ΤΕΧΝΟΛΟΓΙΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ 2019-2020

Εργασία RStarTree

Εσκιόγλου Μαρία 3237,
Λυσσούδη Πασχαλίνα 3116

Εισαγωγή

Στην εργασία αναλύσαμε πειραματικά την δομή των R* - tree , μιας εξελιγμένης μορφής των απλών R- tree, που έχουμε συναντήσει μέχρι σήμερα. Πιο συγκεκριμένα, αναλύσαμε την επίδραση που έχει αυτή η δομή σε **ερωτήματα εύρους** (range queries) και σε **ερωτήματα κοντινότερων γειτόνων** (nearest neighbor queries). Μετέπειτα, συγκρίναμε πειραματικά τις δύο αυτές μεθόδους σε κατάσταση απλοϊκής προσέγγισης, δηλαδή σειριακής αναζήτησης, ούτως ώστε να γίνει πιο εύκολα αντιληπτή η αξία της συγκεκριμένης δομής.

Τα χωρικά δεδομένα που χρησιμοποιήθηκαν για την εργασία προήλθαν από το **Open Street Maps**, το μεγαλύτερο ηλεκτρονικό χάρτη ελεύθερης άδειας, ούτως ώστε τα συμπεράσματα μας να έχουν εφαρμογή και στον πραγματικό κόσμο.

Όσον αφορά τη προγραμματιστική προσέγγιση, ο κώδικας λειτουργεί με βάση δύο θεμελιώδεις αρχές : αφενός είναι γενικός, δηλαδή λειτουργεί για **περισσότερες από 2 διαστάσεις** (άσχετα που τα παραδείγματα που θα δοθούν θα είναι για 2) αφετέρου διαχειρίζεται σωστά τη μνήμη, μιας και η **αποθήκευση του δένδρου πραγματοποιείται στη δευτερεύουσα μνήμη** (σκληρός δίσκος) και όχι στη κύρια. Η γλώσσα που χρησιμοποιήθηκε για την ανάπτυξη του project είναι η **Java**.

Ανάλυση

Το project χωρίζεται σε 6 πακέτα, το καθένα προορισμένο ξεχωριστά για το σκοπό του. Υπάρχουν δύο βασικά πακέτα, τα rStarTree και Utilities, με το πρώτο να περιλαμβάνει τα υπόλοιπα 4 πακέτα dataToObject, interfaces, nodes και spatial μαζί με τις κλάσεις ResourceManager, RStarTree και StorageManager. Θα αναλυθούν με τέτοιο τρόπο ούτως ώστε να είναι κατανοητή η ροή του προγράμματος.

Στην ανάλυση των κλάσεων δεν γίνεται εκτενής αναφορά στις μεθόδους κάθε κλάσης, καθώς σχολιάζονται εμβόλιμα μέσα στον κώδικα. Ο σχολιασμός γινεται με χρήση Javadoc πριν το σώμα κάθε μεθόδου, ενώ υπάρχουν ανά περιπτώσεις και σχόλια γραμμής για ευκολότερη κατανόηση.

Interfaces

Τα interfaces είναι απαραίτητα ούτως ώστε να εκμεταλλευτούμε την κληρονομικότητα της γλώσσας στο έπακρο. Περιλαμβάνει τα **DiskQuery** και **SpatialQuery**, το μεν για την αλληλεπίδραση με τον δίσκο, το δε για την αναζήτηση (range και k-nn) στο δένδρο. Έπειτα έχουμε το **DTOConvertible**,για την μεταφορά στον δίσκο και **RStarNodeInterface** , για να καθορίσουμε τις βασικές λειτουργίες ενός κόμβου R*, όπως εισαγωγή ή δημιουργία id.

DataToObject

Όπως γίνεται κατανοητό, η μεταφορά τον δεδομένων από αριθμούς σε αντικείμενα με νόημα είναι θεμέλιος λίθος για τη σωστή διαχείριση της μνήμης, καθώς και για τη σωστή μετατροπή του σε byte stream και την εγγραφή του στο δίσκο. Αυτό επί της ουσίας υλοποιεί η abstract κλάση **AbstractDTO**, που κληρονομεί την Serializable. Μετέπειτα οι άλλες κλάσεις(**MBR,Node, Point, Tree**) επεκτείνουν αυτή τη γενική και η κάθε μία ασχολείται αποκλειστικά με αυτό που το όνομα της περιγράφει, δηλαδή την αποτύπωση των minimum bounding rectangle, των κόμβων, των σημείων και του δένδρου αντίστοιχα.

Spatial

Σε αυτό το πακέτο πραγματοποιούνται οι διεργασίες που αφορούν το χώρο, εξού και το όνομα. Όπως είναι γνωστό ήδη από τη θεωρία ο χώρος στα R-trees μπορεί να έχει δύο προσδιοριστικές έννοιες, την έννοια του χώρου σαν ορθογώνιο (rectangle) και την έννοια του χώρου σαν σημείο (point). Η κλάση **Rectangle** περιέχει τις διαστάσεις του ορθογωνίου, καθώς και τις απαραίτητες λειτουργίες του, όπως τομή ορθογωνίων, υπολογισμός εμβαδόν και φυσικά ανανέωση του ορθογωνίου. Ανάλογα, αλλά πιο απλά, η κλάση **Point** περιέχει τα προσδιοριστικά στοιχεία του, όπως id, συντεταγμένη και διαστάσεις καθώς και μία συνάρτηση απόστασης μεταξύ σημείων. Τέλος , υπάρχει η κλάση του χωρικού συγκριτή, δηλαδή ένας **spatial comparator** που αφού ξεκαθαριστεί το αντικείμενο σύγκρισης (ορθογώνια ή σημεία που συγκρίνουμε) τελεί τις ανάλογες πράξεις . Για παράδειγμα, σε ένα ενδεχόμενο split γυρνάει 0 ή 1 για το αν πρέπει ή όχι να πραγματοποιηθεί για μία δεδομένη λίστα καταχωρήσεων. Σε αυτό το σημείο είναι άξιο αναφοράς να πούμε ότι οι

comparators απλοποιούν πολύ τα προβλήματα στη java καθώς μας αποτρέπει από το να κάνουμε συγκρίσεις σε κάθε σημείο του κώδικα.

Nodes

Αυτό το πακέτο, όπως είναι αντιληπτό από το όνομα, αφορά τα nodes , δηλαδή τους κόμβους του δένδρου. Σε αυτό το σημείο είναι ανάγκη να τονιστεί ότι γίνεται διαχωρισμός των εξωτερικών με εσωτερικών κόμβων στο δένδρο, καθώς υπάρχει ποιοτική διαφορά στη προσέγγιση και συμπεριφορά τους. Έτσι, έχουμε τη γενική κλάση **RStarNode** , που κληρονομεί τα κατάλληλα interfaces , ούτως ώστε να πάρει τα χαρακτηριστικά που χρειάζεται για την ταυτοποίηση του σαν κόμβο και παράλληλα να γίνει πιο εύκολη η διαχείριση στο δίσκο. Με τη σειρά τους, οι **Leaf** και **Internal** κληρονομούν τις ιδιότητες της **RStarNode**. Η ποιοτικότερη διαφορά των δύο αυτών κλάσεων είναι ότι η leaf δείχνει σε points, δηλαδή πραγματικά σημεία, ενώ η internal δείχνει σε nodes, δηλαδή κόμβους. Τέλος, στο κομμάτι του split, τα πράγματα έγιναν αρκετά περίπλοκα, καθώς το split του R- tree διαφέρει από του R*. Πιο συγκεκριμένα, ο αλγόριθμος διαχωρισμού σε ένα R-tree αρχικοποιεί δύο ομάδες , με τις μακρύτερες σε απόσταση καταχωρήσεις, κι έπειτα βάζει μέσα στις δύο αυτές ομάδες κατάλληλα και τις υπόλοιπες καταχωρήσεις. Στο R*-tree, αντίθετα, επιλέγεται ένας άξονας, είτε οριζόντιος είτε κάθετος, και εξετάζονται όλες οι πιθανές διαμοιράσεις από τη μία πλευρά της διαχωριστικής γραμμής.[1]

RStarTree

(ξεχωριστή κλάση, ανήκει στο γενικό πακέτο rStarTree)

Πλέον , έχουμε φτάσει σε σημείο που μπορούμε να χρησιμοποιήσουμε όλες τις προηγούμενες κλάσεις για να πετύχουμε τον αρχικό μας στόχο, δηλαδή να δομήσουμε ένα R* δένδρο. Στη συγκεκριμένη κλάση συνδυάζουμε όλες τις απαραίτητες συναρτήσεις για το δένδρο μας. Πιο συγκεκριμένα, υπάρχει η δυνατότητα φόρτωσης του δένδρου από τη δευτερεύουσα μνήμη στη κύρια, για ανάκληση των δεδομένων καθώς και το αντίθετο για αποθήκευση τους. Επιπλέον, υπάρχουν οι συναρτήσεις εισαγωγής (**insert** και **insertAt**/ γενικά ή συγκεκριμένα), οι συναρτήσεις επιλογής και προσαρμογής (**chooseLeaf** και **adjustParentOf**) , καθώς και οι συναρτήσεις του διαχωρισμού (**splitInternal**, **splitLeaf**). Τέλος, υπάρχει και η δυνατότητα για query search , είτε πρόκειται για **knn** είτε για **range**. Τόσο η knn όσο και οι range έχουν τις βασικές/θεμελιώδεις συναρτήσεις(**knnSearch** και **rangeSearch**) και έπειτα τις αναδρομικές (**doknnSearch** και **doRangeSearch**) . Η ποιοτική διαφορά τους είναι ότι υπολογίζονται στις πρώτες οι κατάλληλες παράμετροι για να καλεστούν οι δεύτερες.

ResourceManager, StorageManager

(ξεχωριστές κλάσεις, ανήκουν στο γενικό πακέτο rStarTree)

Μετέπειτα, υπάρχουν οι **managers**, **resource** και **storage** αντίστοιχα, με εξεζητημένο ρόλο αμφότεροι. Ο απλός , ο **storage**, διαχειρίζεται **τα I/O μεταξύ κύριας μνήμης** και δίσκου και αφορά ξεκάθαρα μόνο το δένδρο, καθώς αυτό κρύβει μεγάλο όγκο πληροφοριών. Από την άλλη, ο **resource** διαχειρίζεται τις πηγές μας, άρα στη συγκεκριμένη **τα αρχεία του open street maps**, καθώς τα φιλτράρει με την αντίστοιχη συνάρτηση **filterOSM**.

Utilities

Το συγκεκριμένο πακέτο είναι γενικό και περιέχει τις κλάσεις **Utilities**, **Constants.** Δεν έχουν ουσιαστικό χαρακτήρα οι συγκεκριμένες κλάσεις αλλά είναι απαραίτητες για να μην υπάρχουν hard-coded σημεία στο project. Η **constant** περιέχει τις σταθερές, όπως ονόματα αρχείων και τιμών, ενώ η κλάση **utilities** ονοματοδοτεί με τις κατάλληλες id τα points.

MainClass

Τέλος, υπάρχει η κλάση MainClass, που περιέχει την **main** συνάρτηση, η οποία «τρέχει» και ρυθμίζει το πρόγραμμα εκεί μέσα. Εκεί μέσα υπάρχουν οι **διάλογοι**, τα **processes** για τα **inputs** και η δημιουργία **index file** .

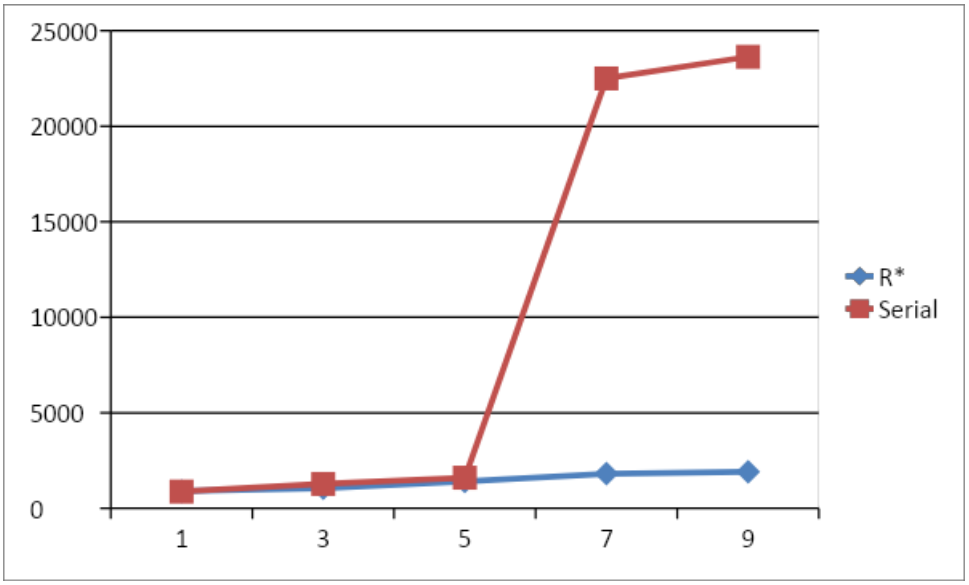
Πειράματα

Για να έχουμε μία καλύτερη εικόνα της αποδοτικότητας ενός R*- tree θα μπορούσαμε να το συγκρίνουμε με τον απλοϊκότερο αλγόριθμο, αυτόν της σειριακής αναζήτησης, ούτως ώστε να βγάλουμε κάποια συμπεράσματα. Σε αυτό το σημείο να τονιστεί ότι ο χρόνος που αναγράφεται στα παρακάτω πειράματα αφορά μόνο τις αναζητήσεις και όχι τη σύσταση του δένδρου. Γίνεται κατανοητό ότι η δημιουργία συνεχώς διαφορετικών δένδρων είναι κοστοβόρα διαδικασία, όμως αν βγάλουμε τον χρόνο προεπεξεργασίας (δεδομένου ότι το κάνουμε μία φορά) τότε αναμφίβολα παρατηρούμε ότι είναι κατά πολύ αποδοτικότερος.

Για το παράδειγμα μας θα χρησιμοποιήσουμε τον χάρτη του ΑΠΘ. Έτσι, ακόμη και σε έναν μικρό (για τα πραγματικά δεδομένα) χάρτη θα δείξουμε ότι ο R* υπερισχύει κατά πολύ της γραμμικής αναζήτησης. Η αναζήτηση μας

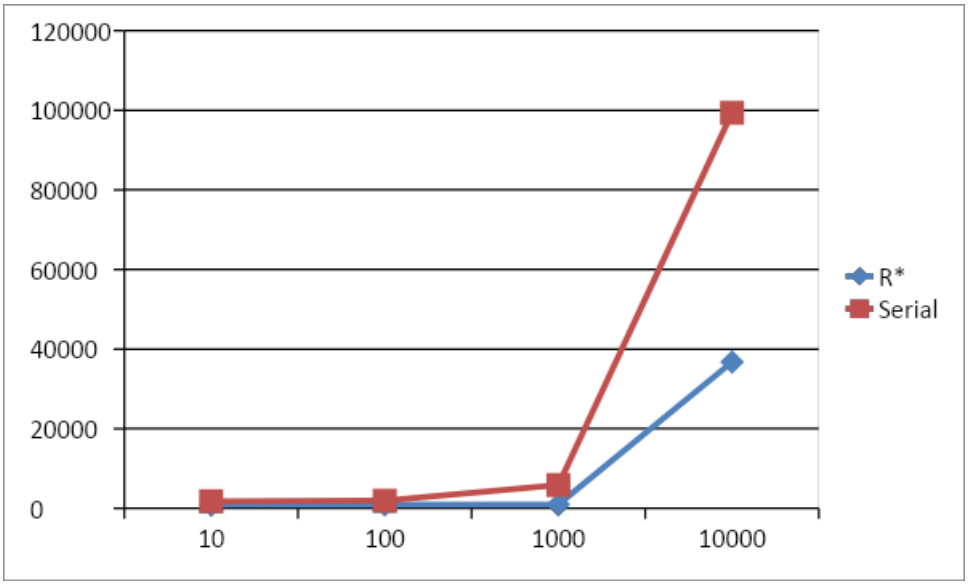
θα αφορά το σημείο με συντεταγμένες : [40.63342 22.95649], όπου εκεί βρίσκεται η Σχολή Θετικών Επιστημών.

Range



Στο συγκεκριμένο παρατηρούμε ότι ο R* δεν αλλάζει δραματικά συμπεριφορά όσο αυξάνεται το εύρος, σε αντίθεση με την σειριακή αναζήτηση όπου αυξάνεται δραματικά μετά από R=5 με αποτέλεσμα να καθίσταται χειρότερος αλγόριθμος.

K – NN



Για τους κοντινότερους γείτονες βλέπουμε ότι ενώ στην αρχή ο γραμμικός τρόπος αναζήτησης συμβαδίζει με το R* , έστω και με λίγο χειρότερες

αποδόσεις, ξαφνικά όταν αυξάνουμε τους γείτονες στους 10.000 εκτινάσσεται κατά πολύ ο χρόνος του, που φτάνει 62.621 ms ενάντια στα 36.756 του R*.

Τελικά Συμπεράσματα

Σαν τελικό συμπέρασμα, με βάση πάντα τα πειράματα μας, θα μπορούσαμε να πούμε ότι το δένδρο R* αποτελεί μία βάσιμη και αξιόπιστη λύση για αναζητήσεις τέτοιου τύπου και ειδικότερα όταν αφορά μεγάλους αριθμητικούς παράγοντες, πόσο μάλλον όταν η εναλλακτική μας λύση αποτελεί η σειριακή αναζήτηση, από την οποία υπερέχει κατά κράτος.

Βιβλιογραφία

[1] Philippe Rigaux, Michel Scholl - Spatial Databases: With Application to GIS