



ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

Διεθνές Πανεπιστήμιο Ελλάδος

Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Scientometrics: Σχεδιασμός και Υλοποίηση Βιβλιομετρικής βάσης μεγάλων δεδομένων

Φοιτητής:

Δημήτριος Μάριος Σφούγγαρης
Αριθμός Μητρώου: 144214

Επιβλέπων:

Αντώνης Σιδηρόπουλος

16 Δεκεμβρίου 2024

Αφιέρωση

Για τη διεκπεραίωση της πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω την μητέρα μου και την σύντροφο μου για την στήριξη και τις θυσίες τους καθ' όλη την διάρκεια των σπουδών μου.

Πρόλογος

Στην σημερινή κοινωνία η σημασία της ανάλυσης δεδομένων επιστημονικής παραγωγής καθώς και της αυξανόμενης ανάγκης για δικτύωση της γνώσης αποτέλεσαν τον λόγο που επέλεξα να ασχοληθώ με τη ανάλυση της βιβλιομετρικής βάσης δεδομένων OpenAlex και τη δημιουργία βάσης δεδομένων γραφημάτων Neo4j. Στη σύγχρονη ερευνητική κοινότητα, οι βιβλιομετρικές βάσεις δεδομένων προσφέρουν πολύτιμα εργαλεία για τη χαρτογράφηση της επιστημονικής προόδου, ενώ οι βάσεις δεδομένων γραφημάτων, όπως το Neo4j, επιτρέπουν την απεικόνιση των σχέσεων μεταξύ δεδομένων με έναν καινοτόμο και αποδοτικό τρόπο. Η συγκεκριμένη πτυχιακή εργασία μου έδωσε τη δυνατότητα να συνδυάσω τις γνώσεις μου στην ανάλυση δεδομένων και την επιστήμη δεδομένων με τη δύναμη των γραφημάτων, προσφέροντάς μου πολύτιμη εμπειρία σε σύγχρονες τεχνολογίες ανάλυσης δεδομένων. Μέσα από την διαδικασία υλοποίησης του θέματος, κατανόησα βαθύτερα τις προκλήσεις που προκύπτουν κατά τη μετατροπή βιβλιομετρικών δεδομένων σε γραφήματα, καθώς και τις τεχνικές για τη βελτιστοποίηση της αποδοτικότητάς τους. Επίσης, είχα την ευκαιρία μέσα από την εκμάθηση της Neo4j να καταφέρω να εφαρμόσω προηγμένες μεθόδους οπτικοποίησης και ανάλυσης δεδομένων που συμβάλλουν στη βελτίωση της κατανόησης των επιστημονικών σχέσεων και συνεργασιών. Συνολικά, αυτή η εργασία μου προσέφερε πολύτιμη γνώση για τη χρήση και την εφαρμογή προηγμένων τεχνολογιών σε πραγματικά ερευνητικά προβλήματα, ενώ παράλληλα ανέδειξε την πρακτική αξία της βιβλιομετρίας και της ανάλυσης γραφημάτων.

Abstract

The purpose of this thesis is to implement and automate the transformation of bibliometric data from the OpenAlex database into a suitable format for importing into a graph database such as Neo4j. The bibliometric data from OpenAlex is complex and diverse, as it includes different types of entities and relationships, with the database spanning several terabytes of data. The insertion of this data into a graph database occurs after necessary intermediate steps, such as transformation and filtering, are completed. The first step in this thesis was an extensive study of the two main components of the project, which were the bibliometric OpenAlex database and the Neo4j graph database. During the analysis and implementation of the application, various data quality issues were identified, which were addressed with appropriate solutions. Throughout the analysis, decisions were made regarding the approach that would best serve the purpose of the project. The careful adjustment of filtering parameters, as well as the selection of suitable commands for bulk data import into Neo4j, proved to be a demanding and time-consuming process. Each decision and parameter, both in the commands and the database and the code, required detailed analysis and understanding of their interdependencies and the potential impact on the overall functioning of the database and the data import process. The choices and decisions made validated the methodology, as demonstrated by the final result and the proper operation of the database's graph structures.

Keywords: graph database, Neo4j, bibliographic datasets, bibliometric datasets, import, Neo4j admin import, incremental, Neo4j desktop, graphs, Cypher, Cypher queries, parallel&distributed processing execution, bibliographic data management infrastructure, data filtering, code modification, import command parameterization, Neo4j parameterization

Περίληψη

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η υλοποίηση και αυτοματοποίηση της μετατροπής των βιβλιομετρικών δεδομένων της βάσης OpenAlex σε κατάλληλη μορφή, ικανή για εισαγωγή σε μια βάση γραφημάτων όπως η Neo4j. Τα βιβλιομετρικά δεδομένα της OpenAlex είναι πολυσύνθετα και ποικιλόμορφα, καθώς περιλαμβάνουν διαφορετικούς τύπους οντοτήτων και σχέσεων, με τη βάση να εκτείνεται σε αρκετά terabytes δεδομένων. Η εισαγωγή αυτών των δεδομένων, αφού περάσουν από τα απαραίτητα ενδιάμεσα στάδια, όπως η μετατροπή και το φιλτράρισμα, γίνεται σε μία βάση γραφημάτων. Αρχικό βήμα της πτυχιακής ήταν η εκτενής μελέτη των δύο βασικών πυλώνων της εργασίας, τα οποία ήταν η βιβλιομετρική βάση OpenAlex και η βάση γραφημάτων Neo4j. Κατά την ανάλυση και υλοποίηση της εφαρμογής, εντοπίστηκαν διάφορα ζητήματα ποιότητας των δεδομένων, τα οποία αντιμετωπίστηκαν με κατάλληλες λύσεις. Στη διάρκεια της ανάλυσης, λήφθηκαν αποφάσεις ως προς τη μέθοδο προσέγγισης που θα εξυπηρετούσε τον στόχο της εργασίας. Η προσεκτική προσαρμογή των παραμέτρων φιλτραρίσματος, καθώς και η επιλογή των κατάλληλων εντολών για μαζική εισαγωγή των δεδομένων στη Neo4j, αποδείχθηκε μια απαιτητική και χρονοβόρα διαδικασία. Κάθε επιλογή και κάθε παράμετρος, τόσο στις εντολές όσο και στη βάση και τον κώδικα, απαιτούσε λεπτομερή ανάλυση και κατανόηση των αλληλεξαρτήσεων και των πιθανών επιπτώσεών τους στη συνολική λειτουργία της βάσης και στη διαδικασία εισαγωγής δεδομένων. Οι επιλογές και οι αποφάσεις που ελήφθησαν δικαίωσαν τη μεθοδολογία, όπως φάνηκε από το τελικό αποτέλεσμα και τη σωστή λειτουργία των γραφημάτων της βάσης.

Λέξεις Κλειδιά: βάση δεδομένων γράφου, neo4j, βιβλιογραφικά σύνολα δεδομένων, βιβλιομετρικά σύνολα δεδομένων, import, neo4j admin import, incremental, neo4j desktop, γραφήματα, cypher, cypher queries, παραλληλοποιήσιμη και κατανεμημένη εκτέλεση επεξεργασιών, υποδομή διαχείρισης βιβλιογραφικών δεδομένων, φιλτράρισμα δεδομένων, τροποποίηση κώδικα, παραμετροποίηση εντολών εισαγωγής, παραμετροποίηση neo4j.

Περιεχόμενα

1	Εισαγωγή	1
1.1	Περιγραφή συνόλων δεδομένων	1
1.2	Αντικείμενο της Πτυχιακής Εργασίας	2
1.3	Διάρθρωση της Εργασίας	2
1.4	Επίλογος	5
2	Εισαγωγή στα Βιβλιομετρικά Datasets και ανάλυση της πλατφόρμας OpenAlex	6
2.1	Εισαγωγή στα Βιβλιομετρικά Δεδομένα	6
2.1.1	Ορισμός των Βιβλιομετρικών Δεδομένων	6
2.1.2	Ιστορική Εξέλιξη και Σημαντικότητα στη Σύγχρονη Έρευνα	6
2.1.3	Βασικοί Τύποι Βιβλιομετρικών Δεδομένων	7
2.2	Κύριες Πηγές Βιβλιομετρικών Δεδομένων	7
2.2.1	Παραδοσιακές Πηγές Βιβλιομετρικών Δεδομένων	7
2.2.2	Ανοιχτές Πηγές Βιβλιομετρικών Δεδομένων	8
2.2.3	Σύγκριση μεταξύ Παραδοσιακών και Ανοιχτών Πηγών	8
2.3	Η Πλατφόρμα OpenAlex	9
2.3.1	Ιστορικό και Εξέλιξη της OpenAlex	9
2.3.2	Στόχοι και Φιλοσοφία της Πλατφόρμας	9
2.3.3	Δομή του Dataset της OpenAlex	10
2.4	Τεχνική Ανάλυση της OpenAlex	12
2.4.1	Δομή και Αρχιτεκτονική της OpenAlex	12
2.4.2	API της OpenAlex	13
2.4.3	Πρόσβαση στα Δεδομένα της OpenAlex	14
2.5	Σύγκριση της OpenAlex με Άλλες Βάσεις Δεδομένων	14
2.5.1	Ανάλυση και Σύγκριση με Scopus και Web of Science	14
2.5.2	Πλεονεκτήματα και Μειονεκτήματα της OpenAlex	14
2.5.3	Εφαρμογές και Πρακτικά Παραδείγματα Χρήσης	15
2.6	Βιβλιομετρικά Δεδομένα και Ηθικές Σκέψεις	16
2.6.1	Ηθικά Ζητήματα στη Χρήση Βιβλιομετρικών Δεδομένων	16
2.6.2	Ζητήματα Διαφάνειας και Ανοιχτής Πρόσβασης	16
2.6.3	Επιπτώσεις των Βιβλιομετρικών Αναλύσεων στην Επιστημονική Κοινότητα	16
2.7	Προοπτικές και Μελλοντικές Κατευθύνσεις	17
2.7.1	Αναδυόμενες Τεχνολογίες και ο Ρόλος της OpenAlex	17
2.7.2	Προκλήσεις και Ευκαιρίες για τα Βιβλιομετρικά Δεδομένα	17

2.7.3	Επεκτάσεις, Μελλοντικές Βελτιώσεις της Πλατφόρμας OpenAlex και τωρινοί περιορισμοί	17
3	Εισαγωγή στη Neo4j	19
3.1	Εισαγωγή	19
3.2	Δομή και Τεχνικά Χαρακτηριστικά της Neo4j	20
3.2.1	Αρχιτεκτονική Λογισμικού της Neo4j	20
3.2.2	Δομή των Κόμβων, των Ακμών και των Ιδιοτήτων	21
3.2.3	Αποθήκευση Δεδομένων στη Neo4j	22
3.2.4	Βελτιστοποίηση Απόδοσης Ερωτημάτων	22
3.2.5	Ασφάλεια και Διαχείριση Δεδομένων	23
3.2.6	Επεκτασιμότητα και Απόδοση της Neo4j	23
3.2.7	Διαχείριση Συναλλαγών και Συνέπεια Δεδομένων	24
3.2.8	Σύγκριση της Neo4j με Άλλες Βάσεις Δεδομένων	24
3.2.9	Σύγκριση με Σχεσιακές Βάσεις Δεδομένων (SQL)	24
3.2.10	Σύγκριση με Μη Σχεσιακές Βάσεις Δεδομένων (NoSQL)	26
3.3	Εφαρμογή της Neo4j στον Σχεδιασμό και την Υλοποίηση Βιβλιομετρικής Βάσης Δεδομένων	27
3.3.1	Αντιστοίχιση Βιβλιομετρικών Δεδομένων στη Δομή Γραφημάτων	27
3.3.2	Ανάλυση και Οπτικοποίηση Δεδομένων	28
3.3.3	Συμπέρασμα Χρήσης της Neo4j για Βιβλιομετρικές Βάσεις Δεδομένων	28
3.4	Πλεονεκτήματα Χρήσης της Neo4j για τον Σχεδιασμό και την Υλοποίηση Βιβλιομετρικής Βάσης Μεγάλων Δεδομένων	29
3.4.1	Δομή Δεδομένων που Αντιστοιχεί Φυσικά στα Βιβλιομετρικά Δεδομένα	29
3.4.2	Επεξεργασία και Ανάλυση Μεγάλων Συνόλων Δεδομένων σε Πραγματικό Χρόνο	29
3.4.3	Ικανότητα Διαχείρισης Πολυπλοκότητας και Πολλαπλών Σχέσεων	29
3.4.4	Υποστήριξη Σύνθετων Ερωτημάτων και Αναλύσεων	30
3.4.5	Επεκτασιμότητα και Προσαρμοστικότητα της Βάσης Δεδομένων για Μελλοντικές Ανάγκες	30
4	Εισαγωγή στην Python και στην γλώσσα ερωταποκρίσεων Cypher	31
4.1	Εισαγωγή στην Python	31
4.1.1	Ιστορία και εξέλιξη της Python	31
4.1.2	Βασικά χαρακτηριστικά της Python που την καθιστούν κατάλληλη για ανάλυση δεδομένων	32
4.1.3	Επισκόπηση των Βασικών Βιβλιοθηκών της Python	33
4.1.4	Python και Neo4j	40
4.2	Εισαγωγή στη γλώσσα Cypher	41
4.2.1	Βασικά Χαρακτηριστικά της Cypher	41
4.2.2	Βασική σύνταξη και δομή της Cypher	42
4.2.3	Πλεονεκτήματα της χρήσης της Cypher	42
4.2.4	Εφαρμογές και χρήση της Cypher	43
4.2.5	Τρόποι εισαγωγής δεδομένων στη Neo4j	43
4.3	Εισαγωγή δεδομένων μέσω Neo4j Admin Import	44

4.3.1	Τι είναι το Neo4j Admin Import	44
4.3.2	Πλεονεκτήματα του Neo4j Admin Import	44
4.3.3	Μειονεκτήματα του Neo4j Admin Import	45
4.3.4	Σύγκριση Cypher και Neo4j Admin Import για εισαγωγή δεδομένων	45
4.4	Συμπεράσματα	45
5	Τεχνική ανάλυση της Εφαρμογής	46
5.1	Εισαγωγή	46
5.2	Σύντομη επισκόπηση της Εφαρμογής	46
5.3	Γενικές Αρχές και Απαιτήσεις	48
5.3.1	Υλικό και Λογισμικό	48
5.3.2	Αρχικά Βήματα Ρύθμισης σε AWS και Τοπικά Περιβάλλοντα	49
5.3.3	Neo4j και Διαδικασία Εισαγωγής Δεδομένων	49
5.4	Ανάκτηση Δεδομένων από το OpenAlex μέσω AWS CLI	55
5.4.1	Περιγραφή των Εντολών AWS S3 για τη Λήψη Δεδομένων	56
5.4.2	Επεξήγηση της Διαδικασίας Φιλτραρίσματος με Χρήση AWS CLI	56
5.4.3	Επισκόπηση της Αποσυμπίεσης και Μορφοποίησης Δεδομένων	56
5.4.4	Σημασία της Προεπεξεργασίας στο Data Pipeline	57
5.5	Εισαγωγή και Διαχείριση Δεδομένων στη Neo4j	57
5.5.1	Τρόποι εισαγωγής δεδομένων στην Neo4j	57
5.5.2	Ανάλυση του script	59
5.5.3	Μετατροπή από JSON σε CSZ Script (json2csv_neo4j.py)	63
5.5.4	Τμήμα Κώδικα για το Neo4j Import (py_admin_import.py)	64
6	Στρατηγικές Βελτιστοποίησης Κώδικα και παραδείγματα εξόδου μέσω Cypher Queries	69
6.1	Βελτιστοποιήσεις Κώδικα για την Αντιμετώπιση των Περιορισμών Χώρου	69
6.1.1	Συμπίεση και Φιλτράρισμα Δεδομένων	69
6.1.2	Κριτήρια Φιλτραρίσματος Δεδομένων	70
6.1.3	Περιορισμοί και Προκλήσεις	71
6.2	Βελτιστοποιήσεις Κώδικα	76
6.2.1	Ανάλυση Κώδικα	76
6.2.2	Αρχείο unzipper.py	77
6.2.3	Ανάλυση και Περιγραφή του Κώδικα στα py_admin_import.py	78
6.2.4	Κατασκευή Εντολών Εισαγωγής (Import Command Construction)	78
6.2.5	Δομή Δεδομένων για Nodes και Relationships (Nodes and Relationships Structure)	78
6.3	Εισαγωγή OpenAlex Dataset στο Neo4j	79
6.3.1	Περιγραφή Εντολής Εισαγωγής	79
6.3.2	Ανάλυση της Εντολής	79
6.3.3	Παράδειγμα Εξόδου από την Εντολή Εισαγωγής στο Neo4j	80
6.3.4	Ενδεικτική Απεικόνιση των Αποτελεσμάτων	82
6.4	VNC Viewer	85
7	Στρατηγικές Βελτιστοποίησης Κώδικα και παραδείγματα εξόδου μέσω Cypher	

Queries	87
7.1 Σύνοψη	87
7.2 Συμπεράσματα	87
7.3 Επίλογος	88
Βιβλιογραφία	89

Κατάλογος σχημάτων

2.1	Αριθμός ερευνητικών έργων ανά κατηγορία στην OpenAlex [11]	10
2.2	Δομή Δεδομένων OpenAlex [14]	12
2.3	Δομή Δεδομένων OpenAlex [16]	13
2.4	Επισκόπηση API OpenAlex [17]	13
2.5	Σύγκριση Δεδομένων Ανάμεσα σε Βάσεις Δεδομένων [20]	14
3.1	Διαφορετικές Προσεγγίσεις Μεταξύ Βάσεων Δεδομένων [34]	20
3.2	Σύγκριση απόδοσης - performance benchmarks [38]	24
3.3	Σύγκριση σχεσιακών με βάσεις γραφημάτων [40]	25
3.4	Παράδειγμα ροής βιβλιομετρικών δεδομένων από την OpenAlex στην Neo4j όπως πραγματοποιήθηκε σε παρόμοια εργασία [41]	28
4.1	Θετικά και αρνητικά χαρακτηριστικά της Python για ανάλυση δεδομένων[53]	32
4.2	Οι βασικές βιβλιοθήκες της Python [54]	33
4.3	Ροή της γλώσσας ερωταποκρίσεων Cypher [70]	41
4.4	Η εντολή Neo4j-admin και οι επεκτάσεις της	44
5.1	Διαδικασία ETL επεξηγηματικά	46
6.1	Γράφημα αναπαράστασης της βάσης και παράδειγμα εξόδου που απεικονίζει τις σχέσεις μεταξύ των κόμβων (Όριο 500)	82
6.2	Λεπτομερής απεικόνιση των σχέσεων ενός κόμβου τύπου Institution με άλλους κόμβους.	82
6.3	Λεπτομερής απεικόνιση των ιδιοτήτων του παραπάνω κόμβου τύπου InstitutionId	83
6.4	Λεπτομερής απεικόνιση των ιδιοτήτων του κόμβου τύπου Concept	83
6.5	Λεπτομερής απεικόνιση των σχέσεων ενός κόμβου τύπου Work με άλλους κόμ- βους τύπου Authors	84
6.6	Λεπτομερής απεικόνιση των ιδιοτήτων του παραπάνω κόμβου τύπου Works στο δεξιά μέρος της εικόνας (Node Properties)	84
6.7	Λεπτομερής απεικόνιση των ιδιοτήτων του κόμβου τύπου Authors (Node Properties)	84

Κατάλογος πινάκων

6.1	Υποσύνολο nodes της βάσης που δημιουργήθηκε	81
6.2	Λεπτομερής απεικόνιση του συνόλου της υφιστάμενης βάσης	85

Κεφάλαιο 1

Εισαγωγή

Στο παρακάτω κεφάλαιο πραγματοποιείται μια σύντομη επισκόπηση της βιβλιομετρικής βάσης δεδομένων που ζητήθηκε να διαχειριστώ στα πλαίσια της πτυχιακής εργασίας. Έπειτα από την εκτενή ανάλυση της, δημιουργήθηκε και καταγράφηκε ένας πρώτος σκελετός της εργασίας. Σημαντικό ρόλο στον καθορισμό της βασικής μεθοδολογίας αποτέλεσε η κατανόηση και η εκμάθηση τόσο της OpenAlex καθώς και των απαιτήσεων της Neo4j.

1.1 Περιγραφή συνόλων δεδομένων

Η OpenAlex είναι ένας βιβλιογραφικός κατάλογος που περιλαμβάνει επιστημονικές εργασίες, συγγραφείς και ιδρύματα, και είναι διαθέσιμη με ελεύθερη πρόσβαση στο κοινό. Πήρε το όνομά της από τη Βιβλιοθήκη της Αλεξάνδρειας. Η λειτουργία της ξεκίνησε τον Ιανουάριο του 2022 από την OurResearch, ως διάδοχος του Microsoft Academic Graph, το οποίο είχε διακοπεί. Η OpenAlex ανταγωνίζεται εμπορικές πλατφόρμες όπως το Web of Science της Clarivate και το Scopus της Elsevier, ενώ συνοδεύεται από εργαλεία βιβλιομετρίας και ένα API για πρόσβαση στα δεδομένα της. Ένας αυξανόμενος αριθμός ερευνητικών έργων αξιοποιεί την OpenAlex ως πηγή δεδομένων. Αυτές οι μελέτες περιλαμβάνουν τόσο βιβλιομετρικές αναλύσεις όσο και αναλύσεις altmetrics, scoring reviews και δημιουργία γραφημάτων γνώσης για την καινοτομία στην τεχνητή νοημοσύνη. Οι περισσότερες από αυτές τις μελέτες αντιμετωπίζουν την OpenAlex ως μια νέα, πρακτική πηγή δεδομένων, χωρίς όμως να γίνεται πάντα μια εις βάθος κριτική για τους περιορισμούς της και των δεδομένων της. Για να αξιολογηθεί εάν η OpenAlex μπορεί να αντικαταστήσει τις κλειστές βάσεις δεδομένων, πρέπει να εξεταστούν διεξοδικά τόσο τα ισχυρά όσο και τα αδύνατα σημεία της. Ορισμένες μελέτες έχουν ήδη αρχίσει να δοκιμάζουν την αξιοπιστία της OpenAlex για βιβλιομετρικές μελέτες. Παρά το γεγονός ότι τα αποτελέσματα αυτών των μελετών δείχνουν πως η OpenAlex γενικά υπερ-χει σε όρους κάλυψης συγκριτικά με άλλες βάσεις δεδομένων, εντοπίστηκαν προβλήματα στην πληρότητα και ακρίβεια των βιβλιογραφικών μεταδεδομένων. Συγκεκριμένα, ορισμένα πεδία, όπως η ημερομηνία δημοσίευσης, ο τόμος, το τεύχος και ο αριθμός σελίδας, οι περιλήψεις, τα ονόματα συγγραφέων, οι αναφορές και παραπομπές, τα ιδρύματα, οι πληροφορίες σχετικά με τους χρηματοδότες, καθώς και η κατάσταση ανοιχτής πρόσβασης, παρατηρήθηκαν να είναι ελλιπή ή να περιέχουν ανακρίβειες. Στην παρούσα μελέτη στα πλαίσια της πτυχιακής μου εργασίας, στοχεύει στο να συνεισφέρει σε αυτές τις προσπάθειες αξιολόγησης της καταλληλότητας της OpenAlex

για βιβλιομετρικές αναλύσεις, εστιάζοντας παράλληλα στα δεδομένα που διαχειρίζεται και στις προκλήσεις που προκύπτουν κατά τη διαδικασία επεξεργασίας τους [1].

1.2 Αντικείμενο της Πτυχιακής Εργασίας

Στόχος της παρούσας πτυχιακής εργασίας είναι η δημιουργία μιας υποδομής για τη μορφοποίηση και προσπέλαση μεγάλων όγκων βιβλιογραφικών δεδομένων σε μια προηγμένη, μη σχεσιακή βάση δεδομένων γραφήματος. Μια τέτοια εφαρμογή έχει πολλαπλά μελλοντικά οφέλη για την ερευνητική κοινότητα, καθώς θα διευκολύνει τη δημιουργία και ανάλυση βάσεων γραφημάτων με σκοπό την εξαγωγή χρήσιμων και αξιόπιστων πληροφοριών για πλήθος ερευνητικών έργων. Η εργασία επικεντρώνεται στην αυτοματοποίηση κάθε διαδικασίας. Αρχικά, στη λήψη των απαραίτητων αρχείων από τη βιβλιομετρική βάση και στη συνέχεια στο φιλτράρισμα αυτών. Είναι προφανές ότι η εισαγωγή ολόκληρου του όγκου δεδομένων της OpenAlex δεν είναι εφικτή. Επομένως, η μορφοποίηση και το φιλτράρισμα αυτών των δεδομένων είναι απαραίτητες διαδικασίες. Στη συνέχεια, η σωστή δομή των τελικών αρχείων, σύμφωνα με τις απαιτήσεις της βάσης γραφημάτων, αποδείχθηκε κρίσιμης σημασίας. Μέσω της ανάλυσης που πραγματοποιήθηκε, διαπιστώθηκαν ορισμένα συμπεράσματα. Η OpenAlex βασίζεται σε μεγάλο βαθμό στις σχέσεις μεταξύ των οντοτήτων της. Ως εκ τούτου, απαιτείται η χρήση μιας ριζοσπαστικής τεχνολογίας βάσης δεδομένων, όπως η Neo4j, η οποία παρέχει ισοδύναμη αξία τόσο στα δεδομένα όσο και στις σχέσεις μεταξύ τους, εξασφαλίζοντας την απαραίτητη ακρίβεια. Η Neo4j διαθέτει την εντολή `neo4j-admin import`, η οποία κρίθηκε κατάλληλη για τη μαζική εισαγωγή δεδομένων. Αυτό επιτρέπει την αποδοτική και ακριβή αποθήκευση όχι μόνο των βασικών βιβλιογραφικών οντοτήτων, όπως συγγραφέας (Author), ερευνητικό έργο (Works), και οργανισμός (Institution), αλλά και των μεταξύ τους συσχετίσεων, όπως για παράδειγμα συγγραφέας (HAS AUTHOR), οργανισμός (HAS INSTITUTION). Η υποδομή που δημιουργήθηκε υποστηρίζει την ακριβή ανάλυση και οπτικοποίηση των δεδομένων και των σχέσεων τους. Επιπλέον, μέσω του υλοποιημένου κώδικα παρέχεται η δυνατότητα επιλογής εισαγωγής μεμονωμένων τμημάτων και ημερομηνιών από τον εκάστοτε χρήστη στο μέλλον, προσφέροντας ευελιξία και προσαρμοστικότητα στη διαδικασία.

1.3 Διάρθρωση της Εργασίας

Η διάρθρωση της πτυχιακής εργασίας έχει ως εξής: Το Κεφάλαιο δύο της πτυχιακής εργασίας επικεντρώνεται στην ανάλυση των βιβλιομετρικών δεδομένων και της πλατφόρμας OpenAlex, καλύπτοντας τις βασικές έννοιες, πηγές δεδομένων και τεχνικές λεπτομέρειες. Αρχικά, εισάγει τον ορισμό των βιβλιομετρικών δεδομένων, εξηγώντας τη σημασία τους στην αξιολόγηση της επιστημονικής δραστηριότητας μέσω δεικτών όπως οι αναφορές (citations), ο δείκτης επιρροής περιοδικών (impact factor) και ο δείκτης h-index. Αναφέρεται στην ιστορική εξέλιξη της βιβλιομετρίας και εξηγεί πώς η OpenAlex αποτελεί μια σύγχρονη πλατφόρμα ανοιχτής πρόσβασης, που παρέχει τα δεδομένα αυτά για ανάλυση και αξιολόγηση. Στη συνέχεια, παρουσιάζονται οι κύριες πηγές βιβλιομετρικών δεδομένων, διακρίνοντας τις παραδοσιακές πηγές, όπως το Scopus και το Web of Science, από τις ανοιχτές πλατφόρμες όπως το Dimensions, το CrossRef και το Google Scholar. Αναλύεται η διαφορά στην αξιοπιστία και προσβασιμότητα

αυτών, υπογραμμίζοντας τα πλεονεκτήματα και μειονεκτήματα της κάθε μιας. Η πλατφόρμα OpenAlex αναλύεται στη συνέχεια ως προς την αρχιτεκτονική της, το dataset, καθώς και το API που προσφέρει. Εξηγείται πώς οργανώνονται οι πέντε κύριες οντότητες της βάσης δεδομένων (έργα, συγγραφείς, περιοδικά, ιδρύματα, θεματικές κατηγορίες), και πώς αυτά τα δεδομένα μπορούν να αξιοποιηθούν για βιβλιομετρικές αναλύσεις, ανάλυση συνεργασιών και θεματικών τάσεων. Το κεφάλαιο ολοκληρώνεται με μια σύγκριση της OpenAlex με άλλες βάσεις δεδομένων, καθώς και την παρουσίαση ηθικών ζητημάτων στη χρήση των βιβλιομετρικών δεδομένων. Γίνεται αναφορά στις προοπτικές βελτίωσης της OpenAlex μέσω νέων τεχνολογιών, όπως η τεχνητή νοημοσύνη και η μηχανική μάθηση, ενώ προτείνονται μελλοντικές κατευθύνσεις και βελτιώσεις στην πλατφόρμα. Συνολικά, το κεφάλαιο προσφέρει μια ολοκληρωμένη ανάλυση της πλατφόρμας OpenAlex, των δυνατοτήτων της και των προκλήσεων που παρουσιάζει στη διαχείριση και ανάλυση βιβλιομετρικών δεδομένων. Το Τρίτο Κεφάλαιο επικεντρώνεται στην εισαγωγή και ανάλυση της βάσης δεδομένων γραφημάτων Neo4j, περιγράφοντας τη δομή, τα τεχνικά χαρακτηριστικά της και τη χρήση της σε βιβλιομετρικές βάσεις δεδομένων. Αρχικά, παρουσιάζεται η ιστορική ανάπτυξη της Neo4j, η οποία σχεδιάστηκε για να ανταποκριθεί στις ανάγκες αποθήκευσης και διαχείρισης δεδομένων με πολύπλοκες σχέσεις, που δεν μπορούν να διαχειριστούν αποτελεσματικά οι παραδοσιακές σχεσιακές βάσεις δεδομένων. Η Neo4j αποθηκεύει δεδομένα με τη μορφή γραφημάτων, όπου οι κόμβοι αντιπροσωπεύουν οντότητες και οι ακμές τις σχέσεις μεταξύ τους, επιτρέποντας τη γρήγορη ανάκτηση των σχέσεων σε πραγματικό χρόνο. Στη συνέχεια, αναλύεται η δομή της Neo4j και τα τεχνικά χαρακτηριστικά της, περιλαμβάνοντας την αρχιτεκτονική client-server, τη μηχανή αποθήκευσης και τη διαχείριση συναλλαγών. Περιγράφονται οι κύριες οντότητες της βάσης (κόμβοι, ακμές, ιδιότητες) και ο τρόπος με τον οποίο αποθηκεύονται και ανακτώνται τα δεδομένα. Τονίζεται επίσης η σημασία της βελτιστοποίησης ερωτημάτων, μέσω τεχνικών όπως τα ευρετήρια και ο βελτιστοποιητής ερωτημάτων. Το κεφάλαιο εξετάζει, επίσης, τις δυνατότητες της Neo4j για ασφάλεια, επεκτασιμότητα και απόδοση, με έμφαση στη διαχείριση δεδομένων σε περιβάλλοντα με μεγάλες και σύνθετες σχέσεις. Εξηγείται πώς η Neo4j υποστηρίζει συναλλαγές συμβατές με το πρότυπο ACID, διασφαλίζοντας την ακεραιότητα και τη συνέπεια των δεδομένων. Ακολουθεί μια σύγκριση της Neo4j με παραδοσιακές σχεσιακές βάσεις δεδομένων (SQL) και άλλες NoSQL βάσεις, επισημαίνοντας τα πλεονεκτήματα και μειονεκτήματα της κάθε προσέγγισης. Η Neo4j υπερέχει όταν οι σχέσεις μεταξύ των δεδομένων είναι σύνθετες και απαιτείται γρήγορη εκτέλεση ερωτημάτων. Τέλος, το κεφάλαιο εστιάζει στην εφαρμογή της Neo4j για τη διαχείριση βιβλιομετρικών δεδομένων, εξηγώντας πώς μπορεί να χρησιμοποιηθεί για την αποθήκευση δημοσιεύσεων, συγγραφέων και παραπομπών. Τονίζεται η ικανότητα της Neo4j να αναπαριστά φυσικά τις σχέσεις μεταξύ δεδομένων, προσφέροντας ταχεία ανάλυση και οπτικοποίηση αυτών των σχέσεων. Το επόμενο κεφάλαιο εστιάζει στην Python και τη γλώσσα ερωταποκρίσεων Cypher, παρέχοντας μια λεπτομερή ανάλυση των δύο αυτών εργαλείων και της χρήσης τους. Αρχικά, γίνεται μια επισκόπηση της Python, ξεκινώντας από την ιστορική της εξέλιξη, τονίζοντας πώς έχει εξελιχθεί από τη δημιουργία της μέχρι σήμερα και γιατί έχει γίνει μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού. Ιδιαίτερη αναφορά γίνεται στα χαρακτηριστικά που την καθιστούν κατάλληλη για ανάλυση δεδομένων, όπως η ευκολία χρήσης και η πλούσια βιβλιοθήκη εργαλείων που προσφέρει. Στη συνέχεια, παρουσιάζονται οι βασικές βιβλιοθήκες της Python που χρησιμοποιούνται στο πλαίσιο της παρούσας εργασίας, όπως οι `gzip`, `os`, `shutil`, `json` και `csv`, με έμφαση στη συμβολή τους στη διαχείριση αρχείων και δεδομένων. Αναλύεται επίσης η σύνδεση της Python με τη βάση δεδομένων Neo4j, εξηγώντας πώς η γλώσσα μπορεί

να χρησιμοποιηθεί για την εκτέλεση ερωτημάτων Cypher και τη διαχείριση δεδομένων γραφημάτων, προσφέροντας ευελιξία και ευκολία στη χρήση. Ακολουθως, εξετάζεται η γλώσσα ερωταποκρίσεων Cypher, η οποία είναι σχεδιασμένη ειδικά για βάσεις δεδομένων γραφημάτων, όπως η Neo4j. Παρουσιάζεται η βασική σύνταξή της, η οποία είναι εμπνευσμένη από την SQL, καθώς και τα πλεονεκτήματα που προσφέρει, όπως η απλότητα και η αποτελεσματικότητα στην αναζήτηση συνδεδεμένων δεδομένων. Επίσης, γίνεται αναφορά στις εφαρμογές της Cypher σε διάφορους τομείς, όπως η ανάλυση κοινωνικών δικτύων και η ανίχνευση απάτης. Το κεφάλαιο ολοκληρώνεται με τη σύγκριση των μεθόδων εισαγωγής δεδομένων στη Neo4j, τόσο μέσω Cypher όσο και μέσω του εργαλείου Neo4j Admin Import. Τονίζεται ότι, ενώ η Cypher είναι ιδανική για μικρότερες εισαγωγές δεδομένων και τη συνεχή διαχείριση γραφημάτων, το Neo4j Admin Import είναι πιο αποδοτικό για μαζικές εισαγωγές μεγάλων όγκων δεδομένων, προσφέροντας ταχύτητα και κλιμάκωση. Το πέμπτο κεφάλαιο επικεντρώνεται στις τεχνικές ανάλυσης δεδομένων και τη χρήση των εργαλείων αυτών στο πλαίσιο της παρούσας εργασίας. Αρχικά, παρουσιάζεται μια επισκόπηση των μεθόδων ανάλυσης δεδομένων που χρησιμοποιούνται ευρέως στη βιβλιομετρική έρευνα, εστιάζοντας σε τεχνικές όπως η ανάλυση δεδομένων και η κατηγοριοποίηση. Δίνεται έμφαση στη σημασία των στατιστικών μεθόδων και των εργαλείων οπτικοποίησης δεδομένων για την κατανόηση και ερμηνεία των αποτελεσμάτων. Στη συνέχεια, αναλύεται η διαδικασία προετοιμασίας των δεδομένων για εισαγωγή στη Neo4j, με στόχο την ανάλυση των βιβλιομετρικών δεδομένων που προέρχονται από την πλατφόρμα OpenAlex. Περιγράφεται ο τρόπος με τον οποίο τα δεδομένα μορφοποιούνται και καθαρίζονται ώστε να είναι συμβατά με τη δομή της βάσης γραφημάτων. Ιδιαίτερη έμφαση δίνεται στη χρήση εργαλείων Python για την επεξεργασία μεγάλων αρχείων και τη διαχείριση των σχέσεων μεταξύ των κόμβων. Στη συνέχεια του κεφαλαίου, εξετάζονται οι μέθοδοι εξαγωγής πληροφοριών από τη Neo4j μέσω της γλώσσας Cypher, με στόχο την ανάλυση των σχέσεων μεταξύ των επιστημονικών δημοσιεύσεων και των συγγραφέων τους. Γίνεται αναφορά στα ερωτήματα που διαμορφώθηκαν για την εξαγωγή των απαιτούμενων δεδομένων και στην ανάλυση των δικτύων συνεργασίας που προκύπτουν από τα δεδομένα αυτά. Το κεφάλαιο ολοκληρώνεται με την παρουσίαση των αποτελεσμάτων της ανάλυσης, εστιάζοντας στα βασικά συμπεράσματα που προκύπτουν από τα δεδομένα της OpenAlex. Δίνεται έμφαση στη σημασία της χρήσης εργαλείων γραφημάτων για την ανάλυση βιβλιομετρικών δεδομένων και στον τρόπο με τον οποίο αυτά τα εργαλεία μπορούν να συμβάλουν στην εξαγωγή χρήσιμων γνώσεων και προτύπων από τα δεδομένα. Το Κεφάλαιο έξι επικεντρώνεται στις στρατηγικές βελτιστοποίησης κώδικα που εφαρμόστηκαν για την αντιμετώπιση των περιορισμών και προβλημάτων κατά την εισαγωγή των δεδομένων της OpenAlex στη βάση δεδομένων Neo4j. Οι βελτιστοποιήσεις αυτές ήταν απαραίτητες λόγω του μεγάλου όγκου και της πολυπλοκότητας των δεδομένων, καθώς και των αυστηρών απαιτήσεων της διαδικασίας εισαγωγής, που καθιστούν τη διαδικασία χρονοβόρα. Στη συνέχεια, περιγράφονται οι στρατηγικές συμπίεσης και φιλτράρισματος δεδομένων. Παρουσιάζονται παραδείγματα για το πόσο μειώθηκαν τα δεδομένα σε κατηγορίες. Επιπλέον, αναλύονται οι προκλήσεις που προέκυψαν κατά τη διάρκεια της διαδικασίας εισαγωγής, όπως οι περιορισμοί στη χωρητικότητα δίσκων, η διαχείριση σφαλμάτων κατά τη λήψη των δεδομένων και οι δυσκολίες στη διαδικασία του import. Τέλος, περιγράφεται η διαδικασία δημιουργίας ειδικών CSV αρχείων, τα οποία προέκυψαν από την κανονικοποίηση και τον καθαρισμό των δεδομένων της OpenAlex. Ο κώδικας που χρησιμοποιήθηκε για τη δημιουργία των CSV περιλάμβανε βελτιστοποιήσεις για να αποφευχθούν διπλότυπα και ασυνέπειες στα δεδομένα, διασφαλίζοντας την ακεραιότητα της βάσης. Η ενότητα ολοκληρώ-

νεται με την παρουσίαση των παραδειγμάτων εξόδου μέσω ερωτημάτων Cypher, καθώς και των στρατηγικών που χρησιμοποιήθηκαν για τη διαχείριση της διαδικασίας import. Επιπροσθέτως, περιγράφεται η χρήση του VNC Viewer περιγράφεται ως λύση για απομακρυσμένη πρόσβαση στο μηχάνημα της σχολής, αναλύοντας λεπτομερώς τη διαδικασία εγκατάστασης και παραμετροποίησης του λογισμικού για την αντιμετώπιση προβλημάτων αλληλεπίδρασης με την επιφάνεια εργασίας. Στο έβδομο και τελευταίο κεφάλαιο γίνεται μια σύντομη επισκόπηση όλης της εφαρμογής και παρουσιάζονται τα συμπεράσματα και οι πιθανές μελλοντικές επεκτάσεις και βελτιστοποιήσεις της παρούσας πτυχιακής εργασίας.

1.4 Επίλογος

Στο κεφάλαιο αυτό πραγματοποιήθηκε μια σύντομη επισκόπηση των κεφαλαίων που θα ακολουθήσουν, καθώς και του περιεχομένου τους. Παρουσιάστηκε μια αρχική ανάλυση της βιβλιομετρικής βάσης δεδομένων που θα αποτελέσει την κύρια πηγή των δεδομένων μας, καθώς και της βάσης στην οποία θα γίνει η εισαγωγή αυτών, με έμφαση στις ιδιαιτερότητες και τις απαιτήσεις της. Στη συνέχεια, θα δοθεί ιδιαίτερη προσοχή στις προκλήσεις και τα συμπεράσματα που προέκυψαν κατά τη διαδικασία της εισαγωγής. Παράλληλα, έγινε μια εισαγωγική περιγραφή των μελλοντικών δυνατοτήτων της υλοποιημένης εφαρμογής. Αποφασίστηκε η δημιουργία ενός ευέλικτου κώδικα που θα επιτρέπει την εισαγωγή βέλτιστων βιβλιογραφικών δεδομένων, τα οποία θα παρέχουν την εκτενέστερη, υψηλότερη ποιότητα και πιο ενδιαφέρουσα δομή βιβλιογραφικής πληροφορίας. Θα αναλυθεί περαιτέρω η ανάγκη για ακρίβεια και ακεραιότητα όσον αφορά τις απαιτήσεις της Neo4j σχετικά με τα εισαγόμενα δεδομένα. Τέλος, παρακάτω θα πραγματοποιηθεί μια πιο εκτενής εισαγωγή στα βιβλιομετρικά σύνολα δεδομένων και ανάλυση της πλατφόρμας OpenAlex, η οποία αποτελεί τον πρώτο τεχνολογικό πυλώνα της διπλωματικής εργασίας.

Κεφάλαιο 2

Εισαγωγή στα Βιβλιομετρικά Datasets και ανάλυση της πλατφόρμας OpenAlex

2.1 Εισαγωγή στα Βιβλιομετρικά Δεδομένα

2.1.1 Ορισμός των Βιβλιομετρικών Δεδομένων

Βασικό εργαλείο για την μελέτη και αξιολόγηση της επιστημονικής δραστηριότητας και παραγωγής, αποτελούν τα βιβλιομετρικά δεδομένα. Περιλαμβάνουν μετρήσεις που αφορούν την δημοσίευση επιστημονικών έργων και την επιρροή τους (citations), τον δείκτη επιρροής των επιστημονικών περιοδικών (impact factor), καθώς επίσης και τους δείκτες παραγωγικότητας των συγγραφέων (h-index). Τα παραπάνω δεδομένα αποτελούν πυλώνα για τη βιβλιομετρική ανάλυση, η οποία χρησιμοποιείται για την αποτίμηση της ποιότητας της έρευνας, την αξιολόγηση της απόδοσης και τέλος για τη λήψη αποφάσεων σε επίπεδο χρηματοδότησης και πρόσληψης [2]. Ένα από τα βασικότερα εργαλεία για την καλύτερη διαχείριση και ανάλυση βιβλιομετρικών δεδομένων είναι η OpenAlex, η οποία ενσωματώνει αναφορές και συνδέσμους μεταξύ ακαδημαϊκών εργασιών. Παρέχει ελεύθερα αυτά τα δεδομένα μέσω ενός ισχυρού API, επιτρέποντας την άμεση πρόσβαση σε ενημερωμένα και αξιόπιστα δεδομένα δίχως περιορισμούς όπως απαιτείται από άλλες παραδοσιακές βάσεις δεδομένων.

2.1.2 Ιστορική Εξέλιξη και Σημαντικότητα στη Σύγχρονη Έρευνα

Η βιβλιομετρία, δηλαδή η ποσοτική ανάλυση των επιστημονικών δημοσιεύσεων, προέρχεται από τις πρώτες προσπάθειες καταγραφής και ανάλυσης της επιστημονικής παραγωγής. Βάση της βιβλιομετρικής ανάλυσης αποτελεί η ιδέα να μετρηθεί η επιρροή ενός επιστημονικού έργου χρησιμοποιώντας τις αναφορές που λαμβάνει από άλλα έργα [3]. Την δεκαετία του 1960, με την δημιουργία του Science Citation Index (SCI), η βιβλιομετρία γνώρισε σημαντική ανάπτυξη, καθώς επέτρεψε την παρακολούθηση των αναφορών σε επιστημονικά άρθρα. Η εξέλιξη αυτή, δημιούργησε νέους δείκτες, όπως ο impact factor και ο h-index, που επιτρέπουν τη λεπτομερή αξιολόγηση της επιστημονικής επιρροής και παραγωγικότητας [2]. Στη σημερινή εποχή, η απαίτηση για πιο ανοιχτά, διαφανή και προσαρμόσιμα εργαλεία οδήγησε στη δημιουργία πλατφορμών όπως η OpenAlex. Αντίθετα με τις παραδοσιακές βιβλιομετρικές βάσεις δεδομένων, η OpenAlex αξιοποιεί την ανοιχτή επιστήμη προσφέροντας στους χρήστες

πρόσβαση σε εκτεταμένα σύνολα δεδομένων, τα οποία μπορούν να χρησιμοποιηθούν για την ανάλυση των τάσεων στην επιστημονική έρευνα, την αξιολόγηση της απόδοσης και τη μελέτη της επιστημονικής συνεργασίας [4].

2.1.3 Βασικοί Τύποι Βιβλιομετρικών Δεδομένων

Τα βιβλιομετρικά δεδομένα χωρίζονται σε διάφορους τύπους, καθένας από τους οποίους προσφέρει διαφορετικές πληροφορίες και αξιοποιείται για συγκεκριμένες αναλύσεις:

- **Δεδομένα Αναφορών (Citations):** Ορίζεται ως ο αριθμός των αναφορών που λαμβάνει ένα επιστημονικό έργο από άλλα έργα και αποτελεί έναν από τους πιο σημαντικούς δείκτες επιρροής. Στην OpenAlex, οι αναφορές αυτές διατίθενται ελεύθερα και μπορούν να αναλυθούν για να προσδιοριστεί η επιρροή ενός άρθρου, μιας μελέτης ή ενός ερευνητή [5].
- **Impact Factor:** Είναι ο δείκτης όπου μετρά τον μέσο αριθμό αναφορών που λαμβάνουν τα άρθρα ενός περιοδικού σε μια συγκεκριμένη χρονική περίοδο. Σημειώνεται ότι η OpenAlex δεν υπολογίζει απευθείας τον impact factor, παρόλα αυτά τα δεδομένα που παρέχει μπορούν να χρησιμοποιηθούν από άλλους για να πραγματοποιήσουν τέτοιους υπολογισμούς [5].
- **h-index και Άλλοι Δείκτες Παραγωγικότητας:** Ο h-index είναι ένας δείκτης που συνδυάζει την ποσότητα και την ποιότητα των δημοσιεύσεων ενός ερευνητή. Ο συγκεκριμένος δείκτης λαμβάνει υπόψη τόσο τον αριθμό των δημοσιεύσεων όσο και τις αναφορές που αυτές λαμβάνουν. Η OpenAlex παρέχει τα απαιτούμενα δεδομένα για τον υπολογισμό τέτοιων δεικτών, δίνοντας στους χρήστες τη δυνατότητα να εκτελέσουν λεπτομερείς αναλύσεις της ερευνητικής παραγωγικότητας [5].

Χρησιμοποιώντας αυτά τα βιβλιομετρικά δεδομένα, η OpenAlex προσφέρει μια πλατφόρμα η οποία διευκολύνει την πρόσβαση και την ανάλυση της επιστημονικής επιρροής, επιτρέποντας την εφαρμογή βιβλιομετρικών μεθόδων για την αξιολόγηση της έρευνας [6].

2.2 Κύριες Πηγές Βιβλιομετρικών Δεδομένων

2.2.1 Παραδοσιακές Πηγές Βιβλιομετρικών Δεδομένων

- **Scopus:** Το Scopus, αποτελεί μια από τις μεγαλύτερες και πιο αξιόπιστες βάσεις δεδομένων για βιβλιομετρικά δεδομένα. Έχει την δυνατότητα να παρέχει λεπτομερείς πληροφορίες για επιστημονικά άρθρα, βιβλία και συνέδρια. Αν και απαιτείται συνδρομητική πρόσβαση στο Scopus, προσφέρει αξιόπιστα και συστηματικά δεδομένα που χρησιμοποιούνται ευρέως σε ακαδημαϊκές και ερευνητικές αναλύσεις. Χρησιμοποιείται κυρίως για αναζήτηση αναφορών, ανάλυση της διάδοσης της έρευνάς τους και εντοπισμό νέων τάσεων στην επιστήμη [7].

- **Web of Science:** Κορυφαία πηγή βιβλιομετρικών δεδομένων που προσφέρει πρόσβαση σε παγκόσμια βιβλιογραφία υψηλής ποιότητας είναι το Web of Science. Χρησιμοποιείται κυρίως για τη βιβλιομετρική ανάλυση και την αξιολόγηση της επιστημονική επιρροής, προσφέροντας δεδομένα αναφορών, υπολογισμούς του impact factor και άλλους δείκτες. Η ευρεία χρήση του Web of Science στην ακαδημαϊκή κοινότητα το καθιστά έναν πολύτιμο πόρο για ερευνητές και ακαδημαϊκούς.

Παρόλη την αξιοπιστία και την ευρεία χρησιμότητα, οι παραδοσιακές πηγές έχουν κάποιες περιοριστικές παραμέτρους. Μια από αυτές είναι η πρόσβαση με συνδρομή και οι αυστηροί κανόνες εισαγωγής περιοδικών στη βάση τους [7].

2.2.2 Ανοιχτές Πηγές Βιβλιομετρικών Δεδομένων

- **Dimensions:** Το Dimensions αποτελεί μια ανοιχτή πλατφόρμα προσφέροντας πρόσβαση σε ένα ευρύ φάσμα βιβλιομετρικών δεδομένων, συμπεριλαμβανομένων δημοσιεύσεων κλινικών δοκιμών και χρηματοδοτημένων ερευνών [8]. Χρησιμοποιώντας το API της πλατφόρμας δίνεται η δυνατότητα στους χρήστες να εκτελούν σύνθετες αναλύσεις και κατ' επέκταση να ενσωματώνουν δεδομένα σε δικές τους εφαρμογές. Εξαιτίας της ανοιχτής φύσης της πλατφόρμας καθίσταται ως ένα ισχυρό εργαλείο για ερευνητές που επιθυμούν να αναλύσουν μεγάλα δεδομένα χωρίς να αντιμετωπίζουν περιορισμούς πρόσβασης.
- **CrossRef:** Το CrossRef παρέχει ανοιχτή πρόσβαση σε πληροφορίες σχετικά με τις επιστημονικές δημοσιεύσεις μέσω της χρήσης DOI (Digital Object Identifier). Μέσω της διασύνδεσης δεδομένων με τα DOI επιτρέπεται η αναζήτηση και ο εντοπισμός δημοσιεύσεων με ακρίβεια και ταχύτητα. Το CrossRef ενισχύει τη βάση δεδομένων της OpenAlex, κάνοντας πλουσιότερο το σύνολο των βιβλιομετρικών δεδομένων που παρέχει [8].
- **Google Scholar:** Το Google Scholar, αν και δεν θεωρείται αυστηρή στην επιλογή πηγών όπως οι παραδοσιακές βάσεις δεδομένων, παραμένει μια εξαιρετικά χρήσιμη ανοιχτή πλατφόρμα για την αναζήτηση ακαδημαϊκών έργων. Το Google Scholar δίνει πρόσβαση σε πλήθος δημοσιεύσεων, συμπεριλαμβανομένων πολλών που δεν καλύπτονται από το Scopus ή το Web of Science [8].

2.2.3 Σύγκριση μεταξύ Παραδοσιακών και Ανοιχτών Πηγών

- **Κριτήρια Αξιολόγησης και Αξιοπιστία:** Ορισμένες από τις παραδοσιακές πηγές, όπως το Scopus και το Web of Science, είναι ευρέως γνωστές για την αυστηρότητα και την επιλογή των δεδομένων που περιλαμβάνουν. Αυτή η αξιοπιστία τους καθιστά τα κύρια εργαλεία για την ακαδημαϊκή αξιολόγηση, παρόλα αυτά οι υψηλές συνδρομές και οι περιορισμοί πρόσβασης μειώνουν τη απήχηση τους [6]. Από την άλλη, ανοιχτές πηγές, όπως το Dimensions και η OpenAlex, προσφέρουν ευκολότερη πρόσβαση στα δεδομένα χωρίς περιορισμούς, ενώ ταυτόχρονα διατηρούν την ακρίβεια και την αξιοπιστία τους. Συγκεκριμένα, η OpenAlex ξεχωρίζει σε αυτόν τον τομέας, εξαιτίας της ενσωμάτωσης δεδομένων από πολλές διαφορετικές πηγές, συμπεριλαμβανομένων των CrossRef και

Google Scholar, προσφέροντας μια ολοκληρωμένη και προσβάσιμη πλατφόρμα για βιβλιομετρική ανάλυση [9].

- Διαθεσιμότητα και Προσβασιμότητα : Βασικός παράγοντας που επηρεάζει την επιλογή βάσης δεδομένων είναι η διαθεσιμότητα των δεδομένων που θα χρησιμοποιήσει ένας ερευνητής. Οι ανοιχτές πηγές παρέχουν ελεύθερη πρόσβαση, σε αντίθεση με τις παραδοσιακές πηγές που είναι προσβάσιμες μόνο μέσω συνδρομής, γεγονός που τις καθιστά ιδιαίτερα ελκυστικές για ερευνητές με περιορισμένο προϋπολογισμό ή πρόσβαση σε ακαδημαϊκές βιβλιοθήκες. Η OpenAlex, επιτρέπει στους χρήστες να εξετάσουν και να αναλύσουν τα δεδομένα χωρίς να προαπαιτείτε να πληρώσουν για συνδρομές, καθιστώντας την μια εξαιρετική επιλογή για ευρύτερη πρόσβαση και ανάλυση της επιστημονικής γνώσης [9].

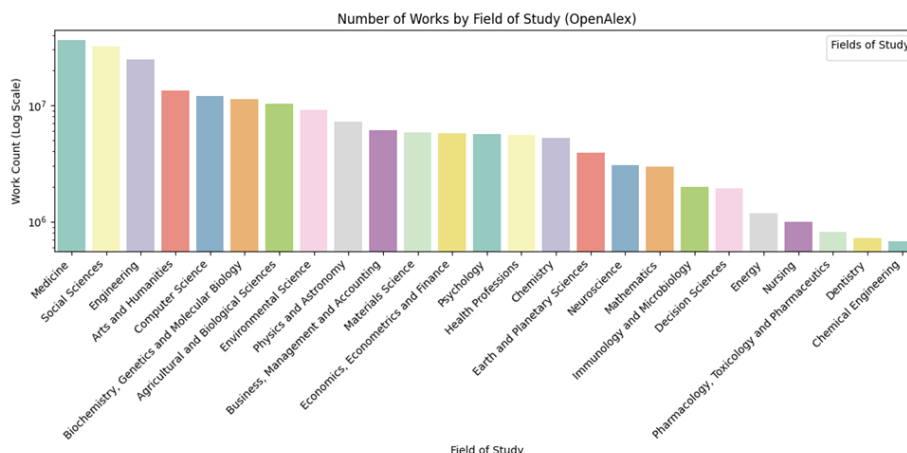
2.3 Η Πλατφόρμα OpenAlex

2.3.1 Ιστορικό και Εξέλιξη της OpenAlex

Η OpenAlex είναι μια ανοιχτή και δωρεάν πλατφόρμα που δημιουργήθηκε ως συνέχεια του Microsoft Academic Graph (MAG), το οποίο έκλεισε τον Δεκέμβριο του 2021. Μετά το κλείσιμο του MAG, δημιουργήθηκε η ανάγκη για ένα σύστημα το οποίο θα μπορεί να το καλύψει και να προσφέρει μια προσβάσιμη και ανοιχτή λύση για τους ερευνητές. Η OpenAlex σχεδιάστηκε με σκοπό να διευκολύνει την πρόσβαση σε ακαδημαϊκά δεδομένα προσφέροντας, ταυτόχρονα, μια πλατφόρμα που ενσωματώνει δεδομένα και από πολλές άλλες πηγές. Είναι διαθέσιμη για ελεύθερη χρήση από την ερευνητική κοινότητα [10]. Το όνομα της είναι εμπνευσμένο από τη Βιβλιοθήκη της Αλεξάνδρειας, έναν από τους πιο διάσημους τόπους γνώσης στην αρχαιότητα. Η επιλογή αυτού του ονόματος αντικατοπτρίζει τη δέσμευση της OpenAlex για την παροχή ανοιχτής πρόσβασης στην ακαδημαϊκή γνώση, με στόχο τη διάδοση και την ανάπτυξη της επιστημονικής έρευνας σε παγκόσμια κλίμακα [10]. Μετά την ίδρυσή της η OpenAlex είχε ραγδαία εξέλιξη, υιοθετώντας σύγχρονες τεχνολογίες και μεθοδολογίες για τη συλλογή, την αποθήκευση και την ανάλυση ακαδημαϊκών δεδομένων. Προσφέρει ελεύθερη και ανοιχτή πρόσβαση στα δεδομένα της, αντιθέτως με τους προκατόχους της που συχνά απαιτούσαν συνδρομές και πρόσβαση μέσω ακαδημαϊκών βιβλιοθηκών, ενισχύοντας την ανοιχτή επιστήμη και διευκολύνοντας τη συνεργασία μεταξύ ερευνητών παγκοσμίως [10].

2.3.2 Στόχοι και Φιλοσοφία της Πλατφόρμας

Βασιζόμενη στην κύρια φιλοσοφία της πλατφόρμας, δηλαδή στην αρχή της ανοιχτή επιστήμης, η οποία υποστηρίζει την ελεύθερη διανομή της γνώσης και την προσβασιμότητα της έρευνας σε όλους τους ενδιαφερόμενους, ανεξαρτήτως γεωγραφικής ή οικονομικής κατάστασης. Στόχος της OpenAlex είναι να προσφέρει ανοιχτή, διαφανή και εύκολα προσβάσιμη πληροφορία στον ακαδημαϊκό τομέα [12]. Η OpenAlex προσπαθεί να παρέχει δεδομένα που είναι τόσο ακριβή όσο και ενημερωμένα, ενσωματώνοντας στοιχεία από πολλαπλές πηγές, όπως το CrossRef, το PubMed και το ORCID. Επιπρόσθετα, η πλατφόρμα δίνει έμφαση στη διαλειτουργικότητα, επιτρέποντας την εύκολη ενσωμάτωση των δεδομένων της σε άλλες εφαρμογές και εργαλεία, μέσω ενός ισχυρού API [13].



Σχήμα 2.1: Αριθμός ερευνητικών έργων ανά κατηγορία στην OpenAlex [11]

2.3.3 Δομή του Dataset της OpenAlex

Η OpenAlex οργανώνει τα δεδομένα της σε πέντε κύριες οντότητες, καθεμία από τις οποίες αντιπροσωπεύει ένα διαφορετικό τύπο της ακαδημαϊκής δραστηριότητας. Αυτές οι οντότητες είναι:

1. **Works (Έργα):** Τα έργα είναι η κεντρική οντότητα στο dataset της OpenAlex και αντιπροσωπεύουν τα διάφορα επιστημονικά έργα, όπως άρθρα, βιβλία, πρακτικά συνεδρίων και διατριβές. Κάθε έργο περιέχει τα εξής βασικά πεδία:
 - **ID:** Ένας μοναδικός αναγνωριστικός κωδικός για κάθε έργο.
 - **Title:** Ο τίτλος του έργου.
 - **Abstract:** Η περίληψη του έργου, όπου αυτή είναι διαθέσιμη.
 - **Publication Date:** Η ημερομηνία δημοσίευσης του έργου.
 - **Authors:** Μια λίστα με τους συγγραφείς που συμμετείχαν στο έργο, συνδεδεμένοι με τις αντίστοιχες εγγραφές συγγραφέων.
 - **Venues:** Οι δημοσιευμένες πηγές, όπως περιοδικά ή συνέδρια, όπου δημοσιεύθηκε το έργο.
 - **References:** Λίστα με τα έργα που αναφέρονται στο συγκεκριμένο έργο, επιτρέποντας την ανάλυση των δικτύων αναφορών.
 - **Concepts:** Θεματικές κατηγορίες στις οποίες κατατάσσεται το έργο
2. **Authors (Συγγραφείς):** Η οντότητα "Authors" περιλαμβάνει δεδομένα για τους συγγραφείς των επιστημονικών έργων. Τα δεδομένα αυτά επιτρέπουν την ανάλυση της παραγωγικότητας των ερευνητών, των συνεργασιών μεταξύ τους και της επιρροής τους στον ακαδημαϊκό χώρο. Τα βασικά πεδία περιλαμβάνουν:
 - **ID:** Ένας μοναδικός αναγνωριστικός κωδικός για κάθε συγγραφέα.
 - **Name:** Το πλήρες όνομα του συγγραφέα.

- **Works:** Λίστα με τα έργα στα οποία έχει συμμετάσχει ο συγγραφέας, συνδεδεμένα με τις αντίστοιχες εγγραφές έργων.
- **Affiliations:** Τα ακαδημαϊκά ή ερευνητικά ιδρύματα με τα οποία σχετίζεται ο συγγραφέας.
- **Citations:** Ο συνολικός αριθμός των αναφορών που έχουν λάβει τα έργα του συγγραφέα.

Αυτά τα δεδομένα είναι χρήσιμα για την ανάλυση της ερευνητικής δραστηριότητας σε επίπεδο ατόμου, επιτρέποντας τη μελέτη της επιστημονικής παραγωγικότητας και των συνεργασιών μεταξύ ερευνητών [10].

3. **Venues (Περιοδικά και Συνέδρια)** Η οντότητα "Venues" αφορά τις δημοσιευμένες πηγές, όπως τα περιοδικά και τα συνέδρια. Αυτή η οντότητα περιέχει τα ακόλουθα βασικά πεδία:

- **ID:** Ένας μοναδικός αναγνωριστικός κωδικός για κάθε περιοδικό ή συνέδριο.
- **Name:** Το όνομα του περιοδικού ή του συνεδρίου.
- **ISSN:** Ο Διεθνής Τυποποιημένος Αριθμός Περιοδικών για τα περιοδικά.
- **Publisher:** Ο εκδοτικός οίκος που εκδίδει το περιοδικό ή το συνέδριο.
- **Works:** Λίστα με τα έργα που έχουν δημοσιευτεί σε αυτό το περιοδικό ή συνέδριο.

Αυτή η οντότητα είναι κρίσιμη για την ανάλυση της επιρροής των δημοσιευμένων πηγών στην ακαδημαϊκή κοινότητα, καθώς και για τη μελέτη της διάδοσης της επιστημονικής γνώσης μέσω διαφορετικών καναλιών [10].

4. **Institutions (Ιδρύματα):** Η οντότητα "Institutions" περιλαμβάνει δεδομένα για τα ακαδημαϊκά και ερευνητικά ιδρύματα. Τα βασικά πεδία περιλαμβάνουν:

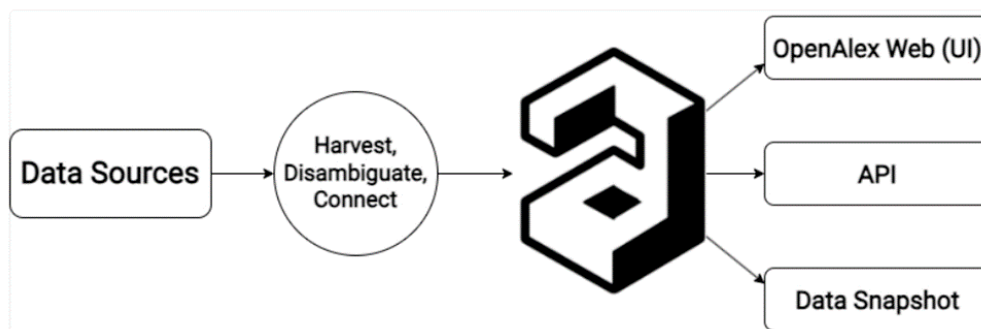
- **ID:** Ένας μοναδικός αναγνωριστικός κωδικός για κάθε ίδρυμα.
- **Name:** Το όνομα του ιδρύματος.
- **Country:** Η χώρα στην οποία βρίσκεται το ίδρυμα.
- **Authors:** Λίστα με τους συγγραφείς που συνδέονται με το ίδρυμα, επιτρέποντας την ανάλυση της ερευνητικής δραστηριότητας ανά ίδρυμα.
- **Works:** Λίστα με τα έργα που έχουν παραχθεί από συγγραφείς του ιδρύματος

Αυτά τα δεδομένα είναι ιδιαίτερα χρήσιμα για την αξιολόγηση της ερευνητικής απόδοσης σε επίπεδο οργανισμού και τη μελέτη των συνεργασιών μεταξύ ιδρυμάτων [10].

5. **Concepts (Θεματικές Κατηγορίες):** Η οντότητα "Concepts" αφορά τις θεματικές κατηγορίες στις οποίες κατατάσσονται τα έργα. Αυτές οι κατηγορίες είναι οργανωμένες σε ιεραρχίες και επιτρέπουν την ανάλυση των θεματικών περιοχών της έρευνας. Τα βασικά πεδία περιλαμβάνουν:

- **ID:** Ένας μοναδικός αναγνωριστικός κωδικός για κάθε θεματική κατηγορία.

- **Name:** Το όνομα της θεματικής κατηγορίας.
- **Level:** Το επίπεδο της κατηγορίας στην ιεραρχία (π.χ., γενικό θέμα ή υποκατηγορία).
- **Works:** Λίστα με τα έργα που ανήκουν σε αυτήν τη θεματική κατηγορία. Αυτή η οντότητα είναι σημαντική για την ανάλυση των επιστημονικών τάσεων και τη μελέτη της εξέλιξης των θεματικών περιοχών στην ακαδημαϊκή έρευνα [10].



Σχήμα 2.2: Δομή Δεδομένων OpenAlex [14]

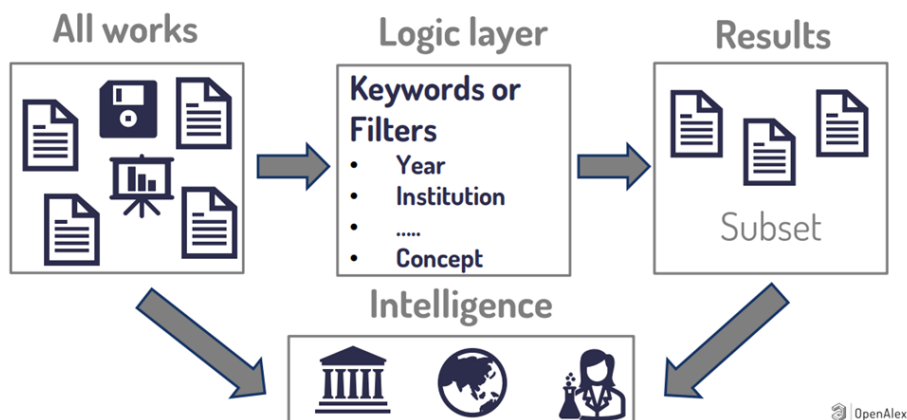
Κάθε μία από αυτές τις οντότητες είναι δομημένη με τρόπο που επιτρέπει τη διασύνδεση με τις άλλες, δημιουργώντας ένα σύνθετο δίκτυο ακαδημαϊκής γνώσης [13]. Τα δεδομένα παρέχονται σε μορφή JSON, διευκολύνοντας την επεξεργασία και ανάλυσή τους σε ποικιλία εργαλείων και πλατφορμών.

2.4 Τεχνική Ανάλυση της OpenAlex

2.4.1 Δομή και Αρχιτεκτονική της OpenAlex

Η δομή της OpenAlex αποτελείται από μια αρχιτεκτονική μικροϋπηρεσιών (microservices architecture), η οποία προσφέρει σημαντικά πλεονεκτήματα όσον αφορά την ευελιξία, την επεκτασιμότητα και την ανθεκτικότητα της πλατφόρμας. Σε αντίθεση με διαφορετικές αρχιτεκτονικές, όπου όλες οι λειτουργίες ενός συστήματος βρίσκονται ενσωματωμένες σε έναν ενιαίο κώδικα, η αρχιτεκτονική που χρησιμοποιείται επιτρέπει την ανάπτυξη, τη συντήρηση και την αναβάθμιση κάθε υπηρεσίας ανεξάρτητα [15]. Βασικό χαρακτηριστικό της αρχιτεκτονικής αυτής είναι πως κάθε μικροϋπηρεσία στην OpenAlex είναι υπεύθυνη για μια συγκεκριμένη λειτουργία, όπως η διαχείριση δεδομένων συγγραφέων, η ανάλυση αναφορών ή η αναζήτηση έργων. Με αυτή η προσέγγιση επιτρέπει την κλιμάκωση της πλατφόρμας σε επίπεδο υποδομής, επίσης διευκολύνει την ενσωμάτωση νέων λειτουργιών και την αντιμετώπιση προβλημάτων, χωρίς να επηρεάζεται ολόκληρο το σύστημα [15]. Για την καλύτερη αποθήκευση και διαχείριση των δεδομένων της, η OpenAlex χρησιμοποιεί κατακευκτωμένες βάσεις δεδομένων. Αυτό έχει ως επακόλουθο τα δεδομένα να είναι πάντα διαθέσιμα και μπορούν να ανακτηθούν γρήγορα, ακόμη και σε περιπτώσεις αυξημένου φόρτου εργασίας ή σε περίπτωση αποτυχίας κάποιου μέρους του συστήματος. Επιπροσθέτως η κατακευκτωμένη βάση δεδομένων είναι κρίσιμη για

την υποστήριξη μεγάλης κλίμακας δεδομένων που διαχειρίζεται η πλατφόρμα, δίνοντας την δυνατότητα αποθήκευσης εκατομμυρίων εγγραφών και τη γρήγορη πρόσβαση σε αυτές.



Σχήμα 2.3: Δομή Δεδομένων OpenAlex [16]

2.4.2 API της OpenAlex

Ένα από τα κυριότερα εργαλεία που προσφέρει η πλατφόρμα για την πρόσβαση και την ανάλυση των δεδομένων της, είναι το API (Application Programming Interface). Το API είναι RESTful, το οποίο έχει ως επακόλουθο ότι βασίζεται σε πρωτόκολλα HTTP και χρησιμοποιεί τις κοινές μεθόδους HTTP, όπως GET, POST, PUT και DELETE, για να επιτρέπει στους χρήστες να αλληλοεπιδρούν με τα δεδομένα. Μέσω του API της OpenAlex παρέχεται πρόσβαση σε όλες τις βασικές οντότητες της πλατφόρμας, δηλαδή τα έργα (works), τους συγγραφείς (authors), τα περιοδικά και συνέδρια (venues), τα ιδρύματα (institutions) και τις θεματικές κατηγορίες (concepts). Ένας ερευνητής μπορεί παράγει σύνθετες αναζητήσεις, να φιλτράρει τα αποτελέσματα σύμφωνα με τα προσωπικά τους κριτήρια και να αντλεί δεδομένα για περαιτέρω ανάλυση σε μορφή JSON [15]. Ένα χαρακτηριστικό του API της OpenAlex που την κάνει να ξεχωρίζει είναι η δυνατότητα μαζικής ανάκτησης δεδομένων, επιτρέποντας τους χρήστες να αντλούν μεγάλες ποσότητες δεδομένων σε μία μόνο κλήση. Το συγκεκριμένο χαρακτηριστικό είναι ιδιαίτερα χρήσιμο για ερευνητές που χρειάζονται ολοκληρωμένα σύνολα δεδομένων για τις μελέτες τους.

```

{
  "meta": {
    "count": 23684895,
    "db_response_time_ms": 13,
    "page": 1,
    "per_page": 25
  },
  "results": [
    {
      "id": "https://openalex.org/works/98221133462",
      "doi": null,
      "titles": "L'essai Trinity (1945) et les risques d'embranchement de l'atmosphère : entre épévation SF, modélisation",
      "display_name": "L'essai Trinity (1945) et les risques d'embranchement de l'atmosphère : entre épévation SF, mod",
      "publication_year": 2029,
      "publication_date": "2029-11-01",
      "openalex": "https://openalex.org/98221133462",
      "best_venue": {
        "id": null,
        "issn": null,
        "issue": null,
        "display_name": null,
        "publication": null,
        "type": null,
        "url": "https://hal.archives-ouvertes.fr/hal-02367541",
        "is_ojs": null,
        "version": null,
        "license": null
      }
    }
  ]
}

```

Σχήμα 2.4: Επισκόπηση API OpenAlex [17]

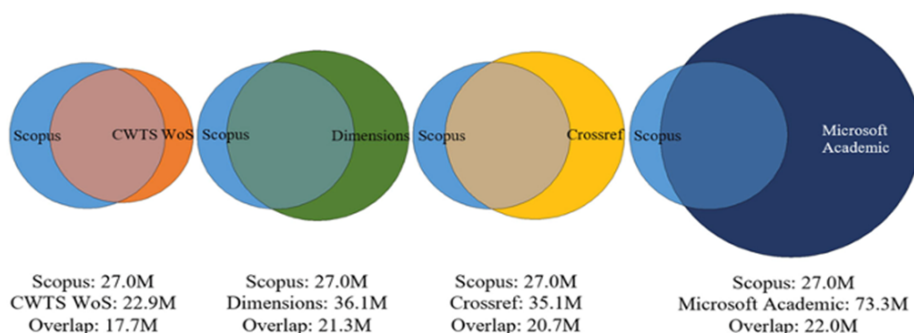
2.4.3 Πρόσβαση στα Δεδομένα της OpenAlex

Ο τρόπος με τον οποίο έχει σχεδιαστεί η OpenAlex γίνεται όσο το δυνατόν πιο προσιτή, προσφέρει πολλαπλές μεθόδους πρόσβασης στα δεδομένα της, πέρα από το API που διαθέτει, οι ερευνητές μπορούν να κατεβάσουν πλήρη σύνολα δεδομένων από την επίσημη ιστοσελίδα της. Τα σύνολα αυτά είναι διαθέσιμα σε μορφή JSON και μπορούν να χρησιμοποιηθούν σε πλατφόρμες ανάλυσης μεγάλων δεδομένων όπως το Hadoop ή το Apache Spark. Ακόμη, υποστηρίζεται η χρήση εργαλείων όπως το Google BigQuery για την εκτέλεση πολύπλοκων ερωτήσεων και την ανάλυση των δεδομένων σε πραγματικό χρόνο. Το συγκεκριμένο ένα από τα πιο ισχυρά εργαλεία για την ανάλυση μεγάλων δεδομένων, καθώς επιτρέπει την επεξεργασία μεγάλων ποσοτήτων δεδομένων σε υψηλή ταχύτητα, απαραίτητο για την ανάλυση και την εξαγωγή χρήσιμων συμπερασμάτων από τα βιβλιομετρικά δεδομένα της OpenAlex [18].

2.5 Σύγκριση της OpenAlex με Άλλες Βάσεις Δεδομένων

2.5.1 Ανάλυση και Σύγκριση με Scopus και Web of Science

Ενώ το Scopus και το Web of Science χαρακτηρίζονται ως από κορυφαίες βάσεις δεδομένων παρόλο που απαιτούν συνδρομή για πρόσβαση, η OpenAlex προσφέρει ελεύθερη και ανοιχτή πρόσβαση, καθιστώντας τα δεδομένα της προσβάσιμα σε ένα ευρύτερο κοινό. Είναι μια πρωτοποριακή ανοιχτή εναλλακτική σε παραδοσιακές βάσεις δεδομένων όπως το Scopus και το Web of Science, ενώ βασική της διαφορά είναι η ελεύθερη πρόσβαση και η φιλοσοφία που διακρίνεται πίσω από τη διαχείριση των δεδομένων. Το Scopus και το Web of Science διακρίνονται για την υψηλή ποιότητα και την αυστηρή επιλογή των πηγών τους, προσφέροντας αξιοπιστία στα βιβλιομετρικά δεδομένα που χρησιμοποιούνται ευρέως για ακαδημαϊκές αξιολογήσεις. Από την άλλη πλευρά, η OpenAlex, αξιοποιώντας μια ποικιλία πηγών, όπως το CrossRef, παρέχει δεδομένα που είναι πιο προσαρμόσιμα και διασυνδεδεμένα, επιτρέποντας πιο ευέλικτες και καινοτόμες χρήσεις των δεδομένων σε διάφορα ερευνητικά πλαίσια [19].



Σχήμα 2.5: Σύγκριση Δεδομένων Ανάμεσα σε Βάσεις Δεδομένων [20]

2.5.2 Πλεονεκτήματα και Μειονεκτήματα της OpenAlex

1. Πλεονεκτήματα

- **Ανοιχτή Πρόσβαση:** Η ανοιχτή πρόσβαση στα δεδομένα της είναι ένα από τα μεγαλύτερα πλεονεκτήματα της OpenAlex. Επιτρέπει σε οποιονδήποτε να εξετάσει, να αναλύσει και να χρησιμοποιήσει τα δεδομένα χωρίς περιορισμούς.
- **Προσαρμοστικότητα:** Η OpenAlex διαθέτει ένα εξαιρετικά ευέλικτο API, το οποίο παρέχει στους χρήστες την δυνατότητα εκτέλεσης σύνθετων αναζητήσεων καθώς και την άντληση δεδομένων τα οποία είναι προσαρμοσμένα με βάση τις ανάγκες του την εκάστοτε στιγμή.
- **Ενσωμάτωση Πολλαπλών Πηγών:** Η OpenAlex καταφέρνει και συγκεντρώνει δεδομένα από πολλαπλές πηγές, γεγονός που προσφέρει μια πιο ολοκληρωμένη εικόνα της ακαδημαϊκής δραστηριότητας [21].
- **Ενσωμάτωση Πολλαπλών Πηγών:** Η OpenAlex καταφέρνει και συγκεντρώνει δεδομένα από πολλαπλές πηγές, γεγονός που προσφέρει μια πιο ολοκληρωμένη εικόνα της ακαδημαϊκής δραστηριότητας.

2. Μειονεκτήματα

- **Αξιοπιστία:** Η αξιοπιστία και η ακρίβεια των βιβλιομετρικών δεδομένων της, για την OpenAlex είναι βασική μεριμνά, παράλληλα με τη ποικιλομορφία των πηγών της μπορεί να οδηγήσει σε διαφορές στην ποιότητα των δεδομένων σε σύγκριση με τις πιο αυστηρές παραδοσιακές βάσεις δεδομένων όπως το Scopus και το Web of Science [7].
- **Υποστήριξη:** Παρά τα ήδη αναφερθέντα πλεονεκτήματα της OpenAlex σε σχέση με πιο παραδοσιακές βάσεις δεδομένων που διαθέτουν συχνή και εκτεταμένη υποστήριξη πελατών καθώς παρέχουν αρκετά εργαλεία ανάλυσης. Παρατηρείται πως ορισμένα από τα χαρακτηριστικά μιας σχεσιακής βάσης λείπουν (ως το 2024) την παρούσα φάση από την OpenAlex και το API της [22].

2.5.3 Εφαρμογές και Πρακτικά Παραδείγματα Χρήσης

Η OpenAlex χρησιμοποιείται σε πολλαπλά ερευνητικά πλαίσια για την ανάλυση της επιστημονικής παραγωγής και επιρροής. Παραδείγματα περιλαμβάνουν:

- **Μελέτες Επιστημονικής Συνεργασίας:** Χρησιμοποιώντας τα δεδομένα της OpenAlex, οι ερευνητές μπορούν να αναλύσουν τα μοτίβα συνεργασίας μεταξύ διαφορετικών ερευνητικών ιδρυμάτων και συγγραφέων, εντοπίζοντας κενά και ευκαιρίες για μελλοντική συνεργασία.
- **Ανάλυση Θεματικών Τάσεων:** Η OpenAlex επιτρέπει την ανάλυση των τάσεων στην έρευνα καθώς παρέχει την δυνατότητα κατηγοριοποίησης των έργων της σε θεματικές ενότητες, εντοπίζοντας ποια θέματα είναι ανά κάθε περίοδο πιο δημοφιλή στο κοινό καθώς και ποια παράγουν πιο δημοφιλή και ποια παρουσιάζουν αυξανόμενο ενδιαφέρον [23].
- **Αξιολόγηση της Ερευνητικής Παραγωγικότητας:** Οι μετρήσεις που παρέχει η OpenAlex μπορούν να χρησιμοποιηθούν για την αξιολόγηση της απόδοσης και της παραγωγικότητας μεμονωμένων ερευνητών ή ολόκληρων ιδρυμάτων καθώς και κάθε owner του κάθε έργου για το ενδιαφέρον του κοινού στο έργο του [23].

2.6 Βιβλιομετρικά Δεδομένα και Ηθικές Σκέψεις

2.6.1 Ηθικά Ζητήματα στη Χρήση Βιβλιομετρικών Δεδομένων

Μια παράμετρος ακόμα την οποία πρέπει να αναλύσουμε είναι τα ηθικά ζητήματα που εγείρονται από την χρήση βιβλιομετρικών δεδομένων, όπως οι αναφορές και οι δείκτες επιρροής. Ένα από τα κυριότερα ηθικά ζητήματα είναι η υπερβολική εξάρτηση από τους βιβλιομετρικούς δείκτες για την αξιολόγηση της ποιότητας της έρευνας καθώς οι δείκτες αυτοί, παρόλο που προσφέρουν σημαντικές πληροφορίες, έρχονται μαζί και με κάποιες αδυναμίες που μπορούν να οδηγήσουν σε παραμορφωμένες εικόνες της επιστημονικής αξίας καθώς και την εξαγωγή λάθος συμπερασμάτων [24]. Η OpenAlex προσφέρει μια εναλλακτική προσέγγιση που ενθαρρύνει την ανοιχτή επιστήμη και την υπεύθυνη χρήση των βιβλιομετρικών δεδομένων. Παρόλα αυτά, ασχέτως από την ανοιχτή πρόσβαση που προσφέρει στα δεδομένα της οι χρήστες της πρέπει να συνεχίσουν να είναι επιφυλακτικοί στη χρήση και στην ανάλυση αυτών, καθώς πρέπει να λαμβάνουν συνεχώς υπόψιν τους τις ηθικές προεκτάσεις και τις πιθανές συνέπειες που μπορεί να έχει η χρήση των βιβλιομετρικών δεικτών στη λήψη αποφάσεων καθώς και να διαθέτουν κριτική σκέψη όσον αφορά στην εμπιστοσύνη τους στα αποτελέσματα [3].

2.6.2 Ζητήματα Διαφάνειας και Ανοιχτής Πρόσβασης

Στον τομέα της βιβλιομετρίας, η διαφάνεια αποτελεί κεντρικό ζήτημα. Η προσβάσιμη ανάλυση καθώς και η αξιοποίηση όλων των δεδομένων πρέπει να γίνεται με τρόπους που προάγουν την ανοιχτή επιστήμη και την ισότιμη διάδοση της γνώσης [25]. Η OpenAlex προωθεί τη διαφάνεια προσφέροντας ανοιχτή πρόσβαση στα δεδομένα της, επιτρέποντας στους χρήστες να εξετάσουν και να επικυρώσουν τα δεδομένα που χρησιμοποιούνται στις αναλύσεις τους. Ωστόσο, η ανοιχτή πρόσβαση συνοδεύεται και από την ανάγκη για υπεύθυνη διαχείριση των δεδομένων. Οι χρήστες της OpenAlex πρέπει να διασφαλίζουν ότι η χρήση των δεδομένων γίνεται με σεβασμό στην επιστημονική ακεραιότητα και ότι οι αναλύσεις τους αντικατοπτρίζουν δίκαιες και ακριβείς αξιολογήσεις της επιστημονικής παραγωγής [26].

2.6.3 Επιπτώσεις των Βιβλιομετρικών Αναλύσεων στην Επιστημονική Κοινότητα

Η επιστημονική κοινότητα έχει επηρεαστεί σημαντικά από τις βιβλιομετρικές αναλύσεις. Οι δείκτες όπως οι αναφορές και ο impact factor μπορούν να επηρεάσουν αποφάσεις σχετικά με τη χρηματοδότηση, την πρόσληψη και την προώθηση ακαδημαϊκών έργων, καθώς και τη φήμη των ερευνητικών ιδρυμάτων διότι μέσω των αναλύσεων και των εργαλείων που χρησιμοποιούνται, όπως προαναφέραμε και παραπάνω οδηγούμαστε σε συμπεράσματα τα οποία με βάση την κριτική σκέψη του καθενός θα πρέπει να υπολογίζουμε τον βαθμό εμπιστοσύνης μας σε αυτά [27]. Η OpenAlex, προσφέροντας ανοιχτή πρόσβαση σε αυτά τα δεδομένα, συμβάλλει στη δημοσιοποίηση της γνώσης αυτής, αλλά ταυτόχρονα απαιτείται προσοχή ώστε η χρήση των βιβλιομετρικών δεδομένων να γίνεται με τρόπο που προάγει την ακριβοδίκαιη αξιολόγηση και την προώθηση της επιστήμης. Η κατάχρηση ή η λανθασμένη χρήση των βιβλιομετρικών δεικτών μπορεί να έχει αρνητικές συνέπειες, όπως η υποβάθμιση της αξίας της πρωτοτυπίας και της καινοτομίας στην έρευνα, καθώς και η ενίσχυση της ανισότητας μεταξύ διαφορετι-

κών ερευνητικών κοινοτήτων [27]. Η OpenAlex προωθεί μια πιο ολιστική προσέγγιση, ενσωματώνοντας πολλαπλά δεδομένα και προωθώντας τη διαφάνεια, ώστε να μετριάσουν αυτές οι αρνητικές επιπτώσεις.

2.7 Προοπτικές και Μελλοντικές Κατευθύνσεις

2.7.1 Αναδυόμενες Τεχνολογίες και ο Ρόλος της OpenAlex

Οι εξελίξεις στην τεχνολογία, όπως η τεχνητή νοημοσύνη (AI), η μηχανική μάθηση (ML), και η ανάλυση μεγάλων δεδομένων (Big Data Analytics), έχουν αρχίσει να επηρεάζουν δραστικά την ανάλυση των βιβλιομετρικών δεδομένων. Η OpenAlex είναι σε θέση να αξιοποιήσει αυτές τις τεχνολογίες για να βελτιώσει την ακρίβεια, την απόδοση και την ευχρηστία των υπηρεσιών της. Για παράδειγμα, η μηχανική μάθηση μπορεί να χρησιμοποιηθεί για να εντοπίσει νέα επιστημονικά πεδία που αναδύονται ή να αναλύσει τα πρότυπα συνεργασίας μεταξύ ερευνητών και ιδρυμάτων. Επίσης, η ανάλυση μεγάλων δεδομένων επιτρέπει την ταχύτερη επεξεργασία και ανάλυση των δεδομένων, παρέχοντας στους χρήστες αποτελέσματα σε πραγματικό χρόνο [28]. Η OpenAlex, με την ανοιχτή και προσαρμόσιμη φύση της, είναι σε θέση να ενσωματώσει αυτές τις τεχνολογίες και να συνεχίσει να καινοτομεί στην παροχή βιβλιομετρικών υπηρεσιών.

2.7.2 Προκλήσεις και Ευκαιρίες για τα Βιβλιομετρικά Δεδομένα

Η διαχείριση και η ανάλυση βιβλιομετρικών δεδομένων γεννάει πολλαπλές προκλήσεις, αλλά ταυτόχρονα προσφέρει σημαντικές ευκαιρίες. Μία από τις κύριες προκλήσεις είναι η διασφάλιση της ποιότητας των δεδομένων, ειδικά όταν αυτά προέρχονται από πολλαπλές και διαφορετικές πηγές. Η OpenAlex, μέσω της ανοιχτής πρόσβασης και της διαφάνειας, προσπαθεί να αντιμετωπίσει αυτήν την πρόκληση, αλλά απαιτείται συνεχής παρακολούθηση και βελτίωση των δεδομένων της καθώς και επικαιροποίηση αυτών. Επίσης επιβάλλεται η συχνή μελέτη ως προς το σχήμα της βάσης και των σχέσεων που δημιουργούνται ώστε η ανάλυση μέσω του API να είναι αποδοτικότερη και να παρέχει πιο ακριβή αποτελέσματα για τον χρήστη [29]. Ταυτόχρονα, η OpenAlex προσφέρει ευκαιρίες για καινοτομία και ανάπτυξη νέων εργαλείων που μπορούν να βελτιώσουν την ανάλυση και την αξιοποίηση των βιβλιομετρικών δεδομένων. Η ανοιχτή φύση της πλατφόρμας επιτρέπει την εύκολη ενσωμάτωση νέων μεθόδων ανάλυσης, δημιουργώντας νέες δυνατότητες για ερευνητές και αναλυτές δεδομένων. Για παράδειγμα ένας αναλυτής δεδομένων θα μπορούσε να εξάγει συμπεράσματα, ενώνοντας την κριτική του σκέψη καθώς και τις αναλυτικές του ικανότητες ώστε να αποκομίσει ακριβή στοιχεία για το κάθε έργο πάνω στο οποίο εργάζεται [30].

2.7.3 Επεκτάσεις, Μελλοντικές Βελτιώσεις της Πλατφόρμας OpenAlex και τωρινοί περιορισμοί

Η OpenAlex πάρα την θέση της ως ένα ισχυρό εργαλείο στον τομέα της ανάλυσης βιβλιομετρικών δεδομένων, όπως είναι λογικό υπάρχουν και πολλά περιθώρια για περαιτέρω βελτιώσεις και επεκτάσεις. Ένα από τα πιθανά κομμάτια βελτίωσης θα μπορούσε να είναι η ενσωμάτωση περισσότερων βιβλιομετρικών δεικτών και η παροχή πιο λεπτομερών αναφορών στον

χρήστη. Επιπλέον, η πλατφόρμα μπορεί να επεκταθεί για να καλύψει νέες επιστημονικές περιοχές ή να βελτιώσει τις υπάρχουσες, προσφέροντας έτσι μια πιο ολοκληρωμένη εικόνα της ακαδημαϊκής δραστηριότητας. Τέλος, η βελτίωση του API για τον χρήστη καθώς και η παροχή περισσότερων εργαλείων ανάλυσης δεδομένων μέσα στην πλατφόρμα θα μπορούσαν να αυξήσουν τη χρηστικότητα και την ελκυστικότητα της OpenAlex σε ένα ευρύτερο κοινό, συμπεριλαμβανομένων των μη εξειδικευμένων χρηστών. Οι παραπάνω επεκτάσεις και πιθανές μελλοντικές βελτιώσεις θα μπορούσαν να συνεισφέρουν σημαντικά σε θέματα πολυπλοκότητας των σχέσεων της υπάρχουσας σχεδίασης. Κυρίως στο φαινόμενο των κυκλικών σχέσεων το οποίο οδηγεί σε ασυνέπειες στα δεδομένα καθώς και σε προβλήματα ομαλοποίησης.

Κεφάλαιο 3

Εισαγωγή στη Neo4j

3.1 Εισαγωγή

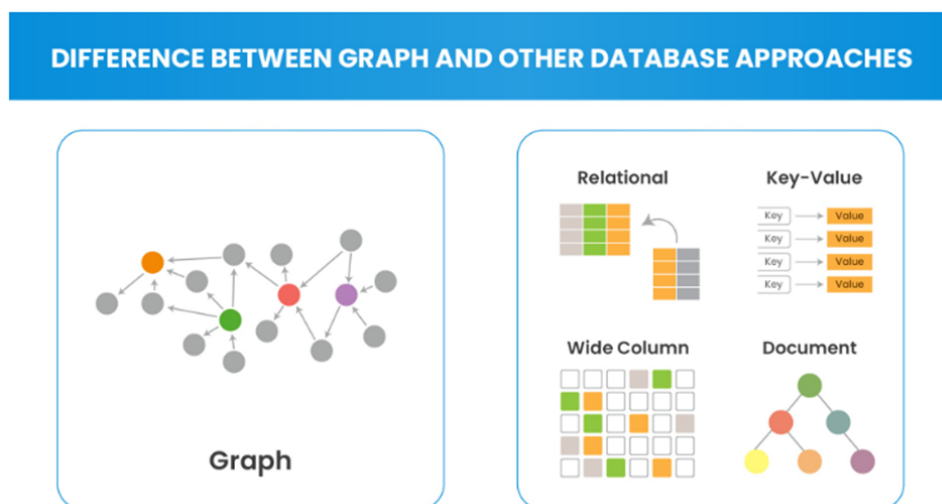
Η Neo4j αποτελεί μια βάση δεδομένων γραφημάτων, η οποία αναπτύχθηκε για να αντιμετωπίσει τις αυξανόμενες ανάγκες για την αποθήκευση και διαχείριση δεδομένων. Η ανάπτυξη της ξεκίνησε τη δεκαετία του 2000 από την Neo Technology, με την επίσημη κυκλοφορία της πρώτης έκδοσης να γίνεται το 2007. Μετά την πρώτη της κυκλοφορία, η Neo4j έχει εξελιχθεί σημαντικά και αποτελεί την πιο διαδεδομένη βάση δεδομένων γραφημάτων. Εκατοντάδες οργανισμοί και επιχειρήσεις να την χρησιμοποιούν για την αποθήκευση και ανάλυση δεδομένων που σχετίζονται με γραφήματα [31]. Βασικός στόχος της Neo4j από την αρχή ήταν να προσφέρει μια βάση δεδομένων που να αποδίδει εξαιρετικά σε περιβάλλοντα όπου οι παραδοσιακές σχεσιακές βάσεις δεδομένων (SQL) αποδεικνύονται ανεπαρκείς. Οι βάσεις δεδομένων γραφημάτων, όπως η Neo4j, είναι ιδανικές για την αποθήκευση δεδομένων που αφορούν σύνθετες σχέσεις, όπως κοινωνικά δίκτυα, συστάσεις προϊόντων, και βιολογικές σχέσεις. Για την αποθήκευση κόμβων (nodes), ακμών (relationships), και ιδιοτήτων (properties), η Neo4j βασίζεται σε μια μηχανή αποθήκευσης που είναι βελτιστοποιημένη. Χρησιμοποιώντας αυτή τη σχεδίαση της δίνεται η δυνατότητα να εκτελεί ερωτήματα σε πραγματικό χρόνο και να εξίσου αποδοτική ακόμα και όταν οι σχέσεις μεταξύ των δεδομένων είναι εξαιρετικά πολύπλοκες [32]. Ορισμός και Σημασία της ως Βάση Δεδομένων Γραφημάτων Η Neo4j, όπως προαναφέρθηκε, επικεντρώνεται στην αποθήκευση και διαχείριση δεδομένων που παρουσιάζονται ως γραφήματα. Τα γραφήματα ορίζονται ως δομές δεδομένων που αποτελούνται από κόμβους και ακμές, όπου οι κόμβοι αντιπροσωπεύουν οντότητες και οι ακμές τις σχέσεις μεταξύ αυτών των οντοτήτων. Αυτό που κάνει τη Neo4j να ξεχωρίζει είναι η ικανότητά της να αποθηκεύει δεδομένα με φυσικό τρόπο, επιτρέποντας έτσι τη γρήγορη και αποδοτική αναζήτηση και ανάλυση των σχέσεων μεταξύ αυτών των δεδομένων. Αντίθετα με τις παραδοσιακές σχεσιακές βάσεις δεδομένων, οι βάσεις δεδομένων γραφημάτων είναι ιδιαίτερα σημαντικές σε περιπτώσεις όπου οι σχέσεις μεταξύ των δεδομένων είναι εξίσου ή περισσότερο σημαντικές από τα ίδια τα δεδομένα. Σε παραδοσιακές σχεσιακές βάσεις δεδομένων, οι σχέσεις μεταξύ των δεδομένων αποθηκεύονται σε πίνακες, κάτι που μπορεί να γίνει εξαιρετικά περίπλοκο και αναποτελεσματικό όταν οι σχέσεις είναι πολλές και σύνθετες. Στη Neo4j οι σχέσεις είναι πρώτης τάξης οντότητες, το οποίο σημαίνει ότι αποθηκεύονται και ανακτώνται άμεσα, χωρίς να απαιτείται πολυεπίπεδη σύνδεση πινάκων (joins) [33].

3.2 Δομή και Τεχνικά Χαρακτηριστικά της Neo4j

Σε αυτό το κεφάλαιο, θα εμβαθύνουμε αναλυτικά στη δομή και τα τεχνικά χαρακτηριστικά της Neo4j. Η κατανόηση των θεμελιωδών στοιχείων είναι απαραίτητη για να αξιολογήσουμε τον τρόπο που η Neo4j μπορεί να εφαρμοστεί βέλτιστα για τη δημιουργία και τη διαχείριση βιβλιομετρικών βάσεων δεδομένων.

3.2.1 Αρχιτεκτονική Λογισμικού της Neo4j

Η Neo4j ακολουθεί μια αρχιτεκτονική client-server, το οποίο σημαίνει πως ο πυρήνας της βάσης δεδομένων εκτελείται σε έναν κεντρικό server, ενώ οι πελάτες (clients) επικοινωνούν με τον server μέσω ενός RESTful API ή μέσω εγγενούς σύνδεσης (Bolt protocol). Αυτή η αρχιτεκτονική της επιτρέπει να αναπτύξει κατακεντρωμένες εφαρμογές και να υποστηρίζει τη χρήση της Neo4j σε περιβάλλοντα με πολλούς χρήστες [31].



Σχήμα 3.1: Διαφορετικές Προσεγγίσεις Μεταξύ Βάσεων Δεδομένων [34]

Η αρχιτεκτονική της Neo4j περιλαμβάνει τα παρακάτω κύρια στοιχεία:

- **Kernel (Πυρήνας):** Για την αποθήκευση και την ανάκτηση δεδομένων, τη διαχείριση των συναλλαγών και την εκτέλεση των ερωτημάτων υπεύθυνος είναι ο πυρήνας. Ο πυρήνας είναι σχεδιασμένος για να είναι αποδοτικός και να επιτρέπει τη γρήγορη πρόσβαση στα δεδομένα.
- **Storage Engine (Μηχανή Αποθήκευσης):** Η μηχανή αποθήκευσης αποτελεί το υποσύστημα της Neo4j που διαχειρίζεται την αποθήκευση των κόμβων, των ακμών, και των ιδιοτήτων σε ακόλουθα αρχεία. Με σκοπό να βελτιστοποιήσει της απόδοση των ερωτημάτων γραφημάτων, χρησιμοποιεί έναν δομημένο τρόπο αποθήκευσης.
- **Transaction Manager (Διαχειριστής Συναλλαγών):** Ο διαχειριστής είναι υπεύθυνος να διασφαλίζει ότι όλες οι συναλλαγές στη βάση δεδομένων είναι ACID-συμβατές. Αυτό

σημαίνει ότι κάθε συναλλαγή είτε θα ολοκληρωθεί επιτυχώς είτε, σε περίπτωση αποτυχίας, θα αναιρεθεί πλήρως, διατηρώντας την ακεραιότητα της βάσης δεδομένων.

- **Graph Database Engine (Μηχανή Γραφημάτων):** Η μηχανή γραφημάτων, εκτελεί τους αλγόριθμους που απαιτούνται για την πλοήγηση και την ανάλυση των σχέσεων στο γράφημα. Είναι, λοιπόν, υπεύθυνη για την εκτέλεση ερωτημάτων και την επεξεργασία των δεδομένων γραφημάτων.
- **Query Optimizer (Βελτιστοποιητής Ερωτημάτων):** Ο βελτιστοποιητής ερωτημάτων αναλύει τα ερωτήματα που υποβάλλονται από τους χρήστες και επιλέγει την καλύτερη δυνατή στρατηγική εκτέλεσης με βάση τη διάρθρωση του γραφήματος και τα διαθέσιμα ευρετήρια.
- **Clustering and Replication:** Το clustering υποστηρίζεται από τη Neo4j, επιτρέποντας την κατανομή των δεδομένων σε πολλαπλούς κόμβους. Αυτό έχει ως επακόλουθο τη βελτίωση, την επεκτασιμότητα και την ανθεκτικότητα, καθώς οι κόμβοι του cluster μπορούν να επαναλάβουν δεδομένα για να εξασφαλίσουν τη διαθεσιμότητα και την ασφάλεια των δεδομένων [35].

3.2.2 Δομή των Κόμβων, των Ακμών και των Ιδιοτήτων

Η βασική δομή της Neo4j αποτελείται από κόμβους, ακμές, και ιδιότητες, τα οποία είναι τα θεμελιώδη στοιχεία ενός γραφήματος:

1. **Κόμβοι (Nodes):** Αντιπροσωπεύουν οντότητες στο γράφημα, όπως άτομα, αντικείμενα ή έννοιες. Κάθε κόμβος μπορεί να έχει ένα ή περισσότερα labels (ετικέτες) που τον κατηγοριοποιούν. Για παράδειγμα, ένας κόμβος μπορεί να έχει την ετικέτα "Person" ή "Product". Οι ετικέτες επιτρέπουν την εύκολη κατηγοριοποίηση και αναζήτηση των κόμβων στο γράφημα.
2. **Ακμές (Relationships):** Αντιπροσωπεύουν τις σχέσεις μεταξύ των κόμβων. Κάθε ακμή έχει μια κατεύθυνση και έναν τύπο (type), όπως "FRIEND OF", "BOUGHT BY", ή "AUTHORED BY". Οι ακμές μπορούν επίσης να έχουν ιδιότητες, οι οποίες αποθηκεύουν επιπλέον δεδομένα σχετικά με τη σχέση, όπως η ημερομηνία που δημιουργήθηκε η 2σχέση ή η δύναμη της σχέσης.
3. **Ιδιότητες (Properties):** Τόσο οι κόμβοι όσο και οι ακμές μπορούν να έχουν ιδιότητες, οι οποίες αποθηκεύουν δεδομένα σε μορφή key-value pairs. Οι ιδιότητες μπορούν να αποθηκεύουν διάφορους τύπους δεδομένων, όπως αριθμούς, συμβολοσειρές.

Η παραπάνω δομή επιτρέπει στη Neo4j να αποθηκεύει πολύπλοκα δίκτυα δεδομένων με φυσικό τρόπο, προσφέροντας εύκολη και γρήγορη ανάκτηση των δεδομένων και των σχέσεων μεταξύ τους [32].

3.2.3 Αποθήκευση Δεδομένων στη Neo4j

Ο τρόπος αποθήκευσης δεδομένων στη Neo4j είναι βελτιστοποιημένος για την αποθήκευση γραφημάτων, και γίνεται σε ακόλουθα αρχεία, τα οποία εξασφαλίζουν την ταχεία ανάκτηση των δεδομένων και τη διατήρηση της συνέπειας. Κάθε κόμβος, ακμή, και ιδιότητα που αποθηκεύεται έχει ένα μοναδικό αναγνωριστικό (ID), το οποίο χρησιμοποιείται για την άμεση πρόσβαση στα δεδομένα. Η δομή αποθήκευσης της Neo4j περιλαμβάνει τα παρακάτω στοιχεία:

- **Node Store:** Αποθηκεύει το σύνολο των δεδομένων των κόμβων, συμπεριλαμβανομένων των αναγνωριστικών τους, των labels, και των αναφορών στις ιδιότητες.
- **Relationship Store:** Αποθηκεύει τα δεδομένα των ακμών (σχέσεων), συμπεριλαμβανομένων των αναγνωριστικών τους, των τύπων των σχέσεων, και των αναφορών στους κόμβους που συνδέονται από τη σχέση.
- **Property Store:** Αποθηκεύει τις ιδιότητες που συνδέονται με τους κόμβους και τις ακμές. Κάθε ιδιότητα έχει ένα αναγνωριστικό, έναν τύπο δεδομένων, και μια τιμή.
- **Schema Store:** Αποθηκεύει πληροφορίες σχετικά με τα σχήματα της βάσης δεδομένων, όπως τα ευρετήρια και τους περιορισμούς [32].

Στο σημείο αυτό να τονίσω ότι η Neo4j χρησιμοποιεί επίσης μια σειρά από ευρετήρια με σκοπό την βελτιστοποίηση της ταχύτητας εκτέλεσης των ερωτημάτων. Συγκεκριμένα, περιλαμβάνουν ευρετήρια για κόμβους με συγκεκριμένα labels, για σχέσεις με συγκεκριμένους τύπους, και για κόμβους ή σχέσεις με συγκεκριμένες ιδιότητες.

3.2.4 Βελτιστοποίηση Απόδοσης Ερωτημάτων

Για την αποδοτική λειτουργία της Neo4j είναι κρίσιμη η βελτιστοποίηση των ερωτημάτων, κυρίως όταν η βάση δεδομένων περιέχει μεγάλα γραφήματα με εκατομμύρια κόμβους και σχέσεις. Η Neo4j χρησιμοποιεί έναν συνδυασμό τεχνικών βελτιστοποίησης που περιλαμβάνουν:

1. **Ευρετήρια (Indexes):** Η χρήση ευρετηρίων επιταχύνει την ανάκτηση κόμβων και σχέσεων με συγκεκριμένα χαρακτηριστικά. Μπορεί να μειώσει σημαντικά τον χρόνο εκτέλεσης ενός ερωτήματος, ειδικά όταν το ερώτημα περιλαμβάνει την αναζήτηση δεδομένων με βάση συγκεκριμένα κριτήρια.
2. **Μηχανή Γραφημάτων (Traversal Optimization):** Η μηχανή γραφημάτων της Neo4j βελτιώνει την πλοήγηση στο γράφημα. Χρησιμοποιεί τεχνικές όπως η χρήση ευρετηρίων για την αρχική εύρεση των κόμβων, και η χρήση αλγορίθμων πλοήγησης που αποφεύγουν την εξερεύνηση αχρείαστων τμημάτων του γραφήματος.
3. **Οκνηρή Φόρτωση (Lazy Loading):** Η Neo4j χρησιμοποιεί οκνηρή φόρτωση για να αποφεύγει την ανάκτηση δεδομένων που δεν είναι άμεσα απαραίτητα. Τα δεδομένα, λοιπόν, φορτώνονται μόνο όταν χρειάζονται, μειώνοντας, έτσι τον φόρτο στη μνήμη και επιταχύνοντας την εκτέλεση των ερωτημάτων.

4. Σχεδιαστής Ερωτημάτων Με Βάση το Κόστος (Cost-Based Query Planner): Ο σχεδιαστής ερωτημάτων της Neo4j έχει τη δυνατότητα να υπολογίζει το κόστος κάθε πιθανού πλάνου εκτέλεσης ενός ερωτήματος και να επιλέγει το πλάνο με το χαμηλότερο κόστος. Το κόστος αυτό λαμβάνει υπόψιν παράγοντες όπως ο αριθμός των κόμβων και των σχέσεων που πρέπει να εξεταστούν, καθώς και το μέγεθος των δεδομένων που πρέπει να φορτωθούν στη μνήμη [35].

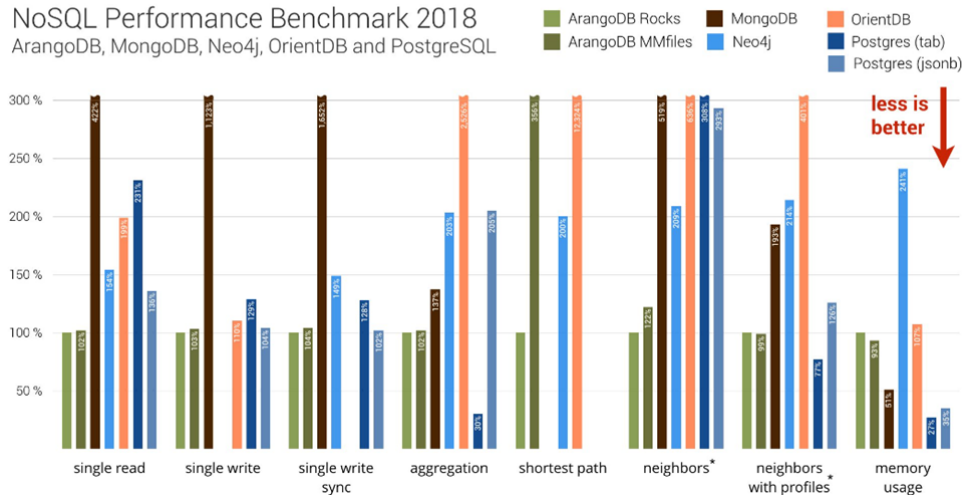
3.2.5 Ασφάλεια και Διαχείριση Δεδομένων

Η Neo4j παρέχει εκτενείς δυνατότητες για τη διασφάλιση της ασφάλειας των δεδομένων και τη διαχείριση της πρόσβασης. Αυτές οι δυνατότητες περιλαμβάνουν:

1. Σύστημα Ρόλων (RBAC): Για την διαχείριση της πρόσβασης στη Neo4j χρησιμοποιείται ένα σύστημα ρόλων, στο οποίο κάθε χρήστης ή ομάδα χρηστών μπορεί να αντιστοιχιστεί σε έναν ρόλο με συγκεκριμένα δικαιώματα πρόσβασης. Αυτό επιτρέπει τον περιορισμό της πρόσβασης σε συγκεκριμένα δεδομένα ή λειτουργίες της βάσης δεδομένων.
2. Έλεγχος Πρόσβασης σε Επίπεδο Δεδομένων: Η Neo4j υποστηρίζει τον έλεγχο πρόσβασης σε επίπεδο δεδομένων. Επιτρέπει τη ρύθμιση δικαιωμάτων πρόσβασης σε συγκεκριμένους κόμβους ή σχέσεις. Αυτό εξασφαλίζει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να δουν ή να τροποποιήσουν συγκεκριμένα δεδομένα.
3. Κρυπτογράφηση Δεδομένων: Η Neo4j υποστηρίζει την κρυπτογράφηση των δεδομένων τόσο κατά τη μεταφορά (in transit) όσο και σε κατάσταση ηρεμίας (at rest). Με αυτόν τον τρόπο διασφαλίζεται ότι τα δεδομένα είναι προστατευμένα από μη εξουσιοδοτημένη πρόσβαση, ακόμα και αν κάποιος αποκτήσει φυσική πρόσβαση στα αποθηκευτικά μέσα ή ανακτήσει δεδομένα από το δίκτυο.
4. Auditing (Λογιστικός Έλεγχος): Η Neo4j παρέχει εργαλεία για την καταγραφή και την ανάλυση των δραστηριοτήτων των χρηστών στη βάση δεδομένων. Αυτές οι δυνατότητες είναι κρίσιμες για τη διασφάλιση της συμμόρφωσης με κανονισμούς και την ανίχνευση πιθανών παραβιάσεων ασφαλείας [36].

3.2.6 Επεκτασιμότητα και Απόδοση της Neo4j

Η Neo4j έχει σχεδιαστεί με σκοπό να προσφέρει υψηλή επεκτασιμότητα και απόδοση, ακόμα και όταν τα δεδομένα αυξάνονται ραγδαία σε μέγεθος. Ο τρόπος με τον οποίο έχει δομηθεί η αρχιτεκτονική της επιτρέπει την προσθήκη περισσότερων κόμβων σε ένα cluster, βελτιώνοντας την απόδοση μέσω της κατανομής του φορτίου. Η επεκτασιμότητα της Neo4j επιτυγχάνεται μέσω της χρήσης κατανεμημένων κόμβων και της επανάληψης των δεδομένων. Αυτό σημαίνει ότι τα δεδομένα μπορούν να αποθηκεύονται σε πολλαπλούς κόμβους ενός cluster, επιτρέποντας την παράλληλη επεξεργασία τους. Με την χρήση αυτού του μηχανισμού εξασφαλίζεται επίσης η ανθεκτικότητα της βάσης δεδομένων, καθώς οι κόμβοι του cluster μπορούν να αναλάβουν το ρόλο άλλων κόμβων σε περίπτωση αποτυχίας [37]. Η απόδοση της Neo4j βελτιώνεται και εξαρτάται από τον τρόπο που αποθηκεύει τα δεδομένα της και την τεχνική αποθήκευσης της στο δίσκο, η οποία είναι βελτιστοποιημένη για την αποθήκευση και ανάκτηση γραφημάτων.



Σχήμα 3.2: Σύγκριση απόδοσης - performance benchmarks [38]

3.2.7 Διαχείριση Συναλλαγών και Συνέπεια Δεδομένων

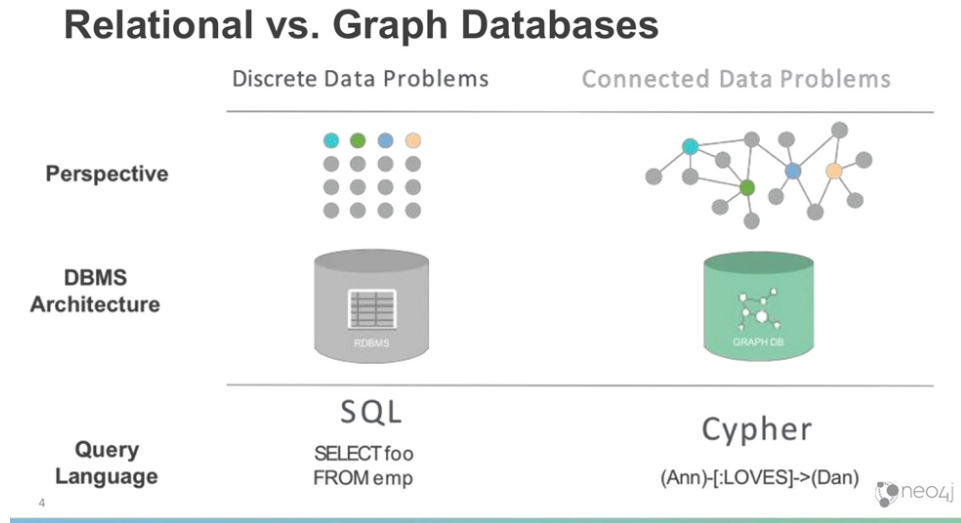
Η Neo4j υποστηρίζει ACID-συμβατές συναλλαγές. Καταφέρνει έτσι να εξασφαλίσει την ακεραιότητα και τη συνέπεια των δεδομένων. Κατ' επέκταση μόνο εάν όλες οι ενέργειες της συναλλαγής ολοκληρωθούν με επιτυχία επιβεβαιώνονται οι αλλαγές στα δεδομένα της βάσης δεδομένων. Σε περίπτωση αποτυχίας οποιουδήποτε μέρους της συναλλαγής, όλα τα δεδομένα επιστρέφονται στην αρχική τους κατάσταση, διασφαλίζοντας την ακεραιότητα [39]. Οι συναλλαγές στη Neo4j περιλαμβάνουν την προσθήκη και διαγραφή κόμβων και ακμών, την ενημέρωση των ιδιοτήτων και τη δημιουργία ή διαγραφή ευρετηρίων. Όλες αυτές οι ενέργειες καταγράφονται σε ένα λογιστικό βιβλίο, το οποίο χρησιμοποιείται για την αποκατάσταση της βάσης δεδομένων σε περίπτωση διακοπής. Μέσω της χρήσης κατακευκασμένων συστημάτων λήψης αποφάσεων εξασφαλίζεται, όπως το Raft, η συνέπεια των δεδομένων. Με τον τρόπο αυτό ελέγχεται αν όλοι οι κόμβοι ενός cluster συμφωνούν για την κατάσταση της βάσης δεδομένων [33].

3.2.8 Σύγκριση της Neo4j με Άλλες Βάσεις Δεδομένων

Σε υφιστάμενο κεφάλαιο, θα επικεντρωθούμε στη σύγκριση της Neo4j με άλλες βάσεις δεδομένων, τόσο σχεσιακές (SQL) όσο και μη σχεσιακές (NoSQL). Η σύγκριση αυτή βασίζεται κυρίως στη δομή, τον τρόπο αποθήκευσης δεδομένων, και απόδοσης, με ιδιαίτερη έμφαση στις ανάγκες σχεδιασμού και υλοποίησης μιας βιβλιομετρικής βάσης δεδομένων.

3.2.9 Σύγκριση με Σχεσιακές Βάσεις Δεδομένων (SQL)

Οι σχεσιακές βάσεις δεδομένων, όπως οι MySQL, PostgreSQL, και Oracle Database, χρησιμοποιούνται ευρέως για την αποθήκευση δομημένων δεδομένων σε μορφή πίνακα. Οι πίνακες αποτελούν τη βασική μονάδα δεδομένων στις SQL βάσεις, όπου τα δεδομένα αποθηκεύονται σε γραμμές και στήλες.



Σχήμα 3.3: Σύγκριση σχεσιακών με βάσεις γραφημάτων [40]

Επεξήγηση Δομής και Τρόπου Αποθήκευσης Δεδομένων σε Σχεσιακή Βάση Δεδομένων

Στις σχεσιακές βάσεις δεδομένων, η δομή των δεδομένων ορίζεται από σχήματα (schemas), τα οποία καθορίζουν τα πεδία, τους τύπους δεδομένων και τις σχέσεις μεταξύ των πινάκων. Οι σχέσεις μεταξύ των πινάκων εκφράζονται μέσω ξένων κλειδιών (foreign keys) και υλοποιούνται μέσω των λεγόμενων JOINS. Τα δεδομένα αποθηκεύονται σε γραμμές μέσα στους πίνακες, και κάθε γραμμή αντιπροσωπεύει μια εγγραφή [22]. Η αποθήκευση των δεδομένων σε μορφή πινάκων είναι αποτελεσματική σε πολλές περιπτώσεις, όπως η αποθήκευση συναλλαγών, με τον περιορισμό ότι τα δεδομένα δεν πρέπει να αποτελούνται από πολύπλοκες σχέσεις. Αυτό συμβαίνει γιατί οι JOIN πράξεις που απαιτούνται για την ανάκτηση σχετικών δεδομένων μπορούν να γίνουν πολύπλοκες και αργές καθώς αυξάνεται ο αριθμός των πινάκων και των σχέσεων.

Πλεονεκτήματα και Μειονεκτήματα της Neo4j έναντι των SQL Βάσεων Δεδομένων

1. Πλεονεκτήματα της Neo4j:

- **Φυσική Αναπαράσταση Σχέσεων:** Η Neo4j αποθηκεύει τις σχέσεις ως πρώτης τάξης οντότητες, επιτρέποντας άμεση πρόσβαση σε αυτές χωρίς την ανάγκη JOIN πράξεων, κάνοντας την εξαιρετικά αποδοτική για εφαρμογές όπου οι σχέσεις είναι περίπλοκες και πολυεπίπεδες.
- **Αποδοτική Εκτέλεση Ερωτημάτων:** Η ταχύτητα με την οποία η Neo4j μπορεί να εκτελεί ερωτήματα που περιλαμβάνουν πολλές σχέσεις είναι γρηγορότερη από μια SQL βάση, καθώς δεν χρειάζεται να ενώσει πολλούς πίνακες.
- **Κλιμακωτή Δομή:** Η Neo4j είναι πιο ευέλικτη από τις σχεσιακές βάσεις δεδομένων όταν αφορά δεδομένα που αυξάνονται σε μέγεθος και πολυπλοκότητα, καθώς δεν απαιτεί αναδιάταξη του σχήματος δεδομένων [22].

2. Μειονεκτήματα της Neo4j:

- Έλλειψη Ωριμότητας για Ορισμένες Χρήσεις: Παρόλο που η Neo4j είναι εξαιρετική για εφαρμογές γραφημάτων και πολύπλοκες σχέσεις μεταξύ δεδομένων, μπορεί να μην είναι η καλύτερη επιλογή για απλές εφαρμογές που απαιτούν κυρίως αυστηρή διαχείριση συναλλαγών ή για απλές αποθήκες δεδομένων.
- Κόστος Εκμάθησης: Η μετάβαση από μια SQL βάση σε μια βάση δεδομένων γραφημάτων όπως η Neo4j απαιτεί χρόνο για να εξοικειωθεί κάποιος με τον νέο τρόπο σκέψης και την αποθήκευση δεδομένων [22].

3.2.10 Σύγκριση με Μη Σχεσιακές Βάσεις Δεδομένων (NoSQL)

Η Neo4j ανήκει στην κατηγορία των NoSQL βάσεων δεδομένων, οι οποίες περιλαμβάνουν επίσης άλλες τύπους βάσεων δεδομένων όπως οι Key-Value Stores (π.χ. Redis, DynamoDB), Document Stores (π.χ. MongoDB, Couchbase), και Column-Family Stores (π.χ. Cassandra, HBase). Κάθε μία από αυτές τις κατηγορίες εξυπηρετεί διαφορετικούς τύπους εφαρμογών και απαιτήσεων.

Key-Value Stores

Οι Key-Value Stores βάσεις δεδομένων, όπως οι Redis και DynamoDB, αποθηκεύουν δεδομένα σε μορφή ζευγών κλειδιού-αξίας (key-value). Η δομή αυτή αποδίδει κυρίως σε εφαρμογές που απαιτούν γρήγορη ανάκτηση δεδομένων με βάση ένα μοναδικό κλειδί.

Σύγκριση με Neo4j:

- Απλότητα και Απόδοση: Οι Key-Value Stores παρόλο που είναι πολύ αποδοτικοί για εφαρμογές που απαιτούν ταχύτατη πρόσβαση σε δεδομένα μέσω μοναδικών κλειδιών, δεν προσφέρουν καμία υποστήριξη για σύνθετες σχέσεις μεταξύ των δεδομένων. Από την άλλη πλευρά, η Neo4j εξυπηρετεί καλύτερα τις ανάγκες σύνθετων γραφημάτων.
- Χρήση Περίπλοκων Σχέσεων: Οι Key-Value Stores δεν είναι ιδανικοί για εφαρμογές με περίπλοκες σχέσεις. Αντιθέτως, η Neo4j μπορεί να διαχειριστεί τέτοιες ανάγκες με φυσικό τρόπο, αποθηκεύοντας τις σχέσεις ως μέρος του γράφου[39].

Document Stores

Οι Document Stores, όπως οι MongoDB και Couchbase, αποθηκεύουν δεδομένα σε μορφή εγγράφων (documents), τα οποία συνήθως είναι σε μορφή JSON ή BSON. Η προσέγγιση αυτή τις κάνουν ιδιαίτερα ευέλικτες και επιτρέπει την αποθήκευση πολύπλοκων και ημιδομημένων δεδομένων.

Σύγκριση με Neo4j:

- Ευελιξία: Οι Document Stores, όπως προανέφερα είναι πολύ ευέλικτες και επιτρέπουν την αποθήκευση ημιδομημένων δεδομένων με διάφορα πεδία και ένθετες δομές. Ωστόσο, η Neo4j προσφέρει μια φυσική αναπαράσταση των σχέσεων μεταξύ των δεδομένων, κάτι που μπορεί να είναι πιο αποδοτικό σε εφαρμογές γραφημάτων.

- **Επεξεργασία Σχέσεων:** Ενώ οι Document Stores μπορούν να αποθηκεύσουν δεδομένα που σχετίζονται μεταξύ τους, η διαχείριση των σχέσεων μπορεί να γίνει περίπλοκη και αναποτελεσματική. Η Neo4j διαχειρίζεται τις σχέσεις με τρόπο που επιτρέπει την εύκολη και αποδοτική ανάκτηση [39].

Column-Family Stores

Οι Column-Family Stores, παραδειγματικά οι Cassandra και HBase, είναι σχεδιασμένοι για την αποθήκευση τεράστιων ποσοτήτων δεδομένων καταναμημένα σε πολλαπλούς κόμβους. Χρησιμοποιούνται συνήθως σε εφαρμογές που απαιτούν υψηλή απόδοση και επεκτασιμότητα. Σύγκριση με Neo4j:

- **Επεκτασιμότητα:** Οι Column-Family Stores είναι σχεδιασμένες για επεκτασιμότητα και μπορούν να διαχειριστούν τεράστιες ποσότητες δεδομένων. Η Neo4j, προσφέρει επεκτασιμότητα, αλλά είναι πιο εξειδικευμένη στη διαχείριση γραφημάτων και μπορεί να μην είναι το ίδιο αποδοτική σε εφαρμογές όπου η επεκτασιμότητα είναι η πρωταρχική ανάγκη.
- **Διαχείριση Σχέσεων:** Οι Column-Family Stores είναι πιο κατάλληλες για εφαρμογές όπου οι σχέσεις μεταξύ των δεδομένων δεν είναι πρωταρχικής σημασίας, αντίθετα η Neo4j εξυπηρετεί καλύτερα τις ανάγκες για διαχείριση πολύπλοκων σχέσεων καθώς αποτελεί και έναν από τους βασικούς σκοπούς της [39].

3.3 Εφαρμογή της Neo4j στον Σχεδιασμό και την Υλοποίηση Βιβλιομετρικής Βάσης Δεδομένων

Σε αυτό κεφάλαιο θα εξετάσουμε την εφαρμογή της Neo4j συγκεκριμένα στη σχεδίαση και την υλοποίηση μιας βιβλιομετρικής βάσης δεδομένων. Θα εστιάσουμε στα ιδιαίτερα χαρακτηριστικά που καθιστούν τη Neo4j ιδανική για αυτήν την κατηγορία εφαρμογών.

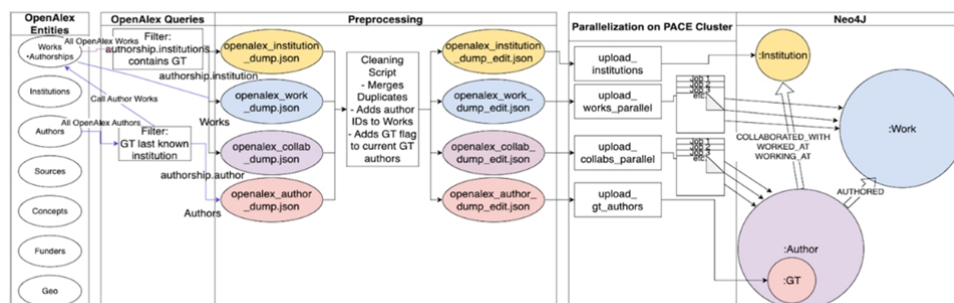
3.3.1 Αντιστοίχιση Βιβλιομετρικών Δεδομένων στη Δομή Γραφημάτων

Τα βιβλιομετρικά δεδομένα χαρακτηρίζονται από περίπλοκες σχέσεις μεταξύ διαφορετικών οντοτήτων, όπως δημοσιεύσεις, συγγραφείς, παραπομπές, και ιδρύματα. Η Neo4j προσφέρει μια αποδοτική δομή για την αποθήκευση αυτών των δεδομένων, όπου οι δημοσιεύσεις μπορούν να αναπαρίστανται ως κόμβοι, οι συγγραφείς ως άλλοι κόμβοι, ενώ οι παραπομπές ως σχέσεις μεταξύ των κόμβων. Για παράδειγμα, ένα γράφημα στη Neo4j μπορεί να περιλαμβάνει κόμβους που αναπαριστούν:

- **Δημοσιεύσεις:** Με ιδιότητες όπως ο τίτλος, το έτος έκδοσης, και το DOI.
- **Συγγραφείς:** Με ιδιότητες όπως το όνομα, η ειδικότητα, και το ίδρυμα.
- **Ιδρύματα:** Με ιδιότητες όπως το όνομα, η τοποθεσία, και οι συνεργασίες.

- Παραπομπές: Ως σχέσεις μεταξύ δημοσιεύσεων, που δείχνουν ποια δημοσίευση αναφέρεται σε ποια άλλη.

Αυτή η δομή έχει ως αποτέλεσμα τη γρήγορη και αποδοτική ανάλυση των σχέσεων μεταξύ των οντοτήτων. Ορισμένα παραδείγματα θα μπορούσαν να είναι η ανίχνευση των πιο αναφερόμενων δημοσιεύσεων, η ανάλυση συνεργασιών μεταξύ συγγραφέων, και η αξιολόγηση της επιρροής των ιδρυμάτων [31].



Σχήμα 3.4: Παράδειγμα ροής βιβλιομετρικών δεδομένων από την OpenAlex στην Neo4j όπως πραγματοποιήθηκε σε παρόμοια εργασία [41]

3.3.2 Ανάλυση και Οπτικοποίηση Δεδομένων

Η Neo4j μπορεί να υποστηρίξει ερωτήματα που εξετάζουν τις σχέσεις μεταξύ των δεδομένων, και τα αποτελέσματα αυτών των ερωτημάτων μπορούν κατ' επέκταση εύκολα να οπτικοποιηθούν ως γράφημα. Για παράδειγμα:

- Ανάλυση Απήχησης: Ανάλυση των κόμβων (δημοσιεύσεις ή συγγραφείς) οι οποίοι έχουν περισσότερη απήχηση σε ένα συγκεκριμένο σύνολο δεδομένων, χρησιμοποιώντας μονάδες μέτρησης όπως η βαθμολογία PageRank ή η κεντρικότητα βαθμού (degree centrality).
- Οπτικοποίηση Συνεργασιών: Οι σχέσεις συνεργασίας μεταξύ συγγραφέων ή ιδρυμάτων μπορούν να οπτικοποιηθούν ως γράφημα, αποκαλύπτοντας κρυφές συνεργασίες ή συνεργασίας σε ένα ερευνητικό πεδίο.
- Εξερεύνηση Κοινότητας: Η Neo4j επιτρέπει την εύκολη αναγνώριση κοινοτήτων μέσα στο γράφημα, τονίζοντας ομάδες συγγραφέων ή δημοσιεύσεων που είναι στενά συνδεδεμένες.

3.3.3 Συμπέρασμα Χρήσης της Neo4j για Βιβλιομετρικές Βάσεις Δεδομένων

Η Neo4j παρέχει ένα σύνολο χαρακτηριστικών και δομών που την καθιστούν ιδανική για τη διαχείριση και την ανάλυση βιβλιομετρικών δεδομένων. Δυνατότητες όπως η φυσική αναπαράσταση των σχέσεων ως γράφημα, η ικανότητα εκτέλεσης σύνθετων ερωτημάτων και αναλύσεων, και η υποστήριξη για επεκτασιμότητα και ασφάλεια την καθιστούν την καλύτερη επιλογή για την ανάπτυξη μιας βιβλιομετρικής βάσης δεδομένων μεγάλης κλίμακας. Επιτρέπει

στους βιβλιομετρικούς αναλυτές να αποκτήσουν βαθύτερη κατανόηση των δεδομένων τους και να εντοπίσουν τάσεις που μπορείς να μην είναι εμφανείς με άλλες μεθόδους [39].

3.4 Πλεονεκτήματα Χρήσης της Neo4j για τον Σχεδιασμό και την Υλοποίηση Βιβλιομετρικής Βάσης Μεγάλων Δεδομένων

Το τελευταίο κεφάλαιο παραθέτω τα πλεονεκτήματα της Neo4j ειδικά για την υλοποίηση μιας βιβλιομετρικής βάσης δεδομένων, λαμβάνοντας υπόψη τις ιδιαίτερες ανάγκες αυτής της εφαρμογής.

3.4.1 Δομή Δεδομένων που Αντιστοιχεί Φυσικά στα Βιβλιομετρικά Δεδομένα

Η βιβλιομετρική ανάλυση, όπως προανέφερα, απαιτεί τη διαχείριση σύνθετων σχέσεων μεταξύ διαφορετικών οντοτήτων, όπως δημοσιεύσεις, συγγραφείς, ιδρύματα και αναφορές. Η Neo4j προσφέρει μια φυσική αναπαράσταση αυτών των σχέσεων μέσω της δομής γραφημάτων της, όπου κάθε οντότητα μπορεί να αναπαρασταθεί ως κόμβος και κάθε σχέση ως ακμή. Για παράδειγμα, μια δημοσίευση μπορεί να συνδέεται με τους συγγραφείς της, τα ιδρύματα τους, και τις παραπομπές που περιέχει. Αυτές οι σχέσεις μπορούν να αναπαρασταθούν εύκολα και αποτελεσματικά σε ένα γράφημα, επιτρέποντας την γρήγορη εκτέλεση σύνθετων ερωτημάτων [42].

3.4.2 Επεξεργασία και Ανάλυση Μεγάλων Συνόλων Δεδομένων σε Πραγματικό Χρόνο

Η Neo4j είναι σχεδιασμένη με σκοπό τη διαχείριση και ανάλυση μεγάλων γραφημάτων, επιτρέποντας την επεξεργασία μεγάλων συνόλων δεδομένων σε πραγματικό χρόνο. Δύναται να αποθηκεύει και να ανακτά δεδομένα με βάση τις σχέσεις τους το οποίο την καθιστά ιδανική για βιβλιομετρικές βάσεις δεδομένων, όπου οι σχέσεις μεταξύ των δημοσιεύσεων και των αναφορών έχουν κύρια σημασία. Η απόδοση της Neo4j σε τέτοιες εφαρμογές είναι συνήθως ανώτερη από αυτή των παραδοσιακών SQL βάσεων δεδομένων, όπου τα ερωτήματα που περιλαμβάνουν πολλές σχέσεις μπορεί να γίνουν πολύ αργά και απαιτητικά σε πόρους [37].

3.4.3 Ικανότητα Διαχείρισης Πολυπλοκότητας και Πολλαπλών Σχέσεων

Η Neo4j ξεχωρίζει για την ικανότητά της να διαχειρίζεται τη πολυπλοκότητα μεταξύ των πολλαπλών σχέσεων. Ειδικότερα σε μια βιβλιομετρική βάση δεδομένων, μπορεί να υπάρχουν πολλαπλές σχέσεις μεταξύ των δεδομένων, όπως οι παραπομπές μεταξύ δημοσιεύσεων, οι συνεργασίες μεταξύ συγγραφέων, και οι σχέσεις μεταξύ ιδρυμάτων. Η Neo4j έχει σχεδιαστεί έτσι ώστε να αποθηκεύσει και να διαχειριστεί αυτές τις σχέσεις με φυσικό τρόπο, επιτρέποντας την εύκολη εκτέλεση ερωτημάτων που αφορούν την πολυπλοκότητα αυτή. Για παράδειγμα, μπορεί κάποιος να εκτελέσει ερωτήματα που αναζητούν όλους τους συγγραφείς που έχουν

συνεργαστεί με έναν συγκεκριμένο συγγραφέα σε μια σειρά δημοσιεύσεων, κάτι που σε μια σχεσιακή βάση δεδομένων θα απαιτούσε πολλαπλές σύνθετες πράξεις JOIN.

3.4.4 Υποστήριξη Σύνθετων Ερωτημάτων και Αναλύσεων

Η Neo4j διαθέτει ένα ισχυρό μηχανισμό εκτέλεσης ερωτημάτων όπου επιτρέπει την ταχεία εκτέλεση σύνθετων αναλύσεων. Οι χρήστες μπορούν να διατυπώνουν ερωτήματα με σκοπό να εξερευνήσουν τις σχέσεις μεταξύ των δεδομένων, να ανακαλύψουν μοτίβα και να διεξάγουν διάφορες αναλύσεις [42]. Η δυνατότητα της Neo4j να εκτελεί αυτά τα ερωτήματα γρήγορα και αποδοτικά την καθιστά ιδανική για την εφαρμογή αυτή, διότι σε μια βιβλιομετρική βάση δεδομένων, τέτοιες αναλύσεις είναι κρίσιμες για την κατανόηση της δομής και της επιρροής της επιστημονικής κοινότητας.

3.4.5 Επεκτασιμότητα και Προσαρμοστικότητα της Βάσης Δεδομένων για Μελλοντικές Ανάγκες

Τέλος, η Neo4j επιτρέπει την εύκολη προσαρμογή της βάσης δεδομένων καθώς τα δεδομένα αυξάνονται σε μέγεθος και πολυπλοκότητα. Η δυνατότητα της να επεκτείνεται τόσο κατακόρυφα όσο και οριζόντια εξασφαλίζει ότι η βιβλιομετρική βάση δεδομένων μπορεί να μεγαλώσει και να προσαρμοστεί στις μελλοντικές ανάγκες χωρίς να απαιτείται σημαντική αναδιάταξη των δεδομένων ή της αρχιτεκτονικής. Αυτό το χαρακτηριστικό είναι ιδιαίτερα σημαντικό σε περιβάλλοντα όπου η ποσότητα και η πολυπλοκότητα των δεδομένων αυξάνονται συνεχώς, όπως συμβαίνει συχνά στις βιβλιομετρικές βάσεις δεδομένων.

Κεφάλαιο 4

Εισαγωγή στην Python και στην γλώσσα ερωταποκρίσεων Cypher

4.1 Εισαγωγή στην Python

4.1.1 Ιστορία και εξέλιξη της Python

Η Python είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού, έχει αναπτυχθεί από τον Guido van Rossum το 1991. Παρόλο που η αρχική της έκδοση σχεδιάστηκε με στόχο να είναι εύχρηστη και να έχει ευανάγνωστη σύνταξη, καθιστώντας την προσβάσιμη τόσο σε αρχάριους όσο και σε έμπειρους προγραμματιστές κατάφερε μέσα από πολλές βελτιστοποιήσεις να εξελιχθεί σημαντικά και με πολλές νέες εκδόσεις που έχουν εισαγάγει βελτιώσεις και νέες δυνατότητες να προσαρμοστεί στις σύγχρονες ανάγκες της προγραμματιστικής κοινότητας και των σύγχρονων αναγκών [43]. Η Python 2.0, που κυκλοφόρησε το 2000, ήταν μια σημαντική αναβάθμιση που εισήγαγε τη συλλογή απορριμμάτων (garbage collection) και την υποστήριξη του Unicode, γεγονός που επέτρεψε τη χρήση της γλώσσας σε διεθνές επίπεδο. Ωστόσο, η πιο σημαντική αλλαγή ήρθε με την Python 3.0 το 2008, η οποία έφερε δραστικές αλλαγές στη γλώσσα, βελτιώνοντας την απόδοση και τη συμβατότητα με νέες τεχνολογίες [44]. Αυτές οι αλλαγές δεν ήταν πλήρως συμβατές με τις προηγούμενες εκδόσεις, γεγονός που οδήγησε σε παράλληλη χρήση των Python 2 και Python 3 για αρκετά χρόνια. Η Python λόγω της συνεχόμενης εξέλιξης της έχει γίνει η βασική γλώσσα επιλογής για πολλούς χρήστες καθώς και για την υλοποίηση πολλών εφαρμογών. Η Python συγκριμένα είναι ευρέως διαδεδομένη και στους κλάδους της ανάλυσης δεδομένων, της μηχανικής μάθησης και της επιστημονικής έρευνας. Η ευρεία υιοθέτησή της οφείλεται στη συνεχή ανάπτυξη της κοινότητας και στην εκτεταμένη βιβλιοθήκη πακέτων που διατίθενται μέσω του Python Package Index (PyPI), που επιτρέπει στους χρήστες να επεκτείνουν τη λειτουργικότητα της γλώσσας για να καλύψουν εξειδικευμένες ανάγκες καθώς και να διατηρούνται up-to-date με τις σύγχρονες ανάγκες μέσω των νέων επεκτάσεων και βιβλιοθηκών που γίνονται διαθέσιμες [45] [46].

4.1.2 Βασικά χαρακτηριστικά της Python που την καθιστούν κατάλληλη για ανάλυση δεδομένων

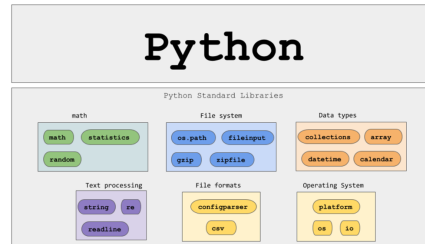
Όπως ανέφερα και παραπάνω η Python έχει καθιερωθεί ως ένα από τα βασικά εργαλεία για την ανάλυση δεδομένων και την επιστημονική υπολογιστική λόγω πολλών χαρακτηριστικών της που την καθιστούν την πλέον βέλτιστη επιλογή για αυτές τις χρήσεις. Αρχικά από τα κυριότερα χαρακτηριστικά της είναι η απλότητα και η ευανάγνωστη σύνταξή της, που επιτρέπουν την ταχύτερη ανάπτυξη κώδικα και την εύκολη συντήρηση του, εξού και ο λόγος που η Python χαρακτηρίζεται ως μια “εύκολη” γλώσσα προγραμματισμού ως προς την εκμάθησή της [47]. Επιπλέον χαρακτηριστικό της είναι η πληθώρα βιβλιοθηκών καθώς και δεκάδων χιλιάδων πακέτων, τα περισσότερα εκ των οποίων διατίθενται δωρεάν προς χρήση και ανοικτού κώδικα. Χαρακτηριστικά παραδείγματα είναι οι NumPy, Pandas, και SciPy, που είναι απαραίτητα για την επεξεργασία και ανάλυση μεγάλων όγκων δεδομένων [48]. Οι δυνατότητες των βιβλιοθηκών αυτών είναι ότι διαθέτουν εργαλεία για σύνθετους μαθηματικούς υπολογισμούς, στατιστικές αναλύσεις και διαχείριση δεδομένων, επιτρέποντας στους χρήστες να εκτελούν πολύπλοκες εργασίες με αποδοτικότητα και ακρίβεια. Επιπλέον, η Python υποστηρίζει παράλληλη και καταναεμημένη επεξεργασία μέσω βιβλιοθηκών όπως το Dask και το PySpark, κάτι που την καθιστά ιδανική για εφαρμογές σε μεγάλες κλίμακες όπως και η δική μου [49]. Ένα άλλο σημαντικό χαρακτηριστικό της Python είναι η διαλειτουργικότητά της με άλλες γλώσσες και πλατφόρμες. Η δυνατότητα της αυτή της επιτρέπει την ομαλή ενσωμάτωσή της σε πολύπλοκα projects όπως και την αξιοποίηση διαφορετικών τεχνολογιών με ευκολία [50]. Η ευελιξία της γλώσσας την καθιστά ιδανική για τη δημιουργία πρωτοτύπων καθώς και για την ανάπτυξη τελικών προϊόντων σε διάφορους τομείς της επιστήμης και της τεχνολογίας [51][52]. Η ανάλυση δεδομένων και η επιστημονική υπολογιστική είναι δυο κλάδοι με συνεχώς αυξανόμενες απαιτήσεις, επομένως η συνεχώς αναπτυσσόμενη Python παραμένει ένα ισχυρό εργαλείο ως προς την επίλυση σύνθετων προβλημάτων και την υποστήριξη της καινοτομίας.



Σχήμα 4.1: Θετικά και αρνητικά χαρακτηριστικά της Python για ανάλυση δεδομένων[53]

4.1.3 Επισκόπηση των Βασικών Βιβλιοθηκών της Python

Οι βιβλιοθήκες της Python αποτελούν βασικό και αναπόσπαστο κομμάτι της γλώσσας και παρέχουν ένα ευρύ φάσμα εργαλείων για την ανάπτυξη εφαρμογών και την επεξεργασία δεδομένων. Στο πλαίσιο αυτού του project, χρησιμοποίησα αρκετές από αυτές τις βιβλιοθήκες για να επιτευχθεί η απαιτούμενη λειτουργικότητα. Οι κύριες βιβλιοθήκες που χρησιμοποιήθηκαν περιλαμβάνουν τις `os`, `subprocess`, `shutil`, `gzip`, `json`, και `csv`.



Σχήμα 4.2: Οι βασικές βιβλιοθήκες της Python [54]

Η βιβλιοθήκη `gzip` και η χρησιμότητά της στην εφαρμογή

Η βιβλιοθήκη `gzip` της Python παρέχει μια απλή και αποδοτική μέθοδο για τη συμπίεση και αποσυμπίεση αρχείων χρησιμοποιώντας τη μορφή `gzip`. Στην πτυχιακή μου εργασία όπου βασικό κομμάτι της υλοποίησης ήταν η αποσυμπίεση και η συμπίεση των `json` αρχείων που επισύναψα από την `OpenAlex` ήταν απαραίτητη η χρήση της στο script. Αυτή η βιβλιοθήκη είναι ενσωματωμένη στη Python, γεγονός που την καθιστά άμεσα διαθέσιμη χωρίς την ανάγκη εγκατάστασης επιπρόσθετων πακέτων [55]. Ο σκοπός της βιβλιοθήκης είναι να διευκολύνει τη διαχείριση μεγάλων όγκων δεδομένων, όπως στην περίπτωση της υφιστάμενης εφαρμογής τα δεδομένα της `OpenAlex`, μειώνοντας το μέγεθος των αρχείων και βελτιώνοντας την ταχύτητα μεταφοράς και αποθήκευσής τους. Η μορφή `gzip` χρησιμοποιείται ευρέως για τη συμπίεση αρχείων, ειδικά σε περιβάλλοντα όπου η αποδοτικότητα του χώρου αποθήκευσης και η ταχύτητα μεταφοράς είναι κρίσιμες. Η βιβλιοθήκη `gzip` της Python έχει το προνόμιο ότι επιτρέπει στους προγραμματιστές να εργάζονται με αρχεία συμπιεσμένα σε αυτή τη μορφή με τον ίδιο τρόπο που θα εργάζονταν με απλά αρχεία, καθιστώντας τη χρήση της πολύ απλή και ευέλικτη και εξοικονομώντας χρόνο στον χρήστη [56]. Η βιβλιοθήκη `gzip` περιλαμβάνει αρκετές βασικές συναρτήσεις που διευκολύνουν τη διαχείριση συμπιεσμένων αρχείων. Οι πιο σημαντικές από αυτές τις συναρτήσεις είναι οι `gzip.open()`, `gzip.compress()`, και `gzip.decompress()` [56]. Η συνάρτηση `gzip.open()` χρησιμοποιείται για το άνοιγμα συμπιεσμένων αρχείων `gzip` είτε σε δυαδική είτε σε μορφή κειμένου. Αυτή η συνάρτηση επιτρέπει τη δημιουργία ενός αρχείου αντικειμένου που μπορεί να χρησιμοποιηθεί για την ανάγνωση ή την εγγραφή δεδομένων σε ένα συμπιεσμένο αρχείο. Το `gzip.open()` μπορεί να χρησιμοποιηθεί με διάφορες λειτουργίες, όπως ανάγνωση ('r'), εγγραφή ('w'), και προσθήκη ('a'), επιτρέποντας έτσι την ευέλικτη διαχείριση αρχείων [55]. Η συνάρτηση `gzip.compress()` παρέχει τη δυνατότητα συμπίεσης δεδομένων σε μορφή `gzip`. Αυτή η συνάρτηση δέχεται ως είσοδο ένα αντικείμενο τύπου `bytes` και επιστρέφει τα δεδομένα σε συμπιεσμένη μορφή. Είναι ιδιαίτερα χρήσιμη σε περιπτώσεις όπου απαιτείται η συμπίεση δεδομένων πριν από τη μεταφορά τους ή την αποθήκευσή τους, για εξοικονόμηση χώρου και χρόνου [57]. Η συνάρτηση `gzip.decompress()` χρησιμοποιείται για την αποσυμπίεση

δεδομένων που έχουν συμπίεστεί με τη μορφή gzip. Αυτή η συνάρτηση δέχεται ως είσοδο τα συμπίεσμένα δεδομένα και επιστρέφει τα αποσυμπίεσμένα δεδομένα σε μορφή bytes. Η χρήση της είναι κρίσιμη σε εφαρμογές όπου τα δεδομένα αποθηκεύονται ή μεταφέρονται σε συμπίεσμένη μορφή και απαιτείται η ανάκτησή τους σε πρωτογενή μορφή για περαιτέρω επεξεργασία [56]. Η βιβλιοθήκη gzip παρέχει τις απαραίτητες λειτουργίες για τη διαχείριση συμπίεσμένων δεδομένων, καθιστώντας την ιδανική για εφαρμογές που ασχολούνται με μεγάλα δεδομένα και απαιτούν βελτιστοποίηση του χώρου αποθήκευσης και της απόδοσης [55][57].

Οι βιβλιοθήκες os και shutil

- Βιβλιοθήκη os

Η βιβλιοθήκη os της Python παρέχει ένα πλήρες σύνολο εργαλείων που αφορά την αλληλεπίδραση με το λειτουργικό σύστημα. Επιτρέπει στους προγραμματιστές να εκτελούν λειτουργίες διαχείρισης αρχείων και καταλόγων, καθώς και άλλες λειτουργίες χαμηλού επιπέδου που σχετίζονται με το σύστημα αρχείων. Με την συγκεκριμένη βιβλιοθήκη, οι προγραμματιστές μπορούν να χειρίζονται αρχεία, να ανακτούν πληροφορίες για το σύστημα και να εκτελούν εντολές του συστήματος μέσω του προγράμματος τους [58]. Ένα από τα βασικά χαρακτηριστικά της αποτελεί η διαλειτουργικότητα της με διάφορα λειτουργικά συστήματα, επιτρέποντας τη δημιουργία φορητών scripts που μπορούν να εκτελούνται σε διαφορετικά περιβάλλοντα χωρίς αλλαγές στον κώδικα. Δίνει τη δυνατότητα διαχείρισης καταλόγων με συναρτήσεις όπως `os.mkdir()` για τη δημιουργία νέων καταλόγων και `os.rmdir()` για τη διαγραφή τους. Επιπλέον, παρέχει εργαλεία για την ανάκτηση πληροφοριών σχετικά με τα αρχεία και τους καταλόγους, όπως η συνάρτηση `os.path.getsize()` για τον προσδιορισμό του μεγέθους ενός αρχείου [59]. Θεωρείται πως είναι κρίσιμη για τη δημιουργία προγραμμάτων που απαιτούν αλληλεπίδραση με το σύστημα αρχείων. Ακόμη, επιτρέπει στους προγραμματιστές να αυτοματοποιούν πολλές από τις διαδικασίες που διαφορετικά θα απαιτούσαν χειροκίνητη παρέμβαση. Η ευελιξία της την καθιστά απαραίτητο εργαλείο για την ανάπτυξη εφαρμογών που χρειάζονται πρόσβαση στο σύστημα αρχείων του λειτουργικού συστήματος [60].

- Βιβλιοθήκη shutil

Προσφέρει συναρτήσεις όπως τη `shutil.copy()`, η οποία επιτρέπει την αντιγραφή αρχείων από μία τοποθεσία σε άλλη, και τη `shutil.move()`, η οποία επιτρέπει τη μετακίνηση αρχείων και καταλόγων. Επιπρόσθετα, η `shutil` υποστηρίζει λειτουργίες όπως η αναπαγωγή δομών καταλόγων, διευκολύνοντας τη διαχείριση μεγάλων συστημάτων αρχείων [61].

Η διευκόλυνση και η αποδοτικότητα που προσφέρει την καθιστούν αναντικατάστατο εργαλείο σε προγράμματα που απαιτούν εκτεταμένη διαχείριση αρχείων. Συγκεκριμένα σε εφαρμογές σαν την υπάρχουσα όπου η απόδοση και η αποδοτικότητα είναι κρίσιμες, η `shutil` εξασφαλίζει ότι οι λειτουργίες αρχείων εκτελούνται με τον βέλτιστο δυνατό τρόπο [58]. Η βιβλιοθήκη `shutil` είναι μια βιβλιοθήκη υψηλού επιπέδου της Python, η οποία επικεντρώνεται στη διαχείριση και τη μετακίνηση αρχείων και καταλόγων. Παρόλο που η `os` παρέχει βασικές λειτουργίες για τη διαχείριση αρχείων, η `shutil` επεκτείνει αυτές τις δυνατότητες, δημιουργώντας πιο σύνθετες και αποδοτικές λειτουργίες όπως η αντιγραφή και η μετακίνηση ολόκληρων δέντρων καταλόγων του συστήματος διαχείρισης αρχείων

[62]. Προσφέρει συναρτήσεις όπως τη `shutil.copy()`, η οποία επιτρέπει την αντιγραφή αρχείων από μία τοποθεσία σε άλλη, και τη `shutil.move()`, η οποία επιτρέπει τη μετακίνηση αρχείων και καταλόγων. Επιπρόσθετα, η `shutil` υποστηρίζει λειτουργίες όπως η αναπαραγωγή δομών καταλόγων, διευκολύνοντας τη διαχείριση μεγάλων συστημάτων αρχείων [61]. Η διευκόλυνση και η αποδοτικότητα που προσφέρει την καθιστούν αναγκατάστατο εργαλείο σε προγράμματα που απαιτούν εκτεταμένη διαχείριση αρχείων. Συγκεκριμένα σε εφαρμογές σαν την υπάρχουσα όπου η απόδοση και η αποδοτικότητα είναι κρίσιμες, η `shutil` εξασφαλίζει ότι οι λειτουργίες αρχείων εκτελούνται με τον βέλτιστο δυνατό τρόπο [58].

- Κύριες Συναρτήσεις που χρησιμοποιήθηκαν

Η συνάρτηση `os.walk()` είναι ένα πανίσχυρο εργαλείο που επιτρέπει τη διάσχιση (traversal) ενός δέντρου καταλόγων, επιστρέφοντας μια γεννήτρια που περιλαμβάνει πληροφορίες για τους καταλόγους και τα αρχεία που περιέχονται σε αυτούς. Αυτό είναι ιδιαίτερα χρήσιμο για προγράμματα που πρέπει να εξετάσουν ή να επεξεργαστούν κάθε αρχείο σε μια συγκεκριμένη διαδρομή [58].

Η συνάρτηση `os.remove()` χρησιμοποιείται για τη διαγραφή αρχείων από το σύστημα αρχείων. Αυτή η συνάρτηση είναι απλή στη χρήση, αλλά ισχυρή, καθώς επιτρέπει την εύκολη διαγραφή αρχείων από το πρόγραμμα χωρίς την ανάγκη παρέμβασης του χρήστη [58]. Η `shutil.copy()` είναι μια συνάρτηση υψηλού επιπέδου που επιτρέπει την αντιγραφή αρχείων από μία τοποθεσία σε άλλη. Αυτή η συνάρτηση διατηρεί τα μεταδεδομένα του αρχείου κατά την αντιγραφή, διασφαλίζοντας ότι το αρχείο στοχεύετε με ακρίβεια στη νέα του θέση [61]. Οι βιβλιοθήκες `os` και `shutil` συνεργάζονται αρμονικά, προσφέροντας όλα τα απαραίτητα εργαλεία για τη διαχείριση αρχείων και καταλόγων σε οποιαδήποτε εφαρμογή Python.]Ορισμένες από τις σημαντικότερες συναρτήσεις που παρέχουν οι βιβλιοθήκες `os` και `shutil` είναι οι `os.walk()`, `os.remove()`, και `shutil.copy()`. Η συνάρτηση `os.walk()` είναι ένα πανίσχυρο εργαλείο που επιτρέπει τη διάσχιση (traversal) ενός δέντρου καταλόγων, επιστρέφοντας μια γεννήτρια που περιλαμβάνει πληροφορίες για τους καταλόγους και τα αρχεία που περιέχονται σε αυτούς. Αυτό είναι ιδιαίτερα χρήσιμο για προγράμματα που πρέπει να εξετάσουν ή να επεξεργαστούν κάθε αρχείο σε μια συγκεκριμένη διαδρομή [58]. Η συνάρτηση `os.remove()` χρησιμοποιείται για τη διαγραφή αρχείων από το σύστημα αρχείων. Αυτή η συνάρτηση είναι απλή στη χρήση, αλλά ισχυρή, καθώς επιτρέπει την εύκολη διαγραφή αρχείων από το πρόγραμμα χωρίς την ανάγκη παρέμβασης του χρήστη [58].

Η `shutil.copy()` είναι μια συνάρτηση υψηλού επιπέδου που επιτρέπει την αντιγραφή αρχείων από μία τοποθεσία σε άλλη. Αυτή η συνάρτηση διατηρεί τα μεταδεδομένα του αρχείου κατά την αντιγραφή, διασφαλίζοντας ότι το αρχείο στοχεύετε με ακρίβεια στη νέα του θέση [61]. Οι βιβλιοθήκες `os` και `shutil` συνεργάζονται αρμονικά, προσφέροντας όλα τα απαραίτητα εργαλεία για τη διαχείριση αρχείων και καταλόγων σε οποιαδήποτε εφαρμογή Python.

Η βιβλιοθήκη `subprocess`

Η βιβλιοθήκη `subprocess` της Python είναι ένα ισχυρό εργαλείο για τη δημιουργία και διαχείριση νέων διεργασιών, αντικαθιστώντας παλαιότερες βιβλιοθήκες όπως οι `os.system`, `os.spawn*` και

os.popen*, καθώς προσφέρει μεγαλύτερη ασφάλεια και ευελιξία. Κύριος σκοπός της είναι η εκτέλεση εξωτερικών προγραμμάτων και εντολών συστήματος, επιτρέποντας την αλληλεπίδραση με αυτές μέσω της Python. Κύρια συνάρτηση αποτελεί η subprocess.run(), με την οποία οι προγραμματιστές μπορούν να εκτελεστούν εντολές στο λειτουργικό σύστημα και να συλλέγουν τα αποτελέσματα [63]. Επιπλέον, παρέχει τη δυνατότητα επικοινωνίας με τις διεργασίες, επιτρέποντας τη δρομολόγηση της εισόδου και εξόδου τους με το σύστημα ή την Python. Ακόμη, προσφέρει αυξημένη ασφάλεια σε σχέση με προηγούμενες μεθόδους, αποφεύγοντας επικίνδυνες πρακτικές, ενώ ταυτόχρονα χαρακτηρίζεται από μεγάλη ευελιξία στη διαχείριση της ροής δεδομένων και την προσαρμογή των αποτελεσμάτων. Συμπερασματικά, η subprocess αποτελεί απαραίτητο εργαλείο για την αυτοματοποίηση και τη διασύνδεση εφαρμογών Python με εξωτερικά προγράμματα, βελτιώνοντας τη συνολική λειτουργικότητα και απόδοση των προγραμμάτων.

Βιβλιοθήκες json και csv

- Χρήση της json για ανάγνωση και εγγραφή δεδομένων JSON

Η συνάρτηση json.load() χρησιμοποιείται για την ανάγνωση JSON δεδομένων από ένα αρχείο και την αποτύπωσή τους σε Python αντικείμενα, όπως λίστες ή λεξικά. Αυτό επιτρέπει την εύκολη επεξεργασία των δεδομένων μέσα σε ένα πρόγραμμα Python. Αντίστοιχα, η συνάρτηση json.dump() επιτρέπει τη μετατροπή των Python αντικειμένων σε μορφή JSON και την αποθήκευσή τους σε αρχεία. Αυτή η διαδικασία είναι κρίσιμη σε εφαρμογές που απαιτούν την αποθήκευση ή τη μεταφορά δεδομένων μέσω δικτύου σε μορφή JSON, όπως για παράδειγμα σε RESTful APIs που χρησιμοποιείται και την υπάρχουσα εργασία [64]. Υποστηρίζει επίσης την επεξεργασία σύνθετων δομών δεδομένων, καθιστώντας την ιδανική για τη διαχείριση δεδομένων που προέρχονται από διάφορες πηγές. Η δυνατότητα αυτή επιτρέπει τη γρήγορη ανάπτυξη εφαρμογών που απαιτούν τη διαχείριση και την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων, διασφαλίζοντας παράλληλα τη συμβατότητα και την ευελιξία [65]. Η βιβλιοθήκη json της Python αποτελεί ένα εργαλείο που επιτρέπει την επεξεργασία δεδομένων σε μορφή JSON (JavaScript Object Notation), ένα δημοφιλές πρότυπο για τη δομή δεδομένων που είναι ευρέως χρησιμοποιούμενο σε εφαρμογές web και APIs. Τα JSON αρχεία είναι γνωστά για την απλότητα και την αναγνωσιμότητά τους, καθιστώντας τα εύκολα στη χρήση από προγραμματιστές και κατανοητά από μηχανές και ανθρώπους. Η βιβλιοθήκη json διευκολύνει τη διαδικασία ανάγνωσης και εγγραφής JSON δεδομένων με μεθόδους που μετατρέπουν τα JSON δεδομένα σε Python αντικείμενα και αντίστροφα [65].

Η συνάρτηση json.load() χρησιμοποιείται για την ανάγνωση JSON δεδομένων από ένα αρχείο και την αποτύπωσή τους σε Python αντικείμενα, όπως λίστες ή λεξικά. Αυτό επιτρέπει την εύκολη επεξεργασία των δεδομένων μέσα σε ένα πρόγραμμα Python. Αντίστοιχα, η συνάρτηση json.dump() επιτρέπει τη μετατροπή των Python αντικειμένων σε μορφή JSON και την αποθήκευσή τους σε αρχεία. Αυτή η διαδικασία είναι κρίσιμη σε εφαρμογές που απαιτούν την αποθήκευση ή τη μεταφορά δεδομένων μέσω δικτύου σε μορφή JSON, όπως για παράδειγμα σε RESTful APIs που χρησιμοποιείται και την υπάρχουσα εργασία [64]. Υποστηρίζει επίσης την επεξεργασία σύνθετων δομών δεδομένων, καθιστώντας την ιδανική για τη διαχείριση δεδομένων που προέρχονται από διάφορες πηγές. Η δυνατότητα αυτή επιτρέπει τη γρήγορη ανάπτυξη εφαρμογών που απαιτούν τη

διαχείριση και την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων, διασφαλίζοντας παράλληλα τη συμβατότητα και την ευελιξία [65].

- Διαδικασία μετατροπής αρχείων χρησιμοποιώντας τη βιβλιοθήκη csv
 Η διαδικασία μετατροπής δεδομένων σε μορφή CSV είναι εξαιρετικά χρήσιμη σε περιπτώσεις όπου απαιτείται η αποθήκευση μεγάλων συνόλων δεδομένων με αποδοτικό τρόπο. Η συνάρτηση `csv.writer()` επιτρέπει τη δημιουργία αρχείων CSV τα οποία παράγονται από δεδομένα που αποθηκεύονται σε Python αντικείμενα, όπως λίστες ή λεξικά. Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική για την ανάλυση δεδομένων και την εξαγωγή τους σε μορφές που μπορούν να χρησιμοποιηθούν σε άλλες εφαρμογές, όπως υπολογιστικά φύλλα ή βάσεις δεδομένων [66].
 Επιπλέον, η συνάρτηση `csv.reader()` επιτρέπει την ανάγνωση δεδομένων από αρχεία CSV και τη μετατροπή τους σε Python αντικείμενα για περαιτέρω επεξεργασία. Αυτό διευκολύνει την εισαγωγή δεδομένων από εξωτερικές πηγές και τη χρήση τους σε προγράμματα Python, επιτρέποντας την εύκολη ενσωμάτωση δεδομένων από διάφορα συστήματα [67]. Η βιβλιοθήκη csv της Python παρέχει εργαλεία για την ανάγνωση και εγγραφή δεδομένων σε μορφή CSV (Comma-Separated Values), μια μορφή που χρησιμοποιείται ευρέως για την αποθήκευση δομημένων δεδομένων. Τα αρχεία CSV είναι ιδιαίτερα χρήσιμα για τη μεταφορά δεδομένων μεταξύ διαφορετικών συστημάτων ή εφαρμογών, καθώς είναι ευανάγνωστα τόσο από μηχανές όσο και από ανθρώπους [67]. Η διαδικασία μετατροπής δεδομένων σε μορφή CSV είναι εξαιρετικά χρήσιμη σε περιπτώσεις όπου απαιτείται η αποθήκευση μεγάλων συνόλων δεδομένων με αποδοτικό τρόπο. Η συνάρτηση `csv.writer()` επιτρέπει τη δημιουργία αρχείων CSV τα οποία παράγονται από δεδομένα που αποθηκεύονται σε Python αντικείμενα, όπως λίστες ή λεξικά. Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική για την ανάλυση δεδομένων και την εξαγωγή τους σε μορφές που μπορούν να χρησιμοποιηθούν σε άλλες εφαρμογές, όπως υπολογιστικά φύλλα ή βάσεις δεδομένων [66].
 Επιπλέον, η συνάρτηση `csv.reader()` επιτρέπει την ανάγνωση δεδομένων από αρχεία CSV και τη μετατροπή τους σε Python αντικείμενα για περαιτέρω επεξεργασία. Αυτό διευκολύνει την εισαγωγή δεδομένων από εξωτερικές πηγές και τη χρήση τους σε προγράμματα Python, επιτρέποντας την εύκολη ενσωμάτωση δεδομένων από διάφορα συστήματα [67].
- Κύριες συναρτήσεις που χρησιμοποιήθηκαν
 Η συνάρτηση `json.load()` είναι ουσιαστική για την ανάγνωση δεδομένων JSON από ένα αρχείο και την αποτύπωσή τους σε Python αντικείμενα. Η διαδικασία αυτή είναι κρίσιμη σε εφαρμογές που απαιτούν την εισαγωγή δεδομένων από αρχεία JSON για περαιτέρω επεξεργασία μέσα στο πρόγραμμα [66]. Η συνάρτηση `json.dump()` χρησιμοποιείται για την αντίστροφη διαδικασία, επιτρέποντας την αποθήκευση Python αντικειμένων σε αρχεία JSON, διασφαλίζοντας τη συμβατότητα των δεδομένων με άλλες εφαρμογές και συστήματα [44].
 Η `csv.writer()` είναι εξαιρετικά χρήσιμη για τη δημιουργία αρχείων CSV από Python αντικείμενα, επιτρέποντας την αποθήκευση δομημένων δεδομένων με τρόπο που είναι εύκολα αναγνώσιμος και χρησιμοποιήσιμος από άλλα συστήματα. Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική σε εφαρμογές ανάλυσης δεδομένων, όπου η αποδοτική διαχείριση και αποθήκευση δεδομένων είναι κρίσιμη [65][67]. Οι συναρτήσεις `json.load()`, `json.dump()`, και `csv.writer()` είναι κεντρικές στη διαχείριση δεδομένων σε μορφή JSON

και CSV και χρησιμοποιούνται ευρέως σε προγράμματα Python για την ανάγνωση, εγγραφή και μετατροπή δεδομένων.

Η συνάρτηση `json.load()` είναι ουσιαστική για την ανάγνωση δεδομένων JSON από ένα αρχείο και την αποτύπωσή τους σε Python αντικείμενα. Η διαδικασία αυτή είναι κρίσιμη σε εφαρμογές που απαιτούν την εισαγωγή δεδομένων από αρχεία JSON για περαιτέρω επεξεργασία μέσα στο πρόγραμμα [66]. Η συνάρτηση `json.dump()` χρησιμοποιείται για την αντίστροφη διαδικασία, επιτρέποντας την αποθήκευση Python αντικειμένων σε αρχεία JSON, διασφαλίζοντας τη συμβατότητα των δεδομένων με άλλες εφαρμογές και συστήματα [44]. Η `csv.writer()` είναι εξαιρετικά χρήσιμη για τη δημιουργία αρχείων CSV από Python αντικείμενα, επιτρέποντας την αποθήκευση δομημένων δεδομένων με τρόπο που είναι εύκολα αναγνώσιμος και χρησιμοποιήσιμος από άλλα συστήματα. Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική σε εφαρμογές ανάλυσης δεδομένων, όπου η αποδοτική διαχείριση και αποθήκευση δεδομένων είναι κρίσιμη [65][67].

Βιβλιοθήκες Pathlib και ZipFile

- Η Διαχείριση των διαδρομών αρχείων σε διαφορετικά λειτουργικά συστήματα με το Pathlib

Μέσω της κλάσης αυτής, οι προγραμματιστές μπορούν να δημιουργούν, να διαχειρίζονται και να επεξεργάζονται διαδρομές αρχείων με διάφορους τρόπους. Συγκεκριμένα η μέθοδος `Path.glob()` επιτρέπει την αναζήτηση αρχείων με βάση μοτίβα, κάτι που είναι ιδιαίτερα χρήσιμο για την εύρεση συγκεκριμένων τύπων αρχείων σε φακέλους και υποφακέλους [68]. Η συγκεκριμένη βιβλιοθήκη παρέχει επίσης ευκολότερη διαχείριση των δικαιωμάτων πρόσβασης και των διαφορετικών ιδιοτήτων των αρχείων, όπως η ανάγνωση και η εγγραφή δεδομένων απευθείας από το αντικείμενο διαδρομής. Με αυτόν τον τρόπο πραγματοποιείται η ανάπτυξη καθαρού και συντηρήσιμου κώδικα [35]. Η βιβλιοθήκη Pathlib της Python προσφέρει έναν αντικειμενοστραφή τρόπο διαχείρισης των file paths, διευκολύνοντας την εργασία με το σύστημα αρχείων ανεξαρτήτως λειτουργικού συστήματος. Η κύρια κλάση που χρησιμοποιείται είναι η `Path`, η οποία μπορεί να αντιπροσωπεύει διαδρομές αρχείων και φακέλων. Αυτό επιτρέπει στους προγραμματιστές να γράφουν κώδικα που είναι φορητός ανάμεσα σε διαφορετικά λειτουργικά συστήματα, όπως τα Windows, το Linux και το macOS, χωρίς να χρειάζεται να ανησυχούν για τις διαφορές στη σύνταξη διαδρομών [69].

Μέσω της κλάσης αυτής, οι προγραμματιστές μπορούν να δημιουργούν, να διαχειρίζονται και να επεξεργάζονται διαδρομές αρχείων με διάφορους τρόπους. Συγκεκριμένα η μέθοδος `Path.glob()` επιτρέπει την αναζήτηση αρχείων με βάση μοτίβα, κάτι που είναι ιδιαίτερα χρήσιμο για την εύρεση συγκεκριμένων τύπων αρχείων σε φακέλους και υποφακέλους [68]. Η συγκεκριμένη βιβλιοθήκη παρέχει επίσης ευκολότερη διαχείριση των δικαιωμάτων πρόσβασης και των διαφορετικών ιδιοτήτων των αρχείων, όπως η ανάγνωση και η εγγραφή δεδομένων απευθείας από το αντικείμενο διαδρομής. Με αυτόν τον τρόπο πραγματοποιείται η ανάπτυξη καθαρού και συντηρήσιμου κώδικα [35].

- Συμπίεση και αποσυμπίεση αρχείων χρησιμοποιώντας τη ZipFile
Η ZipFile υποστηρίζει επίσης τη δημιουργία συμπιεσμένων αρχείων με διαφορετικούς αλγόριθμους συμπίεσης, όπως ZIP DEFLATED και ZIP BZIP2, ανάλογα με τις απαιτή-

σεις του χρήστη. Αυτό επιτρέπει τη βελτιστοποίηση της διαδικασίας συμπίεσης ανάλογα με τις ανάγκες απόδοσης ή αποθήκευσης. Επιπλέον, η `ZipFile` μπορεί να χειριστεί μεγάλα αρχεία ZIP μέσω της υποστήριξης του προτύπου ZIP64, το οποίο είναι απαραίτητο για αρχεία που υπερβαίνουν τα 4GB [68]. Η βιβλιοθήκη `ZipFile` της Python είναι ένα εργαλείο για τη δημιουργία, ανάγνωση, εγγραφή και εξαγωγή αρχείων ZIP. Τα αρχεία ZIP είναι μια δημοφιλής μορφή για τη συμπίεση και την αρχειοθέτηση δεδομένων, επιτρέποντας την αποθήκευση μεγάλων όγκων δεδομένων σε μικρότερο χώρο και την ευκολότερη μεταφορά τους [68]. Η κλάση `ZipFile` παρέχει μια σειρά από μεθόδους για τη διαχείριση αρχείων ZIP. Η μέθοδος `ZipFile.open()` επιτρέπει το άνοιγμα και την επεξεργασία αρχείων εντός του αρχείου ZIP, είτε σε λειτουργία ανάγνωσης είτε σε λειτουργία εγγραφής. Η μέθοδος `ZipFile.write()` χρησιμοποιείται για την προσθήκη αρχείων σε ένα αρχείο ZIP, ενώ η μέθοδος `ZipFile.extract()` επιτρέπει την εξαγωγή αρχείων από ένα αρχείο ZIP στο σύστημα αρχείων [68].

Η `ZipFile` υποστηρίζει επίσης τη δημιουργία συμπιεσμένων αρχείων με διαφορετικούς αλγόριθμους συμπίεσης, όπως ZIP DEFLATED και ZIP BZIP2, ανάλογα με τις απαιτήσεις του χρήστη. Αυτό επιτρέπει τη βελτιστοποίηση της διαδικασίας συμπίεσης ανάλογα με τις ανάγκες απόδοσης ή αποθήκευσης. Επιπλέον, η `ZipFile` μπορεί να χειριστεί μεγάλα αρχεία ZIP μέσω της υποστήριξης του προτύπου ZIP64, το οποίο είναι απαραίτητο για αρχεία που υπερβαίνουν τα 4GB [68].

- Κύριες συναρτήσεις που χρησιμοποιήθηκαν

Η συνάρτηση `Path.glob()` χρησιμοποιείται για την αναζήτηση αρχείων που ταιριάζουν σε ένα συγκεκριμένο μοτίβο εντός ενός φακέλου ή υποφακέλου. Αυτό είναι εξαιρετικά χρήσιμο για τη διαχείριση μεγάλων συνόλων αρχείων, όπου ο χρήστης χρειάζεται να εντοπίσει συγκεκριμένα αρχεία βάσει της επέκτασής τους ή άλλων χαρακτηριστικών [35]. Έπειτα χρησιμοποιήθηκε η συνάρτηση `ZipFile.open()` που επιτρέπει την πρόσβαση σε αρχεία εντός ενός αρχείου ZIP χωρίς να απαιτείται η πλήρης αποσυμπίεση του αρχείου. Αυτό είναι χρήσιμο για την ανάγνωση ή τη μετατροπή δεδομένων απευθείας από το συμπιεσμένο αρχείο, εξοικονομώντας χρόνο και πόρους συστήματος [68]. Αντίστοιχα, η συνάρτηση `ZipFile.write()` χρησιμοποιείται για την προσθήκη νέων αρχείων σε ένα αρχείο ZIP, επιτρέποντας την αρχειοθέτηση δεδομένων σε συμπιεσμένη μορφή για μελλοντική χρήση ή μεταφορά [68]. Οι συναρτήσεις `Path.glob()`, `ZipFile.open()` και `ZipFile.write()` είναι θεμελιώδεις για εργασία, όπως και η υπάρχουσα, η όποια βασίζεται στην διαχείριση αρχείων και διαδρομών στην Python.

Η συνάρτηση `Path.glob()` χρησιμοποιείται για την αναζήτηση αρχείων που ταιριάζουν σε ένα συγκεκριμένο μοτίβο εντός ενός φακέλου ή υποφακέλου. Αυτό είναι εξαιρετικά χρήσιμο για τη διαχείριση μεγάλων συνόλων αρχείων, όπου ο χρήστης χρειάζεται να εντοπίσει συγκεκριμένα αρχεία βάσει της επέκτασής τους ή άλλων χαρακτηριστικών [35]. Έπειτα χρησιμοποιήθηκε η συνάρτηση `ZipFile.open()` που επιτρέπει την πρόσβαση σε αρχεία εντός ενός αρχείου ZIP χωρίς να απαιτείται η πλήρης αποσυμπίεση του αρχείου. Αυτό είναι χρήσιμο για την ανάγνωση ή τη μετατροπή δεδομένων απευθείας από το συμπιεσμένο αρχείο, εξοικονομώντας χρόνο και πόρους συστήματος [68]. Αντίστοιχα, η συνάρτηση `ZipFile.write()` χρησιμοποιείται για την προσθήκη νέων αρχείων σε ένα αρχείο ZIP, επιτρέποντας την αρχειοθέτηση δεδομένων σε συμπιεσμένη μορφή για μελλοντική χρήση ή μεταφορά [68].

4.1.4 Python και Neo4j

Ανάλυση της σχέσης μεταξύ Python και Neo4j

Η Neo4j εκτός από τις βασικές δυνατότητες, προσφέρει διάφορες βιβλιοθήκες και εργαλεία που επιτρέπουν τη διασύνδεση με εφαρμογές Python. Χαρακτηριστική βιβλιοθήκη μεταξύ άλλων είναι η neo4j-python-driver, η οποία επιτρέπει στους προγραμματιστές να εκτελούν ερωτήματα Cypher και να διαχειρίζονται δεδομένα σε βάσεις γραφημάτων Neo4j μέσω της Python. Η συγκεκριμένη βιβλιοθήκη μετατρέπει τη Python σε μια ισχυρή γλώσσα για την εκτέλεση ερωτημάτων γραφημάτων, τη δημιουργία κόμβων και σχέσεων, την εξαγωγή δεδομένων για ανάλυση, και την ανάπτυξη εφαρμογών που χρησιμοποιούν γραφικά μοντέλα δεδομένων. Δημιουργώντας δυναμικά ερωτήματα Cypher μέσα από τις εφαρμογές Python επιτρέπει στους προγραμματιστές την αυτοματοποίηση και την ανάλυση πολύπλοκων δικτύων. Τέλος, άλλες βιβλιοθήκες Python, όπως η py2neo, παρέχουν ακόμα μεγαλύτερη ευκολία στη διαχείριση δεδομένων γραφημάτων, προσφέροντας αντικειμενοστραφείς διεπαφές για την αλληλεπίδραση με τα δεδομένα.

Παράδειγμα χρήσης της Python για διασύνδεση με τη Neo4j

Η βασική διαδικασία διασύνδεσης της Python με τη Neo4j περιλαμβάνει την εγκατάσταση του neo4j-python-driver μέσω του pip, τη σύνδεση με τον διακομιστή Neo4j και την εκτέλεση ερωτημάτων Cypher. Ένα απλό παράδειγμα σύνδεσης με τη βάση Neo4j είναι το εξής:

```
FROM neo4j import GraphDatabase CLASS
Neo4jConnection:
DEF \ init\ (self, uri, user, password):
self.driver = GraphDatabase.driver(uri, auth=(user,password)) DEF
close(self): self.driver.close()
DEF run\_query(self, query): With self.driver.session() as
session: Result session.run(query)
Return [record for record in result]
Conn = Neo4jConnection("bolt://localhost:7687", "neo4j",
"password") Result = conn.run.query("MATCH (n) RETURN n LIMIT
10")
Print(result) Conn.close()
```

Στο συγκεκριμένο παράδειγμα, δημιουργείται μια σύνδεση με τον διακομιστή Neo4j και εκτελείται ένα απλό ερώτημα Cypher που επιστρέφει δέκα κόμβους από τη βάση δεδομένων.

Πλεονεκτήματα και μειονεκτήματα της χρήσης Python με τη Neo4j

1. Πλεονεκτήματα:

- Ευκολία στη χρήση : Η Python είναι γνωστή για την απλή και κατανοητή σύνταξή της, καθιστώντας τη χρήση της ιδιαίτερα φιλική για προγραμματιστές που θέλουν να διαχειριστούν δεδομένα γραφημάτων.

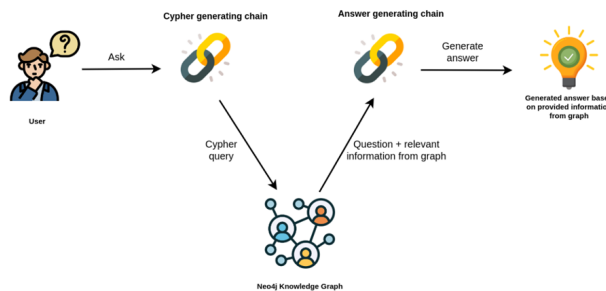
- Ευέλικτη διαχείριση δεδομένων: Η Python, μέσω βιβλιοθηκών, μπορεί να επεξεργάζεται δεδομένα που εξάγονται από τη Neo4j, επιτρέποντας τη δημιουργία πολύπλοκων αναλύσεων και οπτικοποιήσεων.
- Υποστήριξη βιβλιοθηκών: Οι βιβλιοθήκες της Python προσφέρουν μια μεγάλη γκάμα εργαλείων που μπορούν να συνεργαστούν με τη Neo4j για ανάλυση δεδομένων, μηχανική μάθηση και επεξεργασία φυσικής γλώσσας.

2. Μειονεκτήματα:

- Απόδοση: Αν και η Python είναι μια εξαιρετικά γρήγορη γλώσσα για ανάπτυξη, μπορεί να υστερεί σε απόδοση όταν χρησιμοποιείται για τη διαχείριση μεγάλων όγκων δεδομένων σε πραγματικό χρόνο.
- Σύνθετα ερωτήματα: Σε περιπτώσεις όπου η πολυπλοκότητα των ερωτημάτων είναι υψηλή και η κλίμακα των δεδομένων είναι μεγάλη, η απόδοση της Python σε συνδυασμό με τη Neo4j μπορεί να επηρεαστεί αρνητικά.

4.2 Εισαγωγή στη γλώσσα Cypher

Η Cypher είναι μια γλώσσα ερωταποκρίσεων σχεδιασμένη ειδικά για τη διαχείριση και την αναζήτηση δεδομένων σε βάσεις δεδομένων γραφημάτων, όπως η Neo4j. Είναι αρκετά εμπνευσμένη από τη σύνταξη της SQL, ενώ προσφέρει μια διαισθητική και εκφραστική προσέγγιση για την εργασία με γραφικά δεδομένα. Έτσι επιτρέπει στους χρήστες να περιγράφουν τα πρότυπα που θέλουν να βρουν στο γράφημα με απλό και κατανοητό τρόπο.



Σχήμα 4.3: Ροή της γλώσσας ερωταποκρίσεων Cypher [70]

4.2.1 Βασικά Χαρακτηριστικά της Cypher

- Δηλωτική φύση: Οι χρήστες περιγράφουν τι θέλουν να ανακτήσουν χωρίς να χρειάζεται να καθορίσουν πώς να το κάνουν, αφήνοντας τη βελτιστοποίηση στον κινητήρα της βάσης δεδομένων.
- Διαισθητική σύνταξη: Χρησιμοποιεί συμβολισμούς και δομές που είναι εύκολα κατανοητές στους χρήστες, καθιστώντας την προσβάσιμη ακόμα και σε όσους δεν έχουν σχετική εμπειρία με γραφικές βάσεις δεδομένων.

- Έκφραση σύνθετων ερωτημάτων: Επιτρέπει την έκφραση σύνθετων ερωτημάτων με λιτό και σαφή τρόπο.
- Επεκτασιμότητα: Υποστηρίζει προηγμένες λειτουργίες, όπως δημιουργία, ενημέρωση και διαγραφή κόμβων και σχέσεων, καθώς και λειτουργίες συσσωμάτωσης και φίλτρων.
- Συμβατότητα: Ενσωματώνεται εύκολα με διάφορες γλώσσες προγραμματισμού μέσω οδηγών και API [31].

4.2.2 Βασική σύνταξη και δομή της Cypher

Η Cypher χρησιμοποιεί ένα σύνολο συμβολισμών και λέξεων-κλειδιών για την περιγραφή των στοιχείων του γραφήματος. Συγκεκριμένα:

- Κόμβοι: Αναπαρίστανται με παρενθέσεις. Π.χ., (n) αντιπροσωπεύει έναν κόμβο.
- Σχέσεις: Αναπαρίστανται με αγκύλες και βέλη. Π.χ., ()-[:RELATION]->() αντιπροσωπεύει μια κατευθυνόμενη σχέση μεταξύ δύο κόμβων.
- Ιδιότητες: Αποθηκεύονται ως ζεύγη κλειδιού-τιμής μέσα σε αγκύλες. Π.χ., (n name:'Alice').
- Ετικέτες: Χρησιμοποιούνται για την κατηγοριοποίηση κόμβων. Π.χ., (n:Person) [71].

Παρακάτω παραθέτω ένα απλό παράδειγμα χρήσης Cypher το αποτέλεσμα του οποίου είναι η ανάκτηση όλων των ατόμων που είναι φίλοι με την Alice :

```
MATCH (a:Person) -[:FRIEND|_OF]->(b:Person)
WHERE a.name = 'Alice'
RETURN b.name
```

4.2.3 Πλεονεκτήματα της χρήσης της Cypher

- Ευκολία μάθησης: Η σύνταξή της είναι απλή και μοιάζει με την φυσική γλώσσα, καθιστώντας την εύκολη στην εκμάθηση.
- Αποδοτικότητα: Επιτρέπει την γρήγορη και αποδοτική αναζήτηση σε μεγάλα και περίπλοκα γραφήματα.
- Σαφήνεια: Τα ερωτήματα είναι ευανάγνωστα και εύκολα στην κατανόηση, διευκολύνοντας τη συντήρηση και τον διαμοιρασμό τους.
- Πλούσιες λειτουργίες: Παρέχει ένα ευρύ φάσμα λειτουργιών για την εκτέλεση πολύπλοκων αναλύσεων και μετασχηματισμών δεδομένων.

4.2.4 Εφαρμογές και χρήση της Cypher

Η Cypher χρησιμοποιείται ευρέως σε εφαρμογές που απαιτούν γρήγορη και αποδοτική αναζήτηση σε συνδεδεμένα δεδομένα. Μερικές από τις εφαρμογές της περιλαμβάνουν:

- Ανάλυση κοινωνικών δικτύων: Δηλαδή για την αναζήτηση και ανάλυση σχέσεων μεταξύ χρηστών και περιεχομένου.
- Συστήματα συστάσεων: Για την εξαγωγή προτύπων με βάση συνδέσεων μεταξύ προϊόντων και χρηστών.
- Ανίχνευση απάτης: Για την εύρεση ύποπτων συναλλαγών και μοτίβων μέσω της ανάλυσης των σχέσεων μεταξύ οντοτήτων.
- Διαχείριση δικτύων: Συμβάλει στην χαρτογράφηση και παρακολούθηση των συνδέσεων σε πολύπλοκα δίκτυα υποδομής [32].

4.2.5 Τρόποι εισαγωγής δεδομένων στη Neo4j

Η Cypher επιτρέπει την εισαγωγή δεδομένων στη Neo4j μέσω δηλωτικών ερωτημάτων, όπως οι εντολές CREATE, MERGE, LOAD CSV και άλλες.

- CREATE και MERGE: Χρησιμοποιούνται για τη δημιουργία νέων κόμβων και σχέσεων. Η CREATE απλά δημιουργεί νέους κόμβους, ενώ η MERGE εξασφαλίζει ότι οι κόμβοι ή οι σχέσεις δεν υπάρχουν ήδη πριν την εισαγωγή, καθιστώντας την χρήσιμη για την αποφυγή διπλών εγγραφών.

Παράδειγμα χρήσης CREATE και MERGE:

```
CREATE (n:Person {name: 'Alice', age: 30})
MERGE (n:Person {name: 'Bob'})
```

- LOAD CSV: Η εντολή LOAD CSV επιτρέπει την εισαγωγή δεδομένων από αρχεία CSV, τα οποία μπορούν να χρησιμοποιηθούν για τη δημιουργία κόμβων, σχέσεων και ιδιοτήτων στη Neo4j.

Παράδειγμα χρήσης LOAD CSV:

```
LOAD CSV WITH HEADERS FROM 'file:///employees.csv' AS
row CREATE (e:Employee {id: row.id, name:
row._name, department: row._department})
```

- Πλεονεκτήματα και Μειονεκτήματα του LOAD CSV:

1. Πλεονεκτήματα:

- Ευελιξία: Επιτρέπει την εισαγωγή δεδομένων από εξωτερικά αρχεία και τον δυναμικό χειρισμό αυτών των δεδομένων.
- Δηλωτική φύση: Παρέχει μια καθαρή και εύκολη στη διαχείριση μέθοδο για την εισαγωγή δεδομένων.

2. Μειονεκτήματα:

- Απόδοση: Για μεγάλες ποσότητες δεδομένων, η απόδοση της μπορεί να είναι περιορισμένη, λόγω της ανάγκης για διαδοχικές εισαγωγές και ενημερώσεις. Σε σενάρια με πολύ μεγάλους όγκους δεδομένων, μπορεί να χρειαστεί μεγάλο χρόνο για την ολοκλήρωση της εισαγωγής.
- Κατανάλωση Πόρων: Η εκτέλεση μεγάλων αρχείων CSV μέσω της Cypher μπορεί να απαιτήσει υψηλή κατανάλωση μνήμης και CPU, οδηγώντας σε προβλήματα κλιμάκωσης [71].

Εάν η χρήση της LOAD CSV είναι αναγκαία σε μεγάλα δεδομένα, παρακάτω παρατίθενται ορισμένες βέλτιστες πρακτικές που θα μπορούσαν να βοηθήσουν στη βελτιστοποίηση:

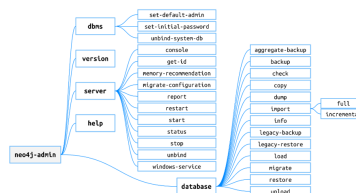
- Χρήση δεικτών: Δημιουργία δεικτών πριν από την εισαγωγή για τη βελτιστοποίηση της απόδοσης.
- Διαχωρισμός εργασιών: Διαχωρισμός των αρχείων CSV σε μικρότερα κομμάτια και εισαγωγή τους τμηματικά.
- Παράλληλες διεργασίες: Εισαγωγή δεδομένων σε παράλληλες διεργασίες για την αύξηση της απόδοσης [71].

4.3 Εισαγωγή δεδομένων μέσω Neo4j Admin Import

4.3.1 Τι είναι το Neo4j Admin Import

Το Neo4j Admin Import είναι ένα εργαλείο γραμμής εντολών που προσφέρει τη δυνατότητα εισαγωγής μεγάλων όγκων δεδομένων στη Neo4j με εξαιρετική απόδοση. Είναι σχεδιασμένο για μαζικές εισαγωγές δεδομένων και αποτελεί την προτιμώμενη μέθοδο όταν αντιμετωπίζουμε τεράστιους όγκους δεδομένων που πρέπει να εισαχθούν στη βάση δεδομένων [72]

4.3.2 Πλεονεκτήματα του Neo4j Admin Import



Σχήμα 4.4: Η εντολή Neo4j-admin και οι επεκτάσεις της

- Απόδοση: Το Neo4j Admin Import είναι πολύ πιο γρήγορο από τις εντολές Cypher για εισαγωγή μεγάλων δεδομένων, καθώς έχει βελτιστοποιηθεί για αυτή τη λειτουργία.
- Κλίμακα: Μπορεί να χειριστεί δισεκατομμύρια κόμβους και σχέσεις, καθιστώντας το ιδανικό για μεγάλα έργα και μαζικές εισαγωγές.
- Αποφυγή Μεμονωμένων Ερωτημάτων: Αντί να εκτελεί χιλιάδες ή εκατομμύρια ερωτήματα Cypher, το Neo4j Admin Import εισάγει τα δεδομένα με απευθείας γραφή σε αρχεία αποθήκευσης της βάσης δεδομένων, παρακάμπτοντας το επίπεδο των ερωτημάτων Cypher.

4.3.3 Μειονεκτήματα του Neo4j Admin Import

- Μόνο για Νέες Βάσεις Δεδομένων: Το Neo4j Admin Import μπορεί να χρησιμοποιηθεί μόνο για την εισαγωγή δεδομένων σε νέες βάσεις δεδομένων [73]. Δεν υποστηρίζει την εισαγωγή δεδομένων σε ήδη υπάρχουσες βάσεις δεδομένων.
- Μη δηλωτικό: Ενώ η Cypher παρέχει ένα ευανάγνωστο, δηλωτικό τρόπο για την εισαγωγή δεδομένων, το Neo4j Admin Import απαιτεί την προετοιμασία των δεδομένων σε συγκεκριμένη μορφή και τη χρήση εργαλείων γραμμής εντολών.

4.3.4 Σύγκριση Cypher και Neo4j Admin Import για εισαγωγή δεδομένων

Η επιλογή μεταξύ Cypher και Neo4j Admin Import εξαρτάται από το σενάριο χρήσης και τον όγκο των δεδομένων:

- Cypher: Είναι καταλληλότερο για μικρότερες εισαγωγές δεδομένων ή για περιπτώσεις όπου η βάση δεδομένων είναι ήδη σε χρήση και οι εισαγωγές δεδομένων γίνονται συχνά ή σε μικρότερες ποσότητες.
- Neo4j Admin Import: Είναι η καλύτερη επιλογή για μεγάλες, μαζικές εισαγωγές δεδομένων, ιδίως κατά τη δημιουργία νέων βάσεων δεδομένων. Η ταχύτητα και η αποδοτικότητα που προσφέρει είναι απaráμιλλη σε περιπτώσεις μεγάλων όγκων δεδομένων.

4.4 Συμπεράσματα

Η Cypher αποτελεί μια ισχυρή γλώσσα για τη διαχείριση γραφημάτων, αλλά όταν πρόκειται για εισαγωγή μεγάλων όγκων δεδομένων, οι επιλογές της μπορεί να μην είναι οι βέλτιστες για όλες τις περιπτώσεις. Η εντολή LOAD CSV είναι ευέλικτη και χρήσιμη για μεσαίες εισαγωγές δεδομένων, αλλά η απόδοσή της σε πολύ μεγάλες κλίμακες είναι περιορισμένη. Για μαζικές εισαγωγές δεδομένων, το Neo4j Admin Import αποτελεί την πιο αποδοτική και γρήγορη λύση, παρότι απαιτεί τη δημιουργία νέας βάσης δεδομένων.

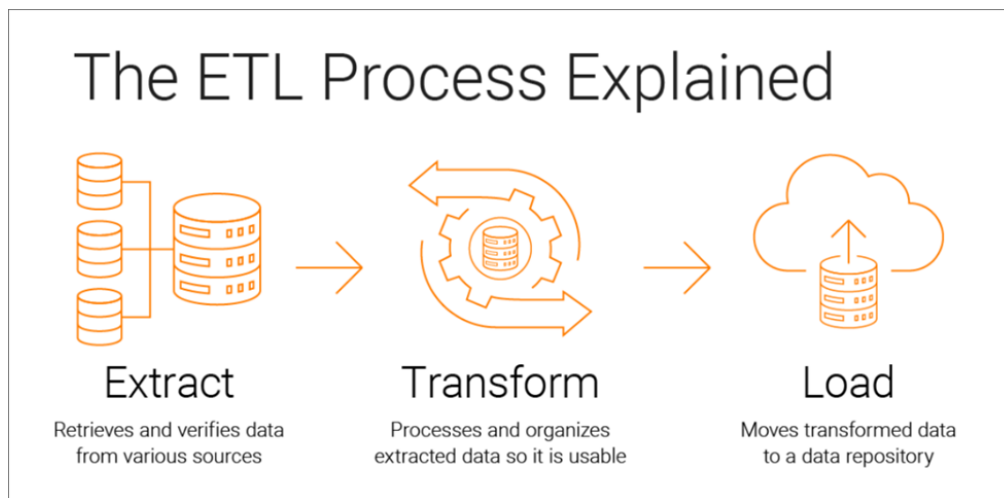
Η επιλογή της κατάλληλης μεθόδου εισαγωγής δεδομένων εξαρτάται από το μέγεθος των δεδομένων, τις απαιτήσεις απόδοσης και τη φύση της εφαρμογής. Η χρήση της Cypher για ερωταποκρίσεις και μικρότερες εισαγωγές μπορεί να είναι ιδανική για εφαρμογές σε συνεχή λειτουργία, ενώ για αρχικές μαζικές εισαγωγές, η χρήση εξειδικευμένων εργαλείων όπως το Neo4j Admin Import είναι αναγκαία.

Κεφάλαιο 5

Τεχνική ανάλυση της Εφαρμογής

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα παρουσιάσω την σχεδίαση και την ανάπτυξη της εφαρμογής που δημιουργήθηκε για την αυτοματοποίηση της διαδικασίας εξαγωγής, μετασχηματισμού και φόρτωσης δεδομένων (ETL) από το OpenAlex στη βάση δεδομένων Neo4j. Η εφαρμογή αυτή έχει ως κύριο στόχο τη διευκόλυνση της ανάλυσης μεγάλων όγκων και συνόλων δεδομένων στον τομέα της ακαδημαϊκής έρευνας, επιτρέποντας τη χρήση γράφων δεδομένων για προηγμένες αναλύσεις και εξαγωγής συμπερασμάτων.



Σχήμα 5.1: Διαδικασία ETL επεξηγηματικά

5.2 Σύντομη επισκόπηση της Εφαρμογής

Όπως προανέφερα η πτυχιακή μου εργασία και η εφαρμογή της σχεδιάστηκε με σκοπό να διευκολύνει την διαδικασία ETL (Extract, Transform, Load), η οποία ξεκινά με την ανάκτηση δεδομένων από το OpenAlex, που φιλοξενείται στο AWS S3, και καταλήγει στην εισαγωγή αυτών των δεδομένων στη βάση δεδομένων Neo4j για την ανάλυση τους μέσω Cypher Queries.

Η αρχιτεκτονική της εφαρμογής βασίζεται στην γλώσσα προγραμματισμού Python, χρησιμοποιώντας μια σειρά από βιβλιοθήκες και scripts που το καθένα διαχειρίζεται και ένα από τα διάφορα στάδια του ETL pipeline.

Η δημιουργία της βάσης δεδομένων Neo4j είναι κεντρικό στοιχείο της εφαρμογής, καθώς επιτρέπει την εκτέλεση σύνθετων ερωτημάτων σε ακαδημαϊκά δεδομένα, αξιοποιώντας τις δυνατότητες που παρέχει η τεχνολογία των γραφημάτων. Η σχεδίαση της εφαρμογής επικεντρώνεται στην αποδοτικότητα και την ακρίβεια, διασφαλίζοντας ότι τα δεδομένα επεξεργάζονται σωστά και εισάγονται με ακρίβεια στη Neo4j, έτοιμα για περαιτέρω ανάλυση. Για να επιτευχθεί αυτό χρειάστηκαν πολλαπλές προσπάθειες και αλλαγές τόσο σε επίπεδο κώδικα κατά την διάρκεια του import καθώς και στο φιλτράρισμα των δεδομένων της OpenAlex ώστε να είναι συμβατά και “καθαρά” για την εισαγωγή τους στην βάση μέσω του neo4j-admin tool.

Οι στόχοι της εφαρμογής περιλαμβάνουν τη διαχείριση μεγάλων όγκων δεδομένων με τρόπο που να επιτρέπει την εύκολη πρόσβαση και ανάλυση, την αυτοματοποίηση των διαδικασιών για την αποφυγή χειροκίνητων παρεμβάσεων, και την εξασφάλιση της ακεραιότητας και της συνοχής των δεδομένων κατά τη μεταφορά τους από την πηγή στον προορισμό.

Η προσέγγιση που ακολουθήθηκε στο φιλτράρισμα βασίστηκε στην παρατήρηση μου ότι αν κάποιο δεδομένο λείπει από μία θεματική κατηγορία, ενδέχεται να εμφανίζεται σε κάποια άλλη. Με αυτόν τον τρόπο, διασφαλίζεται ότι τα κρίσιμα δεδομένα που αφορούν τις σχέσεις μεταξύ των οντοτήτων διατηρούνται, ενώ τα κενά δεδομένα ή δεδομένα που δεν συμβάλλουν στη δημιουργία relationships απορρίπτονται. Όλα τα φιλτραρισμένα δεδομένα διατηρούνται σε συμπίεσμένη μορφή, για λόγους μείωσης των αναγκών αποθηκευτικού χώρου και ώστε να βρίσκεται το απομακρυσμένο μηχάνημα στην καλύτερη δυνατή κατάσταση.

Αυτή η προσέγγιση αναπτύχθηκε αρχικά τοπικά, σε μικρής κλίμακας βάση με την δημιουργία πειραματικών βάσεων, χρησιμοποιώντας έναν προσωπικό υπολογιστή με Windows 11, επεξεργαστή Intel CPU core i5-12400f, 32 GB RAM και 80 GB ελεύθερο χώρο στο δίσκο. Ωστόσο, καθώς η κλίμακα των δεδομένων έπρεπε να αυξηθεί, η διαδικασία ολοκληρώθηκε τελικά σε απομακρυσμένο μηχάνημα με λειτουργικό σύστημα Linux με μεγαλύτερο διαθέσιμο δίσκο. Αυτή η αλλαγή επέτρεψε την κλιμάκωση της επεξεργασίας και την αντιμετώπιση των προβλημάτων χώρου, διασφαλίζοντας την επιτυχή ολοκλήρωση του φιλτραρίσματος των δεδομένων πριν την εισαγωγή τους στη βάση Neo4j. Μετά το φιλτράρισμα, τα δεδομένα μετατρέπονται σε μορφή CSV μέσω του script json2csv_neo4j.py [67]. Αυτή η μετατροπή είναι απαραίτητη, καθώς η βάση δεδομένων Neo4j χρησιμοποιεί ειδικής μορφής CSV αρχεία για την κατασκευή των γράφων, των σχέσεων και των ιδιοτήτων της. Μέσω του script αυτού έγινε εφικτό να μετατρέψω τα φιλτραρισμένα JSON δεδομένα σε μορφή που να είναι συμβατή με τις απαιτήσεις της Neo4j, εξασφαλίζοντας τη σωστή αναπαράσταση των δεδομένων στις δομές της βάσης μετά το import.

Τέλος, το script py_admin_import.py αναλαμβάνει την αυτοματοποίηση της διαδικασίας εισαγωγής των δεδομένων στη Neo4j. Χρησιμοποιεί εντολές για την εισαγωγή των CSV αρχείων στη βάση δεδομένων, δημιουργώντας τους απαραίτητους κόμβους και τις σχέσεις που ορίζονται από τα δεδομένα του OpenAlex. Η διαδικασία αυτή εξασφαλίζει ότι τα δεδομένα είναι πλήρως ενσωματωμένα στη βάση δεδομένων και έτοιμα για περαιτέρω ανάλυση μέσω των δυνατοτήτων που παρέχει η Neo4j.

Η ενσωμάτωση με το AWS ήταν επίσης ένα κρίσιμο κομμάτι του data pipeline, καθώς η εφαρμογή για να δημιουργηθεί έπρεπε πρώτα να υπάρξει αλληλεπίδραση με το AWS S3 για την ανάκτηση των απαραίτητων αρχείων. Χρησιμοποιώντας εντολές του AWS CLI ενσωματωμέ-

νες σε Python, η εφαρμογή εξασφαλίζει ότι λαμβάνονται τα σωστά δεδομένα με βάση συγκεκριμένα κριτήρια, όπως ημερομηνίες ενημέρωσης ή τύπους αρχείων καταφέροντας έτσι να κάνουμε `import` τα πιο επικαιροποιημένα κομμάτια της OpenAlex. Αυτή η διαδικασία είναι απαραίτητη για τη σωστή λειτουργία του pipeline, διασφαλίζοντας ότι τα δεδομένα που χρησιμοποιούνται είναι πάντα τα πιο ενημερωμένα και σχετικά με τις ανάγκες της εφαρμογής.

5.3 Γενικές Αρχές και Απαιτήσεις

Η επιτυχημένη ανάπτυξη και εκτέλεση της εφαρμογής που σχεδιάστηκε για την επεξεργασία δεδομένων από το OpenAlex και την εισαγωγή τους στη βάση δεδομένων Neo4j απαιτεί συγκεκριμένες υλικοτεχνικές προδιαγραφές και ρυθμίσεις. Σε αυτήν την ενότητα, θα αναλυθεί τόσο οι απαιτήσεις σε υλικό και λογισμικό καθώς και τα αρχικά βήματα εγκατάστασης σε περιβάλλοντα AWS και τοπικά καθώς και τις απαραίτητες ρυθμίσεις που έγιναν.

5.3.1 Υλικό και Λογισμικό

Για την εκτέλεση της εφαρμογής, το σύστημα απαιτεί επεξεργαστική ισχύ και χώρο αποθήκευσης ικανούς να διαχειριστούν μεγάλα σύνολα δεδομένων. Οι βασικές προδιαγραφές περιλαμβάνουν:

- **Επεξεργαστή (CPU):** Ένας πολυπύρηνος επεξεργαστής, όπως ο Intel CPU core I5-12400f ή αντίστοιχος, είναι απαραίτητος για την ταχεία επεξεργασία δεδομένων και την εκτέλεση σύνθετων εντολών.
- **Μνήμη RAM:** Τουλάχιστον 32 GB RAM για να διασφαλιστεί η αποδοτική διαχείριση των δεδομένων κατά την εκτέλεση των scripts.
- **Χώρο Αποθήκευσης:** Ο χώρος στο δίσκο πρέπει να είναι τουλάχιστον 300 GB για την αποθήκευση των αποσυμπιεσμένων δεδομένων και των τελικών αρχείων CSV που θα χρησιμοποιηθούν για την εισαγωγή στη Neo4j. Στην αρχική φάση της ανάπτυξης, ένας χώρος 80 GB ήταν αρκετός για την τοπική επεξεργασία μικρότερων δεδομένων, όμως για την πλήρη κλιμάκωση απαιτείται σημαντικά μεγαλύτερος χώρος. Πρέπει να τονίσουμε ότι κατά την διάρκεια του `import` και κατά την εισαγωγή των δεδομένων δημιουργούνται προσωρινά αρχεία. Επομένως ο συνολικός υπολογισμός των δεδομένων εισαγωγής και της χωρητικότητας που πρέπει να διατίθεται στο σύστημα δεν είναι ένα προς ένα.
- **Λειτουργικό Σύστημα:** Η εφαρμογή μπορεί να εκτελεστεί τόσο σε Windows όσο και σε Unix-like περιβάλλοντα προχωρώντας σε μικρές αλλαγές, κυρίως όσον αφορά τις διαδρομές.
- **Python 3.8 ή νεότερη:** Η εφαρμογή χρησιμοποιεί Python 3.8 ή νεότερη, εκμεταλλευόμενη τις βιβλιοθήκες που είναι απαραίτητες για την αλληλεπίδραση με το AWS S3, τη διαχείριση των JSON αρχείων και την εισαγωγή δεδομένων στη Neo4j.

5.3.2 Αρχικά Βήματα Ρύθμισης σε AWS και Τοπικά Περιβάλλοντα

Η εγκατάσταση και ρύθμιση της εφαρμογής μπορεί να πραγματοποιηθεί τόσο σε AWS όσο και σε τοπικό περιβάλλον. Για την πραγματοποίηση της εγκατάστασης ακολουθούνται τα παρακάτω βήματα:

1. Τοπική Ρύθμιση:

- Εγκατάσταση Python και εξαρτήσεων: Γίνεται εγκατάσταση της έκδοσης Python 3.8 ή νεότερη και τις απαραίτητες βιβλιοθήκες μέσω pip. Οι βασικές βιβλιοθήκες που απαιτούνται περιλαμβάνουν boto3 για την αλληλεπίδραση με το AWS S3, neo4j για την εισαγωγή στη βάση δεδομένων και άλλες βιβλιοθήκες όπως os, json, και subprocess.
- ύθμιση περιβάλλοντος: Δημιουργούνται τα κατάλληλα directories για την αποθήκευση των δεδομένων που θα ληφθούν από το AWS S3, καθώς και για τα προσωρινά και τελικά αρχεία που θα χρησιμοποιηθούν στη Neo4j. Το script περιλαμβάνει τη ρύθμιση της τοπικής βάσης διαδρομών, η οποία μεταφράζεται κατάλληλα για κάθε λειτουργικό σύστημα.
- Λήψη δεδομένων: Χρησιμοποιείται το aws cli για τη λήψη των απαραίτητων δεδομένων από το OpenAlex. Αυτό μπορεί να γίνει είτε μέσω απευθείας εντολών στο AWS CLI είτε μέσω του script που παρέχεται, το οποίο εκτελεί τις κατάλληλες εντολές για τη λήψη των αρχείων από τα S3 buckets [74].

2. Ρύθμιση σε AWS

- Διαμόρφωση EC2 Instance: Δημιουργείται ένα EC2 instance στο AWS με τις παραπάνω προδιαγραφές υλικού. Η επιλογή του τύπου του instance εξαρτάται από τις απαιτήσεις σε υπολογιστική ισχύ και χώρο αποθήκευσης. Συνιστάται η χρήση instance τύπου m5.large ή καλύτερου, με επαρκή χώρο αποθήκευσης (EBS volumes) για την αποθήκευση και επεξεργασία των δεδομένων.
- Εγκατάσταση Python και εξαρτήσεων στο AWS: Ακριβώς όπως και στη τοπική ρύθμιση, εγκαθίσταται η Python και οι απαραίτητες βιβλιοθήκες. Είναι απαραίτητο οι διαδρομές στα scripts να είναι σωστά ρυθμισμένες για το περιβάλλον AWS.
- Διασφάλιση ασφάλειας και πρόσβασης: Ορίζονται τα σωστά δικαιώματα ασφαλείας για την πρόσβαση στα S3 buckets. Το EC2 instance πρέπει να έχει τα απαραίτητα IAM roles για την πρόσβαση στα δεδομένα του OpenAlex [74]. Μετά την αρχική ρύθμιση και την εγκατάσταση των απαιτούμενων εξαρτήσεων, η διαδικασία εισαγωγής δεδομένων στη βάση δεδομένων Neo4j είναι ένα κρίσιμο στάδιο στην εφαρμογή. Απαιτείται ειδική προετοιμασία τόσο των δεδομένων όσο και του περιβάλλοντος για να διασφαλιστεί ότι η εισαγωγή θα πραγματοποιηθεί χωρίς προβλήματα και με την καλύτερη δυνατή απόδοση.

5.3.3 Neo4j και Διαδικασία Εισαγωγής Δεδομένων

Η Neo4j είναι μια βάση δεδομένων γραφημάτων που έχει σχεδιαστεί για να διαχειρίζεται πολύπλοκες σχέσεις μεταξύ δεδομένων με αποδοτικό τρόπο. Η διαδικασία εισαγωγής δεδομένων

σε μια τέτοια βάση δεδομένων είναι περισσότερο απαιτητική από την εισαγωγή δεδομένων σε παραδοσιακές σχεσιακές βάσεις δεδομένων, καθώς πρέπει να διασφαλιστεί ότι όλες οι σχέσεις (relationships) και οι κόμβοι (nodes) είναι σωστά ευθυγραμμισμένοι και οργανωμένοι.

Παρακάτω απαριθμούνται τα βασικότερα σημεία ρύθμισης Neo4j:

- Εγκατάσταση:

Η εγκατάσταση της Neo4j μπορεί να γίνει τοπικά ή σε περιβάλλον cloud, ανάλογα με τις ανάγκες του έργου. Συνιστάται η χρήση της τελευταίας έκδοσης της Neo4j (όπως η 5.23) για να εκμεταλλευτείτε όλες τις σύγχρονες δυνατότητες και βελτιώσεις της πλατφόρμας. Κάθε νέα έκδοση παρέχει παραπάνω δυνατότητες ως προς τον τρόπο δημιουργίας των Local DBMS καθώς και βάσεων δεδομένων ενώ παράλληλα διευκολύνει αρκετά την εμπειρία του χρήστη με την Neo4j.

- Ρύθμιση παραμέτρων:

Η ρύθμιση της Neo4j περιλαμβάνει τη διαμόρφωση της βάσης δεδομένων για να υποστηρίξει μεγάλα σύνολα δεδομένων και σύνθετα queries. Αυτό μπορεί να περιλαμβάνει την αύξηση της διαθέσιμης μνήμης, την ενεργοποίηση caching και την προετοιμασία για την υποδοχή μεγάλων αρχείων CSV. Για την παραμετροποίηση του Neo4j πρέπει να εντοπιστεί αρχικά το Neo4j.conf αρχείο το οποίο είναι προσβάσιμο είτε ακολουθώντας την διαδρομή που οδηγεί στο φάκελο του DBMS και συγκεκριμένα εντός του φακέλου conf είτε μέσω της εφαρμογής Neo4j Desktop και του Local DBMS. Εντός του αρχείου συμπεριλαμβάνονται αρκετές εντολές σε μορφή σχολίων. Κάθε μια από αυτές μπορεί να έχει μεγάλη επίπτωση στις λειτουργίες και στην αποδοτικότητα της εφαρμογής. Επομένως για οποιαδήποτε εντολή εισαχθεί ή ενεργοποιηθεί απαιτεί προσοχή και γνώση των επιπτώσεων από τον χρήστη. Οι εντολές που χρησιμοποιήθηκαν για τους σκοπούς της εργασίας είναι η dbms.memory.heap.initial_size, η dbms.memory.heap.max_size, η dbms.memory.pagecache.size και η dbms.security.auth_enabled. Αρχικά χρησιμοποιήθηκε η dbms.memory.heap.initial_size η οποία καθορίζει το αρχικό μέγεθος της heap μνήμης που θα χρησιμοποιεί η Neo4j όταν ξεκινάει. Σκοπό της μνήμης είναι κυρίως η εκτέλεση ερωτημάτων και η αποθήκευση προσωρινών δεδομένων τόσο κατά τη διάρκεια της λειτουργίας της βάσης δεδομένων όσο και κατά την διαδικασία του import όπου δημιουργεί προσωρινά δεδομένα εντός του φακέλου temp που αναλύθηκε παραπάνω. Ουσιαστικά ο τρόπος με τον οποίο διαμορφώνει την Neo4j είναι κατά το ξεκίνημα όπου η JVM (Java Virtual Machine) δεσμεύει επιλεγμένα GB μνήμης για την heap memory. Έπειτα χρησιμοποιήθηκε η εντολή dbms.memory.heap.max_size η οποία καθορίζει το μέγιστο μέγεθος που μπορεί να λάβει η heap μνήμη της Neo4j κατά τη διαδικασία της εισαγωγής των δεδομένων. Είναι, δηλαδή, το ανώτατο όριο της μνήμης που μπορεί να χρησιμοποιηθεί από τη Neo4j για την εκτέλεση των διαδικασιών της. Αυτό σημαίνει πως αν το σύστημα χρειαστεί περισσότερη μνήμη κατά την εκτέλεση πολύπλοκων ερωτημάτων ή κατά τη διαχείριση πολλών δεδομένων, η heap μνήμη θα αυξηθεί έως και την τιμή που έχει οριστεί στο dbms.memory.heap.max_size. Απαραίτητη προϋπόθεση είναι ότι η τιμή της μνήμης που θα δοθεί κατά την έναρξη της διαδικασίας να μην απέχει πολύ από την μέγιστη τιμή που μπορεί να λάβει κατά την διάρκεια ώστε να αποφευχθούν οι απότομες αυξομειώσεις της χρήσης της μνήμης. Ακόμη πρέπει να ληφθεί υπόψιν ότι οι καθοριζόμενες τιμές που δίνονται στην heap μνήμη πρέπει να υποστηρίζονται από τον υπολογιστή, καθώς διαφορετικά η διαδικασία δημιουργίας της βάσης θα οδηγήσει σε bottleneck του

συστήματος και κατ' επέκταση στη διακοπή της διαδικασίας. Έπειτα η επόμενη εντολή που προστέθηκε είναι η `dbms.memory.pagecache.size`. Η ρύθμιση αυτή καθορίζει το μέγεθος της page cache, η οποία χρησιμοποιείται για την προσωρινή αποθήκευση των πιο συχνά χρησιμοποιούμενων δεδομένων του δίσκου στη μνήμη RAM, ώστε να βελτιωθεί η ταχύτητα των ερωτημάτων. Η page cache αφορά τους κόμβους, τις σχέσεις και τις ιδιότητες που χρησιμοποιούνται συχνά. Ορίζοντας για παράδειγμα την page cache σε 2 GB, τότε η Neo4j θα διατηρήσει τα πιο πρόσφατα και συχνά χρησιμοποιούμενα δεδομένα στη RAM για ταχύτερη πρόσβαση. Όσο μεγαλύτερο είναι το μέγεθος της page cache, τόσο περισσότερα δεδομένα μπορούν να αποθηκευτούν προσωρινά στη μνήμη αντί να διαβάζονται συνεχώς από τον δίσκο. Τέλος, μια από τις πιο σημαντικές εντολές ήταν η `dbms.security.auth_enabled`. Αυτή η λογική εντολή απενεργοποιεί τον μηχανισμό αυθεντικοποίησης της Neo4j. Από προεπιλογή, η Neo4j απαιτεί τη χρήση διαπιστευτηρίων κωδικών για την πρόσβαση στη βάση δεδομένων. Όταν η ρύθμιση αυτή είναι false, η Neo4j επιτρέπει την πρόσβαση σε όλους τους χρήστες χωρίς ταυτοποίηση. Αυτή η ρύθμιση είναι συχνά χρήσιμη σε περιβάλλοντα ανάπτυξης όταν η βάση δεδομένων προστατεύεται με άλλους μηχανισμούς ασφάλειας ή εξαιτίας καθημερινής χρήσης όπου η ανάγκη αυθεντικοποίησης καθυστερεί την εκκίνηση και την λειτουργία της βάσης.

Προετοιμασία Δεδομένων για την δημιουργία της βάσης δεδομένων

- Μετατροπή σε CSV:

Τα δεδομένα που έχουν φιλτραριστεί και αποσυμπίεστεί μέσω των διαφόρων scripts πρέπει να μετατραπούν σε ειδικής μορφής CSV το οποίο θα είναι συμβατό με το format που υποστηρίζεται από τη Neo4j. Αυτό πραγματοποιείται μέσω script, το οποίο χειρίζεται τη μετατροπή από JSON σε CSV, διασφαλίζοντας ότι όλα τα απαραίτητα πεδία είναι σωστά διαμορφωμένα για την εισαγωγή στη Neo4j.

- Κατανομή Δεδομένων:

Είναι σημαντικό τα δεδομένα να χωριστούν σωστά σε κόμβους και σχέσεις, καθώς η Neo4j βασίζεται σε αυτές τις δομές για την αποδοτική αναζήτηση και ανάλυση. Το script διευκολύνει αυτή τη διαδικασία διαχωρίζοντας τα δεδομένα σε διάφορα αρχεία CSV που αντιπροσωπεύουν τους κόμβους (όπως έργα και συγγραφείς) και τις σχέσεις τους (όπως η συγγραφή) καθώς και τις επικεφαλίδες τους. Κατά την εισαγωγή δεδομένων στη Neo4j μέσω της εντολής `neo4j-admin import`, τα header files (αρχεία κεφαλίδων) είναι απαραίτητα για τον καθορισμό της δομής και της μορφής των δεδομένων που περιέχονται στα CSV αρχεία. Η ανάγκη ύπαρξης των header files είναι ζωτικής σημασίας για την σωστή λειτουργία του `import`, καθώς υποδεικνύουν τις ονομασίες των στηλών στο CSV και τον τύπο κάθε στήλης (αν είναι ιδιότητα κόμβου, σχέση ή αναγνωριστικό). Αυτό βοηθά τη Neo4j να καταλάβει πώς να χειριστεί κάθε δεδομένο, δηλαδή αν πρόκειται για κόμβο, σχέση, ή ιδιότητα. Αναλυτικά τα header files συνεισφέρουν στον καθορισμό αναγνωριστικών ID's. Παραδειγματικά, στην περίπτωση κόμβων, το header file μπορεί να ορίσει ποια στήλη αντιπροσωπεύει το μοναδικό αναγνωριστικό (ID) για τον κάθε κόμβο. Αυτό είναι σημαντικό για να διασφαλιστεί ότι κάθε κόμβος ή σχέση αναγνωρίζεται με μοναδικό τρόπο. Επιπροσθέτως βοηθάει στην δομή δεδομένων καθορίζοντας ποια πεδία του CSV αντιπροσωπεύουν τα αναγνωριστικά των κόμβων (source και target nodes)

που συνδέονται από τη σχέση. Μπορεί επίσης να ορίσει ιδιότητες των σχέσεων, όπως το όνομα ή η την ημερομηνία της σχέσης. Περαιτέρω τα header files βοήθησαν σημαντικά στην διαδικασία του debugging. Όταν η Neo4j διαβάζει ένα CSV αρχείο με σωστά δομημένη κεφαλίδα, είναι ευκολότερο να εντοπίσει τυχόν σφάλματα εισαγωγής ή προβλήματα με τα δεδομένα, καθώς γνωρίζει τι είδους πληροφορίες αναμένονται σε κάθε στήλη. Μέσω του debugging των header files και των αρχείων logs που δημιουργούνται αυτόματα κατά τη διαδικασία εισαγωγής δεδομένων, δίνεται η δυνατότητα βελτιστοποίησης της διαδικασίας εισαγωγής.

Εισαγωγή στη Neo4j

Η εισαγωγή δεδομένων στη Neo4j γίνεται μέσω του εργαλείου neo4j-admin που παρέχει η πλατφόρμα. Το script αυτοματοποιεί αυτή τη διαδικασία, κατασκευάζοντας τις κατάλληλες εντολές εισαγωγής για τα δεδομένα που έχουν προετοιμαστεί. Αυτές οι εντολές καθορίζουν τη διαδρομή των αρχείων CSV, τον τύπο των δεδομένων που εισάγονται (κόμβοι ή σχέσεις), και τη βάση δεδομένων στην οποία θα γίνει η εισαγωγή. Η neo4j-admin εντολή, παρόλο που αποτελεί μια συνθέτη εντολή ως προς τον τρόπο λειτουργίας της, διαθέτει ένα μεγάλο σύνολο παραμέτρων οι οποίες είναι εξαιρετικά σημαντικές για την εύρυθμη λειτουργία του import καθώς και για το σχήμα της βάσης και των δεδομένων που θα εισαχθούν τελικά. Παρακάτω αναλύονται οι παράμετροι που χρησιμοποιήθηκαν, καθώς και η επιρροή της κάθε μιας στην διαδικασία εισαγωγής.

1. skip-duplicate-nodes:

Κατά την εισαγωγή η παράμετρος αυτή παραλείπει κόμβους που είναι διπλότυποι, δηλαδή κόμβους που έχουν το ίδιο αναγνωριστικό (ID). Κατά τη διάρκεια του import, η Neo4j εξετάζει κάθε κόμβο που προστίθεται και ελέγχει αν υπάρχει ήδη ένας κόμβος με το ίδιο: ID[73]. Αν εντοπίσει διπλός κόμβος, αυτός απλώς παραλείπεται και δεν προστίθεται στη βάση δεδομένων. Δεν επιτρέπεται η αντικατάσταση (overwrite) ή η ανανέωση των κόμβων με τα ίδια IDs, μόνο η παράλειψή τους.

2. skip-bad-relationships:

Η παράμετρος αυτή παρακάμπτει τις "κακές" σχέσεις, δηλαδή σχέσεις που περιέχουν σφάλματα ή είναι αδύνατο να εισαχθούν. Αυτό συνήθως συμβαίνει όταν μια σχέση αναφέρεται σε έναν κόμβο που δεν υπάρχει (ανύπαρκτα START_ID ή END_ID). Κατά την εισαγωγή, κάθε σχέση ελέγχεται για να διαπιστωθεί εάν και τα δύο άκρα της σχέσης (:START_ID και :END_ID) αναφέρονται σε υπαρκτούς κόμβους [73]. Αν κάποια από τα άκρα της σχέσης δεν αναφέρεται σε κόμβο που έχει ήδη εισαχθεί, η σχέση θεωρείται "bad" (κακή) και αγνοείται. Αυτό το φίλτρο διασφαλίζει ότι μόνο έγκυρες σχέσεις, οι οποίες συνδέουν υπαρκτούς κόμβους, θα εισαχθούν στη βάση δεδομένων διασφαλίζοντας έτσι την ακεραιότητα της βάσης.

3. ignore-empty-strings:

Κατά την εισαγωγή η παράμετρος αυτή αγνοεί τις κενές συμβολοσειρές στις ιδιότητες των κόμβων και των σχέσεων. Εάν κάποιο πεδίο σε ένα CSV αρχείο είναι κενό (""), η τιμή του δεν θα αποθηκευτεί στη βάση δεδομένων. Αν η Neo4j συναντήσει μια κενή τιμή για μια ιδιότητα (property) σε έναν κόμβο ή σε μια σχέση, αυτή η παράμετρος προκαλεί την

παράλειψη της τιμής. Το αποτέλεσμα είναι η μη δημιουργία ή ενημέρωση της ιδιότητας με κενή τιμή [73].

4. `overwrite-destination`:

Κατά την εισαγωγή η παράμετρος αυτή δίνει την επιλογή το `import` να αντικαθιστά τα υπάρχοντα δεδομένα στον προορισμό εισαγωγής (`destination`). Εάν υπάρχει ήδη μια βάση δεδομένων στον κατάλογο που προορίζεται για την εισαγωγή, αυτή η βάση θα διαγραφεί και θα αντικατασταθεί από τα νέα δεδομένα [73]. Πριν από την έναρξη της εισαγωγής, η Neo4j ελέγχει αν υπάρχει υπάρχουσα βάση δεδομένων στον φάκελο προορισμού. Αν βρεθεί, διαγράφεται το περιεχόμενό της και τα νέα δεδομένα που εισάγονται αντικαθιστούν πλήρως τα παλαιότερα δεδομένα. Αυτό είναι χρήσιμο όταν θέλουμε να ανανεώσουμε πλήρως τα δεδομένα και να μην διατηρηθούν παλαιότερα στοιχεία. Το παραπάνω αποδείχθηκε χρήσιμο κατά την διάρκεια του `debugging`.

5. `multiline-fields=true`:

Κατά την εισαγωγή η παράμετρος αυτή επιτρέπει την εισαγωγή πολυγραμμικών δεδομένων στα πεδία των CSV αρχείων, δηλαδή δεδομένα που εκτείνονται σε πολλές γραμμές [73]. Κατά την επεξεργασία των CSV αρχείων, η Neo4j αναγνωρίζει πολυγραμμικά δεδομένα και τα εισάγει ως ένα συνεχές πεδίο αντί να τα διαχωρίσει σε διαφορετικές εγγραφές ή να προκαλέσει σφάλμα. Αυτό είναι χρήσιμο για μεγάλες περιγραφές ή άλλες πληροφορίες που περιέχουν ενσωματωμένες αλλαγές γραμμής.

6. `bad-tolerance=10000000000000000`:

Κατά την εισαγωγή η παράμετρος αυτή επιτρέπει την παράκαμψη έως και ενός συγκεκριμένου αριθμού "κακών" εγγραφών (`bad records`) [73]. Το όριο αυτό είναι εξαιρετικά υψηλό, επιτρέποντας στη διαδικασία εισαγωγής να παρακάμψει τεράστιο αριθμό σφαλμάτων. Τα σφάλματα τα οποία παρακάμπτει δεν αλλοιώνουν την δομή ή την ακεραιότητα της βάσης. Ουσιαστικά αν η Neo4j εντοπίσει εγγραφές που περιέχουν σφάλματα (π.χ. προβληματικές σχέσεις ή κόμβους), αυτές οι εγγραφές παραλείπονται μέχρι να φτάσει το όριο των 1 δισεκατομμυρίων. Μόλις φτάσει αυτό το όριο, η διαδικασία εισαγωγής θα διακοπεί.

7. `max-off-heap-memory=90%'` :

Κατά την εισαγωγή η παράμετρος αυτή καθορίζει το μέγιστο ποσοστό της διαθέσιμης μνήμης του συστήματος που θα χρησιμοποιηθεί εκτός `heap` (`off-heap`), δηλαδή μνήμη που δεν διαχειρίζεται απευθείας η Java Virtual Machine (JVM). Ουσιαστικά η Neo4j διαχειρίζεται τις εισαγωγές δεδομένων χρησιμοποιώντας μνήμη εκτός `heap` για πιο αποδοτική επεξεργασία μεγάλων δεδομένων [73]. Η επιλογή αυτή περιορίζει την `off-heap` μνήμη στο 90%' της συνολικής μνήμης του συστήματος, αφήνοντας ένα μικρό ποσοστό για τη `heap` και άλλες λειτουργίες του συστήματος. Αυτή η εντολή και η τιμή της πρέπει να συνάδει και με τις τιμές που αναφέραμε παραπάνω σχετικά με το αρχείο `neo4j.conf`.

8. `threads=2`:

Κατά την εισαγωγή η παράμετρος αυτή καθορίζει πόσα νήματα (`threads`) θα χρησιμοποιηθούν κατά τη διάρκεια του `import`. Με την τιμή 2, η διαδικασία θα εκτελείται σε δύο παράλληλες ροές (`threads`) [73]. Η Neo4j χρησιμοποιεί δύο νήματα ταυτόχρονα για

την εισαγωγή δεδομένων, επιτρέποντας τη διαχείριση περισσότερων δεδομένων ανά μονάδα χρόνου και βελτιώνοντας την ταχύτητα εισαγωγής. Αυτή η παράμετρος επιτρέπει την αύξηση της απόδοσης σε συστήματα με περισσότερους πυρήνες. Στην προκειμένη περίπτωση το σύστημα που χρησιμοποιήθηκε διέθετε 2 πυρήνες, εξού και η τιμή της μεταβλητής.

9. `ignore-extra-columns=true`:

Κατά την εισαγωγή η παράμετρος παραλείπει τις επιπλέον στήλες που μπορεί να υπάρχουν στα αρχεία CSV και δεν καθορίζονται από το header file. Δηλαδή, εάν ένα CSV αρχείο περιέχει περισσότερες στήλες από αυτές που ορίζονται στην κεφαλίδα, αυτές οι στήλες αγνοούνται και δεν εισάγονται στη βάση δεδομένων. Επομένως η Neo4j εξετάζει τις στήλες του CSV αρχείου και τις αντιστοιχίζει με τις στήλες που ορίζονται στο header file. Αν συναντήσει επιπλέον στήλες που δεν αναφέρονται στο header, η παράμετρος – `ignore-extra-columns=true` προκαλεί την παράλειψή τους, αντί να δημιουργήσει σφάλμα ή να διακόψει την εισαγωγή. Αυτό είναι ιδιαίτερα χρήσιμο όταν τα αρχεία CSV περιέχουν επιπλέον πληροφορίες που δεν χρειάζονται για τη βάση δεδομένων ή που μπορεί να προέρχονται από άλλες διαδικασίες ή όταν ο όγκος των δεδομένων που εισάγεται είναι αρκετά μεγάλος και είναι αδύνατη η μελέτη της πληροφορίας.

10. `auto-skip-subsequent-headers=true`:

Κατά την εισαγωγή η παράμετρος αυτή παραλείπει αυτόματα τις επόμενες κεφαλίδες (headers) που βρίσκονται σε αρχεία CSV μετά την πρώτη γραμμή. Ουσιαστικά αν η Neo4j συναντήσει επιπλέον headers μετά την πρώτη γραμμή, αυτές οι κεφαλίδες παραλείπονται αυτόματα. Αυτό επιτρέπει τη συνέχιση της εισαγωγής χωρίς διακοπή, αποφεύγοντας σφάλματα λόγω πολλαπλών headers στο ίδιο αρχείο.

11. `high-parallel-io=auto`:

Κατά την εισαγωγή η παράμετρος αυτή επιτρέπει τη χρήση υψηλής παράλληλης εισαγωγής/εξαγωγής δεδομένων (parallel I/O) και προσαρμόζεται αυτόματα στις δυνατότητες του συστήματος. Η Neo4j προσαρμόζεται αυτόματα για να χρησιμοποιήσει τη μέγιστη διαθέσιμη δυνατότητα εισαγωγής/εξαγωγής δεδομένων στο σύστημα, βελτιστοποιώντας την ταχύτητα της εισαγωγής [73]. Ανάλογα με το σύστημα, χρησιμοποιεί πολλαπλές ροές για την ανάγνωση και εγγραφή δεδομένων, βελτιώνοντας την απόδοση.

12. Διασφάλιση Ακεραιότητας Δεδομένων:

Κατά την εισαγωγή, είναι κρίσιμο να διασφαλιστεί ότι όλα τα δεδομένα εισάγονται σωστά χωρίς απώλειες. Αυτό περιλαμβάνει τη διασφάλιση ότι οι σχέσεις μεταξύ των κόμβων είναι πλήρεις και ότι δεν υπάρχουν προβλήματα στη δομή των δεδομένων που θα μπορούσαν να προκαλέσουν σφάλματα στα queries. Χαρακτηριστικό παράδειγμα τέτοιων προβλημάτων είναι οι ασυμβατότητες στη δομή των δεδομένων. Το dataset της OpenAlex είναι μεγάλο και σύνθετο, με δεδομένα που αφορούν επιστημονικές δημοσιεύσεις, συγγραφείς, θεματικές κατηγορίες και άλλα στοιχεία. Καθώς αυτά τα δεδομένα διασυνδέονται, μπορεί να υπάρχουν περίπλοκες σχέσεις και τύποι δεδομένων που ενδέχεται να μην είναι άμεσα συμβατοί με τη Neo4j. Για παράδειγμα, αν τα δεδομένα περιλαμβάνουν ασαφείς σχέσεις ή αναντιστοιχίες μεταξύ αναγνωριστικών (IDs), μπορεί να δημιουργηθούν λάθη κατά την εισαγωγή, οδηγώντας σε ελλιπή ή λανθασμένα

δεδομένα στους κόμβους και τις σχέσεις. Εκτός από πιθανές αναντιστοιχίες μπορεί να δημιουργηθούν θέματα κατά την εισαγωγή των δεδομένων τα οποία η Neo4j μπορεί να μην τα θεωρήσει “καθαρά”. Η OpenAlex είναι μια μεγάλη πλατφόρμα που συγκεντρώνει δεδομένα από διάφορες πηγές. Είναι λογικό επομένως ότι κάποια από αυτά τα δεδομένα θα περιέχουν σφάλματα ή να μην είναι πλήρως καθαρά. Ακόμη ασυνέπειες, ελλείψεις ή διπλοεγγραφές μπορούν να προκαλέσουν ζητήματα κατά την εισαγωγή στη Neo4j. Για παράδειγμα, αν υπάρχουν κενά δεδομένα στα πεδία που θεωρούνται απαραίτητα (π.χ.IDs), η Neo4j μπορεί να μην καταφέρει να ολοκληρώσει την εισαγωγή ή να δημιουργηθούν κόμβοι και σχέσεις χωρίς ουσία. Αυτό οδηγεί στην άσκοπη χρήση του αποθηκευτικού χώρου με δεδομένα τα οποία δεν αποφέρουν καμία αναλυτική χρησιμότητα. Τέλος είναι πολύ σημαντικό να επισημανθεί ότι τέτοια είδους δεδομένων και η ύπαρξη τους σε πολλά σημεία της βάσης παράλληλα με την πολυπλοκότητα των ιδιοτήτων των entities τους, έκανε το formatting των csv και κυρίως των headers μια δύσκολη διαδικασία. Απαιτείται λοιπόν ακριβής ορισμός των ιδιοτήτων που πρέπει να διαβάσει η Neo4j, τόσο σε type όσο και σε αντιστοιχία.

Βελτιστοποίηση της Βάσης Δεδομένων

- **Indexing:** Μετά την εισαγωγή των δεδομένων, η δημιουργία ευρετηρίων (indexes) στις πιο σημαντικές ιδιότητες των κόμβων μπορεί να βελτιώσει δραστικά την απόδοση των queries στη Neo4j. Τα ευρετήρια επιτρέπουν την ταχύτερη αναζήτηση δεδομένων και είναι ένα από τα κύρια βήματα βελτιστοποίησης της βάσης δεδομένων μετά την αρχική εισαγωγή.
- **Ανασυγκρότηση της Βάσης Δεδομένων:** Η Neo4j περιλαμβάνει εργαλεία για την ανασυγκρότηση της βάσης δεδομένων μετά την εισαγωγή μεγάλων όγκων δεδομένων. Αυτό βοηθά στη βελτιστοποίηση της χρήσης του δίσκου και στην αποφυγή κατακερματισμού, διασφαλίζοντας τη μακροπρόθεσμη απόδοση της βάσης δεδομένων.

Με τα παραπάνω βήματα και παραμέτρους ολοκληρώνεται η ρύθμιση και προετοιμασία του συστήματος για την επεξεργασία και εισαγωγή δεδομένων από το OpenAlex στη Neo4j. Η σωστή υλοποίηση όλων των παραπάνω διασφαλίζει ότι η εφαρμογή θα λειτουργεί αποδοτικά, ακόμη και όταν διαχειρίζεται μεγάλα και πολύπλοκα σύνολα δεδομένων. Στα επόμενα κεφάλαια θα αναλυθούν περαιτέρω οι τεχνικές λεπτομέρειες των επιμέρους scripts και η λογική πίσω από τη σχεδίαση τους.

5.4 Ανάκτηση Δεδομένων από το OpenAlex μέσω AWS CLI

Η ανάκτηση των δεδομένων από το OpenAlex αποτελεί το πρώτο βήμα στο data pipeline της εφαρμογής. Σε αυτή την ενότητα θα αναλύσουμε τις εντολές AWS S3 που χρησιμοποιούνται για τη λήψη των δεδομένων και θα εξηγήσουμε τη διαδικασία φιλτραρίσματος με χρήση AWS CLI.

5.4.1 Περιγραφή των Εντολών AWS S3 για τη Λήψη Δεδομένων

Η διαδικασία ξεκινά με τη χρήση εντολών aws s3 για την ανάκτηση δεδομένων από το OpenAlex, το οποίο φιλοξενεί τα δεδομένα του σε μορφή gzip (.gz) στο Amazon S3. Αυτές οι εντολές εκτελούνται μέσα από το script και επιτρέπουν την περιήγηση και λήψη αρχείων από συγκεκριμένα directories του OpenAlex.

Εντολές aws s3 ls και aws s3 sync

1. aws s3 ls: Αυτή η εντολή χρησιμοποιείται για να εμφανιστούν τα περιεχόμενα ενός bucket ή directory στο S3. Για παράδειγμα, για να εντοπίσουμε τα δεδομένα των authors που ενημερώθηκαν σε μια συγκεκριμένη ημερομηνία, χρησιμοποιούμε την εντολή:

```
aws s3 ls "s3://openalex/data/author"
--no-sign-request --region us-ea
```

Η εντολή αυτή θα επιστρέψει μια λίστα με τα διαθέσιμα αρχεία, μαζί με το μέγεθός τους και την ημερομηνία τελευταίας ενημέρωσης. Αυτή η πληροφορία είναι κρίσιμη για τον προσδιορισμό των αρχείων που θα ληφθούν.

2. aws s3 sync: Η εντολή αυτή χρησιμοποιείται για τον συγχρονισμό δεδομένων μεταξύ του S3 και ενός τοπικού φακέλου. Για παράδειγμα, για να κατεβάσουμε τα δεδομένα των έργων που ενημερώθηκαν σε μια συγκεκριμένη ημερομηνία, χρησιμοποιούμε:

```
aws s3 sync "s3://openalex/data/works/updated\_date=2024-02-21"
```

Αυτή η εντολή θα κατεβάσει όλα τα αρχεία από το συγκεκριμένο directory στο S3 στον τοπικό φάκελο [74].

Οι εντολές αυτές ενσωματώθηκαν σε script, το οποίο διαχειρίστηκε την αυτοματοποίηση αυτών των λήψεων και διασφάλισε ότι τα σωστά αρχεία κατέβηκαν με βάση τις απαιτήσεις του project.

5.4.2 Επεξήγηση της Διαδικασίας Φιλτραρίσματος με Χρήση AWS CLI

Μετά τη λήψη των δεδομένων, η διαδικασία φιλτραρίσματος είναι κρίσιμη για να περιοριστούν τα δεδομένα μόνο σε αυτά που είναι απαραίτητα για την εφαρμογή. Αυτό γίνεται με τη βοήθεια του AWS CLI και των εντολών φιλτραρίσματος που περιλαμβάνονται στο script.

5.4.3 Επισκόπηση της Αποσυμπίεσης και Μορφοποίησης Δεδομένων

Η προ επεξεργασία των δεδομένων αποτελεί ένα καθοριστικό στάδιο στην αλυσίδα επεξεργασίας, από τη λήψη δεδομένων από το OpenAlex μέχρι την εισαγωγή τους στη Neo4j. Αυτό το στάδιο περιλαμβάνει την αποσυμπίεση των δεδομένων, τη φιλτράρισμά τους με συγκεκριμένα κριτήρια, και τη μορφοποίησή τους σε κατάλληλη μορφή για τη βάση δεδομένων Neo4j.

Η διαδικασία αρχίζει με την αποσυμπίεση των αρχείων GZIP (.gz) που λαμβάνονται από το

AWS S3. Το script `unzipper.py` αναλαμβάνει την αποσυμπίεση αυτών των αρχείων, χρησιμοποιώντας τη βιβλιοθήκη `gzip` της Python την οποία ανέλυσα παραπάνω. Στόχος είναι να προετοιμαστούν τα δεδομένα για την επόμενη φάση της διαδικασίας, η οποία είναι το φιλτράρισμα. Ακολουθεί η φάση φιλτραρίσματος. Το script αυτό επέλεξε συγκεκριμένα "keys" από τα JSON αρχεία, τα οποία είναι κρίσιμα για την εφαρμογή. Τα επιλεγμένα δεδομένα σχετίζονται κυρίως με `relationships` μεταξύ των διαφόρων οντοτήτων της OpenAlex, όπως οι σχέσεις μεταξύ `authors` και `works`. Η λογική πίσω από την επιλογή αυτών των δεδομένων είναι να διασφαλιστεί ότι αν κάποια δεδομένα λείπουν από μία θεματική κατηγορία, αυτά μπορεί να υπάρχουν σε κάποια άλλη, διατηρώντας έτσι τα κρίσιμα στοιχεία για τη δημιουργία σχέσεων. Τα δεδομένα που δεν οδηγούν σε `relationships` ή είναι κενά απορρίπτονται, ενώ τα φιλτραρισμένα δεδομένα παραμένουν σε συμπίεσμένη μορφή για να αντιμετωπιστούν τα προβλήματα χώρου αποθήκευσης. Το τελικό στάδιο της προ επεξεργασίας περιλαμβάνει τη μετατροπή των φιλτραρισμένων δεδομένων σε μορφή CSV μέσω του κώδικα. Αυτή η μετατροπή εξασφαλίζει ότι τα δεδομένα είναι κατάλληλα μορφοποιημένα για την εισαγωγή τους στη βάση δεδομένων Neo4j, διασφαλίζοντας την ακρίβεια και την αποδοτικότητα της διαδικασίας εισαγωγής.

5.4.4 Σημασία της Προεπεξεργασίας στο Data Pipeline

Η προεπεξεργασία των δεδομένων είναι κρίσιμη για την επιτυχή λειτουργία του data pipeline. Η διαδικασία αυτή διασφαλίζει ότι μόνο τα απαραίτητα δεδομένα θα προχωρήσουν στο επόμενο στάδιο, μειώνοντας έτσι τον όγκο των δεδομένων που πρέπει να επεξεργαστεί η Neo4j και βελτιώνοντας την απόδοσή της. Επιπλέον, η σωστή μορφοποίηση των δεδομένων σε CSV επιτρέπει την εύκολη και αποδοτική εισαγωγή στη βάση δεδομένων, διευκολύνοντας την ανάλυση και τη διαχείριση των δεδομένων στη συνέχεια.

Η χρήση συμπίεσμένων και φιλτραρισμένων δεδομένων βοηθά επίσης στη διαχείριση του αποθηκευτικού χώρου, μειώνοντας την ανάγκη για μεγάλους πόρους και διατηρώντας τα κόστη σε λογικά επίπεδα. Αυτή η προσέγγιση εξασφαλίζει ότι τα δεδομένα που τελικά αποθηκεύονται στη Neo4j είναι τα πιο χρήσιμα και αποδοτικά, βελτιώνοντας την ποιότητα και την ακρίβεια των αναλύσεων που μπορούν να πραγματοποιηθούν πάνω σε αυτά.

5.5 Εισαγωγή και Διαχείριση Δεδομένων στη Neo4j

5.5.1 Τρόποι εισαγωγής δεδομένων στην Neo4j

Η διαδικασία εισαγωγής των δεδομένων στη Neo4j μπορεί να πραγματοποιηθεί με δυο τρόπους. Την αρχική ολική εισαγωγή (`full import`) και τις μεταγενέστερες ανανεώσεις δεδομένων (`incremental updates`).

Ολική Εισαγωγή (Full Import)

Για την αρχική εισαγωγή των δεδομένων χρησιμοποιήθηκε η εντολή `neo4j-admin import full`, η οποία μου επιτρέπει την εισαγωγή μεγάλων συνόλων δεδομένων σε μία κίνηση. Η εντολή αυτή απαιτεί τα δεδομένα να είναι προετοιμασμένα σε μορφή CSV, κάτι που επιτεύχθηκε μέσω των scripts επεξεργασίας που αναπτύχθηκαν. Το `neo4j-admin import full` χρησιμοποιήθηκε για να

εισάγει δεδομένα από το OpenAlex, συμπεριλαμβανομένων των σχέσεων μεταξύ των οντοτήτων, όπως ενδεικτικά authors, works, και institutions.

```
def construct\_import\_command(
parent\_folder: Path,
headers, import\_parent,
file\_type="csv",
label\_filter=None,
part\_filter=None,
date\_filter=None,
print\_command=False,
debug=False,):
\# Construct the import command import\_command = [
f"\{import\_parent\}/bin/neo4j-admin", "database", "import",
"full",
"--skip-duplicate-nodes",
"--skip-bad-relationships",
"--ignore-empty-strings",
"--overwrite-destination",
"--multiline-fields=true",
"--bad-tolerance=10000000000000000", "--
max-off-heap-memory=90\%",
"--threads=2",
"--ignore-extra-columns=true",
"--auto-skip-subsequent-headers=true", "-
-high-parallel-io=auto",]
```

Ανανεώσεις Δεδομένων (Incremental Updates)

Μετά την αρχική εισαγωγή, είναι δυνατή η ανανέωση της βάσης μέσω της εντολής neo4j-admin import incremental. Αυτή η προσέγγιση επιτρέπει να εισαχθούν νέες καταχωρήσεις ή να ενημερώνουμε υπάρχουσες, χωρίς να απαιτείται η πλήρης αναδόμηση της βάσης δεδομένων. Η χρήση αυτής της εντολής εξασφαλίζει ότι η βάση δεδομένων παραμένει ενημερωμένη με τις πιο πρόσφατες πληροφορίες από το OpenAlex, ενώ παράλληλα μειώνεται ο χρόνος και οι πόροι που απαιτούνται για την εισαγωγή δεδομένων.

```
neo4j-admin database import incremental [-h]
[--expand-commands]--force [--verbose]
[--auto-skip-subsequent-headers [=true|false]]
[--ignore-empty-strings [=true|false]]
[--ignore-extra-column [=true|false]]
[--legacy-style-quoting [=true|false]]
[--multiline-fields [=true|false]]
[--normalize-types [=true|false]]
[--skip-bad-entries-logging [=true|false]]
[--skip-bad-relationships [= --skip-duplicate-nodes
```

```
[=true|false]]
[--strict[=true|false]]
[--trim-strings[=true|false]]
[--additional-config=<file>]
[--array-delimiter=<char>]
[--bad-tolerance=<num>]
[--delimiter=<char>]
[--high-parallel-io=on|off|auto]
[--id-type=string|integer|actual]
[--input-encoding=<character-set>]
[--max-off-heap-memory=<size>]
[--quote=<char>]
[--read-buffer-size=<size>]
[--report-file=<path>]
[--stage=all|prepare|build|merge]
[--threads=<num>]
--nodes=[<label>[:<label>]...=]
<files>... [--nodes=[<label>[:<label>]...=]
<files>...]]... [--relationships=[<type>=]<files>...]]
... <database>
```

Αυτή η στρατηγική διασφαλίζει ότι η βάση δεδομένων μας είναι πάντα ενημερωμένη και έτοιμη για την εκτέλεση σύνθετων ερωτημάτων και αναλύσεων. Η εστίαση μας στην αποδοτική διαχείριση δεδομένων μέσω της Neo4j εξυπηρετεί τον κύριο σκοπό του έργου, που είναι η ανάλυση μεγάλων ακαδημαϊκών δεδομένων με ευελιξία και ακρίβεια.

5.5.2 Ανάλυση του script

Το script `awsdownloader.py` είναι ένα κρίσιμο κομμάτι του συστήματος που αναπτύχθηκε για τη διαχείριση και λήψη δεδομένων από το OpenAlex μέσω AWS S3. Σε αυτό το υποκεφάλαιο, θα αναλύσουμε λεπτομερώς τη λειτουργία του script, εξετάζοντας τη δομή του, τις βασικές του συναρτήσεις, και τη ροή της διαδικασίας λήψης δεδομένων.

Λεπτομερής Ανάλυση του Script

Το `awsdownloader.py` έχει ως κύριο στόχο τη λήψη και επεξεργασία μεγάλων όγκων δεδομένων από το OpenAlex. Η διαδικασία αυτή περιλαμβάνει την καταγραφή των διαθέσιμων δεδομένων, την επιλογή των κατάλληλων ημερομηνιών, και την τελική λήψη και αποθήκευση των δεδομένων σε τοπικό σύστημα. Αρχικά, το script ορίζει ορισμένες σταθερές και μεταβλητές που χρησιμοποιούνται για τη διαχείριση των διαδρομών των αρχείων, τις επαναληπτικές προσπάθειες λήψης (retries), και τα αρχεία εξόδου (output files) που καταγράφουν τα αποτελέσματα των εντολών AWS S3.

Το `awsdownloader.py` ξεκινά με τη φόρτωση των απαραίτητων βιβλιοθηκών, όπως το `subprocess` για την εκτέλεση εντολών στο shell, και το `defaultdict` από το `collections` για την αποθήκευση των ημερομηνιών κατά θεματική κατηγορία. Ένα από τα πρώτα βήματα που εκτελεί το script είναι

η μετάφραση της διαδρομής του τοπικού συστήματος (local base path) ανάλογα με το λειτουργικό σύστημα που χρησιμοποιείται, είτε Windows είτε Unix-like. Αυτό επιτυγχάνεται μέσω της συνάρτησης `translate_path(path_str)`. Αυτό αποδείχθηκε ιδιαίτερα χρήσιμο διότι το testing και το debugging καθώς και το download χρειάστηκε να γίνει σε περιβάλλοντα Windows(τοπικός υπολογιστής) αλλά και Linux(απομακρυσμένο μηχάνημα).

Στη συνέχεια, το script εκτελεί μια σειρά από εντολές `aws s3 ls` για την καταγραφή των αρχείων που βρίσκονται στα S3 buckets του OpenAlex. Οι εντολές αυτές εκτελούνται μέσω της συνάρτησης `run_command(command)`, η οποία επιλέγει το κατάλληλο shell για την εκτέλεση των εντολών ανάλογα με το λειτουργικό σύστημα.

Μια από τις σημαντικότερες λειτουργίες του script είναι η `list_s3_files(commands, output_file, output_file_dates)`, η οποία καταγράφει τα διαθέσιμα αρχεία από το S3 και αποθηκεύει τις ημερομηνίες των αρχείων ώστε να γίνει η επιλογή της πιο επικαιροποιημένης μορφής της βάσης μεταξύ των διαφόρων θεματικών κατηγοριών του OpenAlex (όπως authors, works, institutions κ.λπ.) ώστε να αποφευχθούν τα duplicate data.

Μετά την επιλογή των ημερομηνιών, το script προχωρά στη λήψη των δεδομένων μέσω της συνάρτησης `download_with_retries(s3_path, local_path, retries)`, η οποία επιχειρεί τη λήψη των αρχείων με πολλαπλές προσπάθειες σε περίπτωση αποτυχίας.

Επεξήγηση Συναρτήσεων

Ο ψευδοκώδικας που περιγράφει τη ροή του `awsdownloader.py` μπορεί να συνοψιστεί ως εξής:

1. Ορισμός σταθερών και μεταβλητών για τις διαδρομές αρχείων και τις παραμέτρους λήψης.
2. Μετάφραση της διαδρομής του τοπικού συστήματος ανάλογα με το λειτουργικό σύστημα.
3. Εκτέλεση εντολών `aws s3 ls` για την καταγραφή των διαθέσιμων αρχείων σε S3.
4. Ανάλυση των αποτελεσμάτων και επιλογή των νέων ημερομηνιών μεταξύ των θεματικών κατηγοριών.
5. Λήψη των δεδομένων από το S3 με επαναληπτικές προσπάθειες.

Κάθε μία από τις παραπάνω διαδικασίες αντιστοιχεί σε μία ή περισσότερες συναρτήσεις του script.

- `translate_path(path_str)`: Μετατρέπει τη διαδρομή του τοπικού συστήματος σε μορφή συμβατή με το λειτουργικό σύστημα (Windows ή Unix-like).
- `run_command(command)`: Εκτελεί μια εντολή shell και επιστρέφει το αποτέλεσμα. Χρησιμοποιεί την κατάλληλη διεπαφή shell ανάλογα με το λειτουργικό σύστημα.
- `list_s3_files(commands, output_file, output_file_dates)`: Εκτελεί τις εντολές `aws s3 ls`, καταγράφει τα διαθέσιμα αρχεία, και επιλέγει τις ημερομηνίες για λήψη δεδομένων.
- `download_with_retries(s3_path, local_path, retries)`: Επιχειρεί να κατεβάσει ένα αρχείο από το S3, με δυνατότητα πολλαπλών επαναλήψεων σε περίπτωση αποτυχίας.

Ροή της Διαδικασίας Λήψης

Η διαδικασία λήψης των δεδομένων μέσω του `awsdownloader.py` ακολουθεί μια συστηματική ροή που ξεκινά με την καταγραφή των διαθέσιμων αρχείων σε S3 και καταλήγει στην αποθήκευση των αρχείων στο τοπικό σύστημα.

Αρχικά, το script αναζητά αρχεία σε S3 για τις διάφορες θεματικές κατηγορίες του OpenAlex. Τα αρχεία αυτά καταγράφονται και οι ημερομηνίες ενημέρωσης (`updated_date`) αποθηκεύονται. Στη συνέχεια, το `awsdownloader.py` επιλέγει τις ημερομηνίες που είναι οι πιο ανανεωμένες για κάθε part μεταξύ όλων των θεματικών κατηγοριών, εξασφαλίζοντας ότι τα δεδομένα που θα ληφθούν είναι συνεπή και πλήρη.

Αφού επιλεγθούν οι ημερομηνίες, το script προχωρά στη λήψη των αρχείων από το S3. Η διαδικασία λήψης είναι σχεδιασμένη ώστε να είναι ανθεκτική σε αποτυχίες, με πολλαπλές επαναλήψεις (`retries`) σε περίπτωση που η λήψη αποτύχει. Το `awsdownloader.py` παρέχει επίσης τη δυνατότητα καταγραφής της προόδου της λήψης, αποθηκεύοντας πληροφορίες για τα αρχεία που έχουν ήδη ληφθεί, αποφεύγοντας έτσι διπλές λήψεις.

Μετά την ολοκλήρωση της λήψης, τα δεδομένα είναι έτοιμα για αποσυμπίεση και περαιτέρω επεξεργασία μέσω άλλων scripts, όπως το `unzipper.py` και το `json'_scraper.py`, προκειμένου να μετατραπούν τελικά σε μορφή κατάλληλη για εισαγωγή στη βάση Neo4j. Τα συγκεκριμένα script θα αναλυθούν στην συνέχεια.

Αποσυμπίεση και Διαμόρφωση των Δεδομένων

Το `unzipper.py` είναι υπεύθυνο για την αποσυμπίεση των αρχείων που έχουν ληφθεί από το OpenAlex. Τα δεδομένα αρχικά έρχονται σε συμπιεσμένη μορφή `.gz`, γεγονός που καθιστά απαραίτητη την αποσυμπίεση τους για να μπορούν να φιλτραριστούν και να προετοιμαστούν για εισαγωγή. Το μέγεθος των αρχείων αυτών ποικίλει, με ορισμένα αρχεία να είναι αρκετά μικρά, ενώ άλλα μπορεί να ξεπερνούν τα 2 GB, όπως φαίνεται στα καταγεγραμμένα `outputs`. Η κύρια λειτουργία του script είναι η `decompress_and_format_json`, η οποία αναλαμβάνει την αποσυμπίεση και τη μετατροπή των αρχείων από τη μορφή JSON στην κατάλληλη μορφή για φιλτράρισμα. Το script εκτελεί την αποσυμπίεση, δημιουργεί νέα αρχεία με τις αποσυμπιεσμένες πληροφορίες, και τα προετοιμάζει για το επόμενο στάδιο, το φιλτράρισμα.

Ψευδοκώδικας για αποσυμπίεση και μορφή των αρχείων json:

```
def decompress\_and\_format\_json(input\_path,
output\_path): with gzip.open(input\_path, 'rb')
as f\_in:
with open(output\_path, 'wb')
as f\_out: shutil.copyfileobj(f\_in, f\_out)
```

Αυτό το παράδειγμα δείχνει την αποσυμπίεση ενός αρχείου `.gz` σε ένα απλό JSON αρχείο. Η διαδικασία αυτή είναι απαραίτητη, δεδομένου ότι τα δεδομένα σε μορφή συμπιεσμένης είναι αδύνατο να φιλτραριστούν και να επεξεργαστούν άμεσα. Επίσης, λόγω του μεγάλου όγκου των δεδομένων, η αποσυμπίεση τους είναι ένα βήμα που απαιτεί σημαντικούς πόρους αποθήκευσης, ιδίως όταν πρόκειται να γίνει τοπικά.

Φιλτράρισμα Δεδομένων

Μετά την αποσυμπίεση, το `json_scrapper.py` αναλαμβάνει να φιλτράρει τα δεδομένα ώστε να διατηρηθούν μόνο τα σημαντικά στοιχεία που σχετίζονται με τις σχέσεις (relationships) μεταξύ των διαφορετικών οντοτήτων της OpenAlex, όπως οι σχέσεις μεταξύ `authors` και `works`. Η αναγκαιότητα αυτής της διαδικασίας αποδεικνύεται από τον τεράστιο όγκο των δεδομένων, όπου η εισαγωγή όλων των αρχικών δεδομένων θα απαιτούσε αδικαιολόγητα μεγάλους πόρους και θα δυσχέραινε την απόδοση της βάσης δεδομένων.

Η σημαντικότερη λειτουργία στο `json_scrapper.py` είναι η `filter_keys`, η οποία επιλέγει συγκεκριμένα "keys" από τα JSON αρχεία, βασισμένα στην ανάλυση των δεδομένων που είναι κρίσιμα για την εφαρμογή. Για παράδειγμα, τα δεδομένα που σχετίζονται με τις σχέσεις μεταξύ των `authors` και των `works` επιλέγονται επειδή αποτελούν τον πυρήνα της πληροφορίας που θα αναλυθεί στη Neo4j.

Η επιλογή των δεδομένων που διατηρούνται γίνεται με στρατηγικό τρόπο. Ο στόχος είναι να διασφαλιστεί ότι ακόμη και αν κάποιο σημαντικό δεδομένο λείπει από μια θεματική κατηγορία, αυτό μπορεί να βρίσκεται σε κάποια άλλη, εξασφαλίζοντας έτσι την πληρότητα της βάσης δεδομένων. Τα δεδομένα που δε συμβάλλουν στη δημιουργία relationships ή που είναι κενά απορρίπτονται. Επιπλέον, για να αντιμετωπιστούν τα προβλήματα χώρου αποθήκευσης, τα φιλτραρισμένα δεδομένα παραμένουν σε συμπίεσμένη μορφή.

Ψευδοκώδικας για αποσυμπίεση αρχείων:

```
Def filter\_keys(input\_json,
important\_keys): filtered\_data = {}
for key in important\_keys:
if key in input\_json:
filtered\_data[key] = input\_json[key]
return filtered\_data
```

Η λειτουργία αυτή διασφαλίζει ότι τα αρχεία JSON που προκύπτουν από το φιλτράρισμα περιέχουν μόνο τα απαραίτητα στοιχεία, μειώνοντας έτσι σημαντικά τον όγκο των δεδομένων που θα εισαχθούν στη βάση.

Ροή Δεδομένων κατά την Αποσυμπίεση και το Φιλτράρισμα

Η συνολική ροή δεδομένων κατά τη διαδικασία αποσυμπίεσης και φιλτραρίσματος περιλαμβάνει τα εξής βήματα:

1. Λήψη Δεδομένων: Τα δεδομένα λαμβάνονται από το AWS S3 σε συμπίεσμένη μορφή (.gz).
2. Αποσυμπίεση: Τα αρχεία αποσυμπίεζονται από το `unzipper.py` και αποθηκεύονται σε νέα αρχεία JSON.
3. Φιλτράρισμα: Τα αποσυμπίεσμένα αρχεία περνούν από το `json_scrapper.py`, όπου φιλτράρονται τα περιττά δεδομένα και διατηρούνται μόνο τα στοιχεία που είναι απαραίτητα για τις σχέσεις στο Neo4j.

4. Επανασυμπίεση (αν απαιτείται): Σε κάποιες περιπτώσεις, τα φιλτραρισμένα δεδομένα ενδέχεται να συμπίεστούν ξανά για να εξοικονομηθεί χώρος αποθήκευσης πριν από την εισαγωγή τους στη Neo4j.

Η διαδικασία αυτή έχει ως αποτέλεσμα τη δημιουργία ενός συνόλου δεδομένων που είναι βελτιστοποιημένο για την εισαγωγή και επεξεργασία στο Neo4j. Με τη χρήση αυτών των scripts, η διαχείριση μεγάλων όγκων δεδομένων καθίσταται αποτελεσματική και αποδοτική, εξασφαλίζοντας ότι η βάση δεδομένων περιέχει μόνο τις πληροφορίες που είναι ουσιαστικά σημαντικές για την ανάλυση και τη διαχείριση των σχέσεων μεταξύ των διαφορετικών οντοτήτων.

5.5.3 Μετατροπή από JSON σε CSZ Script (json2csv_neo4j.py)

Το script json2csv_neo4j.py παίζει κρίσιμο ρόλο στη διαδικασία μετατροπής των δεδομένων από τη μορφή JSON στη μορφή CSV, προκειμένου να εισαχθούν στη βάση δεδομένων Neo4j. Η συγκεκριμένη διαδικασία είναι απαραίτητη λόγω των απαιτήσεων της Neo4j, η οποία χρησιμοποιεί τη μορφή CSV για την εισαγωγή δεδομένων, ειδικά όταν πρόκειται για μεγάλα σύνολα δεδομένων όπως αυτά που διαχειρίζεται η OpenAlex.

Οι βασικές λειτουργίες του script περιλαμβάνουν την ανάγνωση των JSON αρχείων, την εξαγωγή των σχετικών πληροφοριών, και τη μορφοποίηση αυτών των πληροφοριών σε δομή CSV, έτσι ώστε να είναι συμβατά με τις απαιτήσεις του Neo4j. Το script αυτό χειρίζεται τα δεδομένα με τέτοιο τρόπο ώστε να διατηρούνται οι απαραίτητες σχέσεις μεταξύ των οντοτήτων (π.χ. authors, works, institutions) και να προετοιμάζονται σωστά για την εισαγωγή στη Neo4j. Παρακάτω αποτυπώνεται σε μορφή ψευδοκώδικα ο τρόπος μετατροπής των αρχείων JSON σε CSV:

1. Φόρτωσε το(τα) αρχείο(α) JSON από τον καθορισμένο κατάλογο. Για κάθε αντικείμενο JSON:
2. Εξήγαγε τα σχετικά πεδία για τους κόμβους και τις σχέσεις.
3. Δομήσε τα εξαγόμενα δεδομένα σε μορφή CSV.
4. Αποθήκευσε τα δομημένα δεδομένα στα κατάλληλα αρχεία CSV.
5. Βεβαιώσου ότι τα αρχεία CSV είναι μορφοποιημένα σύμφωνα με τις απαιτήσεις εισαγωγής του Neo4j.
6. Διαχειρίσου τυχόν ειδικές περιπτώσεις ή ελλιπή δεδομένα με τρόπο που να αποφεύγονται σφάλματα κατά την εισαγωγή.

Βασικές Συναρτήσεις

Παρακάτω παρατίθενται οι βασικές συναρτήσεις που χρησιμοποιούνται για την μετατροπή των αρχείων:

- Load JSON Files: Η πρώτη λειτουργία του script είναι να εντοπίσει και να φορτώσει τα αρχεία JSON από τον καθορισμένο κατάλογο. Αυτό γίνεται μέσω της συνάρτησης load_json_files, η οποία ανοίγει τα αρχεία JSON και προετοιμάζει τα δεδομένα για επεξεργασία.

- **Extract and Structure Data:** Η συνάρτηση `extract_data` αναλαμβάνει την εξαγωγή των σχετικών δεδομένων από κάθε αντικείμενο JSON. Αυτό περιλαμβάνει τη συλλογή πληροφοριών όπως τα IDs, τα ονόματα, και τις σχέσεις μεταξύ των οντοτήτων. Η λογική πίσω από την επιλογή των δεδομένων βασίζεται στη διατήρηση των `relationships` που είναι κρίσιμα για τη δημιουργία του γραφήματος στη Neo4j.
- **Format to CSV:** Αφού εξαγάγουν τα δεδομένα, αυτά μορφοποιούνται σε μορφή CSV μέσω της συνάρτησης `format_to_csv`. Αυτή η συνάρτηση εξασφαλίζει ότι τα δεδομένα είναι συμβατά με τις απαιτήσεις εισαγωγής του Neo4j, δηλαδή ότι οι στήλες έχουν τις σωστές επικεφαλίδες και ότι τα δεδομένα είναι στη σωστή μορφή όπως ανέλυσα σε παραπάνω κεφάλαια.
- **Save CSV Files:** Τα μορφοποιημένα δεδομένα αποθηκεύονται σε αρχεία CSV με τη χρήση της συνάρτησης `save_csv_files`. Κάθε τύπος οντότητας (π.χ. `author`, `work`, `institution`) αποθηκεύεται σε ξεχωριστό αρχείο CSV, έτοιμο για εισαγωγή στη βάση δεδομένων.

Ροή Δεδομένων κατά τη μετατροπή

Η ροή των δεδομένων κατά τη μετατροπή είναι η εξής:

1. **Input:** Τα αρχεία JSON διαβάζονται από τον καθορισμένο κατάλογο.
2. **Processing:** Μέσω των διαφόρων λειτουργιών του `script`, τα δεδομένα εξάγονται, μορφοποιούνται και προετοιμάζονται για την εισαγωγή τους στη Neo4j.
3. **Output:** Τα τελικά δεδομένα αποθηκεύονται σε αρχεία CSV που είναι έτοιμα για την εισαγωγή τους στη βάση δεδομένων.

Καθώς η Neo4j απαιτεί δεδομένα σε μορφή CSV για την αποδοτική επεξεργασία και ανάλυση, η σωστή μορφοποίηση και διαχείριση των δεδομένων σε αυτή τη φάση είναι κρίσιμη [31]. Το `script json2csv_neon4j.py` διασφαλίζει ότι τα δεδομένα από την OpenAlex προετοιμάζονται κατάλληλα, με τα σωστά `relationships` και τις απαραίτητες πληροφορίες, ώστε να εισαχθούν χωρίς προ- βλήματα στη Neo4j.

Αυτό το `script` είναι αναπόσπαστο μέρος της διαδικασίας ETL (Extract, Transform, Load) που αναπτύχθηκε για τη συγκεκριμένη εφαρμογή, διασφαλίζοντας την ομαλή ροή και την ακρίβεια των δεδομένων από την αρχική πηγή μέχρι την τελική τους αποθήκευση στη βάση δεδομένων γραφημάτων.

5.5.4 Τμήμα Κώδικα για το Neo4j Import (`py_admin_import.py`)

Στο υποκεφάλαιο αυτό θα αναλύσουμε το `script py_admin_import.py`, το οποίο είναι υπεύθυνο για την αυτοματοποίηση της διαδικασίας εισαγωγής δεδομένων στη βάση δεδομένων Neo4j. Αυτό το `script` συνδυάζει πολλές λειτουργίες για να διασφαλίσει την ομαλή και αποτελεσματική εισαγωγή μεγάλων όγκων δεδομένων, όπως τα δεδομένα της OpenAlex. Συγκεκριμένα το `script py_admin_import.py` χρησιμοποιεί μια σειρά από εντολές και μεθόδους για να προετοιμάσει και να εκτελέσει την εισαγωγή δεδομένων στη Neo4j. Οι κύριες λειτουργίες του περιλαμβάνουν τη διαχείριση των διαδρομών αρχείων, το φιλτράρισμα των αρχείων CSV, την κατασκευή των εντολών εισαγωγής και την εκτέλεση αυτών των εντολών.

Διαχείριση Διαδρομών Αρχείων

Η πρώτη σημαντική λειτουργία του script είναι η `translate_path()`, η οποία διασφαλίζει ότι οι διαδρομές των αρχείων είναι συμβατές με το λειτουργικό σύστημα στο οποίο εκτελείται το script. Αυτό είναι ιδιαίτερα σημαντικό όταν η εφαρμογή τρέχει σε διαφορετικά περιβάλλοντα (Windows ή Unix-like συστήματα).

```
def translate_path(path_str):
    if os.name == "nt": path = PureWindowsPath(path_str)
    else: path = PurePosixPath(path_str) if
    os.name == "nt":
    translated_path = str(path).replace("/",
    "\textbackslash").replace else: translated_path
    =str(path).replace("\textbackslash", "/").replace
    return translated_path
```

Η συνάρτηση αυτή μετατρέπει τις διαδρομές των αρχείων ανάλογα με το λειτουργικό σύστημα, καθιστώντας το script ευέλικτο και λειτουργικό σε διάφορα περιβάλλοντα.

Κατασκευή και Φιλτράρισμα των Εντολών Εισαγωγής

Η `construct_import_command()` είναι η καρδιά του script, καθώς εδώ γίνεται η κατασκευή της εντολής εισαγωγής που θα εκτελεστεί στη Neo4j. Η συνάρτηση αυτή φιλτράρει τα αρχεία CSV από το φάκελο εισαγωγής και τα οργανώνει με βάση τα patterns που ακολουθούν συγκεκριμένα κριτήρια. Τα αρχεία αυτά περιλαμβάνουν πληροφορίες για entities όπως authors, works, και institutions, καθώς και τις μεταξύ τους σχέσεις (relationships). Το script φροντίζει να αναγνωρίσει και να συμπεριλάβει τα κατάλληλα αρχεία για κάθε entity και relationship. Οι επιλογές φιλτραρίσματος γίνονται βάσει προκαθορισμένων patterns, όπως φαίνεται παρακάτω:

```
file_patterns_csv = \{
    "author_files": r"author\_d+\textbackslash{}.csv",
    "author_concept_files":
    r"author\_concept\_d+\textbackslash{}.csv",
    "author_has_concept_files":
    r"author\_has\_concept\_d+\textbackslash
    "author_has_institution_files":
    r"author\_has\_institution\_d+\textbackslash
    "author_institution_files":
    r"author\_institution\_d+\textbackslash
    "concept_files":
    r"concept\_d+\textbackslash{}.csv",
    "institution_files":
    r"institution\_d+\textbackslash{}.csv",
    "institution_concept_files":
    r"institution\_concept\_d+\textbacksla
    "institution_has_concept_files":
```

```

r"institution\_has\_concept\_\\d+\\t "publisher\_files":r"publisher\_\\d+\\text
"publisher\_concept\_files":r"publisher\_concept\_\\d+\\textbackslash
{"publisher\_has\_concept\_files":r"publisher\_has\_concept\_\\d+\\t
extb "source\_files":
r"source\_\\d+\\textbackslash{}.csv",
"source\_concept\_files":
r"source\_concept\_\\d+\\textbackslash{}.csv",
"work\_files":
r"work\_\\d+\\textbackslash{}.csv",
"work\_author\_files":
r"work\_author\_\\d+\\textbackslash{}.csv",
"work\_concept\_files":
r"work\_concept\_\\d+\\textbackslash{}.csv",
"work\_institution\_files":
r"work\_institution\_\\d+\\textbackslash{}.c
"work\_work\_files":
r"work\_work\_\\d+\\textbackslash{}.csv",
"work\_has\_author\_files":
r"work\_has\_author\_\\d+\\textbackslash{}.c
"work\_has\_concept\_files":
r"work\_has\_concept\_\\d+\\textbackslash{}
"work\_has\_institution\_files":
r"work\_has\_institution\_\\d+\\textbac
"work\_has\_work\_files":
r"work\_has\_work\_\\d+\\textbackslash{}.csv",\}
file\_patterns\_csv\_zip = \{"author\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_author\\textbacksl
"author\_concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_author\\_
"author\_has\_concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_aut
"author\_institution\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_auth
"author\_has\_institution\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\ "concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_concept\\textback
"institution\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_institution\\
"institution\_concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_ins
"institution\_has\_concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+ "publisher\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_publisher\\text
"publisher\_concept\_files":
r"\textbackslash{}d\{8\}\_part\_\\d+\\_publi

```

```

"publisher\_has\_concept\_files":
r"\textbackslash{}d\{8\}_part\_d+\_ "source\_files":
r"\textbackslash{}d\{8\}_part\_d+\_source\textbacksl
"source\_concept\_files":
r"\textbackslash{}d\{8\}_part\_d+\_source\_
"source\_has\_concept\_files":
r"\textbackslash{}d\{8\}_part\_d+\_"work\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\textbackslash{
"work\_author\_files":
r"\textbackslash{}d\
{8\}_part\_d+\_work\_author"work\_concept\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\_conc
"work\_institution\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\_
"work\_work\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\_work\te
"work\_has\_author\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\_
"work\_has\_concept\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\
"work\_has\_institution\_files":
r"\textbackslash{}d\{8\}_part\_d+\_w
"work\_has\_work\_files":
r"\textbackslash{}d\{8\}_part\_d+\_work\_has\\}

```

Αυτά τα patterns επιτρέπουν στο script να εντοπίζει τα σωστά αρχεία, όπως π.χ. αρχεία που αφορούν authors, works, και τα αντίστοιχα relationships. Στη συνέχεια, τα φιλτραρισμένα αρχεία χρησιμοποιούνται για την κατασκευή της τελικής εντολής εισαγωγής.

Εκτέλεση της Εισαγωγής

Αφού κατασκευαστεί η εντολή, το script την εκτελεί χρησιμοποιώντας το os.system(), εξασφαλίζοντας ότι όλα τα δεδομένα εισάγονται στη βάση δεδομένων Neo4j με την απαραίτητη ακρίβεια και αποτελεσματικότητα. Παρακάτω παρουσιάζεται ένα παράδειγμα εντολής εισαγωγής:

```

\# Construct the import command
import\_command = [
f"\{import\_parent\}/bin/neo4j-admin",
"database", "import", "full",
"--skip-duplicate-nodes",
"--skip-bad-relationships", "--
ignore-empty-strings",
"--overwrite-destination", "--
multiline-fields=true",
"--bad-tolerance=1000000000000000", "--maxoff-heap-
memory=90\%",

```

```
--threads=2",  
--ignore-extra-columns=true",  
--auto-skip-subsequent-headers=true", "--  
highparallel-io=auto", ]
```

Η εντολή αυτή χρησιμοποιεί τις παραπάνω παραμέτρους οι οποίες έχουν αναλυθεί διεξοδικά ως προς την χρησιμότητα και την λειτουργία του σε παραπάνω κεφάλαια με στόχο την εξασφάλιση της ομαλής εισαγωγής των δεδομένων.

Ανάλυση της Ροής Εισαγωγής

Η ροή της εισαγωγής δεδομένων ξεκινά με την προετοιμασία των paths και την κατασκευή των εντολών εισαγωγής. Τα αρχεία CSV που περιέχουν τα δεδομένα των entities και των relationships φιλτράρονται και οργανώνονται σύμφωνα με τα patterns.

Αφού κατασκευαστεί η εντολή, εκτελείται για να εισαγάγει όλα τα δεδομένα στη Neo4j, δημιουργώντας τους κόμβους και τις σχέσεις που απαιτούνται. Το φιλτράρισμα των δεδομένων είναι κρίσιμο καθώς διασφαλίζει ότι μόνο τα απαραίτητα δεδομένα εισάγονται στη βάση. Με αυτόν τον τρόπο, το script διαχειρίζεται αποτελεσματικά τον όγκο των δεδομένων και αποφεύγει περιττές εισαγωγές, που θα μπορούσαν να επηρεάσουν την απόδοση της βάσης δεδομένων. Το py'_admin'_import.py είναι ένα ισχυρό εργαλείο που αυτοματοποιεί και βελτιστοποιεί τη διαδικασία εισαγωγής δεδομένων, καθιστώντας το αναπόσπαστο μέρος της διαχείρισης των δεδομένων OpenAlex στο πλαίσιο του παρόντος project.

Κεφάλαιο 6

Στρατηγικές Βελτιστοποίησης Κώδικα και παραδείγματα εξόδου μέσω Cypher Queries

Σε αυτή την ενότητα, θα αναλύσουμε τις στρατηγικές βελτιστοποίησης που εφαρμόστηκαν για την αντιμετώπιση και για τον τρόπο που αντιμετωπίστηκαν τα διάφορα θέματα καθώς και των προβλημάτων της βάσης κατά την διάρκεια του import των δεδομένων της OpenAlex στη Neo4j. Οι περιορισμοί αυτοί οφείλονται στο μεγάλο όγκο των δεδομένων και στην πολυπλοκότητα τους καθώς και στις αυστηρές απαιτήσεις της διαδικασίας εισαγωγής της βάσης δεδομένων Neo4j, η οποία απαιτεί τα αρχεία εισαγωγής να βρίσκονται σε συγκεκριμένους φακέλους του συστήματος αρχείων. Για παράδειγμα, με κάθε προσθήκη αρχείων στον φάκελο του import από τον οποίο θα διαβάζει τα δεδομένα η Neo4j οδηγεί την διαδικασία να είναι εξαιρετικά χρονοβόρα.

6.1 Βελτιστοποιήσεις Κώδικα για την Αντιμετώπιση των Περιορισμών Χώρου

6.1.1 Συμπίεση και Φιλτράρισμα Δεδομένων

Η διαδικασία φιλτραρίσματος και συμπίεσης των δεδομένων είχε ως αποτέλεσμα τη σημαντική μείωση του όγκου των δεδομένων, καθιστώντας τα πιο διαχειρίσιμα για το σύστημα και επιτρέποντας την αποτελεσματική τους αποθήκευση και επεξεργασία. Στην αρχική τους μορφή, τα δεδομένα της OpenAlex (ολόκληρη, όλα τα update dates των parts), αποθηκευμένα σε αρχεία .json.tar.gz, κατέλαβαν συνολικά 433.10 GB.

Μέσω της διαδικασίας φιλτραρίσματος, τα δεδομένα μειώθηκαν σε σημαντικό βαθμό. Ολόκληρη η βάση, για παράδειγμα, μειώθηκε από 433,1 GB σε 173 GB, συμπεριλαμβανομένων σε αρχεία .csv.zip. Αυτή η μείωση επιτεύχθηκε με την επιλογή συγκεκριμένων κλειδιών (keys) που ήταν απαραίτητα για τη διατήρηση των κρίσιμων σχέσεων (relationships) μεταξύ των οντοτήτων της βάσης. Τα παρακάτω είναι τα πρακτικά μεγέθη των δεδομένων για τις διαφορετικές κατηγορίες της OpenAlex:

- institution:

Συνολικό Μέγεθος: 0.17 GB (187154873 Bytes)

- concept:
Συνολικό Μέγεθος: 0.13 GB (91986932 Bytes)
- publisher:
Συνολικό Μέγεθος: 0.12 GB (3348997 Bytes)
- source:
Συνολικό Μέγεθος: 0.34 GB (365724837 Bytes)
- author:
Συνολικό Μέγεθος: 52.31 GB (56166736075 Bytes)
- works:
Συνολικό Μέγεθος: 380.03 GB (408227975886 Bytes)
- Overall Total Size: Συνολικό Μέγεθος: 433.10 GB (465042927600 Bytes)

Αξίζει να σημειωθεί ότι αυτά τα μεγέθη είναι ενδεικτικά και μπορεί να αλλάξουν ελαφρώς με την πάροδο του χρόνου, καθώς η βάση δεδομένων OpenAlex είναι δυναμική και ανανεώνεται τακτικά, επομένως αυτά τα μεγέθη που αναγράφονται παραπάνω είναι την δεδομένη στιγμή που πραγματοποιήθηκε η λήξη των δεδομένων. Η OpenAlex είναι μοναδική στο είδος της, καθώς παρέχει μια εκτενή και διαρκώς εξελισσόμενη βάση δεδομένων με πλήθος εγγράφων που αφορούν ακαδημαϊκά έργα, συγγραφείς, οργανισμούς και άλλα σχετικά δεδομένα [10]. Η δυνατότητα να αποθηκεύσουμε και να επεξεργαστούμε αυτά τα δεδομένα σε ένα περιβάλλον όπως το Neo4j απαιτούσε στρατηγικές βελτιστοποίησης που προσαρμόστηκαν ειδικά στις ανάγκες και τους περιορισμούς του έργου.

6.1.2 Κριτήρια Φιλτραρίσματος Δεδομένων

Το φιλτράρισμα των δεδομένων βασίστηκε σε συγκεκριμένα κριτήρια που εξασφάλισαν ότι τα κρίσιμα δεδομένα για τη δημιουργία σχέσεων δεν θα απορριφθούν.

Συγκεκριμένα:

- Διατήρηση Σχέσεων (Relationships): Τα "keys" που διατηρήθηκαν ήταν αυτά που οδηγούν στη δημιουργία σχέσεων μεταξύ διαφορετικών οντοτήτων, όπως για παράδειγμα οι σχέσεις μεταξύ authors και works. Ο αριθμός των σχέσεων μεταξύ των entities της OpenAlex αποτελεί περιπλοκή διαδικασία. Κάθε σχέση μπορεί να είναι η αρχή και μιας επόμενης σχέσης επομένως αυτή η αλυσιδωτή σύνδεση μεταξύ των στοιχείων είναι πολύ κρίσιμο να γίνει με προσοχή ώστε να μην χαθεί οποιαδήποτε αλυσίδα πληροφορίας. Η διαδικασία του linking pairs (Δημιουργία των relationships) κατά την διάρκεια του import ήταν και η πιο χρονοβόρας διότι απαιτεί πολλά resources τόσο από την Neo4j όσο και από το σύστημα. Η διατήρηση αυτών των δεδομένων ήταν απαραίτητη για την ακεραιότητα της βάσης δεδομένων Neo4j, επομένως δεν γινόταν να προστεθεί κανένα περαιτέρω φιλτράρισμα διότι θα είχε ως επίπτωση στην μη πληρότητα της πληροφορίας των δεδομένων μας.

- **Απόρριψη Μη Κρίσιμων Δεδομένων:** Τα δεδομένα που δεν συνεισέφεραν στη δημιουργία σχέσεων απορρίφθηκαν για να μειωθεί ο όγκος των δεδομένων. Ωστόσο, αυτή η απόρριψη έγινε με προσοχή ώστε να διασφαλιστεί ότι δεν θα δημιουργηθούν κενά στις σχέσεις της βάσης.
- **Αντικατάσταση Κενών Εγγραφών:** Στην περίπτωση που κάποια δεδομένα ήταν ελλιπή ή περιείχαν κενές εγγραφές, αυτές οι εγγραφές αντικαταστάθηκαν με κενές λίστες ή κενές συμβολοσειρές, ανάλογα με τον τύπο των δεδομένων. Αυτό επέτρεψε τη διατήρηση της συνέπειας στη βάση δεδομένων.

Η διαδικασία φιλτραρίσματος και συμπίεσης είχε σημαντικό αντίκτυπο στη μείωση του όγκου των δεδομένων. Ένα συγκεκριμένο παράδειγμα παρουσιάζει τη διαφορά μεταξύ του αρχικού και του φιλτραρισμένου μεγέθους ενός αρχείου (entity: authors, updated_date: 2024-08-26, part : part_000):

- Αρχείο πριν το φιλτράρισμα:
Μέγεθος: 430.40 MiB
- Αρχείο μετά το φιλτράρισμα:
Μέγεθος: 66.81 MiB

Ένα άλλο παράδειγμα αρχείου(entity: works, updated_date: 2024-08-26, part : part_000):

- Αρχείο πριν το φιλτράρισμα:
Μέγεθος: 781.39 MiB
- Αρχείο μετά το φιλτράρισμα: Μέγεθος: 140.04 MiB

Αυτό το παράδειγμα δείχνει πώς η διαδικασία φιλτραρίσματος μπορεί να μειώσει σημαντικά το μέγεθος των δεδομένων, διατηρώντας παράλληλα τις κρίσιμες πληροφορίες.

6.1.3 Περιορισμοί και Προκλήσεις

Κατά την υλοποίηση της εφαρμογής του import, υπήρξαν ορισμένοι περιορισμοί και προκλήσεις που έπρεπε να αντιμετωπιστούν, από τους οποίους κάποιοι ξεπεράστηκαν, ενώ κάποιοι δεν ήταν δυνατό να βρεθεί λύση.

Χωρητικότητα Δίσκων

Ο περιορισμός της χωρητικότητας των δίσκων στο τοπικό μηχάνημα(80 GB)με ανάγκασε να εφαρμόσω πολύ αυστηρό φιλτράρισμα για να μειωθεί το μέγεθος της βάσης διατηρώντας όμως τις κρίσιμες και απαραίτητες πληροφορίες ώστε να απεικονιστεί ολόκληρη η εικόνα της OpenAlex. Όπως παρατήρησα στο τοπικό σύστημα η διαδικασία του import είναι μια πολυεπίπεδη διαδικασία, πολύ χρονοβόρα και αρκετά περίπλοκη. Παρόλο που αξιοποίησα το μέγιστο της μνήμης του μηχανήματος τόσο τοπικά όσο και απομακρυσμένα, το import χρειαζόταν μεγάλο χρονικό διάστημα για να ολοκληρωθεί, το οποίο έκανε κάθε προσπάθεια πολύ χρονοβόρα. Χρησιμοποιώντας την εντολή `watch -n 5 du -sh /path/to/folder` σε ένα παράλληλο terminal

γινόταν η παρακολούθηση του αποθηκευτικού χώρου παράλληλα με τη χρήση της εντολής `htop` που εγκαταστάθηκε μέσω του `sudo apt-get install htop` πραγματοποιήθηκε ο απαραίτητος έλεγχος χρήσης της CPU. Τέλος η εντολή `iostat -x 5` εξυπηρέτησε στον έλεγχο δραστηριότητας του δίσκου.

Διαχείριση Σφαλμάτων Λήψης

Εάν κάποιο αρχείο αποτύχει να κατέβει μετά από έναν ορισμένο αριθμό προσπαθειών (π.χ. 10), το αρχείο απορρίπτεται διότι μπορεί να είναι αλλοιωμένο και να δημιουργήσει θέμα στην δημιουργία της βάσης. Αυτό μπορεί να οδηγήσει σε απώλεια δεδομένων που είναι απαραίτητα για τη δημιουργία σχέσεων στη βάση. Αν ένα αρχείο που μετά από τον ορισμένο αριθμό προσπαθειών λήψης αποτύχει να κατεβεί τότε αυτό μπορεί να προκαλέσει έλλειψη πληροφοριών που θα οδηγήσει σε ελλιπή σχέσεις μεταξύ κάποιων στοιχείων. Αυτό έχει ως επακόλουθο τα αρχεία με ελλιπείς σχέσεις να απορριφθούν από το `import`, αυτό συμβαίνει εξαιτίας των εντολών που προστέθηκαν κατά την διαδικασία του `neo4j admin import` οι οποίες απορρίπτουν ό,τι διπλότυπο ή ελλιπές.

Επιλογή Εγγραφών

Η επιλογή των εγγραφών που θα διατηρηθούν στη βάση ήταν μια εξαιρετικά περίπλοκη διαδικασία, καθώς απαιτούσε μια ισορροπία μεταξύ της μείωσης του μεγέθους των δεδομένων και της διατήρησης των κρίσιμων πληροφοριών που είναι απαραίτητες για τη λειτουργικότητα της βάσης. Κατά την εφαρμογή του φιλτραρίσματος, το οποίο θα μπορούσε να μειώσει το συνολικό μέγεθος της OpenAlex μέχρι και στα 47 GB, προχώρησα στην απόρριψη ολόκληρων αρχείων που περιείχαν "missing entries". Επειδή το μέγεθος αυτό ήταν απαγορευτικά μικρό για τις απαιτήσεις μας, δημιούργησα μια πιο ήπια στρατηγική. Με αυτή τη στρατηγική φιλτραρίσματος λοιπόν της παρούσας δουλειάς, αυτά τα missing entries αντικαταστάθηκαν από κενές λίστες ή κενές συμβολοσειρές, ανάλογα με τον τύπο των δεδομένων, χωρίς να απορρίπτεται ολόκληρο το αρχείο και να χάνουμε κρίσιμη πληροφορία.

Δυσκολίες κατά την Διαδικασία του import

Η διαδικασία του `neo4j admin import` είναι εξαιρετικά απαιτητική τόσο από άποψη χρόνου όσο και από άποψη πόρων του συστήματος. Από την αρχή της υλοποίησης της εφαρμογής οπού και έγινε πειραματική δοκιμή με πολύ μικρό αριθμό δεδομένων (100Mb), ο χρόνος που απαιτήθηκε από το σύστημα ώστε να ολοκληρωθεί η διαδικασία του `import` και να εμφανιστεί η βάση ήταν εξ αρχής αρκετό. Η διαδικασία χωρίζεται σε 4 στάδια. Το εργαλείο αρχικά διαβάσει τα δεδομένα που αφορούν τους κόμβους, ελέγχοντας για τη συνοχή τους, όπως την ύπαρξη των απαραίτητων ετικετών και τη μοναδικότητα των IDs. Στη συνέχεια, το εργαλείο διαβάσει τα δεδομένα των σχέσεων, εξασφαλίζοντας ότι τα IDs των κόμβων που αναφέρονται στους πίνακες σχέσεων υπάρχουν και ότι οι τύποι των σχέσεων είναι έγκυροι. Κατά τη διάρκεια του `import`, οι κόμβοι και οι σχέσεις δημιουργούνται αρχικά στη μνήμη. Το εργαλείο εκτελεί διάφορες εργασίες στο παρασκήνιο όπως η ανάθεση των ετικετών. Οι ετικέτες που σχετίζονται με κάθε κόμβο ανατίθενται ώστε να μπορούν να ταξινομηθούν οι κόμβοι. Κάθε κόμβος και σχέση αποκτά τις ιδιότητες που έχουν δηλωθεί στα αρχεία CSV. Οι σχέσεις χρησιμοποιούν τα IDs των κόμβων για να συνδέσουν μεταξύ τους τους κόμβους. Αυτές οι ενέργειες γίνονται

αρχικά σε προσωρινές δομές στη μνήμη, ώστε να επιτυγχάνεται γρήγορη διαχείριση και ταξινόμηση. Η διαδικασία αυτή γίνεται τμηματικά, κάθε τμήμα από τα τέσσερα βασικά στοιχεία (Node import, Relationship import, Properties Import, Mapping) έχει τα δικά του πιθανά προβλήματα που μπορεί να εγερθούν όπως θέματα data corruption, ανακρίβειες δεδομένων που να οδηγήσουν σε μεγάλο αριθμό προβληματικών συνδέσεων. Για την αποφυγή τους έχω θέσει ένα ανώτερο όριο στις παραμέτρους του αποδεκτού αριθμού που θα δεχτεί η βάση μας. Ειδικά, δεν θα ήταν δυνατό να χειριστεί τα δεδομένα και τις ανακρίβειες τους. Όλη αυτή η διαδικασία, τα 4 τμήματα του import, καθώς και ο έλεγχος που κάνει κάθε παράμετρος της neo4j admin import εντολής, κάνει την διαδικασία πολύ χρονοβόρα. Κατά την διάρκεια των import, αντιμετώπισα αποσυνδέσεις είτε από τον απομακρυσμένο υπολογιστή είτε εξαιτίας κακής σύνδεσης. Ένα διαφορετικό πρόβλημα που αντιμετώπισα την πως η διαδικασία μπορεί να σταματούσε σε κάποιο σημείο του import δίχως εμφάνιση κάποιου μηνύματος λάθους. Σε κάθε περίπτωση έπειτα από την δημιουργία των nodes κάθε ένα μικρό μέρος του upload απαιτούσε περίπου στις 3-4 ώρες για να συνεχίσει. Κάθε τμήμα από τα 4 που προανέφερα είχε το δικό του loading time (0- 100%), επομένως συνολικά η διαδικασία απαιτούσε τόσο ο τοπικός όσο και ο απομακρυσμένος Η/Υ να είναι ανοιχτοί και να τρέχουν για πολλές ώρες προκαλώντας υπερθέρμανση της συσκευής.

Δεδομένα της OpenAlex

Η OpenAlex πάρα την χρησιμότητα της, την δωρεάν χρήση της, και του όγκου των ερευνητικών έργων που προσφέρει για τους χρήστες, τα δεδομένα της παρουσιάζουν αρκετά προβλήματα. Αρχικά, τα δεδομένα που σχετίζονται με τα άρθρα, όπως οι τίτλοι, οι περιλήψεις (abstracts) και τα έτη δημοσίευσης, μπορεί να είναι ελλιπή ή ανακριβή. Αυτό οφείλεται κυρίως στην πηγή των δεδομένων, που είναι πολλές φορές ακαδημαϊκές εκδόσεις ή τρίτα συστήματα, τα οποία δεν παρέχουν πάντα πλήρη ή σωστά στοιχεία. Οι σχέσεις μεταξύ άρθρων, ερευνητών και θεσμών μπορεί να μην είναι πάντα σωστές. Αυτό μπορεί να προκύψει από σφάλματα στην αυτόματη εξαγωγή δεδομένων, όπως η λανθασμένη ταυτοποίηση ενός ερευνητή ή ενός θεσμού με ένα συγκεκριμένο έργο ή δημοσίευση. Επίσης στην OpenAlex παρατηρούνται συχνά διπλότυπα δεδομένα. Για παράδειγμα, η ίδια δημοσίευση μπορεί να εμφανίζεται περισσότερες από μία φορές με ελαφρώς διαφορετικά στοιχεία, όπως διαφορετικές εκδόσεις του τίτλου ή παραλλαγές στο όνομα του συγγραφέα. Αυτά τα διπλότυπα προκύπτουν λόγω της συνένωσης δεδομένων από διαφορετικές πηγές, οι οποίες δεν είναι πάντα συντονισμένες. Τα ονόματα των οργανισμών και των ερευνητών μπορεί να μην είναι συνεπή, με παραλλαγές σε ορθογραφία, τυπογραφία ή μορφοποίηση. Αυτό μπορεί να προκαλέσει δυσκολίες στην ανάλυση και στην ταυτοποίηση των σωστών οντοτήτων. Η OpenAlex συλλέγει δεδομένα από πολλές εξωτερικές πηγές, και αυτό μπορεί να οδηγήσει σε προβλήματα ποιότητας δεδομένων. Για παράδειγμα, αν οι εξωτερικές πηγές παρέχουν ανακριβή ή μη ενημερωμένα δεδομένα, αυτά τα προβλήματα θα περάσουν και στην OpenAlex. Ανάλογα με την πηγή, μπορεί να υπάρχουν ανακρίβειες στις βιβλιογραφικές αναφορές, τα citations και τα έτη δημοσίευσης. Το citation graph που δημιουργείται από την OpenAlex (δηλαδή ο τρόπος με τον οποίο συνδέονται οι αναφορές μεταξύ των άρθρων) μπορεί να περιέχει κενά ή λάθη. Ορισμένα άρθρα μπορεί να μην έχουν όλες τις αναφορές τους σωστά καταγεγραμμένες ή να εμφανίζουν λανθασμένα citations. Υπάρχουν περιπτώσεις όπου το ίδιο άτομο εμφανίζεται ως δύο διαφορετικοί ερευνητές λόγω διαφορετικών μορφών ονόματος (π.χ., μικρές παραλλαγές στη γραφή ή τη χρήση μεσαίων αρχικών) καθώς και με το ίδιο

ID. Αυτό είναι ένα συνηθισμένο πρόβλημα στα δεδομένα της OpenAlex, ειδικά σε πολυπληθείς τομείς όπου πολλοί ερευνητές έχουν παρόμοια ονόματα. Όλες αυτές οι ιδιαιτερότητες και τα προβλήματα της OpenAlex επηρέασαν σημαντικά τόσο στην διαδικασία του import αλλά και στην δημιουργία του σωστού format για τα csv αρχεία τα οποία με βάση την Neo4j πρέπει να είναι αυστηρώς δημιουργημένα και σωστά ώστε να τα διαβάσει με βάση την αυστηρή πολιτική της.

Δημιουργία ειδικών CSV αρχείων

Εξαιτίας των δεδομένων της OpenAlex τα οποία χρειαζόντουσαν διαχείριση, η διαδικασία του formatting και της δημιουργίας των CSV αρχείων ήταν αρκετά περίπλοκη. Τα βασικά προβλήματα που αντιμετώπισα περιλαμβάνουν διπλότυπα δεδομένα, ασυνέπειες στις σχέσεις και τις ιδιότητες, θέματα με τον όγκο των δεδομένων, καθώς και προβλήματα κατά τη μετατροπή των δεδομένων από JSON σε CSV. Χρειάστηκε η αφιέρωση χρόνου λόγω αρκετών δοκιμών κατά την προετοιμασία και τον καθαρισμό των δεδομένων πριν την εισαγωγή στο Neo4j ώστε να εξασφαλιστεί η ακριβής και η πληρότητα της βάσης. Όπως ανέφερα και παραπάνω στην OpenAlex, πολλές φορές τα δεδομένα προέρχονται από διάφορες εξωτερικές πηγές, και αυτό οδηγεί σε ασυνέπειες. Αυτό δημιουργεί πρόβλημα όταν τα δεδομένα πρέπει να εισαχθούν σε μια βάση γνώσεων όπως το Neo4j, που βασίζεται σε αυστηρές σχέσεις μεταξύ κόμβων. Κατέστη επομένως αναγκαίο να καθαριστούν κατάλληλα αυτά τα δεδομένα, ειδικά το αποτέλεσμα θα είναι η ύπαρξη διπλότυπων κόμβων που αντιστοιχούν στην ίδια οντότητα. Επιπλέον, η κανονικοποίηση των ιδιοτήτων είναι απαραίτητη. Ένα παράδειγμα κανονικοποίησης των ιδιοτήτων είναι οι ημερομηνίες δημοσίευσης που πρέπει να έχουν το ίδιο format. Στην OpenAlex, μπορεί να υπάρχουν διαφορετικές αναπαραστάσεις της ημερομηνίας, όπως "2020-06-15" ή "15 Ιουνίου 2020", κάτι που προκαλεί ασυνέπειες κατά την εισαγωγή στο Neo4j. Το ίδιο ισχύει και για άλλες ιδιότητες όπως οι τίτλοι άρθρων, που μπορεί να περιέχουν διαφορετική μορφοποίηση (π.χ. κεφαλαία γράμματα σε μία περίπτωση, πεζά σε άλλη). Ένα άλλο σημαντικό ζήτημα που αντιμετώπισα είναι η ασυμβατότητα των τύπων δεδομένων μεταξύ της OpenAlex και του Neo4j. Η OpenAlex αποθηκεύει δεδομένα σε μορφή JSON, όπου ορισμένες ιδιότητες μπορεί να είναι πιο πολύπλοκες, όπως arrays ή nested objects. Για παράδειγμα, οι συγγραφείς μιας δημοσίευσης μπορεί να αποθηκεύονται σε μορφή array με πολλαπλές τιμές, ενώ στο Neo4j, οι κόμβοι αναμένουν απλές τιμές ή σχέσεις μεταξύ κόμβων. Αυτό σημαίνει ότι χρειάστηκε να γίνει διαχωρισμός των πιο σύνθετων δεδομένων σε μία πιο απλή μορφή που μπορεί να υποστηρίξει το Neo4j. Επιπλέον, άλλοι τύποι δεδομένων, όπως strings, αριθμοί, ή ακόμη και Boolean, επρεπε να ελεγχθούν για να διασφαλιστεί ότι είναι συμβατά με το σχήμα του Neo4j. Για παράδειγμα, μια ιδιότητα που σε κάποια εγγραφή της OpenAlex μπορεί να είναι string, σε μια άλλη εγγραφή μπορεί να είναι αριθμός. Αυτό μπορεί να δημιουργήσει προβλήματα κατά τη δημιουργία των CSV αρχείων, και πρέπει να διασφαλιστεί ότι οι τιμές είναι συνεπείς. Επίσης, η Neo4j υποστηρίζει συγκεκριμένους τύπους δεδομένων (π.χ. ημερομηνίες, αριθμούς) και οι ιδιότητες πρέπει να έχουν το σωστό format ώστε να μπορούν να αναγνωριστούν σωστά από τη βάση.

Με βάση τα παραπάνω πραγματοποιήθηκε η διαδικασία φιλτραρίσματος δεδομένων, η οποία στηρίχθηκε σε μια στρατηγική που περιελάμβανε την επιλογή συγκεκριμένων κλειδιών (keys) για τη διατήρησή τους. Προκειμένου να διασφαλιστεί η διατήρηση των πιο κρίσιμων στοιχείων της βάσης τα κλειδιά που επιλέχθηκαν για διατήρηση περιλαμβάνουν τα εξής, παραδείγματα:

- id: Το μοναδικό αναγνωριστικό για κάθε εγγραφή.

- `display_name`: Το όνομα που εμφανίζεται για κάθε εγγραφή.
 - `updated_date`: Η ημερομηνία της τελευταίας ενημέρωσης της εγγραφής.
 - `created_date`: Η ημερομηνία δημιουργίας της εγγραφής.
 - `x_concepts`: Οι έννοιες που σχετίζονται με την εγγραφή.
 - `x_concepts@id`: Το μοναδικό αναγνωριστικό για κάθε σχετική έννοια.
 - `x_concepts@display_name`: Το όνομα της έννοιας που σχετίζεται με την εγγραφή.
 - `doi`: Ο αναγνωριστικός κωδικός DOI για κάθε εργασία.
 - `type`: Ο τύπος της εγγραφής.
 - `type_id`: Το αναγνωριστικό του τύπου της εγγραφής.
1. Για τις εγγραφές που αφορούν `authors`, παραδείγματα:
 - `affiliations@institution@id`: Το αναγνωριστικό του ιδρύματος με το οποίο σχετίζεται ο συγγραφέας.
 - `affiliations@institution@display_name`: Το όνομα του ιδρύματος με το οποίο σχετίζεται ο συγγραφέας.
 2. Για τις εγγραφές που αφορούν `works`, παραδείγματα:
 - `authorships@author@id`: Το αναγνωριστικό του συγγραφέα της εργασίας.
 - `authorships@author@display_name`: Το όνομα του συγγραφέα της εργασίας.
 - `authorships@institutions@id`: Το αναγνωριστικό του ιδρύματος με το οποίο σχετίζεται η εργασία.
 - `authorships@institutions@display_name`: Το όνομα του ιδρύματος με το οποίο σχετίζεται η εργασία.
 - `concepts@id`: Το αναγνωριστικό της έννοιας που σχετίζεται με την εργασία.
 - `concepts@display_name`: Το όνομα της έννοιας που σχετίζεται με την εργασία.
 - `related_works`: Άλλες εργασίες που σχετίζονται με την συγκεκριμένη εργασία.
 3. Για τις εγγραφές που αφορούν `sources`, παραδείγματα:
 - `publisher`: Ο εκδότης του πηγής.
 4. Για τις εγγραφές που αφορούν `concepts`, παραδείγματα:
 - `description`: Η περιγραφή της έννοιας.
 - `host_organization_name@id`: Το αναγνωριστικό του οργανισμού που φιλοξενεί την έννοια.

- `host_organization_name@display_name`: Το όνομα του οργανισμού που φιλοξενεί την έννοια.

5. Για τις εγγραφές που αφορούν `institutions`:

- `type`: Ο τύπος του ιδρύματος.

Αυτά τα κλειδιά επιλέχθηκαν ώστε να διατηρηθούν οι απαραίτητες σχέσεις μεταξύ των δεδομένων και να εξασφαλιστεί η συνέπεια της βάσης. Περιλαμβάνει τα πιο σημαντικά κλειδιά που εξασφαλίζουν ότι τα βασικά στοιχεία και οι σχέσεις της βάσης θα διατηρηθούν ανέπαφα. Για παράδειγμα, στο επίπεδο των `authors`, διατηρούνται οι σχέσεις με τα `institutions` μέσω των κλειδιών `'affiliations@institution@id'` και `'affiliations@institution@display_name'`. Στο επίπεδο των `works`, διατηρούνται οι σχέσεις με τους `authors` και τα `concepts` μέσω των κλειδιών `'authorships@author@id'`, `'authorships@author@display_name'`, και `'concepts@id'`.

Η διαδικασία φιλτραρίσματος, ενώ βοήθησε στην αποτελεσματική μείωση του όγκου των δεδομένων, έγινε με προσοχή ώστε να διασφαλιστεί ότι δεν θα χαθούν κρίσιμες πληροφορίες.

6.2 Βελτιστοποιήσεις Κώδικα

Στην υλοποίηση του κώδικα για τη διαχείριση και επεξεργασία των δεδομένων της βάσης OpenAlex, έχουν ενσωματωθεί πολλές βελτιστοποιήσεις που εξασφαλίζουν την αποδοτική χρήση του χρόνου και την κλιμάκωση του συστήματος. Παρακάτω αναλύεται το script που χρησιμοποιήθηκε end-to-end για την διαδικασία του `import` και την λειτουργικότητα του.

6.2.1 Ανάλυση Κώδικα

Στην υλοποίηση του κώδικα για την επεξεργασία των δεδομένων της OpenAlex, υπάρχουν συγκεκριμένα σημεία που ενσωματώνουν έξυπνες λύσεις για τη διαχείριση των επαναλήψεων και της αποθήκευσης των αρχείων. Παρακάτω θα περιγράψω πώς υλοποιούνται αυτά τα σημεία κώδικα.

Διαχείριση Επαναλήψεων (Retries)

Σε σημεία του κώδικα όπου απαιτείται λήψη αρχείων από εξωτερικές πηγές, έχει ενσωματωθεί μια δικλείδα ασφαλείας που διαχειρίζεται τυχόν αποτυχίες κατά τη διαδικασία λήψης. Συγκεκριμένα, υπάρχει μια διαδικασία επαναλήψεων που εκτελείται αν η λήψη αποτύχει.

Παράδειγμα Λειτουργίας: Όταν γίνεται προσπάθεια λήψης ενός αρχείου, και αυτή αποτύχει (π.χ. λόγω διακοπής του δικτύου ή άλλου προσωρινού σφάλματος ή λόγω απόρριψης του αρχείου), η λήψη δεν απορρίπτεται αμέσως. Αντίθετα, ο κώδικας περιλαμβάνει ένα `"try"` που επιχειρεί τη λήψη του αρχείου ξανά και ξανά, μέχρι να επιτευχθεί το μέγιστο αριθμό επαναλήψεων που έχει προκαθοριστεί (συνήθως 10 επαναλήψεις). Μεταξύ των επαναλήψεων, υπάρχει ένα καθορισμένο χρονικό διάστημα αναμονής. Αυτή η στρατηγική επιτρέπει την ολοκλήρωση της λήψης ακόμη και αν υπάρξουν προσωρινές διακοπές, εξαλείφοντας την ανάγκη για ανθρώπινη παρέμβαση όπως ήταν αναγκαίο από την μεγάλη διάρκεια της λήψης.

Επιλογή Διαγραφής ή Διατήρησης Αρχείων

Κατά τη διάρκεια της επεξεργασίας των αρχείων, ο κώδικας παρέχει στον χρήστη τη δυνατότητα να επιλέξει αν τα πρωτότυπα αρχεία θα διατηρηθούν ή θα διαγραφούν μετά την ολοκλήρωση της επεξεργασίας. Αυτή η δυνατότητα υλοποιείται μέσω συγκεκριμένων επιλογών στον κώδικα, που ελέγχουν την κατάληξη των αρχείων μετά την επεξεργασία.

Παράδειγμα Λειτουργίας: Όταν ολοκληρωθεί η επεξεργασία ενός αρχείου, ο κώδικας ελέγχει αν ο χρήστης έχει ενεργοποιήσει την επιλογή διατήρησης του αρχείου. Αν αυτή η επιλογή είναι ενεργή, το αρχείο μετακινείται σε έναν ειδικό φάκελο για διατήρηση. Διαφορετικά, αν η επιλογή δεν είναι ενεργή, το αρχείο διαγράφεται άμεσα από το σύστημα. Αυτή η προσέγγιση επιτρέπει στον χρήστη να ελέγχει τη χρήση του χώρου αποθήκευσης, ιδιαίτερα σε συστήματα με περιορισμένη χωρητικότητα.

Φιλτράρισμα και Ανανέωση Αρχείων Καταγραφής

Ένα ακόμη σημείο βελτιστοποίησης αφορά το φιλτράρισμα των δεδομένων και τη διαχείριση των αρχείων καταγραφής (logs). Ο κώδικας διατηρεί αρχεία καταγραφής που ενημερώνονται μόνο όταν η διαδικασία φιλτραρίσματος ολοκληρωθεί επιτυχώς. Αν το φιλτράρισμα αποτύχει για οποιονδήποτε λόγο, το αρχείο καταγραφής παραμένει αμετάβλητο.

Παράδειγμα Λειτουργίας: Κατά την επεξεργασία ενός αρχείου, αν το φιλτράρισμα ολοκληρωθεί επιτυχώς, το αρχείο καταγραφής ενημερώνεται για να καταγράψει την επιτυχία της διαδικασίας και να αποτρέψει την επανεκτέλεση της ίδιας διαδικασίας στο μέλλον. Αν όμως η διαδικασία φιλτραρίσματος αποτύχει, το αρχείο καταγραφής δεν ενημερώνεται, και το αρχείο παραμένει στο αρχικό του στάδιο για να επαναληφθεί η προσπάθεια φιλτραρίσματος αργότερα. Αυτή η υλοποίηση αποδείχτηκε χρήσιμη στα αρχικά στάδια ελέγχου του φιλτραρίσματος.

Αποτροπή Διπλής Εργασίας και Ακεραιότητα Δεδομένων

Ο κώδικας περιλαμβάνει μηχανισμούς που διασφαλίζουν ότι δεν θα γίνει διπλή επεξεργασία των ίδιων δεδομένων, αποτρέποντας έτσι την απώλεια χρόνου και πόρων.

Παράδειγμα Λειτουργίας: Κατά την εκκίνηση της διαδικασίας, ο κώδικας ελέγχει αν τα δεδομένα έχουν ήδη υποστεί επεξεργασία. Αν διαπιστωθεί ότι η επεξεργασία έχει ολοκληρωθεί επιτυχώς στο παρελθόν, ο κώδικας παραλείπει αυτή τη διαδικασία και προχωρά στην επόμενη, εξοικονομώντας χρόνο και αποφεύγοντας την επαναλαμβανόμενη εργασία.

6.2.2 Αρχείο unzipper.py

Διαχείριση Φιλτραρίσματος και Ανανεώσεων

Στο unzipper.py, υπάρχει μια κρίσιμη διαδικασία που ελέγχει την επιτυχία του φιλτραρίσματος των δεδομένων πριν από τη διαγραφή του αρχικού αρχείου. Η συγκεκριμένη διαδικασία διασφαλίζει ότι αν το φιλτράρισμα αποτύχει για οποιονδήποτε λόγο, όπως π.χ. λόγω απρόσμενης μορφής των δεδομένων, το αρχικό αρχείο παραμένει ανέπαφο. Αυτό υλοποιείται μέσω ενός μηχανισμού που διατηρεί το πρωτότυπο αρχείο μέχρι να ολοκληρωθεί επιτυχώς η διαδικασία φιλτραρίσματος. Μόνο όταν επιβεβαιωθεί ότι το φιλτράρισμα ολοκληρώθηκε χωρίς σφάλματα,

το αρχείο διαγράφεται. Αυτή η προσέγγιση αποτρέπει τη μη αναστρέψιμη απώλεια δεδομένων και προσφέρει μια δικλείδα ασφαλείας που εξασφαλίζει την ακεραιότητα των δεδομένων.

Συμπίεση και Αποθήκευση

Μετά την επιτυχή ολοκλήρωση του φιλτραρίσματος, η διαδικασία προχωρά στη συμπίεση των αρχείων. Σε αυτό το στάδιο, η διαχείριση του χώρου αποθήκευσης είναι κεντρικής σημασίας. Το σύστημα επανασυμπιέζει τα αρχεία με στόχο να διατηρηθούν μόνο τα απολύτως απαραίτητα δεδομένα για τη λειτουργία της βάσης. Αυτό επιτυγχάνεται με την επιλογή συγκεκριμένων "keys" που είναι κρίσιμα για τη δημιουργία σχέσεων (relationships) στη βάση δεδομένων, ενώ όλα τα υπόλοιπα δεδομένα απορρίπτονται.

6.2.3 Ανάλυση και Περιγραφή του Κώδικα στα `py_admin_import.py`

Ο κώδικας στο αρχείο `py_admin_import.py` προσφέρει μία σειρά από προηγμένες λειτουργίες και βελτιστοποιήσεις για την αποδοτική διαχείριση μεγάλου όγκου δεδομένων και την κλιμάκωση της διαδικασίας εισαγωγής τους σε βάσεις δεδομένων όπως η Neo4j. Παρακάτω παρουσιάζονται αναλυτικά οι στρατηγικές και οι τεχνικές που χρησιμοποιούνται στον κώδικα, με έμφαση στις δυνατότητες που παρέχουν στους χρήστες και στις δικλείδες ασφαλείας που έχουν ενσωματωθεί.

6.2.4 Κατασκευή Εντολών Εισαγωγής (Import Command Construction)

Ο κύριος κορμός του κώδικα αφορά την κατασκευή εντολών εισαγωγής δεδομένων στη βάση Neo4j. Ο κώδικας επιτρέπει στον χρήστη να καθορίσει φίλτρα για τα αρχεία που θέλει να εισάγει στη βάση, όπως:

- **Label Filters:** Φίλτρα με βάση τα labels (π.χ. `institution`, `concepts`), που επιτρέπουν την εστίαση μόνο στα αρχεία που σχετίζονται με συγκεκριμένες οντότητες.
- **Part Filters:** Φίλτρα με βάση τα μέρη (parts) των αρχείων, επιτρέποντας την εισαγωγή μόνο συγκεκριμένων κομματιών ενός μεγάλου αρχείου.
- **Date Filters:** Φίλτρα ημερομηνιών, που επιτρέπουν την εισαγωγή δεδομένων που έχουν ενημερωθεί σε συγκεκριμένες ημερομηνίες.

Αυτή η ευελιξία επιτρέπει στον χρήστη να βελτιστοποιήσει τη διαδικασία εισαγωγής, εισάγοντας μόνο τα απαραίτητα δεδομένα και αποφεύγοντας την περιττή επεξεργασία.

6.2.5 Δομή Δεδομένων για Nodes και Relationships (Nodes and Relationships Structure)

Ο κώδικας διαχωρίζει τα αρχεία που αφορούν Nodes από αυτά που αφορούν relationships και κατασκευάζει ξεχωριστές εντολές για την εισαγωγή τους στη βάση Neo4j. Αυτή η προσέγγιση εξασφαλίζει ότι τα δεδομένα εισάγονται με σωστή δομή και ότι τα relationships δημιουργούνται με βάση τα σωστά nodes, μειώνοντας τις πιθανότητες λαθών κατά την εισαγωγή.

6.3 Εισαγωγή OpenAlex Dataset στο Neo4j

Σε αυτό το κεφάλαιο θα αναλύσω τη διαδικασία εισαγωγής του συνόλου δεδομένων από το OpenAlex στη βάση δεδομένων Neo4j. Χρησιμοποιώντας την εντολή `neo4jadmin database import`, πραγματοποιήσα μια πλήρη εισαγωγή δεδομένων, διατηρώντας τη δομή γραφήματος που χαρακτηρίζει τη Neo4j και επιτρέποντας την αποτελεσματική ανάλυση των σχέσεων μεταξύ πλήθους των δεδομένων.

6.3.1 Περιγραφή Εντολής Εισαγωγής

Η εντολή που χρησιμοποιήθηκε ήταν η ακόλουθη:

```
Def construct\_import\_command(  
  Parent\_folder: Path,  
  Headers, Import\_parent,  
  File\_type='csv',  
  Label\_filter=None,  
  Part\_filter=None,  
  Date\_filter=None,  
  Print\_command=False  
  Debug=False,  
)  
\#Construct the import command  
Import\_command = [  
  F"\{import\_parent\}/bin/neo4j-admin",  
  "database", "import", "full",  
  "--skip-duplicate-nodes", "--  
  skip-bad-relationships", "--  
  ignore-empty-strings", "--  
  overwrite-destination", "--  
  multiline-fields=true",  
  "--nad-tolerance=10000000000000000", "--  
  max-off-heap-memory=90%", "--  
  threads=2",  
  "--auto-skip-subsequent-headers=true", "--  
  high-parallel-io=auto", ]
```

6.3.2 Ανάλυση της Εντολής

Φιλτράρισμα και Καθαρισμός Δεδομένων

Χρησιμοποιήθηκαν επιλογές όπως `--skip-duplicate-nodes` και `--skip-bad-relationships` για να αποφευχθούν προβλήματα με διπλότυπα δεδομένα και κακές σχέσεις κατά την εισαγωγή. Αυτό εξασφάλισε ότι τα δεδομένα που εισάγονται είναι όσο το δυνατόν πιο καθαρά και συνεπή, μειώνοντας την ανάγκη για περαιτέρω επεξεργασία ή καθαρισμό.

Διαχείριση Κενών Συμβολοσειρών

Η επιλογή `ignore-empty-strings` χρησιμοποιήθηκε για να αγνοηθούν οι κενές συμβολοσειρές κατά την εισαγωγή. Αυτό είναι ιδιαίτερα χρήσιμο όταν τα δεδομένα περιέχουν κενές τιμές που δεν έχουν νόημα να αποθηκευτούν ως ιδιότητες στους κόμβους ή στις σχέσεις.

Διαχείριση Κόμβων και Σχέσεων

Οι κόμβοι εισήχθησαν στη βάση δεδομένων με βάση τα IDs τους, όπως `PublisherId`, `WorkId`, `SourceId`, κ.λπ., και οι σχέσεις διαμορφώθηκαν με βάση τα προκαθορισμένα headers των CSV αρχείων. Η εντολή επέτρεψε την εισαγωγή διαφορετικών τύπων κόμβων (όπως Συγγραφείς, Ιδρύματα, Έργα κ.λπ.) και τη σύνδεσή τους μέσω σχέσεων, όπως `AuthorHasInstitution` και `WorkHasConcept`.

6.3.3 Παράδειγμα Εξόδου από την Εντολή Εισαγωγής στο Neo4j

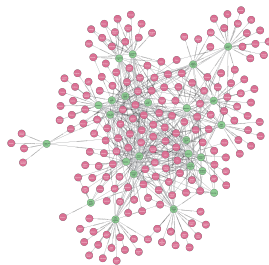
Μετά την εισαγωγή των δεδομένων στη βάση Neo4j, μπορούμε να χρησιμοποιήσουμε το ερώτημα Cypher `MATCH (n)-[r]->(m) RETURN n,r,m` για να ανακτήσουμε τους κόμβους και τις σχέσεις που δημιουργήθηκαν. Παρακάτω παρουσιάζεται ένα παράδειγμα εξόδου με τη μορφή πίνακα: Αυτός ο πίνακας είναι απλώς ένα μικρό υποσύνολο των δεδομένων που εξάγονται με το συγκεκριμένο ερώτημα. Σε μεγάλες βάσεις δεδομένων όπως η υφιστάμενη οπού ο όγκος των δεδομένων της OpenAlex που καταχωρήθηκαν στην Neo4j ήταν τεράστιος, η γραφική διεπαφή δεν μπορεί να παρουσιάσει τόσο μεγάλες εκροές δεδομένων πλήρως, λόγω του τεράστιου όγκου των εγγραφών.

Κόμβος (n)	Σχέση (r)	Κόμβος (m)
InstitutionId:"https://openalex.org/I4210121561",τύπος: "government",updated_date:"2024-05-26",created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C71924100", όνομα: "Medicine"
InstitutionId:"https://openalex.org/I4210121561",τύπος: "government",updated_date:"2024-05-26",created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C86803240", όνομα: "Biology"
InstitutionId:"https://openalex.org/I4210121561",τύπος: "government",updated_date:"2024-05-26",created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C142724271", όνομα: "Pathology"
InstitutionId:"https://openalex.org/I4210121561",τύπος: "government",updated_date:"2024-05-26",created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C126322002", όνομα: "Internalmedicine"
InstitutionId:"https://openalex.org/I4210121561",τύπος: "government", updated_date: "2024-05-26", created_date:"2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C54355233", όνομα: "Genetics"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHas Concept	ConceptId: "https://openalex.org/C203014093", όνομα: "Immunology"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date: "2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C159047783", όνομα: "Virology"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date:"2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C512399662", όνομα: "Familymedicine"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date:"2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C17744445", όνομα: "Politicalscience"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date:"2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId:"https://openalex.org/C185592680", όνομα: "Chemistry"
InstitutionId: "https://openalex.org/I4210121561", τύπος: "government", updated_date: "2024-05-26", created_date:"2022-02-02", όνομα: "Jefferson County"	InstitutionHasConcept	ConceptId: "https://openalex.org/C15744967", όνομα: "Psychology"

Πίνακας 6.1: Υποσύνολο nodes της βάσης που δημιουργήθηκε

6.3.4 Ενδεικτική Απεικόνιση των Αποτελεσμάτων

Οι παρακάτω εικόνες παρουσιάζουν το αποτέλεσμα της εντολής `MATCH (n)-[r]->(m) RETURN n,r,m`, η οποία εκτελείται σε ένα περιβάλλον Neo4j για την απεικόνιση των κόμβων (nodes) και των σχέσεων (relationships) τους. Η εντολή αυτή ανακτά όλους τους κόμβους και τις σχέσεις τους από τη βάση δεδομένων με προσδιορισμένο όριο στα αποτελέσματα ώστε να μην υπερφορτωθεί η Neo4j για να μπορέσει να τα επιστρέψει σε μορφή γραφήματος. Στην πρώτη εικόνα, εμφανίζεται μια ολοκληρωμένη αναπαράσταση των κόμβων και των σχέσεων που ανακτήθηκαν. Η συγκεκριμένη απεικόνιση δείχνει τη δομή της βάσης δεδομένων, συμπεριλαμβανομένων των κόμβων που αντιπροσωπεύουν ιδρύματα, συγγραφείς, έργα, πηγές και άλλες οντότητες. Οι σχέσεις μεταξύ αυτών των κόμβων απεικονίζονται με γραμμές που συνδέουν τους κόμβους, δείχνοντας έτσι τις σχέσεις τους. Ταυτόχρονα, για κάθε κόμβο, με την επιλογή του μπορούμε να περιηγηθούμε και στις ανάλογες ιδιότητες που έχουν εισαχθεί ως πληροφορία με βάση τα “keys” που επιλέξαμε κατά την διάρκεια του φιλτραρίσματος.



Σχήμα 6.1: Γράφημα αναπαράστασης της βάσης και παράδειγμα εξόδου που απεικονίζει τις σχέσεις μεταξύ των κόμβων (Όριο 500)

Στη δεύτερη εικόνα, εστιάζω σε έναν συγκεκριμένο κόμβο, παρουσιάζοντας λεπτομερώς τις σχέσεις που έχει αυτός ο κόμβος με άλλους κόμβους. Στο παράδειγμα αυτό, παρουσιάζεται ένας κόμβος τύπου Institution, ο οποίος συνδέεται με διάφορους κόμβους Concept.



Σχήμα 6.2: Λεπτομερής απεικόνιση των σχέσεων ενός κόμβου τύπου Institution με άλλους κόμβους.

Στην τρίτη εικόνα, παρουσιάζονται αναλυτικά οι ιδιότητες αυτού του κόμβου Institution. Κάθε ιδιότητα που απεικονίζεται είναι αποτέλεσμα της διαδικασίας εισαγωγής που περιελάμβανε τα επιλεγμένα keys.

Node properties

Institution	InstitutionId
create	2022-02-02
d_date	
display	Aspide Medical
_name	(France)
type_id	https://openalex.org/institution-types/company
update	2024-07-
d_date	13T00:49:20.780263
<elementId>	4:9db93172-0464-4bf1-b0bf-f83189698eb1:9
<id>	9
InstitutionId	https://openalex.org/4210094119

Σχήμα 6.3: Λεπτομερής απεικόνιση των ιδιοτήτων του παραπάνω κόμβου τύπου InstitutionId

Η παρακάτω εικόνα, αποτελεί συνέχεια της τρίτης, και απεικονίζει τις ιδιότητες του κόμβου Concept ο οποίος είναι ο προορισμός της σχέσης InstitutionHasConcept που συνδέει τον παραπάνω κόμβο Institution με τον κόμβο Concept. Στο παράδειγμα αυτό, παρουσιάζεται ένας κόμβος τύπου Concept, ο οποίος συνδέεται με τον κόμβο Institution, και αναλυτικά οι ιδιότητες του. Η απεικόνιση αυτή, αν και χρήσιμη για την κατανόηση της δομής και των σχέσεων στη βάση δεδομένων, παρουσιάζει μόνο ένα μικρό υποσύνολο των δεδομένων που υπάρχουν στη βάση. Σε περιπτώσεις μεγάλων βάσεων δεδομένων, όπως στην περίπτωση μας που έγινε εισαγωγή ολόκληρης της OpenAlex στη Neo4j, η γραφική διεπαφή δεν είναι σε θέση να εμφανίσει πλήρως όλα τα δεδομένα και τις σχέσεις τους λόγω του τεράστιου όγκου πληροφοριών.

Node properties

Concept	ConceptId
display	General surgery
_name	
<elementId>	4:9db93172-0464-4bf1-b0bf-f83189698eb1:21229097
<id>	21229097
ConceptId	https://openalex.org/C61434518

Σχήμα 6.4: Λεπτομερής απεικόνιση των ιδιοτήτων του κόμβου τύπου Concept

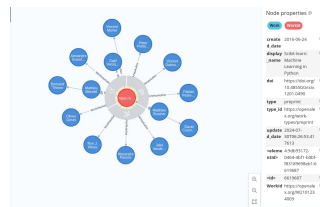
Στην παρακάτω εικόνα, εστιάζω σε έναν διαφορετικό κόμβο, παρουσιάζοντας λεπτομερώς τις σχέσεις που έχει αυτός ο κόμβος με άλλους κόμβους. Στο παράδειγμα αυτό, παρουσιάζεται

ένας κόμβος τύπου Work, ο οποίος συνδέεται με διάφορους κόμβους Authors.



Σχήμα 6.5: Λεπτομερής απεικόνιση των σχέσεων ενός κόμβου τύπου Work με άλλους κόμβους τύπου Authors

Στην επόμενη εικόνα, παρουσιάζονται αναλυτικά οι ιδιότητες αυτού του κόμβου Works. Κάθε ιδιότητα που απεικονίζεται όπως προανέφερα είναι αποτέλεσμα της διαδικασίας εισαγωγής που περιελάμβανε τα επιλεγμένα keys.



Σχήμα 6.6: Λεπτομερής απεικόνιση των ιδιοτήτων του παραπάνω κόμβου τύπου Works στο δεξιά μέρος της εικόνας (Node Properties)

Η παρακάτω εικόνα, αποτελεί συνέχεια της παραπάνω σχήματος, και απεικονίζει τις ιδιότητες του κόμβου Author ο οποίος είναι ο προορισμός της σχέσης WorkHasAuthors που συνδέει τον παραπάνω κόμβο Works με τον κόμβο Authors. Στο παράδειγμα αυτό, παρουσιάζεται ένας κόμβος τύπου Authors, ο οποίος συνδέεται με τον κόμβο Works, και αναλυτικά οι ιδιότητες του. Όπως παραπάνω επικεντρωθήκαμε στις ιδιότητες του κόμβου Works τώρα θα εστιάσουμε στις ιδιότητες του συνδεδεμένου κόμβου Authors στην επόμενη εικόνα.



Σχήμα 6.7: Λεπτομερής απεικόνιση των ιδιοτήτων του κόμβου τύπου Authors (Node Properties)

Τέλος, στην παρακάτω εικόνα γίνεται μια ολική απεικόνιση της βάσης και του αριθμού των nodes, relationships και properties που εισήχθησαν.

Entity Label	Entily Name	Entity Count
1	["InstitutionId", "Institution"]	27.808.142
2	["AuthorId", "Author"]	38.254.456
3	["SourceId", "Source"]	27.682.440
4	["Work", "WorkId"]	56.192.718
5	["Concept", "ConceptId"]	27.867.780

Πίνακας 6.2: Λεπτομερής απεικόνιση του συνόλου της υφιστάμενης βάσης

6.4 VNC Viewer

Για να επιτύχω απομακρυσμένη σύνδεση με το μηχάνημα της σχολής μέσω VPN χρησιμοποιήθηκε κάποιο VNC πρόγραμμα. Στην προκειμένη περίπτωση επιλέχθηκε το VNC Viewer. Για την σωστή εγκατάσταση και παραμετροποίηση του ακολούθησα μια σειρά από βήματα και ρυθμίσεις που περιλάμβαναν εγκαταστάσεις και διαμορφώσεις τόσο στον τοπικό υπολογιστή όσο και στο απομακρυσμένο μηχάνημα. Αρχικά, συνδέθηκα στο δίκτυο της σχολής μέσω του VPN client που μου παρείχε η σχολή, χρησιμοποιώντας του διαπιστευτήριους κωδικούς μου για να αποκτήσω πρόσβαση. Μετά την επιτυχημένη σύνδεση, έκανα έλεγχο επικοινωνίας με το απομακρυσμένο μηχάνημα μέσω της εντολής ping για να επιβεβαιώσω ότι το σύστημα είναι προσβάσιμο. Στη συνέχεια, προχώρησα στην εγκατάσταση του VNC Viewer στον τοπικό μου υπολογιστή, κατεβάζοντας την εφαρμογή από την επίσημη ιστοσελίδα και ακολουθώντας τα βήματα εγκατάστασης. Αφού ολοκληρώθηκε η εγκατάσταση, δημιούργησα μια νέα σύνδεση εισάγοντας την IP διεύθυνση του απομακρυσμένου μηχανήματος στο VNC Viewer. Για να μπορέσω να αλληλοεπιδράσω με την επιφάνεια εργασίας του απομακρυσμένου μηχανήματος κατά την σύνδεση όμως απαιτήθηκαν περαιτέρω ενέργειες. Αναλυτικά, εγκατέστησα τον VNC Server στο απομακρυσμένο μηχάνημα χρησιμοποιώντας το SSH για να συνδεθώ στο σύστημα και να τρέξω τις απαραίτητες εντολές. Η εγκατάσταση έγινε μέσω των εντολών `sudo apt update` και `sudo apt install tightvncserver`, ενώ κατά την πρώτη εκκίνηση του VNC Server ορίστηκε κωδικός πρόσβασης για την απομακρυσμένη σύνδεση. Προσαρμόστηκε επίσης η επιφάνεια εργασίας χρησιμοποιώντας την εντολή `vncserver -geometry 1920x1080 -depth 24` για να επιτευχθεί η σωστή ανάλυση και βάθος χρώματος. Στη συνέχεια, για να γίνει ασφαλής η σύνδεση μέσω VNC, χρησιμοποίησα SSH port forwarding με την εντολή `ssh -L 5901:localhost:5901 -N -f -l username remote_ip_address`, όπου η θύρα 5901 χρησιμοποιήθηκε για την προώθηση της απομακρυσμένης σύνδεσης. Μετά από αυτά τα βήματα, συνδέθηκα με επιτυχία μέσω του VNC Viewer εισάγοντας `asidiropneo4j.iee.ihu.gr:5901` ως ip address, αλλά αντιμετώπισα το πρόβλημα της γκρι οθόνης χωρίς δυνατότητα αλληλεπίδρασης με την επιφάνεια εργασίας. Για να το διορθώσω, επεξεργάστηκα το αρχείο διαμόρφωσης `.xstartup` χρησιμοποιώντας την εντολή `nano /vnc/.xstartup`. Μέσα σε αυτό, πρόσθεσα την εντολή `exec startxfce4` αντί της προκαθορισμένης εντολής, έτσι ώστε να φορτώσει το XFCE περιβάλλον εργασίας. Έπειτα, αποθήκευσα τις αλλαγές και έδωσα τα απαραίτητα δικαιώματα στο αρχείο με την εντολή `sudo chmod +x /vnc/.xstartup`, ακολουθώντας το με μια επανεκκίνηση του VNC Server με την εντολή `vncserver -kill :1` και `vncserver` για να εφαρμοστούν οι αλλαγές. Αφότου προχώρησα στην επανεκκίνηση του VNC Server, η επιφάνεια εργασίας εμφανίστηκε σωστά αλλά η αλληλεπίδραση ήταν ακόμα αδύναμη. Επομένως για να εξασφαλιστεί η σωστή λειτουργία της επιφάνειας εργασίας, προ-

χώρησα στην εγκατάσταση του περιβάλλοντος XFCE και του LightDM display manager μέσω των εντολών `sudo apt install xfce4 xfce4-goodies` και `sudo apt install lightdm` αντίστοιχα. Αυτές οι εγκαταστάσεις επέτρεψαν την ομαλή εκκίνηση και χρήση του γραφικού περιβάλλοντος μέσω VNC, ενώ η τελική επανεκκίνηση του VNC Server διασφάλισε την πλήρη λειτουργικότητα της απομακρυσμένης επιφάνειας εργασίας. Με αυτά τα βήματα ολοκλήρωσα τη διαδικασία και απέκτησα πλήρη πρόσβαση και έλεγχο στο απομακρυσμένο μηχάνημα μέσω του VNC Viewer, επιλύοντας τα προβλήματα εμφάνισης και αλληλεπίδρασης.

Κεφάλαιο 7

Στρατηγικές Βελτιστοποίησης Κώδικα και παραδείγματα εξόδου μέσω Cypher Queries

7.1 Σύνοψη

Στα παραπάνω κεφάλαια αναλύθηκαν όλες οι ενέργειες και πρακτικές που ακολουθήθηκαν για την κατασκευή της βιβλιογραφικής βάσης. Στο κεφάλαιο αυτό θα επικεντρωθούμε στα συμπεράσματα που αποκομίστηκαν από την παραπάνω διαδικασία. Η εργασία είχε ως στόχο την εξαγωγή, την μορφοποίηση και την εισαγωγή των δεδομένων της βιβλιογραφικής βάσης OpenAlex σε μια βάση γράφου όπως η Neo4j. Η υλοποίηση που πραγματοποιήθηκε έγινε στα πλαίσια προπτυχιακής εργασίας επομένως δεν αποκλείεται να υπάρχουν και άλλες βέλτιστες πρακτικές. Τόσο τα κομμάτια κώδικα όσο και τα δημιουργημένα αρχεία παρέχουν την ευελιξία για μελλοντική επέκταση και τροποποίηση. Το γεγονός αυτό διευκολύνει και επιτρέπει την επεξεργασία και τροποποίηση οποιουδήποτε κομματιού της εφαρμογής με ανάλογα πιο σύγχρονης υλοποίησης αποφέροντας παρόμοια αποτελέσματα.

7.2 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία, αυτοματοποιήθηκε η διαδικασία εξαγωγής, μορφοποίησης και εισαγωγής των δεδομένων της OpenAlex. Το σύστημα που επιλέχθηκε για την εισαγωγή ήταν η Neo4j, η οποία αποτελεί μια βάση δεδομένων γραφημάτων. Η δημιουργία της εφαρμογής χωρίστηκε σε δύο τμήματα: το τμήμα δημιουργίας του απαραίτητου κώδικα για την αυτοματοποίηση των διαδικασιών και το τμήμα κατανόησης και δημιουργίας αρχείων συμβατών με το απαιτούμενο format της βάσης γραφημάτων. Η συνένωση αυτών των δύο τμημάτων επέτρεψε την πραγματοποίηση της μαζικής εισαγωγής δεδομένων απευθείας από την OpenAlex στη βάση γραφημάτων Neo4j. Για να αντιμετωπιστεί ο υπερβολικός όγκος δεδομένων, δεν ήταν δυνατή η χρήση των απλών ερωτημάτων της βάσης, επομένως έγινε απαραίτητη η εκμάθηση και χρήση της εντολής `neo4j-admin import`. Η χρήση της εντολής, σε συνδυασμό με τις σωστές παραμέτρους και τα κατάλληλα μορφοποιημένα αρχεία, δημιούργησε μια ολοκληρωμένη βάση με κόμβους, καθώς και τις σχετικές συνδέσεις και ιδιότητές τους. Μέσω αυτής της

διαδικασίας, έγινε δυνατή η αποτελεσματική αναπαράσταση των βιβλιογραφικών δεδομένων της OpenAlex. Η αυτοματοποιημένη και μαζική εισαγωγή των δεδομένων, καθώς και η τμηματική δημιουργία τους, σε συνδυασμό με τη δυνατότητα μέγιστης αξιοποίησης του συστήματος μέσω συγκεκριμένων εντολών και παραμέτρων, αποτέλεσαν σημαντικό παράγοντα για την επιλογή αυτής της προσέγγισης στην υλοποίηση της πτυχιακής αυτής. Η γλώσσα ερωτημάτων της Neo4j, παρά την αδυναμία της για μαζική εισαγωγή δεδομένων, είναι απόλυτα κατάλληλη για χρήση σε σκοπούς ενημέρωσης ή επεξεργασίας της βάσης μέσω απλών ερωτημάτων και εντολών. Βασικός στόχος της πτυχιακής εργασίας ήταν η ολοκληρωμένη αναπαράσταση των βιβλιογραφικών δεδομένων σε μορφή γραφημάτων. Η επίτευξη αυτού του στόχου επέτρεψε την εξαγωγή συμπερασμάτων και την απεικόνιση του συνόλου των πληροφοριών της βάσης μέσω της γραφικής διεπαφής που παρέχει η εφαρμογή Neo4j. Μέσω αυτής της εφαρμογής, μπορεί να αυτοματοποιηθεί η διαδικασία δημιουργίας βάσεων γραφημάτων και να διευκολυνθεί μελλοντικά η ανάλυση δεδομένων.

7.3 Επίλογος

Εν κατακλείδι, η παρούσα πτυχιακή εργασία απέδειξε ότι η τεχνολογία Neo4j μπορεί να αποτελέσει ένα ισχυρό εργαλείο για τη διαχείριση μεγάλων συνόλων δεδομένων, όπως τα βιβλιομετρικά δεδομένα, και να προσφέρει δυνατότητες που υπερβαίνουν αυτές των παραδοσιακών σχεσιακών βάσεων δεδομένων. Η ευελιξία και η προσαρμοστικότητα του μοντέλου γραφημάτων της Neo4j επιτρέπουν την εύκολη παραμετροποίησή του, ενώ οι στρατηγικές βελτιστοποίησης που αναπτύχθηκαν μπορούν να εξασφαλίσουν την αποδοτική διαχείριση δεδομένων σε μεγάλες κλίμακες. Η εργασία αυτή ανοίγει τον δρόμο για περαιτέρω έρευνα και ανάπτυξη στη διαχείριση βιβλιομετρικών δεδομένων, καθώς προσφέρει μια σταθερή βάση για τη μελλοντική ανάπτυξη συστημάτων που θα διευκολύνουν την ανάλυση επιστημονικής πληροφορίας και τη λήψη αποφάσεων στον ακαδημαϊκό χώρο.

Βιβλιογραφία

- [1] Neo4j, *Graphs in Automotive and Manufacturing*, Available: <https://neo4j.com/blog/graphs-in-automotive-and-manufacturing>. Accessed: SEPT. 6, 2024.
- [2] E. Garfield, “The history and meaning of the journal impact factor”, *JAMA*, τόμ. 295, αρθμ. 1, σσ. 90–93, 2006.
- [3] H. F. Moed, *Citation Analysis in Research Evaluation*. Springer Science & Business Media, 2005.
- [4] J. E. Hirsch, “An index to quantify an individual’s scientific research output”, *Proceedings of the National Academy of Sciences*, τόμ. 102, αρθμ. 46, σσ. 16 569–16 572, 2005.
- [5] L. Bornmann και H. D. Daniel, “What do citation counts measure? A review of studies on citing behavior”, *Journal of Documentation*, τόμ. 64, αρθμ. 1, σσ. 45–80, 2008.
- [6] H. F. Moed, *Applied Evaluative Informetrics*. Springer, 2016.
- [7] M. Thelwall, “Dimensions: A competitor to Scopus and the Web of Science?”, *Journal of Informetrics*, τόμ. 14, αρθμ. 4, σ. 101 059, 2020.
- [8] N. D. Bellis, *Bibliometrics and Citation Analysis: From the Science Citation Index to Cybermetrics*. Scarecrow Press, 2009.
- [9] J. Bar-Ilan, “Which h-index?—A comparison of WoS, Scopus, and Google Scholar”, *Scientometrics*, τόμ. 74, αρθμ. 2, σσ. 257–271, 2008.
- [10] J. Priem και H. Piwowar, “OpenAlex: A fully open index of scholarly works, authors, venues, institutions, and concepts”, *arXiv preprint*, τόμ. arXiv:2205.01833, 2022.
- [11] Ideamotive, *Python for Data Analysis*, Available: <https://www.ideamotive.co/blog/python-for-data-analysis>. Accessed: SEPT. 8, 2024.
- [12] P. Wouters, M. Thelwall, K. Kousha και L. Waltman, *Bibliometrics in the Digital Age: Web-Based Methods and the Changing Landscape of Scholarly Impact*. Springer, 2015.
- [13] D. Shotton, “Open citations”, *Nature*, τόμ. 502, αρθμ. 7471, σσ. 295–297, 2012.
- [14] Atomcamp, *Top 10 Python Libraries for Data Scientists*, Available: <https://www.atomcamp.com/top-10-python-libraries-for-data-scientists/>. Accessed: SEPT. 8, 2024.
- [15] R. T. Fielding και R. N. Taylor, “Principled design of the modern web architecture”, *ACM Transactions on Internet Technology (TOIT)*, τόμ. 2, αρθμ. 2, σσ. 115–150, 2000.

- [16] LangChain, *LangChain Neo4j-Cypher Template*, Available: <https://github.com/langchain-ai/langchain/blob/master/templates/neo4j-cypher/README.md>. Accessed: SEPT. 9, 2024.
- [17] Neo4j, *Neo4j Upgrade and Migration Guide*, Available: <https://neo4j.com/docs/upgrade-migration-guide/current/version-5/migration/breaking-changes/>. Accessed: SEPT. 9, 2024.
- [18] J. Dean και S. Ghemawat, “MapReduce: Simplified data processing on large clusters”, 2008.
- [19] M. E. Falagas, E. I. Pitsouni, G. A. Malietzis και G. Pappas, “Comparison of PubMed, Scopus, Web of Science, and Google Scholar: Strengths and weaknesses”, *FASEB Journal*, τόμ. 22, αριθμ. 2, σσ. 338–342, 2008.
- [20] GitHub, *Top 10 Python Libraries for Data Scientists*, Available: <https://www.atomcamp.com/top-10-python-libraries-for-data-scientists/>. Accessed: SEPT. 8, 2024.
- [21] A. Martín-Martín, E. Orduna-Malea, M. Thelwall και E. D. López-Cózar, “Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories”, *Journal of Informetrics*, τόμ. 12, αριθμ. 4, σσ. 1160–1177, 2018.
- [22] R. Cattell, “Scalable SQL and NoSQL data stores”, *ACM SIGMOD Record*, τόμ. 39, αριθμ. 4, σσ. 12–27, 2011.
- [23] A. W. Harzing και S. Alakangas, “Google Scholar, Scopus and the Web of Science: A longitudinal and cross-disciplinary comparison”, *Scientometrics*, τόμ. 106, αριθμ. 2, σσ. 787–804, 2016.
- [24] D. Hicks, P. Wouters, L. Waltman, S. D. Rijcke και I. Rafols, “The Leiden Manifesto for research metrics”, *Nature*, τόμ. 520, αριθμ. 7548, σσ. 429–431, 2015.
- [25] D. Moher, F. Naudet, F. M. I. A. Cristea, J. P. Ioannidis και S. N. Goodman, “Assessing scientists for hiring, promotion, and tenure”, *PLOS Biology*, τόμ. 16, αριθμ. 3, e2004089, 2018.
- [26] V. Larivière και C. R. Sugimoto, *The Journal Impact Factor: A Brief History, Critique, and Discussion of Adverse Effects*. Springer, 2019.
- [27] S. Haustein και V. Larivière, “The use of bibliometrics for assessing research: Possibilities, limitations and adverse effects”, στο *Incentives and Performance*. Springer, 2015, σσ. 121–139.
- [28] J. Bollen, H. V. de Sompel, J. A. Smith και R. Luce, “Toward alternative metrics of journal impact: A comparison of download and citation data”, *Information Processing & Management*, τόμ. 41, αριθμ. 6, σσ. 1419–1440, 2005.
- [29] M. Thelwall και P. Wilson, “Mendeley readership altmetrics for medical articles: An analysis of 45 fields”, *Journal of the Association for Information Science and Technology*, τόμ. 67, αριθμ. 8, σσ. 1962–1972, 2016.
- [30] J. Priem και B. M. Hemminger, “Scientometrics 2.0: New metrics of scholarly impact on the social Web”, *First Monday*, τόμ. 15, αριθμ. 7, 2010.

- [31] D. Fernandes και J. Bernardino, “Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB”, στο *Proc. 8th Int. Conf. Cloud Comput. Services Sci. (CLOSER)*, Accessed: SEPT. 14, 2024, 2018, σσ. 351–354. διεύθν.: <https://pdfs.semanticscholar.org/5164/3300e31bf2c79031c10c8a2007b89c8deb6a.pdf>.
- [32] P. S. Foundation, *Python Documentation: Subprocess Module*, Available: <https://docs.python.org/3/library/subprocess>. Accessed: SEPT. 16, 2024.
- [33] E. Bollen, B. Kuijpers, V. Soliani και A. Vaisman, “Temporal paths in real-world sensor networks”, *ISPRS International Journal of Geo-Information*, τόμ. 10, αρθμ. 2, σ. 3613, 2024.
- [34] Neo4j, *Documentation for Neo4j Graph Algorithms*, Available: <https://neo4j.com/docs/graph-algorithms/current/introduction/>. Accessed: SEPT. 10, 2024.
- [35] “Neo4j Model Documentation”. Accessed: SEPT. 14, 2024. (), διεύθν.: <https://neo4j.com/docs/model/>.
- [36] R. Angles, “A comparison of current graph database models”, στο *Proceedings of the 28th IEEE International Conference on Data Engineering Workshops (ICDEW)*, 2012, σσ. 171–177.
- [37] Neo4j, *Neo4j Manual v1.8*, Available: <http://docs.neo4j.org/chunked/stable/index.html>. Accessed: SEPT. 16, 2024, 2012.
- [38] Neo4j, *Documentation for Neo4j Operations*, Available: <https://neo4j.com/docs/operations-manual/current/installation/>. Accessed: SEPT. 10, 2024.
- [39] D. Fernandes και J. Bernardino, “Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB”, στο *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER)*, Available: <https://pdfs.semanticscholar.org/5164/3300e31bf2c79031c10c8a2007b89c8deb6a.pdf>. Accessed: SEPT. 14, 2024, 2018, σσ. 374–379.
- [40] Neo4j, *Neo4j Community Forum*, Available: <https://community.neo4j.com/>. Accessed: SEPT. 11, 2024.
- [41] Neo4j, *Neo4j Graph Data Science Library Documentation*, Available: <https://neo4j.com/docs/graph-data-science/current/>. Accessed: SEPT. 11, 2024.
- [42] R. Angles, M. Arenas, P. Barceló, J. Hogan, J. Reutter και D. Vrgovc, “Foundations of modern query languages for graph databases”, *ACM Computing Surveys*, τόμ. 50, αρθμ. 5, 68:1–68:40, 2017.
- [43] R. Maan και A. Sharma, “The evolution of Python programming language”, 2020.
- [44] Z. Zhang, H. Zhu, Y. T. M. Wen, Y. Liu και Y. Xiong, “How do Python framework APIs evolve? An exploratory study”, στο *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, σσ. 81–92.
- [45] M. Lutz, *Programming Python*. O’Reilly Media, Inc., 2001.
- [46] J. A. Hoffer, V. Ramesh και H. Topi, *Modern Database Management*. Pearson, 2016.
- [47] E. Bommarito και M. Bommarito, “An empirical analysis of the Python Package Index (PyPI)”, *arXiv preprint*, τόμ. arXiv:1907.11073, 2019.

- [48] A. Nagpal και G. Gabrani, “Python for data analytics, scientific and technical applications”, στο *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, σσ. 140–145.
- [49] F. Pérez, B. E. Granger και J. D. Hunter, “Python: An ecosystem for scientific computing”, *Computing in Sci. & Eng.*, τόμ. 13, αρθμ. 2, σσ. 13–21, 2010.
- [50] W. McKinney, *Python for Data Analysis*. O’Reilly Media, Inc., 2022.
- [51] K. R. Srinath, “Python—the fastest growing programming language”, *Int. Res. J. Eng. Technol.*, τόμ. 4, αρθμ. 12, σσ. 354–357, 2017.
- [52] F. Pérez και B. E. Granger, “IPython: A system for interactive scientific computing”, *Computing in Sci. & Eng.*, τόμ. 9, αρθμ. 3, σσ. 21–29, 2007.
- [53] Neo4j, *Neo4j Desktop Documentation*, Available: <https://neo4j.com/docs/desktop/current/>. Accessed: SEPT. 11, 2024.
- [54] TutorialsPoint, *Neo4j Tutorials*, Available: <https://www.tutorialspoint.com/neo4j/index.htm>. Accessed: SEPT. 12, 2024.
- [55] “Gzip module in Python”. Accessed: SEPT. 11, 2024. (), διεύθν.: <https://www.askpython.com/python-modules/gzip-module-in-python>.
- [56] “Python Documentation: Gzip Module”. Accessed: SEPT. 11, 2024. (), διεύθν.: <https://docs.python.org/3/library/gzip.html>.
- [57] M. Kösters κ.ά., “Revisiting Pathlib”, *login: Usenix Magazine*, τόμ. 42, αρθμ. 2, 2017.
- [58] A. Miller κ.ά., “Automating file operations via Python”, στο *2022 Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, 2022, σσ. 1907–1913.
- [59] “Python Documentation: OS Module”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/os.html>.
- [60] H. P. Langtangen, *Python Scripting for Computational Science*. Springer Berlin Heidelberg, 2008.
- [61] D. Beazley, “Revisiting Pathlib”, *login: Usenix Magazine*, τόμ. 42, αρθμ. 2, 2017.
- [62] “Python Documentation: Shutil Module”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/shutil.html>.
- [63] N. Francis, A. Green, P. Guagliardo κ.ά., “Formal semantics of the language Cypher”, *CoRR*, τόμ. abs/1802.09984, 2018, Available: <https://arxiv.org/abs/1802.09984>. Accessed: Dec. 16, 2024.
- [64] C. Christodoulakis, E. B. Munson, M. Gabel, A. D. Brown και R. J. Miller, “Pytheas: Pattern-based table discovery in CSV files”, *Proc. VLDB Endowment*, τόμ. 13, αρθμ. 12, σσ. 2075–2089, 2020.
- [65] “json—JSON encoder and decoder”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/json.html>.
- [66] “csv—CSV File Reading and Writing”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/csv.html>.
- [67] “Using Python on Windows”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/using/windows.html#the-full-installer>.

-
- [68] “pathlib—Object-oriented filesystem paths”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/pathlib.html>.
- [69] “zipfile—Work with ZIP archives”. Accessed: SEPT. 12, 2024. (), διεύθν.: <https://docs.python.org/3/library/zipfile.html>.
- [70] Neo4j, *Neo4j Community Resources*, Available: <https://neo4j.com/developer/community-resources/>. Accessed: SEPT. 12, 2024.
- [71] I. Explained, *Graph Database Explained*, Available: <https://www.it-explained.com/words/graph-database-explained-explained>. Accessed: SEPT. 4, 2024.
- [72] Neo4j, *Neo4j Admin Import Documentation*, Available: <https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/neo4j-admin-import/>. Accessed: SEPT. 14, 2024.
- [73] ArangoDB, *NoSQL Performance Benchmark 2018*, Available: <https://arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/>. Accessed: SEPT. 4, 2024.
- [74] A. D. Library, “Reconfiguring Edge Computing for Dynamic Workloads”, *Proceedings of the ACM*, τόμ. 13, αρθμ. 4, 2024, Available: <https://dl.acm.org/doi/fullHtml/10.1145/3626203.3670539>. Accessed: SEPT. 3, 2024.