Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Συγκομιδή Δεδομένων Παγκόσμιου Ιστού για Επιστημονικά Δημοσιεύματα σε NoSQL Βάσεις Δεδομένων

Διπλωματική Εργασία

ΑΠΟΣΤΟΛΟΣ ΦΩΤΟΠΟΥΛΟΣ

Επιβλέπων
Βασιλακόπουλος Μιχαήλ
Αναπληρωτής Καθηγητής

Βόλος, Ιούλιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Συγκομιδή Δεδομένων Παγκόσμιου Ιστού για Επιστημονικά Δημοσιεύματα σε NoSQL Βάσεις Δεδομένων

Διπλωματική Εργασία

ΑΠΟΣΤΟΛΟΣ ΦΩΤΟΠΟΥΛΟΣ

Επιτροπή επίβλεψης

Επιβλέπων	Συνεπιβλέπουσα	Συνεπιβλέπουσα
Βασιλακόπουλος Μιχαήλ	Τουσίδου Ελένη	Τσαλαπάτα Χαρίκλεια
Αναπληρωτής Καθηγητής	μέλος ΕΔΙΠ	μέλος ΕΔΙΠ

Βόλος, Ιούλιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ο/Η συγγραφέας αυτής της εργασίας βεβαιώνει ότι κάθε βοήθεια την οποία είχε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης βεβαιώνει ότι έχει αναφέρει τις όποιες πηγές από τις οποίες έκανε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται επακριβώς, είτε παραφρασμένες.

University of Thessaly
Faculty of Engineering
Department of Electrical & Computer Engineering

Web Scarping of Scientific Publications' Data to NoSQL Databases

Diploma Thesis

APOSTOLOS FOTOPOULOS

Supervisor
Michael Vassilakopoulos
Associate Professor

Volos, July 2019

Περίληψη

Ο όγκος των δημοσιεύσεων που σχετίζονται με την επιστήμη των υπολογιστών είναι υπερβολικά μεγάλος. Εκτός αυτού τεράστιος είναι και η ο ρυθμός αύξησης όλων αυτών των δημοσιεύσεων, ιδιαίτερα τα τελευταία χρόνια. Μάλιστα αυτός ο ρυθμός είναι λογικό να αυξάνεται συνεχώς με την αύξηση του πληθυσμού και την επέκταση των διάφορων κλάδων της επιστήμης των υπολογιστών. Οπότε πολύ σημαντική κρίνεται η οργάνωση όλων αυτών των δεδομένων που αφορούν τις δημοσιεύσεις (συγγραφείς, εκδότης, χρονολογία έκδοσης κ.λπ.) και η επεξεργασία τους για την εξαγωγή πολύτιμων αποτελεσμάτων.

Αυτό αποτελεί και το Θέμα της συγκεκριμένης διπλωματικής εργασίας. Δηλαδή, η συγκομιδή δεδομένων που αφορούν την έκδοση δημοσιεύσεων, σχετικών με την επιστήμη των υπολογιστών και η επεξεργασία τους από ορισμένους αλγόριθμους με σκοπό την εξαγωγή διαφόρων πληροφοριών. Τα δεδομένα συλλέγονται από την διαδικτυακή βάση δεδομένων DBLP και αποθηκεύονται σε μία NoSQL βάση δεδομένων. Η βάση δεδομένων που χρησιμοποιείται από την συγκεκριμένη εφαρμογή είναι η Neo4j η οποία αποτελεί μία μη σχεσιακή βάση δεδομένων τύπου γράφου. Τα δεδομένα που εισάγονται στη Neo4j οργανώνονται σε μία συγκεκριμένη μορφή με σκοπό την εύκολη κατανόηση τους και την διευκόλυνση της διαχείρισής τους.

Σκοπός την εφαρμογής είναι η οργάνωση των δεδομένων και στη συνέχεια η απάντηση ερωτημάτων όπως ποιο είναι το πιο κατάλληλο περιοδικό για την δημοσίευση ενός άρθρου, ποιος ήταν ο πιο ενεργός συγγραφέας σε κάποιο συγκεκριμένο έτος ή ποιος είναι ο εκδότης που συνεργάζεται περισσότερο με ένα ορισμένο συγγραφέα και άλλων ερωτημάτων τέτοιου είδους. Τέλος να αναφερθεί πως η εφαρμογή έχει στηθεί με τη χρήση της Java γλώσσας προγραμματισμού.

Λέξεις Κλειδιά

Συγκομιδή δεδομένων, Ανάλυση δεδομένων, Επιστημονικά άρθρα, Συστήματα σύστασης, Συστάσεις για δημοσιεύσεις, dblp, Neo4j, NoSQL, Βάσεις δεδομένων γράφου, Cypher, Java

Abstract

The amount of publications related to computer science is extremely large. In addition, the growth rate of the number of publications is enormous, especially in recent years. In fact, it's logical for this rate to increase steadily as the population grows and the various branches of computer science expand. Therefore, it is very important to organize all this data regarding the publications (authors, publisher, date of publication, etc.) and to process them for the extraction of valuable results.

The subject of this diploma thesis is the collection of data that are relevant with scientific publications about computer science and their processing by certain algorithms for the purpose of extracting various information. The data are collected by DBLP, an online database and are stored in a NoSQL database. The database used by this application is Neo4j, which is a non-relational graph database. The data entered in Neo4j is organized in a specific format so that they can easily be understood and managed.

The purpose of the application is to organise all these data and then to answer questions such as which is the most appropriate magazine for the publication of an article, who was the most active author in a particular year or who is the publisher who collaborates most with a certain author and others similar to these. Lastly, it should be mentioned that the application was written in Java programming language.

Keywords

Data collection, Data analysis, Scientific articles, Recommender systems, Recommendations for publishing, dblp, Neo4j, NoSQL, Graph databases, Cypher, Java

Στην οικογένεια μου.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την στήριξη που μου προσέφεραν όλα αυτά τα χρόνια. Η βοήθειά τους ήταν αυτή που μου έδωσε τη δυνατότητα να ολοκληρώσω επιτυχώς τις σπουδές μου. Στη συνέχεια θα ήθελα να ευχαριστήσω τους φίλους μου Σωτήρη Ευαγγέλου και Λευτέρη Χαζτηευφραιμίδη για τη συνεργασία και την αλληλοβοήθεια σε πολλά θέματα των σπουδών μου όλα αυτά τα χρόνια. Τέλος ευχαριστώ ιδιαίτερα και τον κύριο Βασιλακόπουλο Μιχαήλ για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω.

Πρόλογος

Αφορμή αυτής της διπλωματική εργασίας αποτελεί η ευρεία ενασχόληση μου με σχεσιακές και μή σχεσιακές βάσεις τα προηγούμενα χρόνια αλλά και η θέληση μου να διευρύνω περαιτέρω τις γνώσεις μου πάνω σε αυτόν τον τομέα. Οι δημοσιεύσεις ερευνών διατελούν σημαντικό ρόλο στην πρόοδο της έρευνας και της τεχνολογίας για αυτό και η εξερεύνηση αυτών αποτελεί βασικό θέμα αυτής της εργσίας. Η παρούσα διπλωματική εργασία εκπονήθηκε ως το τελευταίο βήμα της ακαδημαικής μου διαδρομής με σκοπό την απόκτηση του διπλώματος του τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας στην πόλη του Βόλου υπό την επίβλεψη του καθηγητή Μιχαήλ Βασιλακόπουλου. Η εργασία πραγματοποιήθηκε από τον Ιανουάριο μέχρι τον Ιούνιο του 2020 κατά τη διάρκεια του 10ου εξαμήνου των πανεπιστημιακών μου σπουδών.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

Τη σήμερον ημέρα ο αριθμός των επιστημονικών δημοσιεύσεων είναι απλά τεράστιος. Σύμφωνα με έρευνα του Πανεπιστημίου της Οτάβα, το 2009 ξεπεράσαμε το όριο των 50 εκατομμυρίων επιστημονικών εργασιών που δημοσιεύθηκαν μετά από το 1665 [14]. Αυτός ο αριθμός αυξάνεται με ραγδαία ταχύτητα, αφού υπολογίζεται πως από το 2010 και μετά έχουμε 2,5 με 3 εκατομμύρια επιστημονικές δημοσιεύσεις το χρόνο [15, 22].

Αυτή η μεγάλη κλιμάκωση στην παραγωγή επιστημονικών δημοσιεύσεων οφείλεται σε ποικίλους λόγους. Το 2014 υπήρχαν περίπου 28.100 ενεργά επιστημονικά περιοδικά [15]. Αν προσθέσουμε σε αυτόν, τον αυξανόμενο αριθμό χαμηλότερου επιπέδου και μη επίσημων επιστημονικών περιοδικών, τότε το σύνολο των επιστημονικών εκδόσεων αυξάνεται δραματικά. Ένας ακόμη βασικός παράγοντας είναι και ο τεράστιος αριθμός των επιστημονικών δημοσιεύσεων παγκοσμίως, ο οποίος αυξάνεται με ρυθμό περίπου 4-5% ετησίως [15].

Έτσι έχοντας ως σκοπό την αποθήκευση όλων αυτών των δεδομένων, που σχετίζονται με τις επιστημονικές δημοσιεύσεις, δημιουργήθηκαν πολλές διαδικτυακές βάσεις δεδομένων. Μία από αυτές είναι και η DBLP (Digital Bibliography & Library Project) η οποία περιέχει ένα μεγάλο αριθμό από δημοσιεύσεις που σχετίζονται με την επιστήμη των υπολογιστών.

1.1 Αντικείμενο της διπλωματικής

Ο σκοπός της εφαρμογής που παρουσιάζει αυτή η διπλωματική εργασία είναι η συγκομιδή ενός μεγάλου αριθμού δεδομένων από τη διαδικτυακή βάση DBLP, η οργάνωση τους στη NoSQL βάση Neo4j και στη συνέχεια η επεξεργασία όλων αυτών των δεδομένων με στόχο την απάντηση συγκεκριμένων ερωτημάτων. Πιο αναλυτικά, η DBLP προσφέρει όλα τα δεδομένα της σε ένα αρχείο xml το οποίο ο καθένας είναι ελεύθερος να το κατεβάσει και να το διαχειριστεί. Το πρώτο κομμάτι της εφαρμογής, που αφορά την εισαγωγή δεδομένων στη Neo4j, χρησιμοποιεί αυτό το xml και με τη χρήση ενός κατάλληλου αλγορίθμου δημιουργεί τους αντίστοιχους κόμβους με τις ιδιότητες που τους χαρακτηρίζουν καθώς και τις σχέσεις μεταξύ τους. Στη συνέχεια, εφόσον έχει δημιουργηθεί η βάση δεδομένων λαμβάνει μέρος το δεύτερο κομμάτι της εφαρμογής που αφορά την απάντηση ορισμένων ερωτημάτων με βάση τα δεδομένα που υπάρχουν στη Neo4j και τις σχέσεις που τα χαρακτηρίζουν. Αυτά τα ερωτήματα είναι αρκετά και διαφέρουν μεταξύ

τους. Αναλυτικότερα, μπορούμε να δούμε τους πιο ενεργούς συγγραφείς/εκδότες/περιοδικά ανάλογα με τον αριθμό των δημοσιεύσεων με τις οποίες συσχετίζονται, τις λέξεις-κλειδιά που χρησιμοποιούνται περισσότερο από την επιστημονική κοινότητα, όλες τις δημοσιεύσεις που περιέχουν μία λέξη-κλειδί, τα δεδομένα μιας συγκεκριμένης δημοσίευσης, τις δημοσιεύσεις στις οποίες έχει συμμετέχει ένας συγκεκριμένος συγγραφέας, τους πιο κοντινούς συγγραφείς/εκδότες/περιοδικά σε ένα συγγραφέα, τις λέξεις-κλειδιά που χρησιμοποιεί περισσότερο ένας συγγραφέας και τέλος τις δημοσιέυσεις/περιοδικά/εκδότες/συγγραφείς που σχετίζονται περισσότερο με ένα σύνολο από λέξεις-κλειδιά κάτι που αποτελεί πολύ χρήσιμη πληροφορία εάν θέλουμε να δημοσιεύσουμε μια εργασία και θέλουμε να βρούμε το κατάλληλο περιοδικό/εκδότη ή κάποιους συγγραφείς που να σχετίζονται με το θέμα της εργασίας μας.

1.2 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν παρόμοιες εργασίες συγκομιδής και επεξεργασίας δεδομένων που σχετίζονται με επιστημονικά δημοσιεύματα.
2. Μελετήθηκε η DBLP και η προσφορά της στην παροχή πληροφοριών για ένα μεγάλο αριθμό επιστημονικών δημοσιεύσεων.
3. Μελετήθηκαν διαδικτυακές βάσεις δεδομένων παρόμοιες με την DBLP και αξιολογήθηκε η χρησιμότητα που θα μπορούσε να είχαν σε μία εφαρμογή τέτοιου είδους.
4. Μελετήθηκε η Neo4j και η συνεισφορά της στην οργάνωση και διαχείριση ενός μεγάλου αριθμού δεδομένων.
5. Αναπτύχθηκε σχεσιακό μοντέλο οργάνωσης όλων των δεδομένων που εξάγονται από την DBLP.
6. Δημιουργήθηκε αλγόριθμος εξαγωγής δεδομένων από το xml που παρέχει η DBLP και εισαγωγής αυτών στη Neo4j βάση δεδομένων.
7. Σχεδιάστηκαν αλγόριθμοι με τους οποίους μπορούν να πραγματοποιηθούν όλα τα ερωτήματα που αναφέρθηκαν παραπάνω.

1.3 Οργάνωση του τόμου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2. Στο Κεφάλαιο 3 αναλύουμε την DBLP, ενώ στο Κεφάλαιο 4 αναλύουμε την Neo4j. Στο Κεφάλαιο 5 αναπτύσσουμε την λογική της εφαρμογής και περιγράφουμε τη λειτουργία της. Στο Κεφάλαιο 6 παρουσιάζεται ένα παράδειγμα εκτέλεσης της εφαρμογής με σκοπό την καλύτερη κατανόηση της, και στη συνέχεια στο Κεφάλαιο 7 παρέχονται ορισμένες οδηγίες για την εγκατάστασή της. Τέλος στο Κεφάλαιο 8 αναφέρονται τα συμπεράσματα και τα τελικά αποτελέσματα της συνολικής προσπάθειας και παρατίθενται ιδέες μελλοντικής επέκτασης της εργασίας.

Κεφάλαιο 2

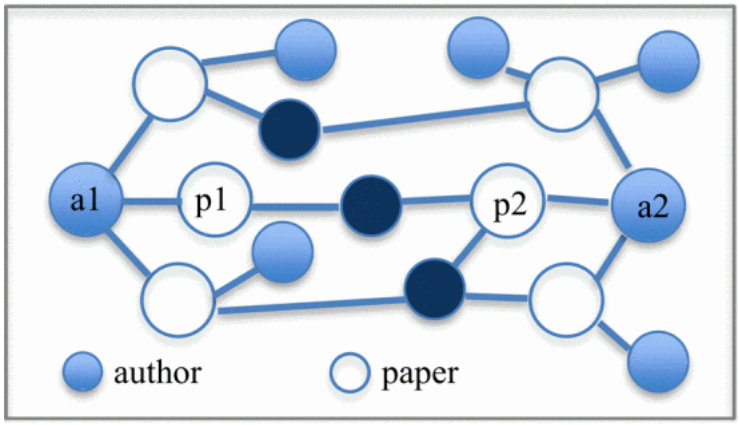
Συγγενικές εργασίες

Αρχικά ας αναφερθούμε σε ορισμένες συγγενικές εργασίες οι οποίες έχουν πραγματοποιηθεί. Όλες τους έχουν ως κοινό στοιχείο τη χρήση των δεδομένων της DBLP, την οργάνωσή τους σε μια μορφή τύπου γράφου και την επεξεργασία τους με κάποιο αλγόριθμο με σκοπό την απάντηση συγκεκριμένων ερωτημάτων, όμοια με την εφαρμογή αυτής της διπλωματικής. Οι θεματολογίες τους αφορούν την αποσαφήνιση συγγραφέων [24] που μοιράζονται κοινό όνομα, βάσει γραφήματος, την ανάλυση των δικτύων από επιστήμονες σχετικούς με την επιστήμη των υπολογιστών και τέλος την συσχέτιση συγγραφέων σύμφωνα με τις δημοσιεύσεις στις οποίες συμμετείχαν μαζί.

2.1 Αποσαφήνιση συγγραφέων βάσει γραφήματος

Σκοπός αυτής της εργασίας [12], είναι η παρουσίαση μιας νέα μεθόδου για την αποσαφήνιση των συγγραφέων σε βιβλιογραφικά δίκτυα. Μια τέτοια μέθοδος βασίζεται σε διασταύρωση μονοπατιών βάθους 2 (σχ. 2.1) χρησιμοποιώντας τοπολογικά μέτρα ομοιότητας για βαθμολόγηση των υποψήφιων κόμβων. Η τοπολογική ομοιότητα χρησιμοποιείται ευρέως στον τομέα της εφαρμογής για να εκτιμηθεί η πιθανότητα ενός άγνωστου συνδέσμου. Μια συνάρτηση ομοιότητας μπορεί να είναι μια καλή προσέγγιση για την ισότητα, επομένως μπορεί να χρησιμοποιηθεί για την αποσαφήνιση, με βάση την υπόθεση ότι οι συγγραφείς με πολλούς κοινούς συν-συγγραφείς είναι παρόμοιοι. Η αποθήκευση των δεδομένων που χρησιμοποιεί η εφαρμογή γίνεται στη βάση Neo4j. Ουσιαστικά η εφαρμογή προσπαθεί διαχωρίσει όλους τους συγγραφείς που μοιράζονται κοινό όνομα και να τους αντιστοιχίσει στις κατάλληλες δημοσιεύσεις.

Χαρακτηριστικό παράδειγμα αποτελεί η ονομασία ”Chen Li” που αποτελεί ένα εξαιρετικά ασαφές όνομα καθώς είναι από τα πιο συχνά ονόματα στην Κίνα. Το όνομα αυτό αναφέρεται 72295 φορές στην DBLP. Στην DBLP υπάρχουν 24 αναφερόμενοι συγγραφείς με το όνομα ”Chen Li”, με 23 από αυτούς να έχουν κάποιο ιδιαίτερο χαρακτηριστικό για να ξεχωρίζουν από τους υπόλοιπους και τον τελευταίο να είναι ένας μεγάλος αριθμός από ασαφείς συγγραφείς ονομαζόμενους ”Chen Li” (σχ. 2.2).

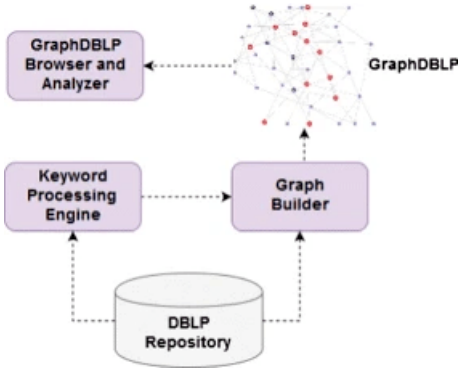


Σχήμα 2.1: Διασταύρωση μονοπατιών βάθους 2 [12]

Σχήμα 2.2: Παράδειγμα κοινών ονομασιών στην DBLP [12]

2.2 Ανάλυση δικτύων επιστημόνων

Αυτό το άρθρο [17] παρουσιάζει το GraphDBLP, ένα σύστημα που μοντελοποιεί τη βιβλιογραφία DBLP ως μία βάση δεδομένων γραφημάτων μέσω της Neo4j για την εκτέλεση ερωτημάτων και την ανάλυση κοινωνικών δικτύων. Το GraphDBLP εμπλουτίζει επίσης τα δεδομένα DBLP μέσω σημασιολογικών ομοιοτήτων λέξεων-κλειδιών που υπολογίζονται μέσω ενός κατάλληλου αλγορίθμου. Παρέχει ένα τρόπο υλοποίησης για τρία ουσιαστικά ερωτήματα που αφορούν την εξερεύνηση της κοινότητας DBLP. Αυτά είναι η διερεύνηση προφίλ συγγραφέα αναλύοντας τα αρχεία δημοσίευσής τους, ο προσδιορισμός των πιο παραγωγικών συγγραφέων σε ένα δεδομένο θέμα και η πραγματοποίηση ανάλυσης κοινωνικού δικτύου σε ολόκληρη την κοινότητα. Κάτι παρόμοιο αποτελεί θέμα και αυτής της διπλωματικής αφού έχει γίνει υλοποίηση αλγορίθμου που εξάγει τις βασικές λέξεις κλειδιά που σχετίζονται με ένα συγγραφέα στην ενότητα 5.3.8 καθώς και τους πιο σχετικούς συγγραφείς/εκδότες/περιοδικά με ένα σύνολο από λέξεις κλειδιά στην ενότητα 5.3.9.



Σχήμα 2.3: Η GraphDBLP εφαρμογή [17]

2.3 Συσχέτιση συγγραφέων

Σε αυτό το ερευνητικό έγγραφο [3] παρουσιάζονται τα αποτελέσματα οπτικοποίησης και ανάλυσης του δικτύου συν-συγγραφέων της DBLP με τη βοήθεια του εργαλείου Gephi. Χρησιμοποιείται το σύνολο δεδομένων από την DBLP σε μορφή μη κατευθυνόμενου γραφήματος. Σε αυτό το γράφημα, οι κόμβοι ανήκουν στην οντότητα ”Συγγραφέας”. Οι κόμβοι είναι διασυνδεμένοι μεταξύ τους ανάλογα με τον συσχετισμό τους στο σύνολο δεδομένων των ερευνητικών εργασιών. Υπάρχουν περισσότερες από μία ακμές μεταξύ δύο κόμβων εάν οι δύο ερευνητές συνέγραψαν μαζί περισσότερα από ένα άρθρα. Βασικός σκοπός της εφαρμογής είναι η κατανόηση της ακαδημαϊκής συνεργασίας και η εύρεση των ερευνητικών κοινοτήτων για συγκεκριμένο τομέα γνώσης. Ένα παρόμοιο ερώτημα εξετάζεται και σε αυτή τη διπλωματική στην ενότητα 5.3.7, το οποίο έχει ως σκοπό να βρει τους πιο κοντινούς συγγραφείς ενός συγκεκριμένου συγγραφέα αναλύοντας τις εργασίες στις οποίες έχουν δουλέψει μαζί.

Το εργαλείο Gephi [5] αποτελεί μια διαδραστική πλατφόρμα οπτικοποίησης και εξερεύνησης πολλών μορφής γραφημάτων. Ουσιαστικά η οργάνωση των δεδομένων είναι αρκετά παρόμοια με αυτή στη Neo4j. Μάλιστα υποστηρίζεται η δυνατότητα διαχείρισης γραφημάτων της Neo4j από το Gephi μέσω ενός plug-in που μπορεί να εγκατασταθεί [16].

Κεφάλαιο 3

Ανάλυση της DBLP

Σε αυτό το κεφάλαιο θα μελετήσουμε την DBLP [8] (Digital Bibliography & Library Project) η οποία αποτελεί την πηγή δεδομένων της συγκεκριμένης εφαρμογής. Θα αναφέρουμε ορισμένα ιστορικά και εννοιολογικά στοιχεία και θα δούμε τον τρόπο με τον οποίο τα δεδομένα είναι δομημένα. Έπειτα θα δούμε το πως μπορούμε να εξάγουμε όλα αυτά τα δεδομένα και τέλος θα εξετάσουμε παρόμοιες διαδικτυακές βάσεις δεδομένων, που σχετίζονται με δημοσιεύσεις με θέμα την επιστήμη των υπολογιστών και θα αξιολογήσουμε την χρησιμότητά τους σε μία τέτοιου είδους εφαρμογή.

3.1 Γενικές πληροφορίες

Η διαδικτυακή βάση δεδομένων DBLP αποτελεί μία ηλεκτρονική αναφορά σε βιβλιογραφικές πληροφορίες για εκδόσεις σχετικές με την επιστήμη των υπολογιστών. Έχει εξελιχθεί από έναν πρώιμο μικρό πειραματικό διακομιστή ιστού σε μια δημοφιλή υπηρεσία ανοιχτών δεδομένων για ολόκληρη την κοινότητα της επιστήμης των υπολογιστών. Η αποστολή της DBLP είναι η υποστήριξη των ερευνητών, της επιστήμη των υπολογιστών, στις καθημερινές τους προσπάθειες παρέχοντας δωρεάν πρόσβαση σε υψηλής ποιότητας βιβλιογραφικά δεδομένα και συνδέσμους προς τις ηλεκτρονικές εκδόσεις των δημοσιεύσεων.

Αρχικά, η υπηρεσία που παρέχεται από την DBLP ξεκίνησε στην ερευνητική ομάδα ”Database Systems and Logic Programming” (DBLP) στο Πανεπιστήμιο του Trier της Γερμανίας και επικεντρώθηκε σε δημοσιεύσεις από αυτόν τον τομέα έρευνας. Με το πέρασμα των χρόνων, η DBLP επεκτάθηκε σταδιακά σε όλους τους τομείς της επιστήμης των υπολογιστών, ενώ το ακρωνύμιο επέζησε. Μερικές φορές, η ετικέτα ”Digital Bibliography & Library Project” έχει υιοθετηθεί ως βασικό όνομα για την DBLP, αλλά έχει χάσει τη σημασία της. Πιο αποδεκτή πλέον θα ήταν η ονομασία ”DBLP computer science bibliography”.

Η DBLP έχει αναπτυχθεί από έναν διακομιστή μικρής κλίμακας που προορίζονταν για τη δοκιμή τεχνολογιών Ιστού και την εξυπηρέτηση μόνο μιας μικρής τοπικής κοινότητας σε έναν ιστότοπο που χρησιμοποιείται από χιλιάδες ανθρώπους παγκοσμίως. Ωστόσο, το λογισμικό που χρησιμοποιείται εξακολουθεί να παραμένει μικρό σε όγκο. Δεν υπάρχει κάποιο σύστημα διαχείρισης βάσεων δεδομένων πίσω από την DBLP (π.χ. PostgreSQL ή MySQL). Οι πληροφορίες αποθηκεύ-

ονται σε αρκετά εκατομμύρια αρχεία στο σύστημα αρχείων. Τα προγράμματα που χρησιμοποιούνται για τη συντήρηση της DBLP έχουν γραφτεί σε C, Perl και Java, τα οποία ενώνονται μεταξύ τους με κώδικα γραμμένο σε shell. Το σύστημα στο οποίο τρέχει η DBLP αποτελείται από ένα μικρό αριθμό μηχανημάτων UNIX (Ubuntu Linux).

3.2 Τα δεδομένα της DBLP

Η DBLP καλύπτει δημοσιεύσεις μόνο από την επιστήμη υπολογιστών. Εστιάζει κυρίως σε διεθνείς εκδόσεις, δηλαδή σε δημοσιεύσεις στα Αγγλικά, αν και υπάρχουν εξαιρέσεις σε αυτόν τον κανόνα. Επίσης, δεδομένου ότι δεν υπάρχουν οι πόροι για την κάλυψη όλων των συγκεντρώσεων, στις οποίες δημοσιεύονται εργασίες, η DBLP προσπαθεί να δώσει προτεραιότητα στην ένταξη των συγκεντρώσεων με βάση την επιστημονική τους αξία καθώς και τη σημασία τους για την κοινότητα της επιστήμης των υπολογιστών. Όσον αφορά τα επιστημονικά πεδία που καλύπτονται από την DBLP είναι πολλά και αποτελούνται και από ένα δίκαιο μερίδιο υβριδικών πεδίων εφόσον ενδιαφέρουν σημαντικά την ερευνητική κοινότητα της επιστήμης των υπολογιστών. Μερικά από τα πεδία που καλύπτονται είναι: αλγόριθμοι, τεχνητή νοημοσύνη, κρυπτολογία, βάσεις δεδομένων, μηχανική μάθηση, επεξεργασία σημάτων, ρομποτική, παγκόσμιος ιστός, κατανεμημένα συστήματα κ.ο.κ.

Τα δεδομένα της DBLP περνούν εκτενείς ελέγχους με σκοπό τη διασφάλιση της εγκυρότητάς και της ποιότητας που τα χαρακτηρίζουν. Οι πίνακες περιεχομένων των πλήρων εργασιών ή των τόμων περιοδικών εξετάζονται μαζικά. Τα απαραίτητα δεδομένα για κάθε δημοσίευση λαμβάνονται απευθείας από τον εκδότη ενός τόμου ή τον διοργανωτή μιας εκδήλωσης. Σε πολύ μικρό αριθμό περιπτώσεων, τα δεδομένα υποβάλλονται από εθελοντές της κοινότητας. Μόλις ληφθούν τα δεδομένα, εφαρμόζεται μια αυστηρή διαδικασία καθαρισμού δεδομένων από έναν υπεύθυνο της ομάδας της DBLP. Αυτή η διαδικασία υποστηρίζεται από μερικούς απλούς αλγόριθμους που ελέγχουν τη συνοχή των δεδομένων, αλλά εκτελείται κυρίως με το χέρι.

Αυτή η διαδικασία χειροκίνητης επιμέλειας έχει τέσσερις βασικούς στόχους:

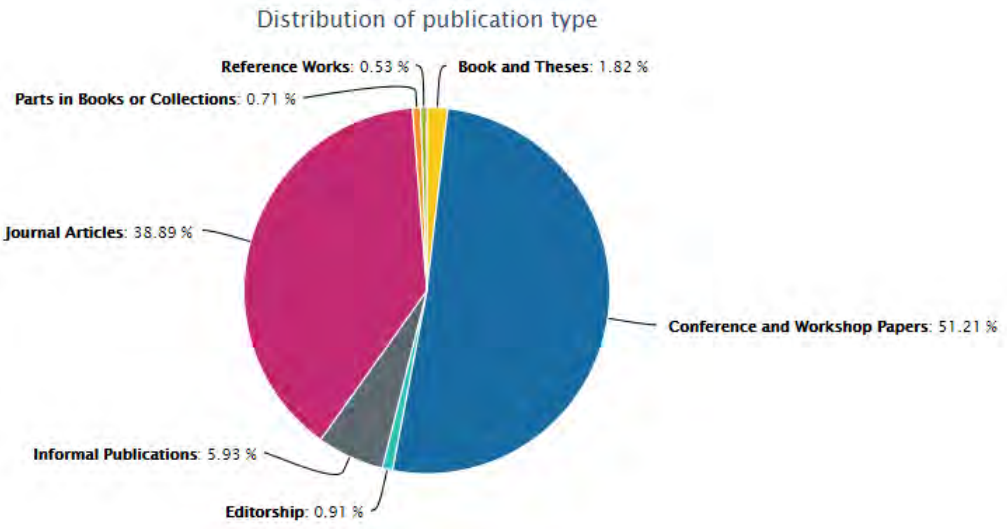
1. Αποκάλυψη τυχών τυπογραφικών λαθών, ορθογραφικών λαθών, αντικείμενων κωδικοποίησης χαρακτήρων κ.λπ. στα ληφθέντα δεδομένα.

2. Ολοκλήρωση των μετα-δεδομένων που λείπουν, όπως επέκταση ελλιπών ονομάτων συγγραφέων (π.χ. συντομευμένα ονόματα ή τμήματα ονομάτων που λείπουν) στην πλήρη φόρμα τους.

3. Αποσαφήνιση ομώνυμων συγγραφέων και ενοποίηση συνώνυμων ονομάτων συγγραφέων.

4. Διόρθωση σφαλμάτων και ολοκλήρωση των στοιχείων που λείπουν που μπορεί να έχουν γίνει εμφανή στο υπάρχον απόθεμα δεδομένων υπό την ένταξη των πρόσφατα διαθέσιμων δεδομένων.

Μόνο αφού εφαρμοστεί ένα πλήρες πέρασμα καθαρισμού δεδομένων στα εισαχθέντα δεδομένα, οι νέες εγγραφές προστίθενται στο σύνολο δεδομένων της DBLP. Ωστόσο, η διαδικασία κα-



Σχήμα 3.1: Διανομή δημοσιεύσεων στην DBLP [7]

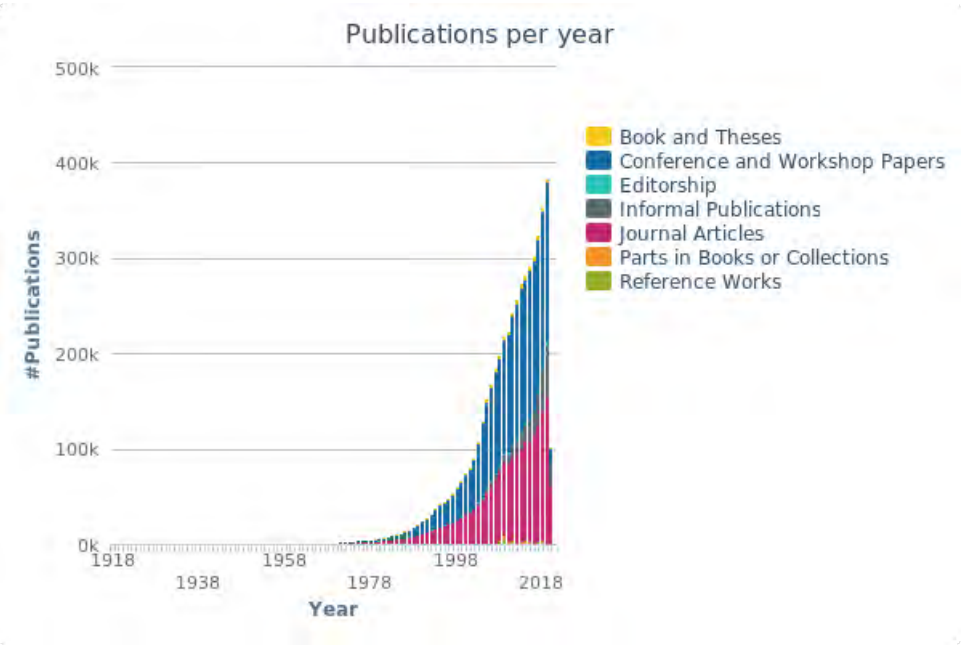
θαρισμού δεδομένων δεν τελειώνει εδώ. Σε μια επαναληπτική διαδικασία, για τις επόμενες ημέρες, τα δεδομένα που προστέθηκαν πρόσφατα παρακολουθούνται από ειδικά προγράμματα για τυχόν ύποπτα σημάδια ασυνέπειας των δεδομένων. Για παράδειγμα, παρατηρείται συχνά ένα συγκεκριμένο φαινόμενο κυματισμού. Την πρώτη ημέρα, μια νέα έκδοση που προστέθηκε βοηθά στην αποκάλυψη παλαιότερων μη αναγνωρισμένων ομώνυμων δεδομένων. Χάρη στη διόρθωση αυτών των πληροφοριών, την επόμενη μέρα, γίνονται εμφανείς περαιτέρω ασυνέπειες σε περαιτέρω αρχεία κ.ο.κ. Ένα τέτοιο επαναληπτικό αποτέλεσμα μπορεί να διαρκέσει αρκετές ημέρες.

Μετρήσεις που έγιναν τον Ιανουάριο του 2019 [6] δείχνουν πως η DBLP περιέχει πάνω από 4,4 εκατομμύρια δημοσιεύσεις, που δημοσιεύθηκαν από περισσότερους από 2,2 εκατομμύρια συγγραφείς. Επιπλέον, η DBLP κατέχει περίπου 40.000 τόμους περιοδικών, περισσότερες από 39.000 εργασίες συνεδρίων και εργαστηρίων και περισσότερες από 80.000 μονογραφίες. Βασικά χαρακτηριστικά των δεδομένων της DBLP φαίνονται στα παρακάτω σχήματα. Στο σχήμα 3.1 μπορούμε να δούμε την διανομή των διάφορων τύπων δημοσιεύσεων, ενώ στο σχήμα 3.2 μπορούμε να δούμε τον αριθμό των δημοσιεύσεων ενός τύπου ανά έτος.

3.3 Οργάνωση των δεδομένων της DBLP

Οποιοσδήποτε μπορεί να έχει πρόσβαση στα δεδομένα της DBLP μέσω ενός αρχείου xml που περιέχει όλο το σύνολο των δεδομένων. Μάλιστα τα δεδομένα που παρέχει η DBLP είναι ελεύθερα προς αντιγραφή, διανομή, τροποποίηση, μεταμόρφωση, και ο καθένας είναι ελεύθερος να δημιουργήσει και να παράγει έργα από αυτά, ακόμη και για εμπορικούς σκοπούς, όλα χωρίς την ανάγκη να ζητήσει κάποια άδεια [9].

Η μορφή DBLP XML διαμορφώνεται σύμφωνα με τη μορφή αρχείου BibTeX * .bib. Η μορφή ορίζεται στο αρχείο DTD το οποίο υπάρχει στον ίδιο κατάλογο. το αρχείο DTD δεν είναι πολύ αυστηρό, καθώς δεν περιορίζει τη σειρά στοιχείων ή την πολλαπλότητα, σκοπός του είναι η δια-



Σχήμα 3.2: Δημοσιεύσεις ανά έτος στην DBLP [10]

τήρηση του ορισμού και η απλοποίηση της κατανόησης των διάφορων στοιχείων του xml. Οι τύποι βιβλιογραφιών που περιέχει το xml είναι οι εξής: article, inproceedings, proceedings, book, incollection, phdthesis, mastersthesis. Όλες οι εγγραφές χαρακτηρίζονται από δύο γνωρίσματα το key και το mdate που αποτελούν ένα μοναδικό κλειδί που ανήκει στο έγγραφο και την τελευταία φορά που το έγγραφο τροποποιήθηκε από την ομάδα της DBLP. Ορισμένα στοιχεία περιέχουν και ένα γνώρισμα που ονομάζεται publtype το οποίο περιέχει ορισμένες επιπλέον πληροφορίες που το χαρακτηρίζουν. Τέτοιες πληροφορίες π.χ. μπορεί να είναι αν πρόκειται για μία ανεπίσημη δημοσίευση με το γνώρισμα ”informal” ή αν πρόκειται για μια δημοσίευση που παρουσιάζει κάποιο λογισμικό με το γνώρισμα ”software”.

Εκτός από γνωρίσματα όλοι οι τύποι βιβλιογραφιών χαρακτηρίζονται και από ορισμένα στοιχεία που τα περιγράφουν όπως οι συγγραφείς, το περιοδικό, ο εκδότης, ο χρόνος δημοσίευσης της εργασίας κ.ο.κ. Να επισημανθεί πως όταν 2 η περισσότεροι συγγραφείς έχουν το ίδιο όνομα ξεχωρίζουν μεταξύ τους με ένα σύνολο αριθμών (Jun Sun, Jun Sun 0001 κ.ο.κ.). Οι εκδότες και τα περιοδικά έχουν μοναδικά ονόματα. Παρακάτω θα αναλύσουμε περαιτέρω όλα αυτά στοιχεία για κάθε διαφορετικό τύπο δημοσίευσης.

3.3.1 Articles

```
<article mdate="2020-02-24" key="journals/jet/KunimotoS04">
    <author>Takashi Kunimoto</author>
    <author>Roberto Serrano</author>
    <title>Bargaining and competition revisited.</title>
    <pages>78-88</pages>
    <month>March</month>
    <year>2004</year>
```

```
    <volume>115</volume>
    <journal>J. Econ. Theory</journal>
    <number>1</number>
    <ee>https://doi.org/10.1016/S0022-0531(03)00131-5</ee>
    <url>db/journals/jet/jet115.html#KunimotoS04</url>
</article>
```

Πρόκειται για άρθρα που έχουν δημοσιευθεί σε κάποιο περιοδικό. Κάθε άρθρο είναι αναγκαίο να περιέχει τους συγγραφείς, τον τίτλο, τον χρόνο έκδοσής του καθώς και το περιοδικό στο οποίο εκδόθηκε. Όλα τα άλλα στοιχεία είναι προαιρετικά. Το πεδίο ”ee” πρόκειται για το σύνδεσμο του άρθρου στον παγκόσμιο ιστό. Το πεδίο ”url” αποτελεί το σύνδεσμο του άρθρου εσωτερικά της DBLP. Ορισμένα άρθρα μπορεί να περιέχουν και μερικά επιπλέον πεδία που δεν φαίνονται παραπάνω όπως το πεδίο ”cdrom” αν το άρθρο υπάρχει σε κάποιο δίσκο η το πεδίο ”note” για επιπρόσθετες πληροφορίες που περιγράφουν το άρθρο. Παρ’ όλα αυτά τέτοιου είδους πεδία συναντιούνται πολύ σπάνια.

3.3.2 Inproceedings

```
<inproceedings mdate="2017-05-20" key="journals/lncs/BLP91">
    <author>Toni Bollinger</author>
    <author>Sven Lorenz</author>
    <author>Udo Pletat</author>
    <title>The LILOG Inference Engine.</title>
    <pages>402-427</pages>
    <year>1991</year>
    <crossref>journals/lncs/1991-546</crossref>
    <booktitle>Text Understanding in LILOG</booktitle>
    <url>db/journals/lncs/lncs546.html#BollingerLP91</url>
    <ee>https://doi.org/10.1007/3-540-54594-8_72</ee>
</inproceedings>
```

Αφορά εργασίες οι οποίες έγιναν στα πλαίσια ενός συνεδρίου ή ενός σεμιναρίου. Απαραίτητα είναι τα πεδία που περιέχουν τον τίτλο της εργασίας, τους συγγραφείς, το χρόνο σύνταξης, τον εσωτερικό σύνδεσμο και τέλος τον τίτλο του τόμου (proceedings) στον οποίο ανήκει η εργασία καθώς και το κλειδί της που συμπεριλαμβάνεται στο πεδίο ”crossref”. Όλα τα υπόλοιπα πεδία δεν είναι αναγκαία.

3.3.3 Proceedings

```
<proceedings mdate="2020-03-27" key="reference/genetic/2015">
    <editor>Amir Hossein Gandomi</editor>
    <editor>Amir Hossein Alavi</editor>
    <editor>Conor Ryan</editor>
    <title>Handbook of Genetic Programming Applications</title>
    <publisher>Springer</publisher>
    <year>2015</year>
    <isbn>978-3-319-20882-4</isbn>
    <ee>https://doi.org/10.1007/978-3-319-20883-1</ee>
```

```
    <url>db/reference/genetic/genetic2015.html</url>
</proceedings>
```

Αφορά τους τόμους των εργασιών που πραγματοποιήθηκαν κατά τη διάρκεια ενός συνεδρίου ή ενός σεμιναρίου. Αναγκαία είναι τα πεδία που περιέχουν τον τίτλο του τόμου, τους συντάκτες, τον εκδότη και το isbn. Οποιοδήποτε άλλο πεδίο είναι προαιρετικό.

3.3.4 Books

```
<book mdate="2019-06-27" key="reference/crypt/2005">
    <editor>Henk C. A. van Tilborg</editor>
    <title>Encyclopedia of Cryptography and Security</title>
    <publisher>Springer</publisher>
    <year>2005</year>
    <isbn>978-0-387-23473-1</isbn>
    <ee>https://doi.org/10.1007/0-387-23483-7</ee>
    <url>db/reference/crypt/crypt2005.html</url>
</book>
```

Πρόκειται για μία μονογραφία ή μία επεξεργασμένη συλλογή άρθρων. Απαραίτητα είναι τα πεδία που περιέχουν τον τίτλο του βιβλίου, τους συντάκτες, τον εκδότη και το isbn. Οποιοδήποτε άλλο πεδίο είναι προαιρετικό. Συνηθισμένα επιπλέον πεδία σε αυτή την κατηγορία είναι το πεδίο ”series” που περιγράφει τη σειρά του βιβλίου καθώς και το πεδίο ”volume” που περιέχει τον τόμο του βιβλίου. Ορισμένες φορές συναντάμε και το πεδίο ”booktitle” το οποίο είτε περιέχει ξανά τον τίτλο του βιβλίου είτε μία γενίκευση αυτού.

3.3.5 Incollections

```
<incollection mdate="2017-05-16" key="series/acvpr/BakB14">
    <author>Slawomir Bak</author>
    <title>Re-identification by Covariance Descriptors.</title>
    <pages>71-91</pages>
    <year>2014</year>
    <booktitle>Person Re-Identification</booktitle>
    <ee>https://doi.org/10.1007/978-1-4471-6296-4_4</ee>
    <crossref>series/acvpr/978-1-4471-6295-7</crossref>
    <url>db/series/acvpr/reident2014.html#BakB14</url>
</incollection>
```

Αφορά έγγραφα που αποτελούν ένα κομμάτι ή ένα κεφάλαιο σε μονογραφία. Αναγκαία είναι τα πεδία που περιέχουν τον τίτλο της εργασίας, τους συγγραφείς, τον χρόνο στον οποίο πραγματοποιήθηκε, τον τίτλο της μονογραφίας στην οποία ανήκει καθώς και το χαρακτηριστικό κλειδί αυτής που βρίσκεται στο πεδίο ”crossref”. Όλα τα άλλα πεδία δεν είναι αναγκαία.

3.3.6 PhD theses

```
<phdthesis mdate="2016-10-05" key="books/daglib/0091856">
    <author>Christian Scheideler</author>
```

```
    <title>Universal routing strategies.</title>
    <pages>I-X, 1-202</pages>
    <year>1996</year>
    <school>University of Paderborn, Germany</school>
    <ee>http://d-nb.info/953115151</ee>
</phdthesis>
```

Πρόκειται για διδακτορικές διατριβές. Απαραίτητα είναι τα πεδία που περιέχουν τον τίτλο της εργασίας, τον συγγραφέα, τον χρόνο σύνταξης της εργασίας καθώς και το εκπαιδευτικό ίδρυμα όπου πραγματοποιήθηκε. Κάθε άλλο πεδίο είναι προαιρετικό.

3.3.7 Master's theses

```
<mastersthesis mdate="2002-01-03" key="phd/VanRoy84">
    <author>Peter Van Roy</author>
    <title>A Prolog Compiler for the PLM.</title>
    <year>1984</year>
    <school>University of California at Berkeley</school>
</mastersthesis>
```

Αφορά μεταπτυχιακές εργασίες. Υπάρχουν ελάχιστες μεταπτυχιακές εργασίες στην DBLP 3.1. Αναγκαία είναι τα πεδία που περιέχουν τον τίτλο της εργασίας, τον συγγραφέα, τον χρόνο σύνταξης της εργασίας καθώς και το εκπαιδευτικό ίδρυμα όπου πραγματοποιήθηκε. Οποιοδήποτε άλλο πεδίο δεν είναι απαραίτητο.

3.4 Εξαγωγή δεδομένων με χρήση Java

Ας δούμε τώρα πως μπορούμε να εξάγουμε όλα αυτά τα δεδομένα με τη χρήση της Java γλώσσας προγραμματισμού. Αρχικά πρέπει να κατεβάσουμε το xml αρχείο της DBLP καθώς και το DTD αρχείο που το περιγράφει. Προτείνεται το xml αρχείο το οποίο θα διαχειριστούμε να μην είναι μεγάλο σε μέγεθος διότι ο μέγιστος αριθμός οντοτήτων που επιτρέπει η Java είναι περιορισμένος. Αυτό φυσικά εξαρτάται και από τις δυνατότητες του μηχανήματος που διαθέτουμε. Θα χρησιμοποιήσουμε ως παράδειγμα το άρθρο που αναφέρεται στην ενότητα 3.3.1.

Οι εντολές που φαίνονται παρακάτω είναι αυτές που μας επιτρέπουν να έχουμε πρόσβαση στα διάφορα στοιχεία του xml αρχείου. Η πρώτη εντολή αυξάνει το σύνολο των οντοτήτων που μπορούμε να διαχειριστούμε. Ωστόσο όπως αναφέραμε πριν αυτός ο αριθμός έχει ένα άνω φράγμα που εξαρτάται από τις ικανότητες του μηχανήματος που χρησιμοποιούμε. Η δεύτερη εντολή ανοίγει το xml αρχείο ενώ οι υπόλοιπες συνεισφέρουν στη δημιουργία μιας μεταβλητής τύπου Document μέσω της οποίας μπορούμε να έχουμε πρόσβαση στα στοιχεία του xml.

```
System.setProperty("entityExpansionLimit", "2000000");
File fXmlFile = new File("dblp.xml");
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(fXmlFile);
doc.getDocumentElement().normalize();
```

Για να αποκτήσουμε πρόσβαση σε όλα τα στοιχεία τα οποία ανήκουν στην κατηγορία ”article” χρησιμοποιούμε την 1η εντολή από το παρακάτω κομμάτι κώδικα. Στη συνέχεια μπορούμε μέσω μίας εντολής επανάληψης να αποκτήσουμε πρόσβαση στα πεδία του κάθε επιστημονικού άρθρου. Με την εντολή στη γραμμή 6 εξάγουμε το κλειδί που αποτελεί ένα γνώρισμα του αντίστοιχου άρθρου. Δηλαδή εφόσον τρέξουμε αυτή την εντολή η μεταβλητή τύπου string key θα πάρει την τιμή ”journals/jet/KunimotoS04”. Με τις εντολές στις γραμμές 7, 8 και 9 εξάγουμε τα πεδία ”title”,”year” και ”journal” αντίστοιχα. Οι τιμές που λαμβάνουν οι μεταβλητές είναι οι εξής: title = ”Bargaining and competition revisited.”, year = ”2004”, journal = ”J. Econ. Theory”.

```
1  NodeList nList = doc.getElementsByTagName("article");
2  for (int temp = 0; temp < nList.getLength(); temp++) {
3     Node nNode = nList.item(temp);
4     if (nNode.getNodeType() == Node.ELEMENT_NODE) {
5        Element eElement = (Element) nNode;
6        String key = eElement.getAttribute("key");
7        String title = eElement.getElementsByTagName("title").item(0).
             getTextContent();
8        String year = eElement.getElementsByTagName("year").item(0).
             getTextContent();
9        String journal = eElement.getElementsByTagName("journal").item(0).
             getTextContent();
10    }
11 }
```

Όταν έχουμε πεδία τα οποία υπάρχουν μία ή περισσότερες φορές τότε αποκτούμε πρόσβαση σε αυτά με τη χρήση μίας εντολής επανάληψης. Με το τέλος της παρακάτω επανάληψης ο πίνακας ”authors” θα έχει τις τιμές: authors[0] = ”Takashi Kunimoto” και authors[1] = ”Roberto Serrano”.

```
1  for (int i=0; i<eElement.getElementsByTagName("author").getLength(); i++) {
2     authors[i] = eElement.getElementsByTagName("author").item(i).
          getTextContent();
3  }
```

3.5 Παρόμοιες βάσεις με την DBLP

Θα κλείσουμε αυτό το κεφάλαιο αναφέροντας ορισμένες βάσεις σχετικές με την DBLP. Δηλαδή διαδικτυακές βάσεις οι οποίες περιέχουν ένα μεγάλο όγκο δεδομένων που σχετίζονται με δημοσιεύσεις στο χώρο της επιστήμης τον υπολογιστών. Επιπλέον θα αξιολογήσουμε το πόσο εύκολο είναι να εξάγει κάποιος δεδομένα από αυτές και συνεπώς τη σημασία που θα είχαν σε μία τέτοιου είδους εφαρμογή.

3.5.1 ACM digital library

Η ψηφιακή βιβλιοθήκη ACM (Association for Computing Machinery) [2] είναι μια πλατφόρμα έρευνας, ανακάλυψης και δικτύωσης που περιέχει μία ολοκληρωμένη συλλογή κειμένων όλων των εκδόσεων ACM, συμπεριλαμβανομένων περιοδικών, συνεδρίων, τεχνικών περιοδικών, ενημερωτικών δελτίων και βιβλίων καθώς και επιμελημένες και φιλοξενούμενες εκδόσεις πλήρους

κειμένου από επιλεγμένους εκδότες. Η ACM επικεντρώνεται αποκλειστικά στον τομέα της επιστήμης τον υπολογιστών και αποτελεί ένα πλούσιο διασυνδεδεμένο σύνολο συνδέσεων μεταξύ συγγραφέων, έργων, ιδρυμάτων και εξειδικευμένων κοινοτήτων. Περιέχει περίπου 2.8 εκατομμύρια δημοσιεύσεις που έχουν εκδοθεί από το 1936 μέχρι σήμερα.

Το να εξάγει κάποιος δεδομένα από την ACM δεν αποτελεί εύκολη διαδικασία. Παρέχεται ένα αρχείο μορφής KBART σε ένα txt αρχείο, ωστόσο η δομή του αποτελείται από συνέδρια και σεμινάρια με συνδέσμους προς κάθε από αυτά. Για να εξάγει κάποιος επαρκείς πληροφορίες για μία δημοσίευση θα πρέπει να διαχειριστεί το html αρχείο που της αντιστοιχεί [4], διαδικασία που είναι αρκετά πιο δύσκολη από την εξαγωγή δεδομένων μέσω ενός οργανωμένου αρχείου xml.

3.5.2 IEEE Xplore

Η ψηφιακή βιβλιοθήκη IEEE Xplore [13] είναι μία ισχυρή πηγή για την ανακάλυψη και την πρόσβαση σε επιστημονικό και τεχνικό περιεχόμενο που δημοσιεύεται από το IEEE (Institute of Electrical and Electronics Engineers) και τους εκδότες του. Η IEEE Xplore παρέχει πρόσβαση σε περισσότερα από τέσσερα εκατομμύρια έγγραφα πλήρους κειμένου από μερικές από τις πιο δημοφιλείς δημοσιεύσεις στον κόσμο στον τομέα της ηλεκτρολογίας, της πληροφορικής και της ηλεκτρονικής, ενώ περίπου 20.000 νέα έγγραφα προστίθενται κάθε μήνα.

Πληροφορίες σχετικά με την εξαγωγή δεδομένων από την IEEE Xplore με οποιοδήποτε τρόπο ήταν δυσεύρετες. Παρ' όλα αυτά, πιθανότατα μία προσέγγιση όπως αυτή που έγινε στην ACM να μπορεί να επιφέρει αποτελέσματα.

3.5.3 The Collection of Computer Science Bibliographies

Πρόκειται για μια συλλογή βιβλιογραφιών που σχετίζονται με στην επιστήμη των υπολογιστών από διάφορες πηγές, που καλύπτουν τις περισσότερες πτυχές της επιστήμης των υπολογιστών [1]. Οι βιβλιογραφίες ενημερώνονται εβδομαδιαίως από τις αρχικές τους τοποθεσίες έτσι ώστε κάθε στιγμή, να μπορεί να βρει κάποιος τις πιο πρόσφατες εκδόσεις. Ο όγκος των δεδομένων είναι τεράστιος αφού εμπεριέχονται περισσότερες από 7 εκατομμύρια αναφορές, κυρίως σε άρθρα περιοδικών, δημοσιεύσεις σε συνέδρια και τεχνικές εκθέσεις και αποτελείται από περίπου 2.3 GBytes καταχωρίσεων BibTeX.

Στην ιστοσελίδα της βάσης μπορεί κάποιος να πραγματοποιήσει αναζητήσεις που θυμίζουν τη μορφή ερωτημάτων από μία γλώσσα διατύπωσης ερωτήσεων, αφού παρέχεται η δυνατότητα χρήσης λογικών τελεστών και αναζήτησης με βάση την τιμή συγκεκριμένων πεδίων όπως "ti" για προσδιορισμό τίτλων ή "au" για προσδιορισμό συγγραφέων. Δεν υπάρχει κάποια μορφής αρχείου που μπορεί να κατεβάσει κάποιος. θα μπορούσε να γίνει μία προσπάθεια εξαγωγής δεδομένων μέσω των αποτελεσμάτων των query, αλλά η δομή των αποτελεσμάτων δεν ακολουθεί κάποια οργανωμένη μορφή, με συνέπεια ένα τέτοιο εγχείρημα να αποτελούσε δύσκολη διαδικασία.

Κεφάλαιο 4

Ανάλυση της Neo4j

Στο πλαίσιο αυτού του κεφαλαίου θα εξετάσουμε τη Neo4j. Αρχικά θα δούμε ορισμένα χαρακτηριστικά της Neo4j, τις δυνατότητές της καθώς και την αρχιτεκτονική της. Στη συνέχεια θα δούμε το πώς ανταποκρίνεται σε big data και θα εξετάσουμε τα πλεονεκτήματα που την χαρακτηρίζουν. Έπειτα θα δούμε πως υλοποιούνται έννοιες όπως availability, consistency κ.λπ. και θα κάνουμε μία σύγκριση με τις σχεσιακές βάσεις δεδομένων. Κλείνοντας θα δούμε τη γλώσσα διατύπωσης ερωτήσεων Cypher που χρησιμοποιείται από την Neo4j αλλά και το πως μπορούμε να διαχειριστούμε δεδομένα στη Neo4j με τη χρήση της Java.

4.1 Βασικά χαρακτηριστικά της Neo4j

Η Neo4j είναι μία NOSQL βάση δεδομένων γράφων ανοιχτού κώδικα. Η αρχική ανάπτυξη ξεκίνησε το 2003, αλλά είναι διαθέσιμη στο κοινό από το 2007. Ο πηγαίος κώδικας είναι γραμμένος σε Java και Scala. Επίσης η Neo4j διαθέτει προγράμματα οδήγησης για δημοφιλείς γλώσσες προγραμματισμού όπως Java, JavaScript, .NET, Python κ.λπ.

Βασικό χαρακτηριστικό της Neo4j και γενικά των βάσεων δεδομένων γράφων είναι ότι οι σχέσεις μεταξύ των κόμβων είναι κανονικές οντότητες. Αυτό σημαίνει ότι μπορούμε να τις δούμε απευθείας, απλά κοιτώντας την μνήμη. Αντίθετα σε άλλες τεχνολογίες βάσεων δεδομένων οι σχέσεις μεταξύ των οντοτήτων υπονοούνται, και πρέπει εμείς να τις συμπεράνουμε συχνά χρησιμοποιώντας επινοημένες ιδιότητες όπως το ξένο κλειδί, ή επεξεργασίες όπως η map-reduce.

Ειδικά όταν μιλάμε για δεδομένα πολύ στενά συνδεδεμένα με σχέσεις διαφορετικών τύπων ή για ερωτήματα διάσχισης του γράφου, οι βάσεις αυτές έχουν πολύ υψηλή απόδοση. Αυτό συμβαίνει επειδή τα ερωτήματα εξετάζουν ένα τμήμα του γράφου αντί το σύνολο των δεδομένων και έτσι αποφεύγονται οι πολλές ενώσεις (joins) των σχεσιακών βάσεων. Έτσι η καθυστέρηση απόκρισης εξαρτάται από το μέγεθος του τμήματος του γράφου που επιλέγουμε να εξερευνήσουμε και όχι από τον όγκο των αποθηκευμένων δεδομένων.

Πέρα από την απόδοση τους, ένα άλλο χαρακτηριστικό των βάσεων δεδομένων γράφου είναι το μη προκαθορισμένο σχήμα τους. Έτσι, είναι πολύ εύκολο να προσθέσουμε νέου τύπου δεδομένα στην υπάρχουσα δομή, όπως κάποιο νέο είδος κόμβων, σχέσεων και ιδιοτήτων ή ακόμα και κάποιον ολόκληρο υπογράφο χωρίς να πειράξουμε τα υπάρχοντα ερωτήματα και την λειτουργικότητα

της εφαρμογής. Αυτό οδηγεί σε αύξηση της ευελιξίας. Λόγω αυτής της ευελιξίας δεν χρειάζεται να μοντελοποιήσουμε από την αρχή πλήρως το πρόβλημα μας.

Η Neo4j:

1. Περιέχει κόμβους, σχέσεις και ιδιότητες.
2. Οι κόμβοι αντιπροσωπεύουν τις οντότητες, έχουν επιγραφές και περιέχουν ιδιότητες. Τα κλειδιά των ιδιοτήτων είναι συμβολοσειρές και οι τιμές τους είναι αυθαίρετοι τύποι δεδομένων.
3. Οι σχέσεις συνδέουν τους κόμβους. Έχουν κατεύθυνση, μια επιγραφή, και έναν αρχικό και τελικό κόμβο. Δεν υπάρχουν αιωρούμενες σχέσεις.
4. Όπως και οι κόμβοι έτσι και οι σχέσεις μπορούν να έχουν ιδιότητες. Αυτή η ικανότητα των σχέσεων να έχουν ιδιότητες είναι πολύ χρήσιμη τόσο σε διάφορους αλγορίθμους που εφαρμόζονται πάνω στους γράφους όσο και στον περιορισμό των ερωτημάτων καθώς αυτά εκτελούνται, καθώς προσθέτει στις σχέσεις σημασιολογία (βάρος, ποιότητα).

Τέλος, η Neo4j είναι μια βάση δεδομένων γράφου με ιδιότητες. Αυτό σημαίνει ότι κάθε σχέση έχει, πέρα από τις ιδιότητες και την επιγραφή, έναν κόμβο αρχής και έναν κόμβο πέρατος και συνεπώς κατεύθυνση. Πολλές φορές όμως, στα πλαίσια μιας εφαρμογής όπως και της παρούσας, δεν θέλουμε οι σχέσεις να έχουν κατεύθυνση. Στην περίπτωση αυτή αντί για σχέσεις διπλής κατεύθυνσης, μπορούμε απλά να αγνοήσουμε την κατεύθυνση των σχέσεων κατά την διάρκεια των ερωτημάτων.

4.2 Αρχιτεκτονική της Neo4j

Το μοντέλο πάνω στο οποίο είναι δομημένη η Neo4j δεν διαφέρει σε πολλά από αυτό των περισσότερων βάσεων δεδομένων. Η Neo4j έχει επιπλέον δύο χαρακτηριστικά. Χρησιμοποιεί κανονική επεξεργασία γράφου (native graph processing) και κανονική αποθήκευση γράφου (native graph storage). Το πρώτο αφορά τον τρόπο που διαχειρίζεται τα ερωτήματα ενώ το δεύτερο τον τρόπο που αποθηκεύει την πληροφορία. Μια βάση δεδομένων γράφου λέμε ότι χρησιμοποιεί κανονική επεξεργασία γράφου αν διαθέτει μια ιδιότητα που ονομάζεται γειτνίαση χωρίς ευρετήριο (index-free adjacency). Αυτό σημαίνει ότι κάθε κόμβος διατηρεί απευθείας αναφορές σε όλους τους γειτονικούς του κόμβους (με την χρήση μιας λίστας από pointers που δείχνουν σε όλους τους γείτονες του κόμβου) και έτσι λειτουργεί σαν ένα μικρό ευρετήριο για την γειτονιά του. Έτσι ο χρόνος απόκρισης σε ερωτήματα εξαρτάται αποκλειστικά από το μέγεθος αυτής της γειτονιάς και όχι από το μέγεθος του γράφου.

Αντίθετα, μια βάση δεδομένων που δεν έχει αυτήν την ιδιότητα χρησιμοποιεί ευρετήρια για να συνδέσει τους κόμβους μεταξύ τους. Αυτά τα ευρετήρια αυξάνουν το υπολογιστικό κόστος όταν προσπαθούμε να διασχίσουμε τον γράφο. Μια βασική πτυχή του σχεδιασμού μια βάσης δεδομένων είναι και ο τρόπος που αποθηκεύονται τα δεδομένα (στην περίπτωση μας οι γράφοι) στην μνήμη.

Όπως αναφέραμε η Neo4j χρησιμοποιεί κανονική αποθήκευση γράφου, κάτι που οδηγεί σε υψηλή απόδοση στα ερωτήματα διάσχισης. Η Neo4j αποθηκεύει τα διάφορα δεδομένα σε διαφορετικά αρχεία αποθήκευσης. Κάθε ένα από αυτά τα αρχεία έχει πληροφορία για ένα τμήμα του γράφου (αλλού οι κόμβοι, αλλού οι σχέσεις, αλλού οι ιδιότητες κ.λπ.). Το γεγονός αυτό και κυρίως το γεγονός ότι διαχωρίζει τα δομικά στοιχεία του γράφου από τις ιδιότητες, οδηγεί σε πολύ γρήγορες διασχίσεις του γράφου.

4.3 Η Neo4j για big data

Όπως αναφέρθηκε η Neo4j χρησιμοποιεί κανονική επεξεργασία γράφου καθώς και κανονική αποθήκευση γράφου. Η κανονική επεξεργασία γράφου επιτρέπει σε κάθε κόμβο να γνωρίζει όλους τους γείτονες του μέσω μιας λίστας από pointers που δείχνουν σε αυτούς. Αυτό σε συνδυασμό με την κανονική αποθήκευση γράφου, κάνει την πρόσβαση σε κόμβους και σχέσεις μια λειτουργία σταθερού χρόνου που μας επιτρέπει να διασχίσουμε εκατομμύρια συνδέσεις ανά δευτερόλεπτο.

Χαρακτηριστικό αποτελεί το παρακάτω πείραμα [20] το οποίο διεξήχθη, με θέμα την αναζήτηση των φίλων που έχουν οι φίλοι ενός χρήστη. Η ώρα εκτέλεσης είναι για 1000 χρήστες ενώ η βάση περιέχει 1.000.000 χρήστες.

Depth	Execution Time – MySQL	Execution Time –Neo4j
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	Not Finished in 1 Hour	2.132

Σχήμα 4.1: Αποτελέσματα πειράματος big data στις Neo4j και MySQL βάσεις [20]

Παρατηρούμε πως με βάθος 2 η Neo4j είναι 60% πιο γρήγορη, με βάθος 3 180 φορές πιο γρήγορη ενώ με βάθος 4 είναι 1.135 πιο γρήγορη. Για βάθος 5 η MySQL δεν επιστρέφει αποτέλεσμα ακόμα και μετά από την ολοκλήρωση μίας ώρας.

4.4 Πλεονεκτήματα της Neo4j

Ας συνοψίσουμε εν συντομία τα πλεονεκτήματα που χαρακτηρίζουν την Neo4j. Αυτά είναι τα εξής:

1. Η Neo4j μας επιτρέπει να δημιουργήσουμε απλά μοντέλα που χαρτογραφούν το πρόβλημα που προσπαθούμε να επιλύσουμε. Τα δεδομένα έχουν μορφή πολύ κοντινή με αυτή στον πραγματικό κόσμο με αποτέλεσμα να γίνονται πολύ πιο εύκολα κατανοητά.

2. Η λίστα των γειτονικών κόμβων που διαθέτει ο κάθε κόμβος καθώς και ο τρόπος με τον οποίο αποθηκεύονται τα δεδομένα στη Neo4j κάνει την διάσχιση κόμβων και σχέσεων για την επίλυση ερωτημάτων ιδιαίτερα αποδοτική.

3. Η Cypher που είναι η γλώσσα διατύπωσης ερωτήσεων της Neo4j και αναλύεται στην ενότητα 4.7 είναι μία πολύ απλή και κατανοητή δηλωτική γλώσσα ερωτημάτων η οποία ειδικεύεται σε γραφήματα και διευκολύνει σημαντικά την σύνταξη ερωτημάτων.

4. Η Neo4j κάνει δυνατή τη δημιουργία δεικτών για συγκεκριμένα δεδομένα της βάσης που εμείς επιλέγουμε, με σκοπό την πιο γρήγορη διαπέραση τους, κάτι που ενισχύει περαιτέρω την απόδοση.

4.5 Διαθεσιμότητα, Συνεκτικότητα, Ανοχή Διαμερισμού

Η Neo4j μας επιτρέπει να δημιουργήσουμε αντίγραφα των δεδομένων μας με τη δημιουργία ενός cluster. Η δομή αποτελείται από τους κεντρικούς διακομιστές που περιέχουν τα δεδομένα μας και από τους διακομιστές αντιγράφων που περιέχουν ένα αντίγραφο όλων των δεδομένων μας με σκοπό την πιο εύκολη πρόσβαση σε αυτά. Κάτι τέτοιο φυσικά αυξάνει σε σημαντικό βαθμό και την διαθεσιμότητα των δεδομένων με αποτέλεσμα η Neo4j να αποτελεί μία βάση που προσφέρει ιδιαίτερα υψηλή διαθεσιμότητα των δεδομένων μας.

Η Neo4j με σκοπό την διατήρηση της συνεκτικότητας των δεδομένων υποστηρίζει όλες τις ιδιότητες του ACID (Atomic, Consistent, Isolated, Durable) [18] μοντέλου. Επιπλέον παρέχονται λειτουργίες με σκοπό τον έλεγχο της συνεκτικότητας των δεδομένων τόσο σε βάσεις που έχουμε δημιουργήσει όσο και σε αντίγραφα ασφαλείας βάσεων που μπορεί να έχουμε. Τέλος η Neo4j δεν έχει ανοχή στο διαμερισμό με συνέπεια αν υπάρχει κάποια βλάβη στο δίκτυο, το σύστημα να μην συνεχίζει την λειτουργία του.

4.6 Σύγκριση με τις Σχεσιακές βάσεις δεδομένων

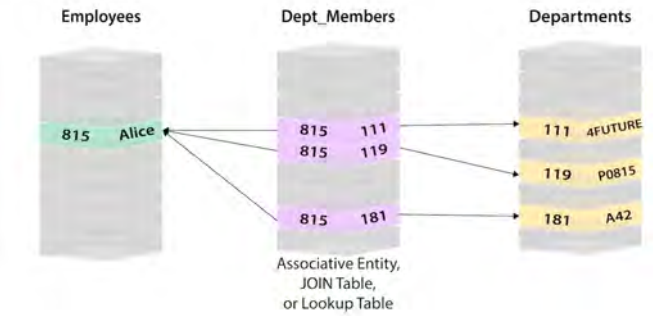
Για να γίνει κατανοητό το τι ακριβώς κερδίζουμε χρησιμοποιώντας τη Neo4j σε σχέση με τις σχεσιακές βάσεις θα εξετάσουμε τις εννοιολογικές διαφορές τον δομών και στη συνέχεια θα δούμε ένα απλό παράδειγμα πραγματοποίησης ενός ερωτήματος και στις δύο περιπτώσεις.

Οι σχεσιακές βάσεις αποθηκεύουν δεδομένα σε πίνακες με προκαθορισμένες στήλες, συγκεκριμένων τύπων και πολλές σειρές αυτών των τύπων πληροφοριών. Οι συνδέσεις υπολογίζονται κατά τη διάρκεια των ερωτήσεων προσαρμόζοντας τα πρωτεύοντα και τα ξένα κλειδιά όλων των γραμμών των συνδεδεμένων πινάκων. Αυτές οι λειτουργίες όμως είναι περίπλοκες, απαιτούν μνήμη και έχουν υψηλό κόστος. Επίσης ανάλογα με τη δομή των δεδομένων μας μπορεί να χρειαστεί να εισάγουμε και κάποιο πίνακα ένωσης (Join) που περιέχει τα ξένα κλειδιά των συμμετεχόντων πινάκων αυξάνοντας περαιτέρω το κόστος λειτουργίας της σύνδεσης.

Οι βάσεις δεδομένων γράφων αποθηκεύουν τα δεδομένα σε μορφή γραφήματος που αποτελείται από κόμβους που ανήκουν σε μία οντότητα και σχέσεις που υπάρχουν μεταξύ των κόμβων. Τα δεδομένα παραμένουν αξιοσημείωτα παρόμοια με τη μορφή τους στον πραγματικό κόσμο. Αυτό μας επιτρέπει να αναζητήσουμε και να προβάλουμε τα δεδομένα μας από οποιοδήποτε σημείο ενδιαφέροντος υποστηρίζοντας πολλές διαφορετικές περιπτώσεις χρήσης. Κάθε κόμβος σε αυτό το μοντέλο περιέχει μία λίστα με όλους τους γειτονικούς κόμβους με τους οποίους έχει κάποια

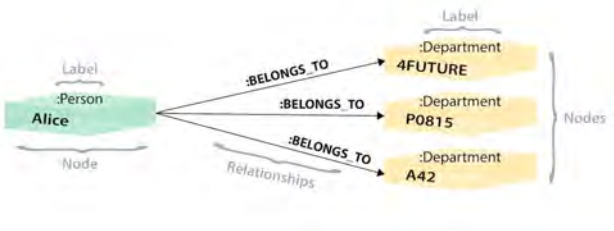
σχέση. Έτσι κάθε φορά που εκτελείται το ισοδύναμο της λειτουργίας ένωσης των σχεσιακών βάσεων η Neo4j χρησιμοποιεί αυτή τη λίστα αποκτώντας απευθείας πρόσβαση στους συνδεδεμένους κόμβους εξαλείφοντας την ανάγκη ακριβών υπολογισμών αναζήτησης και αντιστοίχησης.

Ας συγκρίνουμε τα δύο μοντέλα και με ένα πρακτικό παράδειγμα. Στο σχεσιακό μοντέλο (σχ. 4.2) αναζητούμε στον πίνακα υπαλλήλων στα αριστερά, ενδεχομένως σε εκατομμύρια σειρές, για να βρούμε τον υπάλληλο με όνομα ”Alice” και id ”815”. Στη συνέχεια αναζητούμε στον μεσαίο πίνακα Join για να εντοπίσουμε όλες τις σειρές που περιέχουν το id της Alice. Αφού ανακτήσουμε όλες τις σχετικές σειρές πηγαίνουμε στον πίνακα των τμημάτων στα δεξιά για να αναζητήσουμε τις τιμές των αναγνωριστικών τμήματος ”111”, ”119” και ”181”. Εφόσον ολοκληρωθεί και αυτή η διαδικασία γνωρίζουμε πλέον πως η Alice ανήκει στα τμήματα με όνομα ”4FUTURE”, ”P0815” και ”A42”.



Σχήμα 4.2: Δομή δεδομένων στο σχεσιακό μοντέλο [19]

Στο μοντέλο δεδομένων γράφων (σχ. 4.3) έχουμε ένα μοναδικό κόμβο για την Alice με επιγραφή (label) ”Person”. Η Alice ανήκει σε 3 διαφορετικά τμήματα συνεπώς δημιουργούμε ένα διαφορετικό κόμβο για καθένα από αυτά με επιγραφή ”Department”. Για να βρούμε σε ποια τμήματα ανήκει η ”Alice” αναζητούμε στο γράφημα τον κόμβο της και στη συνέχεια διασχίζουμε όλες της σχέσεις με επιγραφή ”BELONGS_TO” για να βρούμε τους κόμβους των τμημάτων με τα οποία είναι σενδεδεμένη. Αυτή είναι η μοναδική διαδικασία που χρειάζεται να κάνουμε.



Σχήμα 4.3: Δομή δεδομένων στο μοντέλο δεδομένων γράφων [19]

4.7 Ανάλυση της Cypher

Η γλώσσα που χρησιμοποιεί η Neo4j για την πραγματοποίηση ερωτημάτων είναι η Cypher. Η Cypher είναι μια δηλωτική γλώσσα ερωτημάτων για βάσεις δεδομένων γράφου. Είναι πολύ εκφρα-

στική, από την άποψη ότι μοιάζει πολύ με τον τρόπο που ο άνθρωπος αντιλαμβάνεται διαισθητικά τους γράφους. Επειδή ακριβώς είναι δηλωτική, επικεντρώνεται στο να ορίσει με σαφήνεια τι να ανασύρουμε από τον γράφο και όχι πως να το ανασύρουμε.

Η Cypher βασίζεται στις βασικές έννοιες και τη λογική της SQL, αλλά διαθέτει πολλές επιπρόσθετες λειτουργίες ειδικές για γραφήματα για να διευκολύνει την εργασία με οποιοδήποτε μοντέλο γραφήματος. Η διατύπωση ερωτημάτων στην SQL αποτελεί μία μη ξεκάθαρη διαδικασία, ιδιαίτερα όταν πρόκειται για πολύπλοκα ερωτήματα. Στη Cypher, η σύνταξη παραμένει περιεκτική και επικεντρώνεται στα στοιχεία των οντοτήτων και τις συνδέσεις μεταξύ τους εκφράζοντας τη διατύπωση εύρεσης ή δημιουργίας δεδομένων πιο οπτικά και καθαρά. Χαρακτηριστικό αποτελεί το παράδειγμα παρακάτω όπου φαίνεται η σύνταξη ενός ερωτήματος για την εύρεση όλων τον υπαλλήλων ενός συγκεκριμένου τμήματος σε SQL και Cypher αντίστοιχα.

```
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"
```

Σχήμα 4.4: Διατύπωση ερωτήματος σε SQL

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

Σχήμα 4.5: Διατύπωση ερωτήματος σε Cypher

Ας δούμε τώρα ορισμένα απλά παραδείγματα διατύπωσης ερωτημάτων με χρήση της Cypher. Στο παρακάτω κομμάτι κώδικα βλέπουμε τον τρόπο με τον οποίο δημιουργούμε νέους κόμβους και τις σχέσεις που έχουν μεταξύ τους. Μετά την εκτέλεση του, θα έχουμε δημιουργήσει ένα κόμβο τύπου ”actor” με την ιδιότητα ”name” να έχει την τιμή ”Tom”. Με όμοιο τρόπο δημιουργείται ένας κόμβος τύπου ”city” που περιέχει την πόλη με όνομα ”Paris”. Τέλος δημιουργείται η σχέση που διέπει τους δύο κόμβους η οποία έχει ως ονομασία την ετικέτα ”LIVES_IN” και έχει κατεύθυνση προς την πόλη. Το παραγόμενο γράφημα είναι αυτό που φαίνεται στο σχήμα 4.6.

```
1   CREATE (a:actor {name:'Tom'})-[:LIVES_IN]->(c:city {name:'Paris'})
```

Σχήμα 4.6: Γράφημα μετά την εισαγωγή δεδομένων

Στο παρακάτω κομμάτι κώδικα βλέπουμε τον τρόπο με τον οποίο μπορούμε να λάβουμε τις ιδιότητες ενός κόμβου. Με την περάτωση του, μας επιστρέφονται τα ονόματα και οι ηλικίες όλων των ηθοποιών που συμμετείχαν στην ταινία με τίτλο ”The Matrix”.

```
MATCH (m:movie {title: 'The Matrix'})<-[:ACTED_IN]-(a:actor)
RETURN a.name, a.age
```

Ας δούμε τώρα πως μπορούμε να θέσουμε ένα πεδίο σε ένα κόμβο που έχει ήδη δημιουργηθεί. Με την εκτέλεση του κώδικα παρακάτω ο ηθοποιός με όνομα ”Tom Cruise” θα αποκτήσει το πεδίο ”age” με τιμή 57. Εάν το πεδίο υπήρχε ήδη τότε η τιμή του θα αλλάξει σε 57.

```
MATCH (a:actor {name:'Tom Cruise'})
SET a.age = '57'
```

Τέλος βλέπουμε πως μπορούμε να διαγράψουμε ένα κόμβο. Η εντολή DETACH προσδιορίζει την διαγραφή των σχέσεως που έχει ο κόμβος με άλλους γειτονικούς κόμβους, εάν υπάρχουν, ενώ η εντολή DELETE διαγράφει τον ίδιο τον κόμβο. Αν ο κόμβος έχει οποιαδήποτε σχέση με κάποιο γειτονικό κόμβο η εντολή DELETE δε θα λειτουργούσε μόνη της, αλλά θα χρειαζόταν και την επιλογή DETACH. Με την εκτέλεση του συγκεκριμένου κώδικα διαγράφονται όλοι οι ηθοποιοί με όνομα ”Tom”.

```
MATCH (a:actor {name:'Tom'})
DETACH DELETE a
```

4.8 Διαχείριση δεδομένων στη Neo4j με χρήση Java

Η διαδικασία διαχείρισης της Neo4j είναι παρόμοια με εκείνη που θα πραγματοποιούσαμε αν είχαμε μία σχεσιακή βάση (π.χ. PostgreSQL ή MySQL). Το μόνο που χρειαζόμαστε είναι το πρόγραμμα οδήγησης (driver) που παρέχει η Neo4J για την Java. Εφόσον το συμπεριλάβουμε στο project μας είμαστε πλέον ικανοί να συνδεθούμε στη Neo4j και να πραγματοποιήσουμε ερωτήματα μέσω της Java. Ο κώδικας που απαιτείται για τη σύνδεση σε μία ενεργή βάση Neo4j βρίσκεται παρακάτω. Το 1ο όρισμα περιέχει τον σύνδεσμο στον οποίο τρέχει η βάση, το 2ο την ονομασία του όνομα χρήστη που έχουμε και το 3ο τον κωδικό που έχουμε ορίσει για να συνδεθούμε σε αυτή.

```
String url = ”jdbc:neo4j:bolt://localhost”;
Connection conn = DriverManager.getConnection(url, ”neo4j”, ”1234”);
```

Εάν θέλουμε να πραγματοποιήσουμε μία εισαγωγή στη βάση τότε ο κώδικας που απαιτείται είναι αυτός που βρίσκεται παρακάτω. Εφόσον εκτελεστεί θα έχουμε δημιουργήσει ένα κόμβο τύπου ”movie” με τίτλο ”The Matrix”.

```
String queryStr = ”create (m:movie {title:'The Matrix'})”);
Statement stmt = conn.createStatement();
stmt.executeUpdate(queryStr);
```

Ενώ εάν θέλουμε να πραγματοποιήσουμε ένα query που εξάγει δεδομένα από την Neo4j, μπορούμε να εκτελέσουμε το παρακάτω κομμάτι κώδικα. Αφού εκτελεστεί η μεταβλητή ”age” θα έχει την ηλικία του ηθοποιού με όνομα ”Tom Cruise”, με την προϋπόθεση πως υπάρχει στη βάση κάποιος ηθοποιός με αυτή την ονομασία.

```
String queryStr = "MATCH (a:actor {name:'Tom Cruise'}) return a.age");
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
if (rs.next()) {
   String age = rs.getString("a.age");
}
```

Κεφάλαιο 5

Παρουσίαση της εφαρμογής

Σε αυτό το κεφάλαιο θα αναλύσουμε εξ ολοκλήρου τις λειτουργίες της εφαρμογής που παρουσιάζει αυτή η διπλωματική. Αυτή αποτελείται από τρία μέρη, την εξαγωγή δεδομένων από την DBLP, την εισαγωγή τους στη Neo4j και την πραγματοποίηση ορισμένων ερωτημάτων πάνω σε αυτά τα δεδομένα με σκοπό την απάντηση συγκεκριμένων προβλημάτων. Πιο πολύ θα σταθούμε στο 3ο μέρος της εφαρμογής αφού τα άλλα δύο μέρη έχουν ήδη εξηγηθεί, ως ένα βαθμό, στα Κεφάλαια 3 και 4 αντίστοιχα. Θα αναλύσουμε το κάθε μέρος ξεχωριστά.

Επισημαίνεται πως η εφαρμογή αποτελείται από δύο βασικά αρχεία Java. Το ”CreateDatabase .java” που αφορά την εξαγωγή δεδομένων από την DBLP και την εισαγωγή τους στη Neo4j και το ”ExecuteQueries.java” που αφορά την εκτέλεση ερωτημάτων. Υπάρχουν και άλλα βοηθητικά αρχεία Java που θα αναφερθούν καθώς αναλύουμε τους διάφορους αλγορίθμους της εφαρμογής.

5.1 Εξαγωγή δεδομένων από την DBLP

Η εξαγωγή των δεδομένων συμβαίνει όπως ακριβώς περιγράφεται στην ενότητα 3.4. Ουσιαστικά παίρνουμε το xml που περιέχει τα δεδομένα της DBLP και εξάγουμε μέσω αυτού τις δημοσιεύσεις που περιέχει καθώς και τα δεδομένα που τις χαρακτηρίζουν. Χρησιμοποιούνται όλα τα στοιχεία, εφόσον υπάρχουν, που παρατίθενται στην ενότητα 3.3. Πληροφορίες που δεν χρησιμοποιούνται είναι αυτές που περιγράφονται από τις ετικέτες ”url”,”cdrom”,”note”,”crossref” καθώς αντί του ”url” εξάγεται το ”ee” πεδίο, ενώ τα πεδία ”cdrom” και ”note” συναντιούνται πολύ σπάνια για να έχει κάποια ουσία η εξαγωγή τους. Το πεδίο ”crossref” περιέχει το κλειδί του τόμου στον οποίο ανήκει μία εργασία (π.χ. μια εργασία σε συνέδριο περιέχει το κλειδί του τόμου στον οποίο ανήκει), αλλά η λογική της συγκεκριμένης εφαρμογής δεν απαιτεί την χρήση του. Σε ορισμένες περιπτώσεις δημοσιεύσεων τύπου ”book” συναντάμε και το πεδίο ”booktitle” το οποίο συνήθως αποτελεί μία γενίκευση του τίτλου του (τις περισσότερες φορές είναι σχεδόν ίδιο με το πεδίο ”title” ή δεν συμπεριλαμβάνεται καθόλου). Το συγκεκριμένο πεδίο δεν το λαμβάνουμε υπόψη (συμβαίνει μόνο στα βιβλία), δηλαδή εξάγουμε μόνο το πεδίο ”title” από τα βιβλία. Όσον αφορά τα γνωρίσματα, το μόνο που χρησιμοποιείται είναι το χαρακτηριστικό κλειδί της κάθε δημοσίευσης.

5.2 Εισαγωγή δεδομένων στη Neo4j

Εφόσον εξάγουμε τα δεδομένα η διαδικασία συνεχίζεται με την εισαγωγή τους στη Neo4j. Ο τρόπος με τον οποίο πραγματοποιείται η εισαγωγή με χρήση της Java παρουσιάστηκε και επεξηγήθηκε στις ενότητες 4.7 και 4.8.

5.2.1 Εισαγωγή δημοσιεύσεων

Η λογική εισαγωγής μίας δημοσίευσης στη βάση είναι η ακόλουθη:

1. Εισαγωγή της δημοσίευσης στη βάση μαζί με το χαρακτηριστικό της κλειδί καθώς και τα απαραίτητα πεδία που την περιγράφουν (εξηγούνται στην ενότητα 3.3).

2. Εισαγωγή των συγγραφέων της δημοσίευσης στη βάση, εφόσον δεν υπάρχουν ήδη και έπειτα δημιουργία της σχέσης ”WROTE” μεταξύ των συγγραφέων και της δημοσίευσης.

3. Εάν πρόκειται για article δημιουργία του περιοδικού, εφόσον δεν υπάρχει, στο οποίο εμπεριέχεται και σύνδεση μεταξύ τους με τη σχέση ”INCLUDES”.

4. Εάν πρόκειται για article, book, proceedings ή PhD thesis δημιουργία του εκδότη από τον οποίο έχει εκδοθεί, εφόσον δεν υπάρχει και στη συνέχεια σύνδεση μεταξύ τους με τη σχέση ”PUBLISHED”.

5. Εισαγωγή των υπολοίπων, προαιρετικών πεδίων εφόσον αυτά δεν είναι κενά.

Εδώ φαίνεται και η σημασία της ευελιξίας που μας παρέχει η Neo4j. Για παράδειγμα μπορεί να έχουμε ένα άρθρο το οποίο να έχει μόνο τον τίτλο, το χρόνο δημιουργίας, τους συγγραφείς του και το περιοδικό στο οποίο δημοσιεύθηκε, ενώ αμέσως μετά να έχουμε ένα διαφορετικό άρθρο που εκτός από όλα αυτά περιέχει το μήνα έκδοσής του καθώς και τον σύνδεσμο του στον παγκόσμιο ιστό. Παρά το γεγονός πως το 1ο άρθρο χαρακτηρίζεται από λιγότερα πεδία δεν χρειάζεται να περιέχει και τα πεδία ”month” και ”ee” με πιθανώς μία κενή τιμή η κάποια άλλη σύμβαση, όπως θα κάναμε δηλαδή σε μία σχεσιακή βάση. Περιέχει μόνο τα πεδία που έχουν κάποια ουσιαστική πληροφορία.

Η εισαγωγή των δημοσιεύσεων γίνεται από μία σειρά από xml που δημιουργούμε εμείς από το αρχικό xml. Αυτά τα xml προτείνεται να είναι μεγέθους ένα με δύο εκατομμύρια γραμμές διότι ο μέγιστος αριθμός οντοτήτων που επιτρέπει η Java είναι περιορισμένος. Αυτό φυσικά εξαρτάται και από τις δυνατότητες του μηχανήματος που διαθέτουμε. Η εφαρμογή ξεκινάει από το ”dblp0.xml” συνεχίζει με το ”dblp1.xml” κ.ο.κ. Μεγαλύτερη ανάλυση αυτής της διαδικασίας γίνεται στην ενότητα 7.3.

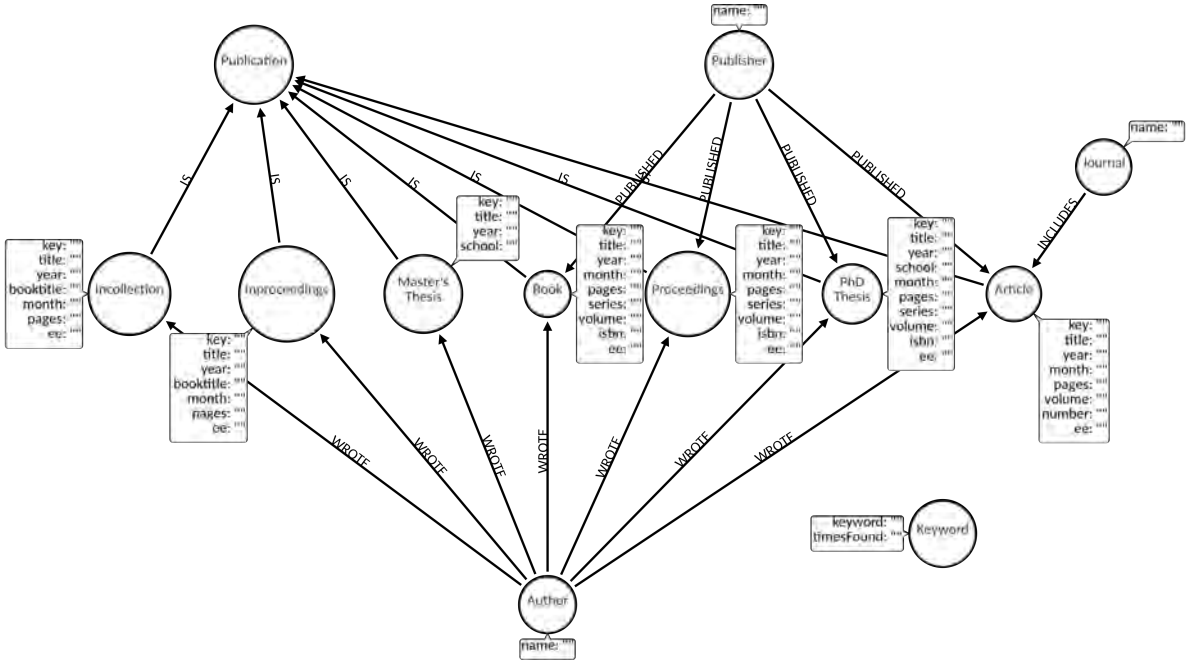
5.2.2 Εύρεση των πιο συνηθισμένων λέξεων-κλειδιών

Μετά από την εισαγωγή των δημοσιεύσεων ακολουθεί μία διαδικασία που έχει ως σκοπό να εξάγει τις πιο συνηθισμένες λέξεις-κλειδιά που χρησιμοποιούνται από την επιστημονική κοινότητα. Οι λέξεις-κλειδιά εξάγονται από τους τίτλους των δημοσιεύσεων που υπάρχουν στη βάση,

ενώ αγνοούνται όσες έχουν μήκος μικρότερο του 5. Η διαδικασία εξετάζει το πόσες φορές υπάρχει κάθε λέξη-κλειδί στους τίτλους των δημοσιεύσεων που απαρτίζουν τη βάση. Ο αριθμός αυτός προκύπτει με την εκτέλεση του παρακάτω query, απλά αντί για τη λέξη ”property” θα υπάρχει η αντίστοιχη. Όπως φαίνεται εξετάζουμε αν η λέξη υπάρχει τόσο στον ενικό όσο και στον πληθυντικό με τη χρήση ενός προγράμματος που μας επιτρέπει την μετατροπή λέξεων της αγγλικής γλώσσας απο ενικό σε πληθυντικό και αντίστροφα, το ”Inflector.java”. Η συνάρτηση ”toLower” της Neo4j μετατρέπει όλα τα κεφαλαία γράμματα του τίτλου σε μικρά. Στη βάση εισάγεται ένας αριθμός λέξεων-κλειδιών που ο χρήστης επιλέγει. Όλες οι λέξεις-κλειδιά εισάγονται στον ενικό, χωρίς κεφαλαία γράμματα. Οι λέξεις-κλειδιά κατατάσσονται σε σειρά με βάση τις φορές που συναντήθηκαν με τη χρήση μιας συνάρτησης συγκρίσεων που υλοποιείται στο ”KeywordRating.java”

```
MATCH (a)-[:IS]-(b:publication)
WHERE toLower(a.title) CONTAINS 'property'
OR toLower(a.title) CONTAINS 'properties'
RETURN count(a) AS timesFound
```

Μετά από το τέλος και της διαδικασίας με στόχο την εισαγωγή των πιο συνηθισμένων λέξεων-κλειδιών η βάση είναι έτοιμη και μπορούμε να ξεκινήσουμε την πραγματοποίηση ερωτημάτων. Η μορφή της βάσης φαίνεται πιο ξεκάθαρα στο σχήμα 5.1.



Σχήμα 5.1: Σχήμα της βάσης μετά την εκτέλεση του ”CreateDatabase .java”

Τέλος παρέχονται στο χρήστη κάποιες επιλογές μέσω μεταβλητών τύπου ”Boolean” (true/-false) καθώς και η τροποποίηση ορισμένων μεγεθών που αφορούν την εύρεση λέξεων- κλειδιών, στην αρχή του προγράμματος. Αυτές είναι:

1. clearDatabase: ενεργοποιεί/απενεργοποιεί την διαγραφή των δεδομένων της βάσης πριν την εισαγωγή των νέων δεδομένων.

2. insertXml: ενεργοποιεί/απενεργοποιεί την εισαγωγή νέων δημοσιεύσεων με τη χρήση xml αρχείων.
3. insertArticles: ενεργοποιεί/απενεργοποιεί την εισαγωγή άρθρων.
4. insertIncollections: ενεργοποιεί/απενεργοποιεί την εισαγωγή εργασιών που υπάρχουν σε μονογραφίες.
5. insertInproceedings: ενεργοποιεί/απενεργοποιεί την εισαγωγή εργασιών που υπάρχουν σε τόμους συνεδρίων.
6. insertBooks: ενεργοποιεί/απενεργοποιεί την εισαγωγή μονογραφιών.
7. insertProceedings: ενεργοποιεί/απενεργοποιεί την εισαγωγή τόμων από συνέδρια.
8. insertPhdtheses: ενεργοποιεί/απενεργοποιεί την εισαγωγή διδακτορικών διατριβών.
9. insertMasterstheses: ενεργοποιεί/απενεργοποιεί την εισαγωγή μεταπτυχιακών εργασιών.
10. findMostUsedKeywords: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο συνηθισμένων λέξεων-κλειδιών.
11. KEYWORD_ARRAY_LENGTH: ο αριθμός των λέξεων-κλειδιών που θα εισαχθούν στη βάση.
12. MIN_KEYWORD_LENGTH: το ελάχιστο μέγεθος μιας λέξης-κλειδί.

5.3 Πραγματοποίηση ερωτημάτων

Ας αναλύσουμε τώρα, το τρίτο και πιο ενδιαφέρον μέρος της εφαρμογής που αφορά την επίλυση διάφορων προβλημάτων με κατάλληλους αλγορίθμους. Για κάθε πρόβλημα θα παρατίθεται το query ή το σύνολο των queries τα οποία μας παρέχουν τα δεδομένα για την επίλυσή του, θα επεξηγείται η λειτουργία του αλλά και η λογική του και τέλος θα απαριθμούνται οι μεταβλητές εισόδου, οι οποίες σχετίζονται με αυτό το πρόβλημα, που ο χρήστης μπορεί να τροποποιήσει. Να επισημάνουμε πως οποιαδήποτε σύγκριση αλφαριθμητικών με την εντολή ”CONTAINS”, γίνεται δίχως κεφαλαία γράμματα, αλλά μόνο σε μικρά. Ο χρήστης μπορεί να εισάγει στο αντίστοιχο πεδίο και κεφαλαία, η μετατροπή γίνεται εσωτερικά μέσω του προγράμματος. Στα παρακάτω παραδείγματα θα εμπεριέχονται και κεφαλαίοι χαρακτήρες (που ο χρήστης μπορεί να εισάγει) για λόγους διευκόλυνσης της κατανόησης.

Επιπλέον να αναφέρουμε πως ο χρήστης μπορεί να επιλέξει εάν θέλει να χρησιμοποιήσει το UI της εφαρμογής αλλάζοντας την μεταβλητή enableUI, τύπου Boolean, σε true και false αντίστοιχα. Αν το UI της εφαρμογής ενεργοποιηθεί τότε ο χρήστης μπορεί να εισάγει όλες τις μεταβλητές που χρειάζεται ο κάθε αλγόριθμος μέσω της κονσόλας. Εκτός αυτού μπορεί να αλλάξει και ορισμένες επιλογές όπως τον αριθμό των προβαλλόμενων αποτελεσμάτων ή το άνοιγμα του συνδέσμου μετά την εύρεση μιας δημοσίευσης από τον αντίστοιχο αλγόριθμο και άλλων που αναφέρονται παρακάτω.

5.3.1 Καταμέτρηση των δεδομένων της βάσης

Ο σκοπός του συγκεκριμένου αλγορίθμου είναι η καταμέτρηση του συνόλου των κόμβων που αποτελούν τις διάφορες οντότητες που υπάρχουν στη βάση. Με το query της γραμμής 1 υπολογίζουμε τον αριθμό όλων των δημοσιεύσεων και τον επιστρέφουμε με την ονομασία ”count”, ενώ με τα queries στις γραμμές 2 και 3 υπολογίζουμε τον αριθμό των άρθρων και των συγγραφέων αντίστοιχα. Οι υπόλοιπες οντότητες καταμετρούνται με όμοιο τρόπο. Εφόσον εξαχθούν όλα τα δεδομένα, τότε προβάλλονται στον χρήστη (η διαδικασία εξαγωγής δεδομένων από τη Neo4j με χρήση Java αναλύθηκε στην ενότητα 4.8). Σκοπός του συγκεκριμένου αλγορίθμου είναι η εξαγωγή στατιστικών στοιχείων καθώς και συμβολή στο debugging [23].

```
1  MATCH (a)-[:IS]-(b:publication) RETURN count(a) AS count
2  MATCH (a:article) RETURN count(a) AS count
3  MATCH (a:author) RETURN count(a) AS count
4  // The same for the rest of the entities.
```

Η μόνη μεταβλητή που σχετίζεται με τον συγκεκριμένο αλγόριθμο είναι η showDatabaseData τύπου Boolean που τον ενεργοποιεί/απενεργοποιεί.

5.3.2 Εύρεση των πιο ενεργών μελών

Τώρα θα αναφερθούμε σε μια σειρά από αλγορίθμους που έχουν ως στόχο την εύρεση των πιο ενεργών μελών. Με τον όρο πιο ενεργά μέλη, εννοούμε την εύρεση των πιο ενεργών συγγραφέων, περιοδικών και εκδοτών.

Ο 1ος αλγόριθμος έχει ως στόχο την εύρεση των πιο ενεργών συγγραφέων για κάποια ορισμένη χρονολογία. Για παράδειγμα το παρακάτω query θα μας επιστρέψει τους 10 πιο ενεργούς συγγραφείς με βάση τον αριθμό των δημοσιεύσεων στις οποίες συμμετείχαν, για τη χρονολογία ”2020”. Στο χρήστη δίνεται η δυνατότητα να αφαιρέσει την χρονολογία με αποτέλεσμα να μας επιστρέφονταν οι 10 πιο ενεργοί συγγραφείς όλων των εποχών.

```
1  MATCH (a:author)-[:WROTE]-(b {year:'2020'})
2  RETURN a.name, count(b) AS publications
3  ORDER BY publications DESC
4  LIMIT 10
```

Όμοια με πριν σκοπός αποτελεί η εύρεση των πιο ενεργών περιοδικών αντί για συγγραφείς. Το παρακάτω query θα μας επιστρέψει τα 10 πιο ενεργά περιοδικά με βάση τον αριθμό των άρθρων που έχουν δημοσιεύσει το 2020. Να υπενθυμίσουμε πως τα περιοδικά συσχετίζονται μόνο με άρθρα (σχ. 5.1) και όχι με άλλες δημοσιεύσεις. Για αυτό το λόγο στο query ψάχνουμε μόνο για κόμβους που ανήκουν στην οντότητα ”article” και όχι οποιουδήποτε τύπου δημοσίευσης. Το query θα δούλευε ακόμη και αν δεν κάναμε αυτή τη διευκρίνιση απλά η περάτωση του θα έπαιρνε παραπάνω ώρα, καθώς θα εξετάζαμε τους κόμβους όλων των δημοσιεύσεων.

```
1  MATCH (a:journal)-[:INCLUDES]-(b:article {year:'2020'})
2  RETURN a.name, count(b) AS articles
3  ORDER BY articles DESC
4  LIMIT 10
```

Τέλος έχουμε την υλοποίηση για τους πιο ενεργούς εκδότες. Οι εκδότες σχετίζονται με 4 είδη δημοσιεύσεων (σχ. 5.1). Το παρακάτω query θα μας επιστρέψει τους 10 πιο ενεργούς εκδότες με βάση τον αριθμό των εκδόσεων που έχουν κάνει το 2020.

```
MATCH (a:publisher)-[:PUBLISHED]-(b {year:'2020'})
RETURN a.name, count(b) AS publications
ORDER BY publications DESC
LIMIT 10
```

Οι μεταβλητές που μπορεί να αλλάξει ο χρήστης και επηρεάζουν τους αλγορίθμους που αναφέρθηκαν είναι οι εξής:

1. mostActiveAuthors: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο ενεργών συγγραφέων.
2. mostActiveJournals: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο ενεργών περιοδικών.
3. mostActivePublishers: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο ενεργών εκδοτών.
4. enableYear: ενεργοποιεί/απενεργοποιεί την διασαφήνιση της χρονολογίας στους 3 αλγορίθμους.
5. mostActiveInYear: η χρονολογία στην οποία θέλουμε να βρούμε τους πιο ενεργούς συγγραφείς/περιοδικά/εκδότες.
6. NUMBER_OF_RESULTS : ο αριθμός των επιστρεφόμενων αποτελεσμάτων των query. Ουσιαστικά ο αριθμός μετά από το LIMIT.

5.3.3 Προβολή των πιο συνηθισμένων λέξεων-κλειδιών

Η εύρεση των πιο συνηθισμένων λέξεων-κλειδιών είναι μία διαδικασία που πραγματοποιείται κατά τη δημιουργία της βάσης αφού εισαχθούν όλα τα δεδομένα. Η λογική της εξηγήθηκε στην ενότητα 5.2.2. Σκοπός του αλγορίθμου είναι η προβολή ενός συνόλου από αυτές στο χρήστη. Η εκτέλεση του συγκεκριμένου query θα μας επιστρέψει τις 10 πιο συχνά χρησιμοποιούμενες λέξεις-κλειδιά.

```
MATCH (a:keyword)
RETURN a.keyword,a.timesFound AS timesFound
ORDER BY timesFound DESC
LIMIT 10
```

Οι μεταβλητές που επηρεάζουν τον συγκεκριμένο αλγόριθμο είναι οι ακόλουθες:

1. showMostUsedKeywords: ενεργοποιεί/απενεργοποιεί την προβολή των πιο συνηθισμένων λέξεων-κλειδιών.
2. NUMBER_OF_RESULTS: αποτελεί των αριθμό των λέξεων-κλειδιών που θέλουμε να λάβουμε.

5.3.4 Εύρεση των δημοσιεύσεων που περιέχουν μία λέξη-κλειδί

Σκοπός αυτού του αλγορίθμου είναι η εύρεση όλων των δημοσιεύσεων που περιέχουν μία λέξη-κλειδί και η προβολή ορισμένων βασικών πεδίων τους (φαίνονται παραπάνω) στον χρήστη. Για παράδειγμα με την εκτέλεση του query παρακάτω θα μας επιστραφούν 10 δημοσιεύσεις που περιέχουν στον τίτλο τους τη λέξη ”crystal”. Η εντολή labels επιστρέφει το όνομα της οντότητας στην οποία ανήκει ένας κόμβος. Στην συγκεκριμένη περίπτωση θα επιστρέψει τον τύπο της δημοσίευσης (άρθρο, διδακτορική διατριβή κ.λπ.) κάτι που μπορεί στη συνέχεια να προβληθεί από τον χρήστη.

```
MATCH (a)-[:IS]-(b:publication)
WHERE toLower(a.title) CONTAINS 'crystal'
OR toLower(a.title) CONTAINS 'crystals'
RETURN a.key, a.title, a.year, labels(a), a.ee, a.school
LIMIT 10
```

Έπειτα με βάση τον τύπο της δημοσίευσης εκτελείται το παρακάτω κομμάτι κώδικα το οποίο μας επιστρέφει το περιοδικό για δημοσιεύσεις τύπου article και τον εκδότη για δημοσιεύσεις τύπου article, book, proceedings, phdthesis. Η συνάρτηση selectQuery εκτελεί ένα query επιστροφής δεδομένων, ενώ η printIfNotNull τυπώνει κάτι μόνο εφόσον δεν είναι null.

```
if (type.equals(”article”)) {
   ResultSet rs2 = selectQuery(conn,”match (a:journal)-[:INCLUDES]-(b:
       article {key: '”+key+”'}) return a.name”);
   if (rs2.next()) {
      String journal = rs2.getString(”a.name”);
      printIfNotNull(”Journal: ”,journal);
   }
}

if (type.equals(”article”) || type.equals(”book”) || type.equals(”
    proceedings”) || type.equals(”phdthesis”)) {
   ResultSet rs2 = selectQuery(conn,”match (a:publisher)-[:PUBLISHED]-(b {
       key: '”+key+”'}) return a.name”);
   if (rs2.next()) {
      String publisher = rs2.getString(”a.name”);
      printIfNotNull(”Publisher: ”,publisher);
   }
}
```

Σχετικές μεταβλητές αποτελούν οι εξής:

1. searchByKeyword: ενεργοποιεί/απενεργοποιεί την εύρεση των δημοσιεύσεων που περιέχουν μία λέξη-κλειδί

2. keywordToSearch: η λέξη-κλειδί που θέλουμε να αναζητήσουμε.

3. NUMBER_OF_RESULTS : ο αριθμός των επιστρεφόμενων αποτελεσμάτων του query.

5.3.5 Προβολή πληροφοριών δημοσίευσης

Ο αλγόριθμος αυτής της ενότητας επικεντρώνεται στην εύρεση όλων των στοιχείων που χαρακτηρίζουν μία δημοσίευση και την προβολή τους στον χρήστη. Η αναζήτηση γίνεται με βάση τον τίτλο της δημοσίευσης ή μέρος του τίτλου της δημοσίευσης. Παρατηρεί κανείς πως στο return μέρος του παραπάνω query επιστρέφουμε όλα τα πιθανά πεδία που μπορεί να περιέχει μία δημοσίευση που έχουμε αποθηκεύσει στη Neo4j. Παρ' όλα αυτά για κάθε δημοσίευση προβάλλονται στο χρήστη μόνο τα πεδία που εμπεριέχουν κάποια τιμή, που δηλαδή δεν είναι null. Το περιοδικό ή ο εκδότης της δημοσίευσης λαμβάνονται όπως και στην ενότητα 5.3.4 και προβάλλονται με την προϋπόθεση ότι υπάρχουν. Εάν ο τίτλος που επιλέξαμε να αναζητήσουμε βρεθεί περισσότερες από μία φορές τότε προβάλλονται όλες οι δημοσιεύσεις με την σειρά. Το παρακάτω παράδειγμα μας επιστρέφει όλους τους συγγραφείς της δημοσίευσης με τίτλο "Binary bat algorithm" μαζί με όλα τα επιπλέον πεδία.

```
1  MATCH (a:author)-[:WROTE]-(b)
2  WHERE toLower(b.title) CONTAINS 'Binary bat algorithm'
3  RETURN a.name, b.key, b.title, labels(b), b.year, b.month, b.number
   , b.volume, b.booktitle, b.series, b.pages, b.isbn, b.ee
4  ORDER BY b.key
```

Εκτός αυτών ο χρήστης μπορεί να επιλέξει και αν θέλει να ανοίξει τον σύνδεσμο παγκοσμίου ιστού (ee) των επιστρεφόμενων δημοσιεύσεων με το προκαθορισμένο πρόγραμμα περιήγησης. Αν ο σύνδεσμος δεν υπάρχει τότε δεν γίνεται κάποια προσπάθεια ανοίγματος. Η διαδικασία φαίνεται παρακάτω.

```
1 if (!(ee == null) && openUrl) {
2    if (Desktop.isDesktopSupported() && Desktop.getDesktop().isSupported(
         Desktop.Action.BROWSE)) {
3       Desktop.getDesktop().browse(new URI(ee));
4    }
5 }
```

Οι σχετικές μεταβλητές που αφορούν την προβολή πληροφοριών δημοσίευσης είναι οι εξής:

1. findDataOfPublication: ενεργοποιεί/απενεργοποιεί την προβολή πληροφοριών μιας δημοσίευσης.
2. openUrl: ενεργοποιεί/απενεργοποιεί το άνοιγμα του συνδέσμου της δημοσίευσης.
3. dataOfPublication: ο τίτλος η μέρος του τίτλου της δημοσίευσης που θέλουμε να αναζητήσουμε.

5.3.6 Προβολή δημοσιεύσεων συγγραφέα

Ο συγκεκριμένος αλγόριθμος έχει ως στόχο την προβολή κάθε δημοσίευσης στην οποία πήρε μέρος ένας συγγραφέας. Η αναζήτηση πραγματοποιείται με βάση το όνομα του συγγραφέα. Τα στοιχεία των δημοσιεύσεων που προβάλλονται στον χρήστη φαίνονται παραπάνω. Αν εκτελέσουμε το παράδειγμα παρακάτω θα μας επιστραφούν όλες οι δημοσιεύσεις του συγγραφέα "E.

F. Codd". Το περιοδικό ή ο εκδότης της κάθε δημοσίευσης λαμβάνονται όπως και στην ενότητα 5.3.4 και προβάλλονται με την προϋπόθεση ότι υπάρχουν. Σε περίπτωση όπου δύο ή περισσότεροι συγγραφείς μοιράζονται το ίδιο όνομα, τότε προβάλλονται όλες οι δημοσιεύσεις του κάθε συγγραφέα στη σειρά. Σε αυτό το query δεν υπάρχει κάποιο όριο στον αριθμό των αποτελεσμάτων.

```
1  MATCH (a:author)-[:WROTE]-(b)
2  WHERE toLower(a.name) CONTAINS 'E. F. Codd'
3  RETURN id(a), b.key, b.title, b.year, labels(b), b.ee, b.school
4  ORDER BY id(a)
```

Οι μεταβλητές που χρησιμοποιούνται από αυτόν τον αλγόριθμο είναι οι ακόλουθες:

1. findPublicationsOfAuthor: ενεργοποιεί/απενεργοποιεί την εύρεση των δημοσιεύσεων ενός συγγραφέα.

2. publicationsOfAuthor: το όνομα με βάση το οποίο γίνεται η αναζήτηση.

5.3.7 Εύρεση των πιο κοντινών φορέων ενός συγγραφέα

Σε αυτή την ενότητα θα αναφερθούμε σε μια σειρά από αλγορίθμους που έχουν ως σκοπό την εύρεση των πιο κοντινών φορέων σε κάποιο συγγραφέα. Με τον όρο πιο κοντινοί φορείς, εννοούμε την εύρεση των πιο κοντινών συγγραφέων, περιοδικών και εκδοτών με βάση τις δημοσιεύσεις του συγγραφέα.

Ο 1ος αλγόριθμος αφορά την εύρεση των πιο κοντινών συγγραφέων ενός συγγραφέα. Με άλλα λόγια πρόκειται για τους συν-συγγραφείς του αναφερόμενου συγγραφέα με τους οποίους έχει δουλέψει τις περισσότερες φορές. Το query που φαίνεται παρακάτω μας επιστρέφει όλους τους συν-συγγραφείς ενός συγκεκριμένου συγγραφέα, στην προκειμένη περίπτωση του "E. F. Codd". Στη συνέχεια καταμετρούμε τις φορές που εμφανίζεται ο κάθε συν-συγγραφέας και τους ταξινομούμε σε φθίνουσα σειρά με τη βοήθεια μιας συνάρτησης συγκρίσεων που βρίσκεται στο αρχείο "AuthorRating.java". Εάν συναντήσουμε περισσότερους από έναν συγγραφείς με το ίδιο όνομα τότε οι λίστες που τους αντιστοιχούν προβάλλονται σε σειρά με βάση το id που έχουν στη βάση. Τα ids παρέχονται αυτόματα από την Neo4j και είναι ακέραιοι αριθμοί.

```
1  MATCH (a:author)-[:WROTE]-(b)-[:WROTE]-(c:author)
2  WHERE toLower(a.name) CONTAINS 'E. F. Codd'
3  RETURN id(a), c.name
4  ORDER BY id(a)
```

Επιπλέον να αναφερθεί πως για κάθε συν-συγγραφέα βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργός μέσω των δημοσιεύσεων στις οποίες συμμετείχε. Αυτή προβάλλεται ως μία επιπρόσθετη πληροφορία στο χρήστη. Η υλοποίηση αυτής της διαδικασίας φαίνεται παρακάτω.

```
1  MATCH (b:author {name:'Al Kim'})-[:WROTE]-(n)
2  RETURN DISTINCT n.year
3  ORDER BY n.year DESC
```

Ο 2ος αλγόριθμος αφορά την εύρεση των πιο κοντινών περιοδικών ενός συγγραφέα. Για να επιφέρει αποτέλεσμα αυτός ο αλγόριθμος πρέπει ο αναφερόμενος συγγραφέας να έχει συμμετάσχει στην σύνταξη ενός αριθμού άρθρων, διότι τα περιοδικά συσχετίζονται μόνο με άρθρα. Όμοια με πριν καταμετρούμε τα διάφορα περιοδικά και τα κατατάσσουμε ανάλογα με τις φορές που εμφανίζονται. Οι συγκρίσεις γίνονται με τη βοήθεια μιας συνάρτησης που βρίσκεται στο αρχείο ”JournalRating.java”. Εάν συναντήσουμε περισσότερους από έναν συγγραφείς με το ίδιο όνομα τότε οι λίστες που τους αντιστοιχούν προβάλλονται σε σειρά με βάση το id που έχουν στη βάση.

```
MATCH (a:author)-[:WROTE]-(b:article)-[:INCLUDES]-(c:journal)
WHERE toLower(a.name) CONTAINS 'E. F. Codd'
RETURN id(a), c.name
ORDER BY id(a)
```

Όπως και πριν για κάθε περιοδικό βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργό μέσω των άρθρων που δημοσίευσε. Αυτή προβάλλεται ως μία επιπρόσθετη πληροφορία στο χρήστη. Η υλοποίηση αυτής της λειτουργικότητας φαίνεται παρακάτω.

```
MATCH (a:journal {name:'IJSSE'})-[:INCLUDES]-(n:article)
RETURN DISTINCT n.year
ORDER BY n.year DESC
```

Ο 3ος και τελευταίος αλγόριθμος που περιγράφεται σε αυτή την ενότητα είναι αυτός που βρίσκει τους πιο κοντινούς εκδότες ενός συγγραφέα. Η διαδικασία είναι παρόμοια. Βρίσκουμε τους πιο κοντινούς εκδότες και τους κατατάσσουμε βάσει τις φορές που εμφανίζονται. Οι συγκρίσεις γίνονται με τη βοήθεια μιας συνάρτησης που βρίσκεται στο αρχείο ”PublisherRating.java”. Εάν συναντήσουμε περισσότερους από έναν συγγραφείς με το ίδιο όνομα τότε οι λίστες που τους αντιστοιχούν προβάλλονται σε σειρά με βάση το id που έχουν στη βάση.

```
MATCH (a:author)-[:WROTE]-(b)-[:PUBLISHED]-(c:publisher)
WHERE toLower(a.name) CONTAINS 'E. F. Codd'
RETURN id(a), c.name
ORDER BY id(a)
```

Τέλος για κάθε εκδότη βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργός με βάση τις δημοσιεύσεις οι οποίες εκδόθηκαν από αυτόν. Αυτή προβάλλεται ως μία επιπρόσθετη πληροφορία στο χρήστη. Η υλοποίηση αυτής της λειτουργικότητας φαίνεται παρακάτω.

```
MATCH (a:publisher {name:'Springer'})-[:PUBLISHED]-(n)
RETURN DISTINCT n.year
ORDER BY n.year DESC
```

Οι μεταβλητές οι οποίες είναι σχετικές με τους αλγορίθμους που περιγράφονται σε αυτή την ενότητα είναι οι εξής:

1. findClosestAuthorsToAuthor: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο κοντινών συγγραφέων ενός συγγραφέα.

2. authorsCloseToAuthor: το όνομα με βάση το οποίο γίνεται η αναζήτηση των πιο κοντινών συγγραφέων.

3. findClosestJournalsToAuthor: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο κοντινών περιοδικών ενός συγγραφέα.
4. journalsCloseToAuthor: το όνομα με βάση το οποίο γίνεται η αναζήτηση των πιο κοντινών περιοδικών.
5. findClosestPublishersToAuthor: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο κοντινών εκδοτών ενός συγγραφέα.
6. publishersCloseToAuthor: το όνομα με βάση το οποίο γίνεται η αναζήτηση των πιο κοντινών εκδοτών.
7. ARRAY_LENGTH: το μέγεθος των λιστών που περιέχουν τους πιο κοντινούς φορείς ενός συγγραφέα (και οι 3 λίστες έχουν το ίδιο μέγεθος).
8. NUMBER_OF_RESULTS: ο αριθμός των φορέων που θέλουμε να προβληθούν. (π.χ. οι λίστες μπορεί να περιέχουν 100 φορείς η καθεμιά αλλά εμείς να θέλουμε να προβάλουμε μόνο τους 10 πρώτους)

5.3.8 Εύρεση των πιο συνηθισμένων λέξεων-κλειδιών ενός συγγραφέα

Η διαδικασία είναι η ίδια με αυτή της ενότητας 5.2.2, απλά αντί να εξετάζουμε πόσες φορές υπάρχει μία λέξη-κλειδί στους τίτλους όλων των δημοσιεύσεων που απαρτίζουν τη βάση, επικεντρωνόμαστε μόνο στις δημοσιεύσεις του αναφερόμενου συγγραφέα. Βασικός στόχος του συγκεκριμένου αλγορίθμου είναι η εξαγωγή των βασικών θεμάτων ενασχόλησης ενός συγγραφέα. Αν έχουμε περίπτωση κοινού ονόματος τότε οι πιο συνηθισμένες λέξεις-κλειδία του κάθε συγγραφέα προβάλλονται διαδοχικά με βάση το id του. Υπενθυμίζεται πως το id είναι ένα πεδίο το οποίο η Neo4j ορίζει αυτόματα και δεν δημιουργείται από εμάς. Για κάθε λέξη-κλειδί με μήκος μεγαλύτερο του 4 εκτελείται ο παρακάτω κώδικας. Φυσικά στη θέση του database management βρίσκεται η αντίστοιχη λέξη-κλειδί. Ομοίως με το όνομα του συγγραφέα.

```
1  MATCH (a:author {name:'E. F. Codd'})-[WROTE]-(b)
2  WHERE toLower(b.title) CONTAINS 'database management'
3  OR toLower(b.title) CONTAINS 'database managements'
4  RETURN count(b) AS timesFound
```

Οι μεταβλητές που σχετίζονται με την εύρεση των πιο συνηθισμένων λέξεων-κλειδιών ενός συγγραφέα είναι οι ακόλουθες:

1. findMostUsedKeywordsOfAuthor: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο συνηθισμένων λέξεων-κλειδιών ενός συγγραφέα.
2. keywordsOfAuthor: το όνομα με βάση το οποίο εκτελείται η αναζήτηση λέξεων-κλειδιών.
3. MIN_KEYWORD_LENGTH: το ελάχιστο μέγεθος μιας λέξης-κλειδί.
4. ARRAY_LENGTH: το μέγεθος της λίστας που περιέχει τις λέξεις κλειδιά.
5. NUMBER_OF_RESULTS: ο αριθμός των λέξεων-κλειδιών που θέλουμε να προβληθούν.

5.3.9 Εύρεση των πιο σχετικών φορέων σε σύνολο από λέξεις-κλειδιά

Στην τελευταία ενότητα αυτού του κεφαλαίου θα αναλύσουμε μια σειρά από αλγορίθμους που έχουν ως σκοπό την εύρεση των πιο σχετικών φορέων σε ένα σύνολο από λέξεις-κλειδιά. Με τον όρο πιο σχετικοί φορείς, εννοούμε την εύρεση των πιο σχετικών δημοσιεύσεων, συγγραφέων, περιοδικών και εκδοτών σε μια ακολουθία από λέξεις-κλειδιά που ο χρήστης επιλέγει. Η χρησιμότητα των συγκεκριμένων αλγορίθμων είναι αρκετά μεγάλη, αφού μπορούμε μέσα από ένα σύνολο από λέξεις κλειδιά να βρούμε τους πιο σχετικούς συγγραφείς πάνω σε ένα θέμα με τους οποίους θα μπορούσαμε να συνεργαστούμε ή τα πιο σχετικά περιοδικά/εκδότες στα οποία μπορούμε να δημοσιεύσουμε την εργασία μας.

Αρχικά να αναφέρουμε πως δίνεται στον χρήστη η επιλογή να αφαιρέσει από τη λίστα των λέξεων-κλειδιών τις πιο συνηθισμένες λέξεις της επιστημονικής κοινότητας. Αυτή η επιλογή προτείνεται, διότι λέξεις όπως ”system” ή ”database” υπάρχουν πολλές φορές στη βάση και αν τις λάβουμε υπόψη θα καθυστερήσουν αρκετά την περάτωση του αλγορίθμου. Εκτός αυτού δεν προσφέρουν και αρκετά στην ποιότητα των αποτελεσμάτων καθώς η ύπαρξη τους σε χιλιάδες τίτλους δεν βοηθά στην εύρεση των πιο σχετικών φορέων πάνω σε ένα θέμα.

Ο 1ος αλγόριθμος αφορά την εύρεση των πιο σχετικών δημοσιεύσεων στο σύνολο από λέξεις-κλειδιά. Η εκτέλεση του είναι απαραίτητη για την εκτέλεση των υπολοίπων που αναλύονται σε αυτή την ενότητα. Για κάθε τίτλο που υπάρχει στη βάση δεδομένων ελέγχουμε πόσες από τις λέξεις-κλειδιά υπάρχουν σε αυτόν. Ανάλογα με το πόσες υπάρχουν η κάθε δημοσίευση αποκτά μία βαθμολογία. Οι λέξεις-κλειδιά έχουν μία αξία ανάλογη με το πόσες λέξεις περιέχουν (μία λέξη-κλειδί μπορεί να είναι μια ολόκληρη φράση). Για παράδειγμα η λέξη-κλειδί ”communications” έχει αξία 1, ενώ η λέξη-κλειδί ”shared databases” έχει αξία 2 κ.ο.κ. Αν ένας τίτλος περιέχει π.χ. 3 λέξεις-κλειδιά η βαθμολογία της δημοσίευσης θα ήταν το άθροισμα των αξιών των λέξεων-κλειδιών που περιέχει. Στη συνέχεια όλες οι σχετικές δημοσιεύσεις κατατάσσονται σε μία λίστα με βάση τη βαθμολογία τους. Στις συγκρίσεις βοηθά μία συνάρτηση συγκρίσεων που έχει υλοποιηθεί στο αρχείο ”PublicationRating.java”. Για κάθε λέξη-κλειδί εκτελείται το παρακάτω query. Η αναφερόμενη λέξη-κλειδί θα βρισκόταν στη θέση της λέξης ”simulation system”.

```
1  MATCH (a)-[:IS]-(c:publication)
2  WHERE toLower(a.title) CONTAINS 'simulation system'
3  OR toLower(a.title) CONTAINS 'simulation systems'
4  RETURN a.key, a.title, a.year
```

Ο 2ος αλγόριθμος χρησιμοποιεί την λίστα από τις πιο σχετικές δημοσιεύσεις για να βρει τα πιο σχετικά περιοδικά. Πάλι χρησιμοποιούμε ένα σύστημα βαθμολόγησης για να κατατάξουμε τα περιοδικά. Για παράδειγμα αν ένα περιοδικό έχει 2 δημοσιεύσεις στην λίστα οι οποίες έχουν με τη σειρά τους βαθμολογία 2 και 3 αντίστοιχα, η βαθμολογία του περιοδικού σε σχέση με το συγκεκριμένο σύνολο λέξεων-κλειδιών θα είναι 5. Όμοίως με την ενότητα 5.3.7 η κατάταξη γίνεται με τη βοήθεια μιας συνάρτησης που βρίσκεται στο αρχείο ”JournalRating.java”. Για κάθε δημοσίευση που υπάρχει στην λίστα εκτελείται το παρακάτω query για να δούμε με πιο περιοδικό (αν υπάρχει) συσχετίζεται. Επιπλέον βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργό το περιοδικό ακριβώς όπως και στην ενότητα 5.3.7.

```
MATCH (a:journal)-[:INCLUDES]-(b:article {key:'DFCOC15'})
RETURN a.name, b.year
```

Ο 3ος αλγόριθμος αφορά την ίδια διαδικασία, απλά αντί για περιοδικά αναφερόμαστε σε εκδότες. Η λογική βαθμολόγησης είναι η ίδια ενώ η κατάταξη των εκδοτών γίνεται με τη βοήθεια μιας συνάρτησης που υπάρχει στο αρχείο ”PublisherRating.java”. Για κάθε δημοσίευση στην λίστα των πιο σχετικών δημοσιεύσεων εκτελείται το παρακάτω query και για κάθε εκδότη που συναντάμε, βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργός όπως στην ενότητα 5.3.7.

```
MATCH (a:publisher)-[:PUBLISHED]-(b {key:'DFCOC15'})
RETURN a.name, b.year
```

Τέλος ο 4ος αλγόριθμος βρίσκει τους πιο σχετικούς συγγραφείς στο σύνολο των λέξεων-κλειδιών. Πάλι χρησιμοποιούμε τη λίστα των πιο σχετικών δημοσιεύσεων για την επίτευξη αυτού του σκοπού. Η λογική βαθμολόγησης είναι η ίδια ενώ η κατάταξη των εκδοτών γίνεται με τη βοήθεια μιας συνάρτησης που υπάρχει στο αρχείο ”AuthorRating.java”. Για κάθε δημοσίευση εκτελείται το παρακάτω query και για κάθε συγγραφέα βρίσκουμε την τελευταία χρονιά στην οποία ήταν ενεργός όπως στην ενότητα 5.3.7.

```
MATCH (a:author)-[:WROTE]-(b {key:'DFCOC15'})
RETURN a.name, b.year
```

Οι μεταβλητές οι οποίες σχετίζονται με τους αλγορίθμους που περιγράφονται σε αυτή την ενότητα είναι οι ακόλουθες:

1. searchBySetOfKeywords: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο σχετικών φορέων σε σύνολο από λέξεις-κλειδιά.

2. removeMostUsedKeywords: ενεργοποιεί/απενεργοποιεί την αφαίρεση των πιο συνηθισμένων λέξεων-κλειδιών από το σύνολο των λέξεων-κλειδιών που έχει εισάγει ο χρήστης.

3. printMostRelevantArticles: ενεργοποιεί/απενεργοποιεί την προβολή των πιο σχετικών δημοσιεύσεων σε ένα σύνολο από λέξεις-κλειδιά (ο 1ος αλγόριθμος αποτελεί αναγκαίο βήμα για την περάτωση των υπολοίπων, οπότε ο χρήστης μπορεί να επιλέξει αν θέλει να δει τα αποτελέσματα του ή όχι).

4. keywords: περιέχει το σύνολο των λέξεων-κλειδιών με οποίο εκτελούνται οι αλγόριθμοι (πρόκειται για ένα πίνακα από γραμματοσειρές).

5. findBestJournalForArticle: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο σχετικών περιοδικών στο σύνολο από λέξεις-κλειδιά.

6. findBestPublisherForPublication: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο σχετικών εκδοτών στο σύνολο από λέξεις-κλειδιά.

7. findBestAuthorForArticle: ενεργοποιεί/απενεργοποιεί την εύρεση των πιο σχετικών συγγραφέων στο σύνολο από λέξεις-κλειδιά.

8. ARRAY_LENGTH: το μέγεθος των λιστών που περιέχουν τους αναφερόμενους φορείς. Η λίστα των συγγραφέων είναι τριπλάσια σε μέγεθος, επειδή μια δημοσίευση μπορεί να περιέχει έναν η περισσότερους συγγραφείς (πάλι είναι πιθανό να μην χωράνε όλοι στη λίστα, αλλά κάτι τέτοιο δεν είναι αναγκαίο αφού ψάχνουμε τους πιο σχετικούς).

9. NUMBER_OF_RESULTS: ο αριθμός των πιο σχετικών φορέων που θέλουμε να προβληθούν.

10. MIN_KEYWORD_LENGTH: το ελάχιστο μέγεθος μιας λέξης-κλειδί η οποία λαμβάνεται υπόψη.

Κεφάλαιο 6

Πειραματική Αξιολόγηση

Σε αυτό το κεφάλαιο αρχικά θα παρουσιαστεί μία πειραματική αξιολόγηση και ο έλεγχος της σωστής λειτουργίας του αλγορίθμου. Για το σκοπό αυτό θα χρησιμοποιήσουμε ένα μικρό σύνολο δεδομένων που θα διευκολύνει την κατανόηση των αποτελεσμάτων. Θα παραθέσουμε τα αποτελέσματα από τα διάφορα queries που αναλύσαμε στο Κεφάλαιο 5 και στη συνέχεια θα επεξηγήσουμε την ορθότητά τους. Έπειτα, αφού κατανοήσουμε την λειτουργία των αλγορίθμων, θα δούμε και την απόδοσή τους για ένα μεγάλο αριθμό δεδομένων. Αυτό θα γίνει με ένα πίνακα όπου για κάθε αλγόριθμο θα φαίνεται ο χρόνος εκτέλεσης.

6.1 Σύνολο δεδομένων

Τα δεδομένα προέρχονται από ένα μικρό αρχείο xml το οποίο μπορούμε να δούμε παρακάτω. Στον ίδιο φάκελο με το xml υπήρχε και το αναγκαίο αρχείο DTD που περιγράφει τη δομή του xml. Όπως μπορούμε να δούμε στο xml εμπεριέχονται 7 δημοσιεύσεις, 3 άρθρα, 2 βιβλία, 1 μέρος μονογραφίας και 1 μεταπτυχιακή εργασία καθώς και όλα τα στοιχεία που τις περιγράφουν. Να σημειωθεί πως τα δεδομένα είναι φανταστικά, δημιουργήθηκαν με σκοπό την πραγματοποίηση του πειράματος και δεν ισχύουν στην πραγματικότητα. Επιπλέον κατά τη δημιουργία της βάσης με το ”CreateDatbase.java” επιλέγουμε να εισάγουμε μόνο 3 από τις πιο συνηθισμένες λέξεις κλειδιά με την διαδικασία που περιγράφεται στην ενότητα 5.2.2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
<article mdate="2017-06-08" key="tr/ucb/erl-m79-28">
   <author>Michael Stonebraker</author>
   <author>Juan Carlos Augusto</author>
   <title>Communication Networks with A Distributed Database System</title>
   <journal>University of California at Berkeley</journal>
   <volume>UCB/ERL M79/28</volume>
   <month>May</month>
   <year>2004</year>
</article>
   <article mdate="2017-06-08" key="tr/ibm/IWBS154">
```

```
    <author>Albert Maier</author>
    <author>Michael Ley</author>
    <author>Thomas Ludwig</author>
    <title>Sort Processing in a Deductive Database System</title>
    <journal>IWBS Report</journal>
    <volume>154</volume>
    <year>2011</year>
    <publisher>IBM Germany Science Center</publisher>
</article>
    <article mdate="2017-06-08" key="tr/ibm/IWBS150">
    <author>Thomas Ludwig 0001</author>
    <author>Bernd Walter</author>
    <title>EFTA: A Database Retrieval Algebra for Feature Terms</title>
    <journal>IWBS Report</journal>
    <volume>150</volume>
    <year>2011</year>
    <publisher>IBM Germany Science Center</publisher>
</article>
<book mdate="2012-03-28" key="reference/assisted/2012">
    <editor>Juan Carlos Augusto</editor>
    <editor>Thomas Ludwig</editor>
    <editor>Achilles Kameas</editor>
    <title>Knowledge Communication with Shared Databases</title>
    <year>2012</year>
    <publisher>IOS Press</publisher>
    <volume>11</volume>
    <isbn>978-1-60750-836-6</isbn>
</book>
<book mdate="2017-05-16" key="reference/vlsi/1999">
    <editor>Achilles Kameas</editor>
    <title>Real Time Database Systems</title>
    <publisher>CRC Press</publisher>
    <year>2012</year>
    <isbn>978-1-4200-4967-1</isbn>
    <ee>https://doi.org/10.1201/9981420049671</ee>
</book>
<incollection mdate="2017-07-12" key="reference/crypt/X11ie">
    <author>Thomas Ludwig</author>
    <author>Juan Carlos Augusto</author>
    <title>Knowledge Communication with Multidimensional Databases</title>
    <pages>808</pages>
    <year>2012</year>
    <booktitle>Encyclopedia of Cryptography and Security</booktitle>
    <ee>https://doi.org/10.1007/978-1-4419-5906-5_4495</ee>
    <crossref>reference/crypt/2011</crossref>
</incollection>
<mastersthesis mdate="2020-03-12" key="ms/Klaas2007">
    <author>Albert Maier</author>
    <title>MLPQ Spatial Constraint Database System</title>
    <year>2012</year>
```

```
64      <school>University of Wisconsin-Madison</school>
65  </mastersthesis>
```

Σχήμα 6.1: Σχήμα της βάσης μετά την εισαγωγή του xml

6.2 Αποτελέσματα αλγορίθμων

Ας δούμε λοιπόν, τα αποτελέσματα των διάφορων αλγορίθμων της εφαρμογής. Η δομή αυτής της ενότητας θα είναι όμοια με αυτή της 5.3, δηλαδή για κάθε αλγόριθμο θα παρατίθενται τα αντίστοιχα αποτελέσματα και θα επεξηγείται σύντομα το πόσο ορθά είναι σε σχέση με τη δομή του xml. Να σημειωθεί πως η μεταβλητή NUMBER_OF_RESULTS ορίζεται σε 10 και η μεταβλητή ARRAY_LENGTH σε 100.

6.2.1 Καταμέτρηση των δεδομένων της βάσης

Παρακάτω μπορούμε να δούμε τον αριθμό των δεδομένων που υπάρχουν στη Neo4j. Τα αποτελέσματα επιβεβαιώνονται και από το σχήμα 6.1.

```
1  Ammount of data in the database:
2  Publications:    7
3  Articles:        3
4  Incollections:   1
5  Inproceedings:   0
6  Books:           2
7  Proceedings:     0
8  PhD theses:      0
```

```
Master's theses: 1
Authors:         8
Journals:        2
Publishers:      3
Keywords:        3
```

6.2.2 Εύρεση των πιο ενεργών μελών

Εδώ βλέπουμε τα πιο ενεργά περιοδικά και τους πιο ενεργούς εκδότες αντίστοιχα. Δεν έχει οριστεί κάποια συγκεκριμένη χρονολογία. Μπορούμε να δούμε από το xml και από το σχήμα 6.1 πως τα αποτελέσματα είναι λογικά.

```
Top 10 most active journals:
Journal                                   Articles
1)IWBS Report                                2
2)University of California at Berkeley      1

Top 10 most active publishers:
Publisher                                Publications
1)IBM Germany Science Center                 2
2)CRC Press                                  1
3)IOS Press                                  1
```

Τα παρακάτω αποτελέσματα είναι οι πιο ενεργοί συγγραφείς του 2012. Οι δημοσιεύσεις που πραγματοποιήθηκαν το 2012 είναι οι 4 τελευταίες του xml αρχείου.

```
Top 10 most active authors in 2012:
Author                                   Publications
1)Juan Carlos Augusto                        2
2)Thomas Ludwig                              2
3)Achilles Kameas                            2
4)Albert Maier                               1
```

6.2.3 Προβολή των πιο συνηθισμένων λέξεων-κλειδιών

Ας δούμε τώρα τις πιο συνηθισμένες λέξεις-κλειδιά που υπολογίστηκαν κατά τη δημιουργία της βάσης. Υπενθυμίζεται πως έχουμε επιλέξει να εισάγουμε μόνο τις 3 κορυφαίες. Μπορούμε να δούμε πως η λέξη ”database” βρέθηκε σε όλες τις δημοσιεύσεις ενώ οι λέξεις ”system” και ”communication” βρέθηκαν 4 και 3 φορές αντίστοιχα.

```
Most used keywords:
Keyword                                  Occurrences
1)database                                   7
2)system                                     4
3)communication                              3
```

6.2.4 Εύρεση των δημοσιεύσεων που περιέχουν μία λέξη-κλειδί

Παρακάτω βλέπουμε όλες τις δημοσιεύσεις που περιέχουν την λέξη-κλειδί ”communication”. Όπως φαίνεται και από το xml πρόκειται για 3 δημοσιεύσεις, ενώ για κάθε μία από αυτές προβάλλονται ορισμένα βασικά στοιχεία.

```
1  Publications containing the keyword ”communication”:
2  ////////////////////////////// 1. //////////////////////////////
3  Type:       Book
4  Title:      Knowledge Communication with Shared Databases
5  Year:       2012
6  Publisher:  IOS Press
7  ////////////////////////////// 2. //////////////////////////////
8  Type:       Incollection
9  Title:      Knowledge Communication with Multidimensional Databases
10 Year:       2012
11 Url:        https://doi.org/10.1007/978-1-4419-5906-5_4495
12 ////////////////////////////// 3. //////////////////////////////
13 Type:       Article
14 Title:      Communication Networks with A Distributed Database
   System
15 Year:       2004
16 Journal:    University of California at Berkeley
17 ////////////////////////////////////////////////////////////////
```

6.2.5 Προβολή πληροφοριών δημοσίευσης

Παρακάτω μπορούμε να δούμε όλες τις πληροφορίες που αφορούν την δημοσίευση με τίτλο ”Knowledge Communication with Shared Databases”.

```
1  Publications with the title ”Knowledge Communication with Shared
   Databases”:
2  ////////////////////////////// 1. //////////////////////////////
3  Type:       Book
4  Title:      Knowledge Communication with Shared Databases
5  Year:       2012
6  Volume:     11
7  Isbn:       978-1-60750-836-6
8  Publisher:  IOS Press
9  Author:     Juan Carlos Augusto
10 Author:     Thomas Ludwig
11 Author:     Achilles Kameas
12 ////////////////////////////////////////////////////////////////
```

6.2.6 Προβολή δημοσιεύσεων συγγραφέα

Εδώ μπορούμε να δούμε όλες τις δημοσιεύσεις στις οποίες συμμετείχε ο συγγραφέας με όνομα ”Thomas Ludwig”. Μπορούμε να παρατηρήσουμε πως υπάρχουν 2 συγγραφείς με το συγκεκριμένο όνομα. Αυτό μπορούμε να το δούμε και στο xml καθώς υπάρχει ο ”Thomas Ludwig” και ο ”Thomas Ludwig 0001” που πρόκειται για διαφορετικά άτομα. Για τον καθένα ξεχωριστά μπορούμε να δούμε τις δημοσιεύσεις που του αναλογούν. Για κάθε δημοσίευση παρουσιάζονται ορισμένα βασικά στοιχεία που την περιγράφουν.

```
1  The author named ”Thomas Ludwig” participated in the writing of
   these publications:
2  ////////////////////////////// 1. //////////////////////////////
3  Type:       Book
4  Title:      Knowledge Communication with Shared Databases
5  Year:       2012
6  Publisher:  IOS Press
7  ////////////////////////////// 2. //////////////////////////////
8  Type:       Incollection
9  Title:      Knowledge Communication with Multidimensional Databases
10 Year:       2012
11 Url:        https://doi.org/10.1007/978-1-4419-5906-5_4495
12 ////////////////////////////// 3. //////////////////////////////
13 Type:       Article
14 Title:      Sort Processing in a Deductive Database System
15 Year:       2011
16 Journal:    IWBS Report
17 Publisher:  IBM Germany Science Center
18 ////////////////////////////////////////////////////////////////
19
20 There are multiple authors named ”Thomas Ludwig”.
21 The author named ”Thomas Ludwig” participated in the writing of
   these publications:
22 ////////////////////////////// 1. //////////////////////////////
23 Type:       Article
24 Title:      EFTA: A Database Retrieval Algebra for Feature Terms
25 Year:       2011
26 Journal:    IWBS Report
27 Publisher:  IBM Germany Science Center
28 ////////////////////////////////////////////////////////////////
```

6.2.7 Εύρεση των πιο κοντινών φορέων ενός συγγραφέα

Παρακάτω φαίνονται οι πιο κοντινοί φορείς του συγγραφέα με όνομα ”Juan Carlos Augusto”. Αρχικά βλέπουμε τους πιο κοντινούς συγγραφείς. Παρατηρούμε πως ο ”Thomas Ludwig” είναι ο μόνος συγγραφέας που έχει συνεργαστεί με τον ”Juan Carlos Augusto” 2 φορές, κάτι που φαίνεται και στο xml. Για αυτό το λόγο έχει rating 2. Στη συνέχεια μπορούμε να δούμε το πιο κοντινό

περιοδικό και τον πιο κοντινό εκδότη αντίστοιχα. Για κάθε φορέα βλέπουμε την τελευταία χρονιά στην οποία ήταν ενεργός.

```
1  The author named "Juan Carlos Augusto" is mostly associated with
   these authors:
2  1)  Name: Thomas Ludwig        ||| Rating: 2  ||| Last Active: 2012
3  2)  Name: Achilles Kameas      ||| Rating: 1  ||| Last Active: 2012
4  3)  Name: Michael Stonebraker  ||| Rating: 1  ||| Last Active: 2004
5
6  The author named "Juan Carlos Augusto" is mostly associated with
   these journals:
7  1)  Name: University of California at Berkeley  ||| Rating: 1 |||
   Last Active: 2004
8
9  The author named "Juan Carlos Augusto" is mostly associated with
   these publishers:
10 1)  Name: IOS Press                          ||| Rating: 1 |||
   Last Active: 2012
```

6.2.8 Εύρεση των πιο συνηθισμένων λέξεων-κλειδιών ενός συγγραφέα

Ας δούμε τώρα τις πιο συνηθισμένες λέξεις-κλειδιά του συγγραφέα με όνομα "Juan Carlos Augusto". Οι πιο συνηθισμένες είναι οι λέξεις "communication" και "database", καθώς υπάρχουν και στις 3 δημοσιεύσεις του συγκεκριμένου συγγραφέα. Υπενθυμίζουμε πως λέξεις με λιγότερα από πέντε γράμματα (στον ενικό) αγνοούνται.

```
1  Most used keywords by author "Juan Carlos Augusto":
2  1)  Keyword: communication        ||| Times found: 3
3  2)  Keyword: database             ||| Times found: 3
4  3)  Keyword: knowledge            ||| Times found: 2
5  4)  Keyword: shared               ||| Times found: 1
6  5)  Keyword: multidimensional     ||| Times found: 1
7  6)  Keyword: network              ||| Times found: 1
8  7)  Keyword: distributed          ||| Times found: 1
9  8)  Keyword: system               ||| Times found: 1
```

6.2.9 Εύρεση των πιο σχετικών φορέων σε σύνολο από λέξεις-κλειδιά

Τέλος ας δούμε τους πιο σχετικούς φορείς σε ένα ορισμένο σύνολο από λέξεις κλειδιά. Οι λέξεις-κλειδιά που χρησιμοποιήθηκαν στην εκτέλεση των αλγορίθμων ήταν οι "Knowledge Communication" και "Shared Databases". Παρατηρούμε πως η πιο σχετική δημοσίευση περιέχει και τις 2 έχοντας βαθμολογία 4. Η βαθμολογία είναι 4 επειδή και οι 2 λέξεις-κλειδιά περιέχουν 2 λέξεις. Αν περιείχαν 3 η αξία τους θα ήταν 3 ξεχωριστά. Έπειτα βλέπουμε πως δεν υπάρχουν σχετικά περιοδικά, καθώς η πρώτη δημοσίευση πρόκειται για βιβλίο, ενώ η δεύτερη για μέρος μονογραφίας και συνεπώς δεν συσχετίζονται με κάποιο περιοδικό. Στη συνέχεια βλέπουμε τον πιο σχετικό εκδότη καθώς συμμετείχε στην έκδοση του βιβλίου με τίτλο "Knowledge Communication with Shared

Databases" και στο τέλος τους πιο σχετικούς συγγραφείς. Οι συγγραφείς "Juan Carlos Augusto" και "Thomas Ludwig" βρίσκονται στην κορυφή της λίστας, με βαθμολογία 6 καθώς συμμετείχαν στη δημιουργία και των 2 πιο σχετικών δημοσιεύσεων.

```
1  Most relevant publications to the set of keywords:
2  1)  Title: Knowledge Communication with Shared Databases
   |||  Rating: 4
3  2)  Title: Knowledge Communication with Multidimensional Databases
   |||  Rating: 2
4
5  Most relevant journals to the set of keywords:
6  No journals were found relevant to these keywords.
7
8  Most relevant publishers to the set of keywords:
9  1)  Name: IOS Press            |||  Rating: 4  |||  Last Active: 2012
10
11 Most relevant authors to the set of keywords:
12 1)  Name: Juan Carlos Augusto  |||  Rating: 6  |||  Last Active: 2012
13 2)  Name: Thomas Ludwig        |||  Rating: 6  |||  Last Active: 2012
14 3)  Name: Achilles Kameas      |||  Rating: 4  |||  Last Active: 2012
```

6.3 Απόδοση αλγορίθμων

Κλείνοντας αυτό το κεφάλαιο θα δούμε την απόδοση όλων των αλγορίθμων για μεγάλο όγκο δεδομένων. Πιο συγκεκριμένα, το dataset που χρησιμοποιήθηκε για την μέτρηση της απόδοσης περιείχε περίπου 100.000 δημοσιεύσεις καθώς και όλα τα στοιχεία που τις περιγράφουν. Να σημειωθεί πως το μηχάνημα που χρησιμοποιήθηκε περιείχε τετραπύρηνο επεξεργαστή και 16GB Ram. Η μεταβλητή NUMBER_OF_RESULTS , που αλλάζει το πόσα αποτελέσματα επιστρέφουν ορισμένοι αλγόριθμοι ορίστηκε σε 10. Οι αλγόριθμοι που επηρεάζονται από αυτή την μεταβλητή φαίνονται στην ενότητα 5.3.

Αλγόριθμος	Χρόνος εκτέλεσης (s)
Καταμέτρηση των δεδομένων της βάσης	0.50
Εύρεση των πιο ενεργών μελών	0.73
Προβολή των πιο συνηθισμένων λέξεων-κλειδιών	0.40
Εύρεση των δημοσιεύσεων που περιέχουν μία λέξη-κλειδί	0.47
Προβολή πληροφοριών δημοσίευσης	0.82
Προβολή δημοσιεύσεων συγγραφέα	1.55
Εύρεση των πιο κοντινών φορέων ενός συγγραφέα	2.87
Εύρεση των πιο συνηθισμένων λέξεων-κλειδιών ενός συγγραφέα	4.65
Εύρεση των πιο σχετικών φορέων σε σύνολο από λέξεις-κλειδιά	6.43

Πίνακας 6.1: Απόδοση αλγορίθμων

Οι παραπάνω χρόνοι εκτέλεσης των αλγορίθμων υπολογίστηκαν από το μέσο όρο των χρόνων που προέκυψαν ύστερα από την εκτέλεση των αλγορίθμων για διαφορετικές εισόδους και για αρκετές επαναλήψεις. Οι συγκεκριμένοι χρόνοι αποτελούν μία προσέγγιση της μέσης περίπτωσης εκτέλεσης του κάθε αλγορίθμου. Μπορούμε να έχουμε αρκετά μεγάλες διακυμάνσεις στους χρόνους ανάλογα με τα δεδομένα που θα εισάγουμε. Για παράδειγμα ο χρόνος εκτέλεσης της προβολής των δημοσιεύσεων ενός συγγραφέα εξαρτάται άμεσα από των αριθμό των δημοσιεύσεων στις οποίες έχει συμμετέχει ο συγγραφέας, ενώ ο χρόνος εκτέλεσης της εύρεσης των πιο σχετικών φορέων σε σύνολο από λέξεις-κλειδιά εξαρτάται από τον αριθμό των λέξεων-κλειδιών που θα εισάγουμε και το πόσο συχνές είναι αυτές.

Κεφάλαιο 7

Οδηγίες εγκατάστασης

Σε αυτό το κεφάλαιο θα αναλύσουμε όλες τις τεχνικές λεπτομέρειες της εφαρμογής. Με άλλα λόγια οτιδήποτε χρειάζεται να κάνει κάποιος για να εγκαταστήσει την εφαρμογή σε ένα σύστημα και να μπορεί να την χρησιμοποιήσει.

7.1 Εγκατάσταση της Neo4j

Πρώτο βήμα αποτελεί η εγκατάσταση της Neo4j. Τη Neo4j μπορούμε να την κατεβάσουμε από τον εξής σύνδεσμο: `https://neo4j.com/download/`. Επιλέγουμε το version που συμβαδίζει με το λειτουργικό σύστημα του υπολογιστή που διαθέτουμε. Η εγκατάσταση είναι αρκετά απλή διαδικασία, απλώς ακολουθάμε τα βήματα που εμφανίζονται. Εάν δεν αλλάξουμε κάτι ο default user θα είναι ο ”neo4j” με κωδικό ”neo4j”. Τον κωδικό του default user ”neo4j” μπορούμε να τον αλλάξουμε την πρώτη φορά που θα συνδεθούμε. Αυτό τον κωδικό θα χρησιμοποιήσουμε και αργότερα για να συνδεθούμε στη βάση μέσω της Java.

7.2 Εγκατάσταση της Java

Μετά την εγκατάσταση της Neo4j απαραίτητη είναι η εγκατάσταση της Java στον υπολογιστή που θα τρέξουμε την εφαρμογή. Για να μπορούμε να τρέξουμε τα αρχεία Java χρειαζόμαστε ένα JDK (Java Development Kit) το οποίο μπορούμε να κατεβάσουμε από τον ακόλουθο σύνδεσμο: `https://www.oracle.com/java/technologies/javase-downloads.html`. Έπειτα το εγκαθιστούμε ανάλογα με το λειτουργικό σύστημα που διαθέτουμε.

7.3 Εγκατάσταση της εφαρμογής

Αρχικά να αναφέρουμε πως τα αρχεία της εφαρμογής είναι διαθέσιμα στην ιστοσελίδα: `https://github.com/ApostolosFotopoulos/Thesis/`.

Ο φάκελος με όνομα ”CreateDatabase” περιέχει το αρχείο Java που δημιουργεί τη βάση, μαζί με όποια άλλα αρχεία χρειάζονται ενώ ο φάκελος ”ExecuteQueries” περιέχει όλα τα αρχεία υπεύθυνα για την πραγματοποίηση ερωτημάτων στη βάση. Τα βασικά αρχεία είναι το ”CreateDatabase

.java" που αφορά την εξαγωγή δεδομένων από την DBLP και την εισαγωγή τους στη Neo4j και το "ExecuteQueries.java" που αφορά την εκτέλεση των ερωτημάτων.

Για τη διαχείριση των αρχείων προτείνεται η χρήση ενός IDE όπως το Netbeans [21] ή το Eclipse [11]. Αν επιλέξουμε να χρησιμοποιήσουμε κάποιο IDE τότε δημιουργούμε 2 διαφορετικά projects με ονόματα "CreateDatabase" και "ExecuteQuerries" και προσθέτουμε τα αντίστοιχα αρχεία Java σε αυτά. Έπειτα είναι απαραίτητο να προσθέσουμε στις βιβλιοθήκες των projects τον driver της Neo4j που μας επιτρέπει να την διαχειριζόμαστε μέσω της Java. Τον συγκεκριμένο driver μπορούμε να τον κατεβάσουμε από αυτό τον σύνδεσμο: `https://github.com/neo4j-contrib/neo4j-jdbc/releases/`. Προτείνεται η έκδοση 3.3.1 καθώς με αυτή έγινε η ανάπτυξη της εφαρμογής.

Εφόσον γίνουν όλα τα παραπάνω είμαστε έτοιμοι να τρέξουμε την εφαρμογή. Το μόνο που χρειαζόμαστε, πλέον, είναι κάποια δεδομένα για να εισάγουμε στη Neo4j. Το xml της DBLP και το σχετικό αρχείο DTD μπορούμε να τα κατεβάσουμε από τον εξής σύνδεσμο: `https://dblp.uni-trier.de/xml/`. Το xml συστήνεται να χωριστεί σε μικρότερα xml αρχεία που είναι σε μέγεθος, ένα με δύο εκατομμύρια γραμμές, διότι ο μέγιστος αριθμός οντοτήτων που επιτρέπει η Java είναι περιορισμένος. Κατά τη δημιουργία της εφαρμογής το μέγιστο αρχείο xml που χρησιμοποιήθηκε ήταν 1 εκατομμύριο γραμμές. Ανάλογα με τις δυνατότητες του μηχανήματος που διαθέτουμε αυτός ο αριθμός μπορεί να αυξηθεί. Τα xml αρχεία τα ονομάζουμε dblp0.xml, dblp1.xml κ.ο.κ.

Εφόσον δημιουργήσουμε τα επιμέρους xml αρχεία τα τοποθετούμε σε ένα φάκελο με όνομα "dblp xmls". Στον φάκελο τοποθετούμε και το αρχείο DTD. Στη συνέχεια τοποθετούμε τον φάκελο μέσα στον φάκελο του project "CreateDatabase". Την παραπάνω διαδικασία μπορούμε να την τροποποιήσουμε όπως θέλουμε αλλάζοντας το παρακάτω κομμάτι κώδικα στο αρχείο "CreateDatabase.java".

```
1 int k=0;
2 File fXmlFile = new File("dblp xmls/dblp"+k+".xml");
3 while(fXmlFile.exists()) {
4    //Insert publications.
5    k++;
6    fXmlFile = new File("dblp xmls/dblp"+k+".xml");
7 }
```

Κεφάλαιο 8

Επίλογος

Στο τελευταίο κεφάλαιο αυτής της διπλωματικής θα συνοψίσουμε τα αποτελέσματα και την προσφορά της και θα περιγράψουμε τα συμπεράσματα που προέκυψαν. Επιπλέον θα αναφέρουμε εν συντομία ορισμένες μελλοντικές επεκτάσεις με σκοπό την βελτίωσή της.

8.1 Σύνοψη και συμπεράσματα

Στη διπλωματική αυτή εργασία αναπτύχθηκε μια εφαρμογή που σκοπεύει στην συγκομιδή δεδομένων που αφορούν την έκδοση δημοσιεύσεων, που σχετίζονται με την επιστήμη των υπολογιστών και την επεξεργασία τους από ορισμένους αλγόριθμους με σκοπό την εξαγωγή διάφορων πληροφοριών.

Αναφέραμε εργασίες παρόμοιες με την παρούσα και επεξηγήσαμε σύντομα το θέμα που πραγματεύονται. Ακόμη αναλύσαμε τη διαδικτυακή βάση DBLP, η οποία αποτελεί και την πηγή δεδομένων της συγκεκριμένης εφαρμογής και είδαμε τον τρόπο με τον οποίο μπορεί κάποιος να εξάγει δεδομένα από αυτή. Επιπλέον είδαμε τα βασικά χαρακτηριστικά και την δομή της Neo4j, η οποία αποτελεί τη βάση στην οποία αποθηκεύουμε τα δεδομένα που εξάγουμε από την DBLP. Αναλύσαμε την γλώσσα πραγματοποίησης ερωτημάτων που χρησιμοποιεί η Neo4j, την Cypher και είδαμε πως μπορούμε να εισάγουμε δεδομένα με χρήση της Java.

Τέλος έγινε λεπτομερής ανάλυση των διάφορων αλγορίθμων της εφαρμογής και των προβλημάτων που επιλύουν. Αυτό έγινε τόσο με την επεξήγηση της λογικής των αλγορίθμων και των query που περιέχουν όσο και με την παρουσίαση ενός παραδείγματος λειτουργίας της εφαρμογής με σκοπό την καλύτερη κατανόηση των αποτελεσμάτων των αλγορίθμων.

Συμπεραίνουμε λοιπόν, πως ο τεράστιος όγκος επιστημονικών δημοσιεύσεων μπορεί να φαντάζει αχανής στην αρχή, παρ' όλα αυτά η οργάνωση του σε μία δομή δεδομένων με συγκεκριμένες οντότητες και συσχετίσεις αποτελεί εφικτή διαδικασία, με αποτέλεσμα να μας παρέχεται η δυνατότητα πραγματοποίησης ποικίλων ερωτημάτων που σκοπεύουν την εξαγωγή πολύτιμων πληροφοριών.

8.2 Μελλοντικές επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί. Παρακάτω θα αναφέρουμε ορισμένους βασικούς τρόπους με τους οποίους κάτι τέτοιο θα μπορούσε να επιτευχθεί.

8.2.1 Αποσαφήνιση συγγραφέων

Ένας τρόπος με τον οποίο θα μπορούσε να βελτιωθεί η εφαρμογή είναι η δημιουργία ενός αλγορίθμου για την αποσαφήνιση των συγγραφέων που υπάρχουν στη βάση εφόσον έχουν εισαχθεί όλα τα δεδομένα σε αυτή. Αυτό αποτελεί και το θέμα της εργασίας που αναφέρθηκε στην ενότητα 2.1. Ένας τέτοιος αλγόριθμος θα επαύξανε σημαντικά την ποιότητα των αποτελεσμάτων σε περιπτώσεις που πολλοί συγγραφείς μοιράζονται ένα συγκεκριμένο όνομα.

8.2.2 Άντληση δεδομένων από περισσότερες πηγές

Εκτός από την DBLP, υπάρχουν και πολλές άλλες διαδικτυακές βάσεις δεδομένων που περιέχουν δημοσιεύσεις σχετικές με την επιστήμη των υπολογιστών. Ορισμένες από αυτές αναφέρθηκαν και αξιολογήθηκαν στην ενότητα 3.5. Αν μπορούσαν να βρεθούν και άλλες πηγές δεδομένων θα βελτιωνόταν η ποιότητα και η γκάμα των αποτελεσμάτων των αλγορίθμων της εφαρμογής.

8.2.3 Μετατροπή σε Web εφαρμογή

Η εφαρμογή όπως είναι δομημένη αυτή τη στιγμή λειτουργεί τοπικά σε όποιο μηχάνημα την εγκαταστήσουμε. Σημαντική βελτίωση θα ήταν η μετατροπή της σε Web εφαρμογή έτσι ώστε ο κάθε χρήστης να μην χρειάζεται να την εγκαθιστά στο προσωπικό του μηχάνημα, αλλά να έχει πρόσβαση σε αυτή μέσω του internet. Για το συγκεκριμένο εγχείρημα θα χρειαζόταν κάποιο server το οποίο θα περιέχει όλα τα δεδομένα της Neo4j και θα πραγματοποιεί τα ερωτήματα που ο κάθε χρήστης επιλέγει. Επιπλέον θα απαιτούνταν η τροποποίηση του κώδικα της εφαρμογής έτσι ώστε να λειτουργεί ως μία Web εφαρμογή.

Βιβλιογραφία

[1] Alf Christian Achilles. The Collection of Computer Science Bibliographies. https://liinwww.ira.uka.de/bibliography/.

[2] ACM. ACM Digital Library. https://dl.acm.org/.

[3] N. Aggrawal και A. Arora. Visualization, analysis and structural pattern infusion of DBLP co-authorship network using Gephi. Στο *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, σελίδες 494–500, 2016. doi: 10.1109/NGCT.2016.7877466.

[4] Donna Bergmark, Paradee Phempoonpanich και Shumin Zhao. Scraping the ACM Digital Library. *SIGIR Forum*, 35:1–7, September 2001. doi: 10.1145/511144.511146.

[5] Gephi Consortium. Gephi Wiki. https://gephi.wordpress.com/tag/database/.

[6] Schloss Dagstuhl Leibniz Centerfor Informatics. DBLP Amount of data. https://dblp.uni-trier.de/faq/1474565.html.

[7] Schloss Dagstuhl Leibniz Centerfor Informatics. DBLP Distribution of publications. https://dblp.uni-trier.de/statistics/distributionofpublicationtype.html.

[8] Schloss Dagstuhl Leibniz Centerfor Informatics. DBLP Information. https://dblp.uni-trier.de/faq/.

[9] Schloss Dagstuhl Leibniz Centerfor Informatics. DBLP License. https://dblp.uni-trier.de/faq/1474677.html.

[10] Schloss Dagstuhl Leibniz Centerfor Informatics. DBLP Publications per year. https://dblp.uni-trier.de/statistics/publicationsperyear.html.

[11] Eclipse foundation. Eclipse Web Page. https://www.eclipse.org/.

[12] V. Franzoni, M. Lepri, Y. Li και A. Milani. Efficient Graph-Based Author Disambiguation by Topological Similarity in DBLP. Στο *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, σελίδες 239–243, 2018. doi: 10.1109/AIKE.2018.00054.

[13] IEEE. IEEE Xplore. https://ieeexplore.ieee.org/Xplore/home.jsp.

[14] Arif E. Jinha. Article 50 million: an estimate of the number of scholarly articles in existence. *Learned Publishing*, 23(3):258–263, July 2010. doi: 10.1087/20100308.

[15] Ware Mark και Mabe Michael. An overview of scientific and scholarly journal publishing. *The STM Report*, March 2015.

[16] Škurla Martin. GSoC 2010 mid-term: Adding support for Neo4j in Gephi. https://gephi.wordpress.com/tag/database/, August 2010.

[17] Mercorio F. Cesarini M. Mezzanzanica, M. GraphDBLP: a system for analysing networks of computer scientists through graph databases. *Multimed Tools Appl*, 77:pages18657–18688, 2018. doi: 10.1007/s11042-017-5503-2.

[18] Chapple Mike. The ACID Database Model. https://www.lifewire.com/the-acid-model-1019731, January 2020.

[19] Neo4j. Graph dbs vs rdbms. https://neo4j.com/developer/graph-db-vs-rdbms/.

[20] Neo4j. How much faster is a graph database, really? https://neo4j.com/news/how-much-faster-is-a-graph-database-really/.

[21] Netbeans. Netbeans Web Page. https://netbeans.org/projects/www/.

[22] Johnson Rob, Mabe Michael και Watkinson Anthony. An overview of scientific and scholarly publishing. *The STM Report*, October 2018.

[23] Margaret Rouse. Debugging. https://searchsoftwarequality.techtarget.com/definition/debugging, December 2019.

[24] N.R. Smalheiser και V.I. Torvik. Author name disambiguation. *Ann. Rev. Info. Sci. Tech.*, 43(1):1–43, 2009. doi: 10.1002/aris.2009.1440430113.

Συντομογραφίες

DBLP	Digital Bibliography & Library Project
ACM	Association for Computing Machinery
IEEE	Institute of Electrical and Electronics Engineers
ACID	Atomic Consistent Isolated Durable
DB	Database
UI	User Interface
SQL	Structured Query Language
JDK	Java Development Kit
IDE	Integrated Development Environment
σχ.	σχήμα
π.χ.	παραδείγματος χάριν
κ.ο.κ	και ούτω καθεξής
κ.λπ.	και λοιπά

Ορολογία - Γλωσσάρι

Ελληνικός όρος	Αγγλικός όρος
αποσαφήνιση συγγραφέων	author name disambiguation
διακομιστής	server
προσθήκη	plug-in
δημοσίευση	publication
άρθρο	article
σεμινάριο	workshop
συνέδριο	conference
εργασία σε συνέδριο	inproceedings
τόμος εργασιών σε συνέδριο	proceedings
τόμος	volume
σειρά	series
βιβλίο	book
μέρος μονογραφίας	incollection
μεταπτυχιακή εργασία	masters thesis
διδακτορική διατριβή	phd thesis
παγκόσμιος ιστός	world wide web
σύνολο δεδομένων	dataset
ερώτημα	query
λογισμικό	software
οντότητα	entity
γλώσσα διατύπωσης ερωτήσεων	query language
μεγάλος όγκος δεδομένων	big data
σχεσιακή βάση δεδομένων	relational database
βάση δεδομένων γράφων	graph database
πηγαίος κώδικας	source code
προγράμματα οδήγησης	drivers
διεπαφή χρήστη	user interface
κονσόλα	console
ξένο κλειδί	foreign key
πρωτεύον κλειδί	primary key
ένωση	join

καθυστέρηση απόκρισης	latency
μη προκαθορισμένο σχήμα	schema-free
συμβολοσειρά	string
κόμβος	node
σχέση	relation
ιδιότητα	property
επιγραφή	label
κανονική επεξεργασία γράφου	native graph processing
κανονική αποθήκευση γράφου	native graph storage
γειτνίαση χωρίς ευρετήριο	index-free adjacency
δείκτης	index
αναπαραγωγή	sharding
διαθεσιμότητα	availability
συνεκτικότητα	consistency
ανοχή διαμερισμού	partition tolerance
αντίγραφο ασφαλείας	back-up
εντοπισμός σφαλμάτων	debugging
λειτουργικό σύστημα	operating system