ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ολοκλήρωση Διασυνδεδεμένων Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΙΩΑΝΝΗ ΚΑΝΑΚΑΚΗ

Επιβλέπων : Βασιλείου Ιωάννης
Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2013

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ολοκλήρωση Διασυνδεδεμένων Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΙΩΑΝΝΗ ΚΑΝΑΚΑΚΗ

Επιβλέπων : Βασιλείου Ιωάννης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Δεκεμβρίου 2013.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Βασιλείου Ιωάννης	Κοντογιάννης Κώστας	Παπασπύρου Νικόλαος
Καθηγητής Ε.Μ.Π.	Αναπλ. Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2013

(Υπογραφή)

.....

ΙΩΑΝΝΗΣ ΚΑΝΑΚΑΚΗΣ

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

Περίληψη

Ο όρος *Διασυνδεδεμένα Δεδομένα (Linked Data)* χρησιμοποιείται για να περιγράψει σύνολα δεδομένων που έχουν δομηθεί, δημοσιευθεί και συνδεθεί σύμφωνα με τους κανόνες που όρισε πρώτος ο Tim Berners-Lee. Τα διασυνδεδεμένα δεδομένα αποτελούν σημαντική πλευρά της εξέλιξης του *Ιστού Δεδομένων (Web of Data).* Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας web-based εφαρμογής που συλλέγει ανομοιογενή δεδομένα από ποικίλες πηγές και τα εντάσσει σε χώρους δεδομένων. Η διαδικασία ολοκλήρωσης περιλαμβάνει τέσσερα κύρια βήματα: συλλογή δεδομένων, μετασχηματισμό σχήματος, αναγνώριση όμοιων οντοτήτων, αποθήκευση των δεδομένων και εκτέλεση ερωτημάτων SPARQL στο σχηματισμένο χώρο δεδομένων. Η εφαρμογή μετατρέπει τα δεδομένα που δεν βρίσκονται σε RDF μορφή σε RDF χρησιμοποιώντας έναν αλγόριθμο που αγνοεί το σημασιολογικό περιεχόμενο, μεταφράζει διαφορετικά σχήματα σε ένα ενιαίο τοπικό σχήμα και συνδέει όμοιες οντότητες. Βασίζεται σε ένα απλό περιβάλλον διεπαφής χρήστη και στα ανοικτού κώδικα εργαλεία R2R Framework και Silk Framework. Τα εισηγμένα δεδομένα σχηματίζουν διακριτά σύνολα δεδομένων που διαμορφώνουν ένα χώρο δεδομένων (dataspace) στον οποίο εφαρμόζονται SPARQL ερωτήματα. Η κύρια συνεισφορά της εφαρμογής είναι η δυνατότητα των χρηστών - με μια σειρά απλών βημάτων - να συνδυάζουν ετερογενή δεδομένα και να εξάγουν χρήσιμες πληροφορίες από αυτά.

Λέξεις Κλειδιά: Ολοκλήρωση Δεδομένων, Διασυνδεδεμένα Δεδομένα

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Linked Data is a term used for describing concrete datasets which have been formatted, exposed, published and connected according to the principles that Tim Berners-Lee first defined. *Linked Data* is a main aspect of the *Web of Data* evolution. The aim of the thesis is the development of a web-based application which collects different pieces of heterogeneous data from various sources and integrates them. The integration process includes four major steps: data collection, schema transformation, entity – resolution, data storage and SPARQL querying over the dataspace. The application converts non-RDF data to RDF using a content unaware algorithm, translates different schemas into a single local schema and links same entities. It is based on a simple user interface and the open source R2R and Silk Frameworks. The imported pieces of data form different datasets that shape a single data space to which SPARQL queries can be executed. The main contribution of the application is that users are able to combine - with little effort - pieces of heterogeneous data and extract useful information from them.

Keywords: *Linked Data, Data Integration, schema translation, entity-resolution, web-application, REST*

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Διασυνδεδεμένα Δεδομένα

1.1.1 Βασικές Αρχές

Ο όρος *Linked Data* προσδιορίζει τις βασικές αρχές που διέπουν τη δημοσίευση και τη σύνδεση δομημένων δεδομένων στο διαδίκτυο. Οι αρχές εισήχθησαν από τον Tim Berners-Lee (Berners-Lee, 2006) και συνοψίζονται στα ακόλουθα σημεία:

- Χρήση URIs ως αναγνωριστικά για τα αντικείμενα του φυσικού κόσμου[1]
- Χρήση HTTP URIs ως αναγνωριστικά με στόχο την εύκολη αναζήτησή τους από τον άνθρωπο
- Παροχή χρήσιμης πληροφορίας βάσει προτύπων (RDF,SPARQL) όταν κάποιος αναζητά ένα URI
- Σύνδεση μεταξύ των URIs ώστε να υπάρχει δυνατότητα μετάβασης και, συνακόλουθα, εύρεσης νέας πληροφορίας

Η κυρίαρχη ιδέα των Διασυνδεδεμένων Δεδομένων είναι η εφαρμογή της υπάρχουσας αρχιτεκτονικής του Διαδικτύου στο διαμοιρασμό δομημένων δεδομένων σε

[1] *Ο όρος «αντικείμενο» δεν αναφέρεται μόνο σε υλικά αντικείμενα αλλά και σε ανθρώπους, οργανισμούς, έννοιες κλπ.*

παγκόσμια κλίμακα. Η σημερινή μορφή του Διαδικτύου συνοπτικά βασίζεται στην έννοια των URIs (Uniform Resource Identifiers) ως παγκόσμιο μηχανισμό μοναδικού προσδιορισμού αντικειμένων, στο πρωτόκολλο HTTP (Hypertext Transfer Protocol) ως παγκόσμιο μηχανισμό πρόσβασης σε διαδικτυακά έγγραφα και στη γλώσσα HTML ως κυρίαρχο πρότυπο παρουσίασης των δεδομένων. Ακόμη, επικρατεί η ιδέα της σύνδεσης των διαδικτυακών εγγράφων – αρχείων που μπορεί να αποθηκεύονται σε διαφορετικές τοποθεσίες (servers). Οι σύνδεσμοι ανάμεσα στα έγγραφα μετατρέπουν το διάσπαρτο περιεχόμενο σε έναν παγκόσμιο χώρο πληροφοριών (Global Information Space).

Όπως προαναφέρθηκε, τα URIs δεν χρησιμοποιούνται για προσδιορισμό μόνο ψηφιακού περιεχομένου αλλά και ως αναγνωριστικά ανθρώπων, τοποθεσιών ή ακόμη και αφηρημένων εννοιών. Έτσι, είναι δυνατό να προσδιοριστεί ένας άνθρωπος με κάποιο URI (π.χ. ο άνθρωπος με όνομα Scott Miller (Heath & Bizer, 2011, p. 9) προσδιορίζεται με το URI `http://biglynx.co.uk/people/scott-miller`). Επίσης, η έννοια «γνωρίζω κάποιον» μπορεί να προσδιοριστεί με το URI `http://xmlns.com/foaf/0.1/knows`.

Το πρωτόκολλο HTTP αποτελεί τον ευρέως διαδεδομένο μηχανισμό πρόσβασης σε διαδικτυακό περιεχόμενο. Στην υπάρχουσα μορφή του Διαδικτύου χρησιμοποιούνται τα HTTP URIs προκειμένου να προσδιοριστεί μοναδικά κάθε μορφής διαθέσιμη πληροφορία. Επομένως, τα Διασυνδεδεμένα Δεδομένα ενθαρρύνουν τη χρήση των HTTP URIs με στόχο τον προσδιορισμό των οντοτήτων του φυσικού κόσμου. Μάλιστα, το πρωτόκολλο HTTP κάνει εφικτή την αναζήτηση αυτών των οντοτήτων.

Η τρίτη αρχή των Διασυνδεδεμένων Δεδομένων απαιτεί την αναπαράσταση της δομημένης πληροφορίας σε μια κοινή μορφή. Η μορφή αυτή συνήθως είναι η *πλατφόρμα περιγραφής πόρων (Resource Description Framework – RDF)* που αποτελεί ένα μοντέλο βασισμένο στη λογική των γράφων (Klyne & J. Carroll, 2004).

Η σύνδεση των ψηφιακών δεδομένων αποτελεί την τέταρτη αρχή των Διασυνδεδεμένων Δεδομένων. Έτσι, η γενική κατεύθυνση είναι όχι μόνο η σύνδεση ανάμεσα σε διαδικτυακά έγγραφα αλλά και μεταξύ διαφόρων ειδών αντικειμένων. Σε αντίθεση με τις υπάρχουσες αδόμητες και χωρίς σημασιολογικό περιεχόμενο διαδικτυακές συνδέσεις, οι σύνδεσμοι μεταξύ οντοτήτων που προσδιορίζονται από URIs αποκτούν σημασία. Αυτό σημαίνει πως οι σύνδεσμοι μπορούν να περιγράψουν σχέσεις του φυσικού κόσμου. Για παράδειγμα, είναι δυνατή η σύνδεση ενός ανθρώπου και κάποιου οργανισμού προκειμένου να περιγραφεί η σχέση εργασίας ανάμεσα στον άνθρωπο και στον οργανισμό αυτό. Οι σύνδεσμοι στα Διασυνδεδεμένα Δεδομένα ονομάζονται *RDF σύνδεσμοι* προκειμένου να διαχωριστούν από τους συνδέσμους ανάμεσα στα διαδικτυακά έγγραφα. Επομένως, σε αντιστοιχία με τον παγκόσμιο χώρο πληροφοριών του σημερινού Διαδικτύου οδηγούμαστε στον *παγκόσμιο χώρο δεδομένων (Global Data Space)*.

1.1.2 Ιστός Δεδομένων – Web of Data

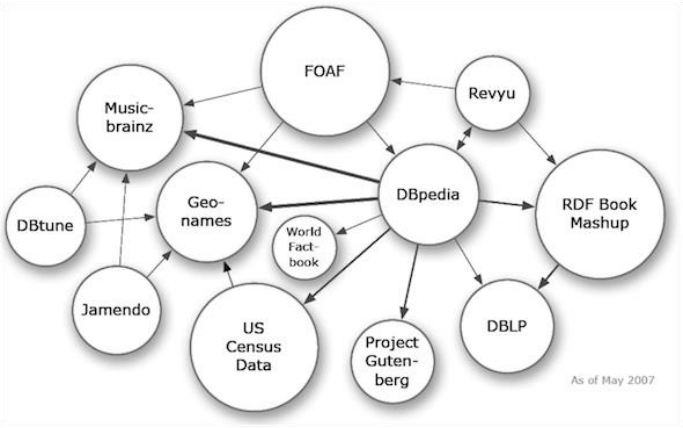
Σήμερα, πολλοί επίσημοι οργανισμοί αλλά και ανεξάρτητα άτομα έχουν αρχίσει να δημοσιοποιούν στο Διαδίκτυο Διασυνδεδεμένα Δεδομένα προσβάσιμα από όλους. Τα δεδομένα αυτά συνιστούν τον *Ιστό Δεδομένων*. Ο Ιστός Δεδομένων σχηματίζει έναν *παγκόσμιο γράφο δεδομένων* που περιλαμβάνει δισεκατομμύρια εγγραφές RDF από ποικίλες πηγές δεδομένων που καλύπτουν όλο το φάσμα πληροφοριών της ανθρώπινης ζωής: γεωγραφικές τοποθεσίες, ανθρώπους, εταιρίες και οργανισμούς, βιβλία, ταινίες, μουσική, γενετικές πληροφορίες και φάρμακα, στατιστικά δεδομένα κ.α..

Ο Ιστός Δεδομένων μπορούμε να πούμε ότι ενθυλακώνεται στο σημερινό Διαδίκτυο και παρουσιάζει ορισμένες κοινές ιδιότητες με αυτό:

- Ο Ιστός Δεδομένων μπορεί να περιέχει οποιαδήποτε πληροφορία
- Ο καθένας μπορεί να δημοσιεύσει δεδομένα σε αυτόν
- Είναι πολύ πιθανή η ύπαρξη αντικρουόμενων πληροφοριών για την ίδια οντότητα του πραγματικού κόσμου
- Οι οντότητες που αναπαρίστανται με URIs συνδέονται με τους συνδέσμους RDF κι έτσι σχηματίζονται μεγάλα σύνολα δεδομένων. Έτσι, είναι δυνατή η εύρεση νέων συνδέσεων και ,τελικά, η πρόσβαση σε μεγαλύτερο όγκο πληροφοριών που δεν ήταν γνωστός εξ' αρχής.
- Οι εκδότες δεδομένων δεν περιορίζονται στον τρόπο με τον οποίο θα δημοσιεύσουν τα δεδομένα που θέλουν
- Η χρήση του πρωτοκόλλου HTTP ως μηχανισμού πρόσβασης και του μοντέλου RDF για περιγραφή των δεδομένων απλοποιεί την πρόσβαση στα δεδομένα σε σύγκριση με τα υπάρχοντα APIs (Abstract Programming Interfaces) που βασίζονται σε ετερογενή μοντέλα δεδομένων και τρόπους πρόσβασης.

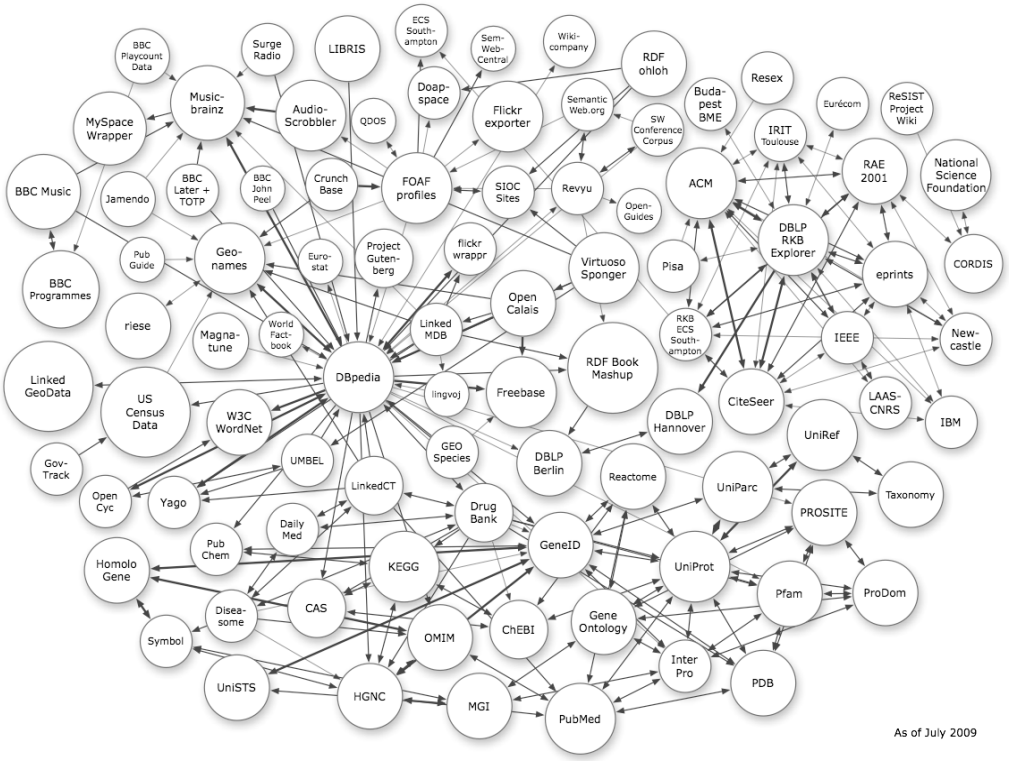
Ο Ιστός Δεδομένων έχει τις ρίζες του στις προσπάθειες της ερευνητικής κοινότητας του Σημασιολογικού Ιστού και συγκεκριμένα στο έργο *W3C Linking Open Data (LOD)*[2] που ξεκίνησε το 2007. Στόχος του έργου είναι να χαρτογραφηθούν τα διάσπαρτα σύνολα δεδομένων που διέπονται από ανοικτές άδειες, να μετατραπούν σε RDF μορφή σύμφωνα με τις αρχές των Διασυνδεδεμένων Δεδομένων και να δημοσιευθούν στο Διαδίκτυο. Η δημόσια πρόσβαση στα δεδομένα αποτελεί άλλο ένα κύριο χαρακτηριστικό των Διασυνδεδεμένων Δεδομένων. Παρακάτω φαίνονται τρία στιγμιότυπα του παγκόσμιου γράφου δεδομένων στην πάροδο του χρόνου:

[2] *http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData*

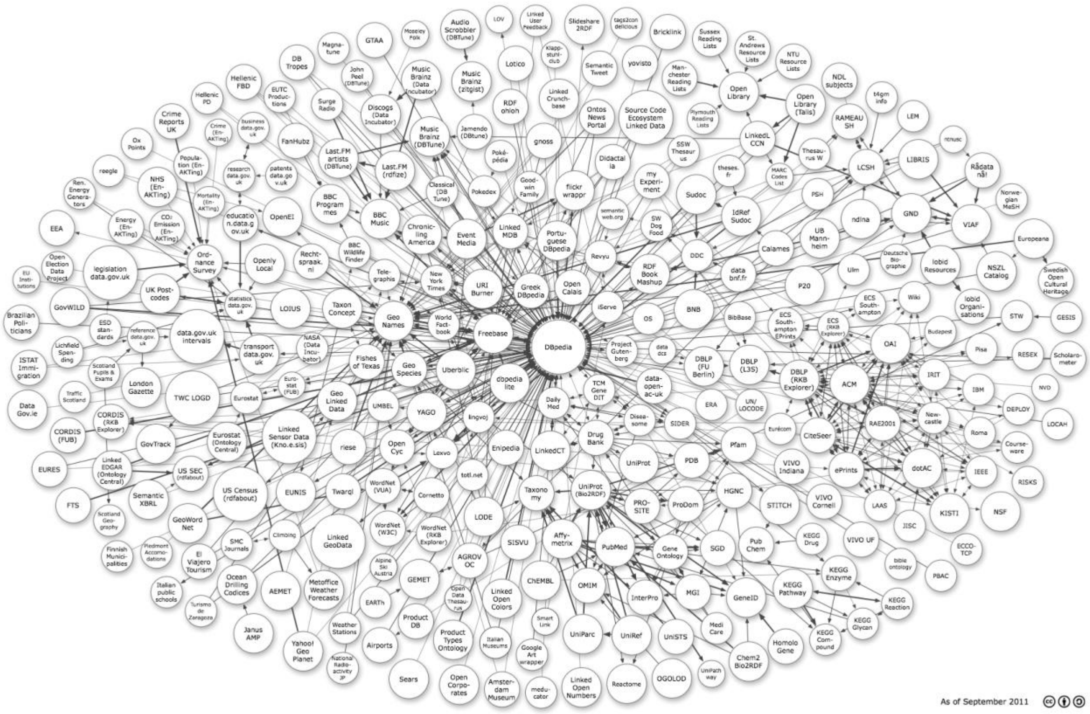


Εικόνα 1.1: Κατάσταση του LOD Cloud, Μάιος 2007

Να σημειωθεί πως ο όρος *LOD Cloud* χρησιμοποιείται για να δηλώσει τον γράφο των δεδομένων. Κάθε κόμβος στον γράφο περιγράφει ένα σύνολο δεδομένων και τα βέλη ορίζουν συνδέσμους ανάμεσα σε σύνολα. Τα πιο παχιά βέλη δείχνουν περισσότερους συνδέσμους μεταξύ των συνόλων και οι πιο μεγάλοι κόμβοι αναπαριστούν μεγαλύτερους όγκους δεδομένων σε αριθμό τριπλετών RDF.



Εικόνα 1.2: Κατάσταση του LOD Cloud, Ιούλιος 2009



Εικόνα 1.3: Κατάσταση του LOD Cloud, Σεπτέμβριος 2011

1.1.3 Προκλήσεις στον Ιστό Δεδομένων

Ο Ιστός Δεδομένων, όπως περιγράφηκε προηγουμένως, αποτελεί τη μετεξέλιξη του σημερινού διαδικτύου. Η υπάρχουσα κατάσταση, ωστόσο, απέχει αρκετά από το όραμα του παγκόσμιου χώρου δεδομένων. Η δυνατότητα παραγωγής και δημοσιοποίησης δεδομένων στη μορφή των Διασυνδεδεμένων Δεδομένων αποτελεί σίγουρα ένα πρώτο βήμα.

Γενικότερος στόχος είναι η εύκολη και γρήγορη μετατροπή υπαρχόντων δεδομένων σε μια κοινή μορφή και στη συνέχεια η σύνδεση μεταξύ τους. Τα μοντέλα των δεδομένων που περικλείονται στο σημερινό Διαδίκτυο είναι ποικίλα αυξάνοντας την ανάγκη για εισαγωγή προτύπων μετατροπής από επίσημους οργανισμούς. Τα πρότυπα αυτά θα γίνουν οι βάσεις για ανάπτυξη εργαλείων και εφαρμογών μετασχηματισμού της υπάρχουσας πληροφορίας κυρίως σε RDF που αποτελεί το πιο διαδεδομένο μοντέλο κοινής αναπαράστασης.

Επιπρόσθετα, πέρα από τα «εμφανή» δεδομένα του διαδικτύου μεταξύ των οποίων συγκαταλέγονται αρχεία κειμένων, φωτογραφιών, βίντεο ή στατιστικών, υπάρχει το λεγόμενο *Βαθύ Διαδίκτυο (Deep Web)* (Madhavan, et al.). Ο όρος αναφέρεται σε πληροφορία που αποκτά κανείς πρόσβαση μέσω HTML σελίδων. Αυτές οι σελίδες συνδέονται με βάσεις δεδομένων και παρουσιάζουν συγκεκριμένες πληροφορίες ανάλογα με το περιεχόμενο που παρέχουν οι χρήστες στις προβαλλόμενες φόρμες εισαγωγής στοιχείων. Έτσι, το περιεχόμενο

των βάσεων δεδομένων δεν είναι προσβάσιμο αυτομάτως. Αν αναλογιστεί κανείς το μέγεθος της παγκόσμιας πληροφορίας που βρίσκεται κρυμμένο σε σχεσιακές βάσεις γίνεται αντιληπτή η ανάγκη για μετατροπή αυτής της πληροφορίας σε RDF[3]. Σήμερα, υπάρχει δυνατότητα μετασχηματισμού μιας σχεσιακής βάσης δεδομένων σε RDF αλλά χωρίς κάποια προτυποποιημένη διαδικασία.

Η σύνδεση των δεδομένων μεταξύ τους ώστε να υπάρχει η δυνατότητα ανακάλυψής τους σε πραγματικό χρόνο αλλά και να αποδοθεί περισσότερη σημασία σε αυτά είναι άλλο ένα σημείο που απαιτεί έμφαση. Η ερευνητική κοινότητα μελετά πολύπλευρα το πρόβλημα της σύνδεσης τόσο από την πλευρά της Σημασιολογίας όσο και από την πλευρά της Διαχείρισης Δεδομένων. Γενικά, η σύνδεση των δεδομένων απαιτείται όταν δυο ή περισσότερες διαφορετικές αναπαραστάσεις πληροφορίας περιγράφουν το ίδιο φυσικό αντικείμενο. Για παράδειγμα αν δυο διαφορετικά URIs[4] περιγράφουν τον ίδιο άνθρωπο θα πρέπει να συνδεθούν. Η ανακάλυψη των συνδέσεων είναι πολύπλοκη διαδικασία κυρίως λόγω της χρήσης διαφορετικών λεξιλογίων για τη συγγραφή των URIs και της διαφορετικής σημασιολογίας των λεξιλογίων.

Τέλος, έντονο ερευνητικό ενδιαφέρον παρουσιάζει η ολοκλήρωση των δεδομένων και κυρίως ο τρόπος με τον οποίο αυτή γίνεται. Πιο συγκεκριμένα, ο υπάρχον όγκος των Διασυνδεδεμένων Δεδομένων ξεπερνά τα 31 δισεκατομμύρια εγγραφές RDF και είναι εύκολα αντιληπτή η αδυναμία χρησιμοποίησης ad hoc όλων αυτών των πληροφοριών. Έτσι, χρειάζεται να προταθούν αρχιτεκτονικές και μοντέλα ώστε να επιτυγχάνεται γρήγορα και αποτελεσματικά η ανακάλυψη και η διαχείριση των δεδομένων.

1.2 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την απλούστευση της διαδικασίας ολοκλήρωσης ετερογενών δεδομένων με μια σειρά απλών βημάτων στα οποία συμμετέχει ενεργά ο χρήστης. Δευτερεύων στόχος είναι η εξαγωγή πληροφοριών που δεν ήταν γνωστές πριν εφαρμοστεί η διαδικασία ολοκλήρωσης. Η εφαρμογή βοηθά στο

[3] *Το μοντέλο RDF δεν είναι η μοναδική επιλογή. Υπάρχουν και άλλες μορφές όπως XML, JSON, OWL κ.α. Ωστόσο, το RDF είναι το πιο απλό και ευέλικτο μοντέλο γι' αυτό είναι ευρέως διαδεδομένο.*

[4] *Η διαφορά των URIs είναι τόσο συντακτική όσο και εννοιολογική, π.χ. τα URIs http://organization1/people/scott-miller,*

http://organization2/staff/Computer_Scientists/ScottMiller

αναφέρονται στο ίδιο πρόσωπο αλλά αυτό δεν είναι αρχικά εμφανές.

σχηματισμό ενός ενιαίου, προσωπικού χώρου διαχείρισης και αναζήτησης της πληροφορίας από ετερογενείς πηγές (personal dataspace). Η διαχείρισή του γίνεται με τεχνολογίες RDF.

Αρχικά, προτείνεται ένας αλγόριθμος μετατροπής μη RDF δεδομένων σε RDF του οποίου η χρήση είναι πολύ σημαντική για την ομογενοποίηση των δεδομένων. Η εφαρμογή του αλγορίθμου εξειδικεύεται στη μετατροπή πινακοειδών (αρχεία csv ή excel) και σχεσιακών δεδομένων. Ο μετασχηματισμός αυτών των τύπων δεδομένων είναι πολύ σημαντικός καθώς μέχρι σήμερα η πληθώρα της πληροφορίας στην οποία βασίζονται οι υποδομές πληροφορικής βρίσκεται σε αυτές τις μορφές. Πυρήνας του αλγορίθμου είναι η απόδοση ενός URI σε κάθε νοητή οντότητα ενός συνόλου δεδομένων που αναπαρίσταται με τη μορφή πίνακα (γραμμή, στήλη, κελί καθώς και στο ίδιο το σύνολο δεδομένων – dataset).

Ακόμη, στόχος της διπλωματικής εργασίας είναι η ανάπτυξη ενός αποδοτικού και εύχρηστου γραφικού περιβάλλοντος ώστε ο χρήστης με απλές ενέργειες να εφαρμόζει σημαντικές διαδικασίες επί των δεδομένων. Η προτεινόμενη εφαρμογή συνδυάζει υπάρχοντα εργαλεία που επιτελούν σημαντικές λειτουργίες της διαδικασίας ολοκλήρωσης σε ένα ενοποιημένο περιβάλλον. Έτσι, μειώνεται ο χρόνος που απαιτείται για την εφαρμογή της διαδικασίας ολοκλήρωσης.

Όπως έχει προαναφερθεί, η εφαρμογή που αναπτύχθηκε προσφέρει τη δυνατότητα στους χρήστες να συνδυάζουν ετερογενή δεδομένα προερχόμενα από διαφορετικές πηγές και να τα συγκεντρώνουν σε έναν ενιαίο χώρο δεδομένων. Με τον τρόπο αυτό είναι δυνατή - μέσω της εφαρμογής SPARQL ερωτημάτων - η άντληση χρήσιμης και ίσως άγνωστης πληροφορίας. Αυτό οφείλεται στο γεγονός ότι τα δεδομένα που εισάγονται είναι πολύ πιθανόν να περιέχουν πληροφορίες από διαφορετική σκοπιά για μια κατάσταση του φυσικού κόσμου με αποτέλεσμα η ένωσή τους να οδηγεί στην εξαγωγή συνδυαστικής γνώσης.

1.2.1 Συνεισφορά

Η διπλωματική έχει ως αντικείμενο την ανάπτυξη ενός εργαλείου σχηματισμού χώρων δεδομένων (dataspaces) από ετερογενείς πηγές με εύκολο, γρήγορο και απλό τρόπο. Η συνεισφορά της αναλύεται στα ακόλουθα σημεία:

1. Ανάπτυξη αλγορίθμου μετατροπής πινακοειδών και σχεσιακών δεδομένων σε RDF
2. Ανάπτυξη απλού και αποδοτικού περιβάλλοντος διεπαφής χρήστη
3. Σύνδεση επιμέρους συστημάτων λογισμικού και ένωσή τους σε ένα ενιαίο πακέτο λογισμικού
4. Απλούστευση της διαδικασίας ολοκλήρωσης και μείωση του χρόνου εφαρμογής της

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2 . Στο Κεφάλαιο 3 αναπτύσσονται τα εργαλεία στα οποία θα βασιστεί το προτεινόμενο σύστημα. Στο Κεφάλαιο 4 αναπτύσσουμε τα θεωρητικά μοντέλα που βασίζεται η ανάπτυξη του συστήματος και η κατανόησή τους κρίνεται απαραίτητη.

2

Σχετικές εργασίες

Στο κεφάλαιο αυτό παραθέτουμε εργασίες που μελετήθηκαν για την ανάπτυξη της εφαρμογής ολοκλήρωσης Διασυνδεδεμένων Δεδομένων. Κάποιες από αυτές περιγράφουν υπάρχοντα εργαλεία και εφαρμογές. Περιλαμβάνονται, ακόμη, εργασίες που πραγματεύονται επιμέρους ερευνητικούς τομείς των Διασυνδεδεμένων Δεδομένων.

2.1 Συλλογή Δεδομένων από το Διαδίκτυο

Οι μέθοδοι πρόσβασης και συλλογής δεδομένων από το διαδίκτυο στη μορφή RDF διαφέρουν ανάλογα με την αρχιτεκτονική του εκάστοτε συστήματος. Οι βασικοί τρόποι πρόσβασης και πλοήγησης στα δεδομένα είναι η μετατροπή των HTTP URIs σε RDF και η διάσχιση των RDF συνδέσμων ώστε να ανακαλυφθούν νέα δεδομένα. Επιπλέον, άλλος ένας τρόπος είναι η χρήση των *SPARQL endpoints*. Πρόκειται για συγκεκριμένες υπηρεσίες που εκτελούν ερωτήματα στη γλώσσα SPARQL και φέρνουν RDF δεδομένα από βάσεις γνώσης. Τα SPARQL ερωτήματα γράφονται είτε από τους χρήστες μέσω κάποιου γραφικού περιβάλλοντος είτε προγραμματιστικά στα πλαίσια κάποιας εφαρμογής. Ο τελευταίος τρόπος είναι και ο πιο συνηθισμένος. Ακόμη, είναι διαθέσιμα έτοιμα σύνολα RDF δεδομένων τα οποία κάποιος χρήστης μπορεί να αποθηκεύσει σε τοπικό υπολογιστή σε μορφή αρχείου. Τέλος, υπάρχουν μηχανές αναζήτησης Διασυνδεδεμένων Δεδομένων που αποθηκεύουν προσωρινά τα δεδομένα που ανακαλύπτουν και παρέχουν ένα API ώστε να μπορούν οι εφαρμογές να έχουν πρόσβαση στα δεδομένα αυτά.

Στα πλαίσια της εφαρμογής η πρόσβαση σε RDF δεδομένα θα γίνει με τοπικά αρχεία (*dumps*) και με τη χρήση SPARQL endpoints.

2.2 Μετατροπή Δεδομένων σε RDF

Η μετατροπή δεδομένων διαφορετικού τύπου σε RDF αποτελεί μια εξελισσόμενη ερευνητική διαδικασία που δεν έχει πλαισιωθεί από πρότυπα, όπως προαναφέρθηκε σε προηγούμενη ενότητα. Οι υπάρχουσες μορφές δόμησης ψηφιακής πληροφορίας είναι ποικίλες αλλά η παρούσα διπλωματική επικεντρώνεται στη μετατροπή πινακοειδών (*tabular*) και σχεσιακών (relational) δεδομένων στο μοντέλο RDF.

2.2.1 Μετατροπή σχεσιακών Δεδομένων σε RDF

Ο μετασχηματισμός σχεσιακής πληροφορίας που βρίσκεται σε βάσεις δεδομένων στο πρότυπο RDF είναι σύνθετη διαδικασία κι έχει προκαλέσει έντονο ερευνητικό ενδιαφέρον. Αποτέλεσμα της ερευνητικής προσπάθειας είναι η ανάπτυξη αρκετών μοντέλων και εργαλείων χωρίς, ωστόσο, να υπάρχει κάποια γενικά αποδεκτή μεθοδολογία.

Ο Tim Berners-Lee (Berners-Lee, 1998) εισήγαγε τις βασικές αρχές μετατροπής σχεσιακών δεδομένων σε RDF. Το σχεσιακό μοντέλο και το μοντέλο RDF είναι αρκετά παρόμοια. Μια σχεσιακή βάση δεδομένων περιλαμβάνει πίνακες, οι οποίοι δομούνται από γραμμές, στήλες και κελιά. Κάθε γραμμή ή αλλιώς εγγραφή είναι το περιεχόμενο των στηλών. Όμοια, ένα αντικείμενο που περιγράφεται σε RDF προσδιορίζεται από τις τιμές των ιδιοτήτων του. Αυτό σημαίνει πως η αντιστοιχία είναι σχεδόν άμεση:

- Μια εγγραφή είναι ένα RDF αντικείμενο
- Κάθε στήλη (πεδίο) είναι μια ιδιότητα (*PropertyType*)
- Η τιμή κάθε πεδίου είναι η τιμή κάθε ιδιότητας

Πέρα, όμως, από αυτή την γενική αντιστοίχιση υπεισέρχονται και άλλοι παράγοντες μοντελοποίησης που πηγάζουν από τον τρόπο οργάνωσης των σχεσιακών βάσεων δεδομένων. Πιο συγκεκριμένα, χρειάζονται να μοντελοποιηθούν τα *πρωτεύοντα κλειδιά* (*primary keys*) των πινάκων, οι περιορισμοί ακεραιότητας, τα *ξένα κλειδιά* (*foreign keys*) ανάμεσα στους πίνακες και άλλα παρόμοια τεχνικά χαρακτηριστικά των σχεσιακών βάσεων.

Η ομάδα RDB2RDF του οργανισμού W3C[5] έχει ως στόχο την ανάπτυξη και την προτυποποίηση γλωσσών μετατροπής από σχεσιακά σε RDF δεδομένα. Μέχρι σήμερα έχουν προταθεί δυο γλώσσες, η *Direct Mapping(DM)* και η *RDB2RDF Mapping Language (R2RML)*. Ωστόσο, δεν είναι ευρέως διαδεδομένες καθώς δεν έχουν αφομοιωθεί από τις υπάρχουσες γνωστές εφαρμογές. Οι περισσότερες τεχνικές που έχουν αναπτυχθεί

[5] *http://www.w3.org/2001/sw/rdb2rdf/*

εφαρμόζουν τη γενική αντιστοιχία που προτάθηκε από τον Tim Berners-Lee και την εμπλουτίζουν με επιπλέον αντιστοιχίσεις. Πάντως, παρά το γεγονός ότι η γενική αντιστοιχία παρέχει ένα καλό επίπεδο μετασχηματισμού, η σύνθετη σημασιολογία που απαιτείται από τις περισσότερες εφαρμογές οδηγεί στην ανάγκη για πιο αναλυτικό μετασχηματισμό. Επιπλέον, έχουν αναπτυχθεί συστήματα που βασίζουν τη μετατροπή σχεσιακών δεδομένων σε RDF σε εξειδικευμένες γλώσσες μετατροπής. Οι τελευταίες λαμβάνουν υπόψιν και τη σημασιολογία που υπεισέρχεται στα δεδομένα κάνοντας χρήση λεξιλογίων RDF και οντολογιών.

Στη μελέτη (S. Sahoo, et al., 2008) παρουσιάζονται τα σημαντικότερα εργαλεία και συστήματα μετατροπής σχεσιακών δεδομένων σε RDF. Η συγκεκριμένη μελέτη αποτελεί και τον πιο σύγχρονο οδηγό για την κάλυψη αυτού του ερευνητικού πεδίου. Ακόμη, ο οργανισμός W3C παρέχει μια συνεχώς ανανεώσιμη λίστα με τα τρέχοντα εργαλεία μετατροπής[6]. Στα πλαίσια της βιβλιογραφικής έρευνας μελετήθηκαν τα εργαλεία: *D2R Server* (Bizer & Cyganiak, 2006), (Bizer & Seaborne, 2004), *Dartgrid* (Wu, et al., 2006) και *Triplify* (Auer, Dietzold, Lehmann, Hellmann, & Aumueller, 2005). Το τρίτο σύστημα πλησιάζει περισσότερο στο προτεινόμενο μοντέλο μετατροπής σχεσιακών δεδομένων αλλά παρουσιάζει ορισμένες διαφορές.

Το Triplify αποτελεί μια απλή και μικρή σε έκταση εφαρμογή που μετατρέπει σχεσιακή πληροφορία σε Διασυνδεδεμένα Δεδομένα. Βασίζεται στην αντιστοίχιση αιτημάτων για HTTP-URIs με ερωτήματα σε SQL. Στη συνέχεια μετατρέπει τα δεδομένα που προκύπτουν ως αποτελέσματα στα ερωτήματα σε Διασυνδεδεμένα Δεδομένα. Βασικός στόχος της εφαρμογής είναι η απλοποίηση της διαδικασίας μετατροπής σχεσιακών δεδομένων σε Διασυνδεδεμένα Δεδομένα και για το λόγο αυτό δεν χρησιμοποιείται κάποια εξειδικευμένη γλώσσα μετατροπής, όπως σε άλλα εργαλεία. Το μόνο που απαιτείται είναι η χρήση της γλώσσας SQL και κάποιων λεξιλογίων του μοντέλου RDF. Η διαδικασία μετατροπής μπορεί να γίνεται είτε σε πραγματικό χρόνο είτε εκ των προτέρων (ETL διαδικασία).

Ο πυρήνας της εφαρμογής είναι ο ορισμός μιας προκαθορισμένης διαμόρφωσης για μια συγκεκριμένη βάση δεδομένων. Η διαμόρφωση είναι μια τριπλέτα της μορφής *(s,φ,μ)* όπου *s* είναι ο προκαθορισμένος χώρος ονομάτων (*namespace*) του σχήματος, *φ* είναι η αντιστοιχία ανάμεσα στα προθέματα του μοντέλου RDF και στα URIs του χώρου ονομάτων, *μ* είναι η αντιστοιχία ανάμεσα στα URL και στα SQL ερωτήματα. Ο προκαθορισμένος χώρος ονομάτων του σχήματος *s* χρησιμοποιείται για τη δημιουργία URI αναγνωριστικών για τις στήλες των πινάκων της βάσης που δεν αντιστοιχούνται σε κάποιο γνωστό λεξιλόγιο RDF. Η αντιστοιχία *φ* ορίζει συντομεύσεις για συχνά χρησιμοποιούμενους χώρους ονομάτων. Τα

[6] *http://www.w3.org/wiki/ConverterToRdf#SQL*

URLs ανάλογα με τη μορφή που έχουν αντιστοιχίζονται σε συγκεκριμένα ερωτήματα SQL. Παρακάτω θα δοθεί συγκεκριμένο παράδειγμα που θα γίνεται κατανοητή η αντιστοιχία.

Προκειμένου να μετατραπούν τα σχεσιακά δεδομένα σε RDF το Triplify χρησιμοποιεί την προσέγγιση *πίνακα-κλάσης (multiple-table-to-class)*. Η προσέγγιση αυτή ομοιάζει με τις βασικές αρχές που προτάθηκαν από τον Tim Berners-Lee και περιλαμβάνει τα εξής χαρακτηριστικά:

- Η πρώτη στήλη κάθε προβολής πρέπει να περιέχει ένα αναγνωριστικό που θα χρησιμοποιείται για τη δημιουργία των αντίστοιχων URIs. Το αναγνωριστικό αυτό μπορεί να είναι το πρωτεύον κλειδί κάθε πίνακα.
- Τα ονόματα των στηλών χρησιμοποιούνται για να δημιουργηθούν URIs που θα περιγράφουν τις ιδιότητες.
- Κάθε κελί περιέχει τιμές ή αναφορές σε άλλες οντότητες και αποτελεί το αντικείμενο (*object*) της εκάστοτε τριπλέτας RDF.

Η μετονομασία των στηλών προϋποθέτει την επέκταση των SQL ερωτημάτων ώστε να ενσωματώνονται αυτόματα τα λεξιλόγια RDF και να σχηματίζονται τα αντικείμενα των τριπλετών. Η επέκταση έχει τις ακόλουθες μορφές:

- *Αντιστοίχιση σε υπάρχοντα λεξιλόγια:* Η μετονομασία των στηλών σε ιδιότητες λεξιλογίων RDF διευκολύνει τη μετατροπή. Για παράδειγμα το ερώτημα `SELECT id, name AS 'foaf:name' FROM users` μετατρέπει τη στήλη name των αποτελεσμάτων στην ιδιότητα name του γνωστού λεξιλογίου FOAF[7].
- *Ιδιότητες αντικειμένων:* Όλες οι ιδιότητες των αντικειμένων θεωρούνται ιδιότητες τύπων δεδομένων (*dataType properties*) και οι τιμές αυτών μετατρέπονται σε λεκτικά RDF. Προσθέτοντας μια αναφορά σε μια άλλη οντότητα που χωρίζεται με το χαρακτήρα '→' από το όνομα της στήλης, το Triplify χρησιμοποιεί την τιμή της στήλης για να δημιουργήσει ένα νέο URI (βλέπε παράδειγμα παρακάτω).
- *Τύποι δεδομένων:* Η εφαρμογή χρησιμοποιεί τη γλώσσα SQL ώστε να λάβει αυτόματα τον τύπο μιας συγκεκριμένης στήλης και να παράξει λεκτικά RDF με αυτούς τους τύπους. Ο χαρακτήρας που χρησιμοποιείται για την απόδοση της τιμής είναι ο '^^'.
- *Γλωσσικές ετικέτες:* Σε όλα τα λεκτικά που είναι συμβολοσειρές προστίθεται μια γλωσσική ετικέτα που προστίθεται στο όνομα της στήλης μετά τον χαρακτήρα '@'.

Ένα παράδειγμα που δείχνει τη χρήση των παραπάνω μετατροπών φαίνεται στη συνέχεια:

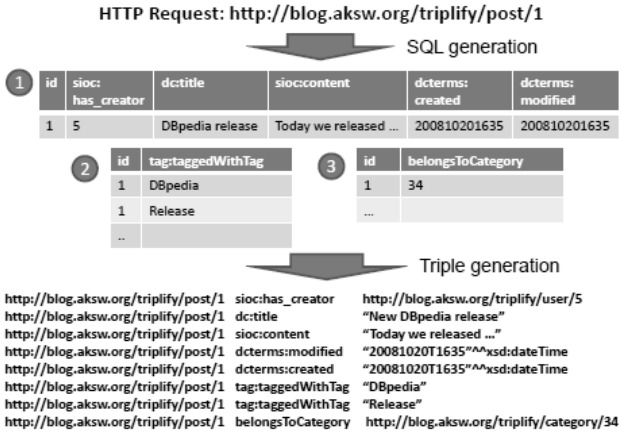
```
SELECT id,
price AS 'price^^xsd:decimal',
desc AS 'rdfs:label@en',
cat AS 'belongsToCategory->category'
```

[7] *FOAF: Friend Of A Friend οντολογία, http://xmlns.com/foaf/spec/*

```
FROM products
```

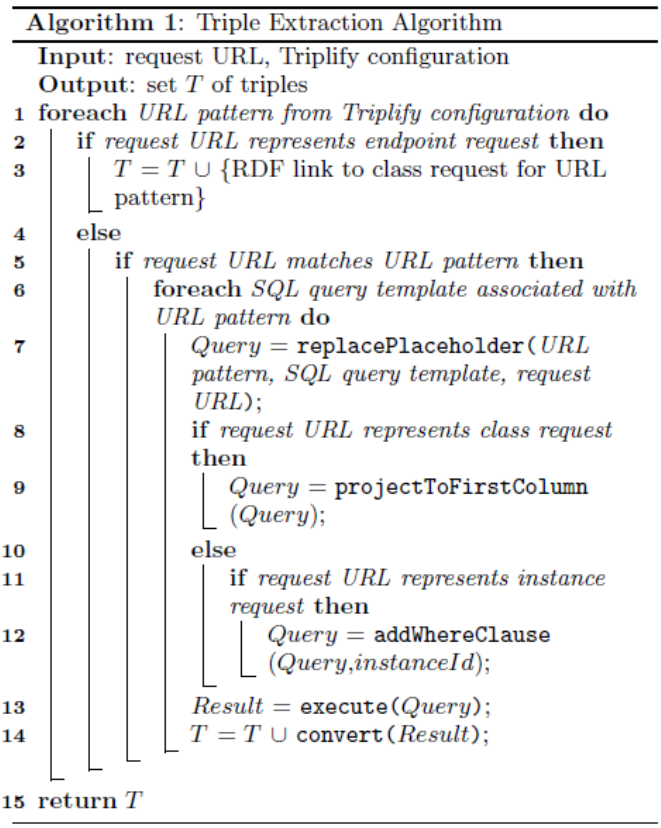
Εδώ, επιλέγονται προϊόντα από τον πίνακα products. Οι ιδιότητες price, desc και cat αντιστοιχίζονται σε κατάλληλες ιδιότητες RDF. Η κατηγορία των προϊόντων αντιστοιχίζεται σε αναφορά στον πίνακα category, δηλαδή κάθε τιμή θα είναι ένα URI που θα προσδιορίζεται από τον πίνακα category.

Ο αλγόριθμος μετατροπής των σχεσιακών δεδομένων σε τριπλέτες RDF λαμβάνει ως είσοδο ένα URL. Στη συνέχεια αναλύει το URL με βάση τα διαχωριστικά '/' και εκτελεί τα αντίστοιχα SQL ερωτήματα. Βασική αρχή των Διασυνδεδεμένων Δεδομένων είναι η δυνατότητα των URIs να μπορούν να αναζητηθούν δηλαδή να είναι προσβάσιμα με το πρωτόκολλο HTTP. Έτσι, το Triplify χρησιμοποιεί την αρχή αυτή ώστε να μετατρέπονται τα σχεσιακά δεδομένα σε Διασυνδεδεμένα. Για παράδειγμα ένα URI που μπορεί να αποτελεί είσοδο για τον αλγόριθμο είναι το `http://myblog.de/triplify/posts`. Το URI αυτό ανταποκρίνεται στον πίνακα posts της βάσης. Για μια συγκεκριμένη εγγραφή-οντότητα της βάσης το αντίστοιχο URI μπορεί να είναι το `http://myblog.de/triplify/posts/13`. Η μετατροπή των URIs σε SQL ερωτήματα γίνεται με τη διαδικασία που περιγράφηκε παραπάνω για συγκεκριμένο πίνακα της βάσης που προκύπτει από την ανάλυση του URI. Σε περίπτωση που ζητείται συγκεκριμένη εγγραφή προστίθεται ο όρος where στο αντίστοιχο SQL ερώτημα. Η ιδιότητα του όρου where είναι το πρωτεύον κλειδί του πίνακα, δηλαδή η πρώτη στήλη, και η τιμή της είναι η τιμή που βρίσκεται στο URI. Παρακάτω φαίνεται σχηματικά ένα παρόμοιο παράδειγμα:



Εικόνα 2.1: Παράδειγμα μετατροπής σχεσιακού περιεχομένου σε Linked Data με το Triplify

Ο αλγόριθμος που ακολουθείται είναι ο εξής:



Εικόνα 2.2: Αλγόριθμος μετατροπής σχεσιακών δεδομένων σε Linked Data

Το Triplify πλησιάζει αρκετά το προτεινόμενο πλάνο μετατροπής σχεσιακών δεδομένων σε Διασυνδεδεμένα. Κοινός στόχος είναι η απόδοση URI που να προσδιορίζει μοναδικά κάθε κελί των πινάκων της βάσης. Επίσης, τα URIs πρέπει να είναι προσβάσιμα με το πρωτόκολλο HTTP. Βασική διαφορά είναι το γεγονός ότι στο Triplify γνωρίζουμε εκ των προτέρων το σχήμα της βάσης που θα μετασχηματιστεί. Η προτεινόμενη μέθοδος είναι γενικότερη και ανεξάρτητη από το σχήμα της βάσης. Αυτό οφείλεται στο γεγονός ότι η μετατροπή σε Διασυνδεδεμένα δεδομένα γίνεται στα πλαίσια ενός συνολικού συστήματος ολοκλήρωσης και όχι ως ανεξάρτητη διαδικασία. Έτσι, είναι αναγκαία η ύπαρξη ενός γρήγορου και κυρίως ευέλικτου τρόπου μετατροπής που να μπορεί να υποστηρίξει την πραγματικού χρόνου ολοκλήρωση δεδομένων. Η διαδικασία θα περιγραφεί αναλυτικά στο Κεφάλαιο 7.

2.2.2 Μετατροπή πινακοειδών Δεδομένων σε RDF

Ο μετασχηματισμός πινακοειδών (tabular) δεδομένων στη μορφή RDF είναι άλλο ένα καίριο ζήτημα στην ολοκλήρωση δεδομένων. Αυτά τα δεδομένα είναι πηγή σημαντικών πληροφοριών (π.χ. στατιστικά στοιχεία) και για το λόγο αυτό είναι αναγκαία η μετατροπή τους σε RDF ώστε να μπορούν να συνδυαστούν και με άλλα δεδομένα.

Μέχρι σήμερα έχουν αναπτυχθεί αρκετά εργαλεία που υποστηρίζουν το μετασχηματισμό σε RDF. Μια λίστα με τις υπάρχουσες εφαρμογές διατηρείται από τον οργανισμό W3C[8]. Για τη μετατροπή σε RDF χρησιμοποιείται κυρίως το λεξιλόγιο *Data Cube Vocabulary*, τα βασικά χαρακτηριστικά του οποίου θα αναλυθούν σε επόμενη ενότητα. Τα πινακοειδή δεδομένα βρίσκονται σε μορφή CSV[9] και Excel αρχείων. Στα πλαίσια της διπλωματικής μελετήθηκε κυρίως το εργαλείο *Open Refine*[10] της εταιρίας Google και η επέκτασή του, *RDF Refine*[11], που επιτρέπει τη μετατροπή CSV αρχείων σε RDF.

Πιο συγκεκριμένα, το Open Refine (Maali, Cyganiak, & Peristeras, 2011) είναι ένα εργαλείο διαχείρισης δεδομένων σε μορφή πινάκων, δηλαδή κυρίως CSV και Excel. Παρέχει ένα σύνολο λειτουργιών που επιτρέπουν γρήγορη διόρθωση, καθαρισμό και μετασχηματισμό δεδομένων. Η βασική έκδοση της εφαρμογής δεν επιτρέπει την απευθείας μετατροπή και εξαγωγή δεδομένων σε RDF. Αντίθετα, χρησιμοποιείται η γλώσσα XML ώστε να μοντελοποιηθούν RDF δεδομένα. Η επέκταση RDF Refine καλύπτει το κενό αυτό παρέχοντας ένα λειτουργικό γραφικό περιβάλλον. Πυρήνας της επέκτασης είναι ο ορισμός των πόρων και των λεκτικών που θα περιληφθούν στον εξαγόμενο RDF γράφο, των σχέσεων μεταξύ τους και των URIs που θα χρησιμοποιηθούν για να προσδιορίσουν τους πόρους. Κάθε υπογράφος του εξαγόμενου γράφου περιλαμβάνει μια σειρά του αρχείου CSV. Ο τελικός γράφος είναι η σύνθεση όλων των υπογράφων.

2.3 Ταίριασμα και Σύνδεση Δεδομένων

Το πρόβλημα του ταιριάσματος δεδομένων αποτελεί ίσως την πιο σημαντική πρόκληση στο χώρο της ολοκλήρωσης δεδομένων. Πρώτη φορά τέθηκε το 1959 (Köpcke & Rahm, 2009) και έκτοτε μελετάται πολύπλευρα τόσο από την πλευρά των Βάσεων Δεδομένων και της Τεχνητής Νοημοσύνης όσο και από την πιο σύγχρονη σκοπιά του Σημασιολογικού Ιστού και της Μηχανικής Μάθησης. Πρόκειται για τη διαδικασία αναγνώρισης οντοτήτων (αναπαραστάσεων πληροφορίας) που αναφέρονται στο ίδιο αντικείμενο του φυσικού κόσμου. Οι οντότητες μπορεί να προέρχονται από διαφορετικές πηγές πληροφορίας αλλά είναι πιθανό να ανήκουν και στο ίδιο σύνολο δεδομένων. Λόγω της πολύπλευρης φύσης του προβλήματος έχουν αναπτυχθεί πολλές τεχνικές, αλγόριθμοι και εργαλεία για την επίλυσή του. Ωστόσο, καμιά προσέγγιση δεν αποτελεί ιδανική λύση.

[8] *http://www.w3.org/wiki/ConverterToRdf#CSV_.28Comma-Separated_Values.29*

[9] *CSV: Comma Seperated Values, http://en.wikipedia.org/wiki/Comma-separated_values*

[10] *http://openrefine.org/*

[11] *http://refine.deri.ie/*

Στο χώρο των Βάσεων Δεδομένων (Elmagarmid, G. Ipeirotis, & S. Verykios, 2007) τα δεδομένα είναι πιθανό να έχουν συντακτικά λάθη, μη έγκυρες ή ακόμη και ψευδείς τιμές. Για το λόγο αυτό είναι επιτακτική η ανάγκη ανάπτυξης τεχνικών ταιριάσματος κάτω από αυτές τις συνθήκες. Έχουν προταθεί αρκετές τεχνικές εύρεσης ταυτόσημων εγγραφών που αφορούν εύρεση ομοιοτήτων τόσο σε λεξιλογικό όσο και σε δομικό επίπεδο. Σε λεξιλογικό επίπεδο έχουν αναπτυχθεί μέθοδοι συντακτικής ομοιότητας (*string similarity*), λεκτικής ομοιότητας (*token-based similarity*) ακόμη και φωνητικής ομοιότητας (*phonetic similarity*). Οι παραπάνω τεχνικές εφαρμόζονται σε πεδία εγγραφών και όχι σε ολόκληρες εγγραφές. Η ομοιότητα ανάμεσα σε εγγραφές με πολλά πεδία είναι δυσκολότερο να εντοπιστεί γι' αυτό και η φύση των μεθόδων που έχουν αναπτυχθεί είναι διαφορετική σε σχέση με τις προηγούμενες. Υπάρχουν μέθοδοι που βασίζονται σε Μηχανική Μάθηση περιλαμβάνοντας στατιστικές προσεγγίσεις. Ακόμη, προτείνονται τεχνικές που λαμβάνουν υπόψιν το πεδίο γνώσης των δεδομένων. Οι τεχνικές αυτές χρησιμοποιούν ειδικές γλώσσες για ταίριασμα αλλά και μετρικές ομοιότητας με βάση την απόσταση των εγγραφών (*distance based similarity*). Οι μετρικές ομοιότητας θεωρούν μια εγγραφή ως ένα σύνολο από πεδία και εφαρμόζουν συντακτικές μεθόδους.

Η φύση του ταιριάσματος δεδομένων αλλάζει ελαφρά στο χώρο του Σημασιολογικού Ιστού. Στόχος αποτελεί το ταίριασμα και η μετέπειτα σύνδεση RDF δεδομένων. Ωστόσο, σε αντίθεση με τις σχεσιακές βάσεις όπου ο τρόπος δόμησης των δεδομένων σε πίνακες και πεδία είναι γνωστός, τα RDF δεδομένα δεν έχουν περιορισμούς σχετικά με τη χρήση ιδιοτήτων. Ακόμη, στις βάσεις δεδομένων συνήθως πρώτα ταιριάζονται τα σχήματα, δηλαδή ο τρόπος δόμησης των δεδομένων, και μετά τα ίδια τα δεδομένα. Αντίθετα, οι αναπαραστάσεις πληροφορίας σε RDF μπορεί να συνδέονται χωρίς απαραίτητα να έχουν κοινές ιδιότητες.

Η σύνδεση συνόλων δεδομένων που έχουν δημοσιευθεί στο LOD γίνεται κυρίως με εφαρμογή κανόνων από τους χρήστες. Η προσέγγιση αυτή είναι αρκετά αποτελεσματική αλλά δαπανηρή σε χρόνο και προσπάθεια. Η δυσκολία έγκειται στην εύρεση κατάλληλων κανόνων ταιριάσματος στα δεδομένα μεταξύ δυο συνόλων. Επιπλέον, απαιτείται επαρκής γνώση των σχημάτων και των λεξιλογίων που χρησιμοποιούν τα σύνολα δεδομένων. Οι τεχνικές Μηχανικής Μάθησης που αναπτύχθηκαν στο χώρο των Βάσεων Δεδομένων μπορούν να εφαρμοστούν και στο Σημασιολογικό Ιστό αλλά είναι αρκετά δύσκολο να βρεθούν αντιπροσωπευτικά σύνολα εκπαίδευσης ειδικά στο χώρο των Διασυνδεδεμένων Δεδομένων.

Στα πλαίσια της διπλωματικής μελετήθηκαν μέθοδοι και εφαρμογές που πραγματεύονται τη σημασιολογική πλευρά του ζητήματος, όπου το ταίριασμα δεδομένων στην ουσία εξειδικεύεται σε σύνδεση συνόλων δεδομένων. Πιο συγκεκριμένα, δόθηκε

έμφαση στη μελέτη των εργαλείων *SERIMI* (Araujo, Hidders, Schwabe, & Vries, 2011) και *SILK* (Volz, Bizer, Gaedke, & Kobilarov, 2009) που συνδέονται με το ταίριασμα ταυτόσημων δεδομένων τόσο σε μορφή RDF όσο και Διασυνδεδεμένων Δεδομένων. Το SERIMI περιορίζει την ανθρώπινη συμμετοχή και αυτόματα βρίσκει κανόνες ταιριάσματος χωρίς προηγούμενη γνώση σχετικά με το είδος, το εννοιολογικό περιβάλλον και τους κανόνες δόμησης των δεδομένων. Το SILK από την άλλη πλευρά βασίζεται στην ανθρώπινη κρίση για το σχηματισμό των συνδέσεων.

Η πρώτη εφαρμογή προσπαθεί να ταιριάξει δεδομένα τόσο με συντακτική όσο και με εννοιολογική ανάλυση. Πιο συγκεκριμένα, με τη χρήση των *ετικετών* στα RDF δεδομένα γίνεται μια πρώτη εκτίμηση για το ποια δεδομένα ταιριάζουν μεταξύ τους βάσει αλγορίθμων συντακτικής ομοιότητας. Παρά το γεγονός ότι η συντακτική ομοιότητα οδηγεί συχνά σε αληθές ταίριασμα δεδομένων, αυτό δεν είναι βέβαιο γενικά. Για παράδειγμα, μπορεί να υπάρχουν δυο αναπαραστάσεις πληροφορίας που περιγράφουν τον πληθυσμό μιας χώρας. Ωστόσο, η πρώτη έχει την ιδιότητα «πληθυσμός» ενώ η δεύτερη την ιδιότητα «αριθμός κατοίκων». Εννοιολογικά οι δυο ιδιότητες είναι ισοδύναμες αλλά συντακτικά είναι τελείως διαφορετικές. Για το λόγο αυτό γίνεται έλεγχος σε σημασιολογικό επίπεδο με τη χρήση ενός αλγορίθμου ομοιότητας.

Το SILK επιτρέπει την εύρεση σχέσεων ανάμεσα σε οντότητες που ανήκουν σε διαφορετικά σύνολα δεδομένων και μπορεί να χρησιμοποιηθεί για τον ορισμό συνδέσεων ανάμεσά τους. Η εφαρμογή παρέχει μια γλώσσα (*Silk-Link Specification Language, Silk-LSL*) προκειμένου να οριστούν τα είδη συνδέσεων που θα ανακαλυφθούν καθώς και ποιες συνθήκες πρέπει να πληρούνται ώστε να έχουν ισχύ αυτές οι συνδέσεις. Οι συνθήκες βασίζονται σε μετρικές ομοιότητας τόσο συντακτικής όσο και εννοιολογικής. Το SILK αποκτά πρόσβαση στα δεδομένα μέσω του πρωτοκόλλου της γλώσσας SPARQL κι έτσι δεν είναι αναγκαίο να αντιγραφούν τοπικά προκειμένου να γίνει η επεξεργασία τους. Αυτό έχει σαν αποτέλεσμα την κατανεμημένη επεξεργασία δεδομένων. Ακόμη, χρησιμοποιεί μεθόδους ευρετηριοποίησης των δεδομένων και περιορισμού του εύρους αναζήτησης ώστε να αυξηθεί η απόδοση και να μειωθεί ο φόρτος του δικτύου.

Το SILK είναι πιο ολοκληρωμένο σύστημα σε σχέση με το SERIMI και έχει βελτιωθεί σε μεγάλο βαθμό από τη στιγμή της δημιουργίας του. Ο γραφικός τρόπος επιλογής των συνδέσεων το καθιστά πιο ευέλικτο και πιο αποτελεσματικό. Επειδή η λογική ανάπτυξης του συστήματος ολοκλήρωσης της παρούσας διπλωματικής είναι η ενεργή συμμετοχή του χρήστη σε όλα τα στάδια επεξεργασίας των δεδομένων ο πυρήνας του SILK χρησιμοποιήθηκε ως συστατικό στοιχείο. Σε επόμενες ενότητες θα δοθούν περισσότερες πληροφορίες σχετικά με τη γλώσσα SILK-LSL, την αρχιτεκτονική του και τη μέθοδο

περιορισμού του εύρους αναζήτησης. Τέλος, να σημειωθεί πως μια συγκριτική έρευνα σχετικά με τα εργαλεία ταιριάσματος δεδομένων είναι η (Köpcke & Rahm, 2009).

2.4 Ταίριασμα Δεδομένων σε επίπεδο λεξιλογίου

Στην προηγούμενη ενότητα ορίστηκε το γενικό πρόβλημα του ταιριάσματος δεδομένων κυρίως σε επίπεδο αναπαράστασης δεδομένων. Η αντιστοίχιση οντολογιών (*ontology matching*) αποτελεί άλλη μια έκφραση του ταιριάσματος στο Σημασιολογικό Ιστό. Βασίζεται σε τεχνικές παρόμοιες με εκείνες που χρησιμοποιούνται στις βάσεις δεδομένων για την αντιστοιχία σχημάτων καθώς και σε μεθόδους Μηχανικής Μάθησης (Heath & Bizer, 2011, p. 102). Η εύρεση αντιστοιχιών ανάμεσα σε όρους και ιδιότητες διαφορετικών λεξιλογίων έχει ιδιαίτερη βαρύτητα στο χώρο των Διασυνδεδεμένων Δεδομένων και αποτελεί ένα σημαντικό βήμα για την ολοκλήρωσή τους.

Η ανακάλυψη νέων πηγών δεδομένων ακολουθώντας RDF συνδέσμους σε πραγματικό χρόνο και η ομαλή ενσωμάτωσή τους από εφαρμογές αποτελούν βασικούς στόχους των Διασυνδεδεμένων Δεδομένων. Οι πηγές δεδομένων χρησιμοποιούν διαφορετικά λεξιλόγια για την αναπαράσταση πληροφορίας με αποτέλεσμα τη μίξη όρων. Αυτό σημαίνει πως οι εφαρμογές χρειάζεται να αντιστοιχούν τα διαφορετικά λεξιλόγια σε ένα τοπικό λεξιλόγιο πριν αρχίσει η επεξεργασία. Η τήρηση ενός κεντρικού ή τοπικού συνόλου αντιστοιχίσεων μεταξύ των λεξιλογίων για όλες τις πηγές δεδομένων είναι αδύνατη λόγω του μεγέθους και της συνεχούς εξέλιξης του Ιστού Δεδομένων. Για το λόγο αυτό είναι αναγκαία η ανάπτυξη εφαρμογών που θα ολοκληρώνουν σταδιακά και κατανεμημένα τις πληροφορίες.

Το *R2R Framework* (Bizer & Schultz, 2010) αποτελεί το πιο ολοκληρωμένο μέχρι ώρας εργαλείο σταδιακής ανακάλυψης νέων πληροφοριών. Βασίζεται στην αρχιτεκτονική *Pay-as-you-Go* που πρεσβεύει την ομαλή ολοκλήρωση δεδομένων και θα αναλυθεί σε επόμενο κεφάλαιο. Η εύρεση δεδομένων αγνώστου λεξιλογίου οδηγεί σε αναζήτηση στο Διαδίκτυο για αντιστοιχίες με το συγκεκριμένο λεξιλόγιο και στη συνέχεια σε εφαρμογή των αντιστοιχίσεων για τη μετατροπή στο τοπικό σχήμα.

Το R2R Framework χρησιμοποιεί τη γλώσσα *R2R Mapping Language* για τον ορισμό των αντιστοιχίσεων που έχει τα ακόλουθα χαρακτηριστικά:

- *Λεπτομερής αντιστοίχιση όρων:* Η μίξη όρων διαφορετικών λεξιλογίων κάνει αναγκαία την υψηλή εκφραστικότητα της γλώσσας σχετικά με την αντιστοιχία των όρων. Αυτό επιτρέπει τον ευέλικτο συνδυασμό αντιστοιχίσεων.
- *Διασύνδεση και ανακάλυψη:* Κάθε αντιστοιχία όρου πρέπει να αναγνωρίζεται με το δικό της URI ώστε να μπορεί να συνδεθεί με ορισμούς όρων και συνόλων δεδομένων. Έτσι, είναι δυνατή η ανακάλυψη νέων όρων με την ακολουθία των URIs.

- *Εκφραστικότητα:* Η γλώσσα χρειάζεται να παρέχει δυνατότητα δομικών μετασχηματισμών στα δεδομένα καθώς και μετατροπών στις τιμές των δεδομένων.
- *Αντιστοιχίσεις σε επίπεδο δεδομένων και σε επίπεδο λεξιλογίου:* Διαφορετικές πηγές δεδομένων χρησιμοποιούν διαφορετικά μοτίβα για την αναπαράσταση της ίδιας πληροφορίας. Για παράδειγμα μπορεί να χρησιμοποιούνται διαφορετικές μονάδες μέτρησης ή διαφορετική σειρά περιγραφής του ονόματος κάποιου ανθρώπου (πρώτα το όνομα και μετά το επώνυμο ή το αντίθετο). Το χαρακτηριστικό αυτό συμπληρώνει το χαρακτηριστικό της Εκφραστικότητας.

Στο R2R Framework έχει αναπτυχθεί μια μέθοδος σύνθεσης αντιστοιχίσεων σε περίπτωση πολλαπλών παρόχων. Η διαδικασία χρειάζεται να ανταποκρίνεται στις εξής ανάγκες:

- *Σύνθεση σε επίπεδο όρων:* Η μέθοδος θα πρέπει να παράγει αντιστοιχίσεις βασισμένη σε όλο το εύρος αντιστοιχίσεων που έχουν ανακαλυφθεί. Αν δεν υπάρχει απευθείας αντιστοιχία για κάποιον όρο τότε χρειάζεται να δημιουργηθεί αλυσίδα αντιστοιχίσεων.
- *Κριτήρια ποιότητας αντιστοιχίσεων:* Λόγω της αμφίβολης ποιότητας των δημοσιευμένων δεδομένων είναι αναγκαία η εφαρμογή μετρικών στις πηγές των αντιστοιχίσεων ώστε να παράγονται τα καλύτερα δυνατά αποτελέσματα.

2.5 Η ποιότητα στο χώρο των Δεδομένων

Η δυνατότητα της ελεύθερης δημοσίευσης πληροφορίας στον ιστό των Διασυνδεδεμένων Δεδομένων έχει ως αποτέλεσμα την εισαγωγή της ποιότητας ως βασικό πεδίο επιστημονικής μελέτης. Η ποιότητα της πληροφορίας είναι υποκειμενική και βασίζεται σε διαφορετικούς παράγοντες όπως η ακρίβεια, η πληρότητα, η συνέπεια, η αντικειμενικότητα κ.α. (Bizer & Cyganiak, 2006). Τα Διασυνδεδεμένα Δεδομένα, όπως και οι περισσότερες διαδικτυακές πηγές πληροφορίας, δεν αντιπροσωπεύουν πλήρως την πραγματικότητα με αποτέλεσμα η παρεχόμενη πληροφορία (Bleiholder & Naumann, 2008) να είναι λανθασμένη, ημιτελής ή ασυνεπής. Για το λόγο αυτό έχουν αναπτυχθεί πολλές μετρικές ποιότητας που πηγάζουν από το χώρο της Θεωρίας Πληροφορίας αλλά και μετρικές που εφαρμόζονται σε διαδικτυακά πληροφοριακά συστήματα.

2.5.1 Μετρικές ποιότητας και διαλογής Δεδομένων

Η αξιολόγηση της ποιότητας μιας πληροφορίας είναι η διαδικασία μέτρησης της ικανοποίησης του καταναλωτή που τη χρησιμοποιεί. Η αξιολόγηση γίνεται με μετρικές σχετικές με το σκοπό χρήσης της πληροφορίας. Οι μετρικές βασίζονται σε δείκτες ποιότητας και υπολογίζουν το βαθμό αξιολόγησης χρησιμοποιώντας συναρτήσεις βαθμονόμησης. Οι δείκτες ποιότητας είναι ποικίλοι και περιλαμβάνουν μετα-πληροφορίες σχετικά με τις

συνθήκες κάτω από τις οποίες δημιουργήθηκε η πληροφορία, τον πάροχο πληροφορίας ή τη βαθμολογία της πληροφορίας από τον ίδιο το χρήστη ή άλλους χρήστες. Οι μετρικές μπορούν να χωριστούν σε τρεις κατηγορίες ανάλογα με το είδος πληροφορίας που χρησιμοποιείται ως δείκτης ποιότητας:

- *Μετρικές βασισμένες στο περιεχόμενο (content-based metrics):* Χρησιμοποιούν την ίδια την πληροφορία ως δείκτη ποιότητας. Αναλύουν το περιεχόμενο της πληροφορίας ή το συγκρίνουν με σχετική πληροφορία. Για παράδειγμα σε περιπτώσεις γλωσσικού περιεχομένου γίνεται χρήση τεχνικών κειμενικής ανάλυσης.
- *Μετρικές βασισμένες στα συμφραζόμενα (context-based metrics):* Χρησιμοποιούν μετα-πληροφορίες σχετικά με το περιεχόμενο της πληροφορίας και τις συνθήκες κάτω από τις οποίες δημιουργήθηκε η πληροφορία ως δείκτη ποιότητας. Ένας σημαντικός δείκτης είναι η αξιοπιστία του παρόχου. Ακόμη, οι μετα-πληροφορίες μπορούν να συνδυαστούν και με το γενικότερο πλαίσιο της εφαρμογής που δέχεται τις πληροφορίες.
- *Μετρικές βασισμένες σε βαθμολογία (rating based metrics):* Βασίζονται σε βαθμολογίες επί της πληροφορίας, των πηγών πληροφορίας ή των παρόχων πληροφορίας. Η βαθμολογία μπορεί να προέρχεται από τους καταναλωτές πληροφορίας ή ειδικούς.

Η αξιολόγηση της πληροφοριακής ποιότητας δεν είναι πάντοτε ακριβής και καταλήγει σε χαμηλής αξίας πληροφορία για τους χρήστες. Ωστόσο, οι τελευταίοι είναι ανεκτικοί μέχρι ενός βαθμού στο γεγονός αυτό. Η γρήγορη πρόσβαση σε τεράστιες ποσότητες δεδομένων είναι σημαντικότερη σε σχέση με την ενόχληση που προκύπτει από την εμφάνιση μη σχετικών πληροφοριών.

Οι πολιτικές διαλογής δεδομένων είναι ευριστικές μέθοδοι που αποφασίζουν αν θα δεχθούν ή όχι κάποια πληροφορία. Αποτελούνται από μετρικές αξιολόγησης της ποιότητας και μια συνάρτηση απόφασης που συναθροίζει τα αποτελέσματα των μετρικών και καταλήγει στο αν η πληροφορία καλύπτει τις προδιαγραφές του χρήστη. Η συνάθροιση προκύπτει από τη βαρύτητα που έχει δοθεί σε κάθε μετρική. Η επιλογή των μετρικών ποιότητας σε μια πολιτική διαλογής δεν είναι πάντα προφανής και συνήθως εξαρτάται από τους ακόλουθους παράγοντες:

- *Διαθεσιμότητα δεικτών ποιότητας:* Η διαθεσιμότητα των δεικτών ποιότητας καθορίζει αν θα χρησιμοποιηθεί η μετρική που τους περιλαμβάνει. Για παράδειγμα, η επικαιρότητα της πληροφορίας είναι διαθέσιμη σε πολλές περιπτώσεις όχι, όμως, η αντικειμενικότητα ή η ακρίβεια.
- *Ποιότητα των δεικτών ποιότητας:* Η επιλογή των μετρικών ποιότητας επηρεάζεται από την ποιότητα των διαθέσιμων δεικτών.

- *Κατανόηση:* Ο παράγοντας αυτός είναι πολύ σημαντικός γιατί ο χρήστης εμπιστεύεται τα αποτελέσματα μετρικών που μπορεί να τις αντιληφθεί. Για το λόγο αυτό οι πιο απλές και κατανοητές μετρικές προτιμώνται συχνότερα.
- *Υποκειμενικά κριτήρια:* Ο καταναλωτής της πληροφορίας είναι πιθανό να έχει συγκεκριμένες προτιμήσεις σχετικά με το ποιες μετρικές θα χρησιμοποιηθούν.

Στο πεδίο της ποιότητας δεδομένων έχουν αναπτυχθεί αρκετές εφαρμογές. Για την παρούσα διπλωματική μελετήθηκε το *WIQA-Information Quality Assessment Framework*. Η εφαρμογή μπορεί να ενσωματωθεί σε μεγαλύτερα συστήματα και παρέχει τη δυνατότητα διαλογής των δεδομένων με βάση πολλές διαφορετικές πολιτικές. Το WIQA είναι σχεδιασμένο ώστε να παρέχει μια ευέλικτη αναπαράσταση της πληροφορίας μαζί με την μετα-πληροφορία που τη συνοδεύει, να επιτρέπει στους χρήστες να εφαρμόζουν διαφορετικές πολιτικές διαλογής και να παρέχει επεξηγήσεις σχετικά με αυτές. Βασίζεται στο μοντέλο του *Named Graph* που αποτελεί γενίκευση του RDF μοντέλου. Πρόκειται για μια συλλογή από RDF τριπλέτες στην οποία μπορεί να αποδοθεί κάποιο URI ώστε να είναι δυνατός ο σχηματισμός μετα-πληροφορίας για τους γράφους. Ακόμη, χρησιμοποιείται η γλώσσα *WIQA-PL* μέσω της οποίας ορίζονται οι δείκτες ποιότητας, οι μετρικές που τους χρησιμοποιούν και ο ορισμός μιας συνάρτησης απόφασης που βασίζεται στις μετρικές.

2.5.2 Συγχώνευση Δεδομένων

Στα πλαίσια της ολοκλήρωσης των δεδομένων, η συγχώνευση ορίζεται (Bleiholder & Naumann, 2008) ως η διαδικασία συνένωσης πολλών εγγραφών που αναπαριστούν το ίδιο αντικείμενο του φυσικού κόσμου σε μια απλή, συνεπή και καθαρή αναπαράσταση. Η συγχώνευση των δεδομένων αποτελεί συνήθως το τρίτο βήμα στη διαδικασία ολοκλήρωσης αφού προηγηθούν η αντιστοίχιση λεξιλογίων και το ταίριασμα των δεδομένων. Στο βήμα αυτό αντιμετωπίζονται αντιθέσεις στα δεδομένα που μπορεί να υπάρχουν ήδη στις πηγές ή να προέκυψαν από τη μέχρι ώρας ολοκλήρωση. Για παράδειγμα είναι πιθανή η αντιστοίχιση δυο ταυτόσημων ιδιοτήτων από διαφορετικά λεξιλόγια κι έτσι η κανονικοποιημένη ιδιότητα να έχει δυο διαφορετικές τιμές. Ακόμη, είναι δυνατό δυο ίδιες ιδιότητες να έχουν διαφορετικές τιμές σε διαφορετικά σύνολα δεδομένων. Οι (Bleiholder & Naumann, 2008) περιγράφουν ένα σύστημα που συγχωνεύει σχεσιακά δεδομένα και περιλαμβάνει τρεις κατηγορίες στρατηγικών αντιμετώπισης αντιθέσεων:

- *Στρατηγικές που αγνοούν τις συγκρούσεις:* Το σύστημα δεν αντιμετωπίζει τη σύγκρουση στα δεδομένα και καθιστά το χρήστη υπεύθυνο για τη διευθέτησή της. Ο χρήστης συνήθως επιλέγει μια από τις αντικρουόμενες τιμές που υπάρχουν για κάποια ιδιότητα.

- *Στρατηγικές αποφυγής συγκρούσεων:* Το σύστημα εφαρμόζει μια προκαθορισμένη καθολική απόφαση για όλα τα είδη συγκρούσεων.
- *Στρατηγικές επίλυσης συγκρούσεων:* Εδώ το σύστημα αποφασίζει για το ποια τιμή δεδομένων είναι η καταλληλότερη για να επιλεχθεί. Για παράδειγμα, είναι δυνατό να αντικατασταθούν οι πολλαπλές τιμές για μια ιδιότητα από το μέσο όρο τους.

Οι (Bleiholder & Naumann, 2008) ορίζουν, ακόμη, ορισμένες έννοιες σχετικές με τη συγχώνευση δεδομένων που είναι ανεξάρτητες από τη δομή της πληροφορίας:

- *Πληρότητα:* Σε επίπεδο σχήματος ή λεξιλογίου, ένα σύνολο δεδομένων είναι πλήρες αν περιέχει όλες τις ιδιότητες που χρειάζονται για τη σωστή αναπαράσταση της πληροφορίας. Σε επίπεδο δεδομένων, ένα σύνολο δεδομένων είναι πλήρες αν περιέχει όλες τις απαραίτητες αναπαραστάσεις πληροφορίας[12]. Η πληρότητα κρίνεται μόνο αν είναι γνωστός ο αριθμός των απαραίτητων ιδιοτήτων και αναπαραστάσεων.
- *Περιεκτικότητα:* Σε επίπεδο σχήματος, ένα σύνολο δεδομένων είναι περιεκτικό αν δεν περιέχει μη χρήσιμες ιδιότητες, δηλαδή δυο ταυτόσημες ιδιότητες με διαφορετικά ονόματα. Σε επίπεδο δεδομένων η περιεκτικότητα εξασφαλίζεται όταν δεν υπάρχουν ταυτόσημες αναπαραστάσεις πληροφορίας με διαφορετικά αναγνωριστικά[13].
- *Συνέπεια:* Ένα σύνολο δεδομένων είναι συνεπές αν δεν έχει καμία αντικρουόμενη πληροφορία.

Το *Sieve* (Mendes, Mühleisen, & Bizer, 2012) είναι μια εφαρμογή που χρησιμοποιεί τις παραπάνω βασικές αρχές της συγχώνευσης ώστε να αντιμετωπίσει περιπτώσεις αντιφάσεων στα δεδομένα. Αποτελεί το τρίτο κατά σειρά συστατικό στοιχείο ενός ενιαίου συστήματος ολοκλήρωσης. Πέρα από τη συγχώνευση, η εφαρμογή έχει τη δυνατότητα να εφαρμόσει και πολιτικές ποιότητας που βασίζονται στο WIQA. Ο καθορισμός τόσο των στρατηγικών διευθέτησης συγκρούσεων στα πλαίσια της συγχώνευσης όσο και των μετρικών και δεικτών ποιότητας γίνεται με χρήση της γλώσσας XML.

2.6 Ολοκλήρωση Δεδομένων

Η ολοκλήρωση των δεδομένων αποτελεί το επιστέγασμα όλων των ενεργειών που αναφέρθηκαν στις προηγούμενες ενότητες. Ξεκινά με τη συλλογή πληροφοριών από ποικίλες πηγές, συνεχίζει με το ταίριασμα και τη σύνδεσή τους (στην περίπτωση των Διασυνδεδεμένων Δεδομένων) και καταλήγει στη διαλογή τους ώστε ο τελικός χρήστης να

[12] *Σε μια Βάση Δεδομένων υπάρχει πληρότητα αν συμπεριλαμβάνονται όλες οι απαραίτητες εγγραφές και στα Διασυνδεδεμένα Δεδομένα αν περιλαμβάνονται όλες οι RDF τριπλέτες.*

[13] *Πρωτεύοντα κλειδιά στις σχεσιακές βάσεις και URIs για RDF δεδομένα.*

έχει μια όσο το δυνατόν καλύτερη εικόνα των πληροφοριών που αναζητά. Έχουν προταθεί αρκετά μοντέλα ολοκλήρωσης δεδομένων με επικρατέστερο το Pay-As-You-Go στο οποίο βασίζεται μέχρι στιγμής και το οικοδόμημα των Διασυνδεδεμένων Δεδομένων.

Οι (Madhavan, et al., 2007) παρουσιάζουν την Pay-as-You-Go ως πρότυπο μοντέλο ολοκλήρωσης για την εταιρία Google. Ωστόσο, οι βασικές αρχές του μοντέλου εφαρμόζονται ήδη στα Διασυνδεδεμένα Δεδομένα. Στα παραδοσιακά συστήματα ολοκλήρωσης υπήρχαν δυο βασικά χαρακτηριστικά. Αρχικά, η πρόσβαση στα δεδομένα γινόταν μέσω ενός καθολικού ενδιάμεσου σχήματος. Το ενδιάμεσο σχήμα χρειαζόταν να αντιστοιχιστεί με καθένα από τα σχήματα των πηγών δεδομένων μέσω γλωσσών αντιστοίχισης. Ακόμη, οι ερωτήσεις στα δεδομένα γίνονταν με γλώσσες όπως η SQL ως προς το ενδιάμεσο σχήμα. Τα ερωτήματα στη συνέχεια μετασχηματίζονταν με βάση τις αντιστοιχίσεις ώστε να καθοριστούν ποιες πηγές είναι σχετικές με το κάθε ερώτημα και πως θα ενωθούν τα δεδομένα από τις διαφορετικές πηγές.

Το νέο μοντέλο ολοκλήρωσης υποστηρίζει την κατηγοριοποίηση σχημάτων (*schema clustering*). Αυτό σημαίνει πως δεν υπάρχει ένα καθολικό ενδιάμεσο σχήμα αλλά πολλά σχήματα χωρισμένα σε εννοιολογικές κατηγορίες. Οι αντιστοιχίσεις ανάμεσα στις διάφορες πηγές δεδομένων δεν είναι απόλυτες εξαιτίας της ανομοιογένειας των πληροφοριών και γίνονται κατά προσέγγιση. Οι βασικοί τρόποι σχηματισμού των αντιστοιχιών είναι είτε η χρήση αυτόματων μεθόδων είτε ο καθορισμός τους από τον άνθρωπο.

Οι ερωτήσεις στα δεδομένα γίνονται με λέξεις κλειδιά και σπανίως με δομημένο τρόπο. Οπότε, τα ερωτήματα πρώτα αποκτούν δομή και στη συνέχεια επιλέγονται οι σχετικές με αυτά πηγές δεδομένων. Λόγω της κατηγοριοποίησης των σχημάτων και της αβεβαιότητας των αντιστοιχίσεων η εύρεση σχετικών πηγών είναι πιο σύνθετη. Η παραπάνω διαδικασία αναφέρεται ως *δρομολόγηση ερωτήματος (query routing)***.** Η αξιολόγηση των απαντήσεων γίνεται με βαθμολογία (*ranking*) λόγω της αβεβαιότητας που υπεισέρχεται στη διαδικασία εύρεσής τους.

Η ολοκλήρωση των δεδομένων γίνεται σταδιακά. Καθώς όλο και περισσότερες πηγές συμπεριλαμβάνονται στο σύστημα τόσο αυξάνεται η εννοιολογική σύνδεση μεταξύ τους. Κάθε χρονική στιγμή τα ερωτήματα απαντώνται με βάση τις υπάρχουσες πληροφορίες του συστήματος και όχι μιας καθολικής γνώσης, όπως στα παραδοσιακά συστήματα ολοκλήρωσης. Η σταδιακή εννοιολογική σύνδεση γίνεται με μηχανισμούς αυτόματης εξαγωγής αντιστοιχίσεων και με επαλήθευσή τους από τους χρήστες. Καθ' όλη τη διάρκεια της ολοκλήρωσης χρειάζεται να μοντελοποιηθεί η αβεβαιότητα ώστε να είναι όσο το δυνατό μικρότερη η επίδρασή της.

Η ολοκλήρωση στο χώρο των Διασυνδεδεμένων Δεδομένων ακολουθεί τις προηγούμενες βασικές αρχές. Ωστόσο, ένα βασικό σημείο διαφοροποίησης είναι η

εννοιολογική αντιστοίχιση των δεδομένων από τις εφαρμογές. Οι τελευταίες χρειάζεται να ανακαλύψουν και όχι να παράξουν σημασιολογικές αντιστοιχίες. Όσο τέτοιες αντιστοιχίσεις δημοσιεύονται στον Ιστό από τρίτους τόσο η ολοκλήρωση των δεδομένων θα γίνεται πιο πλήρης. Αυτή είναι η παρούσα κατάσταση στον Ιστό Δεδομένων όσο αφορά εφαρμογές που ολοκληρώνουν δεδομένα (Heath & Bizer, 2011) . Φυσικά, έχουν αναπτυχθεί συστήματα που παράγουν σημασιολογικές αντιστοιχίες αλλά ανεξάρτητα από τα πλαίσια της ολοκλήρωσης. Η σταδιακή δημοσίευση και ανακάλυψη αντιστοιχιών εισάγει και μια διαφορετική διάσταση στην ολοκλήρωση: το οικοδόμημα των Διασυνδεδεμένων Δεδομένων στηρίζεται σε πολλούς πυλώνες (Bizer & Schultz, 2010). Η έκδοση των αντιστοιχίσεων γίνεται από τρίτους παρόχους που μπορεί να είναι διαφορετικοί από τους καταναλωτές πληροφορίας και τους εκδότες πηγών δεδομένων. Η προσπάθεια, λοιπόν, της ολοκλήρωσης δεν βαραίνει αποκλειστικά τις εφαρμογές, όπως συμβαίνει σήμερα με τα υπάρχοντα συστήματα ολοκλήρωσης.

Το *LDIF* (Schultz, Matteini, Isele, Bizer, & Becker, 2012) αποτελεί μια πρόσφατη ανοικτού κώδικα εφαρμογή ολοκλήρωσης Διασυνδεδεμένων Δεδομένων, υλοποιημένη στη γλώσσα Scala[14]. Περιλαμβάνει και τα πέντε στάδια ολοκλήρωσης (πρόσβαση στα δεδομένα, μετασχηματισμός σε ενιαίο σημασιολογικό σχήμα, σύνδεση, συγχώνευση και διαλογή). Μπορεί να χρησιμοποιηθεί από άλλες εφαρμογές χειρισμού δεδομένων ως υποσύστημα παροχής ολοκληρωμένης πληροφορίας ώστε να είναι δυνατή η περαιτέρω επεξεργασία της. Το LDIF περιλαμβάνει αρκετά από τα εργαλεία που αναλύθηκαν σε προηγούμενες ενότητες, όπως το R2R Framework, το SILK και το Sieve.

Πέρα από τις ενέργειες ολοκλήρωσης παρέχει δυνατότητες χρονοπρογραμματισμού των διαδικασιών. Υποστηρίζει την πρόσβαση σε RDF δεδομένα είτε με τη χρήση crawling είτε με την απευθείας εισαγωγή αρχείων RDF. Η έξοδος του συστήματος είναι ένα αρχείο που περιέχει όλα τα δεδομένα που εισήχθησαν στο σύστημα στην ενιαία εννοιολογική μορφή. Τα δεδομένα αυτά κατηγοριοποιούνται σε γράφους και κάθε γράφος συνοδεύεται από τα αποτελέσματα των μετρικών ποιότητας και μετα-πληροφορία σχετικά με την προέλευσή του. Το LDIF ελέγχει τη ροή δεδομένων και την αποθήκευση των ενδιάμεσων αποτελεσμάτων. Για τον παραλληλισμό της επεξεργασίας τα δεδομένα χωρίζονται σε γράφους που αναπαριστούν πηγές δεδομένων και χρησιμοποιούνται νήματα για την επεξεργασία τους. Ωστόσο, δεν περιλαμβάνει την ολοκλήρωση δεδομένων από άλλες πηγές όπως σχεσιακές βάσεις και αρχεία CSV ή Excel. Στο σημείο αυτό διαφοροποιείται το προτεινόμενο σύστημα ολοκλήρωσης της παρούσας διπλωματικής.

[14] *http://www.scala-lang.org/*

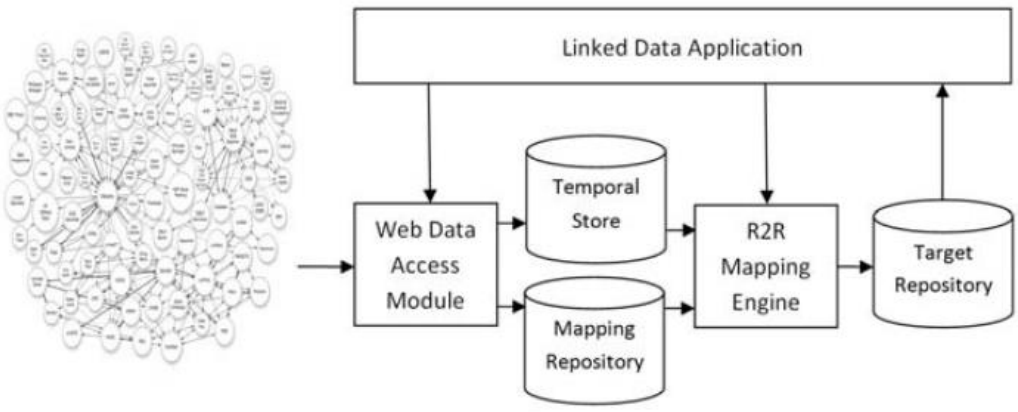
3

Εργαλεία

Στην ενότητα αυτή παρουσιάζονται αναλυτικά τα εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη του συστήματος ολοκλήρωσης Διασυνδεδεμένων Δεδομένων.

3.1 R2R Framework

Το R2R Framework χρησιμοποιείται για τη σημασιολογική αντιστοίχιση ανάμεσα στα διαφορετικά λεξιλόγια με τα οποία αναπαρίστανται τα δεδομένα. Η εφαρμογή έχει υλοποιηθεί στη γλώσσα Java και η αρχιτεκτονική της φαίνεται παρακάτω:



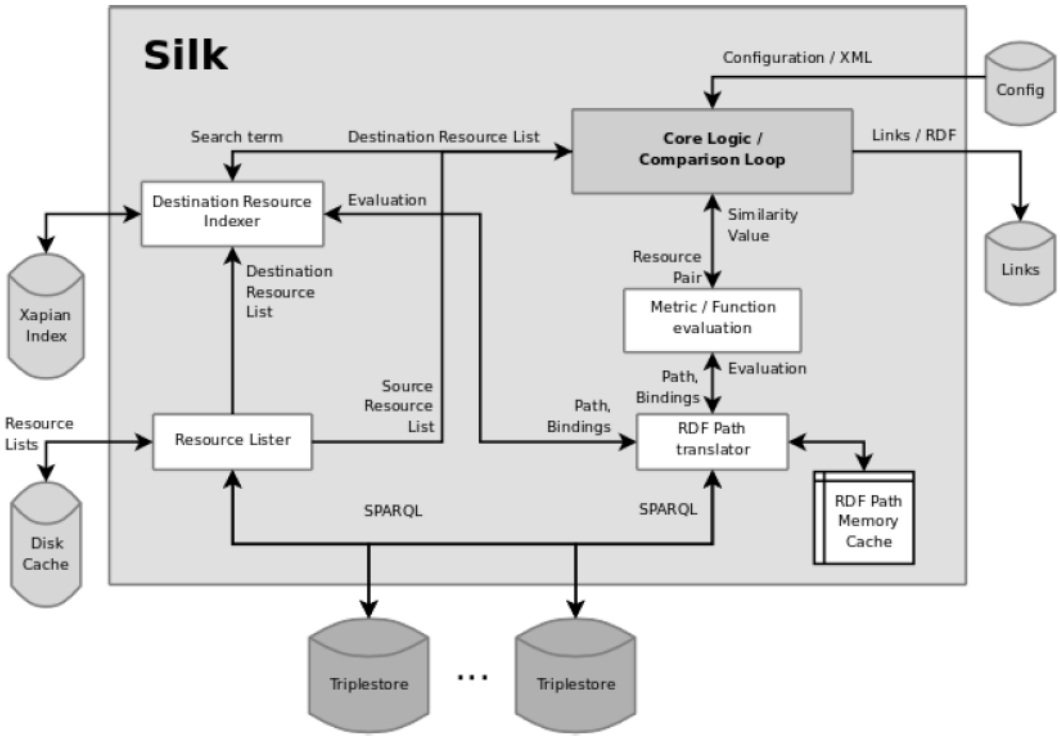
Εικόνα 3.1: Η αρχιτεκτονική του R2R Framework

Αρχικά, χρησιμοποιείται μια μονάδα πρόσβασης στα δεδομένα που συλλέγει τις πληροφορίες ακολουθώντας RDF συνδέσμους. Στη συνέχεια τα δεδομένα αποθηκεύονται σε

μια προσωρινή θέση (*Temporal Store*) που μπορεί να είναι είτε η τοπική μνήμη είτε κάποια βάση RDF σε μορφή γράφων. Κάθε γράφος που ονομάζεται *dataset graph* (DSG_n) αντιστοιχεί σε μια πηγή δεδομένων και προσδιορίζεται από ένα URI. Οι αντιστοιχίσεις που ανακαλύπτονται στον Ιστό αποθηκεύονται στην *αποθήκη αντιστοιχίσεων (Mapping Repository)*. Πέρα από τις αντιστοιχίες που ανακαλύπτονται με τους RDF συνδέσμους η εφαρμογή εφαρμόζει ερωτήματα σε μηχανές αναζήτησης ώστε να υπάρχει μεγαλύτερο εύρος γνώσης. Ακόμη, η *μηχανή των αντιστοιχίσεων (R2R Mapping Engine)* περιέχει το λεξιλόγιο κοινής αναπαράστασης και σε αυτό μετατρέπονται όλα τα εισερχόμενα δεδομένα. Τα μετασχηματισμένα δεδομένα αποθηκεύονται στο *Target Repository* και εφαρμόζονται ερωτήματα σε αυτά.

3.2 Silk Framework

Το Silk Framework χρησιμοποιείται για την αναγνώριση URIs που περιγράφουν το ίδιο αντικείμενο του φυσικού κόσμου. Υπάρχουν πολλοί τρόποι εκτέλεσης της εφαρμογής: τοπική εκτέλεση, χρήση της τεχνικής Map – Reduce καθώς και χρήση ενός γραφικού περιβάλλοντος ορισμού των συνδέσεων. Στα πλαίσια του συστήματος ολοκλήρωσης θα χρησιμοποιηθεί το API που παρέχεται και θα ορίζεται γραφικά κάθε φορά το αρχείο προδιαγραφής των συνδέσεων στη γλώσσα Silk-LSL. Η αρχιτεκτονική του συστήματος φαίνεται στην παρακάτω εικόνα:



Εικόνα 3.2: Η αρχιτεκτονική του SILK

Αρχικά, συλλέγονται οι λίστες με τα URIs που πρόκειται να συγκριθούν (*Resource Lists*). Η συλλογή των URIs που αποτελούν τις «πηγές» γίνεται με ερωτήσεις σε SPARQL endpoints και τοπική αποθήκευση των απαντήσεων. Τα URIs που αποτελούν τους «στόχους» δεικτοδοτούνται ώστε να μπορούν να αναζητηθούν σε αυτά πληροφορίες όπως ιδιότητες. Για κάθε URI – πηγή ελέγχονται όλα τα υποψήφια URIs – στόχοι. Σε κάθε σύγκριση αποτιμώνται τα κριτήρια ομοιότητας που έχουν οριστεί από το χρήστη. Οι τιμές που είναι στη μορφή *RDF Path* (Volz, Bizer, Gaedke, & Kobilarov, 2009) μετατρέπονται σε SPARQL ερωτήματα (*RDF Path Translator*) και στέλνονται στο κατάλληλο SPARQL endpoint για εκτέλεση. Τα αποτελέσματα αποθηκεύονται τοπικά. Αν ένα ζευγάρι URIs αποτιμάται βάσει των κριτηρίων ομοιότητας πάνω από το προκαθορισμένο κατώφλι τότε αποθηκεύεται στη μνήμη. Μετά από όλες τις συγκρίσεις για ένα URI – πηγή ένας προκαθορισμένος αριθμός αποτελεσμάτων που παρουσιάζουν τη μεγαλύτερη ομοιότητα γράφεται στο αρχείο εξόδου.

Η σύγκριση όλων των ζευγαριών ανάμεσα στις οντότητες δυο πηγών δεδομένων είναι δαπανηρή τόσο από άποψη χρονικής απόδοσης όσο και μνήμης. Για το λόγο αυτό χρησιμοποιείται η δεικτοδότηση των οντοτήτων που βρίσκονται στο στόχο σύμφωνα με κάποια ιδιότητα. Με βάση τη δεικτοδότηση αυτή αναζητούνται κατάλληλες οντότητες ώστε να συγκριθούν. Έτσι, μειώνεται το πλήθος των υποψήφιων προς σύγκριση οντοτήτων αλλά είναι πιθανή η απώλεια κάποιων συνδέσεων γιατί η αναζήτηση δεν είναι βέλτιστη.

3.3 Εξωτερικές Βιβλιοθήκες

Αρκετές από τις λειτουργίες της εφαρμογής βασίζονται σε εξωτερικές βιβλιοθήκες και frameworks. Συγκεντρωτικά, οι εξωτερικές μονάδες λογισμικού που χρησιμοποιήθηκαν είναι:

- *Apache Jena*: Το framework αυτό είναι ζωτικής σημασίας για πολλά συστήματα διαχείρισης RDF Δεδομένων. Στην προκειμένη περίπτωση χρησιμοποιήθηκε για την ανάγνωση και ενσωμάτωση RDF Δεδομένων καθώς και για την επεξεργασία και εκτέλεση SPARQL ερωτημάτων.
- *Apache POI*: Σημαντικό framework που, στα πλαίσια της εφαρμογής, χρησιμοποιείται για την ανάγνωση excel αρχείων.
- *Βιβλιοθήκη MySQLConnector*: Η βιβλιοθήκη αυτή χρησιμοποιείται για την επικοινωνία της εφαρμογής με σχεσιακές βάσεις δεδομένων (σύνδεση, εισαγωγή sql ερωτημάτων, εξαγωγή και επεξεργασία αποτελεσμάτων).
- *Βιβλιοθήκη OpenCSV*: Βιβλιοθήκη για την ανάγνωση csv αρχείων.

4

Θεωρητικό υπόβαθρο

Στην ενότητα αυτή θα αναλυθούν τα βασικά θεωρητικά μοντέλα στα οποία στηρίζεται το σύστημα ολοκλήρωσης. Πιο συγκεκριμένα, θα γίνει εμφανές πως ένα URI που περιγράφει κάποιον πόρο μπορεί να αναζητηθεί και έτσι να έχουμε σύνδεση των δεδομένων, πως μπορούν να περιγραφούν τα δεδομένα με λεξιλόγια, ποιες είναι οι γλώσσες που θα χρησιμοποιηθούν από το σύστημα για τον σημασιολογικό μετασχηματισμό και τη σύνδεση των δεδομένων και ,τέλος, ποιο είναι το γενικό μοντέλο αρχιτεκτονικής που θα βασιστεί η προτεινόμενη εφαρμογή.

4.1 Το μοντέλο RDF

4.1.1 Ο συνδυασμός του μοντέλου RDF και του πρωτοκόλλου HTTP

Όπως έχει προαναφερθεί, κάθε URI αναζητείται με το πρωτόκολλο HTTP και παρέχει μια περιγραφή του αντικειμένου που προσδιορίζει. Η μέθοδος αυτή εφαρμόζεται στα HTML έγγραφα και όμοια μπορεί να εφαρμοστεί σε URIs που προσδιορίζουν άλλου είδους αντικείμενα, όπως ανθρώπους, έννοιες κλπ. Οι περιγραφές των αντικειμένων ενσωματώνονται σε αρχεία HTML. Αυτές που πρόκειται να διαβαστούν από τους ανθρώπους βρίσκονται σε μορφή HTML και εκείνες που διαβάζονται από υπολογιστές αναπαριστώνται ως RDF.

Η σύνδεση του πρωτοκόλλου HTTP και του μοντέλου RDF γίνεται με μια διαδικασία που ονομάζεται *διαπραγμάτευση περιεχομένου (context negotiation)* (Heath & Bizer, 2011). Σύμφωνα με αυτή, κάθε HTTP αίτημα προσδιορίζει τι είδους αναπαράσταση περιεχομένου επιθυμεί: αν δηλώνεται περιεχόμενο σε HTML τότε ο εξυπηρετητής του αιτήματος επιστρέφει δεδομένα σε HTML αλλιώς αν προσδιορίζεται περιεχόμενο σε RDF στέλνεται αντίστοιχη απάντηση. Υπάρχουν δυο τρόποι που κάνουν ένα URI να μπορεί να αναζητηθεί με το πρωτόκολλο HTTP.

Η πρώτη στρατηγική ονομάζεται *303 URIs* και σύμφωνα με αυτή ο εξυπηρετητής στέλνει μια απάντηση με κωδικό `303 See Other` και το URI του εγγράφου που περιγράφει το αντικείμενο του αιτήματος. Αυτό ονομάζεται *303 ανακατεύθυνση (303 redirect)*. Ο πελάτης λαμβάνει το νέο URI και μέσω του πρωτοκόλλου HTTP παίρνει το έγγραφο που περιγράφει το αντικείμενο. Ο δεύτερος τρόπος ονομάζεται *hash URI* και βασίζεται στο (προαιρετικό) χαρακτηριστικό σύμβολο # των URIs που επιτρέπει τον χωρισμό τους σε κύριο και δευτερεύον μέρος. Το τελευταίο ονομάζεται και *αναγνωριστικό τεμαχίου (fragment identifier)*. Όταν ένα πελάτης θέλει ένα hash URI τότε το πρωτόκολλο HTTP απαιτεί το αναγνωριστικό τμήμα του URI να αποσπαστεί πριν γίνει το αίτημα στον εξυπηρετητή. Αυτό σημαίνει πως το hash URI δεν μπορεί να ληφθεί απευθείας και, επομένως, δεν περιγράφει απαραίτητα ένα διαδικτυακό αρχείο. Αυτό έχει σαν αποτέλεσμα ένα τέτοιο URI να μπορεί να χρησιμοποιηθεί ως αναγνωριστικό για κάποιο αντικείμενο χωρίς να προκληθεί ασάφεια[15].

Τα hash URIs απαιτούν λιγότερες ανταλλαγές HTTP μηνυμάτων και μειώνουν τις χρονικές καθυστερήσεις. Ωστόσο, κάθε φορά που ζητείται ένα hash URI επιστρέφονται όλα τα URIs που έχουν κοινό hash fragment παρόλο που μπορεί ο πελάτης να ενδιαφέρεται για ένα μόνο URI. Αν οι περιγραφές αποτελούνται από πολλές τριπλέτες RDF τότε είναι πιθανό να μεταφέρονται μεγάλος αριθμός μη χρήσιμων δεδομένων. Από την άλλη πλευρά τα 303 URIs είναι ευέλικτα γιατί προσδιορίζουν μοναδικά κάθε αντικείμενο. Για το λόγο αυτό χρησιμοποιούνται σε περιγραφές μεγάλων συνόλων δεδομένων. Τα hash URIs χρησιμεύουν στην αναγνώριση όρων λεξιλογίων τα οποία αποτελούνται συνήθως από περιορισμένο αριθμό τριπλετών. Η απευθείας μεταφορά όλου του λεξιλογίου (ακόμη κι αν ζητείται ένας όρος του) είναι χρήσιμη καθώς είναι πιθανό στη συνέχεια να ζητηθούν κι άλλοι όροι από κάποια εφαρμογή που, όμως, έχουν ήδη μεταφερθεί. Έτσι, περιορίζεται σε κάποιο βαθμό ο αριθμός των μηνυμάτων HTTP.

Οι παραπάνω τεχνικές κάνουν εφικτή την αναζήτηση αντικειμένων αλλά και τη σύνδεσή τους οδηγώντας στο σχηματισμό του χώρου των Διασυνδεδεμένων Δεδομένων.

[15] *Η ασάφεια έγκειται στο γεγονός αν ένα URI αντιπροσωπεύει το ίδιο το αντικείμενο ή κάποια HTML σελίδα που το περιγράφει.*

4.1.2 Η χρήση λεξιλογίων για την αναπαράσταση πληροφορίας

Το μοντέλο RDF παρέχει ένα γενικό και αφηρημένο τρόπο οργάνωσης των δεδομένων και δεν υποστηρίζει την κατηγοριοποίηση με βάση συγκεκριμένο τομέα της ανθρώπινης ζωής (π.χ. οικονομικό, βιολογικό, στατιστικό κλπ.). Για το λόγο αυτό έχουν αναπτυχθεί ταξινομίες, λεξιλόγια και οντολογίες που περιγράφονται από γλώσσες όπως η *OWL (The Web Ontology Language)*[16] ή η *RDFS (RDF Schema)*[17]. Μια λίστα με τα υπάρχοντα λεξιλόγια είναι η *Linked Open Vocabularies (LOV)*[18]. Στα πλαίσια της παρούσας διπλωματικής θα χρησιμοποιηθούν κάποια λεξιλόγια ώστε να μπορεί να γίνει η ολοκλήρωση δεδομένων.

4.1.2.1 Data Cube Vocabulary

Το Data Cube Vocabulary[19] υλοποιήθηκε προκειμένου να είναι δυνατή όχι μόνο η δημοσίευση στον Ιστό Δεδομένων στατιστικής και γενικότερα πολυδιάστατης πληροφορίας αλλά και η σύνδεσή της με άλλες πηγές δεδομένων. Το λεξιλόγιο χρησιμοποιεί το μοντέλο RDF και τις αρχές των Διασυνδεδεμένων Δεδομένων και στηρίζεται στο πρότυπο *SDMX ISO*[20]. Μπορεί να χρησιμοποιηθεί για τη δημοσίευση δεδομένων ερευνών, αρχείων σε μορφή CSV ή Excel αλλά και αναλυτικής πληροφορίας (*OLAP*) που βασίζονται στο θεωρητικό μοντέλο του *κύβου (cube)*.

Μερικά από τα οφέλη της δημοσίευσης πολυδιάστατης πληροφορίας στο Διαδίκτυο είναι τα εξής:

- Οι στατιστικές παρατηρήσεις μπορούν πλέον να συνδέονται με άλλες πηγές δεδομένων. Για παράδειγμα τα διαγράμματα που αναπαριστούν πληροφορία μπορούν να συνδέονται με τις πηγές των δεδομένων που χρησιμοποιούν ώστε να είναι δυνατή η επαλήθευσή τους από τρίτους.
- Τα στατιστικά δεδομένα προσδίδουν αξία σε μεμονωμένες πληροφορίες εφόσον συνδεθούν με αυτές. Έτσι, ο χρήστης της ολοκληρωμένης πλέον πληροφορίας έχει καλύτερη εικόνα για την πραγματικότητα.

[16] *http://www.w3.org/2004/OWL/*

[17] *http://www.w3.org/TR/rdf-schema/*

[18] *http://lov.okfn.org/dataset/lov/*

[19] *http://www.w3.org/TR/vocab-data-cube/*

[20] *http://www.sdmx.org/*

- Η διαδικασία δημοσίευσης από τους παρόχους στατιστικών δεδομένων γίνεται πιο ευέλικτη αφού δεν χρειάζονται ειδικά APIs για την πρόσβαση από εφαρμογές, όπως γινόταν μέχρι σήμερα.

4.2 Η χρήση γλωσσών κατά την ολοκλήρωση Δεδομένων

Στις προηγούμενες ενότητες αναπτύχθηκαν τα στάδια και οι διαδικασίες που λαμβάνουν χώρα κατά τη διάρκεια της ολοκλήρωσης των δεδομένων. Σχεδόν σε όλα τα στάδια χρησιμοποιούνται ειδικές γλώσσες που ορίζουν και μοντελοποιούν τον τρόπο με τον οποίο θα γίνουν οι ενέργειες.

4.2.1 Η γλώσσα R2R Mapping Language

Η συγκεκριμένη γλώσσα είναι παρόμοια με την SPARQL και βασίζεται στο μοντέλο RDF. Χρησιμοποιείται για την εννοιολογική αντιστοίχιση όρων λεξιλογίων, παρέχει λειτουργίες συντακτικής μετατροπής των δεδομένων και είναι σημασιολογικά ανεξάρτητη από τα δεδομένα.

Ένα παράδειγμα αντιστοίχισης στη γλώσσα R2R είναι το ακόλουθο:

```
01: <http://thirdparty.org/mappingDbpediaPersonToFoafPerson>
02: rdf:type r2r:Mapping ;
03: r2r:prefixDefinitions "dbpedia-owl: <http://dbpedia.org/ontology/>.
04: foaf: <http://xmlns.com/foaf/0.1/>" ;
05: r2r:sourcePattern "?SUBJ rdf:type dbpedia-owl:Person" ;
06: r2r:targetPattern "?SUBJ rdf:type foaf:Person" ;
07: dc:creator <http://thirdparty.org/andreas> ;
08: dc:date "2010-06-11"^^xsd:date .
09:
10: <http://thirdparty.org/mappingRuntimeLinkedmdbToFreebase>
11: rdf:type r2r:Mapping ;
12: r2r:prefixDefinitions
13: "movie: <http://data.linkedmdb.org/resource/movie/> .
14: fb: <http://rdf.freebase.com/ns/>" ;
15: r2r:sourcePattern "?SUBJ movie:runtime ?runtime" ;
16: r2r:targetPattern "?SUBJ fb:film.film.runtime ?generatedURI
17: fb:film.film_cut.runtime ?'runtime'^^xsd:float" ;
18: r2r:transformation "?generatedURI = concat(?SUBJ, 'Runtime')" ;
19: r2r:sourceDataset <http://mappings.dbpedia.org/r2r/linkedmdbVOID>;
20: r2r:targetDataset <http://mappings.dbpedia.org/r2r/freebaseVOID> ;
21: dc:creator <http://thirdparty.org/andreas> ;
22: dc:date "2010-06-11"^^xsd:date .
```

Η βασική έννοια αντιστοίχισης είναι ο όρος *r2r:Mapping* που έχει ως χαρακτηριστικά την πηγή *r2r:sourcePattern* και τον στόχο *r2r:targetPattern*. Η πηγή (Γραμμή 16) μπορεί να αποτελείται από εκφράσεις παρόμοιες με εκείνες της SPARQL που αποτιμώνται έναντι του Ιστού Δεδομένων. Οι εκφράσεις δεν πρέπει να περιλαμβάνουν μεταβλητές με εξαίρεση την αναγκαία μεταβλητή *?SUBJ* με την οποία αποτιμώνται τα URIs των πόρων. Παρόμοια, ο στόχος χρησιμοποιείται για να παράγει τριπλέτες στο ζητούμενο λεξιλόγιο. Για παράδειγμα

στις γραμμές 16-17 χρησιμοποιείται η έκφραση που χρησιμοποιεί η βάση δεδομένων *Freebase* για την αναπαράσταση ταινιών. Ακόμη, τα *r2r:Mappings* περιέχουν προτάσεις μετασχηματισμού με τις οποίες τροποποιούνται τα δεδομένα πριν ενσωματωθούν στο λεξιλόγιο – στόχο. Προσδιορίζονται με το χαρακτηριστικό *r2r:transformation*. Στη γραμμή 18 η μεταβλητή *?generatedURI* του λεξιλογίου – στόχου είναι η συνένωση της μεταβλητής *?SUBJ* και της λέξης *"Runtime"*.

Η γλώσσα παρέχει επίσης τον τύπο *r2r:hasMapping* προκειμένου να συνδέσει αντιστοιχίσεις με όρους άλλων λεξιλογίων. Στην παρακάτω σύνδεση ενώνεται ο όρος *runtime* της βάσης *DBPedia* με την αντιστοίχιση *FreebaseFilmRuntimeToRuntime* στην οποία έχει αποδοθεί URI. Έτσι, όταν ζητείται ο όρος *runtime* επιστρέφεται και το URI της αντιστοίχισης που χρησιμοποιείται για να βρεθεί η αντιστοίχιση.

```
<http://dbpedia.org/ontology/runtime> r2r:hasMapping
<http://mappings.dbpedia.org/r2r/FreebaseFilmRuntimeToRuntime>
```

Αναλυτικότερες πληροφορίες για τη γλώσσα μπορούν να βρεθούν στον ιστότοπο του R2R Framework[21].

4.2.2 Η γλώσσα Silk-LSL

Η γλώσσα Silk-LSL χρησιμοποιείται για τον ορισμό μεθόδων που προσδιορίζουν αν υπάρχει σημασιολογική σχέση ανάμεσα σε δυο οντότητες, τον καθορισμό των μεταβλητών καθώς και για άλλες λειτουργίες χειρισμού του συστήματος. Στη συνέχεια ακολουθεί ένα αναλυτικό παράδειγμα με το οποίο περιγράφονται βασικά χαρακτηριστικά της γλώσσας.

```
01 <Silk>
02 <DataSource id="dbpedia">
03 <EndpointURI>http://dbpedia.org/sparql</EndpointURI>
04 <Graph>http://dbpedia.org</Graph>
05 <DoCache>1</DoCache>
06 <PageSize>10000</PageSize>
07 </DataSource>
08 <DataSource id="geonames">
09 <EndpointURI>http://localhost:8890/sparql</EndpointURI>
10 </DataSource>
11 <Interlink id="cities">
12 <LinkType>owl:sameAs</LinkType>
13 <SourceDataset dataSource="dbpedia" var="a">
14 <RestrictTo>{?a rdf:type dbpedia:City } UNION { ?a rdf:type dbpedia:PopulatedPlace
}</RestrictTo>
15 </SourceDataset>
16 <TargetDataset dataSource="geonames" var="b">
17 <RestrictTo>?b gn:featureClass gn:P</RestrictTo>
18 </TargetDataset>
19 <LinkCondition>
20 <AVG>
21 <MAX>
22 <Compare metric="jaroSimilarity" optional="1">
23 <Param name="str1" path="?a/rdfs:label[@lang 'en']" />
24 <Param name="str2" path="?b/gn:alternateName[@lang 'en']" />
25 </Compare>
26 <Compare metric="jaroSimilarity" optional="1">
27 <Param name="str1" path="?a/rdfs:label" />
```

[21] *http://wifo5-03.informatik.uni-mannheim.de/bizer/r2r/spec/#specification*

```
28 <Param name="str2" path="?b/gn:name" />
29 </Compare>
30 </MAX>
31 <Compare metric="maxSimilarityInSets" optional="1" weight="3">
32 <Param name="set1" path="?a/foaf:page" />
33 <Param name="set2" path="?b/gn:wikipediaArticle" />
34 <Param name="submetric" value="stringEquality" />
35 </Compare>
36 <MAX>
37 <Match metric="numSimilarity" optional="1">
38 <Param name="num1" path="?a/p:populationEstimate" />
39 <Param name="num2" path="?b/gn:population" />
40 </Match>
41 <Match metric="numSimilarity" optional="1">
42 <Param name="num1" path="?a/dbpedia:populationTotal" />
43 <Param name="num2" path="?b/gn:population" />
44 </Match>
45 </MAX>
46 <Compare metric="numSimilarity" optional="1" weight="0.7">
47 <Param name="num1" path="?a/wgs84_pos:lat" />
48 <Param name="num2" path="?b/wgs84_pos:lat" />
49 </Compare>
50 <Compare metric="numSimilarity" optional="1" weight="0.7">
51 <Param name="num1" path="?a/wgs84_pos:long" />
52 <Param name="num2" path="?b/wgs84_pos:long" />
53 </Compare>
54 </AVG>
55 </LinkCondition>
56 <Thresholds accept="0.9" verify="0.7" />
57 <Limit max="1" method="metric_value" />
58 <Output acceptedLinks="accepted_links.n3" verifyLinks="verify_links.n3"
mode="truncate" />
59 </Interlink>
60 </Silk>
```

Στο παραπάνω παράδειγμα στόχος είναι η εύρεση συνδέσεων *owl:sameAs* (πεδίο <*LinkType*>, Γραμμή 12) ανάμεσα στα URIs που χρησιμοποιούνται από τις βάσεις *DBPedia* και *GeoNames* για την αναγνώριση πόλεων. Για τον προσδιορισμό των πηγών χρησιμοποιείται το πεδίο <*DataSource*> όπου προσδιορίζεται το όνομα της πηγής, το endpoint URI (υποχρεωτικό πεδίο) καθώς και άλλες παράμετροι όπως το πλήθος των αποτελεσμάτων που θα αποθηκευτούν στη μνήμη (πεδίο <*PageSize*>). Ακόμη, προσδιορίζεται ο ρόλος των πηγών με τα πεδία <*SourceDataset*> και <*TargetDataset*>. Εφόσον στόχος είναι η σύνδεση πόλεων η αναζήτηση περιορίζεται σε οντότητες των κλάσεων *dbpedia:City* και *dbpedia:PopulatedPlace* με το πεδίο <*RestrictTo*>.

Το πεδίο <*LinkCondition*> είναι ο πυρήνας του καθορισμού των συνδέσεων και αποτελείται από μετρικές που συνδυάζονται μεταξύ τους ώστε να προκύψει η τελική απόφαση για τη σύνδεση. Ο συνδυασμός γίνεται με συναρτήσεις συνάθροισης που στη συγκεκριμένη περίπτωση είναι ο μέσος όρος των μετρικών (πεδίο <*AVG*>, Γραμμή 20). Οι μετρικές περιλαμβάνουν σύγκριση των ονομάτων των πόλεων (Γραμμή 22) καθώς και σύγκριση των ονομάτων με τα αντίστοιχα της *Wikipedia*. Επίσης, σε επίπεδο αριθμών γίνεται σύγκριση ανάμεσα στις ιδιότητες *dbpedia:populationEstimate*, *dbpedia:populationTotal* και *gn:population* καθώς και ανάμεσα στις γεωγραφικές συντεταγμένες (Γραμμές 46 και 50). Τελικά, κρατείται η μεγαλύτερη τιμή για κάθε μια από τις μετρικές αυτές. Τα ζεύγη που παρουσιάζουν ομοιότητα μεγαλύτερη από το κατώφλι (πεδίο <*Threshold*>, Γραμμή 56) συνδέονται.

4.2.3 Η γλώσσα XML για τον ορισμό μετρικών ποιότητας και πολιτικών συγχώνευσης δεδομένων

Το Sieve χρησιμοποιείται για τη συγχώνευση των δεδομένων σε μια ενιαία αναπαράσταση καθώς και για την εφαρμογή κριτηρίων ποιότητας σε αυτά. Οι λειτουργίες αυτές ορίζονται με τη γλώσσα XML.

Παρακάτω παραθέτουμε τον ορισμό μιας μετρικής ποιότητας με χρήση των διαστάσεων *επικαιρότητα (recency)* και *φήμη (reputation)*:

```
1 <Sieve>
2   <QualityAssessment>
3     <AssessmentMetricid="sieve:recency">
4     <ScoringFunction class="TimeCloseness">
5       <Param name="timeSpan "value="7"/>
6       <Input path="?GRAPH/provenance:lastUpdated"/>
7     </ScoringFunction>
8     </AssessmentMetric>
9     <AssessmentMetricid="sieve:reputation">
10      <ScoringFunctionclass="ScoredList">
11        <Param name="priority"
12          value="http://pt.wikipedia.org http://en.wikipedia.org"/>
13      </ScoringFunction>
14    </AssessmentMetric>
15  </QualityAssessment>
16</Sieve>
```

Στη διάσταση της επικαιρότητας θα χρησιμοποιηθεί η μετρική *TimeCloseness* για να μετρηθεί η απόσταση μεταξύ δυο ημερομηνιών. Χρησιμοποιείται η τρέχουσα ημερομηνία και η ημερομηνία τελευταίας τροποποίησης του γράφου εισόδου. Όσο πιο πρόσφατα ανανεωμένος είναι ο γράφος τόσο μεγαλύτερο θα είναι το αποτέλεσμα της διάστασης *επικαιρότητα*. Παρόμοια μέτρηση γίνεται και με τη διάσταση *φήμη* όπου γίνεται χρήση της μετρικής *ScoredList*. Η τελευταία δέχεται ως όρισμα μια λίστα από υποψήφιους γράφους που θα βαθμολογηθούν ανάλογα με τη θέση τους στη λίστα. Στο παραπάνω παράδειγμα οι τιμές που προέρχονται από το γράφο `http://pt.wikipedia.org` έχουν μεγαλύτερη προτεραιότητα από τον `http://en.wikipedia.org`.

Ένα παράδειγμα ορισμού πολιτικής διαχείρισης αντιφάσεων στα δεδομένα φαίνεται παρακάτω:

```
1<Sieve>
2  <Fusion>
3   <Class name="dbpedia:Settlement">
4     <Property name="rdfs:label">
5       <FusionFunction class="PassItOn"/>
6     </Property>
7     <Property name="dbpedia-owl:areaTotal">
8       <FusionFunction class="KeepSingleValueByQualityScore"
9                       metric="sieve:reputation"/>
10    </Property>
11    <Property name="dbpedia-owl:populationTotal">
```

```
12      <FusionFunction class="KeepSingleValueByQualityScore"
13                      metric="sieve:recency"/>
14    </Property>
15  </Class>
16 </Fusion>
17</Sieve>
```

Εδώ, η συγχώνευση επιδρά σε κάθε στοιχείο της κλάσης *dbpedia:Settlement* και συνενώνει την πληροφορία από τις ιδιότητες *areaTotal* και *populationTotal*. Οι δυο ιδιότητες χρησιμοποιούν τη συνάρτηση συγχώνευσης *KeepSingleValueByQualityScore* που κρατά την τιμή με την υψηλότερη βαθμολογία. Η τελευταία προκύπτει από τη μετρική ποιότητας του προηγούμενου βήματος. Η *areaTotal* παίρνει την τιμή με τη μεγαλύτερη *φήμη* και η *populationTotal* παίρνει την πιο πρόσφατη τιμή.

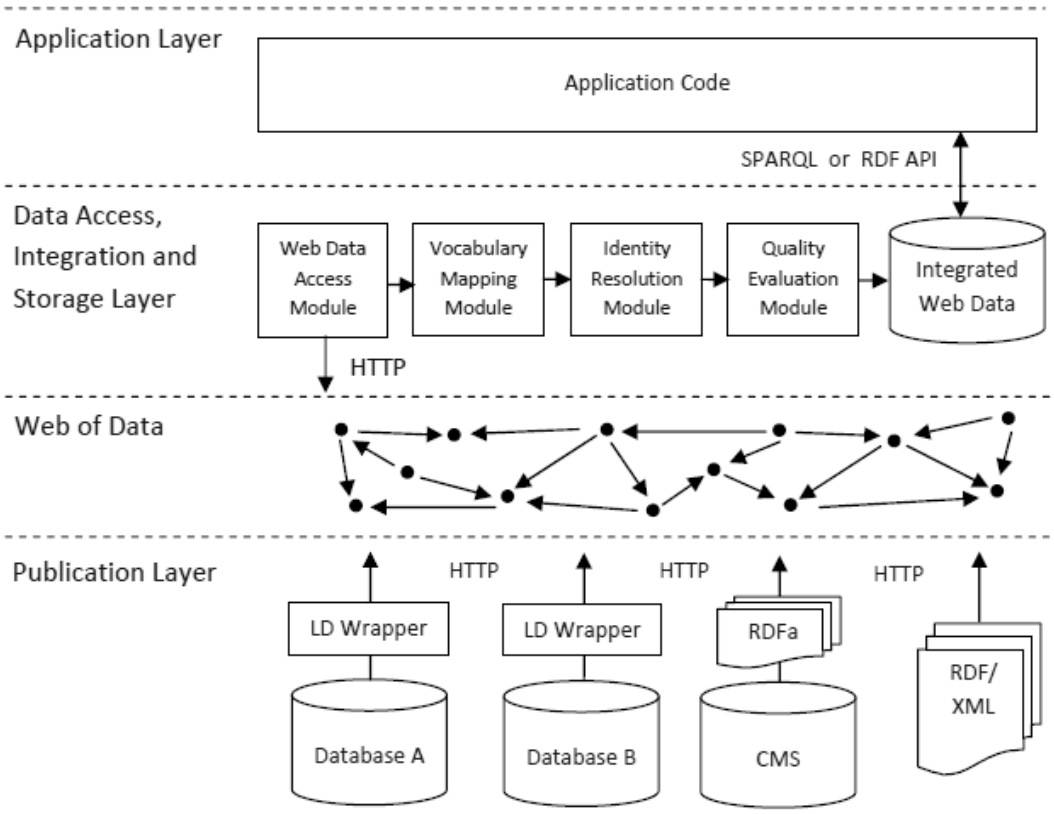
4.3 Η αρχιτεκτονική εφαρμογών Διασυνδεδεμένων Δεδομένων

Στην ενότητα αυτή θα αναλυθούν τα μοντέλα αρχιτεκτονικής βάσει των οποίων δομούνται τα συστήματα ολοκλήρωσης δεδομένων. Παρά το γεγονός ότι κάθε σύστημα, ανάλογα με τις ιδιαιτερότητές του, έχει διαφορετικά σχεδιαστικά χαρακτηριστικά τρεις είναι οι γενικές κατηγορίες αρχιτεκτονικής (Heath & Bizer, 2011, pp. 97-98):

- *Μοντέλο διάσχισης:* Οι εφαρμογές διασχίζουν τον Ιστό Δεδομένων ακολουθώντας RDF συνδέσμους (*crawling*). Στη συνέχεια ολοκληρώνουν και καθαρίζουν τα δεδομένα παρουσιάζοντας μια ενοποιημένη μορφή της πληροφορίας. Η αρχιτεκτονική ομοιάζει με την αντίστοιχη των μηχανών αναζήτησης και είναι κατάλληλη για τη σταδιακή ολοκλήρωση των δεδομένων. Ένα μειονέκτημά της είναι η συχνότητα της διάσχισης με αποτέλεσμα τα δεδομένα ενδεχομένως να μην είναι πολύ πρόσφατα. Αυτό γιατί κατά τη διάρκεια μιας διάσχισης οι πληροφορίες που συλλέγονται αποθηκεύονται και δεικτοδοτούνται τοπικά και για κάποιο χρονικό διάστημα δεν αναζητούνται εκ νέου. Το μοντέλο ολοκλήρωσης Pay-As-You-Go υλοποιείται στην ουσία από αυτή την αρχιτεκτονική.
- *Μοντέλο αναζήτησης σε πραγματικό χρόνο:* Η διαδικασία αναζήτησης γίνεται τη στιγμή που ζητούνται τα δεδομένα με αποτέλεσμα οι πληροφορίες να είναι πάντοτε ανανεωμένες. Ωστόσο, η αναζήτηση μεγάλου όγκου δεδομένων σε πραγματικό χρόνο είναι δαπανηρή υπολογιστικά.
- *Μοντέλο χρήσης ερωτημάτων:* Η αρχιτεκτονική αυτή βασίζεται στην ύπαρξη έτοιμων δομημένων ερωτημάτων τα οποία στέλνονται σε προκαθορισμένες πηγές δεδομένων. Το μοντέλο είναι κατάλληλο σε περιπτώσεις που οι πηγές παρέχουν SPARQL endpoints. Το βασικό του πλεονέκτημα είναι πως τα δεδομένα είναι ανανεωμένα και δεν χρειάζεται να

αποθηκευτούν τοπικά. Ωστόσο, η ένωση των δεδομένων που αποκτώνται από τις διαφορετικές πηγές είναι σύνθετη και χρονοβόρα καθώς αυξάνεται ο αριθμός των πηγών.

Η ολοκλήρωση των Διασυνδεδεμένων Δεδομένων βασίζεται κυρίως στο πρώτο μοντέλο καθώς αποτελεί μια ισορροπημένη λύση ανάμεσα στη χρονική απόδοση και στη «φρεσκάδα» της πληροφορίας. Μια γενική εικόνα για το πως δομείται σε επίπεδο λογισμικού ένα σύστημα ολοκλήρωσης δίνεται παρακάτω:



Εικόνα 4.1: Η αρχιτεκτονική της διάσχισης RDF συνδέσμων (crawling)

Το επίπεδο δημοσίευσης δεδομένων αναπαριστά το πως οι διαφόρων μορφών πληροφορίες μετατρέπονται σε RDF και αποκτούν URIs σχηματίζοντας τον Ιστό Δεδομένων. Το επόμενο επίπεδο δείχνει τα βήματα που γίνονται από ένα σύστημα ολοκλήρωσης και, τέλος, ακολουθούν οι εφαρμογές που επεξεργάζονται περαιτέρω τα ενοποιημένα πλέον δεδομένα. Τόσο το LDIF όσο και το σύστημα της παρούσας διπλωματικής υλοποιούν τις διαδικασίες που λαμβάνουν χώρα στο τρίτο κατά σειρά επίπεδο.

5

Ανάλυση Απαιτήσεων Συστήματος

5.1 Περιγραφή Λειτουργιών

Ο πυρήνας της εφαρμογής επιμερίζεται σε πέντε βασικά υποσυστήματα: σύστημα εισαγωγής πηγών δεδομένων (importing module), σύστημα μετασχηματισμού λεξιλογίων (schema mappings module), σύστημα συνδέσεων δεδομένων (linking module), σύστημα εκτέλεσης ερωτημάτων SPARQL (queries module) και σύστημα διαχείρισης χώρων δεδομένων (dataspace module). Στον πυρήνα προστίθενται το σύστημα επικοινωνίας με τον client και της παρουσίασης των δεδομένων σε αυτόν (presentation module) καθώς και το σύστημα διαχείρισης της υπηρεσίας (service module).

Το πρώτο υποσύστημα έχει ως στόχο την εισαγωγή πηγών δεδομένων, όπως σχεσιακές βάσεις, SPARQL endpoints καθώς και αρχείων. Επίσης, περιλαμβάνει λειτουργίες μετατροπής δεδομένων σε RDF μορφή και το σχηματισμό των διακριτών συνόλων δεδομένων (datasets). Η διαχείριση των χώρων δεδομένων γίνεται από το dataspace module. Η μονάδα αυτή είναι υπεύθυνη για το σταδιακό σχηματισμό του dataspace, την αποθήκευση των μετα-δεδομένων που συνοδεύουν τα σύνολα δεδομένων (provenance) καθώς και την αποθήκευση των χώρων δεδομένων στη μνήμη ώστε να είναι δυνατή η επαναχρησιμοποίησή τους. Μαζί με το σύστημα εισαγωγής πηγών δεδομένων, αποτελεί τον κορμό της εφαρμογής.

Το σύστημα μετασχηματισμού λεξιλογίων έχει ως πυρήνα το R2R Framework και μεταφράζει τα υπάρχοντα λεξιλόγια των datasets σε όρους που ορίζει ο χρήστης. Δέχεται ως είσοδο ένα dataset και ένα σύνολο από αντιστοιχίες όρων (mappings). Η έξοδός του είναι το ίδιο dataset με τροποποιημένο σχήμα ανάλογα με τα mappings που έχουν εισαχθεί. Το σύστημα συνδέσεων δέχεται ως είσοδο δυο datasets και κάποιες παραμέτρους με τις οποίες θα γίνουν οι συνδέσεις και παράγει τριπλέτες που περιγράφουν τους συνδέσμους. Οι λειτουργίες του συστήματος βασίζονται στο SILK Framework και οι σύνδεσμοι προστίθενται στον υπάρχοντα χώρο δεδομένων (dataspace). Τα SPARQL ερωτήματα από τον χρήστη εφαρμόζονται στο χώρο δεδομένων μέσω του συστήματος ερωτημάτων (query module).

Το σύστημα διαχείρισης της υπηρεσίας περιλαμβάνει βασικές λειτουργίες της εφαρμογής, όπως επικοινωνία με το τοπικό σύστημα αρχείων και σύνδεση του συστήματος επικοινωνίας με τον πυρήνα της εφαρμογής. Στις παρακάτω ενότητες περιγράφονται πιο αναλυτικά οι λειτουργίες και ο σκοπός χρήσης κάθε υποσυστήματος.

5.1.1 Υποσύστημα διαχείρισης συνόλων δεδομένων και μετα-δεδομένων

Λειτουργίες

1. **Είσοδος / Έξοδος συνόλων δεδομένων:** το importing module εισάγει datasets ενώ τα schema mapping module και linking module ζητούν datasets.
2. **Διαχείριση μετα-δεδομένων των datasets:** Τα μετα-δεδομένα περιλαμβάνουν κυρίως πληροφορίες που σχετίζονται με τις διάφορες ενέργειες που γίνονται στα σύνολα δεδομένων κατά τη διάρκεια λειτουργίας της εφαρμογής, όπως μετάφραση σχημάτων-λεξιλογίων και σύνδεση οντοτήτων.
3. **Διαχείριση του σχηματιζόμενου dataspace:** Ο χώρος δεδομένων που σταδιακά σχηματίζεται από την εισαγωγή των datasets ελέγχεται από το σύστημα αυτό. Ο χώρος δεδομένων ανανεώνεται κάθε φορά που επικυρώνεται μια διαδικασία μετάφρασης σχήματος (schemaMapping task) ή μια διαδικασία σύνδεσης οντοτήτων (linking task). Στην πρώτη περίπτωση εισάγονται τα RDF δεδομένα που έχουν προκύψει από το μετασχηματισμό του σχήματος ενός dataset (βλέπε περιγραφή schema mapping module) και στη δεύτερη περίπτωση οι σύνδεσμοι ανάμεσα στα δεδομένα.
4. **Αποθήκευση του σχήματος-λεξιλογίου του dataspace:** Το σχήμα διατηρείται στην τοπική μνήμη και ανανεώνεται κάθε φορά που επικυρώνεται μια διαδικασία μετάφρασης σχήματος. Με τον τρόπο αυτό σχηματίζεται ένα γενικό σχήμα-λεξιλόγιο που περιγράφει το χώρο δεδομένων.
5. **Αποθήκευση των χώρων δεδομένων στη μνήμη:** Ο χρήστης έχει τη δυνατότητα να αποθηκεύει το χώρο δεδομένων που έχει δημιουργήσει ώστε να μπορεί να εφαρμόσει

ερωτήματα σε μελλοντικό χρόνο. Ακόμη, μαζί με το χώρο δεδομένων αποθηκεύονται και τα tasks που έχουν γίνει.

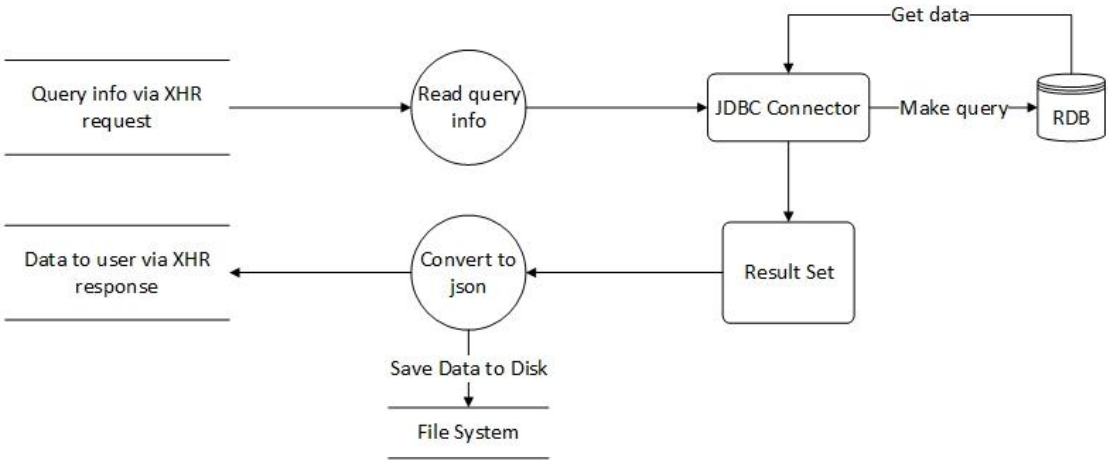
5.1.2 Υποσύστημα εισαγωγής πηγών

Λειτουργίες

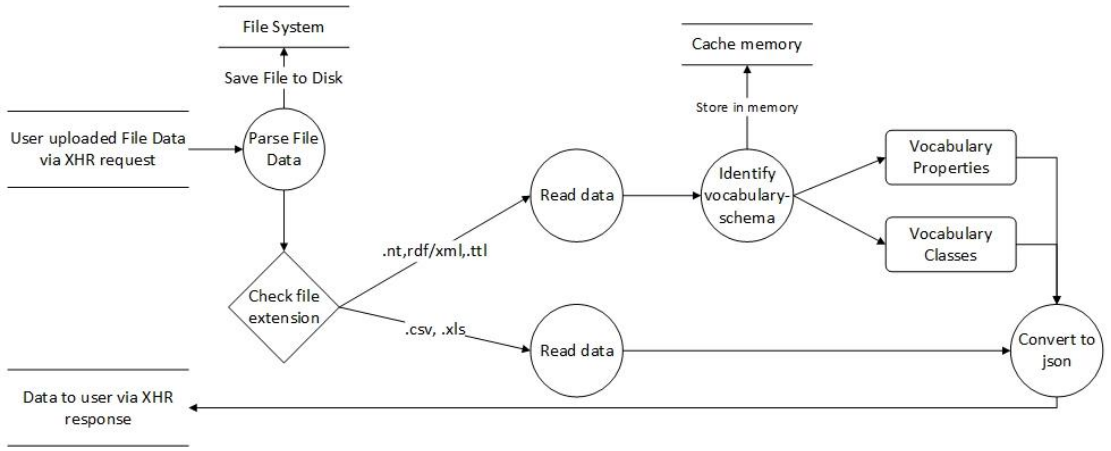
1. **Είσαγωγή πηγών δεδομένων:** Οι πηγές είναι σχεσιακές βάσεις δεδομένων, sparql endpoints και τοπικά αρχεία στις μορφές .csv,.xls, .ttl, .xml/rdf και .nt .
2. **Εξαγωγή μετα-πληροφορίας από σχεσιακές βάσεις:** Η μετα-πληροφορία αφορά το σχήμα που εισάγεται και περιλαμβάνει τα ονόματα των σχέσεων, τα ονόματα και τους τύπους δεδομένων κάθε πεδίου και τις σχέσεις ανάμεσα στους πίνακες (ξένα κλειδιά).
3. **Εξαγωγή λεξιλογίου-σχήματος από RDF δεδομένα:** Το λεξιλόγιο-σχήμα διακρίνεται σε κλάσεις και ιδιότητες.
4. **Ανάκτηση δεδομένων από τις πηγές:** Αφού ο χρήστης εισάγει μια πηγή στο σύστημα μπορεί να λάβει δεδομένα από αυτή. Στην περίπτωση της σχεσιακής βάσης και των sparql endpoints η λήψη δεδομένων γίνεται με την εφαρμογή ερωτημάτων (σε γλώσσα sql και sparql αντίστοιχα). Στην περίπτωση ενός αρχείου-πηγής τα δεδομένα λαμβάνονται μέσω της φόρτωσης (upload) του αρχείου στο σύστημα. Κατά τη διάρκεια της αλληλεπίδρασης του χρήστη με τις πηγές τα δεδομένα αποθηκεύονται στο τοπικό σύστημα αρχείων. Επίσης, μετατρέπονται σε μορφή JSON[22] ώστε να είναι δυνατή η παρουσίασή τους στο γραφικό περιβάλλον της εφαρμογής.
5. **Μετατροπή τοπικών αρχείων σε RDF datasets:** Τα δεδομένα που δεν βρίσκονται σε RDF μορφή μετατρέπονται σε τριπλέτες με τη διαδικασία που περιγράφεται στο Κεφάλαιο 7. Έτσι, προκύπτει ένα dataset το οποίο χαρακτηρίζεται από μια οντολογία. Δεδομένα που είναι ήδη σε RDF μορφή δεν επιδέχονται κάποιο μετασχηματισμό και σχηματίζουν αυτόματα ένα dataset.

Παρακάτω παρουσιάζονται μερικά σχεδιαγράμματα ροής δεδομένων που δείχνουν την πορεία των δεδομένων κατά τη διάρκεια των διαδικασιών που περιγράφηκαν:

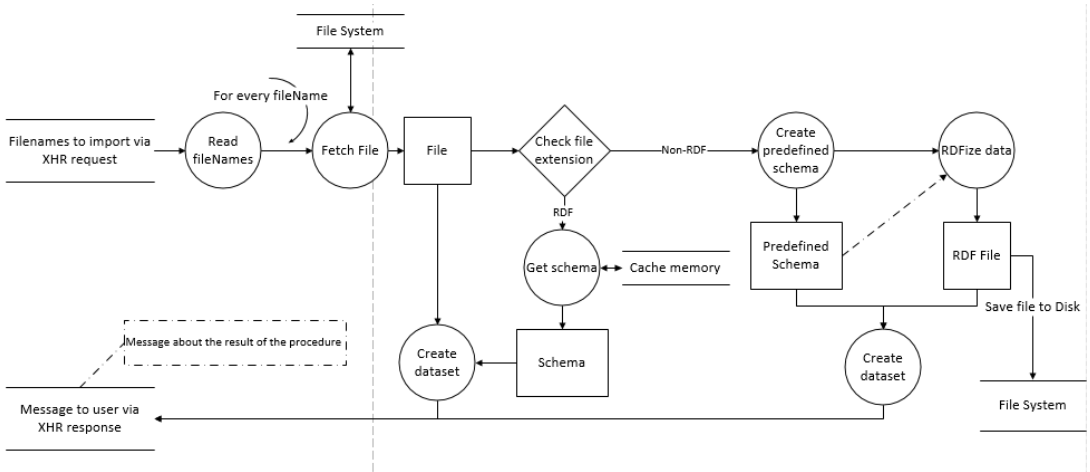
[22] *JSON:Javascript Object Notation, τρόπος αναπαράστασης πληροφορίας που χρησιμοποιείται ευρέως σε web εφαρμογές για τη μεταφορά δεδομένων*



Εικόνα 5.1: Εφαρμογή ερωτήματος σε βάση δεδομένων



Εικόνα 5.2: Εισαγωγή αρχείου στην εφαρμογή



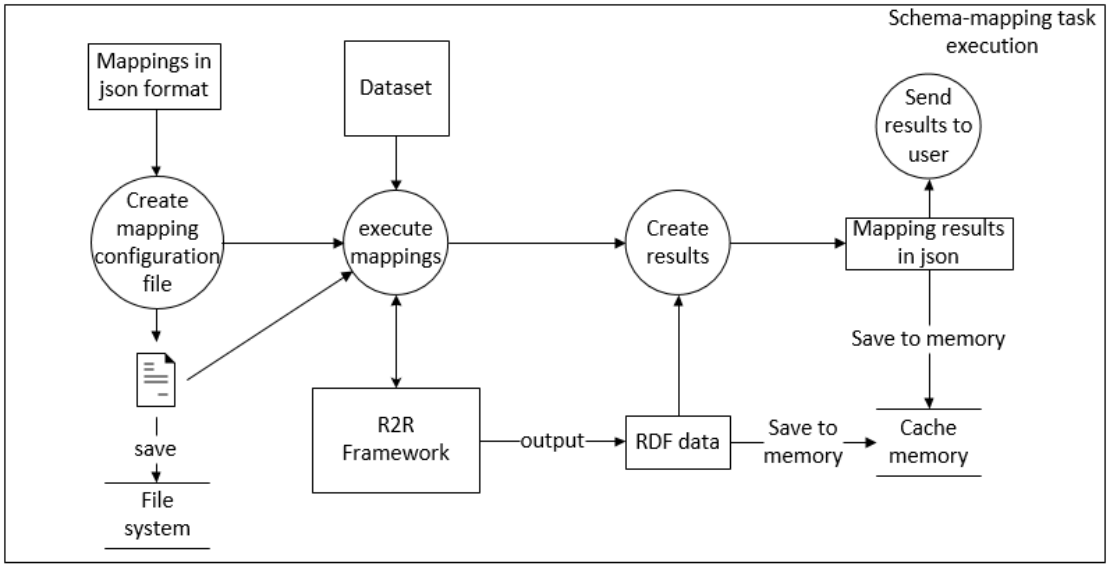
Εικόνα 5.3: Δημιουργία dataset

5.1.3 Υποσύστημα μετασχηματισμού λεξιλογίων

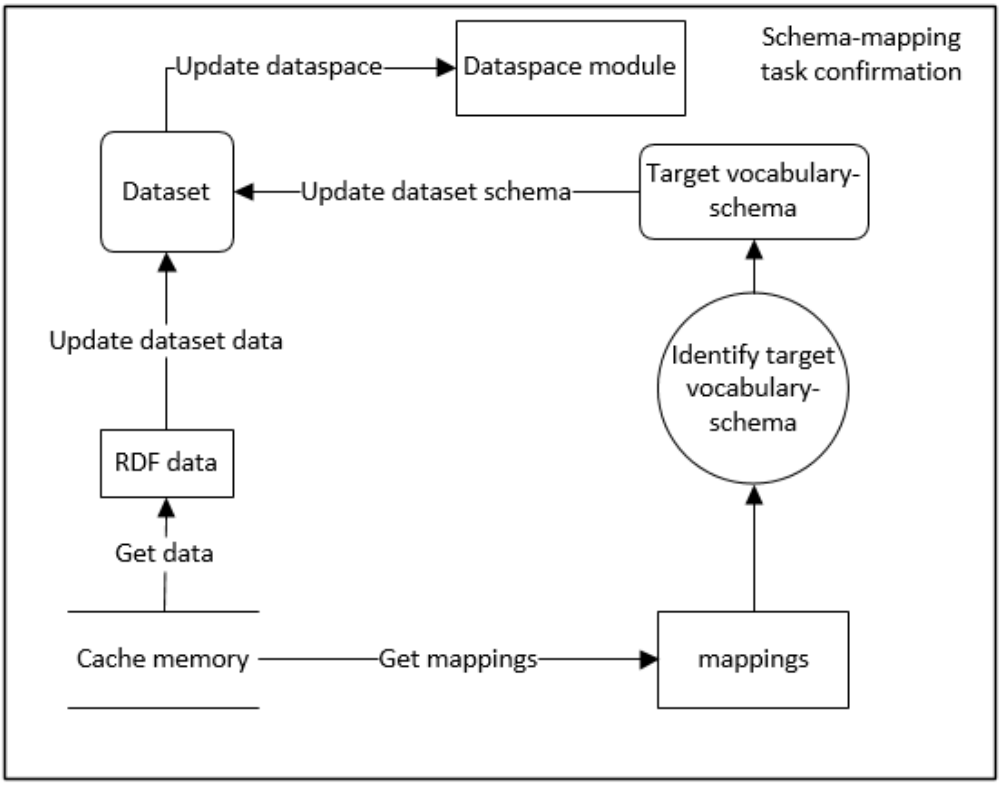
Λειτουργίες
1. **Διαχείριση διαδικασιών μετάφρασης λεξιλογίου-σχήματος (schemaMapping task):** Η διαδικασία μετάφρασης αποτελεί βασικό στοιχείο του συστήματος. Κατά τη διάρκεια αυτής ο χρήστης επιλέγει τα mappings μεταξύ των όρων και τα εφαρμόζει στο αντίστοιχο dataset. Κάθε τέτοια διαδικασία συνδέεται αποκλειστικά με ένα dataset. Κατά συνέπεια, δημιουργούνται τόσα schemaMapping tasks όσα και τα datasets που έχουν σχηματιστεί. 2. **Σχηματισμός αρχείου αντιστοίχισης όρων λεξιλογίων (mappings):** Κάθε διαδικασία μετάφρασης δέχεται ως είσοδο τις αντιστοιχίσεις ανάμεσα σε όρους που έχει εισάγει ο χρήστης και παράγει το αντίστοιχο αρχείο. Οι αντιστοιχήσεις βρίσκονται σε μορφή JSON και προέρχονται από τον client. Το αρχείο που προκύπτει είναι σε .ttl μορφή και περιλαμβάνει τα δεδομένα των αντιστοιχήσεων οργανωμένα βάσει της οντολογίας του R2R Framework. 3. **Επικοινωνία με το R2R Framework:** Όπως έχει προαναφερθεί, το R2R Framework αποτελεί τη βάση του υποσυστήματος αυτού. Το αρχείο των αντιστοιχήσεων που δημιουργείται δίνεται ως είσοδος στο R2R Framework και από το τελευταίο παράγονται τα κατάλληλα αποτελέσματα που αποθηκεύονται στη μνήμη. Ο σχηματισμός των mappings είναι μια επαναλαμβανόμενη διαδικασία η οποία σταματά μόλις ο χρήστης επιβεβαιώσει το αντίστοιχο task. Τότε τα δεδομένα που έχουν παραχθεί από το R2R Framework αποθηκεύονται στο τοπικό σύστημα αρχείων. Επίσης, τα δεδομένα αυτά εισάγονται μέσω του dataspace module στο dataspace της εφαρμογής. 4. **Εξαγωγή λεξιλογίου-στόχου από τις αντιστοιχήσεις όρων:** Από τα mappings που αντιστοιχούν σε κάθε dataset προκύπτει το λεξιλόγιο-στόχος (βλέπε Ενότητα 4.1) που στην ουσία αποτελεί και το τελικό λεξιλόγιο του dataset μιας και δεν υπάρχει κάποια επιπλέον δυνατότητα αλλαγής του. Το λεξιλόγιο αυτό συμπεριλαμβάνεται στο συνολικό λεξιλόγιο-σχήμα του dataspace της εφαρμογής μόλις ο χρήστης επιβεβαιώσει ένα mapping-task. 5. **Παρουσίαση αποτελεσμάτων στο χρήστη:** Το R2R Framework δεν παρέχει πληροφορίες σχετικά με τα αποτελέσματα των αντιστοιχήσεων. Ωστόσο, προκειμένου να κριθεί η ορθότητα των mappings χρειάζεται να παρουσιαστεί στο χρήστη η κατάλληλη πληροφορία. Έτσι, αυτό που εξάγεται από το σύστημα είναι ο αριθμός των παραγόμενων τριπλετών που περιλαμβάνουν τους όρους κάθε αντιστοίχησης.

6. **Εισαγωγή λεξιλογίων:** Ο χρήστης μπορεί να εισάγει από τοπικό αρχείο κάποιο υπάρχον λεξιλόγιο ώστε να το χρησιμοποιήσει στη μετάφραση του σχήματος ενός dataset.

Τα παρακάτω διαγράμματα αναπαριστούν σχηματικά τις κυριότερες προαναφερθείσες λειτουργίες:



Εικόνα 5.4: Διαδικασία εκτέλεσης ενός mapping task

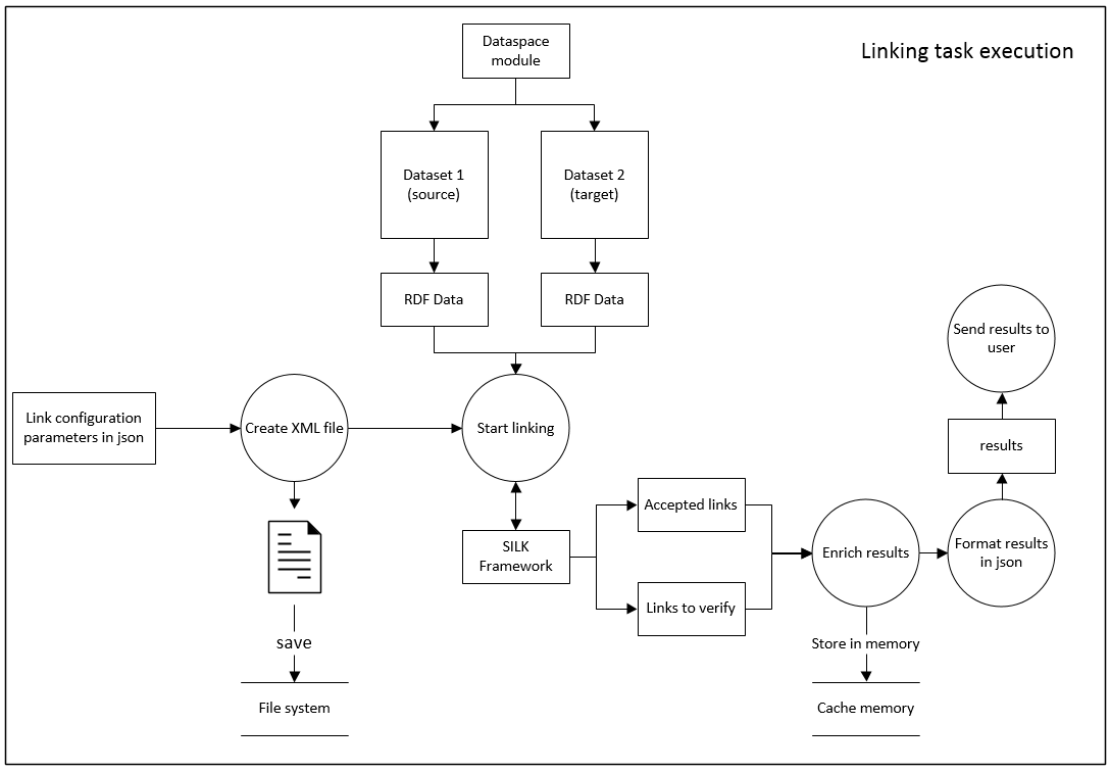


Εικόνα 5.5: Διαδικασία επιβεβαίωσης ενός mapping task

5.1.4 Υποσύστημα συνδέσεων οντοτήτων

Λειτουργίες
1. **Διαχείριση διαδικασιών σύνδεσης (linking task):** Η διαδικασία αυτή αποτελεί βασικό στοιχείο του συστήματος. Κατά τη διάρκεια αυτής ο χρήστης επιλέγει τις παραμέτρους βάσει των οποίων θα συνδεθούν παρόμοιες οντότητες διαφορετικών datasets. Κάθε διαδικασία προσπαθεί να συνδέσει οντότητες από δυο datasets. Ο αριθμός των διαδικασιών εξαρτάται από τον χρήστη και δεν είναι προκαθορισμένος. 2. **Σχηματισμός αρχείου XML στη γλώσσα LSL:** Κάθε linking task έχει ως δομικό στοιχείο ένα αρχείο XML στο οποίο περιγράφονται οι παράμετροι της σύνδεσης. Οι παράμετροι ορίζονται από το χρήστη μέσω του γραφικού περιβάλλοντος της εφαρμογής. 3. **Επικοινωνία με το SILK Framework:** Το SILK Framework αποτελεί τη βάση του υποσυστήματος αυτού. Το αρχείο XML κάθε διαδικασίας σύνδεσης, μαζί με τα αντίστοιχα datasets, δίνονται ως είσοδος. Το SILK Framework αξιοποιεί τα δεδομένα των datasets προκειμένου να ανακαλύψει παρόμοιες οντότητες και παράγει ως έξοδο τις συνδέσεις ανάμεσα στις οντότητες αυτές. Οι παραγόμενοι σύνδεσμοι μπορούν να χωριστούν σε κατηγορίες (και να γραφούν σε διαφορετικά αρχεία) ανάλογα με το βαθμό ομοιότητας (confidence) που παρουσιάζουν. Στο linking module ορίζονται δυο τέτοιες κατηγορίες: οι σύνδεσμοι που θεωρούνται ως «αποδεκτοί» και οι σύνδεσμοι που χρειάζονται επιβεβαίωση προτού θεωρηθούν έγκυροι. Τα παράθυρα ομοιότητας (confidence windows) για κάθε κατηγορία ορίζονται γραφικά από τον χρήστη. 4. **Παρουσίαση αποτελεσμάτων στο χρήστη:** Το SILK Framework υποστηρίζει δυο μορφές εξόδου: Η πρώτη είναι η απλή καταγραφή σε αρχείο των τριπλετών που περιγράφουν τις συνδέσεις (κάθε τριπλέτα είναι και μια σύνδεση) και η δεύτερη είναι η καταγραφή των τριπλετών σε μορφή XML όπου προστίθεται ο βαθμός ομοιότητας (confidence) ανάμεσα στις συνδεόμενες οντότητες. Έχοντας ως βάση αυτές τις εξόδους το linking module παρουσιάζει πιο αναλυτικές πληροφορίες που σχετίζονται με τον κάθε σύνδεσμο, όπως την τιμή που έχουν οι ιδιότητες των δυο οντοτήτων που συνδέονται. 5. **Ανανέωση του dataspace:** Κάθε φορά που επικυρώνεται μια διαδικασία σύνδεσης, οι σύνδεσμοι που έχουν προκύψει από αυτή προστίθενται μέσω του dataspace module στο σχηματιζόμενο dataspace.

Στο σχήμα που ακολουθεί παρουσιάζονται τα βήματα εκτέλεσης ενός linking task:



Εικόνα 5.6: Διαδικασία εκτέλεσης ενός linking task

5.1.5 Υποσύστημα εφαρμογής SPARQL ερωτημάτων

Λειτουργίες
1. **Εκτέλεση SPARQL ερωτημάτων:** τα ερωτήματα, γραμμένα σε γλώσσα SPARQL, προέρχονται από το γραφικό περιβάλλον της εφαρμογής και εφαρμόζονται στο dataspace που έχει δημιουργηθεί. 2. **Παρουσίαση αποτελεσμάτων στο χρήστη:** Τα αποτελέσματα των ερωτημάτων μετατρέπονται στη μορφή JSON προκειμένου να μεταφερθούν και να παρουσιαστούν στο πρόγραμμα-πελάτη (γραφικό περιβάλλον εφαρμογής).

5.1.6 Υποσύστημα επικοινωνίας με τον client

Λειτουργίες
1. **Έλεγχος και επαλήθευση των παραμέτρων των http αιτημάτων του προγράμματος πελάτη:** Κάθε φορά που το πρόγραμμα-πελάτης της εφαρμογής θέλει να επικοινωνήσει με τον server χρησιμοποιεί ένα URI και το πρωτόκολλο HTTP. Οι κλήσεις αυτές (κλήσεις AJAX) περιέχουν παραμέτρους και δεδομένα που

ελέγχονται από το σύστημα επικοινωνίας. Σε περίπτωση που είναι ορθές οι πληροφορίες εκτελούνται οι αντίστοιχες ενέργειες της υπηρεσίας. Σε αντίθετη περίπτωση η κλήση απορρίπτεται και επιστρέφεται ενημερωτικό μήνυμα.

2. **Μεταφορά των μηνυμάτων του συστήματος-πυρήνα στο γραφικό περιβάλλον:** Σε περιπτώσεις αποτυχίας κάποιας διαδικασίας δημιουργούνται ενημερωτικά μηνύματα τα οποία μεταφέρονται στο χρήστη μέσω του συστήματος αυτού.

5.1.7 Υποσύστημα διαχείρισης της υπηρεσίας

Λειτουργίες

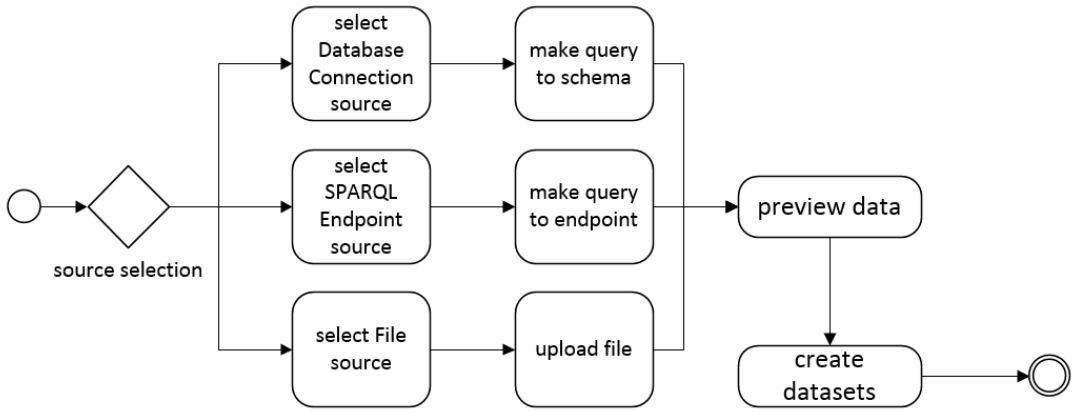
1. **Εποπτεία των επιμέρους υποσυστημάτων:** Το υποσύστημα αυτό διαδραματίζει κεντρικό ρόλο στην εφαρμογή καθώς ελέγχει την επικοινωνία ανάμεσα στα υπόλοιπα υποσυστήματα. Ακόμη, είναι εκείνο το υποσύστημα όπου θα επηρεαστεί κυρίως κατά την προσθήκη και άλλων υποσυστημάτων που θα υποστηρίζουν επιπλέον λειτουργίες της διαδικασίας ολοκλήρωσης (π.χ. υποσύστημα ελέγχου ποιότητας δεδομένων).
2. **Διαχείριση των τοπικών αρχείων της εφαρμογής:** Η εφαρμογή χρησιμοποιεί αρχεία και φακέλους προκειμένου να αποθηκευτούν τα δεδομένα των datasets, τα δεδομένα που εισάγονται από τις διάφορες πηγές καθώς και οι παράμετροι των διαδικασιών μετάφρασης λεξιλογίων και συνδέσεων. Η διαχείριση του τοπικού συστήματος αρχείων γίνεται κεντρικά σε αυτό το υποσύστημα.
3. **Σύνδεση του συστήματος επικοινωνίας με τον πυρήνα της εφαρμογής:** Το σύστημα διαχείρισης συνδέει το σύστημα επικοινωνίας με όλα τα υπόλοιπα συστήματα με διαφανή τρόπο.
4. **Διαχείριση των URIs των λεξιλογίων:** Κατά τη διάρκεια εκτέλεσης της εφαρμογής εισάγονται στο σύστημα διάφοροι όροι λεξιλογίων. Όπως είναι γνωστό, οι όροι αναπαρίστανται με URIs που αποτελούνται από το χώρο ονομάτων του λεξιλογίου (vocabulary namespace) και το όνομα του όρου. Ακόμη, κάθε χώρος ονομάτων συνοδεύεται από ένα πρόθεμα (prefix). Τα ζεύγη namespace-prefix αποθηκεύονται κεντρικά από το service module ώστε να προσφέρονται σε όποιο υποσύστημα τα χρειάζεται. Ακόμη, περιλαμβάνονται λειτουργίες αναζήτησης και καταχώρησης τέτοιων ζευγών.

5.2 *Περιγραφή σεναρίων χρήσης*

Στην ενότητα αυτή θα παρουσιαστούν τα βασικά σενάρια χρήσης της εφαρμογής. Στα πλαίσια των σεναρίων αυτών πραγματοποιούνται και οι λειτουργίες των υποσυστημάτων που αναλύθηκαν στην προηγούμενη ενότητα.

5.2.1 Εισαγωγή πηγών και σχηματισμός datasets

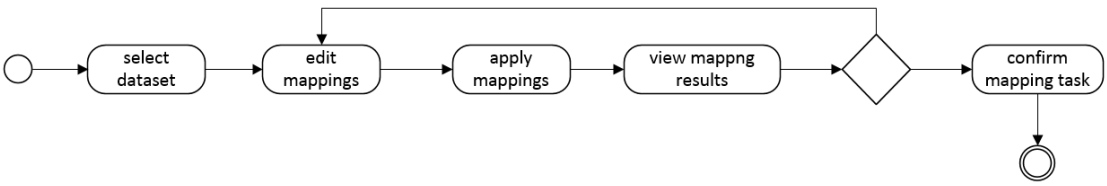
Στο σενάριο αυτό ο χρήστης επιλέγει τις πηγές που θέλει να εισάγει. Μόλις εισαχθεί κάποια πηγή μπορεί να λάβει δεδομένα από αυτή. Μετά από την επισκόπηση των δεδομένων μπορεί να δημιουργήσει τα datasets και να ολοκληρώσει τη διαδικασία.



Εικόνα 5.7: Activity diagram για την εισαγωγή πηγών και τη δημιουργία συνόλων δεδομένων

5.2.2 Μετάφραση του λεξιλογίου-σχήματος ενός dataset (schema mapping task)

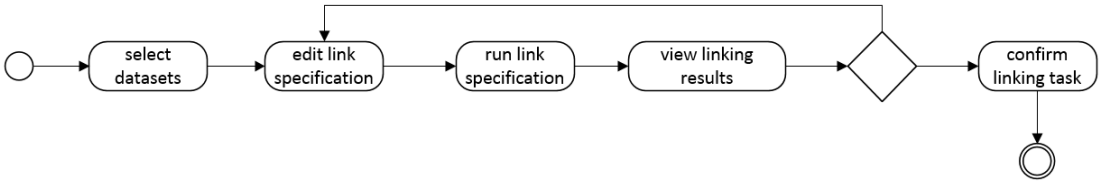
Στόχος του σεναρίου αυτού είναι ο ορισμός, η εκτέλεση και η επικύρωση ενός mapping task βάσει του οποίου θα μετασχηματιστεί το λεξιλόγιο ενός dataset. Ο χρήστης αρχικά επιλέγει ένα dataset και στη συνέχεια ορίζει τις αντιστοιχίες ανάμεσα στους όρους του σχήματος-λεξιλογίου. Στο σημείο αυτό μπορεί να εφαρμόσει τα mappings στο dataset και να δει τα αποτελέσματα. Η διαδικασία εφαρμογής και επισκόπησης των mappings μπορεί να είναι επαναλαμβανόμενη και σταματά όταν ο χρήστης επικυρώσει το task.



Εικόνα 5.8: Activity diagram για τη δημιουργία, εκτέλεση και επικύρωση ενός mapping task

5.2.3 Σύνδεση οντοτήτων ανάμεσα σε δυο datasets (linking task)

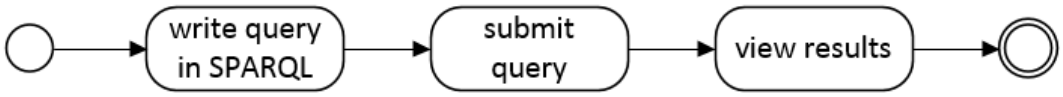
Η διαδικασία δημιουργίας και εκτέλεσης ενός linking task είναι αντίστοιχη με εκείνη ενός mapping task. Παρόμοια με πριν, ορίζεται και παραμετροποιείται το linking task μέσω του γραφικού περιβάλλοντος και στη συνέχεια εκτελείται. Ο χρήστης βλέπει τα αποτελέσματα που είναι οι σύνδεσμοι ανάμεσα στις οντότητες των δυο datasets και είτε επαναλαμβάνει τη διαδικασία παραμετροποίησης και εκτέλεσης είτε επικυρώνει το task.



Εικόνα 5.9: Activity diagram για τη δημιουργία, εκτέλεση και επικύρωση ενός linking task

5.2.4 Εφαρμογή ερωτημάτων SPARQL στο dataspace

Το σενάριο αυτό είναι πολύ απλό και περιλαμβάνει τη συγγραφή ενός SPARQL ερωτήματος στο γραφικό περιβάλλον της εφαρμογής, την καταχώρησή του για εκτέλεση στο dataspace και την επισκόπηση των αποτελεσμάτων.



Εικόνα 5.9: Activity diagram για τη σύνταξη ενός sparql ερωτήματος και την επισκόπηση των αποτελεσμάτων

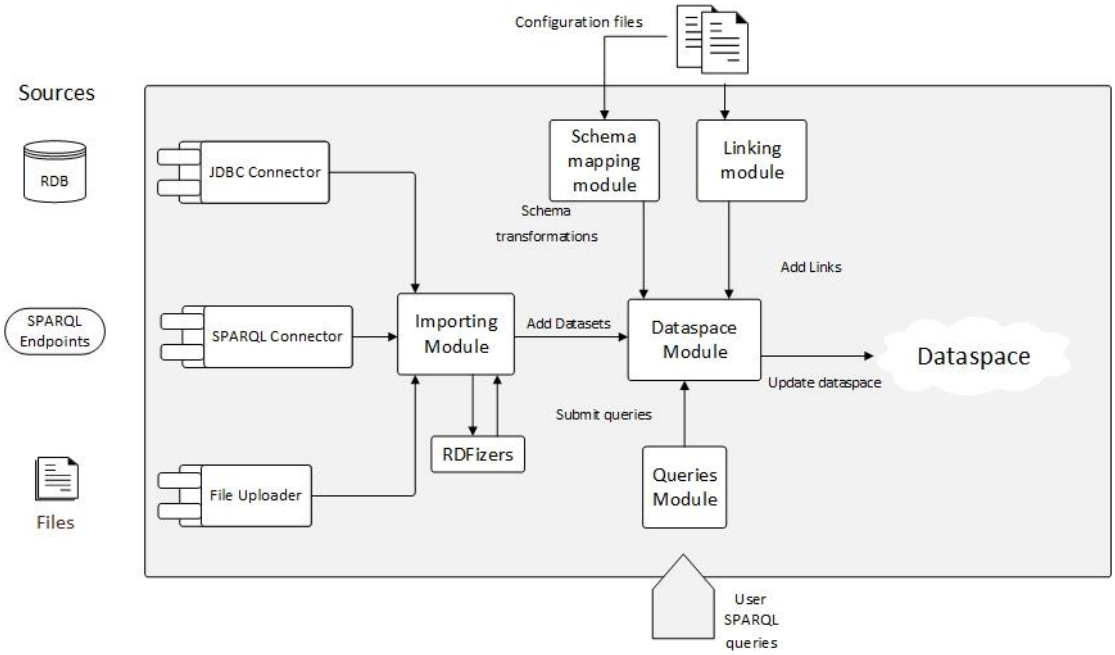
6

Σχεδίαση Συστήματος

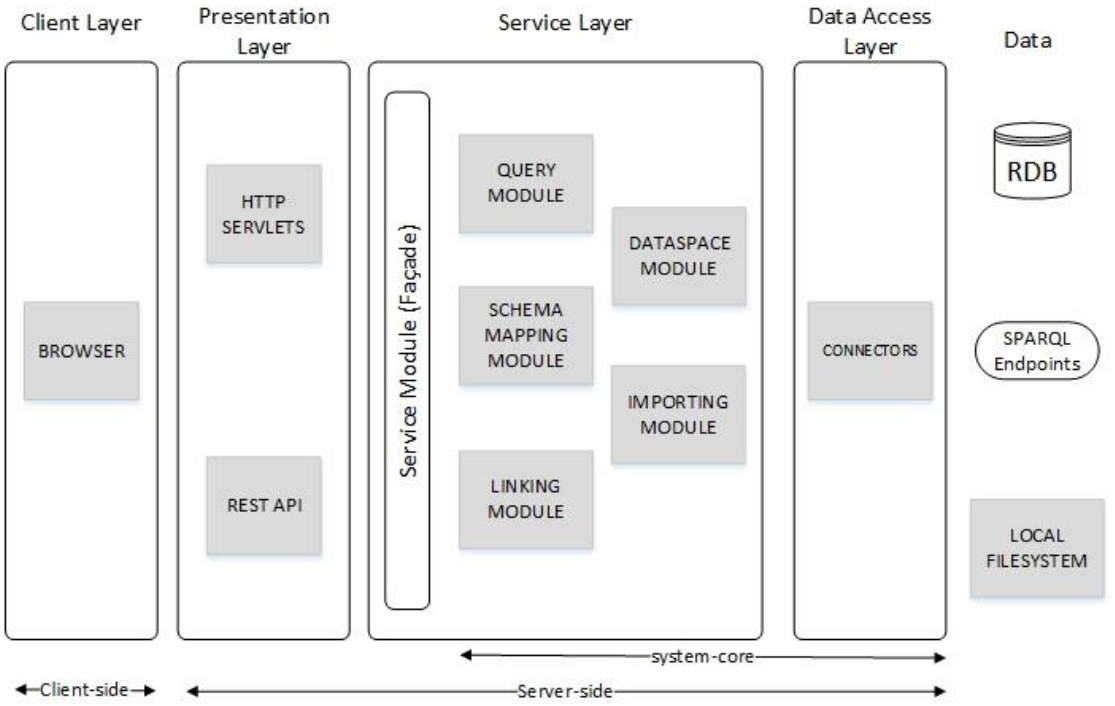
Το σύστημα ολοκλήρωσης δεδομένων αποτελεί μια web εφαρμογή και βασίζεται στις ιδέες της αρχιτεκτονικής REST. Η τεχνολογία REST προσδίδει μεγαλύτερη ευελιξία στην επικοινωνία του client και του server καθώς υπάρχει χαμηλή σύζευξη (coupling) ανάμεσά τους. Προσφέρει, επίσης, δυνατότητες επέκτασης των λειτουργιών της εφαρμογής χωρίς να επηρεάζονται οι ήδη υπάρχουσες λειτουργίες (scalability). Η χρήση της συμβάλει στην ανάπτυξη ενός ενιαίου API, η σημασιολογία του οποίου είναι ανεξάρτητη από την υλοποίηση της εφαρμογής. Τέλος, η REST αρχιτεκτονική είναι συναφής με την ιδέα του Ιστού Δεδομένων καθώς βασίζεται στην απόδοση ενός URI σε έναν πόρο.

6.1 Αρχιτεκτονική εφαρμογής (Server - Side)

Στην Εικόνα 6.1 φαίνεται η αρχιτεκτονική του συστήματος-πυρήνα καθώς και οι αλληλεπιδράσεις των επιμέρους υποσυστημάτων. Στην Εικόνα 6.2 παρουσιάζονται τα επίπεδα στα οποία οργανώνεται η εφαρμογή.

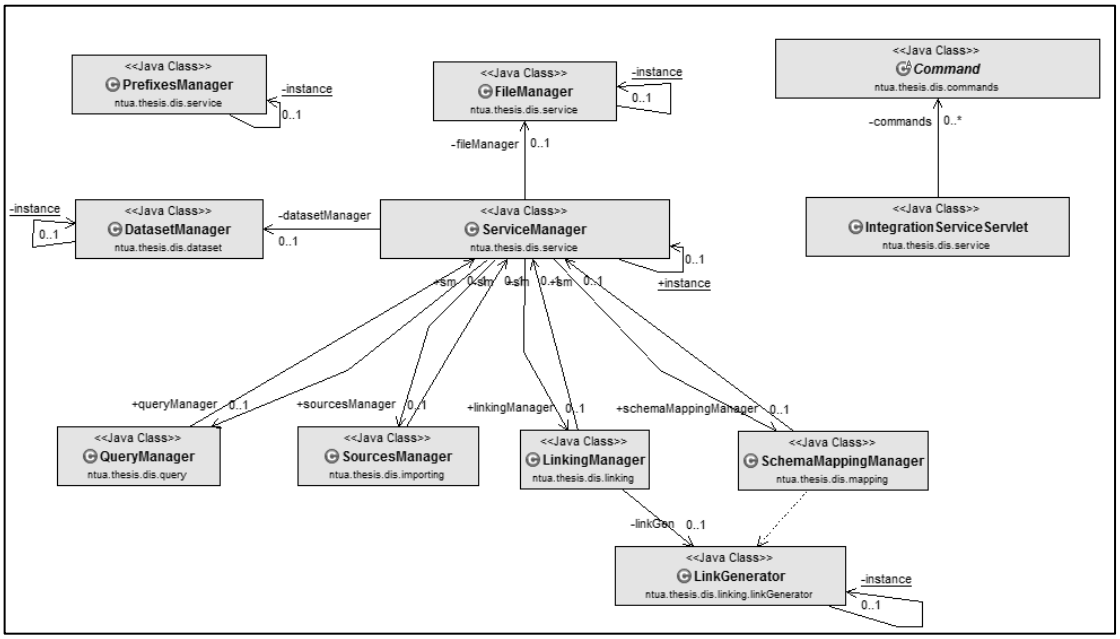


Εικόνα 6.1: Αλληλεπίδραση των υπο-συστημάτων της εφαρμογής

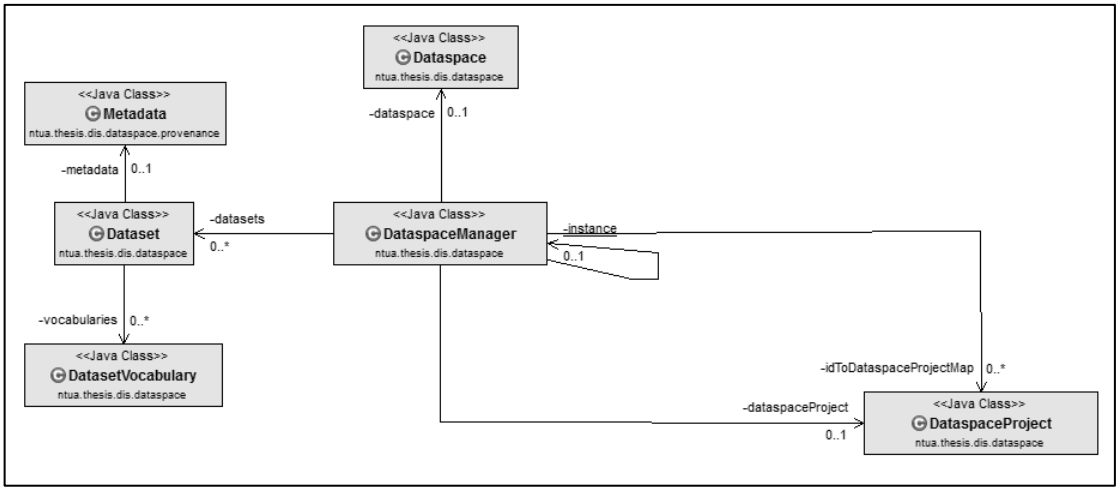


Εικόνα 6.2: Multi-tier αρχιτεκτονική συστήματος

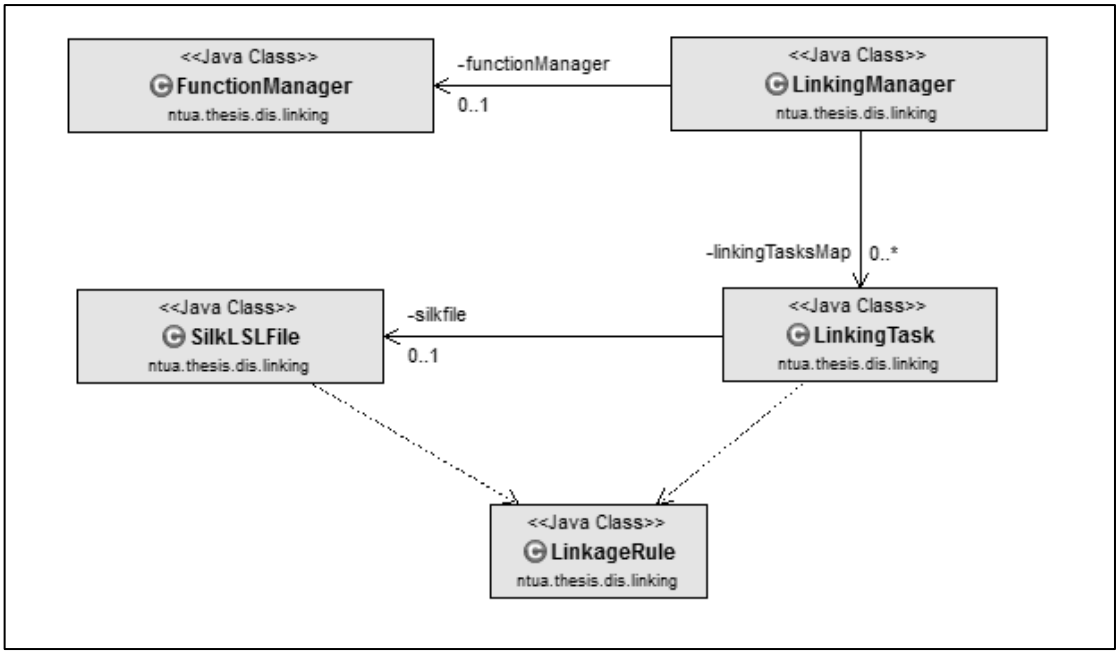
6.1.1 Διαγράμματα κλάσεων



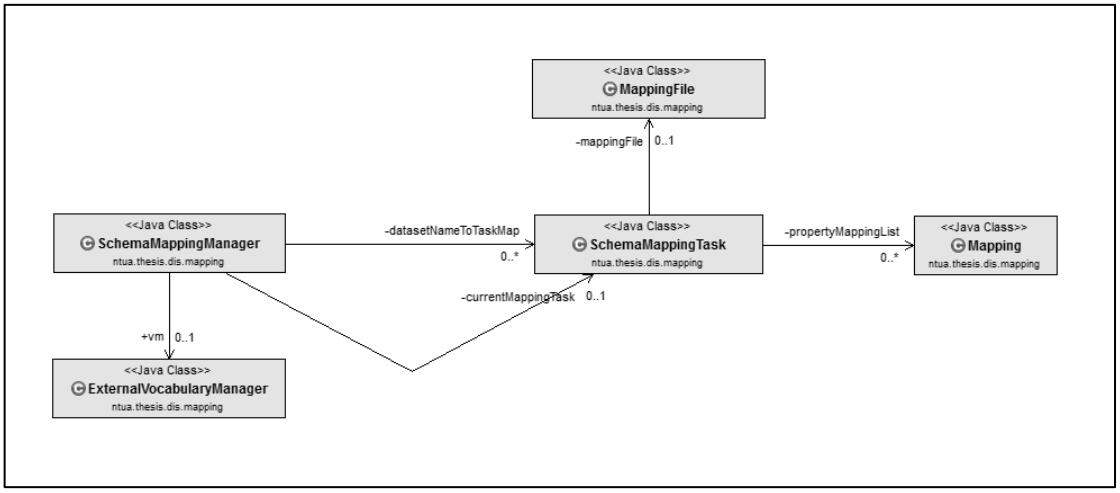
Εικόνα 6.3: Κεντρικό διάγραμμα κλάσεων της εφαρμογής



Εικόνα 6.4: Διάγραμμα κλάσεων του υποσυστήματος διαχείρισης χώρου δεδομένων



Εικόνα 6.5: Διάγραμμα κλάσεων του υποσυστήματος συνδέσεων



Εικόνα 6.6: Διάγραμμα κλάσεων του υποσυστήματος μετασχηματισμού σχημάτων

6.1.2 Περιγραφή κλάσεων και μεθόδων

Υποσύστημα εισαγωγής πηγών	
SourcesManager	Αποτελεί την κεντρική κλάση του υποσυστήματος, αφού περιλαμβάνει λειτουργίες σχετικές με την εισαγωγή των πηγών και την μεταφορά δεδομένων από αυτές.
▪ *uploadFile*: φόρτωση αρχείου στο σύστημα και αποθήκευση στο τοπικό σύστημα αρχείων	

▪ *convertToJSON*: μετατροπή ενός excel αρχείου σε json μορφή ▪ *createDatasets*: δημιουργία datasets από εισηγμένα αρχεία δεδομένων ▪ *getJSONFileForQuery*: δημιουργία ενός αρχείου για την αποθήκευση δεδομένων στη μορφή JSON που προέρχονται από ένα sql ερώτημα ▪ *getFileForSPARQLQuery*: δημιουργία ενός αρχείου για την αποθήκευση δεδομένων ενός sparql ερωτήματος	
DBConnectionSource	Κλάση που μοντελοποιεί τη σύνδεση σε μια βάση δεδομένων. Μέσω της κλάσης αυτής γίνεται η μεταφορά δεδομένων με τη χρήση SQL ερωτημάτων.
▪ *setConnection*: ορισμός και έναρξη της σύνδεσης με τη βάση δεδομένων ▪ *getMetadata*: λήψη των μετα-δεδομένων του σχήματος ▪ *submitQuery*: εφαρμογή ενός sql ερωτήματος και λήψη αποτελεσμάτων	
SPARQLEndpointSource	Με την κλάση αυτή μοντελοποιείται ένα SPARQL endpoint από το οποίο λαμβάνονται RDF δεδομένα μέσω ερωτημάτων.
▪ *setURL*: ορισμός του URL που έχει το endpoint ▪ *executeSPARQLQuery*: εκτέλεση ενός sparql ερωτήματος στο endpoint ▪ *getStatements*: λήψη των τριπλετών ενός sparql ερωτήματος	
Converter	Interface που παρέχει λειτουργίες μετατροπής δεδομένων σε RDF. Η λειτουργία του είναι πολύ βασική καθώς χρησιμοποιείται στη δημιουργία των datasets.
▪ *getRDFProperties*: λήψη των rdf ιδιοτήτων που έχουν προκύψει από τις στήλες των πινακοειδών και σχεσιακών δεδομένων ▪ *getRDFClasses*: λήψη των rdf κλάσεων που χαρακτηρίζουν τα πινακοειδή και σχεσιακά δεδομένα ▪ *setInputFile*: ορισμός του αρχείου εισόδου ▪ *setOutputFile*: ορισμός του αρχείου εξόδου ▪ *setRecordClass*: ορισμός της κλάσης που χαρακτηρίζει την εγγραφή ▪ *setDatasetClass*: ορισμός της κλάσης που χαρακτηρίζει το dataset	
JSONToRDFConverter	Κλάση που υλοποιεί το interface Converter. Μετατρέπει δεδομένα που βρίσκονται σε JSON σε RDF.
Οι μέθοδοι της κλάσης αυτής υλοποιούν τις μεθόδους του αντίστοιχου interface	
TabularToRDFConverter	Κλάση που υλοποιεί το interface Converter. Μετατρέπει δεδομένα csv και xls αρχείων σε RDF.
Οι μέθοδοι της κλάσης αυτής υλοποιούν τις μεθόδους του αντίστοιχου interface	

ExcelParser	Κλάση που χρησιμοποιείται για την ανάγνωση xls αρχείων.
▪ *convertToJSON*: ανάγνωση και μετατροπή των δεδομένων ενός αρχείου excel σε μορφή JSON	
VocabularyExtractor	Βασική κλάση με τη βοήθεια της οποίας αναγνωρίζεται το λεξιλόγιο ενός RDF αρχείου. Χρησιμοποιείται σχεδόν σε όλα τα υποσυστήματα της εφαρμογής.
▪ *readData*: ανάγνωση ενός rdf αρχείου ▪ *getRDFClasses*: λήψη των κλάσεων του λεξιλογίου που χαρακτηρίζει τα rdf δεδομένα ▪ *getRDFProperties*: λήψη των ιδιοτήτων του λεξιλογίου που χαρακτηρίζει τα rdf δεδομένα	

Υποσύστημα διαχείρισης συνόλων δεδομένων και μετα-δεδομένων	
DataspaceManager	Μέσω της κλάσης αυτής γίνεται η διαχείριση των datasets που έχουν εισαχθεί καθώς και του dataspace.
▪ *getDataset*: με όρισμα το όνομα ενός dataset επιστρέφεται το αντίστοιχο dataset ▪ *addDataset*: καταχώρηση ενός dataset ▪ *getDatasetNames*: επιστρέφονται τα ονόματα των datasets που έχουν εισαχθεί ▪ *addLinkingTaskMetadataToDataset*: προστίθενται μετα-δεδομένα για το linking task (στη μορφή JSON) στο αντίστοιχο dataset ▪ *addMappingTaskMetadataToDataset*: προστίθενται μετα-δεδομένα για το mapping task (στη μορφή JSON) στο αντίστοιχο dataset ▪ *getDatasetMetadata*: επιστρέφονται τα μετα-δεδομένα ενός dataset σε μορφή JSON ▪ *getMetadata*: επιστρέφονται τα μετα-δεδομένα ενός ή περισσοτέρων datasets σε μορφή JSON ▪ *getLinkingTasksMetadata*: επιστρέφονται τα μετα-δεδομένα όλων των linking tasks που έχουν επικυρωθεί ▪ *addToDataspaceVocabulary*: προστίθενται όροι στο λεξιλόγιο του dataspace κάθε φορά που επικυρώνεται ένα mapping task ▪ *getDataspaceVocabulary*: επιστρέφεται το λεξιλόγιο του dataspace στη μορφή JSON ▪ *addToDataspaceModel*: επεκτείνεται το μοντέλο του dataspace με νέα δεδομένα κάθε φορά που επικυρώνεται ένα mapping task ▪ *addLinksToDataspaceModel*: προστίθενται οι σύνδεσμοι που έχουν προκύψει από την εκτέλεση ενός linking task όταν το τελευταίο επικυρώνεται ▪ *executeQueryToDataspace*: εκτελείται ένα SPARQL ερώτημα στο dataspace και επιστρέφονται τα αποτελέσματα	

<table>
<tr><td colspan="2">

- *getAllDataspaceProjectMetadata*: διαβάζονται όλα τα μετα-δεδομένα του συστήματος σχετικά με τα projects που έχουν αποθηκευτεί. Σημαντική μέθοδος για την εκκίνηση της εφαρμογής.
- *loadDataspaceProjectMetadata*: φόρτωση των μετα-δεδομένων ενός αποθηκευμένου project
- *createDataspaceProjectMetadata*: δημιουργία μετα-δεδομένων για ένα project
- *loadDataspaceProject*: φόρτωση των δεδομένων ενός project και σχηματισμός του αντίστοιχου dataspace
- *saveDataspaceProject*: αποθήκευση ενός project στο τοπικό σύστημα αρχείων
- *createDataspaceProject*: σχηματισμός ενός νέου project
- *buildCurrentDataspace*: βοηθητική μέθοδος για το σχηματισμό του dataspace ώστε να είναι δυνατή η εκτέλεση ερωτημάτων σε αυτό

</td></tr>
<tr><td>DataspaceProject</td><td>Σημαντική κλάση που μοντελοποιεί το project που σχετίζεται με τη διαδικασία σχηματισμού του dataspace. Κάθε project αποθηκεύεται στο τοπικό σύστημα αρχείων με συγκεκριμένη δομή αρχείων και φακέλων. Περιλαμβάνει τα αρχεία των datasets που έχουν εισαχθεί, τα μετα-δεδομένα τους καθώς και μετα-δεδομένα που αφορούν τη διαδικασία σχηματισμού του dataspace.</td></tr>
<tr><td colspan="2">

- *setLastModified*: ορισμός της τελευταίας ημερομηνίας τροποποίησης του project
- *get/set Name*: λήψη και ορισμός αντίστοιχα του ονόματος του project
- *getId* : λήψη του κωδικού του project. Σημαντική ιδιότητα των αντικειμένων της κλάσης καθώς με βάση αυτή γίνεται η αναζήτηση, φόρτωση και αποθήκευση των projects στο σύστημα.

</td></tr>
<tr><td>Dataset</td><td>Βασική κλάση που μοντελοποιεί ένα σύνολο δεδομένων.</td></tr>
<tr><td colspan="2">

- *get/set File*: λήψη/ορισμός του αρχείου που περιέχει τα δεδομένα του Dataset
- *get/set Vocabulary*: λήψη/ορισμός του λεξιλογίου του Dataset
- *get/set Metadata*: λήψη/ορισμός των μετα-δεδομένων του Dataset

</td></tr>
<tr><td>Dataspace</td><td>Σημαντική κλάση που μοντελοποιεί το χώρο δεδομένων που σχηματίζεται από τα εισηγμένα datasets .</td></tr>
<tr><td colspan="2">

- *build:* μέθοδος με την οποία σχηματίζεται το dataspace. Συγκεκριμένα, φορτώνονται στη μνήμη όλα τα αρχεία που αντιστοιχούν στα datasets
- *stopQueryExecution:* με τη μέθοδο αυτή διακόπτεται η εκτέλεση ενός sparql ερωτήματος στο dataspace
- *executeQuery:* με τη μέθοδο αυτή εκτελείται ένα sparql ερώτημα στο dataspace
- *empty*: καθαρισμός του dataspace

</td></tr>
</table>

▪ *getNamedGraphs*: λήψη των ονομαστικών γράφων του dataspace. Κάθε γράφος αντιστοιχεί και σε ένα εισηγμένο dataset ▪ *get/set SavedQueries*: μέθοδοι για τη λήψη και ορισμό των αποθηκευμένων ερωτημάτων. Τα ερωτήματα αυτά προέρχονται από τον QueryManager .

DatasetVocabulary	Η κλάση αυτή μοντελοποιεί το λεξιλόγιο-σχήμα ενός dataset.
Συνήθεις μέθοδοι get και set για τον ορισμό και τη λήψη των κλάσεων και των ιδιοτήτων του dataset.	
Metadata	Κλάση που αναπαριστά τα μετα-δεδομένα που συνοδεύουν ένα dataset.
▪ *addImportingMetadata, addLinkingMetadata, addMappingMetadata*: προσθήκη μετα-δεδομένων σχετικά με την εισαγωγή, τον μετασχηματισμό λεξιλογίου και τη διαδικασία σύνδεσης ενός dataset. Υπάρχουν και αντίστοιχες μέθοδοι get για τη λήψη των μετα-δεδομένων αυτών ▪ *hasRDFSource*: μέθοδος που δείχνει αν ένα dataset έχει ως πηγή ένα SPARQL endpoint ή ένα τοπικό αρχείο rdf	

Υποσύστημα μετασχηματισμού λεξιλογίων	
SchemaMappingManager	Αποτελεί την κεντρική κλάση του υποσυστήματος καθώς επικοινωνεί με τα υπόλοιπα υποσυστήματα και ελέγχει τις διαδικασίες μετασχηματισμού λεξιλογίων (schemaMapping tasks).
▪ *getRelatedMappingTask*: επιστρέφεται το mapping task ενός dataset ▪ *addMappingToTask*: προσθήκη ενός mapping στο mapping task ▪ *createMappingTask*: δημιουργία ενός mapping task ▪ *executeMappingTask*: εκτέλεση ενός mapping task ▪ *confirmMappingTask*: επιβεβαίωση ενός mapping task ▪ *addMappingsToGenerator*: προσθήκη των mappings ενός task στον LinkGenerator ▪ *getSourceVocabulary*: λήψη του λεξιλογίου ενός dataset στη μορφή JSON ▪ *getExternalVocabularies*: λήψη των εξωτερικών λεξιλογίων που έχουν εισαχθεί από το χρήστη ▪ *addExternalVocabulary*: προσθήκη εξωτερικού λεξιλογίου στο σύστημα	
SchemaMappingTask	Κλάση που μοντελοποιεί τη διαδικασία μετάφρασης ενός dataset. Κατασκευάζει το αρχείο με τα mappings και επικοινωνεί με το R2R Framework.

▪ *set/get InputDataset*: ορισμός/λήψη του dataset στο οποίο εφαρμόζεται το task ▪ *set/get PropertyMappingList*: ορισμός/λήψη των mappings του task ▪ *createMappingFile*: δημιουργία του αρχείου των mappings ▪ *set/get MappingFileLocation*: ορισμός/λήψη της τοποθεσίας του mapping file στο τοπικό σύστημα αρχείων ▪ *set/get OutputFile*: ορισμός/λήψη του αρχείου που αποθηκεύονται τα rdf δεδομένα που παράγονται από το task ▪ *start*: έναρξη του task (εδώ γίνεται η χρήση του R2R Framework) ▪ *confirm*: επιβεβαίωση ενός task ▪ *createMappingResultOverview*: δημιουργία των αποτελεσμάτων του task σε JSON μορφή	
MappingFile	Η κλάση αυτή αναπαριστά το αρχείο που δίνεται ως είσοδος στο R2R Framework και περιλαμβάνει τις λειτουργίες εγγραφής του αρχείου στο τοπικό σύστημα αρχείων.
▪ *save*: αποθήκευση του αρχείου στο τοπικό σύστημα αρχείων ▪ *writeMappings*: εγγραφή των mappings στο αρχείο ▪ *writePrefixes*: εγγραφή των prefixes στο αρχείο	
Mapping	Κλάση που μοντελοποιεί την αντιστοιχία ανάμεσα σε όρους λεξιλογίων σύμφωνα με το R2R Framework.
▪ *set/get SourcePattern*: ορισμός/λήψη του source pattern του mapping ▪ *set/get TargetPatterns*: ορισμός/λήψη των target patterns του mapping ▪ *setPrefixesMap*: ορισμός των ζευγών prefix-namespace του mapping ▪ *getTargetVocabulary*: λήψη των όρων που ανήκουν στα target patterns του mapping και αποτελούν το λεξιλόγιο-στόχο γι' αυτό το mapping	
TargetVocabularyManager	Η κλάση αυτή χρησιμοποιείται για τη φόρτωση λεξιλογίων στο σύστημα.
▪ *importVocabulary*: εισαγωγή ενός λεξιλογίου στο σύστημα ▪ *getAllVocabularies*: λήψη όλων των εισηγμένων λεξιλογίων	

Υποσύστημα συνδέσεων	
LinkingManager	Είναι η κεντρική κλάση του υποσυστήματος καθώς επικοινωνεί με τα υπόλοιπα υποσυστήματα και ελέγχει τις διαδικασίες συνδέσεων (linking tasks).
▪ *getFunctions*: λήψη των συναρτήσεων του Silk Framework από τον FunctionManager	

▪ *createPredefinedLinkingTasks*: δημιουργία προτεινόμενων linking tasks με βάση τα υπάρχοντα mappings ▪ *createLinkingTask*: δημιουργία (και εκτέλεση) ενός linking task ▪ *confirmLinkingTask*: επικύρωση ενός linking task ▪ *getLinkingTaskResultLinks*: λήψη των συνδέσεων που έχουν προκύψει για ένα linking task ▪ *getLinkingTasks*: λήψη όλων των επικυρωμένων linking tasks ▪ *getVariablesOfPattern*: εξαγωγή των μεταβλητών από ένα sparql pattern ▪ *getPathsFromSPARQLClause*: εξαγωγή των ιδιοτήτων που προκύπτουν από μια μεταβλητή ενός sparql WHERE clause	
LinkingTask	Κλάση που αναπαριστά μια διαδικασία σύνδεσης. Κατασκευάζει το αρχείο XML και επικοινωνεί με το Silk Framework.
▪ *set/get LinkingFileLocation*: ορισμός/λήψη της τοποθεσίας του αρχείου XML του task ▪ *setOutputLinksLocation*: ορισμός της τοποθεσίας των αρχείων εξόδου του task ▪ *createLinkingFile*: δημιουργία του αρχείου XML του task ▪ *start*: έναρξη του task (χρήση του Silk Framework) ▪ *getAcceptedOutputLinks*: λήψη των συνδέσεων που θεωρούνται ως αποδεκτές για τον χρήστη ▪ *getOutputLinksToVerify*: λήψη των συνδέσεων που χρειάζονται επιβεβαίωση από το χρήστη	
SilkLSLFile	Με την κλάση αυτή μοντελοποιείται το αρχείο XML που δίνεται ως είσοδος στο Silk Framework.
▪ *save*: αποθήκευση του αρχείου στο τοπικό σύστημα αρχείων ▪ *writeDatasources*: καταγραφή στο XML αρχείο του τμήματος των πηγών δεδομένων ▪ *writeFilter*: καταγραφή στο XML αρχείο του τμήματος για το φίλτρο ▪ *writeOutputs*: καταγραφή στο XML αρχείο του τμήματος της εξόδου ▪ *writePrefixes*: καταγραφή στο XML αρχείο του τμήματος των prefixes ▪ *writeInterlinks*: καταγραφή στο XML αρχείο του τμήματος του κανόνα σύνδεσης	
FunctionManager	Κλάση που μοντελοποιεί τις συναρτήσεις που χρησιμοποιεί το Silk Framework.
Μέθοδοι που κατασκευάζουν τους ορισμούς των συναρτήσεων σε JSON	
LinkageRule	Είναι η κλάση που αναπαριστά τον κανόνα βάσει του οποίου θα γίνουν οι συνδέσεις. Δέχεται τις παραμέτρους σε μορφή json και παράγει το αντίστοιχο τμήμα του αρχείου XML.
▪ *traverseTree*: διάσχιση του κανόνα και παραγωγή του αντίστοιχου τμήματος XML	

▪ *writeTo*: εγγραφή του κανόνα στο XML αρχείο	
LinkGenerator	Κλάση με την οποία παράγονται οι προτεινόμενες συνδέσεις. Ο SchemaMappingManager τοποθετεί τα mappings των tasks και ο LinkingManager λαμβάνει τα αποτελέσματα.
▪ *generateLinks*: έναρξη της διαδικασίας παραγωγής των συνδέσεων ▪ *addDatasetMapping*: προσθήκη ενός mapping ▪ *findEqualPropertyMappings*: εύρεση όμοιων ιδιοτήτων ανάμεσα στα mappings	

Υποσύστημα εφαρμογής SPARQL ερωτημάτων	
QueryManager	Αποτελεί τη μοναδική κλάση του υποσυστήματος και βοηθά στην εκτέλεση Sparql ερωτημάτων στο dataspace της εφαρμογής. Επίσης, μορφοποιεί κατάλληλα τα δεδομένα ώστε να παρουσιαστούν στο γραφικό περιβάλλον της εφαρμογής.
▪ *executeQuery*: μεταφορά ενός SPARQL ερωτήματος στον DatasetManager για εκτέλεση και λήψη αποτελεσμάτων ▪ *getDataspaceVocabulary*: λήψη του λεξιλογίου του dataspace από τον DatasetManager ▪ *getPrefixes*: λήψη των prefixes του συστήματος για την παρουσίασή τους στο γραφικό περιβάλλον της εφαρμογής	

Υποσύστημα διαχείρισης της υπηρεσίας	
ServiceManager	Είναι η κεντρική κλάση του υποσυστήματος και γενικά της εφαρμογής καθώς με τη χρήση αυτής γίνεται η επικοινωνία των υποσυστημάτων. Ακόμη, βασικός της ρόλος είναι η σύνδεση του επιπέδου (υποσυστήματος) επικοινωνίας με τον πυρήνα της εφαρμογής.
▪ *initializeCommands*: βασική μέθοδος με την οποία αντιστοιχίζονται οι εντολές με τις κλάσεις Command	
FileManager	Με την κλάση αυτή οργανώνονται τα αρχεία της εφαρμογής στο τοπικό σύστημα αρχείων
▪ *loadDataspaceMetadata:* φόρτωση των μετα-δεδομένων ενός dataspace που σχετίζεται με ένα project. Η φόρτωση γίνεται βάσει του id του αντίστοιχου project ▪ *saveDataspaceMetadata:* αποθήκευση των μετα-δεδομένων του dataspace που σχετίζεται με ένα project στο φάκελο του project του τοπικού συστήματος αρχείων ▪ *loadWorkspaceMetadata*: φόρτωση των μετα-δεδομένων της εφαρμογής που	

<table>
<tr><td colspan="2">περιλαμβάνουν πληροφορίες για το ποια projects έχουν αποθηκευτεί
▪ saveWorkspaceMetadata: αποθήκευση των μετα-δεδομένων της εφαρμογής που περιλαμβάνουν πληροφορίες για τα εισηγμένα projects. Τα μετα-δεδομένα είναι στην ουσία τα ids των projects.
▪ writeDatasetToProjectDir: εγγραφή του αρχείου που περιέχει τα δεδομένα ενός dataset στον κατάλληλο φάκελο του project
▪ getDataspaceProjectDir: λήψη του φακέλου που είναι αποθηκευμένο ένα project
▪ deleteDataspaceProjectDir: διαγραφή του φακέλου ενός project από τη μνήμη
▪ createDataspaceProjectDir: δημιουργία του φακέλου ενός project στη μνήμη</td></tr>
<tr><td>PrefixesManager</td><td>Πρόκειται για την κλάση που περιλαμβάνει τον κεντρικό κατάλογο των ζευγών namespace-prefix. Επιτελεί και όπως λειτουργίες, όπως αναζήτηση και εισαγωγή ζευγών.</td></tr>
<tr><td colspan="2">▪ init: αρχικοποίηση και εισαγωγή κάποιων γνωστών ζευγών prefix-namespace
▪ addPrefix: προσθήκη ενός ζεύγους prefix-namespace
▪ getPrefix: λήψη ενός ζεύγους prefix-namespace
▪ getVocabularyTermInJSONFormat: λήψη πληροφορίας σε JSON για το prefix, το namespace και το όνομα ενός όρου λεξιλογίου</td></tr>
<tr><td>IntegrationServiceServlet</td><td>Είναι η κλάση στην οποία βασίζεται το υποσύστημα επικοινωνίας της εφαρμογής. Ακόμη, στην κλάση αυτή στέλνονται όλα τα αιτήματα του προγράμματος-πελάτη και εκείνη με τη σειρά της τα προωθεί στις κατάλληλες κλάσεις για τη διεκπεραίωσή τους.</td></tr>
<tr><td colspan="2">▪ service: αναλύεται το path του http αιτήματος, βρίσκεται η αντίστοιχη εντολή και ενεργοποιείται η κατάλληλη κλάση Command</td></tr>
</table>

<table>
<tr><th colspan="2">Υποσύστημα επικοινωνίας με τον client</th></tr>
<tr><td>Command</td><td>Είναι βασική για το υποσύστημα καθώς μοντελοποιεί την «εντολή» που στέλνει ο client στο server. Κληρονομεί την κλάση IntegrationServiceServlet. Με τη σειρά της, η κλάση Command κληρονομείται από όλες τις άλλες κλάσεις-εντολές του συστήματος επικοινωνίας που αναπαριστούν επιμέρους λειτουργίες-εντολές του γραφικού περιβάλλοντος της εφαρμογής. Οι υπόλοιπες κλάσεις του υποσυστήματος χωρίζονται σε κατηγορίες ανάλογα με το υποσύστημα που επικοινωνούν. Όλες οι κλάσεις υλοποιούν τις μεθόδους doGet και doPost ώστε να εκτελεστούν οι απαραίτητες ενέργειες. Για το λόγο αυτό αυτό δεν θα αναλυθούν περισσότερο οι μέθοδοι των ακόλουθων κλάσεων.</td></tr>
</table>

General	
PrefixesCommand	Εντολή για την εισαγωγή/λήψη ζευγών namespace-prefix.
GetDatasetMetadataCommand	Εντολή για τη λήψη των μετα-δεδομένων για κάποιο dataset.
GetLinkingTasksMetadataCommand	Εντολή για τη λήψη πληροφοριών για τα linking tasks που έχουν γίνει.
Imporitng	
CreateDatasetsCommand	Εντολή για δημιουργία των datasets από τα εισηγμένα δεδομένα.
GetRelationalMetadataCommand	Εντολή για τη λήψη των μετα-δεδομένων του εισηγμένου σχεσιακού σχήματος βάσης δεδομένων.
ImportJSONDataCommand	Εντολή για εισαγωγή δεδομένων στη μορφή JSON.
ImportRelationalQueryCommand	Εντολή για εισαγωγή δεδομένων από sql ερώτημα.
ImportSPARQLEndpointCommand	Εντολή για εισαγωγή ενός sparql endpoint.
ImportTabularFileCommand	Εντολή για εισαγωγή ενός πινακοειδούς αρχείου (csv και xls).
ImportTriplesFileCommand	Εντολή για εισαγωγή RDF αρχείου που περιέχει τριπλέτες.
Mapping	
ExternalVocabularyCommand	Εντολή για εισαγωγή ενός λεξιλογίου με τη βοήθεια υπάρχοντος τοπικού αρχείου καθώς και για τη λήψη των ονομάτων των εισηγμένων λεξιλογίων.
GetDatasetVocabularyCommand	Εντολή για λήψη του λεξιλογίου ενός dataset.
GetUndefinedNamespacesCommand	Εντολή για λήψη των namespaces για τα οποία δεν έχει βρεθεί κάποιο prefix.
StartMappingTaskCommand	Εντολή για έναρξη ενός mapping task.
SaveMappingTaskCommand	Εντολή για αποθήκευση (επιβεβαίωση) ενός mapping task.
Linking	

GetLinkageRuleFunctionsCommand	Εντολή για τη λήψη και παρουσίαση των συναρτήσεων του Silk Framework στο γραφικό περιβάλλον.
GetLinksCommand	Εντολή για τη λήψη των συνδέσεων που έχουν προκύψει από κάποιο linking task.
GetPathInfoCommand	Εντολή για τη λήψη πληροφορίας σχετικά με τα paths ενός SPARQL ερωτήματος.
GetRecommendedLinkingTasks Command	Εντολή για τη λήψη των προτεινόμενων linking tasks που έχουν εξαχθεί από το σύστημα.
StartLinkingTaskCommand	Εντολή για έναρξη ενός linking-task.
SaveLinkingTaskCommand	Εντολή για αποθήκευση (επιβεβαίωση) ενός linking-task.
Queries	
ExecuteSPARQLQueryCommand	Εντολή για την εκτέλεση ενός sparql ερωτήματος
StopSPARQLQueryExecution Command	Εντολή για τη διακοπή της εκτέλεσης ενός sparql ερωτήματος
GetDataspaceQueriesCommand	Εντολή για τη λήψη των ερωτημάτων που έχουν γίνει στο dataspace σε παρελθόντα χρόνο
SaveDataspaceQueriesCommand	Εντολή για την αποθήκευση των ερωτημάτων που έχουν γίνει στο dataspace
RenameSPARQLQueryCommand	Μετονομασία ενός ερωτήματος
SPARQLQueryCommand	Εντολή για τη δημιουργία και τη φόρτωση ενός ερωτήματος
BuildDataspaceCommand	Εντολή για το σχηματισμό του dataspace όταν έχουν εισαχθεί όλα τα datasets
Dataspace	
DataspaceProjectCommand	Εντολή για τη δημιουργία, τη φόρτωση και την αποθήκευση ενός dataspace project
GetAllDataspaceProjectMetadata Command	Εντολή για τη λήψη όλων των μετα-δεδομένων που αφορούν τα αποθηκευμένα projects.
GetDataspaceVocabularyCommand	Εντολή για τη λήψη του λεξιλογίου του dataspace που ανήκει στο τρέχον project
GetNamedGraphsCommand	Εντολή για τη λήψη των ονομαστικών γράφων

	που υπάρχουν στο τρέχον dataspace

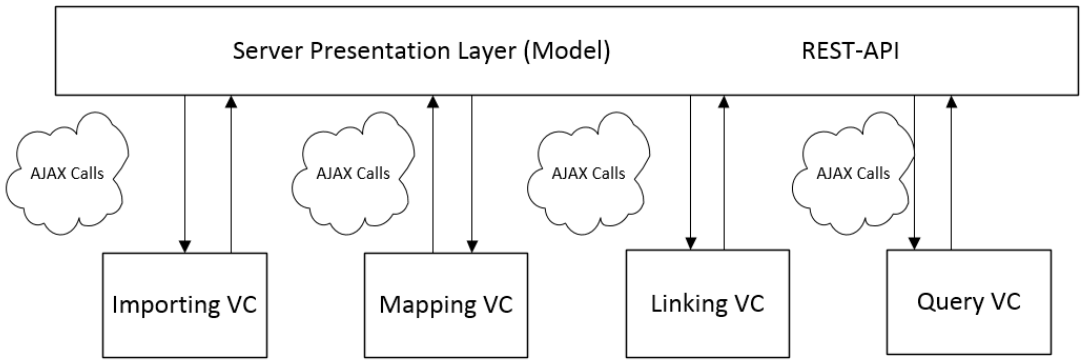
6.2 Αρχιτεκτονική προγράμματος - πελάτη (Client - Side)

Το πρόγραμμα-πελάτης που εκτελείται με τη βοήθεια του browser διαδραματίζει σημαντικό ρόλο καθώς με αυτό αλληλεπιδρά ο χρήστης ώστε να γίνουν όλες οι λειτουργίες της διαδικασίας ολοκλήρωσης. Το σκεπτικό πάνω στο οποίο βασίστηκε η οργάνωση και η αρχιτεκτονική του κώδικα του client είναι η αλληλουχία των βημάτων που απαιτούνται για την ολοκλήρωση δεδομένων. Όπως έχει προαναφερθεί, η διαδικασία αυτή απαιτεί την εισαγωγή των δεδομένων, το μετασχηματισμό των διαφορετικών σχημάτων σε ένα ενιαίο σχήμα και τη σύνδεση των δεδομένων. Το γραφικό περιβάλλον της εφαρμογής ακολουθώντας την παραπάνω διαδικασία οργανώνεται σε βήματα. Σε κάθε στάδιο-βήμα υλοποιούνται οι αντίστοιχες ενέργειες:

- **Βήμα 1:** εισάγονται οι πηγές και λαμβάνονται δεδομένα από αυτές. Στο τέλος του βήματος αυτού σχηματίζονται τα σύνολα δεδομένων
- **Βήμα 2:** σκοπός του βήματος αυτού είναι η δημιουργία mapping tasks προκειμένου τα διαφορετικά λεξιλόγια-σχήματα των datasets να μετατραπούν σε ένα ενιαίο λεξιλόγιο-σχήμα
- **Βήμα 3:** στο βήμα αυτό ορίζονται και εκτελούνται διαδικασίες συνδέσεων παρόμοιων οντοτήτων
- **Βήμα 4:** αποτελεί το τελικό στάδιο της εφαρμογής, όπου εφαρμόζονται SPARQL ερωτήματα στο χώρο δεδομένων (dataspace) που έχει δημιουργηθεί

Να σημειωθεί πως τα βήματα αυτά αφορούν στη διαδικασία σχηματισμού ενός νέου dataspace. Η εφαρμογή παρέχει τη δυνατότητα φόρτωσης ενός υπάρχοντος dataspace ώστε ο χρήστης να εφαρμόσει εκ νέου και να εκτελέσει ξανά ερωτήματα που έχουν αποθηκευτεί στο αντίστοιχο project σε παρελθόντα χρόνο.

Σε κάθε στάδιο το πρόγραμμα-πελάτης αλληλεπιδρά με τον server χρησιμοποιώντας κλήσεις AJAX. Μια σχηματική αναπαράσταση της αρχιτεκτονικής φαίνεται στο ακόλουθο σχήμα:



Εικόνα 6.7: Μοντέλο MVC για το πρόγραμμα-πελάτη της εφαρμογής

Να σημειωθεί πως ο client είναι σχεδιασμένος σύμφωνα με το αρχιτεκτονικό σχήμα MVC (Model-View-Controller). Το μοντέλο (model) περιλαμβάνει τις διαδικασίες που εκτελούνται στο σύστημα καθώς και τα ίδια τα δεδομένα. Στην προκειμένη περίπτωση το μοντέλο αντιστοιχεί στο επίπεδο-υποσύστημα παρουσίασης (presentation layer-module) της εφαρμογής. Ο ελεγκτής (Controller) είναι εκείνη η μονάδα που επικοινωνεί με τον server ανταλλάσσοντας μηνύματα HTTP. Η μονάδα View περιλαμβάνει όλες τις λειτουργίες παρουσίασης των πληροφοριών και των δεδομένων στο χρήστη. Ο client περιέχει έναν ελεγκτή για κάθε στάδιο της διαδικασίας και αρκετές μονάδες παρουσίασης.

Με βάση τον τρόπο που οργανώνεται το πρόγραμμα-πελάτης γίνεται αντιληπτό πως είναι δυνατή η επέκταση της εφαρμογής χωρίς να επηρεαστούν οι υπάρχουσες λειτουργίες. Η επέκταση μπορεί να γίνει με τον εμπλουτισμό του REST API και την προσθήκη μιας ή περισσότερων μονάδων VC. Βέβαια, αυτό συνεπάγεται την παροχή περισσότερων λειτουργιών από την εφαρμογή που βρίσκεται στον server.

7

Υλοποίηση

7.1 Λεπτομέρειες υλοποίησης

Δύο βασικά θέματα που πραγματεύεται η εφαρμογή είναι η μετατροπή δεδομένων σε RDF και η εξαγωγή προτεινόμενων διαδικασιών συνδέσεων με βάση τα mappings που έχουν εισαχθεί. Τα θέματα αυτά αναλύονται στις δυο επόμενες ενότητες.

7.1.1 Μετατροπή δεδομένων σε RDF

Η μετατροπή δεδομένων που δεν βρίσκονται σε μορφή RDF ή αλλιώς triplification είναι πολύ σημαντική διαδικασία σε ένα σύστημα ολοκλήρωσης ετερογενών δεδομένων. Η σημασία της έγκειται στο γεγονός ότι τα δεδομένα που βρίσκονται στην ίδια μορφή είναι πιο εύκολο να αναλυθούν και να επεξεργαστούν. Με τον τρόπο αυτό αυξάνεται η διαλειτουργικότητα των δεδομένων του συστήματος (data interoperability). Η προτεινόμενη διαδικασία μετατροπής βασίζεται σε πινακοειδή και σχεσιακά δεδομένα. Ακόμη, προκειμένου να μειωθεί το μέγεθος του παραγόμενου αρχείου χρησιμοποιείται το συντακτικό Turtle (Terse RDF Triple Language) της γλώσσας RDF, όπως αυτό περιγράφεται στην τελευταία σύσταση του οργανισμού W3C[23].

Προκειμένου να πραγματοποιηθεί η μετατροπή των δεδομένων σε RDF χρειάζεται ένα RDF σχήμα. Το προτεινόμενο σχήμα φαίνεται στην Εικόνα 7.2. Πιο συγκεκριμένα,

[23] *http://www.w3.org/TR/turtle/*

ορίζονται οι κλάσεις Dataset και Record καθώς και ιδιότητες που προκύπτουν από το όνομα κάθε στήλης. Η κλάση Dataset αναπαριστά το σύνολο δεδομένων και η κλάση Record αναπαριστά την γραμμή. Κάθε γραμμή συνδέεται μέσω των ιδιοτήτων με τα κελιά της και κάθε κελί συνδέεται με την τιμή του μέσω της ιδιότητας hasValue. Το σύνολο δεδομένων και κάθε γραμμή, στήλη και κελί αναπαρίστανται με ένα URI. Έτσι, αποκτά νόημα η σύνδεση των κελιών που αναπαριστούν την ίδια οντότητα του φυσικού κόσμου. Για παράδειγμα, αυτό είναι ιδιαίτερα σημαντικό σε σύνολα στατιστικών δεδομένων όπου οι στήλες μπορεί να είναι γεωγραφικές παράμετροι, όπως ονόματα περιοχών, πόλεων ή χωρών. Συνδέοντας τα κελιά στην ουσία συνδέονται τα URIs των αντίστοιχων πόλεων με αποτέλεσμα να είναι δυνατός ο συνδυασμός διαφορετικών συνόλων δεδομένων.

Προκειμένου να μετατραπούν τα πινακοειδή και σχεσιακά δεδομένα σε RDF χρειάζεται η διάσχισή τους. Έτσι, χρησιμοποιείται μια κλασσική επαναληπτική δομή ώστε να προσπελαστεί κάθε κελί του πίνακα (συνόλου δεδομένων). Ο σχηματισμός των URIs και των τριπλετών βασίζεται στον ακόλουθο πίνακα:

baseURI	`http://localhost:8080/`

ζεύγος vocabulary namespace URI – prefix	σχεσιακά δεδομένα:
	`baseURI+"rdb/"+schema_name+"/schemaDef#"`
	`"rdb-"+schema_name`

	πινακοειδή δεδομένα αρχείων:
	`baseURI+"tab/"+file_name+"/schemaDef#"`
	`"tab-"+file_name`

datasetURI	`baseURI + dataset_name`
rowURI (subject URI)	`datasetURI + "/rowID/" + rowID`
predicateURI	`vocabulary_namespaceURI + column_name`
cell value URI (object URI)	`rowURI + column_name + "/" + cell_value`

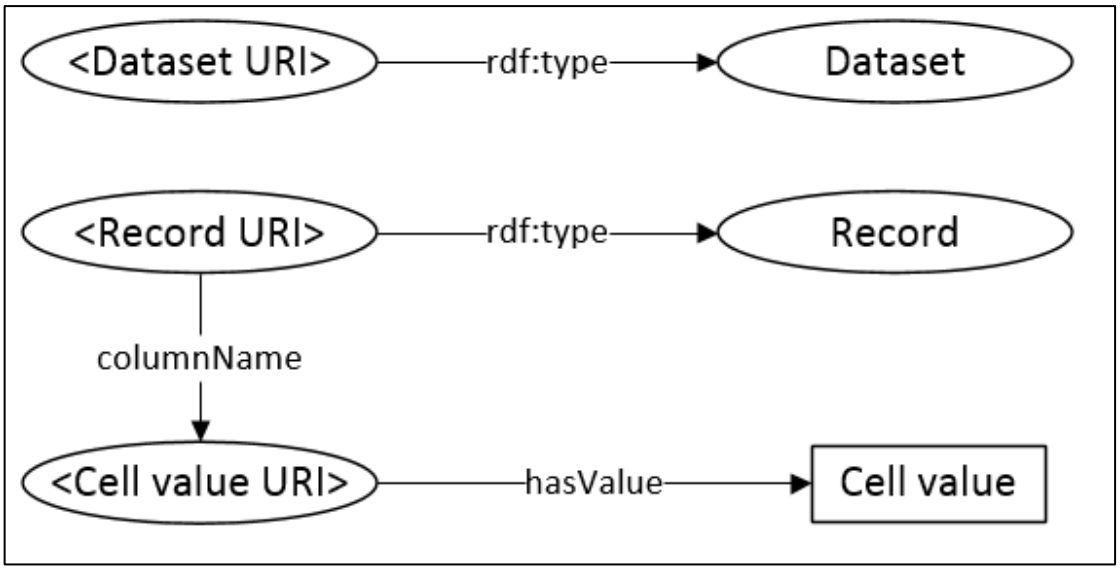
Σύμφωνα με τα παραπάνω προκύπτει ο αλγόριθμος μετατροπής που φαίνεται στην Εικόνα 7.1. Στο βήμα 1 εισάγονται τα απαραίτητα ζεύγη prefixes-namespaces ώστε να είναι δυνατή η ανάγνωση του προκύπτοντος RDF αρχείου. Στη συνέχεια (βήμα 2) δημιουργούνται οι ιδιότητες της οντολογίας από τα ονόματα των στηλών. Στο σημείο αυτό θα πρέπει να σημειωθεί πως τα ονόματα των στηλών μεταβάλλονται ελαφρώς σε περίπτωση που περιέχουν

κάποιον χαρακτήρα μη συμβατό με το συντακτικό turtle[24]. Τα ονόματα των στηλών θεωρείται απαραίτητο να βρίσκονται στην πρώτη γραμμή του συνόλου δεδομένων. Τα βήματα 3 και 4 αφορούν το σχηματισμό των τριπλετών σύμφωνα με την οντολογία-σχήμα του συνόλου δεδομένων.

Αλγόριθμος μετατροπής σε RDF

```
1. Εισαγωγή απαραίτητων ζευγών prefixes-namespaces
2. Εξαγωγή RDF ιδιοτήτων από τα ονόματα των στηλών του αρχείου
3. Εισαγωγή rdf statement για τον προσδιορισμό του dataset
4. Για κάθε σειρά του αρχείου {
      a. Σχηματισμός Subject που προσδιορίζει την σειρά
      b. Για κάθε στήλη του αρχείου {
            i. Σχηματισμός Predicate που προσδιορίζει τη στήλη
           ii. Σχηματισμός Object που προσδιορίζει το κελί
          iii. Σχηματισμός επιπρόσθετου RDF statement που
               προσδιορίζει την τιμή του κελιού
      }
      c. Καταγραφή της RDF αναπαράστασης της σειράς στο νέο αρχείο
   }
```

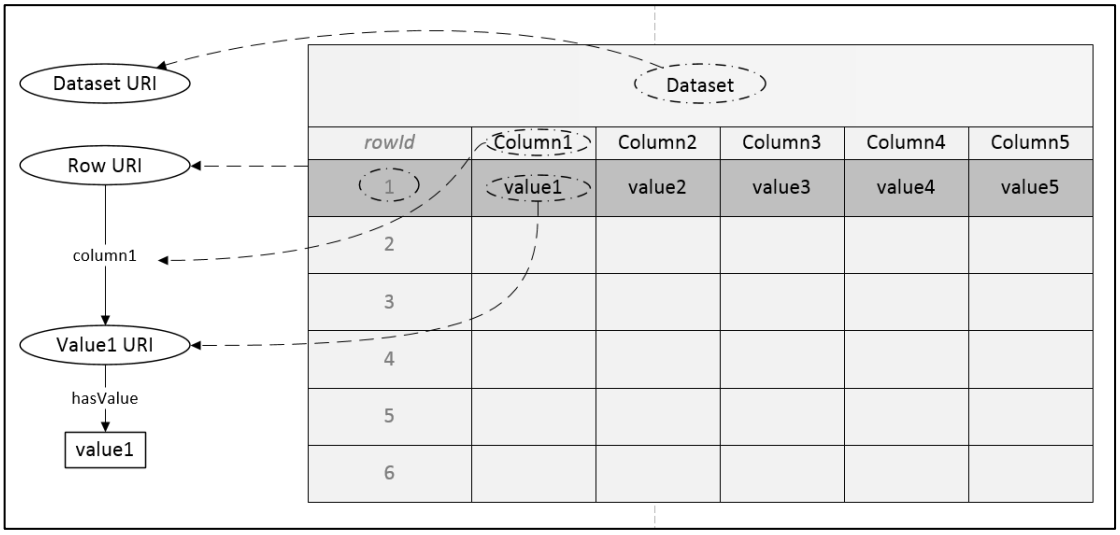
Εικόνα 7.1: Αλγόριθμος μετατροπής σε RDF



Εικόνα 7.2: Οντολογία-σχήμα για τη μοντελοποίηση ενός συνόλου δεδομένων οργανωμένο σε πίνακα

[24] *Η μεταβολή αφορά την αντικατάσταση του μη συμβατού χαρακτήρα με το σύμβολο της κάτω παύλας (underscore) «_».*

Στην Εικόνα 7.3 φαίνεται ένα παράδειγμα μετατροπής ενός συνόλου δεδομένων σε RDF. Τα βέλη δείχνουν την αντιστοιχία ανάμεσα στις τιμές του συνόλου δεδομένων και στα URIs των οντοτήτων που προκύπτουν. Αναλυτικά, οι παραγόμενες τριπλέτες σε turtle μορφή φαίνονται στον πίνακα που ακολουθεί. Με *voc-prefix* συμβολίζεται το εκάστοτε prefix του dataset που σχηματίζεται.



Εικόνα 7.3: Παράδειγμα μετατροπής μιας γραμμής ενός συνόλου δεδομένων σε RDF

```
<http://localhost:8080/exampleDataset> rdf:type voc-prefix:Dataset.

<http://localhost:8080/exampleDataset/rowID/1> rdf:type voc-prefix:Record;
voc-prefix:Column1 <http://localhost:8080/exampleDataset/rowID/1/Column1/value1>.

<http://localhost:8080/exampleDataset/rowID/1/Column1/value1>
voc-prefix:hasValue value1.
```

Αναπαράσταση μιας γραμμής σε RDF (συγκεκριμένα σε συντακτικό turtle)

7.1.2 Εξαγωγή προτεινόμενων linking tasks βασισμένα στα mappings

Σκοπός της μετάφρασης των λεξιλογίων-σχημάτων των datasets είναι η ομογενοποίηση των διαφορετικών οντολογιών σε ένα ενιαίο σχήμα. Με τον τρόπο αυτό όροι με διαφορετική ονομασία αλλά παρόμοιας σημασίας μπορούν να μεταφραστούν σε έναν κοινό όρο. Μεταφράζοντας, λοιπόν, τα επιμέρους διαφορετικά λεξιλόγια σε ένα κοινό σχήμα είναι πιο εύκολη η εξαγωγή συνδέσεων.

Πιο συγκεκριμένα, συγκεντρώνοντας τα mappings για τα σύνολα δεδομένων που προέρχονται από πινακοειδή και σχεσιακά δεδομένα, βρίσκονται οι κοινοί όροι. Η επιλογή μόνο αυτών των datasets οφείλεται στο γεγονός ότι είναι γνωστό το σχήμα που τα χαρακτηρίζει με αποτέλεσμα να είναι πιο εύκολη η αυτόματη συγγραφή του κανόνα

συνδέσεων για το Silk Framework. Στην ουσία αυτό που προσπαθεί να επιτύχει ο αλγόριθμος είναι να αναγνωρίσει τις όμοιες ιδιότητες μεταξύ των datasets και κατ' επέκταση τις σημασιολογικά όμοιες στήλες των αρχικών πινακοειδών και σχεσιακών δεδομένων. Ακόμη, οι κοινοί όροι που αντιπροσωπεύουν κλάσεις απορρίπτονται αφού δεν έχει πρακτικό ενδιαφέρον η εξαγωγή σύνδεσης από αυτούς. Το φιλτράρισμα αυτό δεν αποτελεί ενέργεια του αλγορίθμου αλλά εφαρμόζεται εξωτερικά από την εφαρμογή (στο υποσύστημα συνδέσεων). Ο αλγόριθμος που χρησιμοποιείται είναι ο ακόλουθος:

Αλγόριθμος εξαγωγής συνδέσεων από mappings

```
1. M = {};
2. Mappings = collectMappings();
3. foreach mapping in Mappings do{
        if (hasLiteralPropertyMapping(mapping)){
             M = M U {mapping};
        }
   }
4. Map<Term,DatasetNamesArray> map = findCommonTerms(M);
5. Links = getLinks(map);
6. return Links;
```

Ο πίνακας Mappings είναι ο συγκεντρωτικός πίνακας όλων των mappings που αφορούν τα datasets. Κάθε mapping ελέγχεται ώστε να εξασφαλιστεί η ύπαρξη ιδιότητας που έχει ως αντικείμενο ένα λεκτικό (literal). Η προϋπόθεση αυτή είναι πολύ σημαντική για να έχει ουσία ο παραγόμενος κανόνας σύνδεσης που θα εισαχθεί στο Silk Framework, καθώς οι συγκρίσεις βασίζονται σε literals. Έτσι, σχηματίζεται το σύνολο M βάσει του οποίου προκύπτει μια αντιστοιχία ανάμεσα σε όρους[25] και στα σύνολα δεδομένων που εμφανίζονται. Για κάθε όρο που αντιστοιχίζεται σε περισσότερα από ένα datasets προκύπτει δυνατότητα σύνδεσης των datasets αυτών με βάση αυτόν τον όρο. Προκειμένου να μειωθεί η επαναληπτικότητα στις συνδέσεις συνδέεται το πρώτο dataset με όλα τα άλλα που ακολουθούν.

[25] *Οι όροι ανήκουν στο λεξιλόγιο του dataspace.*

7.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Η εφαρμογή αναπτύχθηκε με τη βοήθεια του εργαλείου (Ολοκληρωμένου Συστήματος Ανάπτυξης, IDE) Eclipse στην έκδοση Juno. Το τμήμα της εφαρμογής που λειτουργεί στο server είναι υλοποιημένο σε Java. Το πρόγραμμα-πελάτης υλοποιήθηκε με τη βοήθεια γνωστών web τεχνολογιών. Πιο συγκεκριμένα, χρησιμοποιείται η γλώσσα HTML για τη σχεδίαση των σελίδων που βλέπει ο χρήστης, η γλώσσα CSS για τη διάταξη των σελίδων και η γλώσσα JavaScript (κυρίως οι ισχυρές βιβλιοθήκες jQuery και jQuery UI) για τη δυνατότητα ανάπτυξης συνθετότερων ενεργειών.

Το σύστημα αποτελεί ένα Dynamic Web Project. Το περιεχόμενο του client βρίσκεται στον φάκελο WebContent ενώ οι κλάσεις της εφαρμογής βρίσκονται στον φάκελο main/src. Ακόμη, οι εξωτερικές βιβλιοθήκες της εφαρμογής είναι συγκεντρωμένες (σε jar αρχεία) στο φάκελο libs. Το project χρειάζεται να τοποθετηθεί σε έναν server, όπως είναι ο Apache Tomcat 7.0, προκειμένου να εκτελεστεί. Η εφαρμογή μπορεί να μετατραπεί σε zip αρχείο κι έτσι είναι δυνατή η εκτέλεσή της σε οποιοδήποτε υπολογιστή μπορεί να υποστηρίξει το περιβάλλον Eclipse. Η μόνη απαίτηση της εφαρμογής είναι η χρήση σύγχρονου browser λόγω της χρήσης των βιβλιοθηκών jQuery και jQuery UI.

Η διαδικασία εγκατάστασης της εφαρμογής στο περιβάλλον Eclipse είναι η ακόλουθη:

1. **Εισαγωγή του Project**: Από το κεντρικό μενού επιλέγουμε File → Import. Στο μενού που παρουσιάζεται μεταβαίνουμε στην κατηγορία General → Existing Projects into Workspace και πατάμε Next. Στην πρώτη επιλογή πατάμε Browse και επιλέγουμε τον φάκελο που είναι αποθηκευμένο το project της εφαρμογής. Επιλέγουμε το project στο πλαίσιο Projects που υπάρχει από κάτω (αν δεν είναι αυτόματα προεπιλεγμένο) και στη συνέχεια πατάμε Finish. Πλέον, μπορούμε να δούμε το project της εφαρμογής στον Package Explorer του Eclipse στην αριστερή πλευρά της οθόνης.
2. **Εισαγωγή νέου server**: Από το κεντρικό μενού επιλέγουμε File → New. Στο μενού που εμφανίζεται μεταβαίνουμε στην κατηγορία Server και επιλέγουμε Server. Στη συνέχεια πατάμε Next. Στην κατηγορία Apache, επιλέγουμε τον Tomcat v7.0 Server (αφήνοντας τις υπόλοιπες επιλογές ως έχουν) και πατάμε Next. Τώρα, πρέπει να διαλέξουμε έναν φάκελο όπου θα αποθηκευτούν τα αρχεία του server. Αφού φτιάξουμε τον φάκελο πατάμε Download and Install ώστε να κατεβάσουμε τα αρχεία. Μόλις ολοκληρωθεί η διαδικασία πατάμε Finish.
3. **Φόρτωση της εφαρμογής στον server**: Στο σημείο αυτό έχει εισαχθεί ο apache tomcat στο eclipse και θα πρέπει να φορτώσουμε σε αυτόν την εφαρμογή. Στο κάτω

μέρος της οθόνης υπάρχει η καρτέλα Servers, όπου φαίνεται ο apache tomcat με την ένδειξη Stopped. Πατάμε δεξί κλικ πάνω του και από το μενού επιλέγουμε Add and Remove. Στην αριστερή στήλη του παραθύρου που εμφανίζεται θα πρέπει να φαίνεται το project της εφαρμογής. Το επιλέγουμε και στη συνέχεια πατάμε Add και μετά Finish.

4. **Έναρξη του server και εκτέλεση της εφαρμογής**: Η εφαρμογή πλέον έχει φορτωθεί στον server. Για να ξεκινήσει ο apache tomcat επιλέγουμε το εικονίδιο Run στο κάτω μέρος της οθόνης. Μόλις ο server ξεκινήσει πηγαίνουμε στα αρχεία του project, στο φάκελο WebContent. Στη συνέχεια επιλέγουμε το αρχείο index.html και πατάμε δεξί κλικ. Επιλέγουμε Run As → Run on Server και αναμένουμε ώστε να εμφανιστεί η αρχική σελίδα της εφαρμογής στον προεπιλεγμένο browser.

8

Έλεγχος

Στην ενότητα αυτή θα περιγραφεί ο έλεγχος της εφαρμογής βάσει ενός σεναρίου χρήσης που καλύπτει όλες τις λειτουργίες που μπορεί να κάνει ένας χρήστης. Το σενάριο με το οποίο θα γίνει ο έλεγχος αρχικά περιλαμβάνει την εισαγωγή διαφορετικών πηγών δεδομένων , τη λήψη δεδομένων από αυτές και τη δημιουργία συνόλων δεδομένων. Στη συνέχεια για κάθε dataset θα εφαρμοστεί η διαδικασία μετασχηματισμού του σχήματός του. Μόλις όλα τα datasets αποκτήσουν ένα ενιαίο σχήμα-λεξιλόγιο θα επιχειρήσουμε να συνδέσουμε μερικά από αυτά ώστε να εμπλουτίσουμε σημασιολογικά το χώρο δεδομένων που έχει σχηματιστεί. Τέλος, θα εφαρμόσουμε SPARQL ερωτήματα στο dataspace ώστε να λάβουμε ολοκληρωμένες απαντήσεις.

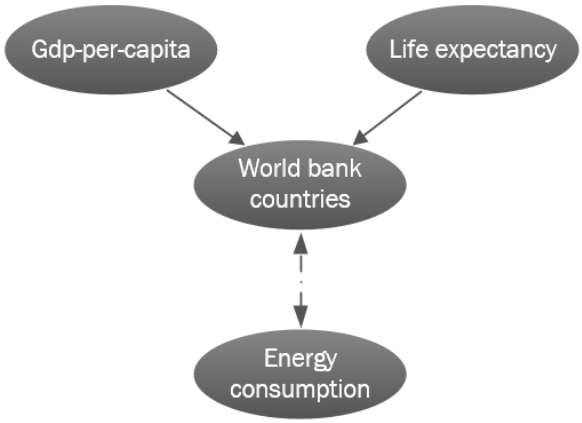
8.1 Μεθοδολογία ελέγχου

Το σενάριο αφορά τον συνδυασμό στατιστικών δεικτών από μια επιχείρηση που δραστηριοποιείται στο χώρο της υγείας. Στη βάση δεδομένων της υπάρχουν στατιστικά στοιχεία σχετικά με το προσδόκιμο ζωής στις χώρες του κόσμου για κάποια έτη. Η επιχείρηση επιθυμεί να συνδυάσει τα δεδομένα αυτά με στατιστικά στοιχεία της Παγκόσμιας Τράπεζας που αφορούν το κατά κεφαλήν εισόδημα και την κατανάλωση ενέργειας ανά χώρα. Για να επιτευχθεί ο στόχος χρειάζεται να εισαχθούν τα κατάλληλα δεδομένα που είναι:

- Τοπικό αρχείο με την κατανάλωση ενέργειας ανά χώρα στη μορφή RDF. Το αρχείο αυτό ελήφθη από την τράπεζα RDF δεδομένων της Παγκόσμιας Τράπεζας

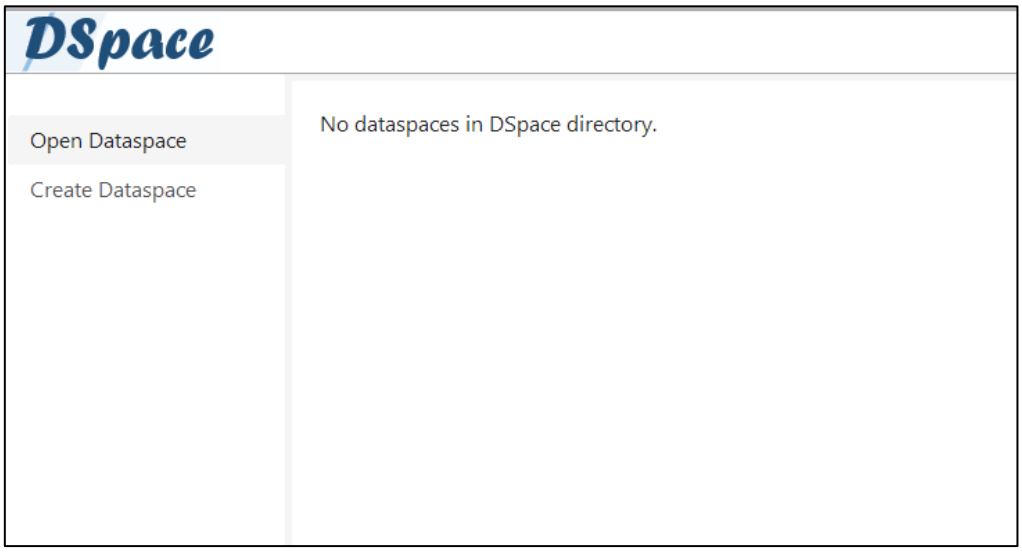
- Τοπικό αρχείο με το κατά κεφαλήν εισόδημα ανά χώρα που είναι σε excel μορφή.
- Δεδομένα που περιλαμβάνουν το προσδόκιμο ζωής ανά χώρα μέσω SQL ερωτήματος στη βάση δεδομένων της επιχείρησης
- Δεδομένα που περιλαμβάνουν πληροφορίες για τις χώρες (όπως τα URIs και τα ονόματά τους) μέσω του SPARQL Endpoint της Παγκόσμιας Τράπεζας

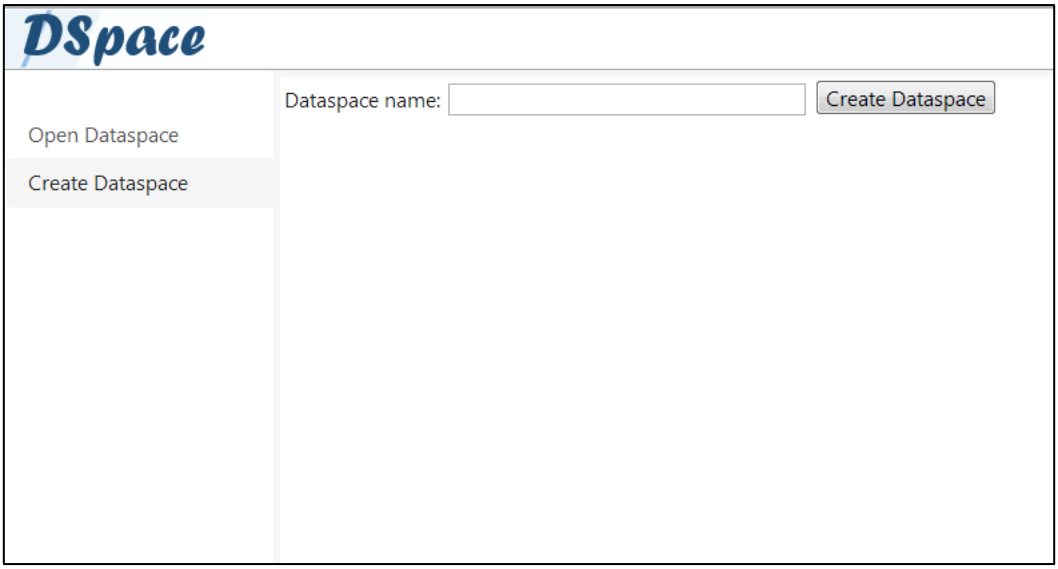
Σχηματικά, ο συνδυασμός των δεδομένων φαίνεται στην παρακάτω εικόνα. Τα συνεχόμενα βέλη δείχνουν τις συνδέσεις που χρειάζεται να γίνουν. Το διακεκομμένο βέλος δείχνει υπάρχουσα σύνδεση μέσω των URIs που προσδιορίζουν τις χώρες. Στην επόμενη ενότητα αναλύουμε τα βήματα εισαγωγής των πηγών, της λήψης δεδομένων από αυτές και τις ενέργειες επί των δεδομένων που πραγματοποιούνται για να σχηματιστεί ο χώρος δεδομένων.



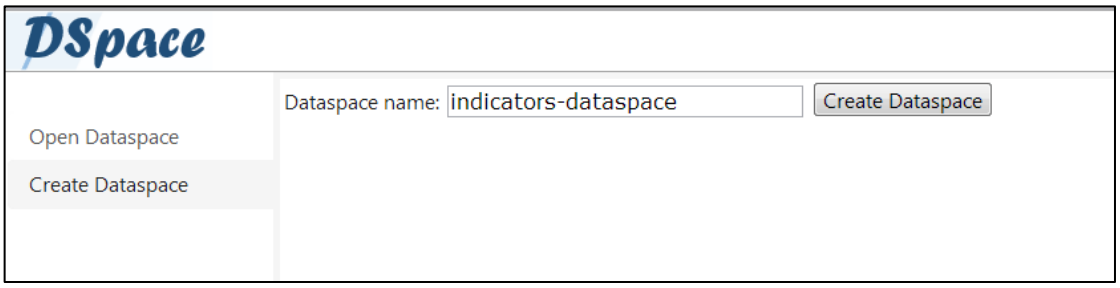
8.2 *Αναλυτική παρουσίαση ελέγχου*

Καθώς ξεκινά η εφαρμογή καλούμαστε να επιλέξουμε αν θα ανοίξουμε έναν αποθηκευμένο χώρο δεδομένων για την εκτέλεση sparql ερωτημάτων ή αν θα δημιουργήσουμε ένα νέο χώρο δεδομένων, όπως φαίνεται στις παρακάτω εικόνες:

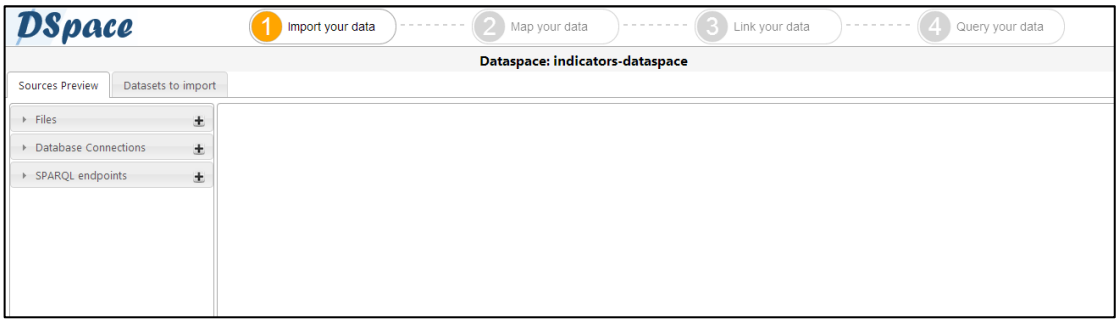




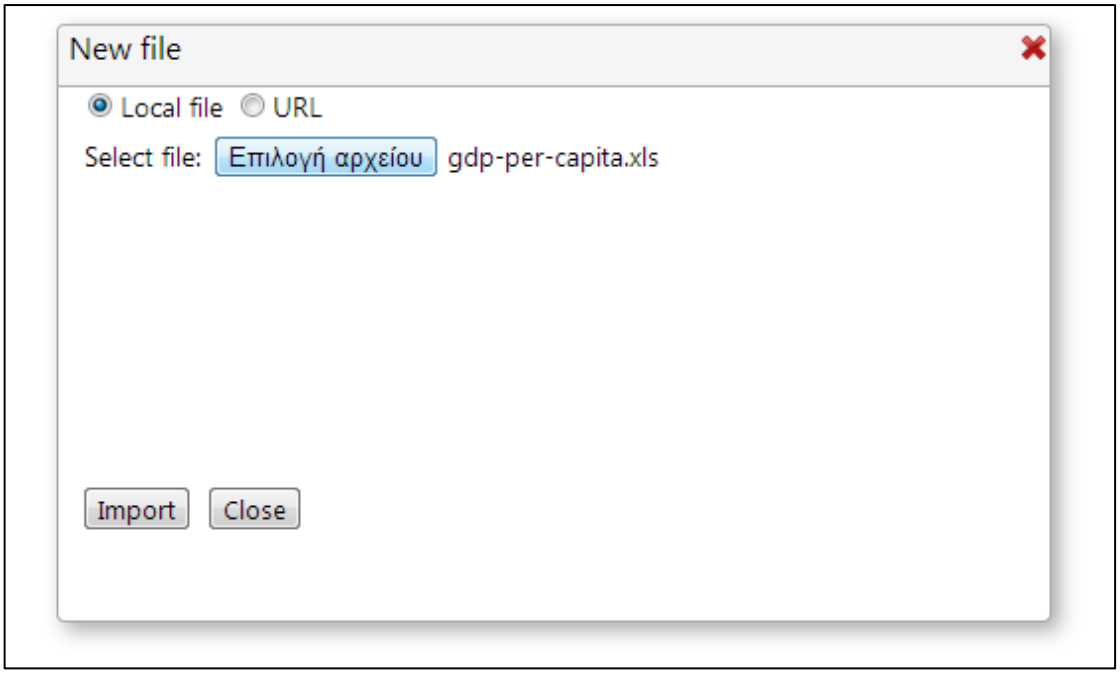
Θα δημιουργήσουμε έναν νέο χώρο δεδομένων με το όνομα *indicators-dataspace*, όπως φαίνεται παρακάτω:



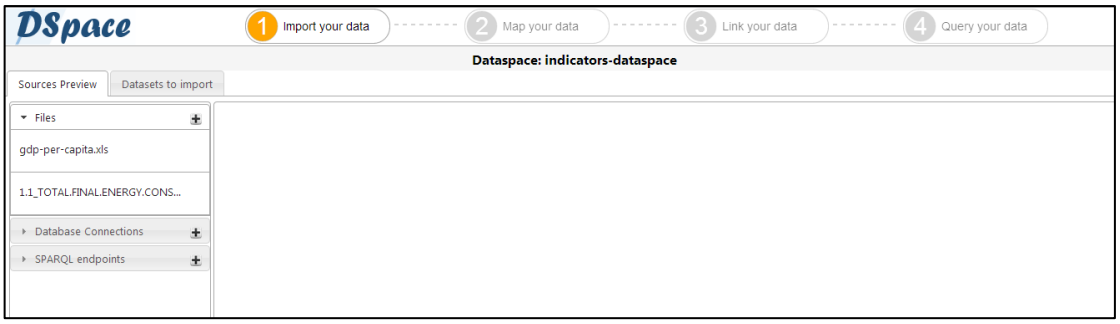
Στο σημείο αυτό πατάμε *Create Dataspace* ώστε να δημιουργηθεί ο χώρος δεδομένων. Στη συνέχεια ξεκινά η διαδικασία σχηματισμού του dataspace με πρώτο βήμα την εισαγωγή δεδομένων, όπως δείχνει η παρακάτω εικόνα:



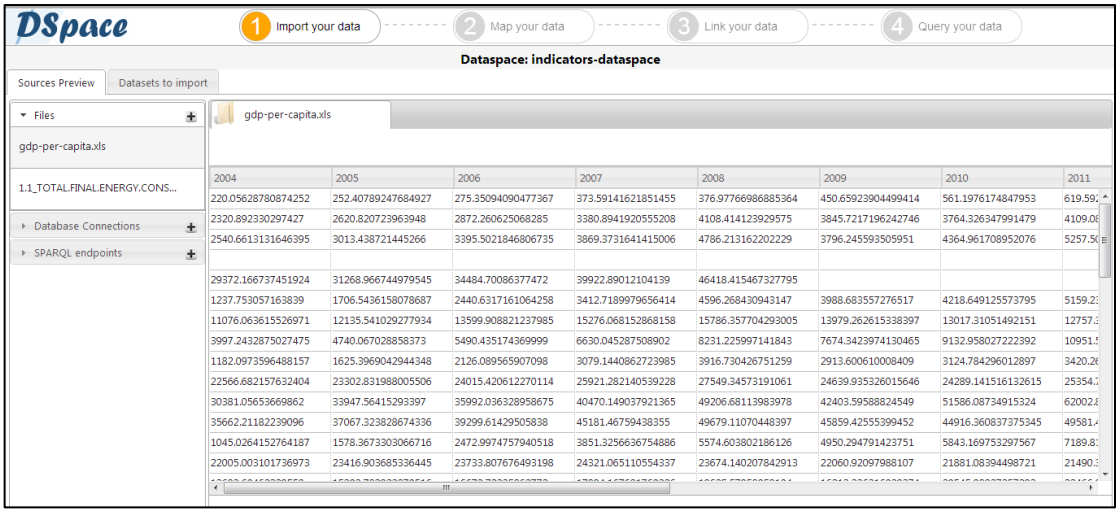
Στην αριστερή στήλη ο χρήστης μπορεί να εισάγει αρχεία, συνδέσεις σε βάση δεδομένων καθώς και SPARQL Endpoints πατώντας τα αντίστοιχα εικονίδια . Αρχικά, εισάγουμε τα αρχεία που περιλαμβάνουν την κατανάλωση ενέργειας (rdf αρχείο) και το κατά κεφαλήν εισόδημα (excel αρχείο) όπως δείχνουν οι εικόνες που ακολουθούν:

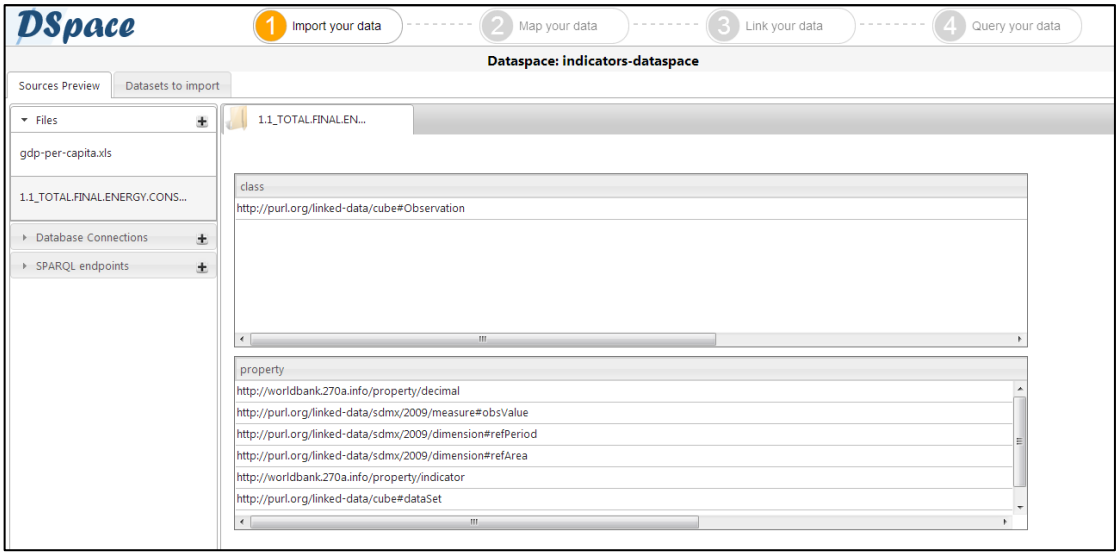


Πατώντας *Import* φορτώνεται το αρχείο στην εφαρμογή. Με τον ίδιο τρόπο εισάγουμε και το rdf αρχείο. Μπορούμε να δούμε τα δυο αρχεία στην αριστερή στήλη σύμφωνα με την ακόλουθη εικόνα:

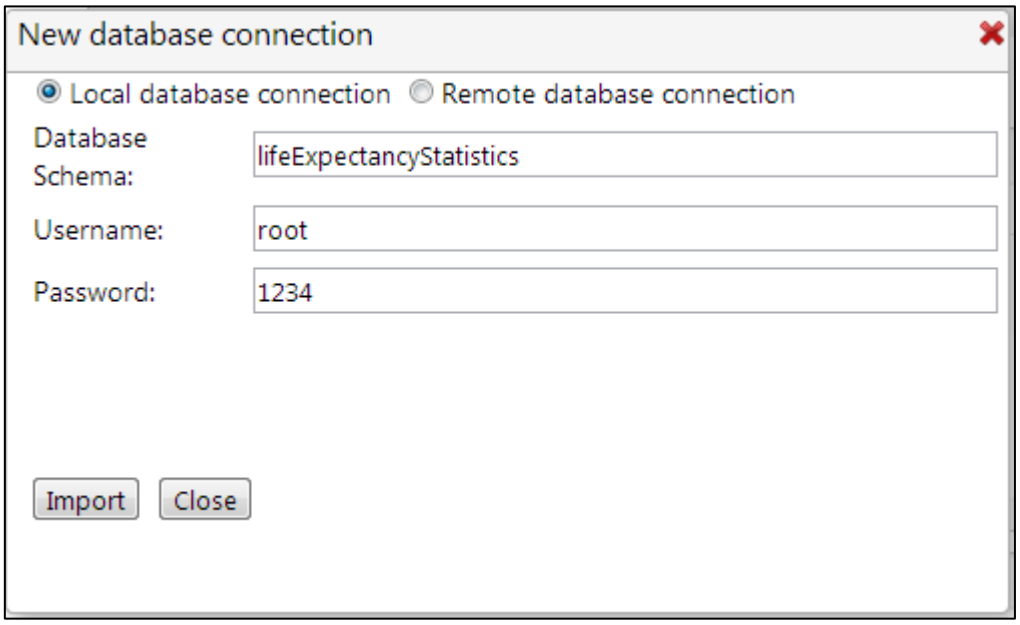


Πατώντας πάνω στα αρχεία μπορούμε να δούμε είτε τα δεδομένα τους (στην περίπτωση των csv ή xls αρχείων) είτε το λεξιλόγιό τους (στην περίπτωση των rdf αρχείων) όπως παρακάτω:

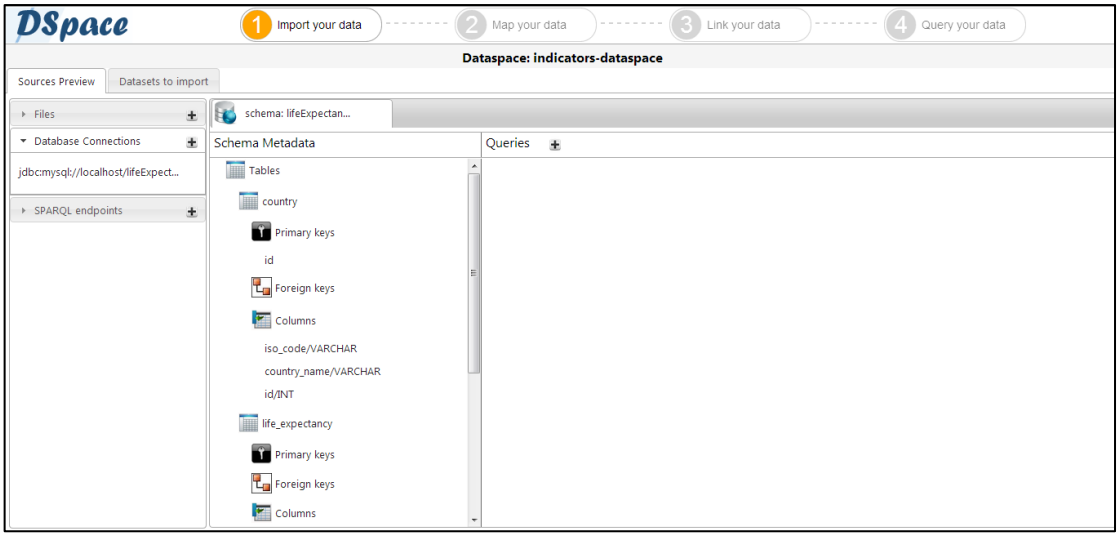




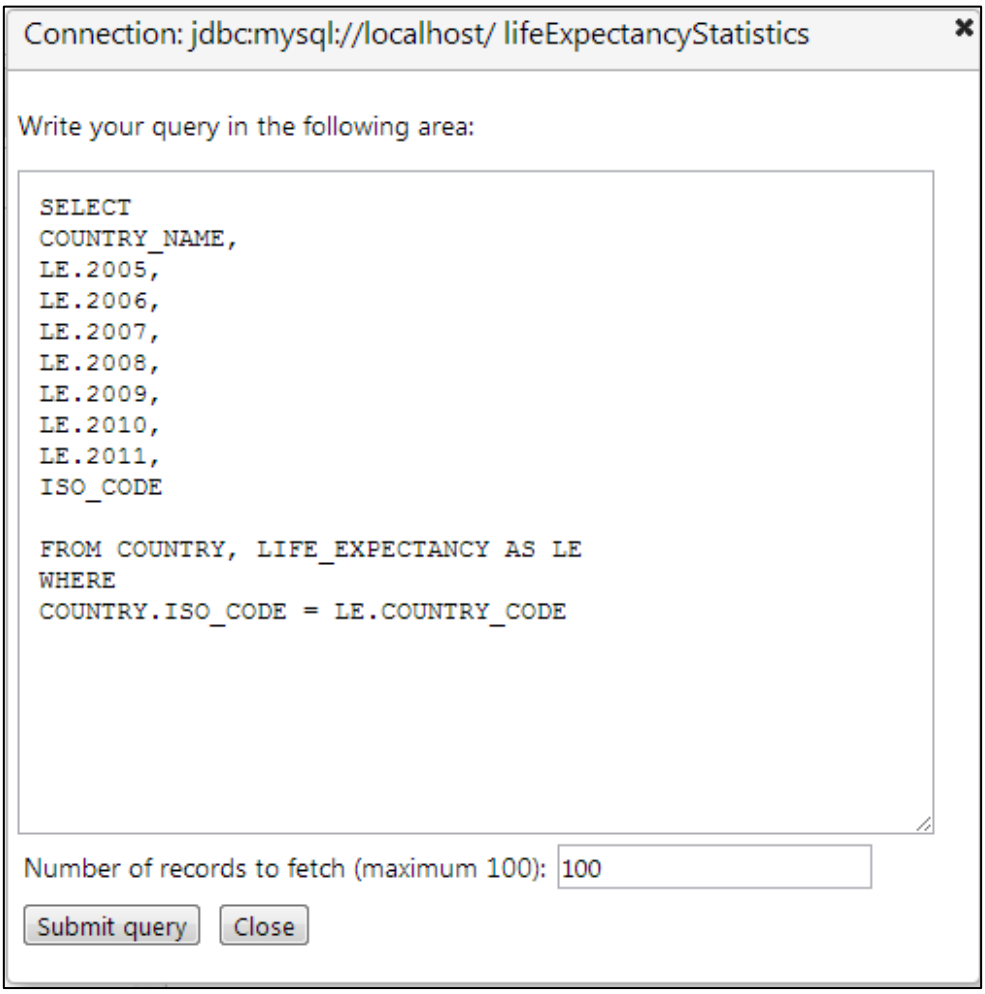
Τώρα μπορούμε να εισάγουμε το σχήμα *lifeExpectancyStatistics* από την τοπική βάση δεδομένων όπως φαίνεται παρακάτω:



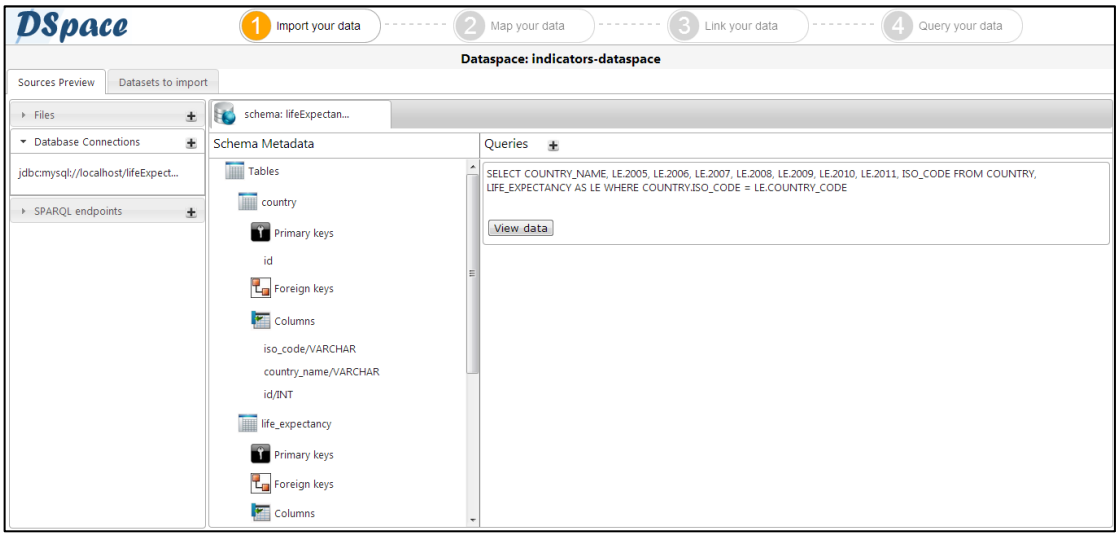
Πατώντας *Import* εισάγουμε το σχήμα και βλέπουμε τα μετα-δεδομένα του (ονόματα και τύπους στηλών, πρωτεύον κλειδί, ξένα κλειδιά) σύμφωνα με την ακόλουθη εικόνα:



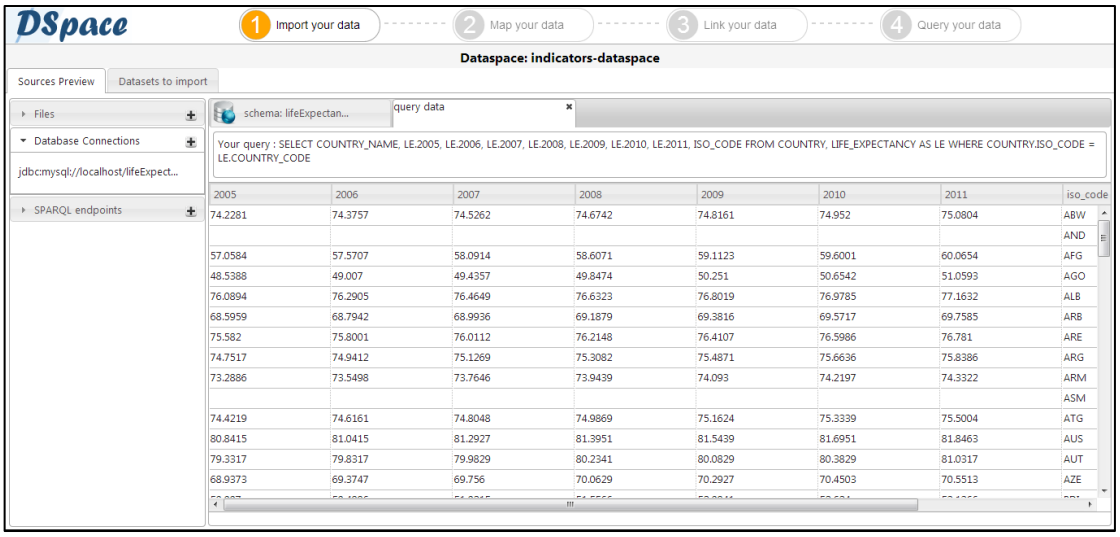
Πατώντας στο εικονίδιο  είναι δυνατή η συγγραφή ενός sql ερωτήματος. Στην προκειμένη περίπτωση το ερώτημα περιλαμβάνει τη συλλογή των ονομάτων των χωρών και των στηλών που δείχνουν το προσδόκιμο ζωής για τα έτη 2005-2011. Ακόμη, επιλέγεται ο αριθμός των εγγραφών που θα παρουσιαστούν στην οθόνη. Πατώντας *Submit* εκτελείται το ερώτημα στη βάση δεδομένων. Η παρακάτω εικόνα δείχνει την εισαγωγή του ερωτήματος:



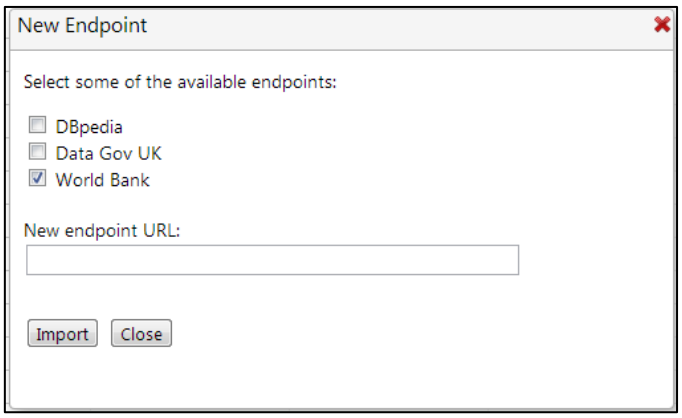
Εφόσον το ερώτημα εκτελεστεί σωστά μπορούμε να το δούμε στο κεντρικό πάνελ της εφαρμογής:



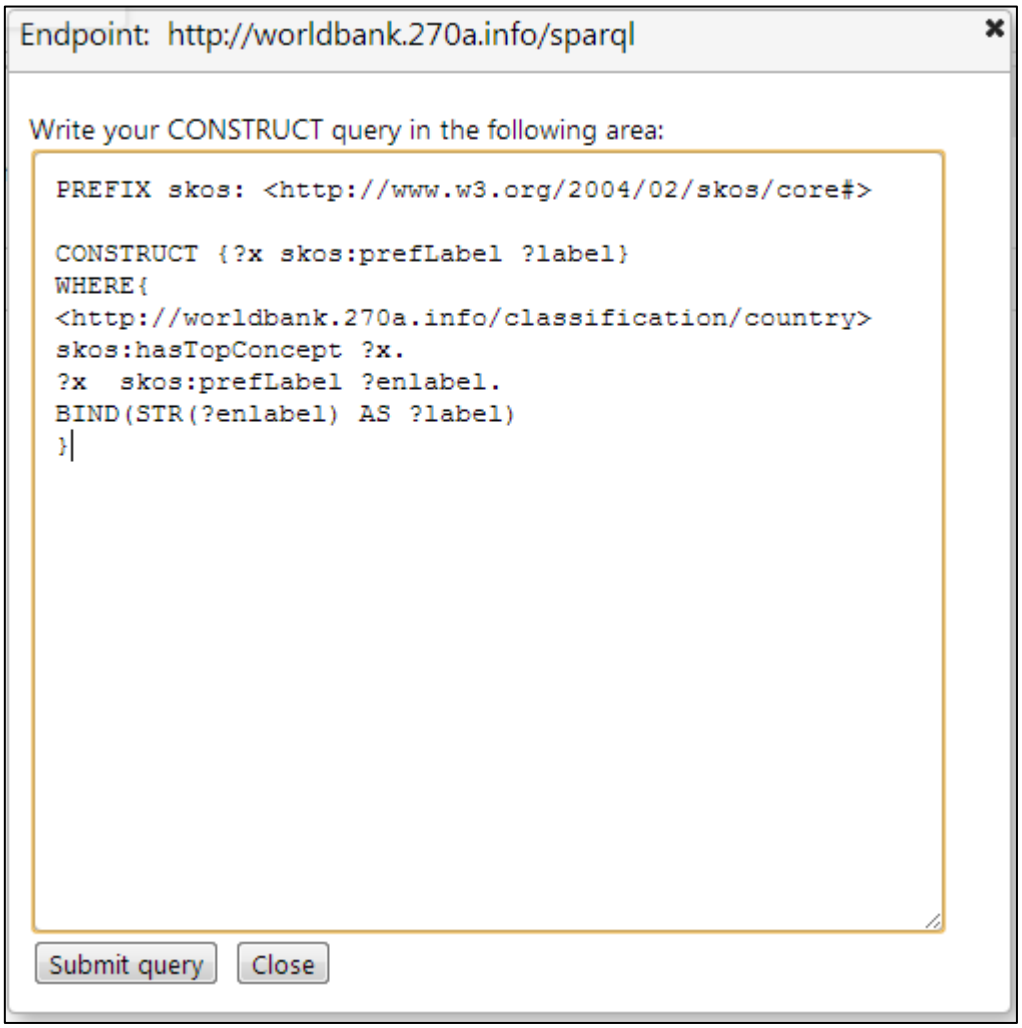
Πατώντας στο πλήκτρο *View Data* μπορούμε να δούμε τα δεδομένα του ερωτήματος:



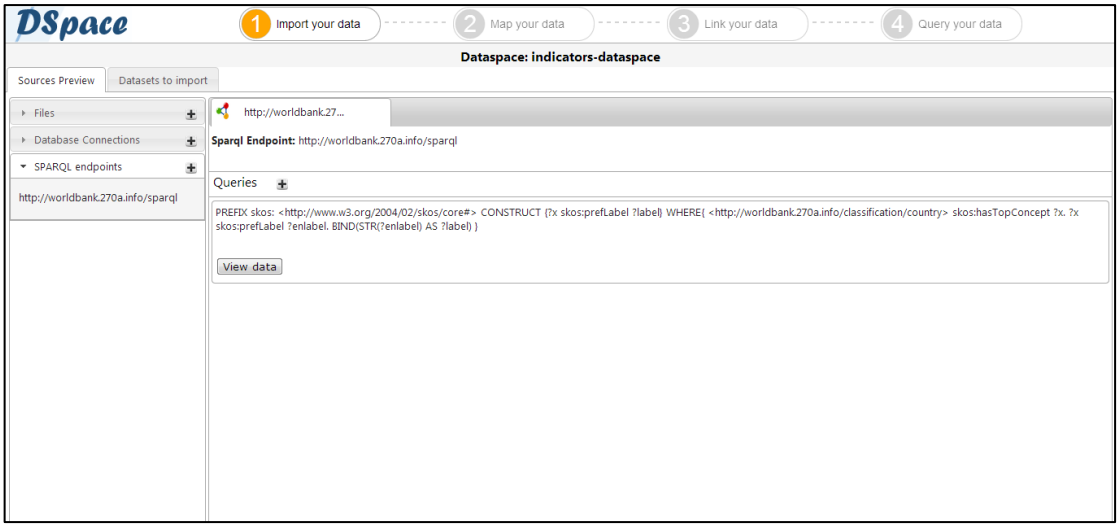
Τελευταία ενέργεια είναι η εισαγωγή του sparql endpoint της Παγκόσμιας Τράπεζας και η εκτέλεση ενός ερωτήματος σε αυτό για να ληφθούν τα URIs και τα ονόματα των χωρών. Το endpoint εισάγεται με το παρακάτω μενού:

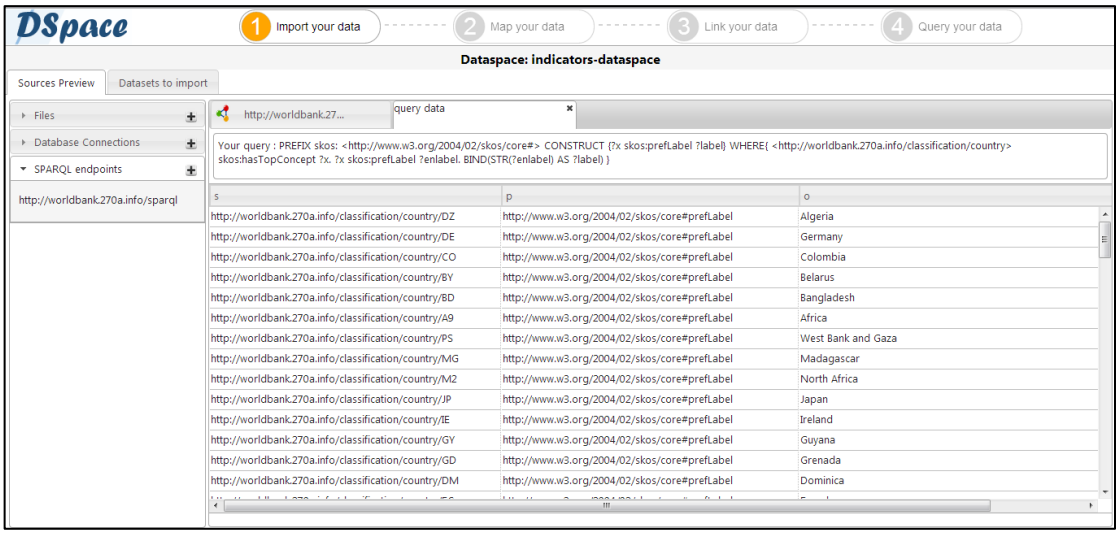


Πατώντας *Import* εισάγεται το endpoint στο σύστημα και μπορούν να εκτελεστούν ερωτήματα σε αυτό μέσω του παραθύρου που φαίνεται στην ακόλουθη εικόνα. Το ερώτημα είναι τύπου CONSTRUCT προκειμένου να κατασκευαστεί ένα τοπικό αρχείο με τα ζητούμενα δεδομένα. Πατώντας *Submit* εκτελείται το ερώτημα στο endpoint.

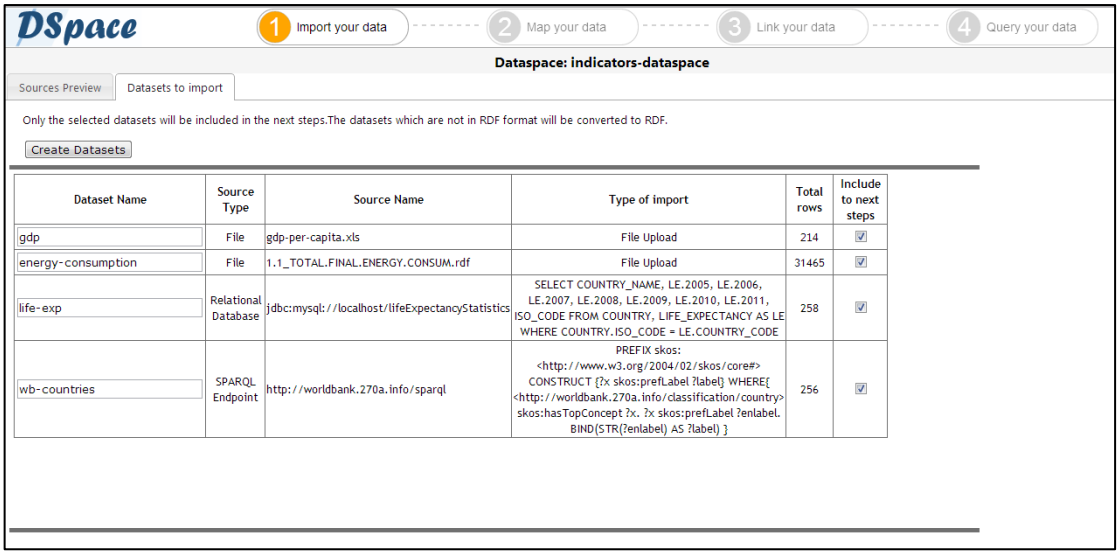


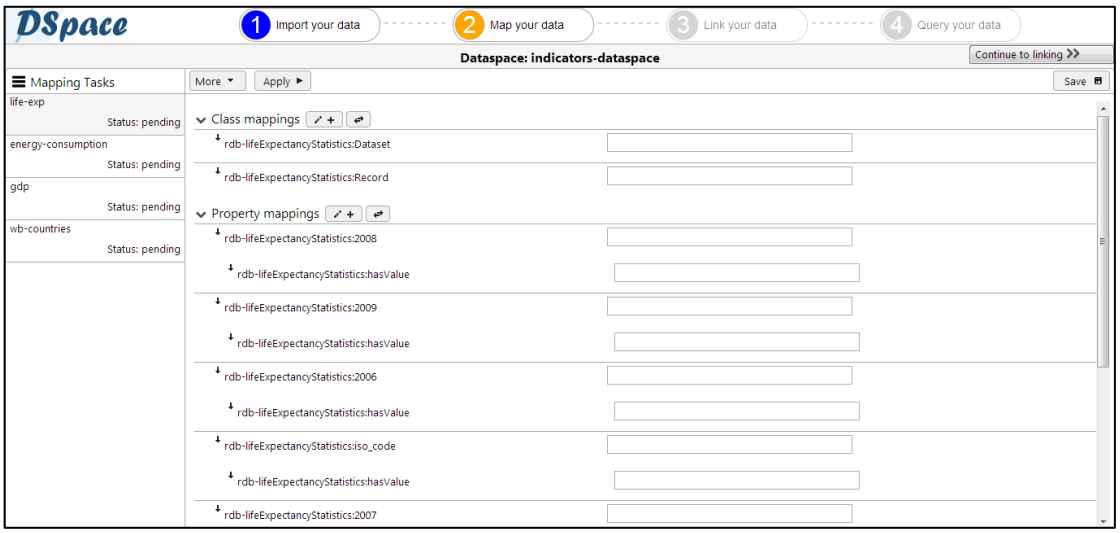
Ο χρήστης μπορεί να δει το αποτέλεσμα του ερωτήματος πατώντας *View Data* στο κεντρικό πάνελ όπως δείχνουν οι επόμενες εικόνες:





Η κεντρική καρτέλα με την ένδειξη *Datasets to Import* παρουσιάζει τα σύνολα δεδομένων που μπορούν να δημιουργηθούν βάσει των δεδομένων που έχουν ληφθεί από τις πηγές. Κάθε γραμμή του πίνακα δείχνει το αντίστοιχο dataset που μπορεί να σχηματιστεί καθώς και πληροφορίες σχετικά με την πηγή του. Ο χρήστης επιλέγει όσα datasets θέλει να δημιουργηθούν αποδίδοντας ένα όνομα στο καθένα. Πατώντας το κουμπί *Create Datasets* αρχίζει η διαδικασία σχηματισμού. Όταν σχηματιστούν τα datasets ο χρήστης μεταβαίνει αυτόματα στο δεύτερο βήμα της εφαρμογής. Η παραπάνω διαδικασία φαίνεται στις παρακάτω εικόνες:

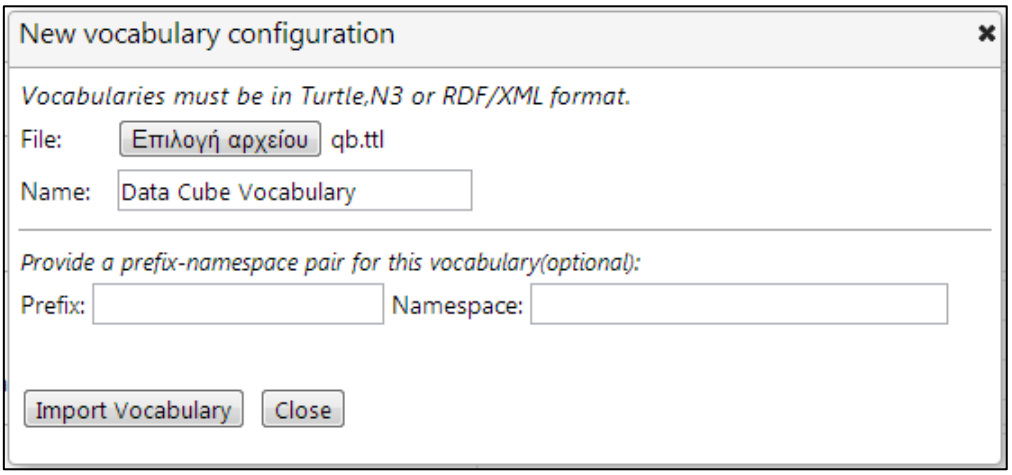




Στην αριστερή στήλη της οθόνης φαίνονται τα mapping tasks που πρέπει να γίνουν για κάθε dataset. Κάθε mapping task περιέχει αντιστοιχίσεις όρων και μπορεί να εκτελεστεί όσες φορές επιθυμεί ο χρήστης. Οι αντιστοιχίσεις αυτές παρουσιάζονται με τη μορφή δυο καταλόγων και δυο στηλών ανά κατάλογο. Ο πρώτος κατάλογος περιλαμβάνει τις κλάσεις και ο δεύτερος τις ιδιότητες του τρέχοντος σχήματος του dataset. Οι όροι αυτοί βρίσκονται στην αριστερή στήλη κάθε καταλόγου. Στη δεξιά στήλη ο χρήστης μπορεί να πληκτρολογήσει το όνομα του όρου στον οποίο θέλει να μετασχηματιστεί ο αντίστοιχος όρος της αριστερής στήλης. Ο χρήστης μπορεί να φτιάξει mappings για όσους όρους επιθυμεί με τη γνώση ότι μόνο αυτοί θα μεταφραστούν. Όσοι όροι δεν έχουν mapping δεν θα μεταφερθούν στο νέο σχήμα.

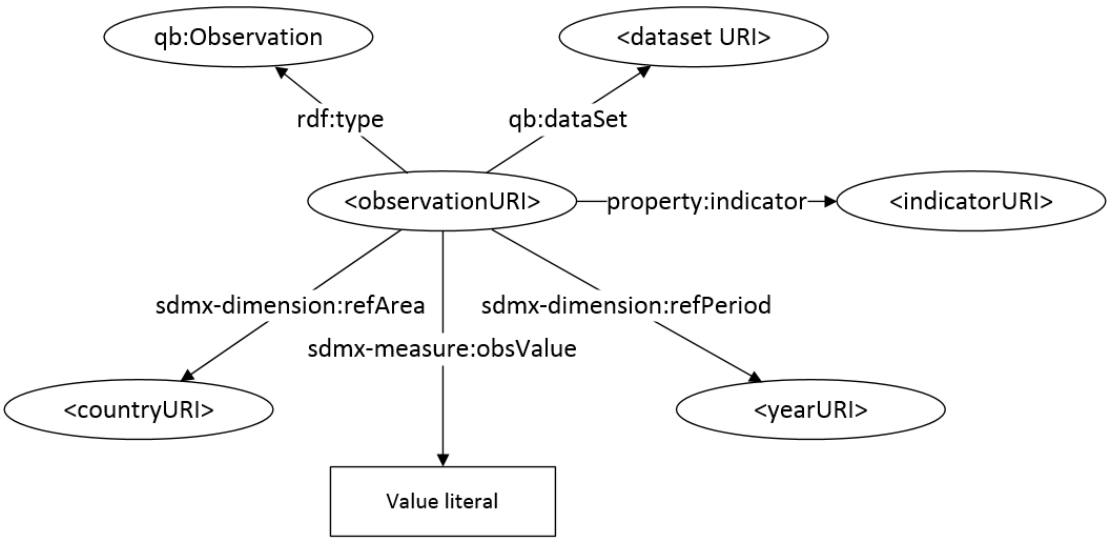
Ακόμη, υπάρχει δυνατότητα εισαγωγής αρχείων που περιέχουν λεξιλόγια πατώντας το πλήκτρο *More* και στη συνέχεια *Add new vocabulary*, όπως φαίνεται στην εικόνα που

ακολουθεί. Για τις ανάγκες του σεναρίου εισάγονται τα λεξιλόγια Data Cube Vocabulary, SDMX-Dimension Vocabulary και SDMX-Measure Vocabulary.

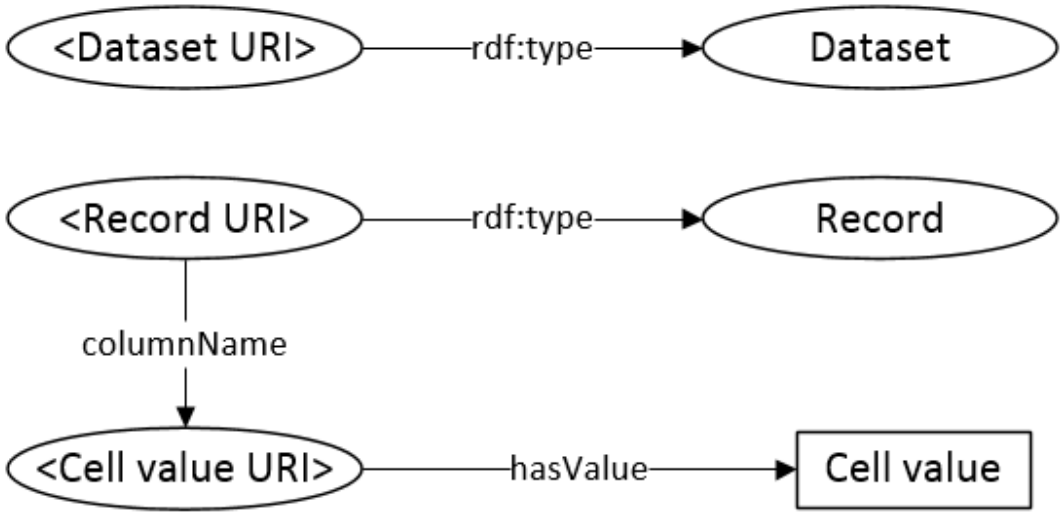


Σε περίπτωση που το ζεύγος prefix-namespace για το αντίστοιχο λεξιλόγιο δεν υπάρχει στο αρχείο που εισάγεται μπορεί να εισαχθεί με τη χρήση του παραπάνω μενού. Πατώντας *Import Vocabulary* εισάγεται το λεξιλόγιο στο σύστημα. Επαναλαμβάνουμε την ίδια διαδικασία και για τα άλλα δυο λεξιλόγια.

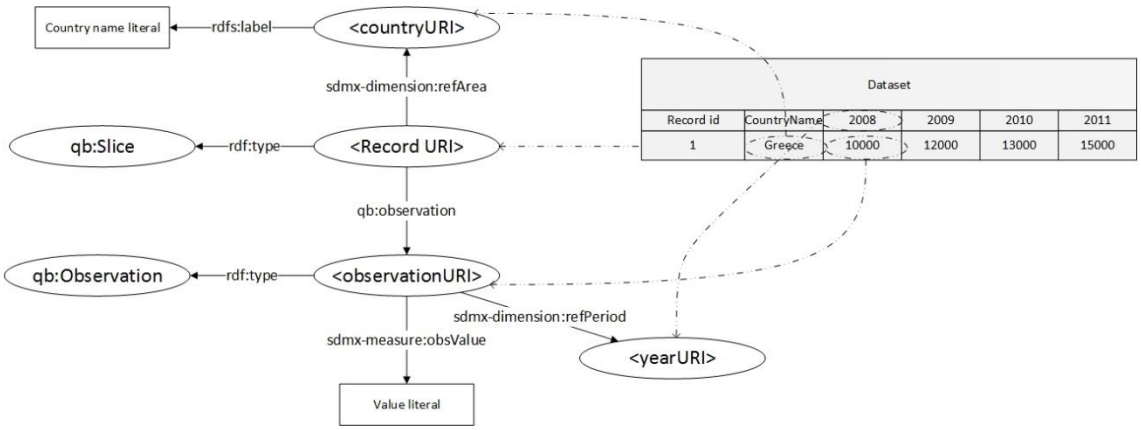
Στο σημείο αυτό χρειάζεται να ορίσουμε τις αντιστοιχίσεις ανάμεσα στους όρους των λεξιλογίων για κάθε dataset. Το σχήμα που χαρακτηρίζει το dataset με τα δεδομένα της κατανάλωσης ενέργειας (energy-consumption) είναι το παρακάτω:



Όπως έχει προαναφερθεί και σε προηγούμενο κεφάλαιο το σχήμα που περιγράφει τα σύνολα δεδομένων life-exp (προερχόμενο από την τοπική βάση δεδομένων) και gdp (προερχόμενο από το αρχείο xls) είναι:



Στόχος είναι λοιπόν να μετατρέψουμε το παραπάνω σχήμα σε ένα νέο που να είναι συμβατό με εκείνο της κατανάλωσης ενέργειας και ταυτόχρονα να είναι στατιστικά ορθό:



Τα datasets life-exp και gdp αποτελούν χρονοσειρές . Τα δεδομένα από όπου προέρχονται τα datasets είναι οργανωμένα σε πίνακες. Κάθε έτος αντιστοιχεί σε μία στήλη και κάθε κελί αντιστοιχεί στο μετρούμενο μέγεθος για την αντίστοιχη χώρα και το αντίστοιχο έτος. Κάθε γραμμή του πίνακα είναι μια «στατιστική φέτα»[26] με βάση την αντίστοιχη χώρα. Η αναλογία αυτή φαίνεται από τις σχέσεις <RecordURI> rdf:type qb:Slice και <RecordURI> sdmx-dimension:refArea <countryURI>, όπου <RecordURI> και <countryURI> οι τριπλέτες που έχουν προκύψει από τον αλγόριθμο μετατροπής για τη γραμμή και το κελί αντίστοιχα. Τώρα, κάθε γραμμή του πίνακα συνδέεται με τα κελιά μέσω των στηλών. Κάθε κελί στην ουσία

[26] *Slice:όρος που χρησιμοποιείται για την ομαδοποίηση στατιστικών παρατηρήσεων κρατώντας μία ή περισσότερες διαστάσεις σταθερή/ές*

είναι μια qb:Observation και αυτό δηλώνουν οι σχέσεις rdf:type και qb:observation. Κάθε «φέτα» έχει κάποιες παρατηρήσεις και όλες οι φέτες μαζί συνιστούν το σύνολο των παρατηρήσεων του dataset. Το <observationURI> είναι το URI κάθε κελιού όπως αυτό έχει προκύψει από τον αλγόριθμο μετατροπής. Για κάθε κελί-παρατήρηση χρειαζόμαστε πρώτον την τιμή της παρατήρησης και δεύτερον τη διάσταση «χρόνο» που δείχνει το πότε έγινε η παρατήρηση. Αυτές οι πληροφορίες φαίνονται από τις ιδιότητες sdmx-measure:obsValue (μετρούμενη τιμή) και sdmx-dimension:refPeriod (χρονολογία). Το παραπάνω σχήμα μπορεί να επιτευχθεί εισάγοντας τρεις αντιστοιχίσεις που φαίνονται παρακάτω:

α) Custom mapping:

Source Pattern	`?SUBJ prefix:2010 ?cellValueUri.` `?cellValueUri prefix:hasValue ?cellValue.`

Target Pattern	`?SUBJ qb:observation ?cellValueUri.`
	`?cellValueUri sdmx-dimension:refPeriod <http://reference.data.gov.uk/id/year/2010>.`
	`?cellValueUri rdf:type qb:Observation.`
	`?cellValueUri sdmx-measure:obsValue ?cellValue.`

β) Mappings με τη βοήθεια του γραφικού περιβάλλοντος της εφαρμογής

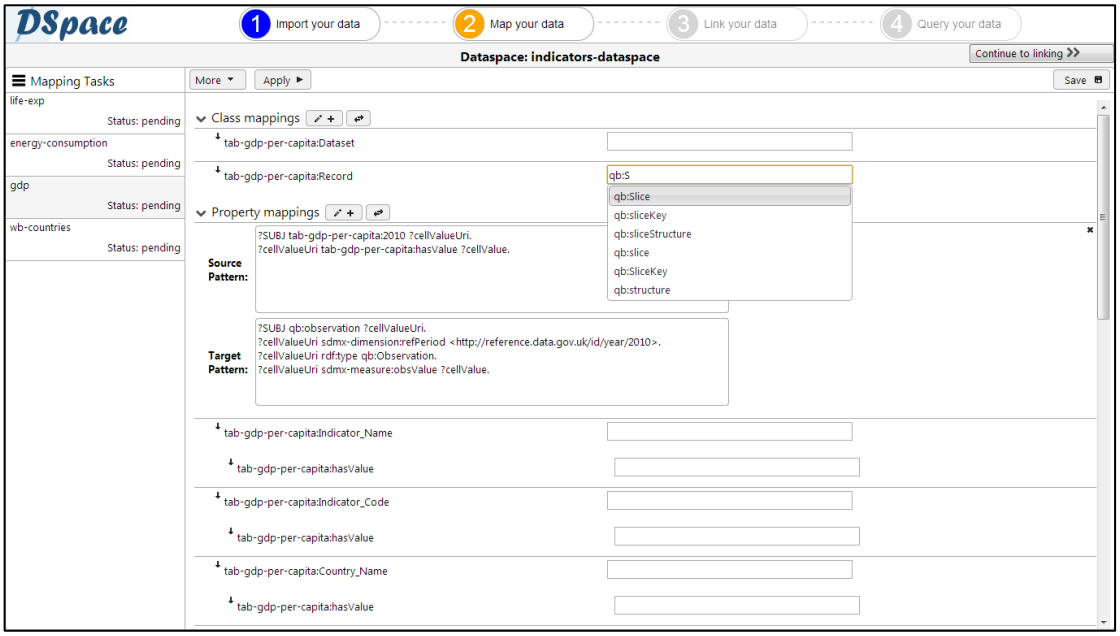
Source Pattern	`?SUBJ prefix:Country_Name ?cellValueUri.` `?cellValueUri prefix:hasValue ?cellValue.`
Target Pattern	`?SUBJ sdmx-dimension:refArea ?cellValueUri.` `?cellValueUri rdfs:label ?cellValue.`

Source Pattern	`?SUBJ rdf:type prefix:Record.`
Target Pattern	`?SUBJ rdf:type qb:Slice`

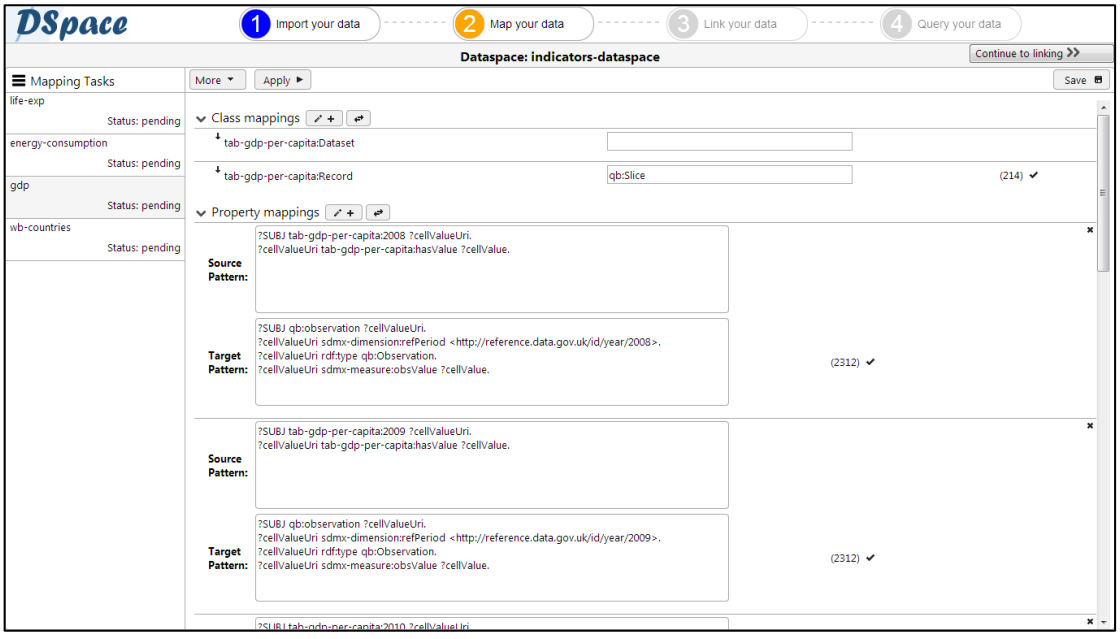
Η χρονολογία είναι ενδεικτική. Ωστόσο, βάσει του σχήματος χρειάζεται να γίνουν τόσα custom mappings όσα και οι χρονολογίες-στήλες επιθυμεί ο χρήστης. Για τις ανάγκες του παραδείγματος έγιναν τρεις αντιστοιχίσεις για τις χρονιές 2008-2010.

Παρακάτω παρουσιάζεται η εισαγωγή των mappings για το σύνολο δεδομένων gdp. Αντίστοιχη διαδικασία εφαρμόζεται για το σύνολο δεδομένων life-exp. Η εισαγωγή προσαρμοσμένης (custom) αντιστοίχισης γίνεται πατώντας το εικονίδιο  :

Τώρα, μπορούμε να εισάγουμε τα άλλα δυο mappings με τη βοήθεια του γραφικού περιβάλλοντος, όπως δείχνει η ακόλουθη εικόνα:

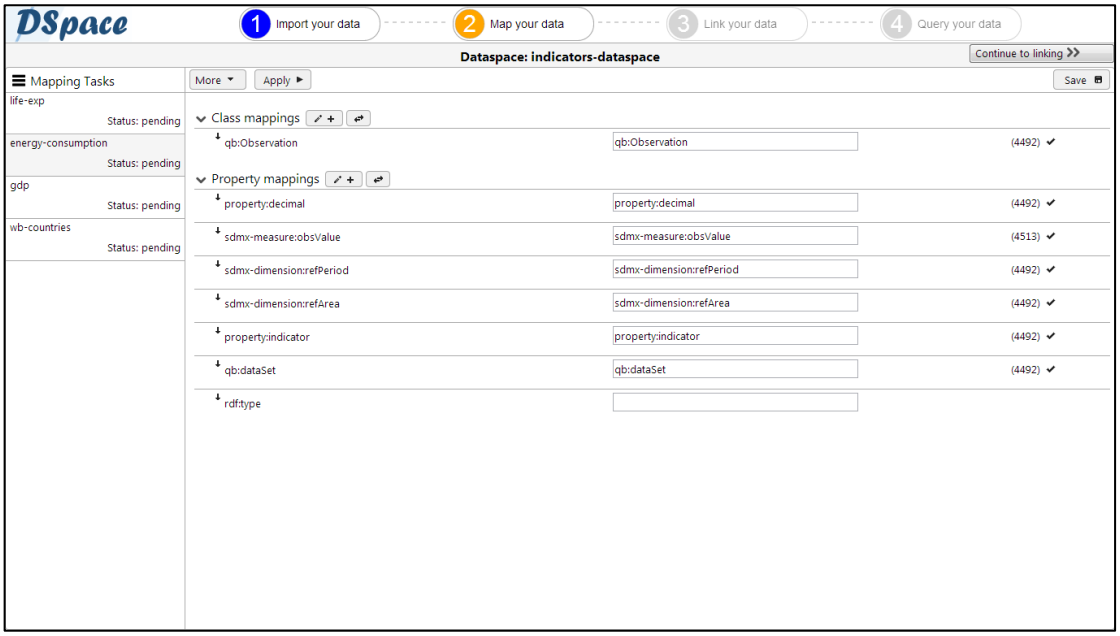


Μόλις συμπληρωθούν όλα τα mappings ο χρήστης μπορεί να πατήσει *Apply* ώστε να εφαρμοστούν στο αντίστοιχο dataset. Η εικόνα που βλέπει ο χρήστης μετά την εφαρμογή των mappings είναι η ακόλουθη:



Δίπλα σε κάθε mapping εμφανίζεται ένας αριθμός που δείχνει το πλήθος των τριπλετών που περιλαμβάνουν τις νέες ιδιότητες κάθε mapping. Με τον τρόπο αυτό δηλώνεται η επιτυχία κάθε αντιστοίχισης. Πατώντας *Save* ο χρήστης οριστικοποιεί το νέο σχήμα του dataset το οποίο είναι πλέον μόνιμο και δεν μπορεί να αλλάξει.

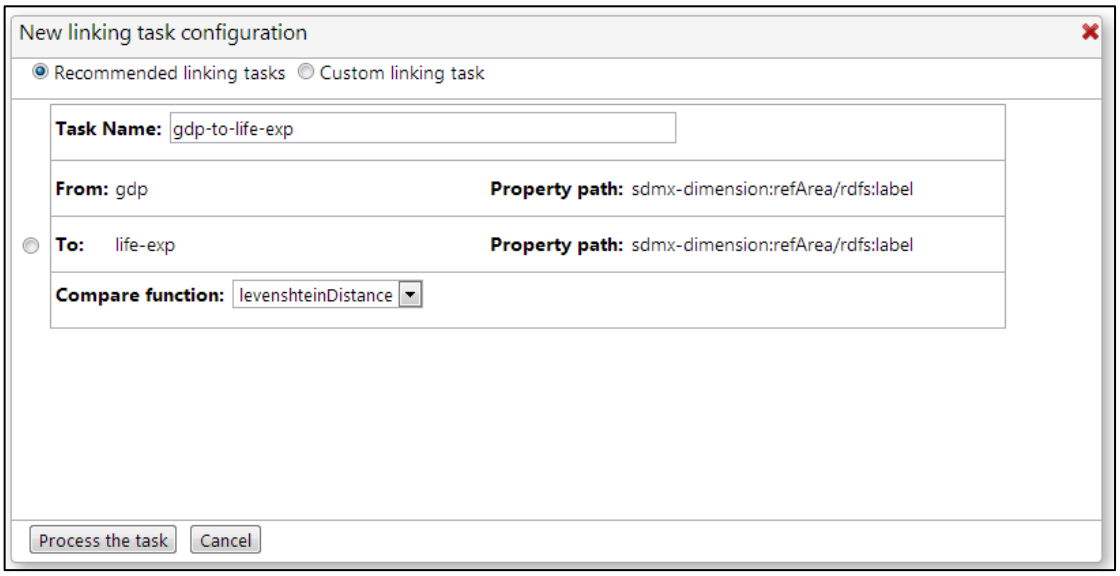
Το σύνολο δεδομένων energy-consumption δεν είναι ανάγκη να μετασχηματιστεί καθώς περιγράφεται από γνωστά λεξιλόγια (Data Cube, SDMX-Dimension, SDMX-Measure). Επομένως, το νέο σχήμα θα είναι ίδιο με τα παλιό. Για το λόγο αυτό χρησιμοποιούμε το εικονίδιο  ώστε οι όροι της αριστερής στήλης να μεταφερθούν στη δεξιά ως έχουν. Η διαδικασία αυτή φαίνεται στην παρακάτω εικόνα:



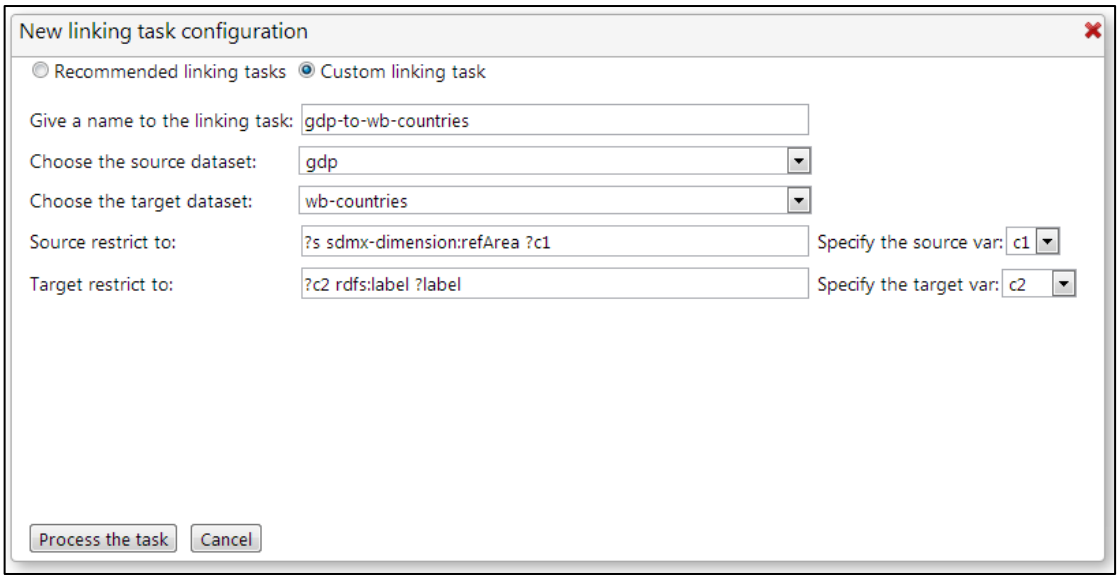
Τέλος, για το σύνολο δεδομένων wb-countries εφαρμόζουμε την απλή αντιστοιχία *skos:prefLabel → rdfs:label* . Πλέον, εφόσον έχει ολοκληρωθεί ο μετασχηματισμός όλων των συνόλων δεδομένων, είναι δυνατή η μετάβαση στο επόμενο βήμα πατώντας το πλήκτρο *Continue to linking.*

Στην κεντρική οθόνη του τρίτου βήματος ο χρήστης μπορεί να εισάγει ένα νέο linking task, πατώντας το εικονίδιο ✚ . Τότε εμφανίζεται το παράθυρο εισαγωγής ενός task. Ο χρήστης μπορεί να δει τα προτεινόμενα από το σύστημα tasks ή να ορίσει ένα δικό του. Στην παρακάτω εικόνα φαίνεται το προτεινόμενο task που προτρέπει το χρήστη να συνδέσει τα σύνολα δεδομένων life-exp και gdp βάσει των property paths *sdmx-dimension:refArea/rdfs:label*. Η σύνδεση αυτή δεν εξυπηρετεί τις ανάγκες του παραδείγματος. Επομένως, θα δημιουργηθούν δυο νέα παρόμοια linking tasks για τη σύνδεση των συνόλων δεδομένων gdp και life-exp με το σύνολο δεδομένων wb-countries βάσει των

URIs των χωρών. Η περιγραφή που ακολουθεί αφορά τη σύνδεση του πρώτου ζεύγους συνόλων δεδομένων.



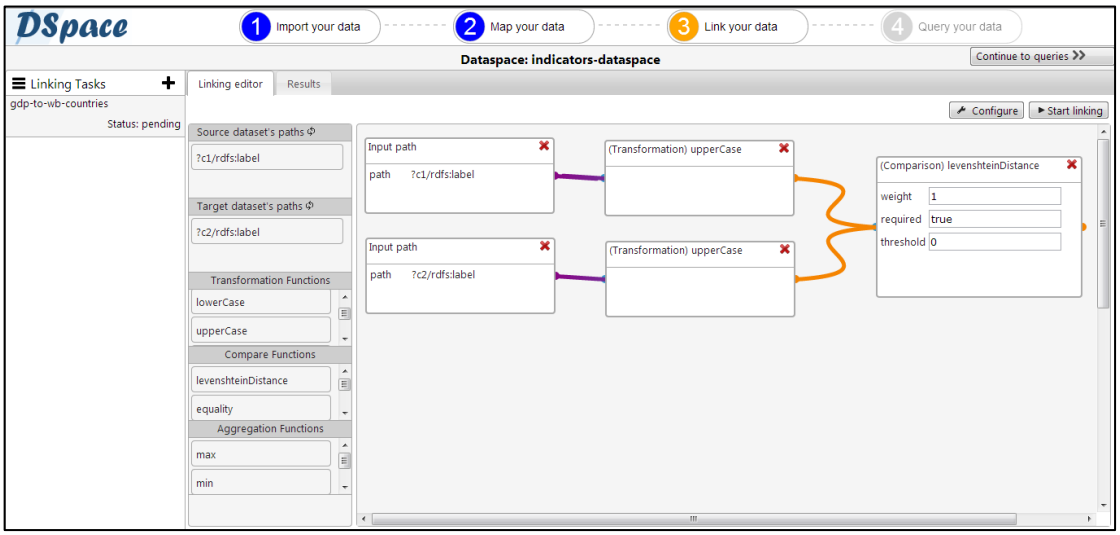
Για την εισαγωγή του task χρειάζονται κάποιες ρυθμίσεις που φαίνονται παρακάτω:



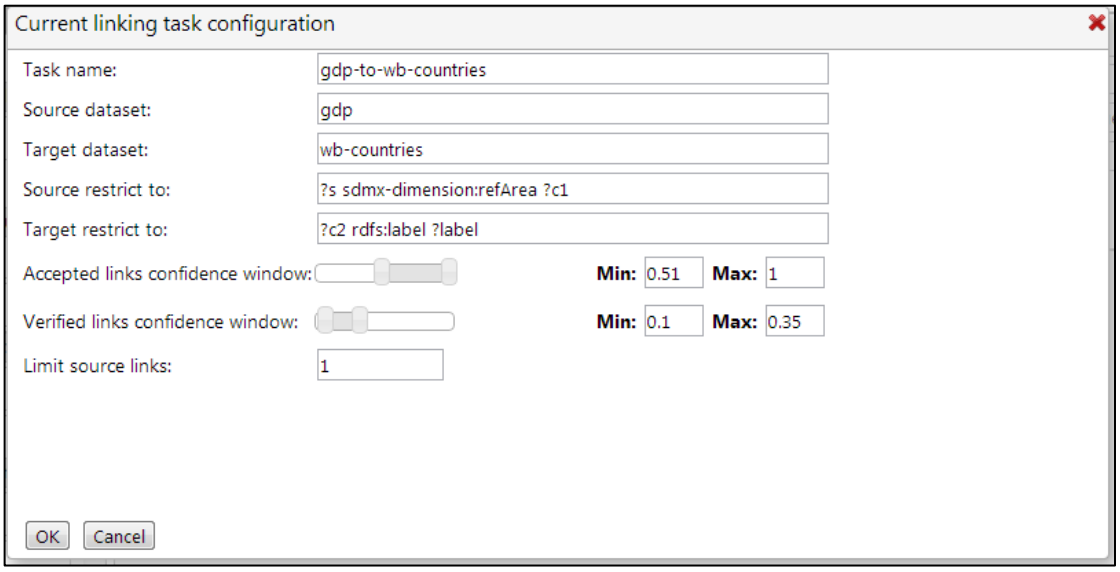
Καθορίζονται τα datasets που πρόκειται να συνδεθούν καθώς και μια συνθήκη επιλογής γραμμένη σε SPARQL ώστε να καθοριστεί ποιες τριπλέτες θα συγκριθούν από το κάθε σύνολο δεδομένων. Ακόμη, επιλέγεται μια μεταβλητή για κάθε συνθήκη που αντιπροσωπεύει τα URIs που θα συγκριθούν. Βάσει των μεταβλητών αυτών βρίσκονται οι ιδιότητες που σχετίζονται με αυτά τα URIs. Πατώντας *Process the task* ενεργοποιείται το κεντρικό πάνελ της εφαρμογής.

Το πάνελ περιλαμβάνει έναν καμβά από όπου ο χρήστης μπορεί να επιλέξει στοιχεία για να σχηματίσει τον κανόνα σύνδεσης καθώς και το πλαίσιο στο οποίο σχηματίζεται γραφικά ο κανόνας. Τα στοιχεία περιλαμβάνουν τις εισόδους του κανόνα (δείχνουν το μονοπάτι ιδιοτήτων που οδηγεί στα στοιχεία που θα συγκριθούν) και τις συναρτήσεις

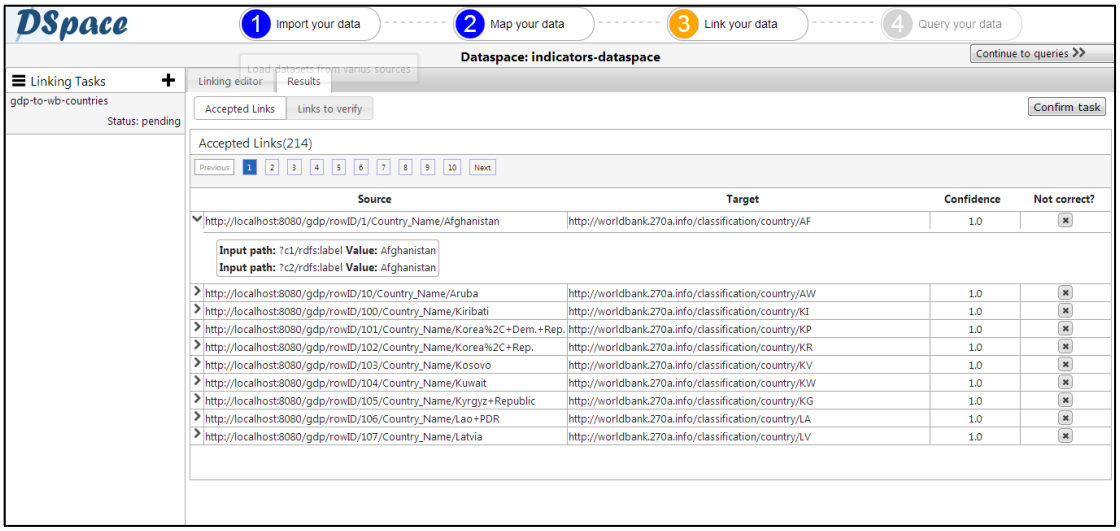
μετασχηματισμού, σύγκρισης και συνάθροισης όπως αυτές ορίζονται από το SILK Framework. Χρησιμοποιώντας την τεχνική *drag n drop* μεταφέρονται τα στοιχεία του καμβά στο πλαίσιο και σχηματίζεται το δέντρο που ρυθμίζει τη σύνδεση (βασισμένο στη γλώσσα Silk-LSL του Silk Framework) . Το δέντρο για το συγκεκριμένο task φαίνεται παρακάτω:



Πατώντας στο πλήκτρο *Configuration* ο χρήστης μπορεί να δει και να αλλάξει τις παραμέτρους του task. Χρειάζεται να οριστούν τα παράθυρα αποδοχής των συνδέσμων ώστε να κατηγοριοποιηθούν σε αυτούς που θεωρούνται αποδεκτοί και σε αυτούς που πρέπει να επαληθευτεί η ορθότητά τους. Η κατηγοριοποίηση βασίζεται στην ισχύ (confidence) κάθε σύνδεσης που παράγεται.

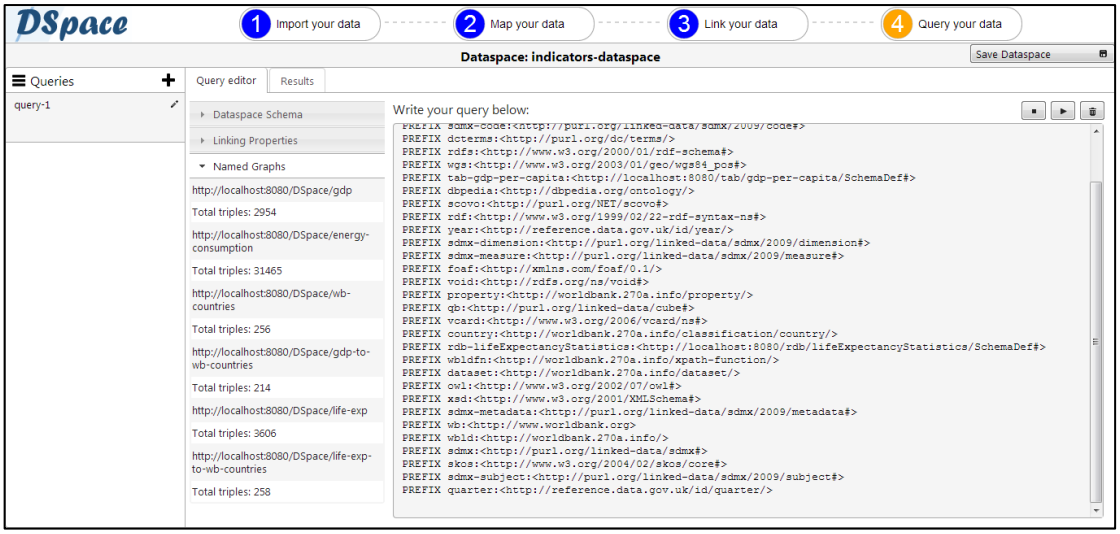


Με το πλήκτρο *Start linking* ξεκινά η διαδικασία της σύνδεσης. Οι σύνδεσμοι που παράγονται από κάθε task φαίνονται στην καρτέλα *Results* όπου ο χρήστης μπορεί να δει τις δυο κατηγορίες συνδέσμων μαζί με τις πληροφορίες που συνοδεύουν κάθε σύνδεσμο. Οι πληροφορίες είναι οι τιμές βάσει των οποίων έγιναν οι συγκρίσεις. Η παρακάτω εικόνα δείχνει τα αποτελέσματα της σύνδεσης των δυο datasets.



Αν ο χρήστης δεν είναι ικανοποιημένος με τους παραγόμενους συνδέσμους μπορεί να αλλάξει τον κανόνα σύνδεσης και να επαναλάβει τη διαδικασία. Ακόμη, μπορεί να μεταφέρει συνδέσμους από τη μια κατηγορία στην άλλη. Για παράδειγμα, αν ένας σύνδεσμος βρεθεί στην κατηγορία των συνδέσμων προς αξιολόγηση ο χρήστης μπορεί να τον μεταφέρει στους αποδεκτούς συνδέσμους. Ο χρήστης επιβεβαιώνει το task πατώντας το πλήκτρο *Confirm*. Τότε οι σύνδεσμοι που είναι σωστοί (αποδεκτοί) προστίθενται στο dataspace της εφαρμογής. Δεν υπάρχει περιορισμός στον αριθμό των linking tasks που μπορούν να γίνουν. Με τον ίδιο ακριβώς τρόπο σχηματίζεται και εκτελείται και το δεύτερο task ανάμεσα στα σύνολα δεδομένων life-exp και wb-countries.

Τελικό στάδιο της εφαρμογής είναι η εκτέλεση ερωτημάτων στο dataspace. Ο χρήστης μπορεί να συντάξει όσα ερωτήματα επιθυμεί και να λάβει τα αντίστοιχα αποτελέσματα. Μάλιστα, τα ερωτήματα αποθηκεύονται προσωρινά στη μνήμη ώστε να μη χρειάζεται να τα γράφει ξανά ο χρήστης. Το κεντρικό panel του βήματος αυτού φαίνεται στην επόμενη εικόνα. Στα αριστερά της οθόνης φαίνονται τα ερωτήματα που έχουν εισαχθεί. Στο κεντρικό τμήμα φαίνονται οι πληροφορίες του dataspace όπως το λεξιλόγιο του χώρου δεδομένων, πληροφορίες σχετικά με τις συνδέσεις που έχουν πραγματοποιηθεί καθώς και οι γράφοι που υπάρχουν στο χώρο δεδομένων. Ανά πάσα στιγμή είναι η δυνατή η μόνιμη αποθήκευση του χώρου δεδομένων στη μνήμη.



Για τις ανάγκες του σεναρίου χρήσης θα εκτελεστούν δυο sparql ερωτήματα στο χώρο δεδομένων που έχει σχηματιστεί. Το πρώτο ερώτημα αφορά την απλή συλλογή των τριών δεικτών (προσδόκιμο ζωής, κατανάλωση ενέργειας και κατά κεφαλήν εισόδημα) ανά χώρα για τα έτη 2008-2010. Το ερώτημα ονομάζεται *all-indicators* και φαίνεται παρακάτω:

```
SELECT ?year ?country ?countryName ?energyConsumption ?gdp
?lifeExpectancy
FROM NAMED <http://localhost:8080/DSpace/energy-consumption>
FROM NAMED <http://localhost:8080/DSpace/gdp>
FROM NAMED <http://localhost:8080/DSpace/life-exp>
FROM NAMED <http://localhost:8080/DSpace/gdp-to-wb-countries>
FROM NAMED <http://localhost:8080/DSpace/life-exp-to-wb-ountries>
FROM NAMED <http://localhost:8080/DSpace/wb-countries>
WHERE{
 GRAPH <http://localhost:8080/DSpace/energy-consumption> {
    SELECT ?country ?year ?countryName ?energyConsumption
    WHERE{
       ?o rdf:type qb:Observation.
       ?o sdmx-dimension:refPeriod ?year.
       ?o sdmx-dimension:refArea ?country.
       ?o sdmx-measure:obsValue ?energyConsumption.
       GRAPH <http://localhost:8080/DSpace/wb-countries> {
         ?country rdfs:label ?countryName
       }
       FILTER(sameTerm(?year,year:2008)|| sameTerm(?year,year:2009)||
```

```
sameTerm(?year,year:2010))
    }
 }
 GRAPH <http://localhost:8080/DSpace/gdp> {
    SELECT ?year ?country ?gdp
    WHERE{
       ?s rdf:type qb:Slice.
       ?s sdmx-dimension:refArea ?gdp_country.
       ?s qb:observation ?o.
       ?o sdmx-dimension:refPeriod ?year.
       ?o sdmx-measure:obsValue ?gdp.
       GRAPH <http://localhost:8080/DSpace/gdp-to-wb-countries> {
         ?gdp_country owl:sameAs ?country
       }
    }
 }
 GRAPH <http://localhost:8080/DSpace/life-exp> {
    SELECT ?year ?country ?lifeExpectancy
    WHERE{
       ?s rdf:type qb:Slice.
       ?s sdmx-dimension:refArea ?life_exp_country.
       ?s qb:observation ?o.
       ?o sdmx-dimension:refPeriod ?year.
       ?o sdmx-measure:obsValue ?lifeExpectancy.
       GRAPH <http://localhost:8080/DSpace/life-exp-to-wb-countries>
       {
         ?life_exp_country owl:sameAs ?country
       }
    }
 }
}
ORDER BY DESC(?year) DESC(?gdp)
```

Στο ερώτημα αυτό επιλέγονται από κάθε γράφο (που αντιστοιχεί σε κάθε dataset) το έτος, το μετρούμενο μέγεθος και το URI της χώρας. Αξίζει να σημειωθεί στο σημείο αυτό πως το URI

είναι εκείνο που χρησιμοποιεί η Παγκόσμια Τράπεζα και επιλέγεται μέσω των συνδέσεων που έχουν γίνει (γράφοι `http://localhost:8080/DSpace/life-exp-to-wb-countries` και `http://localhost:8080/DSpace/gdp-to-wb-countries`). Με τον τρόπο αυτό έχουμε μια κοινή αναπαράσταση για τα URIs των χωρών.

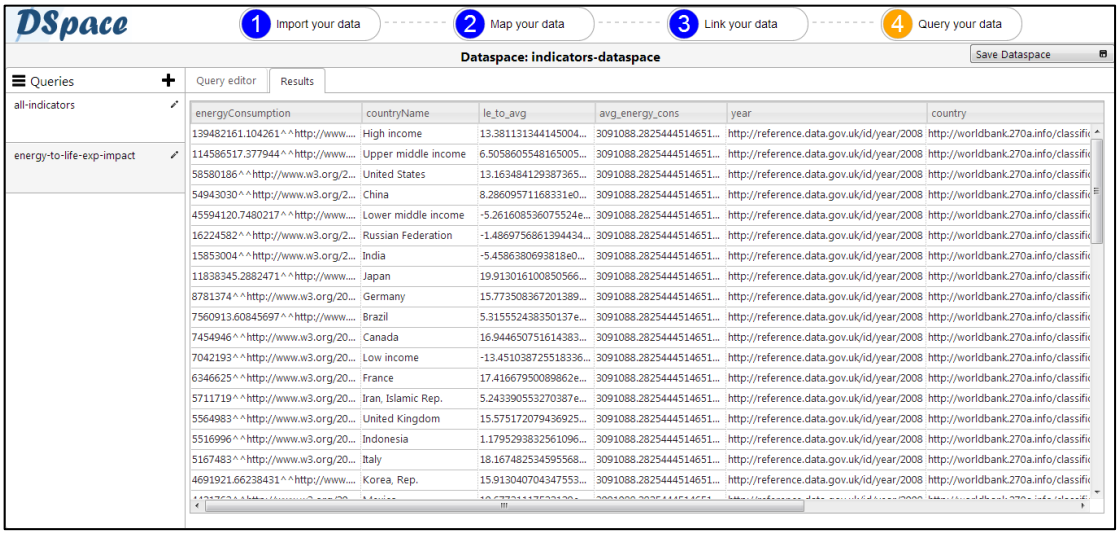
Με το δεύτερο sparql ερώτημα (*energy-to-life-exp-impact*) θέλουμε να δούμε την επίδραση της κατανάλωσης ενέργειας στο προσδόκιμο ζωής ανά χώρα για τα έτη 2008-2010. Πιο συγκεκριμένα, για κάθε έτος υπολογίζεται ο μέσος όρος της παγκόσμιας κατανάλωσης ενέργειας και του προσδόκιμου ζωής. Στη συνέχεια επιλέγονται οι χώρες που έχουν κατανάλωση ενέργειας πάνω από το μέσο όρο και υπολογίζεται η διαφορά του προσδόκιμου ζωής σε αυτές από τον αντίστοιχο μέσο όρο. Η δομή του ερωτήματος είναι η ακόλουθη:

```
SELECT ?year ?country ?countryName ?energyConsumption
?avg_energy_cons ?le_to_avg
FROM NAMED <http://localhost:8080/DSpace/energy-consumption>
FROM NAMED <http://localhost:8080/DSpace/gdp>
FROM NAMED <http://localhost:8080/DSpace/life-exp>
FROM NAMED <http://localhost:8080/DSpace/gdp-to-wb-countries>
FROM NAMED <http://localhost:8080/DSpace/life-exp-to-wb-ountries>
FROM NAMED <http://localhost:8080/DSpace/wb-countries>
WHERE{
 GRAPH <http://localhost:8080/DSpace/energy-consumption> {
    SELECT ?year ?country ?countryName ?energyConsumption
?avg_energy_cons
    WHERE{
       ?o rdf:type qb:Observation.
       ?o sdmx-dimension:refPeriod ?year.
       ?o sdmx-dimension:refArea ?country.
       ?o sdmx-measure:obsValue ?energyConsumption.
       GRAPH <http://localhost:8080/DSpace/wb-countries> {
         ?country rdfs:label ?countryName
       }
       FILTER(sameTerm(?year,year:2008)|| sameTerm(?year,year:2009)||
sameTerm(?year,year:2010)).
       {
         SELECT ?year (AVG(?energyConsumption) AS ?avg_energy_cons)
         WHERE{
```

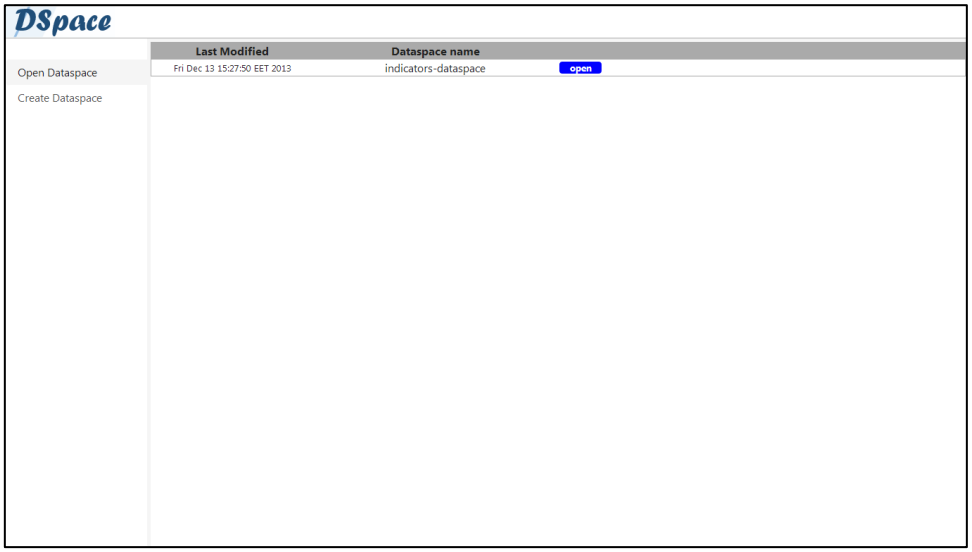
```
            ?o rdf:type qb:Observation.
            ?o sdmx-dimension:refPeriod ?year.
            ?o sdmx-dimension:refArea ?country.
            ?o sdmx-measure:obsValue ?energyConsumption.
            GRAPH <http://localhost:8080/DSpace/wb-countries> {
                 ?country rdfs:label ?countryName
            }
            FILTER(sameTerm(?year,year:2008)||
sameTerm(?year,year:2009)|| sameTerm(?year,year:2010)).
         }
         GROUP BY (?year)
       }
    }
    HAVING(?energyConsumption >= ?avg_energy_cons)
 }
 GRAPH <http://localhost:8080/DSpace/life-exp> {
    SELECT    ?year    ?country    ?countryName    ((?lifeExpectancy-
?avg_life_exp)/?avg_life_exp*100 AS ?le_to_avg)
    WHERE{
        ?s rdf:type qb:Slice.
        ?s sdmx-dimension:refArea ?life_exp_country.
        ?s qb:observation ?o.
        ?o sdmx-dimension:refPeriod ?year.
        ?o sdmx-measure:obsValue ?lifeExpectancy.
        GRAPH <http://localhost:8080/DSpace/life-exp-to-wb-countries>
        {
           ?life_exp_country owl:sameAs ?country
        }
        {
         SELECT ?year (AVG(?life_exp) AS ?avg_life_exp)
         WHERE{
                 ?s rdf:type qb:Slice.
                 ?s sdmx-dimension:refArea ?life_exp_country.
                 ?s qb:observation ?o.
                 ?o sdmx-dimension:refPeriod ?year.
```

```
                    ?o sdmx-measure:obsValue ?life_exp.

            }

            GROUP BY (?year)

          }

      }

  }

}

ORDER BY ?year DESC(?energyConsumption)
```

Τα αποτελέσματα που προκύπτουν φαίνονται στην παρακάτω εικόνα:



Τέλος, αφού έχει αποθηκευτεί ο χώρος δεδομένων στη μνήμη, πατώντας στο λογότυπο της εφαρμογής πάνω αριστερά επιστρέφουμε στην αρχική σελίδα. Εκεί βλέπουμε να υπάρχει ο χώρος δεδομένων που σχηματίστηκε και μπορεί να ανοιχθεί ξανά ώστε να εκτελεστούν τα υπάρχοντα ή νέα ερωτήματα.



9

Επίλογος

9.1 Σύνοψη και συμπεράσματα

Στα πλαίσια της διπλωματικής εργασίας παρουσιάστηκε ο αλγόριθμος μετατροπής πινακοειδών και σχεσιακών δεδομένων σε RDF και εφαρμόστηκε, όπως φάνηκε και από το σενάριο χρήσης, σε πραγματικά δεδομένα. Ακόμη, αναπτύχθηκε το γραφικό περιβάλλον της εφαρμογής το οποίο απλοποίησε τη διαδικασία ολοκλήρωσης. Το σενάριο χρήσης που παρουσιάστηκε στην προηγούμενη ενότητα επιβεβαίωσε την ευκολία της ολοκλήρωσης των ετερογενών δεδομένων που εισήχθησαν καθώς και την εξαγωγή χρήσιμης πληροφορίας από αυτά. Έτσι, η διπλωματική συνεισέφερε ενεργά στην επίλυση των θεμάτων ομογενοποίησης των δεδομένων και συντόμευσης της ολοκλήρωσής τους.

Βέβαια, αυτό δεν σημαίνει πως τα προβλήματα αυτά επιλύθηκαν πλήρως. Η ευελιξία του αλγορίθμου μετατροπής είναι σχετικά περιορισμένη καθώς λαμβάνει υπόψιν ένα συγκεκριμένο τρόπο αναπαράστασης της πληροφορίας, όπως αναφέρθηκε και στην Ενότητα 7.1. Επίσης, από τη διαδικασία ολοκλήρωσης απουσιάζει ένα εξίσου σημαντικό μέρος που είναι αυτό της ποιότητας των δεδομένων.

9.2 Μελλοντικές επεκτάσεις

Υπάρχουν, σαφώς, μεγάλα περιθώρια βελτίωσης της παρούσας εφαρμογής. Οι κυριότερες ενέργειες που μπορούν να πραγματοποιηθούν είναι:

- Ενσωμάτωση περισσότερων πηγών δεδομένων, όπως Web APIs, καθώς και επέκταση των τύπων των αρχείων που μπορούν να εισαχθούν
- Ταχύτερη μετατροπή μεγάλου όγκου δεδομένων σε RDF με τη χρήση τεχνικών παράλληλης επεξεργασίας
- Εισαγωγή υποσυστήματος ποιότητας και εγκυρότητας δεδομένων. Οι θεωρητικές βάσεις του υποσυστήματος αυτού μελετήθηκαν και αναλύθηκαν στα πλαίσια της διπλωματικής
- Δυνατότητα για δημοσίευση μέρους ή ολόκληρου του dataspace της εφαρμογής στο LOD Cloud
- Ενσωμάτωση της λογικής της αρχιτεκτονικής Pay-as-you-Go στο σύστημα με τη δημοσίευση των mappings και με την αναζήτηση mappings στον Ιστό
- Βελτιώσεις που αφορούν το γραφικό περιβάλλον της εφαρμογής

10

Βιβλιογραφία

[1] Madhavan, J., R. Jeffery, S., Cohen, S., Dong, X., Ko, D., Yu, C., et al. (2007). Webscale Data Integration: You can only afford to Pay As You Go. *Google, Inc.*

[2] Wu, Z., Chen, H., Wang, H., Wang, Y., Mao, Y., Tang, J., και συν. (2006). Dartgrid: a Semantic Web Toolkit for Integrating Heterogeneous Relational Databases.

[3] Araujo, S., Hidders, J., Schwabe, D., & Vries, A. (2011). SERIMI – Resource Description Similarity, RDF Instance Matching and Interlinking.

[4] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., & Aumueller, D. (2005). Triplify – Light-Weight Linked Data Publication from Relational Databases.

[5] Berners-Lee, T. (1998). *Relational Databases on the Semantic Web,Design Issue Note*. Ανάκτηση από http://www.w3.org/DesignIssues/RDB-RDF.html

[6] Berners-Lee, T. (2006). *Linked Data - Design Issues*. Ανάκτηση από http://www.w3.org/DesignIssues/LinkedData.html

[7] Bizer, C., & Cyganiak, R. (2006). D2R Server – Publishing Relational Databases on the Semantic Web.

[8] Bizer, C., & Cyganiak, R. (2006). Quality-driven information filtering using the WIQA policy framework.

[9] Bizer, C., & Schultz, A. (2010). The R2R Framework: Publishing and Discovering Mappings on the Web. *1st International Workshop on Consuming Linked Data (COLD 2010).* Shanghai.

[10] Bizer, C., & Seaborne, A. (2004). D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs.

[11] Bleiholder, J., & Naumann, F. (2008). Data Fusion. *ACM Comput. Surv*.

[12] Elmagarmid, A., G. Ipeirotis, P., & S. Verykios, V. (2007). Duplicate Record Detection: A Survey. *Ieee Transactions On Knowledge And Data Engineering, 19.*

[13] Heath, T., & Bizer, C. (2011). *Linked Data Evolving the Web into a Global Data Space.* Morgan & Claypool.

[14] Isele, R., Harth, A., Umbrich, J., & Bizer, C. (2010). Ldspider:An open-source crawling framework for the web of linked data. *ISWC 2010 Posters & Demonstrations Track: Collected Abstracts*, *658.*

[15] Klyne, G., & J. Carroll, J. (2004). *Resource Description Framework (RDF): Concepts and*. Ανάκτηση από http://www.w3.org/TR/rdf-concepts/

[16] Köpcke, H., & Rahm, E. (2009). Frameworks for entity matching: A comparison.

[17] Maali, F., Cyganiak, R., & Peristeras, V. (2011). Re-using Cool URIs:Entity Reconciliation Against LOD Hubs. *LDOW2011.* Hyderabad, India.

[18] Mendes, P., Mühleisen, H., & Bizer, C. (2012). Sieve: Linked Data Quality Assessment and Fusion. *LWDM 2012.* Berlin, Germany.

[19] S. Sahoo, S., Halb, W., Hellmann, S., Idehen, K., Thibodeau, T., Auer, S., et al. (2008). *A Survey of Current Approaches for Mapping of Relational Databases to RDF*. W3C RDB2RDF Incubator Group.

[20] Schultz, A., Matteini, A., Isele, R., Bizer, C., & Becker, C. (2012). LDIF - Linked Data Integration Framework. *21st International World Wide Web Conference (WWW2012).* Lyon, France.

[21] Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). Silk – A Link Discovery Framework for the Web of Data. *LDOW 2009.*