
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

# Μοντελοποίηση Βιβλιογραφικών Μεγάλων Δεδομένων σε Neo4j

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Κωνσταντίνου Ι. Βαβούρη**

**Επιβλέπων :** Δρ. Σιδηρόπουλος Αντώνιος
Αναπληρωτής Καθηγητής, ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Σεπτέμβριος 2022

.

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

# Μοντελοποίηση Βιβλιογραφικών Μεγάλων Δεδομένων σε Neo4j

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Κωνσταντίνου Ι. Βαβούρη**

**Επιβλέπων :** Δρ. Σιδηρόπουλος Αντώνιος
Αναπληρωτής Καθηγητής ΔΙ.ΠΑ.Ε.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 1 Οκτωβρίου 2022.

| *(Υπογραφή)* | *(Υπογραφή)* | *(Υπογραφή)* |
|---|---|---|
| ..... | ..... | ..... |
| Δρ. Αντώνης Σιδηρόπουλος | Δρ. Αικατερίνη Ασδρέ | Δρ. Παναγιώτης Αδαμίδης |
| Αναπληρωτής Καθηγητής ΔΙ.ΠΑ.Ε. | ΔΙ.ΠΑ.Ε. | Καθηγητής ΔΙ.ΠΑ.Ε. |

Θεσσαλονίκη, Οκτώβριος 2022

*(Υπογραφή)*

.....

**Κωνσταντίνος Βαβούρης**

Πληροφορικής & Τηλεπικοινωνιών Ε.Κ.Π.Α

# Πρόλογος

Η διάδοση του διαδικτύου σε όλες τις πτυχές της ανθρώπινης ύπαρξης έχει ενοποιήσει ψηφιακά ολόκληρο τον πλανήτη. Παράλληλα δημιούργησε μια πλημμύρα δεδομένων με κύρια χαρακτηριστικά τον μεγάλο όγκο, τον καταιγιστικό ρυθμό παραγωγής και την ισχυρή διασύνδεση τους. Η εταιρεία IDC προβλέπει ότι έως το 2023, 49 δισεκατομμύρια συσκευών, καταναλωτών και επιχειρήσεων θα παράγουν 103 ZB νέων δεδομένων[1]. Σε δεδομένα αυτής της κλίμακας έχει παρατηρηθεί ότι μεγαλύτερη αξία έγκειται στον τρόπο διασύνδεσής τους. Η ανακάλυψη μη προφανών συσχετίσεων μεταξύ συνόλων δεδομένων είναι κρίσιμης σημασίας για μια επιχείρηση. Ο εντοπισμός τους μπορεί να πυροδοτήσει την λήψη των κατάλληλων επιχειρηματικών αποφάσεων που θα δημιουργήσουν ανταγωνιστικό πλεονέκτημα. Οι παραδοσιακές σχεσιακές τεχνολογίες αποθήκευσης και επεξεργασίας αδυνατούν να χειριστούν δεδομένα με αυτά τα χαρακτηριστικά. Το κυριότερο μειονέκτημά τους έγκειται στην αδυναμία τους να χειριστούν τόσο δυναμικά, αδόμητα δεδομένα καθώς και να αναπαραστήσουν φυσικά τις μεταξύ τους συσχετίσεις. Η μη σχεσιακή τεχνολογία βάσεων δεδομένων γράφων Neo4j υπόσχεται να αντιμετωπίσει αποτελεσματικά και αποδοτικά αυτήν την κατάσταση. Βασιζόμενη σε μια εντελώς νέα φιλοσοφία αντιμετωπίζει τις συνδέσεις μεταξύ των δεδομένων ως εξίσου σημαντικές με τα ίδια τα δεδομένα. Συνεπώς φροντίζει να τις αποθηκεύει μαζί με τα υπόλοιπα δεδομένα χρησιμοποιώντας ένα ευέλικτο μοντέλο που μπορεί να αναδιαταχθεί δυναμικά καθώς μεταβάλλονται οι ανάγκες του χρήστη. Ουσιαστικά η παρούσα διπλωματική διερευνά τις δυνατότητες της μη σχεσιακής τεχνολογίας βάσεων δεδομένων γράφου Neo4j να διαχειριστεί αποδοτικά και αποτελεσματικά όλα τα προβλήματα που εμφανίζονται σε πολύπλοκες ροές επεξεργασίας πολύ μεγάλων δεδομένων με έμφαση στα βιβλιογραφικά δεδομένα.

# Περίληψη

Σκοπός της διπλωματικής είναι η δημιουργία μιας υποδομής που θα διαχειριστεί αποτελεσματικά και αποδοτικά μεγάλους όγκους βιβλιογραφικών δεδομένων. Τα δεδομένα αυτά προέρχονται από τα σύνολα δεδομένων Mag και Aminer και παρουσιάζουν όλα τα χαρακτηριστικά που αναφέρθηκαν προηγουμένως. Πρόκειται για σύνολα δεδομένων που παράγουν δυο πολύ γνωστοί οργανισμοί συλλογής και καταλογοποίησης της παγκόσμιας βιβλιογραφικής γνώσης. Αρχικά μελετήθηκαν εκτενώς τα δυο σύνολα δεδομένων και εντοπίστηκαν τα χαρακτηριστικά γνωρίσματα που είναι κρίσιμης σημασίας και πρέπει να ενσωματωθούν στην εφαρμογή μας. Επίσης εντοπίστηκαν διάφορα προβλήματα ποιότητας που τα χαρακτηρίζουν και επισημάνθηκαν τρόποι αντιμετώπισης. Τα συμπεράσματα αποκρυσταλλώθηκαν σε ένα αρχικό μοντέλο δεδομένων ιδιοτήτων γράφου. Το μοντέλο αυτό είναι από την φύση του αρκετά ευέλικτο και επιτρέπει την εύκολη αναδιάταξή του (Model Refactoring), ώστε να ανταποκρίνεται με μεγαλύτερη ακρίβεια στις μεταβαλλόμενες ανάγκες των χρηστών και στην πιο αποδοτική εκτέλεση των ερωτημάτων. Η υποδομή απαρτίζεται από τρία διακριτά τμήματα χαμηλής σύζευξης (Low Coupling Components) που επικοινωνούν απρόσκοπτα χωρίς χωρικούς και χρονικούς περιορισμούς. Στο πρώτο τμήμα αναπτύχθηκε ένας ελεύθερης διανομής εξυπηρετητής βάσης δεδομένων γράφου Neo4j σε περιβάλλον Debian-Linux στην υπολογιστική υποδομή του πανεπιστημίου. Σε αυτόν τον εξυπηρετητή τρέχει αδιάλειπτα η Neo4j βάση δεδομένων γράφου, όπου και αποθηκεύονται τα βιβλιογραφικά δεδομένα που υπαγορεύει το μοντέλο δεδομένων. Η φόρτωση των δεδομένων στην βάση δεδομένων γράφου γίνεται μέσω μιας εφαρμογής πελάτη (Client Application) σε γλώσσα προγραμματισμού Python. Στα πλαίσια της διπλωματικής διερευνήθηκαν δυο προσεγγίσεις φόρτωσης της βάσης δεδομένων μέσω μιας εφαρμογής πελάτη. Η πρώτη εισάγει τα δεδομένα άμεσα στην βάση κάνοντας χρήση του επίσημα υποστηριζόμενου από το Neo4j οδηγού της Python. Η δεύτερη εισάγει τα δεδομένα έμμεσα στην βάση κάνοντας χρήση της εντολής import του εργαλείου διαχείρισης neo4j-admin. Τελικά για λόγους απόδοσης επιλέξαμε να χρησιμοποιήσουμε την δεύτερη. Η εφαρμογή πελάτης ουσιαστικά αποτελεί το δεύτερο τμήμα της δημιουργημένης υποδομής και βασίζεται στην βιβλιοθήκη κατανεμημένης και παραλληλοποιημένης εκτέλεσης Dask. Η εφαρμογή πελάτης εκτελείται σειριακά αλλά και παραλληλοποιημένα από την γραμμή εντολών (Script Mode Executionin Command Prompt) ενός εικονικού μηχανήματος του πανεπιστημίου. Στο ίδιο μηχάνημα είναι εγκατεστημένος και ο εξυπηρετητής της βάσης δεδομένων Neo4j. Ο σκοπός της εφαρμογής πελάτη είναι διττός και εκφράζεται μέσω δυο πολύπλοκων σταδίων επεξεργασίας κατά τμήματα (Batch Processing). Στο πρώτο στάδιο εφαρμόζεται μια ροή προεπεξεργασίαστων δεδομένων (Data Preprocessing Pipeline). Το στάδιο αυτό είναι απολύτως απαραίτητο μιας και τα προσφερόμενα δεδομένα είναι διαφορετικής ποιότητας. Συγκεκριμένα περιλαμβάνει πολλά υπό στάδια που αποσκοπούν να φορτώσουν τα δεδομένα από το απομακρυσμένο μέσο αποθήκευσης, να τα φιλτράρουν σύμφωνα με τις προδιαγραφές της εργασίας, να εξάγουν τα απαραίτητα χαρακτηριστικά, να τα ομογενοποιήσουν και τέλος να τα καθαρίσουν. Στο δεύτερο στάδιο η εφαρμογή πελάτης προσπελάζει την βάση γράφου ακολουθώντας δυο διαφορετικές προσεγγίσεις. Στην πρώτη προσπελάζει άμεσα την βάση δεδομένων μέσω του επίσημα υποστηριζόμενου από το Neo4j οδηγού Python. Συγκεκριμένα η εφαρμογή εκτελεί μαζικής κλίμακας λειτουργίες CRUD στη βάση δεδομένων μέσω απομακρυσμένων συνδέσεων με το πρωτόκολλο bolt. Στην δεύτερη προσπελάζει έμμεσα τη βάση δεδομένων μέσω του εργαλείου διαχείρισης neo4j-admin import. Πρακτικά στη δεύτερη προσέγγιση ο ρόλος της εφαρμογής πελάτη είναι η δημιουργία των ειδικής μορφής αρχείων csv, τα οποία σε μεταγενέστερη φάση θα αποτελέσουν την είσοδο του εργαλείου neo4j-admin import. Τέλος στο τρίτο κομμάτι της υποδομής αποθηκεύονται τα ανεπεξέργαστα δεδομένα. Ουσιαστικά πρόκειται για έναν λογαριασμό Google

Drive στον οποίο έχει αγορασθεί επιπλέον αποθηκευτικός χώρος ώστε να καλυφθούν οι αυξημένες αποθηκευτικές ανάγκες της υποδομής. Σε αυτόν έχει δημιουργηθεί μια ιεραρχικά εμφωλευμένη δομή από καταλόγους που αντανακλά τον φυσικό τρόπο οργάνωσης των δεδομένων. Στη συνέχεια η παραπάνω ιεραρχική δομή προσαρτάται στο υπολογιστικό μηχάνημα της σχολής μέσω της εφαρμογής Google Drive Ocamlfuse. Στο σημείο αυτό κρίνεται σκόπιμο να αναφέρουμε ότι η επιλογή της παραπάνω υποδομής έγινε ώστε να μην εμφανίζει χωρικά και χρονικά εμπόδια. Μπορεί να χρησιμοποιηθεί οπουδήποτε, σε οποιαδήποτε χρονική στιγμή και από οποιονδήποτε πιστοποιημένο χρήστη. Επίσης μέσω της βιβλιοθήκης Dask η εφαρμογή είναι ανεξάρτητη και του τεχνολογικού υποστρώματος που θα εκτελεστεί. Μπορεί να εκτελεστεί αυτόματα και με ελάχιστες ρυθμίσεις από τη μεριά του χρήστη σε ο,τιδήποτε υλικό διαθέτει το υπολογιστικό σύστημα που θα επιλεγεί. Η εργασία ισχυρίζεται πως μια τέτοια υποδομή είναι κρίσιμης σημασίας για την οργάνωση δεδομένων που αφορούν συγγραφείς, επιστημονικές εργασίες, μέσα δημοσίευσης των εργασιών και οργανισμούς εργασίας των συγγραφέων. Μπορεί να διευκολύνει την αναζήτηση προϋπάρχουσας επιστημονικής γνώσης, επιστημονικού δυναμικού και μέσων δημοσίευσης για μελλοντικές ερευνητικές συνεργασίες.

**Λέξεις Κλειδιά: βάση δεδομένων γράφου, neo4j, βιβλιογραφικά σύνολα δεδομένων, εφαρμογή πελάτης σε python, dask, google drive, google cloud storage, neo4j community edition server, αλγόριθμοι γράφων, αλγόριθμοι κεντρικότητας, pagerank, αλγόριθμοι ανίχνευσης κοινότητας, αλγόριθμοι ομοιότητας κόμβων, αλγόριθμοι εύρεσης διαδρομής, cypher, παραλληλοποιήμενη και κατανεμημένη εκτέλεση επεξεργασιών, υποδομή διαχείρισης βιβλιογραφικών δεδομένων**

Modeling of Huge Volumes of Bibliographic Data in Neo4j

Konstantinos Vavouris

# Abstract

The purpose of this dissertation is to create an infrastructure that will effectively and efficiently manage large volumes of bibliographic data. These data are derived from the Mag and Aminer datasets and exhibit all the characteristics previously mentioned. These are data sets produced by two well-known organizations for the collection and cataloging of global bibliographic knowledge. Initially, the two data sets were studied extensively and the characteristics that are of critical importance and need to be integrated into our application were identified. Various quality problems that characterize them were also identified and ways of dealing with them were pointed out. The conclusions were crystallized into a graph property data model. This model is by its nature quite flexible and allows easily its refactoring in order to respond more accurately to the changing needs of users and to the more efficient execution of queries. The infrastructure is made up of three distinct low coupling components that communicate seamlessly without spatial and temporal limitations. In the first part, a community edition Neo4j graph database server was installed in a Debian - Linux environment on the university's computing infrastructure. This server runs the Neo4j graph database continuously, where the bibliographic data dictated by the data model are stored. The database is accessed through a client application written in Python programming language. In the context of this thesis, two different approaches were investigated in order to access the database through the client application. The first approach directly accesses the database using the officially supported by Neo4j Python driver. The second approach accesses the database indirectly using the import command of the neo4j-admin administration tool. Finally, for reasons of optimal performance, we chose the second one. The client application is essentially the second part of the created infrastructure and is based on the distributed and parallelized execution Dask library. The client application is executed serially and in parallel from the command line (Script Mode) of the same virtual machine, where the Neo4j database server is also installed. The purpose of the client application is twofold and is expressed through two complex stages of processing in batches. In the first stage, a preprocessing pipeline is applied to the messy data. This stage is absolutely necessary since the data offered by different organizations are of different quality. In particular, it includes several sub-stages that aim to load the data from the remote storage medium, filter it according to the bibliographic domain use case specifications, extract the necessary features, homogenize it and finally clean them. In the second stage, the client application accesses the graph database following two different approaches. In the first, it directly accesses the database through Neo4j's officially supported Python driver. In particular, the application performs large-scale CRUD operations on the database through remote connections over the bolt protocol. In the second it accesses the database indirectly through the administration tool neo4j-admin import. Practically in the second approach the role of the client application is limited just to create the csv files which in a later phase will be the input of the neo4j-admin import tool. Finally, the raw data are stored in the third part of the infrastructure. It is essentially a Google Drive account in which additional storage space has been purchased to meet the infrastructure's increased storage needs.

It has a hierarchically nested directory structure that reflects the natural way that data are organized.The above hierarchical structure is then attached to the university'sserverthrough the Google Drive Ocamlfuse application. At this point, it is critical to mention that the selection of the above infrastructure was made so as not be bounded by spatial and temporal obstacles. It can be used anywhere, at any time and by any authenticated user. Also, through Dask library, the application is independent of the technological substrate that will be executed. It can be performed automatically and with minimal settings by the user on any hardware infrastructure is currently available. The paper claims that such an infrastructure is critical for organizing data related to authors, scholarly papers, publishing venues, and authors' affiliations. It can facilitate the search for pre-existing scientific knowledge, scientific potential and means of publication for future research collaborations.

**Keywords: graph database, neo4j, bibliographic datasets, python client application, dask, google drive, google cloud storage, neo4j community edition server, graph algorithms, centrality algorithms, page rank, community detection algorithms, node Similarity algorithms, link prediction algorithms, path finding algorithms, cypher, parallel and distributed processing, bibliographic datasets processing infrastructure**

# Ευχαριστίες

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα αναπληρωτή καθηγητή Δρ. Σιδηρόπουλο Αντώνιο, για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωση της.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| OAG | Open Academic Graph |
| AMiner | Alpha Miner |
| IMSELab | Information Management & Software Engineering Laboratory |
| API | Application Programming Interface |
| GD | Google Drive |
| GCP | Google Cloud Platform |
| GCS | Google Cloud Storage |
| JSON | JavaScript Object Notation |
| IoT | Internet of Things |
| CRUD | Create – Retrieve – Update - Delete |
| ACID | Atomicity – Consistency – Isolation - Durability |
| SQL | Structured Query Language |
| APOC | Awesome Procedures on Cypher |
| GDS | Graph Data Science |
| CE | Community Edition |
| EE | Enterprise Edition |
| RBACS | Role-Based Access Control System |
| GC | Garbage Collector |
| GIL | Global Interpreter Lock |
| GPU | General Processing Unit |
| AWS | Amazon Web Services |
| SSH | Secure Socket Shell |
| HP | Hewlett-Packard |
| RAM | Random Access Memory |
| DB | Dask Bag |
| DDF | Dask DataFrame |
| IaaS | Infrastructure as a Service |
| GCC | Google Cloud Console |
| IAM | Identity and Access Management |
| REST | Representational State Transfer |

| | |
|---|---|
| SaaS | Software as a Service |
| Gmail | Google Mail |
| ETL | Extract - Transform - Load |
| GRASP | General Responsibility Assignment Software Patterns |
| CUDF | Cuda DataFrame |
| DOI | Digital Object Identifier |
| OS | Operating System |
| JVM | Java Virtual Machine |
| DBMS | DataBase Management System |
| OOPS | Ordinary Object Pointers |
| FGCC | Full Garbage Collection Cycle |
| CSV | Comma Seperated Values |
| URI | Uniform Resource Identifier |
| HTTP | HyperText Transfer Protocol |
| SVG | Scalable Vector Graphics |
| PNG | Portable Network Graphics |
| URL | Uniform Resource Locator |
| HTML | HyperText Markup Language |
| TPU | Tensor Processing Unit |
| TFLOPS | Tera Floating Point Operations Per Second |
| AI | Artificial Intelligence |
| NoSQL | Not Only SQL |
| PaaS | Platform as a Service |
| IP | Internet Protocol |
| PCPUs | Physical CPUs |
| VCPUs | Virtual CPUs |
| MIGs | Managed Instance Groups |
| UIGs | Unmanaged Instance Groups |
| SPARQL | SPARQL Protocol and RDF Query Language |
| RDF | ResourceDescriptionFramework |
| ΣΔΒΔ | Σύστημα Διαχείρισης Βάσεων Δεδομένων |
| IDC | International Data Corporation |
| ZB | Zettabyte |

GCSF Google Conduce Sistem de Fișiere (Google Drive Filesystem είναι η Αγγλική μετάφραση από τα Ρουμάνικα)

# Κεφάλαιο 1ο: Εισαγωγή

## 1.1 Εισαγωγή

Στο κεφάλαιο αυτό επιχειρείται μια συνοπτική επισκόπηση του βιβλιογραφικού τομέα προβλήματος που κληθήκαμε να διαχειριστούμε στα πλαίσια της διπλωματικής εργασίας. Καταβλήθηκε προσπάθεια να καταγραφεί ένας πρώτος βασικός σκελετός εργασίας. Αυτός βασίστηκε τόσο σε ιδέες που προέρχονται από άλλες παρόμοιες εργασίες, όσο και σε προσωπικές επιλογές. Ουσιαστικά στο κεφάλαιο αυτό καθορίστηκε η βασική μεθοδολογία εργασίας.

## 1.2 Περιγραφή συνόλων δεδομένων

Η διπλωματική εργασία βασίζεται στο σύνολο δεδομένων Open Academic Graph - OAG 2.1 το οποίο αποτελεί την συνένωση δυο άλλων μεγάλων ακαδημαϊκών συλλογών: του Microsoft Academic Graph – Mag (https://academic.microsoft.com/) και του AMiner (https://www.aminer.org/). Και τα δυο προσφέρουν υπηρεσίες αναζήτησης και εξόρυξης βιβλιογραφικών δεδομένων από διαφορετικά ετερογενή ακαδημαϊκά και ερευνητικά κοινωνικά δίκτυα. Ενσωματώνουν στις βάσεις δεδομένων γράφου τους τεράστιες συλλογές δεδομένων επιτρέποντας την εύκολη και γρήγορη αναζήτηση ερευνητικών και ακαδημαϊκών πληροφοριών ανά εργασία, συγγραφέα, μέσο δημοσίευσης και οργανισμό εργασίας του συγγραφέα. Επειδή τα σύνολα αυτά εξελίσσονται διαρκώς στον χρόνο, στην συγκεκριμένη έκδοση χρησιμοποιήθηκαν τα στιγμιότυπα που πάρθηκαν τον Ιούλιο (Mag) και τον Οκτώβριο (AMiner) του 2020. Το OAG διαθέτει και αρχεία αντιστοίχισης υψηλής ακρίβειας των εργασιών, συγγραφέων, μέσων δημοσίευσης και οργανισμών των δυο συνόλων δεδομένων που το απαρτίζουν [2]–[4].

## 1.3 Αντικείμενο διπλωματικής

Στόχος της διπλωματικής εργασίας είναι η δημιουργία μιας υποδομής για την αποθήκευση και προσπέλαση πολύ μεγάλων όγκων βιβλιογραφικών δεδομένων σε μια νέας τεχνολογίας μη σχεσιακή βάση δεδομένων γράφου. Μια τέτοια υποδομή μπορεί να αποδειχτεί μελλοντικά εξαιρετικά σημαντική ώστε τα μέλη του ερευνητικού εργαστηρίου IMSELab να τη χρησιμοποιούν ώστε να αντλούν χρήσιμες πληροφορίες για διεξαγωγή έρευνας. Συγκεκριμένα η διπλωματική μελετά την μορφή των ανοιχτών βιβλιογραφικών δεδομένων που είναι διαθέσιμα (Open Academic Graph) και επισημαίνει τις ιδιαίτερες προκλήσεις που παρουσιάζουν. Σε αυτές συγκαταλέγονται ο μεγάλος όγκος, ο καταιγιστικός ρυθμός μεταβολής και η ισχυρή διασύνδεσή τους. Η εργασία διαπιστώνει ότι οι συσχετίσεις μεταξύ των βιβλιογραφικών δεδομένων υποκρύπτουν εξίσου σημαντική αξία που η ανακάλυψη τους μπορεί να δημιουργήσει ισχυρό ανταγωνιστικό πλεονέκτημα στον τελικό χρήστη. Στην συνέχεια συμπεραίνει ότι η παραδοσιακή τεχνολογία των σχεσιακών βάσεων δεδομένων αδυνατεί να ανταποκριθεί σε αυτές τις προκλήσεις. Απαιτείται λοιπόν μιας ριζοσπαστικής φιλοσοφίας τεχνολογία βάσης δεδομένων που δίνει ισοδύναμη αξία στα δεδομένα με τις μεταξύ τους συσχετίσεις. Η εργασία υποστηρίζει ότι μια τέτοια τεχνολογία είναι η μη σχεσιακή βάση δεδομένων γράφου Neo4j. Μια από τις σημαντικότερες αρετές αυτής της τεχνολογίας είναι το ευέλικτο μοντέλο δεδομένων που τη συνοδεύει. Αυτό επιτρέπει την αποδοτική και ακριβή αποθήκευση όχι μόνο των βασικών βιβλιογραφικών οντοτήτων συγγραφέας (Author), επιστημονική εργασία (Paper), μέσο δημοσίευσης (Venue), Οργανισμός εργασίας συγγραφέα (Organization), αλλά και των μεταξύ τους συσχετίσεων δημοσιεύεται (IS_PUBLISHED_IN), αναφέρεται (CITES), συγγραφή (HAS_AUTHOR), εργάζεται (IS_AFFILIATED_AT_ORGANIZATION). Μάλιστα η δυναμική φύση του μοντέλου είναι τέτοια

που επιτρέπει την ριζική αναδιάταξη του ακόμα και κατά το τελικό στάδιο, ώστε να υποστηρίξει αποδοτικότερα την εκτέλεση κρίσιμης σημασίας ερωτημάτων που δεν ήταν δυνατόν να εντοπιστούν κατά τα αρχικά στάδια δημιουργίας του μοντέλου. Η υποδομή που δημιουργήθηκε υποστηρίζει επίσης τη συχνή ενημέρωση της αποθηκευμένης στη βάση δεδομένων βιβλιογραφικής πληροφορίας. Η δυνατότητα αυτή γίνεται αποκλειστικά μέσω αρχείων δεδομένων και όχι με τη χρήση κάποιου απομακρυσμένου API ανά στοιχείο. Η επιλογή αυτή έγινε ώστε να αποφύγουμε τους περιορισμούς χρήσης των πόρων που επιβάλλει σε ένα API ο δημιουργός του και για να επιταχύνουμε την διαδικασία ανάκτησης μεγάλου όγκου δεδομένων.

## 1.4 Σχετικές εργασίες

Έχουν εκπονηθεί πολλές εργασίες σχετικές με τους δυο βασικούς πυλώνες της διπλωματικής εργασίας. Όσον αφορά τον πρώτο πυλώνα που αφορά τις μη σχεσιακές βάσεις δεδομένων γράφων ο Justin J. Miller στην εργασία του με τίτλο “Graph Database Applications and Concepts with Neo4j” [5] συγκρίνει τις σχεσιακές με τις μη σχεσιακές βάσεις δεδομένων γράφων και επισημαίνει τα αδιαμφισβήτητα πλεονεκτήματα της δεύτερης τεχνολογίας στη μοντελοποίηση ισχυρά συνδεδεμένων δεδομένων. Δεδομένα με τα παραπάνω χαρακτηριστικά εμφανίζονται σε τομείς των κοινωνικών δικτύων, της βιοπληροφορικής, της σχεδίασης εφαρμογών και συστημάτων συστάσεων κ.α. Η διπλωματική εργασία κατέληξε στο συμπέρασμα ότι η τεχνολογία του Neo4j μπορεί να εφαρμοστεί επιτυχώς και στον τομέα των βιβλιογραφικών δεδομένων. Στην εργασία τους οι Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen και Dawn Wilkins με τίτλο “A Comparison of a Graph Database and a Relational Database” [6] συγκρίνουν το δημοφιλές σύστημα διαχείρισης μη σχεσιακών βάσεων δεδομένων Neo4j με το επίσης δημοφιλές σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων MySQL ως την υποκείμενη τεχνολογία για την αποθήκευση και προσπέλαση δεδομένων ενός συστήματος προέλευσης δεδομένων. Τέλος στην εργασία του ο Renzo Angles με τίτλο “A Comparison of Current Graph Database Models” [7] κάνει μια επισκόπηση των χαρακτηριστικών των μοντέλων δεδομένων των υφιστάμενων τεχνολογιών βάσεων δεδομένων γράφου και επισημαίνει τα ισχυρά και αδύνατα σημεία τους. Στις τεχνολογίες που συγκρίνονται συμπεριλαμβάνεται και η τεχνολογία του Neo4j που χρησιμοποιήθηκε στη τρέχουσα διπλωματική εργασία.

Όσον αφορά τον δεύτερο πυλώνα που αφορά την χρήση της βιβλιοθήκης Dask ο Matthew Rocklin με τίτλο “Dask: Parallel Computation with Blocked Algorithms and Task Scheduling” [8] περιγράφει τα χαρακτηριστικά της συγκεκριμένης βιβλιοθήκης που μπορούν να χρησιμοποιηθούν για την εκτέλεση πολύπλοκων υπολογισμών σε μεγάλους όγκους δεδομένων που δεν μπορούν να αποθηκευτούν στη μνήμη του υπολογιστή. Επίσης στην εργασία τους οι Mathieu Dugre, Val ´ erie Hayot-Sasson και Tristan Glatard με τίτλο “A Performance Comparison of Dask and Apache Spark for Data-Intensive Neuroimaging Pipelines” συγκρίνουν την απόδοση των δυο πιο δημοφιλών βιβλιοθηκών Dask και Apache Spark για την επεξεργασία των δεδομένων που εμφανίζονται σε απαιτητικές ροές νευροαπεικόνισης [9]. Οι Fail Gafarov, Dmitriy Minullin και Viliuza Gafarova στην εργασία τους με τίτλο “Dask-Based Efficient Clustering of Educational Texts”[10] αναγνωρίζουν την πολυπλοκότητα των επεξεργασιών που υπεισέρχονται στην συσταδοποίηση μεγάλου όγκου εκπαιδευτικών κειμένων και προτείνουν ένα καινοτόμο σύστημα συσταδοποίησης που βασίζεται στην βιβλιοθήκη παραλληλοποιημένης και κατανεμημένης επεξεργασίας Dask. Τέλος στην εργασία του ο James Crist με τίτλο “Dask & Numba: Simple Libraries for Optimizing Scientific Python Code” [11] περιγράφει πως μπορούμε να χρησιμοποιήσουμε τον υψηλής απόδοσης μεταγλωττιστή της Python Numba και

την ευέλικτη βιβλιοθήκη Dask προκειμένου να επιταχύνουμε την εκτέλεση αριθμητικών υπολογισμών σε γλώσσα προγραμματισμού Python.

Παρακινούμενοι από τις παραπάνω εργασίες βασιστήκαμε στις δυο αυτές καινοτόμες τεχνολογίες ώστε να μπορέσουμε να διαχειριστούμε αποδοτικά και αποτελεσματικά τα βιβλιογραφικά δεδομένα της υποδομής που κατασκευάσαμε. Βασικοί παράγοντες επιλογής των δυο αυτών τεχνολογιών αποτέλεσαν ο μεγάλος όγκος των βιβλιογραφικών δεδομένων που αδυνατούσαν να αποθηκευτούν ταυτόχρονα στην κύρια μνήμη, η ανομοιογένεια, η διαφορετική ποιότητα, οι πολύπλοκες επεξεργαστικές ροές που έπρεπε να εφαρμοστούν σε αυτά, ο καταιγιστικός ρυθμός ενημέρωσης και η ισχυρή διασύνδεση των βιβλιογραφικών δεδομένων.

## 1.5 Διάρθρωση εργασίας

Η διάρθρωση της διπλωματικής έχει ως εξής:

Στο Κεφάλαιο 2 γίνεται μια γενική επισκόπηση του πρώτου βασικού τεχνολογικού πυλώνα στο οποίο βασίστηκε η διπλωματική. Συγκεκριμένα περιγράφονται οι δυνατότητες του Neo4j ως ένα σύστημα διαχείρισης μη σχεσιακών βάσεων δεδομένων γράφου. Στην συνέχεια συγκρίνονται οι σχεσιακές με τις μη σχεσιακές βάσεις δεδομένων γράφου και επισημαίνονται τα αναμφισβήτητα πλεονεκτήματα της δεύτερης. Σε αυτά συγκαταλέγονται, ανάμεσα σε πολλά άλλα, ένα πολύ δυναμικό μοντέλο δεδομένων και η εύκολη στην εκμάθηση γλώσσα ερωτοαποκρίσεων Cypher. Τέλος αναφέρονται οι αλγορίθμοι γράφου που μπορούν να εφαρμοστούν στα αποθηκευμένα βιβλιογραφικά δεδομένα της βάσης γράφου ώστε να αξιολογηθεί η αλληλεπιδραστικότητα και σημαντικότητα κάθε οντότητας της υποδομής διαχείρισης βιβλιογραφικών δεδομένων.

Στο Κεφάλαιο 3 γίνεται μια γενική επισκόπηση του δεύτερου βασικού τεχνολογικού πυλώνα στο οποίο βασίστηκε η διπλωματική. Συγκεκριμένα περιγράφονται οι δυνατότητες παραλληλοποιημένης και κατανεμημένης εκτέλεσης κώδικα Python σε πολύ μεγάλα σύνολα δεδομένων. Επιπρόσθετα περιγράφεται η γλώσσα προγραμματισμού Python και αναφέρονται οι αρετές που την ανέδειξαν ως την κατεξοχήν γλώσσα συγγραφής κώδικα για πολύπλοκες ροές ανάλυσης δεδομένων και εκμάθησης μοντέλων μηχανικής μάθησης. Γίνεται επίσης αναφορά στο μηχανισμό GIL της Python που μπορεί να υποβαθμίσει σημαντικά την απόδοση μιας εφαρμογής σε ταυτοχρονικά περιβάλλοντα εκτέλεσης. Στην συνέχεια αναφερόμαστε στην βιβλιοθήκη Dask που είναι από τις πιο ευρέως χρησιμοποιούμενες βιβλιοθήκες παραλληλοποιημένης εκτέλεσης κώδικα Python σε κατανεμημένες συστοιχίες υπολογιστικών συστημάτων (Clusters). Ειδικότερα αναφερονται οι πιο χρήσιμες δυνατότητες της βιβλιοθήκης που συνετέλεσαν και αυτές καθοριστικά στην επιλογή της. Η επιλογή του δεύτερου τεχνολογικού πυλώνα επιβλήθηκε λόγω του τεράστιου όγκου δεδομένων και των πολύπλοκων ροών επεξεργασίας που έπρεπε να εφαρμοστούν σε αυτά.

Στο Κεφάλαιο 4 παρουσιάζονται αναλυτικά τα στάδια σχεδίασης και ανάπτυξης της εφαρμογής. Συγκεκριμένα περιγράφονται οι αποθηκευτικές και επεξεργαστικές προκλήσεις που ανέκυψαν και παραθέτονται τρόποι αντιμετώπισης τους. Αναφέρονται επίσης οι δυο βασικές προσεγγίσεις που ακολουθήθηκαν για την αποδοτική και αποτελεσματική εισαγωγή πολύ μεγάλων όγκων βιβλιογραφικών δεδομένων σε μια βάση δεδομένων γράφου.

Στο Κεφάλαιο 5 περιγράφεται η συνολική αρχιτεκτονική δομή της εφαρμογής. Ειδικότερα περιγράφεται η λειτουργικότητα των επιμέρους τμημάτων που την απαρτίζουν καθώς και πως σχετίζονται μεταξύ τους. Επίσης επεξηγούνται αναλυτικά διάφορες σημαντικές λεπτομέρειες υλοποίησης καθώς και ο τρόπος εκτέλεσης της εφαρμογής πελάτη Python στο υπολογιστικό μηχάνημα του πανεπιστημίου. Στα πλαίσια της διπλωματικής εξετάστηκαν δυο εκδόσεις εφαρμογών

πελάτη για την εισαγωγή των βιβλιογραφικών δεδομένων στην βάση δεδομένων γράφου. Η πρώτη έκανε χρήση του οδηγού Python που υποστηρίζεται από το Neo4j. Η δεύτερη έκανε χρήση της εντολής import του εργαλείου διαχείρισης neo4j-admin. Τελικά η δεύτερη προτιμήθηκε ως η πλέον αποδοτική και αποτελεσματική επιλογή.

Στο Κεφάλαιο 6 παρουσιάζεται η συνολική μεθοδολογία εργασίας. Σε αυτήν περιλαμβάνεται η αρχική διερευνητική ανάλυση των συνόλων δεδομένων, ώστε να γίνει κατανοητή σε βάθος η δομή και ποιότητά τους. Στην συνέχεια ακολουθεί το στάδιο προεπεξεργασίας των δεδομένων, όπου επιχειρείται η συμπλήρωση και η ομογενοποίηση των ελλειπουσών τιμών. Ακολουθεί το στάδιο φιλτραρίσματος των δεδομένων, όπου επιλέγονται μόνο εκείνα τα στιγμιότυπα των οντοτήτων του μοντέλου δεδομένων που ικανοποιούν τις προδιαγραφές του τομέα προβλήματος των βιβλιογραφικών δεδομένων που μελετούμε. Στη συνέχεια ακολουθεί το στάδιο καθαρισμού των ανεπεξέργαστων δεδομένων, όπου επιχειρείται η επίλυση των περισσότερων προβλημάτων που εντοπίστηκαν στο προηγούμενο στάδιο. Το επόμενο στάδιο περιλαμβάνει την δημιουργία του μοντέλου δεδομένων γράφου. Φυσικά αυτό θα αποτελέσει μια πρώτη έκδοση του μοντέλου το οποίο αναμένεται μελλοντικά να αλλάξει αρκετές φορές ανταποκρινόμενο στις μεταβαλλόμενες ανάγκες των χρηστών. Στο στάδιο που ακολουθεί εφαρμόζονται όλες οι απαραίτητες βελτιστοποιήσεις/ παραμετροποιήσεις ώστε να αυξηθεί η αποδοτικότητα εκτέλεσης της εφαρμογής Python που επιχειρεί να εισάγει τα δεδομένα μέσω του οδηγού Python. Σε αυτό εφαρμόστηκαν όλες οι βέλτιστες πρακτικές διαχείρισης της κύριας και δευτερεύουσας μνήμης και συγγραφής αποτελεσματικών ερωτημάτων Cypher. Οι παραμετροποιήσεις αυτές είναι κρίσιμες κυρίως για την πρώτη προσέγγιση. Σε αυτήν επιχειρείται να εισαχθούν τα δεδομένα στην βάση δεδομένων γράφου μέσω του επίσημα υποστηριζόμενου από το Neo4j οδηγού της Python. Στο επόμενο στάδιο φορτώνονται τα επεξεργασμένα δεδομένα στη βάση δεδομένων γράφου βάσει του μοντέλου που αποκρυσταλλώθηκε σε προηγούμενο στάδιο. Το στάδιο αυτό υλοποιήθηκε μέσω μιας εφαρμογής πελάτη σε γλώσσα προγραμματισμού Python. Ενώ μελετήθηκαν δυο προσεγγίσεις φόρτωσης των δεδομένων στη βάση δεδομένων γράφου τελικά επιλέχθηκε η λύση όπου μια εφαρμογή πελάτης Python δημιουργεί τα ειδικής μορφής αρχεία csv, τα οποία στη συνέχεια αποτελούν την είσοδο του διαχειριστικού εργαλείου neo4j-admin import. Επίσης, εκτελούνται οι κατάλληλες ερωτήσεις Cypher για την εξαγωγή χρήσιμων στατιστικών συμπερασμάτων.

Στο κεφάλαιο 7 γίνεται αναφορά στις τεχνολογίες και τα προγραμματιστικά περιβάλλοντα υλοποίησης της εφαρμογής. Συγκεκριμένα περιγράφοντα ο επαγγελματικού επιπέδου συντάκτης κειμένων EmEditor. Σε αυτόν βασιστήκαμε ώστε να μπορέσουμε να ανοίξουμε τα ανεπεξέργαστα αρχεία κειμένου streaming Json lines. Το βήμα αυτό ήταν υποχρεωτικό, ώστε να καταφέρουμε να αποκτήσουμε προκαταρκτικά την αναγκαία ορατότητα ως προς την ποιότητά τους. Επίσης σε αυτόν βασιστήκαμε για να αυτοματοποιήσουμε τον τεμαχισμό των αρχείων μας σε περισσότερα άλλα μικρότερου μεγέθους (File Partitions). Το βήμα αυτό επιβλήθηκε λόγω του τεράστιου μεγέθους των αρχικών αρχείων. Στην συνέχεια αρκετά από αυτά μπόρεσαν να αποθηκευτούν ταυτόχρονα στη μνήμη κάθε κόμβου εργάτη της βιβλιοθήκης Dask (Dask Workers) και να υποστούν περαιτέρω επεξεργασίες. Στη συνέχεια χρησιμοποιήσαμε τη διαδικτυακή εφαρμογή arrows.app που συνοδεύει το Neo4j. Μέσω αυτής δημιουργήσαμε μια γραφική απεικόνιση του μοντέλου δεδομένων της βάσης δεδομένων γράφου. Η εφαρμογή αυτή παρέχει δυνατότητες εύκολής αποθήκευσης της τρέχουσας έκδοσης του μοντέλου σε διάφορες μορφές εικόνας στον σκληρό δίσκο, στο Google Drive, αλλά και στη μνήμη του διαφυλλιστή ιστού (Web Browser Storage) που χρησιμοποιήθηκε για την ανάπτυξη του. Ειδικά η τελευταία δυνατότητα αποδείχτηκε εξαιρετικά χρήσιμη μιας και από την φύση του το μοντέλο δεδομένων του Neo4j είναι εξαιρετικά ευμετάβλητο. Τέλος προκειμένου να ανταπεξέλθουμε

στο μεγάλο αποθηκευτικό και επεξεργαστικό φορτίο των συνόλων δεδομένων μας χρησιμοποιήσαμε ισχυρά εικονικά μηχανήματα (Virtual Machines). Συγκεκριμένα αρχικά κατά τη φάση της συγγραφής του κώδικα Python χρησιμοποιήθηκαν τα δωρεάν μηχανήματα που παρέχουν οι πλατφόρμες Google Colab και Kaggle.com. Αυτά χρησιμοποιήθηκαν καταρχήν για την συγγραφή και εξσφαλμάτωση του μεγαλύτερου όγκου του κώδικα Python που έτρεχε επάνω σε αντιπροσωπευτικά υποσύνολα των δεδομένων. Γρήγορα όμως έγινε αντιληπτό ότι δεν επαρκούσαν και προσανατολισθήκαμε στα πληρωμένα πανίσχυρα εικονικά μηχανήματα που προσφέρει η υποδομή GCP της Google. Σε αυτά τα μηχανήματα έτρεχε παραλληλοποιημένα ο κώδικας Python στο σύνολο των δεδομένων. Ωστόσο λόγοι κόστους επέβαλαν τελικά την εκτέλεση του κώδικα Python τοπικά στο εικονικό μηχάνημα που τρέχει και ο εξυπηρετητής της βάσης δεδομένων γράφου στο υπολογιστικόσύστημα της σχολής. Η εκτέλεση αυτή έγινε σταδιακά από τη γραμμή εντολών του μηχανήματος της σχολής στο οποίο προηγουμένως εγκαθιδρύσαμε μια σύνδεση ssh. Σε αυτήν την τελική φάση ο κώδικας Python έτρεχε σε μορφή αρχείου κώδικα Python παρασκηνιακά μέσω γραμμής εντολών σε μια σύνοδο screen. Αυτό έγινε ώστε να αντιμετωπιστούν προβλήματα διακοπών ρεύματος και απωλειών συνδεσιμότητας στο απομακρυσμένο μηχάνημα του εξυπηρετητή.

Στο Κεφάλαιο 8 παρουσιάζονται τα συμπεράσματα που προέκυψαν και αναφέρονται μελλοντικές επεκτάσεις και βελτιστοποιήσεις της παρούσας διπλωματικής εργασίας.

## 1.6 Επίλογος

Στο εισαγωγικό αυτό κεφάλαιο έγινε μια πρώτη εισαγωγική περιγραφή των βιβλιογραφικών συνόλων δεδομένων τα οποία θα διαχειριστούμε στα πλαίσια της διπλωματικής μας εργασίας. Πρόκειται για τα βιβλιογραφικά δεδομένα τα οποία συλλέγουν με αυτοματοποιημένες διαδικασίες δυο πασίγνωστοι οργανισμοί καταλογοποίησης της βιβλιογραφικής γνώσης (Mag, Aminer). Ωστόσο αποφασίστηκε να επικεντρώσουμε την προσοχή μας στο σύνολο δεδομένων Mag. Πρόκειται για το σύνολο δεδομένων με την εκτενέστερη, υψηλότερης ποιότητας και πιο ενδιαφέρουσα δομή βιβλιογραφική πληροφορία. Στην συνέχεια αναφέρουμε άλλες παρόμοιες εργασίες οι οποίες αποτέλεσαν σημαντικές πηγές έμπνευσης και συνέβαλαν καθοριστικά στην τελική επιλογή των δυο βασικών τεχνολογικών πυλώνων στις οποίες βασίστηκε η διπλωματική. Τέλος περιγράφεται συνοπτικά η διάρθρωση της εργασίας προκειμένου να διευκολύνεται η πλοήγηση του αναγνώστη στις ενότητες ενδιαφέροντος. Στα κεφάλαιο που ακολουθεί επεξηγείται η μη σχεσιακή τεχνολογία βάσεων δεδομένων γράφου, που αποτελεί και τον πρώτο τεχνολογικό πυλώνα της διπλωματικής.

# Κεφάλαιο 2ο: Το Neo4j ως Πλατφόρμα Διαχείρισης Βάσεων Δεδομένων Γράφων

## 2.1 Εισαγωγή

Τα βιβλιογραφικά σύνολα δεδομένων χαρακτηρίζονται από έξι ιδιότητες των οποίων ο αντίστοιχος αγγλικός όρος έχει ως αρχικό γράμμα το «V»: όγκος (Volume), ταχύτητα (Velocity), ποικιλία (Variety), ποιότητα (Veracity), αξία (Value) και συνδεσιμότητα (Valence)[12]. Η πρώτη αναφέρεται στον όγκο των παραγόμενων δεδομένων. Η δεύτερη αναφέρεται στην ταχύτητα δημιουργίας και συλλογής των δεδομένων. Η τρίτη αναφέρεται στις διαφορετικές πηγές συλλογής των δεδομένων. Η τέταρτη αναφέρεται στην ακρίβεια και σπουδαιότητα των δεδομένων που πρόκειται να αναλυθούν. Η πέμπτη αναφέρεται στις ποιοτικές και ποσοτικές συνέπειες που προκαλούνται σε έναν οργανισμό από την εφαρμογή των συμπερασμάτων που προέκυψαν με την επεξεργασία τους. Τέλος η έκτη αναφέρεται στη σθεναρότητα και πυκνότητα με την οποία συνδέονται τα δεδομένα ενός συνόλου. Δεδομένα με τα παραπάνω χαρακτηριστικά απαιτούν καινοτόμες τεχνολογίες αποθήκευσης και επεξεργασίας που θα είναι ικανές να τα επεξεργαστούν αποδοτικά και αποτελεσματικά. Στο κεφάλαιο αυτό αναφερόμαστε στην τεχνολογία των μη σχεσιακών βάσεων δεδομένων γράφου Neo4j που αποτελεί και τον πρώτο βασικό τεχνολογικό πυλώνα υλοποίησης της διπλωματικής εργασίας.

## 2.2 Γενικά

Το Neo4j αποτελεί την κορυφαία πλατφόρμα διαχείρισης μη σχεσιακών βάσεων δεδομένων γράφων (No-SQL Graph Databases). Παρέχεται σε δυο βασικές εκδόσεις: μια ελεύθερης διανομής CE και μια εμπορικής διανομής EE. Η δεύτερη παρέχει επιπλέον χαρακτηριστικά που επιτρέπουν την εύκολη κλιμάκωση και υψηλή διαθεσιμότητα σε απαιτητικά επιχειρηματικά περιβάλλοντα. Σε αυτά συγκαταλέγονται η υψηλού επιπέδου ασφάλεια, η συχνή δημιουργία αντιγράφων ασφαλείας, η αδιάλειπτη διαθεσιμότητα των υπηρεσιών και η συνεχής παροχή τεχνικής υποστήριξης. Επιπρόσθετα παρέχονται υψηλότερα όρια στο πλήθος και στα μεγέθη των στιγμιότυπων Neo4j καθώς και στο πλήθος των κόμβων και των συσχετίσεων που τα απαρτίζουν. Στους παρακάτω πίνακες (Πίνακας 2.1 και Πίνακας 2.2) απεικονίζονται αντίστοιχα η πλήρη γκάμα των βασικών χαρακτηριστικών και δυνατοτήτων απόδοσης και κλιμάκωσης που υποστηρίζει η κάθε έκδοση.

Πίνακας 2.1: Βασικά χαρακτηριστικά υποστηριζόμενα από τις δυο βασικές εκδόσεις του Neo4j ( Πηγή [13])

| A/A | Χαρακτηριστικό | Community Edition | Enterprise Edition |
|---|---|---|---|
| 1 | Property graph model | ✓ | ✓ |
| 2 | Native graph processing & storage | ✓ | ✓ |
| 3 | ACID-compliant transactions | ✓ | ✓ |
| 4 | Cypher graph query language | ✓ | ✓ |
| 5 | Neo4j Browser with syntax highlighting | ✓ | ✓ |
| 6 | Bolt Protocol | ✓ | ✓ |
| 7 | Language drivers for C#, Java, JavaScript & Python | ✓ | ✓ |
| 8 | High-performance native API | ✓ | ✓ |
| 9 | High-performance caching | ✓ | ✓ |
| 10 | Cost-based query optimizer | ✓ | ✓ |
| 11 | Graph algorithms library to support AI initiatives | ✓ | ✓ |
| 12 | Fast writes via native label indexes | ✓ | ✓ |
| 13 | Composite indexes | ✓ | ✓ |
| 14 | Full-text node & relationship indexes | ✓ | ✓ |
| 15 | Store copy | ✓ | ✓ |
| 16 | Auto-reuse of space | ✓ | ✓ |
| 17 | Multiple databases (beyond the system and default databases) | × | ✓ |
| 18 | Slotted and Pipelined Cypher runtimes | × | ✓ |
| 19 | Property-existence constraints | × | ✓ |
| 20 | Node Key constraints | × | ✓ |
| 21 | Listing and terminating running queries | × | ✓ |
| 22 | Role-based access control | × | ✓ |
| 23 | Sub-graph access control | × | ✓ |
| 24 | LDAP and Active Directory integration | × | ✓ |
| 25 | Kerberos security option | × | ✓ |

Πίνακας 2.2: Χαρακτηριστικά απόδοσης και κλιμάκωσης υποστηριζόμενα από τις δυο βασικές εκδόσεις του Neo4j ( Πηγή [13])

| Α/Α | Χαρακτηριστικό | Community Edition | Enterprise Edition |
|---|---|---|---|
| 1 | Causal Clustering for global scale applications | × | ✓ |
| 2 | Enterprise lock manager accesses all cores on server | × | ✓ |
| 3 | Intra-cluster encryption | × | ✓ |
| 4 | Offline backups | ✓ | ✓ |
| 5 | Online backups | × | ✓ |
| 6 | Encrypted backups | × | ✓ |
| 7 | Rolling upgrades | × | ✓ |
| 8 | Automatic cache warming | × | ✓ |
| 9 | Routing and load balancing with Neo4j Drivers | × | ✓ |
| 10 | Advanced monitoring | × | ✓ |
| 11 | Graph size limitations | 4 billion nodes, 34 billion relationships, and 68 billion properties | 4 billion nodes, 34 billion relationships, and 68 billion properties |
| 12 | Bulk import tool | ✓ | ✓ |
| 13 | Bulk import tool, resumable | × | ✓ |

Το Neo4j περιλαμβάνει μια ευρεία γκάμα εργαλείων που επιτρέπουν τον τελικό χρήστη, προγραμματιστή, αναλυτή, διαχειριστή να αλληλεπιδράσει αποδοτικά και αποτελεσματικά με μεγάλες συλλογές συνδεδεμένων δεδομένων σε διαφορετικά περιβάλλοντα. Η πλατφόρμα επιτρέπει την δημιουργία και σύνδεση με τοπικά ή απομακρυσμένα στιγμιότυπα Neo4j σε διάφορους τύπους εγκαταστάσεων (Single Instance Server, CloudServer, Docker, Kubernetes). Η αλληλεπίδραση με κάθε στιγμιότυπο μπορεί να γίνει είτε μέσω μιας γραμμής εντολών cypher-shell, είτε μέσω ενός διαφυλλιστή ιστού, είτε μέσω της εφαρμογής Neo4j Desktop. Ουσιαστικά κάθε Neo4j στιγμιότυπο αποτελεί έναν εξυπηρετητή που τρέχει ένα σύστημα διαχείρισης πολλαπλών βάσεων δεδομένων γράφων με τον περιορισμό ότι σε κάθε χρονική στιγμή μπορεί να είναι ενεργή μόνο μια από αυτές. Κάθε στιγμιότυπο μπορεί να επεκτείνει τη λειτουργικότητα του εγκαθιστώντας επιπλέον βιβλιοθήκες. Η εγκατάσταση μπορεί να γίνει είτε σε επίπεδο Neo4j στιγμιότυπου, με καθολική εφαρμογή σε όλες τις επιμέρους βάσεις δεδομένων που διαχειρίζεται, είτε σε επίπεδο ατομικής βάσης δεδομένων. Επιπλέον η πλατφόρμα επιτρέπει την εύκολη εγκατάσταση και χρήση χρήσιμων εφαρμογών (Graph App Gallery). Αυτές οι εφαρμογές παρέχουν χρήσιμες λειτουργικότητες που διευκολύνουν την οπτικοποίηση/ εξερεύνηση της βάσης γράφου (Neo4j Bloom), τη μεταφορά των υπαρχόντων δεδομένων από οποιασδήποτε τεχνολογίας σχεσιακή βάση δεδομένων σε μια βάση δεδομένων γράφου (Neo4j ETL) κ.α. Επίσης επιτρέπει τους προγραμματιστές να συνδέουν εύκολα τις εφαρμογές τους με μια βάση δεδομένων γράφου οπουδήποτε και αν έχει αναπτυχθεί αυτή μέσω του πρωτοκόλλου Bolt. Επίσημα προσφέρει οδηγούς για μια πληθώρα δημοφιλών γλωσσών προγραμματισμού (.Net, Java, Spring, JavaScript, Go, Python Drivers). Μια ενθουσιώδης κοινότητα προγραμματιστών συνεισφέρει

εθελοντικά οδηγούς για πολλές άλλες γλώσσες προγραμματισμού (Ruby, PHP, Perl, C/C++, R Drivers)[5][14]. Τέλος η πλατφόρμα υποστηρίζει μια ευρεία γκάμα εγκαταστάσεων (Cloud, Cluster, Docker, Kubernetes Deployments ) καλύπτοντας όλες τις ανάγκες[15]–[17].

## 2.3 Η Βάση δεδομένων γράφου

Σήμερα χρησιμοποιείται μια μεγάλη ποικιλία τεχνολογιών σχεσιακών και μη σχεσιακών βάσεων δεδομένων (No-SQL Databases). Κάθε μια εμφανίζει πλεονεκτήματα και μειονεκτήματα που περιορίζουν τη χρήση τους σε συγκεκριμένους περιπτώσεις. Εκεί όμως που μειονεκτούν σημαντικά είναι στη διαχείριση των σχέσεων ισχυρά συνδεδεμένων και μεγάλου όγκου συνόλων δεδομένων. Αυτό οφείλεται στο γεγονός ότι πρέπει να υπολογίσουν τις συνδέσεις των δεδομένων κατά τον χρόνο εκτέλεσης του ερωτήματος. Έτσι υποβαθμίζεται σημαντικά η ταχύτητα εκτέλεσης και αυξάνεται η πολυπλοκότητά του. Αντίθετα σε μια βάση δεδομένων γράφου οι συνδέσεις μεταξύ των δεδομένων θεωρούνται εξίσου σημαντικές με τα ίδια τα δεδομένα. Συνεπώς μαζί με τα δεδομένα αποθηκεύονται και οι συνδέσεις τους επιτρέποντας την αποδοτική διάσχιση πολύ βαθιά συνδεδεμένων δεδομένων. Με τον τρόπο αυτό μπορούν να εκτελεστούν αποδοτικά και αποτελεσματικά πολύπλοκα ερωτήματα που μπορούν να αποκαλύψουν σημαντικά πρότυπα μέσα στα δεδομένα που θα πυροδοτήσουν τη λήψη κρίσιμων επιχειρηματικών αποφάσεων. Τα παραπάνω χαρακτηριστικά καθιστούν τη βάση δεδομένων γράφου μια σημαντική τεχνολογία που χρησιμοποιείται σε όλες τις κρίσιμης σημασίας εφαρμογές μεγάλων ιδιωτικών και κυβερνητικών οργανισμών [15]–[19].

## 2.4 Σύγκριση σχεσιακών και βάσεων δεδομένων γράφου

Η σχεσιακή τεχνολογία των βάσεων δεδομένων βασίζεται στο σχεσιακό μοντέλο (Relational Model) για την περιγραφή της δομής των δεδομένων που πρόκειται να αποθηκεύσει. Αυτό το μοντέλο βασίζεται στους πίνακες για την αποθήκευση πληροφοριών για τις οντότητες ενός τομέα προβλήματος. Κάθε πίνακας περιλαμβάνει ένα σταθερό αριθμό στηλών με προκαθορισμένους τύπους δεδομένων που περιγράφουν το είδος κάθε χαρακτηριστικού της οντότητας που θέλουμε να αποθηκεύσουμε. Αυτό το μοντέλο είναι ιδανικό για στατικά, δομημένα δεδομένα, μεσαίας κλίμακας. Καθώς όμως οι ανάγκες των χρηστών και των επιχειρήσεων μεταβάλλονται σημαντικά με το πέρασμα του χρόνου το μοντέλο δεν μπορεί να ανταποκριθεί σε αυτές άμεσα και με το ελάχιστο κόστος συντήρησης. Ακόμα και μια ελάχιστη αλλαγή στο είδος των δεδομένων που πρέπει να αποθηκευτούν είναι ικανή να πυροδοτήσει μια χιονοστιβάδα αλλαγών στο μοντέλο δεδομένων καθώς και στον κώδικα που βασίζεται σε αυτό. Επίσης αυτό το μοντέλο αδυνατεί να χειρισθεί αποδοτικά και αποτελεσματικά τις συσχετίσεις μεταξύ των δεδομένων. Το όνομα του μοντέλο είναι παραπλανητικό, εφόσον ο μαθηματικός όρος «σχεσιακός» αναφέρεται σε ένα πίνακα, που αποτελεί το βασικό δομικό στοιχείο του μοντέλου και όχι στις συσχετίσεις μεταξύ των οντοτήτων. Κάθε συσχέτιση μεταξύ δυο οντοτήτων αναπαρίσταται με έναν περιορισμό αναφορικής ακεραιότητας. Ουσιαστικά σε αυτό το μοντέλο μια συσχέτιση υπολογίζεται κατά τον χρόνο εκτέλεσης εκτελώντας πολύπλοκες και χρονοβόρες λειτουργίες σύζευξης μεταξύ πινάκων. Όταν όμως το πλήθος και το μέγεθος των πινάκων που συμμετέχουν σε μια λειτουργία σύζευξης αυξάνεται, η απόδοση υποβαθμίζεται σημαντικά. Τέλος ο όγκος, η ταχύτητα και η φύση των δεδομένων που παράγονται καθημερινά στο διαδίκτυο αυξάνεται με εκθετικό ρυθμό. Η ραγδαία εξάπλωση των κοινωνικών δικτύων και η ενσωμάτωση σε κάθε πτυχή της κοινωνικής και εργασιακής μας ζωής εφαρμογών του διαδικτύου των πραγμάτων (Internet of Things) έχει ως αποτέλεσμα την παραγωγή τεράστιων ροών αδόμητων δεδομένων. Αυτά τα δεδομένα πρέπει να αποθηκευτούν και να επεξεργαστούν (πολλές φορές σε σχεδόν πραγματικό χρόνο) ώστε να προκύψουν χρήσιμα συμπεράσματα που θα καθοδηγήσουν τις κατάλληλες επιχειρηματικές

αποφάσεις. Όπως εμφατικά αναφέρει ο αναλυτής δεδομένων Ben Squire, «όταν προσπαθούμε να αναλύσουμε δεδομένα με την σχεσιακή τεχνολογία «είναι σαν να προσπαθούμε να λύσουμε τον κύβο του Rubik κοιτάζοντας μόνο την μια του πλευρά» [20]. Η σχεσιακή τεχνολογία των βάσεων δεδομένων φαίνεται να μην μπορεί να υποστηρίξει αποδοτικά και αποτελεσματικά ροές επεξεργασίας αυτής της κλίμακας σε δεδομένα με αυτά τα χαρακτηριστικά.

Αντίθετα σε μα βάση δεδομένων γράφου ακολουθείται ένα εντελώς διαφορετικό μοντέλο που δίνει ισοδύναμη αξία στα δεδομένα και στις μεταξύ τους συσχετίσεις. Μια συσχέτιση πλέον αποθηκεύεται φυσικά μέσα στο μοντέλο (Native Graph Storage) και δεν υπολογίζεται δυναμικά κατά τον χρόνο εκτέλεσης κάνοντας χρήση ενός ευρετηρίου (Index-free Adjacency). Αυτό επιταχύνει σημαντικά την αποδοτικότητα εκτέλεσης ενός ερωτήματος, εφόσον μια πολύπλοκη λειτουργία σύζευξης ισοδυναμεί πλέον με μια γρήγορη διάσχιση ενός μέρους του γράφου. Η τελευταία αποτελεί λειτουργία σταθερού χρόνου ανεξαρτήτως μεγέθους των δεδομένων. Επιπλέον το μοντέλο γράφου ανταποκρίνεται πλήρως σε οποιεσδήποτε αλλαγές απαιτήσουν οι μεταβαλλόμενες ανάγκες των χρηστών. Το μοντέλο είναι αρκετά ευέλικτο και επιτρέπει τη γρήγορη αλλαγή των αποθηκευμένων δεδομένων ώστε να υποστηριχθούν οι νέες απαιτήσεις λειτουργικότητας των χρηστών.

Όλα τα παραπάνω καθώς και η ραγδαία εξάπλωση και άλλων No-SQL Τεχνολογιών (Key-Value, Document-based, Column-based NoSQL databases) οδηγούν στο συμπέρασμα ότι μια σχεσιακή βάση δεδομένων δεν είναι ιδανική για όλες τις περιπτώσεις χρήσης. Όταν τα δεδομένα είναι δομημένα, αλλάζουν σπάνια, είναι μέτριου μεγέθους και δεν περιλαμβάνουν πολύπλοκες και βαθιές συσχετίσεις μεταξύ τους, τότε μια σχεσιακή βάση δεδομένων αποτελεί την ιδανική επιλογή. Αντίθετα σε εφαρμογές που διαχειρίζονται μεγάλους όγκους αδόμητων, ισχυρά συνδεδεμένων, δυναμικά μεταβαλλόμενων δεδομένων μια βάση δεδομένων γράφου αποτελεί σαφώς καλύτερη επιλογή. Γενικά εφαρμογές με τα παραπάνω χαρακτηριστικά είναι αυτές που χρησιμοποιούνται στα κοινωνικά δίκτυα (Social Networks), στις μηχανές συστάσεων πραγματικού χρόνου (Real-time Recommendation Engines), στις εφαρμογές ανίχνευσης απάτης (Fraud Detection) [16], [17], [19], [21]κ.α.

## 2.5 Το μοντέλο ιδιοτήτων ετικετών γράφου

Μια από τις πρωταρχικές εργασίες που πρέπει να γίνουν κατά την ανάπτυξη μιας Βάσης Δεδομένων οποιασδήποτε τεχνολογίας είναι ο καθορισμός ενός αποτελεσματικού μοντέλου δεδομένων. Αυτό περιγράφει την δομή των δεδομένων που θα αποθηκευτούν και καθορίζει καταλυτικά την αποδοτικότητα προσπέλασης τους. Το Neo4j υποστηρίζει το μοντέλο ιδιοτήτων γράφου, σύμφωνα με το οποίο κάθε τομέας προβλήματος μπορεί να αναπαρασταθεί αφαιρετικά χρησιμοποιώντας κόμβους, συσχετίσεις και ιδιότητες.

Οι κόμβοι αναπαριστούν οντότητες του τομέα προβλήματος οι οποίες μας ενδιαφέρουν και για τις οποίες επιθυμούμε να αποθηκεύσουμε πληροφορίες. Τα δεδομένα αποθηκεύονται ως ιδιότητες και ουσιαστικά αποτελούν ζεύγη κλειδιών και τιμών που περιγράφουν χαρακτηριστικά τους γνωρίσματα. Οι κόμβοι μπορούν να ομαδοποιηθούν και με ετικέτες, οι οποίες αναπαριστούν τους ρόλους των κόμβων στον τομέα προβλήματος. Κάθε κόμβος μπορεί να διαδραματίζει πολλαπλούς ρόλους μέσα στον τομέα προβλήματος και συνεπώς μπορεί να του ανατεθούν πολλαπλές ετικέτες. Οι ετικέτες διαδραματίζουν σημαντικό ρόλο για την αποδοτική διάσχιση της βάσης δεδομένων του γράφου κατά την εκτέλεση ερωτημάτων.

Οι συσχετίσεις αναπαριστούν σημαντικές αλληλεπιδράσεις ζευγαριών κόμβων μέσα στον τομέα προβλήματος που μας ενδιαφέρει. Κάθε συσχέτιση μπορεί να έχει κατεύθυνση, τύπο και ιδιότητες. Η κατεύθυνση καθορίζει την φορά τις σύνδεσης δυο κόμβων, χωρίς αυτό να αποκλείει και την

αποδοτική αμφίδρομη προσπέλαση της. Ο τύπος καθορίζει το είδος της συσχέτισης μεταξύ δυο κόμβων. Κάθε συσχέτιση μπορεί να έχει μόνο έναν τύπο, αλλά επιτρέπονται πολλαπλές συσχετίσεις διαφορετικών τύπων μεταξύ δυο κόμβων. Τέλος οι ιδιότητες αποτελούν ζεύγη κλειδιών και τιμών που περιγράφουν χαρακτηριστικά γνωρίσματα των συσχετίσεων.

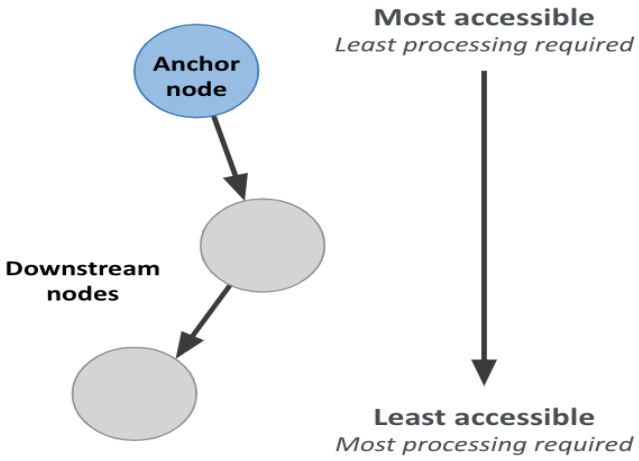
Το μοντέλο δεδομένων γράφου είναι ευέλικτο (Agile), δεν επιβάλλει τη χρήση κάποιου σχήματος (Schema - Free), είναι γραφικό (Whiteboard - Friendly) και δεν επιτρέπεται να περιέχει «σπασμένους συνδέσμους» (No broken links). Το συγκεκριμένο μοντέλο επιτρέπει την εύκολη επέκταση, συρρίκνωση και ανακατασκευή του, ώστε μελλοντικά να καλύψει τις διαρκώς μεταβαλλόμενες ανάγκες του χρήστη και της επιχείρησης (Agile). Επίσης δεν απαιτεί πολύπλοκες και χρονοβόρες επεξηγήσεις της λειτουργικότητάς του. Ακόμη και ένας μη ειδικός αντιλαμβάνεται αμέσως τη δομή και τη σημασιολογία των συνδέσεων των επιμέρους τμημάτων του, όπως ακριβώς αντιλαμβανόμαστε και το νόημα ενός σχεδιαγράμματος που είναι σχεδιασμένο σε έναν πίνακα (Whiteboard - Friendly). Επίσης το μοντέλο δεν επιβάλλει την εκ των προτέρων δημιουργία ενός σχήματος με ευρετήρια ή περιορισμούς (Schema - Free). Ένα ευρετήριο αυξάνει την αποδοτικότητα εκτέλεσης ενός ερωτήματος, ενώ ένας περιορισμός μοντελοποιεί με μεγαλύτερη ακρίβεια τους κανόνες του τομέα προβλήματος. Δεν είναι λίγες οι περιπτώσεις όπου ένα αρχικό μοντέλο έπρεπε να υποστεί σημαντική αναδιάταξη ώστε να περιγράφει με μεγαλύτερη ακρίβεια τον τομέα προβλήματος που αναπαριστά. Προς αυτήν την κατεύθυνση μπορεί να οδηγήσει και η βελτίωση της αποδοτικότητας του συστήματος, αλλά και η ταχύτητα εκτέλεσης των ερωτημάτων. Όλα τα παραπάνω καθιστούν το μοντέλο δεδομένων γράφου ιδανικό για την μοντελοποίηση πολύ πυκνών διασυνδεδεμένων, αδόμητων συνόλων δεδομένων. [15]–[17], [19], [22], [23].

Τέλος θα πρέπει να τονιστεί ότι κανένα μοντέλο δεν είναι ιδανικό για όλες τις περιπτώσεις χρήσης (One Size Fits All). Αντίθετα κάθε μοντέλο θα πρέπει να σχεδιάζεται για να αναπαραστήσει με τη μεγαλύτερη δυνατή ακρίβεια τον συγκεκριμένο τομέα προβλήματος που έχουμε επικεντρωθεί. Όπως αναφέρθηκε, η διαδικασία σχεδιασμού ενός βέλτιστου μοντέλου δεδομένων στο Neo4j είναι δυναμική. Συνήθως ο σχεδιαστής ξεκινάει με ένα αρχικό μοντέλο που περιγράφει τις βασικές πτυχές του τομέα προβλήματος που καλείται να αναπαραστήσει. Στην πορεία και, καθώς η κατανόηση του τομέα προβλήματος βελτιώνεται, το μοντέλο δεδομένων αναδιατάσσεται (Data Model Refactoring), ώστε να ενσωματώνει την επιπλέον γνώση που αποκτήθηκε. Αναδιάταξη του μοντέλου μπορεί να επιβάλλουν και άλλοι παράγοντες που έχουν να κάνουν με την βελτίωση της αποδοτικότητας λειτουργίας του συστήματος και της ταχύτητας εκτέλεσης των ερωτημάτων επάνω στα δεδομένα της βάσης. Εξάλλου ας μην ξεχνούμε ότι και οι ανάγκες των χρηστών μεταβάλλονται σημαντικά με το πέρασμα του χρόνου. Συνεπώς θα πρέπει το χρησιμοποιούμενο μοντέλο να είναι αρκετά ευέλικτο ώστε να ανταποκριθεί σε οποιασδήποτε έκτασης αλλαγές χωρίς ιδιαίτερη σχεδιαστική/ προγραμματιστική προσπάθεια. Δεν είναι τυχαίο που πολλοί αποκαλούν το Neo4j schema – free μιας και δεν προϋποθέτει την εξ αρχής δημιουργία ενός ολοκληρωμένου και στατικού μοντέλου δεδομένων. Αντίθετα αυτό διαμορφώνεται στην πορεία αυξητικά και επαναληπτικά ώστε να προσεγγίσει στα τελευταία στάδια με μεγαλύτερη ακρίβεια τις απαιτήσεις των τελικών χρηστών. Συνεπώς το μοντέλο είναι πλήρως εναρμονισμένο με τις Agile μεθοδολογίες ανάπτυξης λογισμικού (Scrum, Extreme programming , Lean Software Development , Kanban) που είναι πλέον κυρίαρχες στη βιομηχανία παραγωγής λογισμικού.

### 2.5.1 Ιεραρχία προσβασιμότητας

Κατά τη δημιουργία ή αναδιάταξη ενός μοντέλου ιδιοτήτων γράφου πολλές φορές κρίνεται χρήσιμο να γίνει μια προκαταρκτική εκτίμηση της αποδοτικότητας του λαμβάνοντας υπόψιν την ιεραρχία

προσβασιμότητας (Hierarchy of Accessibility). Ουσιαστικά η ιεραρχία αυτή δηλώνει την προσπάθεια που καταβάλει το σύστημα του Neo4j προκειμένου να προσπελάσει τα δεδομένα που χειρίζονται οι κρίσιμης σημασίας ερωτήσεις της εφαρμογής μας. Η ιεραρχία αναπτύσσεται από πάνω προς τα κάτω με αύξουσα σειρά δυσκολίας προσπέλασης και απεικονίζεται στοΣχήμα 2.1.



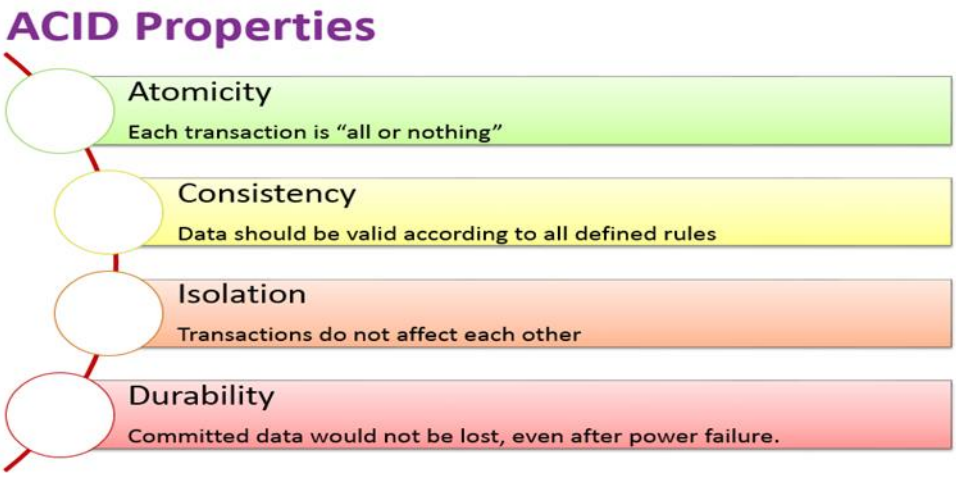

Σχήμα 2.1: Η ιεραρχία προσβασιμότητας δηλώνει πόσο εύκολα προσπελάσιμα είναι τα δεδομένα του μοντέλου ιδιοτήτων γράφου (Πηγή [24])

Συγκεκριμένα η ιεραρχία αυτή κατά αύξουσα σειρά δυσκολίας έχει ως εξής[24]:

1. Ετικέτα κόμβου αγκύρωσης, ιδιότητες κόμβου αγκύρωσης με ευρετήριο
2. Τύποι σχέσεων
3. Μη ευρετηριασμένες ιδιότητες κόμβου αγκύρωσης
4. Ετικέτες περισσότερο απομακρυσμένων κόμβων του μονοπατιού προσπέλασης
5. Ιδιότητες σχέσης, ιδιότητες περισσότερο απομακρυσμένων κόμβων του μονοπατιού προσπέλασης

## 2.6 Η γλώσσα ερωταποκρίσεων cypher

Η Cypher αποτελεί την γλώσσα ερωταποκρίσεων του Neo4j. Χρησιμοποιείται για την εκτέλεση CRUD ερωτημάτων επάνω σε μια βάση δεδομένων γράφου. Κάθε προσπέλαση που επιχειρεί μια ερώτηση Cypher σε μια βάση δεδομένων γράφου στο Neo4j πρέπει να πραγματοποιηθεί στο πλαίσιο μιας δοσοληψίας (Transaction). Μια δοσοληψία που εκτελείται στο ΣΔΒΔ του Neo4j αποτελεί μια ατομική μονάδα εργασίας (Atomic Unit of Work) και είναι πλήρως συμβατή με τις ιδιότητες ACID, όπως απεικονίζεται στο Σχήμα 2.2. Η ατομικότητα διασφαλίζει ότι μια δοσοληψία επιτυγχάνει ή αποτυγχάνει συνολικά. Συνεπώς μερικές τροποποιήσεις που μπορεί να οδηγήσουν σε απώλεια ή διάβρωση των δεδομένων δε γίνονται αποδεκτές. Η συνέπεια διασφαλίζει ότι κάθε τροποποίηση στη βάση δεν παραβιάζει τους κανόνες και περιορισμούς που έχουν προσδιοριστεί. Συνεπώς κάθε αποδεκτή δοσοληψία πρέπει να οδηγεί τη βάση σε μια συνεπή κατάσταση, χωρίς όμως να εγγυάται ότι είναι και σωστή. Η απομόνωση διασφαλίζει ότι καμία δοσοληψία δεν αλληλοεπιδρά ανεξέλεγκτα με οποιαδήποτε άλλη σε ένα ταυτοχρονικό περιβάλλον εκτέλεσης. Συνεπώς μετα την παράλληλη εκτέλεση πολλών δοσοληψιών η βάση οδηγείται στην ίδια κατάσταση που θα οδηγούνταν αν οι ίδιες δοσοληψίες εκτελούνταν σειριακά. Η ανθεκτικότητα διασφαλίζει ότι τα αποτελέσματα μιας επιτυχημένης δοσοληψίας διατηρούνται μόνιμα στη βάση ακόμα και αν συμβεί ένα έκτακτο γεγονός, όπως μια αστοχία υλικού ή λογισμικού. Οι ιδιότητες αυτές εξασφαλίζουν την υψηλή αξιοπιστία και ακεραιότητα των δεδομένων της βάσης που παρέχει και η παραδοσιακή τεχνολογία των σχεσιακών βάσεων δεδομένων [17], [19], [21], [25].




Σχήμα 2.2: Κάθε δοσοληψία στο Neo4jυποστηρίζει τις ιδιότητες ACID ( Πηγή [26])

Επιφανειακά μοιάζει με την γλώσσα SQL που χρησιμοποιείται σε ένα σχεσιακό σύστημα βάσης δεδομένων. Η γλώσσα χρησιμοποιεί κάποιες κοινές εντολές, αλλά οι ομοιότητες σταματούν εκεί. Η Cypher βασίζεται σε μια εντελώς νέα φιλοσοφία αποθήκευσης και προσπέλασης ισχυρά συνδεδεμένων και αδόμητων δεδομένων. Σύμφωνα με αυτήν οι συνδέσεις δεν υπολογίζονται δυναμικά κατά τον χρόνο εκτέλεσης ενός ερωτήματος εκτελώντας πολύπλοκες και χρονοβόρες λειτουργίες σύζευξης (Join Operations). Αντίθετα αποθηκεύονται αποδοτικά με τα ίδια τα δεδομένα καθιστώντας ταχύτατη της εκτέλεσή τους. Η Cypher αποτελεί μια καθαρά δηλωτική γλώσσα. Σε αυτή δηλώνουμε μόνο τι θέλουμε να προσπελάσουμε και όχι πως θα το επιτύχουμε αυτό. Την εκτέλεση αναλαμβάνει στο παρασκήνιο μια μηχανή εκτέλεσης, η οποία, αφού εκτελέσει διάφορες βελτιστοποιήσεις, καταστρώνει και εκτελεί ένα αποδοτικό πλάνο εκτέλεσης. Συνεπώς η μορφή των δεδομένων που θέλουμε να προσπελάσουμε περιγράφεται με τη μορφή προτύπων. Ένα πρότυπο περιγράφει με οπτικό τρόπο (Ascii - Art) μια ακολουθία κόμβων και συσχετίσεων που περιγράφουν τα δεδομένα με τα οποία θέλουμε να αλληλοεπιδράσουμε. Όλα τα παραπάνω χαρακτηριστικά καθιστούν την Cypherπολύ εύκολη στην κατανόηση, εκμάθηση και χρήση παρέχοντας ταυτόχρονα επαγγελματικού επιπέδου χαρακτηριστικά και δυνατότητες [15]–[17], [19], [21].

### 2.6.1 Τρόπος εκτέλεσης ερωτήματος cypher στο Neo4j

Κάθε ερώτημα Cypher πριν εκτελεστεί στο Neo4j υποβάλλεται στον δημιουργό σχεδίων εκτέλεσης (Cypher Query Execution Planner). Αρμοδιότητα του είναι η καταρχήν δημιουργία πολλών ενναλακτικών σχεδίων εκτέλεσης (Execution Plans) για κάθε μεμονωμένο ερώτημα. Από αυτά επιλέγει σε τελική φάση το πιο αποδοτικό βασιζόμενος σε διάφορα στατιστικά στοιχεία για τη βάση δεδομένων, όπως το πλήθος των κόμβων κάποιας ετικέτας, το πλήθος των συσχετίσεων κάποιου τύπου, τα διαθέσιμα ευρετήρια κλπ. Σε γενικές γραμμές ένα πλάνο εκτέλεσης (Execution Plan) είναι μια ανεστραμμένη δενδροειδής δομή από κόμβους, όπου κάθε κόμβος αναπαριστά έναν τελεστή. Ένας τελεστής αντιπροσωπεύει ένα τύπο επεξεργασίας επάνω στα δεδομένα. Γενικά υπάρχουν δυο τύποι τελεστών: ο οκνηρός (Lazy) και ο πρόθυμος (Eager). Ο πρώτος αρχίζει να παραδίδει τα αποτελέσματά που παράγει στον γονικό του τελεστή αμέσως μόλις αρχίζει να τα δημιουργεί. Αντίθετα ο δεύτερος περιμένει πρώτα να ολοκληρωθεί το σύνολο της επεξεργασίας που εκτελεί, πριν καταστήσει διαθέσιμα τα παραγόμενα αποτελέσματα στον γονικό του τελεστή. Τέτοιου είδους τελεστές έχουν υψηλές απαιτήσεις μνήμης και, αν δεν προσεχθούν, μπορεί να υποβαθμίσουν σοβαρά την αποδοτικότητα εκτέλεσης ενός ερωτήματος. Ουσιαστικά μπορούμε να παρομοιάσουμε ένα πλάνο εκτέλεσης ενός ερωτήματος Cypher με την βέλτιστη συνταγή εκτέλεσης ενός φαγητού. Η εκτέλεση

ενός πλάνου εκτέλεσης ξεκινά από τους κόμβους φύλλα με κατεύθυνση από κάτω προς τα πάνω. Συνήθως οι κόμβοι φύλλα αποτελούνται από τελεστές που αναζητούν ή σαρώνουν τα δεδομένα στο μέσο αποθήκευσης. Τέτοιου είδους προσπελάσεις είναι χρονοβόρες και θα πρέπει να επιλέγονται αποδοτικοί τελεστές. Στην συνέχεια διοχετεύουν τα αποτελέσματα στους πατρικούς τους κόμβους για να υποστούν περαιτέρω επεξεργασίες. Τελικά τα αποτελέσματα του ερωτήματος καταλήγουν στον κορυφαίο κόμβο (Root Node) που είναι και υπεύθυνος να τα παρουσιάσει στον τελικό χρήστη. Μέσω της διαδικασίας ελέγχου του προφίλ ενός ερωτήματος (Query Profiling) μπορούμε να πάρουμε αποφάσεις για τους καταλληλότερους τελεστές που θα απαρτίσουν το πλάνο εκτέλεσης και θα βελτιστοποιήσουν τη χρήση των πόρων του συστήματος [15], [17].

## 2.7 Βιβλιοθήκες επέκτασης Neo4j

Η Cypher είναι μια πανίσχυρη και πολύ εκφραστική γλώσσα που ενσωματώνει πολύ αποδοτικές υλοποιήσεις βασικών κομματιών λειτουργικότητας. Πολλές φορές όμως απαιτείται ο προγραμματιστής να χρησιμοποιήσει τον μηχανισμό επέκτασης που συνοδεύει το Neo4j και να γράψει τις δικές τους διαδικασίες και συναρτήσεις. Αυτές συνήθως γράφονται στη γλώσσα προγραμματισμού Java και υλοποιούν λιγότερο συνηθισμένες λειτουργικότητες, όπως αλγορίθμους γράφων που εκτελούνται σε παραλληλοποιημένο υλικό, πολύπλοκες μετατροπές της μορφής των δεδομένων κ.α. Προκειμένου η Cypher να διευκολύνει ακόμα περισσότερο όσους αναπτύσσουν εφαρμογές, συνοδεύεται και από πολλές βιβλιοθήκες που επεκτείνουν τη λειτουργικότητά της. Με αυτόν τον τρόπο οι προγραμματιστές μπορούν να χρησιμοποιήσουν απευθείας συχνά χρησιμοποιούμενα κομμάτια λειτουργικότητας βελτιώνοντας τον χρόνο ανάπτυξης και την ποιότητα των εφαρμογών τους. Παρακάτω ακολουθεί μια σύντομη περιγραφή δυο ευρέως χρησιμοποιούμενων βιβλιοθηκών.

### 2.7.1 Βιβλιοθήκη APOC

Πρόκειται για τη μεγαλύτερη και πιο ευρέως χρησιμοποιούμενη βιβλιοθήκη επέκτασης του Neo4j. Περιλαμβάνει πάνω από 450 διαδικασίες και συναρτήσεις που παρέχουν συχνά χρησιμοποιούμενες λειτουργικότητες σε διάφορους τομείς, όπως η ενσωμάτωση και μετατροπή της μορφής των δεδομένων, επεξεργασίες κειμένου, μαθηματικές συναρτήσεις, φόρτωση της βάσης δεδομένων γράφου με δεδομένα διαφόρων μορφών, εκτέλεση CRUD λειτουργιών κατά τμήματα, μαζική αναδιάταξη της δομής του γράφου, υλοποιήσεις αλγορίθμων συμπίεσης/ αποσυμπίεσης/ κατακερματισμού κ.α. Κάθε υποπρόγραμμα μπορεί να κληθεί είτε ως μέρος κάποιου κώδικα Cypher, ή αυτόνομα μέσα από κάποια άλλη εφαρμογή πελάτη ή την γραμμή εντολών Cypher-Shell. Διατίθεται σε δυο εκδόσεις: βασική (Core) και πλήρης (Full). Η πρώτη περιλαμβάνει τις βασικότερες λειτουργικότητες, οι οποίες δεν παρουσιάζουν εξαρτήσεις από εξωτερικά πακέτα λογισμικού και δεν απαιτούν ρυθμίσεις από τη μεριά του χρήστη. Η δεύτερη αποτελεί υπερσύνολο της πρώτης και περιλαμβάνει επιπλέον λειτουργικότητες που απαιτούνται σε πιο εξειδικευμένα περιβάλλοντα ανάπτυξης του εξυπηρετητή Neo4j (Neo4j Sandbox, Docker, Aura κ.α.). Στα πλαίσια της διπλωματικής εγκαταστάθηκε η βασική έκδοση 4.4.0.2 της βιβλιοθήκης προκειμένου να προσπελαστούν οι πολύτιμες λειτουργικότητες που προσφέρει [27].

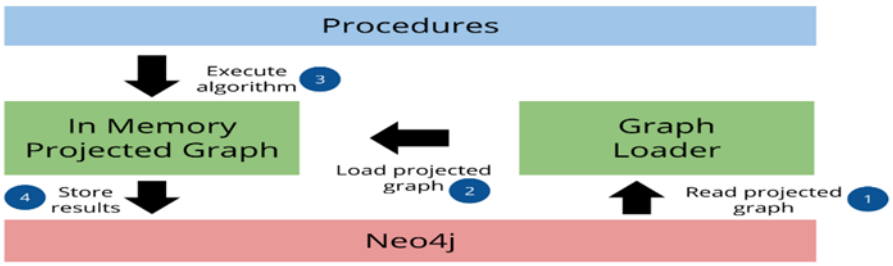
### 2.7.2 Βιβλιοθήκη GDS

Πρόκειται για μια πανίσχυρη βιβλιοθήκη που επιτρέπει τους αναλυτές δεδομένων και τους ειδικούς μηχανικής μάθησης να προβλέπουν την δυναμική συμπεριφορά συστημάτων ή ομάδων κοινωνικών δικτύων και να δημιουργούν πολύπλοκες ροές επεξεργασίας μηχανικής μάθησης. Χρησιμοποιεί

αλγορίθμους γράφων που εντοπίζουν και αναλύουν τις συνδέσεις και τις δομές δικτύων που ενυπάρχουν σε μεγάλους όγκους δεδομένων, ώστε να εξάγουν χρήσιμα συμπεράσματα επάνω στα οποία μπορούν να βασιστούν οι μελλοντικές επιχειρηματικές αποφάσεις. Στα συμπεράσματα περιλαμβάνονται η εύρεση του πόσο σημαντικός είναι κάθε σύνδεσμος, την αλληλεπιδραστικότητα κάθε οντότητας του μοντέλου δεδομένων, την εύρεση ομάδων συνδέσμων σε ένα δίκτυο κ.α.

Η βιβλιοθήκη προσφέρεται σε δυο εκδόσεις: ελεύθερης διανομής και εμπορική. Η πρώτη προσφέρει πλήρη πρόσβαση στην λειτουργικότητα της βιβλιοθήκης, αλλά η εκτέλεση των αλγόριθμων είναι λιγότερο αποδοτική μιας και οι δυνατότητες παραλληλοποιημένης εκτέλεσης τους περιορίζεται στους τέσσερις πυρήνες. Αντίθετα στην εμπορική έκδοση η εκτέλεση των αλγορίθμων κλιμακώνει ελεύθερα στο σύνολο των πυρήνων του μηχανήματος που τρέχει ο αλγόριθμος. Επίσης υποστηρίζει επιπλέον χαρακτηριστικά, όπως το σύστημα ελέγχου πρόσβασης βάσει ρόλου RBACS, αποθήκευση απεριόριστου αριθμού μοντέλων στον κατάλογο των μοντέλων, δημοσίευση και μόνιμη αποθήκευση ενός αποθηκευμένου μοντέλου στον σκληρό δίσκο [28].

Ένα τυπικό σενάριο χρήσης της βιβλιοθήκης περιλαμβάνει δυο φάσεις: τη φάση ανάπτυξης και τη φάση λειτουργίας. Στην πρώτη φάση θα πρέπει να ρυθμίσουμε κατάλληλα το υπολογιστικό σύστημα που είναι εγκατεστημένη η βιβλιοθήκη, να ορίσουμε τις όψεις του γράφου που θα προβάλουμε στη μνήμη σωρού και να επιλέξουμε την ακολουθία αλγορίθμων που θα απαρτίζουν τη ροή επεξεργασίας. Σε αυτήν τη φάση κρίνεται σκόπιμο να πραγματοποιηθεί και μια εκτίμηση των απαιτήσεων μνήμης κάθε αλγόριθμου. Υπάρχουν δύο είδη πόρων που πρέπει να έχουμε υπόψιν: την προβεβλημένη στη μνήμη σωρού όψη του γράφου και τις δομές δεδομένων του κάθε αλγορίθμου. Στη δεύτερη φάση θα πρέπει να δημιουργήσουμε τις προβολές των όψεων της βάσης δεδομένων γράφου στη μνήμη σωρού, να εκτελέσουμε τους αλγόριθμους, να αναλύσουμε και να αποθηκεύσουμε τα αποτελέσματα που παράγονται στη μνήμη ή στον σκληρό δίσκο ανάλογα με τον τρόπο λειτουργίας του αλγόριθμου [13]. Κάθε προβολή μιας όψης του γράφου στη μνήμη μπορεί να είναι επώνυμη ή ανώνυμη. Η πρώτη δημιουργείται αυτόματα κατά την εκτέλεση ενός αλγόριθμου και μετά την ολοκλήρωση της εκτέλεσης χάνεται. Αντίθετα ένας επώνυμος γράφος έχει ένα όνομα και αποθηκεύεται στον κατάλογο γράφων. Με αυτόν τον τρόπο μια επώνυμη προβολή είναι διαθέσιμη σε πολλαπλούς αλγόριθμους, χωρίς να χρειάζεται να υπολογίζεται από την αρχή εξοικονομώντας χρόνο. Επιπλέον μπορούμε να βασιστούμε σε έναν επώνυμο γράφο και να δημιουργήσουμε εύκολα υπογράφους που εκφράζουν πιο εξειδικευμένες όψεις ενός αποθηκευμένου στον δίσκο κομματιού του γράφου. Σε αυτές τις όψεις μπορούν στη συνέχεια να εκτελεστούν και άλλοι αλγόριθμοι μιας σύνθετης ροής επεξεργασίας. Τα βήματα που απαρτίζουν ένα τυπικό σενάριο χρήσης της βιβλιοθήκης απεικονίζονται στο Σχήμα 2.3[12], [29].




Σχήμα 2.3: Τυπικό σενάριο χρήσης της βιβλιοθήκης GDS ( Πηγή [30])

#### 2.7.2.1 Τύποι αλγορίθμων γράφων GDS

Οι αλγόριθμοι της βιβλιοθήκης είναι υψηλής πολυπλοκότητας και η εκτέλεση τους σε κομμάτια μεγάλων γράφων εξαιρετικά χρονοβόρα. Για αυτό στην πλειοψηφία τους έχουν εφαρμοστεί βελτιστοποιήσεις και παραλληλοποιηθεί, ώστε να επιταχυνθεί ο χρόνος εκτέλεσης τους. Καθένας από αυτούς συνοδεύεται από κάποια χαρακτηριστικά γνωρίσματα (Algorithm Traits). Ουσιαστικά πρόκειται για ορισμένες πλευρές του κομματιού του γράφου που θα εκτελεστούν, που οδηγούν τον συγκεκριμένο αλγόριθμο να παράγει βέλτιστα αποτελέσματα. Σε αυτά συγκαταλέγονται τα εξής: directed, undirected, homogeneous, heterogeneous και weighted. Το πρώτο αναφέρεται σε αλγόριθμους όπου η κατεύθυνση του γράφου είναι σημαντική. Το δεύτερο σε αλγόριθμους όπου η κατεύθυνση των συνδέσεων δε λαμβάνεται υπόψιν. Το τρίτο αναφέρεται σε αλγόριθμους όπου οι διαφορετικοί τύποι κόμβων και συσχετίσεων αντιμετωπίζονται ομοιόμορφα. Το τέταρτο αναφέρεται σε αλγόριθμους που μπορούν να διαχωρίζουν τους διαφορετικούς τύπους κόμβων και συσχετίσεων. Τέλος το πέμπτο αναφέρεται σε αλγόριθμους που επιτρέπουν μέσω κατάλληλων ρυθμίσεων να ορίσουμε κάποιες ιδιότητες των κόμβων ή των συσχετίσεων ως βάρη. Η σημασιολογία των βαρών (Απόσταση, Χρόνος, Κόστος, Σπουδαιότητα κ.α.) αλλάζει κάθε φορά ανάλογα με την περίπτωση χρήσης του συγκεκριμένου τομέα προβλήματος. Γενικά οι αλγόριθμοι κατηγοριοποιούνται σε ένα από τρία επίπεδα ωριμότητας (Maturity Tiers): production-quality, beta και alpha. Στο πρώτο ανήκουν όλοι οι αλγόριθμοι που έχουν δοκιμαστεί σε επίπεδο παραγωγικής λειτουργίας και έχει αποδειχθεί ότι είναι σταθεροί, αποδοτικοί και κλιμακώνουν χωρίς προβλήματα. Στο δεύτερο ανήκουν οι αλγόριθμοι που είναι έτοιμοι να ενταχθούν στο επίπεδο Production-quality. Τέλος στο τρίτο ανήκουν όλοι οι αλγόριθμοι που είναι ακόμα σε πειραματική φάση και δεν αποκλείεται μελλοντικά να αντικατασταθούν από νέες πιο αποδοτικές υλοποιήσεις. Όλοι οι αλγόριθμοι χρησιμοποιούνται μέσω της εκτέλεσης της αντίστοιχης διαδικασίας στον διαφυλλιστή (Neo4j Browser), τη γραμμή εντολών (Neo4j Cypher-Shell) ή και μια εφαρμογή πελάτη σε γλώσσα προγραμματισμού Python (Neo4j Python Client Application). Στην τελευταία περίπτωση απαιτείται και η εγκατάσταση στην εφαρμογή πελάτη των κατάλληλων βιβλιοθηκών. Κάθε αλγόριθμος υποστηρίζει κάποιους από τους εξής τρόπους λειτουργίας (Operation Modes): stream, stats, mutate και write. Ο πρώτος επιστρέφει στον χρήστη τα αναλυτικά αποτελέσματα του αλγόριθμου σαν σειρές από εγγραφές. Ο δεύτερος επιστρέφει στον χρήστη μια μοναδική γραμμή με τα συνολικά στατιστικά αποτελέσματα της εφαρμογής του αλγόριθμου. Ο τρίτος τροποποιεί το προβεβλημένο στη μνήμη κομμάτι του γράφου επάνω στο οποίο εφαρμόστηκε ο αλγόριθμος και επιστρέφει στον χρήστη μια μοναδική γραμμή με τα συνολικά στατιστικά αποτελέσματα. Τέλος ο τέταρτος τροποποιεί το κομμάτι του γράφου που είναι μόνιμα αποθηκευμένο στον σκληρό δίσκο και επιστρέφει στον χρήστη μια γραμμή με τα συνολικά στατιστικά αποτελέσματα της εφαρμογής του αλγόριθμου. Στο σημείο αυτό επιβάλλεται να αναφέρουμε ότι δεν υποστηρίζουν όλοι οι αλγόριθμοι όλους τους παραπάνω τρόπους λειτουργίας. Στη συνέχεια ακολουθεί μια συνοπτική περιγραφή των σημαντικότερων από αυτών [12], [29].

##### 2.7.2.1.1 Κεντρικότητα

Οι αλγόριθμοι κεντρικότητας χρησιμοποιούνται για να προσδιορίσουν τη σημασία μιας οντότητας σε ένα δίκτυο. Δηλαδή η κεντρικότητα έχει να κάνει με την κατανόηση του ποια οντότητα είναι πιο σημαντική σε ένα δίκτυο. Υπάρχουν διαφορετικοί τύποι αλγορίθμων κεντρικότητας, κάθε ένας από αυτούς σχεδιάστηκε ώστε να μετράει την σημαντικότητα από διαφορετικές οπτικές γωνίες. Παρακάτω (Πίνακας 2.3) απεικονίζονται όλοι οι αλγόριθμοι κεντρικότητας που υποστηρίζει η βιβλιοθήκη κατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Maturity Level)[29].

Πίνακας 2.3: Αλγόριθμοι κεντρικότητας βιβλιοθήκης GDS κατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Πηγή [31])

| | Production-quality | Beta | Alpha |
|---|---|---|---|
| Algorithms Centrality | Page Rank<br>Article Rank<br>Eigenvector Centrality<br>Betweenness Centrality<br>Degree Centrality | Closeness Centrality | Harmonic Centrality<br>HITS<br>Influence Maximization |

#### 2.7.2.1.2 Ανίχνευση κοινότητας

Οι αλγόριθμοι ανίχνευσης κοινότητας ονομάζονται ακόμα και αλγόριθμοι συσταδοποίησης (Clustering Algorithms) ή και αλγόριθμοι τμηματοποίησης (Partitioning Algorithms). Χρησιμοποιούνται για να προσδιορίσουν ομάδες οντοτήτων καθώς και πόσο ανθεκτικές είναι οι ομάδες αυτές. Βασίζονται στην ιδέα ότι οι οντότητες που είναι μέλη μιας ομάδας περιέχουν πολλές περισσότερες σχέσεις μεταξύ τους από ότι με άλλες οντότητες που δεν ανήκουν στην ίδια ομάδα. Υπάρχουν διαφορετικοί τύποι αλγορίθμων ανίχνευσης κοινότητας, κάθε ένας από αυτούς σχεδιάστηκε ώστε να μετράει τη συμμετοχή σε μια κοινότητα από διαφορετική οπτική γωνία. Στον πίνακα που ακολουθεί (Πίνακας 2.4) απεικονίζονται όλοι οι αλγόριθμοι ανίχνευσης κοινότητας που υποστηρίζει η βιβλιοθήκη κατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Maturity Level)[29].

Πίνακας 2.4: Αλγόριθμοι ανίχνευσης κοινότητας βιβλιοθήκης GDS κατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Πηγή [32])

| | Production-quality | Beta | Alpha |
|---|---|---|---|
| Algorithms Community Detection | Louvain<br>Label Propagation<br>Weakly Connected Components<br>Triangle Count<br>Local Clustering Coefficient | K-1 Coloring<br>Modularity Optimization | Strongly Connected Components<br>Speaker-Listener Label Propagation<br>Approximate Maximum k-cut<br>Conductance metric |

#### 2.7.2.1.3 Εύρεση μονοπατιού

Οι αλγόριθμοι εύρεσης μονοπατιού χρησιμοποιούνται για να προσδιορίσουν μονοπάτια μεταξύ κόμβων. Βασίζονται στους αλγόριθμους αναζήτησης γράφων, αλλά με μια διαφορά. Προσπαθούν να εντοπίζουν τα βέλτιστα κάθε φορά μονοπάτια. Η έννοια του βέλτιστου διαφοροποιείται κάθε φορά ανάλογα με την περίπτωση χρήσης του τομέα προβλήματος που εφαρμόζονται. Υπάρχουν διαφορετικοί τύποι αλγορίθμων εύρεσης μονοπατιού, κάθε ένας από αυτούς σχεδιάστηκε ώστε να βρίσκει το βέλτιστο μονοπάτι από διαφορετική οπτική γωνία. Ακολούθως (Πίνακας 2.5)

απεικονίζονται όλοι οι αλγόριθμοι εύρεσης μονοπατιού που υποστηρίζει η βιβλιοθήκη κατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Maturity Level)[29].

Πίνακας 2.5: Αλγόριθμοι εύρεσης μονοπατιού βιβλιοθήκης GDSκατηγοριοποιημένοι ως προς το επίπεδο ωριμότητας (Πηγή [33])

| | Production-quality | Beta | Alpha |
|---|---|---|---|
| Algorithms Pathfinding | Delta-Stepping Single-Source Shortest Path<br>Dijkstra Source-Target Shortest Path<br>Dijkstra Single-Source Shortest Path<br>A* Shortest Path<br>Yen's Shortest Path<br>Breadth First Search<br>Depth First Search | Random Walk | Minimum Weight Spanning Tree<br>All Pairs Shortest Path |

## 2.8 Επίλογος

Στο κεφάλαιο αυτό εστίασε στη μη σχεσιακή τεχνολογία βάσεων δεδομένων γράφου. Συγκεκριμένα έγινε συνοπτική περιγραφή των δυνατοτήτων των δυο βασικών εκδόσεων της πιο δημοφιλούς πλατφόρμας διαχείρισης μη σχεσιακών βάσεων δεδομένων γράφου Neo4j. Στη συνέχεια ακολούθησε μια σύγκριση με την παραδοσιακή τεχνολογία των σχεσιακών βάσεων δεδομένων. Άμεσα επισημάνθηκαν τα αναμφισβήτητα πλεονεκτήματα της πρώτης τεχνολογίας στη διαχείριση συνόλων δεδομένων με χαρακτηριστικά σαν αυτά που εμφανίζουν τα βιβλιογραφικά δεδομένα. Έπειτα περιγράφηκε το μοντέλο δεδομένων ιδιοτήτων γράφου και επισημάνθηκαν οι αρετές που το καθιστούν ως το πλέον κατάλληλο για τη μοντελοποίηση μεγάλων όγκων, ευμετάβλητων και υψηλής διασύνδεσης βιβλιογραφικών συνόλων δεδομένων. Στην συνέχεια έγινε αναφορά στις δυνατότητες της γλώσσας ερωτοαποκρίσεων Cypher του Neo4j. Ειδικότερα αναφερθήκαμε στις παρασκηνιακές διαδικασίες που διενεργούνται ώστε ένα ερώτημα Cypherνα εκτελεστεί αποδοτικά. Μετά επισημάνθηκε η ευέλικτη φύση της γλώσσας Cypher του Neo4j και περιγράφηκαν δυο πολύ δημοφιλείς βιβλιοθήκες επέκτασης της λειτουργικότητας του Neo4j: APOC και GDS. Στα πλαίσια της διπλωματικής η πρώτη παρείχε χρήσιμες και συχνά χρησιμοποιούμενες λειτουργικότητες, ενώ η δεύτερη παρείχε πολύ αποδοτικές υλοποιήσεις αλγόριθμων γράφων που μπορούν να χρησιμοποιηθούν για τον υπολογισμό χρήσιμων βιβλιομετρικών μετρικών. Ωστόσο από την αρχή έγινε αντιληπτό ότι η επιτυχημένη επεξεργασία των βιβλιογραφικών δεδομένων με τα παραπάνω χαρακτηριστικά έπρεπε να συνεπικουρηθεί και από μια βιβλιοθήκη Python που θα κατένειμε τις επεξεργασίες στους πολλαπλούς πυρήνες του υπολογιστικού μηχανήματος. Η βιβλιοθήκη αυτή περιγράφεται στο επόμενο κεφάλαιο και αποτελεί τον δεύτερο βασικό τεχνολογικό πυλώνα υλοποίησης της διπλωματικής εργασίας.

# Κεφάλαιο 3ο: Παραλληλοποιημένη Εκτέλεση Python σε Μεγάλα Σύνολα Δεδομένων

## 3.1 Εισαγωγή

Τα χαρακτηριστικά που παρουσιάζουν τα βιβλιογραφικά δεδομένα θέτουν σημαντικές προκλήσεις όσον αφορά την επεξεργασία τους. Για την αποτελεσματική και αποδοτική διαχείριση τους απαιτείται η χρήση μιας βιβλιοθήκης σε μια γλώσσα προγραμματισμού που να είναι κατάλληλη για τη διαχείριση big data. Μια τέτοια βιβλιοθήκη θα πρέπει να εμφανίζει αρετές που θα επιτρέπουν την εύκολη κλιμάκωση και κατανομή των επεξεργασιών σε όλο το φάσμα των κόμβων επεξεργασίας του υπολογιστικού συστήματος. Μάλιστα αυτό θα πρέπει να το επιτυγχάνει με τρόπο διάφανο και μετις ελάχιστες ρυθμίσεις από την μεριά του χρήστη. Ιδανικά ο χρήστης θα πρέπει να εξακολουθεί να θεωρεί ότι όλο το φάσμα των επεξεργασιών λαμβάνουν χώρα τοπικά στο υπολογιστικό του μηχάνημα. Επιπρόσθετα θα πρέπει να υποστηρίζει απροβλημάτιστα την απομακρυσμένη προσπέλαση των δεδομένων, την εκτέλεσή των επεξεργασιών με διάφορους τρόπους, την εύκολή εγκατάστασή της σε ποικιλία διαμορφώσεων κ.α. Στα πλαίσια της διπλωματικής επιλέχθηκε να χρησιμοποιηθεί η βιβλιοθήκη DASK σε γλώσσα προγραμματισμού Python και αποτελεί τον δεύτερο βασικό τεχνολογικό πυλώνα υλοποίησης της βιβλιογραφικής υποδομής.

## 3.2 Η γλώσσα προγραμματισμού Python

Η Python είναι μια διερμηνευόμενη, υψηλού επιπέδου, γενικού σκοπού γλώσσα προγραμματισμού. Σχεδιάστηκε με τη φιλοσοφία να είναι εύκολη στην εκμάθηση και να επιτρέπει στους χρήστες να παράγουν γρήγορα υψηλής ποιότητας κώδικα για μια ευρεία γκάμα εφαρμογών. Συγκεκριμένα ο Python κώδικας που χρησιμοποιεί διάφορα ιδιώματα της γλώσσας (Pythonic Code) είναι πολύ δημοφιλής για την εύκολη αναγνωσιμότητά του, τη συμπαγή σύνταξη, την ταχύτητα εκτέλεσης και τη μικρή επιβάρυνση σε πόρους του υπολογιστή. Η γλώσσα υποστηρίζει όλα τα δημοφιλή στυλ προγραμματισμού, όπως τον δομημένο, διαδικαστικό, συναρτησιακό, αντικειμενοστραφή και λογικό προγραμματισμό. Ο τύπος δεδομένων κάθε αντικειμένου αποφασίζεται δυναμικά κατά τον χρόνο εκτέλεσης και όχι εξαρχής (Dynamically Typed). Επίσης η γλώσσα χρησιμοποιεί έναν συλλέκτη σκουπιδιών GC που επιτρέπει την αυτόματη διαχείριση της μνήμης βάσει του αριθμού των αναφορών σε αυτήν. Έτσι απελευθερώνεται ο προγραμματιστής από την υποχρέωση να κάνει τις κατάλληλες εργασίες διαχείρισης μνήμης, αυξάνοντας την παραγωγικότητά του. Η γλώσσα σχεδιάστηκε εξαρχής ώστε να είναι επεκτάσιμη μέσω του μηχανισμού των ενοτήτων (Modules). Αντί να υλοποιεί ένα μεγάλο μέρος της λειτουργικότητάς της στον πυρήνα, επιλέχτηκε αυτή να υλοποιείται ως ξεχωριστές ενότητες οι οποίες μπορούν εύκολα να ενσωματωθούν σε διαφορετικές εφαρμογές. Όλα τα παραπάνω χαρακτηριστικά καθιστούν τη γλώσσα ιδανική για χρήση σε εφαρμογές με υψηλές αποθηκευτικές και επεξεργαστικές απαιτήσεις, όπως αυτές που παρουσιάζονται στους τομείς της τεχνητής νοημοσύνης, μηχανικής μάθησης, εξόρυξης δεδομένων, big data κ.α [34].

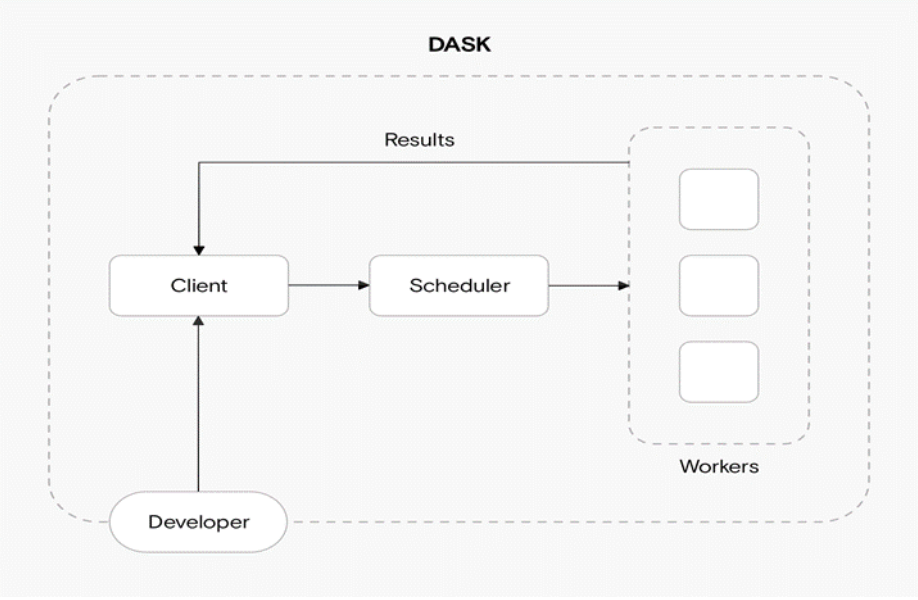
### 3.2.1 Ο μηχανισμός GIL της Python

Πρόκειται για ένα μηχανισμό που δημιουργήθηκε για να προστατέψει τα προγράμματα που τρέχουν παραλληλοποιημένα σε πολλαπλά νήματα εκτέλεσης από συνθήκες ανταγωνισμού (Race Conditions). Ως γνωστόν διαφορετικά κομμάτια κώδικα εκτελούνται με μη ασφαλή τρόπο σε ταυτοχρονικά νήματα εκτέλεσης (Threaded Non-Safe Execution). Αυτό προκύπτει, διότι δεν είναι καθορισμένη η

σειρά προσπέλασης της διαμοιραζόμενης μνήμης από τα νήματα εκτέλεσης με αποτέλεσμα αυτή να αλλάζει με ανεξέλεγκτο τρόπο. Ουσιαστικά με αυτόν τον μηχανισμό παρέχεται εγγύηση ότι σε κάθε χρονική στιγμή μόνο ένα νήμα εκτέλεσης αποκτά πρόσβαση στον διερμηνευτή εντολών της Python και συνεπώς στη μνήμη. Με αυτήν την στρατηγική διασφαλίζεται η ορθή προσπέλαση της μνήμης από όλα τα νήματα εκτέλεσης, αλλά περιορίζει σημαντικά την απόδοση των προγραμμάτων που εκτελούνται ταυτόχρονα. Υπάρχουν κάποιες απαιτητικές λειτουργίες που δεν επηρεάζονται από αυτόν τον μηχανισμό. Σε αυτές συγκαταλέγονται οι λειτουργίες εισόδου/ εξόδου, επεξεργασίας αριθμητικών δεδομένων σε πίνακες NumPy, επεξεργασίες εικόνων κλπ. Για τα υπόλοιπα είδη επεξεργασιών ο μηχανισμός θα πρέπει να λαμβάνεται σοβαρά υπόψη. Υπάρχουν υλοποιήσεις της Python που παρέχουν έναν τέτοιο μηχανισμό (CPython, Cython, PyPy), ενώ κάποιες άλλες εστιάζουν στην πλήρη εκμετάλλευση των πλεονεκτημάτων του ταυτοχρονισμού και δεν τον υποστηρίζουν (Jython, IronPython). Στη βιβλιοθήκη Dask το πρόβλημα επιλύεται επιλέγοντας τον κατάλληλο τύπο δρομολογητή. Θα πρέπει όμως να εντοπίσουμε το κυρίαρχο είδος επεξεργασιών του προγράμματος και να ξεκαθαρίσουμε αν υπόκειται στους περιορισμούς που επιβάλλει ο μηχανισμός GIL.

## 3.3 Η βιβλιοθήκη παραλληλοποιημένης & κατανεμημένης εκτέλεσης Dask

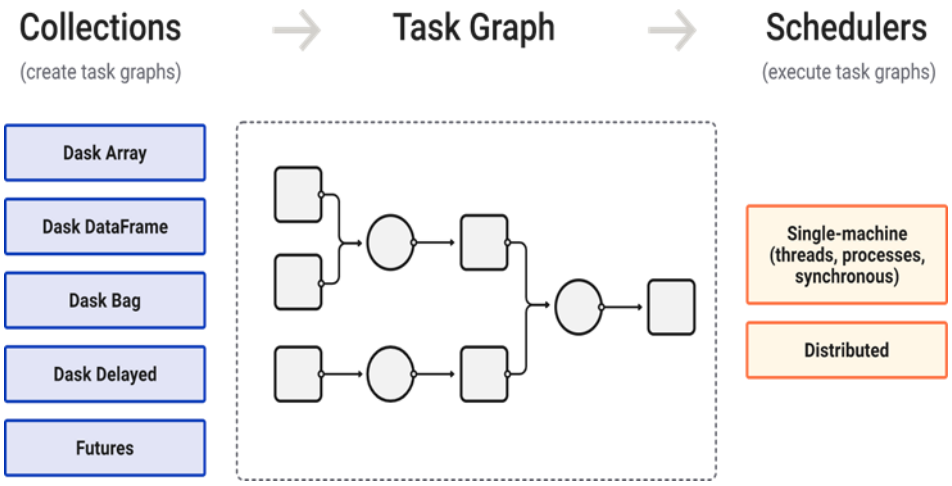
Πρόκειται για μια ευέλικτη βιβλιοθήκη που επιτρέπει την παράλληλη εκτέλεση κώδικα Python επιταχύνοντας σημαντικά τον χρόνο εκτέλεσης μιας εφαρμογής. Η εκτέλεση μπορεί να γίνει, είτε τοπικά χρησιμοποιώντας τους πυρήνες ενός υπολογιστικού μηχανήματος, είτε κατανεμημένα χρησιμοποιώντας τους κόμβους μιας συστοιχίας υπολογιστικών μηχανημάτων (Cluster Infrastructure). Αυτή η κλιμάκωση γίνεται με τρόπο διάφανο προς τον τελικό χρήστη μιας και η βιβλιοθήκη αναλαμβάνει στο παρασκήνιο την πραγματοποίηση όλων των δύσκολων τεχνικών λεπτομερειών. Συγκεκριμένα ένας δρομολογητής διαχωρίζει την αρχική δουλειά σε πολλαπλές παράλληλης εκτέλεσης εργασίες τις οποίες στη συνέχεια δρομολογεί προς εκτέλεση στους κόμβους εργάτες. Στο τέλος συλλέγει τα αποτελέσματα και τα παρουσιάζει στους χρήστες. Η αρχιτεκτονική της βιβλιοθήκης απεικονίζεται στο Σχήμα 3.1.




Σχήμα 3.1: Αρχιτεκτονική βιβλιοθήκης Dask ( Πηγή [35])

Η βιβλιοθήκη επίσης χειρίζεται αποτελεσματικά πολύ μεγάλα σύνολα δεδομένων που δεν μπορούν να αποθηκευτούν ταυτόχρονα στην κύρια μνήμη του υπολογιστή. Αυτό το επιτυγχάνει μέσω υψηλού

επιπέδου δομών δεδομένων που σε λογικό επίπεδο αποτελούν συλλογές πολλών άλλων μικρότερου μεγέθους. Κάθε μια από αυτές τις μικρότερες δομές δεδομένων αποθηκεύονται και εκτελούνται σε διαφορετικό υλικό, ενώ οι υψηλού επιπέδου δομές δεδομένων της βιβλιοθήκης αναλαμβάνουν τον συντονισμό τους. Ο χρήστης πιθανότατα είναι ήδη εξοικειωμένος με αυτές τις δομές μιας και αποτελούν επεκτάσεις δομών πολύ δημοφιλών βιβλιοθηκών (Numpy Arrays, Pandas DataFrames, Python-Toolz – Py Spark Iterators). Όλα αυτά επιτυγχάνονται με τρόπο διάφανο προς τον τελικό χρήστη, ο οποίος νομίζει ότι επενεργεί σε μια ενιαία δομή δεδομένων. Η βιβλιοθήκη βασίζεται στην αρχή της οκνηρής εκτέλεσης (Lazy Evaluation). Σύμφωνα με αυτήν ένα αποτέλεσμα δεν υπολογίζεται αμέσως στο σημείο που καλείται μια εντολή. Αντίθετα η εκτέλεση του αναβάλλεται για ακριβώς τη χρονική στιγμή που θα ζητηθεί (Just-in-time Execution). Για αυτό τον σκοπό κάθε εντολή δεν εκτελείται αμέσως, αλλά αντίθετα δημιουργείται στο παρασκήνιο ένας γράφος εκτέλεσης πολλαπλών εργασιών. Πρόκειται για έναν κατευθυνόμενο ακυκλικό γράφο που περιγράφει οπτικά τη συνταγή εκτέλεσης κάθε εργασίας που πρέπει να ακολουθηθεί όταν θα ζητηθούν τα αποτελέσματα από τον χρήστη. Εσωτερικά ένας τέτοιος γράφος εκτέλεσης εργασιών μπορεί να αναπαρασταθεί ως ένα λεξικό που απεικονίζει κλειδιά σε τιμές ή εργασίες. Μια εργασία μπορεί εσωτερικά να αναπαρασταθεί ως μια συστοιχία (tuple), όπου το πρώτο στοιχείο της είναι το όνομα μιας Python συνάρτησης, και τα υπόλοιπα στοιχεία τα ορίσματα με τα οποία θα κληθεί όταν εκτελεστεί. [8]. Κάνοντας χρήση της παραπάνω αρχής είναι εύκολη η παραλληλοποίηση υπαρχόντων κομματιών κώδικα Python. Στην παραπάνω αρχή βασίζονται τόσο οι υψηλού επιπέδου (Dask Arrays, Bags, DataFrames), όσο και τα χαμηλού επιπέδου (Delayed, Futures) API's της βιβλιοθήκης [36]. Η φιλοσοφία λειτουργίας της βιβλιοθήκης απεικονίζεται στο Σχήμα 3.2.




Σχήμα 3.2: Τρόπος λειτουργίας βιβλιοθήκης Dask ( Πηγή [37])

Όλα τα παραπάνω χαρακτηριστικά καθιστούν την βιβλιοθήκη ιδανική επιλογή για χρήση σε πολύπλοκες ροές επεξεργασίας πολύ μεγάλων δεδομένων σαν αυτές που εμφανίζονται στους τομείς της μηχανικής μάθησης, τεχνητής νοημοσύνης, εξόρυξης δεδομένων κ.α [38].

### 3.3.1 Χρήσιμες δυνατότητες βιβλιοθήκης Dask

Η επιλογή της συγκεκριμένης βιβλιοθήκης βασίστηκε κυρίως στην υποστήριξη σε δομές δεδομένων και προγραμματιστικές τεχνικές που είναι παρόμοιες σε αίσθηση με ορισμένες από τις πιο δημοφιλείς στην γλώσσα προγραμματισμού Python (Numphy Arrays, Pandas Dataframes, Iterators, Partial, Generators). Το πλεονέκτημα όμως που παρουσιάζει είναι ότι διευκολύνει την οριζόντια (Scaling Out) και κατακόρυφη κλιμάκωση (Scaling Up) πολύ απαιτητικών υπολογισμών σε πολλαπλούς κόμβους.

Αυτό μάλιστα το επιτυγχάνει με τρόπο ανεξάρτητο του είδους των υπολογισμών, του τεχνολογικού υποστρώματος των υπολογιστικών κόμβων (Cpu's, Gpu's) καθώς και της αρχιτεκτονικής του υπολογιστικού συστήματος στο οποίο είναι εγκατεστημένη. Η βιβλιοθήκη υποστηρίζει ενδιαφέρουσες δυνατότητες, από τις οποίες οι πιο σημαντικές περιγράφονται παρακάτω.

#### 3.3.1.1 Χαμηλού επιπέδου API's

##### 3.3.1.1.1 Delayed

Πρόκειται για μια χαμηλού επιπέδου προγραμματιστική διεπαφή εφαρμογών που υλοποιεί την έννοια της οκνηρής εκτέλεσης (Lazy Evaluation) ενός υπολογισμού. Σύμφωνα με αυτήν την αρχή, ένας πολύπλοκος υπολογισμός δεν υπολογίζεται αμέσως, αλλά αναβάλλεται για ένα μεταγενέστερο στάδιο στο οποίο θα ζητηθεί με ρητό τρόπο. Αντίθετα στο παρασκήνιο δημιουργείται ένας γράφος εκτέλεσης εργασιών. Σε αυτόν κάθε κόμβος αναπαριστά την εκτέλεση μιας συνάρτησης που υλοποιεί ένα μικρό κομμάτι μιας μεγαλύτερης πιο πολύπλοκης επεξεργασίας. Μια ακμή αναπαριστά το αποτέλεσμα που παράγει ένας κόμβος και διοχετεύεται ως είσοδος στον επόμενο κόμβο επεξεργασίας. Αυτός ο τρόπος εκτέλεσης μιας επεξεργασίας παρουσιάζει σημαντικά πλεονεκτήματα. Καταρχήν, μας επιτρέπει να καθορίσουμε την πλήρη αλυσίδα υπολογισμών που συνιστούν μια πολύπλοκη ροή επεξεργασίας, χωρίς να απαιτείται να περιμένουμε τον υπολογισμό των ενδιάμεσων αποτελεσμάτων. Επίσης πολύπλοκες επεξεργασίες χωρίζονται σε πολλούς επιμέρους μικρότερους υπολογισμούς με αποτέλεσμα να προκαλούν ένα σημαντικά μικρότερο αποτύπωμα στους πόρους του υπολογιστή. Το σημαντικότερο πλεονέκτημα όμως είναι ότι μας επιτρέπει να χειριστούμε αποδοτικά όλα εκείνα τα προβλήματα που εμφανίζονται σε ένα κατανεμημένο περιβάλλον επεξεργασίας. Ουσιαστικά ο γράφος εκτέλεσης που παράγεται αποτελεί μόνο μια αφαιρετική περιγραφή των βημάτων εκτέλεσης ενός πολύπλοκου υπολογισμού χωρίς να προσδιορίζονται πολλές λεπτομέρειες που είναι κρίσιμης σημασίας σε ένα κατανεμημένο υπολογιστικό σύστημα. Σε αυτές συγκαταλέγονται το ποιος κόμβος επεξεργασίας θα εκτελέσει κάθε υπολογισμό, την φυσική θέση αποθήκευσης των απαιτούμενων δεδομένων, τον χειρισμό των ενδιάμεσων αποτελεσμάτων και τον χώρο που αποθηκεύονται τα παραγόμενα αποτελέσματα. Όλα αυτά υπολογίζονται δυναμικά από τον δρομολογητή κατά τον χρόνο εκτέλεσης του ερωτήματος εκτελώντας παρασκηνιακά πολλαπλές βελτιστοποιήσεις που αυξάνουν την αποδοτικότητα εκτέλεσης πολύπλοκων υπολογισμών. Σε αυτό το API βασίζονται όλες οι υψηλού επιπέδου δομές δεδομένων της βιβλιοθήκης (Dask Dataframes, Dask Bags, Dask Arrays). Ωστόσο αποδεικνύεται εξαιρετικά χρήσιμο για την παραλληλοποίηση και παλιότερων τμημάτων κώδικα που υλοποιούν επεξεργασίες που δεν ταιριάζουν φυσικά με τις παραπάνω δομές δεδομένων. Και όλα αυτά απαιτώντας ελάχιστες αλλαγές στον υπάρχοντα κώδικα (Code Base) από τον τελικό χρήστη [36], [39].

##### 3.3.1.1.2 Futures

Πρόκειται για τη δεύτερη προγραμματιστική διεπαφή εφαρμογών API της βιβλιοθήκης και υλοποιεί την έννοια της ανυπόμονης εκτέλεσης (Eager Evaluation) ενός υπολογισμού. Σύμφωνα με αυτήν την αρχή, ένας πολύπλοκος υπολογισμός υπολογίζεται αμέσως όταν ζητηθεί, αλλά το παραγόμενο αποτέλεσμα μπορεί να μην είναι άμεσα διαθέσιμο. Ουσιαστικά ένα future αντιπροσωπεύει ένα απομακρυσμένο αποτέλεσμα του οποίου η εκτέλεση εξελίσσεται στον χρόνο. Όταν το αποτέλεσμα υπολογισθεί παραμένει αποθηκευμένο στον απομακρυσμένο κόμβο εργάτη που ανέλαβε να το εκτελέσει. Το αποτέλεσμα καθίσταται διαθέσιμο στον χρήστη μόνο όταν το ζητήσει. Πρόκειται για ένα εξελιγμένο API που προσφέρει πληθώρα χαρακτηριστικών που έχουν να κάνουν με τον τρόπο υποβολής εργασιών προς εκτέλεση, τη μετακίνηση των απαιτούμενων δεδομένων εισόδου στους κόμβους εκτέλεσης των futures, την αναμονή λήψης των αποτελεσμάτων, τον συντονισμό των

εργασιών με τους πελάτες που τις υπέβαλαν και τους κόμβους εργάτες που ανέλαβαν να τις εκτελέσουν κλπ. Αποτελεί επέκταση των concurrent.futures της Python, αλλά μπορεί να χρησιμοποιηθεί αποκλειστικά με τον κατανεμημένο δρομολογητή (Distributed Scheduler).

#### 3.3.1.2 Πρόσβαση σε απομακρυσμένα σύνολα δεδομένων

Η βιβλιοθήκη υποστηρίζει αυτόματη απομακρυσμένη πρόσβαση σε πολλούς τύπους αποθηκών δεδομένων (Data Stores). Σε αυτούς συγκαταλέγονται τοπικά και διαδικτυακά συστήματα αρχείων, συστήματα αρχείων σε υποδομές υπολογιστικού νέφους (Amazon Web Services, Google Cloud, Microsoft Azure, Apache Hadoop κλπ). Το μόνο που απαιτείται είναι να προσθέσουμε το κατάλληλο κατά περίπτωση πρωτόκολλο ως πρόθεμα στην διαδρομή καταλόγων που οδηγεί στον πόρο που επιθυμούμε να προσπελάσουμε. Στον πίνακα που ακολουθεί (Πίνακας 3.1) αναφέρονται οι πιο δημοφιλείς επιλογές καθώς και οι βιβλιοθήκες από τις οποίες εξαρτώνται.

Πίνακας 3.1: Υποστηριζόμενα μέσα αποθήκευσης βιβλιοθήκης Dask (Πηγή [40])

| Α/ Α | Αποθήκη Δεδομένων | Πρωτόκολλο | Βιβλιοθήκη |
|---|---|---|---|
| 1 | Τοπικό ή Διαδικτυακό Σύστημα Αρχείων | file:// | -- |
| 2 | Amazon S3 | s3:// | s3fs |
| 3 | Google Cloud Storage | gcs:// , gs:// | gcsfs |
| 4 | Microsoft Azure Storage | adl://, abfs:// , az:// | adlfs |
| 5 | Hadoop File System | hdfs:// | (υπάρχει υλοποίηση ενσωματωμένη στην βιβλιοθήκη PyArrow) |
| 6 | HTTP(s) | http:// , https:// | -- |

Στα πλαίσια αυτής της διπλωματικής αρχικά επιλέχθηκε ως μέσο αποθήκευσης των βιβλιογραφικών δεδομένων το GCS. Στους παράγοντες που επηρέασαν την επιλογή μας συγκαταλέγεται η υψηλή αξιοπιστία, διαθεσιμότητα, κλιμάκωση, ασφάλεια, ευρωστία, ευχρηστία καθώς και το πολύ χαμηλό κόστος ανάλογα με την χρήση της συγκεκριμένης υποδομής (Pay As You Use). Ωστόσο στην πορεία της εξέλιξης της εργασίας έγινε αντιληπτό ότι η προσπέλαση των δεδομένων από οποιαδήποτε μηχάνημα εκτός υπολογιστικής υποδομής της Google δημιουργούσε εξερχόμενη κίνηση (Egress Traffic) η οποία χρεώνονταν με σημαντικά ποσά. Συνεπώς, σε πρώτη φάση επιλέχθηκε το σύνολο των δεδομένων να αντιγραφούν σε μια πανομοιότυπη ιεραρχικά εμφωλευμένη δομή σε έναν λογαριασμό GoogleDrive. Οι αποθηκευτικές ικανότητες αυτού του λογαριασμού αναβαθμίστηκαν σε 2 TB, ώστε να καταστεί ικανός να ανταποκριθεί στις αυξημένες αποθηκευτικές ανάγκες της εργασίας. Σε δεύτερη φάση ο λογαριασμός αυτός προσαρτήθηκε τοπικά στο υπολογιστικό μηχάνημα της σχολής. Στο ίδιο υπολογιστικό μηχάνημα έχει εγκατασταθεί ο εξυπηρετητής Neo4j και επιπλέον εκτελείται και η εφαρμογή πελάτης σε Python. Η προσάρτηση έγινε μέσω του προγράμματος ocamlfuse της Google. Το πρόγραμμα αυτό δημιουργεί ένα FUSE σύστημα αρχείων που επιτρέπει την προσάρτηση του Google Drive σε περιβάλλον Linux.

#### 3.3.1.3 Χειρισμός συμπιεσμένων δεδομένων

Η βιβλιοθήκη υποστηρίζει αυτόματη αποσυμπίεση των δεδομένων κατά τη διαδικασία της ανάγνωσης από το μέσο αποθήκευσης. Υποστηρίζει όλα τα δημοφιλή πρότυπα συμπίεσης με την εγκατάσταση της κατάλληλης βιβλιοθήκης. Σε αυτά συγκαταλέγονται gzip, bz2, xz, snappy, και lz4. Στα πλαίσια της διπλωματικής επιλέχθηκε το πρότυπο συμπίεσης lz4. Η επιλογή βασίστηκε στην εξαιρετικά υψηλή ταχύτητα συμπίεσης/ αποσυμπίεσης μιας και μπορεί να κλιμακώνει γραμμικά ανάλογα με τους

πυρήνες του υπολογιστικού συστήματος. Επίσης πρόκειται για ένα μη απωλεστικό αλγόριθμο συμπίεσης (Lossless Compression Algorithm) που πετυχαίνει και πολύ καλό λόγο συμπίεσης (Compression Ratio). Για τους παραπάνω λόγους έχει επικρατήσει ως ο πιο δημοφιλής αλγόριθμος συμπίεσης/ αποσυμπίεσης σε μεγάλα δεδομένα διαδικτύου [41].

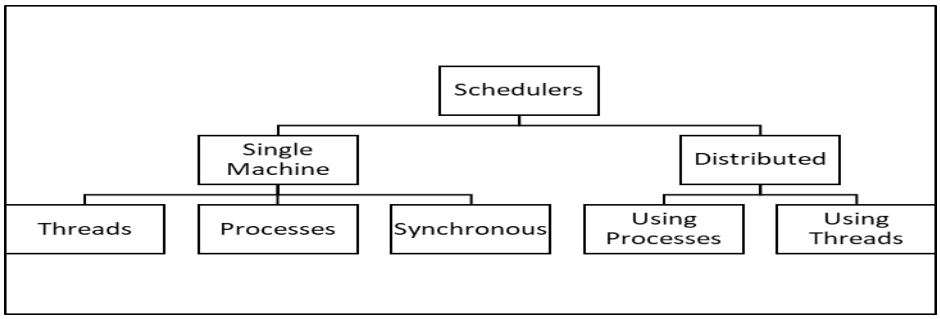
#### 3.3.1.4 Αναζήτηση σε πολύπλοκες δομές καταλόγων

Η βιβλιοθήκη υποστηρίζει την αυτόματη αναζήτηση και εύρεση πολλαπλών αρχείων που ακολουθούν παρόμοια πρότυπα ονοματολογίας. Όλες οι μέθοδοι εισόδου/ εξόδου δεδομένων διαφόρων μορφών (Text, Avro, Json, Csv, Hdf, Parquet) υποστηρίζουν αυτήν τη δυνατότητα μέσω της χρήσης χαρακτήρων μπαλαντέρ (Wild Cards) σε όλη τη διαδρομή καταλόγων προς ένα αρχείο. Κάθε χαρακτήρας μπαλαντέρ αντικαθίσταται κατά την εκτέλεση με ένα κείμενο ή κάποιον αριθμό σύμφωνα με τους κανόνες του κελύφους του Unix (Glob - Unix Style Pathname Pattern Expansion). Είναι γνωστό ότι σε περιπτώσεις πολύπλοκων ροών επεξεργασίας που απαιτούν την προσπέλαση πολλαπλών αρχείων από εξαιρετικά δαιδαλώδεις δομές καταλόγων η εργασία αποδεικνύεται εξαιρετικά χρονοβόρα και προγραμματιστικά επίπονη. Η βιβλιοθήκη απαλλάσσει τον προγραμματιστή από την υποχρέωση συγγραφής του αντίστοιχου κώδικα και του επιτρέπει να επικεντρώσει την προσοχή του στην παραλληλοποίηση των διαδικασιών. Με αυτόν τον τρόπο αυξάνει κατακόρυφα την αποδοτικότητά του και μειώνει τον χρόνο ανάπτυξης των εφαρμογών του. Η δυνατότητα αυτή είναι διαθέσιμη σε όλες τις υπηρεσίες απομακρυσμένης προσπέλασης δεδομένων, εκτός των πρωτοκόλλων http(s).

#### 3.3.1.5 Υποστήριξη πολλαπλών δρομολογητών

Η βιβλιοθήκη υποστηρίζει πολλούς δρομολογητές (Schedulers) με διαφορετικά χαρακτηριστικά απόδοσης. Αποστολή τους είναι ο συντονισμός των πελατών που αιτούνται να εκτελεστούν εργασίες, με τους κόμβους που θα αναλάβουν τελικά την εκτέλεση τους. Ουσιαστικά κάθε δρομολογητής επιτελεί έναν ρόλο ανάλογο με αυτόν ενός τροχονόμου. Διαμοιράζει έξυπνα τα δεδομένα και την εργασία σε διαφορετικούς κόμβους εργάτες, ρυθμίζει τη μεταξύ τους επικοινωνία και συλλέγει τα παραγόμενα αποτελέσματα για να τα παρουσιάσει σε κατάλληλη μορφή στον τελικό χρήστη. Φυσικά ο καθένας το επιτυγχάνει με διαφορετική φιλοσοφία η οποία δεν είναι εφαρμόσιμη σε όλες τις περιπτώσεις. Δεν είναι λίγες και οι περιπτώσεις συνδυαστικής χρήσης των παραπάνω δρομολογητών κατά την εκτέλεση διαφορετικών κομματιών του ίδιου αρχείου κώδικα Python.

Γενικά οι δρομολογητές χωρίζονται σε δυο ομάδες: τοπικός (Single-machine Scheduler) και κατανεμημένος (Distributed Scheduler). Η πρώτη κατηγορία εκτελεί τις ροές επεξεργασίας αποκλειστικά στις διεργασίες (Processes) και τα νήματα εκτέλεσης (Threads) του τοπικού μηχανήματος που εκτελείται ο κώδικας Python. Αντίθετα ο δεύτερος πέρα από το τοπικό μηχάνημα μπορεί να κλιμακώσει οριζόντια (Scaling Out) την εκτέλεση μιας ροής επεξεργασίας στους πολλαπλούς κόμβους μιας συστοιχίας υπολογιστικών κόμβων (Cluster). Η κατηγοριοποίηση απεικονίζεται στο Σχήμα 3.3.




Σχήμα 3.3: Δρομολογητές βιβλιοθήκης Dask

#### 3.3.1.5.1 Τοπικός δρομολογητής νημάτων

Εκτελεί τους υπολογισμούς σε μια μόνο διεργασία χρησιμοποιώντας τα διαθέσιμα νήματα εκτέλεσης του τοπικού υπολογιστικού μηχανήματος. Επειδή τα νήματα εκτέλεσης εκτελούνται στην ίδια διεργασία δεν σπαταλούνται πόροι για μετακινήσεις δεδομένων. Αυτό έχει ως αποτέλεσμα η απόδοση να είναι υψηλή. Ωστόσο για να επιτευχθεί πραγματικός παραλληλισμός θα πρέπει ο εκτελούμενος κώδικας να μην κυριαρχείται από υπολογισμούς σε απλά Python αντικείμενα (λίστες, αλφαριθμητικά, αντικείμενα, ευρετήρια), αλλά από επεξεργασίες αριθμητικών δεδομένων σε υψηλού επιπέδου δομές δεδομένων, όπως NumPy Arrays, Pandas DataFrames κλπ. Αυτός ο περιορισμός επιβάλλεται από τον μηχανισμό GIL που θέτει η Python σε όλα τα αντικείμενά της και στον οποίο έχουμε ήδη αναφερθεί σε προηγούμενη ενότητα. Ο τοπικός δρομολογητής νημάτων είναι ο εξ ορισμού δρομολογητής των υψηλού επιπέδου δομών δεδομένων Dask Array, Dask DataFrame και του χαμηλού επιπέδου API Dask Delayed.

#### 3.3.1.5.2 Τοπικός δρομολογητής διεργασιών

Εκτελεί τους υπολογισμούς στις διαφορετικές διαθέσιμες διεργασίες εκτέλεσης του τοπικού υπολογιστικού μηχανήματος. Αυτή η μεθοδολογία μας προφυλάσσει από το πρόβλημα του GIL που επιβάλλει η Python στα απλά αντικείμενά της, επιτυγχάνοντας πραγματικό παραλληλισμό. Ιδιαίτερη προσοχή όμως πρέπει να επιδειχθεί σε περιπτώσεις επεξεργασιών που περιλαμβάνουν μετακινήσεις πολύ μεγάλων δεδομένων στην κάθε διεργασία. Συνήθως τέτοιες μετακινήσεις είναι χρονοβόρες, υποβαθμίζουν σοβαρά την απόδοση και είναι προτιμότερο να αποφεύγονται. Ο τοπικός δρομολογητής διεργασιών είναι ο εξ ορισμού δρομολογητής της υψηλού επιπέδου δομής δεδομένων Dask Bag.

#### 3.3.1.5.3 Σύγχρονος τοπικός δρομολογητής μοναδικού νήματος εκτέλεσης

Εκτελεί όλους τους υπολογισμούς σε ένα μοναδικό νήμα εκτέλεσης του τοπικού υπολογιστικού μηχανήματος. Δεν επιδεικνύει καθόλου παραλληλισμό και χρησιμοποιείται αποκλειστικά για σκοπούς εκσφαλμάτωσης.

#### 3.3.1.5.4 Κατανεμημένος δρομολογητής

Αντίθετα με ότι δηλώνει το παραπλανητικό του όνομα, ο κατανεμημένος δρομολογητής μπορεί να εκτελέσει τους υπολογισμούς και στις διαθέσιμες διεργασίες/ νήματα εκτέλεσης ενός τοπικού υπολογιστικού μηχανήματος. Είναι πιο σύγχρονος δρομολογητής και πετυχαίνει συνήθως υψηλότερη απόδοση από τους προηγούμενους τοπικούς δρομολογητές. Υλοποιεί το χαμηλού επιπέδου API Futures, παρέχει έναν διαγνωστικό πίνακα με χρήσιμες πληροφορίες απόδοσης του εκτελούμενου κώδικα και λαμβάνει πιο έξυπνες αποφάσεις όσον αφορά τις μετακινήσεις δεδομένων μεταξύ των κόμβων που εκτελούν τους υπολογισμούς. Επιπρόσθετα είναι απόλυτα πατραμετροποιήσιμος όσον

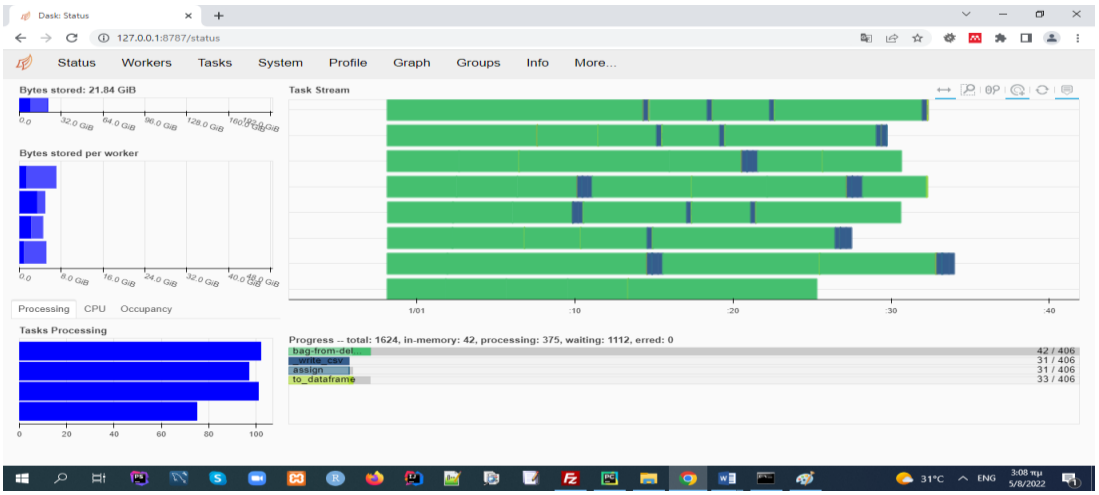
αφορά την ανάθεση συγκεκριμένων επεξεργασιών σε συγκεκριμένους κόμβους επεξεργασίας. Όλα τα παραπάνω χαρακτηριστικά τον καθιστούν την πιο δημοφιλή επιλογή μεταξύ των χρηστών. Εκεί όμως που ο δρομολογητής υπερτερεί είναι στη διαχείριση πολύ απαιτητικών ροών επεξεργασίας σε πολύ μεγάλους όγκους δεδομένων. Σε τέτοιες περιπτώσεις επιβάλλει τον έξυπνο καταμερισμό του φόρτου εργασίας ανάμεσα στους διαφορετικούς κόμβους που απαρτίζουν μια συστοιχία υπολογιστικών συστημάτων. Ο ρόλος του δρομολογητή είναι ο δίκαιος διαμοιρασμός των δεδομένων και των γράφων εκτέλεσης των διαφορετικών χρηστών στους διαφορετικούς κόμβους εκτέλεσης της υπολογιστικής συστοιχίας, η αποδοτική επικοινωνία τους, η συλλογή των αποτελεσμάτων και τελικά η παρουσίασή τους στον τελικό χρήστη. Παράλληλα, τα ασύγχρονα, καθοδηγούμενα από γεγονότα χαρακτηριστικά του τον καθιστούν αποτελεσματικό στη διαχείριση των απαιτήσεων επεξεργασίας ενός μεταβλητού πλήθους χρηστών από ένα μεταβλητό πλήθος κόμβων επεξεργασίας (κόμβοι προστίθενται δυναμικά ή αφαιρούνται λόγω αστοχιών υλικού/ λογισμικού). Όλα τα παραπάνω επιτυγχάνονται με ελάχιστες ρυθμίσεις από τη μεριά του χρήστη, ο οποίος πλέον μπορεί να επικεντρώσει την προσπάθειά του στην αποδοτική παραλληλοποίηση των επεξεργασιών του. Στα πλαίσια της διπλωματικής επιλέχθηκε η αποκλειστική χρήση του κατανεμημένου δρομολογητή με χρήση διεργασιών. Η επιλογή αυτή βασίστηκε στην παρατήρηση ότι όλες οι επεξεργασίες είναι ανεξάρτητες μεταξύ τους και μπορούν να εφαρμοστούν παράλληλα σε διαφορετικά κομμάτια των συνολικών δεδομένων. Άλλος ένας λόγος που επηρέασε τη συγκεκριμένη επιλογή είναι και το γεγονός ότι οι κυρίαρχες επεξεργασίες που εφαρμόστηκαν επηρεάζονται από τον μηχανισμό GIL.

#### 3.3.1.6 Dask dashboard

Σε περιβάλλοντα παράλληλων και κατανεμημένων υπολογισμών η παρακολούθηση της εξέλιξης πολύπλοκων ροών επεξεργασίας καθίσταται εξαιρετικά δύσκολη. Επίσης, οι συνηθισμένες τεχνικές εκσφαλμάτωσης κώδικα Python δεν μπορούν να εφαρμοστούν, μιας και οι επιμέρους υπολογισμοί που απαρτίζουν μια πολύπλοκη επεξεργασία κατανέμονται πλέον σε απομακρυσμένα υπολογιστικά μηχανήματα. Όλα τα παραπάνω καθιστούν αδύνατη την ορατότητα της αποτελεσματικότητας των εκτελούμενων επεξεργασιών. Προκειμένου η βιβλιοθήκη Dask να μας βοηθήσει σε τέτοιες περιπτώσεις παρέχει τον Dask Dashboard. Ουσιαστικά πρόκειται για μια διαδικτυακή εφαρμογή με χρήσιμα στατιστικά στοιχεία της εξέλιξης των υπολογισμών και της χρήσης των πόρων στους πολλαπλούς κόμβους μιας συστοιχίας υπολογιστικών κόμβων. Οι πληροφορίες προβάλλονται συνήθως σε πραγματικό χρόνο και απεικονίζονται σε εύκολα κατανοήσιμα γραφικά σχεδιαγράμματα. Η γραφική διεπαφή του πίνακα είναι προσπελάσιμη μέσω του συνδέσμου http://localhost:8787/status που δημιουργείται αυτόματα όταν δημιουργούμε ένα αντικείμενο Client. Επίσης μπορούμε να δούμε τον σύνδεσμο και μέσω της μεταβλητής client.dashboard_link, όπου client είναι μια μεταβλητή τύπου Client που δημιουργήσαμε προηγουμένως. Μέσω αυτού του πίνακα μπορεί ο χρήστης να εξάγει πολύτιμα συμπεράσματα για την εξέλιξη των υπολογισμών, τη χρήση των πόρων και την αποδοτικότητα των υποκείμενων διεργασιών που διενεργούνται σε κατανεμημένο περιβάλλον. Συγκεκριμένα παρέχει πληροφορίες για την εκτέλεση των διαφόρων εργασιών στους κατανεμημένους κόμβους, την επικοινωνία δεδομένων μεταξύ των κόμβων επεξεργασίας και με τον δρομολογητή, τη χρήση της κύριας μνήμης και του σκληρού δίσκου, το ποσοστό χρήσης της CPU από κάθε κόμβο επεξεργασίας κ.α. Στο σημείο αυτό κρίνεται σκόπιμο να γίνει αναφορά σε κάποια σημαντικά ζητήματα. Πρώτον, ο Dask Dashboard είναι διαθέσιμος μόνο μέσω του κατανεμημένου δρομολογητή (Distributed Scheduler). Δεύτερον, σε ορισμένα κατανεμημένα περιβάλλοντα εκτέλεσης η εξ ορισμού θύρα 8787, μέσω της οποίας μπορεί ο χρήστης να προσπελάσει τη γραφική διεπαφή του Dask Dashboard, μπορεί να είναι μπλοκαρισμένη. Αυτό το πρόβλημα μπορεί να αντιμετωπιστεί απελευθερώνοντας τελείως την κίνηση στην θύρα 8787 με τη συγγραφή κατάλληλων κανόνων

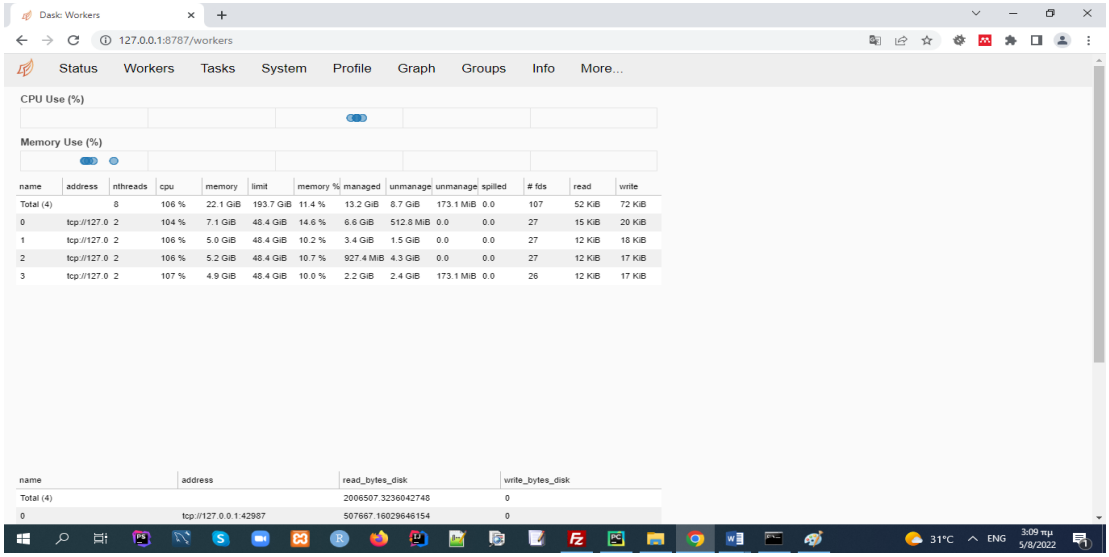
τοίχους προστασίας (Firewall Rules). Επίσης το πρόβλημα αντιμετωπίζεται με τεχνικές SSH port forwarding, ώστε τα πακέτα της θύρας 8787 να ανακατευθύνονται σε άλλη ελεύθερη προς χρήση θύρα. Στα πλαίσια της διπλωματικής το Dask Dashboard κατέστη διαθέσιμο μέσω της τεχνικής ssh port forwarding κατά τη ssh σύνδεση με τον απομακρυσμένο μηχάνημα της σχολής. Τέλος, πρέπει να αναφέρουμε ότι προαπαιτούμενο για τη χρήση του πίνακα είναι και η εγκατάσταση της βιβλιοθήκης bokeh και σε ορισμένες περιπτώσεις (ανάλογα με τη διαμόρφωση της υποκείμενης υποδομής) και η εγκατάσταση του jupyter-server-proxy [42]. Στην συνέχεια απεικονίζονται διάφορα σχεδιαγράμματα του Dask Dashboard με χρήσιμες πληροφορίες κατά την εκτέλεση της εφαρμογής πελάτη στο υπολογιστικό μηχάνημα της σχολής. Όλα τα σχεδιαγράμματα αφορούν τη δημιουργία των αρχείων csv για τους κόμβους Authors του συνόλου δεδομένων ‘Mag’ σε μια τοπική συστοιχία (Local Cluster) τεσσάρων κόμβων επεξεργασίας (Workers) σε κάθε έναν από τους οποίους ανατέθηκαν δυο νήματα εκτέλεσης.

Στην γραφική διεπαφή Dask Dashboard status στο Σχήμα 3.4 απεικονίζεται μια σύνοψη της κατάστασης της υπολογιστικής συστοιχίας σε μια συγκεκριμένη χρονική στιγμή της εκτέλεσης. Συγκεκριμένα στην ενότητα Bytes stored per worker απεικονίζεται το ποσό της κύριας μνήμης που δεσμεύει κάθε κόμβος εργάτης. Στην ενότητα Bytes stored τη συνολική δέσμευση κύριας μνήμης από όλους τους κόμβους της συστοιχίας. Επίσης απεικονίζεται και το ποσό των δεδομένων που η κύρια μνήμη δεν επαρκεί να αποθηκεύσει και διαρρέουν στον σκληρό δίσκο. Αυτό το κομμάτι μνήμησείναι επιθυμητό να είναι ελαχιστοποιημένο μιας και απαιτεί χρονοβόρες προσπελάσεις στη δευτερεύουσα μνήμη που γενικά υποβαθμίζουν την αποδοτικότητα εκτέλεσης μιας εφαρμογής. Το φαινόμενο αυτό αντιμετωπίζεται προσθέτοντας επιπλέον πόρους στην υπολογιστική συστοιχία (Κύρια Μνήμη, Κόμβοι Επεξεργασίας). Στην ενότητα Tasks Processing απεικονίζεται η κατανομή των έτοιμων προς εκτέλεση εργασιών σε κάθε υπολογιστικό κόμβο της συστοιχίας. Από το σχήμα προκύπτει μια σχεδόν ομοιόμορφη κατανομή των εργασιών μεταξύ των διαθέσιμων κόμβων επεξεργασίας και συνεπώς μια βέλτιστη αξιοποίηση του υποκείμενου υλικού. Στην ενότητα Progress απεικονίζεται η πρόοδος της εκτέλεσης κάθε εργασίας. Για κάθε εργασία χρησιμοποιείται και μια ξεχωριστή οριζόντια χρωματισμένη ράβδος. Το χρώμα της κάθε ράβδου βρίσκεται σε συμφωνία με το κάθε χρωματισμένο πλαίσιο της ενότητας Task Stream. Η μπάρα κάθε εργασίας χωρίζεται σε τρία τμήματα. Το πιο δεξιό τμήμα μιας μπάρας έχει γκρι χρώμα και αντιστοιχεί σε εργασίες που είναι έτοιμες προς εκτέλεση, αλλά δεν εκτελούνται λόγω προσωρινής μη διαθεσιμότητας κάποιου κόμβου επεξεργασίας της συστοιχίας. Το πιο αριστερό τμήμα μιας μπάρας έχει διαφανές χρώμα και αντιστοιχεί σε εργασίες που έχει ολοκληρωθεί η εκτέλεσή τους και έχουν αποδεσμεύσει τον χώρο κύριας μνήμης που κατείχαν. Τέλος, το μεσαίο τμήμα μιας μπάρας έχει το ίδιο συμπαγές χρώμα με το αριστερό τμήμα της μπάρας και αντιστοιχεί σε εργασίες που έχουν ολοκληρώσει την εκτέλεσή τους, αλλά εξακολουθούν να δεσμεύουν τον χώρο της κύριας μνήμης που ήταν αποθηκευμένες. Τέλος στην ενότητα Task Stream απεικονίζεται η χρονική ροή εκτέλεσης των διαφόρων εργασιών σε κάθε νήμα εκτέλεσης. Κάθε γραμμή αντιστοιχεί σε ένα νήμα εκτέλεσης, ενώ κάθε χρωματισμένο πλαίσιο σε μια διαφορετική εκτελούμενη εργασία. Ένα πλαίσιο με λευκό χρώμα υποδηλώνει το αντίστοιχο νήμα εκτέλεσης είναι σε ανενεργή κατάσταση. Ένα πλαίσιο με κόκκινο χρώμα υποδηλώνει επικοινωνία μεταξύ των διαφορετικών νημάτων εκτέλεσης. Και οι δυο αυτές περιπτώσεις υποδηλώνουν μη αποδοτική διαμόρφωση της συστοιχίας εκτέλεσης και απαιτούν ριζική αναδιάταξη του υποκείμενου κώδικα, ώστε να εξαλειφθούν.



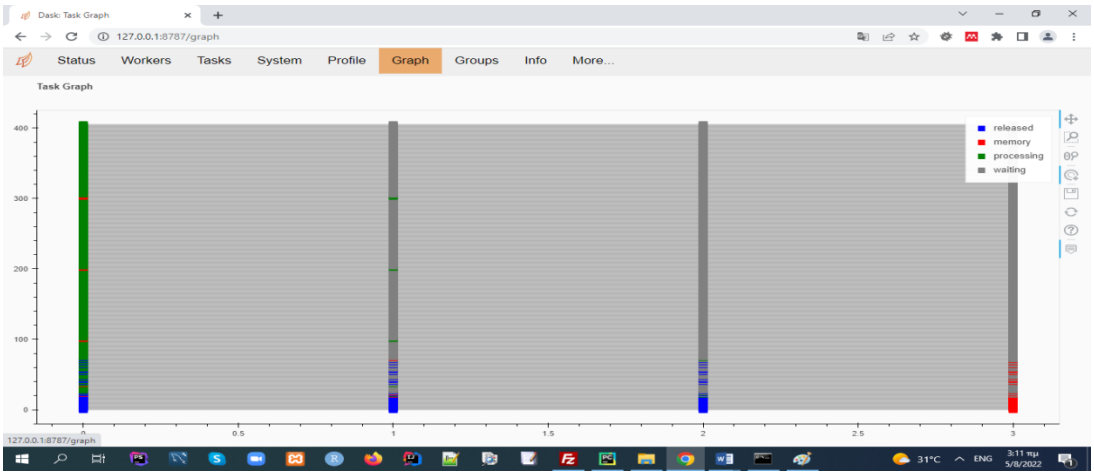

Σχήμα 3.4: Στιγμιότυπο του Dask dashboard status

Στην γραφική διεπαφή Dask Dashboard workers στο Σχήμα 3.5 απεικονίζονται πληροφορίες σχετικά με όλους τους κόμβους επεξεργασίας της συστοιχίας που έχουμε διαμορφώσει. Ανάμεσα στα άλλα απεικονίζει όλους τους κόμβους επεξεργασίας που συνιστούν τη συστοιχία, το πλήθος νημάτων εκτέλεσης καθώς και το ποσοστό χρήσης της CPU και της κύριας μνήμης του κάθε ενός. Από το Σχήμα 3.5 προκύπτει ότι η χρήση της CPU και της μνήμης από κάθε κόμβο εργάτη είναι βέλτιστα διαμοιρασμένη.



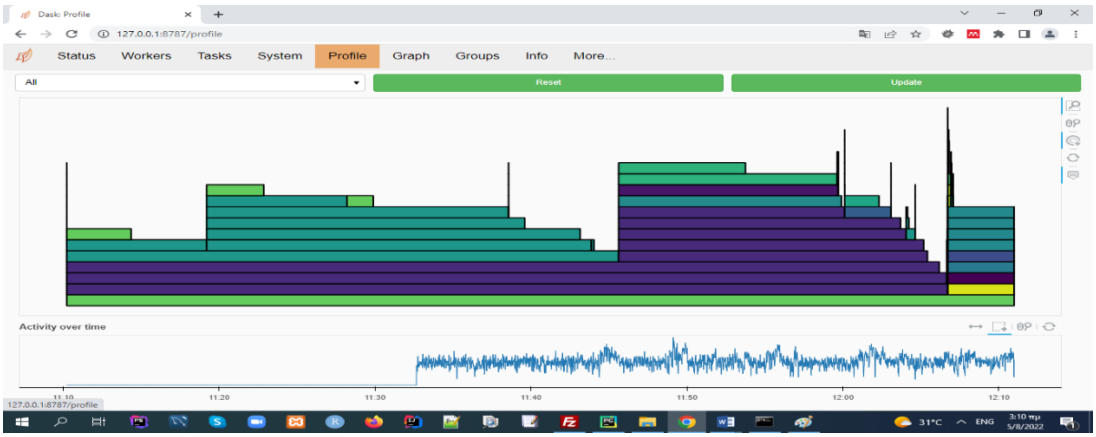

Σχήμα 3.5: Στιγμιότυπο του Dask dashboard workers

Στην γραφική διεπαφή Dask Dashboard graph στο Σχήμα 3.6 απεικονίζεται ο γράφος εκτέλεσης εργασιών για κάθε κόμβο εκτέλεσης καθώς και οι μεταξύ τους συσχετίσεις. Από το σχήμα προκύπτει μηδενική αλληλεξάρτηση των εργασιών κάθε κόμβου εργάτη. Αυτό το γεγονός οδηγεί σε βέλτιστη ταχύτητα εκτέλεσης κάθε εργασίας χωρίς αναμονές για ολοκλήρωση της εκτέλεσης άλλων σχετιζόμενων εργασιών.




Σχήμα 3.6: Στιγμιότυπο του Dask dashboard graph

Στη γραφική διεπαφή Dask Dashboard profile στο Σχήμα 3.7 απεικονίζεται σε πολύ λεπτομερειακό επίπεδο η σύνθεση των υπολογισμών που λαμβάνουν χώρα συνολικά σε όλα τα νήματα εκτέλεσης όλων των εργατών της συστοιχίας. Πρόκειται για ένα διάγραμμα φλόγας (Flame Chart) που απεικονίζει μια στοίβα από ορθογώνια παραλληλόγραμμα, κάθε ένα από τα οποία αναπαριστά μια εργασία. Σε αυτό κάθε εργασία υλοποιείται μέσω μιας Python συνάρτησης, η οποία καλείται μέσα από την συνάρτηση που υλοποιεί την ακριβώς από κάτω εργασία. Σε περίπτωση που επιθυμούμε να επικεντρώσουμε την προσοχή μας σε μια συγκεκριμένη εργασία απλά την επιλέγουμε. Σε περίπτωση που επιθυμούμε να μελετήσουμε το μίγμα των υπολογισμών σε μια συγκεκριμένη χρονική περίοδο απλά την επιλέγουμε από το χρονοδιάγραμμα (Activity Overtime) που βρίσκεται ακριβώς από κάτω. Υπάρχει επίσης και η δυνατότητα να επικεντρώσουμε τη μελέτη μας και σε ένα συγκεκριμένο είδος υπολογισμού απλά επιλέγοντάς το από το πτυσσόμενο μενού που βρίσκεται επάνω αριστερά. Τέλος, ας σημειωθεί ότι το συγκεκριμένο διάγραμμα δεν ενημερώνεται σε πραγματικό χρόνο, αλλά μόνο αφού κάνουμε κλικ στο κουμπί Update επάνω δεξιά.




Σχήμα 3.7: Στιγμιότυπο του Dask dashboard profile

#### 3.3.1.7 Εγκατάσταση σε ποικιλία διαμορφώσεων

Η βιβλιοθήκη διευκολύνει την εύκολη εγκατάσταση της σε μια μεγάλη γκάμα υπολογιστικών συστημάτων, καλύπτοντας ένα ευρύ φάσμα αναγκών. Σε αυτές περιλαμβάνονται εγκατάσταση σε Kubernetes (Deploy DASK on Kubernetes), μέσω Docker (Deploy DASK on Docker), σε υποδομές υπολογιστικού νέφους (Deploy DASK on Amazon Web Services, Google Cloud, Microsoft Azure κ.α.).

## 3.4 Επίλογος

Το κεφάλαιο αυτό αναφέρθηκε στον δεύτερο τεχνολογικό πυλώνα στον οποίο βασίστηκε την υλοποίηση της εφαρμογής. Αυτός περιλαμβάνει τη γλώσσα προγραμματισμού Python και την βιβλιοθήκη παραλληλοποιημένης και κατανεμημένης επεξεργασίας μεγάλων όγκων δεδομένων DASK. Αρχικά παρουσιάστηκαν τα σημαντικότερα χαρακτηριστικά της γλώσσας προγραμματισμού Python, που την καθιέρωσαν ως την κατεξοχήν γλώσσα επεξεργασίας και ανάλυσης πολύ μεγάλων όγκων δεδομένων. Ιδιαίτερη αναφορά έγινε στον μηχανισμό GIL της Python, ο οποίος μπορεί να αποτελέσει σημαντικό ανασχετικό παράγοντα στην επίτευξη υψηλής απόδοσης σε παραλληλοποιημένα περιβάλλοντα εκτέλεσης. Στη συνέχεια αναλύθηκε ο τρόπος εσωτερικής λειτουργίας της βιβλιοθήκης και αναφέρθηκαν τα σημαντικότερα χαρακτηριστικά της. Αυτά τα τελευταία είναι που καθοδήγησαν την τελική απόφασή επιλογής. Ωστόσο η βιβλιοθήκη DASK και η γλώσσα προγραμματισμού Python από μόνεςτους δεν επαρκούν για την επιτυχημένη διαχείριση των βιβλιογραφικών δεδομένων. Από την αρχή έγινε ξεκάθαρο ότι έπρεπε να προσδιοριστεί παράλληλα και μια κατάλληλη ροή επεξεργασίας πολλαπλών σταδίων τύπου ETL κατά τμήματα η οποία θα μετασχημάτιζε τα μπερδεμένα δεδομένα (Messy Data) σε μια πιο ευνοϊκή μορφή που σε επόμενη φάση θα διευκόλυνε την εισαγωγή τους στην βάση δεδομένων γράφου. Αυτή η προσαρμοσμένη στην περίπτωση χρήσης μας ροή επεξεργασίας ETL αποτελεί το αντικείμενο μελέτης του επόμενου κεφαλαίου.

# Κεφάλαιο 4ο: Σχεδιασμός και Ανάπτυξη Εφαρμογής

## 4.1 Εισαγωγή

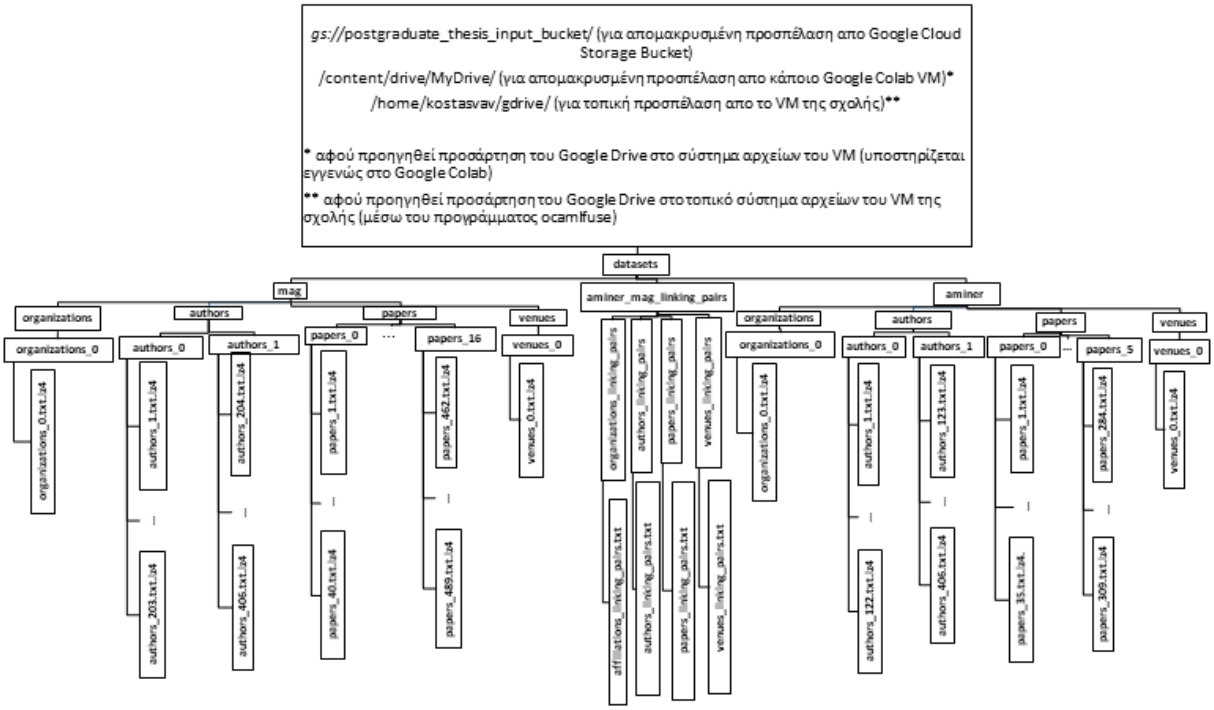
Η επιλογή των δυο τεχνολογικών πυλώνων στην οποία βασίστηκε η υποδομή της διπλωματικής δεν επαρκεί για την υλοποίηση της. Σημαντική προσπάθεια έπρεπε να καταβληθεί και στον σχεδιασμό της, ώστε να υπερκεραστούν εμπόδια σε δυο βασικούς τομείς. Ο πρώτος αφορά στο μέσο αποθήκευσης μεγάλων όγκων δεδομένων. Ο δεύτερος αφορά στο μίγμα των επεξεργασιών που πρέπει να εφαρμοστούν στα δεδομένα αυτά, ώστε να μετασχηματιστούν σε μια πιο κατάλληλη μορφή. Η μορφή αυτή σε μεταγενέστερο χρόνο θα διευκολύνει την εισαγωγή τους στη βάση δεδομένων γράφου. Όσον αφορά τον τομέα της αποθήκευσης πρέπει να σχεδιαστεί μια οργανωτική δομή που θα διευκολύνει την απομακρυσμένη προσπέλασή τους χωρίς να επιβάλλονται χρονικά και χωρικά εμπόδια. Επιπλέον η οργανωτική δομή αποθήκευσης πρέπει να αντανακλά φυσικά τον τρόπο οργάνωσης των δεδομένων στον πραγματικό κόσμο. Όσον αφορά τον τομέα των επεξεργασιών πρέπει να σχεδιάσουμε μια ETL τύπου ροή επεξεργασίας κατά τμήματα. Αυτήν θα αποτελείται από επακριβώς καθορισμένα στάδια κάθε ένα από τα οποία θα υλοποιεί μια συγκεκριμένη αρμοδιότητα. Επειδή όμως η πολυπλοκότητα των επεξεργασιών που πρέπει να εφαρμοστούν στα δεδομένα είναι υψηλή, καθένα από αυτά τα στάδια απαιτεί τον περαιτέρω διαμερισμό του σε υπό στάδια (Top – Down Analysis). Επίσης τα στάδια αυτά πρέπει να εμφανίζουν εξαιρετικά χαμηλό βαθμό σύνδεσης ώστε η εφαρμογή μας να είναι αρθρωτή (Modular).

## 4.2 Γενικά

Καθόλη τη διάρκεια της συγγραφής της, η παρούσα διπλωματική εργασία αντιμετώπιζε δυο προβλήματα. Το πρώτο ήταν η αποτελεσματική και αποδοτική διαχείριση/ αποθήκευση τεράστιων συνόλων δεδομένων. Το δεύτερο ήταν η αποτελεσματική και αποδοτική εκτέλεση πολύπλοκών ροών επεξεργασίας κατά τμήματα σε μεγάλα δεδομένα (Big Data Batch Processing Pipelines). Τα παραπάνω δυο προβλήματα επιδείνωσε η έλλειψη σε τοπικό επίπεδο υπολογιστικού μηχανήματος με σημαντικές αποθηκευτικές και υπολογιστικές δυνατότητες. Ενδεικτικά αναφέρουμε ότι στα αρχικά στάδια της εργασίας χρησιμοποιήθηκε ένας απαρχαιωμένος φορητός υπολογιστής τύπου NoteBook HP Pavilion Sleekbook 15. Ο υπολογιστής αυτός έχει επεξεργαστή IntelPentiumCPU 987 @ 1,50 GHZ, εγκατεστημένη RAM 4 GB και συνολικό σκληρό δίσκο 464 GB, από τα οποία ελεύθερα ήταν μόνο περίπου 100 GB. Το υλικό αυτό έτρεχε σε λειτουργικό σύστημα Microsoft Windows 10 Pro τεχνολογίας 64 bit. Το παραπάνω υπολογιστικό μηχάνημα δεν μπορούσε ούτε σε βασικό επίπεδο να ανταπεξέλθει στις υπολογιστικά απαιτητικές επεξεργαστικές και αποθηκευτικές ανάγκες της διπλωματικής εργασίας. Ωστόσο στην σημερινή εποχή η έλλειψη σε τοπικό επίπεδο ισχυρών αποθηκευτικών και υπολογιστικών δυνατοτήτων δεν αποτελεί πρόβλημα. Πλέον η υπολογιστική και αποθηκευτική ισχύς είναι κινητή, διάχυτη και εύκολα προσπελάσιμη χωρίς γεωγραφικούς, τεχνολογικούς ή οικονομικούς περιορισμούς. Όλα τα παραπάνω έγιναν εφικτά μέσω της παγκόσμιας διάχυσης του διαδικτύου σε κάθε επαγγελματικό, προσωπικό και κοινωνικό τομέα. Η ενσωμάτωση είναι τόσο ισχυρή, που πλέον το διαδίκτυο έχει ξεφύγει από τον παραδοσιακό του ρόλο, ως μια λεωφόρος προσφοράς πληροφοριών. Πλέον έχει μετατραπεί σε μια παγκόσμια, εύκολα προσπελάσιμη πλατφόρμα προσφοράς αποθηκευτικής και υπολογιστικής ισχύος. Η συγκεκριμένη εργασία έκανε συστηματική χρήση της παραπάνω δυνατοτήτων, ώστε να καταφέρει να υπερκεράσει τα προβλήματα που αναφέρθηκαν παραπάνω. Στη συνέχεια ακολουθεί μια αναλυτική περιγραφή των σχεδιαστικών αποφάσεων που ακολουθήθηκαν σε όλη την διάρκεια της εργασίας.

## 4.3 Διαχείριση αποθηκευτικών απαιτήσεων

Μια από τις πρωταρχικές σχεδιαστικές αποφάσεις που έπρεπε να γίνουν είναι η επιλογή της οργανωτικής δομής των συνολικών δεδομένων. Η απόφαση αυτή επηρεάζει αποφασιστικά την αποδοτικότητα, την αποτελεσματικότητα και την ευκολία υλοποίησης των επόμενων σταδίων της εργασίας. Έχει παρατηρηθεί ότι, όταν η επιλεχθείσα οργανωτική δομή αντανακλά φυσικά την δομή των δεδομένων στον πραγματικό κόσμο, τότε τα επόμενα στάδια υλοποιούνται ομαλά. Στα πλαίσια της συγκεκριμένης εργασίας έχει επιλεχθεί η ιεραρχικά εμφωλευμένη δομή που απεικονίζεται στο Σχήμα 4.1.

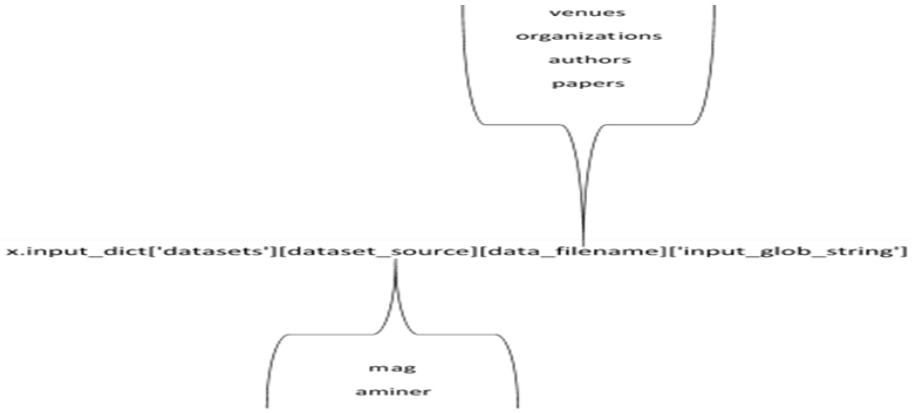



Σχήμα 4.1: Ιεραρχικά εμφωλευμένη οργανωτική δομή συνόλων δεδομένων

Η παραπάνω οργανωτική δομή εκτός του ότι αντανακλά φυσικά την οργανωτική δομή των ίδιων των δεδομένων, διευκολύνει επίσης και το διάβασμα τους από τη βιβλιοθήκη Dask. Συγκεκριμένα αναφέρουμε ότι η ονοματολογία και η δομή των αρχείων και των καταλόγων επιλέχθηκε για να εξυπηρετήσει τον παραπάνω σκοπό. Με αυτόν τον τρόπο απελευθερώνεται ο προγραμματιστής από την υποχρέωση υλοποίησης των αρχικών βασικών σταδίων μιας ροής επεξεργασίας (Εύρεση, Διάβασμα, Αποσυμπίεση Δεδομένων). Συνεπώς μπορεί να επικεντρώσει την προγραμματιστική του προσπάθεια στα επόμενα πιο απαιτητικά στάδια επεξεργασιών. Στα πλαίσια της διπλωματικής εργασίας έχει υλοποιηθεί η παραπάνω ιεραρχικά εμφωλευμένη δομή καταλόγων στην υποδομή υπολογιστικού νέφους της Google (GCS, GD). Από τις παραπάνω επιλογές επιλέχθηκε τελικά η υποδομή Google Drive λόγω της χαμηλότερης χρέωσης. Η επιλογή GCS εγκαταλείφθηκε λόγω της σημαντικής εξερχόμενης κίνησης που προκαλεί και η οποία χρεώνεται με σημαντικά ποσά, όταν προκαλείται από υπολογιστικά μηχανήματα εκτός της υπολογιστικής υποδομής της Google. Στη συνέχεια, η παραπάνω ιεραρχικά εμφωλευμένη δομή καταλόγων προσαρτήθηκε τοπικά στο εικονικό μηχάνημα του πανεπιστημίου, όπου έχει εγκατασταθεί και ο εξυπηρετητής της βάσης δεδομένων Γράφου Neo4j. Επειδή το εικονικό μηχάνημα λειτουργεί σε περιβάλλον Debian-Linux, για την

προσάρτηση χρησιμοποιήθηκε το πρόγραμμα ocamlfuse. Άλλες ισοδύναμες επιλογές σε περιβάλλον Linux αποτελούν το RClone και το GCSF. Αντίθετα, για κάποιο εικονικό μηχάνημα που λειτουργεί σε περιβάλλον windows ή macOS, θα μπορούσε να χρησιμοποιηθεί και το πρόγραμμα RaiDrive ή ισοδύναμα και τα εμπορικά προγράμματα Netdrive, Webdrive κ.α.

Στο σημείο αυτό κρίνεται σκόπιμο να περιγραφεί ο τρόπος προγραμματιστικής υλοποίησης της παραπάνω ιεραρχικά εμφωλευμένης δομής καταλόγων στην εφαρμογή πελάτης Python. Η προγραμματιστική υλοποίηση αφορά δυο όψεις: το αλφαριθμητικό προσπέλασης των οντοτήτων των δεδομένων (Input Glob String) και τις δομές δεδομένων αποθήκευσης των δεδομένων μετα την επεξεργασία τους (Dask Bags, Dask DataFrames). Όσον αφορά την πρώτη όψη, χρησιμοποιήθηκε μια ιεραρχικά εμφωλευμένη δομή λεξικού τεσσάρων επιπέδων. Το κορυφαίο επίπεδο καθορίζει τον κορυφαίο κατάλογο των συνόλων δεδομένων. Δεν χρησιμοποιείται κάποια μεταβλητή, αλλά η στατική αλφαριθμητική τιμή ‘datasets’. Το κατώτερο επίπεδο καθορίζει τη μορφή του input glob string. Δεν χρησιμοποιείται κάποια μεταβλητή, αλλά η στατική αλφαριθμητική τιμή ‘input_glob_string’. Αντίθετα, για τα άλλα δυο ενδιάμεσα επίπεδα χρησιμοποιείται μια μεταβλητή η οποία διαδραματίζει τον ρόλο του διακόπτη. Κάθε μια από αυτές καθορίζει το κομμάτι των συνόλων δεδομένων στο οποίο επιθυμούμε να επικεντρωθούμε. Συγκεκριμένα το δεύτερο επίπεδο ορίζεται από την μεταβλητή dataset_source η οποία ορίζει το σύνολο δεδομένων το οποίο θα προσπελάσουμε. Οι δυνατές τιμές της μεταβλητής είναι ‘mag’, ‘aminer’ ανάλογα με το σύνολο δεδομένων που επιθυμούμε να απομονώσουμε. Το τρίτο επίπεδο ορίζεται από την μεταβλητή data_filename η οποία ορίζει τις οντότητες που θα προσπελάσουμε από το σύνολο δεδομένων το οποίο επιλέξαμε προηγουμένως. Οι δυνατές τιμές της μεταβλητής είναι ‘organizations’, ‘venues’, ‘authors’, ‘papers’ ανάλογα με τις οντότητες του συνόλου δεδομένων που επιθυμούμε να απομονώσουμε. Η προγραμματιστική υλοποίηση της πρώτης όψης περιγράφεται στο Σχήμα 4.2.




Σχήμα 4.2: Προγραμματιστική υλοποίηση πρώτης όψης της ιεραρχικά εμφωλευμένης δομής καταλόγων

Το αποτέλεσμα της προγραμματιστικής υλοποίησης της πρώτης όψης απεικονίζεται στο Σχήμα 4.3.

```
x.input_dict

{'datasets': {
    'aminer': {
      'organizations': {'input_glob_string': '/gdrive/datasets/aminer/organizations/organizations_*/organizations_*.txt.lz4'},
      'authors': {'input_glob_string': '/gdrive/datasets/aminer/authors/authors_*/authors_*.txt.lz4'},
      'papers': {'input_glob_string': '/gdrive/datasets/aminer/papers/papers_*/papers_*.txt.lz4'},
      'venues': {'input_glob_string': '/gdrive/datasets/aminer/venues/venues_*/venues_*.txt.lz4'}},
    'mag': {
      'organizations': {'input_glob_string': '/gdrive/datasets/mag/organizations/organizations_*/organizations_*.txt.lz4'},
      'authors': {'input_glob_string': '/gdrive/datasets/mag/authors/authors_*/authors_*.txt.lz4'},
      'papers': {'input_glob_string': '/gdrive/datasets/mag/papers/papers_*/papers_*.txt.lz4'},
      'venues': {'input_glob_string': '/gdrive/datasets/mag/venues/venues_*/venues_*.txt.lz4'}}}}
```

Σχήμα 4.3: Τα αλφαριθμητικά προσπέλασης όλων των οντοτήτων των συνόλων δεδομένων

Όσον αφορά την δεύτερη όψη, χρησιμοποιήθηκε επίσης μια ιεραρχικά εμφωλευμένη δομή λεξικού τεσσάρων επιπέδων. Τα τρία κορυφαία επίπεδα έχουν πανομοιότυπη λειτουργικότητα με αυτήν που περιγράφηκε στην προγραμματιστική υλοποίηση της πρώτης όψης. Διαφορά υπάρχει μόνο στο κατώτερο επίπεδο. Αυτό περιγράφει τη δομή δεδομένων που θα χρησιμοποιηθεί για την αποθήκευση των επεξεργασμένων δεδομένων της οντότητας του συνόλου δεδομένων που επιθυμούμε να προσπελάσουμε. Συγκεκριμένα το κατώτερο επίπεδο ορίζεται από τη μεταβλητή data_structure. Οι δυνατές τιμές της μεταβλητής είναι ‘db’, ‘ddf’ ανάλογα με την δομή δεδομένων (Dask Bag, Dask DataFrame) που θα χρησιμοποιήσουμε για να αποθηκεύσουμε την επεξεργασμένη οντότητα του συνόλου δεδομένων που επιθυμούμε να απομονώσουμε. Κατά σχεδιαστική παράβαση η μεταβλητή data_structure μπορεί να πάρει και την τιμή ‘npartitions’. Αυτήν εκφράζει το πλήθος των αρχείων στον δίσκο (Number of Partitions) που απαρτίζουν τη δομή δεδομένων της οντότητας του συνόλου δεδομένων που επιθυμούμε να απομονώσουμε. Μια ορθότερη σχεδιαστική επιλογή θα αποτελούσε η χρήση μιας διαφορετικής ιεραρχικά εμφωλευμένης δομής λεξικού που θα την περιλαμβάνει. Η προγραμματιστική υλοποίηση της δεύτερης όψης περιγράφεται στο Σχήμα 4.4.


Σχήμα 4.4: Προγραμματιστική υλοποίηση δεύτερης όψης της ιεραρχικά εμφωλευμένης δομής καταλόγων

### 4.3.1 Google cloud storage

Πρόκειται για μια πολύ δημοφιλή διαδικτυακή υπηρεσία Restfull αρχιτεκτονικής για την αποθήκευση και προσπέλαση δεδομένων στην υποδομή υπολογιστικού νέφους της Google. Βασίζεται στο μοντέλο προσφοράς της υποδομής ως υπηρεσία IaaS. Σύμφωνα με αυτό, ο χρήστης μπορεί να αποθηκεύει και να προσπελάζει πολύ μεγάλους όγκους δεδομένων οποιουδήποτε τύπου στην πληροφοριακή υποδομή της Google, χωρίς να τον απασχολούν λεπτομέρειες, όπως η γεωγραφική θέση και τα τεχνολογικά χαρακτηριστικά της υποκείμενης υποδομής. Επίσης η χρέωση γίνεται ανάλογα με την χρήση, και εξαρτάται από τον συνολικό όγκο και χρονική διάρκεια αποθήκευσης των δεδομένων. Τα δεδομένα αποθηκεύονται σε μια ιεραρχικά εμφωλευμένη δομή που περιλαμβάνει οργανισμούς (Organizations), έργα (Projects), δοχεία (Buckets) και αντικείμενα (Objects). Συνήθως ένας οργανισμός αντιστοιχεί σε μια εταιρεία, ένα έργο σε μια από τις εφαρμογές που αναπτύσσονται εντός της εταιρείας, ένα δοχείο ισοδυναμεί με έναν κατάλογο, ενώ τα αντικείμενα με τα αρχεία που αποθηκεύονται σε κάθε κατάλογο. Ο χρήστης μπορεί εύκολα να αλληλοεπιδράσει με την υπηρεσία με τους εξής τρόπους: με την γραφική διεπαφή της κονσόλας GCC, το εργαλείο της γραμμής εντολών (Gsutil), το REST API (Json, Xml) ή μέσω των βιβλιοθηκών της γλώσσας προγραμματισμού (Client Libraries) που έχουν αναπτυχθεί για την εφαρμογή. Μόλις τα δεδομένα αποθηκευτούν στην υποδομή, η υπηρεσία υποστηρίζει εξελιγμένους μηχανισμούς ασφάλειας και ελέγχου προσπέλασης. Σε αυτούς περιλαμβάνονται ο IAM, που καθορίζει ποιος (Principal) και με ποιο τρόπο (Permissions) μπορεί να προσπελάζει τους πόρους. Η πολιτική ασφάλειας μπορεί να καθοριστεί είτε σε επίπεδο δοχείου, είτε αντικείμενων που περιέχονται σε ένα δοχείο. Συγκεκριμένα σε κάποιους χρήστες (Principals - Individual users, Groups, Domains, the Public) ανατίθενται ρόλοι (σύνολο από Permissions) που με τη σειρά τους καθορίζουν τα δικαιώματα που έχουν επάνω σε συγκεκριμένους πόρους. Η υπηρεσία υποστηρίζει ισχυρή κρυπτογράφηση τόσο στη μεριά του εξυπηρετητή (Server-side Encryption), όσο και στη μεριά του χρήστη (Client-side Encryption). Στην πρώτη περίπτωση τα δεδομένα κρυπτογραφούνται αφού φθάσουν στην υποδομή της Google και πριν αποθηκευτούν στον σκληρό δίσκο. Στην δεύτερη περίπτωση τα δεδομένα κρυπτογραφούνται στη μεριά του χρήστη πριν αποσταλούν την υποδομή της Google. Αφού φθάσουν στον εξυπηρετητή, κρυπτογραφούνται ξανά. Η υπηρεσία υποστηρίζει και άλλους μηχανισμούς ασφάλειας. Σε αυτούς περιλαμβάνονται ο καθορισμός μιας πολιτικής διατήρησης (Retention Policy) σε επίπεδο δοχείου. Σύμφωνα με αυτήν, μπορεί να καθοριστεί μια χρονική περίοδο για ένα δοχείο και μόνο μετά το πέρας της οποίας μπορούν τα περιεχόμενα του να διαγραφούν ή να τροποποιηθούν. Υπάρχει επίσης η δυνατότητα κλειδώματος αυτής της περιόδου, ώστε αυτή να μην μπορεί να μειωθεί ή να αφαιρεθεί. Τέλος, υποστηρίζεται η επαναφορά διαγραμμένων/ τροποποιημένων αντικειμένων μέσω της αποθήκευσης διαφορετικών εκδόσεων τους (Object Versioning).

### 4.3.2 Google drive

Βασίζεται στο μοντέλο προσφοράς του λογισμικού ως υπηρεσία SaaS. Σύμφωνα με αυτό, ο χρήστης μπορεί να ανεβάζει, επεξεργάζεται, αποθηκεύει και διαμοιράζει με άλλους χρήστες τα προσωπικά του αρχεία μέσω μιας απλής γραφικής διεπαφής προσπελάσιμης μέσω ενός διαφυλλιστή ιστού. Η προσπέλαση και ο διαμοιρασμός των αρχείων του Google Drive μπορεί να γίνει και από οποιαδήποτε συσκευή (Desktop, Tablet, Mobile) έχει εγκατεστημένη την εφαρμογή Google Drive. Απλά απαιτείται μια σύντομή διαδικασία συγχρονισμού και όλα τα αρχεία του Google Drive συμπεριλαμβανομένων των αλλαγών που έχουν υποστεί είναι αμφίδρομα διαθέσιμα σε όλες τις συσκευές, εξοικονομώντας πολύτιμο τοπικό αποθηκευτικό χώρο. Όταν το πλήθος των αποθηκευμένων αρχείων αυξάνεται σημαντικά, η υπηρεσία προσφέρει προχωρημένες δυνατότητες αναζήτησης συγκεκριμένων αρχείων

βάσει ιδιοκτήτη, περιεχομένου, ημερομηνία τροποποίησης, τοποθεσίας αποθήκευσης και ονόματος αρχείου. Η υπηρεσία παρέχει προχωρημένες δυνατότητες διαμοιρασμού και συνεργασίας των αποθηκευμένων αρχείων. Συγκεκριμένα μέσω της δυνατότητας διαμοιρασμού μπορούν να καθορίστουν ποια άλλα άτομα μπορούν να προσπελάζουν ένα συγκεκριμένο αρχείο και τι δικαιώματα έχουν πάνω στο αρχείο (απλής ανάγνωσης, σχολιασμού, διόρθωσης). Τέλος παρέχονται δυνατότητες βασικής επεξεργασίας όλων των τύπων εγγράφων διαχειρίζονται τα προγράμματα της σουίτας Office. Σε αυτά περιλαμβάνονται έγγραφα κειμένου, φύλλα επεξεργασίας, παρουσιάσεις διαφανειών, φόρμες, διαφόρων τύπων σχεδιαγράμματα, ιστοσελίδες κ.α.

### 4.3.3 Σύγκριση

Ουσιαστικά πρόκειται για παρόμοιες αλλά όχι ακριβώς ίδιες υπηρεσίες αποθήκευσης αρχείων στην υποδομή υπολογιστικού νέφους της Google. Η κύρια διαφορά έγκειται στο ότι η Google Cloud Storage είναι προσανατολισμένη περισσότερο στις ανάγκες όσων αναπτύσσουν επαγγελματικές εφαρμογές. Αντίθετα, το Google Drive είναι προσανατολισμένο περισσότερο στις προσωπικές ανάγκες του μέσου χρήστη. Για αυτό το GCS παρέχει χαρακτηριστικά που είναι σημαντικά για έναν επαγγελματία, όπως πλήρως προσαρμόσιμα δικαιώματα IAM, δημιουργία πολλαπλών αντιγράφων ασφάλειας βάση περιοχής (Geo-redundancy), ευέλικτη χρέωση ανάλογα με τη χρήση (Pay as you Go), υψηλή διαθεσιμότητα των δεδομένων και κλιμάκωση των αναγκών αποθήκευσης ανάλογα με τη ζήτηση. Αντίθετα το Google Drive παρέχει σταθερά προγράμματα μηνιαίας χρέωσης, ένα εσωτερικό εργαλείο συγχρονισμού των αρχείων που περιέχει με τις εκδόσεις των αρχείων σε άλλους υπολογιστές, καμία δυνατότητα δημιουργίας πολλαπλών αντιγράφων ασφαλείας και έναν βασικό μηχανισμό ασφάλειας βάσει των κωδικών σύνδεσης σε έναν λογαριασμό ηλεκτρονικού ταχυδρομείου Gmail. Επιπρόσθετα, παρέχει βασικές δυνατότητες επεξεργασίας των περισσότερων αρχείων που μπορούν να επεξεργαστούν στην σουίτα εφαρμογών Office και δυνατότητες εύκολου διαμοιρασμού των αρχείων μεταξύ των διαφόρων χρηστών.

## 4.4 Διαχείριση υπολογιστικών απαιτήσεων

Τα σύνολα δεδομένων που κληθήκαμε να χειριστούμε στα πλαίσια της διπλωματικής εργασίας ήταν τεράστια. Επίσης οι επεξεργασίες που απαιτήθηκαν να εφαρμοστούν σε αυτά ήταν εξαιρετικά πολύπλοκες. Λαμβάνοντας υπόψιν και τους περιορισμούς πόρων (υπολογιστικοί, αποθηκευτικοί, χρονικά όρια χρήσης) που επιβάλλει οποιαδήποτε υπολογιστική πλατφόρμα, η εφαρμογή τέτοιων επεξεργασιών σε σύνολα δεδομένων τέτοιου μεγέθους ήταν ανέφικτη. Για την αντιμετώπιση του προβλήματος εφαρμόστηκαν διάφορες τεχνικές μείωσης του μεγέθους των προς επεξεργασία δεδομένων (Data Reduction Techniques). Συγκεκριμένα εφαρμόστηκε εκτεταμένα η τεχνική της μείωσης της διάστασης των δεδομένων (Dimensionality Reduction). Από τα πρώτα στάδια κάθε ροής επεξεργασίας στις βασικές δομές δεδομένων της βιβλιοθήκης (Dask Data Frame, Bag) επιδιώχθηκε η μείωση των χαρακτηριστικών αποκλειστικά στα απολύτως απαραίτητα. Αυτό είχε ως αποτέλεσμα τη συρρίκνωση του όγκου των δεδομένων που έπρεπε να μετακινηθούν μεταξύ των διαφόρων σταδίων μιας σύνθετης ροής επεξεργασίας. Επιπρόσθετα σε όλη την έκταση του κώδικα καταβλήθηκε προσπάθεια να περιοριστεί η χρήση μεγάλων DataFrames ως παράμετροι στις διάφορες ρουτίνες που υλοποιούν ενδιάμεσους υπολογισμούς. Συγκεκριμένα κάθε ενδιάμεση ρουτίνα έχει ως παραμέτρους μόνο τις ελάχιστες πληροφορίες που απαιτούνται για την προσπέλαση της δομής που περιέχει τα δεδομένα και όχι τα ίδια τα δεδομένα. Η προσπέλαση των δεδομένων γίνεται μόνο στη ρουτίνα που υλοποιεί το τελευταίο στάδιο μιας ροής επεξεργασίας. Οι παραπάνω βελτιστοποιήσεις επιβλήθηκαν λόγω της κατανεμημένης φύσης της βιβλιοθήκης, που κατανέμει επεξεργασίες και δεδομένα στους

διαθέσιμους κόμβους εκτέλεσης της υπολογιστικής υποδομής. Η μη εφαρμογή τους θα είχε ως αποτέλεσμα την δημιουργία μιας χιονοστιβάδας δεδομένων που θα έπρεπε να μετακινηθούν μεταξύ του δρομολογητή και τους κόμβους εργάτες καθώς και μεταξύ των ενδιάμεσων σταδίων ενός πολύπλοκου υπολογισμού, υποβαθμίζοντας σημαντικά την απόδοση. Επιπρόσθετα, η ίδια η βιβλιοθήκη Dask επέτρεψε την εύκολη παραλληλοποίηση των επεξεργασιών στου πολλαπλούς πυρήνες ενός εικονικού μηχανήματος. Η βιβλιοθήκη επιτρέπει την αυτόματη κατάτμηση των δεδομένων σε πολλαπλά τμήματα/ αρχεία (Partitions) και την αποδοτική δρομολόγησή τους στους διάφορους κόμβους εργάτες. Τα παραπάνω σε συνδυασμό με τα χαμηλού επιπέδου API's (Delayed, Futures) και τις αποδοτικές δομές δεδομένων υψηλού επιπέδου επιτρέπουν τη δημιουργία CPU bound εφαρμογών. Τέτοιου είδους εφαρμογές «στρεσάρουν» στο έπακρο τους διαθέσιμους πόρους ενός υπολογιστικού μηχανήματος και συνεπώς εμφανίζουν σημαντικά μικρότερους χρόνους εκτέλεσης [36], [39].

Οι παραπάνω βελτιστοποιήσεις επέδρασσαν θετικά στην αποδοτικότητα εκτέλεσης της εφαρμογής. Ωστόσο ο όγκος των δεδομένων εξακολουθούσε να είναι σημαντικός. Οι δυνατότητες του εικονικού μηχανήματος που παρέχει η δωρεάν έκδοση της πλατφόρμας Kaggle αδυνατούσε να ολοκληρώσει τις πολύπλοκες επεξεργασίες σε λογικά χρονικά διαστήματα. Συγκεκριμένα, αναφέρουμε ότι η δωρεάν έκδοση του εικονικού μηχανήματος που παρέχει η πλατφόρμα περιλαμβάνει έναν επεξεργαστή Intel(R) Xeon(R) CPU @ 2.30GHz (1 CPU), με δυο πυρήνες (2 Cores/ CPU) και με δυο νήματα εκτέλεσης ανά πυρήνα (2 Threads/ Core). Επίσης, η διαθέσιμη RAM είναι 16 GB, ενώ ο διαθέσιμος σκληρός δίσκος περίπου 70 GB. Το μηχάνημα αυτό χρησιμοποιήθηκε κυρίως στην φάση της ανάπτυξης της εφαρμογής και αποδείχτηκε εξαιρετικά χρήσιμο στην εξσφαλμάτωση του κώδικα που έτρεχε σε αντιπροσωπευτικά υποσύνολα των συνόλων δεδομένων. Ωστόσο για την τελική δοκιμή της εφαρμογής χρησιμοποιήθηκε ένα εικονικό μηχάνημα από την υπολογιστική υποδομή της σχολής. Συγκεκριμένα, χρησιμοποιήθηκε ένα εικονικό μηχάνημα που περιλάμβανε οκτώ εικονικούς επεξεργαστές Intel(R) Xeon(R) CPU E5620 @ 2.40GHz (8 vCPU's), με ένα φυσικό πυρήνα ανά υποδοχή μητρικής (1 pCores/ Socket), με οκτώ νήματα εκτέλεσης ανά πυρήνα (8 Threads/ Core). Επίσης η διαθέσιμη RAM ήταν 58GB, ενώ ο διαθέσιμος σκληρός δίσκος περίπου 689 GB. Τα παραπάνω στοιχεία ανακτήθηκαν με την εκτέλεση στην γραμμή εντολών της παρακάτω εντολής.

```
lscpu
```

Ο σωρευτικός χώρος αποθήκευσης που καταλαμβάνουν τα ειδικής μορφής αρχεία csv στον σκληρό δίσκο υπολογίστηκε με την εκτέλεση στην γραμμή εντολών της παρακάτω εντολής.

```
du -sh data/produced_csv_files/mag/{entity_type}/{entity}/
```

* Αν entity_type = nodes Τότε entity = venues, organizations, authors, papers, fieldstudies

* Αν entity_type = relationships Τότε entity = IS_PUBLISHED_IN, CITES, HAS_AUTHOR, IS_RELATED_TO_FIELD_STUDY, IS_AFFILIATED_AT_ORGANIZATION

Με αυτό το ισχυρό μηχάνημα κατέστη δυνατή η ολοκλήρωση της δημιουργίας της βάσης δεδομένων γράφου, με απόδοση που περιγράφεται στον παρακάτω πίνακα (Πίνακας 4.1).

Πίνακας 4.1: Χρονικές & αποθηκευτικές απαιτήσεις δημιουργίας αρχείων csv συνόλου δεδομένων Mag (στο εικονικό μηχάνημα της σχολής)

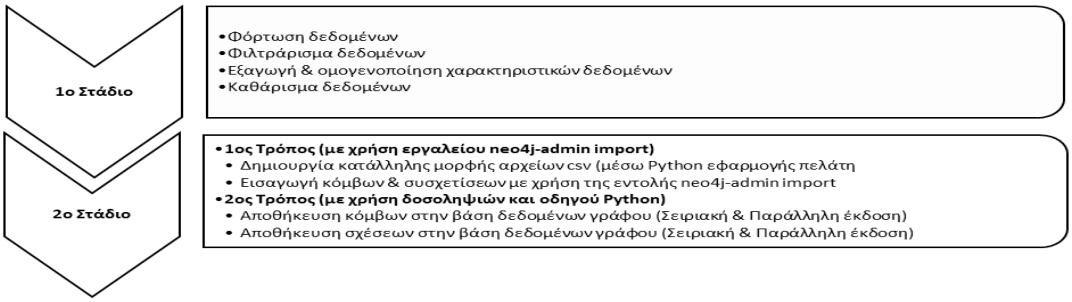
| **Χρονικές & Αποθηκευτικές Απαιτήσεις Δημιουργίας CsvΑρχείων Συνόλου Δεδομένων Mag (InitialGraph Data Model) (Time and Storage Requirements for Creating Mag Dataset Csv Files)** | | | | | | |
|---|---|---|---|---|---|---|
| **Κόμβοι (Nodes)** | | | | | | |
| **A/A** | **Ετικέτα Κόμβου** | **Συνολικό Πλήθος Κόμβων (1 Κόμβος/ Γραμμή Csv)** | **Μέγιστο Πλήθος Ιδιοτήτων/ Κόμβο** | **Συνολικός Χρόνος Δημιουργίας Αρχείων Csv (Total Time to Create Csv Files)** / **Κατανεμημένος Δρομολογητής (Distributed Scheduler)** / **Processes (4 workers, 2 Threads/ Worker) (Parallel Version)** | **Συνολικός ΑποθηκευτικόςΧώρος Αρχείων Csv (Total Csv Files Storage)** / **Ασυμπίεστα Αρχεία (Uncompressed Files)** | **Απόδοση Δημιουργίας Αρχείων csv (Πλήθος Κόμβων – Γραμμών/ Sec)** / **Processes (4 workers, 2 Threads/ Worker) (Parallel Version)** |
| 1 | Venue | 53422 | 3 | 00d:00h:00m:6.68s | 3.2 MB | 7997.31 |
| 2 | Organization | 25776 | 2 | 00d:00h:00m:3.18s | 1.3 MB | 8105.66 |
| 3 | Author | 243477150 | 2 | 00d:04h:16m:45.89s | 7.5GB | 15804.16 |
| 4 | Paper | 89677467** | 4 | 00d:04h:21m:24.51s | 12 GB | 5717.58 |
| 5 | FieldStudy | 671751* | 1 | 00d:09h:37m:34.17s | 20 MB | 19.38 |
| | Σύνολο | 333905566 | 12 | 00d:18h:15m:54.43s | 19GB | 5078.07 |
| **A/A** | **ΤύποςΣχέσης** | **Συνολικό Πλήθος Σχέσεων** | **Μέγιστο Πλήθος ιδιοτήτων/ Σχέση** | **Συνολικός Χρόνος Αποθήκευσης στην Βάση Δεδομένων** / **Distributed Scheduler** / **Processes (4 workers, 2 Threads/ Worker) (Parallel Version)** | **Συνολικός ΑποθηκευτικόςΧώρος Αρχείων Csv (TotalCsvFilesStorage)** / **Ασυμπίεστα Αρχεία (Uncompressed Files)** | **Απόδοση Δημιουργίας Αρχείων csv (Πλήθος Σχέσεων – Γραμμών / Sec)** / **Processes (4 workers, 2 Threads/ Worker) (Parallel Version)** |
| 1 | IS_PUBLISHED_IN | 69064868 | 2 | 00d:04h:44m:49.83s | 4.1 GB | 4041.26 |
| 2 | CITES | 1301545464 | 0 | 00d:04h:20m:53.29s | 34 GB | 83148.36 |
| 3 | HAS_AUTHOR | 642370608 | 1 | 00d:05h:24m:33.09s | 21 GB | 32987.61 |
| 4 | IS_RELATED_TO_FIELD_STUDY | 681400897 | 1 | 00d:04h:46m:06.93s | 38 GB | 39692.65 |
| 5 | IS_AFFILIATED_AT_ORGANIZATION | 165313597 | 0 | 00d:03h:35m:08.37s | 7.8GB | 12806.70 |
| | Σύνολο | 2859695434 | 4 | 00d:22h:51m:31.51s | 104.9GB | 34750.80 |

* Αφορά την εισαγωγή αποκλειστικά μοναδικών τιμών του πεδίου πρωτεύοντος κλειδιού name των FieldStudies

** Αφορά την εισαγωγή φιλτραρισμένων Papers. Συγκεκριμένα επιλέχθηκαν Papersπου περιλαμβάνουν μη κενή τιμή στο πεδίο doiκαι στο πεδίο doc_type δεν περιέχουν την τιμή "patent"

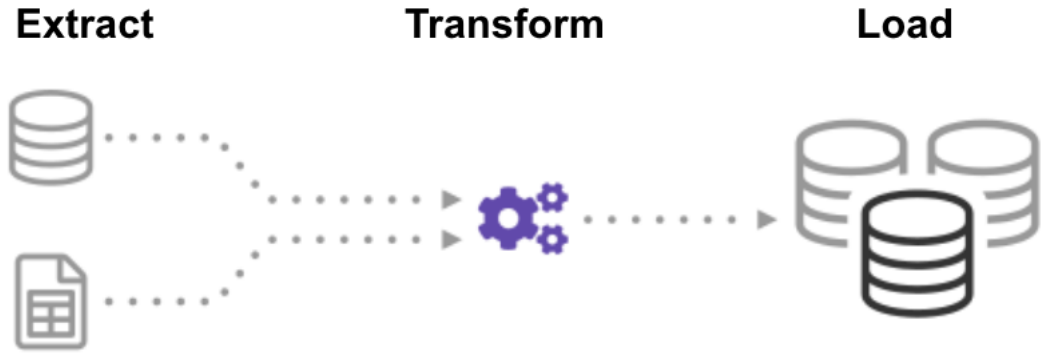
Ο μεγάλος όγκος δεδομένων, η πολυπλοκότητα των επεξεργασιών, καθώς και οι αρχές λειτουργίας της βιβλιοθήκης DASK επέβαλε να εφαρμόσουμε μια πολύπλοκη ροή επεξεργασιών δυο βασικών σταδίων (Two Basic Stages Processing Pipeline). Το πρώτο στάδιο (Raw Data Digest) βασίστηκε στην αρχή της οκνηρής εκτέλεσης (Lazy Evaluation) και αποσκοπεί στην φόρτωση των αρχικών δεδομένων σε κατάλληλες δομές δεδομένων υψηλού επιπέδου της βιβλιοθήκης. Ουσιαστικά απαρτίζεται από πολλά επιμέρους υπό στάδια κάθε ένα από τα οποία μετατρέπει σταδιακά τα δεδομένα σε μια πιο κατάλληλη μορφή που θα αποτελέσει την είσοδο για το επόμενο δεύτερο βασικό στάδιο επεξεργασίας. Αρχικά περιλαμβάνει τη φόρτωση των ανεπεξέργαστων δεδομένων ως έγκυρα αντικείμενα Json από το απομακρυσμένο μέσο αποθήκευσης Google Drive που προσαρτήθηκε τοπικά στο υπολογιστικό μηχάνημα της σχολής που εκτελούνταν η εφαρμογή πελάτης σε Python. Μετά στο επόμενο υπό στάδιο φιλτράρονται από μη επιθυμητές εγγραφές βάση κατάλληλων κριτηρίων που υπαγορεύουν οι προδιαγραφές της εργασίας. Στη συνέχεια αυτά αποτελούν την είσοδο του επόμενου υπό σταδίου, όπου εξάγονται τα σημαντικότερα χαρακτηριστικά με παράλληλη ομογενοποίησή τους (σε περίπτωση που υπάρχουν ελλείπουσες τιμές αντικαθίστανται με εξ' ορισμού τιμές). Τέλος, καθαρίζονται από μη έγκυρους χαρακτήρες. Η παραπάνω ροή επεξεργασίας υλοποιείται μέσω της δομής δεδομένων υψηλού επιπέδου Dask Bag το οποίο στη συνέχεια μετατρέπεται σε ένα Dask DataFrame πολλών διαμερίσεων (partitions), ένα για κάθε αρχείο ανεπεξέργαστων δεδομένων στο μέσο αποθήκευσης. Το δεύτερο στάδιο (Processed Data Database Insertion) αποσκοπεί στο να εισάγει τα επεξεργασμένα δεδομένα του πρώτου σταδίου στη βάση δεδομένων γράφου, ακολουθώντας δυο προσεγγίσεις. Στην πρώτη προσέγγιση επιχειρείται η άμεση εισαγωγή των δεδομένων μέσω του επίσημα υποστηριζόμενου από το Neo4j οδηγού Python. Στη δεύτερη προσέγγιση επιχειρείται η

έμμεση εισαγωγή των δεδομένων μέσω της εντολής import του εργαλείου διαχείρισης neo4j-admin. Και στις δυο προσεγγίσεις ακολουθήθηκαν οι βέλτιστες πρακτικές του Neo4j και διαχωρίστηκε η διαδικασία εισαγωγής των κόμβων από την αντίστοιχη διαδικασία εισαγωγής των συσχετίσεων. Η παραπάνω ροή επεξεργασίας απεικονίζεται στο Σχήμα 4.5.




Σχήμα 4.5: Ροή επεξεργασίας δυο σταδίων για εισαγωγή συνόλου δεδομένων Mag στην βάση δεδομένων γράφου

Η παραπάνω ροή επεξεργασίας που επιλέχθηκε για τη διαχείριση των υπολογιστικών απαιτήσεων της εφαρμογής αποτελεί μια τροποποιημένη υλοποίηση της παραδοσιακής ροής επεξεργασίας ETL κατά τμήματα (ETL Processing Pipeline in Batches). Η τελευταία αποτελεί μια αυτοματοποιημένη διαδικασία η οποία διαβάζει τα ανεπεξέργαστα δεδομένα, εξάγει από αυτά την πληροφορία που είναι χρήσιμη για την ανάλυσή, τη μετασχηματίζει σε μια μορφή κατάλληλη για τις επιχειρηματικές ανάγκες και τέλος την αποθηκεύει σε ένα μέσο αποθήκευσής. Τυπικό χαρακτηριστικό κάθε τέτοιας ροής επεξεργασίας είναι η από τα πρώτα στάδια ισχυρή συμπύκνωση των προς αποθήκευση δεδομένων ώστε να μειωθεί το μέγεθός τους και να αυξηθεί η απόδοση. Άλλες αρετές που συνέβαλαν στην επιλογή της είναι η απλότητά, η ξεκάθαρη ακολουθιακή δομή και ο χαμηλός βαθμός σύζευξης των επιμέρους τμημάτων που την απαρτίζουν. Η δομή μιας ροής επεξεργασίας ETL απεικονίζεται στο Σχήμα4.6.




Σχήμα4.6: Τα τμήματα μιας ροής επεξεργασίας ETL – Extract Transform Load (Πηγή [43])

## 4.5 Εισαγωγή δεδομένων στην βάση δεδομένων γράφου

### 4.5.1 Μέσω δοσοληψιών και οδηγού Python

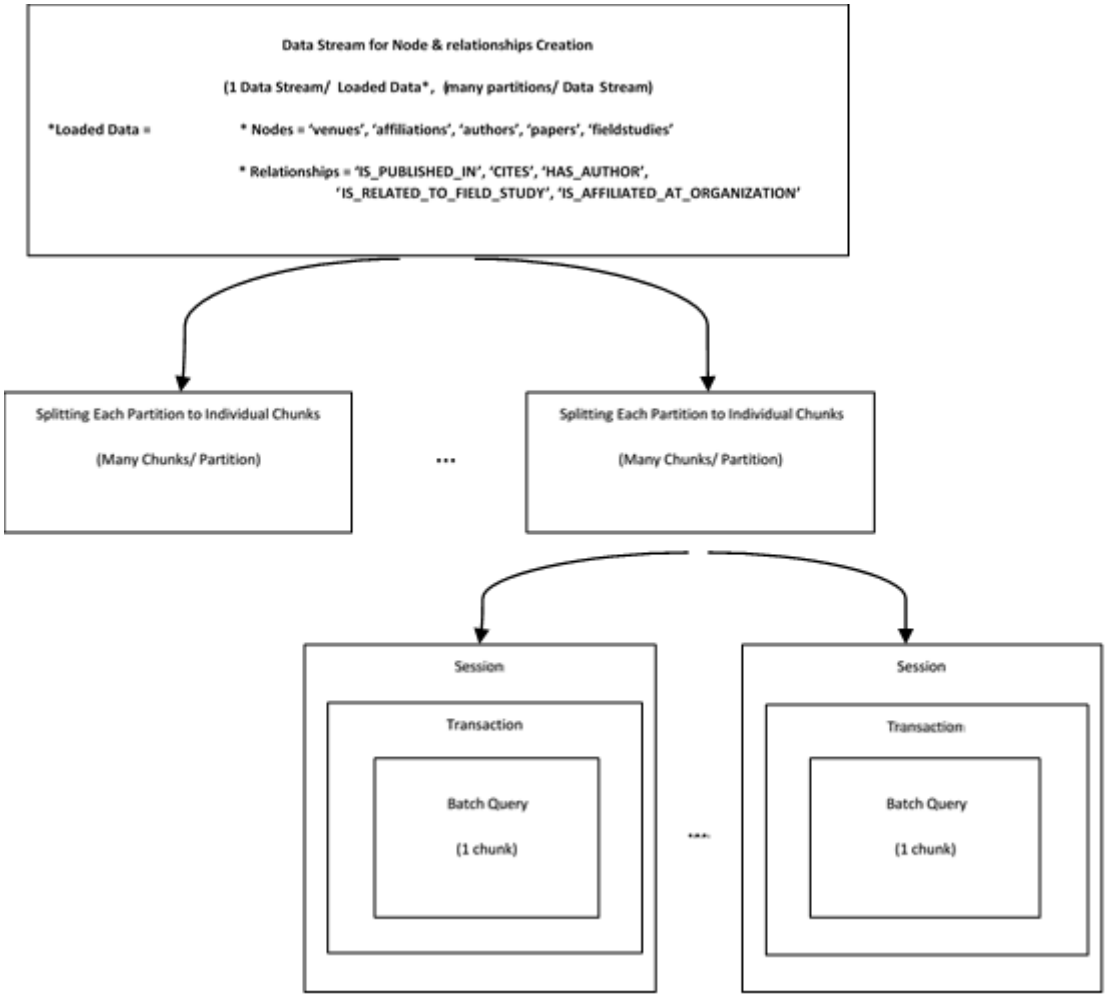
#### 4.5.1.1 Ροή εργασίας οδηγού Python

Η εφαρμογή πελάτης αλληλοεπιδρά με τη βάση δεδομένων Neo4j μέσω ενός αντικειμένου Driver. Πρόκειται για ένα αντικείμενο που συνιστά τη ραχοκοκαλιά της εφαρμογής μιας και διεκπεραιώνει όλη τη ροή εργασιών αλληλεπίδρασης με την βάση δεδομένων. Γενικά η πρόσβαση ενός χρήστη με

μια βάση δεδομένων Neo4j οργανώνεται σε συνόδους εργασίας (Sessions), δοσοληψίες (Transactions) και ερωτήματα (Queries). Μια σύνοδος συνιστά μια δεξαμενή, όπου μπορούν να εκτελούνται σειριακά πολλαπλές δοσοληψίες. Κάθε σύνοδος αναφέρεται σε μια βάση δεδομένων, που μπορεί να καθοριστεί κατά τη δημιουργία του αντικειμένου της συνόδου. Το χαρακτηριστικό αυτό είναι ιδιαίτερα σημαντικό στην Neo4j EE που επιτρέπει την αλληλεπίδραση με πολλαπλές βάσεις δεδομένων. Αντίθετα στην Neo4j CE κάθε σύνοδος αναφέρεται μόνο στην εξ' ορισμού βάση δεδομένων neo4j, που είναι και η μοναδική βάση με την οποία μπορούμε να αλληλοεπιδράσουμε. Με την εκκίνηση μιας δοσοληψίας η σύνοδος ανακτά μια σύνδεση από τη δεξαμενή συνδέσεων του αντικειμένου Driver. Η σύνδεση αυτή χρησιμοποιείται για όσο η σύνοδος εκτελεί τη δοσοληψία. Σε περίπτωση επιτυχούς ολοκλήρωσης ή απόρριψής (Transaction Rolled Back) της δοσοληψίας, η σύνδεση απελευθερώνεται. Όταν κλείνει μια δοσοληψία απελευθερώνονται όλες οι συνδέσεις που χρησιμοποιεί και οποιαδήποτε ενεργές μη ολοκληρωμένες δοσοληψίες απορρίπτονται. Μια δοσοληψία αποτελεί μια ατομική μονάδα εργασίας (Atomic Unit of Work) που είτε επιτυγχάνει ή αποτυγχάνει ως σύνολο. Είναι πολύ σημαντικό κάθε δοσοληψία να ολοκληρώνεται, μιας και έτσι μπορεί να απελευθερώσει τη χρήση σε κομμάτια της βάσης δεδομένων (Κόμβους, Σχέσεις) και της μνήμης που είχε κλειδώσει προσωρινά προκειμένου να χρησιμοποιήσει. Το Neo4j υποστηρίζει τρία είδη δοσοληψιών: ρητές (Explicit Transactions), έμμεσες (Implicit/ Auto-commit Transactions) και τις διαχειριζόμενες δοσοληψίες (Managed Transactions/ Transaction Functions). Η πρώτη περιλαμβάνει τις δοσοληψίες που διαχειρίζεται άμεσα ο χρήστης. Η εκκίνηση, επιτυχία ή αποτυχία μιας δοσοληψίας δρομολογείται με αποκλειστική ευθύνη του προγραμματιστή εντός της εφαρμογής πελάτη. Έχει το πλεονέκτημα ότι μπορεί να εκτελέσει σειριακά στο πλαίσιο μιας δοσοληψίας πολλαπλές ερωτήσεις Cypher. Η δεύτερη περιλαμβάνει τις δοσοληψίες που διαχειρίζεται αυτόματα το Neo4j. Έχει το μειονέκτημα ότι μπορεί να εκτελέσει μόνο μια ερώτηση Cypher στο πλαίσιο μιας δοσοληψίας. Το μεγάλο της πλεονέκτημα έγκειται στην απλότητα, μιας και αποδεσμεύει τον προγραμματιστή από το βάρος της διαχείρισης των δοσοληψιών. Η τρίτη περιλαμβάνει τις δοσοληψίες που δέχονται ως παράμετρο μια συνάρτηση που καθορίζει την εργασία που πρέπει να εκτελεστεί στα πλαίσια της δοσοληψίας. Το μεγάλο της πλεονέκτημα έγκειται στο γεγονός ότι ενσωματώνει μηχανισμό αυτόματης πολλαπλής επαναεκτέλεσης της εργασίας σε περίπτωση που η δοσοληψία που την περιλαμβάνει αποτύχει. Το πλήθος των επαναεκτελέσεων καθορίζεται για μια συγκεκριμένη χρονική περίοδο που ρυθμίζεται μέσω της παραμέτρου max_transaction_retry_time κατά τη δημιουργία του αντικειμένου neo4j.Driver. Στα πλαίσια της διπλωματικής και όσον αφορά την πρώτη προσέγγιση φόρτωσης της βάσης δεδομένων γράφου, επιλέχθηκε κάθε προσπέλαση στην βάση δεδομένων γράφου να γίνεται μέσω δοσοληψιών του τρίτου είδους.

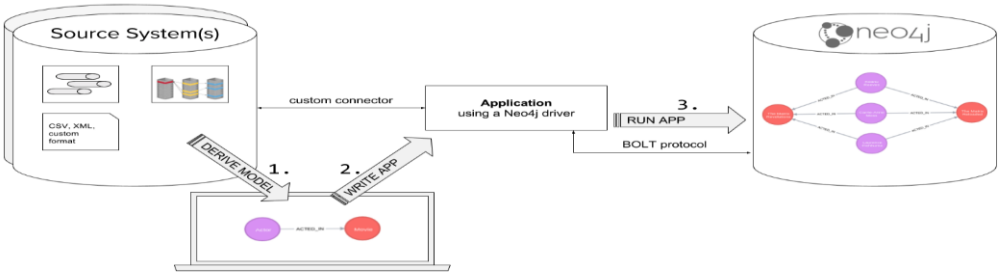
Σε όλη την έκταση της διπλωματικής καταβλήθηκε ιδιαίτερη προσπάθεια ώστε να μην υπάρξει παρέκκλιση από τα γενικά πρότυπα ανάθεσης αρμοδιοτήτων λογισμικού GRASP. Πρόκειται ουσιαστικά για ένα σύνολο εμπειρικών κανόνων που βοηθούν τον προγραμματιστή να αναθέτει αρμοδιότητες σε αντικείμενα με μεθοδικό και λογικό τρόπο. Σύμφωνα με τα πρότυπα GRASP, η δημιουργία ενός αντικειμένου θα πρέπει να ανατίθεται μόνο σε εκείνα τα σημεία μιας εφαρμογής που θα χρησιμοποιηθεί. Στα πλαίσια της εφαρμογής η αρμοδιότητα δημιουργίας ενός αντικειμένου EstablishANeo4jGraphDatabaseConnection.driver ανατίθεται στην συνάρτηση LoadDataToNeo4jGraphDatabase.query(). Αυτό σημαίνει ότι, κάθε φορά που εκτελείται η εν λόγω συνάρτηση, δημιουργείται και ένα καινούργιο αντικείμενο Driver. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι η δημιουργία πολλαπλών αντικειμένου Driver επιφέρει μια επιβάρυνση. Ωστόσο επιβλήθηκε λόγω της κατανεμημένης φύσης της βιβλιοθήκης DASK και της σχεδιαστικής απόφασής να εκτελείται κάθε κόμβος εργάτης στο νήμα εκτέλεσης μιας ξεχωριστής διεργασίας. Στη συνέχεια

αυτό το αντικείμενο δημιουργεί ένα αντικείμενο συνόδου που θα αποτελέσει το πλαίσιο εκτέλεσης μιας δοσοληψίας. Επειδή σε κάθε χρονική στιγμή σε μια σύνοδο μπορεί να εκτελείται μόνο μια δοσοληψία, επιλέχθηκε η δημιουργία πολλαπλών αντικειμένων συνόδου. Όταν μια σύνοδος ολοκληρώσει την εργασία της, τότε τερματίζεται, ώστε να μη σπαταλούνται πόροι. Ωστόσο, επειδή η δημιουργία ενός νέου αντικειμένου συνόδου και μιας νέας δοσοληψίας επιφέρει σημαντική επιβάρυνση, επιλέχθηκε η δοσοληψία που εκτελείται σε αυτή να αποθηκεύει μαζικά ένα μεγάλο κομμάτι ενός dataframe partition (DataFrame Partition Chunk). Στο σημείο αυτό κρίνεται σκόπιμο να αναφέρουμε ότι οποιεσδήποτε τροποποιήσεις διενεργούνται σε μια βάση δεδομένων στα πλαίσια μιας δοσοληψίας αποθηκεύονται στη μνήμη σωρού. Συνεπώς πρέπει να επιδειχθεί προσοχή, ώστε, όταν εκτελείται μια πολύπλοκη δοσοληψία, να μην προκαλείται κατάρρευση της συνόδου λόγω ανεπάρκειας μνήμης. Η γενική ροή εκτέλεσης της εφαρμογής πελάτη μέσω του οδηγού Python περιγράφεται στο Σχήμα 4.7.




Σχήμα 4.7: Γενική ροή εκτέλεσης εφαρμογής πελάτη για εισαγωγή δεδομένων μέσω του οδηγού Python

Η ροή εργασίας του οδηγού python περιγράφεται στο Σχήμα 4.8.




Σχήμα 4.8: Ροή εργασιών μαζικής εισαγωγής δεδομένων μέσω του οδηγού Python (Πηγή [44])

#### 4.5.1.2 Διαχείριση δοσοληψιών Neo4j

Το Neo4j χρησιμοποιεί το επίπεδο απομόνωσης read-committed isolation level. Σύμφωνα με αυτό, μια δοσοληψία μπορεί να βλέπει μόνο τα δεδομένα που έχουν εκτελεστεί επιτυχώς στα πλαίσια άλλων δοσοληψιών και όχι αυτά των δοσοληψιών που εκκρεμούν. Προκειμένου να υποστηρίξει το παραπάνω επίπεδο απομόνωσης, το σύστημα διαχείρισης του Neo4j χρησιμοποιεί έναν μηχανισμό κλειδωμάτων. Γενικά υποστηρίζει δυο τύπους κλειδώματος: shared/ Read lock και Exclusive/ Write Lock. Ο πρώτος τύπος χρησιμοποιείται μόνο για ανάγνωση δεδομένων και εξασφαλίζει ότι μια δοσοληψία δεν προσπελάζει έναν πόρο την ίδια στιγμή που αυτός τροποποιείται από μια άλλη δοσοληψία. Είναι δυνατόν να επιβληθούν πολλαπλά διαμοιραζόμενα κλειδώματα από διαφορετικά ταυτοχρονικά νήματα εκτέλεσης, υπό την προϋπόθεση ότι στον πόρο δεν έχει επιβληθεί κάποιο αποκλειστικό κλείδωμα. Ο δεύτερος τύπος χρησιμοποιείται για ανάγνωση και τροποποίηση δεδομένων και εξασφαλίζει ότι καμία άλλη δοσοληψία δεν μπορεί να προσπελάσει με οποιονδήποτε τρόπο (ανάγνωση, τροποποίηση) τον συγκεκριμένο πόρο που τον έχει αποκλειστικά κλειδώσει μια διαφορετική δοσοληψία. Ενδεικτικά αναφέρονται μερικές λειτουργίες εντολών Cypher που επιβάλουν κλειδώματα στους πόρους.

- Η προσθήκη, τροποποίηση ή διαγραφή μιας ιδιότητας ενός κόμβου ή συσχέτισης επιβάλλει ένα αποκλειστικό κλείδωμα στον κόμβο ή την συσχέτιση
- Η δημιουργία ή διαγραφή ενός κόμβου επιβάλλει ένα αποκλειστικό κλείδωμα στον κόμβο
- Η δημιουργία ή διαγραφή μιας συσχέτισης επιβάλλει ένα αποκλειστικό κλείδωμα στη συσχέτιση αλλά και στους δυο κόμβους που συμμετέχουν στη συσχέτιση

Ο παραπάνω μηχανισμός κλειδωμάτων μπορεί να προκαλέσει συνθήκες ανταγωνισμού για πόρους του βάσης γράφου σε περιπτώσεις όπου πολλαπλά ταυτοχρονικά νήματα εκτέλεσης εκτελούν τροποποιήσεις σε επικαλυπτόμενα κομμάτια του Γράφου. Σε τέτοιες περιπτώσεις, το σύστημα διαχείρισης του Neo4j εντοπίζει την πιθανότητα να συμβεί ένα αδιέξοδο, μαρκάρει την δοσοληψία για επαναεκτέλεση και απαντά με ένα μήνυμα σφάλματος. Στο σημείο αυτό αξίζει να σημειωθεί ότι η δοσοληψία που απέτυχε εξακολουθεί να διακρατεί τα κλειδώματα που είχε αποκτήσει σε πόρους του γράφου και δεν τα απελευθερώνει παρά μόνο μετά από την επιτυχή εκτέλεσή της. Για αυτό είναι σημαντικό να υλοποιούνται σε επίπεδο εφαρμογής πελάτη τεχνικές επαναεκτέλεσης των δοσοληψιών που απέτυχαν λόγο συνθηκών αδιεξόδου, ώστε να απελευθερώνουν τους κλειδωμένους πόρους προκειμένου να μπορούν να χρησιμοποιηθούν από άλλες δοσοληψίες.

Κάθε δοσοληψία πριν την εκτέλεσή της αποθηκεύεται στη μνήμη σωρού (Heap Memory) του υπολογιστικού μηχανήματος που λειτουργεί ως εξυπηρετητής της βάσης δεδομένων γράφου. Συνεπώς, είναι σημαντικό να μη δρομολογούνται προς εκτέλεση μεγάλες δοσοληψίες που δεν χωράνε στην διαθέσιμη μνήμη σωρού, ώστε να μη δημιουργηθούν σφάλματα ανεπάρκειας μνήμης (Out of Memory Errors). Προσοχή όμως πρέπει να επιδειχθεί, ώστε να μην δρομολογούνται προς εκτέλεση

υπερβολικά μικρές δοσοληψίες που υποβαθμίζουν την αποδοτικότητα της εφαρμογής. Σε περίπτωση που μια δοσοληψία ολοκληρωθεί επιτυχώς, αποθηκεύεται στον σκληρό δίσκο και καθίσταται ορατή στις υπόλοιπες δοσοληψίες. Αντίθετα, αν προκύψει κάποιο σφάλμα, τότε η δοσοληψία απορρίπτεται στο σύνολο της ώστε να μην παραβιαστούν οι αρχές ACID. Είναι εξαιρετικά σημαντικό κάθε δοσοληψία να ολοκληρώνεται επιτυχώς, ώστε να απελευθερώνει τα κλειδώματα που έχει επιβάλλει σε διάφορους πόρους της βάσης δεδομένων (Κόμβους, Συσχετίσεις, Ευρετήρια, Σχήματα). Ειδικότερα, σε ένα έντονα ταυτοχρονικό περιβάλλον εκτέλεσης δοσοληψιών, όπως αυτό που δημιουργήθηκε στα πλαίσια της διπλωματικής εργασίας, επιβάλλεται οι δοσοληψίες να είναι μικρού μεγέθους, ώστε να μην δημιουργούνται εκτεταμένα προβλήματα αδιεξόδων που υποβαθμίζουν την αποδοτικότητα ενημέρωσης της βάσης δεδομένων γράφου [45].

### 4.5.2 Μέσω εργαλείου neo4j-admin

#### 4.5.2.1 Δημιουργία ειδικής μορφής αρχείων csv

Αρχικά απαιτείται η δημιουργία των ειδικής μορφής αρχείων csv που θα περιέχουν τα δεδομένα που θα εισαχθούν στην βάση δεδομένων γράφου. Τα αρχεία αυτά θα πρέπει να αντανακλούν πιστά το μοντέλο δεδομένων ιδιοτήτων γράφου στο οποίο καταλήξαμε. Συγκεκριμένα, απαιτείται να δημιουργηθεί ένα ξεχωριστό αρχείο csv για κάθε διαφορετική οντότητα (Κόμβοι, Συσχετίσεις) του μοντέλου δεδομένων γράφου. Κάθε αρχείο csv απαρτίζεται από δυο τμήματα: την επικεφαλίδα και τον κορμό. Η επικεφαλίδα αποτελεί την πρώτη γραμμή του αρχείου και περιέχει πεδία-κλειδιά που περιγράφουν τις αντίστοιχες τιμές που περιλαμβάνονται στον κορμό του αρχείου. Ο κορμός αποτελεί το υπόλοιπο κομμάτι του αρχείου και περιλαμβάνει μια γραμμή για κάθε ξεχωριστή οντότητα. Μεταξύ της επικεφαλίδας και του κορμού ενός αρχείου csv δεν περιλαμβάνεται κάποια κενή γραμμή. Απαιτείται προσεκτική αντιστοίχιση μεταξύ των πεδίων της κεφαλίδας και των αντίστοιχων τιμών του κορμού, ώστε να μη δημιουργηθούν προβλήματα ασυμβατότητας τύπων κατά τη φόρτωση των δεδομένων. Επίσης, τόσο στα πεδία της επικεφαλίδας όσο και στις τιμές του κορμού ενός αρχείου csv απαιτείται να χρησιμοποιηθεί ομοιόμορφα ο ίδιος χαρακτήρας διαχωρισμού. Σε περίπτωση που τα αρχεία csv που θα δημιουργηθούν είναι πολύ μεγάλα σε μέγεθος υπάρχει δυνατότητα διαχωρισμού της επικεφαλίδας από τον κορμό και η αποθήκευσή του ως ξεχωριστό αρχείο. Η επιλογή αυτή μπορεί να γίνει κυρίως για λόγους ευελιξίας, ώστε κάθε φορά που απαιτούνται αλλαγές στην επικεφαλίδα να μη χρειάζεται να ανοίγονται από την αρχή τεράστια αρχεία και να εισάγονται αδικαιολόγητες καθυστερήσεις. Για τους ίδιους λόγους ευελιξίας, υπάρχει δυνατότητα διάσπασης και των περιεχομένων του κορμού ενός αρχείου csv σε πολλαπλά αρχεία csv. Σε μια τέτοια περίπτωση απαιτείται η χρήση κατάλληλων κανονικών εκφράσεων (Regular Expressions) που να περιγράφουν ομοιόμορφα και συνοπτικά τα ονόματα των πολλαπλών αρχείων csv κορμού που συνιστούν μια πηγή δεδομένων [46].

Κάθε αρχείο κεφαλίδας κόμβου περιλαμβάνει υποχρεωτικά ένα πεδίο :ID και ένα :LABEL. Το πρώτο πεδίο αντιπροσωπεύει οποιοδήποτε πεδίο έχουμε αποφασίσει να αποτελεί το μοναδικό αναγνωριστικό μιας συγκεκριμένης οντότητας κόμβων. Στην πλειοψηφία των οντοτήτων κόμβων ως μοναδικό αναγνωριστικό χρησιμοποιήθηκε το πεδίο id: Long. Μοναδική εξαίρεση αποτέλεσε η οντότητα κόμβου FieldStudy για την οποία χρησιμοποιήθηκε ως μοναδικό χαρακτηριστικό το πεδίο name: String. Αυτή η παρέκκλιση επιβλήθηκε από το γεγονός ότι στα αρχικά ανεπεξέργαστα δεδομένα των αρχείων json κατά γραμμές (Streaming Json Lines Files) δεν περιέχεται κάποια πληροφορία id για τις οντότητες FieldStudies. Επιχειρήθηκε επιτυχώς η ανάθεση σε κάθε οντότητα FieldStudy ενός τεχνητά δημιουργημένου id: Long μέσω μιας κλάσης Python που υλοποίησετην συγκεκριμένη λειτουργικότητα. Ωστόσο η προσέγγιση αυτή τελικά δεν υιοθετήθηκε λόγω της πάγιας πρακτικής του

Neo4j να μην αποθηκεύονται στη βάση δεδομένων γράφου τεχνητά δημιουργημένα πεδία που δεν διαθέτουν ουσιαστικό πληροφοριακό περιεχόμενο. Στο σημείο αυτό κρίνεται επιβεβλημένο να σημειωθεί ότι για κάθε οντότητα κόμβου υπάρχει ισχυρή διαβεβαίωση μοναδικότητας των πεδίων id's, τόσο σε επίπεδο μεμονωμένων αρχείων, όσο και συνολικά σε επίπεδο πολλαπλών αρχείων μιας πηγής δεδομένων. Η διαβεβαίωση ισχύει τελικά και για το πεδίο name: String κάθε οντότητας FieldStudy. Η τελευταία διαβεβαίωση κατέστη δυνατή εξαιτίας της ειδικής μέριμνας που ελήφθη στο αντίστοιχο κομμάτι της εφαρμογής Python που αφορά τη δημιουργία τωνοντοτήτων FieldStudy. Συγκεκριμένα στο μοναδικό αρχείο csv που δημιουργείται αποθηκεύονται αποκλειστικά οι μοναδικές τιμές του πεδίου name: String κάθε οντότητας FieldStudy. Για κάθε επικεφαλίδα οντότητας κόμβων δεν χρησιμοποιήθηκαν idspaces. Το δεύτερο πεδίο αντιπροσωπεύει την ετικέτα (ή τις ετικέτες) που θα αποδώσουμε σε κάθε τύπο κόμβο, ώστε να τον ομαδοποιεί κατάλληλα. Ως γνωστόν σε κάθε κόμβο μπορούν να αποδοθούν ταυτόχρονα πολλαπλές ετικέτες, που τον ομαδοποιούν κατάλληλα σύμφωνα με τις προδιαγραφές του συγκεκριμένου τομέα προβλήματος που μοντελοποιήθηκε. Συνεπώς, όλες αυτές οι ετικέτες διαχωρίζονται χρησιμοποιώντας τον κατάλληλο χαρακτήρα διαχωρισμού και αποτελούν μια τιμή τύπου λίστας στο πεδίου :LABEL. Στα παραπάνω δυο πεδία μπορούν να προστεθούν και οποιονδήποτε αριθμός πεδίων που αντιπροσωπεύουν τις ιδιότητες του μοντέλου που έχουμε αποφασίσει να αποθηκευτούν στη βάση δεδομένων γράφου. Κάθε τέτοια ιδιότητα είναι της μορφής name: field_type, όπου name είναι το όνομα της ιδιότητας και field_type ο τύπος δεδομένων. Το Neo4j υποστηρίζει μια μεγάλη ποικιλία τύπων δεδομένων ιδιοτήτων που καλύπτει τις ανάγκες κάθε τομέα προβλήματος. Στα πλαίσια της διπλωματικής εργασίας αποφασίστηκε να δημιουργηθούν, μέσω της εφαρμογής πελάτη Python, για τους κόμβους τα αρχεία επικεφαλίδων csv που απεικονίζονται στον παρακάτω πίνακα (Πίνακας 4.2).

Πίνακας 4.2: Δημιουργημένα αρχεία επικεφαλίδων csv για τις οντότητες κόμβων

| Α/ Α | Κόμβος | Όνομα Αρχείου Επικεφαλίδας | Περιεχόμενα Αρχείου Επικεφαλίδας |
|---|---|---|---|
| 1 | Venue | venues_header.csv | id:ID,name,type,:LABEL |
| 2 | Organization | organizations_header.csv | id:ID,name,:LABEL |
| 3 | Author | authors_header.csv | id:ID,name,:LABEL |
| 4 | Paper | papers_header.csv | id:ID,name,doi,type,:LABEL |
| 5 | FieldStudy | fieldstudies_header.csv | name:ID,:LABEL |

Στα πλαίσια της διπλωματικής εργασίας αποφασίστηκε επίσης να δημιουργηθούν,μέσω της εφαρμογής πελάτη Python, για τους κόμβους τα αρχεία κορμού csv που απεικονίζονται στον πίνακα που ακολουθεί (Πίνακας 4.3).

Πίνακας 4.3: Δημιουργημένα αρχεία κορμού csv για τις οντότητες κόμβων

| Α/ Α | Κόμβος | Ονόματα Αρχείων Κορμού | Ενδεικτικά Περιεχόμενα Αρχείων Κορμού (2 πρώτες γραμμές) |
|---|---|---|---|
| 1 | Venue | venues_body_0.csv | 465895,Eureka,Journal,Venue<br>1137746,The Artist and Journal of Home Culture,Journal,Venue |
| 2 | Organization | organizations_body_0.csv | 4605,Illinois College of Optometry,Organization<br>9507,Sangji University,Organization |
| 3 | Author | authors_body_000.csv - authors_body_405.csv | 584,Gözde ÖzdikmenliDemir,Author<br>859,Gy. Tolmár,Author |
| 4 | Paper | papers_body_000.csv - papers_body_488.csv | 15,Führung und Delegation,10.1007/978-3-642-32197-9_8,,Paper<br>125,Intellipse: A Knowledge Based Tool for an |

| | | | |
|---|---|---|---|
| | | | Integrated Project Support Environment,10.1007/978-1-4613-1033-4_15,,Paper |
| 5 | FieldStudy | fieldstudies_body_0.csv | political science,FieldStudy<br>delegation,FieldStudy |

Κάθε αρχείο κεφαλίδας συσχέτισης περιλαμβάνει υποχρεωτικά τα πεδία :START_ID, :END_ID και :TYPE. Το πρώτο αντιπροσωπεύει το :ID του κόμβου αφετηρίας της συσχέτισης. Το δεύτερο αντιπροσωπεύει το :ID του κόμβου προορισμού της συσχέτισης. Τέλος, το τρίτο αντιπροσωπεύει τον τύπο της συσχέτισης. Στο Neo4j τα δυο πρώτα πεδία είναι υποχρεωτικά, μιας και η απουσία ενός από αυτά καθιστά την συσχέτιση «σπασμένη» και συνεπώς δεν αποθηκεύεται στην βάση δεδομένων γράφου. Για κάθε «σπασμένη» συσχέτιση αποθηκεύεται και μια εγγραφή στο αρχείο /var/lib/neo4j/ import.report που δημιουργείται αυτόματα με την επιτυχή ολοκλήρωση της εντολής neo4j-admin import. Ως γνωστόν κάθε συσχέτιση μπορεί να έχει μόνο ένα τύπο. Ωστόσο επιτρέπεται πολλαπλοί διαφορετικοί τύποι συσχετίσεων να συνδέουν δυο κόμβους. Παρόμοια με τους κόμβους, επιτρέπεται η προσθήκη και επιπλέον πεδίων που αντιπροσωπεύουν τις ιδιότητες του μοντέλου δεδομένων τις οποίες επιθυμούμε να αποθηκεύσουμε στη βάση δεδομένων γράφου. Στα πλαίσια της διπλωματικής εργασίας αποφασίστηκε να δημιουργηθούν, μέσω της εφαρμογής πελάτη Python, για τις συσχετίσεις τα αρχεία επικεφαλίδων csv που απεικονίζονται στον πίνακα που ακολουθεί (Πίνακας 4.4).

Πίνακας 4.4: Δημιουργημένα αρχεία επικεφαλίδων csv για τις οντότητες συσχετίσεων

| Α/ Α | Συσχέτιση | Όνομα Αρχείου Επικεφαλίδας | Περιεχόμενα Αρχείου Επικεφαλίδας |
|---|---|---|---|
| 1 | CITES | cites_header.csv | :START_ID,:END_ID,:TYPE |
| 2 | IS_PUBLISHED_IN | is_published_in_header.csv | :START_ID,publisher,:END_ID,year:int,:TYPE |
| 3 | IS_RELATED_TO_FIELD_STUDY | is_related_to_field_study_header.csv | :START_ID,:END_ID,fieldOfStudyWeight:float,:TYPE |
| 4 | HAS_AUTHOR | has_author_header.csv | :START_ID,:END_ID,authorOrderInThePaper:int,:TYPE |
| 5 | IS_AFFILIATED_AT_ORGANIZATION | Is_affiliated_at_organization_header.csv | :START_ID,:END_ID,:TYPE |

Στα πλαίσια της διπλωματικής εργασίας αποφασίστηκε επίσης να δημιουργηθούν, μέσω της εφαρμογής πελάτη Python, για τις συσχετίσεις τα αρχεία κορμού csv που απεικονίζονται στον πίνακα που ακολουθεί (Πίνακας 4.5).

Πίνακας 4.5: Δημιουργημένα αρχεία κορμού csv για τις οντότητες συσχετίσεων

| Α/ Α | Συσχέτιση | Ονόματα Αρχείων Κορμού | Ενδεικτικά Περιεχόμενα Αρχείων Κορμού (2 πρώτες γραμμές) |
|---|---|---|---|
| 1 | CITES | cites_body_000.csv<br>-<br>cites_body_488.csv | 125,1564733196,CITES<br>125,1970185999,CITES |
| 2 | IS_PUBLISHED_IN | is_published_in_body_000.csv<br>-<br>is_published_in_body_488.csv | 285,North-Holland,173952182,2012,IS_PUBLISHED_IN<br>524,North-Holland,131663046,2002,IS_PUBLISHED_IN |

| | | | |
|---|---|---|---|
| 3 | IS_RELATED_TO_FIELD_STUDY | is_related_to_field_study_body_000.csv<br>-<br>is_related_to_field_study_body_488.csv | 15,political science,0.3147,IS_RELATED_TO_FIELD_STUDY<br>15,delegation,0.38966,IS_RELATED_TO_FIELD_STUDY |
| 4 | HAS_AUTHOR | has_author_body_000.csv<br>-<br>has_author_body_405.csv | 2104020361,584,0,HAS_AUTHOR<br>2087060482,584,0,HAS_AUTHOR |
| 5 | IS_AFFILIATED_AT_ORGANIZATION | is_affiliated_at_organization_body_000.csv<br>-<br>is_affiliated_at_organization_body_488.csv | 2002579779,169199633,IS_AFFILIATED_AT_ORGANIZATION<br>1995014452,169199633,IS_AFFILIATED_AT_ORGANIZATION |

Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι, όταν τα δεδομένα ανακτώνται με αυτοματοποιημένες διαδικασίες από διαφορετικές πηγές, χαρακτηρίζονται από χαμηλή ποιότητα. Το γεγονός αυτό επισημάνθηκε εκτενώς κατά το πρώτο στάδιο της διερευνητικής ανάλυσης των συνόλων δεδομένων Mag και Aminer. Ενδεικτικά αναφέρεται η παρουσία πολλαπλών χαρακτήρων προώθησης του κέρσορα στην αρχή της γραμμής“\r” (Carriage Return), χαρακτήρων οριζόντιου στηλογνώμονα “\t” (Horizontal Tab) καθώς και χαρακτήρων προώθησης μιας γραμμής μπροστά “\n” (Line Feed) στον κορμό των πεδίων Papers.title και Papers.publisher. Αυτές οι αλλαγές γραμμής είχαν ως αποτέλεσμα η αλφαριθμητική τιμή του πεδίου να εκτείνεται σε πολλαπλές γραμμές στο αρχείο csv. Ωστόσο το εργαλείο neo4j-admin import είναι εξ’ ορισμού ρυθμισμένο, ώστε να θεωρεί ότι η τιμή κάθε πεδίου μιας γραμμής ενός αρχείου csv που δέχεται ως είσοδο ότι δεν εκτείνεται σε πολλαπλές γραμμές. Αυτό είχε ως αποτέλεσμα την αποτυχία εισαγωγής των δεδομένων στη βάση δεδομένων και τον τερματισμό με κάποιο κωδικό σφάλματος. Το εργαλείο neo4j-admin import επιτρέπει στην τιμή ενός πεδίου να εκτείνεται σε πολλαπλές γραμμές μέσω κατάλληλης ρύθμισής του. Παρόλα αυτά δεν ακολουθήθηκε αυτή η προσέγγιση, γιατί υποβαθμίζει σοβαρά την απόδοση. Αντίθετα, επιλέχθηκε το πρόβλημα να αντιμετωπιστεί απομακρύνοντας όλους τους παραπάνω χαρακτήρες κατά την φάση καθαρισμού των δεδομένων. Επίσης μέσα από την εφαρμογή πολύπλοκων ροών επεξεργασίας μπορεί να εισαχθούν απρόβλεπτα σφάλματα. Ενδεικτικά αναφέρεται η αυτόματη μετατροπή όλων των τιμών τύπου integer σε float σε εκείνες τις στήλες ενός dataframe που περιέχουν κάποιες τιμές null. Σε αυτές τις περιπτώσεις απαιτούνται μακροσκελείς διαδικασίες καθαρισμού και ομογενοποίησης τους, ώστε να αντιμετωπιστούν τα προβλήματα ποιότητας που ανακύπτουν. Στην τρέχουσα προσέγγιση μαζικής εισαγωγής δεδομένων που επιλέξαμε να ακολουθήσουμε, τα αρχεία csv θα αποτελέσουν την είσοδο του εργαλείου neo4j-admin. Αυτό το εργαλείο δεν παρέχει καμία δυνατότητα καθαρίσματος, μετατροπής, μορφοποίησης των δεδομένων κατά την διαδικασία φόρτωσής τους στη βάση δεδομένων γράφου. Αντίθετα, πρέπει τα αρχεία csv να είναι έτοιμα και στην κατάλληλη μορφή, όταν θα αποτελέσουν την είσοδο στο εργαλείο neo4j-admin. Σε αντίθετη περίπτωση, η διαδικασία εισαγωγής θα αποτύχει μιας και το συγκεκριμένο εργαλείο είναι εξαιρετικά «ευαίσθητο» όσον αφορά την ποιότητα και τη μορφή των δεδομένων που αναμένει ως είσοδο. Συνεπώς η όλη διαδικασία καθαρίσματος και μετατροπής των δεδομένων σε κατάλληλη μορφή αποτελεί αποκλειστική ευθύνη της εφαρμογής πελάτη Python. Αφού παραχθούν τα αρχεία csv και πριν αποτελέσουν την είσοδο στο εργαλείο neo4j-admin, ελέγχεται εκτενώς η εγκυρότητά τους μέσω της διαδικτυακής υπηρεσίας http://csvlint.io/.

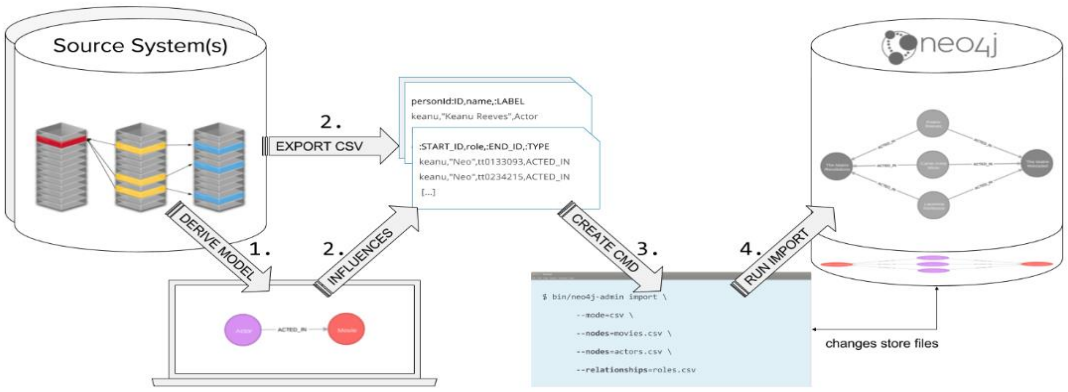
#### 4.5.2.2 Χρήση του εργαλείου neo4j-admin import

Το neo4j-admin αποτελεί ένα εργαλείο διαχείρισης που είναι προσπελάσιμο από τον φάκελο /usr/bin της τρέχουσας εγκατάστασης του Neo4j. Στον φάκελο αυτό περιλαμβάνονται και άλλα εργαλεία που

είναι χρήσιμα σε όσους αναπτύσσουν και διαχειρίζονται εφαρμογές σε Neo4j, όπως cypher-shell κ.α. Το εργαλείο υποστηρίζει πολλαπλές εντολές με διάφορες λειτουργικότητες. Σε αυτές συγκαταλέγονται εντολές δημιουργίας αντιγράφων ασφαλείας (dump, backup), επαναφοράς μιας κατεστραμμένης βάσης από κάποιο αντίγραφο (load, restore), μετακίνησης ενός τοπικού αντιγράφου μιας βάσης σε ένα στιγμιότυπο Neo4jπου τρέχει σε μια υποδομή υπολογιστικού νέφους (push-to-cloud), συστάσεις παραμετροποίησης της διαθέσιμης μνήμης του μηχανήματος, όπου είναι εγκατεστημένο το στιγμιότυπο του Neo4j (memrec) κ.α. Κάποιες από αυτές τις λειτουργικότητες είναι διαθέσιμες και στις δυο βασικές εκδόσεις του Neo4j, ενώ άλλες διατίθενται αποκλειστικά στην εμπορική έκδοση. Ωστόσο στην τρέχουσα ενότητα θα επικεντρωθούμε αποκλειστικά στην εντολή import. Η εντολή αυτή επιτρέπει την μαζική φόρτωση δεδομένων σε μια βάση δεδομένων γράφου Neo4j. Το συγκεκριμένο εργαλείο χρησιμοποιείται σε περιπτώσεις τοπικών εγκαταστάσεων (Neo4j Desktop, Neo4j EE Docker image, Neo4j Server). Για οποιοδήποτε άλλο τύπο εγκατάστασης πολύ μεγάλων όγκων δεδομένων συνιστάται η χρήση του εργαλείου Kettle import. Στα πλαίσια της διπλωματικής επιλέξαμε να χρησιμοποιήσουμε ως τον πρώτο τεχνολογικό πυλώνα ένα στιγμιότυπο του Neo4j CE. Ως γνωστόν αυτή η έκδοση του Neo4j δεν υποστηρίζει τη διαχείριση πολλαπλών βάσεων δεδομένων. Ο χρήστης περιορίζεται στη χρήση αποκλειστικά της εξ' ορισμού βάσης δεδομένων neo4j. Η συγκεκριμένη επιλογή θέτει κάποιους περιορισμούς που διαφοροποιούν τον τρόπο εργασίας. Καταρχήν απαιτείται ο εξυπηρετητής που τρέχει το στιγμιότυπο Neo4j ως υπηρεσία να είναι κλειστός. Προκειμένου να το επιτύχουμε αυτό πρέπει στην γραμμή εντολών να εκτελέσουμε την παρακάτω εντολή.

```
sudo systemctl stop neo4j.service
```

Στο σημείο αυτό επιβάλλεται να τονίσουμε ότι το συγκεκριμένο εργαλείο χρησιμοποιείται για την καταρχήν προετοιμασία των δεδομένων που πρόκειται να εισαχθούν σε μια βάση δεδομένων που δεν υπάρχει ήδη ή αν υπάρχει να είναι εντελώς άδεια. Αν επιχειρηθεί εισαγωγή δεδομένων σε μια βάση που περιέχει δεδομένα η όλη διαδικασία θα αποτύχει με κωδικό σφάλματος. Για αυτό μπορεί να χρησιμοποιηθεί μόνο μια φορά για την καταρχήν φόρτωση μιας νέας βάσης δεδομένων. Ο τρόπος λειτουργίας του εργαλείου neo4j-admin import περιγράφεται στο Σχήμα 4.9.




Σχήμα 4.9: Ροή εργασιών μαζικής εισαγωγής δεδομένων μέσω του εργαλείου neo4j-admin import (Πηγή [44])

Η συνήθης πρακτική υπαγορεύει να μην πειράζουμε την εξ΄ ορισμού βάση δεδομένων neo4j, αλλά η εισαγωγή να γίνεται σε μια εντελώς καινούργια ανύπαρκτη βάση δεδομένων. Στα πλαίσια της διπλωματικής επιλέξαμε η βάση αυτή να λέγεται neo4j.db. Στην συνέχεια μεταβαίνουμε στο αρχείο

ρυθμίσεων /etc/neo4j/neo4j.conf και καθορίζουμε το όνομα της εξ' ορισμού βάσης δεδομένων που θα χρησιμοποιήσουμε θέτοντας την παρακάτω ρύθμιση.

```
dbms.default_database=neo4j.db
```

Επίσης στο ίδιο αρχείο ρυθμίσεων /etc/neo4j/neo4j.conf αποδίδουμε στην μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη τις τιμές που απεικονίζονται παρακάτω.

```
dbms.memory.heap.initial_size=10g
dbms.memory.heap.max_size= 10g
dbms.memory.pagecache.size=40g
```

Τέλος μεταβαίνουμε στην γραμμή εντολών του καταλόγου εγκατάστασης του στιγμιότυπου Neo4j και εκτελούμε την παρακάτω εντολή προκειμένου να φορτώσουμε τα δεδομένα στην βάση δεδομένων neo4j.db. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι η εντολή πρέπει να εκτελεστεί στη γραμμή εντολών ως μια συνεχόμενη γραμμή. Στη παρακάτω απεικόνιση της σύνταξης της εντολής έχουν εισαχθεί αρκετές αλλαγές γραμμής, ώστε να βελτιωθεί η αναγνωσιμότητά της.

```
/usr/bin/neo4j-admin import
--database=neo4j.db
--force=true
--skip-duplicate-nodes
--skip-bad-relationships
--ignore-empty-strings
--
nodes=/home/kostasvav/data/produced_csv_files/mag/nodes/venues/header_files/venues_hea
der.csv,'/home/kostasvav/data/produced_csv_files/mag/nodes/venues/body_files/venues_bo
dy_\d+\.csv'
--
nodes=/home/kostasvav/data/produced_csv_files/mag/nodes/organizations/header_files/org
anizations_header.csv,'/home/kostasvav/data/produced_csv_files/mag/nodes/organizations
/body_files/organizations_body_\d+\.csv'
--
nodes=/home/kostasvav/data/produced_csv_files/mag/nodes/authors/header_files/authors_h
eader.csv,'/home/kostasvav/data/produced_csv_files/mag/nodes/authors/body_files/author
s_body_\d+\.csv'
--
nodes=/home/kostasvav/data/produced_csv_files/mag/nodes/papers/header_files/papers_hea
der.csv,'/home/kostasvav/data/produced_csv_files/mag/nodes/papers/body_files/papers_bo
dy_\d+\.csv'
--
nodes=/home/kostasvav/data/produced_csv_files/mag/nodes/fieldstudies/header_files/fiel
dstudies_header.csv,'/home/kostasvav/data/produced_csv_files/mag/nodes/fieldstudies/bo
dy_files/fieldstudies_body_\d+\.csv'
--
relationships=/home/kostasvav/data/produced_csv_files/mag/relationships/cites/header_f
iles/cites_header.csv,'/home/kostasvav/data/produced_csv_files/mag/relationships/cites
/body_files/cites_body_\d+\.csv'
--
relationships=/home/kostasvav/data/produced_csv_files/mag/relationships/has_author/hea
der_files/has_author_header.csv,'/home/kostasvav/data/produced_csv_files/mag/relations
hips/has_author/body_files/has_author_body_\d+\.csv'
```

Στο σημείο αυτό είναι σημαντικό να διευκρινιστούν ορισμένες παραμέτρους που χρησιμοποιήθηκαν κατά την εκτέλεση της παραπάνω εντολής. Καταρχήν χρησιμοποιούνται πολλαπλές πηγές δεδομένων για κάθε διαφορετική οντότητα (Κόμβος, Συσχέτιση). Η επιλογή αυτή έγινε για λόγους ευελιξίας και αύξησης της αποδοτικότητας εκτέλεσης της εντολής. Ουσιαστικά κάθε πηγή δεδομένων

αντιπροσωπεύει ένα διαφορετικό αρχείο (ή αρχεία) άντλησης δεδομένων τα οποία αφορούν αποκλειστικά μια συγκεκριμένη οντότητα. Η κάθε πηγή δεδομένων σηματοδοτείται με την παράμετρο –nodes ή την παράμετρο –relationships, ανάλογα με το αν η πηγή δεδομένων αναφέρεται σε κόμβο ή συσχέτιση. Η κάθε επικεφαλίδα αποφασίστηκε για λόγους ευελιξίας να αποθηκευτεί σε ξεχωριστό αρχείο. Επίσης αποφασίστηκε η διάσπαση του κορμού εξαιρετικά μεγάλων αρχείων csv σε πολλαπλά αρχεία csv, τα ονόματα των οποίων περιγράφονται συνοπτικά με χρήση κανονικών εκφράσεων. Μοναδική εξαίρεση αποτέλεσαν τα αρχεία κορμού csv της οντότητας FieldStudy. Αυτά λόγω του μικρού τους μεγέθους (μερικές εκατοντάδες χιλιάδες γραμμών) αποφασίστηκε να αποθηκευτούν σε ένα μοναδικό αρχείο csv. Σε κάθε πηγή δεδομένων πάντοτε το αρχείο επικεφαλίδας προηγείται των αρχείων κορμού, χωρίς να παρεμβάλλεται αναμεσά τους κάποιος κενός χαρακτήρας. Για τα αρχεία κορμού δεν χρησιμοποιήθηκε συμπίεση. Προκειμένου να αντιμετωπιστούν προβλήματα με διπλότυπους κόμβους ή με «σπασμένες» συσχετίσεις επιλέχθηκε κατά την εισαγωγή οι γραμμές του αντίστοιχου αρχείου csv που αναπαριστούν προβληματικούς κόμβους ή συσχετίσεις να παραλείπονται. Ωστόσο μέσω της εφαρμογής πελάτη Python εξασφαλίστηκε τη μη ύπαρξη διπλότυπων τιμών στα πεδία των οντοτήτων που διαδραματίζουν τον ρόλο του αναγνωριστικού. Αυτό έγινε ώστε να επιταχυνθεί σημαντικά η διαδικασία εισαγωγής των δεδομένων στην βάση δεδομένων γράφου. Επίσης επιλέχθηκε να μην αποθηκεύονται στην βάση δεδομένων γράφου ιδιότητες κόμβων ή συσχετίσεων με τιμή το κενό αλφαριθμητικό [47]. Όσον αφορά την παράμετρο --multiline-fields επιλέχθηκε να μην χρησιμοποιηθεί, ώστε να ισχύει η εξ’ ορισμού τιμή της (False). Αυτή η επιλογή έγινε ώστε να μην επιτρέπεται η τιμή ενός πεδίου των αρχείων csv να εκτείνεται σε πολλαπλές γραμμές, γεγονός που θα υποβάθμιζε σημαντικά την απόδοση της διαδικασίας εισαγωγής. Χρησιμοποιήθηκε η παράμετρος --force=true ώστε σε περίπτωση που η βάση δεδομένωνυπάρχει ήδη και περιέχει δεδομένα αυτή να διαγραφεί. Η χρήση αυτής της παραμέτρου ήταν απολύτως απαραίτητη, γιατί προηγήθηκαν αρκετές ανεπιτυχείςεκτελέσεις της εντολής neo4j-admin import, οι οποίες εισήγαγαν ένα μέρος των δεδομένων στη βάση δεδομένων γράφου. Ωστόσο, όπως αναφέραμε και σε προηγούμενη ενότητα, το συγκεκριμένο εργαλείο απαιτεί η εισαγωγή να γίνει σε μια βάση δεδομένων που δεν υπάρχει ήδη, ή σε διαφορετική περίπτωση η βάση δεδομένων να είναι εντελώς κενή. Η εισαγωγή περιλαμβάνει τέσσερα στάδια, είναι ταχύτατη και τα στατιστικά της στοιχεία απεικονίζονται στο πίνακα που ακολουθεί (Πίνακας 4.6). Αυτό εν μέρει δικαιολογείται από το γεγονός ότι το εργαλείο neo4j-admin επενεργεί σε μια «σταματημένη» βάση δεδομένων και δεν βασίζεται σε δοσοληψίες.

Πίνακας 4.6: Εκτιμώμενα στατιστικά στοιχεία εισαγωγής δεδομένων με το εργαλείο neo4j-admin import ανά βήμα και συνολικά

| Α/ Α | Βήμα | Τύπος Οντότητας | Εκτιμώμενος αριθμός οντοτήτων | Εκτιμώμενη χρήση χώρου στο δίσκο | Εκτιμώμενη απαιτούμενη χρήση μνήμης | Χρόνος Εκτέλεσης |
|---|---|---|---|---|---|---|
| 1 | Εισαγωγή Κόμβων(1/4) | Κόμβος | 345.66 M | 42.38 GiB* | 5.245 GiB* | 00h:14m:27s:297ms |
| 2 | Εισαγωγή Σχέσεων(2/4) | Σχέση | 2.86 G | 145.4 GiB* | 4.729 GiB* | 02h:06m:50s:107ms |
| 3 | Σύνδεση Σχέσεων(3/4) | -- | -- | -- | 4.417 GiB* | 2h:48m:16s:61ms |
| 4 | Μετα-επεξεργασία (4/4) | -- | -- | -- | 1020 MiB** | 00h:31m:57s:606ms |
| Σύνολο | | | 345.66 ΜΚόμβοι, 854.19 ΜΙδιότητες Κόμβων, 2.86 GΣχέσεις, 1.46 GΙδιότητες Σχέσεων | 187.8GiB* | 5.245 GiB* | 05h:50m:43s:781ms |

* 1 GiB = $1024^3$ bytes = 1,073,741,824 bytes (1 GB ≈ 0.93 GiB) , όπου GiB = gibibyte
** 1 MiB = $1024^2$ bytes= 1,048,576bytes (1 MB ≈ 0.95 MiB) , όπου MiB = mebibyte

Τέλος καθορίζουμε τα δικαιώματα του χρήστη neo4j για την νέα βάση δεδομένων neo4j.db, εκτελώντας στην γραμμή εντολής τις εντολές που απεικονίζονται παρακάτω.

```
sudo chown -R neo4j:neo4j /var/lib/neo4j/data/transactions/neo4j.db
sudo chown -R neo4j:neo4j /var/lib/neo4j/data/databases/neo4j.db
```

Πλέον η βάση δεδομένων γράφου είναι έτοιμη προς εξερεύνηση μέσω της διαδικτυακής εφαρμογής Neo4j browser. Συνδεόμαστε στον Neo4jbrowser με τα διαπιστευτήρια του χρήστη neo4j. Στη συνέχεια, επιλέγουμε την βάση δεδομένων neo4j.db ως την τρέχουσα ενεργή βάση εργασίας, εκτελώντας στην γραμμή εντολών του Neo4j browser την παρακάτω εντολή.

```
:use neo4j.db
```

Προκειμένου να εμφανιστούν όλες οι διαθέσιμες βάσεις δεδομένων που διαχειρίζεται το σύστημα διαχείρισης βάσεων δεδομένων του Neo4j που τρέχει στον εξυπηρετητή, εκτελείται η παρακάτω εντολή.

```
show databases
```

Προκειμένου να εμφανίσουμε όλους τους περιορισμούς της τρέχουσας ενεργής βάσης δεδομένων neo4j.db που τρέχει στον εξυπηρετητή εκτελείται η παρακάτω εντολή.

```
show constraints
```

Στο σημείο αυτό αξίζει να αναφερθεί ότι, όταν ακολουθείται η προσέγγιση της φόρτωσης μιας βάσης δεδομένων μέσω του εργαλείου neo4j-admin import, η βάση δεδομένων θα πρέπει να είναι εντελώς άδεια και επίσης να μην έχει καθόλου περιορισμούς ή ευρετήρια. Οι περιορισμοί και τα τυχόν ευρετήρια θα δημιουργηθούν υποχρεωτικά εκ των υστέρων και αφούτα δεδομένα φορτωθούν πλήρως στην βάση δεδομένων γράφου. Αντίθετα, στην προσέγγιση φόρτωσης μιας βάσης δεδομένων μέσω δοσοληψιών και του οδηγού μιας γλώσσας προγραμματισμού ο χρήστης έχει την ευελιξία να δημιουργήσει τους περιορισμούς ή τα ευρετήρια πριν ή μετα την διαδικασία εισαγωγής των δεδομένων. Συνεπώς, προκειμένου να δημιουργηθούν όλοι οι απαραίτητοι περιορισμοί, εκτελούνται διαδοχικά στην γραμμή εντολών του Neo4j browser οι περιορισμοί που απεικονίζονται στον παρακάτω πίνακα (Πίνακας 4.7).

Πίνακας 4.7: Περιορισμοί που πρέπει να εισαχθούν στην βάση δεδομένων γράφου μετά την φόρτωση της από το εργαλείο neo4j-admin import

| Α/ Α | Όνομα Περιορισμού | Τύπος Περιορισμού | Τύπος Οντότητας | Χρόνος Εκτέλεσης | Εντολή Cypher |
|---|---|---|---|---|---|
| 1 | venues_uniqueness_restriction | UNIQUENESS | Κόμβος | 28s | CREATE CONSTRAINT venues_uniqueness_restriction IF NOT EXISTS ON (v:Venue) ASSERT v.id IS UNIQUE |
| 2 | organizations_uniqueness_restriction | UNIQUENESS | Κόμβος | 1s | CREATE CONSTRAINT organizations_uniqueness_restriction IF NOT EXISTS ON (o:Organization) ASSERT o.id IS UNIQUE |
| 3 | field_studies_uniqueness_restriction | UNIQUENESS | Κόμβος | 5s | CREATE CONSTRAINT field_studies_uniqueness_restriction IF NOT EXISTS ON (fs:FieldStudy) ASSERT fs.name IS UNIQUE |
| 4 | papers_uniqueness_restriction | UNIQUENESS | Κόμβος | 22m:44s | CREATE CONSTRAINT papers_uniqueness_restriction IF NOT EXISTS ON (p:Paper) ASSERT p.id IS UNIQUE |
| 5 | authors_uniqueness_restriction | UNIQUENESS | Κόμβος | 13m:8s | CREATE CONSTRAINT authors_uniqueness_restriction IF NOT EXISTS ON (a:Author) ASSERT a.id IS UNIQUE |

## 4.6 Επίλογος

Στο κεφάλαιο αυτό αναφέρθηκαν οι σχεδιαστικές αποφάσεις που πάρθηκαν αναφορικά με την βιβλιογραφική υποδομή. Όσον αφορά τον σχεδιασμό του μέσου αποθήκευσης των βιβλιογραφικών δεδομένων, κατασκευάστηκε μια ιεραρχικά εμφωλευμένη δομή από καταλόγους που αντανακλά πλήρως τον φυσικό τρόπο οργάνωσής τους. Δύο λύσεις κρίθηκαν ως οι πλέον κατάλληλες να φιλοξενήσουν την παραπάνω οργανωτική δομή: το Google Cloud Storage και το Google Drive. Και οι δυο λύσεις αποτελούν προϊόντα της Google με διαφορετικό όμως προσανατολισμό. Τελικά, το Google Drive αναγνωρίστηκε ως η πιο οικονομικά συμφέρουσα επιλογή. Επίσης, οπουδήποτε κρίθηκε αναγκαίο, περιγράφηκαν με περισσότερη λεπτομέρεια επιπλέον πληροφορίες υλοποίησης. Στη συνέχεια αποκρυσταλλώθηκε η ETL τύπου επεξεργασία που έπρεπε να εφαρμοστεί. Αυτή αποφασίστηκε να περιλαμβάνει δυο βασικά στάδια: Το πρώτο αποσκοπεί στην προεπεξεργασία των συνόλων δεδομένων, ώστε να περιέλθουν σε μια πιο επιθυμητή μορφή. Τα δεδομένα αυτά στη συνέχεια αποτελούν την είσοδο του δεύτερου σταδίου που αποσκοπεί στη φόρτωσή τους στη βάση δεδομένων γράφου. Η πολυπλοκότητα των επεξεργασιών επέβαλε την περαιτέρω ανάλυση κάθε σταδίου σε υπό στάδια. Το πρώτο στάδιο αναλύθηκε επιπλέον, ώστε να περιλαμβάνει διαδοχικά υπό στάδια φόρτωσης, φιλτραρίσματος, εξαγωγής - ομογενοποίησης και τέλος καθαρίσματος των δεδομένων. Το δεύτερο στάδιο αναλύθηκε επιπλέον ώστε να περιλαμβάνει δυο επιπλέον υπό στάδια. Το πρώτο υπό στάδιο αποσκοπεί στη φόρτωση των κόμβων στη βάση δεομένων γράφου. Το δεύτερο υπό στάδιο αποσκοπεί στη φόρτωση των συσχετίσεων στην βάση δεδομένων γράφου. Η τελευταία ανάλυση είναι σύμφωνη με τις καθιερωμένες βέλτιστες πρακτικές εισαγωγής δεδομένων σε μια βάση δεδομένων γράφου. Στο επόμενο κεφάλαιο περιγράφεται η αρχιτεκτονική δομή της υπό δημιουργίας βιβλιογραφικής υποδομής. Συγκεκριμένα, προσδιορίζονται τα επιμέρους τμήματα που τη συνθέτουν καθώς και ο τρόπος σύνδεσης τους. Με αυτόν τον τρόπο αποκτούμε μια πιο σχηματοποιημένη εικόνα της υπό δημιουργίας βιβλιογραφικής υποδομής.

# Κεφάλαιο 5ο: Αρχιτεκτονικά Τμήματα Υποδομής Βιβλιογραφικών Δεδομένων

## 5.1 Εισαγωγή

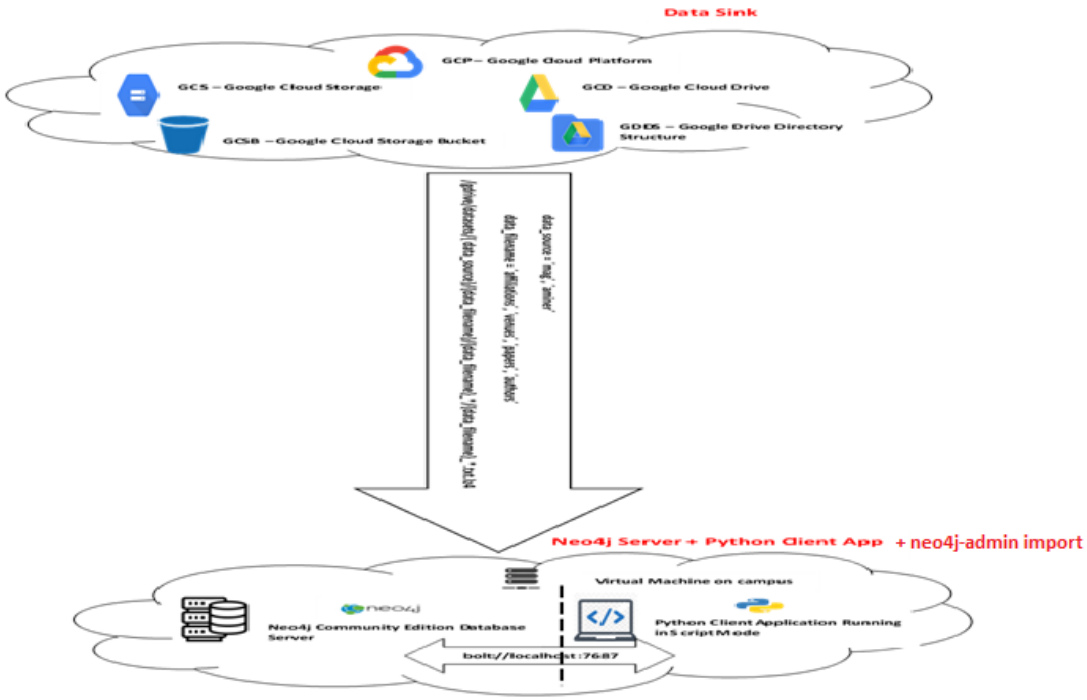
Μετα την φάση της σχεδίασης ακολουθεί η φάση προσδιορισμού της αρχιτεκτονικής δομής της βιβλιογραφικής υποδομής. Σε αυτήν προσδιορίζονται τα επιμέρους τμήματα που τη συνθέτουν καθώς και πως αυτά συνδέονται μεταξύ τους. Ουσιαστικά μια αρχιτεκτονική δομή αποτελεί έναν σύνολο από βέλτιστες τεχνικές που επιβάλλεται να ακολουθήσουμε, ώστε να δημιουργηθεί μια σωστά δομημένη και λειτουργική εφαρμογή. Γενικά οι τεχνικές αφορούν και λειτουργικότητες προσκηνίου (Front-end) και παρασκηνίου (Back-end). Στα πλαίσια της παρούσας διπλωματικής, οι εφαρμογές θα εκτελεστούν αποκλειστικά στην γραμμή εντολών σε Script Mode. Συνεπώς όλες οι λειτουργικότητες θα αφορούν στο παρασκήνιο. Κύριο μέλημα κατά την επιλογή της συγκεκριμένης αρχιτεκτονικής δομής είναι τα διάφορα τμήματά της να επικοινωνούν απρόσκοπτα χωρίς χωρικά και χρονικά εμπόδια. Εξίσου σημαντική προδιαγραφή αποτέλεσε και το ότι τα διάφορα τμήματα έπρεπε να παρουσιάζουν ελάχιστες αλληλεξαρτήσεις και να μην υπάρχει αλληλοεπικάλυψη αρμοδιοτήτων. Ειδικά η τελευταία απαίτηση κρίθηκε ως ιδιαίτερα σημαντική, ώστε τα διάφορα τμήματα να αναπτυχθούν με ανεξάρτητο τρόπο. Κάτι τέτοιο θα επέτρεπε μελλοντικά την εξ' ολοκλήρου αντικατάσταση ενός τμήματος της υποδομής με ένα άλλο πιο σύγχρονης τεχνολογίας με ελάχιστη προγραμματιστική προσπάθεια. Τέλος, μια καλά δομημένη αρχιτεκτονική διευκολύνει την οπτικοποίηση και την εις βάθος κατανόηση της βιβλιογραφικής υποδομής που δημιουργήθηκε.

## 5.2 Γενικά

Η υποδομή που δημιουργήθηκε περιλαμβάνει τμήματα λογισμικού και υλικού. Τα τμήματα λογισμικού περιλαμβάνουν μια εφαρμογή πελάτης Python που είναι υπεύθυνη για τη μαζική δημιουργία των απαραίτητων αρχείων csv σύμφωνα με το μοντέλο ιδιοτήτων γράφου που αποκρυσταλλώθηκε. Στη συνέχεια αυτά τα αρχεία csv θα αποτελέσουν την είσοδο στην εντολή neo4j-admin import που θα εισάγει τελικά τα δεδομένα στη βάση δεδομένων γράφου. Η εισαγωγή των δεδομένων έγινε σύμφωνα με τις βέλτιστες πρακτικές του Neo4j. Σύμφωνα με αυτές η εισαγωγή των κόμβων (και των σχετικών με αυτούς ιδιοτήτων) πρέπει να πραγματοποιείται ξεχωριστά και να προηγείται της εισαγωγής των σχέσεων (και των σχετικών με αυτές ιδιοτήτων). Επίσης κατά τη διάρκεια της εισαγωγής αποκρυσταλλώθηκε μια εκτίμηση του μεγέθους, των αποθηκευτικών απαιτήσεων και της αποδοτικότητας εισαγωγής του κάθε κομματιού της βάσης δεδομένων γράφου. Αυτό επιτεύχθη μέσω των εκτιμήσεων που απεικόνιζε στην οθόνη του υπολογιστή κατά τη διάρκεια της εκτέλεσής της εντολής neo4j-admin import.

Όσον αφορά τα τμήματα υλικού, περιλαμβάνουν επίσης δυο κομμάτια καθένα από τα οποία εξυπηρετεί διαφορετικές ανάγκες λειτουργικότητας. Το πρώτο κομμάτι εξυπηρετεί ως απομακρυσμένη δεξαμενή των ανεπεξέργαστων, συμπιεσμένων βιβλιογραφικών δεδομένων. Συγκεκριμένα έχουν δημιουργηθεί δυο εκδόσεις του απομακρυσμένου μέσου αποθήκευσης. Η πρώτη έχει τη μορφή ενός κουβά GCS –στην υποδομή υπολογιστικού νέφους της Google. Η δεύτερη έχει τη μορφή ενός λογαριασμού Google Drive στον οποίο έχει αγορασθεί επιπλέον αποθηκευτικός χώρος 2 TB, ώστε να μπορέσει να ανταποκριθεί στις υψηλές αποθηκευτικές απαιτήσεις της εφαρμογής. Και στα δυο αυτά μέσα αποθήκευσης έχει δημιουργηθεί μια ιεραρχικά εμφωλευμένη δομή καταλόγων, όπου και αποθηκεύτηκαν τα αρχικά δεδομένα. Τελικά επιλέχθηκε στα πλαίσια της διπλωματικής να χρησιμοποιηθεί η Google Drive έκδοση του μέσου αποθήκευσης. Η επιλογή αυτή βασίστηκε κυρίως

σε λόγους κόστους. Στην πορεία της διπλωματικής διαπιστώθηκε ότι, όταν ένα υπολογιστικό μηχάνημα εκτός υπολογιστικής υποδομής της Google διαβάζει μαζικά αρχεία από έναν κουβά GCS, δημιουργεί εξερχόμενη κίνηση η οποία χρεώνεται με σημαντικά ποσά. Ο λογαριασμός Google Drive προσαρτήθηκε στο τοπικό υπολογιστικό μηχάνημα της σχολής μέσω του προγράμματος ocamlfuse. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι θα μπορούσαν να δημιουργηθούν και επιπλέον απομακρυσμένα μέσα αποθήκευσης, δεδομένου ότι μια από τις αρετές της βιβλιοθήκης Dask είναι η υποστήριξη μεγάλης ποικιλίας μέσων αποθήκευσης. Το δεύτερο κομμάτι περιλαμβάνει ένα εικονικό μηχάνημα της σχολής και διαδραματίζει διπλό ρόλο. Καταρχήν λειτουργεί ως εξυπηρετητής της βάσης δεδομένων γράφου Neo4j. Σε αυτόν έχει εγκατασταθεί και λειτουργεί αδιάλειπτα ένα ελεύθερης διανομής αυτόνομο στιγμιότυπο εξυπηρετητή Neo4j. Το στιγμιότυπο αυτό εμπλουτίστηκε από δυο βιβλιοθήκες επέκτασης του Neo4j (APOC, GDS) των οποίων οι λειτουργικότητες κρίθηκαν σημαντικές. Το ίδιο ακριβώς μηχάνημα χρησιμοποιείται και για την εκτέλεση από τη γραμμή εντολών της Python εφαρμογής πελάτη καθώς και της εντολής neo4j-admin import που είναι αυτήν που φορτώνει τελικά τα δεδομένα στην βάση δεδομένων γράφου. Οι παραπάνω ρόλοι επιφέρουν στο εικονικό μηχάνημα της σχολής σημαντικό υπολογιστικό και αποθηκευτικό αποτύπωμα. Η αρχιτεκτονική δομή της δημιουργηθείσας υποδομής απεικονίζεται στο Σχήμα 5.1.




Σχήμα 5.1: Αρχιτεκτονικά τμήματα βιβλιογραφικής υποδομής

## 5.3 Εκτέλεση Εφαρμογών Πελάτη Python

Αρχικά ανοίγεται στο τοπικό μηχάνημα ένα τερματικό σε περιβάλλον Windows. Από εκεί δημιουργείται μια ssh σύνδεση με τον εξυπηρετητή sshtou μηχανήματος (kostasvav@asidirop.iee.ihu.gr), όπου είναι εγκατεστημένος ο εξυπηρετητής της βάσης δεδομένων Γράφου Neo4j (localhost). Κατά την εγκαθίδρυση αυτής της σύνδεσης καθορίζονται και οι κανόνες προώθησης πακέτων των θυρών 7474, 7687, 7473 και 8787. Η εντολή που εκτελείται από τη γραμμή εντολών είναι η εξής:

```
ssh -L7474:localhost:7474 -L7687:localhost:7687 -L7473:localhost:7473 -L8787:localhost:8787 kostasvav@asidirop.iee.ihu.gr
```

Στην συνέχεια εκτελείται η παρακάτω εντολή:

```
screen
```

Ουσιαστικά αυτή η εντολή λειτουργεί ως ένας πολυπλέκτης πολλαπλών συνόδων κελύφους (Shell Sessions) μέσα σε μια ssh σύνδεση. Επιτρέπει σε μια διεργασία να τρέχει ακόμα και όταν η ssh σύνδεση με έναν απομακρυσμένο εξυπηρετητή τερματιστεί ή ακόμα και όταν κλείσουμε εντελώς τον τοπικό υπολογιστή. Το βήμα αυτό επιβλήθηκε από το γεγονός ότι η εισαγωγή των διαφόρων κομματιών της βάσης δεδομένων γράφου είναι αρκετά χρονοβόρα και μπορεί να διακοπεί από την εμφάνιση κάποιου απρόβλεπτου γεγονότος. Σε αυτά συγκαταλέγονται η διακοπή ρεύματος, η διακοπή σύνδεσης με το διαδίκτυο ή η διακοπή της ssh σύνδεσης με τον απομακρυσμένο εξυπηρετητή. Η αποσύνδεση της τρέχουσας εκτελούμενης διεργασίας από την ssh σύνδεση γίνεται εκτελώντας την παρακάτω εντολή.

```
ctrl (το κρατάμε συνεχώς πατημένο) + a (το πατάμε και μετα το απελευθερώνουμε) + d
```

Η επανασύνδεσή της σε κάποιο μετέπειτα χρόνο επιτυγχάνεται εκτελώντας την παρακάτω εντολή.

```
screen –r <session-id>
```

, όπου <session-id> είναι ο αριθμός της συνόδου screen με την οποία επιθυμούμε να επανασυνδεθούμε.

Στην συνέχεια και αφού ολοκληρωθεί επιτυχώς η εκτέλεση της εφαρμογής πελάτη Python εντοπίζετε ο αριθμός της συνόδου screen εκτελώντας την παρακάτω εντολή.

```
screen -ls
```

Για να τερματιστεί οριστικά η screen σύνοδος εκτελείται η παρακάτω εντολή.

```
screen -XS <session-id> quit
```

, όπου <session-id> είναι ο αριθμός της συνόδου screen που εντοπίσαμε προηγουμένως και της οποίας την εκτέλεση επιθυμούμε να τερματίσουμε.

Αφού ολοκληρωθεί η εισαγωγή και εκκινήσουμε ξανά τον εξυπηρετητή Neo4j συνδεόμαστε στη βάση δεδομένων γράφου μέσω του διαφυλλιστή, εισάγοντας στην γραμμή διεύθυνσης τον παρακάτω σύνδεσμο.

```
http://localhost:7474/browser
```

Αφού εισάγουμε τα κατάλληλα στοιχεία πιστοποίησης για τον χρήστη neo4j συνδεόμαστε με την απομακρυσμένη βάση δεδομένων γράφου. Στη συνέχεια εκτελούμε εντολές Cypher του Πίνακας 5.1, ώστε να ελέγξουμε ότι η εισαγωγή του αντίστοιχου κομματιού στην βάση δεδομένων γράφου έγινε ορθά. Ουσιαστικά κάθε εντολή υπολογίζει και επιστρέφει το πλήθος των κόμβων/ συσχετίσεων ενός συγκεκριμένου τύπου που έχουν τελικά εισαχθεί στην βάση δεδομένων γράφου. Στο σημείο αυτό επιβάλλεται να αναφέρουμε ότι στον Πίνακας 5.1 χρησιμοποιήθηκαν οι εκδόσεις των εντολών Cypher που κάνουν χρήση του count store. Το countstore αποθηκεύει μεταδεδομένα πληθικότητας για διάφορες οντότητες (Κόμβους, Συσχετίσεις) της βάσης δεδομένων. Ουσιαστικά αυτά τα στατιστικά στοιχεία συμβουλεύεται ο δρομολογητής πλάνων εκτέλεσης (Query Planner), ώστε να επιλέξει το πιο αποδοτικό πλάνο που πρόκειται να εκτελέσει. Η ανάκτηση της πληθικότητας μιας οντότητας από το count store είναι λειτουργία σταθερού χρόνου και συνεπώς σημαντικά ταχύτερη από οποιαδήποτε άλλη μορφή ερωτήματος Cypher.

Πίνακας 5.1: Εντολές Cypher για έλεγχο ορθής εισαγωγής δεδομένων στην βάση δεδομένων γράφου (Χρήση count store σε Κόμβους και σε Σχέσεις)

| Α/ Α | Τύπος Οντότητας | Χρόνος Εκτέλεσης ( ms) | Πλήθος Επιστρεφόμενων Οντοτήτων | Εντολή Cypher | Χρήση count store |
|---|---|---|---|---|---|
| 1 | Κόμβος | 4 | 53422 | MATCH (n:Venue) RETURN count(n) AS Total_Venue_Nodes | NAI |
| 2 | Κόμβος | 4 | 25776 | MATCH (n:Organization) RETURN count(n) AS Total_Organization_Nodes | NAI |
| 3 | Κόμβος | 3 | 243477150 | MATCH (n:Author) RETURN count(n) AS Total_Author_Nodes | NAI |
| 4 | Κόμβος | 4 | 89677467 | MATCH (n:Paper) RETURN count(n) AS Total_Paper_Nodes | NAI |
| 5 | Κόμβος | 3 | 671751 | MATCH (n:FieldStudy) RETURN count(n) AS Total_FieldStudy_Nodes | NAI |
| 6 | Σχέση | 1 | 69064868 | MATCH () –[r: IS_PUBLISHED_IN]-> (:Venue) RETURN count(r) AS Total_IS_PUBLISHED_IN_Relationships | NAI |
| 7 | Σχέση | 2 | 1158002750 | MATCH () –[r:CITES]-> (:Paper) RETURN count(r) AS Total_CITES_Relationships | NAI |
| 8 | Σχέση | 2 | 305674133 | MATCH () –[r:HAS_AUTHOR]-> (:Author) RETURN count(r) AS Total_HAS_AUTHOR_Relationships | NAI |
| 9 | Σχέση | 1 | 681400897 | MATCH () –[r:IS_RELATED_TO_FIELD_STUDY]-> (:FieldStudy) RETURN count(r) AS Total_IS_RELATED_TO_FIELD_STUDY_Relationships | NAI |
| 10 | Σχέση | 2 | 165313597 | MATCH () –[r:IS_AFFILIATED_AT_ORGANIZATION]-> (:Organization) RETURN count(r) AS Total_IS_AFFILIATED_AT_ORGANIZATION_Relationships | NAI |

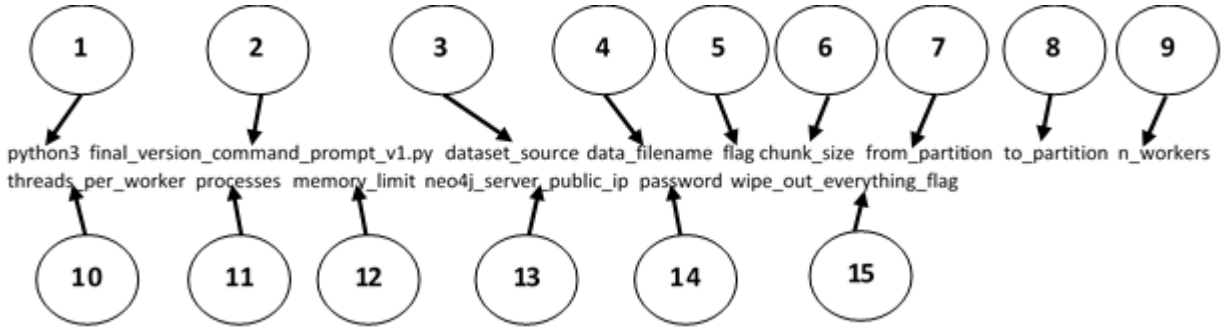
### 5.3.1 Εφαρμογή πελάτης για μαζική εισαγωγή δεδομένων

#### 5.3.1.1 Μέσω δοσοληψιών και οδηγού Python

Στην συγκεκριμένη προσέγγιση ο ρόλος της εφαρμογής πελάτη είναι η απευθείας εισαγωγή των δεδομένων στην βάση δεδομένων γράφου. Συνεπώς κατά την αρχική φάση απαιτείται να ρυθμίσουμε τη μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη σε υψηλή τιμή. Υψηλή τιμή της μνήμης σωρού υπαγορεύει μεγάλη ικανότητα εκτέλεσης πολύπλοκων δοσοληψιών. Υψηλή τιμή της ενδιάμεσης βοηθητικής μνήμης υπαγορεύει ταχύτατη ανάγνωση δεδομένων από τη βάση δεδομένων γράφου, μιας και ελαχιστοποιούνται πλέον οι χρονοβόρες προσπελάσεις στον σκληρό δίσκο. Όλη η υπόλοιπη κύρια μνήμη θα χρησιμοποιηθεί αποκλειστικά από τους κόμβους εργάτες της εφαρμογής πελάτη. Μεταβαίνουμε λοιπόν στο αρχείο ρυθμίσεων /etc/neo4j/neo4j.conf και αποδίδουμε στη μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη τις τιμές που απεικονίζονται παρακάτω.

```
dbms.memory.heap.initial_size=10g
dbms.memory.heap.max_size= 10g
dbms.memory.pagecache.size=24g
```

Στην γραμμή εντολών εκκινούμε την εφαρμογή πελάτη σε Python εκτελώντας την εντολή που απεικονίζεται στο Σχήμα 5.2.




Σχήμα 5.2: Τρόπος εκτέλεσης εφαρμογής πελάτη για εισαγωγή δεδομένων στην βάση δεδομένων γράφου

Η εντολή περιλαμβάνει σημαντικό αριθμό επιπλέον παραμέτρων που επιτρέπουν πλήρη ευελιξία του κομματιού της βάσης δεδομένων που επιθυμούμε να εισάγουμε σε κάθε εκτέλεση. Η σημασιολογία της κάθε παραμέτρου επεξηγείται παρακάτω.

1. python3: Για να προσπελάσουμε τον διερμηνευτή εντολών της έκδοσης v.3.x της Python που είναι εγκατεστημένη στο μηχάνημα.
2. final_version_command_prompt_v1.py: Το όνομα του Python αρχείου πηγαίου κώδικα.
3. dataset_source: Το όνομα του συνόλου δεδομένων στο οποίο επιθυμούμε να εστιάσουμε. Μπορεί να πάρει μια από τις εξής δυο τιμές: ‘mag’, ‘aminer’.
4. data_filename: Το είδος του κόμβου με το οποίο επιθυμούμε να αλληλοεπιδράσουμε. Μπορεί να πάρει μια από τις εξής τιμές: ‘venues’, ‘organizations’, ‘authors’, ‘papers’.
5. flag: Το κομμάτι του γράφου με το οποίο επιθυμούμε να αλληλοεπιδράσουμε κάθε φορά. Το κομμάτι μπορεί να αφορά κόμβους ή συσχετίσεις. Σε περίπτωση που αφορά κόμβους μπορεί να πάρει μια από τις εξής τιμές: ‘venues’, ‘organizations’, ‘authors’, ‘papers’, ‘fieldstudies’. Σε περίπτωση που αφορά συσχετίσεις μπορεί να πάρει μια από τις εξής τιμές: ‘CITES’, ‘IS_PUBLISHED_IN’, ‘IS_RELATED_TO_FIELD_STUDY’, ‘HAS_AUTHOR’, ‘IS_AFFILIATED_AT_ORGANIZATION’.

6. chunk_size: Το μέγεθος του κομματιού βάσει του οποίου θα γίνεται η εισαγωγή των δεδομένων. Η επιλογή του καθορίζεται κυρίως από το μέγεθος της μνήμης σωρού που είναι διαθέσιμη στο υπολογιστικό μηχάνημα που τρέχει ο εξυπηρετητής της βάσης δεδομένων γράφου.
7. from_partition: Τον αριθμό του αρχείου (Partition) από το οποίο θα αρχίσει η εισαγωγή των δεδομένων στη βάση δεδομένων γράφου. Ο αριθμός αυτός συμπεριλαμβάνεται στην εισαγωγή. Σε περίπτωση που δοθεί ως τιμή το -1 η εισαγωγή ξεκινάει από το partition 0.
8. to_partition: Τον αριθμό του αρχείου (Partition) στο οποίο θα σταματήσει η εισαγωγή των δεδομένων στη βάση δεδομένων γράφου. Ο αριθμός αυτός δεν συμπεριλαμβάνεται στην εισαγωγή. Σε περίπτωση που δοθεί ως τιμή το -1 η εισαγωγή σταματάει στο τελευταίο partition.
9. n_workers: Το πλήθος των κόμβων εργατών που επιθυμούμε να διαμορφώσουμε. Καθορίζεται από τους διαθέσιμους πόρους του μηχανήματος στο οποίο θα εκτελεστεί η εφαρμογή πελάτης. Για τιμή 1 επιλέγουμε σειριακή εκτέλεση. Για μεγαλύτερη τιμή επιλέγουμε παραλληλοποιημένη εκτέλεση.
10. threads_per_worker: Το πλήθος των νημάτων εκτέλεσης σε κάθε κόμβο επεξεργασίας. Καθορίζεται από τις υπολογιστικές δυνατότητες του μηχανήματος στο οποίο θα εκτελεστεί η εφαρμογή πελάτης.
11. processes: Το αν κάθε κόμβος επεξεργασίας θα εκτελείται σε ξεχωριστή διαδικασία ή στην ίδια. Πρόκειται για λογική μεταβλητή που, αν της εκχωρηθεί τιμή Αληθής, κάθε κόμβος θα τρέχει τις επεξεργασίες του σε ξεχωριστή διεργασία. Αν της εκχωρηθεί ψευδής τιμή, τότε όλοι οι κόμβοι θα εκτελούν τις επεξεργασίες τους στη μια και μοναδική διεργασία, χρησιμοποιώντας τα νήματα εκτέλεσης που του έχουν ανατεθεί. Γενικά, αν οι επεξεργασίες κάθε κόμβου περιλαμβάνουν αντικείμενα Python που κρατάνε το GIL, τότε είναι προτιμότερο να της ανατεθεί Αληθής τιμή. Αντίθετα, σε αριθμητικούς υπολογισμούς ή σε διαδικασίες εισόδου- εξόδου που δεν κρατάνε το GIL συνιστάται να της ανατεθεί η ψευδής τιμή.
12. memory_limit: Τη συνολική διαθέσιμη μνήμη που πρόκειται να διατεθεί στους κόμβους εργάτες (Workers) για την λειτουργία τους.
13. neo4j_server_public_ip: Το όνομα εξυπηρετητή ή η εξωτερική διεύθυνση πρωτοκόλλου διαδικτύου (Public/ External ip Address) του μηχανήματος που λειτουργεί ως εξυπηρετητής της βάσης δεδομένων γράφου.
14. password: Το συνθηματικό σύνδεσης στην εξ ορισμού βάση δεδομένων γράφου neo4j.
15. wipe_out_everything_flag: Μια λογική μεταβλητή που καθορίζει αν με κάθε επανεκτέλεση της εφαρμογής πελάτη, θα διαγράφονται πλήρως όλα τα προηγούμενα περιεχόμενα της βάσης δεδομένων γράφου. Η εκχώρηση Αληθής τιμής διαγράφει τα πάντα, ενώ η εκχώρηση Ψευδής τιμή τα διατηρεί και απλά προσθέτει επιπλέον κομμάτια. Επειδή τα περιεχόμενα της βάσης δεδομένων γράφου είναι πολύ μεγάλα δεν συνιστάται η διαγραφή όλων των περιεχομένων της βάσης δεδομένων μέσω αυτής της μεταβλητής, μιας και αποδεικνύεται εξαιρετικά χρονοβόρα. Αντίθετα συνιστάται η διαγραφή ολόκληρου του εικονικού μηχανήματος που τρέχει ο εξυπηρετητής της βάσης δεδομένων και η εξ αρχής δημιουργία ενός καινούργιου.

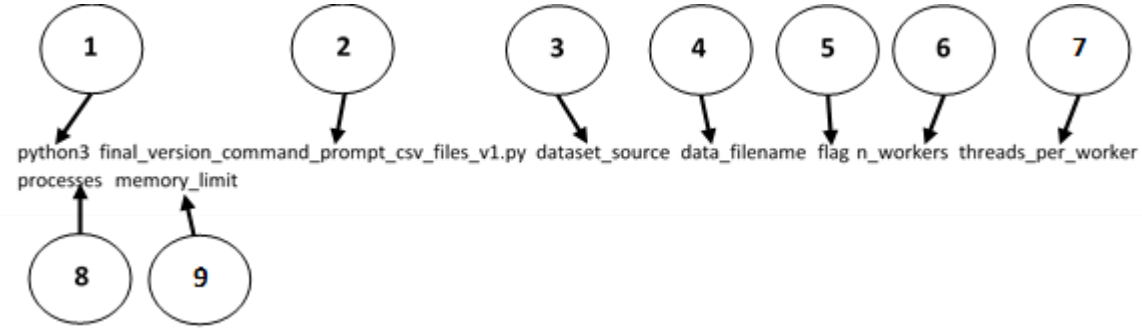
#### 5.3.1.2 Μέσω του εργαλείου neo4j-admin import

Στη τρέχουσα προσέγγιση ο ρόλος της εφαρμογής πελάτη είναι έμμεσος και αφορά στη δημιουργία μιας δομής καταλόγων στο υπολογιστικό μηχάνημα της σχολής που θα περιέχει τα κατάλληλης μορφής αρχεία csv για όλες τις οντότητες του μοντέλου δεδομένων (Αρχεία Κεφαλίδων και Κορμού). Η άμεση μαζική εισαγωγή των δεδομένων στη βάση δεδομένων γράφου θα γίνει μεταγενέστερα μέσω της εκτέλεσης της εντολής neo4j-admin import από τη γραμμή εντολών. Συνεπώς, κατά την αρχική φάση απαιτείται να ρυθμίσουμε τη μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη σε χαμηλή τιμή. Με αυτόν τον τρόπο αφήνουμε όλη την υπόλοιπη κύρια μνήμη να χρησιμοποιηθεί αποκλειστικά από τους κόμβους εργάτες της Python εφαρμογής πελάτη. Μεταβαίνουμε λοιπόν στο αρχείο

ρυθμίσεων /etc/neo4j/neo4j.conf και αποδίδουμε στη μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη τις τιμές που απεικονίζονται παρακάτω.

```
dbms.memory.heap.initial_size=2g
dbms.memory.heap.max_size=2g
dbms.memory.pagecache.size=3g
```

Στην γραμμή εντολών εκκινούμε την εφαρμογή πελάτη σε Python εκτελώντας την εντολή που απεικονίζεται στο Σχήμα 5.3.




Σχήμα 5.3: Τρόπος εκτέλεσης εφαρμογής πελάτη για δημιουργία αρχείων csv

Η εντολή περιλαμβάνει σημαντικό αριθμό επιπλέον παραμέτρων που επιτρέπουν πλήρη ευελιξία του κομματιού της βάσης δεδομένων που επιθυμούμε να εισάγουμε σε κάθε εκτέλεση. Η σημασιολογία της κάθε παραμέτρου επεξηγείται παρακάτω.

1. python3: Για να προσπελάσουμε τον διερμηνευτή εντολών της έκδοσης v.3.x της Python που είναι εγκατεστημένη στο μηχάνημα.
2. final_version_command_prompt_csv_files_v1.py: Το όνομα του Python αρχείου πηγαίου κώδικα.
3. dataset_source: Το όνομα του συνόλου δεδομένων στο οποίο επιθυμούμε να εστιάσουμε. Μπορεί να πάρει μια από τις εξής δυο τιμές: ‘mag’, ‘aminer’.
4. data_filename: Το είδος του κόμβου με το οποίο επιθυμούμε να αλληλοεπιδράσουμε. Μπορεί να πάρει μια από τις εξής τιμές: ‘venues’, ‘organizations’, ‘authors’, ‘papers’.
5. flag: Το κομμάτι του γράφου με το οποίο επιθυμούμε να αλληλοεπιδράσουμε κάθε φορά. Το κομμάτι μπορεί να αφορά κόμβους ή συσχετίσεις. Σε περίπτωση που αφορά κόμβους, μπορεί να πάρει μια από τις εξής τιμές: ‘venues’, ‘organizations’, ‘authors’, ‘papers’, ‘fieldstudies’. Σε περίπτωση που αφορά συσχετίσεις, μπορεί να πάρει μια από τις εξής τιμές: ‘CITES’, ‘IS_PUBLISHED_IN’, ‘IS_RELATED_TO_FIELD_STUDY’, ‘HAS_AUTHOR’, ‘IS_AFFILIATED_AT_ORGANIZATION’.
6. n_workers: Το πλήθος των κόμβων εργατών που επιθυμούμε να διαμορφώσουμε. Καθορίζεται από τους διαθέσιμους πόρους του μηχανήματος στο οποίο θα εκτελεστεί η εφαρμογή πελάτης. Για τιμή 1 επιλέγουμε σειριακή εκτέλεση. Για μεγαλύτερη τιμή επιλέγουμε παραλληλοποιημένη εκτέλεση.
7. threads_per_worker: Το πλήθος των νημάτων εκτέλεσης σε κάθε κόμβο επεξεργασίας. Καθορίζεται από τις υπολογιστικές δυνατότητες του μηχανήματος στο οποίο θα εκτελεστεί η εφαρμογή πελάτης.
8. processes: Το αν κάθε κόμβος επεξεργασίας θα εκτελείται σε ξεχωριστή διαδικασία ή στην ίδια. Πρόκειται για λογική μεταβλητή που, αν της εκχωρηθεί τιμή Αληθής, κάθε κόμβος θα τρέχει τις επεξεργασίες του σε ξεχωριστή διεργασία. Αν της εκχωρηθεί ψευδής τιμή, τότε όλοι οι κόμβοι θα εκτελούν τις επεξεργασίες τους στη μια και μοναδική διεργασία χρησιμοποιώντας τα νήματα εκτέλεσης που του έχουν ανατεθεί. Γενικά, αν οι επεξεργασίες κάθε κόμβου περιλαμβάνουν

αντικείμενα Python που κρατάνε το GIL, τότε είναι προτιμότερο να της ανατεθεί Αληθής τιμή. Αντίθετα, σε αριθμητικούς υπολογισμούς ή σε διαδικασίες εισόδου- εξόδου που δεν κρατάνε το GIL, συνιστάται να της ανατεθεί η ψευδής τιμή.

9. memory_limit: Τη συνολική διαθέσιμη μνήμη που πρόκειται να διατεθεί στους κόμβους εργάτες (Workers) για την λειτουργία τους.

## 5.4 Επίλογος

Στο κεφάλαιο αυτό περιγράφηκε αναλυτικά τη συνολική αρχιτεκτονική δομή της βιβλιογραφικής υποδομής που στοχεύουμε να δημιουργήσουμε. Η αρχιτεκτονική απαρτίζεται από κομμάτια λογισμικού και υλικού που συνεργάζονται αρμονικά μεταξύ τους. Τα κομμάτια λογισμικού περιλαμβάνουν μια εφαρμογή πελάτη σε γλώσσα προγραμματισμού Python. Η πρώτη εφαρμογή πελάτης υλοποιεί μια ETL τύπου ροή επεξεργασίας δυο σταδίων. Σκοπός της είναι ο μετασχηματισμός και η φόρτωση των βιβλιογραφικών δεδομένων στη βάση δεδομένων γράφου. Ο μετασχηματισμός των δεδομένων βασίστηκε στη βιβλιοθήκη DASK, ενώ όσον αφορά τη φόρτωση ακολουθήθηκαν δυο προσεγγίσεις: μέσω του επίσημα υποστηριζόμενου οδηγού Python του Neo4j και μέσω της εντολής import του εργαλείου διαχείρισης neo4j-admin. Επίσης επεξηγείται αναλυτικά ο τρόπος εκτέλεσης των δυο εφαρμογών πελάτη που υλοποιούν τις δυο προσεγγίσεις φόρτωσης της βάσης δεδομένων γράφου. Τέλος, όσον αφορά τα τμήματα υλικού της υποδομής, αυτά αφορούν καταρχήν την αποθήκευση των αρχικών ανεπεξέργαστων βιβλιογραφικών δεδομένων και το υπολογιστικό μηχάνημα του πανεπιστημίου όπου θα λάβουν χώρα οι επεξεργασίες. Η αναλυτική επεξήγησή τους έγινε σε άλλες ενότητες της διπλωματικής εργασίας. Αξίζει να αναφερθεί ότι λόγοι έλλειψης πόρων επέβαλαν στο εικονικό μηχάνημα της σχολής που μας παραχωρήθηκε να διαδραματίσει διπλό ρόλο. Σε ένα μέρος του αποφασίστηκε να εγκατασταθεί ο εξυπηρετητής της βάσης δεδομένων γράφου Neo4j. Στο υπόλοιπο μέρος του αποφασίστηκε να εκτελεστούν στη γραμμή εντολών η εφαρμογή πελάτης σε Script Mode καθώς και η εντολή neo4j-admin import. Γενικά ο διαμοιρασμός των πόρων του εικονικού μηχανήματος έγινε με σχετική δικαιοσύνη. Στο σημείο αυτό αξίζει να σημειώσουμε την εξαιρετικά μεγάλη πολυπλοκότητα του εγχειρήματος δημιουργίας της βιβλιογραφικής υποδομής. Προκειμένου να την αντιμετωπίσουμε βασιστήκαμε σε μια καλά σχεδιασμένη ερευνητική μεθοδολογία πολλαπλών βημάτων προκαθορισμένης ακολουθιακής δομής. Αυτήν αποτέλεσε τον οδηγό, ώστε να μην αποπροσανατολιστούμε στο όλο εγχείρημα και αποτελεί αντικείμενο του επόμενου κεφαλαίου.

# Κεφάλαιο 6ο: Ερευνητική Μεθοδολογία Εργασίας

## 6.1 Εισαγωγή

Στα πλαίσια της διπλωματικής εργασίας κληθήκαμε να υλοποιήσουμε μια υποδομή διαχείρισης βιβλιογραφικών δεδομένων. Αυτού του είδους τα δεδομένα παρουσιάζουν κάποια χαρακτηριστικά που καθιστούν δύσκολη την αποτελεσματική διαχείρισή τους. Σε αυτά συγκαταλέγονται ο μεγάλος όγκος, ο συχνός ρυθμός μεταβολής, η ανομοιογένεια όταν προέρχονται από διαφορετικές πηγές και η ισχυρή διασύνδεσή τους. Έχει αποδειχθεί ότι ειδικά στο τελευταίο χαρακτηριστικό υποκρύπτεται σημαντική αξία που μπορεί να πυροδοτήσει τις κατάλληλες αποφάσεις που θα δημιουργήσουν πλεονέκτημα έναντι του ανταγωνισμού. Η ανακάλυψή πολύτιμων μη προφανών συσχετίσεων μπορεί να αποκαλύψει μη αποδοτικά πρότυπα λειτουργίας, μη αναμενόμενες τάσεις, λανθασμένες τεχνικές προσέγγισης διαφόρων ζητημάτων, μη αποδοτική διαχείριση κεφαλαίων κ.α. Συνεπώς η ανακάλυψή τους είναι κομβικής σημασίας για έναν οργανισμό ιδιαίτερα στο σημερινό άκρως ανταγωνιστικό και ευμετάβλητο διεθνές τοπίο, όπου τα περιθώρια λάθους είναι περιορισμένα.

Από τα αρχικά στάδια έγινε αντιληπτό ότι, προκειμένου να αντιμετωπίσουμε επιτυχημένα τις παραπάνω προκλήσεις, έπρεπε να βασιστούμε σε μια μεθοδολογία αυστηρά καθορισμένων σταδίων. Κάθε ένα από τα στάδια αυτά έπρεπε να αποτελεί μέλος μιας επακριβώς καθορισμένης ακολουθιακής δομής και να υλοποιεί ένα προκαθορισμένο σύνολο αρμοδιοτήτων χαμηλής αλληλοεπικάλυψης. Ταυτόχρονα κάθε στάδιο έπρεπε να επιδεικνύει χαμηλό βαθμό σύνδεσης με όλα τα υπόλοιπα, ώστε η απομάκρυνση τους να αφαιρεί από τη συνολική ροή επεξεργασίας μόνο την αντίστοιχη λειτουργικότητα που υλοποιεί το συγκεκριμένο στάδιο. Κάθε στάδιο έπρεπε να λειτουργεί με έναν plug-and-play τρόπο, ώστε η προσθαφαίρεσή του στη συνολική δομή επεξεργασίας να μην παρεμποδίζει τις λειτουργικότητες των υπολοίπων σταδίων. Τέλος τα ακολουθούμενα στάδια έπρεπε να διευκολύνουν την επαναληπτική και σταδιακή υλοποίηση της υποδομής, ώστε να αντιμετωπιστεί η αυξημένη πολυπλοκότητα που τη χαρακτηρίζει. Τα επιμέρους στάδια της μεθοδολογίας που επιλέχθηκε απεικονίζονται στο Σχήμα 6.1.


Σχήμα 6.1: Στάδια μεθοδολογίας υλοποίησης βιβλιογραφικής υποδομής

## 6.2 Στάδια μεθοδολογίας εργασίας

### 6.2.1 1ο Στάδιο – Διερευνητική ανάλυση συνόλων δεδομένων

Προκαταρκτικά απαιτείται να εκτελέσουμε μια ενδελεχή διερευνητική ανάλυση των συνόλων δεδομένων (Datasets Exploratory Analysis). Ένα τέτοιο στάδιο επιβάλλεται όταν τα δεδομένα προέρχονται από διαφορετικές πηγές, συλλέγονται με διαφορετικές αυτοματοποιημένες διαδικασίες και είναι διαφορετικής ποιότητας. Το στάδιο αυτό αποσκοπεί στο να κατανοήσουμε βαθιά την ποιότητα και τις συσχετίσεις τους. Μια τέτοια διερευνητική διαδικασία αποκαλύπτει προβληματικές (λανθασμένες, ελλείπουσες, ασύμβατες, ανομοιογενείς) τιμές και επιτρέπει τον ρεαλιστικό έλεγχο υποθέσεων σχετικά με τα σύνολα δεδομένων. Η φάση αυτή καθορίζει σε μεγάλο βαθμό την αποδοτικότητα και αποτελεσματικότητα των επόμενων σταδίων επεξεργασίας. Στα πλαίσια αυτής της διπλωματικής η διερευνητική ανάλυση θα πραγματοποιηθεί κάνοντας έξυπνη χρήση των προηγμένων δυνατοτήτων αναζήτησης και φιλτραρίσματος της επαγγελματικής έκδοσης του συντάκτη κειμένου EmEditor.

#### 6.2.1.1 Mag

- Πεδίο Papers.id: Υπάρχει παντού χωρίς διπλότυπες τιμές. Δεν περιέχει ως τιμές το κενό αλφαριθμητικό και το αλφαριθμητικό «&NA;».
- Πεδίο Papers.title: Υπάρχει παντού χωρίς διπλότυπες τιμές. Δεν περιέχει ως τιμές το κενό αλφαριθμητικό και το αλφαριθμητικό «&NA;». Δεν περιέχει ως τιμές μόνο αριθμούς ή ακαταλαβίστικους χαρακτήρες. Σε κάποιες περιπτώσεις περιέχει πολλαπλούς χαρακτήρες“\r”, “\t” και “\n”.
- Πεδίο Papers.keywords: Δεν εμφανίζεται πουθενά. Η πληροφορία δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Σε αυτήν την περίπτωση η πληροφορία μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Aminer (Mag – Aminer Papers Outer Cross-Reference). Για εκείνες τις περιπτώσεις που δεν θα υπάρξει αντιστοιχία θα εκχωρηθεί η εξ’ ορισμού τιμή της κενής λίστας.
- Πεδίο Papers.lang: Δεν εμφανίζεται πουθενά. Σε αυτήν την περίπτωση η πληροφορία θα προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Aminer (Mag – Aminer Papers Outer Cross-Reference). Για εκείνες τις εγγραφές για τις οποίες δεν θα υπάρξει κάποια αντιστοιχία θα εκχωρηθεί η εξ’ ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Papers.year: Υπάρχει σε όλες τις εγγραφές Papers και δεν υπάρχουν ελλείπουσες τιμές.
- Πεδίο Papers.publisher: Υπάρχει παντού, απλά σε κάποιες εγγραφές έχει ως τιμή την εξ’ ορισμού τιμή του κενού αλφαριθμητικού. Σε κάποιες από τις παραπάνω περιπτώσεις η πληροφορία publisher μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Aminer (Mag – Aminer Papers Outer Cross-Reference). Για εκείνες τις περιπτώσεις που δε θα προκύψει κάποια συσχέτιση προφανώς θα εξακολουθήσουν να έχουν την τιμή του κενού αλφαριθμητικού. Σε κάποιες περιπτώσεις περιέχει πολλαπλούς χαρακτήρες “\r”, “\t” και “\n”.
- Πεδίο Papers.doc_type:Υπάρχει σε όλες τις εγγραφές Papers, όμως η πλειοψηφία των εγγραφών έχει ως τιμή το κενό αλφαριθμητικό.
- Πεδίο Papers.doi: Στην πλειοψηφία των εγγραφών δεν υπάρχει καθόλου. Για αυτές τις εγγραφές η πληροφορία θα προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Aminer (Mag – Aminer Papers Outer Cross-Reference). Για εκείνες τις εγγραφές για τις οποίες δεν θα υπάρξει κάποια αντιστοιχία θα εκχωρηθεί η εξ’ ορισμού τιμή του κενού αλφαριθμητικού.

- Πεδίο Papers.venue: Δεν υπάρχει σε όλες τις εγγραφές των αρχείων Papers. Σε εκείνες που υπάρχει η τιμή του είναι ένα μη κενό αντικείμενο Json (Non-Empty Json Object) που περιέχει το πεδίο name και το πεδίο id. Δεν εμφανίζονται σε όλες τις περιπτώσεις και τα δυο παραπάνω πεδία. Συγκεκριμένα το πεδίο Papers.venue.name εμφανίζεται πάντα, ενώ το πεδίο Papers.venue.id όχι. Το πεδίο venue.name δεν περιλαμβάνει την τιμή «&NA;». Σε καμία εγγραφή το πεδίο δεν έχει ως τιμή ένα κενό αντικείμενο Json. Σε αυτές τις περιπτώσεις οι ελλείπουσες τιμές θα συμπληρωθούν με εσωτερική αλληλοσυσχέτιση με το αρχείο Venues του συνόλου δεδομένων Mag (MagPapers – Venues Inner Cross-Reference ). Επίσης κάποιες ελλείπουσες τιμές θα συμπληρωθούν με εξωτερική αλληλοσυσχέτιση των αρχείων Papers με τις αντίστοιχες εγγραφές του συνόλου δεδομένων Aminer. Τέλος για εκείνες τις εγγραφές για τις οποίες δε θα καταστεί δυνατή η διασταύρωση της πληροφορίας venue θα εκχωρηθεί η εξ' ορισμού τιμή του κενού αντικειμένου Json.
- Πεδίο Papers.authors: Πρόκειται για μια λίστα με αντικείμενα Json, κάθε ένα από τα οποία μπορεί να περιέχει οποιονδήποτε συνδυασμό των πεδίων name, id, org και org_id. Υπάρχει σε όλες τις εγγραφές των αρχείων Papers και δεν υπάρχει ως τιμή η κενή λίστα [] ή κάποια λίστα που να περιέχει αποκλειστικά άδεια αντικείμενα Json της μορφής {} ή κάποια λίστα που να περιέχει αποκλειστικά αντικείμενα Json της μορφής {"name": "&NA;"}. Το πεδίο name σε καμία εγγραφή δεν περιέχει την τιμή «&NA;». Δεν είναι υποχρεωτικό να εμφανίζονται όλα τα παραπάνω πεδία. Συγκεκριμένα εμφανίζονται όλοι οι δυνατοί συνδυασμοί όσον αφορά την εμφάνισή τους στα αντικείμενα της λίστας των authors. Σε αυτές τις περιπτώσεις οι ελλείπουσες τιμές θα συμπληρωθούν με εσωτερική αλληλοσυσχέτιση με τα αρχεία Authors και Affiliations του συνόλου δεδομένων Mag (Mag Papers – Authors – Affiliations Inner Cross - Reference). Οι υπόλοιπες ελλείπουσες τιμές θα συμπληρωθούν με εξωτερική αλληλοσυσχέτιση με τα αρχεία Papers του συνόλου δεδομένων Aminer (Mag – Aminer Outer Cross - Reference). Τελικά για οποιαδήποτε εγγραφή των αρχείων Papers δεν καταστεί δυνατόν να ανακτηθεί ένα κομμάτι της πληροφορίας authors θα της εκχωρηθεί η κατάλληλη εξ' ορισμού τιμή.
- Πεδίο Authors.pubs: Υπάρχει σε όλες τις εγγραφές των αρχείων Authors. Επίσης δεν περιέχει σε καμία εγγραφή την τιμή της κενής λίστας ή μια λίστα που να περιέχει αποκλειστικά άδεια αντικείμενα Json. Κάθε αντικείμενο περιέχει τα πεδία i, r, που εκφράζουν το id του paper που έγραψε ένας συγγραφέας καθώς και τη σειρά του στη συγγραφική ομάδα. Όταν εμφανίζεται το πεδίο "pubs" τα πεδία "i" και "r" εμφανίζονται πάντα σε όλα τα αντικείμενα της λίστας. Το πεδίο i μπορεί να διαδραματίσει κομβικό ρόλο για την ανάκτηση πληροφοριών κατά τη διαδικασία της εσωτερικής αλληλοσυσχέτισης με τα αρχεία papers του ίδιου συνόλου δεδομένων. Χρησιμοποιείται συνδυαστικά με το name ή και το id του κάθε αντικειμένου author του πεδίου papers.authors του οποίου τις ελλείπουσες τιμές προσπαθούμε να εντοπίσουμε και να συμπληρώσουμε. Πρέπει να ταυτίζεται πλήρως με το id της εγγραφής του αρχείου paper με το οποίο προσπαθούμε να το αλληλοσυσχετίσουμε εσωτερικά. Κάθε φορά που εμφανίζεται το πεδίο pubs εμφανίζονται και τα δυο παραπάνω πεδία.
- Πεδίο Authors.last_known_aff_id:Υπάρχει σε αρκετές, αλλά όχι σε όλες τις εγγραφές των αρχείων Authors. Δεν αντιστοιχεί πάντα στο πεδίο Papers.authors.org_id των αρχείων Papers. Επειδή όμως το πεδίο last_known_aff_id δεν εμφανίζεται πάντα σε όλες τις εγγραφές των αρχείων Authors, δεν είναι ασφαλής τρόπος ανάκτησης του authors.org_id. Τέλος για εκείνες τις περιπτώσεις που δεν καθίσταται δυνατός ο εντοπισμός του org_id θα του εκχωρηθεί η κατάλληλη εξ' ορισμού τιμή.
- Πεδίο Affiliations.type: Δεν υπάρχει σε καμία από τις εγγραφές του αρχείου Affiliations. Η πληροφορία δεν μπορεί να ανακτηθεί με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Μπορεί να επιχειρηθεί να ανακτηθεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων του συνόλου δεδομένων AMiner. Για κάθε άλλη περίπτωση θα της εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδία Authors.orgs & Authors.org:Δεν υπάρχουν σε καμία εγγραφή των αρχείων Authors. Η πληροφορία αυτή είναι διαθέσιμη μέσω των πεδίων Papers.authors.org & Papers.authors.org_id.

- Πεδίο Papers.fos: Υπάρχει στις περισσότερες εγγραφές, αλλά όχι σε όλες. Δεν υπάρχει ως τιμή η κενή λίστα ή μια λίστα με αποκλειστικά άδεια αντικείμενα Json {} ή μια λίστα με αποκλειστικά αντικείμενα της μορφής {“name”: “&NA;”}. Η πληροφορία αυτή δεν μπορεί να ανακτηθεί με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Επίσης δεν μπορεί να ανακτηθεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων AMiner που δεν περιέχουν το συγκεκριμένο πεδίο. Συνεπώς για κάθε εγγραφή που δεν περιέχει το πεδίο θα της εκχωρηθεί η εξ’ ορισμού τιμή της κενής λίστας.
- Πεδίο Papers.references:Υπάρχει σε αρκετές, αλλά όχι σε όλες τις εγγραφές. Επίσης δεν υπάρχει ως τιμή η κενή λίστα. Για κάθε εγγραφή που δεν περιέχει το πεδίο η πληροφορία δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση από τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Επίσης δεν είναι ανακτήσιμη ούτε με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές του συνόλου δεδομένων AMiner μιας και δεν περιέχουν το πεδίο αυτό. Συνεπώς για κάθε τέτοια εγγραφή θα της εκχωρηθεί η εξ’ ορισμού τιμή της κενής λίστας.
- Πεδίο Authors.position:Δεν υπάρχει σε καμία εγγραφή των αρχείων Authors. Η πληροφορία αυτή δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Μπορεί να συμπληρωθεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές Authors του συνόλου δεδομένων AMiner. Για κάθε εγγραφή που δεν καταστεί δυνατόν η συμπλήρωση της πληροφορίας θα εκχωρηθεί η εξ’ ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Authors.tags:Δεν υπάρχει σε καμία εγγραφή των αρχείων Authors. Η πληροφορία αυτή δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Μπορεί να να συμπληρωθεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές Authors του συνόλου δεδομένων AMiner. Για κάθε εγγραφή που δεν καταστεί δυνατή η συμπλήρωση της πληροφορίας θα εκχωρηθεί η εξ’ ορισμού τιμή της κενής λίστας.
- Πεδίο Venues.JournalId, Venues.ConferenceId: Όλες οι εγγραφές περιλαμβάνουν ένα από τα παραπάνω πεδία. Τα πεδία καθορίζουν τον τύπο του μέσου δημοσίευσης και είναι αμοιβαία αποκλειόμενα. Συνεπώς, όταν εμφανίζεται το πεδίο “JournalId” στο πεδίο Venues.type, θα εκχωρηθεί η τιμή “ Journal”. Αντίθετα, όταν εμφανίζεται το πεδίο “ConferenceId”, στο πεδίο Venues.type θα εκχωρηθεί η τιμή “Conference”.

#### 6.2.1.2 Aminer

- Πεδίο Papers.id:Υπάρχει παντού χωρίς διπλότυπες τιμές.Δεν περιέχει ως τιμές το κενό αλφαριθμητικό και το αλφαριθμητικό «&NA;».
- Πεδίο Papers.title: Σε ελάχιστες περιπτώσεις δεν εμφανίζεται, ενώ στις υπόλοιπες δεν έχει διπλότυπες τιμές.Δεν περιέχει ως τιμές το κενό αλφαριθμητικό και το αλφαριθμητικό «&NA;». Επίσης σε αρκετές περιπτώσεις περιέχει ως τιμή αποκλειστικά αριθμούς ή συνδυασμό αριθμών και μπερδεμένων χαρακτήρων. Δεν κατέστη δυνατό να ξεκαθαριστεί τι αναπαριστά αυτός ο αριθμός ή αν επρόκειτο για σφάλμα κατά την συλλογή των δεδομένων. Για αυτές τις εγγραφές δεν εντοπίστηκε κάποια αντιστοιχία ανάμεσα στα σύνολα δεδομένων Aminer και Mag. Συνεπώς καθίσταται αδύνατη η ανάκτηση της πληροφορίας του τίτλου του εγγράφου. Επειδή όμως διαθέτουν πολλές άλλες χρήσιμες πληροφορίες επιλέχτηκε να συμπεριληφθούν στη βάση δεδομένων γράφου. Σε όλες τις παραπάνω περιπτώσεις οι ελλείπουσες και προβληματικές τιμές μπορούν νασυμπληρωθούν με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Mag. Για κάθε εγγραφή που δεν θα καταστεί δυνατόν να ανακτηθεί η τιμή της μέσω της παραπάνω διαδικασίας, θα της εκχωρηθεί η εξ’ ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Papers.keywords:Δεν υπάρχει σε όλες τις εγγραφές. Επίσης δεν υπάρχει ως τιμή η κενή λίστα ή κάποια λίστα που να περιέχει άδεια αντικείμενα Json. Για κάθε εγγραφή που δεν υπάρχει το πεδίοη πληροφορία δεν είναι πλέον ανακτήσιμη με αλληλοσυσχέτιση με άλλα αρχεία του ίδιου ή διαφορετικού συνόλου δεδομένων. Συνεπώς θα της εκχωρηθεί η εξ’ ορισμού τιμή της κενής λίστας.

- Πεδίο Papers.lang:Υπάρχει στην συντριπτική πλειοψηφία των εγγραφών Papers, με εξαίρεση κάποιες ελάχιστες περιπτώσεις. Σε κάθε τέτοια περίπτωση θα της εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο year:Δεν υπάρχει σε όλες τις εγγραφές, ενώ, όπου εμφανίζεται, δεν υπάρχουν ελλείπουσες τιμές. Σε εκείνες τις περιπτώσεις που το πεδίο δεν εμφανίζεται θα εκχωρηθεί η εξ' ορισμού τιμή -1. Η τελική τιμή αυτού του πεδίου σε κάθε εγγραφή θα καθοριστεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές Paperstου συνόλου δεδομένων Mag.
- Πεδίο Papers.publisher: Δεν υπάρχει πουθενά. Συνεπώς η τιμή του θα προκύψει από την εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές του συνόλου δεδομένων Mag (AMiner–Mag Papers Outer Cross - Reference).
- Πεδίο doc_type: Δεν υπάρχει πουθενά. Συνεπώς η πληροφορία μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Mag(Mag-Aminer Papers Outer Cross-reference).Για εκείνες τις εγγραφές Papers για τις οποίες δε θα προκύψει κάποια συσχέτιση θα εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο doi:Στην πλειοψηφία των εγγραφών εμφανίζεται το πεδίο. Για τις υπόλοιπες εγγραφές η πληροφορία μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Mag (Mag-Aminer Papers Outer Cross-reference). Για εκείνες τις εγγραφές για τις οποίες δεν θα υπάρξει κάποια αντιστοιχία, θα εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Papers.venue:Δεν υπάρχει σε όλες τις εγγραφές Papers. Σε εκείνες τις περιπτώσεις που εμφανίζεται η τιμή του είναι ένα μη κενό Jso παντικείμενο (Non-Empty Json Object)που περιέχει το πεδίο name και το πεδίο id. Δεν είναι υποχρεωτικό να εμφανίζονται και τα δυο παραπάνω πεδία. Σε κάποιες περιπτώσεις το πεδίο έχει ως τιμή ένα κενό αντικείμενο Json. Σε όλες τις παραπάνω περιπτώσεις οι ελλείπουσες τιμές θα συμπληρωθούν με εσωτερική αλληλοσυσχέτιση με το αρχείο Venue του συνόλου δεδομένων Aminer (Aminer Papers–Venues Inner Cross-Reference ). Επίσης κάποιες ελλείπουσες τιμές μπορεί να συμπληρωθούν με εξωτερική αλληλοσυσχέτιση των αρχείων Papers των συνόλων δεδομένων Mag και Aminer(Mag – Aminer Papers Outer Cross-Reference ).Τέλος για εκείνες τις εγγραφές, για τις οποίες δε θα καταστεί δυνατή η διασταύρωση της πληροφορίας του μέσου δημοσίευσης, θα εκχωρηθεί στο πεδίο venue η εξ' ορισμού τιμή του κενού αντικειμένου Json.
- Πεδίο Papers.authors:Υπάρχει σε όλες τις εγγραφές Papers. Πρόκειται για μια λίστα με αντικείμενα Json, κάθε ένα από τα οποία μπορεί να περιέχει οποιονδήποτε συνδυασμό των πεδίων name, id και org. Το πεδίο id αντιστοιχεί στον author, ενώ το πεδίο org_id δεν εμφανίζεται σε καμία εγγραφή. Σε κάποιες περιπτώσεις έχει ως τιμή την κενή λίστα [] ή σε κάποιες άλλες περιπτώσεις μια λίστα που να περιέχει αποκλειστικά άδεια αντικείμενα Json της μορφής {} ή σε κάποιες άλλες περιπτώσεις περιέχει αποκλειστικά αντικείμενα Json της μορφής {"name": "&NA;". Σε αυτές τις περιπτώσεις στο πεδίο authors θα εκχωρηθεί ως τιμή η κενή λίστα και δε θα επιχειρηθεί η ολική ανακατασκευή του από τα υπόλοιπα αρχεία του συνόλου δεδομένων AMinerμε εσωτερική αλληλοσυσχέτιση, μιας και δεν υπάρχει καμία αντιστοιχία. Σε κάποιες άλλες περιπτώσεις περιέχει ένα έγκυρο αντικείμενο Json, όπου το πεδίο "name" έχει την τιμή "&NA;". Δεν είναι υποχρεωτικό να εμφανίζονται όλα τα παραπάνω πεδία. Σε αυτές τις περιπτώσεις οι ελλείπουσες τιμές μπορούν να συμπληρωθούν με εσωτερική αλληλοσυσχέτιση με τα αρχεία Authors και Affiliations του συνόλου δεδομένων Aminer (AminerPapers–Authors–Affiliations Inner Cross - Reference). Οι υπόλοιπες ελλείπουσες τιμές μπορούν να συμπληρωθούν με εξωτερική αλληλοσυσχέτιση με τα αρχεία Papers του συνόλου δεδομένων Mag (Mag–Aminer Outer Cross - Reference). Τελικά, για όποια εγγραφή των αρχείων Papers δεν μπορέσει να ανακτηθεί η πληροφορία σύνδεσης με τους συγγραφείς τους, θα της εκχωρηθεί η κατάλληλη εξ' ορισμού τιμή.
- Πεδίο Authors.pubs: Δεν εμφανίζεται σε όλες τις εγγραφές των αρχείων Authors. Προφανώς πρόκειται για συγγραφείς οι οποίοι δεν έχουν πραγματοποιήσει ακόμα καμία δημοσίευση ή δεν είναι διαθέσιμη η πληροφορία. Συνεπώς, το αντίστοιχο πεδίο n_pubs, που εκφράζει το πλήθος των

δημοσιεύσεων του κάθε συγγραφέα, θα έχει την τιμή 0 ή δεν θα εμφανίζεται καθόλου. Σε καμία εγγραφή δεν περιέχει ως τιμή την κενή λίστα ή μια λίστα που να περιέχει αποκλειστικά άδεια αντικείμενα Json. Κάθε αντικείμενο περιέχει τα πεδία i, r, που εκφράζουν το id του paper που έγραψε ένας συγγραφέας καθώς και τη σειρά του μέσα στην συγγραφική ομάδα. Όταν εμφανίζεται το πεδίο "pubs", τα πεδία "i" και "r" εμφανίζονται πάντα σε όλα τα αντικείμενα της λίστας. Το πεδίο i διαδραματίζει κομβικό ρόλο για την ανάκτηση πληροφοριών κατά τη διαδικασία της εσωτερικής αλληλοσυσχέτισης με τα αρχεία papers του ίδιου συνόλου δεδομένων.Χρησιμοποιείται συνδυαστικά με το name ή και το id του κάθε αντικειμένου author του πεδίου authors των αρχείων papers του οποίου τις ελλείπουσες τιμές προσπαθούμε να εντοπίσουμε και να συμπληρώσουμε. Πρέπει να ταυτίζεται πλήρως με το id της εγγραφής του αρχείου paper με το οποίο προσπαθούμε να το αλληλοσυσχετίσουμε εσωτερικά. Κάθε φορά που εμφανίζεται το πεδίο pubs, εμφανίζονται και τα δυο παραπάνω πεδία.

- Πεδίο Authors.last_known_aff_id: Δεν υπάρχει σε καμία εγγραφή των αρχείων Authors. Η πληροφορία μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις εγγραφές των αρχείων Mag που περιέχουν την πληροφορία αυτή (AMiner–Mag Outer Cross Reference).
- Πεδίο Affiliations.type:Υπάρχει παντού σε όλες τις εγγραφές του αρχείου Affiliations. Δεν υπάρχει πουθενά ως τιμή το κενό αλφαριθμητικό ή την τιμή "&NA;".
- Πεδία Authors.orgs & Authors.org: Δεν υπάρχουν σε όλες τις εγγραφές των αρχείων Authors. Επίσης τα πεδία αυτά εμφανίζονται σε όλους τους δυνατούς μεταξύ τους συνδυασμούς. Το πεδίο Authors.org εμπεριέχεται μέσα στη λίστα Authors.orgs. Το πεδίο Authors.org αντιστοιχεί στο πεδίο Papers.authors.org.
- Πεδίο Papers.fos: Δεν υπάρχει σε καμία εγγραφή. Η πληροφορία αυτή δεν μπορεί να ανακτηθεί με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Μπορεί να γίνει προσπάθεια να ανακτηθεί με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Papers του συνόλου δεδομένων Mag. Για κάθε εγγραφή που δε θα βρεθεί κάποια αντιστοιχία, θα της εκχωρηθεί η εξ' ορισμού τιμή της κενής λίστας.
- Πεδίο Papers.references: Δεν υπάρχει σε καμία εγγραφή. Η πληροφορία αυτή δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση από τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Για κάποιες εγγραφές είναι ανακτήσιμη με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές του συνόλου δεδομένων Mag που περιέχουν το αντίστοιχο πεδίο. Για κάθε άλλη περίπτωση, θα της εκχωρηθεί η εξ' ορισμού τιμή της κενής λίστας.
- Πεδίο Authors.position: Δεν υπάρχει στην πλειοψηφία των εγγραφών του συνόλου δεδομένων. Δεν περιέχει την τιμή του κενού αλφαριθμητικού ή την τιμή "&NA;". Επίσης, η πληροφορία αυτή δεν είναι ανακτήσιμη με εσωτερική αλληλοσυσχέτιση απο τα υπόλοιπά αρχεία του ίδιου συνόλου δεδομένων. Δεν είναι ανακτήσιμη ούτε με εξωτερική αλληλοσυσχέτιση από την αντίστοιχη εγγραφή του συνόλου δεδομένων Mag που δεν περιέχει το πεδίο. Για κάθε τέτοια εγγραφή, θα της εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Authors.tags: Δεν υπάρχει στην πλειοψηφία των εγγραφών του συνόλου δεδομένων. Επίσης, δεν υπάρχει πουθενά η τιμή της κενής λίστας ή μια λίστα που να περιέχει αποκλειστικά άδεια αντικείμενα Json. Αποτελεί μια λίστα από αντικείμενα Json, κάθε ένα από τα οποία περιέχει τα πεδία "t" και "w".Όταν εμφανίζεται το πεδίο "tags" το πεδίο "t" εμφανίζεται πάντα, ενώ το πεδίο "w" δεν εμφανίζεται σε ορισμένες περιπτώσεις. Για εκείνες τις εγγραφές που δεν περιέχουν το πεδίο, η τιμή του δεν μπορεί να βρεθεί με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων. Επίσης, δεν μπορεί να βρεθεί ούτε με εξωτερική αλληλοσυχέτιση με τις αντίστοιχες εγγραφές του συνόλου δεδομένων που δεν περιέχουν το πεδίο αυτό. Συνεπώς, σε αυτές τις περιπτώσεις, θα τις εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.
- Πεδίο Venues.JournalId, Venues. JournalId.ConferenceId: Δεν υπάρχουν πουθενά. Τα δυο αυτά πεδία καθορίζουν τον τύπο δημοσίευσης ενός Paper. Ο τύπος του μέσου δημοσίευσης δεν μπορεί να προκύψει με εσωτερική αλληλοσυσχέτιση με τα υπόλοιπα αρχεία του ίδιου συνόλου δεδομένων που δεν περιέχουν την συγκεκριμένη πληροφορία. Συνεπώς, μπορεί να προκύψει με εξωτερική αλληλοσυσχέτιση με τις αντίστοιχες εγγραφές των αρχείων Venues του συνόλου

δεδομένων Mag. Για κάθε εγγραφή που δεν θα μπορέσει να συμπληρωθεί η τιμή της, θα της εκχωρηθεί η εξ' ορισμού τιμή του κενού αλφαριθμητικού.

#### 6.2.1.3 Linking Pairs

Σε αρκετές εγγραφές δεδομένων σύζευξης των οντοτήτων (Papers, Authors, Venues, Affiliations) συνόλων δεδομένων (Mag, Aminer) εντοπίστηκαν διπλότυπες περιπτώσεις σύνδεσης. Συγκεκριμένα, αναφέρουμε μια περίπτωση όπου ένας κωδικός "mid" του συνόλου δεδομένων Mag αντιστοιχούσε ταυτόχρονα σε δυο κωδικούς "aid" του συνόλου δεδομένων που όμως αναφέρονται στο ίδιο αντικείμενο (Papers, Authors, Venues, Affiliations). Η περίπτωση αυτή εμφανίστηκε και αντίστροφα και αφορούσε διπλότυπους κωδικούς "aid" του συνόλου δεδομένων Aminer. Αυτή η κατάσταση μπορεί να δημιουργήσει σημαντικά προβλήματα κατά τη συγχώνευση των αντίστοιχων αρχείων των δυο συνόλων δεδομένων, μιας και δημιουργεί διπλότυπες εγγραφές στο τελικό αρχείο που προκύπτει. Σε κάθε περίπτωση οι διπλότυπες εμφανίσεις πρέπει να απορρίπτονται.

### 6.2.2 2ο Στάδιο – Προεπεξεργασία δεδομένων

Στο στάδιο αυτό θα πρέπει να επιχειρηθεί η συμπλήρωση των περισσοτέρων τιμών των πεδίων των οντοτήτων των δυο συνόλων δεδομένων. Συγκεκριμένα, αν σε οποιαδήποτε ιδιότητα μιας εγγραφής δεν υπάρχει ή εντοπιστεί προβληματική τιμή, προσπαθεί να την αντικαταστήσει με εσωτερική ή εξωτερική αλληλοσυσχέτιση της συγκεκριμένης εγγραφής με τις αντίστοιχες εγγραφές άλλων αρχείων. Η εσωτερική αλληλοσυσχέτιση προσπαθεί να εντοπίσει τη σωστή τιμή σε άλλα αρχεία του ίδιου συνόλου δεδομένων. Αντίθετα, η εξωτερική αλληλοσυσχέτιση προσπαθεί να εντοπίσει τη σωστή τιμή σε αρχεία διαφορετικού συνόλου δεδομένων. Συνιστάται αρχικά να εφαρμόζεται μια διαδικασία εσωτερικής αλληλοσυσχέτισης η οποία θα ακολουθείται από μια διαδικασία εξωτερικής αλληλοσυσχέτισης των αρχείων. Η διαδικασία εσωτερικής αλληλοσυσχέτισης μπορεί να βασιστεί στην πλεονασματική πληροφορία που περιέχεται στις υπόλοιπες οντότητες του ίδιου συνόλου δεδομένων. Τελικά, θα περιλαμβάνει τη συγχώνευση δυο δομών δεδομένων (Dask Bag, Dask DataFrame). Η διαδικασία εξωτερικής αλληλοσυσχέτισης μπορεί να βασιστεί στα στατιστικά στοιχεία σύνδεσης (Linking Pairs) που είναι διαθέσιμα στην ιστοσελίδα του OAG V2.1. Γενικά, η διαδικασία εξωτερικής αλληλοσυσχέτισης μπορεί να υλοποιηθεί ακολουθώντας δυο διαφορετικές προσεγγίσεις. Στην πρώτη προσέγγιση αρχικά συγχωνεύονται σε δυο στάδια οι δομές δεδομένων που αναπαριστούν τις αντίστοιχες οντότητες των δυο συνόλων δεδομένων. Αυτή η συγχώνευση δυο σταδίων περιλαμβάνει αρχικά τη συγχώνευση της δομής δεδομένων του ενός συνόλου δεδομένων με τη δομή δεδομένων που αναπαριστά τα Linking Pairs. Στην συνέχεια η δομή δεδομένων που δημιουργείται συγχωνεύεται με τη δομή δεδομένων της αντίστοιχης οντότητας του άλλου συνόλου δεδομένων. Έπειτα στην τεράστια δομή δεδομένων που δημιουργείται αλληλοσυχετίζονται οι αντίστοιχες τιμές των πεδίων ενδιαφέροντος, ώστε να αντιμετωπιστούν προβληματικές τιμές. Τέλος, στην τεράστια δομή δεδομένων που δημιουργείται εφαρμόζεται μια ροή επεξεργασίας δεδομένων η οποία τελικά παράγει τα ειδικής μορφής αρχεία csv που θα αποτελέσουν την είσοδο της εντολής neo4j-adminimport. Στη δεύτερη προσέγγιση αρχικά συγχωνεύεται σε ένα στάδιο η δομή δεδομένων ενός συνόλου δεδομένων με τη δομή δεδομένων που αναπαριστά τα Linking Pairs. Στη συνέχεια, στη δομή δεδομένων που δημιουργείται εφαρμόζεται μια ροή επεξεργασίας δεδομένων η οποία τελικά παράγει τα ειδικής μορφής αρχεία csv που θα αποτελέσουν την είσοδο της εντολής neo4j-admin import. Τέλος, η πληροφορία της δομής δεδομένων που αναπαριστά το άλλο σύνολο δεδομένων ενσωματώνεται στη βάση δεδομένων μέσω μιας εφαρμογής πελάτη σε Python. Αυτή εκτελεί εκτεταμένης κλίμακας λειτουργίες αναδιάταξης σε όλη την έκταση της υφιστάμενης βάσης δεδομένων (που περιέχει τα δεδομένα του άλλου συνόλου δεδομένων), χρησιμοποιώντας χρήσιμες

λειτουργικότητες της βιβλιοθήκης APOC. Συνιστάται η χρήση ενός ισχυρού εικονικού μηχανήματος προκειμένου να αντιμετωπιστούν αποδοτικά και αποτελεσματικά οι απαιτητικοί υπολογισμοί μεγάλης κλίμακας του συγκεκριμένου σταδίου. Το στάδιο αυτό δεν υλοποιήθηκε λόγω έλλειψης χρόνου και λόγω της απόφασης να εισαχθούν στη βάση δεδομένων γράφου αποκλειστικά δεδομένα του συνόλου δεδομένων Mag.

### 6.2.3 3ο Στάδιο – Φιλτράρισμα δεδομένων

Η φάση αυτή προσπαθεί να εντοπίσει και να απορρίψει όλες εκείνες τις βιβλιογραφικές οντότητες των οποίων οι τιμές σε ορισμένα πεδία δεν είναι συμβατές με τις προδιαγραφές του τομέα προβλήματος. Συγκεκριμένα, στα πλαίσια της διπλωματικής αποφασίστηκε να απορρίπτονται όλες οι οντότητες Papers οι οποίες δεν περιέχουν το πεδίο doi ή που περιέχουν κενή αλφαριθμητική τιμή στο πεδίο αυτό. Επίσης, αποφασίστηκε να απορρίπτονται όλες οι εγγραφές Papers που περιέχουν ως τιμή του πεδίου doc_type την τιμή «Patent». Λόγω του σταδίου του φιλτραρίσματος των οντοτήτων Papers, τελικά αποθηκεύτηκαν μόνο περίπου το 37.32 % των αρχικά διαθέσιμων κόμβων Papers (89677467 κόμβοι σε σύνολο 240.255.240 κόμβων). Η επιλογή αυτή έγινε ώστε από τα αρχικά στάδια να απορρίψουμε όλες εκείνες τις εγγραφές Papers που είναι εξαιρετικά χαμηλής ποιότητας. Η χαμηλή ποιότητα αναφέρεται στο πλήθος των πεδίων που είτε δεν περιέχουν καθόλου ή που έχουν κενή τιμή ή τιμή ακατάληπτη. Τέτοιου είδους προβλήματα ανακύπτουν συχνά στις αυτοματοποιημένες διαδικασίες συλλογής δεδομένων από διάφορες πηγές. Όλες οι άλλες οντότητες του τομέα προβλήματος δε φιλτράρονται και περνάνε κανονικά στο επόμενο στάδιο επεξεργασίας.

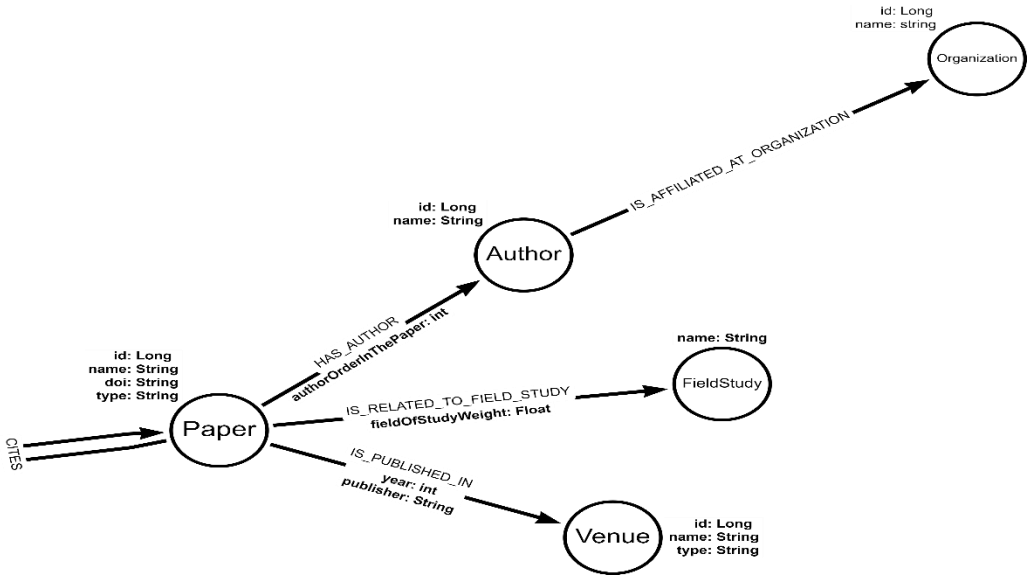
### 6.2.4 4ο Στάδιο –Καθαρισμός δεδομένων

Η φάση αυτή προσπαθεί να βελτιώσει τα περισσότερα από τα προβλήματα που εντοπίστηκαν κατά τις προηγούμενες φάσεις. Σε εκείνες τις ιδιότητες που δεν καταστεί δυνατός ο εντοπισμός της τιμής τους, εκχωρείται μια εξ' ορισμού τιμή. Επίσης, σε ορισμένα πεδία έγινε προσαρμογή του τύπου δεδομένων, ώστε να βελτιωθεί η αποδοτικότητα των επόμενων σταδίων της ροής επεξεργασίας (Data Processing Pipeline) σε αυτά. Για όσα αντικείμενα του πεδίου Authors.tags το πεδίο "t" μετά τον καθαρισμό του από τους ακαταλαβίστικους χαρακτήρες έχει ως τιμή το κενό αλφαριθμητικό, τότε αφαιρείται ολοκληρωτικά το αντίστοιχο αντικείμενο από τη λίστα. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι η πλατφόρμα Neo4j έχει ως πάγια πολιτική να μην αποθηκεύει κενές ιδιότητες. Αυτό έχει ως αποτέλεσμα οι ιδιότητες αυτές να μην εμφανίζονται στους αντίστοιχους κόμβους ή συσχετίσεις, όταν σε μεταγενέστερη φάση επιχειρηθεί η φόρτωση των δεδομένων στη βάση δεδομένων γράφου. Συνεπώς, σε οποιοδήποτε ερώτημα που περιλαμβάνει την συγκεκριμένη ιδιότητα, ο αντίστοιχος κόμβος ή συσχέτιση που δεν θα την περιλαμβάνουν δε θα επιστρέφεται ως αποτέλεσμα. Σε όσες περιπτώσεις στο πεδίο Papers.title ή Papers.publisher περιέχονται πολλαπλοί χαρακτήρες προώθησης του κέρσορα στην αρχή της γραμμής "\r" (Carriage Return), χαρακτήρες οριζόντιου στηλογνώμονα "\t" (Horizontal Tab) καθώς και χαρακτήρες προώθησης μιας γραμμής μπροστά "\n" (Line Feed), αυτοί αφαιρούνται. Τέλος, σε όλα τα πεδία κειμένου θα εφαρμοστούν επεξεργασίες που απομακρύνουν τους κενούς ή μη αποδεκτούς χαρακτήρες.

### 6.2.5 5ο Στάδιο – Δημιουργία αρχικού μοντέλου δεδομένων βάσης γράφου

Στα πλαίσια της διπλωματικής κληθήκαμε να διαχειριστούμε δυο πολύ μεγάλα σύνολα βιβλιογραφικών δεδομένων. Από την αρχή υιοθετήθηκε η προσέγγιση να επικεντρώσουμε την προσοχή μας στο ένα από τα δυο και μελλοντικά να ενσωματώσουμε την πληροφορία του άλλου σε αυτό. Αυτή η προσέγγιση επιβλήθηκε από το μεγάλο μέγεθός τους και από τη σημαντική ετερογένεια

που παρουσίαζαν. Συγκεκριμένα, επιλέχθηκε το Mag να αποτελέσει το βασικό μας σύνολο μιας και είναι το μεγαλύτερο σε μέγεθος σύνολο δεδομένων και έχει την υψηλότερη ποιότητα δεδομένων. Στη συνέχεια, σε δεύτερη φάση, μπορούμε να ενσωματώσουμε και την πληροφορία του συνόλου Aminer στη βάση δεδομένων γράφου. Αποτέλεσμα αυτής της επιλογής είναι να ξεκινήσουμε με ένα λιτό αρχικό μοντέλο δεδομένων που απεικονίζεται στο Σχήμα 6.2.




Σχήμα 6.2: Αρχικό μοντέλο βάσης δεδομένων γράφου

Αυτό το μοντέλο δεδομένων περιλαμβάνει τους παρακάτω κόμβους:

1. Paper: αναπαριστά τις οντότητες των επιστημονικών εργασιών/ άρθρων.
2. Author: αναπαριστά τις οντότητες των συγγραφέων.
3. Venue: αναπαριστά τις οντότητες των μέσων δημοσίευσης των εργασιών.
4. Organization: αναπαριστά τις οντότητες των οργανισμών/ ινστιτούτων απασχόλησης των συγγραφέων.
5. FieldStudy: αναπαριστά τις οντότητες των επιστημονικών τομέων που ανήκουν οι επιστημονικές εργασίες.

Αυτό το μοντέλο δεδομένων περιλαμβάνει επίσης και τις παρακάτω σχέσεις:

1. CITES: αναπαριστά τη συσχέτιση αναφοράς πηγών μεταξύ επιστημονικών εργασιών.
2. IS_PUBLISHED_IN: αναπαριστά τη συσχέτιση επιστημονικών εργασιών και μέσων δημοσίευσης.
3. HAS_AUTHOR: αναπαριστά τη συσχέτιση επιστημονικών εργασιών και συγγραφέων.
4. IS_RELATED_TO_FIELD_STUDY: αναπαριστά τη συσχέτιση επιστημονικών εργασιών και επιστημονικών τομέων στους οποίους ανήκουν.
5. IS_AFFILIATED_AT_ORGANIZATION: αναπαριστά τη συσχέτιση συγγραφέων και οργανισμών στους οποίους απασχολούνται.

Στη συνέχεια ακολουθούν οι κόμβοι και οι μεταξύ τους σχέσεις σε γραφική μορφή Ascii - Art:

```
(:Paper) - [:CITES] -> (:Paper)

(:Paper) - [:IS_PUBLISHED_IN] -> (:Venue)

(:Paper) - [:HAS_AUTHOR] -> (:Author)

(:Paper) - [:IS_RELATED_TO_FIELD_STUDY] -> (:FieldStudy)

(:Author ) - [:IS_AFFILIATED_AT_ORGANIZATION] -> (:Organization)
```

Οι παραπάνω κόμβοι και σχέσεις εμπλουτίστηκαν με αρκετές ιδιότητες. Οι περισσότερες έχουν διακοσμητικό ρόλο, δηλαδή επιλέχθηκαν ώστε να αποθηκεύουν χρήσιμες πληροφορίες για τις οντότητες και τις συσχετίσεις τους. Οι πληροφορίες αυτές μπορούν στην συνέχεια να ανακτηθούν σε κατάσταση παραγωγικής λειτουργίας εκτελώντας ενδιαφέρουσες ερωτήσεις σε Cypher. Ωστόσο σε κάθε κόμβο προστέθηκε μια ιδιότητα id η οποία δεν περιέχει ουσιαστικά δεδομένα, αλλά περισσότερο σχετίζεται με την ακεραιότητα των δεδομένων (Data Integrity) και τη βελτίωση της αποδοτικότητας διάσχισης του γράφου κατά την εκτέλεση ερωτημάτων. Εξαίρεση αποτελούν οι κόμβοι τύπου FieldStudy οι οποίοι δεν περιέχουν ιδιότητα id ως μοναδικό αναγνωριστικό, αλλά μια ιδιότητα name. Για αυτό τον σκοπό, σε κάθε κόμβο έχει δημιουργηθεί ένας περιορισμός μοναδικότητας ιδιότητας κόμβου (Unique Node Property Constraint) ως προς την ιδιότητα id (name για τους κόμβους τύπου FieldStudy). Αυτός ο περιορισμός θα εξασφαλίσει τη μη ύπαρξη διπλότυπων κόμβων κατά τη μαζική φόρτωση των δεδομένων στη βάση δεδομένων γράφου. Επίσης, επειδή με κάθε περιορισμό δημιουργείται αυτόματα και ένα ευρετήριο (ως προς την ιδιότητα που αναφέρεται ο περιορισμός), θα επιταχυνθεί σημαντικά και κάθε μελλοντική διάσχιση του γράφου ως προς την ιδιότητα id (name για τους κόμβους τύπου FieldStudy). Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι στα πλαίσια της διπλωματικής δε δημιουργήθηκαν επιπλέον διαφορετικούς τύπους περιορισμών (Node property Existence Constraints, Relationship Property Existence Constraints, Node Key Constraints), μιας και η ελεύθερης διανομής έκδοση του Neo4j CE που χρησιμοποιήθηκε δεν τους υποστηρίζει (ενώ η Neo4j EE τους υποστηρίζει). Στην συνέχεια, ακολουθούν οι περιορισμοί μοναδικότητας ιδιότητας κόμβου που δημιουργήθηκαν.

### 6.2.6 6ο Στάδιο - Εφαρμογή παραμετροποιήσεων/ βελτιστοποιήσεων Neo4j

Οι παραμετριοποιήσεις/ βελτιστοποιήσεις που περιγράφονται στην ενότητα αυτή εφαρμόζονται κυρίως στην πρώτη προσέγγιση εισαγωγής των δεδομένων με την οποία πειραματιστήκαμε. Η προσέγγιση αυτή εισάγει απευθείας τα δεδομένα στη βάση δεδομένων γράφου, κάνοντας χρήση δοσοληψιών και του επίσημα υποστηριζόμενου από το Neo4j οδηγού Python. Τελικά, για λόγους αποδοτικότητας εισάγαμε τα δεδομένα στη βάση γράφου, ακολουθώντας τη δεύτερη προσέγγιση που κάνει χρήση της εντολής import του εργαλείου διαχείρισης neo4j-admin.

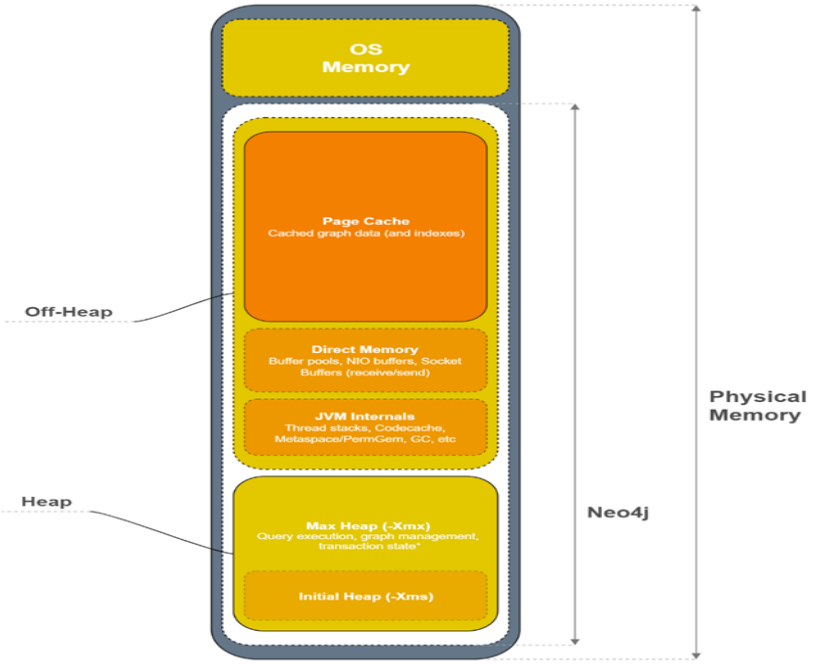
```
CREATE CONSTRAINT papers_uniqueness_restriction IF NOT EXISTS ON
(p:Paper) ASSERT  p.id IS UNIQUE

CREATE CONSTRAINT authors_uniqueness_restriction IF NOT EXISTS ON
(a:Author) ASSERT  a.id IS UNIQUE

CREATE CONSTRAINT venues_uniqueness_restriction IF NOT EXISTS ON
(v:Venue) ASSERT  v.id IS UNIQUE

CREATE CONSTRAINT field_studies_uniqueness_restriction IF NOT EXISTS ON
(fs:FieldStudy) ASSERT  fs.name IS UNIQUE

CREATE CONSTRAINT organizations_uniqueness_restriction IF NOT EXISTS ON
(o:Organization) ASSERT  o.id IS UNIQUE
```

#### 6.2.6.1 Παραμετροποίηση μνήμης

##### 6.2.6.1.1 Παραμετροποίηση κύριας μνήμης

Η ορθολογική διαχείριση της διαθέσιμης μνήμης στο υπολογιστικό μηχάνημα που τρέχει ο εξυπηρετητής Neo4j διαδραματίζει καταλυτικό ρόλο στην αποδοτική λειτουργία της βάσης

δεδομένων γράφου. Ωστόσο, δε υπάρχει ένας γενικός κανόνας που μπορεί να εφαρμοστεί σε όλες τις περιπτώσεις. Αντίθετα, οι ρυθμίσεις επιλέγονται κατά περίπτωση και βασίζονται σε εκτιμήσεις του τρέχοντος συνολικού μεγέθους, της αναμενόμενης μελλοντικής επέκτασης (μεγέθυνση/σμίκρυνση) και των κυρίαρχων προτύπων προσπέλασης της βάσης (Database Access Patterns). Παρόλα αυτά, υπάρχουν κάποιες βέλτιστες πρακτικές που έχουν αποδειχτεί χρήσιμες και καλό είναι να ακολουθούνται. Σε γενικές γραμμές, η συνολική διαθέσιμη μνήμη του μηχανήματος όπου τρέχει ο εξυπηρετητής Neo4j κατανέμεται με βάση τον τύπο Total Physical Memory = Heap + Off-Heap + OS Memory και απεικονίζεται στο Σχήμα 6.3.



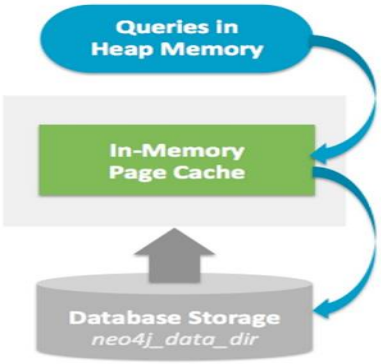

Σχήμα 6.3: Κατανομή διαθέσιμης μνήμης υπολογιστικού μηχανήματος (Πηγή [48])

Η μνήμη λειτουργικού συστήματος (OS Memory) δεσμεύεται για την ομαλή λειτουργία του ίδιου του λειτουργικού συστήματος. Χρησιμοποιείται για την αποδοτική εκτέλεση διεργασιών που δε σχετίζονται κατά ανάγκη με τη λειτουργία του ίδιου του εξυπηρετητή Neo4j. Το μέγεθός της καθορίζεται από τον φόρτο εργασίας (Server Workload) του εξυπηρετητή όπου τρέχει το Neo4j στιγμιότυπο και το μέγεθος της μνήμης RAM που διαχειρίζεται. Σε περίπτωση που το μηχάνημα χρησιμοποιείται αποκλειστικά για την εξυπηρέτηση του Neo4jτότε μια επιλογή από 2 – 4 GB, κρίνεται λογική. Σε κάθε περίπτωση η τελική επιλογή θα πρέπει να καλύπτει άνετα τις ανάγκες του λειτουργικού συστήματος, ώστε να μην ανταλλάσσει συχνά δεδομένα μεταξύ κύριας μνήμης και σκληρού δίσκου, υποβαθμίζοντας την απόδοση.

Η μνήμη σωρού (JVM Heap Memory) είναι το κομμάτι της μνήμης όπου αποθηκεύονται τα δεδομένα της τρέχουσας συνόδου εκτέλεσης, όπως τα δημιουργημένα αντικείμενα των κλάσεων, οι πίνακες, μεταβλητές κλπ. Επίσης, σε αυτήν την μνήμη αποθηκεύονται και τα δεδομένα που σχετίζονται με τη διαχείριση των βάσεων δεδομένων από το σύστημα διαχείρισης βάσεων δεδομένων, η κατάσταση των εκτελούμενων δοσοληψιών (Transactions State) και τα δεδομένα των ερωτήσεων (Query Execution) που εκτελούνται στο πλαίσιο των δοσοληψιών. Η μνήμη αυτή θα πρέπει να είναι γρήγορη και μεγάλη σε μέγεθος, ώστε να μπορούν να υποστηριχτούν αποδοτικά όλες οι παραπάνω απαιτητικές λειτουργίες. Ωστόσο, δεν πρέπει να είναι και υπερβολικά μεγάλη, γιατί μπορεί να οδηγήσει σε μακροχρόνιες λειτουργίες αυτόματης συλλογής αχρησιμοποίητων/ διαγραμμένων αντικειμένων GC

που θα μείωνε την αποδοτική λειτουργία του συστήματος. Σε καμία περίπτωση δεν πρέπει να εκχωρηθούν περισσότερα από 31 GB στη μνήμη σωρού, καθώς αυτό θα απενεργοποιήσει τη συμπίεση των δεικτών αντικειμένων (Compressed OOPS) στο JVM και θα κάνει λιγότερο αποτελεσματική τη χρήση της. Συνήθως μια επιλογή από 8 –16 GB κρίνεται λογική ανάλογα με της ροές επεξεργασίας και τη διαθέσιμη μνήμη του υπολογιστικού μηχανήματος.

Η ενδιάμεση μνήμη (Page Cache) χρησιμοποιείται για την αποθήκευση κομματιών των δεδομένων της βάσης δεδομένων Neo4j καθώς και των ευρετηρίων προσπέλασής τους (Indexes). Είναι σημαντικό να αποθηκεύονται όσο το δυνατόν μεγαλύτερα κομμάτια της βάσης γράφου σε αυτήν την ενδιάμεση μνήμη για να αποφεύγονται χρονοβόρες προσπελάσεις στον σκληρό δίσκο που υποβαθμίζουν την απόδοση. Η επιλογή του μεγέθους είναι ευρεστική (Heuristic) και εξαρτάται από το αν η βάση δεδομένων ήδη υφίσταται ή πρόκειται να εισαχθεί. Στην πρώτη περίπτωση καθορίζεται ως μέγεθος ενδιάμεσης μνήμης το συνολικό άθροισμα των μεγεθών των αρχείων $NEO4J_HOME/data/graph.db/neostore.*.db συν μια επιπλέον πρόβλεψη επέκτασης 20 % του προηγούμενου υπολογισμένου συνολικού μεγέθους. Στην δεύτερη περίπτωση εισάγεται στη βάση ένα ποσοστό των προς εισαγωγή δεδομένων (1/ 100) και το μέγεθος που προκύπτει το πολλαπλασιάζεται με αυτό το ποσοστό (100). Η χρησιμότητα της ενδιάμεσης βοηθητικής μνήμης απεικονίζεται στο Σχήμα 6.4.




Σχήμα 6.4: Χρησιμότητα ενδιάμεσης βοηθητικής μνήμης (Πηγή [49])

Ως χρήστες συνιστάται να ρυθμίζουμε, μέσω του αρχείου /etc/neo4j/neo4j.conf, μόνο τη μνήμη σωρού και τη βοηθητική ενδιάμεση μνήμη. Το υπόλοιπο διαθέσιμο κομμάτι μνήμης ανατίθεται για την ομαλή λειτουργία του λειτουργικού συστήματος. Όσον αφορά την μνήμη σωρού, μπορούμε να καθορίσουμε την αρχική και τη μέγιστη ποσότητα μνήμης που μπορεί να της ανατεθεί. Συνίσταται οι δυο ρυθμίσεις να έχουν την ίδια τιμή, ώστε το JVM να μην αλλάζει το μέγεθος της μνήμης σωρού. Σε διαφορετική περίπτωση, απαιτείται μια πλήρης σάρωση συλλογής σκουπιδιών (Full Garbage Collection) στη μνήμη σωρού με αρνητικές συνέπειες στην απόδοση του συστήματος. Όσον αφορά την ενδιάμεση βοηθητική μνήμη, είναι επιθυμητό να ρυθμιστεί σε υψηλή τιμή, ώστε να αποφεύγονται συχνές μακροχρόνιες προσπελάσεις δεδομένων της βάσης δεδομένων στον σκληρό δίσκο. Μεταξύ των δυο παραπάνω μνημών υπάρχει μια δυναμική ισορροπία που την καθορίζει το είδος της επεξεργασίας που κυριαρχεί κάθε φορά στις ροές επεξεργασίας που εφαρμόζουμε στη βάση δεδομένων Neo4j. Σε περίπτωση που η ροή επεξεργασίας χαρακτηρίζεται από έντονες και μακροσκελείς λειτουργίες εγγραφής/ ενημέρωσης μεγάλων όγκων δεδομένων στη βάση, τότε είναι προτιμότερο να εκχωρούμε μεγαλύτερο κομμάτι μνήμης στη μνήμη σωρού και να μειώνουμε αντίστοιχα το κομμάτι της ενδιάμεσης βοηθητικής μνήμης. Προς αυτήν την κατεύθυνση πρέπει να κινηθούμε και σε περίπτωση που η ροή επεξεργασίας μας περιλαμβάνει πολλές και μακροσκελείς ταυτοχρονικές δοσοληψίες (Concurrent Transactions) που πρέπει να εκτελεστούν ταχύτατα επάνω σε

μεγάλο όγκο δεδομένων. Αντίθετα, σε περίπτωση που οι προς εκτέλεση δοσοληψίες είναι λίγες και περιλαμβάνουν κυρίως αναγνώσεις δεδομένων από την βάση δεδομένων στον σκληρό δίσκο, τότε πρέπει να μεγιστοποιήσουμε το κομμάτι μνήμης που εκχωρούμε στην ενδιάμεση βοηθητική μνήμη για λόγους αύξησης της απόδοσης. Προς αυτήν την κατεύθυνση πρέπει να κινηθούμε και σε περίπτωση που η βάση δεδομένων μας περιλαμβάνει πολλά ευρετήρια που επιταχύνουν την προσπέλαση του γράφου μας κατά την εκτέλεση των ερωτημάτων μας.

Οι παραπάνω κανόνες έχει αποδειχτεί στην πράξη ότι στις περισσότερες περιπτώσεις λειτουργούν ικανοποιητικά. Ωστόσο το Neo4j παρέχει αυτοματοποιημένες προτάσεις μνήμης μέσω της εντολής memrec του εργαλείου διαχείρισης neo4j-admin. Συγκεκριμένα, παρέχει λογικές συστάσεις καθορισμού της αρχικής και μέγιστης ποσότητας μνήμης σωρού καθώς και της ενδιάμεσης μνήμης. Στην συνέχεια, μπορούμε να τις εφαρμόσουμε ρυθμίζοντας τις κατάλληλες εντολές του αρχείου ρυθμίσεων /etc/neo4j/neo4j.conf.

```
neo4j-admin memrec --memory=58g --verbose
```

Τελικά αποφασίστηκε η παρακάτω κατανομή μνήμης.

```
dbms.memory.heap.initial_size=22300m
dbms.memory.heap.max_size=22300m
dbms.memory.pagecache.size=25g
```

#### 6.2.6.1.2 Παραμετροποίηση δευτερεύουσας μνήμης

Το Neo4j υλοποιεί την έννοια της λογικής διαγραφής έναντι της φυσικής διαγραφής. Δηλαδή οποιαδήποτε διαγραμμένη οντότητα της βάσης γράφου (Κόμβος, Σχέση, Ιδιότητα, Ετικέτα, Τύπος Σχέσης κ.α.) δεν τη διαγράφει αμέσως, ούτε απελευθερώνει τον αντίστοιχο χώρο στο λειτουργικό σύστημα. Αντίθετα την επισημαίνει ως διαγραμμένη και συνεπώς υποψήφια προς χρήση, όταν μελλοντικά απαιτηθεί. Αυτό το επιτυγχάνει μέσω του μηχανισμού των αρχείων καταγραφής δοσοληψιών και μέσωτης χρήσης διαφορετικών αρχείων για τα id's των διαγραμμένων οντοτήτων. Ουσιαστικά, αυτά τα αρχεία περιέχουν τα id's των διαγραμμένων οντοτήτων (ένα ξεχωριστό αρχείο για κάθε διαφορετική οντότητα) και λειτουργούν ως κάδοι ανακύκλωσης των διαθέσιμων προς χρήση id's. Συγκεκριμένα, κάθε φορά που διαγράφεται μια οντότητα από τη βάση δεδομένων γράφου, ο αντίστοιχος χώρος που καταλάμβανε στο αρχείο μαρκάρεται (δεν απελευθερώνεται) ως ελεύθερος, δημιουργείται ένα αρχείο καταγραφής για τη δοσοληψία που έκανε την διαγραφή και το id της διαγραμμένης οντότητας αποθηκεύεται στο αντίστοιχο αρχείο των id's. Για να είμαστε πιο ακριβείς, η δοσοληψία που πραγματοποίησε τη διαγραφή καθώς και το αντίστοιχο id της διαγραμμένης οντότητας αποθηκεύονται σε πρώτη φάση στην ενδιάμεση βοηθητική μνήμη. Σε μεταγενέστερη φάση και μετά από την εκτέλεση ενός σημείου ελέγχου (Checkpoint Execution) αποθηκεύονται τα δεδομένα στα αντίστοιχα αρχεία στον σκληρό δίσκο. Αυτός ο μηχανισμός της λογικής διαγραφής έχει το πλεονέκτημα της αυξημένης απόδοσης και κλιμάκωσης. Ωστόσο, σε περιβάλλον παραγωγικής λειτουργίας, όπου οι εισαγωγές και διαγραφές πραγματοποιούνται με καταιγιστικό ρυθμό, μπορεί να οδηγήσει στο παράδοξο φαινόμενο να αυξάνεται το μέγεθος της βάσης δεδομένων γράφου μετά από μαζικές διαγραφές οντοτήτων της βάσης. Επίσης, τα αρχεία αποθήκευσης των οντοτήτων του γράφου μπορεί να γίνουν αρκετά αραιά και κατακερματισμένα. Ανάλογα με το μέγεθός τους, τον βαθμό κατακερματισμού και την ένταση χρήσης της βάσης γράφου, μπορεί να επηρεαστεί αρνητικά η

απόδοση. Όπως εξηγήσαμε προηγουμένως, αυτό οφείλεται στον μηχανισμό της λογικής διαγραφής του Neo4j που επηρεάζει τρείς διαφορετικούς τύπους αρχείων. Καταρχήν, στα αρχεία στον δίσκο που είναι αποθηκευμένες οι ίδιες οι οντότητες που διαγράφηκαν (αρχεία neostore.*store.db). Μετά στα αρχεία που περιέχουν τα id's των διαγραμμένων οντοτήτων προς μελλοντική επαναχρησιμοποίησή τους (αρχεία neostore.*store.db.id). Τέλος, στα αρχεία καταγραφής δοσοληψιών (Transaction Log Files, Logical Files) που δημιουργούνται για κάθε δοσοληψία διαγραφής που εκτελέστηκε. Η πολιτική ελέγχου των αρχείων καταγραφής είναι απολύτως προσαρμόσιμη από τον τελικό χρήστη και περιγράφεται στην υποενότητα που ακολουθεί. Ο χρήστης μέσω κατάλληλων ρυθμίσεων στο αρχείο /etc/neo4j/neo4j.conf του Neo4j μπορεί να καθορίσει μια πιο ήπια ή επιθετική πολιτική διατήρησης των αρχείων καταγραφής επηρεάζοντας την συνολική επιβάρυνση στον σκληρό δίσκο. Η συμπεριφορά των αρχείων με τα id's είναι απολύτως προβλέψιμη. Κάθε φορά που διαγράφουμε κάποιες οντότητες, τα αντίστοιχα αρχεία αυξάνονται σε μέγεθος, μιας και τα id'sτων διαγραμμένων οντοτήτων προστίθενται σε αυτά. Αντίθετα, κάθε φορά που εισάγουμε νέες οντότητες στην βάση δεδομένων γράφου, τα αντίστοιχα αρχεία μειώνονται σε μέγεθος, μιας και αφαιρούνται id's από τα αρχεία αυτά, για να αποδοθούν στις οντότητες που δημιουργούνται. Ωστόσο, ο χώρος που δεσμεύουν οι διαγραμμένες οντότητες στα αντίστοιχα αρχεία δε μειώνεται, μιας και ο χώρος αυτός απλά επισημαίνεται ως διαγραμμένος, αλλά δεν αποδίδεται στο λειτουργικό σύστημα. Σε επίπεδο ρυθμίσεων Neo4j, δεν υπάρχει καμία δυνατότητα ελέγχου αυτού του χώρου. Ωστόσο, το Neo4j παρέχει κάποια εργαλεία συμπύκνωσης αυτού του χώρου σε αυτόν που πραγματικά χρησιμοποιείται. Συγκεκριμένα, το εργαλείο διαχείρισης neo4j-admin παρέχει μια εντολή copy που επιτρέπει τη συμπυκνωμένη δημιουργία ενός καινούργιου αντιγράφου ασφαλείας της βάσης γράφου, χωρίς να λαμβάνει υπόψιν τις λογικά διαγραμμένες οντότητες. [50], [51].

#### 6.2.6.1.2.1 Παραμετροποίηση αρχείων καταγραφής δοσοληψιών

Το Neo4j αποθηκεύει κάθε εκτελεσμένη δοσοληψία που τροποποιεί την βάση δεδομένων γράφου σε ένα αρχείο καταγραφής (Transaction Log File). Τα αρχεία αυτά εξ' ορισμού αποθηκεύονται στον κατάλογο <neo4j-home>/data/transactions/<database-name>, όπου <neo4j-home> είναι ο κατάλογος εγκατάστασης του Neo4j και <database-name> το όνομα της βάσης δεδομένων. Ωστόσο, η τοποθεσία αποθήκευσης μπορεί να αλλάξει μέσω της ρύθμισης dbms.directories.transaction.logs.root στο αρχείο ρυθμίσεων του Neo4j/etc/neo4j/neo4j.conf. Τα αρχεία αυτά διαδραματίζουν πολύ σημαντικό ρόλο σε περιπτώσεις που απαιτηθεί επαναφορά της βάσης μετά από ένα απρόβλεπτο καταστροφικό γεγονός. Ωστόσο, σε περιπτώσεις που πραγματοποιούνται μαζικής κλίμακας δοσοληψίες τροποποίησης μιας βάσης δεδομένων, μπορεί να οδηγήσει στην αποθήκευση υπερβολικά μεγάλου αριθμού αρχείου δοσοληψιών που δεσμεύει πολύτιμο αποθηκευτικό χώρο στον δίσκο. Δεν είναι λίγες οι περιπτώσεις που το αρχείο δοσοληψιών μπορεί να δεσμεύει περισσότερο αποθηκευτικό χώρο από την ίδια τη βάση δεδομένων. Γενικά, μπορούν να εφαρμοστούν δυο πολιτικές διαχείρισης των αρχείων καταγραφής συναλλαγών: ελάχιστη και πλήρης. Η πρώτη μπορεί να εφαρμοστεί σε περιπτώσεις μιας μαζικής αρχικής εισαγωγής δεδομένων σε υπολογιστικά συστήματα, όπου παρατηρείται έλλειψη αποθηκευτικών πόρων. Η δεύτερη μπορεί να εφαρμοστεί σε περιπτώσεις παραγωγικής λειτουργίας, όπου οι αιτίες που μπορεί να οδηγήσουν σε αστοχία του συστήματος είναι πολλές και απρόβλεπτες. Σε μια τέτοια περίπτωση, το σύνολο των αρχείων καταγραφής αποτελεί τον μοναδικό τρόπο επαναφοράς της βάσης δεδομένων σε μια συνεπή κατάσταση πριν το καταστροφικό γεγονός. Προς αυτήν την κατεύθυνση, συνιστάται η χρήση μιας αφιερωμένης συσκευής αποθήκευσης ώστε να επιτύχουμε υψηλή απόδοση. Συγκεκριμένα, μέσω της ρύθμισης dbms.tx_log.rotation.retention_policy καθορίζεται ένα ορόσημο που, αν ξεπεραστεί, να κουρεύονται (Pruning) τα αποθηκευμένα αρχεία συναλλαγών (Retention Policy) τα οποία εμπίπτουν στην αποφασισμένη πολιτική περιστροφής

(Rotation Policy). Το ορόσημο αυτό μπορεί να καθοριστεί είτε ως πλήθος αρχείων, ημερών, ωρών, δοσοληψιών, μέγιστου μεγέθους στον δίσκου κ.α. Η εξ' ορισμού τιμή είναι 7 ημέρες. Στα πλαίσια της διπλωματικής επιλέχθηκε η παρακάτω ρύθμιση, ώστε να αποθηκεύονται μόνο λίγα αρχεία καταγραφής.

```
dbms.tx_log.rotation.retention_policy=10 files
```

Επίσης, μέσω της ρύθμισης dbms.tx_log.rotation.size καθορίζουμε άλλο ένα ορόσημο που είναι το συνολικό μέγεθος όλων των αρχείων δοσοληψιών στον κατάλογο αποθήκευσής τους. Σύμφωνα με αυτό τα παλιότερα αρχεία δοσοληψιών κουρεύονται, ώστε να μην ξεπεραστεί αυτό το όριο. Η εξ' ορισμού τιμή είναι 250 MB και αυτήν επιλέχθηκε στα πλαίσια της διπλωματικής.

```
dbms.tx_log.rotation.size=250 MB
```

Εξίσου σημαντικές είναι και οι ρυθμίσεις που καθορίζουν το πόσο συχνά θα πυροδοτείται ένα σημείο ελέγχου (Checkpoint), ώστε να εφαρμοστούν οι παραπάνω πολιτικές διατήρησης και περιστροφής. Ουσιαστικά, σε κάθε σημείο ελέγχου όλες οι εκκρεμείς δοσοληψίες της βοηθητικής μνήμης αποθηκεύονται στον κατάλογο αποθήκευσης των δοσοληψιών και στην συνέχεια κουρεύονται. Στην ελεύθερης διανομής έκδοση του Neo4j ένα σημείο ελέγχου πυροδοτείται περιοδικά είτε βάσει ενός συγκεκριμένου χρονικού διαστήματος ή βάσει του πλήθους δοσοληψιών, όποιο από τα δυο συμβεί πρώτο. Η εξ' ορισμού τιμή της ρύθμισης dbms.checkpoint.interval.time είναι 15m, ενώ της ρύθμισης dbms.checkpoint.interval.tx είναι 100000. Στα πλαίσια της διπλωματικής επιλέχθηκαν οι παρακάτω δυο ρυθμίσεις. Γενικά, ο συχνός καθορισμός σημείων ελέγχου μειώνει τον χρόνο επαναφοράς μιας βάσης δεδομένων, αλλά επιβαρύνει τις διαδικασίες εισόδου/ εξόδου του συστήματος.

```
dbms.checkpoint.interval.time=30m
dbms.checkpoint.interval.tx=10
```

#### 6.2.6.2 Βελτιστοποιήση ερωτημάτων Cypher

Υπάρχουν πολλές βελτιστοποιήσεις που μπορούμε να εφαρμοστούν, ώστε να επιταχυνθεί η ταχύτητα εκτέλεσης ενός ερωτήματος. Κάποιες από αυτές εκμεταλλεύονται την καλύτερη κατανόηση του τομέα προβλήματος για τον οποίο κατασκευάστηκε μια εφαρμογή. Κάποιες άλλες βασίζονται στη βαθύτερη γνώση του τρόπου εκτέλεσης ενός ερωτήματος από τη μηχανή εκτέλεσης του Neo4j.

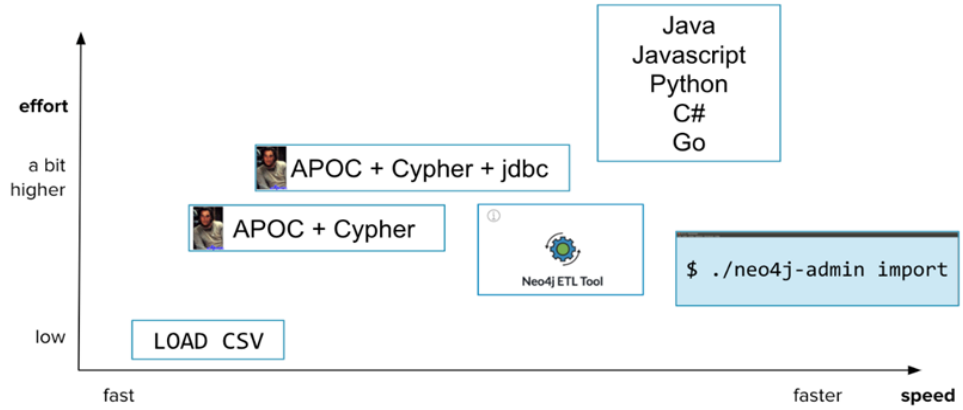
Καταρχήν, επιβάλλεται ο χρήστης να χρησιμοποιεί στις εφαρμογές του αποκλειστικά τις νεότερες εκδόσεις των οδηγών που υποστηρίζονται επίσημα από το Neo4j (Neo4j Officially Supported Drivers). Με αυτόν τον τρόπο, είναι σίγουρος ότι χρησιμοποιεί τα πλέον σύγχρονα, ασφαλή και αποδοτικά χαρακτηριστικά τους. Επίσης, παρέχεται ισχυρή εγγύηση διαλειτουργικότητας με τα υπόλοιπα προϊόντα Neo4j, αφού έχουν αναπτυχθεί από ομάδες με συμβατό τεχνολογικό υπόβαθρο. Ιδιαίτερη προσοχή πρέπει να επιδειχθεί στο σημείο αυτό, ώστε η έκδοση του οδηγού της γλώσσας προγραμματισμού στην οποία αναπτύσσεται η εφαρμογή να είναι συμβατή με την έκδοση και τον τύπο του προϊόντος Neo4j που επιθυμεί ο χρήστης να χρησιμοποιήσει. Οι οδηγοί που έχουν αναπτυχθεί από την κοινότητα των χρηστών (Neo4j Community Supported Drivers) χαρακτηρίζονται από χαμηλότερο βαθμό ωριμότητας, ασφάλειας και αποδοτικότητας.

Προκειμένου να επιταχυνθεί ακόμα περισσότερο η ταχύτητα εκτέλεσης ενός ερωτήματος, ενθαρρύνεται ο τελικός χρήστης να επαναδιατυπώνει έξυπνα τα ερωτήματα του. Αρχικά, επιβάλλεται οι χρήστες να κάνουν χρήση παραμέτρων και όχι κυριολεκτικών τιμών στα ερωτήματα που συντάσσουν. Με αυτόν τον τρόπο μπορεί το Neo4j να αποθηκεύει σε μια ενδιάμεση μνήμη ένα μόνο πλάνο εκτέλεσης για πολλαπλά παρόμοια ερωτήματα. Στην συνέχεια, μπορεί να το χρησιμοποιεί επαναλαμβανόμενα με διαφορετικές τιμές που δίνονται ως είσοδοι, χωρίς να χρειάζεται να το επαναυπολογήσει. Στη λογική της έξυπνης επαναδιατύπωσης των ερωτημάτων συνιστάται ακόμα οι χρήστες να δημιουργούν απλά ερωτήματα που δεν εκτελούν πολύπλοκες λειτουργίες και να επενεργούν στη βάση δεδομένων σε κομμάτια λογικού μεγέθους (Batch Reading, Writing, Deleting, Updating). Τέλος, πρέπει από την αρχική φάση της δημιουργίας του μοντέλου δεδομένων να το εμπλουτίζουν με ευρετήρια ως προς τις ιδιότητες στις οποίες θα βασιστούν τα κρίσιμης σημασίας ερωτήματα. Αυτό βέβαια προϋποθέτει μια πρώιμη γνώση του τύπου ερωτημάτων που πρόκειται να υποστηρίξει η εφαρμογή. Για αυτό συστήνεται η ανάπτυξη της εφαρμογής να βασιστεί σε μια ευέλικτη διαδικασία ανάπτυξης (Agile Development Process) που δίνει μεγάλη βαρύτητα στη συχνή και από τα πρώτα στάδια επικοινωνία της ομάδας ανάπτυξης με τους τελικούς χρήστες. Με αυτόν τον τρόπο, αποκαλύπτεται από τα πρώτα στάδια το ρεαλιστικό πρότυπο αλληλεπίδρασης των χρηστών με την εφαρμογή, οδηγώντας σε ποιοτικές εφαρμογές που ανταποκρίνονται στις ανάγκες τους. Η δημιουργία ενός ευρετηρίου μπορεί να γίνει και έμμεσα μέσω ενός περιορισμού. Με αυτόν τον τρόπο, και επιταχύνεται η ταχύτητα εκτέλεσης ενός ερωτήματος και παράλληλα προστατεύεται η ακεραιότητα των δεδομένων πριν την εισαγωγή τους στη βάση δεδομένων.

Τέλος, σημαντική βελτιστοποίηση επιτυγχάνεται απενεργοποιώντας τον επαναϋπολογισμό του πλάνου εκτέλεσης ενός ερωτήματος (Query Preplanning). Ως γνωστόν, για κάθε ερώτημα προς εκτέλεση το Neo4j δημιουργεί μια πληθώρα εναλλακτικών πλάνων εκτέλεσης. Στην συνέχεια, επιλέγει το βέλτιστο από αυτά λαμβάνοντας υπόψιν διάφορα στατιστικά στοιχεία της βάσης Neo4j. Σε περιβάλλον πραγματικής ανάπτυξης και λειτουργίας μιας εφαρμογής, τα στατιστικά αυτά αλλάζουν δυναμικά και ανάλογα με την αλληλεπίδραση των χρηστών. Είναι συχνό το φαινόμενο, ενώ ένα πλάνο εκτέλεσης ερωτήματος περιμένει να εκτελεστεί, τα στατιστικά στοιχεία στα οποία βασίστηκε η δημιουργία και επιλογή του να αλλάζουν με καταιγιστικούς ρυθμούς και να ξεπερνιούνται κάποιες τιμές ορόσημα. Αυτό πυροδοτεί τον αυτόματο επαναϋπολογισμό ενός νέου πλάνου, εισάγοντας σημαντική καθυστέρηση στον χρόνο εκτέλεσης του ερωτήματος. Ιδιαίτερα σε απαιτητικές ροές επεξεργασίας που περιλαμβάνουν μαζικές εγγραφές και ενημερώσεις σε πολύ μεγάλα τμήματα της βάσης δεδομένων γράφου δημιουργείται μια χιονοστιβάδα από επαναϋπολογισμούς πλάνων εκτέλεσης ερωτημάτων. Επιπρόσθετα, τέτοιοι συχνοί επαναϋπολογισμοί πυροδοτούν συχνά κύκλους πλήρους σάρωσης και συλλογής αχρησιμοποίητων αντικειμένων FGCC από τη μνήμη σωρού, εισάγοντας επιπλέον καθυστερήσεις στην εκτέλεση ενός ερωτήματος. Η απενεργοποίηση του επαναϋπολογισμού του πλάνου εκτέλεσης ενός ερωτήματος μπορεί να γίνει, είτε σε επίπεδο ερωτήματος Cypher, είτε καθολικά για όλα τα ερωτήματα μέσω κατάλληλων ρυθμίσεων στο αρχείο neo4j.conf [52].

### 6.2.7 7ο Στάδιο - Φόρτωση δεδομένων στη βάση δεδομένων γράφου

Υπάρχουν πολλές εναλλακτικές επιλογές όσον αφορά τη φόρτωση δεδομένων σε μια βάση δεδομένων γράφου ενός συστήματος διαχείρισης βάσεων Neo4j. Η τελική επιλογή βασίζεται στον όγκο των δεδομένων, τον διαθέσιμο χρόνο και την εξοικείωση του χρήστη με τα διάφορα εργαλεία. Οι διαθέσιμες επιλογές απεικονίζονται στο Σχήμα 6.5.




Σχήμα 6.5: Διαθέσιμες επιλογές εισαγωγής δεδομένων σε μια βάση δεδομένων γράφου (Πηγή [44])

Στα πλαίσια της διπλωματικής μας ενδιαφέρει η ταχύτατη, απομακρυσμένη εισαγωγή πολύ μεγάλων όγκων βιβλιογραφικών δεδομένων στη βάση δεδομένων γράφου. Αρχικά, επιχειρήθηκε η διαδικασία εισαγωγής των δεδομένων να γίνει μέσω ενός προγράμματος πελάτη (Client Application) σε γλώσσα προγραμματισμού Python που κάνει χρήση του αντίστοιχου οδηγού (Python Driver). Η προσέγγιση αυτή παρουσιάζει αρκετά πλεονεκτήματα, μιας και μας επιτρέπει να καθορίζουμε με μεγάλη ακρίβεια τα δεδομένα που θα εισαχθούν, τις προεπεξεργασίες που θα εκτελέσουμε σε αυτά, τον παραλληλισμό των επεξεργασιών (Parallel Processing) και το μέγεθος των κομματιών (Batch Processing) που θα γράφονται σε κάθε προσπέλση στην βάση. Στο σημείο αυτό αξίζει να αναφερθεί άλλο ένα σημαντικό πλεονέκτημα. Το API κάθε οδηγού είναι ανεξάρτητο της υποκείμενης τοπολογίας της βάσης δεδομένων (Topologically Agnostic). Συνεπώς, διάφορες τοπολογικές διαφορές απαιτούν ελάχιστες προγραμματιστικές αλλαγές (π.χ. μόνο στην μορφή του URI σύνδεσης με τη βάση δεδομένων). Για την εγκατάσταση επιλέχθηκε η τελευταία έκδοσή του οδηγού Python για αυξημένη σταθερότητα και υποστήριξη στην πλήρη γκάμα δυνατοτήτων. Τέλος αξίζει να αναφέρουμε ότι ο οδηγός Python δεν υποστηρίζει σύνδεση μέσω του πρωτοκόλλου HTTP, αλλά μόνο μέσω του πρωτοκόλλου bolt. Σε περίπτωση που ο χρήστης επιθυμεί αποκλειστικά σύνδεση HTTP, θα πρέπει να χρησιμοποιήσει τους υπόλοιπους ανεπίσημους οδηγούς που προσφέρει η κοινότητα των χρηστών (Community Supported Drivers). Ωστόσο, επισημαίνεται ότι οι τελευταίοι ποικίλουν σημαντικά σε ωριμότητα, χαρακτηριστικά και βαθμό υποστήριξης. Τελικά, στα πλαίσια της διπλωματικής δεν επιλέχθηκε η παραπάνω προσέγγιση κυρίως λόγω χαμηλής αποδοτικότητας εκτέλεσης στο μηχάνημα της σχολής. Αντίθετα, υιοθετήθηκε η προσέγγιση εισαγωγής των δεδομένων που κάνει χρήση της εντολής import του εργαλείου διαχείρισης neo4j-admin. Η εντολή αυτή επιτρέπει την προκαταρκτική ταχύτατη φόρτωση σε μια Neo4j βάση δεδομένων πολύ μεγάλων όγκων δεδομένων σε μορφή CSV. Η εντολή υποστηρίζει μόνο τη δημιουργία κόμβων και συσχετίσεων μαζί με τις ιδιότητες, τις ετικέτες και τους τύπους τους. Ουσιαστικά, η εντολή αυτή προετοιμάζει τα δεδομένα για φόρτωση σε μια βάση δεδομένων που δεν υφίσταται, αλλά θα δημιουργηθεί εκ των υστέρων μαζί με τα απαιτούμενα ευρετήρια και περιορισμούς. Σε περίπτωση που επιχειρηθεί εισαγωγή σε μια υφιστάμενη βάση δεδομένων που περιέχει περιεχόμενα, η εντολή επιστρέφει με κωδικό λάθους. Στα πλαίσια της διπλωματικής επιλέχθηκε η CE έκδοση του Neo4j Server, η οποία δεν υποστηρίζει την δημιουργία και διαχείριση πολλαπλών βάσεων δεδομένων (ενώ η Enterprise έκδοση την υποστηρίζει). Ο χρήστης επιτρέπεται να αλληλοεπιδράσει μόνο με την εξ' ορισμού βάση δεδομένων neo4j που δημιουργείται αυτόματα. Αυτό το γεγονός διαφοροποίησε σημαντικά την μεθοδολογία εργασίας. Συγκεκριμένα, απαιτήθηκε να γίνει η εκτέλεση της εντολής neo4j-admin import, έχοντας προηγουμένως κλείσει τον

εξυπηρετητή Neo4j. Επιπρόσθετα, καθορίστηκε η εισαγωγή να γίνει στη βάση δεδομένων neo4j.db, η οποία καθορίστηκε στο αρχείο ρυθμίσεων /etc/neo4j/neo4j.conf να αποτελεί την εξ΄ ορισμού βάση δεδομένων. Τέλος καθορίστηκε ο χρήστης neo4j ως ο ιδιοκτήτης της συγκεκριμένης βάσης δεδομένων. Όπως προκύπτει από το Σχήμα 6.5, η δεύτερη προσέγγιση είναι η πιο γρήγορη και απαιτεί μικρότερη προγραμματιστική προσπάθεια.

### 6.2.8 8ο Στάδιο - Ερωτήσεις Cypher για εξαγωγή συμπερασμάτων από βάση γράφου

Μετα τη φόρτωση των δεδομένων στη βάση δεδομένων γράφου επιβάλλεται η εκτέλεση ορισμένων ερωτημάτων Cypher. Τέτοιου είδους ερωτήματα αποκαλύπτουν πολύτιμη βιβλιογραφική γνώση, η οποία μπορεί να κάνει πιο αποτελεσματικές και αποδοτικές μελλοντικές ερευνητικές συνεργασίες. Συγκεκριμένα, μπορεί να επιταχύνει σημαντικά την επάνδρωση μελλοντικών ερευνητικών ομάδων με το κατάλληλο ερευνητικό και ακαδημαϊκό προσωπικό. Επίσης, μπορεί να διευκολύνει την ανακάλυψη προϋπάρχουσας επιστημονικής γνώσης, η οποία θα μπορούσε να αποτελέσει την κατάλληλη βάση για νέες σημαντικές ανακαλύψεις. Τέλος, αξίζει να σημειώσουμε ότι τέτοιου είδους ερωτήματα είναι πολύ σημαντικό να εκτελούνται και για λόγους εκσφαλμάτωσης της πληροφορίας που είναι ήδη αποθηκευμένη στη βάση δεδομένων γράφου. Στην συνέχεια, ακολουθούν ορισμένα χρήσιμα ερωτήματα καθώς και τα αποτελέσματα που παράγουν. Τα ερωτήματα συντάχθηκαν στη γλώσσα ερωτοαποκρίσεων Cypher του Neo4j και εκτελέστηκαν από τη γραμμή εντολών ενός webbrowser. Τα αποτελέσματα εμφανίζονται σε μορφή πίνακα, ενώ επίσης παρατίθενται και οι χρόνοι εκτέλεσης κάθε ερωτήματος. Τέλος υπενθυμίζουμε ότι στα πλαίσια της διπλωματικής φορτώθηκαν στη βάση δεδομένων γράφου αποκλειστικά πληροφορίες από το σύνολο δεδομένων Mag. Αυτό έγινε γιατί το συγκεκριμένο σύνολο δεδομένων περιέχει τον μεγαλύτερο όγκο βιβλιογραφικής πληροφορίας και επίσης περιέχει τα πιο ενδιαφέροντα πεδία.

Το παρακάτω ερώτημα εμφανίζει όλους τους συγγραφείς επιστημονικών εργασιών με όνομα «Antonis Sidiropoulos».

```
MATCH (a:Author {name: 'Antonis Sidiropoulos'}) RETURN a.id AS Author_Id,
a.name AS Author_Name
```

Στον πίνακα που ακολουθεί (Πίνακας 6.1) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε 5m:51s.

Πίνακας 6.1: Εμφανίζει όλους τους συγγραφείς επιστημονικών εργασιών με όνομα «Antonis Sidiropoulos»

| A/ A | Author_Id | Author_Name |
|---|---|---|
| 1 | 2167018829 | "Antonis Sidiropoulos" |
| 2 | 2183708182 | "Antonis Sidiropoulos" |
| 3 | 2785497204 | "Antonis Sidiropoulos" |

Από το παραπάνω ερώτημα ανακύπτει το συμπέρασμα ότι υπάρχουν τρείς διαφορετικοί συγγραφείς με το ίδιο όνομα "Antonis Sidiropoulos". Ουσιαστικά όμως πρόκειται για τον ίδιο συγγραφέα για τον οποίο έχουν εισαχθεί τρείς διαφορετικές εγγραφές στο σύνολο δεδομένων Mag. Τέτοιες καταστάσεις είναι συνήθεις, όταν τα δεδομένα συλλέγονται με αυτοματοποιημένες διαδικασίες από ετερογενείς πηγές δεδομένων. Στα πλαίσια της διπλωματικής δεν καταβλήθηκε καμία προσπάθεια απάλειψης πλεονασματικών εγγραφών οποιασδήποτε οντότητας του συνόλου δεδομένων. Στο σημείο αυτό αξίζει

επίσης να επισημανθεί ότι η εκτέλεση του παραπάνω ερωτήματος διήρκησε 5m:51s. Ο αυξημένος χρόνος εκτέλεσης που απαιτείται δικαιολογείται από το γεγονός ότι δεν έχει καθοριστεί κάποιο ευρετήριο ως προς το πεδίο Authors.name, στο οποίο βασίστηκε η αναζήτηση της παραπάνω εντολής Cypher. Αντίθετα αν εκτελέσουμε την έκδοση της εντολής Cypher που απεικονίζεται παρακάτω, τότε ο χρόνος εκτέλεσης συντομεύεται σημαντικά. Συγκεκριμένα, η εκτέλεση πλέον απαιτεί μόνο 2 ms. Αυτό οφείλεται στο γεγονός ότι αμέσως μετά τη φόρτωση των δεδομένων στη βάση γράφου δημιουργήσαμε έναν περιορισμό μοναδικότητας ως προς το πεδίο Authors.id. Ένας τέτοιος περιορισμός δημιουργεί αυτόματα και ένα ευρετήριο ως προς αυτό το πεδίο το οποίο επιταχύνει σημαντικά την ταχύτητα εκτέλεσης του ερωτήματος.

```
MATCH (a:Author) WHERE a.id IN [2785497204, 2183708182, 2167018829]
RETURN a.id AS Author_Id, a.name AS Author_Name
```

Το παρακάτω ερώτημα εμφανίζει μια λίστα με τους τίτλους επιστημονικών εργασιών, καθώς και το συνολικό τους πλήθος που δημοσίευσαν όλοι οι συγγραφείς με όνομα «Antonis Sidiropoulos».

```
MATCH (p:Paper) -[:HAS_AUTHOR]-> (a:Author) WHERE a.id in [2785497204,
2183708182, 2167018829] RETURN collect(p.name) AS Paper_Title_List, count(*)
AS Total_Published_Papers
```

Στον παρακάτω πίνακα (Πίνακας 6.2) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε μόλις 12 ms.

Πίνακας 6.2: Εμφανίζει όλες τις επιστημονικέςεργασίες που δημοσίευσε ο «Antonis Sidiropoulos»

| A/ A | Paper_Title_List | Total_Published_Papers |
|---|---|---|
| 1 | ["A human inspired handover policy using Gaussian Mixture Models and haptic cues", "Towards Progressive Automation of Repetitive Tasks Through Physical HumanRobot Interaction", "Humanrobot collaborative object transfer using human motion prediction based on Dynamic Movement Primitives", "A pHRI Framework for Modifying a Robots Kinematic Behaviour via Varying Stiffness and Dynamical System Synchronization", "SincBased Dynamic Movement Primitives for Encoding Pointtopoint Kinematic Behaviors", "A new perspective to automatically rank scientific conferences using digital libraries", "Rainbow ranking: an adaptable, multidimensional ranking method for publication sets", "A Survey of Link Analysis Ranking", "A citationbased system to assist prize awarding", "CDNs Content Outsourcing via Generalized Communities", "The Science of Science and a Multilayer Network Approach to Scientists Ranking", "A latencybased object placement approach in content distribution networks", "Discovery and Existence of Communities in the World Wide Web", "Gazing at the skyline for star scientists", "The fractal dimension of a citation curve: quantifying an individuals scientific output using the geometry of the entire curve", "A Scientists Impact over Time: The Predictive Power of Clustering with Peers", "Replication Based on Objects Load under a Content Distribution Network", "The Rainbow over the Greek Departments of Computer Science/Engineering: a Bibliometric Study", "Bibliometric indices for the assessment of the citation curve tail", "Prefetching in Content Distribution Networks via Web Communities Identification and Outsourcing", "Generalized Hirsch hindex for disclosing latent facts in citation networks", "CDNsim: A simulation tool for content distribution networks", "Ranking and identifying influential scientists versus mass producers by the Perfectionism Index", "Generalized comparison of graphbased ranking algorithms for publications and authors"] | 24 |

Το παρακάτω ερώτημα εμφανίζει για κάθε μια δημοσίευση του συγγραφέα «Antonis Sidiropoulos»μια λίστα με όλα τα id'στων επιστημονικών δημοσιεύσεων που κάνουν αναφορά σε αυτήν καθώς και το συνολικό τους πλήθος.

```
MATCH (p:Paper) -[:HAS_AUTHOR]-> (a:Author) WHERE a.id  in [2785497204,
2183708182, 2167018829] WITH p

MATCH  (p1:Paper) -[:CITES]-> (p) RETURN p.name AS Paper_Name_To_CITE,
collect(p1.id)      AS      Papers_CITES_Id_List,      count(*)      AS
Total_Paper_Citations
```

Στον παρακάτω πίνακα (Πίνακας 6.3) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε μόλις 8 ms.

Πίνακας 6.3: Εμφανίζει για κάθε μια δημοσίευση του συγγραφέα «Antonis Sidiropoulos» όλες τις επιστημονικές δημοσιεύσεις που κάνουν αναφορά σε αυτήν

| A/ A | Paper_Name_To_CITE | Papers_CITES_Id_List | Total_Paper_Citations |
|---|---|---|---|
| 1 | "A human inspired handover policy using Gaussian Mixture Models and haptic cues" | [3010746975] | 1 |
| 2 | "Towards Progressive Automation of Repetitive Tasks Through Physical HumanRobot Interaction" | [3026868555, 3006805849, 2980798007, 2968786418, 2965187964, 2893993964, 2888531196, 2884325870] | 8 |
| … | | | |
| 19 | "Ranking and identifying influential scientists versus mass producers by the Perfectionism Index" | [2488100808, 2484952566, 2439923854, 2291789472, 2134738500, 1802033262, 2910261844, 2800029236, 2607460842, 2588648404, 2520618487] | 11 |
| 20 | "Generalized comparison of graphbased ranking algorithms for publications and authors" | [2475781821, 2301233086, 2300984719, 2244694242, 2169859662, 2156307668, 2154347050, 2136192388, 2132519602, 2124301083, 2123387439, 2113052506, 2109298920, 2097027133, 2090710508, 2073863438, 2047035253, 2038282574, 2037246199, 2034845343, 2034577262, 2008960107, 2005375780, 2003680804, 1976570247, 1974605302, 1973629664, 1972830890, 1972039597, 1967611524, 1893145963, 1793882406, 3014970865, 2976209669, 2970802143, 2951508496, 2945583244, 2913000901, 2909200042, 2903145242, 2896529804, 2786610953, 2782575411, 2769051235, 2609596385, 2520143316, 1546285949, 1536660331, 1258377788] | 49 |

Από το αποτέλεσμα προκύπτει το συμπέρασμα ότι είκοσι (20) από τις είκοσι τέσσερις (24) συνολικά δημοσιεύσεις του συγγραφέα «Antonis Sidiropoulos» κατάφεραν να προσελκύσουν αναφορές από άλλες επιστημονικές εργασίες. Ωστόσο, δεν θα πρέπει να ξεχνάμε ότι η οντότητα Papers του συνόλου δεδομένων Mag φιλτραρίστηκε ως προς τα πεδία Papers.doi και Papers.type. Αυτό είχε ως αποτέλεσμα περίπου μόνο το 1/3 των συνολικών οντοτήτων τύπου Papersνα καταφέρουν να «περάσουν» το στάδιο του φιλτραρίσματος. Συνεπώς, δημιουργήθηκαν κόμβοι μόνο για αυτούς στη βάση δεδομένων γράφου. Εξάλλου, ας μην ξεχνούμε και την πάγια τακτική του Neo4j να μη δημιουργεί «σπασμένες» σχέσεις, δηλαδή σχέσεις όπου λείπει τουλάχιστον ο κόμβος σε ένα από τα δυο άκρα της. Αυτό είχε ως αποτέλεσμα να μην μπορέσουν να δημιουργηθούν και αποθηκευτούν στη βάση ένας πολύ μεγάλος αριθμός σχέσεων τύπου CITES. Αυτό μπορεί να επιβεβαιωθεί και από το αρχείο import.report που δημιουργείται αυτόματα μετα την εκτέλεση της εντολής neo4j-admin import. Συνεπώς, δεν μπορεί να αποκλειστεί το ενδεχόμενο και οι υπόλοιπες τέσσερις (4) δημοσιεύσεις του συγγραφέα «Antonis Sidiropoulos» να προσέλκυσαν σημαντικό αριθμό αναφορών, αλλά από επιστημονικές δημοσιεύσεις που δεν κατάφεραν τελικά να περάσουν επιτυχώς το στάδιο του φιλτραρίσματος. Επίσης, δεν μπορεί να αποκλειστεί και το ενδεχόμενο ότι και οι υπόλοιπες είκοσι (20) επιστημονικές δημοσιεύσεις, που εμφανίζονται ήδη στα αποτελέσματα, να κατάφεραν να προσελκύσουν επιπλέον αναφορές από επιστημονικές δημοσιεύσεις που επίσης δεν «πέρασαν» το στάδιο του φιλτραρίσματος.

Το παρακάτω ερώτημα εμφανίζει το όνομα και το id όλων των οργανισμών στους οποίους εργάστηκε ο «Antonis Sidiropoulos».

```
MATCH (a:Author) -[:IS_AFFILIATED_AT_ORGANIZATION]->(o:Organization) WHERE
a.id  in [2785497204, 2183708182, 2167018829] RETURN DISTINCT a.name AS
Author_Name, o.name AS Organization_Name, o.id AS Organization_Id
```

Στον πίνακα που ακολουθεί (Πίνακας 6.4) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε μόλις 2 ms.

Πίνακας 6.4: Εμφανίζει όλους τους οργανισμούς στους οποίους εργάστηκε ο «Antonis Sidiropoulos»

| A/ A | Author_Name | Organization_Name | Organization_Id |
|---|---|---|---|
| 1 | "Antonis Sidiropoulos" | "Aristotle University of Thessaloniki" | 21370196 |
| 2 | "Antonis Sidiropoulos" | "Alexander Technological Educational Institute of Thessaloniki" | 77990126 |

Το παρακάτω ερώτημα εμφανίζειτα ονόματα όλων των συγγραφέων και το συνολικό πλήθος δημοσιεύσεων που έχουν κάνει οι οποίοι κατά την περίοδο της δημοσίευσης κάποιας εργασίας τους εργάζονταν σε κάποιο από τους οργανισμούς στους οποίους έχει εργαστεί και ο «Antonis Sidiropoulos».

```
MATCH (a:Author) -[:IS_AFFILIATED_AT_ORGANIZATION]->(o:Organization) WHERE
a.id  in [2785497204, 2183708182, 2167018829] WITH DISTINCT o

MATCH (p:Paper) -[:HAS_AUTHOR]-> (b:Author) -
[:IS_AFFILIATED_AT_ORGANIZATION]->(o) RETURN DISTINCT b.name AS
Author_Name, o.name AS Organization_Name, count(DISTINCT p) AS
Total Papers Per Author
```

Στον παρακάτω πίνακα (Πίνακας 6.5) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε μόλις 52 ms.

Πίνακας 6.5: Εμφανίζει όλους τους συγγραφείς και τις συνολικές τους δημοσιεύσεις οι οποίοι εργάστηκαν σε κάποιο οργανισμό στον οποίο έχει εργαστεί και ο «Antonis Sidiropoulos»

| A/ A | Author_Name | Organization_Name | Total_Papers_Per_Author |
|---|---|---|---|
| 1 | "Panagiotis D. Bamidis" | "Aristotle University of Thessaloniki" | 322 |
| 2 | "Stathis Th. Konstantinidis" | "Aristotle University of Thessaloniki" | 44 |
| … | | | |
| 19610 | "Dimitris Ipsakis" | "Alexander Technological Educational Institute of Thessaloniki" | 1 |
| 19611 | "Konstantinos Vosniakos" | "Alexander Technological Educational Institute of Thessaloniki" | 2 |

Το παρακάτω ερώτημα εμφανίζειτο πλήθος όλων των συγγραφέων οι οποίοι δεν έχουν κάνει καμία δημοσίευση.

```
MATCH (a:Author)
WHERE NOT () -[:HAS_AUTHOR]-> (a)
RETURN count(DISTINCT a) AS Total_Authors_With_No_Publications
```

Στον πίνακα που ακολουθεί (Πίνακας 6.6) απεικονίζονται τα αποτελέσματα που επιστρέφονται σε μόλις 38m:31s.

Πίνακας 6.6: Εμφανίζει το πλήθοςτων συγγραφέωνοι οποίοι δεν έχουν κάνει καμία δημοσίευση

| A/ A | Total_Authors_With_No_Publications |
|---|---|
| 1 | 155992834 |

Φυσικά και στα αποτελέσματα του παραπάνω ερωτήματος θα πρέπει να λάβουμε ξανά υπόψιν το μειωμένο πλήθος οντοτήτων τύπου Papers οι οποίοι περνούν επιτυχημένα από τη φάση του φιλτραρίσματος και τελικά αποθηκεύεται ένας κόμβος για αυτούς στη βάση δεδομένων γράφου. Δεν αποκλείεται αρκετοί από αυτούς τους συγγραφείς να έχουν αρκετές δημοσιεύσεις για τις οποίες όμως δεν αποθηκεύτηκε στη βάση δεδομένων γράφου ένας κόμβος τύπου Paper, γιατί δεν πληρούσαν τις προδιαγραφές του σταδίου του φιλτραρίσματος.

### 6.2.9 9ο Στάδιο – Δημιουργία τελικού μοντέλου δεδομένων βάσης γράφου

Κατά την ενσωμάτωση των δεδομένων του συνόλου Aminer στο σύνολο Mag απαιτείται σημαντικής έκτασης αναδιοργάνωση της αρχικής έκδοσης του μοντέλου δεδομένων. Αυτή η ανάγκη επιβάλεται λόγω της ετερογένειας που παρουσιάζουν τα δύο σύνολα δεδομένων. Συγκεκριμένα, αναφέρεται η εμφάνιση διαφορετικών πεδίων μόνο σε ένα από τα δυο σύνολα, η ανάγκη συμπλήρωσης των

ελλειπουσών τιμών κάποιων πεδίων ενός συνόλου από τα αντίστοιχα πεδία του άλλου συνόλου. Άλλοι λόγοι που επιβάλουν την αναδιάταξη του μοντέλου είναι οι βέλτιστες πρακτικές μοντελοποίησης των κόμβων, συσχετίσεων και ιδιοτήτων, αλλά και η βελτίωση της προσβασιμότητας των δεδομένων κατά την εκτέλεση κρίσιμων ερωτημάτων από τους τελικούς χρήστες. Συγκεκριμένα, υλοποιήθηκαν οι ακόλουθοι μετασχηματισμοί:

Όσον αφορά τους κόμβους προστέθηκαν στο μοντέλο δεδομένων οι εξής.

1. Keyword: αναπαριστά τις οντότητες των λέξεων κλειδιών που περιγράφουν τις θεματικές περιοχές που καλύπτουν οι επιστημονικές εργασίες/ άρθρα.
2. ResearchArea: αναπαριστά τις οντότητες των επιστημονικών περιοχών ενδιαφέροντος των συγγραφέων.
3. Position: αναπαριστά τις οντότητες των επιστημονικών θέσεων που καταλαμβάνουν οι συγγραφείς στους οργανισμούς απασχόλησής τους.
4. Affiliation: δεν αναπαριστά πραγματικές οντότητες του τομέα προβλήματος. Ουσιαστικά, πρόκειται για τεχνικούς ενδιάμεσους κόμβους που χρησιμοποιήθηκαν για να υλοποιήσουν τις τριαδικές συσχετίσεις μεταξύ των οντοτήτων των συγγραφέων, οργανισμών εργασίας τους και των θέσεων που καταλαμβάνουν σε αυτές. Αυτή η τεχνικής φύσης αναδιάταξη αποτελεί παραβίαση μιας βασικής αρχής του μοντέλου ιδιοτήτων γράφου του Neo4j που απαγορεύει την αποθήκευση μη πραγματικών δεδομένων. Ωστόσο ήταν απαραίτητη, μιας και το μοντέλο του Neo4j υποστηρίζει μόνο δυαδικές σχέσεις μεταξύ των κόμβων.

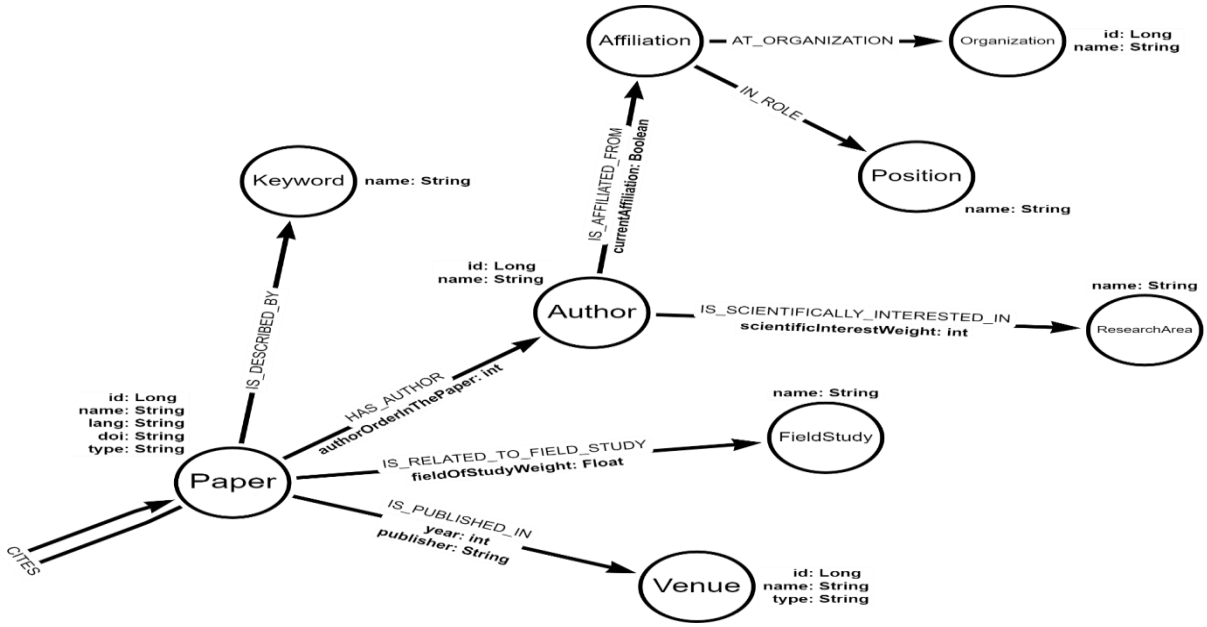
Όσον αφορά τις σχέσεις προστέθηκαν στο μοντέλο δεδομένων οι εξής:

1. IS_DESCRIBED_BY: αναπαριστά τη συσχέτιση επιστημονικών εργασιών και λέξεων κλειδιών που τις περιγράφουν.
2. IS_SCIENTIFICALLY_INTERSTED_IN: αναπαριστά τη συσχέτιση συγγραφέων και επιστημονικών περιοχών ενδιαφέροντος.
3. IN_ROLE: αναπαριστά τη συσχέτιση απασχολήσεων και επιστημονικών θέσεων που καταλαμβάνουν οι συγγραφείς στους οργανισμούς απασχόλησης.
4. IS_AFFILIATED_FROM&AT_ORGANIZATION: προέκυψαν από τη διάσπαση της σχέσης IS_AFFILIATED_AT_ORGANIZATION του αρχικού μοντέλου δεδομένων. Είναι απαραίτητες για την υλοποίηση της τριαδικής συσχέτισης μεταξύ των συγγραφέων, οργανισμών εργασίας τους και των θέσεων που καταλαμβάνουν σε αυτές.

Στη συνέχεια ακολουθούν οι κόμβοι και οι μεταξύ τους σχέσεις σε μορφή AsciiArt.

```
(:Paper) - [:IS_DESCRIBED_BY] -> (:Keyword)

(:Author) - [:IS_SCIENTIFICALLY_INTERSTED_IN] -> (:ResearchArea)

(:Affiliation) - [:IN_ROLE] -> (:Position)

(:Author) - [:IS_AFFILIATED_FROM] -> (:Affiliation)

(:Affiliation) - [:AT_ORGANIZATION] -> (:Organization)
```

Πλέον το τελικό μοντέλο δεδομένων αποκτά την μορφή που απεικονίζεται στο Σχήμα 6.6.




Σχήμα 6.6: Τελικό μοντέλο βάσης δεδομένων γράφου

Το στάδιο αυτό δεν υλοποιήθηκε λόγω έλλειψης χρόνου και της απόφασής να εισαχθούν στη βάση δεδομένων γράφου αποκλειστικά δεδομένα του συνόλου δεδομένων Mag.

## 6.3 Επίλογος

Στο κεφάλαιο αυτό σκιαγραφήθηκε η μεθοδολογία εργασίας πολλών βημάτων που καθοδήγησε την ερευνητική μας προσπάθεια. Αρχικά, διερευνήθηκαν αναλυτικά τα σύνολα δεδομένων μας, ώστε να εκτιμηθεί η ποιότητά τους. Στο βήμα αυτό επισημάνθηκαν τα προβλήματα που εντοπίστηκαν και προτάθηκαν διάφοροι τρόποι αντιμετώπισής τους. Στο επόμενο βήμα επιχειρήθηκε μια προεπεξεργασία των συνόλων δεδομένων, ώστε να συμπληρωθούν οι τυχόν ελλείπουσες τιμές των οντοτήτων από αποθηκευμένη πλεονασματική πληροφορία. Η προσπάθεια επιχειρήθηκε μέσω διαδικασιών εσωτερικής και εξωτερικής αλληλοσυσχέτισης με τα υπόλοιπα αρχεία των συνόλων δεδομένων. Ωστόσο εγκαταλείφθηκε λόγω έλλειψης χρόνου. Στα επόμενα δύο βήματα τα δεδομένα φιλτραρίστηκαν σύμφωνα με τις προδιαγραφές της εργασίας και καθαρίστηκαν, ώστε να βελτιωθεί η ποιότητά τους. Το επόμενο βήμα ήταν πολύ σημαντικό, μιας και σε αυτό αποκρυσταλλώθηκαν οι οντότητες και τα χαρακτηριστικά τους που είναι σημαντικά για την υποδομή της διπλωματικής. Αυτά οδήγησαν στη δημιουργία μιας αρχικής έκδοσης ενός μοντέλου δεδομένων. Το επόμενο βήμα ήταν εξίσου σημαντικό με το προηγούμενο. Σε αυτό προτάθηκαν διάφορες παραμετροποιήσεις – βελτιστοποιήσεις που θα αυξήσουν την απόδοση του Neo4j. Οι προτεινόμενες παραμετροποιήσεις αφορούσαν τη μνήμη (κύρια και δευτερεύουσα) καθώς και του τρόπο εκτέλεσης των ερωτημάτων Cypher. Στο επόμενο βήμα φορτώθηκαν τα βιβλιογραφικά δεδομένα στη βάση δεδομένων γράφου. Ο παρών ερευνητής κατέληξε σε δυο τεχνικές. Η μια έκανε χρήση του εργαλείου διαχείρισης neo4j-admin import και η άλλη μέσω μιας εφαρμογής πελάτη σε Python που έκανε χρήση του οδηγού Python. Τελικά, επιλέχθηκε η τεχνική που κάνει χρήση της εντολής neo4j-admin import ως η πλέον αποδοτική. Στην συνέχεια, εκτελέστηκαν διάφορες ερωτήσεις Cypher που αποσκοπούν στην αποκάλυψη χρήσιμης γνώσης που είναι αποθηκευμένη στη βάση δεδομένων γράφου. Στο τελικό βήμα, επιχειρήθηκε η αναδιάταξη του μοντέλου δεδομένων, ώστε να ενσωματώσει περισσότερη βιβλιογραφική πληροφορία. Το βήμα επιβλήθηκε, προκειμένου να ανταποκρίνεται με μεγαλύτερη ακρίβεια στις μεταβαλλόμενες ανάγκες των χρηστών και στην πιο αποδοτική εκτέλεση των

ερωτημάτων Cypher. Το βήμα αυτό δεν υλοποιήθηκε. Στο σημείο αυτό κρίθηκε απαραίτητο να καθοριστούν οι τεχνολογίες και τα προγραμματιστικά περιβάλλοντα που θα βοηθούσαν να υλοποιηθούν οι σχεδιαστικές και αρχιτεκτονικές αποφάσεις των προηγούμενων ενοτήτων. Οι τεχνολογίες που επιλέχθηκαν περιγράφονται στο επόμενο κεφάλαιο.

# Κεφάλαιο 7ο: Τεχνολογίες Υλοποίησης - Προγραμματιστικά Περιβάλλοντα

## 7.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφονται οι τεχνολογίες που κατέστησαν δυνατή την υλοποίηση της βιβλιογραφικής υποδομής της διπλωματικής εργασίας. Για κάθε μια από αυτές παρατίθεται μια συνοπτική περιγραφή των σημαντικότερων δυνατοτήτων της και επεξηγείται το κομμάτι της υποδομής που υλοποιεί.

## 7.2 EmEditor professional

Πρόκειται για έναν εύχρηστο, γρήγορο, επεκτάσιμο επεξεργαστή κειμένου (Plain Text, Json, Csv) που τρέχει σε λειτουργικό σύστημα Windows. Διατίθεται σε εκδόσεις 32 bit και 64 bit, με τη δεύτερη να είναι πιο γρήγορη και να μπορεί να ανοίξει πολύ μεγαλύτερα αρχεία. Συγκεκριμένα, η δεύτερη έκδοση μπορεί να ανοίξει αρχεία έως 16 TB ή 1,099 δισεκατομμύρια γραμμές, ανάλογα με το ποιο όριο θα επιτευχθεί πρώτο. Αυτό το επιτυγχάνει μέσω μιας ρύθμισης που καθορίζει το άνω όριο του αρχείου που επιθυμεί ο χρήστης να ανοίξει. Όταν κατά το άνοιγμα του αρχείου αυτό το όριο ξεπεραστεί, τότε ο επεξεργαστής σταματά να αποθηκεύει στη μνήμη RAM τα επιπλέον κομμάτια του αρχείου και τα αποθηκεύει σε προσωρινά αρχεία στον σκληρό δίσκο. Για αρχεία ακόμα μεγαλύτερα από τα παραπάνω ήδη υψηλά όρια, ο χρήστης μπορεί να χρησιμοποιήσει τον ελεγκτή μεγάλων αρχείων (Large File Controller). Μέσω αυτού μπορεί να ελέγξει πλήρως τη διαδικασία ανοίγματος του αρχείου. Συγκεκριμένα, μπορεί να παρακολουθεί το συνολικό μέγεθος του αρχείου που προσπαθεί να ανοίξει καθώς και το τρέχον ποσοστό που έχει ήδη ανοίξει. Τέλος, μπορεί να σταματήσει ανά πάσα στιγμή τη διαδικασία και να καθορίσει εκ νέου ένα διαφορετικό κομμάτι ενός τεράστιου αρχείου που επιθυμεί να ανοίξει. Αφού ο χρήστης εκτελέσει σε κάθε κομμάτι τις απαιτούμενες επεξεργασίες, το αρχείο κειμένου αποθηκεύεται στον σκληρό δίσκο ως ένα ενιαίο αρχείο. Ο χρήστης έχει τη δυνατότητα να διασπάσει ένα αρχείο σε πολλαπλά άλλα αρχεία, βάσει είτε ενός συγκεκριμένου αριθμού γραμμών ή ενός σελιδοδείκτη. Υποστηρίζεται και η αντίστροφη διαδικασία, δηλαδή η συνένωση πολλαπλών διαφορετικών αρχείων σε ένα ενιαίο αρχείο. Και οι δυο παραπάνω διαδικασίες είναι πλήρως ρυθμιζόμενες από τον χρήστη, μέσω ενός εύχρηστου οδηγού. Ο EmEditor είναι σχεδιασμένος να λειτουργεί σε πολλαπλά νήματα εκτέλεσης μιας μόνο διεργασίας (Single-Process, Multithreaded Design). Αυτό του επιτρέπει να ανοίγει γρήγορα και ταυτόχρονα πολλαπλά αρχεία κειμένου και να εκτελεί πολλαπλές επεξεργασίες σε αυτά με το μικρότερο αποτύπωμα στους πόρους του υπολογιστή. Παρέχει μια μεγάλη γκάμα ταχύτατων λειτουργιών που καλύπτουν και τον απαιτητικό χρήστη. Συγκεκριμένα, παρέχει πανίσχυρες λειτουργίες ταξινόμησης, αναζήτησης, αντικατάστασης, φιλτραρίσματος, κατακόρυφης και οριζόντιας επιλογής, εντοπισμού/ διαγραφής διπλότυπων γραμμών/ στηλών κ.α. Επιπλέον υποστηρίζει προχωρημένες λειτουργίες αυξητικής αναζήτησης, αντικατάστασης και φιλτραρίσματος πάνω στα αποτελέσματα προηγούμενων λειτουργιών. Με αυτόν τον τρόπο επιτυγχάνεται ακόμα πιο λεπτομερή μελέτη και του πιο μικρού υποσυνόλου δεδομένων. Και όλα αυτά μέσω ενός εύχρηστου, πλήρως προσαρμόσιμου γραφικού περιβάλλοντος, που βελτιώνει την εμπειρία του χρήστη. Ο EmEditor υποστηρίζει ακόμα πιο πολύπλοκες λειτουργικότητες που δεν είναι εξ' ορισμού ενσωματωμένες στην εφαρμογή. Αυτό το επιτυγχάνει μέσω των μηχανισμών επέκτασης των πρόσθετων (Plug-ins), των μακροεντολών (Macros) και της εγκατάστασης εξωτερικών εργαλείων (External Tools). Επίσης, υποστηρίζει πλήρως το σύνολο χαρακτήρων Unicode, επιτρέποντας το άνοιγμα και επεξεργασία πολυγλωσσικών

κειμένων και διευκολύνει την αλλαγή της κωδικοποίησης πολλαπλών αρχείων. Τέλος, ο επεξεργαστής αποτελεί πολύτιμο εργαλείο και για τον προγραμματιστή, μιας και επιτρέπει τη ρύθμιση του περιβάλλοντος εργασίας για πάνω από 20 γλώσσες προγραμματισμού [53].

## 7.3 Neo4j arrows.app

Πρόκειται για μια διαδικτυακή εφαρμογή (Web Application) που χρησιμοποιείται για τη σχεδίαση του μοντέλου μιας βάσης δεδομένων γράφου. Είναι εύχρηστη και παρέχει βασικές λειτουργίες σχεδίασης και μορφοποίησης όλων των δομικών στοιχείων που απαρτίζουν το μοντέλο δεδομένων ενός γράφου (Nodes, Relationships, Labels, Properties). Επιπλέον, δίνει τη δυνατότητα εξαγωγής και αποθήκευσης της σχεδίασης σε διάφορες μορφές. Σε αυτές συγκαταλέγονται εικόνες διαφόρων τύπων και αναλύσεων (SVG, PNG), ως κώδικας Cypher, σε μορφή Json και ως σχήμα GraphQL. Επίσης, διευκολύνει τον διαμοιρασμό της σχεδίασης με άλλα μέλη της ομάδας του έργου μέσω ενός URL. Η εφαρμογή αυτή χρησιμοποιήθηκε για τον σχεδιασμό του σχεδιαγράμματος του μοντέλου δεδομένων της βάσης δεδομένων γράφου της εφαρμογής. Είναι προσπελάσιμο μέσω του συνδέσμου https://arrows.app/.

## 7.4 Εικονικά μηχανήματα

Αποτελούν τμήματα μιας μεγαλύτερης φυσικής υπολογιστικής υποδομής, στην οποία εκτελούνται ανεξάρτητα. Βασίζονται στην αρχή της εικονοποίησης (Virtualization). Σύμφωνα με αυτήν, διαφορετικά τμήματα λογισμικού μπορούν να προσομοιώσουν λειτουργικότητες κομματιών υλικού και να αποτελέσουν τα δομικά στοιχεία τα οποία μπορούν να συνθέσουν πιο ισχυρά εικονικά μηχανήματα. Αυτά συντίθενται από διαφορετικά κομμάτια υλικού, λειτουργικά συστήματα και ρυθμίσεις, καλύπτοντας πλήρως τις διαφοροποιημένες ανάγκες μεγάλης δεξαμενής χρηστών. Και όλα αυτά με τρόπο διάφανο προς τον τελικό χρήστη, ο οποίος νομίζει ότι επενεργεί στο δικό του αφιερωμένο φυσικό μηχάνημα. Το μεγαλύτερο πλεονέκτημα που προσφέρουν τα εικονικά μηχανήματα είναι η παροχή μηχανημάτων πλήρως προσαρμοσμένων στις ανάγκες του κάθε χρήστη. Με αυτόν τον τρόπο επιτυγχάνονται τεράστιες οικονομίες κλίμακας και αύξηση της αποδοτικότητας. Άλλο ένα σημαντικό πλεονέκτημα που προσφέρουν είναι η ευελιξία χρήσης του μηχανήματος. Κάθε μηχάνημα που δημιουργείται χρησιμοποιείται μόνο για όσο το επιθυμεί ο χρήστης. Όταν πλέον δεν χρειάζεται, διαγράφεται, απελευθερώνοντας τους πόρους που κατείχε στην υποκείμενη υπολογιστική υποδομή.

### 7.4.1 Δωρεάν παραχωρημένα

#### 7.4.1.1 Google colab

Πρόκειται για ένα δωρεάν περιβάλλον εκτέλεσης κώδικα Python που τρέχει μέσα από την υποδομή υπολογιστικού νέφους της Google. Επιτρέπει στον χρήστη να δημιουργεί, εκτελεί και διαμοιράζει Jupiter Notebooks με μηδενική αρχική παραμετροποίηση. Κάθε notebook ουσιαστικά αποτελεί μια λίστα από κελιά δυο τύπων: κελιά κώδικα, κελιά κειμένου. Τα πρώτα περιέχουν κώδικα Python που μπορεί να εκτελεστεί και να δημιουργήσει αποτελέσματα. Τα δεύτερα περιλαμβάνουν πλούσιο επεξηγηματικό κείμενο πάνω στον κώδικα, όπως εικόνες, HTML μορφοποιημένο κείμενο, Latex κ.α. Κάθε ένα από αυτά τα notebooks αποθηκεύεται στον λογαριασμό Google Drive και μέσω ενός συνδέσμου μπορούν να διαμοιρασθούν σε οποιονδήποτε ενδιαφερόμενο. Το κάθε notebook τρέχει σε υλικό του υπολογιστικού νέφους της Google, που είναι γεωγραφικά απομακρυσμένο. Όλο αυτό γίνεται με τρόπο διάφανο προς τον τελικό χρήστη, ο οποίος με τους τοπικούς πόρους του

διαχειρίζεται μόνο τα αποτελέσματα που επιστρέφονται. Κάθε υπολογιστικό μηχάνημα περιλαμβάνει επαρκείς αποθηκευτικούς και υπολογιστικούς πόρους καθώς και δυνατότητα χρήσης GPU και TPU για ακόμα μεγαλύτερη επιτάχυνση του χρόνου εκτέλεσης του κώδικα. Οι υπολογιστικές ικανότητες ενός μηχανήματος Google Colab ουσιαστικά είναι παρόμοιες με αυτές ενός μέσου σημερινού φορητού υπολογιστή. Υπάρχουν χρονικοί περιορισμοί στη χρήση των πόρων κάθε εικονικού μηχανήματος, για να αποφευχθούν καταστάσεις κατάχρησης τους και διακοπής της ομαλής διαθεσιμότητας της υπηρεσίας. Η υπηρεσία είναι προσπελάσιμη από τη διεύθυνση https://colab.research.google.com/, ενώ για τη σύνδεση απαιτείται η ύπαρξη ενεργού λογαριασμού ηλεκτρονικού ταχυδρομείου Gmail.

#### 7.4.1.2 Kaggle

Πρόκειται για μια δημοφιλή πλατφόρμα που φιλοξενεί διαφόρων τύπων διαγωνισμούς στον τομέα της μηχανικής μάθησης. Σε αυτούς συγκαταλέγονται διαγωνισμοί εμπορικοί (Featured Competitions), ερευνητικοί (Research Competitions), προσλήψεων (Recruitment Competitions) κ.α. Η πρώτη κατηγορία περιλαμβάνει την επίλυση δύσκολων προκλήσεων που παρουσιάζονται στον εμπορικό τομέα. Οι συμμετέχοντες απαιτείται να διαθέτουν υψηλό επίπεδο τεχνικών γνώσεων και συνήθως παρέχεται στον νικητή κάποιο χρηματικό έπαθλο. Η δεύτερη κατηγορία περιλαμβάνει την επίλυση πειραματικής φύσεως προβλημάτων που παρουσιάζονται σε συγκεκριμένους ερευνητικούς τομείς. Συνήθως αυτά τα προβλήματα απαιτούν καινοτόμες λύσεις, οι συμμετέχοντες πρέπει να διαθέτουν πολύ υψηλό επίπεδο τεχνικών γνώσεων, ενώ δεν προσφέρεται κάποιο έπαθλο. Η τρίτη κατηγορία περιλαμβάνει την επίλυση κάποιων προκλήσεων που θέτουν συγκεκριμένες εταιρείες και συνήθως ο νικητής ανταμείβεται με μια συνέντευξη από την οποία πιθανότατα θα προκύψει μια ευκαιρία εργασίας. Η πλατφόρμα παρέχει δωρεάν στους συμμετέχοντες των διαγωνισμών ένα εικονικό μηχάνημα (Virtual Machine) στο οποίο τρέχει κώδικας Python σε περιβάλλον Jupiter notebook χωρίς καμία αρχική παραμετροποίηση. Η λειτουργικότητα και εμφάνισή του είναι παρόμοια με αυτή του Google Colab. Κάθε μηχάνημα είναι εξοπλισμένο με σημαντικούς αποθηκευτικούς και υπολογιστικούς πόρους. Ουσιαστικά, οι υπολογιστικές ικανότητες ενός μηχανήματος Kaggle είναι παρόμοιες με αυτές ενός μέσου σημερινού φορητού υπολογιστή με μόνη διαφορά ότι τρέχει στην υποδομή υπολογιστικού νέφους της Google. Συνεπώς, οποιαδήποτε τοπική διακοπή της διαθεσιμότητας του διαδικτύου δεν διακόπτει την εκτέλεση του κώδικά στο εικονικό μηχάνημα. Προκαλεί μόνο προσωρινή διακοπή της εμφάνισης των παραγόμενων αποτελεσμάτων στον τοπικό υπολογιστή. Όμως, επειδή οι πόροι που παρέχονται είναι περιορισμένοι και για να αποφευχθεί η κατάχρηση τους, έχουν οριστεί λογικά όρια χρήσης. Αυτά περιορίζουν τον μέγιστο χρόνο συνεχόμενης εκτέλεσης ενός πειράματος, τον μέγιστο χρόνο συνεχόμενης αδράνειας και τον μέγιστο χρόνο χρήσης της GPU/ TPU. Η υπηρεσία είναι προσπελάσιμη από τη διεύθυνση https://www.kaggle.com/, ενώ για τη σύνδεση απαιτείται η ύπαρξη ενεργού λογαριασμού στην πλατφόρμα.

#### 7.4.1.3 Σύγκριση

Ουσιαστικά τα δυο εικονικά μηχανήματα (Virtual Machines) είναι λειτουργικά ισοδύναμα, μιας και τα δυο επιτρέπουν τη δωρεάν εκτέλεση Python κώδικα σε περιβάλλον Jupiter Notebook, χωρίς να απαιτούν κάποια αρχική παραμετροποίηση. Επίσης, και τα δυο χρησιμοποιούν τις υποδομές υπολογιστικού νέφους GCP. Επιπρόσθετα, η πλατφόρμα Kaggle επιτρέπει άμεσα την εκτέλεση και R κώδικα, ενώ στο Google Colab απαιτείται από τον χρήστη κάποια επιπλέον παραμετροποίηση. Όπως προκύπτει από τον πίνακα που ακολουθεί (Πίνακας 7.1), το εικονικό μηχάνημα της Kaggle είναι υπολογιστικά πιο ισχυρό. Συγκεκριμένα, προσφέρει διπλάσιες CPU στο τυπικό περιβάλλον εκτέλεσης

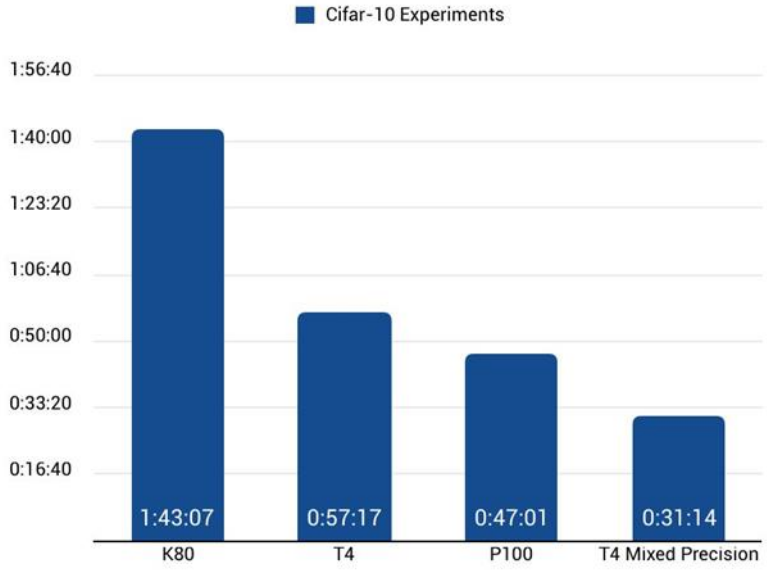
που δεν κάνει χρήση της GPU. Από τον παρακάτω πίνακα (Πίνακας 7.1) προκύπτει ότι διαφορά υπάρχει και ως προς την αρχιτεκτονική της παρεχόμενης GPU. Συγκεκριμένα, το Google Colab παρέχει μια GPU απαρχαιωμένης αρχιτεκτονικής Kepler (Nvidia Tesla K80), ενώ το Kaggle παρέχει μια GPU νεότερης αρχιτεκτονικής Pascal (Nvidia Tesla P100). Όπως προκύπτει (Σχήμα 7.1), η Nvidia Tesla P100 είναι πιο ισχυρή από τη Nvidia Tesla K80, μιας και μπορεί να εκτελεί περισσότερους από διπλάσιους υπολογισμούς κινητής υποδιαστολής το δευτερόλεπτο TFLOPS. Επίσης, λόγω της πιο απαρχαιωμένης αρχιτεκτονικής δεν την υποστηρίζουν οι νεότερες βιβλιοθήκες, που επιτρέπουν την επιταχυνόμενη εκτέλεση κώδικα σε GPU. Ως προς την μόνιμη αποθηκευτική ικανότητα που προσφέρεται, δεν υπάρχουν σημαντικές διαφορές, με εξαίρεση τη μνήμη RAM, που στο μηχάνημα της Kaggle είναι ελαφρά περισσότερη. Όταν αλλάζει το περιβάλλον εκτέλεσης, ώστε να κάνει χρήση της GPU, και τα δυο μηχανήματα περιορίζουν τη διαθέσιμη μνήμη και τον σκληρό δίσκο. Η πλατφόρμα Kaggle υποδιπλασιάζει επιπρόσθετα και τις CPU's που διαθέτει. Ωστόσο το Google Colab πλεονεκτεί στο ότι συνεργάζεται πλήρως με το GoogleDrive. Συνεπώς, ο χρήστης μπορεί σχετικά φθηνά να αγοράσει επιπρόσθετο χώρο αποθήκευσης στο Google Drive και να το προσαρτήσει στο περιβάλλον Google Colab, χωρίς ο επιπλέον χώρος να μετράει στα δωρεάν όρια χρήσης που του προσφέρει η πλατφόρμα. Αντίθετα, το Kaggle δεν υποστηρίζει αυτό το προϊόν, αλλά παρέχει υποστήριξη στο GCS. Διαφορές όμως υπάρχουν και στα χρονικά όρια συνεχόμενης χρήσης, με το Google Colab να υπερτερεί, όπως προκύπτει από τον παρακάτω πίνακα (Πίνακας 7.3). Ωστόσο η πλατφόρμα Kaggle παρέχει κάποιες επιπλέον χρήσιμες λειτουργικότητες. Συγκεκριμένα, αυτοματοποιεί τη μελλοντική δρομολόγηση εκτέλεσης ενός notebook στο παρασκήνιο και αποθήκευση των αποτελεσμάτων, επιτρέπει επιλογή της εικόνας παραμετροποιήσεων Docker (Docker Image) επάνω στο οποίο θα βασιστεί το κάθε Notebook και επιτρέπει τη σύνδεση με επιπλέον υπηρεσίες Google (BigQuery, Cloud Storage, Cloud AI Platform). Τέλος, διαφορές υπάρχουν και στη δυνατότητα αναβάθμισης σε κάποια πληρωμένη έκδοση της υπηρεσίας, που παρέχει πρόσβαση σε περισσότερους και καλύτερους πόρους για ακόμα μεγαλύτερη επιτάχυνση της εκτέλεσης των πειραμάτων. Και οι δυο αναβαθμισμένες εκδόσεις της υπηρεσίας Google Colab (Colab Pro, Colab Pro+) είναι διαθέσιμες σε περιορισμένο αριθμό χωρών (USA, Canada, Japan, Brazil, Germany, France, India, United Kingdom, Thailand). Αντίθετα, στην πλατφόρμα Kaggle δεν υπάρχουν γεωγραφικοί περιορισμοί στο πλάνο αναβάθμισης που παρέχεται.

Πίνακας 7.1: Σύγκριση προδιαγραφών CPU Google Colab και Kaggle ( Πηγή [54])

| CPU-only VMs | | |
|---|---|---|
| Παράμετρος | Google Colab | Kaggle |
| Μοντέλο CPU | Intel(R) Xeon(R) | Intel(R) Xeon(R) |
| Συχνότητα CPU | 2.30 GHz | 2.30 GHz |
| Αριθμός πυρήνων CPU | 2 | 4 |
| Οικογένεια CPU | Haswell | Haswell |
| Διαθέσιμη RAM | 12 GB | 16 GB |
| Διαθέσιμος σκληρός δίσκος | 60 GB | 70 GB |

Πίνακας 7.2: Σύγκριση προδιαγραφών GPU Google Colab και Kaggle ( Πηγή [54])

| GPU only VMs | | |
|---|---|---|
| Παράμετρος | Google Colab | Kaggle |
| GPU | Nvidia K80/ T4 | Nvidia P100 |
| Μνήμη GPU | 12GB/ 16GB | 16GB |
| Συχνότητα GPU | 0.82GHz/ 1.59GHz | 1.32GHz |
| Απόδοση | 4.1 TFLOPS / 8.1 TFLOPS | 9.3 TFLOPS |
| Υποστήριξη Μικτής Ακρίβειας | Όχι/ Ναι | Όχι |
| Έτος κυκλοφορίας GPU | 2014/ 2018 | 2016 |
| Πλήθος πυρήνων CPU | 2 | 2 |
| Διαθέσιμη RAM | 12GB (αναβαθμίσιμηέως 26.75GB) | 12GB |
| Σκληρός δίσκος | 358GB | 5GB |




Σχήμα 7.1: Σύγκριση απόδοσης μεταξύ των υποστηριζόμενων GPU's από Google Colab και Kaggle ( Πηγή [54])

Πίνακας 7.3: Χρονικά όρια εκτέλεσης Google Colab και Kaggle ( Πηγή [54])

| Χρονικά Όρια Εκτέλεσης | | |
|---|---|---|
| Παράμετρος | Google Colab | Kaggle |
| Μέγιστος χρόνος συνεχόμενης εκτέλεσης | 24 ώρες | 12 ώρες |
| Μέγιστος ανενεργός χρόνος | 90 λεπτά | 60 λεπτά |

### 7.4.2 Νοικιασμένα σε μια υποδομή υπολογιστικού νέφους

#### 7.4.2.1 Γενική επισκόπηση υποδομής υπολογιστικού νέφους Google

Παρέχει μια σουίτα υπηρεσιών υπολογιστικού νέφους (Cloud Computing Services) που τρέχουν στην ίδια υποδομή με τις υπηρεσίες που προσφέρει η Google στους χρήστες της (Google Search, Gmail, Google Drive, YouTube). Αυτό έχει ως αποτέλεσμα να επιτυγχάνονται τεράστιες οικονομίες κλίμακας και να αξιοποιείται το τεχνολογικό υπόβαθρο της εταιρείας προς όφελος των τελικών χρηστών. Οι προσφερόμενες υπηρεσίες καλύπτουν ένα ευρύ φάσμα αναγκών των χρηστών, όπως εικονικά υπολογιστικά μηχανήματα εκτέλεσης πολύπλοκων επεξεργασιών, περιβάλλοντα ανάπτυξης καινοτόμων εφαρμογών (διαδικτυακών, μηχανικής μάθησης, τεχνητής νοημοσύνης, διαδικτύου των πραγμάτων), αποθήκευσης μεγάλων όγκων δεδομένων (δομημένων, αδόμητων), βάσεις δεδομένων (SQL, NoSQL), δικτύωσης, ασφάλειας, ανάλυσης μεγάλου όγκου δεδομένων (Big Data Analytics), εργαλεία διαχείρισης κ.α. Οι υπηρεσίες προσφέρονται μέσω τριών μοντέλων: Software as a Service (SaaS), Platform as a Service (PaaS) και Infrastructure as a Service (IaaS). Το πρώτο περιγράφει τις υπηρεσίες που επιτρέπουν τον τελικό χρήστη να προσπελάσει μέσω ενός διαφυλλιστή ιστού ή γραφικής διεπαφής εφαρμογής οποιαδήποτε εφαρμογή τρέχει στην υποδομή του παρόχου (Cloud Provider). Το δεύτερο περιγράφει τις υπηρεσίες που επιτρέπουν τον τελικό χρήστη να αναπτύξει μια εφαρμογή χρησιμοποιώντας γλώσσες προγραμματισμού, εργαλεία, βιβλιοθήκες και εργαλεία του παρόχου. Τέλος, το τρίτο περιγράφει τις υπηρεσίες που επιτρέπουν στον τελικό χρήστη να χρησιμοποιήσει αποθηκευτικές, επεξεργαστικές, δικτυακές υποδομές, όπου θα τρέχει μια εφαρμογή του. Σε όλα τα παραπάνω μοντέλα υπηρεσίας ο χρήστης δε διαχειρίζεται ή ελέγχει τις λεπτομέρειες της υποκείμενης υποδομής. Συνεπώς, μπορεί να επικεντρωθεί στην ανάπτυξη της εφαρμογής του, επιταχύνοντας τον χρόνο ανάπτυξης και την ποιότητα του τελικού προϊόντος.[19] Στα πλαίσια της διπλωματικής έγινε χρήση των υπηρεσιών Compute Engine (IaaS- Infrastructure as a Service), Vertex AI (PaaS- Platform as a Service) και Google Cloud Storage (IaaS- Infrastructure as a Service). Η πρώτη χρησιμοποιήθηκε κατά την φάση της ανάπτυξης του κώδικα για τη δημιουργία ενός εικονικού μηχανήματος που λειτουργούσε ως αφιερωμένος εξυπηρετητής μιας βάσης δεδομένων Neo4j CE. Η δεύτερη δημιούργησε ένα στιγμιότυπο JupyterLab notebook όπου έτρεξε η εφαρμογή πελάτη σε γλώσσα προγραμματισμού Python και η οποία φόρτωσε τα δεδομένα (κόμβους, συσχετίσεις, περιορισμούς, ετικέτες, ιδιότητες) στην απομακρυσμένη βάση δεδομένων γράφου που ήταν εγκατεστημένη και λειτουργούσε στις υποδομές της σχολής. Αυτή η υπηρεσία δημιούργησε αυτόματα και ένα εικονικό μηχάνημα όπου έτρεχε ο εξυπηρετητής του JupyterLab στιγμιότυπου. Τέλος, η τρίτη δημιούργησε μια ιεραρχικά εμφωλευμένη δομή καταλόγων (Google Cloud Storage Bucket) που περιέχει αποθηκευμένα τα συμπιεσμένα ανεπεξέργαστα δεδομένα μορφής Streaming Json Lines. Όλες οι παραπάνω υπηρεσίες προσφέρονται σε μια Pay–As-You-Use βάση. Δηλαδή ο τελικός χρήστης πληρώνει μόνο για το χρονικό διάστημα χρήσης της υπηρεσίας. Εξαίρεση σε αυτόν τον κανόνα αποτελούν τυχόν επιπρόσθετοι πόροι του συστήματος (Σκληροί δίσκοι, Μνήμη RAM, ip διευθύνσεις) που είναι ενσωματωμένοι στην υπηρεσία που χρησιμοποιεί. Παρόλο που η υπηρεσία μπορεί να είναι απενεργοποιημένη, ο χρήστης εξακολουθεί να τους δεσμεύει και συνεπώς να χρεώνεται με κάποια μικρά επιπλέον ποσά. Γενικά, παρέχεται πλήρη διαφάνεια και έλεγχος ως προς τη χρήση και χρέωση των υπηρεσιών της πλατφόρμας μέσω των στατιστικών στοιχείων της κονσόλας. Η πλατφόρμα επιτρέπει τη διάφανη κλιμάκωση των προσφερόμενων υπηρεσιών, ώστε να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες των χρηστών. Αυτό απελευθερώνει τον τελικό χρήστη από την ανάγκη πραγματοποίησης πολυέξοδων επενδύσεων, επαναλαμβανόμενων εξόδων διαχείρισης και χρονοβόρων διαδικασιών μεταφοράς των δεδομένων και των υπηρεσιών του σε άλλες πιο ισχυρές

υποδομές. Όλα διεκπεραιώνονται αυτόματα από την υποδομή υπολογιστικού νέφους της Google και με τις απολύτως ελάχιστες ρυθμίσεις από την πλευρά του χρήστη.

#### 7.4.2.2 Εικονικά μηχανήματα υπολογιστικού νέφους Google

Μέσω της υπηρεσίας Compute Engine ο χρήστης έχει τη δυνατότητα να δημιουργήσει πλήρως προσαρμόσιμα εικονικά μηχανήματα που τρέχουν στην υποδομή της Google. Μέσω της κονσόλας, της γραμμής εντολών ή του API μπορεί κάποιος να δημιουργήσει ένα απλό στιγμιότυπο ή μια ομάδα στιγμιότυπων εικονικών μηχανημάτων (Instance Groups). Η πρώτη δυνατότητα περιλαμβάνει ατομικά εικονικά μηχανήματα που κατασκευάζονται είτε με βάση τα χαρακτηριστικά κάποιων οικογενειών εικονικών μηχανημάτων (Virtual Machines Families), είτε είναι πλήρως προσαρμόσιμα από τον τελικό χρήστη. Επιπλέον, διατίθενται οικογένειες μηχανημάτων γενικού σκοπού, προσανατολισμένες στους υπολογισμούς, στη μνήμη ή στη χρήση GPUs για εξαιρετικά απαιτητικές ροές επεξεργασίας. Από την άλλη, ο τελικός χρήστης έχει τη δυνατότητα να δημιουργήσει ένα πλήρως προσαρμοσμένο στις ανάγκες του μηχάνημα. Σε αυτήν την περίπτωση ο χρήστης μπορεί να επιλέξει την εικόνα βάσει της οποίας θα δημιουργηθεί ο δίσκος εκκίνησης του λειτουργικού συστήματος, την γεωγραφική περιοχή όπου θα βρίσκεται φυσικά το μηχάνημα, το μέγεθος του σκληρού δίσκου, της μνήμης, το πλήθος των εικονικών επεξεργαστών και προαιρετικά το είδος και το πλήθος των GPUs. Στο σημείο αυτό κρίνεται σκόπιμο να αναφέρουμε ότι το πλήθος των CPUs αναφέρεται σε εικονικές CPUs (vCPUs – virtual CPUs) και όχι σε φυσικές CPUs (pCPUs – physical CPUs). Το πλήθος τους υπολογίζεται από τον παρακάτω τύπο.

$$vCPUs = threads\_per\_physical\_core \times physical\_cores\_per\_socket \times number\_of\_sockets \quad (7.1)$$

Η δεύτερη δυνατότητα ουσιαστικά αποτελείται από πολλαπλά εικονικά μηχανήματα που μπορούμε να τα διαχειριστούμε με ενιαίο τρόπο. Υποστηρίζονται δυο τύποι ομάδων στιγμιότυπων: Managed Instance Groups (MIGs) και Unmanaged Instance Groups (UIGs). Ο πρώτος τύπος περιλαμβάνει κάθε ομάδα εικονικών μηχανημάτων που δημιουργήθηκε βάσει ενός κοινού προτύπου (Instance Template). Συνεπώς, όλα τα στιγμιότυπα είναι του ίδιου τύπου μηχανήματος, βασίζονται στην ίδια εικόνα για τη δημιουργία του δίσκου εκκίνησης του λειτουργικού συστήματος και περιλαμβάνουν τις ίδιες ρυθμίσεις. Ουσιαστικά, χρησιμοποιούνται σε περιπτώσεις που θέλουμε να επιτύχουμε υψηλή διαθεσιμότητα (Regional coverage, Auto healing, load balancing), αυτόματη κλιμάκωση και αυτόματες ενημερώσεις των εφαρμογών μας. Ο δεύτερος τύπος περιλαμβάνει κάθε ομάδα ετερογενών εικονικών μηχανημάτων. Δεν υποστηρίζουν αυτόματη κλιμάκωση (Auto-scaling) και αυτόματη επιδιόρθωση των στιγμιότυπων (Auto-healing). Στα πλαίσια της διπλωματικής δημιουργήθηκαν πολλαπλά απλά στιγμιότυπα εικονικών μηχανημάτων που εξυπηρετούσαν ως αφιερωμένοι εξυπηρετητές βάσεων δεδομένων Γράφου Neo4j ή ως εξυπηρετητές JupyterLab notebooks όπου έτρεχε η εφαρμογή πελάτης σε Python για την εισαγωγή των δεδομένων στη βάση δεδομένων γράφου.

## 7.5 Επίλογος

Στο κεφάλαιο αυτό παρατέθηκαν οι τεχνολογίες στις οποίες βασίστηκε η υλοποίηση της υποδομής της διπλωματική. Συγκεκριμένα, αρχικά χρησιμοποιήθηκε ο επαγγελματικού επιπέδου συντάκτης κειμένου EmEditor. Χάρη σε αυτόν άνοιξαν τα τεράστια αρχεία κειμένου JSON Streaming Lines και ολοκληρώθηκε το βήμα της αρχικής διερευνητικής ανάλυσης των συνόλων δεδομένων. Επίσης, με την βοήθεια αυτού του συντάκτη αυτοματοποιήθηκε η διαδικασία τεμαχισμού των αρχείων JSON σε μικρότερα μεγέθη, ώστε να είναι πιο διαχειρίσιμα. Στην συνέχεια, χρησιμοποιήθηκε η διαδικτυακή εφαρμογή arrows.app, ώστε να δημιουργηθούν τα σχήματα των δυο εκδόσεων του μοντέλου

δεδομένων. Τέλος, η έλλειψη σε τοπικό επίπεδο ενός ισχυρού υπολογιστικού μηχανήματος, οδήγησε στη χρήση διαφόρων τύπων εικονικών μηχανημάτων. Αυτά γενικά ανήκουν σε δυο κατηγορίες ανάλογα με το κόστος χρήσης. Στην πρώτη κατηγορία ανήκουν τα δωρεάν εικονικά μηχανήματα που προσφέρουν οι πλατφόρμες https://colab.research.google.com/ και https://www.kaggle.com/. Τα μηχανήματα αυτά χρησιμοποιήθηκαν κατά την αρχική φάση της ανάπτυξης του κώδικα Python και έτρεχαν σε μικρά αντιπροσωπευτικά υποσύνολα των βιβλιογραφικών δεδομένων. Στη δεύτερη κατηγορία ανήκουν τα νοικιασμένα εικονικά μηχανήματα της υποδομής υπολογιστικού νέφους της Google. Τα μηχανήματα αυτά χρησιμοποιήθηκαν κατά την τελική φάση ανάπτυξης του κώδικα Python και έτρεχαν στα πλήρη σύνολα των βιβλιογραφικών δεδομένων. Όλα τα παραπάνω εικονικά μηχανήματα έτρεχαν διαδραστικά κώδικα Python σε μορφή notebook και σε περιβάλλον Linux. Στο τέλος της ερευνητικής προσπάθειας καταλήξαμε στο συμπέρασμα ότι οι δυο βασικοί τεχνολογικοί πυλώνες στους οποίους βασιστήκαμε αποτελούν βιώσιμες λύσεις και μπορεί να οδηγήσουν σε βιβλιογραφικές υποδομές με αξιόλογη απόδοση. Ωστόσο. οι σημαντικές προκλήσεις που αναδείχτηκαν αναμένεται να πυροδοτήσουν συμπληρωματικές προς την παρούσα ερευνητικές προσπάθειες. Αυτές θα διερευνήσουν επιπλέον τεχνολογίες υλοποίησης, όπως νέες βιβλιοθήκες, γλώσσες προγραμματισμού, βάσεις δεδομένων, ροές επεξεργασίας, μέσα απομακρυσμένης αποθήκευσης και προσπέλασης μεγάλων όγκων δεδομένων κ.α. Μερικές ιδέες και κατευθύνσεις προτείνονται στο επόμενο κεφάλαιο που είναι και το τελευταίο της παρούσας διπλωματικής εργασίας.

# Κεφάλαιο 8ο: Επίλογος

## 8.1 Εισαγωγή

Στα προηγούμενα κεφάλαια περιγράφηκαν αναλυτικά όλες οι όψεις που υπεισέρχονται στην κατασκευή της βιβλιογραφικής υποδομής. Στο παρόν κεφάλαιο επικεντρώνουμε την προσοχή μας στα συμπεράσματα που αποκομίσαμε από την παραπάνω διαδικασία. Ωστόσο, στο σημείο αυτό επιβάλλεται να αναφέρουμε ότι η παρούσα διπλωματική εργασία απλά διερεύνησε μια κατεύθυνση υλοποίησης. Δεν αποκλείεται μελλοντικές ερευνητικές προσπάθειες να διαφοροποιήσουν κομμάτια αυτής της υλοποίησης, αντικαθιστώντας τα με άλλα πιο αποδοτικά και αποτελεσματικά. Εξάλλου, όπως επισημάνθηκε σε πολλά σημεία της διπλωματικής εργασίας, τα διάφορα τμήματα που συνθέτουν την αρχιτεκτονική δομή της υποδομής είναι χαμηλής σύζευξης. Αυτό διευκολύνει την αντικατάσταση οποιουδήποτε κομματιού της υποδομής με άλλα πιο σύγχρονης υλοποίησης χωρίς ιδιαίτερη προγραμματιστική προσπάθεια.

## 8.2 Σύνοψη και συμπεράσματα

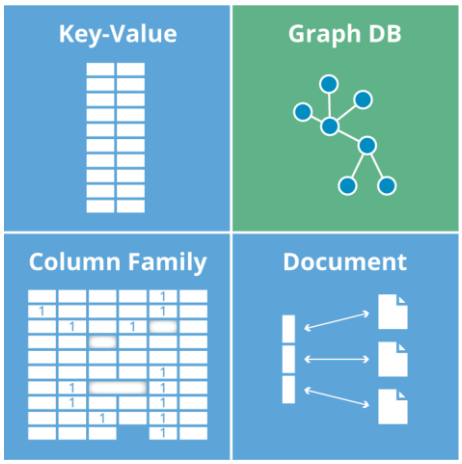
Στα πλαίσια της διπλωματικής εργασίας δημιουργήθηκε μια υποδομή παραλληλοποιημένης και κατανεμημένης διαχείρισης μεγάλων βιβλιογραφικών δεδομένων. Το σύστημα που δημιουργήθηκε βασίστηκε σε δυο τεχνολογικού πυλώνες. Τη μη σχεσιακή τεχνολογία δημιουργίας βάσεων δεδομένων γράφων (Neo4j NoSQL Graph Database) και τη βιβλιοθήκη κατανεμημένης και παραλληλοποιημένης εκτέλεσης εφαρμογών Python Dask. Ο πρώτος πυλώνας επέτρεψε τη δημιουργία μιας βάσης δεδομένων που βασίζεται σε μια ριζοσπαστική φιλοσοφία η οποία δίνει ίση αξία στις συνδέσεις με τα ίδια τα δεδομένα. Για αυτό τον λόγο, φροντίζει να τις αποθηκεύει αποδοτικά μαζί με τα δεδομένα. Η τεχνολογία αυτή επέτρεψε την αποδοτική εκτέλεση έξυπνων ερωτημάτων που σχετίζονται με τον βαθμό διασύνδεσης των δεδομένων. Σε αυτά συγκαταλέγονται ερωτήματα που αναπαριστούν μεταβλητού μήκους μονοπάτια, εύρεση κυκλικών μονοπατιών, εύρεση συντομότερου μονοπατιού κ.α. Τέτοιου είδους ερωτήματα δεν υποστηρίζονται αποτελεσματικά από την παραδοσιακή τεχνολογία των σχεσιακών βάσεων δεδομένων. Επέτρεψε επίσης την παραγωγή μετρικών επάνω στα βιβλιογραφικά δεδομένα που αποκαλύπτουν νέα γνώση και η οποία μπορεί να χρησιμοποιηθεί για τη λήψη αποφάσεων. Η χρήση ενός ευέλικτου μοντέλου δεδομένων που μπορεί να αλλάξει δυναμικά ανάλογα με τις ανάγκες των χρηστών, και η εύκολη κατανόησή του από τον χρήστη συμπεριλαμβάνονται στις ποιότητες που συνέβαλαν στην επιλογή της. Ο δεύτερος πυλώνας επέτρεψε να εκτελέσουμε τμηματικά, παραλληλοποιημένα και κατανεμημένα πολύπλοκους υπολογισμούς ανεξάρτητα των ιδιαίτερων χαρακτηριστικών της υπολογιστικής υποδομής. Έτσι κατέστη δυνατόν να δημιουργηθεί μια CPU bound εφαρμογή με μειωμένους χρόνους εκτέλεσης. Η όλη υποδομή απαρτίζεται από τμήματα με χαμηλό βαθμό σύζευξης. Συνεπώς, αν μελλοντικά απαιτηθεί, μπορεί να αντικατασταθεί οποιοδήποτε τμήμα της με ένα άλλο πιο σύγχρονης τεχνολογίας, χωρίς να πυροδοτηθεί μια χιονοστιβάδα αλλαγών σε όλη την έκταση της υποδομής. Στόχος της δημιουργημένης υποδομής είναι η έξυπνη οργάνωση της βιβλιογραφικής γνώσης. Μέσω αυτής της υποδομής μπορεί μελλοντικά να συντομευθεί η διαδικασία εντοπισμού χρήσιμης γνώσης, συνεργατών και μέσων δημοσίευσης από μελλοντικά ερευνητικά σχήματα.

## 8.3 Μελλοντικές επεκτάσεις

Η συγκεκριμένη διπλωματική εργασία αναπτύχθηκε στο υλικό ενός εικονικού μηχανήματος της υπολογιστικής υποδομής του πανεπιστημίου. Μια όμως από τις μεγαλύτερες αρετές της βιβλιοθήκης

Dask είναι η εύκολη κλιμάκωση των υπολογισμών στους κόμβους που απαρτίζουν οποιασδήποτε αρχιτεκτονικής κατανεμημένη υπολογιστική συστοιχία. Η οριζόντια και κατακόρυφη κλιμάκωση που επιτυγχάνει η βιβλιοθήκη γίνεται ανεξάρτητα των ιδιαίτερων χαρακτηριστικών του υπολογιστικού συστήματος που εκτελείται. Μπορεί με τρόπο διάφανο προς τον τελικό χρήστη να εκτελεστεί αποδοτικά σε έναν μεμονωμένο υπολογιστή, σε μια συστοιχία υπολογιστικών κόμβων, σε ένα υπολογιστικό σύστημα υψηλής απόδοσης, σε ένα εικονικό μηχάνημα μιας υποδομής υπολογιστικού νέφους κλπ. Εξίσου ευέλικτη είναι η βιβλιοθήκη και ως προς το είδος των υπολογιστικών κόμβων που χρησιμοποιούνται. Η συγκεκριμένη διπλωματική εργασία χρησιμοποίησε αποκλειστικά τους πυρήνες ενός υπολογιστή. Μελλοντικές ερευνητικές προσπάθειες θα μπορούσαν να εστιάσουν στην εκτέλεση των υπολογισμών σε πολλαπλές GPU’s συντομεύοντας ακόμα περισσότερο τον χρόνο εκτέλεσης μιας εφαρμογής. Στην ίδια λογική, μελλοντικές ερευνητικές προσπάθειες θα μπορούσαν να διερευνήσουν τη χρήση μεικτών αρχιτεκτονικών σχημάτων από κόμβους επεξεργασίας που περιλαμβάνουν και CPU’s και GPU’s. Εξάλλου, η βιβλιοθήκη Dask δίνει τη δυνατότητα στον χρήστη να επιλέξει τους κόμβους επεξεργασίας που θα αναλάβουν την εκτέλεση των πιο απαιτητικών υπολογισμών, αλλά και των υπόλοιπων πόρων που θα δεσμεύσουν. Προς αυτήν την κατεύθυνση, θα πρέπει να ληφθούν υπόψιν και επιπλέον βιβλιοθήκες, όπως η CUDF και η DASK-CUDF. Οι βιβλιοθήκες αυτές υποστηρίζουν δομές δεδομένων παρόμοιας αίσθησης με ένα Pandas DataFrame, που όμως εκτελούνται ταχύτατα στους πολλαπλούς πυρήνες μιας GPU, εκμηδενίζοντας τους χρόνους εκτέλεσης.

Προς την κατεύθυνση της αποδοτικής και αποτελεσματικής διαχείρισης μεγάλων όγκων δεδομένων, αξίζει επίσης μελλοντικά να διερευνηθούν και επιπλέον τεχνολογίες που θα αντικαταστήσουν τους δυο βασικούς τεχνολογικούς πυλώνες. Όσον αφορά τον πρώτο τεχνολογικό πυλώνα, θα μπορούσε να διερευνηθεί η πιθανότητα χρήσης και άλλων τύπων μη σχεσιακών βάσεων δεδομένων (NoSQL Databases) που το μοντέλο τους δεν βασίζεται στους γράφους. Σε αυτές συγκαταλέγονται βάσεις δεδομένων ζευγών κλειδιού – τιμής (Key-Value NoSQL Databases), βάσεις δεδομένων βασισμένες σε έγγραφα (Document-based Databases), βάσεις δεδομένων βασισμένες σε στήλες (Column-based Databases) κ.α., όπως απεικονίζεται στο Σχήμα 8.1.

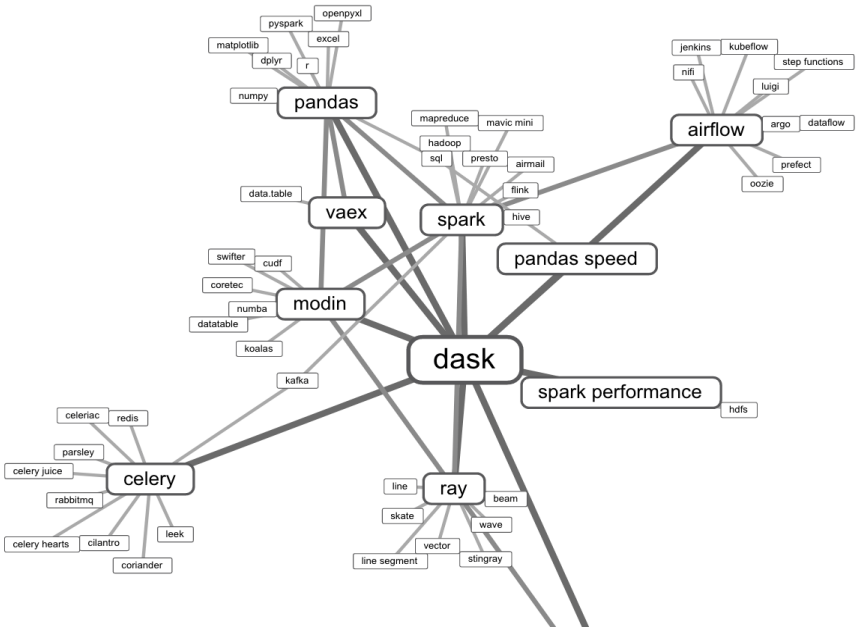



Σχήμα 8.1: Διάφορες μη σχεσιακές τεχνολογίες βάσεων δεδομένων (Πηγή [55])

Προς την ίδια κατεύθυνση θα μπορούσαν να εκπονηθούν συγκριτικές μελέτες αξιολόγησης της απόδοσης των διαφορετικών μη σχεσιακών τεχνολογιών βάσεων δεδομένων μεταξύ τους καθώς και με την παραδοσιακή τεχνολογία των σχεσιακών βάσεων δεδομένων. Ωστόσο ο παρών ερευνητής θεωρεί ότι είναι εξαιρετικά απίθανο να βρεθεί οποιαδήποτε τεχνολογία βάσεων δεδομένων που δε βασίζεται στους γράφους που να διαχειρίζεται εξίσου αποδοτικά τις σχέσεις μεταξύ των δεδομένων.

Ωστόσο, θα μπορούσαν μελλοντικές ερευνητικές προσπάθειες να εστιάσουν και σε άλλες τεχνολογίες βάσεων δεδομένων γράφου που χρησιμοποιούν όμως ίδιο μοντέλο δεδομένων. Υποψήφιες τεχνολογίες θα μπορούσαν να είναι οι υπεργράφοι (Hypergraphs) και οι αποθήκες τριπλετών (Triple Stores). Η πρώτη βασίζεται σε ένα διαφορετικής φιλοσοφίας μοντέλο δεδομένων που μπορεί να υλοποιήσει εύκολα συσχετίσεις «πολλά – προς - πολλά». Συγκεκριμένα, βασίζεται στην υπερακμή για να υλοποιήσει μια συσχέτιση. Η διαφορά με το μοντέλο ιδιοτήτων γράφου του Neo4j έγκειται στο ότι μια υπερακμή μπορεί να συνδέει οποιοδήποτε αριθμό κόμβων σε οποιοδήποτε άκρο της. Η δεύτερη βασίζεται σε ένα διαφορετικής φιλοσοφίας μοντέλο δεδομένων που προέρχεται από τον χώρο του σημασιολογικού ιστού (Semantic Web). Συγκεκριμένα, βασίζεται σε τριπλέτες της μορφής υποκείμενο-κατηγόρημα-αντικείμενο για να υλοποιήσει μια συσχέτιση. Κάθε τέτοια συσχέτιση μπορεί να ενσωματώσει χρήσιμη σημασιολογική πληροφορία σύμφωνα με το πρότυπο RDF. Στη συνέχεια, η σημασιολογική πληροφορία μπορεί να εξαχθεί με ερωτήσεις στη γλώσσα SPARQL.

Όσον αφορά τον δεύτερο τεχνολογικό πυλώνα, θα μπορούσε να διερευνηθεί η πιθανότητα χρήσης και άλλων βιβλιοθηκών και frameworks που υποστηρίζουν παραλληλοποιημένη και κατανεμημένη επεξεργασία. Οι σημαντικότερες βιβλιοθήκες και frameworks περιγράφονται στο Σχήμα 8.2[56].




Σχήμα 8.2: Βιβλιοθήκες παραλληλοποιημένης και κατανεμημένης επεξεργασίας μεγάλων δεδομένων ( Πηγή [56])

Στην συνέχεια ακολουθεί ένας συγκριτικός πίνακας των δυνατοτήτωντων παραπάνω βιβλιοθηκών [56].

Πίνακας 8.1: Σύγκριση βιβλιοθηκών παραλληλοποιημένης και κατανεμημένης επεξεργασίας μεγάλων δεδομένων (Πηγή [56])

| | Maturity | Popularity | Ease of Adoption | Scaling Ability | Use Cases | Main scaling Strategy |
|---|---|---|---|---|---|---|
| Dask | A | B | B | 1 TB+ | Data Science/ General | Clusters |
| Vaex | C | C | A | 100 GB+ | Data Science | Lazy Loading |
| Rapids (Cudf) | B | C | C | 100 GB+ | Data Science | GPU's |
| Modin | C | C | A | 10 GB+ | DataFrame | Clusters |
| Ray | A | A | B | 1 TB+ | General | Clusters |

Στα πλαίσια της παρούσας διπλωματικής χρησιμοποιήθηκε η βιβλιοθήκη Dask προκειμένου να υλοποιηθούν ροές επεξεργασίας μεγάλων δεδομένων κατά τμήματα (Batch Processing Pipelines). Προς την κατεύθυνση της επιτάχυνσης της ταχύτητας επεξεργασίας δεδομένων, θα μπορούσαν να χρησιμοποιηθούν και άλλα frameworks επεξεργασίας ροών, όπως το Kafka ή και το χαρακτηριστικό Structured Streaming της βιβλιοθήκης Spark. Αυτά αντιμετωπίζουν αποτελεσματικά τα προβλήματα απόδοσης που παρουσιάζει η παραδοσιακή ροή ETL. Στις αρετές που παρουσιάζουν συγκαταλέγονται η δυνατότητα διαβάσματος δεδομένων από διαφορετικές πηγές σε σχεδόν πραγματικό χρόνο, η χρήση ενός API για δυναμική επεξεργασία των δεδομένων κ.α.

Στην παρούσα διπλωματική εργασία φορτώθηκε στη βάση δεδομένων γράφου ένα υποσύνολο των διαθέσιμων πεδίων του συνόλου δεδομένων Mag. Η απόφασή αυτή βασίστηκε αποκλειστικά στο μεγάλο μέγεθος και το πλήθος πολύ σημαντικών και ενδιαφερόντων πεδίων που περιλαμβάνονται στο συγκεκριμένο σύνολο δεδομένων. Εξάλλου, εξαρχής σκοπός της διπλωματικής ήταν να διερευνήσει τη δυνατότητα και αποδοτικότητα των δυο βασικών πυλώνων της εργασίας (Dask, Neo4j) να αποθηκεύσουν και να διαχειριστούν πολύ μεγάλους όγκους βιβλιογραφικών δεδομένων. Ωστόσο, μελλοντικές ερευνητικές προσπάθειες θα μπορούσαν να διερευνήσουν τη δυνατότητα εμπλουτισμού της βιβλιογραφικής πληροφορίας που είναι αποθηκευμένη στη βάση δεδομένων γράφου της υποδομής με επιπλέον χρήσιμα δεδομένα, κάνοντας χρήση επιπλέον ανοικτών βιβλιογραφικών πηγών δεδομένων (Aminer, OpenCitations κ.α). Σε αυτήν την περίπτωση θα πρέπει να υπερνικηθούν σημαντικές προκλήσεις, όπως η διαφορετική μορφή διάθεσης των δεδομένων, η ανομοιογένεια και ο πλεονασμός που εμφανίζουν. Τέλος, επισημαίνεται ότι η τρέχουσα υποδομή μπορεί να εμπλουτιστεί με επιπλέον βιβλιογραφικές πληροφορίες που δημιουργούνται από την εκτέλεση αλγορίθμων γράφου. Αυτές οι πληροφορίες είναι εξαιρετικά χρήσιμες, μιας και αξιολογούν την οργάνωση και τον βαθμό διασύνδεσης των δεδομένων στα πλαίσια της επιστημονικής κοινότητας.

Οι επιπλέον ερευνητικές προσπάθειες θα πυροδοτηθούν από την τάση που παρατηρείται μεγάλοι ιδιωτικοί και κυβερνητικοί οργανισμοί να μεταφέρουν το σύνολο των κρίσιμης σημασίας δεδομένων τους σε σχήματα οργάνωσης NoSQL. Επιπρόσθετα, οι ίδιοι οργανισμοί κάνουν εκτεταμένη χρήση βιβλιοθηκών και frameworks παραλληλοποιημένης και κατανεμημένης επεξεργασίας, ώστε να αντιμετωπιστεί ο μεγάλος όγκος, ο υψηλός ρυθμός παραγωγής, η κατανεμημένη φύση, η ανάγκη αυτοματοποιημένης επεξεργασίας σε πραγματικό χρόνο, οι πολύπλοκες ροές επεξεργασίας που πρέπει να εφαρμοστούν και η ισχυρή διασύνδεσήτων δεδομένων τους. Μόνο κάτω από αυτό το πρίσμα τα σύγχρονα δεδομένα μπορούν να επεξεργαστούν αποτελεσματικά και αποδοτικά και να δημιουργήσουν μέγιστη αξία για τον χρήστη που τα καταναλώνει και ισχυρό ανταγωνιστικό πλεονέκτημα για τον οργανισμό που τα διαχειρίζεται.

## 8.4 Επίλογος

Στο τελευταίο κεφάλαιο της διπλωματικής εργασίας καταλήγουμε στο συμπέρασμα ότι οι δυο βασικοί τεχνολογικοί πυλώνες που χρησιμοποιήθηκαν, όντως αποτελούν ρεαλιστικές επιλογές για την επιτυχή υλοποίηση μιας βιβλιογραφικής υποδομής. Επίσης, υποδεικνύουν την ορθή κατεύθυνση που πρέπει να ακολουθήσουν μελλοντικές ερευνητικές προσπάθειες. Οι μελλοντικές εργασίες αξίζει να διερευνήσουν και άλλες εναλλακτικές προτάσεις που θα αντικαταστήσουν τους δυο βασικούς τεχνολογικούς πυλώνες. Όσον αφορά τον πρώτο πυλώνα, η εργασία ισχυρίζεται ότι ο εγγενής τρόπος αποθήκευσης των σχέσεων μαζί με τα δεδομένα του Neo4j οδηγεί σε βέλτιστη απόδοση προσπέλασης τους. Ωστόσο, αξίζει να διερευνηθούν και άλλες μη σχεσιακές τεχνολογίες βάσεων δεδομένων γράφου που βασίζονται όμως σε διαφορετικό μοντέλο. Πολλά υποσχόμενη επιλογή αποτελούν οι υπεργράφοι οι οποίοι διαφοροποιούνται από το Neo4jστο ότι κάθε σχέση μπορεί σε κάθε άκρο της να συνδέει πολλαπλούς κόμβους. Όσον αφορά τον δεύτερο πυλώνα, αξίζει να διερευνηθεί η ευκολία και αποδοτικότητα διαχείρισης των βιβλιογραφικών δεδομένων από άλλες παραλληλοποιημένες βιβλιοθήκες και frameworks. Κάποιες από αυτές κατανέμουν τους πολύπλοκους υπολογισμούς στους πολλαπλούς πυρήνες των CPU’s ενός υπολογιστικού μηχανήματος. Κάποιες άλλες κατανέμουν τους υπολογισμούς στους χιλιάδες πυρήνες των GPU’s ενός υπολογιστικού μηχανήματος. Τέλος, κάποιες άλλες υποστηρίζουν και μικτά σχήματα στα οποία επιτρέπεται στον τελικό χρήστη να επιλέξει την εκτέλεση των πιο απαιτητικών υπολογισμών σε GPU’sκαι τους υπόλοιπους στις διαθέσιμες CPU’s. Στο κεφάλαιο παρατίθεται ένα σχήμα με τις πιο σημαντικές διαθέσιμες επιλογές καθώς και ένας συγκριτικός πίνακας που μπορεί να διευκολύνει την τελική επιλογή. Τέλος, επιβάλλεται η χρήση και άλλων πιο σύγχρονων μορφών ροών επεξεργασίας με καλύτερα χαρακτηριστικά απόδοσης από τη βασική ETL. Η παρούσα διπλωματική εργασία ισχυρίζεται ότι έθεσε τις σωστές βάσεις και κατευθύνσεις δημιουργίας βιβλιογραφικών συστημάτων με αξιόλογες αποδόσεις.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] IDC Team, “A Data-Driven World Needs Its Data Protected and the Global DataSphere ECURITY”.

[2] F. Zhang *et al.*, “OAG: Toward linking large-scale heterogeneous entity graphs,” 2019. doi: 10.1145/3292500.3330785.

[3] A. Sinha *et al.*, “An overview of microsoft academic service (MAS) and applications,” *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, pp. 243–246, May 2015, doi: 10.1145/2740908.2742839.

[4] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “ArnetMiner: Extraction and mining of academic social networks,” 2008. doi: 10.1145/1401890.1402008.

[5] J. J. Miller, “Graph Database Applications and Concepts with Neo4j”.

[6] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, “A Comparison of a Graph Database and a Relational Database A Data Provenance Perspective,” *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE ’10*, 2010, doi: 10.1145/1900008.

[7] R. Angles, “A comparison of current graph database models,” 2012. doi: 10.1109/ICDEW.2012.31.

[8] M. Rocklin, “Dask: Parallel Computation with Blocked algorithms and Task Scheduling,” 2015. doi: 10.25080/majora-7b98e3ed-013.

[9] M. Dugre, V. Hayot-Sasson, and T. Glatard, “A performance comparison of dask and apache spark for data-intensive neuroimaging pipelines,” 2019. doi: 10.1109/WORKS49585.2019.00010.

[10] F. Gafarov, D. Minullin, and V. Gafarova, “Dask-Based Efficient Clustering of Educational Texts ⋆ Fail Gafarov 1[0000−0003−4704−154X] , Dmitriy Minullin 1[0000−0001−7713−5251] , and Viliuza Gafarova 2[0000−0001−7215−4007],” 2021.

[11] J. Crist, “Dask & Numba: Simple libraries for optimizing scientific python code,” *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 2342–2343, 2016, doi: 10.1109/BIGDATA.2016.7840867.

[12] M. Needham and A. E. Hodler, *Graph algorithms: Practical examples in Apache Spark & Neo4j*. 2020.

[13] Neo4j Team, “Introduction - Operations Manual.” https://neo4j.com/docs/operations-manual/current/introduction/#enterprise-edition (accessed Dec. 25, 2021).

[14] Neo4j Team, “Drivers & Language Guides - Developer Guides.” https://neo4j.com/developer/language-guides/ (accessed Jan. 14, 2022).

[15] Neo4j Team, “Getting Started with Neo4j - Developer Guides.” https://neo4j.com/developer/get-started/ (accessed Sep. 01, 2021).

[16] R. van Bruggen, *Learning Neo4j*. 2014.

[17] Neo4j, “Neo4j: The World Leading Graph Database,” *Neo4J.Org*, 2016.

[18] A. Kollegger, “Graph Databases for Beginners: Native vs. Non-Native Graph Technology - DZone Database.” https://dzone.com/articles/graph-databases-for-beginners-native-vs-non-native (accessed Oct. 10, 2021).

[19] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases - New Opportunities for Connected Data*. 2015.

[20] Neo4j Team, “5 Graph Data Science Basics Everyone Should Know”.

[21] Neo4j Team, “H E L P I N G T H E W O R L D M A K E S E N S E O F D A T A The Graph Database Buyer’s Guide.”

[22] Neo4j Team, “Graph Data Modeling - Developer Guides.” https://neo4j.com/developer/data-modeling/ (accessed Sep. 07, 2021).

[23] Neo4j Team, “Graph Data Modeling for Neo4j - Graph Data Modeling for Neo4j.” https://neo4j.com/graphacademy/training-gdm-40/00-graph-data-modeling-about/ (accessed Sep. 07, 2021).

[24] Neo4j Team, “Graph Data Modeling Core Principles - Graph Data Modeling for Neo4j.” https://neo4j.com/graphacademy/training-gdm-40/03-graph-data-modeling-core-principles/ (accessed Jul. 14, 2022).

[25] Μ. Παναγοπούλου, “‘NoSQL databases : Ποιοτική και Ποσοτική Σύγκριση μεταξύ των Cassandra, BaseX και Mongodb.’”

[26] A. Ibrahim, “Relational Database transactions (ACID) – Ahmed Ibrahim.” https://www.ahmed-ibrahim.com/relational-database-transaction-acid-in-dbms/ (accessed Dec. 27, 2021).

[27] Neo4j Team, “Neo4j Labs - Neo4j Graph Data Platform.” https://neo4j.com/labs/ (accessed Oct. 14, 2021).

[28] Neo4j Team, “Introduction - Neo4j Graph Data Science.” https://neo4j.com/docs/graph-data-science/current/introduction/ (accessed Dec. 08, 2021).

[29] Neo4j Team, “Neo4j Graph Data Science - Developer Guides.” https://neo4j.com/developer/graph-data-science/ (accessed Sep. 07, 2021).

[30] Neo4j Team, “Common usage - Neo4j Graph Data Science.” https://neo4j.com/docs/graph-data-science/current/common-usage/ (accessed Jan. 09, 2022).

[31] Neo4j Team, “Centrality - Neo4j Graph Data Science.” https://neo4j.com/docs/graph-data-science/current/algorithms/centrality/ (accessed Jul. 15, 2022).

[32] Neo4j Team, “Community detection - Neo4j Graph Data Science.” https://neo4j.com/docs/graph-data-science/current/algorithms/community/ (accessed Jul. 15, 2022).

[33] Neo4j Team, “Path finding - Neo4j Graph Data Science.” https://neo4j.com/docs/graph-data-science/current/algorithms/pathfinding/ (accessed Jul. 15, 2022).

[34] the free encyclopedia Wikipedia, “Python (programming language) - Wikipedia.” https://en.wikipedia.org/wiki/Python_(programming_language) (accessed Nov. 10, 2021).

[35] M. Schmitt, “Understanding Dask Architecture: Client, Scheduler, Workers.” https://www.datarevenue.com/en-blog/understanding-dask-architecture-client-scheduler-workers (accessed Jan. 24, 2022).

[36] J. C. Daniel, “Data science with Python and Dask,” p. 296, 2019.

[37] Dask Team, “Dask — Dask documentation.” https://docs.dask.org/en/latest/ (accessed Jul. 14, 2022).

[38] Dask Team, “Dask — Dask documentation.” https://docs.dask.org/en/latest/ (accessed Nov. 19, 2021).

[39] S. Salan and J. McGuffee, *Data Science with Python DASK*, vol. 32, no. 5. 2017.

[40] Dask Team, “Connect to remote data — Dask documentation.” https://docs.dask.org/en/latest/how-to/connect-to-remote-data.html (accessed Feb. 17, 2022).

[41] The free encyclopedia Wikipedia, “LZ4 (compression algorithm) - Wikipedia.” https://en.wikipedia.org/wiki/LZ4_(compression_algorithm) (accessed Jul. 14, 2022).

[42] Dask Team, “Dashboard Diagnostics — Dask documentation.” https://docs.dask.org/en/latest/dashboard.html (accessed Jul. 14, 2022).

[43] Panoply, “Modern Data Management,” *Panoply.com*, 2017.

[44] Neo4j Team, “Overview of Importing Data into Neo4j - Importing Data with Neo4j 4.x.” https://neo4j.com/graphacademy/training-importing-data-40/01-import-40-overview-importing/ (accessed Jul. 14, 2022).

[45] A. Vukotic and N. Watt, “Neo4j in Action,” p. 350, 2013, Accessed: Jul. 14, 2022. [Online]. Available: http://books.google.com/books?id=61GdmgEACAAJ&pgis=1

[46] Neo4j Team, “Neo4j Admin import - Operations Manual.” https://neo4j.com/docs/operations-manual/current/tutorial/neo4j-admin-import/ (accessed Jul. 14, 2022).

[47] Neo4j Team, “Import - Operations Manual.” https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/neo4j-admin-import/ (accessed Jul. 14, 2022).

[48] J. Rocha, “Understanding memory consumption - Knowledge Base.” https://neo4j.com/developer/kb/understanding-memory-consumption/ (accessed Jul. 15, 2022).

[49] Neo4j Team, “Neo4j Performance Tuning - Developer Guides.” https://neo4j.com/developer/guide-performance-tuning/ (accessed Jul. 15, 2022).

[50] Neo4j Team, “Space reuse - Operations Manual.” https://neo4j.com/docs/operations-manual/current/performance/space-reuse/#space-reuse-id-files (accessed Jul. 14, 2022).

[51] D. Canzano, “Understanding Database Growth - Knowledge Base.” https://neo4j.com/developer/kb/understanding-database-growth/?_gl=1*1f5iui9*_ga*MzQ5OTI3OTkxLjE2MTU1NjAyMjk.*_ga_DL38Q8KGQC*MTY1NDMzODQ2NS42MDcuMC4xNjU0MzM4NDY1LjA. (accessed Mar. 05, 2022).

[52] M. H. Namaki, F. A. R. R. Chowdhury, R. Islam, J. R. Doppa, and Y. Wu, “Learning to speed up query planning in graph databases,” 2017.

[53] Emurasoft Team, “Text Editor Features – EmEditor (Text Editor).” https://www.emeditor.com/text-editor-features/ (accessed Oct. 08, 2021).

[54] A. Kazemnejad, “How to do Deep Learning research with absolutely no GPUs - Part 2 - Amirhossein Kazemnejad’s Blog.” https://kazemnejad.com/blog/how_to_do_deep_learning_research_with_absolutely_no_gpus_part_2/ (accessed Feb. 26, 2022).

[55] B. Sasaki, “The Wide Spectrum of Graph Database Technologies.” https://neo4j.com/blog/other-graph-database-technologies/?ref=blog (accessed Feb. 10, 2022).

[56] M. Schmitt, “Scaling Pandas: Dask vs Ray vs Modin vs Vaex vs RAPIDS.” https://datarevenue.com/en-blog/pandas-vs-dask-vs-vaex-vs-modin-vs-rapids-vs-ray (accessed Feb. 19, 2022).

# ΠΑΡΑΡΤΗΜΑ 1 : Εγκατάσταση Neo4j Community Edition Server

Η παρακάτω βηματική διαδικασία περιγράφει τις διαδικασίες εγκατάστασης ενός Neo4J Server Community Edition v.4.4.4 σε ένα υπολογιστικό σύστημα που τρέχει σε λειτουργικό σύστημα Debian v.11.

1. Το εργαλείο wget είναι ένα εργαλείο παρασκηνίου που είναι απαραίτητο για το κατέβασμα διαφόρων πακέτων λογισμικού από το διαδίκτυο. Υποστηρίζει τα πρωτόκολλα Http, Https, Ftp καθώς και προσπέλαση μέσω διακομιστών μεσολάβησης Http (HTTP Proxies). Αν δεν είναι διαθέσιμο στην τρέχουσα διανομή Linux εγκαθιστούμε το πακέτο wget εκτελώντας την παρακάτω εντολή.

```
sudo apt install wget -y
```

2. Το πακέτο apt (Advanced Packaging Tool) περιλαμβάνει ένα σύνολο από εργαλεία που διευκολύνουν την ανάκτηση, ρύθμιση, εγκατάσταση και απεγκατάσταση λογισμικού σε Debian τύπου διανομές Unix. Ενημερώνουμε τις πληροφορίες των πακέτων λογισμικού και των εξαρτήσεων τους από τις ρυθμισμένες πηγές του διαδικτύου εκτελώντας την παρακάτω εντολή.

```
sudo apt-get update
```

3. Για να δουλέψει ομαλά το ο εξυπηρετητής Neo4j απαιτείται να είναι εγκατεστημένη η έκδοση 11 ενός Java JDK – JavaDevelopmentKit. Επειδή στην τρέχουσα διανομή Linux δεν είναι διαθέσιμη την προσθέτουμε στο εργαλείο διαχείρισης πακέτων apt εκτελώντας την παρακάτω εντολή.

```
sudo apt-get install openjdk-11-jdk -y
```

4. Στην συνέχεια προκειμένου να εξακριβώσουμε τι είδους εκδόσεις Java είναι εγκατεστημένες στο τρέχων μηχάνημα (επιτρέπεται να είναι εγκατεστημένες πολλαπλές εκδόσεις) εκτελούμε την παρακάτω εντολή.

```
sudo update-java-alternatives --list
```

Σε περίπτωση που συνυπάρχουν πολλαπλές εκδόσεις Java επιβάλλεται να ορίσουμε την έκδοση java-1.11.0-openjdk-amd64 ως την εξ ορισμού εκτελώντας την παρακάτω εντολή.

```
sudo update-java-alternatives --jre --set java-1.11.0-openjdk-amd64
```

5. Επιβεβαιώνουμε την εγκατάσταση της σωστής έκδοσης 11 Java εκτελώντας την παρακάτω εντολή.

```
java --version
```

Προσθέτουμε στο εργαλείο διαχείρισης πακέτων apt το πακέτο λογισμικού για την έκδοση v.1.4.4.4 του Neo4j Community Edition εκτελώντας διαδοχικά τις παρακάτω εντολές.

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -
echo 'deb https://debian.neo4j.com stable latest' | sudo tee -a /etc/apt/sources.list.d/neo4j.list
sudo apt-get update
```

Εξακριβώνουμε τι είδους εκδόσεις Neo4j Community Edition είναι διαθέσιμες στο εργαλείο διαχείρισης πακέτων apt εκτελώντας την παρακάτω εντολή.

```
sudo apt list -a neo4j
```

6. Από τις διαθέσιμες εκδόσεις Neo4j μας ενδιαφέρει η πιο πρόσφατη έκδοση 1.4.4.8. Την εγκαθιστούμε εκτελώντας την παρακάτω εντολή.

```
sudo apt-get install neo4j=1:4.4.8 -y
```

7. Εξακριβώνουμε την σωστή εγκατάσταση του Neo4j Community Edition εκτελώντας την παρακάτω εντολή.

```
neo4j --version
```

8. Στην συνέχεια ρυθμίζουμε το αρχείο ρυθμίσεων του Neo4j (/etc/neo4j/neo4j.conf) εκτελώντας την παρακάτω εντολή.

```
sudo nano /etc/neo4j/neo4j.conf
```

8.1. Ενότητα Network Connector Configuration:

8.1.1. Εξ' ορισμού ο Neo4jServer δέχεται μόνο τοπικές συνδέσεις (127.0.0.1 - localhost). Αυτός ο τρόπος λειτουργίας χρησιμοποιείται κυρίως για λόγους εξσφαλμάτωσης και πειραματισμού. Αντίθετα εάν επιθυμούμε να επιτρέπονται και εξωτερικές συνδέσεις με άλλους Servers βγάζουμε την παρακάτω εντολή εκτός σχολίου.

```
dbms.default_listen_address=0.0.0.0
```

8.1.2. Βγάζουμε την παρακάτω εντολή εκτός σχολίου σε περίπτωση που θέλουμε να μην γίνεται αυθεντικοποίηση (μέσω ονόματος χρήστη και συνθηματικού) με κάθε σύνδεση στην βάση, αλλιώς την αφήνουμε όπως έχει.

```
dbms.security.auth_enabled=false
```

8.1.3. Βγάζουμε την παρακάτω εντολή εκτός σχολίου, και της εκχωρούμε ως τιμή την εξωτερική/ δημόσια διεύθυνση του εξυπηρετητή όπου τρέχει η Neo4j βάση δεδομένων. Σε περίπτωση που επιθυμούμε να επιτρέψουμε αποκλειστικά τοπικές συνδέσεις δεν τροποποιούμε την τιμή της.

```
dbms.default_advertised_address=localhost
```

8.1.4. Βγάζουμε εκτός σχολίων τις παρακάτω εντολές που αφορούν τον Http connector.

```
dbms.connector.http.enabled=true
dbms.connector.http.listen_address=:7474
dbms.connector.http.advertised_address=:7474
```

Η πρώτη ενεργοποιεί τον connector, η 2η θέτει την θύρα ακρόασης και κληρονομεί την τιμή που καθορίστηκε στην ρύθμιση `dbms.default_listen_address` και η 3η θέτει την θύρα μέσω της οποίας μπορούν οι χρήστες να προσπελάζουν την βάση και παράλληλα κληρονομεί την τιμή που καθορίστηκε στην ρύθμιση `dbms.default_advertised_address`.

8.1.5. Βγάζουμε εκτός σχολίων τις παρακάτω εντολές που αφορούν τον Bolt connector.

```
dbms.connector.bolt.enabled=true
dbms.connector.bolt.listen_address=:7687
dbms.connector.bolt.advertised_address=:7687
```

Η πρώτη ενεργοποιεί τον connector, η 2η θέτει την θύρα ακρόασης και κληρονομεί την τιμή που καθορίστηκε στην ρύθμιση `dbms.default_listen_address` και η 3η θέτει την θύρα μέσω της οποίας μπορούν οι χρήστες να προσπελάζουν την βάση και παράλληλα κληρονομεί την τιμή που καθορίστηκε στην ρύθμιση `dbms.default_advertised_address`.

8.1.6. Ο https connector είναι απενεργοποιημένος βγάζοντας εκτός σχολίου την παρακάτω εντολή. Επίσης απενεργοποιημένες είναι και όλες οι ρυθμίσεις που αφορούν την πολιτική SSL.

```
dbms.connector.https.enabled=false
```

8.2. **Ενότητα Memory Settings:** Εδώ ρυθμίζουμε την μνήμη σωρού και την ενδιάμεση βοηθητική μνήμη σύμφωνα με τις προτάσεις της εντολής memrectου εργαλείου διαχείρισης neo4j-admin και των βέλτιστων πρακτικών διαχείρισης μνήμης που περιγράφονται στην αντίστοιχη ενότητα της εργασίας. Στα πλαίσια της διπλωματικής η διαθέσιμη μνήμη του μηχανήματος είναι 58 GB. Από αυτά τα 18 GB αποφασίστηκε να διατεθούν για τη εκτέλεση της εφαρμογής πελάτης σε Pythonπου είναι υπεύθυνη για την εισαγωγή των δεδομένων στην βάση δεδομένων γράφου. Τα υπόλοιπα 40 GB αποφασίστηκε να διατεθούν για την εκτέλεση του εξυπηρετητή Neo4j. Για μια αρχική εκτίμηση της κατανομής της μνήμης του μηχανήματος εκτελούμε την παρακάτω εντολή.

```
neo4j-admin memrec --memory=40g --verbose
```

Συγκεκριμένα βγάζουμε εκτός σχολίου και ρυθμίζουμε τις τιμές των παρακάτω εντολών.

```
dbms.memory.heap.initial_size=12g
dbms.memory.heap.max_size=12g
dbms.memory.pagecache.size=22g
```

8.3. **Ενότητα OtherNeo4jsystemproperties:** Απενεργοποιούμε εντελώς τον επαναυπολογισμό του σχεδίου εκτέλεσης κάθε ερωτήματος (Disabling Query Replaning) ρυθμίζοντας τις παρακάτω εντολές.

```
cypher.min_relplan_interval=1
cypher.statistics_divergence_threshold=1
```

Για αποθήκευση των αλλαγών πληκτρολογούμε ctrl + O και Enter. Για την έξοδο από τον συντάκτη πληκτρολογούμε ctrl + X.

9. Εκκινούμε τον εξυπηρετητή βάσης δεδομένων Neo4j ως υπηρεσία εκτελώντας την παρακάτω εντολή.

```
sudo systemctl start neo4j.service
```

10. Ελέγχουμε την κατάσταση του εξυπηρετητή βάσης δεδομένων Neo4j εκτελώντας την παρακάτω εντολή.

```
sudo systemctl status neo4j.service
```

11. Οποτεδήποτε απαιτηθεί μπορούμε να κάνουμε τις απαιτούμενες ρυθμίσεις στο αρχείο ρυθμίσεων neo4j.conf. Σε περίπτωση που η υπηρεσία είναι ήδη ενεργοποιημένη για να ενεργοποιηθούν οι αλλαγές που πραγματοποιήσαμε θα πρέπει να ξανά εκκινήσουμε την υπηρεσία. Η επανεκκίνηση της υπηρεσίας πραγματοποιείται εκτελώντας την παρακάτω εντολή.

```
sudo systemctl restart neo4j.service
```

12. Τερματίζουμε τον εξυπηρετητή βάσης δεδομένων Neo4j εκτελώντας την παρακάτω εντολή.

```
sudo systemctl stop neo4j.service
```

13. Τέλος οποτεδήποτε απαιτηθεί να επανακτήσουμε πρόσβαση στην γραμμή εντολών πληκτρολογούμε ctrl + C.

# ΠΑΡΑΡΤΗΜΑ 2 : Μαζική αντιγραφή Αρχείων από Google Cloud Storage σε Google Drive

Στα πλαίσια της διπλωματικής εργασίας παρουσιάστηκε η ανάγκη αντιγραφής του συνόλου της ιεραρχικά εμφωλευμένης δομής καταλόγων των δεδομένων μας από τον κουβά του googleCloudStorage σε μια αντίστοιχη ιεραρχικά εμφωλευμένη δομή καταλόγων στο Google Drive. Η μεταφορά αυτή έπρεπε να γίνει με ιδιαίτερη προσοχή γιατί δημιουργεί εξερχόμενη κίνηση (EgressTraffic) από τον κουβά του GCS και σε ορισμένες περιπτώσεις χρεώνεται ακριβά. Συγκεκριμένα σε οποιαδήποτε εξερχόμενη κίνηση προκαλείται από οποιονδήποτε υπολογιστή εκτός της υποδομής υπολογιστικού νέφους της Google επιβάλλονται χρεώσεις. Επίσης χρεώσεις επιβάλλονται και σε οποιαδήποτε εξερχόμενη χρέωση προκαλείται μεταξύ διαφορετικών περιοχών (ηπείρων) εντός της υποδομής υπολογιστικού νέφους της Google. Λαμβάνοντας υπόψιν το μεγάλο

μέγεθος των αρχείων που έπρεπε να μεταφερθούν θα είχε ως αποτέλεσμα ένα σημαντικό κόστος. Γενικά η αντιγραφή μεγάλου όγκου δεδομένων από το Google Cloud Storage στο GoogleDrive, μπορεί να γίνει μέσω του εργαλείου της γραμμής εντολών gsutil και περιλαμβάνει δυο στάδια: Μεταφορά από το GCS – Google Cloud Storage στο GCE – Google Cloud Engine και μεταφορά από το GCE – Google Cloud Engine στο Google Drive. Στο πρώτο στάδιο ως Google Cloud Engine μπορεί να χρησιμοποιηθεί ένα εικονικό μηχάνημα της υποδομής υπολογιστικού νέφους της Google. Σε αυτό απαιτείται εγκατάσταση της python και του gcloud. Εναλλακτικά μπορεί να χρησιμοποιηθεί και ένα Google Colab Notebook που η χρήση του είναι δωρεάν και δεν απαιτεί επιπλέον παραμετροποίηση. Στα πλαίσια της διπλωματικής επιλέχθηκε η χρήση ενός Google Colab Notebook που είναι προσπελάσιμο μέσω του συνδέσμου https://colab.research.google.com/ και απλά απαιτεί βασική αυθεντικοποίηση μέσω ενός έγκυρου λογαριασμού Gmail. Σε αυτό εκτελέστηκαν οι παρακάτω εντολές.

```
from google.colab import drive
drive.mount('/content/drive')
from google.colab import auth
auth.authenticate_user()


project_id = 'dimiourgia-postgraduate-thesis'
!gcloud config set project {project_id}
!gsutil ls
bucket_name = 'postgraduate_thesis_input_bucket'
dataset_source = 'mag'
data_filename = 'papers'
folder_number = 16


!gsutil -m cp
gs://{bucket_name}/datasets/{dataset_source}/{data_filename}/{data_filename}_{fo
lder_number}/*
/content/drive/MyDrive/datasets/{dataset_source}/{data_filename}/{data_filename}
_{folder_number}
```

Στο σημείο αυτό κρίνεται σκόπιμο να επισημάνουμε ότι η αντιγραφή έγινε σταδιακά ανά σύνολο δεδομένων (Mag, Aminer), όνομα αρχείων (papers, authors, venues, affiliations) και αριθμό φακέλου. Συνεπώς ο χρήστης πρέπει να αλλάζει τις μεταβλητές dataset_source , data_filename και folder_number ανάλογα με το κομμάτι της ιεραρχικής δομής καταλόγων που θέλει να αντιγράψει. Επίσης επισημαίνεται η χρήση της παραμέτρου -m στην εντολή gsutil. Αυτό έγινε ώστε να εκμεταλλευθούμε την παραλληλοποιημένη αντιγραφή πολλαπλών αρχείων επιταχύνοντας σημαντικά τον χρόνο εκτέλεσης της διαδικασίας. Τέλος επιβάλλεται να αναφέρουμε ότι κατά την αντιγραφή πολλαπλών αρχείων κάποιες φορές διακόπηκε η λήψη κάποιων μεμονωμένων αρχείων. Σε αυτές τις περιπτώσεις στον κατάλογο προορισμού του google Drive εμφανίστηκε ένα μερικώς κατεβασμένο αρχείο με επέκταση *.gstmp. Σε αυτές τις περιπτώσεις απαιτείται η μεμονωμένη αντιγραφή των

συγκεκριμένων αρχείων εκτελώντας την παρακάτω εντολή, φροντίζοντας κάθε φορά να αλλάζουμε το όνομα του αρχείου που θέλουμε να αντιγραφεί.

```
!gsutil cp gs://{bucket_name}/datasets/{dataset_source}/{data_filename}/{data_filename}_{folder_number}/papers_382.txt.lz4 /content/drive/MyDrive/datasets/{dataset_source}/{data_filename}/{data_filename}_{folder_number}
```

Στο δεύτερο στάδιο η εξερχόμενη κίνηση που δημιουργείται από το Google Cloud Engine προς το Google Drive είναι δωρεάν. Στο στάδιο αυτό δεν απαιτείται κάποια επιπλέον ενέργεια μιας και έχουμε ήδη προσαρτήσει το Google Drive (Google Drive Mount) στην τοπική δομή καταλόγων του Google Colab notebook. Συνεπώς τα αρχεία έχουν ήδη αντιγραφεί στον αντίστοιχο φάκελο του Google Drive.

# ΠΑΡΑΡΤΗΜΑ 3 : Εγκατάσταση Βιβλιοθηκών Επέκτασης Neo4j APOC & GDS

**Εγκατάσταση βιβλιοθήκης APOC - Awesome Procedures on Cypher**

10. Στο αρχείο ρυθμίσεων του Neo4j /etc/neo4j/neo4j.conf βγάζουμε εκτός σχολίου την παρακάτω εντολή.

```
dbms.directories.plugins=/var/lib/neo4j/plugins
```

11. Στο ίδιο αρχείο θέτουμε τις παρακάτω δυο εντολές.

```
dbms.security.procedures.unrestricted=apoc.*

dbms.security.procedures.allowlist=apoc.coll.*,apoc.load.*,apoc.*
```

12. Στην συνέχεια θα πρέπει να κατεβάσουμε στον φάκελο /var/lib/neo4j/plugins το jar αρχείο τις βιβλιοθήκης APOC. Επειδή η βιβλιοθήκη κάνει χρήση των εσωτερικών API'stου Neo4j θα πρέπει τα δυο πακέτα να είναι συμβατά. Συνεπώς θα πρέπει τα δυο πρώτα ψηφία των αριθμών εκδόσεων τις να είναι τα ίδια. Θα χρησιμοποιήσουμε την Core έκδοση τις βιβλιοθήκης. Συνεπώς στο command prompt εισάγουμε την παρακάτω εντολή.

```
sudo wget https://github.com/neo4j-contrib/neo4j-apoc-procedures/releases/download/4.4.0.2/apoc-4.4.0.2-core.jar -P /var/lib/neo4j/plugins
```

13. Στην συνέχεια εισάγουμε τις παρακάτω δυο εντολές.

```
sudo chown neo4j:neo4j /var/lib/neo4j/plugins/apoc-4.4.0.2-core.jar

sudo chmod 755 /var/lib/neo4j/plugins/apoc-4.4.0.2-core.jar
```

Στην συνέχεια εκκινούμε την υπηρεσία (ή την επανεκκινούμε σε περίπτωση που είχε ήδη ξεκινήσει) και μεταβαίνουμε στην διεπαφή σύνδεσης με την Neo4j βάση δεδομένων πληκτρολογώντας http://localhost:7474/browser/ στην γραμμή διευθύνσεων του διαφυλλιστή. Στην συνέχεια παρέχοντας τα διαπιστευτήριά τις κάνουμε κλικ στο κουμπί Connect για να συνδεθούμε απομακρυσμένα στην βάση δεδομένων γράφου. Εκεί επιβεβαιώνουμε την ορθή εγκατάσταση τις βιβλιοθήκης εισάγοντας την παρακάτω εντολή.

```
call apoc.help('apoc')
```

Η παραπάνω εντολή εμφανίζει την λίστα όλων των εγκατεστημένων διαδικασιών και συναρτήσεων. Ισοδύναμα ελέγχουμε την εγκατεστημένη έκδοση της βιβλιοθήκης εκτελώντας την παρακάτω συνάρτηση.

```
RETURN apoc.version();
```

**Εγκατάσταση βιβλιοθήκης GDS – Graph Data Science**

1. Στο αρχείο ρυθμίσεων του Neo4j /etc/neo4j/neo4j.conf βγάζουμε εκτός σχολίου την παρακάτω εντολή.

```
dbms.directories.plugins=/var/lib/neo4j/plugins
```

2. Στο ίδιο αρχείο θέτουμε τις παρακάτω δυο εντολές.

```
dbms.security.procedures.unrestricted=apoc.*,gds.*
dbms.security.procedures.allowlist=apoc.coll.*,apoc.load.*,apoc.*,gds.*
```

Στην συνέχεια θα πρέπει να κατεβάσουμε στον φάκελο /var/lib/neo4j/plugins το jar αρχείο τις βιβλιοθήκης APOC. Επειδή η βιβλιοθήκη κάνει χρήση των εσωτερικών API's του Neo4j θα πρέπει τα δυο πακέτα να είναι συμβατά. Συνεπώς θα πρέπει τα δυο πρώτα ψηφία των αριθμών εκδόσεων τις να είναι τα ίδια. Συνεπώς στο command prompt εισάγουμε την παρακάτω εντολή προκειμένου να κατεβάσουμε στον κατάλογο /var/lib/neo4j/plugins την συμπιεσμένη έκδοση v.2.0.4 της βιβλιοθήκης GDS– Graph Data Science.

```
sudo wget https://graphdatascience.ninja/neo4j-graph-data-science-2.0.4.zip -P /var/lib/neo4j/plugins
```

Σε περίπτωση που το περιβάλλον Linux δεν αναγνωρίζει την εντολή unzip την εγκαθιστούμε εκτελώντας την παρακάτω εντολή.

```
sudo apt-get install unzip
```

Στην συνέχεια μεταβαίνουμε στον φάκελο /var/lib/neo4j/plugins/και εκτελούμε την

```
sudo unzip neo4j-graph-data-science-2.0.4.zip
```

παρακάτω εντολή προκειμένου να αποσυμπιέσουμε το Jar αρχείο που κατεβάσαμε προηγουμένως.

Τέλος εκτελούμε την παρακάτω εντολή προκειμένου να διαγράψουμε το zip αρχείο που κατεβάσαμε προηγουμένως στον φάκελο /var/lib/neo4j/plugins.

3. Στην συνέχεια εισάγουμε τις παρακάτω δυο εντολές.

```
sudo chown neo4j:neo4j /var/lib/neo4j/plugins/neo4j-graph-data-science-2.0.4.jar
sudo chmod 755 /var/lib/neo4j/plugins/neo4j-graph-data-science-2.0.4.jar
```

Στην συνέχεια εκκινούμε την υπηρεσία (ή την επανεκκινούμε σε περίπτωση που είχε ήδη ξεκινήσει) και μεταβαίνουμε στην διεπαφή σύνδεσης με την Neo4j βάση δεδομένων πληκτρολογώντας http://localhost:7474/browser/ στην γραμμή διευθύνσεων του διαφυλλιστή. Στην συνέχεια παρέχοντας τα διαπιστευτήριά τις κάνουμε κλικ στο κουμπί Connect για να συνδεθούμε απομακρυσμένα στην βάση δεδομένων γράφου. Εκεί επιβεβαιώνουμε την ορθή εγκατάσταση τις βιβλιοθήκης εισάγοντας την παρακάτω εντολή.

```
CALL gds.list()
```

Η παραπάνω εντολή εμφανίζει την λίστα όλων των εγκατεστημένων αλγορίθμων γράφου. Ισοδύναμα ελέγχουμε την εγκατεστημένη έκδοση της βιβλιοθήκης εκτελώντας την παρακάτω συνάρτηση.

```
RETURN gds.version()
```

# ΠΑΡΑΡΤΗΜΑ 4 : Στρατηγική δημιουργίας & επαναφοράς αντιγράφων ασφαλείας στο Neo4j CE

Σε οποιοδήποτε περιβάλλον λειτουργίας ενός συστήματος (Παραγωγής, Ανάπτυξης) απαιτείται να προσδιοριστεί μια ολοκληρωμένη στρατηγική δημιουργίας αντιγράφων ασφαλείας. Αυτήν θα πρέπει να συμπληρώνεται και από μια στρατηγική επαναφοράς ενός αντιγράφου ασφαλείας σε μια προγενέστερη ασφαλή κατάσταση. Μια τέτοια στρατηγική είναι πολύ σημαντική γιατί αποτρέπει την υποβάθμιση της ακεραιότητας (Integrity) και διαθεσιμότητας (Availability) των δεδομένων της βάσης δεδομένων από απρόβλεπτα καταστροφικά γεγονότα που μπορεί να λάβουν χώρα. Επίσης διευκολύνει την εκτέλεση συνηθισμένων εργασιών διαχείρισης, όπως μετάπτωση μιας βάσης δεδομένων από ένα στιγμιότυπο σε ένα άλλο, αναβάθμιση της έκδοσης του συστήματος Neo4j, ανακατανομή του διαθέσιμου χώρου αποθήκευσης κ.α. Το Neo4j προσφέρει πολλές επιλογές δημιουργίας αντιγράφων ασφαλείας και επαναφοράς τους. Δεν υπάρχει μια καθολική στρατηγική που εφαρμόζεται σε όλες τις περιπτώσεις χρήσης. Αντίθετα η επιλογή της ποιο κατάλληλης στρατηγικής αποφασίζεται κατά περίπτωση και επηρεάζεται από πληθώρα παραγόντων. Σε αυτούς συγκαταλέγονται ο τύπος εγκατάστασης, ο όγκος των δεδομένων, η αποδεκτή χρονική περίοδος μη διαθεσιμότητας της βάσης, το είδος των αντιγράφων ασφαλείας που επιθυμούμε να δημιουργήσουμε (πλήρη, αυξητικά, απομακρυσμένα) κ.α. Κάποιες από αυτές τις δυνατότητες είναι διαθέσιμες και στις δυο βασικές εκδόσεις του Neo4j, ενώ κάποιες άλλες αποκλειστικά στην εμπορική έκδοση. Υπενθυμίζουμε ότι στα πλαίσια της διπλωματικής έχει επιλεγεί να χρησιμοποιηθεί ένα αυτόνομο στιγμιότυπο Neo4j Server

CE που τρέχει σε ένα εικονικό μηχάνημα της σχολής σε περιβάλλον Debian-Linux. Συνεπώς η στρατηγική δημιουργίας και επαναφοράς αντιγράφων ασφαλείας που επιλέξαμε να χρησιμοποιήσουμε περιλαμβάνει τα εξής. Μπορούμε να δημιουργήσουμε μόνο πλήρη αντίγραφα ασφαλείας της τρέχουσας εξ' ορισμού βάσης δεδομένων neo4j.db. Υπενθυμίζουμε ότι η συγκεκριμένη βάση ορίστηκε ως η εξ' ορισμού βάση δεδομένων στο αρχείο ρυθμίσεων etc/neo4j/neo4j.conf. Μαζί με την εξ' ορισμού βάση δεδομένων επιβάλλεται να δημιουργήσουμε ένα αντίγραφο ασφαλείας και για την βάση δεδομένων system. Πρόκειται για την βάση δεδομένων που περιέχει χρήσιμες πληροφορίες διαμόρφωσης για όλες τις βάσεις δεδομένων που διαχειρίζεται το τρέχων σύστημα διαχείρισης βάσεων δεδομένων. Σε αυτές συγκαταλέγονται πληροφορίες κατάστασης λειτουργίας, διαμορφώσεις ασφάλειας, ορισμούς σχημάτων, ρόλοι χρηστών για όλες τις υποκείμενες βάσεις δεδομένων. Ο χρήστης συνιστάται να περιορίζει την αλληλεπίδρασή του με αυτήν την βάση δεδομένων αποκλειστικά για σκοπούς άντλησης πληροφοριών για το σύστημα του Neo4j. Τα αντίγραφα ασφαλείας έχουν την μορφή αρχείων αρχειοθέτησης. Για λόγους αυξημένης ασφαλείας των αντιγράφων ασφαλείας συνιστάται τα αρχεία των αντιγράφων ασφαλείας να αποθηκεύονται σε ένα απομακρυσμένο μέσο. Αποδεκτές επιλογές θα μπορούσαν να είναι ο σκληρός δίσκος ενός διαφορετικού υπολογιστικού συστήματος, ένας λογαριασμός Google Drive/ Storage κ.α. Κατά την διάρκεια της διαδικασίας το σύστημα διαχείρισης της βάσης δεδομένων Neo4j πρέπει να είναι κλειστό. Τόσο η διαδικασία δημιουργίας αντιγράφου ασφαλείας μιας βάσης, όσο και η φάση επαναφοράς της πραγματοποιούνται με την εκτέλεση κατάλληλων εντολών του εργαλείου διαχείρισης neo4j-admin. Συγκεκριμένα η εντολή neo4j-admin dump δημιουργεί ένα αντίγραφο ασφαλείας, ενώ η εντολή neo4j-admin load επαναφέρει το αντίγραφο ασφαλείας. Και οι δυο αυτές εντολές πρέπει να εκτελεστούν από τον χρήστη neo4j, ώστε να μην υπάρχουν προβλήματα με τα δικαιώματα πρόσβασης διαφόρων αρχείων. Τέλος ανεξάρτητα από τα αρχεία της βάσης δεδομένων επιβάλλεται να δημιουργηθούν αντίγραφα ασφαλείας και για μια πληθώρα άλλων αρχείων. Σε αυτά συγκαταλέγονται το αρχείο ρυθμίσεων etc/neo4j/neo4j.conf, όλα τα αρχεία που σχετίζονται με την SSL κρυπτογράφηση καθώς και τα αρχεία των πρόσθετων (plugins) που τυχόν χρησιμοποιήθηκαν στην τρέχουσα εγκατάσταση. Στα πλαίσια της διπλωματικής δημιουργήσαμε αντίγραφο ασφαλείας για το αρχείο ρυθμίσεων και τον φάκελο πρόσθετα (plugins) μιας και εγκαταστήσαμε τις βιβλιοθήκες APOC και GDS.

1. Λαμβάνοντας υπόψιν τα παραπάνω μεταβαίνουμε στη γραμμή εντολών του εικονικού μηχανήματος στο οποίο τρέχει ο εξυπηρετητής βάσης δεδομένων Neo4j και σταματάμε το σύστημα διαχείρισης βάσεων δεδομένων Neo4j εκτελώντας την παρακάτω εντολή.

```
sudo /etc/init.d/neo4j stop
```

2. Στην συνέχεια δημιουργούμε τον κατάλογο αποθήκευσης των αντιγράφων ασφαλείας εκτελώντας την παρακάτω εντολή.

```
mkdir data/dumps/neo4j
```

3. Αμέσως μετά αλλάζουμε τα δικαιώματα των φακέλων dumps και neo4j που δημιουργήσαμε παραπάνω εκτελώντας τις παρακάτω εντολές.

```
sudo chmod 777 /home/kostasvav/data/dumps
sudo chmod 777 /home/kostasvav/data/dumps/neo4j
```

4. Στην συνέχεια συνδεόμαστε ως χρήστης neo4j εκτελώντας την παρακάτω εντολή.

```
sudo su - neo4j
```

5. Προκειμένου να δημιουργήσουμε ένα αντίγραφο ασφαλείας για την βάση δεδομένων neo4j.db εκτελούμε την παρακάτω εντολή.

```
/usr/bin/neo4j-admin dump --database=neo4j.db --
to=/home/kostasvav/data/dumps/neo4j/neo4j.db.dump
```

6. Προκειμένου να δημιουργήσουμε ένα αντίγραφο ασφαλείας για την βάση δεδομένων system εκτελούμε την παρακάτω εντολή.

```
/usr/bin/neo4j-admin dump --database=system --
to=/home/kostasvav/data/dumps/neo4j/system.dump
```

7. Στην συνέχεια αποσυνδεόμαστε ως χρήστης neo4j. Αντιγράφουμε το αρχείο ρυθμίσεων etc/neo4j/neo4j.conf στον φάκελο /home/kostasvav/data/dumps/neo4j/ εκτελώντας την παρακάτω εντολή.

```
cp /etc/neo4j/neo4j.conf /home/kostasvav/data/dumps/neo4j/
```

8. Αντιγράφουμε τα περιεχόμενα του φακέλου /var/lib/neo4j/plugins στον φάκελο /home/kostasvav/data/dumps/neo4j/ εκτελώντας την παρακάτω εντολή.

```
cp -r /var/lib/neo4j/plugins/
/home/kostasvav/data/dumps/neo4j/plugins/
```

Πλέον ο φάκελος home/kostasvav/data/dumps/neo4j/ περιέχει οτιδήποτε χρειάζεται να αποθηκευτεί σε ένα απομακρυσμένο μέσο αποθήκευσης, ώστε να διαφυλαχτεί από απρόβλεπτα καταστροφικά γεγονότα. Στο σημείο αυτό αξίζει να παρατηρήσουμε ότι κάθε αντίγραφο βάσης δεδομένων καταλαμβάνει μικρότερο χώρο αποθήκευσης από την αρχική βάση δεδομένων.