ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

NOSQL βάσεις δεδομένων για τη διαχείριση γράφων. Διερεύνηση-Εγκατάσταση και απλή εφαρμογή

ΓΙΑΝΝΗΣ ΜΠΑΜΠΑΡΑΚΟΣ ΑΜ:2300

ΓΙΩΡΓΟΣ ΑΔΑΜ ΑΜ:2266

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΒΑΣΙΛΕΙΟΣ ΤΑΜΠΑΚΑΣ

ΠΕΡΙΛΗΨΗ

Οι βάσεις δεδομένων από τη στιγμή που εμφανίστηκαν υποστηρίζουν καθημερινά πολλές ανθρώπινες δραστηριότητες, οι οποίες αν γίνονταν χειρωνακτικά θα απαιτούσαν πολύ περισσότερο χρόνο και κόπο.

Όμως με την αύξηση των δεδομένων που έπρεπε να καταχωρούνται σε αυτές, την εμφάνιση των κοινωνικών δικτύων και του **IoT** τα οποία αύξησαν το βαθμό διασύνδεσης μεταξύ όλων των πραγμάτων, διαπιστώθηκαν οι αδυναμίες στις σχεσιακές βάσεις δεδομένων.

Τη λύση ήρθαν να δώσουν οι NoSQLβάσεις δεδομένων οι οποίες με το πιο χαλαρό μοντέλο δεδομένων που διαθέτουν μπορούν αν επεξεργαστούν τα δεδομένα ταχύτερα και πιο εύκολα. Ιδιαίτερα, οι βάσεις δεδομένων γράφων προσφέρουν ένα σύνολο από επιπλέον δυνατότητες που αν τις χρησιμοποιήσουμε θα μπορούσαμε να διευκολύνουμε τη ζωή και την εργασία πολλών ανθρώπων.

Λέξεις-κλειδιά: Βάσεις Δεδομένων, NoSQL, Βάσεις Δεδομένων γράφων, Neo4J, Cypher, Python

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

Σκοπός της εργασίας είναι να μελετηθούν οι NoSQLβάσεις δεδομένων γράφων και οι δυνατότητες που προσφέρουν σε σύγκριση με τις σχεσιακές βάσεις δεδομένων. Για το σκοπό αυτό, πέρα από τη μελέτη στο θεωρητικό επίπεδο θα υλοποιηθεί και μια εφαρμογή μέσω της οποίας θα μελετήσουμε τον τρόπο χρήσης τους.

Στο Κεφάλαιο 1 θα περιγράψουμε το απαραίτητο θεωρητικό υπόβαθρο για να μπορέσουμε να κατανοήσουμε τις NoSQLβάσεις δεδομένων. Θα ξεκινήσουμε με μια σύντομη ιστορική αναδρομή στις βάσεις δεδομένων και μια αναφορά στις σχεσιακές βάσεις. Κατόπιν θα περιγράψουμε τις NoSQLβάσεις και την αναγκαιότητα που ήρθαν για να καλύψουν, τις βάζεις δεδομένων γράφων τις οποίες θα συγκρίνουμε με τις σχεσιακές. Τέλος θα αναφερθούμε στη Βάση δεδομένων Neo4jκαι τη γλώσσα ερωτημάτων cypher, κάνοντας μια μικρή ανάλυσή της και παρουσιάζοντας ορισμένα παραδείγματα.

Στο Κεφάλαιο 2 θα παρουσιάσουμε την εφαρμογή που υλοποιήσαμε για να μελετήσουμε τον τρόπο χρήσης των NoSQLβάσεων δεδομένων. Θα ξεκινήσουμε από το στάδιο του σχεδιασμού της εφαρμογής μέχρι την υλοποίησή της και θα παρουσιάσουμε τον τρόπο χρήσης της.

Στο Κεφάλαιο 3 θα αναφέρουμε τα συμπεράσματα της εργασίας μας και τις προτάσεις για μελλοντικές επεκτάσεις – λειτουργίες.

ΚΕΦΑΛΑΙΟ 1. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΙΣΗ

1.1 Εισαγωγή

Στο κεφάλαιο θα παρουσιάσουμε όλες τις απαραίτητες θεωρητικές γνώσεις για να μπορέσουμε να προχωρήσουμε στη συνέχεια στην υλοποίηση της εφαρμογής.
Θα ξεκινήσουμε με μια ιστορική αναδρομή στις βάσεις δεδομένων και θα συνεχίσουμε με μια συνοπτική παρουσίαση των σχεσιακών βάσεων δεδομένων. Στη συνέχεια θα παρουσιάσουμε αναλυτικά τις NoSQLβάσεις δεδομένων, τα χαρακτηριστικά τους και τις κατηγορίες τους. Κατόπιν θα αναφερθούμε στην κατηγορία των NoSQLβάσεων δεδομένων γράφων.
Έχοντας ολοκληρώσει τη θεωρητική παρουσίαση, προχωράμε στη σύγκριση των σχεσιακών βάσεων δεδομένων με τις βάσεις δεδομένων διαχείρισης γράφων. Κατόπιν παρουσιάζουμε εν συντομία τη βάση δεδομένων γράφων Neo4jκαι τη γλώσσα ερωτημάτων Cypher.

1.2 Βάσεις Δεδομένων

1.2.1 Ιστορική Αναδρομή

Τα πρώτα βήματα (πολύ πριν εμφανιστούν οι ηλεκτρονικοί υπολογιστές) για τη δημιουργία των πρώιμων βάσεων δεδομένων έγιναν με την δημιουργία των πρώτων αρχείων (κυβερνητικά, επιχειρηματικά και ιατρικά) και των βιβλιοθηκών. Την εποχή εκείνη ανέκυψε τα προβλήματα της διαχείρισης (προσθήκης, διαγραφής και ενημέρωσης) και αποθήκευσης του αρχείου. Όταν εμφανίστηκαν οι υπολογιστές προσέφεραν ένα πλήθος από λύσεις και εναλλακτικές.
Τη δεκαετία του 1960 με τη διαθεσιμότητα αποθήκευσης άμεσης πρόσβασης αναπτύχθηκαν δύο κύρια μοντέλα δεδομένων - το μοντέλο CODASYL (Conference on Data System Language) και το ιεραρχικό μοντέλο IMS (Information Management System).Οι εφαρμογές είχαν πρόσβαση σε δεδομένα ακολουθώντας δείκτες από τη μία εγγραφή στην άλλη. Η αποθήκευση ήταν άμεσα εξαρτημένη από τον τύπο των δεδομένων που αποθηκεύονταν. Έτσι, η προσθήκη ενός επιπλέον πεδίου στη βάση δεδομένων απαιτούσε την επανεγγραφή ολόκληρου του σχήματος.

Το σχεσιακό μοντέλο βάσης δεδομένων σχεδιάστηκε από τον E.F. Codd το 1970. Αυτό το μοντέλο υποστήριζε ότι οι εφαρμογές πρέπει να αναζητούν δεδομένα βάσει περιεχομένου και όχι όπως γινόταν μέχρι εκείνη τη στιγμή να ακολουθούν συνδέσμους. Το μοντέλο που πρότεινε είχε δύο βασικά συστατικά το Instance, ένα πίνακα με σειρές και στήλες και το σχήμα, το οποίο καθορίζει τη δομή, συμπεριλαμβανομένου του ονόματος της σχέσης, του ονόματος και του τύπου κάθε στήλης. Στο μοντέλο του, το σχήμα της βάσης δεδομένων, αποσυνδέεται από την αποθήκευση των δεδομένων στο φυσικό μέσο.

Το 1976 προτάθηκε από τον Chenένα νέο μοντέλο βάσης δεδομένων που ονομάζεται Entity-Relationship, ή ER. Αυτό το μοντέλο επέτρεψε στους σχεδιαστές να επικεντρωθούν περισσότερο στα δεδομένα παρά στη λογική δομή του πίνακα.

Μεταξύ του 1960 και του 1980 οι βάσεις δεδομένων γνώρισαν μεγάλη εξέλιξη και παρουσιάστηκαν τρία μοντέλα για την οργάνωση των δεδομένων - το μοντέλο δικτύου, το ιεραρχικό μοντέλο και το σχεσιακό μοντέλο. Το 1985 αναπτύχθηκε και η έννοια της αντικειμενοστραφούς βάσης δεδομένων.

Το 1998 ο Carlo Strozzi χρησιμοποίησε τον όρο NoSQL για να ονομάσει την ελαφριά, βάση δεδομένων ανοιχτού κώδικα που είχε κατασκευάσει και δεν υποστήριζε την τυπική διεπαφή SQL. Αργότερα ο Eric Evans, υπάλληλος της Rackspace, επανάφερε τον όρο NoSQL προσπαθώντας να βαφτίσει το σύνολο των μη σχεσιακών, κατανεμημένων συστημάτων δεδομένων που είχαν ξεκινήσει και είχαν κάνει την εμφάνισή τους και που συχνά δεν ήταν συμβατές με τις ιδιότητες ACID[2].

1.2.2 Σχεσιακές Βάσεις Δεδομένων

Η σχεσιακή βάση δεδομένων είναι μια συλλογή δεδομένων με προκαθορισμένες σχέσεις μεταξύ τους. Αυτά τα στοιχεία οργανώνονται σαν ένα σύνολο πινάκων με στήλες και σειρές. Οι πίνακες χρησιμοποιούνται για να κρατήσουν πληροφορίες σχετικά με τα αντικείμενα που θα αναπαρασταθούν στη βάση δεδομένων. Κάθε στήλη σε έναν πίνακα περιέχει ένα συγκεκριμένο είδος δεδομένων και ένα πεδίο αποθηκεύει την πραγματική τιμή ενός χαρακτηριστικού. Οι σειρές στον πίνακα αντιπροσωπεύουν μια συλλογή σχετικών τιμών ενός αντικειμένου ή οντότητας. Κάθε σειρά σε έναν πίνακα θα μπορούσε να επισημανθεί με ένα μοναδικό αναγνωριστικό που ονομάζεται πρωτεύον κλειδί και σειρές μεταξύ πολλών πινάκων μπορούν να δημιουργήσουν σχέσεις χρησιμοποιώντας ξένα κλειδιά. Αυτά τα δεδομένα είναι

προσβάσιμα με πολλούς διαφορετικούς τρόπους χωρίς να αναδιοργανώνονται οι ίδιοι οι πίνακες βάσης δεδομένων.

Η κύρια διεπαφή που χρησιμοποιείται για την επικοινωνία με τις σχεσιακές βάσεις δεδομένων είναι η SQL (Structured Query Language). Η SQL έγινε πρότυπο του Αμερικανικού Ινστιτούτου Εθνικών Προτύπων (ANSI) το 1986. Το πρότυπο ANSI SQL υποστηρίζεται από όλους τους δημοφιλείς σχεσιακούς μηχανισμούς βάσης δεδομένων και ορισμένοι από αυτούς το έχουν επεκτείνει για να υποστηρίζουν συγκεκριμένη λειτουργικότητα που προσφέρουν. Η SQL χρησιμοποιείται και για τις τέσσερις λειτουργίες CRUD (create, read, update, delete).

Οι σχεσιακές βάσεις δεδομένων υποστηρίζουν συναλλαγές (transactions). Μια συναλλαγή είναι μία ή περισσότερες εντολές SQL που εκτελούνται ως ακολουθία λειτουργιών που σχηματίζουν μία λογική μονάδα εργασίας. Οι εντολές που περιλαμβάνονται σε μια συναλλαγή είτε θα εκτελεστούν όλες επιτυχώς ή θα αποτύχουν όλες. Όλες οι συναλλαγές είναι συμβατές με τις ιδιότητες ACID(θα παρουσιαστούν στην παράγραφο 1.2.3)[1].

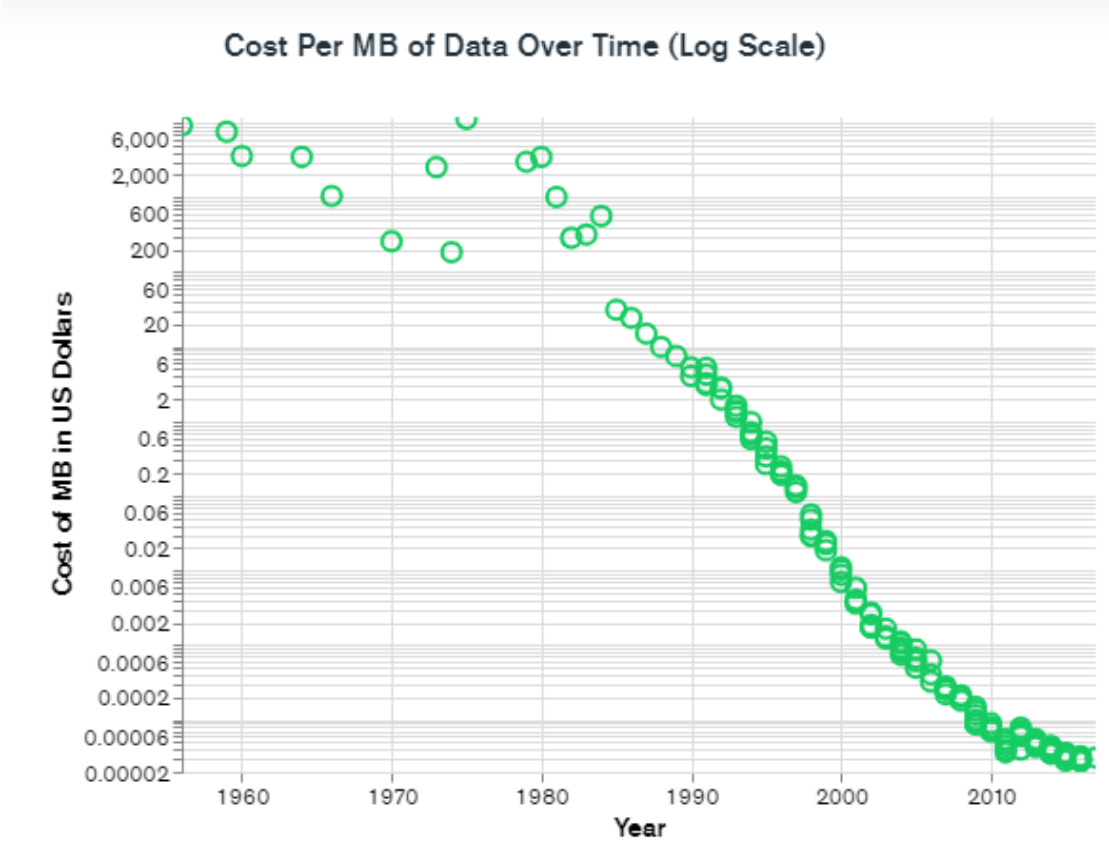
1.2.3 NoSQL Βάσεις Δεδομένων

Όταν χρησιμοποιούμε τον όρο «βάση δεδομένων NoSQL», συνήθως αναφερόμαστε σε οποιαδήποτε μη σχεσιακή βάση δεδομένων. Η βάση δεδομένων NoSQL είναι ένα μη σχεσιακό σύστημα διαχείρισης δεδομένων, το οποίο δεν απαιτεί κάποιο σταθερό σχήμα για την αποθήκευση των δεδομένων. Αποφεύγει τις συσχετίσεις (join) και μπορεί εύκολα να κλιμακωθεί. Ο κύριος σκοπός της χρήσης μιας βάσης δεδομένων NoSQL είναι για κατανεμημένα συστήματα δεδομένων με τεράστιες ανάγκες αποθήκευσης δεδομένων. Η NoSQL χρησιμοποιείται για Big *Data* και εφαρμογές ιστού σε πραγματικό χρόνο. Για παράδειγμα, εταιρείες όπως το Twitter, το Facebook και η Google συλλέγουν terabyte δεδομένων και τα αποθηκεύουν σε NoSQL βάσεις δεδομένων [6].

Μια κοινή παρανόηση είναι ότι οι βάσεις δεδομένων NoSQL ή οι μη σχεσιακές βάσεις δεδομένων δεν αποθηκεύουν καλά τα δεδομένα των σχέσεων. Οι βάσεις δεδομένων NoSQL μπορούν να αποθηκεύουν δεδομένα σχέσεων, απλά τα αποθηκεύουν διαφορετικά από τις σχεσιακές βάσεις δεδομένων. Στην πραγματικότητα, σε σύγκριση με τις βάσεις δεδομένων SQL, πολλοί θεωρούν ότι τα δεδομένα σχέσεων μοντελοποίησης σε βάσεις δεδομένων NoSQL είναι ευκολότερα

από ό, τι στις βάσεις δεδομένων SQL, επειδή τα σχετικά δεδομένα δεν χρειάζεται να χωρίζονται μεταξύ πινάκων. Τα μοντέλα δεδομένων NoSQL επιτρέπουν την ένθεση σχετικών δεδομένων σε μία δομή δεδομένων.

Ο Carl Strozzi παρουσίασε την έννοια της NoSQL το 1998 για την ελαφριά σχεσιακή βάση δεδομένων ανοιχτού κώδικα που είχε υλοποιήσει. Κατόπιν το 2000 εμφανίστηκε η βάση δεδομένων γράφων Neo4j.Την εποχή εκείνη παρατηρήθηκε και μια σημαντική μείωση του κόστους αποθήκευσης. Πέρασαν πλέον οι μέρες που χρειάζονταν να δημιουργήσουμε ένα περίπλοκο, δύσκολο στη διαχείριση μοντέλο δεδομένων, με σκοπό τη μείωση της αναπαραγωγής δεδομένων. Οι προγραμματιστές (και όχι ο χώρος αποθήκευσης) καθόριζαν το κύριο κόστος ανάπτυξης λογισμικού. Οι δυνατότητες που προσέφεραν οι βάσεις δεδομένων NoSQL σε συνδυασμό με τη μείωση του κόστους αποθήκευσης, έδωσε μια επιπλέον ώθηση στην ενασχόληση όλων και περισσότερων ανθρώπων για τη βελτιστοποίησή τους με αποτέλεσμα την εμφάνιση νέων προϊόντων και τη βελτίωση των δυνατοτήτων τους. Όλα τα παραπάνω οδήγησαν στο να αποκτήσουν οι NoSQL βάσεις δεδομένων ένα τμήμα της αγοράς.



Εικόνα 1 Μείωση του κόστους αποθήκευσης με την πάροδο του χρόνου[7]

Καθώς το κόστος αποθήκευσης μειωνόταν σημαντικά και γρήγορα, αυξήθηκε και ο όγκος των δεδομένων που έπρεπε να αποθηκεύονται. Αυτά τα δεδομένα ήταν σε όλα τα σχήματα και μεγέθη - δομημένα, ημιδομημένα και πολυμορφικά - και ο καθορισμός του σχήματος εκ των προτέρων έγινε σχεδόν αδύνατος. Οι βάσεις δεδομένων NoSQL επιτρέπουν στους προγραμματιστές να αποθηκεύουν τεράστιες ποσότητες μη δομημένων δεδομένων, δίνοντάς τους μεγάλη ευελιξία.

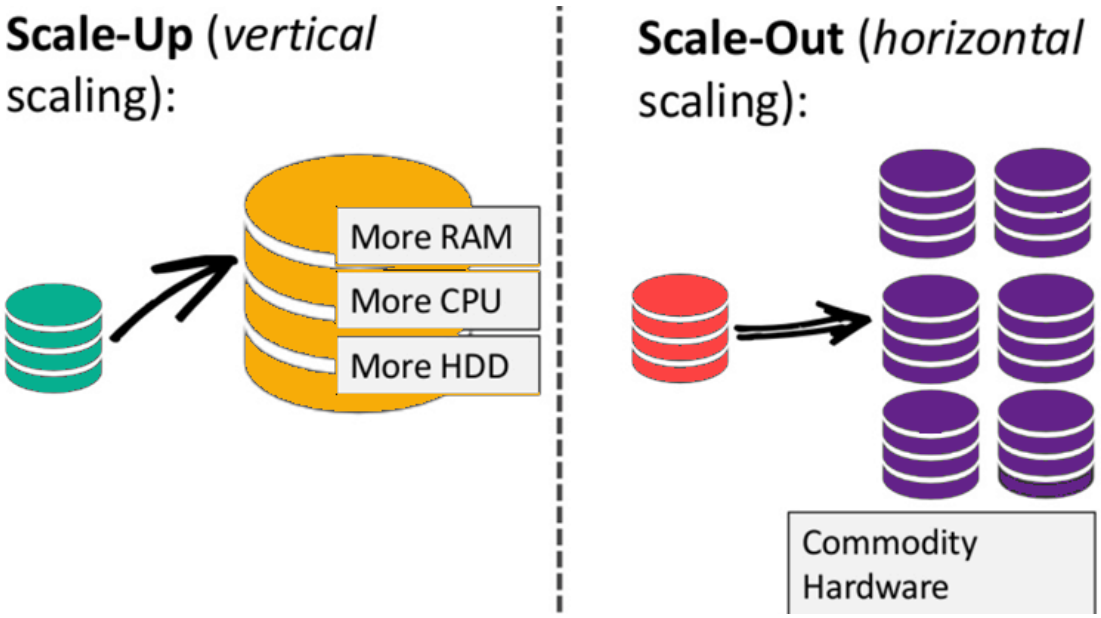
Επιπλέον, σημαντικές αλλαγές γίνονταν και στον τρόπο της ανάπτυξης του λογισμικού με την εμφάνιση του Agile Manifesto, με αποτέλεσμα οι μηχανικοί λογισμικού να επανεξετάζουν τις διαδικασίες και τον τρόπο ανάπτυξής του. Διαπίστωσαν την ανάγκη ταχείας προσαρμογής στις μεταβαλλόμενες απαιτήσεις με άμεσες αλλαγές στο λογισμικό τους που θα έφταναν μέχρι το μοντέλο της βάσης δεδομένων. Οι βάσεις δεδομένων NoSQL τους έδωσαν αυτήν την ευελιξία.

Ταυτόχρονα, αυξήθηκε η δημοτικότητά του cloud computing και οι προγραμματιστές άρχισαν να χρησιμοποιούν δημόσια cloud για να φιλοξενήσουν τις εφαρμογές και τα δεδομένα τους. Ήθελαν τη δυνατότητα να διανέμουν δεδομένα σε πολλούς διακομιστές και περιοχές για να κάνουν τις εφαρμογές τους ανθεκτικές, να επεκτείνουν αντί να αναβαθμίζουν. Ορισμένες βάσεις δεδομένων NoSQL όπως η MongoDB παρείχαν αυτές τις δυνατότητες [7].

Τί εννοούμε όμως με τους όρους επέκταση και αναβάθμιση; Όσο το πλήθος των δεδομένων που αποθηκεύονται σε μια βάση δεδομένων μεγαλώνει, τόσο ο χρόνος απόκρισής της γίνεται πιο αργός.

Για να επιλύσουμε αυτό το πρόβλημα, θα μπορούσαμε να «αναβαθμίσουμε scale - up» τα συστήματά μας αναβαθμίζοντας το υπάρχον υλικό μας. Αυτή η διαδικασία είναι ακριβή.

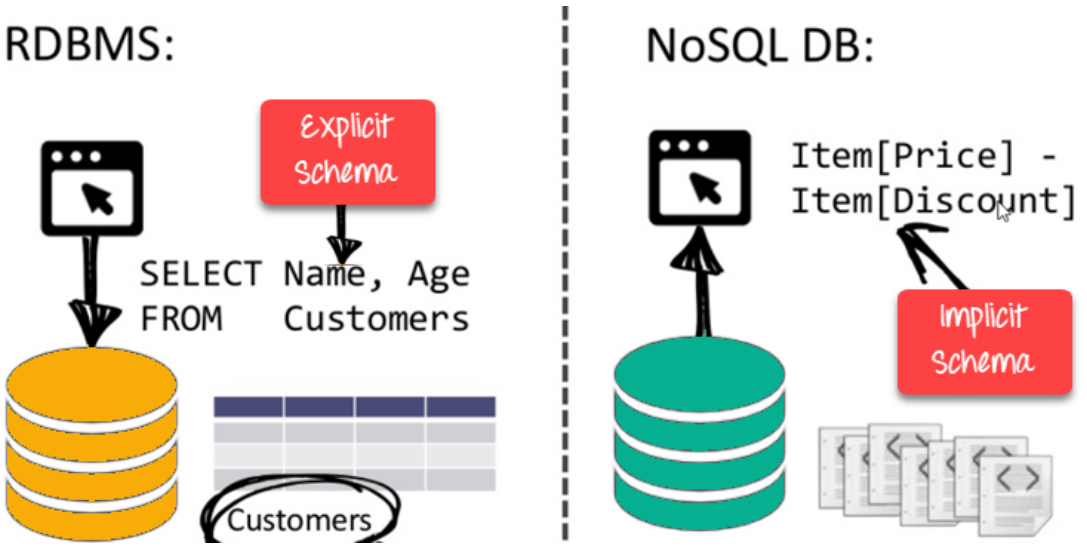
Η εναλλακτική λύση για αυτό το ζήτημα είναι η διανομή φορτίου βάσης δεδομένων σε πολλούς κεντρικούς υπολογιστές όποτε αυξάνεται το φορτίο. Αυτή η μέθοδος είναι γνωστή ως «επέκταση – scale out».



Εικόνα 2Αναβάθμιση – Επέκταση[6]

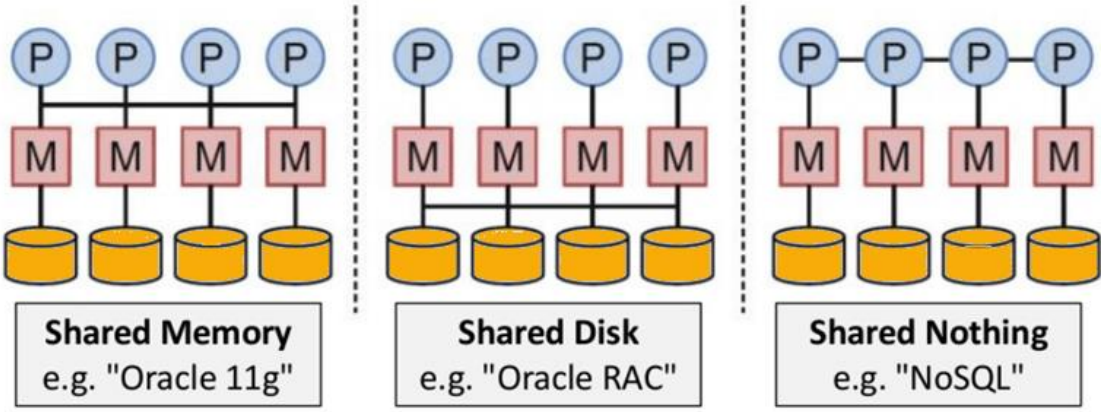
Ας δούμε τώρα τα πιο σημαντικά χαρακτηριστικά των NoSQL βάσεων δεδομένων:

- Μη σχεσιακές
  - Οι βάσεις δεδομένων NoSQL δεν ακολουθούν ποτέ το σχεσιακό μοντέλο.
  - Δεν παρέχουν πίνακες με επίπεδες εγγραφές σταθερής στήλης.
  - Περιέχουν αυτόνομα σύνολα-οντότητες ή BLOB (Binary Large OBject - μια συλλογή δυαδικών δεδομένων που αποθηκεύονται ως μία οντότητα)
  - Δεν απαιτείται αντιστοίχιση μεταξύ των φυσικών αντικειμένων και των σχεσιακών πινάκων καθώς και κανονικοποίηση των δεδομένων.
  - Δεν υπάρχουν σύνθετες λειτουργίες όπως, συνδέσεις ακεραιότητας αναφοράς, ACID κ.λπ.
- Χωρίς σχήμα
  - Οι βάσεις δεδομένων NoSQL είναι είτε χωρίς σχήμα είτε έχουν χαλαρά σχήματα.
  - Δεν απαιτείται κανένας ορισμός του σχήματος των δεδομένων.
  - Προσφέρει ετερογενείς δομές δεδομένων στο ίδιο μοντέλο.



Εικόνα 3 Οι NoSQLείναι Schemafree[6]

- Απλό API
    - Προσφέρει εύχρηστες διεπαφές για αποθήκευση και εκτέλεση ερωτημάτων στα δεδομένα.
    - Τα API επιτρέπουν μεθόδους χειρισμού και επιλογής δεδομένων χαμηλού επιπέδου.
    - Πρωτόκολλα βασισμένα σε κείμενο που χρησιμοποιούνται κυρίως με HTTP REST με JSON.
    - Βάσεις δεδομένων με δυνατότητα χρήσης για διαδικτυακές εφαρμογές.
- Διανέμονται
    - Πολλαπλές βάσεις δεδομένων NoSQL μπορούν να εκτελεστούν με κατανεμημένο τρόπο
    - Προσφέρει δυνατότητες αυτόματης κλιμάκωσης και fail-over
    - Συχνά η ιδέα του ACID μπορεί να θυσιαστεί στο βωμό της επεκτασιμότητας και της απόδοσης
    - Συνήθως δεν υπάρχει σύγχρονη αναπαραγωγή μεταξύ κατανεμημένων κόμβων, υπάρχει ασύγχρονη πολλαπλή αντιγραφή, peer-to-peer, HDFS Replication
    - Παρέχει μόνο τελική συνέπεια
    - Αρχιτεκτονική Shared Nothing. Αυτό επιτρέπει λιγότερο συντονισμό και υψηλότερη κατανομή [6].

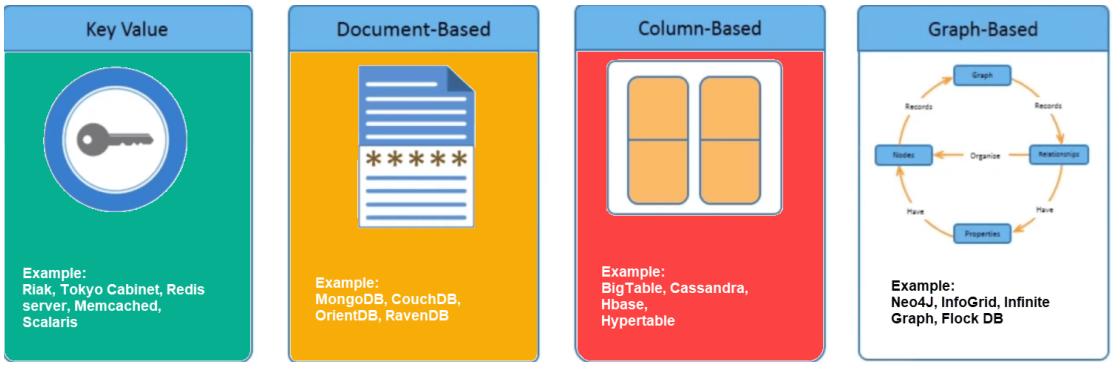


Εικόνα4Αρχιτεκτονική SharedNothing[6]

Οι βάσεις δεδομένων NoSQL κατηγοριοποιούνται κυρίως σε τέσσερις τύπους:

- Key-value Pair based
- Column based
- Document-oriented
- Graph based

Κάθε κατηγορία έχει μοναδικά χαρακτηριστικά και περιορισμούς. Καμία από τις παραπάνω κατηγορίες βάσεις δεδομένων δεν είναι καλύτερη για την επίλυση όλων των προβλημάτων. Οι υπεύθυνοι σχεδίασης πρέπει να επιλέξουν τη βάση δεδομένων με βάση τις ανάγκες των προς σχεδίασης συστημάτων.



Εικόνα 4 Κατηγορίες NoSQLβάσεων δεδομένων[6]

- Key Value Pair Based: Τα δεδομένα αποθηκεύονται σε ζεύγη κλειδιών / τιμών. Οι συγκεκριμένες βάσεις δεδομένων έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να χειρίζονται πολλά δεδομένα και βαρύ φορτίο. Οι βάσεις δεδομένων αποθήκευσης ζεύγους κλειδιού-τιμής αποθηκεύουν δεδομένα σαν ένα πίνακα κατακερματισμού όπου κάθε κλειδί είναι μοναδικό και η τιμή μπορεί να είναι

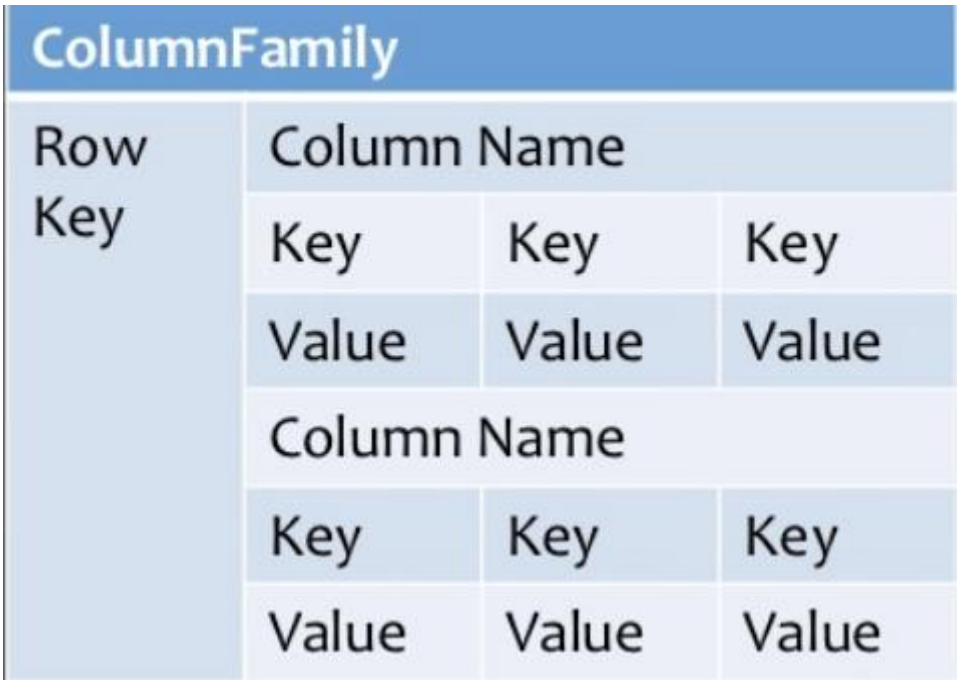
JSON, BLOB (Binary Large OBject), συμβολοσειρά κ.λπ. Μια τιμή μπορεί συνήθως να ανακτηθεί μόνο με αναφορά του κλειδιού της, οπότε τα ερωτήματα για την ανάκτηση των τιμών είναι απλά. Οι βάσεις δεδομένων κλειδιού-τιμής είναι ιδανικές για περιπτώσεις χρήσης όπου πρέπει να αποθηκευτούν μεγάλες ποσότητες δεδομένων, αλλά δεν χρειάζεται η εκτέλεση σύνθετων ερωτημάτων για την ανάκτησή τους. Αυτό το είδος βάσης δεδομένων NoSQL χρησιμοποιείται κυρίως σε συλλογές, λεξικά, σχετιζόμενους πίνακες ,για το περιεχόμενο του καλαθιού αγορών κ.λπ. Το Redis, το Dynamo, το Riak είναι μερικά παραδείγματα της συγκεκριμένης κατηγορίας, τα οποία βασίζονται κυρίως στο Dynamo της Amazon.

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Εικόνα 5Κατηγορία Key-valuepair[6]

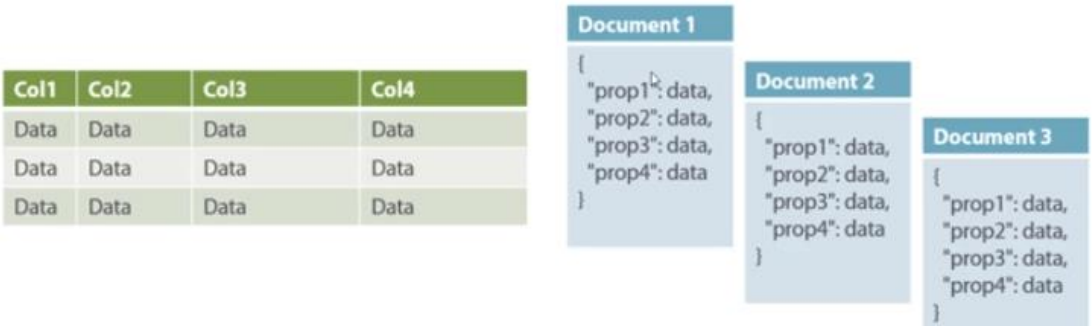
- Column based: οι συγκεκριμένες βάσεις δεδομένων χρησιμοποιούνται για την αποθήκευση δεδομένων σε πίνακες, γραμμές και δυναμικές στήλες. Οι συγκεκριμένες βάσεις παρέχουν μεγάλη ευελιξία σε σχέση με τις σχεσιακές βάσεις δεδομένων, επειδή κάθε γραμμή δεν απαιτείται να έχει τις ίδιες στήλες με τις άλλες γραμμές. Μπορεί να θεωρηθούν ότι είναι δισδιάστατες βάσεις δεδομένων κλειδιού-τιμής. Οι συγκεκριμένες βάσεις δεδομένων είναι ιδανικές όταν πρέπει να αποθηκευτούν μεγάλες ποσότητες δεδομένων και μπορούμε να προβλέψουμε ποια θα είναι τα μοτίβο ερωτημάτων που θα υποβληθούν. Χρησιμοποιούνται συνήθως για την αποθήκευση δεδομένων Internet of

Things και δεδομένων προφίλ χρήστη. Η Cassandra και η HBase είναι δύο από τις πιο δημοφιλείς βάσεις δεδομένων της κατηγορίας αυτής.



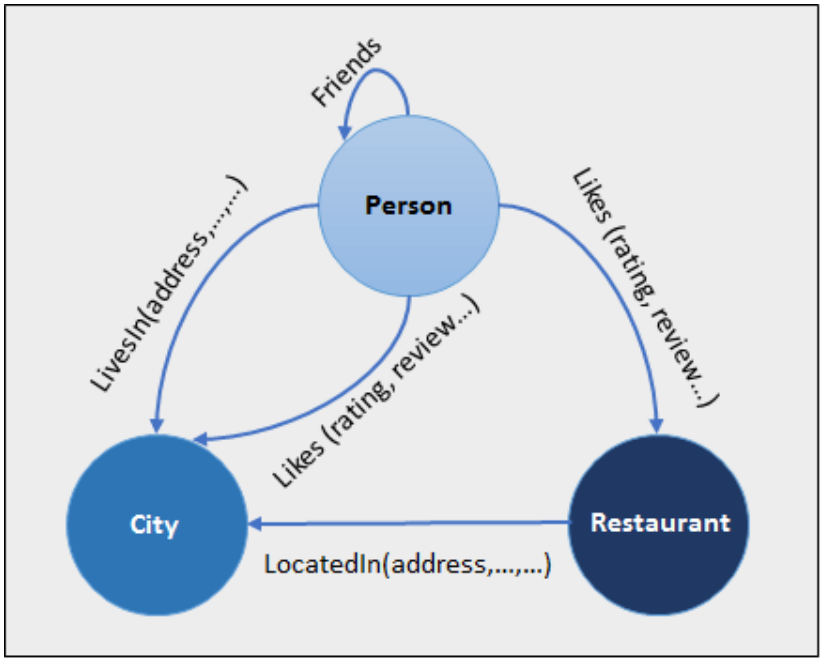
Εικόνα 6 Κατηγορία Columnbased[6]

- Document-Oriented: αποθηκεύουν τα δεδομένα σε έγγραφα παρόμοια με αντικείμενα JSON. Κάθε έγγραφο περιέχει ζεύγη πεδίων και τιμών. Οι τιμές μπορεί συνήθως να είναι μια ποικιλία τύπων, όπως συμβολοσειρές, αριθμοί, booleans, πίνακες ή αντικείμενα και οι δομές τους ευθυγραμμίζονται συνήθως με αντικείμενα που υλοποιούν οι προγραμματιστές στον κώδικά τους. Λόγω της ποικιλίας των τύπων τιμών πεδίου και των ισχυρών γλωσσών που χρησιμοποιούνται για τη δόμηση των ερωτημάτων, οι βάσεις δεδομένων εγγράφων είναι ιδανικές για μια μεγάλη ποικιλία περιπτώσεων χρήσης και μπορούν να χρησιμοποιηθούν ως βάση δεδομένων γενικού σκοπού. Μπορούν να κλιμακωθούν οριζόντια για να φιλοξενήσουν μεγάλους όγκους δεδομένων. Οι πιο δημοφιλείς βάσεις δεδομένων της κατηγορίας είναι η MongoDB, Amazon SimpleDB, CouchDB κ.λπ. Στην εικόνα που ακολουθεί μπορούμε να δούμε ένα παράδειγμα με την αντιστοίχιση του τρόπου αποθήκευσης σε μια σχεσιακή βάση δεδομένων (αριστερά) και το μετασχηματισμό της σε Document Oriented (αριστερά).



Εικόνα 7 Σχεσιακές και DocumentOriented[6]

- Graph based: μια βάση δεδομένων γράφων αποθηκεύει οντότητες καθώς και τις σχέσεις μεταξύ αυτών των οντοτήτων. Η οντότητα αποθηκεύεται ως κόμβος με τη σχέση σαν ακμή που ξεκινά ή καταλήγει σε αυτόν. Μια ακμή δίνει μια σχέση μεταξύ των κόμβων. Κάθε κόμβος και ακμή έχουν ένα μοναδικό αναγνωριστικό. Σε σύγκριση με μια σχεσιακή βάση δεδομένων όπου οι πίνακες είναι χαλαρά συνδεδεμένοι, μια βάση δεδομένων γράφων είναι πολλαπλού σχεσιακού χαρακτήρα. Οι βάσεις δεδομένων γράφων υπερέχουν σε περιπτώσεις χρήσης όταν πρέπει να αναζητήσετε μοτίβο όπως κοινωνικά δίκτυα, εντοπισμό απάτης και μηχανές προτάσεων. Οι Neo4J, Infinite Graph, OrientDB, FlockDB είναι μερικές δημοφιλείς βάσεις δεδομένων στην κατηγορία αυτή [7].



Εικόνα9Graphbased[6]

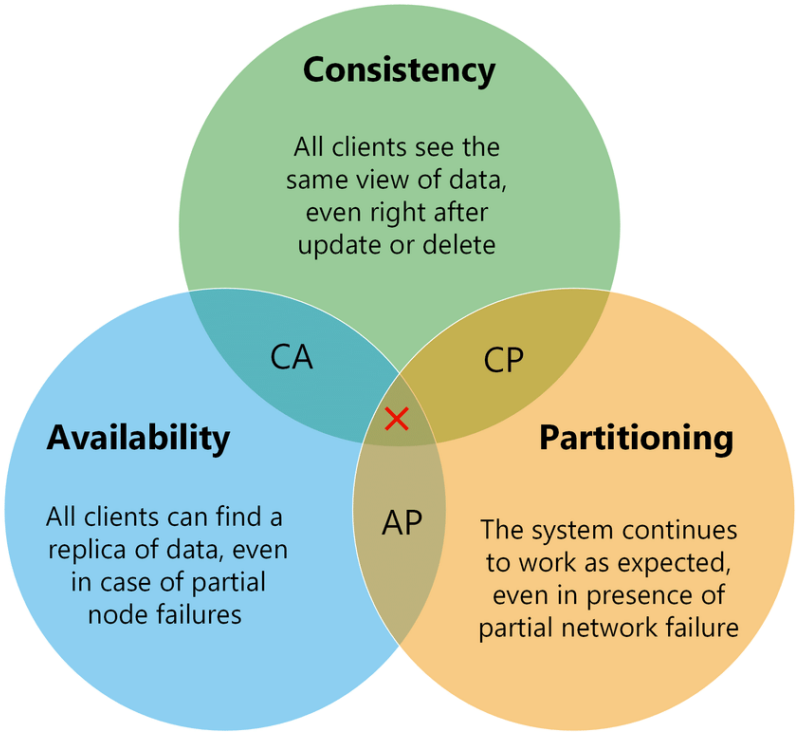
Στο σημείο αυτό πιστεύουμε ότι θα πρέπει να αναφερθούμε στον τρόπο με τον οποίο εκτελούνται οι δοσοληψίες σε μια NoSQL βάση δεδομένων και τη σχέση της με τις SQL. Οι σχεσιακές βάσεις δεδομένων, για να προστατέψουν την ακεραιότητα των δεδομένων, χρησιμοποιούν την έννοια της δοσοληψίας, η οποία παρέχει τέσσερις εγγυήσεις που μπορούν να συνοψιστούν στο ακρωνύμιο ACID.

- **Atomicity** (Ατομικότητα): όλες οι λειτουργίες θα επιτύχουν ή θα αποτύχουν ως μία μονάδα.
- **Consistency** (Συνοχή): οι λειτουργίες δεν θα παραβιάζουν συγκεκριμένους περιορισμούς που έχουμε ορίσει για τα δεδομένα στο σύνολό τους.
- **Isolation** (Απομόνωση): κάθε λειτουργία αποκρύπτεται από την προβολή έως ότου ολοκληρωθεί ολόκληρη η συναλλαγή.
- **Durability** (Ανθεκτικότητα): όλες οι αλλαγές στα δεδομένα διατηρούνται με ασφάλεια.

Για παράδειγμα αν θέλω να μεταφέρω ένα χρηματικό ποσό από ένα λογαριασμό σε ένα άλλο η δοσοληψία περιλαμβάνει δύο λειτουργίες, πρώτα την ανάληψη των χρημάτων από το λογαριασμό και μετά την κατάθεση στον άλλο λογαριασμό. Αν αποτύχει κάποια λειτουργία θα πρέπει να αποτύχει η δοσοληψία στο σύνολό της γιατί αν για παράδειγμα αποτύχει η λειτουργία της κατάθεσης θα έχουν αφαιρεθεί τα χρήματα από τον αρχικό λογαριασμό και δε θα έχουν μεταφερθεί κάπου.

Στο σημείο αυτό θα πρέπει να αναφέρουμε το θεώρημα CAP, το οποίο δηλώνει ότι είναι αδύνατο για ένα κατανεμημένο κατάστημα δεδομένων να προσφέρει περισσότερες από δύο στις τρεις εγγυήσεις:

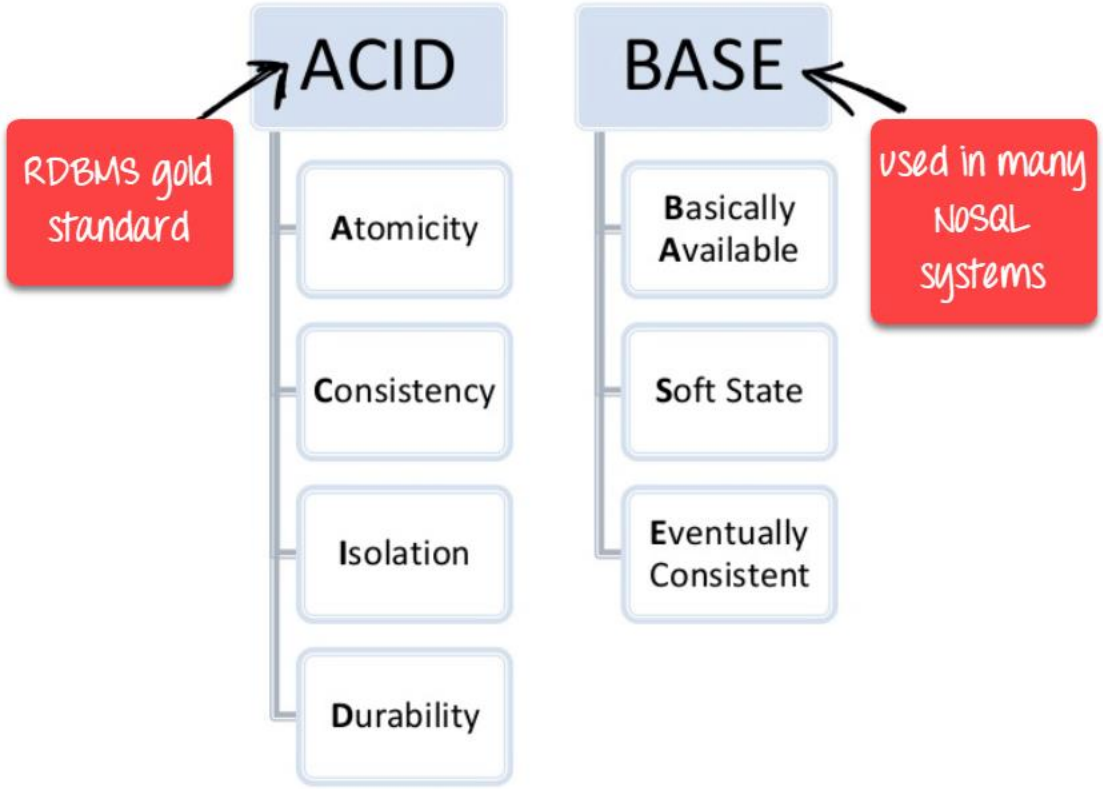
- **Consistency**: Τα δεδομένα πρέπει να παραμείνουν συνεπή ακόμη και μετά την εκτέλεση μιας λειτουργίας. Αυτό σημαίνει ότι μετά τη σύνταξη των δεδομένων, κάθε μελλοντικό αίτημα ανάγνωσης θα πρέπει να περιέχει αυτά τα δεδομένα. Για παράδειγμα, μετά την ενημέρωση της κατάστασης παραγγελίας, όλοι οι πελάτες θα πρέπει να μπορούν να δουν τα ίδια δεδομένα.
- **Availability**: Η βάση δεδομένων πρέπει να είναι πάντα διαθέσιμη και να ανταποκρίνεται στα αιτήματα. Δεν πρέπει να έχει κανένα χρόνο διακοπής.
- **Partition Tolerance**: σημαίνει ότι το σύστημα πρέπει να συνεχίσει να λειτουργεί ακόμη και αν η επικοινωνία μεταξύ των διακομιστών δεν είναι σταθερή. Για παράδειγμα, οι διακομιστές μπορούν να χωριστούν σε πολλές ομάδες που ενδέχεται να μην επικοινωνούν μεταξύ τους.



Εικόνα 8Θεώρημα CAPκαι συνδυασμοί ιδιοτήτων του[16]

Σε μια NoSQL βάση δεδομένων δεν εφαρμόζεται το ACID(εκτός από την περίπτωση της MongoDB), αλλά στις περισσότερες από αυτές εφαρμόζεται το BASE:

- **BasicallyAvailable**: σημαίνει ότι η βάση δεδομένων είναι διαθέσιμη συνεχώς σύμφωνα με το θεώρημα CAP.
- **Softstate**: σημαίνει ακόμη και χωρίς είσοδο. η κατάσταση του συστήματος μπορεί να αλλάξει.
- **Eventualconsistency**: σημαίνει ότι το σύστημα θα γίνει συνεπές με την πάροδο του χρόνου[6].



Εικόνα11ACIDvsBASE[6]

Ολοκληρώνοντας την παρουσίαση των NoSQL βάσεων δεδομένων θα πρέπει να παρουσιάσουμε συνοπτικά τις διαφορές με τις SQL βάσεις δεδομένων:

	SQL βάσεις δεδομένων	NoSQL βάσεις δεδομένων
Μοντέλο αποθήκευσης δεδομένων	Πίνακες με σταθερές σειρές και στήλες.	Ανάλογα με τον τύπο: • JSON document, • ζεύγος Κλειδιού-τιμής, • πίνακες με σειρές και δυναμικές στήλες, • κόμβοι και ακμές
Ιστορία	Αναπτύχθηκαν στη δεκαετία του 1970 με έμφαση στη μείωση της επικάλυψης δεδομένων.	Αναπτύχθηκαν στα τέλη της δεκαετίας του 2000 με έμφαση στην κλιμάκωση και επιτρέπει την ταχεία ανάπτυξη εφαρμογών που

		βασίζεται σε πρακτικές DevOps και Agile.
Κύριος σκοπός	Γενικού σκοπού	γενικός σκοπός, μεγάλες ποσότητες δεδομένων με απλά ερωτήματα αναζήτησης, μεγάλες ποσότητες δεδομένων με προβλέψιμα μοτίβο ερωτημάτων, κοινωνικά δίκτυα, συνδεδεμένα δεδομένα
Σχήμα	Άκαμπτο	Εύκαμπτο
Αναβάθμιση	Κάθετη (κλιμάκωση με μεγαλύτερο διακομιστή)	Οριζόντια (κλιμάκωση σε περισσότερους διακομιστές)
Συναλλαγές πολλαπλών εγγραφών ACID	Υποστηρίζεται	Οι περισσότερες δεν υποστηρίζουν πολλαπλές εγγραφές ACID. Ωστόσο, ορισμένες, όπως η MongoDB το υποστηρίζουν.
Συσχετίσεις	Συνήθως απαιτείται	Δεν απαιτείται

[12]

1.2.4 Βάσεις Δεδομένων διαχείρισης γράφων

Πριν ξεκινήσουμε την αναφορά στις Βάσεις Δεδομένων διαχείρισης γράφων, θα πρέπει να περιγράψουμε το λόγο που έχουν γίνει διάσημες και έχουν αποκτήσει ένα σημαντικό μερίδιο στην αγορά.

Όλοι έχουμε παρατηρήσει ότι ζούμε σε έναν κόσμο που τα πάντα συνδέονται μεταξύ τους. Ιδιαίτερα με την ανάπτυξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών η σύνδεση έγινε εμφανής και πιο στενή ακόμα και σε αντικείμενα που δε

μπορούσαμε να φανταστούμε. Διαπιστώνουμε επομένως ότι δεν υπάρχουν μεμονωμένα στοιχεία, αλλά πλούσια συνδεδεμένοι τομείς. Μόνο μια βάση δεδομένων που θα μπορέσει να ενσωματώσει εγγενώς τις σχέσεις αυτές, είναι σε θέση να αποθηκεύει, να επεξεργάζεται και να υποβάλλει αποτελεσματικά ερωτήματα στις συνδέσεις. Οι βάσεις δεδομένων που αναφέραμε δημιουργούν τις συνδέσεις κατά το χρόνο εκτέλεσης του ερωτήματος (μέσω δαπανηρών λειτουργιών JOIN), μια βάση δεδομένων γραφήματος αποθηκεύει συνδέσεις παράλληλα με τα δεδομένα.

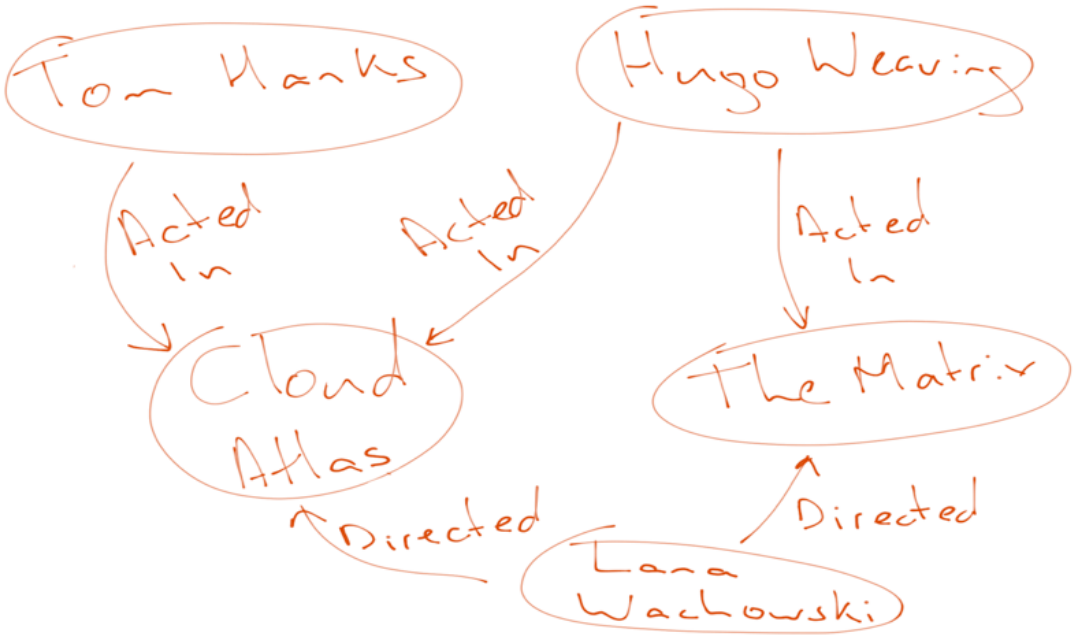
Μια βάση δεδομένων γράφων είναι μια βάση δεδομένων που έχει σχεδιαστεί για να αντιμετωπίζει τις σχέσεις μεταξύ των δεδομένων εξίσου σημαντικά με τα ίδια τα δεδομένα. Ο βασικός σκοπός της δεν είναι να προσαρμόζει τα δεδομένα στο μοντέλο αλλά να διατηρεί τα δεδομένα και να τα αποθηκεύει στο μοντέλο χωρίς να τα περιορίζει. Για το σκοπό αυτό, τα δεδομένα αποθηκεύονται όπως τα σχεδιάζουμε, δείχνοντας πώς κάθε μεμονωμένη οντότητα συνδέεται ή σχετίζεται με άλλες.

Πιστεύουμε ότι στο σημείο αυτό θα πρέπει να αναφέρουμε το βασικό στοιχείο μιας βάσης δεδομένων γράφων, το οποίο είναι το μοντέλο δεδομένων γραφήματος, το οποίο αποτελείται από βασικά δομικά στοιχεία όπως κόμβους, σχέσεις και ιδιότητες. Τα δεδομένα θα αποθηκευτούν στους κόμβους ή στις σχέσεις.

Το μοντέλο δεδομένων γραφήματος αναφέρεται συχνά ως «whiteboard-friendly». Όταν ξεκινάμε με το σχεδιασμό μιας βάσης δεδομένων, αρχίζουμε με το σχεδιασμό ενός μοντέλου δεδομένων. Αρχικά σχεδιάζουμε παραδείγματα δεδομένων στον πίνακα και τα συνδέουμε με άλλα δεδομένα που έχουν σχεδιαστεί για να δείξουμε πώς συνδέονται μεταξύ τους τα διαφορετικά αντικείμενα. Στη συνέχεια, όταν προσπαθούμε να εφαρμόσουμε το μοντέλο του πίνακα σε ένα σχεσιακό μοντέλο, αυτό αναδιαμορφώνεται και διαρθρώνεται ώστε να χωρά σε κανονικοποιημένους πίνακες.

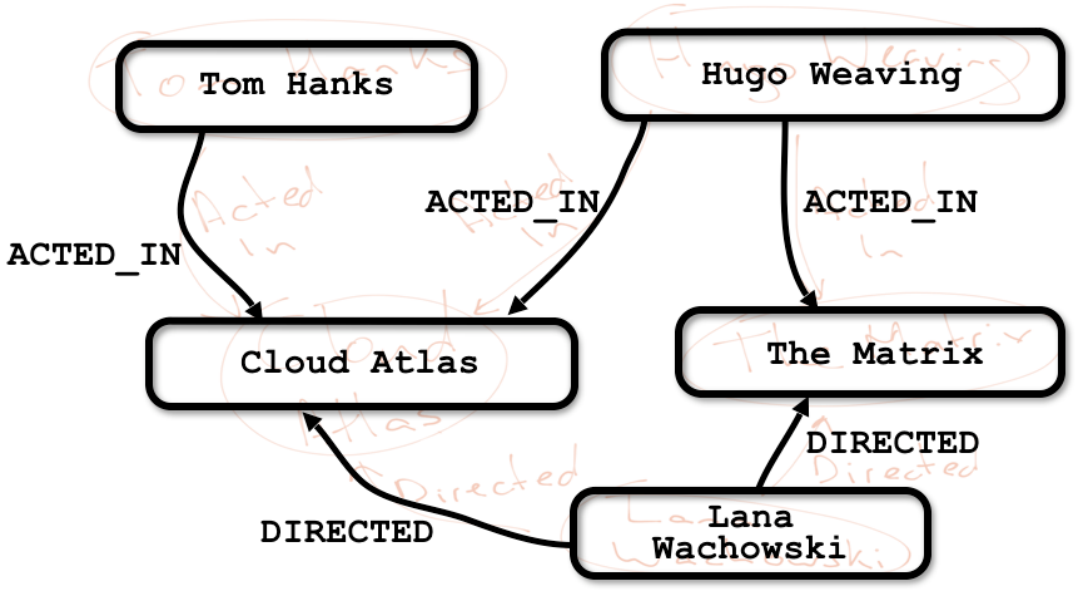
Αν όμως αντί να το αποθηκεύσουμε σε ένα σχεσιακό μοντέλο θέλουμε να το αποθηκεύσουμε σε ένα μοντέλο δεδομένων γραφήματος τότε δεν τροποποιείται το μοντέλο δεδομένων για να ταιριάξει σε μια κανονικοποιημένη δομή πίνακα, αλλά το μοντέλο δεδομένων γραφήματος παραμένει ακριβώς όπως σχεδιάστηκε στον πίνακα. Για το λόγο αυτό αναφέραμε στην αρχή ότι το μοντέλο δεδομένων γραφήματος αναφέρεται ως «whiteboard-friendly».

Στην εικόνα που ακολουθεί μπορούμε να δούμε ένα σύνολο δεδομένων που σχετίζονται με την ταινία “TheMatrix” σχεδιασμένο σε έναν πίνακα.



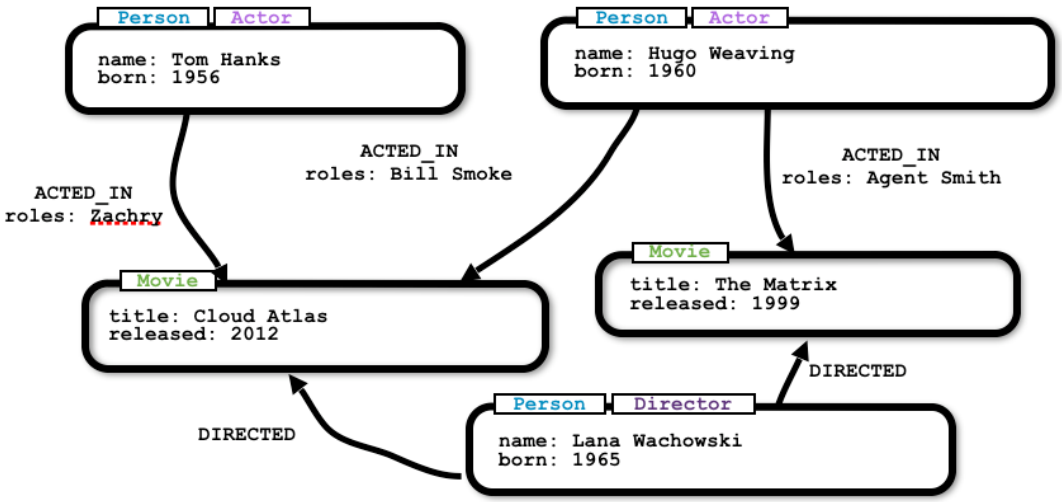
Εικόνα 9 Σχέδιο σχετικά με την ταινία "TheMatrix"[8]

Στη συνέχεια, καταλήγουμε στις οντότητες που θα χρησιμοποιήσουμε καθώς και στους τύπους των σχέσεων για να δημιουργήσουμε την προβολή κόμβου / σχέσης για το μοντέλο γραφήματος ιδιοτήτων, όπως μπορούμε να δούμε στην εικόνα που ακολουθεί.



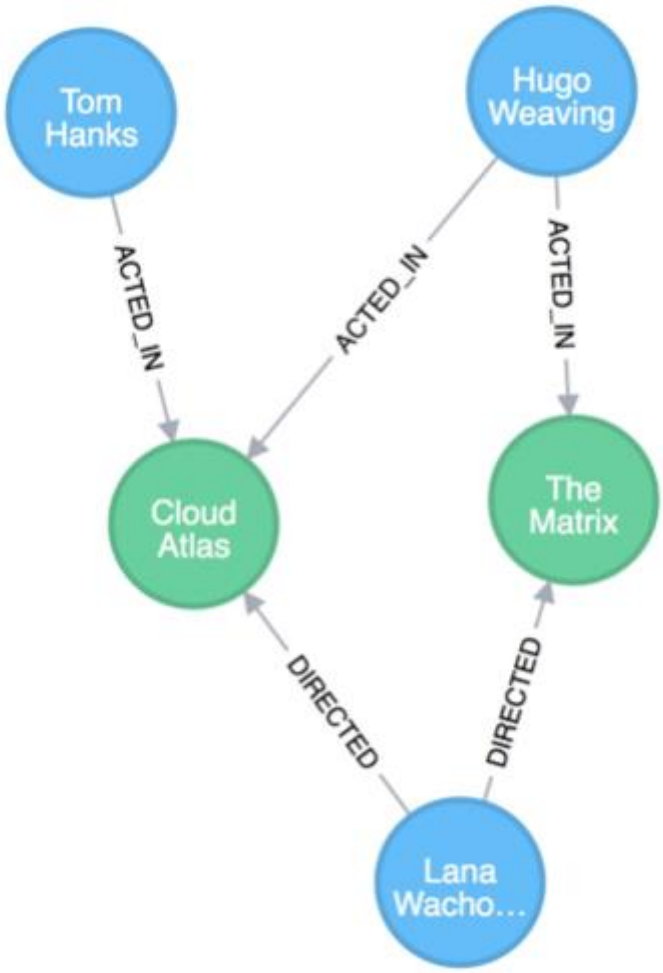
Εικόνα 10 Πίνακας αντιστοίχισης κόμβων σχέσεων του μοντέλου γραφήματος ιδιοτήτων[8]

Στο επόμενο βήμα, προσθέτουμε ετικέτες και καθορίζουμε τις ιδιότητες των κόμβων και των σχέσεων για το μοντέλο γραφήματος ιδιοτήτων, όπως βλέπουμε στην εικόνα που ακολουθεί:



Εικόνα 11 Προσθήκη ετικετών και ιδιοτήτων[8]

Τέλος, μπορούμε να δούμε το μοντέλο δεδομένων και να διασφαλίσουμε ότι ταιριάζει με αυτό που σχεδιάσαμε στον πίνακα. Στην εικόνα που ακολουθεί μπορούμε να δούμε το μοντέλο δεδομένων όπως έχει προκύψει μετά από εισαγωγή σε μια βάση δεδομένων γράφων και παρατηρούμε ότι είναι σχεδόν πανομοιότυπο με το αρχικό μοντέλο που είχε σχεδιαστεί στον πίνακα.

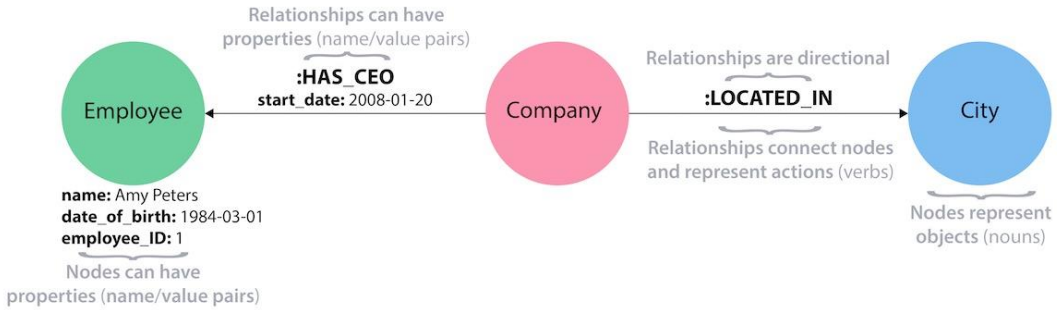


Εικόνα 12 Τελικό μοντέλο δεδομένων[8]

Για να κατανοήσουμε καλύτερα τα δομικά στοιχεία του μοντέλου δεδομένων γραφήματος αλλά και τη διαδικασία σχεδιασμού του, μπορούμε να παρουσιάσουμε ένα απλό παράδειγμα για ένα μικρό σύνολο δεδομένων και να ακολουθήσουμε όλα τα βήματα του τρόπου δημιουργίας ενός μοντέλου δεδομένων γραφήματος από αυτό. Το παράδειγμά μας είναι:

Two people, **Sally** and **John**, are friends. Both **John** and **Sally**have read the *book*, **GraphDatabases**.

Μπορούμε να χρησιμοποιήσουμε τις πληροφορίες σε αυτήν τη δήλωση για να δημιουργήσουμε το μοντέλο μας προσδιορίζοντας τα στοιχεία ως ετικέτες, κόμβους και σχέσεις. Το αποτέλεσμα που θα έχουμε θα είναι το ακόλουθο:

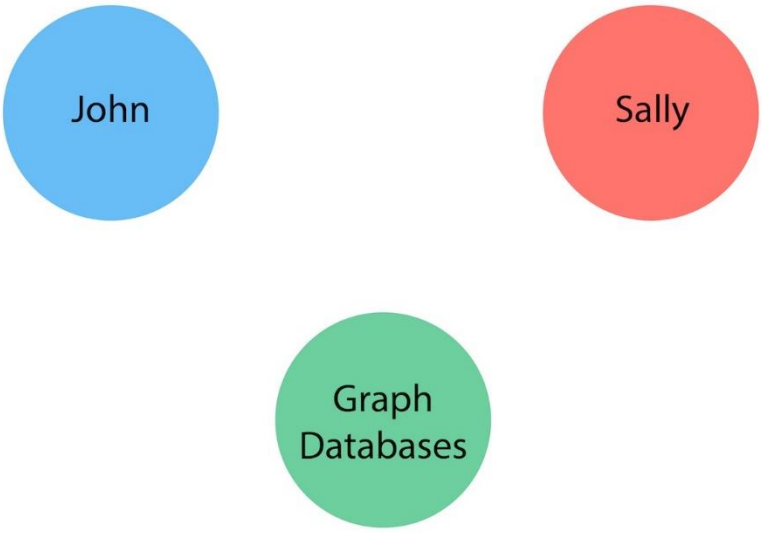


Εικόνα 13 Μοντέλο με βάση τη δήλωση[8]

Οι πρώτες οντότητες που θα προσδιορίσουμε είναι οι κόμβοι. Οι κόμβοι είναι ένα από τα δύο δομικά στοιχεία που σχηματίζουν ένα γράφημα (το άλλο δομικό στοιχείο είναι οι σχέσεις).

Οι κόμβοι χρησιμοποιούνται συχνά για την αναπαράσταση οντοτήτων. Οι κόμβοι μπορούν να κρατήσουν οποιονδήποτε αριθμό χαρακτηριστικών (ζευγάρια κλειδί-τιμή) τα οποία ονομάζονται ιδιότητες. Οι κόμβοι μπορούν να επισημανθούν με ετικέτες, που αντιπροσωπεύουν τους διαφορετικούς ρόλους τους. Οι ετικέτες των κόμβων μπορούν επίσης να χρησιμεύσουν για την προσθήκη μεταδεδομένων.

Για αναγνωρίσουμε τους κόμβους από μια περιγραφή, θα πρέπει να βρούμε τα ουσιαστικά εκείνα που έχουν μοναδική εννοιολογική ταυτότητα. Στο παράδειγμά μας είναι η Sally, ο Johnκαι οι GraphDatabases. Σχεδιάζοντας τους κόμβους θα έχουμε την ακόλουθοι εικόνα:

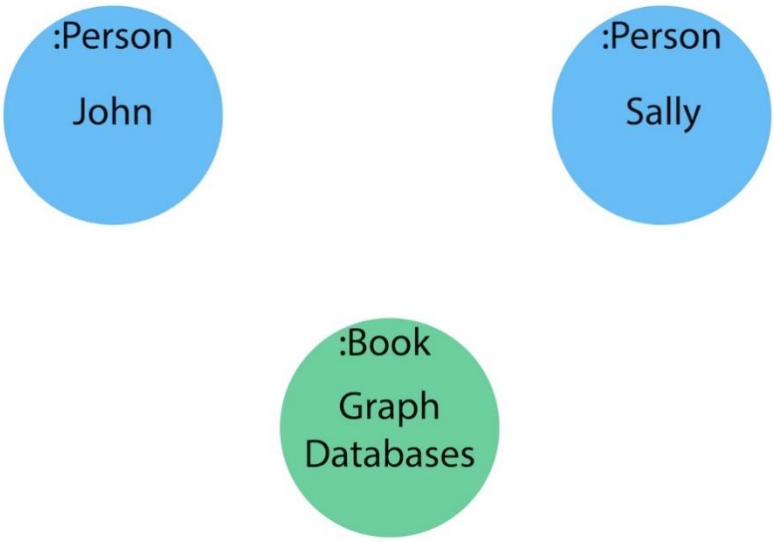


Εικόνα 14 Κόμβοι παραδείγματος[8]

Το επόμενο βήμα μετά από τον ορισμό των κόμβων, είναι να αποφασίσουμε ποιες ετικέτες (εάν υπάρχουν) θα αντιστοιχίσουμε στους κόμβους μας για να τους ομαδοποιήσουμε(ή κατηγοριοποιήσουμε).Το βήμα αυτό είναι προαιρετικό.

Μια ετικέτα χρησιμοποιείται για την ομαδοποίηση κόμβων σε σύνολα. Όλοι οι κόμβοι με την ίδια ετικέτα ανήκουν στο ίδιο σύνολο. Πολλά ερωτήματα βάσης δεδομένων μπορούν να λειτουργήσουν με αυτά τα σύνολα αντί για ολόκληρο το γράφημα, καθιστώντας τα ερωτήματα ευκολότερα στη σύνταξη και πιο αποτελεσματικά. Ένας κόμβος μπορεί να φέρει καμία, μία ή περισσότερες ετικέτες.

Για να μάθουμε αν μπορούμε να ομαδοποιήσουμε τα αντικείμενα στο σενάριό μας Sally και John, θα ξεκινήσουμε εντοπίζοντας τους ρόλους των κόμβων μας (John, Sally, Graph Databases) που αναφέρονται στη δήλωση. Μπορούμε να βρούμε δύο διαφορετικούς τύπους αντικειμένων στη δήλωση, τον τύπο Personκαι τον τύπο Book.Επομένως μπορούμε να αντιστοιχήσουμε τις ετικέτες στους κόμβους και να ανανεώσουμε το γράφημα:



Εικόνα 15 Κόμβοι και ετικέτες παραδείγματος[8]

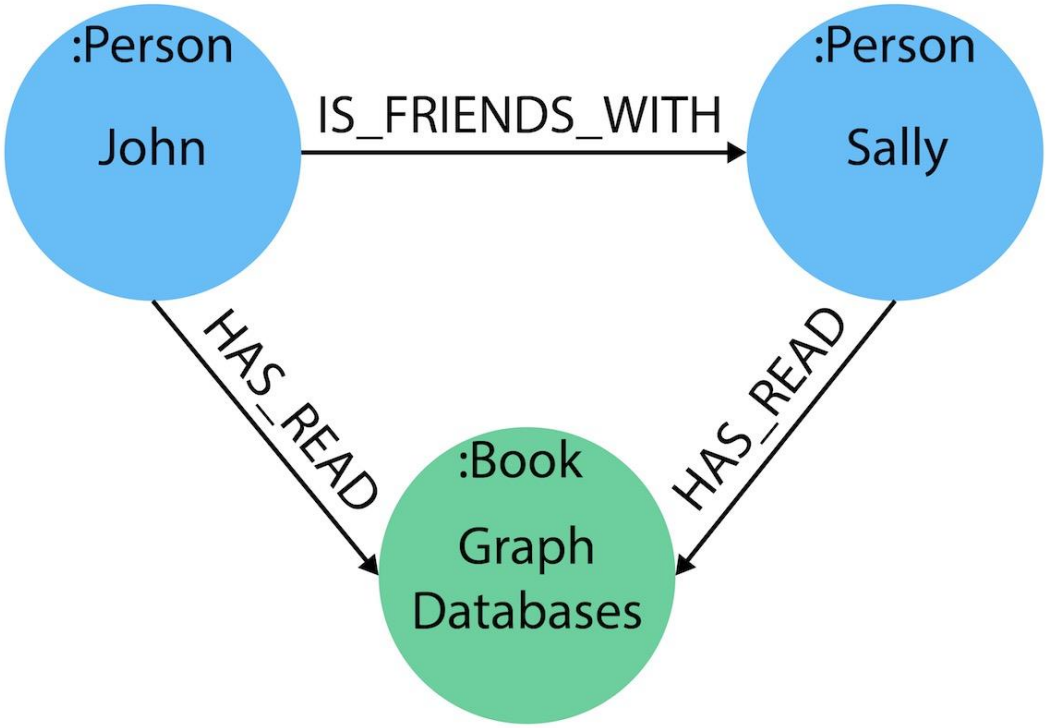
Οι σχέσεις παρέχουν κατευθυνόμενες, ονομασμένες, σημασιολογικά σχετικές συνδέσεις μεταξύ δύο κόμβων (π.χ. Υπάλληλος ΕΡΓΑΖΕΤΑΙ Εταιρεία). Μια σχέση έχει τα ακόλουθα χαρακτηριστικά:

- μια κατεύθυνση,
- έναν τύπο,
- έναν κόμβο έναρξης και
- έναν τελικό κόμβο.

Για να μπορέσουμε να αναγνωρίσουμε προσπαθούμε να προσδιορίσουμε ενέργειες ή ρήματα στο κείμενο που μας έχει δοθεί. Επομένως μπορούμε να βρούμε τις σχέσεις μεταξύ των κόμβων John, Sally και Graph Database οι οποίες είναι:

- John is friends with Sally
- Sally is friends with John
- John has read Graph Databases
- Sally has read Graph Databases

Επομένως οι κόμβοι John και Sally (με την ετικέτα Person) μπορούν να συνδεθούν μεταξύ τους με τη σχέση «is friends with». Ο Johnκαι η Sallyέχουν διαβάσει και τα δύο το βιβλίο Graph Databases, έτσι μπορούμε να συνδέσουμε κάθε έναν από τους κόμβους με τον κόμβο Graph Databases (με την ετικέτα Book) με μια σχέση «has read».



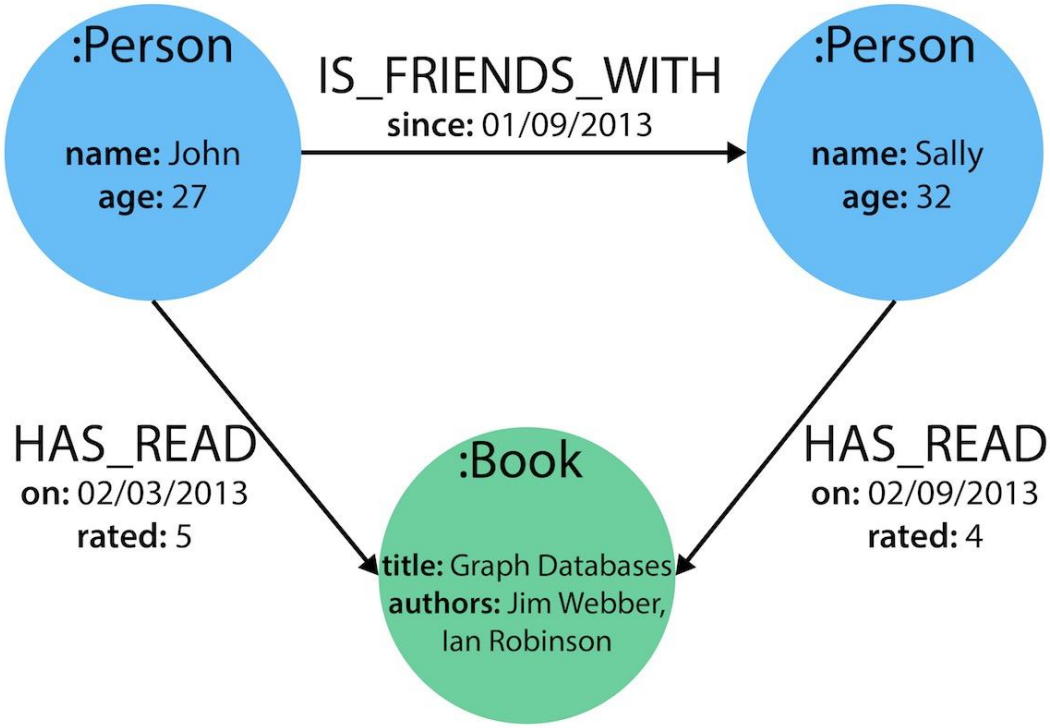
Εικόνα 16 Σχέσεις του παραδείγματος[8]

Ιδιότητες μπορούμε να προσθέσουμε και στις σχέσεις, οι οποίες (τις περισσότερες φορές) έχουν ποσοτικές ιδιότητες, όπως βάρη, κόστος, αποστάσεις, βαθμολογίες, χρονικά διαστήματα ή δυνατότητες. Για να εντοπίσουμε τις ιδιότητες που χρειαζόμαστε θα πρέπει να καθορίσουμε τις ερωτήσεις που μπορεί να χρειαστεί να απαντήσουμε.

Για το σενάριο που μελετάμε και συγκεκριμένα για τους John και Sally, μπορούμε να παραθέσουμε μερικές ερωτήσεις όπως:

- Πότε έγιναν φίλοι ο John και η Sally; Ή πόσο καιρό ήταν φίλοι;
- Ποια είναι η μέση βαθμολογία του βιβλίου Graph Databases;
- Ποιος είναι ο συγγραφέας του βιβλίου Graph Databases;
- Πόσο χρονών είναι η Sally;
- Πόσο χρονών είναι ο John;
- Ποιος είναι μεγαλύτερος, Sally ή John;
- Ποιος διάβασε πρώτα το βιβλίο Graph Databases, η Sally ή ο John;

Από αυτήν τη λίστα ερωτήσεων, μπορούμε να προσδιορίσουμε τα χαρακτηριστικά που πρέπει να αποθηκεύσουμε στις οντότητες του μοντέλου δεδομένων μας για να απαντήσουμε σε αυτές τις ερωτήσεις.



Εικόνα 17 Ιδιότητες σχέσεων [8]

Με το τελικό μοντέλο δεδομένων που αναπτύξαμε, μπορούμε πλέον να απαντήσουμε σε κάθε ερώτηση που ορίσαμε στη λίστα μας. Φυσικά, μπορούμε να αναπτύξουμε και να αλλάξουμε το μοντέλο με την πάροδο του χρόνου και να προσθέσουμε ή να αφαιρέσουμε σχέσεις, κόμβους, ιδιότητες και ετικέτες. Η ευελιξία και η απλότητα του μοντέλου δεδομένων γραφημάτων επιτρέπει στους χρήστες να ελέγχουν εύκολα τη δομή δεδομένων και να την ενημερώνουν ανάλογα με τις μεταβαλλόμενες ανάγκες της.

Η απόδοση μια βάσης δεδομένων γράφων δεν επηρεάζεται από το πλήθος των σχέσεων ανάμεσα στους κόμβους, οπότε δύο κόμβοι μπορούν να μοιράζονται οποιονδήποτε αριθμό ή τύπο σχέσης. Επίσης, παρότι αναφέραμε στα χαρακτηριστικά των σχέσεων την κατεύθυνση, μπορούμε να πλοηγηθούμε προς οποιαδήποτε κατεύθυνση [8].

1.3 Σύγκριση Σχεσιακών με Βάσεις Δεδομένων Διαχείρισης Γράφων

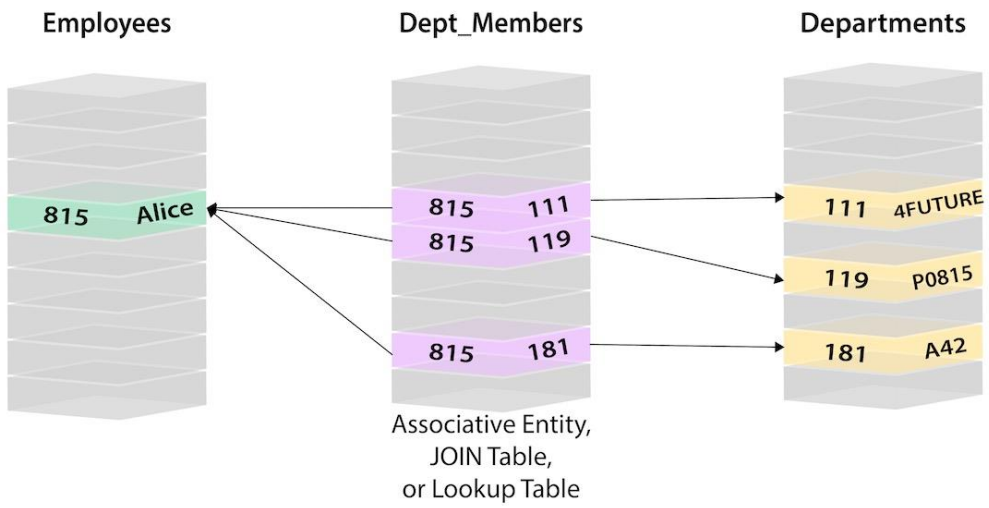
Οι σχεσιακές βάσεις δεδομένων έχουν υπάρξει το άρμα της εργασίας των εφαρμογών λογισμικού από τη δεκαετία του 80 και συνεχίζουν να χρησιμοποιούνται από πολλές

εφαρμογές, ως και σήμερα. Αποθηκεύουν τα δεδομένα σε καλά δομημένες μορφές - πίνακες με προκαθορισμένες στήλες συγκεκριμένων τύπων και σε πολλές γραμμές οι οποίες περιέχουν τις σχετικές πληροφορίες. Λόγω της ακαμψίας της οργάνωσής τους, οι σχεσιακές βάσεις δεδομένων απαιτούν από τους προγραμματιστές και τις εφαρμογές να δομήσουν αυστηρά τα δεδομένα που χρησιμοποιούνται στις εφαρμογές τους.

Στις σχεσιακές βάσεις δεδομένων, οι αναφορές σε άλλες σειρές και πίνακες καθορίζονται με αναφορά σε βασικά χαρακτηριστικά κλειδιών μέσω στηλών ξένων κλειδιών (foreignkeys). Οι συνδέσεις υπολογίζονται κατά την ώρα του ερωτήματος αντιστοιχίζοντας τα πρωτεύοντα και ξένα κλειδιά όλων των σειρών στους συνδεδεμένους πίνακες. Αυτές οι λειτουργίες είναι χρονοβόρες, απαιτούν μνήμη και έχουν εκθετικό κόστος.

Όταν υπάρχουν σχέσεις πολλά προς πολλά στο μοντέλο δεδομένων που θέλουμε να απεικονίσουμε με μια σχεσιακή βάση δεδομένων έναν πίνακα συσχέτισης ο οποίος θα κρατά τα ξένα κλειδιά και των δύο συμμετεχόντων πινάκων. Ο πίνακας συσχέτισης αυξάνει το κόστος της διασύνδεσης των δεδομένων που αναφέραμε. Στην εικόνα που ακολουθεί μπορούμε να δούμε την έννοια της σύνδεσης ενός ατόμου (από τον πίνακα προσώπων) σε ένα τμήμα (στον πίνακα τμήματος) δημιουργώντας έναν πίνακα συσχέτισης προσώπου-τμήματος που περιέχει το αναγνωριστικό του ατόμου σε μια στήλη και το αναγνωριστικό του σχετικού τμήματος στο επόμενη στήλη.

Η διαδικασία αυτή της συσχέτισης καθιστά την κατανόηση των συνδέσεων πολύ δύσκολη επειδή πρέπει να γνωρίζουμε, τις τιμές του πρωτεύοντος κλειδιού τόσο του ατόμου αλλά και του τμήματος (το οποίο απαιτεί επιπλέον αναζητήσεις) για να μάθουμε ποιο άτομο συνδέεται με ποια τμήματα. Αυτοί οι τύποι δαπανηρών δραστηριοτήτων σύνδεσης συχνά αντιμετωπίζονται με αποδιαμόρφωση των δεδομένων για τη μείωση του αριθμού των απαραίτητων συνδέσεων, παραβιάζοντας έτσι την ακεραιότητα των δεδομένων σε μια σχεσιακή βάσης δεδομένων.



Εικόνα 18 Σχεσιακό μοντέλο, πίνακας συσχέτισης[9]

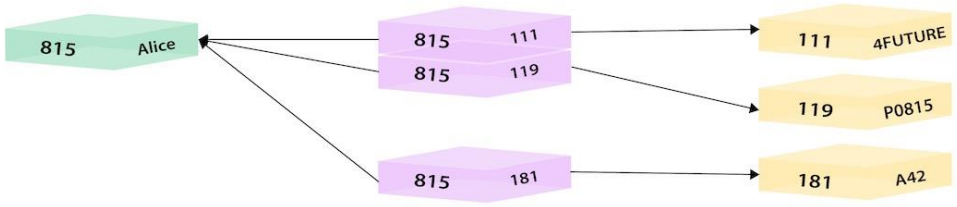
Δυστυχώς όμως δεν υπήρχε κάποια εναλλακτική λύση για την αντιμετώπιση της συγκεκριμένης δυσκολίας, το οποίο με την ευρεία υποστήριξη για τις σχεσιακές βάσεις δεδομένων καθιστούσαν δύσκολο για τα εναλλακτικά μοντέλα να εισέλθουν στην αγορά. Παρόλα αυτά, με την εισαγωγή των NoSQL βάσεων δεδομένων στην αγορά, έγινε αντιληπτή η σημασία των συνδέσεων μεταξύ δεδομένων και άνοιξε ο δρόμος για τη γέννηση των βάσεων δεδομένων γράφων. Σχεδιάστηκαν με σκοπό να παρέχουν το μεγαλύτερο πλεονέκτημα στον συνδεδεμένο κόσμο στον οποίο ζούμε σήμερα.

Σε αντίθεση με άλλα συστήματα διαχείρισης βάσεων δεδομένων, οι σχέσεις είναι εξίσου σημαντικές στο μοντέλο δεδομένων γράφων με τα ίδια τα δεδομένα. Αυτό σημαίνει ότι δεν απαιτείται να συνάγουμε συνδέσεις μεταξύ οντοτήτων χρησιμοποιώντας ειδικές ιδιότητες όπως για παράδειγμα ξένα κλειδιά κ.λπ.

Συγκεντρώνοντας κόμβους και σχέσεις σε συνδεδεμένες δομές, οι βάσεις δεδομένων γράφων μάς επιτρέπουν να κατασκευάσουμε απλά και εξελιγμένα μοντέλα που αντιστοιχούν στενά στον τομέα που θέλουμε να μελετήσουμε. Τα δεδομένα παραμένουν εξαιρετικά παρόμοια με τη μορφή τους στον πραγματικό κόσμο - μικρές, κανονικοποιημένες, αλλά πλούσια συνδεδεμένες οντότητες. Αυτό μας δίνει τη δυνατότητα να εκτελούμε ερωτήματα και να προβάλουμε τα δεδομένα από οποιοδήποτε σημείο ενδιαφέροντος, υποστηρίζοντας πολλές διαφορετικές περιπτώσεις χρήσης.

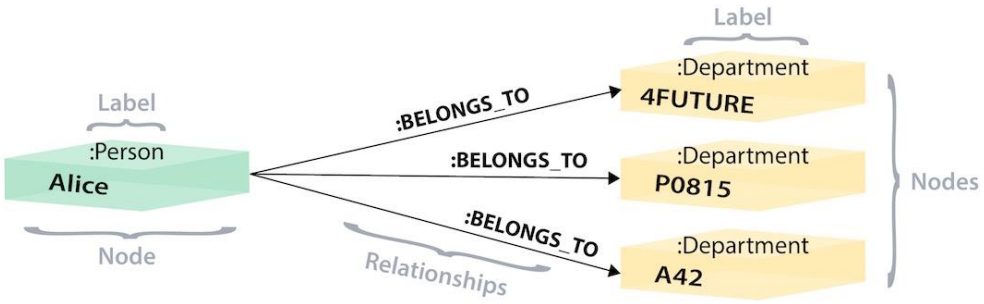
Η πιο σημαντική διαφορά είναι ότι κάθε κόμβος (οντότητα ή χαρακτηριστικό) στο μοντέλο βάσης δεδομένων γράφων περιέχει άμεσα μια λίστα με εγγραφές σχέσεων που αντιπροσωπεύουν τις σχέσεις του με άλλους κόμβους. Αυτά τα αρχεία σχέσεων οργανώνονται κατά τύπο και κατεύθυνση και μπορεί να περιέχουν επιπλέον χαρακτηριστικά. Όταν θέλουμε να εκτελέσουμε την αντίστοιχη λειτουργία με τη συσχέτιση στις σχεσιακές βάσεις δεδομένων, η βάση δεδομένων γράφων χρησιμοποιεί αυτήν τη λίστα, οπότε έχει απευθείας πρόσβαση στους συνδεδεμένους κόμβους χωρίς να υπάρχει καμία ανάγκη για ακριβούς υπολογισμούς αναζήτησης και αντιστοίχισης. Επομένως η απόδοση μιας βάση δεδομένων γράφων είναι πολύ υψηλή, ιδιαίτερα σε ερωτήματα συσχέτισης.

Επομένως αν θέλουμε να συγκρίνουμε τα δύο μοντέλα δεδομένων, αυτό των σχεσιακών βάσεων δεδομένων με αυτό των βάσεων δεδομένων γράφων, μπορούμε να αντιληφθούμε ότι η δομή του μοντέλου των βάσεων δεδομένων γράφων θα είναι πιο απλή και εκφραστική. Ας δούμε όμως με ένα παράδειγμα τις διαφορές τους. Έστω ότι έχουμε το παράδειγμα της ακόλουθης εικόνας και ψάχνουμε στον πίνακα Person (στα αριστερά, ο οποίος μπορεί να περιέχει δυνητικά εκατομμύρια εγγραφές) για να βρούμε τον χρήστη Alice και το ID της 815. Στη συνέχεια, ψάχνουμε τον πίνακα Person-Department (ο πίνακας στη μέση) για να εντοπίσουμε όλα τα σειρές που αναφέρονται στο ID που αντιστοιχεί στην Alice (815). Μόλις ανακτήσουμε τις 3 σχετικές σειρές, πηγαίνουμε στον πίνακα Department στα δεξιά για να αναζητήσουμε τις πραγματικές τιμές των αναγνωριστικών τμήματος (111, 119, 181). Τώρα γνωρίζουμε ότι η Alice ανήκει στα τμήματα 4Future, P0815 και A42.



Εικόνα 19 Συσχέτιση ατόμου - τμήματος σχεσιακές βάσεις δεδομένων[9]

Ας δούμε πώς το παραπάνω μοντέλο δεδομένων από μια σχεσιακή βάση δεδομένων μπορεί να απεικονιστεί σε μια βάση δεδομένων γράφων. Στην εικόνα που ακολουθεί μπορούμε να δούμε την απεικόνιση αυτή.



Εικόνα 20Συσχέτιση Aliceμε τα τμήματα που ανήκει με χρήση Β.Δ γράφων[9]

Στο μοντέλο δεδομένων των γράφων, έχουμε έναν μόνο κόμβο για την Alicestον οποίο έχουμε προσθέσει την ετικέτα του Person. Η Alice ανήκει σε 3 διαφορετικά Department, οπότε δημιουργούμε έναν κόμβο για κάθε ένα και προσθέτουμε σε καθένα την ετικέτα του Department. Για να μάθουμε σε ποια τμήματα ανήκει η Alice, ψάχνουμε στο γράφημα για τον κόμβο της Alice και, στη συνέχεια, διασχίζουμε όλες τις σχέσεις BELONGS_TO από την Aliceγια να βρούμε τους κόμβους του τμήματος με το οποίο είναι συνδεδεμένος.

Εκτός όμως από το μοντέλο δεδομένων, θα πρέπει να συγκρίνουμε και τις δυνατότητες αναζήτησης που μας παρέχουν οι σχεσιακές και οι βάσεις δεδομένων γράφων. Η αναζήτηση σχεσιακών βάσεων δεδομένων είναι εύκολη με τη βοήθεια της SQL - μιας δηλωτικής γλώσσας ερωτημάτων που μας επιτρέπει να δημιουργήσουμε εύκολα και γρήγορα μια ερώτηση σε ένα εργαλείο βάσης δεδομένων.

Οι βάσεις δεδομένων γράφων έχουν μια αντίστοιχη γλώσσα, τη Cypher, μια δηλωτική γλώσσα ερωτημάτων γράφων, η οποία στηρίζεται στις βασικές έννοιες και τις ρήτρες της SQL, αλλά διαθέτει πολλές πρόσθετες λειτουργίες ειδικά για το γράφημα με σκοπό να διευκολύνει την εργασία με αυτό.

Ας θυμηθούμε την προσπάθεια (που σίγουρα έχουμε καταβάλει κάποτε) για να γράψουμε μια ερώτηση SQL με μεγάλο αριθμό συσχετίσεων. Σύντομα ξεχνάμε τι κάνει πραγματικά το ερώτημα λόγω του όλου τεχνικού θορύβου στη σύνταξη της SQL. Στη Cypher, η σύνταξη παραμένει συνοπτική και επικεντρώνεται σε στοιχεία

σχετικά με το μοντέλο και τις συνδέσεις μεταξύ τους, εκφράζοντας το μοτίβο εύρεσης ή δημιουργίας δεδομένων πιο οπτικά και καθαρά.

Τη γλώσσα Cypher θα την περιγράψουμε σε επόμενη ενότητα, αλλά για να μπορέσουμε να κατανοήσουμε τη διαφορά ενός ερωτήματος SQL και ενός ερωτήματος Cypher, πιστεύουμε ότι είναι απαραίτητο ένα σύντομο παράδειγμα. Στο μοντέλο που παρουσιάσαμε παραπάνω ας δούμε πώς θα είναι μια ερώτηση σε SQL που απαριθμεί τους υπαλλήλους στο Τμήμα Πληροφορικής και πώς συγκρίνεται με τη δήλωση Cypher:

SQL
SELECT name FROM Person LEFT JOIN Person_Department ON Person.Id = Person_Department.PersonId LEFT JOIN Department ON Department.Id = Person_Department.DepartmentId WHERE Department.name = "IT Department"

Cypher
MATCH (p:Person)-[:WORKS_AT]->(d:Dept) WHERE d.name = "IT Department" RETURN p.name

Όπως μπορούμε να δούμε από τα δύο παραπάνω ερωτήματα, το ερώτημα στη Cypherείναι πολύ πιο απλό και ακόμα αν και δε γνωρίζουμε πολλά για τη συγκεκριμένη γλώσσα είναι αρκετά κατανοητό: θέλω να βρω όλα τα Personγια τα οποία υπάρχει σχέση WORKS_ATμε κάποιο Dept όπου το όνομα του Departmentείναι IT Department. Από τα Person που θα βρούμε θέλουμε μόνο το όνομα (name).

Αντίθετα το SQLερώτημα έχει ένα μεγάλο τμήμα με συσχετίσεις Joinπου θα πρέπει να γνωρίζουμε τον τρόπο διασύνδεσης, αποπροσανατολίζοντας τον αναγνώστη (ή ακόμα και τον συγγραφέα του ερωτήματος) από τον πραγματικό σκοπό του, την εύρεση των ονομάτων των υπαλλήλων που απασχολούμε στο Departmentμε όνομα IT Department [9].

Έχοντας παρουσιάσει κάποιες διαφορές ανάμεσα στις σχεσιακές βάσεις δεδομένων και στις βάσεις δεδομένων γράφων, ας δούμε και τα πλεονεκτήματα που προσφέρουν οι βάσεις δεδομένων γράφων:

- **Μικρότερος χρόνος εκτέλεσης ερωτημάτων ιδίως για μεγάλα σύνολα δεδομένων**: πολύ καλύτερη απόδοση κατά την αναζήτηση σε μεγάλα σύνολα δεδομένων. Οι σχεσιακές βάσεις δεδομένων έχουν κάπως περιορισμένη ικανότητα χειρισμού πολλαπλών συνδέσεων, ειδικά σε μεγάλα σύνολα δεδομένων χωρίς να εισάγουν περιττά επίπεδα πολυπλοκότητας. Ένα σύνθετο σχεσιακό ερώτημα που περιλαμβάνει join μεταξύ πινάκων επιβαρύνει αρκετά τη βάση δεδομένων με αποτέλεσμα να υπάρχει καθυστέρηση στην εξαγωγή των αποτελεσμάτων. Αντίθετα οι βάσεις δεδομένων γράφων υπερέχουν στην αναζήτηση σε μεγάλα σύνολα δεδομένων. Όταν τα δεδομένα αποθηκεύονται στη φυσική τους μορφή (σε μορφή γράφων) τα ερωτήματα εκτελούνται ταχύτερα σε σχέση με μια σχεσιακή δομή δεδομένων που αποθηκεύει τα δεδομένα προσαρμόζοντάς τα ώστε να ταιριάζουν σε ένα μοντέλο πίνακα. Η χρονική πολυπλοκότητα μεταξύ των δύο αναζητήσεων είναι εμφανής όταν οι αναζητήσεις γίνονται σε δεδομένα μεγέθους petabytes.
- **Ταυτόχρονη ενημέρωση και εκτέλεση ερωτήματος δεδομένων**: οι βάσεις δεδομένων γράφων προσφέρουν τη μοναδική δυνατότητα ενημέρωσης δεδομένων σε πραγματικό χρόνο, επιτρέποντας ταυτόχρονα να εκτελούνται ταυτόχρονα ερωτήματα στα δεδομένα[5].
- **Συγκεντρωτικά ερωτήματα**: Οι βάσεις δεδομένων γράφων, εκτός από τα παραδοσιακά ερωτήματα, μπορούν να εκτελέσουν συγκεντρωτικά ερωτήματα σε συγκεκριμένες κατηγορίες ομάδων που είναι αδιανόητα ή ανέφικτα σε σχεσιακές βάσεις δεδομένων. Λόγω του περιορισμού του μοντέλου του πίνακα, τα συγκεντρωτικά ερωτήματα σε μια σχεσιακή βάση δεδομένων περιορίζονται σε μεγάλο βαθμό από τον τρόπο ομαδοποίησης των δεδομένων. Αντίθετα, τα μοντέλα γράφων είναι πιο ευέλικτα για την ομαδοποίηση και τη συγκέντρωση σχετικών δεδομένων.
- **Συνδυασμός και ιεράρχηση σε πολλές διαστάσεις**: οι βάσεις δεδομένων γράφων μπορούν να συνδυάσουν πολλαπλές διαστάσεις για τη διαχείριση μεγάλων δεδομένων, συμπεριλαμβανομένων χρονοσειρών, δημογραφικών,

γεωγραφικών διαστάσεων κ.λπ. με μια ιεραρχία λεπτομέρειας σε διαφορετικές διαστάσεις[15].

- **Δυναμικό σχήμα**: σε αντίθεση με τις σχεσιακές βάσεις δεδομένων, οι οποίες απαιτούν τα δεδομένα να συμμορφώνονται με ένα καθορισμένο σχήμα που εκχωρήθηκε κατά την έναρξη του πίνακα, οι βάσεις δεδομένων γράφων επιτρέπουν την προσθήκη ή την αφαίρεση κόμβων και σχέσεων σε οποιοδήποτε σημείο κατά τη διάρκεια του κύκλου ζωής των δεδομένων.
- **Ενσωμάτωση μοντέλων AI / ML**: οι βάσεις δεδομένων γράφων προσφέρουν πλούσιες και προσβάσιμες πληροφορίες από τα δεδομένα τους, προσθέτοντας βάθος και νόημα σε μοντέλα AI που δεν είναι διαθέσιμα από σχεσιακές βάσεις δεδομένων.

Όμως οι βάσεις δεδομένων γράφων έχουν και κάποια μειονεκτήματα:

- **Έλλειψη ειδικών – ανάγκη εκπαίδευσης στελεχών**: παρόλο που οι βάσεις δεδομένων γράφων έχουν κερδίσει δημοτικότητα από τις αρχές της δεκαετίας του 2000, εξακολουθεί να υπάρχει μόνο ένας περιορισμένος αριθμός τεχνολόγων που είναι άνετοι να εργαστούν με αυτό το εργαλείο, πόσο μάλλον να είναι ειδικοί σε αυτό. Όταν μια εταιρεία αποφασίζει να μετακινηθεί σε μια βάση δεδομένων γράφων, πρέπει να αποφασίσει να επενδύσει σημαντικές ώρες και πόρους για την εκπαίδευση στελεχών.
- **Τα δεδομένα είναι τόσο πλούσια όσο τα δημιουργήσαμε**: η απλή μετεγκατάσταση των δεδομένων σχεσιακών βάσεων δεδομένων σε μια βάση δεδομένων γράφων δεν θα μετατρέψει αυτόματα σε ένα πλουσιότερο σύνολο δεδομένων. Ο μηχανικός δεδομένων πρέπει να δημιουργήσει ένα νέο σχήμα, συνήθως από το μηδέν, και να συγκεντρώσει τα δεδομένα για να ταιριάξει σε αυτό το νέο καλούπι. Αυτό, πάλι, απαιτεί πολύ χρόνο και πόρους[5].

1.4Neo4J

Έχοντας παρουσιάσει στις προηγούμενες ενότητες τις NoSQLβάσεις δεδομένων καθώς και τις βάσεις δεδομένων γράφων, θα προχωρήσουμε στην επιλογή μιας βάσης δεδομένων γράφων την οποία θα παρουσιάσουμε εν συντομία και στη συνέχεια θα τη χρησιμοποιήσουμε για να υλοποιήσουμε την εφαρμογή μας.

Προσπαθώντας να βρούμε μια βάση δεδομένων γράφων την οποία θα χρησιμοποιούσαμε για την εφαρμογή μας, διαπιστώσαμε ότι υπήρχαν αρκετές. Στην

εικόνα που ακολουθεί μπορούμε να δούμε την κατάταξη των βάσεων δεδομένων γράφων σχετικά με τη δημοτικότητά τους:

Rank			DBMS	Database Model	Score		
Jan 2021	Dec 2020	Jan 2020			Jan 2021	Dec 2020	Jan 2020
1.	1.	1.	Neo4j	Graph	53.79	-0.85	+2.13
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	32.97	-0.57	+1.46
3.	↑ 4.	↑ 4.	OrientDB	Multi-model	5.33	+0.04	+0.22
4.	↓ 3.	↓ 3.	ArangoDB	Multi-model	5.29	-0.22	+0.08
5.	5.	↑ 6.	JanusGraph	Graph	2.58	-0.07	+0.81
6.	↑ 7.	↑ 7.	Amazon Neptune	Multi-model	2.31	-0.20	+0.58
7.	↓ 6.	↓ 5.	Virtuoso	Multi-model	2.15	-0.44	-0.50
8.	8.	8.	GraphDB	Multi-model	2.11	+0.07	+0.98
9.	9.	↑ 14.	FaunaDB	Multi-model	1.91	-0.11	+1.11
10.	10.	↓ 9.	Dgraph	Graph	1.56	-0.13	+0.51
11.	11.	↑ 13.	Stardog	Multi-model	1.47	+0.00	+0.66
12.	12.	↓ 11.	TigerGraph	Graph	1.40	-0.07	+0.39
13.	↑ 14.	↓ 12.	AllegroGraph	Multi-model	1.19	+0.13	+0.33
14.	↓ 13.	↓ 10.	Giraph	Graph	1.13	-0.01	+0.11
15.	15.	↑ 28.	Nebula Graph	Graph	0.92	+0.06	+0.86
16.	16.	↓ 15.	Blazegraph	Multi-model	0.87	+0.03	+0.23
17.	↑ 18.	↓ 16.	Graph Engine	Multi-model	0.70	+0.00	+0.13

Εικόνα 21 Δημοτικότητα βάσεων δεδομένων γράφων [4]

Όπως μπορούμε να δούμε από την κατάταξη η βάση δεδομένων Neo4jπαραμένει πρώτη τουλάχιστον ένα χρόνο και με μεγάλη διαφορά από τη δεύτερη βάση δεδομένων γράφων την JanusGraphπου είναι στην έκτη θέση.

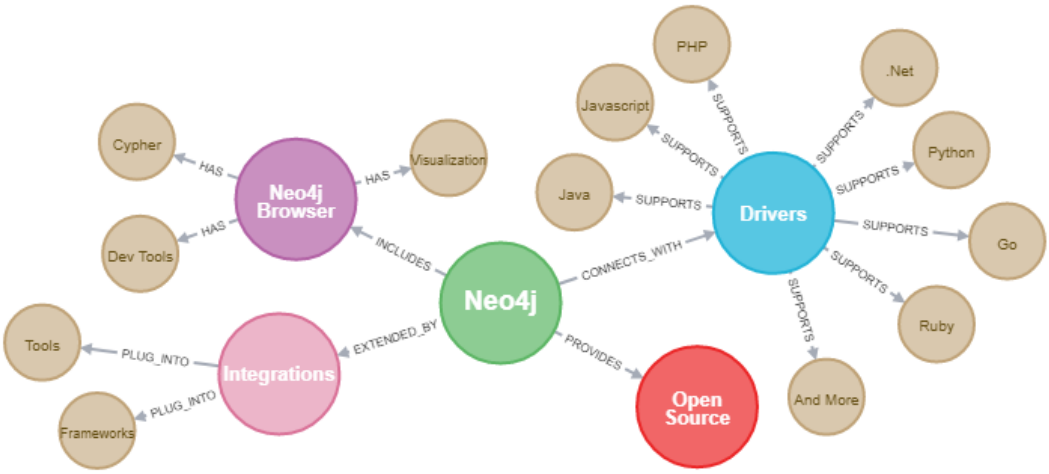
Η Neo4j είναι μια ανοιχτή κώδικα, NoSQL, εγγενής βάση δεδομένων γράφων που παρέχει ένα συμβατό με ACID σύστημα συναλλαγών για τις εφαρμογές.

Η Neo4j λέγεται ότι είναι μια εγγενής βάση δεδομένων γράφων, επειδή εφαρμόζει αποτελεσματικά το μοντέλο γράφων έως το επίπεδο αποθήκευσης. Παρέχει επίσης πλήρη χαρακτηριστικά βάσης δεδομένων, όπως συμμόρφωση συναλλαγών ACID, υποστήριξη cluster και runtime fail over. Η Neo4j υποστηρίζει τη δική της γλώσσα ερωτημάτων που ονομάζεται Cypher και θα παρουσιαστεί στην επόμενη ενότητα.

Τα πλεονεκτήματα που διαθέτει η Neo4jκαι την έχουν φέρει πρώτη σε προτιμήσεις είναι:

- Ευέλικτο μοντέλο δεδομένων: παρέχει ένα ευέλικτο απλό αλλά ισχυρό μοντέλο δεδομένων, το οποίο μπορεί εύκολα να αλλάξει ανάλογα με τις απαιτήσεις των εφαρμογών.

- Πληροφορίες σε πραγματικό χρόνο: παρέχει αποτελέσματα βασισμένα σε δεδομένα σε πραγματικό χρόνο.
- Υψηλή διαθεσιμότητα: παρέχει υψηλή διαθεσιμότητα για εφαρμογές σε πραγματικό χρόνο με εγγυήσεις συναλλαγών.
- Εύκολη ανάκτηση: με τη βοήθεια της Neo4j, μπορούμε όχι μόνο να απεικονίζουμε αλλά και να αναζητήσουμε εύκολα συνδεδεμένα δεδομένα γρηγορότερα σε σύγκριση με άλλες βάσεις δεδομένων.
- Γλώσσα ερωτήματος Cypher: παρέχει μια δηλωτική γλώσσα ερωτημάτων για την οπτική αναπαράσταση του γραφήματος, χρησιμοποιώντας μια σύνταξη ascii-art. Οι εντολές αυτής της γλώσσας είναι σε μορφή αναγνώσιμη από τον άνθρωπο και μπορεί κάποιος να ξεκινήσει να τη χρησιμοποιεί άμεσα.
- Προγράμματα οδήγησης για δημοφιλείς γλώσσες προγραμματισμού: Η ανάπτυξη εφαρμογών χρησιμοποιώντας την Neo4j είναι απλή. Παρέχονται ένα πλήθος από προγράμματα οδήγησης για γλώσσες όπως .Net, Java (επίσης Spring), JavaScript και Python. Επίσης αναπτύσσονται συνεχώς άλλα από την ενεργή κοινότητα των συνεργατών.
- Ευκολία εισαγωγής δεδομένων: η Neo4jπαρέχει ένα σύνολο από εργαλεία για εισαγωγή δεδομένων από άλλες πηγές[9].



Εικόνα 22 Δυνατότητες της Neo4j [9]

1.5 Cypher

1.5.1 Εισαγωγή

Η Cypher είναι μια δηλωτική γλώσσα ερωτήματος γράφων που επιτρέπει την εκφραστική και αποτελεσματική ενημέρωση και διαχείριση του γραφήματος. Έχει σχεδιαστεί ώστε να είναι κατάλληλη τόσο για προγραμματιστές όσο και για επαγγελματίες. Η Cypher σχεδιάστηκε με ένα διπλό σκοπό την απλότητα χωρίς όμως να χάνει τη δύναμή της.

Η Cypher εμπνέεται από μια σειρά διαφορετικών προσεγγίσεων και βασίζεται σε καθιερωμένες πρακτικές για εκφραστική αναζήτηση. Πολλές από τις λέξεις-κλειδιά, όπως η WHERE και η ORDER BY, εμπνέονται από τη SQL. Σε ότι αφορά το pattern matchingέχει δανειστεί την προσέγγιση της SPARQL. Οι κατασκευές της Cypher, έχουν βασιστεί στην αγγλική πεζογραφία και την εικονογραφία, με αποτέλεσμα να διευκολύνεται τόσο η σύνταξη των ερωτημάτων αλλά και η ανάγνωσή τους.

Η Cypher δανείζεται τη δομή της από την SQL. Αποτελείται από ένα σύνολο ερωτημάτων τα οποία μπορούν να συνδεθούν μεταξύ τους και η έξοδος του ενός να τροφοδοτεί την είσοδο του επομένου.

Ας ξεκινήσουμε την παρουσίαση των πιο σημαντικών ερωτημάτων:

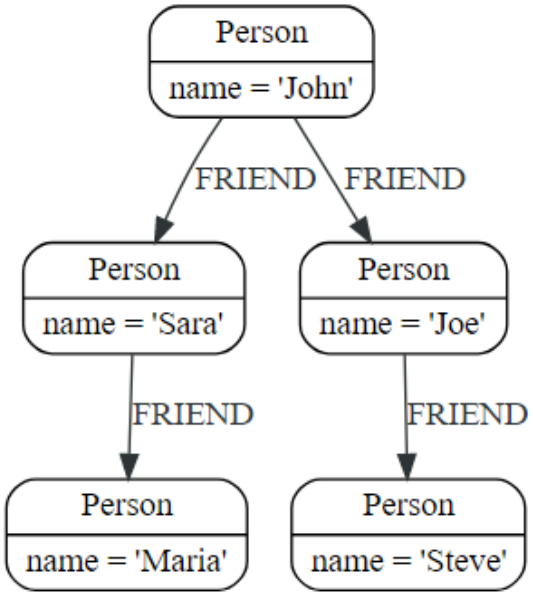
- **MATCH**: αναζητά το μοτίβο γραφήματος που ταιριάζει με αυτό που αναζητάμε. Αυτός είναι ο πιο συνηθισμένος τρόπος λήψης δεδομένων από ένα γράφημα.
- **WHERE**: δεν είναι από μόνο του ερώτημα αλλά μέρος του MATCH, OPTIONAL MATCH και WITH. Προσθέτει περιορισμούς σε ένα μοτίβο ή φιλτράρει το ενδιάμεσο αποτέλεσμα από το WITH.
- **RETURN**: τί θέλουμε να επιστρέψουμε.
- **CREATE, DELETE**: για δημιουργία κόμβων ή διαγραφή τους

Αρχικά θα δημιουργήσουμε ένα απλό γράφημα χρησιμοποιώντας το ερώτημα CREATEγια τη δημιουργία των κόμβων, με το ακόλουθο ερώτημα:

```
CREATE (john:Person {name: 'John'})
CREATE (joe:Person {name: 'Joe'})
CREATE (steve:Person {name: 'Steve'})
CREATE (sara:Person {name: 'Sara'})
```

```
CREATE (maria:Person {name: 'Maria'})
CREATE (john)-[:FRIEND]->(joe)-[:FRIEND]->(steve)
CREATE (john)-[:FRIEND]->(sara)-[:FRIEND]->(maria)
```

Το αποτέλεσμα των παραπάνω εντολών CREATE είναι η δημιουργία του ακόλουθου γραφήματος:



Ας δούμε τώρα πώς μπορούμε να βρούμε το χρήστη Johnκαι τους φίλους του. Το ερώτημα που θα εκτελέσουμε είναι:

```
MATCH (john {name: 'John'})-[:FRIEND]->()-[:FRIEND]->(fof)
RETURN john.name, fof.name
```

Το αποτέλεσμα του παραπάνω ερωτήματος είναι:

```
+----------------------+
| john.name | fof.name |
+----------------------+
| "John"    | "Maria"  |
| "John"    | "Steve"  |
+----------------------+
2 rows
```

Στη συνέχεια θα προσθέσουμε φίλτρα στο ερώτημά μας. Θέλουμε να φέρουμε τα ονόματα των χρηστών και των φίλων τους, αλλά μόνο αυτά που το όνομα των φίλων τους αρχίζει με το γράμμα S.

```
MATCH (user)-[:FRIEND]->(follower)
WHERE user.name IN ['Joe', 'John', 'Sara', 'Maria', 'Steve'] AND follower.name =~ 'S.*'
RETURN user.name, follower.name
```

Το αποτέλεσμα του ακόλουθου ερωτήματος είναι:

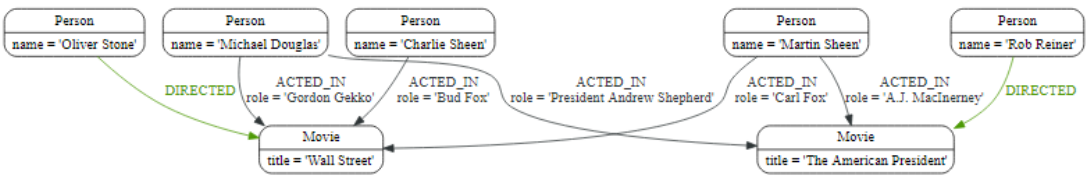
```
+---------------------------+
| user.name | follower.name |
+---------------------------+
| "Joe"     | "Steve"       |
| "John"    | "Sara"        |
+---------------------------+
2 rows
```

1.5.2 MATCH

Το ερώτημα MATCH επιτρέπει να καθορίσουμε τα μοτίβο που θα αναζητήσει η Neo4j στη βάση δεδομένων. Το MATCH συνδέεται συχνά με το WHERE το οποίο προσθέτει περιορισμούς ή κατηγορίες στα μοτίβο MATCH, καθιστώντας τα πιο συγκεκριμένα.

ToMATCH μπορεί να συμβεί στην αρχή του ερωτήματος ή αργότερα, πιθανώς μετά από ένα μεWITH. Εάν είναι στην αρχή του ερωτήματος, τότε η Neo4j θα σχεδιάσει μια αναζήτηση για να βρει τα αποτελέσματα που ταιριάζουν με τον όρο και τυχόν σχετικές κατηγορίες που ορίζονται με κάποιο WHERE.

Για την καλύτερη κατανόηση του ερωτήματος MATCHθα χρησιμοποιήσουμε το ακόλουθο γράφημα



Και θα εκτελέσουμε μια σειρά από ερωτήματα:

Το ερώτημα

```
MATCH (n)
RETURN n
```

Θα επιστρέψει όλους τους κόμβους της βάσης δεδομένων.

Αν θέλουμε να επιστραφούν όλοι οι κόμβοι που έχουν μια ετικέτα θα εκτελέσουμε το ακόλουθο ερώτημα:

```
MATCH (movie:Movie)
RETURN movie.title
```

Το σύμβολο --το οποίο σημαίνει σχετίζονται με, χωρίς να λαμβάνει υπόψη του τον τύπο ή την κατεύθυνση της σχέσης. Με το ακόλουθο ερώτημα θα επιστραφούν όλες οι ταινίες που directoreίναι ο OliverStone.

```
MATCH (director { name: 'Oliver Stone' })--(movie)
RETURN movie.title
```

Αν θέλουμε να λάβουμε υπόψη μας την κατεύθυνση της σχέσης θα χρησιμοποιήσουμε το --> ή το <--. Για παράδειγμα με το ακόλουθο ερώτημα καθορίζουμε την κατεύθυνση της σχέσης.

```
MATCH (:Person { name: 'Oliver Stone' })-->(movie)
RETURN movie.title
```

1.5.3 RETURN

Το τί θα επιστρέψουμε από το ερώτημα καθορίζεται με το RETURN. Από ένα ερώτημα μπορούμε να επιστρέψουμε κόμβους, σχέσεις ή ιδιότητες.

Αν θέλουμε να επιστρέψουμε κόμβο τότε θα πρέπει να το δηλώσουμε όπως στο ακόλουθο ερώτημα:

```
MATCH (n { name: 'B' })
RETURN n
```

Αν θέλουμε να επιστρέψουμε σχέση θα πρέπει να τη δηλώσουμε στο RETURN όπως στο ερώτημα που ακολουθεί:

```
MATCH (n { name: 'A' })-[r:KNOWS]->(c)
RETURN r
```

Αν θέλουμε να επιστρέψουμε ιδιότητα θα πρέπει να την αναφέρουμε όπως για παράδειγμα το ακόλουθο ερώτημα στο οποίο θα επιστέψουμε το nametου κόμβου:

```
MATCH (n { name: 'A' })
RETURN n.name
```

Τέλος αν θέλουμε να επιστρέψουμε όλους τους κόμβους και τις σχέσεις θα πρέπει να βάλουμε ένα * μετά από το RETURNόπως στο ακόλουθο ερώτημα:

```
MATCH p =(a { name: 'A' })-[r]->(b)
RETURN *
```

1.5.4 CREATE

Χρησιμοποιείται για τη δημιουργία κόμβων. Αν χρησιμοποιήσουμε το ακόλουθο ερώτημα θα δημιουργηθούν δύο κόμβοι. Επομένως αν θέλουμε να δημιουργήσουμε περισσότερους από έναν κόμβους τους χωρίζουμε με ,

```
CREATE (n),(m)
```

Για τη δημιουργία ενός κόμβου με label εκτελούμε το ακόλουθο ερώτημα:

```
CREATE (n:Person)
```

Αν θέλουμε σε ένα κόμβο να προσθέσουμε περισσότερα από ένα label τότε θα εκτελέσουμε το ερώτημα:

```
CREATE (n:Person:Swedish)
```

Στη συνέχεια θα δούμε ένα ερώτημα το οποίο προσθέτει εκτός από label και ιδιότητες σε ένα κόμβο:

```
CREATE (n:Person { name: 'Andy', title: 'Developer' })
```

Αν θέλουμε να προσθέσουμε μια σχέση ανάμεσα σε δύο κόμβους θα πρέπει πρώτα απ' όλα να τους αναζητήσουμε και στη συνέχεια να προσθέσουμε τη σχέση, όπως στο ερώτημα που ακολουθεί, το οποίο προσθέτει τη σχέση RELTYPE μεταξύ τον κόμβων ακαι b.

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'A' AND b.name = 'B'
CREATE (a)-[r:RELTYPE]->(b)
RETURN type(r)
```

Αν θέλουμε να προσθέσουμε ιδιότητες στη σχέση που θα δημιουργήσουμε θα τροποποιήσουμε λίγο το προηγούμενο παράδειγμα και θα έχουμε:

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'A' AND b.name = 'B'
CREATE (a)-[r:RELTYPE { name: a.name + '<->' + b.name }]->(b)
RETURN type(r), r.name
```

1.5.5 DELETE

Χρησιμοποιείται για διαγραφή κόμβων. Θα πρέπει να δοθεί προσοχή, γιατί όταν διαγράφουμε έναν κόμβο τότε διαγράφονται και όλες οι σχέσεις που ξεκινούν ή καταλήγουν σε αυτόν.

Με το ερώτημα που ακολουθεί διαγράφουμε έναν κόμβο (αφού πρώτα τον αναζητήσουμε)

```
MATCH (n:Person { name: 'UNKNOWN' })
DELETE n
```

Για διαγραφή όλων των κόμβων και των σχέσεων πρώτα θα αναζητήσουμε όλους τους κόμβους και μετά θα τους διαγράψουμε:

```
MATCH (n)
DETACH DELETE n
```

1.5.6 ORDERBY

Αν θέλουμε να ταξινομήσουμε τα αποτελέσματα ενός ερωτήματος με βάση κάποια ιδιότητα τότε θα χρησιμοποιήσουμε το ORDERBY. Στο ακόλουθο ερώτημα βλέπουμε πώς μπορούμε να ταξινομήσουμε τα αποτελέσματα ενός ερωτήματος με βάση μια ιδιότητα:

```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.name
```

Αν θέλουμε να χρησιμοποιήσουμε δύο ιδιότητες απλά θα τροποποιήσουμε το προηγούμενο ερώτημα σε:

```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.age, n.name
```

Για φθίνουσα ταξινόμηση θα προσθέσουμε το DESC στο τέλος των ιδιοτήτων του ORDERBY:

```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.name DESC
```

[10]

ΚΕΦΑΛΑΙΟ2. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗ

2.1 Εισαγωγή

Η εφαρμογή που θα υλοποιήσουμε για να μελετήσουμε τις βάσεις δεδομένων των γράφων θα είναι μια απλή εφαρμογή η οποία θα αντλεί στοιχεία από το Twitter και θα τα εισάγει σε μια βάση δεδομένων γράφων.

Για την υλοποίηση της εφαρμογής χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Python και τις σχετικές βιβλιοθήκες, για βάση δεδομένων χρησιμοποιήσαμε τη Neo4jκαι για την εκτέλεση των απαραίτητων ερωτημάτων στη βάση δεδομένων χρησιμοποιήσαμε τη Cypher.

Στις ενότητες που ακολουθούν θα περιγράψουμε το σχεδιασμό της εφαρμογής και στη συνέχεια τα βήματα της υλοποίησής της, ξεκινώντας με την εγκατάσταση της βάσης δεδομένων Neo4j. Στη συνέχεια θα παρουσιάσουμε (εν συντομία) τις απαραίτητες βιβλιοθήκες της Python που χρησιμοποιήσαμε. Τέλος θα δούμε τον τρόπο της εισαγωγής και αναζήτησης των δεδομένων στη βάση δεδομένων Neo4j.

2.2 Σχεδιασμός

Αρχικά μελετήσαμε τις δυνατότητες πρόσβασης που είχαμε για το Twitter. Διαπιστώσαμε ότι παρέχονταν δυνατότητες αναζήτησης των tweet και άντλησης στοιχείων τόσο για τα tweet όσο και για τους ανθρώπους που τα έκαναν. Επιλέξαμε να αναζητήσουμε tweetμε βάση κάποιο hashtag και στη συνέχεια να αντλήσουμε τα ονόματα και τις τοποθεσίες (αν είχαν δηλωθεί) των ατόμων που τα έκαναν.

Μελετήσαμε τις βιβλιοθήκες της Pythonπου έπρεπε να χρησιμοποιήσουμε τόσο για σύνδεση και άντληση δεδομένων από το Twitterαλλά και για σύνδεση και επικοινωνία με τη Neo4j.

Τέλος αποφασίσαμε, για να είναι πιο φιλικό προς το χρήστη, να δημιουργήσουμε και ένα GUIτο οποίο θα παρέχει βασικές λειτουργίες στο χρήστη του, δηλαδή:

- Προσθήκη hashtag
- Αναζήτηση στο twitterμε βάση το hashtag και επιστροφή των 40 πρώτων αποτελεσμάτων. Στη συνέχεια προσθήκη των αποτελεσμάτων στη Neo4jκαι δημιουργία του γράφου.

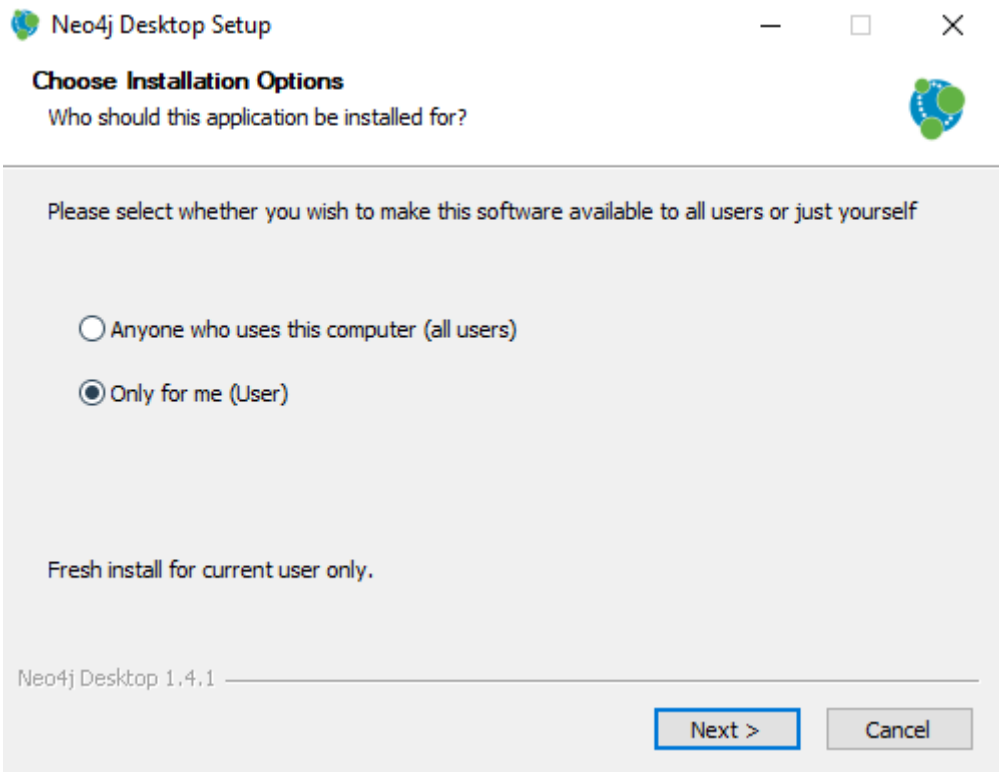
- Διαγραφή όλων των δεδομένων από τη Neo4j.

2.3 Υλοποίηση

2.3.1 Εγκατάσταση neo4j

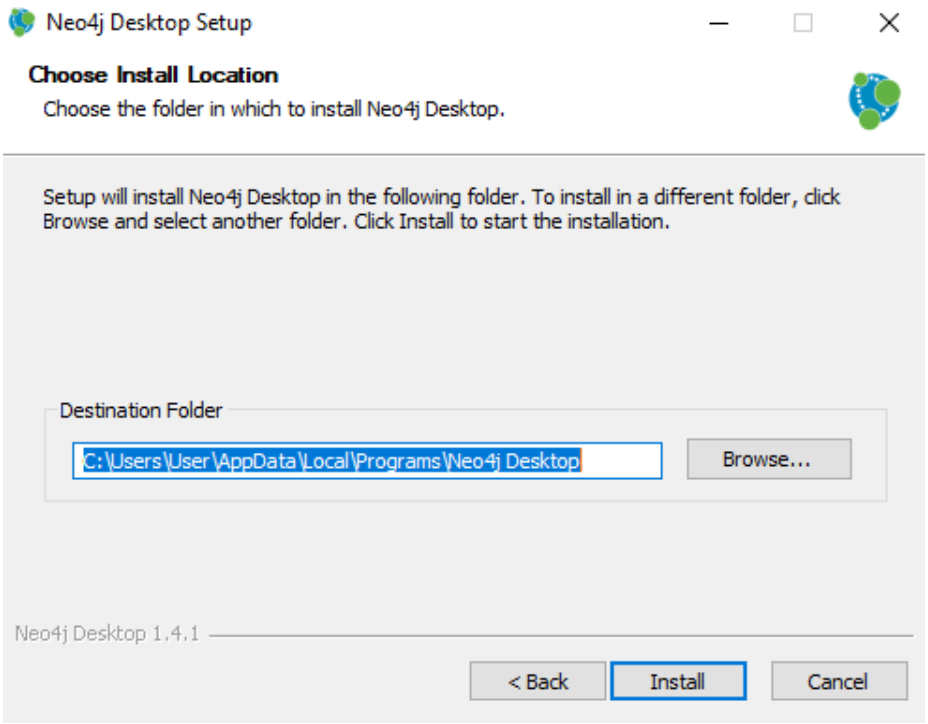
Αρχικά θα πρέπει να επισκεφθούμε το δικτυακό τόπο της Neo4j https://neo4j.com και να επιλέξουμε Products και στη συνέχεια Download Desktop. Για να προχωρήσουμε στη λήψη της εφαρμογής εγκατάστασης θα πρέπει να συμπληρώσουμε τα στοιχεία μας και θα μας εμφανιστεί ένα Softwarekey το οποίο θα το χρησιμοποιήσουμε με την ολοκλήρωση της εγκατάστασης για την ενεργοποίησή της.

Ξεκινάμε την εγκατάσταση της βάσης δεδομένων Neo4j και εμφανίζετε η πρώτη οθόνη για επιλογή της διαθεσιμότητας της βάσης στους χρήστες του υπολογιστή:



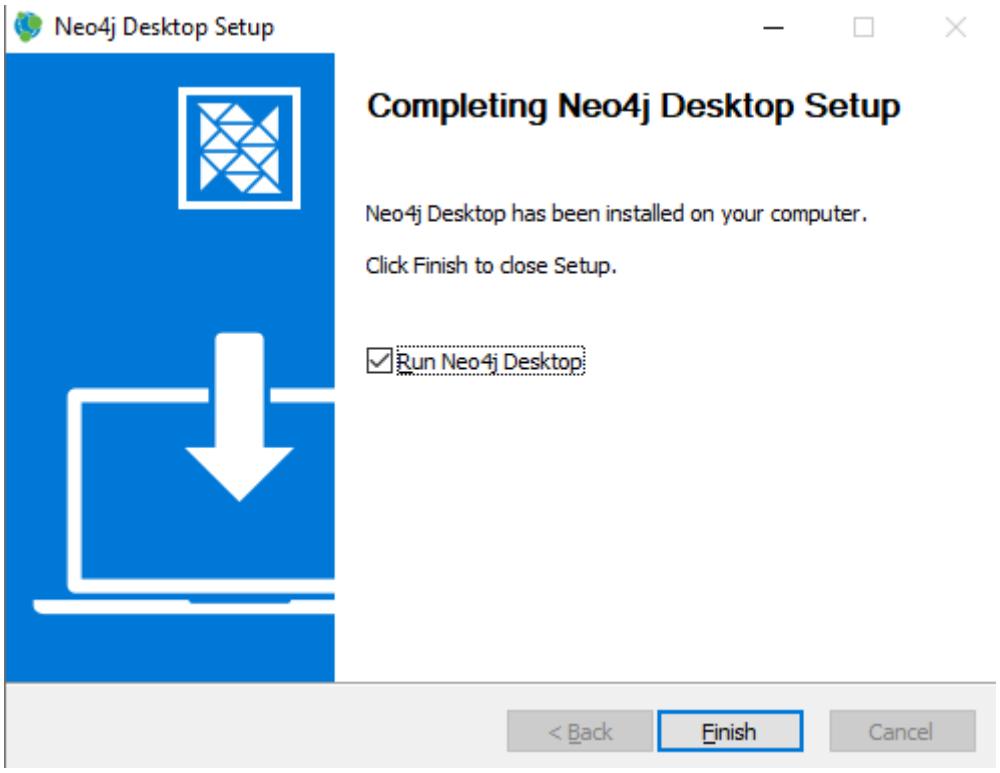
Εικόνα 23Επιλογή χρήστη για εγκατάσταση Neo4j

Κατόπιν επιλέγουμε τον κατάλογο εγκατάστασης και πατάμε το Install:



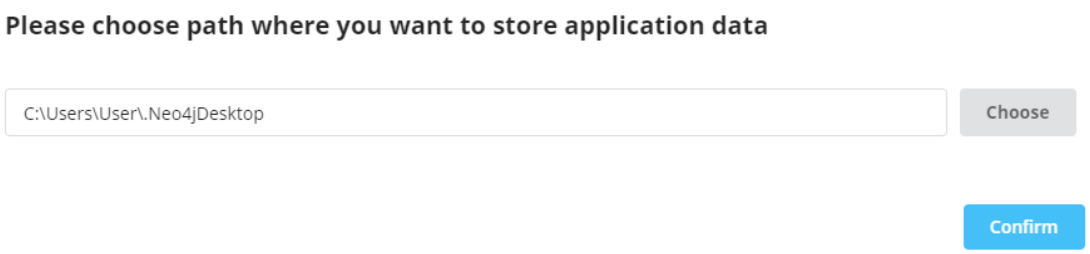
Εικόνα 24Επιλογή Destinationfolder

Η εγκατάσταση ολοκληρώνεται και εμφανίζεται σχετικό μήνυμα για την εκτέλεση του Neo4j.



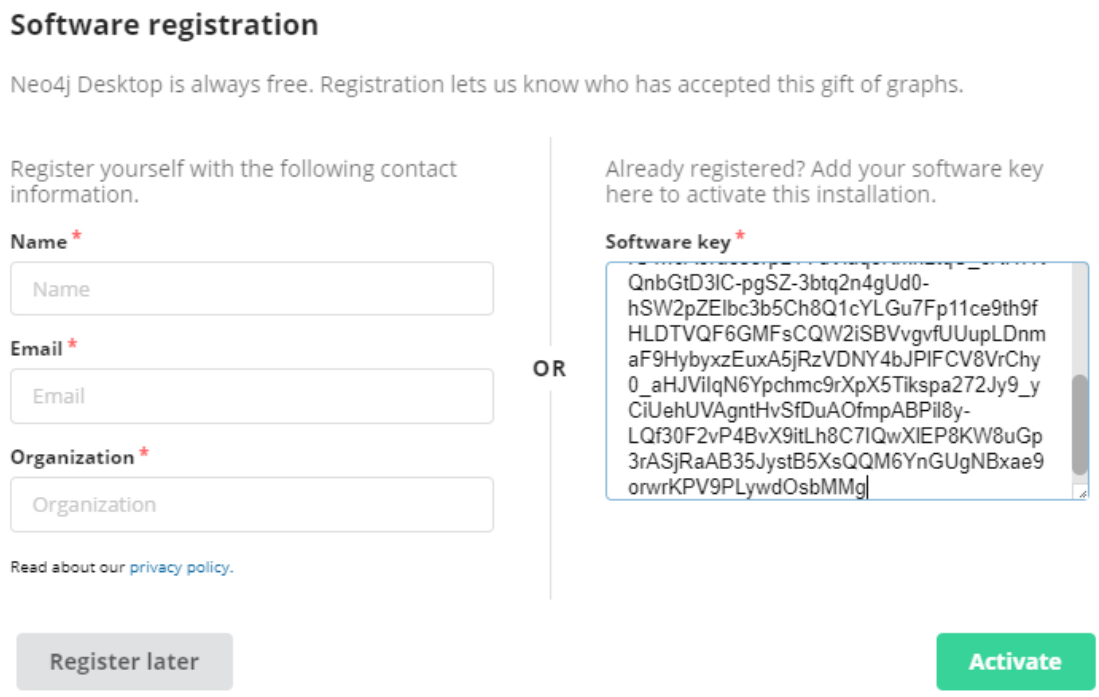
Εικόνα 25Ολοκλήρωση εγκατάστασης

Με την έναρξη της εφαρμογής θα πρέπει να αποδεχτούμε την άδεια χρήσης και επιλέγουμε τον κατάλογο όπου η εφαρμογή θα αποθηκεύει τα δεδομένα της και πατάμε το Confirm.



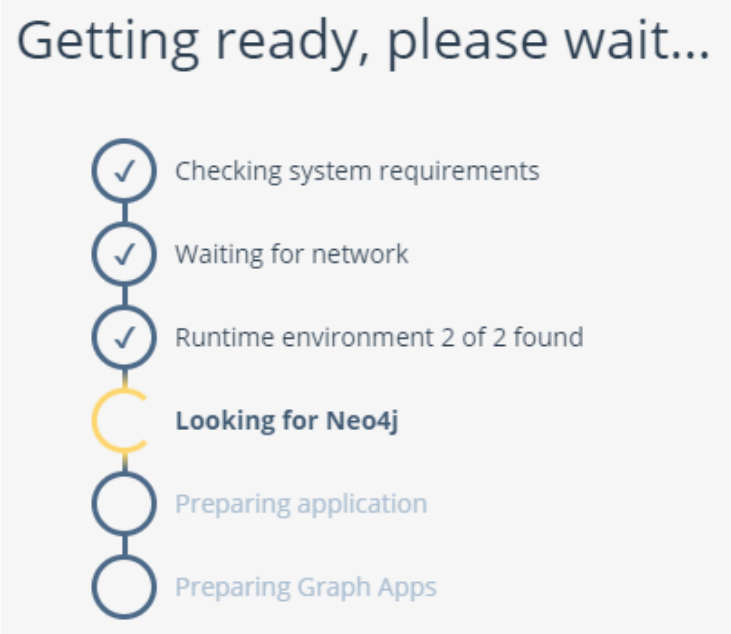
Εικόνα 26 Επιλογή αποθήκευσης application data

Στη συνέχεια θα πρέπει να την ενεργοποιήσουμε επικολλώντας το κλειδί που μας εμφάνισε κατά τη λήψη της εφαρμογής



Εικόνα 27 Softwareregistration

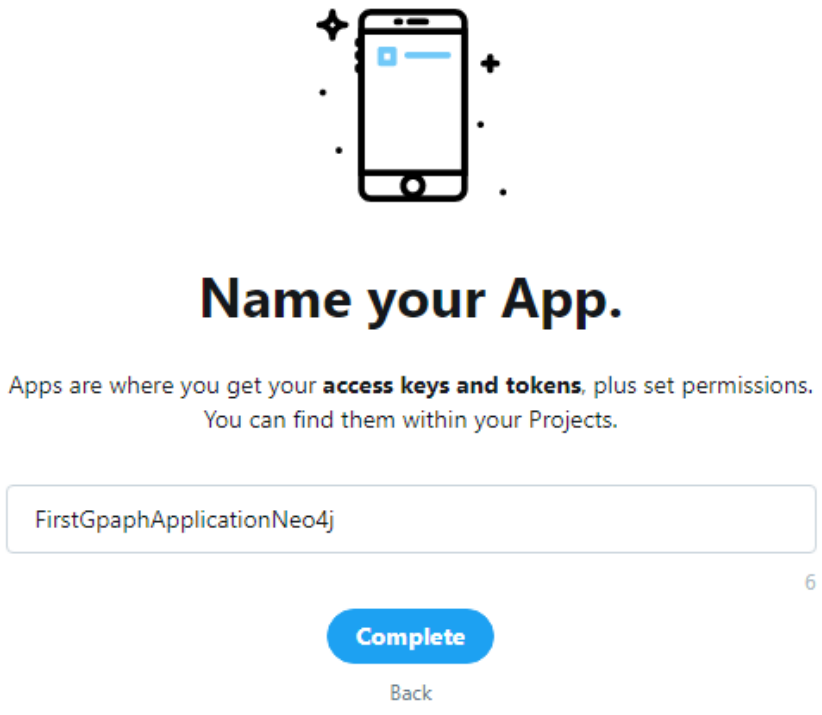
Με την ενεργοποίηση, προχωρά το τελικό στάδιο της προετοιμασίας για την πρώτη χρήση:



Εικόνα 28Πρώτη χρήση Neo4j

2.3.2 Twitter

Στη συνέχεια πρέπει να συνδεθούμε στο Twitter και να δημιουργήσουμε ένα νέο Application



Εικόνα 29 Όνομα εφαρμογής

Αν πατήσουμε το Complete θα εμφανιστούν τα ακόλουθα στοιχεία τα οποία θα τα χρησιμοποιήσουμε στο πρόγραμμά μας έτσι ώστε να αντλήσουμε τα στοιχεία που θέλουμε από το Twitter.

- **APIkey**: Im2lEiMPVX0JBGCKuGVGPlGB7
- **APIsecretkey**: TGLDEEBgsV9hragKspDcrjuUCdeTaA0lTkaSijqOJ0mbBIXd4m
- **BearerToken**: AAAAAAAAAAAAAAAAAAAAAPkXMAEAAAAA4f8TqlybUcj17aQ6SxVFhLfzoyc%3DuQm3xOzQK57Xy6lsGd8oFttMbsUNWsMdqS3iALx5nTREmp4fll
- **Access token**: 1352620927222231046-JVHFLmci1rmSetr81uEuWS15VxPXn5
- **Accesstokensecret**: pojfaxT2LroEuP5Kt79Q2O6OGEc8I8YCpT3XAT8as8QLf

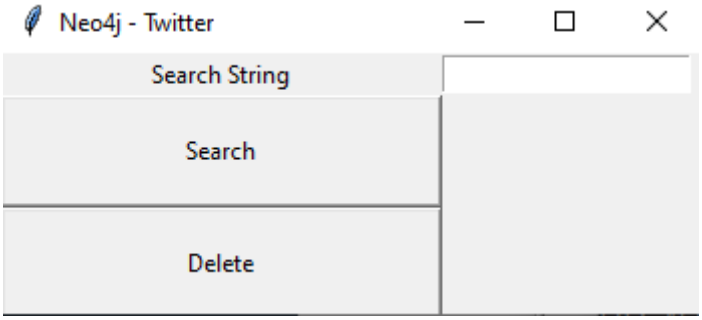
2.3.3 Pythonκαι Βιβλιοθήκες

Όπως αναφέραμε η εφαρμογή μας θα υλοποιηθεί σε Python, επομένως θα πρέπει να κατεβάσουμε την python και να την εγκαταστήσουμε.
Η λήψη θα γίνει αν επισκεφτούμε το δικτυακό τόπο https://www.python.org και επιλέξουμε Downloads και θα γίνει λήψη της τελευταίας έκδοσης, στην περίπτωσή μας η 3.9.1. Αφού ολοκληρωθεί η λήψη προχωράμε στην εγκατάσταση της Python στον υπολογιστή μας ακολουθώντας τις οδηγίες που εμφανίζονται στην οθόνη.
Από την Python θα χρησιμοποιήσουμε δύο βιβλιοθήκες την tweepy και την neo4j τις οποίες θα εγκαταστήσουμε με βάση τις οδηγίες που ακολουθούν:

- Tweepy: για σύνδεση στο Twitter, αναζήτηση και άντληση tweet.
- Neo4j: για σύνδεση στη βάση δεδομένων Neo4j και αποστολή ερωτημάτων στη γλώσσα Cypher.
- Tkinter: για τη δημιουργία του GUI.

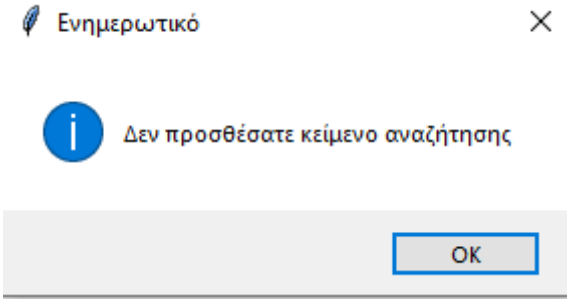
2.4 Δυνατότητες

Όταν εκτελέσουμε την εφαρμογή θα εμφανιστεί ένα παράθυρο όπου μπορούμε να εισάγουμε τη λέξη που θέλουμε να αναζητήσουμε στο Twitter.



Εικόνα 30 Κεντρική οθόνη προγράμματος

Αν δεν εισάγουμε λέξη τότε θα εμφανιστεί ενημερωτικό μήνυμα και δε θα εκτελεστεί το ερώτημα προς το Twitter.



Εικόνα 31 Ενημερωτικό μήνυμα

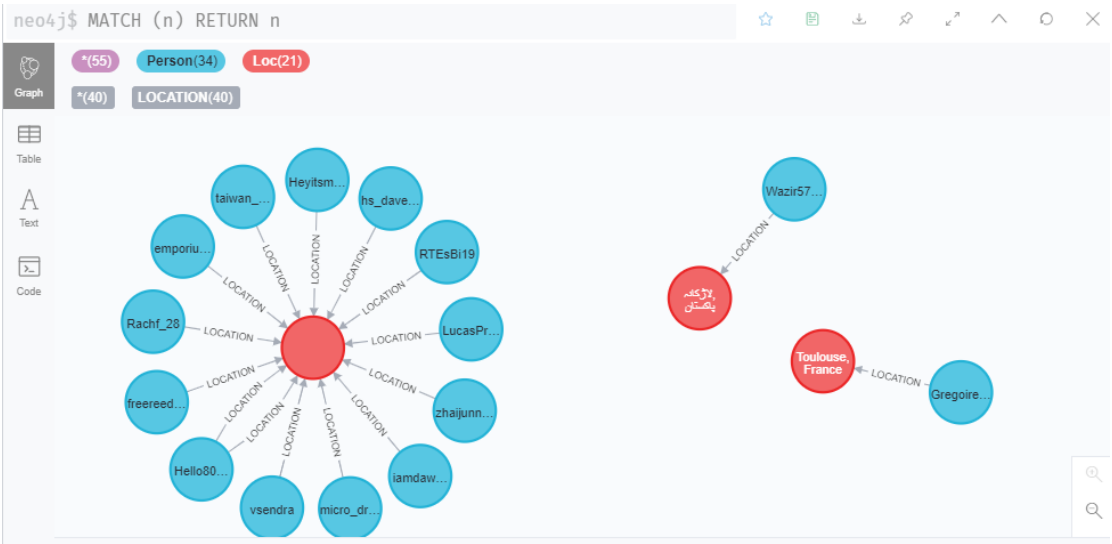
Αν η λέξη που εισάγουμε δεν περιέχει τον χαρακτήρα # τότε θα τον προσθέσουμε και θα εκτελέσουμε την αναζήτηση. Στην αναζήτηση θα ανακτήσουμε το όνομα και την τοποθεσία του ανθρώπου που έχει κάνει το tweet.

Τα αποτελέσματα του ερωτήματος στο Twitterεμφανίζονται και σε μορφή κειμένου (στο παράθυρο Commandαπ' όπου εκτελούμε το πρόγραμμα) και ένα μικρό τμήμα τους είναι το ακόλουθο (στην αρχή της γραμμής είναι η τοποθεσία – όπου υπάρχει - και μετά το όνομα):

Heyitsme19_18
Paris, France CynthiaFleury
zhaijunnase
Paris ThomasCoste
لاڑکانہ, پاکستان *Wazir57059255*
freereed59

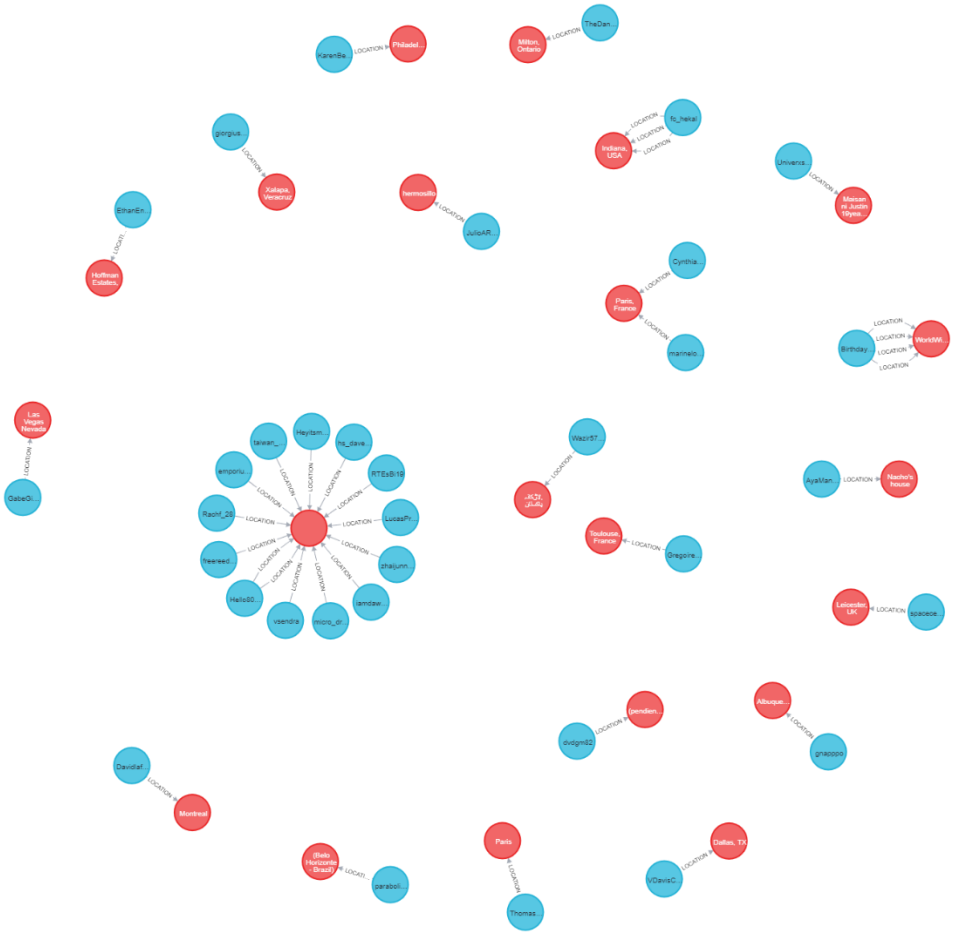
Hello80010937
WorldWide BirthdayMates
WorldWide BirthdayMates
Paris, France marineloyen
Indiana, USA fc_hekal
Philadelphia, PAKarenBeChirico

Τα παραπάνω αποτελέσματα εισάγονται στη βάση δεδομένων Neo4jδημιουργώντας τους κατάλληλους κόμβους και σχέσεις. Στην εικόνα που ακολουθεί μπορούμε να τα δούμε ένα τμήμα του γράφου που δημιουργήθηκε.



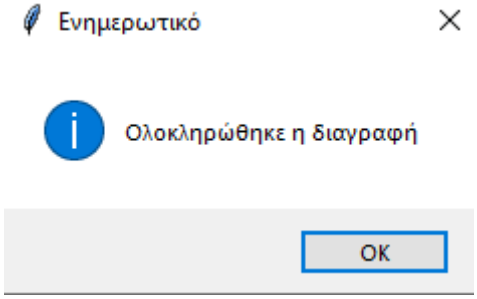
Εικόνα 32 Τμήμα του γράφου που δημιουργήθηκε

Και συνολικά με εξαγωγή από το Neo4j έχουμε το γράφο:



Εικόνα 33 Ο συνολικός γράφος

Αν πατήσουμε το πλήκτρο Deleteθα εκτελεστεί η κατάλληλη εντολή της γλώσσας Cypherστη βάση δεδομένων Neo4j για να διαγράψει όλους τους κόμβους και τις σχέσεις και θα εμφανιστεί το ακόλουθο μήνυμα:



Εικόνα 34 Ενημερωτικό μήνυμα ολοκλήρωσης διαγραφής

ΚΕΦΑΛΑΙΟ 3. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

3.1 Συμπεράσματα

Ο σκοπός της πτυχιακής ήταν να μελετηθούν οι NoSQLβάσεις δεδομένων δίνοντας όμως ιδιαίτερη βαρύτητα στις βάσεις δεδομένων γράφων. Παρουσιάσαμε εν συντομία τις σχεσιακές βάσεις δεδομένων και στη συνέχεια μελετήσαμε τις NoSQLβάσεις. Είδαμε την ανάγκη που ήρθαν να καλύψουν και τα πλεονεκτήματα που προσφέρουν καθώς και τις κατηγορίες που μπορούμε να τις διακρίνουμε.

Στη συνέχεια προχωρήσαμε στη μελέτη της κατηγορίας των βάσεων δεδομένων γράφων και τη συγκρίναμε με τις σχεσιακές βάσεις δεδομένων. Παρουσιάσαμε συνοπτικά τη βάση δεδομένων γράφων Neo4j, μια από τις πιο δημοφιλής βάση δεδομένων καθώς και τη γλώσσα ερωτημάτων που προσφέρει την Cypher.

Τέλος υλοποιήσαμε μια μικρή εφαρμογή για να αντλήσουμε δεδομένα από ένα κοινωνικό δίκτυο και να τα εισάγουμε στη Neo4j. Επιλέξαμε το Twitterκαι υλοποιήσαμε μια εφαρμογή σε Pythonπου αντλεί tweetκαι εισάγει το όνομα και την τοποθεσία του ανθρώπου που το έκανε.

3.2 Προτάσεις – Μελλοντικές επεκτάσεις

Η εφαρμογή η οποία αναπτύξαμε ήταν κυρίως για να ελέγξουμε τον τρόπο λειτουργίας και να δούμε τις δυνατότητες των βάσεων δεδομένων γράφων. Σίγουρα μπορεί να εξελιχτεί. Για παράδειγμα, εκτός από τη δυνατότητα να εισάγουμε ένα hashtag, θα μπορούσαμε:

- Να εισάγουμε ημερομηνίες από και έως
- Να δημιουργούμε ένα πιο πλούσιο γράφημα το οποίο να περιέχει περισσότερες πληροφορίες για τα άτομα που έκαναν τα tweet.
- Δημιουργία συνθέτου προγράμματος με AI/MLανάλυσης συναισθημάτων για το περιεχόμενο κάποιου post η κάποιου προϊόντος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ξενόγλωσση

[1] AWS (n.d.) WhatisaRelationalDatabase?, Διαθέσιμοαπόhttps://aws.amazon.com/relational-database/

[2] Berg K, Seymour T. J. (2012) History Of Databases, Διαθέσιμοαπόhttps://www.researchgate.net/publication/298332910_History_Of_Databases

[3] Cox G. (2017) Introduction to Graph Databases, Διαθέσιμοαπόhttps://www.compose.com/articles/introduction-to-graph-databases/

[4] DB-Engines (2021), DB-Engines Ranking of Graph DBMS, Διαθέσιμοαπόhttps://db-engines.com/en/ranking/graph+dbms

[5] George J. (2020) Graph Databases for the Public Sector, Διαθέσιμοαπόhttps://towardsdatascience.com/graph-databases-for-the-public-sector-1a50d0563fff

[6] Guru99 (n.d.) NoSQL Tutorial: Types of NoSQL Databases, What is & Example, Διαθέσιμοαπόhttps://www.guru99.com/nosql-tutorial.html

[7] MongoDB (n.d.) What is NoSQL?, Διαθέσιμοαπόhttps://www.mongodb.com/nosql-explained

[8] neo4j (n.d.) Graph Modeling Guidelines, Διαθέσιμοαπόhttps://neo4j.com/developer/guide-data-modeling/

[9] neo4j (2016) The Definitive Guideto Graph Databasesfor the RDBMS Developer, Διαθέσιμοαπόhttps://neo4j.com/whitepapers/rdbms-developers-graph-databases-ebook/

[10] neo4j (2020) The Neo4j Cypher Manual v4.2, Διαθέσιμοαπόhttps://neo4j.com/docs/cypher-manual/current/

[11] SabalTech (2020) Introduction to Graph Databases, Διαθέσιμοαπόhttps://www.sabaltech.com/post/introduction-to-graph-databases

[12] Schaefer L. (n.d.)NoSQL vs SQL Databases, Διαθέσιμοαπόhttps://www.mongodb.com/nosql-explained/nosql-vs-sql

[13] Schulz Y. (2020) A quick primer on graph databases, Διαθέσιμοαπόhttps://www.itworldcanada.com/blog/a-quick-primer-on-graph-databases/434215

[14] tutorialsPoind (n.d.) Neo4j – Overview, Διαθέσιμοαπόhttps://www.tutorialspoint.com/neo4j/neo4j_overview.htm

[15] Wu M. (2019) What Are the Major Advantages of Using a Graph Database?, Διαθέσιμοαπόhttps://dzone.com/articles/what-are-the-pros-and-cons-of-using-a-graph-databa

[16] Zhang D. (2020) System Design Topics: CAP Theorem, Διαθέσιμοαπόhttp://dannyzhang.run/2020/03/21/system-desing-1/

Παράρτημα 1 - Κώδικας Προγράμματος

```
fromtkinterimport *
from tkinter import messagebox
import tweepy as tw
from neo4j import GraphDatabase
from neo4j.exceptions import ServiceUnavailable

uri = "bolt://localhost:7687" #διευθυνση τοπικού neo4j
driver = GraphDatabase.driver(uri, auth=("neo4j", "1413")) #username, password



def deleteALL(tx):
tx.run("MATCH (n) "
        "DETACH DELETE n")

def create_and_return_friendship(tx, person1_name, person2_name): #μέθοδος για δημιουργία κόμβων και διασύνδεση αυτών
query = (
      "MERGE (p1:Loc { city: $person1_name }) "
      "MERGE (p2:Person { name: $person2_name }) "
      "CREATE (p2)-[:LOCATION]->(p1) "
      "RETURN p1, p2"
   )
result = tx.run(query, person1_name=person1_name, person2_name=person2_name)

try:
return [{"p1": record["p1"]["city"], "p2": record["p2"]["name"]}
for record in result]
except ServiceUnavailable as exception:
logging.error("{query} raised an error: \n {exception}".format(
        query=query, exception=exception))
```

```
    raise

#από εδώ και κάτω η σύνδεση με το twitter
consumer_key= 'Im2lEiMPVX0JBGCKuGVGPlGB7'
consumer_secret= 'TGLDEEBgsV9hragKspDcrjuUCdeTaA0lTkaSijqOJ0mbBIXd4m'
access_token= '1352620927222231046-JVHFLmci1rmSetr81uEuWS15VxPXn5'
access_token_secret= 'pojfaxT2LroEuP5Kt79Q2O6OGEc8I8YCpT3XAT8as8QLf'

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


window = Tk()
window.geometry('350x220')
window.title('Neo4j - Twitter')
lbl = Label(window, text="Search String",width=10)
lbl.grid(column=0, row=0)
lbl2 = Label(window, text="Search Date",width=10)
lbl2.grid(column=0, row=1)
lbl3 = Label(window, text="No. of posts",width=10)
lbl3.grid(column=0, row=2)
txt = Entry(window,width=20)
txt.grid(column=1, row=0)
txt2 = Entry(window,width=20)
txt2.grid(column=1, row=1)
txt3 = Entry(window,width=20)
txt3.grid(column=1, row=2)


def clicked():
res = txt.get()
if len(res)>0:
if (res.find('#')==-1):
```

```
res='#'+res
tweets = tw.Cursor(api.search,
                   q=res,
lang="en",
since=txt2.get()).items(int(txt3.get())) #πόσα αποτελέσματα

    users_locs = [[tweet.user.screen_name, tweet.user.location] for tweet in tweets]
with driver.session() as session:
for x in range(len(users_locs)):  #για όλα τα αποτελέσματα
print("User:"+users_locs[x][1]+",                    Location:"+users_locs[x][0])
#ταεμφανίζωκαιμετάταπροσθέτωστηβάση
session.write_transaction(create_and_return_friendship,          users_locs[x][1],
users_locs[x][0])

else:
messagebox.showinfo('Ενημερωτικό',"Δεν προσθέσατε κείμενο αναζήτησης")
def clearNeo4j():
with driver.session() as session:
    session.write_transaction(deleteALL)
messagebox.showinfo('Ενημερωτικό',"Ολοκληρώθηκε η διαγραφή")

btn = Button(window, text="Search", command=clicked,height = 3, width = 30)
btn.grid(column=0, row=4)
btn = Button(window, text="Delete", command=clearNeo4j,height = 3, width = 30)
btn.grid(column=0, row=6)
window.mainloop()
```