ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Δημιουργία Υπηρεσίας Συστάσεων σε Βάση Δεδομένων Γράφων για Εφαρμογή σε Έξυπνες Πόλεις

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βασίλειος Α. Χαρλαύτης

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Δημιουργία Υπηρεσίας Συστάσεων σε Βάση Δεδομένων Γράφων για Εφαρμογή σε Έξυπνες Πόλεις

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βασίλειος Α. Χαρλαύτης

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Μαρτίου 2017.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Βαρβαρίγος	Δημήτριος Ασκούνης
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2017

.....
Βασίλειος Α. Χαρλαύτης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην επιστήμη των υπολογιστών, μια βάση δεδομένων γράφων (graph database) είναι μια βάση δεδομένων που χρησιμοποιεί δομές γράφων για τα σημασιολογικά ερωτήματα, με τους κόμβους και τις ακμές να αντιπροσωπεύουν και να αποθηκεύουν δεδομένα. Κυρίαρχη έννοια των βάσεων αυτών είναι ο γράφος (με τους κόμβους και τις ακμές του), ο οποίος συσχετίζει άμεσα στοιχεία δεδομένων στον αποθηκευτικό χώρο. Οι σχέσεις αυτές επιτρέπουν στα δεδομένα να συνδεθούν μεταξύ τους άμεσα, και στις περισσότερες περιπτώσεις ανακτώνται με μία μόνο λειτουργία. Αυτό έρχεται σε αντίθεση με τις συμβατικές σχεσιακές βάσεις δεδομένων, όπου οι σχέσεις μεταξύ των δεδομένων είναι αποθηκευμένες στα ίδια τα δεδομένα, και τα ερωτήματα αναζήτησης χρησιμοποιούν την έννοια των JOIN πινάκων για την ανάκτηση και τη συλλογή των σχετικών δεδομένων.

Οι βάσεις δεδομένων γράφων επιτρέπουν, από τον σχεδιασμό τους, την γρήγορη ανάκτηση πολύπλοκων ιεραρχικών δομών που είναι δύσκολο να μοντελοποιηθούν σε σχεσιακά συστήματα. Για το λόγο αυτό έχουν αρχίσει τα τελευταία χρόνια να χρησιμοποιούνται ευρέως σε εφαρμογές κοινωνικών δικτύων και σε συστήματα συστάσεων δεδομένου ότι αποτυπώνουν καλύτερα τις χαλαρές δομές τους και τους δίνουν τη δυνατότητα να εξελίσσονται και να κλιμακώνονται με μεγαλύτερη ευελιξία.

Στα πλαίσια της διπλωματικής αυτής θα υλοποιηθεί μια βάση δεδομένων γράφων βασισμένη στην Neo4j και θα μελετηθεί ο τρόπος που μπορούν να υλοποιηθούν και να διαμορφωθούν ερωτήματα και τεχνικές διάσχισης του γράφου (traversal) που να επιτρέπουν υπηρεσίες προστιθέμενης αξίας που βασίζονται στη μηχανική μάθηση (machine learning). Συγκεκριμένα, θα προσομοιωθεί με Neo4j μια βάση δεδομένων γράφων που θα αποθηκεύει δεδομένα από αισθητήρες που βρίσκονται σε μια έξυπνη πόλη (για παράδειγμα αισθητήρες ατμοσφαιρικής ρύπανσης, θερμοκρασίας, υγρασίας και άλλες πηγές ανοιχτών δεδομένων) και θα υλοποιηθούν ερωτήματα συστάσεων για δυνητικούς χρήστες εφαρμογών που χρησιμοποιούν τη βάση δεδομένων.

Λέξεις Κλειδιά: Βάσεις δεδομένων γράφων, Neo4j, Cypher, Συστήματα συστάσεων, Έξυπνες πόλεις, Ανοιχτά δεδομένα, Συστάσεις πραγματικού χρόνου, Φιλτράρισμα με βάση το περιεχόμενο, Συνεργατικό φιλτράρισμα, NoSQL

Abstract

In computer science, a graph database is a database that uses graph structures for semantic queries, with nodes and edges to represent and store data. A basic concept of these bases is the graph (with nodes and edges), which directly correlates data items in the store. These relationships allow data to be linked together directly, and in most cases, are retrieved in a single operation. This contrasts with conventional relational databases, where relationships between data stored in the data itself, and the search queries for this data using the concept of JOIN tables for the recovery and collection of relevant data.

The graph databases allow, by their design, fast retrieval of complex hierarchical structures that are difficult to model in relational systems. For this reason, they have begun in recent years to be widely used in both Social network applications and Recommender systems as they better reflect their loose structures and enable them to evolve and escalate with more flexibility.

In this thesis, we will build a graph database application using Neo4j and we will study ways to implement queries and traversal techniques that enable value-added services based on machine learning. Specifically, we will build a Neo4j graph database that stores data from sensors located in a smart city (for example air pollution concentrations, temperature, relative humidity and other open data sources) and we will implement recommendations queries for potential users of applications using the database.

Keywords: Graph databases, Neo4j, Cypher, Recommender systems, Smart Cities, Open data, Real time recommendations, Content based filtering, Collaborative filtering, NoSQL

Στους Γονείς μου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά την καθηγήτρια μου κ. Θεοδώρα Βαρβαρίγου που μου εμπιστεύθηκε αυτή τη διπλωματική, δίνοντας μου την ευκαιρία να γνωρίσω το εξαιρετικά ενδιαφέρον πεδίο των βάσεων δεδομένων γράφων. Ένα μεγάλο ευχαριστώ, θα ήθελα να πω στο διδάκτορα ερευνητή Αντώνη Λίτκε και στον υποψήφιο διδάκτορα Γιώργο Παλαιοκρασσά για την πολύτιμη βοήθεια τους σε όλα τα στάδια υλοποίησης της εργασίας. Οφείλω να πω ότι ήταν πάντα διαθέσιμοι, όποτε και αν τους χρειάστηκα, και φρόντιζαν να μου αφιερώνουν αρκετό από το χρόνο τους για να συζητάμε και να επιλύουμε τα προβλήματα που συναντούσα. Φυσικά ένα ευχαριστώ είναι λίγο για τους φίλους που όλα αυτά τα χρόνια βρίσκονται δίπλα μου και που χωρίς αυτούς δεν θα είχα καταφέρει να φτάσω μέχρι εδώ. Τέλος, ένα πολύ μεγάλο ευχαριστώ στα αδέρφια μου και στους γονείς μου που με ανέχονται και με στηρίζουν τόσα χρόνια.

Βασίλειος Χαρλαύτης,
Αθήνα, 3 Μαρτίου 2017.

Πίνακας περιεχομένων

1 *Εισαγωγή*

The world is being re-shaped by the convergence of social, mobile, cloud, big data, community and other powerful forces. The combination of these technologies unlocks an incredible opportunity to connect everything together in a new way and is dramatically transforming the way we live and work.

- Marc Benioff

Ένα ποσοστό μεγαλύτερο από το 50 % του παγκόσμιου πληθυσμού ζει στις μέρες μας σε αστικά κέντρα. Σύμφωνα με το Ταμείο των Ηνωμένων Εθνών για τη Δημογραφία (UNFPA), αυτή η μετακίνηση πληθυσμών από την κυρίως αγροτική στην κυρίως αστική ζωή θα συνεχιστεί για αρκετά χρόνια. Τόσο μεγάλες και περίπλοκες κοινότητες ανθρώπων σχηματίζουν πόλεις και μεγαλουπόλεις που αναπόφευκτα τείνουν να εξελιχθούν σε μέρη με μεγάλες δυσκολίες διαχείρισης των υποδομών τους. Δυσκολίες στη διαχείριση των αποβλήτων, στην έλλειψη επαρκών πόρων, στην ατμοσφαιρική ρύπανση, σε θέματα δημόσιας υγείας, σε θέματα κυκλοφοριακής συμφόρησης και γενικά σε πεπαλαιωμένες και ανεπαρκείς υποδομές, είναι μερικά από τα προβλήματα που δημιουργούνται. Η επείγουσα ανάγκη για την αντιμετώπισή τους οδηγεί πολλές πόλεις παγκοσμίως στην αναζήτηση εξυπνότερων τρόπων διαχείρισης. Συνεπώς, η διασφάλιση βιώσιμων συνθηκών μέσα στο πλαίσιο της ταχύτατης αυτής αστικοποίησης των πληθυσμών απαιτεί καλύτερη κατανόηση της έννοιας των *Έξυπνων Πόλεων.*

Οι *Έξυπνες Πόλεις* είναι ένα όραμα αστικής ανάπτυξης που επιδιώκει την ασφαλή χρησιμοποίηση πληθώρας τεχνολογιών πληροφορικής και επικοινωνιών (ICT) καθώς και του Internet of Things (IoT), για τη διαχείριση των αστικών υποδομών. Παραδείγματα τέτοιων υποδομών είναι τα σχολεία, οι βιβλιοθήκες, οι αστικές συγκοινωνίες, τα νοσοκομεία, οι σταθμοί παραγωγής ηλεκτρικής ενέργειας ή οι εγκαταστάσεις διαχείρισης αποβλήτων, αλλά και πολλές ακόμα κοινωνικές υπηρεσίες. Ο στόχος για τη δημιουργία μιας *Έξυπνης Πόλης* είναι η ικανοποίηση των κοινωνικών αναγκών και η βελτίωση της ποιότητας ζωής των πολιτών. Το κλειδί για την επίτευξη των παραπάνω στόχων είναι η εκμετάλλευση των πληροφοριών που συλλέγονται καθημερινά στις πόλεις. Από τη φύση τους, οι πόλεις συλλέγουν μεγάλους όγκους πληροφοριών για θέματα που τις αφορούν, όπως πληροφορίες για την κίνηση στους δρόμους, για τα επίπεδα ατμοσφαιρικής ρύπανσης, για τη θερμοκρασία, για τα πάρκα και τους χώρους πρασίνου, για την κατανάλωση νερού, αλλά και αρκετές ακόμα. Όλες αυτές οι πληροφορίες συνθέτουν μια πληθώρα από ξεχωριστά σύνολα δεδομένων.

Ο τεράστιος όγκος δεδομένων που δημιουργείται αποτελεί αυτό που αποκαλούμε σήμερα *Μεγάλα Δεδομένα (Big Data)*. Συλλογές, δηλαδή, συχνά περίπλοκων και διαρκώς μεταβαλλόμενων δεδομένων που συγκεντρώνονται από διάφορους αισθητήρες ή υπηρεσίες. Η αξία των μεγάλων δεδομένων έγκειται στο γεγονός ότι μπορούν να αναλυθούν και να διαχειριστούν με τέτοιο τρόπο ώστε να προσφέρουν σημαντική γνώση για τη λήψη καλύτερων αποφάσεων. Η έξυπνη και αποτελεσματική διαχείριση των δεδομένων αυτών αποτελεί σημαντική πρόκληση για κάθε σύγχρονη πόλη.

Ένα πρώτο βήμα για την αποτελεσματική αξιοποίηση των *Μεγάλων* Δεδομένων είναι το άνοιγμα τους και η ελεύθερη δημόσια διανομή τους, δημιουργώντας έτσι τα λεγόμενα *Ανοικτά Δεδομένα (Open Data)*. Τα Ανοιχτά Δεδομένα συμβάλλουν στην ενίσχυση της καινοτομίας και στη δημιουργία αστικών υπηρεσιών που θα είναι διαθέσιμες οποτεδήποτε και οπουδήποτε, καθώς δίνουν τη δυνατότητα σε φορείς ανάπτυξης λογισμικού να τα μετατρέψουν σε χρήσιμες εφαρμογές.

Τα Ανοιχτά Δεδομένα είναι ο δρόμος που οδηγεί στη δημιουργία Έξυπνων Πόλεων με χαμηλό κόστος και υψηλή απόδοση. Ένα, όμως, ακόμα ισχυρότερο εργαλείο για τη δημιουργία καινοτόμων υπηρεσιών είναι ο συνδυασμός διαφορετικών συνόλων ανοικτών δεδομένων. Για παράδειγμα, όταν τα δεδομένα ατμοσφαιρικής ρύπανσης συνδυάζονται με δεδομένα οδικής κυκλοφορίας και συγκοινωνιών ή με δεδομένα καιρικών συνθηκών και ανάλυσης συναισθήματος των πολιτών, μπορούν να οδηγήσουν στη δημιουργία καινοτόμων υπηρεσιών που θα έχουν σημαντικό αντίκτυπο στην διατήρηση της ασφάλειας, της καλής υγείας και της ευημερίας των κατοίκων μιας πόλης.

Υπάρχουν φυσικά και σημαντικές προκλήσεις. Τα κλασικά συστήματα των σχεσιακών βάσεων δεδομένων (RDBMS), με τις αυστηρές δομές τους και τη δυσκολία συσχέτισης των αποθηκευμένων πληροφοριών, δεν μπορούν να ανταπεξέλθουν στη διαχείριση αυτών των περίπλοκων και διαρκώς μεταβαλλόμενων δεδομένων. Η αναζήτηση νέων τεχνολογιών που επιτρέπουν την αποθήκευση και τη διαχείριση τέτοιων δεδομένων, καθώς και η χρησιμοποίηση συστημάτων που παράγουν ωφέλιμη γνώση από αυτά, είναι μερικές από τις προκλήσεις που συναντάμε. Να αναφέρουμε τέλος ότι, η υλοποίηση *Έξυπνων Πόλεων* με κύριο προσανατολισμό την εκμετάλλευση των τεχνολογιών, είναι ένα πρώτο βήμα προς τη σωστή κατεύθυνση, πολλές φορές όμως υστερεί στην αξιοποίηση του ανθρώπινου παράγοντα των πόλεων που είναι ίσως και ο πιο σημαντικός.

1.1 Αντικείμενο της Διπλωματικής

Το κίνητρο της διπλωματικής αυτής είναι η δημιουργία ενός καινοτόμου οικοσυστήματος στο οποίο οι πολίτες θα αλληλεπιδρούν άμεσα με τα ανοικτά δεδομένα που παράγονται από μια *έξυπνη πόλη*, ανοίγοντας έτσι νέες δυνατότητες για τη δημιουργία υπηρεσιών που θα έχουν ως σκοπό τόσο την άμεση ικανοποίηση των ίδιων των πολιτών όσο και την ανάδειξη μεθόδων για τη γενική βελτίωση της ποιότητας ζωής τους.

Στη συγκεκριμένη εργασία, θα χρησιμοποιήσουμε ανοικτά δεδομένα ατμοσφαιρικής ρύπανσης και καιρικών συνθηκών που βρήκαμε για το Camden Borough του Λονδίνου, σε συνδυασμό με δεδομένα καθημερινών δραστηριοτήτων δυνητικών χρηστών, με απώτερο σκοπό τη δημιουργία υπηρεσίας συστάσεων πραγματικού χρόνου.

Όπως αναφέρθηκε στην εισαγωγή, τα κλασικά συστήματα των σχεσιακών βάσεων δεδομένων αδυνατούν να ανταπεξέλθουν στις απαιτήσεις διαρκώς εξελισσόμενων εφαρμογών που πυρήνας της λειτουργίας τους είναι η ανάλυση και η διαχείριση των σχέσεων που συνδέουν τα δεδομένα τους. Ένα παράδειγμα της αδυναμίας διαχείρισης ανάλογων εφαρμογών από τις σχεσιακές βάσεις δεδομένων, μπορούμε να βρούμε και στην εργασία *"Smart City: A Rule-based Tourist Recommendation System"* των Ago Luberg, Tanel Tammet και Priit Hivr, όπου αναφέρεται χαρακτηριστικά:

"The reasoning engine employed by the system has to handle a large amount of data …"

"Fetching all this data from the conventional relational database takes too much time. In particular, it is hopeless to run the reasoning engine by querying all the potential premises of rules from the conventional database for each rule application during derivation …"

"Hence we employ a principle that all - or relevant for this day - data from the relational database is loaded into the fast in-memory database handled by the prover."

Έτσι, αποφασίσαμε για τους σκοπούς της εργασίας να χρησιμοποιήσουμε τη βάση δεδομένων γράφων Neo4j. Η Neo4j είναι μια βάση δεδομένων που παρέχει ευελιξία ανάπτυξης (schema free) και εκμεταλλεύεται στο έπακρο τις σχέσεις μεταξύ των δεδομένων της.

Στόχος της εργασίας μας είναι να μελετήσουμε τρόπους και τεχνικές μοντελοποίησης των διαθέσιμων δεδομένων για την εισαγωγή τους στη βάση δεδομένων γράφων Neo4j, καθώς και τεχνικών διάσχισης του γράφου που θα ευνοούν τη δημιουργία υπηρεσίας συστάσεων πραγματικού χρόνου.

1.2 Οργάνωση κειμένου

Η διπλωματική αποτελείται από τα ακόλουθα κεφάλαια:

2 Βάσεις Δεδομένων Γράφων και Neo4j

Γίνεται παρουσίαση των βάσεων δεδομένων γράφων και των πλεονεκτημάτων τους έναντι των κλασικών σχεσιακών βάσεων δεδομένων.

3 Ανάλυση Δεδομένων

Γίνεται αναλυτική παρουσίαση των δεδομένων που θα χρησιμοποιηθούν και των χαρακτηριστικών των δεδομένων αυτών.

4 Μοντελοποίηση Δεδομένων

Παρουσιάζεται η πορεία μοντελοποίησης των δεδομένων σε δομές δεδομένων γράφων. Γίνεται επίσης αναφορά σε προβλήματα που συναντάμε κατά τη μοντελοποίηση και σε τεχνικές αντιμετώπισής τους.

5 Υλοποίηση Βάσης Δεδομένων Γράφων

Παρουσιάζεται η διαδικασία δημιουργίας της βάσης δεδομένων γράφων με αναφορές σε κομμάτια κώδικα cypher.

6 Έλεγχος Υλοποίησης

Γίνεται έλεγχος ορθής υλοποίησης της βάσης δεδομένων μέσω αξιολόγησης των απαντήσεων που δίνει σε τετριμμένα ερωτήματα.

7 Δημιουργία Συστάσεων

Παρουσιάζεται η πορεία δημιουργίας ερωτημάτων συστάσεων προς τη βάση δεδομένων γράφων και η αξιολόγησή τους.

8 Επίλογος

Γίνεται μια σύνοψη της διπλωματικής.

2 *Βάσεις Δεδομένων Γράφων και Neo4j*

Ο κόσμος μας αποτελείται από μια πληθώρα αλληλένδετων πραγμάτων. Όλα τα πράγματα συνδέονται μεταξύ τους, κάποτε με εμφανείς και άλλοτε με λιγότερο εμφανείς σχέσεις. Αυτό φυσικά έχει επιπτώσεις στον τρόπο με τον οποίο διαχειριζόμαστε την πραγματικότητα στα υπολογιστικά συστήματα. Επηρεάζει τον τρόπο με τον οποίο αποθηκεύουμε τα δεδομένα μας σε ένα σύστημα βάσεων δεδομένων, αλλά και τον τρόπο που αλληλοεπιδρούμε με το σύστημα για την εξυπηρέτηση διαφορετικών εφαρμογών. Για αρκετές δεκαετίες, γινόταν προσπάθεια διαχείρισης συνόλων δεδομένων υψηλού βαθμού συσχέτισης μα τη χρήση των σχεσιακών βάσεων δεδομένων. Τα τελευταία χρόνια, όμως, παρατηρείται μια τάση χρησιμοποίησης NoSQL βάσεων δεδομένων με αρκετά έντονο το ενδιαφέρον για τις βάσεις δεδομένων γράφων (graph databases). Η τάση αυτή προέρχεται κυρίως από τη μεγάλη εμπορική επιτυχία εταιριών όπως η Google, το Facebook και το Twitter, που όλες έχτισαν τα επιχειρηματικά μοντέλα τους πάνω σε ιδιόκτητες τεχνολογίες γράφων.

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα πλεονεκτήματα που προσφέρει η χρησιμοποίηση βάσεων δεδομένων γράφων έναντι των κλασικών σχεσιακών βάσεων δεδομένων, σχετικά με την ανάπτυξη εφαρμογών που διαχειρίζονται πολύπλοκα και διαρκώς μεταβαλλόμενα σύνολα δεδομένων. Θα ξεκινήσουμε όμως με μια σύντομη αναφορά στα κλασικά σχεσιακά συστήματα βάσεων δεδομένων.

2.1 Σχεσιακά συστήματα βάσεων δεδομένων

Οι σχεσιακές βάσεις δεδομένων σχεδιάστηκαν για να κωδικοποιούν έντυπα και δομές πινάκων, κάτι το οποίο κάνουν εξαιρετικά καλά εδώ και αρκετά χρόνια. Συναντάνε όμως σοβαρά προβλήματα όταν ο όγκος των δεδομένων που αποθηκεύουν αυξάνει δραματικά ή όταν καλούνται να μοντελοποιήσουν τις πολύπλοκες σχέσεις που συνδέουν τα δεδομένα στον πραγματικό κόσμο. Ενώ, λοιπόν, τα σχεσιακά συστήματα βάσεων δεδομένων αποδίδουν – και θα συνεχίσουν ενδεχομένων να αποδίδουν – εξαιρετικά καλά για ένα σύνολο εφαρμογών, στην ενότητα αυτή θα αναφερθούμε στα σημαντικότερα σημεία υστέρησης που παρουσιάζουν, οδηγώντας έτσι στην αναζήτηση ενναλακτικών συστημάτων διαχείρισης των δεδομένων.

Ένα από τα προβλήματα που συναντάμε στα σχεσιακά συστήματα βάσεων δεδομένων είναι η δυσκολία διαχείρισης πολύ μεγάλου όγκου δεδομένων. Όσο οι πίνακες των σχεσιακών βάσεων δεδομένων μεγαλώνουν, τόσο η απόδοση των ερωτημάτων και η γρήγορη απόκριση του συστήματος μειώνονται. Φυσικά, για ένα μεγάλο αριθμό εφαρμογών το συγκεκριμένο ζήτημα μπορεί να μην αποτελεί πρόβλημα, αλλά για τη διαχείριση του τεράστιου όγκου των *ανοικτών δεδομένων* (με τα οποία θα ασχοληθούμε και στην παρούσα διπλωματική) είναι ένα ζήτημα που πρέπει να ληφθεί σοβαρά υπόψη.

Σημαντικό πρόβλημα για τα σχεσιακά συστήματα βάσεων δεδομένων, αποτελεί και η δυσκολία διαχείρισης των σχέσεων μεταξύ των δεδομένων. Όσο πολυπλοκότερο είναι το πεδίο που θέλουμε να μοντελοποιήσουμε, τόσο πιο πολύπλοκα γίνονται τα σχεσιακά μοντέλα και η δυσκολία διαχείρισης των σχεσιακών συστημάτων αυξάνεται δραματικά. Συγκεκριμένα, όσο περισσότερες είναι οι συνδέσεις (joins) μεταξύ πινάκων που απαιτούνται για να απαντήσουν στα ερωτήματα κάποιου χρήστη, τόσο πιο πολύπλοκη και επίπονη γίνεται η διαδικασία διαχείρισής τους από τα σχεσιακά συστήματα. Υπάρχει ένα πραγματικό όριο στον αριθμό συνδέσεων (join operations) τους οποίους μπορεί να διαχειριστεί αποδοτικά ένα σχεσιακό σύστημα. Εάν υπερβούμε αυτό τον αριθμό τότε το σύστημα αδυνατεί να ανταποκριθεί.

Επίσης ένα σημαντικό μειονέκτημα των σχεσιακών βάσεων δεδομένων, είναι το ότι επιβάλλουν την ύπαρξη ενός πολύ συγκεκριμένου σχήματος (schema) πριν ακόμα αρχίσουμε να εισάγουμε τα δεδομένα μας σε αυτές. Σε πολλές όμως περιπτώσεις, το πεδίο μιας εφαρμογής είναι πολύ δύσκολο να μοντελοποιηθεί με τη χρήση ενός αυστηρού μοντέλου που θα πρέπει να ισχύει για όλα τα δομικά στοιχεία του πεδίου. Θα θέλαμε δηλαδή, την ύπαρξη ενός πιο ευέλικτου μοντέλου που θα μας παρέχει μεγαλύτερη ελευθερία κινήσεων κατά την ανάπτυξη μιας εφαρμογής.

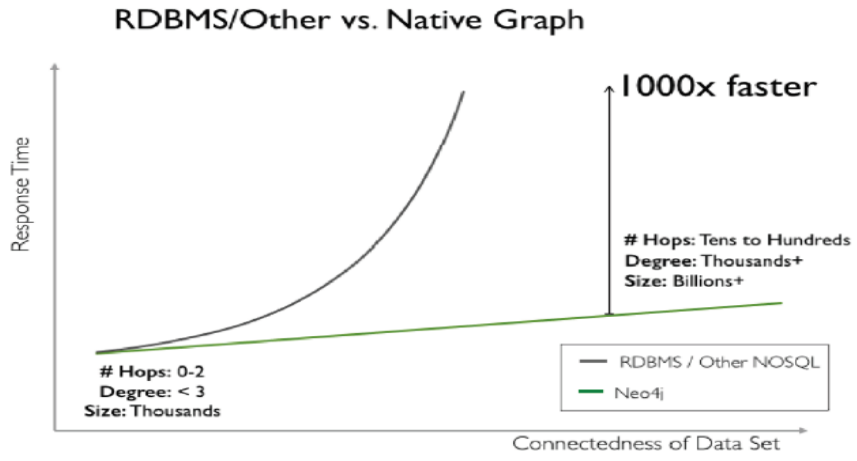
Από όσα αναφέρθηκαν παραπάνω, συμπεραίνουμε πως οι σχεσιακές βάσεις δεδομένων – αν και κατάλληλες για ένα σύνολο εφαρμογών – δεν μπορούν να ανταποκριθούν αποδοτικά σε απαιτήσεις εφαρμογών που διαχειρίζονται ένα μεγάλο όγκο πολύπλοκων και διαρκώς μεταβαλλόμενων δεδομένων. Οδηγούμαστε έτσι, στην αναζήτηση λύσεων σε εναλλακτικές τεχνολογίες βάσεων δεδομένων. Σε τεχνολογίες που θα μας δίνουν μεγαλύτερη ελευθερία κατά την πορεία ανάπτυξης των εφαρμογών και θα εκμεταλλεύονται στο έπακρο τις σχέσεις που συνδέουν τα δεδομένα μας.

2.2 Βάσεις δεδομένων γράφων

Στην προηγούμενη ενότητα είδαμε ότι οι σχεσιακές βάσεις δεδομένων αδυνατούν να διαχειριστούν αποδοτικά τις συνδέσεις μεταξύ των δεδομένων. Συγκεκριμένα, στα σχεσιακά συστήματα τα δεδομένα κατακερματίζονται και αποθηκεύονται σε συλλογές δεδομένων ασύνδετες μεταξύ τους. Η σύνδεση των δεδομένων γίνεται κατά την εκτέλεση των ερωτημάτων που στέλνονται προς τη βάση δεδομένων, με χρήση των πινάκων σύνδεσης (join tables) ή με τη χρήση ξένων κλειδιών (foreign keys), με αποτέλεσμα τη μείωση της απόδοσης του συστήματος και τη δυσκολία διαχείρισης πολύπλοκων ερωτημάτων. Αντίθετα, στις βάσεις δεδομένων γράφων οι συνδέσεις μεταξύ των δεδομένων αποθηκεύονται αυτούσιες στη βάση δεδομένων. Όλες οι συνδέσεις που υπάρχουν στον πραγματικό κόσμο, αποθηκεύονται ως σχέσεις μεταξύ των δεδομένων της βάσης δεδομένων.

Ένας από τους σημαντικότερους λόγους για να επιλέξει κάποιος τη χρησιμοποίηση των βάσεων δεδομένων γράφων είναι η μεγάλη απόδοση που προσφέρουν κατά τη διαχείριση διασυνδεδεμένων δεδομένων. Ενώ στις σχεσιακές βάσεις δεδομένων οι συνδέσεις μεταξύ πινάκων μειώνουν την απόδοση του συστήματος καθώς ο όγκος των αποθηκευμένων δεδομένων αυξάνει, στις βάσεις δεδομένων γράφων η απόδοση παραμένει σχεδόν σταθερή. Αυτό οφείλεται στο γεγονός ότι τα ερωτήματα προς τις βάσεις δεδομένων γράφων αφορούν ένα πολύ συγκεκριμένο κομμάτι του γράφου. Ως αποτέλεσμα, ο χρόνος εκτέλεσης κάθε ερωτήματος είναι ανάλογος του μεγέθους του γράφου που θα διατρέξει το ερώτημα αυτό και δεν σχετίζεται με το συνολικό μέγεθος ολόκληρου του γράφου.

Στην παρακάτω εικόνα βλέπουμε πως η απόδοση της Neo4j παραμένει σχεδόν σταθερή καθώς αυξάνει ο όγκος και η διασύνδεση των αποθηκευμένων δεδομένων. Το αντίθετο παρατηρούμε να συμβαίνει με τα σχεσιακά συστήματα ή με άλλα NoSQL συστήματα βάσεων δεδομένων:



Ένα ακόμα χαρακτηριστικό παράδειγμα στη διαφορά απόδοσης των βάσεων δεδομένων γράφων (συγκεκριμένα της Neo4j) έναντι των σχεσιακών βάσεων δεδομένων παρουσιάζεται στο βιβλίο των Partner και Vukotic, *"Neo4j in Action"*. Συγκεκριμένα, οι Partner και Vukotic κάνουν ένα πείραμα θέλοντας να δείξουν την μεγάλη διαφορά στην ταχύτητα απόκρισης των βάσεων δεδομένων γράφων έναντι των σχεσιακών συστημάτων, όταν αυτά πρέπει να διαχειριστούν διασυνδεδεμένα δεδομένα.

Στο πείραμά τους αναζητούν φίλους φίλων μέχρι ένα βάθος 5 επιπέδων για ένα κοινωνικό δίκτυο 1,000,000 ανθρώπων, όπου κάθε άνθρωπος έχει κατά μέσο όρο 50 φίλους. Τα αποτελέσματα του πειράματος φαίνονται στον πίνακα που ακολουθεί:

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Παρατηρούμε ότι για βάθος δύο επιπέδων (οι φίλοι – των – φίλων μου), τόσο η σχεσιακή βάση δεδομένων όσο και η βάση δεδομένων γράφων ανταποκρίνονται αρκετά καλά και δεν είμαστε σε θέση να αντιληφθούμε κάποια πρακτική διαφορά. Ωστόσο, όταν φτάνουμε σε βάθος τριών επιπέδων (οι φίλοι – των – φίλων – των – φίλων μου), είναι πλέον ξεκάθαρο ότι η σχεσιακή βάση δεδομένων δεν μπορεί να ανταπεξέλθει στις απαιτήσεις ενός συστήματος πραγματικού χρόνου. Τα 30 δευτερόλεπτα είναι απαράδεκτος χρόνος για ένα τέτοιο σύστημα. Αντίθετα, η Neo4j εξακολουθεί να παρουσιάζει ένα πολύ μικρό χρόνο απόκρισης (168 ms). Σε βάθος τεσσάρων επιπέδων η σχεσιακή βάση φαίνεται ότι είναι ουσιαστικά άχρηστη ενώ η Neo4j βρίσκεται πλέον στα όρια του αποδεκτού χρόνου για ένα σύστημα πραγματικού χρόνου. Τέλος, σε βάθος πέντε επιπέδων η σχεσιακή βάση δεδομένων δεν τερματίζει ποτέ ενώ η Neo4j ολοκληρώνει το ερώτημα σε 2.1 δευτερόλεπτα, έχοντας όμως διατρέξει σχεδόν ολόκληρο το γράφο του 1,000,000 ανθρώπων.

Θα κλείσουμε την ενότητα αυτή, παρουσιάζοντας ένα ακόμα σημαντικό χαρακτηριστικό των βάσεων δεδομένων γράφων. Σε αντίθεση με τις σχεσιακές βάσεις δεδομένων που απαιτούν ένα αυστηρά προκαθορισμένο σχήμα για την αποθήκευση των δεδομένων τους, οι βάσεις δεδομένων γράφων δίνουν μεγάλη ευελιξία επιτρέποντας την ανάπτυξη εφαρμογών σταδιακά και με βάση τις απαιτήσεις που προκύπτουν κατά την πορεία της δημιουργίας μιας εφαρμογής. Οι γράφοι ευνοούν από τη φύση τους τη σταδιακή εξέλιξη ενός μοντέλου καθώς υπάρχει η δυνατότητα προσθήκης νέων τύπων σχέσεων ή κόμβων, στον ήδη υπάρχοντα γράφο, χωρίς να επηρεαστούν τα υπάρχοντα ερωτήματα ή η λειτουργικότητα της εφαρμογής. Έτσι, η ευελιξία μιας βάσης δεδομένων γράφων μας επιτρέπει να μην χρειάζεται να κάνουμε εξαντλητική μοντελοποίηση του πεδίου της εφαρμογής πριν καν αρχίσουμε τη διαδικασία της υλοποίησης. Αντίθετα μάλιστα, μας δίνει τη δυνατότητα σταδιακής εξέλιξης του μοντέλου και διαρκούς προσαρμογής του για την εξυπηρέτηση των πραγματικών αναγκών της εφαρμογής.

2.3 Neo4j και περιπτώσεις χρήσης

Για την υλοποίηση της εργασίας μας θα χρησιμοποιήσουμε τη βάση δεδομένων γράφων Neo4j. Η Neo4j είναι η δημοφιλέστερη βάση δεδομένων γράφων και βρίσκεται στην πρώτη θέση της κατάταξης του DB-Engines για την κατηγορία των συστημάτων διαχείρισης βάσεων δεδομένων γράφων.

Η Neo4j χρησιμοποιεί το μοντέλο γράφων με ετικέτες και ιδιότητες και συνοδεύεται από μια εξαιρετική γλώσσα ερωτημάτων τη cypher. Τόσο το μοντέλο γράφων με ετικέτες και ιδιότητες όσο και η cypher παρουσιάζονται αναλυτικά στα κεφάλαια που ακολουθούν.

Υπάρχει μια πληθώρα εφαρμογών στις οποίες απευθύνεται η Neo4j και υπόσχεται σημαντική βελτίωση της απόδοσης και του τρόπου διαχείρισης τους. Μερικές από τις εφαρμογές παρουσιάζονται παρακάτω:

- Συστήματα συστάσεων πραγματικού χρόνου (Real time recommendation engine)
- Κοινωνικά δίκτυα (Social network)
- Ανίχνευση απάτης (Fraud detection)
- Αναζήτηση με χρήση δομών γράφων (Graph based search)
- Διαχείριση κρίσιμων δεδομένων (Master data management)

Από την παραπάνω λίστα, τα συστήματα συστάσεων είναι πιθανώς η δημοφιλέστερη εφαρμογή της Neo4j και για το σκοπό αυτό θα τη χρησιμοποιήσουμε στην παρούσα εργασία. Με την επιλογή της Neo4j έχουμε τη δυνατότητα να μοντελοποιήσουμε ένα άγνωστο πεδίο σταδιακά, χωρίς να χρειάζεται να σπαταλήσουμε ατελείωτες ώρες για την εξαντλητική μοντελοποίηση του πεδίου πριν την έναρξη των διαδικασιών υλοποίησης. Επίσης, με την ταχύτατη διάσχιση των σχέσεων που παρέχεται από την Neo4j, μπορούμε να εκτελούμε πολύπλοκα ερωτήματα σε πραγματικό χρόνο, εξάγοντας χρήσιμα συμπεράσματα, από τα πιο πρόσφατα διαθέσιμα δεδομένα, για την παραγωγή έγκυρων και αξιόπιστων συστάσεων προς τους χρήστες της εφαρμογής.

3 Ανάλυση Δεδομένων

Για την υλοποίηση της βάσης δεδομένων χρησιμοποιήθηκαν δεδομένα ατμοσφαιρικής ρύπανσης και καιρικών συνθηκών που βρήκαμε για το Camden Borough του Λονδίνου από το London Air Quality Network. Στο κεφάλαιο αυτό θα γίνει αναλυτική παρουσίαση των δεδομένων και των χαρακτηριστικών τους.

3.1 Ατμοσφαιρική ρύπανση (air pollution)

Ατμοσφαιρική ρύπανση είναι η απελευθέρωση ρύπων (σωματιδίων και επιβλαβών αερίων) στην ατμόσφαιρα . Οι εκπομπές των ρύπων αυτών μπορεί να προέρχονται από φυσικά αίτια ή να είναι αποτέλεσμα της ανθρώπινης δραστηριότητας και έχουν επιπτώσεις στην ανθρώπινη υγεία.

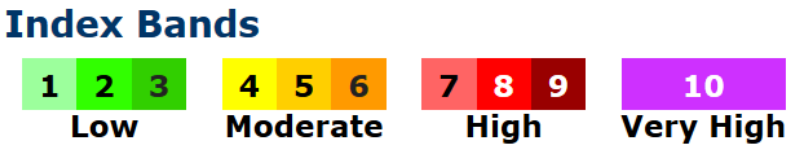
Στο Λονδίνο παρουσιάζουν ενδιαφέρον πέντε βασικοί ρύποι:

- το διοξείδιο του αζώτου (NO2)
- το όζον (O3)
- τα σωματίδια PM10
- τα σωματίδια PM2.5 και
- το διοξείδιο του θείου (SO2)

Κάθε ένας από τους ρύπους αυτούς προέρχεται από διαφορετικές πηγές, έχει διαφορετικές επιπτώσεις στην υγεία και διαφορετική χημική συμπεριφορά, κάνοντας έτσι την προσπάθεια κατανόησης και ελέγχου της ατμοσφαιρικής ρύπανσης αρκετά πολύπλοκη.

Ο δείκτης που χρησιμοποιείται για τη μέτρηση της ατμοσφαιρικής ρύπανσης καλείται Daily Air Quality Index – DAQI και με βάση το Τμήμα Τροφίμων, Περιβάλλοντος και Αγροτικών Υποθέσεων (DEFRA) του Ηνωμένου Βασιλείου, χωρίζεται σε τέσσερα επίπεδα: από Low (1) έως Very High (10).

Οι τιμές που παίρνει ο δείκτης DAQI αντιστοιχούν σε μια κλίμακα από το 1 μέχρι το 10 όπως φαίνεται στην εικόνα που ακολουθεί:



Εικόνα 3.1 Ημερήσιος δείκτης ατμοσφαιρικής ρύπανσης

και υποδεικνύουν το επίπεδο της ατμοσφαιρικής ρύπανσης σε μια περιοχή. Οι τιμές αυτές συνδέονται με συμβουλές για την υγεία και το επίπεδο δραστηριότητας των κατοίκων της περιοχής, οι οποίες συνοψίζονται στον ακόλουθο πίνακα:

Επίπεδο Ατμοσφαιρικής Ρύπανσης	Τιμή του δείκτη DAQI	Ευπαθείς Ομάδες	Γενικός Πληθυσμός
Low	1 – 3	Απολαύστε τις καθημερινές σας δραστηριότητες.	Απολαύστε τις καθημερινές σας δραστηριότητες.
Moderate	4 – 6	Ενήλικες και παιδιά με αναπνευστικά προβλήματα, και ενήλικες με καρδιακά προβλήματα που παρουσιάζουν συμπτώματα δυσκολίας, πρέπει να μειώσουν τις επίπονες δραστηριότητες, ειδικά σε εξωτερικούς χώρους.	Απολαύστε τις καθημερινές σας δραστηριότητες.
High	7 – 9	Ενήλικες και παιδιά με αναπνευστικά προβλήματα, και ενήλικες με καρδιακά προβλήματα πρέπει να μειώσουν τις επίπονες δραστηριότητες, ειδικά σε εξωτερικούς χώρους και ιδιαίτερα αν παρουσιάζουν συμπτώματα δυσκολίας. Άτομα με άσθμα πιθανώς θα χρειαστεί να κάνουν πιο συχνά τις εισπνοές τους. Οι ηλικιωμένοι πρέπει επίσης να μειώσουν τη σωματική δραστηριότητα.	Οποιοσδήποτε παρουσιάζει κάποια δυσφορία, όπως πόνο στα μάτια, βήχα ή πονόλαιμο πρέπει να μειώσει της δραστηριότητες του, ειδικά σε εξωτερικούς χώρους.
Very High	10	Ενήλικες και παιδιά με αναπνευστικά προβλήματα, ενήλικες με καρδιακά προβλήματα και ηλικιωμένοι, πρέπει να μειώσουν τις επίπονες δραστηριότητες. Άτομα με άσθμα πιθανώς θα χρειαστεί να κάνουν πιο συχνά τις εισπνοές τους.	Μειώστε οποιαδήποτε σωματική δραστηριότητα ειδικά σε εξωτερικούς χώρου, ιδιαίτερα αν παρουσιάζεται συμπτώματα όπως βήχας ή πονόλαιμος.

Πίνακας 3.1 Συμβουλές υγείας ανάλογα με το επίπεδο ατμοσφαιρικής ρύπανσης

Είναι χρήσιμο να αναφέρουμε ότι ο υπολογισμός της τιμής του DAQI εξαρτάται από τις τιμές των συγκεντρώσεων των διαφόρων ρύπων που υπάρχουν στην ατμόσφαιρα. Τα άνω και κάτω όρια των συγκεντρώσεων για κάθε τιμή του δείκτη είναι διαφορετικά ανάλογα με το είδος του ρύπου. Τα όρια αυτά για τους πέντε ρύπους που θα απασχολήσουν την εργασία μας παρουσιάζονται στους πίνακες που ακολουθούν:

Index	1	2	3	4	5	6	7	8	9	10
Band	Low	Low	Low	Moderate	Moderate	Moderate	High	High	High	Very High
µg/m³	0-67	68-134	135-200	201-267	268-334	335-400	401-467	468-534	535-600	601 or more

Πίνακας 3.2 Δείκτης ατμοσφαιρικής ρύπανσης και όρια συγκεντρώσεων για το NO2

Index	1	2	3	4	5	6	7	8	9	10
Band	Low	Low	Low	Moderate	Moderate	Moderate	High	High	High	Very High
μg/m³	0-33	34-66	67-100	101-120	121-140	141-160	161-187	188-213	214-240	241 or more

Πίνακας 3.3 Δείκτης ατμοσφαιρικής ρύπανσης και όρια συγκεντρώσεων για το O3

Index	1	2	3	4	5	6	7	8	9	10
Band	Low	Low	Low	Moderate	Moderate	Moderate	High	High	High	Very High
μg/m³	0-16	17-33	34-50	51-58	59-66	67-75	76-83	84-91	92-100	101 or more

Πίνακας 3.4 Δείκτης ατμοσφαιρικής ρύπανσης και όρια συγκεντρώσεων για τα σωματίδια PM10

Index	1	2	3	4	5	6	7	8	9	10
Band	Low	Low	Low	Moderate	Moderate	Moderate	High	High	High	Very High
μgm^{-3}	0-11	12-23	24-35	>36-41	>42-47	>48-53	54-58	59-64	65-70	71 or more

Πίνακας 3.5 Δείκτης ατμοσφαιρικής ρύπανσης και όρια συγκεντρώσεων για τα σωματίδια PM2.5

Index	1	2	3	4	5	6	7	8	9	10
Band	Low	Low	Low	Moderate	Moderate	Moderate	High	High	High	Very High
μg/m³	0-88	89-177	178-266	267-354	355-443	444-532	533-710	711-887	888-1064	1065 or more

Πίνακας 3.6 Δείκτης ατμοσφαιρικής ρύπανσης και όρια συγκεντρώσεων για το SO2

Στους πίνακες αυτούς βλέπουμε ότι μονάδα μέτρησης των συγκεντρώσεων των ρύπων είναι τα μg/m^3 και ότι τα πιθανά επίπεδα συγκέντρωσης κάθε ρύπου χωρίζονται σε δέκα επιμέρους διαστήματα. Κάθε διάστημα αντιστοιχεί σε διαφορετική τιμή του ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης (DAQI). Η τελική τιμή του δείκτη λαμβάνεται ως ίση με τη μεγαλύτερη τιμή στην οποία έχει αντιστοιχιστεί η συγκέντρωση κάποιου από τους πέντε ρύπους που μελετάμε. Αν για παράδειγμα κάποια ώρα έχουμε μετρήσεις των συγκεντρώσεων και των πέντε ρύπων, τότε έχουμε πέντε διαφορετικές τιμές για το δείκτη DAQI. Η τελική τιμή του δείκτη θα ισούται με τη μεγαλύτερη από τις επιμέρους τιμές.

3.1.1 Δεδομένα ατμοσφαιρικής ρύπανσης

Τα δεδομένα ατμοσφαιρικής ρύπανσης που χρησιμοποιήθηκαν για τη δημιουργία της βάσης χωρίζονται σε δύο κατηγορίες:

1. Δεδομένα σε μορφή csv αρχείων που περιέχουν τις πληροφορίες τον πινάκων του δείκτη ατμοσφαιρικής ρύπανσης σε σχέση με τη συγκέντρωση κάθε ρύπου (πίνακες 3.2 – 3.6).
2. Δεδομένα σε μορφή csv αρχείων που περιέχουν την καταγραφή της συγκέντρωσης κάθε ρύπου ανά δεκαπέντε λεπτά για το Camden Borough του Λονδίνου και για την περίοδο μεταξύ 1 Ιανουαρίου 2010 και 1 Ιανουαρίου 2011.

Τα δεδομένα πάρθηκαν από το London Air Quality Network και το Department for Environment, Food and Rural Affairs του Ηνωμένου Βασιλείου. Πριν την εισαγωγή των δεδομένων στη βάση μας φροντίσαμε να διαγράψουμε τις κενές γραμμές των csv αρχείων ώστε να μην δημιουργείται επιπλέον πολυπλοκότητα κατά τη διάρκεια της φόρτωσής τους σε αυτή.

Παρακάτω παρατίθεται ενδεικτικά ένα μέρος των αρχείων για το διοξείδιο του αζώτου (NO2):

Index	Band	ug/m3
1	Low	0-67
2	Low	68-134
3	Low	135-200
4	Moderate	201-267
5	Moderate	268-334
6	Moderate	335-400
7	High	401-467
8	High	468-534
9	High	535-600
10	Very High	601-1000

Πίνακας 3.7 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "no2_index.csv"

Site	Species	ReadingDateTime	Value	Units	Provisional or Ratified
Camden-St Martins College (NOX 2)	NO2	01/01/2010 00:00	57.3	ug/m3	R
Camden-St Martins College (NOX 2)	NO2	01/01/2010 00:15	56.2	ug/m3	R
Camden-Bloomsbury	NO2	01/01/2010 00:30	53.5	ug/m3	R
Camden-St Martins College (NOX 1)	NO2	01/01/2010 00:45	32.1	ug/m3	R
Camden-St Martins College (NOX 1)	NO2	01/01/2010 01:00	44.6	ug/m3	R

Πίνακας 3.8 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "no2_concentration.csv"

3.2 Καιρικές συνθήκες

Εκτός από τα δεδομένα ατμοσφαιρικής ρύπανσης, στη βάση δεδομένων θα προστεθούν και δεδομένα καιρικών συνθηκών. Οι συνθήκες που επικρατούν καθημερινά διαδραματίζουν σημαντικό ρόλο στην ικανοποίηση ή μη των πολιτών, σχετικά με τις εξωτερικές δραστηριότητες που θα πραγματοποιήσουν. Ακόμα, ο καιρός είναι ένας από τους πιο καθοριστικούς παράγοντες για το πως θα εξελιχθούν τα επίπεδα ατμοσφαιρικής ρύπανσης μιας ημέρας. Όταν για παράδειγμα επικρατούν υγρές ή ανεμώδης συνθήκες τα επίπεδα ατμοσφαιρικής ρύπανσης παραμένουν χαμηλά. Αντίθετα, κάτω από ζεστές και χωρίς ανέμους συνθήκες είναι πιθανό να συγκεντρωθούν μεγάλες ποσότητες ρύπων οδηγώντας στα αποκαλούμενα 'επεισόδια ρύπανσης'.

Τα δεδομένα καιρού που θα ληφθούν υπόψιν στη συγκεκριμένη υλοποίηση είναι:

- Η θερμοκρασία (ºC)
- Η σχετική υγρασία (%)
- Η βροχοπτώσεις (mm)
- Η ταχύτητα των ανέμων (m/s)

Όλα τα δεδομένα αφορούν την περίοδο από 1 Ιανουαρίου 2010 έως 1 Ιανουαρίου 2011, είναι με δειγματοληψία κάθε δεκαπέντε λεπτά και έχουν παρθεί από το London Air Quality Network για το Camden Borough του Λονδίνου. Στη συνέχεια θα ακολουθήσει αναλυτική παρουσίαση των δεδομένων αυτών.

3.2.1 Θερμοκρασία

Η θερμοκρασία είναι ένας από τους σημαντικότερους παράγοντες που μπορεί να επηρεάσουν τις εξωτερικές δραστηριότητες κάποιου ατόμου. Κάθε χρήστης της εφαρμογής θα μπορεί να δηλώσει εκτός των άλλων και ένα διάστημα με την προτίμηση θερμοκρασίας (π.χ. προτίμηση θερμοκρασιών από 8 ºC έως 23 ºC).

Το αρχείο με τα δεδομένα θερμοκρασίας είναι το 'temperature.csv' και όλες οι μετρήσεις έχουν γίνει σε ºC. Περιέχει στήλες με το σημείο στο οποίο έχει γίνει η μέτρηση (Site), την ημερομηνία και την ώρα (ReadingDateTime), την τιμή της μέτρησης (value) και τη μονάδα μέτρησης (Units).

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

Site	Species	ReadingDateTime	Value	Units	Provisional or Ratified
Camden-St Martins College (NOX 1)	TMP	01/01/2010 00:00	2	oC	P
Camden-St Martins College (NOX 1)	TMP	01/01/2010 00:15	2	oC	P
Camden-St Martins College (NOX 1)	TMP	01/01/2010 00:30	2	oC	P
Camden-St Martins College (NOX 1)	TMP	01/01/2010 00:45	2	oC	P
Camden-St Martins College (NOX 1)	TMP	01/01/2010 01:00	2	oC	P

Πίνακας 3.9 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "temperature.csv"

3.2.2 Σχετική Υγρασία

Όπως με τη θερμοκρασία, κάθε χρήστης μπορεί να δηλώσει επίσης τις προτιμήσεις του σε σχετική υγρασία.

Το αρχείο με τα δεδομένα σχετικής υγρασίας είναι το 'relative_humidity.csv'και όλες οι μετρήσεις είναι σε ποσοστό επί της εκατό. Περιέχει στήλες με το σημείο στο οποίο έχει γίνει η μέτρηση (Site), την ημερομηνία και ώρα (ReadingDateTime), την τιμή της μέτρησης (value) και τη μονάδα μέτρησης (Units).

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

Site	Species	ReadingDateTime	Value	Units	Provisional or Ratified
Camden-St Martins College (NOX 1)	RHUM	01/01/2010 00:00	74	%	P
Camden-St Martins College (NOX 1)	RHUM	01/01/2010 00:15	75	%	P
Camden-St Martins College (NOX 1)	RHUM	01/01/2010 00:30	74	%	P
Camden-St Martins College (NOX 1)	RHUM	01/01/2010 00:45	75	%	P
Camden-St Martins College (NOX 1)	RHUM	01/01/2010 01:00	76	%	P

Πίνακας 3.10 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "relative_humidity.csv"

3.2.3 Βροχοπτώσεις

Οι μετρήσεις για τη βροχόπτωση έχουν γίνει σε mm και μας δίνουν πληροφορία για την ένταση της βροχής σε κάποια περιοχή. Τα δεδομένα που χρησιμοποιήσαμε είχαν ελάχιστη τιμή 0 mm και μέγιστη τιμή 5mm. Συνεπώς για την υλοποίηση της εργασίας θεωρήσαμε ότι η κλίμακα των βροχοπτώσεων κυμαίνεται από 0 έως 5 και κάθε χρήστης της εφαρμογής θα μπορεί να δηλώσει προτίμηση σε βροχή δίνοντας ένα διάστημα εντός της κλίμακας αυτής. Για παράδειγμα η προτίμηση ενός χρήστης σε λίγη έως καθόλου βροχή θα αντιστοιχεί σε ένα διάστημα [0,1].

Το αρχείο με τα δεδομένα των βροχοπτώσεων είναι το 'rainfall.csv'. Περιέχει στήλες με το σημείο στο οποίο έχει γίνει η μέτρηση (Site), την ημερομηνία και την ώρα (ReadingDateTime), την τιμή της μέτρησης (value) και τη μονάδα μέτρησης (Units).

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

Site	Species	ReadingDateTime	Value	Units	Provisional or Ratified
Camden-St Martins College (NOX 1)	RAIN	01/01/2010 00:00	0	mm	P
Camden-St Martins College (NOX 1)	RAIN	01/01/2010 00:15	0	mm	P
Camden-St Martins College (NOX 1)	RAIN	01/01/2010 00:30	0	mm	P
Camden-St Martins College (NOX 1)	RAIN	01/01/2010 00:45	0	mm	P
Camden-St Martins College (NOX 1)	RAIN	01/01/2010 01:00	0	mm	P

Πίνακας 3.11 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "rainfall.csv"

3.2.4 Ταχύτητα Ανέμων

Όλα οι μετρήσεις για την ταχύτητα των ανέμων έχουν γίνει σε m/s. Επειδή, ένα μέρος της εφαρμογής αποτελούν οι προτιμήσεις των χρηστών σε καιρικές συνθήκες και επειδή θεωρήσαμε ότι κάποιος χρήστης δεν μπορεί να δηλώνει προτίμηση για ταχύτητα ανέμων σε m/s, έχουμε εισάγει στο σημείο αυτό ένα ακόμα επίπεδο ολοκλήρωσης. Συγκεκριμένα, ενώ οι μετρήσεις μας είναι σε m/s, όταν εισάγονται στη βάση δεδομένων αντιστοιχίζονται – ανάλογα με την τιμή τους – στην κλίμακα μποφόρ και από εκεί σε ένα επίπεδο ανέμων. Τα επίπεδα ανέμων είναι τέσσερα (light wind, high wind, gale force, storm force, hurricane force) και η κλίμακα μποφόρ παίρνει τιμές από το 0 έως το 12. Έτσι, κάποιος χρήστης μπορεί να δηλώσει την προτίμησή του είτε με βάση την κλίμακα μποφόρ είτε με βάση το επίπεδο ανέμων. Οι προτιμήσεις των χρηστών αποθηκεύονται στη βάση ως ένα κλειστό διάστημα της κλίμακας μποφόρ (π.χ. [0,2]).

Το αρχείο με τα στοιχεία για την κλίμακα μποφόρ είναι το 'beaufort_scale.csv' και περιέχει στήλες με την τιμή της κλίμακας μποφόρ (Beaufort number), την περιγραφή του ανέμου ανάλογα με την τιμή της κλίμακας (Description), τα ελάχιστα (Wind speed_min (m/s)) και τα μέγιστα (Wind speed_max (m/s)) όρια ταχύτητας των ανέμων για κάθε τιμή της κλίμακας και τέλος το επίπεδο των ανέμων (Level).

Τα περιεχόμενα του αρχείου φαίνονται στον πίνακα που ακολουθεί:

Beaufort number	Description	Wind speed_min (m/s)	Wind speed_max (m/s)	Level
0	Calm	0	0.3	Light Wind
1	Light Air	0.3	1.6	Light Wind
2	Light Breeze	1.6	3.4	Light Wind
3	Gentle Breeze	3.4	5.5	Light Wind
4	Moderate Breeze	5.5	8	Light Wind
5	Fresh Breeze	8	10.8	Light Wind
6	Strong Breeze	10.8	13.9	High Wind
7	Near Gale	13.9	17.2	High Wind
8	Gale	17.2	20.8	Gale Force
9	Strong Gale	20.8	24.5	Gale Force
10	Storm	24.5	28.5	Storm Force
11	Violent Storm	28.5	32.7	Storm Force
12	Hurricane Force	32.7	70	Hurricane Force

Πίνακας 3.12 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "beaufort_scale.csv"

Το αρχείο με τα δεδομένα της ταχύτητας των ανέμων είναι το 'wind.csv'. Περιέχει στήλες με το σημείο στο οποίο έχει γίνει η μέτρηση (Site), την ημερομηνία και την ώρα (ReadingDateTime), την τιμή της μέτρησης (value) και τη μονάδα μέτρησης (Units).

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

Site	Species	ReadingDateTime	Value	Units	Provisional or Ratified
Camden-St Martins College (NOX 1)	WSPD	01/01/2010 00:00	0.8	m/s	P
Camden-St Martins College (NOX 1)	WSPD	01/01/2010 00:15	1.4	m/s	P
Camden-St Martins College (NOX 1)	WSPD	01/01/2010 00:30	1.1	m/s	P
Camden-St Martins College (NOX 1)	WSPD	01/01/2010 00:45	1.5	m/s	P
Camden-St Martins College (NOX 1)	WSPD	01/01/2010 01:00	1.2	m/s	P

Πίνακας 3.13 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "wind.csv"

3.3 Εξωτερικές Δραστηριότητες

Για την εισαγωγή των εξωτερικών δραστηριοτήτων δεν χρησιμοποιήσαμε κάποιο αρχείο δεδομένων. Οι εξωτερικές δραστηριότητες είναι απλά κάποια ονόματα δραστηριοτήτων και σχετίζονται με τους χρήστες της εφαρμογής. Το περπάτημα, το τρέξιμο, η ποδηλασία, η εξωτερική φωτογράφιση ή η επίσκεψη σε αξιοθέατα είναι μερικές από αυτές. Κατά τη δημιουργία της βάσης δεδομένων μπορούμε να εισάγουμε εξ αρχής μερικές προεπιλεγμένες δραστηριότητες. Στην παρούσα υλοποίηση όμως, αποφασίσαμε κάθε δραστηριότητα να εισάγεται στη βάση δεδομένων όταν την επιλέξει κάποιος χρήστης.

3.4 Χρήστες

Η ολοκλήρωση της εφαρμογής γίνεται με τα δεδομένα των χρηστών. Οι χρήστες είναι οι τελικοί αποδέκτες του συστήματος και κάθε ένας από αυτούς για να μπορέσει να χρησιμοποιήσει την εφαρμογή πρέπει πρώτα να πραγματοποιήσει μια εγγραφή στο σύστημα. Στη συνέχεια, κάποιος χρήστης μπορεί να δηλώσει προτιμήσεις τόσο σε καιρικές συνθήκες όσο και σε δραστηριότητες που του αρέσουν. Επίσης, οι χρήστες έχουν τη δυνατότητα να βαθμολογούν ανά πάσα στιγμή κάποια δραστηριότητα που έκαναν και οι αξιολογήσεις αυτές αποθηκεύονται άμεσα στο ιστορικό τους. Με βάση το ιστορικό αλλά και το προφίλ του, ο κάθε χρήστης θα μπορεί να ζητάει προτεινόμενες δραστηριότητες. Φυσικά, ανεξάρτητα από αυτά, θα μπορεί να βλέπει την τρέχουσα αξιολόγηση κάθε δραστηριότητας από άλλους χρήστες της εφαρμογής σε πραγματικό χρόνο. Κάθε χρήστης μπορεί τέλος να αλλάζει τις πληροφορίες του προφίλ του και των προτιμήσεων του.

Δυστυχώς, κατά την υλοποίηση της εφαρμογής δεν υπήρχαν διαθέσιμα πραγματικά δεδομένα χρηστών, οπότε χρειάστηκε να υλοποιήσουμε εικονικούς χρήστες για την εφαρμογή μας. Συνολικά δημιουργήθηκαν εκατό διαφορετικοί χρήστες και περίπου 50.000 αξιολογήσεις δραστηριοτήτων για το Μάρτιο του 2010. Πολλά από τα δεδομένα αυτά παράχθηκαν με τη βοήθεια του Mockaroo, μιας γεννήτριας πραγματικών δεδομένων για τον έλεγχο εφαρμογών.

3.4.1 Εγγραφή Χρηστών

Αρχικά έχουμε τα δεδομένα για την εγγραφή χρηστών στην εφαρμογή. Κάθε χρήστης έχει μοναδικό username και μοναδικό e-mail. Στη βάση αποθηκεύονται επίσης στοιχεία για τον κωδικό, το όνομα, το φύλο και την ηλικία των χρηστών. Να επισημάνουμε εδώ ότι στην υλοποίηση μας το πεδίο του κωδικού αποθηκεύεται αυτούσιο στην βάση δεδομένων. Για λόγους ασφαλείας όμως, στην περίπτωση δημιουργίας πραγματικής εφαρμογής, το πεδίο αυτό θα πρέπει να περνάει από διαδικασία κρυπτογράφησης πριν αποθηκευτεί στη βάση. Ενδεικτικά να αναφέρουμε ότι για αυτό το σκοπό θα μπορούσε να χρησιμοποιηθεί το bcrypt.

Το αρχείο με τα δεδομένα των χρηστών είναι το ‘users.cvs’ και ένα μικρό κομμάτι του παρουσιάζεται στον ακόλουθο πίνακα:

full_name	username	password	email	gender	age
Susan Hernandez	susanhernandez553	mmcKYMGpSs	shernandez0@examiner.com	Female	26
James Carter	jamescarter555	Pjum3pm3AJR	jcarter1@prnewswire.com	Male	50
Doris Green	dorisgreen140	rbSnHzd	dgreen2@ox.ac.uk	Female	34
Theresa Lynch	theresalynch34	i1ycwRdBNh8e	tlynch3@techcrunch.com	Female	23

Πίνακας 3.14 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου “users.csv”

3.4.2 Προτιμήσεις καιρικών συνθηκών

Κάθε χρήστης, μετά από την εγγραφή του στην εφαρμογή μπορεί να δηλώσει προτιμήσεις καιρικών συνθηκών και εξωτερικών δραστηριοτήτων. Συγκεκριμένα, για τις καιρικές συνθήκες, μπορεί να δηλώσει:

- Μέγιστη και ελάχιστη θερμοκρασία (σε ^{o}C)
- Μέγιστη και ελάχιστη σχετική υγρασία (%)
- Μέγιστη και ελάχιστη τιμή βροχόπτωσης (0 έως 5)
- Προτίμηση στα επίπεδα ανέμων
- Προτίμηση στα επίπεδα ατμοσφαιρικής ρύπανσης

Η προτίμηση στους ανέμους μπορεί να δηλωθεί είτε με βάση την κλίμακα μποφόρ (0 έως 12), είτε με βάση τα επίπεδα ανέμων (light wind, high wind, gale force, storm force, hurricane force). Σε κάθε περίπτωση, οι προτιμήσεις αυτές μεταφράζονται σε κλίμακα μποφόρ πριν αποθηκευτούν στη βάση.

Η προτίμηση σε ατμοσφαιρική ρύπανση δηλώνεται είτε με βάση το DAQI (1 έως 10), είτε με βάση τα επίπεδα ρύπων (Low, Moderate, High, Very High). Όμοια με τους ανέμους, και οι προτιμήσεις της ατμοσφαιρικής ρύπανσης μεταφράζονται στην κλίμακα του daily air quality index πριν αποθηκευτούν στη βάση.

Σε περίπτωση που κάποιος χρήστης δεν δηλώσει προτιμήσεις καιρού κατά την εγγραφή του, υπάρχει η δυνατότητα αυτόματης εισαγωγής προτιμήσεων με τις μέγιστες και τις ελάχιστες τιμές για κάθε κατηγορία.

Κατά την υλοποίηση της συγκεκριμένης εργασίας έγινε μαζική εισαγωγή προτιμήσεων για όλους τους χρήστες που δημιουργήθηκαν, από το αρχείο ‘users_conditionsPreferences.csv’. Το αρχείο περιέχει στήλες με το όνομα χρήστη (username) και στήλες με τις προτιμήσεις του αντίστοιχου χρήστη για κάθε κατηγορία καιρικών συνθηκών.

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

username	max Pol	min Pol	max Temp	min Temp	max Rain	min Rain	max RHum	min RHum	max Wind	min Wind
adamfisher158	6	1	17	0	0	0	88	31	2	1
alansanders227	6	1	17	0	1	0	88	31	2	1
amandareynolds960	6	1	17	0	2	0	88	31	2	0
angelamoore322	6	1	17	0	0	0	88	32	2	1
bettyhansen603	6	1	17	0	1	0	89	31	2	0
beverlywilson536	6	1	17	0	1	0	88	31	2	1

Πίνακας 3.15 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου “users_conditionsPreferences.csv”

3.4.3 Προτιμήσεις Δραστηριοτήτων

Οι προτιμήσεις των δραστηριοτήτων χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει δραστηριότητες που αρέσουν σε κάποιους χρήστες της εφαρμογής, ενώ η δεύτερη κατηγορία περιλαμβάνει δραστηριότητες που ορισμένοι χρήστες αντιπαθούν.

Το αρχείο με της δραστηριότητες της πρώτης κατηγορίας είναι το ‘users_likes.csv’. Περιέχει στήλες με το username του χρήστη (username), με το όνομα της δραστηριότητας (activity), και με μία τιμή από το 1 μέχρι το 10 που αντιπροσωπεύει το ‘πόσο’ αρέσει η δραστηριότητα στο χρήστη (score).

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

username	activity	score
alansanders227	Cycling	10
alansanders227	Outdoor Photography	7
alansanders227	Sightseeing	6
amandareynolds960	Cycling	10
amandareynolds960	Outdoor Photography	8

Πίνακας 3.16 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "users_likes.csv"

Το αρχείο με τις δραστηριότητες της δεύτερης κατηγορίας είναι το 'users_dislikes.csv'. Περιέχει στήλες με το username του χρήστη (username) και με το όνομα της δραστηριότητας (activity). Να σημειωθεί ότι κατά την αποθήκευση των dislikes στη βάση δεδομένων, προστίθεται αυτόματα και η τιμή του dislike που ισούται με 0.

Ενδεικτικό κομμάτι του αρχείου παρατίθεται στον ακόλουθο πίνακα:

username	activity
roseboyd415	Walking
roseboyd415	Gardening
carolynday261	Outdoor Photography
frankharrison890	Cycling
sharonvasquez55	Sightseeing

Πίνακας 3.17 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "users_dislikes.csv"

3.4.4 Αξιολογήσεις Δραστηριοτήτων

Οι αξιολογήσεις δραστηριοτήτων διαφέρουν από τις προτιμήσεις δραστηριοτήτων. Κάθε χρήστης μπορεί να έχει κάποιες προτιμήσεις σε δραστηριότητες όπως αναφέρθηκε στην προηγούμενη παράγραφο. Μπορεί, δηλαδή, να δηλώσει ότι του αρέσει το τρέξιμο με ένα 'βάρος – αρεσκείας' από το 1 έως το 10. Ανεξάρτητα από αυτό όμως, μπορεί να αξιολογεί τη συγκεκριμένη δραστηριότητα οποιαδήποτε στιγμή της ημέρας με βάση την ικανοποίηση του από τη συγκεκριμένη δραστηριότητα την τρέχουσα στιγμή.

Αξιολογήσεις μπορούμε επομένως να έχουμε, από οποιονδήποτε χρήστη, για οποιαδήποτε δραστηριότητα και οποιαδήποτε στιγμή. Για λόγους ομαδοποίησης των αξιολογήσεων σε ένα πλαίσιο χρόνου όμοιο με τους χρόνους δειγματοληψίας των καιρικών φαινομένων και της ατμοσφαιρικής ρύπανσης, κάθε αξιολόγηση τοποθετείται στο τέταρτο της ώρας που αντιστοιχεί στα πιο πρόσφατα δεδομένα καιρικών συνθηκών. Αν για παράδειγμα κάποιος χρήστης αξιολογήσει τη δραστηριότητα 'περπάτημα' στις 10:10 πμ στις 03/03/2010, τότε η συγκεκριμένη αξιολόγηση θα αντιστοιχίζεται στο πρώτο τέταρτο της ώρας καθώς η πιο πρόσφατη δειγματοληψία καιρικών δεδομένων έγινε στις 10:00 πμ.

Για την υλοποίηση και τον έλεγχο της παρούσας εργασίας, έγινε μαζική δημιουργία και εισαγωγή αξιολογήσεων από το αρχείο 'rating_activities.csv'. Το αρχείο αυτό περιέχει στήλες με το όνομα χρήστη (username), την ημέρα και την ώρα (dateTime), τη δραστηριότητα (activity), και την αξιολόγηση (rating) του κάθε χρήστη.

Ενδεικτικό κομμάτι του αρχείο επισυνάπτεται στον πίνακα που ακολουθεί:

username	dateTime	activity	rating
brendaray923	16/03/2010 23:00	Walking	10
theresalynch34	21/03/2010 00:45	Dog Walking	8
theresalynch34	09/03/2010 00:30	Walking	9
brendaray923	14/03/2010 21:45	Walking	8
phillipmendoza194	02/03/2010 12:45	Dog Walking	9

Πίνακας 3.18 Ενδεικτικό κομμάτι των περιεχομένων του αρχείου "rating_activities.csv"

Να σημειωθεί εδώ ότι όλες οι αξιολογήσεις που δημιουργήθηκαν, αφορούν το μήνα Μάρτιο του 2010 και τις ώρες από 06:00 μέχρι 00:59 κάθε ημέρας. Σκοπός της συγκέντρωσης όλων των δεδομένων σε ένα συγκεκριμένο μήνα, ήταν η επίτευξη της απαραίτητης πυκνότητας δεδομένων που θα επιτρέψει στη συνέχεια τόσο την υλοποίηση όσο και τον έλεγχο ερωτημάτων που θα βασίζεται στο ‘συνεργατικό φιλτράρισμα’ (collaborative filtering).

4 Μοντελοποίηση Δεδομένων

Στο κεφάλαιο αυτό θα περιγράψουμε τη διαδικασία μοντελοποίησης των δεδομένων μας για την εισαγωγή τους στη Neo4j. Θα ακολουθήσουμε την πραγματική πορεία δημιουργίας των μοντέλων γράφων, ξεκινώντας από τα αρχικά μοντέλα που δημιουργήσαμε και δείχνοντας τη διαδικασία εξέλιξης και βελτίωσης τους, ώστε να καταλήξουμε στα μοντέλα που ικανοποιούν τελικά τις απαιτήσεις της εφαρμογής. Θα μελετήσουμε επίσης προβλήματα που δημιουργήθηκαν κατά την μοντελοποίηση καθώς και τις τεχνικές που χρησιμοποιήσαμε για την αντιμετώπισή τους.

4.1 Μοντέλα και μοντελοποίηση με χρήση δομών γράφων

Πριν ξεκινήσουμε τη διαδικασία μοντελοποίησης, θεωρούμε σκόπιμο να αναφερθούμε εδώ στην έννοια τον μοντέλων γενικά και στο ‘μοντέλο γράφων με ετικέτες και ιδιότητες’ (labeled property graph model), το οποίο χρησιμοποιείται από τις περισσότερες βάσεις δεδομένων γράφων και φυσικά από τη Neo4j.

4.1.1 Μοντελοποίηση δεδομένων στον πραγματικό κόσμο

Η διαδικασία μοντελοποίησης είναι μια αφαιρετική διαδικασία που εξυπηρετεί συγκεκριμένο στόχο ή σκοπό. Ακολουθείται σε περιπτώσεις που θέλουμε να αναδείξουμε τις πτυχές ενός προβλήματος με τέτοιο τρόπο ώστε να είναι δυνατή η δομημένη και διαχειρίσιμη επεξεργασία τους. Θα μπορούσαμε να πούμε πως δεν είναι ποτέ εφικτή η καθολική αναπαράσταση ενός προβλήματος όπως αυτό υπάρχει στην πραγματικότητα. Είναι δυνατές μόνο αφαιρετικές επιλογές και απλοποιήσεις χαρακτηριστικών του, κάποιες από τις οποίες είναι χρήσιμες για την εξυπηρέτηση ορισμένων αναγκών και κάποιες άλλες για την επίτευξη διαφορετικών στόχων.

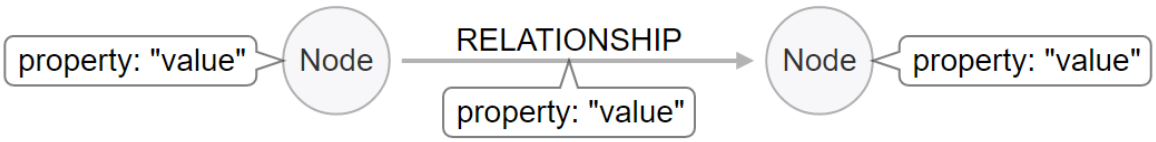
Όπως με όλα τα μοντέλα, αναπόφευκτα το ίδιο ισχύει και για τα μοντέλα γράφων. Το στοιχείο όμως που διαφοροποιεί τις τεχνικές μοντελοποίησης με χρήση γράφων, σε σχέση με άλλε τεχνικές που ενδεχομένως θα μπορούσαν να χρησιμοποιηθούν, είναι η μεγάλη συγγένεια που έχουν οι Γράφοι (ως λογικές δομές) με τα φυσικά μοντέλα. Για παράδειγμα, οι κλασικές σχεσιακές βάσεις δεδομένων απαιτούν ένα πρώτο βήμα αναπαράστασης του κόσμου που θέλουν να περιγράψουν σε λογικά μοντέλα, και στη συνέχεια ένα ακόμα βήμα μετατροπής των λογικών αυτών μοντέλων σε άλλα φυσικά μοντέλα που χρησιμοποιούνται για την πραγματική εισαγωγή των δεδομένων στη βάση δεδομένων. Αυτές οι μετατροπές όμως, εισάγουν ‘σημασιολογικές παραφωνίες’ όσον αφορά την αντίληψη που έχουμε για τον κόσμο και την πραγματική αναπαράσταση του στη βάση δεδομένων. Με τις βάσεις δεδομένων γράφων το συγκεκριμένο πρόβλημα μειώνεται σημαντικά.

Από τη φύση τους, οι Γράφοι ταυτίζονται με τον τρόπο που συνηθίζουμε να μοντελοποιούμε τα δεδομένα μας: μοντελοποιούμε χρησιμοποιώντας κύκλους και κουτιά και στη συνέχεια περιγράφουμε τις σχέσεις που συνδέουν τις οντότητες αυτές ενώνοντάς τες με βέλη και γραμμές. Θα μπορούσαμε να πούμε ότι ο πιο φυσικός τρόπος αναπαράστασης ενός προβλήματος είναι οι Γράφοι, και η χρησιμοποίηση μια βάσης δεδομένων γράφων, όπως η Neo4j, μας επιτρέπει να αποθηκεύσουμε τα δεδομένα μας όπως τα έχουμε σχεδιάσει στο μοντέλο μας.

4.1.2 Μοντέλο γράφων με ετικέτες και ιδιότητες

Σύμφωνα με όσα αναφέρθηκαν στην προηγούμενη παράγραφο, μια βάση δεδομένων γράφων χρησιμοποιεί δομές γράφων για τα σημασιολογικά ερωτήματα και την αποθήκευση των πληροφοριών. Ένας γράφος αποτελείται από κόμβους (nodes) και ακμές (edges), όπου οι κόμβοι αντιπροσωπεύουν συνήθως τις οντότητες (entities) ενός πεδίου ορισμού (domain) και οι ακμές τις σχέσεις (relationships) που τους συνδέουν. Σε μια βάση δεδομένων γράφων, τόσο οι κόμβοι όσο και οι ακμές μπορούν να αντιπροσωπεύουν και να αποθηκεύουν δεδομένα.

Το μοντέλο γράφων με ετικέτες και ιδιότητες (labeled property graph model), είναι η δομή που χρησιμοποιούν οι περισσότερες βάσεις δεδομένων γράφων για την αποθήκευση των πληροφοριών τους. Το ίδιο μοντέλο υιοθετείται και από τη Neo4j, τη βάση δεδομένων γράφων που επιλέξαμε για την υλοποίηση της εργασίας μας.

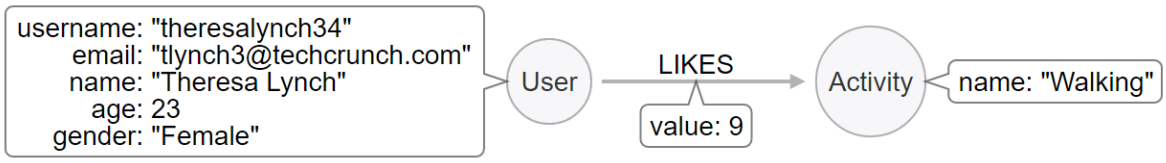


Εικόνα 4.1 Μοντέλο γράφων με ετικέτες και ιδιότητες

Τα χαρακτηριστικά του μοντέλου συνοψίζονται ακολούθως:

- Ένας γράφος με ετικέτες και ιδιότητες αποτελείται από κόμβους (nodes), σχέσεις (relationships), ιδιότητες (properties), και ετικέτες (labels).
- Οι κόμβοι περιέχουν ιδιότητες. Μπορούμε να φανταστούμε ένα κόμβο σαν ένα αρχείο που αποθηκεύει ιδιότητες σε μορφή ζευγαριών ‘ονόματος – τιμής’ (key-value pairs). Στη Neo4j τα ονόματα (property-keys) είναι συμβολοσειρές (strings) και οι τιμές (values) είναι είτε συμβολοσειρές Java, είτε πρωτόγονοι τύποι δεδομένων (π.χ. integer) είτε και πίνακες αυτών.
- Οι σχέσεις (relationships) περιέχουν επίσης ιδιότητες. Οι ιδιότητες που αποθηκεύονται στις σχέσεις παρέχουν επιπλέον πληροφορίες και μεταδεδομένα κατά τη διάσχιση των γράφων (π.χ. βάρη, ημερομηνίες, περιορισμούς διάσχισης κλπ.).
- Στους κόμβους μπορούμε να τοποθετήσουμε ετικέτες (labels). Οι ετικέτες επισημαίνουν το ρόλο που παίζει κάθε κόμβος μέσα στο σύνολο δεδομένων και ταυτόχρονα αποτελούν ένα μέσο ομαδοποίησης των κόμβων σε διακριτές ομάδες.
- Οι σχέσεις συνδέουν τους κόμβους μεταξύ τους και ορίζουν τη δομή και τη σημασιολογία του γράφου. Κάθε σχέση έχει πάντα μια κατεύθυνση, ένα μοναδικό όνομα, έναν αρχικό κόμβο και ένα τελικό κόμβο. Να επισημάνουμε εδώ ότι δεν μπορεί ποτέ να υπάρξει αιωρούμενη σχέση. Πάντα, δηλαδή, μια σχέση ξεκινάει από ένα κόμβο (αρχικός κόμβος) και καταλήγει σε έναν άλλο κόμβο (τελικός κόμβος).

Στην εικόνα που ακολουθεί (εικόνα 4.2), βλέπουμε ένα παράδειγμα μοντελοποίησης της σχέσης “LIKES” μεταξύ ενός χρήστη (user) και μιας δραστηριότητας (activity) με χρήση του μοντέλου γράφων με ετικέτες και ιδιότητες:



Εικόνα 4.2 Παράδειγμα μοντελοποίησης με χρήση του μοντέλου γράφων με ετικέτες και ιδιότητες

Παρατηρώντας την εικόνα, εύκολα μπορούμε να καταλάβουμε ότι η χρήστης με όνομα “Theresa Lynch” έχει δηλώσει πως της αρέσει η δραστηριότητα “Walking” με ένα βάρος ίσο με 9. Στο παράδειγμα αυτό μπορούμε να διακρίνουμε όλα τα χαρακτηριστικά του μοντέλου που αναφέρθηκαν προηγουμένως:

- Αρχικά, έχουμε ένα γράφο που αποτελείται από δύο κόμβους και από μια σχέση που τους συνδέει.
- Στον κόμβο του χρήστη έχει προστεθεί η ετικέτα “User”, ενώ στον κόμβο της δραστηριότητας έχει προστεθεί η ετικέτα “Activity”.
- Οι δύο κόμβοι εκτός από την ετικέτα τους, έχουν αποθηκευμένες και ιδιότητες στη μορφή ονόματος-τιμής. Βλέπουμε ότι ο κόμβος χρήστη έχει τις ιδιότητες username, email, name, age και gender, ενώ ο κόμβος της δραστηριότητας έχει μια ιδιότητα name.
- Η σχέση “LIKES” δίνει δομή και σημασιολογία στο γράφο, ενώνοντας τους δύο κόμβους και δείχνοντας πως ένας χρήστης έχει δηλώσει ότι του αρέσει μια συγκεκριμένη δραστηριότητα. Παρατηρούμε ακόμα ότι η σχέση έχει ένα μοναδικό όνομα (LIKES), έχει μια κατεύθυνση (από το χρήστη προς τη δραστηριότητα), έναν αρχικό κόμβο (user) και ένα τελικό κόμβο (activity).
- Τέλος, βλέπουμε ότι η σχέση “LIKES” έχει αποθηκευμένη μια ιδιότητα σε μορφή ονόματος-τιμής. Συγκεκριμένα έχει την ιδιότητα με το όνομα “value” και την τιμή 9, που μας δίνει μια επιπλέον πληροφορία για τα δεδομένα μας. Μας λέει δηλαδή ότι η χρήστης όχι μόνο έχει δηλώσει πως της αρέσει η συγκεκριμένη δραστηριότητα, αλλά έχει προσδιορίσει και το βαθμό με τον οποίο της αρέσει.

4.2 Πορεία μοντελοποίησης των δεδομένων

Αναλυτική παρουσίαση των δεδομένων που θα χρησιμοποιήσουμε έχει γίνει στο κεφάλαιο 3. Σκοπός της συγκεκριμένης ενότητας είναι η παρουσίαση της πορείας που ακολουθήθηκε κατά την μοντελοποίηση των δεδομένων, η παρουσίαση των προβλημάτων που αντιμετωπίσαμε καθώς και των μοντέλων που τελικά χρησιμοποιήσαμε. Ξεκινάμε όμως πρώτα με την περιγραφή των βασικών αρχών που διέπουν τη διαδικασία μοντελοποίησης των δεδομένων για την εισαγωγή τους στη Neo4j.

4.2.1 Βασική τεχνική μοντελοποίησης

Ένα σημαντικό πλεονέκτημα του μοντέλου γράφων είναι το γεγονός ότι όχι μόνο απεικονίζει τα πράγματα και τις σχέσεις που τα συνδέουν με τον ίδιο ακριβώς τρόπο που τα σκεφτόμαστε, αλλά αποτελεί και μια εικόνα των ερωτήσεων που θέλουμε να απαντήσουμε. Με λίγα λόγια η διαδικασία μοντελοποίησης σε γράφους μπορεί να συνοψισθεί ως η προσπάθεια δημιουργίας δομών γράφων που εκφράζουν τα ερωτήματα που θέλουμε να απαντήσουμε.

Τα βήματα που ακολουθούμε προς μια τέτοια κατεύθυνση είναι:

1. Περιγραφή των στόχων που θέλουμε να ικανοποιεί το μοντέλο μας.
2. Μετατροπή των στόχων αυτών σε ερωτήματα που θα πρέπει να απαντηθούν.
3. Προσδιορισμός των *οντοτήτων* και των *σχέσεων* που εμφανίζονται στα παραπάνω ερωτήματα.
4. Δημιουργία του γράφου με βάση τις *οντότητες* και τις *σχέσεις* που εντοπίσαμε.

Ένα κρίσιμο σημείο στην παραπάνω διαδικασία είναι ο προσδιορισμός των *οντοτήτων* και των *σχέσεων* του πεδίου μας. Πως επιλέγουμε τι είναι *οντότητα* και τι είναι *σχέση*; Τι είναι *ιδιότητα*, τι είναι *ετικέτα* και τι όχι; Η απλή απάντηση στα ερωτήματα αυτά είναι ότι μπορούμε να εντοπίσουμε τα συγκεκριμένα στοιχεία ελέγχοντας τη φυσική γλώσσα που χρησιμοποιούμε για να περιγράψουμε τα ερωτήματα:

- Τα ουσιαστικά, όπως “user” ή “activity”, συνήθως αποτελούν τους κόμβους του γράφου μας, στους οποίους προσθέτουμε και τις αντίστοιχες ετικέτες.
- Ό,τι προσδιορίζει ή αναφέρεται στα ουσιαστικά αυτά, όπως “name” ή “email”, αποτελεί τις *ιδιότητες* των *κόμβων*.
- Τα ρήματα, όπως το “likes”, συνήθως είναι οι *σχέσεις* που συνδέουν τους *κόμβους* του πεδίου μας και χρησιμοποιούνται ως ονόματα των *σχέσεων* του γράφου.
- Ό,τι προσδιορίζει ή αναφέρεται στα ρήματα αυτά, όπως κάποιο “weight” ή κάποια άλλα μεταδεδομένα, αποτελεί τις *ιδιότητες* των σχέσεων.

Φυσικά, θα πρέπει να λάβουμε υπόψιν ότι δεν υπάρχει μια ενιαία πρακτική που να ταιριάζει σε κάθε περίπτωση, αλλά θα πρέπει να διαμορφώνουμε τα μοντέλα που δημιουργούμε κάθε φορά λαμβάνοντας υπόψιν τις ιδιαιτερότητες του κάθε προβλήματος.

Με βάση τα όσα είπαμε παραπάνω, μπορούμε πλέον να προχωρήσουμε στη διαδικασία μοντελοποίησης ξεκινώντας από την εισαγωγή των δεδομένων ατμοσφαιρικής ρύπανσης.

4.2.2 Εισαγωγή δεδομένων ατμοσφαιρικής ρύπανσης

Το πρώτο βήμα κατά τη διαδικασία της μοντελοποίησης προέκυψε από την ανάγκη εισαγωγής των μετρήσεων ατμοσφαιρικής ρύπανσης (ενότητα 3.1) στη βάση δεδομένων. Το σκεπτικό ήταν ότι θα έχουμε κάποιους πολίτες που θα εκτελούν καθημερινά στην πόλη τους κάποιες εξωτερικές δραστηριότητες. Θα μπορούν επίσης, με βάση την ικανοποίηση που πήραν από τη συγκεκριμένη δραστηριότητα, να την αξιολογούν με μια τιμή από 0 έως 10. Εμείς, θέλουμε να συσχετίσουμε την ικανοποίηση των πολιτών με την ατμοσφαιρική ρύπανση που υπάρχει στην πόλη. Έτσι, θα μπορούμε να προτείνουμε δραστηριότητες σε κάθε πολίτη ανάλογα με τις προτιμήσεις του και τα επίπεδα ρύπανσης που επικρατούν.

Ένα ερώτημα που προκύπτει επομένως έχει ως εξής: «Ποιο είναι τώρα το επίπεδο ατμοσφαιρικής ρύπανσης σε κάποια περιοχή;». Επειδή όμως, το επίπεδο ατμοσφαιρικής ρύπανσης εξαρτάται από τη συγκέντρωση που έχουν πέντε διαφορετικοί ρύποι (ενότητα 3.1), το παραπάνω ερώτημα θα μπορούσε να αναλυθεί σε επιμέρους ερωτήματα της μορφής: «Ποια είναι τώρα η συγκέντρωση ενός ρύπου σε κάποια περιοχή;».

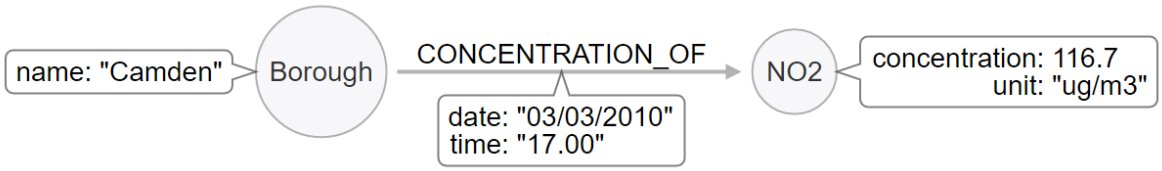
Εικόνα 4.3 Μοντέλο γράφων για την τιμή συγκέντρωσης ενός ρύπου σε μια περιοχή

Όπως φαίνεται στην παραπάνω εικόνα, από το τελευταίο ερώτημα μπορούμε να εντοπίσουμε τα εξής στοιχεία του γράφου:

- Κόμβους που αντιστοιχούν σε κάποια περιοχή και έχουν την *ετικέτα* "Borough".
- Ιδιότητες των κόμβων "Borough" με όνομα "name" και με τιμή το όνομα του κάθε Borough, π.χ. "Camden".
- Κόμβους που αντιστοιχούν σε κάποιο ρύπο και έχουν ως ετικέτα το όνομα του αντίστοιχου ρύπου, π.χ. "NO2" για το διοξείδιο του αζώτου.
- Τις ιδιότητες "concentration" και "unit" για τους κόμβους που αντιστοιχούν σε ρύπους.
- Τη σχέση "CONCENTRATION_OF" που συνδέει μια περιοχή με τη συγκέντρωση κάποιου ρύπου.

Το μόνο στοιχείο που δε φαίνεται να έχει μοντελοποιηθεί στην εικόνα είναι ο χρόνος. Στο ερώτημα που έχουμε θέσει, ο χρόνος κρύβεται πίσω από το 'τώρα' όταν ρωτάμε 'ποια είναι **τώρα** η συγκέντρωση ενός ρύπου;'.

Μια πρώτη ιδέα για την μοντελοποίηση του χρόνου, είναι να τον θεωρήσουμε ως *ιδιότητα* της σχέσης "CONCENTRATION_OF". Ο χρόνος προσδιορίζει τη σχέση αυτή και αποτελεί ένα είδος μεταδεδομένων που μας πληροφορεί για τη στιγμή που καταγράφηκε μια συγκέντρωση. Το μοντέλο που προκύπτει φαίνεται στην εικόνα που ακολουθεί.



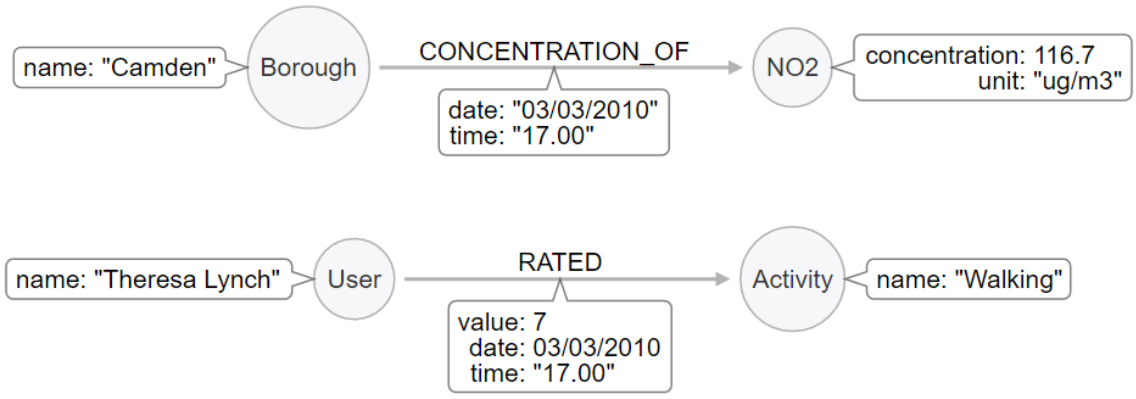
Εικόνα 4.4 Μοντελοποίηση του χρόνου ως ιδιότητα της σχέσης "concentration_of"

Εκ πρώτης όψεως, το μοντέλο που δημιουργήσαμε φαίνεται πως απαντάει στο ερώτημα που έχουμε θέσει. Εύκολα μπορούμε να δούμε τη συγκέντρωση που έχει ένας ρύπος σε μια περιοχή καθώς και τη στιγμή που έγινε η αντίστοιχη μέτρηση. Παρ' όλα αυτά, το συγκεκριμένο μοντέλο έχει δύο βασικά μειονεκτήματα που το καθιστούν ακατάλληλο για την εξυπηρέτηση των απαιτήσεων της εφαρμογής που θέλουμε να οικοδομήσουμε.

Για να εντοπίσουμε το πρώτο μειονέκτημα πρέπει να φανταστούμε πως θα εξελιχθεί ο γράφος μας με την προσθήκη πολλών δεδομένων συγκέντρωσης. Ουσιαστικά, έχουμε δημιουργήσει ένα μοναδικό κόμβο για μια περιοχή και πάνω σε αυτόν ενώνουμε χιλιάδες και εκατοντάδες χιλιάδες μετρήσεις συγκεντρώσεων. Οδηγούμαστε έτσι σε έναν υπερφορτωμένο κόμβο με ένα τεράστιο αριθμό σχέσεων, Ακόμα χειρότερα όμως, αν θέλουμε να βρούμε τη συγκέντρωση των ρύπων για μια συγκεκριμένη ημερομηνία, θα πρέπει να διατρέξουμε όλες τις σχέσεις "CONCENTRATION_OF' που συνδέονται στην επιθυμητή περιοχή και να κοιτάξουμε όλες τις ιδιότητες "date" και "time" των σχέσεων αυτών.

Το δεύτερο μειονέκτημα εντοπίζεται εάν λάβουμε υπόψιν τον τελικό μας σκοπό. Όπως αναφέρθηκε στην αρχή της παραγράφου, σκοπός μας είναι να συσχετίσουμε αξιολογήσεις πολιτών με μετρήσεις ατμοσφαιρικής ρύπανσης και καιρικών φαινομένων. Στην περίπτωση που διατηρήσουμε την

τρέχουσα μοντελοποίηση θα καταλήξουμε τελικά στους δυο ασύνδετους γράφους που απεικονίζονται παρακάτω:



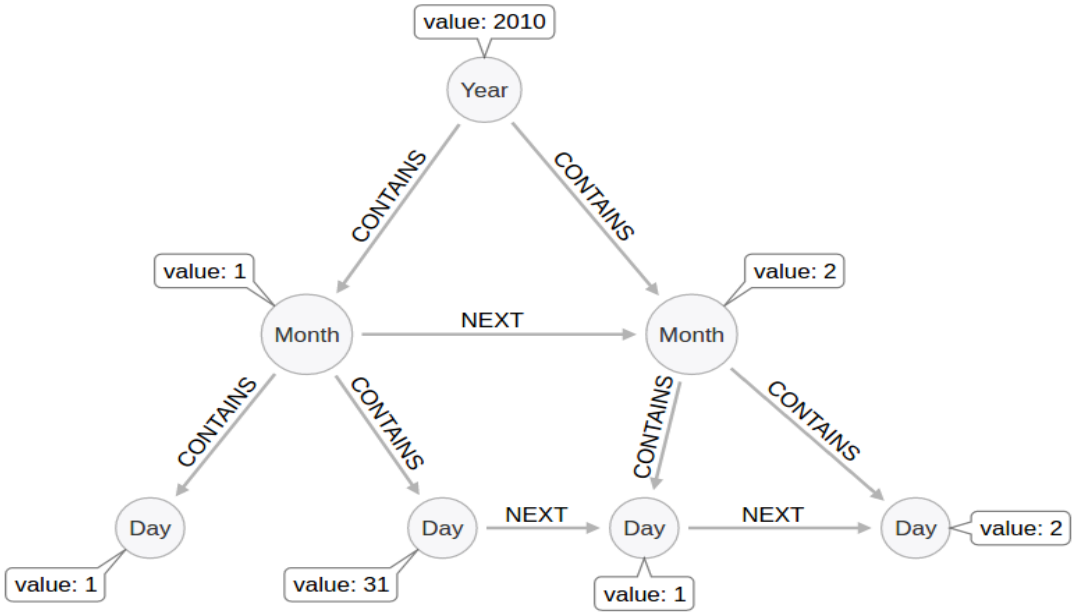
Εικόνα 4.5 Μοντελοποίηση του χρόνου ως *ιδιότητα* σχέσεων

Για να συσχετίσουμε τα δεδομένα των δυο γράφων θα πρέπει να διατρέξουμε τον κάθε ένα ξεχωριστά και στη συνέχεια να συγκρίνουμε τα αποτελέσματα των διασχίσεων για να καταλήξουμε σε κάποιο πιθανό συμπέρασμα. Μια διαδικασία αρκετά πολύπλοκη και χρονοβόρα που δεν ευνοεί τη δημιουργία του συστήματος συστάσεων πραγματικού χρόνου που θέλουμε να υλοποιήσουμε.

Το βασικό σημείο που αγνοήσαμε και οδηγηθήκαμε σε αδιέξοδο, είναι το γεγονός ότι ο 'χρόνος' αποτελεί θεμελιακό στοιχείο της τελικής μας εφαρμογής. Πρέπει λοιπόν, να αλλάξουμε τρόπο σκέψης και να μοντελοποιήσουμε το χρόνο ως τέτοιο. Η σκέψη αυτή μας οδηγεί στην επόμενη παράγραφο και στη δημιουργία των δέντρων χρόνου.

4.2.3 Δημιουργία δέντρων χρόνου

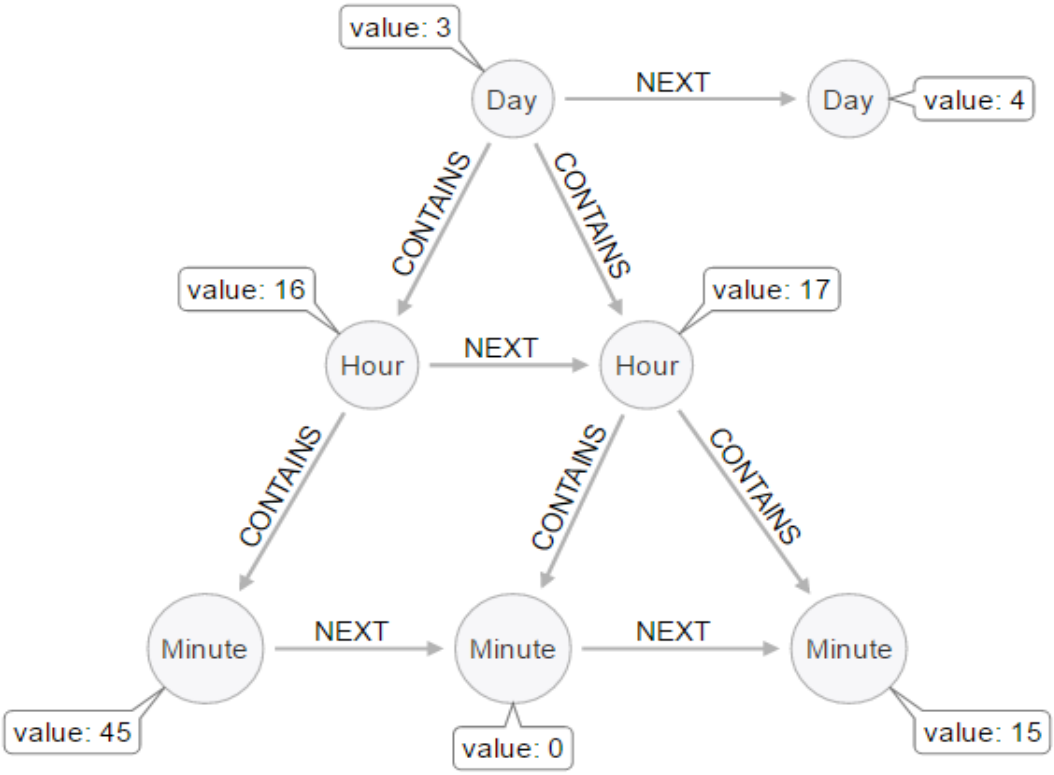
Ένας ενδιαφέρον τρόπος για τη μοντελοποίηση του χρόνου στη Neo4j είναι τα δέντρα χρόνου (time trees). Ένα δέντρο είναι μια υποκατηγορία γράφου με συγκεκριμένη ιεραρχία κόμβων και συγκεκριμένη δομή. Το δέντρο που θα χρησιμοποιηθεί στην εργασία παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 4.6 Μοντέλο δέντρων χρόνου (Έτη – Μήνες – Ημέρες)

Βλέπουμε ότι για τη δημιουργία του δέντρου έχουν χρησιμοποιηθεί *κόμβοι* για την αναπαράσταση των ετών, των μηνών και των ημερών. Κάθε κόμβος περιέχει μια ιδιότητα "value" με την τιμή που τον χαρακτηρίζει. Συνδέεται επίσης μέσω των *σχέσεων* "CONTAINS" με κατώτερης ιεραρχίας κόμβους (εάν τις ακολουθήσουμε με την κανονική τους φορά) και με ανώτερης ιεραρχίας κόμβους (εάν τις ακολουθήσουμε προς την αντίθετη φορά). Τέλος, κάθε κόμβος συνδέεται με κόμβους του ίδιου επιπέδου μέσω των *σχέσεων* "NEXT", τις οποίες μπορούμε να διασχίσουμε με την κανονική φορά, για να βρούμε μεταγενέστερους κόμβους, ή προς την αντίθετη φορά, για προγενέστερους κόμβους. Να σημειωθεί ότι οι ανώτεροι σε ιεραρχία κόμβοι είναι τα έτη, τα οποία επίσης συνδέονται μεταξύ τους με *σχέσεις* "NEXT".

Το δέντρο χρόνου που δημιουργήσαμε περιλαμβάνει τα έτη 2010 και 2011, τα οποία σχετίζονταν με τα διαθέσιμα δεδομένα. Όπως φαίνεται στην παραπάνω εικόνα, κάθε έτος περιέχει τους μήνες και τις ημέρες του, αλλά η ιεραρχία συνεχίζει μέχρι το βάθος λεπτών. Οι μέρες περιέχουν ώρες και οι ώρες λεπτά, όπως φαίνεται ακολούθως:



Εικόνα 4.7 Μοντέλο δέντρων χρόνου (Ημέρες – Ώρες – Λεπτά)

Οι ώρες του δέντρου αριθμούνται από 0 έως 23, ενώ στο δέντρο δεν περιέχονται όλα τα λεπτά, παρά μόνο τα τέσσερα τέταρτα της ώρας (0, 15, 30, 45). Αυτό στηρίζεται στο γεγονός ότι η δειγματοληψία των δεδομένων μας γίνεται κάθε 15 λεπτά. Αγνοώντας τα ενδιάμεσα λεπτά απλοποιούμε την πολυπλοκότητα του μοντέλου και μπορούμε να αναφερόμαστε σε κάθε τέταρτο της ώρας δείχνοντας στο πρώτο λεπτό του. Σε αυτό το λεπτό συνδέονται τόσο οι μετρήσεις των δεδομένων που έχουμε όσο και οι βαθμολογήσεις των χρηστών της εφαρμογής. Εάν θα μας ενδιέφερε η γνώση της πληροφορίας που θα περιεχόταν στα ενδιάμεσα λεπτά ή και στα δευτερόλεπτα, θα μπορούσαμε να αποθηκεύουμε επιπλέον μια *ιδιότητα* timestamp() στους κόμβους δεδομένων που θα συνδέονται στο δέντρο. Θα μπορούσαμε ακόμα, να επεκτείνουμε το μοντέλο ώστε να περιέχονται όλοι οι κόμβοι λεπτών και δευτερολέπτων. Για τους σκοπούς της

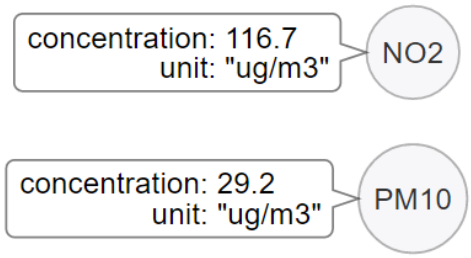
συγκεκριμένης εργασίας όμως, κάτι τέτοιο δεν θεωρήθηκε σκόπιμο και χρησιμοποιήθηκε τελικά το μοντέλο που παρουσιάστηκε παραπάνω.

Με τη χρήση των δέντρων χρόνου ξεπεράσαμε το πρόβλημα των δύο ασύνδετων γράφων που αναφέρθηκε στην παράγραφο 4.2.2. Τα δέντρα χρόνου αποτελούν πλέον θεμέλιο της βάσης δεδομένων που θα δημιουργήσουμε και πάνω τους θα συνδεθούν όλα τα υπόλοιπα δεδομένα.

Στην υλοποίησή μας τα δέντρα δημιουργήθηκαν εξ αρχής στη βάση δεδομένων για λόγους απλότητας υλοποίησης. Παρά το γεγονός αυτό, τόσο η δομή τους όσο και η χρήση των schema free βάσεων δεδομένων γράφων μας δίνουν τη δυνατότητα δυναμικής επέκτασής τους κατά την πορεία λειτουργίας της εφαρμογής.

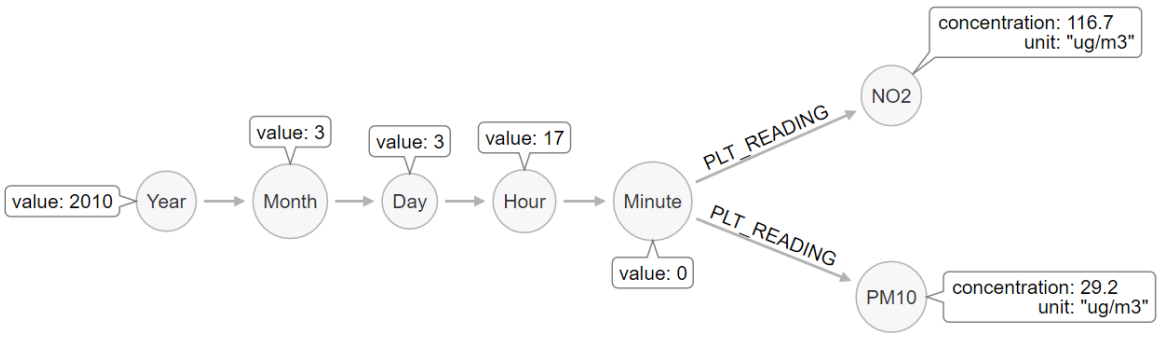
4.2.4 Σύνδεση δεδομένων ατμοσφαιρικής ρύπανσης στα δέντρα χρόνου

Έχοντας δημιουργήσει τα δέντρα χρόνου μπορούμε πλέον εύκολα να προχωρήσουμε στην σύνδεση των υπόλοιπων δεδομένων πάνω σε αυτά. Για την εισαγωγή των δεδομένων ατμοσφαιρικής ρύπανσης υπενθυμίζουμε ότι τα δεδομένα αποτελούνται από μετρήσεις συγκεντρώσεων πέντε διαφορετικών ρύπων. Η κάθε μέτρηση αποθηκεύεται ως τιμή μιας ιδιότητας “concentration” σε κόμβους ρύπων που χαρακτηρίζονται από ετικέτες με το όνομα του αντίστοιχου ρύπου (NO2, O3, SO2, PM10, PM25). Παράδειγμα δύο τέτοιων κόμβων φαίνεται στην εικόνα που ακολουθεί:



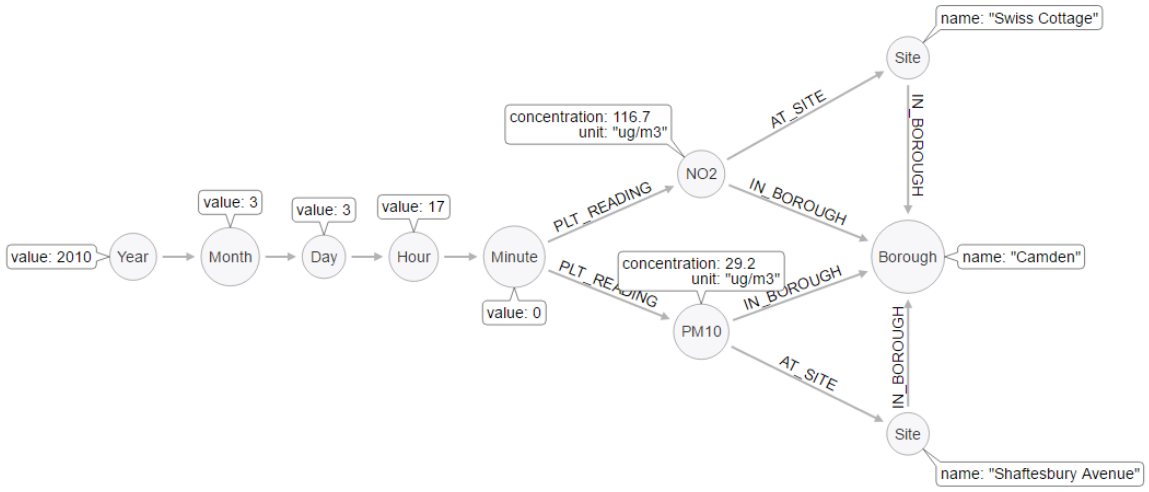
Εικόνα 4.8 Μοντέλο κόμβων ρύπων για την αποθήκευση των συγκεντρώσεων

Αφού δημιουργηθούν οι παραπάνω κόμβοι, συνδέονται μέσω μιας σχέσης «PLT_READING” πάνω στον κόμβο-λεπτό του δέντρου χρόνου στο οποίο αντιστοιχούν. Αν υποθέσουμε ότι οι παραπάνω μετρήσεις πάρθηκαν στις 03/03/2010 και ώρα 17:00, τότε το μοντέλο μας διαμορφώνεται ως εξής:



Εικόνα 4.9 Μοντελοποίηση της σχέσης “PLT_READING”

Επίσης, οι κόμβοι συγκεντρώσεων συνδέονται με κόμβους “Borough” και “Site” υποδεικνύοντας την περιοχή και τον τόπο που καταγράφηκε η μέτρηση, όπως παρουσιάζεται παρακάτω:

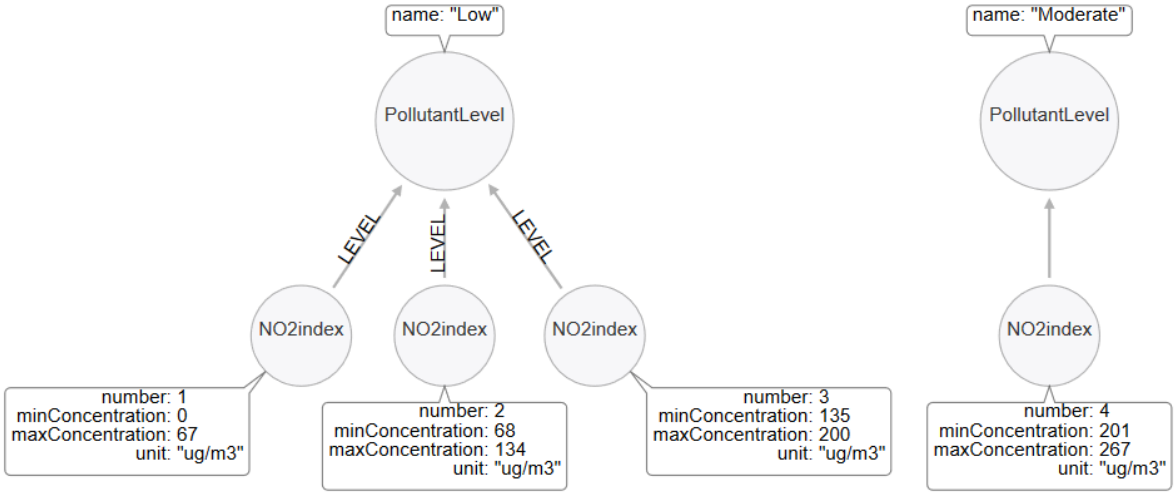


Εικόνα 4.10 Σύνδεση κόμβων ρύπων με “Site” και “Borough”

Με την ολοκλήρωση του παραπάνω μοντέλου είμαστε πλέον σε θέση να απαντήσουμε στο ερώτημα «ποια είναι τώρα η συγκέντρωση ενός ρύπου σε κάποια περιοχή;». Για να μπορέσουμε να απαντήσουμε όμως στο ερώτημα «ποιο είναι τώρα το επίπεδο ατμοσφαιρικής ρύπανση;» χρειάζεται να εμπλουτίσουμε λίγο το μοντέλο μας προσθέτοντας τα δεδομένα για τον ημερήσιο δείκτη ατμοσφαιρικής ρύπανσης και για τα επίπεδα ρύπανσης.

4.2.5 Εισαγωγή δεδομένων ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης

Για την εισαγωγή των δεδομένων ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης, που παρουσιάζονται στου πίνακες της ενότητας 3.1, μπορούμε να χρησιμοποιήσουμε τη μορφή του παρακάτω μοντέλου:



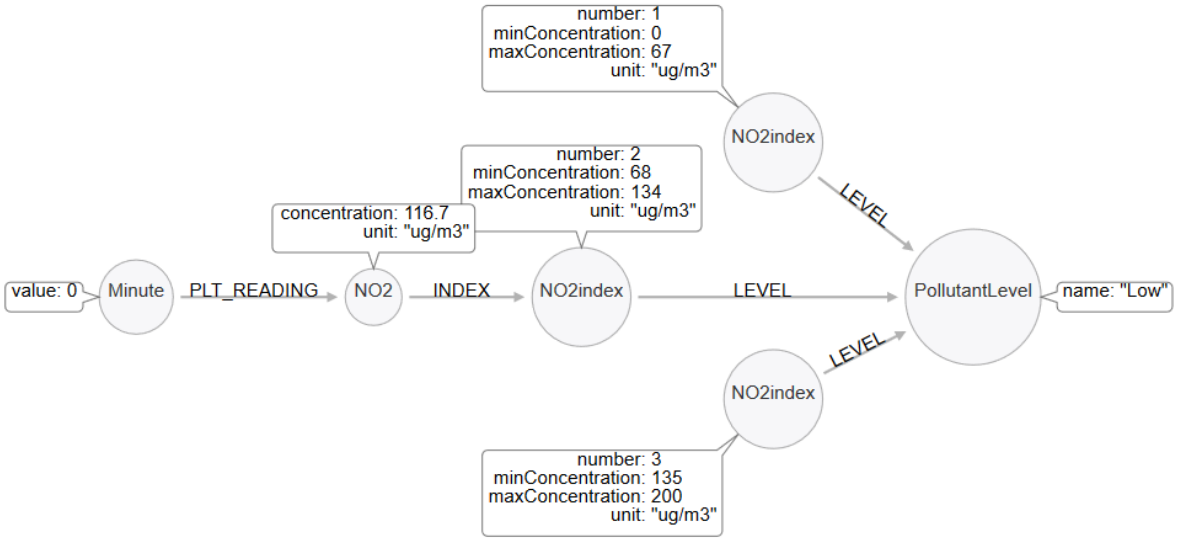
Εικόνα 4.11 Μοντελοποίηση δεδομένων ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης

Όπως φαίνεται στην εικόνα, το μοντέλο αποτελείται από κόμβους ‘PollutantLevel’ και κόμβους δεικτών, όπως ο ‘NO2index’. Συνολικά θα έχουμε τέσσερις κόμβους “PollutantLevel” με τιμές

“Low”, “Moderate”, “High” και “Very High”. Ενώ, στους κόμβους αυτούς θα συνδέονται οι κόμβοι-δείκτη κάθε ρύπου (“NO2index”, “O3index”, “SO2index”, “PM10index”, “PM25index”) με τον εξής τρόπο:

1. Όποιος κόμβος-δείκτη έχει *ιδιότητα* “number” με τιμή 1, 2 ή 3, συνδέεται μέσω της σχέσης “level” στο “PollutantLevel” με όνομα “Low”
2. Όποιος κόμβος-δείκτη έχει *ιδιότητα* “number” με τιμή 4, 5 ή 6, συνδέεται μέσω της σχέσης “level” στο “PollutantLevel” με όνομα “Moderate”
3. Όποιος κόμβος-δείκτη έχει *ιδιότητα* “number” με τιμή 7, 8 ή 9, συνδέεται μέσω της σχέσης “level” στο “PollutantLevel” με όνομα “High”
4. Όποιος κόμβος-δείκτη έχει *ιδιότητα* “number” με τιμή 10, συνδέεται μέσω της σχέσης “level” στο “PollutantLevel” με όνομα “Very High”

Μετά τη δημιουργία του μοντέλου μπορούμε πλέον να συνδέσουμε τους κόμβους συγκεντρώσεων στις τιμές του δείκτη DAQI όπως φαίνεται στην εικόνα που ακολουθεί:



Εικόνα 4.12 Μοντελοποίηση της σχέσης “INDEX” για τους κόμβους ρύπων

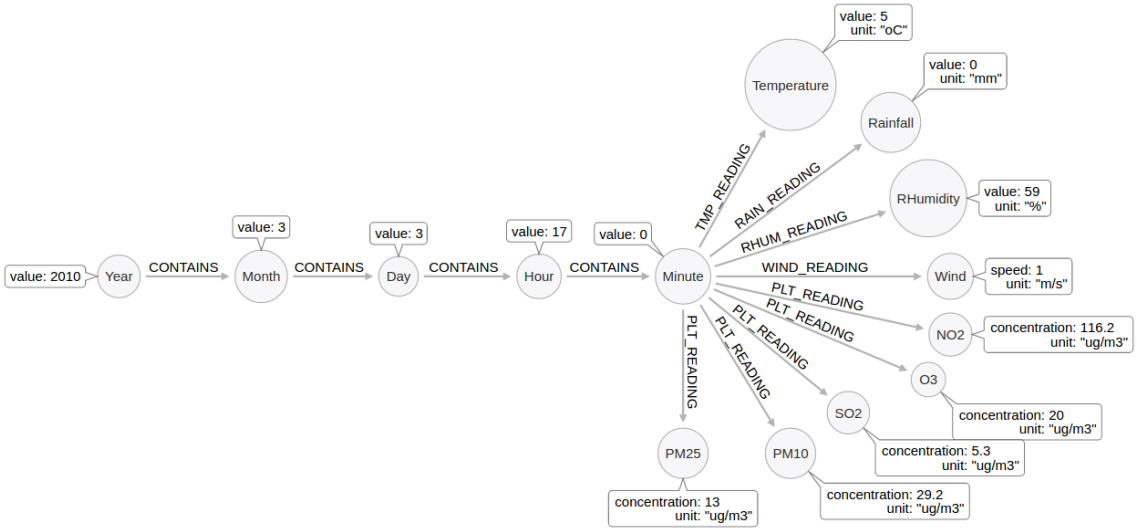
Έχομε ολοκληρώσει, έτσι, την αναπαράσταση του πεδίου που μελετούσαμε έως τώρα. Μπορούμε να απαντήσουμε στο ερώτημα «ποιο είναι τώρα το επίπεδο ατμοσφαιρικής ρύπανσης σε κάποια περιοχή;» και να συνεχίσουμε με την εισαγωγή των υπόλοιπων δεδομένων.

4.2.6 Εισαγωγή δεδομένων καιρικών συνθηκών

Ακολουθώντας την ίδια λογική που μελετήσαμε στις προηγούμενες παραγράφους, εισάγουμε και τα δεδομένα των καιρικών συνθηκών. Πρώτα δημιουργούμε τους κόμβους στους οποίους θα αποθηκευτούν οι μετρήσεις μας:

Εικόνα 4.13 Μοντέλα κόμβων καιρικών συνθηκών

και ύστερα συνδέουμε τους κόμβους που δημιουργήθηκαν, με τον αντίστοιχο κόμβο-λεπτό των δέντρων χρόνου:



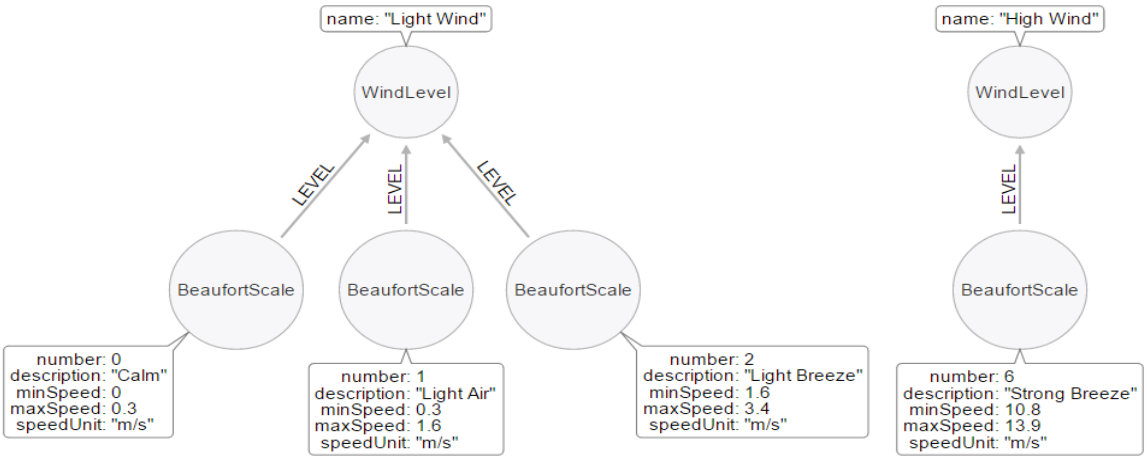
Εικόνα 4.14 Σύνδεση κόμβων καιρικών συνθηκών στα δέντρα χρόνου

4.2.7 Εισαγωγή δεδομένων κλίμακας μποφόρ

Στην παράγραφο 4.2.5 δείξαμε τη διαδικασία σύνδεσης των συγκεντρώσεων του κάθε ρύπου με τον ημερήσιο δείκτη ατμοσφαιρικής ρύπανσης. Με τον τρόπο αυτό παρουσιάσαμε ένα πιο ολοκληρωμένο μοντέλο της πραγματικότητας και προσθέσαμε στο γράφο μας την έννοια των επιπέδων ρύπανσης. Όμοια αποφασίσαμε να εργαστούμε και για τις ταχύτητες των ανέμων. Με σκοπό τη δημιουργία σημασιολογίας που θα μας παρέχει συμπεράσματα σε πιο φιλική μορφή για τους χρήστες της εφαρμογής, προσθέσαμε στο γράφο μας ένα επίπεδο κόμβων με την κλίμακα μποφόρ και ένα ακόμα με τα επίπεδα ανέμων.

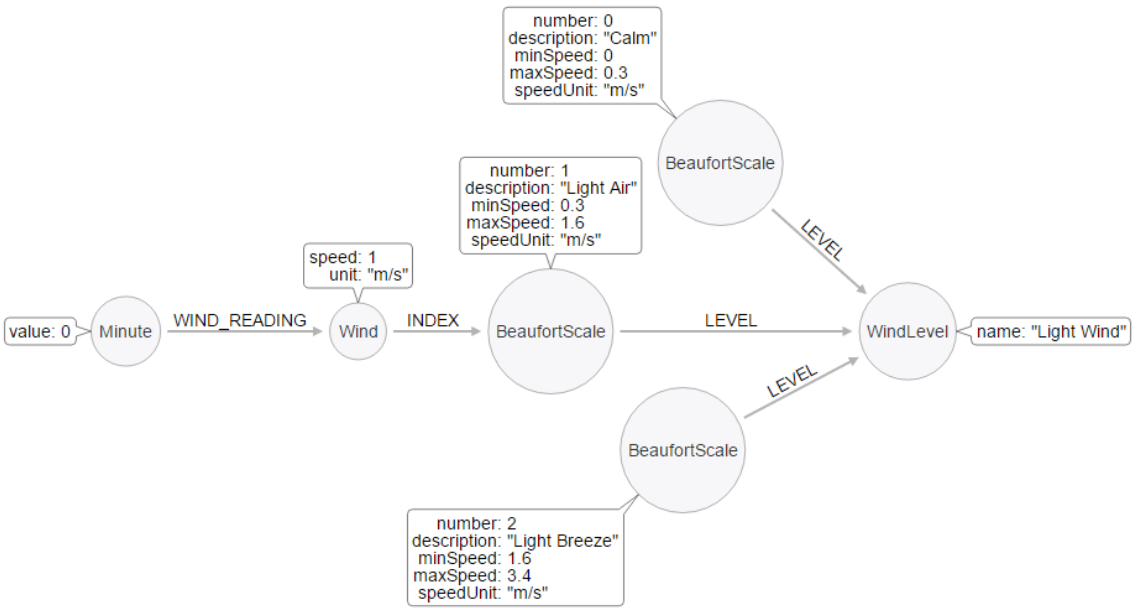
Οι κόμβοι της κλίμακας μποφόρ είναι δεκατρείς, με τιμές από 0 έως 12 και, εκτός από την τιμή τους, περιέχουν τα όρια της ταχύτητας ανέμων σε m/s που τους αντιστοιχούν. Τα επίπεδα των ανέμων είναι πέντε: Light Wind, High Wind, Gale Force, Storm Force, Hurricane. Αναλυτικότερες πληροφορίες περιέχονται στην παράγραφο 3.2.4.

Ένα ενδεικτικό κομμάτι του γράφου για τα δεδομένα της κλίμακας μποφόρ φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.15 Μοντελοποίηση δεδομένων κλίμακας μποφόρ

ενώ, η σύνδεση των κόμβων-ανέμων με την κλίμακα μποφόρ έχει ως εξής:

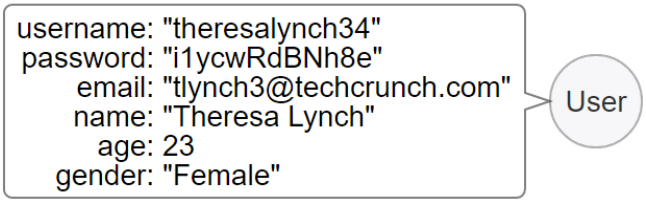


Εικόνα 4.16 Μοντελοποίηση της σχέσης "INDEX" για τους κόμβους των ανέμων

Στο σημείο αυτό έχουμε ολοκληρώσει τη μοντελοποίηση του πεδίου μας όσον αφορά τον τόπο, τον χρόνο και τα διαθέσιμα δεδομένα μετρήσεων. Είμαστε σε θέση να απαντήσουμε σε οποιαδήποτε ερώτημα σχετικό με τα επίπεδα συγκέντρωσης των ρύπων ή τις καιρικές συνθήκες και για οποιαδήποτε στιγμή. Επόμενο βήμα είναι η εισαγωγή των χρηστών και η σύνδεσή τους με όσα είπαμε παραπάνω, για τη δημιουργία ενός ενιαίου μοντέλου έτοιμου να απαντήσει στα ερωτήματα συστάσεων για τα οποία προορίζεται.

4.2.8 Εισαγωγή χρηστών

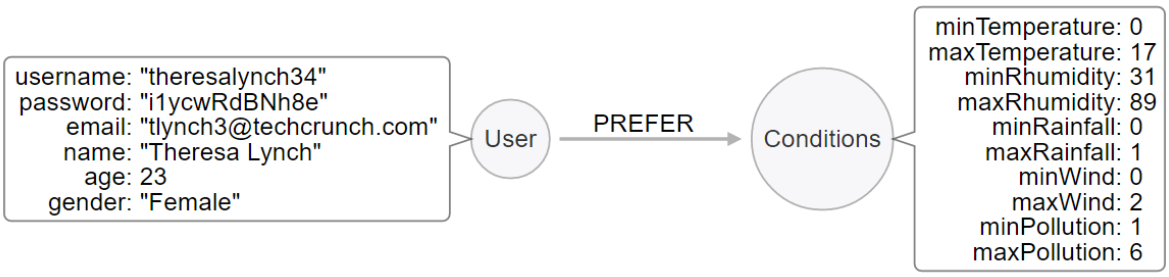
Η μοντελοποίηση των χρηστών είναι σχετικά απλή υπόθεση, αφού όπως είναι λογικό, κάθε χρήστης θα αντιπροσωπεύεται από ένα κόμβο με την ετικέτα "User". Ο κόμβος θα περιέχει τις απαραίτητες πληροφορίες για των προσδιορισμό του κάθε χρήστη σε μορφή *ιδιοτήτων*. Μερικές από αυτές είναι το username, το email, η ηλικία (age), το φύλο (gender). Ένας τυπικός κόμβος χρήστη παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 4.17 Μοντέλο για τους κόμβους χρηστών

4.2.9 *Προτιμήσεις καιρικών φαινομένων και επιπέδων ρύπανσης*

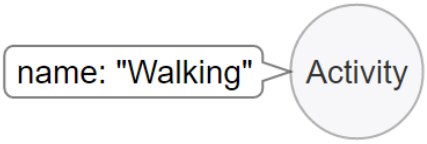
Κάθε χρήστης, μετά την εγγραφή τους στην εφαρμογή, θα μπορεί να δηλώσει κάποιες προτιμήσεις σε καιρικά φαινόμενα και σε επίπεδα ρύπανσης. Αυτές οι προτιμήσεις θα αποθηκεύονται σε ένα κόμβο "Conditions" και θα συνδέονται με το χρήστη μέσω της σχέσης "PREFER". Το μοντέλο που προκύπτει παρουσιάζεται ακολούθως:



Εικόνα 4.18 Μοντελοποίηση προτιμήσεων χρήστη σε καιρικές συνθήκες και επίπεδο ρύπανσης

4.2.10 *Δραστηριότητες*

Οι δραστηριότητες μοντελοποιούνται ως κόμβοι "Activity" με μια απλή ιδιότητα "name", την ονομασία δηλαδή της δραστηριότητας. Ένας κόμβος της δραστηριότητας "Walking" φαίνεται στην ακόλουθη εικόνα:



Εικόνα 4.19 Μοντέλο για τους κόμβους δραστηριοτήτων

Παρά την απλότητα του συγκεκριμένου μοντέλου, οι δραστηριότητες παίζουν ίσως το σημαντικότερο ρόλο για την επίτευξη των στόχων της εργασίας. Οι δραστηριότητες είναι τα αντικείμενα που θα προτείνονται στους χρήστες της εφαρμογής και πάνω σε αυτές θα συνδέονται τόσο οι προτιμήσεις των χρηστών όσο και οι αξιολογήσεις τους. Όπως θα δούμε και στις παραγράφους που ακολουθούν, με μια απλή ματιά στο γράφο, ξεκινώντας από τον κόμβο κάποιας δραστηριότητας, θα μπορούμε να βρούμε όλους τους χρήστες που σχετίζονται με αυτή.

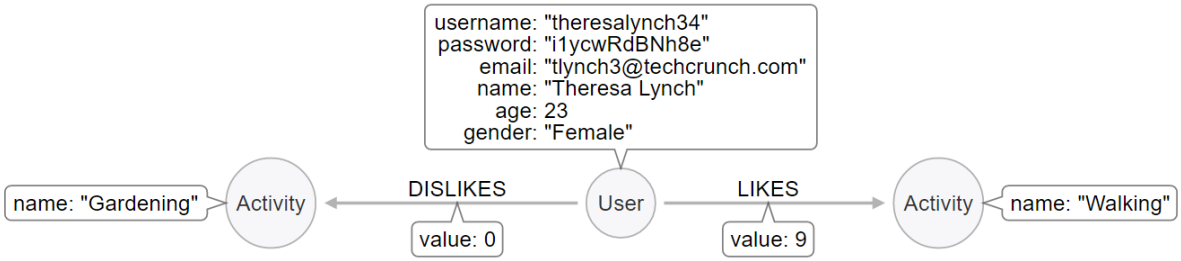
Συνεπώς παρατηρούμε ότι οι δραστηριότητες είναι οι κόμβοι που συνδέουν τους χρήστες μεταξύ τους και μας επιτρέπουν να συμπεράνουμε ομοιότητες και διαφορές πολύ χρήσιμες για την υλοποίηση του συστήματος συστάσεων.

4.2.11 *Προτιμήσεις δραστηριοτήτων*

Οι προτιμήσεις δραστηριοτήτων είναι τα 'Likes' και τα 'Dislikes' που καταχωρούν οι χρήστες σε σχέση με τις δραστηριότητες της εφαρμογής. Οι προτιμήσεις αυτές είναι γενικές και ουσιαστικά αντιπροσωπεύουν τα χόμπι του κάθε χρήστη. Εάν δηλαδή σε κάποιον αρέσει το περπάτημα σαν

εξωτερική δραστηριότητα, τότε ο κόμβος του χρήστη συνδέεται με τη δραστηριότητα “Walking” με τη βοήθεια μιας σχέσης “LIKES”. Εάν αντίθετα, κάποιος χρήστης αντιπαθεί πλήρως μια δραστηριότητα μπορεί να συνδεθεί με αυτή μέσω της σχέσης “DISLIKES”. Επιπροσθέτως, μια χρήσιμη πληροφορία για την παροχή συστάσεων είναι η αποθήκευση βαρών (weights) στις παραπάνω σχέσεις. Αυτά τα βάρη θα μας ενημερώνουν για το πόσο πολύ αρέσει σε κάποιον μια ορισμένη δραστηριότητα. Για τις σχέσεις “LIKES” ο χρήστης έχει τη δυνατότητα να ορίσει την προτίμηση του χρησιμοποιώντας μια τιμή από 1 έως 10, ενώ στις σχέσεις “DISLIKES” αποθηκεύεται αυτόματα η τιμή 0.

Μια αναπαράσταση των σχέσεων ‘likes’ και ‘dislikes’ φαίνεται στην ακόλουθη εικόνα:

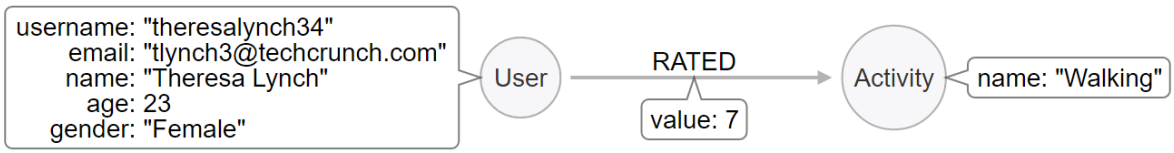


Εικόνα 4.20 Μοντελοποίηση των σχέσεων “likes” και “dislikes”

4.2.12 Αξιολόγηση δραστηριοτήτων

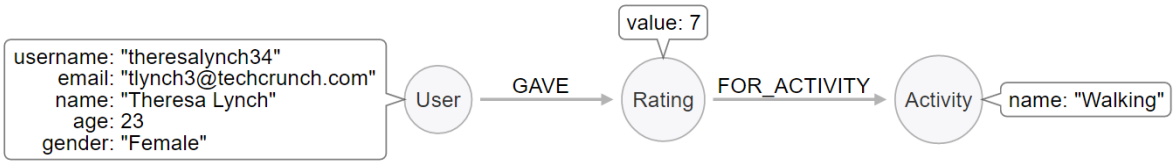
Η τελευταία παράγραφος της μοντελοποίησης σχετίζεται με την αξιολόγηση των δραστηριοτήτων. Όπως έχουμε αναφέρει και στην αρχή του κεφαλαίου, κάθε χρήστης θα εκτελεί ορισμένες εξωτερικές δραστηριότητες. Ανάλογα με την ικανοποίηση του από αυτές, θα μπορεί να τις αξιολογεί χρησιμοποιώντας την κλίμακα από 0 έως 10.

Μια μοντελοποίηση της αξιολόγησης δραστηριοτήτων θα μπορούσε να είναι ως εξής:



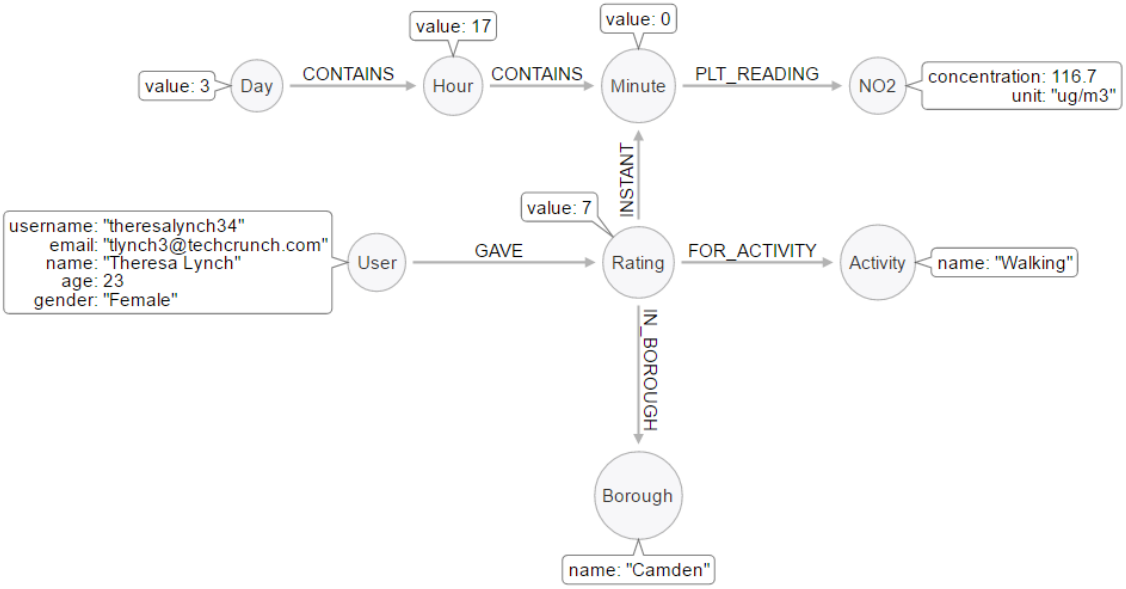
Εικόνα 4.21 Μοντελοποίηση της αξιολόγησης δραστηριοτήτων με χρήση της σχέσης “RATED”

Παρόλα αυτά, η προηγούμενη μοντελοποίηση εμφανίζει τα ίδια προβλήματα “χρόνου” που παρουσιάστηκαν στην παράγραφο 4.2.2. Στην περίπτωση αυτή, ενώ έχουμε ήδη τους κόμβους του χρήστη και της δραστηριότητας, εάν τους συνδέσουμε απευθείας δεν εκμεταλλευόμαστε την δομή των δέντρων χρόνου που έχουμε κατασκευάσει. Πως όμως μπορούμε να συνδέσουμε τη σχέση “RATED” με το χρόνο; Η απάντηση είναι ότι δεν μπορούμε, καθώς σε ένα γράφο δεν είναι ποτέ δυνατό να συνδέσουμε μια σχέση πάνω σε μια άλλη σχέση. Η λύση στο πρόβλημα αυτό δίνεται με τη χρήση ενός ενδιάμεσου κόμβου:



Εικόνα 4.22 Μοντελοποίηση της αξιολόγησης δραστηριοτήτων με χρήση ενδιάμεσου κόμβου “Rating”

Βλέπουμε, ότι αντί να πούμε πως κάποιος χρήστης βαθμολόγησε μια δραστηριότητα, λέμε ότι ο χρήστης καταχώρησε μια βαθμολογία για κάποια δραστηριότητα. Εύκολα, τώρα, μπορούμε να συνδέσουμε τον κόμβο “Rating” τόσο με τα δέντρα χρόνου που έχουμε δημιουργήσει όσο και με την περιοχή στην οποία πραγματοποιήθηκε η δραστηριότητα και η αξιολόγησή της:



Εικόνα 4.23 Σύνδεση “Rating” στα δέντρα χρόνου και στους κόμβους “Borough”

Συνοψίζοντας, στο κεφάλαιο 4 αναλύσαμε την πορεία δημιουργίας των μοντέλων γράφων που θα χρησιμοποιηθούν στην εφαρμογή. Δημιουργήσαμε δέντρα χρόνου που προχωράνε μέχρι το βάθος «λεπτών» (Minutes) και πάνω σε αυτά συνδέσαμε τις διαθέσιμες μετρήσεις καιρικών συνθηκών και ατμοσφαιρικής ρύπανσης. Τέλος, δημιουργήσαμε χρήστες που συνδέονται με τις προτιμήσεις τους σε «συνθήκες» και «δραστηριότητες» και έχουν τη δυνατότητα να βαθμολογούν καθημερινά την εκτέλεση διαφόρων δραστηριοτήτων. Στο κεφάλαιο που ακολουθεί παρουσιάζεται η διαδικασία υλοποίησης του συστήματος.

5 Υλοποίηση Βάσης Δεδομένων Γράφων

Στο κεφάλαιο αυτό θα παρουσιάσουμε την πορεία υλοποίησης της βάσης δεδομένων γράφων. Ξεκινάμε με μια ενότητα όπου περιγράφουμε όλα τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν και τη διαδικασία εγκατάστασης τους. Συνεχίζουμε με μία σύντομη παρουσίαση της γλώσσας ερωτημάτων Cypher και καταλήγουμε στην αναλυτική παρουσίαση της υλοποίησης.

5.1 Προγραμματιστικά εργαλεία και εγκατάσταση

5.1.1 Προγραμματιστικά εργαλεία

Η εργασία υλοποιήθηκε σε διανομή linux (Ubuntu 16.04 LTS) με χρήση της βάσης δεδομένων γράφων Neo4j και της γλώσσας ερωτημάτων Cypher. Οι εκδόσεις των εργαλείων που χρησιμοποιήθηκαν είναι:

- Neo4j-community-3.1.1-unix
- Cypher v.3.1

Έγινε επίσης δοκιμή της υλοποίησης σε Windows 10 με χρήση των εκδόσεων:

- Neo4j-community_windows-x64_3_1_1
- Cypher v.3.1

5.1.2 Προδιαγραφές συστήματος

Όλες οι δοκιμές έγιναν σε προσωπικό υπολογιστή με χαρακτηριστικά:

- Processor: Intel® Core™ i5-5200 CPU @ 2.20 GHz 2.20 GHz
- RAM: 8,00 GB (4,00 GB σε χρήση)
- System type: 64-bit Operating System, x64-based processor
- HD Free space: 500 MB

Η τελική υλοποίηση αποτελείται από μια βάση δεδομένων γράφων με:

- 460.107 κόμβους (nodes) και
- 3.496.318 σχέσεις-ακμές (relationships)

5.1.3 Εγκατάσταση σε linux

Για την εγκατάσταση της Neo4j σε linux ακολουθούμε τα παρακάτω βήματα:

1. Κατεβάζουμε την έκδοση neo4j-community-3.1.1 από εδώ.

2. Αποσυμπιέζουμε το φάκελο με τα αρχεία της Neo4j στο επιθυμητό directory.
3. Ανοίγουμε ένα νέο terminal (ctrl + alt + t)
4. Μεταβαίνουμε στο παραπάνω directory με χρήση της εντολής cd

   Για παράδειγμα,

   ```
   cd ./Desktop/neo4jfolder
   ```

5. Εκτελούμε τη Neo4j με την εντολή,

   ```
   ./bin/neo4j console
   ```

   Εναλλακτικά, αντί για 'neo4j console' μπορούμε να χρησιμοποιούμε τις εντολές: 'neo4j start' για εκκίνηση της Neo4j και 'neo4j stop' για τερματισμό.
6. Μεταβαίνουμε στη διεύθυνση http://localhost:7474 από τον web browser της επιλογής μας.
7. Αλλάζουμε τον κωδικό για το λογαριασμό 'neo4j'.

Για τη δημιουργία της βάσης δεδομένων που υλοποιήθηκε στην παρούσα εργασία, ακολουθούμε (σε linux) τα παρακάτω βήματα:

1. Κατεβάζουμε και αποσυμπιέζουμε το φάκελο με τα αρχεία (βλέπε παράρτημα).
2. Πηγαίνουμε στο φάκελο της Neo4j (στο directory δηλαδή του βήματος 2 τη διαδικασίας εγκατάστασης).
3. Μέσα στο φάκελο 'import' δημιουργούμε ένα νέο φάκελο με όνομα 'thesis_vcharlaftis'.
4. Μέσα στο φάκελο 'thesis_vcharlaftis' βάζουμε τους τρείς φακέλους (airpollution, users, weather) με τα csv αρχεία που υπάρχουν στο φάκελο data της εργασίας.
5. Εκτελούμε τη Neo4j (βλέπε το βήμα 5 της εγκατάστασης)
6. Σε ένα νέο terminal ανοίγουμε το neo4j-shell με την εντολή:

   ```
   /directory/of/noe4j/folder/bin/neo4j-shell
   ```

7. Στο neo4j-shell κάνουμε copy-paste όλο τον κώδικα που υπάρχει στο αρχείο cypher.cql της εργασίας μας και περιμένουμε να ολοκληρωθεί η εκτέλεση όλων των εντολών.

Σημείωση: Η παραμετροποίηση της Neo4j σε linux γίνεται από το αρχείο 'neo4j.conf' που υπάρχει στο: `/directory/of/noe4j/folder/conf/`. Από το αρχείο αυτό μπορούμε να ρυθμίσουμε:

- Το όνομα της βάσης δεδομένων:

  # The name of the database to mount
- Το Java Heap Size:

  # Java Heap Size: by default the Java heap size is dynamically

  # calculated based on available system resources.

  # Uncomment these lines to set specific initial and maximum

  # heap size.
- Καθώς και να ενεργοποιήσουμε το neo4j-shell:

  # Enable a remote shell server which Neo4j Shell clients can log in to.

5.1.4 Εγκατάσταση σε windows

Για την εγκατάσταση της Neo4j σε windows ακολουθούμε τα παρακάτω βήματα:

1. Κατεβάζουμε την έκδοση neo4j-community-3_1_1 από εδώ (ή για 32 bit εδώ).
2. Εκτελούμε το αρχείο (exe) και ακολουθούμε τις οδηγίες εγκατάστασης.
3. Μόλις ολοκληρωθεί η εγκατάσταση τρέχουμε τη Neo4j και μας ανοίγει ένα νέο παράθυρο διαλόγου.
4. Εκεί επιλέγουμε την τοποθεσία που θέλουμε να αποθηκευτεί η βάση μας και πατάμε 'start'.
5. Μόλις στο 'Status' εμφανίσει την ένδειξη "Neo4j is ready", μεταβαίνουμε στη διεύθυνση http://localhost:7474/ από τον web browser της επιλογής μας.
6. Αλλάζουμε τον κωδικό για το λογαριασμό 'neo4j'.

Για τη δημιουργία της βάσης δεδομένων που υλοποιήθηκε στην παρούσα εργασία, ακολουθούμε (σε windows) τα παρακάτω βήματα:

1. Κατεβάζουμε και αποσυμπιέζουμε το φάκελο με τα αρχεία (βλέπε παράρτημα).
2. Δημιουργούμε ένα νέο φάκελο με το όνομα 'import' μέσα στην τοποθεσία που έχουμε επιλέξει στο 'βήμα 4' της εγκατάστασης.
3. Αντιγράφουμε (copy) τους τρεις φακέλους (airpollution, users, weather) με τα δεδομένα που υπάρχουν μέσα στο φάκελο 'data' της εργασίας και κάνουμε επικόλληση (paste) μέσα στο φάκελο import που δημιουργήσαμε στο 'βήμα 1'.
4. Ανοίγουμε τη Neo4j (βλέπε διαδικασία εγκατάσταση βήμα 4 και βήμα 5).
5. Τρέχουμε μια – μια της εντολές που υπάρχουν στο αρχείο 'windows_cypher.csv' από τη γραμμή εντολών του web browser στη διεύθυνση http://localhost:7474/ (σημείωση: κάθε εντολή τελειώνει με ';').

   Εναλλακτικά, μπορούμε να τρέξουμε μαζικά όλες τις εντολές χρησιμοποιώντας κάποια από τα εργαλεία μαζικής εκτέλεσης εντολών που παρέχει η Neo4j για windows.

5.2 Συνοπτική παρουσίαση της γλώσσας Cypher

Η υλοποίηση της εργασίας έγινε με χρήση της γλώσσας ερωτημάτων Cypher. Η cypher είναι μια δηλωτική (declarative) γλώσσα ερωτημάτων εμπνευσμένη από την SQL. Το μεγάλο της πλεονέκτημα είναι το συντακτικό της, το οποίο σου δίνει την εντύπωση ότι ζωγραφίζεις τους κόμβους και τα μονοπάτια του γράφου που αναζητάς με χρήση χαρακτήρων ascii.

Εικόνα 5.1 Αναπαράσταση κόμβων και μονοπατιών με χρήση της γλώσσας Cypher

Παρακάτω παρουσιάζουμε εν συντομία τα βασικά χαρακτηριστικά της:

5.2.1 Κόμβοι (nodes).

Η αναπαράσταση των κόμβων στη cypher γίνεται με τη χρήση παρενθέσεων. Οι κενές παρενθέσεις `( )` ταιριάζουν με οποιονδήποτε κόμβο στη βάση μας, ενώ αν θέλουμε να μπορούμε να αναφερθούμε σε κάποιο κόμβο, δίνουμε ένα όνομα μεταβλητής μέσα σε αυτόν π.χ. `(node)` ή `(u)`.

Οι κόμβοι στη Neo4j μπορεί να περιέχουν *ετικέτες*. Στη cypher οι ετικέτες ακολουθούν πάντα μετά από άνω και κάτω τελεία ':' εντός των παρενθέσεων του κόμβου, π.χ. `(:Label)` ή `(node:Label)`.

Κάθε κόμβος μπορεί επίσης να έχει μια ή περισσότερες ιδιότητες. Οι ιδιότητες στη cypher περιέχονται μέσα σε άγκιστρα όπως για παράδειγμα το όνομα κάποιου χρήστη `(u:User {name: "Theresa Lynch"})`.

5.2.2 Σχέσεις (relationships).

Οι σχέσεις στη cypher παριστάνονται με δύο παύλες - - και τη χρήση του μεγαλύτερο > ή μικρότερο < για τον προσδιορισμό της κατεύθυνσης. Έτσι, η αναπαράσταση `(a)--(b)` δείχνει ότι μια σχέση συνδέει τους κόμβους a και b χωρίς να μας ενδιαφέρει η κατεύθυνση, ενώ η αναπαράσταση `(a)-->(b)` δείχνει μια σχέση από τον κόμβο a προς τον κόμβο b.

Στις σχέσεις, όπως και στους κόμβους, μπορούμε να δώσουμε κάποιο όνομα μεταβλητής. Για να μπορέσουμε να ορίσουμε κάποια μεταβλητή σε μια σχέση χρησιμοποιούμε αγκύλες. Για παράδειγμα `(a)-[r]->(b)`, όπου στη σχέση που συνδέει τους κόμβους a και b έχουμε δέσει τη μεταβλητή r.

Τα ονόματα των σχέσεων, όπως και οι ετικέτες των κόμβων, ακολουθούν πάντα μετά από άνω και κάτω τελεία ':', ενώ οι ιδιότητες περιέχονται μέσα σε άγκιστρα: `(a)-[l:LIKES {val:5}]->(b)`

5.2.3 Εκφράσεις (clauses).

Υπάρχει ένα πλούσιο ρεπερτόριο εκφράσεων που μπορούν να χρησιμοποιηθούν. Παρακάτω αναφέρουμε επιγραμματικά κάποιες βασικές εκφράσεις:

`MATCH`

Το match εντοπίζει τα μοτίβα που του ορίζουμε μέσα στη βάση δεδομένων γράφων.

`WHERE`

Το where παρέχει κριτήρια για το φιλτράρισμα των μοτίβων.

`CREATE`

Το create δημιουργεί κόμβους και σχέσεις μέσα στη βάση δεδομένων.

`MERGE`

Το merge διασφαλίζει ότι τα μοτίβα που του παρέχουμε υπάρχουν μέσα στη βάση δεδομένων, αλλιώς τα δημιουργεί.

`DELETE`

Το delete διαγράφει κόμβους, σχέσεις και ιδιότητες από τη βάση δεδομένων.

`SET`

Με το set ορίζουμε τις τιμές των ιδιοτήτων που περιέχονται σε κόμβους ή σε σχέσεις.

`WITH`

Το with μας επιτρέπει να συνδέουμε ερωτήματα και να μεταφέρουμε αποτελέσματα από το ένα στο άλλο.

`LOAD CSV`

Το load csv επιτρέπει την εισαγωγή δεδομένων από csv αρχεία μέσα στη βάση δεδομένων.

5.3 Διαδικασία υλοποίησης

Η υλοποίηση που θα ακολουθήσει βασίζεται στα μοντέλα που παρουσιάσαμε στο κεφάλαιο 4 και στα δεδομένα που περιγράψαμε στο κεφάλαιο 3. Θα γίνει παρουσίαση της πορείας υλοποίησης με αναφορές σε στιγμιότυπα του γράφου που παράγονται και σε κομμάτια κώδικα cypher που χρησιμοποιούνται σε κάθε βήμα. Για το πλήρες κείμενο του κώδικα παραπέμπουμε τους αναγνώστες στο παράρτημα που υπάρχει στο τέλος της εργασίας.

5.3.1 Δημιουργία ευρετηρίων (indexes) και περιορισμών (constraints)

Πριν ξεκινήσουμε να βλέπουμε τον κώδικα της υλοποίησης θα αναφερθούμε στα ευρετήρια (indexes) και στους περιορισμούς (constraints). Όπως κάθε βάση δεδομένων έτσι και η Neo4j διαθέτει ευρετήρια. Τα *ευρετήρια* διευκολύνουν τη γρήγορη εύρεση δεδομένων μέσα στη βάση δεδομένων και στη Neo4j χρησιμοποιούνται επίσης ως σημεία πρόσδεσης *περιορισμών*.

Τα *ευρετήρια* της Neo4j δημιουργούνται μια φορά με τη χρήση του `CREATE INDEX ON` . Αφορούν μια συγκεκριμένη ιδιότητα ενός συνόλου κόμβων και εφόσον δημιουργηθούν χρησιμοποιούνται αυτόματα από τη βάση δεδομένων όταν χρειάζεται να γίνει εύρεση των συγκεκριμένων κόμβων. Ένα παράδειγμα δημιουργίας ευρετηρίου για το όνομα χρήστη των κόμβων `:User` έχει ως εξής:

```
CREATE INDEX ON :User(username);
```

Μπορούμε επίσης να αφαιρέσουμε κάποιο ευρετήριο με χρήση του `DROP INDEX ON` ως εξής:

```
DROP INDEX ON :User(username);
```

Τα *ευρετήρια* δημιουργούνται συχνά σε ιδιότητες κόμβων για της οποίες χρησιμοποιούμε την εντολή `MERGE` και είναι επίσης καλή πρακτική να δημιουργούνται σε ιδιότητες κόμβων τους οποίους χρησιμοποιούμε πολλές φορές ως αρχή των ερωτημάτων μας.

Οι *περιορισμοί* αναφέρονται σε ιδιότητες κόμβων και σκοπός τους είναι να εγκαθιδρύσουν κάποιο περιορισμό μοναδικότητας ή ύπαρξης. Παράδειγμα δημιουργίας περιορισμού μοναδικότητας στο όνομα χρήστη των κόμβων `:User` έχει ως εξής:

```
CREATE CONSTRAINT ON (u:User) ASSERT u.username IS UNIQUE;
```

Όμοια με τα *ευρετήρια* μπορούμε να αφαιρέσουμε κάποιο περιορισμός με χρήση του `DROP`:

```
DROP CONSTRAINT ON (u:User) ASSERT u.username IS UNIQUE;
```

Στην υλοποίησή μας χρησιμοποιήσαμε *ευρετήρια* για τους κόμβους των δέντρων χρόνου, αφού αποτελούν αρκετά συχνά (σχεδόν πάντα) τα σημεία έναρξης των ερωτημάτων μας:

```
//Create Time Tree Indexes
CREATE INDEX ON :Year(value);
CREATE INDEX ON :Month(value);
CREATE INDEX ON :Day(value);
CREATE INDEX ON :Hour(value);
CREATE INDEX ON :Minute(value);
```

Περιορισμούς χρησιμοποιήσαμε για όσες ιδιότητες θέλαμε να διασφαλίσουμε τη μοναδικότητά τους:

```
//Create Constraints
CREATE CONSTRAINT ON (u:User) ASSERT u.username IS UNIQUE;
CREATE CONSTRAINT ON (u:User) ASSERT u.email IS UNIQUE;
CREATE CONSTRAINT ON (a:Activity) ASSERT a.name IS UNIQUE;
CREATE CONSTRAINT ON (b:Borough) ASSERT b.name IS UNIQUE;
CREATE CONSTRAINT ON (i:NO2index) ASSERT i.number IS UNIQUE;
CREATE CONSTRAINT ON (i:O3index) ASSERT i.number IS UNIQUE;
CREATE CONSTRAINT ON (i:SO2index) ASSERT i.number IS UNIQUE;
CREATE CONSTRAINT ON (i:PM25index) ASSERT i.number IS UNIQUE;
CREATE CONSTRAINT ON (i:PM10index) ASSERT i.number IS UNIQUE;
CREATE CONSTRAINT ON (pl:PollutantLevel) ASSERT pl.name IS UNIQUE;
CREATE CONSTRAINT ON (bs:BeaufortScale) ASSERT bs.number IS UNIQUE;
CREATE CONSTRAINT ON (wl:WindLevel) ASSERT wl.name IS UNIQUE;
```

5.3.2 Δέντρα Χρόνου

Η πρώτη δομή που εισάγουμε στη βάση δεδομένων είναι τα δέντρα χρόνου που αποτελούν το θεμέλιο όσων θα ακολουθήσουν στη συνέχεια.

Αρχικά γίνεται η δημιουργία των δέντρων χρόνου για τα έτη 2010 και 2011:

```
//Query 1 - Create Time Tree
WITH range(2010, 2011) AS years, range(1,12) AS months
FOREACH (year IN years |
    CREATE (y:Year {value: year})
    FOREACH (month IN months |
        CREATE (m:Month {value: month})
        MERGE (y)-[:CONTAINS]->(m)
        FOREACH (day IN (
            CASE
                WHEN month IN [1,3,5,7,8,10,12] THEN range(1,31)
```

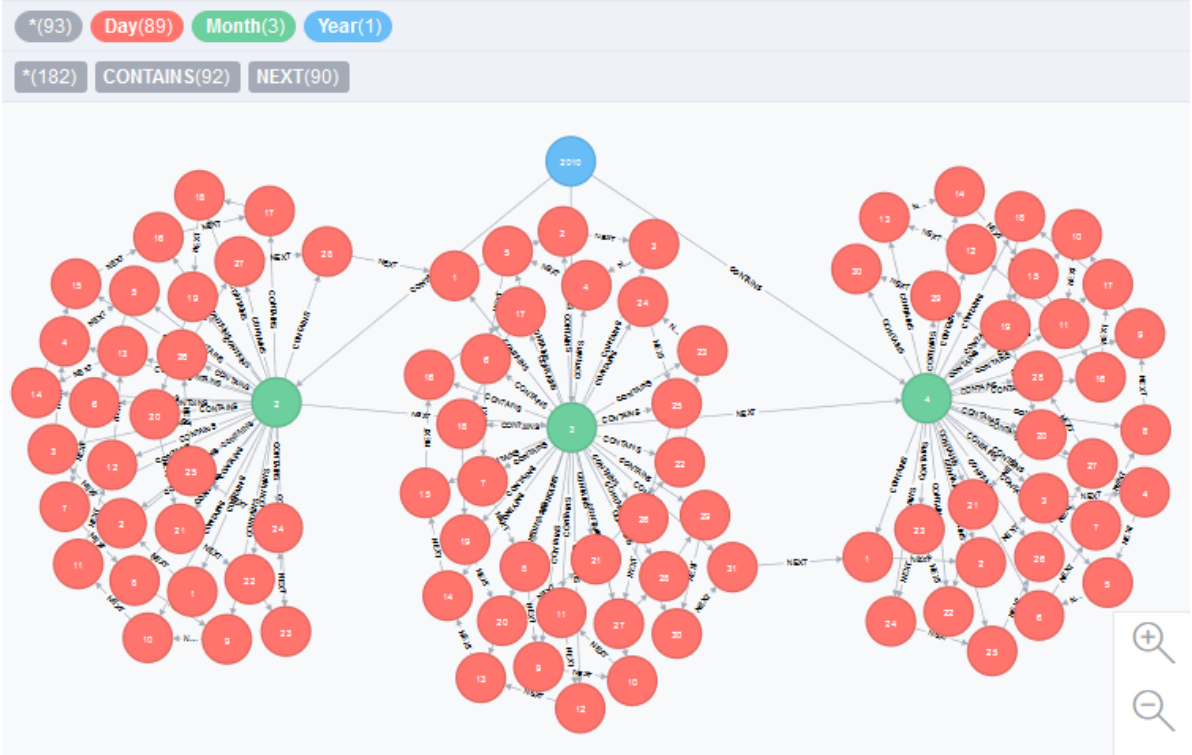
```
                WHEN month = 2 THEN
                    CASE
                        WHEN year % 4 <> 0 THEN range(1,28)
                        WHEN year % 100 <> 0 THEN range(1,29)
                        WHEN year % 400 <> 0 THEN range(1,28)
                        ELSE range(1,29)
                    END
                ELSE range(1,30)
            END) |
            CREATE (d:Day {value: day})
            MERGE (m)-[:CONTAINS]->(d)
            FOREACH (hour IN range(0,23) |
                CREATE (h:Hour {value: hour})
                MERGE (d)-[:CONTAINS]->(h)
                FOREACH (minute IN [0,15,30,45] |
                    CREATE (min:Minute {value: minute})
                    MERGE (h)-[:CONTAINS]->(min))))))
;
```

Ενδιαφέρον στο παραπάνω κομμάτι κώδικα, παρουσιάζει το `FOREACH` όπου γίνεται η δημιουργία των ημερών. Αν και εμείς εισάγουμε μόνο τα έτη 2010 και 2011, για λόγους πληρότητας στο κομμάτι αυτό γίνεται έλεγχος για τα δίσεκτα έτη και αποφασίζεται αν στο Φεβρουάριο θα πρέπει να προστεθούν 28 ή 29 ημέρες.

Στη συνέχεια συνδέουμε τους κόμβους του ίδιου επιπέδου (year-year, month-month κλπ) με σχέσεις `-[:NEXT]->`. Ακολουθεί μόνο το κομμάτι κώδικα που συνδέει τους κόμβους των ετών μεταξύ τους, αλλά με την ίδια λογική συνδέονται και οι κόμβοι των υπόλοιπων επιπέδων:

```
//Query 2 - Connect Years Sequentially
MATCH (year:Year)
WITH year
ORDER BY year.value
WITH collect(year) AS years
FOREACH (i IN range(0, size(years)-2) |
    FOREACH (year1 IN [years[i]] |
        FOREACH (year2 IN [years[i+1]] |
            CREATE UNIQUE (year1)-[:NEXT]->(year2)
            )
        )
    )
;
```

Ένα μέρος του γράφου που δημιουργήθηκε στη βάση δεδομένων, μετά την ολοκλήρωση εισαγωγής των δέντρων χρόνου, φαίνεται στην παρακάτω εικόνα:



Εικόνα 5.2 Αναπαράσταση δέντρων χρόνου στη βάση δεδομένων

Βλέπουμε ένα κόμβο για το έτος 2010 με γαλάζιο χρώμα, τρείς από τους μήνες που αυτό περιέχει με κόμβους πράσινου χρώματος καθώς και τις ημέρες των μηνών με κόκκινους κόμβους. Μπορούμε επίσης να διακρίνουμε τις σχέσεις `:CONTAINS` που συνδέουν κόμβους διαφορετικών επιπέδων και τις σχέσεις `:NEXT` που συνδέουν τους μήνες με τους μήνες και τις ημέρες με τις ημέρες.

5.3.3 Δεδομένα ατμοσφαιρικής ρύπανσης

Από το σημείο αυτό ξεκινάει η διαδικασία εισαγωγής των διαθέσιμων δεδομένων με χρήση του `LOAD CSV`. Όπως αναφέρθηκε στην αρχή του κεφαλαίου, η υλοποίηση της εργασίας πραγματοποιήθηκε σε διανομή linux και το “path” φόρτωσης των αρχείων csv είναι διαφορετικό από ότι σε windows (για αναλυτικότερες οδηγίες ανατρέξτε στις παραγράφους 5.1.3, 5.1.4 και στο αρχείο 'windows_cypher.csv’ της εργασίας).

Για κάθε ένα ρύπο προηγείται η δημιουργία της ιεραρχίας ‘ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης - επιπέδου συγκέντρωσης’ όπως περιγράφηκε στην παράγραφο 4.2.5:

```
//Query 7 -Import no2 index data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/airpollution/no2_index.csv' AS line
WITH line, split(line.`ug/m3`, "-") AS limits
WITH line, toInteger(limits[0]) AS minLimit, toInteger(limits[1]) AS
maxLimit
```

```
//Create index nodes
CREATE (no2i:NO2index)
SET no2i.number = toInteger(line.Index),
    no2i.minConcentration = minLimit,
    no2i.maxConcentration = maxLimit,
    no2i.unit = "ug/m3"
//Merge level node and add LEVEL relationship
MERGE (pl:PollutantLevel {name: line.Band})
CREATE (no2i)-[:LEVEL]->(pl)
;
```

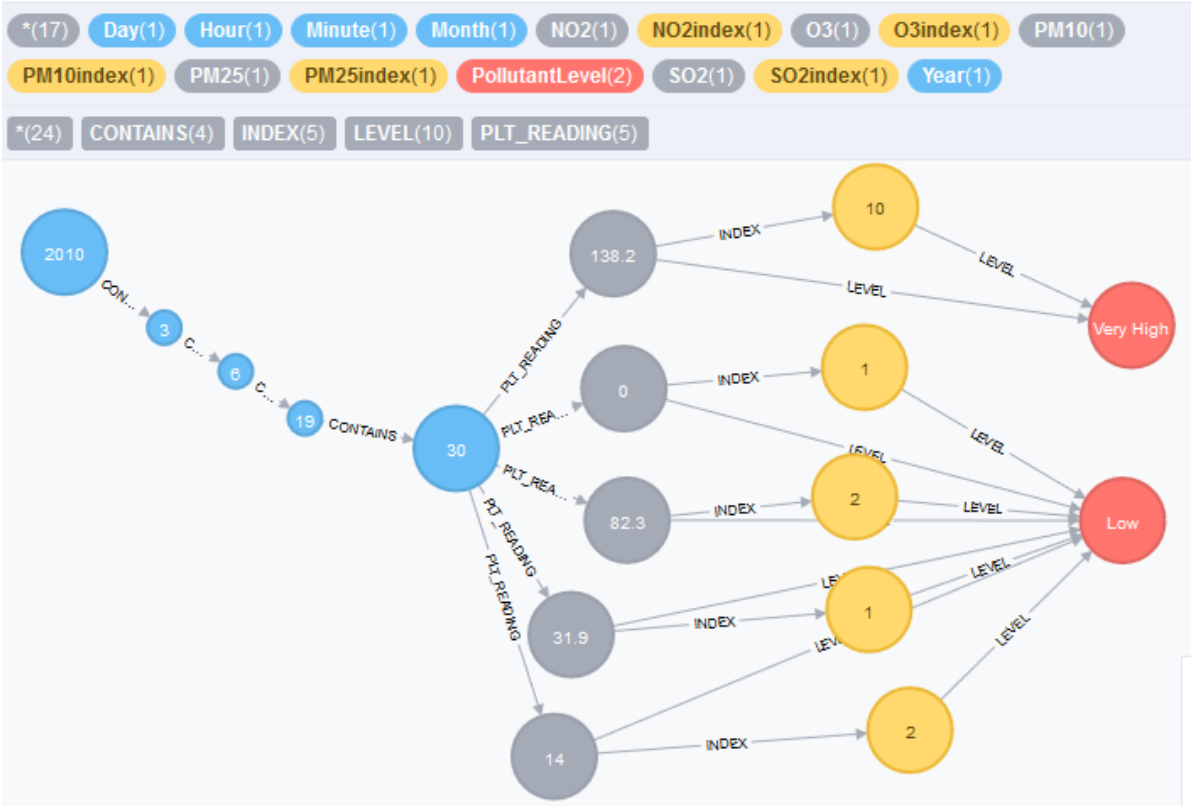
και έπεται η εισαγωγή των συγκεντρώσεων:

```
//Query 8 - Import NO2 concentration data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/airpollution/no2_concentration.csv' AS line
WITH line, split(line.Site, '-') AS boroughSite,
split(line.ReadingDateTime, ' ') AS dateTime
WITH line, boroughSite[0] AS borough, boroughSite[1] AS site,
split(dateTime[0], '/') AS date, split(dateTime[1], ':') AS time
WITH line, borough, site, toInteger(date[0]) AS day, toInteger(date[1]) AS
month, toInteger(date[2]) AS year, toInteger(time[0]) AS hour,
toInteger(time[1]) AS minute
MATCH (y:Year {value: year})-[:CONTAINS]->(m:Month {value: month})-
[:CONTAINS]->(d:Day {value: day})
MERGE (d)-[:CONTAINS]->(h:Hour {value: hour})
MERGE (h)-[:CONTAINS]->(min:Minute {value: minute})
//Connect concentration reading to Already Existing Minute
CREATE (min)-[:PLT_READING]->(no2:NO2)
SET no2.concentration = toFloat(line.Value),
    no2.unit = "ug/m3"
//Create other NO2 relationships
MERGE (b:Borough {name: borough})
CREATE (no2)-[:IN_BOROUGH]->(b)
MERGE (s:Site {name: site})
CREATE (no2)-[:AT_SITE]->(s)
//Connect site and borough if they aren't already connected
MERGE (s)-[:IN_BOROUGH]->(b)
WITH no2, round(toFloat(line.Value)) AS concentration
//Find and match index node
MATCH (no2i:NO2index)-[:LEVEL]->(pl:PollutantLevel)
WHERE no2i.minConcentration <= concentration <= no2i.maxConcentration
```

```
//Connect no2 to index
CREATE (no2)-[:INDEX]->(no2i)
CREATE (no2)-[:LEVEL]->(pl)
;
```

Όμοια γίνεται και η εισαγωγή των συγκεντρώσεων για τους τέσσερις εναπομείναντες ρύπους (O3, SO2, PM10, PM25).

Ενδεικτικό κομμάτι του γράφου που δημιουργείται φαίνεται στην εικόνα που ακολουθεί:



Εικόνα 5.3 Αναπαράσταση γράφου για τις μετρήσεις συγκεντρώσεων στη βάση δεδομένων

Στην παραπάνω εικόνα βλέπουμε όλους τους κόμβους του δέντρου χρόνου με γαλάζιο χρώμα να οδηγούν στις 06/03/2010 και ώρα 19:30. Από εκεί βλέπουμε τις συγκεντρώσεις των πέντε διαφορετικών ρύπων με γρι χρώμα να δείχνουν προς τον ημερήσιο δείκτη ατμοσφαιρικής ρύπανσης (κίτρινο χρώμα) και προς το επίπεδο ατμοσφαιρικής ρύπανσης (κόκκινο χρώμα).

5.3.4 Δεδομένα καιρικών συνθηκών

Τα δεδομένα των καιρικών συνθηκών καταχωρούνται στη βάση δεδομένων με τρόπο ανάλογο της εισαγωγής των δεδομένων ατμοσφαιρικής ρύπανσης. Παραθέτουμε ενδεικτικά την εισαγωγή των δεδομένων θερμοκρασίας:

```
//Query 17 - Import temperature data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/weather/temperature.csv' AS line
WITH line, split(line.Site, '-') AS boroughSite,
```

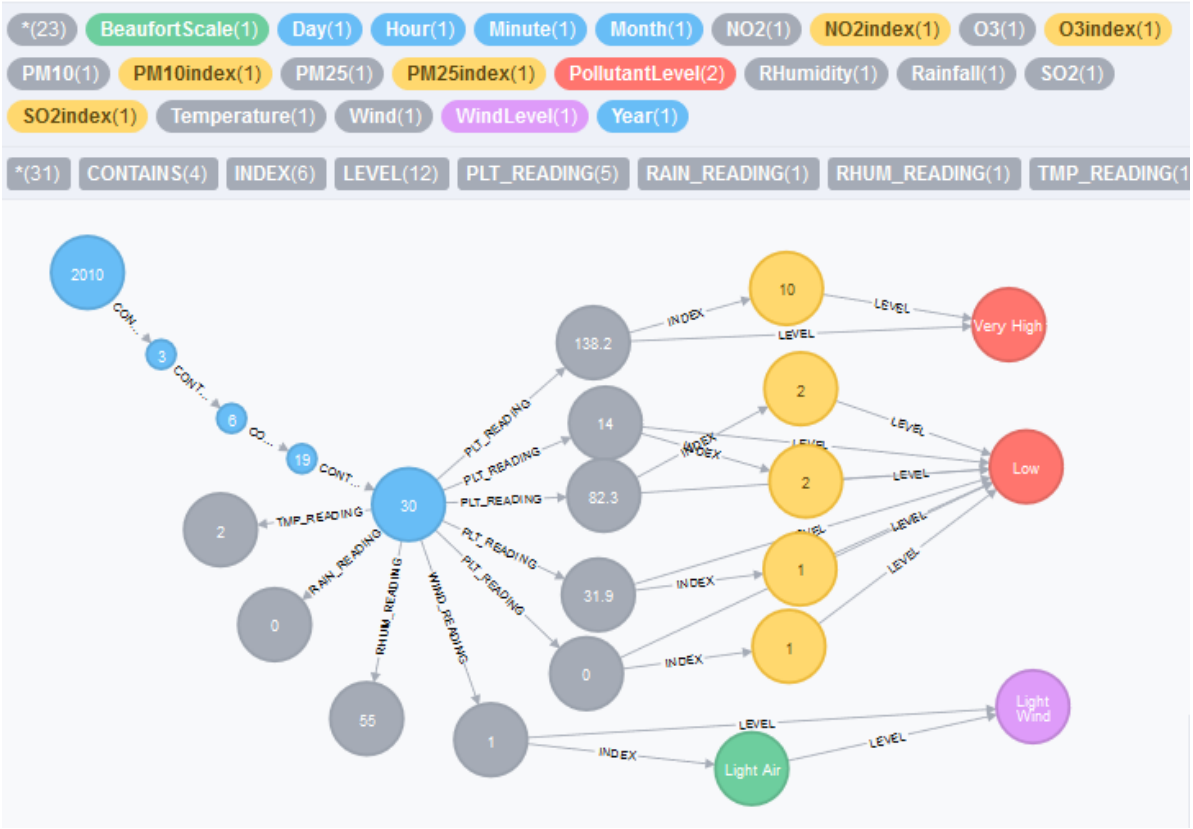
```
    split(line.ReadingDateTime, ' ') AS dateTime
WITH line, boroughSite[0] AS borough, boroughSite[1] AS site,
    split(dateTime[0], '/') AS date, split(dateTime[1], ':') AS time
WITH line, borough, site, toInteger(date[0]) AS day, toInteger(date[1]) AS
month,
    toInteger(date[2]) AS year, toInteger(time[0]) AS hour,
toInteger(time[1]) AS minute
MATCH (y:Year {value: year})-[:CONTAINS]->(m:Month {value: month})-
[:CONTAINS]->(d:Day {value: day})
//Hours are handled as 24-Hour time from 0 to 23
//Connect Hour to Already Existing Day
MERGE (d)-[:CONTAINS]->(h:Hour {value: hour})
//Minutes are Zero Based with a Range from 0 to 59
//Connect Minute to Already Existing Hour
MERGE (h)-[:CONTAINS]->(min:Minute {value: minute})
//Connect temperature reading to Already Existing Minute
CREATE (min)-[:TMP_READING]->(tmp:Temperature)
SET tmp.value = toInteger(line.Value),
    tmp.unit = line.Units
//Create other temperatue relationships
MERGE (b:Borough {name: borough})
CREATE (tmp)-[:IN_BOROUGH]->(b)
MERGE (s:Site {name: site})
CREATE (tmp)-[:AT_SITE]->(s)
//Connect site and borough if they aren't already connected
MERGE (s)-[:IN_BOROUGH]->(b)
;
```

Να υπενθυμίσουμε ότι μόνο για τα δεδομένα ανέμων δημιουργήθηκε ένα επιπλέον επίπεδο ολοκλήρωσης με την εισαγωγή των κόμβων για την κλίμακα μποφόρ και τα επίπεδα ανέμων (όπως περιγράφεται στην παράγραφο 4.2.7). Η διαδικασία είναι παρόμοια με την εισαγωγή του ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης και παρουσιάζεται ακολούθως:

```
//Query 20 - Import beaufort scale data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/weather/beaufort_scale.csv' AS line
//Create beaufort scale nodes
CREATE (bs:BeaufortScale)
SET bs.number = toInteger(line.`Beaufort number`),
    bs.description = line.Description,
```

```
    bs.minSpeed = toFloat(line.`Wind speed_min (m/s)`),
    bs.maxSpeed = toFloat(line.`Wind speed_max (m/s)`),
    bs.speedUnit = "m/s"
//Create, if not exist, Wind level node
MERGE (wl:WindLevel {name: line.Level})
//Connenct bs and wl nodes
CREATE (bs)-[:LEVEL]->(wl)
;
```

Μετά από την ολοκλήρωση της εισαγωγής και των δεδομένων των καιρικών συνθηκών έχουμε την παρακάτω εικόνα του γράφου:



Εικόνα 5.4 Αναπαράσταση γράφου για δεδομένα ρύπων και καιρικών συνθηκών

5.3.5 Χρήστες (Users)

Για κάθε ένα χρήστη που εγγράφεται στην εφαρμογή, δημιουργείται ένας κόμβος `:User` και τα χαρακτηριστικά του χρήστη αποθηκεύονται ως ιδιότητες του κόμβου αυτού. Για τη δημιουργία των χρηστών χρησιμοποιείται το `MERGE` αντί του `CREATE`. Αυτή η πρακτική μας εξασφαλίζει τη μοναδικότητα του κάθε χρήστη στη βάση δεδομένων, καθώς το `MERGE` ελέγχει πρώτα την ύπαρξη άλλου χρήστη με το παρεχόμενο username και εφόσον τέτοιος χρήστης δεν υπάρχει, τον δημιουργεί και θέτει τις τιμές των ιδιοτήτων του κόμβου ίσες με τις τιμές που παρέχονται από το χρήστη:

```
//Query 22 - Import users data
USING PERIODIC COMMIT 500
```

```
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/users/users.csv' AS line
MERGE (u:User {username: line.username})
ON CREATE SET
    u.name = line.full_name,
    u.email = line.email,
    u.gender = line.gender,
    u.age = toInteger(line.age),
    u.password = line.password
;
```

Σημειώνουμε εδώ ότι, η τιμή της ιδιότητας 'password' ισούται με την τιμή που παρέχεται από το χρήστη. Για λόγους ασφαλείας όμως, σε πραγματική υλοποίηση της εφαρμογής, το πεδίο αυτό θα πρέπει να περνάει πρώτα από διαδικασία κρυπτογράφησης.

5.3.6 Προτιμήσεις καιρικών συνθηκών και επιπέδου ρύπων

Μετά την καταχώρηση των χρηστών ακολουθεί η εισαγωγή των προτιμήσεων τους, τόσο σε περιβαλλοντικές συνθήκες όσο και σε δραστηριότητες. Για τις προτιμήσεις περιβαλλοντικών συνθηκών δημιουργείται ένας κόμβος `:Conditions` για τον κάθε χρήστη και μέσα σε αυτόν τον κόμβο αποθηκεύονται οι μέγιστες και οι ελάχιστες τιμές που δηλώνει ο χρήστης. Κατά τη δημιουργία του, ο κόμβος συνδέεται επίσης μέσω μιας σχέσης `[:PREFER]`, με τον αντίστοιχο χρήστη:

```
//Query 23 - Import users_conditionsPreferences.csv
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/users/users_conditionsPreferences.csv' AS line
MATCH (u:User {username: line.username})
CREATE (u)-[:PREFER]->(c:Conditions)
SET c.minTemperature = toInteger(line.minTemp),
    c.maxTemperature = toInteger(line.maxTemp),
    c.minRainfall = toInteger(line.minRain),
    c.maxRainfall = toInteger(line.maxRain),
    c.minRhumidity = toInteger(line.minRHum),
    c.maxRhumidity = toInteger(line.maxRHum),
    c.minWind = toInteger(line.minWind),
    c.maxWind = toInteger(line.maxWind),
    c.minPollution = toInteger(line.minPol),
    c.maxPollution = toInteger(line.maxPol)
;
```

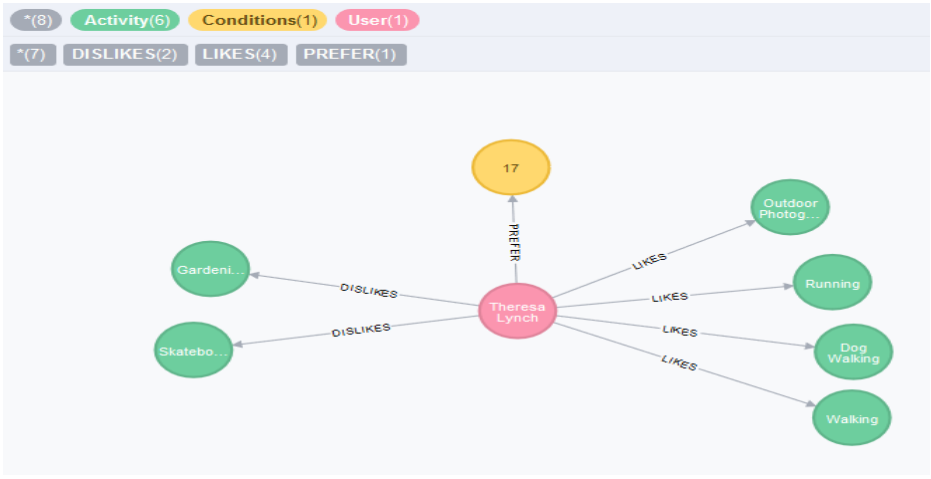
5.3.7 *Προτιμήσεις και δημιουργία δραστηριοτήτων*

Ακολουθεί η αποθήκευση των προτιμήσεων των χρηστών σε δραστηριότητες. Στην υλοποίηση μας δεν έγινε ανεξάρτητη εισαγωγή των κόμβων των δραστηριοτήτων στη βάση δεδομένων. Αντί αυτού, χρησιμοποιήθηκε η εντολή `MERGE` κατά την αποθήκευση των προτιμήσεων των χρηστών όπως φαίνεται παρακάτω:

```
//Query 24 - Import users_likes data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/users/users_likes.csv' AS line
MATCH (u:User {username: line.username})
MERGE (a:Activity {name: line.activity})
MERGE (u)-[l:LIKES]->(a)
SET l.value = toInteger(line.score)
;
//Query 25 - Import users_dislikes data
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/users/users_dislikes.csv' AS line
MATCH (u:User {username: line.username})
MERGE (a:Activity {name: line.activity})
MERGE (u)-[dl:DISLIKES]->(a)
SET dl.value = 0
;
```

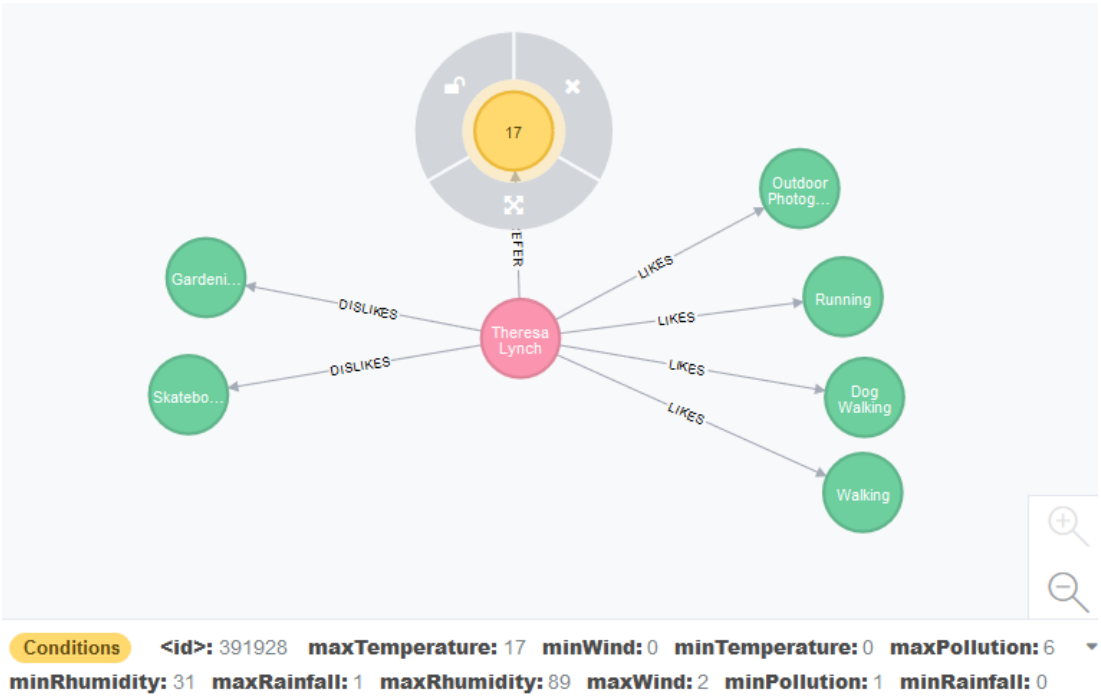
Από το παραπάνω κομμάτι κώδικα, παρατηρούμε πως όταν δημιουργείται μια σχέση ‘likes’, αποθηκεύεται σε αυτή και μια ιδιότητα ‘value’ που δείχνει το μέγεθος της προτίμησης του χρήστη στη συγκεκριμένη δραστηριότητα. Αντίθετα, κατά τη δημιουργία των ‘dislikes’ η ιδιότητα ‘value’ τίθεται αυτομάτως στο 0.

Το ενδεικτικό προφίλ κάποιου χρήστη στο γράφο μας, μετά από την εισαγωγή των χρηστών και των προτιμήσεων τους, έχει ως εξής:



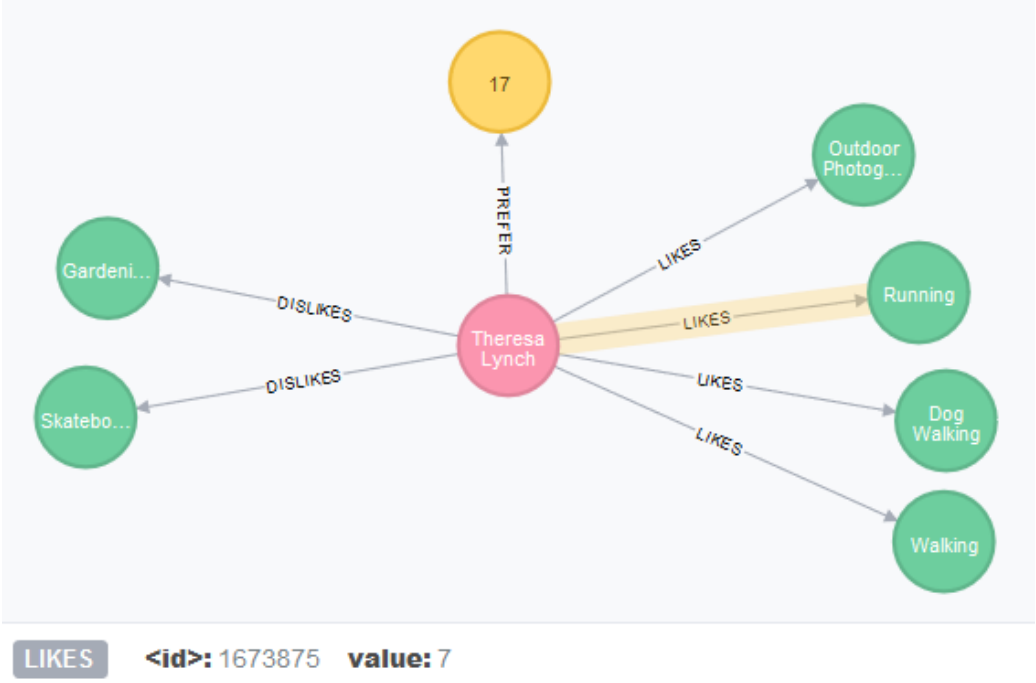
Εικόνα 5.5 Αναπαράσταση χρήστη και προτιμήσεων στη βάση δεδομένων

Εάν στον παραπάνω γράφο επιλέξουμε τον κόμβο με τις προτιμήσεις συνθηκών, θα μπορέσουμε να δούμε αναλυτικά τις προτιμήσεις του χρήστη:



Εικόνα 5.6 Τιμές προτιμήσεων καιρικών συνθηκών και επιπέδου ρύπων στη βάση δεδομένων

Μπορούμε επίσης, πατώντας στις σχέσεις 'likes', να δούμε το βάρος με το οποίο έχει δηλώσει ο χρήστης ότι του αρέσει μια δραστηριότητα:



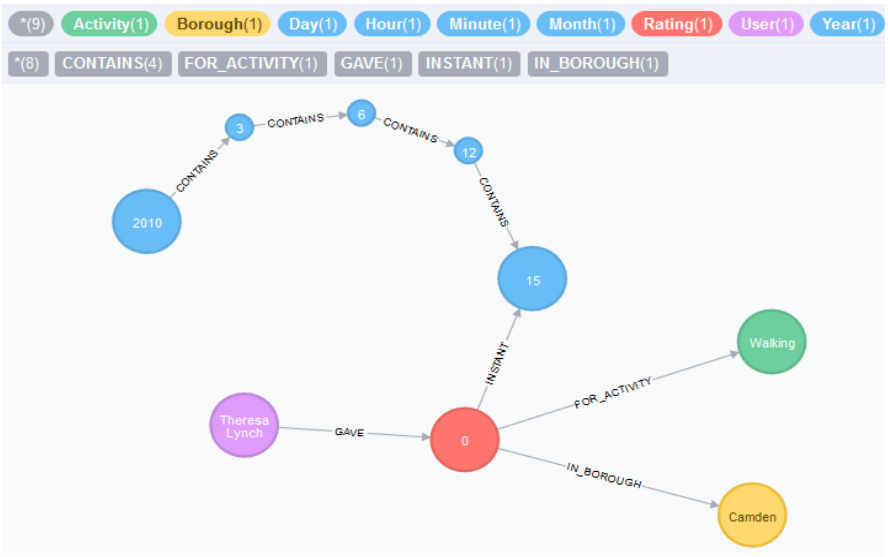
Εικόνα 5.7 Τιμή της ιδιότητας "value" σε σχέσεις "LIKES" στη βάση δεδομένων

5.3.8 *Αξιολόγηση δραστηριοτήτων*

Η υλοποίηση της βάσης ολοκληρώνεται με την προσθήκη αξιολογήσεων πραγματικού χρόνου από τους χρήστες της εφαρμογής για δραστηριότητες που έχουν πραγματοποιήσει. Κάθε κόμβος αξιολόγησης συνδέεται τόσο με τον αντίστοιχο χρήστη και δραστηριότητα όσο και με τα δέντρα χρόνου και την περιοχή στην οποία καταχωρήθηκε:

```
//Query 26 - Import rating_activities
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM
'file:///thesis_vcharlaftis/users/rating_activities.csv' AS line
WITH line, split(line.dateTime, ' ') AS dateTime
WITH line, split(dateTime[0], '/') AS date, split(dateTime[1], ':') AS
time
WITH line, toInteger(date[0]) AS day, toInteger(date[1]) AS month,
    toInteger(date[2]) AS year, toInteger(time[0]) AS hour,
toInteger(time[1]) AS minute
MATCH (:Year {value: year})-[:CONTAINS]->(:Month {value: month})-
[:CONTAINS]->
    (:Day {value: day})-[:CONTAINS]->(:Hour {value: hour})-[:CONTAINS]-
>(min:Minute)
WHERE minute - 15 < min.value <= minute
MATCH (u:User {username: line.username})
MERGE (a:Activity {name: line.activity})
MERGE (b:Borough {name: "Camden"})
CREATE (u)-[:GAVE]->(r:Rating {value: toInteger(line.rating)})-[:INSTANT]-
>(min),
    (r)-[:FOR_ACTIVITY]->(a), (r)-[:IN_BOROUGH]->(b)
;
```

Μια εικόνα του γράφου μας παρουσιάζεται παρακάτω:



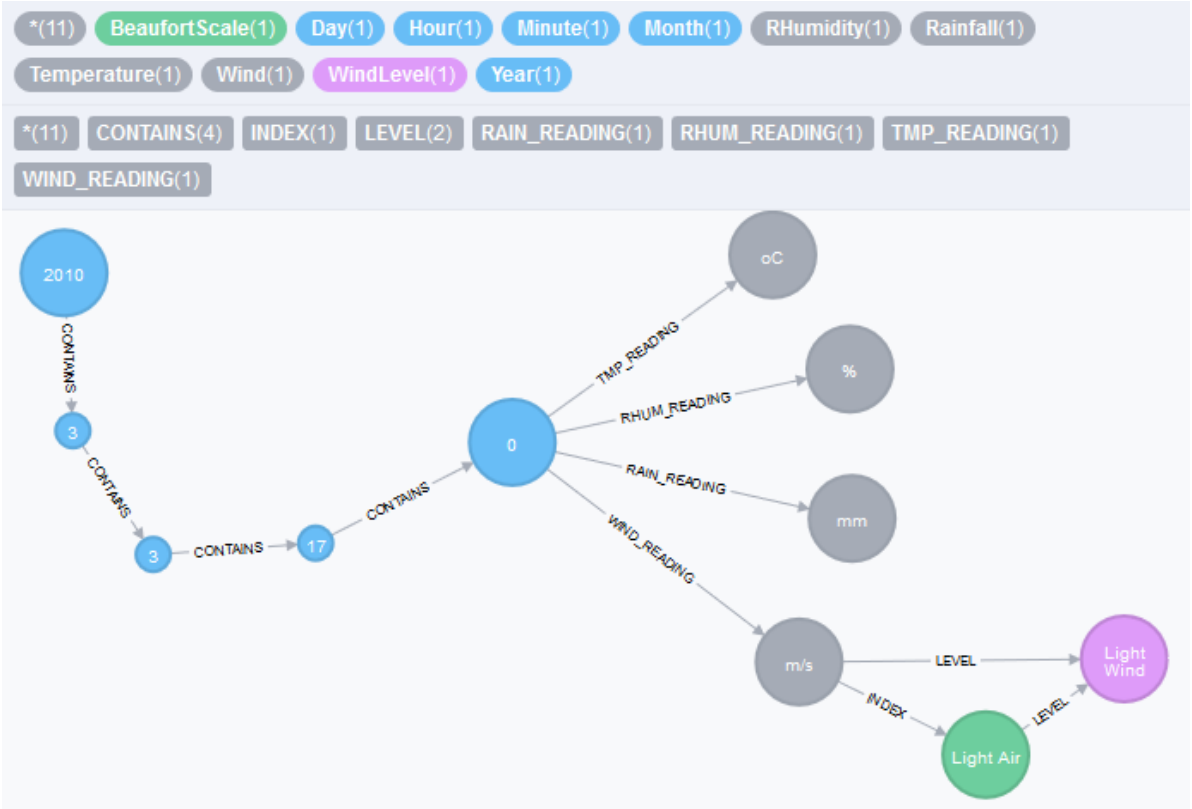
Εικόνα 5.8 Αναπαράσταση αξιολογήσεων (rating) στη βάση δεδομένων

6 Έλεγχος Υλοποίησης

Στο κεφάλαιο αυτό απαντάμε σε κάποια τετριμμένα ερωτήματα με σκοπό να ελέγξουμε την ορθότητα της υλοποίησης μας. Για να μπορέσουμε να προχωρήσουμε στην υλοποίηση συστάσεων, πρέπει πρώτα να ελέγξουμε την ορθότητα των μοντέλων μας, απαντώντας στα ερωτήματα που μας ώθησαν να τα κατασκευάσουμε. Τυχαία επιλέξαμε να πραγματοποιήσουμε τους ελέγχους μας στις 03/03/2010 και ώρα 17:00, αλλά σε πραγματικό χρόνο τόσο η ημερομηνία όσο και η ώρα θα περνιούνται ως παράμετροι στα ερωτήματα που θα στέλνονται προς τη βάση δεδομένων.

6.1 Εύρεση θερμοκρασίας

Μια ερώτηση που θα πρέπει να απαντάει η βάση δεδομένων που δημιουργήσαμε είναι η ερώτηση: «Τι θερμοκρασία έχει τώρα (03/03/2010 17:00);». Το κομμάτι του γράφου για τη συγκεκριμένη ημέρα και ώρα παρουσιάζεται στην ακόλουθη εικόνα:



Εικόνα 6.1 Αναπαράσταση μετρήσεων καιρικών συνθηκών στη βάση δεδομένων

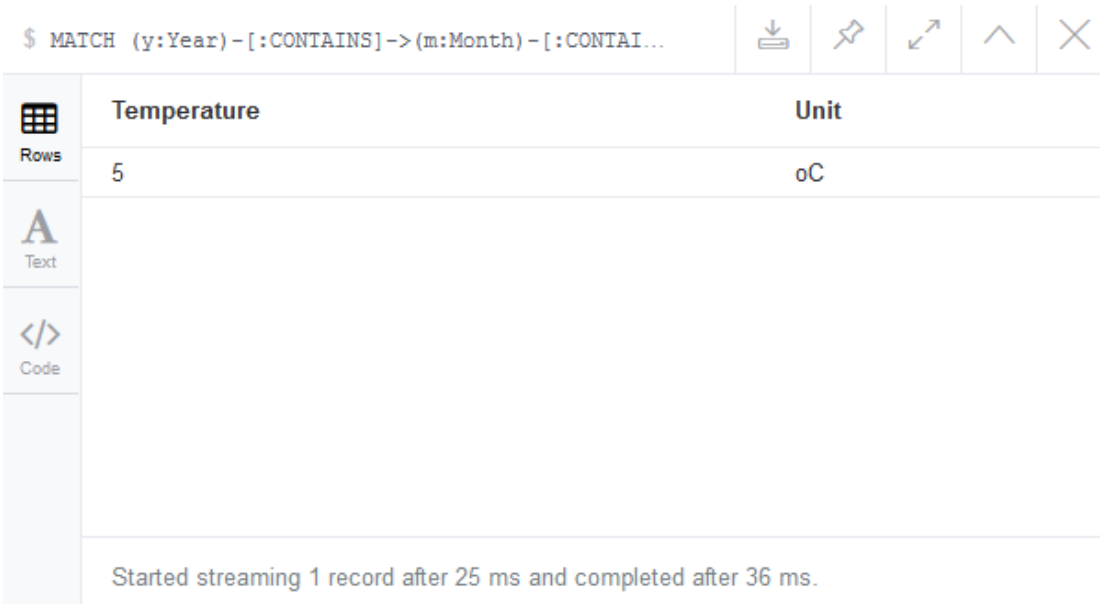
Για να απαντήσουμε στην ερώτηση αυτή, θα πρέπει να διασχίσουμε το μονοπάτι των δέντρων χρόνου που οδηγεί από τον κόμβο του έτους 2010 στο κόμβο του λεπτού που μας αφορά, και από

εκεί να διασχίσουμε τη σχέση [:TMP_READING] που οδηγεί στην αποθηκευμένη τιμή θερμοκρασίας για το συγκεκριμένο λεπτό.

Τα παραπάνω επιτυγχάνονται με το ακόλουθο ερώτημα cypher προς τη βάση δεδομένων:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
OPTIONAL MATCH (min)-[:TMP_READING]->(t:Temperature)
RETURN t.value AS Temperature, t.unit AS Unit
;
```

που επιστρέφει το εξής αποτέλεσμα:

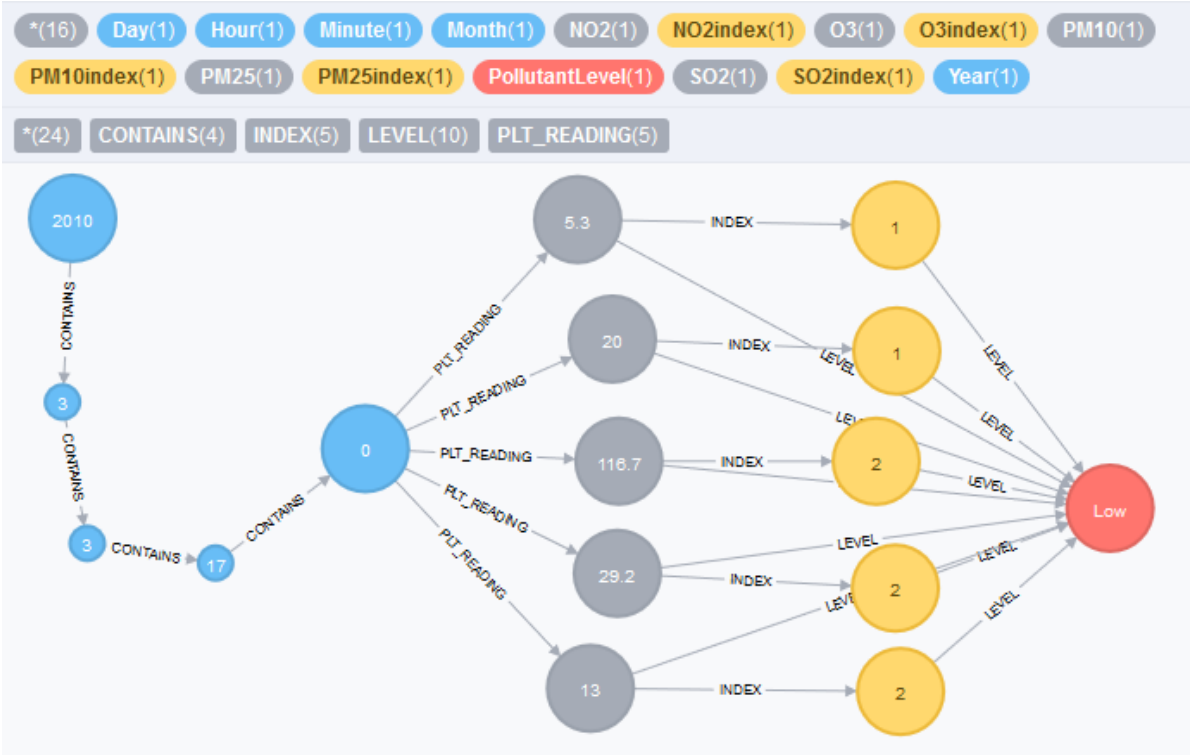


Να σημειώσουμε ότι για τη διάσχιση της σχέσης [:TMP_READING] έχουμε χρησιμοποιήσει την έκφραση της cypher OPTIONAL MATCH. Η έκφραση αυτή αναζητάει το μονοπάτι που της έχουμε ορίσει. Εάν το βρεί επιστρέφει τα δεοδομένα που ζητήσαμε, ενώ σε διαφορετική περίπτωση επιστρέφει null. Ο λόγος που χρησιμοποιήσαμε το OPTIONAL MATCH αντί του MATCH έγγειται στο γεγονός ότι υπάρχει το ενδεχόμενο να μην έχει γίνει καταγραφή κάποιας μέτρησης από τους αισθητήρες της πόλης για κάποιο λεπτό.

Με ανάλογη διαδικασία, μπορούμε να απαντήσουμε σε ερωτήσεις για τις τιμές μετρήσεων των υπόλοιπων συνθηκών. Ένα λίγο μεγαλύτερο ενδιαφέρον παρουσιάζει η εύρεση του επιπέδου ατμοσφαιρικής ρύπανσης που περιγράφεται στην ακόλουθη παράγραφο.

6.2 *Εύρεση επιπέδου ατμοσφαιρικής ρύπανσης*

Η βάση δεδομένων θα πρέπει επίσης να μπορεί να απαντάει στην ερώτηση: «Ποιο είναι τώρα το επίπεδο ατμοσφαιρικής ρύπανσης (03/03/2010 17:00);». Όπως βλέπουμε στην ακόλουθη εικόνα:



Εικόνα 6.2 Αναπαράσταση συγκεντρώσεων ρύπων και επιπέδων ρύπανσης στη βάση δεδομένων

για να απαντήσουμε στην ερώτηση αυτή, δεν αρκεί η διαδικασία που ακολουθήθηκε στην ενότητα 6.1 για την εύρεση της θερμοκρασίας. Σε αυτή την περίπτωση πρέπει, όχι μόνο να διασχίσουμε τις σχέσεις `[:PLT_READING]` για να βρούμε την τιμή συγκέντρωσης του κάθε ρύπου, αλλά και να επεξεργαστούμε τις τιμές αυτές. Συγκεκριμένα, πρέπει να μάθουμε σε τι τιμή ημερήσιου δείκτη ατμοσφαιρικής ρύπανσης αντιστοιχεί η κάθε συγκέντρωση και να ταξινομήσουμε τις προκύπτουσες τιμές σε φθίνουσα σειρά. Στη συνέχεια πρέπει να διαλέξουμε τη μεγαλύτερη από τις παραπάνω τιμές και να επιστρέψουμε το επίπεδο ατμοσφαιρικής ρύπανσης στο οποίο αντιστοιχεί.

Το ερώτημα cypher που απευθύνουμε προς τη βάση δεδομένων παρουσιάζεται παρακάτω:

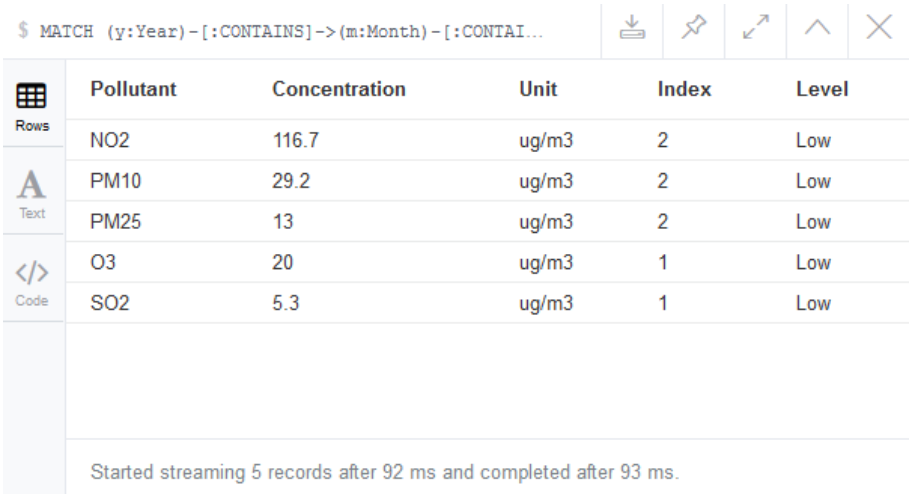
```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
OPTIONAL MATCH (min)-[:PLT_READING]->()-[:INDEX]->(i)-[:LEVEL]->
    (pl:PollutantLevel)
RETURN pl.name AS Level, i.number AS Index
ORDER BY Index DESC LIMIT 1;
```

και επιστρέφει της τιμές “Low level” και “index 2” όπως φαίνεται στην εικόνα:

Φυσικά ο χρήστης της εφαρμογής έχει τη δυνατότητα να ζητήσει και αναλυτικές πληροφορίες για όλες τις τιμές συγκέντρωσης. Για να απαντήσουμε σε μια τέτοια απαίτηση, μπορούμε να χρησιμοποιούμε το ακόλουθο ερώτημα:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
OPTIONAL MATCH (min)-[:PLT_READING]->(pol)-[:INDEX]->(i)-[:LEVEL]-
>(pl:PollutantLevel)
RETURN labels(pol)[0] AS Pollutant, pol.concentration AS Concentration,
pol.unit AS Unit,
    i.number AS Index, pl.name AS Level
ORDER BY Index DESC, Concentration DESC
;
```

Το οποίο επιστρέφει τα παρακάτω αποτελέσματα:

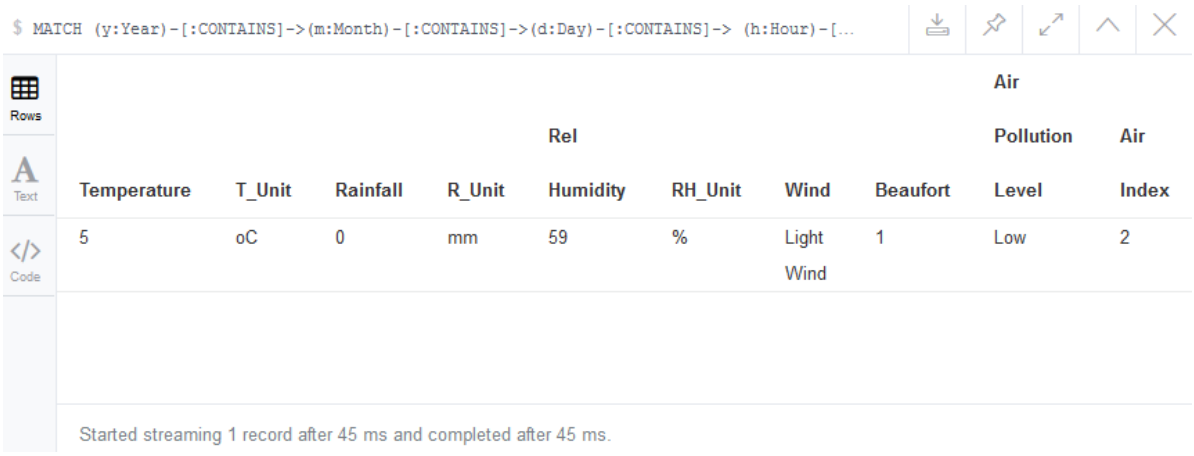


6.3 *Εύρεση καιρικών συνθηκών και ατμοσφαιρικής ρύπανσης*

Εδώ θα δείξουμε ένα ερώτημα που μπορεί να επιστρέψει συνολικά όλες τις μετρήσεις που έχουν καταγραφεί για κάποιο λεπτό:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH y, m, d, h, min
OPTIONAL MATCH (min)-[:TMP_READING]->(t:Temperature)
OPTIONAL MATCH (min)-[:RAIN_READING]->(r:Rainfall)
OPTIONAL MATCH (min)-[:RHUM_READING]->(rh:RHumidity)
OPTIONAL MATCH (min)-[:WIND_READING]->(w:Wind)-[:INDEX]->
    (bs:BeaufortScale)-[:LEVEL]->(wl:WindLevel)
OPTIONAL MATCH (min)-[:PLT_READING]->(pol)-[:INDEX]->(i)-[:LEVEL]->
    (pl:PollutantLevel)
RETURN t.value AS Temperature, t.unit AS T_Unit, r.value AS Rainfall,
r.unit AS R_Unit,
    rh.value AS `Rel Humidity`, rh.unit AS RH_Unit, wl.name AS Wind,
bs.number AS Beaufort,
    pl.name AS `Air Pollution Level`, i.number AS `Air Index`
ORDER BY `Air Index` DESC LIMIT 1
;
```

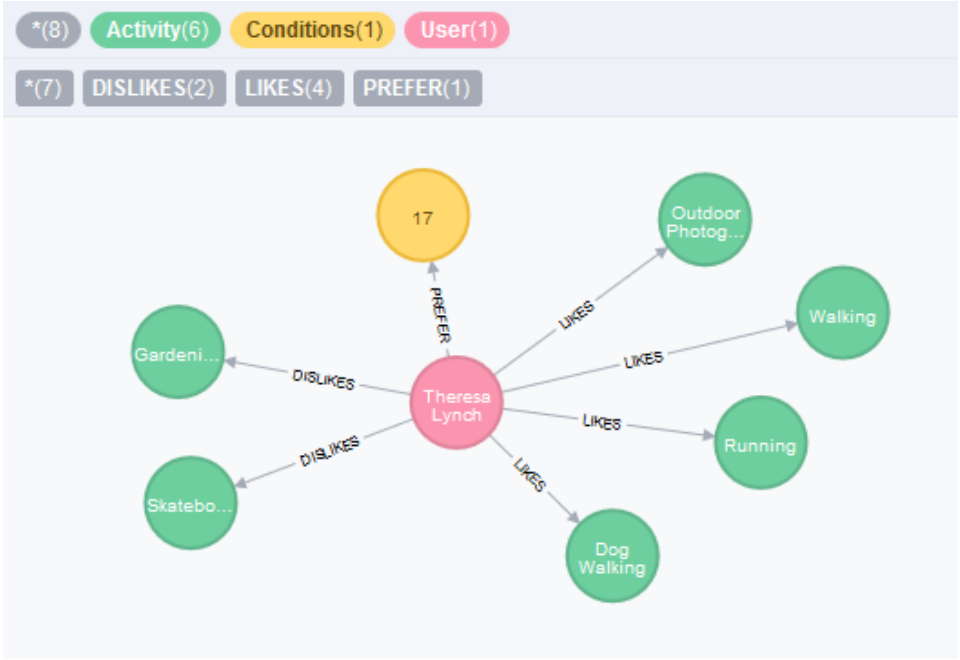
Το παραπάνω ερώτημα επιστρέφει τον ακόλουθο πίνακα τιμών:



6.4 *Εύρεση προτιμήσεων χρήστη*

Εκτός από τις μετρήσεις καιρικών συνθηκών και ατμοσφαιρικής ρύπανσης, θα πρέπει να μπορούμε να βρούμε μέσα από το γράφο μας και τις προτιμήσεις ενός χρήστη. Με δεδομένο το όνομα κάποιου χρήστη θα πρέπει να μπορούμε να εντοπίσουμε τις προτιμήσεις του τόσο σε περιβαλλοντικές συνθήκες όσο και σε δραστηριότητες.

Ένα παράδειγμα χρήστη φαίνεται στην εικόνα που ακολουθεί:

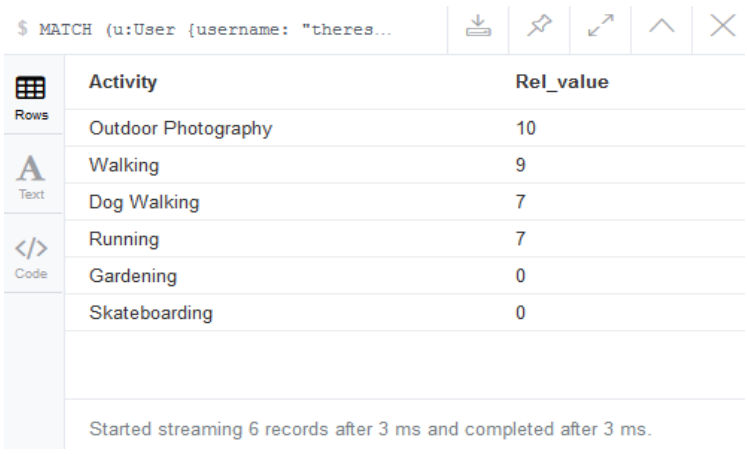


Εικόνα 6.3 Αναπαράσταση χρήστη και προτιμήσεων στη βάση δεδομένων

και οι προτιμήσεις του σε δραστηριότητες δίνονται με το ακόλουθο ερώτημα:

```
MATCH (u:User {username: "theresalynch34"})-[r:LIKES|DISLIKES]->
    (a:Activity)
RETURN a.name AS Activity, r.value AS Rel_value
ORDER BY Rel_value DESC
;
```

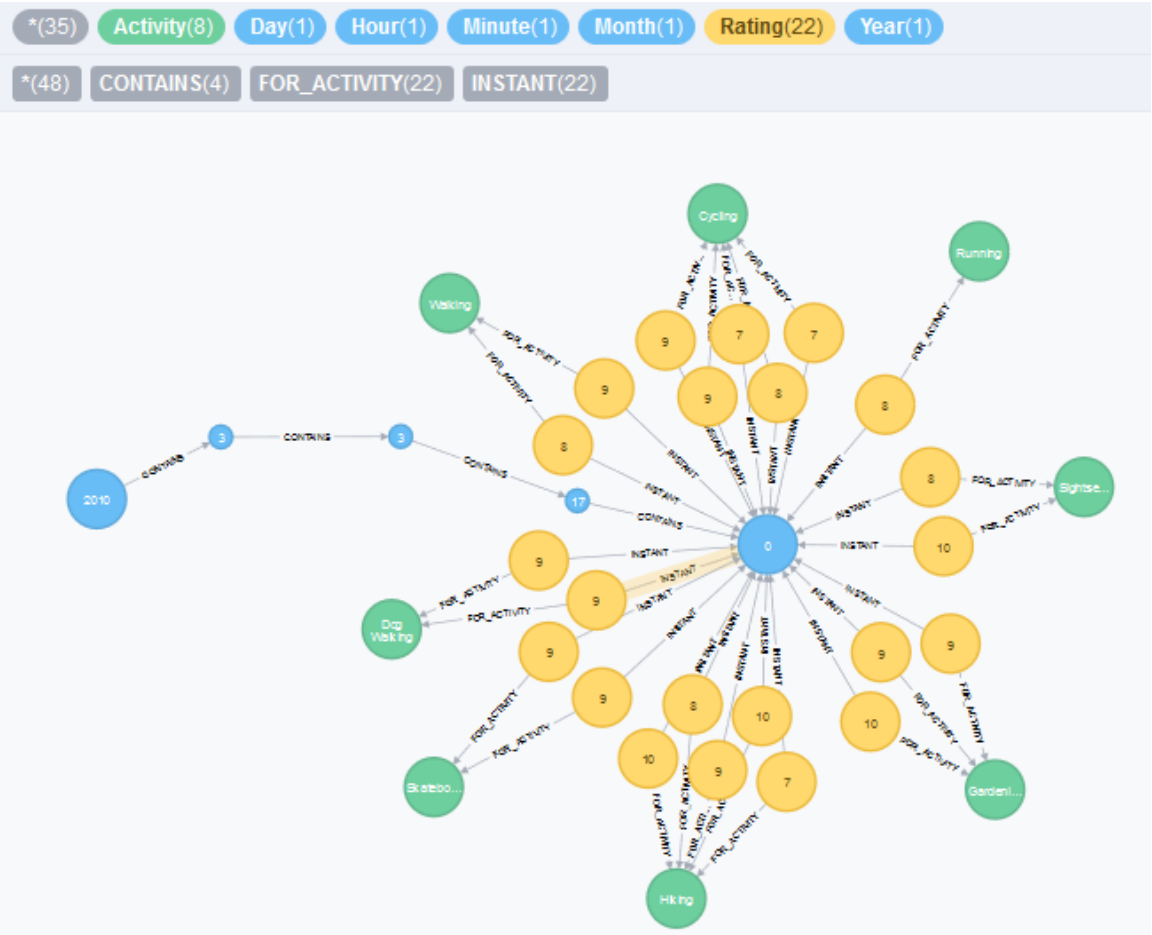
το οποίο επιστρέφει τα εξής αποτελέσματα:



6.5 *Βαθμολογίες δραστηριοτήτων*

Το τελευταίο ερώτημα που θα παρουσιάσουμε στο κεφάλαιο αυτό είναι η εύρεση της τιμής των αξιολογήσεων με τις οποίες έχουν βαθμολογήσει τις δραστηριότητες οι χρήστες της εφαρμογής για μια δεδομένη χρονική στιγμή (03/03/2010 17:00).

Το κομμάτι του γράφου που θα μελετήσουμε παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 6.4 Αξιολογήσεις δραστηριοτήτων στη βάση δεδομένων

και το ερώτημα cypher που θα απευθύνουμε προς τη βάση δεδομένων έχει ως εξής:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH min
MATCH (min)<-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a:Activity),
      (u:User)-[:GAVE]->(r)
RETURN u.name AS User, r.value AS Rating, a.name AS Activity
;
```

Ενδεικτικά, τα πρώτα από τα αποτελέσματα της εκτέλεσης του ερωτήματος φαίνονται στην εικόνα που ακολουθεί:

User	Rating	Activity
Willie Sanchez	10	Gardening
Victor Montgomery	7	Hiking
Jimmy Woods	8	Walking
Ryan Scott	10	Sightseeing
Fred Greene	9	Gardening
Phillip Porter	8	Running
Daniel Armstrong	8	Cycling
Donald Russell	9	Gardening
Susan Hernandez	10	Hiking
Mary Williamson	7	Cycling
Angela Moore	8	Sightseeing

Με το παραπάνω ερώτημα ολοκληρώσαμε τον έλεγχο της υλοποίησης. Έχουμε απαντήσει πλέον στα ερωτήματα που μας ώθησαν να δημιουργήσουμε τα μοντέλα μας και έχουμε διαπιστώσει την ορθότητα τους. Αναλυτική καταγραφή όλων των τετριμμένων ερωτημάτων μπορείτε να δείτε ανατρέχοντας στο αρχείο ‘queries.csv’ της εργασίας (βλέπε παράρτημα).

7 Δημιουργία Συστάσεων

Στο παρόν κεφάλαιο θα δούμε πως μπορούμε να χρησιμοποιήσουμε τη Neo4j για παραγωγή συστάσεων. Όλη η ανάλυση που έχει προηγηθεί μας οδηγεί σε αυτό ακριβώς το σημείο όπου θέλουμε να επικεντρώσουμε την προσοχή μας. Στη μελέτη, δηλαδή, των μεθόδων και των πρακτικών με τις οποίες μπορούμε να χρησιμοποιήσουμε τη βάση δεδομένων γράφων που κατασκευάσαμε για την παραγωγή συστάσεων πραγματικού χρόνου προς τους χρήστες της εφαρμογής μας.

Θα ξεκινήσουμε με μια συνοπτική αναφορά στα συστήματα συστάσεων και θα ακολουθήσει η πορεία δημιουργίας ερωτημάτων συστάσεων με βάση τα μοντέλα που έχουμε κατασκευάσει έως τώρα.

7.1 Συστήματα συστάσεων

Τα συστήματα συστάσεων είναι εργαλεία λογισμικού και τεχνικές για την παραγωγή συστάσεων σε χρήστες εφαρμογών και ιστοσελίδων. Ο γενικός όρος που χρησιμοποιείται για να δηλώσει τα στοιχεία που προτείνει ένα σύστημα συστάσεων είναι ο όρος "*αντικείμενο*" και κάθε σύστημα συστάσεων επικεντρώνεται σε ένα πολύ συγκεκριμένο είδος "*αντικειμένου*" (π.χ. ταινίες, ειδήσεις, δραστηριότητες). Το *αντικείμενο* αυτό, καθορίζει τον τρόπο με τον οποίο θα σχεδιαστεί το σύστημα καθώς και τις τεχνικές που θα ακολουθηθούν για την παραγωγή συστάσεων.

Οι συστάσεις που παράγονται είναι συνήθως προσωποποιημένες. Έτσι, διαφορετικοί χρήστες ή ομάδες χρηστών δέχονται διαφορετικά αποτελέσματα συστάσεων. Υπάρχουν ωστόσο και μη-προσωποποιημένες συστάσεις που είναι ευκολότερο να υλοποιηθούν. Ένα παράδειγμα τέτοιων συστάσεων είναι τα 5 πιο δημοφιλή τραγούδια μιας εβδομάδας ή τα ευπώλητα βιβλία ενός βιβλιοπωλείου.

Στην απλούστερη τους μορφή, οι συστάσεις είναι ταξινομημένες λίστες αντικειμένων με βάση κάποια βαθμολογία. Για την παραγωγή αυτής της βαθμολογίας, τα συστήματα συστάσεων βασίζονται στις προτιμήσεις και τις ιδιαιτερότητες των χρηστών, τις οποίες συλλέγουν είτε άμεσα από δηλώσεις των ίδιων των χρηστών ή αξιολογήσεις αντικειμένων, είτε έμμεσα ερμηνεύοντας την δραστηριότητα των χρηστών.

Υπάρχει μια μεγάλη ποικιλία τεχνικών που χρησιμοποιούνται από τα συστήματα συστάσεων. Μπορούμε όμως να ομαδοποιήσουμε τις τεχνικές αυτές σε δύο βασικές μεθόδους:

1. **Φιλτράρισμα με βάση το περιεχόμενο (content-based filtering).** Το φιλτράρισμα με βάση το περιεχόμενο είναι η μέθοδος παραγωγής συστάσεων που βασίζεται σε ανάλυση τόσο των χαρακτηριστικών του αντικειμένου προς σύσταση όσο και των προτιμήσεων του χρήστη-στόχου. Παράδειγμα χρήσης της συγκεκριμένης μεθόδου, είναι η σύσταση ταινιών σε κάποιο χρήστη που γνωρίζουμε ότι του αρέσουν οι περιπέτειες και οι ταινίες επιστημονικής φαντασίας. Για να προτείνουμε κάποια ταινία στο συγκεκριμένο χρήστη θα ανατρέξουμε στα χαρακτηριστικά των ταινιών που έχουμε διαθέσιμες και θα του προτείνουμε ταινίες του είδους 'περιπέτεια' ή/και 'επιστημονική φαντασία'.
2. **Συνεργατικό φιλτράρισμα (collaborative filtering)**. Το συνεργατικό φιλτράρισμα είναι η μέθοδος παραγωγής συστάσεων που βασίζεται στη συμπεριφορά και στο ιστορικό ενός χρήστη (π.χ. αντικείμενα που έχει αγοράσει ή δραστηριότητες που έχει βαθμολογήσει ο

χρήστης στο παρελθόν), καθώς και στην ομοιότητα της συμπεριφοράς του με τη συμπεριφορά άλλων χρηστών. Ουσιαστικά η μέθοδος συλλέγει και αναλύει πληροφορίες για τη συμπεριφορά των χρηστών και στη συνέχεια προβλέπει τι είναι πιθανό να αρέσει σε κάποιο χρήστη με βάση την ομοιότητά που έχει με άλλους χρήστες. Ένα μεγάλο πλεονέκτημα στο συνεργατικό φιλτράρισμα είναι το ότι δεν στηρίζεται σε ανάλυση περιεχομένου και ως εκ τούτου είναι ικανό να προτείνει σύνθετα αντικείμενα (π.χ. ταινίες) χωρίς να χρειάζεται να 'καταλάβει' την ουσία των ίδιων των αντικειμένων.

Άλλες τεχνικές που χρησιμοποιούνται συχνά είναι οι συστάσεις με βάση τα δημογραφικά χαρακτηριστικά του πληθυσμού (demographic recommendations) ή οι συστάσεις με βάση τις σχέσεις μεταξύ των μελών ενός κοινωνικού συνόλου (social recommendations). Στην πράξη όμως, οι περισσότερες εφαρμογές χρησιμοποιούν υβριδικά συστήματα συστάσεων. Συστήματα, δηλαδή, που εκμεταλλεύονται τους συνδυασμούς διαφορετικών τεχνικών για την παραγωγή των αποτελεσμάτων τους. Χαρακτηριστική προς αυτή την κατεύθυνση είναι και η δήλωση των μελών της ομάδας *BellKor* που κέρδισε το *Netflix prize* το Σεπτέμβριο του 2009:

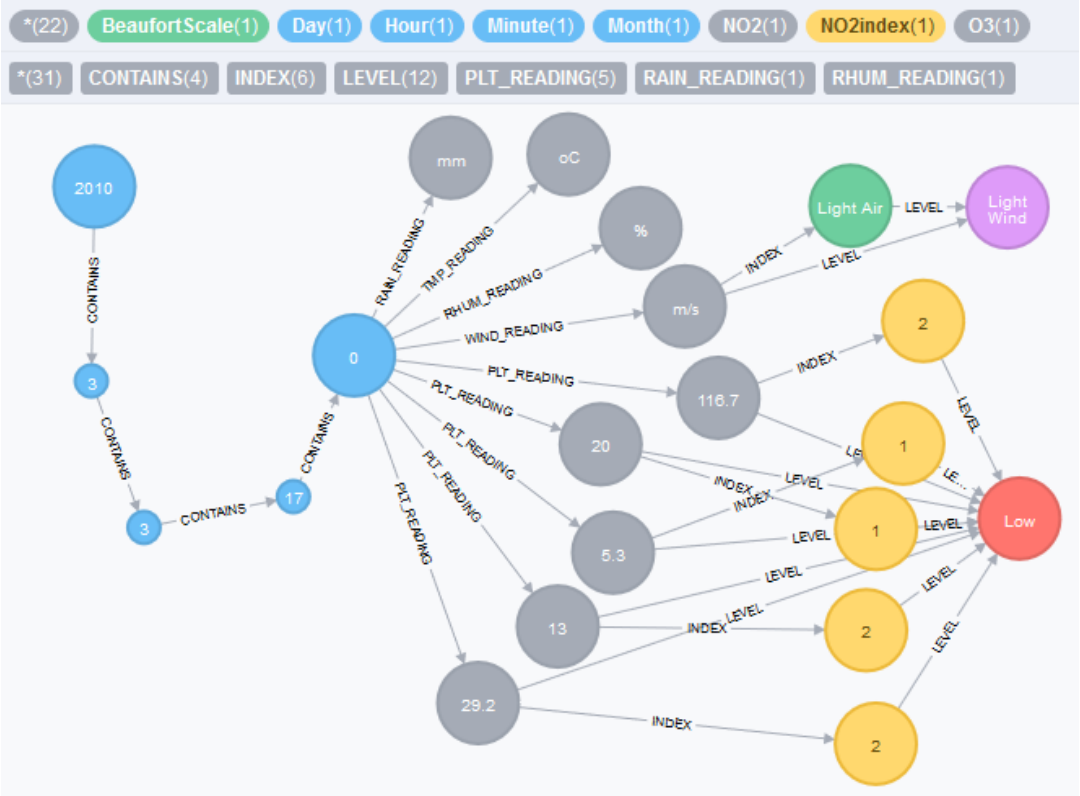
"Our final solution consists of blending 107 individual predictors. Predictive accuracy is substantially improved when blending multiple predictors. Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. Consequently, our solution is an ensemble of many methods."

Σε κάθε περίπτωση, οποιαδήποτε τεχνική και αν χρησιμοποιήσουμε, η ουσία παραμένει ίδια: "Οι αλγόριθμοι συστάσεων εγκαθιδρύουν σχέσεις μεταξύ χρηστών και αντικειμένων". Η παραγωγή αποτελεσματικών συστάσεων εξαρτάται από την κατανόηση των παραπάνω σχέσεων καθώς και από την κατανόηση της ποιότητας των σχέσεων αυτών. Οι Γράφοι είναι ίσως οι καταλληλότερες δομές για την αναπαράσταση των πυκνά συνδεδεμένων δομών δεδομένων που προκύπτουν από τις παραπάνω απαιτήσεις. Η αποθήκευση και η μελέτη των δεδομένων αυτών με χρήση βάσεων δεδομένων γράφων επιτρέπει σε μια εφαρμογή να απεικονίζει σε πραγματικό χρόνο τις επιπτώσεις των δραστηριοτήτων κάθε χρήστη και όχι να βασίζεται σε προϋπολογισμένα αποτελέσματα παλαιών δεδομένων.

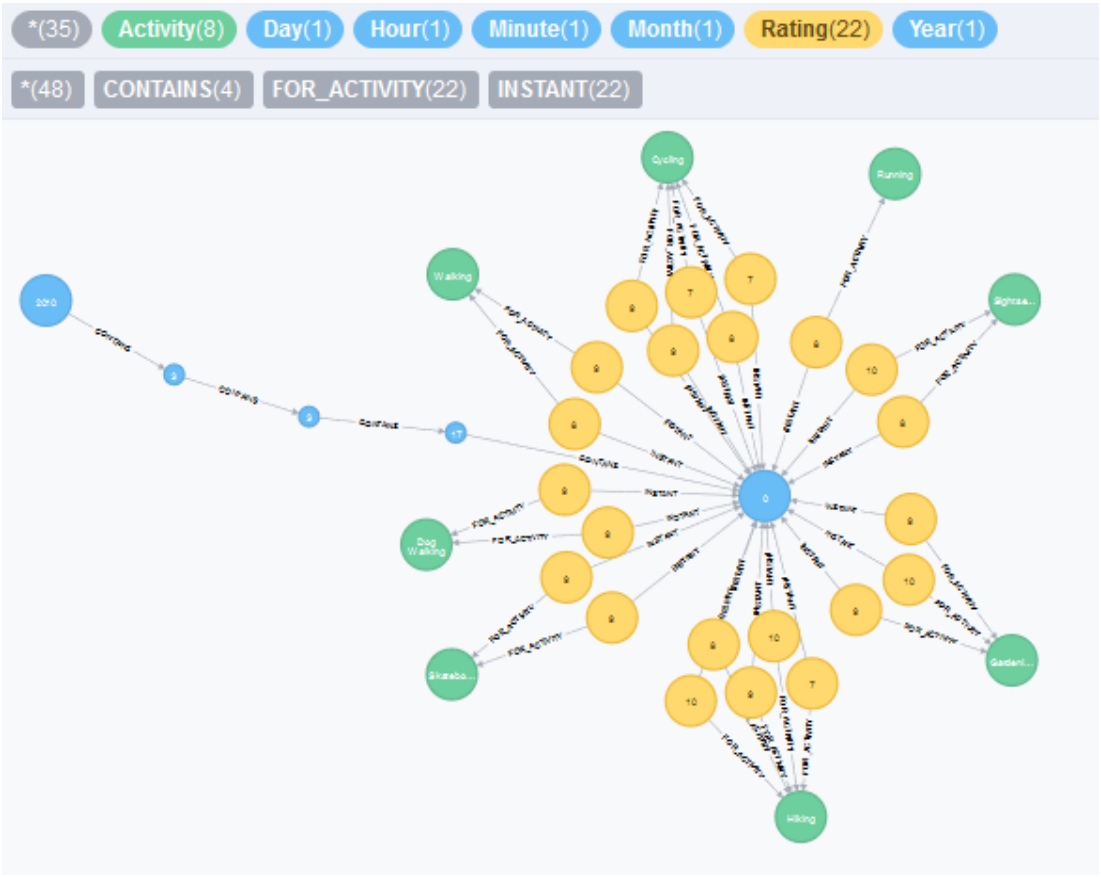
7.2 Ερωτήματα συστάσεων

Στην ενότητα αυτή θα παρουσιάσουμε ορισμένα ενδιαφέροντα ερωτήματα που θα μας επιτρέψουν να παράγουμε συστάσεις για τους χρήστες της εφαρμογής μας με βάση τη γνώση που έχουμε για το πεδίο μας. Έχουμε επιλέξει ενδεικτικά μια ημέρα και ώρα (03/03/2010 17:00) με σκοπό να δείξουμε ερωτήματα που παράγουν αποτελέσματα σε πραγματικό χρόνο και βασίζονται στις πιο πρόσφατες πληροφορίες που είναι αποθηκευμένες στη βάση μας για το χρόνο αυτό.

Πριν ξεκινήσουμε τη μελέτη των ερωτημάτων ας δούμε πρώτα τα κομμάτια του γράφου στα οποία θα επικεντρωθούν τα ερωτήματά μας. Στην πρώτη εικόνα που ακολουθεί βλέπουμε τις μετρήσεις των καιρικών συνθηκών και των συγκεντρώσεων των ρύπων, ενώ στη δεύτερη εικόνα έχουμε μια απεικόνιση των αξιολογήσεων που έχουν δοθεί από χρήστες της εφαρμογής σε πραγματικό χρόνο. Τόσο οι μετρήσεις όσο και τα ratings των χρηστών αφορούν το πρώτο τέταρτο της ώρας, (όπως έχουμε αναλύσει στην παράγραφο 3.4.4).



Εικόνα 7.1 Δεδομένα καιρικών συνθηκών και συγκεντρώσεις ρύπων για τις 03/03/2010 και ώρα 17:00



Εικόνα 7.2 Αξιολογήσεις δραστηριοτήτων για τις 03/03/2010 και ώρα 17:00

7.2.1 Δημοφιλείς δραστηριότητες

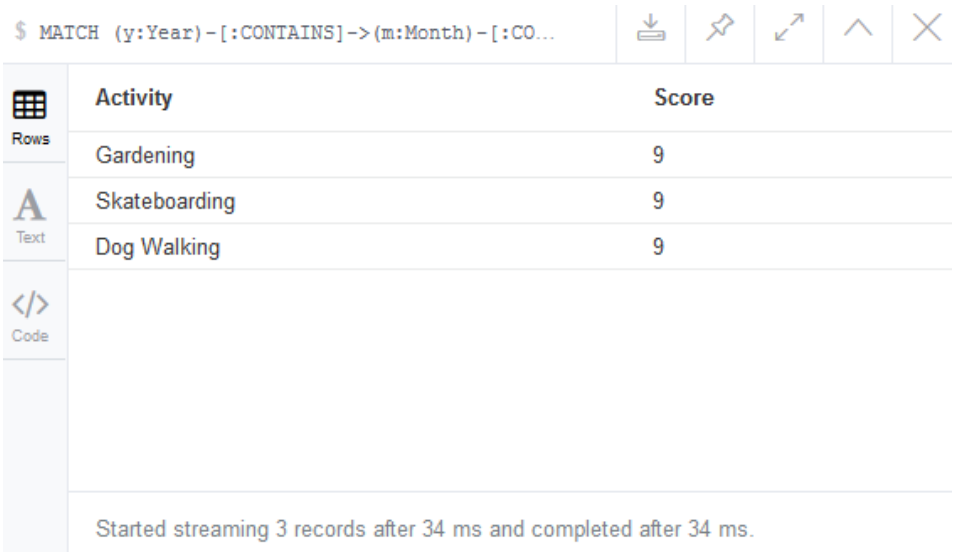
Το πρώτο ερώτημα με το οποίο θα ασχοληθούμε είναι η εύρεση των δημοφιλέστερων δραστηριοτήτων στην παρούσα στιγμή. Για το σκοπό αυτό αρκεί να ομαδοποιήσουμε τις βαθμολογίες των χρηστών ανάλογα με τη δραστηριότητα στην οποία απευθύνονται και στη συνέχεια να υπολογίσουμε το μέσο όρο αξιολογήσεων για κάθε δραστηριότητα.

Το ερώτημα Cypher επισυνάπτεται παρακάτω:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
MATCH (min)-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a:Activity)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

Με το πρώτο `MATCH` στο παραπάνω ερώτημα βρίσκουμε το κόμβο-λεπτού που μας ενδιαφέρει διασχίζοντας τα δέντρα χρόνου. Με το δεύτερο `MATCH` συγκεντρώνουμε της βαθμολογίες που έχουν δοθεί για τις δραστηριότητες, ενώ με το `WITH` αποθηκεύουμε στις μεταβλητές "RatingSum" και "RatingsNum" το άθροισμα και τον αριθμό των αξιολογήσεων αντίστοιχα για κάθε δραστηριότητα. Τέλος, επιστρέφουμε το όνομα κάθε δραστηριότητας μαζί με το μέσο όρο των βαθμολογιών που έχει προκύψει για αυτή (από την ακέραια διαίρεση των "RatingSum" και "RatingsNum") σε φθίνουσα σειρά. Το `LIMIT 3` που υπάρχει στο τέλος του ερωτήματος προέκυψε από τη σύμβαση να μην επιστρέφουμε στο χρήστη όλη τη διαθέσιμη λίστα αλλά μόνο τα τρία πιο δημοφιλή αποτελέσματα.

Το αποτέλεσμα του ερωτήματος επιστρέφει σε 34ms και παρουσιάζεται στην ακόλουθη εικόνα:

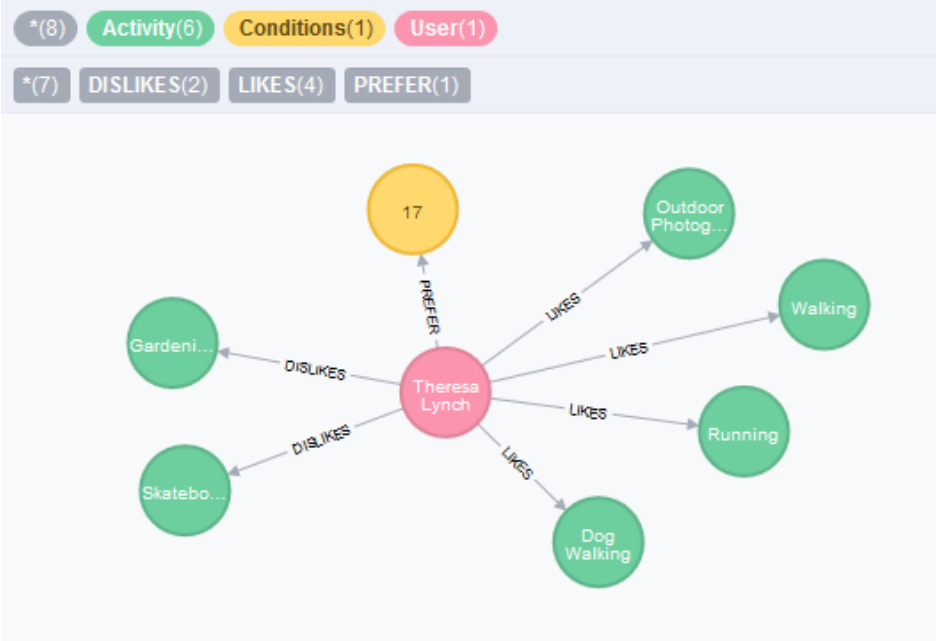


Να σημειώσουμε ότι παρά την απλότητα του, το συγκεκριμένο ερώτημα είναι αρκετά χρήσιμο σε πολλές περιπτώσεις. Για παράδειγμα, σε περίπτωση που οποιαδήποτε άλλη μέθοδος που θα

χρησιμοποιήσουμε για την παραγωγή συστάσεων αποτύχει, τότε μπορούμε πολύ γρήγορα να πάρουμε αποτελέσματα από το παραπάνω ερώτημα. Επίσης, είναι χρήσιμο σε περίπτωση που δεν έχουμε κάποια άλλη διαθέσιμη γνώση για το χρήστη-στόχο, είτε γιατί δεν έχει δηλώσει ο ίδιος τις προτιμήσεις του, είτε γιατί δεν υπάρχει ιστορικό της δραστηριότητας του.

7.2.2 Προτιμήσεις χρηστών

Τι θα γίνει όμως, αν στην παραπάνω περίπτωση θέλουμε να προτείνουμε δραστηριότητες στη χρήστη "Theresa Lynch";



Εικόνα 7.3 Προτιμήσεις δραστηριοτήτων για τη χρήστη "Theresa Lynch"

Παρατηρούμε ότι η συγκεκριμένη χρήστης έχει δηλώσει πως αντιπαθεί της δραστηριότητες "Gardening" και "Skateboarding". Συνεπώς, από τα τρία αποτελέσματα που επιστρέψαμε στην προηγούμενη παράγραφο, τα δύο είναι ουσιαστικά άχρηστα για τη συγκεκριμένη χρήστρια.

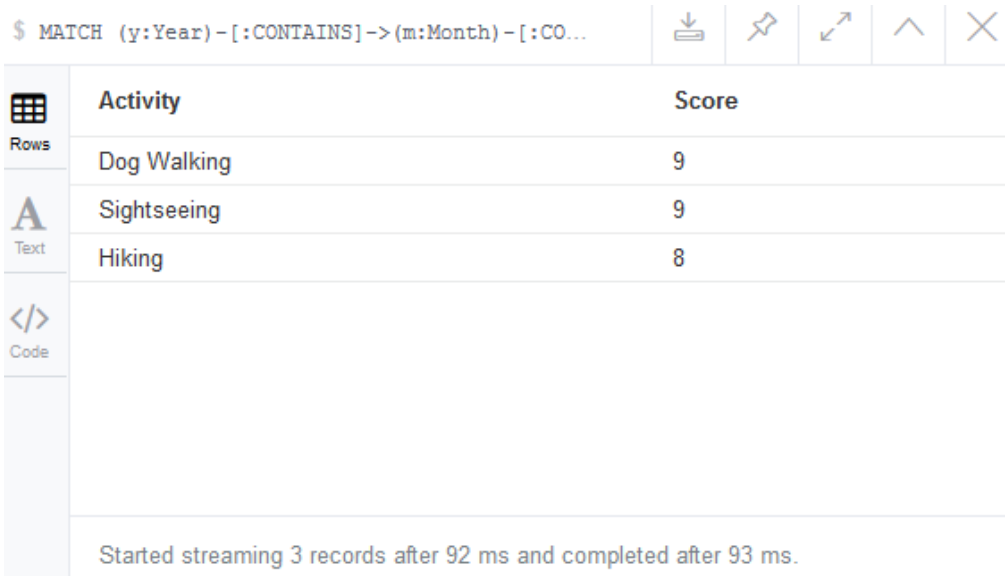
Ένας από τους βασικότερους κανόνες για να μπορούμε να παράγουμε αξιόλογα αποτελέσματα συστάσεων είναι: να μην επιστρέφουμε στο χρήστη αντικείμενα τα οποίο γνωρίζουμε πως δεν του αρέσουν. Αυτό μας οδηγεί στο επόμενο ερώτημα cypher:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
MATCH (theresa:User {username: "theresalynch34"}),
    (min)-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
```

```
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

Το παραπάνω ερώτημα επιστρέφει πάλι τις δημοφιλέστερες δραστηριότητες, με τη διαφορά ότι από την αξιολόγηση, έχουμε διαγράψει τις δραστηριότητες που δεν αρέσουν στη Theresa. Παρατηρούμε αυτή τη φορά, ότι στο δεύτερο `MATCH` βρίσκουμε όχι μόνο τις αξιολογήσεις των δραστηριοτήτων αλλά και το χρήστη-στόχο του ερωτήματός μας. Επίσης, με το `WHERE NOT` που συνοδεύει το συγκεκριμένο `MATCH`, δηλώνουμε πως στο ταίριασμα δεν θέλουμε να συμπεριληφθούν οι δραστηριότητες που αντιπαθεί ο χρήστης μας.

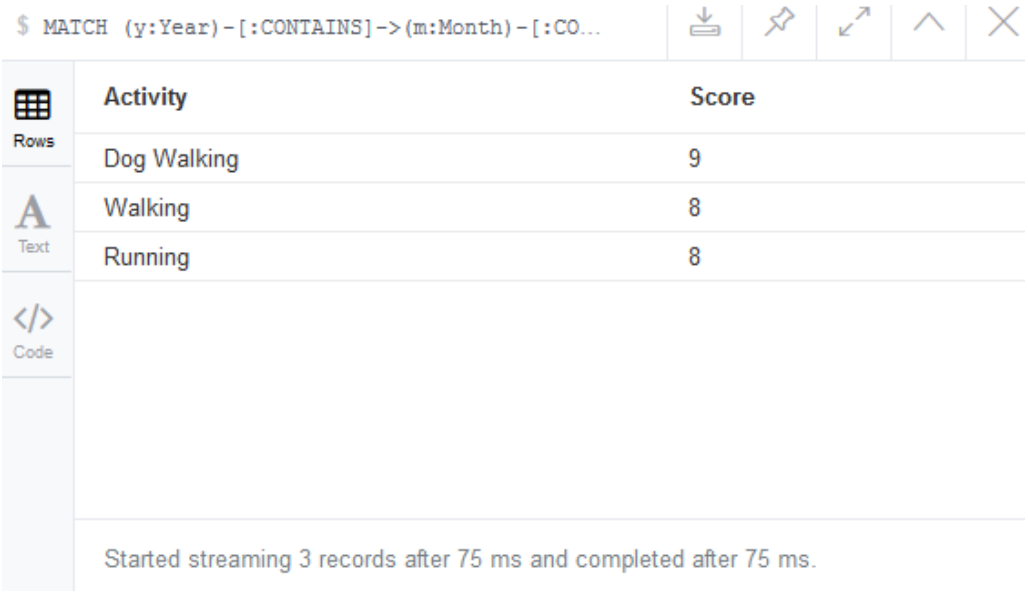
Το αποτέλεσμα του ερωτήματος επιστρέφει σε 93ms και παρουσιάζεται στην ακόλουθη εικόνα:



Ένα ακόμα ερώτημα που λαμβάνει υπόψιν τις προτιμήσεις του χρήστη είναι το ακόλουθο:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
MATCH (theresa:User {username: "theresalynch34"})-[:LIKES]->(a:Activity),
    (min)-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

Στο ερώτημα αυτό επιστρέφουμε συστάσεις μόνο με δραστηριότητες που αρέσουνε στο χρήστη-στόχο:



Ένα τέτοιο ερώτημα είναι χρήσιμο σε περιπτώσεις, όπου ο χρήστης θα ζητήσει να δει την αξιολόγηση αποκλειστικά και μόνο των δραστηριοτήτων που του αρέσουν. Παρά το ότι οι δραστηριότητες που προτείνονται στο χρήστη, μέσω του παραπάνω ερωτήματος, είναι δραστηριότητες που σίγουρα του αρέσουν, υπάρχει ένα σημαντικό μειονέκτημα στη συγκεκριμένη προσέγγιση. Με ένα τέτοιο ερώτημα, τα αποτελέσματα των συστάσεων μας περιορίζονται σε ένα πολύ συγκεκριμένο υποσύνολο δραστηριοτήτων (ίσο με το σύνολο των δραστηριοτήτων που αρέσουν στο χρήστη-στόχο) και δεν μπορούμε να προτείνουμε καινούργιες δραστηριότητες που ενδεχομένως να αρέσουν στο χρήστη μας.

7.2.3 Δημογραφικά χαρακτηριστικά χρηστών

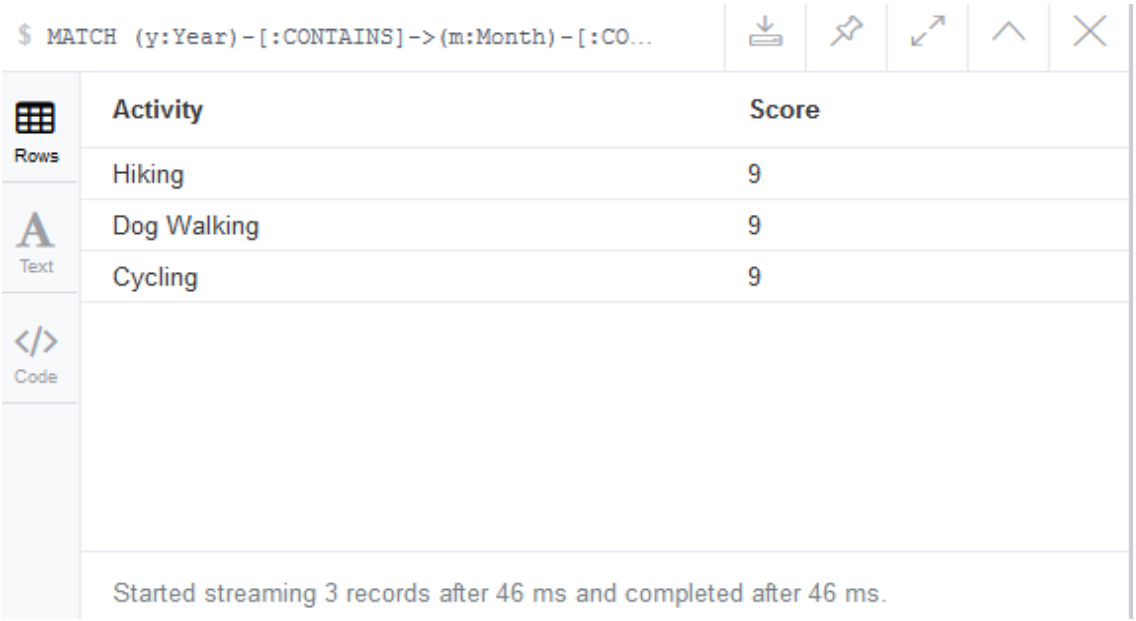
Μια ακόμα χρήσιμη πληροφορία που έχουμε στη διάθεσή μας και μπορούμε να την εκμεταλλευτούμε για την παραγωγή πιο στοχευμένων συστάσεων, είναι τα δημογραφικά χαρακτηριστικά των χρηστών μας.

Για παράδειγμα, μπορούμε να υλοποιήσουμε το ακόλουθο ερώτημα:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
MATCH (theresa:User {username: "theresalynch34"}),
    (min)-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a:Activity),
    (r)<-[:GAVE]-(u:User)
WHERE (theresa.age - 5) <= u.age <= (theresa.age + 5)
    AND NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
```

```
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3;
```

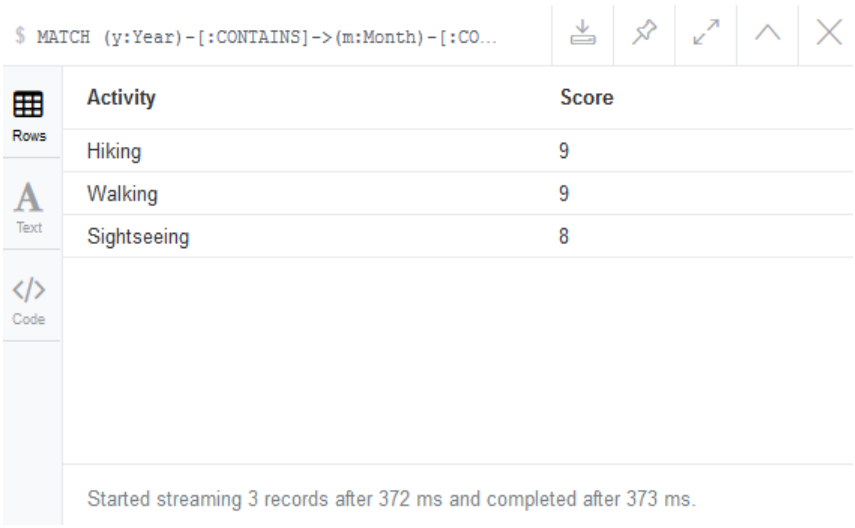
Για τα τελικά αποτελέσματα των συστάσεων, στο τελευταίο αυτό ερώτημα, λαμβάνονται υπόψιν μόνο βαθμολογίες που έχουν καταχωρηθεί από ηλικιακές ομάδες που βρίσκονται κοντά στην ηλικία του χρήστη-στόχου (±5 έτη):



Μπορούμε επίσης να υλοποιήσουμε το ερώτημα:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
MATCH (theresa:User {username: "theresalynch34"}),
    (min)-[:INSTANT]-(r:Rating)-[:FOR_ACTIVITY]->(a:Activity),
    (r)<-[:GAVE]-(u:User)
WHERE (theresa.age - 5) <= u.age <= (theresa.age + 5)
    AND theresa.gender = u.gender
    AND NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3;
```

Όπου λαμβάνεται υπόψιν τόσο η ηλικία, όσο και το φύλο του χρήστη στόχου. Τα αποτελέσματα επιστρέφουν σε 373ms και παρουσιάζονται παρακάτω:

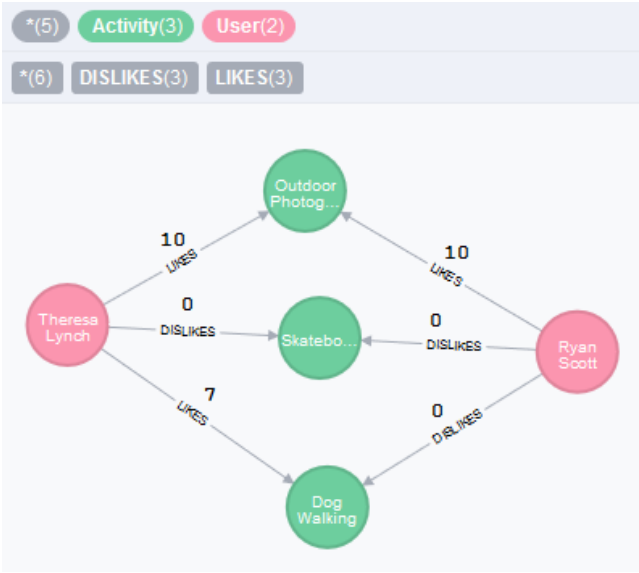


7.2.4 Μετρικές ομοιότητας (Similarity metrics)

Μια τεχνική που χρησιμοποιείται κατά κόρον σε συστήματα συστάσεων που βασίζονται στο συνεργατικό φιλτράρισμα (collaborative filtering) είναι η χρήση μετρικών ομοιότητας για τον προσδιορισμό της ομοιότητας μεταξύ των χρηστών της εφαρμογής. Στην παράγραφο αυτή θα παρουσιάσουμε δυο απλές μετρικές ομοιότητας καθώς και την απόδοση τους σε ερωτήματα προς τη βάση δεδομένων γράφων που έχουμε δημιουργήσει.

Το σκεπτικό πίσω από τη χρησιμοποίηση των μετρικών ομοιότητας είναι ότι μπορούμε, με βάση κάποια κριτήρια, να μοντελοποιήσουμε τους χρήστες της εφαρμογής ως διανύσματα και να υπολογίσουμε την ομοιότητα των διανυσμάτων αυτών. Στη συνέχεια, αφού εντοπίσουμε τους χρήστες που βρίσκονται πιο κοντά στο χρήστη-στόχο, μελετάμε τη συμπεριφορά τους και εξάγουμε τις ανάλογες συστάσεις.

Το κριτήριο που θα χρησιμοποιήσουμε στη δική μας εφαρμογή για τη μοντελοποίηση των χρηστών σε διανύσματα είναι οι προτιμήσεις σε δραστηριότητες.



Εικόνα 7.4 Προτιμήσεις χρηστών σε δραστηριότητες

Οι δύο χρήστες που απεικονίζονται στην παραπάνω εικόνα συνδέονται μεταξύ τους μέσω των προτιμήσεων σε δραστηριότητες που έχουν δηλώσει στην εφαρμογή. Ακόμα, σε κάθε προτίμηση

είναι αποθηκευμένη μια τιμή που αντιπροσωπεύει το βάρος-αρεσκείας του χρήστη για τη συγκεκριμένη δραστηριότητα. Μπορούμε λοιπόν, να μοντελοποιήσουμε τους χρήστες μας ως διανύσματα με τον ακόλουθο τρόπο:

$$\overrightarrow{user} = < Outdoor\ Photography, Skateboarding, Dog\ Walking >$$

$$\overrightarrow{theresa} = \langle 10, \quad 0, \quad 7 \rangle$$

$$\overrightarrow{ryan} = \langle 10, \quad 0, \quad 0 \rangle$$

Βλέπουμε ότι κάθε στήλη των διανυσμάτων αντιστοιχεί σε μια συγκεκριμένη δραστηριότητα και έχει τιμή ίση με την τιμή "value" της σχέσης που συνδέει το χρήστη με τη δραστηριότητα αυτή.

Ευκλείδεια απόσταση (Euclidean distance)

Η πρώτη μετρική που θα μελετήσουμε είναι η απλή ευκλείδεια απόσταση. Η μετρική αυτή περιγράφει την απόσταση δύο σημείων $x=(x_1, x_2)$ και $y=(y_1, y_2)$ στο καρτεσιανό επίπεδο με την ακόλουθη σχέση:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Ενώ, σε ένα χώρο n-διαστάσεων παίρνει την ακόλουθη μορφή:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

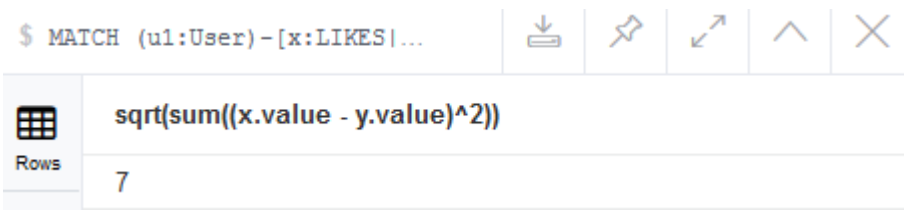
Με βάση την παραπάνω σχέση, μπορούμε να υπολογίσουμε την απόσταση των χρηστών της εικόνας 7.4 ως εξής:

$$d(\boldsymbol{theresa}, \boldsymbol{ryan}) = \sqrt{(10 - 10)^2 + (0 - 0)^2 + (7 - 0)^2} = 7$$

Την ίδια απόσταση, μπορούμε να υπολογίσουμε με χρήση της γλώσσας cypher και των μαθηματικών συναρτήσεων `sqrt()` και `sum()` που μας προσφέρει η γλώσσα:

```
MATCH (u1:User)-[x:LIKES|DISLIKES]->(a:Activity),
      (u2:User)-[y:LIKES|DISLIKES]->(a)
WHERE u1.username = "theresalynch34" AND
      u2.username = "ryanscott235"
RETURN sqrt(sum((x.value - y.value)^2))
;
```

Το αποτέλεσμα παραμένει το ίδιο:



Γενικεύοντας την παραπάνω διαδικασία έχουμε τη δυνατότητα να υπολογίσουμε με χρήση της cypher την απόσταση μεταξύ ενός χρήστη-στόχου και άλλων χρηστών της εφαρμογής:

```
MATCH (user:User)-[x:LIKES|DISLIKES]->(a:Activity),
      (otherUser:User)-[y:LIKES|DISLIKES]->(a)
WHERE user.username = "theresalynch34"
RETURN user.username, otherUser.username, sqrt(sum((x.value - y.value)^2))
;
```

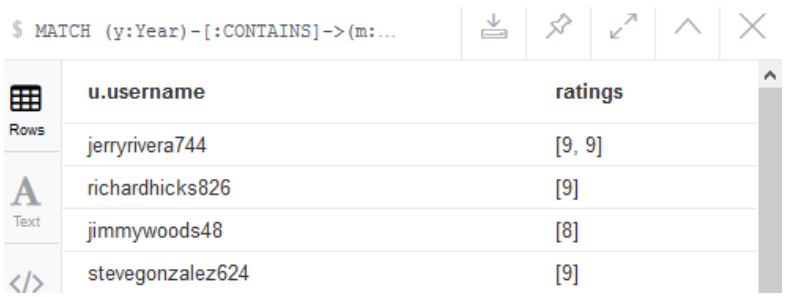
Είμαστε πλέον σε θέση να κατασκευάσουμε ένα πιο πολύπλοκο ερώτημα cypher που θα παράγει συστάσεις σε πραγματικό χρόνο χρησιμοποιώντας την ευκλείδεια απόσταση για να υπολογίσει την ομοιότητα μεταξύ ενός χρήστη-στόχου, και των χρηστών που έχουν δώσει βαθμολογίες για κάποιες δραστηριότητες σε μια συγκεκριμένη χρονική στιγμή. Το ερώτημα που θα παρουσιάσουμε, αναφέρεται στο κομμάτι του γράφου που απεικονίζεται στην εικόνα 7.2 και τα βήματα που ακολουθεί για τη δημιουργία των συστάσεων είναι:

1. Εύρεση του κόμβου-λεπτού που αντιστοιχεί στη χρονική στιγμή που εκτελείται το ερώτημα.
2. Συγκέντρωση όλων των χρηστών που έχουν βαθμολογήσει κάποια δραστηριότητα στο χρονικό παράθυρο που μελετάμε, καθώς και τις τιμές των βαθμολογιών αυτών.
3. Παραγωγή ζευγαριών διανυσμάτων για το χρήστη-στόχο και κάθε χρήστη του βήματος 2.
4. Υπολογισμός των αποστάσεων μεταξύ των διανυσμάτων με χρήση της σχέσης για την ευκλείδεια απόσταση.
5. Ταξινόμηση των αποστάσεων σε αύξουσα σειρά.
6. Επιλογή των 5 χρηστών με τη μικρότερη απόσταση από το χρήστη-στόχο.
7. Παραγωγή συστάσεων με βάση την τρέχουσα δραστηριότητα των χρηστών του βήματος 5.

Για καλύτερη κατανόηση του ερωτήματος θα παρουσιάσουμε σταδιακά των κώδικα cypher που υλοποιεί τα παραπάνω βήματα και θα καταλήξουμε στο συνολικό ερώτημα και την αξιολόγηση του.

Η υλοποίηση των δυο πρώτων βημάτων είναι σχετικά εύκολη και το ερώτημα cypher έχει ως εξής:

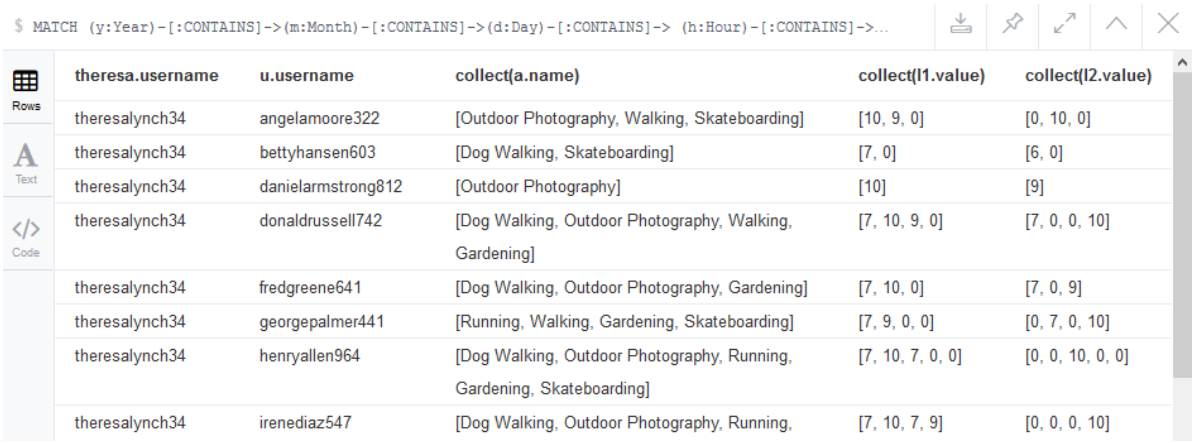
```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-
    (r:Rating)<-[:GAVE]-(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
RETURN u.username, collect(r.value) AS ratings
;
```



Τα ζευγάρια διανυσμάτων του βήματος 3 παράγονται με την προσθήκη του δεύτερου `MATCH`:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]-
>(a:Activity),
    (u)-[l2:LIKES|DISLIKES]->(a)
RETURN theresa.username, u.username, collect(a.name), collect(l1.value),
collect(l2.value)
ORDER BY u.username
;
```

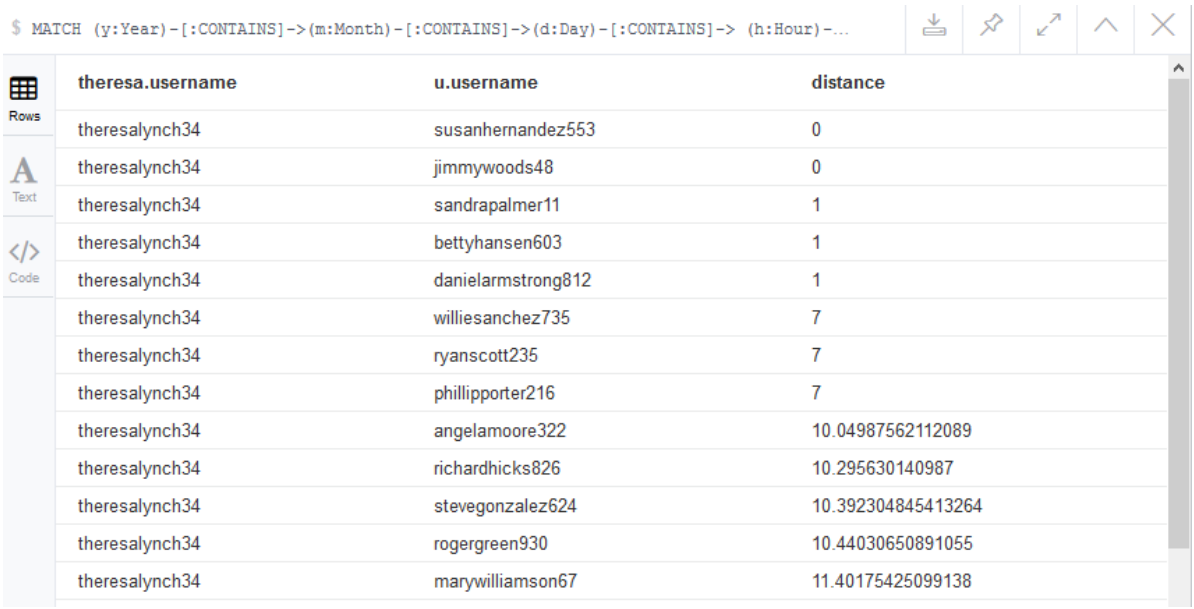
και ένα κομμάτι των αποτελεσμάτων φαίνεται στην ακόλουθη εικόνα:



theresa.username	u.username	collect(a.name)	collect(l1.value)	collect(l2.value)
theresalynch34	angelamoore322	[Outdoor Photography, Walking, Skateboarding]	[10, 9, 0]	[0, 10, 0]
theresalynch34	bettyhansen603	[Dog Walking, Skateboarding]	[7, 0]	[6, 0]
theresalynch34	danielarmstrong812	[Outdoor Photography]	[10]	[9]
theresalynch34	donaldrussell742	[Dog Walking, Outdoor Photography, Walking, Gardening]	[7, 10, 9, 0]	[7, 0, 0, 10]
theresalynch34	fredgreene641	[Dog Walking, Outdoor Photography, Gardening]	[7, 10, 0]	[7, 0, 9]
theresalynch34	georgepalmer441	[Running, Walking, Gardening, Skateboarding]	[7, 9, 0, 0]	[0, 7, 0, 10]
theresalynch34	henryallen964	[Dog Walking, Outdoor Photography, Running, Gardening, Skateboarding]	[7, 10, 7, 0, 0]	[0, 0, 10, 0, 0]
theresalynch34	irenediaz547	[Dog Walking, Outdoor Photography, Running,	[7, 10, 7, 9]	[0, 0, 0, 10]

Ακολουθεί ο υπολογισμός των αποστάσεων του βήματος 4 και η ταξινόμηση του βήματος 5:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]-
>(a:Activity),
    (u)-[l2:LIKES|DISLIKES]->(a)
RETURN theresa.username, u.username, sqrt(sum((l1.value - l2.value)^2)) AS
distance
ORDER BY distance
;
```

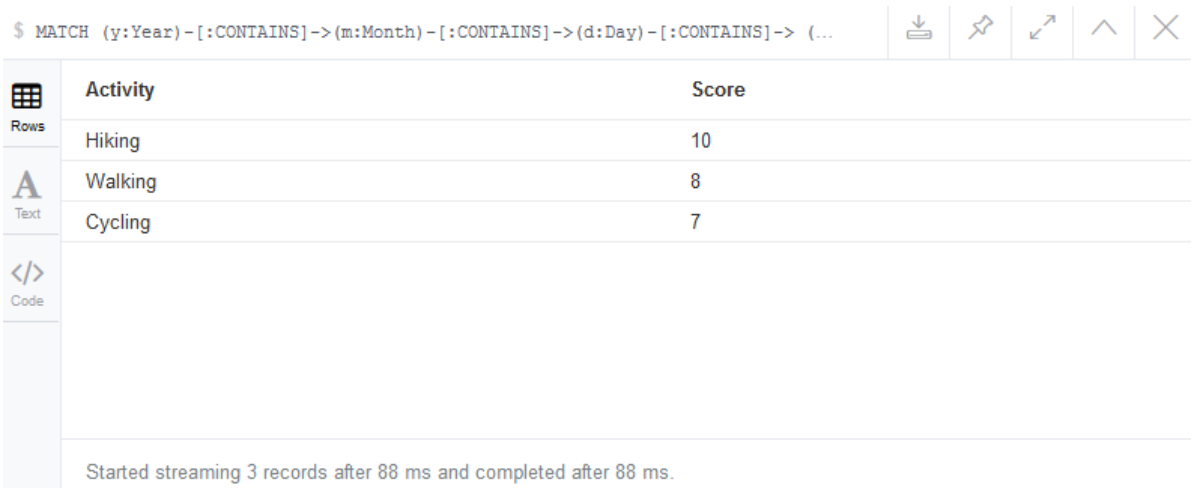


theresa.username	u.username	distance
theresalynch34	susanhernandez553	0
theresalynch34	jimmywoods48	0
theresalynch34	sandrapalmer11	1
theresalynch34	bettyhansen603	1
theresalynch34	danielarmstrong812	1
theresalynch34	williesanchez735	7
theresalynch34	ryanscott235	7
theresalynch34	phillipporter216	7
theresalynch34	angelamoore322	10.04987562112089
theresalynch34	richardhicks826	10.295630140987
theresalynch34	stevegonzalez624	10.392304845413264
theresalynch34	rogergreen930	10.44030650891055
theresalynch34	marywilliamson67	11.40175425099138

Τέλος, το ερώτημα ολοκληρώνεται με την επιλογή των 5 πιο κοντινών χρηστών (LIMIT 5) και την παραγωγή των συστάσεων:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]->
      (a:Activity),
      (u)-[l2:LIKES|DISLIKES]->(a)
WITH theresa, u, ratings, sqrt(sum((l1.value - l2.value)^2)) AS distance
ORDER BY distance LIMIT 5
UNWIND ratings AS r
MATCH (r)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

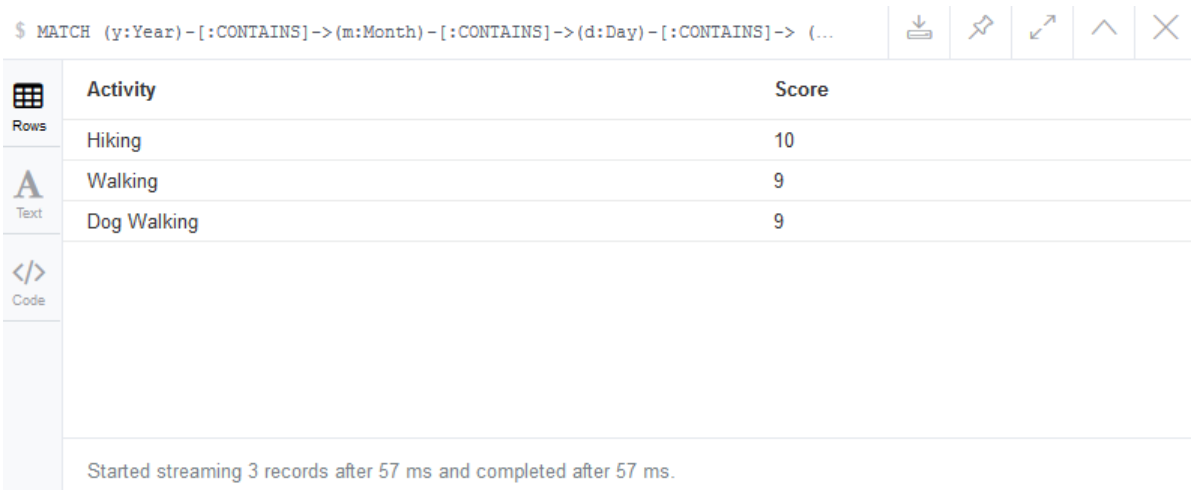
Το ερώτημα αυτό εκτελείται σε 88ms και παράγει τα ακόλουθα αποτελέσματα:



Μπορούμε επίσης, μέσα στα ζευγάρια διανυσμάτων να συμπεριλάβουμε ως πρώτο στοιχείο την ηλικία των χρηστών λαμβάνοντας έτσι υπόψιν τα δημογραφικά χαρακτηριστικά και παίρνοντας το ακόλουθο ερώτημα:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]-
>(a:Activity),
    (u)-[l2:LIKES|DISLIKES]->(a)
WITH theresa, u, ratings,
    sqrt((theresa.age - u.age)^2 + sum((l1.value - l2.value)^2)) AS
distance
ORDER BY distance LIMIT 5
UNWIND ratings AS r
MATCH (r)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

Με τα εξής αποτελέσματα:



Ομοιότητα συνημίτονου (cosine similarity)

Το cosine similarity είναι μια ακόμα μετρική που χρησιμοποιείται συχνά για τον υπολογισμό της ομοιότητας δύο μη μηδενικών διανυσμάτων. Η ομοιότητα δυο διανυσμάτων A και B, υπολογίζεται με βάση το συνημίτονο της γωνίας που σχηματίζουν και σε ένα χώρο n-διαστάσεων δίνεται από τη σχέση:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^{\,2}} \sqrt{\sum_{i=1}^{n} B_i^{\,2}}}$$

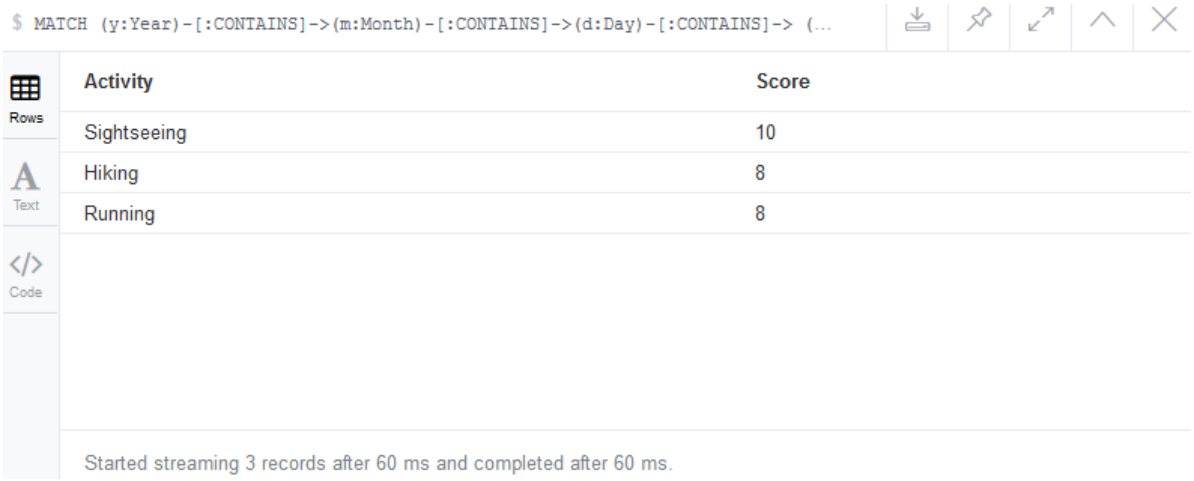
Το συνημίτονο της γωνίας δύο διανυσμάτων παίρνει τιμές από -1 έως 1, με τα δύο διανύσματα να θεωρούνται απόλυτα ίδια όταν η τιμή είναι 1 και τελείως αντίθετα όταν η τιμή είναι -1.

Η λογική στην υλοποίηση ενός ερωτήματος για συστάσεις δραστηριοτήτων με χρήστη του cosine similarity είναι ανάλογη της λογικής που χρησιμοποιήθηκε στην ευκλείδεια απόσταση και εδώ θα παρουσιάσουμε μόνο το ερώτημα που κατασκευάστηκε:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 3 AND
    h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES]-
>(a:Activity),
    (u)-[l2:LIKES]->(a)
WITH theresa, u, ratings, sum(l1.value * l2.value) AS l1l2Product,
    sqrt(REDUCE(l1Acc = 0.0, x IN collect(l1.value) | l1Acc + x^2)) AS
l1Length,
    sqrt(REDUCE(l2Acc = 0.0, y IN collect(l2.value) | l2Acc + y^2)) AS
l2Length
```

```
WITH theresa, u, ratings, l1l2Product / (l1Length * l2Length) AS
similarity
ORDER BY similarity DESC
LIMIT 5
UNWIND ratings AS r
MATCH (r)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

και τα αποτελέσματά του:



Να σημειωθεί ωστόσο, ότι στο παραπάνω ερώτημα δεν έχουμε συμπεριλάβει τις σχέσεις “dislikes” (όπως κατά τον υπολογισμό της ευκλείδειας απόστασης) καθώς οι σχέσεις αυτές έχουν μηδενικές τιμές και μπορεί να οδηγήσουν σε δημιουργία μηδενικών διανυσμάτων. Επίσης, να σημειωθεί ότι στο δικό μας πεδίο όλα τα συνημίτονα θα είναι θετικά και κοντά στην τιμή 1, αφού οι τιμές του “value” των σχέσεων “likes” κυμαίνονται από 1 έως 10.

Πριν ολοκληρώσουμε το κεφάλαιο αυτό και την παρουσίαση των ερωτημάτων που δημιουργήθηκαν κρίνουμε σκόπιμο να παραθέσουμε δύο ακόμα παραδείγματα εκτέλεσης. Συγκεκριμένα, θα εκτελέσουμε το ερώτημα που δημιουργήθηκε με χρήση της ευκλείδειας απόστασης σε δύο ακόμα περιπτώσεις με διαφορετικές συνθήκες ρύπανσης και διαφορετικό αριθμό αξιολογήσεων από αυτά που φαίνονται στις εικόνες 7.1 και 7.2.

Πρώτα θα ξεκινήσουμε με ένα παράδειγμα για τις 24/03/2010 και ώρα 10:15. Με το ερώτημα που ακολουθεί μπορούμε να δούμε το επίπεδο ατμοσφαιρικής ρύπανσης τη συγκεκριμένη στιγμή:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
```

```
    h.value = 10 AND min.value = 15
OPTIONAL MATCH (min)-[:PLT_READING]->()-[:INDEX]->(i)-[:LEVEL]-
>(pl:PollutantLevel)
RETURN pl.name AS Level, i.number AS Index
ORDER BY Index DESC LIMIT 1
;
```

Μπορούμε επίσης να μετρήσουμε τον αριθμό των αξιολογήσεων για τη συγκεκριμένη ημέρα και ώρα με το παρακάτω ερώτημα:

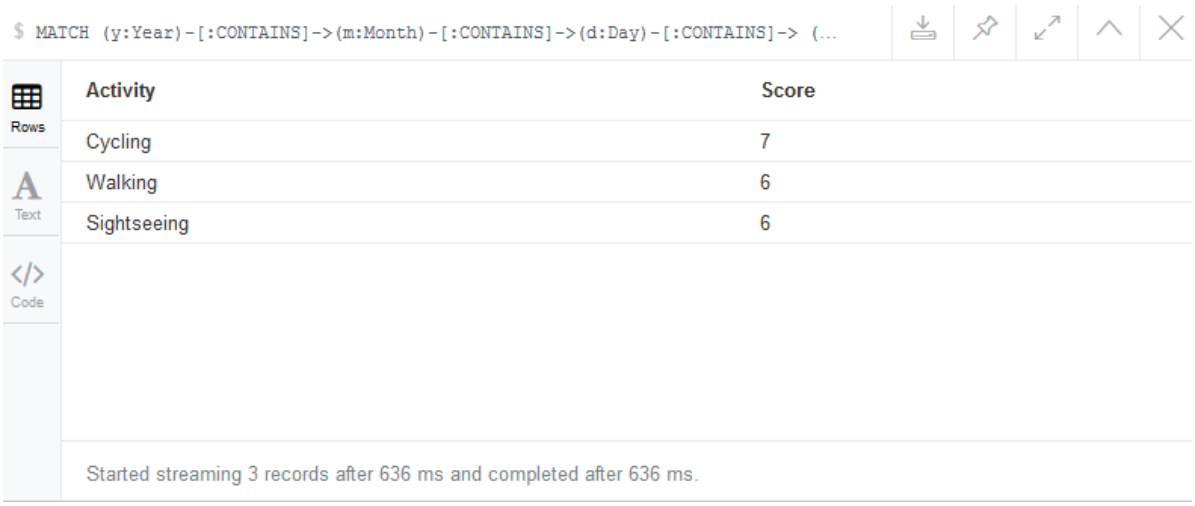
```
MATCH p1 = (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-
[:CONTAINS]->(h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
    h.value = 10 AND min.value = 15
RETURN count(r) AS `Number of Ratings`
;
```

Από τα παραπάνω αποτελέσματα βλέπουμε ότι το επίπεδο ατμοσφαιρικής ρύπανσης για τη συγκεκριμένη στιγμή είναι “Moderate” και ότι οι χρήστες της εφαρμογής έχουν καταχωρήσει 1057 αξιολογήσεις δραστηριοτήτων.

Εκτελώντας το ερώτημα σύστασης:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
    h.value = 10 AND min.value = 15
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]-
>(a:Activity),
    (u)-[l2:LIKES|DISLIKES]->(a)
WITH theresa, u, ratings,
    sqrt((theresa.age - u.age)^2 + sum((l1.value - l2.value)^2)) AS
distance
ORDER BY distance LIMIT 100
UNWIND ratings AS r
MATCH (r)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

παίρνουμε τα εξής αποτελέσματα συστάσεων:

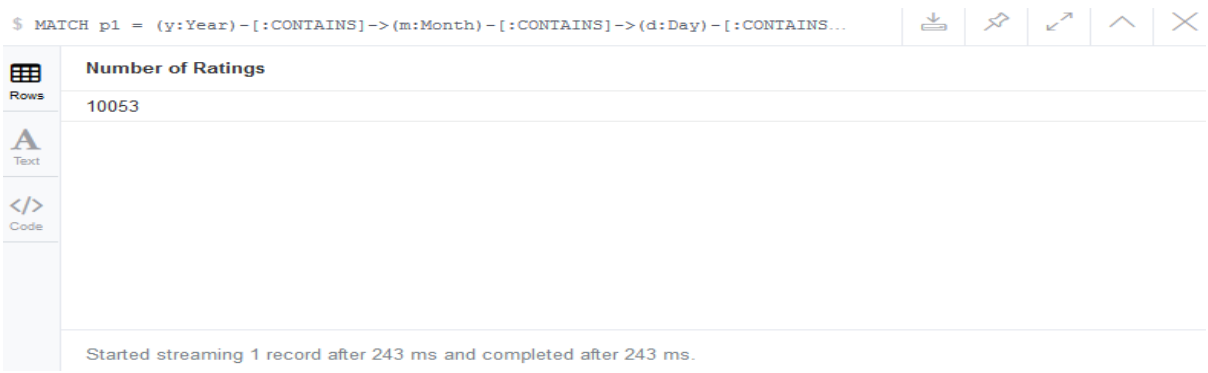


Θα κλείσουμε με ένα τελευταίο παράδειγμα για τις 24/03/2010 και ώρα 19:15. Όμοια με πριν βρίσκουμε ότι το επίπεδο ατμοσφαιρικής ρύπανσης είναι “Moderate”:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
    h.value = 19 AND min.value = 15
OPTIONAL MATCH (min)-[:PLT_READING]->()-[:INDEX]->(i)-[:LEVEL]-
>(pl:PollutantLevel)
RETURN pl.name AS Level, i.number AS Index
ORDER BY Index DESC LIMIT 1
;
```

και ότι έχουμε 10.053 αξιολογήσεις δραστηριοτήτων:

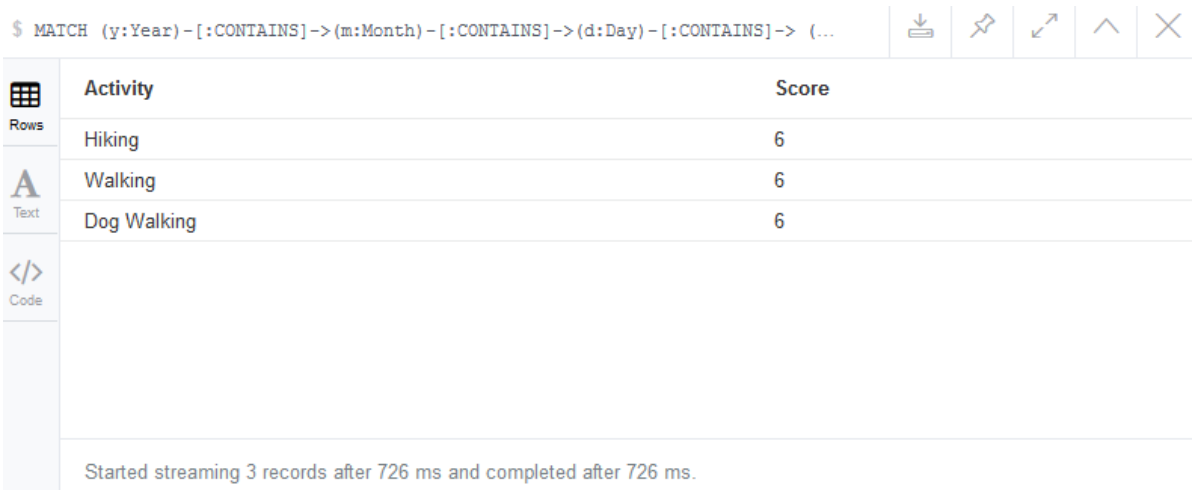
```
MATCH p1 = (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-
[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
    h.value = 19 AND min.value = 15
RETURN count(r) AS `Number of Ratings`
;
```



Εκτελώντας το ερώτημα σύστασης:

```
MATCH (y:Year)-[:CONTAINS]->(m:Month)-[:CONTAINS]->(d:Day)-[:CONTAINS]->
    (h:Hour)-[:CONTAINS]->(min:Minute)-[:INSTANT]-(r:Rating)<-[:GAVE]-
(u:User)
WHERE y.value = 2010 AND m.value = 3 AND d.value = 24 AND
    h.value = 19 AND min.value = 15
WITH u, collect(r) AS ratings
MATCH (theresa:User {username: "theresalynch34"})-[l1:LIKES|DISLIKES]-
>(a:Activity),
    (u)-[l2:LIKES|DISLIKES]->(a)
WITH theresa, u, ratings,
    sqrt((theresa.age - u.age)^2 + sum((l1.value - l2.value)^2)) AS
distance
ORDER BY distance LIMIT 100
UNWIND ratings AS r
MATCH (r)-[:FOR_ACTIVITY]->(a:Activity)
WHERE NOT (theresa)-[:DISLIKES]->(a)
WITH a, sum(r.value) AS RatingSum, count(r) AS RatingsNum
RETURN a.name AS Activity, (RatingSum / RatingsNum) AS Score
ORDER BY Score DESC, RatingsNum DESC
LIMIT 3
;
```

παίρνουμε τα αποτελέσματα:



Στο τελευταίο παράδειγμα, είχαμε σχεδόν 10.000 χρήστες στους οποίους έτρεξε ο αλγόριθμος του ερωτήματος σύστασης. Παρά το γεγονός αυτό, βλέπουμε πως η βάση δεδομένων μας επέστρεψε τα αποτελέσματα των συστάσεων σε 726ms. Ο συγκεκριμένος χρόνος είναι αυξημένος σε σχέση με τα προηγούμενα παραδείγματα, αλλά μπορούμε να πούμε ότι εξακολουθεί να βρίσκεται στα όρια που θεωρούνται αποδεκτά για την υλοποίηση συστήματος συστάσεων πραγματικού χρόνου.

8 *Επίλογος*

Στην εργασία αυτή μελετήσαμε τις δυνατότητες της βάσης δεδομένων γράφων Neo4j για την υποστήριξη συστήματος συστάσεων πραγματικού χρόνου. Αφού ολοκληρώσαμε μια σύντομη εισαγωγή στις κύριες έννοιες των βάσεων δεδομένων γράφων, παρουσιάσαμε στη συνέχεια τα πλεονεκτήματα τους έναντι της επιλογής των κλασικών σχεσιακών βάσεων. Κάναμε ειδική αναφορά στα χαρακτηριστικά της Neo4j, καθώς ήταν η βάση δεδομένων που χρησιμοποιήθηκε για την υλοποίηση της εργασίας μας, και παρουσιάσαμε αναλυτικά τα δεδομένα που αποτέλεσαν τον κύριο πυρήνα της εφαρμογής μας.

Περιγράψαμε αναλυτικά την πορεία μοντελοποίησης των διαθέσιμων δεδομένων και τους τρόπους αντιμετώπισης των προβλημάτων που συναντήσαμε σε αυτή. Ακολούθησε η υλοποίηση της εφαρμογής με αναφορά σε κομμάτια κώδικα της γλώσσας ερωτημάτων cypher, ο έλεγχος ορθής λειτουργίας και η υλοποίηση ερωτημάτων-συστάσεων τα οποία επέστρεφαν συστάσεις δραστηριοτήτων στους χρήστες της εφαρμογής.

Τέλος, παρατηρήσαμε ότι οι χρόνοι απόκρισης του συστήματος που δημιουργήθηκε ήταν εξαιρετικά γρήγοροι και τις περισσότερες φορές δεν ξεπερνούσαν τα 200 ms. Το γεγονός αυτό καθιστά ένα τέτοιο σύστημα ιδανικό για την υλοποίηση εφαρμογών συστάσεων πραγματικού χρόνου, σε αντίθεση με τα κλασικά σχεσιακά συστήματα που αδυνατούν να υποστηρίξουν αποδοτικά εφαρμογές που βασίζονται στις σχέσεις μεταξύ των δεδομένων.

Βιβλιογραφία

[1] Ian Robinson, Jim Webber, and Emil Eifrem, Graph Databases, O'Reilly Media, Inc., June 2015

[2] Rik Van Bruggen, Learning Neo4j, Packt Publishing, August 2014

[3] Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, and Jonas Partner, Neo4j in Action, Manning Publications, December 2014

[4] Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor, Recommender Systems Handbook, Springer New York, May 2010

[5] Mehmed Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms, Wiley-IEEE Press, September 2011

[6] Ago Luberg, Tanel Tammet, and Priit Hirv, Smart City: A Rule-based Tourist Recommendation System, Springer Vienna, 2011

[7] Smart Cities Open Data Guide, Advice, best practices and tools for creating a data-driven city, SmartCitiesCouncil.

[8] Xavier Amatriain, Recommender Systems Collaborative Filtering and other approaches, Netflix, July 2014

[9] Andrea Caragliu, Chiara Del Bo, and Peter Jijkamp, Smart cities in Europe, 3rd Central European Conference in Regional Science – CERS, 2009

[10] Markus Helfert, Karl-Heinz Krempels, Cornel Klein, Brian Donnellan, Oleg Gusikhin (Eds.), Smart Cities, Green Technologies, and Intelligent Transport Systems, Springer International Publishing, 2015

[11] https://neo4j.com/

[12] https://www.graphgrid.com/modeling-time-series-data-with-neo4j/

[13] http://graphaware.com/neo4j/2014/08/20/graphaware-neo4j-timetree.html

[14] http://www.jexp.de/blog/html/load_csv_tips.html

[15] https://neo4j.com/blog/real-time-recommendation-engine-data-science/

[16] https://neo4j.com/graphgist/a7c915c8-a3d6-43b9-8127-1836fecc6e2f

[17] https://en.wikipedia.org/wiki/Graph_database

[18] https://en.wikipedia.org/wiki/Smart_city

[19] https://en.wikipedia.org/wiki/Recommender_system

[20] http://www.londonair.org.uk/LondonAir/Default.aspx

[21] https://uk-air.defra.gov.uk/air-pollution/

Παράρτημα

Τον κώδικα για τη δημιουργία της βάσης δεδομένων καθώς και όλα τα αρχεία δεδομένων που χρησιμοποιήθηκαν στην παρούσα εργασία μπορείτε να τα βρείτε στον ακόλουθο σύνδεσμο:

https://drive.google.com/open?id=0B8RfqpH7xrOKeTlpTTVNaXJHQW8

Να σημειώσουμε ότι στο φάκελο cypher υπάρχουν τέσσερα αρχεία: Το αρχείο 'linux_cypher.csv' περιέχει τον κώδικα για τη δημιουργία της βάσης δεδομένων σε διανομές linux. Το αρχείο 'windows_cypher.csv' περιέχει των κώδικα για τη δημιουργία της βάσης δεδομένων σε windows. Το αρχείο 'queries.csv' περιέχει τα τετριμμένα ερωτήματα καθώς και τα ερωτήματα συστάσεων που δημιουργήθηκαν για τους σκοπούς της εργασίας. Το αρχείο 'testing.csv' περιέχει διάφορα ερωτήματα cypher για έλεγχο της βάσης δεδομένων και εξαγωγή συμπερασμάτων.