ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"NoSQL Βάσεις Δεδομένων Κειμένων:
Διερεύνηση συστημάτων και εφαρμογή σε θέματα κοινωνικών δικτύων"

Κορδούλη Μαρία : Α.Μ. 2145

Τσίγκρος Γεώργιος : Α.Μ. 2244

ΕΠΙΒΛΕΠΩΝ: Ταμπακάς Βασίλειος

ΠΑΤΡΑ 2023

ΠΕΡΙΛΗΨΗ

Στη σημερινή εποχή με την ανάπτυξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών οι άνθρωποι μπορούν να έχουν πρόσβαση σε εκατομμύρια πληροφορίες που παράγονται συνεχώς. Πώς όμως μπορούν να αναζητήσουν εύκολα και γρήγορα τις πληροφορίες αυτές; Η επιστήμη της Πληροφορικής παρέχει τη δυνατότητα της αποθήκευσης των πληροφοριών σε Βάσεις Δεδομένων μέσω των οποίων μπορούν να εκτελεστούν εντολές αποθήκευσης, ενημέρωσης, αναζήτησης και διαγραφής.

Επειδή όμως όλες οι πληροφορίες που παράγονται δεν έχουν την ίδια μορφή ούτε μπορούν να χρησιμοποιηθούν για τον ίδιο σκοπό από τους χρήστες, έχουν υλοποιηθεί διαφορετικές τεχνολογίες μέσω των οποίων μπορούν να υποστηριχθούν σχεδόν όλες οι απαιτήσεις.

Οι πιο γνωστές τεχνολογίες είναι οι σχεσιακές βάσεις δεδομένων και οι νεότερες NoSQL που υποστηρίζουν λειτουργίες για big data.

Λέξεις-κλειδιά: Βάσεις Δεδομένων, Σχεσιακές, NoSQL, MongoDB

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο 1 θα παρουσιάσουμε το απαραίτητο θεωρητικό υπόβαθρο για την κατανόηση των NoSQL βάσεων δεδομένων. Θα ξεκινήσουμε με μια σύντομη ιστορική αναδρομή στις βάσεις δεδομένων και στη συνέχεια θα παρουσιάσουμε τις δύο μεγάλες κατηγορίες τους τις σχεσιακές και τις NoSQL. Κατόπιν θα προσπαθήσουμε να τις συγκρίνουμε παρουσιάζοντας τα σημεία που υπερτερούν. Θα αναφερθούμε στο Twitter και στη λειτουργεία του Twitter API, στη γλώσσα προγραμματισμού Python και γιατί την επιλέξαμε για την υλοποίηση της εφαρμογής. Το κεφάλαιο θα ολοκληρωθεί με μια αναφορά στην MongoDB.

Στο κεφάλαιο 2 θα αναφερθούμε στα βήματα που ακολουθήσαμε για το σχεδιασμό και την υλοποίηση της εφαρμογής για τον έλεγχο της λειτουργίας της βάσης δεδομένων MongoDB. Θα ξεκινήσουμε με το σχεδιασμό της εφαρμογής, στη συνέχεια θα δούμε τα σημαντικότερα στάδια υλοποίησης και τέλος θα αναφερθούμε στις δυνατότητές της με ενδεικτικές οθόνες λειτουργίας.

Στο κεφάλαιο 3 θα παρουσιάσουμε τα συμπεράσματα που αποκομίσαμε μετά από την ανάπτυξη της εφαρμογής και θα αναφέρουμε προτάσεις για μελλοντικές επεκτάσεις της.

ΚΕΦΑΛΑΙΟ 1. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΙΣΗ

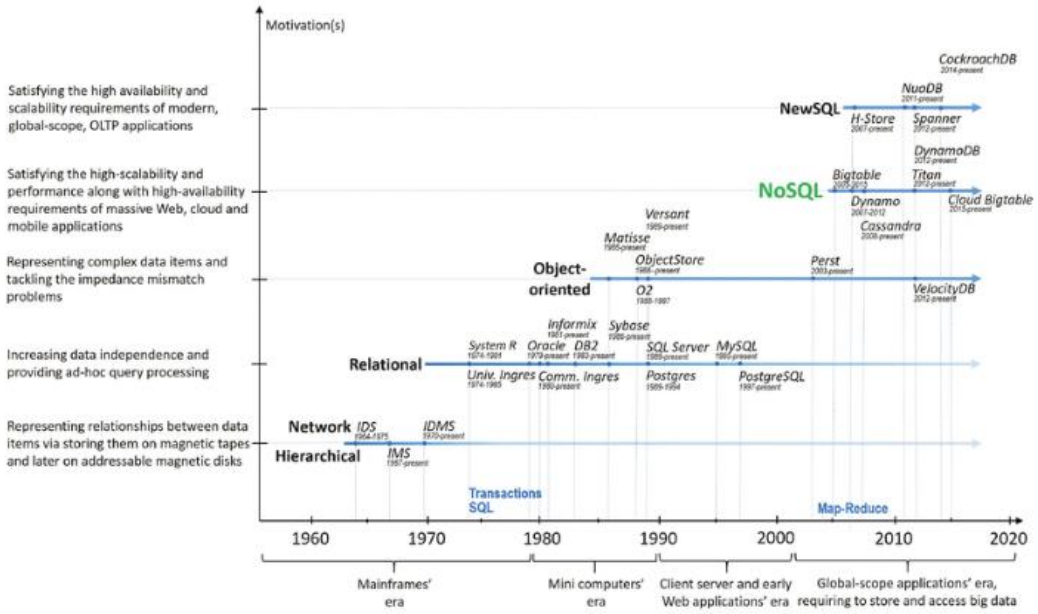
1.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τις απαραίτητες πληροφορίες για την κατανόηση των βάσεων δεδομένων και ιδιαίτερα των απαιτήσεων που ήρθαν να καλύψουν οι NoSQL βάσεις δεδομένων. Θα ξεκινήσουμε με την ιστορική αναδρομή όπου θα παρακολουθήσουμε την εξέλιξή τους από τη δεκαετία του 1960 έως σήμερα. Στη συνέχεια θα αναφερθούμε στις σχεσιακές και στις NoSQL βάσεις δεδομένων, παρουσιάζοντας τα πλεονεκτήματα, μειονεκτήματα αλλά και τις δυνατότητές τους και θα προχωρήσουμε στη μεταξύ τους σύγκριση. Κατόπιν θα αναφερθούμε στο Twitter, στους λόγους που χρησιμοποιείτε ως μέσο έρευνας, στα χαρακτηρίστηκα και στον τρόπο λειτουργίας του Twitter API. Επιπροσθέτως θα κάνουμε μια ιστορική αναδρομή στην γλώσσα προγραμματισμού Python, στα χαρακτηρίστηκα της και στους λόγους που την επιλέξαμε. Τέλος, θα ολοκληρώσουμε το κεφάλαιο με την παρουσίαση της βάσης δεδομένων MonogDB.

1.2 Βάσεις Δεδομένων

1.2.1 Ιστορική Αναδρομή

Οι πρώτες σκέψεις για την αναγκαιότητα της δημιουργίας των βάσεων δεδομένων, έγιναν με τη δημιουργία των πρώτων αρχείων κυρίως σε βιβλιοθήκες, κυβερνητικές υπηρεσίες αλλά και στον ιατρικό κλάδο. Η ανάγκη για τη διατήρηση των αρχείων, την αναζήτηση σε αυτά και την ανάκτηση πληροφοριών ανάγκασε τους επιστήμονες να βρουν τρόπους για την αποθήκευση, ευρετηρίαση και ανάκτηση δεδομένων. Με την εμφάνιση και εξέλιξη των ηλεκτρονικών υπολογιστών αλλά και των Τεχνολογιών Πληροφορικής και Επικοινωνιών, οι διαδικασίες αυτές επιταχύνθηκαν και παρουσιάστηκαν ένα πλήθος από λύσεις και εναλλακτικούς τρόπους προσέγγισης [3]. Στην εξέλιξη των βάσεων δεδομένων από τη δεκαετία του 1960 έως σήμερα μπορούμε να διακρίνουμε διάφορες γενιές όπως ιεραρχική, δικτυακή, σχεσιακή, αντικειμενοστραφή, NoSQL και NewSQL Στην εικόνα που ακολουθεί μπορούμε να δούμε τις γενιές αυτές, τα έτη που εμφανίστηκαν και ορισμένα αντιπροσωπευτικά συστήματα διαχείρισης βάσεων δεδομένων.

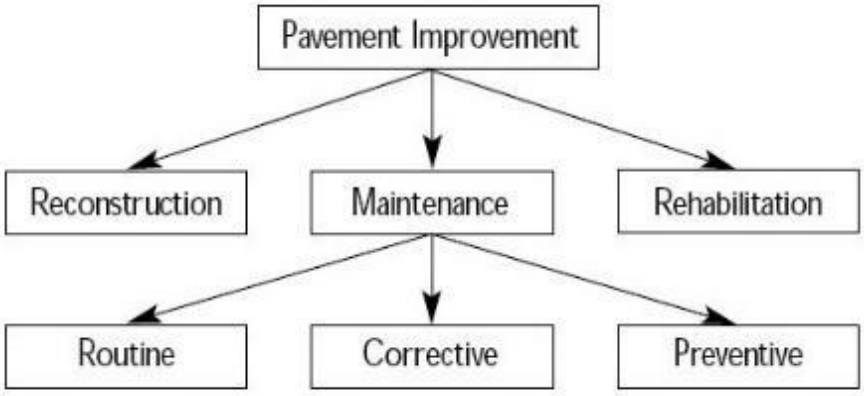


Εικόνα 1 Η εξέλιξη των τεχνολογιών Βάσεων Δεδομένων και ορισμένα αντιπροσωπευτικά συστήματα [10]

Η πρώτη γενιά εμφανίστηκε στα μέσα της δεκαετίας του 1960 μέχρι τις αρχές της δεκαετίας του 1970. Σκοπός των βάσεων δεδομένων που βασίσθηκαν στις αρχές της ήταν η παροχή ενός αποτελεσματικού τρόπου αποθήκευσης και πρόσβασης σε δεδομένα μέσω της δημιουργίας ενός συστήματος διαχείρισης βάσεων δεδομένων γενικής χρήσης. Τα συστήματα αυτά βασίστηκαν στα ιεραρχικά μοντέλα και τα μοντέλα δεδομένων δικτύου και είναι το IDS (Intrusion Detection System), IMS (IP Multimedia Subsystem) και IDMS (Integrated Database Management System). Τα συγκεκριμένα συστήματα για την αποθήκευση των δεδομένων χρησιμοποιούσαν την τεχνική των διασυνδεόμενων αρχείων με αποτέλεσμα η βάση δεδομένων να έχει είτε ιεραρχική δομή είτε δομή δικτύου. Οι διαδικασίες που παρείχαν για την πλοήγηση στα αποθηκευμένα δεδομένα, ήταν διαδικασίες χαμηλού επιπέδου, δηλαδή βασίζονταν στα δεδομένα, στον τύπο τους και στον τρόπο αποθήκευσή τους. Αυτό είχε σαν αποτέλεσμα τα προγράμματα που παρείχαν στους χρήστες πρόσβαση στη βάση δεδομένων να ήταν πολύπλοκα και δύσκολα στην τροποποίησή τους [10].

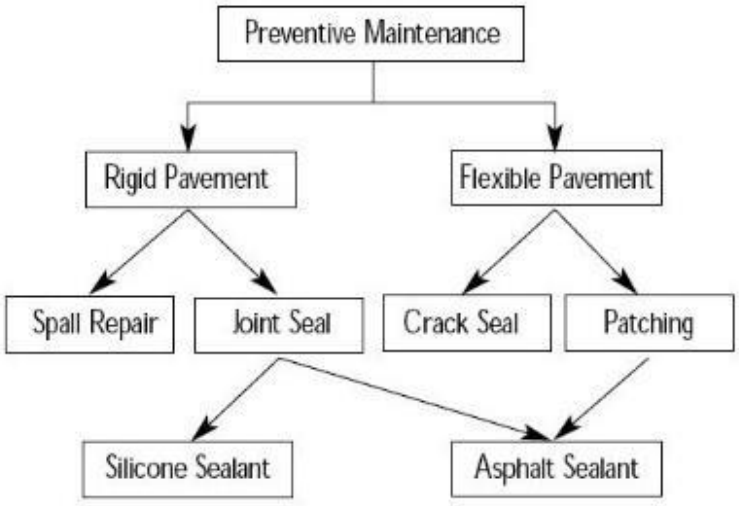
Το ιεραρχικό μοντέλο οργανώνει τα δεδομένα σε μια δομή παρόμοια με ένα δέντρο. Στην κορυφή υπάρχει η κύρια ρίζα, η οποία είναι η αρχή του δέντρου. Τα υπόλοιπα δεδομένα είναι τοποθετημένα κάτω από την κύρια ρίζα, συνδεόμενα με τον κύριο κόμβο ως κόμβοι παιδιών. Το μοντέλο συμβάλλει στην αναπαράσταση της σχέσης «ένα προς πολλά» μεταξύ δύο τύπων δεδομένων. Όπως μπορούμε να δούμε στο παράδειγμα στην εικόνα που ακολουθεί, οι κόμβοι συνδέονται μεταξύ τους

σχηματίζοντας μια σχέση γονέα-παιδιού. Κάθε θυγατρικός κόμβος θα έχει μόνο έναν γονέα.



Εικόνα 2 Ιεραρχικό μοντέλο [18]

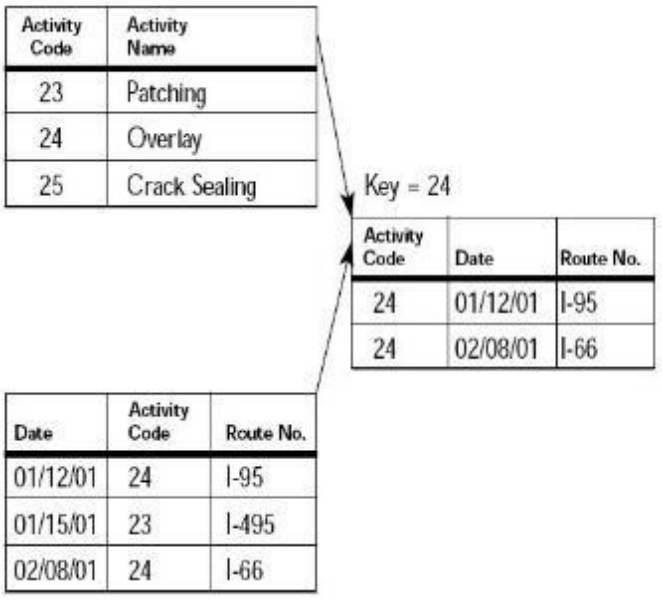
Το μοντέλο δικτύου ήταν μια επέκταση του ιεραρχικού μοντέλου, το οποίο οργανώνει τα δεδομένα σε μια δομή παρόμοια με ένα γράφημα. Σε αντίθεση με το ιεραρχικό μοντέλο, ένας κόμβος στο μοντέλο δικτύου μπορεί να έχει πολλούς γονείς. Επιπλέον, τα δεδομένα στη βάση δεδομένων που δημιουργούνται χρησιμοποιώντας το μοντέλο δικτύου μπορούν να έχουν περισσότερες σχέσεις. Το μοντέλο αυτό είχε δύο πλεονεκτήματα, πρώτον η πρόσβαση στα δεδομένα γινόταν πιο εύκολα και πολύ πιο γρήγορα χάρη του τρόπου σύνδεσής τους και δεύτερον μπορεί να υπάρξει η αναπαράσταση της σχέσης πολλά σε πολλά [18].



Εικόνα 3 Μοντέλο δικτύου [18]

Στις αρχές της δεκαετίας του 1970, είχαμε την εμφάνιση της δεύτερης γενιάς, η οποία βασιζόταν στο μοντέλο σχεσιακών δεδομένων που είχε προτείνει ο Codd. Στο μοντέλο αυτό τα δομημένα δεδομένα αντιπροσωπεύονταν από πλειάδες, οι οποίες ομαδοποιούνταν σε σχέσεις για να μπορέσουν να περιγράψουν τα φυσικά δεδομένα. Με τον τρόπο αυτό, της αποσύνδεσης της λογικής από τη φυσική αναπαράσταση των δεδομένων υπήρχε ένας ουσιαστικός βαθμός ανεξαρτησίας των δεδομένων. Βασιζόμενοι σε αυτή την αποσύνδεση της λογικής από τη φυσική αναπαράσταση οι επιστήμονες του κλάδου της πληροφορικής μπόρεσαν να υλοποιήσουν δηλωτικές γλώσσες ερωτημάτων οι οποίες ήταν προσανατολισμένες στο σύνολο και να καταφέρουν να απαλλάξουν τους προγραμματιστές εφαρμογών από το βάρος της υλοποίησης προγραμμάτων για την πρόσβαση στα δεδομένα. Την ίδια δεκαετία, έκανε την εμφάνισή της η Structured Query Language (SQL) η οποία κυριάρχησε σα γλώσσα για τον καθορισμό, τον χειρισμό και την αναζήτηση δεδομένων. Τα συστήματα αυτής της γενιάς, λόγω της απλότητας και της ευκολίας της χρήσης τους, αντικατέστησαν τα ιεραρχικά και τα δικτυακά συστήματα [10].

Στο σχεσιακό μοντέλο σαν βασική δομή δεδομένων είναι οι πίνακες, οι οποίοι ονομάζονται και σχέσεις. Κάθε πίνακας έχει γραμμές και στήλες. Μια γραμμή αντιπροσωπεύει μια εγγραφή ενώ μια στήλη αντιπροσωπεύει ένα χαρακτηριστικό. Το πλεονέκτημα του συγκεκριμένου μοντέλου είναι ότι με τη βοήθειά του μπορούν να αναπαρασταθούν μία προς πολλές και πολλές προς πολλές σχέσεις [18].

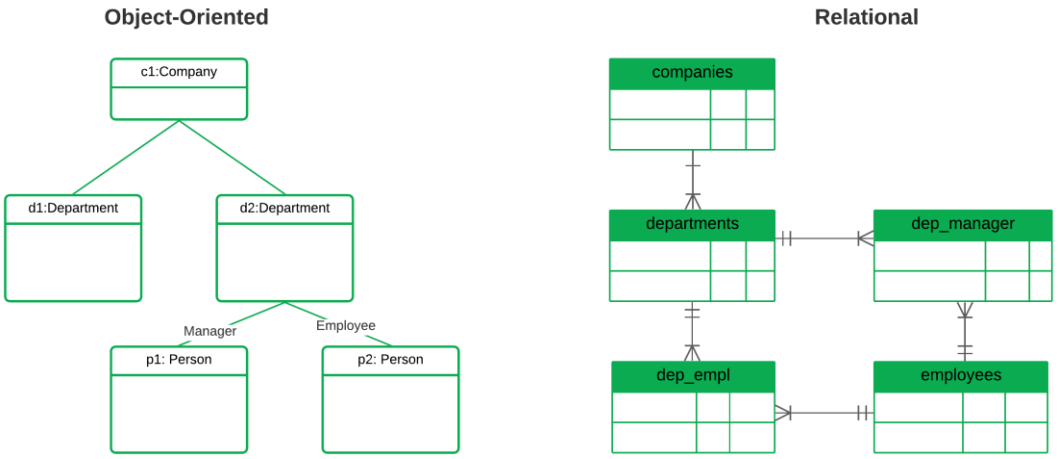


Εικόνα 4 Σχεσιακό μοντέλο [18]

Όσο περνούσαν τα χρόνια, κυριαρχούσαν οι σχεσιακές βάσεις δεδομένων σε όλο και περισσότερους τομείς. Σε ορισμένους τομείς της επιστήμης και των επαγγελματιών παρέμεναν διάφορα προβλήματα. Για παράδειγμα τομείς όπως το Computer Aided Design/Manufacturing (CAD/CAM), Geographic Information Systems (GIS) και πολλές άλλες εφαρμογές που απαιτούσαν πολύπλοκες δομές δεδομένων, διαπίστωναν ότι το σχεσιακό μοντέλο είχε περιορισμένες δυνατότητες μοντελοποίησης. Προβλήματα άρχισαν όμως να αντιμετωπίζουν και οι προγραμματιστές που χρησιμοποιούσαν αντικειμενοστραφείς γλώσσες προγραμματισμού, οι οποίοι αντιμετώπιζαν προβλήματα αντιστοίχισης των πολυεπίπεδων αντικειμένων με τους μονοδιάστατους πίνακες του σχεσιακού μοντέλου.

Τα προβλήματα αυτά οδήγησαν στις αρχές της δεκαετίας του 1980, στην εμφάνιση των αντικειμενοστραφών συστημάτων βάσεων δεδομένων, οι οποίες αποθηκεύουν δεδομένα ως αντικείμενα που προσδιορίζονται από ένα μοναδικό κωδικό (OID). Τα αποθηκευμένα δεδομένα ταξινομούνται σε κλάσεις, τα οποία μπορούν να οργανωθούν ιεραρχικά έτσι ώστε να μπορούν οι κληρονομηθούν οι λειτουργίες των κλάσεων από υποκλάσεις. Όμως τα συστήματα αυτά δεν κατάφεραν να κυριαρχήσουν, γιατί είχαν γίνει πολύ μεγάλες επενδύσεις σε σχεσιακές βάσεις δεδομένων και η μετάβαση απαιτούσε νέες επενδύσεις [10].

Ας δούμε τώρα τη βασική διαφορά ανάμεσα στις σχεσιακές και τις αντικειμενοστραφείς βάσεις δεδομένων. Όπως έχουμε αναφέρει, οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν πίνακες, με κάθε σειρά στον πίνακα να αντιπροσωπεύει μια εγγραφή. Οι στήλες σε μια σειρά αντιπροσωπεύουν τα χαρακτηριστικά μιας μεμονωμένης εγγραφής. Οι συσχετίσεις μεταξύ των εγγραφών ("Μια εταιρεία έχει πολλούς υπαλλήλους. Ένας υπάλληλος ανήκει σε μια εταιρεία") επιτυγχάνονται με ξένα κλειδιά. Σε αντίθεση μια αντικειμενοστραφής βάση δεδομένων αποθηκεύει και διαχειρίζεται αντικείμενα απευθείας στο δίσκο του διακομιστή της βάσης δεδομένων. Δεν υπάρχουν πίνακες, γραμμές, στήλες, ξένα κλειδιά. Υπάρχουν μόνο αντικείμενα. Οι συσχετίσεις μεταξύ αντικειμένων μπορούν να δημιουργηθούν και να διατηρηθούν, το οποίο οδηγεί σε ισχυρή και γρήγορη αναζήτηση δεδομένων σε πολύπλοκες σχέσεις [21].



Εικόνα 5 Σχεσιακές και αντικειμενοστραφείς βάσεις δεδομένων [21]

Στις αρχές της δεκαετίας του 2000, παρουσιάστηκε μια έκρηξη στη χρήση των μέσων κοινωνικής δικτύωσης, των κινητών συσκευών και του Διαδικτύου, με συνέπεια να υπάρξει μια σημαντική αύξηση των δομημένων, ημιδομημένων και μη δομημένων δεδομένων που δημιουργούνταν από τις εφαρμογές. Οι εφαρμογές αυτές έχουν συγκεκριμένες απαιτήσεις από τα συστήματα βάσεων δεδομένων, οι οποίες εστιάζονται στη:

- δυνατότητα οριζόντιας επεκτασιμότητας μέσω της οποίας μπορεί να επιτευχθεί γραμμική προσαρμογή στις τεράστιες ποσότητες δεδομένων και στον αυξανόμενο ρυθμό επεξεργασίας ερωτημάτων χρησιμοποιώντας πρόσθετους πόρους,
- υψηλή διαθεσιμότητα και ανοχή σε σφάλματα για να μπορούν να ανταποκριθούν σε αιτήματα πελατών ακόμη και σε περίπτωση αστοχίας υλικού/λογισμικού ή σε περίπτωση αναβάθμισης,
- αξιοπιστία συναλλαγών για υποστήριξη συνεπών δεδομένων και
- διατηρησιμότητα σχήματος βάσης δεδομένων για μείωση του κόστους εξέλιξης του.

Η επίτευξη των παραπάνω απαιτήσεων από μια σχεσιακή βάση δεδομένων είναι πολύ δύσκολη γιατί πρώτον το σχεσιακό σχήμα καθιστά την επεκτασιμότητα των σχεσιακών βάσεων δεδομένων δαπανηρή λόγω του πολύπλοκου μετασχηματισμού δεδομένων. Δεύτερον, η κλιμάκωση αυτών των συστημάτων απαιτεί τη μετακίνησή τους σε αυτόνομους διακομιστές με πιο βελτιωμένο υλικό, μια διαδικασία δαπανηρή η οποία θέτει το σύστημα εκτός λειτουργίας για μεγάλο χρονικό διάστημα. Η διαδικασία της

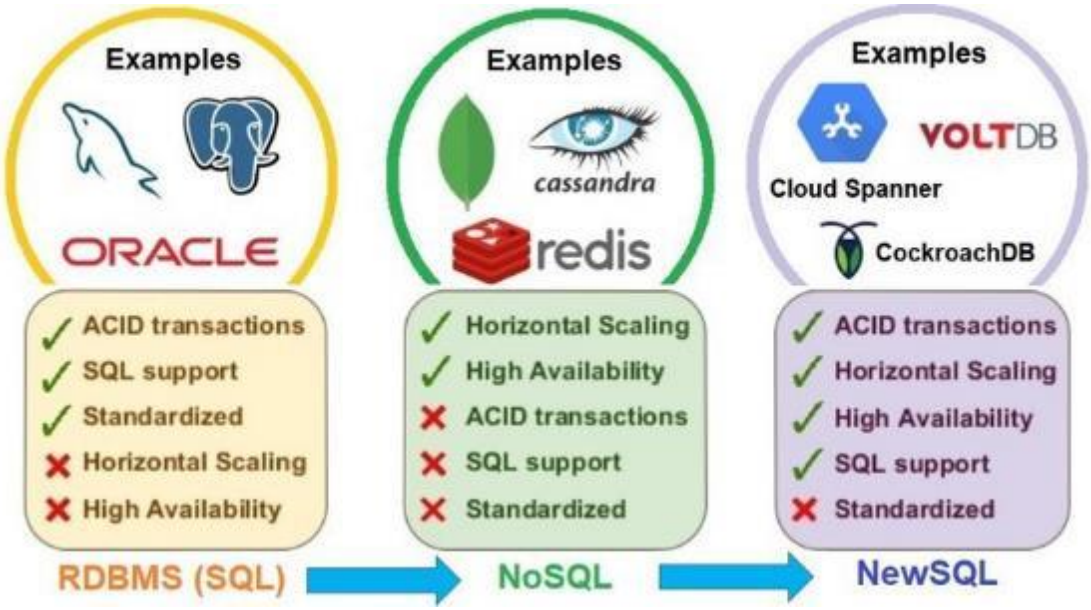
κλιμάκωσης οδηγεί και σε ένα άλλο πρόβλημα εφόσον αυξάνεται η πολυπλοκότητα και η σύνδεση των συστημάτων.

Επειδή είναι αδύνατο να επιτευχθούν όλες οι παραπάνω απαιτήσεις (που αρκετές φορές είναι αμοιβαία αποκλειόμενες), συνήθως επιλέγετε μια ενδιάμεση λύση, η θυσία αυτών που δεν είναι άμεσα απαραίτητες για την ορθή λειτουργεία της εφαρμογής. Αυτή η προσέγγιση οδήγησε στην εμφάνιση της τέταρτης γενιάς βάσεων δεδομένων των NoSQL βάσεων δεδομένων, οι οποίες ικανοποιούν τις απαιτήσεις επεκτασιμότητας και διαθεσιμότητας. Τα βασικά χαρακτηριστικά των NoSQL βάσεων δεδομένων είναι:

- Δεν υποστηρίζουν πλήρως τις ιδιότητες ACID, με αποτέλεσμα να επιτυγχάνεται πιο εύκολα η κλιμάκωση, η υψηλή διαθεσιμότητα και η χαμηλή καθυστέρηση που απαιτούν οι διαδικτυακές εφαρμογές.
- Υιοθετούν πιο ευέλικτα μοντέλα δεδομένων που μπορούν να είναι χωρίς σχήματα και τα δεδομένα μπορεί να ερμηνευτούν στο επίπεδο της εφαρμογής.
- Υπάρχει έξυπνη χρήση κατανεμημένων δεικτών, κατακερματισμού και προσωρινής αποθήκευσης για γρήγορη πρόσβαση και αποθήκευση δεδομένων.
- Τα δεδομένα μπορούν εύκολα να αναπαραχθούν και να χωριστούν οριζόντια σε τοπικούς και απομακρυσμένους διακομιστές.
- Παρέχουν μια φιλική προς τον ιστό πρόσβαση μέσω μιας απλής διεπαφής πελάτη ή πρωτοκόλλου για την αναζήτηση δεδομένων.

Η πέμπτη γενιά εμφανίστηκε στα τέλη της δεκαετίας του 2000 βασιζόμενη σε μια κατηγορία σχεσιακών βάσεων δεδομένων που στοχεύουν στην αντιμετώπιση των υψηλών απαιτήσεων κλιμάκωσης και αξιοπιστίας των σύγχρονων εφαρμογών. Η γενιά αυτή παρέχει μια νέα αρχιτεκτονική μέσω της οποίας η επεκτασιμότητα και η απόδοση βελτιώνεται. Ονομάζονται NewSQL data stores [27], καθώς παρέχουν ορισμένες λειτουργίες της σχεσιακής τεχνολογίας, όπως συναλλαγές πολλαπλών ACID και γλώσσα ερωτήματος SQL.

Οι ομοιότητες και οι διαφορές μεταξύ των σχεσιακών, NoSQL και NewSQL βάσεων δεδομένων, απεικονίζονται στην εικόνα που ακολουθεί:



Εικόνα 6 Ομοιότητες και οι διαφορές μεταξύ των σχεσιακών, NoSQL και NewSQL [24]

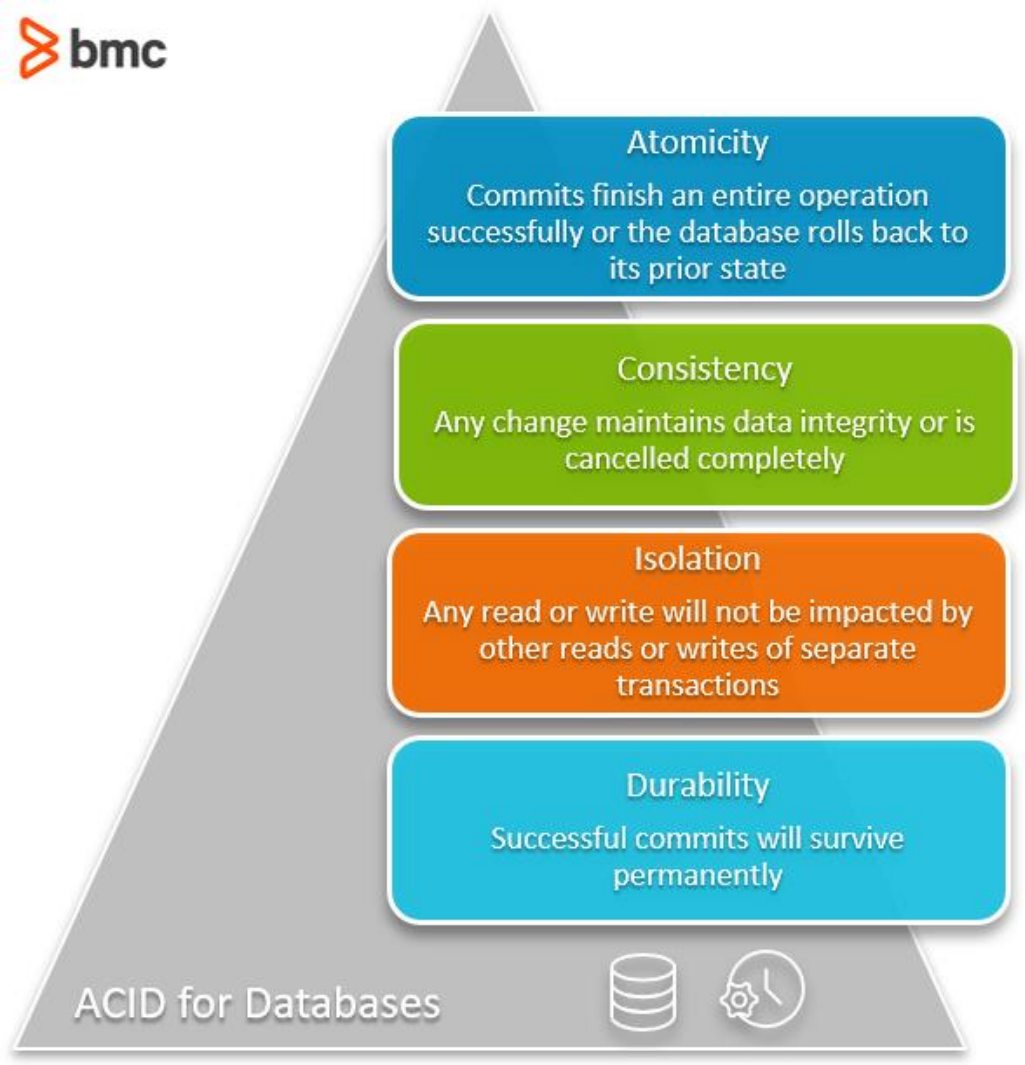
1.2.2 Σχεσιακές βάσεις δεδομένων

Οι σχεσιακές βάσεις δεδομένων προσφέρουν στους χρήστες ένα επίπεδο αφαίρεσης. Βασίζονται στο σχεσιακό μοντέλο, έναν διαισθητικό, τρόπο αναπαράστασης των δεδομένων σε πίνακες και σχέσεις μεταξύ τους. Κάθε γραμμή σε πίνακα είναι μια εγγραφή με ένα μοναδικό αναγνωριστικό που ονομάζεται πρωτεύον κλειδί και χρησιμοποιείται για τον μοναδικό προσδιορισμό της. Οι στήλες του πίνακα απεικονίζουν χαρακτηριστικά των δεδομένων και κάθε εγγραφή έχει συνήθως μια τιμή για κάθε χαρακτηριστικό. Η υλοποίηση των σχέσεων μεταξύ των πινάκων – η διασύνδεσή τους – γίνεται με τη βοήθεια των ξένων κλειδιών.

Το RDBMS έχει πολλά πλεονεκτήματα όπως η αφαίρεση, η πρόσβαση πολλών χρηστών, η αυτόματη βελτιστοποίηση για αναζήτηση, οι ιδιότητες ACID που επιτρέπουν την υποστήριξη συναλλαγών σε εξαιρετικά εύκολη γλώσσα ερωτήσεων κλπ.

Τέσσερις κρίσιμες ιδιότητες καθορίζουν τις συναλλαγές σε μια σχεσιακή βάση δεδομένων: ατομικότητα (Atomicity), συνέπεια (Consistency), απομόνωση (Isolation) και ανθεκτικότητα (Durability) - συνήθως αναφέρονται ως ACID [23]. Συναλλαγή σε μια βάση δεδομένων είναι κάθε πράξη που πραγματοποιείται μέσα σε αυτή, όπως η δημιουργία νέας εγγραφής, η ενημέρωση δεδομένων της ή η διαγραφή της. Οι παραπάνω ιδιότητες διασφαλίζουν την αξιόπιστη επεξεργασία των συναλλαγών στη βάση δεδομένων.

Οι αλλαγές που πραγματοποιούνται σε μια βάση δεδομένων πρέπει να εκτελούνται με προσοχή για να διασφαλιστεί ότι τα δεδομένα που περιέχει δεν καταστρέφονται. Η εφαρμογή των ιδιοτήτων ACID σε κάθε τροποποίηση μιας βάσης δεδομένων είναι ο καλύτερος τρόπος για να διατηρηθεί η ακρίβεια και η αξιοπιστία μιας βάσης δεδομένων.



Εικόνα 7 Οι ιδιότητες ACID [33]

Ας δούμε τώρα αναλυτικά τις ιδιότητες ACID:

- Atomicity: κατά τη διάρκεια εκτέλεσης μιας συναλλαγής, η οποία αποτελείται από πολλά βήματα μπορεί να συμβούν διάφορα λάθη και να μην ολοκληρωθούν όλα τα βήματα. Η εφαρμογή όμως που ζήτησε την εκτέλεση της συναλλαγής, δε θα γνωρίζει ποια βήματα εκτελέστηκαν και ποια όχι. Η ιδιότητα της ατομικότητας αναλαμβάνει τη δέσμευση ότι είτε θα εκτελεστούν όλα τα

βήματα είτε όχι. Αυτό σημαίνει ότι αν προκύψει κάποιο σφάλμα κατά τη διάρκεια της εκτέλεσης η βάση δεδομένων επιστρέφει στην κατάσταση πριν από την έναρξη της εκτέλεσης της συναλλαγής.

- Consistency: Η συνέπεια αναφέρεται στη διατήρηση των περιορισμών ακεραιότητας δεδομένων που έχουν καθοριστεί. Μια συνεπής συναλλαγή δεν παραβιάζει τους περιορισμούς ακεραιότητας που τίθενται στα δεδομένα από τους κανόνες της βάσης δεδομένων. Η επιβολή της συνέπειας διασφαλίζει ότι εάν μια βάση δεδομένων, κατά την εκτέλεση κάποιας συναλλαγής, εισέλθει σε ορθή κατάσταση (εάν προκύψει παραβίαση περιορισμών ακεραιότητας δεδομένων) η συναλλαγή θα ακυρωθεί και η βάση δεδομένων θα επανέλθει στην προηγούμενη κατάσταση. Ένας άλλος τρόπος διασφάλισης συνέπειας σε μια βάση δεδομένων σε κάθε συναλλαγή είναι επίσης η επιβολή δηλωτικών περιορισμών που τίθενται στη βάση δεδομένων. Ένα παράδειγμα δηλωτικού περιορισμού μπορεί να είναι ότι όλοι οι λογαριασμοί πελατών πρέπει να έχουν θετικό υπόλοιπο.
- Isolation: Σε μια βάση δεδομένων μπορεί να ζητηθεί να εκτελεστούν πολλές συναλλαγές ταυτόχρονα. Απομόνωση σημαίνει ότι οι συναλλαγές μπορούν να εκτελούνται παράλληλα έχοντας την ψευδαίσθηση ότι δεν υπάρχει συναλλαγή που εκτελείται την ίδια χρονική στιγμή. Δηλαδή φαίνεται ότι το σύστημα εκτελεί μόνο μία συναλλαγή τη φορά. Καμία άλλη συναλλαγή που εκτελείται παράλληλα δεν έχει ορατότητα στις μη υποβληθείσες τροποποιήσεις της βάσης δεδομένων που πραγματοποιήθηκαν από άλλες συναλλαγές. Για να επιτευχθεί η απομόνωση, είναι απαραίτητος ένας μηχανισμός διασφάλισης. Με τη συγκεκριμένη ιδιότητα μπορούμε να διασφαλίσουμε την ακόλουθη περίπτωση: αν μια πλατφόρμα ηλεκτρονικού εμπορίου έχει από ένα είδος 10 τεμάχια και την ίδια χρονική στιγμή προσπαθούν δύο χρήστες της να αγοράσουν ο ένας 6 τεμάχια και ο άλλος 5, τότε θα εκτελεστεί η μια αγορά από τις δύο και ο άλλος θα ενημερωθεί με το σωστό υπόλοιπο.
- Durability: Η ανθεκτικότητα διασφαλίζει ότι οι αλλαγές που πραγματοποιούνται στη βάση δεδομένων (συναλλαγές) με επιτυχία θα παραμείνουν, ακόμη και σε περίπτωση βλάβης του συστήματος. Η ανθεκτικότητα επιτυγχάνεται με τη χρήση changelogs στα οποία η βάση δεδομένων αναφέρεται κατά την επανεκκίνησή της [33].

1.2.3 NoSQL βάσεις δεδομένων

Με τη συνεχή ανάπτυξη του Διαδικτύου και του υπολογιστικού νέφους, εμφανίστηκαν διάφοροι τύποι εφαρμογών, οι οποίες είχαν μεγαλύτερες απαιτήσεις από τις βάσεις δεδομένων ιδιαίτερα στα ακόλουθα θέματα:

- Ταυτόχρονη ανάγνωση και εγγραφή με μικρή καθυστέρηση: Οι υπάρχουσες βάσεις δεδομένων απαιτούσαν πολλούς πόρους για την κάλυψη των αναγκών για υψηλή ταυτόχρονη ανάγνωση και γραφή με χαμηλή καθυστέρηση. Ένας σημαντικός παράγοντας ικανοποίησης των χρηστών των εφαρμογών ήταν η άμεση απόκρισή τους στα αιτήματα των χρηστών, το οποίο ήταν σε άμεση εξάρτηση με την απόδοση των βάσεων δεδομένων.
- Αποτελεσματική αποθήκευση και πρόσβαση σε μεγάλα δεδομένα: Οι εφαρμογές που χρειάζονταν πρόσβαση σε πολλά δεδομένα, όπως για παράδειγμα οι μηχανές αναζήτησης, απαιτούσαν μια βάση δεδομένων με δυνατότητες αποτελεσματικής αποθήκευσης δεδομένων και να μπορούν να ανταποκριθούν στις ανάγκες πολλών ταυτόχρονων ερωτημάτων.
- Υψηλή επεκτασιμότητα και υψηλή διαθεσιμότητα: Με τον αυξανόμενο αριθμό ταυτόχρονων αιτημάτων και δεδομένων, η βάση δεδομένων πρέπει να είναι σε θέση να υποστηρίζει εύκολη επέκταση και αναβαθμίσεις και να εξασφαλίζει ταχεία αδιάλειπτη εξυπηρέτηση.
- Χαμηλότερο κόστος διαχείρισης και λειτουργίας: Με τη δραματική αύξηση των δεδομένων, το κόστος της βάσης δεδομένων, συμπεριλαμβανομένου του κόστους υλικού, του λογισμικού και του λειτουργικού κόστους, έχει αυξηθεί. Επομένως, είναι απαραίτητο να μειωθεί το κόστος για την αποθήκευση μεγάλων δεδομένων.

Εκείνη τη χρονική στιγμή, στην κορυφή των βάσεων δεδομένων ήταν οι σχεσιακές βάσεις δεδομένων, οι οποίες όμως προσπαθώντας να ανταπεξέλθουν στα παραπάνω ζητήματα διαπιστώθηκε ότι είχαν ορισμένους εγγενείς περιορισμούς:

- Αργή ανάγνωση και γραφή: Μια σχεσιακή βάση δεδομένων έχει μια συγκεκριμένη λογική πολυπλοκότητα. Όσο το μέγεθος των δεδομένων αυξάνεται, εμφανίζονται αδιέξοδα και άλλα ζητήματα ταυτόχρονης πρόσβασης, γεγονός που οδήγησε στην ραγδαία μείωση της αποτελεσματικότητας των διαδικασιών ανάγνωσης και εγγραφής.

- Περιορισμένη χωρητικότητα: Οι υπάρχουσες σχεσιακές βάσεις δεδομένων δεν μπόρεσαν να υποστηρίξουν μεγάλα δεδομένα σε μηχανές αναζήτησης.
- Δύσκολη επέκταση: Ο μηχανισμός συσχέτισης πολλών πινάκων που υπάρχει στη σχεσιακή βάση δεδομένων, έγινε ο κύριος περιοριστικός παράγοντας της επεκτασιμότητας της βάσης δεδομένων [14].

Για την επίλυση πολλών από τις παραπάνω ανάγκες, εμφανίστηκε μια νέα γενιά βάσεων δεδομένων, οι βάσεις δεδομένων «NoSQL». Οι βάσεις δεδομένων NoSQL (αναφέρεται και ως «not only SQL» ή «non-SQL») είναι μια προσέγγιση στο σχεδιασμό βάσεων δεδομένων που επιτρέπει την αποθήκευση και την αναζήτηση δεδομένων εκτός των παραδοσιακών δομών που βρίσκονται σε σχεσιακές βάσεις δεδομένων.

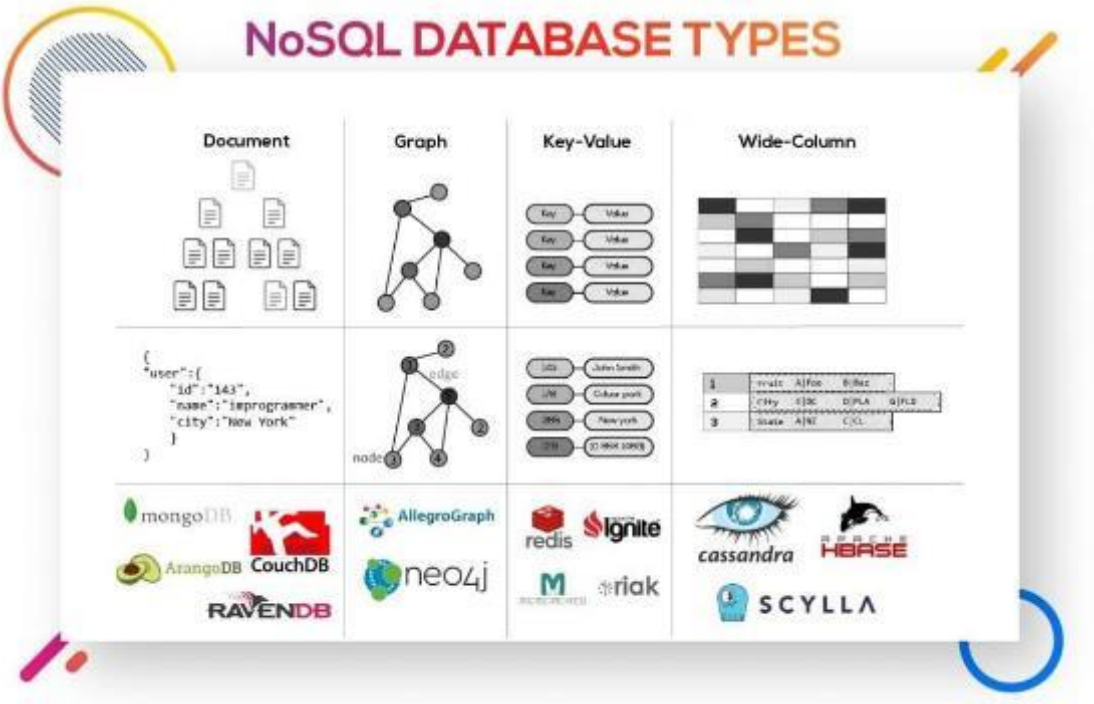
Αντί για τη χρήση πινάκων, μια τυπική δομή μιας σχεσιακής βάσης δεδομένων, οι βάσεις δεδομένων NoSQL, συγκεντρώνουν δεδομένα σε μια δομή δεδομένων, όπως το έγγραφο JSON. Δεδομένου ότι αυτός ο σχεδιασμός μη σχεσιακής βάσης δεδομένων δεν απαιτεί τον καθορισμό κάποιου σχήματος, προσφέρει γρήγορη δυνατότητα διαχείρισης μεγάλων και τυπικά μη δομημένων συνόλων δεδομένων.

Η NoSQL είναι επίσης τύπος κατανεμημένης βάσης δεδομένων, το οποίο σημαίνει ότι οι πληροφορίες αντιγράφονται και αποθηκεύονται σε διάφορους διακομιστές, που μπορεί να είναι απομακρυσμένοι ή τοπικοί. Αυτό διασφαλίζει τη διαθεσιμότητα και την αξιοπιστία των δεδομένων. Εάν συμβεί και κάποιος από τους διακομιστές βγει εκτός σύνδεσης, οπότε τα δεδομένα που έχει είναι και αυτά εκτός σύνδεσης, η υπόλοιπη βάση δεδομένων μπορεί να συνεχίσει να λειτουργεί.

Στις βάσεις δεδομένων NoSQL μπορούμε να διακρίνουμε τους ακόλουθους τύπους:

- Key-value store: θεωρείται συνήθως η απλούστερη μορφή βάσεων δεδομένων NoSQL. Αυτό το μοντέλο δεδομένων χωρίς σχήμα οργανώνεται σε ένα λεξικό το οποίο αποτελείται από ζεύγη κλειδιών-τιμών, όπου κάθε στοιχείο έχει ένα κλειδί και μια τιμή. Το κλειδί θα μπορούσε να μοιάζει με κάτι παρόμοιο που βρίσκεται σε μια βάση δεδομένων SQL, όπως ένα αναγνωριστικό καλαθιού αγορών, ενώ η τιμή είναι μια σειρά δεδομένων, όπως κάθε μεμονωμένο στοιχείο στο καλάθι αγορών αυτού του χρήστη. Συνήθως χρησιμοποιείται για την αποθήκευση και αναζήτηση πληροφοριών περιόδου λειτουργίας χρήστη. Ωστόσο, δεν είναι ιδανικό όταν πρέπει να εκτελούνται αναζητήσεις με πολλές εγγραφές ταυτόχρονα.

- Document store: οι βάσεις δεδομένων εγγράφων αποθηκεύουν δεδομένα ως έγγραφα. Μπορούν να είναι χρήσιμα στη διαχείριση ημι-δομημένων δεδομένων και τα δεδομένα συνήθως αποθηκεύονται σε μορφές JSON, XML ή BSON. Αυτό διατηρεί τα δεδομένα μαζί όταν χρησιμοποιούνται σε εφαρμογές, μειώνοντας τον όγκο της μετάφρασης που απαιτείται για τη χρήση των δεδομένων. Οι προγραμματιστές αποκτούν επίσης μεγαλύτερη ευελιξία, καθώς τα σχήματα δεδομένων δεν χρειάζεται να ταιριάζουν σε όλα τα έγγραφα.
- Graph store: Αυτός ο τύπος βάσης δεδομένων συνήθως περιέχει δεδομένα από ένα γράφημα γνώσεων. Τα στοιχεία δεδομένων αποθηκεύονται ως κόμβοι, ακμές και ιδιότητες. Οποιοδήποτε αντικείμενο, μέρος ή άτομο μπορεί να είναι ένας κόμβος. Ένα άκρο καθορίζει τη σχέση μεταξύ των κόμβων.
- Wide-column store: Οι συγκεκριμένες βάσεις δεδομένων αποθηκεύουν τα δεδομένα σε πίνακες, γραμμές με κύρια διαφορά όμως ότι χρησιμοποιούν δυναμικές στήλες. Με τον όρο δυναμικές στήλες, εννοούμε ότι η κάθε γραμμή δεν έχει τις ίδιες στήλες με τις άλλες γραμμές. Με τον τρόπο αυτό παρέχουν μεγάλη ευελιξία σε σχέση με τις σχεσιακές βάσεις δεδομένων [15].

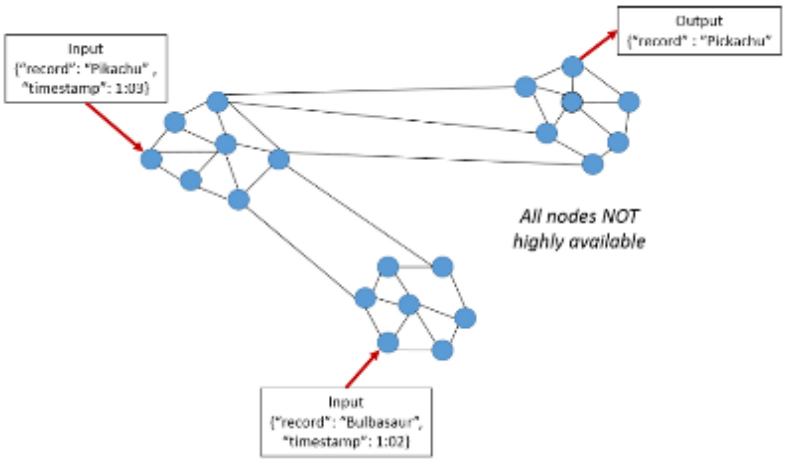


Εικόνα 8 Τύποι βάσεων δεδομένων NoSQL [26]

Πριν ολοκληρώσουμε την παρουσίασή μας στις βάσεις δεδομένων NoSQL θα πρέπει να αναφέρουμε τον τρόπο με το οποίο διαχειρίζονται τις συναλλαγές. Στις σχεσιακές βάσεις, είχαμε αναφέρει ότι ακολουθούν το ACID.

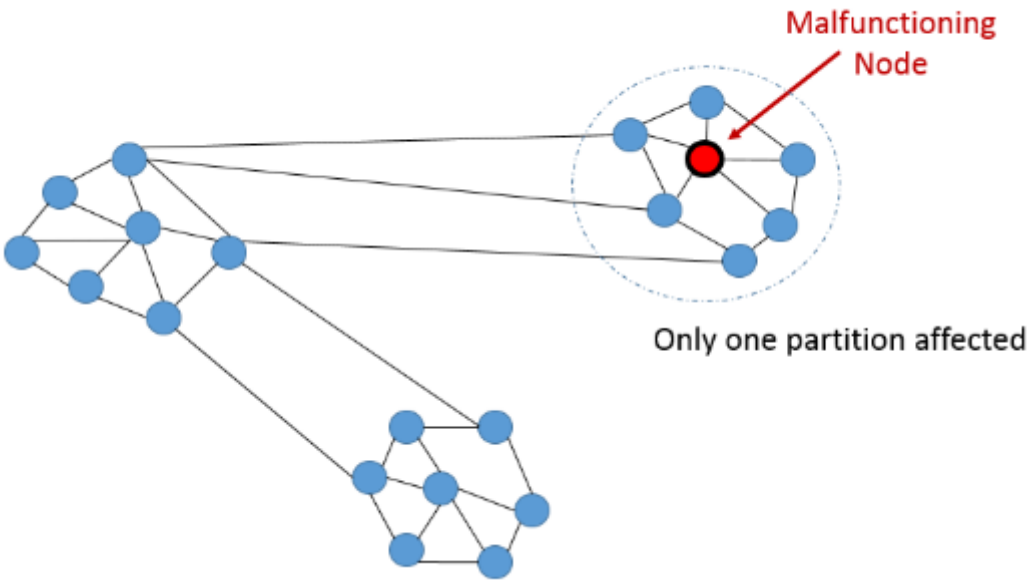
Το 2000, ο καθηγητής Eric Brewer παρουσίασε το θεώρημα CAP. Δηλαδή, Consistency (Συνέπεια), Availability (Διαθεσιμότητα), tolerance of network Partition (ανοχή τμημάτων δικτύου). Το θεώρημα CAP χρησιμοποιείται για να ενημερώσει τους σχεδιαστές συστημάτων για τις αντισταθμίσεις που πρέπει να κάνουν, ενώ σχεδιάζουν δίκτυα κοινόχρηστων δεδομένων. Το θεώρημα CAP δηλώνει ότι σε συστήματα κοινόχρηστων δεδομένων μέσω δικτύου ή κατανεμημένα συστήματα, μπορούμε να επιτύχουμε μόνο δύο από τις τρεις εγγυήσεις για μια βάση δεδομένων [14]. Ας δούμε όμως τί σημαίνει κάθε ένας όρος από το θεώρημα CAP:

- Consistency: σημαίνει ότι όλοι οι πελάτες βλέπουν τα ίδια δεδομένα ταυτόχρονα, ανεξάρτητα από τον κόμβο στον οποίο συνδέονται σε ένα κατανεμημένο σύστημα. Για να επιτευχθεί η συνέπεια, κάθε φορά που τα δεδομένα γράφονται σε έναν κόμβο, πρέπει να προωθούνται αμέσως ή να αναπαράγονται σε όλους τους άλλους κόμβους του συστήματος πριν η εγγραφή θεωρηθεί επιτυχής. Στην εικόνα που ακολουθεί μπορούμε να δούμε ένα παράδειγμα συνέπειας. Βλέπουμε δύο καταχωρήσεις (Input) σε δύο διαφορετικές χρονικές περιόδους (1:02 και 1:03). Αυτή τη στιγμή το σύστημα δεν είναι 100% συνεπές γιατί υπάρχουν ακόμα κόμβοι που έχουν διατηρήσει την παλαιότερη τιμή. Παρόλα αυτά, υπάρχει η γνώση στους κόμβους για το αν έχουν την τελευταία τιμή και όταν ερωτηθούν θα απαντήσουν με τη σωστή τιμή.



Εικόνα 9 Consistency CAP Theorem [22]

- Availability: σημαίνει ότι κάθε μη αποτυχημένος κόμβος επιστρέφει μια απάντηση για όλα τα αιτήματα ανάγνωσης και εγγραφής σε εύλογο χρονικό διάστημα, ακόμη και αν ένας ή περισσότεροι κόμβοι είναι εκτός λειτουργίας. Ένας άλλος τρόπος για να δηλωθεί αυτό είναι όλοι οι λειτουργικοί κόμβοι στο κατανεμημένο σύστημα να επιστρέφουν μια έγκυρη απάντηση για οποιοδήποτε αίτημα, χωρίς αποτυχία ή εξαίρεση.
- Partition Tolerance: σημαίνει ότι το σύστημα συνεχίζει να λειτουργεί παρά την αυθαίρετη απώλεια μηνύματος ή αποτυχία μέρους του συστήματος. Με άλλα λόγια, ακόμη και αν υπάρχει διακοπή δικτύου στο κέντρο δεδομένων και μερικοί από τους υπολογιστές είναι απρόσιτοι, το σύστημα εξακολουθεί να λειτουργεί.



Εικόνα 10 Partiotion tolerance CAP Theorem [22]

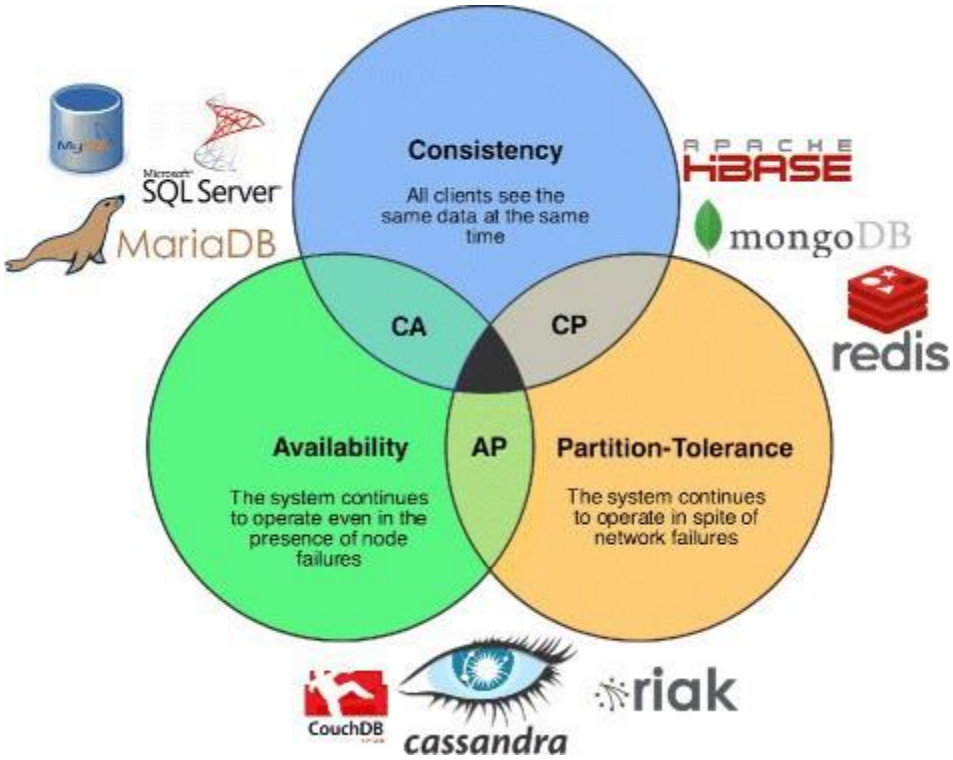
Με τη βοήθεια του θεωρήματος CAP μπορούμε να ταξινομήσουμε τα συστήματα σε τρεις κατηγορίες:

- CP (Consistent and Partition Tolerant) βάση δεδομένων: Μια βάση δεδομένων CP παρέχει συνέπεια και ανοχή κατάτμησης σε βάρος της διαθεσιμότητας. Όταν εμφανίζεται ένα πρόβλημα μεταξύ δύο κόμβων, το σύστημα πρέπει να τερματίσει τον μη συνεπή κόμβο (δηλαδή τον κάνει μη διαθέσιμο) μέχρι να επιλυθεί το πρόβλημα. Το διαμέρισμα αναφέρεται σε διακοπή επικοινωνίας

μεταξύ κόμβων εντός κατανεμημένου συστήματος. Δηλαδή, εάν ένας κόμβος δεν μπορεί να λάβει μηνύματα από έναν άλλο κόμβο στο σύστημα, υπάρχει ένα πρόβλημα μεταξύ των δύο κόμβων, το οποίο μπορεί να οφείλεται σε αποτυχία δικτύου, διακοπή διακομιστή ή οποιονδήποτε άλλο λόγο.

- Βάση δεδομένων AP (Available and Partition Tolerant): Μια βάση δεδομένων AP παρέχει διαθεσιμότητα και ανοχή κατάτμησης σε βάρος της συνέπειας. Όταν εμφανιστεί ένα πρόβλημα, όλοι οι κόμβοι παραμένουν διαθέσιμοι, αλλά αυτοί που βρίσκονται στο άκρο με το πρόβλημα, ενδέχεται να επιστρέψουν σε μια παλαιότερη έκδοση δεδομένων. Όταν επιλυθεί το πρόβλημα, οι βάσεις δεδομένων AP συνήθως επανασυγχρονίζουν τους κόμβους για να επιδιορθώσουν όλες τις ασυνέπειες στο σύστημα.
- Βάση δεδομένων CA (Consistent and Available): Η CA παρέχει συνέπεια και διαθεσιμότητα.

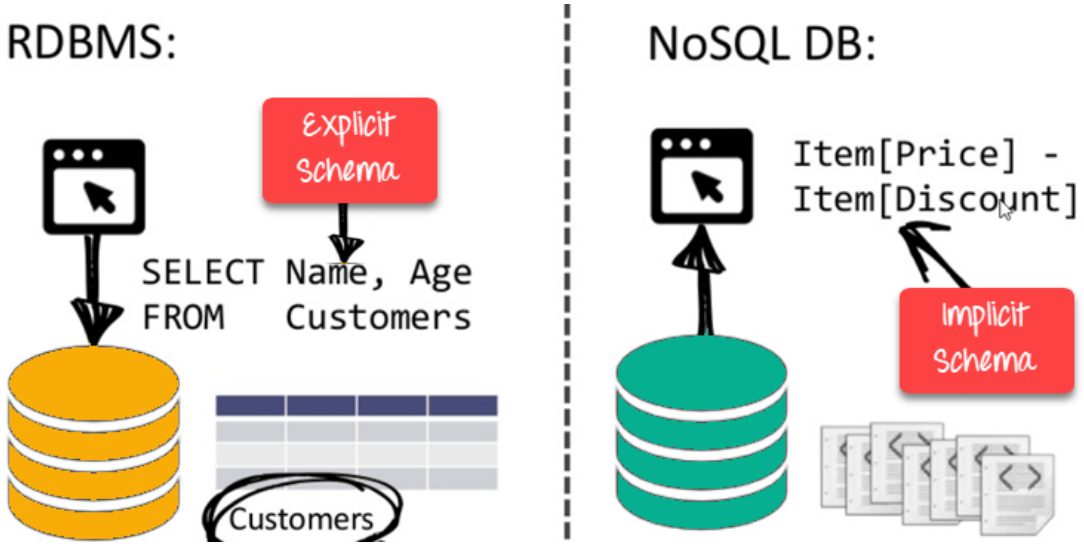
Στην εικόνα που ακολουθεί μπορούμε να δούμε την ταξινόμηση των διαφορετικών βάσεων δεδομένων με βάση το θεώρημα CAP [17].



Εικόνα 11 Ταξινόμηση βάσεων δεδομένων με το θεώρημα CAP [17]

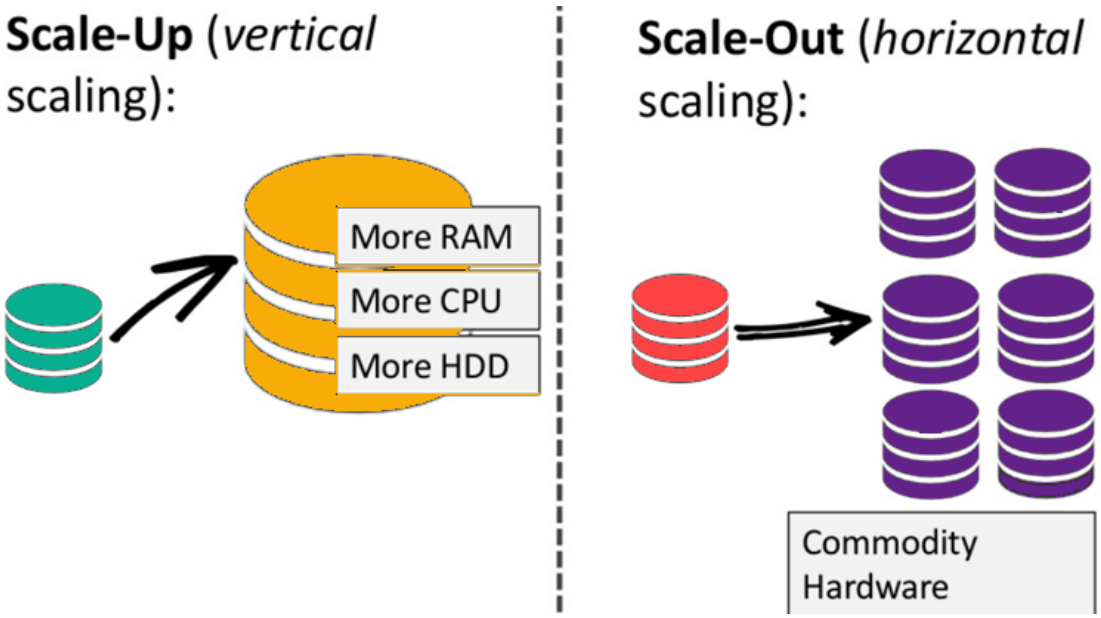
1.3 Σύγκριση σχεσιακών με NoSQL

Έχοντας παρουσιάσει στις προηγούμενες ενότητες τις σχεσιακές και τις NoSQL βάσεις δεδομένων καθώς και τα πλεονεκτήματα και τα μειονεκτήματά τους, στην ενότητα αυτή θα παρουσιάσουμε τις πιο σημαντικές διαφορές που έχουμε αναφέρει.
Αρχικά θα αναφέρουμε ότι οι σχεσιακές βάσεις δεδομένων έχουν ένα σταθερό σχήμα, ενώ οι NoSQL είναι είτε χωρίς σχήμα είτε έχουν ένα πιο χαλαρό σχήμα.



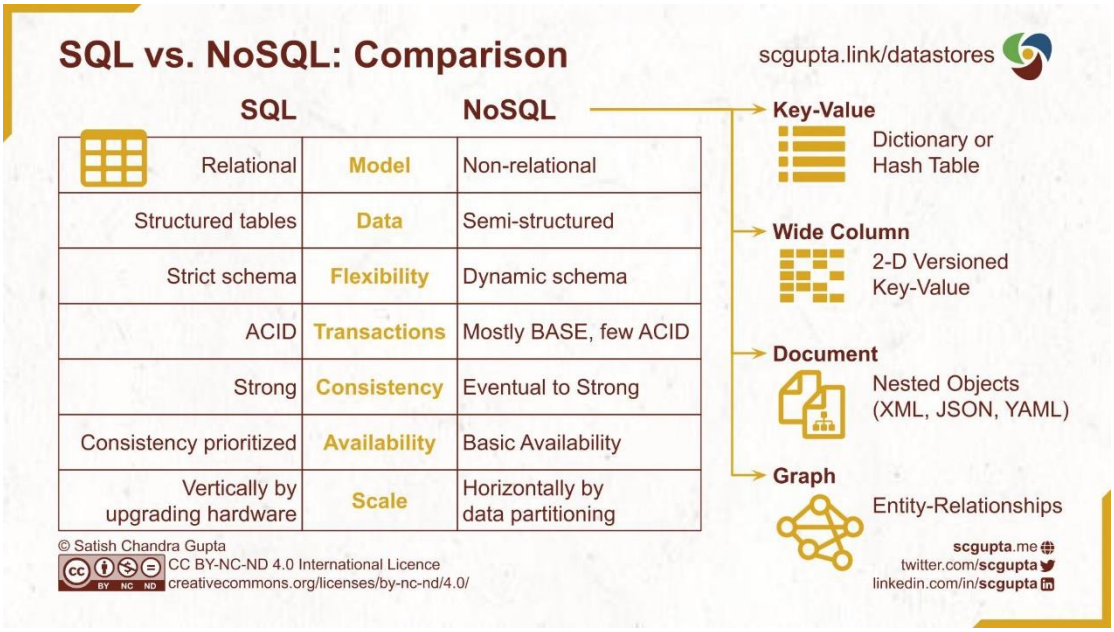
Εικόνα 12 Σχεσιακές – NoSQL σχήμα [30]

Επίσης ως προς τη δυνατότητα επέκτασης οι σχεσιακές βάσεις δεδομένων υποστηρίζουν την Scale Up, την κάθετη δηλαδή επέκταση, όπου θα πρέπει να προστεθεί επιπλέον εξοπλισμός στον εξυπηρετητή, όπως μνήμη, επεξεργαστής και σκληρός δίσκος. Αυτό βέβαια περιορίζει τις δυνατότητες επέκτασης από τις δυνατότητες του εξυπηρετητή.
Αντίθετα οι NoSQL βάσεις δεδομένων παρέχουν τη δυνατότητα για Scale Out, την οριζόντια επέκταση η οποία γίνεται με την προσθήκη νέων εξυπηρετητών – κόμβων στο δίκτυο των NoSQL βάσεων δεδομένων.



Εικόνα 13 Σχεσιακές - NoSQL δυνατότητα επέκτασης [30]

Ως προς τις συναλλαγές θα πρέπει να αναφέρουμε ότι οι σχεσιακές υποστηρίζουν το ACID ενώ οι NoSQL δεν το υποστηρίζουν (η νέα έκδοση της MongoDB παρέχει την υποστήριξη σε ACID συναλλαγές). Όλες οι διαφορές μπορούν να συνοψιστούν στον πίνακα της εικόνας που ακολουθεί [12]:

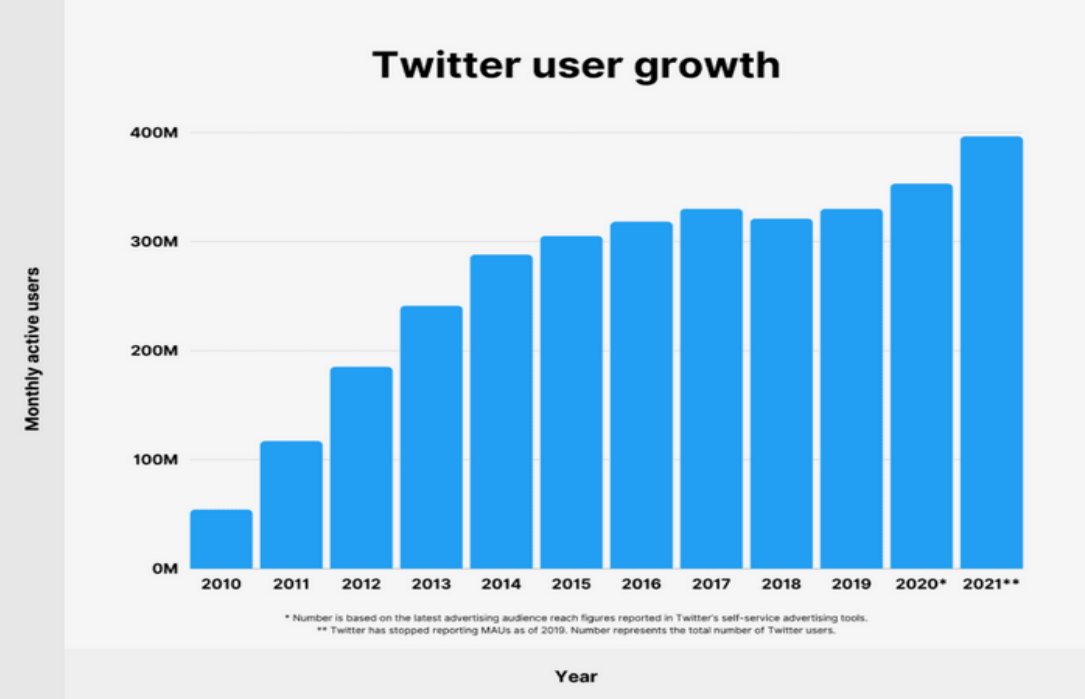


Εικόνα 14 Σύγκριση SQL και NoSQL [12]

1.4 Twitter

1.4.1 Περιγραφή

Το Twitter είναι ένα από τα δημοφιλέστερα μέσα κοινωνικής δικτύωσης το οποίο δημιουργήθηκε στις 21 Μαρτίου του 2006 στο Σαν Φρανσίσκο της Καλιφόρνια από τους Jack Dorsey, Evan Williams και Biz Stone και δημοσιεύθηκε τον Ιούλιο του ίδιου έτους. Η υπηρεσία σήμερα έχει 396 εκατομμύρια ενεργούς χρήστες (2022). Με τόσο μεγάλο κοινό, το Twitter προσελκύει σταθερά τους χρήστες να μεταφέρουν σε αυτό τις απόψεις τους για οποιοδήποτε ζήτημα ενδιαφέροντος.



Εικόνα 15 Ανάπτυξη χρηστών του Twitter [5]

Είναι μια υπηρεσία microblogging (μια μορφή blogging που επιτρέπει να γράφονται σύντομες ενημερώσεις κειμένου) που επιτρέπει στους χρήστες την δημοσίευση και την ανάγνωση μηνυμάτων γνωστά και ως Tweet. Οι συμμετέχοντες χρησιμοποιούν το Twitter για να συνομιλήσουν με άτομα, ομάδες και το ευρύ κοινό. Τα Tweets περιορίζονται σε 140 χαρακτήρες , εμφανίζονται στο προφίλ του χρήστη και στα άτομα που τον ακολουθούν (Followers) [4].
Τα Tweets μπορεί να περιλαμβάνουν σκέψεις του χρήστη, μία είδηση και γενικότερα μια πληροφορία. Εκτός από χαρακτήρες μπορεί να περιλαμβάνουν εικόνα, βίντεο, συνδέσμους και τοποθεσία. Στα Tweets επιπλέον συναντάμε δύο ειδικούς χαρακτήρες(#,@). Το hashtag(#) είναι μια λέξη που ακολουθείτε από το σύμβολο «#» και βοηθάει στην ομαδοποίηση των Tweets με όμοιο θέμα. Το «@» ακολουθούμενο από κάποιο username χρησιμοποιείτε ώστε να απευθυνθούμε σε κάποιον συγκεκριμένο χρήστη. Τέλος υπάρχει η δυνατότητα αναδημοσίευσης των Tweets (retweet) με την

χρήση του «RT». Υπολογίζεται πως κάθε μέρα δημοσιεύονται περισσότερα από 500 εκατομμύρια tweets.

1.4.2 Έρευνα μέσω του Twitter

Για πολλές έρευνες γίνεται λήψη και ανάλυση δεδομένων από το Twitter. Αυτό συμβαίνει για μερικούς από του παρακάτω λόγους:

- Το Twitter διευκολύνει την εύρεση και παρακολούθηση συνομιλιών
- Το Twitter χρησιμοποιεί hashtags που διευκολύνουν τη συλλογή, την ταξινόμηση και την επέκταση των αναζητήσεων κατά τη συλλογή δεδομένων
- Είναι εύκολο να γίνει συλλογή δεδομένων καθώς τα σπουδαία γεγονότα και οι ειδήσεις τείνουν να επικεντρώνονται γύρω από ένα hashtag
- Τα APIs του Twitter είναι ανοιχτά και προσβάσιμα σε σχέση με τα αντίστοιχα που παρέχουν άλλες πλατφόρμες κοινωνικής δικτύωσης. Αυτό καθιστά το Twitter πιο ευνοϊκό για προγραμματιστές που δημιουργούν εργαλεία για πρόσβαση σε δεδομένα.
- Πολλοί ερευνητές κάνουν χρήση του Twitter στην καθημερινότητα τους. Αυτό καθιστά την πλατφόρμα οικία και κάνει πιο άνετη την έρευνα. [32]

1.4.3 Twitter API

Ο όγκος των δεδομένων που μπορούν να δημιουργηθούν από έναν τόσο μεγάλο πληθυσμό είναι αρκετός για να κάνει τους ερευνητές να ενδιαφέρονται για πρόσβαση στο API του Twitter.

Το Twitter παρέχει πρόσβαση σε δεδομένα μέσω της πλατφόρμας Application Programming Interphase (API). Το API είναι μια διεπαφή που γεφυρώνει το χάσμα μεταξύ δύο εφαρμογών, αυτή στο τηλέφωνό του χρήστη και το λογισμικό της εταιρείας παρέχοντας παράλληλα ένα επιπλέον επίπεδο ασφάλειας. Επιτρέπει στους ερευνητές και σε άλλους χρήστες να παρακολουθούν και αναπόφευκτα να συλλέγουν tweets με συγκεκριμένα hashtags ή λέξεις-κλειδιά.

Εικόνα 16 Παράδειγμα λειτουργείας API [19]

Τα APIs ως ερευνητικό εργαλείο προσφέρουν σημαντικές ευκαιρίες για τις μελέτες που είναι είτε ποιοτικές, είτε ποσοτικές. Η πλατφόρμα API επιτρέπει την εύκολη αυτοματοποίηση της συλλογής, ανάλυσης και αποθήκευσης δεδομένων [8].

1.4.4 Τύποι APIs

Τα APIs μπορούν να κατηγοριοποιηθούν με βάση το κοινό και το πεδίο στο οποίο απευθύνονται. Υπάρχουν τέσσερις κύριοι τύποι APIs που χρησιμοποιούν οι προγραμματιστές:

• **Ιδιωτικά APIs** (Private APIs): Αυτά τα API διατίθενται μόνο στην εσωτερική ομάδα μιας εταιρείας, προκειμένου να ενισχυθεί η παραγωγικότητα και η διαφάνεια. Οι προγραμματιστές που εργάζονται για την εταιρεία μπορούν να χρησιμοποιήσουν αυτά τα APIs όπως απαιτείται, αλλά οι προγραμματιστές τρίτων δεν μπορούν.

• **APIs συνεργατών** (Partner APIs): Αυτά τα API κοινοποιούνται εξωτερικά, αλλά μόνο σε όσους έχουν επιχειρηματική σχέση με την εταιρεία που παρέχει το API. Ορισμένες επιχειρήσεις χρησιμοποιούν APIs συνεργατών επειδή θέλουν να ελέγχουν ποιος μπορεί να έχει πρόσβαση στους πόρους τους και πώς χρησιμοποιούνται αυτοί οι πόροι.

• **Ανοιχτά APIs** (Open APIs): Τα ανοιχτά APIs, γνωστά και ως δημόσια APIs, είναι διαθέσιμα για εξωτερική χρήση. Ενώ ορισμένα ανοιχτά APIs είναι δωρεάν, άλλα απαιτούν συνδρομή για χρήση, η οποία συχνά κλιμακώνεται με βάση τη χρήση.

• **Σύνθετα APIs** (Composite APIs): Αυτά τα APIs επιτρέπουν να ομαδοποιούνται κλήσεις ή αιτήματα για να λαμβάνετε μία ενοποιημένη απάντηση από διαφορετικούς διακομιστές. Εάν χρειάζονται δεδομένα από διαφορετικές εφαρμογές, θα χρησιμοποιηθεί ένα σύνθετο API. Εναλλακτικά, αντί να πραγματοποιούνται πέντε ξεχωριστές κλήσεις API διαδοχικά, μπορεί να πραγματοποιηθεί μία χρησιμοποιώντας ένα σύνθετο API [2].

1.4.5 Χαρακτηριστικά του Twitter API

Μερικά από τα σημαντικότερα χαρακτηριστικά του Twitter API:

- Υπάρχουν τέσσερα κύρια αντικείμενα: Tweets, Οντότητες, Μέρη και Χρήστες
- Υπάρχουν καθημερινοί περιορισμοί: Οι κλήσεις και οι αλλαγές στο API περιορίζονται από tokens πρόσβασης για να προστατεύεται η πλατφόρμα από κατάχρηση
- Βασίζεται σε HTTP και όχι σε SSL
- Υπάρχουν συγκεκριμένα μέτρα ώστε να προσαρμόζεται η λειτουργεία API στο κοινωνικό δίκτυο, όπως περιορισμοί βιβλιοθήκης, δημιουργία σελιδοποίησης και συγκεκριμένες παράμετροι.

1.4.6 Λειτουργεία Twitter API

Το Twitter API χρησιμοποιεί τρεις τρόπους αναζήτησης μέσω της εφαρμογής λέξεων-κλειδιών:

- Την αναζήτηση τάσεων, η οποία επιτρέπει στους ερευνητές να παρακολουθούν τη δημοτικότητα των λέξεων-κλειδιών σε ένα συγκεκριμένο χρονικό διάστημα
- Τη βασική αναζήτηση στο Twitter που τείνει να παρακολουθεί άτομα που χρησιμοποιούν τις λέξεις-κλειδιά στα tweets και τα re-tweet τους.
- Την αναζήτηση της επιρροής. Τα δεδομένα από τις αναζητήσεις ταξινομούνται συνήθως σε tweets και re-tweets. Τέτοιες πληροφορίες παρέχονται σε διάφορες μορφές, συμπεριλαμβανομένων γραφημάτων χρονοσειρών.

Η αναζήτηση τάσεων, η βασική αναζήτηση στο Twitter και η αναζήτηση επιρροής αποτελούν τη βάση διαφόρων μοντέλων που έχουν χρησιμοποιήσει οι ερευνητές για να μελετήσουν αναρίθμητες συμπεριφορές και μοτίβα συμβάντων στο Twitter [13].

Το API παρέχει μια λίστα μεθόδων που μπορούν να χρησιμοποιήσουν οι δύο εφαρμογές για να επικοινωνήσουν. Περιλαμβάνουν:

- GET για ανάκτηση δεδομένων.
- POST για τη δημιουργία δεδομένων.
- PUT για ενημέρωση δεδομένων.
- DELETE για κατάργηση δεδομένων.

Τα βήματα για τη χρήση ενός API είναι τα ακόλουθα:

- Επιλογή ενός API: Υπάρχει η δυνατότητα επιλογής ενός API από τα δημοφιλή όπως το Facebook API. Μπορεί επίσης να γίνει επιλογή σύμφωνα με το κόστος ή να επιλεχθεί ένα δωρεάν API
- Απόκτηση κλειδιού API: Ένα κλειδί API χρησιμοποιείται για τον προσδιορισμό ενός έγκυρου πελάτη, τον ορισμό δικαιωμάτων πρόσβασης και την καταγραφή των αλληλεπιδράσεων με το API. Ορισμένα API διαθέτουν ελεύθερα τα κλειδιά τους, ενώ άλλα απαιτούν από τους πελάτες να πληρώσουν. Θα πρέπει να πραγματοποιηθεί εγγραφή στην υπηρεσία και στη συνέχεια εκχωρείτε ένα μοναδικό αναγνωριστικό, το οποίο θα συμπεριληφθεί στις κλήσεις.
- Έλεγχος εγχειριδίου ενεργειών: Γίνεται χρήση του εγχειριδίου για τον τρόπο λήψης του κλειδιού, τον τρόπο αποστολής αιτήσεων και τους πόρους μπορούν να ληφθούν από το διακομιστή
- Εγγραφή αιτήματος σε ένα τελικό σημείο: Γίνεται εγγραφή του πρώτου αιτήματος. Η ευκολότερη μέθοδος είναι να χρησιμοποιηθεί ένας HTTP Client για να βοηθήσει στη δομή και την αποστολή των αιτημάτων.
- Σύνδεση της εφαρμογής: Για να πραγματοποιηθεί σύνδεση της εφαρμογής μπορούν να χρησιμοποιηθούν μία ή περισσότερες γλώσσες όπως Python, Java, JavaScript (και NodeJS), PHP και πολλές άλλες [8].

1.5 Python

1.5.1 Ιστορική Αναδρομή

Η γλώσσα προγραμματισμού Python δημιουργήθηκε από Ολλανδό Γκίντο βαν Ρόσσουμ (Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica το 1989. Ο Γκίντο βαν Ρόσσουμ εμπνεύστηκε την ονομασία από μια σειρά του BBC που ονομαζόταν “Monty’s Python Flying” καθώς ήθελε να έχει ένα σύντομο και μοναδικό όνομα. Θεωρείται διάδοχος της γλώσσας προγραμματισμού ABC, καθώς αυτή υπήρξε η βασική πηγή έμπνευσης για τον Γκίντο βαν Ρόσσουμ [1].

Αρχικά χρησιμοποιήθηκε ως γλώσσα σεναρίων στο κατανεμημένο λειτουργικό σύστημα Amoeba. Η πρώτη έκδοση κυκλοφόρησε το 1991. Η εξέλιξη της Python είναι ταχύτατη μέσω νέων εκδόσεων και με τη χρήση των εργαλείων Python Enhancement Proposals ("PEPs"), τα οποία είναι τυποποιημένα κείμενα που περιέχουν γενικές πληροφορίες, οδηγίες, προτάσεις και περιγραφές για τυχόν νέα χαρακτηριστικά της γλώσσας. Το 2000 κυκλοφόρησε η Python 2.0 και το 2008 η έκδοση 3.0 γνωστή και ως py3k ή python 3000. Η πιο πρόσφατη έκδοση είναι η 3.9.2 και δημοσιεύτηκε το 2021.

Η Python είναι κατά το μεγαλύτερο μέρος η ίδια στις σειρές εκδόσεων, αλλά με διαφορές σε διάφορες λεπτομέρειες, κυρίως στη λειτουργία των ενσωματωμένων αντικειμένων όπως είναι τα λεξικά και οι συμβολοσειρές [35].

Η Python είναι αντικειμενοστραφής, ερμηνευμένη και διαδραστική γλώσσα προγραμματισμού. Παρέχει δομές δεδομένων υψηλού επιπέδου όπως λίστα, πλειάδες, σύνολα, πίνακες συσχετίσεων, δυναμική πληκτρολόγηση και σύνδεση, λειτουργικές μονάδες,, εξαιρέσεις, αυτόματη διαχείριση μνήμης κ.λπ.

Το μοντέλο της Python είναι ευέλικτα επεκτάσιμο, παρέχει ενσωματωμένα στοιχεία (εντολές, τύπους αντικειμένων, κ.λπ.) και επιτρέπει στους προγραμματιστές να προσθέτουν τα δικά τους στοιχεία ανάλογα με τις ανάγκες τους και το σύστημα που χρησιμοποιούν. Πολλές μεγάλες εταιρείες όπως NASA, Google, YouTube, BitTorrent κ.α. χρησιμοποιούν την γλώσσα προγραμματισμού Python [1].

Εικόνα 17 Το λογότυπο της Python [25]

1.5.2 Χαρακτηριστικά της Python

- Απλή: Η Python είναι μια απλή και μινιμαλιστική γλώσσα. Η ψευδο-κωδικοποιημένη φύση της Python είναι ένα από τα μεγαλύτερα πλεονεκτήματά της. Επιτρέπει να επικεντρωθείτε στη λύση του προβλήματος και όχι στην ίδια τη γλώσσα.
- Εύκολη στην εκμάθηση: Η Python έχει απλή σύνταξη που την καθιστά εύκολη στην εκμάθηση.
- Ελεύθερος και Ανοιχτός Κώδικας: Μπορείτε να διανείμετε ελεύθερα αντίγραφα του λογισμικού, να διαβάσετε τον πηγαίο κώδικα του, να κάνετε αλλαγές σε αυτό, να χρησιμοποιήσετε κομμάτια του σε νέα δωρεάν προγράμματα. Αυτός είναι ένας από τους λογούς που καθιστά την Python τόσο καλή.
- Γλώσσα υψηλού επιπέδου: Όταν γράφετε προγράμματα στην Python, δεν χρειάζεται ποτέ να ασχοληθείτε με τις λεπτομέρειες χαμηλού επιπέδου, όπως η διαχείριση της μνήμης που χρησιμοποιείται από το πρόγραμμά σας κ.λπ.
- Φορητότητα: Λόγω της φύσης ανοιχτού κώδικα, η Python αλλάζει για να λειτουργεί σε πολλές πλατφόρμες. Όλα τα προγράμματα Python μπορούν να λειτουργήσουν σε οποιαδήποτε πλατφόρμα χωρίς να απαιτούνται αλλαγές

αρκεί να είστε προσεκτικοί για να αποφύγετε τυχόν λειτουργίες που εξαρτώνται από το σύστημα. Μπορείτε να χρησιμοποιήσετε την Python σε Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, 1 Windows CE και PocketPC.

- Ερμηνεία: Ένα πρόγραμμα γραμμένο σε μια μεταγλωττισμένη γλώσσα όπως η C ή η C++ μετατρέπεται από τη γλώσσα-πηγή σε μια γλώσσα που ομιλείτε από τον υπολογιστή σας χρησιμοποιώντας έναν μεταγλωττιστή. Όταν εκτελείτε το πρόγραμμα, το λογισμικό σύνδεσης/φόρτωσης αντιγράφει το πρόγραμμα από το σκληρό δίσκο στη μνήμη και αρχίζει να το εκτελεί. Η Python, δεν χρειάζεται μεταγλώττιση. Απλά εκτελείτε το πρόγραμμα απευθείας από τον πηγαίο κώδικα. Εσωτερικά, η Python μετατρέπει τον πηγαίο κώδικα σε μια ενδιάμεση μορφή που ονομάζεται bytecodes , στη συνέχεια το μεταφράζει στη μητρική γλώσσα του υπολογιστή και το εκτελεί.
- Αντικειμενοστρεφής: Η Python υποστηρίζει προγραμματισμό προσανατολισμένο στη διαδικασία καθώς και αντικειμενοστρεφή προγραμματισμό. Σε γλώσσες προσανατολισμένες στη διαδικασία, το πρόγραμμα βασίζεται σε διαδικασίες ή συναρτήσεις που δεν είναι παρά επαναχρησιμοποιήσιμα κομμάτια προγραμμάτων. Στις αντικειμενοστρεφείς γλώσσες, το πρόγραμμα είναι χτισμένο γύρω από αντικείμενα που συνδυάζουν δεδομένα και λειτουργικότητα. Η Python έχει έναν πολύ ισχυρό αλλά απλοϊκό τρόπο να κάνει OOP(Object-oriented programming), ειδικά σε σύγκριση με μεγάλες γλώσσες όπως η C ++ ή η Java.
- Επεκτασιμότητα: Εάν χρειάζεστε ένα κρίσιμο κομμάτι κώδικα να τρέξει πολύ γρήγορα ή θέλετε να έχετε κάποιο κομμάτι αλγορίθμου το οποίο να μην είναι ανοιχτό, μπορείτε να κωδικοποιήσετε αυτό το μέρος του προγράμματός σας σε C ή C ++ και μετά να το χρησιμοποιήσετε από το πρόγραμμα Python.
- Δυνατότητα ενσωμάτωσης: Μπορείτε να ενσωματώσετε την Python στα προγράμματά C/C++ για να δώσετε δυνατότητες «scripting» για τους χρήστες του προγράμματός.
- Εκτεταμένες βιβλιοθήκες: Η πρότυπη βιβλιοθήκη Python είναι πολύ μεγάλη. Αυτό βοηθάει ώστε να υπάρχουν πολλές δυνατότητες που περιλαμβάνουν κανονικές εκφράσεις, δημιουργία τεκμηρίωσης, δοκιμές μονάδων, βάσεις δεδομένων, προγράμματα περιήγησης ιστού, CGI, ftp, email, XML, XML-RPC, HTML, αρχεία WAV, κρυπτογραφία, GUI (γραφικές διεπαφές χρήστη) και άλλα πράγματα που εξαρτώνται από το σύστημα [29].

1.5.3 Γιατί Python;

Η Python είναι μια κορυφαία γλώσσα προγραμματισμού για την επιστήμη των δεδομένων, και η MongoDB με το ευέλικτο και δυναμικό σχήμα της, είναι ένας εξαιρετικός συνδυασμός για τη δημιουργία σύγχρονων εφαρμογών ιστού, JSON API, επεξεργαστών δεδομένων και άλλων εφαρμογών. Η MongoDB έχει εγγενές

πρόγραμμα οδήγησης Python και μια ομάδα μηχανικών αφοσιωμένων στο να διασφαλίσει ότι η MongoDB και η Python συνεργάζονται άψογα. Η PyMongo, η τυπική βιβλιοθήκη προγραμμάτων οδήγησης MongoDB για την Python, είναι εύκολη στη χρήση και προσφέρει ένα API για πρόσβαση σε βάσεις δεδομένων, συλλογές και έγγραφα. Τα αντικείμενα που ανακτώνται από τη MongoDB μέσω της PyMongo είναι συμβατά με λεξικά και λίστες, ώστε να μπορούμε εύκολα να τα χειριζόμαστε, να τα αναπαράγουμε και να τα εκτυπώνουμε [20].

1.6 MongoDB

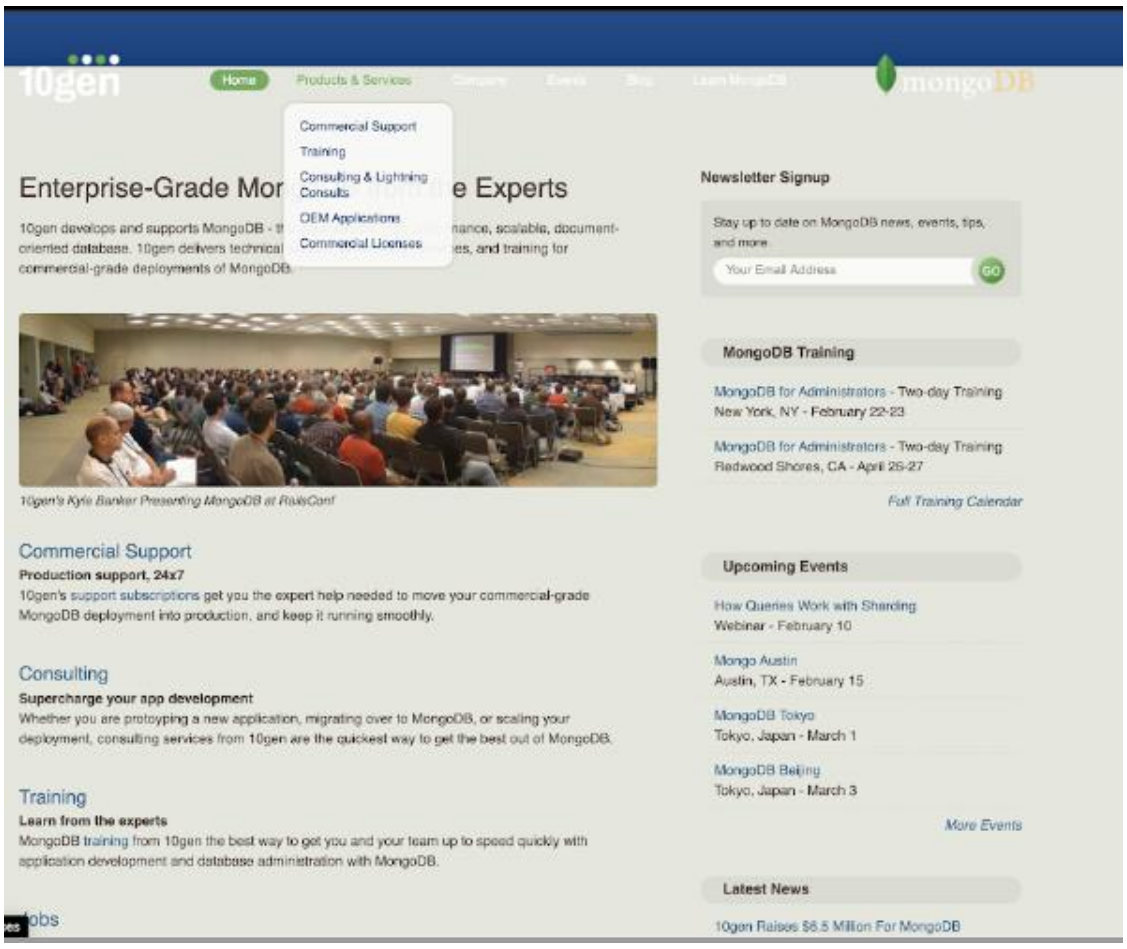
1.6.1 Ιστορική Αναδρομή

Η MongoDB είναι μια βάση δεδομένων που ανήκει στην κατηγορία NoSQL και αναπτύχθηκε στη γλώσσα προγραμματισμού C++. Η λέξη Mongo προέρχεται από τον όρο Humongous, που σημαίνει εξαιρετικά μεγάλο ή τεράστιο.

Από το 2007, που εμφανίστηκε στην αγορά, η MongoDB έχει βελτιώσει την απόδοσή της σημαντικά, επειδή καθιστά την αποθήκευση δεδομένων γρηγορότερη και ευκολότερη χρησιμοποιώντας δυναμικά σχήματα βάσεων δεδομένων παρόμοια με το JSON αντί για παραδοσιακά συστήματα σχεσιακών βάσεων δεδομένων πινάκων και SQL.

Όπως αναφέραμε η MongoDB εμφανίστηκε το 2007. Η ανάπτυξή της ξεκίνησε από τον οργανισμό 10gen (που είχε έδρα τη Νέα Υόρκη), ο οποίος τώρα ονομάζεται MongoDB Inc. Οι προγραμματιστές που ξεκίνησαν την προσπάθεια ήταν ο Kevin P. Ryan, Dwight Merriman και ο Eliot Horowitz, οι ιδρυτές του DoubleClick [7]. Οι τρεις προγραμματιστές, ξεκίνησαν την προσπάθεια για την ανάπτυξη της MongoDB γιατί αντιμετώπιζαν σοβαρά προβλήματα κλιμάκωσης της σχεσιακής βάσης δεδομένων κατά την ανάπτυξη εφαρμογών διαδικτύου στην εταιρεία τους. Την εποχή εκείνη είχε διαδοθεί από εμπορικές σχεσιακές βάσεις δεδομένων όπως η Oracle, η κάθετη κλιμάκωση όπου οι πόροι συγκεντρώνονταν μόνο σε έναν κεντρικό υπολογιστή που εξυπηρετούσε τη βάση δεδομένων για να επιτύχουν μεγάλο εύρος ζώνης. Με τον τρόπο αυτό, δημιουργούσαν ένα μοναδικό σημείο το οποίο αν έβγαινε εκτός λειτουργίας, τότε ολόκληρη η εφαρμογή θα έβγαινε και αυτή εκτός λειτουργίας. Ο αρχικός τους σκοπός ήταν η υλοποίηση μιας βάσης δεδομένων που θα μπορεί να επεξεργάζεται δεδομένα σε

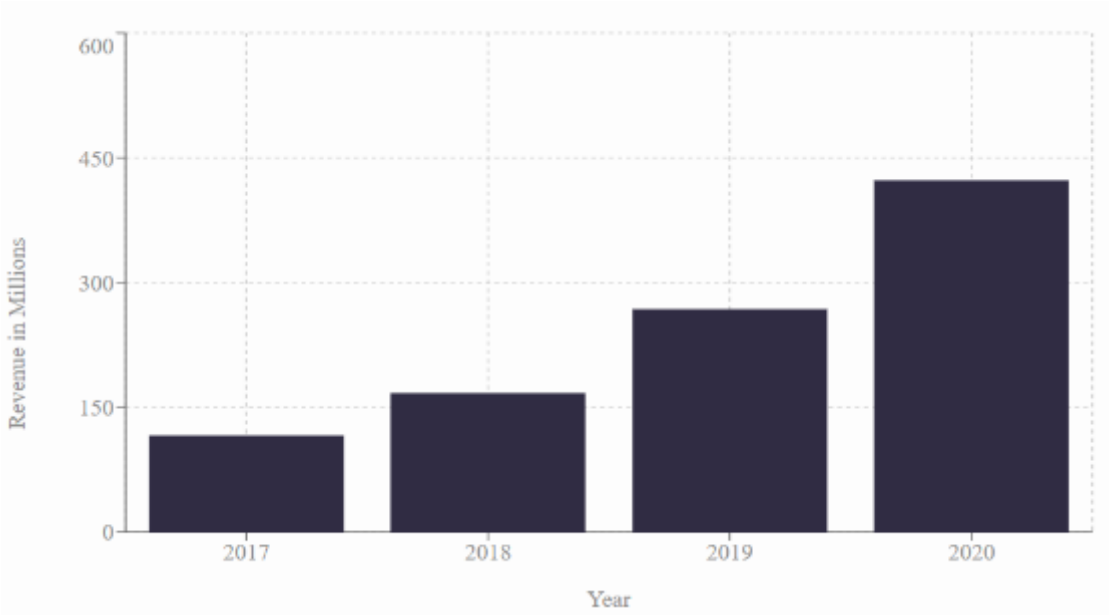
πολύ μεγάλες ποσότητες και θα μπορούσε να κλιμακωθεί οριζόντια [11]. Για την υποστήριξη της προσπάθειάς τους, έλαβαν χρηματοδότηση αξίας 81 εκατομμυρίων δολαρίων, από τις εταιρείες Intel Capital, Red Hat, New Enterprise Associates (NEA), Union Square Ventures, Flybridge Capital Partners, Sequoia Capital και In-Q-Tel [7]. Το 2008 παρουσίασαν τη MongoDB, μια βάση δεδομένων ανοιχτού κώδικα, την οποία συνέχισαν να εξελίσσουν και να βελτιώνουν. Οι υπηρεσίες που προσέφεραν κυρίως ήταν υποστήριξη για την συνεχή και ορθή λειτουργία των βάσεων δεδομένων MongoDB, συμβουλευτική για την υλοποίηση νέων εφαρμογών ή τη μεταφορά παλαιών από άλλες βάσεις δεδομένων σε MongoDB και εκπαίδευση.



Εικόνα 18 Προσφορά υπηρεσιών από την 10gen για τη MongoDB [11]

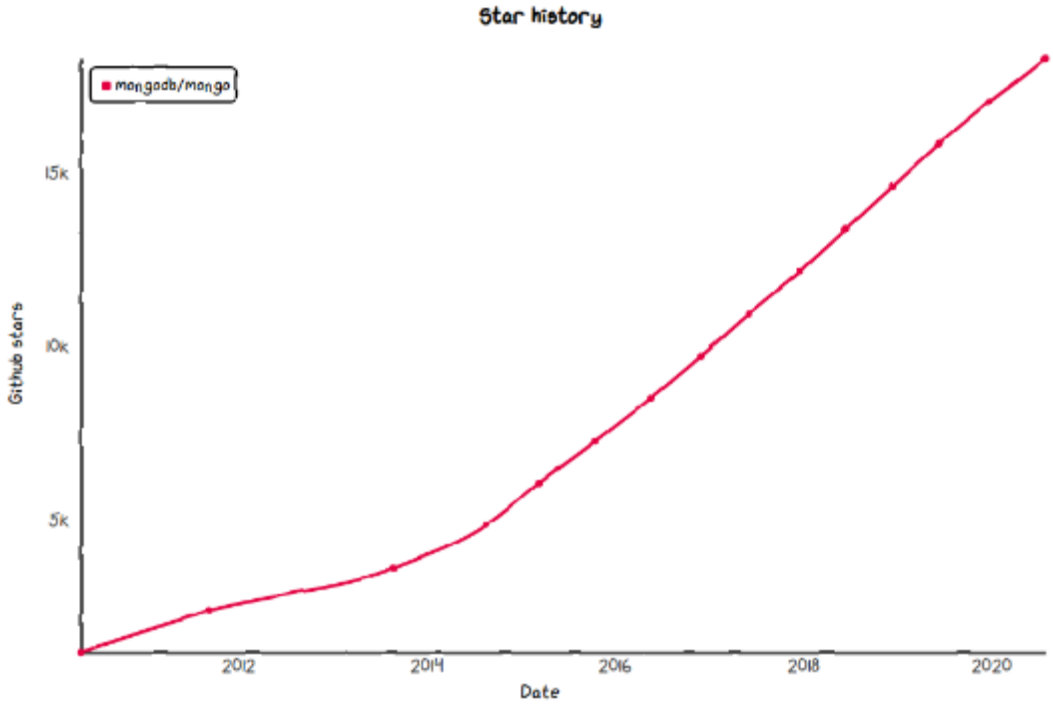
Μέχρι το 2013 είχαν προστεθεί και νέες υπηρεσίες, όπως υπηρεσίες monitoring, backup και μια ειδική διανομή για επιχειρήσεις με εξειδικευμένες λειτουργίες. Την ίδια χρονιά μετονομάστηκε και η εταιρεία από 10gen σε MongoDB Inc.

Το 2015, η εταιρεία άλλαξε τον τρόπο που προσέφερε τα προϊόντα της, ακολουθώντας παρόμοια λογική με τις άλλες εταιρείες. Προσέφερε δύο προϊόντα, το MongoDB Enterprise Advanced για χρήση και λειτουργία σε κεντρικούς υπολογιστές της κάθε εταιρείας και το MMS που ήταν ένα cloud service για τη MongoDB. Το 2016 προστέθηκε επιπλέον ένα προϊόν το MongoDB Professional για το οποίο παρείχαν υποστήριξη μέσα σε δύο ώρες από την αναφορά σφάλματος όλες τις μέρες του έτους. Το ίδιο έτος παρουσιάστηκε και η έκδοση MongoDB Atlas, ο απλούστερος, ισχυρότερος και οικονομικότερος τρόπος εκτέλεσης της MongoDB στο Cloud.
Το 2017 εισήχθη στο χρηματιστήριο και από τότε παρουσιάζει συνεχώς αυξανόμενα έσοδα.



Εικόνα 19 Έσοδα της εταιρείας MongoDB Inc. [11]

Παράλληλα με την αύξηση των εσόδων, παρατηρήθηκε και αύξηση χρήσης της βάσης δεδομένων MongoDB σε εφαρμογές. Στο ακόλουθο γράφημα μπορούμε να δούμε την αύξηση των εφαρμογών που κάνουν χρήση της MongoDB σαν βάση δεδομένων και πώς επιβραβεύονται από τους χρήστες με τα Github stars.



Εικόνα 20 Εξέλιξη των Github stars σε εφαρμογές με MongoDB [11]

1.6.2 Δυνατότητες/Χαρακτηριστικά

Η MongoDB είναι μια βάση δεδομένων NoSQL που αποθηκεύει τα δεδομένα με τη μορφή ζεύγους κλειδιών-τιμών. Είναι μια βάση δεδομένων ανοιχτού κώδικα, εγγράφων που παρέχει υψηλή απόδοση και επεκτασιμότητα μαζί με τη μοντελοποίηση δεδομένων και τη διαχείριση δεδομένων τεράστιων συνόλων δεδομένων. Επίσης η MongoDB παρέχει τη δυνατότητα αυτόματης κλιμάκωσης?????? και μπορεί να εγκατασταθεί σε πολλές πλατφόρμες λειτουργικών συστημάτων όπως Windows, Linux κ.λπ.

Αναφέραμε ότι η MongoDB είναι μια βάση δεδομένων εγγράφων, τί είναι όμως ένα έγγραφο; Ένα έγγραφο είναι μια δομή δεδομένων με ζεύγη ονόματος-τιμής όπως στη JSON. Είναι πολύ εύκολο να χαρτογραφηθεί (αντιστοιχηθεί) οποιοδήποτε αντικείμενο οποιασδήποτε γλώσσας προγραμματισμού με ένα έγγραφο MongoDB. Για παράδειγμα ένα αντικείμενο μαθητής έχει χαρακτηριστικά γνωρίσματα όνομα, rollno και μαθήματα, όπου τα μαθήματα είναι μια λίστα.

```
{
        name : "Stduytonight",
        rollno : 1,
        subjects : ["C Language", "C++", "Core Java"]
}
```

Η MongoDB αποτελείται από ένα σύνολο βάσεων δεδομένων. Κάθε βάση δεδομένων αποτελείται από Συλλογές (Collections). Τα δεδομένα στο MongoDB αποθηκεύονται σε συλλογές. Το παρακάτω σχήμα απεικονίζει την τυπική δομή βάσης δεδομένων στο MongoDB. Μια βάση δεδομένων είναι ένα σύνολο που μπορεί να περιέχει συλλογές (collection).

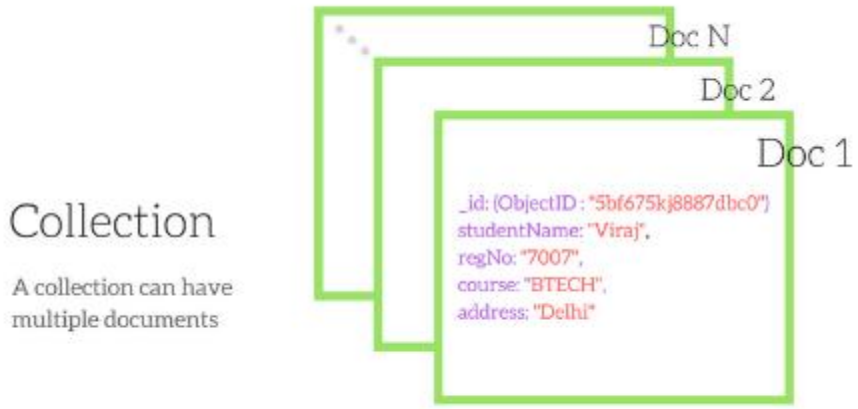
Εικόνα 21 Δομή της βάσης δεδομένων MongoDB [28]

Η συλλογή είναι ένα σύνολο εγγράφων MongoDB. Τα έγγραφα που ανήκουν στις συλλογές είναι ισοδύναμα με μια σειρά δεδομένων σε πίνακα σε μια σχεσιακή βάση δεδομένων. Όμως, οι συλλογές στη MongoDB δεν σχετίζονται με κανένα καθορισμένο σχήμα σε σύγκριση με το RDBMS. Οι συλλογές είναι ένας τρόπος αποθήκευσης σχετικών δεδομένων. Όντας χωρίς σχήμα, οποιοσδήποτε τύπος εγγράφου μπορεί να αποθηκευτεί σε μια συλλογή, αν και συνιστάται ομοιότητα για την αποτελεσματικότητα του ευρετηρίου. Μια συλλογή δημιουργείται φυσικά μόλις δημιουργηθεί το πρώτο έγγραφο σε αυτήν.

Μπορούμε να χρησιμοποιήσουμε χώρο ονομάτων για λογική ομαδοποίηση των συλλογών. Για παράδειγμα: Μπορεί να υπάρχει μία συλλογή που ονομάζεται db.studytonight.users για να αποθηκεύει τις πληροφορίες χρήστη, στη συνέχεια μπορεί να υπάρχουν συλλογές όπως το db.studytonight.forum.questions και το db.studytonight.forum. όπου μπορούμε να αποθηκεύουμε τις ερωτήσεις και τις απαντήσεις στο forum.

Το έγγραφο στο MongoDB είναι ένα σύνολο ζευγαριών κλειδιού-τιμής. Τα έγγραφα μπορούν να έχουν δυναμικό σχήμα, δηλαδή τα έγγραφα της ίδιας συλλογής δεν χρειάζεται να διαθέτουν το ίδιο σύνολο πεδίων.

Στην εικόνα που ακολουθεί, μπορούμε να δούμε ότι υπάρχει μια συλλογή με διάφορα έγγραφα μέσα της. Όλα τα έγγραφα έχουν πληροφορίες όπως το όνομα του φοιτητή, το μάθημα τη διεύθυνση και ένα πεδίο με όνομα _id το οποίο αντιπροσωπεύει το κύριο αναγνωριστικό κλειδιού του κάθε εγγράφου. Όπως μπορούμε να δούμε το _id δεν είναι ένας απλός αριθμός αλλά ένα αντικείμενο (Object). Η προκαθορισμένη μορφή του _id είναι ένας συνδυασμός αναγνωριστικού μηχανήματος, χρονικής σήμανσης και αναγνωριστικού διεργασίας, αλλά ο χρήστης μπορεί να το αλλάξει σε οτιδήποτε.



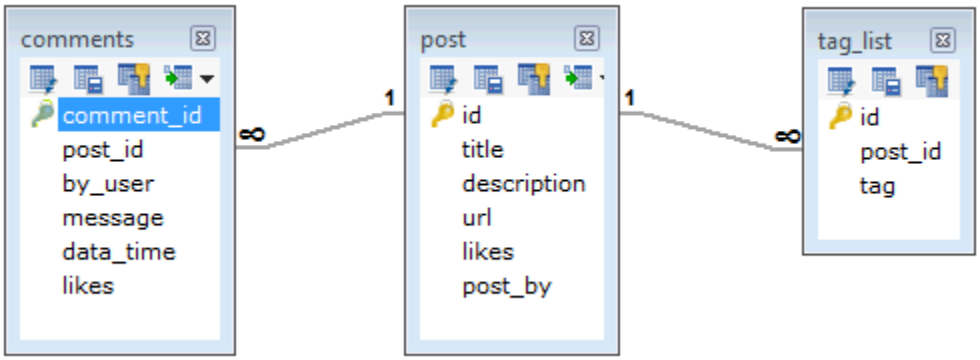
Εικόνα 22 Συλλογή και έγγραφα [28]

Ας περιγράψουμε τώρα τη διαδικασία που θα ακολουθήσουμε για να μετασχηματίσουμε μια σχεσιακή βάση δεδομένων σε MongoDB. Έστω ότι πρέπει να καταχωρούμε αναρτήσεις (post) με τις ακόλουθες ιδιότητες:

- Κάθε ανάρτηση είναι ξεχωριστή (περιέχει μοναδικό τίτλο, περιγραφή και url).

- Κάθε ανάρτηση μπορεί να έχει μία ή περισσότερες ετικέτες.
- Κάθε ανάρτηση έχει το όνομα του εκδότη της και τον συνολικό αριθμό των likes.
- Κάθε ανάρτηση μπορεί να έχει μηδενικά ή περισσότερα σχόλια και τα σχόλια πρέπει να περιέχουν όνομα χρήστη, μήνυμα, χρόνο δεδομένων και επισήμανση «μου αρέσει».

Έστω ότι έχουμε σχεδιάσει την ακόλουθη σχεσιακή βάση δεδομένων:



Εικόνα 23 Σχεδιασμός σχεσιακής βάσης δεδομένων [16]

Όμως στη MongoDB, ο σχεδιασμός σχήματος θα έχει μία συλλογή με την ακόλουθη γενική δομή:

```
{
_id: POST_ID
title: TITLE_OF_POST,
description: POST_DESCRIPTION,
by: POST_BY,
url: URL_OF_POST,
tags: [TAG1, TAG2, TAG3],
likes: TOTAL_LIKES,
comments: [
{
user: 'COMMENT_BY',
message: TEXT,
datecreated: DATE_TIME,
```

```
like: LIKES
},
{
user: 'COMMENT_BY',
message: TEST,
dateCreated: DATE_TIME,
like: LIKES
}}}
```

Όπως παρατηρούμε κάθε έγγραφο θα περιέχει όλες τις σχετικές με αυτό πληροφορίες, χωρίς να είναι απαραίτητο να σπάνε οι πληροφορίες αυτές σε πίνακες και να υπάρχουν και οι απαραίτητες συνδέσεις μεταξύ τους.

Ας δούμε τώρα τα βήματα για τη δημιουργία μιας βάσης δεδομένων, της συλλογής και της προσθήκης εγγράφων στη συλλογή. Δίνοντας την ακόλουθη εντολή λέμε στη MongoDB να δημιουργήσει, αν δεν υπάρχει και στη συνέχεια να κάνει ενεργή τη βάση δεδομένων που αναφέρουμε (Database_Name). Αν υπάρχει η βάση δεδομένων τότε την κάνει ενεργή.

```
use Database_Name
```

Αν θέλουμε να δούμε ποιες βάσεις δεδομένων υπάρχουν χρησιμοποιούμε την ακόλουθη εντολή και η MongoDB θα εμφανίσει όλες τις βάσεις δεδομένων που έχουν δημιουργηθεί:

```
show dbs
```

Αν θέλουμε να διαγράψουμε μια βάση δεδομένων θα πρέπει αρχικά να την επιλέξουμε με την εντολή use και στη συνέχεια να δώσουμε την εντολή:

```
db.dropDatabase()
```

η οποία θα διαγράψει και τα σχετιζόμενα αρχεία δεδομένων. Έχοντας δημιουργήσει τη βάση δεδομένων και επιλέγοντάς την, το επόμενο βήμα είναι να δημιουργήσουμε τη συλλογή. Για τη δημιουργία της θα πληκτρολογήσουμε την εντολή:

```
db.createCollection(name, options)
```

Το name είναι το όνομα που θέλουμε να δώσουμε στη συλλογή και το options είναι μια προαιρετική παράμετρος με τη βοήθεια της οποίας καθορίζουμε το μέγεθος της μνήμης και το ευρετήριο. Οι τιμές που μπορούμε να εισάγουμε

Πεδίο	**Τύπος**	**Περιγραφή**
Capped	Boolean	(Προαιρετικό) Εάν έχει οριστεί ως αληθές, ενεργοποιεί μια συλλογή με περιορισμούς η οποία είναι μια συλλογή σταθερού μεγέθους που αντικαθιστά αυτόματα τις παλαιότερες καταχωρίσεις της όταν φτάσει στο μέγιστο μέγεθός της. Εάν οριστεί ως αληθές, πρέπει επίσης να καθοριστεί και η παράμετρος του μεγέθους.
AutoIndexID	Boolean	(Προαιρετικό) Εάν έχει οριστεί ως αληθές, θα δημιουργηθεί αυτόματα ευρετήριο στο πεδίο ID. Η προεπιλεγμένη τιμή είναι ψευδής.
Size	Αριθμός	(Προαιρετικό) Καθορίζει ένα μέγιστο μέγεθος σε byte για μια περιορισμένη συλλογή. Εάν το πεδίο Capped είναι αληθές, τότε πρέπει να δοθεί τιμή σ' αυτό το πεδίο.
Max	Αριθμός	(Προαιρετικό) Καθορίζει τον μέγιστο αριθμό εγγράφων που επιτρέπονται στη συλλογή με περιορισμούς.

Για να δούμε ποιες συλλογές έχουν δημιουργηθεί σε μια βάση δεδομένων θα πρέπει να δώσουμε την ακόλουθη εντολή:

```
show collections
```

Για να διαγράψουμε μια συλλογή θα δώσουμε την εντολή:

```
db.COLLECTION_NAME.drop()
```

Για να εισάγουμε ένα έγγραφο σε μια συλλογή θα πρέπει να δώσουμε την εντολή:

```
db.COLLECTION_NAME.insert(document)
```

Για παράδειγμα για να εισάγουμε ένα έγγραφο, θα δίναμε:

```
db.COLLECTION_NAME.insert(
  {
    course: "java",
    details: {
      duration: "6 months",
      Trainer: "Sonoo jaiswal"
    },
    Batch: [ { size: "Small", qty: 15 }, { size: "Medium", qty: 25 } ],
    category: "Programming language"
  }
)
```

Για να εμφανιστούν όλα τα έγγραφα που υπάρχουν σε μια συλλογή θα πρέπει να εκτελέσουμε την εντολή:

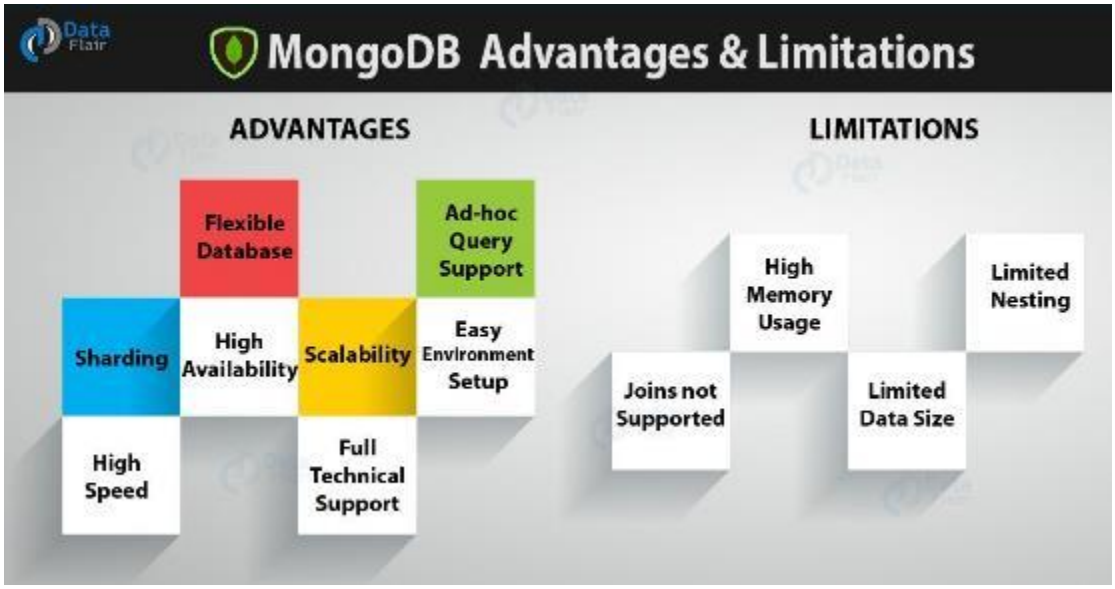
```
db.COLLECTION_NAME.find()
```

Για να διαγράψουμε έγγραφα από μια συλλογή θα δώσουμε την παρακάτω εντολή. Αν στο DELETION_CRITERIA βάλουμε την τιμή {} τότε θα διαγραφούν όλες οι εγγραφές [16].

```
db.collection_name.remove (DELETION_CRITERIA)
```

1.6.3 Πλεονεκτήματα/Μειονεκτήματα

Μετά από την ιστορική αναδρομή στη MongoDB, θα παρουσιάσουμε τα πλεονεκτήματα, αλλά και τα μειονεκτήματα που μπορεί να έχει η χρήση της σε εφαρμογές.



Εικόνα 24 Πλεονεκτήματα/Μειονεκτήματα της MongoDB [9]

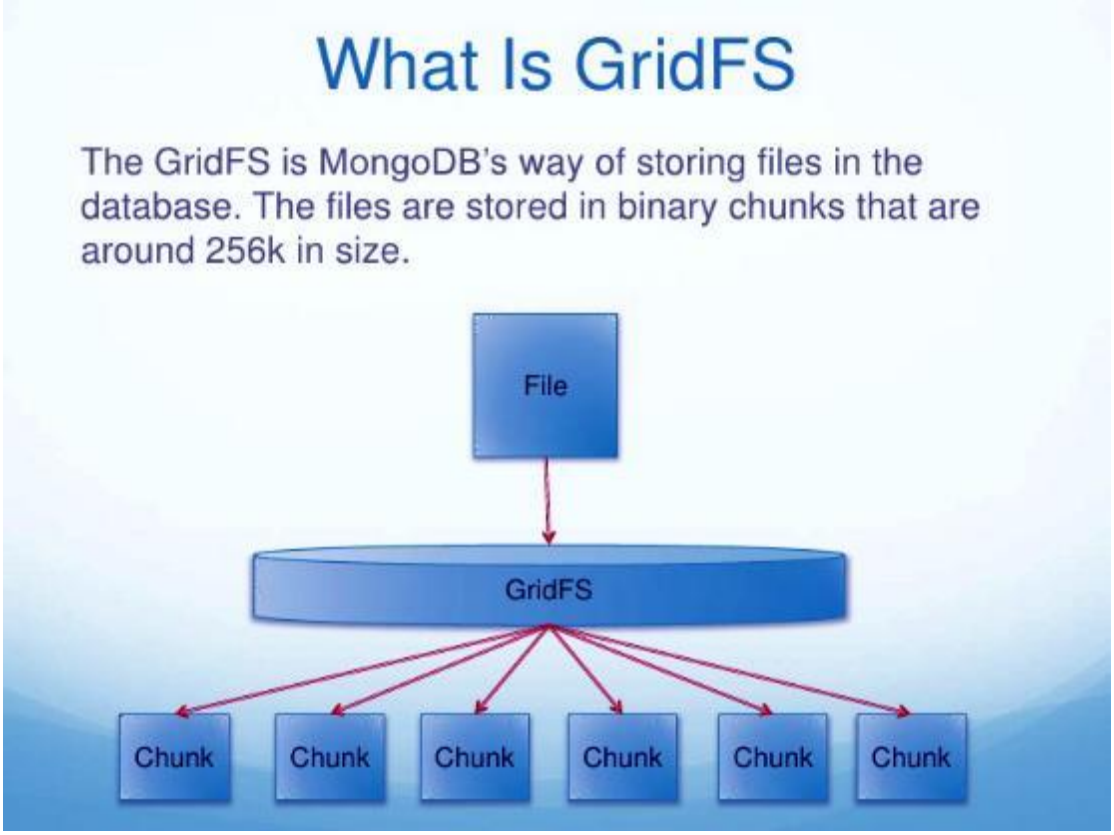
Ας ξεκινήσουμε με τα πλεονεκτήματα:

- Απλότητα και ευκολία εγκατάστασης: Η MongoDB, ως σύστημα διαχείρισης βάσεων δεδομένων, είναι πολύ πιο απλή και οι πολυπλοκότητες που συνοδεύουν τις σχεσιακές βάσεις δεδομένων καταργούνται. Η αποθήκευση προσανατολισμένη προς έγγραφα JSON προσθέτει περισσότερη απλότητα. Η εφαρμογή βάσης δεδομένων, σε αντίθεση με τις σχεσιακές βάσεις δεδομένων, είναι ευκολότερη στη ρύθμιση και προσφέρει ένα πρόγραμμα σε JavaScript για εκτέλεση ερωτημάτων στη βάση δεδομένων. Η βάση δεδομένων είναι επίσης εύκολη στην κλιμάκωση, ένα από τα σημαντικότερα οφέλη της MongoDB.
- Ευελιξία: Το σχήμα της MongoDB δεν είναι προκαθορισμένο δηλαδή υπάρχει μια δυναμική σχηματική αρχιτεκτονική που λειτουργεί με μη δομημένα δεδομένα και αποθήκευση. Αυτό σημαίνει ότι διαφορετικοί τύποι δεδομένων μπορούν να αποθηκευτούν σε ξεχωριστά έγγραφα σε μία συλλογή. Τα έγγραφα διαφέρουν ως προς το περιεχόμενο, τα πεδία και τα μεγέθη. Το αποτέλεσμα

είναι αυξημένη ελευθερία και ευελιξία για την αποθήκευση ποικίλων τύπων δεδομένων.

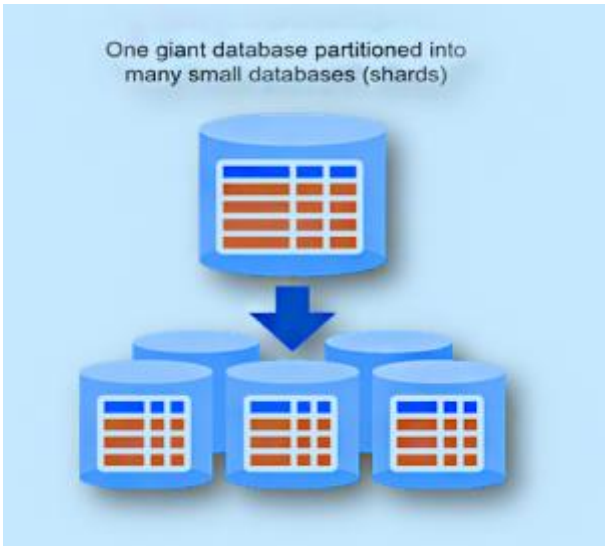
- Αξιοπιστία, data replication και υψηλή διαθεσιμότητα: Η MongoDB επιτρέπει στους χρήστες να αναπαράγουν δεδομένα σε πολλούς διακομιστές που λειτουργούν με την τεχνολογία mirroring, γεγονός που εξασφαλίζει την αξιοπιστία των δεδομένων. Σε περίπτωση διακοπής λειτουργίας διακομιστή, κάποιος άλλος που λειτουργεί ως mirror είναι ακόμα διαθέσιμος και η επικοινωνία με τη βάση δεδομένων παραμένει αμετάβλητη. Το replication των δεδομένων και οι δυνατότητες gridFS δεν βελτιώνουν μόνο την αξιοπιστία της MongoDB, αλλά και τη διαθεσιμότητά της. Σαν αποτέλεσμα, η εφαρμογή προσφέρει υψηλή απόδοση, καθιστώντας την ένα από τα πιο σημαντικά πλεονεκτήματα του MongoDB. Το GridFS είναι μια δυνατότητα αποθήκευσης και ανάκτησης αρχείων. Για αρχεία μεγαλύτερα από 16 MB αυτή η δυνατότητα είναι πολύ χρήσιμη. Το GridFS διαιρεί ένα έγγραφο σε μέρη που ονομάζονται κομμάτια (chunks) και τα αποθηκεύει σε ξεχωριστά έγγραφα. Αυτά τα κομμάτια έχουν προεπιλεγμένο μέγεθος 255kB εκτός από το τελευταίο κομμάτι. Όταν θα αναζητήσουμε με τη βοήθεια του GridFS ένα αρχείο, αυτό αναλαμβάνει να συγκεντρώσει όλα τα κομμάτια.

Εικόνα 25 Υψηλή διαθεσιμότητα [9]



Εικόνα 26 GridFS MongoDB [9]

- Υποστηρίζει Auto Sharding και Auto Failover: Μεγάλα δεδομένα για αποθήκευση διανέμονται σε διάφορους διακομιστές συνδεδεμένους στην MongoDB. Με τον τρόπο αυτό επιτυγχάνεται η ελαχιστοποίηση του κινδύνου αποτυχίας διακομιστή που προκύπτει από την αδυναμία χειρισμού μεγάλων δεδομένων. Δηλαδή αν υπάρξει περίπτωση κατά την οποία ο διακομιστής δεν μπορεί να χειριστεί τα δεδομένα λόγω του μεγέθους τους, τα διαιρεί αυτόματα περαιτέρω χωρίς διακοπή της δραστηριότητας και τα κατανέμει σε πολλούς διακομιστές.



Εικόνα 27 Auto sharding [9]

- Ad-hoc υποστήριξη ερωτημάτων: Ένα ad-hoc ερώτημα είναι ένα μη τυπικό ερώτημα. Δημιουργείται για την απόκτηση πληροφοριών εάν και όταν απαιτείται. Η MongoDB προσφέρει μια βελτιωμένη δυνατότητα ad-hoc ερωτημάτων. Αυτό επιτρέπει σε μια εφαρμογή να προετοιμαστεί για τα επόμενα ερωτήματα που μπορεί να προκύψουν στο μέλλον.
- Τεχνική υποστήριξη: Η MongoDB προσφέρει τεχνική υποστήριξη για τις διάφορες υπηρεσίες που παρέχει. Υπάρχει τεχνική υποστήριξη στα φόρουμ της κοινότητας, για το Atlas ή το Cloud Manager καθώς και το Enterprise ή το Ops Manager. Επίσης σε περίπτωση οποιουδήποτε προβλήματος, η επαγγελματική ομάδα υποστήριξης πελατών είναι έτοιμη να βοηθήσει τους πελάτες [6].
- Απόδοση: Με την αποθήκευση της πλειοψηφίας των δεδομένων στη μνήμη RAM, η απόδοση ερωτήματος στη MongoDB είναι πολύ πιο γρήγορη. Αντί να αναζητά τα δεδομένα από το σκληρό δίσκο κατά τη διάρκεια ενός ερωτήματος, τα αναζητά από τη μνήμη RAM και τα αποτελέσματα των ερωτημάτων είναι πολύ πιο γρήγορα. Όμως, για να επωφεληθούμε από αυτή τη βελτιωμένη απόδοση, θα πρέπει να έχουμε διαθέσιμη αρκετή μνήμη RAM και να είμαστε προσεκτικοί στον καθορισμό index.

Όπως όλα τα πράγματα στον κόσμο, έτσι και η MongoDB εκτός από τα πλεονεκτήματα, έχει και ορισμένα μειονεκτήματα όπως:

- Δεν υποστηρίζει συναλλαγές (transaction): Αν και όλο και λιγότερες εφαρμογές απαιτούν συναλλαγές, εξακολουθούν να υπάρχουν κάποιες που χρειάζονται

συναλλαγές για να ενημερώσουν πολλά έγγραφα/συλλογές. Αν η εφαρμογή μας απαιτεί τη χρήση συναλλαγών, τότε δεν πρέπει να χρησιμοποιηθεί η MongoDB.

- Έλλειψη Join: Η πράξη της ένωσης μεταξύ εγγράφων είναι δύσκολο να επιτευχθεί, παρότι με την έκδοση 3.2 εισήχθησαν τα left outer join.
- Ευρετηρίαση (Indexing): Για να επιτευχθούν οι μεγάλες ταχύτητες που αναφέραμε στα πλεονεκτήματα, θα πρέπει να επιλεγούν και οι κατάλληλοι index. Αν δεν επιλεγούν οι κατάλληλοι index, τότε η MongoDB θα λειτουργεί με αρκετά χαμηλή ταχύτητα.
- Υψηλή χρήση μνήμης: όπως είδαμε στα πλεονεκτήματα είναι απαραίτητη η χρήση μεγάλης μνήμης RAM. Επίσης αν η βάση μας απαιτεί δεδομένα με χρήση join, επειδή δεν παρέχεται η συγκεκριμένη δυνατότητα από τη MongoDB θα πρέπει να χρησιμοποιηθούν διπλότυπα δεδομένα που οδηγούν στην αύξηση του πλεονασμού δεδομένων που καταλαμβάνει περιττό χώρο στη μνήμη [34].

ΚΕΦΑΛΑΙΟ 2. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα βήματα που ακολουθήσαμε για να σχεδιάσουμε και να υλοποιήσουμε την εφαρμογή συλλογή Tweets και καταχώρησή τους στη βάση δεδομένων MongoDB. Θα δούμε τα βήματα της εγκατάστασης της MongoDB, της καταχώρισης των στοιχείων για την ανάπτυξη της εφαρμογής για τη συλλογή Tweets και τις απαραίτητες βιβλιοθήκες της γλώσσας προγραμματισμού Python για τη σύνδεση με τη βάση δεδομένων MongoDB, τη σύνδεση με το Twitter και το γραφικό περιβάλλον χρήστη. Τέλος θα παρουσιάσουμε τις δυνατότητες που προσφέρει η εφαρμογή στο χρήστη.

2.2 Σχεδιασμός

Αρχικά μελετήσαμε τις δυνατότητες πρόσβασης που είχαμε για το Twitter. Διαπιστώσαμε ότι παρέχονταν δυνατότητες αναζήτησης των tweets και άντλησης στοιχείων τόσο για τα tweets όσο και για τους ανθρώπους που τα έκαναν, χωρίς βέβαια να παρέχετε δωρεάν η δυνατότητα για αναζήτηση σε συγκεκριμένο διάστημα. Η αναζήτηση των tweets περιοριζόταν στις τελευταίες επτά ημέρες.
Επιλέξαμε να αναζητήσουμε tweets με βάση κάποιο hashtag και στη συνέχεια να αντλήσουμε τα ακόλουθα:

- Μοναδικό id tweet,
- Όνομα χρήστη,
- Τοποθεσία,
- Ημερομηνία που έγινε το tweet,
- Κείμενο του tweet.

Μελετήσαμε τις βιβλιοθήκες της Python που έπρεπε να χρησιμοποιήσουμε τόσο για σύνδεση και άντληση δεδομένων από το Twitter αλλά και για σύνδεση και επικοινωνία με τη MongoDB.
Τέλος αποφασίσαμε, για να είναι πιο φιλικό προς το χρήστη, να δημιουργήσουμε και ένα GUI το οποίο θα παρέχει βασικές λειτουργίες στο χρήστη του, δηλαδή:

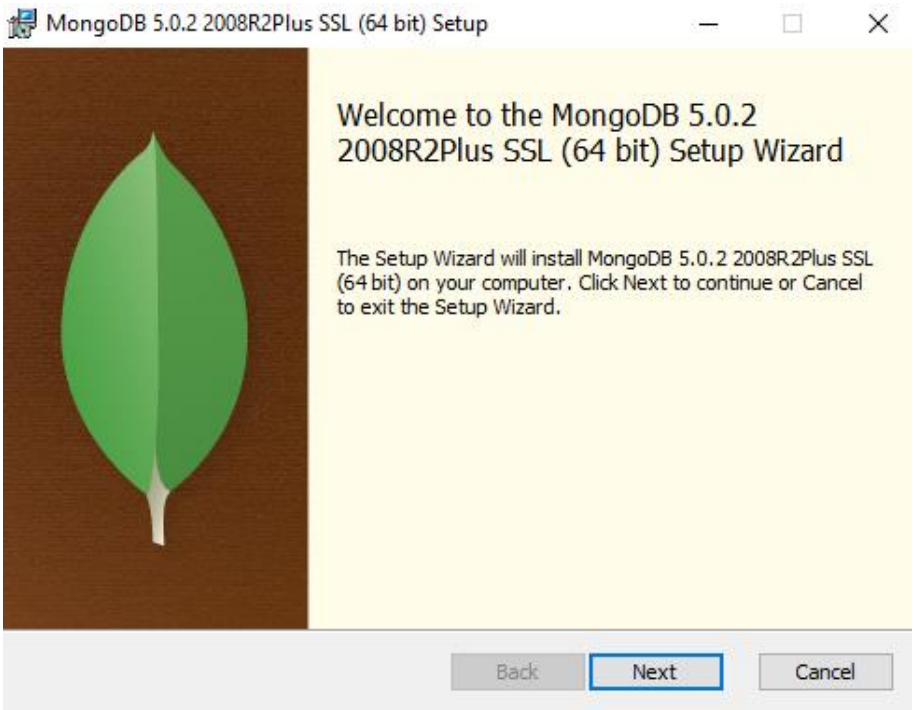
- Προσθήκη hashtag

- Επιλογή πλήθους tweets που θέλουμε να αναζητήσουμε
- Αναζήτηση στο twitter με βάση το hashtag και το πλήθος που έχουμε εισάγει.
- Στη συνέχεια για κάθε ένα tweet ελέγχουμε αν έχει προστεθεί ήδη στη βάση δεδομένων MongoDB αναζητώντας το με βάση το tweet id (που είναι μοναδικό). Αν δεν υπάρχει το προσθέτουμε στη βάση δεδομένων.
- Αναζήτηση με hashtag ή κείμενο στο κείμενο των tweets. Η αναζήτηση γίνεται προσπαθώντας να βρούμε το κείμενο χωρίς διάκριση πεζών κεφαλαίων. Τα αποτελέσματα (tweet) θα εμφανιστούν σε νέο παράθυρο.

2.3 Υλοποίηση

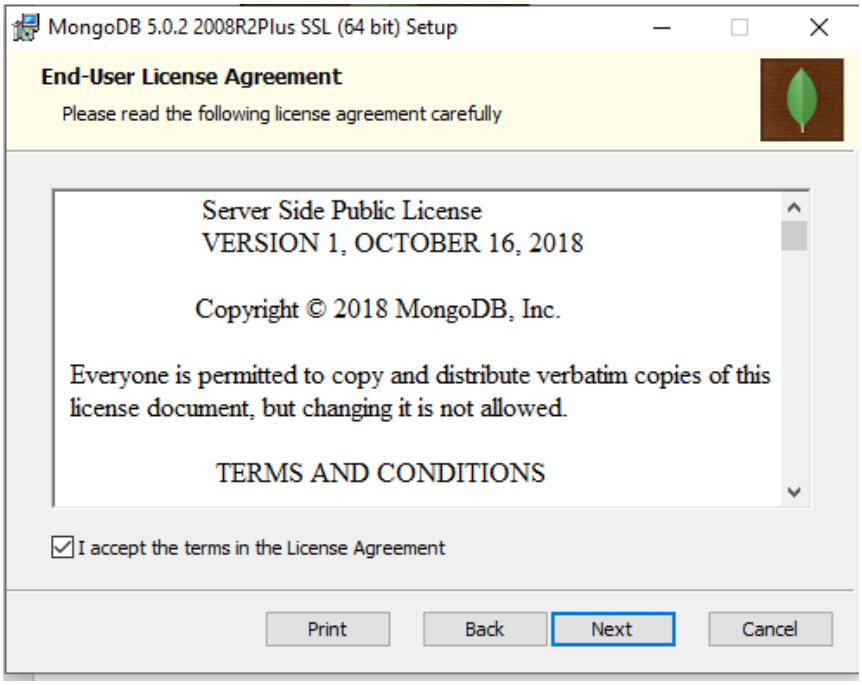
2.3.1 Εγκατάσταση MongoDB

Για να κατεβάσουμε τη MongoDB θα πρέπει να μεταβούμε στη διεύθυνση https://www.mongodb.com/try/download/community όπου έχοντας επιλέξει τον Community Server θα πρέπει να καθορίσουμε την πλατφόρμα που θέλουμε να τον εγκαταστήσουμε και να επιλέξουμε Download. Εκτελούμε το αρχείο που κατεβάσαμε και θα εμφανιστεί η πρώτη εικόνα της εγκατάστασης που μας πληροφορεί για την έκδοσης της MongoDB.



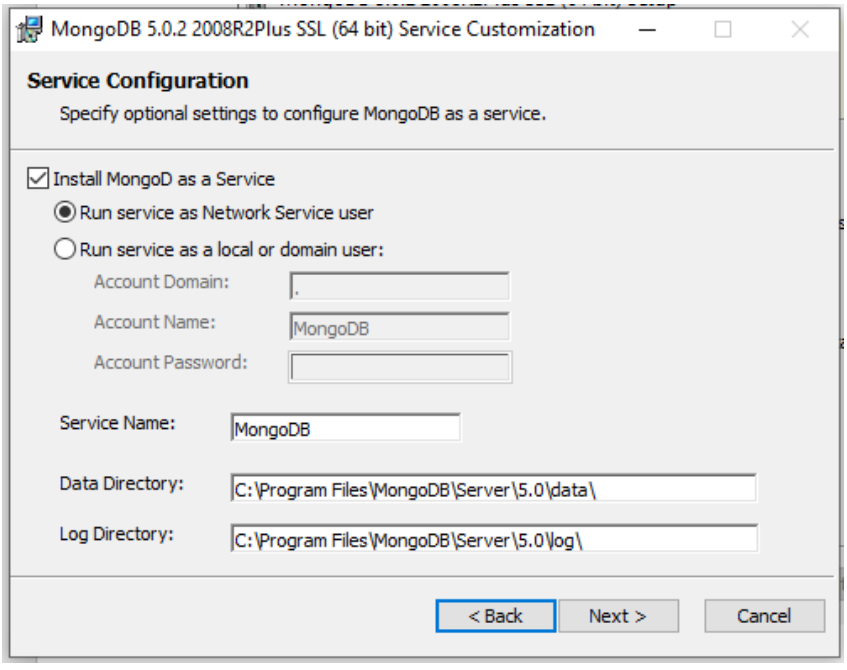
Εικόνα 28 Πρώτη οθόνη εγκατάστασης

Η επόμενη οθόνη που εμφανίζεται είναι οι όροι χρήσης τους οποίους πρέπει να αποδεχτούμε και να πατήσουμε το Next.



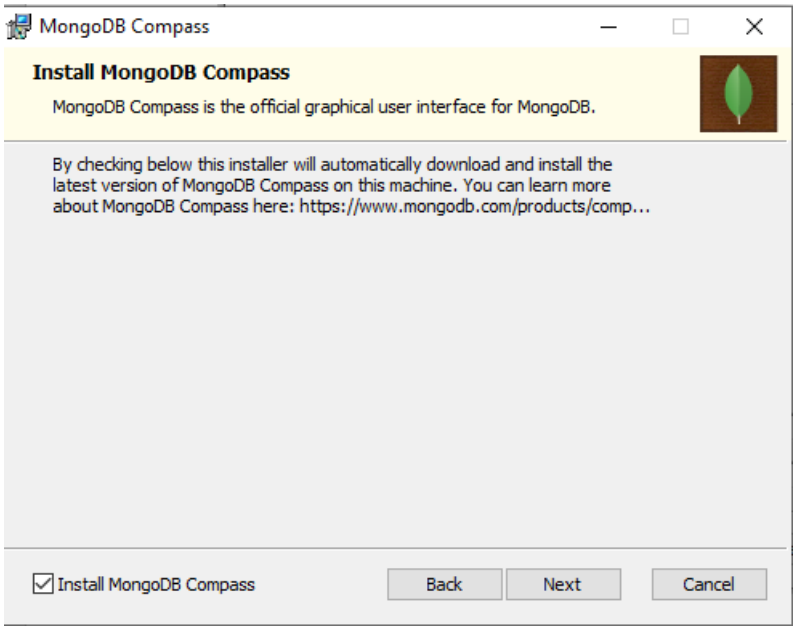
Εικόνα 29 Όροι Χρήσης

Στη συνέχεια θα ερωτηθούμε αν θέλουμε να εγκατασταθεί η MongoDB σαν υπηρεσία και το όνομα που θέλουμε να έχει.



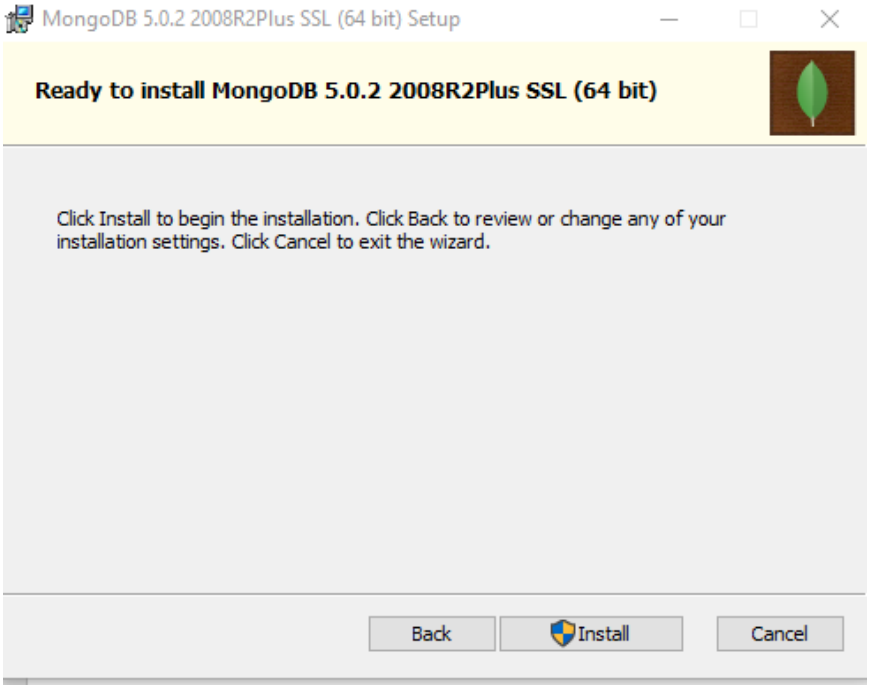
Εικόνα 30 Απόφαση για τον τρόπο εγκατάστασης

Κατόπιν θα επιλέξουμε αν θέλουμε να εγκατασταθεί και το MongoDB Compass ένα γραφικό περιβάλλον διεπαφής χρήστη για το χειρισμό της βάσης δεδομένων. Επιλέγουμε την εγκατάσταση του.

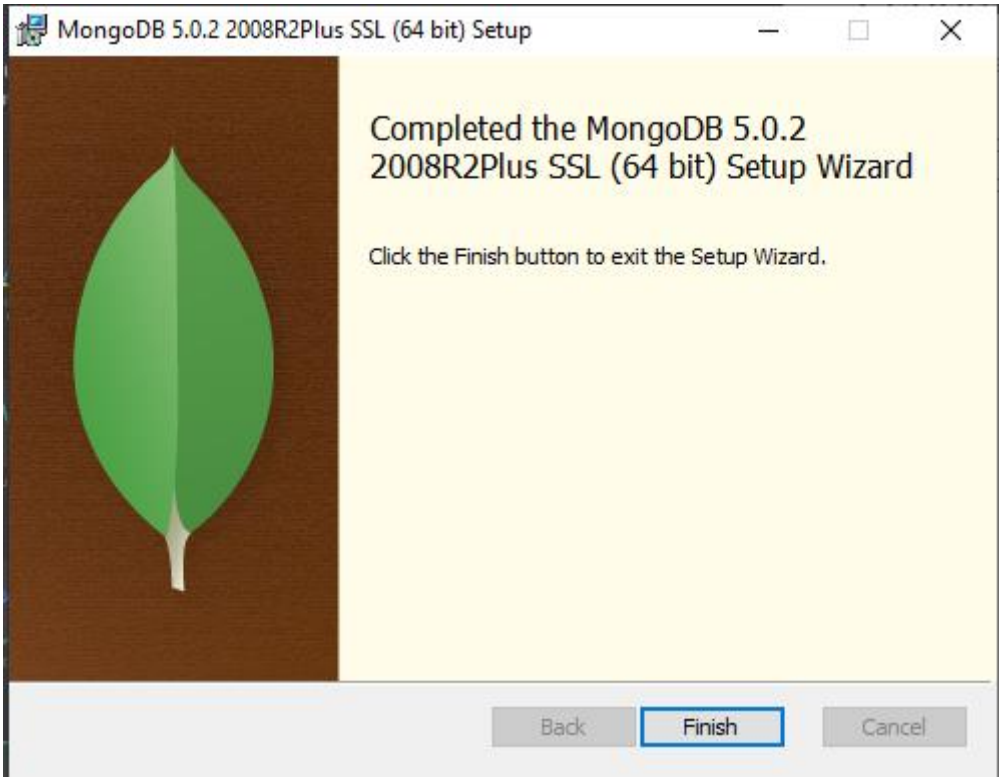


Εικόνα 31 Επιλογή για εγκατάσταση του MongoDB Compass

Η τελευταία οθόνη της εγκατάστασης μας ενημερώνει ότι έχουν ολοκληρωθεί οι ρυθμίσεις και απομένει να πατήσουμε το πλήκτρο Install για την έναρξη της εγκατάστασης της βάσης δεδομένων MongoDB.

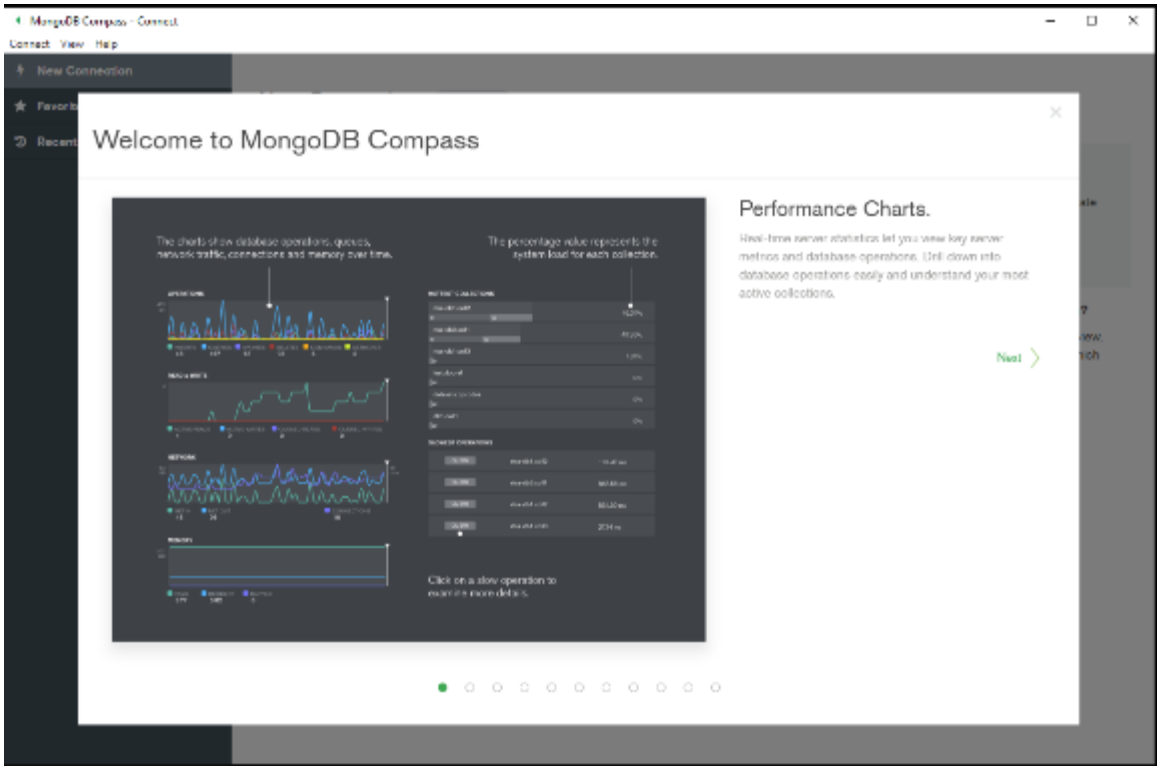


Αφού ολοκληρωθεί η εγκατάσταση της MongoDB θα εμφανιστεί ενημερωτική οθόνη. Θα πρέπει να θυμόμαστε ότι η MongoDB έχει εγκατασταθεί σαν υπηρεσία, επομένως για να μπορέσουμε να την ξεκινήσουμε ή να την σταματήσουμε, θα πρέπει να έχουμε πρόσβαση στις υπηρεσίες του υπολογιστή μας. Για να γίνει αυτό μπορούμε να κάνουμε δεξί κλικ στο εικονίδιο «Αυτός ο υπολογιστής» και να επιλέξουμε Διαχείριση. Στη συνέχεια επιλέγουμε το Υπηρεσίες και Εφαρμογές και κατόπιν το Υπηρεσίες όπου μπορούμε να αναζητήσουμε την υπηρεσία της MongoDB και να την ξεκινήσουμε ή να την σταματήσουμε.



Εικόνα 32 Ολοκλήρωση εγκατάστασης MongoDB

Έχοντας επιλέξει να εγκαταστήσουμε και το MongoDB Compass, θα εμφανιστεί και η αρχική του οθόνη, όπου για να μπορέσουμε να έχουμε πρόσβαση στη χρήση του, θα πρέπει να επιλέξουμε τις πληροφορίες που πρέπει να αποστέλλονται στη MongoDB και αφορούν από αναφορές σφαλμάτων, στατιστικά χρήσης του, αν θέλουμε να γίνεται αυτόματη ενημέρωση του προγράμματος κ.λπ. Με την ολοκλήρωση των επιλογών μας έχουμε πρόσβαση στο γραφικό περιβάλλον εργασίας του MonogoDB Compass και κατ' επέκταση και στη βάση δεδομένων MongoDB.

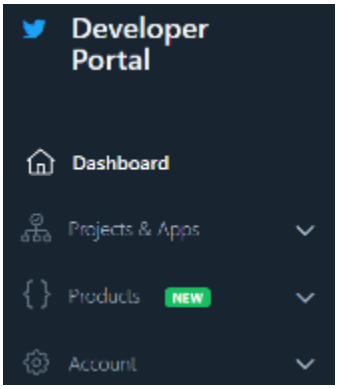


Εικόνα 33 MongoDB Compass

2.3.2 Twitter

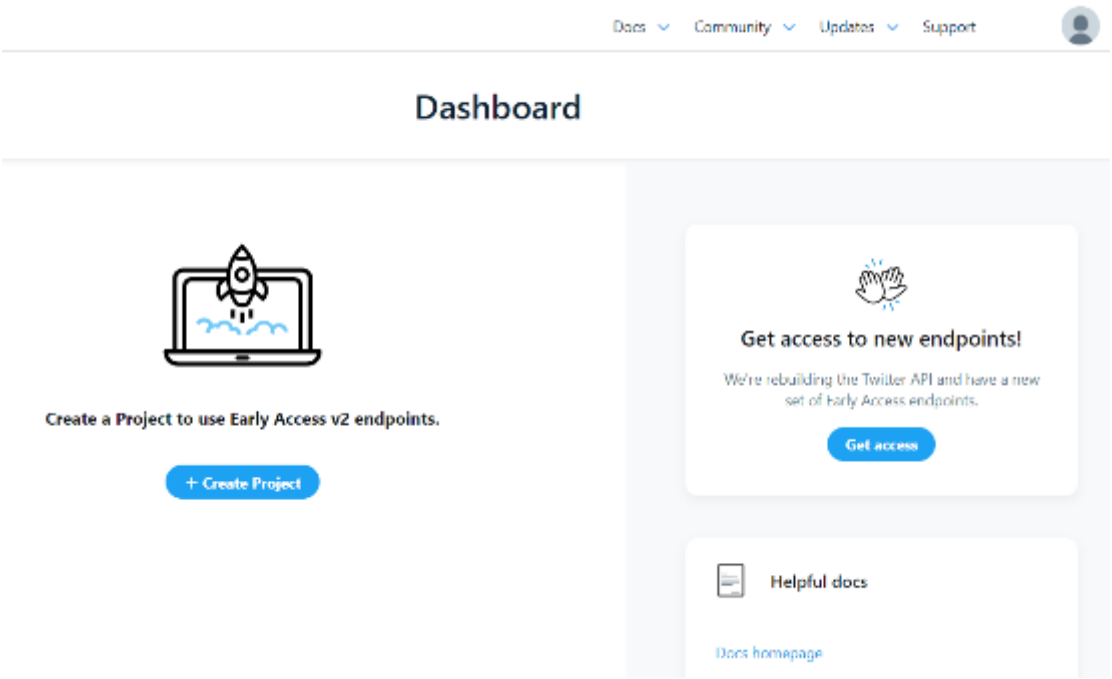
Στην ενότητα αυτή θα περιγράψουμε τα απαραίτητα βήματα για να δημιουργήσουμε μια εφαρμογή που θα αντλεί δεδομένα από το Twitter. Για να γίνει αυτό, θα πρέπει να δημιουργήσουμε ένα λογαριασμό στο Twitter (αν δεν έχουμε ήδη) και στη συνέχεια να επισκεφθούμε τη σελίδα apps.twitter.com. Στη σελίδα αυτή θα πρέπει να εισάγουμε το όνομα χρήστη και τον κωδικό πρόσβασης που έχουμε για το Twitter και μετά από έλεγχο θα μεταβούμε στην κεντρική σελίδα για την ανάπτυξη εφαρμογών για το Twitter.

Στο αριστερό τμήμα της οθόνης μπορούμε να δούμε το μενού επιλογών που έχουμε διαθέσιμο και τις εφαρμογές που έχουμε ήδη δηλώσει (αν υπάρχει κάποια θα εμφανιστεί κάτω από το μενού επιλογής Projects & Apps).



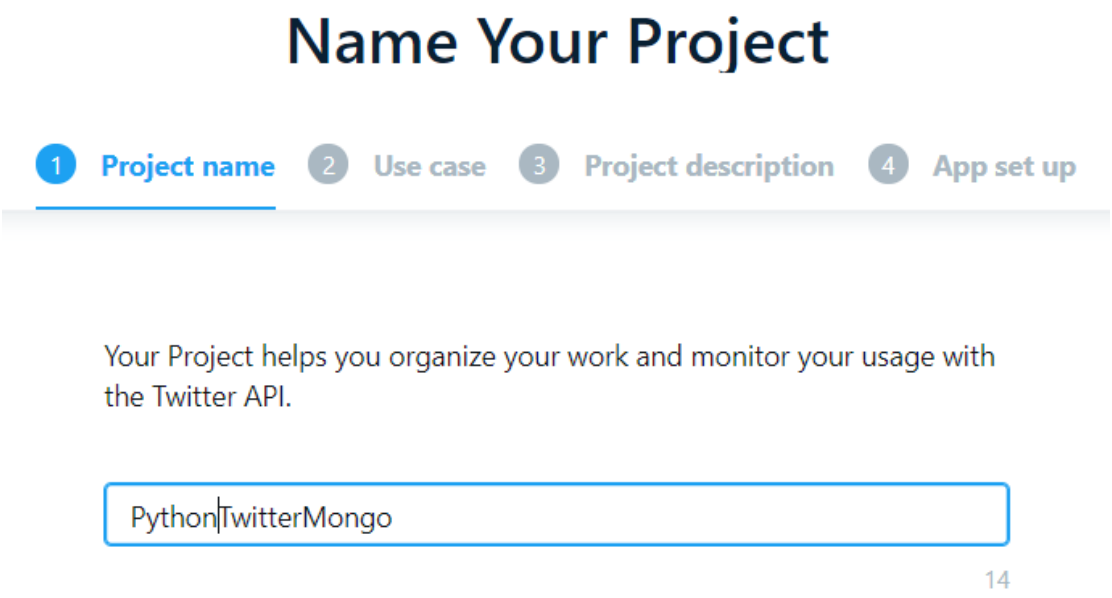
Εικόνα 34 Μενού επιλογών

Η κεντρική οθόνη περιέχει τις επιλογές για δημιουργία ενός νέου Project, πρόσβαση σε βοηθήματα και στην κοινότητα κ.λπ.



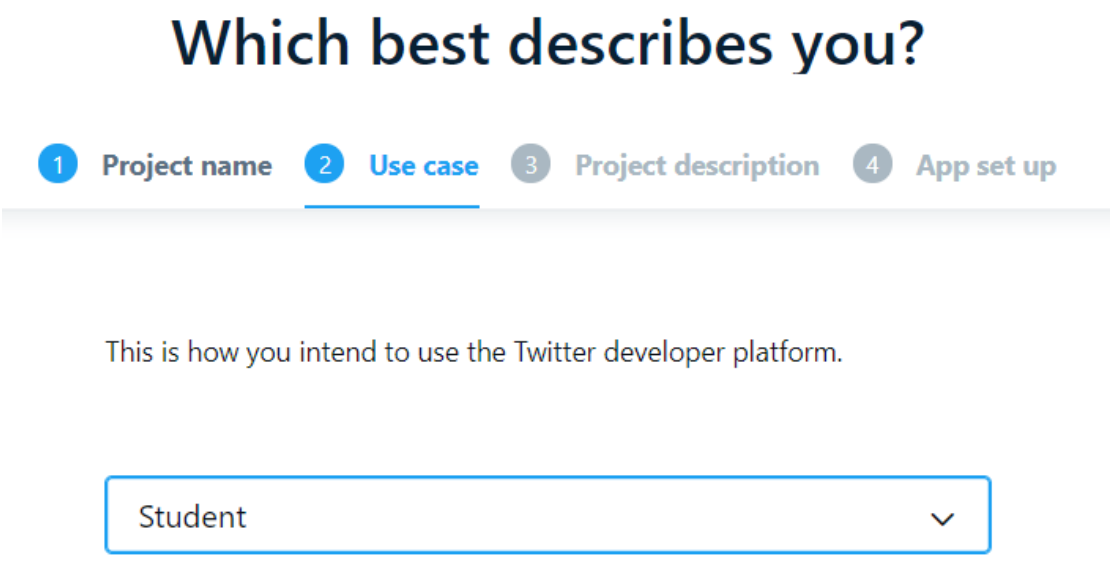
Εικόνα 35 Κεντρική οθόνη

Επιλέγουμε Create Project και θα ξεκινήσουμε να εισάγουμε τα στοιχεία που απαιτεί το Twitter για τη δημιουργία εφαρμογών που θα έχουν πρόσβαση στα δεδομένα του. Αρχικά θα πρέπει να εισάγουμε το όνομα του Project.



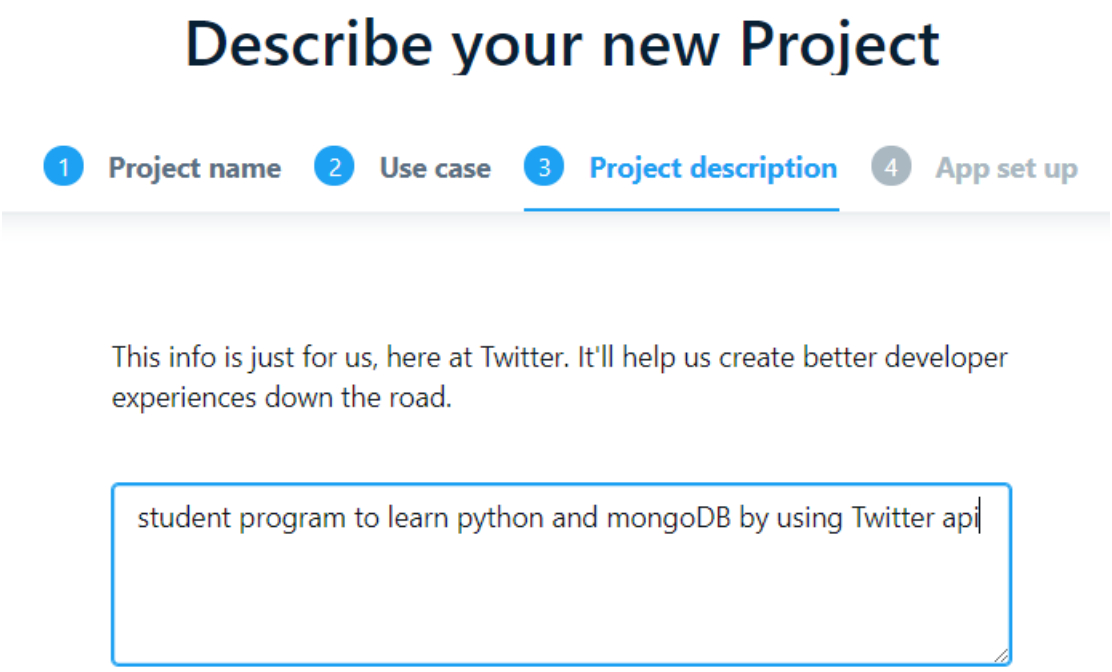
Εικόνα 36 Ονοματοδοσία προγράμματος

Στη συνέχεια το Twitter μας ζητά να επιλέξουμε το λόγο της χρήσης της εφαρμογής μας. Επιλέγουμε το Student.



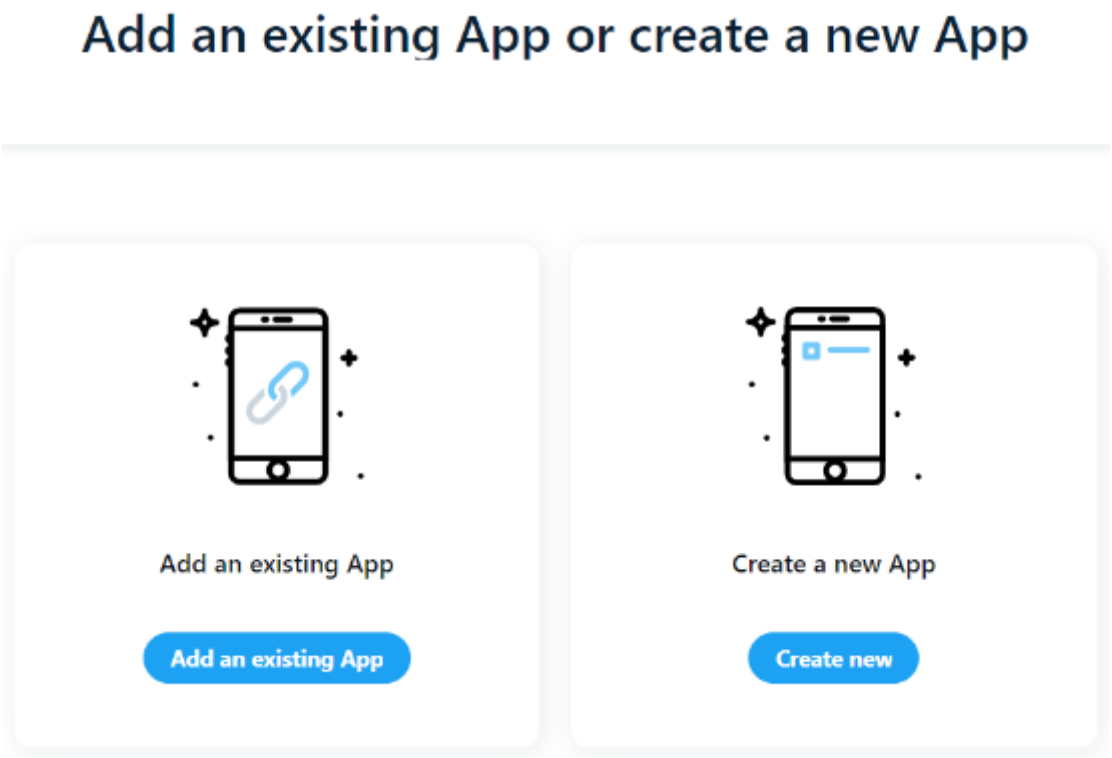
Εικόνα 37 Επιλογή λόγου χρήσης project

Κατόπιν εισάγουμε μια σύντομη περιγραφή για το Project



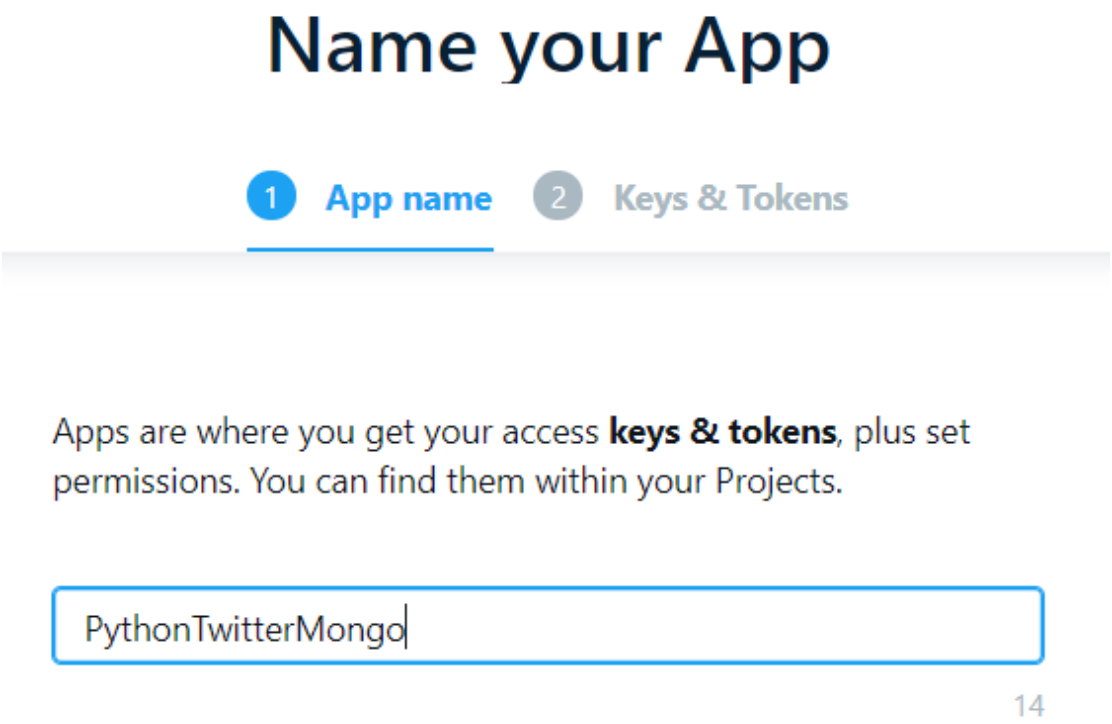
Εικόνα 38 Περιγραφή χρήσης project

Ολοκληρώνοντας την εισαγωγή της περιγραφής, έχει δημιουργηθεί το Project και πρέπει να προχωρήσουμε στο επόμενο βήμα της επιλογής της δημιουργίας μιας νέας εφαρμογής ή της αντιστοίχισης μιας υπάρχουσας.

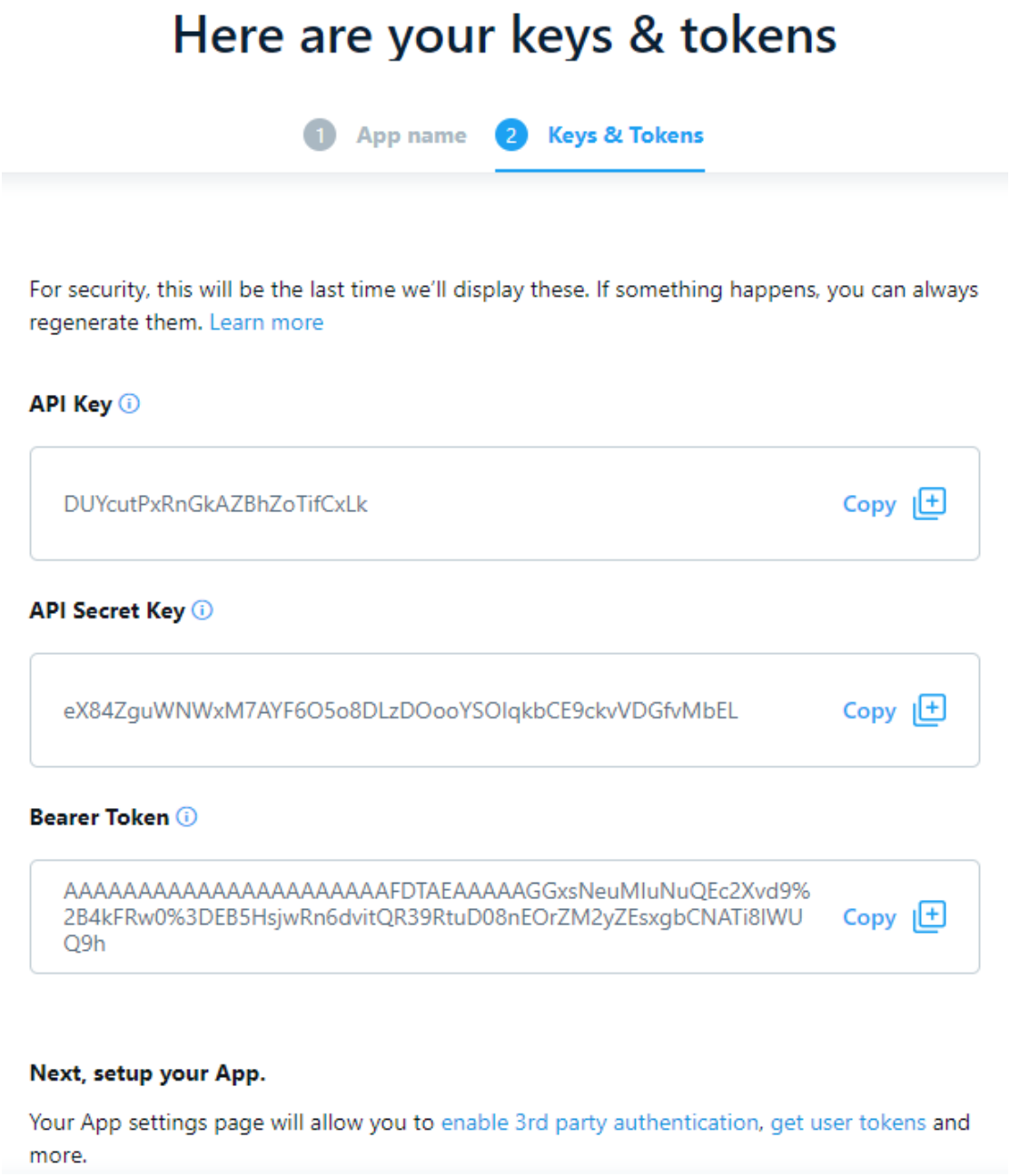


Εικόνα 39 Αντιστοίχηση ή δημιουργία εφαρμογής σε project

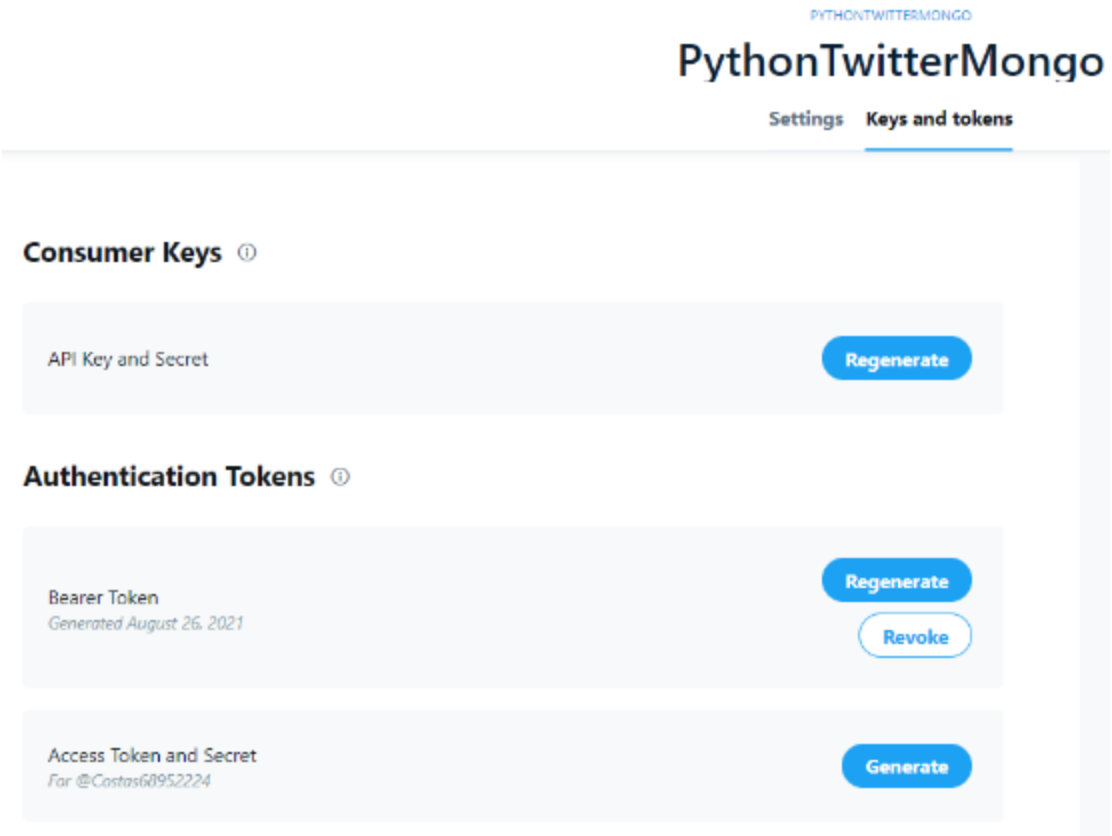
Επιλέγουμε τη δημιουργία μιας νέας εφαρμογής και το επόμενο που θα πρέπει να κάνουμε είναι η πληκτρολόγηση του ονόματος της εφαρμογής. Επιλέγουμε το ίδιο όνομα με το project.



Όταν εισάγουμε το όνομα θα εμφανιστούν τα απαραίτητα token για τη δημιουργία της εφαρμογής και της σύνδεσης με το Twitter για την άντληση ή εισαγωγή δεδομένων.



Από την παραπάνω λίστα απουσιάζουν τα Access token τα οποία για να τα δημιουργήσουμε θα πρέπει να επιλέξουμε από το αριστερό μενού την εφαρμογή μας, όπου μπορούμε να επαναδημιουργήσουμε τα API keys, το Bearer Token και να δημιουργήσουμε το Access Token.



Επιλέγοντας τη δημιουργία, θα παραχθούν και θα εμφανιστούν τα Access Token και Secret. Επομένως τα απαραίτητα token και key για την εφαρμογή μας είναι:

- **API key**: DUYcutPxRnGkAZBhZoTifCxLk
- **API secret key**: eX84ZguWNWxM7AYF6O5o8DLzDOooYSOIqkbCE9ckvVDGfvMbEL
- **Bearer Token**: AAAAAAAAAAAAAAAAAAAAAFDTAEAAAAAGGxsNeuMIuNuQEc2Xvd9%2B4kFRw0%3DEB5HsjwRn6dvitQR39RtuD08nEOrZM2yZEsxgbCNATi8lWUQ9h
- **Access Token**: 1126015992734334976-OAospvjY457xrKvUGawsd6X2YDGchh
- **Access Token Secret**: CLG3sV35OppbxiS6JE86lYcg9limPn2WZa1wtqVomg6G3

Το API έκδοση 2 του Twitter είναι εξοπλισμένο με ένα νέο σύνολο παραμέτρων που ονομάζονται πεδία, το οποίο επιτρέπει στους προγραμματιστές να επιλέξουν μόνο τα δεδομένα που θέλουν από κάθε αντικείμενο στο πρόγραμμά τους. Κάθε tweet έχει ένα σύνολο από αντικείμενα και το κάθε ένα αντικείμενο από αυτά αποτελείται από ένα σύνολο από παραμέτρους. Τα αντικείμενα που μπορούμε να διακρίνουμε είναι:

- Tweet
- User field
- Media
- Poll

- Place

Από τα παραπάνω αντικείμενα, εμείς θα χρησιμοποιήσουμε το Tweet και το User field. Από το Tweet θα χρησιμοποιήσουμε τις ακόλουθες παραμέτρους:

- id: ένα μοναδικό αναγνωριστικό για κάθε ένα tweet. Είναι string που συνήθως αποτελείται από αριθμούς.
- text: το κείμενο του tweet σε UTF-8
- created_at: η ημερομηνία δημιουργία του tweet. Θα επιστραφεί μια ημερομηνία σε μορφή ISO 8601 για παράδειγμα 2019-06-04T23:12:08.000Z

Από το User field θα χρησιμοποιήσουμε τα ακόλουθα:

- name: το όνομα του χρήστη όπως έχει δηλωθεί
- location: Η τοποθεσία που καθορίζεται στο προφίλ του χρήστη, εάν ο χρήστης παρείχε κάποια τιμή. Επειδή είναι μια τιμή ελεύθερης μορφής, ενδέχεται να μην υποδεικνύει μια έγκυρη τοποθεσία.

2.3.3 Python και βιβλιοθήκες

Όπως αναφέραμε η εφαρμογή θα υλοποιηθεί σε Python, γι' αυτό θα πρέπει να κατεβάσουμε την έκδοση 3.8.10 γιατί σύμφωνα με το https://www.python.org οι εκδόσεις 3.9 δε μπορούν να εκτελεστούν σε windows 7 και καλό θα ήταν να έχουμε μεγαλύτερη συμβατότητα.
Αφού εγκαταστήσουμε την Python εγκαθιστούμε και το PyMongo για τη διασύνδεση της Python με τη MongoDB με την ακόλουθη εντολή:

```
pip install pymongo==3.11.2
```

Το αποτέλεσμα που θα έχουμε είναι στην εικόνα που ακολουθεί, όπου μπορούμε να δούμε ότι γίνεται λήψη και εγκατάσταση του PyMongo:

```
C:\Users\cosmos>pip install pymongo==3.11.2
Collecting pymongo==3.11.2
  Downloading pymongo-3.11.2-cp38-cp38-win_amd64.whl (383 kB)
     |████████████████████████████████| 383 kB 930 kB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.11.2
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the 'c:\users\cosmos\appdata\local\programs\python\python38\python.exe -m pip install
--upgrade pip' command.
```

Εικόνα 40 Εγκατάσταση PyMongo

Για να κάνουμε την πρώτη δοκιμή της Python με τη Mongo ανοίγουμε το IDLE shell της Python και γράφουμε:

```
>>> import pymongo
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> client
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

Οι παραπάνω εντολές εισάγουν τη βιβλιοθήκη της PyMongo στην Python, οπότε μπορούμε να τη χρησιμοποιήσουμε. Στη συνέχεια προσπαθούμε να συνδεθούμε και εμφανίζουμε το αποτέλεσμα της σύνδεσης.
Το επόμενο βήμα είναι να δημιουργήσουμε τη βάση στη Mongo

```
>>> db = client.tweet
>>> db
Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'tweet')
```

Με την πρώτη εντολή δηλώνουμε ότι θέλουμε να επιλέξουμε τη βάση δεδομένων tweet και αν δεν υπάρχει να δημιουργηθεί.
Για να δούμε τα collection στη βάση δεδομένων δίνουμε:

```
db.list_collection_names()
```

Στη συνέχεια καθορίζουμε το collection που θα χρησιμοποιήσουμε, το tweets (db.tweets) και η τιμή που επιστρέφεται είναι ένα instance από το collection:

```
>>> tweets=db.tweets
>>> tweets
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'tweet'), 'tweets')
```

Στη συνέχεια δημιουργούμε τα δεδομένα που θέλουμε να εισάγουμε:

```
tw={
        "title": "My tweet",
        "author": "George",
        "country": "Greece"
        }
```

Για να εισάγουμε το tw στο collection θα γράψουμε:

```
result=tweets.insert_one(tw)
result
<pymongo.results.InsertOneResult object at 0x000001E8B94D2100>
```

Αυτό σημαίνει ότι έχει εισαχθεί το αντικείμενο στη συλλογή που έχουμε επιλέξει. Πώς όμως μπορούμε να εμφανίσουμε τα αποτελέσματα που είναι στη Mongo; Αυτό μπορεί να γίνει με τη βοήθεια της εντολής find της MongoDB και της βιβλιοθήκης pprint για την καλύτερη εμφάνιση των αποτελεσμάτων. Η find επιστρέφει όλα τα περιεχόμενα της συλλογής, περιμένουμε να εμφανίσει ένα, αφού μόνο ένα έχουμε εισάγει.

```
import pprint
>>> for doc in tweets.find():
        pprint.pprint(doc)



{'_id': ObjectId('60b3226ce0fab269127ac755'),
 'author': 'George',
 'country': 'Greece',
 'title': 'My tweet'}
```

Για την άντληση των tweet από την Python, θα χρησιμοποιήσουμε την βιβλιοθήκη της Python Tweepy. Για την εγκατάστασή του θα πρέπει να δώσουμε την ακόλουθη εντολή:

```
pip install tweepy
```

Με τις ακόλουθες εντολές, αποστέλλουμε τα token και τα κλειδιά που έχουμε πάρει κατά την εγγραφή μας στο Twitter και μπορούμε να έχουμε πρόσβαση στο API για αναζήτηση, δημιουργία κ.λπ.

```
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)
```

Για την αναζήτηση των tweets, θα χρησιμοποιήσουμε τη μέθοδο του API, search. Η συγκεκριμένη μέθοδος δέχεται διάφορες παραμέτρους από τις οποίες την q και την lanq. Η q είναι για να εισάγουμε τις λέξεις που θέλουμε να αναζητήσουμε. Το API διαθέτει και ειδικά ερωτήματα τα οποία μπορούμε να τα δώσουμε σαν παράμετρο στο q, εμείς όμως θα χρησιμοποιήσουμε μόνο τη λέξη που θέλουμε να αναζητήσουμε σαν ένα string. Το lang περιορίζει τα αποτελέσματα σε συγκεκριμένη γλώσσα.
Η μέθοδος αυτή επιστρέφει μια συλλογή από tweet που ικανοποιούν το ερώτημα που έχουμε δώσει.
Στη συγκεκριμένη έκδοση του API, δεν υπάρχει η δυνατότητα να αναζητήσουμε tweet με χρήση εύρους ημερομηνιών που να είναι παλαιότερα από μία εβδομάδα. Υπάρχουν δυνατότητες για αναζήτηση στις τελευταίες 30 ημέρες αλλά και αναζήτηση από το 2006, όμως είναι για Premium χρήστες [31].
Τέλος για τη δημιουργία της γραφικής διεπαφής με το χρήστη θα χρησιμοποιήσουμε το Tkinter. Είναι μια βιβλιοθήκη στη Python για τη δημιουργία απλών γραφικών διεπαφών.
Για τη χρήση της θα πρέπει να γίνει import στην Python με τη χρήση της ακόλουθης εντολής:

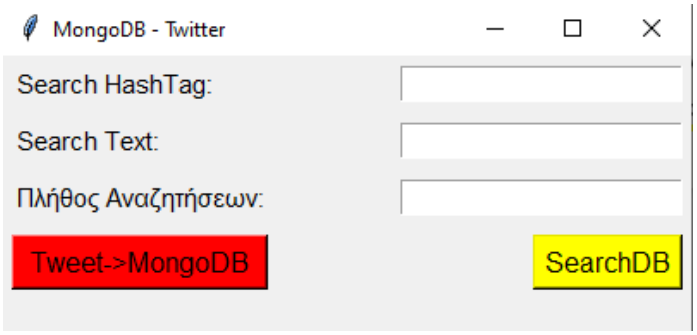
```
import Tkinter
```

Στη συνέχεια μπορούμε να δημιουργήσουμε ένα παράθυρο, δίνοντάς του τίτλο, μέγεθος και προσθέτοντάς του Label, Text field, Button κ.λπ. Επίσης μπορούμε να

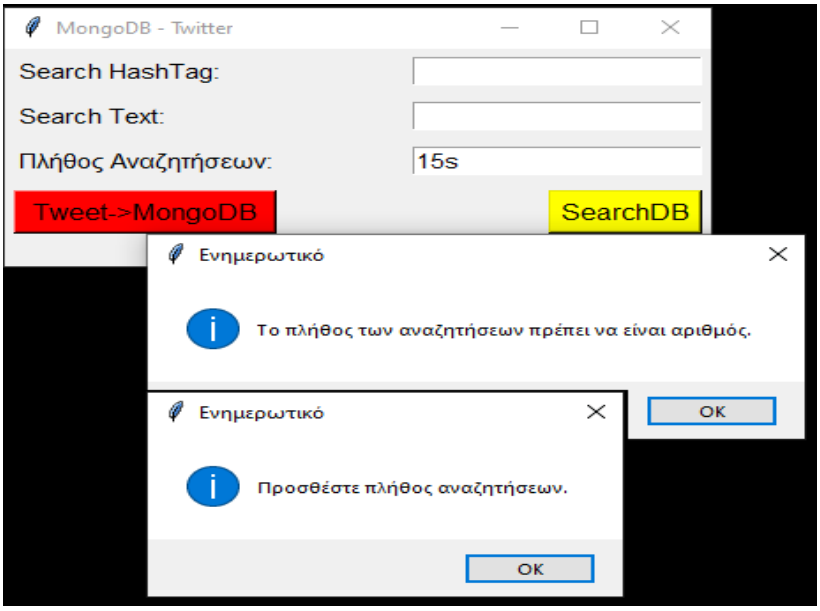
αντιστοιχήσουμε ενέργειες σε button, έτσι ώστε όταν τα πατήσει ο χρήστης να εκτελείται κάποια μέθοδος.

2.4 Δυνατότητες

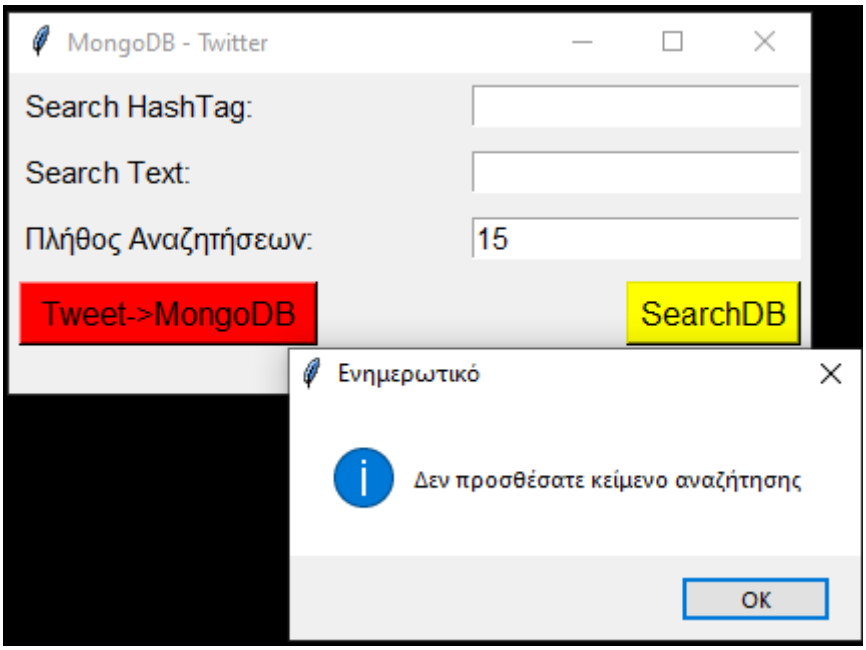
Για την εκτέλεση του προγράμματος, έχοντας εγκαταστήσει την Python στον υπολογιστή, αρκεί ένα διπλό κλικ και θα ανοίξει η γραφική διεπαφή που είναι αρκετά απλή.



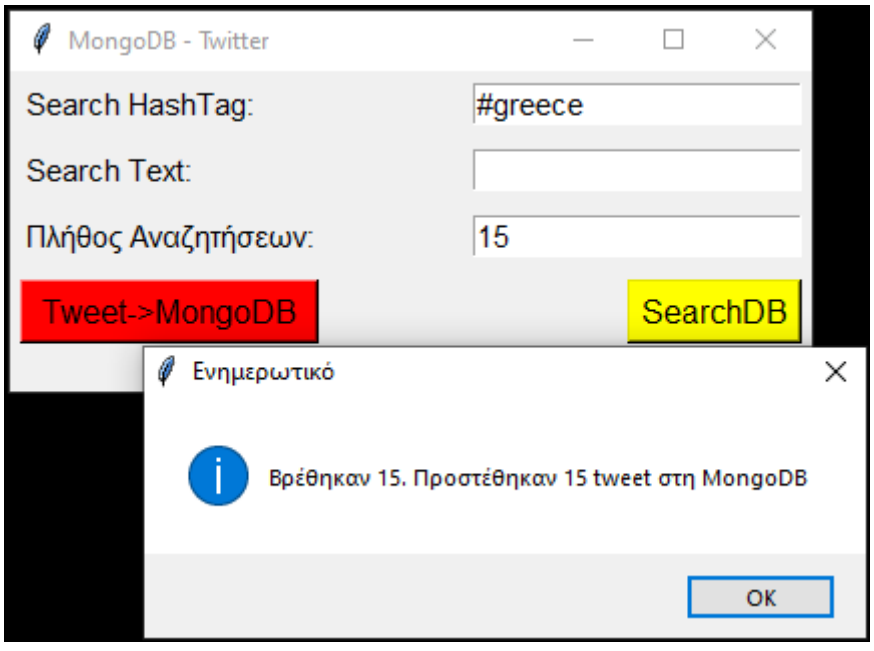
Στην οθόνη αυτή μπορούμε να δούμε ότι στο πεδίο “Search HashTag” μπορούμε να συμπληρώσουμε το hashtag που θέλουμε να αναζητήσουμε στο Tweeter. Το πεδίο “Search Text” λειτουργεί μόνο για αναζήτηση δεδομένων στην βάση. Αν το Πλήθος Αναζητήσεων δεν είναι αριθμός ή είναι κενό τότε θα εμφανιστεί κατάλληλο ενημερωτικό μήνυμα:



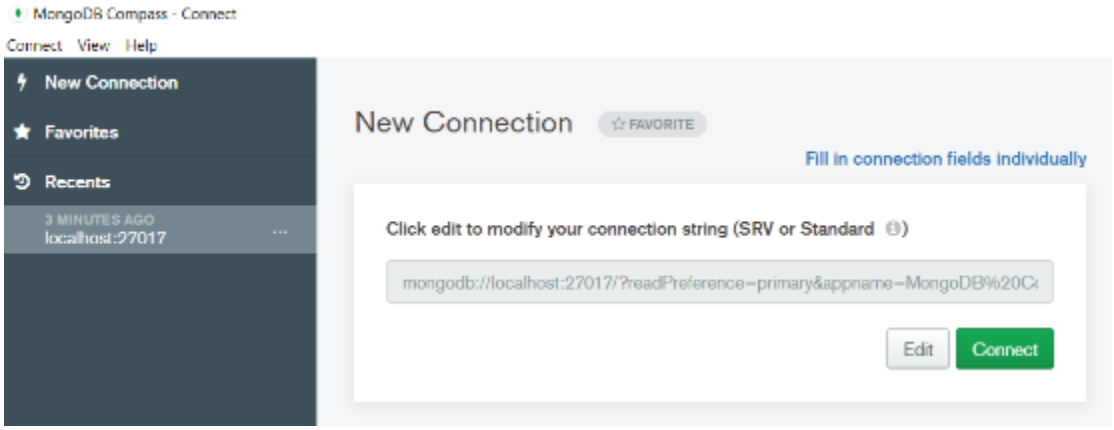
Αντίστοιχο ενημερωτικό μήνυμα θα έχουμε αν δε συμπληρώσουμε το hashtag που θέλουμε να αναζητήσουμε:



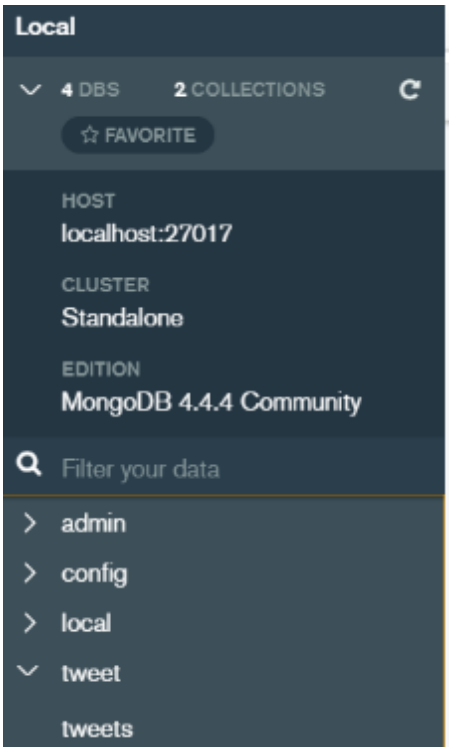
Έστω ότι θέλουμε να αναζητήσουμε το hashtag #greece, αν το συμπληρώσουμε στο πλαίσιο κειμένου και πατήσουμε το button Tweet->MongoDB θα εκτελεστεί μια αναζήτηση για τα tweets της τελευταίας εβδομάδας και θα επιστραφούν 15 τα οποία εάν δεν υπάρχουν ήδη θα εισαχθούν στη βάση δεδομένων, όπως μας λέει και το ενημερωτικό μήνυμα.



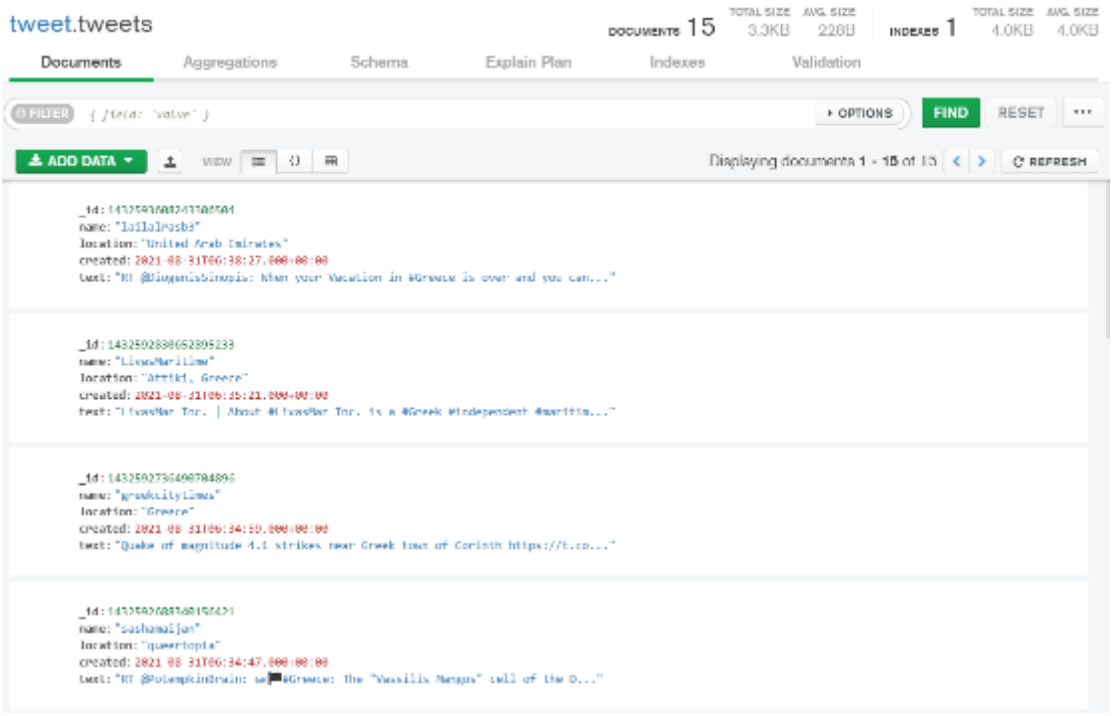
Για να ελέγξουμε ότι έχουν γίνει αυτές οι προσθήκες, μπορούμε να χρησιμοποιήσουμε το MongoDB Compass. Στην αρχική του οθόνη επιλέγουμε localhost για σύνδεση στον τοπικό server και πατάμε το button Connect για να συνδεθούμε.



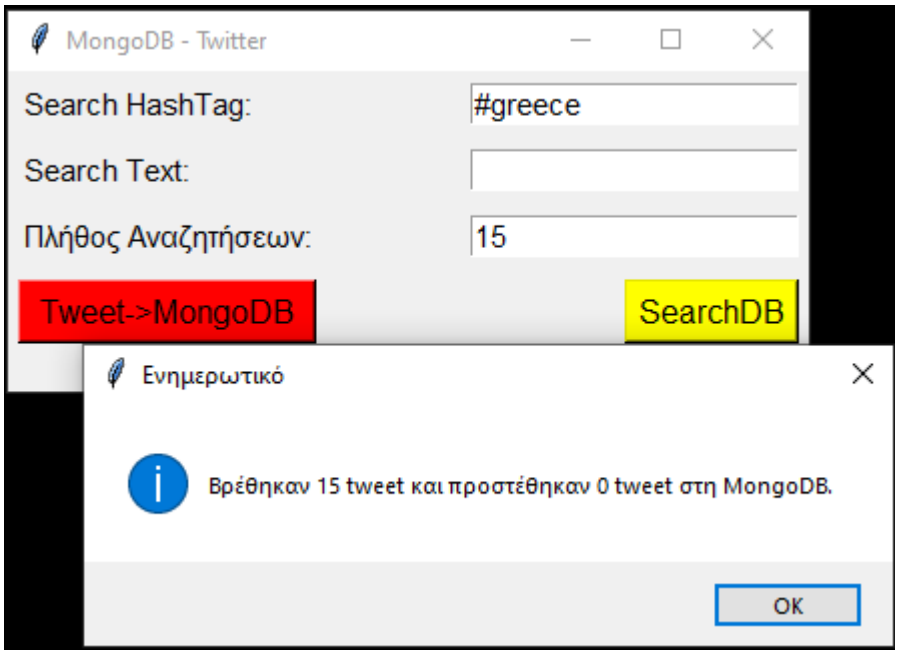
Στην αριστερή πλευρά της οθόνης θα δούμε τις βάσεις δεδομένων που έχουν δημιουργηθεί και αν επιλέξουμε τη βάση δεδομένων tweet θα δούμε και τη συλλογή της tweets.



Επιλέγουμε τη συλλογή tweets και στη δεξιά οθόνη θα εμφανιστούν τα περιεχόμενα της συλλογής που όπως βλέπουμε είναι 15 (στην κορυφή αναφέρει DOCUMENTS 15) και μπορούμε να δούμε και το περιεχόμενό τους.

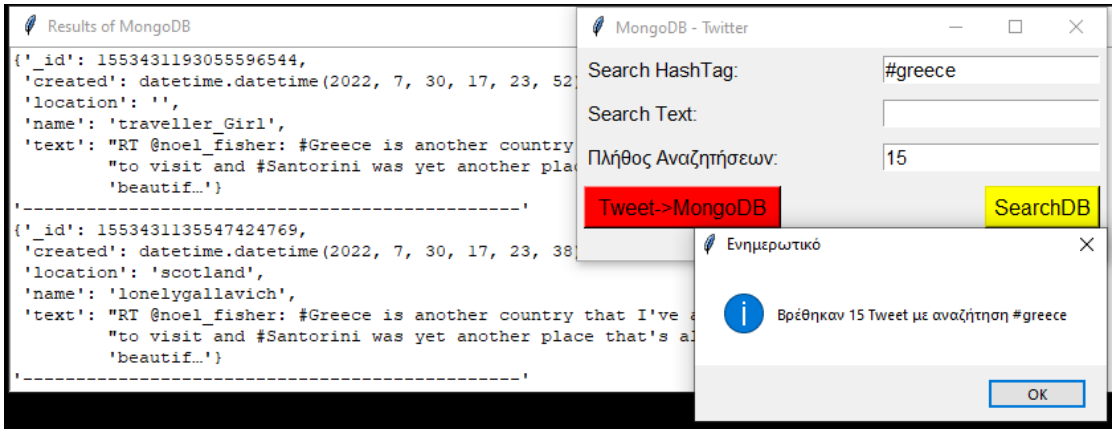


Αν θελήσουμε να αναζητήσουμε ξανά το ίδιο hashtag χωρίς να αλλάξουμε το πλήθος των tweets που θέλουμε να μας επιστραφούν θα δούμε το ακόλουθο μήνυμα:



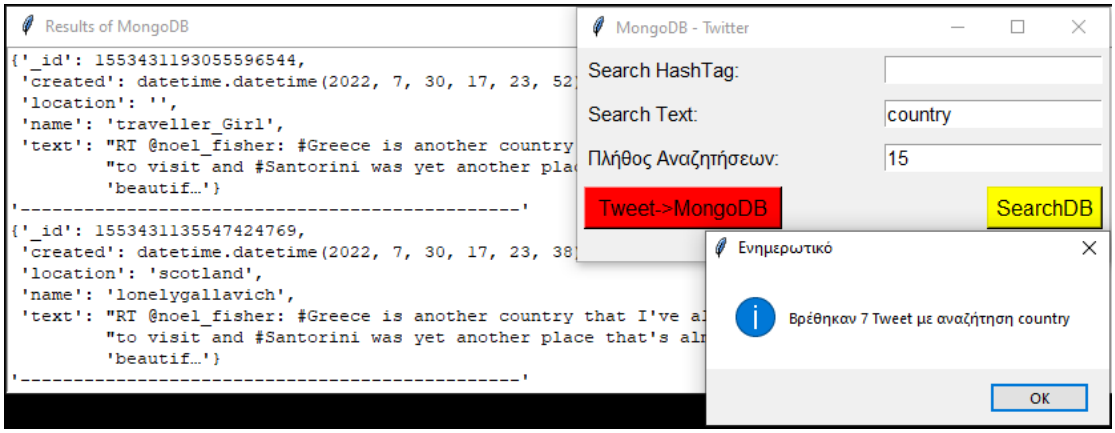
Το οποίο σημαίνει ότι βρέθηκαν από το Twitter 15 tweets αλλά υπήρχαν ήδη στη βάση δεδομένων οπότε δεν τα προσθέσαμε ξανά. Υπενθυμίζουμε ότι ο έλεγχος ύπαρξης γίνεται με το μοναδικό id κάθε tweet το οποίο και εισάγουμε στη βάση δεδομένων. Οπότε πριν εισάγουμε ένα νέο tweet στη MongoDB, ελέγχουμε την ύπαρξή του με την εντολή find και αν δεν υπάρχει το προσθέτουμε, ενώ αν υπάρχει προχωράμε στο επόμενο tweet.

Αν θέλουμε να αναζητήσουμε από τη βάση δεδομένων ένα hashtag (π.χ. το #greece) μπορούμε συμπληρώνοντας το πεδίο “SearchHastag” και το πλήθος αναζητήσεων να πατήσουμε το button SearchDB και θα εμφανιστεί ένα παράθυρο με τα tweets που το περιέχουν.



Αν θέλουμε να κάνουμε αναζήτηση με εύρεση στο κείμενο των tweets μπορούμε να αναζητήσουμε οποιαδήποτε λέξη ή τμήμα λέξης στο πεδίο “Search Text” και αυτό θα επιστραφεί στο ειδικό παράθυρο με τα αποτελέσματα από τα tweets.

Δηλαδή αν γράψουμε τη λέξη country θα δούμε:



Όπως βλέπουμε έχουν επιστραφεί επτά tweets και μπορούμε να δούμε το id του καθενός, την ημερομηνία δημιουργίας του, το χρήστη και την τοποθεσία και τέλος το κείμενο του tweet.

ΚΕΦΑΛΑΙΟ 3. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

3.1 Συμπεράσματα

Ο σκοπός της πτυχιακής ήταν η μελέτη των βάσεων δεδομένων NoSQL και η ανάπτυξη μιας δοκιμαστικής εφαρμογής για τη διασύνδεσή τους με τα μέσα κοινωνικής δικτύωσης. Αρχικά για να μπορέσουμε να αντιληφθούμε τη διαφορά των βάσεων NoSQL από τους άλλους τύπους, μελετήσαμε συστηματικά τις βάσεις δεδομένων και τους τύπους τους. Αντιληφθήκαμε την ανάγκη που οι βάσεις δεδομένων NoSQL ήρθαν να καλύψουν, σε ποιες κατηγορίες χωρίζονται και πώς μπορούν να χρησιμοποιηθούν.
Στη συνέχεια επιλέξαμε από τα μέσα κοινωνικής δικτύωσης το Twitter, ένα μέσο που χρησιμοποιείται από σχεδόν όλο τον κόσμο και περιέχει μια πλούσια θεματολογία. Μελετήσαμε τις δυνατότητες πρόσβασης σε αυτό και επιλέξαμε σα γλώσσα προγραμματισμού την Python για την υλοποίηση της εφαρμογής.
Αναζητήσαμε τις διαθέσιμες βιβλιοθήκες για διασύνδεση τόσο με το Twitter όσο και με τη βάση δεδομένων MonboDB και προχωρήσαμε στην καταγραφή των απαιτήσεων που θα έπρεπε να καλύπτει η εφαρμογή και την υλοποιήσαμε.

3.2 Προτάσεις

Η εφαρμογή που υλοποιήσαμε ήταν κυρίως για τον έλεγχο της διασύνδεσης ενός μέσου κοινωνικής δικτύωσης με μια βάση δεδομένων NoSQL, της δυνατότητας δηλαδή αναζήτησης δεδομένων στο μέσο κοινωνικής δικτύωσης και εισαγωγής/αναζήτησής τους στη βάση δεδομένων. Η εφαρμογή αυτή θα μπορούσε να εξελιχτεί προσθέτοντας τις ακόλουθες δυνατότητες:

- Αναζήτηση επιπλέον πληροφοριών από τα tweets και καταγραφή τους στη βάση δεδομένων.
- Δυνατότητα αναζήτησης στο χρήστη για ένα εύρος ημερομηνιών, tweet χρήστη, περιοχής κ.λπ.

- Δυνατότητα συμπλήρωσης ονόματος χρήστη, αναζήτησης των tweets του και εισαγωγής τους στη βάση δεδομένων.
- Δυνατότητα για διαγραφή tweet από τη βάση δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Akshit J., Reema P., Nishant D., Python: The Most Advanced Programming Language for Computer Science Applications, Διαθέσιμο από https://www.scitepress.org/Papers/2020/103079/103079.pdf

[2] Anna F. (2021) The Ultimate Guide to Accessing & Using APIs , Διαθέσιμο από https://blog.hubspot.com/website/application-programming-interface-api

[3] Berg K., Seymour T. J. (2012) History Of Databases, Διαθέσιμο από https://www.researchgate.net/publication/298332910_History_Of_Databases

[4] Boyd, Golder, & Lotan (2010) Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter , Διαθέσιμο από https://www.danah.org/papers/TweetTweetRetweet.pdf

[5] Brian D. (2022) How Many People Use Twitter in 2022? [New Twitter Stats], Διαθέσιμο από https://backlinko.com/twitter-users

[6] Bruce D. (n.d.) Understanding the Pros and Cons of MongoDB, διαθέσιμο από https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb/

[7] ByteScout, (n.d.) MONGODB HISTORY AND ADVANTAGES, Διαθέσιμο από https://bytescout.com/blog/2019/09/mongodb-history-and-advantages.html

[8] Clint F. (2021) How to Get, Use, & Benefit From Twitter's API, Διαθέσιμο από https://blog.hubspot.com/website/how-to-use-twitter-api

[9] DataFlair (2018) Advantages of MongoDB | Disadvantages of MongoDB, Διαθέσιμο από https://data-flair.training/blogs/advantages-of-mongodb/

[10] Davoudian A., Chen L., Liu M. (2018) A survey on NoSQL stores, Διαθέσιμο από https://www.researchgate.net/publication/324640550_A_survey_on_NoSQL_stores

[11] DeJoy P. (2020) A Short History of MongoDB, Διαθέσιμο από https://petedejoy.com/writing/mongodb

[12] Gupta S.C (2021) SQL vs. NoSQL Database: When to Use, How to Choose, Διαθέσιμο από https://towardsdatascience.com/datastore-choices-sql-vs-nosql-database-ebec24d56106

[13] Humphreys, L. (2011). Who's watching whom? A study of interactive technology and surveillance, Διαθέσιμο από https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=912fe14767ec5c7e6e662d3ddaf55497b3366524

[14] Han J., Haihong E., Le G. & Du J. (2011) Survey on NoSQL database, Διαθέσιμο από https://ieeexplore.ieee.org/document/6106531/

[15] IBM Cloud Education (2019) NoSQL Databases, Διαθέσιμο από https://www.ibm.com/cloud/learn/nosql-databases

[16] JavaTpoint (n.d.) Data Modeling in MongoDB, Διαθέσιμο από https://www.javatpoint.com/mongodb-data-modeling

[17] Kumar B. (2020) CAP Theorem and NoSQL Databases, Διαθέσιμο από https://medium.com/@kumar.barmanand/cap-theorem-and-nosql-databases-589e26e15905

[18] Lithmee, (2019) What is the Difference Between Hierarchical Network and Relational Database Model, Διαθέσιμο από https://pediaa.com/what-is-the-difference-between-hierarchical-network-and-relational-database-model/

[19] Martin G. (2020) The Business Value of API-First Design, Διαθέσιμο από https://auth0.com/blog/the-business-value-of-api-first-design/

[20] MongoDB (n.d.) How to Use Python with MongoDB, Διαθέσιμο από https://www.mongodb.com/languages/python
[21] MongoDB (n.d.) What Is an Object-Oriented Database?, Διαθέσιμο από https://www.mongodb.com/databases/what-is-an-object-oriented-database
[22] Nazrul S.S. (2018) CAP Theorem and Distributed Database Management Systems, Διαθέσιμο από https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e
[23] Oracle (n.d.) What is a Relational Database (RDBMS)?, Διαθέσιμο από https://www.oracle.com/database/what-is-a-relational-database/
[24] Padhy R. P. (2018) Google Spanner:A NewSQL Journey or Beginning of the End of the NoSQL Era, Διαθέσιμο από https://medium.com/rabiprasadpadhy/google-spanner-a-newsql-journey-or-beginning-of-the-end-of-the-nosql-era-3785be8e5c38
[25] Python (n.d), Διαθέσιμο από https://www.python.org/
[26] Programmer (2020) BEST NOSQL DATABASES 2021 – MOST POPULAR AMONG PROGRAMMERS, Διαθέσιμο από https://www.improgrammer.net/most-popular-nosql-database/
[27] Stonebraker, M. (2012) New opportunities for New SQL, Διαθέσιμο από https://15799.courses.cs.cmu.edu/fall2013/static/papers/p10-stonebraker.pdf
[28] StudyTonight (n.d.) Overview of MongoDB, Διαθέσιμο από https://www.studytonight.com/mongodb/overview-of-mongodb
[29] Swaroop C. H. (2005) A Byte of Python, Διαθέσιμο από https://www.ibiblio.org/swaroopch/byteofpython/files/120/byteofpython_120.pdf
[30] Taylor D. (2021) NoSQL Tutorial: Types of NoSQL Databases, What is & Example, Διαθέσιμο από https://www.guru99.com/nosql-tutorial.html
[31] Tweepy (n.d.) API Reference, Διαθέσιμο από https://docs.tweepy.org/en/stable/api.html#API.search
[32] Wasim A. (2015) Using Twitter as a data source: An overview of social media research tools, Διαθέσιμο από https://blogs.lse.ac.uk/impactofsocialsciences/2015/07/10/social-media-research-tools-overview/
[33] Watts S. (2020) ACID Explained: Atomic, Consistent, Isolated & Durable, Διαθέσιμο από https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/
[34] Xtivia (2019) The Pros and Cons of MongoDB, διαθέσιμο από https://www.virtual-dba.com/blog/pros-and-cons-of-mongodb/
[35] Αγγελιδάκης Ν. (2015) Εισαγωγή στον προγραμματισμό με την Python, Διαθέσιμο από http://aggelid.mysch.gr/pythonbook/INTRODUCTION_TO_COMPUTER_PROGRAMMING_WITH_PYTHON.pdf
[36] MongoDB (n.d.) Database Commands, Διαθέσιμο από https://www.mongodb.com/docs/manual/reference/command/#database-commands
[37] MongoDB (n.d.) Operators, Διαθέσιμο από https://www.mongodb.com/docs/manual/reference/operator/

Παράρτημα 1 – Πίνακας Εντολών MongoDB

Παρακάτω παρουσιάζεται ο πίνακας των εντολών τις MongoDB, καθώς και τα Operators. Τα Operators στη MongoDB είναι ειδικές εντολές που εκτελούν λειτουργίες σε ένα έγγραφο ή συλλογή εγγράφων.

I. Aggregation Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

aggregate	Database Command	aggregate	db.runCommand(
			{
			aggregate: "<collection>" || 1, pipeline: [<stage>, <...>],
			explain: <boolean>, allowDiskUse: <boolean>,
			cursor: <doc>,
			maxTimeMS: <int>,
			bypassDocumentValidation: <boolean>,
			readConcern: <doc>,
			collation: <doc>,
			hint: <string or doc>,
			comment: <string>,
			writeConcern: <doc>
			}
				Εκτελεί εργασίες συγκέντρωσης χρησιμοποιώντας το aggregation pipeline το οποίο επιτρέπει στο χρήστη να εκτελεί επεξεργασία δεδομένων από μια εγγραφή ή άλλη πηγή.

	mongosh Command	db.collection.aggregate	db. collectionaggregate([<pipeline>], { <options> })	
		db.aggregate	db.aggregate([<pipeline>], { <options> })	

count	Database Command	count	db.runCommand(
			{
			count: <collection or view>,
			query: <document>,
			limit: <integer>,
			skip: <integer>,
			hint: <hint>,
			readConcern: <document>,
			collation: <document>,
			comment: <any>
			}
			)
				Επιστρέφει τον αριθμό των εγγράφων σε μια συλλογή.

	mongosh Command	db.collection.count	db.collection.count (query, options)	

distinct	Database Command	distinct	db.runCommand(
			{
			distinct: "<collection>",
			key: "<field>",
			query: <query>,
			readConcern: <read concern document>,
			collation: <collation document>,
			comment: <any>
			}
			)
				Επιστρέφει τις διακριτές τιμές που βρέθηκαν για ένα καθορισμένο κλειδί σε μια συλλογή ή μια προβολή.

	mongosh Command	db.collection.distinct	db.collection.distinct(field, query, options)	

mapReduce	Database Command	mapReduce	db.runCommand(
			{
			mapReduce: <string>,
			map: <string or JavaScript>,
			reduce: <string or JavaScript>,
			finalize: <string or JavaScript>,
			out: <output>,
			query: <document>,
			sort: <document>,
			limit: <number>,
			scope: <document>,
			jsMode: <boolean>,
			verbose: <boolean>,
			bypassDocumentValidation: <boolean>,
			collation: <document>,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Επιτρέπει την εκτέλεση επεξεργασίας και ανάλυσης δεδομένων σε μεγάλες συλλογές δεδομένων και την παραγωγή στατιστικών και αναφορών που μπορούν να χρησιμοποιηθούν για περαιτέρω ανάλυση.

	mongosh Command	db.collection.mapReduce	db.collection.mapReduce(
			<map>,
			<reduce>,
			{
			out: <collection>,
			query: <document>,
			sort: <document>,
			limit: <number>,
			finalize: <function>,
			scope: <document>,
			jsMode: <boolean>,
			verbose: <boolean>,
			bypassDocumentValidation:
			<boolean>
			}
			)	

II. Geospatial Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

geoSearch	Database Command		db.runCommand(
			{
			geoSearch : "<collection>",
			search: <filter documents>,
			near : <Coordinates of a point>,
			maxDistance: <Max distance>,
			limit : <Max docs number>,
			readConcern: <read concern document>,
			comment: <any>
			}
			)
				Εκτελεί μια γεωχωρική ερώτηση που χρησιμοποιεί τη λειτουργία δείκτη στοίβας της MongoDB. Η λειτουργία δείκτη στοίβας (haystack index) επιτρέπει στο σύστημα να βρίσκει γρήγορα τα στοιχεία που βρίσκονται κοντά σε μια καθορισμένη γεωγραφική τοποθεσία. Έχει καταργηθεί στη MongoDB 5.0.

III. Query and Write Operation Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

delete	Database Command	delete	db.runCommand(
			{
			delete: <collection>,
			deletes: [
			{
			q : <query>,
			limit : <integer>,
			collation: <document>,
			hint: <document|string>
			},
			...
			],
			comment: <any>,
			let: <document>, // Added in MongoDB 5.0
			ordered: <boolean>,
			writeConcern: { <write concern> }
			}
			)
				Διαγράφει έγγραφα από μια συλλογή.

	mongosh Command	db.collection.deleteOne	db.collection.deleteOne(
			<filter>,
			{
			writeConcern: <document>,
			collation: <document>,
			hint: <document|string> // Available starting in MongoDB 4.4
			}
			)
				Διαγράφει ένα έγγραφο.

	mongosh Command	db.collection.deleteMany	db.collection.deleteMany(
			<filter>,
			{
			writeConcern: <document>,
			collation: <document>
			}
			)
				Διαγράφει πολλά έγγραφα.

	mongosh Command	db.collection.findOneAndDelete	db.collection.findOneAndDelete(
			<filter>,
			{
			writeConcern: <document>,
			projection: <document>,
			sort: <document>,
			maxTimeMS: <number>,
			collation: <document>
			}
			)
				Διαγράφει ένα μεμονωμένο έγγραφο με βάση τα κριτήρια filter και sort, επιστρέφοντας το διαγραμμένο έγγραφο.

find	Database Command	find	db.runCommand({ find: <string>, filter: <document>, sort: <document>, projection: <document>, hint: <document or string>, skip: <int>, limit: <int>, batchSize: <int>, singleBatch: <bool>, comment: <any>,	Εκτελεί μια ερώτηση και επιστρέφει το πρώτο σύνολο αποτελεσμάτων και το αναγνωριστικό του cursor, από το οποίο ο πελάτης μπορεί να δημιουργήσει έναν cursor.

			maxTimeMS: <int>, readConcern: <document>, max: <document>, min: <document>, returnKey: <bool>, showRecordId: <bool>, tailable: <bool>, oplogReplay: <bool>, noCursorTimeout: <bool>, awaitData: <bool>, allowPartialResults: <bool>, collation: <document>, allowDiskUse : <bool>, let: <document> // Added in MongoDB 5.0 })	
	mongosh Command	db.collection.find	db.collection.find(query, projection, options)	Επιλέγει έγγραφα σε μια συλλογή ή προβολή και επιστρέφει έναν cursor στα επιλεγμένα έγγραφα.
	mongosh Command	db.collection.findOne	db.collection.findOne(query, projection, options)	Επιστρέφει ένα έγγραφο που ικανοποιεί τα καθορισμένα κριτήρια ερωτήματος στη συλλογή ή την προβολή . Εάν πολλά έγγραφα ικανοποιούν το ερώτημα, αυτή η μέθοδος επιστρέφει το πρώτο έγγραφο σύμφωνα με τη φυσική σειρά που αντικατοπτρίζει τη σειρά των εγγράφων στο δίσκο
findAndModify	Database Command	findAndModify	db.runCommand({ findAndModify: <collection-name>, query: <document>, sort: <document>, remove: <boolean>, update: <document or aggregation pipeline>, new: <boolean>, fields: <document>, upsert: <boolean>, bypassDocumentValidation: <boolean>, writeConcern: <document>, collation: <document>, arrayFilters: <array>, hint: <document|string>,	Τροποποιεί και επιστρέφει ένα μεμονωμένο έγγραφο.

			comment: <any>, let: <document> // Added in MongoDB 5.0 })	
	mongosh Command	db.collection.findAndModify	db.collection.findAndModify({ query: <document>, sort: <document>, remove: <boolean>, update: <document or aggregation pipeline>, // Changed in MongoDB 4.2 new: <boolean>, fields: <document>, upsert: <boolean>, bypassDocumentValidation: <boolean>, writeConcern: <document>, collation: <document>, arrayFilters: [<filterdocument1>, ...], let: <document> // Added in MongoDB 5.0 });	Τροποποιεί και επιστρέφει ένα μεμονωμένο έγγραφο.
getLastError	Database Command	getLastError	db.runCommand({ getLastError: 1, <option1>: <value>, <option2>: <value>, ... })	Επιστρέφει το τελευταίο σφάλμα που συνέβη στην τελευταία ενέργεια εγγραφής στην MongoDB
	mongosh Command	db.getLastError	db.getLastError()	Επιστρέφει το τελευταίο σφάλμα που συνέβη στην τελευταία ενέργεια εγγραφής στην MongoDB
	mongosh Command	db.getLastErrorObj	db.getLastErrorObj()	Επιστρέφει ένα αντικείμενο που περιέχει πληροφορίες σχετικά με το τελευταίο σφάλμα που συνέβη στην τελευταία ενέργεια εγγραφής στην MongoDB.
	mongosh Command	db.getPrevError	db.getPrevError()	Επιστρέφει το προηγούμενο σφάλμα που συνέβη στη βάση δεδομένων του MongoDB.
getMore	Database Command	getMore	db.runCommand({ getMore: <long>, collection: <string>, batchSize: <int>, maxTimeMS: <int>,	Χρησιμοποιείται σε συνδυασμό με εντολές που επιστρέφουν έναν cursor σε έγγραφα, όπως η

			comment: <any> })	εντολή find και aggregate, για να επιστρέψει τα επόμενα πακέτα εγγραφών που δείχνει επί του παρόντος ο cursor.

insert	Database Command	insert	db.runCommand(
			{
			insert: <collection>,
			documents: [<document>, <document>, <document>, ...],
			ordered: <boolean>,
			writeConcern: { <write concern> },
			bypassDocumentValidation: <boolean>,
			comment: <any>
			}
			)
				Εισάγει ένα ή περισσότερα έγγραφα και επιστρέφει ένα έγγραφο που περιέχει την κατάσταση όλων των εισαχθέντων εγγράφων.

	mongosh Command	db.collection.insertOne	db.collection.insertOne(
			<document>,
			{
			writeConcern: <document>
			}
			)
				Εισάγει ένα μεμονωμένο έγγραφο σε μια συλλογή.

	mongosh Command	db.collection.insertMany	db.collection.insertMany(
			[<document 1> , <document 2>, ...],
			{
			writeConcern: <document>,
			ordered: <boolean>
			}
			)
				Εισάγει πολλά έγγραφα σε μια συλλογή.

update	Database Command	update	db.runCommand(
			{
			update: <collection>,
			updates: [
			{
			q: <query>,
			u: <document or pipeline>,
			c: <document>, // Added in MongoDB 5.0
			upsert: <boolean>,
			multi: <boolean>,
			collation: <document>,
			arrayFilters: <array>,
			hint: <document|string>
			},
			...
			],
			ordered: <boolean>,
			writeConcern: { <write concern> },
			bypassDocumentValidation: <boolean>,
			comment: <any>,
			let: <document> // Added in MongoDB 5.0
			}
			)
				Ενημερώνει ένα ή περισσότερα έγγραφα.

	mongosh Command	db.collection.updateOne	db.collection.updateOne(<filter>, <update>, { upsert: <boolean>,	Ενημερώνει ένα μεμονωμένο έγγραφο.

			writeConcern: <document>, collation: <document>, arrayFilters: [<filterdocument1>, ...], hint: <document|string> // Available starting in MongoDB 4.2.1 })	

	mongosh Command	db.collection.updateMany	db.collection.updateMany(
			<filter>,
			<update>,
			{
			upsert: <boolean>,
			writeConcern: <document>,
			collation: <document>,
			arrayFilters: [<filterdocument1>, ...],
			hint: <document|string> // Available starting in MongoDB 4.2.1
			}
			)
				Ενημερώνει όλα τα έγγραφα που ταιριάζουν με το καθορισμένο φίλτρο για μια συλλογή.

	mongosh Command	db.collection.replaceOne	db.collection.replaceOne(
			<filter>,
			<replacement>,
			{
			upsert: <boolean>,
			writeConcern: <document>,
			collation: <document>,
			hint: <document|string> // Available starting in 4.2.1
			}
			)
				Αντικαθιστά ένα μεμονωμένο έγγραφο στη συλλογή με βάση το φίλτρο.

	mongosh Command	db.collection.findOneAndReplace	db.collection.findOneAndReplace(
			<filter>,
			<replacement>,
			{
			projection: <document>,
			sort: <document>,
			maxTimeMS: <number>,
			upsert: <boolean>,
			returnDocument: <string>,
			returnNewDocument: <boolean>,
			collation: <document>
			}
			)	

	mongosh Command	db.collection.findOneAndUpdate	db.collection.findOneAndUpdate(
			<filter>,
			<update document or aggregation pipeline>, // Changed in MongoDB 4.2
			{
			projection: <document>,
			sort: <document>,
			maxTimeMS: <number>,
			upsert: <boolean>,
			returnDocument: <string>,
			returnNewDocument: <boolean>,
			collation: <document>,
			arrayFilters: [<filterdocument1>, ...]
			}
			)
				Ενημερώνει ένα μεμονωμένο έγγραφο με βάση τα κριτήρια φίλτρου και ταξινόμησης.

IV. Query Plan Cache Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

planCacheClear	Database Command	planCacheClear	db.runCommand(
			{
			planCacheClear: <collection>,
			query: <query>,
			sort: <sort>,
			projection: <projection>,
			comment: <any>
			}
			)
				Αδειάζει τη μνήμη της προσωρινής αποθήκευσης.

	mongosh Command	getPlanCache .clear	db.collection.getPlanCache().clear()	
	mongosh Command	PlanCache.clearPlansByQuery	db.collection.getPlanCache() .clearPlansByQuery(<query>, <projection>, <sort>)	Αδειάσει τη μνήμη της προσωρινής αποθήκευσης για ένα συγκεκριμένο ερώτημα.

planCacheClearFilters	Database Command		db.runCommand(
			{
			planCacheClearFilters: <collection>,
			query: <query pattern>,
			sort: <sort specification>,
			projection: <projection specification>,
			collation: { <collation> },
			comment: <any>
			}
			)
				Αναιρεί τυχόν φίλτρα που έχουν οριστεί στην προσωρινή αποθήκευση.

planCacheListFilters	Database Command	planCacheListFilters	db.runCommand(
			{
			planCacheListFilters: <collection>
			}
			)
				Επιστρέψει τα φίλτρα που έχουν οριστεί στην προσωρινή αποθήκευση.

planCacheSetFilter	Database Command	planCacheSetFilter	db.runCommand(
			{
			planCacheSetFilter: <collection>,
			query: <query>,
			sort: <sort>,
			projection: <projection>,
			collation: { <collation> },
			indexes: [<index1>, <index2>, ...],
			comment: <any>
			}
			)
				Ορίζει ένα φίλτρο ευρετηρίου για μια συλλογή. Εάν υπάρχει ήδη ένα φίλτρο ευρετηρίου για το σχήμα ερωτήματος, η εντολή αντικαθιστά το προηγούμενο φίλτρο ευρετηρίου.

V. Authentication Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**
	mongosh Command	db.auth	db.auth(<username>, <password>)	Επιτρέπει σε ένα χρήστη τον έλεγχο ταυτότητας στη βάση δεδομένων.

logout	Database Command	logout	db.runCommand(
			{
			logout: 1
			}
			)
				Επιστρέφει ένα μήνυμα σφάλματος στο αρχείο καταγραφής ανά προσπάθεια αποσύνδεσης.

VI. User Management Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

createUser	Database Command	createUser	db.runCommand(
			{
			createUser: "<name>",
			pwd: passwordPrompt(), // Or "<cleartext password>"
			customData: { <any information> },
			roles: [
			{ role: "<role>", db: "<database>" } | "<role>", ...
			],
			writeConcern: { <write concern> },
			authenticationRestrictions: [
			{ clientSource: ["<IP|CIDR range>", ...], serverAddress: ["<IP|CIDR range>", ...] },
			...
			],
			mechanisms: ["<scram-mechanism>", ...], //Available starting in MongoDB 4.0
			digestPassword: <boolean>,
			comment: <any>
			}
			)
				Δημιουργεί έναν νέο χρήστη στη βάση δεδομένων όπου εκτελείτε την εντολή.

	mongosh Command	db.createUser	db.createUser("<user>", { <writeConcern> })	

dropAllUsersFromDatabase	Database Command	dropAllUsersFromDatabase	db.runCommand(
			{
			dropAllUsersFromDatabase: 1,
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Καταργεί όλους τους χρήστες από τη βάση δεδομένων στην οποία εκτελείτε την εντολή.

	mongosh Command	db.dropAllUsers	db.dropAllUsers(<writeConcern>)	

dropUser	Database Command	dropUser	db.runCommand(
			{
			dropUser: "<user>",
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Καταργεί έναν μόνο χρήστη από τη βάση δεδομένων στην οποία εκτελείτε την εντολή.

	mongosh Command	db.dropUser	db.dropUser("<username>", { <writeConcern> })	

grantRolesToUser	Database Command	grantRolesToUser	db.runCommand(
			{
			grantRolesToUser: "<user>",
			roles: [<roles>],
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Παραχωρεί πρόσθετους ρόλους σε έναν χρήστη.

	mongosh Command	db.grantRolesToUser()	db.grantRolesToUser("<username>", [<roles>], { <writeConcern> })	
revokeRolesFromUser	Database Command	revokeRolesFromUser	db.runCommand({	Καταργεί έναν ή περισσότερους

			revokeRolesFromUser: "<user>", roles: [{ role: "<role>", db: "<database>" } | "<role>", ...], writeConcern: { <write concern> }, comment: <any> })	ρόλους από έναν χρήστη στη βάση δεδομένων όπου υπάρχουν οι ρόλοι.
	mongosh Command	db.revokeRolesFromUser	db.revokeRolesFromUser("<username>", [<roles>], { <writeConcern> })	
updateUser	Database Command	updateUser	db.runCommand({ updateUser: "<username>", pwd: passwordPrompt(), // Or "<cleartext password>" customData: { <any information> }, roles: [{ role: "<role>", db: "<database>" } | "<role>", ...], authenticationRestrictions: [{ clientSource: ["<IP>"	"<CIDR range>", ...], serverAddress: ["<IP>",	"<CIDR range>", ...] }, ...], mechanisms: ["<scram-mechanism>", ...], digestPassword: <boolean>, writeConcern: { <write concern> }, comment: <any> })	Ενημερώνει το προφίλ του χρήστη στη βάση δεδομένων στην οποία εκτελείτε την εντολή.

usersInfo	Database Command	usersInfo	db.runCommand(
			{
			usersInfo: <various>,
			showCredentials: <Boolean>,
			showCustomData: <Boolean>,
			showPrivileges: <Boolean>,
			showAuthenticationRestrictions: <Boolean>,
			filter: <document>,
			comment: <any>
			}
			)
				Επιστρέφει πληροφορίες για έναν ή περισσότερους χρήστες.

VII. Role Management Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

createRole	Database Command	createRole	db.adminCommand(
			{
			createRole: "<new role>",
			privileges: [
			{ resource: { <resource> }, actions: ["<action>", ...] }, ...
			],
			roles: [
			{ role: "<role>", db: "<database>" } | "<role>", ...
			],
			authenticationRestrictions: [
			{
			clientSource: ["<IP>" | "<CIDR range>", ...],
			serverAddress: ["<IP>" | "<CIDR range>", ...]
			}, ...
			],
			writeConcern: <write concern document>,
			comment: <any>
			}
			)
				Δημιουργεί έναν ρόλο και καθορίζει τα προνόμιά του . Ο ρόλος ισχύει για τη βάση δεδομένων στην οποία εκτελείτε την εντολή.

	mongosh Command	db.createRole	db.createRole(
			{
			role: "<name>",
			privileges: [
			{ resource: { <resource> }, actions: ["<action>", ...] }, ...
			],
			roles: [
			{ role: "<role>", db: "<database>" } | "<role>", ...
			],
			authenticationRestrictions: [
			{
			clientSource: ["<IP>" | "<CIDR range>", ...],
			serverAddress: ["<IP>" | "<CIDR range>", ...]
			}, ...
			]
			}	

dropRole	Database Command	dropRole	db.runCommand(
			{
			dropRole: "<role>",
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Διαγράφει έναν ρόλο που ορίζεται από το χρήστη στη τη βάση δεδομένων στην οποία εκτελείτε την εντολή.

	mongosh Command	db.dropRole	use products db.dropRole(" rolename ", { writeConcern " })	

dropAllRolesFromDatabase	Database Command	dropAllRolesFromDatabase	db.runCommand(
			{
			dropAllRolesFromDatabase: 1,
			writeConcern: { <write concern> },
			comment: <any>
			}
				Διαγράφει όλους τους ρόλους που ορίζονται από το χρήστη στη βάση δεδομένων

			)	όπου εκτελείτε την εντολή.
	mongosh Command	db.dropAllRoles	db.dropAllRoles({ writeConcern })	

grantPrivilegesToRole	Database Command	grantPrivilegesToRole	db.runCommand(
			{
			grantPrivilegesToRole: "<role>",
			privileges: [
			{
			resource: { <resource> }, actions: [
			"<action>", ...]
			},...
			],
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Εκχωρεί πρόσθετα δικαιώματα σε έναν ρόλο που ορίζεται από το χρήστη και ορίζεται στη βάση δεδομένων στην οποία εκτελείται η εντολή.

	mongosh Command	db.grantPrivilegesToRole	db.grantPrivilegesToRole("< rolename >", [{ resource: { <resource> }, actions: ["<action>", ...] }, ...], { < writeConcern > })	

grantRolesToRole	Database Command	grantRolesToRole	db.runCommand(
			{
			grantRolesToRole: "<role>",
			roles: [
			{ role: "<role>", db: "<database>" },
			...
			],
			writeConcern: { <write concern> },
			comment: <any>
			}
			)
				Παραχωρεί ρόλους σε ρόλο που ορίζεται από το χρήστη .

	mongosh Command	db.grantRolesToRole	db.grantRolesToRole("<rolename>", [<roles>], { <writeConcern> })	

invalidateUserCache	Database Command	invalidateUserCache	db.runCommand(
			{
			invalidateUserCache: 1
			}
			)
				Καθαρίζει την προσωρινή μνήμη των πληροφοριών χρήστη, συμπεριλαμβανομένων των διαπιστευτηρίων και των ρόλων.

revokePrivilegesFromRole	Database Command	revokePrivilegesFromRole	db.runCommand(
			{
			revokePrivilegesFromRole: "<role>",
			privileges: [
			{ resource: { <resource> }, actions: [
			"<action>", ...] }, ...
			],
			writeConcern: <write concern document>,
			comment: <any>
			}
			)
				Αφαιρεί τα καθορισμένα προνόμια από τον ρόλο που καθορίζεται από τον χρήστη στη βάση δεδομένων όπου εκτελείται η εντολή.

	mongosh Command	db.revokePrivilegesFromRole	db.revokePrivilegesFromRole("<rolename>", [	

			{ resource: { <resource> }, actions: ["<action>", ...] }, ...], { <writeConcern> })	
revokeRolesFromRole	Database Command	revokeRolesFromRole	db.runCommand({ revokeRolesFromRole: "<role>", roles: [{ role: "<role>", db: "<database>" } | "<role>", ...], writeConcern: { <write concern> }, comment: <any> })	Καταργεί τους καθορισμένους κληρονομημένους ρόλους από έναν ρόλο.
	mongosh Command	db.revokeRolesFromRole	db.revokeRolesFromRole("<rolename>", [<roles>], { <writeConcern> })	
rolesInfo	Database Command	rolesInfo	db.runCommand({ rolesInfo: { role: <name>, db: <db> }, showAuthenticationRestrictions: <Boolean>, showBuiltinRoles: <Boolean>, showPrivileges: <Boolean>, comment: <any> })	Επιστρέφει πληροφορίες κληρονομικότητας και προνομίων για καθορισμένους ρόλους, συμπεριλαμβανομένων τόσο των ρόλων που ορίζονται από το χρήστη όσο και των ενσωματωμένων ρόλων .
updateRole	Database Command	updateRole	db.runCommand({ updateRole: "<role>", privileges: [{ resource: { <resource> }, actions: ["<action>", ...] }, ...], roles: [{ role: "<role>", db: "<database>" } | "<role>", ...], authenticationRestrictions: [{ clientSource: ["<IP>" | "<CIDR range>", ...], serverAddress: ["<IP>", ...] }, ...] writeConcern: <write concern document>, comment: <any> })	Ενημερώνει έναν ρόλο που ορίζεται από το χρήστη .
	mongosh Command	db.updateRole	db.updateRole("<rolename>",	

			{ privileges: [{ resource: { <resource> }, actions: ["<action>", ...] }, ...], roles: [{ role: "<role>", db: "<database>" } | "<role>", ...], authenticationRestrictions: [{ clientSource: ["<IP>" | "<CIDR range>", ...], serverAddress: ["<IP>", | "<CIDR range>", ...] }, ...] }, { <writeConcern> })	

VIII. Replication Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**
applyOps	Database Command	applyOps	db.runCommand({ applyOps: [<operations>] })	Εσωτερική εντολή που εφαρμόζει τις εγγραφές oplog στο τρέχον σύνολο δεδομένων.

hello	Database Command	hello	db.runCommand(
			{
			hello: 1
			}
			)
				Επιστρέφει ένα έγγραφο που περιγράφει το ρόλο του mongod server.

	mongosh Command	db.hello	db.hello()	

replSetAbortPrimaryCatchUp	Database Command	replSetAbortPrimaryCatchUp	db.runCommand(
			{
			replSetAbortPrimaryCatchUp: 1
			}
			)
				Αναγκάζει το κύριο μέλος του συνόλου αντιγράφων να διακόψει τον συγχρονισμό και στη συνέχεια να ολοκληρώσει τη μετάβαση σε κύριο μέλος

replSetFreeze	Database Command	replSetFreeze	db.runCommand(
			{
			replSetFreeze: <seconds>
			}
			)
				Παγώνει προσωρινά το σύνολο αντιγράφων (replica set) και να αποτρέπει τη διανομή εγγραφών σε όλα τα μέλη του συνόλου αντιγράφων

	mongosh Command	rs.freeze	rs.freeze()	

replSetGetConfig	Database Command	replSetGetConfig	db.adminCommand(
			{
			replSetGetConfig: 1,
			commitmentStatus: <boolean>,
			comment: <any>
			}
			)
				Επιστρέφει ένα έγγραφο που περιγράφει την τρέχουσα διαμόρφωση του συνόλου αντιγράφων .

	mongosh Command	rs.config	rs.config()	

replSetGetStatus	Database Command	replSetGetStatus	db.adminCommand(
			{
			replSetGetStatus: 1
			}
			)
				Επιστρέφει την κατάσταση του συνόλου αντιγράφων από την οπτική γωνία του διακομιστή που επεξεργάστηκε την εντολή

	mongosh Command	rs.status	rs.status()	
replSetInitiate	Database Command	replSetInitiate	db.runCommand({ replSetInitiate : <config_document>	Αρχικοποιεί ένα νέο σύνολο αντιγράφων

			})	
	mongosh Command	r s.initiate	r s.initiate()	

replSetMaintenance	Database Command	replSetMaintenance	db.runCommand(
			{
			replSetMaintenance: <boolean>
			}
			)
				Ενεργοποιεί ή απενεργοποιεί τη λειτουργία συντήρησης για ένα δευτερεύον μέλος ενός συνόλου αντιγράφων.

replSetReconfig	Database Command	replSetReconfig	db.adminCommand(
			{
			replSetReconfig: <new_config_document>,
			force: <boolean>,
			maxTimeMS: <int>
			}
			)
				Εφαρμόζει μια νέα διαμόρφωση σε ένα υπάρχον σύνολο αντιγράφων.

	mongosh Command	rs.reconfig	rs.reconfig(
			<configuration>,
			{
			"force" : <boolean>,
			"maxTimeMS" : <int>
			}
			)	

replSetResizeOplog	Database Command	replSetResizeOplog	db.adminCommand(
			{
			replSetResizeOplog: <int>,
			size: <double>,
			minRetentionHours: <double>
			}
			)
				Αλλάζει δυναμικά το μέγεθος του oplog για ένα σύνολο αντιγράφων. Διατίθεται μόνο για τη μηχανή αποθήκευσης WiredTiger.

replSetStepDown	Database Command	replSetStepDown	db.adminCommand(
			{
			replSetStepDown: <seconds>,
			secondaryCatchUpPeriodSecs: <seconds>,
			force: <true|false>
			}
			)
				Αναγκάζει τον τρέχοντα primary να αποχωρήσει από το ρόλο του και να γίνει secondary, προκαλώντας έτσι μια εκλογή.

	mongosh Command	rs.stepDown	rs.stepDown(stepDownSecs, secondaryCatchUpPeriodSecs)	

replSetSyncFrom	Database Command	replSetSyncFrom	db.adminCommand(
			{
			replSetSyncFrom: "hostname<:port>"
			}
			)
				Παράκαμψη ρητού τρόπου επιλογής ενός μέλους για αντιγραφή δεδομένων.

	mongosh Command	rs.syncFrom	rs.syncFrom()	

IX. Sharding Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

abortReshardCollection	Database Command	abortReshardCollection	db.adminCommand(
			{
			abortReshardCollection: "<database>.<collection>"
			}
			)
				Ματαιώνει μια λειτουργία αναδιαμοιρασμού .

	mongosh Command	sh.abortReshardCollection	sh.abortReshardCollection(<namespace>)	

addShard	Database Command	addShard	db.adminCommand(
			{
			addShard: "<replica_set>/<hostname><:port>",
			maxSize: <size>,
			name: "<shard_name>"
			}
			)
				Προσθέτει ένα shard σε ένα sharded cluster.

	mongosh Command	sh.addShard(<url>)	sh.addShard("<replica_set>/<hostname><:port>")	

addShardToZone	Database Command	addShardToZone	db.adminCommand(
			{
			addShardToZone: <string>,
			zone: <string>
			}
			)
				Συσχετίζει ένα shard με μια ζώνη.

	mongosh Command	sh.addShardToZone	sh.addShardToZone(shard, zone)	

balancerCollectionStatus	Database Command	balancerCollectionStatus	db.adminCommand(
			{
			balancerCollectionStatus: "<db>.<collection>"
			}
			)
				Επιστρέφει πληροφορίες σχετικά με το εάν τα τμήματα μιας διαμερισμένης συλλογής είναι ισορροπημένα

	mongosh Command	sh.balancerCollectionStatus	sh.balancerCollectionStatus(<namespace>)	

balancerStart	Database Command	balancerStart	db.adminCommand(
			{
			balancerStart: 1,
			maxTimeMS: <number>
			}
			)
				Ξεκινά το balancer thread.

	mongosh Command	sh.startBalancer	sh.startBalancer(<timeout>, <interval>)	

balancerStatus	Database Command	balancerStatus	db.adminCommand(
			{
			balancerStatus: 1
			}
			)
				Επιστρέφει ένα έγγραφο που περιέχει πληροφορίες σχετικά με την κατάσταση του εξισορροπητή.

	mongosh Command	sh.isBalancerRunning	> sh.isBalancerRunning()	

balancerStop	Database Command	balancerStop	db.adminCommand(
			{
			balancerStop: 1,
			maxTimeMS: <number>
			}
			)
				Απενεργοποιεί τον ισορροπητή σε ένα sharded cluster.

	mongosh Command	sh.stopBalancer	sh.stopBalancer()	

clearJumboFlag	Database Command	clearJumboFlag	db.adminCommand(
			{
			clearJumboFlag: "<database>.<collection>",
			bounds: <array>
			}
			)
			Ή
			db.adminCommand(
			{
			clearJumboFlag: "<database>.<collection>",
			find: <query>
			}
			)
				Καθαρίζει το jumbo flag για ένα chunk.

cleanupOrphaned	Database Command	cleanupOrphaned	db.runCommand(
			{
			cleanupOrphaned: "<database>.<collection>",
			startingFromKey: <minimumShardKeyValue>, // deprecated
			secondaryThrottle: <boolean>, // deprecated
			writeConcern: <document> // deprecated
			}
			)
				Αφαιρεί τα δεδομένα που δεν ανήκουν σε κάποιο chunk ενός shard και έχουν τιμές κλειδιού κατανομής εκτός των εύρων που καλύπτονται από τα chunks που διαχειρίζεται το shard αυτό.

cleanupReshardCollection	Database Command	cleanupReshardCollection	db.adminCommand(
			{
			cleanupReshardCollection: "<database>.<collection>"
			}
			)
				Η εντολή cleanupReshardCollection καθαρίζει τα μεταδεδομένα μιας αποτυχημένης λειτουργίας resharding.

commitReshardCollection	Database Command	commitReshardCollection	db.adminCommand(
			{
			commitReshardCollection: "<database>.<collection>"
			}
			)
				Εξαναγκάζει μια λειτουργία resharding να αποκλείσει τις εγγραφές και να ολοκληρωθεί

	mongosh Command	sh.commitReshardCollection()	sh.commitReshardCollection(<namespace>)	

configureCollectionBalancing	Database Command	configureCollectionBalancing	db.adminCommand(
			{
			configureCollectionBalancing: "<db>.<collection>",
			chunkSize: <num>,
			defragmentCollection: <bool>
			}
			)
				Διαμορφώνει τις ρυθμίσεις του balancer για μια sharded συλλογή, όπως η ρύθμιση του μεγέθους του chunk και η αποσύνθεση της συλλογής.

enableSharding	Database Command	enableSharding	db.adminCommand({	

			enableSharding: "<database name>" })	Δημιουργεί μια βάση δεδομένων.
	mongosh Command	sh.enableSharding	sh.enableSharding(<database>, <primary shard> // Optional. Available starting in MongoDB 4.2.2 (and 4.0.14))	

flushRouterConfig	Database Command		db.adminCommand(
			{
			flushRouterConfig: "<db.collection>"
			}
			)
				Αναγκάζει ένα mongod/ mongos στιγμιότυπο να ενημερώσει τα metadata δρομολόγησης που έχουν αποθηκευτεί στην κρυφή μνήμη.

getShardMap	Database Command	getShardMap	getShardMap	Είναι μια εσωτερική εντολή που υποστηρίζει τη λειτουργία του sharding
getShardVersion	Database Command	getShardVersion	getShardVersion	Υποστηρίζει τη λειτουργία του sharding στο MongoDB και δεν αποτελεί μέρος του σταθερού API που εκτίθεται στον πελάτη.

isdbgrid	Database Command	isdbgrid	db.runCommand(
			{
			isdbgrid: 1
			}
			)
				Αυτή η εντολή επαληθεύει ότι μια διεργασία είναι ένα mongos.

listShards	Database Command	listShards	db.runCommand(
			{
			listShards: 1
			}
			)
				Επιστρέφει μια λίστα με τα ρυθμισμένα shards σε ένα sharded cluster.

moveChunk	Database Command	moveChunk	db.adminCommand({ moveChunk : <namespace> , find : <query> , to : <string>, forceJumbo: <boolean>, // Starting in MongoDB 4.4 _secondaryThrottle : <boolean>, writeConcern: <document>, _waitForDelete : <boolean> })	Μετακινεί τα chunks ανάμεσα στα shards.
	mongosh Command	sh.moveChunk	sh.moveChunk(namespace, query, destination)	
movePrimary	Database Command	movePrimary	db.adminCommand({ movePrimary: <databaseName>, to: <newPrimaryShard> })	Επανακαθορίζει το primary shard όταν αφαιρείτε ένα shard από

				ένα sharded cluster.

moveRange	Database Command	moveRange	db.adminCommand(
			{
			toShard: <ID of the recipient shard>,
			min: <min key of the range to move>,
			max: <max key of the range to move>, // optional
			forceJumbo: <bool>, // optional
			waitForDelete: <bool>, // optional
			writeConcern: <write concern>, // optional
			secondaryThrottle: <bool> // optional
			}
			)
				Μετακινεί τα εύρη μεταξύ των shards.

mergeChunks	Database Command	mergeChunks	db.adminCommand(
			{
			mergeChunks: <namespace>,
			bounds : [
			{ <shardKeyField>: <minFieldValue> },
			{ <shardKeyField>: <maxFieldValue> }
			]
			}
			)
				Παρέχει τη δυνατότητα να συνδυάσετε τα chunks σε ένα μόνο shard.

refineCollectionShardKey	Database Command	refineCollectionShardKey	db.adminCommand(
			{
			refineCollectionShardKey: "<database>.<collection>",
			key: { <existing key specification>, <suffix1>: <1|"hashed">, ... }
			}
			)
				Βελτιώνει το shard key μιας συλλογής προσθέτοντας μια κατάληξη στο υπάρχον κλειδί.

removeShard	Database Command	removeShard	db.adminCommand(
			{
			removeShard : <shardToRemove>
			}
			)
				Ξεκινά τη διαδικασία αφαίρεσης ενός shard από ένα sharded cluster.

removeShardFromZone	Database Command	removeShardFromZone	db.adminCommand(
			{
			removeShardFromZone: <string>,
			zone: <string>
			}
			)
				Αφαιρεί τη συσχέτιση μεταξύ ενός shard και μιας ζώνης.

	mongosh Command	sh.removeShardFromZone	sh.removeShardFromZone("<zone>", "<shard>")	
reshardCollection	Database Command	reshardCollection	db.runCommand({ reshardCollection: "<database>.<collection>", key: <shardkey>, unique: <boolean>, numInitialChunks: <integer>, collation: { locale: "simple" }, zones: [{ min: <document with same shape as shardkey>, max: <document with same shape as shardkey>, zone: <string> | null }, ...	Η εντολή reshardCollection αλλάζει το κλειδί των κατακερματισμένων δεδομένων μιας συλλογής και αλλάζει την κατανομή των δεδομένων .

			] })	
	mongosh Command	sh.reshardCollection	sh.reshardCollection(db.collection, key, [options])	

setAllowMigrations	Database Command	setAllowMigrations	db.adminCommand(
			{
			setAllowMigrations: "<db>.<collection>",
			allowMigrations: <true|false>
			}
			)
				Αποτρέπει την έναρξη νέων αυτόματων μετεγκαταστάσεων σε μια συλλογή, εμποδίζει τις εντολές μετεγκατάστασης που βρίσκονται σε εξέλιξη να ολοκληρωθούν και αποκλείει τη συλλογή από νέους γύρους ισορροπίας.

setShardVersion	Database Command	setShardVersion	setShardVersion	Υποστηρίζει τη λειτουργία του sharding.

shardCollection	Database Command	shardCollection	db.adminCommand(
			{
			shardCollection: "<database>.<collection>",
			key: { <field1>: <1|"hashed">, ... },
			unique: <boolean>,
			numInitialChunks: <integer>,
			presplitHashedZones: <boolean>,
			collation: { locale: "simple" },
			timeseries: <object>
			}
			)
				Διαμερίζει ένα σύνολο εγγραφών για να διανείμει τα έγγραφά του σε διακομιστές (shards)

		sh.shardCollection	sh.shardCollection("<database>.<collection>", { <shard key fields> })	

shardingState	Database Command	shardingState	db.adminCommand(
			{
			shardingState: 1
			}
			)
				Αναφέρει εάν το mongod είναι μέλος ενός sharded cluster.

split	Database Command	split	db.adminCommand(
			{
			split: <database>.<collection>,
			<find|middle|bounds>
			}
			)
				Δημιουργεί ένα νέο chunk.

	mongosh Command	sh.splitAt	sh.splitAt("<namespace>", "<query>")	Διαιρεί ένα chunk στην τιμή του shard key που καθορίζεται από το ερώτημα (query).
	mongosh Command	sh.splitFind	sh.splitFind("<namespace>", "<query>")	Διαιρεί το κομμάτι που περιέχει την

				τιμή του κλειδιού κατανομής που καθορίζεται από το ερώτημα στο σημείο της μέσης τιμής του κομματιού
splitVector	Database Command	splitVector	splitVector(database.collection, document)	Είναι μια εντολή που χρησιμοποιείται για λειτουργίες μεταδεδομένων σε sharded clusters.
unsetSharding	Database Command	unsetSharding	unsetSharding	Είναι μια εσωτερική εντολή που υποστηρίζει τη λειτουργία του sharding (κατακερματισμού.

updateZoneKeyRange	Database Command	updateZoneKeyRange	db.adminCommand(
			{
			updateZoneKeyRange: <string>,
			min: <document>,
			max: <document>,
			zone: <string> | <null>
			}
			)
				Προσθέτει ή αφαιρεί τη συσχέτιση μεταξύ ενός εύρους δεδομένων που έχει κατακερματιστεί και μιας ζώνης.

	mongosh Command	sh.updateZoneKeyRange	sh.updateZoneKeyRange(namespace, minimum, maximum, zone)	

X. Sessions Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

abortTransaction	Database Command	abortTransaction	db.adminCommand(
			{
			abortTransaction: 1,
			txnNumber: <long>,
			writeConcern: <document>,
			autocommit: false,
			comment: <any>
			}
			)
				Τερματίζει τη συναλλαγή πολλών εγγράφων και επαναφέρει τυχόν αλλαγές δεδομένων που έγιναν από τις λειτουργίες εντός της συναλλαγής.

	mongosh Command	Session.abortTransaction( )	session.abort_transaction()	

commitTransaction	Database Command	commitTransaction	db.adminCommand(
			{
			commitTransaction: 1,
			txnNumber: <long>,
			writeConcern: <document>,
			autocommit: false,
			comment: <any>
			}
			)
				Αποθηκεύει τις αλλαγές που έγιναν από τις λειτουργίες στη συναλλαγή πολλών εγγράφων και τερματίζει τη συναλλαγή.

	mongosh Command	Session.commitTransacti on	Session.commitTransaction()	
	mongosh Command	Session.withTransaction	Session.withTransaction(<function> [, <options>])	Εκτελεί μια καθορισμένη συνάρτηση μέσα σε μια συναλλαγή

endSessions	Database Command	endSessions	db.runCommand(
			{
			endSessions: [{ id : <UUID> }, ...]
			}
			)
				Ανακαλεί τις συγκεκριμένες συνεδρίες και τις λήγει.

killAllSessions	Database Command	killAllSessions	db.runCommand(
			{
			killAllSessions: [{ user: <user>, db: <dbname> }, ...]
			}
			)
				"Σκοτώνει" όλες τις περιόδους σύνδεσης για τους καθορισμένους χρήστες.

killAllSessionsByPa ttern	Database Command	killAllSessionsByPattern	db.runCommand(
			{
			killAllSessionsByPattern: [<pattern>, ...]
			}
			)
				"Σκοτώνει" όλες τις περιόδους λειτουργίας που ταιριάζουν με οποιοδήποτε από τα καθορισμένα μοτίβα.

killSessions	Database Command	killSessions	db.runCommand(
			{
			killSessions: [{ id : <UUID> }, ...]
			}
			)
				"Σκοτώνει" συγκεκριμένες συνεδρίες.

refreshSessions	Database Command	refreshSessions	db.runCommand({ refreshSessions: [{ id : <UUID> }, ...]	Ανανεώνει τις αδρανείς περιόδους σύνδεσης.

			})	

startSession	Database Command	startSession	db.runCommand(
			{
			startSession: 1
			}
			)
				Ξεκινά μια νέα συνεδρία.

	mongosh Command	Mongo.startSession	Mongo.startSession(<options>)	

XI. Administration Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

cloneCollectionAsCapped	Database Command	cloneCollectionAsCapped	db.runCommand(
			{
			cloneCollectionAsCapped: <existing collection>,
			toCollection: <capped collection>,
			size: <capped size>,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Αντιγράφει μια μη-οριοθετημένη συλλογή ως μια νέα οριοθετημένη συλλογή."

collMod	Database Command	collMod	db.runCommand(
			{
			collMod: <collection or view>,
			<option1>: <value1>,
			<option2>: <value2>,
			...
			}
			)
				Καθιστά δυνατή την προσθήκη επιλογών σε μια συλλογή ή την τροποποίηση ορισμών προβολής.

compact	Database Command	compact	db.runCommand(
			{
			compact: <collection name>
			}
			)
				Επανεγγράφει και ανασυγκροτεί όλα τα δεδομένα και τα ευρετήρια σε μια συλλογή.

convertToCapped	Database Command	convertToCapped	db.runCommand(
			{
			convertToCapped: <collection>,
			size: <capped size>,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Μετατρέπει μια υπάρχουσα, χωρίς περιορισμούς συλλογή σε μια συλλογή με περιορισμένη κάλυψη εντός της ίδιας βάσης δεδομένων.

create	Database Command	create	db.runCommand(
			{
			create: <collection or view name>,
			capped: <true|false>,
			timeseries: {
			timeField: <string>,
			metaField: <string>,
			granularity: <string>
			},
			expireAfterSeconds: <number>,
			clusteredIndex: <document>, // Added in MongoDB 5.3
			changeStreamPreAndPostImages: <document>, // Added in MongoDB 6.0
			autoIndexId: <true|false>,
			size: <max_size>,
			max: <max_documents>,
			storageEngine: <document>,
			validator: <document>,
			validationLevel: <string>,
			validationAction: <string>,
			indexOptionDefaults: <document>,
				Δημιουργεί μια συλλογή ή μια προβολή.

			viewOn: <source>, pipeline: <pipeline>, collation: <document>, writeConcern: <document>, encryptedFields: <document>, comment: <any> }	

currentOp	Database Command	currentOp	db.adminCommand(
			{
			currentOp: 1
			}
			)
				Επιστρέφει ένα έγγραφο που περιέχει πληροφορίες για λειτουργίες σε εξέλιξη για την παρουσία της βάσης δεδομένων.

drop	Database Command	drop	db.runCommand(
			{
			drop: <collection_name>,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Αφαιρεί μια ολόκληρη συλλογή από μια βάση δεδομένων.

	mongosh Command	drop	db.collection.drop({ writeConcern: <document> })	

drop Database	Database Command	drop Database	db.runCommand(
			{
			dropDatabase: 1,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Καταργεί την τρέχουσα βάση δεδομένων.

dropConnections	Database Command	dropConnections	db.adminCommand(
			{
			dropConnections: 1,
			hostAndPort : ["host1:port1", "host2:port2", ...],
			comment: <any>
			}
			)
				Αποσύρει τις εξερχόμενες συνδέσεις στην καθορισμένη λίστα κεντρικών υπολογιστών.

fsync	Database Command	fsync	db.runCommand(
			{
			fsync: 1,
			lock: <Boolean>,
			comment: <any>
			}
			)
				Αδειάζει τις εκκρεμείς εγγραφές στο αποθηκευτικό επίπεδο και κλειδώνει τη βάση δεδομένων για να επιτρέψει αντίγραφα ασφαλείας

getAuditConfig	Database Command	getAuditConfig	db.adminCommand(
			{
			getAuditConfig: 1
			}
			)
				Ανακτά λεπτομέρειες σχετικά με τη διαμόρφωση και τα φίλτρα ελέγχου.

getClusterParameter	Database Command	getClusterParameter	db.adminCommand({ getClusterParameter: <parameter> | [<parameter>, <parameter>] | ""*""	Ανακτά τις τιμές παραμέτρων του cluster από όλους τους

			})	κόμβους σε ένα cluster.

getDefaultRWConcern	Database Command	getDefaultRWConcern	db.adminCommand(
			{
			getDefaultRWConcern: 1 ,
			inMemory: <boolean>,
			comment: <any>
			}
			)
				Ανακτά τις καθολικές προεπιλεγμένες ρυθμίσεις για ανάγνωση ή εγγραφή.

getParameter	Database Command	getParameter	db.adminCommand(
			{
			getParameter: <value>,
			<parameter> : <value>,
			comment: <any>
			}
			)
				Ανακτά τις επιλογές διαμόρφωσης.

killCursors	Database Command	killCursors	db.runCommand(
			{
			killCursors: <collection>,
			cursors: [<cursor id1>, ...], comment: <any>
			}
			)
				"Σκοτώνει" τους καθορισμένους δείκτες για μια συλλογή.

killOp	Database Command	killOp	db.adminCommand(
			{
			killOp: 1,
			op: <opid>,
			comment: <any>
			}
			)
				Τερματίζει μια λειτουργία όπως καθορίζεται από το αναγνωριστικό λειτουργίας.

	mongosh Command	db.killOp	db.killOp(<opid of the query to kill>)	

listCollections	Database Command	listCollections	db.runCommand(
			{
			listCollections: 1,
			filter: <document>,
			nameOnly: <boolean>,
			authorizedCollections: <boolean>,
			comment: <any>
			}
			)
				Επιστρέφει μια λίστα συλλογών στην τρέχουσα βάση δεδομένων.

	mongosh Command	db.getCollectionInfos	db.getCollectionInfos(filter, nameOnly, authorizedCollections)	Επιστρέφει μια σειρά εγγράφων με πληροφορίες συλλογής ή προβολής.
	mongosh Command	db.getCollectionNames	db.getCollectionNames()	Επιστρέφει έναν πίνακα που περιέχει τα ονόματα όλων των συλλογών και προβολών στην τρέχουσα βάση δεδομένων ή εάν εκτελείται με έλεγχο πρόσβασης, τα ονόματα των συλλογών σύμφωνα με τα

				δικαιώματα του χρήστη.

listDatabases	Database Command	listDatabases	db.adminCommand(
			{
			listDatabases: 1
			}
			)
				Επιστρέφει ένα έγγραφο που παραθέτει όλες τις βάσεις δεδομένων και επιστρέφει βασικά στατιστικά στοιχεία βάσης δεδομένων.

logRotate	Database Command	logRotate	db.adminCommand(
			{
			logRotate: <target>
			}
			)
				Περιστρέφει τα αρχεία καταγραφής MongoDB για να αποτρέψει ένα μεμονωμένο αρχείο από το να καταλαμβάνει πολύ χώρο.

renameCollection	Database Command	renameCollection	db.runCommand(
			{
			renameCollection: "<source_namespace>",
			to: "<target_namespace>",
			dropTarget: <true|false>,
			writeConcern: <document>,
			comment: <any>
			}
			)
				Αλλάζει το όνομα μιας υπάρχουσας συλλογής.

	mongosh Command	db.collection.renameCollection	db.collection.renameCollection(target, dropTarget)	

rotateCertificates	Database Command	rotateCertificates	db.runCommand(
			{
			rotateCertificates: 1,
			message: "<optional log message>"
			}
			)
				Εκτελεί online εναλλαγή πιστοποιητικού TLS.

setFeatureCompatibilityVersion	Database Command	setFeatureCompatibilityVersion	db.adminCommand(
			{
			setFeatureCompatibilityVersion: <version>,
			writeConcern: { wtimeout: <timeout> }
			}
			)
				Ενεργοποιεί ή απενεργοποιεί λειτουργίες που διατηρούν δεδομένα που δεν είναι συμβατά προς τα πίσω.

setClusterParameter	Database Command	setClusterParameter	db.adminCommand(
			{
			setClusterParameter: { <parameter>: <value>
			}
			}
			)
				Τροποποιεί τις επιλογές διαμόρφωσης για όλους τους κόμβους σε ένα σύμπλεγμα.

setParameter	Database Command	setParameter	db.adminCommand(
			{
			setParameter: 1,
			<parameter>: <value>
			}
			)
				Τροποποιεί τις επιλογές διαμόρφωσης.

setDefaultRWConcern	Database Command	setDefaultRWConcern	db.adminCommand({	Ορίζει τις καθολικές

			setDefaultRWConcern : 1, defaultReadConcern: { <read concern> }, defaultWriteConcern: { <write concern> }, writeConcern: { <write concern> }, comment: <any> })	προεπιλεγμένες επιλογές ανάγνωσης και εγγραφής για την ανάπτυξη.

shutdown	Database Command	shutdown	db.adminCommand(
			{
			shutdown: 1,
			force: <boolean>
			timeoutSecs: <int>,
			comment: <any>
			}
			)
				Τερματίζει τη διαδικασία mongodή mongos.

XII. Indexing Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

createIndexes	Database Command	createIndexes	db.runCommand(
			{
			createIndexes: <collection>,
			indexes: [
			{
			key: {
			<key-value_pair>,
			<key-value_pair>, ...
			},
			name: <index_name>,
			<option1>, <option2>, ...
			},
			{ ... },
			{ ... }
			],
			writeConcern: { <write concern> },
			commitQuorum: <int|string>,
			comment: <any>
			}
			)
				Δημιουργεί ένα ή περισσότερα ευρετήρια σε μια συλλογή.

	mongosh Command	db.collection.createIndex	db.collection.createIndex(keys, options, commitQuorum)	
	mongosh Command	db.collection.createIndexes	db.collection.createIndexes([keyPatterns], options, commitQuorum)	
dropIndexe	Database Command	dropIndexe	db.collection.dropIndex("index_name")	Αφαιρεί ένα ευρετήριο από μια συλλογή.

dropIndexes	Database Command	dropIndexes	db.runCommand(
			{
			dropIndexes: <string>,
			index: <string|document|arrayofstrings>,
			writeConcern: <document>, comment: <any>
			}
			)
				Αφαιρεί ευρετήρια από μια συλλογή.

listIndexes	Database Command	listIndexes	db.runCommand (
			{
			listIndexes: "<collection-name>",
			cursor: { batchSize: <int> },
			comment: <any>
			}
			)
				Εμφανίζει όλα τα ευρετήρια για μια συλλογή.

	mongosh Command	db.collection.getIndexes	db.collection.getIndexes()	

reIndex	Database Command	reIndex	db.runCommand(
			{
			reIndex: <collection>
			}
			)
				Αναδημιουργεί όλα τα ευρετήρια σε μια συλλογή.

	mongosh Command	db.collection.reIndex	db.collection('myCollection').reIndex()	
totalIndexSize	mongosh Command	db.collection.totalIndexSize	db.collectionName.totalIndexSize()	Επιστρέφει το συνολικό μέγεθος όλων των ευρετηρίων για τη συλλογή.

hideIndex	mongosh Command	hideIndex	db.collection.hideIndex(<index>)	Αποκρύπτει ένα υπάρχον ευρετήριο από το πρόγραμμα σχεδιασμού ερωτημάτων.
unhideIndex	mongosh Command	unhideIndex	db.collection.unhideIndex(<index>)	Εμφανίζει ένα υπάρχον ευρετήριο από το πρόγραμμα σχεδιασμού ερωτημάτων.
checkShardingIndex	Database Command	checkShardingIndex	db.runCommand({ checkShardingIndex: "myDatabase.myCollection" })	Είναι μια εσωτερική εντολή που υποστηρίζει τη λειτουργία του sharding.

XIII. Diagnostic Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

buildInfo	Database Command	buildInfo	db.runCommand(
			{
			buildInfo: 1
			}
			)
				Εμφανίζει στατιστικά στοιχεία σχετικά με την δομή της MongoDB

collStats	Database Command	collStats	db.runCommand(
			{
			collStats: <string>,
			scale: <int>
			}
			)
				Αναφέρει στατικά στοιχεία χρήσης αποθήκευσης για μια καθορισμένη συλλογή.

	mongosh Command	db.collection.stats	db.collection.stats(<option>)	
	mongosh Command	db.collection.dataSize	db.collectionName.dataSize()	Επιστρέφει το μέγεθος της συλλογής σε byte.
	mongosh Command	db.collection.estimatedDocumentCount	db.collection.estimatedDocumentCount(<options>)	Επιστρέφει τον αριθμό όλων των εγγράφων σε μια συλλογή ή προβολή.
	mongosh Command	db.collection.isCapped	db.collectionName.isCapped()	Επιστρέφει true εάν η συλλογή είναι είναι στο μέγιστο μέγεθος, διαφορετικά επιστρέφει false.
	mongosh Command	db.collection.latencyStats	db.collection.latencyStats({ histograms: <boolean> })	Αν η τιμή του histograms είναι true, τότε η μέθοδος latencyStats() προσθέτει ιστογράμματα καθυστέρησης (latency histograms) στο επιστρεφόμενο έγγραφο.
	mongosh Command	db.collection.storageSize	db.collectionName.storageSize()	Επιστρέφει το συνολικό μέγεθος αποθηκευτικού χώρου σε bytes που έχει εκχωρηθεί σε αυτήν τη συλλογή για την αποθήκευση εγγράφων.
	mongosh Command	db.collection.totalSize	db.collectionName.totalSize()	Επιστρέφει το συνολικό μέγεθος των

				δεδομένων της συλλογής συν το μέγεθος κάθε ευρετηρίου στη συλλογή.

connPoolStats	Database Command	connPoolStats	db.runCommand(
			{
			connPoolStats: 1
			}
			)
				Αναφέρει σε στατιστικά αναφορές για τις εξερχόμενες συνδέσεις από αυτήν την εγκατάσταση της MongoDB προς άλλες εγκαταστάσεις της MongoDB στο σύστημα.

connectionStatus	Database Command	connectionStatus	db.runCommand(
			{
			connectionStatus: 1,
			showPrivileges: <boolean>
			}
			)
				Αναφέρει την κατάσταση πιστοποίησης για την τρέχουσα σύνδεση.

dataSize	Database Command	dataSize	db.runCommand(
			{
			dataSize: <string>,
			keyPattern: <document>,
			min: <document>,
			max: <document>,
			estimate: <boolean>
			}
			)
				Επιστρέφει το μέγεθος δεδομένων για ένα εύρος δεδομένων. Προορίζεται για εσωτερική χρήση.

dataSize	Database Command	dataSize	db.runCommand(
			{
			dbHash: 1,
			collections: [<collection1>, ...]
			}
			)
				Επιστρέφει το μέγεθος δεδομένων για ένα εύρος δεδομένων. Προορίζεται για εσωτερική χρήση

dbStats	Database Command	dbStats	db.runCommand(
			{
			dbStats: 1,
			scale: <number>,
			freeStorage: 0
			}
			)
				Αναφέρει τα στατιστικά χρήσης αποθήκευσης για τη συγκεκριμένη βάση δεδομένων.

dbHash	Database Command	dbHash	db.runCommand(
			{
			dbHash: 1,
			collections: [<collection1>, ...]
			}
			)
				Επιστρέφει την κατακερματισμένη τιμή (hash value) μιας βάσης δεδομένων και των συλλογών της.

driverOIDTest	Database Command	driverOIDTest	db.runCommand({ eval: "driverOIDTest()" })	Εσωτερική εντολή που μετατρέπει ένα αναγνωριστικό αντικειμένου

				(ObjectId) σε αλφαριθμητικό για την υποστήριξη των δοκιμών.

explain	Database Command	explain	db.runCommand(
			{
			explain: <command>,
			verbosity: <string>,
			comment: <any>
			}
			)
				Επιστρέφει πληροφορίες για την εκτέλεση διαφόρων λειτουργιών.

	mongosh Command	db.collection.explain	db.collection.explain().<method(...)>	
features	Database Command	features	db.runCommand({features})	Αναφέρει τις δυνατότητες που είναι διαθέσιμες στην τρέχουσα εγκατάσταση του MongoDB.

getCmdLineOpts	Database Command	getCmdLineOpts	db.adminCommand(
			{
			getCmdLineOpts: 1
			}
			)
				Επιστρέφει ένα έγγραφο με τις παραμέτρους εκτέλεσης για την εγκατάσταση του MongoDB και τις επιλογές που έχουν αναλυθεί.

getLog	Database Command	getLog	db.adminCommand(
			{
			getLog: <value>
			}
			)
				Επιστρέφει πρόσφατα μηνύματα καταγραφής.

hostInfo	Database Command	hostInfo	db.adminCommand(
			{
			hostInfo: 1
			}
			)
				Επιστρέφει δεδομένα που αντανακλούν το υποκείμενο σύστημα του κεντρικού υπολογιστή.

listCommands	Database Command	listCommands	db.runCommand(
			{
			listCommands: 1
			}
			)
				Λίστα με όλες τις εντολές βάσης δεδομένων που παρέχονται από την τρέχουσα εγκατάσταση του mongod.

lockInfo	Database Command	lockInfo	db.adminCommand(
			{
			lockInfo: 1
			}
			)
				Εσωτερική εντολή που επιστρέφει πληροφορίες για κλειδώματα που κατέχονται αυτήν τη στιγμή ή εκκρεμούν.

ping	Database Command	ping	db.runCommand({	Εσωτερική εντολή που

			ping: 1 })	αναφέρεται στη συνδεσιμότητα μεταξύ των διακομιστών στην εγκατάσταση.

profile	Database Command	profile	db.runCommand(
			{
			profile: <level>,
			slowms: <threshold>,
			sampleRate: <rate>,
			filter: <filter expression>
			}
			)
				Ενεργοποιεί, απενεργοποιεί ή διαμορφώνει το database profiler

serverStatus	Database Command	serverStatus	db.runCommand(
			{
			serverStatus: 1
			}
			)
				Επιστρέφει στατιστικά στοιχεία συλλογής σχετικά με την χρήση πόρων και την κατάσταση της εγκατάστασης στο επίπεδο της συλλογής.

shardConnPoolStats	Database Command	shardConnPoolStats	{ shardConnPoolStats: 1 }	Επιστρέφει πληροφορίες σχετικά με τις ομαδοποιημένες και τις προσωρινά αποθηκευμένες συνδέσεις στο κοινόχρηστο χώρο συγκέντρωσης συνδέσεων.

top	Database Command	top	db.runCommand(
			{
			top: 1
			}
			)
				Επιστρέφει τα ακατέργαστα στατιστικά χρήσης για κάθε βάση δεδομένων.

validate	Database Command	validate	db.runCommand(
			{
			validate: <string>, // Collection name
			full: <boolean>, // Optional
			repair: <boolean>, // Optional, added in MongoDB 5.0
			metadata: <boolean> // Optional, added in MongoDB 5.0.4
			}
			)
				Σαρώνει τα δεδομένα και τα ευρετήρια μιας συλλογής για να ελέγξει την ορθότητά τους.

	mongosh Command	db.collection.validate	db.collection.validate(<documents>)	

validateDBMetadata	Database Command	validateDBMetadata	db.runCommand(
			{
			validateDBMetadata: 1,
			apiParameters: {
			version: <string>,
				Σαρώνει τα δεδομένα και τα ευρετήρια μιας συλλογής για να

			strict: <boolean>, deprecationErrors: <boolean> }, db: <string>, collection: <string>, })	ελέγξει την ορθότητά τους.
whatsmyuri	Database Command	whatsmyuri	db.runCommand('whatsmyuri')	Επιστρέφει πληροφορίες σχετικά με τον τρέχοντα client.

XIV. Free Monitoring Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

getFreeMonitoringStatus	Database Command	getFreeMonitoringStatus	db.adminCommand(
			{
			getFreeMonitoringStatus: 1
			}
			)
				Επιστρέφει την κατάσταση της δωρεάν παρακολούθησης.

	mongosh Command	db.getFreeMonitoringStatus	db.getFreeMonitoringStatus()	

setFreeMonitoring	Database Command	setFreeMonitoring	db.adminCommand(
			{
			setFreeMonitoring: 1,
			action: "<enable|disable>"
			}
			)
				Ενεργοποιεί / απενεργοποιεί τη δωρεάν παρακολούθηση κατά τη διάρκεια της εκτέλεσης.

XV. System Events Auditing Commands				
Κατηγορία Εντολής	**Τύπος Εντολής**	**Εντολή**	**Σύνταξη**	**Ορισμός**

logApplicationMessage	Database Command	logApplicationMessage	db.runCommand(
			{
			logApplicationMessage: <string>
			}
			)
				Αναρτά ένα προσαρμοσμένο μήνυμα στο αρχείο καταγραφής ελέγχου.

[36]

XVI. Operators		
Όνομα	**Σύνταξη**	**Περιγραφή**
$	db.collection.find({ <array>: <condition> ... }, { "<array>.$": 1 }) db.collection.find({ <array.field>: <condition> ...}, { "<array>.$": 1 })	Προβάλλει το πρώτο στοιχείο σε έναν πίνακα που ταιριάζει με τη συνθήκη του ερωτήματος.
$[<identifier>]	{ <update operator>: { "<array>.$[<identifier>]" : value } }, { arrayFilters: [{ <identifier>: <condition> }] }	Λειτουργεί ως σύμβολο κράτησης θέσης για την ενημέρωση όλων των στοιχείων που ταιριάζουν με τη arrayFilters συνθήκη για τα έγγραφα που αντιστοιχούν στη συνθήκη ερωτήματος.
$abs	{ $abs: <number> }	Επιστρέφει την απόλυτη τιμή ενός αριθμού.

$accumulator	{
	$accumulator: {
	init: <code>,
	initArgs: <array expression>, // Optional
	accumulate: <code>,
	accumulateArgs: <array expression>,
	merge: <code>,
	finalize: <code>, // Optional
	lang: <string>
	}
	}
		Επιστρέφει το αποτέλεσμα μιας συνάρτησης συσσωρευτή που καθορίζεται από τον χρήστη.

$add	{ $add: [<expression1>, <expression2>, ...] }	Προσθέτει αριθμούς για να επιστρέψει το άθροισμά τους.
$addFields	{ $addFields: { <newField>: <expression>, ... } }	Προσθέτει νέα πεδία στα έγγραφα.
$addToSet	{ $addToSet: { <field1>: <value1>, ... } }	Προσθέτει στοιχεία σε έναν πίνακα μόνο εάν δεν υπάρχουν ήδη στο σύνολο.
$all	{ <field>: { $all: [<value1> , <value2> ...] } }	Αντιστοιχεί σε πίνακες που περιέχουν όλα τα στοιχεία που έχουν καθοριστεί στο ερώτημα.
$allElementsTrue	{ $allElementsTrue: [<expression>] }	Επιστρέφει true αν κανένα στοιχείο ενός συνόλου δεν αποδίδεται ως false, διαφορετικά επιστρέφει false.
$and	{ $and: [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }	Ενώνει όρους ερωτήματος με μια λογική AND και επιστρέφει όλα τα έγγραφα που ταιριάζουν με τις συνθήκες και των δύο προτάσεων.
$and	{ $and: [<expression1>, <expression2>, ...] }	Επιστρέφει true μόνο όταν όλες οι παραστάσεις της αξιολογούνται ως true.
$anyElementTrue	{ $anyElementTrue: [<expression>] }	Επιστρέφει true αν κάποιο στοιχείο ενός συνόλου αποδίδεται ως true, διαφορετικά επιστρέφει false.
$arrayElemAt	{ $arrayElemAt: [<array>, <idx>] }	Επιστρέφει το στοιχείο στην καθορισμένη θέση ενός πίνακα.
$arrayToObject	[[["item", "abc123"], ["qty", 25]]]	Μετατρέπει έναν πίνακα από ζεύγη κλειδιού-τιμής σε ένα έγγραφο.
$avg	{ $avg: <expression> }	Επιστρέφει τον μέσο όρο αριθμητικών τιμών. Αγνοεί μη-αριθμητικές τιμές.
$binarySize	{ $binarySize: <string or binData> }	Επιστρέφει το μέγεθος του περιεχομένου μιας δοσμένης συμβολοσειράς ή δυαδικής τιμής σε bytes.
$bitsAllClear	{ <field>: { $bitsAllClear: <numeric bitmask> } } { <field>: { $bitsAllClear: < BinData bitmask> } } { <field>: { $bitsAllClear: [<position1>, <position2>, ...] } }	Αντιστοιχίζει αριθμητικές ή δυαδικές τιμές στις οποίες ένα σύνολο θέσεων bit έχουν όλες τιμή 0.

$bitsAllSet	{ <field>: { $bitsAllSet: <numeric bitmask> } } { <field>: { $bitsAllSet: < BinData bitmask> } } { <field>: { $bitsAllSet: [<position1>, <position2>, ...] } }	Αντιστοιχίζει αριθμητικές ή δυαδικές τιμές στις οποίες ένα σύνολο θέσεων bit έχουν όλες τιμή 1.
$bitsAnyClear	{ <field>: { $bitsAnyClear: <numeric bitmask> } } { <field>: { $bitsAnyClear: < BinData bitmask> } } { <field>: { $bitsAnyClear: [<position1>, <position2>, ...] } }	Αντιστοιχίζει αριθμητικές ή δυαδικές τιμές στις οποίες οποιοδήποτε bit από ένα σύνολο θέσεων bit έχει τιμή 0.
$bitsAnySet	{ <field>: { $bitsAnySet: <numeric bitmask> } } { <field>: { $bitsAnySet: < BinData bitmask> } } { <field>: { $bitsAnySet: [<position1>, <position2>, ...] } }	Αντιστοιχίζει αριθμητικές ή δυαδικές τιμές στις οποίες οποιοδήποτε bit από ένα σύνολο θέσεων bit έχει τιμή 1.

$bottom	{
	$bottom:
	{
	sortBy: { <field1>: <sort order>, <field2>: <sort order> ... },
	output: <expression>
	}
	}
		Επιστρέφει το κάτωθι στοιχείο εντός μιας ομάδας, σύμφωνα με τη συγκεκριμένη σειρά ταξινόμησης.

$bottomN	{
	$bottomN:
	{
	n: <expression>,
	sortBy: { <field1>: <sort order>, <field2>: <sort order> ... },
	output: <expression>
	}
	}
		Επιστρέφει μια συγκέντρωση των τελευταίων n πεδίων εντός μιας ομάδας, σύμφωνα με τη συγκεκριμένη σειρά ταξινόμησης.

$bucket	{
	$bucket: {
	groupBy: <expression>,
	boundaries: [<lowerbound1>, <lowerbound2>, ...],
	default: <literal>,
	output: {
	<output1>: { <$accumulator expression> },
	...
	<outputN>: { <$accumulator expression> }
	}
	}
	}
		Κατηγοριοποιεί τα εισερχόμενα έγγραφα σε ομάδες, ονομαζόμενες buckets, βάσει μιας καθορισμένης έκφρασης και ορίων bucket.

$bucketAuto	{
	$bucketAuto: {
	groupBy: <expression>,
	buckets: <number>,
	output: {
	<output1>: { <$accumulator expression> },
	...
	}
	granularity: <string>
	}
	}
		Κατηγοριοποιεί εισερχόμενα έγγραφα σε ένα συγκεκριμένο αριθμό ομάδων, που ονομάζονται κουβάδες, με βάση μια καθορισμένη έκφραση.

$ceil	{ $ceil: <number> }	Επιστρέφει το μικρότερο ακέραιο μεγαλύτερο ή ίσο με το καθορισμένο αριθμό.

$changeStream	{
	$changeStream: {
	allChangesForCluster: <boolean>,
	fullDocument: <string>,
	fullDocumentBeforeChange: <string>,
	resumeAfter: <int>
	showExpandedEvents: <boolean>,
	startAfter: <document>
	startAtOperationTime: <timestamp>
	}
		Επιστρέφει έναν κέρσορα αλλαγής (Change Stream) για τη συλλογή.

	}	
$cmp	{ $cmp: [<expression1>, <expression2>] }	Επιστρέφει: 0 αν οι δύο τιμές είναι ισοδύναμες, 1 αν η πρώτη τιμή είναι μεγαλύτερη από τη δεύτερη, και -1 αν η πρώτη τιμή είναι μικρότερη από τη δεύτερη.

$collStats	{
	$collStats:
	{
	latencyStats: { histograms: <boolean> },
	storageStats: { scale: <number> },
	count: {},
	queryExecStats: {}
	}
	}
		Επιστρέφει στατιστικά στοιχεία σχετικά με μια συλλογή.

$comment	db.collection.find({ <query>, $comment: <comment> })	Προσθέτει ένα σχόλιο σε ένα πρότυπο ερώτημα (query predicate).
$concat	{ $concat: [<expression1>, <expression2>, ...] }	Συνενώνει οποιονδήποτε αριθμό από συμβολοσειρές.
$concatArrays	{ $concatArrays: [<array1>, <array2>, ...] }	Συνενώνει πίνακες για να επιστρέψει το συνενωμένο πίνακα.
$cond	{ $cond: { if: <boolean-expression>, then: <true-case>, else: <false-case> } }	Ένας τριαδικός τελεστής που αξιολογεί μια έκφραση και, ανάλογα με το αποτέλεσμα, επιστρέφει την τιμή μίας από τις άλλες δύο εκφράσεις.

$convert	{
	$convert:
	{
	input: <expression>,
	to: <type expression>,
	onError: <expression>, // Optional.
	onNull: <expression> // Optional.
	}
	}
		Μετατρέπει μια τιμή σε ένα καθορισμένο τύπο.

$count	{ $count: <string> }	Επιστρέφει μια καταμέτρηση του αριθμού των εγγράφων.
$currentDate	{ $currentDate: { <field1>: <typeSpecification1>, ... } }	Ορίζει την τιμή ενός πεδίου στην τρέχουσα ημερομηνία, είτε ως ημερομηνία είτε ως timestamp.

$dateAdd	{
	$dateAdd: {
	startDate: <Expression>,
	unit: <Expression>,
	amount: <Expression>,
	timezone: <tzExpression>
	}
	}
		Προσθέτει μια συγκεκριμένη μονάδα χρόνου σε ένα αντικείμενο ημερομηνίας.

$dateDiff	{
	$dateDiff: {
	startDate: <Expression>,
	endDate: <Expression>,
	unit: <Expression>,
	timezone: <tzExpression>,
	startOfWeek: <String>
	}
	}
		Επιστρέφει τη διαφορά μεταξύ δύο ημερομηνιών.

$dateFromParts	{
	$dateFromParts : {
	'year': <year>, 'month': <month>, 'day': <day>,
		Δημιουργεί ένα αντικείμενο BSON Date δεδομένης της ημερομηνίας και των στοιχείων της.

	'hour': <hour>, 'minute': <minute>, 'second': <second>, 'millisecond': <ms>, 'timezone': <tzExpression> } }	
$dateFromString	{ $dateFromString: { dateString: <dateStringExpression>, format: <formatStringExpression>, timezone: <tzExpression>, onError: <onErrorExpression>, onNull: <onNullExpression> } }	Επιστρέφει μια ημερομηνία/ώρα ως αντικείμενο ημερομηνίας.

$dateSubtract	{
	$dateSubtract: {
	startDate: <Expression>,
	unit: <Expression>,
	amount: <Expression>,
	timezone: <tzExpression> }
	}
		Αφαιρεί έναν αριθμό μονάδων χρόνου από ένα αντικείμενο ημερομηνίας.

$dateToParts	{
	$dateToParts: {
	'date' : <dateExpression>,
	'timezone' : <timezone>,
	'iso8601' : <boolean>
	}
	}
		Επιστρέφει ένα έγγραφο που περιέχει τα συστατικά μέρη μιας ημερομηνίας.

$dateToString	{ $dateToString: { date: <dateExpression>, format: <formatString>, timezone: <tzExpression>, onNull: <expression> } }	Επιστρέφει την ημερομηνία ως μορφοποιημένο κείμενο.

$dateTrunc	{
	$dateTrunc: {
	date: <Expression>,
	unit: <Expression>,
	binSize: <Expression>,
	timezone: <tzExpression>,
	startOfWeek: <Expression>
	}
	}
		Κόβει τα δευτερόλεπτα, τα λεπτά ή τις ώρες από μια ημερομηνία και επιστρέφει την ημερομηνία με ακέραιες τιμές για τα κομμάτια που δεν έχουν κοπεί.

$dayOfMonth	{ $dayOfMonth: <dateExpression> }	Επιστρέφει τη μέρα του μήνα για μια ημερομηνία ως έναν αριθμό από 1 έως 31.
$dayOfWeek	{ $dayOfWeek: <dateExpression> }	Επιστρέφει την ημέρα της εβδομάδας για μια ημερομηνία ως έναν αριθμό από 1 (Κυριακή) έως 7 (Σάββατο).
$dayOfYear	{ $dayOfYear: <dateExpression> }	Επιστρέφει την ημέρα του έτους για μια ημερομηνία ως έναν αριθμό από 1 έως 366 (δίσεκτο έτος).
$denseRank	{ $denseRank: { } }	Επιστρέφει τη σχετική θέση ενός ταξινομημένου εγγράφου. Δεν υπάρχουν κενά στις θέσεις.

$densify	{
	$densify: {
	field: <fieldName>,
	partitionByFields: [<field 1>, <field 2> ... <field n>],
	range: {
	step: <number>,
		Δημιουργεί νέα έγγραφα σε μια ακολουθία εγγράφων όπου ορισμένες τιμές σε ένα πεδίο λείπουν.

	unit: <time unit>, bounds: < "full" || "partition" > || [< lower bound >, < upper bound >] } } }	

$derivative	{
	$derivative: {
	input: <expression>,
	unit: <time unit>
	}
	}
		Επιστρέφει το μέσο ρυθμό μεταβολής εντός του καθορισμένου παραθύρου.

$divide	{ $divide: [<expression1>, <expression2>] }	Επιστρέφει το αποτέλεσμα της διαίρεσης του πρώτου αριθμού με τον δεύτερο.
$documentNumber	{ $documentNumber: { } }	Επιστρέφει τη θέση ενός εγγράφου.
$documents	{ $documents: <expression> }	Επιστρέφει έγγραφα από εκφράσεις εισόδου.
$each	{ $addToSet: { <field>: { $each: [<value1>, <value2> ...] } } }	Τροποποιεί τους τελεστές $push και $addToSet για να προσθέτουν πολλά στοιχεία για ενημερώσεις πίνακα.
$elemMatch	{ <field>: { $elemMatch: { <query1>, <query2>, ... } } }	Προβάλλει το πρώτο στοιχείο σε έναν πίνακα που ταιριάζει με την καθορισμένη $elemMatch συνθήκη.
$eq	{ <field>: { $eq: <value> } }	Αντιστοιχίζει τιμές που είναι ίσες με μια καθορισμένη τιμή. Επιστρέφει true αν οι τιμές είναι ισοδύναμες.
$exists	{ field: { $exists: <boolean> } }	Ταιριάζει έγγραφα που έχουν το συγκεκριμένο πεδίο που έχει καθοριστεί.

$expMovingAvg	{
	$expMovingAvg: {
	input: <input expression>,
	N: <integer>,
	alpha: <float>
	}
	}
		Επιστρέφει τον εκθετικό κινούμενο μέσο όρο για την αριθμητική έκφραση.

$expr	{ $expr: { <expression> } }	Επιτρέπει τη χρήση εκφράσεων συγκέντρωσης (aggregation expressions) εντός της γλώσσας ερωτημάτων (query language).

$expMovingAvg	{
	$expMovingAvg: {
	input: <input expression>,
	N: <integer>,
	alpha: <float>
	}
	}
		Επιστρέφει τον εκθετικό κινούμενο μέσο όρο για την αριθμητική έκφραση.

$expr	{ $expr: { <expression> } }	Επιτρέπει τη χρήση εκφράσεων συγκέντρωσης (aggregation expressions) εντός της γλώσσας ερωτημάτων (query language).

$facet	{ $facet:
	{
	<outputField1>: [<stage1>, <stage2>, ...],
	<outputField2>: [<stage1>, <stage2>, ...],
	...
	}
	}
		Επεξεργάζεται πολλαπλά aggregation pipelines εντός ενός σταδίου για τα ίδια εισαγωγικά έγγραφα.

$fill	{ $fill: {	Συμπληρώνει τιμές σε πεδία που λείπουν ή έχουν τιμή null σε ένα έγγραφο.

	partitionBy: <expression>,
	partitionByFields: [<field 1>, <field 2>, ... , <field n>],
	sortBy: {
	<sort field 1>: <sort order>,
	<sort field 2>: <sort order>,
	...,
	<sort field n>: <sort order>
	},
	output: {
	<field 1>: { value: <expression> },
	<field 2>: { method: <string> },
	...
	}
	}
	}	

$filter	{
	$filter:
	{
	input: <array>,
	cond: <expression>,
	as: <string>,
	limit: <number expression>
	}
	}
		Επιλέγει ένα υποσύνολο του πίνακα για να επιστρέψει έναν πίνακα με μόνο τα στοιχεία που ταιριάζουν με τη συνθήκη φίλτρου.

$first	{ $first: <expression> }	Επιστρέφει μια τιμή από το πρώτο έγγραφο για κάθε ομάδα.

$firstN	{
	$firstN:
	{
	input: <expression>,
	n: <expression>
	}
	}
		Επιστρέφει έναν καθορισμένο αριθμό στοιχείων από την αρχή ενός πίνακα.

$floor	{ $floor: <number> }	Επιστρέφει το μεγαλύτερο ακέραιο αριθμό μικρότερο ή ίσο με τον καθορισμένο αριθμό.

$function	{
	$function: {
	body: <code>,
	args: <array expression>,
	lang: "js"
	}
	}
		Ορίζει μια προσαρμοσμένη συνάρτηση συγκέντρωσης.

$geoIntersects	{
	<location field>: {
	$geoIntersects: {
	$geometry: {
	type: "<GeoJSON object type>" ,
	coordinates: [<coordinates>]
	}
	}
	}
	}
		Επιλέγει γεωμετρίες που τέμνονται με μια γεωμετρία GeoJSON. Ο δείκτης 2dsphere υποστηρίζει το $geoIntersects.

$geoNear	{ $geoNear: { <geoNear options> } }	Επιστρέφει μια ταξινομημένη ροή εγγράφων βάσει της απόστασης σε ένα γεωχωρικό σημείο.

$geoWithin	{
	<location field>: {
	$geoWithin: {
	$geometry: {
		Επιλέγει γεωμετρίες που βρίσκονται εντός μιας οριοθετημένης γεωμετρίας GeoJSON. Οι δείκτες 2dsphere και 2d υποστηρίζουν το $geoWithin.

	type: <"Polygon" or "MultiPolygon"> , coordinates: [<coordinates>] } } } }	

$getField	{
	$getField: {
	field: <String>,
	input: <Object>
	}
	}
		Επιστρέφει την τιμή ενός καθορισμένου πεδίου από ένα έγγραφο.

$graphLookup	{
	$graphLookup: {
	from: <collection>,
	startWith: <expression>,
	connectFromField: <string>,
	connectToField: <string>,
	as: <string>,
	maxDepth: <number>,
	depthField: <string>,
	restrictSearchWithMatch: <document>
	}
	}
		Εκτελεί μια αναδρομική αναζήτηση σε μια συλλογή.

$group	{
	$group:
	{
	_id: <expression>, // Group key
	<field1>: { <accumulator1> : <expression1> },
	...
	}
	}
		Ομαδοποιεί τα έγγραφα εισαγωγής με μια καθορισμένη έκφραση αναγνωριστικού και εφαρμόζει τις εκφράσεις συσσωρευτή, εάν καθορίζεται, σε κάθε ομάδα.

$gt	{ field: { $gt: value } }	Αντιστοιχίζει τιμές που είναι μεγαλύτερες από μια καθορισμένη τιμή.
$gte	{ field: { $gte: value } }	Αντιστοιχίζει τιμές που είναι μεγαλύτερες ή ίσες με μια καθορισμένη τιμή. Επιστρέφει true αν η πρώτη τιμή είναι μεγαλύτερη από ή ίση με τη δεύτερη.
$hour	{ $hour: <dateExpression> }	Επιστρέφει την ώρα μιας ημερομηνίας ως έναν αριθμό από 0 έως 23.

$ifNull	{
	$ifNull: [
	<input-expression-1>,
	...
	<input-expression-n>,
	<replacement-expression-if-null>
	]
	}
		Επιστρέφει είτε το μη-κενό αποτέλεσμα της πρώτης έκφρασης είτε το αποτέλεσμα της δεύτερης έκφρασης εάν η πρώτη έκφραση οδηγεί σε κενό αποτέλεσμα.

$in	{ field: { $in: [<value1>, <value2>, ... <valueN>] } }	Ταιριάζει με οποιαδήποτε από τις τιμές που καθορίζονται σε έναν πίνακα. Επιστρέφει ένα boolean που υποδηλώνει εάν μια καθορισμένη τιμή υπάρχει σε έναν πίνακα.

$inc	{ $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }	Αυξάνει την τιμή του πεδίου κατά το καθορισμένο ποσό.
$indexOfArray	{ $indexOfArray: [<array expression>, <search expression>, <start>, <end>] }	Αναζητά μια τιμή σε έναν πίνακα και επιστρέφει τον δείκτη του πίνακα της πρώτης εμφάνισης.
$indexOfBytes	{ $indexOfBytes: [<string expression>, <substring expression>, <start>, <end>] }	Αναζητά μια υποσυμβολοσειρά σε ένα string και επιστρέφει το UTF-8 δείκτη byte της πρώτης εμφάνισης.
$indexOfCP	{ $indexOfCP: [<string expression>, <substring expression>, <start>, <end>] }	Αναζητά μια υποσυμβολοσειρά σε ένα string και επιστρέφει το UTF-8 δείκτη code point της πρώτης εμφάνισης.
$indexStats	{ $indexStats: { } }	Επιστρέφει στατιστικά στοιχεία σχετικά με τη χρήση κάθε ευρετηρίου για τη συλλογή.

$integral	{
	$integral: {
	input: <expression>,
	unit: <time unit>
	}
	}
		Επιστρέφει την προσέγγιση της περιοχής κάτω από μια καμπύλη.

$isArray	{ $isArray: [<expression>] }	Καθορίζει εάν ο τελεστής είναι ένας πίνακας. Επιστρέφει ένα boolean.
$isNumber	{ $isNumber: <expression> }	Καθορίζει εάν η έκφραση αντιστοιχεί σε ακέραιο, διπλό, δεκαδικό ή μακροσκοπικό αριθμό.
$isoDayOfWeek	{ $isoDayOfWeek: <dateExpression> }	Επιστρέφει τον αριθμό της ημέρας της εβδομάδας στη μορφή ISO 8601, κυμαινόμενος από το 1 (για τη Δευτέρα) έως το 7 (για την Κυριακή).
$isoWeek	{ $isoWeek: <dateExpression> }	Επιστρέφει τον αριθμό της εβδομάδας στη μορφή ISO 8601, κυμαινόμενος από 1 έως 53.
$isoWeekYear	{ $isoWeekYear: <dateExpression> }	Επιστρέφει τον αριθμό του έτους σε μορφή ISO 8601.
$jsonSchema	{ $jsonSchema: <JSON Schema object> }	Επικυρώνει τα έγγραφα σε σχέση με το συγκεκριμένο σχήμα JSON.
$last	{ $last: <expression> }	Επιστρέφει το τελευταίο στοιχείο του πίνακα.

$lastN	{
	$lastN:
	{
	input: <expression>,
	n: <expression>
	}
	}
		Επιστρέφει έναν καθορισμένο αριθμό στοιχείων από το τέλος ενός πίνακα.

$let	{
	$let:
	{
	vars: { <var1>: <expression>, ... },
	in: <expression>
	}
	}
		Ορίζει μεταβλητές για χρήση εντός του πεδίου εφαρμογής μιας υποέκφρασης και επιστρέφει το αποτέλεσμα της υποέκφρασης.

$limit	{ $limit: <positive 64-bit integer> }	Διαβιβάζει τα πρώτα n έγγραφα χωρίς τροποποίηση στη διοχέτευση όπου n είναι το καθορισμένο όριο. Για κάθε έγγραφο εισόδου, εξάγει είτε ένα έγγραφο (για τα πρώτα n έγγραφα) είτε μηδέν έγγραφα (μετά τα πρώτα n έγγραφα).
$linearFill	{ $linearFill: <expression> }	Συμπληρώνει τα κενά και τις τιμές που λείπουν σε ένα παράθυρο χρησιμοποιώντας γραμμική απόσταση με βάση τις γύρω τιμές του πεδίου.
$listSessions	{ $listSessions: <document> }	Εμφανίζει όλες τις περιόδους σύνδεσης που ήταν ενεργές για αρκετό καιρό για να μεταδοθούν στη system.sessions συλλογή.
$ln	{ $ln: <number> }	Υπολογίζει το φυσικό λογάριθμο ενός αριθμού.
$locf	{ $locf: <expression> }	Συμπληρώνει τα κενά και τις τιμές που λείπουν σε ένα παράθυρο στην τελευταία μη μηδενική τιμή του πεδίου.
$log	{ $log: [<number>, <base>] }	Υπολογίζει το λογάριθμο ενός αριθμού στην καθορισμένη βάση.
$log10	{ $log10: <number> }	Υπολογίζει το λογάριθμο στη βάση 10 ενός αριθμού.

$lookup	{
	$lookup:
	{
	from: <collection to join>,
	localField: <field from the input documents>,
	foreignField: <field from the documents of the "from" collection>,
	as: <output array field>
	}
	}
		Εκτελεί μια αριστερή εξωτερική σύζευξη με μια άλλη συλλογή στην ίδια βάση δεδομένων να φιλτράρει έγγραφα από τη "συνδεδεμένη" συλλογή για επεξεργασία.

$lt	{ $lt: [<expression1>, <expression2>] }	Αντιστοιχίζει τιμές που είναι μικρότερες από μια καθορισμένη τιμή. Επιστρέφει true αν η πρώτη τιμή είναι μικρότερη από τη δεύτερη.
$lte	{ $lte: [<expression1>, <expression2>] }	Αντιστοιχίζει τιμές που είναι μικρότερες ή ίσες με μια καθορισμένη τιμή. Επιστρέφει true αν η πρώτη τιμή είναι μικρότερη ή ίση με τη δεύτερη.
$ltrim	{ $ltrim: { input: <string>, chars: <string> } }	Αφαιρεί τα κενά ή τους καθορισμένους χαρακτήρες από την αρχή μιας συμβολοσειράς.
$map	{ $map: { input: <expression>, as: <string>, in: <expression> } }	Εφαρμόζει μια υποέκφραση σε κάθε στοιχείο ενός πίνακα και επιστρέφει τον πίνακα των αποτελεσμάτων σε σειρά.
$match	{ $match: { <query> } }	Φιλτράρει τη ροή εγγράφων για να επιτρέπει μόνο τα αντίστοιχα έγγραφα να περνούν χωρίς τροποποίηση στο επόμενο στάδιο διοχέτευσης.

$max	{ $max: { <field1>: <value1>, ... } }	Επιστρέφει τη μεγαλύτερη τιμή έκφρασης για κάθε ομάδα.

$maxN	{
	$maxN:
	{
	input: <expression>,
	n: <expression>
	}
	}
		Επιστρέφει τις n μεγαλύτερες τιμές σε έναν πίνακα.

$merge	{ $merge: { into: <collection> -or- { db: <db>, coll: <collection> }, on: <identifier field> -or- [<identifier field1>, ...], // Optional let: <variables>, // Optional whenMatched: <replace|keepExisting|merge|fail|pipeline>, // Optional whenNotMatched: <insert|discard|fail> // Optional } }	Γράφει τα αποτελέσματα ενός aggregation pipeline σε μια συλλογή.
$mergeObjects	{ $mergeObjects: <document> }	Συνδυάζει πολλά έγγραφα σε ένα έγγραφο.
$meta	{ $meta: <metaDataKeyword> }	Προβάλλει τη βαθμολογία του εγγράφου που έχει ανατεθεί κατά την λειτουργία $text.
$millisecond	{ $millisecond: <dateExpression> }	Επιστρέφει τα χιλιοστά δευτερολέπτων μιας ημερομηνίας ως έναν αριθμό από το 0 έως το 999.
$min	{ $min: { <field1>: <value1>, ... } }	Ενημερώνει το πεδίο μόνο εάν η καθορισμένη τιμή είναι μικρότερη από την υπάρχουσα τιμή πεδίου.

$minN	{
	$minN:
	{
	input: <expression>,
	n: <expression>
	}
	}
		Επιστρέφει τα n μικρότερα στοιχεία ενός πίνακα.

$minute	{ $minute: <dateExpression> }	Επιστρέφει τα λεπτά μιας ημερομηνίας ως έναν αριθμό από το 0 έως το 59.
$mod	{ field: { $mod: [divisor, remainder] } }	Επιστρέφει το υπόλοιπο της διαίρεσης του πρώτου αριθμού με τον δεύτερο.
$month	{ $month: <dateExpression> }	Επιστρέφει το υπόλοιπο της πρώτης αριθμητικής παράστασης διαιρεμένης με τη δεύτερη.
$mul	{ $mul: { <field1>: <number1>, ... } }	Πολλαπλασιάζει την τιμή του πεδίου με το καθορισμένο ποσό.
$multiply	{ $multiply: [<expression1>, <expression2>, ...] }	Πολλαπλασιάζει αριθμούς για να επιστρέψει το γινόμενό τους.
$ne	{ field: { $ne: value } }	Επιστρέφει την τιμή true εάν οι τιμές δεν είναι ισοδύναμες.

$near	{
	<location field>: {
	$near: {
	$geometry: {
	type: "Point" ,
	coordinates: [<longitude> , <latitude>]
	},
		Επιστρέφει γεωχωρικά αντικείμενα που βρίσκονται κοντά σε ένα σημείο. Απαιτεί ένα γεωχωρικό δείκτη. Οι δείκτες 2dsphere και 2d υποστηρίζουν το $near.

	$maxDistance: <distance in meters>, $minDistance: <distance in meters> } } }	

$nearSphere	{
	$nearSphere: {
	$geometry: {
	type : "Point",
	coordinates : [<longitude>, <latitude>]
	},
	$minDistance: <distance in meters>,
	$maxDistance: <distance in meters>
	}
	}
		Επιστρέφει γεωχωρικά αντικείμενα κοντά σε ένα σημείο σε έναν σφαιρικό χώρο. Απαιτεί ένα γεωχωρικό ευρετήριο. Τα ευρετήρια 2dsphere και 2d υποστηρίζουν το $nearSphere.

$nin	{ field: { $nin: [<value1>, <value2> ... <valueN>] } }	Δεν ταιριάζει με καμία από τις τιμές που καθορίζονται σε έναν πίνακα.
$nor	{ $nor: [{ <expression1> }, { <expression2> }, ... { <expressionN> }] }	Ενώνει όρους ερωτήματος με μια λογική NOR και επιστρέφει όλα τα έγγραφα που αποτυγχάνουν να ταιριάζουν και με τις δύο ρήτρες.
$not	{ $not: [<expression>] }	Αντιστρέφει το αποτέλεσμα μιας έκφρασης ερωτήματος και επιστρέφει εγγραφές που δεν ταιριάζουν με την έκφραση ερωτήματος
$objectToArray	{ $objectToArray: <object> }	Μετατρέπει ένα έγγραφο σε έναν πίνακα εγγράφων που αντιπροσωπεύουν ζεύγη κλειδιού-τιμής.
$or	{ $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }	Συνδυάζει όρους ερωτήματος ερωτήματος με μια λογική OR και επιστρέφει όλα τα έγγραφα που ταιριάζουν με τις συνθήκες οποιουδήποτε όρου. Επιστρέφει true όταν οποιαδήποτε από τις εκφράσεις του αξιολογείται ως true.
$out	{ $out: { db: "<output-db>", coll: "<output-collection>" } }	Γράφει τα αποτελέσματα του aggregation pipeline σε μια συλλογή (collection). Για να χρησιμοποιηθεί το $out stage, πρέπει να είναι το τελευταίο στάδιο (stage) στο pipeline.
$planCacheStats	{ $planCacheStats: { } }	Επιστρέφει πληροφορίες για την κρυφή μνήμη σχεδιασμού για μια συλλογή.
$pop	{ $pop: { <field>: <-1 | 1>, ... } }	Αφαιρεί το πρώτο ή το τελευταίο στοιχείο ενός πίνακα.

$position	{
	$push: {
	<field>: {
	$each: [<value1>, <value2>, ...],
	$position: <num>
	}
	}
	}
		Τροποποιεί τον τελεστή $push ώστε να καθορίσει τη θέση στον πίνακα για να προσθέσει στοιχεία.

$pow	{ $pow: [<number>, <exponent>] }	Ανεβάζει έναν αριθμό στην καθορισμένη δύναμη.
$project	{ $project: { <specification(s)> } }	Ανασχηματίζει κάθε έγγραφο στη ροή, όπως την πρόσθεση νέων πεδίων ή την αφαίρεση υπαρχόντων πεδίων. Για κάθε εισαγόμενο έγγραφο, εξάγει ένα έγγραφο.

$pull	{ $pull: { <field1>: <value|condition>, <field2>: <value|condition>, ... } }	Καταργεί όλα τα στοιχεία πίνακα που ταιριάζουν με ένα καθορισμένο ερώτημα.
$pullAll	{ $pullAll: { <field1>: [<value1>, <value2> ...], ... } }	Καταργεί όλες τις τιμές που ταιριάζουν από έναν πίνακα.
$push	{ $push: { <field1>: <value1>, ... } }	Προσθέτει ένα στοιχείο σε έναν πίνακα. Επιστρέφει έναν πίνακα τιμών εκφράσεων για έγγραφα σε κάθε ομάδα.
$rand	{ $rand: {} }	Επιστρέφει ένα τυχαίο δεκαδικό αριθμό από το 0 έως το 1.
$range	{ $range: [<start>, <end>, <non-zero step>] }	Δημιουργεί έναν πίνακα με μια ακολουθία ακεραίων αριθμών βάσει των εισαγόμενων παραμέτρων.
$rank	{ $rank: { } }	Επιστρέφει τη θέση ενός εγγράφου σε σχέση με άλλα ταξινομημένα έγγραφα.
$redact	{ $redact: <expression> }	Αναδιαμορφώνει κάθε έγγραφο στη ροή περιορίζοντας το περιεχόμενο για κάθε έγγραφο με βάση τις πληροφορίες που είναι αποθηκευμένες στα ίδια τα έγγραφα

$reduce	{
	$reduce: {
	input: <array>,
	initialValue: <expression>,
	in: <expression>
	}
	}
		Εφαρμόζει μια έκφραση σε κάθε στοιχείο ενός πίνακα και συνδυάζει τα αποτελέσματα σε μια μοναδική τιμή.

$regex	{ <field>: { $regex: /pattern/, $options: '<options>' } } { <field>: { $regex: 'pattern', $options: '<options>' } } { <field>: { $regex: /pattern/<options> } }	Επιλέγει έγγραφα όπου οι τιμές τους αντιστοιχούν σε μια καθορισμένη κανονική έκφραση (regular expression).
$regexFind	{ $regexFind: { input: <expression> , regex: <expression>, options: <expression> } }	Εφαρμόζει μια κανονική έκφραση (regex) σε μια συμβολοσειρά και επιστρέφει πληροφορίες για την πρώτη ταιριαστή υποσυμβολοσειρά.
$regexFindAll	{ $regexFindAll: { input: <expression> , regex: <expression>, options: <expression> } }	Εφαρμόζει μια κανονική έκφραση (regex) σε μια συμβολοσειρά και επιστρέφει πληροφορίες για όλες τις ταιριαστές υποσυμβολοσειρές.
$regexMatch	{ $regexMatch: { input: <expression> , regex: <expression>, options: <expression> } }	Εφαρμόζει μια κανονική έκφραση (regex) σε μια συμβολοσειρά και επιστρέφει ένα boolean που υποδηλώνει εάν βρέθηκε μια ταιριαστή υποσυμβολοσειρά ή όχι.
$rename	{$rename: { <field1>: <newName1>, <field2>: <newName2>, ... } }	Μετονομάζει ένα πεδίο.
$replaceAll	{ $replaceAll: { input: <expression>, find: <expression>, replacement: <expression> } }	Αντικαθιστά όλες τις εμφανίσεις μιας ταιριαστής συμβολοσειράς σε ένα κείμενο.
$replaceOne	{ $replaceOne: { input: <expression>, find: <expression>, replacement: <expression> } }	Αντικαθιστά την πρώτη εμφάνιση μιας ταιριαστής συμβολοσειράς σε ένα κείμενο.
$replaceRoot	{ $replaceRoot: { newRoot: <replacementDocument> } }	Αντικαθιστά ένα έγγραφο με το καθορισμένο ενσωματωμένο έγγραφο.
$replaceWith	{ $replaceWith: <replacementDocument> }	Αντικαθιστά ένα έγγραφο με το καθορισμένο ενσωματωμένο έγγραφο.
$reverseArray	{ $reverseArray: <array expression> }	Επιστρέφει έναν πίνακα με τα στοιχεία σε αντίστροφη σειρά.

$round	{ $round : [<number>, <place>] }	Στρογγυλοποιεί έναν αριθμό σε έναν ακέραιο αριθμό ή σε έναν καθορισμένο αριθμό δεκαδικών ψηφίων.
$rtrim	{ $rtrim: { input: <string>, chars: <string> } }	Αφαιρεί κενά ή τους καθορισμένους χαρακτήρες από το τέλος μιας συμβολοσειράς.
$sample	{ $sample: { size: <positive integer N> } }	Επιλέγει τυχαία τον καθορισμένο αριθμό εγγράφων από την εισαγωγή του.
$sampleRate	{ $sampleRate: <non-negative float> }	Επιλέγει τυχαία έγγραφα σε μια δεδομένη συχνότητα.
$second	{ $second: <dateExpression> }	Επιστρέφει τα δευτερόλεπτα για μια ημερομηνία ως έναν αριθμό μεταξύ 0 και 60.
$set	{ $set: { <field1>: <value1>, ... } }	Ορίζει την τιμή ενός πεδίου σε ένα έγγραφο. Προσθέτει νέα πεδία σε έγγραφα.
$setDifference	{ $setDifference: [<expression1>, <expression2>] }	Επιστρέφει ένα σύνολο με τα στοιχεία που εμφανίζονται στο πρώτο σύνολο αλλά όχι στο δεύτερο σύνολο.
$setEquals	{ $setEquals: [<expression1>, <expression2>, ...] }	Επιστρέφει true εάν τα σύνολα που δίνονται ως είσοδο έχουν τα ίδια μοναδικά στοιχεία.

$setField	{
	$setField: {
	field: <String>,
	input: <Object>,
	value: <Expression>
	}
	}
		Προσθέτει, ενημερώνει ή αφαιρεί ένα συγκεκριμένο πεδίο σε ένα έγγραφο.

$setIntersection	{ $setIntersection: [<array1>, <array2>, ...] }	Επιστρέφει ένα σύνολο με τα στοιχεία που εμφανίζονται σε όλα τα σύνολα εισόδου.
$setIsSubset	{ $setIsSubset: [<expression1>, <expression2>] }	Επιστρέφει (true) εάν όλα τα στοιχεία του πρώτου συνόλου εμφανίζονται στο δεύτερο σύνολο.
$setOnInsert	db.collection.updateOne(<query>, { $setOnInsert: { <field1>: <value1>, ... } }, { upsert: true })	Ορίζει την τιμή ενός πεδίου εάν μια ενημέρωση έχει ως αποτέλεσμα την εισαγωγή ενός εγγράφου. Δεν επηρεάζει τις λειτουργίες ενημέρωσης που τροποποιούν υπάρχοντα έγγραφα.
$setUnion	{ $setUnion: [<expression1>, <expression2>, ...] }	Επιστρέφει ένα σύνολο με στοιχεία που εμφανίζονται σε οποιοδήποτε από τα εισαγόμενα σύνολα.

$setWindowFields	{
	$setWindowFields: {
	partitionBy: <expression>,
	sortBy: {
	<sort field 1>: <sort order>,
	<sort field 2>: <sort order>,
	...,
	<sort field n>: <sort order>
	},
	output: {
	<output field 1>: {
	<window operator>: <window operator parameters>,
	window: {
	documents: [<lower boundary>, <upper boundary>],
	range: [<lower boundary>, <upper boundary>],
	unit: <time unit>
	}
		Ομαδοποιεί έγγραφα σε παράθυρα και εφαρμόζει έναν ή περισσότερους τελεστές στα έγγραφα σε κάθε παράθυρο.

	}, <output field 2>: { ... }, ... <output field n>: { ... } } } }	

$shift	{
	$shift: {
	output: <output expression>,
	by: <integer>,
	default: <default expression>
	}
	}
		Επιστρέφει την τιμή από μια έκφραση που εφαρμόζεται σε ένα έγγραφο σε μια συγκεκριμένη θέση σχετικά με το τρέχον έγγραφο στην έξοδο

$size	db.collection.find({ field: { $size: 2 } });	Επιλέγει έγγραφα εάν το πεδίο πίνακα έχει καθορισμένο μέγεθος. Επιστρέφει τον αριθμό των στοιχείων στον πίνακα.
$skip	{ $skip: <positive 64-bit integer> }	Παραλείπει τα πρώτα n έγγραφα όπου n είναι ο καθορισμένος αριθμός παράβλεψης και μεταβιβάζει τα υπόλοιπα έγγραφα χωρίς τροποποίηση στη διοχέτευση.
$slice	{ $slice: [<array>, <n>] }	Τροποποιεί τον τελεστή $push για να περιορίσει το μέγεθος των ενημερωμένων πινάκων. Επιστρέφει ένα υποσύνολο ενός πίνακα. Περιορίζει τον αριθμό των στοιχείων που προβάλλονται από έναν πίνακα.

$sort	{
	$push: {
	<field>: {
	$each: [<value1>, <value2>, ...],
	$sort: <sort specification>
	}
	}
	}
		Τροποποιεί τον τελεστή $push ώστε να αναδιατάσσει τα έγγραφα που αποθηκεύονται σε έναν πίνακα. Αναδιατάσσει τη ροή του εγγράφου με ένα καθορισμένο κλειδί ταξινόμησης.

$sortArray	$sortArray: {
	input: <array>,
	sortBy: <sort spec>
	}
		Ταξινομεί έναν πίνακα με βάση τα στοιχεία του.

$sortByCount	{ $sortByCount: <expression> }	Ομαδοποιεί τα εισερχόμενα έγγραφα με βάση την τιμή μιας καθορισμένης έκφρασης και, στη συνέχεια, υπολογίζει τον αριθμό των εγγράφων σε κάθε ξεχωριστή ομάδα.
$split	{ $split: [<string expression>, <delimiter>] }	Χωρίζει μια συμβολοσειρά σε υποσυμβολοσειρές με βάση έναν διαχωριστικό χαρακτήρα. Επιστρέφει έναν πίνακα υποσυμβολοσειρών.
$sqrt	{ $sqrt: <number> }	Υπολογίζει την τετραγωνική ρίζα.
$stdDevPop	{ $stdDevPop: <expression> }	Επιστρέφει την τυπική απόκλιση του πληθυσμού των εισαγόμενων τιμών.
$stdDevSamp	{ $stdDevSamp: <expression> }	Επιστρέφει την τυπική απόκλιση δείγματος των εισαγόμενων τιμών.
$strcasecmp	{ $strcasecmp: [<expression1>, <expression2>] }	Πραγματοποιεί σύγκριση συμβολοσειρών χωρίς διάκριση πεζών-κεφαλαίων και επιστρέφει: 0 εάν οι δύο συμβολοσειρές είναι ισοδύναμες, 1 εάν η πρώτη συμβολοσειρά είναι μεγαλύτερη

		από τη δεύτερη και -1 εάν η πρώτη συμβολοσειρά είναι μικρότερη από τη δεύτερη.
$strLenBytes	{ $strLenBytes: <string expression> }	Επιστρέφει τον αριθμό των UTF-8 κωδικοποιημένων bytes σε μια συμβολοσειρά.
$strLenCP	{ $strLenCP: <string expression> }	Επιστρέφει τον αριθμό των UTF-8 κωδικοποιημένων σημείων σε μια συμβολοσειρά.
$substrBytes	{ $substrBytes: [<string expression>, <byte index>, <byte count>] }	Επιστρέφει την υποσυμβολοσειρά μιας συμβολοσειράς.
$substrCP	{ $substrCP: [<string expression>, <code point index>, <code point count>] }	Επιστρέφει το υποσύμβολο συμβολοσειράς.
$subtract	{ $subtract: [<expression1>, <expression2>] }	Εκτελεί αφαίρεση μεταξύ δύο τιμών, ανάλογα με τον τύπο τους.
$sum	{ $sum: <expression> }	Επιστρέφει το άθροισμα αριθμητικών τιμών. Αγνοεί μη αριθμητικές τιμές.

$switch	$switch: {
	branches: [
	{ case: <expression>, then: <expression> },
	{ case: <expression>, then: <expression> },
	...
	],
	default: <expression>
	}
		Όταν βρίσκει μια έκφραση η οποία εκτιμάται ως αληθής (true), το $switch εκτελεί μια καθορισμένη έκφραση και βγαίνει από τη ροή ελέγχου.

$text	{
	$text:
	{
	$search: <string>,
	$language: <string>,
	$caseSensitive: <boolean>,
	$diacriticSensitive: <boolean>
	}
	}
		Πραγματοποιεί αναζήτηση κειμένου.

$toBool	{
	$toBool: <expression>
	}
		Μετατρέπει την τιμή boolean.

$toDate	{
	$toDate: <expression>
	}
		Μετατρέπει την τιμή σε ημερομηνία

$toDecimal	{
	$toDecimal: <expression>
	}
		Μετατρέπει την τιμή σε Decimal.

$toDouble	{
	$toDouble: <expression>
	}
		Μετατρέπει την τιμή σε διπλή.

$toLong	{
	$toLong: <expression>
	}
		Converts value to a long.

$toLower	{ $toLower: <expression> }	Μετατρέπει μια συμβολοσειρά σε πεζά γράμματα.

$toObjectId	{
	$toObjectId: <expression>
	}
		Μετατρέπει την τιμή σε ObjectId.

$top	{ $top: { sortBy: { <field1>: <sort order>, <field2>: <sort order> ... }, output: <expression>	Επιστρέφει το πρώτο στοιχείο στην ομάδα σύμφωνα με την καθορισμένη σειρά ταξινόμησης.

	} }	

$topN	{
	$topN:
	{
	n: <expression>,
	sortBy: { <field1>: <sort order>, <field2>: <sort order> ... },
	output: <expression>
	}
	}
		Επιστρέφει μια συλλογή των n πρώτων πεδίων στην ομάδα, σύμφωνα με την καθορισμένη σειρά ταξινόμησης.

$toString	{
	$toString: <expression>
	}
		Μετατρέπει μια τιμή σε μια συμβολοσειρά.

$toUpper	{ $toUpper: <expression> }	Μετατρέπει μια συμβολοσειρά σε κεφαλαία γράμματα.
$trim	{ $trim: { input: <string>, chars: <string> } }	Αφαιρεί τα κενά ή τους καθορισμένους χαρακτήρες από την αρχή και το τέλος μιας συμβολοσειράς.
$trunc	{ $trunc : [<number>, <place>] }	Αποκόβει έναν αριθμό στο πλησιέστερο ακέραιο ή σε ένα συγκεκριμένο δεκαδικό ψηφίο.
$tsIncrement	{ $tsIncrement: <expression> }	Επιστρέφει τη διαδοχική τακτική από ένα χρονικό στίγμα ως έναν ακέραιο.
$tsSecond	{ $tsSecond: <expression> }	Επιστρέφει τα δευτερόλεπτα από ένα χρονικό στίγμα ως έναν ακέραιο.
$type	{ field: { $type: <BSON type> } }	Επιλέγει έγγραφα αν ένα πεδίο είναι του καθορισμένου τύπου δεδομένων. Επιστρέφει τον τύπο δεδομένων BSON του πεδίου.
$unset	{ $unset: { <field1>: "", ... } }	Αφαιρεί/εξαιρεί πεδία από έγγραφα.

$unsetField	{
	$unsetField: {
	field: <String>,
	input: <Object>,
	}
	}
		Αφαιρεί ένα συγκεκριμένο πεδίο από ένα έγγραφο.

$unwind	{ $unwind: <field path> }	Αποδομεί ένα πεδίο πίνακα από τα εισερχόμενα έγγραφα, προκειμένου να εξάγει ένα έγγραφο για κάθε στοιχείο του πίνακα.
$week	{ $week: <dateExpression> }	Επιστρέφει τον αριθμό της εβδομάδας για μια ημερομηνία ως έναν αριθμό μεταξύ 0 (και 53.
$where	{ $where: <string|JavaScript Code> }	Ταιριάζει έγγραφα που ικανοποιούν μια έκφραση JavaScript.
$year	{ $year: <dateExpression> }	Επιστρέφει το έτος μιας ημερομηνίας σε μορφή αριθμού.

$zip	{
	$zip: {
	inputs: [<array expression1>, ...],
	useLongestLength: <boolean>,
	defaults: <array expression>
	}
	}
		Συγχωνεύει δύο πίνακες μαζί.

[37]

Παράρτημα 2 - Κώδικας Προγράμματος

```
import tweepy as tw
import pymongo
from pymongo import MongoClient
import pprint
from tkinter import *
from tkinter import messagebox

client = MongoClient()
db = client.tweet
dbtweets=db.tweets

consumer_key= 'n3zJrpPGimkeMiViYTEjzc0nm'
consumer_secret=
'mO4Vli9sc0tRZzzPMQFcbEMcokyYWejNm9gq1G8HeoV3x23z7C'
access_token= '1332804311752052738-feAbedMtVFEd0JiAiYuYqbqMYwNoIO'
access_token_secret= 'peLikF10VZGSv9SjIOT3Oczq94XncSVj8ktueWv3izK2Z'

auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)

window = Tk()
window.geometry('400x160')
window.title('MongoDB - Twitter')
window.columnconfigure(0, weight=1)
window.columnconfigure(1, weight=3)

lbl = Label(window, text="Search HashTag:", font=("Arial", 11))
lbl.grid(column=0, row=0, sticky=W, padx=5, pady=5)

lblTxtNew = Label(window, text="Search Text:", font=("Arial", 11))
lblTxtNew.grid(column=0, row=1, sticky=W, padx=5, pady=5)

txt = Entry(window, font=("Arial", 11))
txt.grid(column=1, row=0, sticky=E, padx=5, pady=5)

txtNew = Entry(window, font=("Arial", 11))
txtNew.grid(column=1, row=1, sticky=E, padx=5, pady=5)

lb2 = Label(window, text="Πλήθος Αναζητήσεων:", font=("Arial", 11))
lb2.grid(column=0, row=2, sticky=W, padx=5, pady=5)

txt2 = Entry(window, font=("Arial", 11))
txt2.grid(column=1, row=2, sticky=E, padx=5, pady=5)
```

```
#methodos gia anazitisi sti MongoDB
def search():
    res = txt.get()
    resNew = txtNew.get()
    number = txt2.get()
    print("Ksekina i anazitisi - hashtag "+res+ " - text" +resNew)
    if len(res) > 0 and len(resNew) > 0:
        messagebox.showinfo('Ενημερωτικό',"Δεν είναι δεκτή η χρήση και των 2 πεδίων
ταυτόχρονα.")
    elif len(number) == 0 :
        messagebox.showinfo('Ενημερωτικό',"Παρακαλώ προσθέστε πλήθος
αναζητήσεων")
    elif len(number) > 0 and number.isdigit() == False:
        messagebox.showinfo ('Ενημερωτικό',"Θα πρέπει να συμπληρωθεί ένας
αριθμός.")
    elif len(res) == 0 and len(resNew) == 0:
        messagebox.showinfo('Ενημερωτικό',"Θα πρέπει να συμπληρωθεί ένα από τα δύο
πεδία αναζήτησης.")
    else:
        plushast = res if res.startswith('#') else "#"+res
        resFinal = resNew if bool(resNew) else plushast
        root = Tk()
        S = Scrollbar(root)
        T = Text(root, height=16, width=150)
        root.title('Results of MongoDB')
        S.pack(side=RIGHT, fill=Y)
        T.pack(side=LEFT, fill=Y)
        S.config(command=T.yview)
        T.config(yscrollcommand=S.set)
        quote = """"""
        counter=0
        for post in dbtweets.find({"text": { "$regex": resFinal, "$options" :'i'
}}).limit(int(number)):
            counter+=1
            quote += pprint.pformat(post)
            quote +='\n'
            quote += pprint.pformat('----------------------------------------------')
            quote +='\n'
            pprint.pprint(post)
        print("Telos anazitisi")
        T.insert(END, quote)
        messagebox.showinfo('Ενημερωτικό',"Βρέθηκαν "+str(counter)+" Tweet με
αναζήτηση "+res +resNew)
#methodos gia anazitisi Tweet kai pros8iki sti bash dedomenon an den iparxoun idi
def clicked():
```

```
   res = txt.get()
   resNew = txtNew.get()
   try:
      numberOfTweet=int(txt2.get())
      if len(resNew) > 0:
         messagebox.showinfo('Ενημερωτικό',"Το search στο Tweet λειτουργεί μόνο με
Hashtag")
      elif len(res) > 0 and len(resNew) > 0:
         messagebox.showinfo('Ενημερωτικό',"Δεν είναι δεκτή η χρήση και των 2
πεδίων ταυτόχρονα.")
      elif len(res)>0:#i anazitisi 8a ginei an exei simplirosei keimeno
         search_words = res if res.startswith('#') else "#"+res #το hashtag που
ψάχνουμε
         date_since = "2022-08-25" #από πότε το ψάχνουμε

         tweets = tw.Cursor(api.search_tweets,
                  q=search_words,
                  lang="en").items(numberOfTweet) #πόσα αποτελέσματα
         counterInserted=0
         counterFound=0
         for tweet in tweets:
            counterFound+=1
            if (not dbtweets.find_one({"_id": tweet.id})):
               counterInserted+=1
               row={"_id":tweet.id, "name":tweet.user.screen_name,

"location":tweet.user.location,"created":tweet.created_at,"text":tweet.text.encode("ut
f-8").decode("utf-8")}
               result=dbtweets.insert_one(row)
               print("First article key is: {}".format(result.inserted_id))
               print("tweet Id: ", tweet.id, end='\n')
               print("name: ", tweet.user.screen_name, end='\n')
               print("location: ", tweet.user.location, end='\n')
               print("date created: ", tweet.created_at, end='\n')
               print("text: ", tweet.text.encode("utf-8").decode("utf-8"), end='\n') #τα
εμφανίζω και μετά τα προσθέτω στη βάση
               print("-----------------------------------------------------------------")
            else:
               print("Id " + str(tweet.id) + " already in database", end='\n')
         messagebox.showinfo('Ενημερωτικό',"Βρέθηκαν " +str(counterFound)+".
Προστέθηκαν "+
         str(counterInserted)+" tweet στη MongoDB")
      else:
         messagebox.showinfo('Ενημερωτικό',"Δεν προσθέσατε κείμενο αναζήτησης")
   except ValueError:
      messagebox.showinfo('Ενημερωτικό',"Ποσθέστε πλήθος αναζητήσεων")
```

```
btn = Button(window, text=" Tweet->MongoDB ", command=clicked,font=("Arial", 12), bg='red')
btn.grid(column=0,row=4, sticky= W, padx=5, pady=5)
btn1 = Button(window, text="SearchDB", command=search,font=("Arial", 12), bg='yellow')
btn1.grid(column=1,row=4, sticky= E, padx=5, pady=5)
window.mainloop()
```