ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ SQL – NOSQL ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ: ΣΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΓΕΩΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Διπλωματική Εργασία

Του

Βασιλείου Δριμζάκα Παπαδόπουλου

Θεσσαλονίκη, Οκτώβριος 2023

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ SQL – NOSQL ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ: ΣΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΓΕΩΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Βασίλειος Δριμζάκας Παπαδόπουλος

Πτυχίο Μηχανικών Γεωπληροφορικής & Τοπογραφίας, ΔΙ.ΠΑ.Ε, 2019

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Δρ. Γεώργιος Ευαγγελίδης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Γεώργιος Ευαγγελίδης	Κολωνιάρη Γεωργία	Καρακασίδης Αλέξανδρος
Καθηγητής	Επίκουρη Καθηγήτρια	Μέλος Ε.ΔΙ.Π.
.....	.....	.....

Βασίλειος Δριμζάκας Παπαδόπουλος

.....

Περίληψη

Λόγω της μεγάλης διαθεσιμότητας γεω-χωρικών δεδομένων, των διαφόρων τύπων με τους οποίους αυτά διατίθενται και της ανάγκης για ανάκτηση και ανάλυσή τους σε πραγματικό χρόνο, καθίσταται επιτακτική ανάγκη η αναζήτηση νέων δομών αποθήκευσής τους. Μέχρι πρόσφατα τα συστήματα διαχείρισης γεω-χωρικών δεδομένων, χρησιμοποιούσαν αποκλειστικά SQL βάσεις δεδομένων. Με την αύξηση του όγκου και των διαφορετικών τύπων (διανυσματικά δεδομένα, ψηφιδωτά δεδομένα (raster) και δεδομένα χρονοσειρών (time-series), όπως δεδομένα από δορυφόρους ή από αερομεταφερόμενα μέσα) των διαθέσιμων δεδομένων, οι παραδοσιακές βάσεις δεδομένων παρουσιάζουν αδυναμίες (δυσκολία αποθήκευσης, ευρετηριοποίησης, όγκος δεδομένων χρονοσειρών (scalability)) και καθυστερήσεις στην διαχείρισή τους. Τα τελευταία χρόνια παρατηρείται αύξηση στη χρήση NoSQL βάσεων δεδομένων ώστε να επιτευχθεί η επίλυση των προβλημάτων που προκύπτουν. Σκοπός της παρούσας εργασίας, ήταν η συγκριτική μελέτη των δύο συστημάτων διαχείρισης βάσεων δεδομένων, ώστε να εξαχθούν συμπεράσματα σχετικά με την αποτελεσματικότητά τους στη διαχείριση διαφόρων τύπων χωρικών δεδομένων, τις δυνατότητες που προσφέρουν, τις αδυναμίες που παρουσιάζουν, και την αποτελεσματικότητά τους, όσων αφορά την ταχύτητα αποθήκευσης, ανάκτησης και επεξεργασίας των γεωχωρικών δεδομένων. Στην παρούσα εργασία πραγματοποιήθηκε βιβλιογραφική ανασκόπηση για την αξιολόγηση SQL – NoSQL βάσεων δεδομένων στην αποθήκευση και επεξεργασία των γεω-χωρικών δεδομένων, καθώς και συγκριτικά τεστ (queries), στις PostGIS και Neo4j βάσεις δεδομένων, για τη συγκριτική τους αξιολόγηση με τη χρήση σημειακών και πολυγωνικών διανυσματικών δεδομένων, καθώς και χωρικών δεδομένων εικόνας. Από τα αποτέλεσμα προέκυψε πως η Neo4j, είναι ταχύτερη ως προς την ανάλυση των δεδομένων, ενώ η PostGIS, παρέχει πληθώρα χωρικών λειτουργιών, που τα συστήματα NoSQL, δεν διαθέτουν.

Λέξεις κλειδιά – Keywords

PostGIS, Neo4j, SQL, NoSQL, Spatial Databases

Abstract

Due to the large availability of geo-spatial data, the different types of available data and the need for real-time retrieval and analysis, it becomes imperative to search for new storage structures. Until recently, geo-spatial data management systems have exclusively used SQL databases. With the increase in the volume and the different types (vector data, raster data and time-series data, such as data from satellites or airborne media) of available data, traditional databases have been suffering from weaknesses (storage difficulties, indexing, scalability, and time-series data volume) and delays in their management. In recent years there has been an increase in the use of NoSQL databases to achieve the solution of the problems that arise. The purpose of this paper, was the comparative study of the two database management systems in order to draw conclusions about their effectiveness in managing different types of spatial data, the capabilities they offer, the weaknesses they present, and their effectiveness in terms of speed of storage, retrieval and processing of geospatial data. In this paper, a literature review was conducted to evaluate SQL - NoSQL databases in the storage and processing of geo-spatial data, as well as comparative (queries), on PostGIS and Neo4j databases, for their comparative evaluation using point and polygonal vector data, as well as spatial image data. The results showed that Neo4j, is faster in terms of data analysis, while PostGIS, provides a plethora of spatial functions, which NoSQL systems, do not have.

Πρόλογος – Ευχαριστίες

Στα πλαίσια της παρούσας μεταπτυχιακής διατριβής, θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην ολοκλήρωσή της και συγκεκριμένα θα ήθελα να ευχαριστήσω πρώτα τον επιβλέποντα Δρ. Γεώργιο Ευαγγελίδη για τις πολύτιμες συμβουλές και τη βοήθειά του καθώς και τα μέλη της επιτροπής αξιολόγησης Δρ. Γεωργία Κολωνιάρη, Επίκουρη Καθηγήτρια και τον Δρ. Αλέξανδρο Καρακασίδη, μέλος Ε.ΔΙ.Π, για τη βοήθειά τους και την αξιολόγηση της παρούσας διατριβής.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω τον Δρ. Κωνσταντίνο Ντούρο, Επίκουρο Καθηγητή στο Διεθνές Πανεπιστήμιο Ελλάδος (ΔΙ.ΠΑ.Ε |IHU), αγαπητό μου φίλο και συνεργάτη, για τις συμβουλές του, καθώς και για την αμέριστη υποστήριξη, κατανόηση και υπομονή που έδειξε κατά τη διάρκεια της περάτωσης των σπουδών μου στο Π.Μ.Σ. στην Εφαρμοσμένη Πληροφορική.

Ακόμη θα ήθελα να ευχαριστήσω τον καλό μου φίλο και συνεργάτη κ. Γεώργιο – Δημήτριο Γκολογκίνα, για την πολύτιμη βοήθειά του στην περάτωση της παρούσας διατριβής.

Τέλος, θέλω να ευχαριστήσω από τα βάθη της καρδιάς μου, την οικογένειά μου για την στήριξη και την βοήθειά τους, καθώς είναι πάντα δίπλα μου, όχι μόνο κατά τη διάρκεια των σπουδών μου στο Π.Μ.Σ στην Εφαρμοσμένη Πληροφορική, αλλά καθημερινά απέναντι σε όλες τις προκλήσεις της καθημερινότητας.

Πίνακας περιεχομένων

Κατάλογος εικόνων

Κατάλογος διαγραμμάτων

Κατάλογος πινάκων

1. Εισαγωγή

Τα τελευταία χρόνια με την ανάπτυξη τεχνολογιών και εφαρμογών δημιουργίας και καταγραφής γεω – χωρικών δεδομένων [1], όπως το διαδίκτυο των πραγμάτων (Internet of Things | IoT), τα μέσα κοινωνικής δικτύωσης (geotagged data) [2] και η ανάπτυξη νέων αισθητήρων καταγραφής (π.χ. δορυφόροι), δημιουργήθηκε η ανάγκη για νέα συστήματα αποθήκευσης, επεξεργασίας και ανάλυσης των δεδομένων σε πραγματικό ή σχεδόν πραγματικό χρόνο (Near Real Time). Μέχρι προσφάτως, τα συστήματα διαχείρισης γεω – χωρικών δεδομένων περιορίζονταν στη χρήση σχεσιακών βάσεων δεδομένων (Relational Data Base Management System |RDBMS) [3], με ποιο δημοφιλές την PostGIS, που αποτελεί επέκτασης της PostgreSQL για τη διαχείριση και αποθήκευση γεω – χωρικών δεδομένων [4]. Εξαιτίας του γεγονότος πως τα γεω – χωρικά δεδομένα αποτελούν εκ φύσεως «μεγάλα δεδομένα» (Big Data) [5], της συνεχούς αύξησης του όγκου των δεδομένων αυτών και της ανάγκης για ανάλυσή τους σε πραγματικό χρόνο (π.χ. πρόληψη και διαχείριση φυσικών καταστροφών), οι σχεσιακές βάσεις δεδομένων, αντιμετωπίζουν προβλήματα, καθώς σε πολλές περιπτώσεις τα δεδομένα προς ανάλυση είναι διαμοιρασμένα σε διαφορετικούς κόμβους (servers) και απαιτείται η παράλληλη πρόσβαση και επεξεργασία τους [6]. Εξαιτίας των διαφόρων τύπων αισθητήρων (real time GPS tracker, κινητά τηλέφωνα, κάμερες, αισθητήρες IoT κ.ά.) που χρησιμοποιούνται για την καταγραφή και τη δημιουργία τους, τα δεδομένα αυτά είναι σε μεγάλο βαθμό αδόμητα (unstructured data) ή ημι – δομημένα (semi – structured), γεγονός που δυσκολεύει τη χρήση βάσεων SQL (Structured Query Language) στα συστήματα διαχείρισής και αποθήκευσής τους, καθώς οι SQL βάσεις δεδομένων είναι σχεδιασμένες για δεδομένα με συγκεκριμένη δομή (structured data), υιοθετούν συγκεκριμένο σχήμα (schema) και η επεκτασιμότητά τους (scalability) είναι περιορισμένη [7]. Εξαιτίας των περιορισμών που άπτονται της χρήσης σχεσιακών βάσεων δεδομένων, η επιστημονική (και επιχειρησιακή) κοινότητα στρέφεται στις NoSQL (Not Only SQL) βάσεις δεδομένων, καθώς πλέον τα δεδομένα προέρχονται από πληθώρα διαφορετικών αισθητήρων και συσκευών, έχοντας διαφορετική δομή μεταξύ τους. Ως αποτέλεσμα, καθίσταται ιδιαίτερα δύσκολος ο σχεδιασμός μίας βάσης δεδομένων με προκαθορισμένο σχήμα. Αντίθετα οι NoSQL βάσεις δεδομένων προσφέρουν τη δυνατότητα ευέλικτου σχεδιασμού της βάσης, χωρίς την ανάγκη ορισμού σχήματος εξ’ αρχής [8]. Επιπλέον, οι NoSQL βάσεις δεδομένων, είναι εξ’ ορισμού κατασκευασμένες για χρήση σε περιβάλλον

συστοιχίας υπολογιστών (Clusters) και διαχείριση εξαιρετικά μεγάλου όγκου δεδομένων, αποτελούν «καλύτερη» επιλογή για εφαρμογές που απαιτούν ταυτόχρονη πρόσβαση στη βάση και ανάλυση δεδομένων από πολλούς χρήστες [9]. Για να επιτευχθεί η δυνατότητα επέκτασης (horizontal scalability) και χρήσης των NoSQL βάσεων σε συστοιχίες υπολογιστών, υπήρξε συμβιβασμός στον σχεδιασμό τους, με αποτέλεσμα να απωλέσουν τις αυστηρά καθορισμένες ιδιότητες ACID (Atomicity – Consistency – Isolation – Durability), ιδιότητες απαραίτητες για ορισμένες εφαρμογές [10] (π.χ. τραπεζικές συναλλαγές).

Δεδομένης της αυξανόμενης χρήσης των NoSQL βάσεων δεδομένων στο χώρο της γεω-πληροφορικής, σκοπός της εργασίας ήταν η διερεύνηση και η συγκριτική αξιολόγηση SQL – NoSQL βάσεων δεδομένων στην αποθήκευση και επεξεργασία των γεω-χωρικών δεδομένων.

Οι επιμέρους στόχοι ήταν οι εξής:

i. Ανασκόπηση και συγκριτική αξιολόγηση SQL – NoSQL βάσεων δεδομένων
ii. Η διερεύνηση των δυνατοτήτων τους, αναφορικά με τα γεω-χωρικά δεδομένα και η αξιολόγηση τους μέσα από την εφαρμογή γεω – χωρικών αναλύσεων

2. Βιβλιογραφική ανασκόπηση

Από την ανασκόπηση της βιβλιογραφίας γίνεται φανερό, ότι υπάρχει αυξανόμενο ενδιαφέρον για την ανάπτυξη και τη βελτίωση των μηχανισμών και των εργαλείων διαχείρισης γεωχωρικών δεδομένων, για πλήθος εφαρμογών, όπως η περίπτωση της δρομολόγησης οχημάτων (routing), όπου οι Bierbauer et al., (2016) [11], βασίστηκαν πάνω στο “έργο” Isochrone (Isochrone project) [12], που χρησιμοποιεί την PostGIS και διερεύνησαν την αποδοτικότητα της χρήσης των NoSQL βάσεων δεδομένων σε σχέση με τις SQL, συγκρίνοντας την PostGIS με τη Neo4j και τη MongoDB, για τον υπολογισμό Ισόχρονων (Isochrones), των μέσων μαζικής μεταφοράς. Στην εργασία τους, χρησιμοποιούν δύο σύνολα δεδομένων, για το Bolzano, Ιταλία (6000 σημεία & 15550 ακμές) και για το San Franciso, ΗΠΑ (33000 σημεία & 93000 ακμές). Κατέληξαν πως η βάση δεδομένων MongoDB, αποδίδει καλύτερα από τις άλλες δύο υπό διερεύνηση βάσεις δεδομένων (Neo4j & PostGIS), καθώς ο χρόνος που χρειάζεται για την επεξεργασία των ερωτημάτων είναι πολύ μικρότερος από τον χρόνο που χρειάζονται η PostGIS και η Neo4j. Ένα άλλο συμπέρασμα στο οποίο κατέληξαν,

είναι πως η PostGIS, είναι πιο αποδοτική συγκριτικά με την Neo4j και στα δύο σύνολα δεδομένων, αλλά επισημαίνουν πως όταν αυξάνεται ο όγκος των δεδομένων, ο χρόνος επεξεργασίας στη Neo4j βελτιώνεται. Ένα άλλο πεδίο στο οποίο οι ερευνητές επικεντρώνονται στα συστήματα διαχείριση βάσεων δεδομένων είναι η τηλεπισκόπηση [13], όπου οι Karmas et al., (2014), δημιούργησαν ένα διαδικτυακό σύστημα ανάλυσης δεδομένων για γεωργικές εφαρμογές (RemoteAgri system). Ως βασικό στοιχείο του συστήματος χρησιμοποίησαν το σύστημα διαχείρισης βάσεων δεδομένων RASDAMAN |Raster Data Manager (Array DBMS) [14], το οποίο είναι ειδικά σχεδιασμένο για τη διαχείριση και ανάλυση μεγάλων δεδομένων με πολλές διαστάσεις (π. χ. πολυφασματικά δεδομένα), καθώς οι παραδοσιακές βάσεις δεδομένων δεν υποστηρίζουν πολυδιάστατους πίνακες δεδομένων μεγάλου μεγέθους. Επιπλέον, έκαναν χρήση του προτύπου WCPS (Web Coverage Processing Service) [15], σχεδιασμένο ειδικά για την επεξεργασία, ανάλυση και εξαγωγή πολύ – επίπεδων (multidimensional) δεδομένων από το OGC (Open Geospatial Consortium) ως γλώσσα ερωτημάτων προς τη βάση. Ένα άλλο παράδειγμα αναφορικά με τη βελτίωση των συστημάτων διαχείρισης βάσεων δεδομένων σε εφαρμογές τηλεπισκόπησης, είναι η εργασία των Jing et al., (2018) [16], οι οποίοι χρησιμοποίησαν το σύστημα HBase (Column DBMS), για την αποθήκευση δεδομένων εικόνων, προερχόμενα από μεθόδους τηλεπισκόπησης. Εκμεταλλεύτηκαν το κατανεμημένο σύστημα (Hadoop Distributed File System) [17] αποθήκευσης που χρησιμοποιεί η HBase, το οποίο χωρίζει την εικόνα σε πυραμίδες (Pyramid image data storage) [18] και συνδυάζοντας τη καμπύλη Hilbert (Hilbert curve) [19], με τον μοναδικό αριθμό πλέγματος (grid index) που προκύπτει από την διαδικασία «πυραμιδοποίησης» της εικόνας βελτίωσαν τη μέθοδο ευρετηριοποίησης δεδομένων (Indexing). Παράλληλα, εγγυώνται τη χωρική γειτνίαση (spatial proximity) των δεδομένων της εικόνας κατά την αποθήκευση, δηλαδή δεδομένα τα οποία είναι γειτνιάζοντα στον γεωγραφικό χώρο, θα αποθηκευτούν μαζί στη βάση (groups). Τέλος, συνέκριναν τη μέθοδό τους με τα συστήματα διαχείρισης MapFile και MySQL Cluster, με το μοντέλο που σχεδίασαν πάνω στο σύστημα MapReduce HBase και κατέληξαν πως η μέθοδός τους είναι αποδοτικότερη στην επεξεργασία ερωτημάτων και προσφέρει καλύτερη επεκτασιμότητα. Στο πεδίο της οικολογίας οι Hein et al., (2021) [20], στο πλαίσιο του προγράμματος “RiverView®” [21], ανέπτυξαν έναν μηχανισμό αποθήκευσης δεδομένων εικόνων, στη Neo4j, με στόχο την εξαγωγή δεδομένων σε (σχεδόν) πραγματικό χρόνο, την επεκτασιμότητα του συστήματος, ώστε να είναι διαθέσιμο και για επεξεργασία μεγάλου

όγκου δεδομένων και τέλος τη διαχείριση διαφορετικών τύπων δεδομένων (heterogeneous data). Εξετάζοντας τις διαθέσιμες επιλογές στη βάση δεδομένων για την αποθήκευση εικόνων, κατέληξαν στο συμπέρασμα πως είναι αποδοτικότερο να αποθηκεύσουν στη βάση τα μετα – δεδομένα (metadata) των εικόνων και τις εικόνες σε μέσα αποθήκευσης (file systems). Για την εφαρμογή της μεθοδολογίας τους, πρώτα δημιουργήθηκαν πυραμίδες για κάθε εικόνα, μετά εφάρμοσαν μία μέθοδο ευρετηριοποίησης (R – Tree) σε κάθε επίπεδο των πυραμίδων και τέλος, αποθήκευσαν τα μετα – δεδομένα των εικόνων, έναν σύνδεσμο (link) για κάθε επιμέρους κομμάτι των εικόνων (tile), καθώς και τον αντίστοιχο δείκτη (index) στη βάση δεδομένων. Συμπέραναν πως, η μέθοδος που εφάρμοσαν είναι αποδοτική όταν απαιτείται πρόσβαση για ανάγνωση των δεδομένων (read access), αλλά επιβραδύνει το σύστημα όταν απαιτείται να γίνουν εγγραφές (write access). Οι Ramiaramanana et al., (2022) [22], χρησιμοποίησαν την Neo4j, και συγκεκριμένα την επέκταση Neo4j Spatial, που επιτρέπει τη διαχείριση γεωχωρικών δεδομένων, με σκοπό την ανάπτυξη συστήματος, βασιζόμενο σε γνωσιακή μέθοδο (knowledge – based method) για την αναγνώριση γεωμορφολογικών σχηματισμών, με βάση τα δομικά τους χαρακτηριστικά και τα χαρακτηριστικά του χώρου που περιβάλλει τους σχηματισμούς. Επέλεξαν τη Neo4j, για το ευέλικτο μοντέλο αποθήκευσης δεδομένων, καθώς και τη διευκόλυνση που προσφέρει η μορφή γραφήματος για την αναπαράσταση των δομικών στοιχείων των σχηματισμών, των γεωμορφολογικών τους χαρακτηριστικών και των τύπων δεδομένων διαθέσιμων για τον κάθε σχηματισμό, τα οποία σχετίζονται μεταξύ τους με πλήθος διαφορετικών τρόπων, γεγονός το οποίο δυσκολεύει τη χρήση σχεσιακών βάσεων δεδομένων. Κατέληξαν πως η χρήση της συγκεκριμένης βάσης δεδομένων ενδείκνυται για την ανάπτυξη του συγκεκριμένου ή παρόμοιων συστημάτων βασισμένα σε γνωσιακές μεθόδους, καθώς ο χρήστης μπορεί εύκολα και γρήγορα να αναγνωρίσει τον επιθυμητό γεωμορφολογικό σχηματισμό, μέσα από τα δομικά του στοιχεία, όπως ορίζεται στο σύστημα, εξοικονομώντας πολύτιμο χρόνο. Εν συντομία παρατίθενται κι άλλοι τομείς όπου ερευνητές επικεντρώθηκαν στη χρήση συστημάτων βάσεων δεδομένων, και στη βελτίωση αυτών. Οι Wu et al., (2017) [23], συνδύασαν τη χρήση SQL (MySQL) και NoSQL (Redis – MongoDB) βάσεων δεδομένων, για την κατασκευή ενός συστήματος που θα καταγράφει, αναλύει σε πραγματικό χρόνο και θα αποθηκεύει γεω – βίντεο [24]. Στόχος είναι ο εντοπισμός «ανωμαλιών» στο βίντεο και η αποστολή ειδοποιήσεων στους ιθύνοντες, για λήψη κατάλληλων μέτρων, με σκοπό την προστασία των πολιτών (public security). Οι Jung et al., (2015) [25], σχεδίασαν ένα σύστημα

(Geo – Targeted Event Observation Viewer | GeoViewer) το οποίο χρησιμοποιεί την βάση δεδομένων MongoDB, για την αποθήκευση και ανάλυση δεδομένων με γεωγραφικές ετικέτες (geo – tagged) που συλλέγονται από τα μέσα κοινωνικής δικτύωσης, με σκοπό να βοηθήσει τους πολίτες αλλά και τις αρμόδιες αρχές, να αποκτήσουν καλύτερη επίγνωση της κατάστασης και να λάβουν βέλτιστες αποφάσεις για την αντιμετώπιση και διαχείριση εκτάκτων αναγκών (π. χ. πυρκαγιές) .

Οι προσπάθειες δεν επικεντρώνονται μόνο στην ανάπτυξη μηχανισμών που στοχεύουν στην αποθήκευση και επεξεργασία δεδομένων, αλλά σε ένα ευρύτερο σύνολο μηχανισμών και εργαλείων διαχείρισης των βάσεων δεδομένων, όπως η εύρεση τρόπων βελτίωσης των ερωτημάτων. Ο Ilba, (2021) [26], ανέπτυξε έναν αλγόριθμο για παράλληλη επεξεργασία γεω – χωρικών ερωτημάτων σε σχεσιακές βάσεις δεδομένων. Ο αλγόριθμός του, χωρίζει τα δεδομένα βάσει της χωρικής ευρετηριοποίησης (spatial index) και τη χρήση των ρητρών (clauses) «LIMIT» και «OFFSET». Στη συνέχεια υπολογίζει και αναθέτει τα δεδομένα προς επεξεργασία στα αντίστοιχα διαθέσιμα νήματα (threads). Με τη χρήση του αλγορίθμου η επεξεργασία επιταχύνθηκε μέχρι και πάνω από τρεις φορές (x3.6) στην SQLite \ SpatiaLite και μέχρι πάνω από πέντε φορές (x5.1) στην PostgreSQL \ PostGIS. Ο συγγραφέας σημειώνει πως οι μετρήσιμοι χρόνοι αφορούν την επεξεργασία και όχι την εισαγωγή των αποτελεσμάτων στη βάση, καθώς η SQLite, δεν υποστηρίζει παράλληλες εγγραφές. Επισημαίνει επίσης, πως η παράλληλη επεξεργασία δεδομένων όταν η βάση έχει λίγες εγγραφές, στην SQLite, δεν είναι αποδοτική, καθώς ο χρόνος που χρειάζεται για τον διαχωρισμό των δεδομένων είναι μεγάλος, με αποτέλεσμα να μειωθεί η απόδοση. Εάν ο όγκος τον δεδομένων στη βάση είναι μεγάλος, η χρήση του αλγορίθμου συνίσταται, καθώς ο χρόνος για τον διαχωρισμό των δεδομένων είναι αμελητέος, στο σύνολο του χρόνου της επεξεργασίας. Στην Postgres \ PostGIS, ο χρόνος ανάλυσης των δεδομένων μειώθηκε ανεξαρτήτως του μεγέθους της βάσης, όσο περισσότερα τα δεδομένα προς ανάλυση, τόσο επιταχύνεται η διαδικασία.

Ένα άλλο αντικείμενο των συστημάτων διαχείρισης βάσεων δεδομένων που απασχολεί τους ερευνητές είναι η διευκόλυνση του χρήστη να θέτει ερωτήματα. Αναφέρεται ενδεικτικά η εργασία των Wang et al., (2019) [27], οι οποίοι σχεδίασαν ένα πλαίσιο εργασίας (framework), μέσω του οποίου επιδιώκουν τη δημιουργία ενός μηχανισμού, που θα επιτρέπει στον χρήστη να θέσει ερωτήματα σε φυσική γλώσσα, η οποία κωδικοποιείται από

τον μηχανισμό σε γλώσσα SQL, θα συνδέεται με τη βάση και θα επιστρέφει τα επιθυμητά αποτελέσματα χρήστη.

Επιπλέον, τα γεω – χωρικά δεδομένα πολλές φορές εμπεριέχουν έναν βαθμό ασάφειας. Γεγονός που οφείλεται σε διάφορες παραμέτρους, όπως για παράδειγμα, η μετατροπή διανυσματικών δεδομένων (vector), σε δεδομένα εικονοστοιχείων (raster), η αβεβαιότητα που εισέρχεται στην ακρίβεια των μετρήσεων πεδίου, στην ακρίβεια της κατηγοριοποίησης των στοιχείων μίας εικόνας (image classification) ή σε μεθόδους παρεμβολής που εφαρμόζονται στα δεδομένα. Επιπλέον, σε κάποιες περιπτώσεις δεν υπάρχουν αυστηρώς ορισμένα κριτήρια για την ανάλυσή τους. Γεγονός που αμβλύνεται συχνά σε περιπτώσεις εφαρμογής όπου χρησιμοποιείται πολυκριτηριακή ανάλυση (multicriteria analysis), καθώς ο χρήστης αναγκάζεται να προχωρήσει σε παραδοχές για ορισμένα κριτήρια, στη βάση της δυαδικής λογικής. Σε αυτή την περίπτωση απαιτείται η εξεύρεση νέων μεθόδων για την ανάλυση των δεδομένων, καθώς η δυαδική λογική της άλγεβρας Boolean (Boolean Algebra) [28], που χρησιμοποιούν οι σχεσιακές βάσεις δεδομένων, δεν ικανοποιούν τις ανάγκες των εφαρμογών. Για τον λόγο αυτό χρησιμοποιείται η «Ασαφής Λογική» (Fuzzy Logic) [29]. Η Ďuračiová (2013) [30], εφάρμοσε τις αρχές της «Ασαφούς Λογικής», σε γεω – χωρική βάση δεδομένων PostGIS, εκτελώντας «ασαφή» ερωτήματα (fuzzy queries) και μέσω πολυκριτηριακής ανάλυσης, λαμβάνοντας υπόψη το ύψος και την χρήση των κτιρίων, την απόστασή τους από έναν ποταμό και εάν βρίσκονται ή όχι μέσα στις γραμμές πλημμύρας, ώστε να ορίσει, ποια κτίρια κινδυνεύουν σε περίπτωση υπερχείλισης ενός ποταμού στην Σλοβακία.

Με την αύξηση της χρήσης των NoSQL βάσεων δεδομένων, οι ερευνητές ήρθαν αντιμέτωποι με τα προτερήματα και τα μειονεκτήματα της χρήσης τους. Αντιλαμβάνεται κανείς, πως παρότι οι NoSQL βάσεις δεδομένων είναι σχεδιασμένες για τη ταυτόχρονη πρόσβαση από πολλούς χρήστες και τη διαχείριση μεγάλων όγκων δεδομένων, δεν βρίσκονται, προς το παρόν, στο επιθυμητό επίπεδο ώστε να επιλύσουν τα προβλήματα και να υπερκεράσουν τις προκλήσεις που θέτει η χρήση των γεωχωρικών δεδομένων.

Ένα από τα προβλήματα που συναντά ο χρήστης στις NoSQL βάσεις δεδομένων, είναι η έλλειψη ενιαίας γλώσσας ερωτημάτων (query language), καθώς διαφορετικές βάσεις χρησιμοποιούν διαφορετικές γλώσσες [31]. Η επιστημονική κοινότητα, εργάζεται για την επίλυση του άνωθεν προβλήματος, ενδεικτικά αναφέρεται πως οι Ong et al. (2014) [32],

κατασκεύασαν έναν επεξεργαστή ερωτημάτων (query processor), ο οποίος δέχεται ερωτήματα στη γλώσσα SQL++ (Semi – structured Query Language) [33], «μεταφράζει» τα ερωτήματα στην γλώσσα ερωτημάτων της εκάστοτε NoSQL βάσης δεδομένων και επιστρέφει τα αποτελέσματα πάλι σε μορφή SQL++. Η κοινοπραξία για τον παγκόσμιο ιστό (World Wide Web Consortium | W3C), ανέπτυξε τη γλώσσα XQuery [34], που βασίζεται στην γλώσσα XML, ως ενιαία γλώσσα ερωτημάτων για τις XML βάσεις δεδομένων. Οι Buneman et al. (1996) [35], ανέπτυξαν την UnQL (Unstructured Query Language) ως γλώσσα ερωτημάτων σε NoSQL βάσεις δεδομένων. Οι Atzeni et al. (2012) [36], σχεδίασαν το εργαλείο SOS (Save Our System), όπου ο χρήστης μπορεί να εκτελεί ερωτήματα μέσω της Java και να λαμβάνει δεδομένα σε μορφή JSON [37].

Επιπλέον, στις NoSQL βάσεις δεδομένων, οι χωρικές λειτουργίες (spatial functions) που διατίθενται για την ανάλυση των γεωχωρικών δεδομένων είναι περιορισμένες, συγκριτικά με τις διαθέσιμες χωρικές λειτουργίες των SQL βάσεων δεδομένων [38]. Οι Makris et al. (2019) [39], στην εργασία τους συγκρίνουν συστήματα βάσεων δεδομένων για γεωχωρικά δεδομένα (Redis, HBase Spatial, MongoDB, Neo4j, PostgreSQL), καταγράφουν τις χωρικές λειτουργίες που διαθέτει το κάθε σύστημα και καθιστούν προφανές πως οι NoSQL βάσεις δεδομένων ανταποκρίνονται στις βασικές ανάγκες των χρηστών για ανάλυση γεωχωρικών δεδομένων (π. χ. απλά ερωτήματα τοπολογίας), αλλά βρίσκονται ακόμη σε πρώιμο στάδιο για να καλύψουν τις πολύπλοκες ανάγκες για την ανάλυση γεωχωρικών δεδομένων. Όμως έχουν την πεποίθηση πως με την αύξηση της εξάρτησης από τα γεω – χωρικά δεδομένα, η επιστημονική κοινότητα θα συνεχίσει να υποστηρίζει την ανάπτυξή τους. Οι Bartoszewski et al. (2019) [40], συνέκριναν δύο συστήματα βάσεων δεδομένων (PostGIS – MongoDB), επισημαίνοντας πως η MongoDB (NoSQL), παρέχει μόνο βασικές χωρικές λειτουργίες και κατέληξαν στο συμπέρασμα πως, ερωτήματα «εμπεριέχεται» (within) και «τέμνει» (intersect), εκτελούνται γρηγορότερα στη MongoDB, αντιθέτως η PostGIS είναι γρηγορότερη όταν το η λειτουργία αφορά την επιλογή γειτνιάζοντος σημείου (neighbor), ιδιαίτερα όταν αυξάνεται η ακτίνα (radius) αναζήτησης. Αναφέρουν πως οι σχεσιακές βάσεις δεδομένων, είναι καλύτερη επιλογή εάν απαιτείται πολύπλοκη ανάλυση των δεδομένων, λόγο της πληθώρας χωρικών λειτουργιών που διαθέτουν, ενώ πιστεύουν πως οι NoSQL είναι καλύτερη επιλογή για την ταυτόχρονη επεξεργασία μεγάλου όγκου

δεδομένων και πως με τη συνεχή εξέλιξή τους, στο μέλλον θα παρέχον πλήθος χωρικών λειτουργιών.

Επιπρόσθετα, ένα κοινό πρόβλημα που παρουσιάζεται τόσο στις SQL όσο και στις NoSQL βάσεις δεδομένων, είναι η έλλειψη κατάλληλων αλγορίθμων και εργαλείων για την ευρετηριοποίηση των δεδομένων, όταν ο όγκος τους αυξάνεται (big data), ώστε να καθίσταται δυνατή η επεξεργασία τους σε πραγματικό χρόνο [41]. Σύμφωνα με τους Khan et al. (2019) [42], η δυσκολία ευρετηριοποίησης των γεωχωρικών δεδομένων έγκειται στην επεκτασιμότητα των δεδομένων (data scalability), το χρόνο που απαιτείται για την ευρετηριοποίηση (index building time) και την υψηλή διάσταση των δεδομένων (high dimensional indexing). Επιπλέον, κάνουν αναφορά στις μεθόδους ευρετηριοποίησης που χρησιμοποιούνται στα συστήματα διαχείρισης βάσεων δεδομένων, για γεω – χωρικά δεδομένα, παραθέτοντας εν συντομία, τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μεθόδου. Τέλος, επισημαίνουν τα προβλήματα που συναντώνται κατά την ευρετηριοποίηση των γεωχωρικών δεδομένων στα συστήματα διαχείρισης βάσεων δεδομένων. Οι Kouiroukidis et al., (2011) [43], διερεύνησαν την χρήση δημοφιλών μεθόδων ευρετηριοποίησης δεδομένων (Hybrid Tree, R* – Tree, iDistance Method), σε πολυδιάστατα δεδομένα, για ερωτήματα γειτνίασης (k Nearest Neighbor). Αναφέρονται στο πρόβλημα της «κατάρας της διάστασης» (dimensionality curse) [44] και διερεύνησαν τον αριθμό των διαστάσεων των δεδομένων κατά τον οποίο, οι μέθοδοι ευρετηριοποίησης παύουν να είναι αποδοτικές, με αποτέλεσμα να απαιτείται σειριακός έλεγχος ολόκληρης της βάσης δεδομένων για την επιστροφή των αποτελεσμάτων σε ερωτήματα kNN. Απέδειξαν πως για την αναζήτηση κοντινότερου γείτονα (kNN search) σε πολυδιάστατα δεδομένα, από οκτώ (8) διαστάσεις και πάνω, οι μέθοδοι ευρετηριοποίησης που χρησιμοποιούνται δεν είναι αποτελεσματικές.

Οι Krommyda et al. (2020) [45], ερευνώντας τη διαχείριση γεωχωρικών δεδομένων στο πλαίσιο του διαδικτύου των πραγμάτων, κάνουν εκτενή αναφορά στις μεθόδους ευρετηριοποίησης γεω – χωρικών δεδομένων, παραθέτοντας αναλυτικά τον τρόπο λειτουργίας της κάθε μεθόδου, τα πλεονεκτήματα και τα μειονεκτήματά τους, καθώς και παραλλαγές τους και τα προβλήματα που αυτές επιλύουν ή αντιμετωπίζουν. Επιπλέον, αναλύουν τα διαθέσιμα συστήματα διαχείρισης βάσεων δεδομένων, το μοντέλο

αποθήκευσης που εφαρμόζουν, τον τρόπο λειτουργίας τους καθώς και την μέθοδο ή μεθόδους ευρετηριοποίησης των δεδομένων που διαθέτουν.

3. Συστήματα διαχείρισης βάσεων δεδομένων

3.1 SQL βάσεις δεδομένων

Τα συστήματα διαχείρισης χωρικών βάσεων δεδομένων (Spatial Database Management System | SDBMS) αποτελούν επέκταση των συστημάτων διαχείρισης βάσεων δεδομένων (DBMS). Οι βασικοί τύποι δεδομένων που χειρίζονται οι βάσεις δεδομένων (π. χ. ακέραιοι αριθμοί, δεκαδικοί, αλφαριθμητικά) επεκτείνονται και περιλαμβάνουν τύπους χωρικών δεδομένων (π. χ. σημεία, γραμμές, πολύγωνα). Επιπλέον, προστίθενται χωρικές λειτουργίες (spatial functions), για τη διαχείριση των γεωχωρικών δεδομένων και επεκτείνεται η γλώσσα ερωτημάτων ώστε να εμπεριέχει τις νέες λειτουργίες. Ακόμη επεκτείνονται οι μέθοδοι ευρετηριοποίησης δεδομένων, για την χρήση τους σε γεω – χωρικά δεδομένα [46]. Τα συστήματα διαχείρισης βάσεων δεδομένων μπορούν να χωριστούν σε δύο κύριες κατηγορίες: SQL και NoSQL, ανάλογα με τη δομή που χρησιμοποιείται για την αποθήκευση των δεδομένων. Στην παρούσα ενότητα αναλύονται τα SQL συστήματα βάσεων δεδομένων.

Ο όρος «SQL DBMS» χρησιμοποιείται για την αναφορά σε σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων. Στα συστήματα αυτά τα δεδομένα αποθηκεύονται σε πίνακες (tables) και οι σχέσεις ορίζονται με μαθηματικές μεθόδους. Οι πίνακες χωρίζονται σε γραμμές (rows) και στήλες (columns), όπου κάθε γραμμή αποτελεί μία εγγραφή στη βάση και οι στήλες ορίζουν το σχήμα της βάσης [47]. Επιπλέον, οι σχεσιακές βάσεις δεδομένων, βασίζονται στις αρχές «ACID» (Atomicity, Consistency, Isolation, Durability), για τη διατήρηση της ακεραιότητας της βάσης και των δεδομένων σε αυτήν. Η «ατομικότητα» (Atomicity), διασφαλίζει πως κάθε «συναλλαγή» (transaction) στη βάση, αντιμετωπίζεται ως μεμονωμένο γεγονός. Η «συνέπεια» (Consistency) διασφαλίζει πως με την εγγραφή των αποτελεσμάτων, μετά από κάθε συναλλαγή, η βάση δεδομένων παραμένει ακέραιη. Η «απομόνωση» (Isolation) εξασφαλίζει πως δεν υπάρχουν ανεπιθύμητα αποτελέσματα μετά από παράλληλη επεξεργασία των δεδομένων της βάσης. Η «ανθεκτικότητα» (Durability),

αναφέρεται στην συνέπεια και συνέχεια των δεδομένων στη βάση, με το πέρας της επεξεργασίας [48]. Παρακάτω παρατίθενται κάποια από τα ευρέως χρησιμοποιούμενα συστήματα διαχείρισης χωρικών βάσεων δεδομένων, ανοικτού κώδικα (open source).

3.1.1 MySQL Spatial

Η MySQL αποτελεί σχεσιακό σύστημα διαχείρισης δεδομένων και υποστηρίζει την αποθήκευση και επεξεργασία γεω – χωρικών δεδομένων μέσω του προσθέτου (plugin) MySQL Spatial [49]. Το πρόσθετο, επεκτείνει τη βασική γλώσσα ερωτημάτων SQL, για τη διαχείριση των γεωχωρικών δεδομένων. Έχει σχεδιαστεί βάσει του προτύπου «Simple Feature Access – Part 2: SQL Option» (SFP) [50], το οποίο ορίζει μεθόδους για την επέκταση των σχεσιακών βάσεων δεδομένων για τη χρήση χωρικών δεδομένων, τις βασικές αρχές της γεωμετρίας των δεδομένων, καθώς και τις λειτουργίες που απαιτούνται για την ανάλυση των δεδομένων στη βάση. Επιπροσθέτως, επεκτείνει τους τύπους χωρικών δεδομένων που δέχεται με τη χρήση του προτύπου SQL / MM Part3: Spatial [51]. Τέλος, παρέχει τη δυνατότητα ευρετηριοποίησης δεδομένων, με δύο μεθόδους: α) R – tree index [52], με τετραγωνική διάσπαση (quadratic splitting), όπου δημιουργείται ελάχιστο πλαίσιο οριοθέτησης (Minimum Bounding Rectangle | MBR) για τη γεωμετρία των δεδομένων, και β) B - tree index [53]. Για τη δημιουργία χωρικής ευρετηριοποίησης, απαιτείται η χρήση της λέξης κλειδί «SPATIAL» και τα δεδομένα στα οποία θα γίνει η ευρετηριοποίηση δεν θα πρέπει να εμπεριέχουν «NULL» τιμές. Η χρήση των άνωθεν μεθόδων ευρετηριοποίησης επιτρέπει τη δυναμική αλλαγή της βάσης δεδομένων, δηλαδή εγγραφή νέων δεδομένων ή διαγραφή δεδομένων, χωρίς να απαιτείται εκ νέου ευρετηριοποίηση από τον χρήστη. Η MySQL Spatial, κάνει χρήση της βιβλιοθήκης GDAL [54], για την εισαγωγή και επεξεργασία γεω – χωρικών δεδομένων εικονοστοιχείων στη βάση δεδομένων.

3.1.2 SpatiaLite

Η SpatiaLite [55], αποτελεί βιβλιοθήκη ανοικτού κώδικα η οποία αναπτύχθηκε ώστε να επεκτείνει το σύστημα διαχείρισης βάσεων δεδομένων SQLite, για χρήση γεωχωρικών δεδομένων. Εφαρμόζει τα πρότυπα SFP και SQL / MM Part3: Spatial, που ορίζουν τις δομές δεδομένων, τους τύπους δεδομένων που δέχεται η βάση και τις λειτουργίες που απαιτούνται για την ανάλυση και επεξεργασία των γεωχωρικών δεδομένων. Υποστηρίζει τη χρήση διανυσματικών αρχείων και αρχείων εικόνων μέσω της βιβλιοθήκης RasterLite2

(αντικατέστησε την βιβλιοθήκη RasterLite) [56]. Επιπλέον, ενσωματώνει τις βιβλιοθήκες: α) GEOS, που χρησιμοποιείται για την ενσωμάτωση επιπλέων χωρικών συσχετίσεων (spatial predicates) στην χωρική ανάλυση, β) PROJ (προηγουμένως PROJ4), για τη μετατροπή συντεταγμένων σε διαφορετικά συστήματα και γ) LIBICONV, που υποστηρίζει τη χρήση διαφορετικών γλωσσών [57]. Για την αποθήκευση συντεταγμένων, χρησιμοποιεί τόσο προβολικές συντεταγμένες (X , Y) όσο και γεωγραφικές συντεταγμένες (lat, long). Χρησιμοποιεί την μέθοδο ευρετηριοποίησης R* tree [58], όπου κάθε κόμβος του δέντρου εμπεριέχει τον δείκτη (index), του ελάχιστου πλαισίου οριοθέτησης, όλων των δεδομένων που εμπεριέχονται στους κόμβους του επόμενου επιπέδου (child nodes). Πρέπει να σημειωθεί πως για την εύρυθμη λειτουργία της άνωθεν μεθόδου ευρετηριοποίησης στην SpatiaLite, θα πρέπει να προηγηθεί δημιουργία πρωτεύοντος κλειδιού σε κάθε πίνακα (table) της βάσης, καθώς σε άλλη περίπτωση, ενδέχεται να επιστραφούν λάθος αποτελέσματα [59].

Η SpatiaLite, δεν απαιτεί πολλούς υπολογιστικούς πόρους (lightweight application), καθώς είναι σχεδιασμένη να χρησιμοποιείται και σε κινητές συσκευές και όπως η SQLite, η SpatiaLite δεν παρέχει δυνατότητα πολλαπλής ταυτόχρονης πρόσβασης [60].

3.1.3 PostGIS

Η PostGIS, αποτελεί επέκταση του συστήματος διαχείρισης βάσεων δεδομένων Postgres, για την αποθήκευση, ανάλυση και διαχείριση των γεωχωρικών δεδομένων. Αποτελεί το πιο διαδεδομένο σύστημα διαχείρισης χωρικών βάσεων δεδομένων, ανοικτού κώδικα [61]. Υποστηρίζει και τα δύο πρότυπα (SFP και SQL / MM Part3: Spatial), που ορίζουν τις δομές δεδομένων, τους τύπους δεδομένων που δέχεται η βάση και τις λειτουργίες που απαιτούνται για την ανάλυση και επεξεργασία των γεωχωρικών δεδομένων. Επιπλέον, υποστηρίζει τη χρήση γεωγραφικών και καρτεσιανών συντεταγμένων, καθώς επίσης τις μετατροπές συντεταγμένων μεταξύ διαφορετικών συστημάτων, μέσω της βιβλιοθήκης PROJ [62]. Διαθέτει περισσότερες από χίλιες γεω – χωρικές λειτουργίες [38], οι οποίες μπορούν να χωριστούν σε πέντε κατηγορίες [63]:

- Διαχείρισης (Management): αναφέρεται στις λειτουργίες για τη διαχείριση των πινάκων, μέσα στη βάση, καθώς και λειτουργίες διαχειριστή που παρέχονται για το ίδιο το σύστημα διαχείρισης χωρικών βάσεων δεδομένων.

- Μετατροπής (Conversion): είναι οι λειτουργίες που επιτρέπουν μετατροπές στη γεωμετρία των δεδομένων, καθώς και μετατροπές στον τύπο των δεδομένων.
- Ανάκτησης (Retrieval): λειτουργίες οι οποίες διευκολύνουν την ανάκτηση των χαρακτηριστικών (attributes) των δεδομένων, καθώς και της γεωμετρίας τους.
- Σύγκρισης (Comparison): λειτουργίες για τη σύγκριση της γεωμετρίας των δεδομένων μεταξύ τους, αναφορικά με την χωρική τους συσχέτιση.
- Δημιουργίας (Creation): λειτουργίες που επιτρέπουν τη δημιουργία νέων γεωμετριών, από ήδη υπάρχουσες.

Επιτρέπει την επεξεργασία πληθώρας χωρικών ερωτημάτων, για τον ορισμό συσχετίσεων μεταξύ των δεδομένων. Τα ερωτήματα μπορούν να χωριστούν σε τρεις κύριες κατηγορίες [64]:

- Ερωτήματα απόστασης (Distance based queries): είναι τα ερωτήματα που θέτει ο χρήστης σχετικά με τη γειτνίαση μεταξύ των αντικειμένων . Η PostGIS μπορεί να υπολογίσει την απόσταση στο γεωγραφικό χώρο (lat,long) και να επιστρέψει την απόσταση σε μέτρα, με τη χρήση της λειτουργίας «ST_Distance_Sphere», η οποία υπολογίζεται λαμβάνοντας υπόψη το σφαιροειδές σχήμα της Γης, με ακτίνα 6.370.986 μέτρα. Είναι πιο γρήγορη από τη λειτουργία «ST_Distance_Spheroid», αλλά λιγότερο ακριβής. Επιπλέον, δεν λαμβάνει υπόψη το σύστημα συντεταγμένων του αντικειμένου και θεωρεί πως χρησιμοποιείται το παγκόσμιο γεωδαιτικό σύστημα συντεταγμένων WGS 84 (lat,long) [65]. Ένα παράδειγμα τέτοιου ερωτήματος είναι, πόσα κτίρια βρίσκονται λιγότερο από εκατό μέτρα από το μέγαρο μουσικής Θεσσαλονίκης;
- Ερωτήματα προσανατολισμού (Direction based queries): χρησιμοποιούνται για την εύρεση της θέσης των αντικειμένων στο χώρο. Ένα παράδειγμα τέτοιου ερωτήματος είναι, πόσα αντικείμενα της βρίσκονται βορειότερα από τον Ισημερινό;

- Ερωτήματα τοπολογίας (Topological queries): είναι ερωτήματα που προσδιορίζουν τις χωρικές συσχετίσεις μεταξύ των αντικειμένων της βάσης. Για παράδειγμα, έχουν δύο πολύγωνα κοινό σύνορο μεταξύ τους;

Η PostGIS, χρησιμοποιεί τρεις διαφορετικές μεθόδους ευρετηριοποίησης γεω – χωρικών δεδομένων. Χρησιμοποιεί τη μέθοδο GiST (Generalized Search Tree) [66], η οποία χρησιμοποιεί την μέθοδο R – Tree (R – Tree over GiST). Τη μέθοδο BRIN (Block Range Index) [67], που η χρήση της συνίσταται σε βάσεις δεδομένων, όπου δεν θα υπάρχουν αλλαγές στα δεδομένα. Είναι γρήγορη μέθοδος ευρετηριοποίησης και παράγει ευρετήρια μικρού μεγέθους (small index size). Τέλος, χρησιμοποιεί και τη μέθοδο SP – GiST (Space – Partitioned Generalized Search Tree) [68], που αποτελεί γενική μέθοδο ευρετηριοποίησης (general index framework) και υποστηρίζει μεθόδους ευρετηριοποίησης, βασιζόμενες στο διαχωρισμό του χώρου. Τέτοιες μέθοδοι είναι οι: Quad – trees [69], k – d trees [70] και radix trees (tries) [71].

Η PostGIS, υποστηρίζει τη χρήση δεδομένων εικόνων μέσω της βιβλιοθήκης GDAL, υποστηρίζοντας διαφόρους τύπους δεδομένων (π.χ. GeoTIFF, PNG, JPEG) και διαθέτει πλήθος λειτουργιών για επεξεργασία, ανάλυση, αποθήκευση και εξαγωγή των δεδομένων αυτών.

3.2 NoSQL βάσεις δεδομένων

Όπως αναφέρθηκε στην προηγούμενη ενότητα, τα συστήματα διαχείρισης βάσεων δεδομένων, δύναται να χωριστούν σε δύο κατηγορίες, SQL συστήματα και τα NoSQL συστήματα. Η παρούσα ενότητα, αναφέρεται στα NoSQL συστήματα διαχείρισης βάσεων δεδομένων. Σημειώνεται εδώ, πως δεν υπάρχει ένας καθολικά αποδεκτός ορισμός για τα NoSQL συστήματα, αλλά ο Cattell, (2011) [72], παραθέτει τις βασικές ιδιότητες που χαρακτηρίζουν τα συστήματα αυτά. Τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων, χωρίζονται σε τέσσερις κατηγορίες, ανάλογα με τον τρόπο που αποθηκεύονται τα δεδομένα σε αυτά [73-74]:

Key – Value (Κλειδί – Τιμή): τα δεδομένα αποθηκεύονται σε ζεύγη, όπου το «Key» είναι το μοναδικό αναγνωριστικό (unique ID), που δείχνει τα αντίστοιχα δεδομένα με τα οποία συσχετίζεται. Επιτρέπει την εκτέλεση λειτουργιών βασιζόμενες στο κλειδί (get, put, delete). Η τιμή (value), δεν έχει περιορισμό ως προς τον τύπο τον δεδομένων που μπορούν να εισαχθούν (π. χ. αλφαριθμητικό, ακέραιος αριθμός, JSON…).

Document (Αρχείο): επεκτείνει το προηγούμενο σύστημα, βασίζεται σε ζεύγη «Key – Value», απαιτείται τα δεδομένα στη «Τιμή» να είναι δομημένα και να διαθέτουν μετα – δεδομένα. Υποστηρίζει μεταξύ άλλων, κωδικοποίηση δεδομένων σε αρχεία XML (Extensible Markup Language), JSON (JavaScript Object Notation), BSON (Binary JSON) και η πιο κοινή κωδικοποίηση για γεω – χωρικά δεδομένα GeoJSON, που χρησιμοποιείται για την αναπαράσταση χωρικής πληροφορίας.

Column (Στήλη): Αποθηκεύει τα δεδομένα σε πίνακες (tables) με γραμμές (rows) και στήλες (columns). Η διαφορά με τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων είναι, πως τα δεδομένα αποθηκεύονται στις στήλες, και όχι στις γραμμές. Οι στήλες χωρίζονται σε «γκρουπ» (column families), όπου το κάθε γκρουπ μπορεί να έχει άπειρο αριθμό στηλών. Με τον τρόπο αυτό είναι γρηγορότερη η πρόσβαση και η αναζήτηση δεδομένων στη βάση, συγκριτικά με την αποθήκευση δεδομένων σε γραμμές, όπου η προσπέλαση της βάσης απαιτεί πολλούς υπολογιστικούς πόρους, ιδίως εάν η κάθε γραμμή περιλαμβάνει πολλές στήλες.

Graph (Γράφος): αποθηκεύει τα δεδομένα ως γράφο, όπου τα δεδομένα αποτελούν τους κόμβους (nodes) και οι σχέσεις μεταξύ τους ορίζονται ως ακμές (edges) του γράφου. Τόσο οι κόμβοι του γράφου, όσο και οι ακμές του, δύνανται να έχουν ιδιότητες (properties). Σε αυτή τη δομή, οι σχέσεις μεταξύ των αντικειμένων αποτελούν προτεραιότητα. Έτσι η ανάκτηση δεδομένων που συσχετίζονται μεταξύ τους, γίνεται ευκολότερη και ταχύτερη από ότι στις σχεσιακές βάσεις δεδομένων, καθώς δεν απαιτείται “JOIN”, μεταξύ πινάκων, μία υπολογιστικά απαιτητική διαδικασία.

Όπως αναφέρθηκε και στην εισαγωγή της παρούσας εργασίας, τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων, υιοθετούν πιο ευέλικτο σχήμα από τα σχεσιακά συστήματα και έχουν καλύτερη επεκτασιμότητα, καθώς είναι σχεδιασμένα για την αποθήκευση,

ανάλυση και διαχείριση πολύ μεγάλων όγκων δεδομένων, πολλές φορές αδόμητων ή ημι – δομημένων δεδομένων και τη χρήση σε κατανεμημένα συστήματα. Για να επιτευχθούν τα άνωθεν όμως απώλεσαν (trade – off) τις ιδιότητες ACID. Αντ' αυτού τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων ακολουθούν το θεώρημα CAP (Consistency – Availability – Partition tolerance) [75] και τις αρχές BASE (Basically Available, Soft State, Eventually Consistent) [76]. Πρέπει να σημειωθεί πως κανένα σύστημα δεν ικανοποιεί και τις τρεις αρχές του θεωρήματος CAP. Παρακάτω παρατίθενται ενδεικτικά τα πιο δημοφιλή NoSQL συστήματα διαχείρισης βάσεων δεδομένων, για κάθε κατηγορία σύμφωνα με το DB – Engines Ranking [77].

3.2.1 Redis (Remote Dictionary Service)

Είναι Key – Value σύστημα διαχείρισης βάσεων δεδομένων στη μνήμη (in – memory), το οποία παρέχει ταχύτατη ανάγνωση και εγγραφή των δεδομένων, καθώς και ανάλυσή τους [78]. Το γεγονός ότι αναπτύσσεται στη μνήμη του συστήματος, σημαίνει ότι περιορίζεται ο όγκος των δεδομένων που μπορεί να αποθηκευτεί στο σύστημα. Η δομή της, επιβάλλει τα δεδομένα ως συλλογή (collection), ορίζοντας το κλειδί ως μοναδικό δείκτη για κάθε αρχείο δεδομένων. Δεν ενέχει περιορισμούς ως προς τον τύπο δεδομένων που μπορούν να χρησιμοποιηθούν ως key ή value. Αποθηκεύει τα ζεύγη συντεταγμένων (long, lat), ορίζοντας σε κάθε ζεύγος ένα μοναδικό κλειδί, ως εγγραφές σε ταξινομημένο σύνολο (score in sorted set). Υποστηρίζει σημειακά δεδομένα, παρέχει δέκα χωρικές λειτουργίες και χρησιμοποιεί χωρικό κατακερματισμό (geohash) [79], για την ευρετηριοποίηση των δεδομένων. Μπορεί να επιτευχθεί η εισαγωγή διαφόρων τύπων χωρικών δεδομένων (πέρα των σημειακών), με τη χρήση GeoJSON μορφής δεδομένων. Επιπλέον, με τη χρήση της εξωτερικής βιβλιοθήκης Lua (Lua library) [80], ο χρήστης μπορεί να εμπλουτίσει τις χωρικές λειτουργίες της βάσης. Τέλος, χρησιμοποιεί τη φόρμουλα Haversine [81], για τον υπολογισμό αποστάσεων. Εξαιτίας της ταχύτητας που προσφέρει, χρησιμοποιείται σε συστήματα παρακολούθησης (tracking systems) [82], τα οποία έχουν αυξημένες απαιτήσεις για την επεξεργασία και οπτικοποίηση των δεδομένων σε πραγματικό χρόνο.

3.2.2 MongoDB

Είναι Document σύστημα διαχείρισης βάσεων δεδομένων, παρέχει εύκολη επεκτασιμότητα, ευελιξία και αποθηκεύει δεδομένα σε μορφή JSON και χρησιμοποιεί μορφή

GeoJSON για την αποθήκευση και διαχείριση γεωχωρικών δεδομένων. Τα δεδομένα αποθηκεύονται σε συλλογές στη βάση (Documents), και αρχεία με διαφορετικές δομές δεδομένων μπορούν να αποθηκευτούν στην ίδια συλλογή (schema – free). Επιπλέον, ο χρήστης μπορεί να αποθηκεύσει αρχεία, το ένα μέσα στο άλλο (nested documents), για τη διευκόλυνση των συσχετίσεων μεταξύ τους [83]. Δέχεται καρτεσιανές (X, Y) και γεωγραφικές συντεταγμένες (long, lat). Υποστηρίζει διάφορους τύπους γεωμετρίας: σημείο, γραμμή, πολύγωνο, λίστα σημείων (MultiPoint), λίστα γραμμών (MultiLineString), λίστα πολυγώνων (MultiPolygon) και συλλογή γεωμετρικών δεδομένων (GeometryCollection). Για την ευρετηριοποίηση των δεδομένων χρησιμοποιεί τις μεθόδους 2d indexes για γεωμετρικά δεδομένα και 2dsphere indexes για γεωγραφικά δεδομένα. Υποστηρίζει τις χωρικές λειτουργίες: $geoIntersects, $geoWithin, $near και $nearSphere [84]. Για τη χρήση γεωχωρικών δεδομένων εικόνας (raster), διατίθεται ο μηχανισμός GridFS [85], ο οποίος παρέχει τη δυνατότητα αποθήκευσης και χρήσης μεγάλων δυαδικών αρχείων (binary files), τα οποία χωρίζει σε μικρότερα αρχεία και τα αποθηκεύει σε συλλογές δεδομένων (documents) στη βάση. Οι Wang et al., (2014) [86], χρησιμοποίησαν τον μηχανισμό GridFS για την αποθήκευση νέφους σημείων (point cloud-LiDAR), ενώ οι Wang et al., (2019) [87], βασίστηκαν στο μηχανισμό, για την ανάπτυξη μίας μεθόδου αποθήκευσης και ανάκτησης δορυφορικών δεδομένων.

3.2.3 Cassandra

Αποτελεί Column σύστημα διαχείρισης βάσεων δεδομένων και χρησιμοποιεί τη γλώσσα ερωτημάτων CQL (Cassandra Query Language) [88]. Το σύστημα αυτό δεν παρέχει (native) τη δυνατότητα χρήσης γεωχωρικών δεδομένων [89], παρά μόνο με τη χρήση του προσθέτου Stratio's Cassandra Lucene Index [90]. Το πρόσθετο επιτρέπει την εισαγωγή, αποθήκευση και επεξεργασία γεω – χωρικών δεδομένων, παρέχει χωρικές λειτουργίες (π. χ. intersection, union, difference). Για την ευρετηριοποίηση των γεω – χωρικών δεδομένων χρησιμοποιεί τη μέθοδο κατακερματισμού του χώρου σε δέντρο προθέματος (geohash recursive prefix tree). Έχουν σχεδιαστεί εργαλεία [91] και επεκτάσεις [88], που παρέχουν πληθώρα εργαλείων για την εισαγωγή και επεξεργασία γεω – χωρικών δεδομένων, για το παρόν σύστημα διαχείρισης βάσεων δεδομένων.

3.2.4 Neo4j

Αποτελεί Graph σύστημα διαχείρισης βάσεων δεδομένων και χρησιμοποιεί τη γλώσσα ερωτημάτων Cypher [92] για την αποθήκευση, επεξεργασία και ανάλυση των δεδομένων. Υποστηρίζει (native) μόνο σημειακά χωρικά δεδομένα. Τα δεδομένα μπορεί να είναι σε Καρτεσιανές ή γεωγραφικές συντεταγμένες και να έχουν δύο (2D) ή τρεις (3D) διαστάσεις. Υποστηρίζει χωρικές λειτουργίες που σχετίζονται με σημειακά δεδομένα (π. χ. point.distance()). Για τον υπολογισμό των αποστάσεων στο επίπεδο, χρησιμοποιείται το Πυθαγόρειο θεώρημα, ενώ για τον υπολογισμό των αποστάσεων στο γεωγραφικό χώρο, χρησιμοποιείται τη φόρμουλα Haversine και εάν τα σημεία έχουν και τρίτη διάσταση, χρησιμοποιείται πρώτα η φόρμουλα Haversine, λαμβάνοντας υπόψη το μέσο υψόμετρο μεταξύ των δύο σημείων και έπειτα χρησιμοποιείται το Πυθαγόρειο θεώρημα, υπολογίζοντας και την υψομετρική διαφορά των σημείων. Για την ευρετηριοποίηση των δεδομένων χρησιμοποιεί δύο μεθόδους. Για την ευρετηριοποίηση σημειακών δεδομένων (Point Index) χρησιμοποιεί καμπύλες πλήρωσης χώρου (space filling curve), για 2D και 3D δεδομένα, και γενικευμένο B+ Tree. Ενώ για την ευρετηριοποίηση των υπολοίπων δεδομένων (π. χ. ιδιότητες, συσχετίσεις) (Range Index), ευρετηριοποιεί τα δεδομένα σε λεξικογραφική σειρά, βάσει του συστήματος συντεταγμένων [93]. Ακόμη, υπάρχει το πρόσθετο Neo4j Spatial [94], το οποίο διευκολύνει τη χρήση γεω – χωρικών δεδομένων, υποστηρίζει τη χρήση επιπλέον τύπων γεωμετρίας (LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection), και προσθέτει πληθώρα χωρικών λειτουργιών για επεξεργασία και ανάλυση των δεδομένων. Μέσω του Neo4j Spatial, καθίσταται δυνατή η εισαγωγή δεδομένων διαφορετικών τύπων (π. χ. Shapefile, CSV, WKT…). Πρέπει να τονιστεί πως το πρόσθετο, για την ευρετηριοποίηση των δεδομένων χρησιμοποιεί τη μέθοδο R – Tree. Η δομή δεδομένων του συστήματος, με γράφους, το καθιστά ιδανική επιλογή για εφαρμογές όπου οι συσχετίσεις μεταξύ των δεδομένων είναι πολύπλοκες, ή είναι πολύ σημαντικές για την εφαρμογή. Επίσης, αποτελούν ιδανική λύση για την ανάλυση δικτύων (network analysis) [95-96]. Από την μελέτη της βιβλιογραφίας και των τεχνικών φυλλαδίων (manuals), προκύπτει πως μέχρι τη στιγμή που γράφεται η παρούσα εργασία, το παρόν σύστημα δεν υποστηρίζει την ανάλυση χωρικών δεδομένων εικόνων (Raster).

4. Συγκριτική μελέτη

Λόγο του ενδιαφέροντος που προσελκύουν τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων, πολλοί ερευνητές έχουν προχωρήσει σε συγκριτικές μελέτες, ανάμεσα στα SQL και τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων, με σκοπό την αξιολόγησή τους και την εύρεση της βέλτιστης επιλογής, για την εκάστοτε περίπτωση μελέτης. Παρακάτω παρατίθενται κάποιες από αυτές.

Οι Agarwal et al., (2017) [9], συνέκριναν δύο συστήματα διαχείρισης βάσεων δεδομένων (MongoDB – PostGIS), βάσει της απόδοσής τους στην επίλυση ενός προβλήματος «σημειακής συγκράτησης» (Point Containment problem). Από τη σύγκριση, με και χωρίς τη χρήση ευρετηριοποίησης των δεδομένων, κατέληξαν στο συμπέρασμα πως το NoSQL σύστημα, εκτέλεσε την ανάλυση και στις δύο περιπτώσεις γρηγορότερα. Παρατήρησαν επίσης, πως η NoSQL βάση, είχε καλύτερη απόδοση όσο αυξάνονταν ο όγκος των δεδομένων, ενώ ο χρόνος ανάλυσης των δεδομένων στην SQL βάση δεδομένων, αυξάνεται εκθετικά όταν αυξάνεται (πολύ) ο όγκος των δεδομένων. Οι Schmid et al., (2015) [38], συνέκριναν δύο NoSQL συστήματα διαχείρισης βάσεων δεδομένων (MongoDB – CouchBase) και ένα SQL σύστημα (PostGIS), βάσει των δυνατοτήτων τους στην ανάλυση γεω – χωρικών δεδομένων, και εκτελώντας δύο ελέγχους αποδοτικότητας (performance tests): α) ερώτημα βάσει της περιγραφικής πληροφορίας (query on attribute information), β) χρήση της τοπολογικής λειτουργίας «within» (request using geo – function “within”). Οι έλεγχοι εκτελέστηκαν με τη χρήση διανυσματικών αρχείων δεδομένων. Συμπέραναν, πως τα NoSQL συστήματα, είναι ταχύτερα στην ανάκτηση πληροφορίας, για ερωτήματα που αναφέρονται στην περιγραφική πληροφορία των δεδομένων. Ο δεύτερος έλεγχος έγινε ανάμεσα στη MongoDB και τη PostGIS βάση δεδομένων και κατέληξαν στο συμπέρασμα πως για τη MongoDB βάση δεδομένων, ο χρόνος ανάλυσης είναι γραμμικός όσο αυξάνεται ο όγκος των δεδομένων, ενώ ο χρόνος της ανάλυσης στην PostGIS βάση δεδομένων, αυξάνεται ταχύτατα όσο αυξάνεται ο όγκος των δεδομένων στη βάση. Τέλος, επισημαίνουν πως τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων υπολείπονται σε χωρικές λειτουργίες των SQL συστημάτων, καθώς διαθέτουν μόνο βασικές χωρικές λειτουργίες. Οι Hasan et al., (2019) [47], έκαναν σύγκριση τεσσάρων NoSQL συστημάτων διαχείρισης βάσεων δεδομένων (MongoDB, Cassandra, CouchBase, Neo4j), με ένα SQL σύστημα (PostGIS), πραγματοποιώντας ανάλυση απόδοσης με τη χρήση δεδομένων σε μορφή αρχείου

GeoJSON. Η σύγκριση των συστημάτων διαχείρισης βάσεων δεδομένων έγινε στις ακόλουθες κατηγορίες: α) χρόνος εγγραφής των δεδομένων από το αρχείο GeoJSON στη βάση δεδομένων, β) χωρικά ερωτήματα με και χωρίς ευρετηριοποίηση των δεδομένων και γ) ερωτήματα για τη περιγραφική πληροφορία των δεδομένων και ερωτήματα θέσης (location based queries). Στα αποτελέσματά τους αναφέρουν πως τα συστήματα NoSQL (Document, MongoDB και CouchBase), είναι τα πιο γρήγορα για την εγγραφή των δεδομένων στη βάση από GeoJSON αρχείο, ενώ το πιο αργό σύστημα είναι η Neo4j. Στην εφαρμογή χωρικών ερωτημάτων απόστασης (distance query), με τη χρήση μεθόδου ευρετηριοποίησης των δεδομένων και χωρίς, η PostGIS βάση δεδομένων ήταν αυτή που χρειάστηκε τον περισσότερο χρόνο για τον υπολογισμό των αποτελεσμάτων. Ενώ όταν τέθηκε συνδυαστικό ερώτημα, για την ανάλυση της περιγραφικής και της χωρικής πληροφορίας των δεδομένων, τα αποτελέσματα έδειξαν εκ νέου την PostGIS βάση δεδομένων ως αυτή που απαιτεί τον περισσότερο χρόνο για την ανάλυση των δεδομένων. Τέλος, οι ερευνητές επισημαίνουν πως με τη δημιουργία «materialized view» (αντικείμενο βάσης δεδομένων που αποθηκεύει τα αποτελέσματα ενός ερωτήματος σε μια προ-υπολογισμένη, φυσική μορφή) στην PostGIS βάση, επιταχύνεται σημαντικά η διαδικασία, αλλά απαιτείται περισσότερος χώρος στον δίσκο. Ο Baas, (2012) [97], συνέκρινε ένα SQL σύστημα διαχείρισης βάσεων δεδομένων (PostGIS), με ένα NoSQL (Neo4j), με σκοπό την εξαγωγή συμπερασμάτων σχετικά με τα πλεονεκτήματα που προσφέρει το NoSQL σύστημα διαχείρισης βάσεων δεδομένων, έναντι του SQL συστήματος, για την αποθήκευση και ανάλυση γεω – χωρικών δεδομένων. Πραγματοποίησε πέντε διαφορετικούς ελέγχους (test), και μέτρησε τον χρόνο απόκρισης του server, για την κάθε βάση δεδομένων (empty operation), μέτρησε τον αριθμό των χωρικών αντικειμένων που βρίσκονται σε ένα προκαθορισμένο πολύγωνο στον χώρο (bounding box count), ζήτησε από τη βάση να βρει όλα τα αντικείμενα που βρίσκονται μέσα σε μία προκαθορισμένη περιοχή και να επιστρέψει το αποτέλεσμα σε αρχείο μορφής GML (Geography Markup Language), αναζήτησε το κοντινότερο σημείο από ένα προκαθορισμένο σημείο και υπολόγισε την απόσταση από αυτό (closest point) και τέλος αναζήτησε τη συντομότερη διαδρομή ανάμεσα σε δύο σημεία (shortest path), με τη χρήση του αλγορίθμου Dijkstra. Από τη μελέτη προέκυψε πως ο χρόνος εισαγωγής των δεδομένων και ο χώρος που καταλαμβάνει η βάση στον δίσκο μετά την αποθήκευσή τους, είναι αμφότερα πολλαπλάσια για την Neo4j βάση δεδομένων συγκριτικά με την PostGIS βάση, για το σύνολο δεδομένων που χρησιμοποίησε. Αντίθετα, η δημιουργία τοπολογίας δικτύου (routing network topology)

εκτελέστηκε έως και τρεις φορές γρηγορότερα στη βάση Neo4j, από ότι στην PostGIS βάση δεδομένων. Αναφορικά με τη χρήση των χωρικών λειτουργιών που παρέχουν τα δύο συστήματα, κατέληξε στο συμπέρασμα πως η δημιουργία οριοθετημένου πολυγώνου (bounding box) εκτελέστηκε γρηγορότερα στην PostGIS βάση. Ενώ για την εύρεση κοντινότερου σημείου (closest point), τα αποτελέσματα από τις δύο βάσεις δεδομένων, ήταν παρόμοια. Στο ερώτημα που έθεσε για την εύρεση συντομότερης διαδρομής, η Neo4j βάση δεδομένων επέστρεψε τα αποτελέσματα έως και τρεις φορές γρηγορότερα από την PostGIS. Αυτό οφείλεται στη δομή της Neo4j βάσης δεδομένων (γράφος), η οποία είναι σχεδιασμένη για τη βελτιστοποίηση ερωτημάτων αυτού του τύπου. Επιπλέον, όπως τονίζει ο ερευνητής στα ευρήματά του, η Neo4j δεσμεύει μεγάλο αριθμό πόρων στη μνήμη, με αποτέλεσμα από κάποιο σημείο και μετά, να επιδεινώνεται η απόδοση του συστήματος διαχείρισης βάσεων δεδομένων. Τέλος, τονίζει πως η Neo4j ενδείκνυται για χρήση σε περιπτώσεις εφαρμογής, όπου απαιτείται η ανάλυση δικτύων και τίθενται ερωτήματα συνδεσιμότητας (connectivity queries).

Από την ανασκόπηση της βιβλιογραφίας αναφορικά με τη σύγκριση συστημάτων διαχείρισης βάσεων δεδομένων SQL και συστημάτων NoSQL προέκυψαν οι παρακάτω συγκεντρωτικοί πίνακες.

Στον Πίνακας 1 παρατίθενται συγκεντρωτικά τα αποτελέσματα από τη σύγκριση των συστημάτων διαχείρισης χωρικών βάσεων δεδομένων που αναφέρθηκαν στις ενότητες 3.1 και 3.2., βάσει των ιδιοτήτων τους, των λειτουργιών που παρέχουν και των τύπων γεω – χωρικών δεδομένων που υποστηρίζουν.

Πίνακας 1. *Σύγκριση SDBMSs, βάσει των ιδιοτήτων, λειτουργιών και των τύπων γεω – χωρικών δεδομένων που υποστηρίζουν*

SDBMS	**Υποστηριζόμενοι τύποι γεωμετρίας**	**Κύριες χωρικές λειτουργίες που υποστηρίζει**	**Μέθοδοι ευρετηριοποίησης**	**Γλώσσα ερωτημάτων**	**Μορφότυποι δεδομένων**

MySQL Spatial	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων, Συλλογή Γεωμετριών	Υποστηρίζει τα πρωτόκολλα του OGC για γεωμετρικές περιπτώσεις [50] [98] (ST_Within, ST_Intersects, ST_Dwithin...)	R - tree, B - Tree	SQL	Όλους τους μορφότυπους δεδομένων που υποστηρίζει η βιβλιοθήκη GDAL
SpatiaLite	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων, Συλλογή Γεωμετριών	Υποστηρίζει τα πρωτόκολλα του OGC για γεωμετρικές περιπτώσεις [50] [98] (ST_Within, ST_Intersects, ST_Dwithin...)	R* tree	SQL	Όλους τους μορφότυπους δεδομένων που υποστηρίζει η βιβλιοθήκη GDAL
PostGIS	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων, Συλλογή Γεωμετριών	Υποστηρίζει τα πρωτόκολλα του OGC για γεωμετρικές περιπτώσεις [50] [98] (ST_Within, ST_Intersects, ST_Dwithin...)	GiST index (Generalized Search Tree), R - tree, B - Tree, BRIN (Block Ranage Index), SP - Gist (Space - Partitioned Generalized Search Tree)	SQL	Όλους τους μορφότυπους δεδομένων που υποστηρίζει η βιβλιοθήκη GDAL
Redis	Σημείο	geoadd, geodist, geohash, geopos, georadius, georadius member, geoencode, geodecode, geopathlen geometry filter	geohash	–	GeoJSON
MongoDB	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων, Συλλογή Γεωμετριών, Feature (γεωμετρικό αντικείμενο με	$geoIntersects, $geoWithin, $near, $nearSphere	2dsphere index, 2d index	–	GeoJSON, Legacy Coordinate Pairs

	περιγραφικά χαρακτηριστικά, Συλλογή Feature				
Cassandra	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων	intersects, contains, is_within, bounding box, buffer, centroid, convex hull, union, difference, intersection	geohash recursive prefix tree	Cassandra Query Language (CQL)	WKT
Neo4j	Σημείο, Γραμμή, Πολύγωνο, Συλλογή Σημείων, Συλλογή Γραμμών, Συλλογή Πολυγώνων, Συλλογή Γεωμετριών	distance(), point()-WGS84 2D - 3D, point()-Cartesian 2D-3D, Contain,Cover, Covered By, Cross, Disjoint, Intersect Window, Overlap Touch, Within, Within Distance, Area, Bbox, Boundary, Distance, Buffer, Centroid, ConvexHull, Envelope	B+ Tree, R - Tree	Cypher	WKT, Shapefile, OSM file, CSV, GeoJSON, RDF

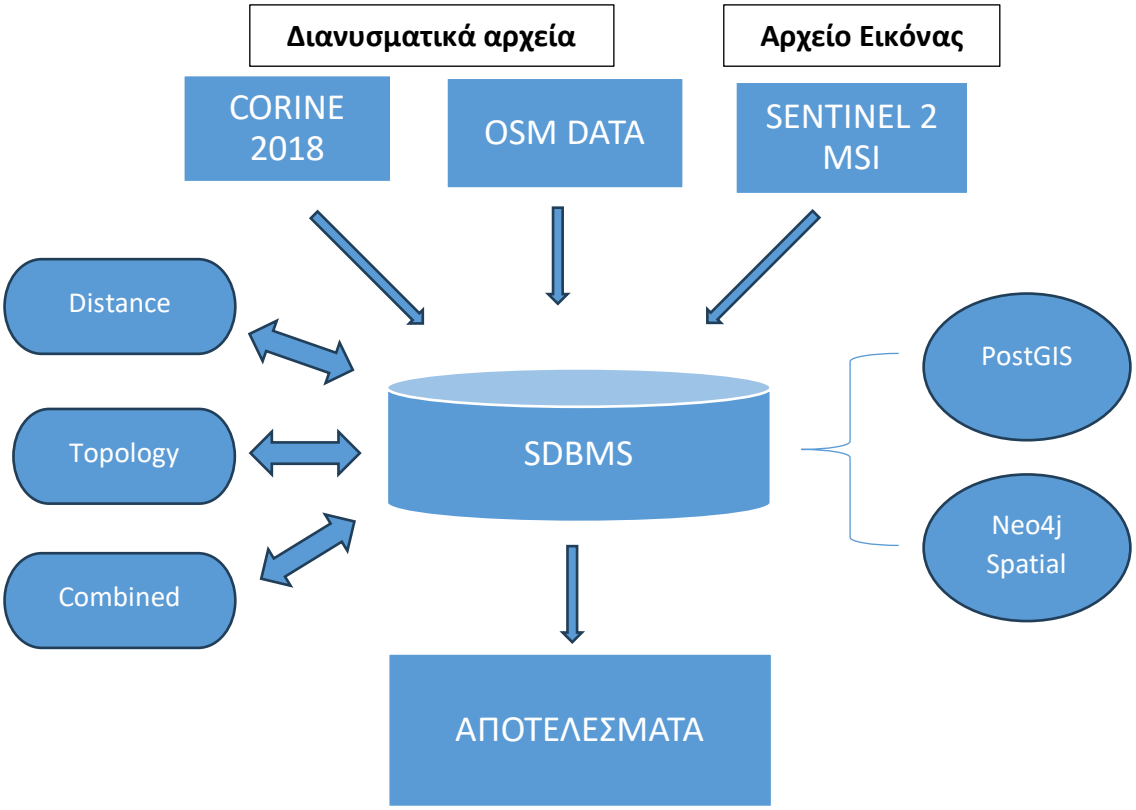
Στον *Πίνακας* 2 συγκρίνονται τα συστήματα διαχείρισης βάσεων δεδομένων, με βάση την δυνατότητα χρήσης τους για γεω – χωρικά δεδομένα, εγγενώς, με τη χρήση προσθέτου (extension) ή τη χρήση εξωτερικής βιβλιοθήκης \ εργαλείων, τα οποία επίσημα υποστηρίζονται από τον κατασκευαστή του συστήματος. Βασικό κριτήριο για τη σύγκριση αποτελεί να αναγνωρίζει κατ’ ελάχιστον έναν τύπο γεωμετρίας και να προσφέρει τουλάχιστον μία λειτουργία για αυτόν.

***Πίνακας 2**. Σύγκριση DBMSs, δυνατότητα χρήσης σε γεω - χωρικά δεδομένα*

DBMS	Εγγενώς	Επέκταση	Εξωτερική Βιβλιοθήκη \ εργαλεία
MySQL	√	√	
SQLite	√	√	
Postgres	√	√	
Redis	√	√	
MongoDB	√		
Cassandra			√
Neo4j	√	√	

5. Μεθοδολογία

Για την αξιολόγηση των SQL – NoSQL βάσεων δεδομένων αναφορικά με τη χρήση γεω-χωρικών δεδομένων πραγματοποιήθηκαν γεωχωρικές διεργασίες σε διανυσματικά και ψηφιδωτά (raster) δεδομένα. Χρησιμοποιήθηκαν η PostGIS (Postgres) και η Neo4j (με την επέκταση – spatial plugin). Όλες οι εργασίες έγιναν σε υπολογιστή Lenovo IdeaPad 5 Pro με 32GB RAM μνήμη, επεξεργαστή AMD Ryzen 9 5900HX με Radeon Graphics, 3.30 GHz, 64 – bit λειτουργικό σύστημα Windows 11 Home και σκληρό δίσκο SSD 1TB. Η μεθοδολογία που ακολουθήθηκε αποδίδεται σχηματικά στο διάγραμμα ροής που ακολουθεί.



***Διάγραμμα 1**. Διάγραμμα ροής*

5.1 Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν στη παρούσα εργασία, αποτελούν ανοιχτά δεδομένα (Open Data). Χρησιμοποιήθηκαν σημειακά δεδομένα, με τους οικισμούς της Ελλάδας, τα οποία ελήφθησαν από την εταιρία Geofabrik, η οποία διανέμει δωρεάν, τα δεδομένα από το Open Street Map (OSM) [99]. Επίσης, έγινε χρήση των διανυσματικών δεδομένων καλύψεων γης (Corine 2018) [100], του προγράμματος παρατήρησης της Γης (Copernicus) [101], της Ευρωπαϊκής Ένωσης. Και στα δύο σύνολα δεδομένων, έγινε προ - επεξεργασία των δεδομένων, στο λογισμικό ανοιχτού κώδικα Quantum GIS (QGIS) [102]. Στα σημειακά δεδομένα, εισήχθησαν τέσσερεις νέες στήλες στον πίνακα ιδιοτήτων (attribute table), στις οποίες εισήχθησαν τα ζεύγη συντεταγμένων τον σημείων σε δύο συστήματα (CGRS87 / Greek Grid – EPSG: 2100 και WGS 84 – EPSG: 4326). Το αρχείο με τους οικισμούς αποτελείται από 12120 εγγραφές, και έγινε εξαγωγή του σε αρχείο Shapefile (EPSG: 32634, UTF-8). Το σύνολο δεδομένων των καλύψεων γης (πολύγωνα), «περιορίστηκε / clipped» στα σύνορα της Ελλάδας, όπως αυτά δίνονται από την Ελληνική Στατιστική Υπηρεσία (ΕΛΣΤΑΤ),

καθώς περιέχει όλες τις καλύψεις γης για όλη την Ευρώπη. Ακολούθως, δημιουργήθηκε μία καινούργια στήλη μέσα στον πίνακα ιδιοτήτων, όπου εισήχθησαν τα αλφαριθμητικά (ονομασίες), που αντιστοιχούν στον κάθε κωδικό κάλυψης γης. Το αρχείο αποτελείται από 49131 εγγραφές και εξήχθη σε αρχείο Shapefile (EPSG: 32634, UTF – 8). Το αρχείο εικόνας (raster) που χρησιμοποιήθηκε στην παρούσα ανάλυση, ελήφθη από τον δορυφόρο Sentinel – 2A από τον αισθητήρα MSI [103]. Είναι σε σύστημα συντεταγμένων EPSG: 32634 (UTM 34N) και ελήφθη τον Ιούνιο του 2022. Η προ επεξεργασία της εικόνας έγινε στο QGIS 3.20.3 όπου έγινε η ατμοσφαιρική διόρθωση των δεδομένων και υπολογίστηκε ο δείκτης βλάστησης NDVI [104].

5.2 Κατασκευή βάσεων δεδομένων

5.2.1 PostGIS

Για την ανάλυση των δεδομένων στο SQL DBMS έγινε χρήση της Postgres (pgAdmin 4 v6.0), με την επέκταση PostGIS (3.3.2). Τα δεδομένα εισήχθησαν στη βάση δεδομένων, μέσω του προσθέτου DBManager στο QGIS 3.20.3. Τα δεδομένα εισήχθησαν σε τρεις πίνακες (tables) στη βάση δεδομένων (Corine 2018, Oikismoi, Raster). Ως μέθοδος ευρετηριοποίησης των δεδομένων, χρησιμοποιήθηκε η μέθοδος ευρετηριοποίησης GiST, που παρέχει η Postgress/PostGIS. Μετά την εισαγωγή των δεδομένων στη βάση μετρήθηκε ο χώρος που καταλαμβάνει η βάση στον δίσκο (disk space), ο οποίος είναι 362.89 MB.

5.2.2 Neo4j

Για την ανάλυση των δεδομένων στο NoSQL σύστημα διαχείρισης βάσεων δεδομένων, έγινε χρήση της Neo4j community 4.4.24 και του προσθέτου Spatial 0.28.1 – neo4j – 4.3.10. Σημειώνεται, πως δεν χρησιμοποιήθηκε η νεότερη έκδοση της Neo4J (5.11.0), καθώς δεν υποστηρίζει μέχρι στιγμής, το πρόσθετο Neo4j Spatial. Τα δεδομένα εισήχθησαν από μορφή Shapefile στη βάση δεδομένων. Η εισαγωγή, επεξεργασία και ανάλυση των δεδομένων έγινε με την Python, με τη χρήση της βιβλιοθήκης py2neo [105], της βιβλιοθήκης geopandas [106]και της βιβλιοθήκης shapely [107].

5.3 Ανάλυση δεδομένων

Για την ανάλυση των δεδομένων και τη σύγκριση των δύο συστημάτων επιλέχθηκαν ερωτήματα τα οποία συναντώνται συχνά στην ανάλυση γεω – χωρικών δεδομένων, σε εύρος επιστημονικών πεδίων (π. χ. οικολογία, γεωργία, κ.ά). Πιο συγκεκριμένα, επιλέχθηκαν:

a) Ερώτημα εγγύτητας (Proximity query) (Q1): Αναζητήθηκαν οι οικισμοί που έχουν λιγότερους από διακόσιους (200) κατοίκους και η κοντινότερη (Ευκλείδεια απόσταση) πόλη σε αυτούς, με πάνω από δέκα χιλιάδες (10000) κατοίκους. Παρακάτω, παρατίθεται ο κώδικας που χρησιμοποιήθηκε για την ανάλυση των δεδομένων στην PostGIS.

```
1. ** Δημιουργία νέου πίνακα που θα περιλαμβάνει τις αποστάσεις μεταξύ
ζευγών σημείων:**
CREATE TABLE closest_points AS
2. ** Επιλογή σημείων (Οικισμών) με βάση το κριτήριο του πληθυσμού
(0<population<200):**
WITH selected_points AS (
    SELECT
        osm_id,
name AS selected_point_name,
        geom AS geom,
        population
    FROM
        oikismoi_2100
    WHERE
        population < 200 and population > 0
),
3.** Επιλογή σημείων (Οικισμών)με βάση το κριτήριο του πληθυσμού
(population >= 10000):**
points_with_greater_population AS (
    SELECT
        osm_id,
        name AS name,
        geom AS geom,
        population
    FROM
        oikismoi_2100
    WHERE
        population >= 10000
)
4. **Συνδυάζει τα δεδομένα από τα δύο προηγούμενα ερωτήματα (2&3) και
πραγματοποιεί τον υπολογισμό για την εύρεση του πλησιέστερου σημείου για
κάθε επιλεγμένο σημείο:**
SELECT
    sp.osm_id AS selected_point_osm_id,
    sp.selected_point_name AS selected_point_name,
    sp.population AS selected_point_population,
    cp.osm_id AS closest_point_osm_id,
    cp.name AS closest_point_name,
    cp.population AS closest_point_population,
    ST_Distance(sp.geom, cp.geom) AS distance,
    sp.geom AS selected_point_geom,
```

```
    cp.geom AS closest_point_geom
FROM
    selected_points sp
CROSS JOIN LATERAL (
    SELECT
        osm_id,
        name AS name,
        geom,
        population
    FROM
        points_with_greater_population
    ORDER BY
        sp.geom <-> geom
    LIMIT 1
) cp;
```

Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήθηκε στη Python, για την ανάλυση των δεδομένων στη Neo4j βάση δεδομένων.

```
import time
start_time = time.time()
from py2neo import Graph
import csv
# Σύνδεση με τη Neo4j
uri = "http://localhost:7474"
username = "username"
password = "password"

graph = Graph(uri, auth=(username, password))
# Εκτέλεση ερωτήματος - πληθυσμός <200 κατοίκων
query_lt_200 = """
MATCH (o1:oikismoi)
WHERE o1.population > 0 AND o1.population < 200
RETURN o1.x AS x, o1.y AS y, o1.name AS name
"""
# Εκτέλεση ερωτήματος - πληθυσμός >=10000 κατοίκων
query_gte_10000 = """
MATCH (o2:oikismoi)
WHERE o2.population >= 10000
RETURN o2.x AS x, o2.y AS y, o2.name AS name
"""
#Εκτέλεση υπολογισμού των αποστάσεων και εύρεση της μικρότερης απόστασης
def execute_query(query):
    result = graph.run(query)
    return [(record["x"], record["y"], record["name"]) for record in
result]

def calculate_distance(coord1, coord2):
    x1, y1, _ = coord1
    x2, y2, _ = coord2
    return ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5

lt_200_coords = execute_query(query_lt_200)
gte_10000_coords = execute_query(query_gte_10000)

results = []

for coord_lt_200 in lt_200_coords:
```

```
    closest_coord = None
    closest_distance = float('inf')
    closest_name = ""
    closest_name_gte_10000 = ""

    for coord_gte_10000 in gte_10000_coords:
        distance = calculate_distance(coord_lt_200, coord_gte_10000)
        if distance < closest_distance:
            closest_distance = distance
            closest_coord = coord_gte_10000
            closest_name_gte_10000 = coord_gte_10000[2]  # Index 2 is the
"name" field

    if closest_coord:
        results.append({
            "coords_lt_200": coord_lt_200,
            "coords_gte_10000": closest_coord,
            "distance": closest_distance,
            "name_lt_200": coord_lt_200[2],  # Index 2 is the "name" field
for lt_200_coords
            "name_gte_10000": closest_name_gte_10000
        })

# Αποθήκευση του αποτελέσματος σε αρχείο
output_csv = "path_file/output_results_with_names1.csv"
with open(output_csv, 'w', newline='') as csvfile:
    fieldnames = ["coords_lt_50", "coords_gte_10000", "distance",
"name_lt_200", "name_gte_10000"]
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
    for result in results:
        writer.writerow(result)

print("CSV file saved:", output_csv)

end_time = time.time()

# υπολογισμός του χρόνου εκτέλεσης
elapsed_time = end_time - start_time
print(f&quot;Elapsed time: {elapsed_time:.2f} seconds&quot;)
```

b) Ερώτημα τοπολογίας (Topology query) (Q2): Επιλέχθηκε μία περιοχή μελέτης και ζητήθηκε από τη βάση να επιστρέψει όλους τους οικισμούς που εμπεριέχονται στην περιοχή (within). Έπειτα, έγινε ερώτημα στη βάση, για τη μεγαλύτερη κάλυψη γης στην περιοχή μελέτης, η οποία συμπίπτει με οικισμό (intersect), καθώς και να εκτελέσει χωρική αποκοπή (clip) την δορυφορική εικόνα και να επιστρέψει μόνο το κομμάτι της εικόνας που βρίσκεται μέσα στην προαναφερθείσα κάλυψη γης. Ακολουθεί ο κώδικας που

χρησιμοποιήθηκε για την ανάλυση των δεδομένων στην PostGIS βάση δεδομένων.

1. **Ανάκτηση της γεωγραφικής έκτασης (extent) της εικόνας:**
```
SELECT ST_Envelope(rast) FROM raster_table;
```

2. **Δημιουργία γεωμετρίας (spatial object) που αντιπροσωπεύει (bounding box) των δεδομένων raster που είναι αποθηκευμένα στον πίνακα:**
```sql
SELECT ST_MakeEnvelope(ST_XMin(envelope), ST_YMin(envelope), ST_XMax(envelope), ST_YMax(envelope), <SRID>) AS raster_geom
FROM (
  SELECT ST_Envelope(rast) AS envelope
  FROM raster_table
) AS subquery;
```

3. **Επιλογή στοιχείων (σημείων) του πίνακα oikismoi που βρισκονται εντός (within)του bounding box του raster:**
```sql
CREATE TABLE points_within_raster AS
SELECT o.*
FROM oikismoi_34n o
WHERE ST_Within(o.geom, (
   SELECT ST_Envelope(rast) FROM raster_table
));
```

4. ** Χωρική ένωση - (Spatial Join) των σημείων με τις καλύψεις γης (landuse):
```sql
SELECT p.*, lu.landuse
FROM filtered_points p
JOIN corine_34n lu ON ST_Intersects(p.geom, lu.geom);
```
5. **Επιλογή του σημείου με την μεγαλύτερη έκταση κάλυψης γης, όπως προέκυψε από το προηγούμενο ερώτημα:**
```sql
SELECT p.*, ST_Area(p.geom) AS polygon_area
FROM filtered_points p
ORDER BY polygon_area DESC
LIMIT 1;
```

6. ** Επιλογή του πολυγώνου κάλυψης γης (corine) με βάση το σημείο (από το προηγούμενο ερώτημα):
```sql
SELECT nc.*
FROM corine_34n nc
JOIN (
   SELECT p.*
   FROM filtered_points p
   ORDER BY ST_Area(p.geom) DESC
   LIMIT 1
) AS largest_point
ON ST_Intersects(nc.geom, largest_point.geom);
```

```
7. ** Δημιουργία νέου πίνακα με το επιλεγμένο πολύγωνο, από το προηγούμενο ερώτημα:**
   ```sql
   CREATE TABLE clipped_polygon AS
   SELECT nc.*, largest_point.population AS point_population,
largest_point.name AS point_name
   FROM corine_34n nc
   JOIN (
      SELECT p.population, p.name, p.geom
      FROM filtered_points p
      ORDER BY ST_Area(p.geom) DESC
      LIMIT 1
   ) AS largest_point
   ON ST_Intersects(nc.geom, largest_point.geom);
   ```

8. ** Χωρική αποκοπή (Clipping)του raster και τη δημιουργία νέου raster στον πίνκα:**
   ```sql
   INSERT INTO new_raster (rast)
   SELECT ST_Clip(r.rast, c.geom) AS clipped_rast
   FROM raster_table r, clipped_polygon c;
   ```
```

c) Συνδυαστικό ερώτημα εγγύτητας και τοπολογίας (Proximity – Topology query) (Q3): Αναζητήθηκαν οι οικισμοί με λιγότερους από 50 κατοίκους. Στους οικισμούς αυτούς, δημιουργήθηκε περιμετρικά μία ζώνη ακτίνας 200μ (buffer). Ακολούθως, αναζητήθηκαν όλες οι καλύψεις γης οι οποίες συμπίπτουν (intersect), με τη ζώνη που δημιουργήθηκε νωρίτερα. Τέλος, έπρεπε να βρεθεί η κυρίαρχη (σε έκταση), χρήση γης για κάθε έναν οικισμό, που συμπίπτει με τη ζώνη των 200 μέτρων. Ακολουθεί ο κώδικας που χρησιμοποιήθηκε για την ανάλυση των δεδομένων στην PostGIS βάση δεδομένων.

```
1. **Μετασχηματισμός συστήματος συντεταγμένων από WGS84 σε Greek Grid/2100 του επιπέδου (layer)Oikismoi και επιλογή στοιχείων του πίνακα με βάση τον πληθυσμό (population):**
```sql
CREATE TABLE reprojected_points AS
SELECT
    ST_Transform(geom, 2100) AS geom,
    osm_id,
    code,
    fclass,
    population,
    name
FROM oikismoi
WHERE population <= 50 AND population > 0;
```
```

```

2. **Δημιουργία ζωνών επιρροής - buffer στα επιλεγμένα στοιχεία του πίνακα:**
```sql
CREATE TABLE buffered_points AS
SELECT
    ST_Buffer(geom, 200) AS geom,
    osm_id,
    code,
    fclass,
    population,
    name
FROM reprojected_points;
```

3. **Μετασχηματισμός συστήματος συντεταγμένων (Greek Grid/2100-->WGS84):**
```sql
CREATE TABLE buffered_points_wgs841 AS
SELECT
    ST_Transform(geom, 4326) AS geom,
    osm_id,
    code,
    fclass,
    population,
    name
FROM buffered_points;
```

4. **Τομή /intersection των 2 πινάκων Corine & Buffer και ένωση (Join) της περιγραφικής πληροφορίας της κάλυψης γης:**
```sql
CREATE TABLE buffered_points_with_land_use AS
SELECT
    bpw.geom,
    bpw.osm_id,
    bpw.code,
    bpw.fclass,
    bpw.population,
    bpw.name,
    nc.landuse
FROM
    buffered_points_wgs841 bpw
LEFT JOIN
    new_corine nc
ON
    ST_Intersects(bpw.geom, nc.geom);
```

5. **Προσθήκη πεδίου/ στήλης (Biggest Land Use) στον πίνακα :**
```sql
ALTER TABLE buffered_points_with_land_use
ADD COLUMN biggest_land_use text;
```

6. **Ενημέρωση του πίνακα με την κατηγορία της μεγαλύτερης σε έκταση κάλυψη γης:**
```sql
UPDATE buffered_points_with_land_use AS bplu
SET biggest_land_use = (
    SELECT landuse
```
```

```
    FROM (
        SELECT landuse, ST_Area(geom) AS area,
               ROW_NUMBER() OVER (PARTITION BY bplu.osm_id ORDER BY
ST_Area(geom) DESC) AS rn
        FROM new_corine nc
        WHERE ST_Intersects(bplu.geom, nc.geom)
    ) ranked_land_use
    WHERE ranked_land_use.rn = 1
);
```

Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήθηκε στη Python, για την εισαγωγή των δεδομένων στη Neo4j βάση δεδομένων

```
# Ενσωμάτωση των γεωχωρικών δεδομένων μορφής Shapefile στη Neo4j
from py2neo import Graph, Node, Relationship
import geopandas as gpd
from shapely.geometry import shape
# Σύνδεση με τη Neo4j
uri = "http://localhost:7474"
username = "username"
password = "password"

graph = Graph(uri, auth=(username, password))

# Εισαγωγή του Shapefile "οικισμοί"
new_shapefile = "path_file/oikismoi_2100.shp"
gdf_new = gpd.read_file(new_shapefile)

# Ορισμός ονομασίας για το νέο layer
new_layer_label = "oikismoi"

# "Διαβασμα" του αρχείου και δημιουργεία σημείων για το καινούργιο layer
for index, row in gdf_new.iterrows():
    geometry = row["geometry"]
    attributes = row.drop("geometry")  # Extract attributes other than
geometry

    # Μετατροπή του αρχείου σε WKT
    geometry_wkt = geometry.wkt

    # Δημιουργία κόμβων με ιδιότητες
    neo4j_node = Node(new_layer_label, geometry=geometry_wkt, **attributes)
    graph.create(neo4j_node)


print("New shapefile data added to Neo4j as a separate layer.")

# Εισαγωγή του Shapefile "corine", καλύψεων γης
new_shapefile2 = "path_file/corine.shp"
gdf_new2 = gpd.read_file(new_shapefile2)

new_layer_label2 = "corine"

for index, row in gdf_new2.iterrows():
    geometry = row["geometry"]
    attributes = row.drop("geometry")

    geometry_wkt = geometry.wkt
```

```
    neo4j_node = Node(new_layer_label2, geometry=geometry_wkt,
**attributes)
    graph.create(neo4j_node)


print("Second shapefile data added to Neo4j as a separate layer.")
```

Ακολουθεί ο κώδικας που χρησιμοποιήθηκε για την ανάλυση των δεδομένων και την εξαγωγή των αποτελεσμάτων.

```
# Εκτέλεσει γεωχωρικών διεργασιών
from pyproj import Transformer
from py2neo import Graph
from shapely.geometry import Point
from shapely.ops import transform
from functools import partial
import pyproj
import geopandas as gpd
from shapely.wkt import loads
# καταγραφή του χρόνου εκτέλεσης των διαδικασιών
import time
start_time = time.time()
## Ανάκτηση των κόμβων από τη Neo4j / ερώτημα με βάση τον πληθυσμό των
οικισμών
node_query = """
MATCH (n:oikismoi)
WHERE n.population > 0 AND n.population <= 50
RETURN n.y AS latitude, n.x AS longitude
"""
node_records = graph.run(node_query).data()
# Δημιουργία ζωνών επιρροής (buffer) στα επιλεγμένα σημεία του προηγούμενου
ερωτήματος
buffer_distance_meters = 200

def create_buffer(x, y, buffer_distance_meters):
    point = Point(x, y)
    buffer_point = point.buffer(buffer_distance_meters)
    return buffer_point

buffer_geoms = []
for record in node_records:
    x = record["longitude"]
    y = record["latitude"]
    buffer = create_buffer(x, y, buffer_distance_meters)
    buffer_geoms.append(buffer)

gdf = gpd.GeoDataFrame(geometry=buffer_geoms, crs="EPSG:2100")
#Ανάκτηση των κόμβων από τη Neo4j /για τις καλύψεις γης (corine)
corine_query = """
MATCH (c:corine)
RETURN c.geometry AS geometry,c.landuse AS landuse
"""

corine_records = graph.run(corine_query).data()
# Επεξεργασία γεωμετρίας και ιδιοτήτων (attributes) "corine"
corine_geoms = []
corine_attributes = []
```

```
for record in corine_records:
    wkt_geometry = record["geometry"]
    shapely_geometry = loads(wkt_geometry)
    corine_geoms.append(shapely_geometry)

    attributes = {
        "landuse": record["landuse"]
        # Add more attributes as needed
    }
    corine_attributes.append(attributes)

corine_gdf = gpd.GeoDataFrame(data=corine_attributes,
geometry=corine_geoms, crs="EPSG:2100")
#προσθήκη της ονομασίας του οικισμού στα buffers
for record in oikismoi_records:
    x = record["longitude"]
    y = record["latitude"]
    oikismoi_name = record["name"]
point = Point(x, y)
    for index, buffer_row in gdf.iterrows():
        if point.within(buffer_row["geometry"]):
            gdf.at[index, "oikismoi_name"] = oikismoi_name
# Τομή (Intersection) μεταξύ του corine και των buffer των οικισμών &
εξαγωγή του αποτελέσματος σε νεό shapefile
intersection_results = gpd.overlay(corine_gdf, gdf, how='intersection')
output_intersection_shapefile = "path_file/intersection_neo8.shp"
intersection_results.to_file(output_intersection_shapefile, encoding='utf-
8')
#Εισαγωγή του intersection shapefile (που προέκυψε από την προηγούμενη
διαδικασία στη Neo4j)
new_shapefile = "path_file/intersection_neo8.shp"
gdf_new = gpd.read_file(new_shapefile)
new_layer_label = "intersection"
for index, row in gdf_new.iterrows():
    geometry = row["geometry"]
    attributes = row.drop("geometry")
    geometry_wkt = geometry.wkt  # Or use geometry.to_json() for GeoJSON
    neo4j_node = Node(new_layer_label, geometry=geometry_wkt, **attributes)
    graph.create(neo4j_node)

print("New shapefile data added to Neo4j as a separate layer.")
# δημιουργία σχέσεων μεταξύ των κόμβων που δημιουργήθηκαν από το
προηγούμενη διαδικασία (intersection) με βάση την ονομασία των οικισμών.
#(σύνδεση διαφορετικών καλύψεων γης που αντιστοιχούν στον ίδιο οικισμό).
query = """
MATCH (n1:intersection), (n2:intersection)
WHERE n1 <> n2 AND n1.oikismoi_n = n2.oikismoi_n
CREATE (n1)-[:SAME_OIKISMOI]->(n2)
"""

graph.run(query)

print("SAME_OIKISMOI relationships created in Neo4j.")
# Εύρεση κόμβων με τη μεγαλύτερη, με βάση την έκταση, κάλυψη γης
query = """
MATCH (n:intersection)
WITH n.oikismoi_n AS oikismoi_n, COLLECT(n) AS nodes
ORDER BY oikismoi_n
WITH oikismoi_n, nodes[0] AS largestNode
MATCH (n)-[:RELATED_TO]->(largestNode)
SET n.blanduse = largestNode.landuse
```

```
"""
graph.run(query)

print("Updated 'blanduse' property for related nodes.")
largest_area_query = """
MATCH (i:intersection)
WITH i.oikismoi_n AS oikismoi_n, COLLECT(i) AS nodes
ORDER BY oikismoi_n
WITH oikismoi_n, nodes[-1] AS largestNode
SET largestNode.Blanduse = largestNode.landuse
"""
propagate_query = """
MATCH (i:intersection)
WITH i.oikismoi_n AS oikismoi_n, COLLECT(i) AS nodes
ORDER BY oikismoi_n
WITH oikismoi_n, nodes[-1] AS largestNode
MATCH (n:intersection)
WHERE n.oikismoi_n = oikismoi_n
SET n.Blanduse = largestNode.Blanduse
"""

graph.run(largest_area_query)
graph.run(propagate_query)

print("Updated Blanduse values in Neo4j 'intersection' layer.")
#άντληση δεδομένων από τη Neo4j
query = """
MATCH (i:intersection)
RETURN i.oikismoi_n AS oikismoi_n,i.landuse AS landuse, i.Blanduse AS
Blanduse, i.geometry AS geometry
"""
result = graph.run(query).data()
#Μετατροπή γεωμετρίας (geometry data) σε αντικείμενα (geometry objects)&
εξαγωγή του αποτελέσματος σε shapefile
    for item in result:
    item["geometry"] = loads(item["geometry"])
    gdf = gpd.GeoDataFrame(result, geometry="geometry")
    output_shapefile =  "path_file/intersection_output.shp"
gdf.to_file(output_shapefile, encoding ="UTF-8")

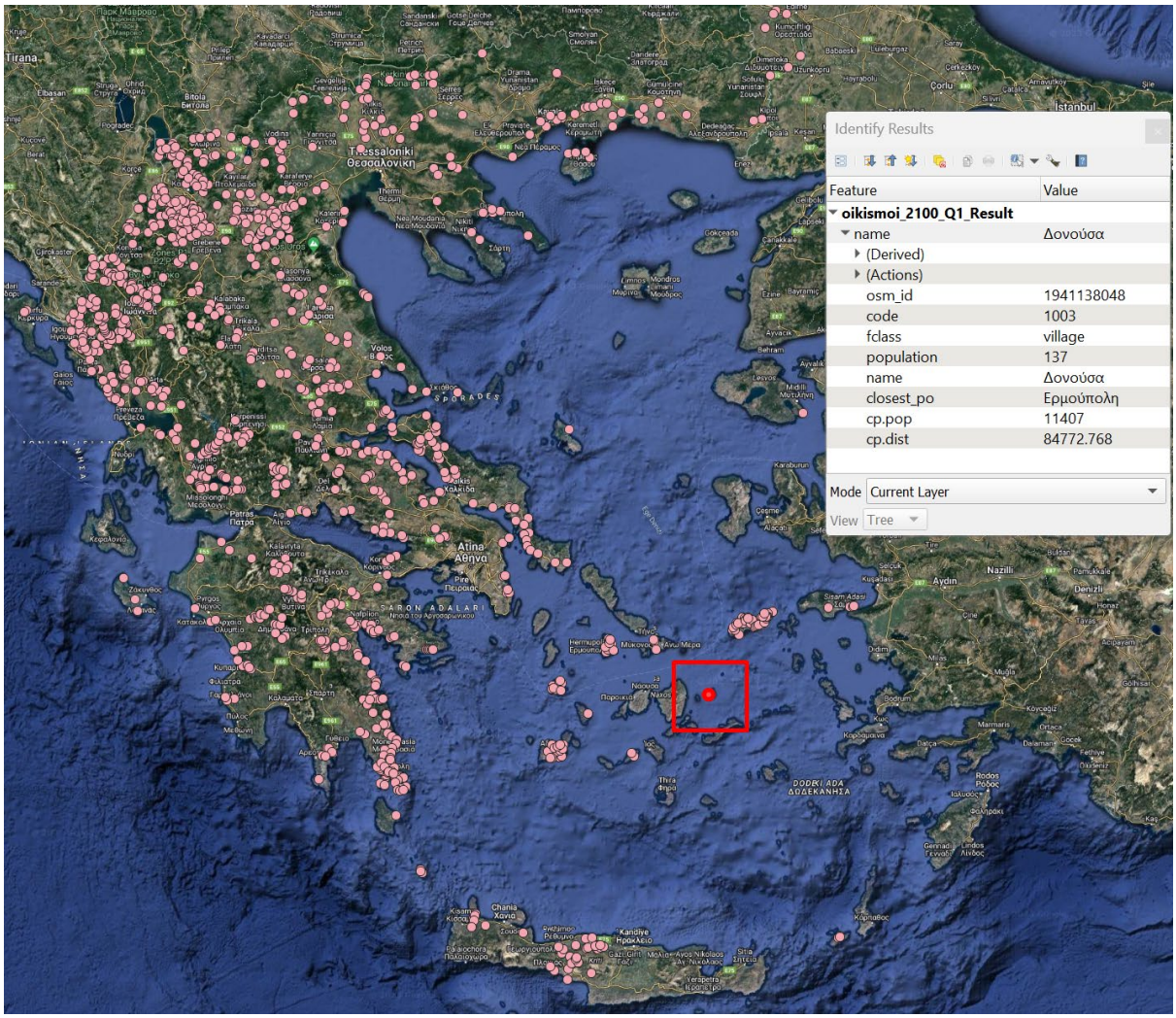
print("Shapefile exported successfully.")
import time
end_time = time.time()
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")
```

Σκοπός ήταν να γίνει μέτρηση του χρόνου που χρειάζεται το κάθε ερώτημα για να επιστρέψει τα ζητούμενα αποτελέσματα, καθώς και εάν υπάρχει χρονική διαφορά, με τη χρήση μεθόδου ευρετηριοποίησης, στην επιστροφή των αποτελεσμάτων. Για την ακριβότερη μέτρηση των χρόνων, όλα τα ερωτήματα εκτελέστηκαν από δέκα (10) φορές και το αποτέλεσμα είναι ο μέσος όρος των χρόνων αυτών.

6. Αποτελέσματα και συζήτηση

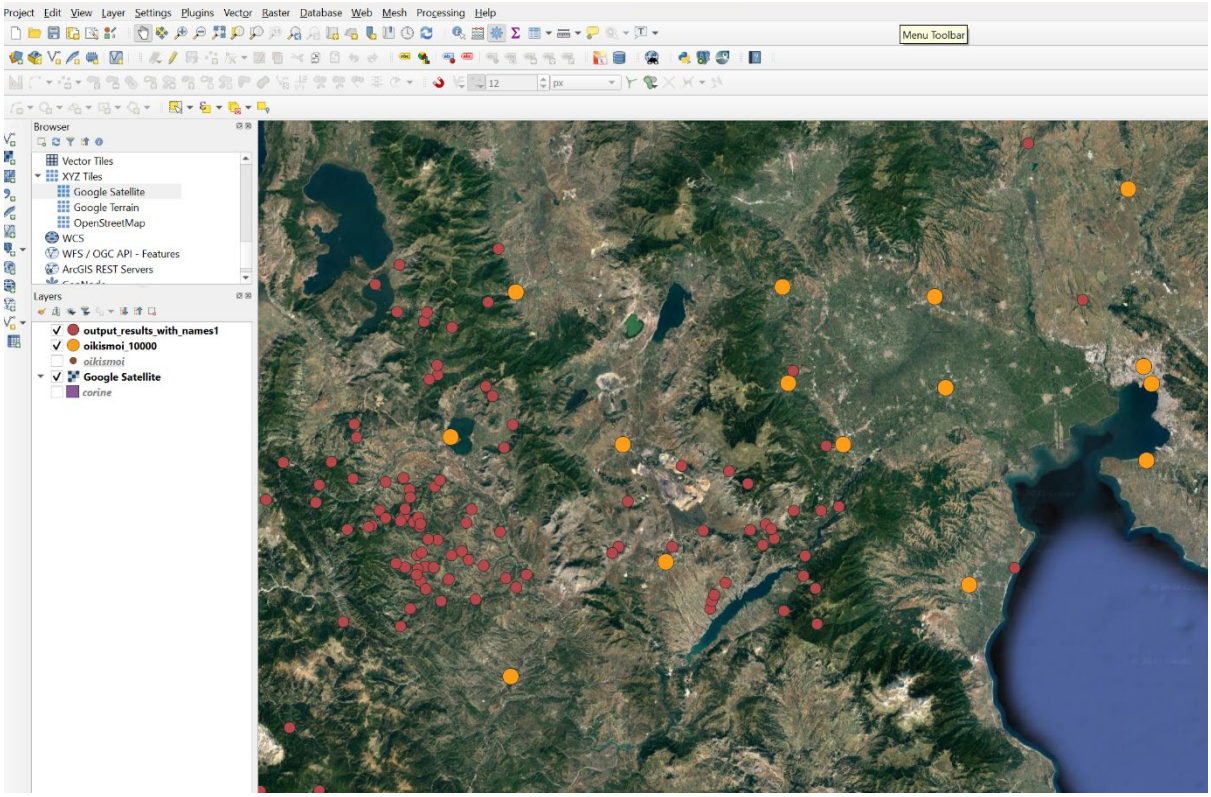
6.1 Ερώτημα Q1

Για το πρώτο ερώτημα, από την ανάλυση των δεδομένων στην PostGIS, προέκυψαν 1084 αποτελέσματα, που αναφέρονται στους οικισμούς με λιγότερους από διακόσιους κατοίκους και τη κοντινότερη πόλη, με πληθυσμό άνω των δέκα χιλιάδων κατοίκων, σε αυτούς. Ο υπολογισμός χωρίς μέθοδο ευρετηριοποίησης χρειάστηκε 971 χιλιοστά του δευτερολέπτου για να επιστρέψει αποτέλεσμα, ενώ με τη χρήση μεθόδου ευρετηριοποίησης χρειάστηκε 940 χιλιοστά του δευτερολέπτου. Ανάμεσα στις δύο μεθόδους, παρατηρείται μείωση του χρόνου ανάλυσης των δεδομένων με τη χρήση ευρετηριοποίησης. Στην **Εικόνα 1**. που ακολουθεί, φαίνεται το αποτέλεσμα όπως οπτικοποιήθηκε στο QGIS.



Εικόνα 1. *Αποτέλεσμα PostGIS για το Q1, οπτικοποίηση των οικισμών στο QGIS*

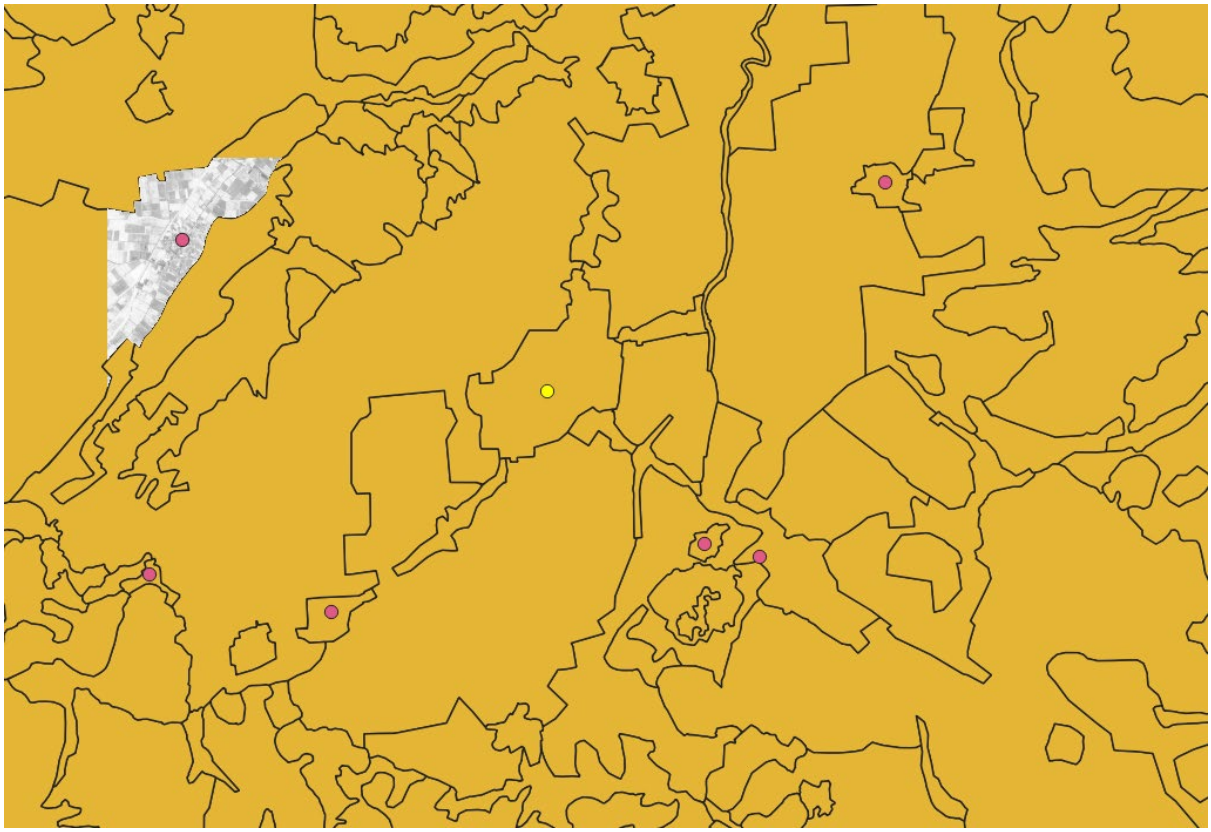
Για το ερώτημα (Q1), η βάση δεδομένων Neo4j, για την ανάλυση των δεδομένων και την εξαγωγή των αποτελεσμάτων χρειάστηκε δύο δευτερόλεπτα (2.09 sec). Παρακάτω φαίνονται τα αποτελέσματα από το ερώτημα στη βάση δεδομένων Neo4j, όπως οπτικοποιήθηκαν στο QGIS.



***Εικόνα 2**. Αποτέλεσμα Neo4j για το Q1, οπτικοποίηση των οικισμών στο QGIS*

6.2 Ερώτημα Q2

Για το δεύτερο ερώτημα, όπου ζητήθηκαν οι οικισμοί που βρίσκονται μέσα στην περιοχή που ορίστηκε ως περιοχή μελέτης, η μεγαλύτερη σε έκταση χρήση γης σε αυτήν και τη χωρική αποκοπή της εικόνας στην έκταση της μεγαλύτερης κάλυψης. Παρακάτω φαίνεται το αποτέλεσμα του ερωτήματος, όπως οπτικοποιήθηκε μέσα στο QGIS. Από το αποτέλεσμα φαίνεται πως το ερώτημα «έτρεξε» σωστά στη βάση, καθώς διακρίνεται το περίγραμμα της μεγαλύτερης σε έκταση χρήσης γης, η οποία συμπίπτει με οικισμό, καθώς και τα δεδομένα της εικόνας, η οποία κόπηκε στα όρια της συγκεκριμένης κάλυψης γης, όπως φαίνεται στην **Εικόνα 3** που ακολουθεί.



***Εικόνα 3**. Αποτέλεσμα Q2 από την PostGIS, οπτικοποίηση στο QGIS*

Ο χρόνος που χρειάστηκε η βάση δεδομένων PostGIS για την επεξεργασία του ερωτήματος και την επιστροφή των αποτελεσμάτων είναι 689 χιλιοστά του δευτερολέπτου, χωρίς εφαρμογή μεθόδου ευρετηριοποίησης ενώ, με τη χρήση μεθόδου ευρετηριοποίησης χρειάστηκε 594 χιλιοστά του δευτερολέπτου.

Το συγκεκριμένο ερώτημα, δεν εκτελέστηκε στη βάση δεδομένων Neo4j, καθώς δεν υποστηρίζει εγγενώς ή μέσω της επέκτασης Neo4j Spatial την ανάλυση και επεξεργασία χωρικών δεδομένων εικόνων.

Στον τομέα της γεωπληροφορικής, τα χωρικά δεδομένα εικόνας αποτελούν σημαντική πηγή πληροφοριών και αναπόσπαστο κομμάτι της χωρικής ανάλυσης, σε πλήθος εφαρμογών. Η αδυναμία επεξεργασίας των δεδομένων αυτών, αποτελεί σημαντικό μειονέκτημα για τους χρήστες των συστημάτων διαχείρισης χωρικών βάσεων δεδομένων και αποθαρρυντικό παράγοντα, όταν βρίσκονται στη διαδικασία επιλογής ενός συστήματος διαχείρισης χωρικών βάσεων δεδομένων για την εκάστοτε περίπτωση εφαρμογής.

6.3 Ερώτημα Q3

Για το τρίτο ερώτημα, όπου αναζητήθηκαν οι οικισμοί με λιγότερους από πενήντα κατοίκους, δημιουργήθηκε ζώνη ακτίνας 200μ γύρο από τον κάθε οικισμό και αφού βρέθηκαν οι καλύψεις γης που συμπίπτουν με τις ζώνες, ζητήθηκε ο προσδιορισμός της κυρίαρχης, σε έκταση, κάλυψης γης γύρο από τον κάθε οικισμό. Η βάση δεδομένων PostGIS χρειάστηκε 2 λεπτά και 16 δευτερόλεπτα για την ανάλυση των δεδομένων και την επιστροφή των αποτελεσμάτων χωρίς να έχει προηγηθεί ευρετηριοποίηση των δεδομένων. Ενώ με την εφαρμογή μεθόδου ευρετηριοποίησης των δεδομένων, η διαδικασία της ανάλυσης και η επιστροφή των αποτελεσμάτων επιταχύνθηκε κατά 50% (1 λεπτό και 6 δευτερόλεπτα). Εδώ παρατηρείται η μεγαλύτερη βελτίωση στον χρόνο που χρειάζεται η βάση για την ανάλυση των δεδομένων, μετά την ευρετηριοποίηση των δεδομένων, καθώς ο χρόνος μειώθηκε στο μισό.

Για το ερώτημα (Q3), η βάση δεδομένων Neo4j, για την ανάλυση των δεδομένων και την εξαγωγή των αποτελεσμάτων χρειάστηκε 33 δευτερόλεπτα και 328 χιλιοστά του δευτερολέπτου (33.328sec) .

Στο συγκεκριμένο ερώτημα γίνεται ιδιαίτερα αντιληπτή η διαφορά ανάμεσα στο χρόνο που χρειάζονται οι δύο βάσεις δεδομένων, για την ανάλυση των δεδομένων και την επιστροφή των αποτελεσμάτων στον χρήστη. Η Neo4j, εκτέλεσε την ανάλυση και επέστρεψε τα αποτελέσματα πάνω τέσσερεις φορές γρηγορότερα από την PostGIS για την ίδια διαδικασία. Η ταχύτητα στην ανάλυση των δεδομένων, αποτελεί βασικό παράγοντα στη γεωπληροφορική, καθώς για πολλές εφαρμογές (π.χ. διαχείριση καταστροφών) ο χρόνος αποτελεί παράγοντα επιτυχίας και οποιαδήποτε καθυστέρηση μπορεί να οδηγήσει σε «αστοχία» ολόκληρου του συστήματος.

7. Συμπεράσματα

Η παρούσα διατριβή είχε ως σκοπό τη συγκριτική αξιολόγηση μεταξύ των SQL και NoSQL συστημάτων διαχείρισης χωρικών βάσεων δεδομένων. Από τη βιβλιογραφική ανασκόπηση προέκυψε ότι τα SQL συστήματα διαχείρισης χωρικών βάσεων δεδομένων, χρησιμοποιούνται ευρέως, εδώ και πολλά χρόνια στον τομέα της γεω – πληροφορικής. Για τον λόγο αυτό, διαθέτουν μεγάλο πλήθος λειτουργιών και διαδικασιών, διευκολύνοντας τη

διενέργεια χωρικών αναλύσεων σε αντίθεση με τα NoSQL συστήματα διαχείρισης. Το «πρόβλημα» στα συστήματα SQL έγκειται στη δυσκολία οριζόντιας επέκτασής τους, σε κατανεμημένα συστήματα, για την υποστήριξη μεγάλων όγκων γεω – χωρικών δεδομένων, διότι περιορίζονται από το υλισμικό (hardware), πάνω στο οποίο είναι ανεπτυγμένα. Καθώς ήδη βρισκόμαστε στην εποχή των «μεγάλων δεδομένων», η επιστημονική κοινότητα στρέφεται στη χρήση NoSQL συστημάτων, τα οποία είναι σχεδιασμένα για εύκολη οριζόντια επέκταση, αποθήκευση πολύ μεγάλου όγκου δεδομένων και αυξημένη ταχύτητα επεξεργασίας των δεδομένων. Υπολείπονται σημαντικά όμως, σε λειτουργίες και διεργασίες χωρικής ανάλυσης, καθώς τα συστήματα NoSQL που διαθέτουν εγγενώς ή μέσω προσθέτου τέτοιες λειτουργίες είναι λίγα και οι λειτουργίες αυτές περιορίζονται σε απλές διεργασίες, που δεν ικανοποιούν τις ανάγκες σύνθετων χωρικών αναλύσεων. Επιπλέον, τα γεω – χωρικά δεδομένα είναι πολύπλοκα στη δομή τους και παρέχονται σε πλήθος διαφορετικών μορφότυπων (format), πολλοί από τους οποίους, μέχρι στιγμής, δεν υποστηρίζονται από τα NoSQL συστήματα. Η επιστημονική κοινότητα, συνεχώς αναζητεί και προτείνει νέους τρόπους για την αποθήκευση, ευρετηριοποίηση, ανάλυση και οπτικοποίηση των σύνθετων χωρικών δεδομένων στα NoSQL συστήματα διαχείρισης βάσεων δεδομένων. Πολλές φορές όμως, οι προτεινόμενες μέθοδοι απαιτούν πολύπλοκες διεργασίες, γνώση σε βάθος των διαφορετικών δομών των γεωχωρικών δεδομένων καθώς και γνώσεις της δομής αποθήκευσης και των μεθόδων ευρετηριοποίησης που χρησιμοποιεί το κάθε σύστημα, ώστε να επιτύχει ο χρήστης τα επιθυμητά αποτελέσματα. Σε άλλη περίπτωση, μπορεί να οδηγήσει το σύστημα να γίνει πολύ αργό ή να καταρρεύσει, υποβαθμίζοντας με αυτόν τον τρόπο τα πλεονεκτήματα των συστημάτων NoSQL.

Από την εφαρμογή των ερωτημάτων που πραγματοποιήθηκαν τόσο σε σύστημα SQL (PostGIS) όσο και σε NoSQL (Neo4j) προέκυψε ότι η Neo4j, είναι γρηγορότερη στην ανάλυση των δεδομένων και την επιστροφή των αποτελεσμάτων, στο συνδυαστικό ερώτημα (Q3: Εγγύτητα και Τοπολογία), όπου αυξήθηκαν και τα δεδομένα προς ανάλυση, η βάση δεδομένων Neo4j, πραγματοποίησε την ανάλυση των δεδομένων και επέστρεψε τα αποτελέσματα δύο φορές γρηγορότερα από την PostGIS. Αντίθετα στο δεύτερο ερώτημα (Q2: Τοπολογία με διανυσματικά δεδομένα και χωρικά δεδομένα εικόνων), δεν κατέστη δυνατή η ανάλυση των δεδομένων στη βάση Neo4j, καθότι δεν υποστηρίζει την ανάλυση χωρικών δεδομένων εικόνας (Raster). Ακόμη στο ερώτημα Q1, η Neo4j χρειάστηκε δύο

φορές περισσότερο χρόνο από την PostGIS, για τη ανάλυση των δεδομένων και την επιστροφή των αποτελεσμάτων. Τα αποτελέσματα από την εφαρμογή του ερωτήματος Q3 που θέσαμε στα δύο συστήματα (PostGIS και Neo4j), συμφωνούν με τη βιβλιογραφία. Αντίθετα τα αποτελέσματα του ερωτήματος Q1, έρχονται σε αντίθεση με τη βιβλιογραφία.

Επιπλέον, πρέπει να επισημανθεί πως δεν υπάρχει κανένα σύστημα το οποίο είναι ιδανικό για όλες τις περιπτώσεις εφαρμογής. Διαφορετικά συστήματα, SQL και NoSQL, προσφέρουν διαφορετικές δυνατότητες που τα καθιστούν, προτιμότερα για διαφορετικές εφαρμογές. Για παράδειγμα, τα NoSQL Graph συστήματα διαχείρισης χωρικών βάσεων δεδομένα είναι αυτά που προτιμώνται για την ανάλυση δικτύων, λόγο της δομής τους, που τα καθιστά πολύ γρηγορά και μπορούν να αποθηκεύσουν και να αναπαραστήσουν ιδανικά ένα δίκτυο. Ενώ για εφαρμογές οι οποίες απαιτούνε σύνθετες χωρικές αναλύσεις, επεξεργασία και αποθήκευση μεγάλου όγκου δεδομένων, καθώς και αυξημένη ταχύτητα ανάλυσης και αποθήκευσης, συνίσταται ο συνδυασμός SQL και NoSQL συστημάτων, με σκοπό την εκμετάλλευση των δυνατοτήτων του κάθε συστήματος, όπως το πλήθος χωρικών λειτουργιών που προσφέρουν τα SQL συστήματα με την ταχύτητα επεξεργασίας και τον όγκο αποθήκευσης των NoSQL συστημάτων. Τέλος, με τον συνεχώς αυξανόμενο όγκο των δεδομένων και την αυξανόμενη ανάγκη για παράλληλη πρόσβαση και επεξεργασία των δεδομένων, οι ερευνητές, οι εταιρίες αλλά και η κοινότητα (community), θα συνεχίσουν να βελτιώνουν τόσο τα SQL συστήματα διαχείρισης χωρικών βάσεων δεδομένων, ώστε να μπορούν να διαχειριστούν μεγάλους όγκους δεδομένων, όσο και τα NoSQL συστήματα, ώστε να παρέχουν περισσότερες δυνατότητες για χωρική ανάλυση.

Βιβλιογραφία

1) Kraak, M.J. and Ormeling, F., 2020. Cartography: visualization of geospatial data. CRC Press.

2) Dunkel, A., 2015. Visualizing the perceived environment using crowdsourced photo geodata. Landscape and urban planning, 142, pp.173-186.

3) Güting, R.H., 1994. An introduction to spatial database systems. the VLDB Journal, 3, pp.357-399.

4) Obe, R. and Hsu, L.S., 2021. PostGIS in action. Simon and Schuster.

5) Lee, J.G. and Kang, M., 2015. Geospatial big data: challenges and opportunities. Big Data Research, 2(2), pp.74-81.

6) Amirian, P., Basiri, A. and Winstanley, A., 2014. Evaluation of data management systems for geospatial big data. In Computational Science and Its Applications–ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30–July 3, 2014, Proceedings, Part V 14 (pp. 678-690). Springer International Publishing.

7) Guo, D. and Onstein, E., 2020. State-of-the-art geospatial information processing in NoSQL databases. ISPRS International Journal of Geo-Information, 9(5), p.331.

8) Agoub, A., Kunde, F. and Kada, M.A.R.T.I.N., 2016. Potential of graph databases in representing and enriching standardized Geodata. Tagungsband der, 36, pp.208-216.

9) Agarwal, S. and Rajan, K.S., 2017. Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries. In Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings (Vol. 17, No. 1, p. 4).

10) Baralis, E., Dalla Valle, A., Garza, P., Rossi, C. and Scullino, F., 2017, December. SQL versus NoSQL databases for geospatial applications. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 3388-3397). IEEE.

11) Bierbauer, R., Krismer, D.I.N., Silbernagl, D. and Specht, G., 2016. Comparing and evaluating routing implementations in SQL and NoSQL database systems.

12) Overview (no date) Isochrone Project. Available at: https://dbs.inf.unibz.it/archive/projects/isochrone/index.html (Accessed: 18 August 2023).

13) Karmas, A., Karantzalos, K. and Athanasiou, S., 2014, July. Online analysis of remote sensing data for agricultural applications. In Proceedings of OSGeo's European Conference on Free and Open Source Software for Geospatial, Bermen, Germany.

14) Welcome to Rasdaman - the world's most flexible and scalable Datacube engine (no date) rasdaman. Available at: https://rasdaman.org/wiki (Accessed: 18 August 2023).

15) Web coverage processing service (WCPS) standard (2023) Open Geospatial Consortium. Available at: https://www.ogc.org/standard/wcps/ (Accessed: 18 August 2023).

16) Jing, W. and Tian, D., 2018. An improved distributed storage and query for remote sensing data. Procedia Computer Science, 129, pp.238-247.

17) Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, May. The hadoop distributed file system. In 2010 IEEE 26th symposium on mass storage systems and technologies (MSST) (pp. 1-10). Ieee.

18) Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J. and Ogden, J.M., 1984. Pyramid methods in image processing. RCA engineer, 29(6), pp.33-41.

19) Moon, B., Jagadish, H.V., Faloutsos, C. and Saltz, J.H., 2001. Analysis of the clustering properties of the Hilbert space-filling curve. IEEE Transactions on knowledge and data engineering, 13(1), pp.124-141.

20) Hein, N. and Blankenbach, J., Evaluation of a NoSQL Database for Storing Big Geospatial Raster Data. GI_Forum 2021,, 9, pp.76-84.

21) FIW e. V. at RWTH Aachen University (no date) RiverView®: FiW e. V. RWTH Aachen. Available at: https://www.fiw.rwth-aachen.de/en/references/riverviewr (Accessed: 18 August 2023).

22) Ramiaramanana, H., Guilbert, E. and Moulin, B., 2022. A COGNITIVE APPROACH FOR LANDSYSTEM IDENTIFICATION USING A GRAPH DATABASE–TOWARDS THE IDENTIFICATION OF LANDFORMS IN CONTEXT. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 4, pp.17-24.

23) Wu, C., Zhu, Q., Zhang, Y., Du, Z., Ye, X., Qin, H. and Zhou, Y., 2017. A NOSQL–SQL hybrid organization and management approach for real-time geospatial data: A case study of public security video surveillance. ISPRS International Journal of Geo-Information, 6(1), p.21.

24) Kumar, R., Sawhney, H.S., Asmuth, J.C., Pope, A. and Hsu, S., 1998, August. Registration of video to geo-referenced imagery. In Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170) (Vol. 2, pp. 1393-1400). IEEE.

25) Jung, C.T., Tsou, M.H. and Issa, E., 2015, March. Developing a real-time situation awareness viewer for monitoring disaster impacts using location-based social media messages in Twitter. In International Conference on Location-Based Social Media Data, Athens, GA, USA March (pp. 12-14).

26) Ilba, M., 2021. Parallel algorithm for improving the performance of spatial queries in SQL: The use cases of SQLite/SpatiaLite and PostgreSQL/PostGIS databases. Computers & Geosciences, 155, p.104840.

27) Wang, B., Shin, R., Liu, X., Polozov, O. and Richardson, M., 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. arXiv preprint arXiv:1911.04942.

28) Sikorski, R., 1969. Boolean algebras (Vol. 2). New York: Springer.

29) Zadeh, L.A., 1988. Fuzzy logic. Computer, 21(4), pp.83-93.

30) Ďuračiová, R., 2013. Querying uncertain data in geospatial object-relational databases using SQL and fuzzy sets. Slovak Journal of Civil Engineering, 21(4), pp.1-12.

31) Nayak, A., Poriya, A. and Poojary, D., 2013. Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, 5(4), pp.16-19.

32) Ong, K.W., Papakonstantinou, Y. and Vernoux, R., 2014. The SQL++ unifying semi-structured query language, and an expressiveness benchmark of SQL-on-Hadoop, NoSQL and NewSQL databases. CoRR, abs/1405.3631

33) Ong, K.W., Papakonstantinou, Y. and Vernoux, R., 2014. The SQL++ query language: Configurable, unifying and semi-structured. arXiv preprint arXiv:1405.3631.

34) Walmsley, P., 2007. XQuery. " O'Reilly Media, Inc.".

35) Buneman, P., Davidson, S., Hillebrand, G. and Suciu, D., 1996, June. A query language and optimization techniques for unstructured data. In Proceedings of the 1996 ACM SIGMOD international conference on Management of data (pp. 505-516).

36) Atzeni, P., Bugiotti, F. and Rossi, L., 2012, March. SOS (save our systems) a uniform programming interface for non-relational systems. In Proceedings of the 15th International Conference on Extending Database Technology (pp. 582-585).

37) Bray, T., 2014. The javascript object notation (json) data interchange format (No. rfc7159).

38) Schmid, S., Galicz, E. and Reinhardt, W., 2015. Performance investigation of selected SQL and NoSQL databases. In Proceedings of the AGILE (pp. 1-5).

39) Makris, A., Tserpes, K., Anagnostopoulos, D., Nikolaidou, M. and de Macedo, J.A.F., 2019, June. Database system comparison based on spatiotemporal functionality. In Proceedings of the 23rd international database applications & engineering symposium (pp. 1-7).

40) Bartoszewski, D., Piorkowski, A. and Lupa, M., 2019. The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases. In Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis: 15th International Conference, BDAS 2019, Ustroń, Poland, May 28–31, 2019, Proceedings 15 (pp. 291-302). Springer International Publishing.

41) Li, S., Dragicevic, S., Castro, F.A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A. and Cheng, T., 2016. Geospatial big data handling theory and methods: A review and research challenges. ISPRS journal of Photogrammetry and Remote Sensing, 115, pp.119-133.

42) Khan, S. and Kannapiran, T., 2019, March. Indexing issues in spatial big data management. In International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India.

43) Kouiroukidis, N. and Evangelidis, G., 2011, September. The effects of dimensionality curse in high dimensional knn search. In 2011 15th Panhellenic Conference on Informatics (pp. 41-45). IEEE.

44) Verleysen, M. and François, D., 2005, June. The curse of dimensionality in data mining and time series prediction. In International work-conference on artificial neural networks (pp. 758-770). Berlin, Heidelberg: Springer Berlin Heidelberg.

45) Krommyda, M. and Kantere, V., 2020, September. Spatial Data Management in IoT systems: A study of available storage and indexing solutions. In 2020 Second International Conference on Transdisciplinary AI (TransAI) (pp. 146-153). IEEE.

46) Rigaux, P., Scholl, M. and Voisard, A., 2002. Spatial databases: with application to GIS. Morgan Kaufmann.

47) SEMI-STRUCTURED, D.P.W.A. and DATA, G.,2019. Mohamad Hasan Evgeny Panidi Vladimir Badenko.

48) Coronel, C. and Morris, S.A., 2022. Database systems: design, implementation and management. Cengage learning.

49) MySQL 8.0 Reference Manual :: 11.4 Spatial Data types (no date) MySQL. Available at: https://dev.mysql.com/doc/refman/8.0/en/spatial-types.html (Accessed: 23 August 2023).

50) Herring, J.R., 2006. Opengis implementation specification for geographic information-simple feature access-part 2: Sql option. Open Geospatial Consortium Inc.

51) Stolze, K., 2003. SQL/MM spatial: The standard to manage spatial data in a relational database system. In BTW 2003–Datenbanksysteme für Business, Technologie und Web, Tagungsband der 10. BTW Konferenz. Gesellschaft für Informatik eV.

55) Guttman, A., 1984, June. R-trees: A dynamic index structure for spatial searching. In Proceedings of the 1984 ACM SIGMOD international conference on Management of data (pp. 47-57).

53) Comer, D., 1979. Ubiquitous B-tree. ACM Computing Surveys (CSUR), 11(2), pp.121-137.

54) Rouault, E., Warmerdam, F., Schwehr, K., Kiselev, A., Butler, H., Łoskot, M., Szekeres, T., Tourigny, E., Landa, M., Miara, I., Elliston, B., Chaitanya, K., Plesea, L., Morissette, D., Jolma, A., Dawson, N., Baston, D., de Stigter, C., & Miura, H. (2023). GDAL (Version 3.7.0) [Computer software]. https://doi.org/10.5281/zenodo.5884351

55) Spatialite: Spatialite (no date) RSS. Available at: https://www.gaia-gis.it/fossil/libspatialite/home (Accessed: 23 August 2023).

56) Librasterlite2: Librasterlite2 (no date) RSS. Available at: https://www.gaia-gis.it/fossil/librasterlite2/index (Accessed: 23 August 2023).

57) Liu, C., Ma, R. and Zhang, L., 2020. Analysis of spatial indexing mechanism and its application in data management: A case study on spatialite database. In IOP Conference Series: Earth and Environmental Science (Vol. 428, No. 1, p. 012037). IOP Publishing.

58) Beckmann, N., Kriegel, H.P., Schneider, R. and Seeger, B., 1990, May. The R*-tree: An efficient and robust access method for points and rectangles. In Proceedings of the 1990 ACM SIGMOD international conference on Management of data (pp. 322-331).

59) Libspatialite v.2.4.0-RC5B experimental spatial index update - gaia-gis.it. Available at: http://www.gaia-gis.it/gaia-sins/SpatialIndex-Update.pdf (Accessed: 23 August 2023).

60) Garnett, R. and Kanaroglou, P., 2016. Qualitative GIS: An open framework using spatialite and open source GIS. Transactions in GIS, 20(1), pp.144-159.

61) Engines ranking (no date) DB. Available at: https://db-engines.com/en/ranking/spatial+dbms (Accessed: 24 August 2023).

62) PROJ contributors (2023). PROJ coordinate transformation software library. Open Source Geospatial Foundation. URL https://proj.org/. DOI: 10.5281/zenodo.5884394

63) Agarwal, S. and Rajan, K.S., 2016. Performance analysis of MongoDB versus PostGIS/PostGreSQL databases for line intersection and point containment spatial queries. Spatial Information Research, 24, pp.671-677.

64) Chaudhry, N. and Yousaf, M.M., 2019, February. Spatial querying of mineral resources using PostGIS. In 2019 2nd International Conference on Advancements in Computational Sciences (ICACS) (pp. 1-6). IEEE.

65) ST_Distance_Sphere. Available at: https://postgis.net/docs/manual-2.1/ST_Distance_Sphere.html (Accessed: 24 August 2023).

66) Hellerstein, J.M., Naughton, J.F. and Pfeffer, A., 1995. Generalized search trees for database systems (pp. 562-573). September.

67) Brin indexes (2023) PostgreSQL Documentation. Available at: https://www.postgresql.org/docs/current/brin.html (Accessed: 24 August 2023).

68) Aref, W.G. and Ilyas, I.F., 2001. Sp-gist: An extensible database index for supporting space partitioning trees. Journal of Intelligent Information Systems, 17, pp.215-240.

69) Finkel, R.A. and Bentley, J.L., 1974. Quad trees a data structure for retrieval on composite keys. Acta informatica, 4, pp.1-9.

70) Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9), pp.509-517.

71) Leis, V., Kemper, A. and Neumann, T., 2013, April. The adaptive radix tree: ARTful indexing for main-memory databases. In 2013 IEEE 29th International Conference on Data Engineering (ICDE) (pp. 38-49). IEEE.

72) Cattell, R., 2011. Scalable SQL and NoSQL data stores. Acm Sigmod Record, 39(4), pp.12-27.

73) Hecht, R. and Jablonski, S., 2011, December. NoSQL evaluation: A use case oriented survey. In 2011 International Conference on Cloud and Service Computing (pp. 336-341). IEEE.

74) Li, Z., 2018. NoSQL databases. Geographic Information Science and Technology Body of Knowledge, 2018(Q2).

75) Brewer, E.A., 2000, July. Towards robust distributed systems. In PODC (Vol. 7, No. 10.1145, pp. 343477-343502).

76) Chandra, D.G., 2015. BASE analysis of NoSQL database. Future Generation Computer Systems, 52, pp.13-21.

77) DB - Engines (no date) DB. Available at: https://db-engines.com/en/ (Accessed: 25 August 2023).

78) Redis Geospatial (no date) Redis. Available at: https://redis.io/docs/data-types/geospatial/ (Accessed: 24 August 2023).

79) Balkić, Z., Šoštarić, D. and Horvat, G., 2012. GeoHash and UUID identifier for multi-agent systems. In Agent and Multi-Agent Systems. Technologies and Applications: 6th KES International Conference, KES-AMSTA 2012, Dubrovnik, Croatia, June 25-27, 2012. Proceedings 6 (pp. 290-298). Springer Berlin Heidelberg.

80) Haber, I. (2016) RedisLabs/geo.lua: A helper library for Redis Geospatial Indices, GitHub. Available at: https://github.com/RedisLabs/geo.lua/tree/master (Accessed: 24 August 2023).

81) Robusto, C.C., 1957. The cosine-haversine formula. The American Mathematical Monthly, 64(1), pp.38-40.

82) Thomas, G., Alexander, G. and Sasi, P.M., 2017, July. Design of high performance cluster based map for vehicle tracking of public transport vehicles in smart city. In 2017 IEEE Region 10 Symposium (TENSYMP) (pp. 1-5). Ieee.

83) Parker, Z., Poe, S. and Vrbsky, S.V., 2013, April. Comparing nosql mongodb to an sql db. In Proceedings of the 51st ACM Southeast Conference (pp. 1-6).

84) Geospatial Queries — MongoDB Manual (no date). Available at: https://www.mongodb.com/docs/manual/geospatial-queries/ (Accessed: 25 August 2023).

85) GridFS — MongoDB Manual (no date). Available at: https://www.mongodb.com/docs/manual/core/gridfs/ (Accessed: 25 August 2023).

86) Wang, W. and Hu, Q., 2014, June. The method of cloudizing storing unstructured LiDAR point cloud data by MongoDB. In 2014 22nd International Conference on Geoinformatics (pp. 1-5). IEEE.

87) Wang, S., Li, G., Yao, X., Zeng, Y., Pang, L. and Zhang, L., 2019. A distributed storage and access approach for massive remote sensing data in Mongodb. ISPRS International Journal of Geo-Information, 8(12), p.533.

88) The Cassandra Query Language (CQL) | Apache Cassandra Documentation (no date). Available at: https://cassandra.apache.org/doc/latest/cassandra/cql/ (Accessed: 25 August 2023).

89) Ben Brahim, M., Drira, W., Filali, F. and Hamdi, N., 2016. Spatial data extension for Cassandra NoSQL database. Journal of Big Data, 3, pp.1-16.

90) Stratio's Cassandra Lucene Index (no date). Available at: https://github.com/Stratio/cassandra-lucene-index (Accessed: August 25, 2023).

91) GeoMesa (no date). Available at: https://www.geomesa.org/ (Accessed: August 25, 2023).

92) Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P. and Taylor, A., 2018, May. Cypher: An evolving query language for property graphs. In Proceedings of the 2018 international conference on management of data (pp. 1433-1445).

93) Spatial functions - Cypher Manual (no date). Available at: https://neo4j.com/docs/cypher-manual/current/functions/spatial/ (Accessed: August 25, 2023).

94) Taverner, C. (2022) Neo4j Spatial. Available at: https://github.com/neo4j-contrib/spatial (Accessed: August 25, 2023).

95) Bhogaram, P., Wu, X., He, M. and Okenwa, O., 2020. Optimal and Critical Path Analysis of State Transportation Network Using Neo4J. International Journal of Urban and Civil Engineering, 14(10), pp.312-317.

96) Tamilmani, R. and Stefanakis, E., 2019. Modelling and analysis of semantically enriched simplified trajectories using graph databases. Advances in Cartography and GIScience of the ICA, 1, p.20.

97)Baas, B.L.P., 2012. Nosql spatial–neo4j versus postgis (Master's thesis).

98) Herring, J., 2011. Opengis® implementation standard for geographic information-simple feature access-part 1: Common architecture [corrigendum].

99) Bennett, J., 2010. OpenStreetMap. Packt Publishing Ltd.

100) Bossard, M., Feranec, J. and Otahel, J., 2000. CORINE land cover technical guide: Addendum 2000 (Vol. 40). Copenhagen: European Environment Agency.

101) Jutz, S. and Milagro-Pérez, M.P., 2020. Copernicus: the European Earth Observation programme. Revista de Teledetección, (56), pp.V-XI.

102) QGIS.org, (2023). QGIS Geographic Information System. QGIS Association. http://www.qgis.org

103) European Space Agency [ESA] (no date) User Guides - Sentinel-2 MSI - Overview - Sentinel Online - Sentinel Online. Available at: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/overview (Accessed: August 28, 2023).

104) Pettorelli, N., 2013. The normalized difference vegetation index. Oxford University Press, USA.

105) Small, N. (2021) The Py2neo Handbook — py2neo 2021.1. Available at: https://py2neo.org/2021.1/ (Accessed: August 28, 2023).

106) Jordahl, K., 2016. GeoPandas Documentation. URL: http://sethc23. github. io/wiki/Python/geopandas. pdf-Download vom, 26, p.2022.

107) Gillies, S., 2013. The shapely user manual. URL https://pypi. org/project/Shapely.