ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΙΟΥΡΓΙΑ INTERFACE ΣΤΑ ΑΓΓΛΙΚΑ ΓΙΑ ΤΗΝ ΓΛΩΣΣΑ SPARQL

ΑΝΔΡΙΑΝΗΣ ΑΘΑΝΑΣΙΟΣ - ΚΟΥΤΕΛΙΕΡΗΣ ΧΡΙΣΤΟΦΟΡΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΠΑΠΑΔΑΚΗΣ ΝΙΚΟΛΑΟΣ

Περιεχόμενα

1. Περίληψη

Σε αυτή την εργασία παρουσιάζουμε ένα εργαλείο που μπορεί να μεταφράσει ένα κείμενο φυσικής γλώσσας ,στην δίκια μας περίπτωση από τα Αγγλικά, σε ένα SPARQL query και να εξάγει τα αποτελέσματα από μια οντολογία OWL. Σχεδιάσαμε μια οντολογία με το εργαλείο Protégé με θέμα τις ποδοσφαιρικές ομάδες και αφού κάναμε ορισμένες εγγραφές ,καταφέραμε να πάρουμε αποτελέσματα τρέχοντας επερωτήματα στην γλώσσα SPARQL με το εργαλείο Twinkle. Έπειτα σχεδιάσαμε ένα Java Interface με το Eclipse και με κατάλληλους συνδυασμούς μέσα στο GUI που δημιουργήσαμε μεταφράσαμε Αγγλικές εκφράσεις σε SPARQL. Για να πάρουμε τα αποτελέσματα από την οντολογία μας, βάλαμε τις βιβλιοθήκες της Jena(ένα Java Interface ειδικό για εφαρμογές του Semantic Web) και αφού συνδυάζαμε το παραγόμενο SPARQL query με την οντολογία παίρναμε το επιθυμητό αποτέλεσμα.

2. Εισαγωγή

2.1. Ο Παγκόσμιος Ιστός (World Wide Web)

Ο Παγκόσμιος Ιστός (World Wide Web) αποτελεί πλέον στην εποχή μας ένα βασικό μέσο επικοινωνίας και μεταφοράς γνώσεων και πληροφοριών. Αν το πρώτο βήμα προς την εξάπλωση του παγκόσμιου ιστού ήταν η τεχνολογική εξέλιξη των δικτύων που επέφερε την ταχύτατη και εύκολη μεταφορά οποιασδήποτε πληροφορίας από τον ένα υπολογιστή στον άλλο, η εδραίωση της γλώσσας υπομνηματισμού html (hypertext markup language) [RHJ] ως η γλώσσα αναπαράστασης της πληροφορίας στον παγκόσμιο ιστό, αποτέλεσε τον «κοινό κώδικα επικοινωνίας» όλης αυτής της διακινούμενης πληροφορίας.

Η html είναι απλή, εύκολη στη χρήση της και ανθρωπίνως αναγνώσιμη, αφού μια σελίδα γραμμένη σε html είναι απλά ένα έγγραφο γραμμένο σε έναν οποιοδήποτε επεξεργαστή κειμένου. Ειδικές σημάνσεις (tags) ερμηνεύουν τον τρόπο παρουσίασης της πληροφορίας, «μαρκάροντάς» την με την ανάλογη σήμανση.

Κατάλληλες εφαρμογές με το όνομα φυλλομετρητές του παγκόσμιου ιστού (Web browsers), «αποκωδικοποιούν» την html πληροφορία και την αναπαριστούν με τον κατάλληλο τρόπο στην οθόνη του υπολογιστή κάθε χρήστη. Έτσι, ο φυλλομετρητής ενεργεί σαν μια ομοιόμορφη γραφική διεπαφή, μέσα από την οποία δίνεται πρόσβαση σε πληροφορίες που διακινούνται στον παγκόσμιο ιστό. Υπερσύνδεσμοι (hyperlinks) συνδέουν τις σελίδες μεταξύ τους, δημιουργώντας ένα γιγάντιο δίκτυο πληροφοριών (εικόνα 1). Ακολουθώντας τους διάφορους υπερσυνδέσμους, κάθε χρήστης έχει πρόσβαση σε κάθε είδους πληροφορία διακινείται στον παγκόσμιο ιστό, μέσα από τον φυλλομετρητή του.

Εικόνα 1: Ο Παγκόσμιος ιστός (World Wide Web)

2.2. Η γλώσσα XML

Αν και η γλώσσα html είναι επαρκής για την διαχείριση, διακίνηση και απεικόνιση κάθε είδους πληροφορίας στον παγκόσμιο ιστό, σε σύγχρονες εφαρμογές, όπως διαμεσολαβητές πληροφοριών (σημασιολογικές πύλες διαδικτύου, ψηφιακά μουσεία και βιβλιοθήκες), εφαρμογές ηλεκτρονικού εμπορίου (δικτυακοί τόποι ηλεκτρονικών αγορών και πωλήσεων), εφαρμογές ηλεκτρονικής μάθησης εξ αποστάσεων ή πλέγματα γνώσης (knowledge grids), χρειάζεται ένας περισσότερο αυτοματοποιημένος τρόπος εύρεσης, ανάλυσης, συλλογής και παρουσίασης της διακινούμενης πληροφορίας.
Ένα νέο πρότυπο, αυτό της XML (extensible Markup Language) [BPMM], επιτρέπει τον διαχωρισμό μεταξύ του τρόπου εμφάνισης της πληροφορίας και της ερμηνείας που δίνεται στην πληροφορία αυτή. Η XML έχει «κληρονομήσει» τα θετικά της html: Είναι απλή, εύκολη στην ανάγνωση και στην γραφή της, καθολικά αποδεκτή και αξιοποιήσιμη από κάθε υπολογιστή συνδεδεμένο στο διαδίκτυο. Όμως, η XML είναι πολύ περισσότερα από αυτά. Στην XML δίνεται η δυνατότητα ορισμού σημάνσεων από τον ίδιο το χρήστη ώστε να αποδίδεται στην διακινούμενη πληροφορία η επιθυμητή σημασιολογία.

Η αναπαράσταση της πληροφορίας σε XML δεν περιορίζεται σε ένα σταθερό σύνολο σημάνσεων, όπως στην html. Αντίθετα, η σήμανση που δίνεται στην πληροφορία δηλώνει και την σημασιολογία που αυτή έχει κάθε φορά. Έτσι, η XML διαθέτει ευελιξία, αφού αλλάζοντας απλά τις σημάνσεις δίνεται μια διαφορετική σημασιολογία στην διακινούμενη πληροφορία, ενώ παράλληλα
διαθέτει και επεκτασιμότητα, αφού η σημασιολογία της πληροφορίας μπορεί να εμπλουτίζεται διαρκώς με νέες σημάνσεις.
Μια σήμανση σε ένα XML έγγραφο μπορεί να «μαρκάρει» πληροφορία που ενδεχομένως περιέχει άλλες σημάνσεις για το «μαρκάρισμα» υποσυνόλων της πληροφορίας αυτής. Έτσι, οι σημάνσεις σε ένα XML έγγραφο αποτελούν ένα σύνολο από σύνθετα και ατομικά αντικείμενα και μπορούν να αναπαρασταθούν σε μια δενδρική μορφή, όπου κάθε κόμβος αποτελεί μια σήμανση του XML εγγράφου και τα παιδιά του είναι οι σημάνσεις που χρησιμοποιούνται από υποσύνολα της πληροφορίας που «μαρκάρεται» από τη συγκεκριμένη σήμανση
Η εγκυρότητα ενός XML εγγράφου ελέγχεται με βάση κατάλληλων XML σχημάτων (XML Schema) και ορισμών τύπου εγγράφου (Document Type Definition- DTD), με βάση των οποίων ελέγχονται συγκεκριμένα κριτήρια εγκυρότητας. Για παράδειγμα, το προηγούμενο XML έγγραφο θα μπορούσε να ελέγχεται ως προς την εγκυρότητά του με βάση το παρακάτω DTD (εικόνα 2).

```
<!DOCTYPE Person [
<!ELEMENT Person (name, born_in, homepage)>
<!ATTLIST Person oid ID #REQUIRED>
<!ELEMENT name      (first, last)>
<!ELEMENT first     (#PCDATA)>
<!ELEMENT last      (#PCDATA)>
<!ELEMENT born_in     (#PCDATA)>
<!ELEMENT homepage      (link)>
<!ELEMENT link EMPTY>
<!ATTLIST link homepage ENTITY #REQUIRED>
]>
```

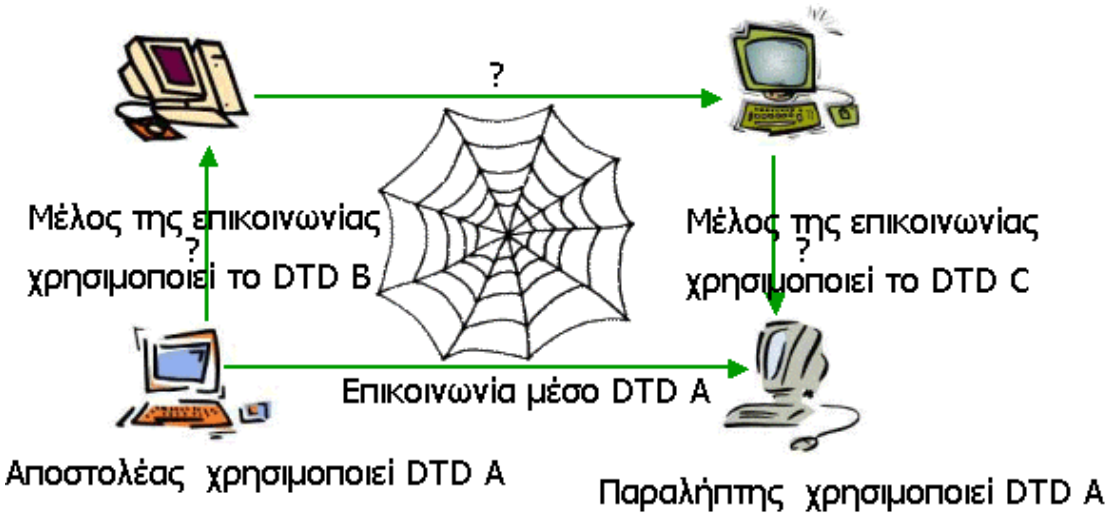
Εικόνα 2: Ορισμός τύπου εγγράφου

Παρά τις αναμφισβήτητα περισσότερες δυνατότητες που προσφέρει η XML σε σύγκριση με την html, δεν μπορεί να χρησιμοποιηθεί για μια

αυτοματοποιημένη διαχείριση της πληροφορίας στο διαδίκτυο. Η διαλειτουργικότητα που μπορεί να προσφέρει η XML περιορίζεται από τις διαφορετικές ερμηνείες που μπορεί να δώσει κανείς στην διακινούμενη πληροφορία. Η ανταλλαγή πληροφορίας βασισμένη στην XML προϋποθέτει την εξ αρχής συμφωνία σε θέματα όπως:

- Ποια είναι η ερμηνεία των σημάνσεων που δίνεται στα XML έγγραφα.
- Ποιοι είναι οι ορισμοί τύπων εγγράφου και τα XML Schemata για την εγκυρότητα των εγγράφων.
- Ποιες είναι οι λειτουργίες στα δεδομένα.

Είναι φανερό, ότι μια τέτοια προσυμφωνία σε θέματα απόδοσης ερμηνείας που δίνεται στην διακινούμενη πληροφορία δεν μπορεί να γίνει, όταν η πληροφορία διακινείται μέσα στον παγκόσμιο ιστό. Ακόμα και αν επιτευχθεί σε συγκεκριμένο αριθμό μελών που συμμετέχουν στην διακίνηση αυτής της πληροφορίας, κάθε νέο μέλος θα αποδίδει τη δική του ερμηνεία, που πιθανώς θα διαφέρει με αυτήν που δίνει κάθε άλλο μέλος. Συνεπώς, ασυμφωνίες στην απόδοση ενιαίας ερμηνείας της πληροφορίας που διακινείται, καθιστά την διαλειτουργικότητα με τη χρήση της XML στο διαδίκτυο αρκετά προβληματική (εικόνα 3).



Εικόνα 3: Η χρήση της XML δεν λύνει τα προβλήματα διαλειτουργικότητας

2.3. Ο Σημασιολογικός Ιστός(Semantic Web)

Η ανάπτυξη του Παγκόσμιου Ιστού (World Wide Web) κατέστησε το διαδίκτυο προσβάσιμο σε εκατομμύρια χρήστες, επιτρέποντας την απρόσκοπτη δημοσιοποίηση και πρόσβαση σε έγγραφα στο διαδίκτυο. Η εκρηκτική ανάπτυξη του Παγκόσμιου Ιστού δημιούργησε προβλήματα "πληροφοριακής υπερφόρτισης". Η παγκόσμια ερευνητική κοινότητα έχει στραφεί εδώ και λίγα χρόνια σε μία νέα κατεύθυνση εξέλιξης του ιστού, η οποία ονομάζεται "Σημασιολογικός Ιστός " (Semantic Web) *Σημασιολογικός Ιστός* (Semantic Web) είναι μια εναλλακτική προσέγγιση στην μεταφορά, αξιοποίηση και παρουσίαση του περιεχομένου των πληροφοριών που διακινούνται στο διαδίκτυο προτάθηκε από τον Tim Berners Lee, ιδρυτή του World Wide Consortium (W3C) και του Παγκόσμιου Ιστού.

Σκοπός του Σημασιολογικού Ιστού είναι η επίτευξη μιας καλύτερης διαλειτουργικότητας ανάμεσα σε εφαρμογές και πληροφορίες, ώστε οι διάφορες υπηρεσίες του διαδικτύου να ικανοποιούν τις ανάγκες των χρηστών με έναν περισσότερο αυτοματοποιημένο τρόπο. Για τον σκοπό αυτό, κάθε είδους πληροφορία στο σημασιολογικό ιστό εκφράζεται με έναν πλούσιο σημασιολογικό τρόπο, ο οποίος είναι όχι μόνο μηχανικά αναγνώσιμος αλλά και μηχανικά κατανοήσιμος και επεξεργάσιμος. Μέσα στην «νέα γενιά του παγκόσμιου ιστού», αυτή του Σημασιολογικού Ιστού, η πλοήγηση στις διάφορες πληροφορίες θα είναι βασισμένη στις διάφορες έννοιες και τις συσχετίσεις μεταξύ τους, ακολουθώντας όχι απλά υπερσυνδέσμους όπως στον συμβατικό ιστό, αλλά υπερσυνδέσμους με σημασιολογικό χαρακτήρα οι οποίοι εκφράζουν την εννοιολογική συσχέτιση μεταξύ των διάφορων πληροφοριών (εικόνα 4).

Εικόνα 4: Οργάνωση της πληροφορίας στο σημασιολογικό ιστό

2.3.1. Σχήματα περιγραφών και περιγραφές πόρων

Στον Σημασιολογικό Ιστό η πληροφορία οργανώνεται με βάση διάφορα σχήματα περιγραφών και με κατάλληλες περιγραφές των διάφορων πληροφοριακών πόρων.
Τα σχήματα περιγραφών και οι περιγραφές των πόρων χαρακτηρίζονται με το όνομα μεταδεδομένα της πληροφορίας, δηλαδή «δεδομένα για τα δεδομένα». Ο Σημασιολογικός Ιστός αξιοποιεί αυτή την «πληροφορία της πληροφορίας» και παρέχει πιο αποτελεσματικούς τρόπους πρόσβασης στις διαθέσιμες πληροφορίες.
Η σημασιολογική οργάνωση της πληροφορίας σε σχήματα περιγραφών έχει ως κύριο γνώρισμα τις διάφορες έννοιες καθώς και τις σχέσεις μεταξύ τους, σε μορφή συσχετίσεων ή σχέσεων υπαλληλίας. Τα σχήματα περιγραφών δεν είναι τίποτα άλλο από λεξιλόγια έγκυρων όρων – ονόματα εννοιών και ιδιοτήτων- που μπορούν να χρησιμοποιηθούν για την περιγραφή των πληροφοριακών πόρων. Οι πληροφοριακοί πόροι μπορεί να είναι αντικείμενα προσπελάσιμα από τον παγκόσμιο ιστό, όπως για παράδειγμα δικτυακές σελίδες, συγκεκριμένα έγγραφα κλπ, αλλά και μη προσπελάσιμα αντικείμενα

όπως για παράδειγμα βιβλία, πίνακες ζωγραφικής, πρόσωπα κτλ. Πολλαπλά λεξιλόγια όρων μπορούν να εφαρμοστούν πάνω από τους ίδιους πληροφοριακούς πόρους και το καθένα να προσφέρει πρόσβαση στην διαθέσιμη πληροφορία με ένα διαφορετικό τρόπο. Περιγραφές πόρων χρησιμοποιούνται για την κατηγοριοποίηση των πληροφοριακών πόρων κάτω από τις διάφορες κλάσεις των σχημάτων περιγραφής.
Ένας πόρος είναι δυνατό να είναι κατηγοριοποιημένος κάτω από πολλαπλές κλάσεις, οι οποίες ενδεχομένως να ανήκουν και σε διαφορετικά σχήματα περιγραφών. Κάθε πόρος διαθέτει διάφορες συσχετίσεις με άλλους πόρους καθώς και ατομικά γνωρίσματα.

2.3.2. Στο πρότυπο των θεματικών χαρτών (topic maps)

Στο πρότυπο των θεματικών χαρτών (topic maps) [ISO/IEE 13250:2000], οι πληροφορίες είναι οργανωμένες σε θέματα (topics). Ένα θέμα μπορεί να είναι ένα οποιαδήποτε αντικείμενο που έχει να κάνει με την πληροφορία του θεματικού χάρτη, όπως για παράδειγμα ένα πρόσωπο, μια αφηρημένη έννοια κλπ. Έτσι για παράδειγμα, σε έναν θεματικό χάρτη που πραγματεύεται πληροφορίες σχετικά με όπερες, πιθανά θέματα ίσως είναι: συνθέτης, Guiseppe Verdi, Giacommo Puccini, Tosca, Ρώμη και άλλα. Από τα θέματα αυτά, άλλα είναι γενικές έννοιες – όπως για παράδειγμα το θέμα συνθέτης, τα οποία ονομάζονται τύποι θεμάτων (topic types) και τα υπόλοιπα είναι κατηγοριοποιημένα κάτω από αυτά. Κάθε θέμα έχει και ένα όνομα (topic name) ορισμένο σε ένα η περισσότερα φάσματα (scopes).
Κάθε θέμα μπορεί να συνδέεται με έναν η περισσότερους πληροφοριακούς πόρους που θεωρούνται σχετικοί με το θέμα αυτό. Οι πόροι αυτοί αποτελούν τα στιγμιότυπα του θέματος και ονομάζονται occurrences. Κάθε τέτοιο στιγμιότυπο αποτελείται από το όνομά του και το ενιαίο αναγνωριστικό πόρου (Uniform Resource Identifier- URI), το οποίο όπως και το topic name μπορεί να ορίζεται σε ένα ή περισσότερα φάσματα. Οι σχέσεις συνάφειας μεταξύ των διαφόρων θεμάτων ονομάζονται στη «γλώσσα» των θεματικών χαρτών associations.

Η σύνταξη της πληροφορίας που αναπαριστάται σε θεματικούς χάρτες γίνεται με τη γλώσσα XTM (XML for Topic Maps) [XTM], η οποία βασίζεται στο συντακτικό της XML.

2.3.3. Το Περιβάλλον Περιγραφής Πόρων (RDF)

Μια ολοκληρωμένη γλώσσα δημιουργίας, ανταλλαγής, επεξεργασίας και επαναχρησιμοποίησης μεταδεδομένων πληροφοριών μέσα στο διαδίκτυο που προωθείται από το World Wide Web Consortium (W3C) είναι το Περιβάλλον Περιγραφής Πόρων (Resource Description Framework - RDF).

Το βασικό μοντέλο δεδομένων του RDF αποτελείται από τρεις βασικούς τύπους αντικειμένων, τους Πόρους (Resources), τις Ιδιότητες (Properties) και τις Προτάσεις (Statements). Οτιδήποτε μπορεί να περιγραφεί με RDF εκφράσεις ονομάζεται πόρος. Σε κάθε πόρο, που όπως είπαμε στην προηγούμενη παράγραφο μπορεί να είναι ένα οποιοδήποτε αντικείμενο προσπελάσιμο ή μη από τον παγκόσμιο ιστό αποδίδεται ένα μοναδικό αναγνωριστικό (Uniform Resource Identifier-URI). Με τη βοήθεια του RDF μοντέλου περιγράφονται οι ιδιότητες ενός πόρου, (γνωρίσματα ή συσχετίσεις με άλλους πόρους) σχηματίζοντας τριάδες της μορφής <υποκείμενο, κατηγόρημα, αντικείμενο>, τις οποίες αποκαλούμε δηλώσεις. Κάθε περιγραφή σε RDF δηλώνει ότι ορισμένα αντικείμενα (δικτυακοί τόποι, πρόσωπα, συγκεκριμένα έγγραφα ή οτιδήποτε) έχουν κάποιες ιδιότητες με ορισμένες τιμές, με κάθε μέρος αυτής της δήλωσης να είναι μοναδικά ορισμένο.

Πέρα από τις RDF περιγραφές των πληροφοριακών πόρων, το μοντέλο δεδομένων RDF αποτελείται και από μια γλώσσα ορισμού σχημάτων (συνόλων κλάσεων και ιδιοτήτων), την RDF Schema Specification Language (RDFS). Με τη βοήθεια του RDFS, είναι δυνατός ο προσδιορισμός μηχανισμών καθορισμού κλάσεων πόρων, καθώς και ο περιορισμός των πιθανών συνδυασμών κλάσεων μεταξύ τους χρησιμοποιώντας κατάλληλες συσχετίσεις. Ένα RDF σχήμα αποτελείται από τις δηλώσεις κλάσεων, γνωρισμάτων και των σχέσεων μεταξύ των κλάσεων. Όμοιοι πόροι είναι ομαδοποιημένοι κάτω από την ίδια κλάση.

Με βάση τα παραπάνω, μπορούμε να διακρίνουμε τρία διαφορετικά επίπεδα αφαίρεσης στο μοντέλο δεδομένων RDF/S: Στο κατώτερο επίπεδο υπάρχουν οι ίδιοι οι πόροι (έγγραφα, δικτυακοί τόποι, πρόσωπα ή οτιδήποτε άλλο). Το επόμενο επίπεδο αφαίρεσης είναι το επίπεδο δεδομένων, όπου γίνεται η περιγραφή των πληροφοριακών πόρων με τη χρήση λεξιλογίων, τα οποία περιγράφονται στο τελευταίο επίπεδο, το επίπεδο σχήματος. Το επίπεδο σχήματος είναι το επίπεδο αφαίρεσης όπου αναπτύσσονται RDF σχήματα για να διευκολύνουν τη σημασιολογική περιγραφή των πόρων. Σε αυτό το επίπεδο, οι κλάσεις αναπαριστούν αφηρημένες οντότητες και αναφέρονται συλλογικά σε σύνολα παρομοίων αντικειμένων.

Για την γραφική απεικόνιση των RDF/S περιγραφών και σχημάτων χρησιμοποιείται ένα μοντέλο κατευθυνόμενων γράφων με ετικέτες τόσο στις ακμές όσο και στους κόμβους που μπορεί εύκολα να συνδυάσει πολλά διαφορετικά λεξιλόγια και να επεκταθεί προσθέτοντας απλώς περισσότερες ακμές. Οι κόμβοι αναπαριστούν αντικείμενα (πόρους ή κλάσεις) και οι ακμές συμβολίζουν σχέσεις μεταξύ των κόμβων (ιδιότητες). Ως κόμβους αναπαριστούμε επίσης και τύπους Literal, δηλαδή αλφαριθμητικά και άλλους βασικούς τύπους δεδομένων, όπως αυτοί ορίζονται στο πρότυπο XML schema. Οι κόμβοι συμβολίζονται ως ελλείψεις και οι ατομικές τιμές ως παραλληλόγραμμα. Οι ακμές μπορεί να είναι τριών ειδών: *απόδοσης γνωρισμάτων (attributes)*, *δημιουργίας στιγμιότυπων* (*instances*) και *υπαλληλίας*. Οι ακμές απόδοσης γνωρισμάτων αναπαριστούν γνωρίσματα κόμβων και σχέσεις μεταξύ τους, ενώ οι ακμές υπαλληλίας χρησιμοποιούνται για να δηλώσουν σε επίπεδο σχήματος ότι ένας κόμβος (κλάση) ή ιδιότητα είναι υποκατηγορία ενός σημασιολογικά ευρύτερου κόμβου ή ιδιότητας αντίστοιχα.

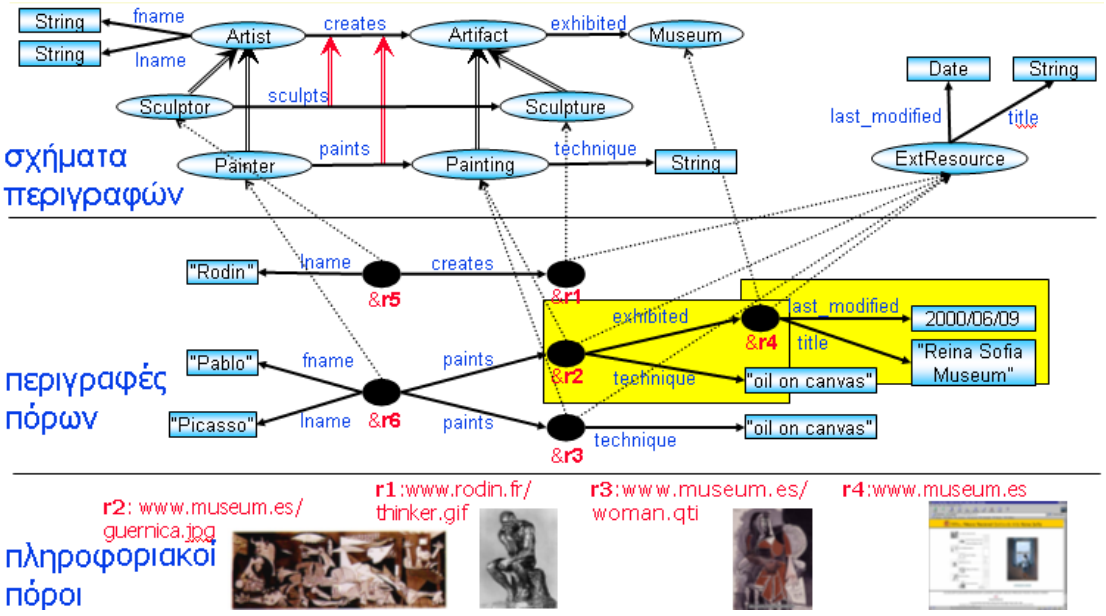
Τέλος, οι ακμές δημιουργίας στιγμιότυπων αποτελούν τη σύνδεση ανάμεσα στα πρότυπα RDF και RDFS, επιτρέποντας τη δημιουργία στιγμιότυπων μίας κλάσης και την απόδοση τύπων σε πληροφοριακούς πόρους που περιγράφονται.

Για τον καλύτερο διαμοιρασμό των RDF περιγραφών πόρων και των σχημάτων περιγραφών στο διαδίκτυο, το μοντέλο RDF «δανείζεται» το συντακτικό της XML.

2.3.4. Παράδειγμα Σημασιολογικής οργάνωσης πληροφορίας σε RDF/S

Με τη χρήση του μοντέλου δεδομένων RDF/S είναι δυνατό να περιγραφεί η σημασιολογική οργάνωση της διακινούμενης πληροφορίας οποιασδήποτε εφαρμογής του σημασιολογικού ιστού, όπως για παράδειγμα μια πολιτιστική πύλη στο διαδίκτυο. Στο παράδειγμα αυτό ας υποθέσουμε πως υπάρχουν τέσσερις πληροφοριακοί πόροι: ένα έγγραφο περιγραφής ενός γλυπτού (ο πόρος με το αναγνωριστικό r1), δυο έγγραφα περιγραφής εικόνων (οι πόροι με τα αναγνωριστικά r2 και r3) και ένας δικτυακός τόπος (ο πόρος με το αναγνωριστικό r4). Ο πόρος r4 περιγράφεται ως ένα ExtResource με ιδιότητες title (με τιμή Reina Sofia Museum) και last_modified (με τιμή 2000/06/09). Ταυτόχρονα είναι κατηγοριοποιημένος κάτω από την έννοια Museum, ώστε να αξιοποιηθεί η σημασιολογική συσχέτιση του με άλλους πόρους, όπως πόροι τεχνουργημάτων (Artifact). Δηλαδή, ο πόρος http:://www.museum.es έχει τίτλο Reina Sofia Museum, τροποποιήθηκε τελευταία φορά στις 2000/06/09, ενώ υπάρχουν τεχνουργήματα που εκθέτονται (exhibited) στον συγκεκριμένο δικτυακό τόπο. Ο πόρος r2 είναι ένα στιγμιότυπο της έννοιας Painting και έχει μια ιδιότητα exhibited που τον συσχετίζει με τον πόρο r4, καθώς και μια ιδιότητα technique με τιμή oil of canvas. Παράλληλα ο πόρος αποτελεί στιγμιότυπο της έννοιας ExtResource, ώστε να του αποδοθούν ιδιότητες χρήσιμες για τον διαχειριστή της σημασιολογικής πύλης. Οι πόροι r1, r2 και r3 ανήκουν σε αυτή την κατηγορία, είναι περιγραφές ταξινομημένες σε πολλαπλές έννοιες (multiple classification). Πάνω από τις περιγραφές των πληροφοριακών πόρων, πολλαπλά σχήματα περιγραφών είναι δυνατό να περιγράψουν τους ίδιους πόρους με διαφορετικά κριτήρια. Έτσι για παράδειγμα, μέσα στη σημασιολογική πύλη η περιγραφή μπορεί να γίνει σύμφωνα με την καλλιτεχνική υπόσταση των πόρων, αλλά και σύμφωνα με τη διοικητική τους πλευρά. Στην πρώτο σχήμα περιγραφών, βασικές έννοιες είναι ο καλλιτέχνης (Artist), το δημιούργημα (Artifact) και τα Μουσεία (Museum). Σχέσεις υπαλληλίας μεταξύ των εννοιών επιτρέπουν την εξειδίκευση της έννοιας *Artist* σε *Sculptor* και *Painting* και της έννοιας *Artifact* σε *Painter* και *Sculpture*. Αυτές είναι και οι κλάσεις του σχήματος, οι οποίες σχετίζονται μεταξύ

τους με ιδιότητες όπως *creates* και *exhibited*. Σχέσεις υπαλληλίας μεταξύ των ιδιοτήτων επιτρέπουν την εξειδίκευση της ιδιότητας *creates* σε *sculpts* και *paints*, ενώ ορισμένες κλάσεις έχουν ατομικά γνωρίσματα, όπως συμβαίνει στις κλάσεις Artist (με τα ατομικά γνωρίσματα fname και lname) και της κλάσης Painting με το γνώρισμα technique.Το άλλο σχήμα περιγραφών χρησιμεύει στην περιγραφή των πόρων βάσει της διοικητικής πληροφορίας που περιέχουν. Η έννοια ExtResource του σχήματος έχει ως γνωρίσματα τον τίτλο (*title*) και την ημερομηνία που τροποποιήθηκε τελευταία φορά (*last_modified*) ένας πόρος της σημασιολογικής πύλης.
Η εικόνα 5 απεικονίζει τους γράφους των RDF σχημάτων περιγραφής και των περιγραφών των πληροφορικών πόρων της πολιτιστικής πύλης.



Εικόνα 5: Περιγραφές πόρων σε μια πολιτιστική πύλη στο διαδίκτυο

2.4. Οντολογίες(Ontologies)

Η Οντολογία σαν λέξη έχει μακρά ιστορία που προέρχεται από την Φιλοσοφία.“*Η μεταφυσική μελέτη της φύσης της ζωής και της ύπαρξης*” είναι ο ορισμός που είχε δώσει ο Αριστοτέλης.

Η έννοια της οντολογίας έχει υιοθετηθεί από την Τεχνητή Νοημοσύνη και σημαίνει “*μια διαμοιρασμένη και κοινή κατανόηση κάποιου τομέα, η οποία μπορεί να ανταλλαγεί μεταξύ ανθρώπων και συστημάτων εφαρμογών*”.

Το πρόβλημα είναι το εξής: Άνθρωποι, οργανισμοί και προγράμματα πρέπει να επικοινωνούν μεταξύ τους. Οι διαφορετικές ανάγκες και το διαφορετικό υπόβαθρο οδηγούν σε αποκλίνουσες οπτικές γωνίες και παραδοχές για πράγματα που στην ουσία είναι ίδια μεταξύ τους. Η έλλειψη κοινής αντίληψης οδηγεί σε προβλήματα στην επικοινωνία μεταξύ ανθρώπων και οργανισμών ,δυσκολίες στον προσδιορισμό των απαιτήσεων και κατά συνέπεια στην ανάπτυξη των προδιαγραφών των συστημάτων.

Οι ανομοιόμορφες μέθοδοι μοντελοποίησης, οι γλώσσες και τα εργαλεία λογισμικού περιορίζουν σοβαρά τη δια-λειτουργικότητα, την επαναχρησιμοποίηση και το διαμοιρασμό εφαρμογών. Τελικά όλα τα παραπάνω οδηγούν στο να ξαναανακαλύπτουμε τον τροχό.

Η λύση είναι η εξάλειψη ή η μείωση της σύγχυσης σχετικά με τις έννοιες και τους όρους και τελικά η απόκτηση κοινής αντίληψης.

Αυτή η κοινή αντίληψη μπορεί να αποτελέσει το ενοποιητικό πλαίσιο ανάμεσα στις διαφορετικές οπτικές γωνίες και να συμβάλει στην βελτίωση της επικοινωνίας, της δια-λειτουργικότητας, και να οδηγήσει σε πλεονεκτήματα αναφορικά με την μηχανική των συστημάτων παρέχοντας δυνατότητες επαναχρησιμοποίησης, βελτιώνοντας την αξιοπιστία, και διευκολύνοντας την ανάπτυξη προδιαγραφών.

2.4.1. Ορισμός

Μια οντολογία είναι μια τυπική (formal), κατηγορηματική (explicit) προδιαγραφή μιας διαμοιρασμένης (shared) εννοιολογικής αναπαράστασης (conceptualization) .Ο όρος ‘εννοιολογική αναπαράσταση’ (conceptualization) αναφέρεται σε ένα αφηρημένο

μοντέλο φαινομένων του κόσμου στο οποίο έχουν προσδιοριστεί οι έννοιες που σχετίζονται με τα φαινόμενα αυτά. Ο όρος ‘κατηγορηματική’ (explicit) σημαίνει ότι το είδος των εννοιών που χρησιμοποιούνται, και οι περιορισμοί που αφορούν την χρήση αυτών των εννοιών είναι προσδιορισμένα με σαφήνεια. Ο όρος ‘αυστηρή’ (formal) αναφέρεται στο ότι η οντολογία πρέπει να είναι μηχανικά αναγνώσιμη. Ο όρος ‘διαμοιρασμένη’ (shared) αναφέρεται στο ότι η οντολογία πρέπει να αποτυπώνει γνώση κοινής αποδοχής στα πλαίσια μιας κοινότητας.

2.4.2. Μορφή

Μια οντολογία μπορεί να πάρει διάφορες μορφές αλλά οπωσδήποτε θα περιλαμβάνει ένα λεξιλόγιο όρων και κάποιας μορφής προδιαγραφές για τη σημασία τους.
Σχετικά με τον βαθμό της τυπικότητας της αναπαράστασης μιας οντολογίας αυτή μπορεί να είναι:

- *Άτυπη* (informal), εκφρασμένη σε μια φυσική γλώσσα.
- *Ημι-άτυπη* (semi-informal): για παράδειγμα διατυπωμένη σε ένα περιορισμένο και δομημένο υποσύνολο κάποιας φυσικής γλώσσας.
- *Ημι-τυπική* (semi-formal): διατυπωμένη σε μια τεχνητή και αυστηρά ορισμένη γλώσσα.
- *Αυστηρά τυπική* (rigorously formal): ορισμοί όρων με αυστηρή σημασιολογία, θεωρήματα και αποδείξεις ιδιοτήτων όπως η ορθότητα (soundness) και η πληρότητα (completeness).

2.4.3. Τα βασικά συστατικά μιας Οντολογίας

Πέντε κατηγορίες συστατικών:

- **Κλάσεις (classes):** Έννοιες που σχετίζονται με ένα πεδίο ή κάποιες εργασίες, οι οποίες είναι συνήθως οργανωμένες σε κάποιο ταξινομικό σύστημα,
  Σε μια οντολογία που αφορά το πανεπιστήμιο: ο ‘φοιτητής’ και ο ‘καθηγητής’ αποτελούν δύο κλάσεις.

- **Σχέσεις (relations):** Ένας τύπος αλληλεπίδρασης μεταξύ εννοιών ενός πεδίου.
  όπως: subclass-of, is-a
- **Συναρτήσεις (functions)** μια ειδική περίπτωση σχέσης στην οποία το ν-οστό στοιχείο της σχέσης προσδιορίζεται μοναδικά από τα ν-1 προηγούμενα στοιχεία.
  παράδειγμα: Η τιμή-μεταχειρισμένου-αυτοκινήτου μπορεί να προσδιορίζεται σαν συνάρτηση της αρχικής τιμής του καινούριου αυτοκινήτου, του μοντέλου του αυτοκινήτου, των χαρακτηριστικών του αυτοκινήτου και των χιλιομέτρων που έχει διανύσει.
- **Αξιώματα (axioms)** αναπαριστούν προτάσεις που είναι πάντα αληθείς
  παράδειγμα: αν ο Φ είναι δευτεροετής φοιτητής τότε μπορεί να εγγραφεί στο επιλεγόμενο μάθημα Μ.
- **Στιγμιότυπα (instances)** αναπαριστούν συγκεκριμένα στοιχεία
  παράδειγμα: ο φοιτητής με το όνομα Νίκος είναι ένα στιγμιότυπο της κλάσης ‘φοιτητής’

2.4.4. Κατηγορίες Οντολογιών

Μερικές χαρακτηριστικές κατηγορίες οντολογιών είναι οι ακόλουθες:

- **Οντολογίες πεδίου ορισμού (domain ontologies)**: αναπαριστούν γνώση γύρω από ένα συγκεκριμένο πεδίο (π.χ. ιατρική, ηλεκτρονικά κλπ.).
- **Οντολογίες μεταδεδομένων (metadata ontologies):** παρέχουν ένα λεξιλόγιο για την περιγραφή του περιεχομένου ηλεκτρονικά διαθέσιμης πληροφορίας.
- **Γενικές ή κοινές οντολογίες (generic or common sense ontologies)**: στοχεύουν στο να αποτυπώσουν γενική γνώση γύρω από τον κόσμο, παρέχοντας βασικές έννοιες όπως ο χρόνος, ο χώρος, τα συμβάντα, κλπ.
- **Οντολογίες αναπαράστασης (representational ontologies):** παρέχουν οντότητες αναπαράστασης χωρίς να προσδιορίζουν τη συγκεκριμένο αναπαριστούν

π.χ. Frame Ontology (Gruber 1993): ορίζει έννοιες όπως frames, slots, slot constraints κ.λ.π.

- **Οντολογίες μεθοδολογίας ή εργασιών (method or task ontologies):** παρέχουν όρους που αναφέρονται σε συγκεκριμένες εργασίες (π.χ. διάγνωση κλπ.)

2.4.5. Μερικές Εφαρμογές των Οντολογιών

- Επικοινωνία μεταξύ ανθρώπων και οργανισμών.
  Παρέχουν ολοκληρωμένο πλαίσιο εννοιών και ορολογίας μεταξύ ανθρώπων με διαφορετικές ανάγκες και οπτικές γωνίες στα πλαίσια ενός οργανισμού. Διευκολύνουν την επικοινωνία των ανθρώπων στα πλαίσια του οργανισμού.
- Δια-λειτουργικότητα (inter-operability) μεταξύ συστημάτων
  Διάφοροι χρήστες χρειάζεται να ανταλλάσσουν δεδομένα ή χρησιμοποιούν διαφορετικά πακέτα λογισμικού. Επίσης χρησιμοποιούνται για την υποστήριξη μετάφρασης μεταξύ διαφορετικών γλωσσών και αναπαραστάσεων.
- Μηχανική Συστημάτων (systems engineering)

2.4.6. Μεθοδολογία Ανάπτυξης Οντολογιών

Δεν υπάρχουν τυποποιημένες μεθοδολογίες για τη ανάπτυξη οντολογιών παρά μόνο εμπειρικοί κανόνες.

Στην εργασία M. Uschold & M. Gruninger 1996) προτείνονται οι ακόλουθες φάσεις για την ανάπτυξη οντολογιών:

- Προσδιορισμός σκοπιμότητας και πεδίου εφαρμογής

  Σημαντικό να ξεκαθαριστεί το γιατί να κατασκευαστεί η οντολογία και ποιες είναι οι πιθανές χρήσεις της. Χρήσιμο είναι επίσης να προσδιοριστεί ποιοι πρόκειται να είναι οι πιθανοί χρήστες της οντολογίας.

- Κατασκευή της οντολογίας
  - Σύλληψη (capture)
    Προσδιορισμός των βασικών εννοιών και των μεταξύ τους σχέσεων.
    Παραγωγή σαφών προδιαγραφούν σε μορφή κειμένου αυτών των εννοιών και των μεταξύ τους σχέσεων.
    Συμφωνία για τους όρους με τους οποίους θα αναφερόμαστε στις έννοιες και σχέσεις.
  - Κωδικοποίηση (coding)
    Ρητή αναπαράσταση της σύλληψης του προηγούμενου σταδίου σε μια τυπική γλώσσα.
  - Ενοποίηση (integration) υπαρχουσών οντολογιών.
    Θα πρέπει (και πως) να χρησιμοποιηθούν υπάρχουσες οντολογίες (ή τμήματα αυτών); Γενικά δύσκολο πρόβλημα.
- Αξιολόγηση (evaluation)
  Έκφραση τεχνικών κρίσεων σχετικά με τις οντολογίες, το σχετιζόμενο με αυτές περιβάλλον λογισμικού, και τη τεκμηρίωση σε σχέση με ένα πλαίσιο αναφοράς
- Τεκμηρίωση (documentation)
  Όλες οι σημαντικές παραδοχές τόσο αναφορικά με τις βασικές έννοιες που ορίζονται στην οντολογία όσο και με τα βασικά δομικά στοιχεία που χρησιμοποιήθηκαν για την έκφραση αυτών των εννοιών στην οντολογία, πρέπει να τεκμηριωθούν.

2.4.7. Η Οντολογία OWL

Η OWL(Web Ontology Language) είναι η πιο πρόσφατη εξέλιξη στις γλώσσες οντολογιών για τον Σημασιολογικό ιστό. Έχει δεχτεί επιρροές από πολλές προϋπάρχουσες γλώσσες οντολογιών. Ακολουθεί το συντακτικό της RDF(S).
Υποστηρίζει

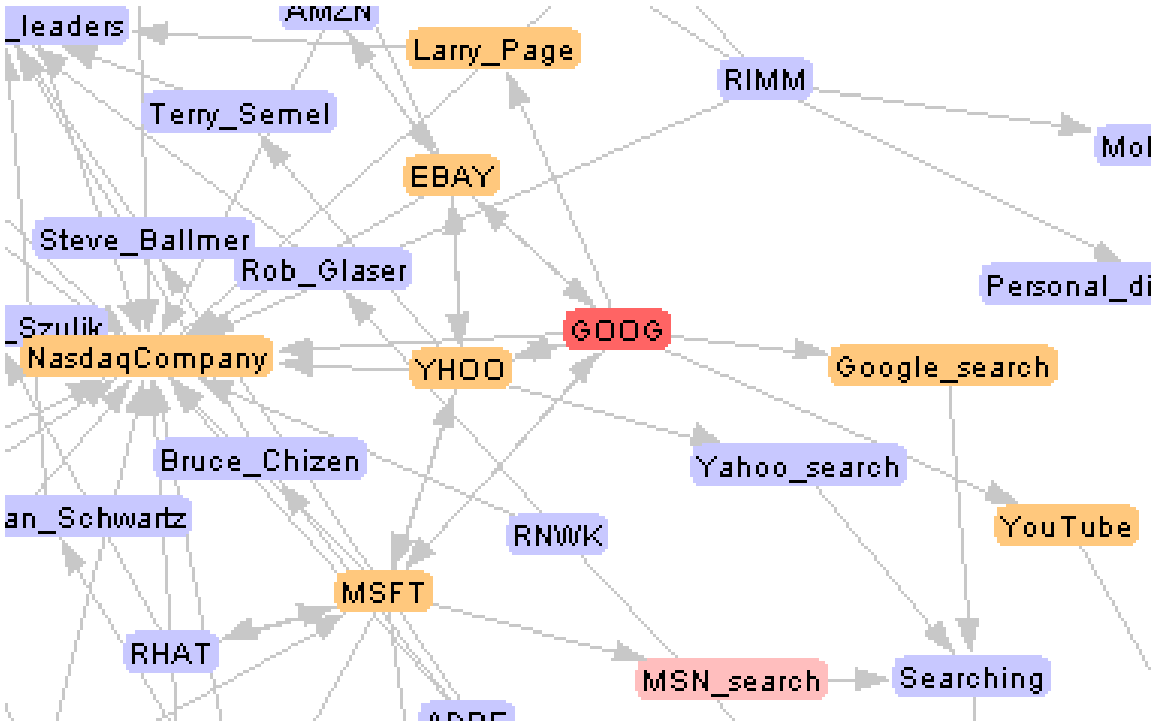
- Σχέσεις μεταξύ των κλάσεων.
- Πληθυσμότητα.
- Ισότητα.
- Περισσότερα χαρακτηριστικά των ιδιοτήτων.
- Κλάσεις απαρίθμησης(χρήση ονοματικών) .

Η OWL χωρίζεται σε επίπεδα λειτουργιών. Διαδοχικά επίπεδα της OWL είναι:

- OWL Lite :Έχει σχεδιαστεί για την έκφραση ιεραρχιών ταξινόμησης και απλών περιορισμών ιδιοτήτων.
- OWL DL :Υποστηρίζει τη μέγιστη δυνατή εκφραστικότητα , χωρίς απώλειες της αποφασισιμότητας.
- OWL Full :Προορίζεται για χρήστες που επιθυμούν μέγιστη εκφραστικότητα και την πλήρη έκφραση του RDF χωρίς όμως εγγυήσεις επιλυσιμότητας.

```
− <xsl:stylesheet version="1.0">
    <xsl:output method="xml" indent="yes" encoding="UTF-8"/>
  − <xsl:template match="/">
      <xsl:apply-templates select="//wsrr-classif:classificationTaxonomy"/>
    </xsl:template>
  − <xsl:template match="wsrr-classif:classificationTaxonomy">
    − <rdf:RDF>
      − <xsl:attribute name="xml:base">
          <xsl:value-of select="wsrr-classif:URI"/>
        </xsl:attribute>
      − <owl:Ontology rdf:about="">
        − <rdfs:label>
            <xsl:value-of select="wsrr-classif:name"/>
          </rdfs:label>
          <xsl:apply-templates select="wsrr-classif:nameLanguageVariants"/>
        − <rdfs:comment>
            <xsl:value-of select="wsrr-classif:comment"/>
          </rdfs:comment>
          <xsl:apply-templates select="wsrr-classif:commentLanguageVariants"/>
        </owl:Ontology>
        <xsl:apply-templates select="wsrr-classif:rootNodes"/>
      </rdf:RDF>
    </xsl:template>
  + <xsl:template match="wsrr-classif:rootNodes | wsrr-classif:childrenNodes"></xsl:template>
  + <xsl:template match="wsrr-classif:nameLanguageVariants"></xsl:template>
  + <xsl:template match="wsrr-classif:commentLanguageVariants"></xsl:template>
  </xsl:stylesheet>
```

Εικόνα 6: Παράδειγμα από κώδικα OWL.



Εικόνα 7: Παράδειγμα από γράφο OWL(Hermes project).

2.5. SPARQL(*SPARQL Protocol and RDF Query Language*)

Η SPARQL είναι Γλώσσα διατύπωσης ερωτήσεων RDF. Τυποποιείται από την *Ομάδα εργασίας πρόσβασης στοιχείων RDF(RDF Data Access Working Group*-DAWG) της World Wide Web Consortium. Θεωρείται , το κεντρικό πρότυπο για προσπέλαση δεδομένων στο Σημασιολογικό Ιστό.

Το W3C ανακοίνωσε στις 15 Ιανουαρίου 2008 την έκδοση της SPARQL. Με την τεχνολογία αναζήτησης SPARQL οι άνθρωποι μπορούν να επικεντρωθούν σε αυτό που θέλουν να ξέρουν αντί στην τεχνολογία βάσης δεδομένων ή στη μορφή δεδομένων που χρησιμοποιούνται παρασκηνιακά για την αποθήκευση δεδομένων. Επειδή οι αναζητήσεις SPARQL εκφράζουν στόχους υψηλού επιπέδου, είναι ευκολότερο να τις επεκτείνουμε σε μη αναμενόμενους πόρους δεδομένων, ή ακόμα να τις εισάγουμε σε νέες εφαρμογές.

Το να προσπαθήσουμε να χρησιμοποιήσουμε το Σημασιολογικό Ιστό χωρίς τη SPARQL είναι σαν να προσπαθούμε να χρησιμοποιήσουμε μια σχεσιακή βάση δεδομένων χωρίς SQL.Η SPARQL δίνει τη δυνατότητα αναζήτησης πληροφοριών από βάσεις δεδομένων και από άλλους, ποικίλους πόρους, στον Παγκόσμιο Ιστό.

Υπάρχουν ήδη 14 γνωστές εφαρμογές της SPARQL, πολλές από τις οποίες είναι ανοιχτού κώδικα.

Η SPARQL Προσπερνάει τους Παραδοσιακούς Περιορισμούς των Τοπικών, Μοναδικών Μορφών Αναζήτησης

Υπάρχουν πολλές επιτυχείς γλώσσες αναζήτησης, συμπεριλαμβανομένων προτύπων όπως SQL και XQuery. Αυτά είχαν σχεδιαστεί αρχικά για αναζητήσεις περιορισμένες σε ένα προϊόν, μια μορφή, τύπο πληροφοριών ή σε τοπικά αποθηκευμένα δεδομένα. Παραδοσιακά, ήταν απαραίτητο να δημιουργηθεί η ίδια αναζήτηση υψηλού επιπέδου διαφορετικά, ανάλογα με την εφαρμογή ή τη συγκεκριμένη διάταξη που επιλέγεται για τη βάση δεδομένων. Και όταν αναζητούνται πολλοί πόροι δεδομένων ήταν απαραίτητο να γραφτεί λογική για το συνδυασμό των αποτελεσμάτων. Αυτοί οι περιορισμοί έχουν επιφέρει υψηλότερο κόστος ανάπτυξης και δημιούργησαν φραγμούς για την ενσωμάτωση νέων πόρων.

Ο στόχος του Σημασιολογικού Ιστού είναι να δώσει τη δυνατότητα στους ανθρώπους να μοιραστούν, να ενσωματώσουν και να επαναχρησιμοποιήσουν τα δεδομένα παγκοσμίως. Η SPARQL έχει σχεδιαστεί για χρήση στον Παγκόσμιο Ιστό και επιτρέπει τις αναζητήσεις σε κατανεμημένους πόρους δεδομένων, ανεξαρτήτως μορφής. Δημιουργώντας μια μοναδική αναζήτηση σε εύρος δεδομένων, είναι ευκολότερο από το να υπάρχουν πολλές αναζητήσεις. Επίσης, στοιχίζει λιγότερο και παρέχει πλουσιότερα αποτελέσματα.
Επειδή η SPARQL δεν έχει περιορισμούς για συγκεκριμένη μορφή βάσης δεδομένων, μπορεί να χρησιμοποιηθεί από το κύμα δεδομένων του Web 2.0 και να το συνδυάσει με άλλους πόρους του Σημασιολογικού Ιστού. Ακόμα, επειδή οι απομακρυσμένοι πόροι μπορεί να μην έχουν το ίδιο 'σχήμα' ή να μοιράζονται τις ίδιες ιδιότητες, η SPARQL έχει σχεδιαστεί για αναζήτηση μη ομοιόμορφων δεδομένων.
Η SPARQL Μετατρέπει την Πρόσβαση Δεδομένων σε Υπηρεσία του Παγκοσμίου Ιστού
Ο συνδυασμός της γλώσσας αναζήτησης SPARQL και του πρωτοκόλλου δημιουργεί μια υπηρεσία στον Παγκόσμιο Ιστό που τρέχει πάνω σε HTTP ή SOAP, παρέχει μια πρότυπη υπηρεσία στον Παγκόσμιο Ιστό για οτιδήποτε ρωτάει μια ερώτηση.
Η SPARQL εστιάζει στην αναζήτηση μοντέλων δεδομένων κι αυτό γλιτώνει χρόνο για τους κατασκευαστές. Δεν υπάρχει ανάγκη για μικρές υπηρεσίες του Παγκοσμίου Ιστού για να βρίσκουν διαφορετικές πλευρές του συστήματος. Αυτό επιτρέπει στο χρήστη της SPARQL να κάνει οποιαδήποτε ερώτηση -- είναι σαν να σχεδίαζαν τη δική τους διεπαφή, αντί να πρέπει να δουλέψουν με ένα περιορισμένο σύνολο προκατασκευασμένων υπηρεσιών.
Η προδιαγραφή SPARQL ορίζει μια γλώσσα αναζήτησης και ένα πρωτόκολλο και συνεργάζεται με άλλες κεντρικές τεχνολογίες του Σημασιολογικού Ιστού από το W3C, τις: Resource Description Framework (RDF) για αναπαράσταση δεδομένων, RDF Schema, Web Ontology Language (OWL) για τη δημιουργία λεξιλογίων και Gleaning Resource Descriptions from Dialects of Languages (GRDDL), για αυτόματη εξαγωγή δεδομένων Σημασιολογικού Ιστού από έγγραφα. Η SPARQL ακόμα χρησιμοποιεί άλλα πρότυπα του W3C που βρίσκονται σε

εφαρμογές του Παγκοσμίου Ιστού όπως τη γλώσσα Web Services Description Language (WSDL).

2.5.1. Συντακτικό SPARQL

Το λεξιλόγιο είναι βασισμένο στη σύνταξη SPIN SPARQL. Το SPIN Modeling Vocabulary καθορίζει ένα ελαφρύ σύνολο κατηγοριών RDFS και ιδιοτήτων που μπορεί να χρησιμοποιηθεί για να ενσωματώσει συστηματικά τις ερωτήσεις SPARQL στα πρότυπα RDF έτσι ώστε μπορούν να εκτελεσθούν με την καθορισμένη με σαφήνεια σημασιολογία. Η βασική ιδέα είναι να χρησιμοποιηθούν οι συγκεκριμένες ιδιότητες RDF για να συνδέσει τις κατηγορίες με τις ερωτήσεις SPARQL έτσι ώστε εκείνες οι ερωτήσεις SPARQL μπορούν να εκτελεσθούν με ένα δεδομένο πλαίσιο. Στο SPARQL, η πρώτη γραμμή της ερώτησης καθορίζει απλά ένα πρόθεμα για το FOAF, έτσι ώστε εσείς δεν χρειάζεται να το δακτυλογραφήσετε κάθε φορά που είναι παραπεφθέν.

Η πρόταση **SELECT** διευκρινίζει τι η ερώτηση πρέπει να επιστρέψει. Οι μεταβλητές **SPARQL** προτάσσονται με καθεμία με ? ή $ και τα δύο είναι ανταλλάξιμα.
Η **FROM** είναι μια προαιρετική πρόταση που παρέχει το URI του συνόλου δεδομένων στη χρήση. Μπορεί να δείξει ένα τοπικό αρχείο, αλλά αυτό θα μπορούσε επίσης να δείξει το URL μιας γραφικής παράστασης κάπου στον Ιστό.
Η πρόταση **WHERE** αποτελείται από μια σειρά triple patterns, που χρησιμοποιούν της σύνταξη Turtle-based. Αυτά τα triples περιλαμβάνουν μαζί αυτό που είναι γνωστό ως σχέδιο γραφικών παραστάσεων.
Η λέξη κλειδί **FILTER** σε SPARQL περιορίζει τα αποτελέσματα μιας ερώτησης με την επιβολή των περιορισμών στις τιμές των συνδεδεμένων μεταβλητών.
Η **OPTIONAL** πρόταση καθορίζει τα πρόσθετα σχέδια γραφικών παραστάσεων που δεν αναγκάζουν τις λύσεις για να απορριφθούν εάν δεν αντιστοιχούνται, αλλά δεσμεύουν στη γραφική παράσταση όταν μπορούν να αντιστοιχηθούν. Για να καθαρίσει περαιτέρω τα

αποτελέσματα μιας ερώτησης, SPARQL έχει τις λέξεις κλειδιά **DISTINCT**, **LIMIT**, **OFFSET**, και **ORDER BY** που λειτουργούν λίγο πολύ όπως τα αντίστοιχα SQL τους.
Η **DISTINCT** μπορεί να χρησιμοποιηθεί μόνο με τις SELECT ερωτήσεις, ενώ τοποθετούνται οι άλλες λέξεις κλειδιά όλες μετά από την πρόταση WHERE μιας ερώτησης.
Η **LIMIT n** τον αριθμό n αποτελεσμάτων που επιστρέφονται από έναν τόνο ερώτησης, ενώ η **OFFSET n** παραλείπει τα πρώτα n αποτελέσματα. **ORDER BY ? VAR** θα ταξινομήσει τα αποτελέσματα με τη φυσική τάξη. Για παράδειγμα: ASC [? VAR] και DESC [? VAR] μπορεί να χρησιμοποιηθεί για να διευκρινίσει την κατεύθυνση του είδους. Φυσικά είναι δυνατό να συνδυαστεί η DINSTICT, η LIMIT, η OFFSET και η ORDER BY σε μια ερώτηση. Για παράδειγμα: **ORDER BY DESC [;date] LIMIT 10** θα μπορούσε να χρησιμοποιηθεί για να βρει τις δέκα πιο πρόσφατες καταχωρήσεις.

2.5.2. Παράδειγμα

Εδώ είναι ένα παράδειγμα σε μια υποτιθέμενη οντολογία. Η ακόλουθη απλή ερώτηση SPARQL είναι «Ποιες είναι οι πρωτεύουσες των Αφρικανικών χωρών;»

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
    abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
    abc:isInContinent abc:Africa .
}
```

Οι μεταβλητές υποδεικνύονται από «?» ή πρόθεμα «$».Θα δώσει αποτέλεσμα σε ένα πινάκα την χώρα από τη μια πλευρά και αντίστοιχα την πρωτεύουσα της από την άλλη.

Ο επεξεργαστής ερώτησης SPARQL θα ψάξει για τα σύνολα των triples που ταιριάζουν με αυτά τα τέσσερα triple-patterns, που δεσμεύουν τις μεταβλητές στην ερώτηση στα αντίστοιχα μέρη κάθε triple.

3. Πως δημιουργήσαμε το GUI

Για να δημιουργηθεί μια σημασιολογική εφαρμογή απαιτείται να δημιουργηθεί μια οντολογία με τα κατάλληλα εργαλεία.

3.1. Protégé

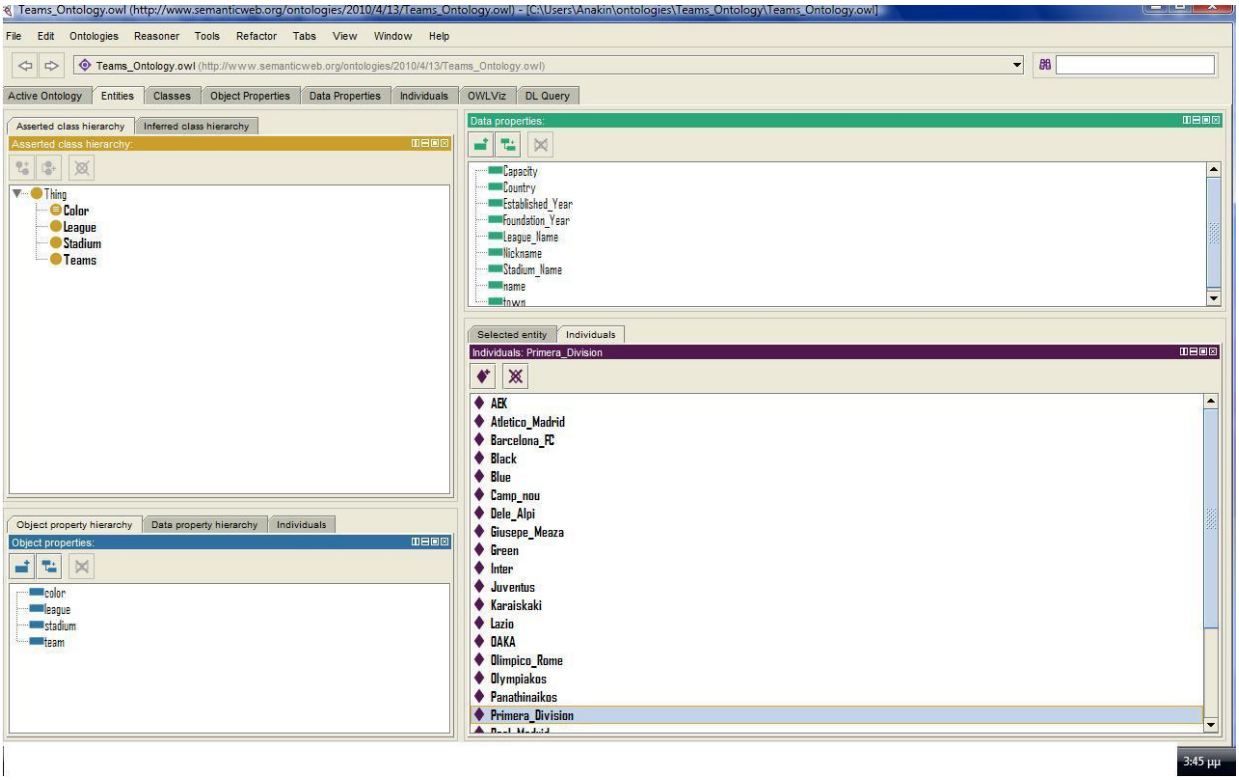
Με το πρόγραμμα Protégé δημιουργήσαμε την οντολογία.

Το Protégé-OWL editor είναι μια επέκταση του Protégé που υποστηρίζει την οντολογική γλώσσα (OWL). OWL είναι το πιο πρόσφατο επίτευγμα στα πρότυπα των οντολογικών γλωσσών, που εγκρίθηκε από το World Wide Web Consortium (W3C) για την προώθηση του οράματος του *Σημασιολογικού Ιστού* . "Μια OWL οντολογία μπορεί να περιλαμβάνει περιγραφές των κλάσεων, των ιδιοτήτων και των περιπτώσεων τους. Λόγω αυτής της οντολογίας, το OWL τυπική σημασιολογία καθορίζει πώς να αντλήσει τις συνέπειες της λογικής, δηλαδή, πραγματικά περιστατικά, δεν κυριολεκτικά υπάρχει στην οντολογία, αλλά που προκύπτουν από τη σημασιολογία. Αυτά τα περιστατικά μπορεί να βασίζονται σε ένα ενιαίο έγγραφο ή πολλαπλές διανεμηθεί έγγραφα που έχουν συνδυαστεί χρησιμοποιώντας ορίζεται OWL μηχανισμών "

Το Protégé-OWL editor δίνει τη δυνατότητα στους χρήστες να:

- Φορτώσουν και να αποθηκεύσουν OWL και RDF οντολογίες.
- Επεξεργαστούν και να απεικονίσουν τις κατηγορίες, τις ιδιότητες, και SWRL κανόνες.
- Ορίσουν λογικά χαρακτηριστικά κατηγορίας με OWL εκφράσεις.
- Εκτελέσουν reasoners όπως ταξινομητές λογικής περιγραφής.
- Επεξεργασία OWL για markup Σημασιολογικού Ιστού.

Η OWL είναι ευέλικτη αρχιτεκτονική του Protégé -καθιστά εύκολο να ρυθμίσετε και να επεκτείνουν το εργαλείο. Protégé-OWL είναι σφιχτά ενσωματωμένη με Jena Framework και έχει μια ανοικτού κώδικα Java API για την ανάπτυξη κατάλληλα προσαρμοσμένου στοιχείων του περιβάλλοντος εργασίας χρήστη ή της αυθαίρετης υπηρεσίας του Σημασιολογικού Ιστού.



Εικόνα 8: Κατά τη διάρκεια σχεδιασμού της οντολογίας με το protégé.

3.2. Twinkle

Το πρόγραμμα twinkle μας βοήθησε πολύ για να ελέγξουμε την ορθότητα των αποτελεσμάτων της SPARQL.

Twinkle είναι μια απλή διεπαφή GUI που χρησιμοποιεί τη μηχανή αναζήτησης ARQ SPARQL . Το εργαλείο πρέπει να είναι χρήσιμο τόσο για τους ανθρώπους που θέλουν να μάθουν τη γλώσσα SPARQL, καθώς και εκείνες που δουλεύουν πάνω στην ανάπτυξη του Σημασιολογικού Ιστού.

Χαρακτηριστικά

- Φορτίο, να επεξεργαστείτε και να αποθηκεύσετε SPARQL ερωτήματα
- Τοποθετήστε δηλώσεις PREFIX σε ερωτήματα ,ρύθμιση namespaces, ώστε να μπορούν γρήγορα να προστεθούν στο ερωτημάτων
- Άκυρο καιρό ερωτήματα λειτουργίας
- Αποθηκεύσετε τα αποτελέσματα σε αρχείο
- Συμπλέκει τοπικά αρχεία και απομακρυσμένα έγγραφα RDF
- Δεδομένα ερωτήματος RDF που πραγματοποιήθηκε σε σχεσιακές βάσεις δεδομένων
- Συμπλέκει online τελικά σημεία SPARQL, όπως DBpedia, reyvu.com και GovTrack.
- Θέτει ερωτήματα με βάση το τυπικό SPARQL, ή το ARQ επεκταθεί σύνταξη που υποστηρίζει COUNT, κλπ.
- Χρησιμοποιήστε ARQ λειτουργίες επέκταση και τις λειτουργίες ακινήτων
- Εφαρμόστε inferencing (π.χ. κανόνες Jena, RDF Schema, OWL οντολογία) κατά την εκτέλεση ερωτημάτων
- Ρυθμίστε τις παραμέτρους που χρησιμοποιούνται συνήθως στις πηγές δεδομένων για γρήγορη πρόσβαση

3.3. Το Jena RDF API

3.3.1. Το Jena API

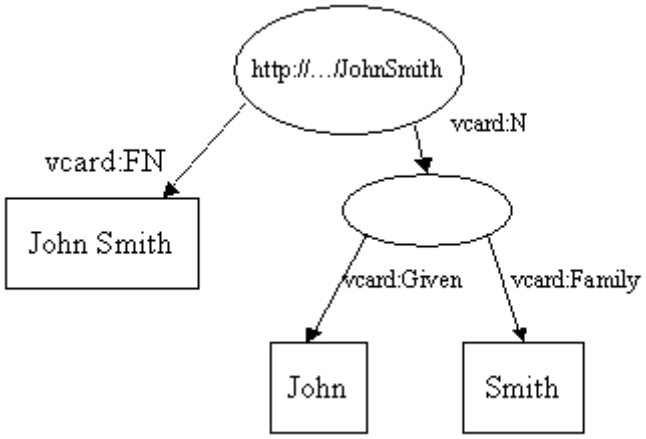
Το Jena είναι ένα Java API που μπορεί να χρησιμοποιηθεί για τη δημιουργία και διαχείριση RDF γραφημάτων. Παρέχει κλάσεις για την αναπαράσταση γραφημάτων, πόρων, ιδιοτήτων και σταθερών(literals). Οι διεπαφές που αναπαριστούν τους πόρους ,τις ιδιότητες και τις σταθερές ονομάζονται Resource, Property και Literal αντίστοιχα.
Στο Jena, ένα γράφημα καλείται “μοντέλο - model” και αναπαρίσταται από τη διεπαφή “Model “.
Μπορούμε να δημιουργήσουμε το παραπάνω γράφημα με τον εξής κώδικα:

```
// some definitions
        static String personURI = "http://somewhere/JohnSmith";
        static String fullName = "John Smith";
// create an empty Model
        Model model = ModelFactory.createDefaultModel();
// create the resource
        Resource johnSmith = model.createResource(personURI);
// add the property
        johnSmith.addProperty(VCARD.FN, fullName);
```

Ξεκινάμε με κάποιους ορισμούς σταθερών, και κατόπιν δημιουργούμε ένα κενό μοντέλο, χρησιμοποιώντας τη μέθοδο createDefaultModel() της κλάσης ModelFactory. Το Jena περιέχει και άλλες υλοποιήσεις της διεπαφής Model. Μια από αυτές, την ModelRDB θα χρησιμοποιήσουμε και στην ανάπτυξη της εφαρμογής και που χρησιμοποιεί μια σχεσιακή βάση δεδομένων.

Στη συνέχεια δημιουργείται ο πόρος “John Smith “, και προστίθεται μια ιδιότητα σε αυτόν. Η ιδιότητα παρέχεται από μια σταθερή(constant) κλάση VCARD η οποία περιέχει αντικείμενα που αναπαριστούν όλους τους ορισμούς του VCARD σχήματος. Το Jena παρέχει σταθερές κλάσεις και για άλλα γνωστά σχήματα όπως είναι το ίδιο το RDF , το Dublin Core και το DAML.

Ακολουθεί ένα πιο σύνθετο παράδειγμα για να δούμε πως μπορούμε να ορίσουμε έναν πόρο σαν τιμή μιας ιδιότητας, αλλά και πως να χρησιμοποιήσουμε έναν κενό κόμβο(blank node).



Εικόνα 9: RDF γράφημα με διάφορες ιδιότητες του πόρου “http://..../JohnSmith”

Εδώ προσθέσαμε μια νέα ιδιότητα, την vcard:N, για να αναπαραστήσουμε το δομή του ονόματος του John Smith. Η vcard:N δέχεται έναν πόρο σαν τιμή. Επίσης χρησιμοποιήσαμε έναν κενό κόμβο για την αναπαράσταση του σύνθετου ονόματος.
Ο κώδικας γι'αυτό το παράδειγμα είναι:

```
// some definitions
        String personURI = "http://somewhere/JohnSmith";
        String givenName = "John";
        String familyName = "Smith";
        String fullName = givenName + " " + familyName;
// create an empty Model
        Model model = ModelFactory.createDefaultModel();
// create the resource
// and add the properties cascading style
        Resource johnSmith
        = model.createResource(personURI)
        .addProperty(VCARD.FN, fullName)
        .addProperty(VCARD.N,
        model.createResource()
        .addProperty(VCARD.Given, givenName)
        .addProperty(VCARD.Family, familyName));
```

3.3.2. Δηλώσεις (Statements)

Ένα RDF μοντέλο αναπαρίσταται ως ένα σύνολο δηλώσεων. Με κάθε κλήση της μεθόδου addProperty προσθέτουμε μια δήλωση στο μοντέλο. Το Jena ορίζει μια μέθοδο listStatements() που επιστρέφει ένα αντικείμενο StmtIterator. Η κλάση StmtIterator επεκτείνει την κλάση Iterator και επιστρέφει όλες τις δηλώσεις ενός μοντέλου. Περιέχει μια μέθοδο nextStatement()που επιστρέφει την επόμενη δήλωση από τον iterator. Η διεπαφή Statement παρέχει μεθόδους πρόσβασης στο υποκείμενο, κατηγόρημα και αντικείμενο μιας δήλωσης. Χρησιμοποιώντας αυτή τη διεπαφή μπορούμε να τυπώσουμε όλες τις δηλώσεις του μοντέλου του προηγούμενου παραδείγματος:

```
// list the statements in the Model
        StmtIterator iter = model.listStatements();
// print out the predicate, subject and object of each statement
        while (iter.hasNext()) {
        Statement stmt = iter.nextStatement(); // get next statement
```

```
        Resource subject = stmt.getSubject(); // get the subject
        Property predicate = stmt.getPredicate(); // get the predicate
        RDFNode object = stmt.getObject(); // get the object
        System.out.print(subject.toString());
        System.out.print(" " + predicate.toString() + " ");
        if (object instanceof Resource) {
        System.out.print(object.toString());
        } else {
// object is a literal
        System.out.print(" \"" + object.toString() + "\"");
        }
        System.out.println(" .");
        }
```

Αφού το αντικείμενο μιας δήλωσης μπορεί να είναι είτε ένας πόρος είτε μία σταθερά, η μέθοδος getObject() επιστρέφει ένα αντικείμενο τύπου RDFNode, που είναι μια υπερκλάση των Resource και Literal. Ο κώδικας χρησιμοποιεί το instanceof για τον προσδιορισμό του σωστού τύπου και τον επεξεργάζεται αναλόγως.

Η έξοδος αυτού του προγράμματος θα μοιάζει κάπως έτσι:

```
http://somewhere/JohnSmith http://www.w3.org/2001/vcard-rdf/3.0#N
anon:14df86:ecc3dee17b:-7fff .
anon:14df86:ecc3dee17b:-7fff http://www.w3.org/2001/vcardrdf/
3.0#Family "Smith" .
anon:14df86:ecc3dee17b:-7fff http://www.w3.org/2001/vcard-rdf/3.0#Given
"John" .
http://somewhere/JohnSmith http://www.w3.org/2001/vcard-rdf/3.0#FN
"John Smith" .
```

Το "anon:14df86:ecc3dee17b:-7fff" είναι ένας εσωτερικό αναγνωριστικό που χρησιμοποιείται από το Jena, αφού ο κενός κόμβος δεν έχει κάποιο URI.

3.3.3. Ανάγνωση και εγγραφή RDF

Το Jena έχει μεθόδους για την ανάγνωση και γραφή RDF σε XML μορφή, που μπορούν να χρησιμοποιηθούν για το γράψιμο του μοντέλου σε ένα αρχείου και την ανάγνωσή του αργότερα.

i. Εγγραφή RDF

Το προηγούμενο παράδειγμα τύπωσε το μοντέλο σε μορφή τριπλετών. Μπορούμε να το τυπώσουμε σε μορφή RDF XML στο καθιερωμένο ρεύμα εξόδου(STDOUT), περνώντας ένα

OutputStream σαν παράμετρο στη μέθοδο model.write:

model.write(System.out);

Το αποτέλεσμα θα είναι:

```
<rdf:RDF
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
>
<rdf:Description rdf:about='http://somewhere/JohnSmith'>
<vcard:FN>John Smith</vcard:FN>
<vcard:N rdf:nodeID="A0"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A0">
<vcard:Given>John</vcard:Given>
<vcard:Family>Smith</vcard:Family>
</rdf:Description>
</rdf:RDF>
```

Βλέπουμε ότι δόθηκε ένα αυθαίρετο URI στον κενό κόμβο ώστε αυτός να μπορεί να αναπαρασταθεί με σύνταξη RDF/XML .

Επίσης μπορούμε να καθορίσουμε τη μορφή με την οποία θα γραφτεί το μοντέλο περνώντας το ανάλογο όρισμα στη μέθοδο write() :

model.write(System.out, "RDF/XML-ABBREV");

model.write(System.out, "N-TRIPLE");

ii. Ανάγνωση RDF

Υποθέτοντας ότι υπάρχει ένα μοντέλο RDF σε ένα αρχείο με το όνομα inputFileName, μπορούμε να το διαβάσουμε με τον ακόλουθο κώδικα:

```
// create an empty model
    Model model = ModelFactory.createDefaultModel();
// use the FileManager to find the input file
    InputStream in = FileManager.get().open( inputFileName );
    if (in == null) {
    throw new IllegalArgumentException(
    "File: " + inputFileName + " not
    found");
    }
// read the RDF/XML file
    model.read(in, "");
```

Το δεύτερο όρισμα στην κλήση της μεθόδου read() είναι το URI που θα χρησιμοποιηθεί για την επίλυση σχετικών URI's. Αν δεν υπάρχουν URI αναφορές (URIrefs) στο αρχείο, αυτό μπορεί να είναι κενό.

3.3.4. Εργασίες πάνω σε ένα μοντέλο

i. Περιήγηση σε ένα μοντέλο

Έχοντας το URI ενός πόρου, το αντικείμενο του πόρου που εκπροσωπεί μπορεί να ανακτηθεί από το μοντέλο χρησιμοποιώντας τη μέθοδο Model.getResource(String uri) .
Αυτή η μέθοδος επιστρέφει ένα αντικείμενο τύπου Resource εάν υπάρχει κάποιο στο μοντέλο, αλλιώς δημιουργεί ένα νέο. πχ:
Resource vcard = model.getResource(johnSmithURI);
Η διεπαφή Resource ορίζει ένα πλήθος μεθόδων για την πρόσβαση στις ιδιότητες ενός πόρου. Η μέθοδος Resource.getProperty(Property p)παρέχει πρόσβαση στην ιδιότητα ενός πόρου, επιστρέφοντας όμως ένα αντικείμενο τύπου Statement και όχι Property όπως κάποιος θα περίμενε. Η επιστροφή ολόκληρης της δήλωσης επιτρέπει στην εφαρμογή να προσπελάσει την τιμή μιας ιδιότητας κάνοντας χρήση μια εκ των μεθόδων πρόσβασής της που επιστρέφει το αντικείμενο της δήλωσης. Για παράδειγμα, για την ανάκτηση του πόρου ο οποίος αποτελεί την τιμή της ιδιότητας vcard:N θα γράψουμε:
Resource name = (Resource) vcard.getProperty(VCARD.N).getObject();
Γενικά, το αντικείμενο μιας δήλωσης μπορεί να είναι ένας πόρος ή μια σταθερά, έτσι η εφαρμογή ξέροντας ότι η τιμή είναι ένας πόρος, κάνει την κατάλληλη μετατροπή (casting) στο αντικείμενο που επιστρέφεται. Για την αποφυγή τέτοιων μετατροπών, το Jena παρέχει μεθόδους που επιστρέφουν αντικείμενα συγκεκριμένου τύπου. Έτσι το παραπάνω κομμάτι κώδικα μπορεί πιο βολικά να γραφεί:
Resource name = vcard.getProperty(VCARD.N).getResource();
Με παρόμοιο τρόπο, ανακτάται και μια σταθερή τιμή μιας ιδιότητας:
String fullName = vcard.getProperty(VCARD.FN).getString();
Σε αυτό το παράδειγμα, ο πόρος έχει μόνο μια ιδιότητα vcard:FN και μια ιδιότητα vcard:N . Το RDF όμως επιτρέπει την επανάληψη μιας ιδιότητας σε έναν πόρο. Για παράδειγμα ο Adam θα μπορούσε να είχε δύο nicknames:
vcard.addProperty(VCARD.NICKNAME,"Smithy").addProperty(VCARD.NICKNAME,"Adman");

Το Jena δεν ορίζει ποιό από τα δύο nicknames του μοντέλου θα επιστραφεί καλώντας την vcard.getProperty(VCARD.NICKNAME). Λύση σε αυτό το πρόβλημα δίνει η μέθοδοςResource.listProperties(Property p) που επιστρέφει έναν iterator με αντικείμενα τύπου Statement που μπορούμε να χρησιμοποιήσουμε για να τα εμφανίσουμε όλα.

```
System.out.println("The nicknames of \""+ fullName + "\" are:");
StmtIterator iter = vcard.listProperties(VCARD.NICKNAME);
while (iter.hasNext()) {
System.out.println(" " +
iter.nextStatement().getObject().toString());
}
```

Ο iterator iter παράγει όλες τις δηλώσεις με υποκείμενο vcard και κατηγόρημα VCARD.NICKNAME, και μπορούμε να παίρνουμε μία μία τις δηλώσεις με την nextStatement(), και από κάθε τέτοια δήλωση να πάρουμε στη συνέχεια το αντικείμενο.
Τέλος, μπορούμε να δούμε όλες τις ιδιότητες ενός πόρου χρησιμοποιώντας τη μέθοδο listProperties() .

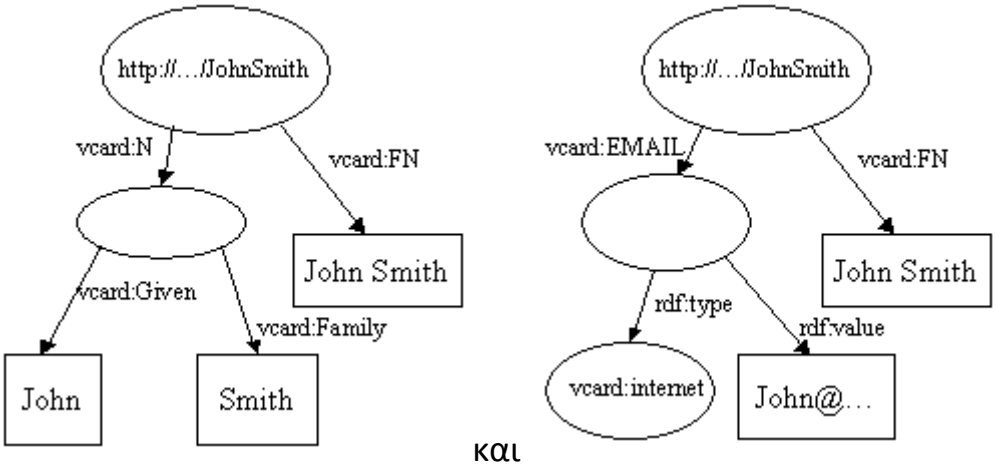
ii. Εκτελώντας αιτήματα σε ένα μοντέλο

Το κυρίως API του Jena υποστηρίζει ένα περιορισμένο τρόπο αιτημάτων.
Η μέθοδος Model.listStatements() τυπώνει όλες τις δηλώσεις του μοντέλου, αλλά βέβαιη χρήση της δεν ενδείκνυται σε μεγάλου μεγέθους μοντέλα. Η μέθοδος Model.listSubjects()είναι παρόμοια, μόνο που επιστρέφει έναν iterator πάνω σε όλους τους πόρους που έχουν ιδιότητες, δηλαδή αποτελούν υποκείμενα σε κάποια δήλωση.
Η μέθοδος Model.listSubjectsWithProperty(Property p, RDFNode o) επιστρέφει έναν iterator πάνω σε όλους τους πόρους που έχουν ιδιότητα p με τιμή o. Όλες οι προηγούμενες μέθοδοι αιτημάτων βασίζονται στη μέθοδο model.listStatements(Selector s). Αυτή η μέθοδος επιστρέφει ένα iterator σε όλες τις δηλώσεις που επιλέγονται από το 's'. Η διεπαφή selector είναι σχεδιασμένη να μπορεί να επεκταθεί, αλλά προς το παρόν η κλάση SimpleSelector είναι η μοναδική υλοποίησή της. Ο δομητής της κλάσης SimpleSelector δέχεται τρεις παραμέτρους:
Selector selector = new SimpleSelector(subject, predicate, object)
Αυτός ο selector θα επιλέξει όλες τις δηλώσεις που έχουν υποκείμενο που ταιριάζει με το subject, κατηγόρημα που ταιριάζει με το predicate,

και αντικείμενο που ταιριάζει με το object. Η τιμή null σε οποιαδήποτε από τις τρεις παραμέτρους ταιριάζει σε οτιδήποτε. Δύο πόροι θεωρούνται ίσοι εφόσον έχουν ίδια ή αποτελούν τον ίδιο κενό κόμβο. Έτσι η παρακάτω εντολή
listStatements(S, P, O)
είναι ισάξια με την
listStatements(new SimpleSelector(S, P, O))

iii. Λειτουργίες πάνω σε μοντέλα

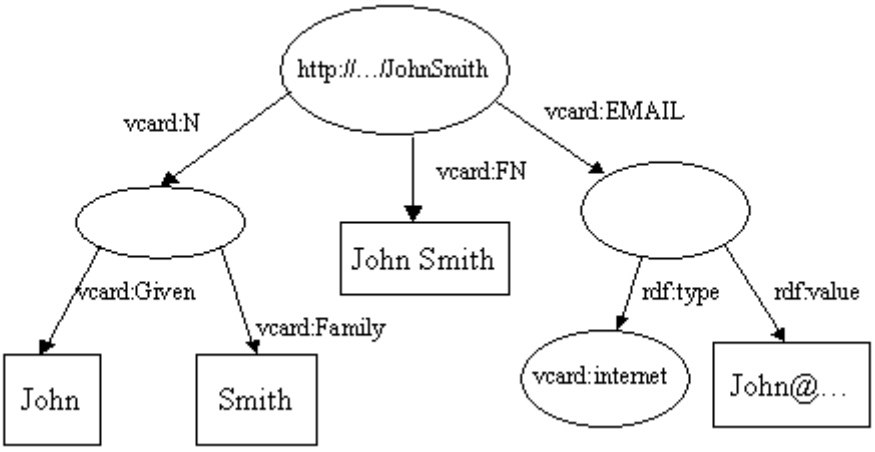
Το Jena παρέχει τρεις λειτουργίες για τη διαχείριση μοντέλων σε συνολικό επίπεδο. Αυτές είναι οι κοινές λειτουργίες της ένωσης, τομής και διαφοράς.
Η ένωση δύο μοντέλων είναι η ένωση των συνόλων των δηλώσεων που αντιπροσωπεύουν κάθε μοντέλο. Αυτή είναι μια από τις κύριες λειτουργίες που υποστηρίζει η σχεδίαση του RDF. Δίνει τη δυνατότητα σε δεδομένα προερχόμενα από διαφορετικές πηγές να συγχωνευτούν. Παίρνουμε σαν παράδειγμα τα δύο μοντέλα:



και

Εικόνα 10 και 11: Δύο RDF γραφήματα με διάφορες ιδιότητες του πόρου "http://..../JohnSmith"

Όταν αυτά συγχωνευτούν, οι δύο κόμβοι http://...JohnSmith συγχωνεύονται σε έναν και αφαιρείται η διπλή έλλειψη vcard:FN παράγοντας το επόμενο μοντέλο:



Εικόνα 12: Ένα RDF γράφημα από συγχώνευση δύο ξεχωριστών"

Αυτό μπορεί να γίνει με τον παρακάτω κώδικα:

```
// read the RDF/XML files
        model1.read(new InputStreamReader(in1), "");
        model2.read(new InputStreamReader(in2), "");
// merge the Models
        Model model = model1.union(model2);
// print the Model as RDF/XML
        model.write(system.out, "RDF/XML-ABBREV");
```

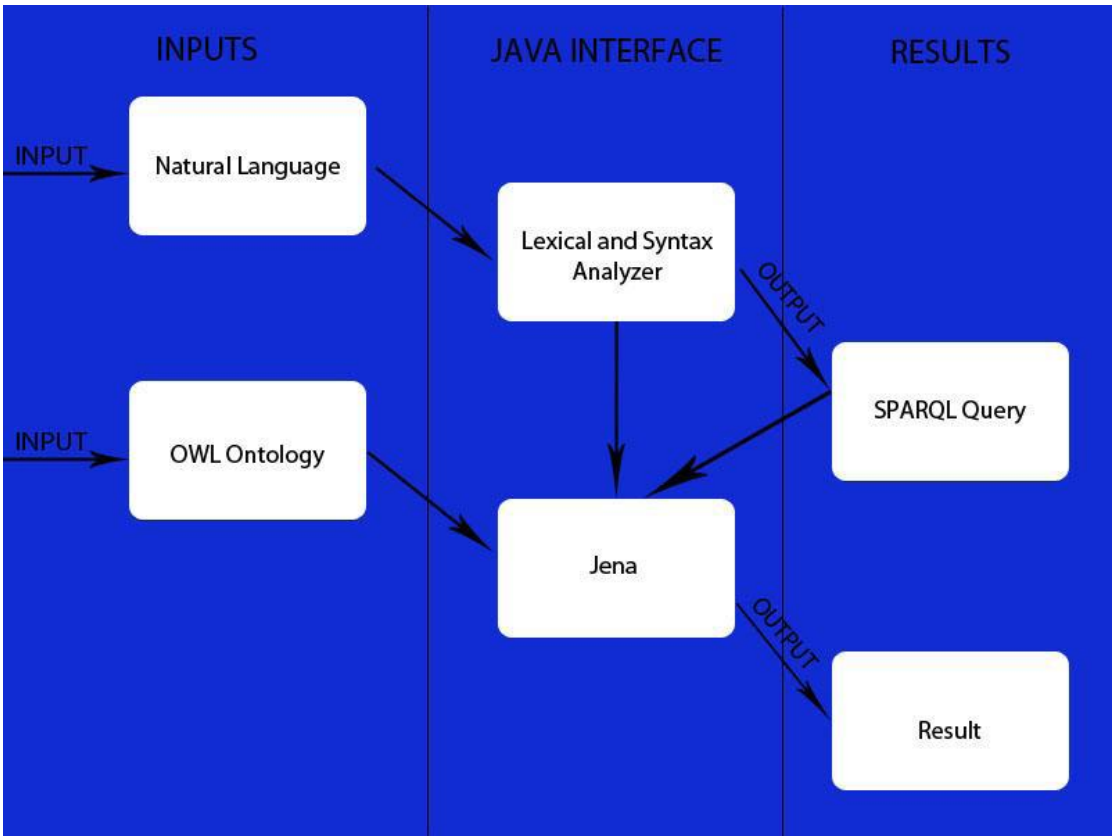
Στην οθόνη θα δούμε το παρακάτω αποτέλεσμα:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#">
<rdf:Description rdf:about="http://somewhere/JohnSmith/">
<vcard:EMAIL>
<vcard:internet>
<rdf:value>John@somewhere.com</rdf:value>
</vcard:internet>
</vcard:EMAIL>
<vcard:N rdf:parseType="Resource">
<vcard:Given>John</vcard:Given>
<vcard:Family>Smith</vcard:Family>
</vcard:N>
<vcard:FN>John Smith</vcard:FN>
</rdf:Description>
</rdf:RDF>
```

Με παρόμοιο τρόπο μπορεί να επιτευχθεί η τομή και η διαφορά των δύο μοντέλων, χρησιμοποιώντας τις μεθόδους .intersection(Model) και .difference(Model).

4. Αρχιτεκτονική του συστήματος

Η αρχιτεκτονική του συστήματος φαίνεται στην εικόνα 13.



Εικόνα 13: Η αρχιτεκτονική του συστήματος.

Χωρίζεται σε 3 επίπεδα που στο σχήμα έχουν χωριστεί οριζόντια.

i. Δεδομένα εισόδου
   Τα δεδομένα που δίνουμε εμείς από το πληκτρολόγιο είναι η εντολή στην φυσική γλώσσα και η διεύθυνση της οντολογίας OWL(Είτε από το σκληρό δίσκο είτε σε μορφή URI).

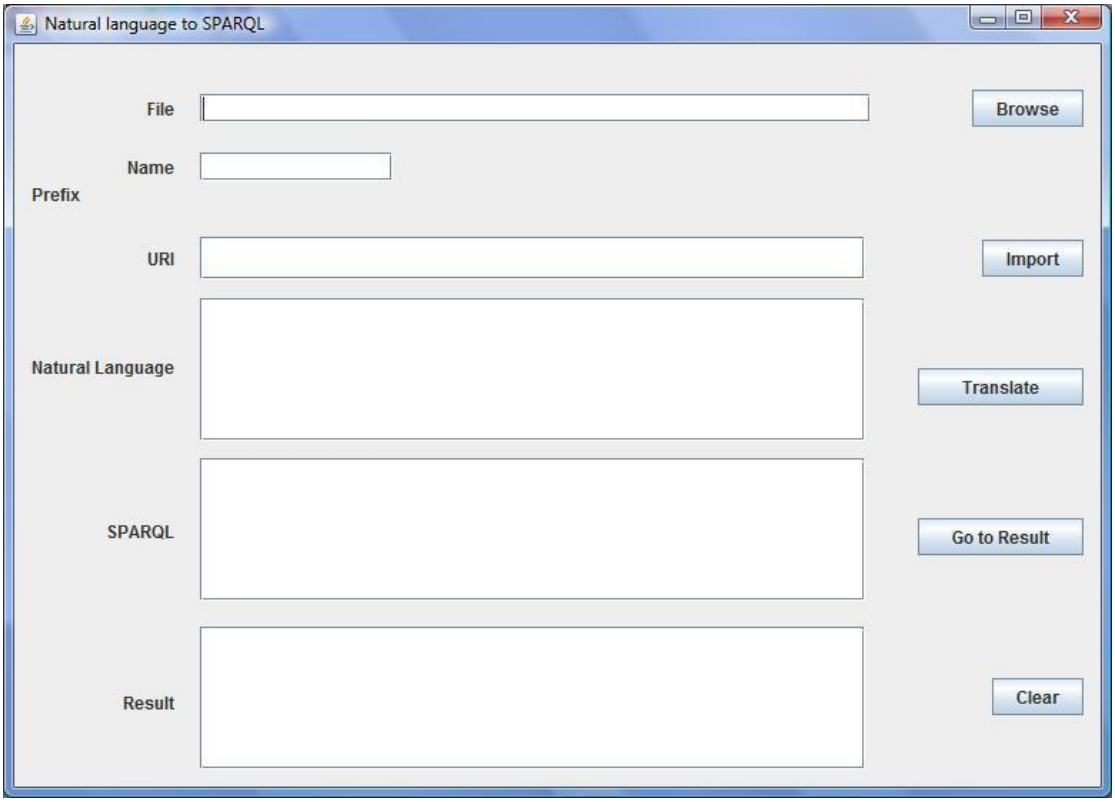
ii. Java Interface
   Μέσω της Java παράγονται τα αποτελέσματα σύμφωνα με τα δεδομένα που έχουμε δώσει εμείς από το πληκτρολόγιο.

iii. Δεδομένα εξόδου
   Τα δεδομένα που μας επιστρέφει η Java.

Μας επιστρέφει το SPARQL query το οποίο είναι το αποτέλεσμα του ερωτήματος της φυσικής γλώσσας που είναι μεταφρασμένη σε SPARQL με το λεξικό το οποίο υπάρχει παρακάτω.
Το άλλο στοιχείο που μας επιστρέφει είναι το αποτέλεσμα του συνδυασμού της εντολής SPARQL με την οντολογία. Αυτό επιτυγχάνεται μέσω του Jena API το οποίο το έχουμε φορτώσει στο πρόγραμμα με μορφή βιβλιοθηκών.

5. Περιγραφή της λειτουργίας του συστήματος

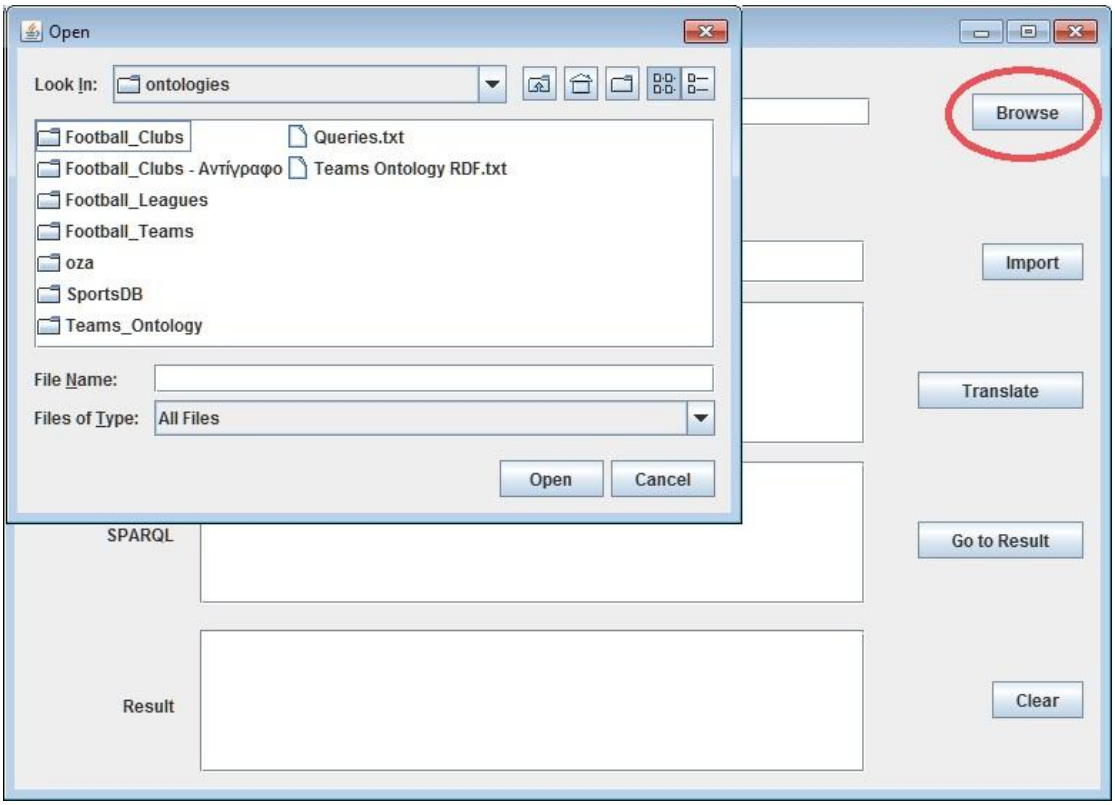
Το GUI το σχεδιάσαμε με τα εργαλεία που υπάρχουν στο πρόγραμμα Eclipse. Είναι ένα Java Interface στο οποίο έχουμε εισάγει της βιβλιοθήκες από το Jena API από ένα Plug-in για το eclipse. Η εικόνα 14 δείχνει το ξεκίνημα με κενά τα text areas. Την λειτουργιά του κάθε JTextArea και JButton θα την περιγράψουμε παρακάτω.



Εικόνα 14: Η αρχική εικόνα του συστήματος.

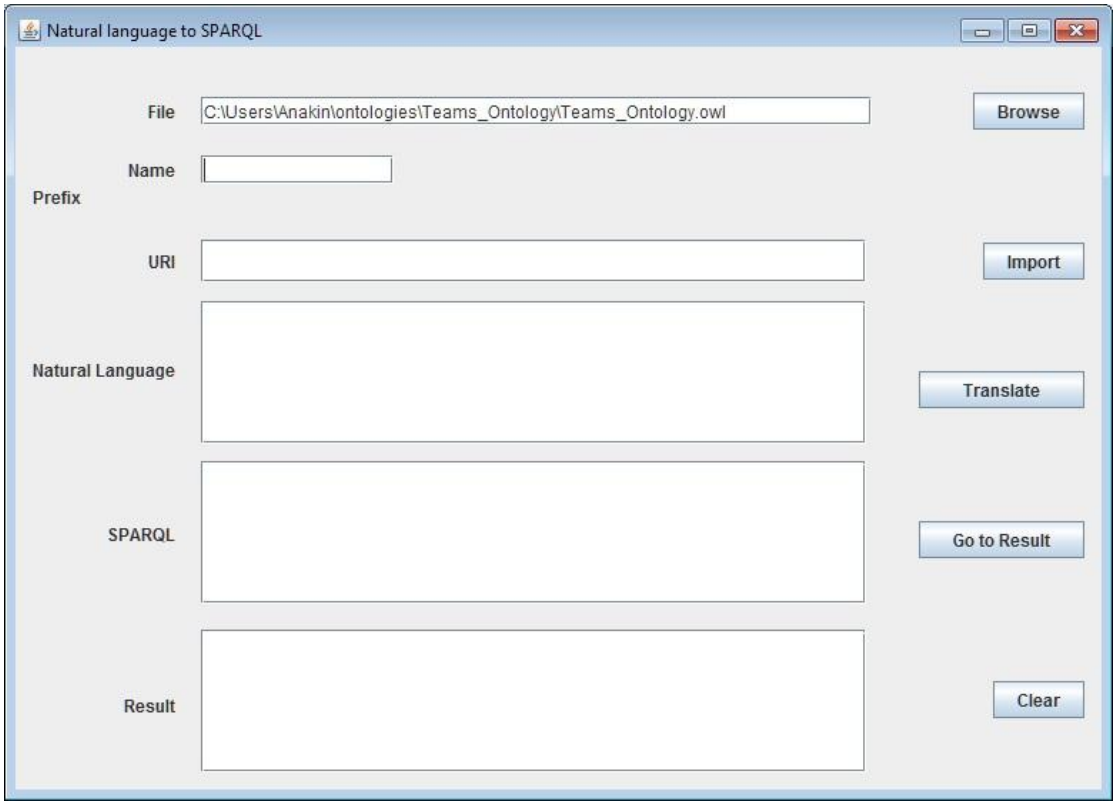
Μέσω φωτογραφιών θα περιγράψουμε την λειτουργιά του προγράμματος μας.

1. Η πρώτη κίνηση είναι να δηλώσουμε την οντολογία για την οποία θέλουμε να εκτελέσουμε την εντολή SPARQL. Είτε αντιγράφουμε το URI της άμα είναι ανεβασμένη στον ιστό είτε δηλώνουμε την θέση της στο σκληρό μας δίσκο. Αυτό γίνεται πατώντας το κουμπί Browse το οποίο μας ανοίγει ένα file explorer(Εικόνα 15).



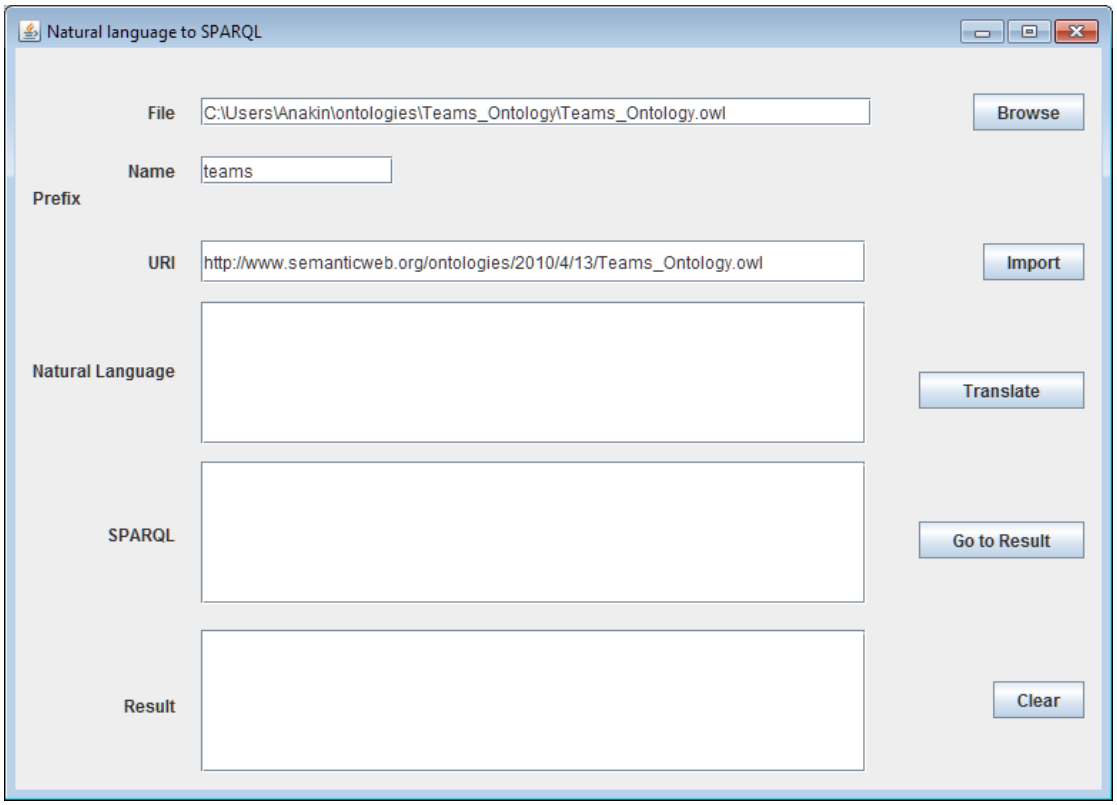
Εικόνα 15

Έπειτα βρίσκουμε την θέση της οντολογίας ,την επιλέγουμε και τέλος πατώντας OK βλέπουμε την διεύθυνση στο πρώτο JTextArea(Εικόνα 16).



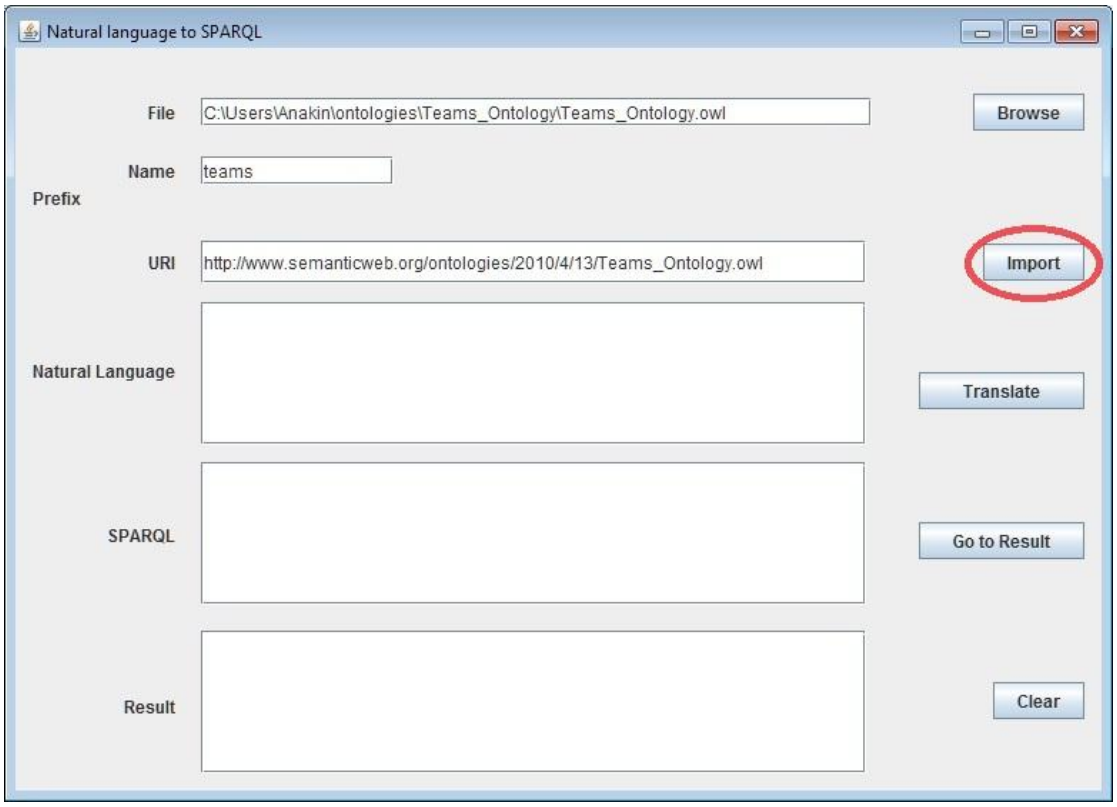
Εικόνα 15

2. Επόμενο βήμα είναι να ασχοληθούμε με το πρόθεμα(prefix). Ουσιαστικά είναι μια συντομογραφία του URI της οντολογίας(άμα δεν είναι ανεβασμένη στον ιστό ,φυσικά είναι ανενεργό από οποιονδήποτε φυλλομετρητή και δουλεύει μόνο στον υπολογιστή που υπάρχει η οντολογία). Στο πεδίο name γράφουμε το όνομα που θέλουμε να έχει το πρόθεμα ενώ στο πεδίο URI εισάγουμε τοURI(Εικόνα 16).

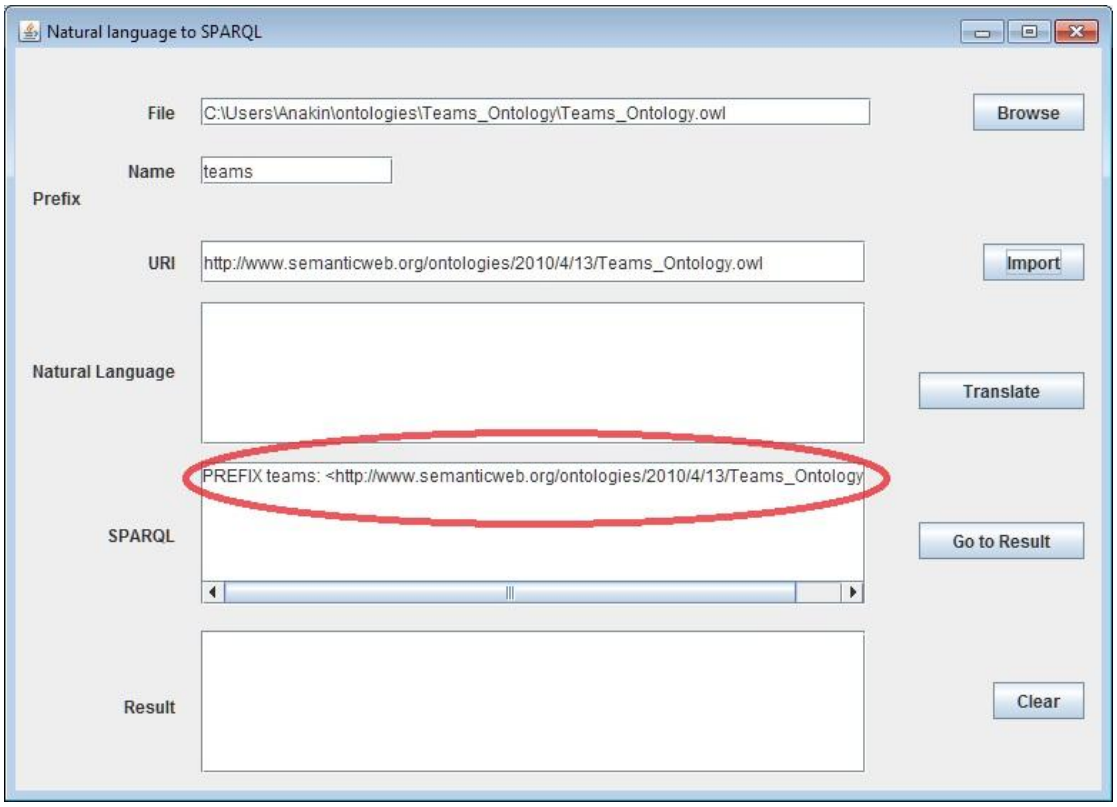


Εικόνα 16

Πατάμε το πλήκτρο Import(Εικόνα 17) και το πρόθεμα εμφανίζεται στο πεδίο της SPARQL(Εικόνα 18).



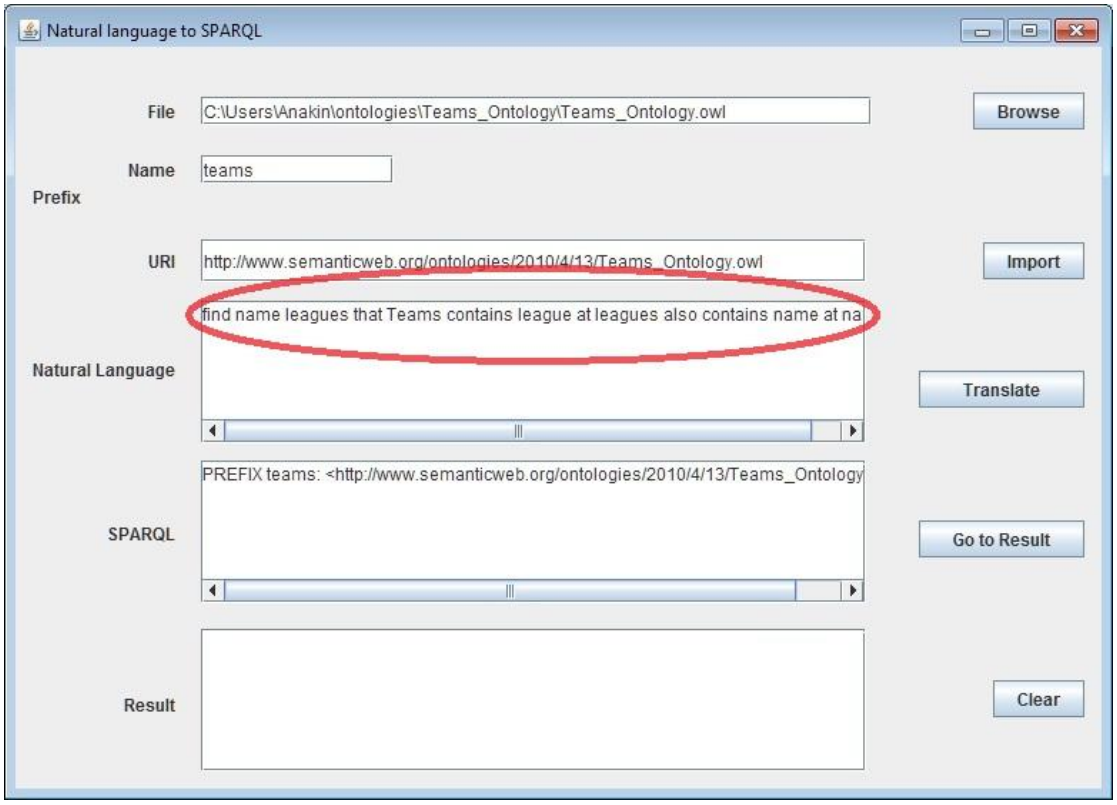
Εικόνα 17



Εικόνα 18

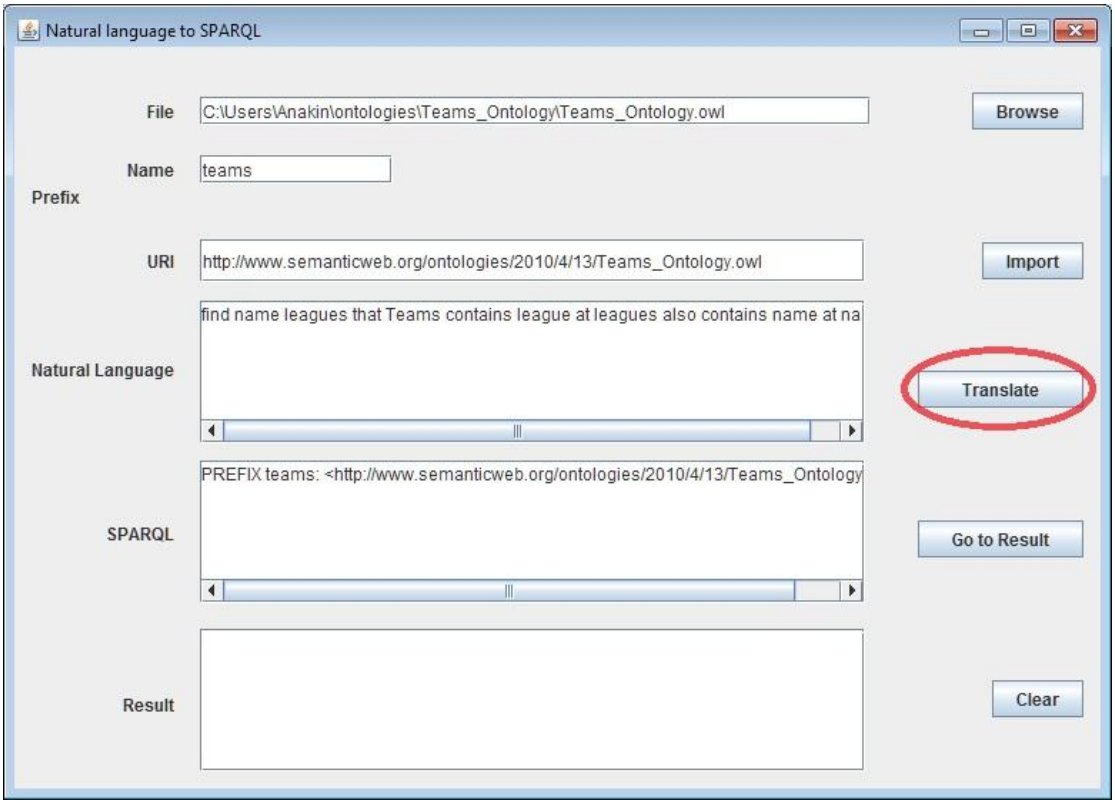
Πρέπει να σημειωθεί ότι μπορούμε να έχουμε παραπάνω από ένα πρόθεμα σε μια εντολή SPARQL.

3. Έπειτα πληκτρολογούμε την εντολή σε φυσική γλώσσα(Εικόνα 19).



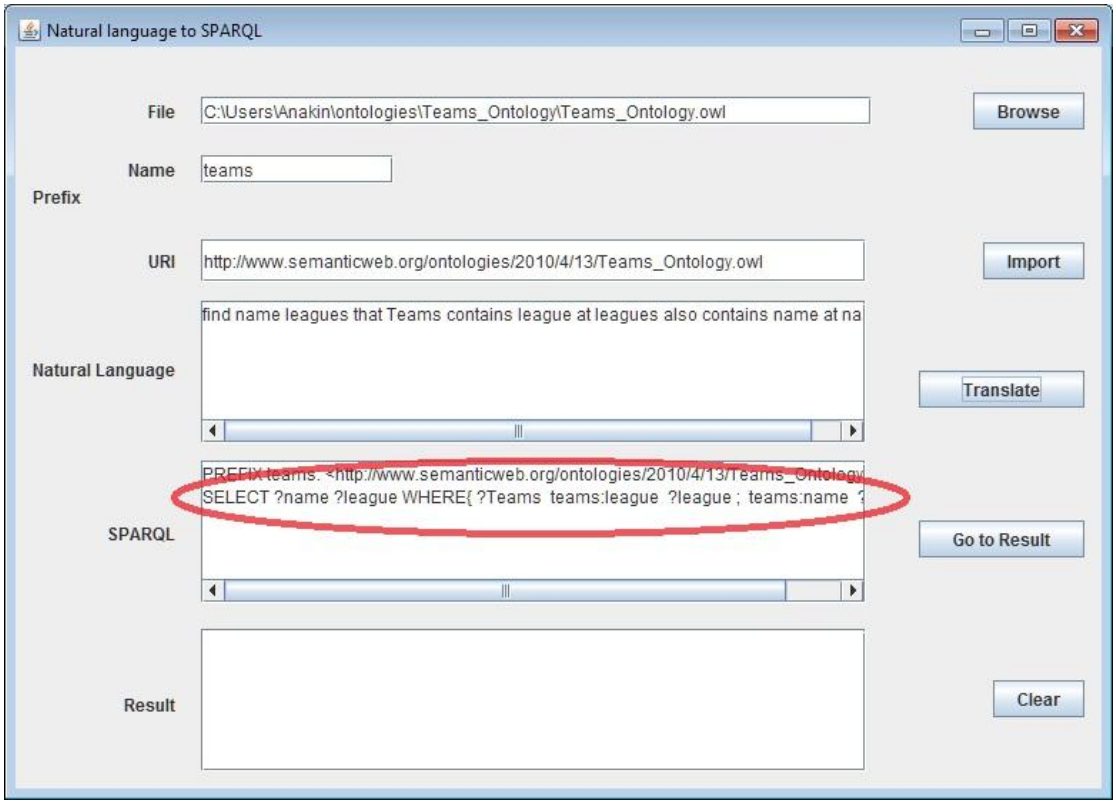
Εικόνα 19

Πατάμε το κουμπί Translate(Εικόνα 20). Εκεί το κείμενο που βρίσκεται στην Text Area χωρίζεται με βάση τα κενά και μεταφράζεται σύμφωνα με το συντακτικό που υπάρχει παρακάτω.



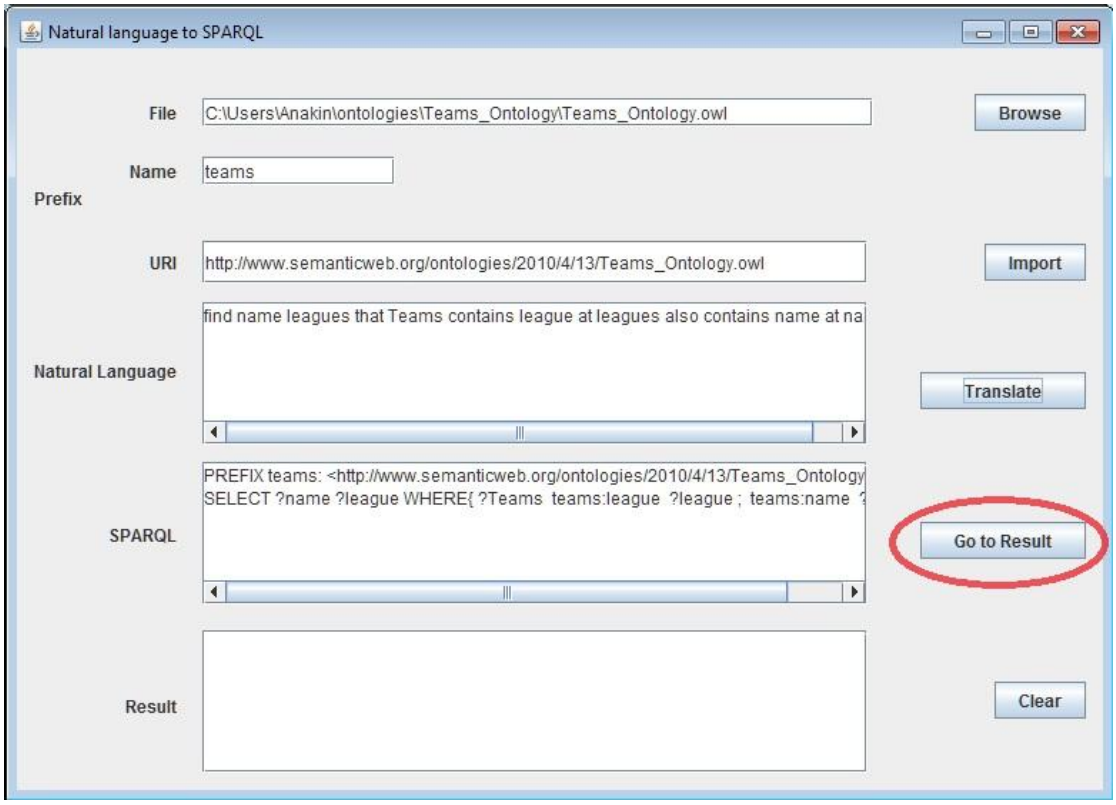
Εικόνα 20

Και το αποτέλεσμα εμφανίζεται στο πεδίο της SPARQL(Εικόνα 21).



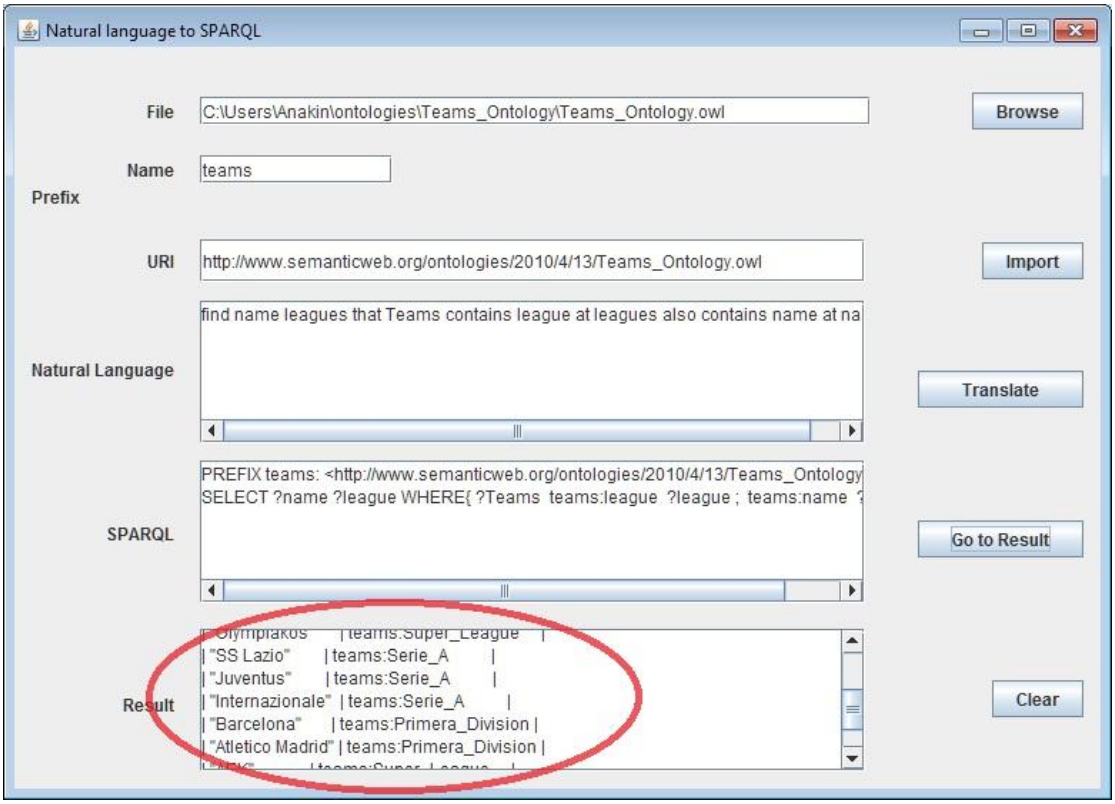
Εικόνα 21

4. Έπειτα πατάμε το κουμπί Go to Result(Εικόνα 22).



Εικόνα 22

Σε αυτό το σημείο το Jena Framework πιάνει δουλειά και συνδυάζει την οντολογία με το ερώτημα SPARQL. Και αυτό το αποτέλεσμα εμφανίζεται στο πεδίο Result(Εικόνα 23)



Εικόνα 23

5. Πατώντας το κουμπί Clear καθαρίζονται όλα τα πεδία και επιστρέφουμε στην οθόνη που εμφανίζεται στην εικόνα 14.

6. Λεξικό και συντακτικό της φυσικής γλώσσας του συστήματος.

Το λεξικό και συντακτικό αυτού του συστήματος έχει σχεδιαστεί για τη συγκεκριμένη οντολογία. Η σύνταξη του κάθε ερωτήματος έχει προσαρμοστεί για το συντακτικό της SPARQL. Αυτός είναι ο λόγος που η σύνταξη των λέξεων στις προτάσεις της φυσικής γλώσσας ,δεν είναι απολύτως ορθός. Είναι case sensitive δηλαδή αν μια λέξη έχει γραφτεί με κεφαλαία ή μικρά γράμματα έχει διαφορά. Τα κενά μεταξύ των προτάσεων είναι πολύ σημαντικά επειδή είναι η βάση για να ξεχωρίσουν τα στοιχεία το ένα από το άλλο στον κώδικα της java. Το ίδιο συμβαίνει με τα σημεία στίξης όπου είναι τα κλειδιά για να ανοίγουν και να κλείνουν οι αγκύλες στην SPARQL.

Παρακάτω είναι χωρισμένα στις εντολές SPARQL, στα στοιχεία της οντολογίας και σε ορισμένες εκφράσεις.

- **ΠΡΟΤΑΣΕΙΣ**
  - find → SELECT
  - that →WHERE
  - want→OPTIONAL
  - but→. FILTER
  - only →regex(
  - sort→ORDER BY
  - ascending→ASC(
  - descending → DESC(

- **ΥΠΟΚΛΑΣΕΙΣ**
  - Teams →?Teams
  - Color→?Color
  - Stadium→?Stadium
  - League→?League

- **ΙΔΙΟΤΗΤΕΣ ΑΝΤΙΚΕΙΜΕΝΩΝ**
  - Stadiums & stadium→?stadium
  - colors→?color
  - leagues→?league
  - teams & team→?team

- **ΙΔΙΟΤΗΤΕΣ ΔΕΔΟΜΕΝΩΝ**
  - name & teamname→?name
  - town→?town
  - country & Country→?Country
  - capacity & Capacity→?Capacity
  - established & Established→?Established_Year
  - founted & Founted→?Foundation_Year
  - leaguename & Lname→?League_Name
  - nickname & Nickname→?Nickname
  - stadiumname & Sname→?Stadium_Name

- **ΕΚΦΡΑΣΕΙΣ**

  - the & is→=
  - . & ! → }
  - and & additional → .
  - also → ;
  - OR → ||
  - AND → &&
  - order &' →)
  - choose & from → (

Οι λέξεις at, by, than, if , you, show, me ,with, those είναι κενά.

- **ΕΞΑΙΡΕΣΕΙΣ**

Μερικές εκφράσεις είναι λίγο περίπλοκες έτσι πρέπει να τις γράψουμε με ένα συγκεκριμένο τρόπο. Χρησιμοποιούμε τις γενικές ιδιότητες των πινάκων για να συνδυάσουμε δυο λέξεις. Το όνομα του πινάκα μετά το χωρισμό του κειμένου που βρίσκεται στο Text Area είναι “words” και οι παρακάτω εκφράσεις είναι κατευθείαν από τον πηγαίο κώδικα της Java. Οι περισσότερες αφορούν το πρόθεμα και τις μαθηματικές εκφράσεις κυρίως για συγκρίσεις.

```
➢ if (words[i].equals("contains")) {
words[i] = "";
words[i + 1] =tName.getText() + ":" + words[i + 1];
}
```

- if (words[i].equals("unique")) {

words[i] = "";

words[i + 1] = "<" + tPrefix.getText() + "#" + words[i + 1] + ">";

}

- if (words[i].equals("equals")) {

words[i] = "";

words[i + 1] = "= '" + words[i + 1] + "')";

}

- if (words[i].equals("greater")) {

words[i] = "";

words[i + 2] = "> '" + words[i + 2] + "')";

}

- if (words[i].equals("less")) {

words[i] = "";

words[i + 2] = "< '" + words[i + 2] + "')";

}

- if (words[i].equals("named")) {

words[i] = "";

words[i + 1] = ",'" + words[i + 1] + "')";

}

7. Παραδείγματα μετάφρασης

Εδώ είναι ορισμένα παραδείγματα από την Οντολογία μας. Πρώτα γράφουμε την έκφραση στα Αγγλικά μετά την παραγόμενη SPARQL και τέλος το αποτέλεσμα. Το πρόθεμα υπάρχει ήδη.

1. Find all teams

Natural Language Expression:
find name
that Teams contains name at name .

SPARQL Query:
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name
WHERE{ ?Teams teams:name ?name }

Result:

name
"Real Madrid"
"Panathinaikos"
"Olympiakos"
"SS Lazio"
"Juventus"
"Internazionale"
"Barcelona"
"Atletico Madrid"
"AEK"

2. Find unique team

Natural Language Expression:
find Teams that Teams unique name is "Barcelona" .

SPARQL Query:
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?Teams WHERE{ ?Teams
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#name>
"Barcelona" }

Result:

Teams
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Barcelona_FC>

3. Find all Italian teams and show only town of Rome

Natural Language Expression:
find name town
that teams contains Country at Country but only Country named Italy and
teams contains name at name additional
if you want show me teams contains town at town
but only town named Rome . !
sort by ascending teams order

SPARQL Query:
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?town
WHERE{ ?team teams:Country ?Country . FILTER regex(?Country ,'Italy') .
?team teams:name ?name .
OPTIONAL{ ?team teams:town ?town
. FILTER regex(?town ,'Rome') } }
ORDER BY ASC(?team)

Result:

name	town
"Internazionale"	
"Juventus"	
"SS Lazio"	"Rome"

4. Show me all stadiums bigger than 70000

Natural Language Expression:
find name Sname Capacity
that teams contains name at name and
stadium contains Capacity at Capacity and
teams contains stadium at stadium and
stadium contains Stadium_Name at Sname
but choose those with Capacity greater than 70000 .
sort by descending Capacity order

SPARQL Query:
```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Stadium_Name ?Capacity
WHERE{ ?team teams:name ?name .
?stadium teams:Capacity ?Capacity .
?team teams:stadium ?stadium .
?stadium teams:Stadium_Name ?Stadium_Name
. FILTER ( ?Capacity > '70000') }
ORDER BY DESC( ?Capacity )
```

Result:

name	Stadium_Name	Capacity
"Barcelona"	"Camp Nou"	"98772"
"Real Madrid"	"Santiago Bernabeu"	"80354"
"Internazionale"	"Giusepe Meaza"	"80074"
"SS Lazio"	"Stadio Olimpico"	"72698"
"AEK"	"Spiros Louis (OAKA)"	"71030"
"Panathinaikos"	"Spiros Louis (OAKA)"	"71030"

5. Stadium – capacity

Natural Language Expression:
find Sname Capacity
that Stadium contains Stadium_Name at Sname also
contains Capacity at Capacity .
sort by Capacity

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?Stadium_Name ?Capacity
WHERE{ ?Stadium teams:Stadium_Name ?Stadium_Name ;
teams:Capacity ?Capacity }
ORDER BY ?Capacity
```

Result:

Stadium_Name	Capacity
"Georgios Karaiskakis"	"33500"
"Vicente Calderon"	"54851"
"Dele Alpi"	"69000"
"Spiros Louis (OAKA)"	"71030"
"Stadio Olimpico"	"72698"
"Giusepe Meaza"	"80074"
"Santiago Bernabeu"	"80354"
"Camp Nou"	"98772"

6. Teams – nicknames

Natural Language Expression:

find name Nickname
that Teams contains name at name also
contains Nickname at Nickname .
sort by name

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Nickname
WHERE{ ?Teams teams:name ?name ;
teams:Nickname ?Nickname }
ORDER BY ?name
```

Result:

name	Nickname
"AEK"	"Kitrinomayroi"
"Atletico Madrid"	"Rojiblancos"
"Barcelona"	"Blaugrana"
"Internazionale"	"Nerazzuri"
"Juventus"	"Vechia Signiora"
"Olympiakos"	"Thrylos"
"Panathinaikos"	"Prasinoi"
"Real Madrid"	"Galacticos"
"SS Lazio"	"Aquile"

7. Stadiums from Spain bigger than 70000

Natural Language Expression:
find name Sname Capacity
that team contains Country at Country but only Country named Spain and
team contains name at name and
stadium contains Capacity at Capacity and
team contains stadium at stadium and
stadium contains Stadium_Name at Sname
but choose those with Capacity greater than 70000 .
sort by descending Capacity order

SPARQL Query:
```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Stadium_Name ?Capacity
WHERE{ ?team teams:Country ?Country . FILTER regex( ?Country ,'Spain') .
?team teams:name ?name .
?stadium teams:Capacity ?Capacity .
?team teams:stadium ?stadium .
?stadium teams:Stadium_Name ?Stadium_Name
. FILTER ( ?Capacity > '70000') }
ORDER BY DESC( ?Capacity )
```

Result:
```
-------------------------------------------------
| name | Stadium_Name | Capacity |
===================================
| "Barcelona" | "Camp Nou" | "98772" |
| "Real Madrid" | "Santiago Bernabeu" | "80354" |
-------------------------------------------------
```

8. Find one team and her stadium

Natural Language Expression:
find name Sname
that team contains name at name but only name named Juventus additional
team contains stadium at stadium and
stadium contains Stadium_Name at Sname .

SPARQL Query:
```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Stadium_Name
WHERE{ ?team teams:name ?name . FILTER regex( ?name ,'Juventus') .
?team teams:stadium ?stadium .
?stadium teams:Stadium_Name ?Stadium_Name }
```

Result:

```
----------------------------
| name | Stadium_Name |
============================
| "Juventus" | "Dele Alpi" |
----------------------------
```

9. Find all teams from Spain

Natural Language Expression:

find name
that team contains Country at Country but only Country named Spain additional
team contains name at name .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name
WHERE{ ?team teams:Country ?Country . FILTER regex( ?Country ,'Spain') .
?team teams:name ?name }
```

Result:

```
---------------------
| name |
=====================
| "Real Madrid" |
| "Barcelona" |
| "Atletico Madrid" |
---------------------
```

10. Teams with Yellow color and OAKA stadium

Natural Language Expression:

find name
that Teams contains color at color and
Teams contains stadium at stadium
but choose from color the unique Yellow ' AND from stadium the unique OAKA ' ' and
Teams contains name at name .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name
WHERE{ ?Teams teams:color ?color .
?Teams teams:stadium ?stadium
. FILTER ( ( ?color =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Yellow> ) && (
?stadium =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#OAKA> ) ) .
?Teams teams:name ?name }
```

Result:

```
---------
| name |
=========
| "AEK" |
```

11. Teams with OAKA stadium

Natural Language Expression:

find name
that Teams contains stadium at stadium
but choose stadium the unique OAKA ' and
Teams contains name at name .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name
WHERE{ ?Teams teams:stadium ?stadium
. FILTER ( ?stadium =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#OAKA> ) .
?Teams teams:name ?name }
```

Result:

```
-------------------
| name |
===================
| "Panathinaikos" |
| "AEK" |
-------------------
```

12. Find Stadium of a team

Natural Language Expression:

find Sname
that Teams contains team at team but choose team the unique Lazio ' and
Teams contains Stadium_Name at Sname .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?Stadium_Name
WHERE{ ?Teams teams:team ?team . FILTER ( ?team =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Lazio> ) .
?Teams teams:Stadium_Name ?Stadium_Name }
```

Result:

```
---------------------
| Stadium_Name |
=====================
| "Stadio Olimpico" |
---------------------
```

13. Find color of teams

Natural Language Expression:
find name colors
that Teams contains color at colors also contains name at name .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?color
WHERE{ ?Teams teams:color ?color ; teams:name ?name }
```

Result:

```
-----------------------------------------------------------------------------------------------
| name | color |
=========================================================================
| "Real Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#White>
| "Panathinaikos" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#White>
| "Panathinaikos" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Green>
| "Olympiakos" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Red>
| "Olympiakos" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#White>
| "SS Lazio" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Blue>
| "Juventus" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Black>
| "Juventus" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#White>
| "Internazionale" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Blue> |
| "Internazionale" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Black> |
| "Barcelona" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Red>
| "Barcelona" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Blue>
| "Atletico Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Red>
| "Atletico Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#White>
| "AEK" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Yellow> |
| "AEK" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Black> |
-----------------------------------------------------------------------------------------------
```

14. Find teams of leagues

Natural Language Expression:
find name leagues
that Teams contains league at leagues also contains name at name .

SPARQL Query:
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?league
WHERE{ ?Teams teams:league ?league ; teams:name ?name }

Result:

name	league
"Real Madrid"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division>
"Panathinaikos"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Super_League>
"Olympiakos"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Super_League>
"SS Lazio"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Serie_A>
"Juventus"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Serie_A>
"Internazionale"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Serie_A>
"Barcelona"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division>
"Atletico Madrid"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division>
"AEK"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Super_League>

15. Find teams of 1 leagues

Natural Language Expression:
find name leagues
that Teams contains league at leagues but choose leagues is unique Primera_Division '
and Teams contains name at name .

SPARQL Query:
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?league
WHERE{ ?Teams teams:league ?league . FILTER (?league =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Divisi
on>)
. ?Teams teams:name ?name

Result:

| name | league |

==========

| "Real Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

| "Barcelona" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

| "Atletico Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

---

16. Find teams stadium town of 1 leagues

Natural Language Expression:

find name Sname town leagues
that Teams contains league at leagues but choose leagues is unique Primera_Division '
and Teams contains name at name
and Teams contains town at town
and Teams contains stadium at stadium
and stadium contains Stadium_Name at Sname .

SPARQL Query:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Stadium_Name ?town ?league
WHERE{ ?Teams teams:league ?league . FILTER ( ?league =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> )
. ?Teams teams:name ?name
. ?Teams teams:town ?town
. ?Teams teams:stadium ?stadium
. ?stadium teams:Stadium_Name ?Stadium_Name }
```

Result:

| name | Stadium_Name | town | league

==========

| "Real Madrid" | "Santiago Bernabeu" | "Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

| "Barcelona" | "Camp Nou" | "Barcelona" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

| "Atletico Madrid" | "Vicente Calderon" | "Madrid" | <http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division> |

---

17. Find teams stadium town of 1 leagues and have red color

Natural Language Expression:

find name Sname town leagues
that Teams contains league at leagues but choose leagues is unique Primera_Division '
and Teams contains color at colors but choose colors the unique Red '
and Teams contains name at name
and Teams contains town at town
and Teams contains stadium at stadium
and stadium contains Stadium_Name at Sname .

SPARQL Querry:

```
PREFIX teams:
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#>
SELECT ?name ?Stadium_Name ?town ?league
WHERE{ ?Teams teams:league ?league . FILTER ( ?league =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Divisi
on> )
. ?Teams teams:color ?color . FILTER ( ?color =
<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Red> )
. ?Teams teams:name ?name
. ?Teams teams:town ?town
. ?Teams teams:stadium ?stadium
. ?stadium teams:Stadium_Name ?Stadium_Name }
```

Result:

name	Stadium_Name	town	league
"Barcelona"	"Camp Nou"	"Barcelona"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division>
"Atletico Madrid"	"Vicente Calderon"	"Madrid"	<http://www.semanticweb.org/ontologies/2010/4/13/Teams_Ontology.owl#Primera_Division>

8. Βιβλιογραφία

1. Nikos Papadakis and Pavlos Kefalas , A tool for access to Relational Databases in natural language , Department of Sciences, Techological Educational Institute of Crete
2. Μάριος Χατζηδημητρίου: Ανάπτυξη λογισμικού για διαχείριση RDF/XML σημάνσεων για το Semantic Web και ενοποίησή του με τα εργαλεία SeEBrowser και Annotation Tool, ΤΕΙ Θεσσαλονίκης – Τμήμα Πληροφορικής
3. Μανόλης Γεργατσούλης : Εισαγωγή στο Σημασιολογικό Παγκόσμιο Ιστό και τις Οντολογίες, Τμήμα Αρχειονομίας & Βιβλιοθηκονομίας, Ιόνιο Πανεπιστήμιο
4. Δρ. Δημήτρης Ιακωβίδης: Απόλυτη Java ,Εκδόσεις ΙΩΝ.
5. Else Lervik and Vegard B. Havdal: Java με UML, Εκδόσεις Κλειδάριθμος.
6. Valentin Tablan , Danica Damljanovic and Kalina Bontcheva ,A natural language query interface to structured information , Department of Computer Science,Univercity of Sheffield,Sheffield,UK
7. Alexander Ran and Raimondas Lencevicius ,Natural Language Query System for RDF Repositories, Nokia Research Center Cambridge, 3 Cambridge Center, Cambridge
8. Andreea-Georgiana Zbranca, Diana Andreea Gorea, Lucian Bentea , Qedia – Natural Language Queries on DBPedia Faculty of Computer Science, “A.I. Cuza” University, Romania
9. http://www.ibm.com/developerworks/xml/library/j-jena/
10. http://www.ibm.com/developerworks/xml/library/j-sparql/
11. http://www.iandickinson.me.uk/articles/jena-eclipse-helloworld/
12. http://en.wikipedia.org/wiki/SPARQL
13. http://en.wikipedia.org/wiki/Resource_Description_Framework
14. http://en.wikipedia.org/wiki/Web_Ontology_Language
15. http://www.csd.uoc.gr/~hy566/
16. http://protege.stanford.edu/
17. http://www.ldodds.com/projects/twinkle/
18. http://marketplace.eclipse.org/
19. http://www.w3.org/2001/sw/sweo/public/UseCases/Tata/
20. http://lpis.csd.auth.gr/mtpx/sw/index.htm

21. http://www.ics.forth.gr/isl/publications/paperlink/athanasis.pdf
22. http://www.w3c.gr/press/pressreleases/2008/01/sparql-pressrelease.el.html
23. http://www.worldlingo.com/ma/enwiki/el/SPARQL
24. http://hermesportal.sourceforge.net/