ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΠΣ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ ΜΕΣΟΛΟΓΓΙ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ NoSQL ΣΥΣΤΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ

ΧΑΡΙΚΛΕΙΑ ΚΑΝΙΟΥΡΗ

Μεσολόγγι 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΠΣ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ ΜΕΣΟΛΟΓΓΙ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ NoSQL ΣΥΣΤΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ

ΧΑΡΙΚΛΕΙΑ ΚΑΝΙΟΥΡΗ

Επιβλέπων καθηγητής

ΝΙΚΟΛΑΟΣ ΚΑΡΟΥΣΟΣ

Μεσολόγγι 2020

UNIVERSITY OF PATRAS

SCHOOL OF ECONOMICS & BUSINESS

DEPARTMENT OF MANAGEMENT SCIENCE AND TECHNOLOGY

FORMER DEPARTMENT OF BUSINESS ADMINISTRATION AT MESSOLONGHI

THESIS

STUDY OF NOSQL DATABASE MANAGEMENT SYSTEMS IN THE CLOUD

CHARIKLIA KANIOURI

Messolonghi 2020

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Νικόλαο Καρούσο για την καθοδήγηση και την πολύτιμη βοήθεια του στην επίλυση των διαφόρων θεμάτων κατά την διάρκεια υλοποίησης της πτυχιακής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου για την ηθική συμπαράσταση που μου προσέφεραν και την υπομονή που επέδειξαν όλο αυτό το διάστημα.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία έχει γίνει μια προσπάθεια να παρουσιαστεί η σύγχρονη τάση όσον αφορά στις βάσεις δεδομένων καθώς οι σημερινές εφαρμογές έχουν αυξημένες υπολογιστικές και αποθηκευτικές απαιτήσεις. Έτσι παρουσιάζονται οι σημαντικότερες NoSQL βάσεις δεδομένων που μπορούν να αναπτυχθούν στο υπολογιστικό νέφος.

Στο πρώτο μέρος της εργασίας και μετά από μελέτη της σχετικής βιβλιογραφίας μέσα από πηγές του διαδικτύου, συνελέγησαν τα χαρακτηριστικά των NoSQL συστημάτων διαχείρισης βάσεων δεδομένων που μπορούν να λειτουργήσουν στο υπολογιστικό νέφος.

Στο δεύτερο μέρος της εργασίας παρουσιάζονται δύο παρόμοιες βάσεις δεδομένων, μία στην Access (σχεσιακή βάση δεδομένων) και μία στην MongoDB Atlas (NoSQL βάση δεδομένων) και η πραγματοποίηση μιας συγκριτικής μελέτης πάνω σε αντίστοιχες λειτουργίες των δύο βάσεων δεδομένων.

Στο τέλος της εργασίας παρουσιάζονται τα συμπεράσματα τα οποία εξήχθησαν από την παραπάνω μελέτη.

ABSTRACT

In this thesis there has been an attempt to present the modern tendency as far as it concerns the databases, since the applications nowadays have increased calculating and storaging demands. Thus, the most important NoSQL databases that can be developed in the cloud are presented.

In the first part and after a study of the relevant bibliography through internet sources, there have been collected the characteristics of NoSQL database management systems that can be operated in the cloud.

In the second part of this thesis, two similar databases are presented, one in Access (relational database) and one in MongoDB Atlas (NoSQL database) and a comparatively study on equivalent operations on those two databases has been accomplished.

At the end the extracted conclusions of this study are presented.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

API	Application Programming Interface
Apps	Applications
DaaS	Desktop as a Service
IaaS	Infracture as a Service
JSON	Javascript Object Notation
NoSQL	Not only SQL
PaaS	Platform as a Service
RDBMS	Relational Database Management System
SaaS	Service as a Service
SQL	Structured Query Language
SSD	Solid State Drive
VM	Virtual Machine

ΑΠΟΔΟΣΗ ΟΡΩΝ

Analytics	Αναλυτικά στοιχεία
Apps	Εφαρμογές
Backup	Αντίγραφο ασφαλείας
Big Data	Μεγάλος όγκος δεδομένων
Cloud computing	Υπολογιστικό νέφος
Cloud server instances	Εικονικοί εξυπηρετητές στο υπολογιστικό νέφος
Cluster machines	Κατανεμημένο σύστημα υπολογιστών
Data store	Αποθήκη δεδομένων
Driver	Πρόγραμμα οδήγησης
Geospatial	Γεωχωρική
Load balancing	Εξισορρόπηση φόρτου
Metadata	Μεταδεδομένα
Native	Ενσωματωμένο / Εγγενές
RDBMS	Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων
Replica	Αντίγραφο
SQL	Δομημένη γλώσσα ερωτημάτων
Timestamping	Χρονοσήμανση
Virtual Machine	Εικονική Μηχανή
Web browser	Φυλλομετρητής

Εισαγωγή

Εδώ και δεκαετίες έγινε φανερή η ανάγκη για συλλογή, αποδοτική αποθήκευση και ανάκτηση δεδομένων. Ο ολοένα αυξανόμενος όγκος των δεδομένων δημιούργησε την ανάγκη ανάπτυξης συστημάτων αποτελεσματικής τους διαχείρισης. Έτσι δημιουργήθηκαν τα Συστήματα Διαχείρισης Βάσεων Δεδομένων (DBMS). Το πιο διαδεδομένο μοντέλο των συστημάτων διαχείρισης βάσεων δεδομένων είναι το σχεσιακό μοντέλο (RDBMS) το οποίο χρησιμοποιείται ακόμα από το μεγαλύτερο πλήθος επιχειρήσεων και όχι μόνο.

Παρ' όλα αυτά, τα σημερινά υπολογιστικά συστήματα καλούνται να διαχειριστούν έναν μεγάλο όγκο δεδομένων. Οι εφαρμογές που χρησιμοποιούνται από τους χρήστες απαιτούν τη συλλογή, διαχείριση και επεξεργασία μεγάλων ποσοτήτων δεδομένων σε μικρό χρονικό διάστημα. Έτσι δημιουργήθηκε η ανάγκη για μοντέλα δεδομένων που δεν είναι σχεσιακά. Τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων είναι πιο ευέλικτα και παρουσιάζουν αρκετά πλεονεκτήματα σε σχέση με τα υπάρχοντα σχεσιακά.

Επίσης, λόγω του μεγάλου κόστους και της δυσκολίας διαχείρισης των διάσπαρτων παραγόμενων πληροφοριών δημιουργείται όλο και περισσότερο η τάση χρήσης τεχνολογιών που μπορούν να αναπτυχθούν στο υπολογιστικό νέφος. Με τον τρόπο αυτό οι χρήστες μπορούν να αποθηκεύουν δεδομένα τα οποία βρίσκονται διάσπαρτα και απομακρυσμένα, γρήγορα και χωρίς μεγάλο οικονομικό κόστος μιας και δεν επιβαρύνονται με κόστος εξοπλισμού και συντήρησης.

Στην εργασία αυτή γίνεται μια συνοπτική παρουσίαση των NoSQL συστημάτων διαχείρισης δεδομένων και μια προσπάθεια κατανόησης κάποιων βασικών λειτουργιών των σχεσιακών και των μη σχεσιακών συστημάτων διαχείρισης δεδομένων στο υπολογιστικό νέφος χρησιμοποιώντας την Access και την MongoDB Atlas. Στη συνέχεια γίνεται η παρουσίαση κάποιων συμπερασμάτων ως προς την λειτουργία τους.

1. Περιγραφή Βασικών Εννοιών

1.1 Βάση Δεδομένων

Μια βάση δεδομένων είναι ένα σύνολο δεδομένων οργανωμένων και αποθηκευμένων σε έναν υπολογιστή με τέτοιο τρόπο ώστε να είναι εύκολα προσπελάσιμα και διαχειρίσιμα. Οργανωμένα με τρόπο ώστε να μπορούν να ενημερωθούν εύκολα από τον διαχειριστή της βάσης δεδομένων. [1]

Η βάση δεδομένων παρέχει μέσω του σχεδιασμού και του τρόπου ιεράρχησης των δεδομένων, τα αποκαλούμενα Συστήματα Διαχείρισης Βάσεων Δεδομένων (DataBase Management Systems - DBMS), τα οποία δίνουν την δυνατότητα γρήγορης άντλησης και ανανέωσης των δεδομένων.

1.2 Μοντέλα Βάσεων Δεδομένων

Ο άνθρωπος από την αρχή που εμφανίστηκε είχε την ανάγκη να συλλέγει και να αποθηκεύει πληροφορίες που τον αφορούσαν. Από τότε μέχρι σήμερα αναπτύχθηκαν διάφορα μοντέλα διαχείρισης δεδομένων, τα οποία στην πορεία των χρόνων διαφοροποιήθηκαν, βελτιώθηκαν και εναρμονίστηκαν με τις ανάγκες της εποχής.
Τα κυριότερα μοντέλα βάσεων δεδομένων [2], [3] είναι:

- Το ιεραρχικό (hierarchical model)
- Το δικτυακό (network model)
- Το σχεσιακό (relational model)
- Το αντικειμενοστραφές (object – oriented model)
- Το αντικειμενοσχεσιακό (object relational model)
- Το NoSQL (not only relational)

Στο **ιεραρχικό μοντέλο** τα δεδομένα ήταν οργανωμένα με τη δομή ενός ανεστραμμένου δέντρου και αυτό το έκανε δυσλειτουργικό μιας και όλες οι αναζητήσεις θα έπρεπε να ξεκινούν από την ρίζα (root), αν σβηνόταν κάποιος πίνακας θα έπρεπε να σβηστούν και οι ‘απόγονοί’ του και επίσης ήταν πολύ δύσκολη η αλλαγή της δομής του ‘δέντρου’.

Στο **δικτυακό μοντέλο** η αναζήτηση στα δεδομένα δεν ήταν υποχρεωτικό να ξεκινούν από την ρίζα (root) μιας και ουσιαστικά η δομή του ήταν συνδυασμός ιεραρχικών βάσεων δεδομένων.

Το **σχεσιακό μοντέλο** βασίζεται σε σχέσεις που αναπτύσσονται μεταξύ πινάκων χωρίς να υπάρχει η έννοια της ιεραρχίας. Η αναζήτηση των δεδομένων βασίζεται στο περιεχόμενο των πινάκων χωρίς να χρειάζεται να ακολουθηθεί κάποια δενδρική διαδρομή. [2]

Στο **αντικειμενοστραφές μοντέλο** η πληροφορία θεωρείται ως ανεξάρτητο αντικείμενο. Με το μοντέλο αυτό το οποίο είναι πιο πρόσφατο, παρέχεται η δυνατότητα διαχείρισης πολύπλοκων δομών δεδομένων αλλά και η κατανομή της πληροφορίας σε πολλαπλά συστήματα, χρησιμοποιεί λογισμικό που έχει αναπτυχθεί

σε κάποια αντικειμενοστραφή γλώσσα προγραμματισμού. Είναι πολύ αποτελεσματικό μοντέλο σε εφαρμογές που υπάρχουν πολύπλοκες και δυναμικές δομές δεδομένων.

Στο **αντικειμενοσχεσιακό μοντέλο** οι βάσεις δεδομένων ουσιαστικά είναι σχεσιακές αλλά συνοδεύονται από κατάλληλο λογισμικό διαχείρισης ώστε να συμπεριφέρεται ως μια βάση δεδομένων μερικώς αντικειμενοστραφή. Είναι εύκολες στη χρήση τους και αυτό τις έχει κάνει αποδεκτές, όμως η ταχύτητα απόκρισης τους είναι μικρή ειδικά σε πολύπλοκες δομές δεδομένων. [3]

Το **NoSQL μοντέλο** βάσεων δεδομένων εμφανίστηκε αργότερα, στις αρχές του 21ου αιώνα και είναι ένα μοντέλο στο οποίο θα αναφερθούμε παρακάτω.

1.3 Σχεσιακές Βάσεις Δεδομένων

Με τον όρο **σχεσιακή βάση δεδομένων** εννοούμε μία συλλογή δεδομένων οργανωμένη σε συσχετισμένους πίνακες που παρέχει ταυτόχρονα ένα μηχανισμό για ανάγνωση, εγγραφή, τροποποίηση ή και πιο πολύπλοκες διαδικασίες πάνω στα δεδομένα.
Οι σχεσιακές βάσεις δεδομένων επινοήθηκαν το 1970 από τον Έντκαρ Κοντ και μέχρι σήμερα αποτελούν την κυρίαρχη τεχνολογία των βάσεων δεδομένων.

Οι πίνακες μιας σχεσιακής βάσης δεδομένων αποτελούνται από γραμμές (εγγραφές) και στήλες (πεδία) όπου εκεί συγκεντρώνονται τα χαρακτηριστικά ομοειδών αντικειμένων.

Ο αριθμός των πεδίων είναι ίδιος για κάθε εγγραφή ενός αντικειμένου και είναι καθορισμένος από την αρχή της δημιουργίας της βάσης δεδομένων.

Οι εγγραφές στον πίνακα προσδιορίζονται μοναδικά μέσω ενός πεδίου που ονομάζεται κλειδί.

Μέσω της γλώσσας SQL(Structured Query Language) επιτυγχάνεται η πρόσβαση στη βάση δεδομένων, ενώ η συσχέτιση μεταξύ των πινάκων επιτυγχάνεται μέσω κοινών πεδίων των πινάκων. Γίνονται ερωτήσεις είτε από το χρήστη, είτε από λογισμικό προς τη βάση δεδομένων και με την εκτέλεση ερωτημάτων ο χρήστης (ή το λογισμικό που εκπροσωπεί το χρήστη) είναι δυνατόν, ανάλογα με τα δικαιώματά του, να δημιουργήσει, να μεταβάλλει, να διαγράψει δεδομένα στη βάση ή να ανασύρει πληροφορίες με σύνθετα κριτήρια αναζήτησης. [4], [5]

1.4 NoSQL Συστήματα Βάσεων Δεδομένων

Τα τελευταία χρόνια έχουν αναπτυχθεί εφαρμογές οι οποίες έχουν μεγάλες υπολογιστικές και αποθηκευτικές απαιτήσεις. Προκειμένου να καλυφθούν οι ανάγκες της εποχής, δημιουργήθηκαν νέες αρχιτεκτονικές στις βάσεις δεδομένων.

Ο όρος NoSQL αρχικά αναφέρεται ως “non SQL” ή “ως μη σχεσιακή – non relational” βάση δεδομένων. Όμως αναφέρεται και ως “Not only SQL” θέλοντας να

επισημάνουν ότι μπορούν επίσης να υποστηρίξουν SQL query languages αλλά και ότι μπορούν να υποστηρίξουν εφαρμογές που είναι φτιαγμένες με περισσότερες από μία γλώσσες προγραμματισμού, ώστε να εκμεταλλεύονται το γεγονός ότι κάθε γλώσσα προγραμματισμού είναι κατάλληλη για την επίλυση και τακτοποίηση διαφορετικών προβλημάτων (polyglot programming).

Ουσιαστικά δεν υπάρχει ακριβής ορισμός του τι είναι ένα σύστημα NoSQL, όμως ξέρουμε ότι δεν είναι RDBMS και πως χρησιμοποιεί έναν εντελώς διαφορετικό τρόπο διαχείρισης των δεδομένων σε σχέση με μια σχεσιακή βάση δεδομένων.

Μια NoSQL βάση δεδομένων παρέχει έναν μηχανισμό για αποθήκευση και ανάκτηση δεδομένων με τρόπο διαφορετικό από αυτόν που χρησιμοποιεί μια σχεσιακή (SQL) βάση δεδομένων.

Τέτοιες βάσεις δεδομένων υπήρχαν από το 1960, όμως χρησιμοποιούνται ευρέως από την αρχή του 21^{ου} αιώνα, από την ανάγκη των Web 2.0 εταιριών.

Η χρήση των NoSQL βάσεων δεδομένων έχει αυξηθεί λόγω του μεγάλου όγκου δεδομένων και των real-time web εφαρμογών.

Τα δεδομένα μιας NoSQL βάσης δεδομένων γενικώς δεν ακολουθούν δομημένη αρχιτεκτονική, όπως για παράδειγμα οι πίνακες και οι βάσεις δεδομένων δεν χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων, αλλά χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων.

Τα NoSQL συστήματα αναπτύχθηκαν και εξελίχθηκαν παράλληλα με τις μεγαλύτερες εταιρίες πληροφορικής στον κόσμο, όπως η Google ή η Amazon, που διαχειρίζονται έναν τεράστιο όγκο δεδομένων. [6]

1.5 Γιατί κάποιος να επιλέξει ένα NoSQL σύστημα διαχείρισης βάσεων δεδομένων

Οι NoSQL βάσεις δεδομένων προσφέρουν [7] :

Δυνατότητα διαχείρισης μεγάλου όγκου δεδομένων

Απλούστερο σχεδιασμό σε σχέση με τα σχεσιακά συστήματα βάσεων δεδομένων. Μπορούν να διαχειριστούν μεγάλες ποσότητες δεδομένων χωρίς κατ' ανάγκη να διατηρούν μία συγκεκριμένη δομή (schema-less).

Απλούστερη οριζόντια κλιμάκωση – scaling σε cluster machines. Οι μέθοδοι υλοποίησης και εφαρμογής τους, αξιοποιούν μια αρχιτεκτονική που επιτρέπει (και ίσως διευκολύνει) την κατανεμημένη λειτουργία του συστήματος.

Μ' αυτόν τον τρόπο το σύστημα μπορεί να «απογειωθεί» σε επιδόσεις καθώς μπορούν να προστεθούν θεωρητικά άπειροι servers, οι οποίοι κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος. Ακόμη, η ενδεχόμενη αδυναμία λειτουργίας ενός server του συστήματος μπορεί να αντιμετωπισθεί εύκολα.

(Η κλιμάκωση σε cluster machines είναι ένα πρόβλημα στις SQL βάσεις δεδομένων)

Ταχύτερες διαδικασίες καθώς η δομή των δεδομένων είναι διαφορετική από αυτές που χρησιμοποιούνται σε σχεσιακές βάσεις δεδομένων.

Μεγαλύτερη **ευελιξία** στη δομή των δεδομένων.

Καλύτερος έλεγχος στην διαθεσιμότητα και τους περιορισμούς που δημιουργούνται από τις τεχνικές δυσκολίες που παρουσιάζονται όταν μια εφαρμογή είναι γραμμένη σε μια object oriented γλώσσα προγραμματισμού.

Συμφέρουν πολύ από **οικονομικής** άποψης καθώς η υλοποίησή τους τις περισσότερες φορές γίνεται μέσω cloud ή virtual environments χωρίς να απαιτείται η αγορά δαπανηρού λογισμικού. Αν θέλουμε να βελτιώσουμε ένα RDBMS προσθέτουμε καλύτερους επεξεργαστές ή μνήμη RAM, ενώ στα NoSQL συστήματα απλά προσθέτουμε κόμβους ώστε να επεξεργάζονται ακόμη περισσότερα δεδομένα ταυτόχρονα.

Ευκολότερη **συντήρηση,** καθώς είναι σχεδιασμένα να λειτουργούν με τέτοιο τρόπο, ώστε να απαιτούν τη λιγότερη δυνατή διαχείριση από τον ανθρώπινο παράγοντα.

Επιπλέον, οποιαδήποτε **αλλαγή** στο μοντέλο δεδομένων ενός συστήματος RDBMS είναι μία διαδικασία που απαιτεί ένα σεβαστό χρονικό διάστημα που η εφαρμογή θα υπολειτουργεί ή δεν θα λειτουργεί καθόλου, ενώ ένα NoSQL σύστημα έχει πολύ λιγότερους (έως μηδαμινούς) περιορισμούς σε αυτό το ζήτημα. [7]

1.6 Cluster machines - Κατανεμημένο σύστημα υπολογιστών

Είναι ένα σύστημα υπολογιστών συνδεδεμένων μεταξύ τους οι οποίοι δουλεύουν εκτελώντας την ίδια εργασία και οι οποίοι ελέγχονται και σχεδιάζονται μέσω λογισμικού. Έτσι επιτυγχάνεται μια ισορροπημένη ροή των δεδομένων μέσω δικτύου από τον ένα διακομιστή στον άλλο. Οι υπολογιστές αυτοί μπορούν να βρίσκονται τοποθετημένοι σε διαφορετικές γεωγραφικές περιοχές. Η ομαλή λειτουργία τους είναι εξασφαλισμένη ακόμα κι αν κάποιος διακομιστής τεθεί εκτός λειτουργίας, οπότε και ανακατευθύνεται η ροή των δεδομένων.

Ένα κατανεμημένο σύστημα χρησιμοποιείται για να αυξηθεί η αποθηκευτική και υπολογιστική ικανότητα, να διασφαλιστεί η απρόσκοπτη πρόσβαση όλων των υπολογιστών στις διάφορες υπηρεσίες, να εξασφαλιστεί η δυνατότητα επεξεργασίας των δεδομένων που προκύπτουν από μια εφαρμογή.

Ο όγκος των δεδομένων και τα αιτήματα που πρέπει εξυπηρετηθούν καθημερινά είναι πάρα πολλά, οπότε για την καλύτερη διαχείριση τους, θα πρέπει να δημιουργηθούν συστήματα δεδομένων που να ικανοποιούν κάποιες εγγυήσεις. [8]

1.7 Θεώρημα CAP

Το θεώρημα CAP ή αλλιώς Brewer’s theorem [10] αναφέρει πως είναι αδύνατον ένα κατανεμημένο σύστημα υπολογιστών να ικανοποιεί ταυτόχρονα περισσότερες από δύο από τις παρακάτω εγγυήσεις:

Συνέπεια (Consistency) – κάθε κόμβος του συστήματος λαμβάνει την πιο πρόσφατη απάντηση ή ένα λάθος.

Διαθεσιμότητα (Availability) - κάθε κόμβος του συστήματος παίρνει μια απάντηση σχετικά με ένα αίτημα (χωρίς να υπάρχει η εγγύηση ότι πρόκειται για την πιο πρόσφατη απάντηση).

Ανοχή Διαμοιρασμού (Partition Tolerance) – κάθε κόμβος του συστήματος συνεχίζει να λειτουργεί ακόμα κι αν μεμονωμένοι κόμβοι είναι εκτός λειτουργίας λόγω βλαβών του δικτύου, διατηρώντας έτσι τα δύο προηγούμενα χαρακτηριστικά την συνέπεια και την διαθεσιμότητα.

Το είδος της βάσης δεδομένων που θα χρησιμοποιήσουμε εξαρτάται από το πρόβλημα που θέλουμε να λύσουμε.

Σύμφωνα με το θεώρημα CAP, οι δυνατοί συνδυασμοί είναι οι εξής:

CA: Εγγυάται την συνέπεια και την διαθεσιμότητα, όχι όμως και την ανοχή στον διαμοιρασμό. Με τον συνδυασμό αυτό, τα δεδομένα διανέμονται σταδιακά από τον ένα κόμβο στον άλλο με την προϋπόθεση ότι οι κόμβοι είναι online. Όμως σε περίπτωση διαμοιρασμού τα δεδομένα δεν θα είναι ενημερωμένα.

CP: Παρέχει συνέπεια και ανοχή στον διαμοιρασμό, αλλά δεν εγγυάται διαθεσιμότητα όταν ένας κόμβος τεθεί εκτός λειτουργίας. Με τον συνδυασμό αυτό, δεν εξασφαλίζεται συνεχής πρόσβαση στα δεδομένα σε περίπτωση βλάβης

AP: Παρέχει διαθεσιμότητα και ανοχή στον διαμοιρασμό των δεδομένων μιας και υπάρχει συνεχής πρόσβαση στα δεδομένα, όμως τα δεδομένα μπορεί να μην είναι ενημερωμένα. Οι κόμβοι παραμένουν online ακόμα κι αν δεν μπορούν να επικοινωνήσουν μεταξύ τους και θα συγχρονίσουν τα δεδομένα τους όταν ολοκληρωθεί ο διαμοιρασμός. Παρ' όλα αυτά δεν είναι σίγουρο πως όλοι οι κόμβοι θα έχουν τα ίδια δεδομένα. [9], [10]

1.8 Χαρακτηριστικά ενός συστήματος διαχείρισης βάσεως δεδομένων

Στην πληροφορική το **ACID** (ατομικότητα, συνέπεια, απομόνωση, μονιμότητα) είναι ένα σύνολο ιδιοτήτων το οποίο εγγυάται ότι οι συναλλαγές στην βάση δεδομένων λειτουργούν αξιόπιστα.

Η ιδέα του ACID είναι να αξιολογεί την δομή βάσεων δεδομένων και εφαρμογών.

Όταν λέμε συναλλαγή εννοούμε μια σειρά ενεργειών που σχετίζονται με τα δεδομένα της βάσης δεδομένων. Έτσι για παράδειγμα, η μεταφορά χρημάτων από έναν τραπεζικό λογαριασμό σε έναν άλλο, παρόλο που για την βάση δεδομένων εμπλέκουν μια σειρά ενεργειών, για τον χρήστη θεωρείται μια συναλλαγή.

Ουσιαστικά μια συναλλαγή είναι η εκτέλεση μέρους ενός προγράμματος που προελαύνει και τροποποιεί – ενημερώνει κάποια δεδομένα.

Για να διασφαλιστεί η ακεραιότητα των δεδομένων και η αξιοπιστία της συναλλαγής, απαιτείται η βάση δεδομένων να διατηρεί τις παρακάτω ιδιότητες συναλλαγών, γνωστές με το ακρωνύμιο ACID.

Ατομικότητα (Atomicity): Πρέπει να γίνουν όλες οι λεπτομέρειες της συναλλαγής σωστά στη βάση δεδομένων, ή καμία. Έτσι ή είναι επιτυχής ολοκληρωτικά ή αποτυχημένη. Ακόμα και μία ενέργεια να αποτύχει, αποτυγχάνει όλη η συναλλαγή.

Συνέπεια(Consistency): Η εκτέλεση μιας συναλλαγής απομονωμένα (δηλαδή, χωρίς ταυτόχρονη εκτέλεση άλλων συναλλαγών) διατηρεί την συνέπεια της βάσης δεδομένων. Διασφαλίζει ότι η συναλλαγή είναι έγκυρη αλλά δεν εγγυάται ότι είναι και σωστή.

Απομόνωση (Isolation): Ακόμα κι αν εκτελούνται ταυτόχρονα πολλές συναλλαγές το σύστημα διασφαλίζει ότι δεν θα είναι ορατές οι συνέπειες σε μια συναλλαγή από μια άλλη ημιτελή συναλλαγή. Έτσι κάθε συναλλαγή δεν ξέρει για τις άλλες συναλλαγές που εκτελούνται ταυτόχρονα.

Μονιμότητα (Durability): Αφού μια συναλλαγή ολοκληρωθεί με επιτυχία, παραμένουν οι αλλαγές που έχει κάνει στη βάση δεδομένων, ακόμα και αν το σύστημα έχει πρόβλημα. [11]

1.9 Υπολογιστικό νέφος (cloud computing)

Το υπολογιστικό νέφος (cloud computing) είναι η διάθεση υπολογιστικών πόρων μέσω διαδικτύου (π.χ. servers, apps κλπ), από κεντρικά συστήματα που βρίσκονται απομακρυσμένα από τον τελικό χρήστη (εταιρεία ή μεμονωμένος χρήστης), τα οποία τον εξυπηρετούν αυτοματοποιώντας διαδικασίες, παρέχοντας ευκολίες και ευελιξία σύνδεσης.

Το cloud computing είναι ίσως ο πιο αποδοτικός τρόπος χρήσης όταν ο όγκος των δεδομένων είναι μεγάλος (Big Data), καθώς ο χρήστης έχει **πρόσβαση σε υποδομές και πόρους** που σε διαφορετική περίπτωση θα έπρεπε να αγοράσει σε υψηλή τιμή. Ο μεμονωμένος χρήστης ή η εταιρεία έχει την δυνατότητα απόκτησης λογισμικού και υλικού σε πολύ πιο συμφέρουσα τιμή ενοικιάζοντάς τες ανάλογα με τις ανάγκες του. Έτσι **εξοικονομεί πόρους**, ενώ ταυτόχρονα δεν ασχολείται και με την συντήρηση και απαρχαίωση του εξοπλισμού.

Ο χρήστης έχει την δυνατότητα σχεδόν **απεριόριστης αποθήκευσης των δεδομένων** του.

Η διαδικασία **backup** και **αποκατάστασης** δεδομένων καθίσταται ευκολότερη σε σχέση με τις παραδοσιακές μεθόδους σε μια φυσική συσκευή. Άλλωστε οι περισσότεροι πάροχοι υπηρεσιών cloud μπορούν να χειριστούν την ανάκτηση της πληροφορίας.

Η **ενσωμάτωση του λογισμικού** γίνεται αυτόματα, χωρίς την ανάγκη κάποιας προσαρμογής των εφαρμογών από τον χρήστη.

Δυνατότητα πρόσβασης της πληροφορίας των χρηστών από οπουδήποτε υπάρχει σύνδεση με το Internet.

Το σύστημα είναι πλήρως λειτουργικό άμεσα όταν ο χρήστης θελήσει να χρησιμοποιήσει το cloud.

Είναι **εύκολο** για τον πάροχο να προσαρμοστεί στις ανάγκες και τις απαιτήσεις των χρηστών επομένως υπάρχει η δυνατότητα ανά πάσα στιγμή για **άμεση κλιμάκωση**

των υποδομών, όταν αυξάνονται οι ανάγκες του πελάτη. Καθώς οι απαιτήσεις μεγαλώνουν, ο τρόπος αυτός αποδεικνύεται να είναι ο πιο αποδοτικός οικονομικά.

Καθιστά δυνατή την **αποδοχή** νέων κατηγοριών/απαιτήσεων, **νέων υπηρεσιών** που έχουν διαδραστικό χαρακτήρα. Άλλωστε το cloud computing επιβάλει τη χρησιμοποίηση νέων μεθόδων εκτέλεσης των εργασιών επεξεργασίας των δεδομένων ή απόκρισης των συστημάτων σε περίπτωση σφαλμάτων, κ.ο.κ., λόγω του ότι οι εργασίες εκτελούνται σε κατανεμημένο σύστημα υπολογιστών.

Παρά τα πολλά πλεονεκτήματα, το cloud computing έχει και μειονεκτήματα τα οποία πρέπει να ληφθούν υπόψιν.

Ο χρήστης θα πρέπει να διαθέτει μια αξιόπιστη σύνδεση στο διαδίκτυο αλλά και να διαλέγει έναν αξιόπιστο πάροχο υπηρεσιών έτσι ώστε να εξασφαλίσει την ασφάλεια των δεδομένων του.

Επίσης ο πάροχος θα πρέπει να είναι σε θέση να επιλύσει τα διάφορα πρόσκαιρα τεχνικά προβλήματα που θα παρουσιαστούν.

Τέλος ο χρήστης θα πρέπει να λάβει υπόψιν το γεγονός ότι τα δεδομένα του είναι ευάλωτα σε εξωτερικές επιθέσεις από χάκερ και να λάβει τα κατάλληλα μέτρα ώστε να προφυλαχθεί. [12]

1.9.1 Κατηγορίες Μοντέλου υπηρεσιών του υπολογιστικού νέφους

Υπάρχουν τέσσερις βασικές κατηγορίες μοντέλων "υπηρεσιών υπολογιστικού νέφους":

Software-as-a-Service (SaaS) – Λογισμικό ως υπηρεσία: Αντί να εγκατασταθεί λογισμικό στο μηχάνημα και στον υπολογιστή του πελάτη επιβαρύνοντάς τον με τακτικές επιδιορθώσεις, συχνές αλλαγές εκδόσεων κτλ., εφαρμογές όπως το Word, CRM (Διαχείριση Σχέσεων Πελατών), ERP (Enterprise Resource Programming), διατίθενται (φιλοξενούνται) μέσω του διαδικτύου για την εξυπηρέτηση του τελικού χρήστη. Ουσιαστικά πρόκειται για λογισμικό που χρησιμοποιείται μέσω του Internet και ‘τρέχει’ σε κάποιο γνωστό φυλλομετρητή (web browser). Η εφαρμογή λειτουργεί σε υποδομές που ανήκουν στον πάροχο του SaaS και πωλούνται συνήθως σε μηνιαία ή ετήσια βάση.

Platform-as-a-Service (PaaS) - Πλατφόρμα ως Υπηρεσία: Είναι μια πλατφόρμα που παρέχεται στον πελάτη σαν υπηρεσία, όπου ο πελάτης έχει πρόσβαση σε συγκεκριμένες εφαρμογές μέσω ενός φυλλομετρητή (web browser). Η πλατφόρμα παρέχει το απαραίτητο λογισμικό, ώστε οι εφαρμογές να λειτουργούν στο υπολογιστικό νέφος. Έτσι ο πελάτης δεν χρειάζεται να αγοράσει τις άδειες λογισμικού για πλατφόρμες, για τα λειτουργικά συστήματα, τις βάσεις δεδομένων και το ενδιάμεσο λογισμικό. Επίσης, τα θέματα ασφάλειας, διαχείρισης, κλιμάκωσης των απαιτήσεων, κλπ., απασχολούν μόνον τον πάροχο της υπηρεσίας και όχι τον πελάτη, ο οποίος πληρώνει μια συνδρομή ανάλογα με τις απαιτήσεις του.

Infrastructure-as-a-Service (IaaS) - Υποδομή ως Υπηρεσία: Πρόκειται για τις απλές-βασικές υλικές συσκευές (raw υπολογιστές) όπως είναι οι εικονικοί υπολογιστές, οι διακομιστές, οι συσκευές αποθήκευσης, η μεταφορά μέσω δικτύου, οι οποίες βρίσκονται (με φυσικό τρόπο) σε ένα κεντρικό σημείο (κέντρο δεδομένων). Υπάρχει η δυνατότητα να προσπελαστούν και να χρησιμοποιηθούν μέσω διαδικτύου, χρησιμοποιώντας τα συστήματα ελέγχου ταυτότητας σύνδεσης και τους κωδικούς πρόσβασης από οποιοδήποτε dumb τερματικό ή συσκευή. Ο πελάτης πληρώνει ανάλογα με τη χρήση και τους πόρους που χρησιμοποιεί ή νοικιάζει και η συντήρηση και ορθή λειτουργία της υλικής υποδομής είναι ευθύνη του παρόχου. Επίσης η κλιμάκωση των υποδομών γίνεται πολύ εύκολα και οποιαδήποτε στιγμή χρειαστεί ο πελάτης.

Desktop-as-a-Service (DaaS) – Επιφάνεια εργασίας ως Υπηρεσία: Η υπηρεσία αυτή προσφέρει μια υποδομή εικονικής επιφάνειας εργασίας (Virtual Desktop Infrastructure - VDI) που φιλοξενείται από έναν πάροχο λύσεων λογισμικού υπολογιστικού νέφους και βασίζεται συνήθως σε ένα μοντέλο μηνιαίας συνδρομής. Το DaaS χρησιμοποιεί μια αρχιτεκτονική πολλαπλών μισθώσεων, πράγμα που σημαίνει ότι μια μοναδική εμφάνιση μιας εφαρμογής εξυπηρετείται σε πολλούς χρήστες, που αναφέρονται ως "ενοικιαστές". Η πληρωμή γίνεται ανά μήνα και ανάλογα με τη χρήση. Ο πάροχος λύσεων λογισμικού υπολογιστικού νέφους είναι υπεύθυνος για τη διαχείριση του υπολογιστικού νέφους και της υποκείμενης υποδομής καθώς και για το επίπεδο εξυπηρέτησης και μπορεί να διαφέρει ανάλογα με τις ανάγκες των χρηστών. Το τελικό αποτέλεσμα αυτής της υποδομής είναι ότι οι χρήστες μπορούν να έχουν πρόσβαση στα δεδομένα και τις εφαρμογές τους από σχεδόν οποιαδήποτε συσκευή, οπουδήποτε. [12]

1.10 Κατηγορίες NoSQL βάσεων δεδομένων

Υπάρχουν οι εξής κατηγορίες NoSQL βάσεων δεδομένων:

1) Key-values Stores,

2) Column Family Stores ή Wide Column Stores

3) Document Databases και

4) Graph Databases.

Η πρώτη κατηγορία, οι **key-value stores**, χρησιμοποιούν έναν hash table (ή dictionary) όπου υπάρχει ένα μοναδικό κλειδί και ένας δείκτης στοχεύοντας σε ένα συγκεκριμένο στοιχείο.

Ο πίνακας αυτός περιέχει δύο στήλες όπου η πρώτη στήλη αφορά στα κλειδιά και η δεύτερη στήλη στις τιμές. Οι key-valued stores βάσεις δεδομένων χειρίζονται τα δεδομένα ως μια εγγραφή που μπορεί να έχει διαφορετικό αριθμό πεδίων για κάθε εγγραφή. Αυτό δίνει ευελιξία και είναι πιο κοντά στις νέες απόψεις του αντικειμενοστραφούς προγραμματισμού ο οποίος τείνει να επικρατήσει καθώς μπορεί να ανταπεξέλθει σε προγράμματα μεγαλύτερου όγκου και πολυπλοκότητας.

Επίσης, χρησιμοποιούν πολύ λιγότερη μνήμη για να αποθηκεύσουν τα ίδια δεδομένα σε σχέση με μια RDB, γεγονός το οποίο κάνει την βάση πιο αποδοτική. [13]

Στις column store βάσεις δεδομένων, κάθε στήλη αποθηκεύεται ξεχωριστά στον δίσκο. Οι **wide column** store βάσεις δεδομένων υποστηρίζουν την δομή των column stores αλλά κάθε στήλη περιέχει πολλές στήλες που χρησιμοποιούνται όλες μαζί με τρόπο παρόμοιο με τους πίνακες στις σχεσιακές βάσεις δεδομένων.

Υπάρχουν κι εδώ κλειδιά τα οποία όμως στοχεύουν σε περισσότερα από ένα στοιχεία.

Οι γραμμές αναγνωρίζονται από ένα μοναδικό row key ενώ οι στήλες είναι οργανωμένες σε column families («οικογένειες στηλών»).

Με τον τρόπο αυτό η πρόσβαση στα δεδομένα γίνεται με ακρίβεια και σε μικρότερο χρονικό διάστημα. [14]

Οι **document oriented stores** είναι όμοιες με τις key-value stores. Τα δεδομένα σε αυτή την περίπτωση είναι οργανωμένα σε key-valued «συλλογές» δεδομένων. Αποθηκεύουν και οργανώνουν τα δεδομένα σαν μια συλλογή από έγγραφα σε ημιδομημένους πίνακες. Οι αποθηκευμένες πληροφορίες μπορεί να είναι διαφορετικές από αντικείμενο σε αντικείμενο. Έτσι δύο εγγραφές μπορούν να έχουν διαφορετικό αριθμό πεδίων.

Αυτές οι βάσεις δεδομένων είναι δημοφιλείς καθώς έχουν ένα δυναμικό σχήμα και ανταποκρίνονται καλύτερα στις σημερινές διαδικτυακές εφαρμογές, όπου αποθηκεύονται διαφορετικού τύπου δεδομένα των οποίων τα χαρακτηριστικά συνεχώς μεταβάλλονται. [15]

Οι **Graph databases**, είναι βασισμένες σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων (ακμές) και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graph model) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους) και να προσαρμόζεται στις απαιτήσεις του χρήστη.

Είναι κατάλληλες για την υλοποίηση βάσεων δεδομένων που είναι δύσκολο να πραγματοποιηθούν με σχεσιακά συστήματα. Η σχέση μεταξύ των δεδομένων έχει και τη μεγαλύτερη σημασία και έτσι είναι εύκολη η απάντηση των ερωτημάτων.

Οι σχέσεις μεταξύ των δεδομένων μπορούν να απεικονιστούν οπτικά, κάτι που είναι πολύ χρήσιμο για δεδομένα που συνδέονται με πολύπλοκο τρόπο. [16]

2. NoSQL βάσεις δεδομένων στο υπολογιστικό νέφος

2.1 Marklogic

Εικόνα 1. Λογότυπο της Marklogic

Η Marklogic είναι μια επιχειρηματική και συναλλακτική πλατφόρμα που συνδυάζει την ευελιξία, την ταχύτητα και την δυνατότητα κλιμάκωσης μιας NoSQL βάσης δεδομένων, με τα αυστηρά εμπορικά χαρακτηριστικά.

Η Marklogic είναι μια βάση δεδομένων που παρέχει, ενσωματώνει, διαμοιράζεται και αποθηκεύει μεγάλο όγκο σημαντικών δεδομένων δίνοντας την δυνατότητα αυτά να παρουσιαστούν ως έγγραφα ή γραφικά ή και σαν σχεσιακά δεδομένα τόσο στον υπολογιστή, όσο και στο υπολογιστικό σύννεφο.

Παρέχει υψηλή διαθεσιμότητα και ασφάλεια των δεδομένων, συμμόρφωση με τις ιδιότητες ACID και δυνατότητα κρυπτογράφησης. Έτσι είναι κατάλληλη για επιχειρήσεις που πρέπει να χειρίζονται μεγάλο όγκο ευαίσθητων δεδομένων. Είναι και η μόνη βάση δεδομένων που φέρει το Common Criteria Certification, διασφαλίζοντας έτσι και πιστοποιώντας την ασφάλεια των δεδομένων που χειρίζεται.

Έχει χαρακτηριστικά όπως τα semantics και bitemporals.

Τα semantics [17] έχουν τη λογική ότι κάθε δημοσιευμένη πληροφορία θα περιέχει μεταδεδομένα (metadata) τα οποία θα είναι κοινά για όλους, θα μπορούν να κατανοούνται από τις μηχανές οι οποίες θα βοηθήσουν στην καλύτερη συλλογή και επεξεργασία τους.

Τα bitemporals επιτρέπουν στις επιχειρήσεις να μπορούν να δουν τα δεδομένα όπως δημιουργήθηκαν στην πορεία του χρόνου, χαρακτηριστικό πολύ κρίσιμο σε εταιρείες με πολύ αυστηρό πλαίσιο, αλλά και με ευρεία εφαρμογή σε εταιρίες ή οργανισμούς που χρειάζεται να κρατούν ιστορικό για τα δεδομένα τους.

Έτσι μπορούν να βρίσκουν τις αλλαγές που γίνονται στην πορεία του χρόνου παρέχοντας την δυνατότητα αυτόματης εύρεσης νέων δεδομένων και των μεταξύ τους σχέσεων ανάμεσα σε δισεκατομμύρια εγγράφων.

Έχει ενσωματωμένες τεχνολογίες JSON, XML, RDF, Javascript, ευρετήρια τύπου geospatial και ευρετήρια τύπου “ask anything”.

Το JSON [18] ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων, εύκολο για τους ανθρώπους να το διαβάσουν και γράψουν, εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών

άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Ο Javascript κώδικας τρέχει μη-παράλληλα με τα δεδομένα και προσφέρει επιπλέον δύναμη και αποτελεσματικότητα.

Στη μεσαία βαθμίδα η Marklogic συνεχίζει με Javascript χρησιμοποιώντας έναν Node.js server [19] ο οποίος μπορεί να διαχειρίζεται και να τροποποιεί τα δεδομένα στη βάση δεδομένων ή μπορεί να κάνει χρήση του Marklogic Java Client API [20] το οποίο δίνει πρόσβαση στην βάση δεδομένων και επιτρέπει στους χρήστες να προχωρούν ταχύτερα, χρησιμοποιώντας την υπάρχουσα γνώση, τις δεξιότητες, τις βιβλιοθήκες, τα εργαλεία και τα περιβάλλοντα ανάπτυξης.

Οι εγγραφές στην βάση δεδομένων έχουν καταγεγραμμένο τον χρόνο που πραγματοποιήθηκαν (timestamping) και έτσι μπορούν να αναγνωριστούν οι αλλαγές, κάτι που είναι πολύ σημαντικό για τις εταιρείες καθώς έτσι μειώνεται ο χρόνος επεξεργασίας και εύρεσης των σχέσεων μέσα σε δισεκατομμύρια ή και τρισεκατομμύρια έγγραφα. [7], [21]

2.2 MONGO DB

Εικόνα 2. Λογότυπο της MongoDB

Η MongoDB είναι η πιο δημοφιλής NoSQL βάση δεδομένων. Είναι μια ανοικτού τύπου cross-platform (δηλαδή μπορεί να 'τρέχει' σε διαφορετικά λειτουργικά συστήματα), document-oriented βάση δεδομένων. Η πλατφόρμα συντηρείται από την MongoDB Inc.

Η MongoDB είναι μια βάση δεδομένων επόμενης γενιάς που επιτρέπει στους χρήστες να δημιουργούν εφαρμογές που προηγουμένως ήταν αδύνατον. Είναι σχεδιασμένη για CIOs (Chief Information Officers), αρχιτέκτονες, DBAs (Database Admistrators) και Ops, προγραμματιστές και αναλυτές.

Ενσωματώνει όλες τις καλές πρακτικές λειτουργίας της εταιρείας MongoDB Inc. που έχουν προκύψει από την εμπειρία της μέσα από την βελτίωση χιλιάδων εφαρμογών της και σε εταιρείες κάθε μεγέθους.

Η έκδοση της για Cloud μπορεί να έχει εργαλεία για την διαχείριση, την διαμόρφωση, την επιδιόρθωση του λογισμικού, την παρακολούθηση, των backups της βάσης δεδομένων και λειτουργεί σε ένα διαμοιρασμένο δίκτυο υπολογιστών.

Η MongoDB αποθηκεύει δεδομένα χρησιμοποιώντας ένα ευέλικτο μοντέλο εγγράφων (document data store) που είναι JSON-like. Τα δεδομένα μπορούν να περιέχουν ένα ή περισσότερα πεδία, πίνακες, δεδομένα σε δυαδική μορφή (binary data) και sub-documents. Τα πεδία μπορούν να διαφέρουν από έγγραφο σε έγγραφο και αυτό δίνει στους προγραμματιστές την ευελιξία να εξελίξουν το μοντέλο των δεδομένων τους ταχύτατα, καθώς αλλάζουν οι απαιτήσεις των εφαρμογών. Οι προγραμματιστές δουλεύουν με την MongoDB επειδή είναι απλή, χρησιμοποιεί μοντέλο JSON-like το οποίο όμως έχει δυναμικό σχεδιασμό επιτρέποντάς τους να κάνουν γρήγορα τις αλλαγές, προσφέροντας υψηλή διαθεσιμότητα, εύκολη επεκτασιμότητα και υψηλή απόδοση στις εφαρμογές.

Οι διάφορες λειτουργίες γίνονται γρηγορότερα, χωρίς ρίσκο και οικονομικότερα. Επίσης, ταιριάζει με οποιαδήποτε υπάρχουσα υποδομή δίνοντας την δυνατότητα αναβάθμισης με ένα κλικ. Έχει ενσωματωμένη λειτουργία κλιμάκωσης και μπορεί να χειρίζεται από μόνη της τα όποια προβλήματα παρουσιαστούν.

Διαθέτει drivers για τις γλώσσες που χρησιμοποιούν οι χρήστες για την κωδικοποίηση και μια εκφραστική query language που τους επιτρέπει να θέτουν, να διαλέγουν και να βρίσκουν δεδομένα χωρίς να χρειάζεται επιπλέον κώδικας.

Είναι από τις πιο αγαπημένες πλατφόρμες των προγραμματιστών γιατί είναι εύκολη στη χρήση, εύκολη στην κλιμάκωση ικανοποιώντας απαιτητικές εφαρμογές και λειτουργεί σε ένα κατανοητό περιβάλλον εργαλείων. [7], [21]

Εφαρμόζει το σύστημα IoT (Internet of Things). Πρόκειται για μια τεχνολογική εξέλιξη όπου πολλές συσκευές μαζί αποτελούν ένα δίκτυο. Κάθε συσκευή ενσωματώνει ηλεκτρονικά μέσα, λογισμικό, αισθητήρες ώστε να επιτρέπει την σύνδεση και την ανταλλαγή δεδομένων μεταξύ τους, είτε σε τοπικό δίκτυο, είτε στο Διαδίκτυο. Το IoT θα λειτουργήσει ως μοχλός ανάπτυξης καινοτομίας και εξέλιξης των επιχειρηματικών μοντέλων, αλλά και οικονομικής ανάπτυξης των επιχειρήσεων. Παράλληλα θα ενταχθεί στη πραγματοποίηση επωφελών έργων για την αντιμετώπιση σοβαρών κοινωνικών και περιβαλλοντικών προβλημάτων.[22]

Προσφέρει εξατομίκευση, real-time analytics, διαχείριση μεγάλου όγκου δεδομένων, ασφάλεια των δεδομένων από επιχειρούμενη απάτη, εφαρμογή σε κινητό, και στα μέσα κοινωνικής δικτύωσης.

Έχει ενσωματωμένη την διαδικασία της αντιγραφής (replication) της βάσης δεδομένων και την τεχνική sharding (θραυματοποίηση) όπου η βάση δεδομένων χωρίζεται σε μικρότερα και ευκολότερα διαχειρίσιμα κομμάτια, τα οποία διανέμονται στους άλλους server του δικτύου πετυχαίνοντας την καλύτερη διαχείριση του φόρτου δεδομένων και την διατήρηση της υψηλής ροής των δεδομένων, αποφεύγοντας απώλεια δεδομένων σε περίπτωση αποτυχίας υλικού. [23]

2.3 APACHE COUCH DB

Εικόνα 3. Λογότυπο της CouchDB

Η Apache CouchDB ανοικτού τύπου NoSQL βάση δεδομένων. Είναι μια document-oriented βάση δεδομένων της εταιρείας Apache CouchDB Inc.

Είναι μια schema-free βάση δεδομένων όπου τα δεδομένα μπορούν να αποθηκεύονται χωρίς να έχει οριστεί εξ' αρχής η δομή τους.

Χρησιμοποιείται από μεγάλες και μικρές εταιρείες και οργανισμούς με μια ποικιλία εφαρμογών όπου οι κλασικές σχεσιακές βάσεις δεδομένων θα ήταν δύσκολο να υλοποιήσουν.

Είναι μία καταπληκτική single-node βάση δεδομένων που 'τρέχει' σε οποιονδήποτε server αλλά και μέσω της αντιγραφής (replication) της βάσης δεδομένων μπορεί να λειτουργεί σε έναν ή και περισσότερους server ή εικονικές μηχανές. Οποιοσδήποτε αριθμός κόμβων μπορεί να έχει ανεξάρτητα replica copies της ίδιας βάσης δεδομένων.

Όταν λειτουργεί σε ένα διαμοιρασμένο σύστημα υπολογιστών(cluster) βελτιώνει τις λειτουργίες του single-node με υψηλότερη ικανότητα και διαθεσιμότητα χωρίς να χρειάζεται κάποια αλλαγή στην επιφάνεια διεπαφής.

Η CouchDB χρησιμοποιεί το ευρέως διαδεδομένο πρωτόκολλο HTTP και το μοντέλο δεδομένων JSON και είναι συμβατή με κάθε λογισμικό που τα υποστηρίζει. Λειτουργεί εξαιρετικά με τα πρόσθετα εξωτερικά εργαλεία όπως HTTP proxy servers - ο οποίος λειτουργεί ως ενδιάμεσος μεταξύ χρήστη-διακομιστή προσκομίζοντας την σελίδα από τον διακομιστή και παραδίδοντάς την στον υπολογιστή του χρήστη που ζήτησε πρόσβαση, load balancers - συσκευές που αναλαμβάνουν να μοιράσουν την κυκλοφορία δεδομένων στο δίκτυο, αυξάνοντας έτσι την απόδοση και την σταθερότητα της εφαρμογής. [24]

Σε ένα cluster με πολλούς κόμβους, τα δεδομένα αντιγράφονται και αποθηκεύονται σε πολλές τοποθεσίες, ώστε να είναι διαθέσιμα όταν χρειαστούν. Το μοναδικό πρωτόκολλο που διαθέτει για αντιγραφή, αποτελεί την βάση για μια νέα γενιά "Offline-first" εφαρμογών για κινητά και για άλλα απαιτητικά περιβάλλοντα στο

διαδίκτυο. Έτσι τα δεδομένα αρχικά γράφονται τοπικά στην συσκευή του χρήστη και στη συνέχεια φορτώνονται περιοδικά και αντιγράφονται στο υπολογιστικό νέφος.

Υποστηρίζει τα ACID Semantics και έτσι μπορεί να χειρίζεται μεγάλο αριθμό χρηστών χωρίς να δημιουργείται σύγκρουση.

Έχει ενσωματωμένη υποστήριξη BLOBs (Binary Large Objects) δηλαδή συλλογές δεδομένων σε δυαδική μορφή αποθηκευμένα σαν μια οντότητα μέσα στην βάση δεδομένων. Ουσιαστικά πρόκειται για εικόνες, ηχητικά αρχεία και άλλες πολυμεσικές εφαρμογές.

Χρησιμοποιεί το προγραμματιστικό μοντέλο MapReduce το οποίο χρησιμοποιείται για την ανάπτυξη εφαρμογών που επεξεργάζονται παράλληλα τεράστιες ποσότητες δεδομένων. Είναι ένα πολύ σημαντικό μοντέλο για την δημιουργία και επεξεργασία μεγάλων βάσεων δεδομένων, αλλά και την ανάλυση μεγάλου όγκου δεδομένων (Big Data). Το πρόβλημα 'σπάει' σε δύο φάσεις με την βοήθεια δύο συναρτήσεων, της map και της reduce και τελικά επιστρέφεται σαν μια τιμή. Έτσι η απάντηση ερωτημάτων γίνεται πιο γρήγορα με την βοήθεια των ευρετηρίων που δημιουργούνται.

Επιπλέον εφαρμόζει την master-master τεχνική στην οποία ο master είναι υπεύθυνος για τον χρονοπρογραμματισμό της εκτέλεσης των εργασιών και σε περίπτωση καθυστέρησης σε έναν κόμβο, αναθέτει την εργασία του σε κάποιον άλλο.[25]

Εφαρμόζει σταδιακή και ανακατευθυνόμενη αντιγραφή (replication) όπου τα αρχεία διασπώνται σε block και το κάθε block αντιγράφεται σε πολλαπλούς κόμβους δεδομένων.

Η CouchDB έχει δημιουργηθεί για server οποιουδήποτε μεγέθους, από τον πιο μικρό όπως ο Raspberry pi μεγέθους όσο μιας πιστωτικής κάρτας, μέχρι μεγάλες εγκαταστάσεις στο υπολογιστικό νέφος.

Επειδή έχει μεγάλη σημασία η αξιοπιστία των δεδομένων, η CouchDB για έναν κόμβο χρησιμοποιεί μια δομή append only (όπου στα δεδομένα μπορούμε μόνο να προσθέσουμε και όχι να μετακινήσουμε ή να αλλάξουμε) και έτσι ακόμα κι αν σταματήσει να λειτουργεί η εφαρμογή (κρασάρισμα), τα δεδομένα είναι ασφαλή. Σε κατανεμημένο σύστημα υπολογιστών με πολλούς κόμβους όλα τα δεδομένα σώζονται σε εφεδρικό backup και έτσι είναι διαθέσιμα όταν ζητηθούν.

2.4 APACHE CASSANDRA

Εικόνα 4.Λογότυπο της Cassanrda

Είναι μια δωρεάν ανοικτού τύπου NoSQL βάση δεδομένων που έχει σχεδιαστεί για να διαχειρίζεται πολύ μεγάλο όγκο δεδομένων, της τάξης των Petabyte (1 Petabyte = 10^{15} bytes) είτε σε απλούς υπολογιστές, είτε σε μεγάλα κατανεμημένα συστήματα υπολογιστών μέσα στο υπολογιστικό νέφος. Είναι ιδανική για εφαρμογές που δεν έχουν περιθώριο να χάσουν μέρος ή το σύνολο των δεδομένων τους.

Ένα από τα κυριότερα χαρακτηριστικά της είναι ότι είναι εξαιρετική σε περιπτώσεις που χρειάζεται οριζόντια κλιμάκωση σε μεγάλα κατανεμημένα δίκτυα (clusters). Η master-less (αποκεντρωμένη) αρχιτεκτονική κάνει την δημιουργία και την επέκταση των δικτύων σχετικά απλή διαδικασία καθώς κάθε κόμβος είναι πανομοιότυπος και δεν υπάρχουν σημεία συμφόρησης (network bottlenecks) τα οποία δημιουργούν καθυστερήσεις. Επίσης δίνεται η δυνατότητα στις επιχειρήσεις να προσθέσουν online και ανά πάσα στιγμή επιπλέον χωρητικότητα για να φιλοξενήσουν επιπλέον πελάτες και επιπλέον δεδομένα.

Η γλώσσα ερωτημάτων της Cassandra (CQL) είναι παρόμοια με την query language της SQL και έτσι οι προγραμματιστές μπορούν εύκολα να μετατρέψουν σε NoSQL μια σχεσιακή βάση δεδομένων.

Χρησιμοποιεί solr, μια ανοικτού τύπου πλατφόρμα αναζήτησης, γραμμένη σε Java που δημιουργήθηκε από την Apache Lucene.

Η εντυπωσιακή ιεραρχία στους μηχανισμούς της κρυφής μνήμης (cache) και η προσεκτική τακτοποίηση των δεδομένων (I/O) στο δίσκο εγγυάται την ασφάλεια και την ταχύτητα.

Η αρχιτεκτονική της αποθήκευσης των δεδομένων είναι παρόμοια με την δομή δεδομένων log-structured merge tree η οποία αποτελείται από δύο ή περισσότερα συστατικά παρόμοια με δέντρο. Οι εγγραφές στέλνονται πρώτα στη μνήμη και στη συνέχεια σε έναν memtable στην μνήμη cache. Όταν ο memtable γεμίσει, τότε τα δεδομένα πηγαίνουν στον δίσκο σε έναν Sorted String Table (SSTable) που είναι ένας ικανοποιητικός τρόπος αποθήκευσης μεγάλου αριθμού από ζεύγη κλειδιού-τιμής, ενώ βελτιστοποιεί την διακίνηση των δεδομένων και την σειριακή τους τακτοποίηση στον δίσκο. Οι εγγραφές στον δίσκο είναι append-only (δηλαδή μπορείς μόνο να

προσθέσεις και όχι να σβήσεις ή να τροποποιήσεις δεδομένα) και επειδή γίνονται με σειριακό τρόπο και όχι τυχαία, είναι πολύ αποτελεσματικές.[26]

Ένα cluster είναι οργανωμένο με την διάταξη του δακτυλίου και εφαρμόζει την τεχνική του διαμερισμού(partition) για να κατανείμει τα δεδομένα στους κόμβους. Τα δεδομένα διαμερίζονται μέσω μιας συνάρτησης κατακερματισμού (hash) και δημιουργούνται τελικά δεδομένα σταθερού μεγέθους 128bit. Ο διαμερισμός αυτός συμπληρώνεται και από ένα άλλο στοιχείο που λέγεται 'snitch' (καρφί) και το οποίο 'κρατάει' σε ένα data center δύο στοιχεία: την IP address του κόμβου και την φυσική του θέση.

Οι εγγραφές στην Cassandra γράφονται σε πολλούς κόμβους ώστε τα δεδομένα να είναι ασφαλή σε περίπτωση κατάρρευσης κάποιου κόμβου. Ο κόμβος στον οποίο αναπαράγονται (αντιγράφονται) τα αρχικά δεδομένα λέγεται replica node. Το 'snitch' χρησιμοποιείται ώστε να διασφαλίζει ότι ο replica κόμβος δεν βρίσκεται στο ίδιο rack για το ίδιο κομμάτι πληροφορίας.

Η Cassandra παρέχει την καλύτερη υποστήριξη της αντιγραφής των δεδομένων στα κέντρα δεδομένων, παρουσιάζοντας τις μικρότερες καθυστερήσεις και δίνει την βεβαιότητα ότι μπορεί να επιβιώσει από οποιαδήποτε διακοπή σε κάποιον κόμβο.

Εφαρμόζει την τεχνική Hinted Handoffs όπου αν κάποιος replica κόμβος δεν είναι διαθέσιμος κάποια στιγμή, η Cassandra θα δημιουργήσει ένα hint (στοιχείο) το οποίο θα υποδεικνύει ότι η εγγραφή θα πρέπει να ξανασταλεί στον ανενεργό κόμβο. Όταν ο κόμβος επανέλθει τα δεδομένα θα σταλούν μέσω του hint στον κόμβο που είχε πρόβλημα αρχικά. Έτσι ενισχύεται το χαρακτηριστικό της συνέπειας των δεδομένων. [27]

2.5 APACHE HADOOP

Εικόνα 5.Λογότυπο της Hadoop

Η Apache Hadoop είναι ένα ελεύθερο ανοικτού τύπου λογισμικό αξιόπιστο για κατανεμημένο σύστημα υπολογιστών (cluster) και με δυνατότητα κλιμάκωσης. Η βιβλιοθήκη της Apache Hadoop επιτρέπει να διαμοιράζει μεγάλο όγκο δεδομένων στους συνδεδεμένους υπολογιστές χρησιμοποιώντας απλά προγραμματιστικά μοντέλα.

Ο πυρήνας της Apache Hadoop core αποτελείται από δύο μέρη. Ένα αποθηκευτικό μέρος γνωστό ως Hadoop Distributed File System (HDFS) και ένα επεξεργαστικό μέρος το MapReduce όπου τα αρχεία χωρίζονται σε μεγάλα blocks και διαμοιράζονται στους κόμβους του δικτύου.

Είναι σχεδιασμένο για κλιμάκωση τόσο από έναν server μέχρι σε χιλιάδες μηχανήματα, καθένα από το οποίο μπορεί να κάνει γρήγορη επεξεργασία δεδομένων και αποθήκευση. Όσους περισσότερους κόμβους έχει τόσο μεγαλύτερη επεξεργαστική δύναμη έχει. Απλά χρειάζεται διαχείριση.

Έχει την δυνατότητα να επεξεργάζεται πολύ γρήγορα μεγάλο όγκο δεδομένων κάθε είδους, χαρακτηριστικό πολύ σημαντικό λαμβάνοντας υπόψιν τον αυξανόμενο όγκο δεδομένων ειδικά από τα μέσα κοινωνικής δικτύωσης αλλά και των συσκευών των συνδεδεμένων στο Internet (IoT).

Τα δεδομένα και η επεξεργασία των εφαρμογών προστατεύονται από δυσλειτουργίες των υπολογιστών. Αν ένας κόμβος βρεθεί εκτός λειτουργίας, οι δουλειές του ανακατευθύνονται σε άλλους κόμβους και έτσι το δίκτυο συνεχίζει να λειτουργεί. Τα πολλαπλά αντίγραφα όλων των δεδομένων αποθηκεύονται αυτόματα. Επίσης όπως σε κάθε NoSQL βάση δεδομένων, μπορεί να αποθηκεύσει μεγάλο όγκο δεδομένων(κείμενα, εικόνες, βίντεο) με μη καθορισμένη δομή εξ' αρχής.

Επειδή είναι ελεύθερο και ανοικτού τύπου ο χρήστης μπορεί να επεξεργαστεί μεγάλη ποσότητα δεδομένων, χρησιμοποιώντας έναν απλό υπολογιστή. [28]

Το Apache Hadoop αποτελείται από τα παρακάτω απαραίτητα μέρη - μοντέλα (modules):

Hadoop Common ή αλλιώς και Hadoop core. Είναι μια συλλογή από τις κοινές υπηρεσίες και βιβλιοθήκες της Hadoop. Θεωρεί ότι τα προβλήματα με το υλικό του δικτύου είναι κάτι πολύ συνηθισμένο και ότι θα πρέπει να διευθετούνται αυτόματα από το λογισμικό της Hadoop. Επίσης περιέχει τα Java αρχεία (JAR) που είναι απαραίτητα για την εκκίνηση του Hadoop. [29]

Hadoop Distributed File System (HDFS). Είναι ένας διανεμημένος φάκελος συστήματος που μπορεί να χειρίζεται μεγάλο όγκο δεδομένων σε έναν απλό υπολογιστή. Χρησιμοποιείται για την κλιμάκωση από έναν υπολογιστή μέχρι και εκατοντάδες ή και χιλιάδες κόμβους.[30]

Hadoop YARN. Είναι το ενδιάμεσο στοιχείο μεταξύ του HDFS και την εκτέλεση μιας εφαρμογής. Συνδυάζει ένα κεντρικό διαχειριστή πόρων, συντονιστές εφαρμογών, και 'πράκτορες' (agents) σε επίπεδο κόμβων που ελέγχουν την εκτέλεση των διαδικασιών σε κάθε κόμβο ξεχωριστά. Μπορεί να διανείμει τους πόρους στις εφαρμογές κατά πως χρειάζεται, αυξάνοντας έτσι την χρησιμότητα των πόρων και την εκτέλεση των εφαρμογών προσφέροντας έτσι έναν δυναμικό τρόπο διανομής των πόρων.

Μέσω του πρόσθετου (pluggable tool) Fair Scheduler υποστηρίζει πολυπρογραμματιστικές μεθόδους επεξεργασίας δεδομένων όχι με το μοντέλο FIFO (first in-first out), αλλά με έναν 'δίκαιο' διαμοιρασμό των πόρων του δικτύου ώστε όλες οι εργασίες να πραγματοποιούνται την ίδια στιγμή.

Έχει ένα χαρακτηριστικό που λέγεται Reservation System και με το οποίο μπορεί να 'κρατά' προκαταβολικά πόρους του δικτύου για σημαντικές επεξεργαστικές εργασίες ώστε να εξασφαλίσει ότι θα λειτουργούν χωρίς πρόβλημα. Οι πόροι που παρακρατούνται μπορεί και να δημιουργήσουν υπερφόρτωση στο δίκτυο και γι' αυτό ο κάθε χρήστης μπορεί να θέσει ένα όριο πέραν του οποίου να απορρίπτουν αιτήματα για κρατήσεις πόρων.

Ένα άλλο αξιοσημείωτο χαρακτηριστικό είναι το YARN Federation το οποίο δίνει την δυνατότητα σε ένα μόνο κόμβο να αυξήσει τον αριθμό των κόμβων στους οποίους μπορεί να επεκταθεί από 10.000 σε δεκάδες χιλιάδες συνδέοντας μεταξύ τους υποδίκτυα που το καθένα έχει τον δικό του διαχειριστή πόρων. Έτσι μετατρέπεται σε ένα ενιαίο δίκτυο που μπορεί να εκτελεί εφαρμογές σε κάθε διαθέσιμο κόμβο. [31]

Hadoop MapReduce. Είναι ένα λογισμικό που μας επιτρέπει να γράφουμε εφαρμογές που διαχειρίζονται αξιόπιστα μεγάλο όγκο δεδομένων παράλληλα, σε μεγάλα κατανεμημένα δίκτυα απλών υπολογιστών.

Η 'δουλειά' του MapReduce είναι να χωρίζει τα εισερχόμενα(inputs) σετ δεδομένων σε ανεξάρτητα τμήματα τα οποία μέσα από την διαδικασία Map τα επεξεργάζονται σε εντελώς παράλληλο τρόπο. Τα εξερχόμενα (outputs) ταξινομούνται και γίνονται εισερχόμενα στις διαδικασίες Reduce.

Το framework της Hadoop έχει την αρχιτεκτονική master-slave και φροντίζει για τον προγραμματισμό των εργασιών. Επίσης επανεκτελεί τις εργασίες που δεν έγιναν. [32]

Η Hadoop διαθέτει πρόσθετα πακέτα λογισμικού που μπορούν να εγκατασταθούν και να προσδώσει κάποια επιπλέον χαρακτηριστικά στις λειτουργίες της Hadoop. [7]

2.6 NEO4J

Εικόνα 6.Λογότυπο της Neo4j

Η Neo4j είναι μια ανοικτού τύπου NoSQL βάση δεδομένων. Είναι η πιο δημοφιλής βάση δεδομένων γράφων (graph database) και Cypher Query Language(CQL) γραμμένη σε Java. Είναι η πρώτη κυρίαρχη δύναμη στην αγορά των βάσεων δεδομένων των γράφων (graph).

Ξεκίνησε να αναπτύσσεται το 2003, αλλά για το κοινό ήταν διαθέσιμη από το 2007 είτε δωρεάν στο GitHub, είτε σαν χρήστης μέσω downloaded εφαρμογής. Υπάρχει και σε απλή έκδοση αλλά και εμπορική. Η εμπορική έκδοση περιλαμβάνει ότι η απλή με επιπλέον δυνατότητες όπως για backup, για clustering και για αντιμετώπιση προβλημάτων.

Ο στόχος της εταιρείας είναι να κάνει την τεχνολογία των γραφικών την κυρίαρχη τάση, συνδέοντας την κοινωνία, τους πελάτες ακόμα και τους ανταγωνιστές ώστε να υιοθετήσουν βέλτιστες πρακτικές.

Χρησιμοποιεί την Cypher που είναι η εμφατική/δηλωτική (declarative) γλώσσα ερωτημάτων της Neo4j, παρόμοια με την SQL αλλά βελτιστοποιημένη για γραφήματα. Η Cypher είναι εύκολη στην εκμάθηση και μπορεί να χρησιμοποιηθεί την δημιουργία και την ανάκτηση των σχέσεων μεταξύ των δεδομένων χωρίς την χρήση πολύπλοκων ερωτημάτων όπως γίνεται για παράδειγμα με την Join.

Η Neo4j ακολουθεί ένα μοντέλο δεδομένων που αναφέρεται ως native property graph model. Αυτό σημαίνει ότι τα δεδομένα αποθηκεύονται σαν σε έναν άδειο ασπροπίνακα και η βάση δεδομένων χρησιμοποιεί δείκτες (pointers) για την πλοήγηση στο γράφημα.

Το γράφημα περιέχει κόμβους (οντότητες) και αυτές οι οντότητες συνδέονται με σχέσεις (edges) που απεικονίζονται στο γράφημα. Η Neo4j δεν χρησιμοποιεί για τα δεδομένα το μοντέλο των ζευγών κλειδί-τιμή, αλλά μπορεί να προσθέτει και να αφαιρεί πεδία ανάλογα με τις απαιτήσεις.

Η απάντηση σε ένα ερώτημα που θα τεθεί είναι πολύ πιο εύκολη ακόμα και αν οι οντότητες τις οποίες αφορά δεν έχουν άμεση σχέση μεταξύ τους. Επίσης το ερώτημα μπορεί να διατυπωθεί μέσα σε 4-5 γραμμές, ενώ αν η βάση δεδομένων ήταν σχεσιακή μπορεί και να χρειαζόταν πολλές γραμμές κώδικα.

Τα δεδομένα τοποθετούνται όλα μαζί δομημένα και μη, με όποια μορφή θέλουμε και είναι πιο βολική, μέσα σε μια ‘λίμνη’ δεδομένων (data lake) όπου μπορούν

ανακτηθούν και να συσχετιστούν αργότερα. Έτσι η ενσωμάτωση των δεδομένων γίνεται πιο γρήγορα.

Χωρίς ευρετήρια ώστε να μειώνεται ο χρόνος που διαβάζονται τα δεδομένα και βελτιώνοντας έτσι την απόδοση όταν η πολυπλοκότητα των δεδομένων αυξάνεται. Η απάντηση των ερωτημάτων γίνεται με δεδομένα σε πραγματικό χρόνο(real time data).

Η Neo4j μπορεί κάνει συνεχείς διασταυρώσεις σε μεγάλα γραφήματα σε βάθος και σε πλάτος χάριν στην παραστατική αναπαράσταση των κόμβων και των μεταξύ τους σχέσεων. Δίνει την δυνατότητα επέκτασης-κλιμάκωσης μέχρι και σε δισεκατομμύρια κόμβους και σε χρήστες με μέτριας ικανότητας υλικού (hardware).

Η Neo4j επίσης παρέχει τα πλήρη χαρακτηριστικά μιας βάσης δεδομένων, είναι η μόνη εμπορική βάση δεδομένων γραφικών που συνδυάζει native αποθήκευση γραφημάτων, βελτιστοποιημένη αρχιτεκτονική κλιμάκωσης για μεγαλύτερη ταχύτητα, είναι συμβατή με τις ιδιότητες ACID διασφαλίζοντας την προβλεψιμότητα των σχέσεων που βασίζονται στα ερωτήματα, αντιμετώπιση σφαλμάτων και ανακατεύθυνση της ροής των εργασιών και έτσι κάνει την χρήση της βάσης δεδομένων γράφων ιδανική για επιχειρήσεις.

Μπορεί να διαβάζει και να γραφεί δεδομένα με ταχύτητα αστραπής διατηρώντας και προστατεύοντας ταυτόχρονα την ακεραιότητά τους, ένα χαρακτηριστικό που αρέσει στους χρήστες.

Έχει ευέλικτο σχήμα γραφικών που μπορεί να αλλάζει και να προσαρμόζεται με τον καιρό, κάνοντας ικανή την υλοποίηση και την προσθήκη νέων σχέσεων που αργότερα θα προσφέρουν συντόμευση, μειώνοντας τον χρόνο που χρειάζεται για να γίνει μια αλλαγή στα δεδομένα.

Προσφέρει οδηγούς (drivers) και μαθήματα σχεδιασμένα να προσφέρουν λεπτομερή παραδείγματα του πως να υλοποιηθεί η Neo4j με την προτιμώμενη γλώσσα προγραμματισμού που περιλαμβάνει Java, JavaScript, .NET, Python κ.α. [7], [33]

2.7 ARANGODB

Εικόνα 7. Λογότυπο της ArangoDB

Η ArangoDB είναι μια native multi-model βάση δεδομένων, δηλαδή είναι μια βάση δεδομένων που είναι σχεδιασμένη να μπορεί να υποστηρίξει πολλαπλά μοντέλα δεδομένων αντί για ένα. Η ArangoDB είναι μια ανοικτού τύπου βάση δεδομένων με ενσωματωμένο ευέλικτο μοντέλο δεδομένων για κείμενα (documents), (γράφους) graphs και ζεύγη κλειδιών-τιμής (key-values).

Κατασκευάζει εφαρμογές υψηλής επίδοσης χρησιμοποιώντας μια βολική γλώσσα ερωτημάτων που μοιάζει στην SQL ή επεκτάσεις της JavaScript.

Διενεργεί συναλλαγές σύμφωνες με τις ιδιότητες ACID, όπου αυτό απαιτείται.

Κλιμακώνεται οριζόντια και κάθετα και όλα αυτά με μερικά απλά κλικ του ποντικιού.

Η εγκατάσταση της ArangoDB σε ένα σύμπλεγμα υπολογιστών είναι τόσο εύκολη όσο η εγκατάσταση μιας εφαρμογής σε κινητό.

Το ευέλικτο μοντέλο δεδομένων και η δυνατότητα συνδυασμού των πολλών τύπων δεδομένων είναι ιδανικό για εφαρμογές κοινωνικής δικτύωσης. Η βάση δεδομένων χρησιμοποιεί την JSON για την αποθήκευση των αρχείων.

Διαθέτει μια πανίσχυρη γλώσσα ερωτημάτων την ArangoDB Query Language(AQL) για ανάκτηση και τροποποίηση των δεδομένων.

Μπορεί να χρησιμοποιηθεί ως μια υπηρεσία που παρέχει την υποδομή για την δημιουργία εφαρμογών διαδικτύου και ταυτόχρονα ένα περιβάλλον server να τρέξει η εφαρμογή αυτή. Έτσι κάνει την συνένωση εφαρμογής και βάσης δεδομένων για την μέγιστη δυνατή διακίνηση δεδομένων.

Χρησιμοποιεί μόνο την JavaScript σε όλα τα επίπεδα. Από το browser μέχρι το τελικό σύστημα υποστήριξης (backend). Σε αντίθεση με άλλες βάσεις που απαιτούν τη χρήση περισσοτέρων.

Μπορεί να εκτελεί ερωτήματα (queries) σε πολλαπλά δεδομένα ή σε συλλογές δεδομένων με επιλεκτική συνέπεια συναλλαγής ή απομόνωσης.

Η βάση δεδομένων λειτουργεί με το μοντέλο master-slave ή διαμοιράζει μεγαλύτερα σετ δεδομένων σε πολλούς servers.

Δίνει την δυνατότητα στην εφαρμογή να αποφασίσει αν χρειάζεται μονιμότητα ή υψηλή επίδοση.

Χρησιμοποιεί τους τελευταίας τεχνολογίας δίσκους SSD και μεγάλες κρυφές μνήμες (cache memory). Έτσι πετυχαίνει ταχύτατη εγγραφή και ανάγνωση δεδομένων.

Είναι fault-tolerant distributed state machine. Fault-tolerant, σημαίνει ότι δίνει την δυνατότητα στο σύστημα να εξακολουθεί να λειτουργεί κανονικά ακόμα κι αν κάποια μέρη του σταματούν να λειτουργούν. Όταν το επίπεδο των σφαλμάτων είναι υψηλό, τότε πρέπει να χρησιμοποιηθούν πολλαπλοί servers που να αποτυγχάνουν ανεξάρτητα ο ένας από τον άλλο. Συνήθως αντίγραφα από έναν server εκτελούνται σε ξεχωριστούς επεξεργαστές ενός κατανεμημένου συστήματος και χρησιμοποιούνται πρωτόκολλα που συνδέουν τις αλληλεπιδράσεις του χρήστη με τα αντίγραφα αυτά (distributed state machine).

Η ανεξαρτησία αποτυχίας των servers κατοχυρώνεται με την φυσική αλλά και την ηλεκτρολογική απομόνωση των επεξεργαστών.[34], [35]

2.8 CLUSTERPOINT

Εικόνα 8. Λογότυπο της Clusterpoint

Η Clusterpoint είναι μια Ευρωπαϊκή εταιρεία τεχνολογίας λογισμικού που ανέπτυξε και υποστηρίζει την πλατφόρμα της βάσης δεδομένων της Clusterpoint.

Η Clusterpoint είναι μια schema-free document-oriented βάση δεδομένων που μπορεί να προσπερνά τα προβλήματα πολυπλοκότητας, κλιμάκωσης και τους περιορισμούς μιας σχεσιακής βάσης δεδομένων. Ελαχιστοποιεί την προσπάθεια του χρήστη ως προς την ολοκληρωμένη διαχείριση της βάσης, την αναζήτηση και την πλατφόρμα για τις αναλύσεις (analytics).

Χρησιμοποιεί μόνο μια πλατφόρμα API όπου συνδυάζονται δεδομένα σχεσιακών βάσεων με την SQL και μηχανές αναζήτησης από διάφορες επιχειρήσεις για την διευθέτηση του ύψους της επίδοσης και της κλιμάκωσης που χρειάζονται οι διαδικτυακές εφαρμογές αλλά και οι εφαρμογές κινητών. Επίσης χρησιμοποιείται όταν ο όγκος των δεδομένων είναι μεγάλος (Big Data) και χρειάζονται εργαλεία ανάλυσης όπως η Hadoop λόγω του όγκου των δεδομένων ή και λόγω των μεγάλων υπολογιστικών φορτίων.

Τα δεδομένα τα διαχειρίζεται μέσω ενός ανοικτού ανεξάρτητου πλατφόρμας λογισμικό (cross platform), με δεδομένα τύπου XML ή JSON μέσω της API για εικονικά μηχανήματα, Python API ή JavaScript Node.js API. Υποστηρίζει

δεδομένα με διαφορετική δομή ή και μη δομημένα κείμενα, ημερομηνίες, αριθμούς, και όλους τους τύπους των δεδομένων που προκύπτουν μετά από επεξεργασία (metadata), τόσο XML όσο και JSON.

Έχει αρχιτεκτονική multi-master όπου ο οποιοδήποτε κόμβος του κατανεμημένου δικτύου μπορεί να χρησιμοποιηθεί ως master σε περίπτωση κατάρρευσης κάποιου κόμβου.

Οριζόντια κλιμάκωσης της βάσης δεδομένων από έναν server μέχρι μερικές χιλιάδες servers που συνδέονται ως μια δομή δικτύου κατανεμημένων υπολογιστών. [36]

Η αρχιτεκτονική της βάσης δεδομένων συνδυάζει μεταφορές δεδομένων σύμφωνες με τις ιδιότητες ACID, αναζήτηση κειμένου και αναλύσεις προσφέροντας υψηλή διαθεσιμότητα δεδομένων, αντιγραφή δεδομένων ώστε να είναι ασφαλή, με

ανεκτικότητα σε σφάλματα (fault tolerant) ώστε να διασφαλίζεται η συνέχιση της λειτουργίας της εφαρμογής μέχρις ενός βαθμού σφαλμάτων.

Δίνει την δυνατότητα διαμοιρασμού ενός εγγράφου με τον ίδιο τρόπο όπως θα γινόταν σε μια SQL βάση δεδομένων. Οι χρήστες μπορούν να κάνουν αναβαθμίσεις σε πραγματικό χρόνο (real time updates), αποτελέσματα αναλύσεων(analytics) σε ερωτήματα SQL τα οποία τα επεξεργάζονται και δίνουν αναφορές σε πολύ μικρό χρονικό διάστημα ακόμα και σε μεγάλες διαμοιρασμένες βάσεις δεδομένων που περιέχουν έγγραφα τύπου XML ή JSON.

Υπάρχει συνέπεια στην μεταφορά των δεδομένων, είναι συμβατή με τις περισσότερες NoSQL βάσεις δεδομένων και μπορεί να εκτελεί τις εφαρμογές που πριν ήταν διαθέσιμες μόνο σε σχεσιακές βάσεις δεδομένων με μεγάλη ταχύτητα.

Real time Big Data analytics, αντιγραφή, διαμοιρασμός φορτίων (loadsharing) και η υψηλή διαθεσιμότητα είναι κάποια βασικά χαρακτηριστικά της Clusterpoint. [37]

2.9 GOOGLE APP ENGINE

Εικόνα 9. Λογότυπο της App Engine

Η Google App Engine είναι μια πλατφόρμα δημιουργίας εφαρμογών για το διαδίκτυο (web applications) και συστημάτων υποστήριξης για κινητά. Παρέχει ενσωματωμένες υπηρεσίες και επιφάνειες διεπαφής προγραμματισμού εφαρμογών (API), NoSQL αποθήκες δεδομένων (datastores), μνήμη memcache, μια συνηθισμένη φόρμα πιστοποίησης χρηστών, load balancing συστήματα που μοιράζουν την εισερχόμενη κυκλοφορία του δικτύου σε διαφορετικά μηχανήματα πετυχαίνοντας έτσι αύξηση της απόδοσης και σταθερότητα της εφαρμογής που φιλοξενείται. Έτσι οι χρήστες μπορούν να δημιουργήσουν εφαρμογές πιο γρήγορα.

Δεν χρειάζεται server. Η Google App Engine κλιμακώνει την εφαρμογή του χρήστη αυτόματα και ανάλογα με τον όγκο της κυκλοφορίας που λαμβάνει στο internet, έτσι οι χρήστες πληρώνουν ανάλογα με την χρήση. Το μόνο που χρειάζεται ένας χρήστης είναι να συνδεθεί με τον κωδικό του και η Google καθορίζει την διαθεσιμότητα της εφαρμογής. Προσφέρεται έτσι αυτόματη κλιμάκωση βασισμένη στην ζήτηση από μηδέν μέχρι και σε εκατομμύρια χρήστες.

Διαθέτει το App Security Scan, ένα λογισμικό που σαρώνει (scan) και ελέγχει αυτόματα για τυχόν τρωτά σημεία στις εφαρμογές όπως είναι mixed content δηλαδή δεδομένα που μπορεί να προέρχονται από ασφαλείς ή και όχι πηγές και προκύπτουν όταν μια ασφαλή ιστοσελίδα επιχειρήσει να χρησιμοποιήσει πόρους που δεν είναι ασφαλείς.

Ένα άλλο είδος κινδύνου που διερευνά είναι τα αρχεία XSS (Cross-Site Scripting) που είναι ένα είδος ιού ο οποίος μολύνει ιστοσελίδες που είναι ασφαλείς.

Έτσι μπορεί να κάνει μια πρώιμη αναγνώριση των κινδύνων και να πετυχαίνει μικρό ποσοστό λαθών. Ο χρήστης μπορεί πολύ εύκολα να εγκαταστήσει, να εκτελέσει, να προγραμματίσει και να διαχειριστεί ελέγχους ασφαλείας από την κονσόλα της Google (Google Cloud Platform Console).

Στο κομμάτι της αναζήτησης, επιτρέπει στους χρήστες να κάνουν αναζητήσεις σε δομημένα δεδομένα, όπως απλό κείμενο, HTML, αριθμούς, ημερομηνίες, γεωγραφικές τοποθεσίες.

Μπορεί να κάνει αναζητήσεις σε διαφορετικές εκδόσεις των εφαρμογών, να λανσάρει σταδιακά καινούργια χαρακτηριστικά, μπορεί να εκτελεί A/B tests δηλαδή δοκιμές

που αφορούν στον σχεδιασμό της εφαρμογής για τον καθορισμό των αλλαγών που χρειάζονται για να πετύχει η εφαρμογή για παράδειγμα αύξηση πωλήσεων.

Η Google App Engine μπορεί να συνεργάζεται και με άλλα εργαλεία ανάπτυξης εφαρμογών όπως τα παρακάτω: Eclipse, Intellij, Maven, Git, Jenkins, PyCharm.

Επιτρέπει στις εφαρμογές να κάνουν εγγραφή χρηστών με τους λογαριασμούς που έχουν στην Google και να δίνουν σ' αυτούς τους χρήστες μοναδικά αναγνωριστικά (ids).

Επιπλέον επιτρέπει στους χρήστες να χρησιμοποιούν το Google Cloud SQL μια πλήρως διαχειρίσιμη υπηρεσία διαδικτύου που δίνει την δυνατότητα δημιουργίας, διαμόρφωσης και χρήση σχεσιακής βάσης δεδομένων μέσα στο υπολογιστικό νέφος της Google.

Χρησιμοποιεί την memcache, μια διανεμόμενη ενσωματωμένη κρυφή μνήμη cache που μπορεί να βελτιώσει κατά πολύ την επίδοση της εφαρμογής. [7]

2.10 IBM CLOUDANT

Εικόνα 10. Λογότυπο της IBM Cloudant

Η IBM Cloudant είναι μια NoSQL JSON document store βάση δεδομένων ιδανική για μεγάλο φόρτο εργασίας με ταυτόχρονες εγγραφές και αναγνώσεις στο υπολογιστικό νέφος. Ο φόρτος εργασίας ουσιαστικά είναι γρήγορα αναπτυσσόμενες διαδικτυακές εφαρμογές και εφαρμογές για κινητά.

Η IBM Cloudant είναι μια πλήρως διαχειρίσιμη επιφάνεια εργασίας ως υπηρεσία (DaaS) που τρέχει στο IBM Cloud.

Παρέχει στον χρήστη μια αδιάκοπη και συμφέρουσα οικονομική εμπειρία τόσο online όσο και offline, εξοικονομώντας στον χρήστη μέχρι και 95% των εξόδων του σε σχέση με το να χρησιμοποιούσε μια υποδομή που χρειαζόταν έξοδα φιλοξενίας (hosting). Είναι διαθέσιμη σε όλες τις γεωγραφικές περιοχές του IBM υπολογιστικού νέφους με περισσότερα από 55 κέντρα δεδομένων σε όλο τον κόσμο.

Έχει την ικανότητα αντιμετώπισης καταστροφών σε οποιαδήποτε ήπειρο και κλιμάκωσης εφαρμογών για μια παγκόσμια κυκλοφορία μέσω μιας οριζόντιας αρχιτεκτονικής κλιμάκωσης που δίνει την δυνατότητα χειρισμού εκατομμυρίων χρηστών και τεράστιου όγκου δεδομένων της τάξης των terrabyte, παρέχοντας μια συνεχή και απρόσκοπτη λειτουργία της επιχείρησης.

Παρέχει την δυνατότητα συγχρονισμού των βιβλιοθηκών στα κινητά και έτσι επιτρέπει στις συσκευές να έχουν πρόσβαση στα δεδομένα και στην αποθήκευσή τους τοπικά.

Όλοι οι εικονικοί servers(instances) της IBM Cloudant αναπτύσσονται σε clusters και έτσι γεφυρώνονται οι ζώνες ανάμεσα στις γεωγραφικές περιοχές, ώστε να επιτυγχάνεται επιπλέον σταθερότητα χωρίς να αυξηθεί το κόστος.

Τα δεδομένα που αποθηκεύονται στο IBM Cloud κρυπτογραφούνται over the wire με την χρήση τυχαία παραγόμενων κλειδιών. Τα υπόλοιπα, δηλαδή αυτά που είναι αποθηκευμένα (at rest) κρυπτογραφούνται με καθορισμένα κλειδιά διαχείρισης από τον χρήστη μέσω του IBM Key Protect που δημιουργεί, προσθέτει και διαχειρίζεται τα κρυπτογραφημένα κλειδιά τα οποία στη συνέχεια μπορούν να συσχετιστούν με τις βάσεις δεδομένων στο υπολογιστικό νέφος και να κρυπτογραφηθούν οι ίδιες οι βάσεις δεδομένων. [38]

Η IBM Cloudant προσφέρει την υπηρεσία EU-managed που διαβεβαιώνει ότι όλα τα δεδομένα και τις υπηρεσίες, τις διαχειρίζονται Ευρωπαίοι πολίτες. Μετακινεί τις εφαρμογές πιο κοντά στις περιοχές που χρειάζονται για μια αδιάκοπη πρόσβαση στα δεδομένα online ή offline.

Οι χρήστες μπορούν να επιλέξουν αν η βάση δεδομένων τους θα είναι σε server με έναν μόνο ‘ενοικιαστή’ (on bare metal computer) ή σε εικονική μηχανή(virtual machine)

Η αρχιτεκτονική offline-first είναι μια προσέγγιση όπου μια εφαρμογή μπορεί να σχεδιαστεί και να γραφεί σαν να μην υπάρχει καθόλου σύνδεση στο internet. Με τον τρόπο αυτό τα δεδομένα γράφονται τοπικά στον υπολογιστή του χρήστη και περιοδικά φορτώνονται και αντιγράφονται στο υπολογιστικό νέφος. Η IBM Cloudant βοηθά ομάδες να δημιουργήσουν Progressive Web Apps δηλαδή εφαρμογές που χρησιμοποιούν τις σύγχρονες δυνατότητες που προσφέρει το διαδίκτυο όπως η αρχιτεκτονική του offline first ή μπορεί να χρησιμοποιεί δεδομένα που έχουν υποστεί επεξεργασία ώστε να είναι πιο οργανωμένα. Οι εφαρμογές αυτές θα μπορούν να λειτουργούν ακόμα και σε συσκευές τελευταίας τεχνολογίας.

Η IBM Cloudant φιλοξενεί (host) βάσεις δεδομένων που έχουν πολλούς διαφορετικούς providers, συμπεριλαμβανομένων της Amazon, Rackspace, SoftLayer, Microsoft Azure.

Φέρει τις πιστοποιήσεις ISO 27001, Soc 2 type 2, HIPAA αναγνωρισμένα πρότυπα τα οποία προσδιορίζουν τις προδιαγραφές για την διαχείριση της ασφάλειας των πληροφοριών.[7]

2.11 AMAZON SIMPLE DB

Εικόνα 11. Λογότυπο της Amazon Simple DB

Η Amazon SimpleDB είναι μια ανοικτού τύπου, υψηλής διαθεσιμότητας NoSQL βάση δεδομένων, data store με την οποία ο χρήστης δεν έχει καθόλου τα βάρη της διαχείρισης. Είναι μια βάση δεδομένων που λειτουργεί σε επιφάνεια εργασίας ως υπηρεσία (DaaS). Οι προγραμματιστές απλά αποθηκεύουν δεδομένα, θέτουν ερωτήματα (queries) και παίρνουν απαντήσεις μέσω των υπηρεσιών του διαδικτύου και η Amazon SimpleDB κάνει τα υπόλοιπα.

Απελευθερωμένη από τις αυστηρές απαιτήσεις μιας σχεσιακής βάσης δεδομένων, η Amazon SimpleDB είναι βελτιωμένη και μπορεί να προσφέρει υψηλή διαθεσιμότητα και ευελιξία με ελάχιστο ή και καθόλου διαχειριστικό βάρος.

Προκειμένου να επιτύχει υψηλή διαθεσιμότητα και σταθερότητα των δεδομένων, η Amazon SimpleDB δημιουργεί, διαχειρίζεται και 'ευρετηριάζει' (index) πολλά αντίγραφα των δεδομένων τα οποία τα διαμοιράζει σε διαφορετικές γεωγραφικές περιοχές.

Με την Amazon SimpleDB ο χρήστης επικεντρώνεται στην ανάπτυξη εφαρμογών χωρίς να τον απασχολεί η μετάδοση των δεδομένων, η διαθεσιμότητά τους, η συντήρηση του λογισμικού, η μορφή (schema) των δεδομένων. Έτσι μπορεί να επικεντρώνεται αποκλειστικά και μόνο στην ανάπτυξη εφαρμογών παρά στην χρονοβόρα και επίπονη διαδικασία της διαχείρισης μιας βάσης δεδομένων.

Η Amazon SimpleDB διαχειρίζεται αυτόματα την μετάδοση των δεδομένων, την συντήρηση του λογισμικού και του υλικού, την αντιγραφή των αντικειμένων και τον συγχρονισμό των εργασιών. Δημιουργεί αυτόματα πολλαπλά αντίγραφα των αποθηκευμένων δεδομένων τα οποία τα διαμοιράζει σε όλες τις γεωγραφικές περιοχές. Αυτό παρέχει υψηλή διαθεσιμότητα και σταθερότητα. Στην απίθανη περίπτωση που το αντίγραφο αποτύχει, η Amazon SimpleDB ανακατευθύνει το αντίγραφο κάπου αλλού.

Η Amazon SimpleDB παρέχει αποδοτικότερη πρόσβαση στα αποθηκευμένα δεδομένα και συναρτήσεις ερωτημάτων που παραδοσιακά χρησιμοποιούνται από δίκτυα σχεσιακών βάσεων δεδομένων, αφήνοντας από την άλλη στην άκρη άλλες πολύπλοκες και όχι και τόσο συχνά χρησιμοποιούμενες λειτουργίες μιας σχεσιακής βάσης δεδομένων.

Τα δεδομένα αποθηκεύονται με μια ελαστική, χωρίς συγκεκριμένο σχήμα (schema-less) μορφή όπου μπορούμε εύκολα να επεκτείνουμε προσθέτοντας ιδιότητες που προκύπτουν από την επεξεργασία των δεδομένων χωρίς να 'σπάσει' η αυστηρή δομή των δεδομένων.

Η κλιμάκωση του δικτύου γίνεται χωρίς κάποια ενέργεια από τη μεριά του προγραμματιστή. Η βάση δεδομένων απλά 'αντιδρά' στην κυκλοφορία του Internet και αυτόματα ζητά αύξηση του όγκου χρήσης του δικτύου.

Διαθέτει υπηρεσία που επιτρέπει στους χρήστες να προσθέτουν γρήγορα δεδομένα και να ανακτούν ή να επεξεργάζονται εύκολα τα δεδομένα αυτά μέσα από ένα απλό σετ από API calls δηλαδή από κλήσεις που γίνονται μέσω του API σε έναν server.

Το ποσό της πληρωμής υπολογίζεται μετά από την ενημέρωση των αναφορών χρήσης του Internet, όπου φαίνονται με λεπτομέρεια τα δεδομένα που χρησιμοποιήθηκαν. Έτσι είναι πιο συμφέρον μιας και χρεώνει μόνο τους πόρους που καταναλώνει ο χρήστης. [7]

2.12 ORACLE CLOUD

Εικόνα 12. Λογότυπο της Oracle Cloud

Η Oracle Cloud είναι μια υπηρεσία που προσφέρεται από την Oracle Corporation και παρέχει servers, αποθήκευση, δίκτυο, εφαρμογές και υπηρεσίες μέσα από ένα παγκόσμιο δίκτυο κέντρων δεδομένων(data centers). Η εταιρεία επιτρέπει αυτές οι υπηρεσίες να παρέχονται κατά παραγγελία στο Internet.

Η Oracle Cloud είναι μια NoSQL βάση δεδομένων που λειτουργεί ως υπηρεσία σε επιφάνεια εργασίας (DaaS) ή ως υποδομή σε επιφάνεια εργασίας(IaaS) ή ως υπηρεσία σε πλατφόρμα (PaaS) ή και ως υπηρεσία λογισμικού (SaaS) ως μια πλήρως διαχειρίσιμη υπηρεσία υπολογιστικού νέφους. Είναι μια key-valued store βάση δεδομένων που υποστηρίζει JSON, με εγγυημένες την ευέλικτη μεταφορά δεδομένων.

Μηδενικά βάρη διαχείρισης. Η Oracle NoSQL δεν χρειάζεται server αλλά παρέχει πλήρη υποστήριξη διαχείρισης που εξοπλίζει την εφαρμογή με αυτόματη κλιμάκωση, μικρές καθυστερήσεις και υψηλή διαθεσιμότητα δεδομένων, ώστε οι προγραμματιστές απλά συνδέουν τις εφαρμογές στην Oracle NoSQL και άμεσα διαβάζουν και γράφουν δεδομένα.

Εξασφαλίζει μια γρήγορη και συνεπή εκτέλεση. Οι προγραμματιστές μπορούν να επικεντρωθούν στην δημιουργία εφαρμογών υψηλής λειτουργικότητας, ενώ η Oracle NoSQL φροντίζει να υπάρχουν μικρές και προβλεπόμενες καθυστερήσεις της τάξης των μονοψήφιων milliseconds. Επίσης φροντίζει για την αυτόματη κλιμάκωση ακόμα κι αν υπάρχει μεγάλος φόρτος εργασίας. [39]

Υποστηρίζει ευέλικτα μοντέλα δεδομένων. Χρησιμοποιεί key valued δεδομένου σχήματος ή ad-hoc JSON δηλαδή κάποια έτοιμα κομμάτια κώδικα σε JSON που λύνουν κάποια συγκεκριμένα προβλήματα των προγραμματιστών και δεν χρειάζονται να γράφουν τεράστια τμήματα κώδικα από την αρχή.

Εκμεταλλεύεται τα οφέλη τόσο μιας NoSQL βάσης δεδομένων όσο και μιας SQL σε μία μόνο πλατφόρμα. Έτσι συνδυάζει την ευελιξία των δεδομένων, την γρήγορη εύρεση και την εύκολη κλιμάκωση της NoSQL με την SQL που βασίζεται σε ερωτήματα και η οποία είναι εκφραστική και έχει ανεξάρτητη πρόσβαση στα δεδομένα.

Οι εφαρμογές είναι εύκολα εξελίξιμες, εκτελέσιμες και κλιμακούμενες. Με μικρότερο κόστος, προσπάθεια και καθυστερήσεις και αυτό είναι και προς όφελος

των επιχειρήσεων που θέλουν να έχουν γρήγορη πρόσβαση στα δεδομένα τους ώστε να έχουν εικόνα για την πορεία της επιχείρησης.[40]

Ως προς την ασφάλεια. Ακολουθεί το SSL πρωτόκολλο σύμφωνα με το οποίο η ανταλλαγή πληροφοριών είναι κωδικοποιημένη και ασφαλής. Χρησιμοποιεί κρυπτογράφηση των δεδομένων at rest δηλαδή των δεδομένων που είναι αποθηκευμένα και που δεν χρησιμοποιούνται σε καθημερινή βάση.

Παρέχει την αυθεντικοποίηση (authentication) OAuth, ένα πλαίσιο που επιτρέπει στις εφαρμογές να αποκτούν περιορισμένη πρόσβαση στους λογαριασμούς των χρηστών σε μια υπηρεσία που ακολουθεί το HTTP πρωτόκολλο, όπως είναι το Facebook, to GitHub, to Digital Ocean. Η πρόσβαση αυτή ισχύει για σταθερές συσκευές αλλά και για κινητά.

Δίνει περιορισμένη πρόσβαση μόνο σε εξουσιοδοτημένους χρήστες. Αυτό χρησιμοποιείται από τις περισσότερες επιχειρήσεις με πάνω από 500 υπαλλήλους.[39]

2.13 EnterpriceDB PPCD

Εικόνα 13. Λογότυπο της EnterpriceDB

Η EnterpriceDB (EDB) είναι μια ιδιωτική εταιρεία που βρίσκεται στην Μασαχουσέτη και η οποία παρέχει λογισμικό και υπηρεσίες βασισμένες σε δωρεάν και ανοικτού τύπου σχεσιακή βάση δεδομένων, την PostgreSQL. Η PostgreSQL έχει σχεδιαστεί να μπορεί να διαχειρίζεται μια μεγάλη γκάμα φορτίων, από ένα μηχάνημα μέχρι αποθήκες δεδομένων (data warehouses) ή ακόμα και υπηρεσίες Διαδικτύου (Web services) με πολλούς χρήστες ταυτόχρονα.

Η EDB έχει δημιουργήσει την βάση δεδομένων να είναι συμβατή με την Oracle ώστε να διευκολύνει την μετακίνηση των φορτίων εργασίας από την Oracle στην EDB Postgres.

Η EDB δίνει την δυνατότητα εγγραφής επί πληρωμή στην πλατφόρμα της ως ένας server, EDB Postgres Advanced Server ή ως υπηρεσία DaaS. Η εγγραφή στην πλατφόρμα συμπεριλαμβάνει εργαλεία, αναβαθμίσεις, υποστήριξη και συντήρηση.

Η Postgres Plus Cloud Database (PPCD) είναι μια NoSQL βάση δεδομένων που ικανοποιεί και τις πιο απαιτητικές ανάγκες των επιχειρηματικών εφαρμογών.

Παρέχει ελαστικότητα κλιμάκωσης, υψηλή διαθεσιμότητα και όλα αυτά μέσω ενός πίνακα ελέγχου. Παρέχει υποστήριξη σε διάφορες βάσεις δεδομένων που είναι συμβατές με τις ιδιότητες ACID και με τις NoSQL βάσεις δεδομένων με δεδομένα τύπου document oriented και key-valued.

Η κονσόλα του PPCD χρησιμοποιείται για να παραταχθούν οι Postgres βάσεις δεδομένων σε cloud server instances (εικονικοί server στο υπολογιστικό νέφος). Δημιουργούνται συμπλέγματα (clusters) εικονικών servers στα οποία υπάρχουν μία master βάση δεδομένων και οι υπόλοιπες είναι αντίγραφα.

Κάθε σύμπλεγμα περιλαμβάνει έναν load balancer στον οποίο μπορούν να συνδέονται οι εφαρμογές. Ο load balancer λαμβάνει τα αιτήματα των εφαρμογών και διαμοιράζει τα αιτήματα των εγγραφών σε όλους τους εικονικούς servers του cluster που είναι read-only. Τα αιτήματα αυτά τελικά καταλήγουν στον αρχικό server δηλαδή τον master. Από την μεριά της εφαρμογής, η σύνδεση στην PostgreSQL ή στην PostgreSQL Plus Advanced Server που βρίσκεται στο υπολογιστικό νέφος δεν έχει καμία διαφορά από τη σύνδεση σε έναν εικονικό server που βρίσκεται σε κάποιον απλό υπολογιστή.

Στην αρχική σύνδεση της εφαρμογής με την Postgres, γίνεται αυθεντικοποίηση και είναι καθορισμένος ο ρόλος του κάθε χρήστη από την αρχή. Έτσι η επιφάνεια

εργασίας και τα απαραίτητα εργαλεία μπορούν να συνδέονται απευθείας στη βάση δεδομένων χρησιμοποιώντας τις πληροφορίες της αρχικής σύνδεσης.

Σε περίπτωση κατάρρευσης ενός server γίνεται αυτόματα ανακατεύθυνση της σύνδεσης και έτσι ελαχιστοποιείται η απώλεια των δεδομένων. Επίσης η αποκατάσταση γίνεται το ταχύτερο δυνατόν.

Διαθέτει μια συσκευή λογικής διαχείρισης του όγκου αποθήκευσης (Logical Volume Manager - LVM) με την βοήθεια της οποίας η κλιμάκωση του όγκου της αποθήκευσης των δεδομένων γίνεται αυτόματα χωρίς να χρειάζεται να διακοπεί η λειτουργία των εφαρμογών. Τα δεδομένα μπορούν να αποθηκευτούν κρυπτογραφημένα. Επίσης διακόπτει την αποθήκευση των δεδομένων στο υποδίκτυο (cluster) προκειμένου να διευκολύνει την άνευ σφαλμάτων κλιμάκωση της αποθήκευσης στο υπολογιστικό νέφος, χωρίς να επηρεάζεται η λειτουργία των βάσεων δεδομένων.

Για την ανάπτυξη των βάσεων δεδομένων χρησιμοποιούνται ιδιωτικές εικονικές μηχανές που δεν διαμοιράζονται ποτέ με άλλους πελάτες. Αυτό όχι μόνο αυξάνει την ασφάλεια και την σταθερότερη εκτέλεση, αλλά επιτρέπει στο χρήστη να χειρίζεται την βάση δεδομένων σαν να είναι δική του, δίνοντάς του την δυνατότητα να κάνει οποιαδήποτε διαμόρφωση και λειτουργική αλλαγή του χρειάζεται. [41], [42]

2.14 AMAZON DYNAMODB

Εικόνα 14. Λογότυπο της Amazon DynamoDB

Η Amazon DynamoDB είναι μια NoSQL βάση δεδομένων, key-valued και document oriented, η οποία μπορεί να λειτουργεί πολύ καλά σε οποιαδήποτε κλίμακα.

Είναι πλήρως διαχειρίσιμη, πολυεπίπεδη, multi-master βάση δεδομένων που λειτουργεί με μηδαμινές καθυστερήσεις. Προσφέρει ασφάλεια δεδομένων, δυνατότητα backup και ανάκτησης δεδομένων.

Περισσότεροι από 100.000 πελάτες της AWS (Amazon Web Services) μιας ασφαλούς πλατφόρμας υπηρεσιών cloud, έχουν επιλέξει την DynamoDB για κινητό, για το διαδίκτυο, για διαφημίσεις τεχνολογίας, για IoT και για πολλές άλλες εφαρμογές που απαιτούν μικρές καθυστερήσεις στην πρόσβαση των δεδομένων, μιας και η DynamoDB μπορεί και ανταποκρίνεται σε milliseconds σε οποιαδήποτε κλίμακα.

Η DynamoDB έχει VPC (Virtual Private Cloud) endpoints που είναι εικονικές συσκευές και δίνουν την δυνατότητα σύνδεσης στις υπηρεσίες της AWS όπου τα τελικά σημεία ενεργοποιούνται χωρίς να χρειάζεται να υπάρχει πρόσβαση στο internet. [43]

Οι εφαρμογές έχουν απεριόριστο χώρο διακίνησης και αποθήκευσης δεδομένων. Η in-memory cache μειώνει τον χρόνο απόκρισης από milliseconds σε microseconds χωρίς να αλλάξει κάτι άλλο στην εφαρμογή.

Η DynamoDB είναι μια βάση δεδομένων χωρίς server οποία μπορεί να κλιμακωθεί αυτόματα και η οποία μπορεί να κρατά συνεχώς αντίγραφα των δεδομένων ώστε να είναι ασφαλή.

Με την DynamoDB δίνεται η δυνατότητα σε εφαρμογές που υπάρχουν παγκοσμίως να έχουν γρήγορα πρόσβαση στα τοπικά δεδομένα μέσω αντιγραφής των δεδομένων σε όλες τις γεωγραφικές περιοχές (regions) της AWS.

Δίνει την δυνατότητα δημιουργίας ευέλικτων και επαναχρησιμοποιούμενων μικρουπηρεσιών που χρησιμοποιούν την DynamoDB σαν μια αποθήκη δεδομένων (data store) για συνεχή και γρήγορη εκτέλεση όπως επίσης και προσωποποιημένες εφαρμογές για κινητά που προσφέρουν ικανοποιητική και εξατομικευμένη εμπειρία στους χρήστες. Η DynamoDB αναλαμβάνει τις λειτουργικές εργασίες και έτσι ο χρήστης επικεντρώνεται μόνο στις εφαρμογές.

Η DynamoDB χρησιμοποιεί πρωτεύοντα κλειδιά για να αναγνωρίσει μοναδικά τα στοιχεία ενός πίνακα και δευτερεύοντα κλειδιά ώστε να υπάρχει επιπλέον ευελιξία στην διατύπωση των ερωτημάτων (queries). Χρησιμοποιεί partition key's value σαν είσοδο σε μια εσωτερική συνάρτηση καταμερισμού. Η έξοδος της συνάρτησης καταμερισμού καθορίζει την διαμέριση (partition) με την οποία θα αποθηκευτεί το αντικείμενο.

Η DynamoDB υποστηρίζει πολύ την συνοχή των δεδομένων ώστε οι εφαρμογές να διαβάζουν τις πιο πρόσφατες τιμές.

Επίσης έχει ενσωματωμένους μετρητές που επιτρέπουν στους χρήστες να αλλάζουν ο καθένας ατομικά τις διάφορες ιδιότητες μέσω της επιφάνειας εργασίας του προγραμματισμού εφαρμογών (API).[7]

Έχει υπηρεσίες όπως είναι το Amazon Redshift και Amazon QuickSight.

Το Amazon Redshift είναι μία πλήρως διαχειρίσιμη αποθήκη δεδομένων στο υπολογιστικό νέφος, της τάξης των petabyte. Η αρχική αποθήκη μπορεί να είναι χωρητικότητας μερικών εκατοντάδων gigabytes και να κλιμακωθεί φτάνοντας την χωρητικότητα των petabytes ή και περισσότερο. Αυτό δίνει την δυνατότητα στον χρήστη να αποκτήσει περισσότερες πολύτιμες γνώσεις (insights) για την επιχείρηση και τους πελάτες.

Το πρώτο βήμα για την δημιουργία μιας αποθήκης δεδομένων είναι να εκκινήσουν ένα σετ κόμβων που ονομάζεται Amazon Redshift cluster. Μόλις ξεκινήσει η λειτουργία του δικτύου, μπορούν να φορτωθούν τα δεδομένα και να ξεκινήσει η ανάλυση των ερωτημάτων (queries). Ασχέτως του όγκου των δεδομένων, το Amazon Redshift μπορεί να εκτελεί τα ερωτήματα πολύ γρήγορα χρησιμοποιώντας τα ίδια εργαλεία που βασίζονται στην SQL και στις εφαρμογές επιχειρηματικής νοημοσύνης και που χρησιμοποιούνται για να βρουν και να αναλύσουν τα δεδομένα μιας επιχείρησης (analytics) και να βγάλουν συμπεράσματα επηρεάζοντας έτσι τις αποφάσεις της διοίκησης.[43]

Το **Amazon QuickSight** είναι μια γρήγορη υπηρεσία επιχειρηματικής νοημοσύνης στο υπολογιστικό νέφος, η οποία μπορεί να προσφέρει σημαντικές πληροφορίες (insights) σε οποιονδήποτε μέσα σε μια επιχείρηση.

Είναι μια πλήρως διαχειρίσιμη υπηρεσία μέσω πινάκων ελέγχου οι οποίοι μπορούν να δημιουργηθούν και να δημοσιοποιηθούν από τον καθένα. Είναι μια υπηρεσία επί πληρωμή ανάλογα με την συνεδρία (pay-per-session). Έτσι αντί να υπάρχει μια μηνιαία συνδρομή, υπάρχει μια χρέωση ανά αναγνώστη της τάξης των $0.30 για μια τριαντάλεπτη συνεδρία με μέγιστο κόστος χρέωσης $5 το μήνα για τον κάθε αναγνώστη. Οι χρήστες έχουν πρόσβαση σε ασφαλείς και διαδραστικούς πίνακες ελέγχου, σε αναφορές (reports) στο email χωρίς προπληρωμένα έξοδα και σύνθετες διαδικασίες.

Οι πίνακες ελέγχου περιλαμβάνουν ML Insights (Machine Learning Insights) τα οποία είναι πολύ ισχυρά χαρακτηριστικά και από τα οποία μπορούμε να ανακαλύψουμε κρυμμένες τάσεις, ακραίες τιμές εκτός πλαισίου, να αναγνωρίσουμε τα κλειδιά και τις δραστηριότητες που δίνουν λειτουργικά και οικονομικά αποτελέσματα σε μια επιχείρηση όπως για παράδειγμα το πλήθος και την τιμή προϊόντων που έχουν πωληθεί, να κάνουν ανάλυση των στοιχείων χωρίς να χρειάζεται να είναι κάποιος ειδικός.

Οι πίνακες ελέγχου είναι προσβάσιμοι από οποιαδήποτε συσκευή και είναι ενσωματωμένοι στις εφαρμογές και στις σελίδες στο διαδίκτυο.[43], [44]

2.15 BIGTABLE

Εικόνα 15. Λογότυπο της BigTable

Η Bigtable είναι ένα συμπιεσμένο, κλειστού τύπου σύστημα αποθήκευσης, υψηλής επίδοσης βασισμένο σε τεχνολογίες της Google, όπως είναι οι Google Fie System, Chubby Lock Service, SSTable.

Το 2015 μια έκδοση της Bigtable έγινε διαθέσιμη ως υπηρεσία. Υπόκειται στην Google Cloud Datastore η οποία αποτελεί ένα μέρος της Google Cloud Platform.

Η Bigtable είναι ένα πρωτότυπο παράδειγμα ενός τύπου NoSQL βάσης δεδομένων που λέγεται wide column store. Πρόκειται για μια δισδιάστατη key-value store που χρησιμοποιεί πίνακες, γραμμές, στήλες αλλά η μορφή των στηλών του ίδιου πίνακα μπορεί να διαφέρει από γραμμή σε γραμμή. Για το λόγο αυτό δεν θεωρείται απλώς μια απλή column store βάση δεδομένων. Αντιστοιχίζει δύο τυχαίες συμβολοσειρές ως κλειδιά για τις στήλες και τις γραμμές και σημειώνει το χρόνο που αυτές οι τιμές σχετίστηκαν (timestamping).

Έχει σχεδιαστεί να μπορεί να κλιμακωθεί σε petabytes σε εκατοντάδες ή χιλιάδες μηχανήματα και είναι πολύ εύκολο να προστεθούν κι άλλα, τα οποία αυτόματα μπορούν να χρησιμοποιήσουν τους πόρους του συστήματος, χωρίς να χρειάζεται καν να γίνει κάποια αναπροσαρμογή. Για παράδειγμα, τα αντίγραφα της Google μπορούν να αποθηκευτούν σε έναν μεγάλο πίνακα όπου το κλειδί θα είναι το domain name της σελίδας, οι στήλες θα περιγράφουν διάφορες ιδιότητες της σελίδας του διαδικτύου με μια από τις στήλες αυτές να ‘κρατάει’ την ίδια την σελίδα. Η στήλη που κρατάει την σελίδα μπορεί να έχει διάφορες χρονοσημασμένες εκδόσεις της σελίδας οι οποίες θα περιγράφουν διαφορετικά αντίγραφα της σελίδας από την χρονική στιγμή που έχει καταχωριστεί στην βάση δεδομένων. Κάθε κελί μπορεί να έχει από καμία έως και κάποιες χρονοσημασμένες εκδόσεις των δεδομένων.

Η χρονοσήμανση (timestamping) εκτός από το ότι κρατάει τις διαφορετικές εκδόσεις, χρησιμοποιείται και για την συλλογή δεδομένων πολύ παλιών τα οποία δεν ισχύουν πλέον.[45]

2.16 GOOGLE CLOUD DATASTORE

Εικόνα 16. Google Cloud Datastore

Η Google Cloud Datastore είναι μια document oriented βάση δεδομένων υψηλής επίδοσης που μπορεί να κλιμακωθεί αυτόματα και χρησιμοποιείται για την ανάπτυξη εφαρμογών.

Η Google Cloud Datastore μπορεί να εκτελεί μια συλλογή λειτουργιών όπου ή όλες πετυχαίνουν ή καμία (ατομικότητα).

Παρέχει μεγάλη διαθεσιμότητα δεδομένων για εγγραφή και ανάγνωση. Η βάση δεδομένων λειτουργεί σε κέντρα δεδομένων της Google που αποθηκεύουν τα ίδια δεδομένα σε διαφορετικές τοποθεσίες ώστε να ελαχιστοποιούν την επίδραση από την μη λειτουργία κάποιου ενδιάμεσου κόμβου.

Παρουσιάζει τεράστια κλιμάκωση με υψηλές επιδόσεις. Χρησιμοποιεί μια αρχιτεκτονική διαμοιρασμού σύμφωνα με την οποία γίνεται αυτόματα κλιμάκωση και σε τέτοιο βαθμό, ανάλογα με τα αποτελέσματα των ερωτημάτων και όχι ανάλογα με την συλλογή των δεδομένων που διαθέτουμε.

Διαθέτει ευελιξία στην αποθήκευση και στην δημιουργία ερωτημάτων. Αντιστοιχίζει τα ερωτήματα στα αντικείμενα (object-oriented) και χρησιμοποιεί γλώσσες προγραμματισμού σεναρίων, όπου τα ‘σενάρια’ έχουν τον βασικό κώδικα της εφαρμογής και συχνά δημιουργούνται ή τροποποιούνται από τον τελικό χρήστη. Έτσι επιτρέπεται ο έλεγχος περισσότερων εφαρμογών. Επίσης διαθέτει μια γλώσσα ερωτημάτων που μοιάζει με την SQL.

Παρέχει ισορροπία ισχυρής συνέπειας. Διαβεβαιώνει ότι η ζητούμενη οντότητα μεταφέρεται με συνέπεια, επιτρέποντας στις εφαρμογές να χειριστούν ένα μεγάλο όγκο δεδομένων και κάνοντας την εμπειρία του χρήστη από την εφαρμογή, ευχάριστη.

Χρησιμοποιεί κρυπτογράφηση των δεδομένων at rest δηλαδή των δεδομένων που είναι αποθηκευμένα και που δεν χρησιμοποιούνται σε καθημερινή βάση και όταν είναι να διαβαστούν από εξουσιοδοτημένο χρήστη, τα αποκρυπτογραφεί αυτόματα.

Ο χρήστης δεν χρειάζεται να ασχοληθεί καθόλου με την διαχείριση καθώς όλα τα βάρη της διαχείρισης ρυθμίζονται από την υπηρεσία. Έτσι ενώ η εφαρμογή

χρησιμοποιεί την υπηρεσία της Cloud Datastore, μπορεί να λαμβάνει ειδοποιήσεις για αναβάθμιση.[46]

2.17 AZURE COSMOS DB

Εικόνα 17. Λογότυπο της Azure Cosmos DB

Η Azure CosmosDB είναι μια κλειστού τύπου πολυμοντελιακή (multi-model) υπηρεσία βάσης δεδομένων της Microsoft. Είναι σχεδιασμένη να μπορεί να διαχειρίζεται δεδομένα σε επίπεδο πλανήτη. Είναι μια NoSQL βάση δεδομένων η οποία κλιμακώνεται οριζόντια.

Όσον αφορά στα δεδομένα, αποθηκεύονται αντικείμενα (items) μέσα σε κοντέινερ (containers). Τα κοντέινερ οργανώνονται σε βάσεις δεδομένων που έχουν περιεχόμενο ανάλογο με το όνομα των κοντέινερς. Τα κοντέινερ είναι δομές χωρίς συγκεκριμένο σχήμα (schemaless). Όταν τοποθετείται ένα item σε ένα container, αυτόματα μπαίνει σε ευρετήριο.

Κάθε container του CosmosDB δίνει την δυνατότητα στους χρήστες να κάνουν εγγραφή και να λαμβάνουν ειδοποιήσεις όταν εισάγονται νέα items ή όταν αναβαθμίζονται items σε ένα container. Αλλαγές στα items μπορούν να γίνουν από οποιοδήποτε σημείο και σε οποιαδήποτε στιγμή από τότε που δημιουργήθηκε το container. Υπάρχει η δυνατότητα αυτόματης διαγραφής ενός item μετά από κάποιο καθορισμένο χρονικό διάστημα από την τελευταία αναβάθμιση.

Χρησιμοποιεί κλειστού τύπου λογισμικό διασύνδεσης εφαρμογών SQL API. Επίσης χρησιμοποιεί τέσσερα διαφορετικά APIs που είναι συμβατά με άλλες βάσεις δεδομένων όπως η MongoDB, η Gemlin, η Cassandra και η Azure Table Storage. Η συμβατότητα αυτή, δίνει την δυνατότητα σε εφαρμογές να συνδεθούν και να ανταλλάξουν δεδομένα ανά την υφήλιο.

Το SQL API επιτρέπει στους πελάτες να δημιουργούν, να αναβαθμίζουν, να διαγράφουν containers και items. Στα items μπορούν να τεθούν ερωτήματα με τη χρήση SQL και JSON. Υπάρχουν συναρτήσεις που μετατρέπουν σύνθετες και τυχαίες λειτουργίες σε λογικές και σύμφωνες με τις ιδιότητες ACID. Τα δεδομένα είναι απομονωμένα από τις αλλαγές που γίνονται όσο εκτελείται η διαδικασία της αποθήκευσης και έτσι η βάση δεδομένων παρουσιάζει συνέπεια.

Από το 2016 έχει προστεθεί η δυνατότητα αυτόματου κατακερματισμού (partitioning) των δεδομένων σε κατακερματισμένα containers, τα οποία ‘γεφυρώνουν’ πολλά partitions με items τα οποία τα διαχειρίζεται ο χρήστης μέσω ενός partition key. Η CosmosDB αποφασίζει αυτόματα σε πόσα partitions θα μοιράσει τα δεδομένα λαμβάνοντας υπόψιν το μέγεθος των δεδομένων και τις ανάγκες του throughput. Όταν προστίθενται ή αφαιρούνται partitions γίνεται ανακατανομή και έτσι τα δεδομένα παραμένουν διαθέσιμα στα εναπομείναντα partitions.

Οι CosmosDB βάσεις δεδομένων μπορούν να είναι διαθέσιμες σε οποιαδήποτε από τις 55+ γεωγραφικές περιοχές της Microsoft Azure, επιτρέποντας στους

προγραμματιστές να τοποθετούν τα δεδομένα τους το κοντινότερο δυνατόν στο σημείο που τα χρησιμοποιούν. Τα δεδομένα του κάθε container αντιγράφονται σε όλες τις ρυθμισμένες περιοχές. Η πρόσθεση ή η αφαίρεση περιοχών γίνεται χωρίς καθυστερήσεις και επίδραση στην εκτέλεση των εργασιών. Όλες οι απαιτούμενες ενέργειες γίνονται αυτόματα από την βάση δεδομένων χωρίς την επέμβαση του χρήστη.

Το αρχικό μοντέλο της CosmosDB περιελάμβανε μόνο μια περιοχή εγγραφής και οι άλλες περιοχές ήταν αντίγραφα με δυνατότητα μόνο ανάγνωσης (read-only). Όμως από τον Μάρτιο του 2018 προστέθηκε η δυνατότητα πραγμάτωσης του μοντέλου multi-master όπου πολλές περιοχές μπορούν να είναι αντίγραφα με δυνατότητα εγγραφής και αυτό σε ένα δίκτυο παγκόσμιο. Περιπτώσεις 'διαμάχης' που πιθανόν προκύπτουν όταν διαφορετικές περιοχές εγγραφής θέλουν να κάνουν ταυτόχρονα εγγραφή, λύνονται είτε με την πολιτική που λέει ότι 'κερδίζει' η τελευταία εγγραφή ή με την χρήση μιας Javascript συνάρτησης.

Υπάρχουν όμως και κάποια μειονεκτήματα:

Ενώ γίνονται αυτόματα backups, μόνο τα δύο τελευταία από αυτά κρατούνται για διάστημα μεγαλύτερο των 8 ωρών. Επίσης η ανάκτηση των backups γίνεται με την βοήθεια της ομάδας υποστήριξης της Microsoft. Παρόλο που υπάρχει προστασία ολόκληρων συλλογών, τα μεμονωμένα έγγραφα δεν είναι προστατευμένα και έτσι για παράδειγμα μπορεί να διαγραφούν κατά λάθος ή να αλλαχθούν σκόπιμα με αποτέλεσμα να μην είναι πολύ ικανοποιητική βάση δεδομένων για μεγάλες επιχειρήσεις.

Δεν έχει ενσωματωμένους drivers για τις γλώσσες που χρησιμοποιεί με αποτέλεσμα πολλές φορές οι προγραμματιστές να έχουν δυσκολίες όταν ένας κώδικας αποτυγχάνει να λειτουργήσει σε άλλο σύστημα. [47]

3. Μελέτη Περίπτωσης

Στο πλαίσιο της μελέτης περίπτωσης κατασκευάστηκαν δύο παρόμοιες βάσεις δεδομένων μία σε Access που είναι μια σχεσιακή βάση δεδομένων και μια στην MongoDB Atlas που είναι μια NoSQL βάση δεδομένων.

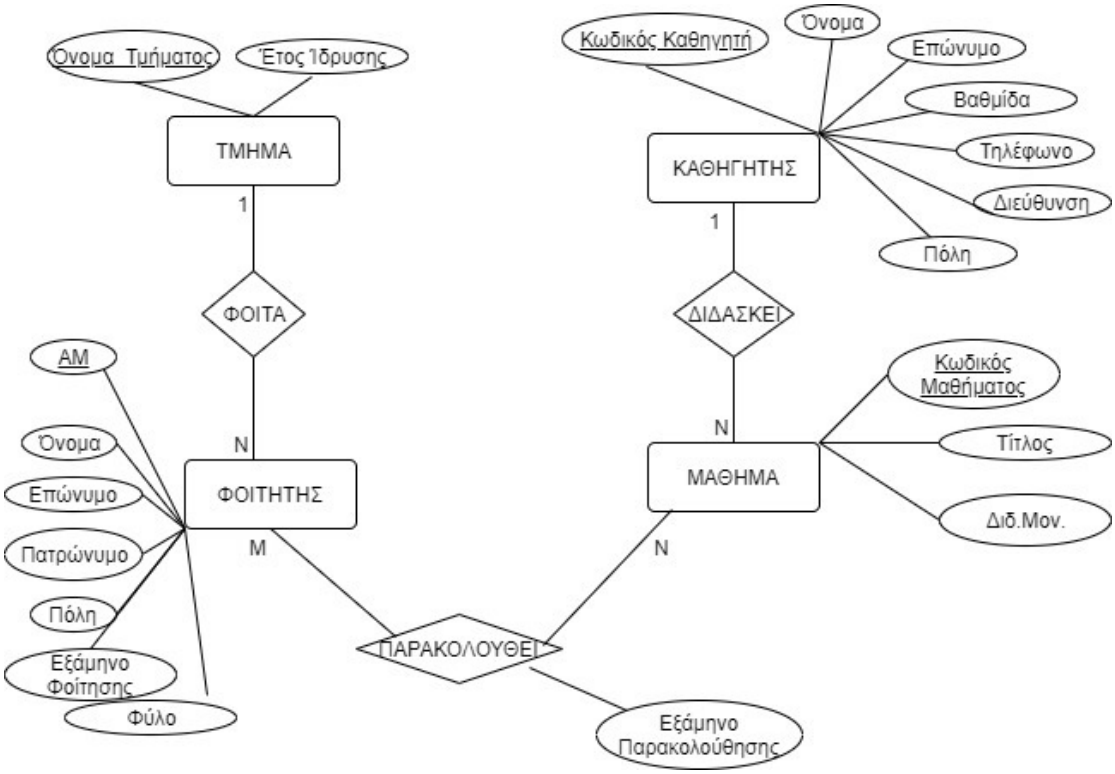
Ο λόγος που χρησιμοποιήθηκε η MongoDB Atlas είναι ότι είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων. Τα χαρακτηριστικά της MongoDB έχουν αναφέρει και πιο πριν. Επαναλαμβάνονται εδώ κάποια από τα πιο βασικά: Είναι μια ανοικτού τύπου cross-platform, document-oriented βάση δεδομένων. Αποθηκεύει δεδομένα χρησιμοποιώντας ένα ευέλικτο JSON-like μοντέλο. Τα πεδία μπορούν να διαφέρουν από έγγραφο σε έγγραφο και αυτό δίνει ευελιξία, αλλά και λόγω του δυναμικού της σχεδιασμού της επιτρέπει γρήγορες και εύκολες αλλαγές προσφέροντας υψηλή απόδοση. Προσφέρει auto-sharding και native replication, ευρετήρια, πλούσια γλώσσα ερωτημάτων (query language). Υποστηρίζει την οριζόντια κλιμάκωση, είναι κατάλληλη για εγγραφές με πολλά δεδομένα. Μπορεί να αναλύει τεράστιο όγκο δεδομένων, μπορεί να αναλύει με γεωχωρική (geospatial) κατηγοριοποίηση. Είναι κατάλληλη τόσο για μικρές εταιρείες όσο και για πολύ μεγάλες. Επίσης είναι εύκολη η αλλαγή ενός σχεσιακού συστήματος βάσης δεδομένων σε MongoDB.

Η βάση δεδομένων και στις δύο περιπτώσεις διαχειρίζεται δεδομένα ενός εκπαιδευτικού ιδρύματος. Μέσα από παραδείγματα έγινε μια προσπάθεια σύγκρισης των δύο βάσεων δεδομένων και καταγραφής των σημείων στα οποία υπερτερεί η καθεμία. Το παράδειγμα δημιουργήθηκε για να καλύψει τις ανάγκες της εργασίας και δεν είναι μια πλήρως αναπτυγμένη βάση δεδομένων.

Αρχικά εντοπίστηκαν οι οντότητες της βάσης που είναι οι εξής: Φοιτητής, Καθηγητής, Τμήμα, Μάθημα.

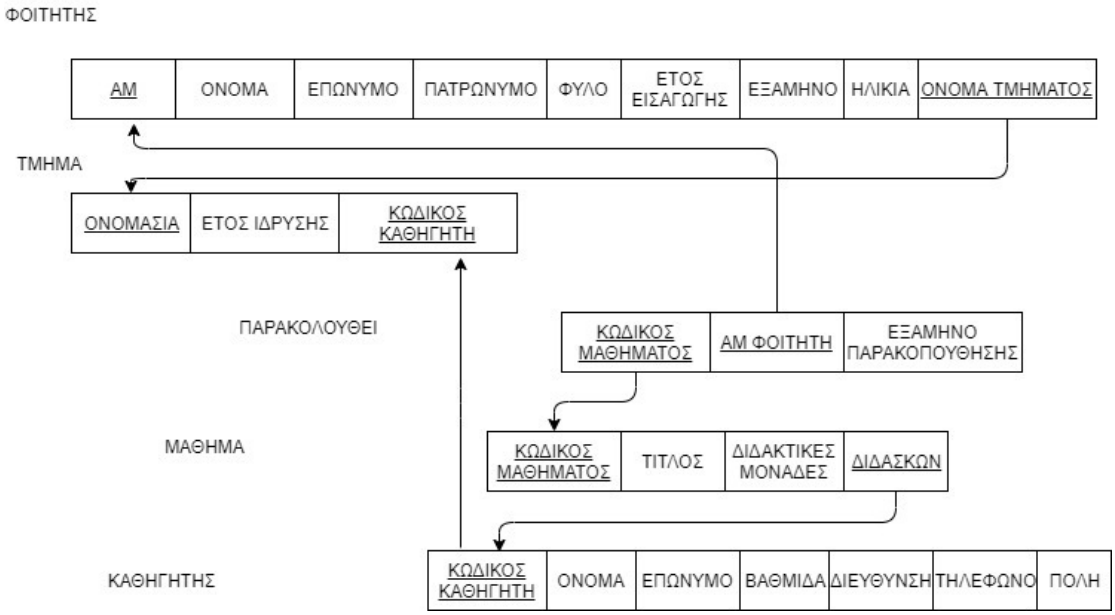
Κάθε οντότητα αποτελεί έναν πίνακα στην Access και μια Collection στην MongoDB Atlas.

Στη συνέχεια ορίστηκαν τα απαραίτητα γνωρίσματα που θα έχει η κάθε εγγραφή. Έτσι δημιουργήθηκε ένα διάγραμμα οντοτήτων-συσχετίσεων και από εκεί φάνηκε πως χρειάζεται ένας επιπλέον πίνακας-οντότητα τον οποίο ονομάστηκε Παρακολουθεί κι αυτό γιατί η σχέση Φοιτητής Μάθημα είναι Μ-Ν. Έτσι στην εικόνα 18 παρουσιάζεται το διάγραμμα Οντοτήτων – Συσχετίσεων για την βάση δεδομένων "Εκπαιδευτικό Ίδρυμα".



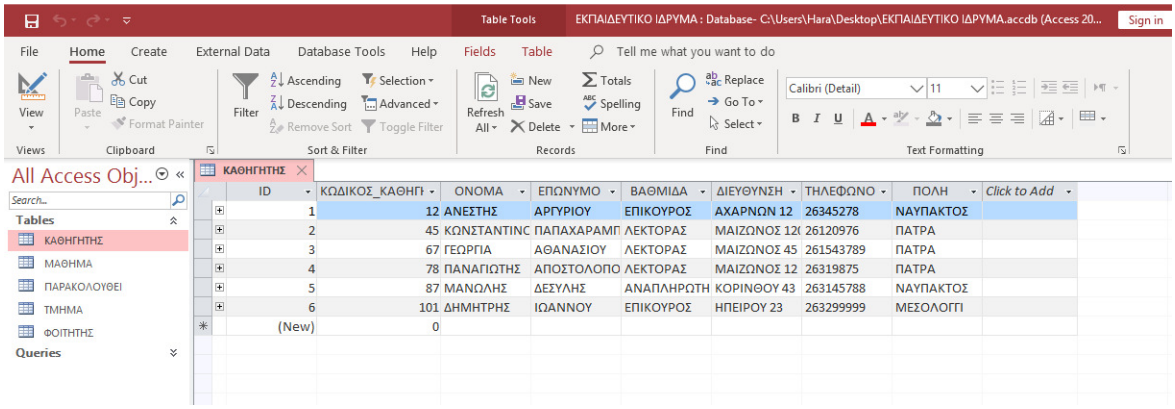
Εικόνα 18. Διάγραμμα Οντοτήτων – Συσχετίσεων για το 'Εκπαιδευτικό Ίδρυμα'

Στη συνέχεια δημιουργήθηκε το σχεσιακό μοντέλο όπως απεικονίζεται στην εικόνα 19.



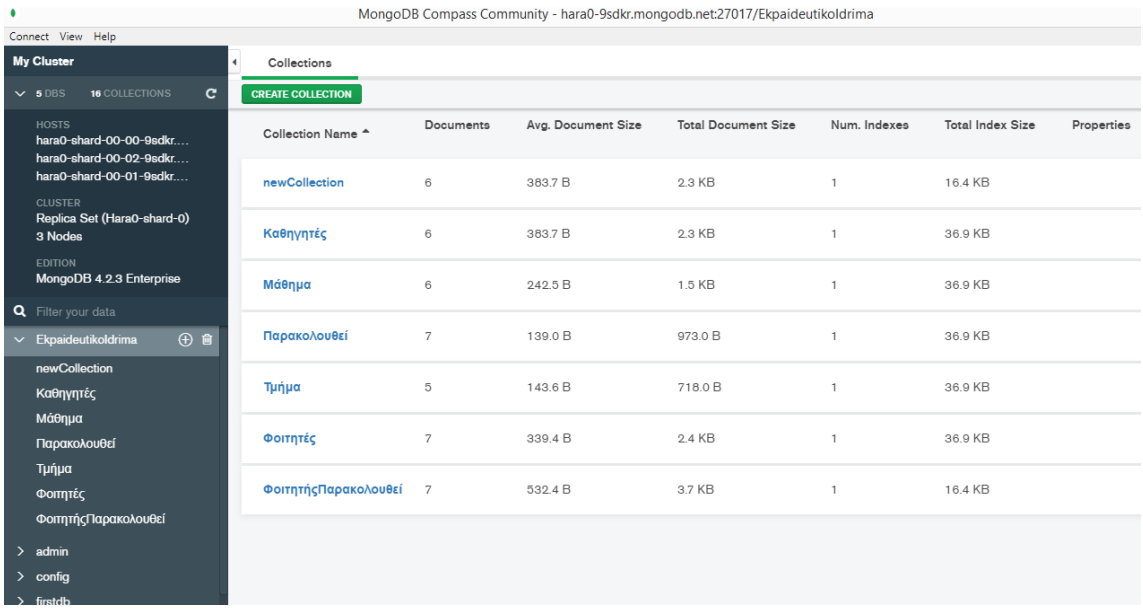
Εικόνα 19. Σχεσιακό Μοντέλο για το 'Εκπαιδευτικό Ίδρυμα'

Στην Access δημιουργήθηκαν οι πίνακες: Φοιτητής, Καθηγητής, Τμήμα, Μάθημα και Παρακολουθεί και εισήχθησαν στοιχεία σε κάθε πίνακα όπως φαίνεται και στην εικόνα 20.



Εικόνα 20. Οι πίνακες στην Access

Ο κάθε πίνακας της Access έγινε μια Collection στην MongoDB Atlas και δημιουργήθηκαν documents για κάθε Collection. Οι Collections που δημιουργήθηκαν στην MongoDB Atlas απεικονίζονται στην εικόνα 21.



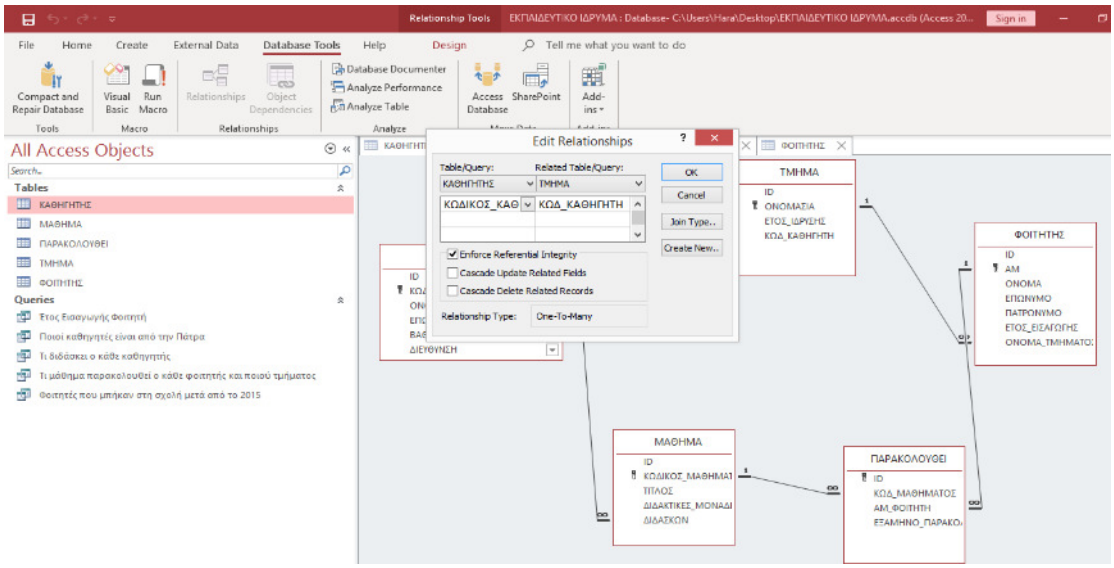
Εικόνα 21. Οι Collections στην MongoDB Atlas

3.1 Για την Access

Αρχικά σε κατάσταση σχεδίασης καθόρισα τα πεδία κάθε πίνακα και τον τύπο δεδομένων του κάθε πεδίου και έβαλα ευρετήριο στο πρωτεύον κλειδί.

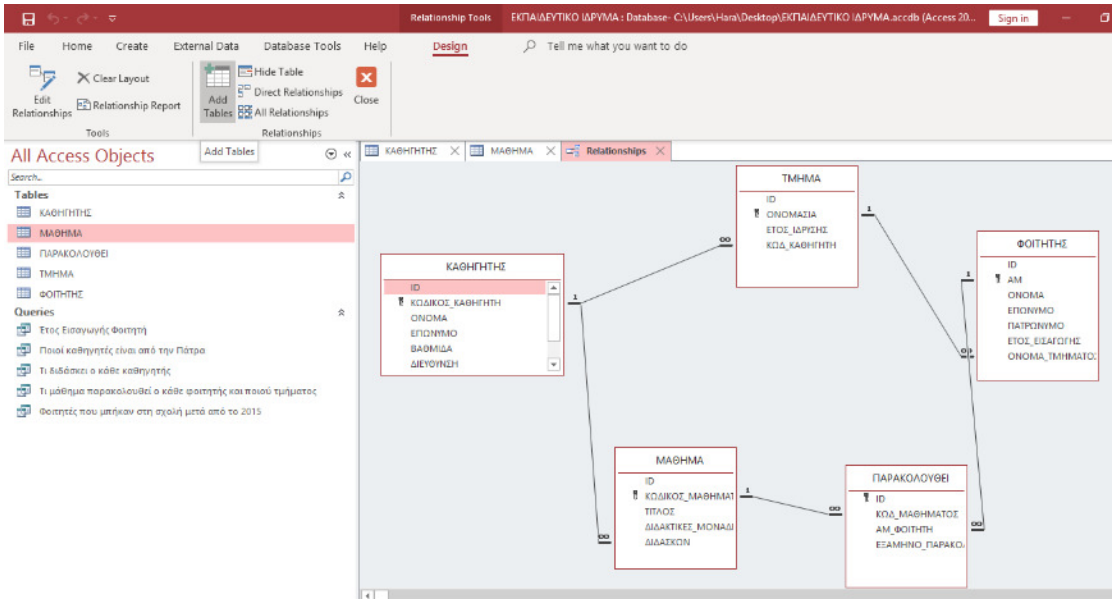
Στη συνέχεια, ανοίγοντας κάθε πίνακα εισήγαγα τα στοιχεία του.

Επόμενο βήμα ήταν οι συσχετίσεις των πινάκων μεταξύ τους η υλοποίηση των οποίων παρουσιάζεται στην εικόνα 22.



Εικόνα 22.Υλοποίηση συσχετίσεων των πινάκων

Και τελικά οι συσχετίσεις είναι όπως φαίνονται στην εικόνα 23:



Εικόνα 23. Συσχετίσεις των πινάκων στην Access

3.1.1 Διατύπωση Ερωτημάτων (Queries)

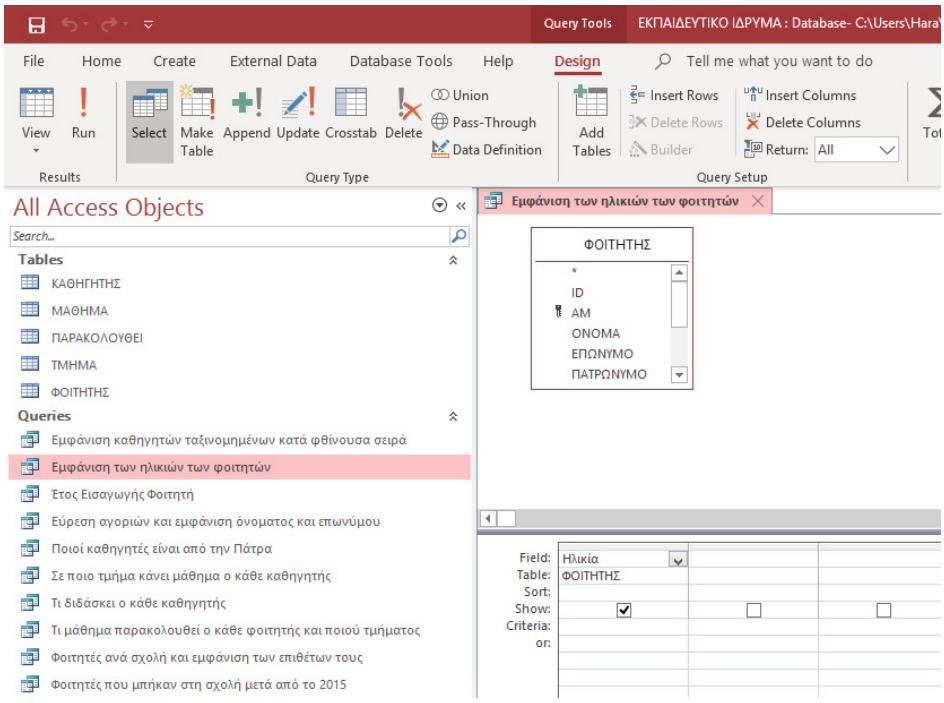
Μέσω της λειτουργίας Δημιουργία -> Ερωτήματος, δημιουργήθηκαν τα εξής ερωτήματα:

3.1.1.1 Εύρεση των ηλικιών των φοιτητών που υπάρχουν στη βάση δεδομένων

Σε SQL η εντολή είναι:

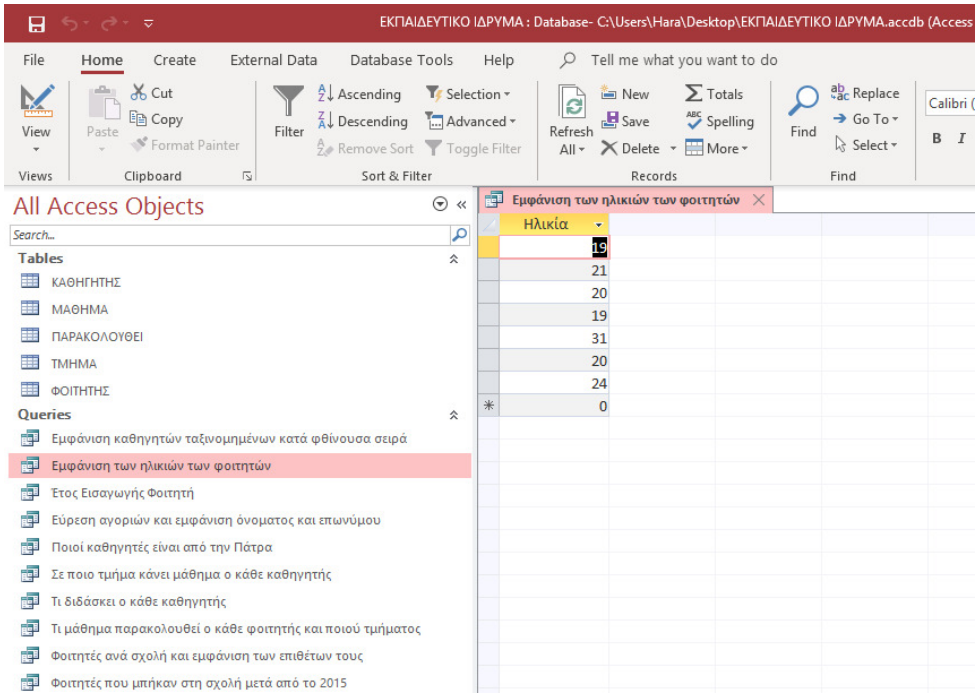
```
SELECT ΦΟΙΤΗΤΗΣ.Ηλικία
FROM ΦΟΙΤΗΤΗΣ;
```

Στην εικόνα 24 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης της ηλικίας των φοιτητών.



Εικόνα 24. Σχεδίαση ερωτήματος εύρεσης ηλικίας φοιτητών στην Access

Τα αποτελέσματα του ερωτήματος παρουσιάζονται στην εικόνα 25.



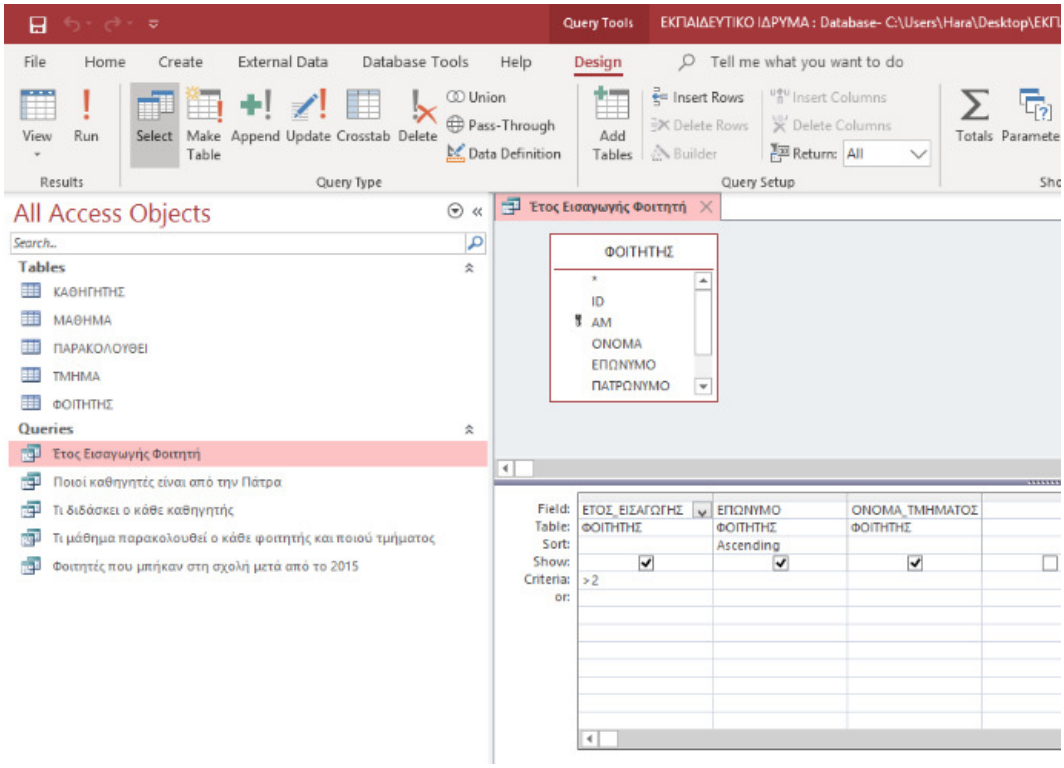
Εικόνα 25.Αποτέλεσμα ερωτήματος εύρεσης ηλικίας φοιτητών στην Access

3.1.1.2 Εμφάνιση του έτους εισαγωγής κάθε φοιτητή. Η εμφάνιση να είναι κατά αλφαβητική σειρά του επωνύμου

Σε SQL η εντολή είναι:

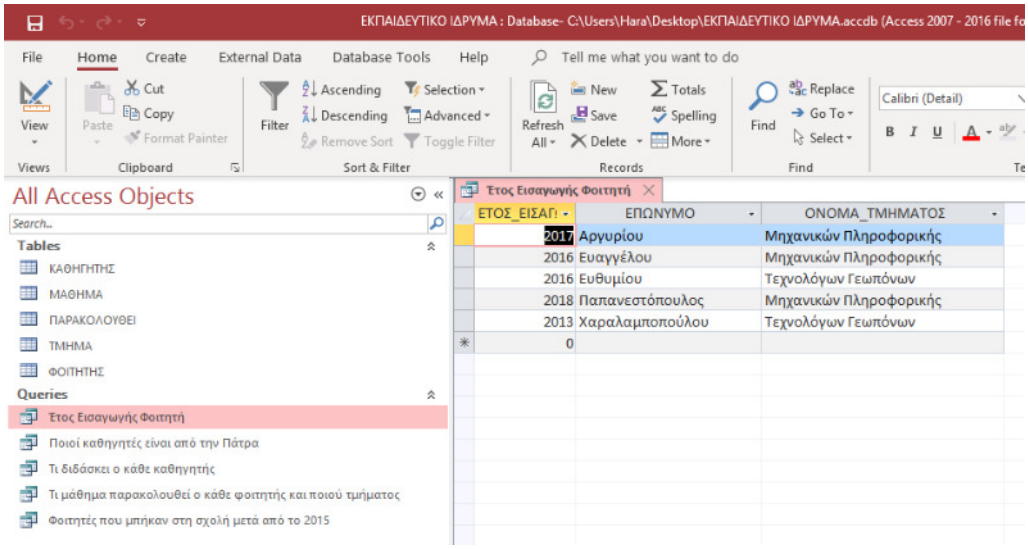
```
SELECT ΦΟΙΤΗΤΗΣ.ΕΤΟΣ_ΕΙΣΑΓΩΓΗΣ, ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ,
ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ_ΤΜΗΜΑΤΟΣ
FROM ΦΟΙΤΗΤΗΣ
WHERE (((ΦΟΙΤΗΤΗΣ.ΕΤΟΣ_ΕΙΣΑΓΩΓΗΣ)>2))
ORDER BY ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ;
```

Στην εικόνα 26 απεικονίζεται η σχεδίαση του ερωτήματος εμφάνισης του έτους εισαγωγής κάθε φοιτητή, κατά αλφαβητική σειρά με βάση το επώνυμο:



Εικόνα 26. Σχεδίαση ερωτήματος εμφάνισης κατά αλφαβητική σειρά των φοιτητών και του έτους εισαγωγής τους στην Access

Τα αποτελέσματα του ερωτήματος φαίνονται στην εικόνα 27.



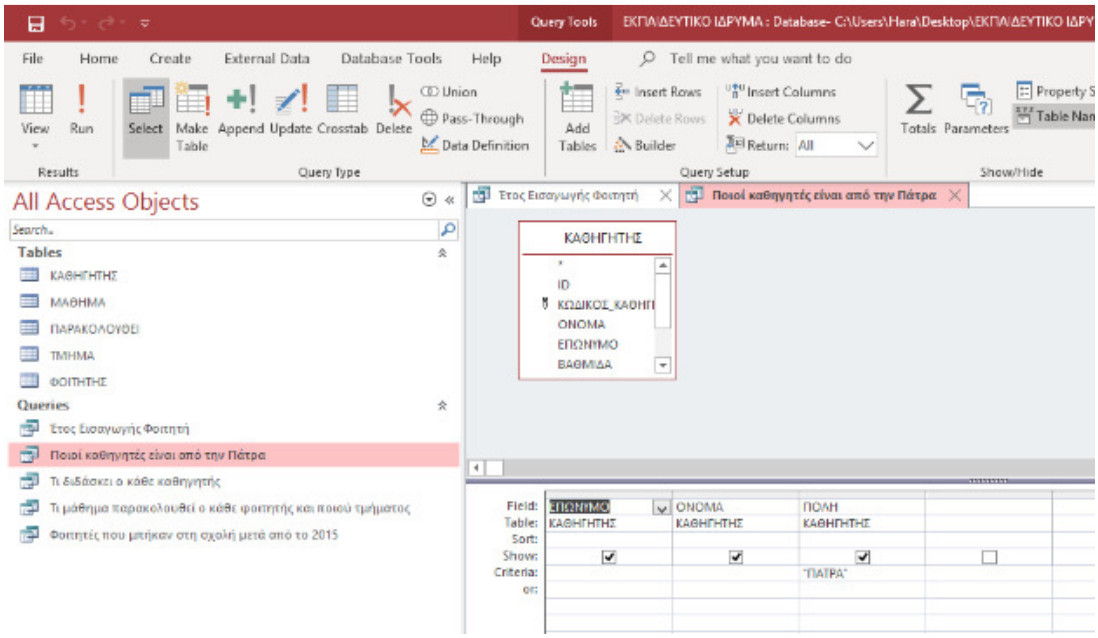
Εικόνα 27. Αποτέλεσμα ερωτήματος εμφάνισης κατά αλφαβητική σειρά των φοιτητών και το έτος εισαγωγής τους στην Access

3.1.1.3 Ποιοι καθηγητές είναι από την Πάτρα

Σε εντολή SQL:

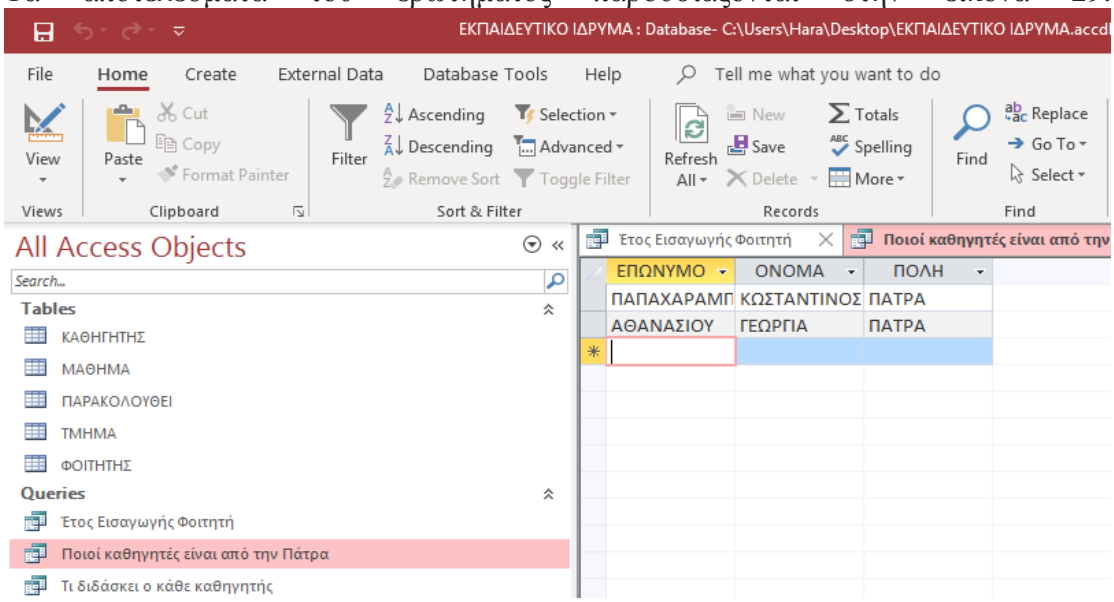
```
SELECT ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ, ΚΑΘΗΓΗΤΗΣ.ΟΝΟΜΑ,
ΚΑΘΗΓΗΤΗΣ.ΠΟΛΗ
FROM ΚΑΘΗΓΗΤΗΣ
WHERE (((ΚΑΘΗΓΗΤΗΣ.ΠΟΛΗ)="ΠΑΤΡΑ"));
```

Στην εικόνα 28 παρουσιάζεται η σχεδίαση του ερωτήματος εύρεσης των καθηγητών που είναι από την Πάτρα:



Εικόνα 28. Σχεδίαση ερωτήματος εμφάνισης των καθηγητών που είναι από την Πάτρα στην Access

Τα αποτελέσματα του ερωτήματος παρουσιάζονται στην εικόνα 29.



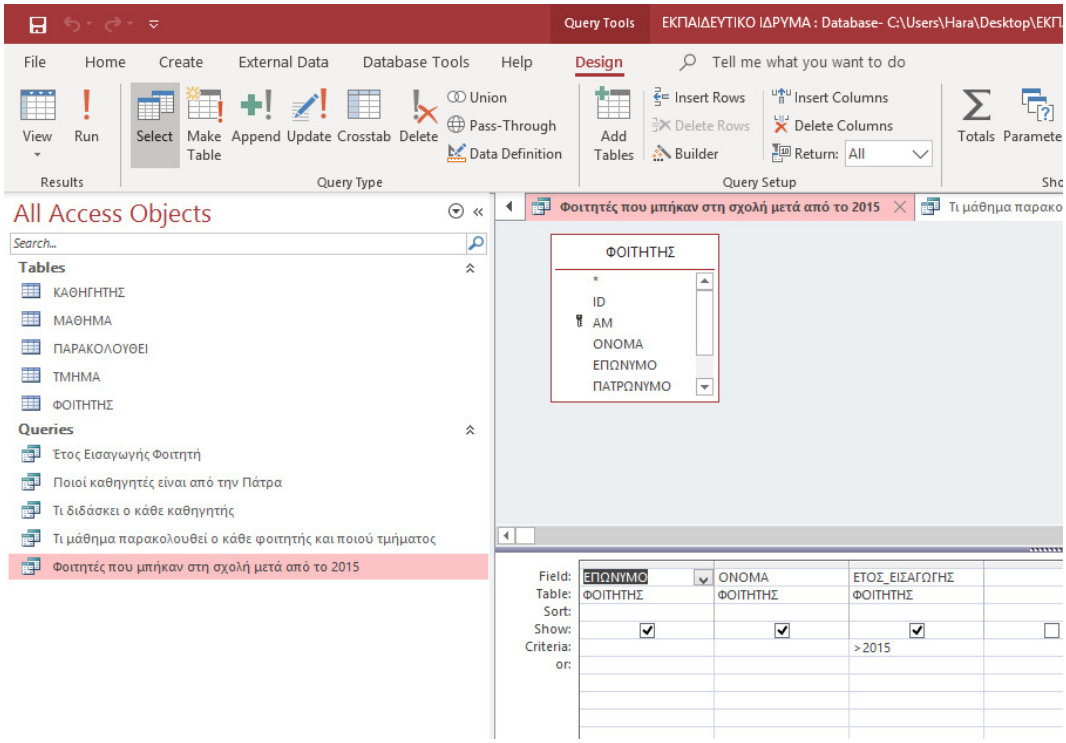
Εικόνα 29. Αποτέλεσμα ερωτήματος εμφάνισης των καθηγητών που είναι από την Πάτρα στην Access

3.1.1.4 Ποιοι φοιτητές μπήκαν στη σχολή μετά το 2015

Σε εντολή SQL:

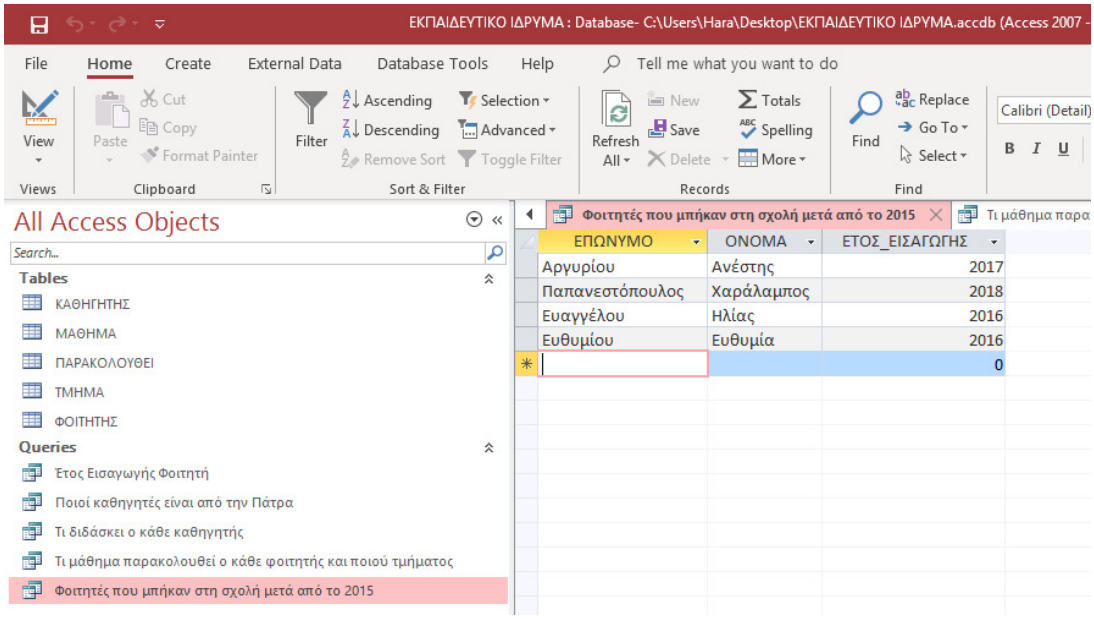
```
SELECT ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ, ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ,
ΦΟΙΤΗΤΗΣ.ΕΤΟΣ_ΕΙΣΑΓΩΓΗΣ
FROM ΦΟΙΤΗΤΗΣ
WHERE (((ΦΟΙΤΗΤΗΣ.ΕΤΟΣ_ΕΙΣΑΓΩΓΗΣ)>2015));
```

Στην εικόνα 30 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των φοιτητών που μπήκαν στη σχολή μετά το 2015.



Εικόνα 30. Σχεδίαση ερωτήματος εύρεσης των φοιτητών που μπήκαν στη σχολή μετά το 2015 στην Access

Τα αποτελέσματα του ερωτήματος παρουσιάζονται στην εικόνα 31.



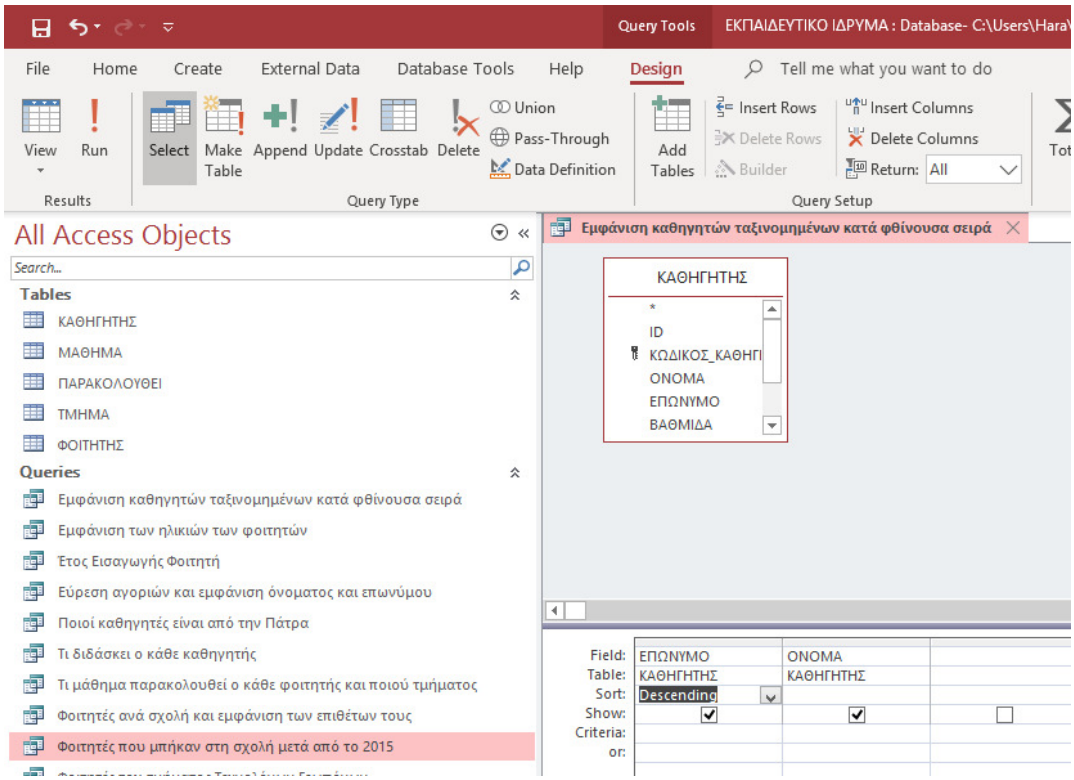
Εικόνα 31. Αποτέλεσμα ερωτήματος εύρεσης των φοιτητών που μπήκαν στη σχολή μετά το 2015 στην Access

3.1.1.5 Εμφάνιση ονόματος και επωνύμου των καθηγητών, ταξινομημένων κατά φθίνουσα σειρά

Σε εντολή SQL:

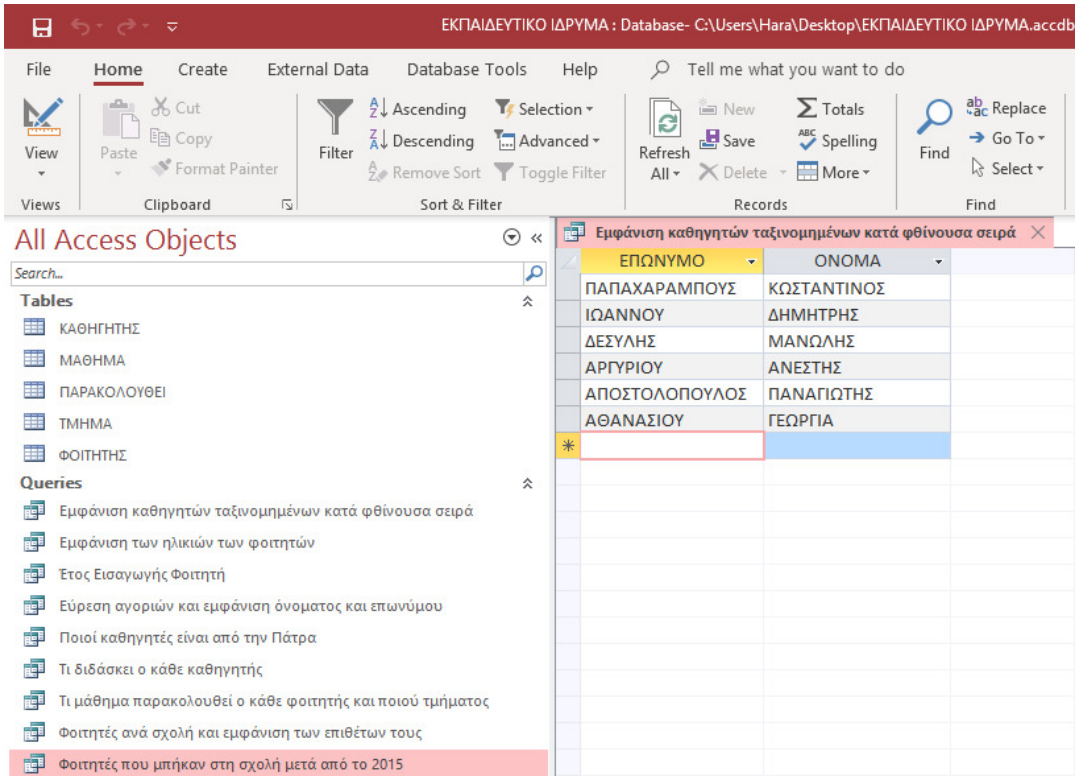
```
SELECT ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ, ΚΑΘΗΓΗΤΗΣ.ΟΝΟΜΑ
FROM ΚΑΘΗΓΗΤΗΣ
ORDER BY ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ DESC;
```

Στην εικόνα 32 απεικονίζεται η σχεδίαση του ερωτήματος εμφάνισης του ονόματος και του επωνύμου των καθηγητών, ταξινομημένων κατά φθίνουσα σειρά σύμφωνα με το επώνυμό τους.



Εικόνα 32. Σχεδίαση ερωτήματος εμφάνισης καθηγητών ταξινομημένων σε φθίνουσα σειρά στην Access

Τα αποτελέσματα εμφανίζονται στην εικόνα 33.



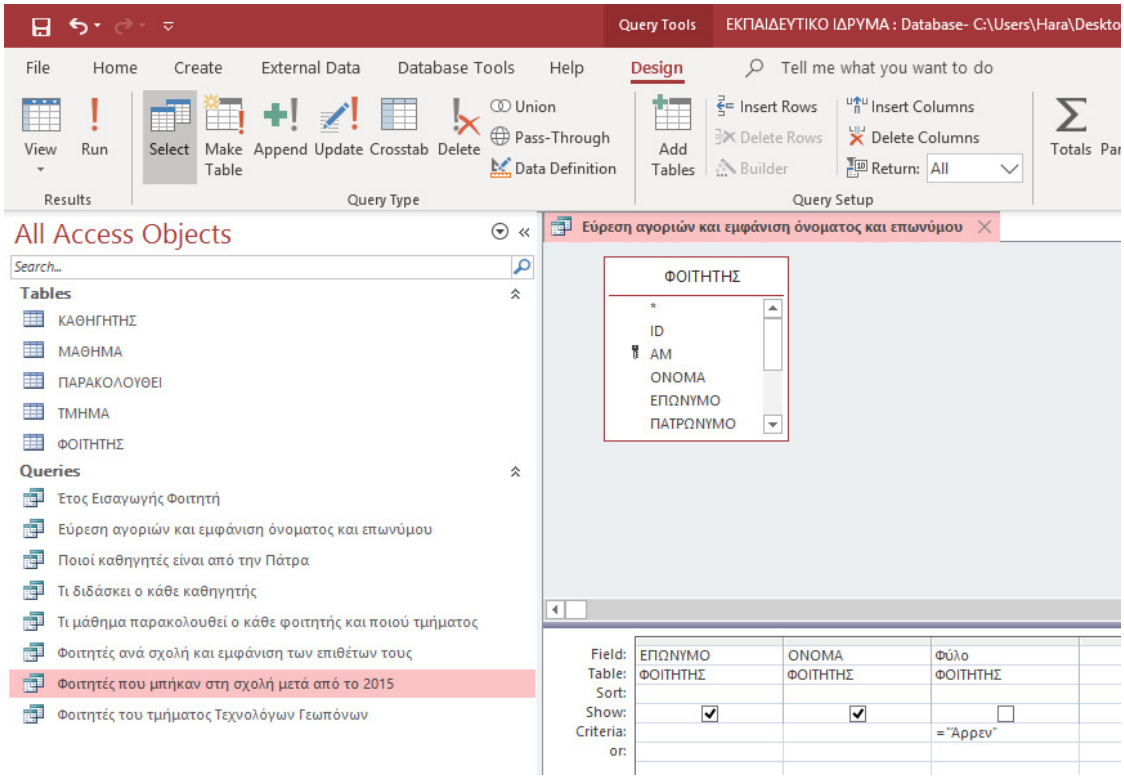
Εικόνα 33. Αποτέλεσμα ερωτήματος εμφάνισης καθηγητών ταξινομημένων σε φθίνουσα σειρά στην Access

3.1.1.6 Εύρεση των αγοριών και εμφάνιση του ονόματος και του επωνύμου, αλλά και του πλήθους τους

Σε εντολή SQL:

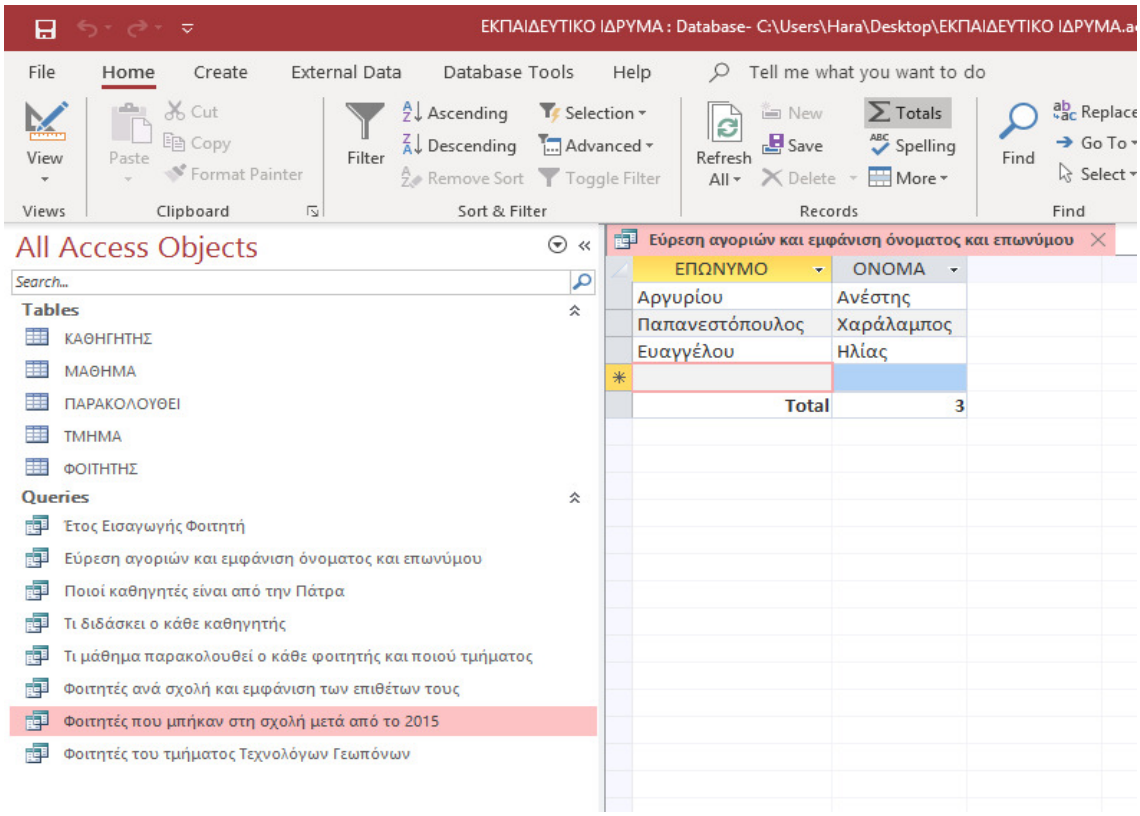
```
SELECT ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ, ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ
FROM ΦΟΙΤΗΤΗΣ
WHERE (((ΦΟΙΤΗΤΗΣ.Φύλο)="Άρρεν"));
```

Στην εικόνα 34 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των αγοριών και η εμφάνιση του ονόματος, του επωνύμου και του πλήθους τους.



Εικόνα 34. Σχεδίαση ερωτήματος εμφάνισης του ονόματος και του επωνύμου των αγοριών στην Access

Τα αποτελέσματα παρουσιάζονται στην εικόνα 35.



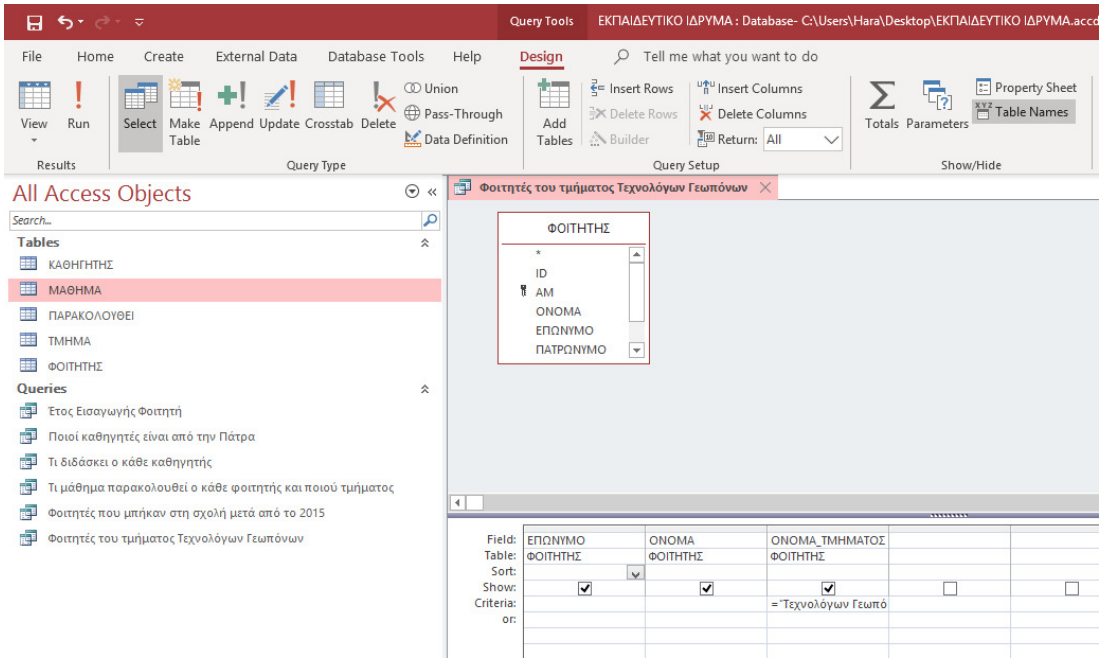
Εικόνα 35. Αποτέλεσμα ερωτήματος εμφάνισης του ονόματος και του επωνύμου των αγοριών στην Access

3.1.1.7 Εύρεση φοιτητών του τμήματος Τεχνολόγων Γεωπόνων

Σε εντολή SQL:

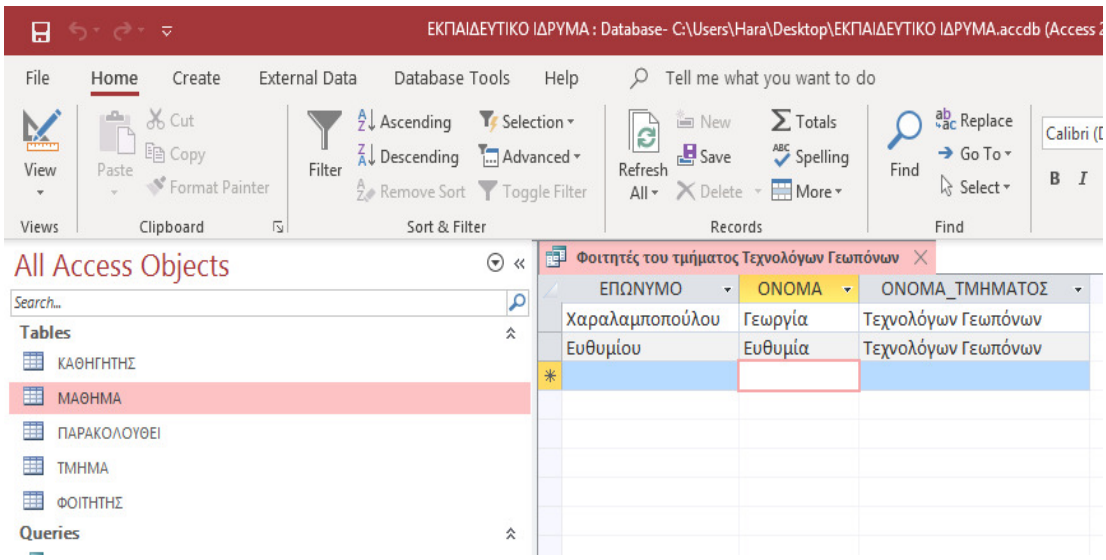
```
SELECT ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ, ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ,
ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ_ΤΜΗΜΑΤΟΣ
FROM ΦΟΙΤΗΤΗΣ
WHERE (((ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ_ΤΜΗΜΑΤΟΣ)="Τεχνολόγων Γεωπόνων"));
```

Στην εικόνα 36 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των φοιτητών του τμήματος Τεχνολόγων Γεωπόνων.



Εικόνα 36. Σχεδίαση ερωτήματος εμφάνισης των φοιτητών του τμήματος Τεχνολόγων Γεωπόνων στην Access

Τα αποτελέσματα παρουσιάζονται στην εικόνα 37.



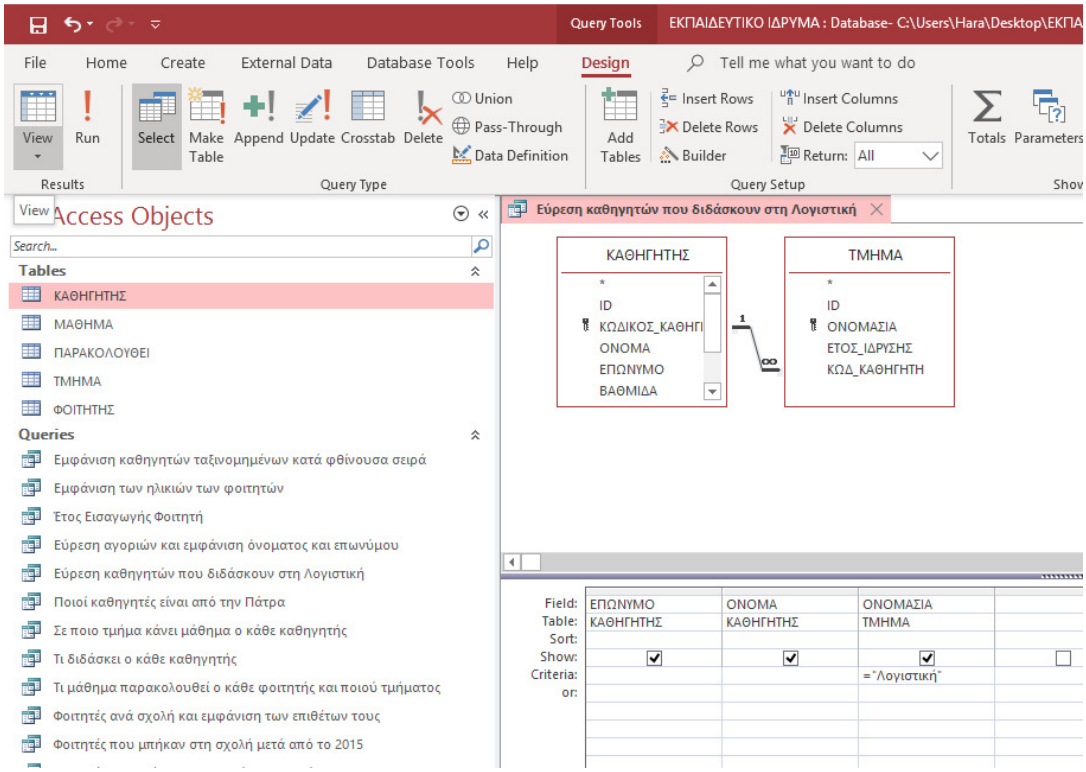
Εικόνα 37. Αποτέλεσμα ερωτήματος εμφάνισης των φοιτητών του τμήματος Τεχνολόγων Γεωπόνων στην Access

3.1.1.8 Εύρεση καθηγητών που διδάσκουν στη Λογιστική

Σε SQL η εντολή είναι:

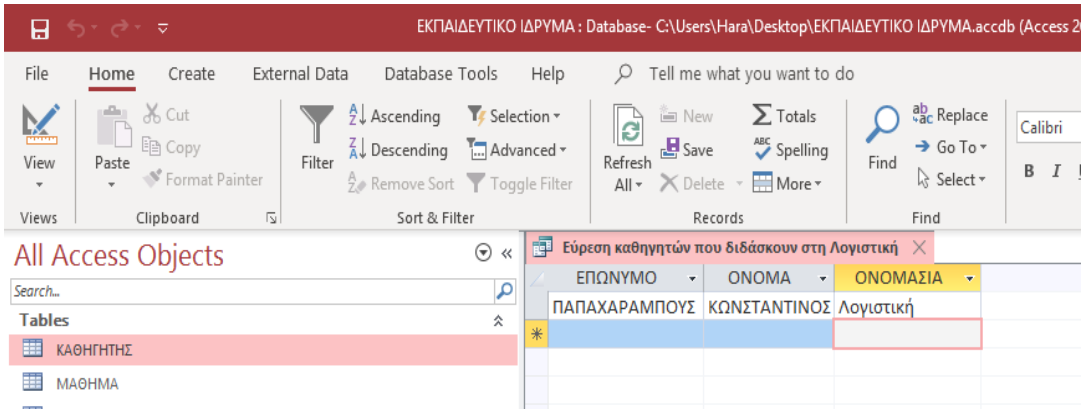
```
SELECT ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ, ΚΑΘΗΓΗΤΗΣ.ΟΝΟΜΑ,
ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ
FROM ΚΑΘΗΓΗΤΗΣ INNER JOIN ΤΜΗΜΑ ON
ΚΑΘΗΓΗΤΗΣ.ΚΩΔΙΚΟΣ_ΚΑΘΗΓΗΤΗ = ΤΜΗΜΑ.ΚΩΔ_ΚΑΘΗΓΗΤΗ
WHERE (((ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ)="Λογιστική"));
```

Στην εικόνα 38 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των καθηγητών που διδάσκουν στη Λογιστική



Εικόνα 38. Σχεδίαση ερωτήματος εύρεσης καθηγητών που διδάσκουν στη Λογιστική στην Access

Το αποτέλεσμα εμφανίζεται στην εικόνα 39.



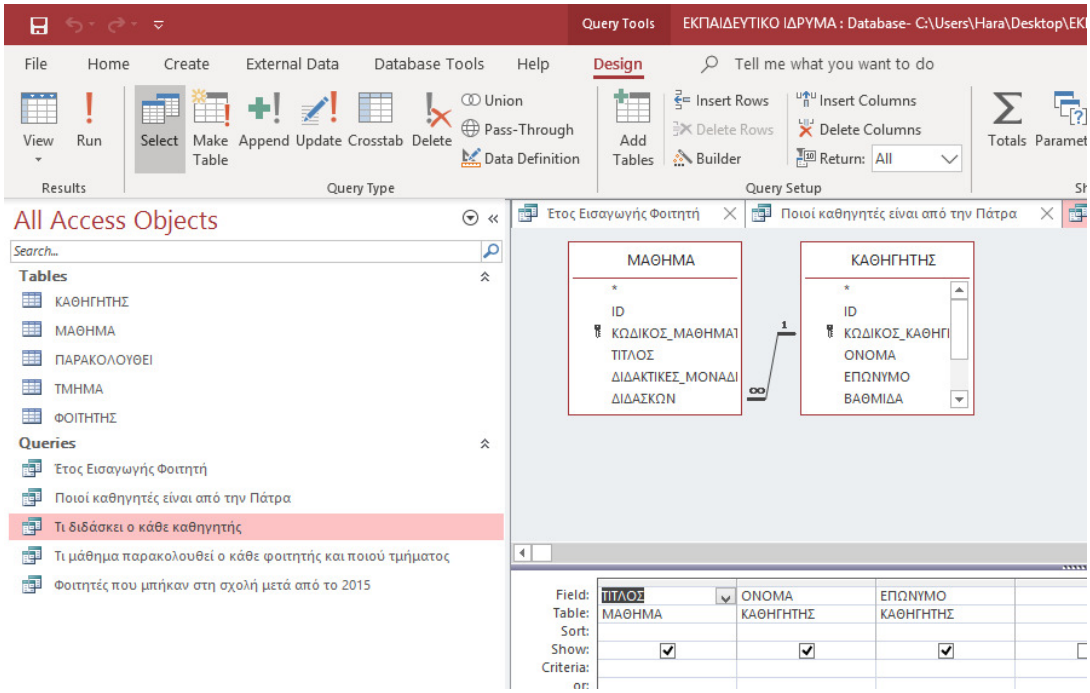
Εικόνα 39. Αποτέλεσμα ερωτήματος εύρεσης καθηγητών που διδάσκουν στη Λογιστική στην Access

3.1.1.9 Τι μάθημα διδάσκει ο κάθε καθηγητής

Σε εντολή SQL:

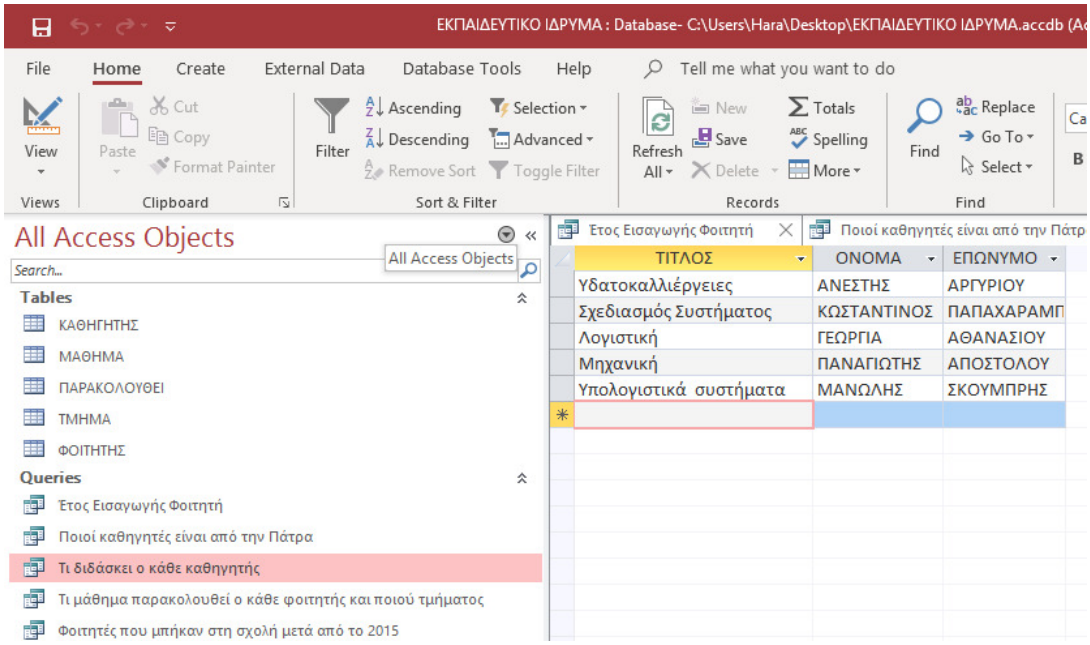
```
SELECT ΜΑΘΗΜΑ.ΤΙΤΛΟΣ, ΚΑΘΗΓΗΤΗΣ.ΟΝΟΜΑ,
ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ
FROM ΚΑΘΗΓΗΤΗΣ INNER JOIN ΜΑΘΗΜΑ ON
ΚΑΘΗΓΗΤΗΣ.ΚΩΔΙΚΟΣ_ΚΑΘΗΓΗΤΗ = ΜΑΘΗΜΑ.ΔΙΔΑΣΚΩΝ;
```

Στην εικόνα 40 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης του μαθήματος που διδάσκει ο κάθε καθηγητής.



Εικόνα 40. Σχεδίαση ερωτήματος εύρεσης των μαθημάτων που διδάσκουν οι καθηγητές στην Access

Και τα αποτελέσματα παρουσιάζονται στην εικόνα 41.



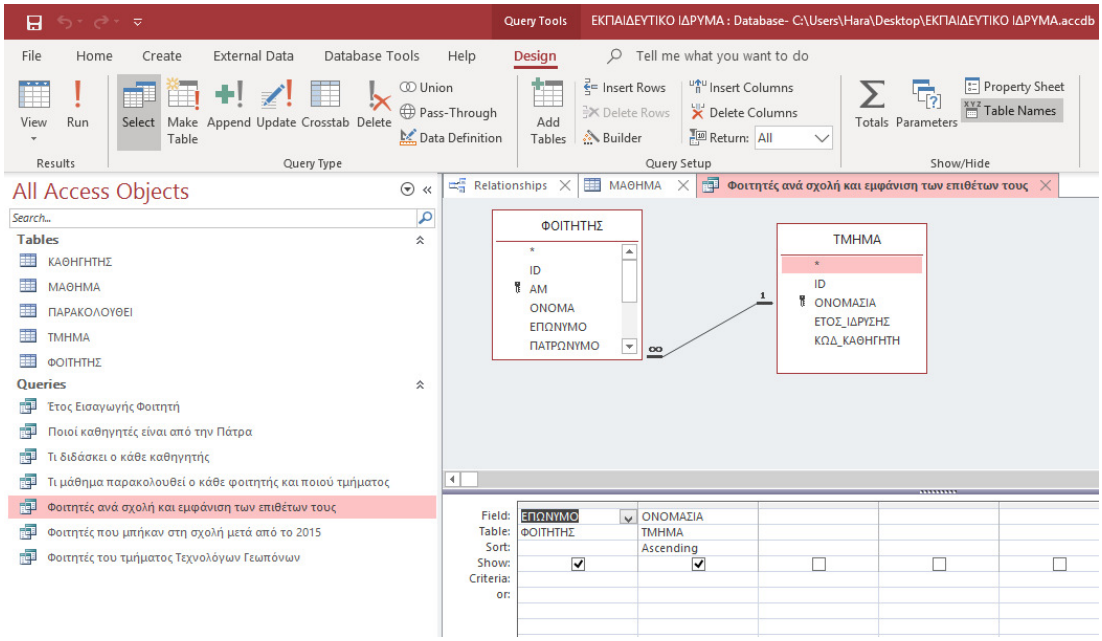
Εικόνα 41. Αποτέλεσμα ερωτήματος εύρεσης των μαθημάτων που διδάσκουν οι καθηγητές στην Access

3.1.1.10 Εύρεση φοιτητών ανά σχολή και εμφάνιση των επωνύμων τους

Σε εντολή SQL:

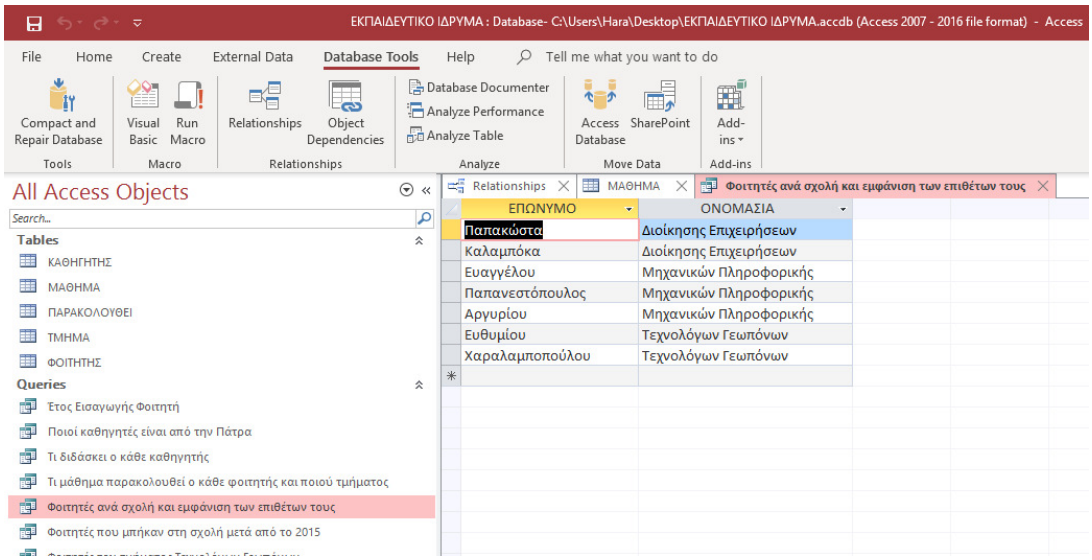
```
SELECT ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ, ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ
FROM ΤΜΗΜΑ INNER JOIN ΦΟΙΤΗΤΗΣ ON ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ = ΦΟΙΤΗΤΗΣ.ΟΝΟΜΑ_ΤΜΗΜΑΤΟΣ
ORDER BY ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ;
```

Στην εικόνα 42 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των φοιτητών ανά σχολή και εμφάνιση των επιθέτων τους:



Εικόνα 42. Σχεδίαση ερωτήματος εύρεσης των φοιτητών ανά σχολή στην Access

Τα αποτελέσματα του ερωτήματος εμφανίζονται στην εικόνα 43:



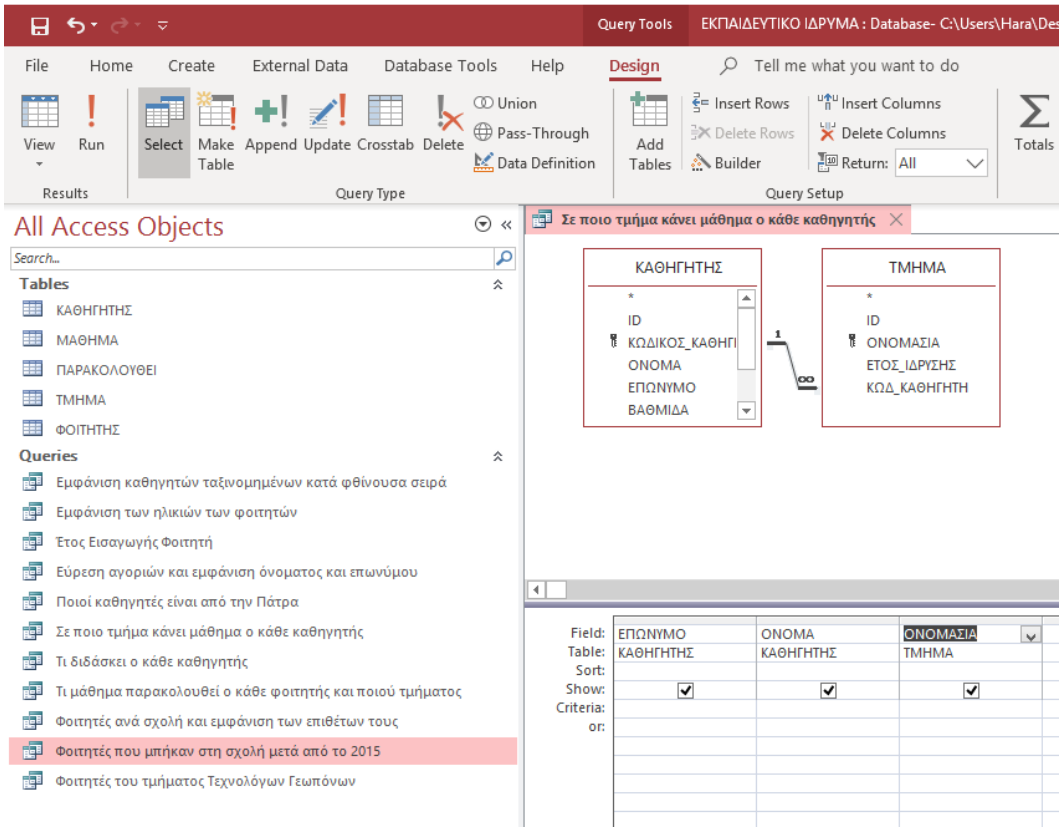
Εικόνα 43. Αποτέλεσμα ερωτήματος εύρεσης των φοιτητών ανά σχολή στην Access

3.1.1.11 Εύρεση καθηγητών που διδάσκουν σε κάθε τμήμα

Σε SQL

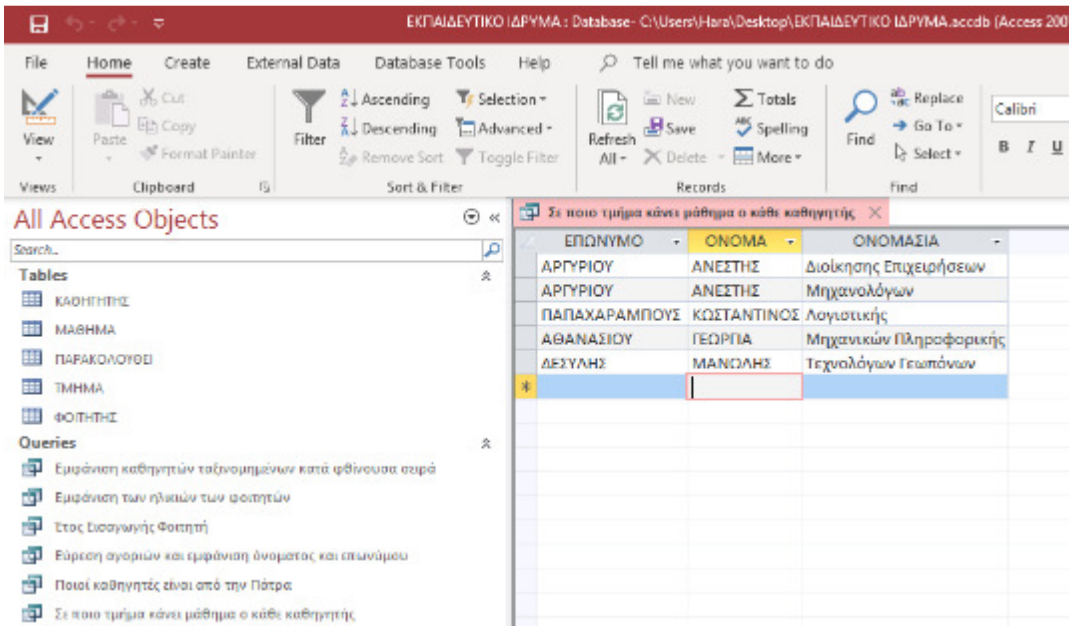
```
SELECT ΚΑΘΗΓΗΤΗΣ.ΕΠΩΝΥΜΟ, ΚΑΘΗΓΗΤΗΣ.ΟΝΟΜΑ, ΤΜΗΜΑ.ΟΝΟΜΑΣΙΑ
FROM ΚΑΘΗΓΗΤΗΣ INNER JOIN ΤΜΗΜΑ ON ΚΑΘΗΓΗΤΗΣ.ΚΩΔΙΚΟΣ_ΚΑΘΗΓΗΤΗ =
ΤΜΗΜΑ.ΚΩΔ_ΚΑΘΗΓΗΤΗ;
```

Στην εικόνα 44 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των καθηγητών που διδάσκουν σε κάθε τμήμα.



Εικόνα 44. Σχεδίαση ερωτήματος εύρεσης των καθηγητών που διδάσκουν σε κάθε τμήμα στην Access

Τα αποτελέσματα του ερωτήματος φαίνονται στην εικόνα 45.



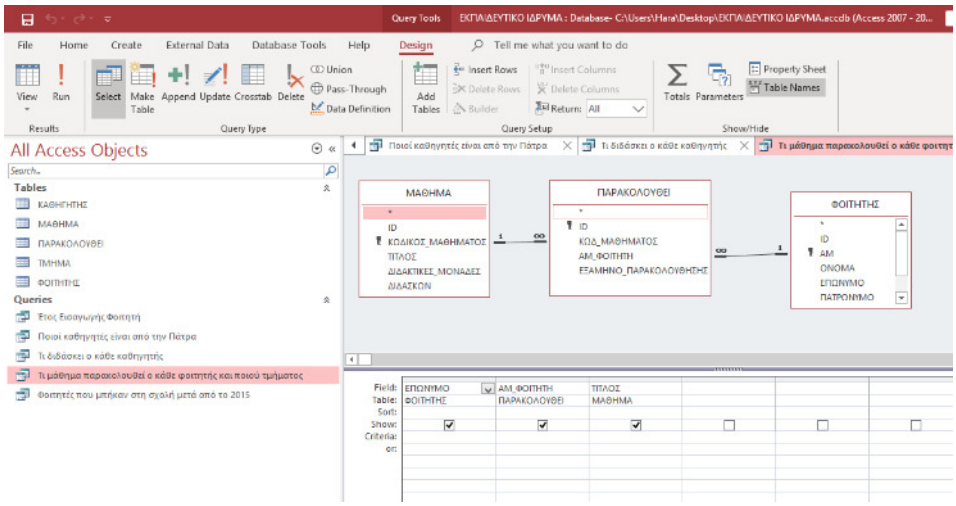
Εικόνα 45. Αποτέλεσμα ερωτήματος εύρεσης των καθηγητών που διδάσκουν σε κάθε τμήμα στην Access

3.1.1.12 Εύρεση μαθημάτων που παρακολουθεί ο κάθε φοιτητής και σε ποιο τμήμα ανήκει

Σε εντολή SQL:

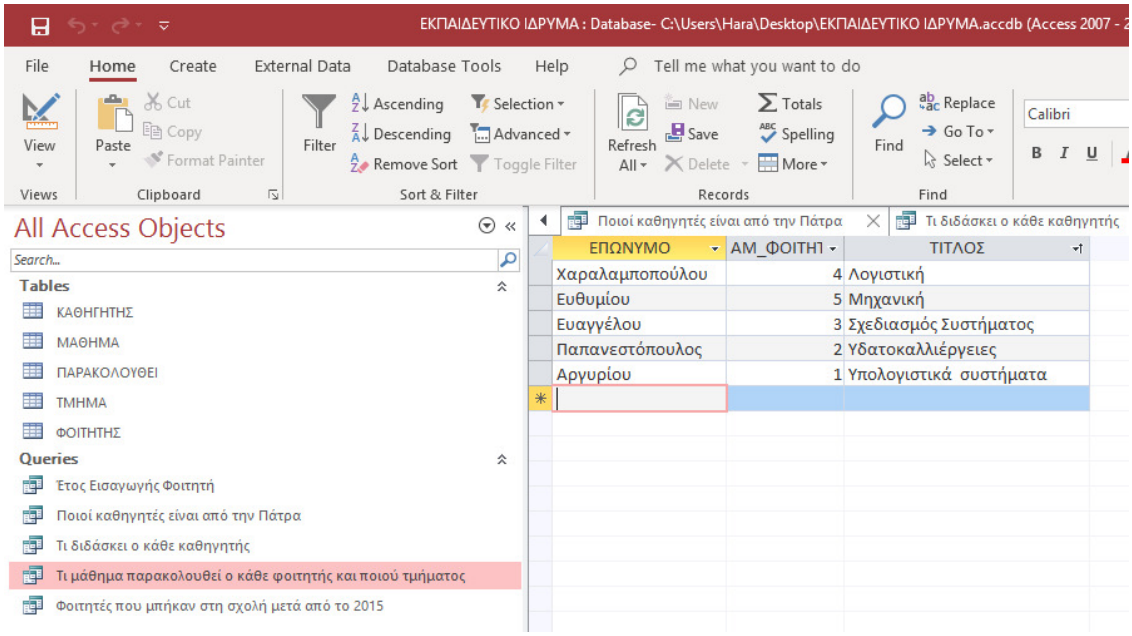
SELECT ΦΟΙΤΗΤΗΣ.ΕΠΩΝΥΜΟ, ΠΑΡΑΚΟΛΟΥΘΕΙ.ΑΜ_ΦΟΙΤΗΤΗ, ΜΑΘΗΜΑ.ΤΙΤΛΟΣ
FROM ΦΟΙΤΗΤΗΣ INNER JOIN (ΜΑΘΗΜΑ INNER JOIN ΠΑΡΑΚΟΛΟΥΘΕΙ ON ΜΑΘΗΜΑ.ΚΩΔΙΚΟΣ_ΜΑΘΗΜΑΤΟΣ = ΠΑΡΑΚΟΛΟΥΘΕΙ.ΚΩΔ_ΜΑΘΗΜΑΤΟΣ) ON ΦΟΙΤΗΤΗΣ.ΑΜ = ΠΑΡΑΚΟΛΟΥΘΕΙ.ΑΜ_ΦΟΙΤΗΤΗ;

Στην εικόνα 46 απεικονίζεται η σχεδίαση του ερωτήματος εύρεσης των μαθημάτων που παρακολουθεί ο κάθε φοιτητής και σε ποιο τμήμα ανήκει.



Εικόνα 46. Σχεδίαση ερωτήματος εύρεσης μαθημάτων που παρακολουθεί ο κάθε φοιτητής και σε ποια σχολή είναι στην Access

Τα αποτελέσματα του ερωτήματος παρουσιάζονται στην εικόνα 47.



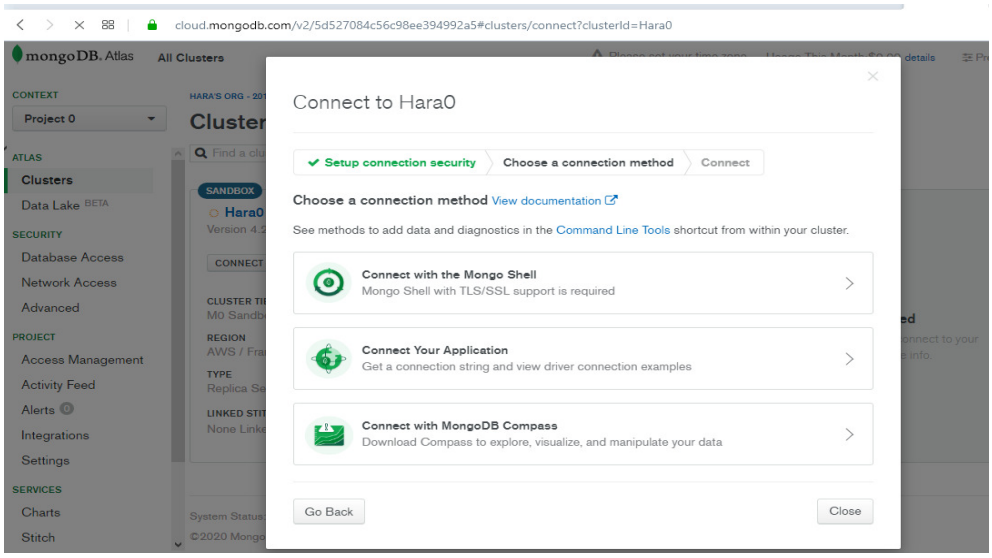
Εικόνα 47. Αποτέλεσμα ερωτήματος εύρεσης μαθημάτων που παρακολουθεί ο κάθε φοιτητής και σε ποια σχολή είναι στην Access

3.2 Για την MongoDB Atlas

Δημιουργήθηκε μια βάση δεδομένων αντίστοιχη της βάσης δεδομένων που κατασκευάστηκε στην Access και την ονομάστηκε EkpaideutikoIdrima. Κάθε πίνακας - οντότητα έγινε μια Collection.

3.2.1 Βασικές διεργασίες στην MongoDB Atlas

Η σύνδεση στην MongoDB Atlas γίνεται με τρεις τρόπους: Μέσω του Mongo Shell από το Command Prompt, μέσω του γραφικού περιβάλλοντος του Compass και μέσω μιας εφαρμογής (application) όπως φαίνεται και στην εικόνα 48.



Εικόνα 48.Τρόποι Σύνδεσης στην MongoDB Atlas

Τόσο για το Shell όσο και για το Compass, θα πρέπει αρχικά να γίνει εγκατάσταση και σύνδεση τους με το cluster του κάθε χρήστη και με χρήση των κωδικών που έχουν οριστεί στην εγκατάσταση της MongoDB Atlas.

Στο περιβάλλον του Mongo Shell, υπάρχει δυνατότητα προβολής των βάσεων δεδομένων που έχουν δημιουργηθεί με την εντολή: show dbs;

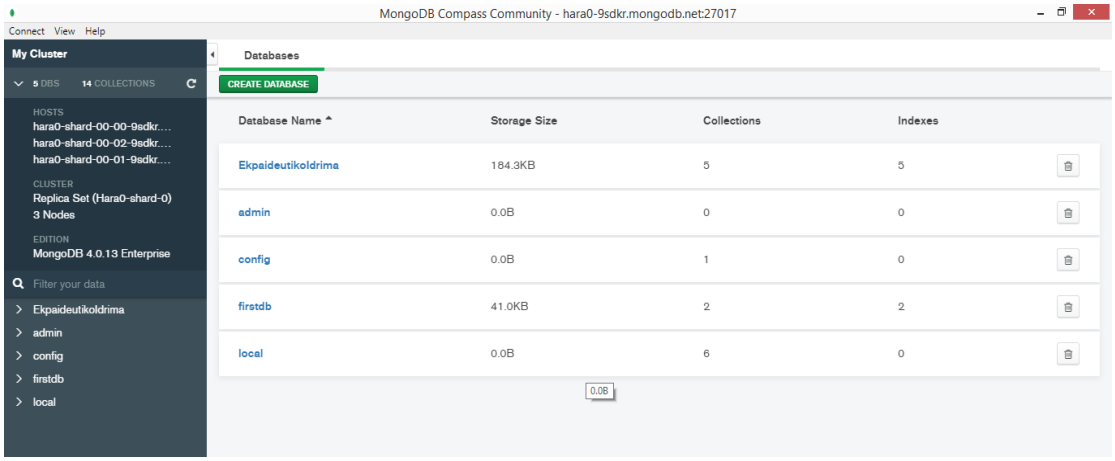
Στην αρχή, μιας και δεν έχουμε φτιάξει καμία βάση δεδομένων, η απάντηση θα είναι πως υπάρχουν οι βάσεις admin και local.

3.2.1.1 Δημιουργία βάσης δεδομένων στην MongoDB Atlas

Για την δημιουργία βάσης δεδομένων στο Mongo Shell υπάρχει η δυνατότητα χρήσης της **εντολής use** η οποία έχει διπλή λειτουργία:

- Αν υπάρχει ήδη η βάση δεδομένων, τότε αλλάζει βάση δεδομένων και μπαίνει στην βάση δεδομένων που της έχει ζητηθεί.
- Αν η βάση δεν υπάρχει, τότε και την δημιουργεί αλλά και αλλάζει βάση δεδομένων κατευθείαν.

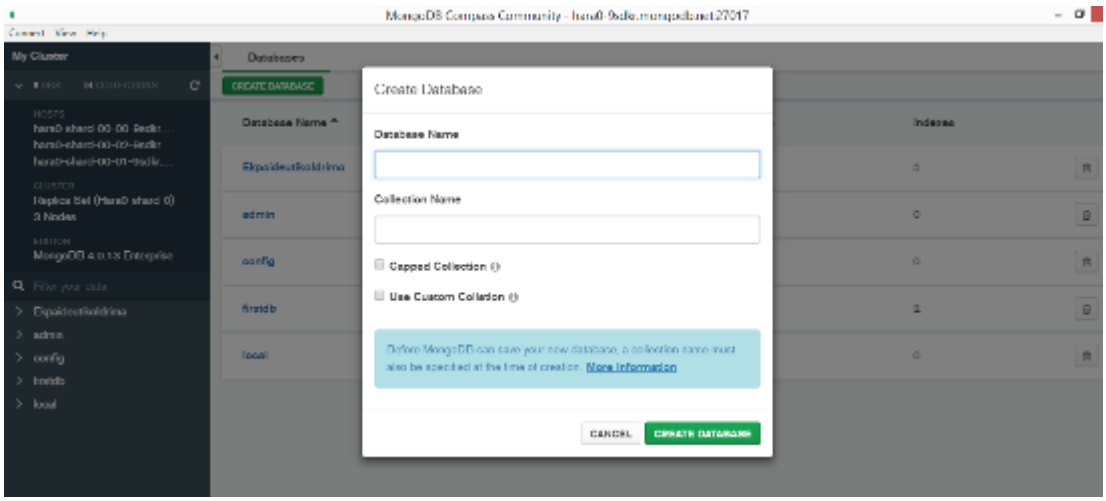
Στο γραφικό περιβάλλον του Compass, εμφανίζονται εξαρχής οι βάσεις δεδομένων που έχουν δημιουργηθεί, όπως φαίνεται στην εικόνα 49.



Εικόνα 49. Δημιουργία βάσης δεδομένων στο γραφικό περιβάλλον της Compass

Η δημιουργία μιας βάσης δεδομένων είναι εύκολη στο γραφικό περιβάλλον της Compass και γίνεται κάνοντας χρήση του "κουμπιού" Create Database που βρίσκεται πάνω αριστερά, όπως φαίνεται στην εικόνα 49.

Στην εικόνα 50 παρουσιάζεται το πλαίσιο διαλόγου που εμφανίζεται και στο οποίο μπορεί να οριστεί το όνομα της βάσης δεδομένων αλλά και το όνομα και τα χαρακτηριστικά μιας Collection.

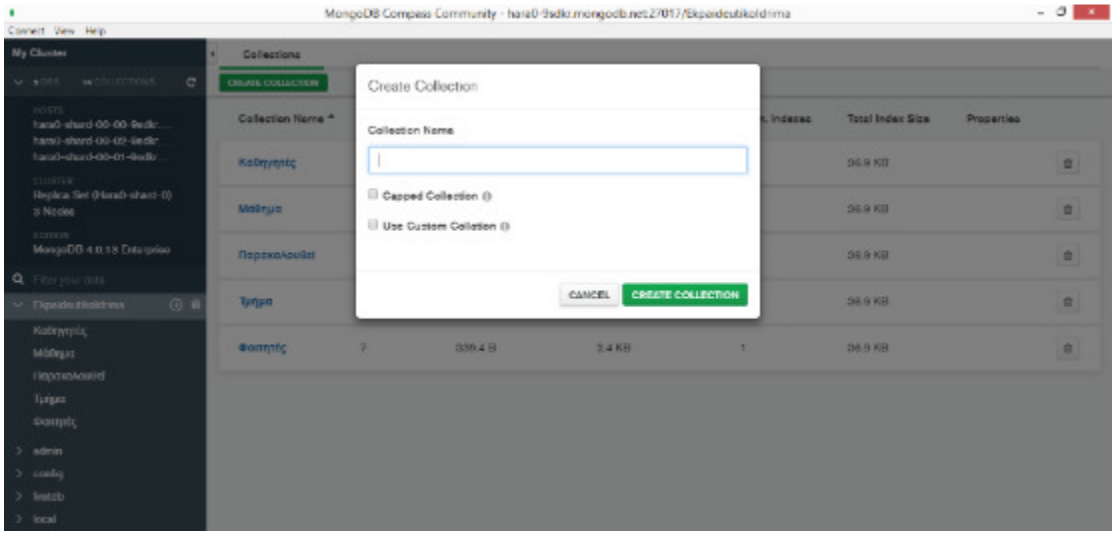


Εικόνα 50. Πλαίσιο διαλόγου ορισμού ονόματος της βάσης δεδομένων

3.2.1.2 Δημιουργία Collection σε μια βάση δεδομένων

Μπορεί να γίνει με την εντολή db.createCollection(όνομα);, οπότε δημιουργείται μια κενή βάση δεδομένων είτε μπορεί να δημιουργηθεί ταυτόχρονα με την εισαγωγή στοιχείων στη βάση δεδομένων. Η εντολή τότε είναι: db.όνομασυλλογής.insertOne(); ή db.όνομασυλλογής.insertMany();

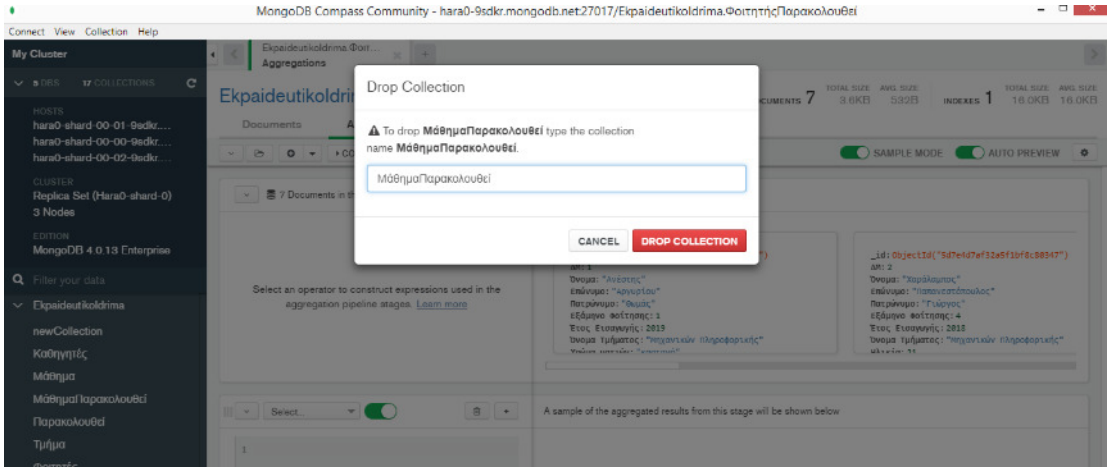
Στο περιβάλλον του Compass η διαδικασία είναι παρόμοια με αυτή που ακολουθήσαμε για να δημιουργήσουμε μια βάση δεδομένων και φαίνεται στην εικόνα 51.



Εικόνα 51. Δημιουργία νέας Collection στην βάση δεδομένων μέσα από το γραφικό περιβάλλον της MongoDB Atlas

3.2.1.3 Διαγραφή Collection σε μια βάση δεδομένων

Στο γραφικό περιβάλλον της MongoDB Atlas, δίνεται η δυνατότητα διαγραφής μιας Collection κάνοντας 'κλικ' στις τρεις τελίτσες που εμφανίζονται δίπλα από το όνομα της Collection και στη συνέχεια επιλέγοντας διαγραφή. Σαν επιβεβαίωση ζητείται να γραφεί το όνομα της Collection που είναι προς διαγραφή, κάτι που φαίνεται στην εικόνα 52.



Εικόνα 52. Διαγραφή Collection από την βάση δεδομένων μέσα από το γραφικό περιβάλλον της MongoDB Atlas

Η εντολή στο MongoShell είναι:

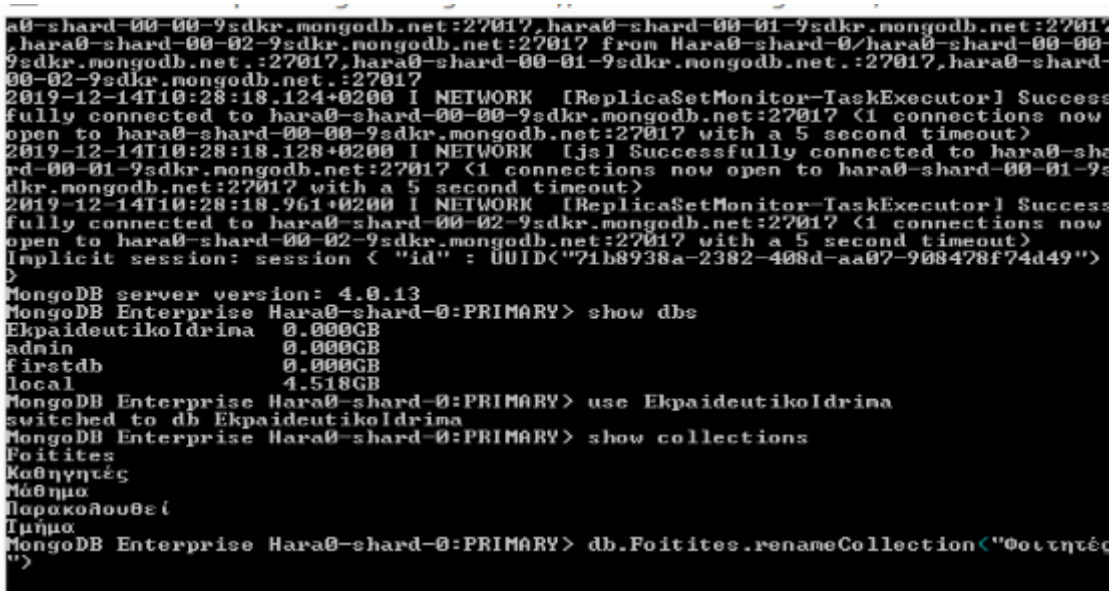
db.<Όνομα Collection>.drop();

db.<Όνομα Collection>.remove();

3.2.1.4 Μετονομασία Collection

Γίνεται με την εντολή db.<παλιό όνομα>.renameCollection("νέοόνομα");

Στην εικόνα 53 και στην εικόνα 54 φαίνεται η μετονομασία του Collection Foitites σε Φοιτητές

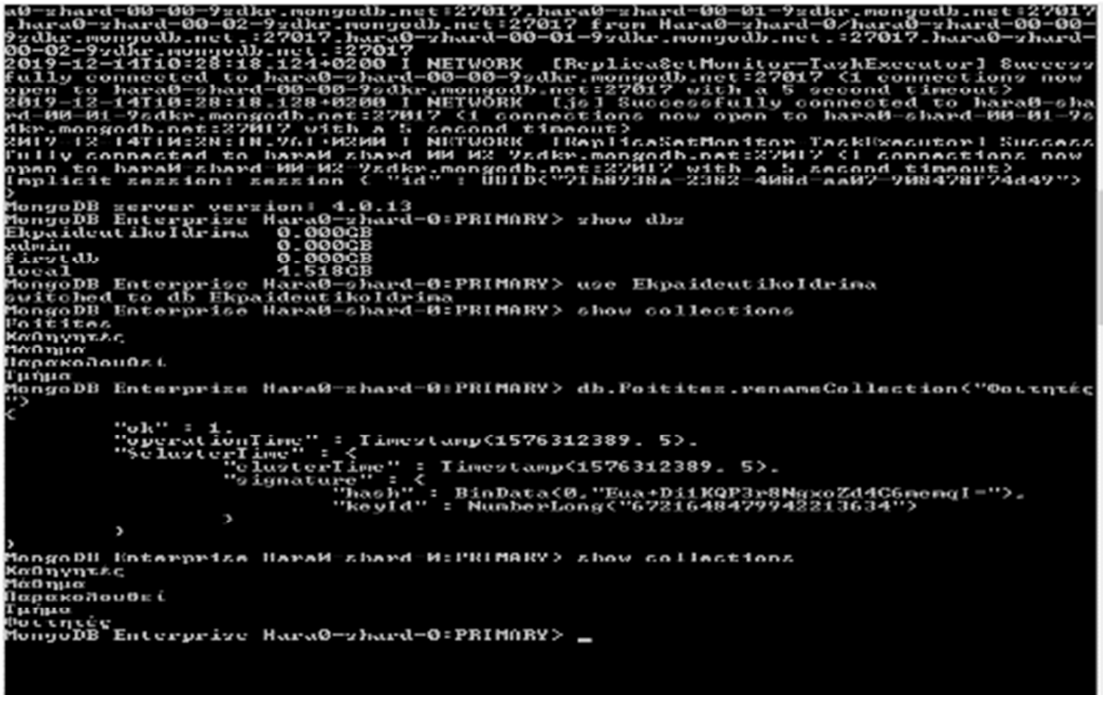


Εικόνα 53.Μετονομασία Collection μέσα από το Mongo Shell

Η εντολή είναι db.Foitites.renameCollection("Φοιτητές");

Εικόνα 54. Μετονομασία Collection μέσα από το Mongo Shell

Και έτσι στην εικόνα 55 φαίνεται πως όταν θα δοθεί η εντολή show collections, δεν εμφανίζεται πλέον το όνομα Foitites αλλά το όνομα Φοιτητές στην λίστα με τις Collections.



Εικόνα 55. Εμφάνιση της νέας λίστας των Collections

Μετονομασία μιας Collection γίνεται μόνο μέσω του MongoShell και όχι μέσω του γραφικού περιβάλλοντος της MongoDB Atlas.

3.2.1.5 Εισαγωγή στοιχείων σε μια Collection

Η εντολή είναι η **insertOne**, η οποία συντάσσεται ως εξής:

db.όνομασυλλογής.insertOne(όνομα πεδίου1: "τιμή πεδίου1", όνομα πεδίου2: "τιμή πεδίου2");

και έτσι για παράδειγμα στην Collection Μάθημα μια εγγραφή (document) δημιουργείται με την εντολή db.Μάθημα.insertOne(Κωδικός Μαθήματος:78, Τίτλος: "Υπολογιστικά Συστήματα", Διδακτικές Μονάδες: 5, Διδάσκων: "Δεσύλης");

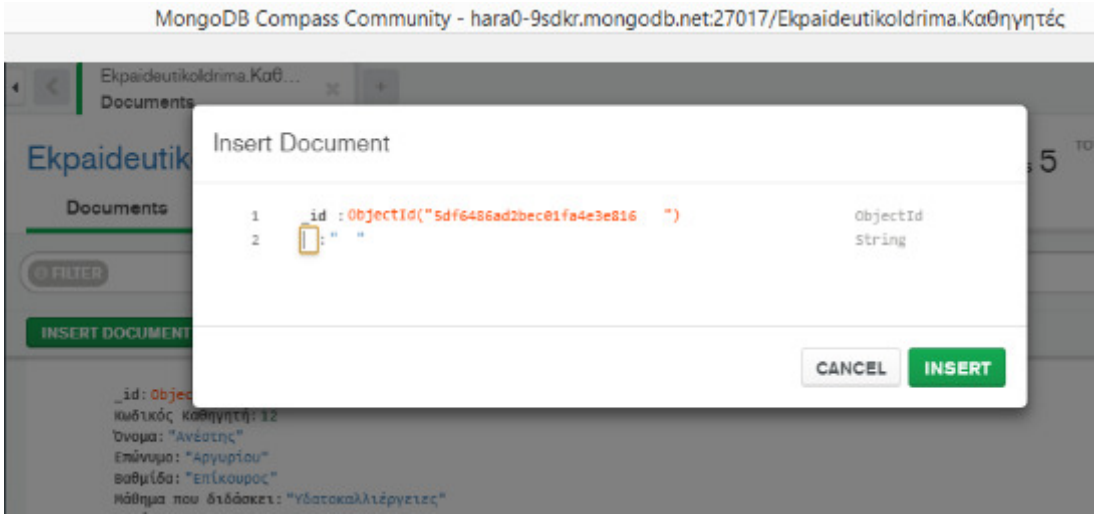
Ένας άλλος τρόπος εισαγωγής εγγραφών σε μια Collection είναι η εντολή **insertMany** με την οποία δίνεται η δυνατότητα εισαγωγής περισσοτέρων της μιας εγγραφών και η οποία συντάσσεται ως εξής:

db.όνομασυλλογής.insertMany({όνομα πεδίου1: "τιμή πεδίου1", όνομα πεδίου2: "τιμή πεδίου2"}, { όνομα πεδίου1: "τιμή πεδίου1", όνομα πεδίου2: "τιμή πεδίου2", όνομα πεδίου3: "τιμή πεδίου3"});

Ένα document μπορεί να έχει εμφωλευμένα πεδία. Μπορεί για παράδειγμα να υπάρχουν τα πεδία Όνομα, Ηλικία, Διεύθυνση και μέσα στο πεδίο Διεύθυνση να υπάρχουν τα πεδία Οδός και Πόλη.

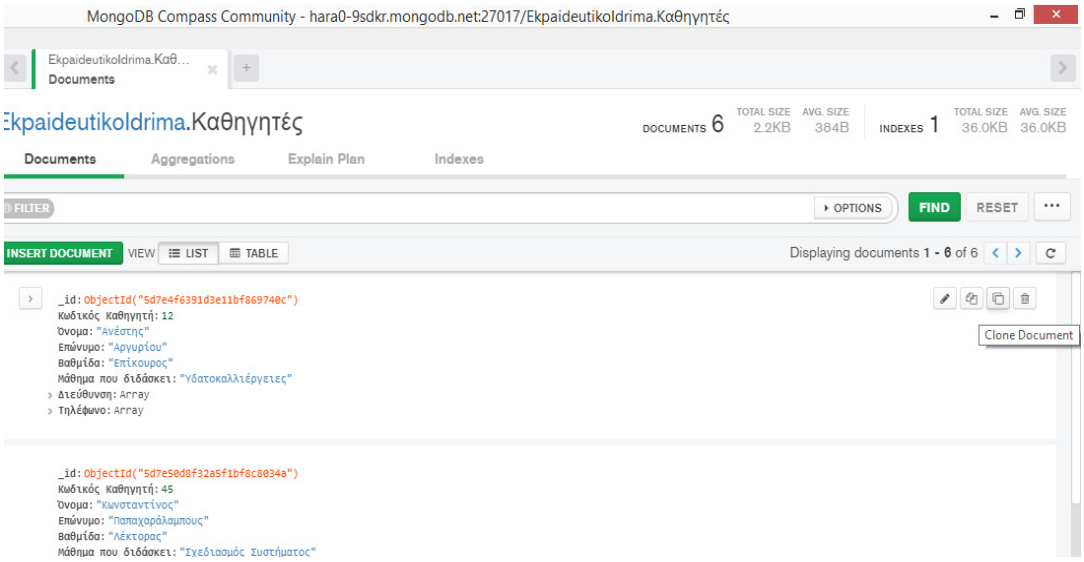
Όταν γίνεται εισαγωγή μιας εγγραφής δίνεται αυτόματα και ένας μοναδικός αριθμός για καθεμία από τις εγγραφές, που λέγεται ObjectId. Ο αριθμός αυτός λειτουργεί ως πρωτεύον κλειδί στη βάση δεδομένων.

Στο περιβάλλον του Compass, επιλέγεται αρχικά η Collection στην οποία θα εισαχθεί μια νέα εγγραφή και στην συνέχεια με το κουμπί Insert Document εμφανίζεται ένα πλαίσιο στο οποίο δίνεται η δυνατότητα καθορισμού των επιθυμητών πεδίων αλλά και του τύπου των τιμών του κάθε πεδίου. Όταν θα επιλεγεί Insert Document δίνεται αυτόματα ένα ObjectId το οποίο είναι μοναδικό για κάθε εγγραφή, όπως απεικονίζεται και στην εικόνα 56.



Εικόνα 56. Εισαγωγή δεδομένων σε μια Collection μέσα από το γραφικό περιβάλλον της MongoDB Atlas

Όπως απεικονίζεται στην εικόνα 57 δίνεται η δυνατότητα αντιγραφής ή και κλωνοποίησης μιας εγγραφής διατηρώντας έτσι τα πεδία μιας εγγραφής και αλλάζοντας τις τιμές τους. Η 'κλωνοποιημένη' εγγραφή θα είναι μια νέα εγγραφή με διαφορετικό ObjectId.



Εικόνα 57.Κλωνοποίηση εγγραφής στο περιβάλλον της Compass

3.2.1.6 Αλλαγή των τιμών των πεδίων σε μια Collection

Η εντολή είναι η update.

db.όνομαCollection.update();

Η εντολή db.Φοιτητές.update({Όνομα: "Ελένη"}, {$set: {Ηλικία:30}}); βρίσκει την εγγραφή που στο πεδίο Όνομα έχει την τιμή Ελένη και αλλάζει την τιμή του πεδίου Ηλικία σε 30.

Αν γραφεί η εντολή db.Φοιτητές.update({Όνομα: “Ελένη”}, {set: {Ηλικία:30}}); Δηλαδή η ίδια με πριν, χωρίς όμως το $ μπροστά από το set, τότε βρίσκει την εγγραφή που στο πεδίο Όνομα έχει την τιμή Ελένη και την αντικαθιστά ολόκληρη μόνο με ένα πεδίο που έχει όνομα Ηλικία και τιμή 30 χωρίς να αλλάξει το ObjectId της εγγραφής.

Σε περίπτωση που ζητηθεί να αλλάξει κάτι που δεν υπάρχει εμφανίζεται το μήνυμα Modified 0.

Υπάρχει η δυνατότητα δημιουργίας ενός πεδίου και αλλαγής του, ενώ αυτό δεν υπάρχει. Έτσι αν για παράδειγμα γραφεί η εντολή:

db.Φοιτητές.update({Όνομα: “Ελένη”}, { Όνομα: “Ελένη”, Επώνυμο: “Καλαμπόκα”},{upsert:true});

δημιουργείται μια νέα εγγραφή (αν δεν υπάρχει εγγραφή) με τιμή Ελένη στο πεδίο Όνομα και αυτό γίνεται λόγω της παραμέτρου upsert .

Η εντολή

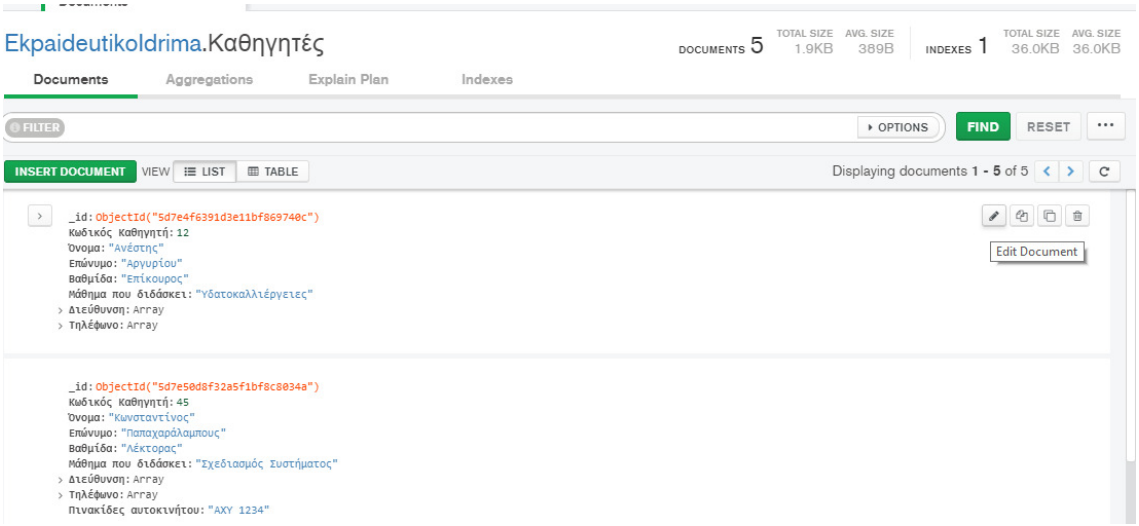
db.Φοιτητές.update({Όνομα: “Ελένη”}, {Όνομα: “Ελένη”, Επώνυμο: “Καλαμπόκα”},{multi:true}); Αλλάζει όλες τις εγγραφές με Όνομα Ελένη και τις κάνει να έχουν Όνομα Ελένη και Επώνυμο Καλαμπόκα.

Με την εντολή update μπορεί να αλλάξει το όνομα ενός πεδίου. Έτσι για παράδειγμα με την εντολή

db.Φοιτητές.update({Όνομα: “Ελένη”}, {$rename: {“Ηλικία”: “age”}});

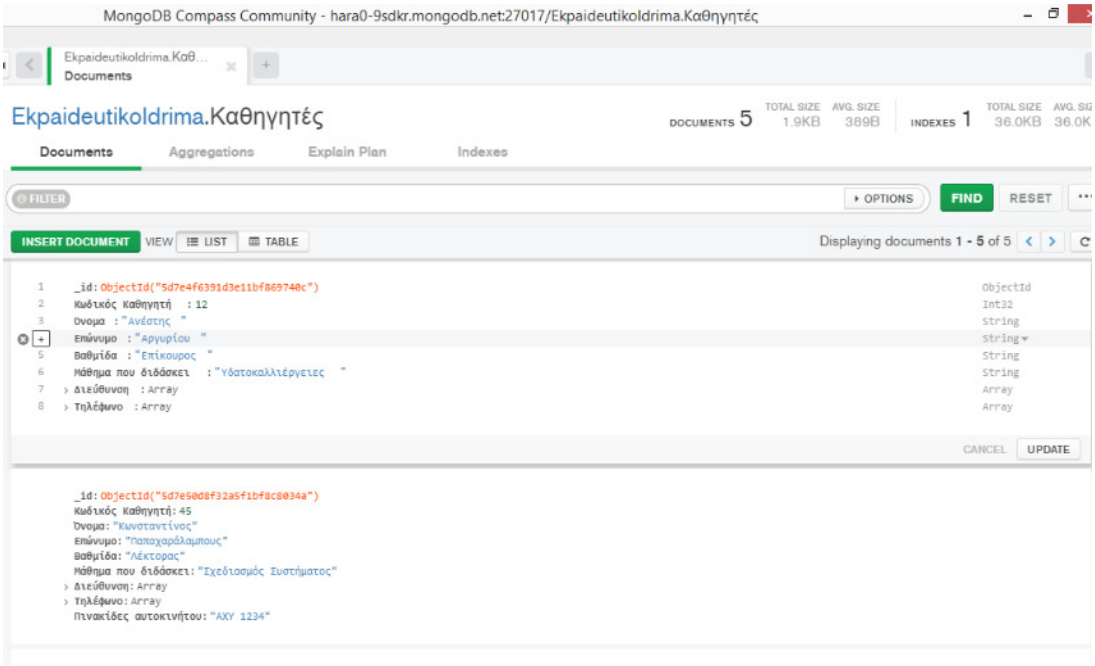
Στην εγγραφή με Όνομα Ελένη αλλάζει το όνομα του πεδίου Ηλικία σε age.

Στο γραφικό περιβάλλον του Compass η διαδικασία αυτή είναι αυτοματοποιημένη. Περνώντας το ποντίκι πάνω από μια εγγραφή εμφανίζεται ένα κουμπί με τίτλο Edit Document όπως φαίνεται και στην εικόνα 58.



Εικόνα 58. Αλλαγή τιμών του πεδίου ενός document

Το οποίο επιλέγοντάς το, δίνει την δυνατότητα αλλαγής της τιμής ενός πεδίου ή την αλλαγή του ονόματος του πεδίου ή ακόμα και με την διαγραφή ενός πεδίου και του περιεχομένου του όπως παρουσιάζεται στην εικόνα 59.



Εικόνα 59. Αλλαγή τιμών του πεδίου ενός document

3.2.1.7 Διαγραφή πεδίου από μια εγγραφή

Μπορεί να διαγραφεί κάποιο πεδίο από μια εγγραφή.

Με την εντολή

db.Φοιτητές.update({Όνομα: "Ελένη"}, {$unset: {Ηλικία:1}}); αφαιρείται το πεδίο Ηλικία από την εγγραφή με Όνομα Ελένη.

3.2.1.8 Διαγραφή ολόκληρης εγγραφής

Γίνεται με την εντολή remove ή με την εντολή deleteOne(); ή deleteMany(); .

Με την εντολή: db.Φοιτητές.deleteOne({Όνομα: "Ελένη"}); διαγράφεται η εγγραφή με τιμή Ελένη στο πεδίο Όνομα.

Με την εντολή db.Φοιτητές.remove({Όνομα: "Ελένη"}); διαγράφονται όλες οι εγγραφές που έχουν Όνομα Ελένη ενώ αν δοθεί η εντολή:

db.Φοιτητές.remove({Όνομα: "Ελένη"},{justOne:true}); διαγράφεται μόνο η πρώτη εγγραφή που θα βρει με Όνομα Ελένη.

Με την εντολή db.Φοιτητές.remove({}); διαγράφονται όλες οι εγγραφές της Collection Φοιτητές.

Στο περιβάλλον του Compass υπάρχει η δυνατότητα χρήσης της αυτοματοποιημένης λειτουργίας κάνοντας κλικ στο κουμπί Delete που εμφανίζεται όταν το ποντίκι 'περνάει' (hover) πάνω από μια εγγραφή.

3.2.2 Ερωτήματα - Queries

3.2.2.1 Εμφάνιση εγγραφών μιας Collection

Η εντολή είναι η find , η οποία συντάσσεται ως εξής:

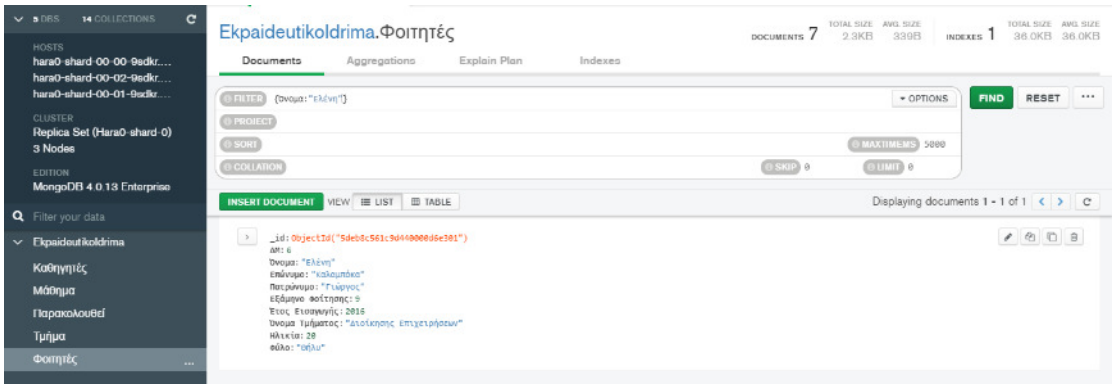
db.όνομαcollection.find(); η οποία βρίσκει και εμφανίζει όλα τα documents του Collection.

Μπορεί να επιλεγεί να εμφανίζεται μόνο ένα μέρος του Collection, για παράδειγμα τις εγγραφές εκείνες που κάποιο πεδίο έχει συγκεκριμένη τιμή. Έτσι αν δοθεί η εντολή

db.Φοιτητές.find({Ηλικία:20}); εμφανίζονται όλες οι εγγραφές που έχουν Ηλικία ίση με 20.

Με την εντολή db.Φοιτητές.find({Όνομα: “Ελένη”, Ηλικία:20}); εμφανίζονται όλες οι εγγραφές της Collection Φοιτητές που έχουν Όνομα Ελένη και Ηλικία ίση με 20.

Στο περιβάλλον του Compass. Στην γραμμή εντολών της find, αν πληκτρολογηθεί {Όνομα: “Ελένη”} εμφανίζονται όλες οι εγγραφές που έχουν όνομα Ελένη όπως απεικονίζεται και στην εικόνα 60.

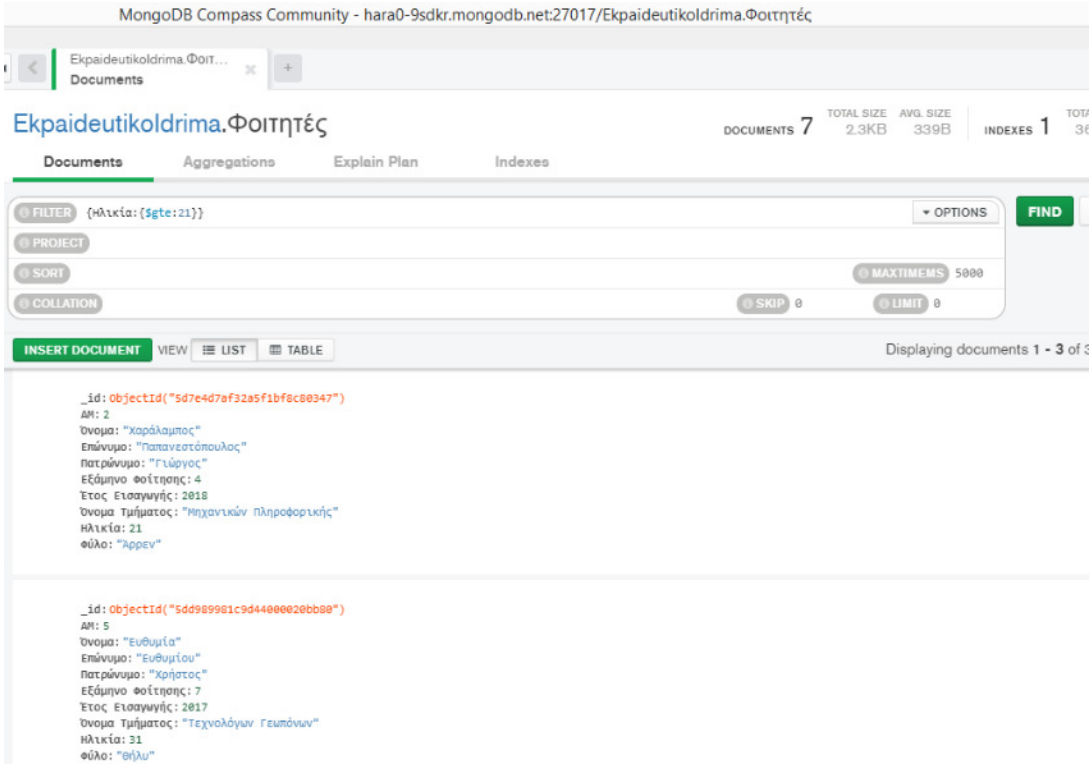


Εικόνα 60. Εύρεση φοιτητών που έχουν όνομα "Ελένη"

Στην εντολή find υπάρχει δυνατότητα χρήσης τελεστών, αριθμητικών, λογικών, συγκριτικών.

Έτσι για παράδειγμα με την εντολή: db.όνομαcollection.find({Ηλικία:{$gt:20}}); εμφανίζονται όλες οι εγγραφές που έχουν Ηλικία μεγαλύτερη του 20.

Στο περιβάλλον του Compass. Στην γραμμή εντολών της find αν πληκτρολογηθεί {Ηλικία:{$gte:21}} εμφανίζονται όλες οι εγγραφές που έχουν Ηλικία μεγαλύτερη ή ίση του 21, όπως φαίνεται και στην εικόνα 61.



Εικόνα 61. Εύρεση φοιτητών με ηλικία μεγαλύτερη από 21 έτη

Οι πιο συνηθισμένοι τελεστές είναι οι εξής:

$eq : =	$abs: απόλυτη τιμή	$max: μέγιστο	$and: και
$gt : >	$add: πρόσθεση	$min:ελάχιστο	$or: ή
$gte: ≥	$multiply:πολλαπλασιασμός	$avg:μέσος όρος	
$lt : <		$sum: άθροισμα	
$lte: ≤			

Η εντολή db.Φοιτητές.find({$or:[{Όνομα: "Ελένη"}, {Ηλικία:20}]); εμφανίζει όλες τις εγγραφές της Collection Φοιτητές που έχουν Όνομα Ελένη ή Ηλικία 20.

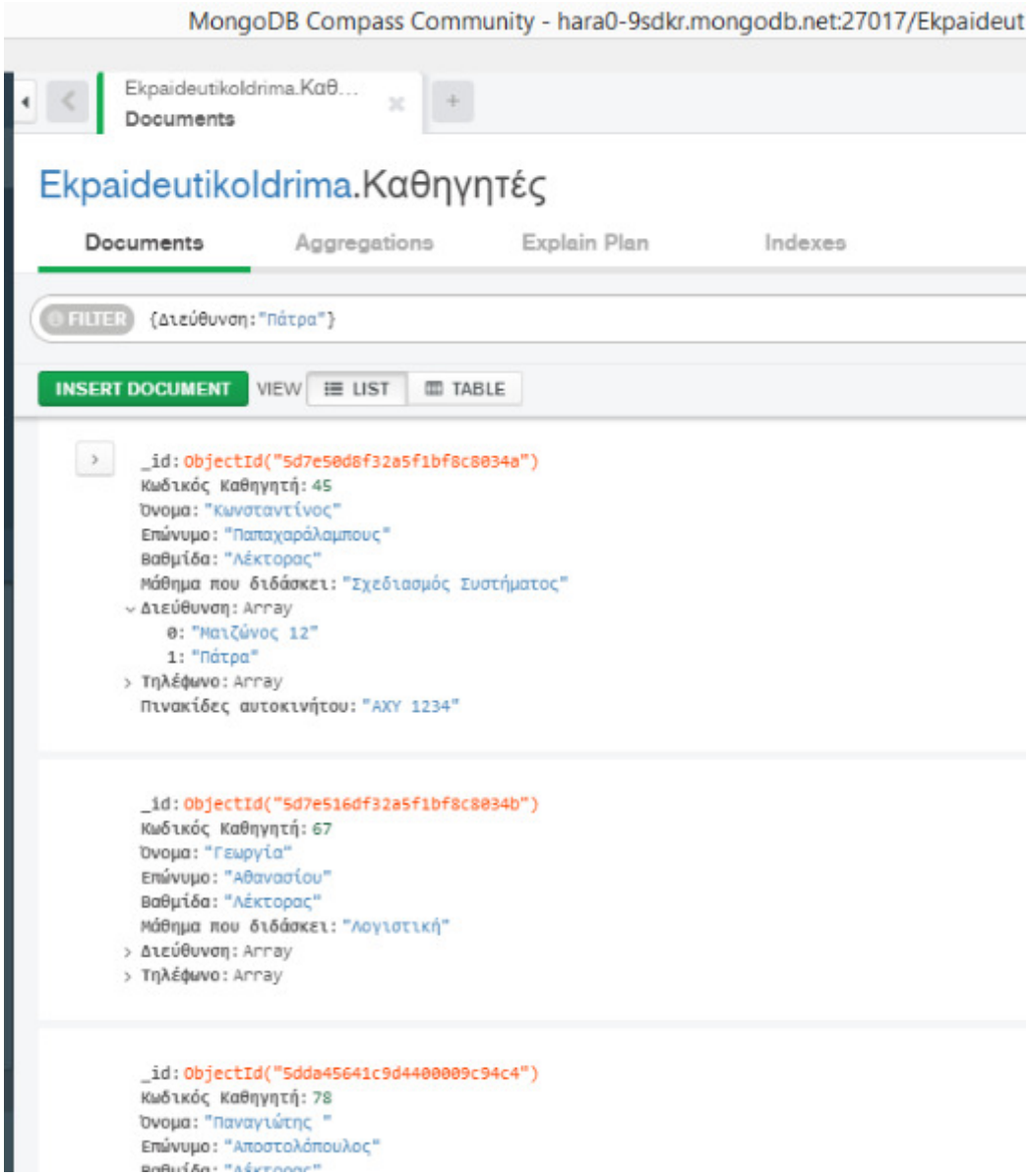
Αν ζητείται η εύρεση των εγγραφών που έχουν μια τιμή σε ένα πεδίο που βρίσκεται εμφωλευμένο σε ένα άλλο, τότε η εντολή είναι η εξής:

db.Καθηγητές.find({Διεύθυνση.Πόλη: "Μεσολόγγι"}); και μας εμφανίζονται όλες οι εγγραφές της Collection Καθηγητές που έχουν σαν πόλη το Μεσολόγγι.

3.2.2.2 Εύρεση καθηγητών που είναι από την Πάτρα

Η εντολή db.Καθηγητές.find(Διεύθυνση: "Πάτρα"); βρίσκει και εμφανίζει όλες τις εγγραφές της Collection Καθηγητές που έχουν στο πεδίο Διεύθυνση την πόλη Πάτρα.

Στην εικόνα 62 απεικονίζονται τα αποτελέσματα της αναζήτησης των καθηγητών που είναι από την Πάτρα. Να σημειώσουμε ότι παρ' όλο που η πόλη είναι εμφωλευμένο έγγραφο στο πεδίο Διεύθυνση, η MongoDB Atlas μπορεί να κάνει αναζήτηση και να βρίσκει όλα τα documents που έχουν την τιμή Πάτρα στο πεδίο Διεύθυνση.



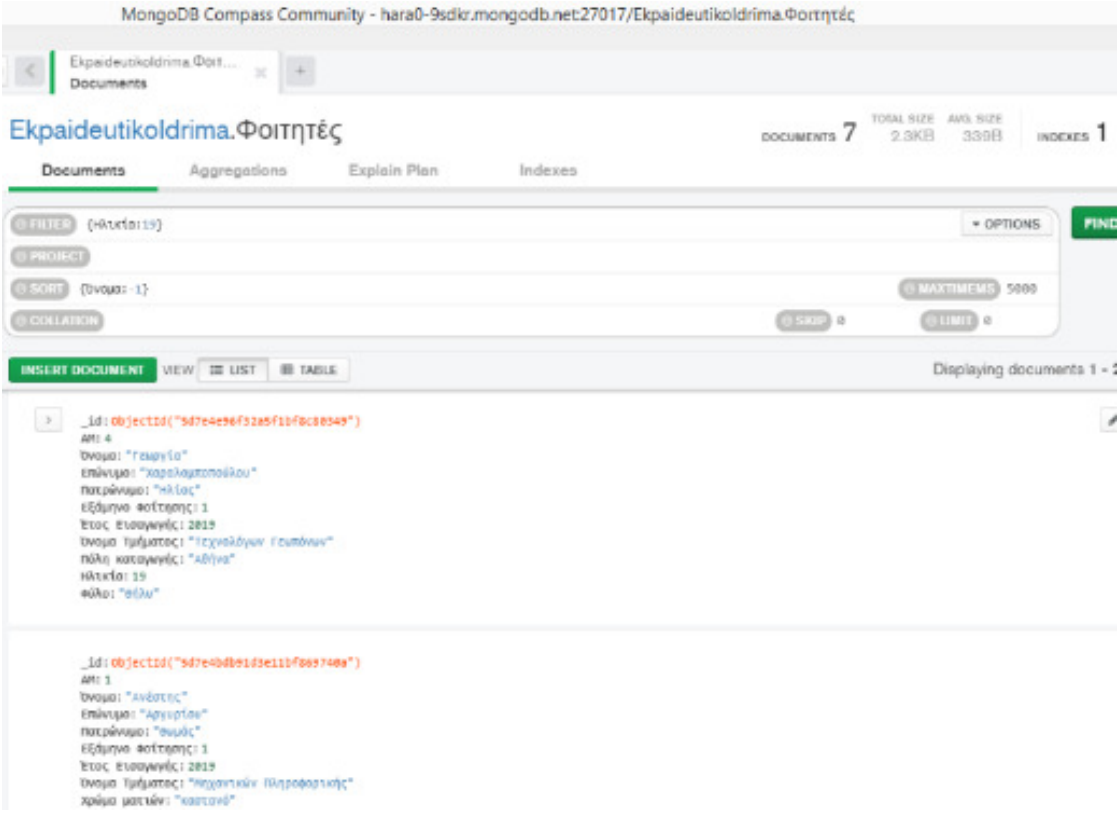
Εικόνα 62. Εύρεση καθηγητών που είναι από την Πάτρα στην MongoDB Atlas

3.2.2.3 Εύρεση και ταξινόμηση εγγραφών

Η εντολή db.Φοιτητές.find().sort({Επώνυμο:1}); Βρίσκει και εμφανίζει ταξινομημένες όλες τις εγγραφές της Collection Φοιτητές κατά αύξουσα σειρά ανάλογα με το πεδίο Επώνυμο.

Η εντολή db.Φοιτητές.find().limit(4).sort({Επώνυμο:-1}); Βρίσκει και εμφανίζει ταξινομημένες όλες τις εγγραφές της Collection Φοιτητές κατά φθίνουσα σειρά ανάλογα με το πεδίο Επώνυμο, όμως τελικά παρουσιάζει μόνο τις 4 πρώτες.

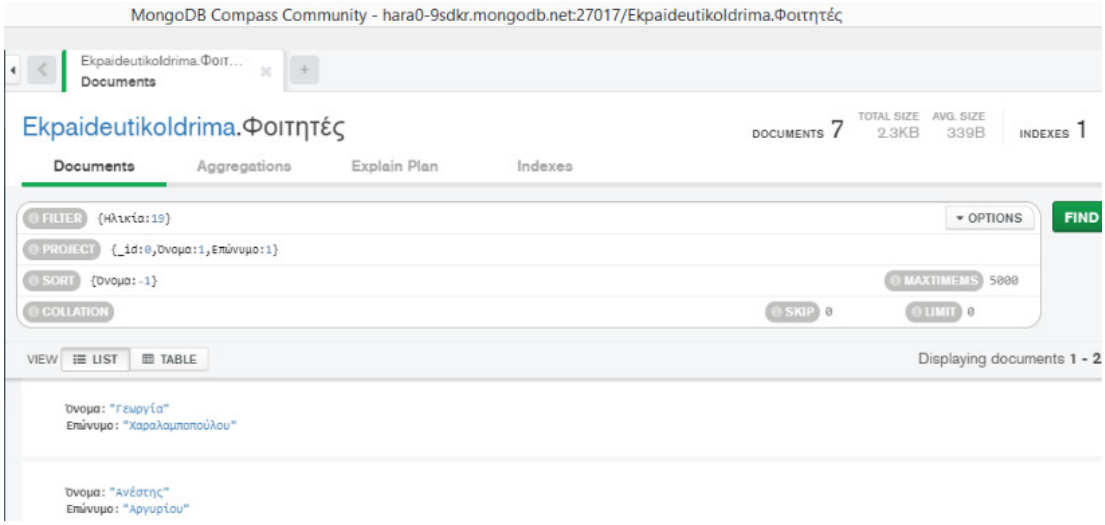
Στο περιβάλλον της Compass. Στην γραμμή εντολών της find πληκτρολογείται {Ηλικία:19}. Στην γραμμή της sort, πληκτρολογείται {Όνομα:-1} και εμφανίζονται όλες οι εγγραφές που έχουν Ηλικία ίση με 19, ταξινομημένες κατά φθίνουσα σειρά, όπως απεικονίζεται στην εικόνα 63. Αν πληκτρολογούταν {Όνομα:1} στην γραμμή του sort, τότε θα εμφανιζόταν τα αποτελέσματα ταξινομημένα κατά αύξουσα σειρά.



Εικόνα 63.Εύρεση φοιτητών με ηλικία 19 έτη και ταξινόμησή τους κατά τη φθίνουσα σειρά του ονόματός τους στην MongoDB Atlas

Αν δεν είναι επιθυμητό να εμφανίζονται όλα τα αποτελέσματα αλλά μόνο κάποιες εγγραφές, τότε θα πρέπει να αλλάξει ο αριθμός του limit από 0, σε αυτόν που είναι επιθυμητός. Έτσι, αν για παράδειγμα γραφεί ο αριθμός 2, τότε θα εμφανίζονται οι 2 πρώτες εγγραφές που πληρούν τα ζητούμενα κριτήρια.

Αν στην εμφάνιση των αποτελεσμάτων είναι καλύτερο να εμφανίζονται συγκεκριμένα πεδία, αυτό μπορεί να καθοριστεί από την γραμμή της συνάρτησης Project. Έτσι αν στο προηγούμενο παράδειγμα γραφεί {_id:0,Όνομα:1,Επώνυμο:1} τότε στα αποτελέσματα δεν θα εμφανίζεται το ObjectId και θα εμφανίζονται μόνο το Όνομα και το Επώνυμο των αποτελεσμάτων, όπως φαίνεται και στην εικόνα 64.



Εικόνα 64. Εύρεση φοιτητών με ηλικία 19 έτη, ταξινόμησή τους κατά τη φθίνουσα σειρά του ονόματός τους και εμφάνιση μόνο του ονόματος και του επωνύμου στην MongoDB Atlas

3.2.3 Aggregations Συνάθροιση – συγκεντρωτικά αποτελέσματα

Οι πιο συνηθισμένες συναρτήσεις συγκεντρωτικών αποτελεσμάτων είναι οι εξής:

$group: Συντάσσεται ως εξής:

{ $group:
{ _id: <έκφραση>,
<field1>: { <συσσωρευτής1> : <έκφραση1> },
...
}
}
ομαδοποιεί τις εγγραφές(documents) με βάση την έκφραση που βάζουμε στο _id. Η συνάρτηση $group επιστρέφει τις διακριτές τιμές της έκφρασης του _id.

$match: { $match: { <query> } } Φιλτράρει τις εγγραφές και κρατάει μόνο εκείνες που ταιριάζουν στην συγκεκριμένη συνθήκη.

$count: { $count: <string> } Μετράει τον αριθμό των εγγραφών που πληρούν μια συνθήκη.

$sort:

{ $sort: { <πεδίο1>: <σειρά ταξινόμησης>, <πεδίο2>: <σειρά ταξινόμησης>... } }

Ταξινομεί όλες τις εγγραφές και τις εμφανίζει σε ταξινομημένη σειρά, φθίνουσα ή αύξουσα κατά ένα επιλεγμένο πεδίο

$project: { $project: { "<πεδίο1>": 0, "<πεδίο2>": 1, ... } }

Καθορίζει ποια πεδία θα περιλαμβάνονται και ποια όχι στην τελική παρουσίαση των αποτελεσμάτων. Επίσης μπορεί να επαναπροσδιορίσει την τιμή πεδίων που υπάρχουν ήδη. Έτσι από την παραπάνω συνάρτηση, στο τελικό αποτέλεσμα θα εμφανίζεται το όνομα και η τιμή του πεδίου 2 αλλά όχι του πεδίου 1.

$limit: { $limit: <θετικός ακέραιος> }

Καθορίζει τον μέγιστο αριθμό των εγγραφών που θα εμφανιστούν, ανάλογα με τον θετικό ακέραιο που βάζουμε στην συνθήκη.

$sum: { $sum: <έκφραση> }

Επιστρέφει το άθροισμα αριθμητικών τιμών ανάλογα με την έκφραση που έχουμε βάλει στην συνθήκη.

$max - $min : Καθορίζουν το ανώτερο και το κατώτερο όριο αντίστοιχα, ενός δείκτη

$avg: { $avg: <έκφραση> }: Επιστρέφει τον μέσο όρο αριθμητικών δεδομένων που τίθενται στην έκφραση της συνθήκης, αγνοώντας δεδομένα που δεν είναι αριθμητικά.

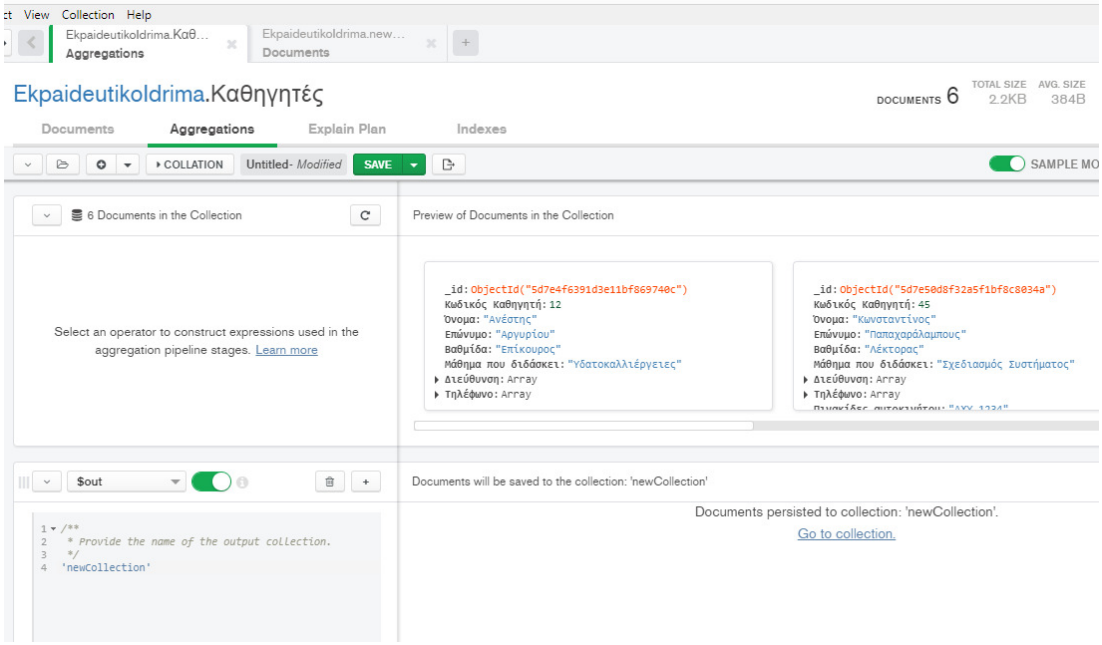
$push: { $push: <έκφραση> } Χρησιμοποιείται σε συνδυασμό με την $group και επιστρέφει έναν πίνακα όλων των τιμών ανάλογα με την έκφραση που βάζουμε στη συνθήκη και αφορά σε εγγραφές που έχουν ομαδοποιηθεί ($group) σύμφωνα με κάποιο κλειδί.

$out: { $out: "<το όνομα της νέας Collection>" }: Η συνάρτηση $out παίρνει τα αποτελέσματα από ένα aggregation και τα τοποθετεί σε μια συγκεκριμένη Collection. Αν αυτή η Collection δεν υπάρχει, την δημιουργεί απευθείας. Αν υπάρχει τότε σβήνει όλες τις εγγραφές και τις αντικαθιστά με τα καινούργια εξαγόμενα από το προηγούμενο aggregation.

Η συνάρτηση $out πρέπει να είναι το τελευταίο στάδιο σ΄ ένα aggregation.

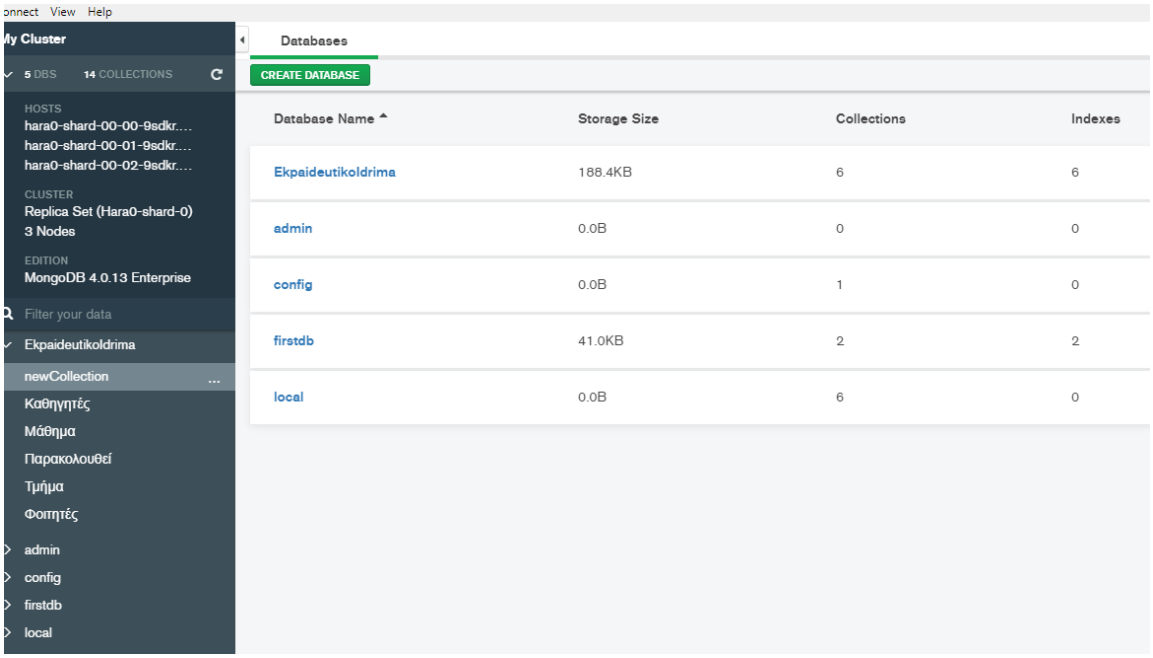
3.2.3.1 Δημιουργία πανομοιότυπης Collection με άλλο όνομα

Στο παράδειγμα που απεικονίζεται στην εικόνα 65 δημιουργήθηκε μια νέα Collection με το όνομα newCollection που ουσιαστικά περιέχει όλα τα documents της Collection Καθηγητές.



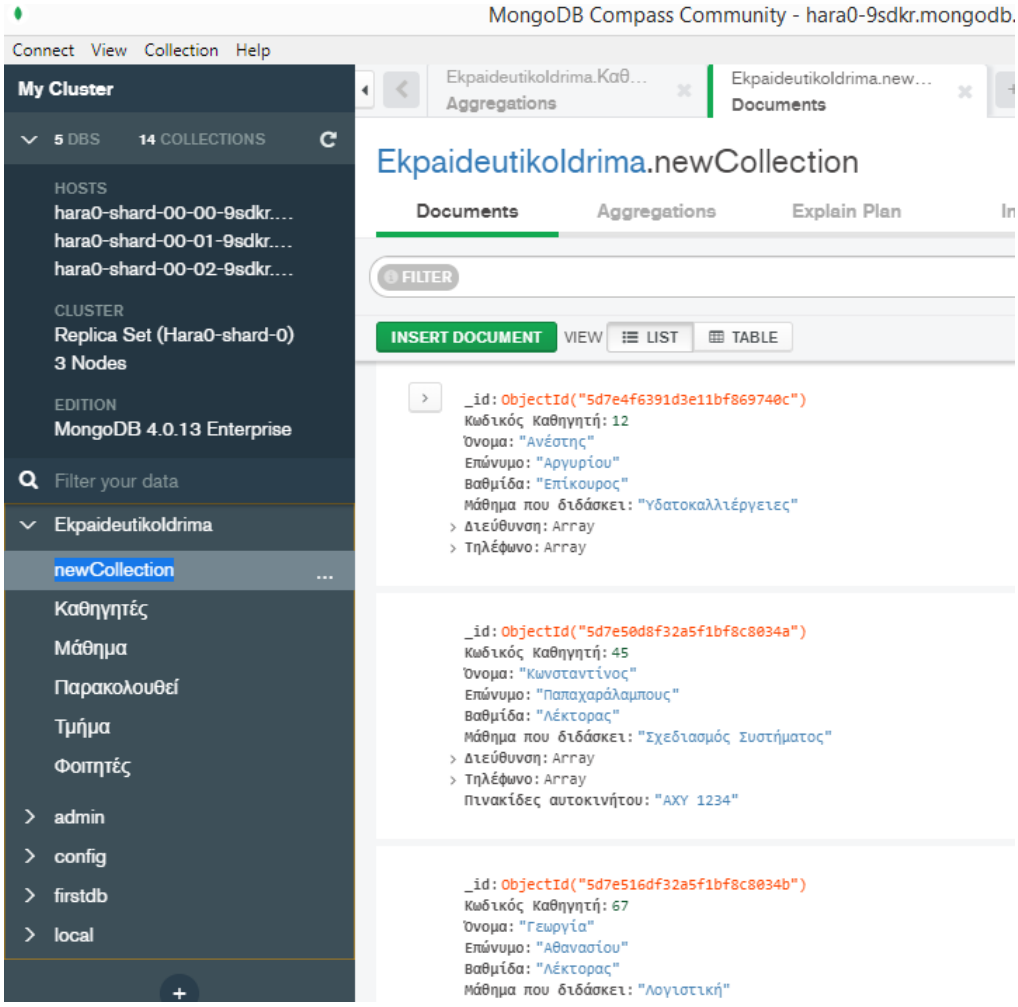
Εικόνα 65. Δημιουργία πανομοιότυπης Collection με άλλο όνομα στην MongoDB Atlas

Το ότι δημιουργήθηκε μια νέα Collection φαίνεται και στην εικόνα 66. Δημιουργήθηκε η Collection με όνομα newCollection μέσα στην βάση δεδομένων EkpaideutikoIdrima



Εικόνα 66. Εμφάνιση της νέας Collection στην βάση δεδομένων

Τα στοιχεία της newCollection φαίνονται στην εικόνα 67 και παρατηρείται ότι είναι ίδια με αυτά της Collection Καθηγητές. Μάλιστα και το ObjectId της κάθε εγγραφής παραμένει το ίδιο.



Εικόνα 67. Εμφάνιση των documents της newCollection

3.2.3.2 Εύρεση του πλήθους των εγγραφών που πληρούν ένα κριτήριο

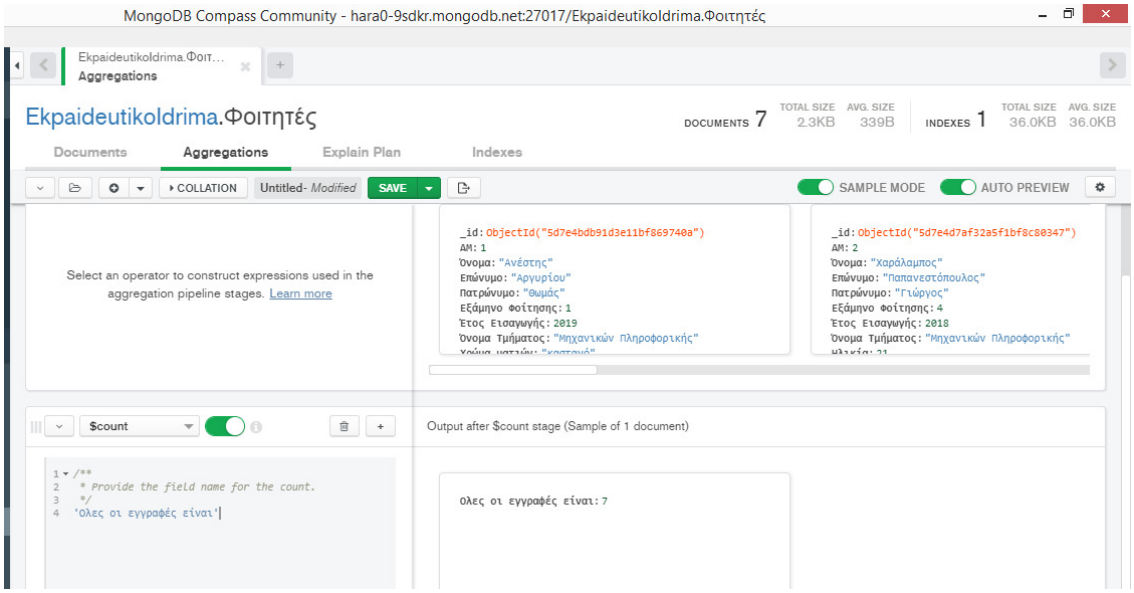
Η εντολή db.Φοιτητές.find().count(); βρίσκει τον αριθμό όλων των εγγραφών που βρίσκονται στην Collection Φοιτητές.

Η εντολή db.Φοιτητές.find({Φύλο: "Άρρεν"}).count(); βρίσκει τον αριθμό των εγγραφών που έχουν τιμή Άρρεν στο πεδίο Φύλο.

Στο περιβάλλον του Compass τα aggregations γίνονται σε στάδια. Η κάθε συνάρτηση προστίθεται σε διαφορετικό στάδιο. Εισερχόμενες εγγραφές στο δεύτερο στάδιο είναι τα αποτελέσματα (εξερχόμενες εγγραφές) από το πρώτο στάδιο κ.ο.κ

Όπως φαίνεται και στην εικόνα 68, στο πρώτο στάδιο επιλέγεται η συνάρτηση $count και συμπληρώνεται: 'Όλες οι εγγραφές είναι'.

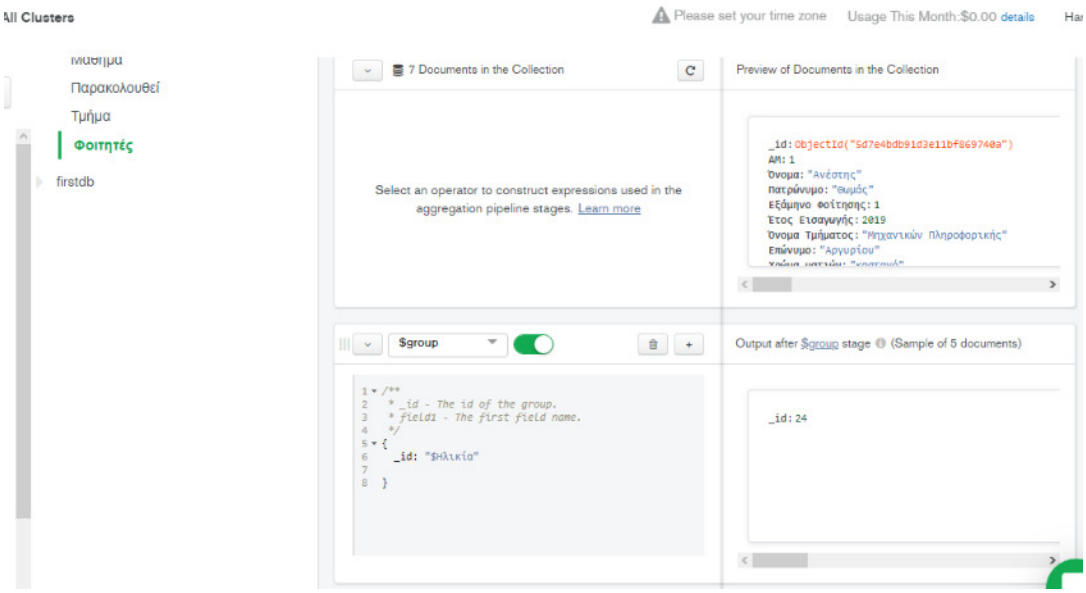
Το αποτέλεσμα είναι: Όλες οι εγγραφές είναι : 7, όσοι και οι εγγεγραμμένοι φοιτητές.



Εικόνα 68. Εύρεση του πλήθους των εγγραφών στην Collection Φοιτητές

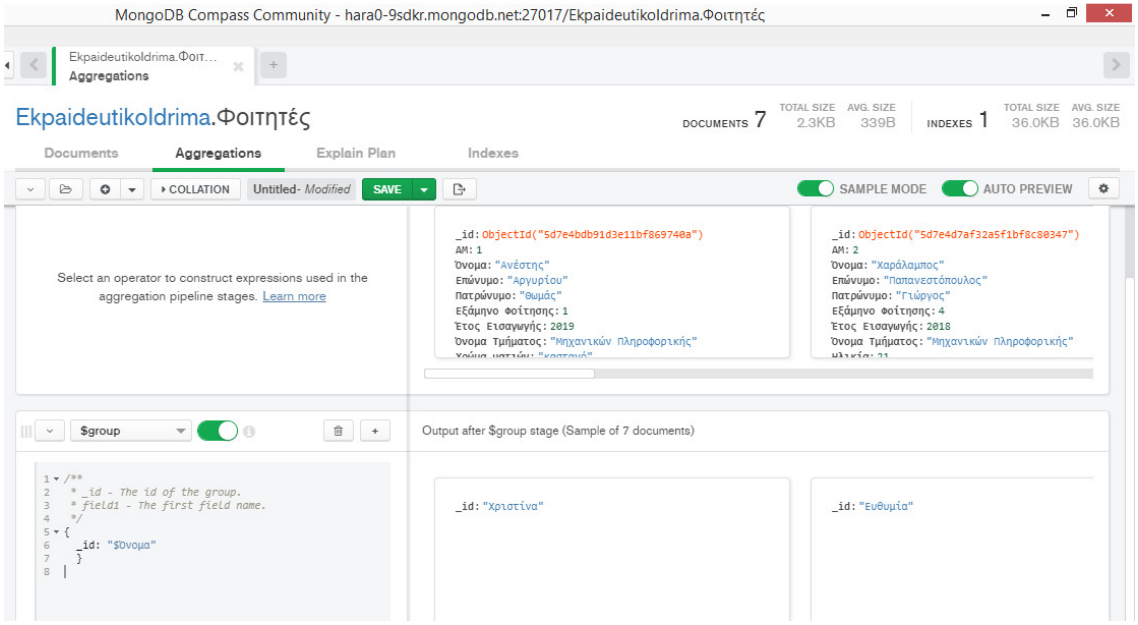
3.2.3.3 Εύρεση των ηλικιών των φοιτητών που υπάρχουν στη βάση δεδομένων

Στο πρώτο στάδιο, επιλέγεται η συνάρτηση $group και η έκφραση είναι:{_id:"$Ηλικία"}, όπως φαίνεται και στην εικόνα 68. Τα αποτελέσματα είναι οι διακριτές τιμές του πεδίου Ηλικία. Έτσι, παρ' όλο που υπάρχουν 7 εγγραφές, δίνει 5 αποτελέσματα και όχι 7 επειδή κάποιες ηλικίες είναι ίδιες.



Εικόνα 69. Εύρεση των διακριτών τιμών των ηλικιών των φοιτητών στην MongoDB Atlas

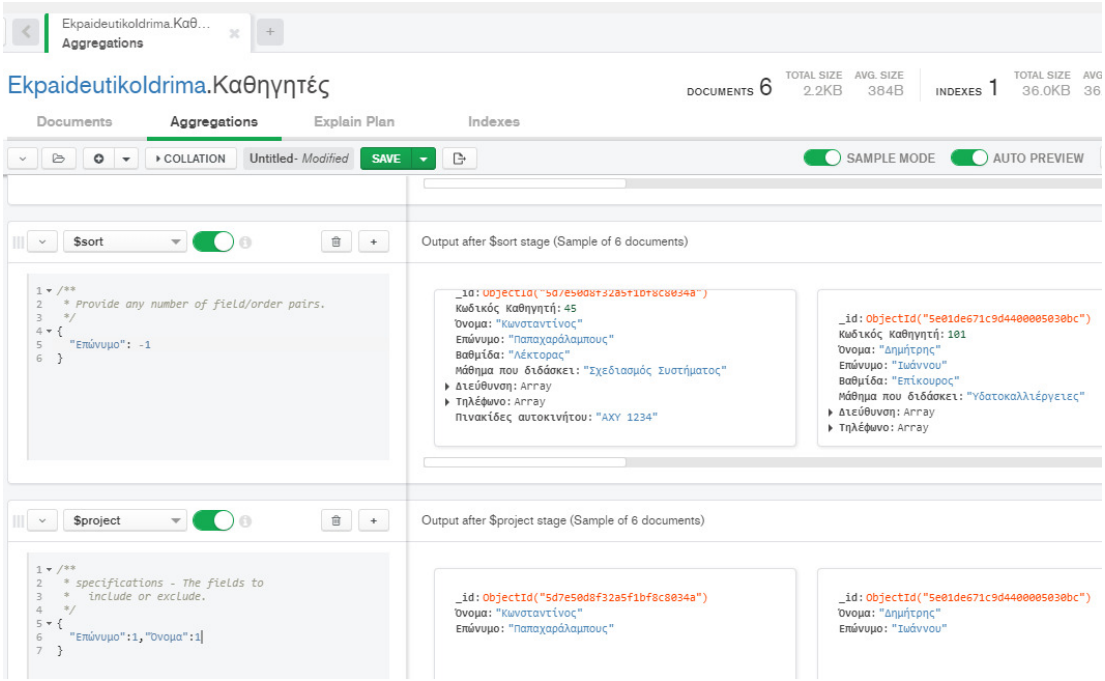
Με την ίδια λογική μπορούμε να βρούμε τα ονόματα των φοιτητών, όπως φαίνεται και στην εικόνα 70.



Εικόνα 70. Εύρεση των διακριτών τιμών των ονομάτων των φοιτητών στην MongoDB Atlas

3.2.3.4 Εμφάνιση ονόματος και επωνύμου των καθηγητών, ταξινομημένων κατά φθίνουσα σειρά

Στο πρώτο στάδιο χρησιμοποιείται η συνάρτηση $sort με την οποία ταξινομούνται τα documents με βάση το επώνυμο και κατά φθίνουσα σειρά, αφού η εντολή ήταν: {$sort: {"Επώνυμο": -1}}. Αν γράφαμε: {$sort: {"Επώνυμο": 1}}.
Στο δεύτερο στάδιο χρησιμοποιείται η συνάρτηση $project όπου καθορίζονται ποια είναι τα πεδία που θα εμφανίζονται.
Τα αποτελέσματα φαίνονται στην εικόνα 71, δεξιά στο χώρο του output.



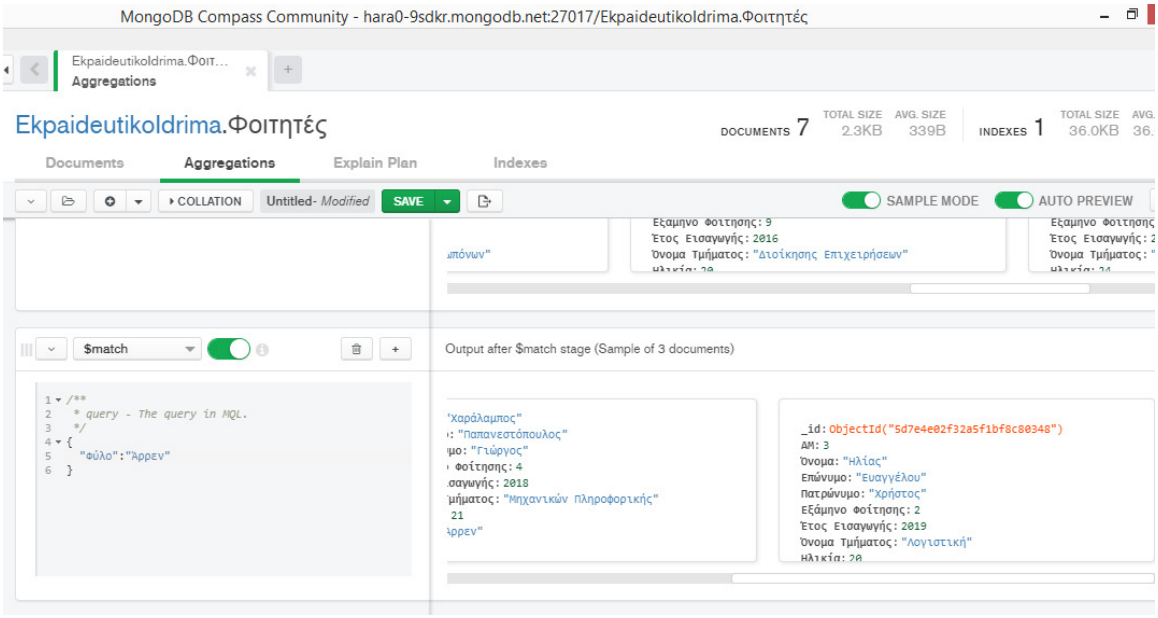
Εικόνα 71. Εμφάνιση ονόματος και επωνύμου των καθηγητών, ταξινομημένων κατά φθίνουσα σειρά στην MongoDB Atlas

Η συνολική εντολή είναι:

```
[ {$sort: {"Επώνυμο": -1}},
  {$project: {"Επώνυμο":1,"Όνομα":1}}
]
```

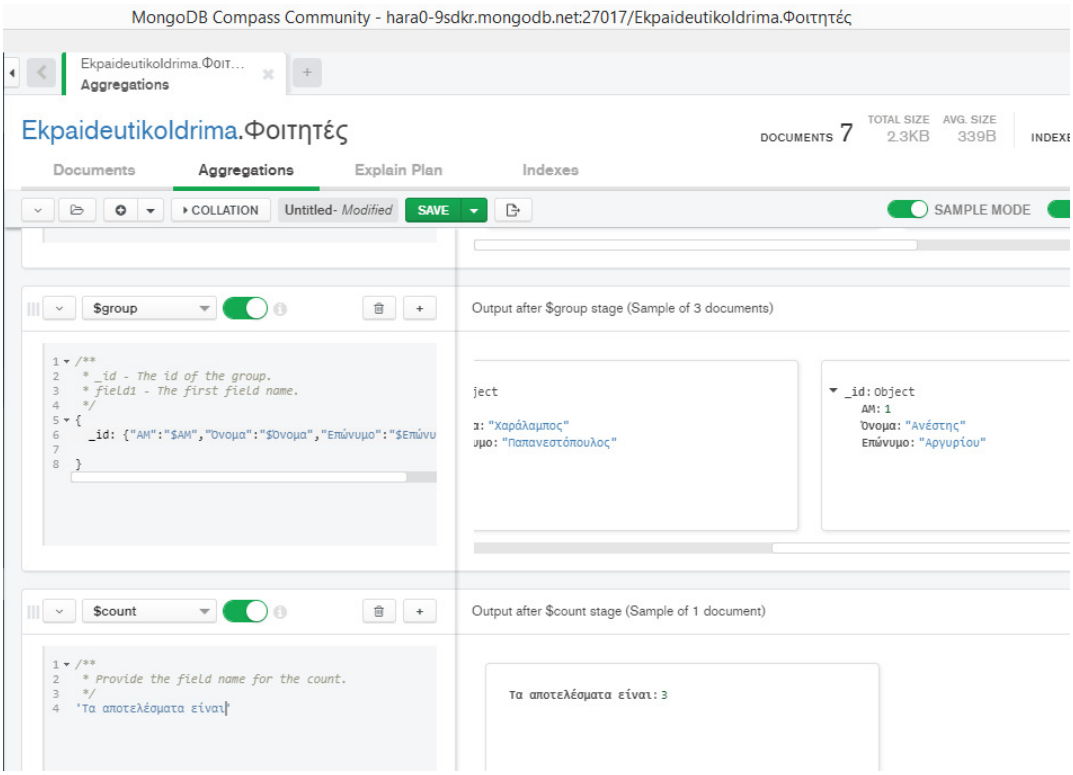
3.2.3.5 Εύρεση των αγοριών και εμφάνιση του ονόματος και του επιθέτου, αλλά και του πλήθους τους

Στο πρώτο στάδιο, όπως απεικονίζεται και στην εικόνα 72, χρησιμοποιείται η συνάρτηση $match η οποία φιλτράρει τις εγγραφές και κρατάει μόνο εκείνες που έχουν ως φύλο Άρρεν.



Εικόνα 72.1^{ο} στάδιο εύρεσης των αγοριών και εμφάνισης του ονόματος, του επιθέτου και του πλήθους τους

Στο δεύτερο στάδιο, το οποίο παρουσιάζεται στην εικόνα 73, χρησιμοποιείται η συνάρτηση $group η οποία ομαδοποιεί τα αποτελέσματα του πρώτου σταδίου και εμφανίζει τις τιμές ανάλογα με το ΑΜ, το όνομα και το επώνυμο. Στην εικόνα 73 επίσης φαίνεται το τρίτο στάδιο, όπου χρησιμοποιείται η συνάρτηση $count για να βρεθεί το πλήθος αυτών που είναι αγόρια.

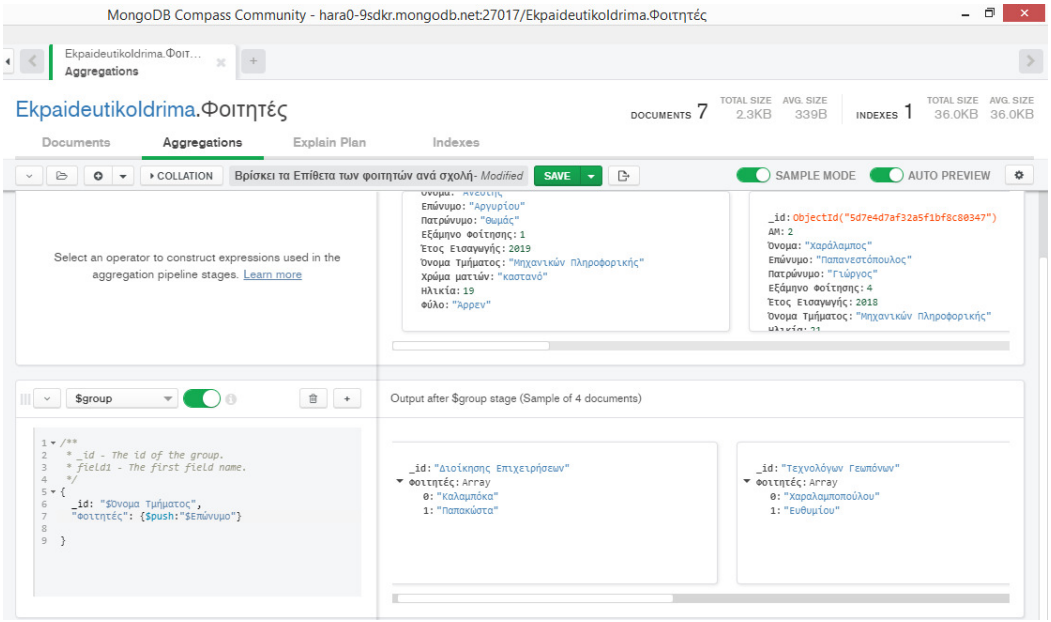


Εικόνα 73.2^{ο} και 3^{ο} στάδιο εύρεσης των αγοριών και εμφάνισης του ονόματος, του επιθέτου και του πλήθους τους

Η συνολική εντολή στο Mongo Shell είναι η εξής:

```
[{$match: { "Φύλο":"Άρρεν"}},
 {$group: { _id: {"ΑΜ":"$ΑΜ", "Όνομα":"$Όνομα", "Επώνυμο":"$Επώνυμο"} }},
 {$count: 'Αποτελέσματα'}
]
```

3.2.3.6 Εύρεση των φοιτητών ανά σχολή και εμφάνιση των επωνύμων τους

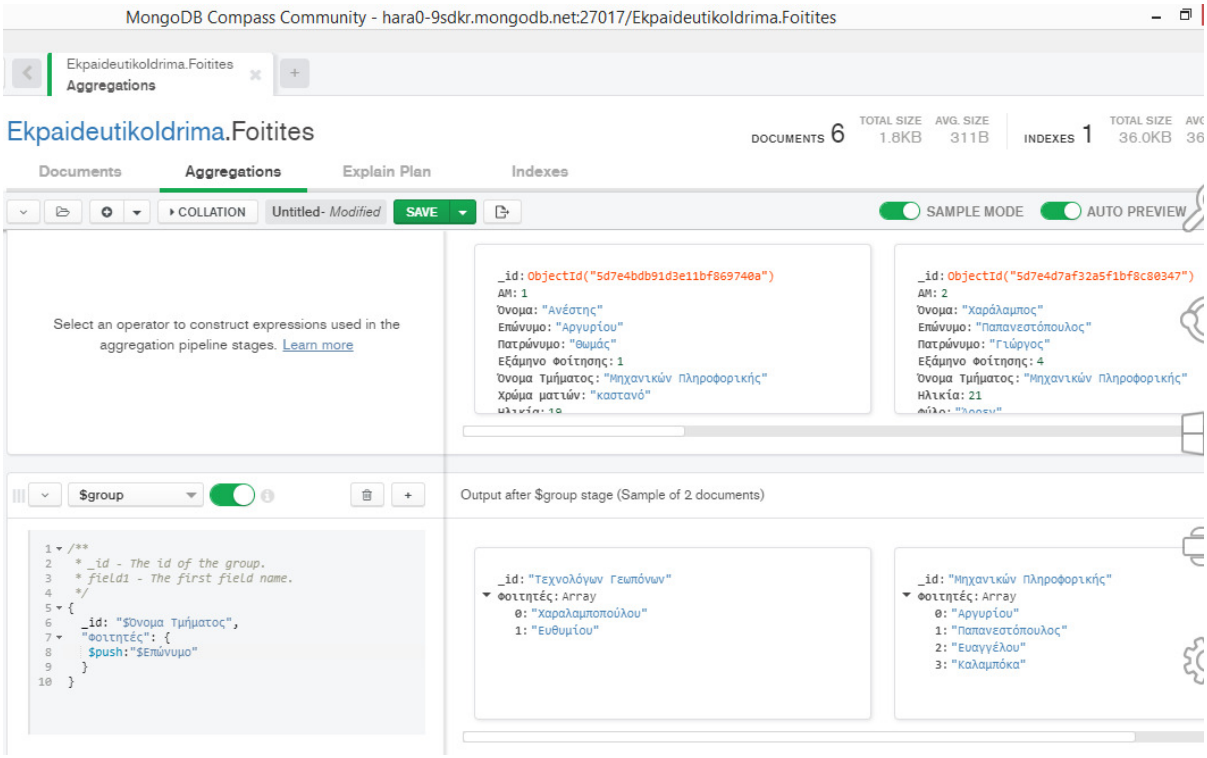


Εικόνα 74. Αποτελέσματα εύρεσης φοιτητών ανά σχολή και εμφάνισης των επιθέτων τους στην MongoDB Atlas

Χρειάζεται μόνο ένα στάδιο, το οποίο φαίνεται στην εικόνα 74 όπου χρησιμοποιείται η συνάρτηση $group και η συνολική εντολή είναι η εξής:

```
[{$group: { _id: "$Όνομα Τμήματος",
            "Φοιτητές": {$push:"$Επώνυμο"}
           }
}]
```

Στην εικόνα 75 απεικονίζονται τα υπόλοιπα αποτελέσματα της αναζήτησης.



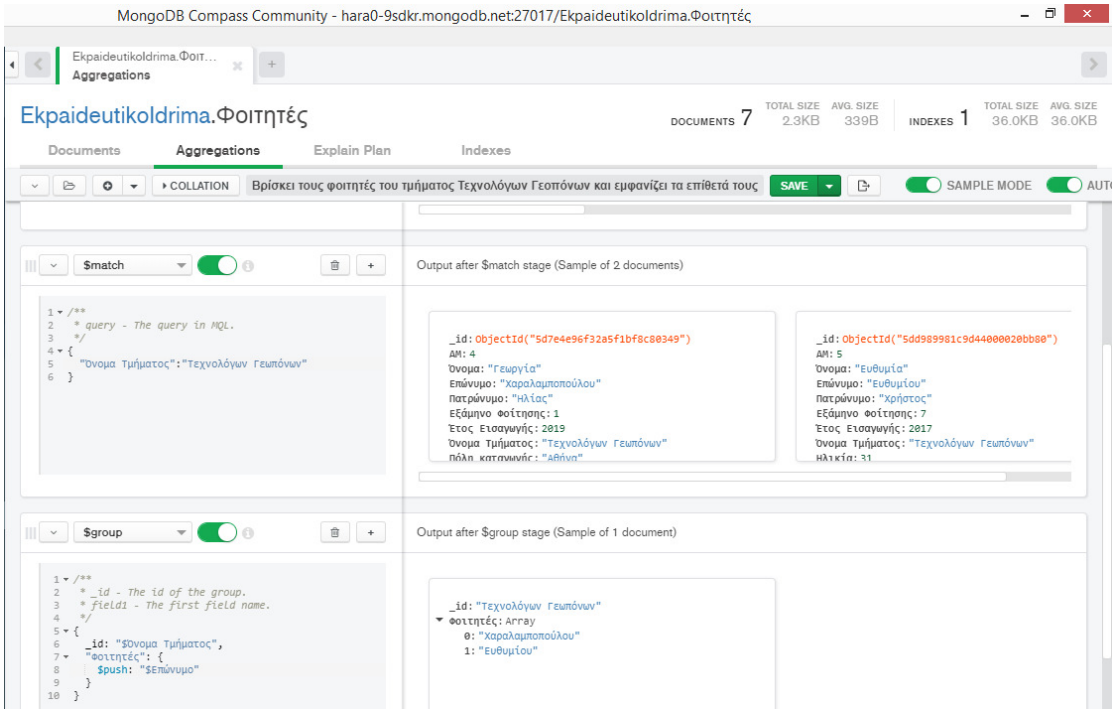
Εικόνα 75. Αποτελέσματα εύρεσης φοιτητών ανά σχολή και εμφάνισης των επιθέτων τους στην MongoDB Atlas

Με την $push, τα αποτελέσματα εμφανίζονται σε πίνακα με το όνομα Φοιτητές.

3.2.3.7 Εύρεση των φοιτητών του Τμήματος Τεχνολόγων Γεωπόνων

Όπως φαίνεται και στην εικόνα 76, στο πρώτο στάδιο χρησιμοποιείται η συνάρτηση $match η οποία φιλτράρει τις εγγραφές και κρατάει μόνο εκείνες που έχουν ως Όνομα Τμήματος το “Τεχνολόγων Γεωπόνων”.

Στο δεύτερο στάδιο, το οποίο επίσης φαίνεται στην εικόνα 76, χρησιμοποιείται η συνάρτηση $group η οποία ομαδοποιεί τα αποτελέσματα του πρώτου σταδίου και εμφανίζει τα ονόματα σε έναν πίνακα με όνομα Φοιτητές και για κάθε τμήμα ξεχωριστά, ανάλογα με το πεδίο Όνομα Τμήματος.



Εικόνα 76. Εύρεση φοιτητών του τμήματος Τεχνολόγων Γεωπόνων στην MongoDB Atlas

Η συνολική εντολή είναι:

```
[{$match: {
                "Όνομα Τμήματος":"Τεχνολόγων Γεωπόνων"
              }},
 {$group: {
                _id: "$Όνομα Τμήματος",
       "Φοιτητές": { $push:"$Επώνυμο"}
              }
}]
```

3.2.4 Συνάθροιση - Join δύο Collections

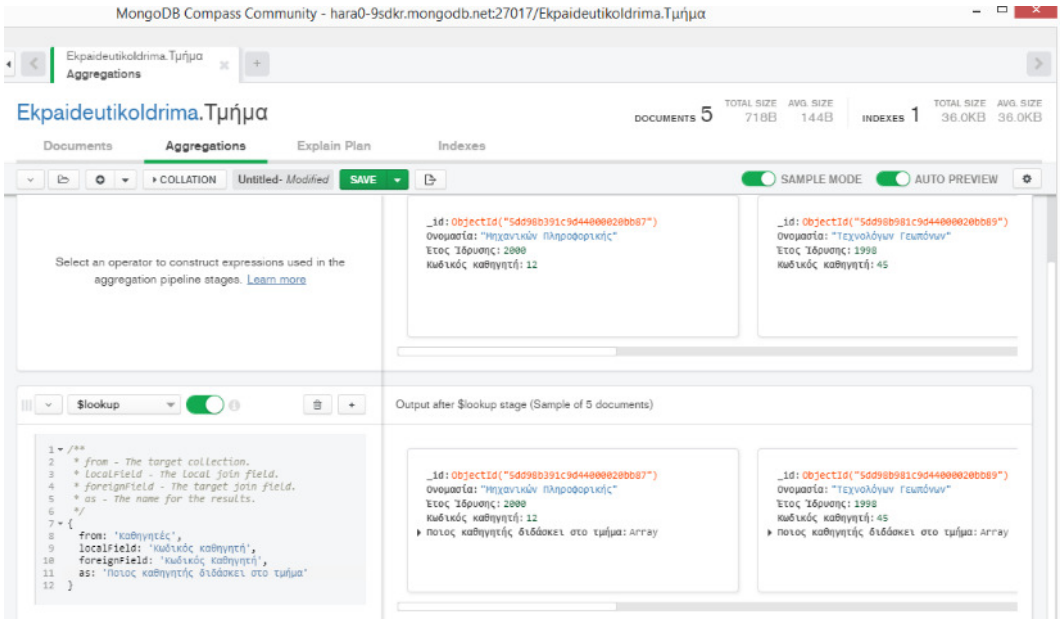
Η εντολή είναι η $lookup η οποία συντάσσεται ως εξής:

```
 db.< όνομα collection 1> . aggregate([
          { $ookup:
                    {
                      from: “collection 1”,
                      localfield: “όνομα πεδίου από την collection 1”,
                      foreignfield: “όνομα πεδίου από την collection 2”,
                       as: “Τίτλος”
                     }
          }
])
```

3.2.4.1 Εύρεση καθηγητών που διδάσκουν σε κάθε τμήμα

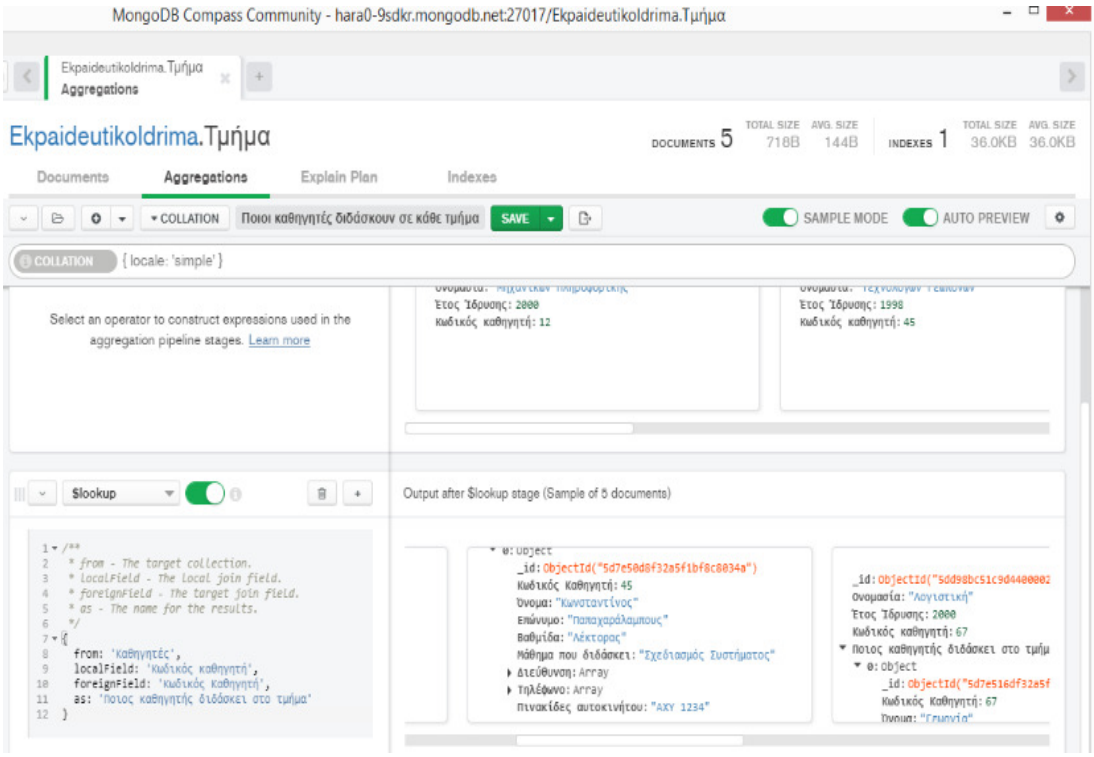
Στο πρώτο στάδιο, το οποίο φαίνεται στην εικόνα 77 γράφεται η εντολή:

```
[{$lookup: {
        from: 'Καθηγητές',
        localField: 'Κωδικός καθηγητή',
        foreignField: 'Κωδικός Καθηγητή',
        as: 'Ποιος καθηγητής διδάσκει στο τμήμα'
}}]
```



Εικόνα 77. Εύρεση καθηγητών που διδάσκουν σε κάθε τμήμα στην MongoDB Atlas

Τα αποτελέσματα παρουσιάζονται στην εικόνα 78 στον χώρο του output, όπου φαίνονται όλα τα στοιχεία της Collection Τμήμα και από κάτω σε μορφή πίνακα όλα τα στοιχεία των καθηγητών που διδάσκουν σε κάθε τμήμα (όπως είναι καταχωρισμένα στην Collection Καθηγητές).

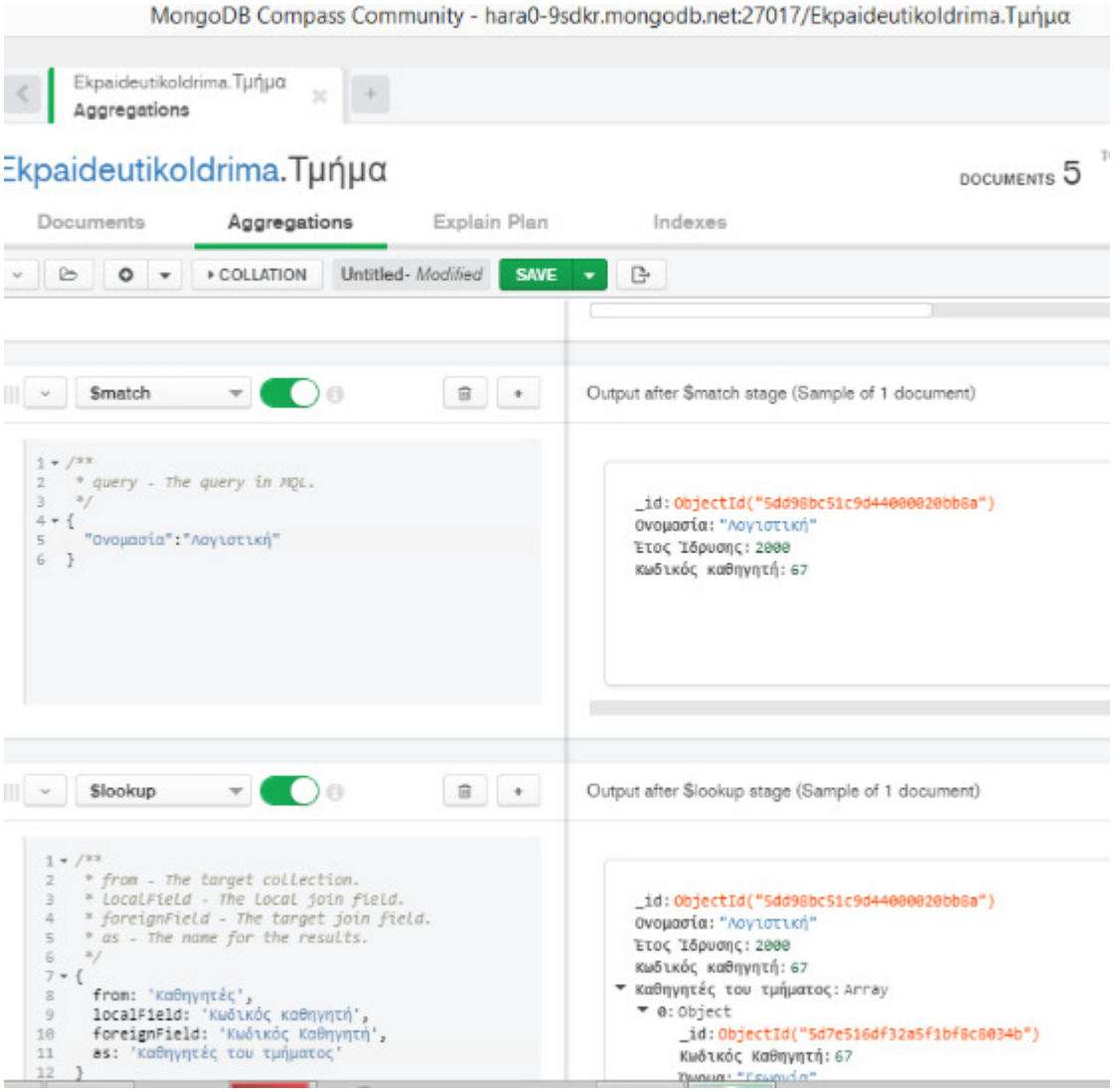


Εικόνα 78. Εύρεση καθηγητών που διδάσκουν σε κάθε τμήμα στην MongoDB Atlas

3.2.4.2 Εύρεση καθηγητών που διδάσκουν στην Λογιστική

Αν δεν αναζητούνται οι καθηγητές όλων των τμημάτων αλλά αν για παράδειγμα αναζητούνται οι καθηγητές του τμήματος Λογιστικής, τότε στο πρώτο στάδιο χρησιμοποιείται η $match η οποία φιλτράρει τις εγγραφές της Collection Τμήμα και κρατάει μόνο εκείνες που έχουν Ονομασία: “Λογιστική”.

Στο δεύτερο στάδιο χρησιμοποιείται η $lookup η οποία συνενώνει τις δύο Collections. Τα δύο στάδια απεικονίζονται στην εικόνα 79.



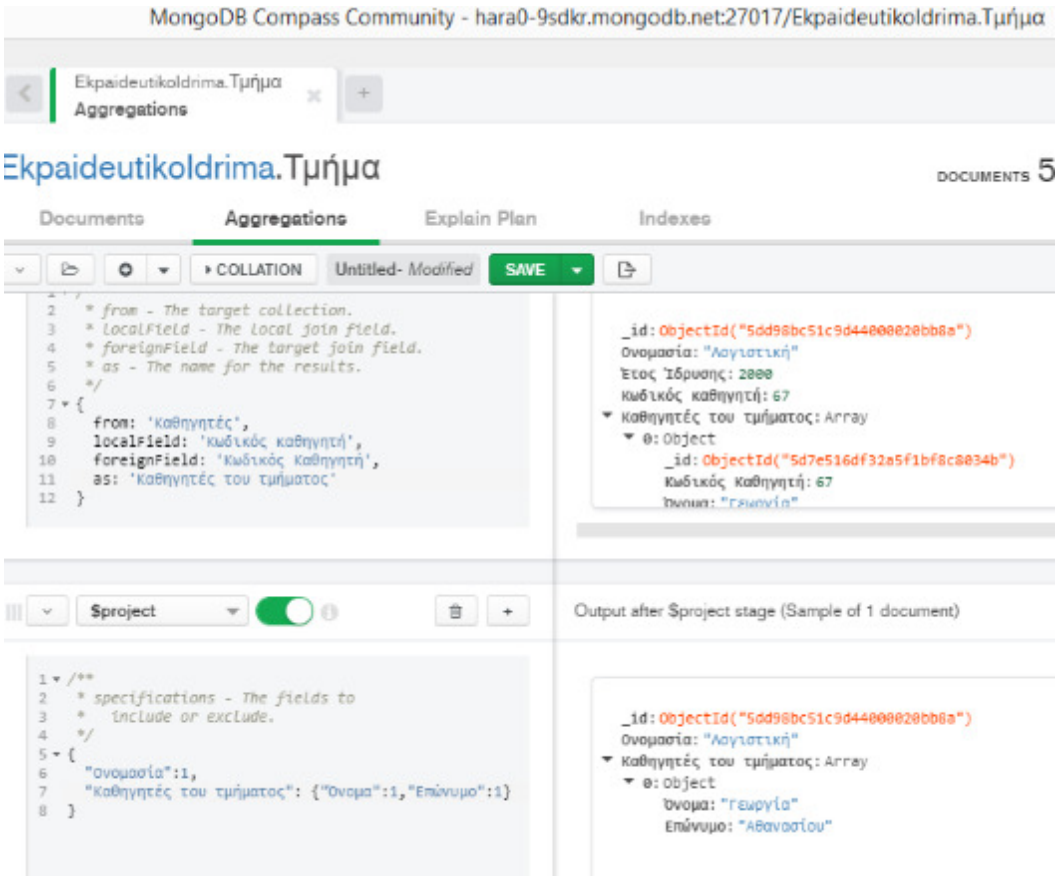
Εικόνα 79. Εύρεση καθηγητών που διδάσκουν στη Λογιστική, στην MongoDB Atlas

Όπως φαίνεται στην εικόνα 79, τα αποτελέσματα δίνονται πάλι με τη μορφή πίνακα. Αρχικά φαίνονται όλα τα στοιχεία του τμήματος Λογιστική (από την Collection Τμήμα) και από κάτω με μορφή πίνακα εμφανίζονται όλα τα στοιχεία των καθηγητών του τμήματος Λογιστική.

Αν δεν χρειάζεται να εμφανίζονται όλα τα στοιχεία της κάθε Collection, τότε υπάρχει η δυνατότητα επιλογής αυτών θα εμφανίζονται και αυτό γίνεται με την εντολή $project. Έτσι χρησιμοποιούνται τρία στάδια, όπως φαίνεται και στην εικόνα 80 και η συνολική εντολή είναι η εξής:

```
db.Τμήμα. aggregate([
[{$match: {
                "Ονομασία":"Λογιστική"
            }},
{$lookup: {
                from: 'Καθηγητές',
                localField: 'Κωδικός καθηγητή',
               foreignField: 'Κωδικός Καθηγητή',
                as: 'Καθηγητές του τμήματος'
```

```
            }},
{$project: {
                "Ονομασία":1,
                 "Καθηγητές του τμήματος": {"Όνομα":1,"Επώνυμο":1}
 }}] )
```



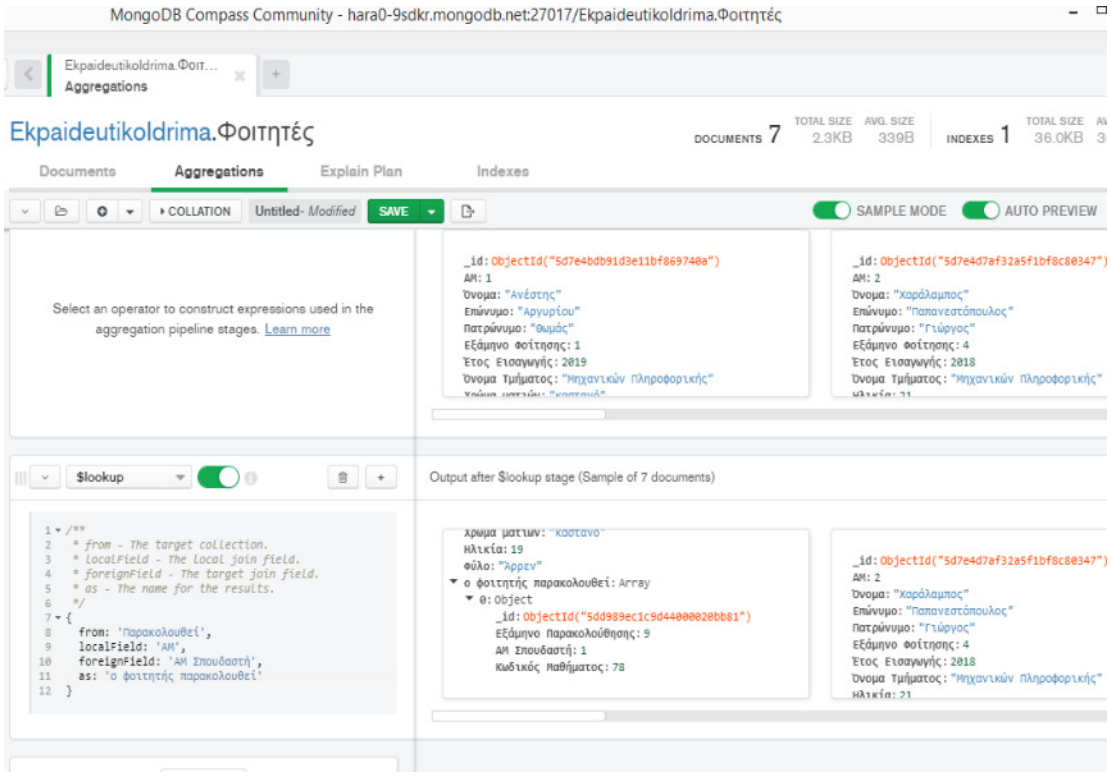
Εικόνα 80. Εύρεση καθηγητών που διδάσκουν στη Λογιστική και εμφάνιση μόνο του ονόματος και του επωνύμου τους, στην MongoDB Atlas

Μ' αυτόν τον τρόπο στα αποτελέσματα εμφανίζονται μόνο το όνομα του τμήματος και το όνομα και το επώνυμο των καθηγητών.

3.2.5 Συνάθροιση – Join τριών Collections

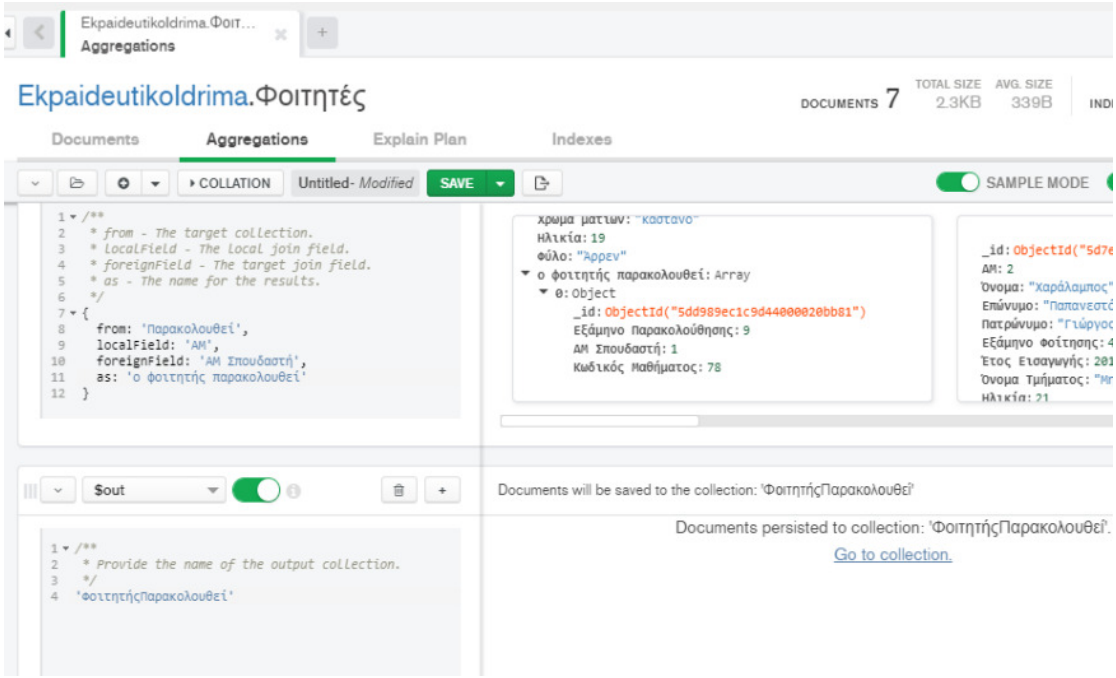
3.2.5.1 Εύρεση μαθημάτων που παρακολουθεί ο κάθε φοιτητής και σε ποιο τμήμα ανήκει

Το ερώτημα αυτό απαιτεί συνένωση (join) τριών Collections, κάτι που δεν υποστηρίζεται από την MongoDB Atlas. Για το λόγο αυτό δημιουργήθηκε μια νέα Collection όπως φαίνεται και στην εικόνα 81, που είναι αποτέλεσμα της συνένωσης των Collections Φοιτητές και Παρακολουθεί.



Εικόνα 81. Συνένωση των Collections Φοιτητές και Παρακολουθεί στην MongoDB Atlas

Η Collection ονομάστηκε ΦοιτητήςΠαρακολουθεί. Προκειμένου να επιτευχθεί αυτό, χρησιμοποιήθηκε η εντολή $out στο δεύτερο στάδιο του aggregation όπως φαίνεται και στην εικόνα 82.

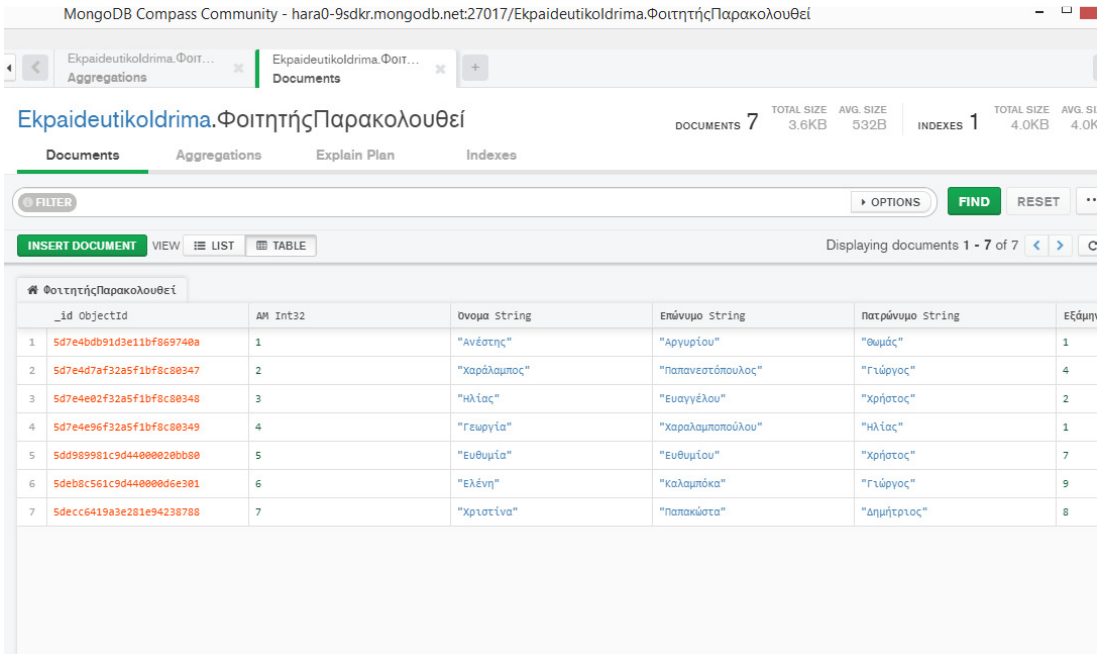


Εικόνα 82. Εξαγωγή των δεδομένων της συνένωσης των δύο Collections σε μια νέα με το όνομα ΦοιτητήςΠαρακολουθεί

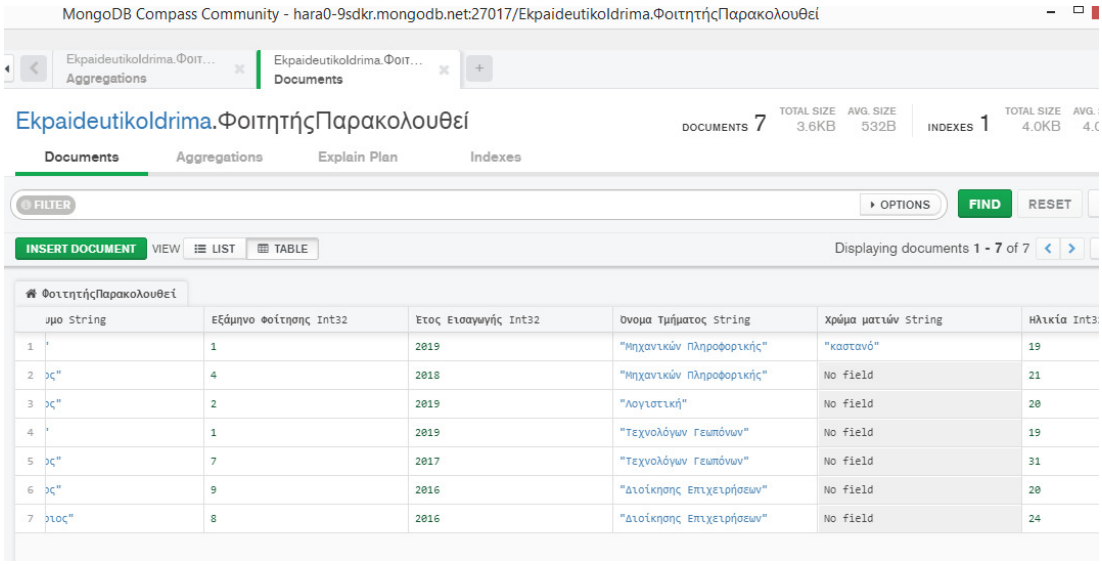
Η συνολική εντολή είναι:

```
db.Φοιτητές.aggregate(
[{    $lookup: {
                    from: 'Παρακολουθεί',
                    localField: 'AM',
                    foreignField: 'AM Σπουδαστή',
                     as: 'ο φοιτητής παρακολουθεί'
                   }},
        {$out: 'ΦοιτητήςΠαρακολουθεί'}
] )
```

Έτσι δημιουργήθηκε η Collection ΦοιτητήςΠαρακολουθεί, τα στοιχεία της οποίας φαίνονται υπό μορφή πίνακα στις δύο εικόνες 83 και 84.



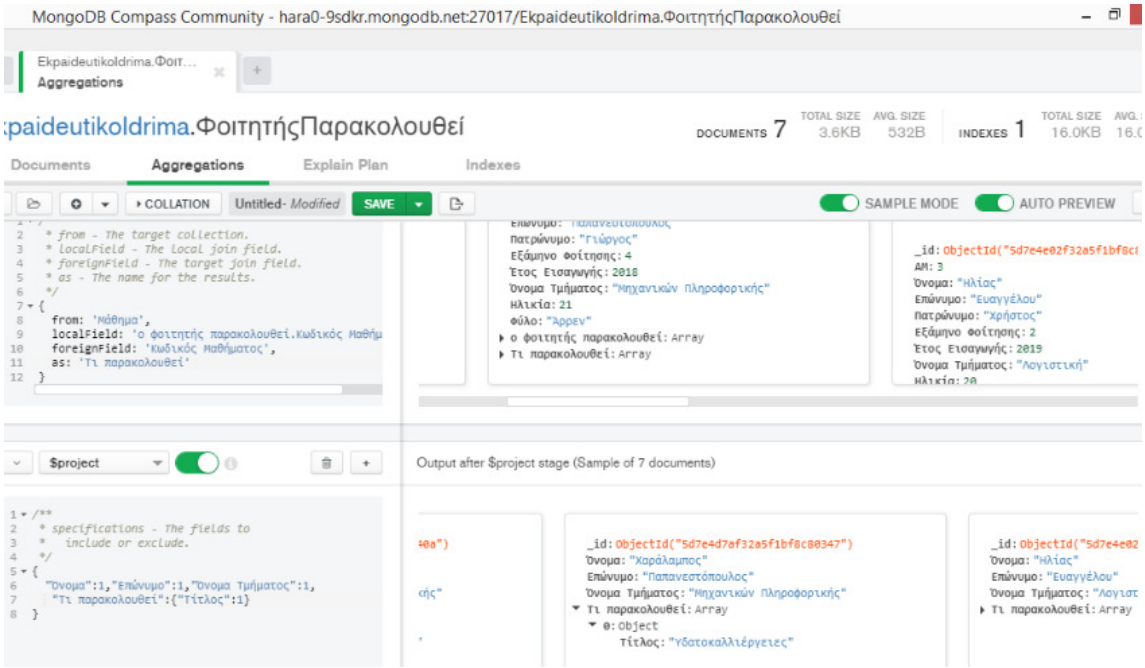
Εικόνα 83. Δεδομένα της Collection ΦοιτητήςΠαρακολουθεί



Εικόνα 84. Δεδομένα της Collection ΦοιτητήςΠαρακολουθεί

Στην συνέχεια, από την νέα Collection Φοιτητής Παρακολουθεί έγινε συνένωση (join) με την Collection Μάθημα. Για το λόγο αυτό στο πρώτο στάδιο χρησιμοποιήθηκε η εντολή $lookup, όπως φαίνεται και στην εικόνα 85.

Στην εικόνα 85 επίσης φαίνεται πως με την $project στο δεύτερο στάδιο του aggregation στα αποτελέσματα εμφανίζεται το επώνυμο, το όνομα και ο τίτλος του μαθήματος που παρακολουθεί ο κάθε φοιτητής



Εικόνα 85. Εμφάνιση των μαθημάτων κάθε φοιτητή και σε ποιο τμήμα ανήκει στην MongoDB Atlas

Η συνολική εντολή στα δύο στάδια είναι:

```
db.ΦοιτητήςΠαρακολουθεί.aggregate(
[{     $lookup: {
                    from: 'Μάθημα',
                    localField: 'ο φοιτητής παρακολουθεί.Κωδικός Μαθήματος',
                    foreignField: 'Κωδικός Μαθήματος',
                    as: 'Τι παρακολουθεί'
                  }},
        {$project: {
                        "Όνομα":1,"Επώνυμο":1,
                         "Τι παρακολουθεί":{"Τίτλος":1}
                          }
           }
]
 )
```

4. Συμπεράσματα

Στην εργασία αυτή επιχειρήθηκε μια σύγκριση των λειτουργιών μιας βάσης δεδομένων στην Access και μιας βάσης δεδομένων στο υπολογιστικό νέφος με την MongoDB Atlas.

Η βάση δεδομένων στην Access είναι μια σχεσιακή βάση δεδομένων και έχει σαν δομή τους πίνακες, οι οποίοι έχουν ένα προκαθορισμένο σχήμα. Το σχήμα αυτό δίνει μια λογική αναπαράσταση της βάσης και της σχέσης μεταξύ των πινάκων της βάσης. Η δημιουργία των βάσεων στην Access είναι εύκολη και αξιόπιστη μιας και χρησιμοποιεί πρωτεύοντα κλειδιά τα οποία απαγορεύουν τα διπλότυπα δεδομένα και εξασφαλίζουν την αξιοπιστία των δεδομένων.

Παρ' όλα αυτά παρουσιάζει δυσκολία σε περίπτωση που χρειάζεται να προστεθεί επιπλέον πεδίο ή πεδία σε έναν υπάρχον πίνακα. Έτσι για παράδειγμα, αν χρειαστεί να προστεθεί ένα επιπλέον τηλέφωνο σε μια εγγραφή ή και κάποιο άλλο επιπλέον πεδίο, θα δημιουργηθεί ένας δυσλειτουργικός πίνακας. Το πρόβλημα αυτό αντιμετωπίζεται δημιουργώντας έναν νέο πίνακα που θα έχει καταχωρισμένα τα τηλέφωνα και στη συνέχεια συσχετίζοντας αυτόν τον πίνακα με τον αρχικό. Όμως αυτό με τη σειρά του δημιουργεί πολλούς πίνακες και το όλο σχήμα γίνεται δυσνόητο για έναν χρήστη, ειδικά αν η βάση δεδομένων έχει πολλά δεδομένα και πολλές εγγραφές. Τέλος το να συνδυάσουμε και να ανακτήσουμε τόσα πολλά δεδομένα από τόσες διαφορετικές μεριές δεν είναι και τόσο αποτελεσματικό.

Η MongoDB Atlas είναι μια NoSQL βάση δεδομένων στο υπολογιστικό νέφος, η οποία δεν δεσμεύει χώρο στον υπολογιστή και παρέχει όλον τον εξοπλισμό που είναι απαραίτητος. Επίσης 'αναλαμβάνει' και τη συντήρηση της βάσης μέσω του υπολογιστικού νέφους.

Οι NoSQL βάσεις δεδομένων παρουσιάστηκαν εδώ και λίγες δεκαετίες, οπότε είναι σχετικά πρόσφατες αλλά έγιναν γνωστές λόγω του ότι έχουν καλύτερη απόδοση και καλύτερη κλιμάκωση (επεκτασιμότητα) όταν ο όγκος των δεδομένων είναι μεγάλος. Επίσης ο λόγος που είναι γνωστές είναι ότι δεν χρησιμοποιούν κάποιο προκαθορισμένο σχήμα (schema-less) και έτσι μπορούν να επεκταθούν πιο εύκολα. Ο τρόπος με τον οποίο αποθηκεύουν δεδομένα είναι πιο κοντά στον τρόπο που σκεφτόμαστε. Έτσι δεν χρησιμοποιούνται πίνακες οι οποίοι θα πρέπει να συνδυαστούν από πριν, αλλά το κάθε document περιέχει όλες τις πληροφορίες και η MongoDB Atlas αναλαμβάνει να βρει αυτό που της ζητάμε.

Το γεγονός αυτό δίνει κατά κάποιο τρόπο την ελευθερία να αποθηκεύονται όποια δεδομένα μας φαίνονται χρήσιμα για το κάθε document ξεχωριστά και τα οποία μπορεί διαφέρουν από document σε document, χωρίς να έχουν καθοριστεί ακριβώς από πριν, αλλά την ώρα που αυτά δημιουργούνται. Χωρίς το φόβο να καταστραφούν τα πάντα κάνοντας μια μικρή αλλαγή ή μια προσθήκη.

Στην Access είναι πολύ εύκολο να δημιουργήσουμε ερωτήματα τα οποία είναι απλά, ενώ στην MongoDB Atlas είναι πιο εύκολο να δημιουργήσουμε πολύπλοκα ερωτήματα. Στην MongoDB Atlas το σχήμα είναι τέτοιο που αντί να έχουμε πίνακες τα δεδομένα ενσωματώνονται ως εμφωλευμένα έγγραφα και αυτό έχει σαν αποτέλεσμα να είναι πιο εύκολη η αναζήτηση δεδομένων μέσα σε αυτά. Έτσι, αν για

παράδειγμα θελήσουμε να αποθηκεύσουμε πολλά τηλέφωνα σε ένα document αυτό γίνεται χρησιμοποιώντας εμφωλευμένα έγγραφα.

Από την άλλη η δομή της MongoDB Atlas παρέχει τη δυνατότητα ύπαρξης διπλότυπων δεδομένων και αυτό ίσως έχει κόστος στον χρόνο εύρεσης στοιχείων κατά τη διάρκεια ενός ερωτήματος.

Η συνολική εντολή σε ένα πολύπλοκο ερώτημα είναι πιο μικρή σε έκταση και πιο απλή να συνταχθεί με την MongoDB Atlas σε σχέση με την Access και την διατύπωση του ερωτήματος σε SQL.

Ένα πρόβλημα που υπάρχει στην MongoDB Atlas είναι τα joins μεταξύ Collections. Στην περίπτωση αυτή και ιδιαίτερα όταν οι Collections είναι πάνω από δύο, τότε η διαδικασία του join είναι πιο πολύπλοκη σε σχέση με την Access.

Επειδή η MongoDB Atlas δημιουργήθηκε σχετικά πρόσφατα από την ανάγκη διαχείρισης του μεγάλου όγκου δεδομένων που δημιουργούνται καθημερινά χρησιμοποιείται για την ανάλυση των δεδομένων αυτών καθημερινά και σε πραγματικό χρόνο. Βρίσκει εφαρμογή στα κινητά και στα μέσα κοινωνικής δικτύωσης προσφέροντας ασφάλεια στα δεδομένα.

Μέσω του cloud τα δεδομένα μπορούν να διαμοιράζονται ανά την υφήλιο και έτσι είναι κοντά σε κάθε χρήστη για γρήγορη πρόσβαση και ανά πάσα στιγμή.

Η επιλογή SQL ή NoSQL συστήματος διαχείρισης βάσης δεδομένων εξαρτάται από τις εφαρμογές που θα αλληλοεπιδράσουν με τα δεδομένα μας. Κατά τη γνώμη μου ένα NoSQL σύστημα διαχείρισης βάσεων δεδομένων μπορεί να ανταπεξέλθει καλύτερα στα σημερινά δεδομένα της εποχής. Η δομή οργάνωσης των δεδομένων είναι πιο “κοντά” στον τρόπο σκέψης του ανθρώπου και μπορεί να εφαρμοστεί τόσο σε περιπτώσεις που ο όγκος των δεδομένων είναι μικρός, όσο και σε περιπτώσεις με μεγάλο όγκο δεδομένων. Επιπλέον λόγω του sharding τα δεδομένα μας μπορούν να είναι πιο ασφαλή σε περίπτωση λάθους αλλά και σε περίπτωση κακόβουλης ενέργειας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] *www.ip.gr*. Ανάκτηση 8 6, 2019, από https://www.ip.gr/Hosting/Τι_είναι_μια_βάση_MySQL-209.html.

[2] *eclass.duth.gr*. Ανάκτηση από https://eclass.duth.gr/modules/document/file.php/OPE02214/DB_1.pdf .

[3] *www.ipet.gr*.Ανάκτηση 1 6, 2020, από http://www.ipet.gr/digitech2/index.php?option=com_content&task=view&id=88&Itemid=57.

[4] *wikipedia*. (2019, 10 16). Ανάκτηση από https://el.wikipedia.org/wiki/Σχεσιακή_βάση_δεδομένων

[5] *http://pdplab.it.uom.gr*. (2013, Ιανουάριος 17). Ανάκτηση από http://pdplab.it.uom.gr/teaching/ince_2e_gr/Text/C5/Relationaldatabases_2.htm.

[6] Λιβιεράτος, Κ. (2013, 7 9). *https://www.linuxinsider.gr/forum/programmatismos/8873/eisagogi-stis-nosql-baseis-dedomenon-hrisi-toy-cassandra*. Ανάκτηση από www.linuxinsider.gr.

[7] *www.predictiveanalyticstoday.com*. (2019). Ανάκτηση από https://www.predictiveanalyticstoday.com/top-free-commercial-sql-no-sql-cloud-databases/.

[8] wikipedia. (2019, 8 28). *https://en.wikipedia.org*. Ανάκτηση από https://en.wikipedia.org/wiki/Computer_cluster.

[9] *https://mytechnetknowhows.wordpress.com*. (2016, 5 31). Ανάκτηση 8 21, 2019, από https://mytechnetknowhows.wordpress.com/2016/05/31/cap-theorem-consistency-availability-and-partition-tolerance/.

[10] *en.wikipedia.org*. (2019, 8 2). Ανάκτηση από https://en.wikipedia.org/wiki/CAP_theorem.

[11] *el.wikipedia.org*. (2019, 6 28). Ανάκτηση 8 21, 2019, από el.wikipedia.org/wiki/ACID.

[12] *el.wikipedia.org*. (2019, 4 3). Ανάκτηση από https://el.wikipedia.org/wiki/Υπολογιστικό_νέφος.

[13] *en.wikipedia.org*. (2019, 9 22). Ανάκτηση από https://en.wikipedia.org/wiki/Key-value_database .

[14] *en.wikipedia.org*. (2019, 5 2). Ανάκτηση από https://en.wikipedia.org/wiki/Wide_column_store .

[15] *en.wikipedia.org*. (2019, 9 18). Ανάκτηση από https://en.wikipedia.org/wiki/Document-oriented_database .

[16] *en.wikipedia.org*. (2019, 9 10). Ανάκτηση από https://en.wikipedia.org/wiki/Graph_database

[17] *el.wikipedia.org*. (2019, 9 30). Ανάκτηση από https://el.wikipedia.org/wiki/Σημασιολογικός_Ιστός .

[18] *www.json.org*. (χ.χ.). Ανάκτηση από https://www.json.org/json-el.html.

[19] *www.w3schools.com*.Ανάκτηση από https://www.w3schools.com/nodejs/nodejs_intro.asp.

[20] *developer.marklogic.com*.Ανάκτηση από https://developer.marklogic.com/products/java.

[21] Violino, B. (2018, 3 9). *www.infoworld.com*. Ανάκτηση από https://www.infoworld.com/article/3260184/how-to-choose-the-right-nosql-database.html.

[22] *wikipedia*. (2019, 7 19). Ανάκτηση από https://el.wikipedia.org/wiki/Διαδίκτυο_των_πραγμάτων.

[23] *Radix DLT*. (2018, 7 1). Ανάκτηση από www.radixdlt.com/post/what-is-sharding/.

[24] *C2.gr*. Ανάκτηση 2019, από https://c2.gr/blog/load-balancing/.

[25] Νεκτάριος, Κ. (2019). Ανάκτηση από ocw.aoc.ntua.gr/modules/document/file.php/ECE132/Σημειώσεις%20Μαθήματος/MapReduce.pdf.

[26] Lohfink, C. (2014, 8 5). *www.slideshare.net/ChrisLohfink1/lsm-trees-37683710*. Ανάκτηση από www.slideshare.net/ChrisLohfink1/lsm-trees-37683710.

[27] Grigorik, I. (2012, 2 6). *www.igvita.com/2012/02/06/sstable-and-log-structured-storage-leveldb/*. Ανάκτηση από www.igvita.com/2012/02/06/sstable-and-log-structured-storage-leveldb/.

[28] Zurek, B. (2019). *www.sas.com/el_gr/insights/big-data/hadoop.html*. Ανάκτηση από www.sas.com/el_gr/insights/big-data/hadoop.html.

[29] *techopedia*. (2019). Ανάκτηση από www.techopedia.com/definition/30427/hadoop-common.

[30] *www.ibm.com*. Ανάκτηση 2019, από www.ibm.com/analytics/hadoop/hdfs.

[31] Margaret Rouse, C. S. (2018, 3). *https://searchdatamanagement.techtarget.com/definition/Apache-Hadoop-YARN-Yet-Another-Resource-Negotiator*. Ανάκτηση από https://searchdatamanagement.techtarget.com/definition/Apache-Hadoop-YARN-Yet-Another-Resource-Negotiator.

[32] *www.thegeekdiary.com*. (2019). Ανάκτηση από www.thegeekdiary.com/understanding-the-hadoop-mapreduce-framework/.

[33] Boyd, R. (2016, 3). *https://neo4j.com/*. Ανάκτηση από https://neo4j.com/developer/graph-database/.

[34] *wikipedia.org*. (2019, 8 18). Ανάκτηση από https://en.wikipedia.org/wiki/State_machine_replication.

[35] *www.arangodb.com*. Ανάκτηση από https://www.arangodb.com/docs/stable/ .

[36] *en.wikipedia.org*. (2019, 5). Ανάκτηση από https://en.wikipedia.org/wiki/Clusterpoint.

[37] *ipfs.io*. (2018, 10 8). Ανάκτηση από https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Clusterpoint.html.

[38] *cloud.ibm.com*. (2019). Ανάκτηση από https://cloud.ibm.com/docs/services/databases-for-postgresql?topic=cloud-databases-key-protect .

[39] *cloud.oracle.com*. (2019). Ανάκτηση από https://cloud.oracle.com/nosqldatabase.

[40] Murthy, K. (2016, 5 14). *https://www.slideshare.net/journalofinformix/bringing-sql-to-nosql-rich-declarative-query-for-nosql* . Ανάκτηση από https://www.slideshare.net/journalofinformix/bringing-sql-to-nosql-rich-declarative-query-for-nosql .

[41] *www.enterprisedb.com*. (2019). Ανάκτηση από https://www.enterprisedb.com/cloud/public-cloud-overview.

[42] *wikipedia*. (2019, 8 9). Ανάκτηση από https://en.wikipedia.org/wiki/EnterpriseDB.

[43] *aws.amazon.com*. (2019). Ανάκτηση από https://aws.amazon.com/quicksight/.

[44] *bestcyprushosting.com*. (2019). Ανάκτηση από https://bestcyprushosting.com/cloudwithme-amazon-web-services-coupons/.

[45] *en.wikipedia.org*. (2019, 10 13). Ανάκτηση από https://en.wikipedia.org/wiki/Bigtable.

[46] *cloud.google.com*. (2019, 6 10). Ανάκτηση από https://cloud.google.com/datastore/docs/concepts/overview.

[47] *en.wikipedia.org*. (2019, 8 11). Ανάκτηση από https://en.wikipedia.org/wiki/Cosmos_DB.