ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ
ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΚΑΤΕΥΘΥΝΣΗ ΔΙΚΤΥΟΚΕΝΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Επεξεργασία Επερωτήσεων με Συνδυασμό Γεωγραφικής Θέσης και Λέξεις-Κλειδιά (Spatio-textual Queries)

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΚΑΡΜΟΥΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων : Χρήστος Δουλκερίδης
Επίκουρος Καθηγητής

Πειραιάς, Μάρτιος 2016

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ
ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΚΑΤΕΥΘΥΝΣΗ ΔΙΚΤΥΟΚΕΝΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Επεξεργασία Επερωτήσεων με Συνδυασμό Γεωγραφικής Θέσης και Λέξεις-Κλειδιά(Spatio-textual Queries)

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΚΑΡΜΟΥΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων : Χρήστος Δουλκερίδης
Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Μαρτίου 2016.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Χρήστος Δουλκερίδης Καθηγητής Παν. Πειραιώς	Μαρία Χαλκίδη Καθηγήτρια Παν. Πειραιώς.	Δημοσθένης Κυριαζής Καθηγητής Παν. Πειραιώς

Πειραιάς, Μάρτιος 2016

(Υπογραφή)

.....

ΣΚΑΡΜΟΥΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Περίληψη

Ο σκοπός της διπλωματικής εργασίας είναι η συγκριτική μελέτη αλγορίθμων εκτέλεσης επερωτημάτων σε πειραματικό ευρετήριο δεδομένων, με συνδυασμό γεωγραφικών και λεκτικών κριτηρίων. Η μελέτη αυτή περιλαμβάνει 4 στάδια. Αρχικά, την ανασκόπηση βιβλιογραφίας για την εκτέλεση ερωτημάτων που συνδυάζουν γεωγραφική θέση και λέξεις κλειδιά, και στην συνέχεια την συλλογή συνόλων δεδομένων για πλήθος διαφορετικών σημείων ενδιαφέροντος. Επιπλέον, υλοποιήθηκαν εφαρμογές εξαγωγής δεδομένων (information extraction) και χρησιμοποιήθηκαν για την συλλογή δεδομένων τουριστικού ενδιαφέροντος στην Ελλάδα, από εξειδικευμένους ιστοτόπους τουριστικών σημείων ενδιαφέροντος και από ιστοτόπους – καταλόγους επιχειρήσεων.

Έπειτα δημιουργήθηκε ευρετήριο IRTree, μια δομή που συνδυάζει το ευρετήριο R-Tree και inverted Index, και του δίνει τη δυνατότητα να αποθηκεύει ταυτόχρονα χωρικά και λεκτικά δεδομένα. Τέλος η εκτέλεση μετρήσεων διαφορετικών αλγορίθμων αναζήτησης πάνω στο υλοποιημένο ευρετήριο. Το πρόβλημα που προσπαθούν να επιλύσουν οι συγκεκριμένοι αλγόριθμοι είναι η εύρεση πλήθους σημείων ενδιαφέροντος με χωρικά και λεκτικά κριτήρια, τα οποία εφαρμόζονται σε γειτονικά σημεία ενδιαφέροντος διαφορετικής κατηγορίας.

Λέξεις Κλειδιά: IRTree, Inverted index, επερωτήματα λεκτικών – χωρικών κριτηρίων

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The purpose of this thesis was to compare two algorithms that can execute queries on a set of data using a combination of spatial and textual criteria. Both of the algorithms use an innovative type of index, named IRTree. This type of index combines the benefits of spatial and textual indexes.

The development of the system included 4 stages. Initially relevant papers were reviewed, that proposed novel methods to index and query data using a combination of geo location and textual relevance. The next step was the collection of data sets for a number of different points of interest. Specifically, a data mining application was developed and used to collect data for points of interest from specialized sites and business directories.

The next phase was the implementation of an index to allow faster execution of the desired query type that is called IR-Tree. The aforementioned index combines two types of indexes – Inverted index for textual relevance criteria and RTree for spatial criteria.

Finally, two algorithms were implemented based on relevant papers and their performance on various input was compared.

Keywords: IR-Tree, top k spatial-textual preference query, index,

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Δουλκερίδη για την βοήθεια του, την επιμονή και υπομονή του, όλο το διάστημα που μεσολάβησε από το πρώτο μάθημα μέχρι το πέρας της παρούσας εργασίας. Ήταν πάντα παρών και χωρίς τη προσφορά του, δεν θα ήταν δυνατή η ολοκλήρωση της.

Επίσης, θα ήθελα να ευχαριστήσω τη γυναίκα μου Σωτηρία, για το κουράγιο και τη δύναμη που μου μετέδωσε καθ' όλη τη διάρκεια φοίτησης στο μεταπτυχιακό.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Κατάλογος εικόνων

1

Εισαγωγή

1.1 Μηχανές αναζήτησης και γεωγραφικά δεδομένα

Η δραματική αύξηση των δεδομένων του Διαδικτύου τα τελευταία χρόνια, με τους ρυθμούς και τα μεγέθη να αλλάζουν συνεχώς, καθιστούν επιτακτική την ανάγκη δημιουργίας αποτελεσματικών ευρετηρίων, ώστε να επιταχυνθούν όλα τα διαφορετικά είδη επερωτημάτων.

Επίσης, η αλλαγή στις συνήθειες των χρηστών, με την αυξανόμενη χρήση κινητών συσκευών, μετέβαλλε και το είδος των δεδομένων για τα οποία οι μηχανές αναζήτησης καλούνται να δημιουργήσουν αποδοτικά ευρετήρια. Συγκεκριμένα, τα δεδομένα παράγονται σε όλο και μεγαλύτερο βαθμό εν κινήσει, και περιέχουν τη θέση του χρήστη. Ταυτόχρονα, ο χρήστης θεωρεί πλέον δεδομένο, πως μια αναζήτηση του στο διαδίκτυο θα περιέχει βέλτιστα αποτελέσματα ως προς την θέση του.

Βέβαια αυτή η αλλαγή, ανοίγει το δρόμο για νέες δυνατότητες. Οι μηχανές αναζήτησης, έχοντας περισσότερες πληροφορίες για την θέση του χρήστη (ή ακόμη και στο ιστορικό των θέσεων και διαδρομών του), και σε συνδυασμό με τις αναζητήσεις του χρήστη, μπορούν να μοντελοποιήσουν τις συνήθειες συνόλου χρηστών όλο και πιο στοχευμένα.

Γίνεται αντιληπτό λοιπόν, πως η βελτιστοποίηση των αλγόριθμων ταξινόμησης, αποθήκευσης και αναζήτησης σε δεδομένα που περιέχουν και γεωγραφική διάσταση, είναι επιτακτική.

Ταυτόχρονα, σε καθημερινή βάση, δημιουργούνται νέα σημεία ενδιαφέροντος σε όλο τον κόσμο. Η πληροφορία που υπάρχει διαθέσιμη για καθένα από αυτά πολλαπλασιάζεται με την χρήση των μέσων κοινωνικής δικτύωσης. Ο χρήστης κατακλύζεται από πληροφορίες και αναζητά νέους τρόπους να τις φιλτράρει με το βέλτιστο τρόπο, ώστε να εξάγει μόνο σχετικά δεδομένα.

Ωστόσο, οι μηχανές αναζήτησης δεν υλοποιούν κάποια λογική που να καλύπτει την παραπάνω ανάγκη. Αυτό που προσφέρεται συνήθως είναι η αναζήτηση μίας κατηγορίας κοντά σε ένα σημείο. Εξειδικευμένες ιστοσελίδες συχνά έχουν πληροφορίες για κοντινά σημεία ενδιαφέροντος. Αλλά αυτή η πληροφορία συνήθως δεν είναι αναζητήσιμη, δεν περιέχει συντεταγμένες και δεν υπάρχει η δυνατότητα ταξινόμησης των αποτελεσμάτων με βάση το περιεχόμενο της αναζήτησης μας. Επίσης είναι περιορισμένη στο είδος του σημείου ενδιαφέροντος που προβάλλεται (πχ για ένα ξενοδοχείο μπορεί να αφορά αποκλειστικά την ύπαρξη ή μη, κοντινών τουριστικών αξιοθέατων).

Στη παρούσα εργασία γίνεται προσπάθεια να καλυφθεί ένας τύπος αναζήτησης που δεν καλύπτεται από τις δημοφιλέστερες μηχανές αναζήτησης. Ο παραπάνω τύπος αναζήτησης προκύπτει από την ανάγκη του χρήστη να εντοπίσει ένα σημείο ενδιαφέροντος με βάση λεκτικά κριτήρια που αφορούν το ίδιο το σημείο, με λεκτικά κριτήρια που αφορούν κοντινά σημεία ενδιαφέροντος. Για παράδειγμα, η αναζήτηση ενός καταλύματος με κάποια στοιχεία (πχ αστέρια, τιμή, παροχές) αποτελεί εύκολη υπόθεση για μια μηχανή αναζήτησης. Αλλά όταν η παραπάνω αναζήτηση προσφέρει πολλά αποτελέσματα, συνήθως ο χρήστης καλείται είτε να προσθέσει κριτήρια που δεν θεωρεί σημαντικά, είτε να επιλέξει τυχαία (ή ψευδό τυχαία καθώς η προτεραιότητα δεν είναι πάντα αντικειμενική αλλά προϊόν υπηρεσίας μηχανής αναζήτησης).

Η αναζήτηση του χρήστη θα μπορούσε να θεωρηθεί ολοκληρωμένη αν δινόταν η δυνατότητα αναζήτησης σημείων ενδιαφέροντος που έχουν σε ακτίνα (που ορίζει ο χρήστης) άλλα σημεία (διαφορετικής κατηγορίας) με χαρακτηριστικά που συμφωνούν με την αναζήτηση του χρήστη.

Κάποια παραδείγματα αναζήτησης είναι η εύρεση:

- των 10 καλύτερων ξενοδοχείων με λεκτικά κριτήρια "κοντά σε παραλία με μπλε σημαία και κοντά σε bar με κοκτειλ".

- των 10 καλύτερων οίκων ευγηρίας με λεκτικά κριτήρια "κοντά σε φαρμακείο και νοσοκομείο με κλινική για νεφροπαθείς".
- των 10 καλύτερων σπιτιών προς ενοικίαση με λεκτικά κριτήρια "κοντά σε πρότυπο σχολείο και πάρκο με γήπεδο".

1.2 Αντικείμενο διπλωματικής

Σκοπός της παρούσας εργασίας είναι η μελέτη αλγορίθμων που έχουν προταθεί για την βελτιστοποίηση των παραπάνω αναζητήσεων. Οι αλγόριθμοι αυτοί έχουν ως στόχο, να υπολογιστούν όσο το δυνατό λιγότεροι συνδυασμοί σημείων ενδιαφέροντος στους οποίους να περιέχονται τα καλύτερα αποτελέσματα και να αντιστοιχηθούν σε μία βαθμολογία ανάλογα με την απόσταση από το Object POI και την συνάφεια με την αναζήτηση του χρήστη. Αυτή η βαθμολογία θα πρέπει να υπολογίζεται δυναμικά, ανάλογα με το βαθμό προτεραιότητας στη λεκτική συνάφεια και την απόσταση μεταξύ των σημείων του συνδυασμού.

Στην υλοποίηση μας απαντάμε στο ερώτημα "εντόπισε τα σημεία ενδιαφέροντος (O) που έχουν σε ακτίνα R, σημεία ενδιαφέροντος F1, F2 που ταιριάζουν στα λεκτικά αναζήτησης Q1,Q2. Επέστρεψε μου τα top k ταξινομημένα με το άθροισμα λεκτικού και χωρικής βαθμολογίας".

Το παραπάνω είδος αναζήτησης παρακινείται από την ανάγκη εύρεσης συνδυασμού σημείων ενδιαφέροντος διαφορετικών κατηγοριών. Συχνά ο χρήστης δεν ενδιαφέρεται για τη θέση που βρίσκεται το σημείο ενδιαφέροντος, αλλά τον απασχολεί να καλύπτονται κριτήρια για γειτονικά σημεία.

Το ευρετήριο που χρησιμοποιήθηκε ονομάζεται IRTree, που στη πραγματικότητα είναι R-Tree με κάθε κόμβο του να περιέχει ανεστραμμένα ευρετήρια (Inverted Indexes).

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής εργασίας συνοψίζεται ως εξής:

1. Μελετήθηκαν αλγόριθμοι δημιουργίας δομών ευρετηρίων και αναζήτησης σε αυτές
2. Υλοποιήθηκαν εφαρμογές εξαγωγής πληροφοριών (scrappers) για πολλαπλές πηγές
3. Υλοποιήθηκε πειραματική δομή ευρετηρίου IR-Tree

4. Υλοποιήθηκαν 2 αλγόριθμοι αναζήτησης σε IR-Tree
5. Πραγματοποιήθηκαν συγκριτικές μετρήσεις μεταξύ των δύο αλγορίθμων

1.3 Δομή του κειμένου

Η δομή της εργασίας συνοψίζεται ως εξής:

- Στο κεφάλαιο 2 παρουσιάζονται εργασίες σχετικές με την θεματική περιοχή που αναλύεται στη συνέχεια
- Στο κεφάλαιο 3 παρουσιάζονται οι δομές ευρετηρίων που χρησιμοποιήθηκαν
- Στο κεφάλαιο 4 αναλύεται το πρόβλημα της αναζήτησης για συνδυασμό σημείων ενδιαφέροντος με χωρικά και λεκτικά κριτήρια
- Στο κεφάλαιο 5 αναλύονται οι αλγόριθμοι αναζήτησης στο IR-Tree
- Στο κεφάλαιο 6 αξιολογούνται και αναλύονται τα αποτελέσματα των συγκριτικών μετρήσεων για τους αλγόριθμους αναζήτησης
- Στο κεφάλαιο 7 περιγράφονται τεχνικές λεπτομέρειες της υλοποίησης

2

Σχετικές εργασίες

Η παρούσα εργασία αναλύει τις θεματικές περιοχές της αναζήτησης με χωρικά και λεκτικά κριτήρια και δημιουργίας αποδοτικών ευρετηρίων. Οι συγκεκριμένες θεματικές περιοχές αποτελούν αντικείμενο συνεχούς έρευνας τα τελευταία χρόνια λόγω της αύξησης του όγκου των δεδομένων και του ποσοστού αυτών που περιέχουν γεωγραφικά στοιχεία.

2.1 Υβριδικά ευρετήρια

Σημαντικό κομμάτι των πολυδιάστατων αναζητήσεων είναι η ύπαρξη αποδοτικών ευρετηρίων, που να μπορούν να στηρίξουν την ευρετηριοποίηση πολλαπλών χαρακτηριστικών. Σε πολλές έρευνες που αναλύουν το πρόβλημα αυτό, παρουσιάζονται συνδυαστικά νέες ή τροποποιημένες δομές ευρετηρίου. Οι πιο συνηθισμένες δομές που προτείνονται είναι οι εξής:

- Δομές που συνδυάζουν το R-Tree (Guttman, 1984) ή το R*-Tree (Beckmann Norbert, 1990) με ανεστραμμένα ευρετήρια Παλαιότερες μελέτες όπως των (Y. Zhou, 2005), χρησιμοποιούσαν τις δύο δομές χαλαρά συνδεδεμένες (ξεχωριστά υλοποιημένες), ενώ νεότερες μελέτες υλοποιούν στενότερη διασύνδεση.
- Δομές που συνδυάζουν ευρετήριο πλέγματος (grid index) με λεκτικό ευρετήριο. Σε αντίθεση με το RTree, τα ευρετήρια πλέγματος χωρίζουν τον χώρο σε ίδιων διαστάσεων παραλληλόγραμμα. Και σε αυτή τη περίπτωση, οι δομές συνδέονται είτε χαλαρά (S. Vaid, 2005) είτε στενά (A. Khodaei, 2010).

2.2 Πολυδιάστατες αναζητήσεις

Η αναζήτηση σε πολλαπλές διαστάσεις αποτελεί σημαντικό αντικείμενο έρευνας τα τελευταία χρόνια. Αρκετές έρευνες έχουν ως στόχο την βελτιστοποίηση ερωτημάτων που αφορούν δύο ή και περισσότερα χαρακτηριστικά σε σύνολα δεδομένων, όπως λεκτικά-χωρικά, χωρικά –χρονικά, κ.α. Οι αλγόριθμοι

αναζήτησης στις δύο διαστάσεις (χωρική-λεκτική) που παρουσιάζονται από σχετικές μελέτες, πραγματοποιούν ως επί το πλείστον τα εξής επερωτήματα:

- Boolean range query – στόχος είναι ο εντοπισμός εγγραφών στο ευρετήριο που περιέχουν όλους του όρους της αναζήτησης και βρίσκονται εντός συγκριμένης ακτίνας από το δοθέν σημείο. Το συγκεκριμένο είδος αναζήτησης μελετάται από τους (S. Vaid, 2005), (Y. Zhou, 2005), (Ramaswamy Hariharan, 2007)
- Boolean k nearest neighbor query – στόχος είναι ο εντοπισμός των k σημείων που πληρούν όλα τα λεκτικά κριτήρια και βρίσκονται πιο κοντά στο δοθέν σημείο. Το συγκεκριμένο είδος αναζήτησης μελετάται από τους (I. D. Felipe, 2008), (Ariel Cary, 2010), (D. Wu, 2012)
- Top k nearest neighbor query – στόχος είναι ο εντοπισμός των k σημείων που βελτιστοποιούν συνάρτηση βαθμολόγησης βασισμένη στην απόσταση και τη λεκτική συνάφεια με το ερώτημα του χρήστη. Το συγκεκριμένο είδος αναζήτησης μελετάται από τους (Gao Cong, 2009), (Ramaswamy Hariharan, 2007), (Zhisheng LI, 2011)

Τέλος, οι μελέτες των (George Tsatsanifos, 2015), (Joao B. Rocha-Junior, 2011), (Joao B. Rocha Junior, 2010) παρουσιάζουν τους αλγόριθμους στους οποίους βασίστηκε η παρούσα εργασία.

3

Δομές ευρετηρίασης

3.1 Εισαγωγή

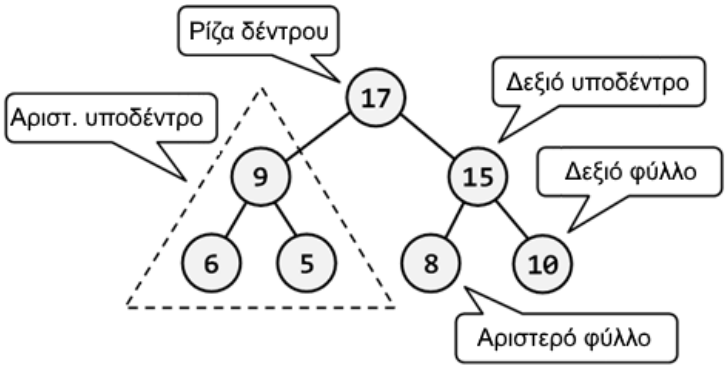
Όλες οι εφαρμογές λογισμικού λειτουργούν με δεδομένα, τα οποία πρέπει να είναι αποθηκευμένα σε κάποιο επίπεδο μνήμης. Υπάρχουν διαφορετικά επίπεδα αποθήκευσης στους υπολογιστές, όπως η κύρια μνήμη (συμπεριλαμβανομένου της κρυφής μνήμης) και ο σκληρός δίσκος. Τα δεδομένα που είναι αποθηκευμένα στο πρώτο επίπεδο μπορούν να προσπελαστούν άμεσα από τον επεξεργαστή του συστήματος. Δεν μπορούμε όμως να διατηρούμε όλα τα δεδομένα στην κύρια μνήμη του συστήματος, είτε εξαιτίας του περιορισμένου αποθηκευτικού χώρου αλλά και σημαντικού κόστους της, είτε γιατί τα δεδομένα δεν μπορούν να διατηρηθούν όταν δεν υπάρχει ρεύμα. Ο σκληρός δίσκος είναι συνήθως φτηνότερος, έχει μεγαλύτερο αποθηκευτικό χώρο και τα αποθηκευμένα δεδομένα μπορούν να διατηρηθούν και με την απουσία ρεύματος. Παρόλα αυτά, το χρονικό κόστος για την προσπέλαση των δεδομένων στο δίσκο είναι δεκαπλάσιο από το αντίστοιχο στην κύρια μνήμη του υπολογιστή. Η οργάνωση των δεδομένων στο δίσκο θα πρέπει να γίνεται με τέτοιο τρόπο ώστε να επιτρέπεται η αποδοτική ανάκτηση τους. Έτσι, η θεμελιώδης ερώτηση που προσπαθεί να απαντήσει η επιστήμη των βάσεων δεδομένων είναι, πως θα γεφυρώσουμε το χάσμα που υπάρχει μεταξύ κύριας μνήμης και σκληρού δίσκου. Μία λύση είναι να χρησιμοποιήσουμε ένα είδος ιεραρχικής δομής δεδομένων, με την οποία θα μπορούμε σε λίγα βήματα να εντοπίσουμε το αντίστοιχο μπλοκ που είναι αποθηκευμένη η ζητούμενη πληροφορία.

Ένα ευρετήριο (index) είναι μια βοηθητική δομή δεδομένων που κάνει πιο αποδοτική την αναζήτηση μιας εγγραφής σε ένα μεγάλο σύνολο εγγραφών, είτε πρόκειται για αντικείμενα σε μια εφαρμογή, αρχεία σε ένα δίσκο ή οποιοδήποτε σύνολο δομών δεδομένων. Το ευρετήριο

καθορίζεται (συνήθως) σε ένα γνώρισμα της εγγραφής. Η δομή του ευρετηρίου καταλαμβάνει μικρότερο χώρο από την ίδια την εγγραφή (οι καταχωρήσεις είναι μικρότερες και λιγότερες). Κάνοντας δυαδική αναζήτηση στο ευρετήριο εντοπίζεται ο δείκτης προς στο μπλοκ όπου αποθηκεύεται η εγγραφή που αναζητάται. Μία από τις πιο ευρέως διαδεδομένες δομές ευρετηρίων είναι οι δεντρικές δομές. Χρησιμοποιούνται σε σχεσιακές βάσεις δεδομένων και λειτουργικά συστήματα.

Ένα δέντρο (tree) σχηματίζεται από κόμβους (nodes), όπου κάθε κόμβος έχει έναν κόμβο-γονέα (parent node) – εκτός από την ρίζα (root) εφόσον είναι ο αρχικός κόμβος του δέντρου – και μηδέν ή περισσότερους κόμβους-παιδιά (child nodes). Ένας κόμβος που δεν έχει παιδιά ονομάζεται κόμβος-φύλλο (leaf ή terminal node), ενώ ένας μη τερματικός κόμβος λέγεται εσωτερικός κόμβος (internal node). Το επίπεδο ενός κόμβου είναι πάντα κατά ένα επίπεδο μεγαλύτερο από το επίπεδο του γονέα του και το επίπεδο της ρίζας είναι πάντα μηδέν. Ένα υποδέντρο (subtree) αποτελείται από έναν κόμβο και όλους τους απογόνους του, δηλαδή από όλους τους κόμβους παιδιά του.

Σε κάθε κόμβο υπάρχουν τόσοι δείκτες όσοι και οι κόμβοι -παιδιά του και μερικές φορές αποθηκεύεται και ένας δείκτης προς το γονέα του. Εκτός από τους δείκτες, ένας κόμβος περιέχει συνήθως και πληροφορίες. Όταν ένα πολύ-επίπεδο ευρετήριο υλοποιείται ως δεντρική δομή, αυτές οι πληροφορίες περιλαμβάνουν τις τιμές του πεδίου ευρετηριοποίησης του αρχείου δεδομένων που χρησιμοποιούνται για να καθοδηγήσουν την αναζήτηση μιας συγκεκριμένης εγγραφής.

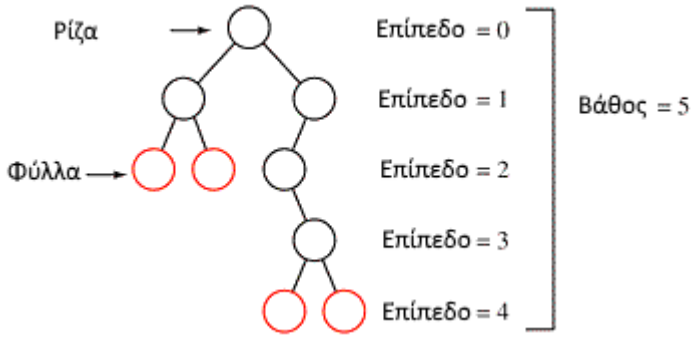


Εικόνα 1- Απλή δεντρική δομή

Ένα δέντρο αναζήτησης μπορεί να χρησιμοποιηθεί ως μηχανισμός αναζήτησης εγγραφών σε ένα αρχείο στο δίσκο. Οι τιμές στο δέντρο θα είναι οι τιμές από ένα πεδίο του αρχείου, που λέγεται πεδίο αναζήτησης και κάθε τιμή στο δέντρο συνδέεται με το αρχείο με ένα δείκτη προς

την εγγραφή που περιέχει αυτή την τιμή στο αρχείο δεδομένων, ή προς το μπλοκ του δίσκου που περιέχει την εγγραφή αυτή. Το δέντρο αποθηκεύεται στο δίσκο τοποθετώντας κάθε κόμβο του σε ένα μπλοκ του δίσκου. Στην εισαγωγή μιας νέας εγγραφής ενημερώνεται το δέντρο με την τιμή της νέας εγγραφής καθώς και με τον δείκτη που δείχνει σε αυτήν.

Ένα δέντρο αναζήτησης όμως δεν παρέχει εξ ορισμού σταθερή απόδοση στην εύρεση των δεικτών που είναι επιθυμητό. Αυτό οφείλεται στο πρόβλημα της εισαγωγής και διαγραφής εγγραφών, καθώς δεν εξασφαλίζεται ότι το δέντρο θα είναι ισοζυγισμένο (balanced), δηλαδή, θα βρίσκονται όλοι οι κόμβοι - φύλλα στο ίδιο επίπεδο. Επίσης, η διαδικασία διαγραφής εγγραφών από το δέντρο, πιθανόν να δημιουργήσει κενούς κόμβους στο δέντρο και ως εκ τούτου, να δημιουργηθεί ένα δέντρο με μεγαλύτερο βάθος από ότι απαιτείται. Ένα ισοζυγισμένο δέντρο εξασφαλίζει ότι δεν θα υπάρχουν κόμβοι σε πολύ υψηλά επίπεδα, οι οποίοι θα απαιτούν πολλές προσπελάσεις μπλοκ κατά την αναζήτηση. Αυτό γίνεται αντιληπτό και από το παρακάτω σχήμα, όταν για τον εντοπισμό ενός συγκεκριμένου φύλλου μπορεί να χρειαστεί να διασχιστούν 2 ή 4 επίπεδα.



Εικόνα 2- Μη ισοζυγισμένο δέντρο

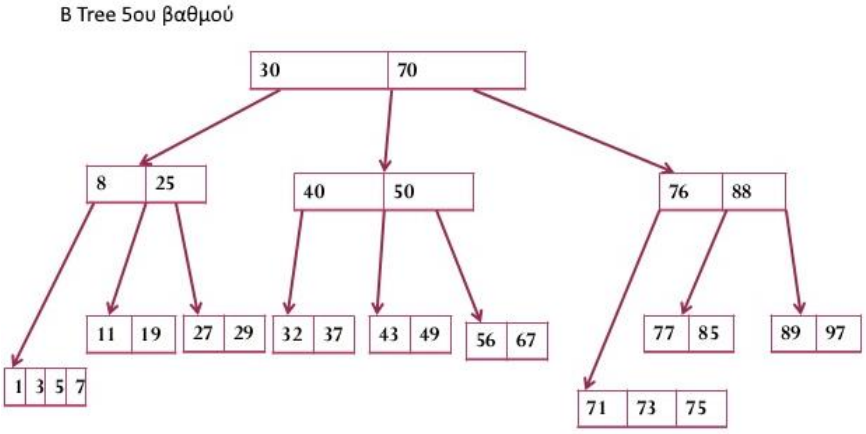
3.2 B-Tree

Τα προβλήματα του μη-ισοζυγισμένου δέντρου, λύνονται με την υλοποίηση κανόνων στην δημιουργία ενός δέντρου, στην προσθήκη και διαγραφή εγγραφών, που προτάθηκαν το 1972 από τους Rudolf Bayer και Edward M. McCreight (R. Bayer, 1972). Η δομή που προτάθηκε ήταν το B-Tree. Ένα B-Tree τάξεως m χρησιμοποιείται ως δομή προσπέλασης σε ένα πεδίο-κλειδί για αναζήτηση εγγραφών, σε ένα σύνολο δεδομένων, παρουσιάζοντας όμως περισσότερους περιορισμούς, που εξασφαλίζουν την ισορροπία του δέντρου. Συγκεκριμένα, οι αλγόριθμοι εισαγωγής και διαγραφής γίνονται περισσότερο πολύπλοκοι, αν και είναι σχετικά απλές διαδικασίες. Μεγαλύτερη σημασία έχουν πολύπλοκες διαδικασίες, όπως η εισαγωγή σε ένα γεμάτο κόμβο ή η διαγραφή από έναν κόμβο, που θα μείνει λιγότερο από το ήμισυ πλήρης.

Οι περιορισμοί για ένα B-Δένδρο είναι οι εξής:

- Κάθε κόμβος περιέχει το πολύ m παιδιά.
- Κάθε μη-τελικός κόμβος, εκτός της ρίζας, περιέχει το πολύ m/2 παιδιά.
- Κάθε μη-τελικός κόμβος με k παιδιά, περιέχει k-1 δείκτες αναζήτησης.
- Η ρίζα περιέχει τουλάχιστον 2 παιδιά εκτός αν είναι ο μόνος κόμβος του δέντρου.
- Όλοι οι κόμβοι – φύλλα είναι στο ίδιο επίπεδο.

Οι παραπάνω κανόνες μπορούν να γίνουν καλύτερα αντιληπτοί, αναλύοντας την εικόνα 5. Πρόκειται για δέντρο 5^{ου} βαθμού, όπου η ρίζα μπορεί να περιέχει από 2 εως 5 παιδιά, οι μη τελικοί κόμβοι από k+1 (3) έως 2k+1 (5) παιδιά ενώ όλα τα φύλλα είναι στο ίδιο επίπεδο.



Εικόνα 3- B-Tree 5ου βαθμού

Στην εισαγωγή εγγραφών στο δέντρο γίνεται εκκίνηση πάντα από τον κόμβο ρίζα (που είναι και κόμβος-φύλλο) στο επίπεδο 0. Όταν γεμίσει ο κόμβος της ρίζας με m-1 τιμές του κλειδιού αναζήτησης, τότε διασπάται σε δύο κόμβους στο επίπεδο 1, όπου εισάγεται η επόμενη τιμή. Η ίδια διαδικασία επαναλαμβάνεται όταν ένας άλλος κόμβος από τη ρίζα είναι γεμάτος και γίνεται μια καταχώρηση σε αυτόν. Τότε, διασπάται και αυτός σε δύο κόμβους στο ίδιο επίπεδο και η μεσαία καταχώρηση μεταφέρεται στο κόμβο γονέα μαζί με δύο δείκτες, που δείχνουν τους κόμβους από τους οποίους προήλθαν. Αν χρειαστεί να διασπαστεί και ο κόμβος-γονέας, τότε η διάσπαση μπορεί να συνεχιστεί μέχρι και την ρίζα, δημιουργώντας ένα νέο επίπεδο κάθε φορά που διασπάται η ρίζα. Κατά τη διαγραφή μιας τιμής, ένας κόμβος μπορεί να μείνει λιγότερο από το ήμισυ γεμάτος. Στην περίπτωση αυτή ενώνεται με τους γειτονικούς κόμβους και αυτή η ένωση μπορεί να συνεχιστεί μέχρι την ρίζα. Επομένως, με την διαγραφή μπορεί να ελαττωθούν τα επίπεδα του δέντρου.

Σύμφωνα με προσομοιώσεις και αναλύσεις που έχουν γίνει, έχει γίνει δεκτό ότι μετά από τυχαίες εισαγωγές και διαγραφές που έχουν γίνει σε ένα δέντρο, οι κόμβοι παραμένουν περίπου κατά 69% γεμάτοι, όταν ο αριθμός των τιμών σταθεροποιηθεί. Με αυτό τον τρόπο, σπάνια θα εμφανίζονται διασπάσεις και ενώσεις κόμβων, ενώ οι εισαγωγές και διαγραφές σε ένα δέντρο γίνονται πολύ πιο αποδοτικές.

3.3 B+Tree

Μία δομή ευρετηρίου που προέκυψε από το B-Tree, είναι το B+Tree (R. Bayer, 1972). Τα B+ Trees αποτελούν μία μορφή ζυγισμένων δέντρων όπου κάθε μονοπάτι από την ρίζα ενός δέντρου έως τα φύλλα του δέντρου έχουν το ίδιο μήκος. Κάθε κόμβος που δεν είναι φύλλο έχει μεταξύ n/2 και n παιδιά, όπου n είναι η τάξη του δέντρου. Τα B+Tree έχουν πολύ καλή απόδοση στην αναζήτηση αλλά δημιουργούν σημαντικό overhead γιατί σπαταλάνε χώρο. Ένας τυπικός κόμβος περιέχει μέχρι το πολύ n-1 κλειδιά και το πολύ n δείκτες(n παιδιά). Τα κλειδιά διατηρούνται στο κόμβο σε σειρά.

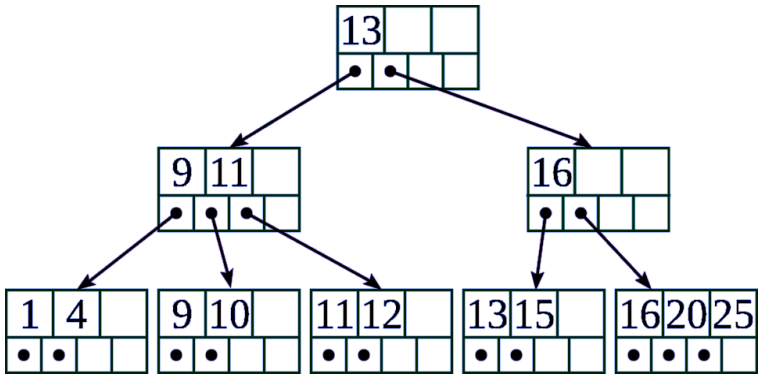
Οι ουσιαστικές διαφορές από το B-Tree είναι οι εξής:

- περιέχει μόνο τους δείκτες στους μη-τελικούς κόμβους
- κάθε κόμβος – φύλλο περιέχει δείκτη προς τον επόμενο κόμβο φύλλο

Οι παραπάνω διαφορές, δίνουν κάποια πλεονεκτήματα στα B+Tree, με αποτέλεσμα να αποτελούν την καταλληλότερη δομή για ευρετήρια σε βάσεις δεδομένων. Συγκεκριμένα:

- Οι μη τελικοί κόμβοι απαιτούν λιγότερο χώρο (καθώς έχουν μόνο δείκτες)
- Είναι δυνατό να διασχιστούν οι κόμβοι - φύλλα σειριακά
- Η διαγραφή από το δέντρο είναι απλούστερη διαδικασία

Στη συνέχεια, η εικόνα 4 παρουσιάζει ένα B+Tree, όπου γίνεται αντιληπτό το κύριο μειονέκτημα της δομής έναντι του B-Tree, η επανάληψη των δεικτών των μη τελικών κόμβων. Αυτό έχει ως αποτέλεσμα να καταλαμβάνει μεγαλύτερο χώρο για την αποθήκευση του.

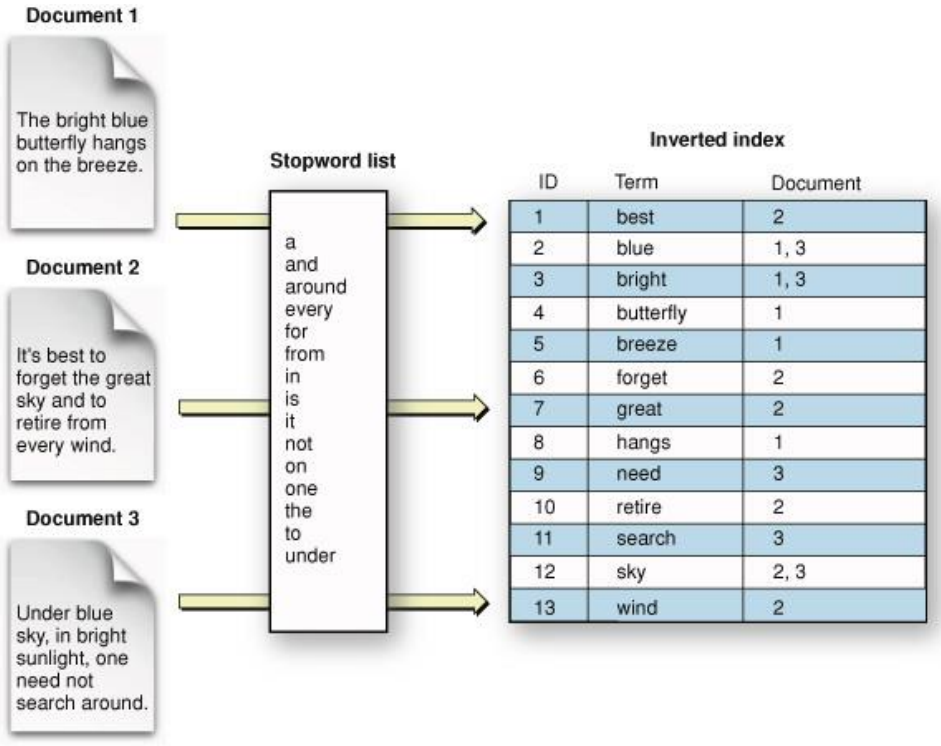


Εικόνα 4- B+Tree

3.4 Ανεστραμμένο ευρετήριο (Inverted Index)

Τα ανεστραμμένα αρχεία είναι ένας μηχανισμός δεικτοδότησης στηριζόμενος σε λέξεις ο οποίος χρησιμοποιείται για αποδοτικότερη αναζήτηση. Τα συστήματα ανάκτησης πληροφορίας χρησιμοποιούν κατά κόρον αυτή τη μέθοδο για την αναζήτηση οποιασδήποτε πληροφορίας. Η αναζήτηση οποιουδήποτε όρου γίνεται πρώτα στους ευρετηριοποιημένους καταλόγους και ύστερα ανακτώνται τα κατάλληλα έγγραφα, αφού υποστούν το ανάλογο φιλτράρισμα. Για την επίτευξη της αναζήτησης με τον τρόπο αυτό, κάθε έγγραφο που εισάγεται στον κατάλογο, ευρετηριοποιείται κατάλληλα και για κάθε έγγραφο που διαγράφεται από τον κατάλογο, ενημερώνονται τα κατάλληλα πεδία στο ευρετήριο, ενώ πάντα υπάρχει και η δυνατότητα δημιουργίας του ανεστραμμένου ευρετηρίου από την αρχή. Το φιλτράρισμα της πληροφορίας κατά τη δημιουργία του ευρετηρίου ακολουθεί μια συγκεκριμένη διαδικασία που περιγράφεται παρακάτω.

Αρχικά, από τα κείμενα τα οποία θα ευρετηριοποιηθούν, φιλτράρεται η πληροφορία εκείνη που είναι ουσιαστική. Εάν για παράδειγμα τα κείμενα είναι λογοτεχνικά θα πρέπει να δοθεί μεγαλύτερη έμφαση στις λέξεις. Εάν πρόκειται για κείμενα θετικών επιστημών θα πρέπει να ληφθούν υπόψιν και οι μαθηματικοί συμβολισμοί. Πάντα θα πρέπει να δίνεται σημασία στο περιεχόμενο, διότι από αυτό προέρχονται και τα περιεχόμενα του ευρετηρίου που δημιουργείται και που ο χρήστης αναζητάει. Το φιλτράρισμα των λέξεων που θα εισαχθούν στο ευρετήριο, γίνεται από τους αντίστοιχους μηχανισμούς οι οποίοι διαχωρίζουν μια ροή γραμμάτων σε λέξεις, φράσεις, συμβολισμούς ή άλλα σημαντικά στοιχεία που ονομάζονται tokens, η αλλιώς διακριτικά. Οι μηχανισμοί που επιτελούν τη συγκεκριμένη διαδικασία ονομάζονται διακριτοποιητές (tokenizers). Οι λέξεις που αγνοούνται καθώς δεν περιέχουν καμία χρήσιμη πληροφορία, αποθηκεύονται σε μία λίστα λέξεων (stopword list) και συνήθως περιέχει άρθρα, συνδέσμους, επιρρήματα και άλλα. Η διαδικασία γίνεται καλύτερα κατανοητή και από την παρακάτω εικόνα.



Εικόνα 5- Ανεστραμμένο ευρετήριο

Από την πληροφορία που συλλέγεται από τους tokenizers, αναγνωρίζονται οι λέξεις εκείνες που ανήκουν σε κάποιες γλωσσικές ενότητες μέσα από άλλους μηχανισμούς, που ονομάζονται γλωσσικές μονάδες. Τα δεδομένα που λαμβάνονται ύστερα από αυτό το στάδιο είναι έτοιμα για ευρετηριοποίηση και αποτελούν το σύνολο των λέξεων που έχουν πραγματική σημασία για το σύσημα. Ο μηχανισμός που επιτελεί τη διαδικασία αυτή ονομάζεται ευρετηριοποιητής. Η γενική μορφή ενός ευρετηρίου προκύπτει από έναν πίνακα, ο οποίος περιλαμβάνει τα κείμενα που έχουμε ευρετηριοποιήσει και τους όρους που έχουμε βρει κατά την ευρετηριοποιήση.

Στα ανεστραμμένα αρχεία η αποθήκευση είναι πιο δυναμική, καθώς δεν αποθηκεύονται μηδενικά κελιά και για το λόγο αυτό χρησιμοποιείται μόνο ένας πίνακας με τους όρους και ένας δείκτης που περιέχει λίστες με έγγραφα που αντιστοιχούν σε αυτόν τον όρο. Μαζί με κάθε έγγραφο που περιέχει τη λέξη αποθηκεύονται και οι αντίστοιχες εμφανίσεις της λέξης στο έγγραφο, ενώ αποθηκεύεται και ένα συνολικό βάρος για κάθε λέξη.

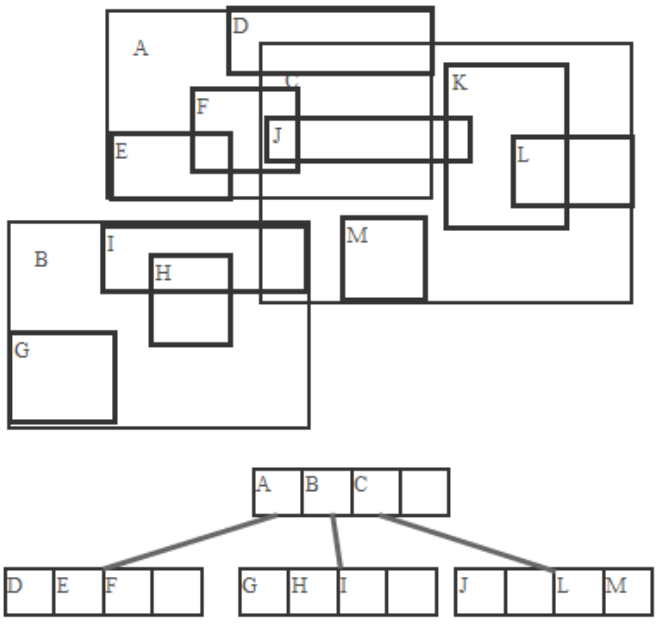
3.5 R-Tree

Το R-tree αποτελεί την επέκταση του B+-tree (Comer, 1979) σε δύο ή υψηλότερες διαστάσεις. Βασικό χαρακτηριστικό της δομής είναι ότι αποθηκεύει πολυδιάστατα ορθογώνια σαν πλήρη αντικείμενα, χωρίς να πραγματοποιεί κάποιου είδους μετασχηματισμό σε σημεία υψηλότερης διάστασης.

Η δομή του R-Tree ακολουθεί συγκεκριμένους κανόνες:

- Η ρίζα έχει τουλάχιστον δύο παιδιά, εκτός εάν είναι φύλλο.
- Κάθε εσωτερικός κόμβος περιλαμβάνει πλήθος παιδιών μεταξύ του m και του M εκτός εάν είναι η ρίζα.
- Κάθε φύλλο περιλαμβάνει πλήθος καταχωρήσεων μεταξύ του m και του M εκτός εάν είναι η ρίζα.
- Όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο (ισοζυγισμένο δένδρο).
- Ένας εσωτερικός κόμβος του δένδρου, περιλαμβάνει ν δείκτες προς τα παιδιά του και το MBR (Minimum Bounding Rectangle) που περικλείει τα παιδιά του.
- Ένα φύλλο του δένδρου περιλαμβάνει δείκτες προς τις εγγραφές του ευρετηρίου και τα πολύγωνα που σχετίζονται με αυτές τις εγγραφές.

Η Εικόνα 6 παρουσιάζει την δομή του ευρετηρίου:

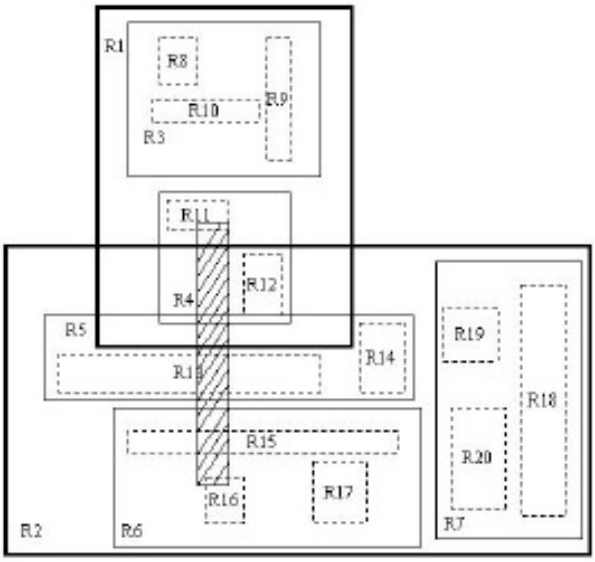


Εικόνα 6- Δομή R-Tree

Το R-tree υποστηρίζει : (α) αναζήτηση δεδομένων, (β) εισαγωγή δεδομένων, (γ) διαγραφή δεδομένων και (δ) ενημέρωση δεδομένων.

3.5.1 Αναζήτηση Δεδομένων

Η εκτέλεση της αναζήτησης σε R-Tree πραγματοποιείται είτε με σημείο είτε με ορθογώνιο συντεταγμένων (MBR) (Εικόνα 7). Ο αλγόριθμος αναζήτησης διασχίζει το δένδρο ξεκινώντας από τη ρίζα, με ένα τρόπο παρόμοιο με την αναζήτηση στο B-tree (Comer, 1979). Δηλαδή συγκρίνει το MBR της ρίζας με το MBR του ερωτήματος μας και διασχίζει το δέντρο για όσα MBR-παιδιά υπάρχει επικάλυψη. Ωστόσο, υπάρχει μία ουσιαστική διαφορά, ότι τα υπό-δένδρα που πρέπει να εξεταστούν κάθε φορά μπορεί να είναι περισσότερα του ενός και αιτία γι' αυτό αποτελεί το γεγονός ότι το R-tree επιτρέπει την επικάλυψη μεταξύ των MBRs στους εσωτερικούς κόμβους του δένδρο.



Εικόνα 7- Αναζήτηση στο R-Tree

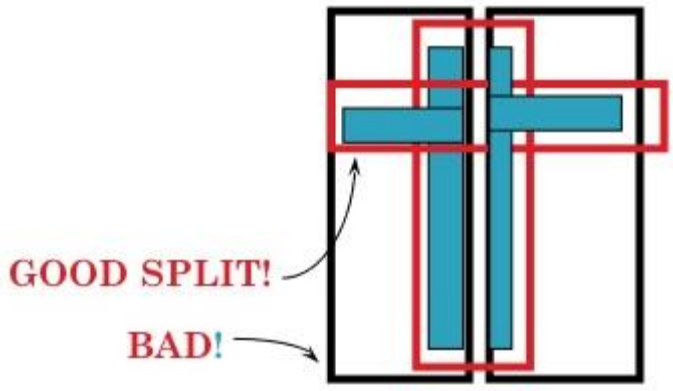
3.5.2 Εισαγωγή Εγγραφών

Για να προστεθεί νέα εγγραφή στο δέντρο, διασχίζεται το δέντρο από τη ρίζα προς τα φύλλα ώστε να επιλεγεί ο κατάλληλος κόμβος. Οι κόμβοι με πλήθος παιδιών άνω του ορίου, διαιρούνται και οι διασπάσεις μεταδίδονται προς τα ανώτερα επίπεδα του δένδρου.

Αρχικά, καλείται η ChooseLeaf που επιστρέφει το πρώτο ορθογώνιο το οποίο περιλαμβάνει εξ' ολοκλήρου το νέο ορθογώνιο. Ο αλγόριθμος συνεχίζει τη διάσχιση του δέντρου, εντοπίζοντας τους κόμβους που αλληλεπικαλύπτονται με το νέο ορθογώνιο. Αν βρεθούν 2 τέτοια ορθογώνια στο ίδιο επίπεδο του δέντρου, επιλέγεται αυτό που θα προκαλέσει την μικρότερη αύξηση στο ορθογώνιο που θα εισαχθεί. Αν επιλεγεί κάποιος κόμβος που έχει ήδη μέγιστο πλήθος κόμβων, καλείται η διαδικασία διάσπασης κόμβων, ώστε να δημιουργηθεί ένας νέος κόμβος. Τέλος καλείται η AdjustTree, προκειμένου να προσαρμοστούν τα περιθώρια των κόμβων και να διαδοθούν οι αλλαγές προς τα πάνω.

3.5.3 Διάσπαση Κόμβων

Κατά την εισαγωγή μιας νέας εγγραφής σε ένα πλήρη κόμβο με M εγγραφές, είναι απαραίτητη η δημιουργία ενός δεύτερου κόμβου και η διαίρεση των M+1 εγγραφών μεταξύ δύο. Ο διαμοιρασμός πρέπει να γίνει έτσι ώστε να ελαχιστοποιείται η πιθανότητα εξέτασης και των δύο νέων κόμβων, σε μια μεταγενέστερη αναζήτηση. Δεδομένου ότι η απόφαση για το αν θα εξεταστεί ένας κόμβος εξαρτάται από την επικάλυψη του MBR με την περιοχή αναζήτησης, η συνολική επιφάνεια των MBRs των δύο νέων κόμβων θα πρέπει να ελαχιστοποιηθεί. Η Εικόνα 8παρουσιάζει δύο διαφορετικές επιλογές για τη διάσπαση κόμβων. Ο διαχωρισμός που σχεδιάζεται με μπλε χρωματισμό αποτελεί την καλύτερη περίπτωση, μιας και η επιφάνεια των δύο MBRs είναι η μικρότερη δυνατή.



Εικόνα 8- Διάσπαση κόμβων

Ο (Guttman, 1984) πρότεινε στην εργασία του τρεις μεθόδους διάσπασης κόμβων, την εξαντλητική (Exhaustive), την τετραγωνική (Quadratic) και τη γραμμική (Linear). Όλες έχουν σχεδιαστεί ώστε να ελαχιστοποιούν την επιφάνεια που καλύπτεται από τα δύο ορθογώνια που προκύπτουν μετά τη διάσπαση. Η πρώτη μέθοδος εντοπίζει την επιφάνεια με το συνολικό ελάχιστο. Ωστόσο, παρουσιάζει εκθετική πολυπλοκότητα ως προς το πλήθος των εγγραφών M σε κάθε κόμβο ή φύλλο του δένδρου κι έτσι ο χρόνος εκτέλεσης και η χρήση της CPU την καθιστά αναποτελεσματική. Οι δύο επόμενες μέθοδοι, προσπαθούν να προσεγγίσουν τη βέλτιστη λύση. Πιο συγκεκριμένα, η τετραγωνική μέθοδος παρουσιάζει τετραγωνική πολυπλοκότητα σε σχέση με την τιμή του M, οδηγώντας σε αυξημένο χρόνο εκτέλεσης, εντούτοις επιτυγχάνει μια καλή προσέγγιση. Η γραμμική μέθοδος είναι η ταχύτερη, αφού η πολυπλοκότητα σε σχέση με την τιμή του M είναι γραμμική, αλλά οδηγείται σε χειρότερη προσέγγιση. Συμπερασματικά, η τετραγωνική μέθοδος διάσπασης είναι καταλληλότερη από τις τρεις, αφού συμβιβάζει την ταχύτητα με την αποδοτικότητα κατά την αναζήτηση, όπως περιγράφεται κατωτέρω. Η μέθοδος για τη δημιουργία των ομάδων περιλαμβάνει δύο στάδια.

Το πρώτο εκτελείται μια φορά και κάνει την επιλογή των εγγραφών που θα αποτελέσουν τα πρώτα στοιχεία των δύο ομάδων. Το κριτήριο που χρησιμοποιείται για τον εντοπισμό των καταλληλότερων εγγραφών, είναι η απόσταση μεταξύ των ορθογωνίων τους. Συγκεκριμένα, αναζητείται το ζεύγος εγγραφών με ορθογώνια που απέχουν όσο το δυνατόν περισσότερο και συνεπώς θα σπαταλούσαν τον μεγαλύτερο χώρο αν τοποθετούνταν στην ίδια ομάδα. Η σπατάλη μετριέται από τον "νεκρό χώρο", δηλαδη από τη διαφορά της επιφάνειας του ορθογωνίου που περικλείει τα ορθογώνια των δύο εγγραφών, από τις επιφάνειές των ορθογωνίων καθεμιάς από τις εγγραφές. Η έννοια του νεκρού χώρου γίνεται καλύτερα αντιληπτή από την Εικόνα 9.

Εικόνα 9- Νεκρός χώρος ανάμεσα σε 2 MBR

Για κάθε ζεύγος των M+1 εγγραφών, υπολογίζεται η τιμή του "νεκρού χώρου" που προκύπτει από την ομαδοποίησή τους και τελικά επιλέγεται το ζεύγος με τη μεγαλύτερη τιμή από αυτές. Οι εγγραφές του τελευταίου θα διαχωριστούν και θα αποτελέσουν τα πρώτα στοιχεία των ομάδων και άρα των δύο νέων κόμβων. Με αυτό τον τρόπο, ελαχιστοποιείται ο "νεκρός χώρος" και συνακόλουθα η πιθανότητα να ακολουθήσουμε περισσότερα του από ένα μονοπάτια προς τα φύλλα, κατά τη διάρκεια μιας αναζήτησης.

Στο δεύτερο στάδιο, εκτελείται μια επαναληπτική διαδικασία για την αντιστοίχιση των υπολοίπων M-1 εγγραφών στις δύο ομάδες. Για κάθε εγγραφή υπολογίζεται η αύξηση που απαιτείται στην επιφάνεια του MBR κάθε ομάδας, προκειμένου να συμπεριλάβει το MBR της εγγραφής. Μετά την ολοκλήρωση των υπολογισμών για όλες τις εγγραφές, επιλέγεται εκείνη που εμφανίζει τη μεγαλύτερη διαφορά μεταξύ των δύο ομάδων και αντιστοιχίζεται στην ομάδα με τη μικρότερη αύξηση. Στη συνέχεια, η διαδικασία εκτελείται επαναληπτικά για το νέο υπόλοιπο των εγγραφών.

3.5.4 Διαγραφή εγγραφών

Η περίπτωση της διαγραφής είναι και αυτή παρόμοια με τον τρόπο διαγραφής στο B-tree (Comer, 1979). Αρχικά καλείται η FindLeaf ώστε να βρεθεί το φύλλο που περιλαμβάνει την εγγραφή. Μετά τη διαγραφή καλείται η CondenseTree ώστε να προσαρμοστούν τα MBRs σε καθένα από τα επίπεδα του δένδρου. Αν μετά την διαγραφή εγγραφής, υπάρχει κόμβος με λιγότερα παιδιά από την ελάχιστη τιμή, διαγράφεται.

Από τη διαδικασία που περιγράφηκε παραπάνω, φαίνεται ότι ο τρόπος διαχείρισης των κόμβων με πλήθος εγγραφών μικρότερο από το ελάχιστο επιτρεπτό, διαφέρει από την περίπτωση του B-tree (Comer, 1979), όπου δύο ή περισσότεροι γειτονικοί κόμβοι με τέτοια χαρακτηριστικά απλώς συγχωνεύονται. Αντιθέτως, στην περίπτωση του R-tree, οι κόμβοι αυτοί διαγράφονται και οι εγγραφές τους επανεισάγονται στη δομή. Τα πλεονεκτήματα της μεθόδου είναι τα ακόλουθα:

- επιτυγχάνει το ίδιο αποτέλεσμα με τη συγχώνευση, αλλά είναι ευκολότερη στην υλοποίησή της αφού χρησιμοποιεί την ήδη έτοιμη μέθοδο εισαγωγής Insert.
- λόγω του γεγονότος ότι ο αριθμός των προσπελάσεων των σελίδων στη μνήμη αποτελεί ένα κρίσιμο μέγεθος της αποδοτικότητας, η τελευταία είναι συγκρίσιμη μεταξύ των δύο

μεθόδων, μιας και οι σελίδες που θα χρησιμοποιηθούν για την επανεισαγωγή θα βρίσκονται ήδη στη μνήμη από τη διαδικασία αναζήτησης (στις περισσότερες των περιπτώσεων).

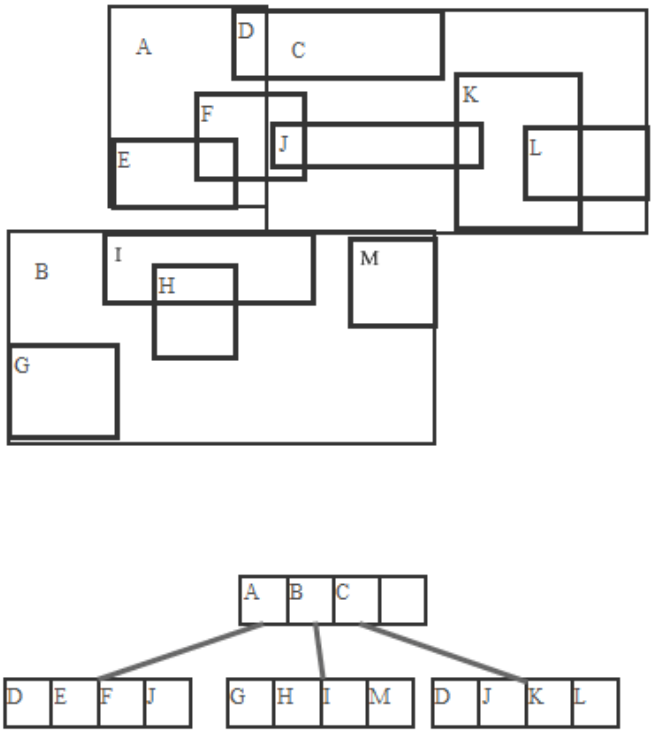
- βελτιώνει σταδιακά τη χωρική δομή του δένδρου και αποτρέπει την επιδείνωσή της, που μπορεί να προκύψει εάν κάθε εγγραφή τοποθετείται συνεχώς κάτω από τον ίδιο κόμβο-γονέα.

3.5.5 Ενημέρωση Δεδομένων

Στην περίπτωση που μια εγγραφή στο δέντρο μεταβληθεί έτσι ώστε να τροποποιηθεί το MBR της, πρέπει να διαγραφεί από τη δομή, να ενημερωθεί με τα νέα δεδομένα και κατόπιν να επανεισαχθεί, αναζητώντας τη νέα κατάλληλη θέση στο δένδρο.

3.6 R-Tree*

Το R*-tree παρουσιάστηκε για πρώτη φορά από τον (Beckmann Norbert, 1990) και η υλοποίησή του βασίστηκε στη θεωρία που είχε αναπτυχθεί για το R-tree, από τον (Guttman, 1984). Οι συγγραφείς μελέτησαν τον τρόπο κατασκευής, τις ιδιότητες και τις λειτουργίες της δομής, στοχεύοντας στη βελτιστοποίηση κάποιων διαδικασιών και στο σχεδιασμό μιας νέας δομής, η οποία θα κάνει αποτελεσματικότερα την ευρετηριοποίηση και την αναζήτηση χωρικών δεδομένων. Η Εικόνα 10 παρουσιάζει τη δομή του R+Tree και μπορούμε να συγκρίνουμε με την Εικόνα 6, τον διαφορετικό τρόπο με τον οποίο δημιουργείται το δέντρο σε σχέση με το R-Tree.



Εικόνα 10- Δομή R+Tree

Το R-tree (Guttman, 1984) υποστηρίζει με αποδοτικό τρόπο, τη δυναμική κατασκευή MBRs από υποσύνολα πλήθους μεταξύ m και M για ένα τυχαίο σύνολο ορθογωνίων, έτσι ώστε να εκτελούνται τυχαίες διαδικασίες ανάκτησης με ορθογώνια ερωτήματος τυχαίου μεγέθους. Η λειτουργία της δομής στηρίζεται στην ευρηστική βελτιστοποίηση της περιοχής του MBR σε κάθε εσωτερικό κόμβο του δένδρου. Αυτό όμως δεν έχει πάντα ως αποτέλεσμα τη βέλτιστη δομή. Υπάρχουν παράμετροι που δύναται να βελτιστοποιηθούν, όπως:

- Το περιθώριο των MBRs
- Η αλληλοεπικάλυψη των MBRs
- Η διαδικασία αποθήκευσης

Οι παραπάνω παράμετροι, όντως είναι σημεία τα οποία χρίζουν βελτιστοποίησης και αφορούν την απόδοση της διαδικασίας ανάκτησης δεδομένων. Όμως αλληλοεπιδρούν μεταξύ τους με έναν πολύπλοκο τρόπο, με αποτέλεσμα η βελτιστοποίηση της μίας να επηρεάζει τις υπόλοιπες. Επιπρόσθετα, αφού τα MBRs των εγγραφών μπορούν να έχουν πολύ διαφορετικά μεγέθη και σχήματα και τα MBRs των εσωτερικών κόμβων της δομής μεγεθύνονται και σμικρύνονται δυναμικά, η επιτυχία μεθόδων που βελτιστοποιούν μία παράμετρο είναι πολύ αβέβαιη.

Εκτελώντας πολυάριθμα και ποικίλα πειράματα, ο (Beckmann Norbert, 1990) σχεδίασε το R*-tree που ενσωματώνει έναν συνδυασμό βελτιστοποίησης της περιοχής, του περιθωρίου και τις επικάλυψης κάθε MBR στους κόμβους και τα φύλλα του δένδρου. Μετά από μια εξαντλητική σύγκριση επιδόσεων, απέδειξε ότι το R*-tree υπερτερεί σε σχέση με τις υπάρχουσες παραλλαγές του R-tree (Linear R-tree, Quadratic R-tree, Greene's R-tree). Η υπεροχή αυτή ισχύει για διαφόρους τύπους ερωτημάτων και πράξεων. Επιπρόσθετα των συνηθισμένων point query, rectangle intersection και rectangle enclosure query, οι συγγραφείς ανέλυσαν τη λειτουργία της νέας δομής στο spatial join ορθογωνίων ή πολυδιάστατων σημείων που αποτελεί την πλέον σημαντική πράξη σε συστήματα Γεωγραφικών και Περιβαλλοντικών Βάσεων Δεδομένων.

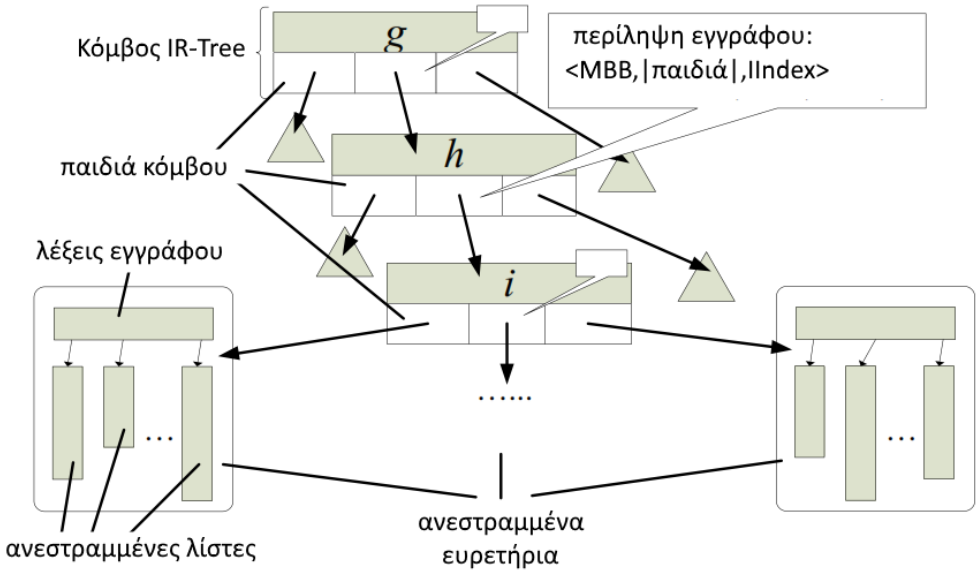
Συμπερασματικά, από πρακτικής άποψης το R*-tree (Beckmann Norbert, 1990) αποτελεί μια ελκυστική μεθοδολογία, αφού υποστηρίζει με αποδοτικό τρόπο σημειακά και χωρικά δεδομένα ταυτόχρονα και η υλοποίησή του κοστίζει ελαφρώς περισσότερο σε σχέση με το R-tree (Guttman, 1984) και τις υπόλοιπες παραλλαγές του (Greene, 1990).

3.7 IR-Tree

Για να υποστηριχθεί η αποτελεσματική αναζήτηση εγγράφων με συνδυασμό λέξεων-κλειδιά και των γεωγραφικών δεδομένων τους, προτάθηκε το IR-Tree (Gao Cong, 2009). Το IR-Tree αποτελεί τη σύνθεση διαφορετικών δομών δεδομένων. Βάση του αποτελεί το R*Tree που στη συνέχεια κάθε κόμβος του συνδέεται με ένα ανεστραμμένο ευρετήριο (inverted index) για όλα τα παιδιά του κόμβου. Αυτό δίνει τη δυνατότητα να πραγματοποιηθεί σε κάθε κόμβο ταυτόχρονα λεκτική αναζήτηση και χωρική σύγκριση του ερωτήματος του χρήστη.

Τα βήματα που ακολουθούνται για την δημιουργία ενός IR-Tree, περιγράφονται συνοπτικά με τα παρακάτω βήματα και την εικόνα 8:

1. Δημιουργείται ένα B-Tree ευρετήριο με κλειδί το μοναδικό αναγνωριστικό της εγγραφής για κάθε feature σύνολο και για το object σύνολο.
2. Δημιουργείται ένα R*Tree για κάθε feature σύνολο και για το object σύνολο.
3. Ενημερώνεται κάθε κόμβος του R*Tree με ένα inverted index για όλα τις εγγραφές που περιέχονται στους κόμβους παιδιά του.



Εικόνα 11- Δομή IR-Tree

4

Ορισμός του προβλήματος

Το πρόβλημα που καλείται να αναλύσει η παρούσα εργασία, αφορά εξελιγμένους τρόπους αναζήτησης σημείων ενδιαφέροντος με χρήση λεκτικών και χωρικών κριτηρίων.

Όπως αναλύθηκε και στο κεφάλαιο 2, το πρόβλημα της αναζήτησης top k σημείων ενδιαφέροντος έχει μελετηθεί εκτενώς, σε περιπτώσεις που η θέση γύρω από την οποία καλούμαστε να αναζητήσουμε είναι στατική.

Η εργασία προσπαθεί να αναλύσει την περίπτωση που δεν έχουμε μια συγκεκριμένη στατική θέση, αλλά θέλουμε να εντοπίσουμε σημεία ενδιαφέροντος ταξινομημένα με ένα συνδυασμό λεκτικών και χωρικών κριτηρίων εφαρμοσμένων πάνω σε γειτονικά σημεία ενδιαφέροντος διαφορετικής κατηγορίας. Ένα είδος ερωτημάτων που ανήκει στην ίδια θεματική ενότητα είναι η εύρεση βέλτιστης τοποθεσίας (optimal location queries)

4.1 Εύρεση top k σημείων ενδιαφέροντος

Δεδομένου ενός συνόλου σημείων ενδιαφέροντος **O** (εφεξής data objects) και ενός πλήθους **c** συνόλων σημείων ενδιαφέροντος $\mathbf{F_i}$ σε **c** διαφορετικές κατηγορίες (εφεξής feature objects), αναλύουμε το πρόβλημα εύρεσης των **k** καλύτερων data objects που έχουν σε ακτίνα **r**, συνδυασμό **c** feature objects, που έχουν λεκτική συνάφεια με το επερώτημα του χρήστη. Η ταξινόμηση των data objects βασίζεται σε σκορ που υπολογίζεται από ένα συνδυασμό του βαθμού συνάφειας και της απόστασης των feature objects.

Ορισμός 1. *Δεδομένου ενός data object* ***p****, ακτίνας* ***r*** *και ένα σύνολο από λέξεις κλειδιά* ***W****, ορίζεται* ***s(t,p)*** *ως το σκορ ενός feature object* ***t (feature score****), που βρίσκεται εντός ακτίνας R από το p και ισούται με* $\boldsymbol{s(t,p) = \alpha \cdot dist(p,t) + (1-\alpha) \cdot \frac{1}{sim(t,W)}}$*, όπου* $\lambda \in [0,1]$ *,* ***sim()*** *είναι συνάρτηση λεκτικής ομοιότητας,* ***dist()*** *είναι συνάρτηση υπολογισμού απόστασης, και το* $\boldsymbol{\alpha}$ *είναι παράμετρος εξομάλυνσης ανάμεσα στο λεκτικό και το χωρικό σκορ.*

Ως συνάρτηση λεκτικής ομοιότητας ανάμεσα στις λέξεις κλειδιά του επερωτήματος του χρήστη **W** και **t.W** των λέξεων κλειδιά του feature object **t**, χρησιμοποιείται η μέθοδος Jaccard και ισούται με : $\boldsymbol{sim(t,W) = \frac{|t.W \cap W|}{|t.W \cup W|} = \frac{\text{πλήθος κοινών λέξεων κλειδιά σε ερώτημα και έγγραφο}}{\text{πλήθος ένωσης λέξεων κλειδιά σε ερώτημα και έγγραφο}}}$

Ορισμός 2. *Το σκορ προτίμησης (****preference score****)* $\tau_i(p)$ *του data object p για το σύνολο feature object της κατηγορίας* $i \in [1, c]$ *ορίζεται ως :*

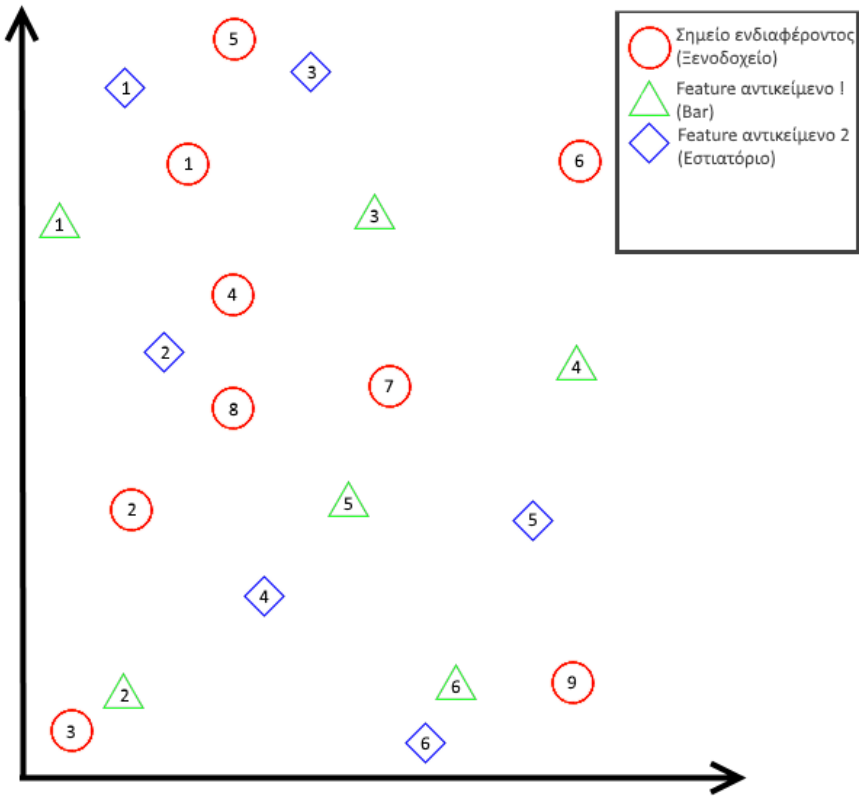
$$\tau_i(p) = min\{\, s(t,p) \mid t \in F_i : dist(p,t) \leq r \text{ and } sim(t, W_i) > 0\}$$

Ορισμός 3. *Το συνολικό σκορ προτίμησης (****overall preference score****)* $\tau(p)$ *του data object ορίζεται ως:* $\tau(p) = min\{\, s(t,p) \mid t \in F_i : dist(p,t) \leq r \text{ and } sim(t, W_i) > 0\}$

Από τους παραπάνω ορισμούς, προκύπτει ο ορισμός του προβλήματος που αναλύει η παρούσα εργασία:

Δεδομένου του ερωτήματος Q, που ορίζεται από ένα ακέραιο k, ακτίνα r και c πλήθος συνόλων λέξεων κλειδιά $\boldsymbol{W_i}$ ***, να βρεθούν k data objects*** $\boldsymbol{p \in O}$ ***με το ελάχιστο δυνατό σκορ.***

Για την καλύτερη κατανόηση του προβλήματος παρατίθεται η Εικόνα 12. Δεδομένων τριών συνόλων σημείων ενδιαφέροντος, όπου O είναι σύνολο ξενοδοχειακών επιχειρήσεων, F1 σύνολο επιχειρήσεων διασκέδασης και το F2 σύνολο επιχειρήσεων εστίασης, το ζητούμενο του προβλήματος είναι η εύρεση των k ξενοδοχείων που ελαχιστοποιούν τη συνάρτηση προτίμησης, που παρουσιάστηκε στον ορισμό 2. Τα σημεία που ελαχιστοποιούν αυτή τη συνάρτηση είναι αυτά που έχουν συνδυαστικά σημεία ενδιαφέροντος από τα σύνολα F1, F2 με μεγάλη λεκτική συνάφεια με τις λέξεις κλειδιά του ερωτήματος και μικρή απόσταση από κάποιο σημείο του O.



Εικόνα 12

F1 ID	Λέξεις κλειδιά F1
1	club, pop, RnB
2	πιάνο, jazz, κελάρι κρασιών
3	καφέ, espresso, συνδρομητικά κανάλια
4	μπύρες, ποικιλίες φαγητού, λουκάνικα
5	club, disco, 80s μουσική
6	ροκ bar, metal, alternative μουσική

F2 ID	Λέξεις κλειδιά F2
1	sushi, κινέζικη κουζίνα, ασιατική
2	σουβλάκια, ψησταριά, ψητοπωλείο
3	πίτσα, μακαρονάδες, ιταλική κουζίνα
4	fast food, burger, σάντουιτς, hot dog
5	τυρόπιτες, προϊόντα σφολιάτας
6	ψητοπωλείο, φαλάφελ

Η παραπάνω εικόνα δείχνει τις θέσεις των σημείων ενδιαφέροντος και των τριών συνόλων και ο πίνακας δείχνει τις λέξεις κλειδιά που περιγράφουν το κάθε σημείο ενδιαφέροντος των συνόλων F1, F2. Έστω ότι η αναζήτηση του χρήστη είναι «***βρες το 1 καλύτερο ξενοδοχείο που είναι κοντά σε club με 80s μουσική και σε ψητοπωλείο με σουβλάκια***». Τα σημεία που καλύπτουν τα λεκτικά κριτήρια είναι τα (1,5) και (2,6) από το F1 και το F2 αντίστοιχα. Όμως τα 5 και το 2 έχουν μεγαλύτερο σκορ από τα 5 και 6, καθώς περιέχουν μόνο 1 από τις 2 λέξεις της αναζήτησης. Η λύση λοιπόν που αναμένουμε να μας προτείνει ένας αλγόριθμος, είναι το σημείο 8 του συνόλου O.

5

Αλγόριθμοι αναζήτησης

Η παρουσίαση ενός αποδοτικού ευρετηρίου αποτελεί το πρώτο βήμα της μελέτης που πραγματεύεται η παρούσα εργασία. Το δεύτερο βήμα είναι η συγκριτική μελέτη .

5.1 Spatio-Textual Preference Search (STPS).

Ο παραπάνω αλγόριθμος υλοποιήθηκε στις γενικές γραμμές του (George Tsatsanifos, 2015). Επικεντρώνεται στην εύρεση έγκυρων συνδυασμών feature object. Διασχίζει λοιπόν πρώτα τα δέντρα των feature και προσθέτει κόμβους του δέντρου ή feature object που πληρούν λεκτικά και χωρικά κριτήρια σε μια ουρά προτεραιότητας. Από τη στιγμή που για κάθε κόμβο υπάρχει η δυνατότητα να υπολογίσουμε την λεκτική ομοιότητα με την αναζήτηση του χρήστη, η ουρά ταξινομείται με φθίνουσα σειρά των τιμών jaccard των κόμβων και των feature object. Έτσι σε κάθε επανάληψη, ο αλγόριθμος παίρνει από την ουρά προτεραιότητας τον συνδυασμό αντικειμένων (είτε είναι κόμβοι είτε αντικείμενα) με το καλύτερο σκορ, όσον αφορά τα λεκτικά κριτήρια. Για αυτό τον συνδυασμό, υπολογίζει τα καλύτερα data object και έπειτα συνεχίζει στην επόμενη επανάληψη του αλγορίθμου όπου πάλι προσθέτει αντικείμενα ή κόμβους στην ουρά και αφαιρεί τον επόμενο καλύτερο συνδυασμό.

Η παραπάνω λογική περιγράφεται καλύτερα με τον παρακάτω ψευδοκώδικα:

Input : Query Q
Output: Result set R sorted based on $\tau(p)$
while *(*$|R| \leq k$*)* **do**
  $C =$ `nextCombinations(`Q`);`
  $R = R \cup$ `getDataObjects(`C`);`
end

Algorithm 1: STPS Algorithm

Εικόνα 13- Αλγόριθμος STPS

Ο αλγόριθμος εντοπίζει συνδυασμούς feature objects που πληρούν τα κριτήρια και προσθέτει στο σύνολο των αποτελεσμάτων όλα τα data object που βρίσκονται εντός της ακτίνας που έχει δοθεί.

Input : Query Q
$heap_i$: heap maintaining entries of F_i
$heap$: heap maintaining valid combinations of feature objects
D_i: set of feature objects of F_i
Output: C valid combinations with highest score
while *(*$\exists i$: not $heap_i$.isEmpty()*)* **do**
  $i \leftarrow$ `nextFeatureSet();`
  $e_i \leftarrow heap_i.pop()$;
  while *(*not e_i is a data object*)* **do**
    for *childEntry in* $e_i.childNodes$ **do**
      $heap_i$.push(childEntry);
    end
    $e_i \leftarrow heap_i$.pop();
  end
  $D_i = D_i \cup e_i$;
  $heap$.push(`validCombinations(`$D_1, ..., e_i, ...D_c$`)`);
  $min_i = s(e_i)$;
  $\tau = max_{i \leq j \leq c}(max_1 + ... + min_j + ... + max_c)$;
  $C \leftarrow heap.top()$;
  if $score(C) \geq \tau$ **then**
    $heap$.pop();
    return C;
  end
end

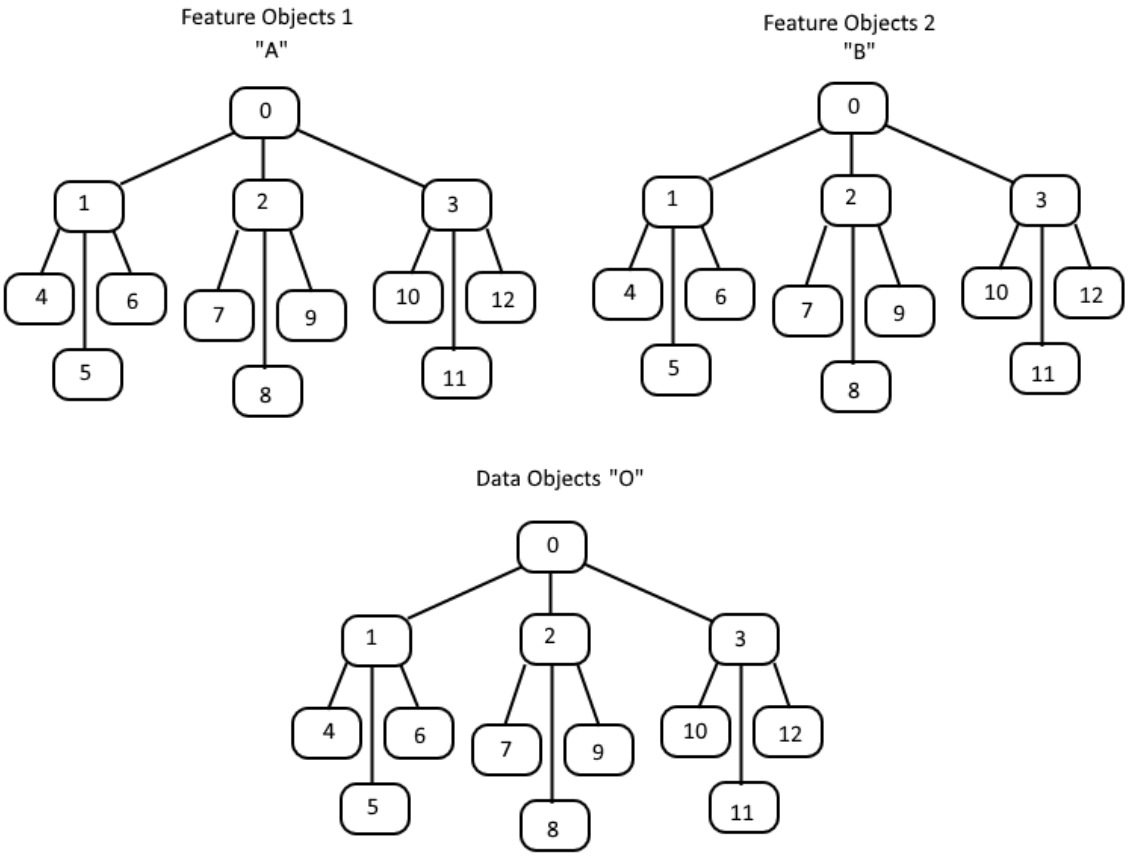
Algorithm 2: Valid Combinations

Εικόνα 14- STPS, συνάρτηση valid combinations

Ο εντοπισμός ενός έγκυρου συνδυασμού περιγράφεται στον παραπάνω ψευδοκώδικα. Υποθέτει πως υπάρχουν c στοίβες που περιέχουν τα feature object των c κατηγοριών. Από κάθε στοίβα εξάγει το πρώτο αντικείμενο. Αν το αντικείμενο είναι κόμβος, βάζει όλα τα παιδιά του κόμβου στη στοίβα και συνεχίζει μέχρι η στοίβα να εξάγει feature object και όχι κόμβο.

Όταν εξαχθεί feature object, αν καλύπτει τα λεκτικά κριτήρια, προστίθεται σε λίστα Di . Το νέο feature object που μπήκε στο Di, ελέγχεται με όλα τα υπόλοιπα feature object που είναι στο Dj όπου j ≠ i. Όλοι οι έγκυροι συνδυασμοί που θα προκύψουν, εισάγονται στην ουρά προτεραιότητας (heap), όπου διατηρούνται ταξινομημένα με το άθροισμα των λεκτικών τους βαθμολογιών. Αν η ουρά προτεραιότητας, έχει συνδυασμό με συνολικό score καλύτερο του ορίου, επιστρέφεται ώστε να εντοπιστούν τα data objects.

Για να γίνει καλύτερα κατανοητός ο αλγόριθμος, παρατίθεται η εικόνα xxx.



Αρχικά εισάγεται στη **heap** ο συνδυασμός (A0,B0). Έπειτα θα εισάγει τους κόμβους A1, A2, A3 στο **heap$_1$** και θα εξάγει το πρώτο αντικείμενο (έστω το A1) με κριτήριο την λεκτική συνάφεια (jaccard). Έτσι θα συνεχίσει μέχρι το **heap$_1$** εξάγει σημείο ενδιαφέροντος και όχι κόμβο. Το σημείο αυτό είναι βέλτιστο όσον αφορά το λεκτικό σκορ και εισάγεται στη λίστα **D$_1$**. Ομοίως θα εισάγει τους κόμβους B1, B2, B3 στο **heap$_2$** και θα εξάγει το αντικείμενο με το μεγαλύτερο score jaccard (έστω το B1), και θα συνεχίσει να εισάγει τα παιδιά κόμβων μέχρι η στοίβα να εξάγει αντικείμενο-σημείο ενδιαφέροντος. Το αντικείμενο θα προστεθεί στη λίστα **D$_2$.**

Έπειτα κάθε νέο αντικείμενο της λίστας $\mathbf{D_1}$ συνδυάζεται με όλα τα αντικείμενα του $\mathbf{D_2}$ και κάθε νέο αντικείμενο της λίστας $\mathbf{D_2}$. Όσοι συνδυασμοί feature object είναι εντός ακτίνας 2 * R, θεωρούνται έγκυροι συνδυασμοί και εισάγονται στην ουρά προτεραιότητας (**heap**).

Για το πρώτο αντικείμενο της στοίβας, εντοπίζεται από το δέντρο των Data object, το ξενοδοχείο που έχει το μικρότερο σκορ προτίμησης. Αν έχει συμπληρωθεί το πλήθος των αποτελεσμάτων που έχουν ζητηθεί, ο αλγόριθμος τερματίζει.

5.2 Three Range Query Score (TRQS)

Ο αλγόριθμος βασίζεται στην υλοποίηση του [xxx] και βασίζεται σε μεγάλο βαθμό στην τεχνική του κλαδέματος (pruning) μιας δεντρικής δομής. Ο αλγόριθμος διατηρεί c πλήθος από λίστες που περιέχουν κόμβους που δεν χρειάζεται να ελεγχθούν, καθώς δεν πληρούν είτε τα λεκτικά είτε τα χωρικά κριτήρια της αναζήτησης του χρήστη. Επίσης διατηρεί μία ουρά προτεραιότητας που ταξινομείται με το συνολικό σκορ για κάθε συνδυασμό κόμβων-αντικειμένων.

Αρχικά διασχίζει κατά μήκος (Breadth first) όλα τα δέντρα στο ίδιο επίπεδο. Για κάθε συνδυασμό κόμβων υπολογίζει αν πρέπει να μπει στην ουρά προτεραιότητας ή να προστεθεί κάποιος από τους κόμβους σε pruning list. Αφού επεξεργαστεί όλους τους κόμβους του επιπέδου 1, συνεχίζει στην επόμενη επανάληψη, για όσο η ουρά προτεραιότητας περιέχει αντικείμενα. Από αυτό το σημείο και μετά, η ουρά προτεραιότητας αναλαμβάνει να καθοδηγήσει με ποιο τρόπο θα γίνει διάσχιση του δέντρου καθώς επιστρέφει πάντα τον συνδυασμό κόμβων ή αντικειμένων με το καλύτερο σκορ.

Η παραπάνω λογική περιγράφεται καλύτερα με τον παρακάτω ψευδοκώδικα:

Input : Query Q
$pruningList_i$: heap maintaining pruning entries of F_i
$heap$: priority queue maintaining valid combinations of data object and feature object
Output: Result set R sorted based on $\tau(p)$

```
e = IRNode(root_O, root_1, root_2);
heap.add(e);
while (!heap_i.isEmpty()) do
    e = heap.pop();
    if (e.obj == dataObj AND e.feature1 == dataObj AND e.feature2 ==
     dataObj) then
        R = R ∪ e ;
    end
    if (e.obj == leaf AND e.feature1 == leaf AND e.feature2 == leaf) then
        rangeQuery(e) ;
    else
        if (e.obj != leaf AND e.feat1 == leaf AND e.feat2 == leaf) then
            for (child i in e.obj) do
                if (isValidCombination (i)) then
                    heap.add(IRNode(i, e.feat1, e.feat2)) ;
                end
            end
        end
        if (e.obj != leaf AND e.feat1 != leaf AND e.feat2 == leaf) then
            for (child i in e.obj) do
                for (child j in e.feat1) do
                    if (isValidCombination (i)) then
                        heap.add(IRNode(i, j, e.feat2)) ;
                    else
                        pruningList_i.add(j) ;
                    end
                end
            end
        end
        ...
    end
end
```

Algorithm 1: Three Range Query Score

Εικόνα 15- Αλγόριθμος ThreeRangeQueryScore

Προστίθεται στην λίστα προτεραιότητας ένα αντικείμενο (IRNode) που περιέχει τις ρίζες των τριών δέντρων. Η κλάση IRNode απεικονίζει ένα συνδυασμό Data Node ή Data Object και c Feature Nodes ή Objects. Το inverted index σε κάθε κόμβο επιτρέπει την σύγκριση των κόμβων με αντικείμενα. Έπειτα για όσο η λίστα περιέχει αντικείμενα IRNode, εξάγουμε το πρώτο αντικείμενο της λίστας. Αν το IRNode περιέχει μόνο φύλλα εκτελείται η rangeQuery μέθοδος για τα 3 φύλλα. Αν όμως περιέχει έστω ένα μη τελικό κόμβο, εξάγονται από το κάθε μη τελικό

κόμβο τα παιδιά του και ελέγχονται όλοι οι συνδυασμοί . Αν κάποιος από τους συνδυασμούς πληροί τα κριτήρια του ερωτήματος, εισάγεται στην ουρά αλλιώς οι κόμβοι που δεν πληρούν τα κριτήρια, εισάγονται στις λίστες κουρέματος (pruning lists) και αγνοούνται σε κάθε μελλοντική επεξεργασία. Για λόγους συντομίας ο ψευδοκώδικας που παρατέθηκε δεν περιλαμβάνει όλους τους δυνατούς συνδυασμούς ελέγχων στα αντικείμενα του IRNode, καθώς γίνεται αντιληπτό πως αναλύονται όσοι μη τελικοί κόμβοι υπάρχουν στο IRNode.

Η μέθοδος rangeQuery λειτουργεί με τον ίδιο τρόπο, δηλαδή ελέγχει όλους τους δυνατούς συνδυασμούς των αντικειμένων των τριών φύλλων που πήρε σαν είσοδο. Κάθε συνδυασμός είτε είναι έγκυρο αποτέλεσμα και επιστρέφεται, είτε εισάγεται στη pruning λίστα.

6

Αξιολόγηση

6.1 Παράμετροι αξιολόγησης

Οι παράμετροι αξιολόγησης που επιλέχθηκαν είναι:

- IO – πλήθος των Input/Output ενεργειών που εκτελέστηκαν στα ευρετήρια
- Time – ο χρόνος ολοκλήρωσης της εκτέλεσης του αλγορίθμου.

Βασικός στόχος ενός αλγόριθμου αναζήτησης είναι να καταλήξει στα ίδια αποτελέσματα με όσο το δυνατό λιγότερη χρήση του ευρετηρίου και στο μικρότερο δυνατό χρόνο. Αν και οι δύο τιμές σε γενικές γραμμές έχουν μία αναλογική σχέση, καθώς η αύξηση των IO σημαίνει αύξηση και χρόνου εκτέλεσης, η ανάλυση και των δύο τιμών μπορεί να οδηγήσει σε χρήσιμα συμπεράσματα για τα πλεονεκτήματα της μίας μεθόδου υπέρ της άλλης.

6.2 Σύνολο δεδομένων

Για τις ανάγκες της εργασίας συγκεντρώθηκαν δεδομένα από πολλαπλές πηγές για τρεις κατηγορίες σημείων ενδιαφέροντος. Οι κατηγορίες που επιλέχθηκαν ήταν: α) καταλύματα, β) διασκέδαση και γ) εστίαση.

Για την συλλογή των δεδομένων υλοποιήθηκαν scrappers (εφαρμογές στοχευμένης εξαγωγής πληροφοριών από ιστοσελίδες), που παραμετροποιήθηκαν για την εξαγωγή δεδομένων από τις πηγές που επιλέχθηκαν. Αρχικά συλλέχθηκαν δεδομένα από 2 μεγάλα site - καταλόγους επιχειρήσεων για όλες τις επαγγελματικές κατηγορίες που περιείχαν.

Στο πρώτο site, συλλέχθηκαν δεδομένα από το html για κάθε επιχείρηση (meta και σχετικά html elements).

Στη περίπτωση του δεύτερου υλοποιήθηκε εφαρμογή που εκτελούσε ερωτήματα αναζήτησης στα web services που εξυπηρετούν την mobile εφαρμογή του site. Για τον εντοπισμό όλων των δυνατών κλήσεων που πραγματοποιεί η παραπάνω mobile εφαρμογή, χρησιμοποιήθηκε proxy debugger εφαρμογή (fiddler). Η εφαρμογή κατέγραφε τις κλήσεις που περνούσαν από τη κινητή συσκευή προς το τοπικό δίκτυο. Έπειτα μοντελοποιήθηκαν οι κλήσεις και οι απαντήσεις σε κλάσεις και υλοποιήθηκε rest client που εκτελούσε περίπου 500 ταυτόχρονες κλήσεις - περίπου 3Μ κλήσεις συνολικά.

Επιπλέον υλοποιήθηκε scrapper και για 3 μεγάλα εξιδεικευμένα site για κάθε μία από τις 3 επαγγελματικές κατηγορίες που στοχεύσαμε:

Για την εστίαση από το ask4food.gr, για τη διασκέδαση από το athinorama.gr και για τα καταλύματα από το booking.com.

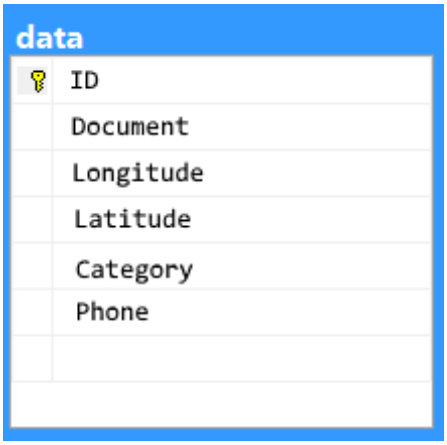
Η εξαγωγή δεδομένων από τα site πραγματοποιήθηκε με τα εξής βήματα:

- Εντοπίστηκαν οι σελίδες αποτελεσμάτων και ο τρόπος που γίνεται πλοήγηση ανάμεσα σε αυτές
- Για μεμονωμένη σελίδα σημείου ενδιαφέροντος εντοπίστηκαν τα html element που περιείχαν πληροφορία που μας ενδιέφερε (είτε ορατή στο χρήστη είτε όχι)
- Καταγράφηκε ο τρόπος κλήσης που υποστηρίζει το site και τυχόν πληροφορίες (header) που είναι απαραίτητες για την ορθή απάντηση
- Υλοποιήθηκε εφαρμογή που καλούσε σειριακά τις σελίδες αποτελεσμάτων και αποθήκευε τις διευθύνσεις (url) των μεμονωμένων σημείων
- Καλούσε τις σελίδες των σημείων και μέσω css query έκανε εξαγωγή των δεδομένων
- Εισήγαγε σε βάση δεδομένων

Με αρχική βάση τα περιεχόμενα του πρώτου site, εμπλουτίστηκαν τα δεδομένα του από τα υπόλοιπα (με κλειδί για την ένωση το τηλέφωνο της επιχείρησης).

Η παραπάνω διαδικασία πραγματοποιήθηκε ώστε να αυξήσουμε το κείμενο που περιγράφει κάθε σημείο ενδιαφέροντος και κατ' επέκταση και την αποτελεσματικότητα των εκάστοτε αναζητήσεων.

Όλα τα δεδομένα αποθηκευτήκαν σε πέντε πίνακες με την ίδια δομή που φαίνεται από το παρακάτω διάγραμμα.



Εικόνα 16- Σχήμα πίνακα ΒΔ γαι τα εξορυγμένα δεδομένα

Από την συγχώνευση και τον καθαρισμό των δεδομένων που συλλέξαμε προέκυψαν συνολικά περίπου 80 χιλιάδες σημεία ενδιαφέροντος σε πανελλαδικό επίπεδο:

- 25941 σημεία ενδιαφέροντος διαμονής
- 25612 σημεία ενδιαφέροντος διασκέδασης
- 30970 σημεία ενδιαφέροντος εστίασης

6.3 Σύστημα αξιολόγησης

Αφού δημιουργήθηκαν τα ευρετήρια για τις 3 κατηγορίες σημείων ενδιαφέροντος, επιλέχθηκε ένα σύνολο λέξεων κλειδιά

6.4 Οργάνωση πειραμάτων

Για την εκτέλεση των πειραμάτων, επιλέχθηκαν 4 μεταβλητές που αφορούν τις δυναμικές τιμές που μπορεί να επιλέξει ο χρήστης στην εκτέλεση του ερωτήματος του:

1) **Top k** – το πλήθος των καλύτερων αποτελεσμάτων που θέλουμε να μας επιστρέψει η εκτέλεση του αλγορίθμου
2) **Ακτίνα R** – η μέγιστη απόσταση που θέλουμε να απέχουν τα feature POIs από το POI

3) **Alpha** – παράμετρος εξομάλυνσης ανάμεσα στη λεκτική συνάφεια του ερωτήματος με το POI και της απόστασης.
4) **Πλήθος λέξεων κλειδιά** – το άθροισμα των λέξεων κλειδιά που περιέχει το ερώτημα του χρήστη.

Ο παρακάτω πίνακας περιέχει τις τιμές των μεταβλητών για τις οποίες πραγματοποιήθηκαν μετρήσεις. Με έντονη γραμματοσειρά, έχουν επισημανθεί οι προεπιλεγμένες τιμές των παραμέτρων για τις μετρήσεις.

Παράμετρος	**Τιμές μετρήσεων**	**Εύρος δυνατών τιμών**
Ακτίνα (σε μέτρα)	50, 100, **150**, 200, 250	[0 – ∞)
Top k	5,10,**15**,20,25	[1 – ∞)
Alpha (smoothing parameter)	0.1, 0.3, **0.5**, 0.7, 0.9	[0.0 – 1.0]
Πλήθος λέξεων-κλειδιά αναζήτησης	1, **3**, 5	[0 – ∞)

Οι μετρήσεις πραγματοποιήθηκαν πολλαπλές φορές ανά συνδυασμό παραμέτρων και υπολογίστηκε ο μέσος όρος των αποτελεσμάτων.

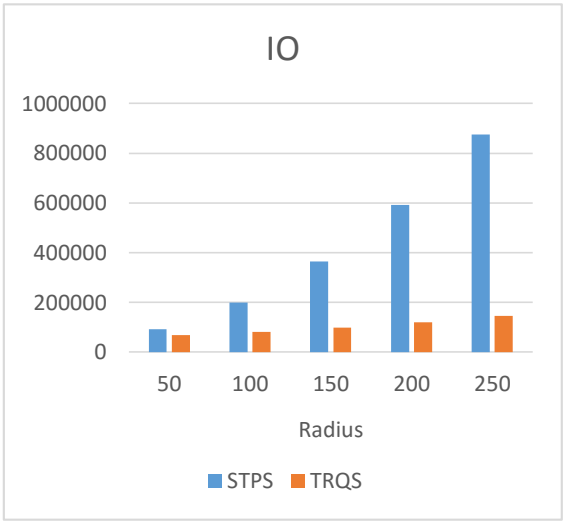
Επιπλέον, για τον έλεγχο της ορθότητας των αποτελεσμάτων δημιουργήθηκε SQL επερώτημα που υλοποιούσε τη λογική της αναζήτησης που μελετάται από τη παρούσα εργασία.

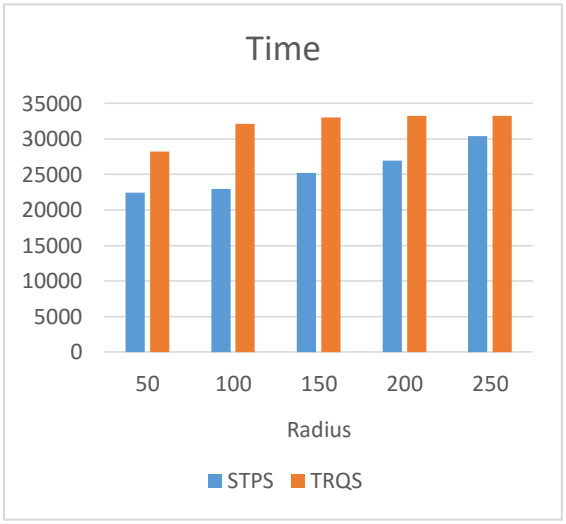
Τα βήματα και ο κώδικας του συγκεκριμένου SQL επερωτήματος παρουσιάζονται στο παράστημα Α.

6.5 *Αποτελέσματα*

6.5.1 *Ακτίνα R*

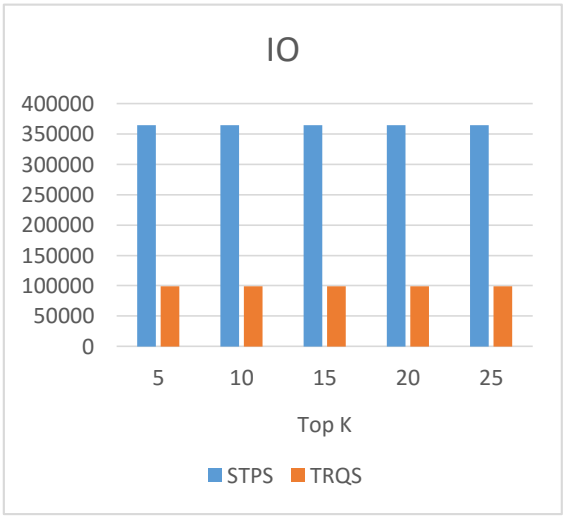
Στη πρώτη μέτρηση μεταβάλλεται η παράμετρος της ακτίνας.

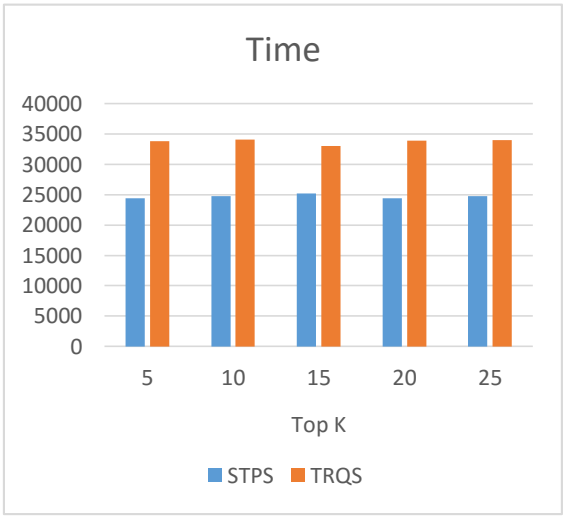




6.5.2 *Top k αποτελέσματα*

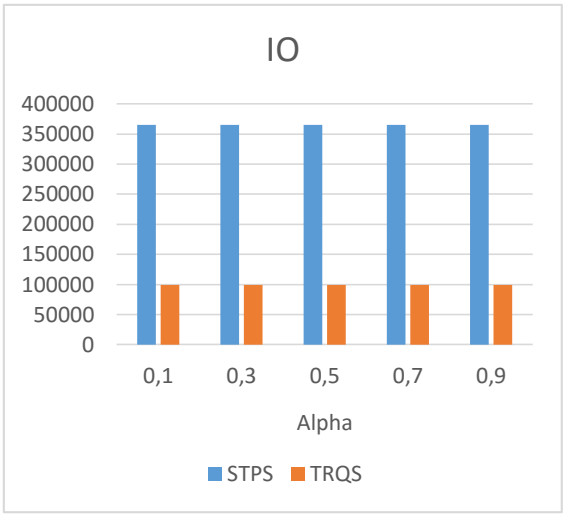
Στη δεύτερη μέτρηση μεταβάλλεται το πλήθος των αποτελεσμάτων που θα επιστρέψει ο αλγόριθμος.

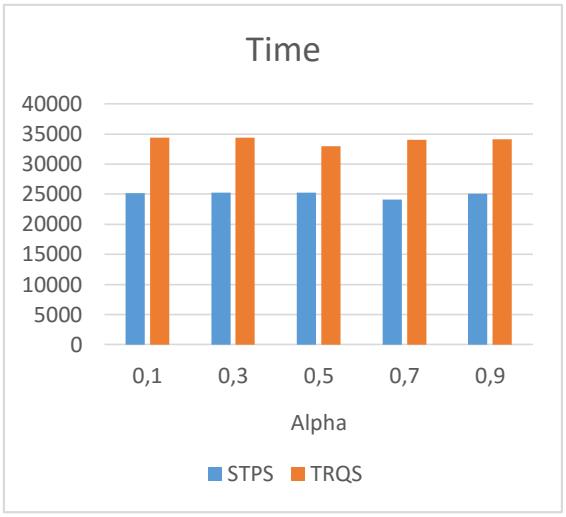




6.5.3 *Alpha*

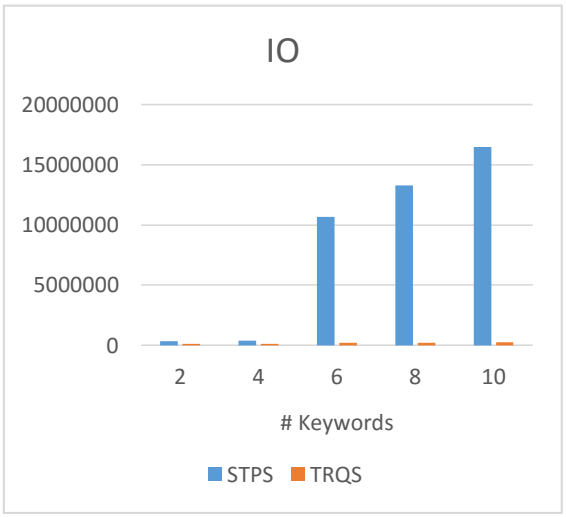
Στη τρίτη μέτρηση μεταβάλλεται η τιμή της παραμέτρου εξομάλυνσης ανάμεσα στο λεκτικό και χωρικό σκορ κάθε σημείου ενδιαφέροντος.

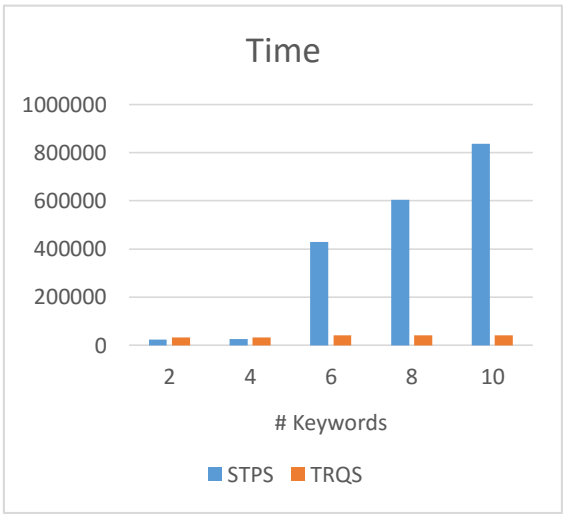




6.5.4 *Πλήθος keywords*

Στη τέταρτη μέτρηση μεταβάλλεται το πλήθος των λέξεων κλειδιά που εισάγει ο χρήστης. Επιλέχθηκαν λέξεις κλειδιά με την μεγαλύτερη συχνότητα εμφάνισης





6.6 Σύνοψη συμπερασμάτων αξιολόγησης

Από τα αποτελέσματα, γίνεται αντιληπτό πως ο αλγόριθμος του STPS έχει πολύ απρόβλεπτα αποτελέσματα. Για μικρές τιμές μεταβλητών, έχει μικρότερες τιμές χρόνου απόκρισης αλλά πολύ μεγαλύτερες τιμές στο πλήθος πρόσβασης στα ευρετήρια (IO). Αυτό μας δείχνει ότι επηρεάζεται σε μεγάλο βαθμό από την επιλογή των λέξεων κλειδιά – όσο πιο γενικές, κοινές είναι οι λέξεις που θα επιλέξουμε, αυξάνεται εκθετικά ο χρόνος απόκρισης. Όμοια συμπεριφορά παρουσιάζει και με την αύξηση της ακτίνας. Αυτό συμβαίνει γιατί ο αλγόριθμος λειτουργεί σε δύο διακριτά βήματα : α) την επιλογή κατάλληλου συνδυασμού feature object με βέλτιστο λεκτικό score, β) την εύρεση του καταλληλότερου σημείου data object για τον συνδυασμό. Αν το πρώτο βήμα επιστρέψει πολλούς έγκυρους συνδυασμούς, αυξάνεται το πλήθος των προσβάσεων στα δέντρα του συνόλου data objects.

Ως αποτέλεσμα ο αλγόριθμος έχει καλύτερους χρόνους απόκρισης για λίγες λέξεις κλειδιά ή μικρή ακτίνα, εφόσον το βήμα (α) επιστρέφει λίγους συνδυασμούς.

Σε αντίθεση ο αλγόριθμος του TRQS επηρεάζεται λιγότερο από τις τιμές των παραμέτρων, και παρουσιάζει πιο αναμενόμενα αποτελέσματα. Επίσης η συσχέτιση ανάμεσα στα αποτελέσματα του IO και του χρόνου απόκρισης είναι πιο ομαλή, γεγονός που μας δείχνει ότι ο αλγόριθμος είναι πιο επεκτάσιμος (scalable) από τον STPS.

7

Τεχνικές λεπτομέρειες

7.1 Λεπτομέρειες υλοποίησης

Η παρούσα εργασία περιλάμβανε δύο στάδια υλοποίησης εφαρμογής: α) υλοποίηση εφαρμογών για τη συλλογή δεδομένων, β) υλοποίηση των ευρετηρίων και των αλγορίθμων αναζήτησης.

7.1.1 Εφαρμογές συλλογής δεδομένων

Υλοποιήθηκαν δύο ειδών εφαρμογές. Η πρώτη περίπτωση αφορά την εκτέλεση κλήσεων σε ιστοσελίδες. Χρησιμοποιήθηκε η βιβλιοθήκη JSoup για την εξαγωγή δεδομένων από ιστοσελίδα χρησιμοποιώντας CSS query. Η χρήση της βιβλιοθήκης είναι εξαιρετικά απλή όπως φαίνεται και από το ακόλουθο παράδειγμα – ο κώδικας κάνει κλήση στη σελίδα με τα αποτελέσματα. Έπειτα αναζητά τα αντικείμενα του html με την πληροφοριά που θέλουμε με τη μέθοδο doc.select. Τέλος, αποθηκεύει σε αντικείμενο τα στοιχεία κάθε εγγραφής.

```
Document doc = Jsoup
.connect("http://www.athinorama.gr/clubbing/guide.aspx?zid=1&ath=0")
.post();
Elements _elements = doc.select("#listtable tr.trspacer");
for (Element e : _elements)
{
    Bar a = new Bar();
    a.name = e.select(".placename").text();
    a.url = "www.athinorama.gr/clubbing/"+ e.select("a").attr("href");
    a.locationText = e.select("a").text().replace(a.name, "").trim();
    a.barCategory = e.select(".greyelement").text();
    lst.add(a);
}
```

Η δεύτερη περίπτωση εφαρμογής υλοποιήθηκε σε .Net και χρησιμοποιήθηκε για την κλήση rest web service. Η παραπάνω λύση προτιμήθηκε για να παρακαμφθούν οι τεχνικές ασφαλείας (captcha) στη περίπτωση της ιστοσελίδας που επιλέχθηκε.

7.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Η υλοποίηση των αλγορίθμων πραγματοποιήθηκε σε γλώσσα προγραμματισμού JAVA και συγκεκριμένα χρησιμοποιήθηκε το τελευταίο διαθέσιμο Java Development Kit (JDK) 1.8. Η ανάπτυξη έγινε στο Eclipse IDE, στην έκδοση MARS.2 (4.5.2).

Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες:

- JDBM - βιβλιοθήκη αποθήκευσης δεδομένων σε Btree δομή σε δίσκο
- NeuStore – πειραματική βιβλιοθήκη αποθήκευσης δομών ευρετηρίου στον δίσκο
- SpatialIndex – βιβλιοθήκη που υλοποιεί τη δομή του R*Tree

Για την υλοποίηση των εφαρμογών συλλογής δεδομένων χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες:

- JSoup – βιβλιοθήκη που την κλήση URL και εκτέλεση ερωτημάτων σε γλώσσα CSS
- Newtonsoft.Json – βιβλιοθήκη για την μετατροπή json σε αντικείμενα

Για την εκτέλεση της εφαρμογής είναι απαραίτητο να υπάρχει εγκατεστημένο το Java Runtime Environment (JRE) στον υπολογιστή του χρήστη, ενώ για την δοκιμή και μεταγλώττιση του κώδικα προτείνεται η εγκατάσταση JDK και κάποιου java IDE (πχ Eclipse IDE).

Τα βήματα για την επεξεργασία του κώδικα και εκτέλεση μέσω του Eclipse είναι τα ακόλουθα:

1. Κάνουμε import το project μέσω του μενού **File > Import… > Existing Projects into workspace**.
2. Μέσω του αρχείου ***config.properties***, μεταβάλλουμε τις ρυθμίσεις που επιθυμούμε. Σε αυτό το αρχείο ορίζουμε την θέση των αρχείων ευρετηρίου, τις παραμέτρους alpha – topK – alpha και τα keywords ανά feature set
3. Έπειτα εκτελούμε τον αγλόριθμο που επιθυμούμε ανοίγοντας το αντίστοιχο αρχείο: ***ThreeRangeScoreParser.java***

STPS.java

Υπάρχει η δυνατότητα να δημιουργηθούν εκ νέου τα index μέσω του αρχείου ***Indexer.java***. Θα πρέπει πρώτα να διαμορφώσουμε κατάλληλα το αρχείο ***SQLRawData.java***, το οποίο είναι υπεύθυνο για την ανάγνωση των δεδομένων από τη βάση δεδομένων.

8

Επίλογος

Στη μελέτη ερευνήθηκαν νέοι τρόποι δημιουργίας ευρετηρίων, που αυξάνουν την πληροφορία που αποθηκεύεται σε ένα ευρετήριο, με σκοπό να εξυπηρετηθούν πιο σύνθετες μορφές ερωτημάτων.

8.1 Σύνοψη και συμπεράσματα

Η συγκριτική μελέτη των δύο αλγορίθμων αναζήτησης που παρουσιάστηκαν, κάνει αρχικά εμφανές πως πρόκειται για ένα ακριβό είδος επερωτήματος. Οι δυνατοί συνδυασμοί αυξάνονται εκθετικά όταν η τιμή του πλήθους των feature object αυξάνεται. Ακόμη και στη περίπτωση που μελετήθηκε, όπου είχαμε 2 feature data sets με περίπου 30k αντικείμενα, οι δυνατοί συνδυασμοί είναι 27 τρισεκατομμύρια. Η ύπαρξη ευρετηρίου είναι επιτακτική όπως επίσης και η υλοποίηση βέλτιστων αλγορίθμων αναζήτησης.

Από τα αποτελέσματα των μετρήσεων, φαίνεται πως οι μεταβλητές που επηρεάζουν περισσότερο τον χρόνο εκτέλεσης των αλγορίθμων είναι το πλήθος των λέξεων κλειδιά και η ακτίνα r. Με δεδομένο ότι επιστρέφονται μόνο σημεία ενδιαφέροντος με τιμή jaccard μεγαλύτερη του μηδέν, όσο αυξάνονται οι λέξεις κλειδιά αυξάνονται και τα υποψήφια σημεία ενδιαφέροντος. Ομοίως όσο αυξάνεται η ακτίνα r, αυξάνονται οι πιθανοί συνδυασμοί.

Η μεταβλητές που επηρεάζουν λιγότερο το χρόνο εκτέλεσης είναι η τιμή alpha και η τιμή top k. Αυτό οφείλεται στο ότι και οι δύο τιμές επηρεάζουν το αποτέλεσμα αφού εντοπιστούν οι συνδυασμοί feature object που καλύπτουν τα λεκτικά και χωρικά κριτήρια.

8.2 Μελλοντικές επεκτάσεις

Στη παρούσα εργασία μελετήσαμε συγκεκριμένα την εύρεση σημείων ενδιαφέροντος με λεκτικά κριτήρια για 2 κατηγορίες ενδιαφέροντος. Ο κώδικας δύναται να επεκταθεί για δυναμικό πλήθος κατηγορών. Επίσης, θα μπορούσε να μεταβληθεί ο κώδικας ώστε να υποστηρίζει κατανεμημένη εκτέλεση των αλγορίθμων του σε διάφορα επίπεδα (παράλληλη διάσχιση στο ίδιο δέντρο ή/και σε όλα τα δέντρα).

9

Βιβλιογραφία

A. Khodaei, C. Shahabi, and C. Li. 2010. Hybrid indexing and seamless ranking of spatial and textual features of web documents. 2010.

Ariel Cary, Ouri Wolfson, Naphtali Rishe. 2010. Efficient and Scalable Method for Processing Top-k Spatial Boolean Queries. 2010.

Beckmann Norbert, Kriegel Hans-Peter, Schneider Ralf, Seeger Bernhard. 1990. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. 1990.

Comer, Douglas. 1979. The ubiquitous B-tree. 1979.

D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. 2012. Joint top-k spatial keyword query processing. 2012.

Dinesh P. Mehta, Sartaj Sahni. 2004. Handbook of Data Structures and Applications. 2004.

Dongxiang Zhang, Kian-Lee Tan, Anthony K. H. Tung. 2013. Scalable Top-K Spatial Keyword Search. 2013.

Gao Cong, Christian S. Jensen, Dingming Wu. 2009. Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. 2009.

George Tsatsanifos, Akrivi Vlachou. 2015. On Processing Top-k Spatio-Textual Preference Queries. 2015.

Greene, Diane. 1990. An Implementation and Performance Analysis of Spatial Data Access Methods. 1990.

Guttman, Antonin. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. 1984.

I. D. Felipe, V. Hristidis, N. Rishe. 2008. Keyword search on spatial databases. 2008.

Joao B. Rocha Junior, Akrivi Vlachou, Christos Doulkeridis, Kjetil Nørvag. 2010. Efficient Processing of Topk Spatial Preference Queries. 2010.

Joao B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, Kjetil Nørvag. 2011. Efficient Processing of Top-k Spatial Keyword Queries. 2011.

Knuth, D. 1973. The Art of Computer Programming. 1973.

Lisi Chen, Gao Cong, Christian S. Jensen, Dingming Wu. 2013. Spatial Keyword Query Processing: An Experimental Evaluation. 2013.

Lisi Chen, Yan Cui, Gao Cong, Xin Cao. 2014. SOPS: A System for Efficient Processing of Spatial-Keyword Publish/Subscribe. 2014.

Man Lung Yiu, Xiangyuan Dai, Nikos Mamoulis,Michail Vaitis. 2007. Top-k Spatial Preference Queries. 2007.

Nahar, Kanthale Deepak B. Prateek. 2014. IR-Tree Implementation using Index Document Search Method . 2014.

R. Bayer, E.M. McCreight. 1972. Organization and maintenance of large ordered indexes. 1972.

R. Elmasri, S.B. Navathe. 2000. Fundamentals of Database Systems. 2000.

Ramaswamy Hariharan, Bijit Hore, Chen Li, Sharad Mehrotra. 2007. Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems. 2007.

S. Vaid, C. B. Jones, H. Joho, and M. Sanderson. 2005. Spatio-textual indexing for geographical search on the web. 2005.

Santhanakrishnan C, Sivaprakasam V, Rajasekar R, Sudhakar D, Lourdu Michael Antony K. 2013. An Efficient Query Processing Search Using Textual and Spatial Relevances. 2013.

Xin Cao, Gao Cong, Christian S. Jensen. 2010. Retrieving Top-k Prestige-Based Relevant Spatial Web Objects. 2010.

Xin Cao, Gao Cong, Christian S. Jensen, Jun Jie Ng, Beng Chin Ooi, Nhan Tue Phan, Dingming Wu. 2012. SWORS: A System for the Efficient Retrieval of Relevant Spatial Web Objects. 2012.

Y. Zhou, X. Xie, C. Wang, Y. Gong, W.-Y. Ma. 2005. Hybrid index structures for location-based web search. 2005.

Zhisheng LI, Ken C.K. LEE, Baihua ZHENG, Wang-Chien LEE, Dik Lun LEE. 2011. IR-tree: An Efficient Index for Geographic Document Search. 2011.

Παράρτημα Α: SQL Query

Για τον έλεγχο της ορθότητας των αποτελεσμάτων των αλγορίθμων που παρουσιάστηκαν, δημιουργήθηκε ένα sql query που επίλυε το πρόβλημα που πραγματεύεται η εργασία.

Τα βήματα που ακολουθεί ο κώδικας είναι συνοπτικά τα εξής:

1. Δημιουργήθηκε πίνακας με στήλες: [id], [γεωγ. μήκος], [γεωγ. πλάτος], [εγγραφή], [κατηγορία] για όλες τις εγγραφές των δεδομένων.
2. Δημιουργήθηκε inverted index πίνακα (α) με στήλες [docID], [word], [count] πάνω στο οποίο υλοποιήθηκε sql index στο πεδίο της λέξης.
3. Εισάγονται σε προσωρινούς πίνακες (Qi) τα query words ανά κατηγορία
4. Γίνεται συνένωση του IIndex πίνακα με κάθε πίνακα query words και αποθηκεύονται σε προσωρινό πίνακα τα αποτελέσματα (docID, jaccard) ανά κατηγορία {Υπολογίζεται η τιμή jaccard ανά object}
5. Εισάγονται τα αποτελέσματα ανά κατηγορία σε νέο προσωρινό πίνακα που περιέχει τις συντεταγμένες και τις τιμές jaccard για τα query words
6. Για κάθε προσωρινό πίνακα δημιουργείται spatial index
7. Έπειτα εκτελείται cross apply του πρώτου πίνακα (data objects) με το δεύτερο (feature 1 objects) και φιλτράρεται με την απόσταση R {καθώς το where χρησιμοποιεί συντεταγμένες, χρησιμοποιείται αποκλειστικά το spatial index}
8. Έπειτα εκτελείται cross apply του πίνακα από το βήμα 7 με το τρίτο πίνακα (feature 2 object) και φιλτράρεται με την απόσταση R
9. Στο πίνακα με τα τελικά αποτελέσματα υπολογίζεται το overall score (α*(distance1+distance2)+(1-a)/(jaccard1) + (1-a)/(jaccard2)
10. Επιστρέφονται μοναδικά object με το καλύτερο score σε φθίνουσα ταξινόμηση

```
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
--SET NOCOUNT ON ;
DECLARE @start DATETIME2,@end DATETIME2;
DECLARE @r int = 150;
DECLARE @a FLOAT = 0.5;
DECLARE @qt1 TABLE ( qword NVARCHAR(20));
DECLARE @qt2 TABLE ( qword NVARCHAR(20));

--1) Εισαγωγή σε table parameter τα keywords. 1 table ανά feature set.
INSERT INTO @qt1( qword )
VALUES ( N'night'), (N'clubs'), (N'μπαρ');
```

```
INSERT INTO @qt2( qword )
VALUES ( N'γυρος'), (N'σουβλακια'), (N'μεζεδοπωλεια'), (N'πιτσα');

--2) Για κάθε FeatureSet
       --Υπολογισμός των document που κάνουν Join με τα qWords και υπολογισμός του
jaccard τους
SET @start = SYSDATETIME();
IF OBJECT_ID('tempdb..#st01') IS NOT NULL DROP TABLE #st01;
SELECT i.docID, COUNT(i.docID) q1WordsDistinctAppearences, SUM(i.cnt)
q1WordsAppearences,
(SELECT CAST(COUNT(ii.docID) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)
docDistinctWords1,
(SELECT CAST(sum(ii.cnt) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)
docWords1,
CAST(COUNT(i.docID) AS FLOAT)/((SELECT CAST(COUNT(*) AS FLOAT) FROM @qt1)
+(SELECT CAST(COUNT(ii.docID) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)-
CAST(COUNT(i.docID) AS FLOAT)) distinctJaccard1,
CAST(SUM(i.cnt) AS FLOAT)/(SELECT CAST(sum(ii.cnt) AS FLOAT) FROM wordIndex ii WHERE
i.docID = ii.docID) Jaccard1
INTO #st01
FROM wordIndex i
JOIN @qt1 t ON i.word = t.qword
GROUP BY i.docID

IF OBJECT_ID('tempdb..#st02') IS NOT NULL DROP TABLE #st02;
SELECT i.docID, COUNT(i.docID) q2WordsDistinctAppearences, SUM(i.cnt)
q2WordsAppearences,
(SELECT CAST(COUNT(ii.docID) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)
docDistinctWords2,
(SELECT CAST(sum(ii.cnt) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)
docWords2,
CAST(COUNT(i.docID) AS FLOAT)/((SELECT CAST(COUNT(*) AS FLOAT) FROM @qt2)
+(SELECT CAST(COUNT(ii.docID) AS FLOAT) FROM wordIndex ii WHERE i.docID = ii.docID)-
CAST(COUNT(i.docID) AS FLOAT)) distinctJaccard2,
CAST(SUM(i.cnt) AS FLOAT)/(SELECT CAST(sum(ii.cnt) AS FLOAT) FROM wordIndex ii WHERE
i.docID = ii.docID) Jaccard2
INTO #st02
FROM wordIndex i
JOIN @qt2 t ON i.word = t.qword
GROUP BY i.docID´´


--3.1) Εισαγωγή σε temp πίνακα όλων των Object
IF OBJECT_ID('tempdb..#st1') IS NOT NULL DROP TABLE #st1;
SELECT hot.ID hotID, hot.doc hotDoc, hot.longitude hotLongitude,hot.latitude
hotLatitude,
geometry::STGeomFromText('POINT ('+CAST(CAST(Longitude AS decimal(13, 2)) AS
varchar)+' '+CAST(CAST(latitude AS decimal(13, 2)) AS varchar)+')',4121) hotGeom
INTO #st1
FROM data hot
where hot.category = 'Διαμονή'

--3.2) Δημιουργία spatial index πάνω στο temp
ALTER TABLE #st1 ADD CONSTRAINT [PK_st21_ID] PRIMARY KEY CLUSTERED( hotID ASC);
CREATE SPATIAL INDEX si_st21_hotGeom
ON #st1(hotGeom)
WITH(BOUNDING_BOX = (xmin=126331, ymin=3864188, xmax=880228, ymax=4621205));

--4.1)Για κάθε FeatureSet: Εισαγωγή σε temp πίνακα όλων των feature POI και
δημιουργία spatial index
IF OBJECT_ID('tempdb..#st2') IS NOT NULL DROP TABLE #st2;
```

```
SELECT bar.ID barID, bar.doc barDoc, bar.longitude barLongitude,bar.latitude
barLatitude ,
geometry::STGeomFromText('POINT ('+CAST(CAST(Longitude AS decimal(13, 2)) AS
varchar)+' '+CAST(CAST(latitude AS decimal(13, 2)) AS varchar)+')',4121) barGeom,
s.q1WordsDistinctAppearences,s.q1WordsAppearences,s.docDistinctWords1,s.docWords1,s.
distinctJaccard1,s.Jaccard1
INTO #st2
FROM data bar
JOIN #st01 s ON bar.ID = s.docID
where bar.category = 'Διασκέδαση'


--4.2) Δημιουργία spatial index πάνω στο temp
ALTER TABLE #st2 ADD CONSTRAINT [PK_st22_ID] PRIMARY KEY CLUSTERED( barID ASC);
CREATE SPATIAL INDEX si_st22_barGeom
ON #st2(barGeom)
WITH(BOUNDING_BOX = (xmin=126331, ymin=3864188, xmax=880228, ymax=4621205));

--5.1) Εισαγωγή σε table parameter τα keywords. 1 table ανά feature set.
IF OBJECT_ID('tempdb..#st3') IS NOT NULL DROP TABLE #st3;
SELECT fag.ID fagID, fag.doc fagDoc, fag.longitude fagLongitude,fag.latitude
fagLatitude,
geometry::STGeomFromText('POINT ('+CAST(CAST(Longitude AS decimal(13, 2)) AS
varchar)+' '+CAST(CAST(latitude AS decimal(13, 2)) AS varchar)+')',4121) fagGeom,
s.q2WordsDistinctAppearences,s.q2WordsAppearences,s.docDistinctWords2,s.docWords2,s.
distinctJaccard2,s.Jaccard2
INTO #st3
FROM data fag
JOIN #st02 s ON fag.ID = s.docID
where fag.category = 'Φαγητό'

--5.2) Δημιουργία spatial index πάνω στο temp

ALTER TABLE #st3 ADD CONSTRAINT [PK_st23_ID] PRIMARY KEY CLUSTERED( fagID ASC);
CREATE SPATIAL INDEX si_st23_fagGeom
ON #st3(fagGeom)
WITH(BOUNDING_BOX = (xmin=126331, ymin=3864188, xmax=880228, ymax=4621205));

--6) Συνδυασμός όλων των αποτελεσμάτων του συνδυασμών data object με feature object
1
--με τα feature object 2 που καλύπτουν τον χωρικό περιορισμό

IF OBJECT_ID('tempdb..#st4') IS NOT NULL DROP TABLE #st4;
SELECT *, hot.hotGeom.STDistance(bar.barGeom) hotBarDist
INTO #st4
FROM #st1 hot
CROSS APPLY #st2 bar
WHERE hot.hotGeom.STDistance(bar.barGeom) BETWEEN 0 AND @r;

--6.2) Δημιουργία spatial index πάνω στο temp
ALTER TABLE #st4 ADD CONSTRAINT [PK_st24_ID] PRIMARY KEY CLUSTERED( hotID ASC,barID
ASC);
CREATE SPATIAL INDEX si_st24_hotGeom
ON #st4(hotGeom)
WITH(BOUNDING_BOX = (xmin=126792, ymin=3864449, xmax=879772, ymax=4621005));

--7) Συνδυασμός όλων των αποτελεσμάτων του προηγούμενου ερωτήματος
--με τα feature object 2 που καλύπτουν τον χωρικό περιορισμό
IF OBJECT_ID('tempdb..#st5') IS NOT NULL DROP TABLE #st5;
SELECT * , hotGeom.STDistance(fag.fagGeom) hotFagDist
INTO #st5
FROM #st4 st1
```

```
CROSS APPLY #st3 fag
WHERE st1.hotGeom.STDistance(fag.fagGeom) BETWEEN 0 AND @r;

--8)Εισαγωγή των αποτελεσμάτων στον τελικό πίνακα
--Στα αποτελέσματα περιέχονται πολλαπλές φορές τα data object (με κάθε έγκυρο
συνδυασμό feature object)
IF OBJECT_ID('tempdb..#st6') IS NOT NULL DROP TABLE #st6;
SELECT
@a*(hotBarDist+hotFagDist) + (1-@a)*( 1/r.distinctJaccard1 + 1/r.distinctJaccard2)
overallScore,
r.distinctJaccard1 barjac,
r.distinctJaccard2 fagjac,
@a*(hotBarDist+hotFagDist) spatialScore,
(1-@a)*( 1/r.distinctJaccard1 + 1/r.distinctJaccard2) textualScore,
hotBarDist,hotFagDist, fagGeom.STDistance(barGeom) barFagDist,
hotID,
barID,
fagID
INTO #st6
FROM #st5 r
ORDER BY 1;

--1) Εισαγωγή στο πίνακα των αποτελεσμάτων των καλύτερων συνδυασμών ανά data object

IF OBJECT_ID('tempdb..#result') IS NOT NULL DROP TABLE #result;
SELECT *,RANK() OVER (PARTITION BY hotId ORDER BY overallScore,barID,hotID) rnk
INTO #result
FROM #st6

SELECT rnk,hotID, barID, fagID,overallScore ,hotBarDist, hotFagDist,barjac,fagjac ,
textualScore
FROM #result
where rnk = 1
ORDER BY overallScore

SET @end = SYSDATETIME();
PRINT DATEDIFF(MILLISECOND, @start,@end);
```

Παράρτημα Β: Ψευδοκώδικας R-Tree

Αλγόριθμος Search
Δεδομένου ενός R tree με ρίζα T, εντοπισμός όλων των εγγραφών με ορθογώνια τα οποία επικαλύπτουν ένα ορθογώνιο αναζήτησης S.

```
[1 Αναζήτηση υπο-δένδρων]
     Εάν το T δεν είναι φύλλο
           Για κάθε εγγραφή E, εάν το EI επικαλύπτει το S,
           κάλεσε την Search για το δένδρο με ρίζα
           τον κόμβο στον οποίο δείχνει το Ep.
     Αλλιώς
[2 Αναζήτηση φύλλων]
           Για κάθε εγγραφή E, εάν το EI επικαλύπτει το S, επέστρεψε το E.
     Τέλος
```

Αλγόριθμος Insert
Εισαγωγή μιας νέας εγγραφής E στη δομή του R tree

```
[1 Εύρεση της θέσης που θα αποθηκευτεί η νέα εγγραφή]
     Κάλεσε την ChooseLeaf για την επιλογή ενός κόμβου-φύλλου L
     στο οποίο θα αποθηκευτεί η εγγραφή E.
[2 Προσθήκη της εγγραφής στον κόμβο-φύλλο]
     Εάν το L έχει χώρο για μια ακόμη εγγραφή
           Αποθήκευσε το E.
     Αλλιώς
           Κάλεσε την SplitNode για τη δημιουργία των L και LL που θα
           περιλαμβάνουν το E και όλες τις εγγραφές που υπήρχαν στο L.
     Τέλος
[3 Διάδοση των αλλαγών προς τα πάνω]
     Εάν δεν έγινε διάσπαση
           Κάλεσε την AdjustTree με όρισμα το L.
     Αλλιώς
           Κάλεσε την AdjustTree με όρισμα το LL.
     Τέλος
[4 Αύξηση του ύψους του δένδρου]
     Εάν το I3 οδήγησε σε διάσπαση της ρίζας
           Δημιούργησε μια νέα ρίζα με παιδιά του δύο νέους
           κόμβους που προέκυψαν μετά τη διάσπαση.
     Τέλος
```

Αλγόριθμος ChooseLeaf
Επιλογή του κόμβου-φύλλου που θα αποθηκευτεί η νέα εγγραφή E

```
[1 Αρχικοποίηση]
     Θέσε το N να είναι η ρίζα.
[2 Έλεγχος για φύλλο]
     Εάν το N είναι φύλλο
           Επέστρεψε το N.
     Αλλιώς
[3 Επιλογή υπο-δένδρου]
           Επέλεξε την εγγραφή στο N της οποίας το ορθογώνιο απαιτεί τη
           μικρότερη αύξηση επιφάνειας για να συμπεριλάβει το νέο δεδομένο.
           Σε περίπτωση ισοπαλίας (δύο κατάλληλα ορθογώνια) επέλεξε το
ορθογώνιο
           με τη μικρότερη επιφάνεια
     Τέλος
[4 Καθοδική διάσχιση του δένδρου έως ότου οδηγηθούμε σε κόμβο φύλλο]
     Θέσε το N να είναι ο κόμβος παιδί στον οποίον "δείχνει" ο δείκτης της
     επιλεγμένης εγγραφής και επανέλαβε από το CL2
```

Αλγόριθμος AdjustTree
Ανοδική διάσχιση του δένδρου από ένα κόμβο-φύλλο L προς τη ρίζα, ρυθμίζοντας τα Ορθογώνια Κάλυψης των εσωτερικών κόμβων και μεταδίδοντας τις διασπάσεις κόμβων όποτε είναι απαραίτητο.

```
[1 Αρχικοποίηση]
     Θέσε το N να είναι το φύλλο L.
     Εάν το L διασπάστηκε προηγουμένως
           Θέσε το NN να είναι το δεύτερο φύλλο που προέκυψε από τη διάσπαση.
Τέλος
[2 Έλεγχος τερματισμού]
     Εάν το N είναι η ρίζα
           Σταμάτησε.
     Αλλιώς
[3 Προσαρμογή του Ορθογωνίου Κάλυψης του κόμβου-γονέα]
     Θεώρησε το P να είναι ο κόμβος-γονέας του N και το E_N η εγγραφή του
     N στο P. Ρύθμισε το E_NI έτσι ώστε να περιλαμβάνει ακριβώς όλα τα ορθογώνια
     των εγγραφών του N.
[4 Διάδοση της διάσπασης κόμβων προς τα πάνω]
     Εάν το N διασπάστηκε προηγουμένως
           Δημιούργησε μια νέα εγγραφή με το pE να δείχνει στο NN και το IE NN
           NN NN να περιλαμβάνει όλα τα ορθογώνια στο NN.
           Εάν υπάρχει διαθέσιμος χώρος
                 Πρόσθεσε την εγγραφή  στο P.
           Αλλιώς
                 Κάλεσε την SplitNode για τη δημιουργία των P και PP που θα
                 περιλαμβάνουν την εγγραφή E και όλες τις εγγραφές που υπήρχαν
                 στο P.
           Τέλος
     Τέλος
[5 Άνοδος στο επόμενο επίπεδο του δένδρου]
     Εάν δεν προκλήθηκε διάσπαση
           Θέσε το N να είναι ο κόμβος P και επανέλαβε από το 2.
     Αλλιώς
           Θέσε το N να είναι ο κόμβος P και το NN να είναι ο κόμβος PP και
           επανέλαβε από το AT2.
     Τέλος
Τέλος
```

Αλγόριθμος QuadraticSplit
Διαίρεση ενός συνόλου M+1 εγγραφών σε δύο ομάδες

[1 Επιλογή της πρώτης εγγραφής για κάθε ομάδα]
Κάλεσε την PickSeeds για την επιλογή δύο εγγραφών που θα είναι τα πρώτα στοιχεία των ομάδων. Αντιστοίχισε κάθε μια από αυτές σε μια ομάδα.
[2 Έλεγχος ολοκλήρωσης]
Εάν όλες οι εγγραφές έχουν αντιστοιχιστεί
Σταμάτησε.
[3 Αντιστοίχιση εγγραφών στην ομάδα που χρειάζεται όλες τις εγγραφές ώστε να περιλαμβάνει τον ελάχιστο αριθμό m εγγραφών]
Αλλιώς, εάν μια ομάδα περιλαμβάνει τόσες λίγες εγγραφές ούτως ώστε θα πρέπει
να αντιστοιχιστούν σε αυτή όλες οι υπόλοιπες για να φτάσει τον ελάχιστο αριθμό εγγραφών m
Αντιστοίχισε τες σε αυτή την ομάδα και σταμάτησε.
Τέλος
[3 Επιλογή της εγγραφής που θα αντιστοιχιστεί σε ομάδα]
Κάλεσε την PickNext για την επιλογή της επόμενης εγγραφής που θα αντιστοιχιστεί σε μια ομάδα. Πρόσθεσε την εγγραφή στην ομάδα που απαιτεί την ελάχιστη αύξηση της επιφάνειας του Ορθογωνίου Κάλυψης της, ώστε να συμπεριλάβει την εγγραφή. Στην περίπτωση της ισοπαλίας, επέλεξε την ομάδα με τη μικρότερη επιφάνεια και κατόπιν αυτή με τις λιγότερες εγγραφές. Επανέλαβε από το 2.

Αλγόριθμος PickSeeds
Επιλογή δύο εγγραφών που θα αποτελέσουν τα πρώτα στοιχεία των ομάδων
[1 Υπολογισμός της κακής απόδοσης εάν ομαδοποιηθούν δύο εγγραφές]
Για κάθε ζεύγος εγγραφών E1 και E2
Δημιούργησε το ορθογώνιο J που περικλείει τα ορθογώνια E1I, E2I
και υπολόγισε το d = area(J)-area(E1I)-area(E2I)
[2 Επιλογή του χειρότερου ζεύγους]
Επέλεξε το ζεύγος ορθογωνίων με τη μεγαλύτερη τιμή του d.

Αλγόριθμος PickNext
Επιλογή μιας εκ των υπολοίπων εγγραφών για αντιστοίχιση σε κάποια ομάδα
[1 Προσδιορισμός του κόστους αντιστοίχισης κάθε εγγραφής σε κάθε ομάδα]
Για κάθε εγγραφή E που δεν έχει αντιστοιχιστεί σε κάποια ομάδα
Υπολόγισε το d1 ως την αύξηση της επιφάνειας του Ορθογωνίου Κάλυψης της Ομάδας 1, που απαιτείται για να συμπεριλάβει το 1 ορθογώνιο EI.
Υπολόγισε αντίστοιχα το d2 για την Ομάδα 2.
[2 Εύρεση της καταλληλότερης εγγραφής για αντιστοίχιση με κάποια ομάδα]
Επέλεξε την εγγραφή με τη μεγαλύτερη διαφορά μεταξύ των d1 και d2.

Αλγόριθμος Delete
Διαγραφή της εγγραφής E από ένα R tree.

```
[1 Εύρεση του κόμβου που περιλαμβάνει την εγγραφή]
     Κάλεσε τη FindLeaf για τον εντοπισμό του κόμβου - φύλλου L που περιλαμβάνει
     την εγγραφή E.
[Έλεγχος τερματισμού]
     Εάν η εγγραφή δε βρεθεί.
           Σταμάτησε.
     Αλλιώς
[2 Διαγραφή εγγραφής]
           Αφαίρεσε την εγγραφή E από τον κόμβο - φύλλο L.
[3 Διάδοση των αλλαγών]
           Κάλεσε την CondenseTree, με όρισμα το L.
[4 Ελάττωση του ύψους του δένδρου]
           Εάν η ρίζα έχει μονάχα ένα κόμβο - παιδί
                 Κάνε τον κόμβο – παιδί να είναι η νέα ρίζα του δένδρου.
           Τέλος
     Τέλος
```

Αλγόριθμος FindLeaf
Δεδομένου ενός R tree με ρίζα το T, εύρεση του κόμβου-φύλλου που περιλαμβάνει την
εγγραφή E.

```
[1 Αναζήτηση υπο-δένδρων]
     Εάν το T δεν είναι φύλλο
           Για κάθε εγγραφή F στο T
                 Εξέτασε εάν το ορθογώνιο FI επικαλύπτει το ορθογώνιο EI. Για
                 κάθε μια τέτοια εγγραφή κάλεσε τη FindLeaf με όρισμα το δένδρο
                 που έχει ρίζα τον κόμβο στον οποίο δείχνει το Fp έως ότου να
                 βρεθεί η εγγραφή E, ή να έχουν εξεταστεί όλες οι εγγραφές.
     Τέλος
[2 Αναζήτηση της εγγραφής σε ένα κόμβο-φύλλο]
     Εάν το T είναι φύλλο
           Για κάθε εγγραφή του
                 Εξέτασε εάν είναι η E. Εάν βρεθεί, επέστρεψε το T.
     Τέλος
```

Αλγόριθμος CondenseTree
Δεδομένου ενός κόμβου-φύλλου L από το οποίο διαγράφηκε κάποια από τις εγγραφές του, κατάργηση του κόμβου εάν έχει μη επαρκές πλήθος εγγραφών και επανατοποθέτηση
των εγγραφών του. Διάδοση της κατάργησης του κόμβου προς τα πάνω εάν είναι απαραίτητο. Προσαρμογή όλων των Ορθογωνίων Κάλυψης στη διαδρομή προς τη ρίζα, κάνοντάς τα μικρότερα εάν είναι δυνατό.

```
[1 Αρχικοποίηση]
     Θέσε το N να είναι ο κόμβος-φύλλο L. Όρισε ως Q το σύνολο των κόμβων που
     έχουν καταργηθεί και θέσε το να είναι κενό.
[2 Εύρεση της εγγραφής γονέα]
     Εάν το N είναι η ρίζα
           Πήγαινε στο 6.
     Αλλιώς
           Θεώρησε το P να είναι ο κόμβος-γονέας του N και το E_N να είναι η
     εγγραφή του κόμβου N στον κόμβο P.
[3 Κατάργησε τον κόμβο με μη επαρκές πλήθος εγγραφών]
     Εάν ο κόμβος N περιλαμβάνει λιγότερες από m εγγραφές
```

```
            Διέγραψε την εγγραφή E_N από τον κόμβο P και πρόσθεσε τον κόμβο N στο
            σύνολο Q.
      Τέλος
[4 Προσαρμογή του Ορθογωνίου Κάλυψης]
      Εάν ο κόμβος N δεν καταργήθηκε
            Προσάρμοσε το ορθογώνιο E_NI ώστε να περιλαμβάνει ακριβώς όλες τις
            εγγραφές του κόμβου N
      Τέλος
[5 Άνοδος στο επόμενο επίπεδο του δένδρου]
      Θέσε το N να είναι ο κόμβος P και επανέλαβε από το 2.
Τέλος
[6 Επανεισαγωγή των ορφανών εγγραφών]
      Επανεισήγαγε όλες τις εγγραφές των κόμβων που του συνόλου Q. Οι εγγραφές
      από κόμβους-φύλλα που έχουν καταργηθεί, επανεισάγονται στα φύλλα του
      δένδρου όπως περιγράφεται από τον αλγόριθμο Insert, αλλά οι εγγραφές από
      κόμβους ανώτερων επιπέδων πρέπει να τοποθετηθούν σε κόμβους που βρίσκονται
      υψηλότερα στο δένδρο, έτσι ώστε τα φύλλα των υποδένδρων των κόμβων αυτών,
      να βρίσκονται στο ίδιο επίπεδο με τα φύλλα του κυρίως δένδρου.
```

Παράρτημα Γ: Ψευδοκώδικας R*-Tree

Αλγόριθμος ChooseSubtree
Επιλογή του κόμβου - φύλλου που θα αποθηκευτεί η νέα εγγραφή E

```
[1 Αρχικοποίηση]
     Θέσε το N να είναι η ρίζα
[2 Έλεγχος για φύλλο]
     Εάν το N είναι φύλλο
           Επέστρεψε το N
     Αλλιώς
     [Επιλογή υπό-δένδρου]
           Εάν οι δείκτες στα παιδία του κόμβου N “δείχνουν” σε φύλλα
                 [προσδιόρισε το ελάχιστο κόστος επικάλυψης]
                 Επέλεξε την εγγραφή του N της οποίας το ορθογώνιο απαιτεί την
                 ελάχιστη αύξηση επικάλυψης για να συμπεριλάβει το νέο
                 ορθογώνιο δεδομένων. Σε περίπτωση ισοπαλίας επέλεξε την
                 εγγραφή με το ορθογώνιο που απαιτεί την ελάχιστη αύξηση της
                 επιφάνειας και κατόπιν την εγγραφή με το ορθογώνιο που έχει
                 τη μικρότερη επιφάνεια
           Εάν οι δείκτες στα παιδιά του κόμβου N δεν “δείχνουν” σε φύλλα
                 [προσδιόρισε το ελάχιστο κόστος επιφάνειας]
                 Επέλεξε την εγγραφή του N με το ορθογώνιο που απαιτεί την
                 ελάχιστη αύξηση επιφάνειας για να συμπεριλάβει το νέο
                 ορθογώνιο δεδομένων. Σε περίπτωση ισοπαλίας επέλεξε την
                 εγγραφή της οποίας το ορθογώνιο έχει τη μικρότερη επιφάνεια
           Τέλος
     [3 Καθοδική διάσχιση του δένδρου έως ότου οδηγηθούμε σε κόμβο-φύλλο]
           Θέσε το N να είναι ο κόμβος παιδί στον οποίο “δείχνει” ο δείκτης
           της επιλεγμένης εγγραφής και επανέλαβε από το 2
```

[προσδιόρισε το σχεδόν ελάχιστο κόστος επικάλυψης]
Ταξινόμησε τα ορθογώνια στο N κατά αύξουσα σειρά με βάση την απαιτούμενη αύξηση επιφάνειας ώστε να συμπεριλάβουν το νέο ορθογώνιο δεδομένων.
Έστω A είναι η ομάδα των πρώτων p εγγραφών

Από τις εγγραφές στην ομάδα A, θεωρώντας όλες τις εγγραφές στο N, επέλεξε την εγγραφή της οποίας το ορθογώνιο απαιτεί τη μικρότερη αύξηση επικάλυψης. Σε περίπτωση ισοπαλίας, λειτούργησε όπως προηγούμενα.

Αλγόριθμος Split
Διαίρεση ενός συνόλου M+1 εγγραφών σε δύο ομάδες

```
[1 Επιλογή άξονα διάσπασης]
     Κάλεσε την ChooseSplitAxis
[2 Επιλογή της κατανομής των δύο ομάδων κατά μήκος του άξονα διάσπασης]
     Κάλεσε την ChooseSplitIndex
[3 Κατανομή των ομάδων]
     Κατένειμε τις εγγραφές σε δύο ομάδες
```

Αλγόριθμος ChooseSplitAxis
Επιλογή του άξονα διάσπασης, κάθετα στον οποίο θα γίνει η κατανομή των εγγραφών

[1 Υπολογισμός των κατανομών]
Για κάθε άξονα
Ταξινόμησε τις εγγραφές κατά την κατώτατη και κατόπιν κατά την ανώτατη τιμή των ορθογωνίων τους και προσδιόρισε όλες τις κατανομές όπως περιγράφεται παραπάνω. Υπολόγισε το S, ως το άθροισμα όλων των τιμών περιθωρίου για όλες τις διαφορετικές κατανομές.
[2 Επιλογή άξονα διάσπασης]
Επέλεξε τον άξονα με το ελάχιστο S ως άξονα διάσπασης

Αλγόριθμος ChooseSplitIndex
Κατανομή των M+1 εγγραφών στις δύο ομάδες, βάσει του άξονα διάσπασης

[1 Επιλογή της κατανομής]
Κατά μήκος του επιλεγμένου άξονα διάσπασης, επέλεξε την κατανομή με την ελάχιστη τιμή επικάλυψης. Σε περίπτωση ισοπαλίας, επέλεξε την κατανομή με την ελάχιστη τιμή επιφάνειας.

Αλγόριθμος InsertData
Εισαγωγή δεδομένων στη δομή του R tree*

[1 Αρχικοποίηση της διαδικασίας εισαγωγής]
Κάλεσε την Insert ξεκινώντας με το επίπεδο των φύλλων ως παράμετρο, για την εισαγωγή ενός νέου ορθογωνίου δεδομένων.

Αλγόριθμος Insert
Εισαγωγή μιας νέας εγγραφής E στη δομή του R tree*

[1 Επιλογή του υπο-δένδρου που θα γίνει η εισαγωγή]
Κάλεσε την ChooseSubtree, με το επίπεδο ως παράμετρο, για να βρεθεί ο κατάλληλος κόμβος N, στο οποίο θα τοποθετηθεί η νέα εγγραφή E.
[2 Έλεγχος πληρότητας του κόμβου]
Εάν ο κόμβος N έχει λιγότερες από M εγγραφές
Τοποθέτησε την εγγραφή E στον κόμβο N.
Εάν ο κόμβος N έχει M εγγραφές
Κάλεσε την OverflowTreatment με το επίπεδο του κόμβου N ως παράμετρο. [για επανατοποθέτηση ή διάσπαση]
[3 Διάδοση της διάσπασης κόμβων προς τα πάνω]
Εάν η κλήση της OverflowTreatment πραγματοποίησε μια διάσπαση
Επέκτεινε την OverflowTreatment προς τα πάνω εάν είναι απαραίτητο.
Εάν η κλήση της OverflowTreatment προκάλεσε διάσπαση της ρίζας
Δημιούργησε μια νέα ρίζα.
[4 Προσαρμογή των ορθογωνίων κάλυψης της διαδρομής που ακολουθήθηκε]
Προσάρμοσε όλα τα ορθογώνια κάλυψης (covering rectangles) της διαδρομής εισαγωγής, έτσι ώστε να είναι πλαίσια ελάχιστης οριοθέτησης που θα περιλαμβάνουν τα ορθογώνια παιδία τους.

Αλγόριθμος OverflowTreatment
Διαχείριση ενός υπερχειλισμένου κόμβου

[1 Επανεισαγωγή των εγγραφών ή διάσπαση του κόμβου]
Εάν το επίπεδο δεν είναι το επίπεδο ρίζας και αυτή είναι η πρώτη κλήση του αλγορίθμου στο δεδομένο επίπεδο κατά την εισαγωγή ενός ορθογωνίου δεδομένων, τότε
Κάλεσε την ReInsert.
Αλλιώς
Κάλεσε την Split.
Τέλος

Αλγόριθμος ReInsert
Επανεισαγωγή των εγγραφών μετά την περίπτωση υπερχείλισης ενός κόμβου της δομής

[1 Υπολογισμός της απόστασης των ορθογωνίων των εγγραφών από το κέντρο του Περιβάλλοντος Ορθογωνίου του κόμβου]
Για όλες τις M+1 εγγραφές ενός κόμβου N, υπολόγισε την απόσταση μεταξύ των κέντρων των ορθογωνίων τους και του κέντρου του Περιβάλλοντος Ορθογωνίου του κόμβου N.
[2 Ταξινόμηση των εγγραφών]
Ταξινόμησε τις εγγραφές σε φθίνουσα σειρά των αποστάσεων που υπολογίστηκαν στο 1.
[3 Αφαίρεση εγγραφών]
Αφαίρεσε τις πρώτες p εγγραφές του κόμβου N και προσάρμοσε το Περιβάλλον Ορθογώνιό του.
[4 Επαν-εισαγωγή των ταξινομημένων εγγραφών]
Στην ταξινόμηση, που πραγματοποιήθηκε στο 2, ξεκινώντας με τη μέγιστη απόσταση (μακρινή επανατοποθέτηση) ή την ελάχιστη απόσταση (κοντινή επανατοποθέτηση), κάλεσε την Insert για να επανεισάγεις τις εγγραφές.