ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Πειραματική σύγκριση τεχνολογιών συστημάτων διαχείρισης βάσεων δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αχιλλέας, Δ. Μιχαηλίδης

Επιβλέπων : Ιάκωβος Ν. Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιανουάριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Πειραματική σύγκριση τεχνολογιών συστημάτων διαχείρισης βάσεων δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αχιλλέας, Δ. Μιχαηλίδης

Επιβλέπων : Ιάκωβος Ν. Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Ιανουαρίου 2020.

.....	.....	.....
Ιάκωβος Βενιέρης	Δήμητρα Θεοδώρα Κακλαμάνη	Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιανουάριος 2020

.....
Αχιλλέας, Δ. Μιχαηλίδης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι πρώτες δύο δεκαετίες του 21ου αιώνα έχουν φέρει μια επανάσταση στον τρόπο αλληλεπίδρασης του ανθρώπου με έναν υπολογιστή. Σε αυτή τη σύγχρονη και συνεχώς αναπτυσσόμενη ψηφιακή κοινωνία, τα δεδομένα και η πληροφορία έχουν κατακλίσει κάθε πτυχή της καθημερινότητάς μας. Η αποθήκευση και η άμεση πρόσβαση σε αυτή την πληροφορία γίνεται όλο και πιο επιτακτική με αποτέλεσμα και τη ραγδαία ανάπτυξη νέων τεχνολογιών και μεθόδων αποθήκευσης και επεξεργασίας των δεδομένων.

Αντικείμενο της συγκεκριμένης διπλωματικής εργασίας είναι η μελέτη και η σύγκριση της ανάκτησης δεδομένων από τα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων σε σύγκριση με τα παραδοσιακά συστήματα διαχείρισης με σκοπό την επιλογή του καταλληλότερου συστήματος για τη μορφή των δεδομένων που έχουμε στη διάθεση μας. Στην παρούσα διπλωματική θα μελετηθούν τα συστήματα διαχείρισης βάσεων δεδομένων MySQL, MongoDB, Elasticsearch και Neo4j πάνω σε γεωγραφικά δεδομένα και δεδομένα κειμένου.

Λέξεις κλειδια: Δεδομένα, Συστήματα Διαχείρισης Βάσεων Δεδομένων, NoSQL

Abstract

The first two decades of the 21st century have brought a revolution in the way humans interact with computers. In this modern and constantly evolving digital society, data and information have flooded every aspect of our daily lives. Storing and accessing this information instantly is becoming increasingly crucial, resulting in the rapid development of new technologies and methods for storing and processing data.

The subject of this thesis is the study and comparison of data retrieval from modern database management systems in comparison with traditional management systems, with the aim of selecting the most suitable system for the type of data we have at our disposal. In this thesis, the database management systems MySQL, MongoDB, Elasticsearch, and Neo4j will be studied based on geographic and text data.

Keywords: Data, Database management systems, NoSQL

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή, κ. Ιάκωβο Βενιέρη, για την ανάθεση της διπλωματικής και για την ευκαιρία που μου δόθηκε μέσω αυτής να αποκτήσω πολλές γνώσεις πάνω σε σύγχρονα θέματα τεχνολογιών βάσεων δεδομένων.

Θα ήθελα επίσης να ευχαριστήσω τη Διδάκτορα Δέσποινα Μερίδου και τον υποψήφιο διδάκτορα Μανώλη Καραμανή για την υποστήριξη και την καθοδήγησή τους κατά τη διάρκεια εκπόνησης της διπλωματικής, τη διαθεσιμότητά τους να απαντήσουν σε κάθε είδους απορία και προβληματισμό.

Τέλος, θα ήθελα να ευχαριστήσω όλους όσους στάθηκαν δίπλα μου στην προσπάθεια μου όλα αυτά τα χρόνια.

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Ανάγκη για άμεση και γρήγορη ανάκτηση δεδομένων

Στη σύγχρονη εποχή της τεχνολογίας και της πληροφορίας τα δεδομένα έχουν καταστεί ένας από τους σημαντικότερους παράγοντες στην ανάλυση της πληροφορίας και στη λήψη αποφάσεων.

Ο όγκος των διαθέσιμων δεδομένων έχει αυξηθεί ραγδαία την τελευταία δεκαετία [1][2] καθώς η διείσδυση της τεχνολογίας στην καθημερινή μας ζωή έχει καταστήσει τη συλλογή δεδομένων για όλες τις πτυχές της καθημερινότητας μας μια διαδικασία τετριμμένη. Δεδομένα παράγονται συνεχώς, από τις πιο μικρές συσκευές που αποτελούν μέρος του σύγχρονου Internet of Things, τη ραγδαία αύξηση του όγκου δεδομένων που παράγονται πλέον μέσω των smartphones μέχρι τις ηλεκτρονικές συναλλαγές οι οποίες έχουν και αυτές με τη σειρά τους αυξηθεί δραματικά την τελευταία 10ετία [3][4], τα κοινωνικά δίκτυα και κάθε συναλλαγή που έχει ένας άνθρωπος με μία μηχανή. Όλα αυτά αποτελούν ένα μικρό παράδειγμα της αύξησης του όγκου των δεδομένων τα τελευταία χρόνια.

Γίνεται αντιληπτό πως η πληροφορία και τα δεδομένα είναι πολύ σημαντικά στο σύγχρονο τρόπο ζωής και είναι απαραίτητη η άμεση διαθεσιμότητα τους σε όλο το κοινωνικοοικονομικό φάσμα. Από τις τράπεζες και τις οικονομικές συναλλαγές, τις εταιρικές στρατηγικές αποφάσεις, τα κοινωνικά δίκτυα και την ανάγκη των ανθρώπων να επικοινωνούν και να καταναλώνουν, την ανάγκη για προσαρμοσμένο και στοχευμένο περιεχόμενο μέχρι τη μηχανική μάθηση για λύση σύνθετων προβλημάτων, ο κοινός παρονομαστής είναι η ανάγκη για τη σωστή αξιοποίηση των δεδομένων.

1.1.1 Οι δυσκολίες της ανάκτησης πληροφορίας από μεγάλο όγκο δεδομένων

Με την τεράστια αύξηση του όγκου δεδομένων δημιουργήθηκαν νέες προκλήσεις στην ανάκτηση της πληροφορίας. Η συνεχής αύξηση των πηγών παραγωγής δεδομένων σε συνδυασμό με τη πολύπλοκη και μη τετριμμένη δομή τους έχουν καταστίσει την ανάκτηση της πληροφορίας αρκετά δύσκολη. Η ανάγκη επίσης για γρήγορη και σχεδόν σε πραγματικό χρόνο λήψη αποφάσεων έχει προσθέσει ένα ακόμα επίπεδο δυσκολίας στην εξαγωγή δεδομένων η οποία επιδεινώνεται και από την ταχύτητα παραγωγής νέων δεδομένων που συνεχώς αυξάνεται.

Ο μεγάλος όγκος δεδομένων έχει επιφέρει και μεγάλες αλλαγές στον τρόπο αποθήκευσης των δεδομένων από πλευράς υλικού [5]. Το παραδοσιακό ένα και μοναδικό μέσο αποθήκευσης (για παράδειγμα ένας σκληρός δίσκος), έχει εξελιχθεί σε ένα πλέγμα με πολλαπλά μέσα αποθήκευσης, που το καθένα αποσκοπεί σε διαφορετικά πράγματα και τα οποία επικοινωνούν και συνεργάζονται ενώ μπορεί να μη βρίσκονται καν στον ίδιο φυσικό χώρο. Όλο αυτό έχει ένα σοβαρό αντίκτυπο στην αποθήκευση των δεδομένων και ανάκτηση της απαιτούμενης πληροφορίας.

Τέλος, πολύ σημαντικός παράγοντας, ειδικά στην εποχή των κοινωνικών δικτύων και της συνεχούς ψηφιοποίησης της ανθρώπινης ζωής και των προσωπικών δεδομένων, είναι η ασφάλεια αυτών των δεδομένων και η εμπιστευτικότητα. Η απαίτηση τα δεδομένα να είναι άμεσα προσβάσιμα και ταυτόχρονα προστατευμένα και ασφαλή από πιθανή κακόβουλη χρήση έχει επιφέρει μια μεγάλη δυσκολία τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο

στο τί είναι αυτό που πρέπει να αποθηκεύεται και πως πρέπει χρησιμοποιείται αυτού του είδους η πληροφορία [6].

1.1.2 Οι δυσκολίες της επιλογής του κατάλληλου συστήματος διαχείρισης βάσεων δεδομένων

Η επιλογή του κατάλληλου συστήματος διαχείρισης βάσεων δεδομένων είναι ένα πολύ σημαντικό βήμα στην επίλυση ενός προβλήματος και μια επιλογή που πρέπει να γίνει στην αρχή της ανάλυσης του προβλήματος καθώς θα επηρεάσει άμεσα την λύση του.

Η δυσκολία επιλογής του σωστού συστήματος εξαρτάται από πολλούς παράγοντες με τον ποιο σημαντικό να είναι το είδος του προβλήματος και το είδος των δεδομένων που θα χρησιμοποιηθούν. Το βασικά ερωτήματα που πρέπει να απαντηθούν είναι το πως και γιατί θα χρησιμοποιηθούν τα δεδομένα. Ο όγκος επίσης των δεδομένων παίζει σημαντικό ρόλο και μια λύση που δεν έχει λάβει υπόψιν της τη επεκτασιμότητα, αν αυτή χρειάζεται, ενδέχεται να αντιμετωπίσει πολλά προβλήματα στο μέλλον. Επίσης πρέπει να ληφθεί υπόψιν η χρήση του συστήματος σε σχέση με τα ήδη υπάρχοντα συστήματα και τις τεχνολογίες που χρησιμοποιεί ένας οργανισμός. Τέλος, το οικονομικό κομμάτι είναι πολύ σημαντικό καθώς επηρεάζει την επιλογή. Αλλιώς θα επιλέξει μια μεγάλη εταιρία η οποία μπορεί να κινηθεί στην αγορά μιας enterprise λύσης, αλλιώς ένας μικρός οργανισμός ή κάποιος μεμονωμένος χρήστης [7] [8].

1.2 Αντικείμενο διπλωματικής εργασίας

Αντικείμενο της συγκεκριμένης διπλωματικής εργασίας είναι η μελέτη και η σύγκριση της ανάκτησης δεδομένων από τα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων σε σύγκριση με τα παραδοσιακά συστήματα διαχείρισης με σκοπό την επιλογή του καταλληλότερου συστήματος για τη μορφή των δεδομένων που έχουμε στη διάθεση μας.

Πιο συγκεκριμένα, στην παρούσα διπλωματική γίνεται σύγκριση των παραδοσιακών σχεσιακών συστημάτων διαχείρισης βάσεων δεδομένων με τα σύγχρονα, μη σχεσιακά συστήματα (NoSQL) στην ταχύτητα ανάκτησης πληροφορίας για γεωγραφικά δεδομένα και δεδομένα κειμένου.

Σκοπός είναι η εύρεση, αν υπάρχει, του καταλληλότερου συστήματος για τα εκάστοτε δεδομένα καθώς και η συσχέτιση της ταχύτητας ανάκτησης της πληροφορίας σε σχέση με το μέγεθος του όγκου των διαθέσιμων δεδομένων.

1.3 Διάρθρωση εργασίας

Το κείμενο της διπλωματικής εργασίας πέραν της εισαγωγής αυτής δομείται με τον παρακάτω τρόπο:

- Στο κεφάλαιο 2 γίνεται μια επεξήγηση των τεχνολογιών που υπάρχουν και θα εξεταστούν στα πλαίσια της παρούσας διπλωματικής καθώς επίσης και των συστημάτων και των εργαλείων που θα χρησιμοποιηθούν
- Στο κεφάλαιο 3 γίνεται μια παρουσίαση της υπάρχουσας βιβλιογραφίας και των αντίστοιχων μελετών που έχουν ήδη γίνει πάνω στο θέμα
- Στο κεφάλαιο 4 γίνεται η παρουσίαση των δεδομένων που χρησιμοποιήθηκαν και εξετάστηκαν, του τρόπου χρήσης των συστημάτων διαχείρισης βάσεων δεδομένων. Παρουσιάζεται ο τρόπος εκτέλεσης των πειραμάτων και όλες οι λεπτομέρειες

εκτέλεσης της τεχνικής πλευράς της παρούσας διπλωματικής. Στο τέλος του κεφαλαίου παρουσιάζονται τα αποτελέσματα

- Στο κεφάλαιο 5 γίνεται μια αξιολόγηση των αποτελεσμάτων με την εξαγωγή συμπερασμάτων καθώς επίσης παρατίθενται και πιθανές μελλοντικές επεκτάσεις

2 Τεχνολογίες

2.1 Συστήματα Διαχείρισης Βάσεων Δεδομένων

Με τον όρο Σύστημα Διαχείρισης Βάσης Δεδομένων, γνωστό ως Database Management system (DBMS), εννοείται είτε κάποιο λογισμικό μέσω του οποίου γίνεται η δημιουργία, η διαχείριση, η συντήρηση και η χρήση μιας ηλεκτρονικής βάσης δεδομένων [9] ή ένα σύνολο αλληλοσυσχετιζόμενων προγραμμάτων που τρέχουν και διαχειρίζονται τα δεδομένα μιας τέτοιας βάσης. Το λογισμικό χρησιμοποιεί στερεότυπες μεθόδους καταλογοποίησης, ανάκτησης, και εκτέλεσης ερωτημάτων σχετικών με τα δεδομένα. Το σύστημα διαχείρισης οργανώνει τα εισερχόμενα δεδομένα με τρόπους χρησιμοποιήσιμους από εξωτερικούς χρήστες [10].

2.2 Σχεσιακές βάσεις δεδομένων

Με τον όρο σχεσιακή βάση δεδομένων εννοείται ένα σύνολο από συσχετισμένους ή μη πίνακες μαζί μηχανισμούς αποθήκευσης, ανάκτησης και επεξεργασίας δεδομένων βασιζόμενη στο σχεσιακό μοντέλο, το οποίο πρώτη φορά επεξηγήθηκε το 1970 από τον Εντγκαρ Κοντ [11]. Ο σκοπός μιας βάσης δεδομένων τέτοιου τύπου είναι η οργανωμένη αποθήκευση και επεξεργασία πληροφορίας με τη χρήση ερωτημάτων προς τη βάση δεδομένων τα οποία γίνονται με τη χρήση της γλώσσας SQL (Structured Query Language) η οποία σχεδιάστηκε και δημιουργήθηκε για αυτό το σκοπό το 1974 [12] και έχει τυποποιηθεί στα πρότυπα ANSI και ISO.

Μια σχεσιακή βάση δεδομένων οργανώνει τα δεδομένα σε ένα ή περισσότερους πίνακες οι οποίοι αποτελούν την αναπαράσταση κάποιας οντότητας στη βάση. Κάθε πίνακας αποτελείται από μία ή περισσότερες σειρές ή εγγραφές οι οποίες αντιπροσωπεύουν διαφορετικές περιπτώσεις της οντότητας καθώς και από μία ή περισσότερες στήλες οι οποίες αντιπροσωπεύουν τις τιμές που αποδίδονται σε κάθε διαφορετική περίπτωση της οντότητας. Κάθε πίνακας μπορεί να χρησιμοποίει από μία ή περισσότερες στήλες ως στήλες μοναδικότητας για κάθε εγγραφή του. Οι πίνακες/οντότητες συνδέονται μεταξύ τους μέσω λογικών συνδέσεων οι οποίες εκφράζονται με χρήση περιορισμών αναφοράς (referential constraint).

2.2.1 Το Σχεσιακό Μοντέλο

Το σχεσιακό μοντέλο για τη διαχείριση βάσεων δεδομένων είναι μια προσέγγιση στη διαχείριση δεδομένων χρησιμοποιώντας μια δομή και μια γλώσσα συμβατή με την κατηγορηματική λογική πρώτης τάξης, που περιεγράφηκε για πρώτη φορά το 1970 από τον Εντγκαρ Κοντ [11].

Βασικό δομικό στοιχείο του Σχεσιακού Μοντέλου που περιέγραψε ο Κοντ είναι η *σχέση*, η οποία είναι ένα υποσύνολο του καρτεσιανού γινομένου ενός ή περισσότερων πεδίων τιμών. Το σχήμα της σχέσης αποτελείται από το όνομα της σχέσης και τα γνωρίσματά της. Ο αριθμός των γνωρισμάτων αποτελεί το βαθμό της σχέσης. Το στιγμιότυπο μιας σχέσης αποτελείται από μία ή περισσότερες ν-πλειάδες. Οι ν-πλειάδες είναι διατεταγμένες, μοναδικές και η διάταξη τους μέσα στο στιγμιότυπο της σχέση δεν έχει σημασία.

Ορίζονται επίσης οι παρακάτω περιορισμοί στο Σχεσιακό μοντέλο:

- Περιορισμός πεδίου τιμών, ο οποίο ορίζει ότι η τιμή κάθε γνωρίσματος πρέπει να είναι μια ατομική τιμή του πεδίου τιμών του γνωρίσματος

- Περιορισμός κλειδιού. Κάθε υποσύνολο γνωρισμάτων για το οποίο δεν υπάρχει ζευγάρι πλειάδων που να έχει τις ίδιες τιμές ονομάζεται υπερκλειδί και το ελάχιστο υπερκλειδί μιας σχέσης ονομάζεται κλειδί. Μια σχέση μπορεί να έχει ένα ή περισσότερα κλειδιά, όμως αυτό που τελικά επιλέγεται αποτελεί τελικά το πρωτεύων κλειδί της σχέσης
- Περιορισμός ακεραιότητας, ο οποίος ορίζει ότι δε μπορεί η τιμή ενός πρωτεύοντος κλειδιού να είναι κενή
- Περιορισμός αναφορικής ακεραιότητας, ο οποίος ορίζεται μεταξύ δύο σχέσεων και απαιτεί την αναφορά μιας πλειάδας της μια σχέσης σε υπαρκτή πλειάδα της άλλης σχέσης

2.2.1.1 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης βάσεων δεδομένων που αναπτύσσεται, διανέμεται και υποστηρίζεται από την Oracle Corporation [13]. Βασικό χαρακτηριστικό της είναι πως πρόκειται για ένα λογισμικό ανοιχτού κώδικα, το οποίο σημαίνει ότι είναι δυνατό για οποιονδήποτε να χρησιμοποιήσει και να τροποποιήσει το λογισμικό. Η πρώτη έκδοση του λογισμικού έγινε διαθέσιμη το 1995 [14] με το όνομα mSQL. Το λογισμικό βάσης δεδομένων MySQL είναι ένα σύστημα πελάτη διακομιστή που αποτελείται από ένα διακομιστή SQL πολλαπλών νημάτων που υποστηρίζει πολλά διαφορετικά προγράμματα και βιβλιοθήκες πελάτη, εργαλεία διαχείρισης και ένα ευρύ φάσμα διασυνδέσεων προγραμματισμού εφαρμογών (API).

Η MySQL είναι γραμμένη σε C και C++ και λειτουργεί σε πολλές πλατφόρμες συστημάτων, όπως το FreeBSD, το Linux, το MacOS, τα Microsoft Windows και το Oracle Solaris. Για τα ερωτήματα στη βάση δεδομένων χρησιμοποιείται ένα ευρύ υποσύνολο της SQL 99, μια τυποποιημένη έκδοση της SQL που προτάθηκε από το Αμερικανικό Εθνικό Ινστιτούτο Προτύπων (ANSI), εμπλουτισμένη με διάφορες επεκτάσεις. Η MySQL προσφέρει συμμόρφωση ως προς την έννοια του ACID, που σημαίνει ότι η συναλλαγές με το σύστημα διαχείρισης της βάσεων δεδομένων διέπονται από ατομικότητα, συνέπεια, απομόνωση και μονιμότητα [16].

Η MySQL είναι ένα κεντρικό στοιχείο της στοίβας λογισμικού ανοιχτού κώδικα LAMP (και άλλων στοιβών "AMP"). Το LAMP είναι ένα αρκτικόλεξο για τα "Linux, Apache, MySQL, Perl/PHP/Python". Αξιοσημείωτοι χρήστες της MySQL σήμερα είναι οι Facebook, YouTube, Netflix, Spotify και άλλοι [15].

2.3 NoSQL βάσεις δεδομένων

Ο όρος NoSQL πρωτοεμφανίστηκε το 1998 και η κύρια ιδέα πίσω από αυτόν είναι η προσπάθεια για ένα σχεσιακό μοντέλο βάσεων δεδομένων χωρίς τη χρήση SQL [17], και στη συνέχεια, στα τέλη της δεκαετίας των 00s, επεκτάθηκε στη μη χρήση πινάκων και σχέσεων, σε πλήρη αντίθεση δηλαδή με το σχεσιακό μοντέλο [18]. Αυτοί οι δύο τύποι των μοντέλων είναι πολύ διαφορετικοί όχι μόνο στη δομή τους αλλά και στον τρόπο που προσφέρουν πρόσβαση στα δεδομένα. Τα κίνητρα για αυτήν την προσέγγιση περιλαμβάνουν: την απλότητα του σχεδιασμού, την απλούστερη "οριζόντια" κλιμάκωση σε συστοιχίες μηχανών (που αποτελεί πρόβλημα για σχεσιακές βάσεις δεδομένων), όπως επίσης και τον καλύτερο έλεγχο της διαθεσιμότητας [19].

Οι δομές δεδομένων που χρησιμοποιούνται από τις βάσεις δεδομένων NoSQL (π.χ. τιμή-κλειδί, στήλη, γράφος ή έγγραφο) είναι διαφορετικές από αυτές που χρησιμοποιούνται από

προεπιλογή στις σχεσιακές βάσεις δεδομένων, κάνοντας ορισμένες λειτουργίες πιο γρήγορα σε NoSQL. Η καταλληλόλητα μιας δεδομένης βάσης δεδομένων NoSQL εξαρτάται από το πρόβλημα που πρέπει να επιλυθεί. Μερικές φορές οι δομές δεδομένων που χρησιμοποιούνται από τις βάσεις δεδομένων NoSQL θεωρούνται επίσης "πιο ευέλικτες" από τους πίνακες σχεσιακών βάσεων δεδομένων.

Έχουν υπάρξει διάφορες προσεγγίσεις για την ταξινόμηση βάσεων δεδομένων NoSQL, καθένα με διαφορετικές κατηγορίες και υποκατηγορίες, μερικές από τις οποίες επικαλύπτονται αλλά οι πιο συνηθισμένες κατηγορίες είναι: βάσεις δεδομένων που βασίζονται σε έγγραφα, βάσεις δεδομένων που βασίζονται σε γράφους, βάσεις δεδομένων βασισμένες σε στήλες και βάσεις κλειδιού τιμής.

2.3.1 Το Εγγραφοκεντρικό μοντέλο

Οι βάσεις δεδομένων με γνώμονα το έγγραφο είναι μία από τις κύριες κατηγορίες των βάσεων δεδομένων τύπου NoSQL. Οι βάσεις δεδομένων εγγράφων αντιτίθενται έντονα με την παραδοσιακή σχεσιακή βάση δεδομένων (RDB). Οι σχεσιακές βάσεις δεδομένων γενικά αποθηκεύουν τα δεδομένα σε ξεχωριστούς πίνακες που ορίζονται από τον προγραμματιστή και ένα μόνο αντικείμενο μπορεί να αναφερθεί από διάφορους πίνακες. Οι βάσεις δεδομένων εγγράφων αποθηκεύουν όλες τις πληροφορίες για ένα δεδομένο αντικείμενο σε μια μοναδική εμφάνιση στη βάση δεδομένων και κάθε αποθηκευμένο αντικείμενο μπορεί να είναι διαφορετικό από κάθε άλλο χαλαρώνοντας την έννοια του σχήματος της βάσης. Αυτό καθιστά την χαρτογράφηση αντικειμένων στη βάση δεδομένων μια απλή εργασία [20].

Η κεντρική ιδέα μιας εγγραφοκεντρικής βάσης δεδομένων είναι φυσικά η έννοια ενός εγγράφου. Παρόλο που κάθε σύστημα διαχείρισης βάσης δεδομένων που βασίζεται σε έγγραφα μπορεί να διαφέρει σε κάποιες λεπτομέρειες από αυτό τον ορισμό, εν γένει, όλα τα συστήματα αποθηκεύουν και κωδικοποιούν δεδομένα (ή πληροφορίες) με κάποια τυπική μορφή ή κωδικοποίηση. Οι κωδικοποιήσεις που χρησιμοποιούνται περιλαμβάνουν XML, YAML, JSON και BSON, καθώς και δυαδικές μορφές όπως έγγραφα PDF και Microsoft Office (MS Word, Excel κ.ο.κ.). Στην ίδια κατηγορία κατατάσσονται και οι βάσεις δεδομένων που χρησιμοποιούνται σας μηχανές αναζήτησης και αποθηκεύουν τα δεδομένα σε μορφή εγγράφων

Η εγγραφοκεντρικές βάσεις δεδομένων έχουν γίνει αρκετά ελκυστικές τα τελευταία χρόνια ιδιαίτερα για τον προγραμματισμό εφαρμογών ιστού, τα οποία υπόκεινται σε συνεχή αλλαγή και όπου η ταχύτητα ανάπτυξης είναι ένα σημαντικό ζήτημα.

Υπάρχουν πολλές υλοποιήσεις εγγραφοκεντρικών βάσεων δεδομένων με τις πιο δημοφιλής να είναι οι MongoDB, CouchDB, Couchbase και Amazon DynamoDB [21].

2.3.1.1 Elasticsearch

Η Elasticsearch είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) με κυρίως σκοπό την αναζήτηση κειμένου. Παρέχει μία κατανεμημένη μηχανή αναζήτησης πλήρους κειμένου (full-text search) με δυνατότητα εξυπηρέτησης πολλαπλών χρηστών, μία HTTP διεπαφή και JSON έγγραφα χωρίς schema. Έχει αναπτυχθεί σε Java και είναι λογισμικό ανοιχτού κώδικα κάτω από τους όρους της άδειας χρήσης Apache License [22]. Η Elasticsearch είναι χτισμένη πάνω από τον apache Lucene, και είναι η δημοφιλέστερη

μηχανή αναζήτησης στην βιομηχανία δεδομένων ακολουθούμενη από το Apache Solr [23], το οποίο είναι επίσης βασισμένο στο Lucene.

Η Elasticsearch μπορεί να χρησιμοποιηθεί για αναζήτηση σε κάθε είδους έγγραφα. Προσφέρει κλιμάκωση αναζήτηση, έχει αναζήτηση σχεδόν πραγματικού χρόνου (real-time search) και υποστηρίζει ύπαρξη πολλών χρηστών. Είναι κατανεμημένη, κάτι που σημαίνει ότι τα indexes της (αντίστοιχα με βάση δεδομένων σε παραδοσιακά DMBS) μπορούν να χωριστούν σε shards και κάθε shard να έχει από καθόλου μέχρι πολλά αντίγραφα. Κάθε κόμβος φιλοξενεί ένα ή περισσότερα shards, και λειτουργεί ως συντονιστής ώστε να διαμοιράζει τις ενέργειες στα κατάλληλα shards. Υποστηρίζεται αυτόματη αναπροσαρμογή φόρτου εργασίας μεταξύ των κόμβων [22].

Βασικές δομές της Elasticsearch είναι:

- τα indexes, που είναι ότι και η βάση για τα σχεσιακά μοντέλα
- τα types, που είναι οι αντίστοιχοι πίνακες του σχεσιακού μοντέλου
- τα documents, κάποιου type που αποτελούν τις αντίστοιχες εγγραφές σε κάποιο πίνακα σχεσιακής βάσης
- τα fields, τα οποία είναι οι ιδιότητες κάθε εγγράφου, κάτι αντίστοιχο με της στήλες ενός πίνακα

Το σύστημα αυτό χρησιμοποιεί το εργαλείο Lucene, μία πολύ ισχυρή βιβλιοθήκη ανοιχτού κώδικα για αναζήτηση πλήρους κειμένου (full-text search) [24], και προσπαθεί να διαθέσει όλα τα χαρακτηριστικά του μέσω του JSON και Java API που παρέχει.

Αξιοσημείωτοι χρήστες της Elasticsearch σήμερα είναι οι Adobe, Facebook, Mozilla, Quora, GitHub, Stack Exchange, Netflix και άλλοι [25].

2.3.1.2 MongoDB

Η MongoDB είναι ένα ελεύθερο και ανοιχτού κώδικα πρόγραμμα βάσης δεδομένων με βάση τα έγγραφα χρησιμοποιώντας έγγραφα τύπου JSON με σχήματα. Η MongoDB αναπτύσσεται από την MongoDB Inc. και δημοσιεύεται με συνδυασμό της Γενικής Δημόσιας Άδειας GNU Affero και της Άδειας Apache. Διατέθηκε για πρώτη φορά τον Φεβρουάριο του 2009. Είναι γραμμένο σε C, C++ και Javascript και είναι διαθέσιμο σε πολλά λειτουργικά συστήματα όπως Windows, Linux και MacOs.

Η MongoDB μπορεί να χρησιμοποιηθεί για αναζήτηση σε κάθε είδους έγγραφα. Ενώ τα έγγραφα αποθηκεύονται χρησιμοποιώντας ένα αρχικό σχήμα, αυτό μπορεί να αλλάζει ανά πάσα στιγμή καθιστώντας τη μια ιδανική λύση για αποθήκευση πληροφορίας που δεν ακολουθεί κάποιο συγκεκριμένο πρότυπο καθώς και για δεδομένα που συνεχώς μεταλλάσσονται, σε αντίθεση με μια σχεσιακή βάση δεδομένων όπου το σχήμα είναι αυστηρό, η επέκταση δυσκολότερη κάτι που έχει σαν αποτέλεσμα να ξοδεύεται λιγότερος χρόνος για την προετοιμασία των δεδομένων για τη βάση δεδομένων.

Η MongoDB μπορεί να εκτελεστεί μεταξύ γεωγραφικά κατανεμημένων κέντρων δεδομένων και των cloud εφαρμογών, παρέχοντας νέα επίπεδα διαθεσιμότητας και κλιμάκωσης. Χρησιμοποιεί τεμαχισμό (sharding) διανέμοντας δεδομένα σε πολλαπλές φυσικές κατατμήσεις που ονομάζονται shards κάτι που επιτρέπει στις εφαρμογές του MongoDB να αντιμετωπίζουν τους περιορισμούς υλικού ενός μόνο διακομιστή, όπως οι δυσχέρειες στη μνήμη RAM ή στο δίσκο I/O, χωρίς να προσθέτει πολυπλοκότητα στην εφαρμογή. Η MongoDB μπορεί επίσης να συντηρεί πολλαπλά αντίγραφα των δεδομένων

που ονομάζονται replica sets χρησιμοποιώντας. Ένα replica set είναι μια πλήρως αυτοσυντηρούμενο shard που βοηθά στην αποφυγή διακοπών της βάσης δεδομένων. Η ανακατασκευή του replication είναι πλήρως αυτοματοποιημένη, εξαλείφοντας την ανάγκη για χειρισμούς από τους διαχειριστές [26].

Οι βασικές δομές της MongoDB είναι:

- οι συλλογές (collection), που αποτελούν ένα σύνολο από έγγραφα, αντίστοιχο ενός πίνακα σε σχεσιακή βάση δεδομένων
- τα έγγραφα (documents), τα οποία αποτελούν τις εγγραφές σε μία συλλογη
- τα πεδία (fields), τα οποία είναι ζευγάρια κλειδιού – τιμής μέσα σε ένα έγγραφο, αντίστοιχο μια στήλης με την τιμής της σε μια σχεσιακή βάση.

Αξιοσημείωτοι χρήστες της MongoDB σήμερα είναι οι MetLife, eBay, Facebook, Adobe, Nokia και άλλοι [27].

2.3.2 Το Γραφοκεντρικό Μοντέλο

Μια βάση δεδομένων με γράφους είναι μια βάση δεδομένων που χρησιμοποιεί έννοιες της θεωρίας γραφημάτων για την αποθήκευση και την επαναφορά δεδομένων. Βασικές δομές της είναι οι κόμβοι, οι οποίοι αντιπροσωπεύουν οντότητες και είναι ισοδύναμοι με τις εγγραφές σε μια σχεσιακή βάση, οι άκρες (edges), που ονομάζονται επίσης σχέσεις (relationships) και είναι οι γραμμές που συνδέουν κόμβους με άλλους κόμβους, αντιπροσωπεύουν δηλαδή τη σχέση μεταξύ οντοτήτων καθώς και ιδιότητες που δίνονται είτε στους κόμβους είτε στις σχέσεις [28].

Η πρόσβαση σε κόμβους και σχέσεις σε μια βάση δεδομένων από γράφους είναι μια αποδοτική λειτουργία σταθερού χρόνου και επιτρέπει την γρήγορη προσπέλαση εκατομμύριων συνδέσεων ανά δευτερόλεπτο ανά. Ανεξαρτήτως του συνολικού μεγέθους των δεδομένων, το συγκεκριμένο μοντέλο υπερέχει στη διαχείριση εξαιρετικά συνδεδεμένων δεδομένων και πολύπλοκων ερωτημάτων. Για αυτό το λόγο έχει υπάρξει μεγάλο ενδιαφέρον για τη χρήση τους για την αξιοποίηση δεδομένων από κοινωνικά μέσα όπως επίσης και για την επεξεργασία δεδομένων σε επιχειρηματικούς κλάδους που περιλαμβάνουν σύνθετες σχέσεις και δυναμικά σχήματα.

Υπάρχουν αρκετές υλοποιήσεις γραφοκεντρικών βάσεων δεδομένων με τις πιο δημοφιλής να είναι οι Neo4j, OrientDB και ArangoDB [29].

2.3.2.1 Neo4j

Η Neo4j είναι ένα σύστημα διαχείρισης βάσεων δεδομένων που αναπτύχθηκε από τη Neo4j, Inc και είναι διαθέσιμο υπό την έκδοση "community edition" ανοιχτού κώδικα με άδεια GPL3, με αδειοδοτημένες επεκτάσεις υψηλής διαθεσιμότητας βάσει των όρων της γενικής δημόσιας άδειας Affero. Το Neo4j υλοποιείται σε Java και είναι προσβάσιμο από λογισμικό γραμμένο σε άλλες γλώσσες χρησιμοποιώντας τη γλώσσα Cypher Query Language (CQL) μέσω ενός τελικού σημείου συναλλαγής HTTP ή μέσω του δυαδικού πρωτοκόλλου bolt [30].

Η Neo4j είναι μια βάσης δεδομένων η οποία προσφέρει την προσπέλαση σε γράφους σε σταθερό χρόνο τόσο σε αναζήτηση σε βάθος όσο και σε αναζήτηση σε πλάτος λόγω της αποτελεσματικής αναπαράστασης κόμβων και σχέσεων. Επιτρέπει την κλιμάκωση έως και δισεκατομμύρια κόμβων σε μέτριο υλικό. Προσφέρει επίσης ένα ευέλικτο σχήμα γράφων

το οποίο μπορεί να προσαρμόζεται με την πάροδο του χρόνου, καθιστώντας δυνατή την υλοποίηση και χρήση νέων σχέσεων ανά πάσα στιγμή με στόχο την ταχύτερη προσαρμογή στην συνεχή αλλαγή των δεδομένων. Η Neo4j παρέχει πλήρεις λειτουργίες βάσης δεδομένων, συμπεριλαμβανομένης της συμβατότητας συναλλαγών ACID, της υποστήριξης χρήσης σε clusters και μηχανισμού ανάκτησης κατά το χρόνου εκτέλεσης [31].

Βασικές δομές της Neo4j είναι:

- οι κόμβοι (nodes), οι οποίοι περιγράφουν οντότητες
- οι σχέσεις (relationships), οι οποίες περιγράφουν πως οι οντότητες/κόμβοι συνδέονται μεταξύ τους
- οι ιδιότητες (properties), οι οποίες περιγράφουν κάθε κόμβο
- οι ετικέτες (labels), οι οποίες χρησιμοποιούνται για να ομαδοποιήσουν κόμβους ή σχέσεις

Αξιοσημείωτοι χρήστες της Neo4j σήμερα είναι οι Walmart, eBay, AirBnB, Financial Times, Monsanto, Nasa και άλλοι [32].

2.4 Το Μοντέλο Κλειδιού - Τιμής

Μια βάση δεδομένων κλειδιού – τιμής (key – value) είναι ένα παράδειγμα αποθήκευσης δεδομένων που έχει σχεδιαστεί για την αποθήκευση, την ανάκτηση και τη διαχείριση πινάκων συσχέτισης (associative array), μια δομή δεδομένων που είναι πιο γνωστή σήμερα ως λεξικό ή κατακερματισμός (hash). Τα λεξικά περιέχουν μια συλλογή αντικειμένων ή αρχείων, τα οποία με τη σειρά τους έχουν πολλά διαφορετικά πεδία μέσα σε αυτά, όπου το καθένα περιέχει δεδομένα. Αυτά τα αρχεία αποθηκεύονται και ανακτώνται χρησιμοποιώντας ένα κλειδί που προσδιορίζει με μοναδικό τρόπο την εγγραφή και χρησιμοποιείται για την ταχεία εύρεση των δεδομένων μέσα στη βάση δεδομένων.

Οι βάσεις δεδομένων κλειδιού – τιμής λειτουργούν με πολύ διαφορετικό τρόπο από τις πιο γνωστές σχεσιακές βάσεις δεδομένων οι οποίες προδιαμορφώνουν τη δομή δεδομένων στη βάση δεδομένων ως μια σειρά από πίνακες που περιέχουν πεδία με καλά καθορισμένους τύπους δεδομένων. Αντίθετα, τα συστήματα κλειδιού – τιμής αντιμετωπίζουν τα δεδομένα ως μία ενιαία συλλογή, η οποία μπορεί να έχει διαφορετικά πεδία για κάθε εγγραφή καθιστώντας τις βάσεις αυτές πιο ευέλικτες.

Μια από τις πιο γνωστές βάσης δεδομένων τύπου κλειδιού – τιμής είναι η Redis. Ο συγκεκριμένος τύπος συστήματος διαχείρισης βάσεων δεδομένων δε μελετήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας.

2.5 Η Γλώσσα Προγραμματισμού Python

Η Python είναι μια γενική, υψηλού επιπέδου διερμηνευμένη, γλώσσα προγραμματισμού που δημιουργήθηκε στις αρχές της δεκαετίας του 1990 από τον Guido van Rossum στο Stichting Mathematisch Centrum στην Ολλανδία ως διάδοχος της γλώσσας προγραμματισμού που ονομάζεται ABC [33].

Η γλώσσα Python είναι μια γλώσσα προγραμματισμού πολλαπλών παραδειγμάτων. Ο αντικειμενοστραφής προγραμματισμός και ο δομημένος προγραμματισμός υποστηρίζονται πλήρως ενώ υποστηρίζονται και πολλά από τα χαρακτηριστικά του λειτουργικού και του συναρτησιακού προγραμματισμό. Η Python χρησιμοποιεί δυναμική πληκτρολόγηση και

συνδυασμό συνδυασμού αναφορών και συλλέκτη απορριμμάτων ανίχνευσης κύκλων για διαχείριση μνήμης. Διαθέτει επίσης δυναμική ανάλυση ονομάτων (late binding), η οποία δεσμεύει τα ονόματα μεθόδων και μεταβλητών κατά την εκτέλεση του προγράμματος. Αντί να έχει όλες τις λειτουργίες του ενσωματωμένες στον πυρήνα του, η Python σχεδιάστηκε για να είναι εξαιρετικά επεκτάσιμη. Αυτή η συμπαγής διαμόρφωση έχει καταστήσει ιδιαίτερα δημοφιλές ως μέσο προσθήκης προγραμματιζόμενων διεπαφών σε υπάρχουσες εφαρμογές [34].

Οι μεγάλες εταιρίες και οργανώσεις που χρησιμοποιούν τη γλώσσα προγραμματισμού Python περιλαμβάνουν τη Βικιπαίδεια, το Google, το το Yahoo!, το CERN, το NASA, το Facebook, το Amazon, το Instagram και το Spotify. Ο ιστότοπος κοινωνικής δικτύωσης ειδήσεων Reddit είναι γραμμένος εξ ολοκλήρου στην Python [35].

3 Βιβλιογραφική Ανασκόπηση

3.1 θεωρητική Μελέτη

Αρκετές συγκρίσεις έχουν γίνει σε θεωρητικό επίπεδο οι οποίες προσπαθούν να εξηγήσουν το κενό που επιδιώκουν να καλύψουν οι μη σχεσιακές βάσεις δεδομένων και να επισημάνουν τα πλεονεκτήματα και τα μειονεκτήματα τους ανάλογα με το πρόβλημα προς επίλυση.

Σε μία από τις πρώτες τέτοιες προσπάθειες, ο Neal Leavitt [19], αφού επεξηγεί τα ήδη των μη σχεσιακών βάσεων δεδομένων, αναλύει τα πλεονεκτήματα τους, όπως η ταχύτητα και η απλότητα, ενώ παράλληλα επισημαίνει και τις ανησυχίες που εγείρουν οι εκπτώσεις που έχουν γίνει για να το επιτύχουν αυτό, όπως η συνέπεια και η αξιοπιστία. Επίσης, κάνει ειδική μνεία στην έλλειψη τεχνογνωσίας πάνω σε αυτές τις τεχνολογίες όπως και στο περιορισμένο οικοσύστημα συστημάτων και εργαλείων γύρω από αυτές.

Ο Robin Hecht και ο Stefan Jablonski στην εργασία με τίτλο "NoSQL Evaluation: A Use Case Oriented Survey" [36], αφού κάνουν μια παρουσίαση των διαθέσιμων NoSQL συστημάτων διαχείρισης βάσεων δεδομένων, πραγματοποιούν μια εκτεταμένη ανάλυση των περιπτώσεων χρήσης της κάθε τεχνολογίας, σε σχέση με τον τρόπο διεπαφής με τη βάση, τον τρόπο αναπαραγωγής (replication) της βάσης, τον κατακερματισμό καθώς και τον τρόπο αντιμετώπισης του ταυτοχρονισμού (concurrency). Το συμπέρασμα στο οποίο καταλήγουν είναι πως δεν υπάρχει καθολική σωστή επιλογή και πως το κάθε πρόβλημα και τα δεδομένα προς αποθήκευση και επεξεργασία επηρεάζουν διαφορετικά την επιλογή της σωστής λύσης. Σε παρόμοιο συμπέρασμα καταλήγουν τόσο η Maria Indrawan-Santiago [37], η οποία αναλύει το πως τα συστήματα εξυπηρετούν τις συναλλαγές (transactions) και τα ad-hoc ερωτήματα όσο και ο Santhosh Kumar Gajendran [38], ο οποίος μελετά τον τρόπο αποθήκευσης στα αποθηκευτικά μέσα και τις τεχνικές αναπαραγωγής των δεδομένων.

3.2 Πειραματική Μελέτη

Αρκετές μελέτες έχουν γίνει πάνω στην πειραματική αξιολόγηση και σύγκριση της συμπεριφοράς των συστημάτων διαχείρισης βάσεων δεδομένων. Κύριο αντικείμενο μελέτης είναι η σύγκριση μεταξύ σχεσιακών και μη σχεσιακών βάσεων δεδομένων σε διάφορα ήδη συνόλων δεδομένων καθώς και στις διάφορες ενέργειες πάνω σε αυτά τα δεδομένα. Οι Yishan Li και Sathiamoorthy Manoharan [38] μελέτησαν την ταχύτητα ανάγνωσης, εγγραφής, διαγραφής καθώς επίσης και την εκκίνηση των συστημάτων βάσεων δεδομένων για δεδομένα κλειδιού-τιμής. Μελετήσαν σχεσιακές βάσεις, στηλοκεντρίκες και εγγραφοκεντρικές βάσεις και το συμπέρασμα που προέκυψε είναι πως για το συγκεκριμένο τύπο δεδομένων οι NoSQL βάσεις δεδομένων συμπεριφέρονται καλύτερα, αν και επισημαίνεται πως σε μεγάλο φορτίο δραστηριότητας ακόμα και μερικές από αυτές τις βάσεις αντιμετωπίζουν δυσκολίες. Σε παρόμοιο συμπέρασμα για απλά σύνολα δεδομένων κατέληξαν ο Alexandru Boicea και λοιποί [39] συγκρίνοντας μια εγγραφοκεντρική βάση (MongoDB) με μια SQL βάση (Oracle), επισημαίνοντας ειδικότερα το πλεονέκτημα μιας μη σχεσιακής βάσης στην ταχύτητα εισαγωγής δεδομένων. Η γρήγορη εισαγωγή δεδομένων καθώς και η γρήγορη ανανέωση τους είναι κάτι που προκύπτει και από την εργασία “Comparing NoSQL MongoDB to an SQL DB” [40] στην οποία όμως επισημαίνεται πως οι σχεσιακές βάσεις υπερτερούν στην αναζήτηση πάνω σε χαρακτηριστικά που δεν είναι indexed όπως και σε σύνθετα aggregate queries. Τέλος όσον αφορά τη σύγκριση σχεσιακών

και μη βάσεων δεδομένων, οι Sarthak Agarwal και KS Rajan [41] σύγκριναν την Postgres (σχεσιακή) και την MongoDB (εγγραφοκεντρική) στην επιλογή δεδομένων με βάση τα γεωγραφικά κριτήρια και κατέληξαν πως η μη σχεσιακή βάση συμπεριφέρεται τάξεις καλύτερα από την σχεσιακή.

Επιπρόσθετα με τα προαναφερθέντα, υπάρχει ένα κομμάτι τις βιβλιογραφίας που μελετά μόνο τις μη σχεσιακές βάσεις και τις διάφορες υλοποιήσεις τους. Η Veronika Abramova και λοιποί [42] αφού μελέτησαν διάφορες μη σχεσιακές βάσεις δεδομένων διαφορετικού τύπου (εγγραφοκεντρικές, στηλοκεντρικές. κλειδιού-τιμής) πάνω σε διαφορετικά ήδη φόρτου εργασίας, κατέληξαν στο συμπέρασμα ότι σε γενικές γραμμές, οι μη σχεσιακές βάσεις μπορούν να χωριστούν σε δύο κατηγορίες, σε βάσεις που είναι βελτιστοποιημένες για απλές αναγνώσεις δεδομένων και σε βάσεις βελτιστοποιημένες για συνεχής αλλαγές πάνω στα δεδομένα (updates). Η διαφορά αυτή κυρίως οφείλεται στο γεγονός πως οι βάσεις που είναι καλύτερες για ανάγνωση δεδομένων χρησιμοποιούν σε μεγάλο βαθμό την πτητική μνήμη αποθήκευσης του υπολογιστή, όπως είναι η μνήμη ταχείας προσπέλασης (RAM), η οποία είναι αρκετά πιο γρήγορη, αν και πιο ακριβή. Τέλος, η εργασία με τον τίτλο “Comparison Between Performanceof Various Database Systemsfor Implementing a Language Corpus” [43] μελετάει την συμπεριφορά των μη σχεσιακών βάσεων δεδομένων στην αναζήτηση κειμένου. Τα συμπεράσματα που προέκυψαν είναι πως μια εγγραφοκεντρική βάση που βασίζεται στη μηχανή Lucene, όπως η Solr και η Elasticsearch, είναι τάξεις ταχύτερη στην εισαγωγή δεδομένων κειμένου κάτι που δεν ισχύει για την εξαγωγή δεδομένων, στην οποία μια βάση στηλοκεντρική, όπως είναι η Cassandra, δείχνει σταθερά καλή συμπεριφορά ανεξαρτήτως ερωτήματος ενώ οι γραφοκεντρικές βάσεις σα τη Neo4j είναι τελείως ακατάλληλες.

4 Μεθοδολογία και Αποτελέσματα

4.1 Σύγκριση στην παρούσα διπλωματική

Σκοπός της παρούσας διπλωματικής εργασίας είναι η σύγκριση της απόδοσης των συστημάτων MySQL, MongoDB, Elasticsearch και Neo4j στην ανάκτηση δεδομένων (select queries). Η σύγκριση θα λάβει χώρα σε γεωγραφικά δεδομένα και δεδομένα κειμένου για διαφορετικά μεγέθη όγκου δεδομένων και για διαφορετικά μεγέθη συνόλων προς ανάκτηση. Η σύγκριση θα πραγματοποιηθεί σε δύο διαφορετικές συνθήκες, σε βάση με δεδομένα στην κρυφή μνήμη (cached data) και σε βάση χωρίς δεδομένα στην κρυφή μνήμη.

4.2 Γεωγραφικά δεδομένα

Τα γεωγραφικά δεδομένα ορίζονται στη σειρά προτύπων ISO/TC 211 ως δεδομένα και πληροφορίες που έχουν έμμεση ή ρητή συσχέτιση με μια θέση σχετικά με τη Γη. Σύμφωνα με το πρότυπο ISO 6709 [44], το διεθνές πρότυπο για την απεικόνιση γεωγραφικού πλάτους, γεωγραφικού μήκους και υψομέτρου γεωγραφικών σημείων, ένα γεωγραφικό σημείο προσδιορίζεται από τα ακόλουθα τέσσερα στοιχεία:

- Η πρώτη οριζόντια συντεταγμένη (y), όπως το γεωγραφικό πλάτος (αρνητικός αριθμός νότια από τον ισημερινό και θετικό βόρειο από τον ισημερινό)
- Δεύτερη οριζόντια συντεταγμένη (x), όπως το γεωγραφικό μήκος (αρνητικές τιμές δυτικά του Πρώτου Μεσημβρινού και θετικές τιμές ανατολικά του Πρώτου Μεσημβρινού)
- Κάθετη συντεταγμένη, δηλαδή ύψος ή βάθος (προαιρετικό)
- Αναγνώριση του συστήματος αναφοράς συντεταγμένων (CRS) (προαιρετικό)

Στα πλαίσια της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκαν τα δύο πρώτα από τα τέσσερα στοιχεία για τον προσδιορισμό ενός γεωγραφικού στοιχείου, δηλαδή το γεωγραφικό πλάτος και το γεωγραφικό μήκος.

4.2.1 Πηγές γεωγραφικών δεδομένων

Για την απόκτηση των γεωγραφικών δεδομένων που χρησιμοποιήθηκαν στην παρούσα διπλωματική, χρησιμοποιήθηκαν διάφορες πήγες με κύρια πηγή το εργαλείο Open Street Map καθώς και σε λιγότερο βαθμό το Google Maps μέσω του Google Maps API. Τα δεδομένα που εξήχθησαν αποτελούν σημεία ενδιαφέροντος στους παραπάνω χάρτες όπως μουσεία, καφετέριες, εστιατόρια προσφέροντας πέρα από τις συντεταγμένες και μια πληροφορία για το είδος του κάθε σημείο που χρησιμοποιήθηκε

4.2.2 Αποθήκευση γεωγραφικών δεδομένων

Συνολικά εξήχθησαν περίπου 150.000 σημεία ενδιαφέροντος. Λόγω του περιορισμένου αριθμού σημείων ενδιαφέροντος, για να εξεταστούν πολλαπλά μεγέθη βάσεων δεδομένων, τα δεδομένα επαναχρησιμοποιήθηκαν με σκοπό στο τέλος να δημιουργηθούν τέσσερις διαφορετικές βάσεις, με 150.000, 500.000, 2.000.000 και 8.000.000 σημεία ενδιαφέροντος η κάθε μια. Για κάθε ένα μέγεθος χρησιμοποιήθηκαν δύο διαφορετικές βάσεις, μια με χρήση ευρετηρίων (index) και μία χωρίς.

Τα δεδομένα αρχικά εξήχθησαν από τις διαθέσιμες πηγές χρησιμοποιώντας τη γλώσσα προγραμματισμού python σε συνδυασμό με τις διαθέσιμες διεπαφές σύνδεσης (APIs) που

προσφέρουν τα προαναφερθέντα εργαλεία και αρχικώς αποθηκεύτηκαν σε όλες τις βάσεις δεδομένων. Στη συνέχεια, με τη χρήση εργαλείων γραμμής εντολών (command line interfaces) που προσφέρουν οι ίδιες οι βάσεις, πραγματοποιήθηκε η αναπαραγωγή των δεδομένων στα υπόλοιπα μεγέθη βάσεων.

Για το μήκος και το πλάτος των γεωγραφικών συντεταγμένων χρησιμοποιήθηκαν αριθμοί δεκαδικοί αριθμοί με ακρίβεια 7 δεκαδικών ψηφίων.

4.2.2.1 Αποθήκευση σε MySQL

Για την αποθήκευση των δεδομένων σε MySQL χρησιμοποιήθηκε το παρακάτω σχήμα:

Πίνακας με τις συντεταγμένες κάθε μέρους

```
CREATE TABLE places (
    id          int          NOT NULL AUTO_INCREMENT,
    name        varchar(25) CHARACTER SET utf8mb4  default NULL,
    lon         real(9,7)    default NULL,
    lat         real(9,7)    default NULL,
    PRIMARY KEY (id)
);
```

Πίνακας με όλες τις πιθανές κατηγορίες

```
CREATE TABLE categories (
    id              int                 not null AUTO_INCREMENT,
    category        varchar(200)        CHARACTER SET utf8mb4   not null,
    PRIMARY key (id),
    UNIQUE (category)
);
```

Πίνακας που συνδέει τα μέρη με την κατηγορία στην οποία ανήκουν

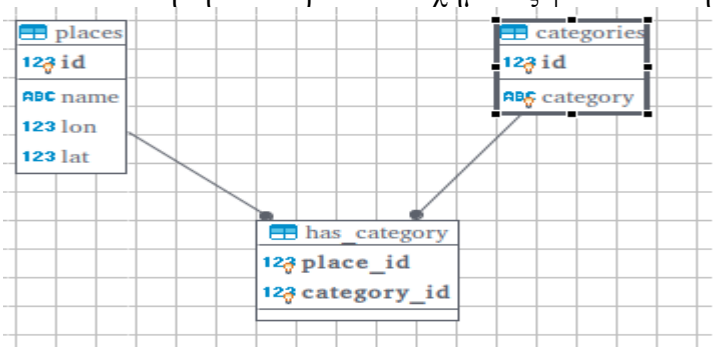
```
CREATE TABLE has_category (
    place_id        int not null,
    category_id     int not null,
    primary key (place_id, category_id),
    foreign key (place_id) REFERENCES places(id),
    foreign key (category_id) REFERENCES categories(id)
);
```

Για τις περιπτώσεις που χρησιμοποιήθηκε index, αυτό έγινε με τον παρακάτω τρόπο

Δημιουργία ευρετηρίου

```
CREATE INDEX lon_lat_idx on places(lon, lat);
```

Η οπτικοποίηση του παραπάνω σχήματος φαίνεται στην εικόνα που ακολουθεί



Εικόνα 1 MySQL σχήμα γεωγραφικών δεδομένων

4.2.2.2 Αποθήκευση σε MongoDB

Η αποθήκευση των γεωγραφικών δεδομένων έγινε με χρήση μιας συλλογής (collection) από έγγραφα, τα οποία περιέχουν την πληροφορία τόσο για το γεωγραφικό σημείο όσο και για την κατηγορία στην οποία ανήκει. Ένα παράδειγμα τέτοιου εγγράφου φαίνεται παρακάτω:

Παράδειγμα εγγράφου της συλλογής places

```
>>> db.places.countOne().pretty()
{
    "_id" : ObjectId("587f95ea8bbdc503ffcc6942"),
    "name" : {
        "greek" : "Ποσειδωνεία",
        "english" : "Ποσειδωνεία"
    },
    "category" : {
        "greek" : "Ελληνικό Εστιατόριο",
        "english" : "Greek Restaurant"
    },
    "loc" : {
        "type" : "Point",
        "coordinates" : [
            24.055531,
            37.714644
        ]
    }
}
```

Στις βάσεις με στις οποίες χρησιμοποιήθηκε ευρετήριο αναζήτησης, η εφαρμογή του έγινε χρησιμοποιώντας την παρακάτω εντολή:

Δημιουργία ευρετηρίου

```
db.places.createIndex( { loc : "2dsphere" } );
```

4.2.2.3 Αποθήκευση σε Elasticsearch

Για την αποθήκευση σε Elasticsearch χρησιμοποιήθηκε ένα ευρετήριο (index) με την ονομασία geodata, στο οποίο υπήρχε μόνο ένας τύπος εγγράφων (type), το places. Ένα παράδειγμα τέτοιου εγγράφου είναι το ακόλουθο:

Παράδειγμα εγγράφου τύπου places στο ευρετήριο geodata

```
{
"_index" : "geodata",
{
  "_index": "geodata",
  "_type": "places",
  "_id": "AWJt02r3nCB8s_vYgzFW",
  "_score": 1,
  "_source": {
    "rating": 0,
    "categories": [
      {
        "foursquareID": "4bf58dd8d48988d1ae941735",
        "categoryList": [
          "Education",
          "College & University",
          "Εκπαίδευση",
          "Κολέγιο & Πανεπιστήμιο"
        ],
        "name": {
          "greek": "Πανεπιστήμιο",
```

```
          "english": "University"
        },
        "weight": 8,
        "categoryID": "58984a8e879fad040d8ce6ba"
      }
    ],
    "name": "ALBA Graduate Business School",
    "seenDetails": 308,
    "location": [
      23.778018951416016,
      37.81781906766782
    ]
  }
}
```

4.2.2.4 Αποθήκευση σε Neo4j

Η αποθήκευση σε Neo4j έγινε χρησιμοποιώντας τους παρακάτω κόμβους:

Παράδειγμα δημιουργίας κόμβου με ετικέτα places

```
CREATE (:places {
id: <some value>,
lon: <some value>,
lat: <some value>
});
```

Παράδειγμα δημιουργίας κόμβου με ετικέτα categories

```
CREATE (:categories {
id: <some value>,
category: <some value>
});
```

Προσωρινοί κόμβοι με την ετικέτα has_category_aux χρησιμοποιήθηκαν για την ευκολότερη δημιουργία της σχέσης IS_OF_CATEGORY που ενώνει τους κόμβους των places με αυτούς με ετικέτα categories. Μετά τη δημιουργία της σχέση ο πίνακας διεγράφη.

Προσωρινοί κόμβοι

```
CREATE (:has_category_aux
{place_id: <some value>,
category_id: <some value>
});
```

Δημιουργία σχέση IS_OF_CATEGORY

```
MATCH (p:places), (c:categories), (hc:has_category_aux)
WHERE p.id = hc.place_id AND c.id=hc.category_id
CREATE (p)-[r:IS_OF_CATEGORY]->(c);
```

Για τα ερωτήματα με ευρετήριο χρησιμοποιήθηκε η παρακάτω εντολή

Δημιουργία ευρετηρίου στον κόμβο places

```
CREATE INDEX ON :places(lon, lat)
```

4.3 Δεδομένα Κειμένου

Τα δεδομένα κειμένου ορίζονται ως τα δεδομένα που αποτελούνται από έναν ή περισσοτέρους χαρακτήρες (γράμματα της αλφαβήτου) οι οποίοι αποθηκεύονται στη μνήμη των υπολογιστών με κάποια ειδική κωδικοποίηση όπως η ASCII κωδικοποίηση ή η UTF.

4.3.1 Πηγές κειμένων

Με τη σύγχρονη και ραγδαία ανάπτυξη του διαδικτύου και την ευκολία με την οποία μπορεί ο καθένας να δημοσιεύσει κάποιο κείμενο, οι πηγές από τις οποίες μπορούν να αντληθούν τα δεδομένα κειμένου για την παρούσα διπλωματική είναι σχεδόν ανεξάντλητες.

Στα πλαίσια της παρούσας διπλωματικής τελικά επιλέχθηκε σα πηγή κειμένου η ιστοσελίδα wikipedia.org λόγω του μεγάλου όγκου κειμένων, όσο και της πιθανής συνάφειας μεταξύ των κειμένων κάτι το οποίο καθιστά τα εξαγώγιμα δεδομένα ιδανικά για τα ζητούμενα της παρούσας εργασίας. Στα πλαίσια της εργασίας ενδιαφερόμασταν μόνο για την αναζήτηση πάνω σε δεδομένα κειμένου και όχι η ποιότητα των δεδομένων.

4.3.2 Αποθήκευση κειμένων

Τα κείμενα εξήχθησαν χρησιμοποιώντας έναν αυτοσχέδιο πρόγραμμα ανίχνευσης διαδικτύου (web crawler) γραμμένο σε python, το οποίο με βάση έναν αρχικό όρο (search term) ξεκίναγε την εξαγωγή κειμένου στη wikipedia από το λήμμα του συγκεκριμένου όρου, και συνέχιζε σε τυχαίες σελίδες-λήμματα όρων που βρέθηκαν μέσα στο αρχικό λήμμα. Η διαδικασία σταματούσε όταν ο όγκος των κειμένων έφτανε ένα συγκεκριμένο αριθμό megabyte στη βάση MySQL. Στη συνέχεια έγινε η εξαγωγή των δεδομένων από τη mysql σε όλα τα υπόλοιπα συστήματα διαχείρισης βάσεων δεδομένων με χρήση των εργαλείων γραμμής εντολών που προσφέρουν τα συστήματα αυτά.

Δημιουργήθηκαν 4 διαφορετικά μεγέθη, 250MB, 500MB, 1GB και 5GB. Για τη δημιουργία της βάσης με μέγεθος 5GB χρησιμοποιήθηκε τυχαίο κείμενο για τα λήμματα, με κάθε λήμμα να αποτελείται από αριθμό χαρακτήρων ίδιο με το μέσος όρο χαρακτήρων για τα κείμενα της βάσης του 1GB. Επιλέχθηκε η παραπάνω διαδικασία για την εξοικονόμηση πόρων και χρόνου.

Για κάθε ένα μέγεθος χρησιμοποιήθηκαν δύο διαφορετικές βάσεις, μια με χρήση ευρετηρίων (index) και μία χωρίς.

4.3.2.1 Αποθήκευση σε MySQL

Για την αποθήκευση των δεδομένων σε MySQL, δημιουργήθηκαν τέσσερις διαφορετικές βάσεις δεδομένων, μια για κάθε ένα από τα διαφορετικά μεγέθη δεδομένων που εξετάστηκαν και χρησιμοποιήθηκε το παρακάτω σχήμα:

Πίνακας αποθήκευσης κειμένων

```
CREATE TABLE Posts (
    id INT(4) NOT NULL AUTO_INCREMENT,
    title VARCHAR(500) NOT NULL,
    content LONGTEXT,
    date_inserted DATETIME,
    PRIMARY KEY (id),
    UNIQUE (title)
);
```

Για τη βάση με το ευρετήριο κειμένου χρησιμοποιήθηκε το παρακάτω SQL ερώτημα για την κατασκευή του:

Δημιουργία ευρετηρίου κειμένου

```
CREATE FULLTEXT INDEX POST_CONTENT ON POSTS(CONTENT);
```

4.3.2.2 Αποθήκευση σε MongoDB

Η αποθήκευση των δεδομένων κειμένου έγινε με χρήση μιας συλλογής (collection) από έγγραφα τα οποία περιέχουν την απαιτούμενη πληροφορία. Χρησιμοποιήθηκαν τέσσερις διαφορετικές βάσεις, μία για κάθε εξεταζόμενο μέγεθος δεδομένων. Ένα παράδειγμα τέτοιου εγγράφου φαίνεται παρακάτω:

Παράδειγμα εγγράφου της συλλογής Posts

```
>>> db.Posts.countOne().pretty()
{
    "_id" : 1,
    "title" : "National Basketball Association",
    "content" : "The National Basketball Association (NBA)a men's
professional..."
    "date_inserted" : ISODate("2018-03-15T00:00:00Z")
}
```

Στις βάσεις με στις οποίες χρησιμοποιήθηκε ευρετήριο αναζήτησης, η εφαρμογή του έγινε χρησιμοποιώντας την παρακάτω εντολή:

Δημιουργία ευρετηρίου

```
db.Posts.createIndex({content:"text"})
```

Η μεταφορά των δεδομένων από τη MySQL στη MongoDB έγινε με τη χρήση του προγράμματος mongify μέσω της παρακάτω εντολής:

Μεταφορά δεδομένων από MySQL σε MongoDB

```
mongify process database.config translation_file.rb
```

Στην παραπάνω εντολή το αρχείο database.config είναι το αρχείο στο οποίο περιγράφονται η βάση από την οποία θα αντιγραφούν τα δεδομένα καθώς και η βάση στην οποία θα μεταφερθούν. Το αρχείο translation_file.rb είναι το αρχείο στο οποίο περιγράφει με ποιο τρόπο θα μεταφερθούν τα δεδομένα από τους πίνακες της MySQL στις συλλογές της MongoDB καθώς και πιθανούς μετασχηματισμούς οι οποίοι ενδέχεται να χρειαστούν.

Αρχειό database.config

```
sql_connection do
  adapter     "mysql2"
  host        "localhost"
  username    "root"
  password    "qwerty"
  database    "wikipedia"
  batch_size  20000
end

mongodb_connection do
  host        "localhost"
  database    "wikipedia"
end
```

Αρχείο translation_file.rb

```
table "Posts" do
    column "id", :key, :as => :integer
    column "title", :string
    column "content", :text
    column "date_inserted", :datetime
    before_save do |row|
        row._id = row.delete('pre_mongified_id')
    end
end
```

4.3.2.3 Αποθήκευση σε Elasticsearch

Για την αποθήκευση σε Elasticsearch χρησιμοποιήθηκε ένα ευρετήριο (index) με την ονομασία wiki, στο οποίο υπήρχε μόνο ένας τύπος εγγράφων (type), το Posts. Ένα παράδειγμα τέτοιου εγγράφου είναι το ακόλουθο:

Παράδειγμα εγγράφου τύπου Posts στο ευρετήριο wiki

```
{
  "_index": "wiki",
  "_type": "Posts",
  "_id": "44",
  "_score": 1,
  "_source": {
    "title": "The Headies Award for Next Rated",
    "content": "The Headies Award…
    "date_inserted": {
      "$date": 1521072000000
    }
  }
```

4.3.2.4 Αποθήκευση σε Neo4j

Η αποθήκευση σε Neo4j έγινε χρησιμοποιώντας τον παρακάτω κόμβου:

Παράδειγμα κόμβου τύπου Posts

```
CREATE (:Posts {
   title: <some_value>,
   content: <some_value>,
   date_inserted: <some_value>
})
```

Η δημιουργία ευρετηρίου κειμένου έγινε με την παρακάτω εντολή

Δημιουργία ευρετηρίου στον κόμβο Posts

```
CREATE INDEX ON :Posts(content);
```

4.4 Περιβάλλον εκτέλεσης πειραμάτων

Για την εκτέλεση των πειραμάτων έπρεπε να επιλεγεί μία πλατφόρμα η οποία θα καθιστούσε τις μετρήσεις όσο το δυνατόν ανεξάρτητες και ανεπηρέαστες από εξωτερικούς παράγοντες. Για το λόγο αυτό επιλέχθηκε η χρήση μιας εικονικής μηχανής (virtual machine) μέσα στην οποία θα πραγματοποιηθούν όλες οι πειραματικές δοκιμές αυτόνομα, μειώνοντας την επιρροή άλλων διεργασιών που ενδεχομένως θα έτρεχαν παράλληλα σε ένα μη απομονωμένο μηχάνημα, όπως ένας προσωπικός υπολογιστής.

Για την δημιουργία της εικονικής μηχανής επιλέχτηκε η υπηρεσία νέφους (cloud) της Google. Στο google cloud δημιουργήθηκε μία εικονική μηχανή στην οποία έγιναν οι εγκαταστάσεις όλων των συστημάτων διαχείρισης βάσεων δεδομένων, συλλέχθηκαν τα δεδομένα και έτρεξαν όλες οι πειραματικές εκτελέσεις.

4.4.1 Google cloud VM specifications

Με την πλατφόρμα Google Cloud Platform (GCP), μπορούν να δημιουργηθούν, να δοκιμαστούν και να αναπτυχθούν εφαρμογές στην εξαιρετικά κλιμακούμενη και αξιόπιστη υποδομή της Google. Το Google Compute Engine επιτρέπει να δημιουργηθούν και να εκτελεστούν εικονικές μηχανές στην υποδομή Google. Το Compute Engine προσφέρει

κλιμάκωση, απόδοση και αξία που δίνει τη δυνατότητα για δημιουργία μεγάλων cluster υπολογιστών στην υποδομή της Google.

Για τις ανάγκες της παρούσας διπλωματικής δημιουργήθηκε μια εικονική μηχανή, compute engine με την ορολογία της Google το οποίο είχε τα παρακάτω χαρακτηριστικά:

- 4 CPUs χρονισμένοι στα 2.2GHz
- 15GB RAM
- 500GB μηχανικό σκληρό δίσκο
- Λειτουργικό Ubuntu 16.04 LTS

4.4.2 Εκδόσεις βάσεων δεδομένων

Στη παραπάνω εικονική μηχανή έγινε και η εγκατάσταση των τεσσάρων συστημάτων διαχείρισης βάσεων δεδομένων. Χρησιμοποιήθηκαν η παρακάτω εκδόσεις για κάθε σύστημα:

- MySQL Ver 14.14 Distrib 5.7.21, for Linux (x86_64) using EditLine wrapper
- MongoDB db version v3.4.9
- Neo4j version 3.3.0
- Elasticsearch version 5.6.3 με lucene version 6.6.1

4.4.3 Παραμετροποίηση Εγκαταστάσεων

Για τη σωστή πειραματική εκτέλεση και σύγκριση ήταν αναγκαίο να αλλαχθούν οι προκαθορισμένες παραμετροποιήσεις (default configuration) των συστημάτων βάσεων δεδομένων. Οι αλλαγές που πραγματοποιήθηκαν ήταν οι ακόλουθες:

- MySQL
    - collation_server=utf8_unicode_ci, για καλύτερη ταξινόμηση και σύγκριση, η οποία ταξινομείται με ακρίβεια σε ένα πολύ ευρύ φάσμα γλωσσών
    - lower_case_table_names=1, για ευκολία στην χρήση των πινάκων
    - innodb-ft-min-token-size=1, ώστε ο κατακερματισμός των λέξεων στο ευρετήριο πλήρους κειμένου να γίνεται με μικρότερη μονάδα ένα γράμμα της αλφαβήτου όπως γίνεται στα άλλα συστήματα προς εξέταση. Η προκαθορισμένη τιμή ήταν 3
- MongoDB
    - Καμία αλλαγή
- Elasticsearch
    - http.max_content_length: 1gb, για τη μεταφορά μεγάλου όγκου δεδομένων με χρήση του πρωτοκόλλου http.
- Neo4j
    - dbms.memory.heap.max_size=13g, για την καλύτερη λειτουργία σε μεγαλύτερα μεγέθη βάσεων δεδομένων

4.5 Ερωτήματα προς τις βάσεις

Κάθε σύστημα διαχείρισης βάσεων δεδομένων προσφέρει διαφορετικό τρόπο επικοινωνίας από και προς τη βάση δεδομένων για την προσπέλαση και επεξεργασία των δεδομένων.

Η MySQL χρησιμοποιεί την Standard Query Language για την εκτέλεση ενεργειών πάνω στα δεδομένα όσο και στο σύστημα διαχείρισης της.

Η MongoDB προσφέρει έναν διαδραστικό εξυπηρετητή διασύνδεσης με τη βάση δεδομένων ο οποίος ονομάζεται MongoShell και επιτρέπει την εκτέλεση ερωτημάτων στις συλλογές εγγράφων γραμμένα στη γλώσσα προγραμματισμού Javascript.

Η Elasticsearch προσφέρει μια διεπαφή εξυπηρέτησης βασισμένη στην αρχιτεκτονική REST (REST API). Αυτό δίνει τη δυνατότητα επικοινωνίας με το σύστημα διαχείρισης μέσω του πρωτοκόλλου μεταφοράς υπερκειμένου (http) και των μεθόδων του. Στην περίπτωση της αναζήτησης στη βάση δεδομένων αρκεί η χρήση της μεθόδου GET ή POST του HTTP μέσω ενός οποιουδήποτε πελάτη HTTP, όπως ένας φυλλομετρητης.

Η Neo4j προσφέρει μια ξεχωριστή γλώσσα διεπαφής με τη βάση δεδομένων, την Cypher Query Language, μια γλώσσα που έχει σχεδιαστεί για να είναι βασισμένη στην αγγλική πεζογραφία και την εικονογραφία για να κάνει τη σύνταξη οπτική και εύκολα κατανοητή.

4.5.1 Ερωτήματα σε κείμενο

Τα ερωτήματα προς τις βάσεις δεδομένων με δεδομένα κειμένου αφορούσαν την ανάκτηση δεδομένων κειμένου με βάση τα παρακάτω κριτήρια:

T1 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο.
Για παράδειγμα αναζητώντας το γράμμα ‘y’ επιστρέφονται όλες οι εγγραφές που το περιέχουν.

T2 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα.
Για παράδειγμα αναζητώντας το γράμμα ‘y’ επιστρέφεται η εγγραφή “Variables x y z” αλλά όχι η εγγραφή “new york”.

T3 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στο τέλος μιας λέξης.
Για παράδειγμα κάνοντας αναζήτηση για το γράμμα ‘y’ επιστρέφονται εγγραφές όπως “why this is good…” αλλά όχι η εγγραφή “new york”.

T4 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης.
Για παράδειγμα κάνοντας αναζήτηση για το γράμμα ‘y’ επιστρέφονται εγγραφές όπως “new york” αλλά όχι η εγγραφή “why this is good…”.

T5 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο κείμενο.
Για παράδειγμα αναζητώντας μια εγγραφή που περιέχει τα τρία γράμματα ‘new’ επιστρέφονται εγγραφές όπως “new York”, “newton”.

T6 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης.
Για παράδειγμα αναζητώντας μια εγγραφή που περιέχει τα τρία γράμματα ‘new’ επιστρέφονται εγγραφές όπως “…renew…”.

T7 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στην αρχή μιας λέξης.
Για παράδειγμα αναζητώντας μια εγγραφή που περιέχει τα τρία γράμματα ‘new’ επιστρέφονται εγγραφές όπως “newton was….”.

T8 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη, αυτούσια ή μέρος σύνθετης λέξης..

Για παράδειγμα αναζητώντας τη λέξη “York” επιστρέφονται εγγραφές όπως “…New York…”, “..Yorkshire…”, “ …NewYorkCity…”.

T9 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια.
Για παράδειγμα αναζητώντας τη λέξη “York” επιστρέφονται εγγραφές όπως “…New York…” αλλά όχι εγγραφές όπως “..Yorkshire…”, “ …NewYorkCity…”.

T10 Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση.
Για παράδειγμα αναζητώντας τη φράση “New York City” επιστρέφονται εγγραφές που περιέχουν τη φράση αυτούσια στο κείμενο τους.

T11 Εύρεση εγγραφών στις οποίες περιέχεται μία λέξη τουλάχιστον 35 χαρακτήρων.

4.5.1.1 Ερωτήματα κειμένου σε MySQL

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης MySQL. Λόγω της φύσης του fulltext index της MySQL πολλά από τα ερωτήματα διαφέρουν στον τρόπο γραφής τους στην απλή αναζήτηση σε σχέση με την αναζήτηση σε ευρετήριο κειμένου. Για το λόγο αυτό τα query παρακάτω παρουσιάζονται σε ζεύγη ώστε να είναι εμφανής η διαφορά, όπου αυτή υπάρχει.

Να σημειωθεί ότι όπου δεν υπάρχει διαφορετικό ερώτημα για την έκδοση με το ευρετήριο κειμένου αυτό σημαίνει ότι το ερώτημα δεν υποστηρίζεται από το ευρετήριο κειμένου της MySQL και ανταυτού χρησιμοποιήθηκε πάλι το ερώτημα χωρίς το ευρετήριο.

T1 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%Y%";
```

T2a Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα (έκδοση για χρήση χωρίς ευρετήριο)

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%y%"
AND p.content REGEXP '[[:<:]]y[[:>:]]';
```

T2b Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα (έκδοση για χρήση με ευρετήριο)

```
select
    COUNT(*)
from
    posts p
where
    MATCH (p.content) against('y' in boolean mode);
```

T3 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στο τέλος μιας λέξης

```
select
    COUNT(*)
from
```

```
    posts p
where
    p.content LIKE "%y%"
AND p.content REGEXP 'y[[:>:]]';
```

T4a Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης (χωρίς χρήση ευρετηρίου)

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%y%"
AND p.content REGEXP '[[:<:]]y';
```

T4b Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης (χρήση ευρετηρίου)

```
select
    COUNT(*)
from
    posts p
where
    MATCH (p.content) against('y*' in boolean mode);
```

T5 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο κείμενο

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%new%";
```

T6α Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης (χωρίς ευρετήριο)

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%new%"
AND p.content REGEXP '[[:<:]]new';
```

T6β Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης (με ευρετήριο)

```
select
    COUNT(*)
from
    posts p
where
    MATCH (p.content) against('new*' in boolean mode);
```

T7 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στην αρχή μιας λέξης

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%new%"
AND p.content REGEXP 'new[[:>:]]';
```

T8 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη, αυτούσια ή μέρος σύνθετης λέξης

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%york%"
```

T9α Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%york%"
AND p.content REGEXP '[[:<:]]york[[:>:]]';
```

T9β Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια (χρήση ευρετηρίου)

```
select
    COUNT(*)
from
    posts p
where
    MATCH (p.content) against('york' in boolean mode)
```

T10α Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
select
    COUNT(*)
from
    posts p
where
    p.content LIKE "%New York City%";
```

T10β Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
select
    COUNT(*)
from
    posts p
where
    MATCH (p.content) against('"New York City"' in boolean mode);
```

T11 Εύρεση εγγραφών στις οποίες περιέχεται μία λέξη τουλάχιστον 35 χαρακτήρων

```
select
    COUNT(*)
from
    posts p
where
    p.content REGEXP '[a-zA-Z]{35}';
```

4.5.1.2 Ερωτήματα κειμένου σε MongoDB

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης MongoDB. Λόγω της φύσης του fulltext index της MongoDB πολλά από τα ερωτήματα διαφέρουν στον τρόπο γραφής τους στην απλή αναζήτηση σε σχέση με την αναζήτηση σε ευρετήριο κειμένου. Για το λόγο αυτό τα query παρακάτω παρουσιάζονται σε ζεύγη ώστε να είναι εμφανής η διαφορά, όπου αυτή υπάρχει.

Να σημειωθεί ότι όπου δεν υπάρχει διαφορετικό ερώτημα για την έκδοση με το ευρετήριο κειμένου αυτό σημαίνει ότι το ερώτημα δεν υποστηρίζεται από το ευρετήριο κειμένου της MongoDB και ανταυτού χρησιμοποιήθηκε πάλι το ερώτημα χωρίς το ευρετήριο.

T1 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο

```
db.Posts.count(
    {
      content: {$regex: /y/, $options: 'i'}
    }
)
```

T2a Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα (έκδοση για χρήση χωρίς ευρετήριο)

```
db.Posts.count(
    {
      content: {$regex: /\by\b/, $options: 'i'}
    })
```

T2b Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα (έκδοση για χρήση με ευρετήριο)

```
db.Posts.count(
    {
       "$text": {"$search": "y"}
    }
)
```

T3 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στο τέλος μιας λέξης

```
db.Posts.count(
    {
      content: {$regex: /y\b/, $options: 'i'}
    }
)
```

T4 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης (χωρις χρήση ευρετηρίου)

```
db.Posts.count(
    {
      content: {$regex: /\by/, $options: 'i'}
    }
)
```

T5 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο κείμενο

```
db.Posts.count(
    {
      content: {$regex: /new/, $options: 'i'}
    }
)
```

T6 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης (χωρίς ευρετήριο)

```
db.Posts.count(
    {
      content: {$regex: /\bnew/, $options: 'i'}
    }
)
```

T7 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στην αρχή μιας λέξης

```
db.Posts.count(
```

```
    {
      content: {$regex: /new\b/, $options: 'i'}
    }
)
```

T8 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη, αυτούσια ή μέρος σύνθετης λέξης

```
db.Posts.count(
    {
      content: {$regex: /york/, $options: 'i'}
    }
)
```

T9α Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια

```
db.Posts.count(
    {
      content: {$regex: /\byork\b/, $options: 'i'}
    }
)
```

T9β Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια (χρήση ευρετηρίου)

```
db.Posts.count(
    {
      "$text": {"$search": "york"}
    }
)
```

T10α Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
db.Posts.count(
    {
      content: {$regex: /\bnew york city\b/, $options: 'i'}
    }
)
```

T10β Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
db.Posts.count(
   {
       "$text": {"$search": "\"new york city\""}
   }
)
```

T11 Εύρεση εγγραφών στις οποίες περιέχεται μία λέξη τουλάχιστον 35 χαρακτήρων

```
db.Posts.count(
    {
      content: {$regex: /\b[a-zA-Z]{35}/, $options: 'i'}
    }
)
```

4.5.1.3 Ερωτήματα κειμένου σε Elasticsearch

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της Elasticsearch.

T1 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο

```
GET /wiki_xl/Posts/_count?q=content:*y*
```

T2 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα (έκδοση για χρήση χωρίς ευρετήριο)

```
GET /wiki_xl/Posts/_count?q=content:"y"
```

T3 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στο τέλος μιας λέξης

```
GET /wiki_xl/Posts/_count?q=content:*y
```

T4 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης (χωρις χρήση ευρετηρίου)

```
GET /wiki_xl/Posts/_count?q=content:y*
```

T5 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο κείμενο

```
GET /wiki_xl/Posts/_count?q=content:*new*
```

T6 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης (χωρίς ευρετήριο)

```
GET /wiki_xl/Posts/_count?q=content:*new
```

T7 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στην αρχή μιας λέξης

```
GET /wiki_xl/Posts/_count?q=content:new*
```

T8 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη, αυτούσια ή μέρος σύνθετης λέξης

```
GET /wiki_xl/Posts/_count?q=content:*York*
```

T9 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια

```
GET /wiki_xl/Posts/_count?q=content:"york"
```

T10 Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
GET /wiki_xl/Posts/_count?q=content:"New York City"
```

T11 Εύρεση εγγραφών στις οποίες περιέχεται μία λέξη τουλάχιστον 35 χαρακτήρων

```
POST /wiki_xl/Posts/_search
{
    "query": {
        "regexp":{
            "content": "[a-zA-Z]{35}"
        }
    }
}
```

4.5.1.4 Ερωτήματα κειμένου σε Neo4j

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της Neo4j.

T1 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο

```
MATCH
   (n:Posts)
WHERE
   toLower(n.content) contains "y"
RETURN
   COUNT(n);
```

T2 Εύρεση εγγραφών στις οποίες υπάρχει ένα συγκεκριμένο γράμμα μέσα στο κείμενο χωρίς να περιστοιχίζεται από άλλα γράμματα

```
MATCH
   (n:Posts)
WHERE
   toLower(n.content) =~ "(?i).*(?:\\W|^)y(?:\\W|$).*"
RETURN COUNT(n);
```

T3 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στο τέλος μιας λέξης

```
MATCH
    (n:Posts)
WHERE
   toLower(n.content) =~ "(?i).*y(?:\\W|$).*"
RETURN
    COUNT(n)
```

T4 Εύρεση εγγραφών στις οποίες ένα συγκεκριμένο γράμμα βρίσκεται στην αρχή μιας λέξης

```
MATCH
    (n:Posts)
WHERE
     toLower(n.content) =~ "(?i).*(?:\\W|^)y.*"
RETURN
     COUNT(n)
```

T5 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο κείμενο

```
MATCH
    (n:Posts)
WHERE
   toLower(n.content) contains "new"
RETURN
    COUNT(n)
```

T6 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στο τέλος μιας λέξης

```
MATCH
    (n:Posts)
WHERE
   toLower(n.content) =~ "(?i).*new(?:\\W|$).*"
RETURN
    COUNT(n)
```

T7 Εύρεση εγγραφών στις οποίες τρία γράμματα βρίσκονται μέσα στην αρχή μιας λέξης

```
MATCH
    (n:Posts)
WHERE
     toLower(n.content) =~ "(?i).*(?:\\W|^)new.*"
RETURN
     COUNT(n)
```

T8 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη, αυτούσια ή μέρος σύνθετης λέξης

```
MATCH
    (n:Posts)
WHERE
     toLower(n.content) contains "york"
RETURN
     COUNT(n)
```

T9 Εύρεση εγγραφών στις οποίες υπάρχει μια ολόκληρη λέξη αυτούσια

```
MATCH
    (n:Posts)
WHERE
     toLower(n.content) =~ "(?i).*(?:\\W|^)york(?:\\W|$).*"
RETURN COUNT(n)
```

T10 Εύρεση εγγραφών στις οποίες περιέχεται μια ολόκληρη φράση

```
MATCH
     (n:Posts)
```

```
WHERE
    toLower(n.content) contains "new york city"
RETURN
    COUNT(n)
```

T11 Εύρεση εγγραφών στις οποίες περιέχεται μία λέξη τουλάχιστον 35 χαρακτήρων

```
MATCH
    (n:Posts)
WHERE
    toLower(n.content) =~ "(?i).*(?:\\W|^)[a-zA-Z]{35}(?:\\W|$).*"
RETURN
    COUNT(n)
```

4.5.2 Ερωτήματα σε γεωγραφικά δεδομένα

Τα ερωτήματα προς τις βάσεις δεδομένων με δεδομένα κειμένου αφορούσαν την ανάκτηση δεδομένων κειμένου με βάση τα παρακάτω κριτήρια:

1. Η εύρεση ενός συγκεκριμένου, τυχαία επιλεγμένου σημείου (G1)
2. Η αναζήτηση σημείων μέσα σε κύκλο διαφορετικής ακτίνας με κέντρο ένα συγκεκριμένο, τυχαία επιλεγμένο σημείο. Η ακτίνα του κύκλου δοκιμάστηκε για 4 διαφορετικές τιμές, 100 μέτρα (G2), 500 μέτρα (G3), ένα χιλιόμετρο (G4) και 2.5 χιλιόμετρα (G5).
3. Η αναζήτηση σημείων μέσα σε κύκλο διαφορετικής ακτίνας με κέντρο ένα συγκεκριμένο, τυχαία επιλεγμένο σημείο τα οποία ανήκουν σε κάποια συγκεκριμένη κατηγορία ενδιαφέροντος, για παράδειγμα καφετέριες ή μουσεία. Η ακτίνα του κύκλου δοκιμάστηκε για 4 διαφορετικές τιμές, 100 μέτρα (G6), 500 μέτρα (G7), ένα χιλιόμετρο (G8) και 2.5 χιλιόμετρα (G9).
4. Η αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με διαφορετικό εμβαδό. Το εμβαδό δοκιμάστηκε για 4 διαφορετικές τιμές:
    i. 12.500 τετραγωνικά μέτρα (G10)
    ii. 1 τετραγωνικό χιλιόμετρο (G11)
    iii. 60 τετραγωνικά χιλιόμετρα (G12)
    iv. 24.000 τετραγωνικά χιλιόμετρα (G13)

4.5.2.1 Ερωτήματα σε γεωγραφικά δεδομένα σε MySQL

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της MySQL.

G1 εύρεση ενός συγκεκριμένου, τυχαία επιλεγμένου σημείου

```
select
      COUNT(*)
FROM PLACES p
WHERE p.LAT = 37.9625552891062
  AND p.LON = 23.6896133422852
```

G2 Αναζήτηση σημείων μέσα σε κύκλο 100 μέτρων

```
select
      COUNT(*)
from
      (SELECT
            id,
            lon,
            lat,
            name,
            (6371 * ACOS (COS(RADIANS(37.9625552891062)) *
```

```
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM
            places
      HAVING
            distance < 0.1
      ORDER BY distance) as distances;
```

G3 Αναζήτηση σημείων μέσα σε κύκλο 500 μέτρων

```
select
      COUNT(*)
from
      (select
            id,
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 0.5
      ORDER BY distance) as distances
```

G4 Αναζήτηση σημείων μέσα σε κύκλο 1000 μέτρων

```
select
      COUNT(*)
from (select
            id,
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM
            places
      HAVING distance < 1
      ORDER BY distance) as distances
```

G5 Αναζήτηση σημείων μέσα σε κύκλο 2500 μέτρων

```
select
      COUNT(*)
from (select
            id,
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 2.5
      ORDER BY distance) as distances
```

G6 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 100 μέτρων

```
select
      COUNT(*)
from (select
            id,
```

```
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 0.1
      ORDER BY distance) as distances,
      categories c,
      has_category h
where distances.id = h.place_id
  AND h.category_id = c.id
  AND lower(c.category) LIKE '%event space%';
```

G7 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 500 μέτρων

```
select
      COUNT(*)
from (select
            id,
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 0.5
      ORDER BY distance) as distances,
      categories c,
      has_category h
where distances.id = h.place_id
  AND h.category_id = c.id
  AND lower(c.category) LIKE '%event space%';
```

G8 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 1000 μέτρων

```
select
      COUNT(*)
from (select
            id,
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 1
      ORDER BY distance) as distances,
      categories c,
      has_category h
where distances.id = h.place_id
  AND h.category_id = c.id
  AND lower(c.category) LIKE '%event space%';
```

G9 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 2500 μέτρων

```
select
      COUNT(*)
from (select
            id,
```

```
            lon,
            lat,
            name,
             (6371 * ACOS (COS (RADIANS(37.9625552891062)) *
COS(RADIANS(lat)) * COS(RADIANS(lon) - RADIANS(23.6896133422852)) + SIN
(RADIANS(37.9625552891062)) * SIN(RADIANS(lat)))) AS distance
      FROM places
      HAVING distance < 2.5
      ORDER BY distance) as distances,
      categories c,
      has_category h
where distances.id = h.place_id
  AND h.category_id = c.id
  AND lower(c.category) LIKE '%event space%';
```

G10 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 12500τμ

```
select
      COUNT(*)
from places p
where p.lat > 37.9625552
  AND p.lat < 37.9625562
  AND p.lon > 23.689613342
  AND p.lon < 23.689613362
```

G11 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 1 τχλμ

```
select
      COUNT(*)
from places p
where p.lat > 37.9605252891061
  AND p.lat < 37.9685852891063
  AND p.lon > 23.6816133122851
  AND p.lon < 23.6896133722853
```

G12 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 60τχλμ

```
select
      COUNT(*)
from places p
where p.lat > 37.9005252891061
  AND p.lat < 37.9685852891063
  AND p.lon > 23.6116133122851
  AND p.lon < 23.6896133722853
```

G13 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 250000τχλμ

```
select
      COUNT(*)
from places p
where p.lat > 36.0005252891061
  AND p.lat < 37.9685852891063
  AND p.lon > 23.6116133122851
  AND p.lon < 24.6896133722853
```

4.5.2.2 Ερωτήματα σε γεωγραφικά δεδομένα σε MongoDB

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της MongoDB.

G1 εύρεση ενός συγκεκριμένου, τυχαία επιλεγμένου σημείου

```
db.places.count({
    "loc.coordinates": [24.05532502519887, 37.711385328159864]
```

```
})
```

G2 Αναζήτηση σημείων μέσα σε κύκλο 100 μέτρων

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$centerSphere": [
                [23.6896133422852, 37.9625552891062], 0.1 / 6371
            ]
        }
    }
})
```

G3 Αναζήτηση σημείων μέσα σε κύκλο 500 μέτρων

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$centerSphere": [
                [23.6896133422852, 37.9625552891062], 0.5 / 6371
            ]
        }
    }
})
```

G4 Αναζήτηση σημείων μέσα σε κύκλο 1000 μέτρων

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$centerSphere": [
                [23.6896133422852, 37.9625552891062], 1 / 6371
            ]
        }
    }
})
```

G5 Αναζήτηση σημείων μέσα σε κύκλο 2500 μέτρων

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$centerSphere": [
                [23.6896133422852, 37.9625552891062], 2.5 / 6371
            ]
        }
    }
})
```

G6 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 100 μέτρων

```
db.places.count({
    "$and": [{
        "loc": {
            "$geoWithin": {
                "$centerSphere": [
                    [23.6896133422852, 37.9625552891062], 0.1 / 6371
                ]
            }
        }
    }, {
        "categories.name.english": "Event Space"
    }]
})
```

G7 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 500 μέτρων

```
db.places.count({
    "$and": [{
        "loc": {
            "$geoWithin": {
                "$centerSphere": [
                    [23.6896133422852, 37.9625552891062], 0.5 / 6371
                ]
            }
        }
    }, {
        "categories.name.english": "Event Space"
    }]
})
```

G8 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 1000 μέτρων

```
db.places.count({
    "$and": [{
        "loc": {
            "$geoWithin": {
                "$centerSphere": [
                    [23.6896133422852, 37.9625552891062], 1 / 6371
                ]
            }
        }
    }, {
        "categories.name.english": "Event Space"
    }]
})
```

G9 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 2500 μέτρων

```
db.places.count({
    "$and": [{
        "loc": {
            "$geoWithin": {
                "$centerSphere": [
                    [23.6896133422852, 37.9625552891062], 2.5 / 6371
                ]
            }
        }
    }, {
        "categories.name.english": "Event Space"
    }]
})
```

G10 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 12500τμ

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$box": [
                [23.689613342, 37.9625552],
                [23.689613362, 37.9625562]
            ]
        }
    }
})
```

G11 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 1 τχλμ

```
db.places.count({
```

```
    "loc": {
        "$geoWithin": {
            "$box": [
                [23.6816133122851, 37.9605252891061],
                [23.6896133722853, 37.9685852891063]
            ]
        }
    }
})
```

G12 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 60τχλμ

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$box": [
                [23.6116133122851, 37.9005252891061],
                [23.6896133722853, 37.9685852891063]
            ]
        }
    }
})
```

G13 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 250000τχλμ

```
db.places.count({
    "loc": {
        "$geoWithin": {
            "$box": [
                [23.6116133122851, 36.0005252891061],
                [24.6896133722853, 37.9685852891063]
            ]
        }
    }
})
```

4.5.2.3 Ερωτήματα σε γεωγραφικά δεδομένα σε Elasticsearch

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της Elasticsearch.

G1 εύρεση ενός συγκεκριμένου, τυχαία επιλεγμένου σημείου

```
POST /places/_count {
  "query": {
    "bool": {
      "must": [{
        "terms": {
          "location": [23.6896133422852, 37.9625552891062]
        }
      }]
    }
  }
}
```

G2 Αναζήτηση σημείων μέσα σε κύκλο 100 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
```

```
        "geo_distance": {
          "distance": "0.1km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G3 Αναζήτηση σημείων μέσα σε κύκλο 500 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_distance": {
          "distance": "0.5km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G4 Αναζήτηση σημείων μέσα σε κύκλο 1000 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_distance": {
          "distance": "1km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G5 Αναζήτηση σημείων μέσα σε κύκλο 2500 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
```

```
        "geo_distance": {
          "distance": "2.5km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G6 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 100 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match": {
          "categories.name.english": "Event Space"
        }
      },
      "filter": {
        "geo_distance": {
          "distance": "0.1km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G7 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 500 μέτρων

```
GET /places/_count {
  "query": {
    "bool": {
      "must": {
        "match": {
          "categories.name.english": "Event Space"
        }
      },
      "filter": {
        "geo_distance": {
          "distance": "0.5km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G8 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 1000 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
```

```
      "must": {
        "match": {
          "categories.name.english": "Event Space"
        }
      },
      "filter": {
        "geo_distance": {
          "distance": "1km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G9 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 2500 μέτρων

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match": {
          "categories.name.english": "Event Space"
        }
      },
      "filter": {
        "geo_distance": {
          "distance": "2.5km",
          "location": {
            "lat": 37.9625552891062,
            "lon": 23.6896133422852
          }
        }
      }
    }
  }
}
```

G10 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 12500τμ

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "bottom_left": {
              "lat": 37.9625552,
              "lon": 23.689613342
            },
            "top_right": {
              "lat": 37.9625562,
              "lon": 23.689613362
            }
          }
        }
      }
```

```
    }
  }
}
```

G11 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 1 τχλμ

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "bottom_left": {
              "lat": 37.9605252891061,
              "lon": 23.6816133122851
            },
            "top_right": {
              "lat": 37.9685852891063,
              "lon": 23.6896133722853
            }
          }
        }
      }
    }
  }
}
```

G12 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 60τχλμ

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "bottom_left": {
              "lat": 37.9005252891061,
              "lon": 23.6116133122851
            },
            "top_right": {
              "lat": 37.9685852891063,
              "lon": 23.6896133722853
            }
          }
        }
      }
    }
  }
}
```

G13 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 250000τχλμ

```
POST /places/_count {
  "query": {
    "bool": {
      "must": {
        "match_all": {}
```

```
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "bottom_left": {
              "lat": 36.0005252891061,
              "lon": 23.6116133122851
            },
            "top_right": {
              "lat": 37.9685852891063,
              "lon": 24.6896133722853
            }
          }
        }
      }
    }
  }
}
```

4.5.2.4 Ερωτήματα σε γεωγραφικά δεδομένα σε Neo4j

Παρακάτω παρουσιάζεται η έκδοση τον ερωτημάτων που επεξηγήθηκαν παραπάνω για το σύστημα διαχείρισης της Neo4j.

G1 εύρεση ενός συγκεκριμένου, τυχαία επιλεγμένου σημείου

```
MATCH (p:places)
WHERE
   p.lat = 37.9625552891062
   and p.lon = 23.6896133422852
RETURN count(p);
```

G2 Αναζήτηση σημείων μέσα σε κύκλο 100 μέτρων

```
MATCH (p:places)
WHERE
   6371 *
   ACOS(
      COS(
          RADIANS(37.9625552891062)
        ) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
              RADIANS(toFloat(p.lon)) -
              RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 0.1
RETURN count(p);
```

G3 Αναζήτηση σημείων μέσα σε κύκλο 500 μέτρων

```
MATCH (p:places)
WHERE
   6371 *
   ACOS(
      COS(
          RADIANS(37.9625552891062)
        ) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
              RADIANS(toFloat(p.lon)) -
```

```
            RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 0.5
RETURN count(p);
```

G4 Αναζήτηση σημείων μέσα σε κύκλο 1000 μέτρων

```
MATCH (p:places)
WHERE
   6371 *
   ACOS(
      COS(
          RADIANS(37.9625552891062)
        ) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
              RADIANS(toFloat(p.lon)) -
              RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 1
RETURN count(p);
```

G5 Αναζήτηση σημείων μέσα σε κύκλο 2500 μέτρων

```
MATCH (p:places)
WHERE
   6371 *
   ACOS(
      COS(
          RADIANS(37.9625552891062)
        ) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
              RADIANS(toFloat(p.lon)) -
              RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 2.5
RETURN count(p);
```

G6 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 100 μέτρων

```
MATCH (p:places) - [:IS_OF_CATEGORY] -> (c:categories)
WHERE
   c.category = 'Event Space'
and 6371 *
    ACOS(
        COS(RADIANS(37.9625552891062)) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
          RADIANS(toFloat(p.lon)) -
            RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 0.1
RETURN count(p);
```

G7 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 500 μέτρων

```
MATCH (p:places) - [:IS_OF_CATEGORY] -> (c:categories)
WHERE
   c.category = 'Event Space'
and 6371 *
    ACOS(
        COS(RADIANS(37.9625552891062)) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
          RADIANS(toFloat(p.lon)) -
            RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 0.5
RETURN count(p);
```

G8 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 1000 μέτρων

```
MATCH (p:places) - [:IS_OF_CATEGORY] -> (c:categories)
WHERE
   c.category = 'Event Space'
and 6371 *
    ACOS(
        COS(RADIANS(37.9625552891062)) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
          RADIANS(toFloat(p.lon)) -
            RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 1
RETURN count(p);
```

G9 Αναζήτηση συγκεκριμένης κατηγορίας σημείων μέσα σε κύκλο 2500 μέτρων

```
MATCH (p:places) - [:IS_OF_CATEGORY] -> (c:categories)
WHERE
   c.category = 'Event Space'
and 6371 *
    ACOS(
        COS(RADIANS(37.9625552891062)) *
        COS(RADIANS(toFloat(p.lat))) *
        COS(
          RADIANS(toFloat(p.lon)) -
            RADIANS(23.6896133422852)
        ) +
        SIN(RADIANS(37.9625552891062)) *
        SIN(RADIANS(toFloat(p.lat)))
      ) < 2.5
RETURN count(p);
```

G10 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 12500τμ

```
MATCH (p:places)
WHERE
   p.lat > 37.9625552
   and p.lat < 37.9625562
   and p.lon > 23.689613342
   and p.lon < 23.689613362
RETURN count(p);
```

G11 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 1 τχλμ

```
MATCH (p:places)
WHERE
   p.lat > 37.9605252891061
   and p.lat < 37.9685852891063
   and p.lon > 23.6816133122851
   and p.lon < 23.6896133722853
RETURN count(p);
```

G12 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 60 τχλμ

```
MATCH (p:places)
WHERE
   p.lat > 37.9005252891061
   and p.lat < 37.9685852891063
   and p.lon > 23.6116133122851
   and p.lon < 23.6896133722853
RETURN count(p);
```

G13 αναζήτηση σημείων μέσα σε ορθογώνιο παραλληλόγραμμο με εμβαδό 250000 τχλμ

```
MATCH (p:places)
WHERE
   p.lat > 36.0005252891061
   and p.lat < 37.9685852891063
   and p.lon > 23.6116133122851
   and p.lon < 24.6896133722853
RETURN count(p);
```

4.6 Μετρικές

Για τη σύγκριση της παρούσας διπλωματικής χρησιμοποιήθηκε η μετρική του χρόνου εκτέλεσης του εκάστοτε ερωτήματος προς τη βάση. Πιο συγκεκριμένα, μετρήθηκε ο χρόνος ο οποίος μεσολάβησε μέχρι το πρόγραμμα πελάτη (client side) να έχει διαθέσιμα τα δεδομένα από τον εξυπηρετητή της εκάστοτε βάσης.

4.7 Πειραματική εκτέλεση

Η πειραματική εκτέλεση για τη μέτρηση του χρόνου απόκρισης της βάσης και της επιστροφής δεδομένων έλαβε χώρα στο Google Virtual Machine το οποίο περιγράφηκε σε προηγούμενο κεφάλαιο.

Για την εκτέλεση των μετρήσεων χρησιμοποιήθηκε το κελύφους bash, το οποίο είναι το προκαθορισμένο κέλυφος του λειτουργικού Ubuntu, για την αυτοματοποίηση και εκτέλεση του προγράμματος πελάτη που ήταν υπεύθυνο για τις μετρήσεις. Το πρόγραμμα πελάτης γράφτηκε στη γλώσσα προγραμματισμού python και τα αποτελέσματα της διαδικασίας αποθηκεύτηκαν σε ένα αρχείο της μορφής csv (comma separated values).

4.7.1 Τρόπος μέτρησης

Για κάθε query που μελετήθηκε στα πλαίσια της παρούσας διπλωματικής εξετάστηκε η συμπεριφορά κάθε συστήματος διαχείρισης βάσεων δεδομένων σε κατάσταση στην οποία το σύστημα μόλις έχει ξεκινήσει και δεν έχει κάποια δεδομένα cached καθώς και η συμπεριφορά όταν έχει τρέξει ήδη κάποια query και έχει κάποια δεδομένα cached.

Για τα query που θεωρήσαμε ότι τρέχουν σε μια βάση δεδομένων με cached data, ελήφθησαν 11 μετρήσεις για κάθε query και απορρίφθηκε η πρώτη μέτρηση, η οποία ήταν μέτρηση σε βάση δεδομένων χωρίς cached δεδομένα. Για τις μετρήσεις της συμπεριφοράς

των ερωτημάτων σε βάση δεδομένων χωρίς cached δεδομένα, ελήφθησαν πάλι 10 μετρήσεις για κάθε ερώτημα. Μεταξύ κάθε μέτρησης σε αυτή την περίπτωση μεσολαβούσε επανεκκίνηση του συστήματος διαχείρισης της βάσης.

Για τη μέτρηση του χρόνου χρησιμοποιήθηκε η προκαθορισμένη βιβλιοθήκη της python με όνομα time ώστε να μετρήσει το χρόνο που παρήλθε από τη στιγμή που το πρόγραμμα ζητάει τα δεδομένα μέχρι να τα έχει διαθέσιμα. Παρακάτω φαίνεται ένα παράδειγμα για το σύστημα της MongoDB.

Μέθοδος για τη μέτρηση του χρόνου για ένα ερώτημα σε MongoDB

```
def execQuery(queryName, queryString, times=10):
    results = []
    sum = 0
    for i in range(1, times + 1):
        start_time = time.time()
        r = requests.get(queryString, params=None)
        elapsed_time = 1000 * (time.time() - start_time)
        results.append(str(elapsed_time))
        sum += elapsed_time
        time.sleep(5)
```

Για την εκτέλεση των ερωτημάτων χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες:

- MySQLdb, για τη σύνδεση στη MySQL
- Pymongo, η προτεινόμενη βιβλιοθήκη διασύνδεση με τη βάση από τους δημιουργούς τα MongoDB
- Requests, που είναι βιβλιοθήκη για την αποστολή και λήψη http request στην python και χρησιμοποιήθηκαν για την εκτέλεση των ερωτημάτων στη Neo4j και την elasticsearch

4.7.2 Περιβαλλοντικές παράμετροι

Για να διασφαλιστεί ότι οι μετρήσεις ήταν όσο πιο ακριβής γίνεται, χωρίς να επηρεάζονται από άλλες παραμέτρους, στο σύστημα που επιλέχθηκε, πριν την εκτέλεση των πειραματικών μετρήσεων, απενεργοποιήθηκαν όλα τα προγράμματα και όλες οι υπηρεσίες που δεν είναι απαραίτητες για την ομαλή λειτουργία του συστήματος. Επιπρόσθετα, κάθε φορά ήταν ενεργό μόνο ένα σύστημα διαχείρισης βάσεων δεδομένων, αυτό για το οποίο έτρεχε εκείνη την ώρα κάποιο ερώτημα.

Για να διασφαλιστεί ότι κάθε ερώτημα δεν επηρεάζεται από κάποιο άλλο που έτρεξε νωρίτερα, για κάθε query που εξετάστηκε, πριν την εκτέλεσή του και την λήψη των μετρήσεων, πραγματοποιούνταν επανεκκίνησης της βάσης δεδομένων. Για τις μετρήσεις σε βάση που δεν περιείχε cached δεδομένα πραγματοποιούνταν επανεκκίνηση πριν από κάθε μέτρηση. Μετά από κάθε επανεκκίνηση μεσολαβούσε ένα διάστημα 30 δευτερολέπτων ώστε να βεβαιωθεί ότι όλες οι υπηρεσίες κάθε συστήματος είναι διαθέσιμες.

4.8 Μετρήσεις

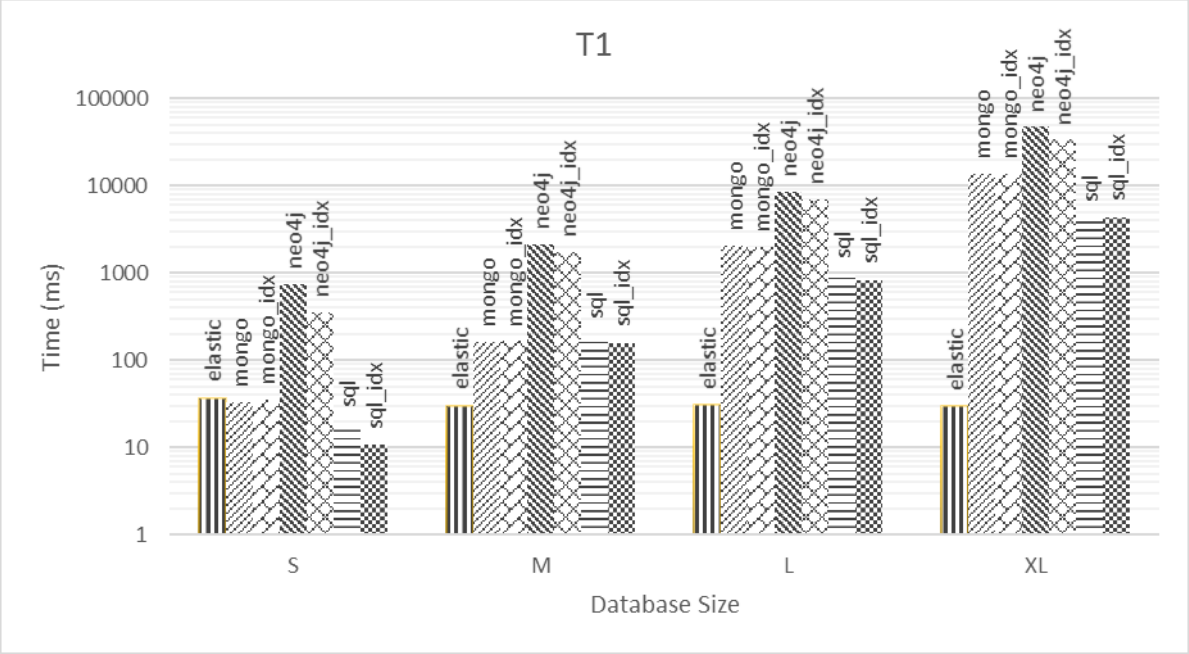
Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα που προέκυψαν από τη σύγκριση των συστημάτων διαχείρισης βάσεων δεδομένων. Στους συγκεντρωτικούς πίνακες και τα διαγράμματα που ακολουθούν παρουσιάζεται ο μέσος όρος των 10 μετρήσεων ανά ερώτημα. Να σημειωθεί ότι στα διαγράμματα χρησιμοποιήθηκε λογαριθμική κλίμακα στον άξονα του χρόνου, ώστε να γίνει πιο κατανοητή η διαφορά στην επίδοση.

4.8.1 Μετρήσεις στα ερωτήματα κειμένου

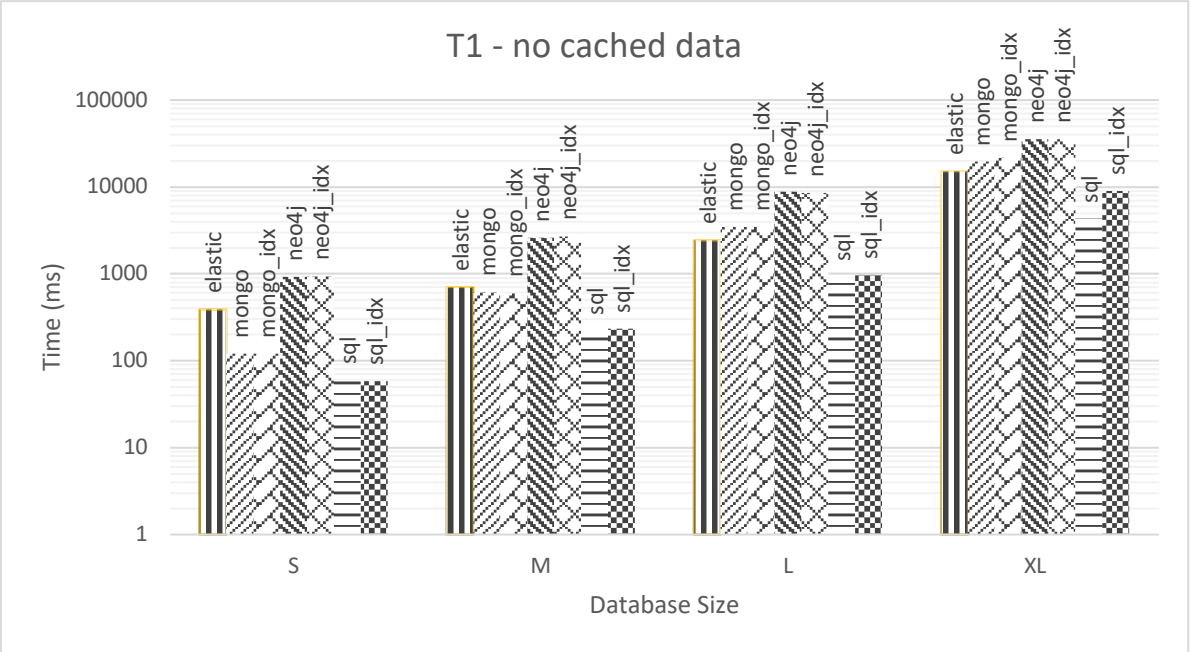
Παρακάτω ακολουθούν τα αποτελέσματα για τα ερωτήματα T1-T11 που περιγράφηκαν στο κεφάλαιο 4.5.1.

	Query T1	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	37.23	30.12	30.55	29.6
	MongoDB	33.38	164.47	2028.81	13483.85
	MongoDB with Index	35.96	165.3	2001.54	13572.46
	Neo4J	749.11	2125.1	8652.54	47215.44
	Neo4j with Index	355.65	1743.46	6937.33	33844.56
	MySQL	18.41	195	935.12	4206.06
	MySQL with Index	10.7	159.5	825.23	4299.09
Without Cache	ElasticSearch	388.15	706.7	2414.04	15231.39
	MongoDB	121.58	617.71	3478.32	19438.48
	MongoDB with Index	120.66	597.55	3494.75	21623.27
	Neo4J	923.58	2582.71	8809.72	35532.29
	Neo4j with Index	933.66	2695.87	8559.87	35404.99
	MySQL	59.78	223.85	904.53	4339.37
	MySQL with Index	58.38	233.27	972.53	8927.94

Αποτελέσματα για ερώτημα T1 σε ms



Διάγραμμα για ερώτημα T1 με cache



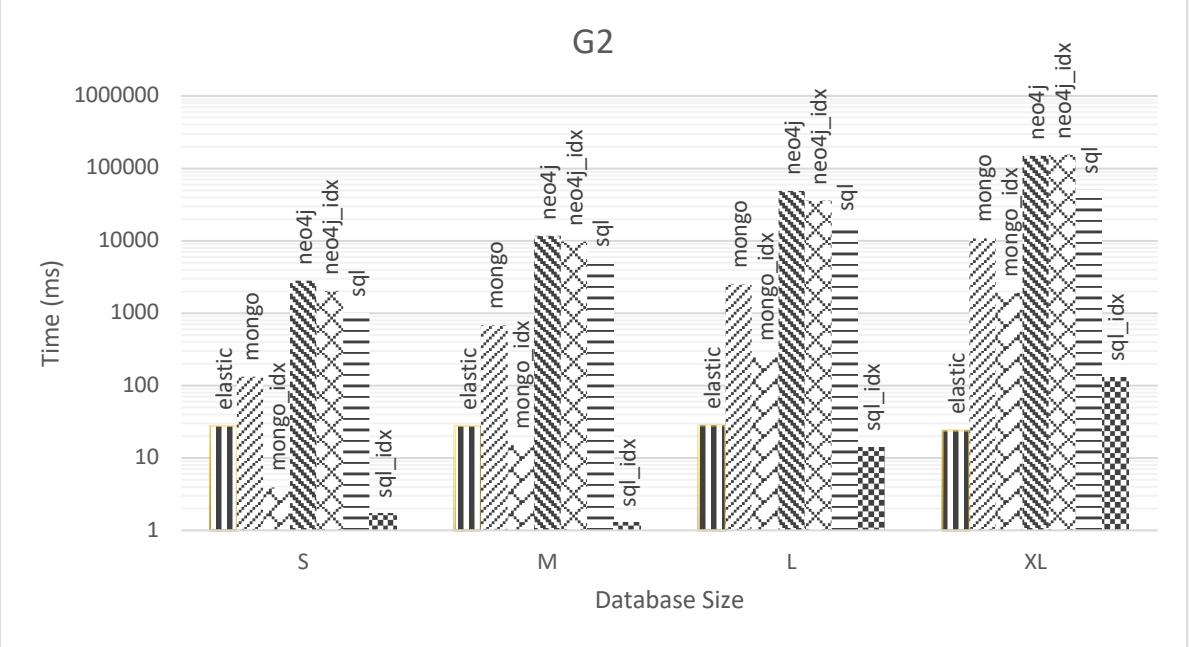
Διάγραμμα για ερώτημα T1 χωρίς cached data

Από τα παραπάνω αποτελέσματα γίνεται αντιληπτό ότι για το συγκεκριμένο ερώτημα, την εύρεση ενός γράμματος στο κείμενο, η Elasticsearch έχει την καλύτερη απόδοση ανεξαρτήτως από το μέγεθος των δεδομένων αποκρινόμενη σταθερά στα 30ms. Από τα υπόλοιπα συστήματα, η MySQL και η MongoDB έχουν καλή απόκριση όταν τα δεδομένα δεν είναι πολλά, επιστρέφοντας αποτελέσματα γύρω στα 100ms ενώ για τα δεδομένα σε βάση χωρίς cached data, όλες οι βάσεις παρουσιάζουν χειρότερη απόδοση, με τη MySQL να είναι ελαφρώς καλύτερη.

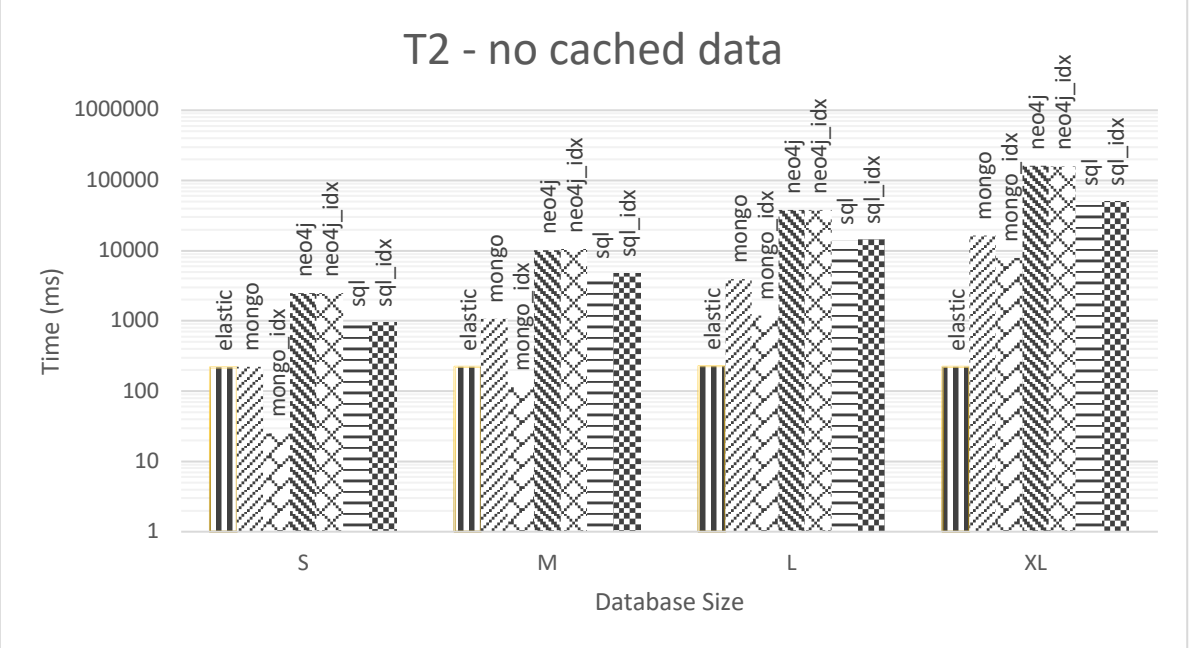
Να σημειωθεί ότι όπως ήταν αναμενόμενο, η απόδοση ήταν ίδια για τις Neo4j, MySQL, MongoDB με και χωρίς χρήση ευρετηρίου, καθώς το είδος του ερωτήματος δε μπορεί να εκμεταλλευτεί την παρουσία του.

	Query T2	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	27.61	27.6	28.31	23.83
	MongoDB	132.03	680.68	2485.57	10845.12
	MongoDB with Index	3.88	14.88	245.3	1924.74
	Neo4J	2837.81	11794.85	48352.08	148355.5
	Neo4j with Index	2016.66	9796.74	35638.79	155301.8
	MySQL	1004.91	4933.68	15108.89	50175.29
	MySQL with Index	1.69	1.28	14.01	130.33
Without Cache	ElasticSearch	218.98	221.23	228.24	222.62
	MongoDB	224.44	1085.14	3938.15	16420.96
	MongoDB with Index	25.15	112.67	1196.34	7847.82
	Neo4J	2498.69	10215.68	38339.56	163706.3
	Neo4j with Index	2498.26	10559.97	38085.14	159145.9
	MySQL	922.4	4548.91	14003.33	49494.06
	MySQL with Index	947.81	4729.94	14313.09	49844.69

Αποτελέσματα για ερώτημα T2 σε ms



Διάγραμμα για ερώτημα T2 με cache



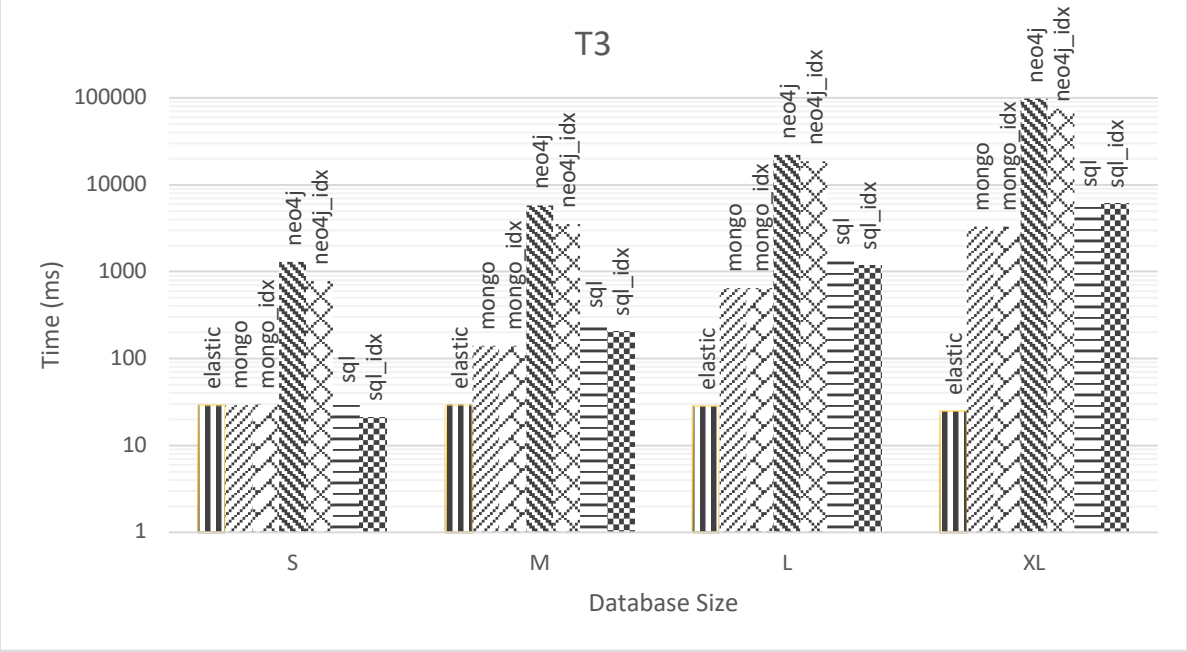
Διάγραμμα για ερώτημα T2 χωρίς cached data

Στις παραπάνω μετρήσεις για το ερώτημα της εύρεσης ενός γράμματος μόνου του μέσα στο κείμενο, γίνεται αντιληπτό ότι η MySQL και η MongoDB κάνουν πολύ καλή χρήση του ευρετηρίου κειμένου, ειδικά για σχετικά μικρό αριθμό δεδομένων, με απόκριση 1-3ms. Όταν ο όγκος των δεδομένων αυξάνεται, η MongoDB χάνει την ικανότητα της να απαντήσει το ίδιο γρήγορα στο ερώτημα και πλησιάζει την επίδοση της όταν δε χρησιμοποιεί ευρετήριο. Η Elasticsearch, αν και πιο αργή σε αρκετές περιπτώσεις, διατηρεί σταθερή και καλή επίδοση ανεξαρτήτως όγκου δεδομένων στα 25ms. Η Neo4j είναι σταθερά η λιγότερο αποδοτική με τη χρήση ευρετηρίου να μη συνδράμει σε καλύτερο αποτέλεσμα και στην καλύτερη των περιπτώσεων να επιστρέφει αποτελέσματα μετά από ένα δευτερόλεπτο

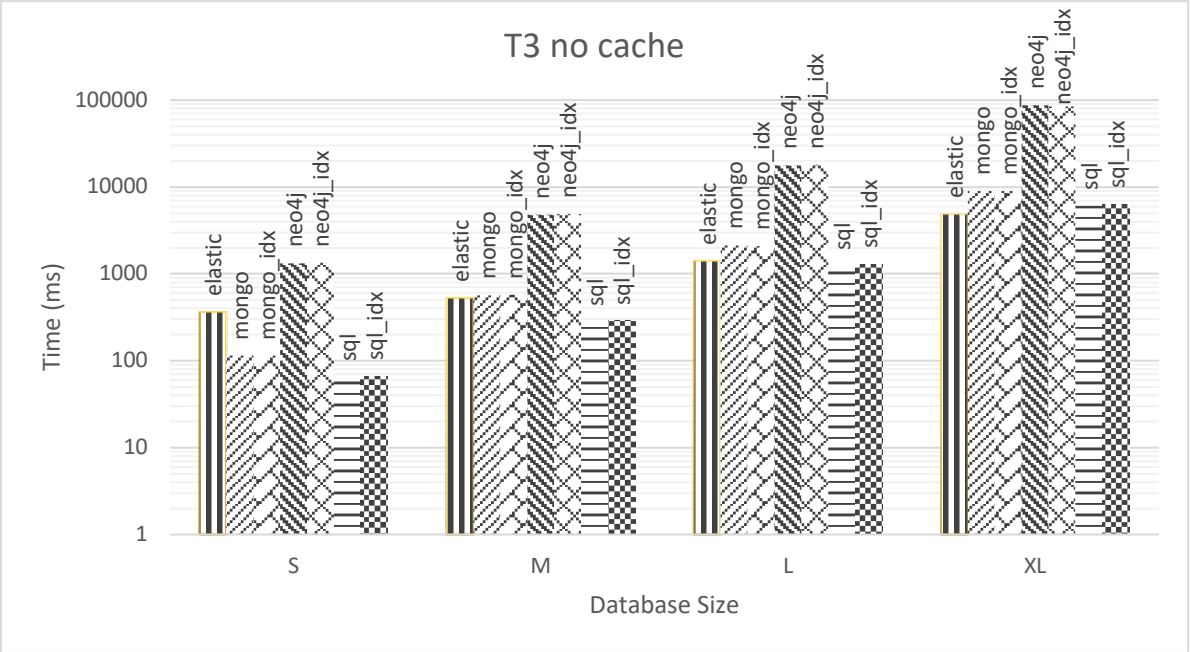
Για τα ερωτήματα που έτρεξαν χωρίς χρήση cache, παρατηρείται μια πολύ καλή απόκριση από τη MongoDB με χρήση ευρετηρίου για μικρό αριθμό δεδομένων ενώ η Elasticsearch και σε αυτή την περίπτωση διατηρεί σταθερή απόδοση γύρω στα 200ms.

	Query T3	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	28.99	29.37	28.42	24.89
	MongoDB	28.8	139.78	648.08	3285.59
	MongoDB with Index	29.75	139.84	649.16	3295
	Neo4J	1306.61	5795.31	21903.91	98263.86
	Neo4j with Index	774.37	3560.59	18704.14	75145.59
	MySQL	34.09	248.4	1317.14	6146.3
	MySQL with Index	21.09	205.01	1169.27	6079.96
Without Cache	ElasticSearch	362.08	532.17	1419.42	4892.37
	MongoDB	115.5	567.12	2109.43	8952.62
	MongoDB with Index	115.25	573.61	2085.46	8979.78
	Neo4J	1318.21	4830.29	17687.89	86627.06
	Neo4j with Index	1343.14	4884.65	18084.02	84521.68
	MySQL	64.7	277.92	1236.79	6097.52
	MySQL with Index	66.42	287.75	1295.01	6295.56

Αποτελέσματα για ερώτημα T3 σε ms



Διάγραμμα για ερώτημα T3 με cache

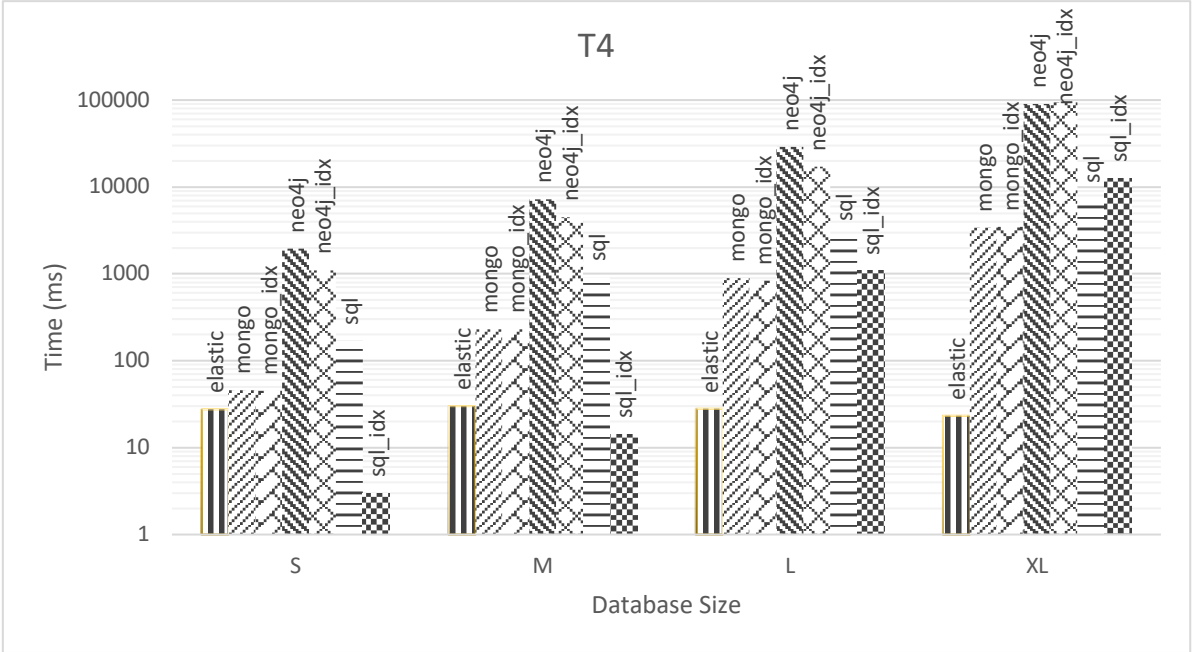


Διάγραμμα για ερώτημα T3 χωρίς cached data

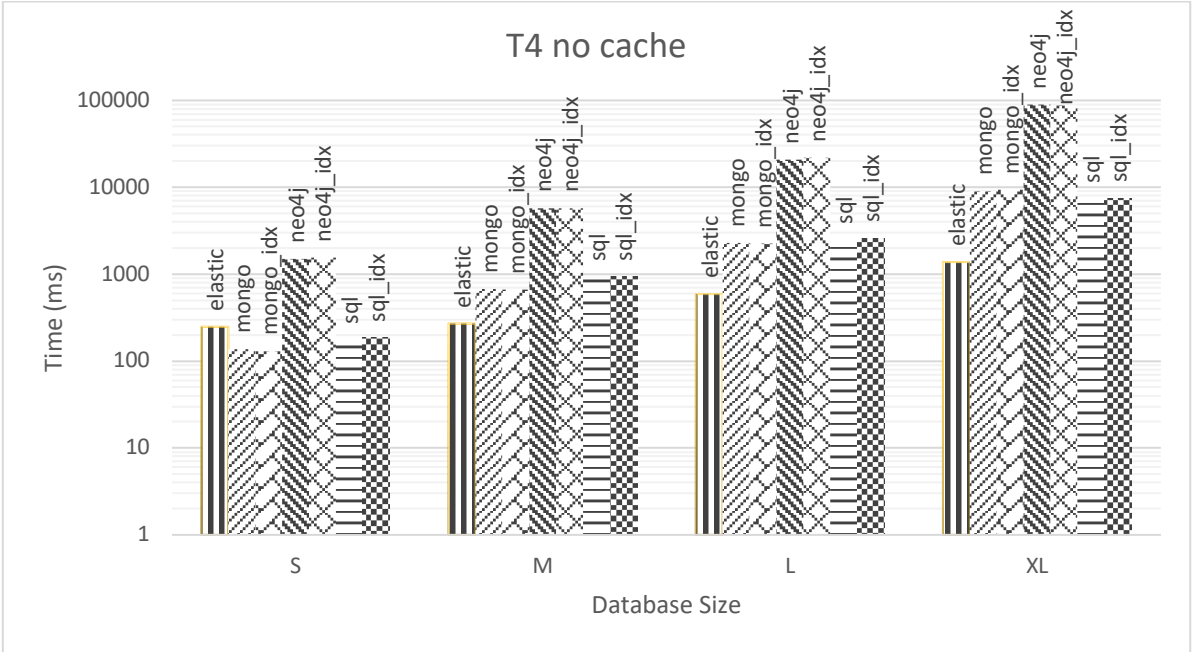
Για το συγκεκριμένο ερώτημα, την εύρεση λέξης στο κείμενο που να τελειώνει με ένα συγκεκριμένο γράμμα, γίνεται αντιληπτό από τα παραπάνω αποτελέσματα πώς η Elasticsearch πάλι έχει σταθερά καλή συμπεριφορά γύρω στα 30ms, ενώ δε γίνεται όπως ήταν αναμενόμενο χρήση του ευρετηρίου κειμένου στα υπόλοιπα συστήματα. Παρόλα αυτά MySQL και MongoDB αποκρίνονται στο φάσμα των 100-200ms για μικρό και μέτριο όγκο δεδομένων. Παρατηρούμε επίσης ότι όλες οι βάσεις παρουσιάζουν παρόμοια συμπεριφορά όταν τα δεδομένα δεν είναι cached με τη Neo4J όμως να είναι πάντοτε μια τάξη χρόνου χειρότερη.

	Query T4	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	27.66	29.8	28.22	23.29
	MongoDB	45.69	230.09	898.73	3407.92
	MongoDB with Index	45.39	231.9	838.88	3450.22
	Neo4J	1952.11	7226.88	28977.2	90490.15
	Neo4j with Index	1095.41	4509.12	17208.43	95453.21
	MySQL	170.59	902.88	2662.62	7380.1
	MySQL with Index	2.98	14.14	1083.76	12617.85
Without Cache	ElasticSearch	247.4	268.86	593.86	1370.92
	MongoDB	136.87	670.58	2278.32	9056.9
	MongoDB with Index	130.09	666.28	2273.63	9364.47
	Neo4J	1504.19	5780.64	20935	89946.85
	Neo4j with Index	1543.68	5734.01	21850.94	87516.98
	MySQL	177.96	915.85	2507.21	7425.05
	MySQL with Index	185	939.42	2573.12	7533.11

Αποτελέσματα για ερώτημα T4 σε ms



Διάγραμμα για ερώτημα T4 με cache



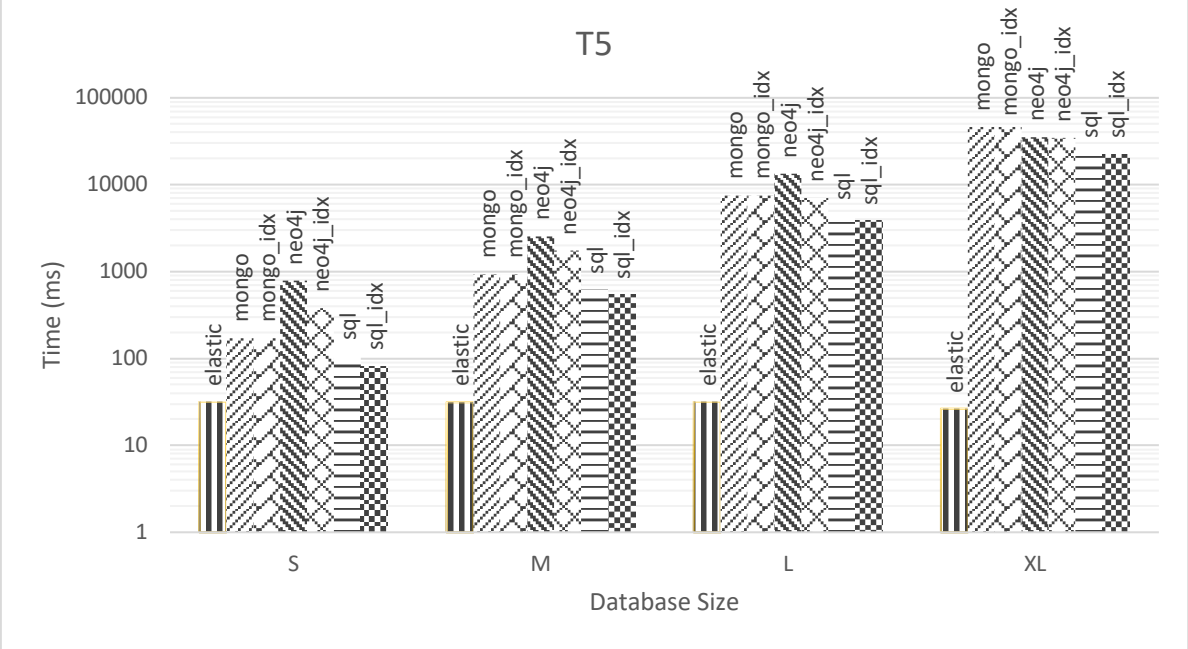
Διάγραμμα για ερώτημα T4 χωρίς cached data

Στο συγκεκριμένο ερώτημα, την εύρεση λέξης μέσα στο κείμενο η οποία ξεκινάει από συγκεκριμένο γράμμα, η MySQL με χρήση ευρετηρίου είναι σαφώς ταχύτερη από όλα τα συστήματα για μικρό όγκο δεδομένων αποκρινόμενη σε <10ms. Όσο ο όγκος αυτών των δεδομένων μεγαλώνει η απόκριση φτάνει να ξεπερνάει ακόμα και το 1 δευτερόλεπτο και να είναι παρόμοια με τη μη χρήση ευρετηρίου, ενώ η Elasticsearch και πάλι παραμένει ανεπηρέαστη αποκρινόμενη σταθερά γύρω στα 25ms. Neo4j και MongoDB αποτυγχάνουν να χρησιμοποιήσουν το ευρετήριο με αποτέλεσμα να επηρεάζεται αρνητικά η επίδοση τους.

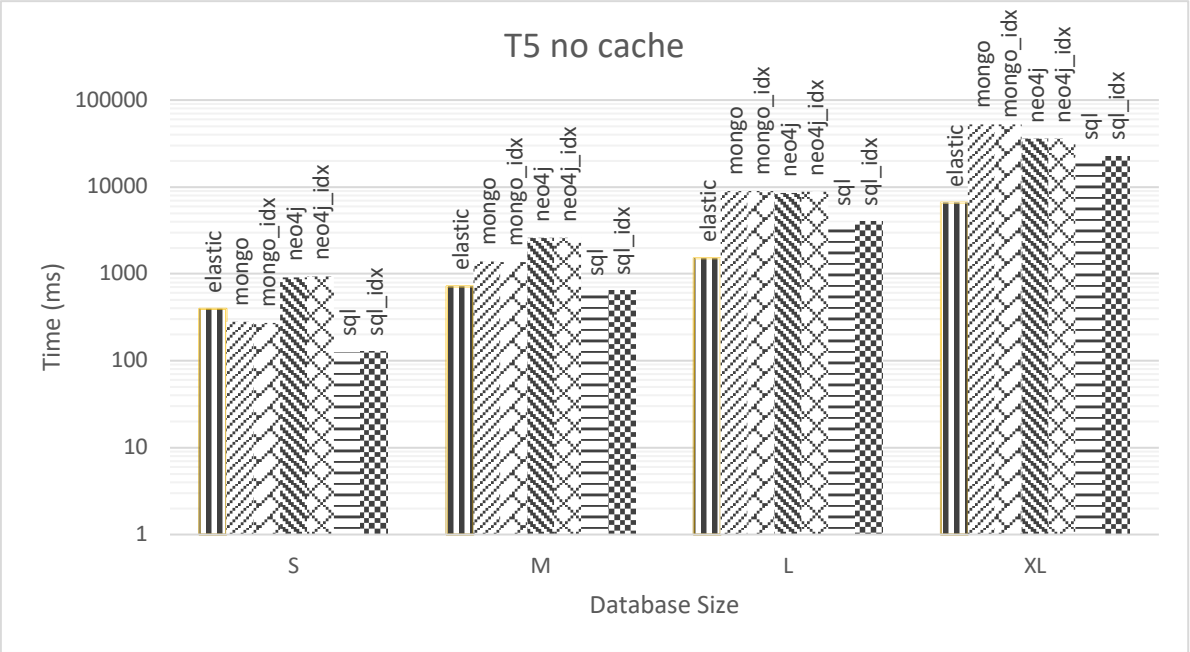
Για τη χρήση του ερωτήματος σε μη cached δεδομένα, η Elasticsearch είναι η μόνη που καταφέρνει να επιστρέψει αποτελέσματα στην τάξη του δευτερολέπτου στη χειρότερη περίπτωση.

	Query T5	*Database Size*			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	31.81	31.48	31.91	26.29
	MongoDB	169.72	927.6	7527.6	45929.97
	MongoDB with Index	170.47	939.56	7483.82	46229.09
	Neo4J	782.01	2545.85	13308.11	34942.69
	Neo4j with Index	377.99	1759.97	7007.79	34486.1
	MySQL	106.6	617.45	4339.69	22765.38
	MySQL with Index	81.09	545.48	3838.59	22445.4
Without Cache	ElasticSearch	396.98	717.16	1520.08	6568.85
	MongoDB	282.23	1395.11	8987.97	52416.95
	MongoDB with Index	272.39	1362.27	8996.6	52071.77
	Neo4J	908.62	2588.85	8534.8	36169.16
	Neo4j with Index	943.6	2583.85	8864.02	36083.59
	MySQL	123.84	615.75	3915.96	22426.6
	MySQL with Index	127.05	638.52	4007.44	22676.61

Αποτελέσματα για ερώτημα T5 σε ms



Διάγραμμα για ερώτημα T5 με cache



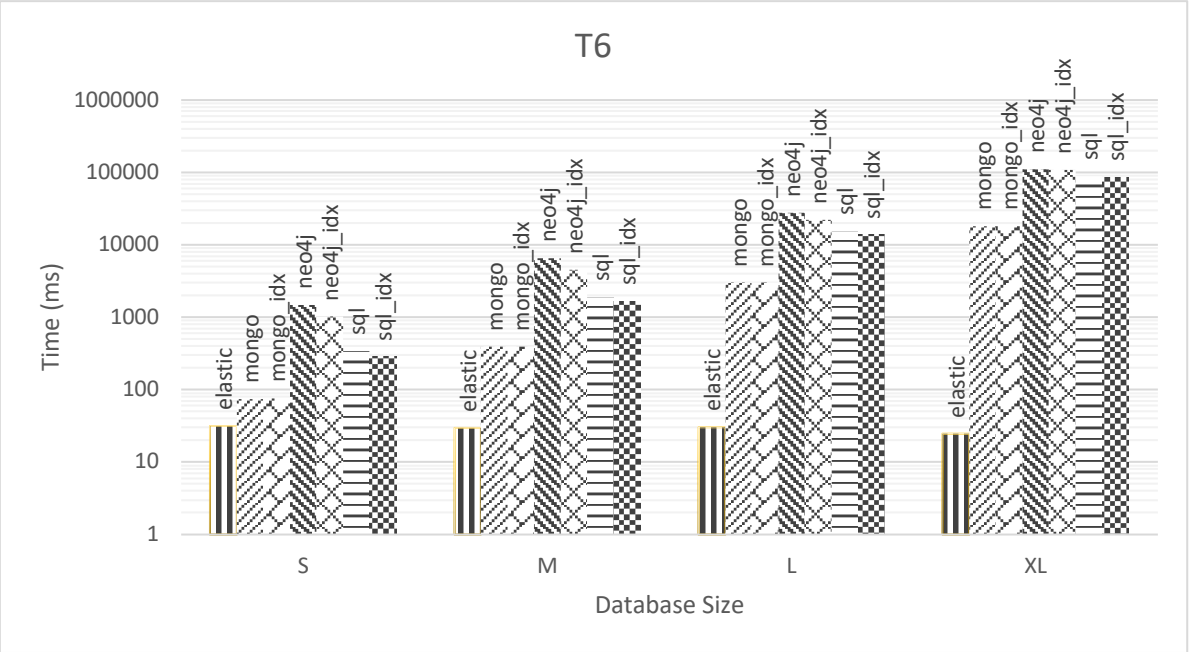
Διάγραμμα για ερώτημα T5 χωρίς cached data

Στο συγκεκριμένο ερώτημα, την εύρεση τριών γραμμάτων τυχαία μέσα στο κείμενο, όπως αναμενόταν κανένα σύστημα δε μπορεί να επωφεληθεί από το ευρετήριο κειμένου εκτός από την Elasticsearch που είναι το μόνο σύστημα που διατηρεί μια σταθερή και γρήγορη απόκριση, πάλι γύρω στα 30ms. MongoDB και MySQL καταφέρνουν μια απόκριση στα 100ms μόνο για μικρό όγκο δεδομένων ενώ ξανά η Neo4j είναι τάξεις χειρότερη από τα υπόλοιπα συστήματα.

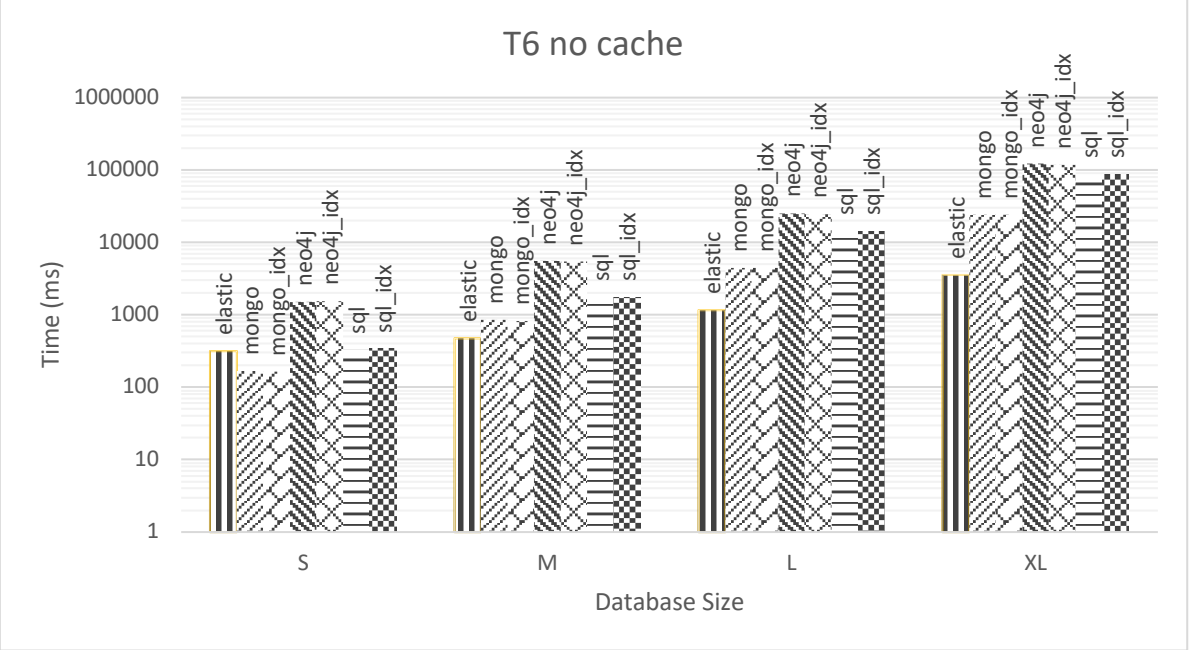
Για τη χρήση του ερωτήματος σε βάση χωρίς δεδομένα, παρατηρούμε ότι όλα τα συστήματα παρουσιάζουν παρόμοια απόδοση.

	Query T6	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	31.59	29.58	30.3	24.69
	MongoDB	74.44	394.69	3008.24	18188.27
	MongoDB with Index	76.04	392.5	3008.66	18219.69
	Neo4J	1480.87	6519.47	27511.73	110873.5
	Neo4j with Index	1018.37	4535.57	22172.29	108758.2
	MySQL	344.86	1876.71	15303.93	87703.81
	MySQL with Index	284.9	1663.37	13871.63	86580.99
Without Cache	ElasticSearch	318.24	470.1	1167.3	3528.55
	MongoDB	167.24	854.98	4437.41	23934.89
	MongoDB with Index	163.95	827.34	4439.81	23915.92
	Neo4J	1497.02	5601.34	24936.07	123040.6
	Neo4j with Index	1530.35	5427.49	24484.8	118991.8
	MySQL	327.46	1736.07	14252.84	86716.47
	MySQL with Index	339.49	1763.09	14302.82	87294.53

Αποτελέσματα για ερώτημα T6 σε ms



Διάγραμμα για ερώτημα T6 με cache



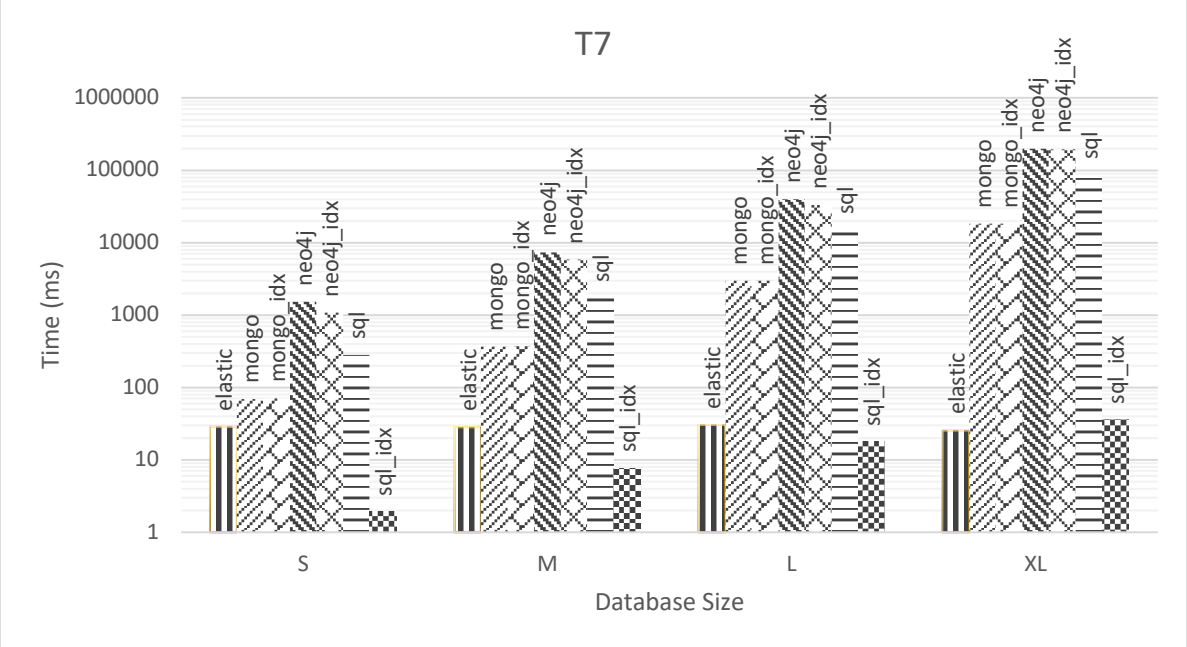
Διάγραμμα για ερώτημα T6 χωρίς cached data

Και στο συγκεκριμένο ερώτημα, στο οποίο γίνεται εύρεση λέξης που τελειώνει με 3 συγκεκριμένα γράμματα, όπως ήταν αναμενόμενο δεν έγινε χρήση του ευρετηρίου κειμένου. Η Elasticsearch ξεχωρίζει και πάλι για την ταχύτητα και την σταθερότητα ανεξαρτήτου μεγέθους δεδομένων με αποκρίσεις μεταξύ 25 και 30 ms. Η MongoDB μπορεί να ανταγωνιστεί αυτές τις ταχύτητες μόνο για μικρό όγκο δεδομένων ενώ για μεγάλο και πολύ μεγάλο όγκο δεδομένων η Elasticsearch είναι 2 και 3 τάξεις μεγέθους ταχύτερη από όλα τα συστήματα.

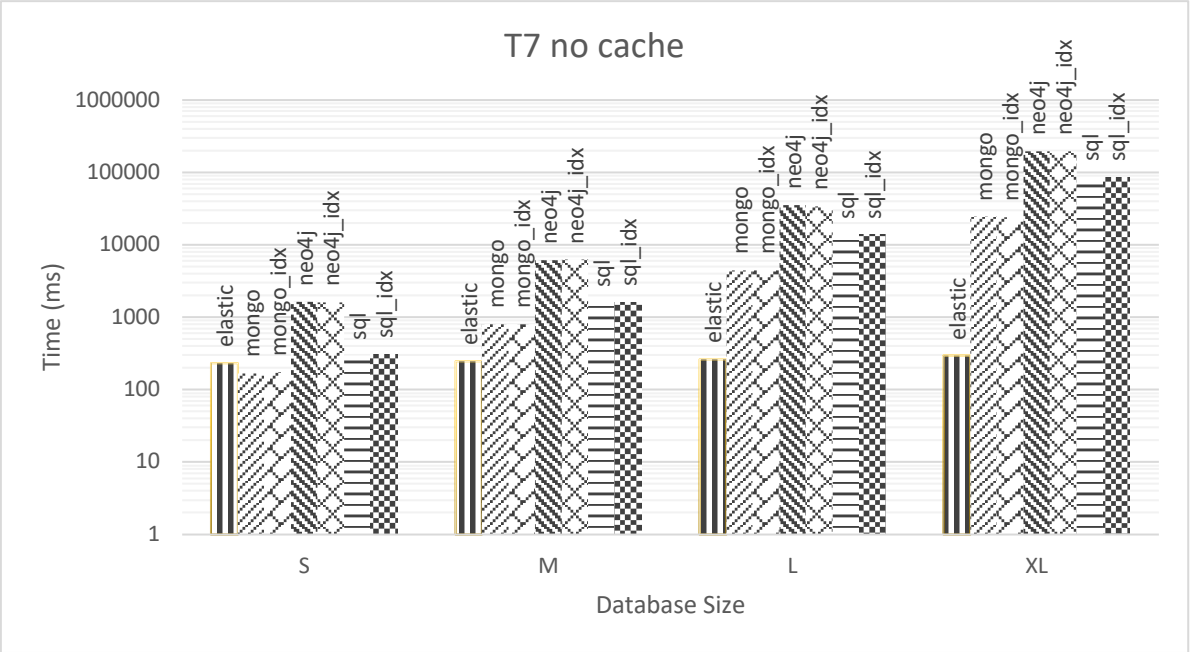
Για την περίπτωση που το ερώτημα τρέχει σε βάση δεδομένων χωρίς cached data, η Elasticsearch είναι η μόνη που καταφέρνει να κρατήσει την απόκριση σχετικά χαμηλά, αν και αυτή ξεπερνά τα 3 δευτερόλεπτα όταν ο όγκος δεδομένων γίνει πολύ μεγάλος.

	Query T7	*Database Size*			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	28.65	28.25	30.45	25.51
	MongoDB	69.24	367.87	2975.12	18368.81
	MongoDB with Index	70.14	373.48	2989.94	18484.15
	Neo4J	1534.77	7390.9	39390.75	198483.7
	Neo4j with Index	1090.66	5884.99	33342.75	193571
	MySQL	281.5	1722.21	15045.97	86030.5
	MySQL with Index	1.91	7.6	17.7	36.35
Without Cache	ElasticSearch	233.75	246.58	264.67	298.49
	MongoDB	168.86	797.05	4409.83	24419.53
	MongoDB with Index	175.98	802.96	4407.12	24176.85
	Neo4J	1646.52	6169.69	35684.04	194937.5
	Neo4j with Index	1604.82	6259.33	33860.11	195976.7
	MySQL	295.02	1560.36	13853.88	86275.34
	MySQL with Index	305.79	1603.77	13950.88	86497.11

Αποτελέσματα για ερώτημα T7 σε ms



Διάγραμμα για ερώτημα T7 με cache



Διάγραμμα για ερώτημα T7 χωρίς cached data

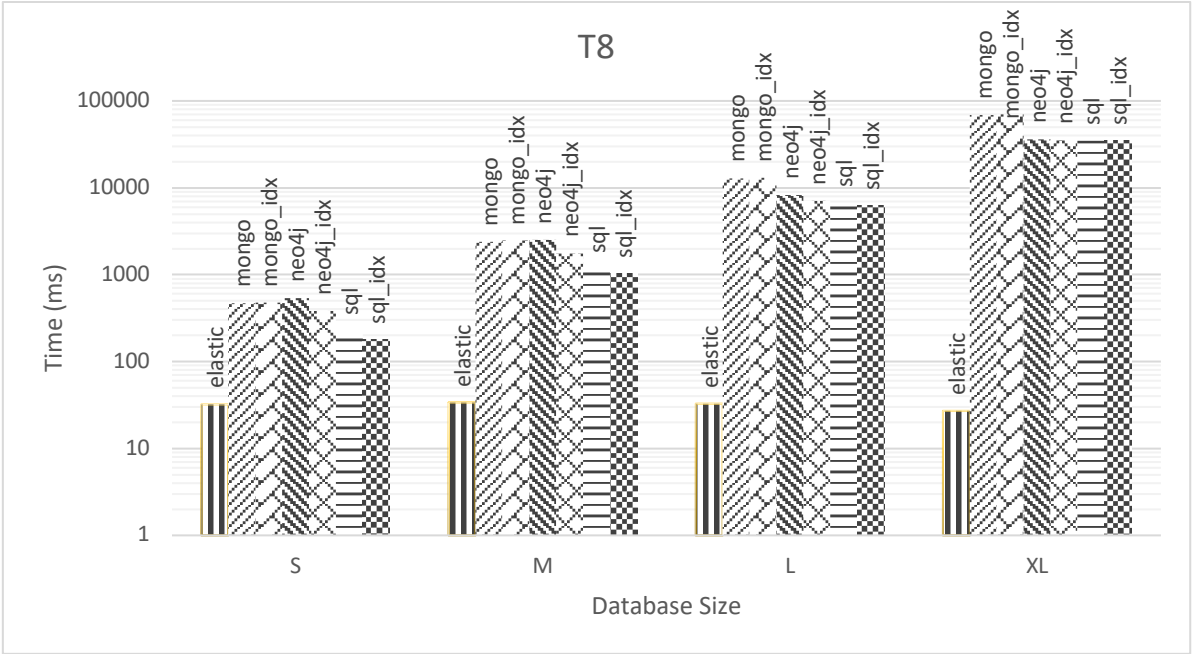
Στο συγκεκριμένο ερώτημα, στο οποίο ζητάμε όλες τις λέξεις σε ένας κείμενο που ξεκινάνε με τρία συγκεκριμένα γράμματα, παρατηρούμε ότι η MySQL με χρήση ευρετηρίου, έχει την ταχύτερη απόκριση και παρότι ο χρόνος επηρεάζεται από τον όγκο των δεδομένων, παραμένει σε πολύ χαμηλά επίπεδα και δε ξεπερνά τα 35ms στη χειρότερη περίπτωση όπου είναι και η μόνη περίπτωση που η Elasticsearch εμφανίζει καλύτερη επίδοση. Η MongoDB και η Neo4j αδυνατούν να πλησιάσουν αυτούς τους χρόνους καθώς δεν εκμεταλλεύονται το ευρετήριο κειμένου, με τη MongoDB όμως να παραμένει ταχύτερη από την MySQL όταν αυτή δε χρησιμοποιεί ευρετήριο.

Για τη χρήση του ερωτήματος σε βάση με μη cached data, η Elasticsearch είναι και πάλι η μόνη που καταφέρνει να αποκριθεί σε ανεκτούς χρόνους ανεξαρτήτως μεγέθους, επιστρέφοντας αποτελέσματα κοντά στα 250ms.

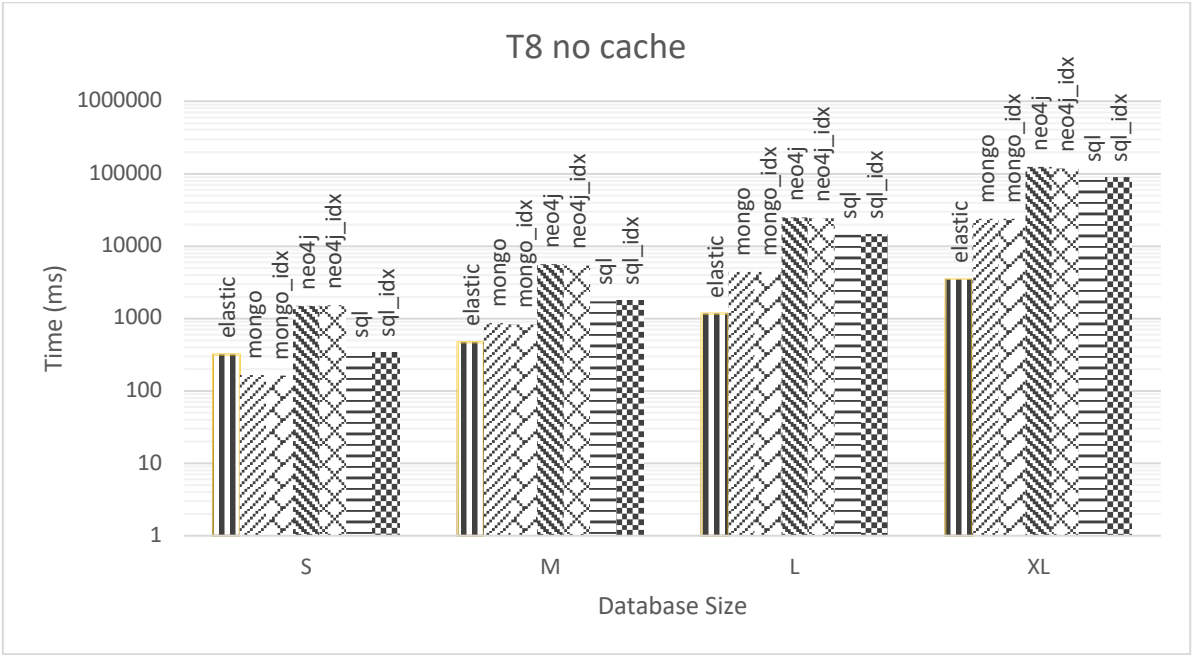
	Query T8	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	32.15	33.62	32.77	27.15
	MongoDB	470.46	2405.01	12877.89	69172.99
	MongoDB with Index	474.22	2461.19	12984.81	69480.22
	Neo4J	535.34	2520.3	8254.04	36055.57
	Neo4j with Index	382.07	1752.37	7125.06	35278.89
	MySQL	221.55	1150.05	6621.19	34608.49
	MySQL with Index	179.72	1021.71	6193.84	34892.13
Without Cache	ElasticSearch	402.95	640.89	1535.73	6332.28
	MongoDB	577	2936.71	14637.56	75829.07
	MongoDB with Index	567.47	2963.55	14485.37	75491.3
	Neo4J	907.73	2570.19	8690.39	36727.84
	Neo4j with Index	939.86	2546.85	8776.12	36338.84
	MySQL	224.78	1095.31	6274.47	34817.11

	MySQL with Index	228.25	1130.53	6412.8	35059.51

Αποτελέσματα για ερώτημα T8 σε ms



Διάγραμμα για ερώτημα T8 με cache



Διάγραμμα για ερώτημα T8 χωρίς cached data

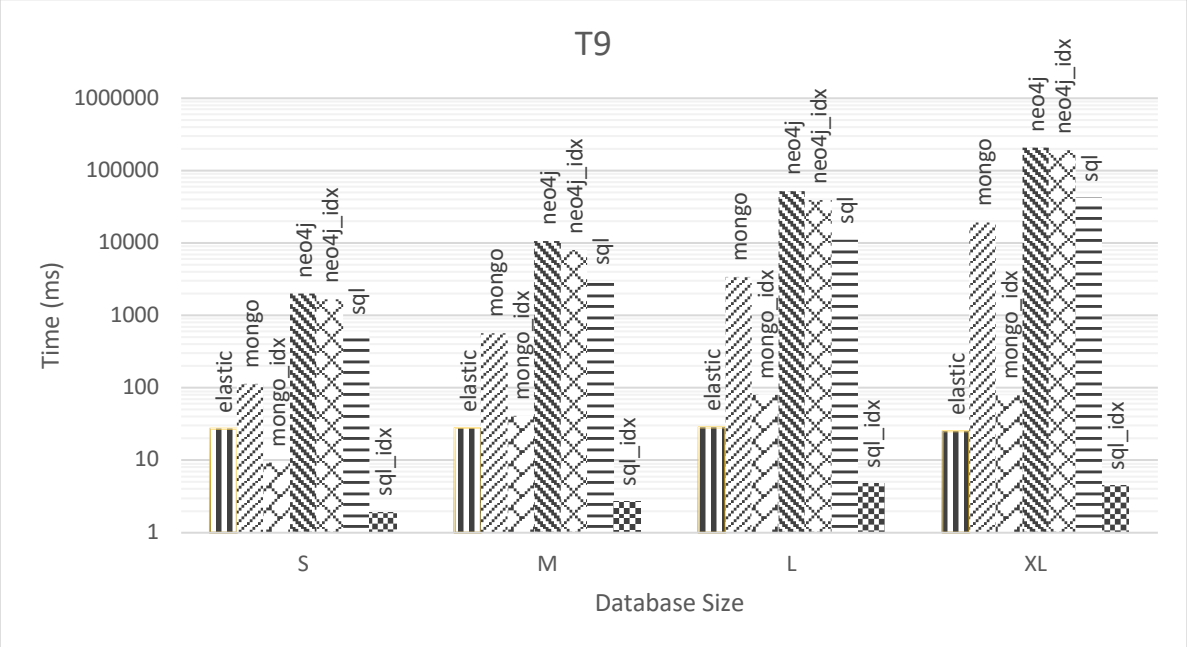
Μελετώντας τα αποτελέσματα για το συγκεκριμένο ερώτημα, στο οποίο ζητείται η εύρεση στο κείμενο μιας ολόκληρης λέξης είτε αυτούσιας είτε μέρος μια σύνθετης λέξη, είναι κατευθείαν προφανές ότι η Elasticsearch υπερτερεί έναντι όλων των άλλων συστημάτων κατά τάξεις μεγέθους ανεξάρτητα από τον όγκο των δεδομένων.

Όταν το ερώτημα εκτελείται σε βάση δεδομένων χωρίς cached data, η απόκριση είναι παρόμοια για όλα τα συστήματα πλην της Neo4j, όσο ο όγκος των δεδομένων μένει σχετικά

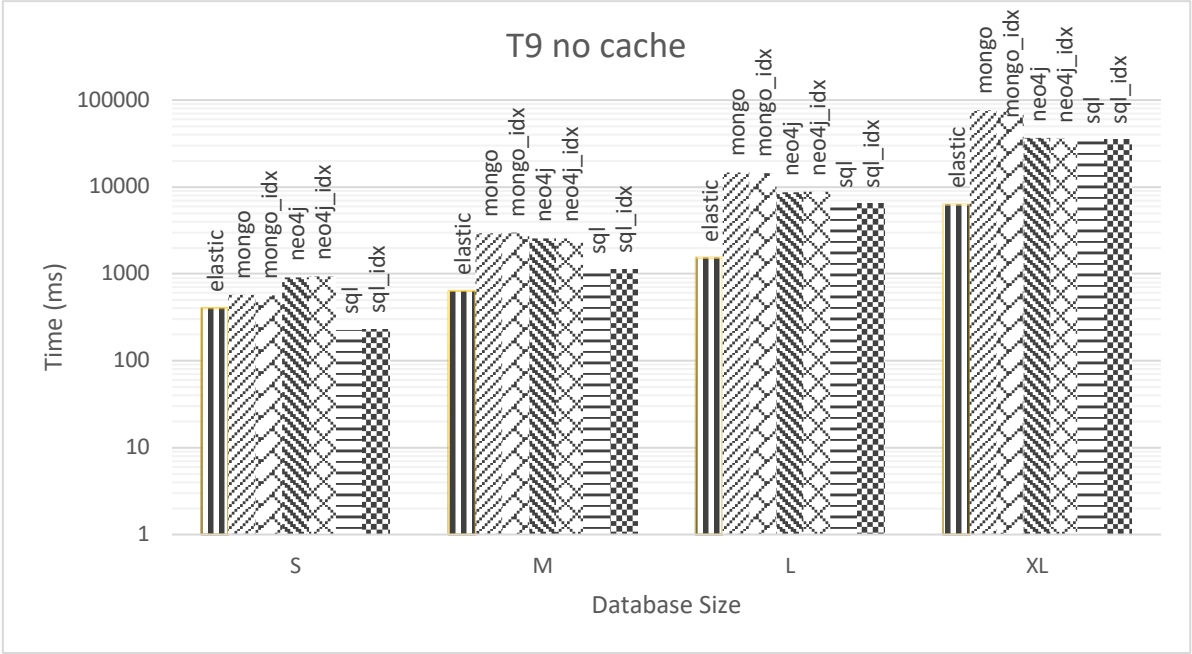
μικρός. Από κει και πέρα η Elasticsearch είναι η μόνη που καταφέρνει να κρατήσει την απόκριση κάτω από 10 δευτερόλεπτα.

	Query T9	*Database Size*			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	27.14	28	28.68	24.98
	MongoDB	112.98	571.96	3431.74	19425.8
	MongoDB with Index	9.22	40.86	80.25	79.26
	Neo4J	2001.52	10688.52	51940.63	209284.5
	Neo4j with Index	1667.97	8030.3	38905.73	191886
	MySQL	601.04	2986.59	11007.56	42625.44
	MySQL with Index	1.93	2.69	4.74	4.51
Without Cache	ElasticSearch	222.28	219.54	242.12	224.4
	MongoDB	212.91	1003.86	4924.03	25086.7
	MongoDB with Index	72.47	312.79	619.59	622.61
	Neo4J	2167.25	9138.49	40700.45	210670
	Neo4j with Index	2183.97	8836.92	41450.24	211166.7
	MySQL	550.44	2643.92	9995.35	42935.02
	MySQL with Index	567.27	2729.68	10267.76	43003.89

Αποτελέσματα για ερώτημα T9 σε ms



Διάγραμμα για ερώτημα T9 με cache



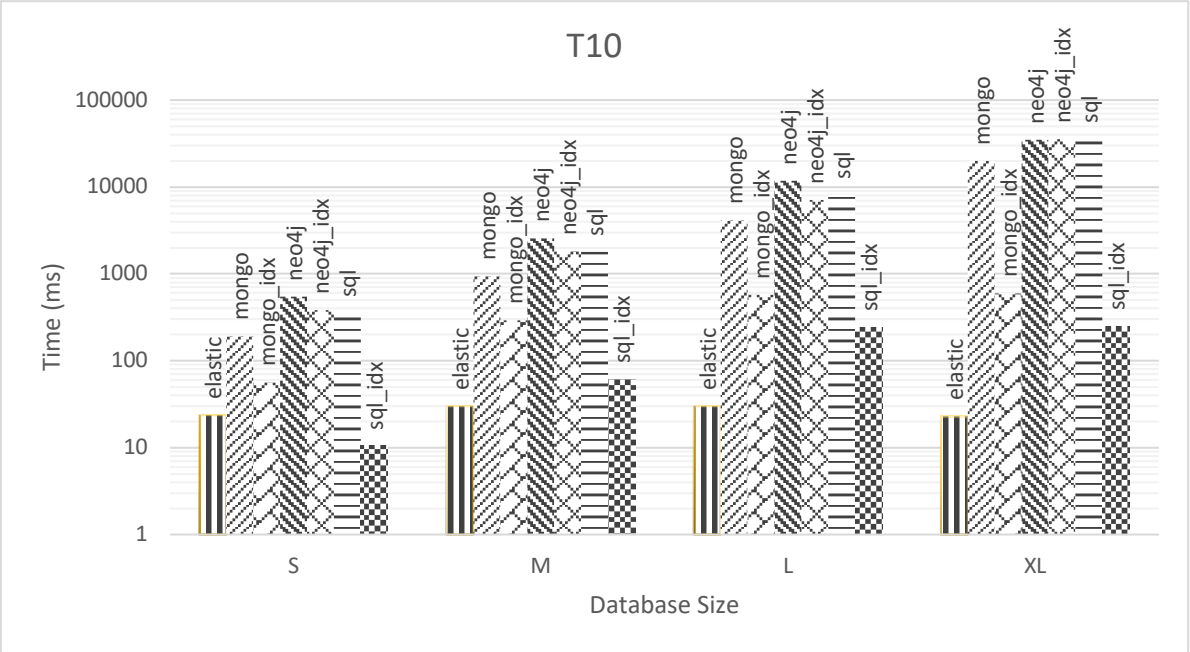
Διάγραμμα για ερώτημα T9 χωρίς cached data

Στο συγκεκριμένο ερώτημα, στην εύρεση αυτούσιων λέξεων μέσα στο κείμενο, η MySQL με χρήση ευρετηρίου έχει την ταχύτερη απόκριση από όλα τα συστήματα και φαίνεται ανεπηρέαστη από τον όγκο των δεδομένων απαντώντας στο ερώτημα σε κάτω από 5ms στη χειρότερη περίπτωση. Εξαιρετικά σταθερή απόκριση έχει και η Elasticsearch η οποία κυμαίνεται στα 25 - 28ms ενώ και η MongoDB με χρήση ευρετηρίου, παρότι είναι έως και 10 φορές πιο αργή από τη MySQL κρατάει την απόκριση στα 80ms στη χειρότερη περίπτωση. Αξίζει να σημειωθεί ότι στη μη χρήση ευρετηρίου, η MongoDB είναι ταχύτερη από την αντίστοιχη περίπτωση της MySQL και Neo4j

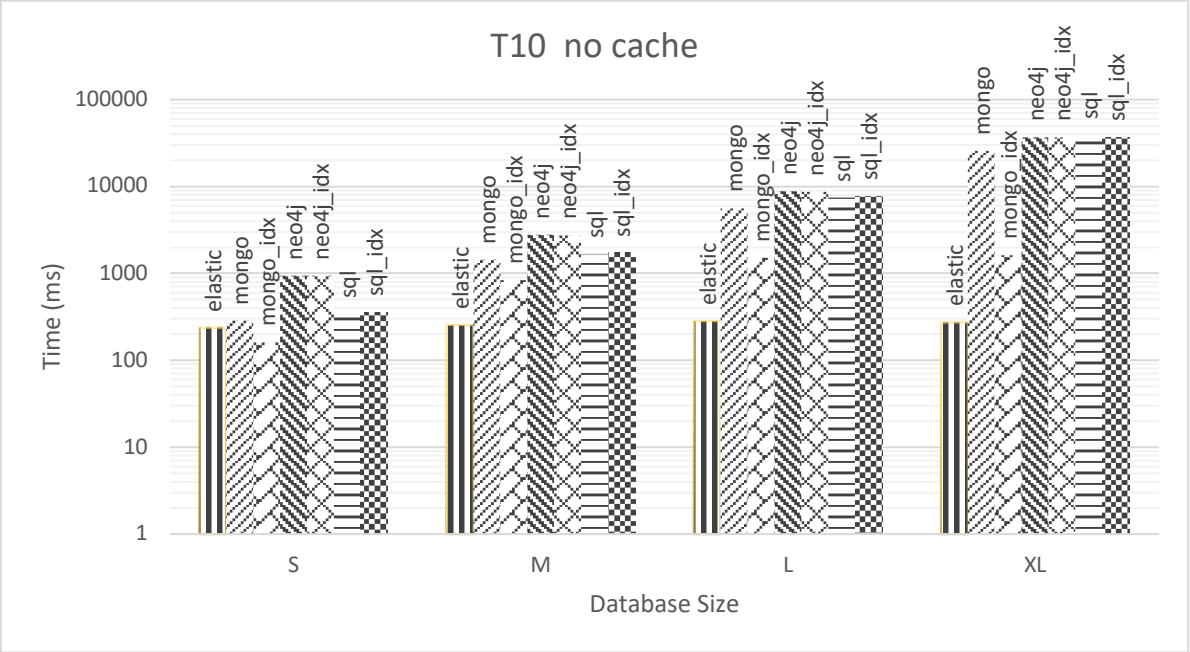
Για την εκτέλεση του ερωτήματος σε βάση χωρίς cached data, κανένα από τα συστήματα δε δείχνει να ξεχωρίζει έναντι του άλλου.

	Query T10	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	23.81	29.95	30.07	22.92
	MongoDB	189.65	937.9	4128	19892.44
	MongoDB with Index	55.91	291.41	571.69	595.8
	Neo4J	548.1	2576.79	11707.08	35155.16
	Neo4j with Index	384.83	1832.98	7121.04	35280.27
	MySQL	350.74	1807.01	8153.05	36350.77
	MySQL with Index	10.56	60.73	243.05	248.53
Without Cache	ElasticSearch	239.5	255.8	282.74	273.17
	MongoDB	285.38	1439.19	5562.54	25535.53
	MongoDB with Index	161.01	833.11	1514.98	1606.71
	Neo4J	940.56	2804.37	8785.71	36711.88
	Neo4j with Index	943.83	2726.4	8730.32	36477.07
	MySQL	342.34	1680.64	7474.19	38379.49
	MySQL with Index	355.76	1734.88	7756.89	36766.06

Αποτελέσματα για ερώτημα T10 σε ms



Διάγραμμα για ερώτημα T10 με cache



Διάγραμμα για ερώτημα T10 χωρίς cached data

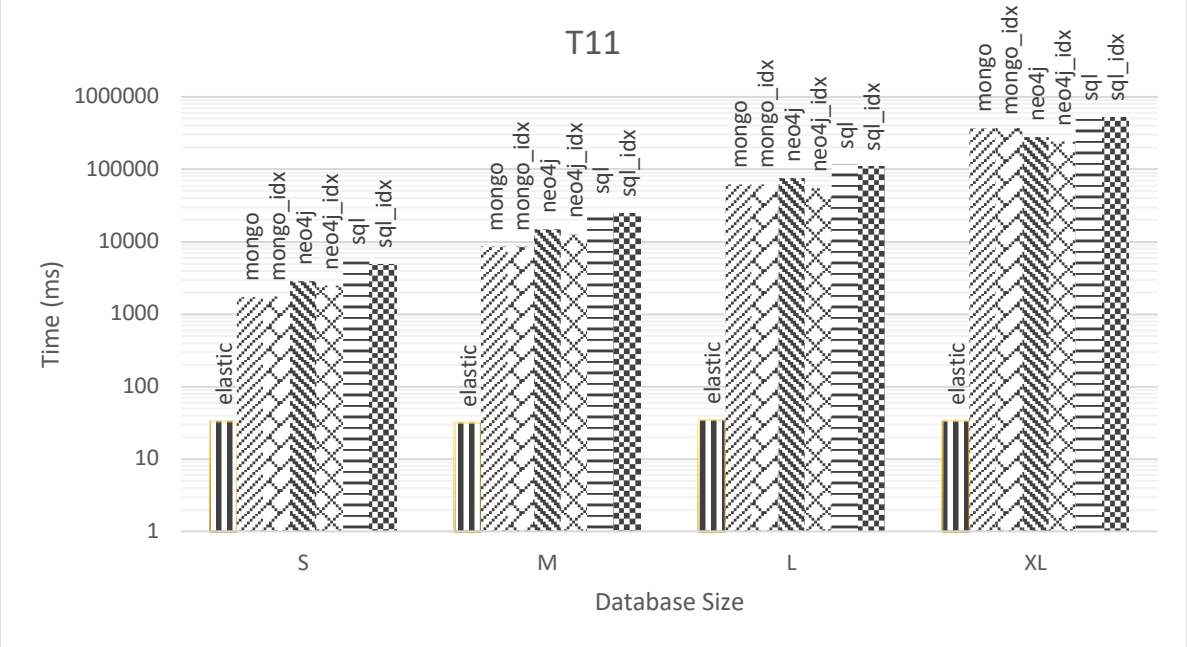
Στο συγκεκριμένο ερώτημα για την εύρεση φράσης μέσα σε κείμενο και πάλι η Elasticsearch δείχνει σταθερή και καλή απόκριση στα 20 με 30 ms. Η MySQL με χρήση ευρετηρίου αποκρίνεται σχετικά γρήγορα για μικρό και μεσαίο όγκο δεδομένων. Η MongoDB με τη χρήση ευρετηρίου είναι μια τάξη μεγέθους πιο γρήγορη από τη MongoDB χωρίς το ευρετήριο όμως σε καμία περίπτωση δε πλησιάζει την ταχύτητα της Elasticsearch.

Στην εκτέλεση του ερωτήματος χωρίς cached data η Elasticsearch είναι η μόνη που καταφέρνει να κρατήσει μια σταθερή απόδοση η οποία μάλιστα είναι και σχετικά μικρή σε σχέση με τα άλλα συστήματα, γύρω στα 250 ms.

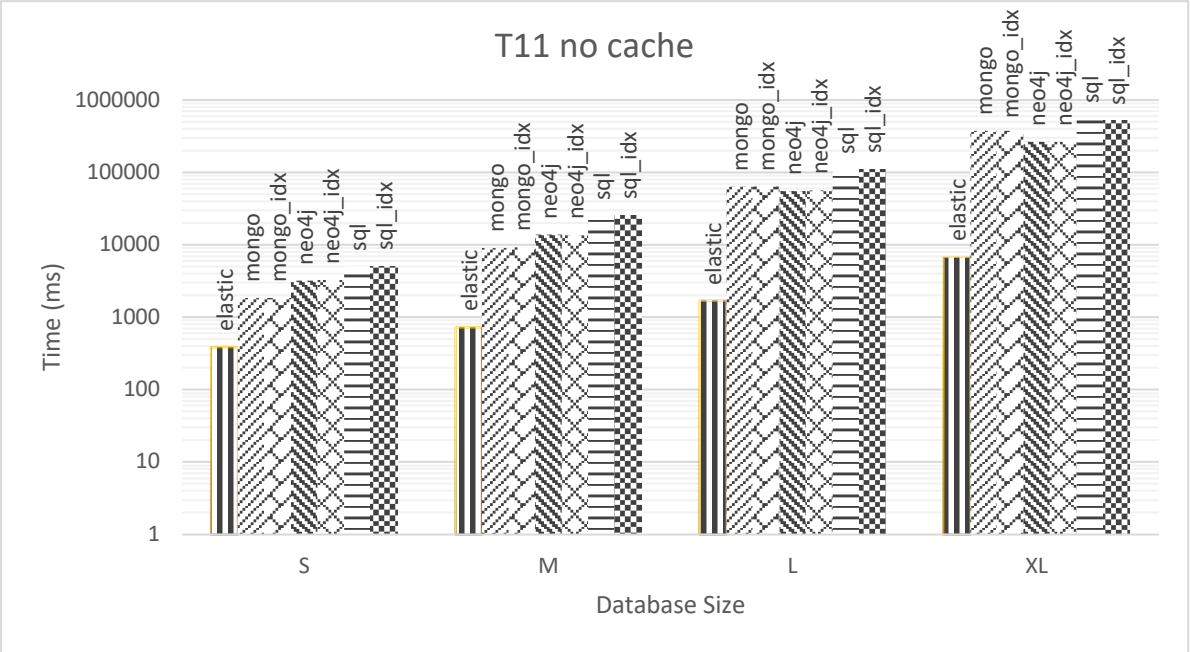
Query T11	***Database Size***

	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	33.3	31.67	34.79	33.82
	MongoDB	1719.79	8621.07	61871.53	369970
	MongoDB with Index	1748.5	8667.43	62224.37	371253.2
	Neo4J	2842.06	14800.05	76323.6	278987.2
	Neo4j with Index	2495.57	12708.89	54668.46	241600.8
	MySQL	5341.17	27293.68	116623.3	522332
	MySQL with Index	4933.16	24878.29	108599.2	522088.6
Without Cache	ElasticSearch	382.23	723.65	1711.04	6807.05
	MongoDB	1855.21	9096.66	64064.72	379199.3
	MongoDB with Index	1835.79	9175.6	63829.97	378005.2
	Neo4J	3227.41	13966.88	55486.19	265400
	Neo4j with Index	3267.33	13628.84	56787.97	264164.1
	MySQL	4978.67	24881.9	108837.8	538583.4
	MySQL with Index	5081.91	25438.52	109855.2	522660.5

Αποτελέσματα για ερώτημα T11 σε ms



Διάγραμμα για ερώτημα T10 με cache



Διάγραμμα για ερώτημα T11 χωρίς cached data

Στο συγκεκριμένο ερώτημα για την εύρεση μιας λέξης με 35 χαρακτήρες, μόνο η Elasticsearch καταφέρνει να αποκριθεί γρήγορα και μάλιστα ανεπηρέαστη από των όγκο των δεδομένων. Όλα τα υπόλοιπα συστήματα ανταποκρίνονται εμφανώς πιο αργά, με τη διαφορά να φτάνεις 3 έως 4 τάξεις μεγέθους όταν ο όγκος δεδομένων γίνει πολύ μεγάλος.

Για τη χρήση του ερωτήματος σε βάση με μη cached data, η Elasticsearch είναι πάντα μια τάξη χρόνου ταχύτερη με το χρόνο απόκρισης να βρίσκεται στα δευτερόλεπτα όταν το μέγεθος των δεδομένων αυξάνεται.

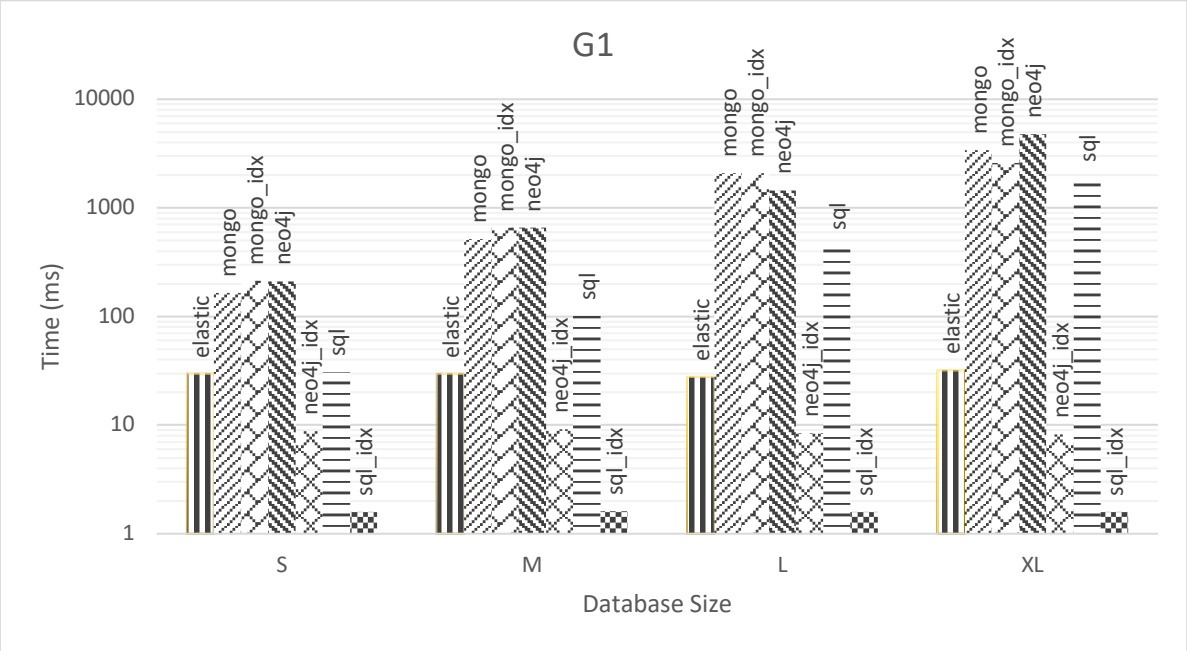
4.8.2 Μετρήσεις στα ερωτήματα σε γεωγραφικά δεδομένα

Παρακάτω ακολουθούν τα αποτελέσματα για τα ερωτήματα G1-G13 που περιεγράφηκαν στο κεφάλαιο 4.5.2.

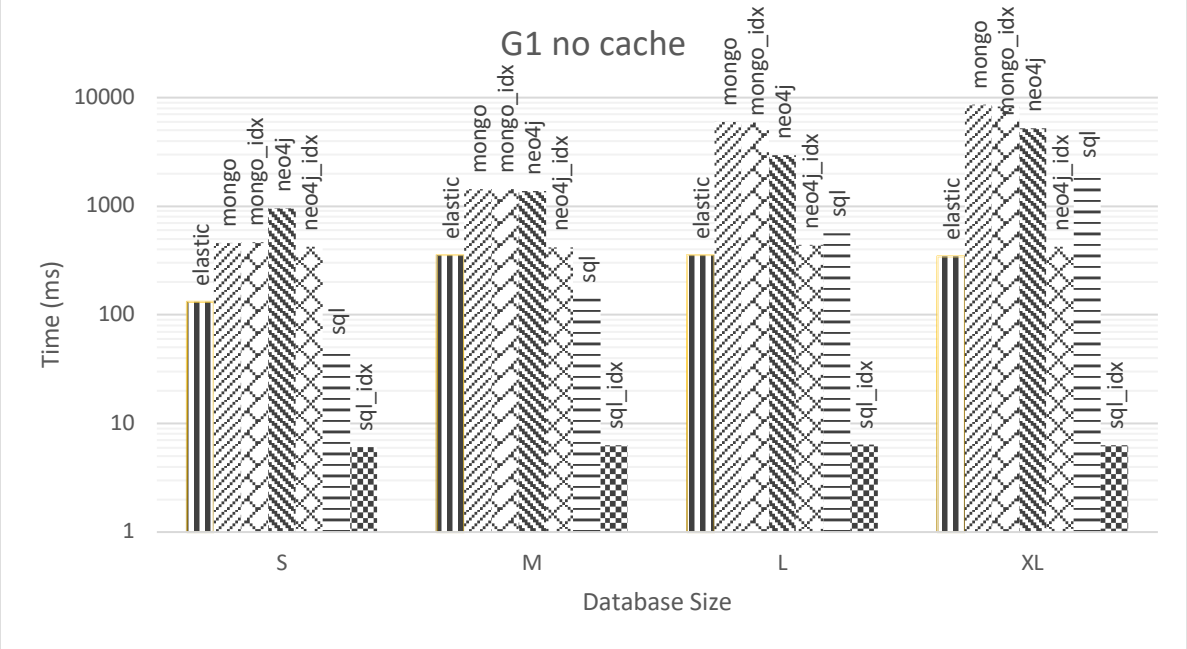
	Query G1	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	29.84	29.98	27.81	31.96
	MongoDB	164.66	512.1	2089.69	3403.6
	MongoDB with Index	214.36	654.9	2076.81	2602.73
	Neo4J	211.46	661.39	1437.27	4800.83
	Neo4j with Index	8.81	9.19	8.47	8.22
	MySQL	30.71	100.97	450.41	1904.02
	MySQL with Index	1.58	1.61	1.58	1.58
Without Cache	Elasticsearch	132.35	353.69	352.95	347.26
	MongoDB	458.75	1430.43	5944.88	8638.79
	MongoDB with Index	466.37	1434.81	5878.15	8304.77
	Neo4J	962.07	1375.54	2958.09	5209.87
	Neo4j with Index	423.74	419.41	438.64	420.81

	MySQL	46.28	140.29	566.44	2001.27
	MySQL with Index	6.05	6.27	6.35	6.28

Αποτελέσματα για ερώτημα G1 σε ms



Διάγραμμα για ερώτημα G1 με cache



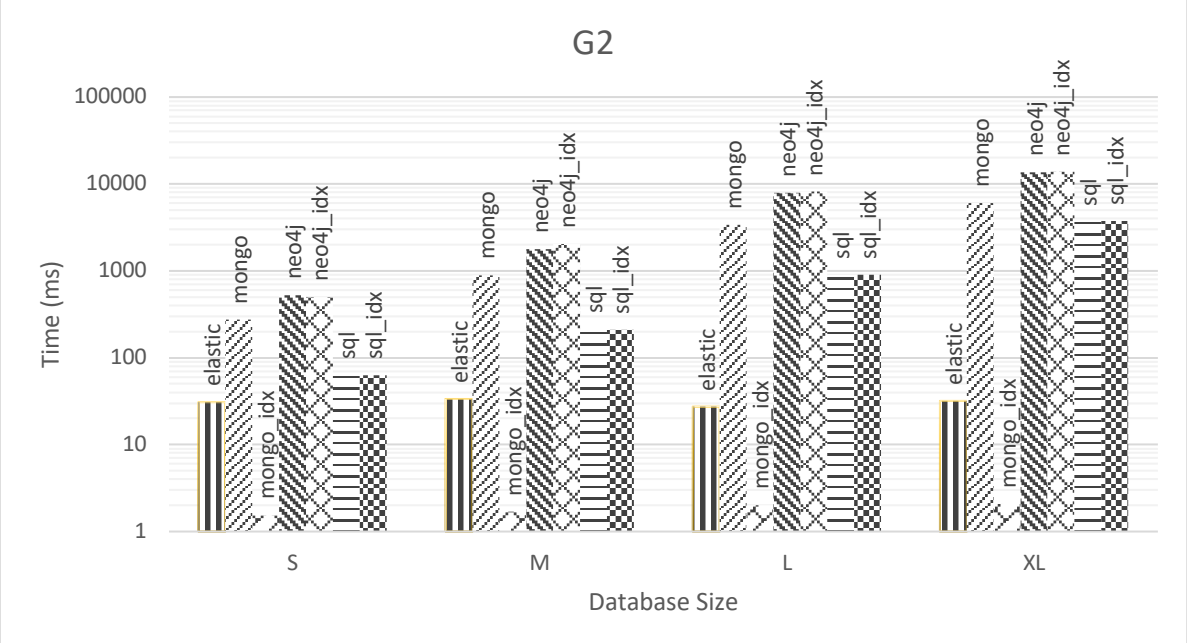
Διάγραμμα για ερώτημα G1 χωρίς cached data

Για το συγκεκριμένο ερώτημα, την εύρεση ενός συγκεκριμένου σημείου, είναι φανερό ότι η MySQL με χρήση ευρετηρίου έχει την πιο γρήγορη απόκριση η οποία μένει ανεπηρέαστη από το μέγεθος των δεδομένων. Το ίδιο συμβαίνει και για τη Neo4j με χρήση ευρετηρίου, η οποία αν και είναι 4 φορές πιο αργή από τη MySQL παραμένει κάτω από τα 10 ms, αλλά και με την Elasticsearch η οποία επιστρέφει αποτελέσματα στα 30 ms. Η MongoDB είναι αισθητά πιο αργή ακόμα και από τη MySQL χωρίς ευρετήριο.

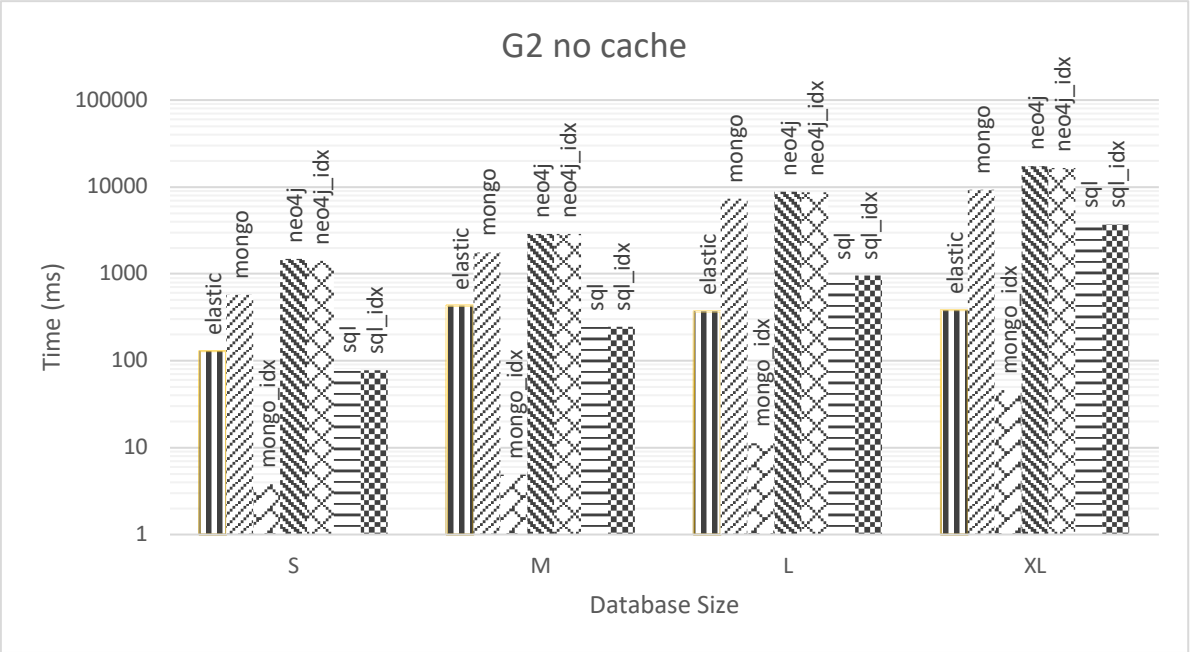
Όσο αφορά τη χρήση του ερωτήματος χωρίς cached data, η MySQL με χρήση ευρετηρίου είναι τάξεις μεγέθους πιο αποτελεσματική από όλα τα άλλα συστήματα, επιστρέφοντας μάλιστα αποτελέσματα σε λιγότερο από 10 ms.

	Query G2	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	30.85	33.61	27.33	31.91
	MongoDB	277.02	879.03	3358.56	6063.75
	MongoDB with Index	1.54	1.7	1.99	2.09
	Neo4J	522.49	1774.56	7872.4	13588.63
	Neo4j with Index	500.75	2023.92	8159.59	13862.4
	MySQL	62.89	213.83	897.67	3716.18
	MySQL with Index	63.01	210.67	904.99	3768.99
Without Cache	ElasticSearch	128.66	433.81	367.42	384.39
	MongoDB	574.35	1765.64	7281.91	9263.67
	MongoDB with Index	3.76	4.86	11.18	45.55
	Neo4J	1475.04	2900.72	8808.67	17303.07
	Neo4j with Index	1402.73	2863.66	8681.36	16476.39
	MySQL	79.9	246.42	962.64	3694.36
	MySQL with Index	78.32	245.32	960.76	3740.04

Αποτελέσματα για ερώτημα G2 σε ms



Διάγραμμα για ερώτημα G2 με cache



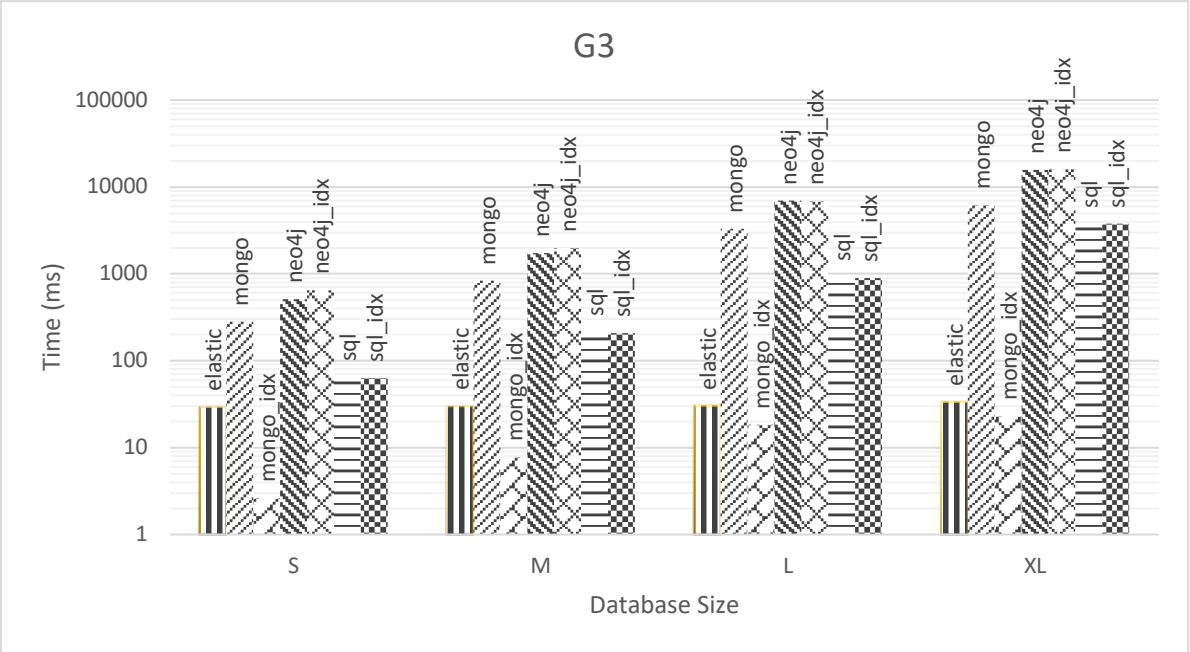
Διάγραμμα για ερώτημα G2 χωρίς cached data

Για το ερώτημα G2, την εύρεση σημείων σε ακτίνα 100 μέτρων, η χρήση του δυσδιάστατου σφαιρικού ευρετηρίου της MongoDB επιτρέπει την επιστροφή δεδομένων σε ελάχιστο χρόνο ανεξαρτήτως του όγκου δεδομένων. Η Elasticsearch έχει σταθερή απόδοση με απόκριση περίπου 30ms ενώ η MySQL μπορεί να ανταγωνιστεί τα δυο προαναφερθέντα συστήματα μόνο για μικρό όγκο δεδομένων.

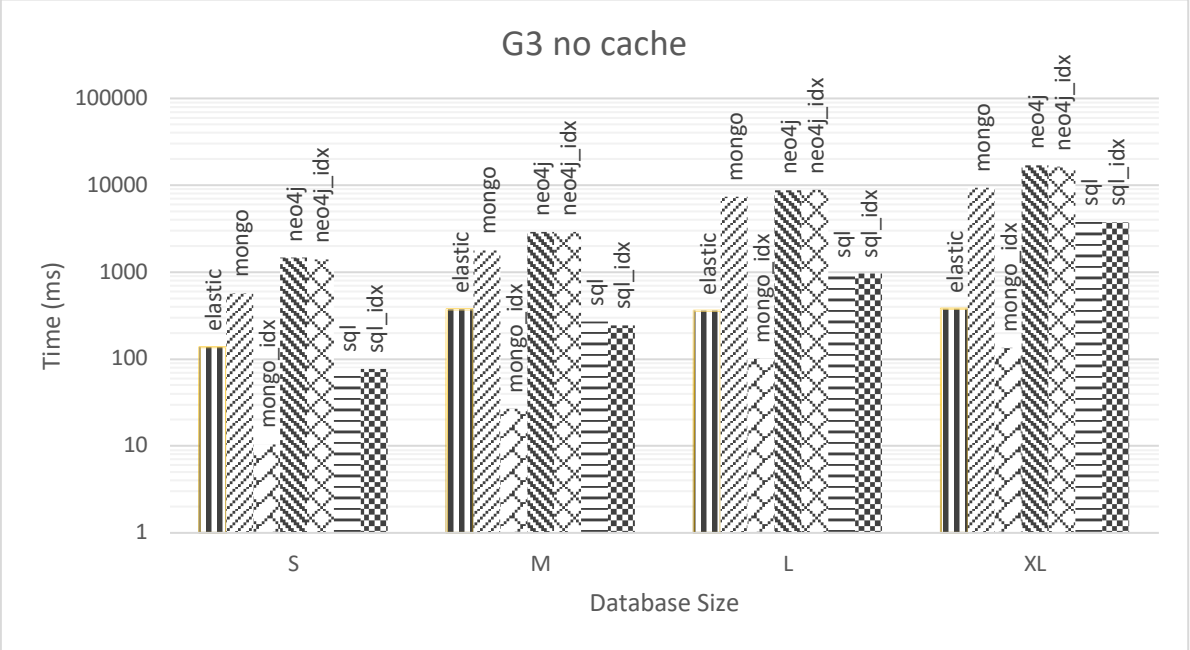
Η MongoDB με χρήση ευρετηρίου δείχνει επίσης πολύ καλή συμπεριφορά και απόκριση ακόμα και σε μη cached data, έχοντας μεγάλη διαφορά με τα υπόλοιπα συστήματα.

	Query G3	**Database Size**			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	29.45	30.16	30.46	33.81
	MongoDB	282.09	830.29	3317.09	6221.41
	MongoDB with Index	2.63	7.79	18.51	23.03
	Neo4J	513.68	1738.53	6933.53	15825.68
	Neo4j with Index	643.93	1983.07	6800.33	15999.92
	MySQL	64.08	215.68	907.22	3799.97
	MySQL with Index	63.47	209.52	897.23	3767.21
Without Cache	ElasticSearch	137.47	372.4	360.9	381.22
	MongoDB	575.05	1788.89	7339.84	9366.78
	MongoDB with Index	10.34	26.8	100.26	135.66
	Neo4J	1481.2	2893.56	8727.96	17074.84
	Neo4j with Index	1395.47	2885.76	8917.49	16430.01
	MySQL	78.98	269.92	960.56	3797.16
	MySQL with Index	77.54	242.95	965.94	3766.94

Αποτελέσματα για ερώτημα G3 σε ms



Διάγραμμα για ερώτημα G3 με cache



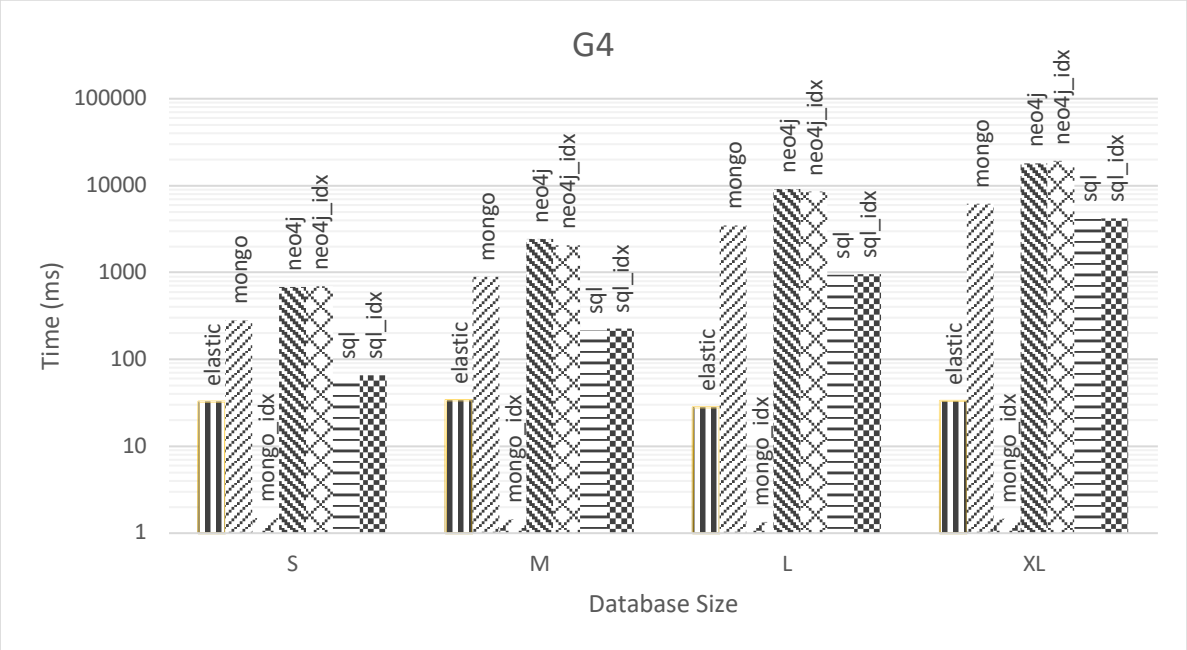
Διάγραμμα για ερώτημα G3 χωρίς cached data

Για το ερώτημα G3, την εύρεση σημείων σε ακτίνα 500 μέτρων, παρατηρούμε την ίδια συμπεριφορά με το query G2.

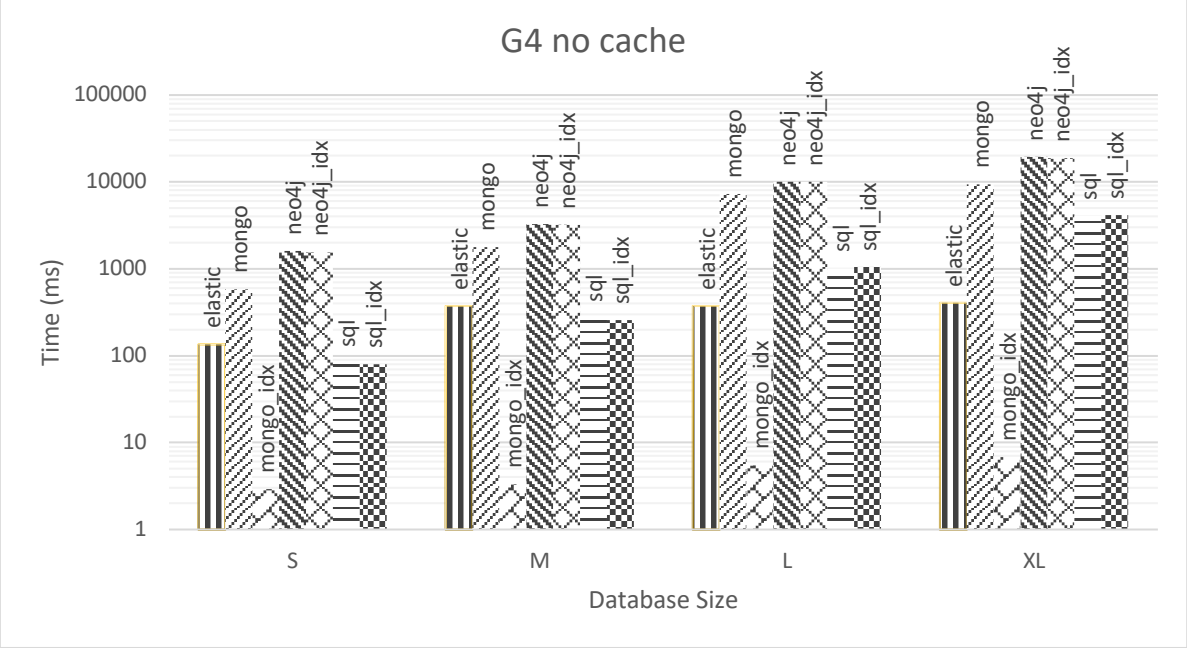
	Query G4	**Database Size**			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	32.54	33.85	28.27	33.2
	MongoDB	281.81	901.41	3479.31	6219.96
	MongoDB with Index	1.48	1.44	1.35	1.46
	Neo4J	685.95	2417	9145.99	17910.65
	Neo4j with Index	695.31	2065.65	8518.18	19340.82

	MySQL	65.96	215.57	939.48	4167.51
	MySQL with Index	65.87	224.79	964.37	4210.36
Without Cache	Elasticsearch	134.21	374.71	371.51	405.28
	MongoDB	578.78	1779.62	7279.44	9405.86
	MongoDB with Index	2.9	3.35	5.45	6.86
	Neo4J	1612.21	3254.59	10000.62	19321.82
	Neo4j with Index	1540.97	3212.38	10059.56	18964.15
	MySQL	81.99	273.48	1023.51	4111.74
	MySQL with Index	80.32	256.31	1044.92	4152.51

Αποτελέσματα για ερώτημα G4 σε ms



Διάγραμμα για ερώτημα G4 με cache

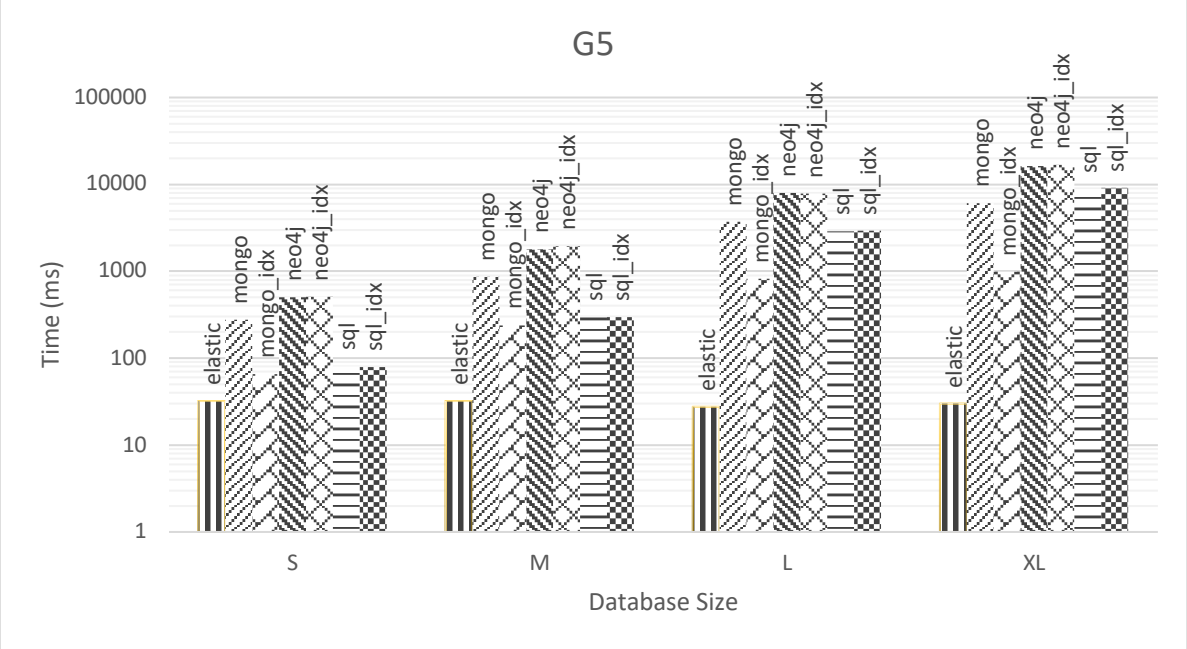


Διάγραμμα για ερώτημα G4 χωρίς cached data

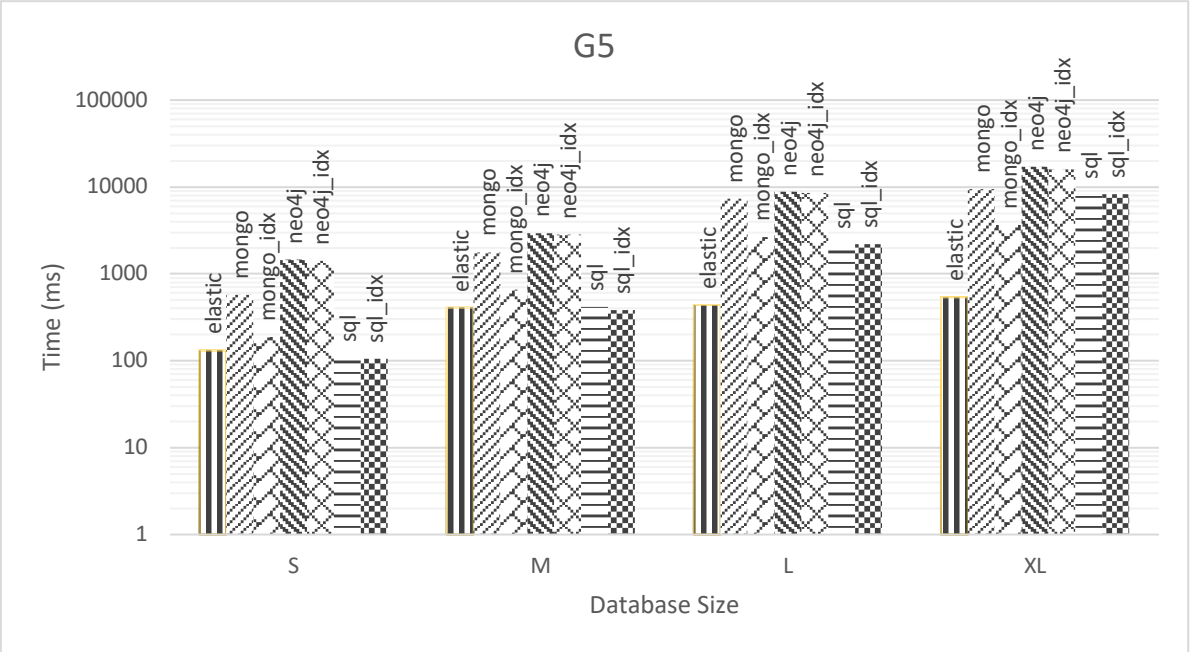
Για το ερώτημα G4, την εύρεση σημείων σε ακτίνα 1000 μέτρων, παρατηρούμε την ίδια συμπεριφορά με το query G3. Αυτό που προκαλεί εντύπωση είναι πως στο σύστημα της MongoDB με δυσδιάστατο ευρετήριο, το ερώτημα έτρεξε όσο γρήγορα έτρεξε για κύκλο με ακτίνα 100 μέτρα (G2) και γρηγορότερα από ότι σε κύκλο ακτίνας 500 μέτρων (G3).

	Query G5	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	32.32	32.05	27.75	29.85
	MongoDB	277.39	859.97	3687.68	6061.92
	MongoDB with Index	65.55	239.76	816.83	993.8
	Neo4J	502.97	1789.19	7980.33	16245.46
	Neo4j with Index	510.69	1959.35	7851.02	16874.31
	MySQL	79.76	300.24	2897.53	8807.2
	MySQL with Index	79.14	295.55	2922.99	9061.66
Without Cache	Elasticsearch	132.08	409.15	436.26	539.75
	MongoDB	574.57	1765.77	7370.91	9388.67
	MongoDB with Index	189.23	658.52	2655.81	3631.23
	Neo4J	1470.09	2907.65	8896.54	17079.57
	Neo4j with Index	1403.82	2824.3	8588.67	15902.48
	MySQL	107.81	414.94	2127.28	8131.21
	MySQL with Index	105.73	381.78	2215.05	8195.44

Αποτελέσματα για ερώτημα G5 σε ms



Διάγραμμα για ερώτημα G5 με cache

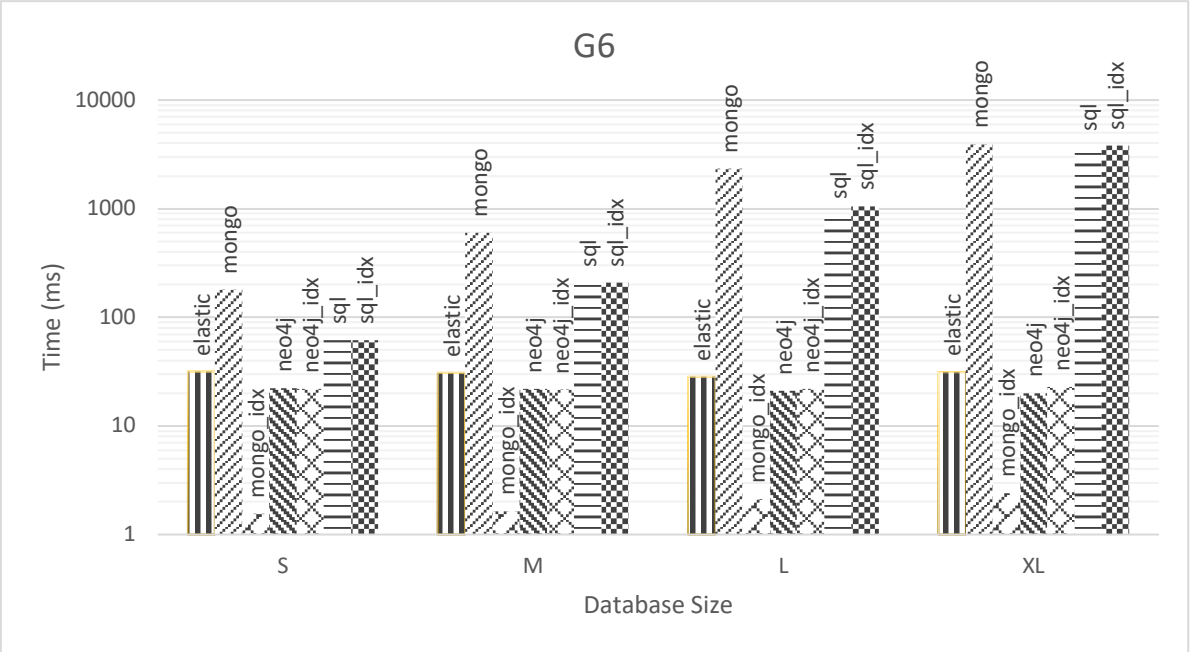


Διάγραμμα για ερώτημα G5 χωρίς cached data

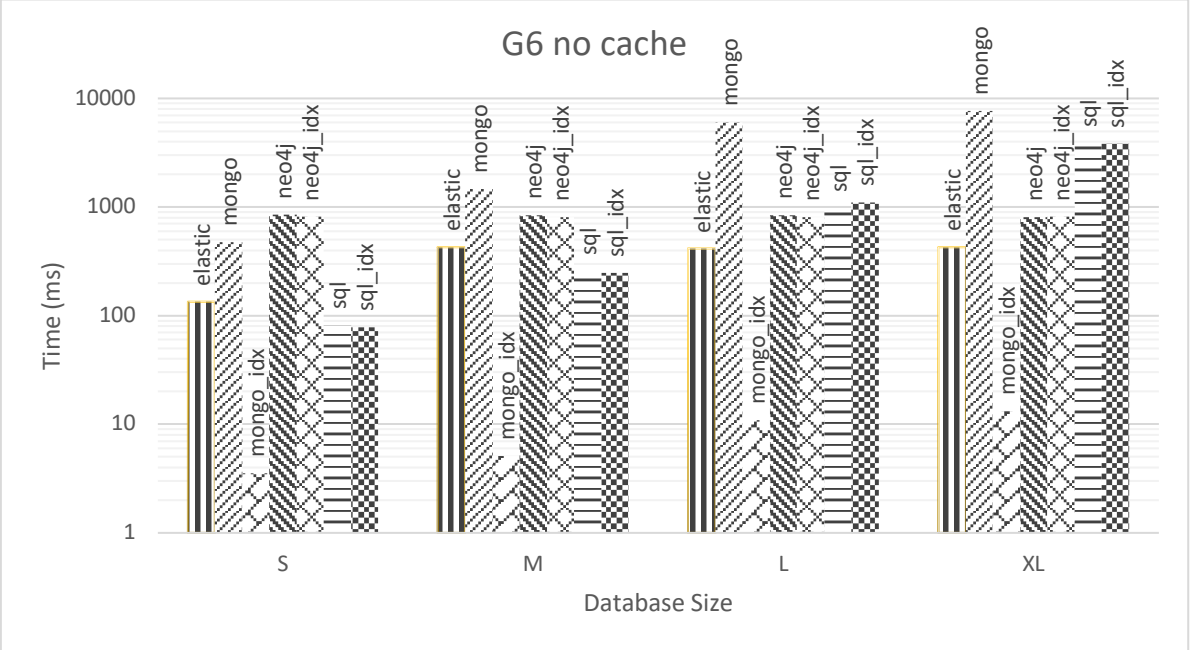
Για το ερώτημα G5, την εύρεση σημείων σε ακτίνα 2500 μέτρων, είναι η πρώτη φορά που παρατηρούμε τη MongoDB με ευρετήριο να έχει χειρότερη απόκριση από κάποιο άλλο σύστημα. Αυτό το σύστημα είναι η Elasticsearch η οποία σταθερά και πάλι επιστρέφει δεδομένα στα 30 ms. Η MongoDB με το δυσδιάστατο ευρετήριο έχει την ίδια απόδοση με τη MySQL μέχρι τη βάση δεδομένων με το μεγαλύτερο μέγεθος, εκεί που η MySQL είναι 10 φορές πιο αργή.

	Query G6	***Database Size***			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	31.53	30.94	28.47	31.15
	MongoDB	180.71	602.75	2327.59	3891.97
	MongoDB with Index	1.55	1.64	2.13	2.4
	Neo4J	22.37	22.05	20.93	19.94
	Neo4j with Index	21.96	21.71	21.82	23
	MySQL	61.75	207.54	877.31	3702.13
	MySQL with Index	61.62	208.96	1054.74	3809.53
Without Cache	Elasticsearch	134.52	425.76	412.79	427.59
	MongoDB	472.38	1463.04	5974.02	7623.9
	MongoDB with Index	3.52	5.09	10.92	13.17
	Neo4J	855.66	834.94	834.19	801.49
	Neo4j with Index	820.96	809.66	806.08	815.31
	MySQL	81.4	258.64	972.68	3907.58
	MySQL with Index	78.31	247.03	1105.29	3847.7

Αποτελέσματα για ερώτημα G6 σε ms



Διάγραμμα για ερώτημα G6 με cache

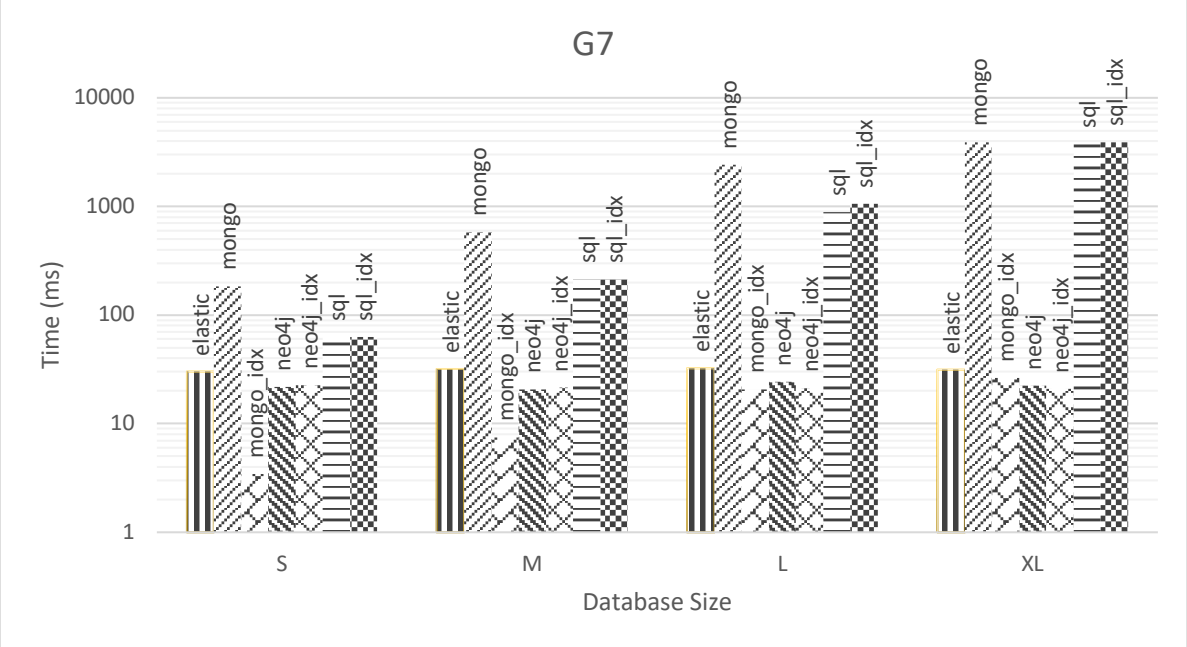


Διάγραμμα για ερώτημα G6 χωρίς cached data

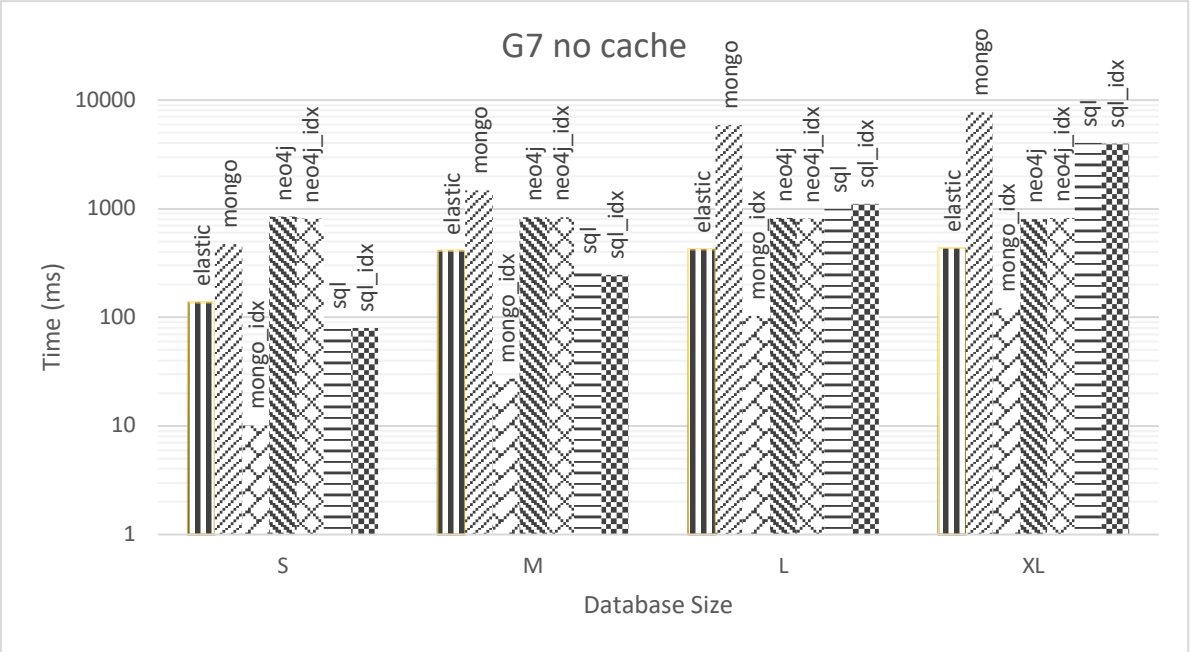
Για το ερώτημα G4, την εύρεση σημείων σε ακτίνα 100 μέτρων που ανήκουν σε κάποια συγκεκριμένη κατηγορία παρατηρούμε ότι πάλι η MongoDB με χρήση ευρετηρίου έχει την ταχύτερη απόκριση. Η διαφορά αυτή τη φόρα, είναι πως η Neo4j, έχει την ίδια γρήγορη και σταθερή απόκριση με την Elasticsearch, με ή χωρίς τη χρήση ευρετηρίου. Αυτό οφείλεται στον τρόπο με τον οποίο είναι γραμμένα τα ερωτήματα στη Neo4j, αφού πρώτα προσπαθεί να προσπελάσει τους κόμβους των σημείων που συνδέονται με μια κατηγορία μέσω μιας ακμής/σχέσης και στη συνέχεια να φιλτράρει ως προς τα γεωγραφικά δεδομένα.

	Query G7	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	30	31.97	32.2	31.25
	MongoDB	183.59	581.33	2418.77	3870.66
	MongoDB with Index	3.42	7.51	20.59	26.36
	Neo4J	21.82	20.71	24.16	22.4
	Neo4j with Index	22.76	21.61	21.1	20.99
	MySQL	62.72	212.54	890.35	3810.93
	MySQL with Index	63.17	213.75	1062.77	3904.54
Without Cache	Elasticsearch	137.39	409.94	422.64	432.91
	MongoDB	469.78	1468.52	5924.06	7730.66
	MongoDB with Index	10.14	27.42	103.19	120.66
	Neo4J	845.26	828.57	820.38	798.93
	Neo4j with Index	815.14	829.47	807.32	824.91
	MySQL	83.91	268.98	996.93	4014.17
	MySQL with Index	79.29	246.76	1112.65	3963.46

Αποτελέσματα για ερώτημα G7 σε ms



Διάγραμμα για ερώτημα G7 με cache

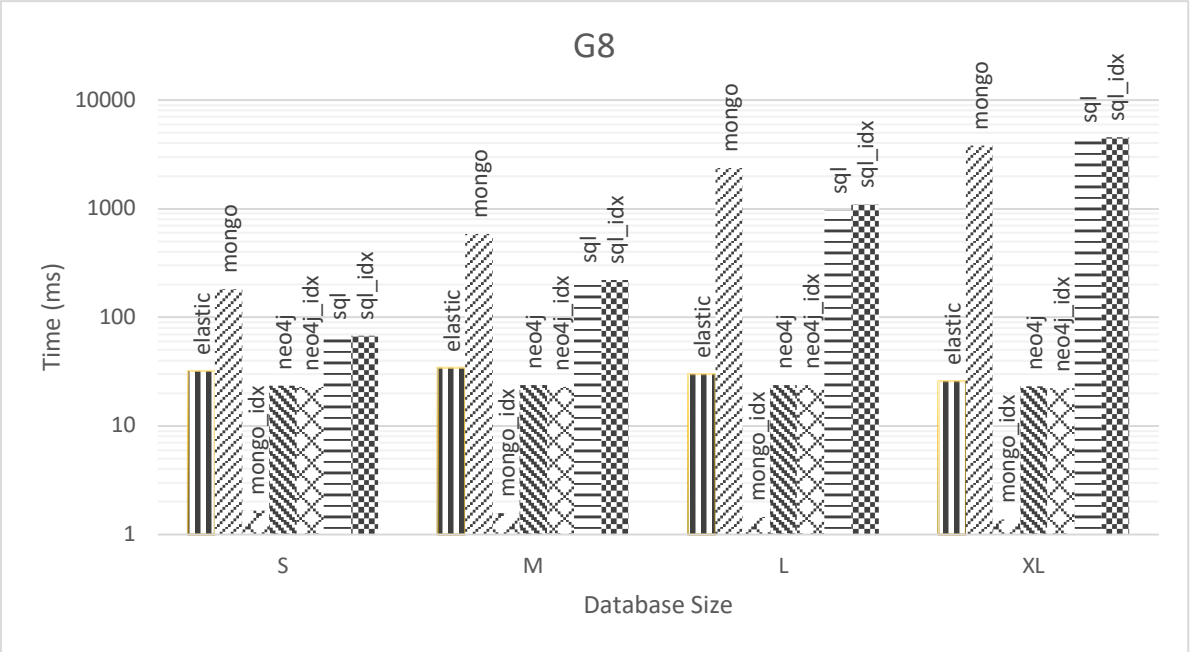


Διάγραμμα για ερώτημα G7 χωρίς cached data

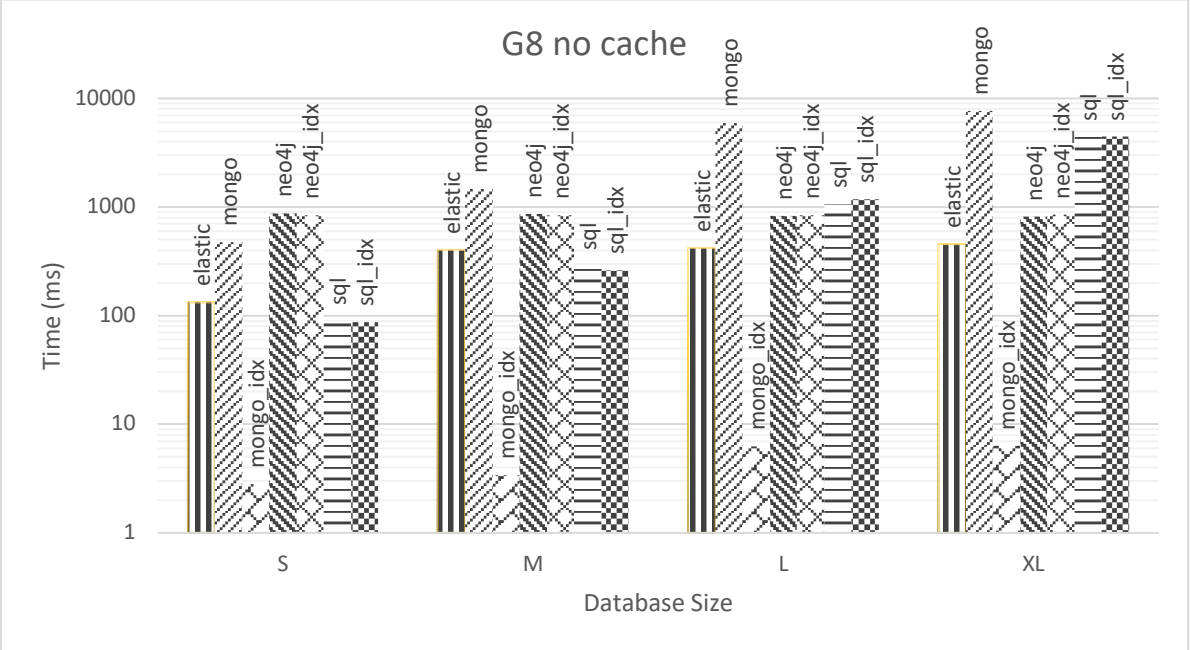
Για το ερώτημα G7, την εύρεση σημείων σε ακτίνα 500 μέτρων που ανήκουν σε κάποια συγκεκριμένη κατηγορία παρατηρούμε τα ίδια αποτελέσματα με το ερώτημα G6.

	Query G8	**Database Size**			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	31.96	34.37	29.86	25.93
	MongoDB	182.93	585.52	2369.33	3795.75
	MongoDB with Index	1.67	1.57	1.46	1.38
	Neo4J	23.49	23.87	23.73	23.01
	Neo4j with Index	22.74	22.97	23.67	22.34
	MySQL	67.56	223.57	966.27	4380.46
	MySQL with Index	68.04	221.54	1098.76	4526.7
Without Cache	Elasticsearch	132.27	400.84	422.64	454.09
	MongoDB	474.63	1465.77	5913.02	7712.54
	MongoDB with Index	2.82	3.41	6.28	6.37
	Neo4J	876	863.87	833.98	819.83
	Neo4j with Index	840.21	835.49	842.79	849.56
	MySQL	88.9	282.72	1058.16	4434.66
	MySQL with Index	87.22	262.3	1179.88	4450.36

Αποτελέσματα για ερώτημα G8 σε ms



Διάγραμμα για ερώτημα G8 με cache

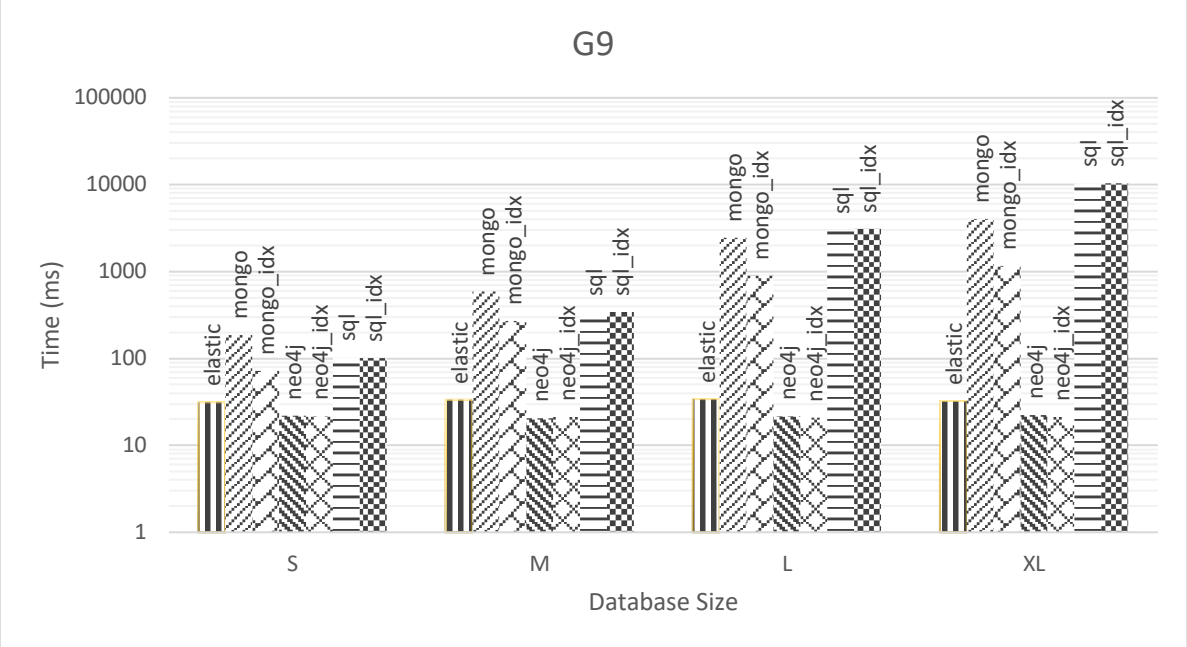


Διάγραμμα για ερώτημα G8 χωρίς cached data

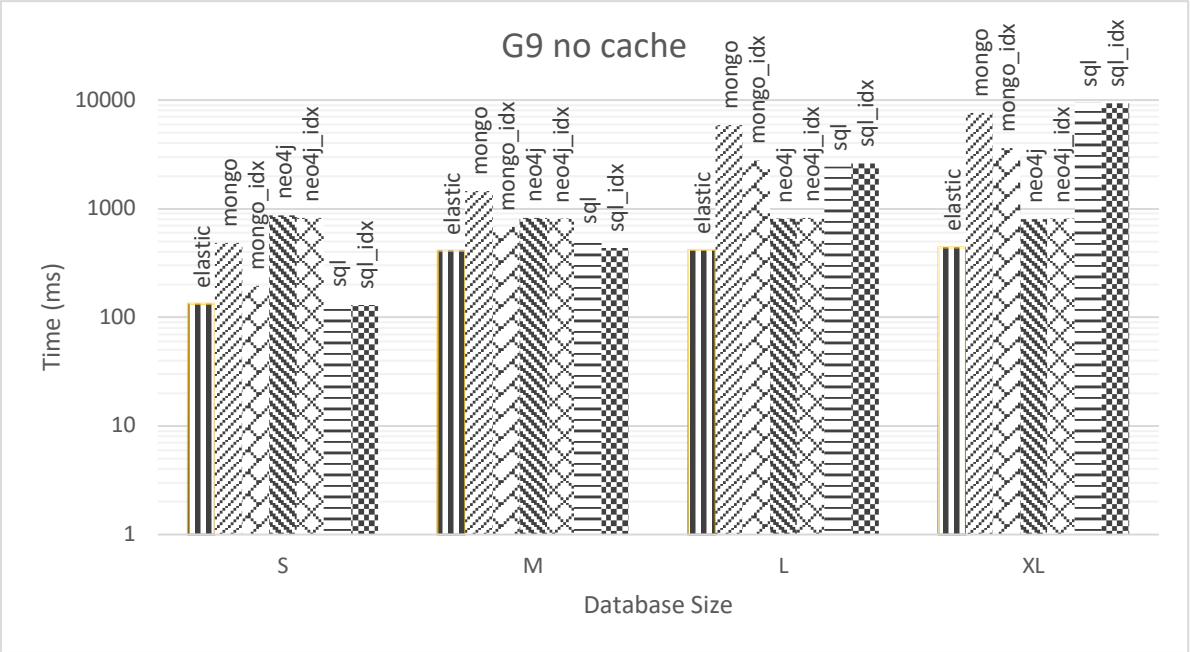
Για το ερώτημα G8, την εύρεση σημείων σε ακτίνα 1000 μέτρων που ανήκουν σε κάποια συγκεκριμένη κατηγορία παρατηρούμε τα ίδια αποτελέσματα με το ερώτημα G6. Όπως και με τα ερωτήματα G2-G6, και εδώ παρατηρούμε γρηγορότερη απάντηση σε ακτίνα 1000 μέτρων από κύκλο ακτίνας 500 μέτρων για τη MongoDB με χρήση ευρετηρίου.

	Query G9	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	31.24	33.77	34.4	32.66
	MongoDB	184.7	591.09	2444.41	3985.73
	MongoDB with Index	72.18	270.46	895.54	1154.86
	Neo4J	22	20.61	21.58	22.16
	Neo4j with Index	21.7	21.21	20.69	21.08
	MySQL	102.09	343.73	3012.74	10309.39
	MySQL with Index	101.92	344.5	3128.58	10419.94
Without Cache	ElasticSearch	134.1	411.56	415.15	439.43
	MongoDB	484.09	1463.1	5921.11	7626.98
	MongoDB with Index	194.05	676.52	2806.35	3610.76
	Neo4J	870.85	823.84	815.62	795.44
	Neo4j with Index	822.19	807.43	821.77	808.75
	MySQL	135.64	516.03	2432.86	9489.12
	MySQL with Index	129.31	435.37	2619.23	9384.31

Αποτελέσματα για ερώτημα G9 σε ms



Διάγραμμα για ερώτημα G9 με cache

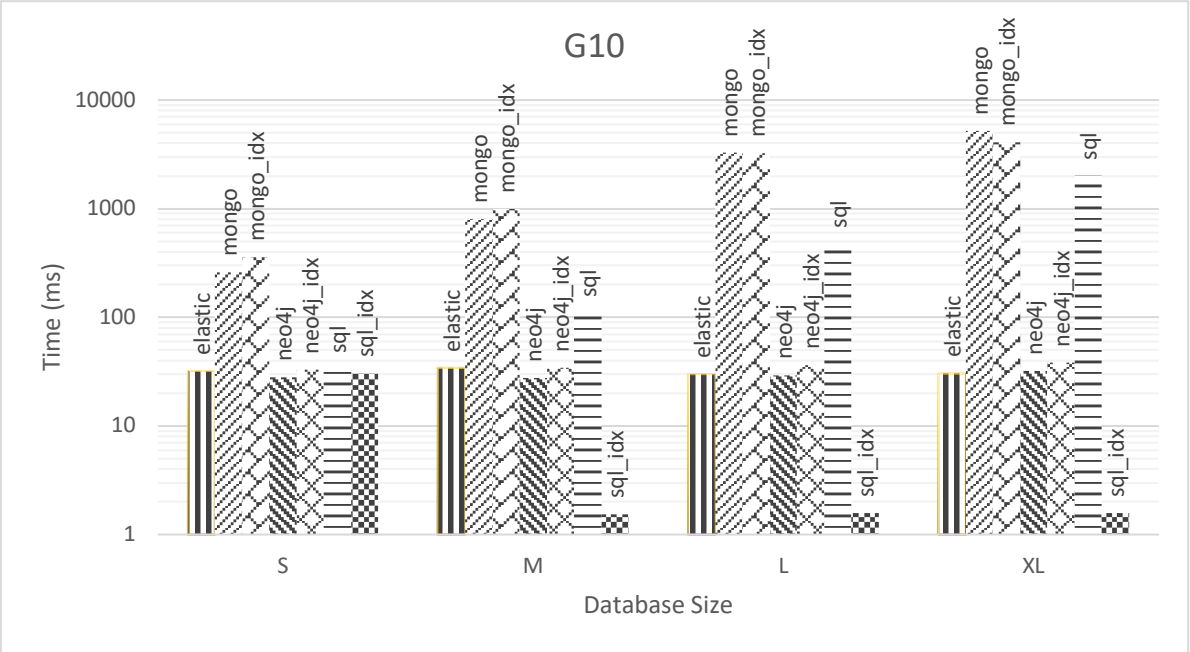


Διάγραμμα για ερώτημα G9 χωρίς cached data

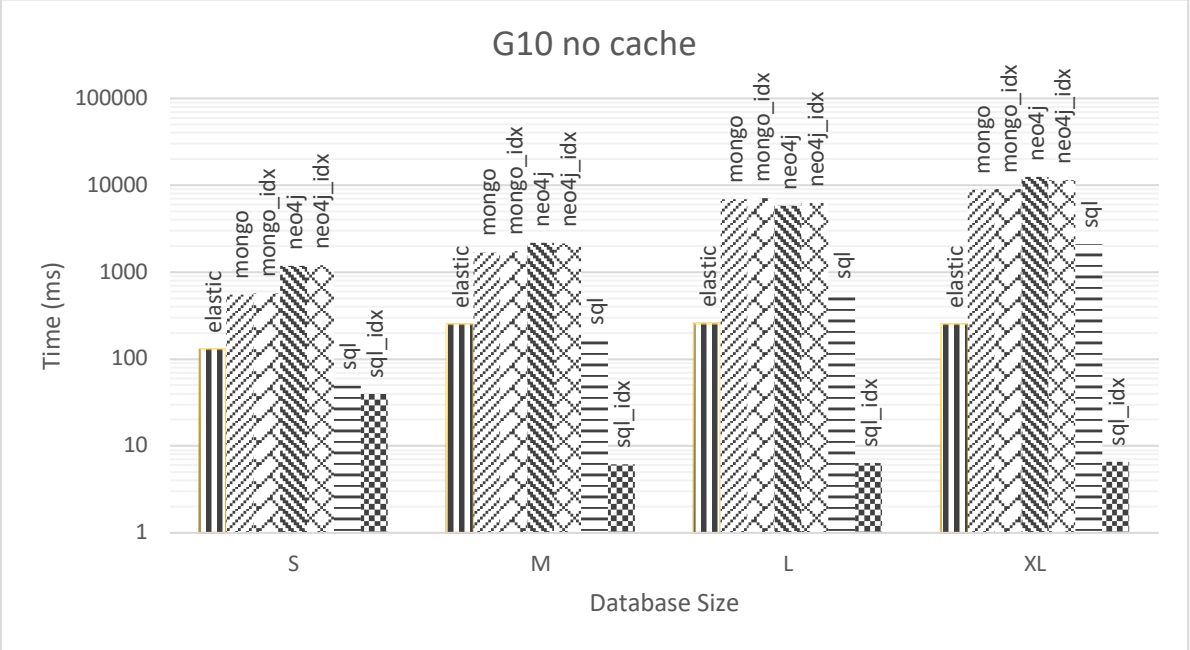
Για το ερώτημα G9, την εύρεση σημείων σε ακτίνα 2500 μέτρων που ανήκουν σε κάποια συγκεκριμένη κατηγορία παρατηρούμε τα ίδια αποτελέσματα με το ερώτημα G6.

	Query G10	**Database Size**			
	Database	Small	Medium	Large	Extra Large
Using Cache	Elasticsearch	32.01	34.24	30.04	30.47
	MongoDB	260.52	803.57	3303.25	5210.44
	MongoDB with Index	359.33	1000.26	3232.88	4067.36
	Neo4J	27.86	27.76	29.27	32.25
	Neo4j with Index	32.97	34.49	36.11	38.15
	MySQL	33.43	108.79	480.73	2005.68
	MySQL with Index	30.2	1.54	1.57	1.57
Without Cache	Elasticsearch	129.56	254.73	255.19	251.54
	MongoDB	553.86	1699.62	6948.62	8917.49
	MongoDB with Index	565.84	1750.79	7174.57	9099.55
	Neo4J	1188.67	2163.33	5839.57	12496.95
	Neo4j with Index	1208.02	2139.56	6227.23	11499.5
	MySQL	49.93	163.21	553.47	2092.04
	MySQL with Index	39.89	6.19	6.4	6.61

Αποτελέσματα για ερώτημα G10 σε ms



Διάγραμμα για ερώτημα G10 με cache



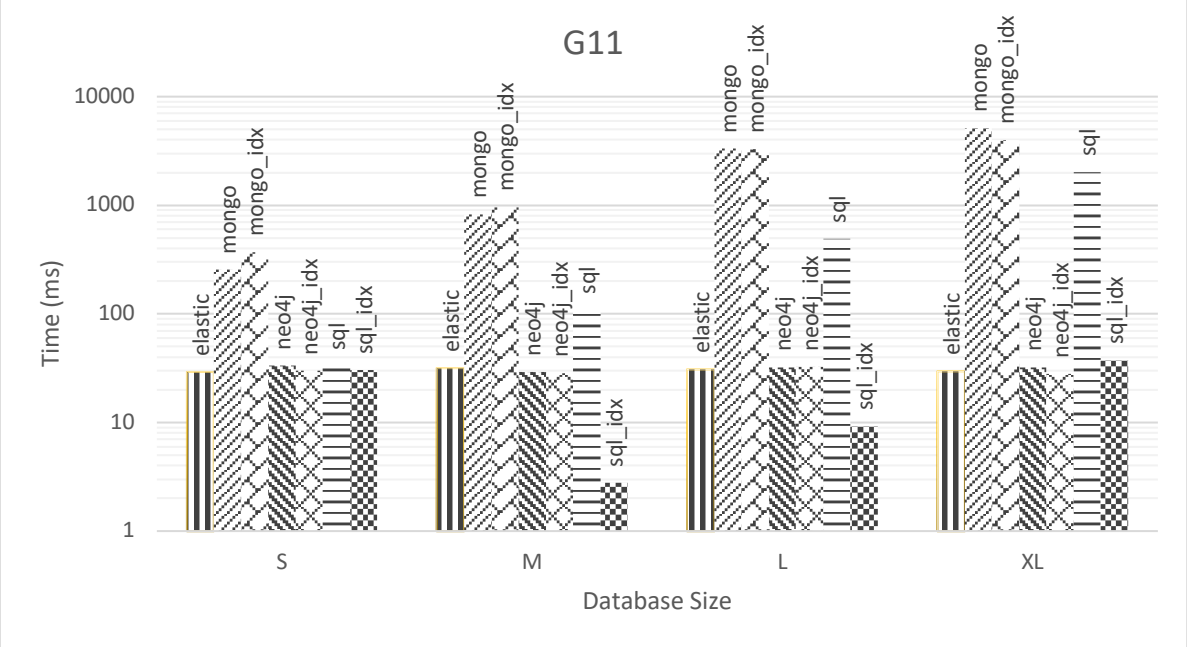
Διάγραμμα για ερώτημα G10 χωρίς cached data

Για το ερώτημα G10, την εύρεση σημείων σε παραλληλόγραμμο εμβαδού 12500 τ.μ., παρατηρούμε ότι η MySQL με χρήση ευρετηρίου έχει εξαιρετική απόδοση και έπονται η Neo4j και η Elasticsearch που έχουν παρόμοια απόκριση ανεξαρτήτου όγκου δεδομένων στα 30ms. Επιπρόσθετα, όπως ήταν αναμενόμενο, η MongoDB δε κατάφερε να χρησιμοποιήσει το δυσδιάστατο ευρετήριο κάτι που την κάνει μαζί μια τάξη μεγέθους πιο αργή ακόμα και από την MySQL χωρίς ευρετήριο. Αξιοσημείωτο είναι πως η MySQL με ευρετήριο είναι πιο αργή στη βάση με το μικρότερο όγκο δεδομένων σε σχέση με τις άλλες 3 μεγαλύτερες βάσεις.

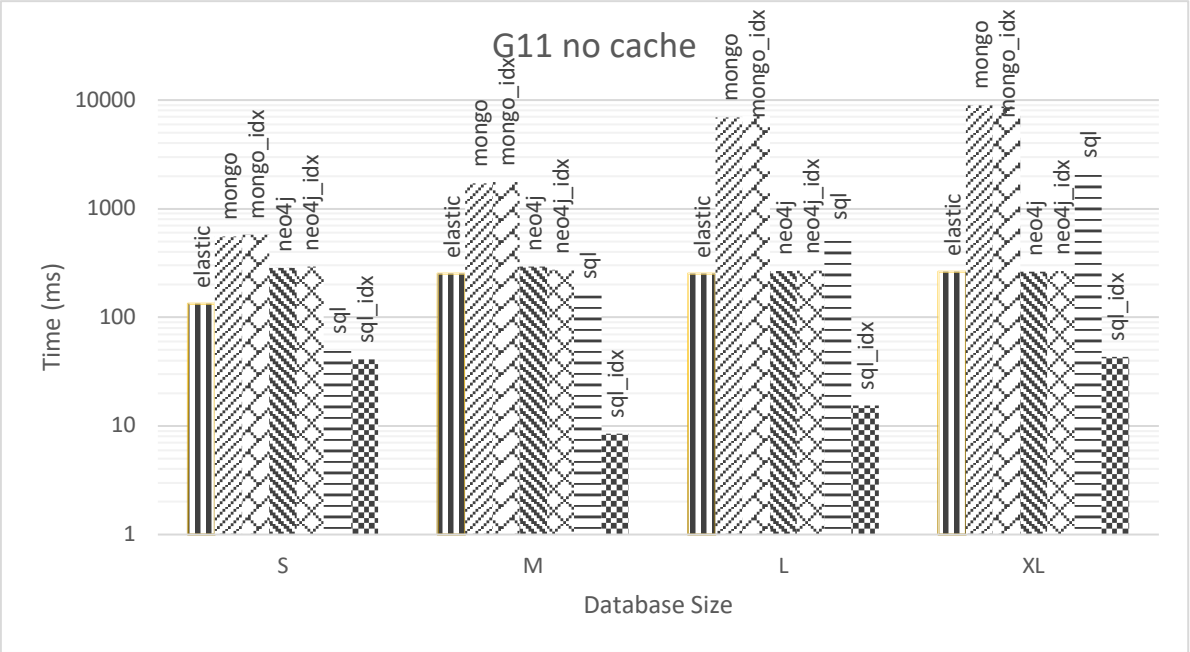
Για την εκτέλεση του ερωτήματος χωρίς cached data, πάλι το σύστημα της MySQL με ευρετήριο ξεχωρίζει για την σχεδόν άμεση απάντηση ανεξαρτήτως μεγέθους της βάσης.

	Query G11	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	29.29	31.53	30.89	29.46
	MongoDB	257.41	817.87	3349.17	5165.04
	MongoDB with Index	369.58	957.68	3288.22	3976.14
	Neo4J	33.6	29.29	32.19	32.03
	Neo4j with Index	30.05	28.5	32.46	28.08
	MySQL	33.2	109.56	491.08	2023.93
	MySQL with Index	30.47	2.77	9.22	37.38
Without Cache	ElasticSearch	133.39	253.34	251.42	264.42
	MongoDB	554.45	1721.81	6971.53	8936.32
	MongoDB with Index	574.37	1752.33	7088.23	8679.37
	Neo4J	285.19	294.32	267.42	263.22
	Neo4j with Index	291.55	275.52	271.66	266.17
	MySQL	49.96	160.01	565.31	2111.98
	MySQL with Index	41.22	8.43	15.37	43.4

Αποτελέσματα για ερώτημα G11 σε ms



Διάγραμμα για ερώτημα G11 με cache

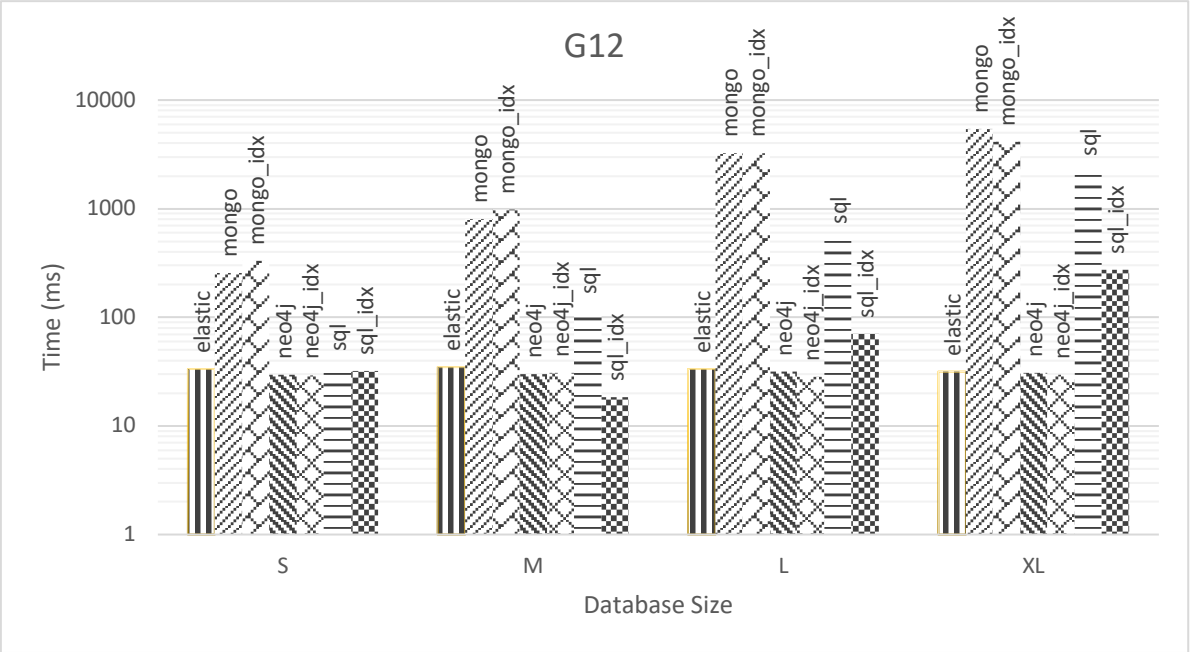


Διάγραμμα για ερώτημα G11 χωρίς cached data

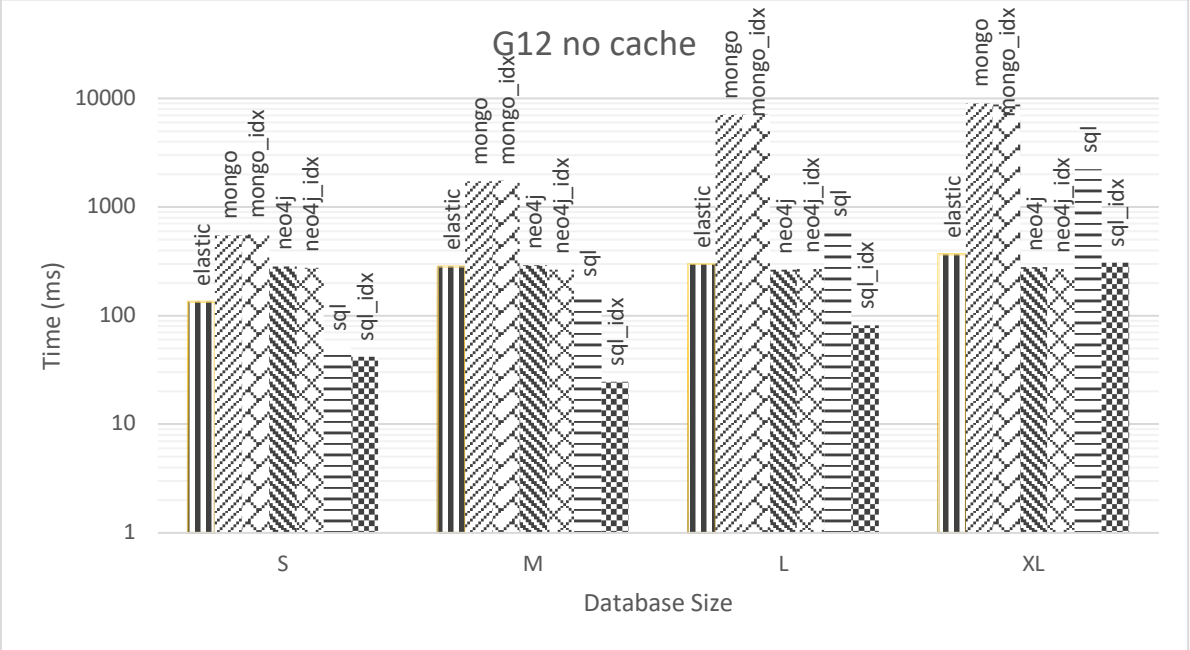
Για το ερώτημα G11, την εύρεση σημείων σε παραλληλόγραμμο εμβαδού 1 τ.χλμ., παρατηρούμε τα ίδια αποτελέσματα με το G9. Η διαφορά είναι πως η MySQL με χρήση ευρετηρίου είναι λίγο πιο αργή και στο ερώτημα στη βάση με το μεγαλύτερο όγκο δεδομένων, δεν παρουσιάζει διαφορά με τη Neo4j ή την ElasticSearch.

	Query G12	**Database Size**			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	33.23	34.93	33.56	31.49
	MongoDB	257.41	799.52	3253.1	5400.39
	MongoDB with Index	332.07	979.41	3232.69	4124.03
	Neo4J	29.55	29.95	31.45	30.95
	Neo4j with Index	29.27	30.6	28.56	29.4
	MySQL	34.63	116.7	518.34	2188.81
	MySQL with Index	32.24	18.48	70.75	273.18
Without Cache	ElasticSearch	135.11	282.07	299.94	367.51
	MongoDB	549.68	1719.76	7070.42	9022.88
	MongoDB with Index	560.32	1744.81	7065.47	8838.06
	Neo4J	284.23	292.63	266.29	279.16
	Neo4j with Index	277.39	263.65	266.91	269.55
	MySQL	51.32	161.62	583.83	2236.06
	MySQL with Index	41.78	24.58	81.52	305.61

Αποτελέσματα για ερώτημα G12 σε ms



Διάγραμμα για ερώτημα G12 με cache

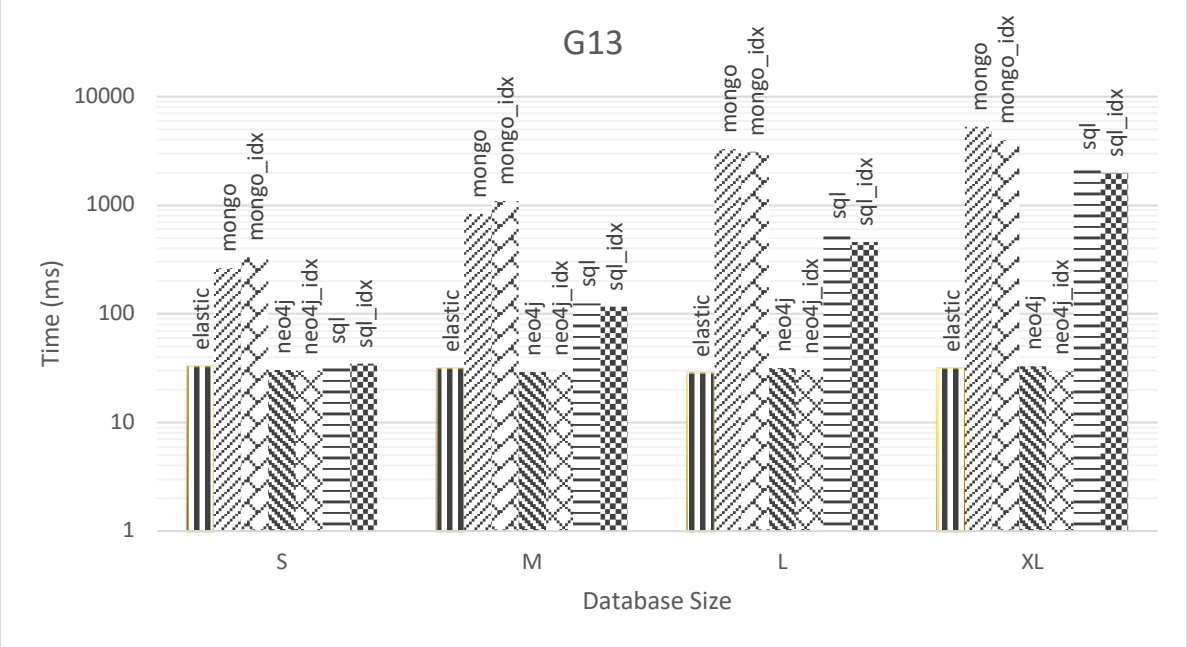


Διάγραμμα για ερώτημα G12 χωρίς cached data

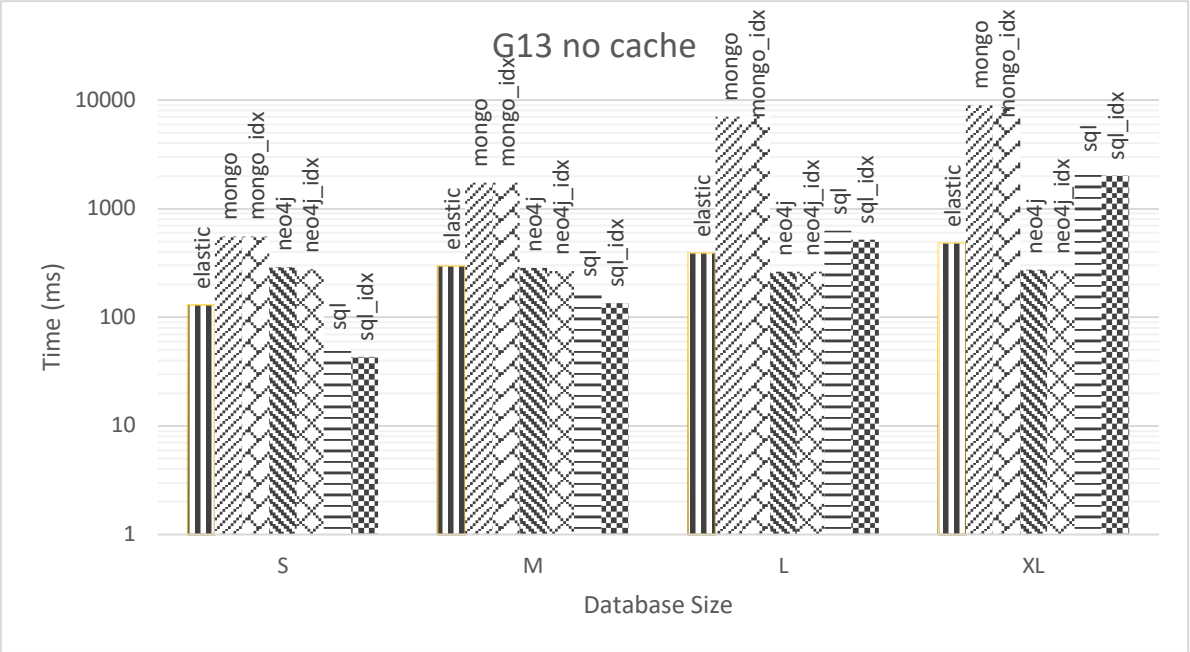
Για το ερώτημα G12, την εύρεση σημείων σε παραλληλόγραμμο εμβαδού 60 τ.χλμ., παρατηρούμε παρόμοια αποτελέσματα με το G9 με τη μεγάλη διαφορά πλέον να είναι πως η MySQL με χρήση ευρετηρίου δε μπορεί ανταποκριθεί στον ίδιο χρόνο με τα 2 προηγούμενα ερωτήματα. Τον ίδιο αντίκτυπο έχει και στην εκτέλεση του ερωτήματος σε βάση με χωρίς cached data, όπου πλέον δε μπορεί να ξεχωρίσει από την Neo4j και την Elasticsearch.

	Query G13	Database Size			
	Database	Small	Medium	Large	Extra Large
Using Cache	ElasticSearch	32.86	31.61	28.76	31.83
	MongoDB	264.07	835.16	3279.7	5265.61
	MongoDB with Index	332.96	1090.13	3110.66	3965.57
	Neo4J	30.22	29.26	31.53	32.83
	Neo4j with Index	30.07	29.06	30.46	29.48
	MySQL	37.18	124.87	535.46	2292.88
	MySQL with Index	35	117.22	458.71	1977.62
Without Cache	ElasticSearch	130.7	297.89	388.15	485.1
	MongoDB	553.19	1726.79	7016.03	8976.89
	MongoDB with Index	555.39	1737.56	7093.72	8610.51
	Neo4J	288.53	286.86	261.39	274.41
	Neo4j with Index	278.66	266.01	263.89	269.77
	MySQL	54.15	165.36	623.88	2378.06
	MySQL with Index	43.08	135.25	518.64	2017.45

Αποτελέσματα για ερώτημα G13 σε ms



Διάγραμμα για ερώτημα G13 με cache



Διάγραμμα για ερώτημα G13 χωρίς cached data

Για το ερώτημα G13, την εύρεση σημείων σε παραλληλόγραμμο εμβαδού 24000 τ.χλμ., παρατηρούμε τα ίδια αποτελέσματα με το G12 όμως πλέον είναι εμφανές ότι η MySQL με χρήση ευρετηρίου δε μπορεί να ανταγωνιστεί τους χρόνους των Neo4j και Elasticsearch παρά μόνο για μικρό όγκο δεδομένων.

5 Συμπεράσματα

5.1 Συμπεράσματα

Μετά την πειραματική εκτέλεση, τη συλλογή και την αξιολόγηση των αποτελεσμάτων για τα ερωτήματα κειμένου, εξάγονται τα εξής συμπεράσματα:

- Το σύστημα διαχείρισης βάσεων δεδομένων της Elasticsearch έχει την σταθερότερη απόδοση ανεξάρτητα από το μέγεθος των δεδομένων και το είδος του ερωτήματος. Μπορεί να χρησιμοποιηθεί για οποιοδήποτε ερώτημα πάνω σε κείμενο έχοντας μια σταθερή και προβλεπόμενη απόδοση..
- Το σύστημα διαχείρισης βάσεων δεδομένων MySQL έχει εξαιρετικό χρόνο απόκρισης όταν η αναζήτηση γίνεται πάνω σε κάποια συγκεκριμένη λέξη καθιστώντας το ιδανικό για αναζήτηση λέξεων κλειδιών μέσα σε κείμενα.
- Η Neo4j σε καμία περίπτωση δε μπορεί να ανταπεξέλθει σε αυτού του είδους την αναζήτηση.
- Σε περίπτωση που δεν είναι εφικτή η χρήση ευρετηρίων, λόγω έλλειψης πόρων για παράδειγμα, η MongoDB υπερτερεί ελαφρώς έναντι των άλλων συστημάτων.
- Αν η απαίτηση είναι η γρήγορη απόκριση ακόμα και όταν το σύστημα δεν έχει δεδομένα στην cache μνήμη, η Elasticsearch πάλι είναι η λύση με τη γρηγορότερη απόκριση.

Συνοψίζοντας όλα τα παραπάνω, είναι φανερό ότι η Elasticsearch αποτελεί την πιο ευέλικτη λύση για την αναζήτηση κειμένου, αυτό όμως δεν αναιρεί το γεγονός πως για συγκεκριμένα προβλήματα, η MySQL και η MongoDB μπορεί να υπερτερούν.

Για τα ερωτήματα σε γεωγραφικά δεδομένα εξάγονται τα εξής συμπεράσματα:

- Η χρήση του δυσδιάστατου ευρετηρίου της MongoDB ενδείκνυται στην αναζήτηση γεωγραφικών σημείων σε κύκλο. Η έλλειψη cached δεδομένων δεν επηρεάζει σημαντικά την απόκριση
- Αν η αναζήτηση πραγματοποιείται σε παραλληλεπίπεδο η Neo4j με την Elasticsearch έχουν μια σταθερή και προβλέψιμη απόδοση. Αν η αναζήτηση γίνεται σε παραλληλεπίπεδα με μικρό εμβαδόν όμως, η MySQL με χρήση ευρετηρίου είναι η ταχύτερη επιλογή.
- Η Elasticsearch, αν και δεν είναι σε κανένα ερώτημα που μελετήθηκε η γρηγορότερη επιλογή, είναι γρήγορη και ταυτόχρονα έχει την πιο προβλέψιμη απόκριση ανεξαρτήτου μεγέθους της βάσης δεδομένων, κάτι που την καθιστά ιδανική για εφαρμογές που ο όγκος των δεδομένων ενδέχεται να είναι μεγάλος.
- Αν η αναζήτηση απαιτεί προσπέλασή και άλλων σχέσεων, ιδίως αυτών που επωφελούνται από τη χρήση γράφων, η Neo4j είναι ιδανική.
- Αν η αναζήτηση γίνεται σε πολύ συγκεκριμένα σημεία, η MySQL με χρήση ευρετηρίου προσφέρει άμεσες απαντήσεις στα ερωτήματα στη βάση.

Συνοψίζοντας, πάλι προκύπτει το συμπέρασμα ότι η Elasticsearch αποτελεί την πιο ευέλικτη λύση. Παρόλα αυτά, η φύση του προβλήματος προς επίλυση και η επιλογή των ερωτημάτων προς τη βάση, επηρεάζουν σε μεγάλο βαθμό την επιλογή του κατάλληλου συστήματος διαχείρισης βάσεων δεδομένων.

Εν κατακλείδι, γίνεται φανερό πως η επιλογή του κατάλληλου συστήματος διαχείρισης βάσεων δεδομένων είναι μια επιλογή που εξαρτάται από τη φύση του προβλήματος και χρειάζεται προσεκτική επιλογή που να καλύπτει τις απαιτήσεις της σχεδιαζόμενης λύσης.

5.2 Μελλοντικές επεκτάσεις

Το πρόβλημα της εύρεσης του σωστού συστήματος διαχείρισης βάσεων δεδομένων είναι ένα πρόβλημα που εξαρτάται από πολλές μεταβλητές. Δεν υπάρχει κάποιο σύστημα που να προσφέρει μια καθολική λύση για όλα τα προβλήματα, όλες τις μορφές δεδομένων καθώς και όλες τις περιπτώσεις χρήσης του συστήματος.

Επεκτάσεις ως προς το είδος των δεδομένων θα μπορούσαν να γίνουν πάνω στην αποθήκευση και προσπέλαση μεγάλων αρχείων, όπως αρχεία pdf και εικόνων. Ένα άλλο ενδιαφέρον πεδίο είναι η μελέτη δεδομένων με μεγάλη αλληλεπίδραση μεταξύ τους μέσω σχέσεων εξάρτησης και πως αυτά τα δεδομένα θα πρέπει να τα χειριστεί κάθε σύστημα ώστε να επιτύχει το καλύτερο δυνατό αποτέλεσμα

Πιθανή επέκταση θα αποτελούσε η πειραματική σύγκριση της παρούσας διπλωματικής αλλά με την χρήση πολλαπλών εξυπηρετητών (clients) να ζητάνε δεδομένα από τα συστήματα διαχείρισης βάσεων δεδομένων ταυτόχρονα. Θα είχε ενδιαφέρον η μελέτη αντίδρασης αυτών των συστημάτων σε διάφορες μορφές φορτίου, από συνεχή σταθερή αναζήτηση σε απότομη αύξηση του φορτίου σε τυχαίες στιγμές. Στα πλαίσια της παραπάνω μελέτης θα μπορούσαν να δοκιμαστούν και οι τεχνικές αντιγραφής (replication) και κατάτμησης της πληροφορίας (sharding) που προσφέρει το κάθε σύστημα.

Τέλος, μια άλλη επέκταση είναι η μελέτη της συμπεριφοράς των συστημάτων διαχείρισης βάσεων δεδομένων στις άλλες τρεις μεθόδους που περιγράφονται στο CRUD (create, read, update, delete) δηλαδή στη δημιουργία δεδομένων στη βάση, στην αλλαγή των δεδομένων και στη διαγραφή τους.

Βιβλιογραφία

[1] J. Gantz and D. Reinsel, "THE DIGITAL UNIVERSE IN 2020:BigData,Bigger Digital Shadows,and Biggest Growth in the Far East," *EMC,* 2012.

[2] Z.-W. Xu, "Cloud-Sea Computing Systems: Towards Thousand-Fold Improvement in Performance per Watt for the Coming Zettabyte Era," *Journal of Computer Science and Technology,* vol. 29, no. 2, pp. 177-181, 2014.

[3] D. Bedell, «Payments Volumes Worldwide,» 07 June 2013. [Ηλεκτρονικό]. Available: https://www.gfmag.com/global-data/economic-data/26gzj8-payments-volumes-worldwide.

[4] European Central Bank, "Payments statistics: methodological notes," European Central Bank.

[5] Plexxi Inc, "The Changing Landscape of Data Storage White Paper," Plexxi Inc, 2016.

[6] European Union , "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Da," 4 5 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EL.

[7] J. R. Lourenço, B. Cabral and P. Carreiro, "Choosing the right NoSQL database for the job: a quality attribute evaluation,".

[8] M. Indrawan-Santiago, "Database Research: Are We at a Crossroad? Reflection on NoSQL," in *2012 15th International Conference on Network-Based Information Systems*, 2012.

[9] Princeton, "Wordnetweb Princeton," [Online]. Available: http://wordnetweb.princeton.edu/perl/webwn?s=database%20management%20system.

[10] TechTerms, "TechTerms," [Online]. Available: https://techterms.com/definition/dbms.

[11] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM,* vol. 13, no. 6, pp. 377-387, 1970.

[12] D. Chamberlin and R. Boyce, "SEQUEL: a structured English query language," in *Conference: Proceedings of 1974 ACM-SIGMOD Workshop on Data Description, Access and Control*, Ann Arbor, Michigan, 1974.

[13] Oracle Coorporation, "dev.mysql.com," Oracle Coorporation, [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

[14] Oracle Coorporation, "dev.mysql.com," Oracle Corporation, [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/history.html.

[15] Oracle Corporation, "mysql.com," Oracle Corporation, [Online]. Available: https://www.mysql.com/customers/.

[16] Oracle Corporation, "dev.mysql.com," Oracle Corporation, [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/features.html.

[17] C. Strozzi, "strozzi.it," [Online]. Available: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page.

[18] "blog.sym-link.com," 2009. [Online]. Available: http://blog.sym-link.com/2009/05/12/nosql_2009.html.

[19] N. Levvit, "Will NoSQL Databases Live Up to Their Promise?," *Technology News*

IEEE Computer Society, 2010.

[20] MongoDB Inc, "www.mongodb.com," MongoDB Inc, [Online]. Available: https://www.mongodb.com/document-databases.

[21] "db-engines.com" [Online]. Available: https://db-engines.com/en/ranking/document+store.

[22] Elastic Co, "www.elastic.co" Elastic Co, [Online]. Available: https://www.elastic.co/products/elasticsearch.

[23] "db-engines.com" [Online]. Available: https://db-engines.com/en/ranking/search+engine.

[24] Apache Foundation, "lucene.apache.org" Apache Foundation, [Online]. Available: https://lucene.apache.org/core/.

[25] Elastic Co, "www.elastic.co" Elastic Co, [Online]. Available: https://www.elastic.co/use-cases/.

[26] MongoDB Inc, "www.mongodb.com" MongoDB Inc, [Online]. Available: https://www.mongodb.com/mongodb-architecture.

[27] MongoDB Inc, "mongodb.com" MongoDB Inc, [Online]. Available: https://www.mongodb.com/who-uses-mongodb.

[28] "neo4j.com" [Online]. Available: https://neo4j.com/developer/graph-database/.

[29] "db-engines.com" [Online]. Available: https://db-engines.com/en/ranking/graph+dbms.

[30] Neo4j Inc, "neo4j.com" Neo4j Inc, [Online]. Available: https://neo4j.com/developer/graph-database/#_what_is_neo4j.

[31] Neo4j Inc, «neo4j.com» Neo4j Inc, [Ηλεκτρονικό]. Available: https://neo4j.com/product/.

[32] Neo4j Inc, "neo4j.com" Neo4j Inc, [Online]. Available: https://neo4j.com/customers/.

[33] [Online]. Available: ttps://raw.githubusercontent.com/python/cpython/master/Misc/HISTORY.

[34] Python Software Foundation, "www.python.orγ" Python Software Foundation, [Online]. Available: https://www.python.org/doc/essays/blurb/.

[35] Python Software Foundation, "wiki.python.org" Python Software Foundation, [Online]. Available: https://wiki.python.org/moin/OrganizationsUsingPython.

[36] R. Hecht and S. Jablonski, "NoSQL Evaluation, A Use Case Oriented Survey," in *Internation Conference on Cloud and Service computing*, 2011.

[37] M. Indrawan-Santiago, "Database Research: Are We At A Crossroad? Reflection on NoSQL," in *15th International Conference on Network-Based Information Systems*, 2012.

[38] S. K. Gajerdan, "A Survey on NoSQL Databases".

[39] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," p. 2013.

[40] A. Boicea, F. Radulescu and L. I. Agapin, "MongoDB vs Oracle - database comparison," in *Third International Conference on Emerging Intelligent Data and Web Technologies*, 2012.

[41] Z. Parker, S. Poe and S. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB," *ACM,* 2013.

[42] S. Agarwal and K. Rajan, "Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries," in *Free and Open Source Software for Geospatial*,

Boston, 2017.

[43] V. Abramova, J. Bernardino and P. Furtado, "EXPERIMENTAL EVALUATION OF NO SQL DATABASES," *International Journal of Database Management Systems,* vol. 6, no. 3, 2014.

[44] D. Upeksha, "Comparison Between Performanceof Various Database Systemsfor Implementing a Language Corpus".

[45] ISO, "iso.org" [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:6709:ed-2:v1:en.