομένα που επηρεάστηκαν, με ελάχιστη δυνατή καθυστέρηση και με τις ελάχιστες δυνατές επιπτώσεις στο υπόλοιπο σύστημα .Ο DBA θα πρέπει να ορίσει και να υλοποιήσει έναν κατάλληλο μηχανισμό ανάκαμψης.

2.4.4 Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)

Σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) (database management system - DBMS) είναι μια συλλογή από προγράμματα που επιτρέπουν στους χρήστες να δημιουργήσουν και να συντηρήσουν μία βάση δεδομένων. Επομένως, το ΣΔΒΔ είναι ένα γενικής χρήσης σύστημα λογισμικού που διευκολύνει τις διαδικασίες ορισμού, κατασκευής και χειρισμού βάσεων δεδομένων για διάφορες εφαρμογές.

Ο στόχος ενός Συστήματος διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) είναι να παράσχει ένα περιβάλλον που είναι και βολικό και αποδοτικό για την ανάκτηση των πληροφοριών από τη βάση δεδομένων.



Εικόνα 8: Σύστημα διαχείρισης βάσεων δεδομένων

Το ΣΔΒΔ παρέχει στους χρήστες και τις εφαρμογές που αναπτύσσουν οι προγραμματιστές ένα επίπεδο λογισμικού για την επεξεργασία ερωτήσεων και προγραμμάτων. Για την πρόσβαση στον ορισμό της βάσης και τα δεδομένα χρησιμοποιείται το επίπεδο προσπέλασης αποθηκευμένων δεδομένων.

Τα επιθυμητά χαρακτηριστικά ενός ΣΔΒΔ είναι τα ακόλουθα:

- μείωση πλεοναζόντων δεδομένων
- ευκολία στην ανάπτυξη νέων εφαρμογών
- μηχανισμοί ασφαλείας
- υποστήριξη δοσοληψιών (transactions)
- ανεξαρτησία δεδομένων από τις εφαρμογές τόσο στο φυσικό επίπεδο της αποθήκευσης, όσο και στις διαφορετικές παρεχόμενες όψεις

Η χρήση ΣΔΒΔ σε σχέση με αποθήκευση σε απλά αρχεία πλεονεκτεί:

- στην ασφάλεια δεδομένων
- στη διαφύλαξη της ακεραιότητας των δεδομένων
- στην ταυτόχρονη πρόσβαση από πολλούς χρήστες και εφαρμογές
- στην καταγραφή στοιχείων χρήσης
- στη ρύθμιση του τρόπου αποθήκευσης σε διαφορετικούς δίσκους και υπολογιστές
- στη ρύθμιση της οργάνωσης της αποθήκευσης των δεδομένων για την βελτιστοποίηση της απόδοσης (π.χ. χρήση δομών δέντρων)
- στη γρήγορη πρόσβαση στα δεδομένα

2.4.5 Οι Πίνακες Βάσεων Δεδομένων

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City
▶	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representati	Obere Str. 57	Berlin
	ANATR	Ana Trujillo Empare	Ana Trujillo	Owner	Avda. de la Constit	México D.F.
	ANTON	Antonio Moreno Ta	Antonio Moreno	Owner	Mataderos 2312	México D.F.
	AROUT	Around the Horn	Thomas Hardy	Sales Representati	120 Hanover Sq.	London
	BERGS	Berglunds snabbköj	Christina Berglund	Order Administrato	Berguvsvägen 8	Luleå
	BLAUS	Blauer See Delikate	Hanna Moos	Sales Representati	Forsterstr. 57	Mannheim
	BLONP	Blondesddsl père el	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg
	BOLID	Bólido Comidas pre	Martín Sommer	Owner	C/ Araquil, 67	Madrid
	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouche	Marseille
	BOTTM	Bottom-Dollar Mark	Elizabeth Lincoln	Accounting Manage	23 Tsawassen Blvd	Tsawassen
	BSBEV	B's Beverages	Victoria Ashworth	Sales Representati	Fauntleroy Circus	London
	CACTU	Cactus Comidas pa	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires
	CENTC	Centro comercial M	Francisco Chang	Marketing Manager	Sierras de Granada	México D.F.
	CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern
	COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 2	Sao Paulo
	CONSH	Consolidated Holdir	Elizabeth Brown	Sales Representati	Berkeley Gardens 1	London
	DRACD	Drachenblut Delikat	Sven Ottlieb	Order Administrato	Walserweg 21	Aachen
	DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinqua	Nantes

Εικόνα 9: Πίνακας βάσης δεδομένων

Οι πίνακες είναι η πιο απλή δομή δεδομένων. Οι βάσεις δεδομένων περιέχουν αντικείμενα που ονομάζονται Πίνακες. Οι Εγγραφές των δεδομένων αποθηκεύονται σ' αυτούς τους πίνακες. Οι Πίνακες περιέχουν Στήλες και Γραμμές με δεδομένα. Οι Γραμμές περιέχουν εγγραφές, όπως μία εγγραφή για κάθε άτομο. Οι Στήλες περιέχουν δεδομένα, όπως First Name, Last Name, Address και City. Πλεονεκτήματα αυτής της δομής δεδομένων είναι η απλότητά της και η άμεση υποστήριξή της από τις περισσότερες γλώσσες προγραμματισμού (π.χ. Pascal). Η στατική φύση των πινάκων είναι το σημαντικότερο μειονέκτημα τους, που τους καθιστά δύσχρηστους σε αλγόριθμους που απαιτείται ευελιξία.

Τα δεδομένα μιας βάσης δεδομένων αποθηκεύονται (οργανώνονται) στις εξής στοιχειώδεις μορφές :

- *Πεδίο (Field)*, είναι το μικρότερο κομμάτι δεδομένων στο οποίο μπορούμε να αναφερθούμε και περιέχει ένα μόνο χαρακτηριστικό ή ιδιότητα ενός στοιχείου της βάσης δεδομένων.
- *Εγγραφή (Record)*, είναι ένα σύνολο από διαφορετικά πεδία που περιέχει όλες τις πληροφορίες για ένα στοιχείο της βάσης δεδομένων.
- *Αρχείο (File)*, είναι ένα σύνολο από πολλά παρόμοια στοιχεία (εγγραφές) της βάσης δεδομένων.
- *Πρωτεύον Κλειδί (Primary Key)*, είναι ένα πεδίο ή συνδυασμός πεδίων που χαρακτηρίζει μοναδικά μια εγγραφή.
- *Κλειδί (Key)*, είναι ένα πεδίο που δεν έχει κατ' ανάγκη μοναδική τιμή και που μπορούμε να το χρησιμοποιήσουμε για να κάνουμε αναζήτηση σ' ένα αρχείο.
- *Ξένο Κλειδί (Foreign Key)*, είναι ένα πεδίο που έχει το ίδιο σύνολο τιμών με το πρωτεύον κλειδί ενός άλλου πίνακα.

2.5 Προγραμματισμός

Στο σημείο αυτό θα αναπτύξουμε την έννοια του προγραμματισμού για τις γλωσσά προγραμματισμού C++ και SQL, παράλληλα θα αναφέρουμε και ορισμένα ιστορικά στοιχεία.

2.5.1 Η γλώσσα C++

Η γλώσσα προγραμματισμού C αναπτύχθηκε στην AT&T με σκοπό τη δημιουργία ενός λειτουργικού συστήματος για τη σειρά υπολογιστών PDP-11, που τελικά έγινε το λειτουργικό σύστημα Unix. Η C αναπτύχθηκε με πρωταρχικό σκοπό την αποδοτικότητα. Η γλώσσα προγραμματισμού C ορίστηκε αρχικά στο κλασσικό σύγγραμμα των Kernigham και Ritchie "The C Programming Language", και ήταν το πρότυπο που χρησιμοποιούσαν όλοι οι προγραμματιστές στη C. Το πρότυπο ANSI για τη C τελικά εγκρίθηκε τον Δεκέμβριο του 1989 και έγινε το επίσημο πρότυπο για τον προγραμματισμό στη C. Το πρότυπο ANSI εισήγαγε αρκετά νέα στοιχεία, που δεν υπήρχαν στην αρχική έκδοση των Kernigham και Ritchie, και άλλαξε κάποια άλλα, έτσι ώστε τα δύο πρότυπα δεν είναι τελείως συμβατά.

Η C++ είναι μια γλώσσα προγραμματισμού «γενικού προορισμού». Κατά την δεκαετία του 90 έγινε η δημοφιλέστερη γλώσσα. Ο Bjarne Stroustrup από τα Bell Labs ανέπτυξε την C++ μέσα στην δεκαετία του 80 (αρχικά ονομάστηκε C από το Classes – κλάσεις)σαν μια βελτίωση της μέχρι τότε γλώσσας C. Επειδή η αντικειμενοστραφής τεχνολογία ήταν καινούργια και όλες οι αντικειμενοστραφείς υλοποιήσεις που υπήρχαν ήταν αρκετά αργές και μη αποδοτικές, ένας δευτερεύων σκοπός της C++ ήταν να διατηρήσει την αποδοτικότητα της C. Το ++ σημαίνει απλά την αύξηση κατά 1 και το C++ απλά δηλώνει την βελτιστοποίηση της C. Η C++ μπορεί να θεωρηθεί μια διαδικαστική γλώσσα με κάποιες επιπλέον δομές, μερικές από τις οποίες προστέθηκαν για αντικειμενοστραφή προγραμματισμό, ενώ άλλες για την βελτίωση του συντακτικού της γλώσσας. Ένα καλογραμμένο πρόγραμμα πρέπει να έχει στοιχεία τόσο αντικειμενοστραφή όσο και κλασσικού διαδικαστικού προγραμματισμού. Η C++ είναι ουσιαστικά μια επεκτάσιμη γλώσσα αφού μπορούμε να ορίσουμε νέους τύπους με τέτοιο τρόπο ώστε να λειτουργούν σαν τους προκαθορισμένους τύπους, που είναι τμήμα της γλώσσας. Η C++ σχεδιάστηκε για

την ανάπτυξη μεγάλων προγραμμάτων και υποστηρίζει τον αντικειμενοστραφή τρόπο προγραμματισμού.

Γενικά η C++ σχεδιάστηκε για να:

- Είναι μια γενικού προορισμού γλώσσα προγραμματισμού .
- Να δίνει την δυνατότητα πολλών μεθόδων προγραμματισμού όπως αντικειμενοστραφής, procedural programming, data abstraction και generic programming .
- Για να δίνει την δυνατότητα επιλογής στον χρήστη .
- Για να είναι όσο πιο συμβατή γίνεται με την C .
- Λειτουργεί δίχως κάποιο περίπλοκο προγραμματιστικό περιβάλλον (το απλό notepad αρκεί).
- Για να αποφύγει στοιχεία που βασίζονται πάνω στο τύπο του λογισμικού που χρησιμοποιείται και του είδους της πλατφόρμας.

Πολλά προγράμματα χρησιμοποιούνε την C++ όπως η Visual Basic ,C++ Builder, Visual C++, .NET Framework κ.α.

2.5.2 Αντικειμενοστραφής Προγραμματισμός

Ο αντικειμενοστραφής προγραμματισμός (Object–oriented programming - OOP) είναι ένα είδος προγραμματισμού υπολογιστών στην οποία το λογισμικό μοντελοποιείται ως ένα σύνολο από αντικείμενα τα οποία αλληλεπιδρούν μεταξύ τους.

Συγκεκριμένα ο αντικειμενοστραφής προγραμματισμός δίνει έμφαση στα παρακάτω :

Objects (Αντικείμενα): Είναι η βάση της δομής του αντικειμενοστραφή προγραμματισμού και περιέχουμε δεδομένα και συναρτήσεις

Abstraction (αφαιρετικότητα): Είναι η δυνατότητα του προγράμματος να αγνοεί ορισμένα στοιχεία των πληροφοριών τα οποία δεν χρησιμοποιούνται την παρούσα στιγμή

Encapsulation (ενθυλάκωση): Βεβαιώνει ότι ο χρήστης ενός αντικειμένου δεν μπορεί να αλλάξει την δομή ενός αντικειμένου αλλά μονάχα να χρησιμοποιήσει της δικιές του μεθόδους

Polymorphism (πολυμορφισμός): Διαφορετικά αντικείμενα μπορούνε να δώσουνε την ίδια απάντηση σε κάποια κλήση επιτρέποντας έτσι την συνδιαλλαγή μεταξύ διαφορετικών αντικειμένων

Inheritance (κληρονομικότητα): Οργανώνει τα 2 παραπάνω επιτρέποντας σε κάποια αντικείμενα να έχουν τις ίδιες δυνατότητες και μεθόδους με κάποια άλλα. Επιτυγχάνεται με τη χρήση κλάσεων –class και αποτελεί μια από τις πιο ευρέως χρησιμοποιούμενες μεθόδου.

Μερικές γλώσσες αντικειμενοστραφή προγραμματισμού είναι οι C, C++, Python, Java, Perl, PHP και άλλες.

2.5.3 Οπτικός Προγραμματισμός

Μια γλώσσα οπτικού προγραμματισμού είναι εκείνη που επιτρέπει τον χρήστη να δημιουργήσει ένα πρόγραμμα σε 2 ή περισσότερες διαστάσεις. Οι τυπικές γλώσσες κειμένου δεν είναι πολυδιάστατες καθώς ο compiler επεξεργάζεται μόνο μια διάσταση σειράς χαρακτήρων. Μια γλώσσα οπτικού προγραμματισμού επιτρέπει την χρήση οπτικών προγραμματιστικών εκφράσεων. Γενικά ένα τέτοιο περιβάλλον παρέχουν γραφικά ή άλλα στοιχεία τα οποία μπορούν να διαχειριστεί ο χρήστης με interactive τρόπο(αλληλεπίδραση). Η Visual Basic, Visual C++ και άλλα τέτοια εργαλεία δεν είναι γλώσσες οπτικού προγραμματισμού (παρόλο το όνομά τους), αλλά γλώσσες κειμένου που απλά χρησιμοποιούν ένα γραφικό GUI για βοήθεια .Μερικές γλώσσες οπτικού προγραμματισμού είναι οι LabVIEW, Prograph, Pict, Tinkertoy, Fabrik, CODE 2.0., και Hyperpascal.

2.5.4 Το συστατικό ADO

Ο όρος ADO είναι το ακρωνύμιο του όρου ActiveX Database Objects. Η Microsoft αντικατέστησε το πρότυπο ODBC (Open Database Connectivity) και προγενέστερες τεχνολογίες προσπέλασης δεδομένων όπως οι DAO και RDO με το ADO.

Το ADO είναι ένα βασιζόμενο στο μοντέλο COM(Component Object Model) σύστημα επικοινωνίας εφαρμογών (Application Programming Interface API) το οποίο αντικαθιστά το βασιζόμενο στη γλώσσα C API του ODBC και διευκολύνει την χρήση αντικειμενοστραφών τεχνικών με βάσεις δεδομένων .

Τα συστατικά ADO κρύβουν μεγάλο μέρος της πολυπλοκότητας που συνεπάγεται η ενασχόληση με τις βασιζόμενες στο μοντέλο COM απόψεις του ADO πίσω από μια διεπιφάνεια παρόμοια με αυτή των κανονικών συστατικών βάσεων δεδομένων της βιβλιοθήκης VCL της Borland. Επιπρόσθετα επιτρέπουν την χρησιμοποίηση δεδομένων από το ADO σε κανονικούς ,ενήμερους ως προς τα δεδομένα μηχανισμούς της βιβλιοθήκης του VCL όπως τα πλέγματα εμφάνισης δεδομένων, πεδία επεξεργασίας κειμένου και γραφήματα. Αυτό είναι εφικτό επειδή τα συστατικά ADO είναι απόγονοι της κλάσης TDataSet και επειδή το συστατικό TDataSource μπορεί να δουλέψει με οποιοδήποτε συστατικό –απόγονο του TDataSet

Το ADO μπορεί να συνεργάζεται με οποιαδήποτε βάση δεδομένων για την οποία υπάρχει πρόγραμμα οδήγησης ODBC. Επιπρόσθετα το ADO παρέχει υποστήριξη για δεδομένα αποθηκευμένα σε μη σχεσιακές μορφές όπως η XML (διάδοχος της HTML)εφόσον υπάρχει ένας παροχέας δεδομένων (data provider) ο οποίος συμμορφώνεται με τις προδιαγραφές του ADO. Στη θεωρεία μπορείτε να εκτελέσετε εντολές SQL σε οποιοδήποτε σύνολο δεδομένων ADO, αλλά μπορεί να απαιτούνται ειδικές μορφές της SQL για την προσπέραση μη σχεσιακών δεδομένων .

Το ADO είναι μια ολοκληρωμένη τεχνολογία, ικανή να παρέχει πλήρη πρόσβαση σε σχεδόν κάθε βάση δεδομένων που χρησιμοποιείται σήμερα.

2.5.5 Πλεονεκτήματα των συστατικών ADO :

- Μεγάλο μέρος του λογισμικού που απαιτείται για την υποστήριξη τους παρέχεται μαζί με το λειτουργικό σύστημα, οπότε δεν χρειάζεται να κάνει κάτι ειδικό ο χρήστης για την εγκατάσταση .
- Λόγω του ότι η εξέλιξη των προγραμμάτων οδήγησης για το BDE είναι βραδύτερη απ' ότι στις προηγούμενες εκδόσεις (αν και το BDE μπορεί να χρησιμοποιεί νεότερα προγράμματα οδήγησης ODBC και αυτά θα συνεχίσουν να συμβαδίζουν με την εξέλιξη των συστημάτων DBMS για αρκετό χρόνο),το ADO μπορεί να είναι το μοναδικό σας εισιτήριο για την προσπέλαση ασυνήθιστων ή προηγμένων τεχνολογιών δεδομένων όπως η XML.
- Τα συστατικά ADO μπορούν να διευκολύνουν την μετάβαση από εργαλεία της Microsoft όπως η Visual C++ στο C++ Builder.
- Τα συστατικά TADOQuery είναι πάντα επεξεργάσιμα ,χωρίς να καταφεύγουν σε αποθηκευμένες ενημερώσεις, UpdateSQL, ή στις πολύπλοκες συνθήκες που καθιστούν επιτυχή μια αίτηση για ανάκτηση δεδομένων (RequestLive). Αξίζει να σημειωθεί το γεγονός ότι είναι διαθέσιμη μια μορφή αποθηκευμένων ενημερώσεων (αποκαλείται batch update-μαζική ενημέρωση- για συστατικά ADO).
- Τα συστατικά ADO επιτρέπουν την ασύγχρονη εκτέλεση κώδικα SQL και την παρακολούθηση της προόδου εκτέλεσης εντολών μέσω χειριστών συμβάντων. Αυτό μπορεί επίσης να χρησιμοποιηθεί για να παρέχει τον εξαιρετικά χρήσιμο μηχανισμό εμφάνισης προόδου, ο οποίος δείχνει το ποσοστό ολοκλήρωσης της εκτέλεσης ενός ερωτήματος ή μιας εντολής .
- Ανόμοια με τα συστατικά BDE, είναι σχετικά ασφαλές να τερματιστεί ένα πρόγραμμα βάσεων δεδομένων χρησιμοποιώντας την Program Reset(πράγμα το οποίο συνήθως προκαλεί ένα μήνυμα σφάλματος out of memory από το BDE κατά την επόμενη εκτέλεση). Ωστόσο ,όταν δεν είναι ασφαλές, συνήθως ο τερματισμός του προγράμματος προκαλεί την κατάρρευση του συστήματος .
- Τέλος ,αν και είναι καλύτερα να κρύβονται όσο το δυνατόν περισσότερο τα ιδιαίτερα χαρακτηριστικά των συστατικών ADO από τα προγράμματα, η εξοικείωση με τις βασικές έννοιες και αρχές του ADO μπορεί να αποδειχτεί

χρήσιμη μελλοντικά π.χ. σε έργα των οποίων η διαχείριση υπαγορεύει την χρήση του ADO .

2.5.6 Τα σημαντικότερα χαρακτηριστικά του ADO Express στη C++

TADOConnection: Υλοποιεί την σύνδεση με μια βάση δεδομένων και διαχειρίζεται τις συναλλαγές.

TRDSConnection: Υλοποιεί την σύνδεση με μία βάση δεδομένων με τρόπο που επιτρέπει την πρόσβαση πολλαπλών βαθμίδων σε ένα σύνολο εγγραφών .Χρησιμοποιήσιμο μόνο με το TADODataSet υποστηρίζει το βοήθημα Microsoft Remote Data Space το οποίο επιτρέπει το πέρασμα ενός συνόλου εγγραφών ADO μεταξύ των βαθμίδων μιας εφαρμογής βάσεων δεδομένων πολλαπλών βαθμίδων .
TADOTable: Επιτρέπει την προσπέλαση ενός πίνακα μιας βάσης δεδομένων βάσει του ονόματός του.

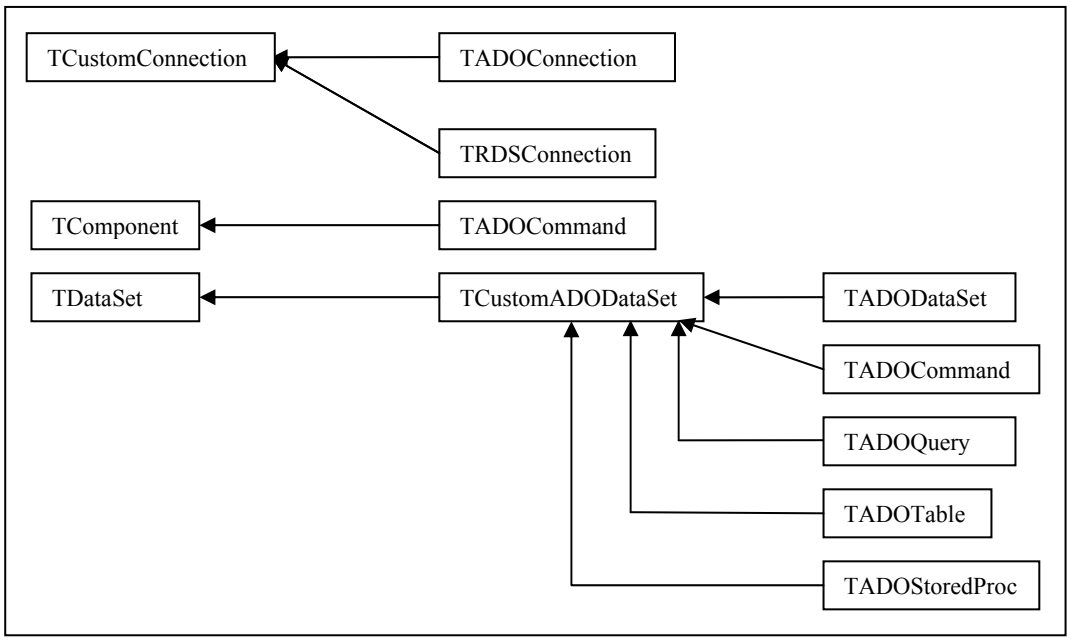
TADOQuery: Προσπελάζει έναν ή περισσότερους πίνακες μιας βάσης δεδομένων χρησιμοποιώντας εντολές SQL .Μπορεί να εκτελεί οποιαδήποτε εντολή SQL.

TADOStoredProc: Εκτελεί μια αποθηκευμένη διαδικασία από μία συγκεκριμένη βάση δεδομένων.

TADODataSet: Ουσιαστικά είναι ίδιο με το TADOQuery εκτός από το γεγονός ότι κατά την εκτέλεση εντολών SQL δεν επιστρέφει ένα σύνολο εγγραφών σαν αποτέλεσμα.

TADOCommand: Ουσιαστικά είναι ίσιο με το TADOQuery. Επιτρέπει την εκτέλεση οποιασδήποτε εντολής SQL αλλά για να προσπελάσετε το σύνολο εγγραφών του αποτελέσματος θα πρέπει να παρέχετε ένα ξεχωριστό αντικείμενο TADODataSet το οποίο θα συνδεθεί με αυτό το σύνολο εγγραφών .

Το δέντρο κληρονομικότητας των συστατικών ADO έχει τις ρίζες του στο συστατικό TDataSet. Ωστόσο, ο άμεσος απόγονος του TDataSet δεν είναι το TADODataSet αλλά η ιεραρχία των συστατικών (η θέση τους δηλαδή στην βιβλιοθήκη VCL)είναι η ακόλουθη :



Εικόνα 10: Η ιεραρχία κλάσεων για τα συστατικά ADO

Το πρωτόκολλο που χρησιμοποιεί το ADO είναι το COM το οποίο είναι ένα πρωτόκολλο επικοινωνίας γνωστό και ως ActiveX κατασκευασμένο από την Microsoft και χρησιμοποιείται για επικοινωνία μεταξύ εφαρμογών. Το ADO χρησιμοποιήθηκε για την επικοινωνία της εφαρμογής με τη βάση δεδομένων.

2.5.7 Η γλώσσα SQL

Η SQL (Structured Query Language - Δομημένη Γλώσσα Ερωτημάτων) είναι μια σχεσιακή γλώσσα Βάσεων δεδομένων η οποία σχεδιάστηκε και υλοποιήθηκε από την IBM στα πλαίσια του ερευνητικού προγράμματος System R που απέβλεπε στην ανάπτυξη ενός προτύπου συστήματος βάσεων δεδομένων (ΣΔΒΔ). Τα δυο πρώτα ΣΔΒΔ που διατέθηκαν ήταν το ORACLE από την Relational Software Inc.(σήμερα ORACLE Corporation) το 1979 και το SQL/DS από την IBM το 1982.

Για να μπορέσουμε να δημιουργήσουμε και να διαχειριστούμε μια βάση δεδομένων, μπορούμε να χρησιμοποιήσουμε ειδικές γλώσσες προγραμματισμού, τις λεγόμενες γλώσσες ερωτημάτων (query languages). Είναι γλώσσες μη διαδικαστικές, τέταρτης γενιάς (4^{th} generation languages). Εμείς απλά διατυπώνουμε με απλές και κατανοητές εντολές το τι πληροφορίες ζητάμε και το ΣΔΒΔ (Σύστημα Διαχείρισης Βάσεων Δεδομένων) αναλαμβάνει να μας απαντήσει. Σήμερα η πιο δημοφιλής και

πιο διαδεδομένη γλώσσα ανάπτυξης και διαχείρισης σχεσιακών βάσεων δεδομένων είναι η SQL.

- Η SQL μάς δίνει τη δυνατότητα να έχουμε πρόσβαση σε μια βάση δεδομένων (database).
- Η SQL μπορεί να εκτελέσει ερωτήματα (queries) σχετικά με μια βάση δεδομένων.
- Η SQL μπορεί να ανακτήσει δεδομένα από μια βάση δεδομένων.
- Η SQL μπορεί να εισαγάγει νέες εγγραφές σε μια βάση δεδομένων.
- Η SQL μπορεί να διαγράψει εγγραφές από μια βάση δεδομένων.
- Η SQL μπορεί να ενημερώσει εγγραφές σε μια βάση δεδομένων.
- Η SQL είναι εύκολη στην εκμάθηση.

Η SQL αποτελείται από εντολές με τα ορίσματα τους, τις οποίες μπορούμε να χρησιμοποιήσουμε με συγκεκριμένους κανόνες σύνταξης για να πάρουμε τα αποτελέσματα που θέλουμε. Με την SQL μπορούμε να δημιουργήσουμε μια βάση δεδομένων και τους πίνακές της με τα αντίστοιχα πεδία, να καταχωρήσουμε δεδομένα στους πίνακες, να τροποποιήσουμε και να διαγράψουμε τα δεδομένα αυτά, να αλλάξουμε τη δομή των πινάκων με προσθήκη και διαγραφή πεδίων και να εμφανίσουμε πληροφορίες (συνδυασμούς από δεδομένα).

Η SQL έχει δύο βασικά τμήματα :

- Τη Γλώσσα Ορισμού Δεδομένων (Data Definition Language - DDL), η οποία περιέχει τις απαραίτητες εντολές για τον ορισμό και την τροποποίηση του σχεσιακού σχήματος καθώς και για τη δημιουργία, την τροποποίηση και τη διαγραφή σχέσεων. Περιέχει ακόμη τις εντολές δημιουργίας και επεξεργασίας όψεων και ορισμού περιορισμών ακεραιότητας.
- Τη Γλώσσα Χειρισμού Δεδομένων (Data Manipulation Language - DML), η οποία περιέχει τις απαραίτητες εντολές για την εμφάνιση (αναζήτηση) δεδομένων καθώς και για την καταχώρηση, τροποποίηση και διαγραφή των εγγραφών (πλειάδων) μιας σχέσης.
- Τέλος, περιέχει εντολές για τον ορισμό και την επεξεργασία συναλλαγών (transactions) και εντολές για την ασφάλεια (authentication).

Τα κύρια πλεονεκτήματα της SQL είναι :

- **Ανεξαρτησία κατασκευαστή**. Αλλάζοντας ΣΔΒΔ συνήθως δεν χρειάζεται να ξαναγραφτεί ο κώδικας SQL
- **Υψηλού επιπέδου γλωσσική δομή** (English Like), που κάνει εύκολη τη χρήση της γλώσσας.
- **Τυποποίηση.** Η τυποποίηση επικυρώθηκε από το ANSI και το ISO.
- **Δυναμική διαχείριση δεδομένων.** Προσφέρει προχωρημένες εντολές επεξεργασίας που επιτρέπουν την ενσωμάτωση της καθώς και τα ερωτήματα πολλών επιπέδων.

2.5.8 Χειρισμός δεδομένων της γλώσσας SQL (Data Manipulation)

Όπως υπονοεί και το όνομά της, η SQL είναι μια σύνταξη για την εκτέλεση ερωτημάτων (queries). Αλλά η γλώσσα της SQL περιλαμβάνει επίσης μια σύνταξη για την ενημέρωση εγγραφών, την εισαγωγή νέων εγγραφών και τη διαγραφή υπαρχόντων εγγραφών.

Αυτές οι εντολές ερωτημάτων και ενημέρωσης αποτελούν μαζί τη Γλώσσα Χειρισμού Δεδομένων (Data Manipulation Language, DML) που αποτελεί κομμάτι της SQL :

- SELECT

- εξάγει δεδομένα από μια βάση δεδομένων.

- UPDATE

- ενημερώνει δεδομένα σε μια βάση δεδομένων.

- DELETE

- διαγράφει δεδομένα από μια βάση δεδομένων.

- INSERT

- εισάγει νέα δεδομένα σε μια βάση δεδομένων.

Η εντολή SELECT :

Η εντολή SELECT επιλέγει στήλες (columns) δεδομένων από μια βάση δεδομένων. Το αποτέλεσμα αποθηκεύεται σε μορφή πίνακα και αποκαλείται result set. Την χρησιμοποιούμε για να εμφανίζουμε (επιλέγουμε) δεδομένα από έναν πίνακα δεδομένων. Συντάσσεται ως εξής :

SELECT ονόματα_στηλών FROM όνομα_πίνακα

Η συνθήκη WHERE :

Το WHERE χρησιμοποιείται για να καθορίσουμε ένα κριτήριο επιλογής (selection criteria). Για να μπορέσουμε να επιλέξουμε δεδομένα υπό συνθήκη από έναν πίνακα, πρέπει να προσθέσουμε ένα where σε μια εντολή select, ως εξής :

SELECT στήλη FROM πίνακα WHERE στήλη συνθήκη τιμή.

Η συνθήκη LIKE *:*

Η συνθήκη LIKE χρησιμοποιείται για να καθορίσουμε μια αναζήτηση για ένα υπόδειγμα (pattern) σε μια στήλη.
Συντάσσεται ως εξής :

SELECT στήλη FROM πίνακα WHERE στήλη LIKE υπόδειγμα

Μπορούμε να χρησιμοποιήσουμε το σύμβολο "%" για να ορίσουμε χαρακτήρες μπαλαντέρ (wildcards) πριν και μετά από το υπόδειγμα.

Οι λογικοί τελεστές AND *και* OR *:*

Τα AND και OR ενώνουν δύο ή περισσότερες συνθήκες σ' ένα WHERE. Ο τελεστής AND εμφανίζει μια γραμμή αν όλες οι συνθήκες είναι αληθείς (true), ενώ ο τελεστής OR εμφανίζει μια γραμμή αν οποιαδήποτε από τις συνθήκες είναι αληθής (true).

Η εντολή INSERT INTO :

Η εντολή INSERT INTO εισάγει νέες γραμμές σ' έναν πίνακα. Συντάσσεται ως εξής :

INSERT INTO όνομα_πίνακα
VALUES (τιμή1, τιμή2, ...)

Μπορούμε επίσης να καθορίσουμε τις στήλες για τις οποίες θέλουμε να εισάγουμε δεδομένα:

INSERT INTO όνομα_πίνακα(στήλη1, στήλη2, ...)
VALUES (τμή1, τιμή2, ...)

Η εντολή UPDATE *:*

Η εντολή UPDATE ενημερώνει ή αλλάζει γραμμές. Συντάσσεται ως εξής :

UPDATE όνομα_πίνακα SET όνομα_στήλης = νέα_τιμή
WHERE όνομα_στήλης = τιμή

Η εντολή DELETE :

Η εντολή DELETE χρησιμοποιείται για να διαγράψουμε γραμμές από έναν πίνακα. Συντάσσεται ως εξής :

DELETE FROM όνομα_πίνακα
WHERE όνομα_στήλης = τιμή

2.5 Προγράμματα που χρησιμοποιήθηκαν

Στο σημείο αυτό θα αναφέρουμε τα προγράμματα που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής και θα δώσουμε μια περιληπτική περιγραφή για το καθένα από αυτά.

C++ Builder

Ο C++ Builder είναι το προγραμματιστικό περιβάλλον που δημιουργήθηκε από την Borland για τη ταχεία ανάπτυξη εφαρμογών στη γλώσσα προγραμματισμού C++. Η δομή του έχει πολύ στενή σχέση με τη Delphi και γι' αυτό πολύ πιστεύουν ότι είναι μια έκδοση της Delphi για C++. Πολλά components που αναπτύσσονται στη Delphi μπορούν να λειτουργήσουν στο C++Builder αν και το αντίθετο δεν είναι πάντα σίγουρο. Περιέχει μια τεράστια ποικιλία από εργαλεία που επιτρέπουν σχεδόν drag-and-drop οπτικό προγραμματισμό κάνοντας την δουλεία των προγραμματιστών πολύ πιο εύκολη. Άλλα δημοφιλή προγράμματα για την ανάπτυξη εφαρμογών είναι τα Delphi και JBuilder από την Borland και η Visual Basic από την Microsoft. Ο C++Builder είχε αρχικά δημιουργηθεί για τη πλατφόρμα των Microsoft Windows, όμως μετέπειτα εκδόσεις ενσωματώνουν την Borland CLX μια cross-platform βιβλιοθήκη οπτικών αντικειμένων που υποστηρίζει Windows και Linux. Το προγραμματιστικό περιβάλλον αυτό χρησιμοποιήθηκε για την δημιουργία της εφαρμογής μας.

Microsoft SQL Server

Ο Microsoft SQL Server είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων. Υποστηρίζει SQL, την ποιο διαδεδομένη γλώσσα βάσεων δεδομένων. Χρησιμοποιεί μια παραλλαγή της SQL την T-SQL (Transact –SQL, βασισμένη στην SQL-92). Ιστορικά να αναφέρουμε ότι η βάση του κώδικα του Sql server αρχικά προήλθε από το Sybase Sql Server και αποτελούσε την είσοδος της Microsoft στο χώρο των βάσεων δεδομένων απέναντι στην Oracle, IBM και Sybase. Η πρώτη έκδοσή της προοριζόταν για το OS/2 (1993) ήταν ουσιαστικά ο ίδιος ο Sybase Sql Server. Αργότερα με την έλευση των Windows NT οι δυο εταιρείες έπαψαν την συνεργασία και επιδίωξαν δικά τους σχεδιαστικά και marketing μονοπάτια. Την παρούσα στιγμή ο Microsoft SQL Server αποτελεί ένα από τα πιο ανταγωνιστικά προϊόντα στο χώρο. Το πρόγραμμα αυτό χρησιμοποιήθηκε για την δημιουργία και διαχείριση της βάσης δεδομένων της εφαρμογής.

MATLAB

Το Matlab εφευρέθηκε στα τέλη του 1970 απο τον Cleve Moler. Το Matlab είναι ένα περιβάλλον που προσφέρει δυνατότητες για μαθηματικούς υπολογισμούς και μία υψηλού επιπέδου προγραμματιστική γλώσσα. Δημιουργήθηκε από την MathWorks, το MATLAB επιτρέπει την ανάλυση δεδομένων, τις γραφικές απεικονίσεις, την ανάπτυξη αλγορίθμων, την εξομοίωση και μοντελοποίηση συστημάτων, την δημιουργία interface και την συνεργασία με άλλα προγράμματα. Χρησιμοποιείται από περισσότερους του ενός εκατομμύρια ανθρώπους στη βιομηχανία και τον ακαδημαϊκό κόσμο.

PHOTOSHOP

Το πρόγραμμα PhotoShop αναπτύχθηκε και εκδόθηκε από την εταιρεία Adobe Systems. Πρόκειται για το ισχυρότερο και δημοφιλέστερο πρόγραμμα επεξεργασίας εικόνας. με το οποίο μπορούμε να επεξεργαστούμε φωτογραφίες και εικόνες. Το πρόγραμμα PhotoShop περιέχει δύο βασικές ομάδες εργαλείων, μια για σχεδίαση και μια για επεξεργασία εικόνας. Η τελευταία έκδοση είναι η Adobe Photoshop CS2 (9.0.1) και κυκλοφόρησε το 2006. Το πρόγραμμα αυτό χρησιμοποιήθηκε για την επεξεργασία και δημιουργία των εικόνων που χρησιμοποιήθηκαν τόσο στην δημιουργία του interface της εφαρμογής όσο και στην δημιουργία της πτυχιακής αυτής εργασίας.

ΚΕΦΑΛΑΙΟ 3

ΟΙ ΑΛΓΟΡΙΘΜΟΙ

Στο κεφάλαιο αυτό, αρχικά θα αναπτύξουμε την έννοια του αλγόριθμου και στην συνέχεια θα προχωρήσουμε στην μελέτη των αλγόριθμων ταξινόμησης Binary Search-Tree , Quad-Tree και R-Tree. Ειδικότερα, θα εξηγήσουμε την λειτουργία τους, θα παρουσιάσουμε τις πιο σημαντικές παραλλαγές τους και θα αναφέρουμε τις πιο διαδεδομένες χρήσεις τους. Τέλος, θα αναφέρουμε κάποια εξωτερικά συμπεράσματα για την απόδοση των αλγορίθμων που παρουσιάζουμε.

3.1. Η έννοια του αλγόριθμου

Η έννοια του αλγόριθμου είναι κεντρική για την πληροφορική και η μελέτη της είναι πολύ ενδιαφέρουσα γιατί αποτελεί την πρώτη ύλη για την εμβάθυνση, αν όχι σε όλες, τουλάχιστον σε πολλές γνωστικές περιοχές της Πληροφορικής, όπως στις βάσεις δεδομένων, τα δίκτυα, την επεξεργασία εικόνας, την τεχνητή νοημοσύνη, την κρυπτογραφία και αλλού. Εξ άλλου είναι γνωστή η εξίσωση:

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Η εξίσωση αυτή διατυπώθηκε το 1976 από τον Niklaus Wirth (που παρουσίασε την πρώτη δομημένη γλώσσα προγραμματισμού, την Pascal) και δηλώνει ότι οι αλγόριθμοι συνυφασμένοι με τις απαραίτητες δομές σε μία αδιάσπαστη ενότητα, αποτελούν τη βάση κάθε προγράμματος, δηλαδή τελικά κάθε εφαρμογής.

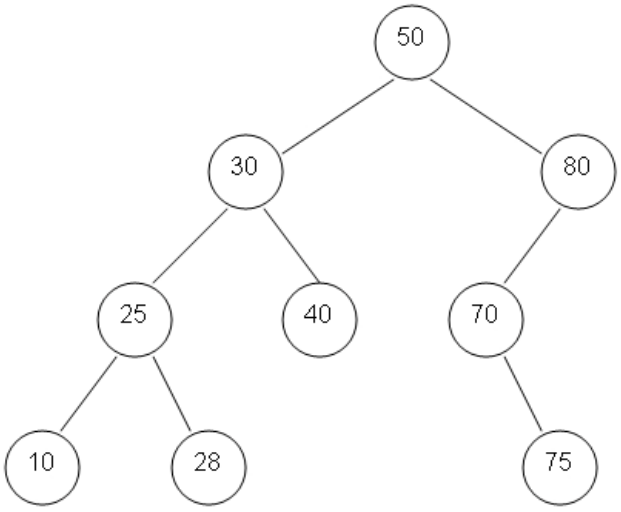
Αλγόριθμος (*algorithm*) είναι μια διαδικασία που επεξεργαζόμενη ορισμένα στοιχεία με *πεπερασμένο* αριθμό βημάτων παράγει ένα συγκεκριμένο επιθυμητό αποτέλεσμα. Κάθε βήμα της διαδικασίας αποτελείται από *μονοσήμαντα εκτελέσιμες* πράξεις. Ουσιαστικά είναι η περιγραφή μιας αυστηρά καθορισμένης σειράς βημάτων για την λύση ενός προβλήματος. Κάθε αλγόριθμος έχει κανένα, ένα ή περισσότερα δεδομένα εισόδου και ένα τουλάχιστον αποτέλεσμα. Οι αλγόριθμοι διατυπώνονται συνήθως σε μια γλώσσα που μπορεί να περιγράψει τη σειρά των βημάτων με σαφή και μαθηματικά ξεκάθαρο τρόπο, χωρίς ασάφειες. Τέτοιες γλώσσες είναι συνήθως οι γλώσσες προγραμματισμού, κάποια συμβολική γλώσσα, αλλά και οι φυσικές γλώσσες καμιά φορά.

3.2 Το Binary Search – Tree

3.2.1 Παρουσίαση της λειτουργίας του BS - Tree

Στα τέλη του 1960 κατασκευαστές συστημάτων υπολογιστών καθώς και ανεξάρτητες ερευνητικές ομάδες ανέπτυξαν ανταγωνιστικά συστήματα αρχείων (file-systems) και τα αποκάλεσαν "μεθόδους προσπέλασης - access methods". Στην εταιρία Sperry Univac (σύμπραξη με το πανεπιστήμιο Case Western Reserve) οι H. Chiat, M. Schwartz, και άλλοι ανέπτυξαν και υλοποίησαν ένα σύστημα το οποίο εκτελούσε εργασίες εισόδου και εύρεσης με τρόπο όμοιο της μεθόδου Binary Search – Tree, την οποία θα περιγράψουμε παρακάτω.

Τα δυαδικά δένδρα αναζήτησης (Binary Search – Tree) αποτελούν ειδική κατηγορία των δυαδικών δένδρων. Είναι *διατεταγμένα*, δηλαδή δένδρα στα οποία η διάταξη των παιδιών κάθε κόμβου έχει σημασία. Στα δυαδικά δένδρα αναζήτησης κάθε κόμβος έχει μεγαλύτερη τιμή από όλους τους κόμβους που βρίσκονται στα αριστερά του και μικρότερη από την τιμή όλων των κόμβων δεξιά του. Το δένδρο της **εικόνας 11** είναι ένα δυαδικό δένδρο αναζήτησης:



Εικόνα 11: Binary Search – Tree

Ένα μη κενό δυαδικό δένδρο αναζήτησης ικανοποιεί τις εξής ιδιότητες:

1. Κάθε στοιχείο (κόμβος) έχει μια τιμή και δεν υπάρχουν δυο στοιχεία με την ίδια τιμή. Κατά συνέπεια όλα τα στοιχεία είναι διακριτά.
2. Οι κόμβοι (αν υπάρχουν) στο αριστερό υποδένδρο της ρίζας είναι μικρότεροι από τη τιμή της ρίζας.
3. Οι κόμβοι (αν υπάρχουν) στο δεξιό υποδένδρο της ρίζας είναι μεγαλύτεροι από τη τιμή της ρίζας.
4. Το αριστερό και το δεξιό υποδένδρο είναι επίσης δυαδικά δένδρα αναζήτησης.

Η υποκείμενη δομή δεδομένων μας επιτρέπει να αναπτύξουμε αλγόριθμους με υψηλές επιδόσεις κατά την εκτέλεση των λειτουργιών αναζήτησης, εισαγωγής, επιλογής, και ταξινόμησης μιας συλλογής δεδομένων, στη μέση περίπτωση. Αποτελεί την καλύτερη μέθοδο για πολλές εφαρμογές και θέτει υποψηφιότητα ως ένας από τους πιο θεμελιώδεις αλγόριθμους της επιστήμης των υπολογιστών.

3.2.2 Παρουσίαση της διαδικασίας εισαγωγής/διαγραφής/αναζήτησης

Αναζήτηση κόμβου:

Ο αλγόριθμος αναζήτησης σε δυαδικό δένδρο αναζήτησης, με ρίζα το Β, αποσκοπεί στον εντοπισμό του κόμβου που έχει μια δεδομένη τιμή Χ (value).

Ξεκινώντας από τη ρίζα σε κάθε κόμβο γίνεται σύγκριση της τιμής Χ με το περιεχόμενο του κόμβου. Αν βρεθούν ίσα η αναζήτηση σταματά. Αν η τιμή είναι μικρότερη της ρίζας η αναζήτηση συνεχίζεται στο αριστερό υποδένδρο αλλιώς συνεχίζεται στο δεξιό υποδένδρο. Ο αλγόριθμος επίσης σταματά αν φθάσουμε σε κάποιο τερματικό κόμβο (φύλλο) χωρίς να βρεθεί η τιμή Χ σε κάποιο κόμβο του δένδρο.

Αλγόριθμος Αναζήτησης Κόμβου Search_Node (ψευδικώδικας)

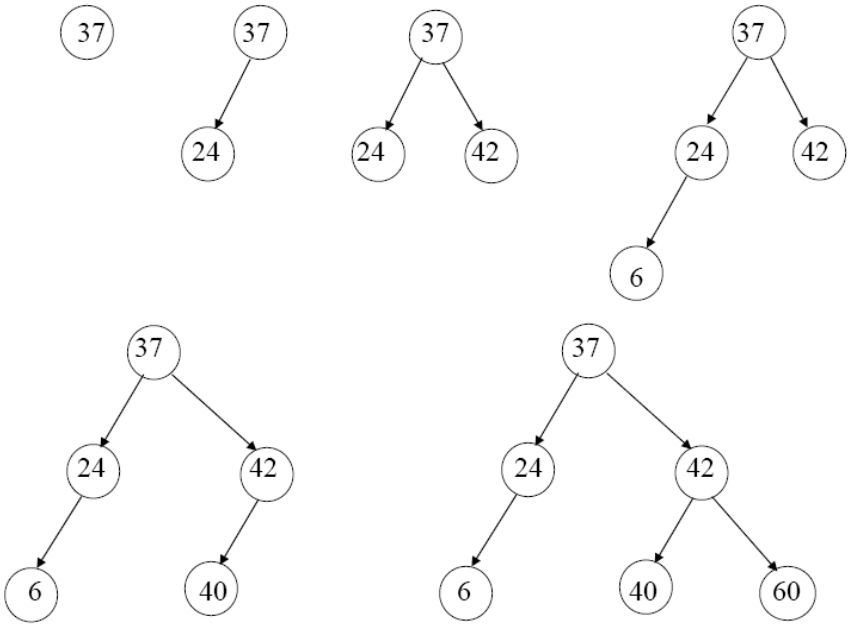
// Δεδομένα Root: ρίζα δένδρου, Χ: ζητούμενος κόμβος, Ω : λογική μεταβλητή, Β : βοηθητική μεταβλητή

```
Ξεκίνα
        Var B := Root, Ω:= false
        Όσο B <> τιμή X και Ω = Ψευδής Κάνε
        Ξεκίνα
                Εαν B = X τότε
                        Ω = true
                Διαφορετικά
                        Εαν X>B Τοτε
                                B=(Δεξιό παιδί του B)
                        Διαφορετικά
                                B=(Αριστερό παιδί του B)
                        Τέλος Εαν
                Τέλος Εαν
        Τέλος
Τέλος
```

Εισαγωγή κόμβου:

Ο αλγόριθμος με βάση τον οποίο εισάγουμε μια τιμή σε δυαδικό δένδρο αναζήτησης είναι ο εξής: Για την εισαγωγή ενός στοιχείου X σε ένα δυαδικό δένδρο αναζήτησης, πρέπει πρώτα να επιβεβαιώσουμε ότι το στοιχείο είναι διαφορετικό από τα ήδη υπάρχοντα στοιχεία, εκτελώντας μια αναζήτηση. Αν η αναζήτηση είναι ανεπιτυχής τότε το στοιχείο εισάγεται στο σημείο όπου τερματίστηκε η αναζήτηση.

Στην **εικόνα 12** βλέπουμε το παράδειγμα δημιουργίας ενός δυαδικού δένδρου αναζήτησης με διαδοχικές εισαγωγές τιμών. Οι τιμές που εισάγονται είναι: 37, 24, 42, 6, 40, 60.

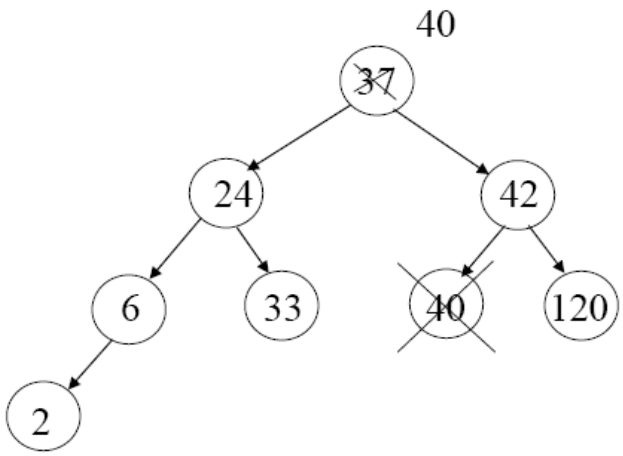


Εικόνα 12: Παράδειγμα δημιουργίας BS - Tree

Διαγραφή κόμβου:

Η διαδικασία της διαγραφής σε δυαδικό δένδρο αναζήτησης είναι πιο σύνθετη από τη διαδικασία της εισαγωγής. Αν ο κόμβος που πρόκειται να διαγραφεί είναι τερματικός (φύλλο), τότε η περίπτωση είναι εύκολη, απλά διαγράφουμε τον κόμβο. Σχετικά εύκολη είναι και η περίπτωση, όπου ο διαγραφόμενος κόμβος έχει μόνο έναν απόγονο, απλά αντικαθίσταται από το παιδί του. Η δυσκολία του προβλήματος παρουσιάζεται όταν το στοιχείο που θα διαγραφεί έχει δύο απογόνους. Στην περίπτωση αυτή ο διαγραφόμενος κόμβος αντικαθίσταται είτε από τον δεξιότερο κόμβο του αριστερού υποδένδρου είτε από τον αριστερότερο κόμβο του δεξιού υποδένδρου.

Στην **εικόνα 13** βλέπουμε το παράδειγμα αφαίρεσης του κόμβου με τιμή 37 από ένα δυαδικό δένδρο αναζήτησης:



Εικόνα 13: Παράδειγμα αφαίρεσης της τιμής 37

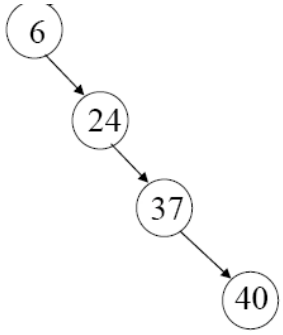
3.2.3 Διάσχιση του BS - Tree

Ουσιαστικά, ισχύει ότι ακριβώς και για το Binary - Tree. Υπάρχουν τρία είδη διάσχισης η προδιατεταγμένη (preorder), η μεταδιατεταγμένη (postorder), και η ενδοδιατεταγμένη (inorder). Είναι προφανές ότι αν διασχίσουμε το δένδρο με τον ενδοδιατεταγμένο τρόπο το αποτέλεσμα που θα προκύψει θα είναι ταξινομημένο, από το μικρότερο στο μεγαλύτερο. Φυσικά η ταξινόμηση δεν ισχύει για τη διάσχιση του δένδρου με κάποιο από τους άλλους τρόπους.

3.2.4 Χαρακτηριστικά επιδόσεων του BS - Tree

Οι χρόνοι εκτέλεσης των αλγορίθμων σε δένδρα δυαδικής αναζήτησης εξαρτώνται από το σχήμα των δένδρων. Το κόστος κάθε διαδικασίας είναι ανάλογο του βάθους (ύψους) του δένδρου. Στην καλύτερη περίπτωση, το δένδρο μπορεί να είναι πλήρως ισορροπημένο (ισοζυγισμένο) με περίπου $\log_2(N)$ κόμβους ανάμεσα στη ρίζα και κάθε εξωτερικό κόμβο, αλλά στη χειρότερη περίπτωση μπορεί να υπάρχουν N κόμβοι στη διαδρομή αναζήτησης. Θα μπορούσαμε να αναμένουμε από τους χρόνους εκτέλεσης να επίσης λογαριθμικοί στη μέση περίπτωση (υποθέτοντας πως όλα τα δεδομένα είναι εξίσου πιθανά) επειδή το πρώτο στοιχείο που εισάγεται γίνεται ρίζα του δένδρου – αν πρέπει να εισαχθούν N κλειδιά τυχαία αυτό το στοιχείο θα χωρίζει τα κλειδιά στη μέση(κατά μέσο όρο), κάτι που θα οδηγήσει σε λογαριθμικούς χρόνους αναζήτησης (με την ίδια λογική στα υποδένδρα). Αυτή η περίπτωση θα ήταν η καλύτερη γι' αυτόν τον αλγόριθμο με εγγυημένο λογαριθμικό χρόνο εκτέλεσης για όλες τις αναζητήσεις. Σε μια πραγματικά τυχαία κατάσταση, η ρίζα έχει την ίδια πιθανότητα να είναι οποιαδήποτε κλειδί, επομένως ένα τέτοιο τέλεια ισορροπημένο δένδρο είναι εξαιρετικά σπάνιο και δεν μπορούμε εύκολα να διατηρήσουμε το δένδρο τέλεια ισορροπημένο μετά από κάθε εισαγωγή. Πάντως, επειδή και τα δένδρα που είναι μη ισορροπημένα σε υψηλό βαθμό σπάνια εμφανίζονται για τυχαία κλειδιά, τα δένδρα μάλλον είναι καλά ισορροπημένα κατά μέσο όρο.

Στη χειρότερη περίπτωση (Worst case behavior) μια αναζήτηση σε δένδρο δυαδικής αναζήτησης με N κόμβους μπορεί να απαιτεί N συγκρίσεις. Αυτή η συμπεριφορά χειρότερης περίπτωσης δεν είναι απίθανη στην πράξη, εμφανίζεται όταν εισάγουμε τα στοιχεία με ταξινομημένη σειρά ή με αντίστροφη σειρά ταξινόμησης σε ένα αρχικά κενό δένδρο χρησιμοποιώντας τον καθιερωμένο αλγόριθμο. Στην χειρότερη περίπτωση, το δυαδικό δένδρο εκφυλίσετε σε μια μορφή η οποία είναι ισοδύναμη με αυτή της απλά συνδεδεμένης λίστας, κάτι που οδηγεί σε τετραγωνικό χρόνο κατασκευής του δένδρου και γραμμικό χρόνο αναζήτησης. Το δένδρο σε αυτή την περίπτωση ονομάζεται λοξό (Skewed – Tree), ένα παράδειγμα λοξού δένδρου φαίνεται στην **εικόνα 14** .



Εικόνα 14: Ένα Δυαδικό δένδρο αναζήτησης χειρότερης περίπτωσης (Skewed tree)

Το πρόβλημα της κακής επίδοσης στην χειρότερη περίπτωση του δυαδικού δένδρου αναζήτησης, εξαλείφεται αν έχουμε τέλεια ισορροπημένο δένδρο (ισοζυγισμένο).

3.2.5 Άλλες εκδοχές του BS - Tree (Variations)

AVL - Trees

Η ονομασία των δένδρων αυτών προέκυψε από τα ονόματα των δύο Ρώσων ερευνητών που τα πρότειναν: του Adelson - Velskii και του Landis το 1962. Η δομή αυτή συναντάται στη βιβλιογραφία και με την ονομασία **ισοζυγισμένα δένδρα κατά ύψος** (height balanced trees).

Ένα δυαδικό δένδρο αναζήτησης λέγεται δένδρο AVL αν για κάθε κόμβο του δένδρου ικανοποιούνται οι εξής δύο συνθήκες:

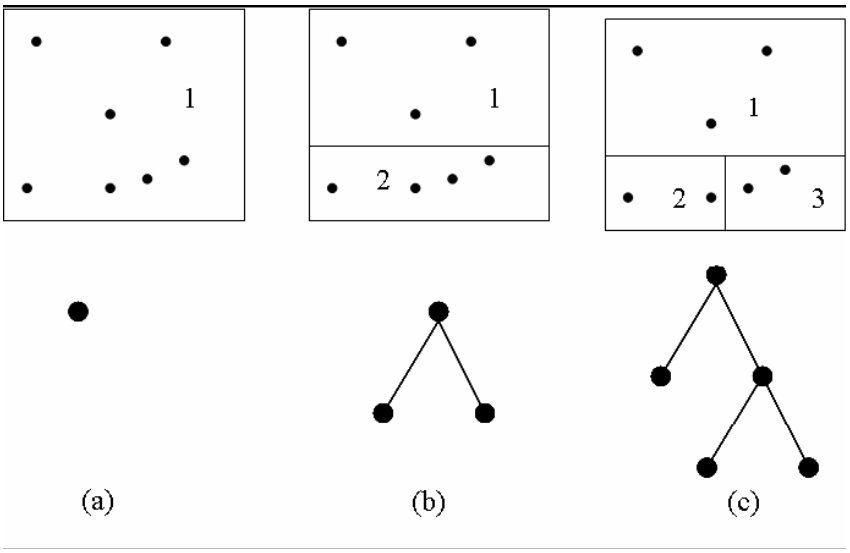
- Η διαφορά των υψών των δύο υποδένδρων του κόμβου είναι μικρότερη από ένα.
- Τα δύο υποδένδρα του κόμβου είναι επίσης δένδρα AVL.

Παρατηρούμε ότι ο ορισμός είναι αναδρομικός και παρέχει ευκαιρίες για ανάλυση. Η αναζήτηση στα δένδρα AVL γίνεται ακριβώς όπως και στα απλά δυαδικά δένδρα αναζήτησης. Η διαφορά των δύο δόμων έγκειται στους σύνθετους αλλά και κομψούς

αλγορίθμους εισαγωγής/διαγραφής κλειδιών στα δένδρα AVL, οι οποίοι εκτελούν περιστροφές (rotations), ώστε να μην παραβιάζεται ο ανωτέρω ορισμός.

K-D - Tree

Το K-D – Tree παρουσιάστηκε το 1975 από τον Bentley και ουσιαστικά είναι ένα πολυδιάστατο δυαδικό δένδρο αναζήτησης (multidimensional binary search tree) όπου K ο αριθμός των διαστάσεων (Dimensions) του διαστήματος αναζήτησης [Bent75]. Το K-D - Tree είναι μια δομή για αποθήκευση πολυδιάστατων σημείων στο χώρο. Διαιρεί τον χώρο σε δύο υπό-χώρους και αποθηκεύει τα σημεία του ενός υπό-χώρου στο αριστερό υποδένδρο και του άλλου στο δεξί. Στην **εικόνα 15** βλέπουμε το διαμοιρασμό ενός δυσδιάστατου χώρου με βάση το K-D - Tree. Αρχικά, ένας μοναδικός κόμβος, ο 1, υπάρχει στο σύστημα και αναλαμβάνει όλο το χώρο, αντιστοιχώντας σε έναν κόμβο. Όταν φτάνει ο επόμενος κόμβος, ο 2, ο χώρος διαμοιράζεται σε δυο μέρη με ίσο φορτίο ως προς την πρώτη διάσταση και κάθε ένας αναλαμβάνει από ένα τμήμα. Αυτό οδηγεί σε διάσπαση του μοναδικού κόμβου του δένδρου στα δύο. Καθώς νέοι κόμβοι εισέρχονται, η διαδικασία συνεχίζεται με ανάλογο τρόπο, με διάσπαση υπαρχόντων φύλλων του δένδρου και αντίστοιχων τμημάτων του χώρου δεδομένων. Σημειώνεται ότι κατά το διαμοιρασμό χρησιμοποιούνται κυκλικά οι διαστάσεις.



Εικόνα 15: Ο διαμοιρασμός του δυσδιάστατου χώρου

3.2.6 Χρήσεις του BS - Tree

Το Binary Search - Tree είναι μια πολύ χρήσιμη δομή δεδομένων. Η πιο συνήθης χρήση του Binary Search – Tree είναι για αποθήκευση και προσπέλαση δεδομένων (Information Storage and Retrieval) σε συστήματα αρχείων ή συστήματα βάσεων δεδομένων, στα οποία ενσωματώνετε. Συνήθως χρησιμοποιείται για την γρήγορη και αποδοτική αναζήτηση ενός στοιχείου σε μια συλλογή δεδομένων. Για παράδειγμα, αν είχαμε μια συνδεδεμένη λίστα από 1000 στοιχεία και θέλαμε να μάθουμε αν ένα συγκεκριμένο στοιχείο υπάρχει μέσα στην λίστα αυτό θα απαιτούσε να αναζητήσουμε σειριακά ένα προς ένα τα στοιχεία ξεκινώντας από την αρχή και προχωρώντας προς το τέλος της λίστας. Αν ήμασταν τυχεροί, ίσως το βρίσκαμε στην αρχή της λίστας. Εξίσου πιθανό θα ήταν το στοιχείο να βρίσκεται στο τέλος της λίστας, το οποίο θα σήμαινε ότι θα έπρεπε να ψάξουμε όλα τα στοιχεία πριν από αυτό. Αυτό μπορεί να μην φαντάζει ως μεγάλο πρόβλημα, ειδικά στις μέρες μας όπου οι υπολογιστές έχουν γίνει πολύ γρήγοροι, αλλά φανταστείτε αν η λίστα ήταν πολύ μεγαλύτερη και οι αναζητήσεις επαναλαμβάνονταν πολλές φορές. Αυτού του είδους οι αναζητήσεις συμβαίνουν συχνά σε web servers και τεράστιες βάσεις δεδομένων, το οποίο δημιουργεί την ανάγκη για πολύ γρήγορες τεχνικές αναζήτησης.

Τα δυαδικά δένδρα αναζήτησης βασίζονται στις αρχές τις περιοδικά επαναλαμβανόμενης διάσπασης (recursive decomposition) οι οποίες είναι παρόμοιες με τις μεθόδους διαίρει κα βασίλευε και μειώνουν τον χρόνο αναζήτησης σημαντικά κάνοντας τα αρκετές φορές πιο γρήγορα από οποιαδήποτε δομή σειριακής αναζήτησης.

3.3 Το Quad – Tree

3.3.1 Παρουσίαση της λειτουργίας του Quad - Tree

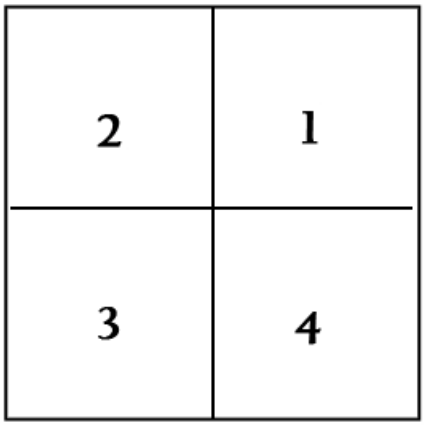
Τα Quad-Trees προτάθηκαν από τους R.Finkel και J.L.Bentley το 1974 [FB74]. Πρόκειται για μια από τις πιο έντονα μελετημένες δομές δεδομένων η οποία στηρίζεται στις αρχές της περιοδικά επαναλαμβανόμενης διάσπασης *divide and conquer* methods) [Aho et al. 1974]).

Στην απλούστερη του μορφή το Quad –Tree είναι όμοιο με το δυαδικό δένδρο, με εξαίρεση ότι κάθε κόμβος-γονέας έχει τέσσερις απογόνους (κάποιοι από τους οποίους μπορεί να είναι κενοί), για αυτό και ονομάζεται τετραδικό (Quad) δένδρο. Το Quad – Tree είναι μια μη ισορροπημένη (unbalanced) χωρική δομή δεδομένων η οποία διασπάει περιοδικά τον χώρο (δυσδιάστατο χώρο) σε τέσσερα ίσου μεγέθους ασυνάρτητα τεταρτημόρια, μέχρι κάθε κόμβος φύλλο να περιέχει ένα μόνο σημείο. Η περιοχή στην οποία διασπάτε ο χώρος μπορεί να είναι τετράγωνη, ορθογώνια ή να έχει αυθαίρετα σχήματα ανάλογα με την εκδοχή του Quad – Tree. Το Quad – Tree το συναντάμε συχνά στην βιβλιογραφία και ως Q - Tree. Υπάρχουν πολλές παραλλαγές του Quad – Tree, όλες όμως έχουν κάποια κοινά χαρακτηριστικά:

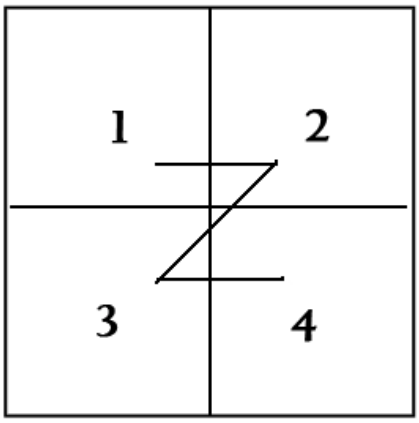
- Διασπούν τον χώρο σε προσαρμόσιμα κελιά
- Κάθε κελί έχει μια μέγιστη χωρητικότητα. Όταν το κελί φθάσει την μέγιστη χωρητικότητα του διασπάτε.
- Ο δενδρικός κατάλογος ακολουθεί τη χωρική διάσπαση του Quad – Tree.

3.3.2 Παρουσίασης της διαδικασίας δημιουργίας ενός Quad – Tree

Η πιο συνηθισμένη μέθοδος απόδοσης τιμών στα τεταρτημόρια ενός Quad – Tree είναι αυτή της **εικόνας 16** (c ordering) όπου οι αριθμοί παρατάσσονται με σειρά αντίθετη αυτής των δεικτών του ρολογιού. Άλλη γνωστή μέθοδος απόδοσης τιμών στα τεταρτημόρια ενός Quad Tree είναι της **εικόνας 17** (Z ordering) όπου οι αριθμοί παρατάσσονται έτσι ώστε να σχηματίζεται ένα νοητό Z.

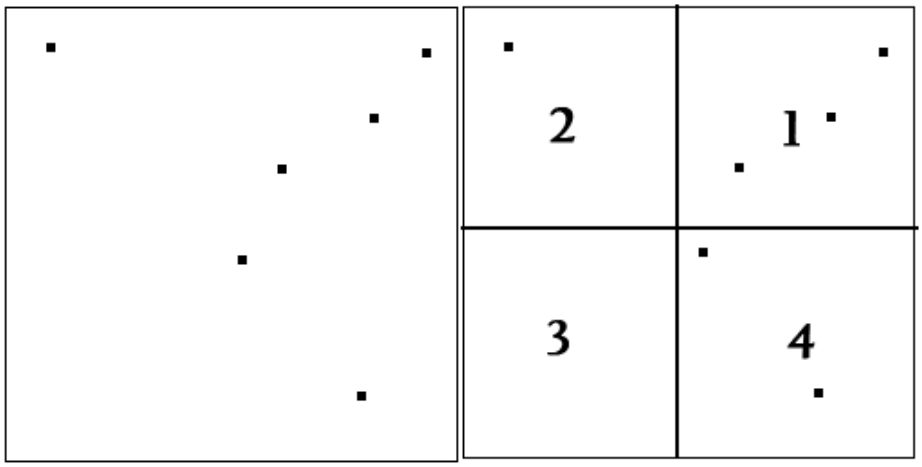


Εικόνα 16: C ordering



Εικόνα 17: Z ordering

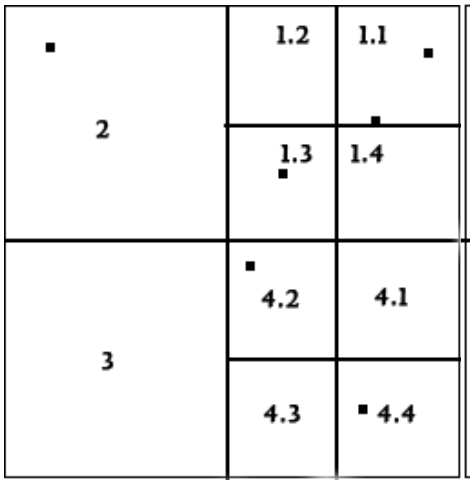
Στις παρακάτω εικόνες παρουσιάζεται το παράδειγμα της δημιουργίας ενός Quad – Tree. Στην πρώτη εικόνα φαίνονται τα σημεία που έχουν επιλεχθεί για την δημιουργία του Quad – Tree. Η πρώτη εικόνα αποτελεί την ρίζα του δένδρου. Στην συνέχεια χωρίζεται σε τεταρτημόρια όπου σε κάθε ένα από αυτά αντιστοιχεί ένας αριθμός (1, 2, 3, 4). Στα τεταρτημόρια που έχουμε περισσότερα του ενός σημεία η διαδικασία συνεχίζεται έως κάθε σημείο να ανήκει σε διαφορετικό τεταρτημόριο, **εικόνα 21**.



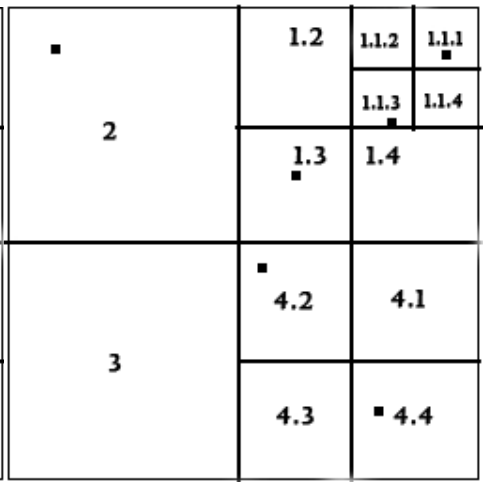
Εικόνα 18: Τα σημεία που επιλέχθηκαν

Εικόνα 19: Επίπεδο 1

Στην **εικόνα 19** παρατηρούμε ότι τα τεταρτημόρια 1 και 4 περιέχουν περισσότερα του ενός σημεία. Η διαδικασία συνεχίζεται, **εικόνα 20**, όπου τα τεταρτημόρια αυτά έχουν χωριστεί εκ νέου σε τέσσερα νέα αλλά το τεταρτημόριο 1.1 συνεχίζει να περιέχει περισσότερα του ενός σημεία. Για το λόγω αυτό η διαδικασία συνεχίζεται και ολοκληρώνεται στην **εικόνα 21** όπου κάθε σημείο ανήκει σε διαφορετικό τεταρτημόριο.

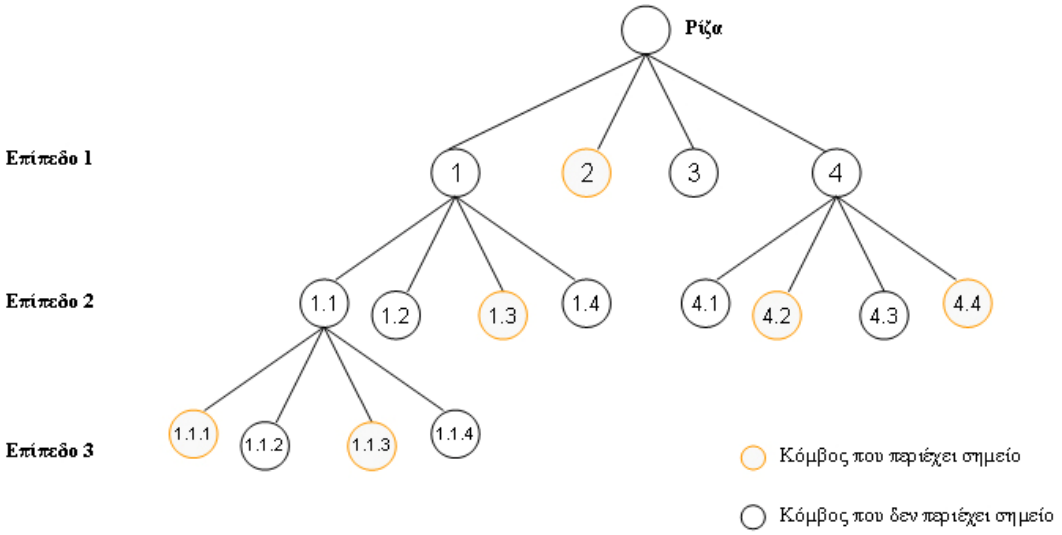


Εικόνα 20: Επίπεδο 2



Εικόνα 21: Επίπεδο 3

Το τετραδικό δένδρο (Quad - Tree) που σχηματίζεται από την εφαρμογή του αλγορίθμου στα σημεία της εικόνας 8, φαίνεται στην **εικόνα 22**.

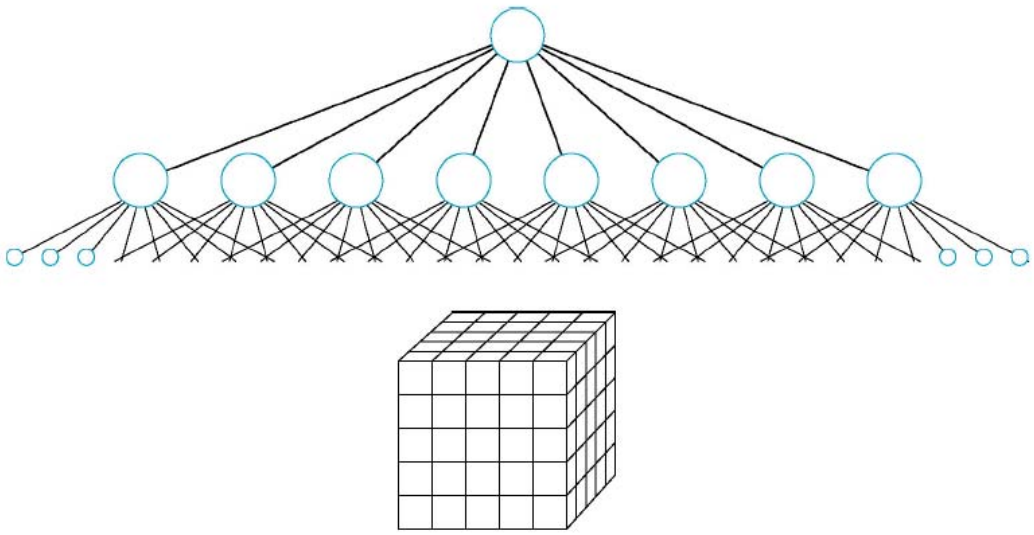


Εικόνα 22: Το Quad – Tree των σημείων της εικόνας 18

3.3.3 Άλλες εκδοχές του Quad-Tree (Variations)

Oct – Tree

Τα Quad – Tree μπορούν να επεκταθούν σε περισσότερες από δύο διαστάσεις χωρίς καμία διαφορά στον τρόπο λειτουργίας του αλγόριθμου. Ένα Quad - Tree τριών διαστάσεων ονομάζεται Oct – tree (οκταδικό δένδρο) επειδή κάθε εσωτερικός του κόμβος έχει 8 παιδιά. Αντίστοιχα ένα Quad – Tree d διαστάσεων ονομάζεται 2^d – Tree , αφού κάθε κόμβος θα έχει 2^d παιδιά.



Εικόνα 23: Παράδειγμα Oct - Tree

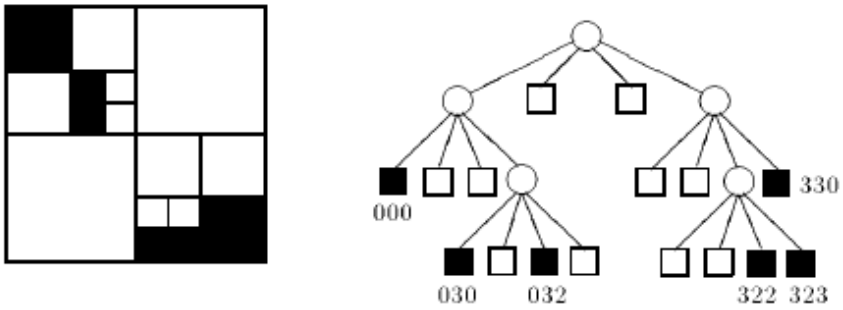
Region Quad - Tree

Το region Quad - Tree είναι το πιο δημοφιλές μέλος της οικογένειας μεθόδων προσπέλασης Quad - Tree και χρησιμοποιείται για την αναπαράσταση δυαδικών εικόνων. Κάθε κόμβος αντιστοιχεί σε ένα τετράγωνο πίνακα από Pixels(η ρίζα αντιστοιχεί σε ολόκληρη την εικόνα). Εάν όλα τα pixel έχουν το ίδιο χρώμα (μαύρο ή άσπρο), τότε ο κόμβος είναι φίλο αυτού του χρώματος. Διαφορετικά ο κόμβος είναι χρώματος γκρίζου και έχει τέσσερα παιδιά. Κάθε ένα από τα παιδιά του κόμβου αυτού αντιστοιχεί σε ένα υπό-πίνακα στον οποίο ο πίνακας του κόμβου έχει τεμαχιστεί.

Κάθε Region Quad – Tree μπορεί να χρησιμοποιηθεί για την αναπαράσταση μιας εικόνας $2^n * 2^n$ pixels, όπου η τιμή κάθε pixel μπορεί να είναι 0 ή 1. Ο κόμβος της ρίζας αντιπροσωπεύει όλη την περιοχή της εικόνας. Αν τα pixel σε μια περιοχή της εικόνας δεν είναι εντελώς 0 ή 1 τότε η περιοχή διασπάτε.

Αν το region Quad – Tree χρησιμοποιηθεί για να αναπαραστήσει ένα σύνολο από δεδομένα σημείων, όπως για παράδειγμα το γεωγραφικό πλάτος και μήκος ενός συνόλου πόλεων, η περιοχές υποδιαιρούνται μέχρι κάθε φύλλο να περιέχει ένα σημείο.

Η εικόνα δείχνει ένα 8 Χ 8 πίνακα pixel και το αντίστοιχο Quad - Tree. Παρατήρηση: Τα μαύρα - άσπρα τετράγωνα αντιπροσωπεύουν μαύρα - άσπρα φύλλα, ενώ οι κύκλοι αντιπροσωπεύουν γκρίζους κόμβους.



Εικόνα 24: Το Region Quad - Tree

Point Region Quad - Tree

Ένα Point Region (PR) Quad – Tree είναι μια εκδοχή του Quad – Tree όπου κάθε κόμβος πρέπει να έχει ακριβώς τέσσερα παιδιά, η φύλλο, το οποίο δεν έχει παιδιά. Το PR Quad – Tree αντιπροσωπεύει μια συλλογή από σημειακά δεδομένα (data points) σε δυο διαστάσεις διασπώντας την περιοχή που περιέχει τα σημειακά δεδομένα σε τέσσερα ίσα τεταρτημόρια (quadrants), υπό-τεταρτημόρια, και συνεχίζει μέχρι κανένας κόμβος - φύλλο να μην περιέχει περισσότερα από ένα σημεία.

3.3.4 Χρήσεις του Quad-Tree

Το Quad – Tree και οι παραλλαγές του έχουν πολλές εφαρμογές, οι περισσότερες από τις οποίες σχετίζονται με τον τομέα της εικόνας. Ειδικότερα το Quad - Tree χρησιμοποιείται:

- Στον τομέα των Computer Graphics για την αναπαράσταση μιας εικόνας ή φωτογραφίας. Επιτρέπει την αναπαράσταση μιας εικόνας σε διαφορετικά επίπεδα ανάλυσης.

- Στον τομέα της ψηφιακής επεξεργασίας εικόνας **(**Image Processing) για την σύμπτυξη (Compact) ή συμπίεση (Compress) μιας εικόνας. Με τον όρο compact εννοούμε την συμπίεση χωρίς απώλειες: Μπορούμε να ανακτήσουμε ακριβώς την πρωτότυπη εικόνα. Αντίθετα, με τον όρο Compress εννοούμε την συμπίεση με απώλειες: Μπορούμε να ανακτήσουμε μια κοντινή προσέγγιση της πρωτότυπης εικόνας.

  Άλλες χρήσεις στον τομέα αυτό είναι για τον τεμαχισμό μίας εικόνας με βάση το χρώμα ή το σχήμα των αντικειμένων που περιέχει ή για τον εντοπισμό αντικειμένων σε μια εικόνα (Object *Modeling)*.

- Το Quad – Tree χρησιμοποιείται από αρκετές μεθόδους για την βάση-περιεχομένου ανάκτηση εικόνας (Content-Based Image Retrieval), με σκοπό την σύλληψη των χωρικών χαρακτηριστικών της εικόνας, για παράδειγμα χρώμα, υφή, σχήμα. Χαρακτηριστικά ερευνητικά πρωτότυπα των μεθόδων αυτών είναι : DISIMA (Lin et al., 2001) , IKONA (Malki et al., 1999) και Virage Image Engine (Gupta & Jain, 1997).

- Χρησιμοποιείται από σχεδιαστικά προγράμματα τύπου CAD - Computer-Aided Design.
- Χρησιμοποιείται στα γεωγραφικά συστήματα πληροφορίας - Geographical Information Systems (GIS).

- Χρησιμοποιείται από στον τομέα της χαρτογράφηση.

- Χρησιμοποιείται από βάσεις δεδομένων εικόνων (Image databases).

- Υλοποιούνται από αρκετά συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) όπως την DB2 (Adler, 2001) και την Oracle (Kothuri et al., 2002)

Τα τελευταία δεκαπέντε χρόνια το Quad – Tree έχει ενσωματωθεί σε πολλές εφαρμογές, είναι αντικείμενο συνεχής έρευνας και ανάπτυξης και έχει πολλά να προσφέρει ακόμα στον τομέα της εικόνας.

3.3. Το R – TREE

3.3.1. Παρουσίαση της λειτουργίας του R-Tree

Τα R - Trees προτάθηκαν από τον Guttman [Gutt84] ως κατ' ευθείαν επέκταση των B+ - Trees [Knut73, Come79] σε *n*-διαστάσεις. Η δομή είναι ένα ισοζυγισμένο δέντρο που αποτελείται από ενδιάμεσους και τερματικούς κόμβους. Ένας *τερματικός κόμβος* είναι του τύπου:

$$(oid, R)$$

όπου *oid* είναι η ταυτότητα (identifier) του αντικειμένου και χρησιμοποιείται για να αναφερόμαστε στο αντικείμενο που είναι αποθηκευμένο στη Βάση δεδομένων. Το *R* είναι η προσέγγιση MBR (Minimum Bounding Rectangle) του αντικειμένου, δηλαδή είναι του τύπου

$$(p_{l\text{-}1}, p_{l\text{-}2}, \ldots, p_{l\text{-}n}, p_{u\text{-}1}, p_{u\text{-}2}, \ldots, p_{u\text{-}n})$$

ο οποίος αναπαριστά τις 2*n* συντεταγμένες τις κάτω-αριστερής (p_l) και της πάνω-δεξιάς (p_u) γωνίας του *n* - διάστατου ορθογωνίου *p*. Ένας *ενδιάμεσος κόμβος* είναι του τύπου

$$(ptr, R)$$

όπου *ptr* είναι ένας δείκτης (pointer) σε έναν κόμβου κατωτέρου επιπέδου του δέντρου και *R* είναι η αναπαράσταση του ορθογωνίου που περικλείει τα στοιχεία του κόμβου αυτού.

Έστω *M* το μέγιστο πλήθος στοιχείων ενός κόμβου και έστω $m \leq M/2$ μια παράμετρος που καθορίζει το ελάχιστο πλήθος στοιχείων ενός κόμβου[1].

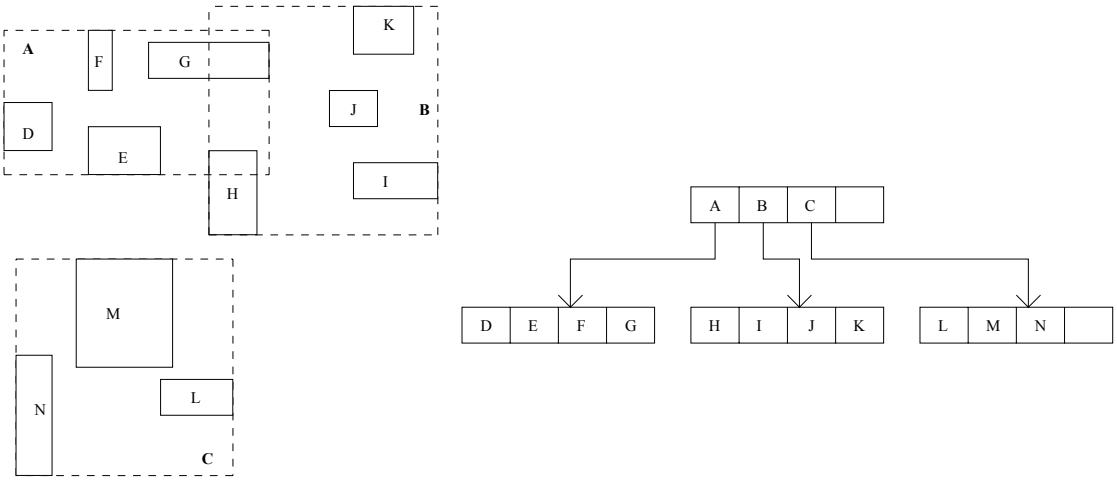
Ένα R - Tree έχει τις ακόλουθες ιδιότητες :

1. Κάθε τερματικός κόμβος περιέχει μεταξύ *m* και *M* στοιχεία εκτός εάν είναι η ρίζα του δέντρου.
2. Σε κάθε στοιχείο (*oid*, *R*) ενός τερματικού κόμβου, το *R* είναι το ελάχιστο ορθογώνιο που περιέχει το αντικείμενο που αναπαρίσταται από το *oid*.

[1] Η παράμετρος *m* είναι ρυθμιζόμενη. Ο Guttman, για παράδειγμα, δοκίμασε τις τιμές $m = M/2$, $m = M/3$, $m = 2$ στην εργασία [Gutt84]. Από την άλλη πλευρά, η παράμετρος *M* είναι συνάρτηση του μεγέθους μιας σελίδας δίσκου σε bytes (π.χ. $M = 50$ για σελίδες των 1024 bytes και τις συντεταγμένες του ορθογωνίου να απαιτούν από 4 bytes η κάθε μια).

3. Κάθε ενδιάμεσος κόμβος περιέχει μεταξύ m και *M* παιδιά εκτός εάν είναι η ρίζα του δέντρου.
4. Σε κάθε στοιχείο (*ptr*, *R*) ενός ενδιάμεσου κόμβου, το *R* είναι το ελάχιστο ορθογώνιο που περιέχει τα ορθογώνια που βρίσκονται στον κόμβο-παιδί.
5. Η ρίζα έχει τουλάχιστον δύο παιδιά εκτός εάν είναι τερματικός κόμβος.
6. Όλοι οι τερματικοί κόμβοι (φύλλα) βρίσκονται στο ίδιο επίπεδο του δέντρου.

Η **εικόνα 25** απεικονίζει ένα σύνολο ορθογωνίων και το αντίστοιχο R - Tree που χτίζεται πάνω σε αυτά τα ορθογώνια (υποθέτουμε μέγιστη χωρητικότητα κόμβου *M* = 4).



Εικόνα 25: Ορισμένα ορθογώνια, οργανωμένα σε ένα R - Tree και το αντίστοιχο R - Tree

3.3.2. Παρουσίασης της διαδικασίας εισαγωγής/αναζήτησης

Η διαδικασία αναζήτησης είναι απλή και λειτουργεί με τρόπο αντίστοιχο όπως στο B+ - Tree: Δοθέντος ενός ορθογωνίου *Q*, ο αλγόριθμος αναζήτησης διατρέχει το δέντρο, ξεκινώντας από τη ρίζα και επιλέγοντας σε κάθε επίπεδο τα υποδένδρα (ίσως περισσότερα από ένα) που αναπαριστώνται από ορθογώνια που επικαλύπτονται με το *Q*.

Αλγόριθμος Αναζήτησης.

Βρες όλα τα αντικείμενα των οποίων τα ορθογώνια τέμνουν ένα ορθογώνιο ερώτησης *Q*.

Βήμα1 Θέτουμε *N* να είναι η ρίζα.

Βήμα 2 Εάν *N* δεν είναι τερματικός κόμβος,

ελέγχουμε κάθε στοιχείο *E* του *N* του οποίου το ορθογώνιο *E.R* τέμνει το *Q*.

Για όλα τα τεμνόμενα *E*

εκτελούμε τον **Αλγόριθμο Αναζήτησης** στο υποδένδρο του οποίου η ρίζα δείχνεται από τον δείκτη *E.ptr*.

Βήμα 3 Εάν *N* είναι τερματικός κόμβος,

ελέγχουμε κάθε στοιχείο *E* του *N* για να εντοπίσουμε αν το *E.R* τέμνει το *Q*.

Εάν αυτό ισχύει, το *E* είναι απάντηση στην ερώτηση.

Όπως η διαδικασία αναζήτησης, έτσι και η διαδικασία εισαγωγής είναι παρόμοια με αυτήν του B+ - Tree: τα νέα δεδομένα προστίθενται στα φύλλα, οι κόμβοι που υπερχειλίζουν διασπώνται και οι διασπάσεις μεταφέρονται προς τα ανώτερα επίπεδα του δέντρου. Όταν ένα νέο ορθογώνιο εισάγεται στη δεντρική δομή, ο αλγόριθμος *ChooseSubtree* επιλέγει τον κόμβο που είναι ο πιο κατάλληλος για την ανάθεση σε αυτόν του νέου στοιχείου.

Αλγόριθμος ChooseSubtree. Επέλεξε έναν τερματικό κόμβο στον οποίο να εισαχθεί ένα νέο στοιχείο *I*.

Βήμα 1 Θέτουμε *N* να είναι η ρίζα.

Βήμα 2 Εάν *N* είναι τερματικός κόμβος,

επιστρέφουμε το *N*.

αλλιώς

Επιλέγουμε το στοιχείο *E* του *N* του οποίου το ορθογώνιο *E.R* απαιτεί ελάχιστη αύξηση εμβαδού για να συμπεριλάβει το *I.R*.

Επιλύουμε τις ισοπαλίες επιλέγοντας το στοιχείο με το μικρότερο εμβαδόν.

Βήμα 3 Θέτουμε *N* να είναι ο κόμβος-παιδί που δείχνεται από τον δείκτη *E.ptr* του επιλεγμένου κόμβου και

επαναλαμβάνουμε από το **Βήμα** 2.

Αφού η απόφαση για το αν θα επισκεφθούμε έναν κόμβο κατά τη διάρκεια της διαδικασίας αναζήτησης εξαρτάται από το εάν το ορθογώνιο που περιβάλλει τον κόμβο αυτόν επικαλύπτεται με την περιοχή αναζήτησης, πρέπει να επιδιώκουμε την ελάχιστη αύξηση του εμβαδού (area) του κόμβου που επιλέγεται για τη νέα εισαγωγή Αυτό το κριτήριο είναι αποφασιστικό για την απόδοση του R - Tree. Διαφορετικά κριτήρια οδηγούν σε τελείως διαφορετικές δεντρικές δομές.

Εάν ο κόμβος που επιλέχθηκε για την εισαγωγή υπερχειλίσει τότε ο αλγόριθμος Split διαιρεί τον κόμβο αυτόν σε δύο νέους κόμβους. Ο Guttman πρότεινε τρεις διαφορετικές μεθόδους διάσπασης με διαφορετικά κόστη εκτέλεσης: τον Linear - Split, τον Quadratic - Split και τον Exhaustive - Split αλγόριθμο. Στην πράξη εφαρμόζονται οι δύο πρώτοι μόνο καθώς ο Exhaustive - Split είναι πολύ αργός για ρεαλιστικές εφαρμογές (απαιτεί τον υπολογισμό περίπου 2^M πιθανών ομαδοποιήσεων για να επιλέξει την πιο κατάλληλη από αυτές). Σημειώνουμε εδώ ότι το ίδιο κριτήριο (ελάχιστη αύξηση εμβαδού) χρησιμοποιείται τόσο στους παραπάνω αλγορίθμους Split όσο και στον αλγόριθμο ChooseSubtree που παρουσιάσαμε προηγουμένως.

Εάν μελετήσουμε τη διαδικασία αναζήτησης στο R - Tree θα συμπεράνουμε ότι οι παράγοντες *κάλυψη* (*coverage*) και *επικάλυψη* (*overlap*) είναι σημαντικοί. Η *κάλυψη* για ένα επίπεδο του R - Tree ορίζεται ως το συνολικό εμβαδόν όλων των ορθογωνίων που χαρακτηρίζουν τους κόμβους του επιπέδου αυτού. Η *επικάλυψη* για ένα επίπεδο του R - Tree ορίζεται ως το συνολικό εμβαδόν του χώρου που ανήκει σε δύο ή περισσότερους επικαλυπτόμενους κόμβους. Είναι προφανές ότι η αποδοτική αναζήτηση με τα R - Trees προϋποθέτει ελαχιστοποίηση και των δύο παραγόντων. Η ελαχιστοποίηση της κάλυψης συνεπάγεται μείωση της 'νεκρής' περιοχής (δηλαδή του κενού χώρου) μεταξύ των κόμβων. Η ελαχιστοποίηση της επικάλυψης είναι ακόμη πιο σημαντική για γρήγορη αναζήτηση: όταν το παράθυρο ερώτησης πέφτει μέσα στον κοινό χώρο *k* επικαλυπτόμενων κόμβων στο επίπεδο *h-l, h* το ύψος του δέντρου, πρέπει να ακολουθηθούν, στη χειρότερη περίπτωση, *k* μονοπάτια προς τα φύλλα (ένα μονοπάτι για κάθε έναν από τους επικαλυπτόμενους κόμβους), με αποτέλεσμα την καθυστέρηση της διαδικασίας αναζήτησης. Για παράδειγμα, αν θεωρήσουμε ένα παράθυρο ερώτησης που βρίσκεται στην κοινή περιοχή των κόμβων A και B του R-tree της εικόνας 24 και τα δύο υποδένδρα που έχουν ως ρίζες τους κόμβους A και B πρέπει να ελεγχθούν, έστω και αν ένα μόνο από αυτά περιλαμβάνει κάποια απάντηση στην ερώτηση.

Άλλες εκδοχές του R - Tree (Variations)

Ο αριθμός των παραλλαγών του R - Tree που έχουν εμφανιστεί στην βιβλιογραφία είναι πολύ μεγάλος. Εμείς θα παρουσιάσουμε τις δύο πιο σημαντικές παραλλαγές του R - Tree το R+ - Tree και το R* - Tree

R+ - Tree

Το R+ - Tree προτάθηκε από τους Sellis κ.α. στην εργασία [Sell87] και μπορεί να θεωρηθεί ως μια επέκταση της δομής K-D-B - Tree για διαχείριση μη σημειακών αντικειμένων. Οι κόμβοι του R+ - Tree σε κάθε επίπεδο, δεν καλύπτουν υποχρεωτικά όλο το χώρο εργασίας. Από την άλλη πλευρά, σε αντίθεση με το R - Tree, το R+ - Tree εγγυάται μηδενική επικάλυψη μεταξύ των ενδιάμεσων κόμβων σε κάθε επίπεδο του δέντρου. Τα ορθογώνια που τέμνουν περισσότερους από έναν κόμβους αποθηκεύονται (σε κατάλληλα κομμάτια) σε όλους αυτούς τους κόμβους. Η δομή του μοιάζει με αυτή του R - Tree: οι τερματικοί κόμβοι είναι του τύπου (*oid*, *R*) και οι ενδιάμεσοι κόμβοι είναι του τύπου (*ptr*, *R*).

Το R+ - Tree έχει τις ακόλουθες ιδιότητες:

1. Για κάθε στοιχείο (*ptr*, *R*) ενός τερματικού κόμβου, το υποδέντρο που έχει ως ρίζα τον κόμβο στον οποίο δείχνει ο *ptr* περιέχει ένα ορθογώνιο *R'* εάν και μόνο εάν το *R'* καλύπτεται πλήρως από το *R*. Η μόνη εξαίρεση είναι όταν το *R'* είναι ορθογώνιο τερματικού κόμβου: σε τέτοια περίπτωση το *R'* πρέπει απλά να τέμνει το *R*.
2. Για κάθε ζεύγος στοιχείων (*ptr*1, *R*1) και (*ptr*2, *R*2) ενός ενδιάμεσου κόμβου, η επικάλυψη μεταξύ των ορθογωνίων *R*1 και *R*2 ισούται με μηδέν.
3. Η ρίζα έχει τουλάχιστον δύο παιδιά εκτός εάν είναι φύλλο.
4. Όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο του δέντρου.

Η λειτουργία αναζήτησης είναι παρόμοια με αυτήν των R - Trees. Η κύρια διαφορά είναι ότι οι υποπεριοχές των R+ - Trees δεν επικαλύπτονται, οπότε έχουμε συνήθως γρηγορότερη αναζήτηση.

R* - Tree

Το R* - Tree [Beck90] παρόλο που προτάθηκε το 1990 θεωρείται ακόμα και σήμερα ως μία κυριαρχική δομή όσον αφορά την απόδοση. Είναι μια παραλλαγή του R - Tree η οποία, σε αντίθεση με το R+ - Tree, δεν αλλάζει τις ιδιότητες της δομής, αλλά χρησιμοποιεί έναν πολύπλοκο αλγόριθμο (forced reinsertion) για να οργανώσει τα ορθογώνια στους κόμβους, επιτυγχάνοντας έτσι βελτιωμένη απόδοση. Όπως εξάγεται από τους αλγόριθμους κατασκευής του R - Tree, η ελάχιστη αύξηση εμβαδού είναι το κριτήριο που αποφασίζει για τον κόμβο που θεωρείται ως ο πιο κατάλληλος για την εκχώρηση ενός νέου στοιχείου (αλγόριθμος ChooseSubtree) ή τον τρόπο με τον οποίο θα ομαδοποιηθούν σε δύο κόμβους τα στοιχεία ενός κόμβου που υπερχειλίζει (αλγόριθμος Split). Το R* - Tree προτείνει συνδυασμούς περισσοτέρων κριτηρίων: ελάχιστη αύξηση εμβαδού (area), ελάχιστη αύξηση περιμέτρου (margin) και ελάχιστη αύξηση επικάλυψης (overlap).

2.3.5. Χρήσεις του R - Tree

Από τις πρώτες μελέτες του R - Tree ήταν φανερό ότι μπορούσε να βρει εφαρμογές σε πολλά πεδία. Τρεις από τους σημαντικότερους τομείς που χρησιμοποιείται είναι τα γεωγραφικά συστήματα πληροφορίας (GIS), ο τομέας σχεδίασης ολοκληρωμένων κυκλωμάτων (σχεδίαση VLSI) και ο τομέας σχεδίασης CAD.

Άλλα πεδία διαχείρισης μη παραδοσιακών δεδομένων (πολυδιάστατων) που χρησιμοποιούν R – Tree είναι οι Βάσεις δεδομένων Εικόνων και Πολυμέσων. Σε εφαρμογές Πολυμέσων, για παράδειγμα, τα αντικείμενα που συμμετέχουν μπορούν να θεωρηθούν ως ορθογώνια παραλληλεπίπεδα στο τρισδιάστατο χώρο-χρονικό (spatio-temporal) σύστημα συντεταγμένων, 3D R - Tree (δύο διαστάσεις για τη χωρική θέση του αντικειμένου συν μία διάσταση για τη χρονική διάρκεια του) και, άρα, να οργανωθούν με χρήση χωρικών δομών δεδομένων. Επιπλέον, χρησιμοποιείται στην διαχείριση γεωδαιτικών δεδομένων (geodetic data), δηλαδή δεδομένα που αντιπροσωπεύουν γεωγραφικό μήκος και πλάτος. Τέλος, έχει ενσωματωθεί σε προτάσεις για ειδικευμένες γλώσσες χωρικών ερωταποκρίσεων (π.χ. PSQL, Spatial SQL).

3.6 Εξωτερικά συμπεράσματα για την απόδοση των αλγορίθμων

Για την απόδοση του Binary Search – Tree μιλήσαμε νωρίτερα, σχολιάζοντας και την χειρότερη περίπτωση λειτουργίας του, στο σημείο αυτό θα σχολιάσουμε την απόδοση των χωρικών δομών δεδομένων που μελετήσαμε, του Quad – Tree και R – Tree. Άλλωστε δεν έχει νόημα η σύγκριση του Binary Search – Tree με χωρικές δομές δεδομένων.

Η απόδοση μιας χωρικής δομής δεδομένων συνήθως αξιολογείται από την ικανότητα της να απαντά σε χωρικές ερωτήσεις. Στο παρελθόν έχουν προταθεί αρκετές δομές που επιδιώκουν τη βέλτιστη ανάκτηση τέτοιων ερωτήσεων, καμία από αυτές, όμως, δεν έχει αποδειχτεί ότι είναι ανώτερη από τις υπόλοιπες για όλες τις περιπτώσεις. Αυτό οφείλεται στην ύπαρξη πολλών παραμέτρων που καθορίζουν την αποδοτικότητα μιας δομής αλλά και στην έλλειψη ενός τυποποιημένου δοκιμαστηρίου (testbed / benchmark) για χωρικές δομές δεδομένων. Οι κύριοι παράγοντες που καθορίζουν την αποδοτικότητα μιας χωρικής δομής δεδομένων :

- **Η Πυκνότητα των δεδομένων :** Αν τα δεδομένα είναι κατανεμημένα πολύ αραιά η επιλογή της μεθόδου γίνεται λιγότερο σημαντική

- **Η κατανομή των δεδομένων :** Υπάρχει εξάρτηση, το πού τίθεται ένα ερώτημα, σε σχέση με την κατανομή των δεδομένων.

- **Το μέγεθος της γεωμετρίας :** Εάν η γεωμετρία είναι μεγάλη σε σχέση με το ερώτημα τότε η επιλογή της μεθόδου έχει λιγότερη σημασία.

- **Ευθυγράμμιση της γεωμετρίας :** το πώς η γεωμετρία είναι ευθυγραμμισμένη σε σχέση με του άξονες X και Y πιθανώς να επηρεάσει την απόδοση της μεθόδου.

- **Το σχήμα της γεωμετρίας :** Εάν οι γεωμετρίες είναι συμμετρικές σε σχήμα (π.χ. ορθογώνιο) τότε η μέθοδος R – Tree μπορεί να είναι πιο αποτελεσματική.

- **Το είδος του ερωτήματος :** Για παράδειγμα τα R- Trees φαίνονται ιδιαίτερα αποτελεσματικά στα ερωτήματα κοντινότερου γείτονα

Λίγα θεωρητικά αποτελέσματα ως προς την απόδοση των R –Trees και Quad - Trees έχουν παρουσιασθεί σχετικά πρόσφατα, και τα οποία επικεντρώνονται τόσο στα χαρακτηριστικά τους όσο και στην απόδοση τους σε χωρικές ερωτήσεις.

Τα συμπεράσματα τα οποία παρουσιάζουμε, βασίζονται στην πλατφόρμα της Oracle η οποία έχει ενσωματώσει τις χωρικές δομές δεδομένων που μελετήσαμε:

- Τα Quad Trees είναι μόνο για αντικείμενα δύο διαστάσεων ενώ τα R-Trees για αντικείμενα μέχρι τεσσάρων διαστάσεων στην υλοποίηση της Oracle, ενώ θεωρητικά μπορούν να χρησιμοποιηθούν και για περισσότερες διαστάσεις.

- Τα Quad – Trees αποδίδουν αρκετά καλά σχεδόν για κάθε είδους σύνολο δεδομένων. Αντίθετα υπάρχουν σύνολα δεδομένων στα οποία τα R – Trees δεν αποδίδουν καλά

- Το R – Tree απαιτεί λιγότερο αποθηκευτικό χώρο για τη δομή, σε σχέση με τα Quad - Trees. Παρατήρηση: Εκτός αν πρόκειται για σημειακά δεδομένα όπου δεν υπάρχει διαφορά.

- Τα R – Trees εκτελούν γρηγορότερα ερωτήματα κοντινότερου γείτονα σε σχέση με Quad - Trees. Αυτό συμβαίνει, γιατί η μέθοδος R – Tree αποθηκεύει τα χωρικά δεδομένα (δημιουργεί την δενδρική δομή) με βάση την τοποθεσία των αντικειμένων, αυτός είναι και ο βέλτιστος τρόπος για ερωτήματα κοντινότερου γείτονα. Επιπλέον, τα Quad – Trees δεν μπορούν να απαντήσουν σε ερωτήματα αυξητικού κοντινότερου γείτονα (Incremental NN Query). Τέλος, η απόδοση του Quad – Tree σε ερωτήματα κοντινότερου γείτονα, κρίνεται πολύ υψηλή σε σχέση με άλλες δομές, όπως και η επίδοση τους σε ορισμένες ερωτήσεις χωρικής διάσπασης, όπως για παράδειγμα Point-in-Polygon search.

- Τα R – Trees μπορούν να χρησιμοποιηθούν για γεωδαιτικά δεδομένα (geodetic data), δηλαδή δεδομένα που αντιπροσωπεύουν γεωγραφικό μήκος

και πλάτος. Τα δεδομένα αυτά συνήθως περιέχουν και μία τρίτη διάσταση για το λόγω αυτό τα Quad – Trees δεν προτείνονται για αυτού του τύπου τα δεδομένα.

- Το R-Tree μπορεί να είναι αργό στις διαδικασίες ενημέρωσης, εισόδου ή εξόδου των δεδομένων. Εάν η επίδοση στις παραπάνω διαδικασίες είναι σημαντική το Quad – Tress είναι καλύτερη επιλογή, αφού εκτελεί γρηγορότερα αυτές τις διαδικασίες.

- Αν γίνονται συνεχείς ενημερώσεις (updates) στα δεδομένα, τα R-Trees γίνονται λιγότερο αποδοτικά. Αν για παράδειγμα έχουμε συνεχής προσθήκη δεδομένων με το χρόνο το R – Tree θα χάσει την αποδοτικότατα του. Ένας τρόπος για να ξεπεράσουμε το πρόβλημα αυτό είναι να ξανά δημιουργήσουμε το R – Tree από την αρχή. Τα Quad – Trees συνεχίζουν να είναι αποδοτικά ακόμα και μετά από πολλές ενημερώσεις.

ΚΕΦΑΛΑΙΟ 4

ΥΛΟΠΟΙΗΣΗ

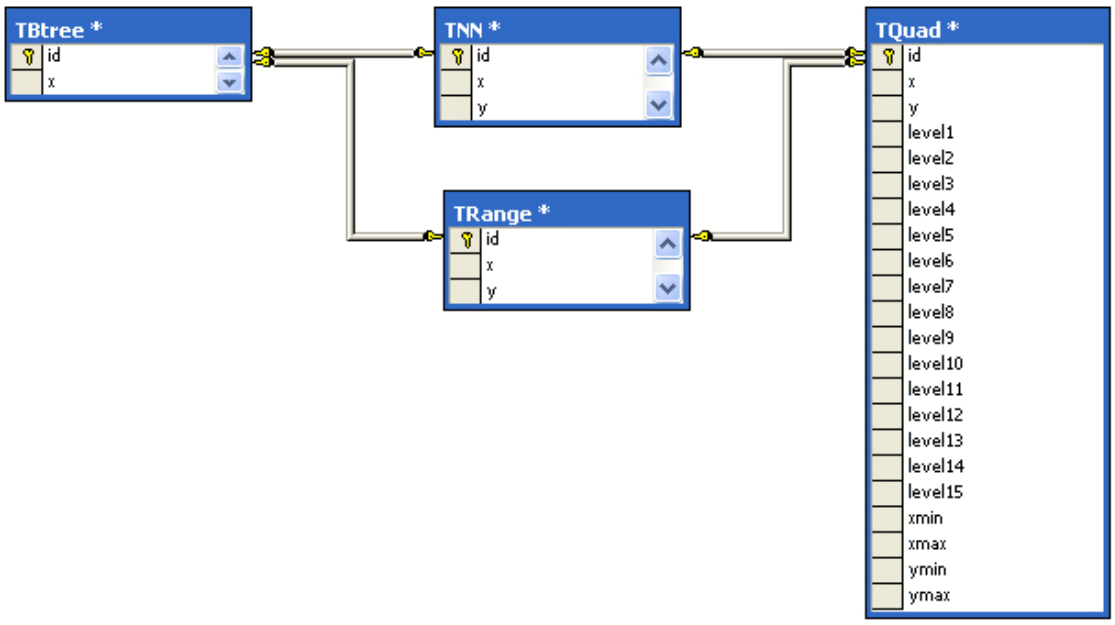
Το αντικείμενο του κεφαλαίου αυτού είναι η περιγραφή της υλοποίησης της εφαρμογής. Παρουσιάζονται, οι λεπτομέρειες υλοποίησης της εφαρμογής, ο λεπτομερής έλεγχος της εφαρμογής και η αξιολόγηση της απόδοσής της.

4.1 Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή παρουσιάζονται οι λεπτομέρειες υλοποίησης της εφαρμογής . Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++ σε συνδυασμό με εντολές SQL. Ο προγραμματισμός έγινε στο περιβάλλον ανάπτυξης Borland C++ Builder. Παράλληλα χρησιμοποιήθηκαν ADO οδηγοί πρόσβασης σε βάση δεδομένων. Για τη δημιουργία και διαχείριση της βάσης δεδομένων της εφαρμογής χρησιμοποιήθηκε ο SQL Server.

4.1.1 Η Βάση δεδομένων

Ως αποθηκευτικό μέσο των δεδομένων χρησιμοποιείται η βάση **data** στο σύστημα του SQL Server. Η βάση δεδομένων περιέχει τους πίνακες: **TBtree, TQuad, TRange, TNN** και παρουσιάζεται στην παρακάτω εικόνα.



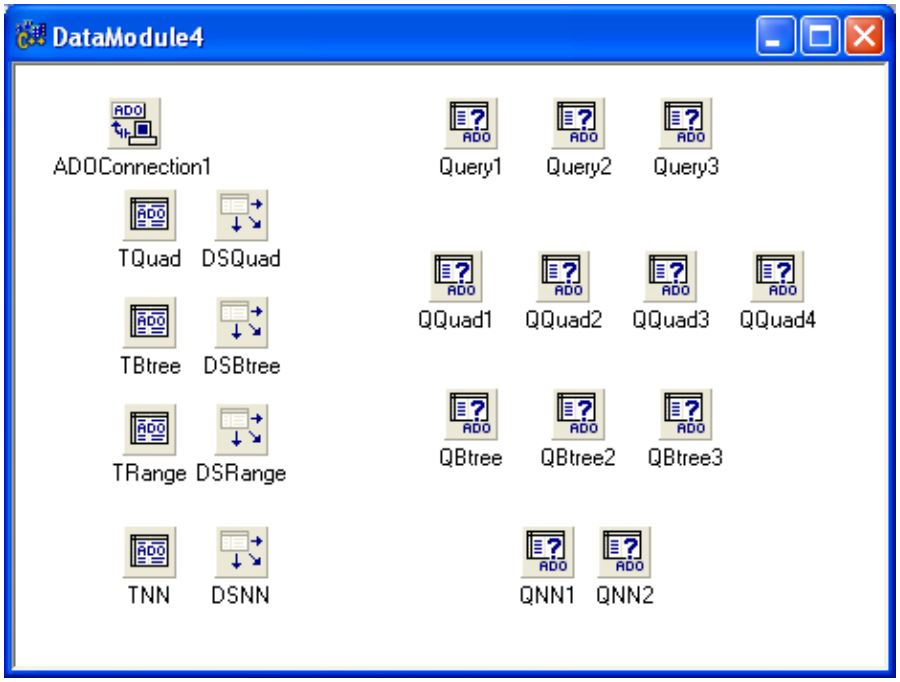
Εικόνα 26: Διάγραμμα των πινάκων

Ο πίνακας TBtree χρησιμοποιείται για την αποθήκευση των δεδομένων του αλγόριθμου ταξινόμησης Binary Search tree.
Ο πίνακας TQuad χρησιμοποιείται για την αποθήκευση των δεδομένων του αλγόριθμου ταξινόμησης Quad Tree.
Ο πίνακας TRange χρησιμοποιείται για την αποθήκευση των δεδομένων του ερωτήματος ακτίνας (Range Query).

Ο πίνακας TNN χρησιμοποιείται για την αποθήκευση των δεδομένων του ερωτήματος κοντινότερου γείτονα (Nearest Neighbour Query).

Ο πίνακας TRange χρησιμοποιείται για την αποθήκευση των δεδομένων του ερωτήματος ακτίνας (Range Query).

Η Επικοινωνία της εφαρμογής με την βάση δεδομένων γίνεται με την βοήθεια ενός Datamodule. Το Datamodule παρέχει τη δυνατότητα σε μια εφαρμογή να συγκρατεί όλα τα μη οπτικά συστατικά της, που έχουν σχέση με τη βάση δεδομένων της. Μέσα στο Datamodule υπάρχουν τοποθετημένα όλα τα συστατικά που επιτρέπουν στην εφαρμογή να αντλεί, να προσθέτει και να τροποποιεί στοιχεία από την βάση δεδομένων. Συγκεκριμένα μέσα σε αυτό βρίσκονται τα ακόλουθα συστατικά:



Εικόνα 27: Το Datamodule της εφαρμογής

ADOConnection1: Υλοποιεί τη σύνδεση με τη βάση δεδομένων. Η σύνδεση επιτυγχάνεται με την βοήθεια ενός αλφαριθμητικού (αλφαριθμητικό σύνδεσης) και το οποίο είναι:

Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=data;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=AMDATHLONXP64BIT;Use Encryption for Data=False;Tag with column collation when possible=False

Όλα τα υπόλοιπα συστατικά του datamodule συνδέονται μέσω αυτού με τη βάση δεδομένων.

TQuad – Tbtree (ADOTable): Παρέχει πρόσβαση στο πίνακα TQuad. Μέσω αυτού γίνεται προσπέλαση των δεδομένων του πίνακα TQuad που βρίσκεται στην βάση δεδομένων. Αντίστοιχα, το συστατικά TBtree παρέχει πρόσβαση στο πίνακα TBtree.

TRange – TNN (ADOTable): Περιέχουν τα δεδομένα (σημεία) τα οποία αποτελούν απάντηση στα ερωτήματα κοντινότερου γείτονα και ακτίνας.

DSQuad – DSBtree – DSRange – DSNN (DataSource): Μέσω αυτών των συστατικών, είναι εφικτή η εμφάνιση των δεδομένων του TQuad, TBtree, TRange και TNN στα DBGrid (πίνακες) της εφαρμογής.

Query1 – Query2 – Query3 – Qquad1 - QQuad2 - QQuad3 - QQuad4 - QBtree1 - QBtree2 - QBtree3 – QNN1 –QNN2: Τα παραπάνω συστατικά εκτελούν διάφορα SQL ερωτήματα κάθε φορά. Εισάγουμε τις εντολές SQL στην ιδιότητα SQL του συστατικού ADOQuery και στην συνέχεια τις εκτελούμε. Το συστατικό ADOQuery μπορεί να ανοίγει με την εντολή Open() και να έχουμε πρόσβαση στα δεδομένα που παίρνουμε από την εκτέλεση του ερωτήματος. Κλείνει με την εντολή Close(). Ένα παράδειγμα εκτέλεσης ADOQuery είναι το παρακάτω, με το οποίο έχουμε πρόσβαση σε όλα τα δεδομένα του πίνακα TQuad.

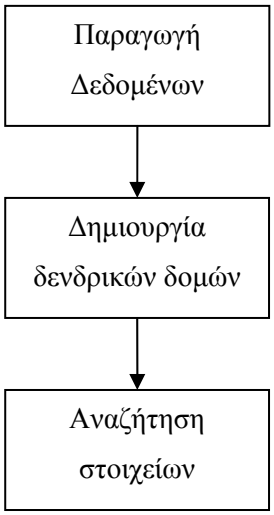
```
DataModule4->QQuad2->SQL->Text="SELECT * FROM TQuad";
DataModule4->QQuad2->ExecSQL();
```

4.1.2 Η εφαρμογή

Η εφαρμογή αποτελείται από τρία επίπεδα:

- Παραγωγή δεδομένων
- Δημιουργία των δενδρικών δομών δεδομένων
- Αναζήτηση δεδομένων

Τα επίπεδα της εφαρμογής φανερώνουν και την φιλοσοφία λειτουργίας της. Αρχικά παράγουμε - εισάγουμε τα δεδομένα στην βάση δεδομένων, με έναν από τους τρόπους που θα περιγράψουμε παρακάτω. Στη συνέχεια τα επεξεργαζόμαστε με χρήση των αλγορίθμων ταξινόμησης δημιουργώντας τις απαραίτητες για την αναζήτηση δενδρικές δομές. Τέλος, στο τρίτο επίπεδο, μπορούμε να εκτελέσουμε τις αναζητήσεις στοιχείων.



Εικόνα 28: Τα επίπεδα της εφαρμογής

4.1.3 Τρόποι παραγωγής και εισόδου των δεδομένων στην εφαρμογή

Η παραγωγή και είσοδος των δεδομένων είναι ένας τομέας της εφαρμογής στον οποίο δόθηκε ιδιαίτερη έμφαση. Η ποιότητα των δεδομένων παίζει καθοριστικό ρόλο στην λύση ενός προβλήματος, αφού αν τα δεδομένα εισόδου δεν είναι ακριβή τότε και η λύση του προβλήματος θα είναι λανθασμένη.

Παραγωγή τυχαίων δεδομένων με χρήση του Matlab:

Το Matlab παρέχει ευρείες δυνατότητες για την παραγωγή τυχαίων δεδομένων από τον χρήστη μέσα από ένα μεγάλο φάσμα συναρτήσεων. Ενδεικτικά αναφέρουμε ορισμένες: *rand, randn, sprand, sprandn, randperm.* Στην εφαρμογή μας χρησιμοποιήσαμε μια από τις δημοφιλέστερες συναρτήσεις παραγωγής τυχαίων δεδομένων στο Matlab, την Rand.

Rand: Η συνάρτηση Rand παράγει ψευδό-τυχαίους αριθμούς. Η ακολουθία των αριθμών που παράγονται καθορίζεται από την κατάσταση της γεννήτριας. Η γεννήτρια παράγει δεκαδικούς αριθμούς στο διάστημα [1.1102e-016 , 1.0000]. Θεωρητικά μπορεί να παράγει περισσότερες άπειρες τιμές.

Η Σύνταξη της rand είναι η ακόλουθη : rand(M,N) όπου M, N οι διαστάσεις ενός πίνακα.

Για την σύνδεση του Matlab με τον SQL Server, την παραγωγή δεδομένων και την εισαγωγή τους στον SQL Server χρησιμοποιήσαμε τον παρακάτω κώδικα.

```
>> con = database('sqlserver','username','password') Σύνδεση Matlab με SQL Server
>> n=32768
>> a = ceil(n.*rand(M,N)); Παραγωγή πίνακα M γραμμών και N στηλών με τυχαία
   δεδομένα
>> name = ('id','x','y')
>>  insert(con,'Table',name,a) Εισαγωγή δεδομένων a στον πίνακα Table στα πεδία
     name
```

Παραγωγή τυχαίων δεδομένων με χρήση της Borland C++:

Για την παραγωγή τυχαίων δεδομένων, μέσα από την εφαρμογή, χρησιμοποιήσαμε τις συναρτήσεις *randomize* και *rand* της C++.

Η randomize εκκινεί την γεννήτρια τυχαίων αριθμών.

Η rand χρησιμοποιεί μια γεννήτρια τυχαίων αριθμών με εύρος από 2^0 μέχρι 2^{32} -1 για να επιστρέφει ψευδό-τυχαίους αριθμούς από το 0 μέχρι το RAND_MAX. Το

RAND_MAX δηλώνεται μέσα στο αρχείο επικεφαλίδας stdlib.h όπου καθορίζεται η μέγιστη του τιμή 0x7FFF (32767).

Η σύνταξη για την παραγωγή τυχαίων δεδομένων στην εφαρμογή μας είναι η ακόλουθη :

```
randomize();
rand();
```

Εισαγωγή δεδομένων από αρχείο:

Η εφαρμογή παρέχει την δυνατότητα στον χρήστη να εισάγει δεδομένα από αρχείο κειμένου (Text Document, *.txt). Η σύνταξη του αρχείου για την εισαγωγή δεδομένων στον πίνακα TQuad είναι η ακόλουθη:

Τιμή_X Τιμή_Y

ενώ στον TBtree

Τιμή_X

αφού είναι μονοδιάστατος.

Ο κώδικας που χρησιμοποιήσαμε για την εισαγωγή των δεδομένων από αρχείο στον πίνακα TQuad είναι ο ακόλουθος:

```
fp=fopen(OpenDialog1->FileName.c_str(),"r");
  do
                {
          fscanf(fp,"%d",&x);
          fscanf(fp,"%d",&y);
          DataModule4->Query1->SQL->Text="INSERT INTO TQuad (id,x,y) VALUES
('"+AnsiString(id)+"','"+AnsiString(x)+"','"+AnsiString(y)+"')";
          DataModule4->Query1->ExecSQL();
          id++;
               }while(!feof(fp));  //telos while
          fclose(fp);
```

Ο αντίστοιχος κώδικας για εισαγωγή δεδομένων στον πίνακα TBtree διαφέρει σε ελάχιστα σημεία.

Εισαγωγή δεδομένων από εικόνα:

Η εφαρμογή παρέχει την δυνατότητα στον χρήστη να εισάγει δεδομένα από εικόνα. Η διαδικασία είναι πού απλή, ανοίγουμε το αρχείο εικόνας και επιλέγουμε τα σημεία με το ποντίκι κάνοντας click. Η εφαρμογή συνεργάζεται με τους παρακάτω τύπους εικόνας:

Bitmaps (*.bmp)

JPEG Image File (*.jpg)

JPEG Image File (*.jpeg)

Ο κώδικας που χρησιμοποιήσαμε για την εισαγωγή των δεδομένων από εικόνα στον πίνακα TQuad είναι ο ακόλουθος:

```
DataModule4->Query1->SQL->Text="INSERT INTO TQuad (id,x,y) VALUES ('"+AnsiString(i)+"','"+AnsiString(X)+"','"+AnsiString(Y)+"')";
DataModule4->Query1->ExecSQL();
```

Στην περίπτωση που ο τύπος αρχείου εικόνας είναι Bitmap (*.bmp), τα σημεία (pixels εικόνας) που επιλέγονται χρωματίζονται κόκκινα, ώστε ο χρήστης να γνωρίζει ποια σημεία έχει εισάγει. Ο κώδικάς που χρησιμοποιούμε στην περίπτωση αυτή είναι ο παρακάτω:

```
for(i=-1;i<=1;i++)
    for(j=-1;j<=1;j++)
     Image1->Canvas->Pixels[X+i][Y+j]=clRed;
```

4.1.3 Δημιουργία των δενδρικών δομών δεδομένων

Στην εφαρμογή υλοποιήσαμε δυο δενδρικές δομές δεδομένων. Το Binary Search – Tree και το Quad - Tree. Οι δομές αυτές στηρίζονται στις στατικές εκδοχές των αλγόριθμων. Αυτό σημαίνει ότι εστιάζουν στη συνεργασία με τη βάση δεδομένων και επεξεργάζονται ένα πεπερασμένο σύνολο δεδομένων, το οποίο πρέπει είναι εκ’ των προτέρων γνωστό.

Υλοποίηση του Binary Search – Tree:

Ο αλγόριθμος Binary Search – Tree προσπελαύνει τα δεδομένα από τον πίνακα TBtree σειριακά ξεκινώντας από το πρώτο στοιχείο και συνεχίζοντας μέχρι το τελευταίο.

Βήμα 1: Το πρώτο στοιχείο που προσπελαύνετε γίνεται ρίζα του δένδρου.

Βήμα 2: Συγκρίνει όλα τα επόμενα στοιχεία με τη ρίζα και όσα είναι μεγαλύτερα από την τιμή της ρίζας τους δίνεται τιμή επόμενου επιπέδου 2, ενώ αντίστοιχα όσα είναι μικρότερα τους δίνεται τιμή επόμενου επιπέδου 1 και η σημαία της ρίζας παίρνει τιμή 1 ώστε να μην συμπεριληφθεί στις επόμενες συγκρίσεις.

Βήμα 3: Προσπελαύνει το επόμενο στοιχείο. Η σημαία του στοιχείου γίνεται 1 και γίνεται σύγκριση του στοιχείου με όλα τα υπόλοιπα που ανήκουν στα **ίδια επίπεδα** και η σημαία τους είναι 0**.** Όσα είναι μεγαλύτερα από την τιμή του στοιχείου θα πάρουν τιμή επόμενου επιπέδου 2 ενώ όσα είναι μικρότερα θα πάρουν τιμή επόμενου επιπέδου 1. Το βήμα αυτό πραγματοποιείται με την χρήση δυναμικού αλφαριθμητικού και διαδοχικών ερωτημάτων SQL.

Βήμα 4: Το βήμα 3 επαναλαμβάνεται μέχρι να φθάσουμε στο τελευταίο στοιχείο, οπού ολοκληρώνεται η δημιουργία της δενδρικής δομής.

Υλοποίηση του Quad – Tree:

Βήμα 1: Ο αλγόριθμος Quad – Tree βρίσκει αρχικά τα μέγιστα X_{max} και $Y_{max.}$ Ορίζει σαν ελάχιστο X_{min} και Y_{min} το μηδέν.

Βήμα 2: Ορισμός του πρώτου επιπέδου.

Όσα στοιχεία ανήκουν στο διάστημα $X_{max}/2 \leq X \leq X_{max}$ και $Y_{max}/2 \leq Y \leq Y_{max}$ τότε παίρνουν τιμή 1.

Όσα στοιχεία ανήκουν στο διάστημα $X_{min} \leq X < X_{max}/2$ και $Y_{max}/2 \leq Y \leq Y_{max}$ τότε παίρνουν τιμή 2.

Όσα στοιχεία ανήκουν στο διάστημα $X_{min} \leq X \leq X_{max}/2$ και $Y_{min} \leq Y < Y_{max}/2$ τότε παίρνουν τιμή 3.

Όσα στοιχεία ανήκουν στο διάστημα $X_{max}/2 \leq X \leq X_{max}$ και $Y_{min} \leq Y < Y_{max}/2$ τότε παίρνουν τιμή 4.

Στο τέλος του βήματος 2 ορίζονται τα νέα X_{min}, X_{max}, Y_{min}, και Y_{max}. Η διαδικασία ορισμού τους είναι η ακόλουθη:

Όσα σημεία έχουν τιμή πρώτου επιπέδου 1 θα έχουν $X_{min} = X_{max}/2$, το X_{max} παραμένει σταθερό και το $Y_{min} = Y_{max}/2$, το Y_{max} παραμένει σταθερό.

Όσα σημεία έχουν τιμή πρώτου επιπέδου 2 θα έχουν το X_{min} σταθερό, το $X_{max} = X_{max}/2$ και το $Y_{min} = Y_{max}/2$, το Y_{max} παραμένει σταθερό.

Όσα σημεία έχουν τιμή πρώτου επιπέδου 3 θα έχουν το X_{min} σταθερό, το $X_{max} = X_{max}/2$ και το Y_{min} παραμένει σταθερό, το $Y_{max} = Y_{max}/2$.

Όσα σημεία έχουν τιμή πρώτου επιπέδου 4 θα έχουν το $X_{min} = X_{max}/2$, το X_{max} παραμένει σταθερό και το Y_{min} παραμένει σταθερό, το $Y_{max} = Y_{max}/2$.

Βήμα 3: Προσπελαύνουμε το πρώτο στοιχείο Χ του πίνακα. Σε όσα στοιχεία ανήκουν στα ίδια επίπεδα με το στοιχείο Χ ορίζουμε τιμή επόμενου επιπέδου με κριτήριο τα X_{min}, X_{max}, Y_{min}, και Y_{max} κάθε στοιχείου. Στη συνέχεια ορίζουμε νέα X_{min}, X_{max}, Y_{min}, και Y_{max} όπως στο βήμα 2.

Βήμα 4: Προσπελαύνουμε το επόμενο στοιχείο και επαναλαμβάνουμε το βήμα 3 μέχρι κανένα από τα στοιχεία να μην ανήκει στο ίδιο επίπεδο, οπότε και ολοκληρώνεται η δημιουργία της δενδρικής δομής.

Ο πίνακας επιπέδων

Ανάλογα με τον αλγόριθμο που εκτελούμε, τα δεδομένα που παράγονται αποθηκεύονται στον πίνακα TQuad ή TBtree. Οι πίνακες αυτοί αποθηκεύουν και τα επίπεδα στα οποία ανήκει το κάθε σημείο. Ειδικότερα, στο BS - Tree τα επίπεδα παίρνουν τις τιμές ένα και δύο, ανάλογα με το αν είναι μικρότερα ή μεγαλύτερα από τον κόμβο – πατέρα. Στο Quad – Tree παίρνουν τις τιμές ένα, δύο, τρία ή τέσσερα ανάλογα με το τεταρτημόριο στο οποίο ανήκουν. Αυτό αποδεικνύεται πολύ χρήσιμο στον έλεγχο της ορθότητας των αποτελεσμάτων, για κάποιον που γνωρίζει το σωστό αποτέλεσμα και θέλει να το εξακριβώσει. Παρόλα αυτά, είναι δυνατό κάποιος να εξακριβώσει την ορθότητα των αποτελεσμάτων χρησιμοποιώντας τον πίνακα επιπέδων, χωρίς να γνωρίζει εξαρχής το σωστό αποτέλεσμα. Αυτό προϋποθέτει όμως γνώση της λειτουργίας των αλγορίθμων αλλά και της εφαρμογής.

4.1.4 Αναζήτηση δεδομένων

Η εφαρμογή χρησιμοποιεί τρεις διαφορετικές τεχνικές αναζήτησης δεδομένων. Η φιλοσοφία κάθε αναζήτησης είναι διαφορετική όπως και το ερώτημα που τίθεται για απάντηση.

Αναζήτηση σημείου - Point Search:

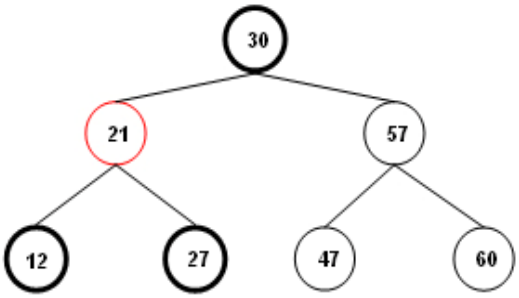
Η αναζήτηση σημείου χρησιμοποιείται μόνο από το Binary Search – Tree. Η αναζήτηση σημείου αποσκοπεί στον εντοπισμό του κόμβου που έχει μια δεδομένη τιμή Χ (σημείο). Ξεκινώντας από τη ρίζα, σε κάθε κόμβο γίνεται σύγκριση της τιμής Χ με το περιεχόμενο του κόμβου. Αν βρεθούν ίσα η αναζήτηση σταματά. Αν η τιμή είναι μικρότερη της ρίζας η αναζήτηση συνεχίζεται στο αριστερό υποδένδρο αλλιώς συνεχίζεται στο δεξιό υποδένδρο. Ο αλγόριθμος επίσης σταματά αν φθάσουμε σε κάποιο φύλλο χωρίς να βρεθεί η τιμή Χ σε κάποιο κόμβο του δένδρου. Η απόδοση της αναζήτησης, εξαρτάται από την διάταξη του δυαδικού δένδρου αν είναι δηλαδή ισορροπημένο ή όχι.

Αναζήτηση κοντινότερου γείτονα – Nearest Neighbor Search:

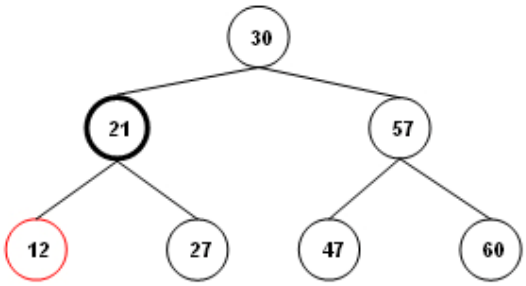
Δοθέντος ενός σημείου Χ, βρες όλα τα σημεία που απέχουν ελάχιστη απόσταση από το Χ. Η αναζήτηση κοντινότερου γείτονα χρησιμοποιείται και από τους δύο αλγόριθμους. Όπως φανερώνει και το όνομα της, αναζητούμε τους κοντινότερους γείτονες ενός σημείου. Συγκεκριμένα, στην περίπτωση του Binary Search – Tree, διακρίνουμε δύο περιπτώσεις:

- Εάν ο κόμβος είναι φύλλο (τερματικός) τότε ως κοντινότερο γείτονα η αναζήτηση εμφανίζει τον κόμβο – πατέρα.
- Εάν ο κόμβος δεν είναι φύλλο (μη τερματικός) τότε η αναζήτηση εμφανίζει ως κοντινότερο γείτονα όλους τους κόμβους - παιδία αν υπάρχουν και τον κόμβο πατέρα.

Ακολουθεί ένα παράδειγμα αναζήτησης κοντινότερου γείτονα σε Binary Search – Tree. Με κόκκινο σημαδεύεται ο κόμβος στον οποίο εφαρμόζουμε αναζήτηση κοντινότερου γείτονα και με έντονο μαύρο τα αποτελέσματα της.

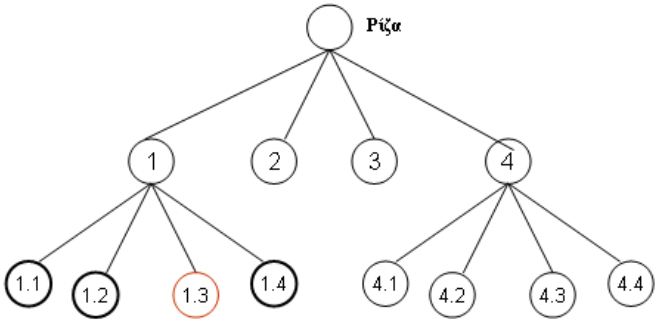


Εικόνα 29: Αναζήτηση σε μη τερματικό κόμβο



Εικόνα 30: Αναζήτηση σε τερματικό κόμβο

Στην περίπτωση του Quad – Tree τα δεδομένα αποθηκεύονται σε κόμβους – φύλλα. Η αναζήτηση εμφανίζει ως κοντινότερο γείτονα όλα τα φύλλα που ανήκουν στο ίδιο επίπεδο και έχουν κοινό κόμβο – πατέρα. Εάν δεν υπάρχουν κόμβοι – φύλλα στο ίδιο επίπεδο με το σημείο που επιλέχτηκε, δεν υπάρχει κοντινότερος γείτονας. Ακολουθεί ένα παράδειγμα αναζήτησης κοντινότερου γείτονα σε Quad – Tree. Με κόκκινο σημαδεύεται ο κόμβος στον οποίο εφαρμόζουμε αναζήτηση κοντινότερου γείτονα και με έντονο μαύρο τα πιθανά αποτελέσματα της.



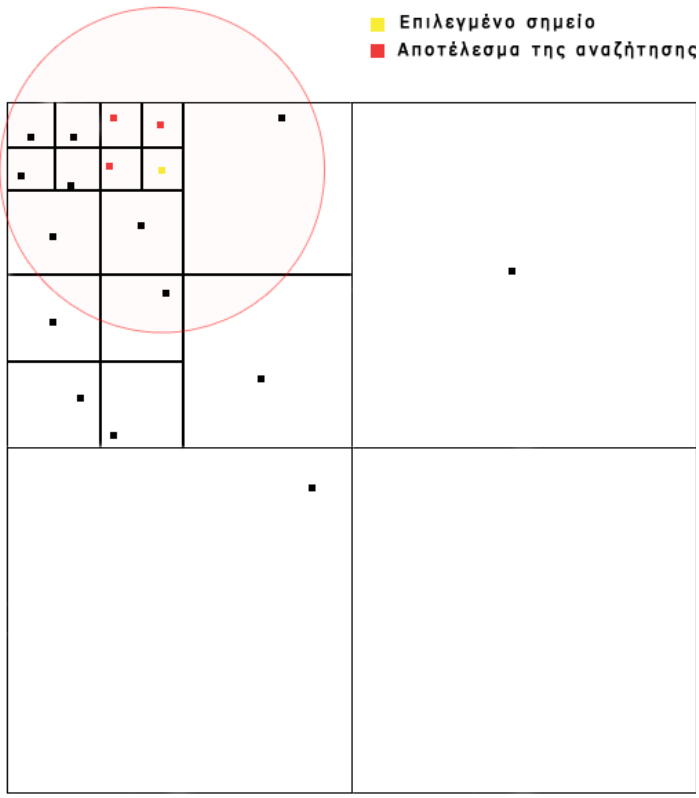
Εικόνα 31 : Αναζήτηση σε Quad - Tree

Η δενδρική δομή ενδείκνυται για αναζητήσεις κοντινότερου γείτονα καθώς μπορεί να εντοπίσει τους κοντινότερους γείτονες ενός σημείου ταχύτατα, λόγω της δομής της. Η απόδοση της αναζήτησης, δεν εξαρτάται από την διάταξη του δένδρου αν είναι δηλαδή ισορροπημένο ή όχι.

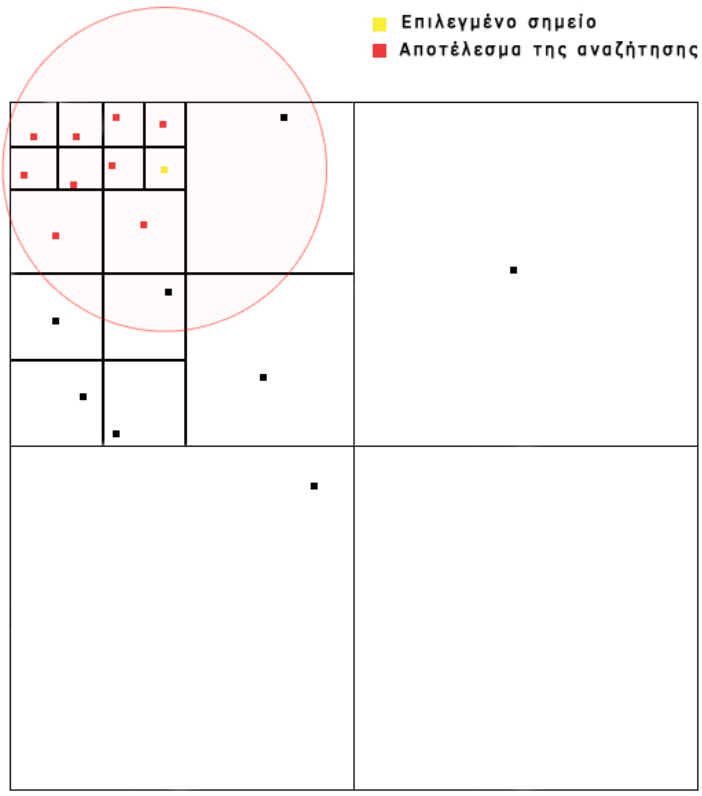
Αναζήτηση περιοχής ή ακτίνας (Range Search)

Δοθείσας μίας ακτίνας (περιοχής) r, βρες όλα τα αντικείμενα που περιέχονται από αυτήν. Η αναζήτηση περιοχής συναντάτε στην βιβλιογραφία και ως αναζήτηση ακτίνας ή εύρους. Η αναζήτηση περιοχής χρησιμοποιείται και από τους δύο αλγόριθμους. Ο τρόπος λειτουργίας της είναι κοινός και για τους δύο. Αρχικά, επιλέγεται ένα από τα υπάρχοντα σημεία, στη συνέχεια δίνεται η ακτίνα r και ο αριθμός των επιπέδων n. Η αναζήτηση θα εμφανίσει όλα τα σημεία που περιέχονται στην δοθείσα ακτίνα και ανήκουν στα ίδια επίπεδα. Η διαδικασία που εκτελεί ο αλγόριθμος είναι η ακόλουθη: Αρχικά, επιλέγονται όλα τα σημεία που έχουν κοινά τα n προηγούμενα επίπεδα. Στην συνέχεια, από τα σημεία που επιλέχθηκαν, εμφανίζονται εκείνα στα οποία η ευκλείδεια απόσταση τους από το σημείο που δόθηκε, είναι μικρότερη από την ακτίνα r.

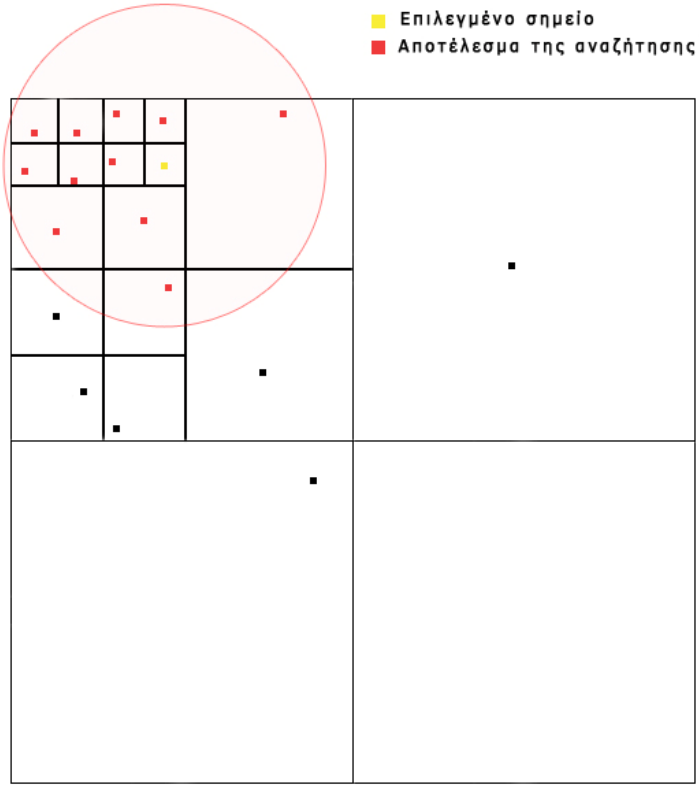
Για την καλύτερη κατανόηση της λειτουργίας της αναζήτηση περιοχής ακολουθεί ένα παράδειγμα. Η παρακάτω εικόνες είναι τεμαχισμένες σε τεταρτημόρια με βάση τον αλγόριθμο του Quad – Tree. Υποθέτουμε ότι κατά την αναζήτηση έχουμε μια σταθερή ακτίνα και αυξάνουμε τα επίπεδα κατά ένα σε κάθε επανάληψη. Με κίτρινο βλέπουμε το επιλεγμένο σημείο ενώ με κόκκινο το αποτέλεσμα της αναζήτησης.



Εικόνα 32: Αναζήτηση ακτίνας για αριθμό επίπεδων 1

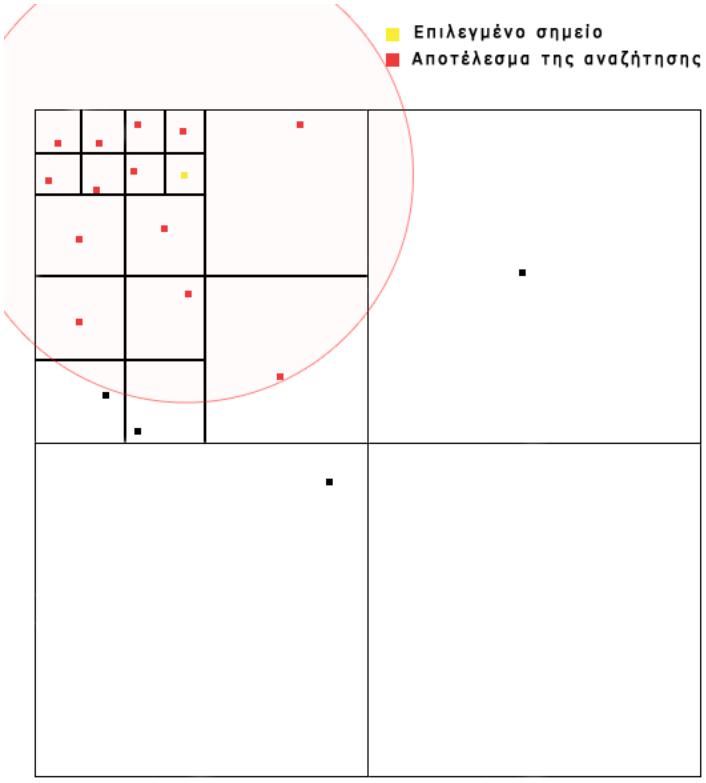


Εικόνα 33: Αναζήτηση ακτίνας για αριθμό επιπέδων 2



Εικόνα 34: Αναζήτηση ακτίνας για αριθμό επιπέδων 3

Στην περίπτωση που έχουμε επιλέξει αριθμό επιπέδων τρία, **εικόνα 34** και αυξήσουμε την ακτίνα το αποτέλεσμα φαίνεται στην **εικόνα 35**. Για το συγκεκριμένο παράδειγμα, επιλέγοντας αριθμό επιπέδων τρία, ουσιαστικά, η αναζήτηση ελέγχει όλα τα σημεία που ανήκουν στο πάνω αριστερά τεταρτημόριο της εικόνας. Αν αυξήσουμε ακόμα ένα επίπεδο, δηλαδή τέσσερα, τότε η αναζήτηση ελέγχει τα σημεία όλης της εικόνας.



Εικόνα 35:Αναζήτηση ακτίνας για αριθμό επιπέδων 3 και μεγαλύτερη ακτίνα

Η αναζήτηση περιοχής συναντάτε συχνά στους τομείς των GIS, της τοπογραφίας και χαρτογράφησης και είναι πολύ χρήσιμη. Για παράδειγμα, φανταστείτε να έπρεπε να εκτελέσουμε αναζήτηση ακτίνας σε ένα σύνολο πεντακοσίων σημείων, τότε θα έπρεπε να υπολογίσουμε την ευκλείδεια απόσταση για κάθε ένα από τα σημεία. Αντίθετα, στην αναζήτηση ακτίνας με βάση την δενδρική δομή επιλέγεται αρχικά ένα σύνολο σημείων και υπολογίζονται οι ευκλείδειες αποστάσεις μόνο γι' αυτό το σύνολο σημείων. Το αποτέλεσμα είναι να έχουμε λιγότερες συγκρίσεις , άρα και πολύ υψηλότερη απόδοση.

Η απόδοση της αναζήτησης, δεν εξαρτάται από την διάταξη του δένδρου αν είναι δηλαδή ισορροπημένο ή όχι.

4.2 Έλεγχος

Η υλοποίηση της εφαρμογής ολοκληρώνεται με το λεπτομερή έλεγχο. Στην πραγματικότητα, ο έλεγχος ήταν συνεχής καθ' όλη την διάρκεια δημιουργίας της εφαρμογής, ενώ γινόταν εκτενέστερος μετά την ολοκλήρωση ενός επιπέδου. Ο αριθμός των ελέγχων που πραγματοποιήσαμε είναι πολυάριθμος, ενώ στο κεφάλαιο αυτό παρουσιάζουμε τις πιο χαρακτηριστικές περιπτώσεις ελέγχου κάθε αλγόριθμου.

4.2.1 Μεθοδολογία ελέγχου

Ο έλεγχος της εφαρμογής πραγματοποιήθηκε με χρήση ομάδων τεχνητών συνόλων δεδομένων. Στον έλεγχο έγινε προσπάθεια να χρησιμοποιηθούν όλες οι λειτουργίες που προσφέρει η εφαρμογή καθώς και να καλυφθούν όλες οι περιπτώσεις ελέγχου της. Στο τέλος, κρίνεται η ισχύς των αποτελεσμάτων.

Συγκεκριμένα, έχουμε δημιουργήσει δύο ομάδες τεχνητών συνόλων δεδομένων:

- *προεπιλεγμένα δεδομένα :* δεδομένα τα οποία επιλέξαμε εμείς εκ' των προτέρων.
- *τυχαία δεδομένα* : δεδομένα τα οποία παρήγαμε τυχαία με την εξίσωση Rand.

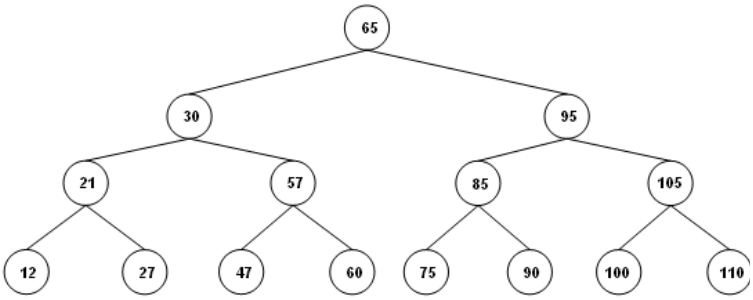
4.2.2 Αναλυτική παρουσίαση ελέγχου

Η ορθότητα των αποτελεσμάτων που εμφανίζει η εφαρμογή μπορεί να ελεγχθεί εύκολα και με ακρίβεια, χρησιμοποιώντας των πίνακα επιπέδων της εφαρμογής.

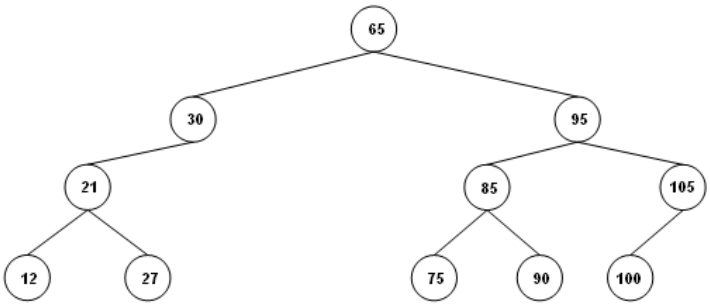
Επιλέξαμε αρχικά να χρησιμοποιήσουμε "προεπιλεγμένα δεδομένα" για τον έλεγχο της εφαρμογής. Ο λόγος ήταν ότι για τα δεδομένα αυτά γνωρίζαμε εκ 'των προτέρων τα σωστά αποτελέσματα (έχοντας κάνει πράξεις με το χέρι) και έτσι είχαμε την δυνατότητα να συγκρίνουμε τα αποτελέσματα με ασφάλεια.

Για το BS - Tree χρησιμοποιήσαμε τρεις ομάδες προεπιλεγμένων δεδομένων:
Το πρώτο σύνολο σχηματίζει ένα μη ισορροπημένο δυαδικό δένδρο αναζήτησης
Το δεύτερο σύνολο σχηματίζει ένα ισορροπημένο δυαδικό δένδρο αναζήτησης
Το τρίτο σύνολο σχηματίζει ένα λοξό δένδρο.

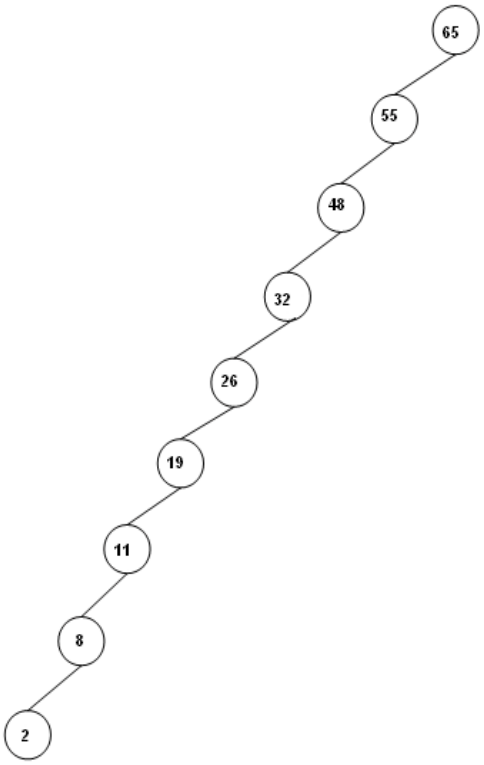
Τα δένδρα που σχηματίζονται από τα τεχνητά σύνολα δεδομένων φαίνονται στις παρακάτω εικόνες:



Εικόνα 36: Το ισορροπημένο δυαδικό δένδρο



Εικόνα 37: Το μη ισορροπημένο δυαδικό δένδρο



Εικόνα 38: Το λοξό δυαδικό δένδρο

Τα αποτελέσματα τόσο της δημιουργίας των δενδρικών δομών όσο και των αναζητήσεων που εφαρμόσαμε στα τρία παραπάνω σύνολα δεδομένων, ήταν σωστά.

Δοκιμάσαμε επίσης πέντε σύνολα τυχαίων δεδομένων με 40, 200, 300, 400 και 500 σημεία. Το πρώτο σύνολο το επιλέξαμε ώστε να μπορούμε να ελέγξουμε την ορθότητα της δενδρικής δομής αλλά και των αποτελεσμάτων της αναζήτησης (κάνοντας πράξεις με το χέρι). Τα επόμενα σύνολα τα επιλέξαμε για να ελέγξουμε την ακρίβεια των αποτελεσμάτων σε μεγαλύτερα σύνολα αριθμών. Σε όλες τις περιπτώσεις τα αποτελέσματα της εφαρμογής ήταν σωστά.

4.3 Απόδοση των υλοποιημένων αλγόριθμων

Αφού η εφαρμογή πέρασε με επιτυχία τον έλεγχο ήρθε η ώρα να αντλήσουμε ορισμένα πειραματικά αποτελέσματα και να κρίνουμε την συμπεριφορά και την απόδοση της. Το σύστημα στο οποίο κάναμε τις δοκιμές βασίζεται σε έναν AMD Athlon 64Bit με συχνότητα λειτουργίας 3200GHz, 512 MB μνήμη, και λειτουργικό σύστημα τα MS Windows XP. Για να πάρουμε τις μετρήσεις χρησιμοποιήσαμε πέντε σύνολα τυχαίων δεδομένων. Οι τιμές των δεδομένων επηρεάζουν την απόδοση των αλγορίθμων διαφορετικά κάθε φορά. Για να έχουμε μια πιο αντικειμενική εικόνα της απόδοσης των αλγορίθμων επαναλάβαμε την διαδικασία πέντε φορές με διαφορετικά σύνολα τυχαίων δεδομένων και αντλήσαμε τα συμπεράσματα από το μέσο όρο των αποτελεσμάτων.

Πίνακα απόδοσης Binary Search - Tree

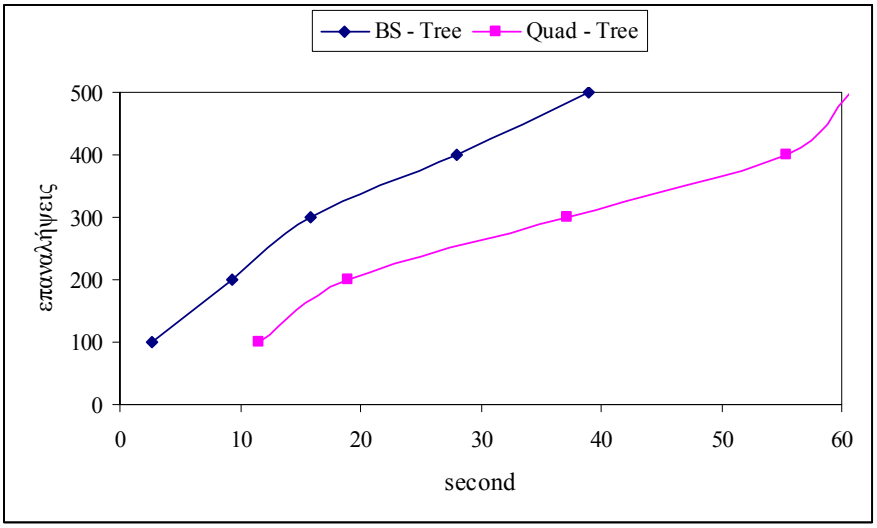
Αριθμός Τυχαίων δεδομένων	1η Επανάληψη	2η Επανάληψη	3η Επανάληψη	4η Επανάληψη	5η Επανάληψη	Μέση τιμή
100	3 sec	3 sec	3 sec	2 sec	2 sec	2.6 sec
200	10 sec	8 sec	10 sec	9 sec	10 sec	9.4 sec
300	17 sec	16 sec	16 sec	15 sec	15 sec	15.8 sec
400	28 sec	28 sec	29 sec	27 sec	28 sec	28 sec
500	40 sec	40 sec	39 sec	38 sec	38 sec	39 sec

Πίνακας απόδοσης Quad-Tree

Αριθμός Τυχαίων δεδομένων	1η Επανάληψη	2η Επανάληψη	3η Επανάληψη	4η Επανάληψη	5η Επανάληψη	Μέση τιμή
100	12 sec	13 sec	11 sec	10 sec	12 sec	11.6 sec
200	19 sec	20 sec	14 sec	24 sec	18 sec	19 sec
300	35 sec	35 sec	36 sec	41 sec	39 sec	37.2 sec
400	57 sec	53 sec	55 sec	55 sec	57 sec	55.4 sec
500	57 sec	62 sec	60 sec	62 sec	63 sec	60.8 sec

Από τους χρόνους δημιουργίας της δενδρικής δομής το συμπέρασμα που προκύπτει είναι το προφανές, όσο περισσότερα τα σημεία τόσο περισσότερος χρόνος απαιτείται για την δημιουργίας της δομής. Επιπλέον, μπορούμε να συμπεράνουμε ότι η κατανομή των σημείων επηρεάζει την διάταξη του δένδρου άρα και την απόδοση του αλγόριθμου δημιουργίας της δομής. Για παράδειγμα, στον πίνακα του Quad – Tree στην πρώτη επανάληψη οι χρόνοι δημιουργίας της δενδρικής δομής είναι η ίδιοι για 400 και 500 σημεία. Αυτό σημαίνει ότι είχαμε κακή κατανομή των σημείων και χαμηλή απόδοση για 400 σημεία ενώ αντίθετα για 500 σημεία είχαμε καλή κατανομή σημείων άρα και υψηλή απόδοση. Τα ίδια συμπεράσματα εξάγουμε και από τον πίνακα απόδοσης του Binary Search – Tree. Όπως παρατηρούμε και από το παρακάτω διάγραμμα, η δημιουργία της δομής του Binary Search – Tree είναι γρηγορότερη συγκριτικά με αυτή του Quad – Tree. Θα πρέπει να λάβουμε υπόψη το γεγονός ότι τα δεδομένα που επεξεργάζεται η δομή του Binary Search – Tree είναι μονοδιάστατα ενώ του Quad – Tree είναι δύο διαστάσεων.

Τα αποτελέσματα των αναζητήσεων ήταν εντυπωσιακά. Ακόμα και στην περίπτωση των πεντακοσίων σημείων (όπως και σε δοκιμές με περισσότερα σημεία) ο χρόνος απόκρισης των ερωτημάτων ήταν της τάξης των milli – second. Μόνο στην περίπτωση της αναζήτησης περιοχής και εφόσον έχουμε επιλέξει τον μέγιστο αριθμό επιπέδων, παρατηρείται μια σχετική καθυστέρηση αν έχουμε μεγάλο πλήθος σημείων. Αυτό οφείλεται στο γεγονός ότι η αναζήτηση ελέγχει όλα τα σημεία του πίνακα και ουσιαστικά έχουμε τον εκφυλισμό της αναζήτησης περιοχής σε μια αναζήτηση με βάση την ευκλείδεια απόσταση. Η υψηλή απόδοση είναι και ο λόγος, για το οποίο οι δενδρικές δομές ενδείκνυνται για αυτού του τύπου τα ερωτήματα.



Διάγραμμα απόδοση δημιουργίας δενδρικών δομών

ΚΕΦΑΛΑΙΟ 5

ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ – MANUAL

Καλώς ήλθατε στο πρόγραμμα Application for data classification. Το πρόγραμμα αυτό έχει ως στόχο να σας εισάγει στους τομείς των δενδρικών δομών δεδομένων και ιδιαίτερα στον τομέα της αναζήτησής με βάση αλγόριθμους ταξινόμησης που στηρίζονται στις δενδρικές δομές Binary Search – Tree και Quad – Tree. Ιδιαίτερη έμφαση δόθηκε στον τομέα της σχεδίασης της εφαρμογής ώστε να είναι ευχάριστη και λειτουργική. Ο χρήστης μπορεί να εκτελέσει οποιαδήποτε λειτουργία ακολουθώντας τρία απλά βήματα (1 2 3) με τη σειρά. Η εφαρμογή δεν επιτρέπει την εκτέλεση των βημάτων με λανθασμένη σειρά.

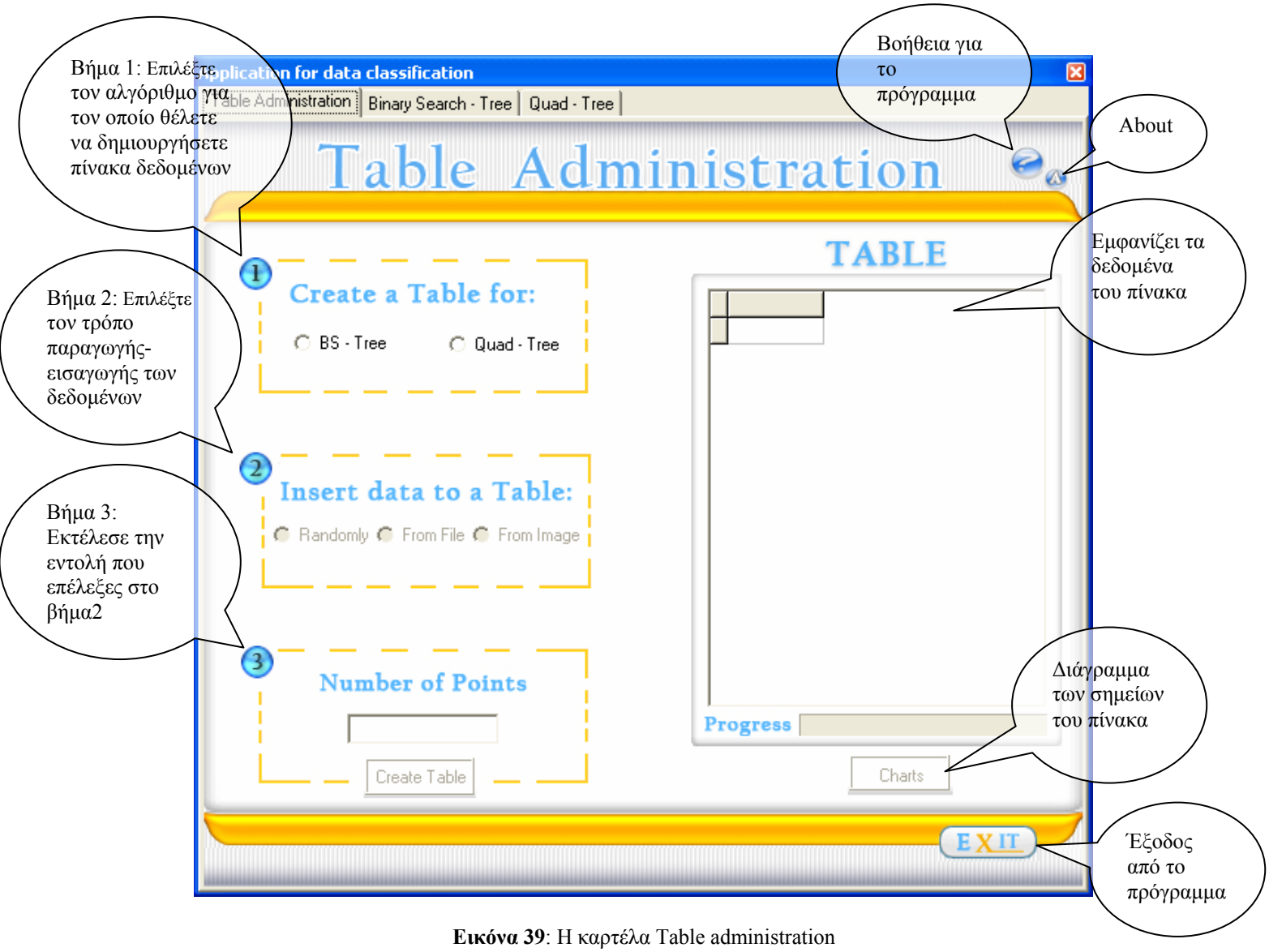
5.1 Καρτέλα 1: Διαχείριση Πινάκων (Table Administration)

Η καρτέλα 1 είναι η πρώτη καρτέλα που εμφανίζεται όταν ξεκινάμε την εφαρμογή και είναι υπεύθυνη για την παραγωγή και είσοδο των δεδομένων στην εφαρμογή.

Υπάρχουν τρεις τρόποι παραγωγής - εισόδου των δεδομένων στην εφαρμογή:

1. Παραγωγή τυχαίων δεδομένων.
2. Εισαγωγή δεδομένων από αρχείο.
3. Εισαγωγή δεδομένων από εικόνα (μόνο για Quad - Tree).

Αρχικά, θα εξηγήσουμε τις βασικές λειτουργίες των κουμπιών της καρτέλα Table administration, κάποιες από τις οποίες είναι κοινές σε όλες τις καρτέλες, όπως για παράδειγμα το help button.

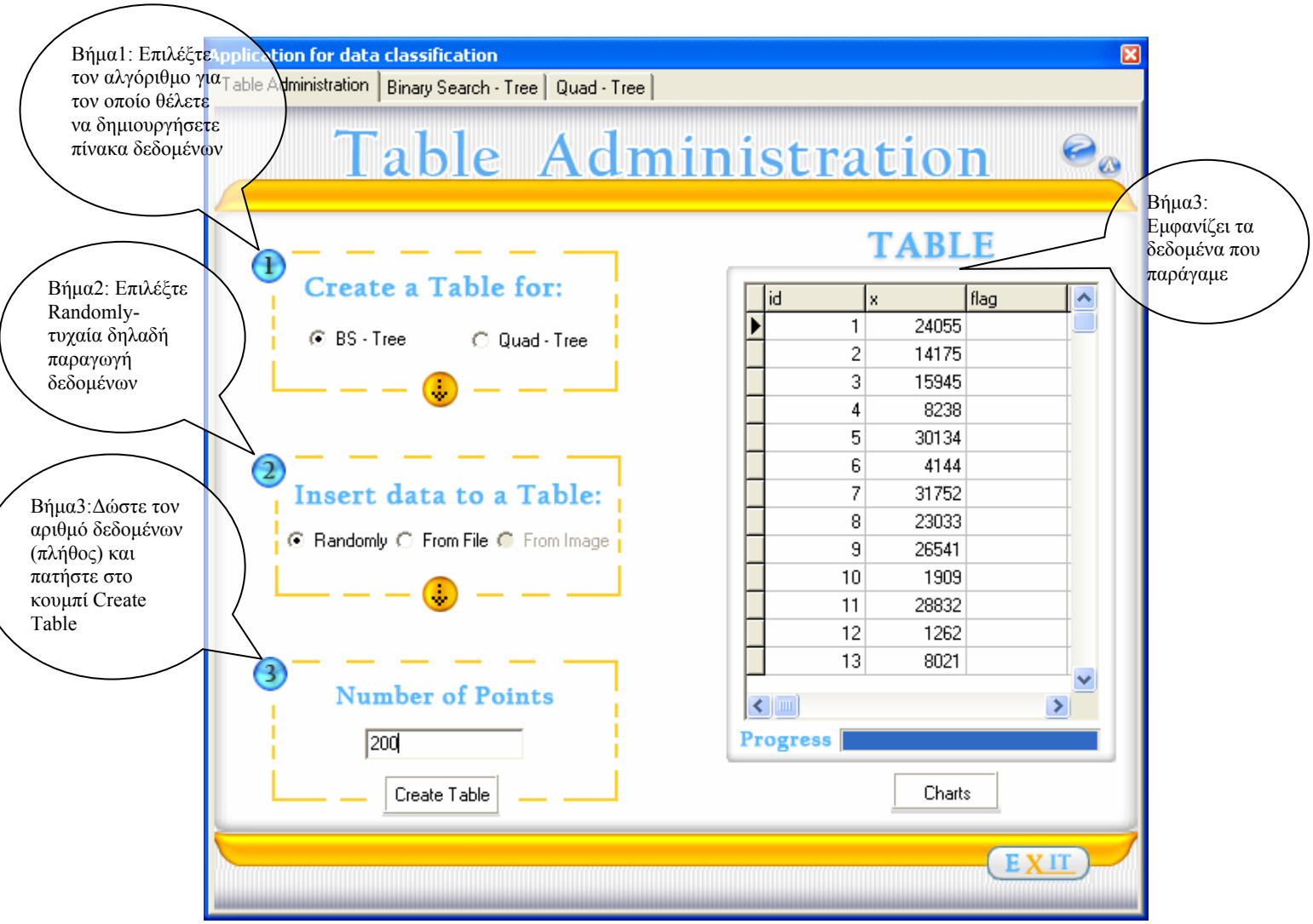


Εικόνα 39: Η καρτέλα Table administration

Ακολουθεί αναλυτικά, η περιγραφή της κάθε διαδικασίας παραγωγής – εισόδου των δεδομένων στην εφαρμογή , βήμα προς βήμα.

5.1.1 Παραγωγή τυχαίων δεδομένων

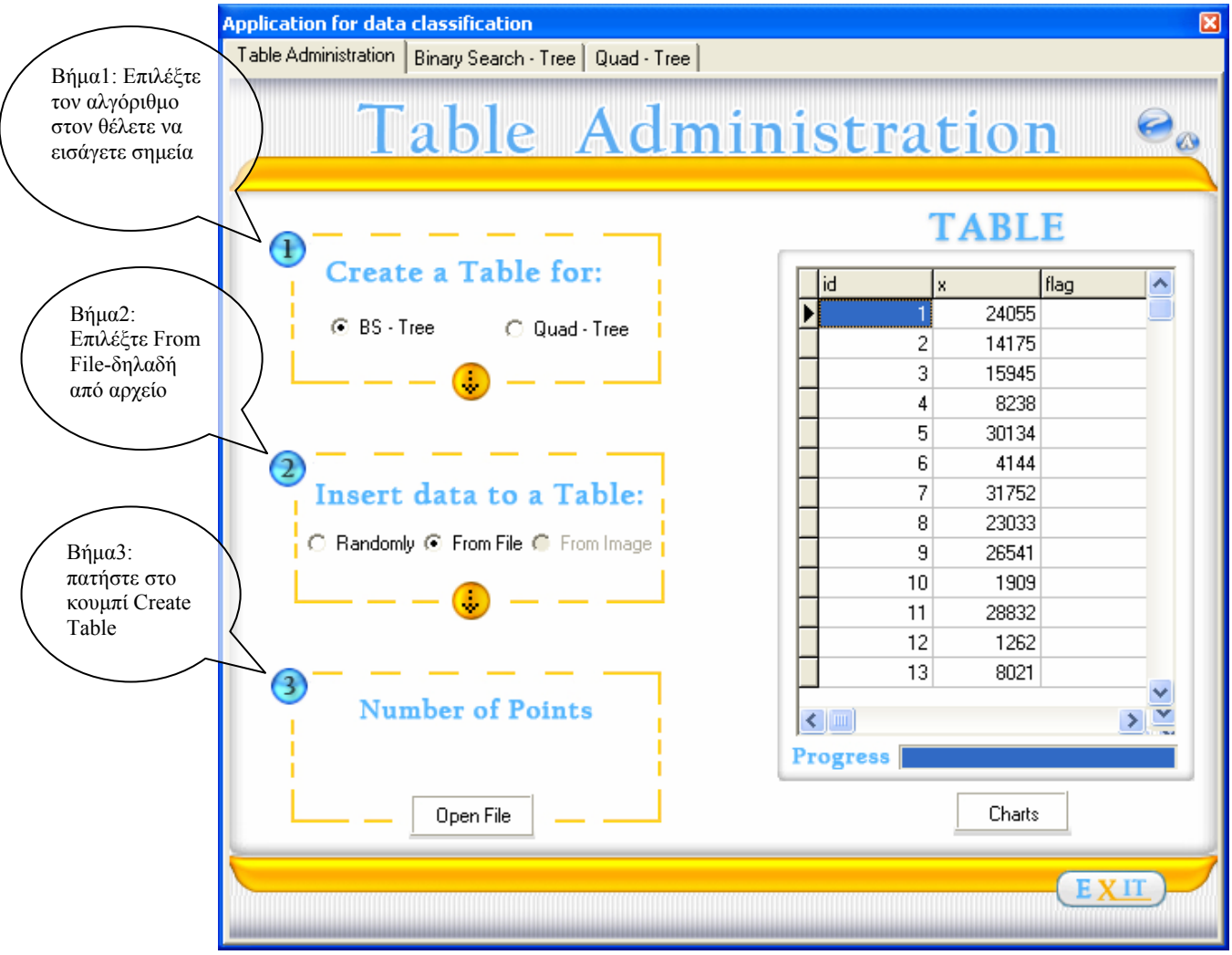
Τα βήματα της διαδικασίας παραγωγής τυχαίων δεδομένων φαίνονται στην **εικόνα 40**.



Εικόνα 40: Βήματα για την παραγωγή τυχαίων δεδομένων

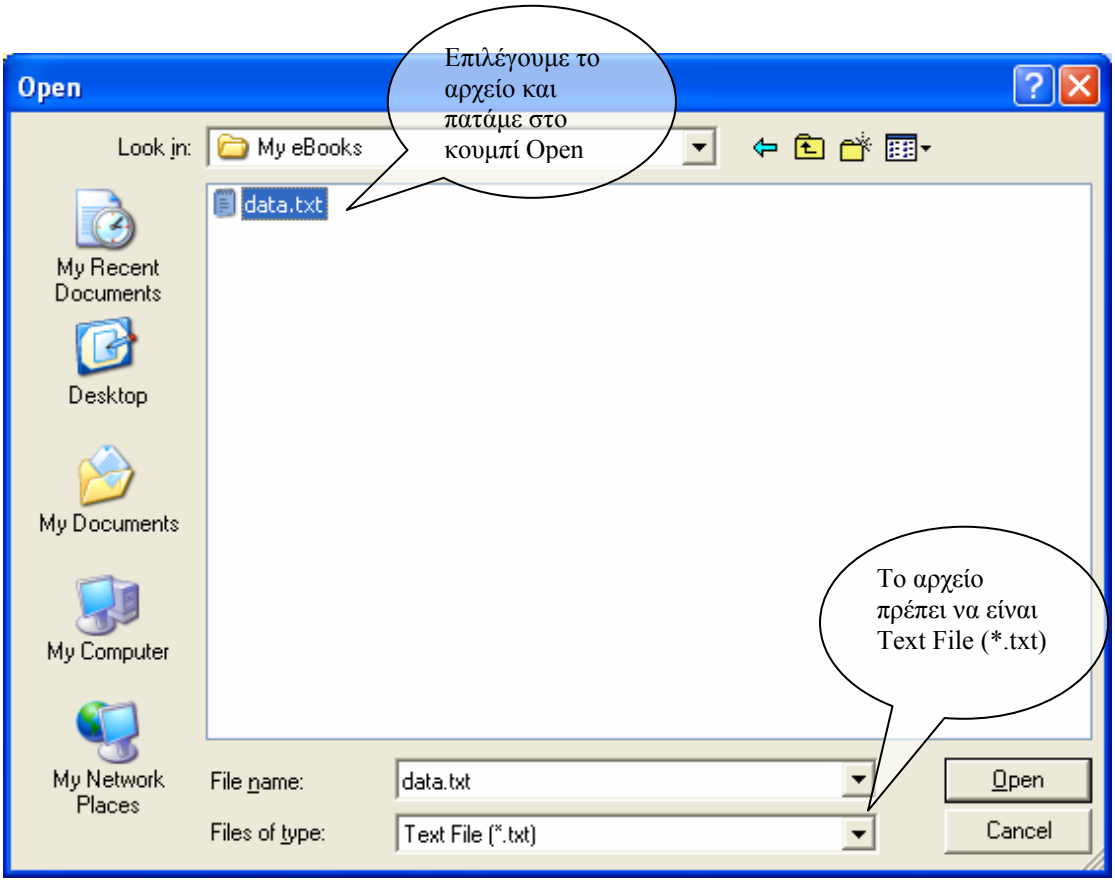
5.1.2 Εισαγωγή δεδομένων από αρχείο

Τα βήματα της διαδικασίας εισαγωγής δεδομένων από αρχείο κειμένου, φαίνονται στην **εικόνα 41**.



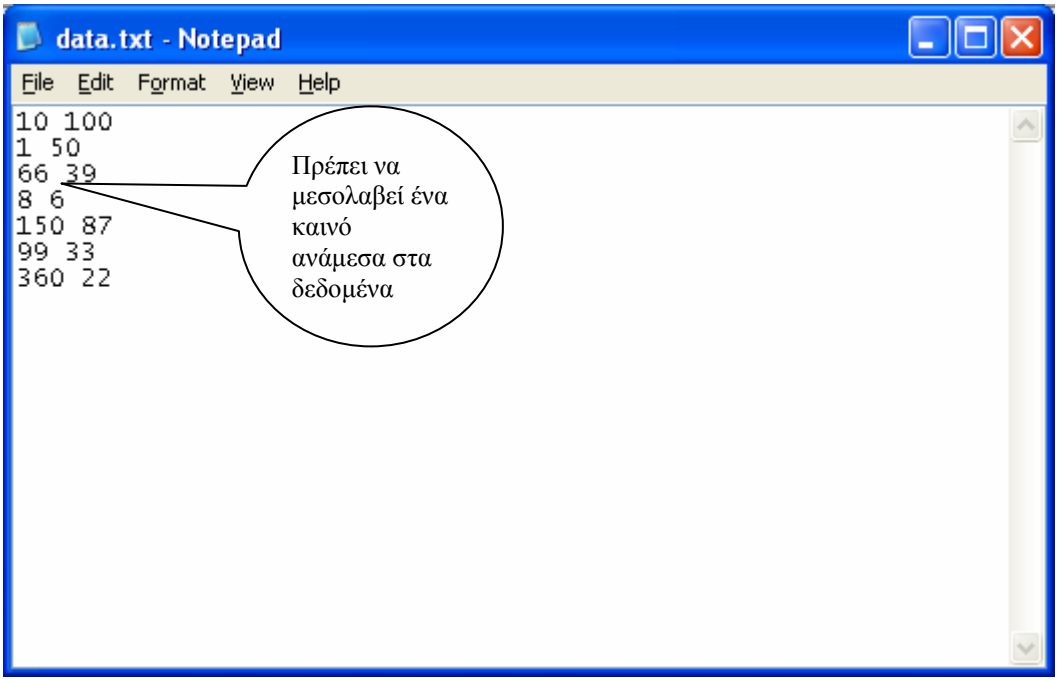
Εικόνα 41: Βήματα για την εισαγωγή δεδομένων από αρχείο

Πατώντας στο κουμπί "Open File" θα ανοίξει το παρακάτω πλαίσιο διαλόγου για την επιλογή του αρχείου, **εικόνα 42**.



Εικόνα 42: Το πλαίσιο για την εισαγωγή του αρχείου

Το αρχείο πρέπει να είναι Text File (*.txt) και τα δεδομένα πρέπει να έχουν την παρακάτω μορφή, **εικόνα 43**.

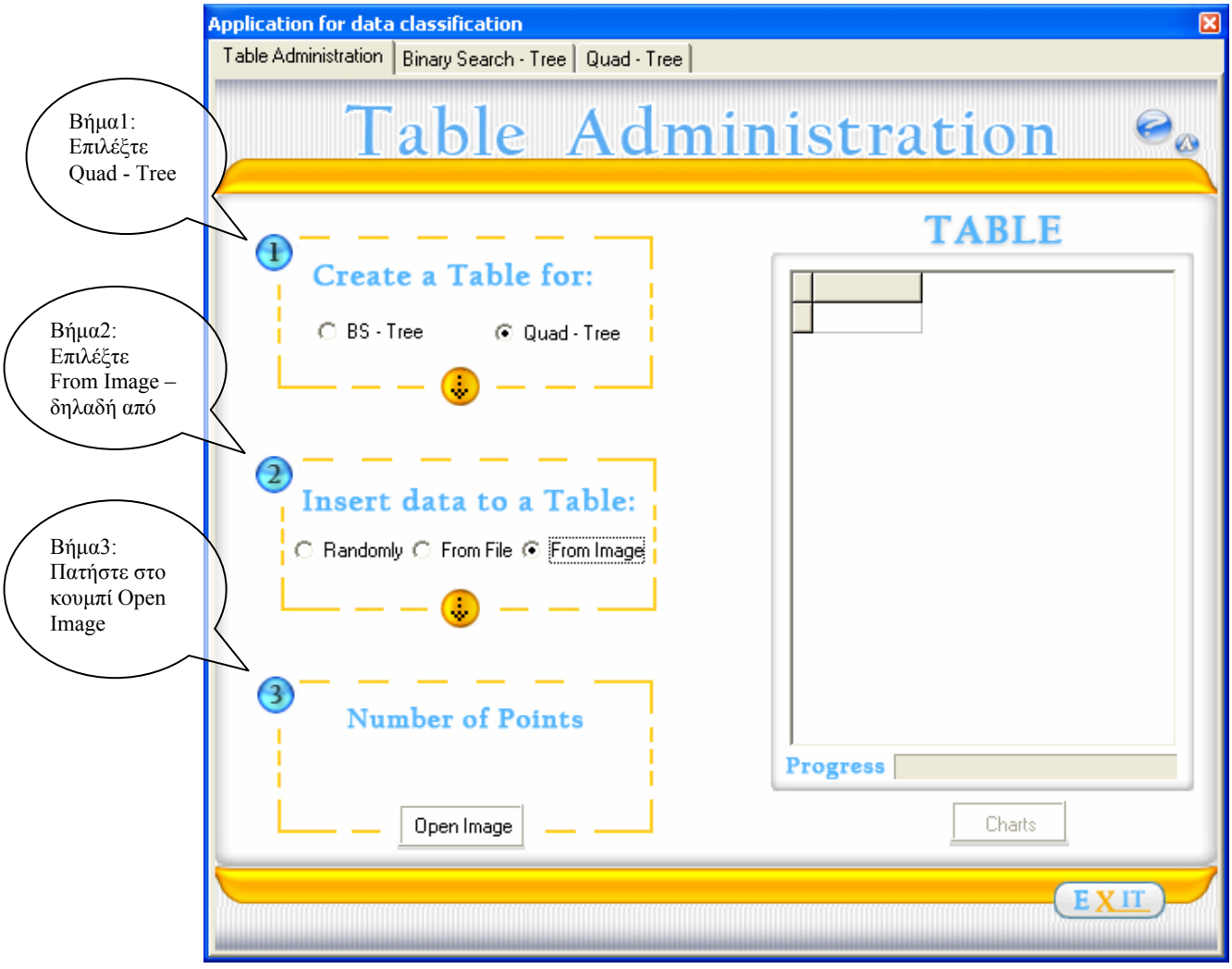


Εικόνα 43: Η μορφή των δεδομένων στο αρχείο txt

Μετά την ολοκλήρωση της διαδικασίας που περιγράψαμε, τα δεδομένα θα εμφανιστούν στον πίνακα της καρτέλας 1 (Table administration). Στο Binary Search – tree, εισάγουμε την μια τιμή κάτω από την άλλη δημιουργώντας μια στήλη, αφού ο πίνακας είναι μονοδιάστατος.

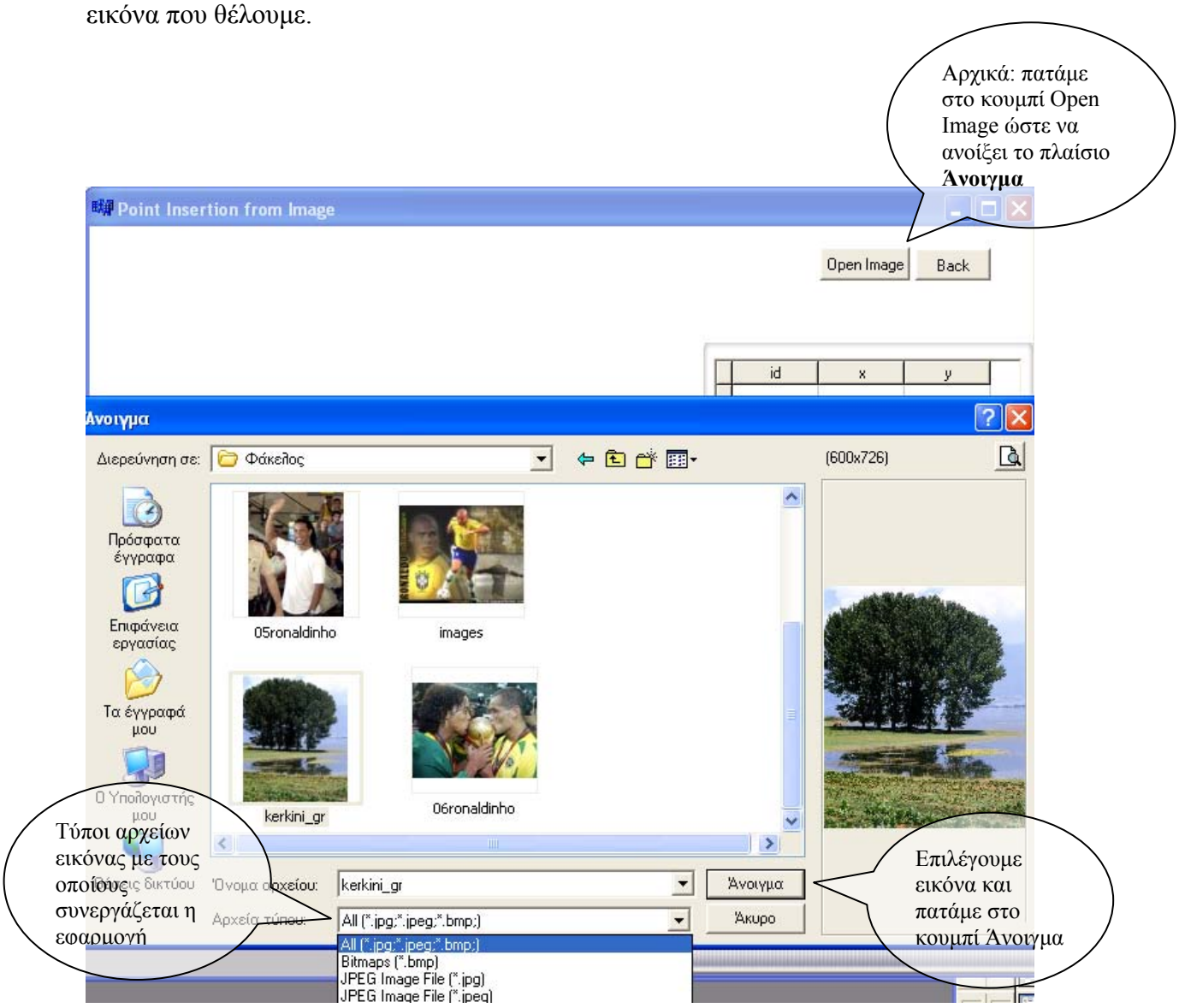
5.1.3 Εισαγωγή δεδομένων από εικόνα

Τα βήματα της διαδικασίας εισαγωγής δεδομένων από εικόνα, φαίνονται στην **εικόνα 44**.



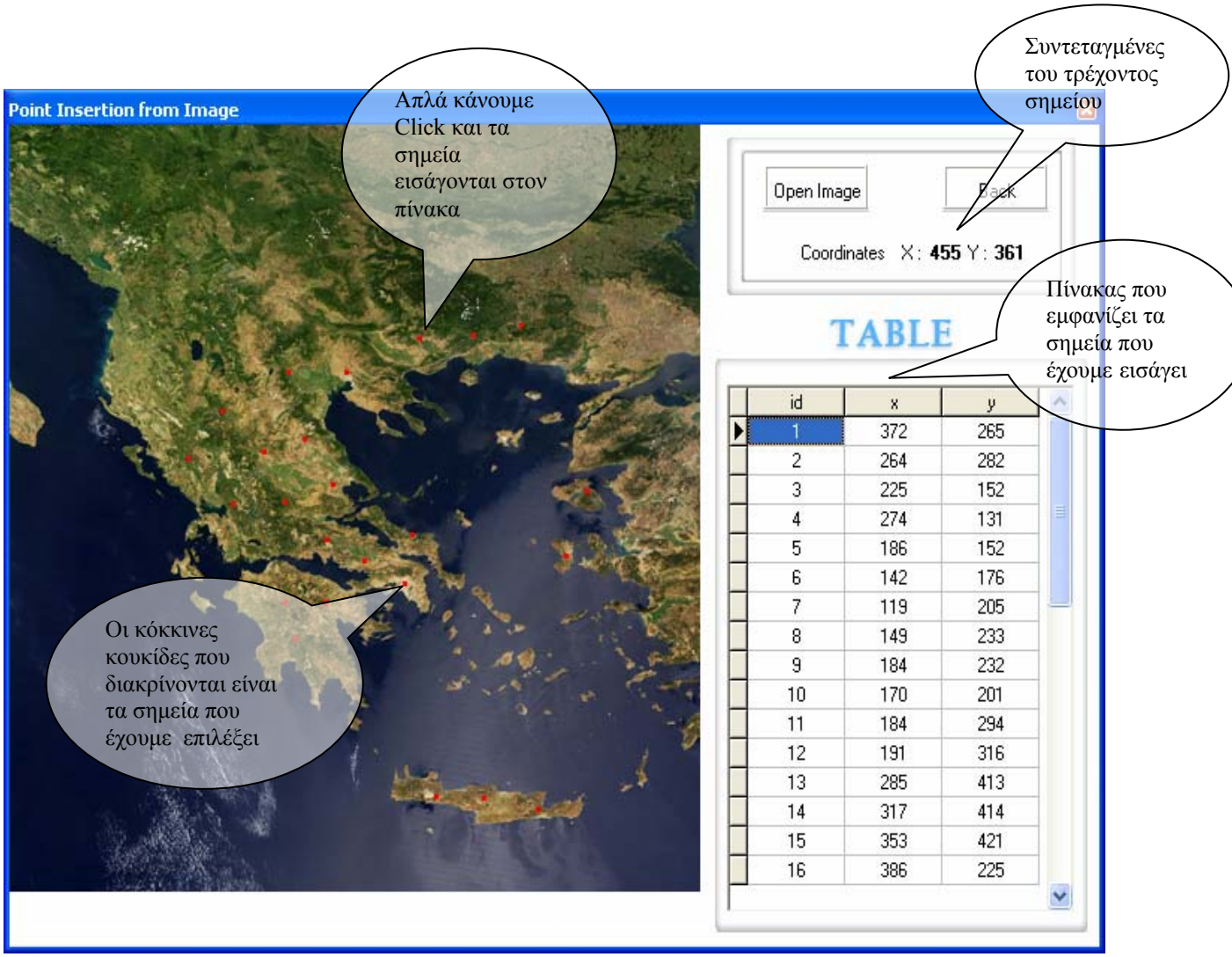
Εικόνα 44: Βήματα για την εισαγωγή δεδομένων από εικόνα

Στην συνέχεια, πατώντας στην καρτέλα που θα εμφανιστεί στο κουμπί Open Image, θα ανοίξει το πλαίσιο της **εικόνας 45**, ώστε να επιλέξουμε και να εισάγουμε την εικόνα που θέλουμε.



Εικόνα 45: Πλαίσιο για την εισαγωγή εικόνας.

Αφού επιλέξουμε και ανοίξουμε μια εικόνα θα εμφανιστεί η καρτέλα της **εικόνας 46**. Στο σημείο αυτό εμείς απλά κάνουμε click στο σημείο της εικόνας που επιθυμούμε να εισάγουμε στον πίνακα δεδομένων. Αν η εικόνα είναι τύπου Bitmap(*.bmp) το σημείο που επιλέγουμε σημαδεύεται με κόκκινο.



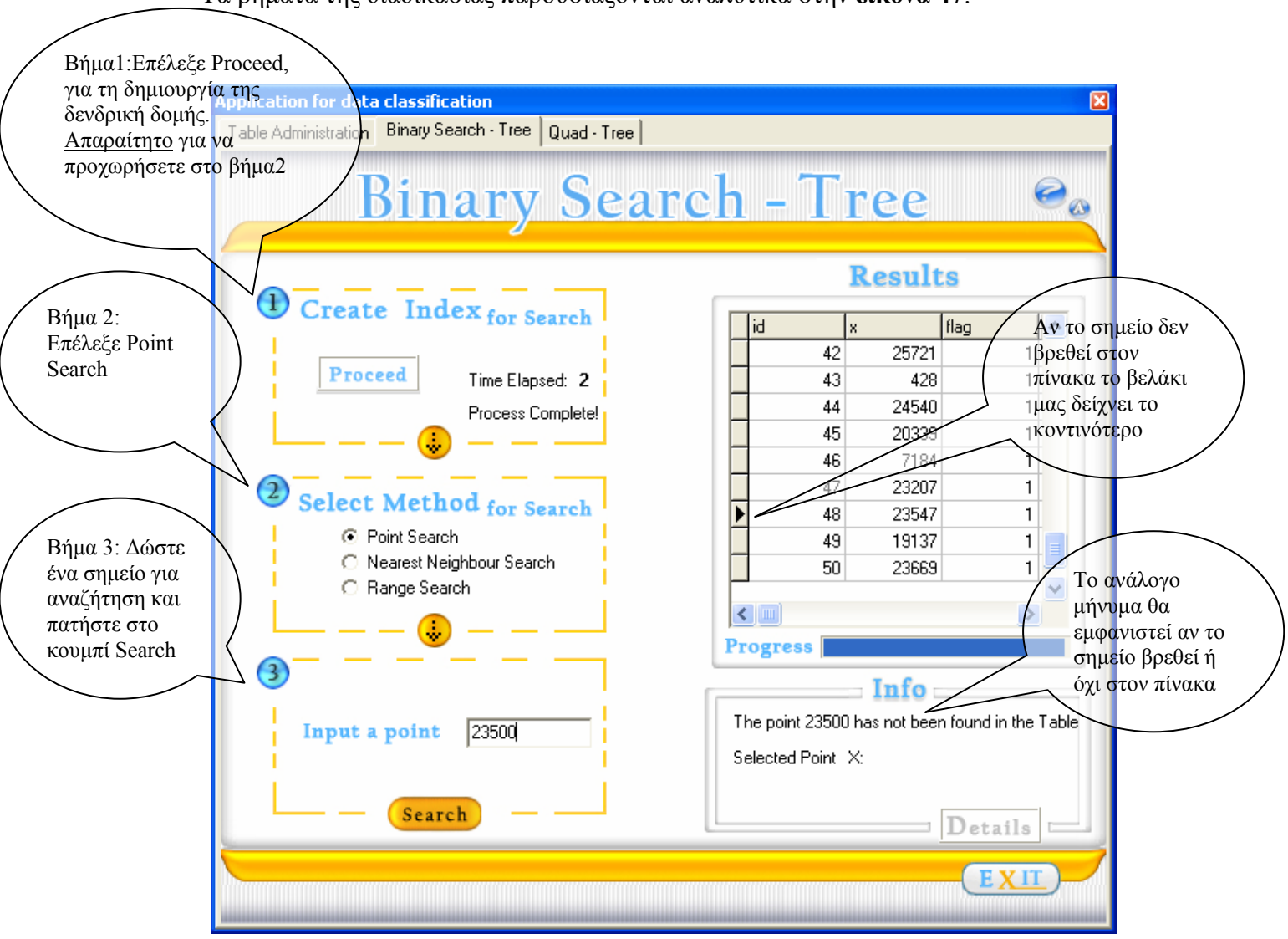
Εικόνα 46: Πλαίσιο για την εισαγωγή δεδομένων από εικόνα

5.2 Καρτέλα 2 : Binary Search – Tree

Η καρτέλα 2 περιέχει όλα τα αντικείμενα που έχουν σχέση με το Binary Search – Tree. Το πρώτο βήμα (δημιουργία της δενδρικής δομής), είναι κοινό για όλες τις αναζητήσεις γι' αυτό το λόγω και προχωράμε κατευθείαν στην εξήγηση του τρόπου λειτουργίας των αναζητήσεων, επεξηγώντας την διαδικασία βήμα - βήμα.

5.2.1 Αναζήτηση σημείου - Point Search

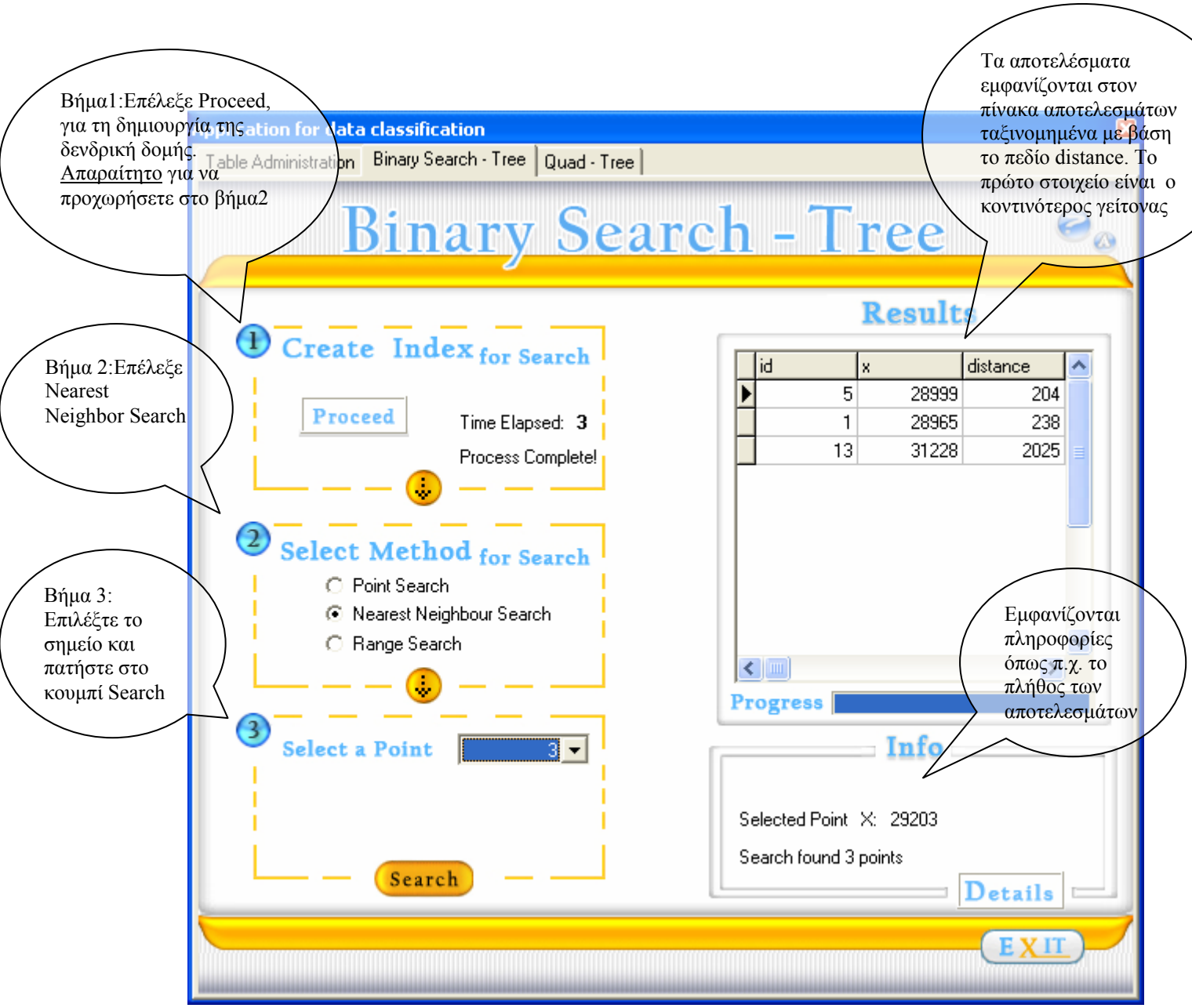
Στην αναζήτηση αυτή ο χρήστης εισάγει ένα σημείο και η εφαρμογή το αναζητάει με βάση την δενδρική δομή που έχει δημιουργήσει. Αν το σημείο περιέχεται στα σημεία που έχουμε εισάγει στην εφαρμογή εμφανίζεται ανάλογο μήνυμα στο πλαίσιο "Info". Αν το σημείο δεν υπάρχει εμφανίζεται στον πίνακα "Results" το κοντινότερο σε αυτό. Τα βήματα της διαδικασίας παρουσιάζονται αναλυτικά στην **εικόνα 47**.



Εικόνα 47: Βήματα για την αναζήτηση Point Search

5.2.2 Αναζήτηση Κοντινότερου γείτονα – Nearest Neighbor Search

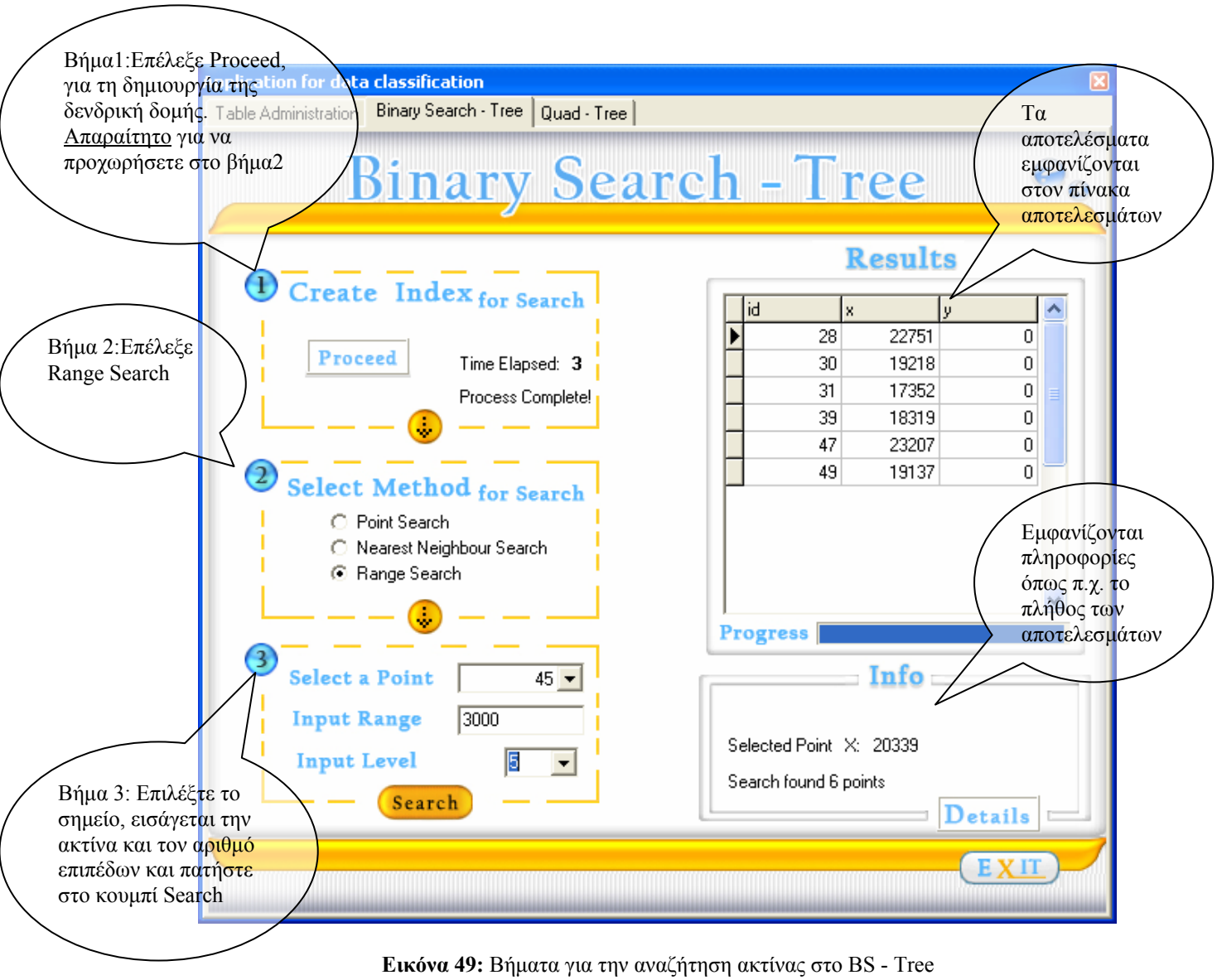
Η αναζήτηση αυτή εμφανίζει τους κοντινότερους γείτονες ενός σημείου με βάση την δενδρική δομή. Στον πίνακα "Results" εμφανίζονται τα αποτελέσματα της αναζήτησης, ταξινομημένα με βάση την απόσταση από το σημείο που επιλέξαμε. Η ταξινόμηση είναι με αύξουσα σειρά. Τα βήματα της διαδικασίας παρουσιάζονται αναλυτικά στην **εικόνα 48**.



Εικόνα 48: Βήματα για την αναζήτηση κοντινότερου γείτονα στο BS - Tree

5.2.3 Αναζήτηση περιοχής – Range Search

Η αναζήτηση περιοχής ή ακτίνας εμφανίζει όλα τα σημεία που περιέχονται στην ακτίνα (περιοχή) που εισάγουμε, με βάση την δενδρική δομή (επίπεδα του δένδρου). Τα βήματα της διαδικασίας παρουσιάζονται αναλυτικά στην **εικόνα 49**.



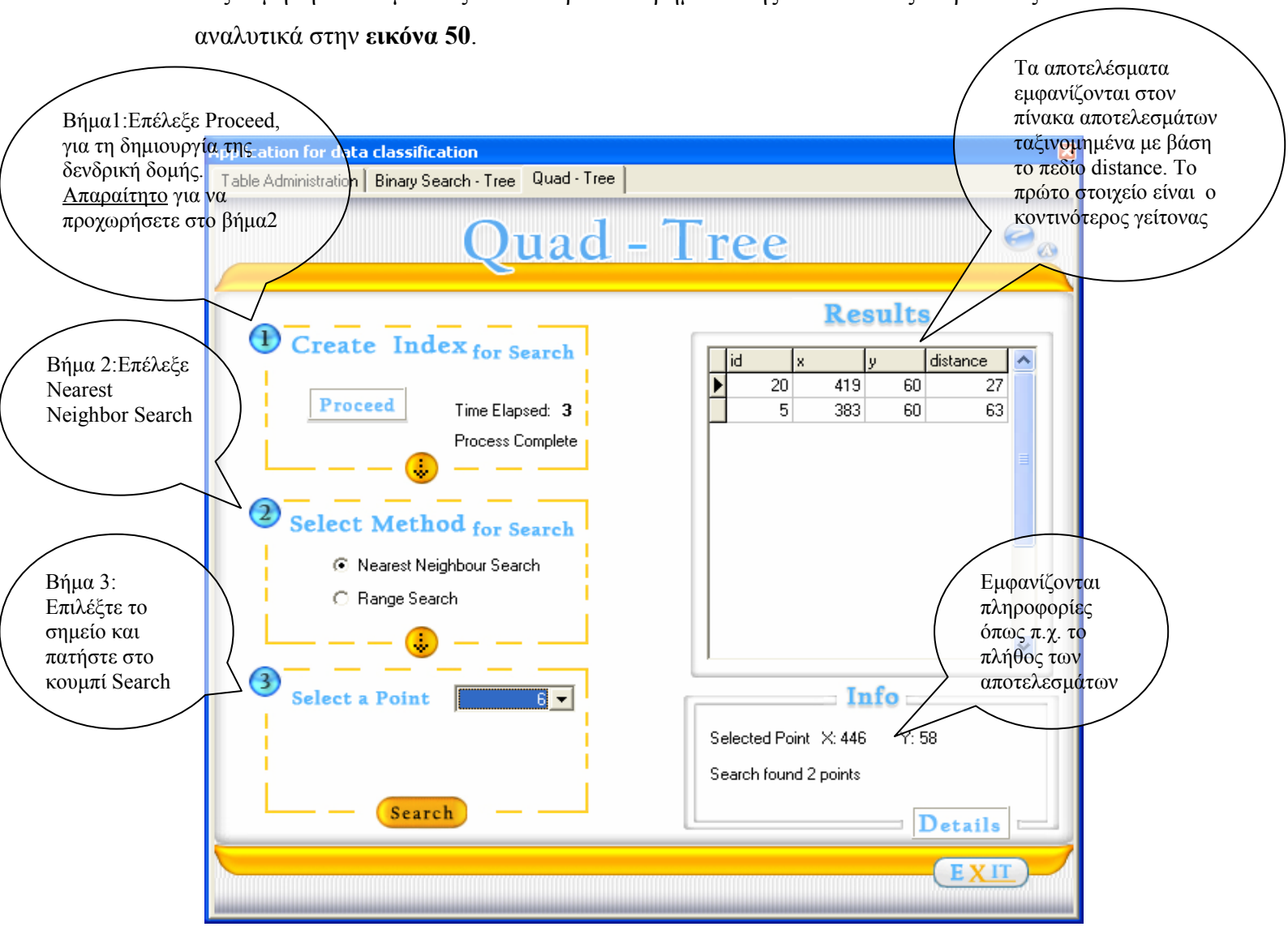
Εικόνα 49: Βήματα για την αναζήτηση ακτίνας στο BS - Tree

5.3 Καρτέλα 3 : Quad - Tree

Η καρτέλα 3 περιέχει όλα τα αντικείμενα που έχουν σχέση με το Quad – Tree. Το πρώτο βήμα (δημιουργία της δενδρικής δομής), είναι κοινό για όλες τις αναζητήσεις γι' αυτό το λόγω και προχωράμε κατευθείαν στην εξήγηση του τρόπου λειτουργίας των αναζητήσεων, επεξηγώντας την διαδικασία βήμα - βήμα.

5.3.1 Αναζήτηση Κοντινότερου γείτονα – Nearest Neighbor Search

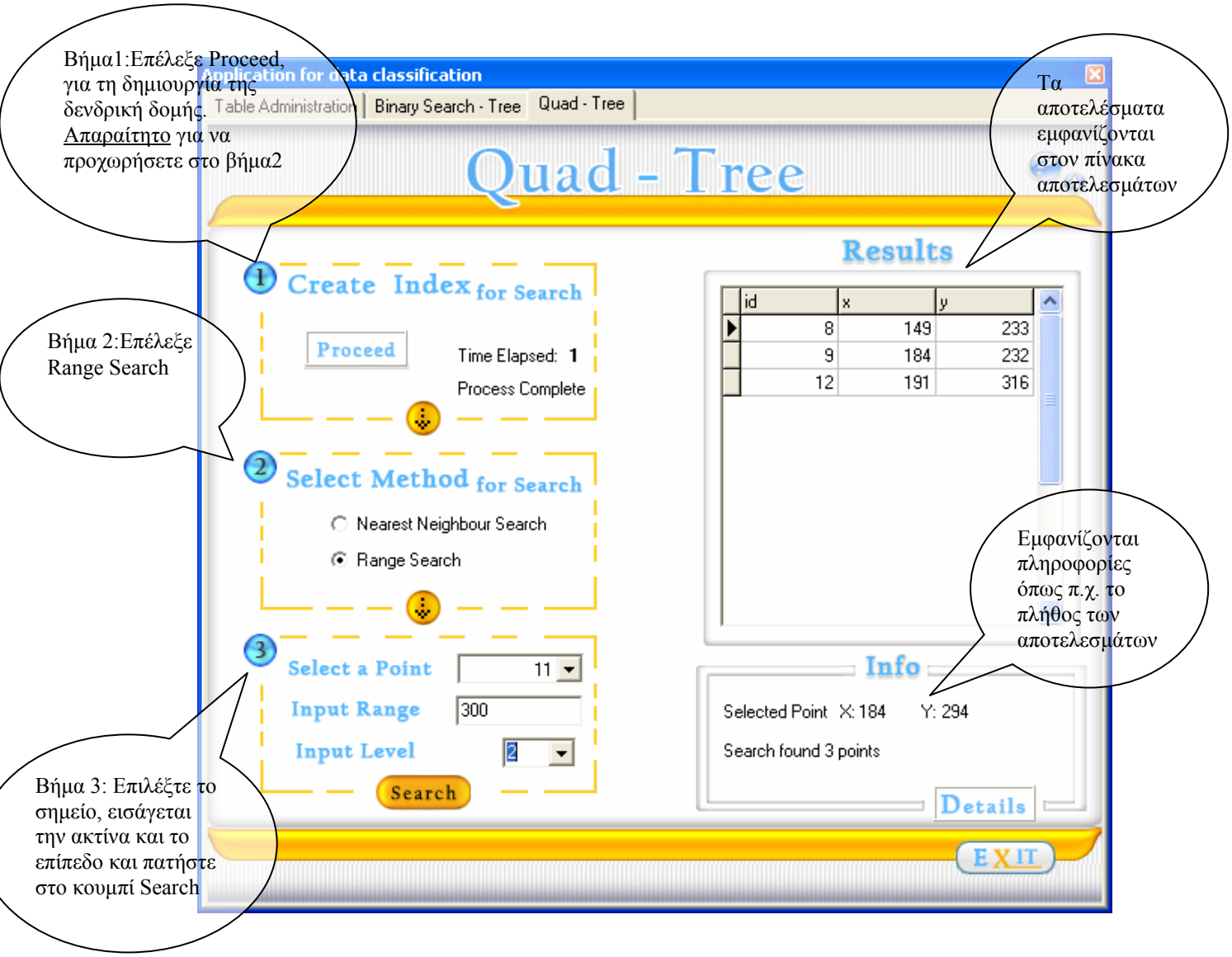
Η αναζήτηση αυτή εμφανίζει τους κοντινότερους γείτονες ενός σημείου με βάση την δενδρική δομή. Στον πίνακα "Results" εμφανίζονται τα αποτελέσματα της αναζήτησης, ταξινομημένα με βάση την απόσταση από το σημείο που επιλέξαμε. Η ταξινόμηση είναι με αύξουσα σειρά. Τα βήματα της διαδικασίας παρουσιάζονται αναλυτικά στην **εικόνα 50**.



Εικόνα 50: Βήματα για την αναζήτηση κοντινότερου γείτονα στο Quad - Tree

5.3.2 Αναζήτηση ακτίνας – Range Search

Η αναζήτηση περιοχής ή ακτίνας εμφανίζει όλα τα σημεία που περιέχονται στην ακτίνα (περιοχή) που εισάγουμε, με βάση την δενδρική δομή (επίπεδα του δένδρου). Τα βήματα της διαδικασίας παρουσιάζονται αναλυτικά στην **εικόνα 51**.



Εικόνα 51: Βήματα για την αναζήτηση ακτίνας στο Quad - Tree

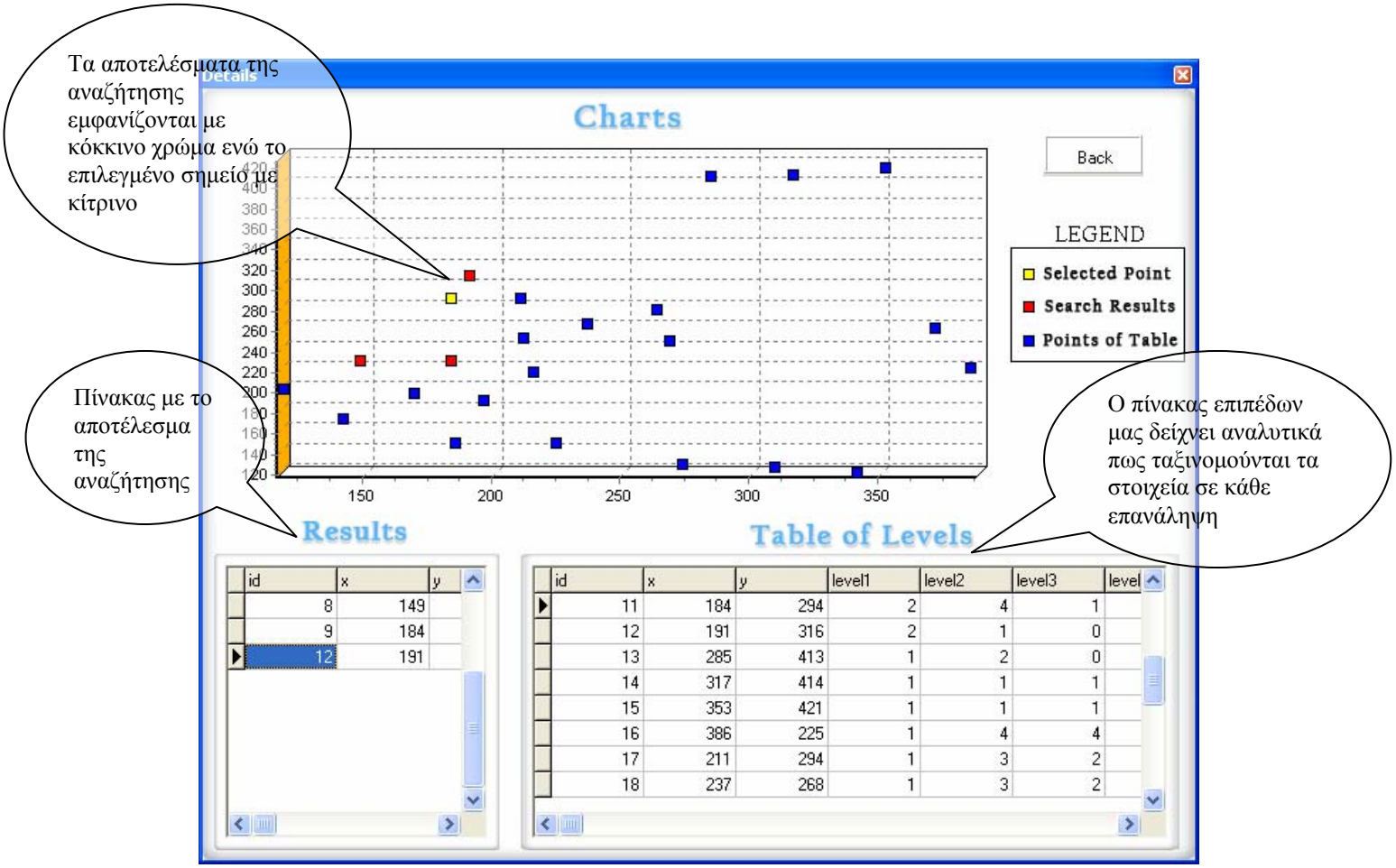
5.4 Η καρτέλα «Λεπτομέρειες - Details»

Πρόκειται για μια πολύ σημαντική καρτέλα, η οποία περιέχει τα διαγράμματα και τους πίνακες επιπέδων και αποτελεσμάτων. Τα διαγράμματα απεικονίζουν τα αποτελέσματα των αναζητήσεων ώστε να γίνουν περισσότερο κατανοητά στον χρήστη. Ο πίνακας επιπέδων αποτελεί ένα πολύ σημαντικό στοιχείο της εφαρμογής καθώς, αν κάποιος διαθέτει την απαραίτητη εμπειρία, μπορεί να επιβεβαιώσει το κατά πόσο η δενδρική δομή δημιουργήθηκε σωστά ή όχι.

Για να εμφανισθεί η καρτέλα αυτή ο χρήστης θα πρέπει να πατήσει στο κουμπί "Details", το οποίο βρίσκεται σε κάθε καρτέλα κάτω από τον πίνακα αποτελεσμάτων (Results), **εικόνα 52**. Το κουμπί αρχικά είναι απενεργοποιημένο και ενεργοποιείται εφόσον ο χρήστης δημιουργήσει την δενδρική δομή και εφαρμόσει σε αυτήν αναζήτηση κοντινότερου γείτονα ή περιοχής.

Εικόνα 52: Το κουμπί Details απενεργοποιημένο και ενεργοποιημένο

Η καρτέλα λεπτομέρειες (Details) φαίνεται στην **εικόνα 53**.



Εικόνα 53: Η καρτέλα λεπτομέρειες

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΛΟΓΟΣ

Στην ενότητα αυτή συνοψίζουμε τα αποτελέσματα της πτυχιακής εργασίας και παραθέτουμε τις μελλοντικές επεκτάσεις που έχουν νόημα και ενδιαφέρον να υλοποιηθούν.

6.1 Σύνοψη και Συμπεράσματα

Από τη μελέτη των αλγόριθμων και την υλοποίηση της εφαρμογής εξάγαμε κάποια συμπεράσματα τα οποία συνοψίζονται στις παρακάτω γραμμές:

Η παραγωγή τυχαίων δεδομένων με συναρτήσεις του Matlab ήταν ταχύτερη σε σχέση με αυτή στην C, εντούτοις η είσοδος των δεδομένων στον πίνακα της βάσης δεδομένων του SQL Server της εφαρμογής, ήταν πολύ αργή.

Το Binary Search – Tree χρησιμοποιείται για δεδομένα μίας διάστασης ενώ το Quad – Tree για δεδομένα δύο διαστάσεων. Το R – Tree μπορεί να χρησιμοποιηθεί για δεδομένα δύο ή και περισσότερων διαστάσεων. Το Quad – Tree μπορεί να επεκταθεί σε περισσότερες από δύο διαστάσεις χωρίς καμία διαφορά στον τρόπο λειτουργίας του αλγόριθμου. Για παράδειγμα, θα μπορούσαμε εύκολα να μετατρέψουμε το Quad – Tree σε Oct – Tree (Quad - Tree τριών διαστάσεων) με την αναπαραγωγή του ίδιου ακριβώς κώδικα που έχουμε για τις προηγούμενες δύο διαστάσεις για ακόμα μία.

Τα δεδομένα εισόδου καθορίζουν σε μεγάλο βαθμό την απόδοση των αλγορίθμων ταξινόμησης. Αυτό σημαίνει ότι ο χρόνος δημιουργίας της δενδρικής δομής εξαρτάται τόσο από το πλήθος των δεδομένων (πυκνότητα) όσο και από την κατανομή τους. Αυτό επιβεβαιώνεται και από τις σχετικές μετρήσεις που αντλήσαμε στον έλεγχο απόδοσης της εφαρμογής. Για παράδειγμα, αν υποθέσουμε ότι τα σημεία στο Quad – Tree έχουν συγκεντρωθεί σε μία περιοχή, ο αλγόριθμος τότε θα πρέπει να δημιουργήσει περισσότερα επίπεδα, με αποτέλεσμα περισσότερο χρόνο για την δημιουργία της δομής.

Το Quad – Tree απαιτεί περισσότερο χρόνο για την δημιουργία της δομής συγκριτικά με το Binary Search – Tree. Θα πρέπει να λάβουμε υπόψη το γεγονός ότι τα δεδομένα που επεξεργάζεται η δομή του Quad – Tree είναι δύο διαστάσεων ενώ του Binary Search – Tree είναι μονοδιάστατα.

Ο χρόνος σε όλους τους τύπους αναζήτησης που εφαρμόσαμε στους αλγόριθμους ταξινόμησης είναι της τάξης των milli second. Μόνο στην περίπτωση της αναζήτησης περιοχής και εφόσον έχουμε επιλέξει το μέγιστο αριθμό επιπέδων, παρατηρείται μια σχετική καθυστέρηση εφόσον έχουμε μεγάλο πλήθος σημείων. Αυτό οφείλεται στο γεγονός ότι η αναζήτηση ελέγχει όλα τα σημεία του πίνακα και ουσιαστικά έχουμε τον εκφυλισμό της αναζήτησης περιοχής σε μια αναζήτηση με βάση την ευκλείδεια απόσταση. Η υψηλή απόδοση των αναζητήσεων είναι και ένα από τα μεγάλα πλεονεκτήματα των δενδρικών δομών.

Συμπεράσματα τρίτων, για τις για τις χωρικές δομές που μελετήσαμε: Η επίδοση του R – Tree σε γενικές γραμμές κρίνεται ανώτερη αυτής του Quad – Tree για την πλατφόρμα της Oracle. Υπάρχουν βέβαια περιπτώσεις κατά τις οποίες το Quad – Tree αποδίδει καλύτερα από το R – Tree. Αυτό οφείλεται στην ύπαρξη πολλών παραμέτρων που καθορίζουν την αποδοτικότητα μιας δομής.

Παρόλο το πλήθος των δομών που έχουν προταθεί, δεν έχει αποδειχτεί, ότι κάποια δομή είναι ανώτερη από τις υπόλοιπες για όλες τις περιπτώσεις.

6.2 Μελλοντικές επεκτάσεις

Από τη μελέτη μας στην πτυχιακή αυτή εργασία προέκυψαν ορισμένα ζητήματα που αποτελούν θέματα μελλοντικής έρευνας:

1. Επέκταση της πτυχιακής κάνοντας χρήση των είδη υλοποιημένων αλγορίθμων σε κάποια εφαρμογή (πολυμέσων, internet, μηχανογράφησης).

2. Χρήση του BS-Tree για την ταξινόμηση αλφαριθμητικών και γενικά δεδομένων εκτός των αριθμητικών ή ταξινόμηση αρχείων στον δίσκο.

3. Υλοποίηση του τρίτου αλγόριθμου που αναλύσαμε στην κεφάλαιο 3 της πτυχιακής, του R-Tree.

4. Περαιτέρω ανάπτυξη και υλοποίηση ορισμένων βασικών παραλλαγών (variations) των αλγορίθμων που παρουσιάσαμε, όπως: Οκταδικό δένδρο κτλ.

5. Με χρήση του πίνακα ταξινόμησης να δημιουργηθεί και καρτέλα που θα παρουσιάζει το δένδρο και πως αυτό έχει σχηματισθεί (οπτικά).

6. Στο Binary Search – Tree, θα ήταν δυνατό με το πάτημα ενός κουμπιού το αρχικό δένδρο να ανασχηματίζεται σε δένδρο AVL, δηλαδή ισορροπημένο.

7. Επέκταση της πτυχιακής κάνοντας χρήση των είδη υλοποιημένων αλγορίθμων με περισσότερες παραμέτρους. Για παράδειγμα στο range query να υπολογίσει όλες τις πόλεις εντός 20 χιλιομέτρων, με πληθυσμό 45000 ατόμων.

ΚΕΦΑΛΑΙΟ 7

ΒΙΒΛΙΟΓΡΑΦΙΑ

Στο κεφάλαιο αυτό υπάρχει συγκεντρωμένη η βιβλιογραφία.. Οι πηγές αυτές ομαδοποιούνται κατά τύπο (έντυπες, ηλεκτρονικές, δημοσιεύσεις) και αποτελούν γενική και προτεινόμενη βιβλιογραφία. Στο ίδιο το κείμενο, όμως, πιθανόν να υπάρχουν παραπομπές και σε άλλες πηγές, μη αναγκαίες συνήθως για τους περισσότερους αναγνώστες.

Ηλεκτρονική

Τα site τα οποία επισκεφθήκαμε ήταν πολλά και το υλικό που συγκεντρώσαμε ογκώδες, παρακάτω παρουσιάζουμε τα site που ξεχωρίσαμε:

Wikipedia WWW Site, The free-content encyclopedia http://en.wikipedia.org

NIST WWW Site, http://www.nist.gov

Oracle WWW Site, http://www.oracle.com, http://otn.oracle.com/products/spatial

R-Tree portal WWW Site, http://www.rtreeeportal.org

Gamedev WWW Site, http://www.gamedev.net

Έντυπη

Τα παρακάτω βιβλία, χρησιμοποιηθήκαν κυρίως για την συγγραφή του δεύτερου κεφαλαίου, ενώ μερικά όπως το "αλγόριθμοι σε C" και "Δομές Δεδομένων Αλγόριθμοι, και εφαρμογές στη C++" βοήθησε στην συγγραφή και του τρίτου.

Δρ. Ι. Κόκκινος, Α. Τσιμπίρης, Βάσεις Δεδομένων ,Τμήμα Πληροφορικής & Επικοινωνιών

Δρ. Α. Τσιμπίρης, Βάσεις Δεδομένων ΙΙ, Τμήμα Πληροφορικής & Επικοινωνιών

C.J. Date, Εισαγωγή στα Συστήματα Βάσεων Δεδομένων, Εκδόσεις Κλειδάριθμος, Έκτη Αμερικάνικη Έκδοση Τόμος Α&Β

Jarrod Hollingworth, Bob Swart, Mark Cashman, Paul Gustavson, *Borland C++ Builder 6 Πλήρες Εγχειρίδιο* ,Εκδόσεις Μ. Γκιούρδας, 1η Αμερικάνικη Έκδοση

Robert Sedgewick, αλγόριθμοι σε C, Εκδόσεις Κλειδάριθμος, 3η Αμερικάνικη Έκδοση

Sartaj Sahni, Δομές Δεδομένων Αλγόριθμοι, και εφαρμογές στη C++, Εκδόσεις ΤΖΙΟΛΑ

Stephen Prata, Η βίβλος της C++ Primer Plus, Εκδόσεις Μ. Γκιούρδας, 3η Αμερικάνικη Έκδοση

Δημοσιεύσεις

Το υλικό του κεφαλαίου 3 : Οι Αλγόριθμοι, προέρχεται κυρίως από τις παρακάτω επιστημονικές αναφορές. Οι αναφορές αυτές είναι ταξινομημένες κατά χρονολογική σειρά.

[Knut73] D. Knuth, *The Art of Computer Programming, vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[FB74] Raphael A. Finkel, Jon Louis Bentley: Quad Trees: A Data Structure for Retrieval on Composite Keys. Acta Inf. 4: 1-9(1974)

[Bent75] J.L. Bentley, “Multidimensional Binary Search Trees Used for Associative Searching”, *Communications of the ACM*, vol. 18, pp. 509-517, 1975.

[Bent75b] J.L. Bentley, and Stanat, D F. "Analysis of range searches in quad trees", *Inf Process Lett* 3, 6 (July 1975}, 170-173

[HS84] Hanan Samet: The Quad tree and Related Hierarchical Data Structures. ACM Comput. Surv. 16(2): 187-260(1984)

[Gutt84] A. Guttman, “R-trees: a dynamic index structure for spatial searching”, *Proceedings of ACM SIGMOD Conference on Management of Data*, June 1984.

[Come79] D. Comer, “The Ubiquitous B-Tree”, *ACM Computing Surveys*, vol. 11(2), pp.121-137, June 1979.

[Sell87] T. Sellis, N. Roussopoulos, C. Faloutsos, "The R+-tree: a dynamic index for multidimensional objects", *Proceedings of the 13th International Conference on Very Large Data Bases* (VLDB), September 1987.

[Beck90] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", *Proceedings of ACM SIGMOD Conference on Management of Data*, May 1990.

[Same90] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[Kame93] I. Kamel, C. Faloutsos, "On Packing R-trees", *Proceedings of the 2nd International Conference on Information and Knowledge Management* (CIKM), November 1993.

[Kame94] I. Kamel, C. Faloutsos, "Hilbert R-tree: An Improved R-tree Using Fractals", *Proceedings of the 20th International Conference on Very Large Data Bases* (VLDB), September 1994.

[Rous95] N. Roussopoulos, S. Kelley, F. Vincent, "Nearest Neighbor Queries", *Proceedings of ACM SIGMOD Conference on Management of Data*, May 1995.

Y. Manolopoulos, A. Nanopoulos, A. Ppapadopoulos, Y Theodoridis,
"R-trees Have Grown Everywhere", ACM Computing Surveys, Vol.V.

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ΣΒΣ	Συστήματα Βάσεων Δεδομένων
ΣΔΒΔ:	Συστήματα Διαχείρισης Βάσεων Δεδομένων
ΧΒΔ:	Χωρικές Βάσεις Δεδομένων
CAD	Computer-Aided Design
GIS:	Geographical Information System
VLSI:	Very Large Scale Integration
OOP	Object – oriented programming
BS - Tree	Binary Search Tree
MBR:	Minimum Bounding Rectangle

Πίνακας Συμβόλων R - Tree

Σύμβολα	**Ορισμοί**
n	Αριθμός διαστάσεων
N	Πλήθος δεδομένων
$p = (p_l, p_u)$	Ορθογώνιο δεδομένων - data - (οριζόμενο από την κάτω-αριστερά και την πάνω-δεξιά γωνία του)
$q = (q_l, q_u)$	Ορθογώνιο ερώτησης - query - (οριζόμενο από την κάτω-αριστερά και την πάνω-δεξιά γωνία του)
M	Μέγιστη χωρητικότητα κόμβου R-tree
m	Ελάχιστη χωρητικότητα κόμβου R-tree
h	Ύψος R-tree
MBR	Εελάχιστο περιβάλλον ορθογώνιο
oid	Ταυτότητα του αντικειμένου

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ