



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Π.Μ.Σ. ΕΠΙΣΤΗΜΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ ΥΠΟΛΟΓΙΣΤΩΝ

**“Μελέτη Βάσεων Δεδομένων Γράφων και Πειραματική
Αξιολόγηση”**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μπαρέκα Μαρία

Επιβλέπων Καθηγητής: Βασιλάκης Κωνσταντίνος

05.2018, Τρίπολη

Περίληψη

Στη παρούσα διπλωματική εργασία εξετάζονται, οι βάσεις δεδομένων γράφων (Graph Data Bases). Αρχικά, γίνεται μία σύντομη επισκόπηση των σχεσιακών βάσεων δεδομένων και κατόπιν παρουσιάζονται οι βάσεις δεδομένων NoSQL, αναπτύσσεται η τεχνολογία τους και πραγματοποιείται μια σύγκριση ανάμεσα τους.

Στη συνέχεια, η διπλωματική εργασία εστιάζει στη νέα γενιά βάσεων, τις βάσεις δεδομένων γράφων, παρουσιάζοντας τη δομή τους και τους λόγους επιλογής τους. Επιπρόσθετα, περιγράφονται ορισμένες από τις πιο διαδομένες βάσεις δεδομένων γράφων. Ακολουθεί η περιγραφή της Cayley, μιας βάσης δεδομένων τύπου γράφων, καθώς και της Gremlin, μιας γλώσσας ερωτημάτων η οποία χρησιμοποιείται από την Cayley. Τέλος, υπάρχει ένα σενάριο χρήσης εφαρμογής της Cayley στην οποία γίνεται χρήση της γλώσσας ερωτημάτων Gremlin.

Abstract

This thesis examines the technology of graph databases. Initially a brief overview of relational databases is given, and subsequently the NoSQL database branch is presented, their technology is detailed and a comparison is given.

Afterwards, this thesis focuses on the new generation of databases, i.e. graph databases, presenting their structure and their benefits. Furthermore, a number of the most widespread databases are described in more detail. This is followed by a detailed presentation of Cayley, which is a graph database, and Gremlin, a query language used by Cayley. Finally, a usage scenario for Cayley, including the use of the Gremlin language, is presented.

Περιεχόμενα

Περίληψη	2
Abstract	3
Περιεχόμενα	4
Κατάλογος Εικόνων	1
Κατάλογος Πινάκων	2
1. Εισαγωγή	1
2. Βιβλιογραφική Ανασκόπηση	3
2.1 Βάσεις Δεδομένων και απαραίτητα χαρακτηριστικά	3
2.2 Ιδιότητες ACID	3
2.3 Θεώρημα CAP	5
2.4 Σχεσιακά Συστήματα Βάσεων Δεδομένων	7
2.5 NoSql	8
2.5.1 Τεχνολογίες NoSql	9
2.5.1.1 Column-oriented (Στηλοκεντρικές Βάσεις Δεδομένων)	9
2.5.1.2 Key-values Stores (Αποθήκες ζευγών κλειδίων – τιμών)	11
2.5.1.3 Document-based stores (Αποθήκες εγγράφων)	12
2.5.1.4 Graph Databases	14
2.6 NoSql ή Βάσεις Δεδομένων	15
3. Βάσεις Δεδομένων γράφων (Graph Data Base)	17
3.1 Δομή Βάσης Δεδομένων Γράφων	17
3.2 Πλεονεκτήματα Βάσεων Δεδομένων Γράφων	18
3.3 Σύγκριση Βάσεων Δεδομένων Γραφών και Σχεσιακών Βάσεων Δεδομένων	19
3.4 Συστήματα Βάσεων Δεδομένων Γράφων	20
3.3.1 Neo4j	21
3.3.2 AllegroGraph	21
3.3.3 Oracle Spatial and Graph	23
3.3.4 OpenLink Virtuoso	24
3.3.5 GraphDB	25
3.3.6 InfiniteGraph	26
3.3.7 Cayley	27
4. CAYLEY	28
4.1 Περιγραφή της Cayley	28
4.2 Εγκατάσταση της Cayley	29
4.3 Το περιβάλλον της Cayley	30

4.3.1 Η Γλώσσα Gremlin	32
4.3.2 Πλεονεκτήματα της Gremlin	33
4.4 Δημιουργία ερωτημάτων στην Gremlin	35
Στοιχείο V	35
Στοιχείο E	36
Στοιχείο id	36
Στοιχείο in	37
Στοιχείο order	37
Στοιχείο out	38
Στοιχείο outE	38
Στοιχείο outV	38
Στοιχείο has	39
5. Σενάριο χρήσης της Caley σε εφαρμογή	41
5.1 Αρχιτεκτονική	41
5.2 Σενάριο Χρήσης Εφαρμογής	41
5.3 Μεταφόρτωση αρχείου στη Cayley	46
5.4 Δημιουργία Ερωτημάτων	47
5.4.1 Δημιουργία Ερωτημάτων στο πεδίο query	47
5.4.2 Δημιουργία Ερωτημάτων στο πεδίο Visualize	50
6. Σύνοψη	55
7. Βιβλιογραφία	56
Βιβλία	56
Άρθρα	56
Ιστότοποι	56
Παρουσιάσεις	57

Κατάλογος Εικόνων

Εικόνα 1: Ιδιότητες ACID	4
Εικόνα 2 Θεώρημα CAP	6
Εικόνα 3 – Συνδυασμοί υποστηριζόμενων ιδιοτήτων βάσει του θεωρήματος CAP	7
Εικόνα 4 – Απεικόνιση hypernodes & hyperedges. Η υπερ-ακμή e_1 εκφράζει μία σχέση ανάμεσα στις κορυφές v_1, v_2, v_3 , ενώ η ακμή e_3 εκφράζει μία σχέση ανάμεσα στις κορυφές v_3, v_5 και v_6 κ.ο.κ.	18
Εικόνα 5 – Αναπαράσταση ενός στιγμιότυπου βάσης δεδομένων γράφων	18
Εικόνα 6 -Σύγκριση σχεσιακών βάσεων δεδομένων και βασέων δεδομένων γράφων στην αύξηση των δεδομένων	20
Εικόνα 7 - Αλλαγή διεύθυνση φακέλου προορισμού.....	29
Εικόνα 8 - Διεύθυνση Cayley.....	30
Εικόνα 9 - Περιβάλλον Cayley	30
Εικόνα 10 - Περιβάλλον Cayley	31
Εικόνα 11 - Απεικόνιση γράφου	33
Εικόνα 12 - Απεικόνιση γράφου	35
Εικόνα 13 -Μεταφόρτωση Βάσης Γράφου στην Cayley	47
Εικόνα 14 - Αποτελέσματα Ερωτήματος που εμφανίζει όλα τα παιδιά της Γεωργίας	48
Εικόνα 15 - Αποτέλεσμα Ερωτήματος όσων έχουν γεννηθεί το 1982.....	49
Εικόνα 16 - Εμφάνιση Ερωτήματος του έτος γέννησης των παιδιών του Βασίλη	50
Εικόνα 17 - Οι Κόμβοι των παιδιών της Γεωργίας.....	51
Εικόνα 18 - Οι κόμβοι εμφανίζουν τον άντρα, τον αδελφό και τα παιδιά της Γεωργίας.....	52
Εικόνα 19 - Οι κόμβοι των εγγονιών της Γεωργίας.....	52
Εικόνα 20 - Οι κόμβοι των παιδιών και των εγγονιών της Γεωργίας	54

Κατάλογος Πινάκων

Πίνακας 1 – Μοντέλο δεδομένων σε μία σχεσιακή βάση δεδομένων.....	10
Πίνακας 2 – Αναπαράσταση αποθήκευσης δεδομένων σχεσιακής βάσης δεδομένων.....	10
Πίνακας 3 - Αποθήκευση δεδομένων με την στηλοκεντρική μέθοδο	11
Πίνακας 4 – Αναπαράσταση της αποθήκευσης δεδομένων με την μέθοδο αποθήκης ζευγών κλειδιών - τιμών	12
Πίνακας 5 – Αναπαράσταση εγγράφου.....	13
Πίνακας 6 – Αναπαράσταση εγγράφου με διαφορετικά δεδομένα	13
Πίνακας 7 – Αναπαράσταση σύνθετου εγγράφου.....	14
Πίνακας 8 – Δεδομένα Σεναρίου Χρήσης Εφαρμογής.....	46

1. Εισαγωγή

Οι Βάσεις Δεδομένων και τα συστήματα διαχείρισής τους αποτελούν εδώ και αρκετές δεκαετίες αναπόσπαστο κομμάτι των συστημάτων λογισμικού αφού αποθηκεύουν και διαχειρίζονται τις πληροφορίες, οι οποίες με τη σειρά τους είναι το συνηθέστατα πλέον σημαντικό συστατικό των πληροφοριακών συστημάτων. Ο κύριος λόγος που τα συστήματα διαχείρισης βάσεων έχουν αποκτήσει τεράστια ανάπτυξη είναι διότι τόσο ο προγραμματιστές όσο και οι επιχειρήσεις ικανοποιούν τις απαιτήσεις τους για διαχείριση δεδομένων μέσα από αυτές.

Όλα αυτά τα χρόνια έχουν αναπτυχθεί αρκετά αξιόλογα λογισμικά συστημάτων διαχείρισης βάσεων δεδομένων, τα οποία είτε είναι ανοιχτού κώδικα είτε κλειστού κώδικα. Μερικά παραδείγματα συστημάτων βάσεων δεδομένων είναι το MariaDB, η οποία ανήκει στα ΣΔΒΔ ανοιχτού κώδικα αλλά και το Microsoft SQL Server το οποίο είναι ένα ΣΔΒΔ κλειστού κώδικα.

Όπως, αναφέρθηκε και προηγουμένως οι Σχεσιακές Βάσεις Δεδομένων (ΣΒΔ) χρησιμοποιούνται εδώ και αρκετές δεκαετίες από προγραμματιστές, ωστόσο οι μεγάλες υπολογιστικές και αποθηκευτικές απαιτήσεις των σημερινών εφαρμογών έχουν ωθήσει τις Βάσεις Δεδομένων στα όρια τους. Παράλληλα, η φύση των δεδομένων που χρησιμοποιούνται σε πολλές νέες κατηγορίες εφαρμογών, όπως οι εφαρμογές σημασιολογικού ιστού ή οι εφαρμογές ανάκτησης πληροφορίας, θέτουν νέες απαιτήσεις για τη διαχείριση των δεδομένων που δεν ικανοποιούνται με βέλτιστο τρόπο από τις κυρίαρχες τεχνολογίες βάσεων δεδομένων, δηλ. τις σχεσιακές βάσεις δεδομένων.

Γι' αυτό το λόγο, τα τελευταία χρόνια έχουν εμφανιστεί νέες αρχιτεκτονικές Βάσεων Δεδομένων, οι οποίες επιχειρούν να καλύψουν τις νέες ανάγκες. Οι αρχιτεκτονικές αυτές έχουν τη γενική ονομασία «NOSQL» (τα αρχικά της οποίας σημαίνουν Not Only SQL), και μερικές από τις “εταιρίες που τις χρησιμοποιούν είναι η Google, η Amazon αλλά και το Facebook.

Το σχεσιακό μοντέλο δυσκολεύεται αρκετά να υποστηρίξει τις σημερινές σύγχρονες εφαρμογές οι οποίες έχουν μεγάλες απαιτήσεις, αφού υπάρχουν αρκετοί περιορισμοί τόσο στην εσωτερική δομή της βάσης όσο και στους μηχανισμούς με τους οποίους οι

εφαρμογές αποκτούν πρόσβαση στα δεδομένα. Τα δυο πιο σημαντικά μειονεκτήματα που εμφανίζουν οι Βάσεις Δεδομένων:

1. Η δημιουργία και η διατήρηση του συστήματος της βάσης δεδομένων είναι μια διαδικασία ιδιαίτερα ακριβή.
2. Η κλιμάκωση τους είναι κακή.

Ακόμα, στα μειονεκτήματα των βάσεων δεδομένων θα προστεθεί το γεγονός ότι η ποσότητα και ο τύπος των δεδομένων, καθώς και το μέγεθός δηλώνονται κατά την αρχική σχεδίαση της βάσης δεδομένων. Ενώ, και η δομή των πινάκων των βάσεων δεδομένων είναι αρκετά περιοριστική για τα αντικείμενα του πραγματικού κόσμου που είναι πολύπλοκα.

Η δομή των πινάκων των βάσεων δεδομένων αποτελείται από γραμμές και στήλες, ενώ η διαίρεση των δεδομένων δημιουργεί αρκετούς πίνακες, οι οποίοι απαιτείται να συνδέονται μεταξύ τους γεγονός που οδηγεί σε μια αρκετά χρονοβόρα διαδικασία αποτίμησης ερωτήσεων αλλά και σχεδιασμού. Επομένως, η δομή των πινάκων μπορεί να χαρακτηριστεί περιοριστική για τα δεδομένα του πραγματικού κόσμου, τα οποία είναι περίπλοκα. Παράλληλα, για οποιαδήποτε αλλαγή ζητηθεί στη βάση δεδομένων απαιτείται παρέμβαση του διαχειριστή για την κατάλληλη διαμόρφωση του σχήματος της βάσης δεδομένων και τη βελτιστοποίηση της λειτουργίας της.

Όλα αυτά τα προβλήματα έχουν οδηγήσει στην μελέτη νέων συστημάτων διαχείρισης τα οποία να είναι ικανά να διαχειρίζονται αποτελεσματικά και αποδοτικά τα σύγχρονα δεδομένα. Ως απάντηση, στα μειονεκτήματα των βάσεων δεδομένων έρχεται η σχετικά καινούργια προσέγγιση της NoSql. Τα NoSql συστήματα μπορούν να χαρακτηριστούν εύκολα στη χρήση και την κατανόηση, προσαρμοστικά, ανθεκτικά, αποδοτικά, ενώ διαθέτουν υψηλή διαθεσιμότητα και επιτρέπουν την παράλληλη επεξεργασία ερωτημάτων. Αξίζει να σημειωθεί ότι οι πιο μεγάλοι δικτυακοί τόποι όπως είναι οι Google, το twitter, η Amazon, το LinkedIn, κ.τ.λ. έχουν επικεντρωθεί στην τεχνολογία συστημάτων NoSql, απομακρυνόμενοι σταδιακά από τη χρήση των σχεσιακών βάσεων.

2. Βιβλιογραφική Ανασκόπηση

Σε αυτήν την ενότητα, θα παρουσιαστούν τόσο η τεχνολογία των Σχεσιακών Βάσεων Δεδομένων, οι οποίες χρησιμοποιούνται εδώ και αρκετές δεκαετίες, όσο και των βάσεων δεδομένων NoSql, για τις οποίες παρατηρείται έντονο ενδιαφέρον κατά το τελευταίο διάστημα. Επίσης, θα πραγματοποιηθεί συγκριτική αξιολόγηση ανάμεσα στις δύο κατηγορίες βάσεων δεδομένων.

2.1 Βάσεις Δεδομένων και απαραίτητα χαρακτηριστικά

Με τον όρο *Βάση Δεδομένων* εννοείται ένα σύνολο πληροφοριών, οι οποίες τυπικά οργανώνονται σε ομάδες, με κάθε ομάδα να αντιστοιχεί σε αντικείμενα του πραγματικού κόσμου. Οι ομάδες είναι οργανωμένες με τέτοιο τρόπο ώστε να υπάρχουν λογικές σχέσεις μεταξύ τους. Οι λογικές σχέσεις επίσης αντιστοιχούν σε σχέσεις μεταξύ των ομάδων που υπάρχουν στον πραγματικό κόσμο. Συνολικά, μία βάση δεδομένων έχει σχεδιαστεί και υλοποιηθεί με σκοπό να χρησιμοποιηθεί για κάποιο σκοπό και να παρέχει συγκεκριμένες λύσεις σε έναν μικρόκοσμο.

Βασικό χαρακτηριστικό των βάσεων δεδομένων είναι ότι παρέχουν ένα κοινό τρόπο διαχείρισης των δεδομένων, προσφέροντας τη δυνατότητα εφαρμογής των βασικών λειτουργιών της πάνω στα αποθηκευμένα δεδομένα, τα οποία μπορεί να αλλάζουν συνεχώς. Επομένως, η βάση δεδομένων είναι ένα δυναμικό σύστημα αποθήκευσης και διαχείρισης δεδομένων και περιγράφει κάθε φορά την κατάσταση του μικρόκοσμου στον οποίο αναφέρεται και τον οποίο εξυπηρετεί.

2.2 Ιδιότητες ACID

Οι βάσεις δεδομένων, για να προστατέψουν την ακεραιότητα των δεδομένων, χρησιμοποιούν την έννοια των δοσοληψιών. Ουσιαστικά, οι δοσοληψίες είναι μια ενότητα εκτέλεσης ενός προγράμματος που προσπελαύνει και ενημερώνει διάφορα δεδομένα. Για να διασφαλιστεί η αξιοπιστία της δοσοληψίας, υφίσταται ένα σύνολο ιδιοτήτων με το ακρωνύμιο ACID (Atomicity, Consistency, Isolation, Durability) – ατομικότητα, συνέπεια, απομόνωση και μονιμότητα. Οι ιδιότητες αυτές περιγράφονται στη συνέχεια.

- **Atomicity (Ατομικότητα):** Η ιδιότητα της ατομικότητας υποδεικνύει ότι είτε όλες οι πράξεις της δοσοληψίας καταχωρούνται στη βάση δεδομένων, είτε δεν

καταχωρείται καμία. Επομένως, η ιδιότητα της ατομικότητας συμβάλλει στη διατήρηση της συνέπειας της βάσης δεδομένων, αφού αν αποτύχει ένα μέρος της δοσοληψίας τότε θα αποτύχει ολόκληρη η δοσοληψία και η βάση θα επιστρέψει στην αρχική της κατάσταση.

- **Consistency (Συνέπεια):** Η ιδιότητα της συνέπειας διασφαλίζει ότι η Βάση Δεδομένων διατηρείται σε μια συνεπή κατάσταση, συγκεκριμένα υποδεικνύοντας ότι κάθε δοσοληψία θα οδηγεί την βάση δεδομένων από τη μια συνεπή κατάσταση στην άλλη. Σε περίπτωση που μια δοσοληψία παραβιάζει κάποιο κανόνα της συνέπειας, ανακαλείται προκειμένου η ΒΔ να έχει μόνο έγκυρα δεδομένα. Ωστόσο επιτρέπεται κατά τη διάρκεια εκτέλεσης της δοσοληψίας να διέρχεται προσωρινά η βάση δεδομένων από μη συνεπείς καταστάσεις, καθώς ο έλεγχος για την τήρηση των κανόνων συνέπειας μπορεί να επιλεγεί να πραγματοποιείται κατά την επικύρωση της δοσοληψίας.
- **Isolation (Απομόνωση):** Η Απομόνωση δίνει τη δυνατότητα στη δοσοληψία να αγνοεί τις υπόλοιπες ταυτόχρονες συναλλαγές, υπό την έννοια ότι η κάθε δοσοληψία δεν επηρεάζεται από τις προσωρινές, μη συνεπείς καταστάσεις που οι άλλες δοσοληψίες δημιουργούν. Το χαρακτηριστικό της συγκεκριμένης ιδιότητας προσφέρει τη δυνατότητα τα δεδομένα της κάθε συναλλαγής να είναι διαθέσιμα μετά την ολοκλήρωση της συναλλαγής, ενώ μια δοσοληψία δεν γνωρίζει για τις υπόλοιπες δοσοληψίες οι οποίες πραγματοποιούνται παράλληλα, ούτε να εκτελεί η ίδια κάποιες ενέργειες για την αποτροπή της αλληλοπαρεμβολής, μια και αυτή διασφαλίζεται από το σύστημα διαχείρισης βάσεων δεδομένων.
- **Durability (Μονιμότητα):** Η ιδιότητα της Μονιμότητας παρέχει τα εγγύα πως όταν η δοσοληψία ολοκληρωθεί επιτυχώς, οι αλλαγές που έχει κάνει στη βάση δεδομένων υπάρχουν, ακόμα και αν προκύψουν αποτυχίες του συστήματος, όπως είναι για παράδειγμα η απώλεια ρεύματος.



Εικόνα 1: Ιδιότητες ACID

2.3 Θεώρημα CAP

Οι σημερινές εφαρμογές εκτελούνται σε πολλούς διακομιστές οι οποίοι είναι καταναμημένοι σε γεωγραφικές περιοχές, με αυτό τον τρόπο υπάρχει ισορροπημένη κυκλοφορία του δικτύου σε ένα σύνολο διακομιστών, ακόμα και όταν σε κάποια τοποθεσία κάποιος διακομιστής τεθεί εκτός λειτουργίας. Σε αυτές τις περιπτώσεις, η κίνηση των δεδομένων ανακατευθύνεται στους διακομιστές οι οποίοι παραμένουν σε λειτουργία.

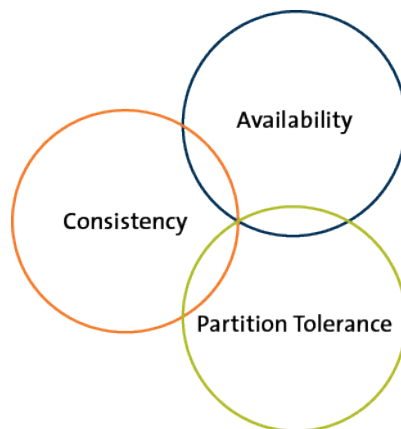
Ένα σύνολο ηλεκτρονικών υπολογιστών οι οποίοι συνεργάζονται αποκαλείται καταναμημένο σύστημα. Ένα καταναμημένο σύστημα δημιουργείται για να αυξηθεί η αποθηκευτική και υπολογιστική ικανότητα, προκειμένου να λυθούν προβλήματα τα οποία από μόνος του ένα υπολογιστής δεν είναι σε θέση να καταφέρει. Στην περίπτωση των εφαρμογών, ένα καταναμημένο σύστημα θέτει δυο στόχους:

1. Να παρέχει στον χρήστη συνεχή, χωρίς προβλήματα λειτουργία, διασφαλίζοντας την απρόσκοπτη πρόσβασή του σε υπηρεσίες όπως είναι για παράδειγμα η χρήση του Facebook. Αυτό επιτυγχάνεται μέσω της ύπαρξης πολλών υπολογιστών, οι οποίοι συνεργάζονται για την αδιάλειπτη λειτουργία της εφαρμογής.
2. Να επεξεργάζονται τα δεδομένα τα οποία προκύπτουν από την εφαρμογή.

Τα οφέλη των καταναμημένων συστημάτων είναι πολλά, αφού παρέχουν μεγαλύτερη υπολογιστική ισχύ και αποθηκευτικό χώρο προκειμένου να εξυπηρετούν τα δισεκατομμύρια αιτημάτων τα οποία προκύπτουν καθημερινά. Ωστόσο, η δημιουργία ενός συστήματος το οποίο συνδέεται με πολλούς υπολογιστές δεν είναι εύκολο εγχείρημα αφού προκύπτουν πολλές προκλήσεις.

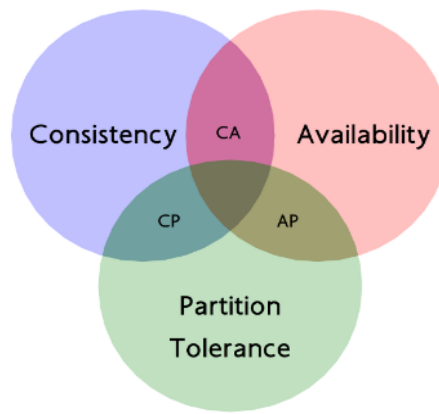
Επομένως, λόγω της ύπαρξης ένας αρκετά πολύπλοκου καταναμημένου συστήματος αποθήκευσης πληροφοριών αναπτύχθηκε το Θεώρημα CAP (Consistency, Availability, Partition tolerance – Συνεπεία, Διαθεσιμότητα, Ανοχή διαμερισμού), γνωστό και ως θεώρημα Brewer, όπου σύμφωνα με αυτό το θεώρημα είναι αδύνατο ένα καταναμημένο σύστημα να παρέχει εγγυήσεις για την ταυτόχρονη παροχή των τριών βασικών ιδιοτήτων, αλλά έχει τη δυνατότητα να υποστηρίζει μόνο δύο από τρεις βασικές ιδιότητες:

- **Συνέπεια (consistency):** η ιδιότητα αυτή αναπτύχθηκε και παραπάνω στις ιδιότητες ACID αλλά σύμφωνα με τον Brewer η κεντρική ιδέα είναι ότι όλοι οι κόμβοι μπορούν να βλέπουν τα ίδια δεδομένα την ίδια στιγμή, ακόμα και όταν τα δεδομένα της βάσης ενημερώνονται.
- **Διαθεσιμότητα (availability):** προϋποθέτει ότι το σύστημα αποκρίνεται σε κάθε αίτημα με μια απάντηση για να ενημερώσει για την επιτυχία του ή τυχόν επιτυχία, καθώς και τα δεδομένα που αποτελούν την απάντηση στο αίτημα, εφ' όσον υπάρχουν.
- **Ανοχή διαμερισμού (partition tolerance):** εγγυάται τη λειτουργία του συστήματος στην περίπτωση που ένα μέρος του συστήματος τεθεί εκτός λειτουργίας λόγω βλάβης στο δίκτυο, η οποία διαμερίζει τους κόμβους σε δύο ή περισσότερα υποσύνολα, όπου οι κόμβοι του κάθε υποσυνόλου μπορούν να επικοινωνήσουν μεταξύ τους αλλά όχι με τους κόμβους άλλων υποσυνόλων. Σε αυτή την περίπτωση το σύστημα πρέπει να είναι σε θέση και να αποκρίνεται στα μηνύματα που δέχεται.



Εικόνα 2 Θεώρημα CAP

Με βάση το θεώρημα CAP, υπάρχουν τρεις διαφορετικοί συνδυασμοί ζευγών βασικών ιδιοτήτων που μπορούν να υποστηρίζονται, όπως φαίνεται στο σχήμα που ακολουθεί.



Εικόνα 3 – Συνδυασμοί υποστηριζόμενων ιδιοτήτων βάσει του θεωρήματος CAP

CA: Ο συνδυασμός αυτός εγγυάται υψηλή διαθεσιμότητα και συνέπεια ωστόσο δεν εγγυάται ανοχή στον διαμερισμό, είναι γνωστός και ως *συνδυασμός υψηλής διαθεσιμότητας*. Σε αυτόν συνδυασμό οι ενημερώσεις διαχέονται σταδιακά από τον έναν κόμβο στους υπόλοιπους, γεγονός που οδηγεί τα δεδομένα σε συνεπή κατάσταση αφού εμφανίζονται τα πιο ενημερωμένα.

CP: Ο συνδυασμός αυτός παρέχει συνεπεία και αντοχή στον διαμερισμό αλλά δεν εγγυάται διαθεσιμότητα, είναι γνωστός και ως *συνδυασμός ισχυρής συνέπειας*. Το βασικό του πρόβλημα είναι ότι δεν παρέχει συνεχή πρόσβαση στα δεδομένα σε περίπτωση βλάβης

AP: Ο συνδυασμός αυτός εγγυάται διαθεσιμότητα και ανοχή στον διαμερισμό, αφού παρέχει συνεχή πρόσβαση στα δεδομένα ωστόσο υπάρχει περίπτωση να μην έχει γίνει ενημέρωση στα δεδομένα. Τα δεδομένα *τελικώς* θα ενημερωθούν, αλλά μπορεί να είναι ορατές μη συνεπείς μορφές τους, και γι' αυτό ο συνδυασμός ονομάζεται *συνδυασμός ενδεχόμενης συνέπειας* (eventual consistency).

2.4 Σχεσιακά Συστήματα Βάσεων Δεδομένων

Παράλληλα με την πρόοδο της θεωρίας των σχεσιακών βάσεων δεδομένων αναπτύχθηκαν τα Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΣΔΒΔ, Relational Database Management Systems – RDBMS), δηλαδή συγκεκριμένα συστήματα που υλοποιούν και καθιστούν διαθέσιμες σχεσιακές βάσεις δεδομένων. Παραδείγματα σχεσιακών συστημάτων βάσεων δεδομένων είναι η Oracle, ο SQL server και η MySQL.

Στο σχεσιακό μοντέλο, η βάση δεδομένων αποτελείται από στοιχεία που ονομάζονται *πίνακες* και υφίσταται σχέσεις μεταξύ αυτών. Οι πίνακες, με την σειρά τους, περιέχουν εγγραφές (records), οι οποίες στο σχεσιακό μοντέλο αντιστοιχούν στις γραμμές του κάθε πίνακα όπου η σειρά εμφάνισης τους δεν έχει απολύτως καμία σημασία.

Για κάθε πίνακα, είναι απαραίτητο κάθε γραμμή (εγγραφή) να μπορεί να οριστεί μοναδικά για αυτό το λόγο χρησιμοποιείται το *πρωτεύον κλειδί*. Το πρωτεύον κλειδί αποτελείται από ένα ή περισσότερα πεδία του πίνακα, όπου κάθε συνδυασμός τιμών τους υπάρχει το πολύ μια φορά στις γραμμές του πίνακα. Υφίσταται επίσης η έννοια του *ξένου κλειδιού*, η οποία προσδιορίζει τον τρόπο με τον οποίο συνδέονται οι πίνακες μεταξύ τους μέσω των σχέσεων. Ουσιαστικά, ο συνδυασμός τιμών των στηλών που ανήκουν σε ένα ξένο κλειδί σε έναν πίνακα, πρέπει να εντοπίζεται σε αντίστοιχες στήλες ενός άλλου πίνακα.

2.5 NoSql

Η προσέγγιση NoSQL, το όνομα της οποίας είναι συντομογραφία της φράσης “Not Only SQL”, ερευνάται από τους προγραμματιστές σαν απάντηση στις νέες απαιτήσεις που έχουν αναπτυχθεί. Η πρώτη παρουσίαση των βάσεων NoSQL πραγματοποιήθηκε το 2009, εμφανίζοντας έναν νέο τρόπο με τον οποίο μπορούν οι πληροφορίες του διαδικτύου να αποθηκευτούν.

Ακόμα, δεν υπάρχει ένας ακριβής ορισμός για το τι περιλαμβάνει ένα σύστημα NoSQL, ωστόσο μπορεί να θεωρηθεί σαν μια ομάδα συστημάτων βάσεων δεδομένων οι οποίες αναπτύσσονται με σκοπό τον αποδοτικό χειρισμό μεγάλων ποσοτήτων πληροφορίας χρησιμοποιώντας έναν διαφορετικό τρόπο από τον κλασσικό που είναι ο σχεσιακός τρόπος αποθήκευσης σε συνδυασμό με τη γλώσσα SQL. Υπάρχουν εταιρίες και οργανισμοί οι οποίοι πραγματοποιούν ευρεία χρήση αυτής της τεχνολογίας προκειμένου να αντεπεξέλθουν στις τεράστιες ποσότητες δεδομένων οι οποίες παράγονται από τους χρήστες τους. Μερικές από τις εταιρίες αυτές είναι η Amazon, το Facebook και η Google.

Τα συστήματα NoSQL είναι σχεδιασμένα έτσι ώστε να αποθηκεύουν μεγάλες ποσότητες δεδομένων ενώ παράλληλα είναι ικανά να επεξεργάζονται τα δεδομένα τα οποία καταναμημένα σε μεγάλο αριθμό servers. Οι Βάσεις Δεδομένων NoSQL αποτελούνται από αδόμητη ή ημιδομημένη αναπαράσταση δεδομένων, πράγμα το

οποίο σημαίνει ότι η δομή καθορίζεται ανάλογα με τις ανάγκες που προκύπτουν, όπως για παράδειγμα η προσθήκη ενός νέου πεδίου μόνο στις εγγραφές που κρίνεται απαραίτητο.

Το μεγάλο πλεονέκτημα των συστημάτων NoSQL είναι η προσαρμοστικότητα που διαθέτουν και αυτός είναι και ένας βασικός λόγος που υιοθετήθηκαν από μεγάλες εταιρίες του χώρου του διαδικτύου στον χειρισμό δεδομένων μεγάλου όγκου, αφού τόσο αποθηκεύουν όσο και ανακτούν μεγάλες ποσότητες δεδομένων χωρίς να επηρεάζονται από τις σχέσεις που υφίστανται μεταξύ αυτών των στοιχείων.

Επιπλέον, οι βάσεις δεδομένων τύπου NoSQL έχουν σκοπό να καλύψουν τις ανάγκες του μεγάλου αριθμού χρηστών των εφαρμογών Internet, πετυχαίνοντας ταχύτατους χρόνους απόκρισης κάτι το οποίο κάνει την εμπειρία χρήσης των εφαρμογών αυτών περισσότερο πετυχημένη.

2.5.1 Τεχνολογίες NoSql

Οι βάσεις δεδομένων NoSQL κατηγοριοποιούνται ανάλογα με τον τρόπο αποθήκευσης δεδομένων. Στη συνέχεια θα περιγράψουμε διάφορους τύπους αποθήκευσης, οι οποίοι είναι: Column-oriented, Key-values Stores, Document-based stores και Graph DB.

2.5.1.1 Column-oriented (Στηλοκεντρικές Βάσεις Δεδομένων)

Αυτή η τεχνολογία αποθηκεύει τα δεδομένα σε στήλες, αντίθετα με τις σχεσιακές βάσεις δεδομένων οι οποίες αποθηκεύουν τα δεδομένα σε γραμμές. Στις στηλοκεντρικές βάσεις οι εγγραφές αποθηκεύονται, ανακτώνται και υπόκεινται σε επεξεργασία στηλοκεντρικά, όπως αναφέρει και το όνομα τους, ενώ υπάρχει δυνατότητα να περιέχουν τεράστιο αριθμό στηλών. Χαρακτηριστικό παράδειγμα είναι το Cassandra το οποίο δημιουργήθηκε από το Facebook.

Για παράδειγμα, σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων παρουσιάζει τα δεδομένα που αντιπροσωπεύει ένα δισδιάστατο πίνακα, στηλών και γραμμών, δηλαδή μια βάση δεδομένων μπορεί να έχει έναν πίνακα με την παρακάτω μορφή.

ROWLD	EMP_ID	LASTNAME	FIRSTNAME	SALARY
001	10	Smith	Joe	40.000
002	11	Jones	Mary	50.000
003	12	Johnson	Cathy	44.000
004	13	Jones	Bob	55.000

Πίνακας 1 – Μοντέλο δεδομένων σε μία σχεσιακή βάση δεδομένων

Σε μια σχεσιακή βάση τα δεδομένα θα αποθηκεύονταν εσωτερικά όπως φαίνεται στον Πίνακα 2.

001:10,Smith,Joe,40000;
002:11,Jones,Mary,50000;
003:12,Johnson,Cathy,44000;
004:13,Jones,Bob,55000;

Πίνακας 2 – Αναπαράσταση αποθήκευσης δεδομένων σχεσιακής βάσης δεδομένων

Για να βελτιωθεί η απόδοση στην επεξεργασία των δεδομένων πρέπει να αποθηκεύονται με τρόπο για να ελαχιστοποιηθεί ο αριθμός των αναζητήσεων στα αρχεία ή τους δίσκους όπου αποθηκεύονται τα δεδομένα της βάσης. Ως λύση στο πρόβλημα αποθήκευσης προτείνεται η στηλοκεντρική μέθοδος, όπως απεικονίζεται Πίνακα 3.

10:001,11:002,12:003,13:004;

Smith:001,Jones:002,Johnson:003,Jones:004;

Joe:001,Mary:002,Cathy:003,Bob:004;

40000:001,50000:002,44000:003,55000:004;

Πίνακας 3 - Αποθήκευση δεδομένων με την στηλοκεντρική μέθοδο

Τα συστήματα διαχείρισης βάσεων δεδομένων έχουν εξελιγμένους και βελτιστοποιημένους μηχανισμούς για να αποθηκεύουν τα δεδομένων τους, προκειμένου να λειτουργούν αποδοτικά εργασίες όπως είναι η τμηματοποίηση, η προσωρινή αποθήκευση και η δεικτοδότηση (partitioning, caching, indexing).

Στην περίπτωση που θα χρειαστεί η εισαγωγή μιας νέας οντότητας στο σύστημα τότε οι στηλοκεντρικές βάσεις δεδομένων είναι λιγότερο αποδοτικές από τις σχεσιακές διότι θα χρειαστεί ενημέρωση όλων των γραμμών της βάσης. Αντίθετα, στην περίπτωση προσθήκης ενός νέου πεδίου, αυτό θα γίνει πολύ εύκολα και αποδοτικά, με την προσθήκη μόνο μίας γραμμής στη βάση.

Επίσης, οι στηλοκεντρικές βάσεις έχουν μεγάλο πλεονέκτημα σε απόδοση στις περιπτώσεις επεξεργασίας υποσυνόλου των πεδίων της βάση, όπως για παράδειγμα σε πράξεις υπολογισμού μέγιστου, ελάχιστου, μέσου όρου και αθροίσματος ιδιαίτερα σε πολύ μεγάλα σύνολα δεδομένων.

2.5.1.2 Key-values Stores (Αποθήκες ζευγών κλειδιών – τιμών)

Αυτό το μοντέλο αποθήκευσης αποτελείται από έναν και μόνο πίνακα ο οποίος περιέχει 2 στήλες, όπου η πρώτη στήλη αφορά τα κλειδιά και η δεύτερη στήλη τις τιμές, ενώ κάθε κλειδί είναι μοναδικό. Οι τιμές αρχειοθετούνται και αργότερα θα προσπελάστουν με βάση τα κλειδιά. Χαρακτηριστική βάση που χρησιμοποιεί την αποθήκευση ζευγών κλειδιών – τιμών είναι η Memcache DB. Οι αποθηκών ζευγών κλειδιών – τιμών έχουν κύρια χρήση στην υλοποίηση συστημάτων προσωρινής μνήμης (caching).

Προκείμενου να κατανοηθεί το μοντέλο αποθήκευσης που χρησιμοποιούν οι αποθήκες ζευγών κλειδιών - τιμών, υπάρχει παρακάτω στον Πίνακα 4 μια αναπαράσταση βάσης όπως στο προηγούμενο παράδειγμα.

ΚΛΕΙΔΙ	ΤΙΜΗ
10_LASTNAME	Smith
10_FIRSTNAME	Joe
10_SALARY	40.000
11_LASTNAME	Jones
11_FIRSTNAME	Mary
11_SALARY	50.000
12_LASTNAME	Johnson
12_FIRSTNAME	Cathy
12_SALARY	40.000
13_LASTNAME	Jones
13_FIRSTNAME	Bob
13_SALARY	55.000

Πίνακας 4 – Αναπαράσταση της αποθήκευσης δεδομένων με την μέθοδο αποθήκης ζευγών κλειδιών - τιμών

2.5.1.3 Document-based stores (Αποθήκες εγγράφων)

Αυτή η κατηγορία βάσης δεδομένων η οποία αποκαλείται και εγγραφοκεντρική, δεν περιορίζεται από κάποιο σχήμα αλλά αποθηκεύει και οργανώνει τα δεδομένα σαν μια συλλογή από έγγραφα σε ημιδομημένους πίνακες, αφού οι χρήστες μπορούν προσθέσουν οι ίδιοι κάποιο πεδίο στον πίνακα χωρίς να υφίσταται κάποιος περιορισμός. Μια δημοφιλή λύση αποθήκης εγγράφων είναι το CouchDB το οποίο δημιουργήθηκε από τον οργανισμό ελεύθερου λογισμικού Apache.

Για παράδειγμα, δύο έγγραφα μπορούν να έχουν διαφορετικά σύνολα πεδίων. Ακόμη και αν τα έγγραφα που δεν ακολουθούν ένα αυστηρό σχήμα, μπορεί να δημιουργηθούν ευρετήρια και να χρησιμοποιούνται στις ερωτήσεις. Στον Πίνακα 5 παρουσιάζεται η πρώτη έγγραφη από το παράδειγμα της προηγούμενης βάσης δεδομένων.

```
{  "EMP_ID": "10",  
  "LASTNAME": "Smith",  
  "FIRSTNAME": "Joe",  
  "SALARY": "40.000",  
}
```

Πίνακας 5 – Αναπαράσταση εγγράφου

Ενώ, ένα δεύτερο έγγραφο μπορεί να μην έχει όλες τις πληροφορίες του προηγούμενου (μπορεί να μην έχει καθοριστεί ακόμα μισθός), όπως παρουσιάζεται στον Πίνακα 6.

```
{  "EMP_ID": "11",  
  "LASTNAME": "Jones",  
  "FIRSTNAME": "Mary",  
}
```

Πίνακας 6 – Αναπαράσταση εγγράφου με διαφορετικά δεδομένα

Ενώ ένα τρίτο έγγραφο μπορεί να έχει περισσότερα πεδία (όπως για παράδειγμα ο μισθός), ή να περιέχει και πίνακες, όπως απεικονίζεται και στον Πίνακα 7.

```
“EMP_ID”: “13”,
“LASTNAME”: “Jones”,
“FIRSTNAME”: “Bob”,
“ΜηνιαίεςΑπουσίες”:
[
{ “MONTH”: “Οκτώβριος”, “Απουσίες”: “1”} ,
{ “MONTH”: “Απρίλιος”, “Απουσίες”: “6”} ,
{ “MONTH”: “Μάιος”, “Απουσίες”: “18”} ,
]
```

Πίνακας 7 – Αναπαράσταση σύνθετου εγγράφου

Οι εγγραφοκεντρικές βάσεις παρέχουν αυτή την ευελιξία, δηλαδή να μετατρέπουν το σχήμα τους σε δυναμικό, δεδομένου της απεριόριστης ευελιξίας που αυτό προσφέρει, τις καθιστούν μια από τις δημοφιλείς επιλογές που χρησιμοποιούνται στις σημερινές διαδικτυακές εφαρμογές, όπου υπάρχει η ανάγκη αποθήκευσης διαφορετικών τύπων δεδομένων των οποίων τα χαρακτηριστικά αλλάζουν με τον καιρό ή ανάλογα με το συγκεκριμένο στοιχείο.

Αν και η εξέλιξη στην τεχνολογία των εγγραφοκεντρικών βάσεων έχει ακόμη δρόμο να διανύσει, υπάρχουν ήδη αρκετές ώριμες και δημοφιλείς λύσεις διαθέσιμες στην αγορά, οι χαρακτηριστικότερες των οποίων είναι οι: MongoDB, CouchDB και RavenDB.

2.5.1.4 Graph Databases

Οι βάσεις δεδομένων γραφών, με τις οποίες θα ασχοληθούμε διεξοδικά στο επόμενο κεφάλαιο, δεν χρησιμοποιούν πίνακες με γραμμές και στήλες ή μια άκαμπτη δομή. Βασίζονται στην θεωρία των γράφων και διαθέτουν ένα ευέλικτο μοντέλο γράφου το οποίο διαμοιράζεται σε πολλά μηχανήματα και προσαρμόζεται στις απαιτήσεις του χρήστη. Το ενδιαφέρον για αυτές τις βάσεις συνεχώς αυξάνεται αφού η κλασσική χρήση των βάσεων γίνεται για να απεικονίσουν δομές όπως τα κοινωνικά δίκτυα ή οι ιστοσελίδες του διαδικτύου, όπου οι σχέσεις μεταξύ των οντοτήτων είναι πολλές και περίπλοκες.

Αποτελούνται από *κόμβους* και *ακμές* που συνδέονται μεταξύ τους, ενώ χαρακτηρίζονται από ιδιότητες. Οι βάσεις δεδομένων γράφων έχουν σχεδιαστεί για να επιτρέπουν απλή και γρήγορη ανάκτηση πολύπλοκων ιεραρχικών δομών που είναι δύσκολο να μοντελοποιηθούν σε σχεσιακά συστήματα. Η καλύτερη χρήση για τις βάσεις δεδομένων γραφών είναι όταν οι σχέσεις επηρεάζουν τα ίδια τα δεδομένα.

Έχουν αναπτυχθεί αρκετές βάσεις δεδομένων γράφων οι σημαντικότερες εκ των οποίων είναι η Neo4j, η FlockDB και η InfiniteGraph.

2.6 NoSql ή Βάσεις Δεδομένων

Οι δύο τεχνολογίες, αυτή των σχεσιακών βάσεων δεδομένων η οποία υφίσταται σχεδόν τρεις δεκαετίες αλλά και αυτή των βάσεων δεδομένων NoSQL, είναι κατάλληλες για διαφορετικούς τύπους εργασιών και εφαρμογών η καθεμιά, για αυτό στις επόμενες παραγράφους θα παρουσιαστούν τα κριτήρια αυτών των δύο τεχνολογιών, όπως αυτά έχουν σχηματιστεί μέχρι τώρα.

Οι σχεσιακές βάσεις δεδομένων επικεντρώνονται στις ιδιότητες ACID, επομένως κάθε δοσοληψία εξετάζεται για αυτές τις ιδιότητες και αν αυτές παραβιάζονται τότε η δοσοληψία απορρίπτεται. Από την άλλη πλευρά, οι βάσεις δεδομένων NoSQL δεν ακολουθούν τις ιδιότητες ACID για την αξιοπιστία της βάσης αλλά βασίζονται στο θεώρημα CAP.

Αρχικά, θα αναφερθούμε στις Σχεσιακές Βάσεις Δεδομένων και τις εργασίες που πραγματοποιούν οι οποίες αυξάνουν τον φόρτο εργασίας και αυτό με την σειρά προκαλούν καθυστερήσεις, οι διεργασίες αυτές είναι:

- Καταγραφή (Logging): οι παραδοσιακές ΒΔ καταγραφούν σε ένα αρχείο στο σκληρό δίσκο το πλήρες ιστορικό των ενεργειών που έχουν εκτελεστεί προκειμένου να διασφαλιστεί η μονιμότητα της ενέργειας.
- Κλείδωμα (Locking): εξασφαλίζεται η αποκλειστική πρόσβαση σε κάθε εγγραφή του πίνακα πριν από εγγραφές και η διαμοιραζόμενη πρόσβαση πριν από τις αναγνώσεις.
- Διαχείριση ενδιάμεσων μνημών (Buffer management): ο χρόνος προσπέλασης των πληροφοριών αυξάνεται αφού τα δεδομένα αποθηκεύονται σε σελίδες δίσκων προκαθορισμένου μεγέθους.

Τα συστήματα βάσεων δεδομένων NoSQL μπορούν με ευκολία να προσαρμόσουν το μοντέλο στα μέτρα τους, αφού δεν έρχονται αντιμέτωπα με τους ανωτέρω

περιορισμούς των σχεσιακών ΒΔ. Πρέπει να σημειωθεί πως οι ΣΒΔ και τα χαρακτηριστικά τους ορισμένες φορές κυριαρχούν γεγονός που υποδεικνύει ότι αυτά τα χαρακτηριστικά έχουν τον κυρίαρχο λόγο για την επιλογή της κατάλληλης τεχνολογίας. Τα χαρακτηριστικά αυτά αναπτύσσονται παρακάτω:

- **Αξιοπιστία:** Οι βάσεις δεδομένων NoSQL δεν διαθέτουν τις ιδιότητες ACID αντίθετα με τις σχεσιακές βάσεις δεδομένων (ΣΒΔ), αυτό σημαίνει ότι εξασφαλίζουν αξιοπιστία μέχρι ενός σημείου αλλά όχι στο βαθμό που προσφέρουν οι ΣΒΔ. Για να εξασφαλιστούν τα χαρακτηριστικά των ιδιοτήτων ACID σε μια βάση δεδομένων NoSQL πρέπει να γίνει με την χρήση κώδικα.
- **Πολυπλοκότητα:** όπως ήδη έχουμε αναφέρει οι βάσεις δεδομένων NoSQL δεν χρησιμοποιούν τη γλώσσα SQL, επομένως για τις αναζητήσεις χρειάζεται είτε η συγγραφή κώδικα (πράγμα το οποίο είναι εύκολο στις απλές αναζητήσεις αλλά είναι χρονοβόρο και ίσως μη αποδοτικό σε περίπλοκες περιπτώσεις), είτε η εκμάθηση και χρήση μίας άλλης γλώσσας ερωτήσεων.
- **Άγνωστη Τεχνολογία** - ο φόβος για το καινούργιο, είναι γεγονός πως οι πιο πολλές εταιρίες νιώθουν ανασφάλεια σε σχέση με τη χρήση βάσεων δεδομένων NoSQL αφού δεν υπάρχει αρκετή εμπειρία στη χρήση τους. Έτσι, μερικές φορές επιλέγονται σχεσιακές βάσεις, παρόλο που στις συγκεκριμένες περιστάσεις η χρήση βάσεων δεδομένων NoSQL θα ήταν η καλύτερη επιλογή.

3. Βάσεις Δεδομένων γράφων (Graph Data Base)

Όπως ήδη έχει αναφερθεί τα συστήματα διαχείρισης βάσεων δεδομένων ιεραρχούν και αποθηκεύουν τα δεδομένα σε συλλογές δεδομένων και μέσω των πινάκων σύνδεσης και με τη χρήση κλειδιών πραγματοποιείται η σύνδεση των δεδομένων. Αποτέλεσμα αυτής της διαδικασίας είναι να μειώνεται η αποδοτικότητα του συστήματος στην απάντηση περίπλοκων ερωτημάτων, ενώ –όπως αναφέρθηκε- η περιγραφή πολύπλοκων σχέσεων του πραγματικού κόσμου στις σχεσιακές βάσεις δεδομένων είναι δυσχερής.

Σήμερα, παρατηρείται έντονο ενδιαφέρον για τις βάσεις δεδομένων γράφων οι οποίες έχουν την δυνατότητα να αποθηκεύουν τα δεδομένα με μορφή γράφου. Έτσι, υπάρχει η δυνατότητα να απεικονίζονται με φυσικό τρόπο στη βάση δεδομένων όλες οι συνδέσεις που υφίστανται στον πραγματικό κόσμο.

3.1 Δομή Βάσης Δεδομένων Γράφων

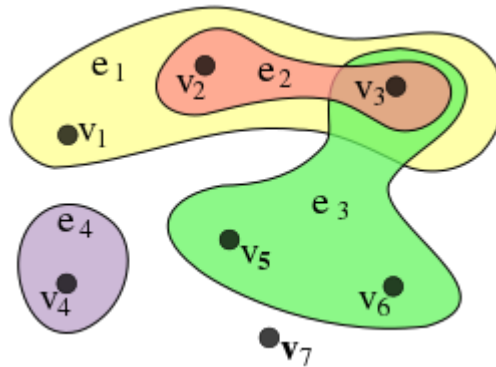
Τα συστήματα βάσεων δεδομένων γράφων χρησιμοποιούν μορφές γράφων για την αναπαράσταση των δεδομένων, όπου περιλαμβάνουν κόμβους, ακμές και ιδιότητες. Οι κόμβοι (nodes) χρησιμοποιούνται για να αναπαριστούν οντότητες, όπως είναι για παράδειγμα τα άτομα ή οι επιχειρήσεις. Οι ακμές (edges) είναι οι συνδέσεις μεταξύ των κόμβων, δηλαδή αναπαριστούν τις σχέσεις ανάμεσα στους κόμβους. Οι κόμβοι και οι ακμές διαθέτουν ιδιότητες οι οποίες προσδιορίζουν τα χαρακτηριστικά τους.

Έχουν αναπτυχθεί αρκετές βάσεις δεδομένων γράφων όπως είναι: Neo4j, GraphDB, FlockdB, InfiniteGraph όπου σε κάθε εφαρμογή τα δεδομένα αποθηκεύονται και υποβάλλονται ερωτήσεις στον γράφο.

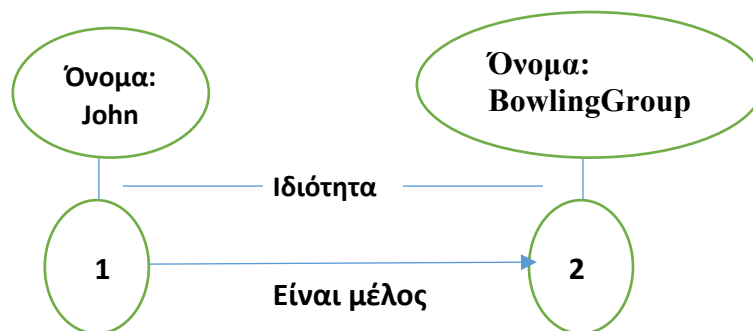
Ουσιαστικά, οι βάσεις δεδομένων γράφων είναι μια αναπτυσσόμενη τεχνολογία βάσεων δεδομένων και θα μπορούσε να οριστεί ως ένα μοντέλο βάσεων δεδομένων το οποίο αναπαριστά την δομή των δεδομένων είτε ως κατευθυνόμενους γράφους είτε με γενικές δομές γράφων, όπου ο χειρισμός των δεδομένων πραγματοποιείται μέσω διάφορων τεχνικών που εφαρμόζονται στους γράφους και η ακεραιότητα των δεδομένων περιορίζεται στην δομή των γράφων. στους γράφους υφίστανται οι δύο ακόλουθες έννοιες

- Hypernodes (υπερ-κόμβοι): αναφέρεται στην ενθυλάκωση των γράφων, δηλαδή ένας κόμβος μπορεί να αναπαριστά έναν άλλο ολόκληρο γράφο.

- Hyperedges (υπερ-ακμές): μια υπερ-ακμή δεν συνδέει μόνο 2 κόμβους αλλά εκφράζει τη σχέση μεταξύ ενός κόμβου από τη μία πλευρά και ενός ακαθόριστου αριθμού κόμβων από την άλλη πλευρά.



Εικόνα 4 – Απεικόνιση hypernodes & hyperedges. Η υπερ-ακμή e_1 εκφράζει μία σχέση ανάμεσα στις κορυφές v_1, v_2, v_3 , ενώ η ακμή e_3 εκφράζει μία σχέση ανάμεσα στις κορυφές v_3, v_5 και v_6 κ.ο.κ.



Εικόνα 5 – Αναπαράσταση ενός στιγμιότυπου βάσης δεδομένων γράφων

Η εικόνα 4 αποτελείται από 2 κόμβους (1,2), έχει μια ακμή με ετικέτα “Είναι μέλος”, ενώ ο κάθε κόμβος έχει μια ιδιότητα “όνομα”. Στην προκειμένη περίπτωση η ακμή του γράφου διαθέτει ένα βέλος γεγονός που δείχνει ότι ο γράφος είναι κατευθυνόμενος.

Ωστόσο, η ακμή ενός γράφου μπορεί να είναι είτε κατευθυνόμενη (directed) είτε μη κατευθυνόμενη (undirected), ωστόσο πιο συχνά στις βάσεις δεδομένων χρησιμοποιούνται οι γράφοι που διαθέτουν κατευθυνόμενες ακμές.

3.2 Πλεονεκτήματα Βάσεων Δεδομένων Γράφων

Σύμφωνα με τον ιδρυτή της Neo4j, Emil Eifrem, οι λόγοι για τους οποίους οι βάσεις δεδομένων γράφων είναι οι προτιμότερες είναι οι ακόλουθοι.

Σήμερα, οι περισσότερες εφαρμογές χειρίζονται δεδομένα τα οποία είναι δομημένα ως γραφήματα, δηλαδή διαθέτουν βαθιές και πολύπλοκες συσχετίσεις. Χαρακτηριστικά παραδείγματα αποτελούν οι ιστότοποι των μέσων κοινωνικής δικτύωσης αλλά και τα συστήματα διαχείρισης περιεχόμενου τα οποία ασχολούνται είτε με εγγενώς ιεραρχικά δεδομένα είτε με δεδομένα σε μορφή γραφήματος.

Ουσιαστικά, το πρόβλημα εστιάζεται στο γεγονός πως δεν μπορούν να με φυσικό και αποτελεσματικό τρόπο να αντιμετωπιστούν οι αναδρομικές σχέσεις με τις παραδοσιακές σχεσιακές βάσεις δεδομένων. Με δεδομένο ότι το περιεχόμενο των δεδομένων το οποίο καθοδηγείται από τον χρήστη, είναι δύσκολο να προκαθορισθεί το ακριβές σχήμα των δεδομένων που θα χειριστεί μία εφαρμογή, και στο σχεσιακό μοντέλο η ύπαρξη προκαθορισμένου μοντέλου είναι απαραίτητη.

Μια βάση δεδομένων γράφων χρησιμοποιεί κόμβους, ακμές μεταξύ των κόμβων αλλά και τις ιδιότητες για να αντιπροσωπεύουν τις πληροφορίες, αντί για πίνακες. Το μοντέλο των βάσεων δεδομένων γράφων είναι ταχύτερο σε μεγάλα σύνολα δεδομένων, αφού το μοντέλο που χρησιμοποιεί δεν βασίζεται σε προκαθορισμένα σχήματα και είναι επίσης ιδανικό για εφαρμογές λήψης αποφάσεων.

3.3 Σύγκριση Βάσεων Δεδομένων Γράφων και Σχεσιακών Βάσεων Δεδομένων

Οι διαφορές μεταξύ των μοντέλων βάσεων δεδομένων γράφων και των μοντέλων σχεσιακών βάσεων δεδομένων είναι αρκετές. Οι πιο σημαντικές από αυτές τις διαφορές αναφέρονται παρακάτω.

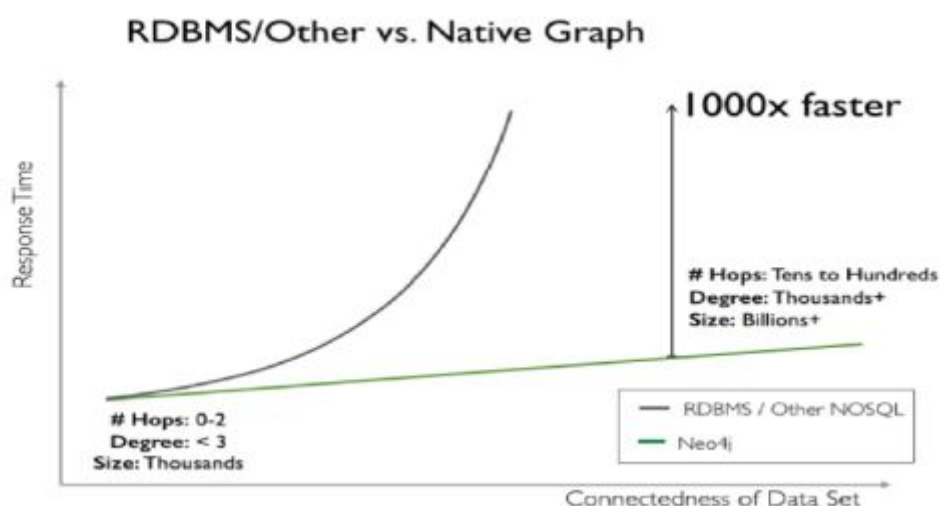
Στο σχεσιακό μοντέλο τα δεδομένα καταχωρούνται σε μια δομή η οποία είναι γνωστή εκ των προτέρων. Αντίθετα, στις βάσεις δεδομένων γράφων η δομή της βάσης δεδομένων γράφων μπορεί να τροποποιηθεί ανάλογα με τις απαιτήσεις. Όσον αφορά την επεκτασιμότητα οι σχεσιακές βάσεις δεδομένων επεκτείνονται δύσκολα αφού το σχήμα είναι σταθερό, ενώ στις βάσεις δεδομένων γράφων έχουν απόλυτη ευελιξία στο περιεχόμενο που θα αποθηκεύσουν.

Οι βάσεις δεδομένων γράφων διαθέτουν μεθόδους προκειμένου να δημιουργούν συσχετίσεις μεταξύ των δεδομένων, ενώ στις σχεσιακές βάσεις δεδομένων η γλώσσα

ερωτημάτων δεν είναι σε θέση να εντοπίζει συσχετίσεις που μπορεί τα δεδομένα να έχουν μεταξύ τους.

Ακόμα, οι βάσεις δεδομένων γράφων προσφέρουν υψηλή απόδοση στη διαχείριση των συνδεδεμένων δεδομένων ωστόσο στις σχεσιακές βάσεις δεδομένων οι συνδέσεις μεταξύ των πινάκων δημιουργούν μείωση στην απόδοση του συστήματος όσο ο όγκος των δεδομένων αυξάνεται, πράγμα το οποίο δεν ισχύει στις βάσεις δεδομένων γράφων η απόδοση παραμένει σταθερή.

Μια βάση δεδομένων γράφων είναι η neo4j, η οποία θα αναλυθεί παρακάτω, όπως παρουσιάζεται στην Εικόνα 6 απεικονίζει την απόδοση της neo4j που όσο ο όγκος των δεδομένων αυξάνεται παραμένει σχεδόν σταθερή, ενώ, στα σχεσιακά συστήματα όσο ο όγκος των δεδομένων αυξάνεται τόσο μειώνεται η απόδοση (αυξάνεται ο χρόνος απόκρισης).



Εικόνα 6 -Σύγκριση σχεσιακών βάσεων δεδομένων και βάσεων δεδομένων γράφων στην αύξηση των δεδομένων

3.4 Συστήματα Βάσεων Δεδομένων Γράφων

Σήμερα υπάρχουν αρκετά συστήματα βάσεων δεδομένων γράφων οι οποίες έχουν δημιουργηθεί μέχρι σήμερα. Θα παρουσιαστούν μερικά λογισμικά με διαφορετικά χαρακτηριστικά, τα οποία είναι: Neo4j, Graph DB, Infinite Graph και η Cayley.

3.3.1 Neo4j

Η βάση δεδομένων γράφων Neo4j είναι γραμμένη σε γλώσσα Java, είναι ανοιχτού κώδικα, δημιουργήθηκε από την εταιρία Neo Technology και κυκλοφόρησε το 2010.

Η Neo4j χρησιμοποιεί ένα σύνολο από κόμβους και κατευθυνόμενες ακμές, ενώ κάθε κόμβος και κάθε ακμή διαθέτει ένα σύνολο ιδιοτήτων προκειμένου να αποθηκεύονται τα δεδομένα. Υποστηρίζει μόνο κατευθυνόμενους κόμβους και για αν υφίσταται αμφίδρομη σχέση μεταξύ των κόμβων τότε πρέπει να υπάρχουν 2 συσχετίσεις αναμεσά τους.

Προκειμένου να θέτουμε αιτήματα, δηλαδή να θέτονται ερωτήσεις και να δίνονται απαντήσεις χρησιμοποιείται η γλώσσα Cypher, η οποία είναι αρκετά εύκολη για τους χρήστες οι οποίοι γνωρίζουν SQL (η οποία χρησιμοποιείται στις σχεσιακές βάσεις δεδομένων). Η Neo4j είναι ικανή να διαχειρίζεται γράφους αρκετών δισεκατομμυρίων κόμβων, σχέσεων και ιδιοτήτων, αφού διασχίζει 1.000.000 συσχετίσεις/δευτερόλεπτο.

Η Neo4j αποτελεί μια βάση δεδομένων γράφων η οποία διατηρεί τις ιδιότητες ACID, έτσι είναι ικανή να διατηρεί πλήρως την ακεραιότητα των δεδομένων, αλλά και την απομόνωση των δοσοληψιών, δηλαδή μέχρι να ολοκληρωθεί μια δοσοληψία αποκρύπτει τις αλλαγές από άλλες διεργασίες.

3.3.2 AllegroGraph

Η AllegroGraph είναι ένα λογισμικό κλειστού κώδικα για διαχείριση βάσεων δεδομένων γράφων που, το οποίο είναι σχεδιασμένο να αποθηκεύει τριπλέτες RDF. Το RDF (Resource Description Framework) είναι ένα πρότυπο για τη μορφοποίηση δομημένων δεδομένων ή μετα-δεδομένων. Χρησιμοποιείται σε διάφορα έργα ανοιχτού λογισμικού, καθώς και σε εμπορικά έργα και έργα κυβερνητικών ή μη οργανισμών. Είναι επίσης υπεύθυνη για την αποθήκευση δεδομένων για το έργο TwitLogic το οποίο εφαρμόζει την λογική του Σημασιολογικού Ιστού στα δεδομένα του Twitter.

Είναι ένα προϊόν της εταιρίας Franz Inc. η οποία αποτελεί κορυφαίο προμηθευτή για εμπορικές βάσεις δεδομένων RDF που διαθέτουν αυξημένες δυνατότητες κλιμάκωσης. Η πρώτη έκδοση της AllegroGraph ήταν διαθέσιμη το 2004.

Η AllegroGraph χρησιμοποιεί την SPARQL, η οποία είναι μια πρότυπη γλώσσα ερωτήσεων για συνδεδεμένα δεδομένα, εξυπηρετώντας τους ίδιους σκοπούς για βάσεις δεδομένων RDF. Είναι πλέον διαθέσιμη για όλες τις πλατφόρμες υλικού και λογισμικού.

Η AllegroGraph παρέχει μια ευρεία σειρά μηχανισμών αναζήτησης και πρόσβασης σε μια βάση δεδομένων RDF:

- **Μηχανή RDFS++:** Η AllegroGraph προσφέρει έναν πολύ γρήγορο και πρακτικό μηχανισμό συμπερασμού RDFS. Ο μηχανισμός RDFS ++ της AllegroGraph διατηρεί με δυναμικό τρόπο το οντολογικό υπόβαθρο που απαιτείται για την εκτέλεση συμπερασμών και δεν περιλαμβάνει διακριτή φάση προ-υπολογισμού και αποθήκευσης συναχθισών τριπλετών (materialization) ώστε να . εκτελούνται πιο αποτελεσματικά τα ερωτήματα. Το ουσιαστικό πρόβλημα με τη διαδικασία του materialization είναι η συντήρησή του, αφού οποιαδήποτε αλλαγή στα πρωτογενή δεδομένα απαιτεί πλήρη επανεπεξεργασία πριν από την εμφάνιση νέων ερωτημάτων. Η δυναμική εκτέλεση του materialization του AllegroGraph απλοποιεί τη συντήρηση της βάσης και μειώνει τον απαιτούμενο χρόνο μεταξύ των αλλαγών στα δεδομένα και της εκτέλεσης των ερωτήσεων.
- **Ερωτήματα SPARQL:** Η AllegroGraph υλοποιεί τη SPARQL, την πρότυπη γλώσσα ερωτημάτων σε βάσεις δεδομένων RDF του W3C, η οποία μπορεί να επιστρέφει απαντήσεις σε RDF ή XML. Η υλοποίηση SPARQL της AllegroGraph ακολουθεί τα πρότυπα διαλειτουργικότητας του W3C, περιλαμβάνει ένα εργαλείο βελτιστοποίησης επερωτήσεων και παρέχει πλήρη υποστήριξη για χειρισμό γράφων.
- **Prolog:** Η RDF Prolog της AllegroGraph παρέχει ισχυρές δυνατότητες για διεξαγωγή συμπερασμών με βάση πολύπλοκους κανόνες ή εκτεταμένη επεξεργασία, πάνω σε δεδομένα RDF. Η εκτέλεση αυτών των συμπερασμών είναι εξαιρετικά δυσχερές να πραγματοποιηθεί μόνο με χρήση RDF/RDFS και OWL, συνεπώς η Prolog παρέχει τα απαραίτητα εργαλεία ώστε να κτιστεί ένα σύστημα κανόνων συμπερασμού πάνω από τον βασικό μηχανισμό συμπερασμού RDFS++.

- **Σύντομη, ισχυρή, βασισμένη βιομηχανικά standards**, εξειδικευμένη για την δημιουργία εννοιών υψηλού επιπέδου (που απαιτούν σύνθετους κανόνες ή επεξεργασία) πάνω σε δεδομένα RDF. Έτσι είναι δύσκολο (ή πολύ περίεργο) να μοντελοποιήσουμε δεδομένα μόνο με RDF / RDFS και OWL. Η Prolog μπορεί επίσης να χρησιμοποιηθεί μαζί με το RDFS++ ως ένα σύστημα βασισμένο σε κανόνες.
- **API χαμηλού επιπέδου**. Παρέχει εξαιρετικά ταχεία πρόσβαση στις τριπλέτες, με βάση τα εσωτερικά χαρακτηριστικά υλοποίησης της AllegroGraph.

Η AllegroGraph υλοποιεί και παρέχει διεπαφές τύπου REST, ενώ έχει επίσης σχεδιαστεί έτσι ώστε οι πόροι του υλικού να αξιοποιούνται με βέλτιστο τρόπο σε όλες τις διαδικασίες διαχείρισης δεδομένων. Η αξιοποίηση του υλικού πραγματοποιείται μέσω του αρχείου διαμόρφωσης της AllegroGraph. Υποστηρίζει απευθείας διασυνδέσεις για διάφορες γλώσσες, Sesame Java, Sesame Jena, Python χρησιμοποιώντας τις Sesame και Lisp. Παράλληλα, η κοινότητα των χρηστών έχει δημιουργήσει διασυνδέσεις ανοικτού κώδικα για C #, Ruby, Clojure, Scala και Perl. Τέλος, παρότι η AllegroGraph ανήκει στις βάσεις δεδομένων γράφων, εφαρμόζει τις ιδιότητες ACID.

3.3.3 Oracle Spatial and Graph

Καθώς οι εφαρμογές εξελίσσονται, αφού εμφανίζονται νέες τεχνολογίες και πλατφόρμες, βρίσκονται συνεχώς νέοι τρόποι για να ενσωματωθούν και να αξιοποιηθούν οι πληροφορίες από τα κοινωνικά δίκτυα. Η εμφάνιση των υπηρεσιών Cloud, της παρακολούθησης μέσω κινητού τηλεφώνου, των κοινωνικών μέσων και των συστημάτων πραγματικού χρόνου δημιουργούν νέες προκλήσεις για τη διαχείριση του όγκου των δεδομένων, αλλά κυρίως για την ανακάλυψη συνδέσεων και σχέσεων.

Για να αντιμετωπιστούν όλες οι παραπάνω προκλήσεις, δημιουργήθηκε το Oracle Database 12c η οποία περιλαμβάνει ένα ευρύ φάσμα λειτουργιών και υπηρεσιών χωρικής ανάλυσης για την αποτίμηση συνθηκών στα δεδομένα με βάση το πόσο κοντά μία οντότητα είναι σε μία άλλη, εάν μία οντότητα βρίσκεται εντός μίας περιοχής, ή για την οπτικοποίηση μοτίβων σε χάρτες και εικόνες. Με την Oracle Database 12c που βασίζεται στο Oracle Cloud, η Oracle εισάγει μία νέα δυνατότητα ανάλυσης γράφων με βάση τις ιδιότητες για να αντιμετωπίσει τυπικές αλλά και καινοτόμες περιπτώσεις

χρήσης γράφων, συμπεριλαμβάνοντας τη διαμόρφωση συστάσεων, τον εντοπισμό κοινοτήτων και οντοτήτων υψηλής επιρροής (influencers), ταίριασμα μοτίβων και εντοπισμό απάτης. Στη βάση δεδομένων αποθηκεύονται με φυσικό τρόπο τα δεδομένα τα οποία δημιουργούνται από εγγενείς σχέσεις μεταξύ των οντοτήτων του πραγματικού κόσμου, όπου οι κόμβοι του γράφου υποδεικνύουν τις οντότητες, τα άκρα υποδηλώνουν τις σχέσεις και οι ιδιότητες αποθηκεύονται ως ζεύγος κλειδιών – τιμών.

Η ευκολία στην ανάπτυξη εφαρμογών και η διαχείριση των δεδομένων επιτυγχάνεται μέσω μια κονσόλας βασισμένης στο Groovy και ένα σύνολο Java API για τη δημιουργία και τη διαχείριση των γράφων, την αφαίρεση κόμβων και ακμών και την αναζήτηση κόμβων και ακμών χρησιμοποιώντας ζεύγη κλειδιών-τιμών. Τα API της Java περιλαμβάνουν την υλοποίηση των διεπαφών Apache TinkerPop.

Η γρήγορη και παράλληλη φόρτωση μεγάλων γράφων επιτυγχάνεται με τη χρήση μιας ειδικής μορφής αρχείου, σχεδιασμένης από την Oracle και ενός βοηθητικού προγράμματος για την εύκολη μετατροπή των πινάκων Oracle και των αρχείων CSV (CSV) σε αυτή τη μορφή. Επίσης, υποστηρίζονται τα ανοικτά πρότυπα γράφων GraphSON, GraphML και GML.

Το χωρικό φιλτράρισμα στα ερωτήματα γράφων μπορεί να βελτιώσει την αποτελεσματικότητα της ανάλυσης γράφων. Μια χωρική γεωμετρία, όπως οι συντεταγμένες μιας διεύθυνσης, μπορεί να αποθηκευτεί ως ιδιότητα και να αναλυθεί. Για παράδειγμα, ένα ερώτημα "εντός απόστασης" μπορεί να καθορίσει αν θα εξεταστεί μία συνδεδεμένη οντότητα στα υπόλοιπα βήματα της ανάλυσης. Η υποστήριξη γεωμετρικών σημείων, γραμμών και πολυγώνων και η χωρική ευρετηρίαση με βάση τις λειτουργίες και η πρόσβαση σε χωρικές λειτουργίες ανάλυσης αυξάνουν σημαντικά την ισχύ του συγκεκριμένου χαρακτηριστικού.

3.3.4 OpenLink Virtuoso

Το σύστημα βάσεων δεδομένων Virtuoso, γνωστό ως και Virtuoso Universal Server, είναι ένα ενδιάμεσο λογισμικό και μηχανισμός βάσης δεδομένων που συνδυάζει τη λειτουργικότητα ενός παραδοσιακού συστήματος διαχείρισης βάσεων δεδομένων (RDBMS), βάσης δεδομένων σχεσιακών αντικειμένων (ORDBMS), εικονικής βάσης

δεδομένων με περιεχόμενα RDF, XML, ελεύθερου κειμένου, διακομιστή web server και λειτουργικότητα διακομιστή αρχείων σε ένα ενιαίο σύστημα.

Αντί να διατηρούμε ξεχωριστούς διακομιστές για κάθε ένα από τα προαναφερθέντα πεδία λειτουργικότητας, το Virtuoso είναι ένας "καθολικός διακομιστής". Ενοποιεί πολλαπλούς διακομιστές υλοποιώντας πολλαπλά πρωτόκολλα. Ο διακομιστής Virtuoso είναι σχεδιασμένος με υποστήριξη για πολλαπλά νήματα ελέγχου και υποστήριξη για εκτέλεση σε συστήματα με πολλαπλούς επεξεργαστές.

Για την χρήση ερωτημάτων υποστηρίζει τη SPARQL και είναι συμβατή με τη Jena, όπου η SPARQL μεταφράζεται σε SQL κατά την ανάλυση του ερωτήματος. Όλες οι τριπλέτες αποθηκεύονται σε έναν μοναδικό πίνακα.

Η Virtuoso υποστηρίζεται από μεγάλο αριθμό πλατφορμών, τόσο 32 bit όσο και 64 bit (Windows, UNIX, Linux (Red Hat, SUSE, Mac OS X)). Έχει τη δυνατότητα να διαχειρίζεται αποτελεσματικά πάνω από 1 δισεκατομμύριο τριπλέτες.

3.3.5 GraphDB

Η GraphDB, που αρχικά ονομαζόταν Sones GraphDB, είναι μια βάση δεδομένων γράφων η οποία αναπτύχθηκε από τη εταιρία Sones και ήταν διαθέσιμη από το 2010 μέχρι το 2012. Το 2014 εξαγοράστηκε από την εταιρία Ontotext και μετονομάστηκε, λαμβάνοντας το σημερινό της όνομα. Έχει αναπτυχθεί σε C# και τρέχει σε πλατφόρμα .Net.

Το λογισμικό αυτό συνδέεται με μια βάση δεδομένων και παρέχει τη δυνατότητα υποβολής και επεξεργασίας ερωτημάτων σχετικά με αυτή τη βάση δεδομένων. Στη συνέχεια οι απαντήσεις από τα ερωτήματα παρουσιάζονται φυσικά και διαισθητικά. Η πρόσβαση στα δεδομένα πραγματοποιείται μέσω της γλώσσας GQL (Graph Query Language) η οποία χρησιμοποιεί μια επέκταση της SQL που έχει προσαρμοστεί για να αλληλεπιδρά με βάσεις δεδομένων γράφων.

Όσον αφορά το σχήμα της βάσης δεδομένων γράφων GraphDB, αυτό αποτελεί επέκταση ενός γράφου. Πολλαπλές ακμές μπορούν να ομαδοποιούνται σε μία υπερ-ακμή (hyperedge), κάθε μία από τις οποίες η οποία μπορεί να αποκτήσει τις δικές της

ιδιότητες: για παράδειγμα οι ιδιότητες μπορεί να επιτρέπουν την χρήση υπολογισμών μεταξύ του συνόλου των κόμβων.

3.3.6 InfiniteGraph

Η InfiniteGraph αποτελεί το σύστημα βάσης δεδομένων γράφων της εταιρείας Objectivity, η οποία αναπτύσσει βάσης δεδομένων με υψηλή δυνατότητα κλιμάκωσης και πραγματοποιεί καταναμημένη διαχείριση δεδομένων. Είναι γραμμένη σε Java, είναι διαθέσιμη στο κοινό από το 2010, ανήκει στην κατηγορία των βάσεων δεδομένων NoSQL και επικεντρώνεται στις βάσεις δεδομένων γράφων.

Οι προγραμματιστές χρησιμοποιούν την InfiniteGraph για μεγάλα σύνολα συνδεδεμένων δεδομένων. Η GraphDB είναι συμβατή με όλα τα λειτουργικά συστήματα, είναι επεκτάσιμη και διαχειρίζεται τα δεδομένα με υψηλή απόδοση.

Η InfiniteGraph είναι κατάλληλη για εφαρμογές και οι υπηρεσίες οι οποίες λύνουν προβλήματα γράφων ή είναι σε θέση να απαντούν σύνθετα ερώτημα όπως είναι για παράδειγμα “Ποια αεροπορική εταιρία διαθέτει τα φθηνότερα αεροπορικά εισιτήρια με επιστροφή για έναν συγκεκριμένο προορισμό, με 2 στάσεις το πολύ και να αναχωρεί συγκεκριμένη μέρα και ώρα”.

Οι εφαρμογές συνδέονται σε ένα καταναμημένο σύστημα στο οποίο η InfiniteGraph χρησιμοποιεί ένα δικό της διακομιστή αρχείων με όνομα AMS, προκειμένου να ενορχωστρωθεί η πρόσβαση στα δεδομένα.

Το περιβάλλον της InfiniteGraph είναι απλό και σκοπό του είναι να αναδείξει την δομή του γράφου, προσφέροντας εύκολη διαχείριση. Επίσης, διαθέτει μία ευέλικτη εφαρμογή πλοήγησης η οποία προσαρμόζεται ανάλογα με τις ανάγκες του μοντέλου. Οι κόμβοι εμφανίζονται μαζί με τις συνδέσεις τους και υπάρχει λεπτομερής εμφάνιση των δεδομένων όταν επιλέγει ένα σημείο του γράφου.

Αυτή η βάση δεδομένων γράφων φαίνεται να έχει αποδοχή από οργανισμούς οι οποίοι ασχολούνται με την υγειονομική περίθαλψη, σε εφαρμογές κοινωνικής δικτύωσης ή εταιρίες που παρέχουν υπηρεσίες ασφάλειας στο διαδίκτυο.

3.3.7 Cayley

Η βάση δεδομένων γράφων Cayley χρησιμοποιείται για εφαρμογές οι οποίες βασίζονται σε γραφήματα σχετιζομένων πληροφοριών . Η Cayley διατίθεται δωρεάν και είναι εύκολη στην χρήση της.

Η αφετηρία για την Cayley υπήρξε ο Γράφος γνώσης της Google ο οποίος χρησιμοποιείται σε μια αναζήτηση και είναι μια βάση δεδομένων “γνώσης”. Ουσιαστικά, σκοπός του είναι να βελτιώσει τα αποτελέσματα της αναζήτησης μέσα από μια γραφική προβολή των πληροφοριών, οι οποίες συλλέγονται από πολλές πηγές. Οι πληροφορίες εμφανίζονται δομημένες σχετικά με το θέμα της αναζήτησης, ενώ παρουσιάζονται και συνδέσεις με άλλους ιστότοπους οι οποίοι ενδέχεται να αφορούν το θέμα αναζήτησης.

Έτσι λοιπόν γνωρίζοντας τη σημασία του γράφου γνώσης, οι προγραμματιστές της Google θέλησαν να δημιουργήσουν μια ανοιχτού κώδικα βάση δεδομένων γράφων για προγραμματιστές εκτός Google, και το αποτέλεσμα είναι η Cayley, η οποία πήρε το όνομα της από τον μαθηματικό Arthur Cayley ο οποίος είναι υπεύθυνος για το θεώρημα γράφων Cayley.

Η Caley είναι πνευματικός διάδοχος του συστήματος GraphD με το οποίο μοιράζεται την ίδια στρατηγική στην αποτίμηση των ερωτήσεων για την επίτευξη ταχύτητας. Είναι γραμμένη με την γλώσσα Go, η οποία είναι κατάλληλη για υλοποίηση backend υπηρεσιών με υψηλή ταχύτητα και δυνατότητα ταυτόχρονης πρόσβασης στα δεδομένα. Η Cayley αξιοποιεί τις εκτενείς βιβλιοθήκες της γλώσσας Go, οι οποίες διευκολύνουν σε σημαντικό βαθμό την ταυτόχρονη πρόσβαση στα δεδομένα.

4. CAYLEY

Σε αυτό το κεφαλαίο, περιγράφεται η βάση δεδομένων γράφων Cayley, η διαδικασία εγκατάστασης της, η γλώσσα ερωτημάτων που χρησιμοποιεί η οποία είναι Gremlin.

4.1 Περιγραφή της Cayley

Η Cayley ανήκει στις βάσεις δεδομένων γράφων και είναι πνευματική διάδοχος του προγράμματος GraphD, με το οποίο διαμοιράζεται την ίδια στρατηγική στα queries προκειμένου να επιτευχθεί υψηλή ταχύτητα. Ουσιαστικά, δεν διαφέρει από τον προκάτοχο της αφού παρέχει τα έξης χαρακτηριστικά:

- Υποστήριξη Restful API.
- Πολλαπλοί τρόποι αποθήκευσης, όπως είναι η MongoDB.
- Υποστήριξη για πολλές γλώσσες ερωτημάτων
- Ευκολία στη χρήση για αρχάριους
- Δυνατότητα χρήσης σαν βιβλιοθήκη για τη κατασκευή περίπλοκων εφαρμογών.
- Είναι λογισμικού ανοιχτού κώδικα.

Η Cayley χρησιμοποιεί κόμβους, ακμές και ιδιότητες και έχει σχεδιαστεί έτσι ώστε να επιτρέπει την απλή και γρήγορη ανάκτηση σύνθετων ιεραρχικών δομών τα οποία δύσκολα μοντελοποιούνται στα σχεσιακά συστήματα.

Τα 3 κύρια μέρη της βάσης δεδομένων γράφων Cayley είναι οι κόμβοι, οι άκρες και οι ιδιότητες, όπου:

- Οι κόμβοι αντιπροσωπεύουν μια οντότητα, όπως είναι για παράδειγμα ένα άτομο.
- Μια ακμή καθορίζει την σχέση μεταξύ των 2 κόμβων. Μια ακμή θα έχει πάντα μια κατεύθυνση, θα εξέρχεται από έναν κόμβο και θα εισέρχεται σε έναν άλλο.
- Η ιδιότητα περιγράφεται με ένα ζεύγος κλειδιού – τιμής η οποία προσδίδεται σε έναν κόμβο ή μια ακμή.

4.2 Εγκατάσταση της Cayley

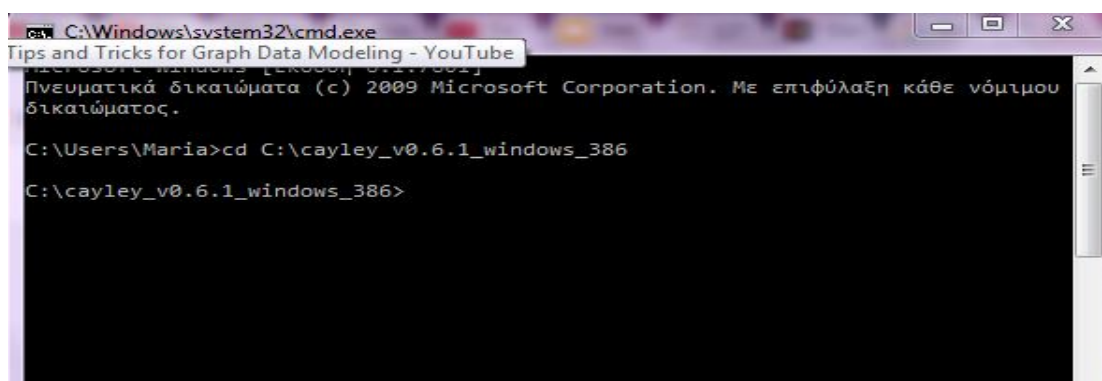
Στη συνέχεια περιγράφεται η διαδικασία εγκατάστασης της βάσης δεδομένων γράφων Cayley για το λειτουργικό σύστημα Windows.

Αρχικά, από τον ιστότοπο της Cayley <https://cayley.io/> και επιλέγοντας την εντολή Download Cayley, εμφανίζεται μια λίστα με διάφορες εκδόσεις της, όπου επιλέγεται η κατάλληλη σύμφωνα με το λειτουργικό σύστημα του υπολογιστή μας.

Στη συνέχεια, μόλις ολοκληρωθεί το κατέβασμα πρέπει να αποσυμπεστεί το αρχείο σε έναν φάκελο.

Έπειτα, μετά την αποσυμπίεση του αρχείου, κάθε φορά που θα πραγματοποιείται χρήση της Cayley, η διαδικασία έχει ως εξής:

1. Με την χρήση drag and drop, σύρουμε τα αρχεία του φακέλου Data, στην εφαρμογή cayley.exe
2. Πραγματοποιείται άνοιγμα της γραμμής εντολών (command prompt), γράφοντας την εντολή cmd στην γραμμή αναζήτησης.
3. Μετά το άνοιγμα του παραθύρου εντολών πληκτρολογείτε η εντολή cd (change directoy) και σύρουμε όλο τον αποσυμπεσμένο φάκελο cayley_v0.6.1_windows_386, το αποτέλεσμα απεικονίζεται στην Εικόνα 7.



Εικόνα 7 - Αλλαγή διεύθυνση φακέλου προορισμού

4. Έπειτα συνεχίζουμε στην γραμμή εντολών γράφοντας την εντολή

```
cayley.exe http --dbpath="30kmoviedata.nq.gz"
```

όπως φαίνεται και στην Εικόνα 8.

```
C:\cayley_v0.6.1_windows_386>cayley.exe http --dbpath="30kmoviedata.nq.gz"
Serving Gephi GraphStream at http://localhost:64210/gephi/gs
Cayley now listening on 127.0.0.1:64210
```

Εικόνα 8 - Διεύθυνση Cayley

5. Μετά την παραπάνω διαδικασία εμφανίζεται μια διεύθυνση, όπου στην προκειμένη περίπτωση είναι <http://127.0.0.1:64210/>
6. Σε ένα browser της επιλογή μας την πληκτρολογούμε και εμφανίζεται το περιβάλλον της Cayley, όπως δείχνει και η Εικόνα 9.



Εικόνα 9 - Περιβάλλον Cayley

4.3 Το περιβάλλον της Cayley

Το περιβάλλον της Cayley παρέχει τις εξής λειτουργικότητες: ερώτημα (query), query shape, οπτικοποίηση των ερωτημάτων (visualize), και εγγραφή (write), όπως δείχνει και η Εικόνα 10.



Εικόνα 10 - Περιβάλλον Cayley

Με την επιλογή *ερώτημα* (query), ο χρήστης δημιουργείται ένα ερώτημα είτε με τη γλώσσα Gremlin είτε με τη γλώσσα Gizmo, και πατώντας το κουμπί Run Query, εμφανίζεται το αποτέλεσμα της ερώτησης. Με την επιλογή *query shape*, παρέχεται μόνο μια απεικόνιση του ερωτήματος, χωρίς να υπάρχει η δυνατότητα δημιουργίας ερωτήματος.

Με την επιλογή *οπτικοποίηση* (virtualize) εκτελείται ένα ερώτημα και το αποτέλεσμά του οπτικοποιείται ως γράφος. Πιο συγκεκριμένα, για να χρησιμοποιήσετε τη λειτουργία απεικόνισης, θα πρέπει να παραχθούν, είτε μέσω ετικετών (tags) είτε επεξεργασίας με Javascript, ένα σύνολο αντικειμένων JSON που περιέχουν τα κλειδιά πηγή (source) και στόχος (target). Αυτοί θα είναι οι σύνδεσμοι και οι κόμβοι θα υπολογιστούν αυτόματα.

Η λειτουργία απεικόνισης προσπαθεί να ανιχνεύσει τους κόμβους είτε ως "πηγή" είτε ως "στόχο". Η πηγή σας αντιπροσωπεύεται ως μπλε κόμβος. Ενώ ο στόχος σας αντιπροσωπεύεται ως πορτοκαλί κόμβος. Η ιδέα είναι ότι η σχέση μεταξύ κόμβων μας οδηγεί από μπλε σε πορτοκαλί (πηγή προς στόχο).

Τέλος με την επιλογή *εγγραφή* υπάρχει η δυνατότητα στη διεπαφή να εγγράψει ή να αφαιρέσει μεμονωμένες τετράδες (quads) πληροφορίας.

4.3.1 Η Γλώσσα Gremlin

Η γλώσσα ερωτημάτων Gremlin, η οποία υποστηρίζεται από αρκετές βάσεις δεδομένων γράφων, χρησιμοποιείται και στην Cayley για ανάκτηση αλλά και τροποποίηση των δεδομένων του γράφου. Ουσιαστικά, η Gremlin είναι μια γλώσσα η οποία εκφράζει τις σύνθετες διαδρομές των γράφων.

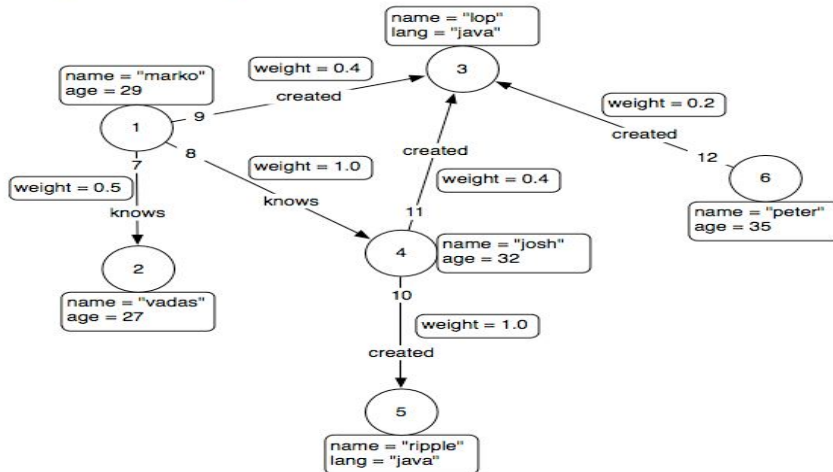
Έχει σχεδιαστεί για να αποθηκεύει τα δεδομένα σαν ζεύγη κλειδί – τιμή, ενώ επίσης αναλύει και χειρίζεται κατευθυνόμενες και πολλαπλές σχέσεις στους γράφους της. Οι λειτουργίες της Gremlin είναι:

- Με τον όρο *κλειδί – τιμή* αναφέρεται στο γεγονός ότι οι 2 κορυφές και οι ακμές που τις συνδέουν μπορούν να έχουν οποιοδήποτε αριθμό ιδιοτήτων.
- Ο όρος *κατευθυνόμενες* αναφέρεται στο γεγονός ότι οι σχέσεις του γράφου έχουν κατευθυντικότητα.
- Τέλος, ο όρος *πολλαπλές* αναφέρεται στο γεγονός ότι μπορεί να υπάρχουν πολλοί τύποι ακμών, έτσι δημιουργούνται πολλοί τύποι συσχετίσεων μεταξύ των κορυφών.

Στην Εικόνα 11 που ακολουθεί εμφανίζεται ένας γράφος ιδιοτήτων, ο οποίος περιέχει ένα σύνολο κόμβων και ένα σύνολο ακμών.

Κάθε σύνολο κόμβων διαθέτει ένα μοναδικό αναγνωριστικό. Επίσης, κάθε κόμβος έχει ένα σύνολο εξερχόμενων ακμών αλλά και ένα σύνολο εισερχόμενων ακμών και μια συλλογή ιδιοτήτων που ορίζεται ως απεικόνιση από το κλειδί στην τιμή.

Το σύνολο των ακμών διαθέτουν τα έξης χαρακτηριστικά: κάθε ακμή έχει ένα μοναδικό αναγνωριστικό, εξέρχεται από μία κορυφή και καταλήγει σε μία κορυφή. Επίσης, διαθέτει μια ετικέτα που υποδηλώνει τον τύπο της σχέσης μεταξύ των δύο κορυφών που συνδέει. Τέλος, κάθε ακμή έχει μια συλλογή ιδιοτήτων που ορίζεται ως απεικόνιση από το κλειδί στην τιμή.



Εικόνα 11 - Απεικόνιση γράφου

4.3.2 Πλεονεκτήματα της Gremlin

Πριν εμβαθύνουμε στα ιδιαίτερα χαρακτηριστικά της Gremlin, είναι καλό να παραθέσουμε μερικά πλεονεκτήματά της.

Η Gremlin είναι χρήσιμη για να δουλεύουμε σε γραφήματα: Μπορούμε να περιηγούμαστε στο γράφημα, να κάνουμε ανανέωση ιδιοτήτων στους κόμβους, να προσθέτουμε ακμές, να αφαιρούμε κόμβους, κ.λπ. Η Gremlin είναι ιδανική για τη διαχείριση του γραφήματος και τη διασφάλιση της ακεραιότητας των δεδομένων.

Η Gremlin επιτρέπει τη δημιουργία ερωτημάτων γραφημάτων: πολλά είδη ερωτημάτων μπορούν να εκφραστούν με πολύ πιο συνοπτικό και κατανοητό τρόπο μέσω πράξεων γραφημάτων από ό,τι με τις παραδοσιακές γλώσσες προγραμματισμού. Εάν αναπτύσσουμε μια εφαρμογή που απαιτεί ερωτήματα γραφήματος, τότε ίσως είναι καλύτερο να χρησιμοποιηθεί η Gremlin.

Η Gremlin μπορεί να εκφράσει συνοπτικά περίπλοκες διαδρομές γραφημάτων: Η Gremlin μπορεί να εκφράσει σε μερικές γραμμές κώδικα κάτι που θα απαιτούσε πολύ περισσότερες γραμμές σε κώδικα Java. Επιπλέον, η Gremlin σχεδιάστηκε για να αναπαριστά περίπλοκες διαδρομές γραφημάτων. Ακόμη και οι σημερινές βάσεις δεδομένων γραφημάτων δεν έχουν την δυνατότητα να εκφράζουν περίπλοκες διαδρομές γραφημάτων - κάτι που η Gremlin μπορεί να κάνει πολύ κομψά.

Η Gremlin είναι χρήσιμη για να εξερευνήση και εκμάθηση σχετικά με γραφήματα: Υπάρχουν πολλά πρόσθετα / συνδέσεις για την Gremlin που το καθιστούν ένα μοναδικό περιβάλλον για να εργαζόμαστε με γραφήματα.

Η Gremlin επιτρέπει την εξερευνήση του Σημασιολογικού Ιστού / Ιστού Δεδομένων: Η Gremlin μπορεί να χρησιμοποιηθεί με γραφήματα RDF. Επιπλέον, η Gremlin διαθέτει ένα σύνδεσμο για το LinkedData Sail το οποίο μας επιτρέπει να εργαζόμαστε με τον Σημασιολογικό Ιστό / Ιστό Δεδομένων σε πραγματικό χρόνο.

Η Gremlin δεν είναι συνδεδεμένη με ένα συγκεκριμένο backend γραφημάτων: η Gremlin μπορεί να χρησιμοποιηθεί με διάφορες βάσεις δεδομένων γραφημάτων. Έτσι, μπορούμε να χρησιμοποιήσετε τον ίδιο κώδικα Gremlin ακόμα και όταν αλλάζετε το backend της βάσης δεδομένων.

Η Gremlin επιτρέπει υπολογισμούς με βάση τη διαδρομή (path-based): Η Gremlin μάς επιτρέπει να εξετάζουμε και να διαμορφώνουμε τις διαδρομές μέσα σε ένα γράφημα. Αυτό παρέχει την ευελιξία να δημιουργηθούν εξελιγμένοι τρόποι για την ανάλυση, την αναζήτηση και την επεξεργασία γραφημάτων.

Η Gremlin είναι επεκτάσιμη και μπορεί να προσαρμοστεί σε κάθε συγκεκριμένη περίπτωση: Είναι δυνατό να επεκταθεί την Gremlin με νέες μεθόδους και βήματα που έχουν καθοριστεί χρησιμοποιώντας Gremlin ή Java. Επιπλέον, είναι δυνατό να προστεθούν άλλες βάσεις δεδομένων γραφημάτων, υλοποιώντας τις διεπαφές που παρέχονται από τα πρότυπα.

Η Gremlin χρησιμοποιεί το Java API: Η εικονική μηχανή για την Gremlin είναι η εικονική μηχανή Java. Ο κώδικας Gremlin μπορεί να επωφεληθεί από το Java API. Με αυτόν τον τρόπο, η Gremlin και η Java λειτουργούν άψογα μαζί.

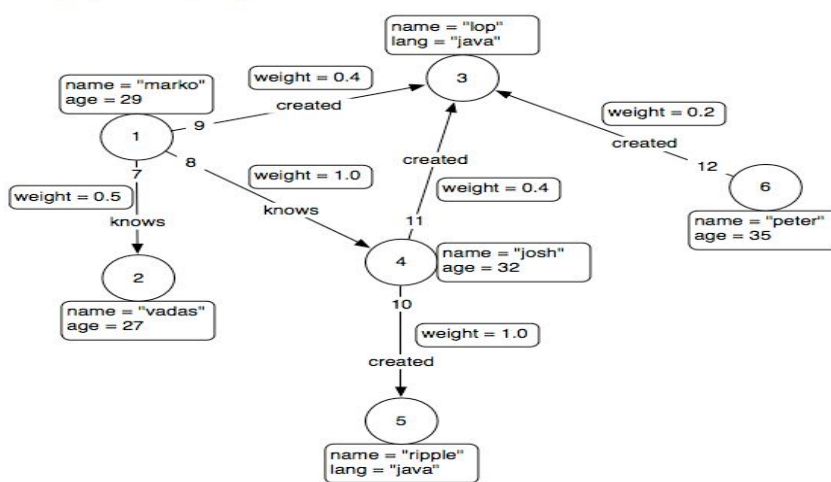
Η Gremlin είναι ενσωματωμένη σε διάφορες γλώσσες JVM: Ο πυρήνας της Gremlin είναι γραμμένος σε Java. Ωστόσο, η Gremlin διανέμεται με προβλέψεις χρήσης για τη γλώσσα Groovy. Με αυτόν τον τρόπο, η Gremlin είναι περισσότερο ένας τρόπος περιγραφής διαδρομών πέρα από μια γλώσσα, που άλλες γλώσσες JVM μπορούν φυσικά να φιλοξενήσουν.

Η Gremlin είναι Turing-πλήρης: Πολλές γλώσσες διαδρομών γραφημάτων μπορούν να αναγνωρίσουν μόνο τα κανονικά μονοπάτια μέσα σε ένα γράφημα. Η Gremlin παρέχει πολυάριθμες κατασκευές μνήμης και υπολογισμών για την υποστήριξη αυθαίρετης αναγνώρισης του μονοπατιού.

4.4 Δημιουργία ερωτημάτων στην Gremlin

Η γλώσσα ερωτημάτων Gremlin χρησιμοποιείται για ερωτήματα, ανάλυση αλλά και χειρισμό των δεδομένων. Ένα ερώτημα Gremlin περιλαμβάνει μια αλυσίδα λειτουργιών οι οποίες αποτιμώνται από αριστερά προς τα δεξιά.

Εισάγονται οι έννοιες για τη λειτουργία των ερωτημάτων στους γράφους. Υπάρχει δυνατότητα επιλογής ενός ή περισσότερων κόμβων όπου με τη χρήση της γλώσσας Gremlin θα διασχιστεί ο γράφος. Παρακάτω παρουσιάζονται τα βασικά στοιχεία για την σύνταξη της γλώσσας Gremlin, ενώ τα σχετικά παραδείγματα αναφέρονται στον γράφο της Εικόνας 12.



Εικόνα 12 - Απεικόνιση γράφου

Στοιχείο V

Το στοιχείο V χρησιμοποιείται για να επιλέξει όλους τους κόμβους του γράφου, με μια συγκριμένη ιδιότητα την οποία ορίζουμε.

```
gremlin> g.V("name", "marko")
==>v[1]
```

Όπως φαίνεται και στην Εικόνα 12, στο παραπάνω παράδειγμα ζητείται σαν ερώτηση να επιστραφούν όσοι κόμβοι έχουν στην ιδιότητα «name» την τιμή «marko», οπότε το αποτέλεσμα του ερωτήματος είναι ο κόμβος 1.

Στοιχείο E

Το στοιχείο E διατρέχει όλες τις ακμές του γράφου. Χρησιμοποιείται για να διατρέξει όλες τις ακμές του γράφου ωστόσο στους μεγάλους γράφους χρησιμοποιείται με προσοχή.

```
Gremlin> g.E
==>e[10][4-created->5]
==>e[7][1-knows->2]
==>e[9][1-created->3]
==>e[8][1-knows->4]
==>e[11][4-created->3]
==>e[12][6-created->3]
```

Όπως φαίνεται και στο παραπάνω παράδειγμα το στοιχείο E διατρέχει όλους τους κόμβους του γράφου και αποτυπώνει τη σχέση που υφίσταται ανάμεσα σε 2 κόμβους. Για παράδειγμα, ανάμεσα στους κόμβους 1 και 2 υπάρχει η σχέση “created”.

Στοιχείο id

Το στοιχείο id επιστρέφει τη μοναδική ταυτότητα του στοιχείου.

```
gremlin> v = g.V("name", "marko").next()
==>v[1]
gremlin> v.id
==>1
gremlin> g.v(1).id
==>1
```

Στο παραπάνω παράδειγμα τίθεται σαν ερώτημα στον γράφο να επιστρέψει το όνομα του κόμβου με όνομα `macro`, επομένως υπάρχει ένας και μοναδικός κόμβος με αυτήν την ταυτότητα και αυτός είναι ο κόμβος 1.

Στοιχείο in

Επιστρέφει τους κόμβους οι οποίοι διαθέτουν εξερχόμενη ακμή με δοσμένο τύπο σχέσης. Για παράδειγμα, η εκτέλεση του κάτωθι ερωτήματος στον γράφο του σχήματος 12 θα επιστρέψει τους κόμβους 1, 4 και 6, καθώς αυτοί διαθέτουν εξερχόμενη ακμή με τύπο σχέσης «`created`».

```
gremlin> v.in("created")
==>v[1]
==>v[4]
==>v[6]
```

Στοιχείο order

Ταξινομεί τα αντικείμενα με βάση την συνάρτηση που ορίζει ο χρήστης. Αν δε δοθεί συνάρτηση, χρησιμοποιείται η βασική σειρά στη ταξινόμηση.

```
gremlin> g.V.name.order
==>josh
==>lop
==>marko
==>peter
==>ripple
==>vadas
```

Το παραπάνω παράδειγμα περιγράφει την ταξινόμηση των κόμβων θέτοντας σαν κριτήριο ταξινόμησης το όνομα τους, το αποτέλεσμα το οποίο προκύπτει η λεξικογραφική και αύξουσα ταξινόμηση των κόμβων σύμφωνα με την τιμή του πεδίου «`name`» που έχει ο καθένας.

Στοιχείο out

Επιστρέφει τους κόμβους οι οποίοι διαθέτουν εισερχόμενη ακμή με δοσμένο τύπο σχέσης. Για παράδειγμα, η εκτέλεση του κάτωθι ερωτήματος στον γράφο του σχήματος 12 θα επιστρέψει τους κόμβους 2, και 4, καθώς αυτοί διαθέτουν εισερχόμενη ακμή με τύπο σχέσης «knows».

```
gremlin> v.out('knows')  
  
==>v[2]  
  
==>v[4]
```

Στοιχείο outE

Επιστρέφει τις εξερχόμενες ακμές του συγκεκριμένου κόμβου.

```
gremlin> v = g.V("name", "marko").next()  
  
==>v[1]  
  
gremlin> v.outE  
  
==>e[7][1-knows->2]  
  
==>e[8][1-knows->4]  
  
==>e[9][1-created->3]
```

Στο παραπάνω παράδειγμα, κάνοντας χρήση του στοιχείου outE για το ερώτημα αλλά και της Εικόνας 12, παρατηρείται ότι το αποτέλεσμα του γράφου στο ερώτημα είναι ότι η επιστροφή όλων των ακμών για τον κόμβο 1. Έτσι το αποτέλεσμα του γράφου σε αυτό το ερώτημα να επιστραφούν οι ακμές και οι ιδιότητες τους για τον κόμβο 1, πιο συγκεκριμένα παρατηρείται ότι ο κόμβος 1 έχει σχέση με τον κόμβο 2 και διαθέτουν την ιδιότητα knows, ο κόμβος 1 με τον κόμβο 4 διαθέτουν επίσης την ιδιότητα knows και ο κόμβος 1 με τον κόμβο 3 διαθέτουν την ιδιότητα created.

Στοιχείο outV

Επιστρέφει τον κόμβο από τον οποίο εξέρχεται η συγκεκριμένη ακμή.

```
gremlin> e = g.e(12)
```

```
==>e[12][6-created->3]
gremlin> e.outV
==>v[6]
```

Σχετικά με το στοιχείο `outV`, στο παραπάνω παράδειγμα παρατηρείται ότι τίθεται σαν ερώτημα από ποιον κόμβο εξέρχεται η ακμή 12 και το αποτέλεσμα το οποίο λαμβάνεται είναι ότι η ακμή 12 συνδέει τους κόμβους 6 και 3, εξερχόμενη από τον κόμβο 6.

Στοιχείο *has*

Επιλέγει ένα στοιχείο αν έχει μια συγκεκριμένη ιδιότητα. Χρησιμοποιεί διάφορες επιλογές για σύγκριση μέσω του `T`, οι επιλογές που μπορούν να χρησιμοποιηθούν αναφέρονται παρακάτω.

- `T.gt` - greater than (μεγαλύτερο από)
- `T.gte` - greater than or equal to (μεγαλύτερο ή ίσο από)
- `T.eq` - equal to (ίσο με)
- `T.neq` - not equal to (διάφορο από)
- `T.lte` - less than or equal to (μικρότερο ή ίσο από)
- `T.lt` - less than (μικρότερο από)
- `T.in` - contained in a list (περιέχεται στην λίστα)
- `T.notin` - not contained in a list (δεν περιέχεται στην λίστα)

Αξίζει να επισημάνουμε ότι η σύνταξη του *has* είναι αντίστοιχη με την `g.V("name", "marko")`, με τη διαφορά ότι η `g.V("name", "marko")` είναι μια αναζήτηση σε κλειδί με ευρετήριο, με αποτέλεσμα να είναι γρήγορη. Σε αντίθεση η εντολή `g.V.has("name", "marko")`, θα διατρέξει όλους τους κόμβους ελέγχοντας για την ιδιότητα `name` σε κάθε κόμβο και θα είναι πολύ πιο αργή.

Η συμπεριφορά του *has* έχει να κάνει με την υλοποίηση της στο σύστημα και τα παραπάνω ισχύουν στα περισσότερα συστήματα. Για παράδειγμα, η *Cayley* θα

προσπαθήσει να χρησιμοποιήσει ευρετήριο οπουδήποτε μπορεί να το κάνει. Δηλαδή είναι σημαντικό να καταλάβουμε την υλοποίηση του κάθε συστήματος όταν γράφουμε ερωτήματα.

```
gremlin> g.v(1).outE.has("weight", T.gte, 0.5f).weight
==>0.5
==>1.0

gremlin> g.V.has('age').name
==>vadas
==>marko
==>peter
==>josh
```

Αρχικά στο πρώτο παράδειγμα της ιδιότητας `has`, ζητείται σαν ερώτημα να επιστρέφει τις εξερχόμενες ακμές του κόμβου 1 οι οποίες διαθέτουν βάρος μεγαλύτερο ή ίσο με 0,5f, έτσι διασχίζεται ολόκληρος ο γράφος και δίνει σαν αποτέλεσμα τους κόμβους με βάρος 0,5 και 1,0.

Στο δεύτερο παράδειγμα του στοιχείου `has`, τίθεται σαν ερώτημα να εμφανίσει τα ονόματα όσων κόμβων έχουν την ιδιότητα «age», επομένως το ερώτημα θα διασχίσει όλους τους κόμβους και θα επιστρέψει όσους διαθέτουν όνομα και ηλικία που στην προκειμένη περίπτωση είναι οι κόμβοι με όνομα `vadas`, `marko`, `peter` και `josh`.

5. Σενάριο χρήσης της Caley σε εφαρμογή

5.1 Αρχιτεκτονική

Στα πλαίσια της εφαρμογής που θα περιγράψουμε ως σενάριο χρήσης της Caley, ως πηγή αρχείου θα χρησιμοποιηθεί μια βάση δεδομένων γράφων η οποία περιγράφει ένα υποθετικό γενεαλογικό δέντρο.

Συγκεκριμένα, το αρχείο το οποίο θα δημιουργήσουμε θα έχει την μορφή τετράδων (N-quads) απλού κείμενου, όπου στο κείμενο αυτό θα περιγράφεται ο γράφος ενός οικογενειακού δέντρου. Επίσης, αυτή η κωδικοποίηση χρησιμοποιείται για την κωδικοποίηση ενός συνόλου δεδομένων RDF.

Η μορφή των τετράδων (N-quads) αποτελεί επέκταση των τριπλετών (N-triples), με την κύρια διαφορά των τετράδων είναι στο ότι επιτρέπουν την κωδικοποίηση πολλαπλών γράφων. Το οικογενειακό δέντρο περιγράφεται στην επόμενη ενότητα.

5.2 Σενάριο Χρήσης Εφαρμογής

Για την απεικόνιση των δεδομένων δημιουργήθηκε μια τετράδα χρησιμοποιώντας τα ονόματα και τις σχέσεις των μελών της οικογένειας, και στη συνέχεια δημιουργούμε ένα αρχείο με όνομα 'familytree.nq'. Η απεικόνιση των δεδομένων παρουσιάζεται παρακάτω στον Πίνακα 8.

Το οικογενειακό δέντρο το οποίο δημιουργήθηκε και θα υπάρξει απεικόνιση μέσω γράφων, στη βάση Cayley αφορά 2 αδέρφια, τη Γεωργία και τον Γιάννη. Πιο συγκεκριμένα, η Γεωργία γεννήθηκε το 1954 και ο Γιάννης το 1956. Η Γεωργία μένει στην Μεγαλόπολη και είναι παντρεμένη με τον Κώστα Μπαρέκα ο οποίος έχει γεννηθεί το 1950. Η Γεωργία έχει 3 παιδιά, τον Βασίλη που γεννήθηκε το 1979, τον Στάθη που γεννήθηκε το 1979 και τη Μαρία που γεννήθηκε το 1988. Με τη σειρά του, ο Γιάννης έχει το επίθετο Οικονόμου, μένει στην Κόρινθο και είναι παντρεμένος με την Άννα η οποία γεννήθηκε το 1960. Έχουν 2 παιδιά τον Γεώργιο που γεννήθηκε το 1980 και την Λίζα που γεννήθηκε το 1985.

Τώρα, ο Βασίλης μένει στην Πάτρα, είναι παντρεμένος με την Αμαλία η οποία γεννήθηκε το 1980 και έχουν 2 παιδιά τον Κωνσταντίνο που γεννήθηκε το 2013 και

την Όλγα που γεννήθηκε το 2017. Η Αμαλία έχει μια αδελφή δίδυμη την Τίνα Χαρίση η οποία μένει στην Άρτα και γεννήθηκε το 1984. Ο Στάθης μένει στην Αθήνα είναι παντρεμένος με την Ελένη που γεννήθηκε το 1986 και έχουν 2 παιδιά, τον Φίλιππο που γεννήθηκε το 2012 και τον Πέτρο που γεννήθηκε το 2014. Η Ελένη έχει έναν αδελφό με το όνομα Νίκος Πέτρου που μένει στην Αθήνα και γεννήθηκε το 1983. Η Μαρία μένει στην Καλαμάτα, είναι παντρεμένη με τον Δημήτρη που έχει γεννηθεί το 1982 και έχουν ένα παιδί τον Ανδρέα που έχει γεννηθεί το 2016. Ο Δημήτρης έχει 2 αδέρφια τον Βασίλη και τον Τέλη που έχουν γεννηθεί το 1987 και το 1993, αντίστοιχα και μένουν και οι 2 στην Καλαμάτα.

Ο Γιώργος μένει στην Αθήνα, παντρεύτηκε τη Χριστίνα που είναι γεννημένη το 1983 και έχουν ένα παιδί με όνομα Κατερίνα που έχει γεννηθεί το 2015. Η Λίζα μένει στην Θεσσαλονίκη είναι παντρεμένη με τον Απόστολο Νίκου που έχει γεννηθεί το 1978 και δεν έχουν παιδιά.

"georgia" "name" "Georgia Barekas" .

"georgia" "born" "1954" .

"georgia" "lives" "Megalopolis" .

"john" "name" "John Oikonomou" .

"john" "born" "1956" .

"john" "lives" "Korinthos" .

"georgia" "sibling" "john" .

"kostas" "name" "Kostas Barekas" .

"kostas" "born" "1950" .

"anna" "name" "Anna Oikonomou" .

"anna" "born" "1960" .

"bill" "name" "Bill Barekas" .

"bill" "born" "1975" .

"steve" "name" "Steve Barekas" .

"steve" "born" "1979" .

"maria" "name" "Maria Barekas" .

"george" "name" "George Oikonomou" .

"george" "born" "1980" .

"liza" "name" "Liza Oikonomou" .

"liza" "born" "1960" .

"amalia" "name" "Amalia Barekas" .

"amalia" "born" "1982" .

"konstantinos" "name" "Konstantinos Barekas" .

"konstantinos" "born" "2013" .

"olga" "name" "Olga Barekas" .

"olga" "born" "2017" .

"tina" "name" "Tina Charisis" .

"tina" "born" "1982" .

"tina" "lives" "Artas" .

"helen" "name" "Helen Barekas" .

"helen" "born" "1986" .

"filippos" "name" "Filippos Barekas" .

"filippos" "born" "2012" .

"petros" "name" "Petros Barekas" .

"petros" "born" "2014" .

"nikos" "name" "Nikos Petrou" .

"nikos" "born" "1983" .

"nikos" "lives" "Athens" .

"jim" "name" "Jim Karlis" .

"jim" "born" "1982" .

"basilis" "name" "Basilis Karlis" .

"basilis" "born" "1987" .

"basilis" "lives" "Kalamatas" .

"telis" "name" "Telis Karlis" .

"telis" "born" "1993" .

"telis" "lives" "Tripolis" .

"andreas" "name" "Andreas Karlis" .

"andreas" "born" "2016" .

"christina" "name" "Christina Oikonomou" .

"christina" "born" "1983" .

"kathrin" "name" "Kathrin Oikonomou" .

"kathrin" "born" "2015" .

"apostolis" "name" "Apostolis Nikou" .

"apostolic" "born" "1978" .

"bill" "lives" "Patras" .
"steve" "lives" "Athens" .
"maria" "lives" "Kalamatas" .
"george" "lives" "Athens" .
"liza" "lives" "Thesallonikis" .
"georgia" "spouse" "kostas" .
"georgia" "child" "bill" .
"georgia" "child" "steve" .
"georgia" "child" "maria" .
"john" "spouse" "anna" .
"john" "child" "george" .
"john" "child" "liza" .
"bill" "spouse" "amalia" .
"bill" "child" "konstantinos" .
"bill" "child" "olga" .
"steve" "spouse" "helen" .
"steve" "child" "filippos" .
"steve" "child" "petros" .
"maria" "spouse" "jim" .
"maria" "child" "andreas" .
"george" "spouse" "christina" .

"george" "child" "kathrin" .

"liza" "spouse" "apostolis" .

"bill" "brother" "steve" .

"bill" "sibling" "maria" .

"steve" "sibling" "maria" .

"george" "sibling" "liza" .

"tina" "sibling" "amalia" .

"nikos" "sibling" "helen" .

"basilis" "brother" "jim" .

"telis" "brother" "jim" .

Πίνακας 8 – Δεδομένα Σεναρίου Χρήσης Εφαρμογής

5.3 Μεταφόρτωση αρχείου στη Cayley

Μετά τη δημιουργία του αρχείου με όνομα 'familytree.nq', το οποίο περιέχει τετράδες με τα δεδομένα ενός οικογενειακού δέντρου, θα πρέπει να το προσθέσουμε στη βάση δεδομένων γράφων Cayley, προκειμένου δημιουργηθούν ερωτήματα και να απαντηθούν.

Η διαδικασία της μεταφόρτωσης αρχικά απαιτεί το άνοιγμα της γραμμής εντολών και στην συνέχεια πληκτρολογείτε η εντολή

```
.\cayley.exe http --dbpath="familytree.nq"
```

όπως φαίνεται και στην Εικόνα 13.

```
C:\cayley_v0.6.1_windows_386>.\cayley.exe http --dbpath="familytree.nq"  
Serving Gephi GraphStream at http://localhost:64210/gephi/gs  
Cayley now listening on 127.0.0.1:64210
```

Εικόνα 13 -Μεταφόρτωση Βάσης Γράφου στην Cayley

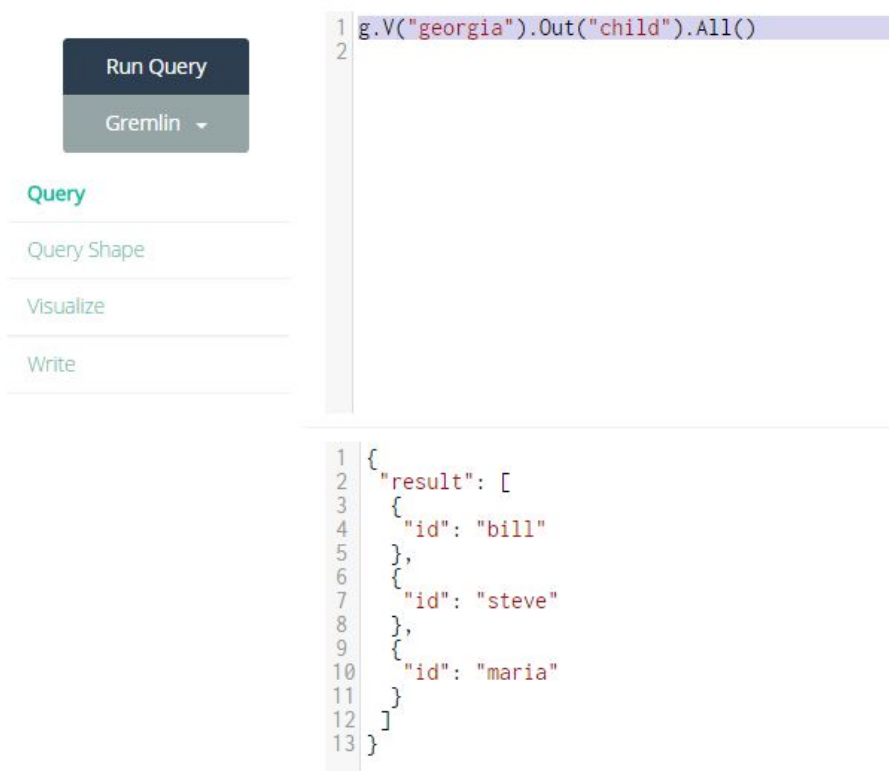
Εμφανίζεται, μια διεύθυνση, στην προκειμένη περίπτωση είναι η 127.0.0.1:64210, την οποία θα ανοίξουμε σε ένα browser και θα εμφανίσει τη βάση που αφορά το οικογενειακό δέντρο που δημιουργήσαμε. Στη συνέχεια, με τη δημιουργία των κατάλληλων ερωτημάτων, η βάση δεδομένων γράφων για το οικογενειακό δέντρο, θα επιστρέφει τα αποτελέσματα στις ερωτήσεις που υποβάλλονται.

5.4 Δημιουργία Ερωτημάτων

Για την δημιουργία ερωτημάτων στην εφαρμογή μας χρησιμοποιείται η γλώσσα Gremlin, όπου παραπάνω έχουν αναφερθεί τα χαρακτηριστικά της, υπάρχει δυνατότητα δημιουργίας ερωτημάτων είτε μέσω του πεδίου ερωτήματος (query) είτε μέσω του πεδίου της απεικόνιση ερωτημάτων (visualize). Επομένως, θα περιγράφεται το κάθε ερώτημα, η σύνταξη του καθώς και το αποτέλεσμα του κάθε ερωτήματος το οποίο θα προκύπτει είτε μέσα από το πεδίο query είτε από το πεδίο visualize.

5.4.1 Δημιουργία Ερωτημάτων στο πεδίο query

Το πρώτο ερώτημα το οποίο θα αφορά όλα τα παιδιά της Γεωργίας, επομένως θα διασχίσει τους κόμβους από την αρχή προς το τέλος χρησιμοποιώντας το στοιχείο out και θα εμφανίσει όλα τα άτομα με την ιδιότητα "child" θα δημιουργηθεί στο πεδίο query και θα έχει σύνταξη: `g.V("georgia").Out("child").All()`, το αποτέλεσμα της Cayley εμφανίζεται στην Εικόνα 14.



The screenshot shows the Cayley web interface. On the left, there is a sidebar with a 'Run Query' button and a 'Gremlin' dropdown menu. Below this are links for 'Query', 'Query Shape', 'Visualize', and 'Write'. The main area contains a query editor with the following code:

```
1 g.V("georgia").Out("child").All()
2
```

Below the query editor, the results are displayed in a JSON format:

```
1 {
2   "result": [
3     {
4       "id": "bill"
5     },
6     {
7       "id": "steve"
8     },
9     {
10      "id": "maria"
11    }
12  ]
13 }
```

Εικόνα 14 - Αποτελέσματα Ερωτήματος που εμφανίζει όλα τα παιδιά της Γεωργίας

Το επόμενο ερώτημα το οποίο θέτουμε στη βάση είναι να μας επιστρέψει όλα τα άτομα τα οποία έχουν γεννηθεί το 1982, επομένως θα χρησιμοποιηθεί το στοιχείο `in`, αφού θα πρέπει να ανακτήσει τους κόμβους από τους οποίους εξέρχονται οι ακμές. Το ερώτημα το οποίο θα γράψουμε στο πεδίο `query` θα έχει σύνταξη: **`g.V("1982").In("born").All()`**. Το αποτέλεσμα της Cayley παρουσιάζεται στην Εικόνα 15.



Run Query
Gremlin ▾

Query
Query Shape
Visualize
Write

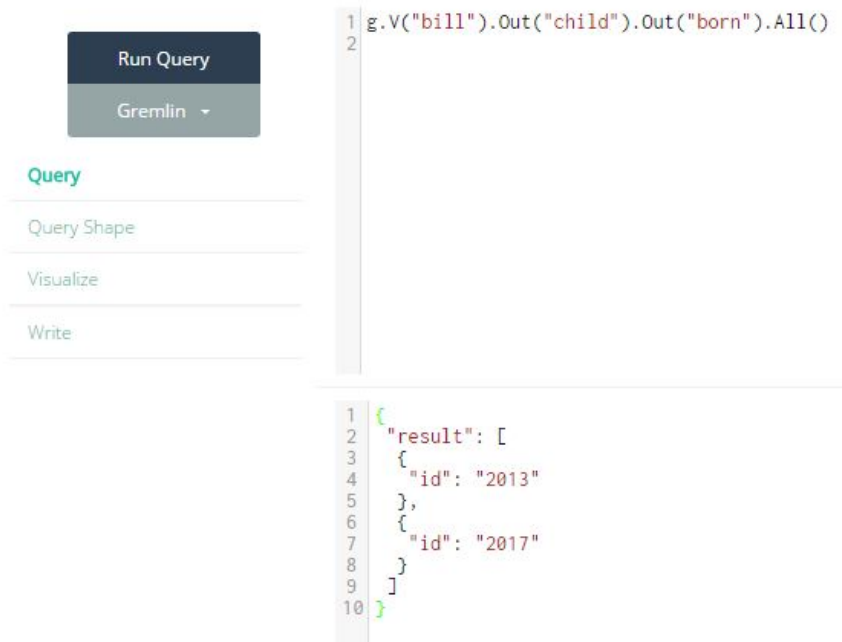
```
1 g.V("1982").In("born").All()
2
```

```
1 {
2   "result": [
3     {
4       "id": "amalia"
5     },
6     {
7       "id": "tina"
8     },
9     {
10      "id": "jim"
11    }
12  ]
13 }
```

Εικόνα 15 - Αποτέλεσμα Ερωτήματος όσων έχουν γεννηθεί το 1982

Το τελευταίο ερώτημα το οποίο θέτουμε στην βάση είναι να επιστρέψει το έτος γέννησης των παιδιών του Βασίλη, επομένως θα χρησιμοποιηθεί το στοιχείο out τόσο για την ιδιότητα “παιδί” όσο και για την ιδιότητα “γέννησης” και θα πρέπει να διασχίζει τους κόμβους από την αρχή προς την τέλος. Το ερώτημα το οποίο θα γράψουμε στο πεδίο query θα έχει σύνταξη: **g.V("bill").Out("child").Out("born").All()**

Το αποτέλεσμα της Cayley παρουσιάζεται στην Εικόνα 16.



The screenshot shows the Cayley web interface. On the left, there is a sidebar with a 'Run Query' button and a 'Gremlin' dropdown menu. Below this are several tabs: 'Query', 'Query Shape', 'Visualize', and 'Write'. The 'Query' tab is active. The main area contains a query editor with the following text:

```
1 g.V("bill").Out("child").Out("born").All()
2
```

Below the query editor, the results pane shows the following JSON output:

```
1 {
2   "result": [
3     {
4       "id": "2013"
5     },
6     {
7       "id": "2017"
8     }
9   ]
10 }
```

Εικόνα 16 - Εμφάνιση Ερωτήματος του έτος γέννησης των παιδιών του Βασίλη

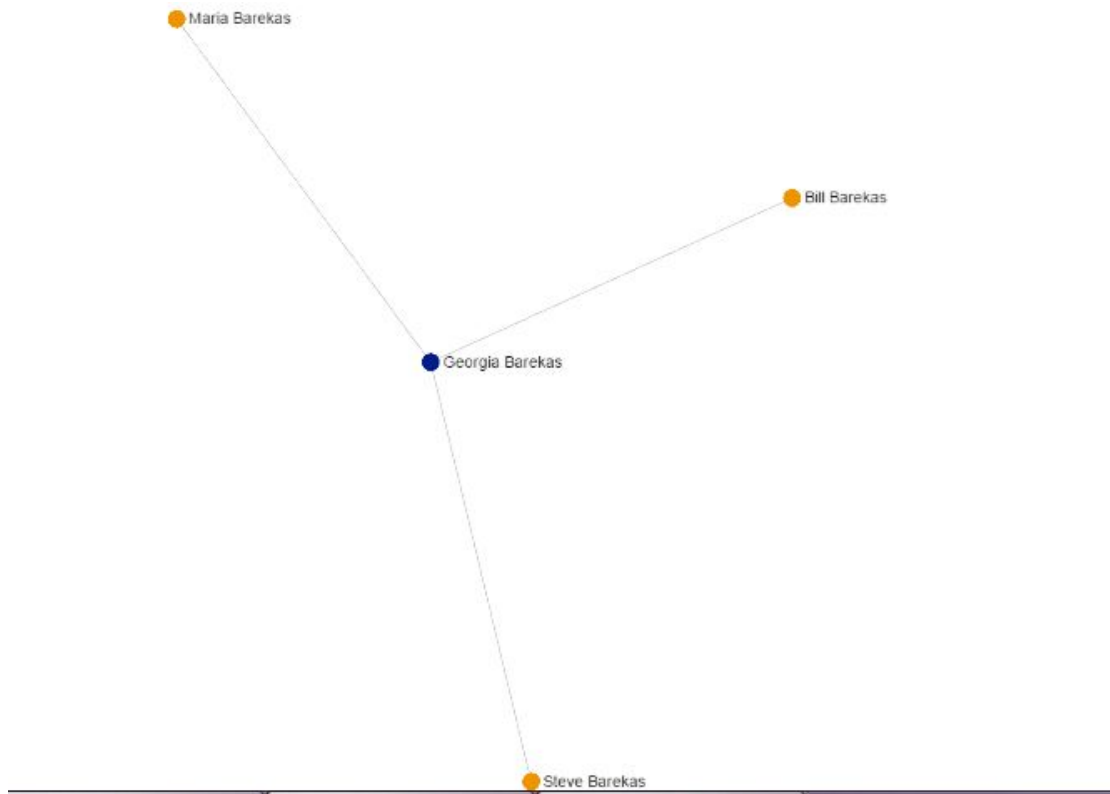
5.4.2 Δημιουργία Ερωτημάτων στο πεδίο Visualize

Τα ερωτήματα στο πεδίο Visualize (απεικόνιση) ανιχνεύει τους κόμβους είτε ως "πηγή" είτε ως "στόχο". Η πηγή θα αντιπροσωπεύεται ως μπλε κόμβος και ο στόχος θα αντιπροσωπεύεται ως πορτοκαλί κόμβος. Ενώ, η σύνταξη η οποία χρησιμοποιείται είναι διαφορετική από το πεδίο query, αφού γίνεται χρήση του στοιχείου `as` προκειμένου να ονομάσει το προηγούμενο βήμα. Παρακάτω περιγράφονται τα ερωτήματα απεικόνισης που πραγματοποιήθηκαν στην Cayley.

Αρχικά, το πρώτα ερώτημα απεικονίζει τα παιδιά της Γεωργίας και τα ονόματα αυτών, η σύνταξη του ερωτήματος είναι η εξής:

```
g.V("georgia").Save("name","source").As("source").Out("child",
"relation").Save("name","target").As("target").All()
```

Αρχικά, ορίζεται ο κόμβος με το όνομα της Γεωργίας, στην συνέχεια αποθηκεύεται το όνομα ως πηγή, έπειτα θα ζητείται να εμφανιστούν οι εξερχόμενοι κόμβοι οι οποίοι έχουν την σχέση παιδί και τέλος αποθηκεύονται σαν στόχος, οπύ προκύπτει και το αποτέλεσμα της Εικόνας 17.

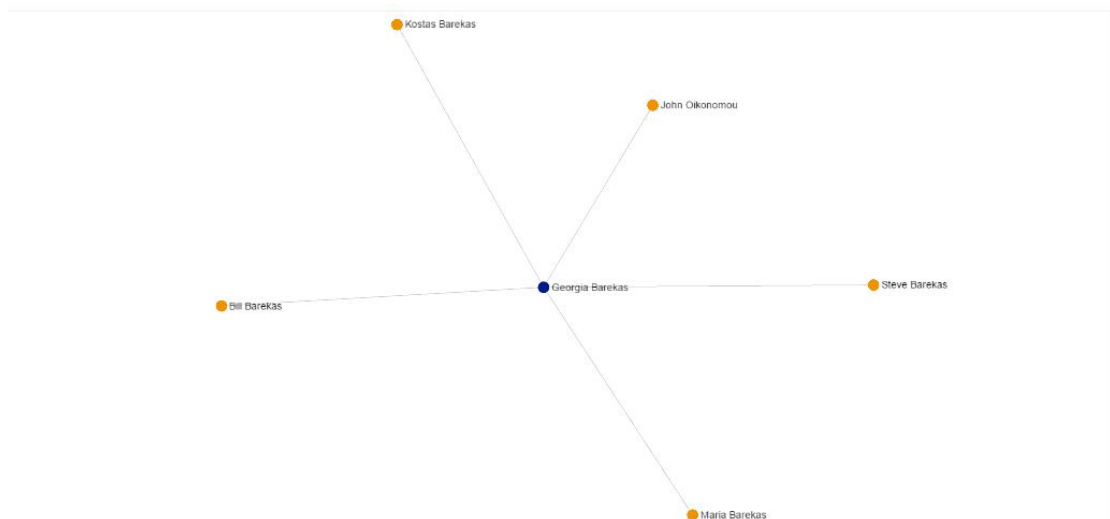


Εικόνα 17 - Οι Κόμβοι των παιδιών της Γεωργίας

Το επόμενο ερώτημα θα παρουσιάζει τον άντρα, τον αδελφό και τα παιδιά της Γεωργίας, η σύνταξη του ερωτήματος έχει την μορφή

```
g.V("georgia").Save("name","source").As("source").Out(["child",
"spouse", "sibling"],
"relation").Save("name","target").As("target").All()
```

Επομένως, η σύνταξη του ερωτήματος ορίζει αρχικά τον κόμβο με το όνομα της Γεωργίας, τον αποθηκεύει με το όνομα του σαν πηγή έτσι ώστε να υπάρχει η οπτικοποίηση του και στην συνέχεια ζητά τους εξερχόμενους κόμβους με ιδιότητα παιδί, αδελφού και άντρα και τους αποθηκεύει το όνομα τους σαν στόχο.



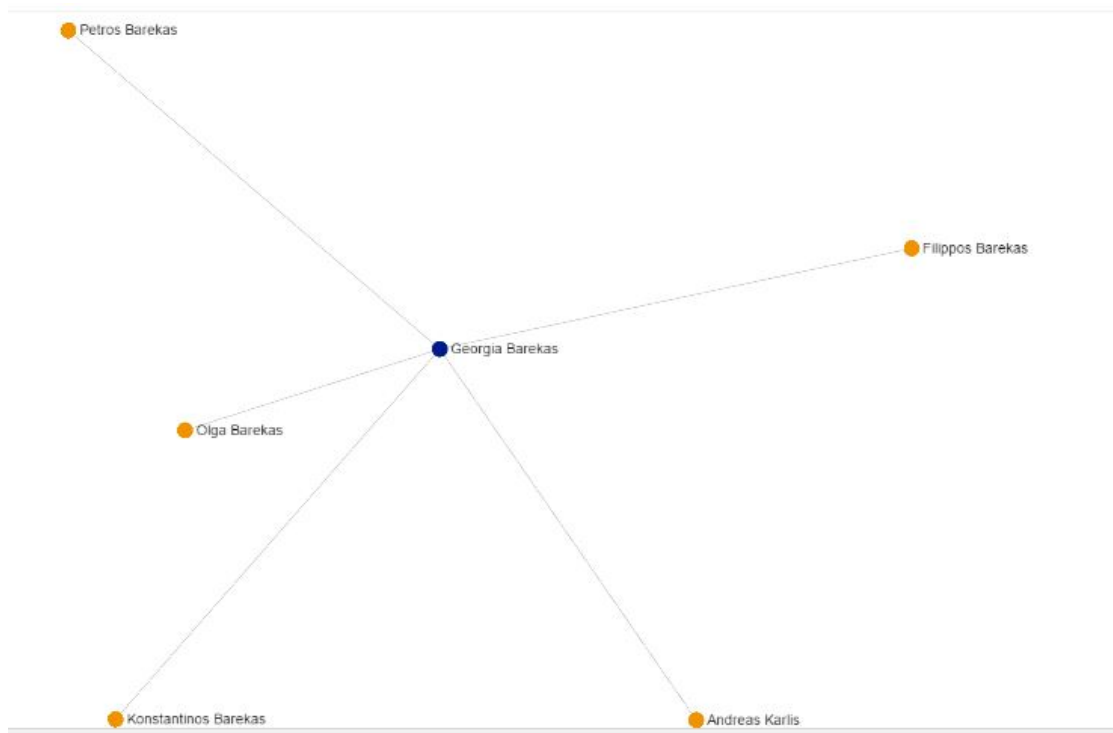
Εικόνα 18 - Οι κόμβοι εμφανίζουν τον άντρα, τον αδελφό και τα παιδιά της Γεωργίας

Το επόμενο ερώτημα θα εμφανίζει τα εγγόνια της Γεωργίας, έχοντας σύνταξη

```

g.V("georgia").Save("name","source").As("source").Out("child",
"relation").Out("child",
"relation").Save("name","target").As("target").All()
  
```

Το αποτέλεσμα του ερωτήματος εμφανίζεται στη Εικόνα 19.



Εικόνα 19 - Οι κόμβοι των εγγονιών της Γεωργίας

Επομένως, το παραπάνω ερώτημα ορίζει τον κόμβο της Γεωργίας, τον αποθηκεύει με το όνομα του σαν πηγή προκειμένου να υπάρξει απεικόνιση και στην συνέχεια ζητάει να εμφανιστούν με το στοιχείο out όλοι εκείνοι οι εξερχόμενοι κόμβοι που διαθέτουν την ιδιότητα του άντρα, του αδελφού και του παιδιού και συνδέονται με την Γεωργία, τέλος αποθηκεύονται τα ονόματα των κόμβων σαν στόχος έτσι ώστε να υπάρχει η οπτικοποίηση τους.

Το επόμενο και τελευταίο ερώτημα οπτικοποίησης εμφανίζει τόσα τα εγγόνια της Γεωργίας όσο και τους ενδιαμέσους κόμβους αυτών, δηλαδή τα παιδιά της, η σύνταξη του έχει ως εξής:

```
graph.V("georgia").Save('name',
'source').As('source').Out('child', 'relation').Save('name',
'target').As('target').ForEach(function(d) {

    g.Emit(d);

    g.V(d.id).Save('name', 'source').As('source').Out('child',
'relation').Save('name',
'target').As('target').ForEach(function(d2) {

        g.Emit(d2);

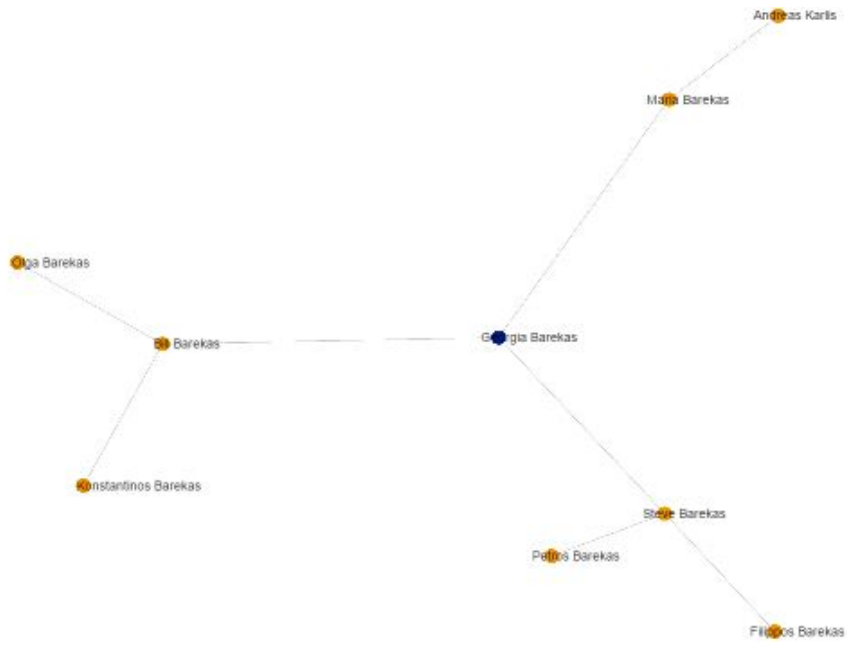
    } );

} );
```

και αποτέλεσμα του ερωτήματος εμφανίζεται στην Εικόνα 20.

Η σύνταξη του παραπάνω ερωτήματος είναι πιο σύνθετη, αρχικά ορίζει τον κόμβο της Γεωργίας, τον αποθηκεύει με το όνομα του σαν πηγή προκειμένου να υπάρξει απεικόνιση και στην συνέχεια ζητάει να εμφανιστούν με το στοιχείο out, όσοι εξερχόμενοι κόμβοι έχουν την ιδιότητα “παιδί” και αποθηκεύει το όνομα αυτών των κόμβων προκειμένου να υπάρξει απεικόνιση των ενδιαμέσων κόμβων, δηλαδή των παιδιών της Γεωργίας.

Στη συνέχεια για να εμφανιστούν και οι τελικοί κόμβοι δηλαδή τα εγγόνια της Γεωργίας, επαναλαμβάνεται το ερώτημα ακριβώς με την ίδια σύνταξη.



Εικόνα 20 - Οι κόμβοι των παιδιών και των εγγονιών της Γεωργίας

6. Σύνοψη

Στη παρούσα διπλωματική εργασία εξετάστηκαν οι βάσεις δεδομένων γράφων (Graph Data Bases). Αρχικά παρατέθηκε μία σύντομη επισκόπηση των σχεσιακών βάσεων δεδομένων και κατόπιν παρουσιάστηκαν οι βάσεις δεδομένων NoSQL, αναπτύχθηκε η τεχνολογία τους και πραγματοποιήθηκε μία συνοπτική σύγκριση μεταξύ τους.

Στη συνέχεια, αναλύθηκαν περαιτέρω οι βάσεις δεδομένων γράφων, και παρουσιάστηκε η δομή τους και τα πλεονεκτήματά τους. Τέλος, παρατέθηκε μία εκτενέστερη περιγραφή της βάσης δεδομένων γράφων Caley, καθώς και της γλώσσας ερωτήσεων Gremlin που αυτή χρησιμοποιεί, ενώ δόθηκε και ένα σύντομο σενάριο χρήσης της Caley σε εφαρμογή.

7. Βιβλιογραφία

Βιβλία

- Ian Robinson, Jim Webber, and Emil Eifrem, Graph Databases, Chapter 1,2 &5, O'Reilly Media

Άρθρα

- Dan Ariely, An Architecture for Massively Scalable, Intelligent Applications, Duke University, May 2016
- Benjamin Erb, Concurrent Programming for Scalable Web Architecture Graph Databases
- Mihail, Google Cayley graph database tutorial - family tree, June 2016
- Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). A Comparison of a Graph Database and a Relational. ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference (ς. 42). New York: ACM New York, NY, USA ©2010

Ιστότοποι

- <https://el.wikipedia.org/wiki/ACID>
- <http://www.linuxinsider.gr/forum/8873/eisagogi-stis-nosql-baseis-dedomenon-xrisi-toy-cassandra>
- https://en.wikipedia.org/wiki/Column-oriented_DBMS
- <https://db-engines.com/en/article/Key-value+Stores>
- https://en.wikipedia.org/wiki/Document-oriented_database
- <https://neo4j.com/developer/graph-database/>
- <https://neo4j.com/developer/graph-db-vs-rdbms/>
- <https://neo4j.com/developer/graph-db-vs-rdbms/>
- <https://en.wikipedia.org/wiki/Neo4j>
- <https://franz.com/agraph/support/documentation/current/agraph-introduction.html>
- <https://www.oracle.com/database/spatial/index.html>
- <http://www.objectivity.com/products/infinitegraph/>

- <https://opensource.googleblog.com/2014/06/cayley-graphs-in-go.html>
- <http://www.i-programmer.info/news/84-database/7492-google-cayley-graph-database.html>
- <https://aws.amazon.com/blogs/big-data/building-a-graph-database-on-aws-using-amazon-dynamodb-and-titan/>
- <http://gremlindocs.spmallette.documentup.com/>
- <https://github.com/tinkerpop/gremlin/wiki/Defining-a-Property-Graph>
- <https://github.com/tinkerpop/gremlin/wiki/The-Benefits-of-Gremlin>

Παρουσιάσεις

- Michael Povolotsky, Virtuoso
- Jarek Wilkiewicz, Shawn Simister, When 2 billion Freebase facts is not enough, SemTechBiz '14 LODLAM Workshop