Πανεπιστήμιο Πειραιά

Τμήμα Ψηφιακών Συστημάτων

Ψηφιακά Συστήματα και Υπηρεσίες

Μεγάλα Δεδομένα και Αναλυτική

Κωνσταντίνος Κοκολόγος : ΜΕ1713

Διπλωματική Εργασία

Επεξεργασία Χωρικών Επερωτήσεων στην MongoDB

Επιβλέπων Καθηγητής : Χρήστος Δουλκερίδης

(ΠΕΙΡΑΙΑΣ 2019)

ΠΕΡΙΛΗΨΗ

Στην συγκεκριμένη εργασία υλοποιήθηκαν κάποιες λειτουργίες της MongoDB που σχετίζονταν με γεωγραφικά δεδομένα. Για να είναι εφικτές αυτές, έγινε αποθήκευση των δεδομένων στην MongoDB με τη χρήση της γλώσσας προγραμματισμού python. Επιπλέον, υλοποιήθηκαν κάποιες συναρτήσεις, μέσω των οποίων ο χρήστης μπορεί να πραγματοποιεί κάποιες λειτουργίες της MongoDB.

Επίπλέον με τη χρήση Virtual Machines πραγματοποιήσαμε το replication στην mongodb, με τη χρήση ενός συστήματος master-slave.

Στη συγκεκριμένη διπλωματική εργασία, αναφερθήκαμε και αναλύσαμε όλους τους όρους που συναντάμε στις παραπάνω υλοποιήσεις αλλά και σε άλλες λειτουργίες.Επίσης, παρουσιάζεται ένας τρόπος για να κάνουμε backup στην MongoDB.

ABSTRACT

In this project were created some operations in MongoDB about spatial data. These data were stored in MongoDB with programming language python. Furthermore the user with these functions will be able to do some operations in MongoDB.

Furthermore with Virtual Machines was created replication in mongodb with the use of master-slave system.

In this project was analyzed and explained all the operations which will see. Also was presented a backup method in MongoDB.

Περιεχόμενα

Περιεχόμενα Εικόνων 5

Περιεχόμενα Πινάκων 6

Ευχαριστίες 7

Εισαγωγή 8

Κεφάλαιο 1: Χωρικά Δεδομένα 10

1.1 Χωρικά Δεδομένα και Συστήματα Γεωγραφικών Πληροφοριών 10

1.2 Χωρικά Δεδομένα και MongoDB 12

Κεφάλαιο 2: Τεχνολογικό Υπόβαθρο 15

2.1 Python 15

2.2 NoSQL Βάσεις Δεδομένων 15

2.3 MongoDB 23

2.4 Αποθήκευση Δεδομένων στη MongoDB 27

Κεφάλαιο 3: Υλοποίηση – Επεξεργασία Δεδομένων 29

3.1 Αποθήκευση Δεδομένων 29

3.2 Διαχείριση Δεδομένων 30

3.3 Υπολογισμός Έυρους για Εστιατόρια και Γειτονίες 31

Κεφάλαιο 4: Replication, Sharding 34

4.1 Replication 34

4.2 Εφαρμογή Replication 35

4.3 Sharding 36

4.4 Πλεονεκτήματα Sharding 37

Κεφάλαιο 5: Συμπεράσματα 40

Βιβλιογραφία 41

Παράρτημα 43

1. Robo 3T 43

2. Εγκατάσταση MongoDB στο Linux 43

3. MongoDB Backup 44

Περιεχόμενα Εικόνων

Εικόνα 1: Key-values database 20
Εικόνα 2: Παράδειγμα χρήσης column stores [17] 21
Εικόνα 3: Graph Database [16] 22
Εικόνα 4: Απεικόνιση graph database [17] 23
Εικόνα 5: Ορισμός της γειτονίας Hell's Kitchen μέσω MongoDB [5] 33
Εικόνα 6: Εύρεση εστιατορίων στη γειτονιά Hell's Kitchen [5] 33
Εικόνα 7: Αλληλεπίδραση εξαρτημάτων σε ένα shared cluster [10] 37

Περιεχόμενα Πινάκων

Πίνακας 1: Σύγκριση MySQL και MongoDB [11]..... 26

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν με κάθε τρόπο στην υλοποίηση αυτής της διπλωματικής εργασίας. Κυρίως θα ήθελα να ευχαριστήσω την οικογένειά μου που με στήριξε καθ’ όλη την διάρκεια των σπουδών μου ώστε να φέρω εις πέρας το συγκεκριμένο μεταπτυχιακό πρόγραμμα.

Επίσης θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ.Χρήστο Δουλκερίδη που με στήριξε και με καθοδήγησε ώστε να μπορέσω να υλοποιήσω αυτήν την διπλωματική εργασία.

Εισαγωγή

Στόχος της παρούσας διπλωματικής εργασίας είναι η διαχείριση, αποθήκευση και επερώτηση μεγάλου όγκου χωρικά δεδομένα. Σκοπός μας είναι η υποστηρίξη χωρικών ερωτημάτων και η εξαγωγή πληροφοριών από αυτά. Για να καταστεί αυτό εφικτό, τα δεδομένα αποθηκεύτηκαν σε NoSQL σύστημα αποθήκευσης δεδομένων, έτσι ώστε να ικανοποιηθούν οι ανάγκες για scalability. Πιο συγκεκριμένα, επιλέχθηκε η κατηγορία document-oriented storage της NoSQL. Για την υλοποίηση της εργασίας, επιλέχθηκε η MongoDB, που ανήκει σε αυτή την κατηγορία και επιπλέον παρέχει μια επέκταση για χωρικά δεδομένα (spatial extension), που μας είναι απαραίτητη στη συγκεκριμένη διπλωματική εργασία.

Για να είναι όλα τα παραπάνω διαθέσιμα, αναπτύχθηκαν κάποιες υλοποιήσεις που θα δούμε αναλυτικά και στη συνέχεια αυτής της αναφοράς. Έτσι, είναι δυνατή η εύρεση σημείων ενδιαφέροντος με βάση ένα συγκεκριμένο εύρος σε σχέση με ένα αρχικό σημείο, το σημείο της επερώτησης. Για παράδειγμα, είναι δυνατόν να βρεθούν εστιατόρια σε απόσταση κάποιων χιλιομέτρων σε σχέση με ένα αρχικό σημείο ενδιαφέροντος. Επιπλέον, με βάση ένα σύνολο συντεταγμένων, που ορίζουν, για παράδειγμα μια περιοχή ή μία γειτονία, είναι δυνατή η εύρεση σημείων που βρίσκονται κοντά ή εντός αυτών των περιοχών. Για να γίνουν όλα αυτά εφικτά χρησιμοποιήθηκε η γλώσσα προγραμματισμού python.

Επιπλέον, γίνεται replication, με τη χρήση ενός συστήματος master-slave, όπου οι εγγραφές του πρώτου αντιγράφονται στο δεύτερο. Τέλος, υποστηρίζεται backup στην MongoDB. Αυτή η διπλωματική εργασία, έχει σαν δευτερεύοντα στόχο να βοηθήσει κάθε χρήστη που θα το διαβάσει να μπορέσει να υλοποιήσει κάποιες βασικές λειτουργίες της MongoDB με τη χρήση της γλώσσας προγραμματισμού python.

Όσον αφορά τη δομή της εργασίας, αρχικά στο πρώτο κεφάλαιο, κάνουμε μια γενική αναφορά στα χωρικά δεδομένα. Αναλύουμε σε ποιες κατηγορίες αυτά μπορούν να ταξινομηθούν και πως επεξεργάζονται τελικά από την MongoDB. Στο κεφάλαιο 2, αναπτύσουμε κάνουμε μία αναφορά στο τεχνολογικό υπόβαθρο της εργασίας. Πιο συγκεκριμένα, αναφερόμαστε στη Python, τις NoSQL βάσεις δεδομένων, στη MongoDB καθώς και αποθηκεύονται τα χωρικά δεδομένα σε αυτή. Έπειτα, στο κεφάλαιο 3,

αναλύουμε την υλοποίηση της διπλωματικής εργασίας. Έτσι, αναφερόμαστε στον τρόπο αποθήκευσης και διαχείρισης των δεδομένων καθώς και στα ερωτήματα που χρησιμοποιούνται. Τέλος, στο κεφάλαιο 4, γίνεται αναφορά στο replication και στο sharding, που χρησιμοποιήθηκε στα πλαίσια της εργασίας. Στο κεφάλαιο 5, παρουσιάζονται κάποια πειραματικά αποτελέσμα, από την υλοποίηση της εργασίας. Ενώ, στο κεφάλαιο 6 κάνουμε στα συμπεράσματα της εργασίας. Τέλος, παραθέτουμε την βιβλιογραφία που χρησιμοποιήθηκε για την διεκπεραίωση αυτής της εργασίας, καθώς και ένα παράρτημα με κάποια επιπλέον στοιχεία για τη MongoDB και το backup.

Κεφάλαιο 1: Χωρικά Δεδομένα

1.1 Χωρικά Δεδομένα και Συστήματα Γεωγραφικών Πληροφοριών

Στις μέρες μας, μια πληθώρα ανθρώπινων δραστηριότητων συνδέονται άμεσα ή έμμεσα με στην τοποθεσία. Πιο συγκεκριμένα, έχει γίνει η υπόθεση ότι σχεδόν το 80% όλων των καθημερινών δραστηριοτήτων των ανθρώπων συνδέονται με την τοποθεσία. Τα δεδομένα που σχετίζονται με την τοποθεσία ονομάζονται γεωγραφικά δεδομένα και χρησιμοποιούνται τόσο από ειδικευμένους όσο και από απλούς χρήστες. Πιο συγκεκριμένα, οι εξιδικευμένοι χρήστες μπορεί να είναι ερευνητές ή να πραγματοποιούν στατιστικές αναλύσεις. Από την άλλη, απλοί χρήστες μπορούν να χρησιμοποιούν γεωγραφικά δεδομένα για την πλοηγησή τους σε κάποια τοποθεσία. Έτσι, μπορούμε να συμπεράνουμε ότι τα γεωγραφικά δεδομένα κατέχουν μία εξέχουσα θέση μεταξύ των διάφορων μορφών δεδομένων. [20]

Με τον όρο χωρικά δεδομένα (spatial data) αναφερόμαστε σε δεδομένα ή πληροφορίες που έχουν κάποιο χωρικό στοιχείο, δηλαδή προσδιορίζουν τη γεωγραφική θέση χαρακτηριστικών και ορίων στη Γη. Τα χωρικά δεδομένα είναι επίσης γνωστά και ως γεωγραφικά δεδομένα (geospatial data) ή γεωγραφικές πληροφορίες (geographic information). Τα γεωγραφικά δεδομένα μπορεί να αναφέρονται σε φυσικά ή τεχνικά σημεία όπως για παράδειγμα ένας ωκεανός ή ένα μουσείο. Για την αποθήκευση χωρικών δεδομένων απαιτείται η αποθηκεύση συντεταγμένων και τοπολογίας έτσι ώστε τα δεδομένα να είναι δυνατόν να χαρτογραφηθούν.

Τα χωρικά δεδομένα χρησιμοποιούνται σε κάθε δραστηριότητα χαρτογράφησης και μπορούν να ταξινομηθούν σε δύο τύπους: ακατέργαστα και παράγωγα. Τα ακατέργαστα δεδομένα, χρησιμοποιούνται σε γεωμορφολογικές χαρτογράφησεις και περιλαμβάνουν πληροφορίες σχετικά με την κατανομή του ύψους, όπως οι ισοϋψής καμπύλες σε έναν τοπογραφικό χάρτη. Τα παράγωγα δεδομένα, από την άλλη, περιλαμβάνουν δεδομένω που προκύπτουν από άλλα προυπάρχονται όπως για παράδειγμα η γωνία κλίσης, η καμπυλότητα και η όψη. Οι θεματικοί χάρτες που απεικονίζουν τη χωρική κατανομή του εδάφους, αποτελούν γεωμορφολογικά προϊόντα χαρτών όπου χρησιμοποιούνται τόσο ακατέργαστα όσο και παράγωγα δεδομένα. Παράγωγα δεδομένω χρησιμοποιούνται επίσης και στην περίπτωση οπτικοποίησης των τοπογραφικών χαρτών. Τέλος οι εναέριες ή

δορυφορικές εικόνες αποτελούν επίσης παράγωγα δεδομένα που είναι χρήσιμα στην εφαρμοσμένη γεωμορφολογική χαρτογράφηση.

Μια διαφορετική κατηγοριοποίηση των χωρικών δεδομένων είναι αυτή των αναλογικών σε σχέση με τα ψηφιακά. Τα παραδοσιακά χωρικά δεδομένα απεικονίζονται σε αναλογική μορφή όπως για παράδειγμα, τυπωμένοι χάρτες και χειρόγραφες εικόνες σε υποσημειώσεις. Παρόλο που τα αναλογικά δεδομένα συνέβαλαν στην ανάπτυξη της γεωμορφολογίας, η ποιοτική και υποκειμενική φύση αυτών των δεδομένων, καθιστά δύσκολη την ανάλυση και την επεξεργασια τους με τη χρήση ηλεκτρονικών υπολογιστών. Ήδη, από τη δεκαετία του 1980, οι ψηφιακοί χάρτες έχουν αντικαταστήσει σε μεγάλο βαθμό τις παραδοσιακές αναλογικές πηγές δεδομένων και οι τοπογραφικοί χάρτες έχουν αντικατασταθεί από ψηφιακά μοντέλα εδάφους (Digital Elevation Model, DEM). Τα ψηφιακά δεδομένα καθιστούν εύκολη την ανάλυσή τους τόσο από τη χρήση προσωπικών υπολογιστών όσο και από τη χρήση γεωγραφικών συστημάτων πληροφοριών. [19, 20]

Τα γεωγραφικά δεδομένα αποθηκεύονται σε βάσεις δεδομένων, μέσω των οποίων είναι δυνατόν να προσπελαστούν. Έτσι σε μια βάση χωρικών δεδομένων είναι δυνατόν να αποθηκευτεί ένα σημείο που μπορεί να αντιστοιχεί σε ένα σπίτι ή ένα μνημείο, μία γραμμή που μπορεί να αποτελεί ένα οδικό τμήμα ή ένα οδικό δίκτυο αλλά και ένα πολύγωνο που μπορεί να αντιστοιχεί σε κάποιο νομός ή μία περιοχή. Σε μία βάση δεδομένων όπου αποθηκεύονται γεωγραφικά δεδομένα είναι δυνατόν να χρησιμοποιηθεί κάποια γλώσσα μέσω της οποίας μπορεί να τίθονται ερωτήματα υψηλού επιπέδου, όπως για παράδειγμα, η SQL. Στην SQL, πέρα από τους απλούς τύπους δεδομένων που είναι δυνατόν να επεξεργάζεται, έχουν προστεθεί και τύποι και λειτουργίες χωρικών δεδομένων.

Μία εφαρμογή που θα μπορούσε να χρησιμοποιήσει χωρικά δεδομένα είναι για παράδειγμα ένα οδικό δίκτυο, με βάση το οποίο μπορεί να προγραμματιστεί η ροή διάφορων τύπων οχημάτων, όπως για παράδειγμα των αυτοκινήτων και των τρένων. Σε αυτή την περίπτωση, τα ερώτηματα προς τη βάση θα σχετίζονται με την εύρεση της συντομότερης ή της μικρότερης χιλιομετρικά σχετικά με μία απόσταση. Μία διαφορετική εφαρμογή θα μπορούσε να είναι βασισμένο σε φυσικούς πόρους και να διαχειρίζεται γεωργικές εκτάσεις, δάση, χώρους αναψυχής, εκτάσεις άγριας πανίδας, κτ.λ.. Αυτή η εφαρμογή θα μπορούσε να χρησιμοποιηθεί για παράδειγμα, για την ανάλυση περιβαλλοντικών επιπτώσεων, τη μοντελοποίηση υπογείων υδάτων, κ.α. Τα ερωτήματα σε αυτή την περίπτωση θα μπορούσαν να σχετίζονταν με την πρόγνωση του καιρού σε μία συγκεκριμένη έκταση. [21]

Τα χωρικά δεδομένα συχνά αποθηκεύονται σε Γεωγραφικών Πληροφοριακών Συστημάτων (Geographic Information Systems, GIS). Μέσω αυτών των συστημάτων, είναι δυνατόν να προσπελαύνονται, να επεξεργάζονται, να διαχειρίζονται, να αναλύονται και τελικά να παρουσιάζονται στον τελικό χρήστη. Έτσι είναι δυνατή η οπτικοποίηση των γεωγραφικών δεδομένων, σε ψηφιακό περιβάλλον αλλά και την ανάδειξη των σχέσεων μεταξύ των διάφορων δεδομένων. Ένα σύστημα γεωγραφικών πληροφοριών αποτελείται από το υλικό, το λογισμικό, τα δεδομένα και τα άτομα που είναι υπεύθυνα για την καταγραφή, τον χειρισμό, την ανάλυση και εμφάνιση όλων γεωγραφικών δεδομένων ή χωρικών δεδομένων. Σκοπός των γεωγραφικών πληροφοριακών συστημάτων είναι η λήψη αποφάσεων, και η διαχείρηση των γεωγραφικών δεδομένων. [20, 21]

Για να είναι δυνατόν τα γεωγραφικά συστήματα να εκτελέσουν όλες τις παραπάνω εργασίες, αποτελούνται από ένα σύνολο υποσυστημάτων. Πιο συγκεκριμένα, διαθέτουν σύστημα επεξεργασίας δεδομένων, το οποίο είναι υπεύθυνο για την λήψη, για παράδειγμα από χάρτες και εικόνες, γεωγραφικών δεδομένων και την αποθήκευσή τους. Στη συνέχεια, αποτελούνται από συστήματα ανάλυσης δεδομένων, ώστε να είναι δυνατή η επεξέργασία των δεδομένων, η απάντηση ερωτημάτων καθώς και η στατιστική τους ανάλυση. Τέλος, διαθέτουν σύστημα χρήσης και διαχείρησης των γεωγραφικών δεδομένων, μέσω των οποίων είναι δυαντόν να προκύπτουν οι διάφορες δομές δεδομένων αλλά και η αλληλεπίδραση των γεωγραφικών δεδομένων. [21]

1.2 Χωρικά Δεδομένα και MongoDB

Η MongoDB έχει τη δυνατότητα να υποστηρίζει ερωτήματα χωρικά δεδομένα. Για να είναι αυτό εφικτό, αποθηκεύει τα χωρικά δεδομένα είτε ως αντικείμενα GeoJSON είτε με την παραδοσιακή μορφή σαν ζεύγη συντεταγμένων. Αυτοί οι δύο τύπο δεδομένων μπορούν να εξυπηρετούν διαφορετικές ανάγκες. Έτσι, με την αποθηκεύση των χωρικών δεδομένων ως αντικείμενα GeoJSON, είναι δυνατός ο υπολογισμός της γεωμετρίας που σχετίζαται με την σφαιρική μορφή που έχει η Γη. Αντίθετα, με τη χρήση γεωγραφικών συντεταγμένων είναι δυνατός ο υπολογισμός αποστάσεων σε ένα ευκλείδειο επίπεδο. Για τον υπολογισμό γεωμετρίας μιας σφαίρα που μοιάζει με Γη, τα χωρικά δεδομένα μετατρέπονται σε αντικείμενα GeoJSON.

Στην περίπτωση των αντικειμένων GeoJSON, χρησιμοποιείται ένα έγγραφο, στο οποίο ορίζουμε τα ακόλουθα πεδία:

- ένα πεδίο που ονομάζεται type και καθορίζει τον τύπο του GeoJSON αντικειμένου
- ένα πεδίο που ονομάζεται coordinates και καθορίζει τις συντεταγμένες ενός αντικειμένου. Κατά τον ορισμό των συντεταγμένων γεωγραφικού πλάτους και μήκους, δίνεται πρώτα το γεωγραφικό μήκος και ύστερα το γεωγραφικό πλάτος. Έγκυρες τιμές γεωγραφικού μήκους είναι μεταξύ -180 και 180, συμπεριλαμβανόμενων και αυτών. Ενώ έγκυρες τιμές γεωγραφικού πλάτους είναι μεταξύ -90 και 90, συμπεριλ.αμβανόμενων και αυτών.

Η δομή, λοιπόν ενός GeoJSON αντικειμένου είναι η ακόλουθη:

```
<field>: { type: <GeoJSON type> , coordinates: <coordinates> }
```

Από την άλλη στην περίπτωση όπου για την αποθήκευση των δεδομένων χρησιμοποιούνται ζεύγη συντεταγμένων, είναι δυνατόν να χρησιμοποιηθεί είτε ένας πίνακα είτε ένα έγγραφο. Σε αυτή την περίπτωση τα δεδομένα αποθηκεύονται με την ακόλουθη μορφή αν χρησιμοποιείται πίνακας:

```
<field>: [ <longitude>, <latitude> ]
```

Διαφορετικά, στην περίπτωση των εγγράφων έχουν την ακόλουθη μορφή:

```
<field>: { <field1>: <longitude>, <field2>: <latitude> }
```

Η MongoDB παρέχει δύο διαφορετικούς τύπους χωρικών ευρετηρίων για την υποστήριξη των χωρικών ερωτημάτων. Οι δύο διαφορετικοί τύποι χωρικών ευρετηρίων είναι οι 2dsphere και 2d. [5]

Αρχικά, μέσω του ευρετηρίου 2dsphere, είναι δυνατή η υποστήριξη ερωτημάτων για τον υπολογισμό γεωμετρίας σε μια γήινη σφαίρα. Ο δείκτης 2dsphere υποστηρίζει όλα τα χωρικά ερωτήματα της MongoDB σχετικά με συμπερίληψη, τομή και εγγύτητα. Για τη δημιουργία ενός 2dsphere ευρετηρίου, χρησιμοποιείται η μέθοδος *db.collection.createIndex ()* και καθορίστε ο τύπος του ευρετηρίου ως 2dsphere.

Ο δείκτης 2dsphere υποστηρίζει δεδομένα που έχουν αποθηκευτεί τόσο ως αντικείμενα GeoJSON όσο και ως παραδοσιακά ζεύγη συντεταγμένων. Στην περίπτωση των ζευγών συντεταγμένων, ο δείκτης μετατρέπει τα δεδομένα σε GeoJSON. [5, 22]

Στην περίπτωση, που γίνεται χρήση του ευρετηρίου 2d, τα δεδομένα αποθηκεύονται σαν σημεία σε ένα δισδιάστατο επίπεδο. Έτσι, είναι δυνατόν να υποστηρίζονται ερωτήματα για τον υπολογισμό γεωμετρίες στο δισδιάστατο Ευκλείδιο επίπεδο. Ο δείκτης 2d, χρησιμοποιείται όταν τα δεδομένα είναι παλιά ή αν η εφαρμογή δεν σκοπεύει να αποθηκεύσετε χωρικά δεδομένα με τη μορφή αντικειμένων GeoJSON. Αν και ο δείκτης αυτός, έχει τη δυνατότητα να υποστηρίξει ερωτήματα $nearSphere που σχετίζονται με την απόσταση, είναι προτιμότερο σε τέτοιου είδους ερωτήματα να χρησιμοποείται ο δείκτης 2dsphere σε συνδυασμό με αντικείμενα GeoJSON. Για τη δημιουργία ένος 2d ευρετηρίου, χρησιμοποιείται η μέθοδος db.collection.createIndex (), με το όρισμα 2d σαν όρισμα. [4, 5]

Κεφάλαιο 2: Τεχνολογικό Υπόβαθρο

2.1 Python

Η Python είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου η οποία δημιουργήθηκε το 1990, από τον Ολλανδό Guido van Rossum. Η Python σαν γλώσσα προγραμματισμού χαρακτηρίζεται από την αναγνωσιμότητα του κώδικά της και την ευκολία χρήσης της τόσο σε μικρής όσο και σε μεγαλύτερης έκτασης εφαρμογές.

Η Python διαθέτει αυτόματη διαχείριση μνήμης και μπορεί να υποστηρίξει πολλαπλά παραδείγματα προγραμματισμού, όπως για παράδειγμα, των αντικειμενοστρεφούς, λειτουργικού και διαδικαστικού. Έχει επίσης πολλές ολοκληρωμένες τυποποιημένες βιβλιοθήκες που παρέχουν κάποιες συνηθισμένες εργασίες, διευκολύνοντας έτσι στην εκτέλεση πολύπλοκων εργασιών αλλά και στην ταχύτητα εκμάθησης της. Επιπλέον το συντακτικό της είναι αρκετά ευέλικτο, καθώς επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες προγραμματισμού όπως η C++ ή η Java.

Οι Python interpreters είναι διαθέσιμοι προς εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας την εκτέλεση του Python κώδικα σε μία ευρεία γκάμα συστημάτων, καθιστώντας την έτσι λογισμικό ανοιχτού κώδικα (open source). Ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του λογισμικού προς χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python. [2, 7]

2.2 NoSQL Βάσεις Δεδομένων

Στις μέρες μας, η ποσότητα των δεδομένων που χρειάζονται να αποθηκευτούν σε μία βάση δεδομένων έχει αυξηθεί δραματικά. Πιο συγκεκριμένα, η εμφάνιση του μεγάλου όγκου δεδομένων είναι άρρηκτα συνδεδεμένη με την ανάπτυξη του Internet και του Cloud Computing. Οι διάφοροι τύποι εφαρμογών που χρησιμοποιούνται απαιτούν τη χρήση

βάσεων δεδομένων που μπορούν να ανταποκριθούν σε αυτές τις ανάγκες. Έτσι, είναι αναγκαία η ταυτόχρονα ανάγνωση και γραφή σε σχετικές γρήγορες ταχύτητες καθώς και η αποτελεσματική επεξεργασία και αποθήκευση μεγάλου όγκου δεδομένων.

Οι παραδοσιακές σχεσιακές βάσεις δεδομένων έχουν να αντιμετωπίσουν πολλές προκλήσεις όταν επεξεργάζονται μεγάλου όγκου δεδομένα. Αυτές οι βάσεις δεδομένων είναι σχεδιασμένες για χρήση όταν τα δεδομένα παρουσιάζουν μεγάλο βαθμό ομοιομορφίας. Επίσης, οι σχεσιακές βάσεις δεδομένων θεωρούν ότι οι ιδιότητες και οι αλληλεπιδράσεις των δεδομένων είναι ορισμένες εξ΄ αρχής. Δυστυχώς, όταν οι παραπάνω υποθέσεις δεν πληρούνται, η σχεσιακή βάση δεδομένων δεν είναι κατάλληλη για την αποθήκευση δεδομένων, όπως στην περίπτωση του μεγάλου όγκου δεδομένων.

Προκειμένου να εξυπηρετηθούν απαιτήσεις, όπως οι παραπάνω, δημιουργείθηκε η βάση δεδομένων NoSQL. Ο όρος NoSQL πρωτοεμφανίστηκε στις αρχές του 2009 σε μια συνάντηση στο SanFrancisco, κάτα την οποία παρουσιάστηκαν αυτή η «νέα» βάση δεδομένων. Σαφής ορισμός σχετικά με ένα σύστημα NoSQL δεν υπάρχει. Το NoSQL, στην κυριολεκτικά, αποτελεί ένα συνδυασμός των λέξεων: No και SQL και αποσκοπεί στην δημιουργία μίας νέας μορφής βάσεων δεδομένων που δεν είναι μόνο σχεσιακές.

Ο όρος NoSQL αναφέρεται σε μια ευρεία ομάδα συστημάτων διαχείρισης βάσεων δεδομένων (Database Management System) που το κύριο χαρακτηριστικό τους είναι το ότι δεν τηρούν το RDBMS μοντέλο (Relational Database Management System), το οποίο και χρησιμοποιείται στην συντριπτική πλειοψηφία των περιπτώσεων, καθώς επίσης χρησιμοποιούν διαφορετικό από τον κλασικό τρόπο της SQL για την διαχείριση και επεξεργασία των δεδομένων μέσα στη βάση (Data Manipulation). Επομένως, τα ονόματα «No RDBMS» και «No relational», θα μπορούσαν να είναι δύο εναλλακτικά ονόματα για το NoSQL. Έτσι, για να δώσουμε την πλήρη έννοια αυτής της νέας μορφής βάσης δεδομένων, χρησιμοποιείται το ακρωνύμιο NoSQL το οποίο επεκτείνεται στο "Not Only SQL". [13, 14]

Οι βάσεις δεδομένων NoSQL είναι κατανεμημένες, μη σχεσιακές (non-relational), σχεδιασμένες για αποθήκευση δεδομένων μεγάλης κλίμακας, ενώ ταυτόχρονα επιτρέπουν την παράλληλη επεξεργασία δεδομένων μοιρασμένα σε έναν μεγάλο αριθμό από servers. Οι NoSQL βάσεις δεδομένων γενικώς δεν χρησιμοποιούν κάποιο δομημένο σύστημα για τα στοιχεία που περιλαμβάνουν, όπως για παράδειγμα πίνακες, οι οποίοι είναι οι δομικές μονάδες για ένα παραδοσιακό σχεσιακό σύστημα, ούτε χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων. Χρησιμοποιούν αποκλειστικά

non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων και είναι κατά κύριο λόγο βελτιστοποιημένες, ώστε να επισυνάπτουν και να ανακτούν τα δεδομένα αυτά .

Τα κύρια χαρακτηριστικά τους συνοψίζονται στα εξής:

1. Η μη χρήση της SQL ως query language
2. Δεν εγγυούνται ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα
3. Έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας.

Τα εν λόγω συστήματα δεν συμμορφώνονται πιστά στους κανόνες ACID (Atomicity, Consistency, Integrity, Durability) των σχεσιακών βάσεων, έτσι ώστε να ανταποκρίνονται σε συνθήκες στις οποίες η απόδοση ενός ερωτήματος είναι πιο σημαντική από την απόλυτη συνέπεια των δεδομένων. Οι βάσεις NoSQL θυσιάζουν τις αυστηρές απαιτήσεις σε συνέπεια για υψηλές ταχύτητες και ελαστικότητα. Επιπλέον, σε αντίθεση με τις σχεσιακές βάσεις που είναι αυστηρά δομημένες, τα δεδομένα στις NoSQL δεν περιορίζονται εν γένει από κάποιο σχήμα. Ένα από τα σημαντικότερα χαρακτηριστικά τους είναι ότι παρέχουν υψηλή διαθεσιμότητα στα δεδομένα του συστήματος. Η φιλοσοφία των NoSQL εστιάζει στα κατανεμημένα συστήματα βάσεων, όπου αδόμητα δεδομένα αποθηκεύονται σε πολλαπλούς κόμβους. Αυτή η κατανεμημένη αρχιτεκτονική επιτρέπει την οριζόντια κλιμάκωση του συστήματος, δίνοντας την δυνατότητα να προστίθενται συνεχώς νέοι πόροι καθώς τα δεδομένα αυξάνονται, χωρίς επιβάρυνση στην απόδοση. Γίνεται αντιληπτό ότι οι υποδομές των NoSQL βάσεων είναι πολύ καλά προσαρμοσμένες στις μεγάλες ανάγκες των μεγάλου όγκου δεδομένων. Για παράδειγμα, η χρήση σχεσιακών βάσεων δεδομένων σε μηχανές αναζήτησης για την αποθήκευση και την αναζήτηση δυναμικών δεδομένων, είναι ανεπαρκής.

Οι βάσεις δεδομένων NoSQL έχουν υιοθετήθεί πλέον από τις μεγαλύτερες εταιρείες στον χώρο του Internet, όπως η Google, η Amazon και το Facebook. Και στις τρείς αυτές περιπτώσεις προέκυψαν προκλήσεις στον χειρισμό τεράστιου όγκου δεδομένων, όπου οι συμβατικές RDBMS, δεν μπορούσαν να ανταπεξέλθουν. Σε τέτοιες περιπτώσεις, οι RDBMS χάνουν όλες τις ιδιότητες τους. Έτσι, το βασικό πλεονέκτημα των NoSQL συστημάτων, είναι η ικανότητα να αποθηκεύουν και να ανακτούν μεγάλες ποσότητες δεδομένων, «αδιαφορώντας» για τις σχέσεις μεταξύ αυτών. Επίσης, οι μέθοδοι υλοποίησης και εφαρμογής τους αξιοποιούν μια αρχιτεκτονική που επιτρέπει (και ίσως διευκολύνει) την κατανεμημένη λειτουργία του συστήματος. Τα χαρακτηριστικά αυτά οδηγούν το σύστημα σε αυξημένες επιδόσεις, αφού παρέχεται η δυνατότητα κλιμάκωσης (θεωρητικά άπειροι

servers όπου κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος). Η σημαντική αυτή αύξηση στην απόδοση και την επεκτασιμότητα για ορισμένα μοντέλα δεδομένων, αντισταθμίζει την μειωμένη ευελιξία του χρόνου εκτέλεσης σε σύγκριση με τη χρήση SQL (RDBMS). Μπορούν να υποστηρίξουν πολλαπλές δραστηριότητες, συμπεριλαμβανομένων διαφόρων αναγνωριστικών και προγνωστικών analytics, την μετατροπή δεδομένων στυλ ETL, και μη κρίσιμες OLTP (για παράδειγμα, την διαχείριση μακράς διάρκειας ή ενδοεπιχειρηματικών συναλλαγών). Αρχικά τα συστήματα αυτά υποκινήθηκαν από Web 2.0 εφαρμογές και είναι σχεδιασμένα για αυξανόμενη κλιμάκωση (scaling) σε χιλιάδες ή εκατομμύρια χρήστες που κάνουν updates καθώς και reads,σε αντίθεση με τα παραδοσιακά DBMS συστήματα και τα data warehouses. Η ιδέα είναι ότι και οι 2 τεχνολογίες μπορούν να συνυπάρξουν και η κάθεμία έχει την δική της θέση. Το κίνημα των NoSQL έχει αναδειχθεί τα τελευταία χρόνια καθώς πολύ πρωτοπόροι του Web 2.0, έχουν υιοθετήσει τα συστήματα αυτά. Εταιρείες όπως το Facebook, Twitter, Digg, Amazon, LinkedIn και η Google, όλες χρησιμοποιούν NoSQL με τον ένα ή με τον άλλο τρόπο.

Η συχνή πλέον χρήση μη-σχεσιακών βάσεων δεδομένων στον τομέα της τεχνολογίας, θα έκανε κάποιον να σκεφτεί ότι τα NoSQL συστήματα έχουν αρχίσει να εκτοπίζουν τα παραδοσιακά σχεσιακά συστήματα. Αυτό στην πραγματικότητα δεν ισχύει. Κάθε ένα από αυτά τα δύο μοντέλα έχει διακριτό ρόλο και είναι κατάλληλο για διαφορετικές εφαρμογές και διαφορετικό φόρτο εργασίας. Τα NoSQL συστήματα δεν εμφανίστηκαν για να αντικαταστήσουν τα σχεσιακά, αλλά για να τα συμπληρώσουν.

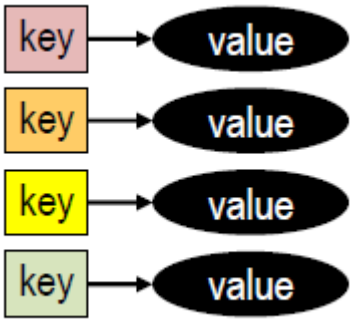
Τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων είναι εξαιρετικά χρήσιμα όταν κάποιος δουλεύει με τεράστιο όγκο δεδομένων, η φύση των οποίων δεν απαιτεί κάποιο σχεσιακό μοντέλο. Για παράδειγμα εταιρείες που συλέγουν μεγάλες ποσότητες «αδόμητων» (unstructured) δεδομένων, στρέφονται όλο και περισσότερο σε μη-σχεσιακές βάσεις. Η κατανεμημένη τους φύση τα καθιστά ιδανικά για μαζική επεξεργασία δεδομένων (όπως για παράδειγμα ενοποίηση, φιλτράρισμα, διαλογή, στατιστικές ενέργειες κλπ). Είναι επίσης πολύ καλά για ανάκτηση και ανταλλαγή δεδομένων μεταξύ μηχανημάτων (machine-to-machine), καθώς και για την επεξεργασία συναλλαγών μεγάλου όγκου, με την προυπόθεση βέβαια χαμηλών απαιτήσεων για ACID ιδιότητες. Ως εκ τούτου, παρέχουν σχετικά φθηνή, υψηλής επεκτασιμότητας αποθήκευση μεγάλου όγκου, όπως για παράδειγμα ιστορικά δεδομένα, αρχεία καταγραφής, αρχεία τηλεφωνικών δεδομένων, ενδείξεις μετρητών και αισθητήρων, καθώς και αποθήκευση ημιδομημένων (semi-structured) ή αδόμητων δεδομένων (unstructured), όπως αρχεία ηλεκτρονικού

ταχυδρομείου, αρχεία XML, έγγραφα, κλπ. Έχουν την ικανότητα της κλιμάκωσης, κάτι στο οποίο υστερούν τα παραδοσιακά συστήματα. Πέρα από τα πλεονεκτήματα κλίμακας, η ίδια η αρχιτεκτονική τους βοηθά στην βελτίωση της απόδοσής τους. Εαν μια σχεσιακή βάση δεδομένων έχει εκατοντάδες χιλιάδες πίνακες, η επεξεργασία των δεδομένων που βρίσκονται στους πίνακες αυτούς θα δημιουργήσει πολλά Locks στα δεδομένα, γεγονός το οποίο θα έχει σαν συνέπεια την υποβάθμιση της απόδοσης του συστήματος. Εν αντιθέση, τα NoSQL συστήματα έχουν πιο αδύναμα μοντέλα συνέπειας των δεδομένων, μπορούν να «θυσιάσουν» την συνοχή για την αποδοτικότητα. Οι NoSQL λύσεις είναι ελκυστικές επειδή μπορούν να διαχειριστούν τεράστιες ποσότητες δεδομένων, σχετικά γρήγορα, σε ένα cluster από servers, οι οποίοι μοιράζονται πόρους μεταξύ τους. Επιπλέον, οι περισσότερες NoSQL λύσεις είναι ανοιχτού κώδικα (open source), κάτι το οποίο τους δίνει πλεονέκτημα τιμής έναντι των συμβατικών εμπορικών βάσεων δεδομένων.

Τα NoSQL συστήματα μπορούν να κατηγοριοποιηθούν ως εξής με βάση την αρχιτεκτονική τους και στο μοντέλο δεδομένων που ακολουθούν :

- Key-values database

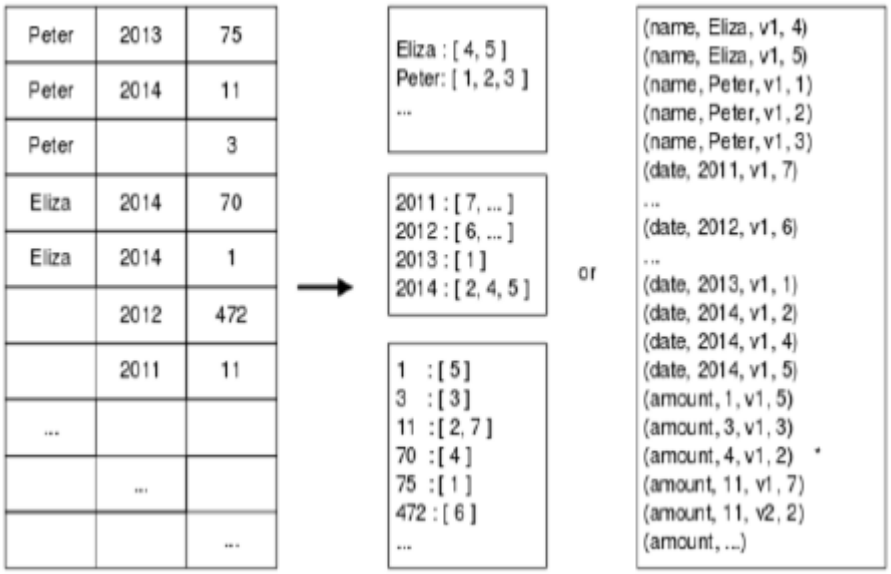
Η πρώτη κατηγορία, οι key-value databases, σημαίνει ότι κάθε τιμή αντιστοιχεί σε ένα κλειδί. Το κλειδί είναι ένα μοναδικό αναγνωριστικό για μια συγκεκριμένη καταχώρηση δεδομένων. Επομένως, δεν πρέπει να επαναλαμβάνεται. Σε αυτού του είδους βάσεων δεδομένω, τα δεδομένα δεν έχουν συγκεκριμένο τύπο ή μορφή, καθώς μπορούν να περιέχουν οποιοδήποτε τύπο δεδομένων όπως εικόνες, βίντεο κ.λπ.. Οι key-value databases αποτελούν το πιο απλό και εύκολο στην υλοποίηση μοντέλο όσον αφορά τις βάσεις NoSQL ενώ ταυτόχτονα αποτελεί τη βάση για όλα τα είδη των NoSQL βάσεων δεδομένων. Οι key-value databases βασίζονται στην ύπαρξη ενός hashtable, όπου με την χρήση ενός map ή dictionary, ένα μοναδικό κλειδί (key) και ένας δείκτης (pointer) «δείχνει» σε ένα συγκεκριμένο στοιχείο. Οι key-value databases συνοδεύονται από μηχανισμούς content caching, για την καλύτερη απόδοση του συστήματος σε μεγάλου όγκου δεδομένων. Παράδειγμα key-value databases είναι οι DynamoDB, FoundationDB, MemcacheDB, Redis, Riak, c-treeACE, Aerospike , Azure Table Storage, LevelDB, Berkeley DB, Oracle NOSQL Database, GenieDB, BangDB, Chordless, Scalaris, Tokyo Cabinet / Tyrant, Voldemort, MemcacheDB. Για να κατανοήσουμε καλύτερα μία key-value databases, μπορούμε να δούμε το ακόλουθο σχήμα. [13, 14, 15, 16]



Εικόνα 1: Key-values database

- Column Family Stores

Στις column family stores, τα δεδομένα αποθηκεύονται σε κελιά (cells), τα οποία ομαδοποιούνται σε στήλες (columns) και όχι σε σειρές (rows) όπως στις σχεσιακές βάσεις δεδομένων. Για την αποθήκευση των δεδομένων χρησιμοποιούνται πίνακες ως μοντέλο δεδομένων για την αποτελεσματική αποθήκευση των δεδομένων. Σε αντίθεση με τις RDBMS, οι πίνακες αυτοί, δεν υποστηρίζουν τη σύνδεση μεταξύ πινάκων. Οι στήλες με την σειρά τους, ομαδοποιούνται σε οικογένειες στηλών (column families), οι οποίες μπορούν να περιλαμβάνουν θεωρητικά άπειρο αριθμό στηλών. Με τον τρόπο αυτό επιτυγχάνονται μεγάλες ταχύτητες σε λειτουργίες search/access, λόγω του ότι όλα τα κελιά που αναφέρονται σε μια στήλη είναι μια συνεχόμενη εγγραφή στο δίσκο. Οι column family stores έχουν εμπνευστεί από τον τομέα των αναλύσεων και της επιχειρησιακής ευφυΐας, όπου η χρήση column stores προσφέρει εφαρμογές υψηλής απόδοσης. Πιο συγκεκριμένα, στις column stores, τα δεδομένα αποθηκεύονται χρησιμοποιώντας ένα σύνολο ζευγών key-values. Τα δεδομένα είναι προσπελάσιμα μέσω ενός πρωτεύον κλειδί ή ένα κλειδί γραμμής. Έτσι, κάθε στήλη αποτελεί το δείκτη της βάσης δεδομένων. Οι column family stores χρησιμοποιούνται από το Bigtable της Google, την υπηρεσία βάσεων δεδομένων NoSQL Big Data της Google. Παραδείγματα τέτοιων βάσεων δεδομένων είναι τα Accumulo, Cassandra, Druid, HBase, Hypertable, Amazon SimpleDB, Cloudata και MonetDB. Στην ακόλουθη εικόνα φαίνεται ένα παράδειγμα μίας column store βάσης δεδομένων. [13, 14, 15, 17]



Εικόνα 2: Παράδειγμα χρήσης column stores [17]

- Document Databases

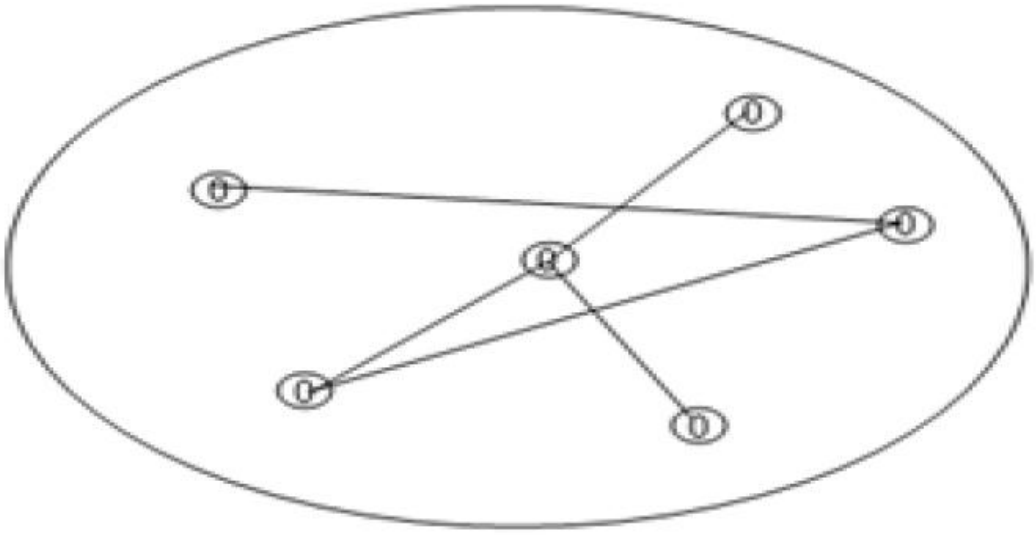
Οι Document databases, ακολουθούν την λογική των key-value stores. Τα δεδομένα σε αυτή την περίπτωση είναι ένα χαλαρά δομημένο σύνολο από ζευγάρια key-value, που οργανώνται φυσικά και λογικά, χωρίς να υπάρχουν περιορισμοί από κάποιο σχήμα. Κάθε εγγραφή (record) και τα συσχετιζόμενα μαζί της δεδομένα, θεωρούνται ως ένα document. Έτσι πολλές φορές, αυτού του τύπου οι βάσεις δεδομένων συγχέονται με τα συστήματα διαχείρισης εγγράφων και περιεχομένου. Τα δεδομένα, σε αυτή την περίπτωση αποθηκεύονται αδόμητα (κείμενα) ή ημιδομημένα σε μορφή JSON, XML ή Binary JSON (BSON) και όχι έγγραφα ή υπολογιστικά φύλλα (αν και αυτά θα μπορούσαν να αποθηκευτούν επίσης). Τα δεδομένα είναι ενθυλακωμένα και κωδικοποιούνται σε ορισμένες τυποποιημένες μορφές ή κωδικοποιήσεις, ενώ προσπελαύνονται μέσω ενός μοναδικού κλειδιού. Επίσης, τα δεδομένα αποθηκεύονται σε ένθετες ιεραρχίες.

Πολλοί θεωρούν ότι οι βάσεις δεδομένων εγγράφων είναι το επόμενο λογικό βήμα από τις απλές key-value stores, καθώς παρέχουν πιο σύνθετες και σημαντικές δομές δεδομένων. Αυτού του είδους βάσεις χρησιμοποιούνται σε εφαρμογές που γίνονται πολλές αναφορές προς τη βάση δεδομένων και το αποτέλεσμα αυτών μπορεί να είναι αναφορές που παράγονται και προκύπτουν από δεδομένα που αλλάζουν συχνά. Οι πιο διαδεδομένες NoSQL βάσεις αυτής της κατηγορίας είναι η MongoDB και η CouchDB. Παράδειγμα άλλων

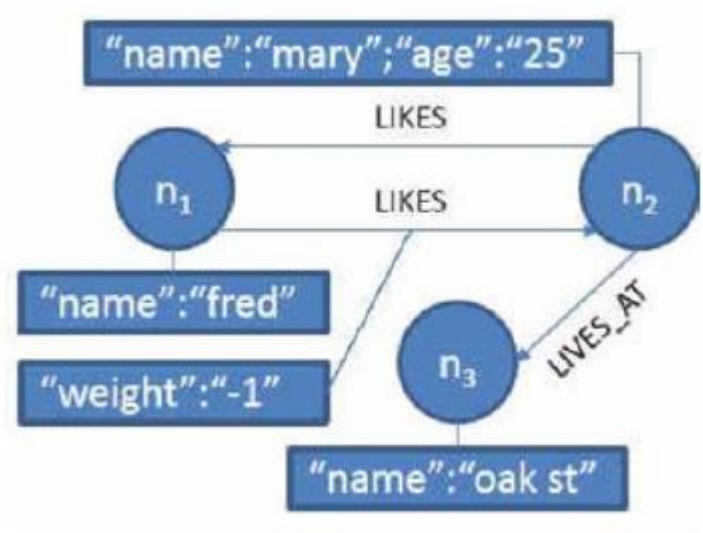
document databases είναι οι Clusterpoint, Apache CouchDB, Couchbase, MarkLogic, MongoDB, Elasticsearch, RethinkDB, NeDB, Terrastore,XML-dbs (BaseX, eXist, Sedna, Qizx). [13, 14, 15, 16]

- Graph Database

  Οι Graph Databases βασίζονται στην θεωρεία γράφων. Χρησιμοποιούν δομές γράφων και πιο συγκεκριμένα κόμβους (nodes), οι οποίοι αντιπροσωπεύουν οντότητες και ακμές που αντιπροσωπεύουν σχέσεις μεταξύ των κόμβων. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graphmodel) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους). Παραδείγματα τέτοιων βάσεων δεδομένων είναι οι Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog, Sparksee, TITAN, InfoGrid, HyperGraphDB, GraphBase. Στο παρακάτω σχήμα παρουσιάζεται μια αφαίρεση μιας βάσης δεδομένων γραφημάτων και μια αναπαράσταση μιας βάσης δεδομένων γραφημάτων. [16]



Εικόνα 3: Graph Database [16]



Εικόνα 4: Απεικόνιση graph database [17]

2.3 MongoDB

Η mongoDB είναι μία open source document oriented non-relational database, η οποία ανήκει στην κατηγορία των NoSQL βάσεων δεδομένων. Αυτό σημαίνει ότι η αποθήκευση των δεδομένων δεν γίνεται σε πίνακες, όπως στις παραδοσιακές σχεσιακές βάσεις δεδομένων, αλλά στην περίπτωση της MongoDB αποθηκεύονται σε σχήματα (schemas) με μορφή παρόμοια με αυτή του προτύπου JSON (JavaScript Object Notation). Πιο συγκεκριμένα, η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα χρησιμοποιώντας μία διαδική αναπαράσταση, που ονομάζεται BSON (Binary JSON). Έτσι, δεν χρειάζεται να χρησιμοποιούνται πίνακες, όπως την περίπτωση των σχεσιακών βάσεων, όπου ο κάθε πίνακας έχει συγκεκριμένο αριθμό πεδίων και οι εγγραφές μπορεί να περιέχουν πολλά κενά εξαιτίας του σταθερού αριθμού πεδίων του πίνακα. Με λίγα λόγια το schema της βάσης δεν είναι απαραίτητα σταθερό. Στην πραγματικότητα, η MongoDB είναι μία βάση δεδομένων που αποτελείται από μία συλλογή εγγράφων στα οποία αποθηκεύονται οι απαιτούμενες πληροφορίες. Στην MongoDB σχετικά μεταξύ τους δεδομένα αποθηκεύονται μαζί για την γρήγορη απάντηση των ερωτημάτων μέσω της γλώσσας ερωτημάτων της MongoDB. Έτσι, διαφορετικά document μεταξύ τους μπορεί να έχουν διαφορετική μορφή. Επιπλέον, δεν είναι αναγκαία η δήλωση της μορφής των εγγράφων στο σύστημα, αφού κάθε έγγραφο είναι περιγραφικό ως προς τη δομή του. Τέλος, θα πρέπει να προσθέσουμε

ότι είναι δυαντόν να προστεθούν πεδία σε κάποιο έγγραφο ανά πάσα στιγμή, χωρίς να επηρεάσει τα υπόλοιπα έγγραφα καθώς και τις υπάρχουσες εγγραφές. [1, 11]

Η μορφή των δεδομένων που αποθηκεύονται σε μία βάση δεδομένων MongoDB ποικίλει. Μπορεί να είναι είτε έγγραφα είτε κάποια δομή δεδομένων, ενώ είναι δυνατή η αλλαγή τους με την πάροδο του χρόνου. Τα δεδομένα που αποθηκεύονται συνήθως αντιστοιχούν σε αντικείμενα στον κώδικα της εφαρμογής, κάνοντας τα ευέλικτα στην επεξεργασία. Τα ερωτήματα προς μία βάση δεδομένων MongoDB, για τη συλλογή δεδομένων είναι δυναμικά και δεν απαιτούν συγκεκριμένους δείκτες. Για τα ερωτήματα χρησιμοποιείται η MongoDB's document-based query language. Στα ερωτήματα αυτά δεν περιέχονται join, καθώς δεν θα είναι εφικτό το scaling. Η mongoDB, όσον αφορά τα δεδομένα που αποθηκεύονται, είναι κατάλληλη για χρήση σε εφαρμογές όπου υπάρχει μεγάλος όγκος δεδομένων, τα οποία αλλάζουν πολύ γρήγορα και απαιτείται ταχύτητα. [18]

Όπως έχουμε ήδη αναφέρει η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα. Κάθε έγγραφο αποτελεί ένας ζεύγος κλειδιών-τιμών (key-value pairs). Κάθε έγγραφο μπορεί να έχει μέγεθος έως και 16MB. Επιπλέον, κάθε έγγραφο έχει ένα μοναδικό _id, με βάση το οποίο δεικτοδοτείται (indexed) αυτόματα. Η τιμή του πεδίου _id παράγεται αυτόματα, αν δεν έχει οριστεί εξ΄ αρχής και χρησιμοποιείται ως το μοναδικό αναγνωριστικό του εγγράφου. Τέλος, θα πρέπει να αναφέρουμε ότι πολλαπλά έγγραφα οργανώνονται σε συλλογές (collections). Οι συλλογές μπορούν να περιέχουν έγγραφα με διαφορετική δομή ως προς το σχήμα (schema), ωστόσο είναι καλή πρακτική να αποθηκεύονται έγγραφα με την ίδια δομή. [18]

Η MongoDB προσφέρει τα ακόλουθα χαρακτηριστικά:

- Ad hoc ερωτήματα

Η MongoDB υποστηρίζει πεδία, ερωτήματα και regular expression για αναζήτηση δεδομένων. Τα ερωτήματα μπορούν να επιστρέφουν συγκεκριμένα πεδία των εγγράφων και μπορεί επίσης, να περιλαμβάνουν συναρτήσεις που ορίζονται από συναρτήσεις της JavaScript. Τα ερωτήματα μπορεί επίσης να ρυθμιστούν να επιστρέφουν ένα τυχαίο δείγμα αποτελεσμάτων ενός δεδομένου μεγέθους.

- Indexing

Τα πεδία σε ένα έγγραφο MongoDB μπορούν να πάρουν την μορφή ευρετηρίου χρησιμοποιώντας πρωτογενείς και δευτερεύοντες δείκτες. Η κατηγοριοποίηση των

δεδομένων σε πραγματικό χρόνο, δίνει τη δυνατότητα και τα εργαλεία για πρόσβαση και ανάλυση των δεδομένων.

- Replication

Το MongoDB παρέχει υψηλή διαθεσιμότητα μέσω αντιγράφων (replica sets). Ένα replica set αποτελείται από δύο ή περισσότερα αντίγραφα δεδομένων. Κάθε replica set μπορεί να έχει το ρόλο πρωτεύοντος ή δευτερεύοντος αντιγράφου κάθε στιγμή. Η εγγραφή και η ανάγνωση δεδομένων γίνονται στο πρωτεύων αντίγραφο. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των πρωτεύων δεδομένων μια τεχνική αντιγραφής. Όταν αποτύχει ένα πρωτεύον αντίγραφο, διεξάγει αυτόματα μια εκλογική διαδικασία στο σύνολο των replica sets για να προσδιοριστεί ποιο δευτερεύον replica set πρέπει να γίνει το κύριο. Τα δευτερεύοντα τμήματα μπορούν προαιρετικά να προσφέρουν λειτουργίες ανάγνωσης, αλλά αυτά τα δεδομένα είναι συνεπή.

- Load balancing

Η MongoDB προσφέρει οριζόντια κλιμάκωση με τη χρήση sharding. Ο χρήστης επιλέγει ένα sharded κλειδί, το οποίο καθορίζει πως τα δεδομένα μιας συλλογής θα διανεμηθούν. Τα δεδομένα χωρίζονται σε εύρη τιμών (με βάση το sharded κλειδί) και κατανέμονται μεταξύ πολλαπλών shards. Ένα shard είναι master με έναν ή περισσότερους slaves. Εναλλακτικά, το shard κλειδί μπορεί να επιτρέψει μια ομοιόμορφη κατανομή δεδομένων.

Το MongoDB μπορεί να τρέχει παράλληλα σε πολλαπλούς servers, εξισορροπώντας το φορτίο ή αντιγράφοντας τα δεδομένα για να διατηρήσει το σύστημα σε λειτουργία σε περίπτωση αποτυχίας υλικού.

- Αποθήκευση αρχείων (File Storage)

Η MongoDB μπορεί να χρησιμοποιηθεί σαν ένα σύστημα αρχείων με λειτουργίες εξισορρόπησης φορτίου και αντιγραφής δεδομένων σε πολλαπλά μηχανήματα για την αποθήκευση αρχείων. Η λειτουργία αυτή ονομάζεται grid file system. Η MongoDB διαθέτει λειτουργίες επεξεργασίας των αρχείων και του περιεχομένου τους που μπορούν να χρησιμοποιηθούν από προγραμματιστές.

- Aggregation

Το MapReduce μπορεί να χρησιμοποιηθεί για την επεξεργασία δεδομένων σε ομάδες και λειτουργίες συγκέντρωσης (aggregation). Η λειτουργία aggregation δίνει την δυνατότητα

στους χρήστες να πάρουν αποτελεσμάτων παρόμοια με αυτά που προκύπτουν από το SQL GROUP BY. Επιπλέον, η λειτουργία aggregation δίνει την δυνατότητα σχηματισμού pipeline αντίστοιχο του Unix. Τέλος, η aggregation προσφέρει τη λειτουργία $lookup για τη συνένωση εγγράφων από πολλαπλά έγγραφα.

- Χρήση JavaScript στην πλευρά του server

Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα, aggregation συναρτήσεις που αποστέλλεται απευθείας στη βάση δεδομένων προς εκτέλεση.

- Capped collections

Η MongoDB υποστηρίζει συλλογές σταθερού μεγέθους που ονομάζονται capped collections. Αυτός ο τύπος συλλογής επιτρέπει την εισαγωγή δεδομένων με τη σειρά και φτάσει σε ένα καθορισμένο μέγεθος, συμπεριφέρεται σαν circular queue.

Με βάση όλα τα παραπάνω χαρακτηριστικά, η MongoDB είναι μια κατανεμημένη βάση δεδομένων που είναι εύκολη στη χρήση [18].

Στη συνέχεια, θα κάνουμε μία σύγκριση της MongoDB με την MySQL. Οι διαφορές μεταξύ τους, φαίνονται στον ακόλουθο πίνακα.

Πίνακας 1: Σύγκριση MySQL και MongoDB [11]

MySQL	**MongoDB**
ACID Transactions	ACID Transactions
Πίνακας (Table)	Συλλογές (Collection)
Γραμμές (Row)	Έγγραφα (Field)
Στήλες (Column)	Πεδία (Field)
Δευτερεύουσα Δεικτοδότηση	Δευτερεύουσα Δεικτοδότηση
JOINs	Ενσωματωμένα Έγγραφα
GROUP_BY	Aggregation Pipeline

Τα δεδομένα στην MySQL δεν διαθέτουν ένα ευέλικτο σχήμα (schema), το οποίο σημαίνει ότι το σχήμα και τα πεδία ενός πίνακα θα πρέπει να είναι προκαθορισμένα πρίν την εισαγωγή των δεδομένων σε αυτόν. Επιπλέον η MySQL αποθηκεύει τα δεδομένα σε γραμμές, σε αντίθεση με την MongoDB, που μπορεί να κρατά documents με μεγάλου όγκου datasets. Επιπλέον, στην MongoDB, είναι δυνατόν οι διαφορετικές εγγραφές να έχουν

διαφορετικά χαρακτηριστικά σε σχέση με τις υπόλοιπες. Επιπλέον, η MongoDB δίνει την δυνατότητα διαχείρισης json αρχείων, ενώ δεν χρειάζεται το σχήμα της βάσης να είναι προκαθορισμένο εξ΄αρχής όπως συμβαίνει με την στην MySQL [11].

2.4 Αποθήκευση Δεδομένων στη MongoDB

Η βασική απόφαση σχετικά με το μοντέλο δεδομένων που θα χρησιμοποηθεί σε κάποια MongoDB εφαρμογή είναι άμεσα συνδεδεμένο με τη δομή των εγγράφων αλλά και τις σχέσεις μεταξύ των δεδομένων. Οι δύο τεχνικές που ακολουθούνται είναι είτε η χρήση εγγράφων με αναφορές (references documents) είτε τα ενσωματωμένα έγγραφα (embedded documents). Στην πρώτη περίπτωση, μέσω των αναφορών είναι δυνατή η αποθήκευση των σχέσεων μεταξύ των δεδομένων με τη χρήση συνδέσμων (links) ή αναφορές (references) από το ένα έγγραφο στο άλλο. Οι εφαρμογές, με τη σειρά τους, μπορούν να αξιοποιήσουν αυτές τις αναφορές για να έχουν πρόσβαση στα δεδομένα που αιτούνται κάθε στιγμή. Αυτή η κατηγορία, ονομάζεται κανονικοποιημένα μοντέλα δεδομένων (normalized data models).

Στην περίπτωση των ενσωματωμένων εγγράφων οι σχέσεις μεταξύ των δεδομένων καθορίζονται με την αποθήκευση των σχετιζόμενων δεδομένα σε ένα και μόνο έγγραφο. Τα έγγραφα της MongoDB επιτρέπουν τις ενσωματωμένες δομές εγγράφων σε ένα πεδίο (field) ή σε πίνακα (array) τέτοιων εγγράφων. Τα μη κανονικοποιημένα μοντέλα δεδομένων (denormalized data models) όπως ονομάζονται, επιτρέπουν στις εφαρμογές να αλληλοεπιδρούν με τα δεδομένα με λιγότερες λειτουργίες από τα κανονικοποιημένα μοντέλα.

Η MongoDB υποστηρίζει τις CRUD λειτουργίες (Create, Read, Update, and Delete) για να αλληλοεπιδρά με τα αποθηκευμένα δεδομένα. Η δημιουργία ή η εισαγωγή ενός εγγράφου σε μια συλλογή πραγματοποιείται με τις εντολές db.collection.insertOne() για ένα έγγραφο και db.collection.insertMany() για πολλαπλά. Η ανάγνωση και ανάκτηση εγγράφων από μια συλλογή γίνεται με την εντολή db.collection.find(). Σε αυτήν την εντολή μπορούμε να ορίσουμε φίλτρα και κριτήρια ώστε να προσδιορίσουμε τα έγγραφα που επιθυμούμε να επιστραφούν. Οι λειτουργίες ενημέρωσης τροποποιούν τα υπάρχοντα έγγραφα σε μια συλλογή. Οι εντολές που πραγματοποιούν ενημερώσεις είναι: db.collection.updateOne(), db.collection.updateMany(), db.collection.replaceOne(). Φίλτρα και κριτήρια μπορούν,

επίσης, να χρησιμοποιηθούν όπως ακριβώς στις εντολές ανάγνωσης. Τέλος, οι εντολές διαγραφής εγγράφων από μια συλλογή, χρησιμοποιούν επίσης, φίλτρα με τον ίδιο τρόπο. Οι εντολές αυτές είναι η db.collection.deleteOne() και η db.collection.deleteMany().

Κεφάλαιο 3: Υλοποίηση – Επεξεργασία Δεδομένων

3.1 Αποθήκευση Δεδομένων

Η εργασία μας σχετίζεται με την εύρεση πληροφοριών σε μία περιοχή ενδιαφέροντος. Οι πληροφορίες είναι για παράδειγμα, εστιατόρια, ξενοδοχεία κ.α. Για να είναι αυτό εφικτό, αποθηκεύουμε όλα τα διαθέσιμα εστιατόρια σε ένα αρχείο τύπου csv, που ονομάζεται restaurants_new.csv και στη συνέχεια, τα αποθηκεύουμε στην MongoDB. Τα διαθέσιμα εστιατόρια αποτελούν χωρικά δεδομένα και διαθέτουν τέσσερες διαφορετικές πληροφορίες. Πιο συγκεκριμένα, κάθε χωρικό δεδομένο, έχει ένα μοναδικό id, καθορίζεται από τις γεωγραφικές συντετγμένες του, όπως το γεωγραφικό μήκος και πλάτος τους καθώς και μία πληροφορία σχετικά με τον τύπο του σημείου και τέλος το ονομά του εστιατορίου.

Για να είναι δυνατή η σύνδεση με τη MongoDB, χρησιμοποιείται ένα μοναδικό αντικείμενο (singleton), που ονομάζεται MongoClient. Ο MongoClient χρησιμοποιείται από την εφαρμογή χρήστη για τη δημιουργία ενός αντικειμένου τύπου MongoDB. Όσον αφορά την εφαρμογή χρήστη, αυτή είναι η μοναδική λειτουργικότητα του συστατικού αυτού, καθώς στα επόμενα στάδια η συμμετοχή του προκαλείται από άλλα εσωτερικά συστήματα και όχι από την εφαρμογή χρήστη.

Έτσι, δημιουργείται το αντικείμενο MongoDB, το οποίο αποτελεί τη μοναδική διεπαφή, μέσω της οποίας ο χρήστης μπορεί να εκτελεί όλες τις λειτουργίες δημιουργίας, ολοκλήρωσης ή ματαίωσης κάποιας συναλλαγής. Το αντικείμενο αυτό, έχει τη δυνατότητα να επικοινωνεί εσωτερικά με τα υπόλοιπα εμπλεκόμενα συστήματα χωρίς ο χρήστης να έχει εικόνα για το πώς γίνεται διαχείριση αυτών των ενεργειών.

Με τις ακόλουθες γραμμές κώδικα, πραγματοποιείται η σύνδεση στην MongoDB:

```
1. from pymongo import MongoClient
2. client = MongoClient('mongodb://localhost:27017/')
3. db = client.test2
```

Για την εισαγωγή των δεδομένων στην MongoDB, χρησιμοποιείται η συνάρτηση insert(), η οποία παίρνει σαν όρισμα ένα όνομα αρχείου, στο οποίο είναι αποθηκευμένα τα δεδομένα. Για παράδειγμα, η παρακάτω συνάρτηση καλείτε με όρισμα το αρχείο restaurants_new.csv, που αναφέραμε παραπάνω για την αποθήκευση των εστιατορίων στην MongoDB.

```
1.  def insert(filename):
2.          with open(filename) as file:
3.                  for line in file:
4.                       print (line)
5.                       data = json.loads(line)
6.                       result = db.testcol1.insert_one (data)
```

3.2 Διαχείριση Δεδομένων

Για να προκύψουν πληροφορίες από τα δεδομένα που έχουμε αποθηκευμένα στη MongoDB, θα πρέπει να θέσουμε ορισμένα ερωτήματα προς αυτήν. Για αυτό το λόγο, έχει υλοποιηθεί η συνάρτηση find_agg(), η οποία παίρνει σαν όρισμα κάποιο ερώτημα προς την MongoDB και επιστρέφει μία γραφική παράσταση με τα σημεία που έχουν προκύψει από το ερώτημα.

```
1.  def find_agg(query):
2.  result = []
3.  cnt = 0
4.  a = []
5.  for x in coll.find(query):
6.          result.append(x)
7.          print(x)
8.          cnt = cnt + 1
9.          a.append(cnt)
10. plt.plot(result)
11. plt.show()
12. plt.scatter(a,result)
13. plt.show()
```

Στη συνέχεια, φαίνεται ένα παράδειγμα ενός ερωτήματος προς την MongoDB. Στο ερώτημα αυτό, δίνεται ένα συγκεκριμένο σημείο και αναζητάμε όλα τα εστιατόρια που βρίσκονται εντός ενός χιλιομέτρου από το σημείο αυτό. Πιο συγκεκριμένα, δίνονται οι γεωγραφικές συντεταγμένες και ο τύπος της τοποθεσίας μέσω $geometry και ζητάμε τα πιο κοντινά εστιατόρια $nearSphere σε μέγιστη απόσταση $maxDistance.

```
query = { "location": { "$nearSphere": { "$geometry": { "type": "Point", "coordinates": [ -73.93414657, 40.82302903 ] }, "$maxDistance": 1000}}}
```

Η MongoDB προσφέρει τέσσερεις διαφορετικές λειτουργίες που μπορούν να χρησιμοποιηθούν σε ερωτήματα χωρικών δεδομένων. Οι τέσσερεις αυτές λειτουργίες είναι οι ακόλουθες

- *$geoIntersects*: Επιλέγει τα χωρικά αντικείμενα που τέμνονται εντός ενός GeoJSON. Η λειτουργία αυτή υποστηρίζεται σε ευρετήρια 2dsphere.
- *$geoWithin*: Επιλέγει χωρικά αντικείμενα που βρίσκονται εντός μιας γεωμετρίας οριοθετημένης από ένα GeoJSON. Η λειτουργία αυτή υποστηρίζεται τόσο σε ευρετήρια 2dsphere όσο και σε 2d.
- *$near*: Επιστρέφει χωρικά αντικείμενα που βρίσκονται κοντά σε ένα συγκεκριμένο σημείο, που δίνεται σαν είσοδος. Η λειτουργία αυτή υποστηρίζεται τόσο σε ευρετήρια 2dsphere όσο και σε 2d.
- *$nearSphere*: Επιστρέφει χωρικά αντικείμενα που βρίσκονται κοντά σε ένα σημείο, που δίνεται σαν είσοδος, σε μια σφαίρα. Η λειτουργία αυτή υποστηρίζεται τόσο σε ευρετήρια 2dsphere όσο και σε 2d.

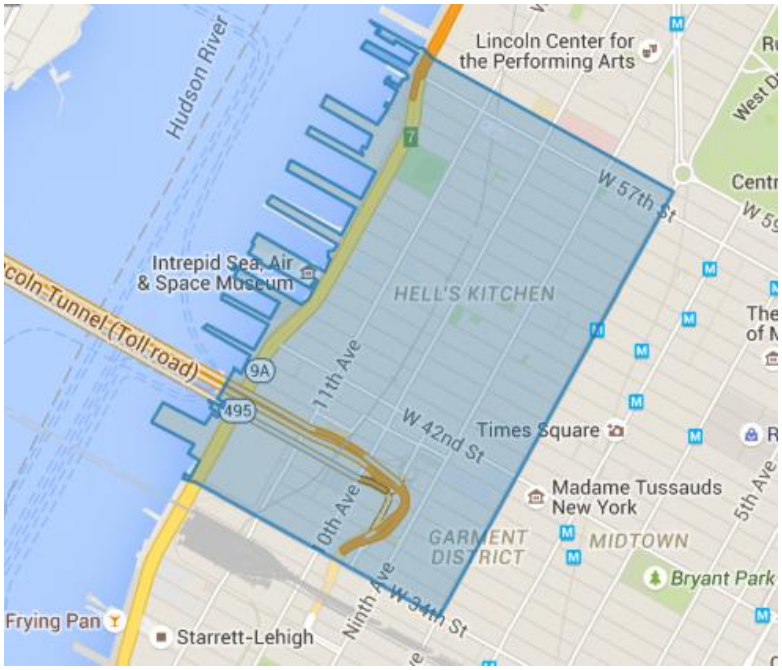
3.3 Υπολογισμός Έυρους για Εστιατόρια και Γειτονίες

Στην προηγούμενη ενότητα, δώσαμε ένα απλό παράδειγμα εύρεσης εστιατορίων σε μία συγκεκριμένη απόσταση με βάση ένα αρχικό σημείο. Στην ενότητα αυτή, με την χρήση μεγαλύτερων περιοχών, όπως για παράδειγμα, είναι μία γειτονιά, θα εντοπίσουμε εστιατόρια σε ένα συγκεκριμένο εύρος. Για να είναι αυτό εφικτό, αποθηκευουμε με ένα σύνολο από γειτονιές MongoDB, με την χρήση της συνάρτησης insert() που περιγράφηκε προηγουμένως.

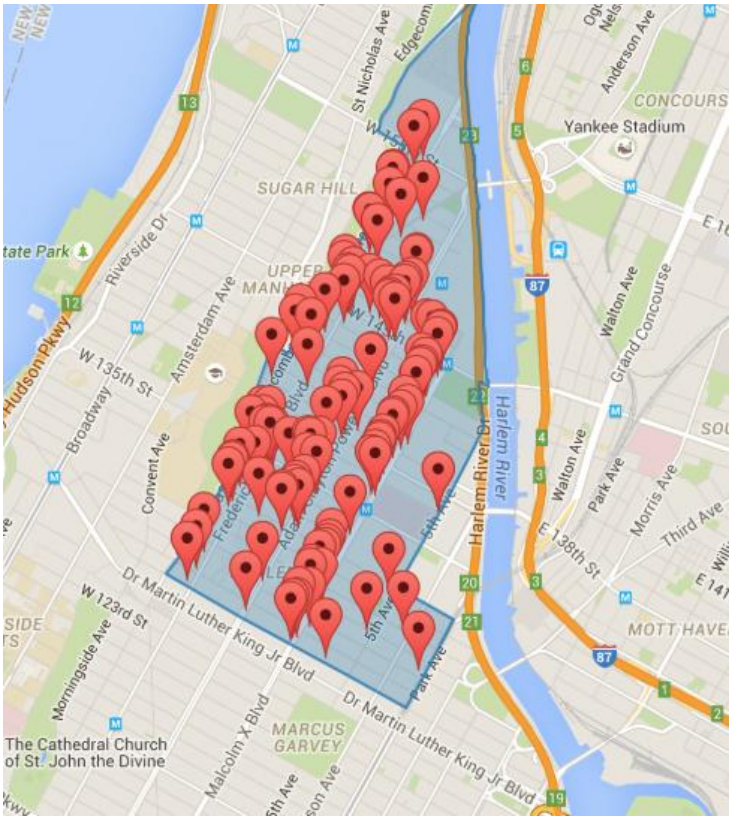
Για να είναι δυνατή η εύρεση εστιατορίων μέσα σε ένα συγκεκριμένο εύρος, αναπτύξαμε τη συνάρτηση range(), που φαίνεται παρακάτω, η οποία όπως και η συνάρτηση *find_agg()* παραπάνω παίρνει σαν όρισμα ένα ερώτημα προς τη MongoDB. Στο κομμάτι κώδικα που ακολουθεί βλέπουμε ένα παράδειγμα ερωτήματος, για την εύρεση εστιατορίων στην γειτονιά Bedford (γραμμή 13). Το ερώτημα αυτό, εφαρμόζεται στην συλλογή *coll1*, όπου είναι αποθηκευμένες όλες οι γειτονιές (γραμμή 14). Τέλος, στη γραμμή 15, επιλέγονται όλα τα έγγραφα που έχουν συγκεκριμένο σχήμα και επιστρέφονται όλα τα εστιατόρια που έχουν κοινές συντεταγμένες με τη γειτονιά “Hell’s Kitchen”.

```
1. import json
2. from pymongo import MongoClient
3. import math
4. import matplotlib.pyplot as plt
5. from bson.code import Code
6. client = MongoClient('mongodb://localhost:27017/')
7. db = client.test2
8. coll = db.testcol1
9. coll1 = db.testcol10
10.def range_find(query):
11.     for x in coll.find(query):
12.           print(x)
13.query1 = { "name": " Hell’s Kitchen " }
14.nb = coll1.find_one(query1)
15.query = { "location": { "$geoWithin": {"$geometry":nb["geometry"]}}}
16.range_find(query)
```

Στη συνέχεια, παραθέτουμε ένα παράδειγμα όπου με βάση μια συγκεκριμένη γειτονια, εμφανίζονται τα εστιατόρια που βρίσκονται σε αυτή την περιοχή.



Εικόνα 5: Ορισμός της γειτονίας Hell's Kitchen μέσω MongoDB [5]



Εικόνα 6: Εύρεση εστιατορίων στη γειτονιά Hell's Kitchen [5]

Κεφάλαιο 4: Replication, Sharding

4.1 Replication

Σκοπός του replication είναι η δυνατότητα αποθήκευσης δεδομένων σε παραπάνω από έναν κόμβους. Αυτό είναι πολύ σημαντικό, καθώς βελτιώνει την διαθεσομότητα των δεδομένων. Για να επιτευχθεί αυτό, αντιγράφουμε τα αποθηκευμένα δεδομένα, από μία βάση δεδομένων, που φιλοξενείται σε κάποιον server σε κάποια άλλη βάση δεδομένων σε έναν διαφορετικό server. Με αυτό τον τρόπο, όλοι οι χρήστες έχουν πρόσβαση στα ίδια δεδομένα χωρίς περιορισμούς και με μεγαλύτερη ταχύτητα. Για να είναι αυτό εφικτό, ακολουθείται μία υλοποίηση “master-slave”. Πιο συγκεκριμένα, ο κύριος κόμβος είναι ο master και με βάση αυτόν όλα τα δεδομένα αντιγράφονται αυτόματα στον κόμβο slave.

Στη συνέχεια, παραθέτουμε τα διαφορετικά είδη replication, που μπορούν να χρησιμοποιηθούν:

1. Binary Replication

Κάθε slave κρατάει μία εγγραφή binary log. Το binary log είναι το αρχείο που περιέχει όλα τα γεγονότα που περιγράφουν αλλαγές όπως για παράδειγμα δημιουργία πινάκων, ενημερώσεις δεδομένων και σβήσιμο πινάκων.

2. GTID Replication

Σε αυτή την περίπτωση χρησιμοποιείται ένας μοναδικός identifier. Πιο συγκεκριμένα, κάθε transaction που γίνεται commit από τον master προς τους slave, χαρακτηρίζεται από έναν μοναδικό identifier, που χρησιμοποιείται σαν ταυτότητα σε όλους τους servers.

3. Multimaster

Τα διάφορα αρχεία μπορούν να τροποποιηθουν από οποιοδήποτε μέλος ενός group. Για να είναι εφιξκτό αυτό, οι κόμβοι θα πρέπει να συνδέονται με τον master, ώστε να μπορούν να τροποποιήθουν. [8]

4.2 Εφαρμογή Replication

Στη συνέχεια, θα περιγράψουμε τα βήματα για να υλοποιήσουμε replication στην MongoDB. Θα υλοποιήσουμε, ένα replication με δύο κόμβους, τον master και τον slave, μέσω Linux και συγκεκριμένα Red Hat. Για να πετύχουμε την υλοποίηση “master-slave”, θα χρησιμοποιηθούν δύο VMs. [9]

1. Αρχικά και στα δύο VMs, προσθέτουμε στο αρχείο /etc/hosts τους ακόλουθους δύο κόμβους:

   192.168.178.182 mongo-repl-1

   192.168.178.109 mongo-repl-2

2. Στη συνέχεια, στο αρχείο */etc/mongod.conf*, και στα δύο VMs, προσθέτω τις ακόλουθες γραμμές:

   - Primary:

     ```
     net:
        port:27017
        bindIp: 127.0.0.1, 192.168.178.182
     replication:
        replSetName: rs0
     ```

   - Secondary:

     ```
     net:
        port:27017
        bindIp: 127.0.0.1, 192.168.178.109
     replication:
        replSetName: rs0
     ```

3. Ακολούθως, κάνουμε restart, το process του MongoDB:

   sudo systemctl restart mongo

4. Στη συνέχεια, κάνουμε τη σύνδεση μεταξύ των δύο VMs

   - Primary:

     rs.initiate()

     rs.add(“mongo-repl-2”)

5. Τέλος, για να γίνει το replication.

   - Primary:

     use exampleDB

     for (var i = 0; i <= 10; i++) db.exampleCollection.insert({ x : i })

   - Secondary:

     db.getMongo().setSlaveOk()

use exampleDB

db.exampleCollection.find()

4.3 Sharding

Με τον όρο sharding αναφερόμαστε σε μία μέθοδο διαμοιρασμού των δεδομένων σε πολλές μηχανές. Η MongoDB χρησιμοποιεί το sharding, για να υποστηρίξει πολύ μεγάλο όγκο δεδομένων αλλά να έχει υψηλότερη απόδοση. Αυτό συμβαίνει καθώς συστήματα βάσεων δεδομένων με μεγάλο όγκο δεδομένων ή εφαρμογές με υψηλό throughput μπορεί να έχουν αρνητικά αποτελέσματα στην επίδοση ενός διακομηστή.

Υπάρχουν δύο διαφορετικοί μέθοδοι για την ανάπτυξη ενός συστήματος, η κατακόρυφη και η οριζόντια κλιμάκωση (scaling). Στη συνέχεια, δίνουμε μια σύντομη περιγραφή αυτών των δύο μεθόδων:

Η κατακόρυφη κλιμάκωση (vertical scaling), έχει σαν αποτέλεσμα την αύξηση της χωρητικότητας ενός μόνο διακομιστή, όπως για παράδειγμα τη χρήση μίας πιο ισχυρής CPU, τη προσθήκη περισσότερης μνήμης RAM ή την αύξηση του χώρου αποθήκευσης. Ωστόσο, αυτό δεν είναι πάντα εφικτό, καθώς διάφοροι περιορισμοί στην διαθέσιμη τεχνολογία μπορεί να περιορίζουν ένα μηχάνημα από το να είναι αρκετά ισχυρό για να εξυπηρετήσει ένα συγκεκριμένο φόρτο εργασίας. Έτσι, υπάρχει ένα πρακτικό μέγιστο για κατακόρυφη κλιμάκωση.

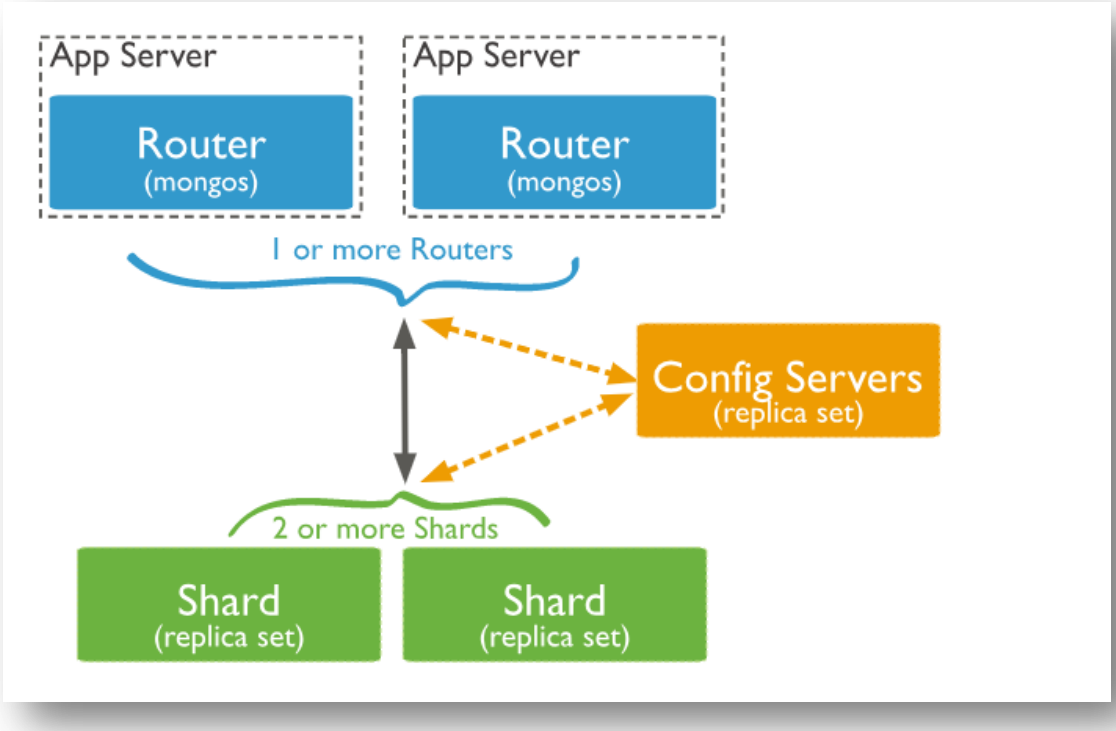
Η οριζόντια κλιμάκωση (horizontal scaling) περιλαμβάνει το διαμοιρασμό ενός συνόλου δεδομένων και του φόρτίου σε πολλαπλούς διακομιστές. Για να το πετύχει αυτό, απαιτείται η προσθήκη επιπλέον διακομιστών ώστε να αυξηθεί η χωρητικότητα του συστήματος, όπως απαιτείται. Έτσι, ενώ η συνολική ταχύτητα ή η χωρητικότητα ενός μηχανήματος μπορεί να μην είναι υψηλή, κάθε μηχάνημα χειρίζεται ένα υποσύνολο του συνολικού φόρτου εργασίας, παρέχοντας, πιθανότατα, καλύτερη απόδοση από ένα μόνο διακομιστή υψηλής ταχύτητας και υψηλής χωρητικότητας. Για να είναι αυτό εφικτό, απαιτείται μόνο η ανάπτυξη επιπλέον διακομιστών σε σχέση με τις ανάγκες του συστήματος. Τελικά, το αποτέλεσμα μπορεί να είναι πιο οικονομικό σε σχέση με τη χρήση υλικών υψηλής τεχνολογίας για ένα μόνο μηχάνημα. Παρόλα αυτά, υπάρχει αυξημένη πολυπλοκότητα τόσο στην υποδομή όσο και στην συντήρηση του συστήματος.

Η MongoDB υποστηρίζει οριζόντια κλιμάκωση μέσω του sharding.

Ένα sharded cluster της MongoDB αποτελείται από τα ακόλουθα στοιχεία:

- **shard**: κάθε shard περιέχει ένα υποσύνολο δεδομένων και μπορεί να υλοποιηθεί με τη χρήση ενός replication set.
- **mongos**: κάθε mongos λειτουργεί σαν ένα router και παρέχει μία διεπαφή μεταξύ των client εφαρμογών και των sharded clusters.
- **config servers:** αποθηκεύει τα metadata και διαμορφώνει τις ρυμίσεις για το cluster.

Στην ακόλουθη εικόνα απεικονίζεται η αλληλεπίδραση των διάφορων εξαρτημάτων ενός shared cluster [10].



Εικόνα 7: Αλληλεπίδραση εξαρτημάτων σε ένα shared cluster [10]

4.4 Πλεονεκτήματα Sharding

1. **Read/Writes**

Η MongoDB διαμοιράζει τις λειτουργίες read και write μεταξύ των shards στο sharded cluster, επιτρέποντας, έτσι, στο κάθε shard να διαχειρίζεται ένα υποσύνολο από τις λειτουργίες στο cluster. Οι λειτουργίες read και write μπορούν να κλιμακώνονται οριζόντια μεταξύ του cluster προσθέτοντας περισσότερα shards.

2. **Χωρητικότητα αποθήκευσης**

Με το sharding είναι δυνατόν ο διαμοιρασμός των δεδομένων μεταξύ των shards σε ένα cluster, επιτρέποντας σε κάθε shard να περιέχει ένα υποσύνολο από τα συνολικά δεδομένα του cluster. Όσο αυξάνονται ο όγκος των δεδομένων αυξάνεται και η χωρητικότητα αποθήκευσης του cluster.

3. Υψηλή Διαθεσιμότητα

Ένα sharded cluster είναι δυνατόν να εκτελεί λειτουργίες read και write ακόμα και αν ένα shard δεν είναι διαθέσιμο. Παρότι, ένα υποσύνολο δεδομένων στα μη διαθέσιμων shards, μπορεί να μην είναι προσβάσιμο για κάποιο χρονικό διάστημα, οι λειτουργίες read και write μπορούν να εκτελούνται στα διαθέσιμα shards με επιτυχία. [10]

Κεφάλαιο 5: Συμπεράσματα

Στην συγκεκριμένη εργασία υλοποιήθηκαν κάποιες λειτουργίες της MongoDB που σχετίζονταν με γεωγραφικά δεδομένα. Για να είναι εφικτές αυτές, έγινε αποθήκευση των δεδομένων στην MongoDB με τη χρήση της γλώσσας προγραμματισμού python. Επιπλέον, υλοποιήθηκαν κάποιες συναρτήσεις, μέσω των οποίων ο χρήστης μπορεί να πραγματοποιεί κάποιες λειτουργίες της MongoDB. Ένα παράδειγμα αυτών των συναρτήσεων είναι η εύρεση κάποιων σημεία ενδιαφέροντος σε μία συγκεκριμένη κλίμακα από ένα αρχικό σημείο. Επιπλέον, Είναι δυνατόν, δοθέντος ενός σημείου ενδιαφέροντος Α, να βρίσκονται ποια σημεία ενδιαφέροντος Β βρίσκονται κοντά στο εκάστοτε σημείο ενδιαφέροντος Α.

Επίπλέον με τη χρήση Virtual Machines πραγματοποιήσαμε το replication στην mongodb, με τη χρήση ενός συστήματος master-slave. Πιο συγκεκριμένα, οι εγγραφές στη βάση δεδομένων του master, γράφονται αυτόματα στην αντίστοιχη βάση δεδομένων στον slave.

Στη συγκεκριμένη διπλωματική εργασία, αναφερθήκαμε και αναλύσαμε όλους τους όρους που συναντάμε στις παραπάνω υλοποιήσεις αλλά και σε άλλες λειτουργίες. Επιπλέον, συγκρίναμε τις δυνατότητες της MongoDB σε συνδυασμό με τις σχεσιακές βάσεις δεδομένων. Τέλος, στα παραρτήματα παρουσιάζεται ο τρόπος με τον οποίο γίνεται εγκατάσταση της MongoDB στο Linux. Επίσης, παρουσιάζεται ένας τρόπος για να κάνουμε backup στην MongoDB.

Βιβλιογραφία

[1] MongoDB. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/MongoDB

[2] Python (programming language). Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/Python_(programming_language)

[3] Robo 3T. Διαθέσιμο από τον διαδικτυακό τόπο: https://robomongo.org/

[4] 2d Indexes. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/core

[5] Geospatial Queries. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/tutorial/geospatial-tutorial/

[6] Install MongoDB Community Edition on Red Hat Enterprise or CentOS Linux. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/

[7] Python. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.python.org/

[8] Replication. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/replication/

[9] Deploy a Replica Set. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/tutorial/deploy-replica-set/

[10] Deploy a Replica Set. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/sharding/

[11] MongoDB and MySQL Compared. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.mongodb.com/compare/mongodb-mysql

[12] MongoDB Backup Methods. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/core/backups/

[13] Jing, H., Haihong, E., Guan, L. and Jian, D. (2011). Survey on NoSQL Database. In: Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on.

[online] Port Elizabeth, South Africa: IEEE, pp. 76-78. Διαθέσιμο από τον διαδικτυακό τόπο: https://ieeexplore.ieee.org/document/6106531/

[14] Shashank, T. (2011). Professional NoSQL. Indianapolis: John Wiley & Sons, Inc.

[15] Strauch C. (2018). NoSQL Databases. 1st ed. [ebook] Stuttgart: Stuttgart Media University. Διαθέσιμο από τον διαδικτυακό τόπο: http://www.christof-strauch.de/nosqldbs.pdf

[16] Vatika, S. and Meenu, D. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), [online] Volume 2(8), p. 20 – 27. Διαθέσιμο από τον διαδικτυακό τόπο: http://ijarcsse.com/Before_August_2017/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf

[17] Cansu B. (2014). SQL vs. NoSQL. 1st ed. [pdf] Trondheim: Norwegian University of Science and Technology. Διαθέσιμο από τον διαδικτυακό τόπο: http://folk.ntnu.no/preisig/HAP_Specials/AdvancedSimulation_files/2014/AdvSim-2014__Birgen_Cansu_Databases.pdf

[18] What is MongoDB?. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.mongodb.com/what-is-mongodb

[19] Takashi O., Yuichi H. and Thad W. (2011). Developments in Earth Surface Processes. Volume: 15, pp. 189 - 224. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.sciencedirect.com/science/article/pii/B9780444534460000070

[20] Spatial Data Use and Dissemination. Instituto Brasileiro de Geogrfia e Estatistica. Διαθέσιμο από τον διαδικτυακό τόπο: https://unstats.un.org/unsd/demographic/meetings/wshops/Chile_31May11/docs/country/brazil02-s10.pdf

[21] Zimanyi E. (2013). Spatial Databases. Department of Computer & Decision Engineering (CoDE), University Libre de Bruxelles. Διαθέσιμο από τον διαδικτυακό τόπο: https://cs.ulb.ac.be/public/_media/teaching/infoh415/spatialnotes.pdf

[22] 2dsphere Indexes. Διαθέσιμο από τον διαδικτυακό τόπο: https://docs.mongodb.com/manual/core/2dsphere/

Παράρτημα

1. Robo 3T

Το Robo 3T είναι ένα γραφικό περιβάλλον για τον χρήστη το όποιο επεξεργάζεται δεδομένα με την χρήση του MongoDB. Το Robo 3T δημιουργήθηκε από τα 3T Software Labs τα όποια σχεδίασαν και το MongoDB. Το Robo 3T ξεχωρίζει έχοντας τα εξής χαρακτηριστικά:

1. Είναι ένα λογισμικό με ελεύθερη χρήση οποιοσδήποτε μπορεί να το δοκιμάσει δωρεάν.
2. Είναι ένα λογισμικό με εύκολο και απλό σχεδιασμό των διεπιφανιών.
3. Βασίζεται στο κώδικα τον οποίο χρησιμοποιείς για την MongoDB.
4. Μπορεί να υλοποιηθεί από όλα τα λειτουργικά συστήματα δηλαδή είναι ένα εργαλείο Cross-Platform.
5. Είναι ένα λογισμικό στο όποιο κυκλοφορούν νέες εκδόσεις οι οποίες υποστηρίζουν τις νέες εξελίξεις στο περιβάλλον του MongoDB.
6. Περιέχουν μια ενεργή κοινότητα προγραμματιστών οι οποίοι βοηθούν ώστε να διαδοθεί το πρόγραμμα.

2. Εγκατάσταση MongoDB στο Linux

Ακολουθώντας τις παρακάτω εντολές γίνεται η εγκατάσταση της mongodb στο linux και στη συνέχεια το MongoDB ξεκινά να εκτελείται:

1. *sudo apt install curl*
2. *curl https://www.mongodb.org/static/pgp/server-4.0.asc | sudo apt-key add*
3. *sudo nano /etc/apt/sources.list.d/mongodb-org-4.0.list*
4. *deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/4.0 main*
5. *sudo apt update*

6. *sudo apt-get install mongodb-org*
7. *sudo systemctl enable mongodb*
8. *sudo systemctl start mongodb*
9. *sudo systemctl status mongodb*
10. *mongo --eval 'db.runCommand({ connectionStatus: 1 })'*
11. *sudo systemctl status mongodb*
12. *sudo systemctl stop mongodb*
13. *sudo systemctl start mongodb*
14. *sudo systemctl restart mongodb*
15. *sudo systemctl disable mongodb*
16. *sudo systemctl enable mongodb*

3. MongoDB Backup

Για να εξασφαλίσουμε ότι δεν θα χάσουμε τα δεδομένα μας, κρατάμε ένα backup αυτών από την MongoDB. Για να το πετύχουμε αυτό, θα χρησιμοποιήσουμε την εντολή *mysqldump*.

1. Μέσω μίας εικονκής μηχανής (Virtual Machine), δημιουργούμε μέσω της MongoDB, μία νέα βάση με το όνομα test και μέσα σε αυτή μία collection με όνομα restaurants, όπως φαίνεται στη συνέχεια:

   ```
   > use test
   ```
   Switched to db test
   ```
   > db.restaurants.insert({`name`: `Pizzeria Sammy`})
   WriteResult({“nInserted”: 1})
   ```
2. Στη συνέχεια, φτιάχνουμε ένα χρήστη στην MongoDB, μέσω της εντολής:

   ```
   > db.createUser({user: “root”, pwd: “12345”, roles: [{role: “readWrite”, db: “test”}]})
   ```

3. Έπειτα, στο φάκελο με όνομα *mongobackup*, αποθηκεύω το αρχείο *mongo-backup.sh, δίνοντας όλα τα δικαιώματα.* Δίνω το όνομα του χρήστη που έφτιαξα και το path για το backup. Πιο συγκεκριμένα το αρχείο *mongo-backup.sh* περιέχει*:*

```
DIR=`date +%m%d%y`
DEST=/mongobackup/$DIR
mkdir $DEST
mongodump -h 127.0.0.1 -u root -d test -p 12345 -o $DEST
```

4. Τέλος, τρέχουμε το παραπάνω αρχείο backup και επιβεβαιώνουμε ότι το backup υπάρχει. Οπότε, το backup έχει πέτυχει. Για να καθορίσουμε, κάθε πότε θα γίνεται backup, χρησιμοποιώ την εντολή crontab.