Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Θεωρία και πρακτική σύνοψης τυποποιημένων τριπλετών RDF** **Theory and practice of the RDF type summary**
Ονοματεπώνυμο Φοιτητή	**Σπυρίδων Καζάνας**
Πατρώνυμο	**Παντελής**
Αριθμός Μητρώου	**ΜΠΠΛ/ 11037**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2014**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χ. Δουληγέρης Καθηγητής	Π. Κοτζανικολάου Λέκτορας	Ι. Παπαδάκης Επίκουρος καθηγητής

Θεωρία και πρακτική σύνοψης τυποποιημένων τριπλετών RDF

Σπυρίδων Καζάνας

Abstract

The objective of this thesis is the developement of an intermediate platform which facilitates searching in RDF datastores (triplestores). The purpose of this platform is to assist the user to define SPARQL queries, without requiring knowledge of the language or the structure of the stored RDF data. The platform allows the user to select a set of words, which are stored in the RDF datastore, as well as to define variables. The platform can correlate the user's input and generate SPARQL queries, which can then be run against the RDF datastore in odrer to get potentialy useful results.

The platform is based on the production of a graph which is called the summary graph. This graph is equivalent to the original RDF graph but it is smaller than that. In this thesis we present the production method of the summary graph as well as the method of using it to meet the information needs of the end user.

Περίληψη

Αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη μιας ενδιάμεσης πλατφόρμας αναζήτησης πληροφοριών σε αποθήκες RDF (triplestores). Σκοπός της πλατφόρμας είναι η υποβοήθηση του χρήστη στον ορισμό ερωτημάτων SPARQL, χωρίς την απαίτηση γνώσης της γλώσσας, ή της δομής της αποθηκευμένης πληροφορίας RDF. Δίνεται η δυνατότητα στον χρήστη να επιλέξει ένα σύνολο από λεκτικά, που βρίσκονται αποθηκευμένα στην αποθήκη πληροφοριών και να ορίσει μεταβλητές. Η πλατφόρμα συσχετίζει τις πληροφορίες και παράγει ερωτήματα SPARQL, τα οποία μπορούν, κατόπιν, να εκτελεστούν στην αποθήκη RDF και να επιστρέψουν πιθανώς χρήσιμα αποτελέσματα.

Η πλατφόρμα στηρίζεται στην σύμπτυξη της πληροφορίας του αρχικού γραφήματος RDF και την παραγωγή ενός ισοδύναμου, μικρότερου και περιεκτικότερου γραφήματος, που ονομάζουμε σύνοψη RDF. Σε αυτή την εργασία παρουσιάζεται η μέθοδος παραγωγής αυτής της σύνοψης και ο τρόπος αξιοποίησής της για την κάλυψη των πληροφοριακών αναγκών του τελικού χρήστη.

Κατάλογος σχημάτων

Λίστα Αλγορίθμων

Περιεχόμενα

1 Εισαγωγή

Ο Σημασιολογικός Ιστός (Semantic Web) αποτελεί το επόμενο εξελικτικό στάδιο του Παγκόσμιου Ιστού (World Wide Web), στο οποίο οι πληροφορίες είναι καθολικά αναγνώσιμες από ανθρώπους αλλά και από μηχανές [3]. Αυτή η επέκταση επιτρέπει την αυτοματοποίηση της επεξεργασίας των πληροφοριών, που βρίσκονται κατανεμημένες σε πολλαπλά σημεία του Ιστού, δίνοντας τη δυνατότητα αποτελεσματικότερης αξιοποίησης των πληροφοριών από τους τελικούς χρήστες. Ο οργανισμός World Wide Web Consortium (W3C) είναι ο βασικός οργανισμός διεθνούς προτυποποίησης της οικογένειας τεχνολογιών που πλαισιώνουν τον Σημασιολογικό Ιστό.

Ένας χρήστης που επιθυμεί να αντλήσει πληροφορίες από μια αποθήκη RDF, είναι αναγκασμένος να χρησιμοποιήσει εξειδικευμένη γλώσσα ερωτημάτων SPARQL. Με τη χρήση της γλώσσας, ένας έμπειρος χρήστης μπορεί να ανακαλύψει τη δομή της αποθηκευμένης πληροφορίας και τελικά να οδηγηθεί στην διατύπωση κατάλληλων ερωτημάτων για την άντληση τις επιδιωκόμενης πληροφορίας.

Ο χρόνος εκμάθησης της γλώσσας και ο χρόνος διατύπωσης κατάλληλων ερωτημάτων αποτελούν περιοριστικούς παράγοντες, που μειώνουν την αποτελεσματικότητα αναζήτησης πληροφοριών σε αποθήκες RDF. Μάλιστα, ο δεύτερος παράγοντας οφείλεται στην άγνοια της δομής της αποθηκευμένης πληροφορίας. Στην εργασία αυτή σχεδιάζεται και κατόπιν υλοποιείται μέθοδος ελαχιστοποίησης των παραπάνω περιοριστικών παραγόντων, με αποτέλεσμα τη διευκόλυνση πρόσβασης του χρήστη στην αποθηκευμένη πληροφορία RDF.

Πιο συγκεκριμένα, επιχειρείται η αντιμετώπιση του προβλήματος με τον ορισμό κατάλληλης μεθόδου ανάλυσης των δεδομένων της αποθήκης RDF και τη δημιουργία εφαρμογής υποβοήθησης χρήστη, αποτελούμενης από γραφική διεπαφή χρήστη και μηχανή παραγωγής ερωτημάτων. Η ανάλυση των δεδομένων γίνεται μια φορά σε ένα προπαρασκευαστικό στάδιο. Το αποτέλεσμα της ανάλυσης αξιοποιείται από τη μηχανή παραγωγής ερωτημάτων και από τη γραφική διεπαφή χρήστη.

Στη γραφική διεπαφή παρουσιάζονται ανθρωπίνως αναγνωρίσιμα συστατικά της αποθήκης RDF, όπως είναι, για παράδειγμα, τα λεκτικά. Ο χρήστης καλείται να επιλέξει έναν ορισμένο αριθμό τέτοιων αναγνωρίσιμων συστατικών και να τα αποστείλει στη μηχανή παραγωγής ερωτημάτων, προκειμένου να λάβει ένα πλήθος από συντακτικώς ορθά ερωτήματα SPARQL. Στη συνέχεια, ακόμα και ο πιο άπειρος χρήστης, μπορεί να εκτελέσει αυτά τα ερωτήματα στην αποθήκη RDF και να λάβει πιθανώς χρήσιμες απαντήσεις. Ένας έμπειρος χρήστης μπορεί να βασιστεί πάνω σε κάποια από τα ερωτήματα και να τα προσαρμόσει όπως θέλει.

Στις επόμενες ενότητες γίνεται αναφορά στις σχετικές τεχνολογίες και παρουσιάζεται η μέθοδος αντιμετώπισης του προβλήματος. Οι πρώτες ενότητες αφιερώνονται στην αποτύπωση του θεωρητικού και εννοιολογικού πλαισίου της εργασίας. Στη συνέχεια παρουσιάζεται η θεωρητική προσέγγιση του προβλήματος. Η πρακτική προσέγγιση περιγράφεται στις τελευταίες ενότητες, όπου γίνεται εκτενής περιγραφή της υλοποίησης.

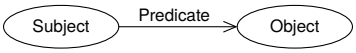
2 Σημασιολογικός Ιστός

Στις επόμενες ενότητες γίνεται αναφορά στις τεχνολογίες που σχετίζονται με αυτή την εργασία.

2.1 Αναγνωριστικά Πόρων

Ενιαίο Αναγνωριστικό Πόρου (URI - Universal Resource Identifier) Το URI αποτελεί γενίκευση του Ενιαίου Εντοπιστή Πόρου (URL - Uniform Resource Locator) και χρησιμεύει στην καθολική ταυτοποίηση ενός συγκεκριμένου πόρου στον Ιστό.

Διεθνοποιημένο Αναγνωριστικό Πόρων (IRI - Internationalized Resource Identifier) Το IRI είναι ένα αλφαριθμητικό Unicode το οποίο ακολουθεί το συντακτικό που ορίζεται στο έγγραφο RFC 3987.Τα IRI αποτελούν μια γενίκευση των URI που επιτρέπει τη χρήση μεγαλύτερου εύρους χαρακτήρων Unicode. Κάθε απόλυτο URI και URL αποτελεί ένα IRI. Αντίθετα, κάθε IRI δεν αποτελεί URI. Όταν κανείς χρησιμοποιεί IRI σε πράξεις που ορίζονται μόνο για URI, πρέπει πρώτα να τα μετατρέπει σε URI.



Εικόνα 2.1: Αναπαράσταση τριπλέτας RDF [11].

2.2 Πλαίσιο Περιγραφής Πόρων (RDF - Resource Description Framework)

Η τεχνολογία RDF είναι μια γενική μέθοδος εννοιολογικής περιγραφής πληροφορίας, με χρήση πληθώρας συντακτικών συμβολισμών και μορφών αποθήκευσης δεδομένων. Η τεχνολογία RDF αποτελεί προδιαγραφή του World Wide Web Consortium (W3C). Η πιο πρόσφατη έκδοση της προδιαγραφής RDF, η οποία ανακοινώθηκε το 2014, είναι η έκδοση 1.1 [11].

Το μοντέλο δεδομένων RDF ομοιάζει με τα διαγράμματα entity–relationship και τα διαγράμματα κλάσεων, αφού βασίζεται στη διατύπωση προτάσεων σχετικά με πόρους (resources) και συγκεκριμένα δικτυακούς πόρους. Κάθε πρόταση έχει τη μορφή τριπλέτας (triple):

υποκείμενο (subject), κατηγόρημα (predicate), αντικείμενο (object)

Το υποκείμενο μιας τριπλέτας μπορεί να είναι IRI (βλ. 2.1), ή κενός κόμβος (blank node). Το κατηγόρημα μπορεί να είναι μόνο IRI. Το αντικείμενο μπορεί να είναι IRI, κενός κόμβος ή λεκτικό. Το υποκείμενο υποδηλώνει τον δικτυακό πόρο, ενώ το κατηγόρημα υποδηλώνει κάποια ιδιότητά του, εκφράζοντας τη συσχέτιση μεταξύ υποκειμένου και αντικειμένου.

Αξίζει να αναφερθεί ότι το IRI, ο κενός κόμβος και το λεκτικό, ονομάζονται μαζί ως όροι RDF (RDF terms). Επίσης, το IRI και το λεκτικό αναφέρονται μαζί ως πόροι (resources). Σε αντίθεση με τα IRI και τα λεκτικά, ένας κενός κόμβος δεν αποτελεί αναγνωριστικό πόρου.

Λεκτικά (Literals) Τα λεκτικά αποτελούν τα πραγματικά δεδομένα ενός γραφήματος RDF. Ένα λεκτικό μπορεί να είναι τυποποιημένο (typed literal), ή απλό (plain literal). Τα απλά λεκτικά δεν έχουν τύπο δεδομένων (datatype) [11].

2.2.1 Γράφημα RDF

Ένα σύνολο τριπλετών RDF ορίζει ένα κατευθυνόμενο πολυγράφημα, όπου κάθε ακμή αντιστοιχεί μονοσήμαντα σε μία τριπλέτα [11]. Η φορά της ακμής ορίζεται από το υποκείμενο προς το αντικείμενο (εικ. 2.1). Το σύνολο κόμβων ενός γραφήματος RDF ταυτίζεται με το σύνολο των υποκειμένων και αντικειμένων των τριπλετών του γραφήματος.

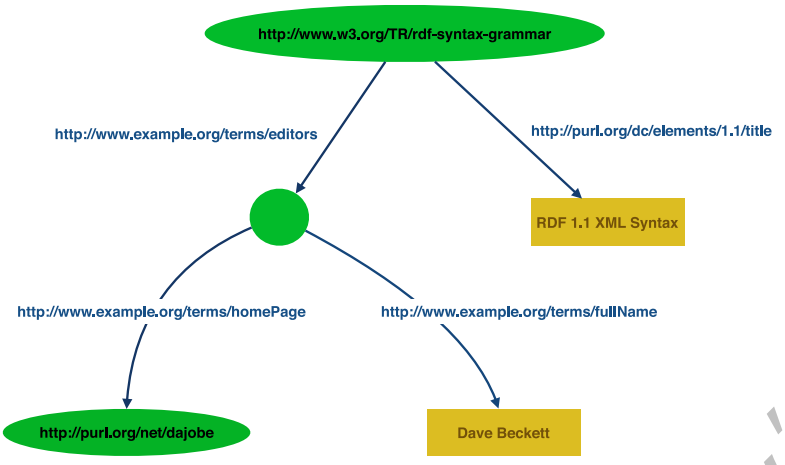
2.2.2 Μορφές Αποθήκευσης Εγγράφων RDF

Ένα έγγραφο RDF αναπαριστά σε αναγνώσιμη μορφή το γράφημα RDF χρησιμοποιώντας ένα συγκεκριμένο συντακτικό. Τέτοια έγγραφα είναι το RDF/XML, Turtle, RDFa, JSON-LD και το TriG (εικ. 2.3). Η χρήση εγγράφων RDF καθιστά δυνατή την ανταλλαγή γραφημάτων RDF μεταξύ συστημάτων.

Ένα συντακτικό RDF μπορεί να προσφέρει πολλούς διαφορετικούς τρόπους κωδικοποίησης του ίδιου γραφήματος RDF, όπως για παράδειγμα, κάνοντας χρήση προθεμάτων χώρων ονομάτων (namespace prefixes), σχετικών IRIs, αναγνωριστικών κενών κόμβων, ή απλώς χρησιμοποιώντας διαφορετική σειρά προτάσεων. Η ιδιότητα αυτή ίσως δυσχεραίνει τη χρήση του εγγράφου RDF, εντούτοις δεν επηρεάζει το νόημα της αποθηκευμένης πληροφορίας [11].

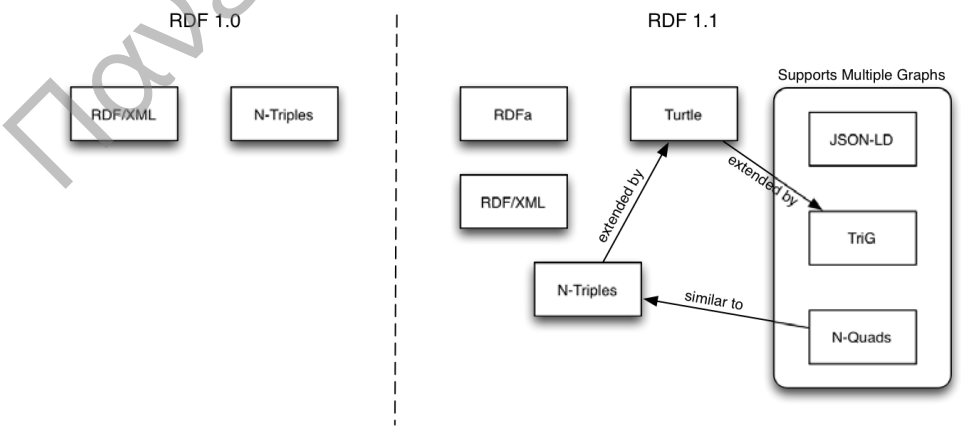
2.2.3 Αποθήκες Πληροφοριών RDF

Καθώς η χρήση πληροφοριών RDF και RDFS αυξάνεται, παρουσιάζεται η ανάγκη χρήσης εξειδικευμένων αποθηκών RDF. Αυτές οι αποθήκες, που ονομάζονται αποθήκες τριπλετών (triple stores) [9], διαφέρουν ως προς τις δυνατότητές τους. Κάποιες εστιάζουν στην παροχή μεθόδων συλλογιστικής (reasoning), όπως το Apache Jena (http://jena.apache.org/), ενώ κάποιες άλλες εστιάζουν στην αποδοτική αποθήκευση και διαχείριση μεγάλης ποσότητας πληροφορίας, όπως το 4store (https://github.com/garlik/4store).




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22—rdf—syntax—ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf—syntax—grammar"
       dc:title="RDF/XML Syntax Specification (Revised)">
    <ex:editor>
      <rdf:Description ex:fullName="Dave Beckett">
  <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
      </rdf:Description>
    </ex:editor>
  </rdf:Description>
</rdf:RDF>
```

Εικόνα 2.2: Παράδειγμα RDF πληροφορίας σε μορφή RDF/XML και αναπαράστασης σε μορφή γραφήματος [12]. Στο γράφημα, οι κόμβοι λεκτικών αποτυπώνονται ως ορθογώνια παραλληλόγραμμα. Διακρίνεται ένας κενός κόμβος που αναπαρίσταται ως ανώνυμος κόμβος.



Εικόνα 2.3: Μορφές αποθήκευσης εγγράφων RDF. Η έκδοση 1.1 του προτύπου RDF ορίζει τρία έγγραφα RDF με υποστήρξη πολλαπλών γραφημάτων RDF.

2.3 Γλώσσα Ερωτημάτων SPARQL

Στην ορολογία του Σημασιολογικού Ιστού, ως "ερώτημα" (query) ορίζεται, γενικά, το σύνολο εκείνων των μέσων και πρωτοκόλλων, που μπορούν να ανακτήσουν πληροφορίες από τον Ιστό Δεδομένων (Web of Data). Για τη διατύπωση των ερωτημάτων απαιτείται η χρήση κάποιας γλώσσας ερωτημάτων, όπως είναι η η SPARQL. Ο οργανισμός W3C έχει οριστικοποιήσει, από το 2013, τη προδιαγραφή 1.1 της γλώσσας [13], όπου, μεταξύ άλλων, ορίζεται το συντακτικό και η σημασιολογία της. Η πρόσβαση στην αποθηκευμένη πληροφορία RDF γίνεται με, κατάλληλα δομημένα, ερωτήματα, που εκτελούνται πάνω σε σε σύνολα από γραφήματα RDF (RDF dataset).

Σε αυτή την ενότητα περιοριζόμαστε στην εισαγωγική περιγραφή της SPARQL με σκοπό την αποτύπωση της σχετικής, με την εργασία, ορολογίας.

2.3.1 Δομή Ερωτήματος SPARQL

Η δομή ενός απλού ερωτήματος SPARQL αποτελείται, με τη σειρά, από τα εξής μέρη:

1. Δηλώσεις προθέματος (prefix declarations), για τη συντομογραφική αναφορά των IRIs. Η λέξη κλειδί PREFIX συσχετίζει μια ετικέτα προθέματος με ένα IRI. Ένα προθεματισμένο όνομα (prefixed name) αποτελείται από την ετικέτα προθέματος και το τοπικό κομμάτι του αντίστοιχου IRI, διαχωρισμένα με άνω κάτω τελεία. Τα προθεματισμένα ονόματα χρησιμοποιούνται για τη συντομογραφική αναπαράσταση μεγάλων IRI.

2. Δήλωση τύπου ερωτήματος και αποτελεσμάτων (result clause), για τη ταυτοποίηση της επιστρεφόμενης πληροφορίας. Ορίζονται διάφοροι τύποι ερωτημάτων, που εξυπηρετούν διαφορετικούς σκοπούς. Μερικοί τύποι είναι οι παρακάτω:

   - Το ερώτημα τύπου SELECT χρησιμεύει στην ανάκτηση αποθηκευμένων δεδομένων από το τελικό σημείο SPARQL, με τη μορφή πίνακα μεταβλητών και τιμών που ικανοποιούν το ερώτημα.
   - Το ερώτημα τύπου CONSTRUCT χρησιμεύει στην ανάκτηση αποθηκευμένων δεδομένων από το τελικό σημείο SPARQL (SPARQL endpoint) και τη κατασκευή γραφήματος RDF από τα αποτελέσματα.
   - Το ερώτημα τύπου ASK χρησιμεύει στη λήψη θετικής ή αρνητικής απάντησης σχετικά με την ύπαρξη αποτελεσμάτων.
   - Το ερώτημα τύπου DESCRIBE επιτρέπει στον εξυπηρετητή να επιστρέψει οποιοδήποτε γράφημα RDF θεωρεί ότι περιγράφει καλύτερα τους συγκεκριμένους πόρους.

3. Ορισμός συνόλων δεδομένων (dataset definition) στα οποία θα γίνει αναζήτηση.

4. Το μοτίβο ερωτήματος (query pattern) που προσδιορίζει τι να αναζητηθεί στα γραφήματα RDF. Κάθε τριπλέτα εντός του μπλοκ, που ορίζει η λέξη WHERE, ονομάζεται μοτίβο επιλογής τριπλετών (triple pattern). Ένα μοτίβο επιλογής τριπλετών μοιάζει με μια τριπλέτα RDF, αλλά διαφέρει ως προς το γεγονός ότι μπορεί να έχει μεταβλητή σε οποιαδήποτε θέση. Η μεταβλητή συντάσσεται με λατινικό ερωτηματικό "?", ή σύμβολο "$", που ακολουθείται από αλφαριθμητικό [13] και μπορεί να ταιριάξει με οποιονδήποτε κόμβο του γράφου RDF. Το σύνολο μοτίβων επιλογής τριπλετών, που συμμετέχουν σε ένα SPARQL ερώτημα αποτελεί το βασικό μοτίβο γράφου (BGP - Basic Graph Pattern).

5. Διαμορφωτές αποτελέσματος (solution modifiers), για τη διαμόρφωση της αποτελεσμάτων (πχ ταξινόμηση με ORDER BY ή ορισμός πλήθους αποτελεσμάτων με LIMIT).

Η SPARQL υποστηρίζει συνάθροιση, υποερωτήματα, άρνηση, δημιουργία τιμών από εκφράσεις, ελέγχους τιμών και περιορισμούς ερωτημάτων με βάση τη προέλευση του γραφήματος RDF. Επιπλέον, δίνει τη δυνατότητα αναζήτησης υποχρεωτικών, ή προαιρετικών BGP, μαζί με τις συζεύξεις και διαζεύξεις τους. Τα αποτελέσματα των ερωτημάτων SPARQL μπορούν να είναι σύνολα αποτελεσμάτων ή γραφήματα RDF.

Το συντακτικό της SPARQL περιλαμβάνει τη λέξη κλειδί "a" η οποία αποτελεί συντομογραφία του URI

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

ή, αλλιώς rdf:type και χρησιμοποιείται σε μοτίβα επιλογής τριπλετών ως κατηγόρημα, για την εύρεση της τάξης του υποκειμένου.

Παρακάτω διακρίνεται η γενική δομή ενός ερωτήματος SPARQL.

```
# 1 — Prefix declarations
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

# 2 — Result clause
SELECT DISTINCT ?name ?email

# 3 — Dataset definition
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>

# 4 — Query pattern
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}

# 5 — Solution modifiers
ORDER BY ?name
LIMIT 5
```

Σύμφωνα με όσα προαναφέραμε, παρατηρώντας το παραπάνω παράδειγμα μπορούμε να κάνουμε τις εξής διαπιστώσεις:

- Η ετικέτα προθέματος είναι το ”foaf” και το IRI είναι το ”http://xmlns.com/foaf/0.1/”.
- Το ”foaf:Person” αποτελεί προθεματισμένο όνομα, που αντιστοιχεί στο IRI ”http://xmlns.com/foaf/0.1/Person”.
- Τα ”?person”, ”?name” και ”?email” είναι μεταβλητές.
- Η λέξη κλειδί ORDER BY ταξινομεί τα αποτελέσματα με βάση την τιμή ?name. Αντίστοιχα, το LIMIT 5, περιορίζει το πλήθος των αποτελεσμάτων σε 5.

Η τελεία στο τέλος καθενός μοτίβου επιλογής τριπλετών συμβολίζει τη λογική πράξη AND μεταξύ αυτού και του επόμενου. Κατά την εκτέλεση του παραπάνω ερωτήματος αντιπαραβάλλονται τα μοτίβα επιλογής τριπλετών με τις αποθηκευμένες τριπλέτες RDF και επιλέγονται μόνο εκείνες οι τριπλέτες που ταυτίζονται με όλα τα μοτίβα επιλογής τριπλετών.

Το μοτίβο ερωτήματος του παραδείγματος μπορεί να μεταφραστεί σε φυσική γλώσσα ως εξής:

> ”Βρες τους κόμβους ?person, που είναι τύπου foaf:Person και οι οποίοι συνδέονται, μέσω foaf:name, με κάποιους κόμβους ?name και, μέσω foaf:mbox, με κάποιους κόμβους ?email”.

2.3.2 Τελικό Σημείο SPARQL

Ένα τελικό σημείο SPARQL είναι μια υπηρεσία που δέχεται ερωτήματα και επιστρέφει απαντήσεις μέσω πρωτοκόλλου HTTP. Υπάρχουν δύο ειδών τελικά σημεία, τα γενικά και τα ειδικά. Τα γενικά απευθύνονται σε οποιαδήποτε δεδομένα RDF είναι προσιτά μέσω Ιστού, ενώ τα ειδικά απευθύνονται σε συγκεκριμένα σύνολα γραφημάτων RDF.

Τα αποτελέσματα των ερωτημάτων SPARQL μπορούν να επιστραφούν και να παρουσιαστούν με διαφορετικές μορφές, μερικές από τις οποίες είναι οι εξής:

- XML: Η προδιαγραφή της SPARQL προσδιορίζει ειδικό λεξιλόγιο XML για την αναπαράσταση των αποτελεσμάτων σε πίνακα.
- JSON: Είναι ιδιαίτερα χρήσιμη σε εφαρμογές Ιστού.
- CSV/TSV: Απλές αναπαραστάσεις κειμένου, χρήσιμες για εισαγωγή δεδομένων σε λογιστικά φύλλα.

- RDF: Ορισμένοι τύποι ερωτημάτων SPARQL (πχ CONSTRUCT) προκαλούν απαντήσεις RDF, οι οποίες μπορούν να μετατραπούν σε οποιαδήποτε από τις υποστηριζόμενες μορφές αποθήκευσης εγγράφων RDF.

2.4 Συνδεδεμένα Δεδομένα (Linked Data)

Ο όρος linked data (διασυνδεδεμένα δεδομένα) αναφέρεται σε μία μέθοδο δημοσιοποίησης δομημένων δεδομένων και των αλληλοσυσχετίσεών τους, με σκοπό την αύξηση της χρησιμότητάς τους. Η μέθοδος αυτή στηρίζεται σε καθιερωμένες τεχνολογίες του Ιστού, όπως HTTP και URI, αλλά αντί να τις χρησιμοποιεί για να εξυπηρετεί ιστοσελίδες για τους ανθρώπινους αναγνώστες, τις επεκτείνει, ώστε να ανταλλάσσουν πληροφορίες με τρόπο που να μπορούν να διαβαστούν αυτόματα από τους υπολογιστές. Αυτό επιτρέπει δεδομένα από διαφορετικές πηγές να συνδέονται και να μπορούν να αναζητηθούν.

Μια μηχανή αναζήτησης που αξιοποιεί την έμφυτη σημασιολογία και αλληλοσυσχέτιση των διασυνδεδεμένων δεδομένων, μπορεί να απαντήσει εύστοχα σε ερωτήσεις που είναι καλά διατυπωμένες σε γλώσσα SPARQL. Μια κοινή μηχανή αναζήτησης, που δεν αξιοποιεί τα διασυνδεδεμένα δεδομένα αλλά, αντίθετα, προσδιορίζει ευρετικά τη σημασιολογία και την αλληλοσυσχέτισή τους, μπορεί να είναι λιγότερο αποτελεσματική.

Στο ακόλουθο παράδειγμα [8][2] αποτυπώνεται αυτή η διαφορά και αναδεικνύεται η χρησιμότητα των διασυνδεδεμένων δεδομένων:

Παράδειγμα Έστω το ερώτημα ”Βρες γονίδια που σχετίζονται με τη μεταγωγή σημάτων και σχετίζονται με πυραμιδικούς νευρώνες;”. Αυτό το ερώτημα ορίζει ένα πεπερασμένο σύνολο σωστών απαντήσεων, το οποίο αποτελείται από τις ονομασίες συγκεκριμένων γονιδίων. Οποιαδήποτε άλλη απάντηση θεωρείται ανακριβής.

- Αν η ερώτηση διατυπωθεί με τις λέξεις κλειδιά ”pyramidal neurons signal transduction”, και αποσταλεί στη μηχανή αναζήτησης Google, επιστρέφονται περίπου 230000 αποτελέσματα, που περιέχουν σχετικές πληροφορίες (δημοσιεύσεις σε επιστημονικά περιοδικά, άρθρα σε εγκυκλοπαίδειες κλπ), αλλά κανένα δεν αποτελεί σωστή απάντηση.

- Αν η ερώτηση διατυπωθεί σε γλώσσα SPARQL (εικ. 2.4) και αποσταλεί σε μηχανή αναζήτησης, που αξιοποιεί συνδεδεμένα δεδομένα, επιστρέφονται 32 αποτελέσματα, καθένα από τα οποία αποτελεί σωστή απάντηση [8][2].

Υπογραμμίζεται ότι οι απαντήσεις δεν βρίσκονται συγκεντρωμένες σε συγκεκριμένο σημείο του Ιστού, αλλά είναι κατανεμημένες σε πολλαπλά σημεία. Το γεγονός ότι είναι διασυνδεδεμένες αποτελεί το κλειδί της αποτελεσματικότητας του ερωτήματος. Επίσης, η αποτελεσματικότητα οφείλεται στην αξιοποίηση της σημασιολογίας των δεδομένων, η οποία είναι έμφυτη σε αυτά, σε αντίθεση με μια κοινή μηχανή αναζήτησης, η οποία προσδιορίζει τη σημασιολογία ευρετικά.

Στο παράδειγμα διαφαίνεται η δυσκολία σύνταξης κατάλληλου ερωτήματος για την κάλυψη των πληροφοριακών αναγκών του χρήστη. Για τη σύνταξη κατάλληλων ερωτήσεων SPARQL απαιτείται η γνώση των περιεχομένων της αποθήκης RDF, που απαιτεί εμπειρία και προσπάθεια από πλευράς χρήστη.

2.5 Επαναδιατύπωση Προβλήματος

Σε αυτή την ενότητα επαναδιατυπώνεται το πρόβλημα, που αντιμετωπίζεται σε αυτή την εργασία και, αναδεικνύεται η αναγκαιότητα σύμπτυξης της πληροφορίας του αρχικού γραφήματος RDF, για την παραγωγή ενός ισοδύναμου αλλά αρκετά περιεκτικότερου γραφήματος. Αυτό το γράφημα ορίζεται ως σύνοψη RDF.

Η σύνοψη περιλαμβάνει τους τύπους (classes) όλου του γραφήματος RDF και τις μεταξύ τους συνδέσεις, όπως αυτές εκφράζονται μέσα από τα κατηγορήματα. Αυτό αποτελεί μια περίληψη των εννοιών που περιέχονται στο RDF γράφημα. Η σύνοψη δίνει τη δυνατότητα στον χρήστη να ξέρει εκ των προτέρων αν το γράφημα RDF μπορεί να ικανοποιήσει τις πληροφοριακές του ανάγκες. Η αλληλεπίδραση του χρήστη με τη σύνοψη γίνεται μέσω εντοπισμού, από τον χρήστη, συγκεκριμένων RDF τύπων (RDF types) της σύνοψης, ή και κατηγορημάτων, με τα οποία συνδέονται αυτοί. Προκειμένου, ο χρήστης, να εντοπίσει τους τύπους

```
1.  prefix go: <http://purl.org/obo/owl/GO#>
2.  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3.  prefix owl: <http://www.w3.org/2002/07/owl#>
4.  prefix mesh: <http://purl.org/commons/record/mesh/>
5.  prefix sc: <http://purl.org/science/owl/sciencecommons/>
6.  prefix ro: <http://www.obofoundry.org/ro/ro.owl#>
7.  SELECT ?gene_name ?process_name
8.  WHERE
9.  { ?pubmed_record ?related mesh:D017966 .
10.   ?article sc:identified_by_pmid ?pubmed_record.
11.   ?gene_record sc:describes_gene_or_gene_product_mentioned_by ?article.
12.   ?protein rdfs:subClassOf ?restriction.
13.   ?restriction owl:onProperty ro:has_function.
14.   ?restriction owl:someValuesFrom ?restriction2.
15.   ?restriction2 owl:onProperty ro:realized as.
16.   ?restriction2 owl:someValuesFrom ?process.
17.   { { ?process ro:part_of go:GO 0007165 }
18.       UNION
19.     { ?process rdfs:subClassOf go:GO 0007165 }}
20.   ?protein rdfs:subClassOf ?parent.
21.   ?parent owl:equivalentClass ?restriction3.
22.   ?restriction3 owl:onProperty sc:is_protein_gene_product_of_dna_described_by.
23.   ?restriction3 owl:hasValue ?gene_record.
24.   ?gene_record rdfs:label ?gene_name.
25.   ?process rdfs:label ?process_name.
26. }
```

Εικόνα 2.4: Παράδειγμα ερωτήματος SPARQL για τη λήψη εύστοχων απαντήσεων [8].

δεδομένων, και τα κατηγορήματα της σύνοψης, προτείνεται, στην εργασία αυτή, μια μεθοδολογία αντιστοίχησης των ανθρωπίνως αναγνωρίσιμων λεκτικών του γραφήματος RDF, σε RDF τύπους και κατηγορήματα της σύνοψης.

Η μεθοδολογία αυτή, όπως θα φανεί στις επόμενες ενότητες, βασίζεται σε ένα κατάλληλα διαμορφωμένο γραφικό περιβάλλον χρήστη (GUI - Graphical User Interface), το οποίο υποδέχεται τις πληροφοριακές ανάγκες του χρήστη εκφρασμένες σε λέξεις-κλειδιά, που αντιστοιχούν σε κόμβους ή ακμές της σύνοψης. Αφού γίνει η αντιστοίχηση των πληροφοριακών αναγκών του χρήστη στους κόμβους της σύνοψης, προτείνεται ένας αλγόριθμος αποδοτικής σύνδεσης των κόμβων αυτών. Τα εναλλακτικά δένδρα που θα προκύψουν μετατρέπονται σε ερωτήματα SPARQL που είναι συμβατά με την αρχική αποθήκη RDF. Συνεπώς, ο χρήστης έχει τη δυνατότητα, μέσω της σύνοψης, να απευθύνει SPARQL ερωτήματα στο αρχικό RDF γράφημα, χωρίς να είναι αναγκασμένος να τα συντάξει. Επίσης, σε περίπτωση όπου δεν υπάρχουν πληροφορίες στην αρχική αποθήκη RDF, η αλληλεπίδραση με τη σύνοψη θα τον ενημερώνει σχετικά.

Αλληλεπίδραση χρήστη με τη σύνοψη RDF Η γραφική διεπαφή χρήστη αποτελείται από γραφικά στοιχεία (πχ inputs, buttons κλπ), τα οποία προσφέρουν τη δυνατότητα επιλογής, από τον χρήστη, εκείνων των λέξεων-κλειδιών που ταιριάζουν καλύτερα στις πληροφοριακές του ανάγκες. Οι λέξεις-κλειδιά βρίσκονται σε κατάλληλα διαμορφωμένη υπηρεσία ευρετηρίου.

3 Σύνοψη RDF

3.1 Βασικές Έννοιες Γραφημάτων

Σε αυτή την ενότητα θα γίνει μια σύντομη, εισαγωγική αναφορά στα γραφήματα (ή γράφους), η οποία θα περιοριστεί σε έννοιες που σχετίζονται άμεσα με το αντικείμενο αυτής της εργασίας. Θα ασχοληθούμε με πεπερασμένα γραφήματα.

Γράφημα Ένα προσανατολισμένο γράφημα είναι ένα διατεταγμένο ζεύγος $G = (V, E)$ αποτελούμενο από το μη κενό σύνολο κόμβων V και το σύνολο ακμών E. Το σύνολο ακμών ορίζεται ως μια διμελής σχέση $E \subseteq V \times V$ επί του συνόλου V, όπου $V \times V = \{(u, v) \,|u, v \in V\}$.

Το γράφημα μπορεί να περιέχει βρόχους αν η σχέση E είναι μια ανακλαστική (αυτοπαθής) σχέση. Αυτό σημαίνει ότι για κάθε $v \in V$, $(v, v) \in E$.

Το γράφημα ονομάζεται μη κατευθυνόμενο αν η σχέση E είναι μια συμμετρική σχέση. Αυτό σημαίνει ότι για κάθε $u, v \in V$, αν $(v, u) \in E$ τότε $(u, v) \in E$. Αντίθετα, αν η σχέση E δεν είναι συμμετρική, το

γράφημα ονομάζεται κατευθυνόμενο, ή διγράφημα.

Οι κορυφές που ανήκουν σε μια ακμή ονομάζονται τελικά σημεία ή τελικές κορυφές της ακμής. Μια ακμή που έχει ίδια τελικά σημεία ονομάζεται βρόχος. Είναι φανερό, ότι μια κορυφή μπορεί να υπάρχει σε ένα γράφημα αλλά να μην ανήκει σε καμία ακμή.

Δύο βασικά χαρακτηριστικά ενός γραφήματος είναι η σειρά του, που ισούται με τον αριθμό των κορυφών $|V|$ και το μέγεθός του, που ισούται με τον αριθμό των ακμών $|E|$.

Ο βαθμός ενός κόμβου είναι ο αριθμός των ακμών που συνδέονται με αυτόν, όπου κάθε βρόχος συνυπολογίζεται δύο φορές.

Πολυγράφημα Πολυγράφημα ή ψευδογράφημα είναι ένα γράφημα το οποίο έχει τουλάχιστον δύο κόμβους που συνδέονται με πολλαπλές ακμές (ονομάζονται και παράλληλες ακμές). Πιο συγκεκριμένα, το σύνολο E είναι ένα πολυσύνολο από μη διατεταγμένα ζεύγη κορυφών, όχι απαραιτήτως διαφορετικά μεταξύ τους.

Ορισμοί Ακολουθεί μια σύντομη αναφορά στις βασικές έννοιες που χρησιμοποιούνται σε αυτή την εργασία [18] .

- Γειτονικοί κόμβοι:
  Δύο κόμβοι, έστω $u, v \in V$, λέγονται γειτονικοί αν και μόνο αν υπάρχει τουλάχιστον μια ακμή $\{u, v\} \in E$ που τους συνδέει. Το σύνολο των γειτονικών κόμβων του v συμβολίζεται με $N(v)$, και ορίζεται ως $N(v) = \{u | \{v, u\} \in E\}$.

- Ομοιότητα γραφημάτων:
  Δύο γραφήματα $G = (V, E)$ και $G' = (V', E')$ είναι όμοια αν[1] $V = V'$ και $E = E'$.

- Ισομορφισμός γραφημάτων:
  Δύο γραφήματα $G = (V, E)$ και $G' = (V', E')$ είναι ισόμορφα[1] αν και μόνο αν υπάρχει αμφιμονοσήμαντη απεικόνιση $f : V \to V'$, με $\{x, y\} \in E \Leftrightarrow \{f(x), f(y)\} \in E'$. Ο ισομορφισμός συμβολίζεται ως $G \simeq G'$.

- Διαδρομή:
  Η διαδρομή[18] που ενώνει τους κόμβους u_i, u_j ενός γραφήματος G συμβολίζεται ως $u_i - u_j$ και ισοδυναμεί με μια ακολουθία της μορφής $(u_i, e_{ik}, u_k, e_{kl}, u_l, ..., u_r, e_{rj}, u_j)$, όπου $e_{st} \in E$ είναι ακμή που ενώνει τους κόμβους u_s και u_t.

- Μήκος διαδρομής:
  Το μήκος διαδρομής ισούται με το πλήθος των ακμών μιας διαδρομής.

- Δρόμος:
  Ως δρόμος ορίζεται μια διαδρομή που έχει μοναδικούς δεσμούς.

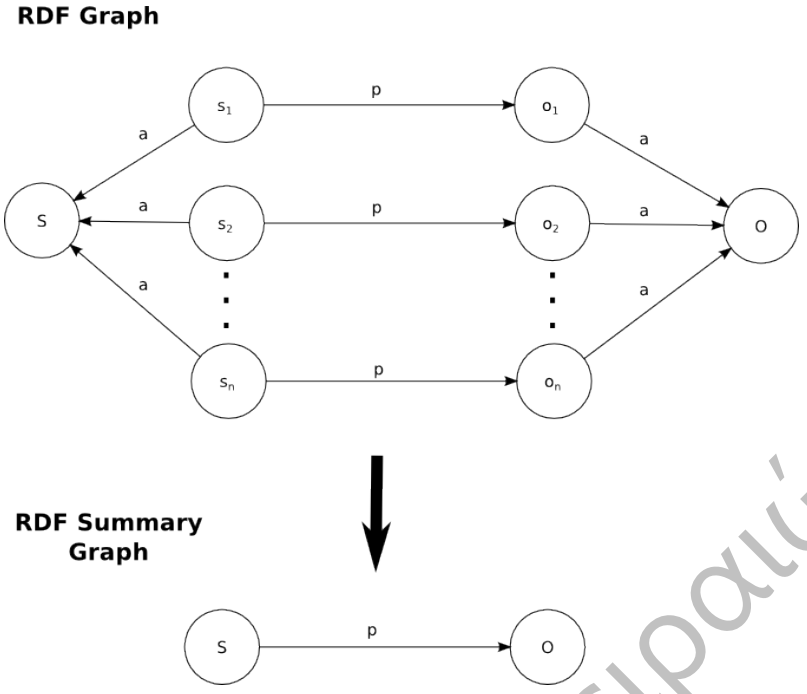
- Κλειστή διαδρομή:
  Μια διαδρομή της οποίας ο πρώτος όρος της ακολουθίας συμπίπτει με τον τελευταίο. Δηλαδή, ο αρχικός κόμβος ισούται με τον τελευταίο.

- Μονοπάτι:
  Ως μονοπάτι ορίζεται μια διαδρομή που έχει έχει μοναδικούς δεσμούς και μοναδικούς κόμβους.

- Συνεκτικότητα γραφήματος:
  Αν για κάθε ζεύγος κόμβων του γραφήματος υπάρχει μονοπάτι που τους ενώνει.

- Κύκλος:
  Κλειστός δρόμος με μήκος που ισούται με το πλήθος των κόμβων του.

- Δένδρο:
  Ένα συνεκτικό, άκυκλο γράφημα. Το πλήθος ακμών $|E|$ και κόμβων ενός δένδρου $T = (E, V)$ δίνεται από τη σχέση
  $$|V| = |E| + 1 \tag{1}$$



Εικόνα 3.1: Διαδικασία αντικατάστασης τριπλετών για τη δημιουργία της σύνοψης RDF. Προσθήκη απλών κόμβων τύπων RDF στη σύνοψη RDF.

3.2 Δημιουργία Σύνοψης RDF

Η σύνοψη RDF αντανακλά τη σημασία των δεδομένων του γραφήματος RDF. Όταν κάθε δεδομένο ανήκει σε μία, ή περισσότερες, τάξεις RDF, η σύνοψη RDF αντιστοιχεί σε ένα γράφημα, οι κόμβοι του οποίου είναι οι τάξεις των δεδομένων. Η ακμή μεταξύ δύο τάξεων (αφετηρία, ακμή, προορισμός) αντιστοιχεί στο κοινό κατηγόρημα όλων των τριπλετών μεταξύ υποκειμένων μιας τάξης (αφετηρία) και αντικειμένων μιας άλλης τάξης (προορισμός).

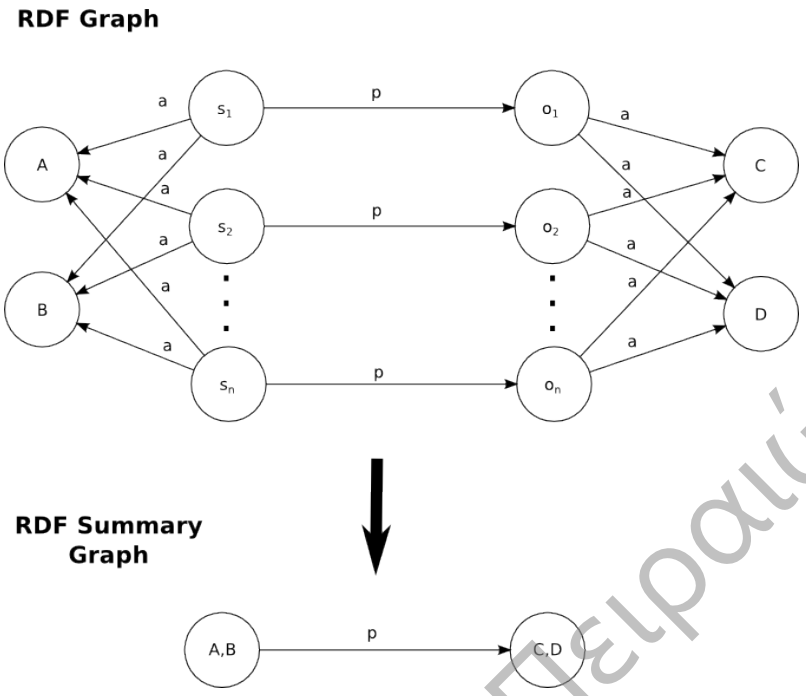
Ένας κόμβος μπορεί να είναι *σύνθετος κόμβος (composite node)* ή *απλός κόμβος (simple node)*. Ένας σύνθετος κόμβος προκύπτει από αντικατάσταση τριπλετών οι οποίες έχουν, στη θέση υποκειμένου ή αντικειμένου, πόρους που ανήκουν στο ίδιο σύνολο τάξεων (εικ. 3.2). Το όνομα του σύνθετου κόμβου περιλαμβάνει τις τάξεις αυτές. Αυτοί οι πόροι δεν χρησιμοποιούνται για τη δημιουργία των επιμέρους απλών κόμβων τάξεων. Ένας απλός κόμβος προκύπτει από αντικατάσταση τριπλετών οι οποίες έχουν στη θέση υποκειμένου ή αντικειμένου, πόρους που ανήκουν στην ίδια τάξη (εικ. 3.1). Το όνομα του απλού κόμβου είναι ίδιο με την κοινή αυτή τάξη. Οι σύνθετοι κόμβοι και οι απλοί κόμβοι αποτελούν τους *κόμβους τύπων RDF (RDF type nodes)* της σύνοψης. Επιπλέον, υπάρχουν και οι *μη τυποποιημένοι κόμβοι (untyped nodes)*. Αυτοί προκύπτουν από αντικατάσταση τριπλετών οι οποίες έχουν, στη θέση υποκειμένου, ή αντικειμένου, πόρους οι οποίοι δεν ανήκουν σε καμία τάξη RDF (εικ. 3.3).

Όπως φαίνεται από τη διαδικασία κατασκευής της σύνοψης RDF, το γράφημα αυτό αποτελεί συμπύκνωση του αρχικού γραφήματος με πολύ μικρότερο μέγεθος. Όπως θα περιγραφεί παρακάτω, η σύνοψη RDF μπορεί να εξυπηρετήσει άμεσα μέρος των δυνατών ερωτημάτων που απευθύνονται στο αρχικό γράφημα RDF. Η δυνατότητα αυτή είναι σημαντική για το κίνημα των συνδεδεμένων δεδομένων, όπου οι χρήστες απευθύνουν ερωτήματα πραγματικού χρόνου σε γραφήματα RDF μεγάλου μεγέθους και περιμένουν την ικανοποίηση αυτών σε εύλογο χρονικό διάστημα.

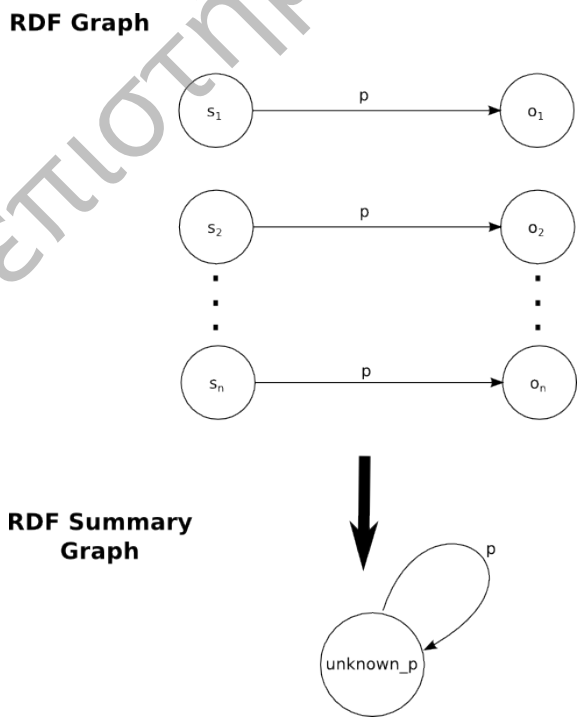
3.2.1 Περιγραφή

Έστω το γράφημα RDF $G_{RDF} = (V_{RDF}, E_{RDF}, P_{RDF}, l)$, όπου P_{RDF} το σύνολο κατηγορημάτων, $E_{RDF} \subseteq V_{RDF} \times P_{RDF} \times V_{RDF}$ μια τριμελής σχέση που ορίζει το σύνολο τριπλετών του γραφήματος, όπου

$$V_{RDF} \times P_{RDF} \times V_{RDF} = \{(s, p, o) \,|s, o \in V_{RDF} \wedge p \in P_{RDF}\}$$



Εικόνα 3.2: Προσθήκη σύνθετων κόμβων τύπων RDF στη σύνοψη RDF.



Εικόνα 3.3: Προσθήκη μη τυποποιημένων κόμβων στη σύνοψη RDF.

Η σημασιολογία του γραφήματος RDF ορίζει μια συνάρτηση επί, την $l : V_{RDF} \to \{0, 1\}$ που απεικονίζει κάθε $x \in V_{RDF}$ στον αριθμό 0 αν $x \in L_{RDF}$ ή στον αριθμό 1 αν $x \notin L_{RDF}$, δηλαδή

$$l(x) = \begin{cases} 1 & , x \in L_{RDF} \\ 0 & , x \notin L_{RDF} \end{cases}$$

Η συνάρτηση αυτή αντιστοιχεί στη συνάρτηση isLiteral() της SPARQL και είναι απαραίτητη για τον διαχωρισμό των λεκτικών από τους υπόλοιπους κόμβους του γραφήματος RDF. Αξίζει να σημειωθεί ότι ορίζονται κι άλλες παρόμοιες συναρτήσεις, οι οποίες όμως δεν αναφέρονται εδώ επειδή δεν χρησιμεύουν στη δημιουργία της σύνοψης. Μπορούμε τώρα να ορίσουμε μερικά χρήσιμα σύνολα.

Διαμέριση Ακμών Γραφήματος RDF Ορίζουμε τη διαμέριση $\{E_1, E_2, E_3\}$ του συνόλου τριπλετών E_{RDF}, θεωρώντας τα μη κενά υποσύνολα $E_1, E_2, E_3 \subseteq E_{RDF}$ ως εξής

- $E_1 = \{(x, p, y) \,|\, (x, p, y) \in E_{RDF} \wedge p =$ 'rdf:type' $\wedge \neg l(y) = 1\}$ είναι το σύνολο των τριπλετών που αναθέτουν τύπους RDF σε URIs (*τριπλέτες τύπων*)
- $E_2 = \{(x, p, y) \,|\, (x, p, y) \in E_{RDF} \wedge \neg p =$ 'rdf:type' $\wedge \neg l(y) = 1\}$ είναι το σύνολο των τριπλετών που ορίζουν τις συνδέσεις μεταξύ URIs και δεν ορίζουν τύπους RDF (*τριπλέτες συνδέσεων*)
- $E_3 = \{(x, p, y) \,|\, (x, p, y) \in E_{RDF} \wedge \neg p =$ 'rdf:type' $\wedge\, l(y) = 1\}$ είναι το σύνολο των τριπλετών που αναθέτουν λεκτικά σε URIs (*τριπλέτες λεκτικών*)

Συνεπώς, αυτά τα σύνολα απαρτίζουν το σύνολο τριπλετών του γραφήματος RDF και είναι ξένα μεταξύ τους, δηλαδή

$$E_1 \bigcap E_2 = E_1 \bigcap E_3 = E_2 \bigcap E_3 = \emptyset$$

$$E_1 \bigcup E_2 \bigcup E_3 = E_{RDF}$$

Διαμέριση Κόμβων Γραφήματος RDF Ορίζουμε τη διαμέριση $K_1 = \{L_{RDF}, C_{RDF}, U_{RDF}\}$ του συνόλου κόμβων V_{RDF}, θεωρώντας τα μη κενά υποσύνολα $L_{RDF}, C_{RDF}, U_{RDF} \subseteq V_{RDF}$ ως εξής

- το *σύνολο των λεκτικών,* $L_{RDF} = \{y| \,(x, p, y) \in E_3\}$, περιέχει τα λεκτικά και τους τύπους τους
- το *σύνολο των τύπων RDF,* $C_{RDF}\, \{y| \,(x, p, y) \in E_1\}$, περιέχει τους τύπους RDF
- το *σύνολο των URI,* $U_{RDF} = \{x| \,(x, p, y) \in E_3 \vee (x, p, y) \in E_2 \vee (y, p, x) \in E_2\}$, περιέχει κόμβους που δεν είναι ούτε λεκτικά ούτε τύποι RDF. Τα στοιχεία αυτού του συνόλου θα τα λέμε URI, παρόλο που μπορεί να είναι και κενοί κόμβοι και παρόλο που και οι τύποι RDF είναι URI. Μπορούμε να διαμερίσουμε αυτό το σύνολο σε δύο άλλα σύνολα, U_1 και U_0.
  - το σύνολο των *τυποποιημένων (typed) URI*

$$U_1 = \{x| \,(x, p, t) \in E_1\} \tag{2}$$

  - το σύνολο των *μη τυποποιημένων (untyped) URI*

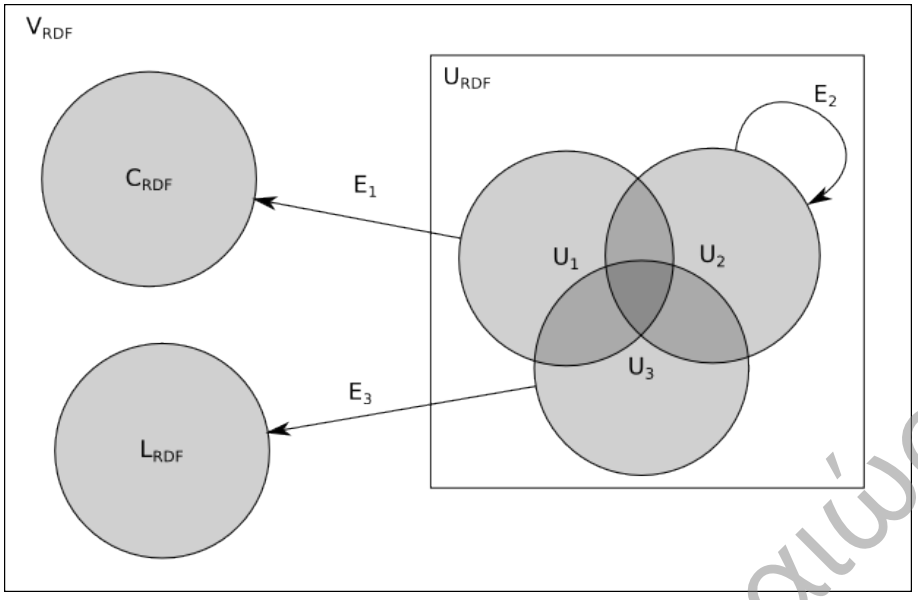
$$U_0 = \{x| \,(x, p, y) \notin E_1 \wedge [(x, p, y) \in E_3 \vee (x, p, y) \in E_2 \vee (y, p, x) \in E_2]\} \tag{3}$$

Θεωρούμε ότι το γράφημα RDF περιέχει οπωσδήποτε ένα στοιχείο από κάθε σύνολο. Αυτά τα σύνολα, που είναι μέλη της διαμέρισης $K = \{L_{RDF}, C_{RDF}, U_1, U_0\}$, απαρτίζουν το σύνολο κόμβων του γραφήματος RDF δηλαδή

$$L_{RDF} \bigcup U_{RDF} \bigcup C_{RDF} \bigcup U_1 \bigcup U_0 = V_{RDF}$$

και είναι ξένα μεταξύ τους, δηλαδή για κάθε $X, Y \in K$ με $X \neq Y$, τότε $X \bigcap Y = \emptyset$.

Μπορούμε να ορίσουμε τα εξής σύνολα:



Εικόνα 3.4: Το σύνολο κόμβων του γραφήματος RDF, V_{RDF} διαμερίζεται στο σύνολο τάξεων RDF, C_{RDF}, στο σύνολο λεκτικών, L_{RDF} και στο σύνολο URI U_{RDF}. Το τελευταίο διαμερίζεται στο σύνολο τυποποιημένων URI U_1 και στο σύνολο μη τυποποιημένων URI $U_0 = U_2 \bigcup U_3 - U_1$, όπου το σύνολο U_2 περιέχει τα διασυνδεδεμένα URI και το σύνολο U_3 περιέχει τα URI που συνδέονται με λεκτικά. Τα βέλη αναπαριστούν τα σύνολα τριπλετών RDF από τα οποία προκύπτουν τα αντίστοιχα σύνολα κόμβων. Τα σκιασμένα τμήματα σημαίνουν ύπαρξη μελών, ενώ τα μη σκιασμένα σημαίνουν ανυπαρξία μελών. Τα τετράγωνα χρησιμεύουν μόνο για την ομαδοποίηση των υποσυνόλων και εκφράζουν την ένωση αυτών.

- το σύνολο των *διασυνδεδεμένων (connected)* URI,

$$U_2 = \{x | (x, p, y) \in E_2 \vee (y, p, x) \in E_2\} \tag{4}$$

- το σύνολο των *URI που αντιστοιχούν σε λεκτικό (labeled URI)*,

$$U_3 = \{x | (x, p, y) \in E_3\} \tag{5}$$

Σύνοψη RDF Ορίζουμε τη σύνοψη RDF ως γράφημα $G = (V_S, E_S, P_S, W_S)$, όπου

- V_S οι κόμβοι της σύνοψης. Θεωρούμε διαμέριση $\{V_{SC}, V_{SU}\}$ του συνόλου V_S, τέτοια ώστε
  - V_{SC} το σύνολο *κόμβων τύπων RDF,* που αποτελείται από *απλούς* και *σύνθετους* κόμβους, όπως αναφέρθηκε στην εισαγωγή αυτής της ενότητας
  - V_{SU} το σύνολο *μη τυποποιημένων κόμβων*, όπως αναφέρθηκε στην εισαγωγή αυτής της ενότητας

  Ισχύει $V_{SC} \bigcap V_{SU} = \emptyset$ και $V_{SC} \bigcup V_{SU} = V_S$.
- $E_S \subseteq V_S \times P_S \times V_S$ μια τριμελής σχέση όπου $V_S \times P_S \times V_S = \{(v, p, u) | v, u \in V_S \wedge p \in P_S\}$. Πρόκειται για το σύνολο *τριπλετών σύνδεσης κόμβων τύπων RDF*, δηλαδή το σύνολο ακμών της σύνοψης που αντικαθιστά το σύνολο τριπλετών συνδέσεων E_2 του γραφήματος RDF, μέσω της συνάρτησης R_C.
- $P_S = P_{RDF} - \{$'rdf:type'$\}$ το σύνολο των *κατηγορημάτων* της σύνοψης. Σημειώνεται ότι το κατηγόρημα 'rdf:type' δεν ενώνει κόμβους της σύνοψης.
- $W_S : E_S \longrightarrow (0, 1)$ η συνάρτηση βάρους ακμής, η οποία απεικονίζει κάθε ακμή σε ένα φυσικό αριθμό που ονομάζεται βάρος ακμής και ο οποίος είναι μεγαλύτερος από το μηδέν και μικρότερος από τη μονάδα (βλ. 3.2.3).

3.2.2 Αντικατάσταση Τριπλετών

Έχοντας ορίσει κάποια βασικά σύνολα, σε αυτή την ενότητα θα περιγράψουμε τον τρόπο που σχηματίζεται η σύνοψη. Παρακάτω, ορίζουμε τον αλγόριθμο αντικατάστασης των τριπλετών.

Η *συνάρτηση αντικατάστασης τριπλετών συνδέσεων* $R_C : E_2 \longrightarrow E_S$ με $R_C \subseteq E_2 \times E_S$ είναι μια συνάρτηση επί και απεικονίζει τριπλέτες συνδέσεων του γραφήματος RDF σε τριπλέτες σύνδεσης κόμβων τύπων RDF της σύνοψης.

Χρησιμοποιείται η *συνάρτηση εύρεσης τύπων* $findtypes : U_1 \bigcap U_2 \rightarrow \mathcal{P}(C_{RDF})$ που αποτελεί τη συνάρτηση απεικόνισης καθενός τυποποιημένου και διασυνδεδεμένου URI σε ένα μη κενό σύνολο από τάξεις RDF[1]. Η συνάρτηση αυτή καθορίζεται ως εξής:

$$findtypes(s) = \{t | (u, p, t) \in E_1 \wedge u = s\}$$

Χρησιμοποιείται η συνάρτηση $getunknownname : P \rightarrow V_{SU}$ που είναι η συνάρτηση απεικόνισης κατηγορήματος σε μη τυποποιημένο κόμβο. Κάθε μη τυποποιημένο και διασυνδεδεμένο URI $u \in (U_2 - U_1 \bigcap U_2)$, που συμμετέχει, ως υποκείμενο ή αντικείμενο, ή και τα δύο, σε διασυνδεδεμένες τριπλέτες RDF (στοιχεία του συνόλου E_2) με κατηγόρημα $p \in P$, κατά τη δημιουργία της σύνοψης RDF μπαίνει ως μέλος του μη τυποποιημένου κόμβου, το όνομα του οποίου ορίζεται από τη συνάρτηση $getunknownname(p)$. Με άλλα λόγια, κάθε μη τυποποιημένο URI μπαίνει στον μη τυποποιημένο κόμβο με όνομα που ορίζεται από το κατηγόρημα του URI.

Ο αλγόριθμος αντικατάστασης τριπλετών RDF υπολογίζει όλα τα σύνολα της σύνοψης RDF, εκτός από το σύνολο βαρών W_S. Αυτό θα προσδιοριστεί με βάση τη συνάρτηση R_C στην επόμενη υποενότητα.

Αλγόριθμος 3.1 Αλγόριθμος αντικατάστασης τριπλετών RDF. Η έξοδος αποτελείται από τις ακμές και τους κόμβους της σύνοψης, καθώς και από το σύνολο που ορίζεται από τη σχέση R_C. Το σύνολο $U_1 \bigcap U_2$ είναι το σύνολο των *τυποποιημένων και διασυνδεδεμένων URI*.

είσοδος: Τριπλέτες συνδέσεων E_2 και τριπλέτες τάξεων E_1 του γραφήματος RDF.
έξοδος: Η συνάρτηση (σύνολο) αντικατάστασης τριπλετών συνδέσεων R_C, το σύνολο τριπλετών συνδέσεων E_S, το σύνολο κόμβων V_S της σύνοψης RDF, το σύνολο κατηγορημάτων P_S της σύνοψης RDF.
$(R_C, E_S, V_S, V_{SC}, V_{SU}, P_S) \leftarrow (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing)$ ▷ Αρχικοποίηση κενών συνόλων.
for $(s, p, o) \in E_2$ **do** ▷ Για κάθε τριπλέτα του συνόλου συνδεδεμένων τριπλετών.
  $P_S \leftarrow P_S \bigcup \{p\}$ ▷ Βάλε το p στο σύνολο κατηγορημάτων της σύνοψης.
  if $s \in U_1 \bigcap U_2$ **then** ▷ Αν το υποκείμενο ανήκει στο σύνολο $U_1 \bigcap U_2$.
    $s_t \leftarrow findtypes(s)$ ▷ Βρες το σύνολο τύπων RDF στους οποίους ανήκει το υποκείμενο.
    $V_{SC} \leftarrow V_{SC} \bigcup \{s_t\}$ ▷ Βάλε το σύνολο τύπων RDF στη σύνοψη, ως κόμβο.
  else
    $s_t = \{getunknownname(p)\}$ ▷ Αντιστοίχισε το p σε όνομα μη τυποποιημένου κόμβου.
    $V_{SU} \leftarrow V_{SU} \bigcup \{s_t\}$ ▷ Βάλε τον μη τυποποιημένο κόμβο στη σύνοψη.
  if $o \in U_1 \bigcap U_2$ **then** ▷ Αν το αντικείμενο ανήκει στο σύνολο $U_1 \bigcap U_2$.
    $o_t \leftarrow findtypes(o)$ ▷ Βρες το σύνολο τύπων RDF στους οποίους ανήκει το αντικείμενο.
    $V_{SC} \leftarrow V_{SC} \bigcup \{o_t\}$ ▷ Βάλε το σύνολο τύπων RDF στη σύνοψη, ως κόμβο.
  else
    $o_t = \{getunknownname(p)\}$ ▷ Αντιστοίχισε το p σε όνομα μη τυποποιημένου κόμβου.
    $V_{SU} \leftarrow V_{SU} \bigcup \{o_t\}$ ▷ Βάλε τον μη τυποποιημένο κόμβο στη σύνοψη.
  $e \leftarrow (s_t, p, o_t)$ ▷ Όρισε την ακμή της σύνοψης.
  $E_S \leftarrow E_S \bigcup \{e\}$ ▷ Βάλε την ακμή στη σύνοψη.
  $r \leftarrow ((s, p, o), e)$ ▷ Όρισε την αντιστοίχιση τριπλέτας RDF σε ακμή της σύνοψης.
  $R_C \leftarrow R_C \bigcup \{r\}$ ▷ Βάλε την αντιστοίχιση στο σύνολο αντιστοιχίσεων τριπλετών RDF σε ακμές σύνοψης.
$V_S \leftarrow V_{SC} \bigcup V_{SU}$ ▷ Η ένωση αυτών των συνόλων αποτελεί το σύνολο κόμβων της σύνοψης RDF.
return R_C,E_S,V_S,P_S

---

[1] Ας σημειωθεί ότι το σύμβολο $\mathcal{P}$ ορίζει δυναμοσύνολο.

3.2.3 Ορισμός Βάρους Ακμής

Ορίζουμε συνάρτηση βάρους ακμής $W_S : E_S \longrightarrow (0,1)$ μια συνάρτηση η οποία απεικονίζει κάθε ακμή σε ένα φυσικό αριθμό που ονομάζεται βάρος ακμής και ο οποίος είναι μεγαλύτερος από το μηδέν και μικρότερος από τη μονάδα. Η συνάρτηση έχει επιλεχθεί έτσι ώστε να εκφράζει και τα δύο κριτήρια επιλογής δένδρων (βλ. 3.4).

Όπως προαναφέρθηκε, μια ακμή αντικαθιστά ένα ορισμένο πλήθος τριπλετών με κοινό κατηγόρημα. Επειδή το βάρος ακμής εξαρτάται από το πλήθος αυτό, είναι απαραίτητος ο ορισμός της συνάρτησης $N_C : E_S \to \mathbb{N}^+$ απεικόνισης κάθε ακμής της σύνοψης σε πλήθος αντικατεστημένων τριπλετών. Η συνάρτηση μπορεί να καθοριστεί ως εξής:

$$N_C(x) = |\{t|\,(t,e) \in R_C \wedge e = x\}|$$

Τώρα μπορούμε να ορίσουμε το *βάρος ακμής της σύνοψης RDF* $W_S(e)$, για κάθε ακμή $e \in E_S$, η οποία αντιστοιχεί σε πλήθος $N_C(e)$ αντικατεστημένων τριπλετών, ως εξής:

$$W_S(e) = \frac{1}{N+1}\left(1 + \frac{1 - \frac{N_C(e)}{N_R}}{N-1}\right) \tag{6}$$

όπου $N_R = |E_2| = |R_C|$ το ολικό πλήθος τριπλετών που αντικαταστάθηκαν από ακμές της σύνοψης RDF και $N = |E_S|$ το πλήθος ακμών της σύνοψης RDF. Η συνάρτηση αποτελεί γραμμικό συνδυασμό της μοναδιαίας συνιστώσας μήκους ακμής και του βάρους τριπλετών της. Αυτό μπορεί να φανεί αν η συνάρτηση γραφεί ως εξής

$$W_S(e) = \frac{1}{N+1}(1 + w_R(e)) \tag{7}$$

όπου

$$w_R(e) = \frac{1 - \frac{N_C(e)}{N_R}}{N-1} \tag{8}$$

είναι το *βάρος τριπλετών ακμής* της σύνοψης RDF, που εξαρτάται από τις αντικαταστάσεις των τριπλετών RDF γραφήματος.

Πρόταση 1. *Η συνάρτηση βάρους τριπλετών ακμής (εξ. 8) είναι κανονικοποιημένη στη μονάδα. Δηλαδή ισχύει η συνθήκη κανονικοποίησης*

$$\sum_{e \in E_S} w_R(e) = 1 \tag{9}$$

Απόδειξη. Αθροίζοντας τα βάρη τριπλετών όλων των ακμών της σύνοψης έχουμε

$$\begin{aligned}
\sum_{e \in E_S} w_R(e) &= \sum_{e \in E_S} \frac{1 - \frac{N_C(e)}{N_R}}{N-1} \\
&= \frac{1}{N-1}\left(\sum_{e \in E_S} 1 - \frac{1}{N_R}\sum_{e \in E_S} N_C(e)\right) \\
&= \frac{1}{N-1}\left(N - \frac{1}{N_R}N_R\right) \\
&= 1
\end{aligned}$$

□

Πρόταση 2. *Η συνάρτηση βάρους ακμής (εξ. 6) είναι κανονικοποιημένη στη μονάδα. Δηλαδή ισχύει η συνθήκη κανονικοποίησης*

$$\sum_{e \in E_S} W_S(e) = 1$$

Απόδειξη. Αθροίζοντας τα βάρη όλων των ακμών της σύνοψης και χρησιμοποιώντας την εξ. 9 έχουμε

$$\begin{aligned}\sum_{e\in E_S} W_S(e) &= \sum_{e\in E_S}\left[\frac{1}{N+1}(1+w_R(e))\right]\\ &= \frac{1}{N+1}\sum_{e\in E_S}(1+w_R(e))\\ &= \frac{1}{N+1}\left[\sum_{e\in E_S}1+\sum_{e\in E_S}w_R(e)\right]\\ &= \frac{1}{N+1}(N+1)\\ &= 1\end{aligned}$$

□

Το βάρος επιλέχθηκε έτσι ώστε να είναι ανάλογο του πλήθους τριπλετών που αντιστοιχούν στην ακμή. Η σταθερά αναλογίας επιλέχτηκε να είναι αρνητική, ώστε το βάρος να μειώνεται με την αύξηση των τριπλετών. Αυτό έχει ως αποτέλεσμα, οποιοσδήποτε αλγόριθμος αναζήτησης συντομότερων μονοπατιών, να προτιμά μονοπάτια με ακμές μικρότερου βάρους, οι οποίες αντιστοιχούν σε μεγαλύτερο πλήθος τριπλετών.

Βάρος Δένδρου Ορίζουμε συνάρτηση βάρους δένδρου $B : M \longrightarrow (0,1)$ η οποία απεικονίζει κάθε δένδρο του συνόλου M των δυνατών υποδένδρων του γραφήματος RDF, σε ένα φυσικό αριθμό που ονομάζεται βάρος δένδρου. Η συνάρτηση καθορίζεται ως ακολούθως:

$$B(T) = \sum_{e\in E} W_S(e) \tag{10}$$

όπου E το σύνολο ακμών του δένδρου T.

Πρόταση 3. *Έστω δύο δένδρα $T_i = (V_i, E_i)$ και $T_j = (V_j, E_j)$ τέτοια ώστε $V_i, V_j \subseteq V_S$ και $E_i, E_j \subset E_S$. Αν $0 < |E_i| < |E_j|$ τότε $B(T_i) < B(T_j)$. Με άλλα λόγια, αν το μέγεθος ενός μη τετριμμένου υποδένδρου της σύνοψης, το οποίο δεν είναι ίδιο με τη σύνοψη RDF, είναι μικρότερο από το μέγεθος ενός άλλου μη τετριμμένου υποδένδρου, τότε το ίδιο ισχύει και για τα αντίστοιχα βάρη τους. Ως μη τετριμμένο υποδένδρο εννοούμε την ίδια τη σύνοψη.*

Απόδειξη. Από την εξ. 10 έχουμε

$$\begin{aligned}B(T_i) &= \sum_{e\in E_i}\left[\frac{1}{N+1}(1+w_R(e))\right]\\ &= \frac{1}{N+1}\left(\sum_{e\in E_i}1+\sum_{e\in E_i}w_R(e)\right)\\ &= \frac{1}{N+1}(|E_i|+a)\end{aligned} \tag{11}$$

όπου $a = \sum_{e\in E_i} w_R(e)$. Αντίστοιχα για το βάρος του άλλου υποδένδρου βρίσκουμε

$$B(T_j) = \frac{1}{N+1}(|E_j|+b) \tag{12}$$

όπου $b = \sum_{e\in E_j} w_R(e)$. Αφαιρώντας τις σχέσεις 11,12, έχουμε:

$$B(T_j) - B(T_i) = \frac{1}{N+1}[(|E_j| - |E_i|) + (b-a)] = \frac{1}{N+1}(k+l) \tag{13}$$

όπου $k = |E_j| - |E_i|$ και $l = b - a$. Ισχύει $k \geq 1$ επειδή $|E_j| > |E_i|$ και είναι διαφορά θετικών φυσικών αριθμών. Επίσης, ισχύει $0 < a < 1$ και $0 < b < 1$ επειδή είναι κανονικοποιημένα αθροίσματα θετικών

φυσικών αριθμών πάνω σε υποσύνολα του συνόλου κανονικοποίησης $E_i, E_j \subset E_S$. Από το σύστημα των δύο τελευταίων ανισώσεων προσδιορίζουμε το $b-a$ ως εξής

$$\left.\begin{array}{rrr} 0< & a< & 1 \\ 0< & b< & 1 \end{array}\right\} \Leftrightarrow \left.\begin{array}{rrr} 0> & -a> & -1 \\ 0< & b< & 1 \end{array}\right\} \Leftrightarrow \left.\begin{array}{rrr} -1< & -a< & 0 \\ 0< & b< & 1 \end{array}\right\}$$

Δηλαδή

$$-1 < \quad b-a < \quad 1 \tag{14}$$

Όμως $l = b-a$ και $k \geq 1$ οπότε έχουμε

$$\left.\begin{array}{rrr} -1< & l< & 1 \\ 1\leq & k & \end{array}\right\} \Leftrightarrow 0 < \quad k+l$$

Πολλαπλασιάζοντας την τελευταία ανίσωση με $N+1>0$, βρίσκουμε ότι $\frac{k+l}{N+1} > 0$. Αντικαθιστώντας το αποτέλεσμα στη σχέση 13 βρίσκουμε

$$B\left(T_j\right) - B\left(T_i\right) > 0$$

και τελικά

$$B\left(T_j\right) > B\left(T_i\right)$$

□

Πρόταση 4. *Το αντίστροφο της τελευταίας πρότασης δεν ισχύει. Αν τα βάρη έχουν σχέση $<$ τότε τα αντίστοιχα μεγέθη έχουν σχέση $\leq$.*

Απόδειξη. Έστω $B\left(T_j\right) < B\left(T_i\right)$. Τότε $B\left(T_j\right) - B\left(T_i\right) < 0$ και από τη σχέση 13 έχουμε

$$k+l<0$$

Όμως ισχύει η σχέση (βλ. εξ. 14) $-1 < \quad l < \quad 1$. Λύνουμε το σύστημα ανισώσεων

$$\left.\begin{array}{rrr} k+l< & 0 & \\ -1< & l< & 1 \end{array}\right\} \Leftrightarrow \left.\begin{array}{rrr} k< & -l & \\ -1< & l< & 1 \end{array}\right\} \Leftrightarrow k<1$$

Κι επειδή $k \in \mathbb{N}$, η τελευταία σχέση γίνεται $k \leq 0$. Αντικαθιστώντας το $k = |E_j| - |E_i|$ στην προηγούμενη σχέση έχουμε

$$|E_j| \leq |E_i|$$

□

Πρόταση 5. *Αν τα βάρη είναι ίσα, τότε τα μεγέθη είναι ίσα.*

Απόδειξη. Εξισώνοντας τις σχέσεις 11,12, έχουμε:

$$\begin{array}{rcl} B\left(T_j\right) & = & B\left(T_i\right) \\ |E_j| + b & = & |E_i| + a \\ |E_i| - |E_j| & = & b-a \end{array}$$

Από τη σχέση 14 έχουμε

$$-1 < \quad |E_i| - |E_j| < \quad 1$$

Η διαφορά των μεγεθών είναι διαφορά φυσικών αριθμών, συνεπώς το αποτέλεσμα είναι φυσικός αριθμός. Ο μόνος φυσικός αριθμός ανάμεσα στο -1 και το 1 είναι το 0, οπότε:

$$|E_j| = |E_i|$$

□

3.3 Επιλογή Κόμβων

Η αλληλεπίδραση του χρήστη με τη σύνοψη γίνεται μέσω εντοπισμού, από τον χρήστη, συγκεκριμένων τύπων δεδομένων της σύνοψης, ή και κατηγορημάτων, με τα οποία συνδέονται αυτοί. Προκειμένου, ο χρήστης, να εντοπίσει τους τύπους δεδομένων και τα κατηγορήματα της σύνοψης, προτείνεται, στην εργασία αυτή, μια μεθοδολογία αντιστοίχησης ανθρωπίνως αναγνωρίσιμων λεκτικών του γραφήματος RDF, σε τύπους δεδομένων και κατηγορήματα της σύνοψης.

Η μεθοδολογία αυτή, όπως θα φανεί στις επόμενες ενότητες, βασίζεται σε ένα κατάλληλα διαμορφωμένο γραφικό περιβάλλον χρήστη , το οποίο υποδέχεται τις πληροφοριακές ανάγκες του χρήστη εκφρασμένες σε λέξεις κλειδιά, που αντιστοιχούν σε κόμβους της σύνοψης.

3.3.1 Μητρώο Δεδομένων

Το *μητρώο δεδομένων (Data Registry)* περιέχει τις αντιστοιχίσεις λέξεων-κλειδιών σε κόμβους της σύνοψης. Ορίζουμε τη συνάρτηση suffix (u) η οποία αντιστοιχεί ένα URI u στην κατάληξή του. Ως κατάληξη ενός URI ορίζεται το μη κενό λεκτικό μετά την τελευταία κάθετο '/'. Για παράδειγμα, η κατάληξη του URI "http://test.uri/some/suffix" ή του "http://test.uri/some/suffix/" είναι "suffix". Ορίζονται τέσσερις απεικονίσεις (συναρτήσεις):

- το σύνολο D_t διατεταγμένων ζευγών $(c, \text{suffix}\,(c)) \in D_t$, που αποτελεί μονοσήμαντη απεικόνιση τάξης RDF $c \in C_{RDF}$ σε κατάληξη του URI της.
- το σύνολο D_p διατεταγμένων ζευγών $(p, \text{suffix}\,(p)) \in D_p$, που αποτελεί μονοσήμαντη απεικόνιση κατηγορήματος της σύνοψης $p \in P_S$ σε κατάληξη του URI της.
- το σύνολο D_v που αποτελεί μονοσήμαντη απεικόνιση καθενός λεκτικού RDF, σε ένα κόμβο της σύνοψης.
- το σύνολο D_{vs} που αποτελεί μονοσήμαντη απεικόνιση καθενός λεκτικού RDF, το οποίο έχει την ιδιότητα να συνδέεται με URIs μέσω του κατηγορήματος 'rdfs:label', σε ένα κόμβο της σύνοψης.

3.4 Σύνδεση Κόμβων

Σύμφωνα με την επαναδιατύπωση του προβλήματος, ένας από τους στόχους της εργασίας είναι η αποδοτική σύνδεση κόμβων της σύνοψης RDF, όπως αυτοί έχουν επιλεγεί από τον χρήστη μέσα από το αντίστοιχο γραφικό περιβάλλον. Η σύνδεση ενός υποσυνόλου κόμβων ενός γραφήματος ορίζει ένα συνεκτικό υπογράφημα. Στην εργασία ενδιαφερόμαστε για άκυκλα, συνεκτικά υπογραφήματα, δηλαδή δένδρα. Γενικά, σε ένα γράφημα, υπάρχει πεπερασμένο πλήθος δένδρων. Για τις ανάγκες της εργασίας δεν αρκεί η επιλογή οποιουδήποτε τέτοιου δένδρου. Απαιτείται η επιλογή συγκεκριμένων δένδρων, με βάση κάποια κριτήρια. Αυτά είναι το πλήθος των ακμών του δένδρου και το αθροιστικό βάρος τριπλετών των ακμών του δένδρου. Μάλιστα το πρώτο κριτήριο είναι πιο σημαντικό από το δεύτερο, συνεπώς η επιλογή των δένδρων θα πρέπει να γίνεται πρώτα με βάση το πρώτο και μετά, σε περίπτωση ισοβαθμίας, με το δεύτερο. Πιο συγκεκριμένα:

- Πλήθος ακμών (μέγεθος δένδρου)
  Το πλήθος των μοτίβων επιλογής τριπλετών του παραγόμενου ερωτήματος SPARQL είναι ανάλογο του μεγέθους του δένδρου προέλευσης (15). Λαμβάνοντας υπόψη το γεγονός ότι η πολυπλοκότητα επεξεργασίας ενός ερωτήματος SPARQL είναι ανάλογη του πλήθους μοτίβων επιλογής τριπλετών, συμπεραίνουμε ότι η πολυπλοκότητα είναι ανάλογη του μεγέθους του δένδρου. Συνεπώς, η απαίτηση ελαχιστοποίησης της πολυπλοκότητας του παραγόμενου ερωτήματος υπαγορεύει τη χρήση κριτηρίου επιλογής δένδρων ελαχίστου μεγέθους.

- Αθροιστικό βάρος τριπλετών ακμών δένδρου
  Το *αθροιστικό βάρος τριπλετών ακμών* ενός δένδρου $T = (E, V)$ ορίζεται ως $\sum_{e \in E} w_R\,(e)$, όπου $w_R\,(e)$ το βάρος τριπλετών της ακμής e του T. Το βάρος τριπλετών ακμής $w_R\,(e)$ είναι ανάλογο του πλήθους των τριπλετών από τις οποίες προήλθε η συγκεκριμένη ακμή, κατά τη δημιουργία της σύνοψης RDF. Η πιθανότητα επιτυχίας του ερωτήματος, είναι ανάλογη του πλήθους τριπλετών. Επιπλέον, θεωρούμε ότι αυτή η πιθανότητα είναι ανάλογη του αθροιστικού βάρους τριπλετών του δένδρου. Ανάμεσα σε δύο διαφορετικά δένδρα ίδιου μεγέθους προτιμάται αυτό που ενώνει μεγαλύτερο

πλήθος τριπλετών. Συνεπώς, η απαίτηση μεγιστοποίησης της πιθανότητας επιτυχίας του παραγόμενου ερωτήματος υπαγορεύει τη χρήση κριτηρίου επιλογής δένδρων ελαχίστου αθροιστικού βάρους τριπλετών.

Το συμπέρασμα που προκύπτει είναι ότι η αποδοτική σύνδεση των επιλεγμένων, από τον χρήστη, κόμβων συντελείται με την επιλογή δένδρων *ελαχίστου μεγέθους* και *ελαχίστου αθροιστικού βάρους τριπλετών ακμών*. Αυτό το πρόβλημα σχετίζεται με την κατηγορία προβλημάτων Steiner Tree, δηλαδή εύρεσης δένδρων Steiner σε γραφήματα, για την οποία υπάρχουν στη βιβλιογραφία σχετικοί αλγόριθμοι (βλ. ενότητα 4). Με τη χρήση του βάρους ακμής που ορίζεται στη σχέση 6 της ενότητας 3.2.3 γίνεται ταύτιση του προβλήματος αποδοτικής σύνδεσης κόμβων με την κατηγορία προβλημάτων Steiner Tree.

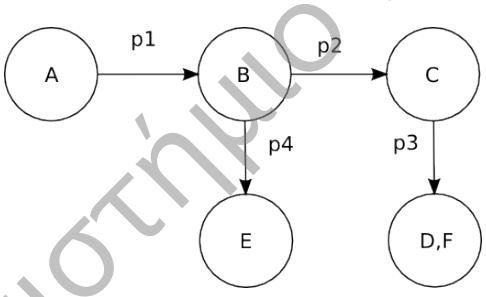
3.5 Μετατροπή Δένδρου σε SPARQL

Το ερώτημα SPARQL αποτελείται από ένα σταθερό και ένα μεταβλητό μέρος. Το σταθερό μέρος περιλαμβάνει τις δηλώσεις προθέματος και τη δήλωση τύπου ερωτήματος και αποτελεσμάτων. Το μεταβλητό μέρος περιλαμβάνει το μοτίβο ερωτήματος, που αποτελείται από μοτίβα επιλογής τριπλετών.

Η μετατροπή ενός δένδρου σε ερώτημα SPARQL βασίζεται στην παραγωγή του μεταβλητού μέρους. Αυτό γίνεται με την αντικατάσταση κάθε ακμής του δένδρου από ένα μοτίβο επιλογής τριπλετών, όπου το κατηγόρημα αντιστοιχεί στην ακμή, ενώ το υποκείμενο και το αντικείμενο αντιστοιχούν σε μοναδικές μεταβλητές. Επιπλέον, για κάθε απλό κόμβο τύπου RDF του γραφήματος προστίθεται ένα μοτίβο επιλογής τριπλετών της μορφής

”μεταβλητή” a ”τάξη κόμβου”

όπου στη θέση του υποκειμένου χρησιμοποιείται η μοναδική μεταβλητή του κόμβου. Στην περίπτωση σύν-




```
PREFIX rdf: <http://www.w3.org/1999/02/22—rdf—syntax—ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf—schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#Class>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
SELECT * WHERE {
    ?1 a A.
    ?2 a B.
    ?3 a C.
    ?4 a D.
    ?4 a F.
    ?5 a E.
    ?1 p1 ?2.
    ?2 p2 ?3.
    ?2 p4 ?5.
    ?3 p3 ?4.
}
```

Εικόνα 3.5: Παράδειγμα μετατροπής δένδρου σε ερώτημα SPARQL. Όλοι οι κόμβοι, είναι τυποποιημένοι. Ο κόμβος D, F είναι ένας σύνθετος κόμβος.

θετου κόμβου τύπων RDF, προστίθενται τόσα μοτίβα επιλογής τριπλετών, όσες και οι τάξεις που τον απαρτίζουν. Αν ο κόμβος είναι μη τυποποιημένος, τότε δεν προστίθεται κανένα τέτοιο μοτίβο επιλογής τριπλετών.

Είναι φανερό ότι το πλήθος των μοτίβων επιλογής τριπλετών του παραγόμενου ερωτήματος SPARQL είναι ανάλογο του πλήθους ακμών του δένδρου προέλευσης και συγκεκριμένα δίνεται από τη σχέση

$$n = 2\,|E| + 1 + c \tag{15}$$

όπου n το πλήθος μοτίβων επιλογής τριπλετών, $|E|$ το πλήθος ακμών και c μια σταθερά που εξαρτάται από το πλήθος σύνθετων κόμβων, η οποία μηδενίζεται όταν το πλήθος τους είναι μηδέν.

4 Αποδοτική Σύνδεση Κόμβων

Όπως αναφέρθηκε στην προηγούμενη ενότητα, το εφαρμοσμένο πρόβλημα αποδοτικής σύνδεσης κόμβων της σύνοψης RDF βασίζεται στο θεωρητικό πρόβλημα εύρεσης δένδρων Steiner σε γραφήματα. Σε αυτή την ενότητα ορίζουμε το πρόβλημα εύρεσης δένδρων Steiner σε γραφήματα (και παραλλαγές αυτού) και εξετάζουμε κάποιους αλγορίθμους που το επιλύουν. Επιπλέον, αναφερόμαστε στο συγγενές, εφαρμοσμένο πρόβλημα της αναζήτησης λέξεων-κλειδιών σε βάση δεδομένων, που αναπαρίσταται ως γράφημα, το οποίο ανάγεται εξολοκλήρου στη λύση του προβλήματος εύρεσης δένδρων Steiner σε γραφήματα.

Για τις ανάγκες της εργασίας επιλέξαμε να λύσουμε το πρόβλημα αποδοτικής επιλογής συνεκτικών δένδρων αναπτύσσοντας έναν αλγόριθμο, ο οποίος περιγράφεται στην ενότητα 4.3.2.

4.1 Κατηγορία Προβλημάτων Steiner Tree

Στην παρούσα εργασία τερματικός κόμβος ορίζεται ως ο κόμβος της σύνοψης RDF που έχει επιλέξει ο χρήστης. Ο τρόπος με τον οποίο γίνεται αυτή η επιλογή θα παρουσιαστεί σε επόμενη ενότητα.

- Πρόβλημα ST - Steiner Tree
  Έστω γράφημα $G = (V, E)$, βάρος ακμής $w : E \longrightarrow \mathbb{R}^+$ και ένα σύνολο τερματικών κόμβων $S \subseteq V$. Η λύση του προβλήματος είναι το δένδρο $T = (V_T, E_T)$ του γραφήματος G, τέτοιο ώστε $S \subseteq V_T \subseteq V$ και $E_T \subseteq E$, το οποίο έχει το ελάχιστο βάρος μεταξύ όλων των δυνατών τέτοιων δένδρων. Η συνάρτηση βάρους είναι $B(T) = \sum_{e \in E_T} w(e)$. Το πρόβλημα ST ανήκει στην κλάση πολυπλοκότητας NP-πλήρης.

- Πρόβλημα GST - Group Steiner Tree
  Έστω γράφημα $G = (V, E)$, βάρος ακμής $w : E \longrightarrow \mathbb{R}^+$ και τα σύνολα $V_1, V_2, ..., V_n \subseteq V$, που αποκαλούνται ομάδες (groups). Η λύση του προβλήματος είναι το δένδρο $T = (V_T, E_T)$ του γραφήματος G, που περιέχει τουλάχιστον ένα κόμβο (τερματικός κόμβος) από κάθε ομάδα V_i και το οποίο έχει το ελάχιστο βάρος μεταξύ όλων των δυνατών τέτοιων δένδρων. Η συνάρτηση βάρους είναι $B(T) = \sum_{e \in E_T} w(e)$. Το πρόβλημα GST ανήκει στην κλάση πολυπλοκότητας NP-πλήρης [7].

- Πρόβλημα top-k Group Steiner Tree (GST-k)
  Μια παραλλαγή του προβλήματος GST είναι το πρόβλημα GST-k, του οποίου η λύση είναι ένα διατεταγμένο σύνολο S δένδρων με πληθικό αριθμό $k = |S|$, καθένα από τα οποία περιέχει τουλάχιστον ένα κόμβο από κάθε ομάδα V_i. Επίσης, το διατεταγμένο σύνολο S περιέχει ταξινομημένα δένδρα $T_i \in S$ με κατάταξη μικρότερου βάρους δηλαδή $B(T_i) \leq B(T_{i+1})$ για $i = 1, 2, ..., k-1$. Η συνάρτηση βάρους είναι $B(T) = \sum_{e \in E_T} w(e)$.

Το πρόβλημα GST μπορεί να μετασχηματισθεί σε πρόβλημα ST [15] στην περίπτωση που οι ομάδες $V_i \subseteq V$ είναι μονοσύνολα ξένα μεταξύ τους και συνεπώς περιλαμβάνουν από έναν διαφορετικό τερματικό κόμβο.

4.2 Αναζήτηση Λέξεων Κλειδιών Με Βάση Δένδρα Steiner

Σημαντική ερευνητική προσπάθεια έχει γίνει [5] στον τομέα που σχετίζεται με το πρακτικό πρόβλημα της αναζήτησης λέξεων-κλειδιών σε γραφήματα με κατάταξη k αποτελεσμάτων (top-k keyword search). Ένα ερώτημα αναζήτησης λέξεων-κλειδιών με κατάταξη k αποτελεσμάτων, πάνω σε γράφημα G ορίζει ένα σύνολο από λέξεις-κλειδιά (keywords)

$$K = K_1, K_2, ..., K_n$$

Αλγόριθμος 4.1 Ο αλγόριθμος DPBF-k για το πρόβλημα εύρεσης των k πρώτων δένδρων Steiner.

είσοδος: γράφημα G, σύνολο λέξεων-κλειδιών P και ομάδες $V_1, \ldots, V_l$
έξοδος: βέλτιστο δένδρο Steiner
Έστω Q_T ουρά προτεραιότητας, ταξινομημένη με αύξουσα σειρά βάρους δένδρων
$Q_T \leftarrow \emptyset$
for $v \in V(G)$ **do**
    if v περιέχει λέξεις-κλειδιά p **then**
        Q_T.ENQUEUE$(T(v,p))$
while $Q_T \neq \emptyset$ **do**
    $T(v,\mathbf{p}) \leftarrow Q_T$.POP() **return** $T(v,\mathbf{p})$ if $\mathbf{p} = P$
    for each $(v,u) \in E$ **do**
        if $w((v,u) \oplus T(v,\mathbf{p})) < w(T(u,\mathbf{p}))$ **then**
            $T(u,\mathbf{p}) \leftarrow (v,u) \oplus T(v,\mathbf{p})$
            Q_T.UPDATE$(T(u,\mathbf{p}))$
    $\mathbf{p_1} \leftarrow \mathbf{p}$
    for each $\mathbf{p_2}$ τέτοιο ώστε $\mathbf{p_1} \cap \mathbf{p_2} = \emptyset$ **do**
        if $w(T(v,\mathbf{p_1}) \oplus T(v,\mathbf{p_2})) < w(T(v,\mathbf{p_1} \cup \mathbf{p_2}))$ **then**
            $T(v,\mathbf{p_1} \cup \mathbf{p_2}) \leftarrow T(v,\mathbf{p_1}) \oplus T(v,\mathbf{p_2})$
            Q_T.UPDATE$(T(v,\mathbf{p_1} \cup \mathbf{p_2}))$

και αναζητά διατεταγμένο σύνολο, έστω E_{CT}, συγκεκριμένου πλήθους $k \equiv |E_{CT}|$, κάθε στοιχείο του οποίου είναι ένα υποδένδρο T του G, τέτοιο ώστε οι κόμβοι του να περιέχουν όλες τις λέξεις-κλειδιά του K, τουλάχιστον από μία φορά, ενώ κάθε άλλο υποδένδρο του T όχι.

Ας σημειωθεί ότι οι περισσότεροι κόμβοι του δένδρου είναι δυνατόν να μην περιέχουν καμία λέξη-κλειδί του K και απλά να συνδέουν άλλους κόμβους, του ίδιου δένδρου, που περιέχουν λέξεις-κλειδιά. Το δένδρο συνήθως ταξινομείται με βάση τη σχέση

$$f_{TREE} = \sum_{e \in E_{CT}} w(e) \tag{16}$$

όπου $w(e)$ είναι το βάρος μιας ακμής e του συγκεκριμένου δένδρου.

Η εύρεση του πρώτου δένδρου ανάγεται στην λύση του κλασσικού θεωρητικού προβλήματος δένδρων Steiner με ομάδες (GST - Group Steiner Tree). Το πρόβλημα εύρεσης περισσοτέρων δένδρων, έστω k, αναφέρεται ως GST-k και συμπίπτει με το προαναφερθέν πρακτικό πρόβλημα της αναζήτησης λέξεων-κλειδιών σε γραφήματα με κατάταξη k αποτελεσμάτων.

Μέχρι σήμερα έχουν γίνει μελέτες πάνω στο πρόβλημα GST και GST-k. Στη συγκεκριμένη δημοσίευση [4], αναφέρονται και μελετώνται κάποιοι αλγόριθμοι, ένας από τους οποίους είναι ο DPBF-k . Στην επόμενη ενότητα παρουσιάζεται ο αλγόριθμος DPBF-k, καθώς και ένας άλλος αλγόριθμος, ο οποίος αποτελεί συνεισφορά της παρούσας εργασίας.

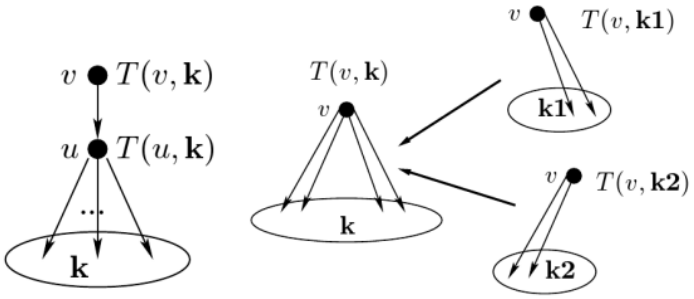
4.3 Αλγόριθμοι Εύρεσης Δένδρων Steiner

Παρά το γεγονός ότι η εύρεση του βέλτιστου δένδρου Steiner είναι NP-πλήρες πρόβλημα, υπάρχουν αποδοτικοί αλγόριθμοι που το λύνουν στην περίπτωση που το πλήθος των λέξεων-κλειδιών (έστω l) είναι μικρό. Ένας τέτοιος παραμετροποιημένος αλγόριθμος είναι ο αλγόριθμος DPBF-k.

4.3.1 Αλγόριθμος DPBF

Ο αλγόριθμος DPBF [4] επιλύει το γενικότερο πρόβλημα GST και βρίσκει πάντα το βέλτιστο δένδρο Steiner. Συνεπώς, είναι βέλτιστος για το πρόβλημα GST.

Έστω μη κατευθυνόμενο γράφημα G με βάρος και έστω ένα σύνολο λέξεων-κλειδιών $P = \{p_1, p_2, ..., p_l\}$ με πληθάριθμο l. Επίσης, έστω μια ομάδα $V_i \in V(G)$, όπου $v \in V_i$ περιέχει το p_i για $i = 1, ..., l$. Το βέλτιστο δένδρο Steiner, του προβλήματος GST, μπορεί να προσδιοριστεί εφαρμόζοντας τις εξισώσεις



Εικόνα 4.1: Διαδικασίες "tree grow" και "tree merge" του αλγορίθμου DPBF και DPBF-k.

(17),(18),(19) για κάθε $v \in V(G)$ και $p \subseteq P$.

$$T(v, \mathbf{p}) = \min\left(T_g(v, \mathbf{p}), T_m(v, \mathbf{p})\right) \tag{17}$$

$$T_g(v, \mathbf{p}) = \min_{(v,u)\in E} \{(v, u) \oplus T(u, \mathbf{p})\} \tag{18}$$

$$T_m(g, \mathbf{p_1} \cup \mathbf{p_2}) = \min_{\mathbf{p_1}\cap\mathbf{p_2}=\emptyset} \left(T_g(v, \mathbf{p_1}), T_m(v, \mathbf{p_2})\right) \tag{19}$$

Ορίζεται το βέλτιστο δένδρο $T(v, \mathbf{p})$, ως το δένδρο με το μικρότερο βάρος ανάμεσα σε όλα τα δένδρα με ρίζα τον κόμβο v, τα οποία περιέχουν όλες τις λέξεις-κλειδιά στο p. Εδώ ως ελάχιστο δένδρο λαμβάνεται το δένδρο με το ελάχιστο βάρος ανάμεσα στα υπόλοιπα δένδρα. Να σημειωθεί ότι $T(v, \mathbf{p})$ μπορεί να μη υπάρχει για κάποια v και $\mathbf{p}$, πράγμα που αντανακλά το γεγονός ότι ο κόμβος v δεν μπορεί να φτάσει κάποιους κόμβους με λέξεις-κλειδιά $\mathbf{p}$, και τότε το βάρος είναι άπειρο. Το $T_g(v, \mathbf{p})$ αντιστοιχεί στην διαδικασία "tree grow" και το $T_m(v, \mathbf{p})$ στη διαδικασία "tree merge" (εικ. 4.1).

Ο αλγόριθμος μπορεί να μετατραπεί, ώστε να λύνει το GST-k πρόβλημα, χωρίς μεταβολή της πολυπλοκότητάς του. Στη σχετική δημοσίευση[4] αναφέρεται ως DPBF-k (βλ. αλγόριθμο 4.1). Τα υπόλοιπα $k-1$ δένδρα δεν είναι βέλτιστα, αλλά προκύπτουν κατά προσέγγιση.

Πολυπλοκότητα Η εύρεση του 1ου δένδρου Steiner (πρόβλημα GST-1) έχει πολυπλοκότητα

- χρονική: $O\left(3^l n + 2^l\left((l + \log n)\, n + m\right)\right)$ και
- χωρική: $O\left(2^l n\right)$

όπου n είναι το πλήθος κόμβων του γραφήματος, m το πλήθος των ακμών και l το πλήθος των λέξεων κλειδιών. Για την εύρεση των υπολοίπων k-1 δένδρων (πρόβλημα GST-k), η πολυπλοκότητα παραμένει η ίδια. Οι υπολογισμοί πολυπλοκότητας ισχύουν μόνο αν χρησιμοποιείται ουρά προτεραιότητας (priority queue) υλοποιημένη με σωρό Fibonacci.

Περιορισμοί Αλγορίθμου Ο αλγόριθμος DPBF-k δεν χρησιμοποιήθηκε σε αυτή την εργασία επειδή αναφέρει ως μοναδικό κριτήριο επιλογής δένδρων το βάρος τους. Αντίθετα, όπως έχει ήδη αναφερθεί, η επιλογή δένδρων της σύνοψης RDF πρέπει να βασίζεται σε δύο κριτήρια: α) το ελάχιστο πλήθος ακμών και β) το βάρος τους.

4.3.2 Αλγόριθμος επίλυσης προβλήματος ST-k Πολλαπλών Μονοπατιών

Στην εργασία αυτή υλοποιήθηκε ένας αλγόριθμος επίλυσης του προβλήματος ST-k χρησιμοποιώντας συντομότερα μονοπάτια. Ο αλγόριθμος αυτός στηρίζεται στην εναλλακτική ευριστική ST-1 για γραφήματα, όπως αναφέρεται στη σελίδα 557 [10] επεκτείνοντάς τον, ώστε να λύνει το πρόβλημα ST-k.

Πιο συγκεκριμένα, έστω U το σύνολο τερματικών κόμβων ενός γραφήματος, με πλήθος $u \equiv |U|$. Αρχικά υπολογίζονται όλες οι δυνατές αναδιατάξεις των τερματικών κόμβων, έστω P, με πλήθος $p \equiv |P| = u!$. Για κάθε αναδιάταξη κατασκευάζεται ένα δένδρο και υπολογίζεται το βάρος του (σχέση 16). Τα δένδρα αυτά αποθηκεύονται σε μια ουρά προτεραιότητας με προτεραιότητα μικρότερου βάρους δένδρου. Κάθε δένδρο κατασκευάζεται ως εξής:

- Αρχικοποίηση: Κατασκευή ενός κενού δένδρου (τρέχον δένδρο)
- Στη συνέχεια, για κάθε κόμβο t της αναδιάταξης P_i:
  - Για κάθε κόμβο v του τρέχοντος δένδρου, υπολογίζουμε το συντομότερο μονοπάτι v-t (από τον κόμβο v στον κόμβο t). Το μονοπάτι v-t είναι το πρώτο μονοπάτι από ένα πλήθος n μονοπατιών τα οποία συνδέουν τους κόμβους v και t διατεταγμένα από το μικρότερο στο μεγαλύτερο μονοπάτι ($n = 1$). Το μικρότερο από αυτά τα μονοπάτια ενσωματώνεται στο τρέχον δένδρο.
- Ο αλγόριθμος διακόπτεται αν το πλήθος δένδρων της ουράς είναι ίσο με το πλήθος k ζητούμενων δένδρων Steiner. Στο τέλος, τα δένδρα που βρίσκονται στη ουρά προτεραιότητας είναι ταξινομημένα κατά αύξουσα σειρά βάρους.

Μέχρι τώρα, ο αλγόριθμος περιορίζεται στην εύρεση το πολύ $u!$ δένδρων. Στην περίπτωση όπου $k > u!$ είναι προφανές ότι θα επιστραφούν λιγότερα δένδρα Steiner από τα ζητούμενα. Έτσι λοιπόν, ο αλγόριθμος επεκτείνεται προκειμένου να ξεπερνά αυτόν τον περιορισμό.

Πιο συγκεκριμένα, ο αλγόριθμος μετασχηματίζεται ως εξής:

- Αρχικοποίηση: Κατασκευή ενός κενού δένδρου (τρέχον δένδρο) και ενός κενού ευρετηρίου μονοπατιών και $n = 0$.
- Μέχρι να μην είναι αληθής η συνθήκη Α (βλέπε επόμενο βήμα αλγορίθμου) ή το πλήθος δένδρων της ουράς να είναι ίσο με το πλήθος k ζητούμενων δένδρων Steiner εκτελείται το παρακάτω βήμα:
  - Αυξάνεται το n κατά μονάδα.
  - Για κάθε κόμβο t της αναδιάταξης P_i:
    - Για κάθε κόμβο v του τρέχοντος δένδρου κατασκευάζουμε σύνολο μονοπατιών $M(t)$ μεταξύ αυτού και του κόμβου t ως εξής:
      - Αν το ευρετήριο μονοπατιών δεν περιέχει κάποιο μονοπάτι $(v,t) \in D_{v,t}$, όπου $D_{v,t}$ είναι το σύνολο μονοπατιών τα οποία συνδέουν τους κόμβους v και t διατεταγμένα από το μικρότερο στο μεγαλύτερο μονοπάτι, τότε (a) υπολογίζεται το συντομότερο μονοπάτι $m_r(v,t)$ (b) αποθηκεύεται στο ευρετήριο με τιμή $r = n$ και (c) προστίθεται στο σύνολο $M(t)$, δηλαδή $M(t) = M(t) \bigcup \{m_n(v,t)\}$.
      - Αν το ευρετήριο μονοπατιών περιέχει κάποιο μονοπάτι (v,t), έστω $m_r(v,t)$, τότε
        - Αν $r = n$, τότε προστίθεται στο σύνολο $M(t)$, δηλαδή $M(t) = M(t) \bigcup \{m_r(v,t)\}$
        - Αν $r < n$ και είναι δυνατός ο υπολογισμός επόμενου συντομότερου μονοπατιού, δηλαδή $n < |D_{v,t}|$, τότε (a) υπολογίζεται το αμέσως επόμενο συντομότερο μονοπάτι (b) αποθηκεύεται στο ευρετήριο με τιμή $r = n$ και (c) προστίθεται στο σύνολο $M(t)$, δηλαδή $M(t) = M(t) \bigcup \{m_r(v,t)\}$
        - Αν $r < n$ και είναι αδύνατος ο υπολογισμός επόμενου συντομότερου μονοπατιού, δηλαδή $n \geq |D_{v,t}|$, τότε (a) προστίθεται το υπάρχον μονοπάτι στο σύνολο $M(t)$, δηλαδή $M(t) = M(t) \bigcup \{m_r(v,t)\}$ και (b) ολοκληρώνεται ο έλεγχος μονοπατιών για το συγκεκριμένο ζεύγος κόμβων (v,t).
    - Το ελάχιστο στοιχείο του συνόλου M(t), δηλαδή το $\min(M(t))$προστίθεται στο τρέχον δένδρο.
- Η συνθήκη Α ορίζεται ως εξής: κάθε ζεύγος κόμβων (v,t) του ευρετηρίου έχει ολοκληρωθεί ο έλεγχος μονοπατιών.

5 Αρχιτεκτονική Εφαρμογής

Λαμβάνοντας υπόψη τα ευρήματα της θεωρητικής προσέγγισης του προβλήματος που περιγράφηκε στην προηγούμενη ενότητα, υλοποιήθηκε μια εφαρμογή που παρέχει υπηρεσίες αναζήτησης πληροφοριών σε γράφημα RDF, βασισμένες στη σύνοψη αυτού.

Ως υπηρεσία αναζήτησης ορίζεται ένα σύνολο αλληλεπιδράσεων μεταξύ χρήστη και εφαρμογής, που οδηγούν στην παραγωγή SPARQL ερωτημάτων, συμβατών με το υποκείμενο γράφημα RDF. Οι αλληλεπιδράσεις δεν προϋποθέτουν τη γνώση της δομής του γραφήματος RDF από τον χρήστη. Η υπηρεσία αναζήτησης βοηθάει τον χρήστη στη δημιουργία κατάλληλων SPARQL ερωτημάτων που αντιστοιχούν στις πληροφοριακές του ανάγκες. Τα ερωτήματα αυτά έχουν αρκετά μεγάλη πιθανότητα να φέρουν χρήσιμα αποτελέσματα όταν απευθυνθούν στο αρχικό γράφημα RDF. Η υπηρεσία αναζήτησης έχει καλύτερη απόδοση όταν τα ερωτήματα, που της απευθύνονται, περιέχουν λεκτικά, που αντιστοιχούν σε τύπους δεδομένων κι όχι στα δεδομένα καθεαυτά. Ας θεωρήσουμε, για παράδειγμα, τρεις περιπτώσεις ερωτημάτων:

1. Συνθέτες γεννημένοι σε πρωτεύουσες κρατών
2. Συνθέτες γεννημένοι στη ”Στοκχόλμη”
3. Είναι ο ”Bach” γεννημένος στην ”Αυστρία”;

Στην πρώτη περίπτωση, το ερώτημα που θα προταθεί από την υπηρεσία αναζήτησης, σίγουρα θα οδηγήσει στη λήψη αποτελεσμάτων. Στη δεύτερη περίπτωση θα οδηγήσει στη λήψη αποτελεσμάτων μόνο αν στο γράφημα RDF υπάρχει τουλάχιστον ένα URI τύπου ”συνθέτες” που να συνδέεται μέσω της σχέσης ”γεννημένος” με το URI που αντιστοιχεί στην πόλη ”Στοκχόλμη”. Η τρίτη περίπτωση δεν αποτελεί αποδεκτό ερώτημα και δεν μπορεί να εξυπηρετηθεί από την υπηρεσία αναζήτησης. Η απάντηση θα μπορούσε να είναι τύπου Αληθές ή Ψευδές, ανάλογα με το αν το URI στο οποίο αντιστοιχεί το λεκτικό ”Bach” συνδέεται με το URI στο οποίο αντιστοιχεί το λεκτικό ”Αυστρία”.

Η εφαρμογή αποτελείται από τρία μέρη (εικ. (5.1)), τα εργαλεία επεξεργασίας των δεδομένων RDF, το γραφικό περιβάλλον χρήστη (Web UI) και τους εξυπηρετητές.

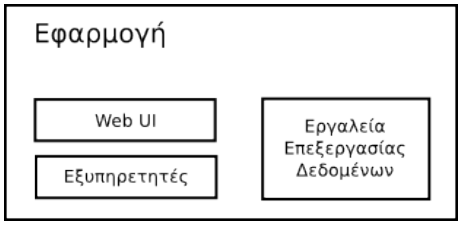
- Τα εργαλεία επεξεργασίας των δεδομένων RDF αναλαμβάνουν τη λήψη, ανάλυση και μετασχηματισμό των δεδομένων RDF σε μορφή κατάλληλη (εικ. 5.3) για χρήση από τους εξυπηρετητές GraphDB και Elasticsearch.
- Το γραφικό περιβάλλον αποτελεί μέρος της υπηρεσίας αναζήτησης και αναλαμβάνει τη διεπαφή με τον χρήστη. Πιο συγκεκριμένα, επιτρέπει τη μετατροπή των πληροφοριακών αναγκών του χρήστη σε λεκτικά, που υπάρχουν στο γράφημα RDF και παραπέμπουν στη σύνοψη αυτού. Με αυτόν τον τρόπο επιτυγχάνεται η επιλογή των κόμβων της σύνοψης που θα συνδεθούν μέσω του αλγορίθμου που περιγράφηκε στην ενότητα 4.3.2.
- Οι εξυπηρετητές αποτελούν μέρος της υπηρεσίας αναζήτησης και είναι οι εξής: εξυπηρετητής GraphDB, εξυπηρετητής ιστού (Web Server) και εξυπηρετητής Elasticsearch. Ο τελευταίος δεν υλοποιήθηκε από εμάς, αλλά χρησιμοποιήθηκε για τις ανάγκες αυτής της εργασίας.

Επιπλέον, στο αρχείο corefunctions.py ορίζονται κάποιες κοινές συναρτήσεις, οι οποίες χρησιμοποιούνται από τα υπόλοιπα προγράμματα. Περιλαμβάνει τις συναρτήσεις αναζήτησης top-k δένδρων Steiner (υλοποίηση αλγορίθμου 4.3.2) και top-k συντομότερων μονοπατιών (υλοποίηση του αλγορίθμου Yen [14] που επιλύει το πρόβλημα ST-k). Επιπλέον, ορίζονται συναρτήσεις χειρισμού των κόμβων, όπως η συνάρτηση is_pnode(n), η οποία αναγνωρίζει αν ο κόμβος n της σύνοψης είναι κατηγόρημα ή όχι.

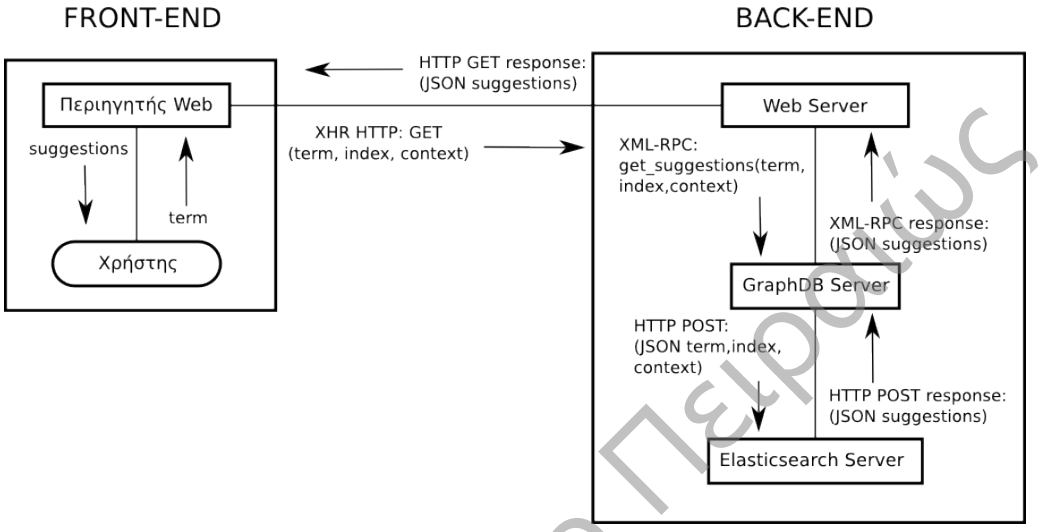
Η υπηρεσία αναζήτησης περιλαμβάνει το γραφικό περιβάλλον χρήστη, τον εξυπηρετητή ιστού (Web Server), τον εξυπηρετητή GraphDB και τον εξυπηρετητή Elasticsearch, όπου καθένα από τα παραπάνω μέρη επικοινωνεί μόνο με τα διπλανά του (εικ. (5.2)). Σε αυτή τη τοπολογία, ο χρήστης είναι αποκομμένος από τους δύο τελευταίους εξυπηρετητές, γνωρίζοντας την ύπαρξη μόνο του εξυπηρετητή ιστού.

Ο εξυπηρετητής GraphDB φορτώνει τη σύνοψη RDF και ορίζει ένα σύνολο μεθόδων αξιοποίησής της. Επιπλέον, παρέχει τη μέθοδο getsuggestions() που χρησιμεύει στην υλοποίηση της υπηρεσίας αυτόματης συμπλήρωσης πεδίων εισόδου (auto complete) του γραφικού περιβάλλοντος χρήστη.

Η σύνδεση στον εξυπηρετητή GraphDB γίνεται με χρήση του πρωτοκόλλου απομακρυσμένης επικοινωνίας διεργασιών (RPC - Remote Process Communication) XML-RPC που επιτρέπει την απομακρυσμένη



Εικόνα 5.1: Συστατικά στοιχεία εφαρμογής.



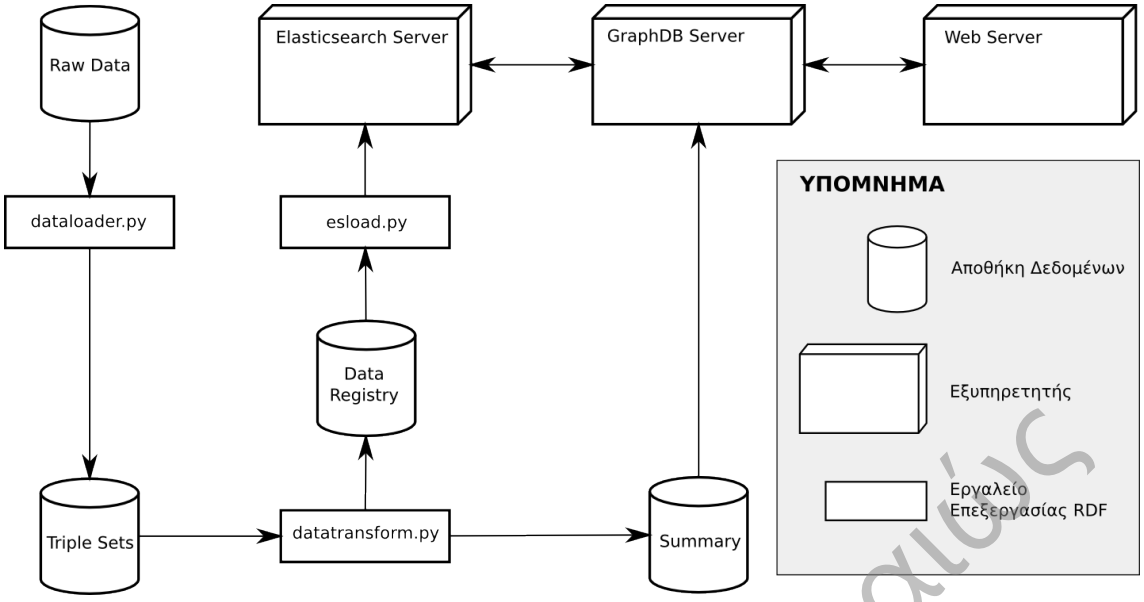
Εικόνα 5.2: Υπηρεσία αυτόματης συμπλήρωσης πεδίων. Οι παράμετροι index και context καθορίζονται από τη κλάση του πεδίου HTML input, όπου πληκτρολογήθηκε ο όρος term.

εκτέλεση όλων των παρεχόμενων μεθόδων του εξυπηρετητή. Οι τελικοί χρήστες, συνδέονται απομακρυσμένα, μέσω περιηγητή, στον εξυπηρετητή ιστού. Ο εξυπηρετητής ιστού έχει πρόσβαση στη μέθοδο execute_query() και get_suggestions() του εξυπηρετητή GraphDB. Οι διαχειριστές συνδέονται στον εξυπηρετητή GraphDB με το πρόγραμμα client.py και έχουν πρόσβαση σε όλες τις παρεχόμενες μεθόδους. Για λόγους ασφάλειας είναι απαραίτητος ο περιορισμός απομακρυσμένης πρόσβασης στον εξυπηρετητή GraphDB. Η χρήση πρωτοκόλλου RPC επιτρέπει τον διαχωρισμό των εξυπηρετητών σε ένα δίκτυο, ώστε να μην είναι απαραίτητη η συμβίωσή τους στον ίδιο υπολογιστή.

Ένα τυπικό σενάριο χρήσης της υπηρεσίας αναζήτησης είναι αυτό στο οποίο ο χρήστης ανοίγει τον περιηγητή Ιστού και συνδέεται στον ιστότοπο του εξυπηρετητή ιστού. Εκεί συμπληρώνει τη φόρμα αναζήτησης υποβοηθούμενος από την υπηρεσία αυτόματης συμπλήρωσης πεδίων. Στη συνέχεια, πατά το κουμπί αναζήτησης και η φόρμα αποστέλλεται, με ασύγχρονο XHR (XML HTTP Request), στον εξυπηρετητή ιστού. Εκεί γίνεται αποκωδικοποίηση των παραμέτρων αναζήτησης (query string) και εκτέλεση της μεθόδου execute_query(), μέσω XML-RPC στον εξυπηρετητή GraphDB. Εκεί γίνεται ανάλυση των παραμέτρων αναζήτησης και πραγματοποιείται αναζήτηση των επιλεχθέντων, από τον χρήστη, κόμβων στη σύνοψη, με σκοπό την εύρεση δένδρων. Στη συνέχεια, τα δένδρα μετατρέπονται σε ερωτήματα SPARQL και επιστρέφονται στον εξυπηρετητή ιστού μαζί με επιπλέον πληροφορίες. Πιο συγκεκριμένα, αυτές οι επιπλέον πληροφορίες είναι η χρονική διάρκεια της αναζήτησης και η περιγραφή σφάλματος στην περίπτωση που προέκυψε σφάλμα κατά την αναζήτηση. Στον εξυπηρετητή ιστού γίνεται λήψη των πληροφοριών και κωδικοποίηση σε μορφή JSON. Κατόπιν, το αντικείμενο JSON αποστέλλεται στον περιηγητή του χρήστη. Τα αποτελέσματα μορφοποιούνται κατάλληλα και παρουσιάζονται στον χρήστη.

5.1 Εργαλεία Επεξεργασίας Δεδομένων RDF

Τα εργαλεία επεξεργασίας δεδομένων RDF διεκπεραιώνουν το προπαρασκευαστικό στάδιο της ανάλυσης δεδομένων, πάνω στην οποία βασίζεται η λειτουργία της υπηρεσίας. Αποτελούνται από τα προγράμματα



Εικόνα 5.3: Προεπεξεργασία γραφήματος RDF με χρήση των αντίστοιχων εργαλείων. Η αποθήκη δεδομένων "Raw Data" συμβολίζει το αρχικό γράφημα RDF.

dataloader.py, datatransform.py και esload.py, που είναι κωδικοποιημένα σε γλώσσα Python 2.7. Η εκτέλεσή τους γίνεται σε περιβάλλον γραμμής εντολών. Τα προγράμματα έχουν δοκιμαστεί και λειτουργούν σε σύστημα Linux.

Στις επόμενες ενότητες παρουσιάζονται τα εργαλεία επεξεργασίας των δεδομένων RDF και γίνεται εκτενής ανάλυση της υλοποίησής τους. Επιπλέον, γίνεται αναφορά στον τρόπο χρήσης τους.

5.1.1 Πρόγραμμα dataloader.py

Το dataloader.py είναι ένα πρόγραμμα που αναλαμβάνει τη σύνδεση με την αποθήκη πληροφοριών RDF και την λήψη των κατάλληλων δεδομένων και την αποθήκευσή τους σε κατάλογο της επιλογής του χρήστη. Η αποθήκη πληροφοριών πρέπει να υποστηρίζει επικοινωνία μέσω πρωτοκόλλου HTTP, με είσοδο ερωτημάτων SPARQL και έξοδο SPARQL/XML. Μια τέτοια αποθήκη πληροφοριών RDF είναι η 4store[2], που αποτελεί ελεύθερο λογισμικό. Το dataloader.py υποστηρίζει το 4store.

Τρόπος χρήσης Το πρόγραμμα εκτελείται από τη γραμμή εντολών με την εντολή

```
python dataloader.py [-h] [-d TEXT] URL OUTPUT_DIR
```

και δέχεται τις εξής παραμέτρους:

- URL
  Το URL του τελικού σημείου (endpoint) SPARQL για τη φόρτωση των δεδομένων.
- OUTPUT_DIR
  Ο κατάλογος εξόδου.
- -h, ή –help
  Εμφάνιση βοήθειας χρήσης προγράμματος.
- -d TEXT, ή –datatype TEXT
  Λεκτικό χαρακτηρισμού του τύπου δεδομένων ενός απλού (plain) literal (προεπιλογή: ’plain’).

[2] Ο πηγαίος κώδικας του 4store βρίσκεται στην εξής διεύθυνση: https://github.com/garlik/4store

Ανάλυση Λειτουργίας Κατά την έναρξη εκτέλεσης του προγράμματος dataloader.py γίνεται επεξεργασία των παραμέτρων εισόδου και δημιουργείται ο κατάλογος εξόδου, σε περίπτωση που δεν υπάρχει. Στη συνέχεια, εκτελείται η συνάρτηση load_data() που είναι υπεύθυνη για τη λήψη όλων των τριπλετών από την αποθήκη πληροφοριών RDF. Τελειώνοντας, αποθηκεύονται τα αρχεία list1.p, list2.p, και list3.p στον κατάλογο εξόδου. Η αποθήκευση των αρχείων γίνεται με τη βιβλιοθήκη cPickle της Python, η οποία επιτρέπει τη μόνιμη αποθήκευση δομών δεδομένων της Python σε αρχεία στο σύστημα.

Η λήψη των πληροφοριών γίνεται με τις συναρτήσεις getlist1(), getlist2() και getlist3(), οι οποίες έχουν παρόμοια δομή. Αρχικά, προετοιμάζεται ένα ερώτημα SPARQL. το οποίο αποστέλλεται μέσω πρωτοκόλλου HTTP, με τη μέθοδο POST, στον εξυπηρετητή SPARQL ερωτημάτων του 4store. Για τη διασφάλιση της πληρότητας του συνόλου ληφθέντων πληροφοριών, απενεργοποιείται το όριο επεξεργασίας ερωτημάτων SPARQL του εξυπηρετητή 4store, αποστέλλοντας την ειδική παράμετρο softlimit με τιμή '-1'. Για την εκτέλεση του ερωτήματος HTTP, χρησιμοποιείται η βιβλιοθήκη Requests της Python. Κατόπιν, γίνεται ανάγνωση της ληφθείσας XML απάντησης από τον εξυπηρετητή 4store, με χρήση της βιβλιοθήκης Xml της Python. Οι πληροφορίες αποθηκεύονται σε δομή λίστας (list) από πλειάδες (tuples).

Φόρτωση Τριπλετών Τύπων Η συνάρτηση getlist1() (βλ. 1) αποστέλλει το SPARQL ερώτημα

SELECT ?s ?type { ?s a ?type } ORDER BY ASC(?type)

το οποίο επιστρέφει εκείνες τις τριπλέτες που έχουν ως κατηγόρημα (predicate) το URI rdf:type, ή, συντομογραφικά, 'a'. Οι τριπλέτες αποθηκεύονται στη λίστα list1 ως πλειάδες της μορφής (URI,TYPE), όπου υπονοείται ότι το κατηγόρημα είναι πάντα το 'rdf:type'. Η λίστα list1 αποτελεί το σύνολο E_1 τριπλετών τάξεων του γραφήματος RDF (βλ. 3.2.1). Τα URI που βρίσκονται στη θέση υποκειμένου απαρτίζουν το σύνολο U_1 των τυποποιημένων URI του γραφήματος RDF (βλ. εξ. 2). Τα URI που βρίσκονται στη θέση αντικειμένου απαρτίζουν το σύνολο τύπων RDF C_{RDF}.

```
def getlist1():

    # returns association of each subject to an rdf type.
    query = "SELECT DISTINCT ?s ?type { ?s a ?type } ORDER BY ASC(?type)"
    data = { "query": query, "output": "sparql" , "soft-limit" : softlimit }
    tree = postquery(url, data)

    # out
    out_s = tree.findall('xmlns:results/xmlns:result/xmlns:binding[1]//', namespaces =
        mynamespaces)
    out_s = [i.text for i in out_s]
    out_t = tree.findall('xmlns:results/xmlns:result/xmlns:binding[2]//', namespaces =
        mynamespaces)
    out_t = [i.text for i in out_t]
    out = zip(out_s, out_t)
    return out
```

Καταλογοποίηση 1: Συνάρτηση getlist1().

Φόρτωση Τριπλετών Συνδέσεων Η συνάρτηση getlist2() αποστέλλει το SPARQL ερώτημα

SELECT DISTINCT ?s ?p ?o WHERE {
?s ?p ?o .
filter(! isLiteral(?o))
MINUS { ?s a ?o }} ORDER BY ASC(?s)

το οποίο επιστρέφει εκείνες τις τριπλέτες που δεν έχουν λεκτικό στη θέση αντικειμένου και δεν έχουν το rdf:type στη θέση κατηγορήματος. Οι τριπλέτες αποθηκεύονται στη λίστα list2 ως πλειάδες της μορφής (subject URI,predicate,object URI). Η λίστα list2 αποτελεί το σύνολο E_2 των τριπλετών συνδέσεων (βλ. 3.2.1). Τα URI που βρίσκονται στη θέση υποκειμένου και αντικειμένου της λίστας αποτελούν το σύνολο U_2 διασυνδεδεμένων URI του γραφήματος RDF (βλ. εξ. 4).

Φόρτωση Τριπλετών Λεκτικών Η συνάρτηση getlist3() αποστέλλει το SPARQL ερώτημα

SELECT DISTINCT ?s ?p ?o { ?s ?p ?o filter(isLiteral(?o)) }

το οποίο επιστρέφει εκείνες τις τριπλέτες που έχουν ως κατηγόρημα (predicate) το URI rdf:type, ή, συντομογραφικά, 'a'. Οι τριπλέτες αποθηκεύονται στη λίστα list3 ως πλειάδες της μορφής (subject, predicate, literal_value, literal_datatype). Η list3 αποτελεί το σύνολο E_3 τριπλετών λεκτικών του γραφήματος RDF (βλ. 3.2.1). Τα URI που βρίσκονται στη θέση υποκειμένου, απαρτίζουν το σύνολο U_3 των URI, του γραφήματος RDF, που αντιστοιχούν σε λεκτικό (βλ. εξ. 5). Τα λεκτικά που βρίσκονται στη τρίτη θέση καθεμιάς πλειάδας της λίστας απαρτίζουν το σύνολο λεκτικών L_{RDF} του γραφήματος RDF.

5.1.2 Πρόγραμμα datatransform.py

Το datatransform.py είναι ένα πρόγραμμα που διαβάζει τα παραγόμενα αρχεία του dataloader.py και δημιουργεί τη σύνοψη RDF (RDF summary) και το μητρώο δεδομένων (Data Registry). Το μητρώο δεδομένων αποτελείται από όλα τα δεδομένα που υπάρχουν στο γράφημα RDF. Αντίθετα, η σύνοψη RDF αποτελείται από τους τύπους RDF των δεδομένων αυτών και τις μεταξύ τους συνδέσεις.

Τρόπος Χρήσης Το πρόγραμμα εκτελείται από τη γραμμή εντολών με την εντολή

python datatransform.py [-h] [-o OUTPUT_DIR] [-d TEXT] [-p TEXT] [-u TEXT] [-c TEXT] [–debug] [–spaths] INPUT_DIR

και δέχεται τις εξής παραμέτρους:

- INPUT_DIR
  Κατάλογος που περιέχει τα αρχεία εισόδου.
- -h, –help
  Εμφάνιση βοήθειας χρήσης προγράμματος.
- -o OUTPUT_DIR, –out OUTPUT_DIR
  Κατάλογος εξόδου για την αποθήκευση των παραγόμενων αρχείων (προεπιλογή: ίδιος με τον κατάλογο εισόδου).
- -d TEXT, –datatype TEXT
  Λεκτικό χαρακτηρισμού του τύπου δεδομένων ενός απλού (plain) literal (προεπιλογή: 'plain').
- -p TEXT, –part TEXT
  Λεκτικό για τον διαχωρισμό του ονόματος τάξης rdf από την αρίθμηση partition κατά τη φάση του partitioning. Ο πρώτος χαρακτήρας πρέπει να είναι χαρακτήρας διαστήματος (προεπιλογή: ' part').
- -u TEXT, –unknown TEXT
  Λεκτικό αντικατάστασης των άγνωστων τάξεων rdf (προεπιλογή: 'unknown').
- -c TEXT, –combo TEXT
  Λεκτικό διαχωρισμού των τάξεων που απαρτίζουν έναν σύνθετο κόμβο τάξεων της σύνοψης RDF (προεπιλογή: '\n').
- –debug
  Ενεργοποιεί επιπλέον ελέγχους αποσφαλμάτωσης (προεπιλογή: 'False').
- –spaths
  Ενεργοποιεί τη διαδικασία υπολογισμού των συντομότερων μονοπατιών μεταξύ όλων των ζευγαριών κόμβων του γραφήματος (προεπιλογή: 'False').

Δημιουργία Μητρώου Δεδομένων Το μητρώο δεδομένων αποτελεί τη διεπαφή του χρήστη με το σύστημα και περιέχει αντιστοιχίσεις των αναγνώσιμων λεκτικών που υπάρχουν στο γράφημα RDF με τα αντίστοιχα URIs αυτών. Πιο συγκεκριμένα, υπάρχουν οι εξής περιπτώσεις αναγνώσιμων λεκτικών:

- rdftypes.p είναι το σύνολο D_t διατεταγμένων ζευγών $(c, \text{suffix}\,(c)) \in D_t$, που αποτελεί μονοσήμαντη απεικόνιση τάξης RDF $c \in C_{RDF}$ σε κατάληξη του URI της.
- predicates.p είναι το σύνολο D_p διατεταγμένων ζευγών $(c, \text{suffix}\,(c)) \in D_p$, που αποτελεί μονοσήμαντη απεικόνιση κατηγορήματος της σύνοψης $p \in P_S$ σε κατάληξη του URI της.
- literal_to_type_and_nodelist.p είναι το σύνολο D_v που αποτελεί μονοσήμαντη απεικόνιση καθενός λεκτικού RDF, σε ένα κόμβο της σύνοψης.
- literal_rdfs_to_type_and_nodelist.p είναι το σύνολο D_{vs} που αποτελεί μονοσήμαντη απεικόνιση καθενός λεκτικού RDF, το οποίο έχει την ιδιότητα να συνδέεται με URIs μέσω του κατηγορήματος 'rdfs:label', σε ένα κόμβο της σύνοψης.

Δημιουργία Σύνοψης RDF Η διαδικασία δημιουργίας της σύνοψης RDF χωρίζεται σε τέσσερα βασικά στάδια:

1. το στάδιο αντικατάστασης τριπλετών RDF
2. το στάδιο μεταφοράς δεδομένων σε γράφημα
3. το στάδιο μετατροπής ακμών σε κόμβους
4. το στάδιο ανάθεσης βαρών σε ακμές

Η σύνοψη αποθηκεύεται στο αρχείο graph.p. Στις επόμενες παραγράφους αναλύονται τα στάδια που αναφέρθηκαν προηγουμένως.

Αντικατάσταση Τριπλετών RDF Κατά την έναρξη εκτέλεσης του προγράμματος datatransform.py γίνεται επεξεργασία των παραμέτρων εισόδου και φορτώνονται τα αρχεία εισόδου list1.p, list2.p και list3.p από τον κατάλογο εισόδου.

Στη συνέχεια ξεκινάει τη φάση αντικατάστασης των δεδομένων, καλώντας τη συνάρτηση replace_nonliterals() (υλοποίηση αλγορίθμου 3.1) η οποία διατρέχει τη λίστα list2, αντικαθιστά τις τριπλέτες

?s ?p ?o .
?s a ?s-type .
?o a ?o-type .

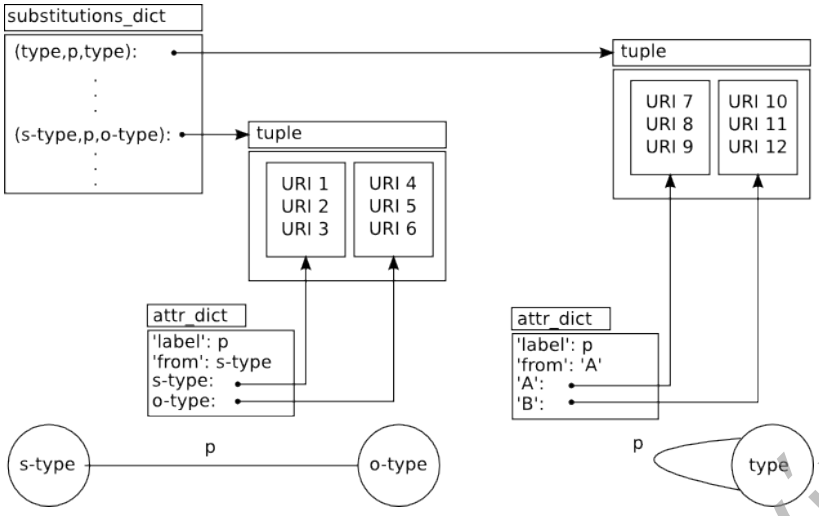
με αντίστοιχες τριπλέτες

?s-type ?p ?o-type

κι επιστρέφει ευρετήριο (substitutions_dict) με εγγραφές της μορφής

$$(stype, p, otype) : (subjects, objects)$$

όπου το subjects είναι μια λίστα που περιέχει όλα τα subjects της τάξης stype και objects είναι μια λίστα που περιέχει όλα τα objects της τάξης otype. Επιπλέον, το ν-οστό στοιχείο της λίστας subjects συνδέεται με το ν-οστό στοιχείο της λίστας objects, μέσω του κατηγορήματος p. Το ευρετήριο εξυπηρετεί στην υλοποίηση της συνάρτησης $R_C : E_2 \to E_{SC}$, όπου το σύνολο κλειδιών αποτελεί το σύνολο ακμών E_{SC}, και ισχύει ότι για κάθε τριπλέτα RDF του συνόλου E_2 υπάρχει ακριβώς μία ακμή του συνόλου E_{SC}, δηλαδή τριπλέτα σύνοψης και για κάθε τριπλέτα σύνοψης υπάρχει τουλάχιστον μία τριπλέτα RDF.



Εικόνα 5.4: Αναπαράσταση μεταφοράς των δεδομένων του ευρετήριο substitutions_dict σε πολυγράφημα MultiGraph() ως ακμή. Το attr_dict είναι το συνοδευτικό ευρετήριο της συγκεκριμένης ακμής.

Μεταφορά Δεδομένων σε Γράφημα Με την κλήση της συνάρτησης process_graph() αρχίζει η επεξεργασία των δεδομένων σε επίπεδο γραφημάτων. Η βιβλιοθήκη Networkx είναι μια βιβλιοθήκη της Python που χρησιμοποιείται για την αναπαράσταση γραφημάτων. Σε κάθε Networkx γράφημα, μια ακμή ορίζεται ως πλειάδα της μορφής (node1,node2,attr_dict), όπου attr_dict είνα το συνοδευτικό ευρετήριο της συγκεκριμένης ακμής. Αυτό το ευρετήριο προορίζεται για την αποθήκευση σχετικών πληροφοριών, όπως το βάρος της ακμής και το όνομά της.

Αρχικά, δημιουργείται ένα γράφημα στο οποίο προστίθεται μια ακμή για κάθε κλειδί του ευρετηρίου substitutions_dict, ενώ οι αντίστοιχες τριπλέτες του κλειδιού προστίθενται στο ευρετήριο που συνοδεύει την ακμή. Το όνομα της ακμής είναι ίδιο με το κατηγόρημα p, ενώ το όνομα του κόμβου προέλευσης και προορισμού είναι ίδιο με το stype και otype αντίστοιχα. Επιπλέον, το όνομα του κόμβου προέλευσης αποθηκεύεται ως κλειδί και αντιστοιχεί στον πίνακα URI του κόμβου. Το ίδιο και ο κόμβος προέλευσης. Διακρίνεται η περίπτωση των αυτοπαθών ακμών, οι οποίες έχουν ίδιο κόμβο προέλευσης και προορισμού (βλ. 5.4). Ειδικά σε αυτή τη περίπτωση, κατά σύμβαση, ο πίνακας των URI προέλευσης αναφέρεται, στο συνοδευτικό ευρετήριο ακμής (attr_dict), με το κλειδί 'A', ενώ ο πίνακας των URI προορισμού αναφέρεται με το κλειδί 'B'.

Το γράφημα αποτελεί αντικείμενο κλάσης networkx.MultiGraph(), το οποίο υποστηρίζει παράλληλες ακμές, δηλαδή ακμές διαφορετικού ονόματος p, που ενώνουν τους ίδιους κόμβους stype και otype. Η συγκεκριμένη κλάση δεν παρέχει εγγενή υποστήριξη για κατευθυνόμενα γραφήματα. Η πληροφορία κατεύθυνσης αποθηκεύεται χειροκίνητα στο συνοδευτικό ευρετήριο ακμής με τη δημιουργία κλειδιού 'from' το οποίο αντιστοιχεί στο όνομα του κόμβου προέλευσης. Για παράδειγμα 'from':'κόμβος προέλευσης.

Αξίζει να σημειωθεί πως το Networkx παρέχει την κλάση networkx.MultiDiGraph() για την υποστήριξη κατευθυνόμενων γραφημάτων με παράλληλες ακμές. Η συγκεκριμένη κλάση κρίθηκε ανεπαρκής, λόγω της αδυναμίας εκτέλεσης αλγορίθμων αναζήτησης ελαχίστων μονοπατιών προς όλες τις κατευθύνσεις. Για το πρόβλημα που έχουμε να επιλύσουμε, η πληροφορία κατεύθυνσης αξιοποιείται οπουδήποτε αλλού εκτός από το στάδιο ανεύρεσης ελαχίστων δένδρων. Ακολουθεί μέρος της υλοποίησης της μεταφοράς δεδομένων ευρετηρίου σε γράφημα Networkx.

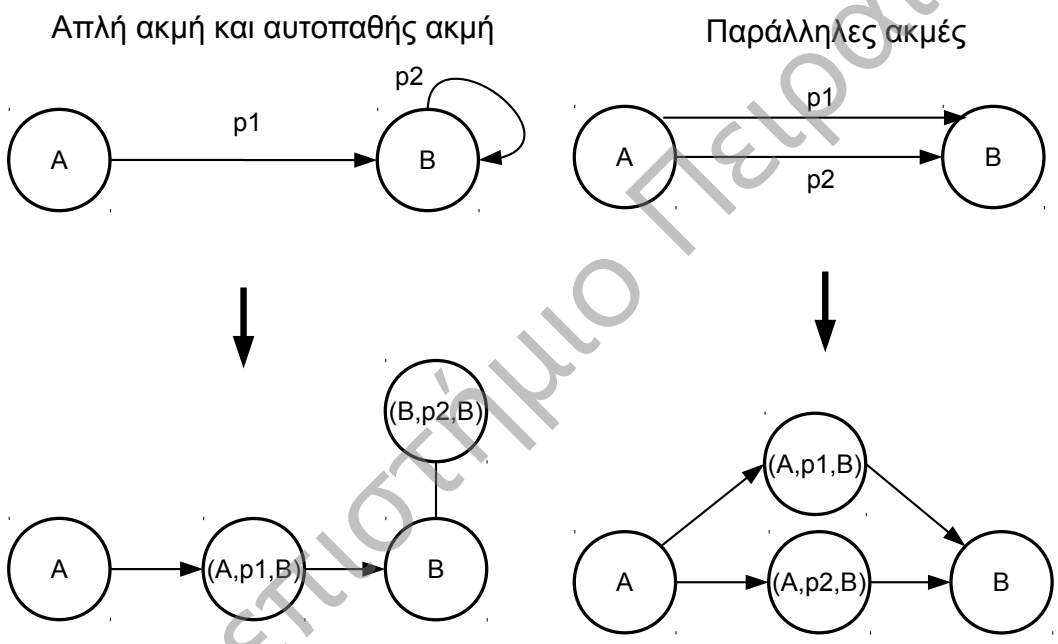
```
# Initial unpartitioned graph g0
g0 = nx.MultiGraph()

# Add substitutions from the dict
for (st,p,ot) in substitutions_dict:
  [st_instances, ot_instances] = substitutions_dict[(st,p,ot)]
  if st != ot:
    g0.add_edge(st,ot,p,{'label':p,st:st_instances, ot:ot_instances,'from':st})
  else:
```

```
    # self loop instances are stored using keys A and B
    # (node names can't be used since they are equal)
    g0.add_edge(st,ot,p,{'label':p,'A':st_instances,'B':ot_instances,'from':'A'})
```

Μετατροπή Ακμών σε Κόμβους Το γράφημα που αποτελεί τη σύνοψη RDF περιέχει παράλληλες ακμές. Σύμφωνα με την τεκμηρίωση της βιβλιοθήκης Networkx, οι αλγόριθμοι εύρεσης ελαχίστων μονοπατιών που παρέχονται από τη βιβλιοθήκη δεν μπορούν να λειτουργήσουν σε γραφήματα με παράλληλες ακμές. Ως λύση, προτείνεται η μετατροπή του πολυγραφήματος σε απλό γράφημα, δηλαδή σε ένα γράφημα με μοναδικές ακμές ανάμεσα στους κόμβους. Ακολουθώντας τις οδηγίες της τεκμηρίωσης αποφασίστηκε η μετατροπή της σύνοψης RDF σε απλό γράφημα.

Η μετατροπή του πολυγραφήματος σε απλό γράφημα γίνεται με την αντικατάσταση κάθε ακμής από έναν κόμβο. Ένας τέτοιος κόμβος θα αναφέρεται στο εξής ως pnode(εικ. 5.5). Η διαδικασία διεκπεραιώνεται από τη συνάρτηση multigraph_to_simple(), η οποία δέχεται, ως όρισμα, ένα πολυγράφημα, δηλαδή αντικείμενο κλάσης MultiGraph() και επιστρέφει το αντίστοιχο απλό γράφημα, δηλαδή αντικείμενο κλάσης Graph().



Εικόνα 5.5: Μετατροπή ακμών σε κόμβους.

Η δομή ενός pnode είναι μια πλειάδα της μορφής

((κόμβος προέλευσης,ακμή,κόμβος προορισμού),συνοδευτικό ευρετήριο ακμής)

όπου το συνοδευτικό ευρετήριο της ακμής έχει μεταφερθεί στο συνοδευτικό ευρετήριο του pnode. Το όνομα του pnode είναι η πρώτη τριπλέτα της παραπάνω πλειάδας, δηλαδή

(κόμβος προέλευσης,ακμή,κόμβος προορισμού)

Ανάθεση Βαρών στις Ακμές Ακολουθεί η φάση ανάθεσης βαρών (weighting) στις ακμές του γραφήματος, η οποία συντελείται με την εκτέλεση της συνάρτησης add_weights(). Για κάθε pnode υπολογίζεται ένα κανονικοποιημένο βάρος που εξαρτάται από το πλήθος τριπλετών T_i του κόμβου pnode. Συγκεκριμένα το βάρος υπολογίζεται από τη σχέση (βλέπε και εξ. 6 στη σελίδα 20):

$$w_i = \frac{1}{N+1}\left(1 + \frac{1-\frac{T_i}{T}}{N-1}\right)$$

όπου T το πλήθος τριπλετών στο γράφημα και N το πλήθος κόμβων pnode.

Ένα pnode τυπικά έχει δύο προσκείμενες ακμές. Στην ειδική περίπτωση που προέρχεται από αντικατάσταση αυτοπαθούς ακμής, το pnode έχει μία ακμή. Στην πρώτη περίπτωση το βάρος καθεμιάς ακμής του pnode είναι $\frac{w_i}{2}$, ενώ στη δεύτερη περίπτωση, το βάρος της μοναδικής ακμής είναι w_i. Αυτό εξασφαλίζει την διατήρηση της συνθήκης κανονικοποίησης όταν γίνεται άθροιση πάνω στο πλήθος των ακμών του γραφήματος.

5.1.3 Εργαλείο esload.py

Το πρόγραμμα esload.py αναλαμβάνει τη ρύθμιση ενός εξυπηρετητή Elasticsearch (έκδοσης 1.2.0+) και τη φόρτωση, σε αυτόν, όλων των πληροφοριών του μητρώου δεδομένων.

Στις επόμενες παραγράφους, γίνεται αναφορά σε ορολογία σχετική με την τεχνολογία Elasticsearch, η οποία διασαφηνίζεται στην ενότητα 5.2.2.

Τρόπος Χρήσης Το πρόγραμμα εκτελείται από τη γραμμή εντολών με την εντολή

python esload.py [-h] [-p PREDICATE INDEX] [-l LITERAL INDEX] INPUT_DIR URL

και δέχεται τις εξής παραμέτρους:

- INPUT_DIR
  Ο κατάλογος που περιέχει τα δεδομένα εισόδου (μητρώο δεδομένων)
- URL
  Το URL του εξυπηρετητή Elastisearch.
- -h, –help
  Εμφάνιση βοήθειας χρήσης προγράμματος.
- -p PREDICATE INDEX
  Προαιρετική παράμετρος για τον ορισμό του ονόματος του ευρετηρίου κατηγορημάτων, το οποίο από προεπιλογή είναι ”suggester_predicates”.
- -l LITERAL INDEX
  Προαιρετική παράμετρος για τον ορισμό του ονόματος του ευρετηρίου λεκτικών, το οποίο από προεπιλογή είναι”suggester_literals”.

Παράδειγμα χρήσης του εργαλείου για φόρτωση των δεδομένων του καταλόγου ./save στον εξυπηρετητή Elasticsearch, με URL http://localhost:9200, χρησιμοποιώντας προεπιλεγμένα ονόματα ευρετηρίων:

python esload.py ./save http://localhost:9200

Ανάλυση Λειτουργίας Στο πρόγραμμα γίνεται αποκλειστικά χρήση της βιβλιοθήκης elasticsearch[3], της Python, για την επικοινωνία με τον εξυπηρετητή Elasticsearch. Η βιβλιοθήκη αποτελεί έναν πελάτη για τον εξυπηρετητή Elasticsearch που παρέχει ένα επίπεδο αφαίρεσης πάνω από τη διεπαφή REST του Elasticseach, βελτιώνοντας τη συνεργασία με τη γλώσσα Python και τις δομές της.

Κατά την εκτέλεση του προγράμματος esload.py, αφού γίνει η επεξεργασία των παραμέτρων εισόδου και διαπιστωθεί η ύπαρξη του εξυπηρετητή Elasticsearch, γίνεται φόρτώση, από τον κατάλογο εισόδου, των αρχείων

- rdftypes.p. Περιέχει τους τύπος RDF του αρχικού γραφήματος RDF, μαζί με τις καταλήξεις τους.
- predicates.p. Περιέχει τα κατηγορήματα που βρίσκονται σε pnodes της σύνοψης και συνδέουν κόμβους της σύνοψης, μαζί με τις καταλήξεις τους.
- literal_to_type_and_nodelist.p. Περιλαμβάνει ευρετήριο αντιστοίχισης των ”literal που συνδέονται με οποιοδήποτε κατηγόρημα με URIs” σε κόμβους της σύνοψης.

[3]Σύμφωνα με την τεκμηρίωση της βιβλιοθήκης [6], η εγκατάσταση γίνεται με την εντολή pip install elasticsearch.

- literal_rdfs_to_type_and_nodelist.p. Περιλαμβάνει ευρετήριο αντιστοίχισης ”literal που αντιστοιχούν σε URIs μόνο μέσω του κατηγορήματος rdfs:label” σε κόμβους της σύνοψης.

Στη συνέχεια, διαγράφονται τυχόν προϋπάρχοντα, ομώνυμα ευρετήρια του εξυπηρετητή Elasticsearch και ορίζονται από την αρχή τα νέα ευρετήρια. Κατόπιν, τοποθετούνται οι τύποι αντιστοιχίσεων[4] των λεκτικών και κατηγορημάτων στον εξυπηρετητή.

Ακολουθεί το στάδιο φόρτωσης των δεδομένων, που βρίσκονται στα αρχεία εισόδου, στον εξυπηρετητή:

- Φόρτωση των αρχείων rdftypes.p, literal_to_type_and_nodelist.p και literal_rdfs_to_type_and_nodelist.p στο ευρετήριο λεκτικών του Elasticsearch. Τα δύο τελευταία αρχεία φορτώνονται σε δύο ξεχωριστούς τύπους αντιστοίχισης.
- Φόρτωση του αρχείου predicates.p στο ευρετήριο κατηγορημάτων του Elasticsearch.

5.2 Εξυπηρετητές

5.2.1 Εξυπηρετητής GraphDB

Ο εξυπηρετητής GraphDB είναι η μηχανή παραγωγής ερωτημάτων SPARQL. Αποτελεί το βασικότερο κομμάτι της εφαρμογής, αφού αναλαμβάνει τη φόρτωση της σύνοψης RDF και τη διεκπεραίωση ερωτήσεων εύρεσης αποτελεσμάτων SPARQL. Επιπλέον, συνδέεται με τον εξυπηρετητή Elasticsearch και παρέχει τη μέθοδο getsuggestions() για την υποστήριξη αυτόματης συμπλήρωσης πεδίων εισόδου του γραφικού περιβάλλοντος χρήστη.

Ο εξυπηρετητής GraphDB αποτελείται από την κλάση GraphDB.py και το εκτελέσιμο πρόγραμμα start_graphdb.py. Στην κλάση GraphDB.py ορίζεται η λειτουργία του εξυπηρετητή και προσφέρεται διεπαφή XML-RPC για την απομακρυσμένη εκτέλεση των μεθόδων της.

Το πρόγραμμα start_graphdb.py είναι ένα εργαλείο που αναλαμβάνει την εκτέλεση του εξυπηρετητή σε περιβάλλον γραμμής εντολών. Κατά την εκκίνησή του, γίνεται έλεγχος των παραμέτρων εισόδου και αρχικοποίηση στιγμιότυπου της κλάσης GraphDB.

Τρόπος Χρήσης Η εκκίνηση του εξυπηρετητή GraphDB γίνεται με το εργαλείο start_graphdb.py, το οποίο εκτελείται από τη γραμμή εντολών με την εντολή

```
python start_graphdb.py [-h] [-i1 PREDICATE INDEX] [-i2 LITERAL INDEX] INPUT_DIR
    ES_URL GDB_URL
```

και δέχεται τις εξής παραμέτρους:

- INPUT_DIR
  Ο κατάλογος που περιέχει τη σύνοψη RDF.
- GDB_URL
  Το επιθυμητό URL του εξυπηρετητή GraphDB.
- ES_URL
  Το URL του εξυπηρετητή Elastisearch.
- i1
  Το όνομα του ευρετηρίου κατηγορημάτων του εξυπηρετητή Elastisearch, το οποίο από προεπιλογή είναι ”suggester_predicates”
- i2
  Το όνομα του ευρετηρίου λεκτικών εξυπηρετητή Elastisearch, το οποίο από προεπιλογή είναι ”suggester_literals”.
- -h, –help
  Εμφάνιση βοήθειας χρήσης προγράμματος.

[4] Βλέπε ενότητα 5.2.2.

5.2.2 Εξυπηρετητής Elasticsearch

Το Elasticsearch είναι ένας εξυπηρετητής υπηρεσιών αναζήτησης που βασίζεται στην τεχνολογία Apache Lucene. Παρέχει μια κατανεμημένη, πολυχρηστική μηχανή αναζήτησης με δυνατότητες αναζήτησης πλήρους κειμένου (full-text), καθώς και μια διεπαφή ιστού REST (Representational state transfer) για την επικοινωνία με τους χρήστες. Τα δεδομένα που δέχεται και επιστρέφει ο εξυπηρετητής είναι έγγραφα JSON. Το Elasticsearch έχει αναπτυχθεί σε γλώσσα Java και ο πηγαίος κώδικας είναι διαθέσιμος ως ανοικτό λογισμικό με άδεια Apache.

Στο Elasticsearch το *έγγραφο (document)* αποτελεί τη βασική μονάδα αναζήτησης κι ευρετηρίασης. Επιπλέον, ορίζονται τα *ευρετήρια (indexes)*, οι *τύποι αντιστοιχίσεων (mapping types)* και τα *πεδία (fields)*. Ένα ευρετήριο αποτελείται από έγγραφα, ενώ κάθε έγγραφο αποτελείται από πεδία. Οι τύποι αντιστοιχίσεων αποτελούν τρόπους διαχωρισμού των εγγράφων ενός ευρετηρίου σε λογικές ομάδες. Θα μπορούσαμε να πούμε ότι ένας τύπος αντιστοίχησης ορίζει έναν *τύπο εγγράφου*, με χαρακτηριστικά όπως το όνομα και το σύνολο των πεδίων του συγκεκριμένου τύπου εγγράφου.

Δομή Υπηρεσίας Ο εξυπηρετητής Elasticsearch χρησιμοποιείται για την υποστήριξη της λειτουργίας αυτόματης συμπλήρωσης πληκτρολόγησης του γραφικού περιβάλλοντος χρήστη. Χρησιμοποιούνται δύο ευρετήρια, ένα για τα κατηγορήματα, που ονομάζεται *ευρετήριο κατηγορημάτων* κι ένα για τα λεκτικά, που ονομάζεται *ευρετήριο λεκτικών*.

Το ευρετήριο κατηγορημάτων περιέχει έναν και μοναδικό τύπο εγγράφου ομώνυμο με το ευρετήριο. Αντίθετα το ευρετήριο λεκτικών περιέχει δύο τύπους εγγράφου: τον τύπο ”literals_all” και τον τύπο ”literals_rdfslabel”. Ο πρώτος περιλαμβάνει όλα τα λεκτικά, ενώ ο δεύτερος μόνο τα λεκτικά που συνδέονται με URIs μέσω του κατηγορήματος rdfs:label. Ο διαχωρισμός έγινε για την υποστήριξη της λειτουργικότητας σε βάσεις δεδομένων που δεν επιλέγουν μόνο το κατηγόρημα rdfs:label για τη λεκτική αναπαράσταση των URI, αλλά χρησιμοποιούν και άλλα κατηγορήματα.

Οι δύο τύποι εγγράφου του ευρετηρίου λεκτικών περιλαμβάνουν το πεδίο ”context” που επιτρέπει την εφαρμογή φίλτρου με βάση αυτό το πεδίο σε έγγραφα του ίδιου ευρετηρίου. Με κατάλληλη χρήση του πεδίου αυτού είναι δυνατή η πραγματοποίηση αναζήτησης εγγράφων συγκεκριμένου τύπου, που βρίσκονται στο ίδιο ευρετήριο.

Η χρήση ενός ευρετηρίου, με δύο τύπους εγγράφων, σε σχέση με τη χρήση δύο ευρετηρίων, με έναν τύπο εγγράφου το καθένα, ελαχιστοποιεί τον απαιτούμενο αποθηκευτικό χώρο, λόγω ευρετηρίασης κοινών εγγράφων.

Το εργαλείο esload.py δεν υποστηρίζει εξυπηρετητές Elasticsearch έκδοσης παλαιότερης από την 1.2.0, λόγω της έλλειψης υποστήριξης του πεδίου context.

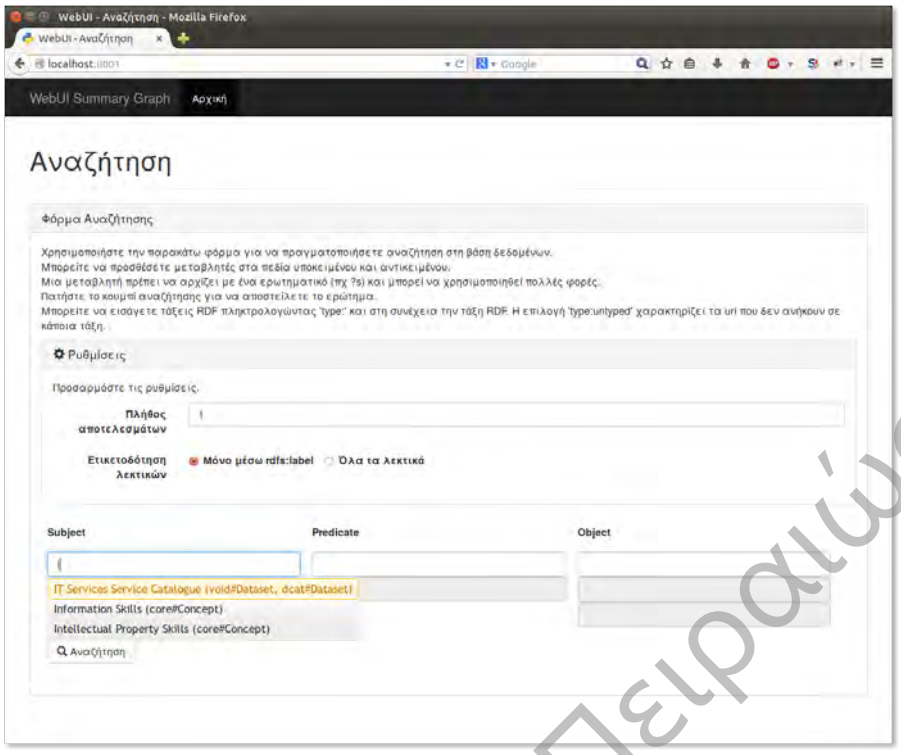
5.3 Γραφικό Περιβάλλον Χρήστη

Το γραφικό περιβάλλον βρίσκεται στον κατάλογο webUI, σε μορφή ιστοσελίδας, αποτελούμενο από αρχεία HTML, CSS και Javascript. Χρησιμοποιούνται τρεις βιβλιοθήκες Javascript: Bootstrap, D3 και jQuery.

Η βιβλιοθήκη Bootstrap 3.0 χρησιμοποιείται για τη δημιουργία ενιαίας εμφάνισης της ιστοσελίδας. Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει αρχεία HTML, CSS και προαιρετικές επεκτάσεις JavaScript, που καθορίζουν τα στοιχεία του περιβάλλοντος (μορφές τυπογραφίας, κουμπιά πλοήγησης, κλπ).

Επιπλέον, χρησιμοποιείται η βιβλιοθήκη D3 για την οπτικοποίηση γραφημάτων, η βιβλιοθήκη bPopup για τη δημιουργία παραθύρου διαλόγου εντός του φυλλομετρητή ιστού (popup) και η βιβλιοθήκη jQuery για τον χειρισμό γεγονότων εισόδου χρήστη και την προσπέλαση στοιχείων HTML.

Δομή Γραφικού Περιβάλλοντος Το γραφικό περιβάλλον αποτελείται από τη φόρμα αναζήτησης και την περιοχή αποτελεσμάτων. Η φόρμα αναζήτησης περιλαμβάνει τη φόρμα ρυθμίσεων, τα πεδία εισόδου λεκτικού και το κουμπί εκτέλεσης της αναζήτησης. Τα πεδία εισόδου λεκτικού είναι τρία: το πεδίο υποκειμένου, το πεδίο κατηγορήματος και το πεδίο αντικειμένου. Καθένα από τα πεδία προσφέρει αυτόματη συμπλήρωση λεκτικών, χρησιμοποιώντας το μητρώο δεδομένων που βρίσκεται στον εξυπηρετητή Elasticsearch (εικ. 5.6). Συγκεκριμένα, το πεδίο κατηγορήματος αντλεί δεδομένα από το ευρετήριο κατηγορημάτων, ενώ



Εικόνα 5.6: Το γραφικό περιβάλλον χρήστη. Διακρίνεται η λειτουργία της υπηρεσίας αυτόματης συμπλήρωσης λεκτικών.

τα πεδία υποκειμένου και αντικειμένου από το ευρετήριο λεκτικών.

Η φόρμα ρυθμίσεων δίνει στον χρήστη τη δυνατότητα να επιλέξει το πλήθος των αποτελεσμάτων της αναζήτησης και την επιλογή της μεθόδου ετικετοδότησης λεκτικών. Η τελευταία ρύθμιση επηρεάζει την υπηρεσία αυτόματης συμπλήρωσης των πεδίων υποκειμένου και αντικειμένου, περιορίζοντας τα λεκτικά ανάλογα με το κατηγόρημα μέσω του οποίου συνδέονται με τα αντίστοιχα URI του γραφήματος RDF. Επιπλέον, η ρύθμιση αυτή επηρεάζει και το περιεχόμενο των αποτελεσμάτων SPARQL.

Το κουμπί αναζήτησης αποστέλλει τη φόρμα αναζήτησης στον Web Server για την εκτέλεση της αναζήτησης. Κατά τη διάρκεια εκτέλεσης της αναζήτησης, εμφανίζεται κινούμενο γραφικό στοιχείο, που σκοπό έχει την ενημέρωση του χρήστη σχετικά με την εξέλιξη της αναζήτησης.

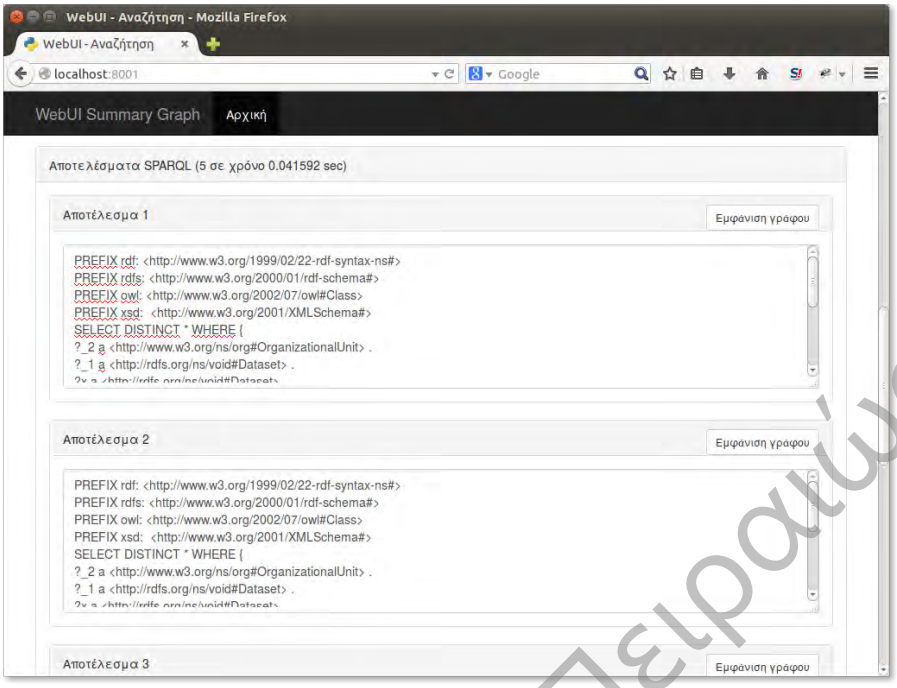
Τα αποτελέσματα της αναζήτησης εμφανίζονται στην περιοχή αποτελεσμάτων του γραφικού περιβάλλοντος (εικ. 5.7) που βρίσκεται κάτω από τη φόρμα αναζήτησης. Κάθε αποτέλεσμα SPARQL εμφανίζεται με τη μορφή κειμένου, εντός πλαισίου κειμένου και μπορεί να επιλεγεί και να αντιγραφεί. Αυτό επιτρέπει την επεξεργασία του ερωτήματος από τον χρήστη. Επιπλέον, κάθε αποτέλεσμα SPARQL συνοδεύεται από το αντίστοιχο δένδρο από το οποίο προέκυψε. Ο χρήστης μπορεί να εμφανίσει το δένδρο πατώντας το κουμπί "Εμφάνιση γράφου" (εικ. 5.8).

6 Βελτιστοποίηση

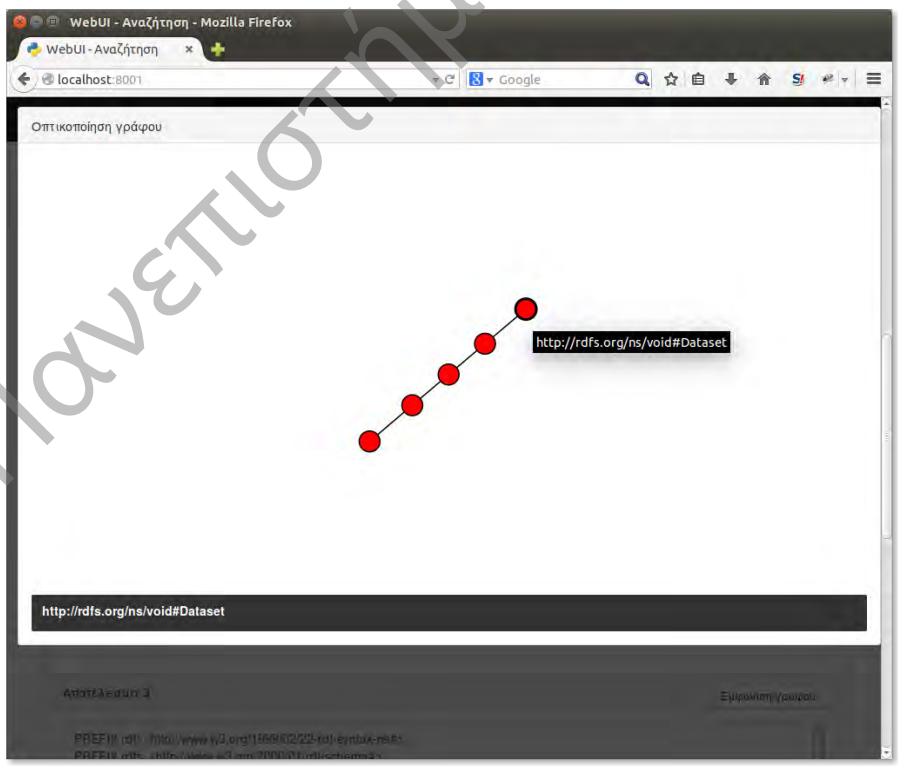
Μελετήθηκε μια μέθοδος βελτιστοποίησης της υπηρεσίας η οποία βασίζεται στην επεξεργασία της σύνοψης RDF.

Κατά τη δημιουργία της σύνοψης RDF, που αναλύθηκε στην ενότητα 3, κάθε τάξη RDF του γραφήματος RDF προστίθεται ως κόμβος στη σύνοψη. Αυτός ο κόμβος θα είναι, είτε κόμβος προέλευσης, είτε κόμβος προορισμού όλων των ακμών κατηγορημάτων που διασυνδέονται με αυτόν.

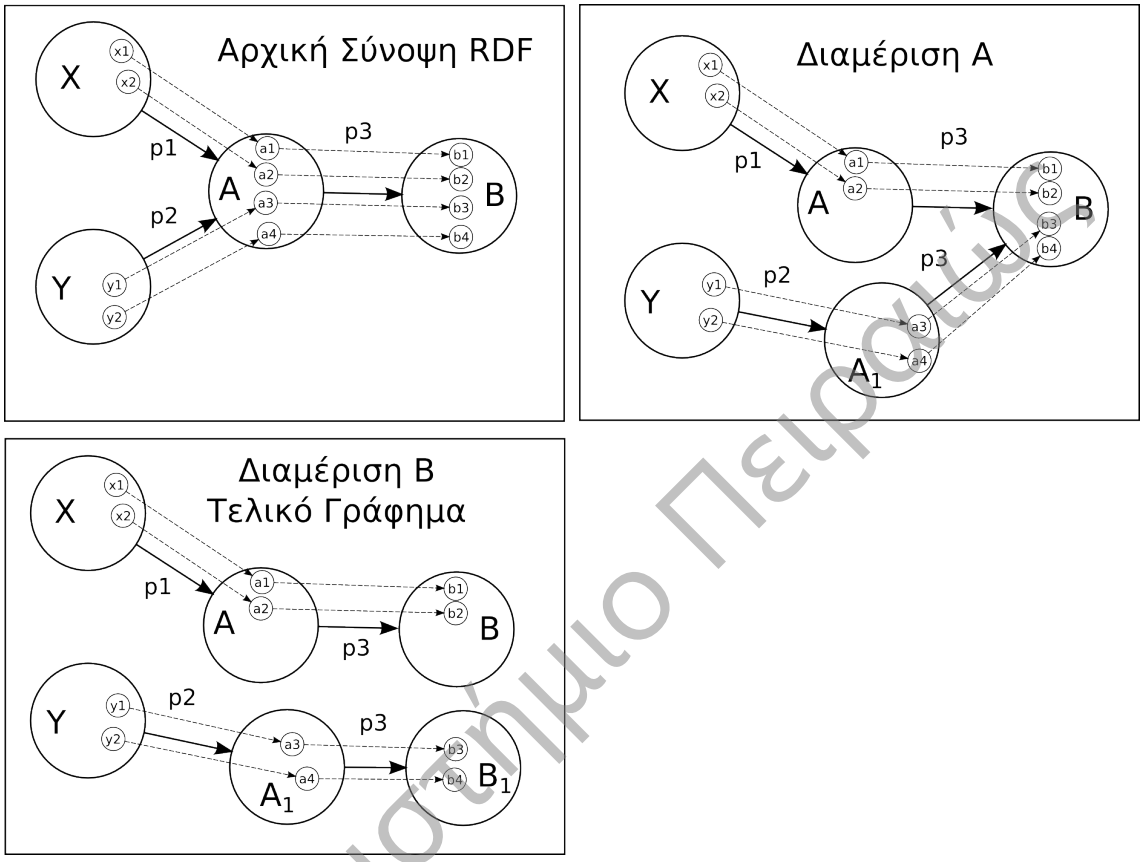
Στη σύνοψη αποτυπώνονται με ακρίβεια μόνο οι αμοιβαίες διασυνδέσεις καθενός κόμβου τύπου RDF με τους γείτονές του και όχι οι διασυνδέσεις με τους γείτονες των γειτόνων του. Με άλλα λόγια αποτυπώνονται με ακρίβεια μονοπάτια μοναδιαίου μήκους και όχι μεγαλύτερου.



Εικόνα 5.7: Αποτελέσματα αναζήτησης.



Εικόνα 5.8: Οπτικοποίηση δένδρου από το οποίο προέρχεται το αντίστοιχο αποτέλεσμα SPARQL.



x1 a X	a1 a A	b1 a B	x1 p1 a1	a1 p3 b1
x2 a X	a2 a A	b2 a B	x2 p1 a2	a2 p3 b2
y1 a Y	a3 a A	b3 a B	y1 p2 a3	a3 p3 b3
y2 a Y	a4 a A	b4 a B	y2 p2 a4	a4 p3 b4

Εικόνα 6.1: Διαμέριση κόμβων σύνοψης RDF. Στο πρώτο βήμα γίνεται διαμέριση του κόμβου A, ως προς το ζεύγος ακμών (X-A, Y-A) και δημιουργείται νέος κόμβος A_1, ίδιου τύπου RDF με τον αρχικό. Σε αυτόν ανατίθενται τα μέλη της ακμής Y-A και όλες οι ακμές που σχετίζονται με αυτά. Στο επόμενο βήμα, ελέγχεται ο κόμβος B και γίνεται διαμέριση με τον ίδιο τρόπο. Οι ακμές που δεν έχουν ετικέτα και απεικονίζονται με διακεκομμένες γραμμές είναι τριπλέτες του γραφήματος RDF, ενώ οι ακμές που απεικονίζονται με συνεχείς γραμμές και έχουν ετικέτα ανήκουν στη σύνοψη. Οι μικροί κόμβου ανήκουν στο γράφημα RDF, ενώ οι μεγάλοι κόμβοι ανήκουν στη σύνοψη. Κάθε μεγάλος κόμβος περικλείει τουλάχιστον έναν μικρό κόμβο, δηλώνοντας με αυτόν τον τρόπο ότι οι μικροί κόμβοι έχουν τύπο τον τύπο του μεγάλου κόμβου στον οποίο περικλείονται.

Για παράδειγμα, ας θεωρήσουμε ένα γράφημα RDF που αποτελείται από τις τριπλέτες της εικόνας 6.1. Η ισοδύναμη σύνοψη RDF, που απεικονίζεται πάνω αριστερά στην εικόνα, αποτυπώνει με ακρίβεια τις αμοιβαίες συνδέσεις τύπων RDF, δηλαδή τα μονοπάτια μοναδιαίου μήκους X-p1-A, Y-p2-A, A-p3-B, αφού αντιστοιχούν σε υπαρκτά μονοπάτια τριπλετών RDF εξ' ορισμού. Η σύνοψη, ωστόσο υπονοεί ότι υπάρχει μονοπάτι μήκους μεγαλύτερου της μονάδας, όπως το X-p1-A-p2-Y το οποίο συνδέει μέλη της τάξης X με μέλη της τάξης Υ. Αυτό δεν ισχύει επειδή αυτό το μονοπάτι της σύνοψης δεν αντιστοιχεί σε κάποιο υπαρκτό μονοπάτι τριπλετών του αρχικού γραφήματος RDF. Αυτό μπορεί να το διαπιστώσει κανείς κοιτάζοντας τις τριπλέτες, που απεικονίζονται με διακεκομμένες γραμμές, στην εικόνα. Με άλλα λόγια, οι τριπλέτες RDF ορίζουν δύο συνεκτικά υπογραφήματα RDF τα οποία στη σύνοψη απεικονίζονται ως ένα συνεκτικό γράφημα.

Αντίθετα, οι αρχικές τριπλέτες μπορούν να αναπαρασταθούν με ακρίβεια όπως στο γράφημα κάτω αριστερά της εικόνας, όπου τόσο ο κόμβος A, όσο και ο κόμβος B, έχουν διαμεριστεί ο καθένας σε δύο αμοιβαία αποκλειόμενα σύνολα. Σε αυτό το γράφημα δεν υπάρχει μονοπάτι ανάμεσα στους κόμβους X και Y. Και οι δύο διαμερίσεις του αρχικού κόμβου A αντιστοιχούν σε μέλη της ίδιας τάξης A, που διαφέρουν μόνο στον τρόπο σύνδεσης με άλλους κόμβους.

Η ακρίβεια του μονοπατιού είναι θεμελιώδης για την εύρεση δένδρων που εκφράζουν ακριβή ερωτήματα στο αρχικό γράφημα RDF. Για αυτόν τον λόγο, εισάγεται ένα επιπρόσθετο βήμα, μετά από τη δημιουργία της σύνοψης RDF, το οποίο ονομάζουμε διαμέριση. Ο αλγόριθμος διαμέρισης εξετάζει τα αρχικά μέλη καθενός κόμβου τύπου RDF ανάλογα με το κατηγόρημα. Πιο συγκεκριμένα, για κάθε τέτοιο κόμβο, εξετάζονται όλα τα ζεύγη διαφορετικών, προσκείμενων ακμών (δηλαδή κατηγορημάτων) για αμοιβαίο αποκλεισμό. Αμοιβαίος αποκλεισμός σημαίνει ότι κανένα μέλος αυτού του τύπου RDF δεν περιέχεται σε τριπλέτες με τα δύο συγκεκριμένα κατηγορήματα υπό εξέταση. Αν τα σύνολα είναι ξένα μεταξύ τους, ο κόμβος προκρίνεται για διχοτόμηση σε δύο νέους κόμβους.

Επιστρέφοντας στο παράδειγμα παρατηρούμε ότι στον κόμβο A της σύνοψης, εξετάζοντας το ζεύγος ακμών (X, p_1, A) και (Y, p_2, A) παρατηρούμε ότι

- το σύνολο τριπλετών RDF που αντικαθίστανται από την ακμή της σύνοψης (X, p_1, A) είναι το

  $$\{(x_1, p_1, a_1), (x_2, p_1, a_2)\}$$

  το οποίο έχει στη μεριά του κόμβου A το σύνολο μελών $\{a_1, a_2\}$,

- το σύνολο τριπλετών RDF που αντικαθίστανται από την ακμή της σύνοψης (Y, p_2, A) είναι το

  $$\{(y_1, p_2, a_3), (y_2, p_2, a_4)\}$$

  το οποίο έχει στη μεριά του κόμβου A το σύνολο μελών $\{a_3, a_4\}$

- Αυτά τα σύνολα μελών είναι ξένα μεταξύ τους, δηλαδή $\{a_1, a_2\} \bigcap \{a_3, a_4\} = \emptyset$, συνεπώς είναι αναγκαία η διχοτόμησή του.

Η ίδια διαδικασία εφαρμόζεται για κάθε κόμβο του γραφήματος, διαδοχικά.

Στην επόμενη ενότητα γίνεται εκτενέστερη ανάλυση του αλγορίθμου.

6.1 Το Πρόβλημα της Διαμέρισης

Η εξήγηση της διαμέρισης βασίζεται στις αντίστοιχες έννοιες της Θεωρίας Συνόλων:

- Δυναμοσύνολο
  Δυναμοσύνολο $\mathcal{P}(A)$ ενός συνόλου A ορίζεται το σύνολο που έχει ως στοιχεία ακριβώς τα υποσύνολα του A. Συνεπώς

  $$\mathcal{P}(A) = \{X | X \subseteq A\}$$

  και ισχύει $\emptyset \in P(A)$ και $A \in \mathcal{P}(A)$ [16].

- Διαμέριση
  Ένα υποσύνολο Φ του συνόλου $\mathcal{P}(A)$ λέγεται διαμέριση του A εάν [16, 17]:

1. $\emptyset \notin \Phi$, δηλαδή το κενό σύνολο δεν αποτελεί στοιχείο της διαμέρισης.
2. $\bigcup \Phi = A$, δηλαδή η ένωση όλων των στοιχείων της διαμέρισης του συνόλου Α ισούται με το σύνολο Α.
3. Εάν $X, Y \in \Phi$ με $X \neq Y$, τότε $X \bigcap Y = \emptyset$, δηλαδή δύο διαφορετικά στοιχεία της διαμέρισης δεν έχουν κοινά στοιχεία.

Με άλλα λόγια, μια διαμέριση ενός συνόλου A είναι ένα σύνολο μη κενών υποσυνόλων του A, τα οποία δεν έχουν κοινά στοιχεία μεταξύ τους, ενώ η ένωσή τους ισούται με το σύνολο A [17].

Έστω το γράφημα RDF $G_{RDF} = (V_{RDF}, E_{RDF}, P_{RDF}, l)$ και η σύνοψη αυτού $G = (V_S, E_S, P_S, W_S)$. Για τη μελέτη του προβλήματος διαμέρισης ορίζουμε τις παρακάτω συναρτήσεις:

- $T : E_S \rightarrow \mathcal{P}(E_2)$ η *συνάρτηση τριπλετών ακμής*, που απεικονίζει κάθε ακμή της σύνοψης σε ένα σύνολο από τριπλέτες του γραφήματος RDF, δηλαδή

$$T(e) = \{e_{RDF} | (e_{RDF}, e_s) \in R_C \wedge e_s = e\}$$

- $A : V_s \rightarrow \mathcal{P}(E_s)$ η *συνάρτηση προσκείμενων ακμών κόμβου*, που απεικονίζει κάθε κόμβο της σύνοψης σε ένα σύνολο που αποτελείται από ακμές της σύνοψης, δηλαδή

$$A(n) = A_{in}(n) \bigcup A_{in}(n)$$

όπου:

  - $A_{in} : V_s \rightarrow \mathcal{P}(E_s)$ η *συνάρτηση εισερχόμενων, προσκείμενων ακμών κόμβου*, που απεικονίζει κάθε κόμβο της σύνοψης σε ένα σύνολο που αποτελείται από ακμές της σύνοψης, δηλαδή

$$A_{in}(n) = \{(s, p, o) | (s, p, o) \in E_S \wedge o = n\}$$

  - $A_{out} : V_s \rightarrow \mathcal{P}(E_s)$ η *συνάρτηση εξερχόμενων, προσκείμενων ακμών κόμβου*, που απεικονίζει κάθε κόμβο της σύνοψης σε ένα σύνολο που αποτελείται από ακμές της σύνοψης, δηλαδή

$$A_{out}(n) = \{(s, p, o) | (s, p, o) \in E_S \wedge s = n\}$$

Τα υποσύνολα αυτά είναι ξένα μεταξύ τους, δηλαδή ισχύει $A_{in}(n) \bigcap A_{in}(n) = \emptyset$.

- $I : E_S \rightarrow \mathcal{P}(U_{RDF})$ η *συνάρτηση μελών κόμβου και ακμής*, που απεικονίζει κάθε ζεύγος (κόμβος, ακμή) της σύνοψης σε ένα σύνολο URI , δηλαδή

$$I(n, e) = \begin{cases} \{u | e = (n_s, p, n_t) \wedge (u, p, v) \in T(e)\} & , n = n_s \\ \{u | e = (n_s, p, n_t) \wedge (v, p, u) \in T(e)\} & , n = n_t \\ \{u | e = (n_s, p, n_t) \wedge [(u, p, v) \in T(e) \vee (v, p, u) \in T(e)]\} & , n = n_s \wedge n = n_t \end{cases}$$

Με άλλα λόγια, έστω κόμβος n και ακμή e, για κάθε τριπλέτα $t \in T(e)$ τις ακμής e, αν ο κόμβος n είναι κόμβος προέλευσης της ακμής e, βάζουμε το υποκείμενο της τριπλέτας στο σύνολο $I(n, e)$, επιπλέον, αν ο κόμβος n είναι κόμβος προορισμού της ακμής e, βάζουμε το αντικείμενο της τριπλέτας στο σύνολο $I(n, e)$.

- $R : E_S \rightarrow \mathcal{P}(R_C)$ η *συνάρτηση συσχετίσεων τριπλετών ακμής,* που απεικονίζει κάθε ακμή της σύνοψης σε ένα υποσύνολο του R_C (βλ. 3.2.2), δηλαδή

$$R(e) = \{(e_{RDF}, e_S) | (e_{RDF}, e_S) \in R_C \wedge e_s = e\}$$

Ένας κόμβος $n \in V_S$ της σύνοψης RDF ορίζει ένα αντίστοιχο σύνολο μελών $M(n) \subseteq U_{RDF}$, έστω

$$M(n) = \bigcup_{e \in A(n)} I(n, e)$$

όπου $A(n) \subseteq E_S$ οι προσκείμενες ακμές του κόμβου n και U_{RDF} το σύνολο των (τυποποιημένων και μη τυποποιημένων) URI του γραφήματος RDF (βλ. 3.2.1). Επιπλέον, όπως είπαμε, για κάθε προσκείμενη ακμή $e \in A(n)$ υπάρχει ένα υποσύνολο μελών $I(n,e)$ του κόμβου n που ορίζεται από τη συγκεκριμένη ακμή e. Ορίζουμε το σύνολο $S(n) \subseteq \mathcal{P}(A)$ που αποτελείται από σύνολα $I(n,e) \in S(n)$ για κάθε $e \in A(n)$, δηλαδή

$$S(n) = \{I(n,e) \,|e \in A(n)\}$$

Το γράφημα είναι κατάλληλα διαμερισμένο και βελτιστοποιημένο, αν και μόνο αν για κάθε $n \in V_S$ και για κάθε $X, Y \in S(n)$ με $X \neq Y$ ισχύει $X \bigcap Y \neq \emptyset$. Διαφορετικά θα λέμε ότι το γράφημα χρειάζεται διαμέριση. Αντίστοιχα αν ένας κόμβος n έχει $X, Y \in S(n)$ με $X \neq Y$ τέτοια ώστε $X \bigcap Y = \emptyset$, ο κόμβος αυτός χρειάζεται διαμέριση.

Η διαμέριση μπορεί να γίνει με κάποιον αλγόριθμο όπως αυτός της επόμενης υποενότητας. Ουσιαστικά, η διαμέριση ενός κόμβου, ως προς ένα ζεύγος $X, Y \in S(n)$, που αντιστοιχεί σε ένα ζεύγος ακμών, συντελείται με τουλάχιστον τρεις τρόπους:

1. με τη μεταφορά ενός εκ των X,Y σε έναν καινούριο κόμβο n_p ίδιου τύπου RDF με τον αρχικό (διχοτόμηση κόμβου)
2. με τη μεταφορά και των δύο σε δύο νέους κόμβους ίδιου τύπου RDF με τον αρχικό
3. με τη μεταφορά υποσυνόλων του X και του Y σε προϋπάρχοντες, από προηγούμενη διχοτόμηση, κόμβους.

Σε κάθε περίπτωση απαιτείται, επιπρόσθετα, η μεταφορά όλων των τριπλετών, στις οποίες αντιστοιχεί το μεταφερόμενο σύνολο μελών, σε νέα ακμή που συνδέει το νέο κόμβο με τους γειτονικούς κόμβους (του αρχικού κόμβου) στους οποίους αντιστοιχούν οι μεταφερόμενες τριπλέτες. Επίσης, πρέπει να διαγράφεται ο αρχικός κόμβος αν μετά τη μεταφορά μελών, το πλήθος μελών του μηδενίζεται, δηλαδή όταν ισχύει $M(n) = 0$.

Η πρώτη μέθοδος είναι αυτή που χρησιμοποιείται από τον αλγόριθμο που αναπτύξαμε και ο οποίος παρουσιάζεται στην επόμενη υποενότητα.

Η αποτελεσματικότητα ενός αλγορίθμου διαμέρισης, εξαρτάται από τη σχετική αύξηση του πλήθους κόμβων που προκαλεί. Ένας αποτελεσματικός αλγόριθμος, σε σχέση με κάποιον άλλο λιγότερο αποτελεσματικό, χαρακτηρίζεται από τη δυνατότητά του να προκαλεί μικρότερη αύξηση του πλήθους κόμβων. Αξίζει να σημειωθεί, ότι η διαμέριση ενέχει τον κίνδυνο εκφυλισμού της σύνοψης σε γράφημα ισόμορφο με το υπογράφημα RDF από το οποίο προήλθε. Αυτή η εκφυλιστική κατάσταση χαρακτηρίζεται από ανυπαρξία κόμβου n με πλήθος μελών $M(n) > 1$.

Όπως έχουμε αναφέρει παραπάνω, το πλήθος προσκείμενων ακμών ενός κόμβου n, ονομάζεται βαθμός του κόμβου και αναφέρεται ως $d(n)$. Αυτό το πλήθος ταυτίζεται με το πλήθος στοιχείων του συνόλου $S(n)$. Το πλήθος μοναδικών συνδυασμών δυάδων του συνόλου αυτού σχετίζεται με τον βαθμό του κόμβου και δίνεται από τη σχέση $\frac{1}{2}d(n)(d(n)-1)$. Αυτό σημαίνει πως για τον έλεγχο όλων των ζευγαριών προσκείμενων ακμών ενός κόμβου απαιτούνται ακριβώς τόσοι έλεγχοι αμοιβαίου αποκλεισμού. Γενικεύοντας, για τον έλεγχο όλου του γραφήματος απαιτούνται

$$\frac{1}{2}\sum_{n \in V_S}\left(d^2(n) - d(n)\right)$$

έλεγχοι αμοιβαίου αποκλεισμού. Αυτή είναι η καλύτερη περίπτωση, όπου το γράφημα είναι βελτιστοποιημένο και δεν υπάρχει ανάγκη διαμέρισης.

6.1.1 Αλγόριθμος Διαμέρισης

Ο αλγόριθμος διαμέρισης χρησιμοποιείται πριν τη φάση ανάθεσης βαρών στις ακμές της σύνοψης. Δέχεται στην είσοδο το γράφημα $G = (V, E, P)$ που αντιστοιχεί στη σύνοψη και εφαρμόζει τις αλλαγές επί τόπου.

Ο αλγόριθμος διαμέρισης εξετάζει κάθε κόμβο της σύνοψης με σειρά μεγαλύτερου βαθμού. Δηλαδή πρώτα εξετάζει τους κόμβους με το μεγαλύτερο πλήθος ακμών. Για κάθε κόμβο n βρίσκει όλες του τις ακμές και τις ταξινομεί κατά φθίνουσα σειρά πλήθους τριπλετών. Στη συνέχεια επιλέγει ζευγάρια διαφορετικών

ακμών, με προτεραιότητα μεγαλύτερου αθροιστικού πλήθους τριπλετών. Υπενθυμίζουμε ότι μια ακμή της σύνοψης RDF αντιστοιχεί σε ένα σύνολο από τριπλέτες με κοινό κατηγόρημα. Το σύνολο των υποκειμένων και το σύνολο των αντικειμένων αποτελούν μέλη των αντίστοιχων κόμβων που συνδέει η ακμή. Για κάθε ζευγάρι (e_j, e_k), ορίζονται δύο σύνολα, το I_j, που περιέχει τα μέλη του κόμβου n για την ακμή $e_{,j}$ και το I_k που περιέχει τα μέλη του κόμβου n για την ακμή e_k και, στη συνέχεια γίνεται έλεγχος αμοιβαίου αποκλεισμού των συνόλων αυτών. Αν αυτά είναι αμοιβαία αποκλειόμενα, δηλαδή ισχύει

$$I_j \bigcap I_k = \emptyset$$

ξεκινά η φάση διχοτόμησης του κόμβου, διαφορετικά ελέγχεται άλλο ζευγάρι ακμών.

Η διχοτόμηση του κόμβου είναι μια διαδικασία που έχει ως σκοπό τη δημιουργία ενός νέου κόμβου n_1 από τον αρχικό n. Αρχικά, δημιουργείται ένας νέος κόμβος n_1 στον οποίο ανατίθεται το μικρότερο, από τα επιμέρους, σύνολο I_j ή I_k. Στη συνέχεια, ο νέος κόμβος συνδέεται με εκείνους τους κόμβους που υπαγορεύουν οι τριπλέτες του. Οι τριπλέτες αυτές αφαιρούνται από τον αρχικό κόμβο, διατηρώντας σταθερό το συνολικό αριθμό τριπλετών στο γράφημα. Κάθε φορά που συμβαίνει μια διχοτόμηση, η διαδικασία επαναλαμβάνεται από την αρχή. Αν ελεγχθούν όλοι οι κόμβοι και όλα τα δυνατά ζευγάρια ακμών, η διαδικασία διαμέρισης τελειώνει.

Αλγόριθμος 6.1 Ο αλγόριθμος διαμέρισης.

είσοδος: Γράφημα $G = (V, E, P, W)$, συνάρτηση R_C απεικόνισης τριπλετών σε ακμές.
FINISHED_FLAG ← False
while not FINISHED_FLAG **do**
    FINISHED_FLAG ← True
    Ταξινόμησε το V κατά φθίνουσα σειρά βαθμού κόμβου.
    for each $n \in V$ **do**
        $E_n \leftarrow A(n)$ ▷ Βρες τις προσκείμενες ακμές του κόμβου n.
        Ταξινόμησε το E_n κατά φθίνουσα σειρά πλήθους τριπλετών.
        edge_combinations $= \{(x, y) \,|\, x, y \in E_n \wedge x \neq y\}$
        Ταξινόμησε το edge_combinations με προτεραιότητα μεγαλύτερου αθροιστικού πλήθους τριπλετών.
        for each $(e_j, e_k) \in$ edge_combinations **do**
            $I_j \leftarrow I(n, e_j)$ ▷ Βρες το σύνολο μελών του κόμβου n για την ακμή e_j.
            $I_k \leftarrow I(n, e_k)$ ▷ Βρες το σύνολο μελών του κόμβου n για την ακμή e_k.
            if $I_j \bigcap I_k = \emptyset$ **then** ▷ Έλεγχος αμοιβαίου αποκλεισμού.
                FINISHED_FLAG ← False ▷ Φάση διχοτόμησης κόμβου.
                $I \leftarrow \min(I_j, I_k)$
                $V \leftarrow V \bigcup \{n_1\}$ ▷ Πρόσθεσε μοναδικό κόμβο n_1 στο γράφημα.

                for each $(n_s, p, n_t) \in E_n$ **do** ▷ Για κάθε προσκείμενη ακμή του n.
                    $e_i \leftarrow (n_s, p, n_t)$
                    $T_i \leftarrow T(e_i)$ ▷ Βρες, από το R_C, το σύνολο τριπλετών της ακμής e_i.
                    if $n_s = n_t$ **then** ▷ Αν η ακμή είναι αυτοπαθής.
                        $T_{both} \leftarrow \{(s, p, o) \,|\, (s, p, o) \in T_i \wedge s, o \in I\}$
                        $T_{subj} \leftarrow \{(s, p, o) \,|\, (s, p, o) \in T_i \wedge s \in I \wedge o \notin I\}$
                        $T_{obj} \leftarrow \{(s, p, o) \,|\, (s, p, o) \in T_i \wedge o \in I \wedge s \notin I\}$
                        $T_{none} \leftarrow \{(s, p, o) \,|\, (s, p, o) \in T_i \wedge s, o \notin I\}$
                        if $T_{none} \neq \emptyset$ **then**
                            $R_C \leftarrow (R_C - R(e_i)) \bigcup (\{e_i\} \times T_{none})$ ▷ Διαγραφή όλων των τριπλετών της ακμής e_i, εκτός των T_{none}.
                        else
                            $E \leftarrow E \bigcup E - \{e_i\}$ ▷ Διαγραφή της ακμής e_i αν δεν αντιστοιχεί σε τριπλέτες.
                        if $T_{both} \neq \emptyset$ **then**
                            $e_1 \leftarrow (n_1, p, n_1)$

33: $E \leftarrow E \bigcup \{e_1\}$ ▷ Πρόσθεσε τη νέα ακμή στο γράφημα.
34: $R_C \leftarrow R_C \bigcup (\{e_1\} \times T_{both})$ ▷ Προσθήκη των τριπλετών T_{both} στη νέα ακμή e_1.
35: **if** $T_{subj} \neq \emptyset$ **then**
36: $e_1 \leftarrow (n_1, p, n)$
37: $E \leftarrow E \bigcup \{e_1\}$ ▷ Πρόσθεσε τη νέα ακμή στο γράφημα.
38: $R_C \leftarrow R_C \bigcup (\{e_1\} \times T_{subj})$ ▷ Προσθήκη των τριπλετών T_{subj} στη νέα ακμή e_1.
39: **if** $T_{obj} \neq \emptyset$ **then**
40: $e_1 \leftarrow (n, p, n_1)$
41: $E \leftarrow E \bigcup \{e_1\}$ ▷ Πρόσθεσε τη νέα ακμή στο γράφημα.
42: $R_C \leftarrow R_C \bigcup (\{e_1\} \times T_{obj})$ ▷ Προσθήκη των τριπλετών T_{obj} στη νέα ακμή e_1.
43: **else** ▷ Αν η ακμή δεν είναι αυτοπαθής.
44: **if** $n_s = n$ **then** ▷ Εύρεση κόμβων προορισμού και προέλευσης.
45: $n_{s1} = n_1$
46: $n_{t1} = n_t$
47: **else**
48: $n_{s1} = n_s$
49: $n_{t1} = n_1$
50: $T_c \leftarrow \emptyset$ ▷ Οι κοινές τριπλέτες.
51: **for** $(s, p, o) \in T$ **do**
52: **if** $s \in I$ **then**
53: $T_c \leftarrow T_c \bigcup \{(s, p, o)\}$
54: **if** $T_c \neq \emptyset$ **then**
55: $e_1 \leftarrow (n_{s1}, p, n_{t1})$
56: $E \leftarrow E \bigcup \{e_1\}$ ▷ Πρόσθεσε τη νέα ακμή στο γράφημα.
57: $R_C \leftarrow R_C \bigcup (\{e_1\} \times T_c)$ ▷ Προσθήκη των τριπλετών T_c στη νέα ακμή e_1.
58: $R_C \leftarrow R_C - (\{e_i\} \times T_c)$ ▷ Διαγραφή των τριπλετών T_c από την προηγούμενη ακμή e_i.
59: **if** $R(e_i) = \emptyset$ **then**
60: $E \leftarrow E \bigcup E - \{e_i\}$ ▷ Διαγραφή της ακμής e_i αν δεν αντιστοιχεί σε τριπλέτες.

6.2 Σύμπτυξη Συνόλου Κόμβων

Θεωρούμε ότι ο χρήστης εκφράζει τις πληροφοριακές του ανάγκες σχηματίζοντας προτάσεις "υποκείμενο κατηγόρημα αντικείμενο", όπου το κατηγόρημα χαρακτηρίζει άμεσα, είτε το υποκείμενο, είτε το αντικείμενο, είτε και τα δύο μαζί.

Κάθε λεκτικό που επιλέγει ο χρήστης αντιστοιχεί σε ένα σύνολο κόμβων της σύνοψης. Η χρησιμότητα της διαδικασίας σύμπτυξης είναι να προκύψει μοναδικός κόμβος από το πεδίο υποκειμένου και μοναδικός κόμβος από το πεδίο αντικειμένου και να ελαχιστοποιηθούν οι δυνατοί συνδυασμοί τερματικών κόμβων.

Έστω γράφημα $G = (V, E, P, W)$. Σκοπός της διαδικασίας σύμπτυξης (merging) ενός συνόλου κόμβων $K \subseteq V$ είναι η παραγωγή ενός γραφήματος G' όπου το K έχει αντικατασταθεί από έναν συμπτυγμένο κόμβο M. Οι προσκείμενες ακμές των κόμβων μελών του συνόλου K, έστω $A \subseteq E$, αντικαθίστανται από ακμές που πρόσκεινται στον κόμβο M.

Περιγραφή Αλγορίθμου Ο αλγόριθμος σύμπτυξης (βλ. αλγ. 6.2) δέχεται στην είσοδο το αρχικό γράφημα $G = (V, E, P, W)$ και το σύνολο κόμβων σύμπτυξης K. Αρχικά δημιουργείται κενό γράφημα εξόδου $G_o = (V_o, E_o, P_o, W_o)$ και ορίζεται το όνομα του συμπτυγμένου κόμβου M. Το M μπορεί να είναι ένα οποιοδήποτε στοιχείο του K. Για κάθε ακμή E, με βάρος w, του γραφήματος εισόδου, γίνεται έλεγχος αν οι κόμβοι προέλευσης, ή και προορισμού, ανήκουν στο σύνολο K, οπότε και παίρνουν το όνομα M. Κατόπιν, ορίζεται η νέα μετονομασμένη ακμή, που διατηρεί το ίδιο κατηγόρημα. Αν η ακμή δεν υπάρχει

στο γράφημα εξόδου, εισάγεται σε αυτό διατηρώντας το βάρος w, διαφορετικά προστίθεται το βάρος w στο προϋπάρχον βάρος της ακμής.

Είναι φανερό ότι ο συγκεκριμένος αλγόριθμος μπορεί να χρησιμοποιηθεί διαδοχικά για τη σύμπτυξη πολλαπλών συνόλων κόμβων $K_1, K_2, \ldots, K_n \subseteq V$. Αυτή η δυνατότητα αξιοποιείται για τη διαδοχική σύμπτυξη του συνόλου κόμβων του πεδίου υποκειμένου και του πεδίου αντικειμένου του γραφικού περιβάλλοντος χρήστη, εφόσον απαιτείται. Χαρακτηριστική είναι η περίπτωση επιλογής, από τον χρήστη, λεκτικού που αντιστοιχεί σε τάξη RDF και όχι σε URI. Σε αυτήν την περίπτωση γίνεται σύμπτυξη όλων των κόμβων της σύνοψης που αντιστοιχούν στην επιλεγμένη τάξη RDF. Αξίζει να σημειωθεί ότι αυτοί οι κόμβοι προέκυψαν από την διαδικασία διαμέρισης της αρχικής σύνοψης RDF.

Αλγόριθμος 6.2 Ο αλγόριθμος σύμπτυξης κόμβων.

είσοδος: Γράφημα $G = (V, E, P, W)$, σύνολο κόμβων σύμπτυξης K.
έξοδος: Γράφημα $G_o = (V_o, E_o, P_o, W_o)$.
Αρχικοποίηση κενού γραφήματος εξόδου $G_o = (V_o, E_o, P_o, W_o)$
Έστω M ένα οποιοδήποτε στοιχείο του K.
for $(s, p, t) \in E$ **do**
    $w \leftarrow$ το βάρος της ακμής (s, p, t) από το W
    if $s \in K$ **then**
        $s \leftarrow M$
    else
        $s \leftarrow s$
    if $t \in K$ **then**
        $t_o \leftarrow M$
    else
        $t_o \leftarrow t$
    $e_o \leftarrow (s_o, p, t_o)$
    if $e_o \in E_o$ **then**
        $w_o \leftarrow$ το βάρος της ακμής e_o από το W_o
        όρισε το βάρος της υπάρχουσας ακμής e_o του E_o να είναι $w_o + w$
    else
        πρόσθεσε στο E_o την ακμή e_o με βάρος ακμής w

7 Συμπεράσματα

Σε αυτήν την εργασία μελετήθηκε η μέθοδος παραγωγής της σύνοψης ενός γραφήματος RDF και ο τρόπος αξιοποίησής της για την κάλυψη των πληροφοριακών αναγκών του τελικού χρήστη. Η σύνοψη RDF αποτέλεσε τη βάση πάνω στην οποία αναπτύχθηκε μια ενδιάμεση πλατφόρμα αναζήτησης πληροφοριών σε αποθήκες RDF. Πιο συγκεκριμένα, περιγράφηκε ο τρόπος παραγωγής ερωτημάτων SPARQL από λεκτικά επιλεγόμενα από τον χρήστη. Επίσης, περιγράφηκε ο τρόπος οργάνωσης, αποθήκευσης και παρουσίασης αυτών των λεκτικών στον χρήστη.

Ακολουθούν μερικές ιδέες για μελλοντική επέκταση της λειτουργικότητας της σύνοψης.

Επίλυση Προβλήματος Εμφάνισης Πανομοιότυπων SPARQL Αποτελεσμάτων Έστω ότι, χρησιμοποιώντας τη βάση δεδομένων του Southampton ο χρήστης επιλέγει συγκεκριμένο κατηγόρημα, το void#subset, δίνει μεταβλητές στα υπόλοιπα πεδία κι επιλέγει να επιστραφούν 4 αποτελέσματα. Τα αποτελέσματα SPARQL που επιστρέφονται είναι όλα ίδια, παρόλο που προέρχονται από διαφορετικά δένδρα. Το πρόβλημα εντοπίζεται στο γεγονός ότι οι κόμβοι αυτών των γραφημάτων, που επιλέχθηκαν από τη συνάρτηση εύρεσης δένδρων Steiner (αλγ. 4.1), αποτελούν διαμερίσεις κόμβων του ίδιου τύπου RDF και συνδέονται μεταξύ τους με τα ίδια κατηγορήματα. Αυτό έχει ως αποτέλεσμα τη μετατροπή των γραφημάτων σε πανομοιότυπα SPARQL ερωτήματα. Σε αυτό το παράδειγμα το ποσοστό πανομοιότυπων ερωτημάτων είναι 3/4 ή 75%. Το ποσοστό εξαρτάται από το πλήθος αποτελεσμάτων. Στο συγκεκριμένο παράδειγμα, αν ο χρήστης επιλέξει 10 αποτελέσματα κι εκτελέσει την ίδια αναζήτηση, θα επιστραφούν περισσότερα πανομοιότυπα ερωτήματα, σε ποσοστό 80%. Αν επιλέξει 20 αποτελέσματα το ποσοστό πανομοιότυπων ερωτημάτων γίνεται 90%, και

με 100 αποτελέσματα 78%.

Θα πρέπει να διερευνηθεί η δυνατότητα επίλυσης αυτού του προβλήματος με την προσθήκη κατάλληλου περιορισμού στη συνάρτηση εύρεσης δένδρων Steiner, ώστε να αποκλείονται δένδρα που αντιστοιχούν σε πανομοιότυπα ερωτήματα SPARQL. Επιπλέον, θα πρέπει να βελτιωθεί ο συντονισμός μεταξύ των σταδίων επιλογής συνδυασμών κόμβων και εύρεσης γραφημάτων, ίσως με τη συγχώνευση των δύο σταδίων σε ένα ενιαίο στάδιο, όπου θα γίνεται χρήση κοινών κριτηρίων επιλογής κόμβων και τελικών γράφων.

Υποστήριξη Πολλαπλών Τριάδων Στο μέλλον σκοπεύουμε να επεκτείνουμε τη λειτουργικότητα της εφαρμογής με την υποστήριξη πολλαπλών τριάδων πεδίων εισόδου στο γραφικό περιβάλλον χρήστη. Ο χρήστης θα μπορεί να προσθαφαιρεί τριάδες πεδίων εισόδου στο γραφικό περιβάλλον, σχηματίζοντας σύνθετα ερωτήματα. Επιπλέον, Θα του δίνεται η δυνατότητα ταύτισης μεταβλητών διαφορετικών τριάδων, με αυτόματη συμπλήρωση της μεταβλητής των αντίστοιχων πεδίων. Πιο συγκεκριμένα, αν ο χρήστης πληκτρολογήσει στο πεδίο υποκειμένου ή αντικειμένου μιας τριάδας τη μεταβλητή ?x, να μπορεί να γίνει αυτόματη συμπλήρωση της ίδιας μεταβλητής στο πεδίο υποκειμένου ή αντικειμένου οποιασδήποτε άλλης τριάδας.

Βιβλιογραφία

[1] V.K. Balakrishnan. *Schaum's Outline of Graph Theory: Including Hundreds of Solved Problems*. Schaum's Outline Series. McGraw-Hill Education, 1997. ISBN: 9780070054899. URL: http://books.google.gr/books?id=1NTPbSehvWsC.

[2] Tim Berners-Lee. "The next web". In: Feb. 2009.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (May 2001), pp. 34–43. URL: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[4] Bolin Ding et al. "Finding Top-k Min-Cost Connected Trees in Databases". In: *ICDE*. 2007, pp. 836–845.

[5] Gang Gou. "Efficient Algorithms for Querying Large-scale Data in Relational, Xml, and Graph-structured Data Repositories". AAI3329251. PhD thesis. 2008. ISBN: 978-0-549-82091-8.

[6] Honza Král. *Python Elasticsearch Client — Elasticsearch 1.2.0 documentation*. 2014. URL: http://elasticsearch-py.readthedocs.org (visited on 09/27/2014).

[7] Gabriele Reich and Peter Widmayer. "Beyond Steiner's Problem: A VLSI Oriented Generalization." In: *WG*. Ed. by Manfred Nagl. Vol. 411. Lecture Notes in Computer Science. Springer, Oct. 5, 2009, pp. 196–210. ISBN: 3-540-52292-1. URL: http://dblp.uni-trier.de/db/conf/wg/wg89.html#ReichW89.

[8] Alan Ruttenberg et al. "Life sciences on the Semantic Web: the Neurocommons and beyond". In: *Briefings in Bioinformatics* 10.2 (2009), pp. 193–204. DOI: 10.1093/bib/bbp004. eprint: http://bib.oxfordjournals.org/content/10/2/193.full.pdf+html. URL: http://bib.oxfordjournals.org/content/10/2/193.abstract.

[9] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. "The Semantic Web Revisited". In: *IEEE Intelligent Systems* 21.3 (2006), pp. 96–101. DOI: http://doi.ieeecomputersociety.org/10.1109/MIS.2006.62.

[10] Steven S. Skiena. *The Algorithm Design Manual*. 2nd. Springer Publishing Company, Incorporated, 2008. ISBN: 1848000693, 9781848000698.

[11] W3C. *RDF 1.1 Concepts and Abstract Syntax*. 2014. URL: http://www.w3.org/TR/rdf11-concepts (visited on 09/24/2014).

[12] W3C. *RDF 1.1 XML Syntax*. 2014. URL: http://www.w3.org/TR/rdf-syntax-grammar (visited on 09/24/2014).

[13] W3C. *SPARQL 1.1 Query Language*. 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (visited on 09/24/2014).

[14] J.Y. Yen. "Finding the k shortest loopless paths in a network". In: *management Science* (1971), pp. 712–716.

[15] Jeffrey Xu Yu, Lu Qin, and Lijun Chang. "Keyword Search in Relational Databases: A Survey". In: *IEEE Data Eng. Bull.* 33.1 (2010), pp. 67–78.

[16] Στυλιανός Α. Ανδρεαδάκης. *Γραμμική Άλγεβρα*. Συμμετρία, 1991.

[17] Δ. Βάρσος κ.α. *Εισαγωγή στην Γραμμική Άλγεβρα*. Vol. Α. Σοφία, 2003. URL: http://math.uoa.gr/algebra/GAI.pdf.

[18] Π. ΤΣΙΚΟΥΡΑΣ. "Σημειώσεις για το μάθημα ΕΦΑΡΜΟΓΕΣ ΘΕΩΡΙΑΣ ΓΡΑΦΗΜΑΤΩΝ". 2013.

Ευρετήριο

Γλωσσάρι

blank node	Κενός κόμβος - *σελ. 8*
composite node	Σύνθετος κόμβος - *σελ. 15*
datatype	Τύπος δεδομένων - *σελ. 8*
endpoint	Τελικό σημείο - *σελ. 11*
Graphical User Interface	Γραφικό περιβάλλον χρήστη - *σελ. 13*
index	Ευρετήριο - *σελ. 39*
IRI	Διεθνοποιημένο Αναγνωριστικό Πόρων - *σελ. 7*
keyword	Λέξι κλειδί - *σελ. 25*
linked data	Διασυνδεδεμένα δεδομένα - *σελ. 12*
list	Λίστα - *σελ. 32*
literal	Λεκτικό - *σελ. 8*
mapping type	Τύπος αντιστοίχησης - *σελ. 39*
merging	Σύμπτυξη - *σελ. 47*
object	Αντικείμενο - *σελ. 8*
partitioning	Διαμέριση - *σελ. 36*
plain literal	Απλό λεκτικό - *σελ. 8*
predicate	Κατηγόρημα - *σελ. 8*
prefixed name	Προθεματισμένο όνομα - *σελ. 10*
priority queue	Ουρά προτεραιότητας - *σελ. 27*
query	Ερώτημα - *σελ. 10*
query pattern	Μοτίβο ερωτήματος - *σελ. 10*
RDF	Πλαίσιο Περιγραφής Πόρων - *σελ. 8*
RDF dataset	Σύνολο γραφημάτων RDF - *σελ. 10*
RDF type	Τύπος RDF - *σελ. 12*
RDF type nodes	Κόμβοι τύπων RDF της σύνοψης RDF - *σελ. 15*
result clause	Τύπος ερωτήματος και αποτελεσμάτων - *σελ. 10*
simple node	Απλός κόμβος - *σελ. 15*
solution modifier	Διαμορφωτής αποτελέσματος - *σελ. 10*
SPARQL endpoint	Τελικό σημείο SPARQL - *σελ. 10*
subject	Υποκείμενο - *σελ. 8*
triple	Τριπλέτα - *σελ. 8*
triple pattern	Μοτίβο επιλογής τριπλετών - *σελ. 10*
tuples	Πλειάδες - *σελ. 32*

typed literal	Τυποποιημένο λεκτικό - *σελ. 8*
untyped nodes	Μη τυποποιημένοι κόμβοι της σύνοψης RDF - *σελ. 15*
URI	Ενιαίο Αναγνωριστικό Πόρων - *σελ. 7*
weighting	Ανάθεση βαρών - *σελ. 36*