



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ  
ΤΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ  
ΤΠΟΛΟΓΙΣΤΩΝ  
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΚΑΙ  
ΣΥΣΤΗΜΑΤΩΝ ΜΑΘΗΣΗΣ

---

## ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΓΝΩΣΗΣ

---

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΘΕΜΑ

Αχλάτης Στέφανος-Σταμάτης (03116149)

<el16149@central.ntua.gr>

Αύγουστος 2020

## **Περιεχόμενα**

- 1. Εισαγωγή σελ 3**
- 2. Μέρος Α: Σχεδιασμός Οντολογίας σελ 7**
- 3. Μέρος Β: Δημιουργία τριάδων RDF σελ 19**
- 4. Εισαγωγή των δεδομένων στο virtuoso σελ 29**
- 5. Μέρος Γ: Ερωτήματα SPARQL σελ 30**
- 6. Προβλήματα που αντιμετωπίστηκαν και προτάσεις διόρθωσης σελ 39**
- 7. Επίλογος σελ 40**
- 8. Βιβλιογραφία σελ 41**

## 1. Εισαγωγή

### 1.1 Σκοπός της εργασίας

Σκοπός της εργασίας είναι η κατασκευή μια σημασιολογικής βάσης γνώσης για τον ΟΑΣΑ με βάση τα standards του W3C που θα μπορέσει να αποτελέσει μέλος του Σημασιολογικού Διαδικτύου.

Στην συνέχεια θα υλοποιηθούν ερωτήματα σε γλώσσα ερωτημάτων του W3C, την SPARQL, που θα αφορούν το σύστημα μεταφορών του ΟΑΣΑ και η απάντηση τους θα δοθεί από τη βάση γνώσης που υλοποιήθηκε νωρίτερα.

### 1.2 Γενική Περιγραφή της εργασίας

Αρχικά, θα κατασκευαστεί μια οντολογία σε OWL2 που θα περιγράφει τις ανάγκες των μέσων μεταφοράς. Η κατασκευή της οντολογίας θα γίνει με χρήση του προγράμματος Protege που με χρήση κατάλληλο UI παράγει αυτόματα την οντολογία σε OWL. Η οντολογία θα αφορά όλα τα μέσα μεταφοράς, αλλά η εργασία θα γίνει πιο συγκεκριμένη για τα Δρομολόγια Αστικών Συγκοινωνιών Αθήνας.

Ως πηγή δεδομένων για τα Δρομολόγια Αστικών Συγκοινωνιών Αθήνας θα έχουμε το <http://geodata.gov.gr/> ένα εγχείρημα για παροχεί ανοικτών δεδομένων για δράσεις και φορείς που λαμβάνουν χώρα στην Ελληνική επικράτεια.

Εκεί θα βρούμε δεδομένα για τα δρομολόγια του ΟΑΣΑ σε μορφή csv, δηλαδή αρχεία που η τιμές τους είναι χωρισμένες με κόμμα, και που θα ακολουθούν το πρότυπο gtfs.

Το πρότυπο gtfs, General Transit Feed Specification, παρέχει ένα template για την επαρκής περιγραφή των δρομολογίων των μέσων μεταφοράς. Περισσότερες πληροφορίες για το πρότυπο gtfs μπορούν να βρεθούν εδώ.

Στην συνέχεια αυτά τα δεδομένα με χρήση κώδικα σε Python3 θα μετατραπούν σε τριάδες RDF σύμφωνα με το πρότυπο W3C, μάλιστα για τα URI θα χρησιμοποιηθεί το εκπαιδευτικό domain <http://www.example.com/>.

Μετά θα πρέπει να χρησιμοποιηθεί το πρόγραμμα OpenLink Virtuosos για την εισαγωγή των δεδομένων και της οντολογίας σε ένα σύστημα που μπορεί να τα κατανόηση, εκεί θα γίνουν ερωτήματα σε γλώσσα SPARQL για να φανεί η δομή του συστήματος και να εκτελεστούν βασικά ερωτήματα που θα μπορούσε να είχε ένας χρήστης των υπηρεσιών του ΟΑΣΑ.

Για την αναπαράσταση των γεωγραφικών σημείων γίνεται κατάλληλη σημασιολογική αναπαράσταση με χρήση του LinkedGeoData.

### 1.3 Τι είναι το Σημασιολογικό Διαδίκτυο - Semantic Web

Αρχικά θα κάνουμε μια σύντομη εισαγωγή στο Σημασιολογικό Διαδίκτυο για να γίνει πλήρως κατανοητός ο στόχος της εργασίας, και θα αρχίσουμε δίνοντας έναν ενδιαφέρον ορισμό του Σημασιολογικού Διαδικτύου:

“Σημασιολογικό Διαδίκτυο είναι ένα σύνολο από αρχές και βέλτιστες τακτικές για την μεταφορά τόσο των δεδομένων όσο και της σημασιολογία των δεδομένων αυτών, μέσω του Web για να εφαρμοστεί σε εφαρμογές.”

Ας αναλύσουμε της έννοιες του προηγούμενου ορισμού μια προς μια.

“Ένα σύνολο από αρχές”

Προτού ο Tim Berners-Lee ανακαλύψει το World Wide Web υπήρχαν πολύ ισχυρότερα hypertext συστηματα, ωστόσο το web χτύχτηκε με βάσει απλές προδιαγραφές που είναι γνωστές ως public standards. Αυτή η απλότητα έδωσε την δυνατότητα στους χρήστες να εφαρμόσουν αυτό το σύστημα μόνοι τους, δηλαδή να δημιουργήσουν δικούς τους web servers, web browsers και κυρίως δικές τους

web pages, και έτσι το σύστημα του Tim Berners-Lee εξελίχθηκε ως το μεγαλύτερο και πιο ευρέως χρησιμοποιούμενο hypertext σύστημα. Ο Tim Berners-Lee ιδρυσε τον οργανισμό W3C για να επιβλέπει αυτές τις αρχές, και το semantic web έχει επίσης χτιστεί σε W3C αρχές:

το RDF ως μοντέλο δεδομένων, η SPARQL ως γλώσσα ερωτημάτων, το RDFs και η OWL για την αποθήκευση λεξιλογίων και οντολογιών.

Έτσι ένα προϊόν μπορεί να χτιστεί με χρήση “σημασιολογικών αρχών” αλλά αν δεν χρησιμοποιήσει τις αρχές που προβλέπονται από το W3C δεν θα μπορεί να επικοινωνήσει και να είναι μέλος του semantic web, όπως ένα σύστημα hypertext του 1985 δεν θα μπορούσε να συνδεθεί στο web χωρίς να χρησιμοποιεί την HTML ή το HTTP standard

“Βέλτιστες τακτικές για την μεταφορά δεδομένων ... μέσω του Web για να εφαρμοστεί σε εφαρμογές.” Το αρχικό web του Berners-Lee είχε σχεδιαστεί για την μεταφορά δεδομένων που θα ήταν κατανοητά και θα διαβάζονταν από τον άνθρωπο. Αν θες να μεταβείς από ένα αεροδρόμιο σε ένα άλλο την επόμενη Κυριακή μπορείς να πας στο website μιας αεροπορικής να συμπληρώσετε μια φόρμα ερωτήματος και μετά να διαβάσεις την απάντηση του ερωτήματος στην οθόνη. Τα website που κάνουν συγκρισει τέτοιων δρομολογίων έχουν προγράμματα που ανακτούν δεδομένα από διάφορα web pages και εξάγουν την πληροφορία που χρειάζονται μέσω μιας διαδικασία που ονομάζεται “screen scraping” προτού χρησιμοποιήσουν τα δεδομένα τους στο δικό του site. Έτσι πριν φτιάξει ένα τέτοιο πρόγραμμα μια εταιρεία συγκρίσεων θα πρέπει να αναλυθεί η HTML δομή κάθε site από τις αεροπορικές εταιρίες που θα συμπεριλάβει στο πρόγραμμα για να δει σε ποιο τμήμα του κώδικα το screen scraping θα πρέπει να εστιάσει. Αν μια από τις αεροπορικές εταιρίες αλλάζει το website της, θα πρέπει να ανανεωθεί το πρόγραμμα της εταιρείας συγκρίσεων με βάση τη νέα HTML δομή του site που άλλαξε.

Ο Berners-Lee ήρθε με την ιδέα των Διασυνδεδεμένων Δεδομένων - Linked Data - σαν ένα σύνολο από βέλτιστες τακτικές για την μεταφορά δεδομένων μέσω του web για να είναι πιο εύκολο στις εφαρμογές να ανακτήσουν δεδομένα από δημόσιες σελίδες χωρίς να υπάρχει ανάγκη για screen scraping. Αυτές οι βέλτιστες τακτικές προτείνουν την χρήση URIs για την ονοματοδοσία πραμάτων και την χρήση αρχών όπως το RDF και η SPARQL.

“σημασιολογία των δεδομένων”

Η ιδέα της σημασιολογίας συνήθως ορίζεται ως “η σημασία των λέξεων”. Οι αρχές των Linked Data και των συσχετιζόμενων standards κάνουν ευκολότερη την μετάδοση δεδομένων και η χρήση των URI μπορεί να δώσει λίγη σημασιολογία δίνοντας πληροφορίες για το περιεχόμενο των όρων.

Παραδείγματος χάριν έχουμε το εξής URI <https://www.ece.ntua.gr/gr/undergraduate/courses/3287> ακόμη και αν δεν γνωρίζουμε τι σημαίνει το 3287 μπορούμε να δούμε ότι το URI αντιστοιχεί στα προπτυχιακά μαθήματα της ΣΗΜΜΥ ΕΜΠ και πιθανότατα το 3287 να είναι ο κωδικός κάποιου μαθήματος. Η αποθήκευση του πλήρους νοημάτων των λέξεων ίσως είναι υπολογιστική σπατάλη αλλά η OWL του W3C μας επιτρέπει να αποθηκεύουμε πολύτιμες πληροφορίες έτσι ώστε να αποκτούμε περισσότερα από τα δεδομένα μας. Ένα πχ αν γνωρίζουμε ότι η σχέση a: hasSpouse είναι Συμμετρική, και αν γνωρίζουμε ότι ο A a:hasSpouse την B τότε μπορούμε να εξάγουμε το συμπέρασμα ότι η B a:hasSpouse τον A. Ένα ακόμη παράδειγμα είναι αν γνωρίζουμε ότι ο όρος “Πόλη” είναι Υποσύνολο του όρου “Νομού” ή ότι ο όρος “αγοράζω” είναι αντίθετος του όρου “πουλά” γνωρίζουμε περισσότερα μετα-δεδομένα για το σύστημα μας που μπορεί να μην παρέχονται άμεσα.

## Σχετικά με τα standards του W3C που χρησιμοποιηθηκαν

### Universal Resource Identifier (URI)

Το URI είναι string που περιγράφει μοναδικά σε παγκόσμιο επίπεδο έναν πόρο και χρησιμοποιούνται ορισμένη ιεραρχική κανόνες για τον σχηματισμό του. Ο όρος URI δημιουργήθηκε για να συμπεριλάβει τα URL και URN. Αυτό σημαίνει ότι τα URL είναι επίσης URI. Τα URN δεν χρησιμοποιούνται τόσο συχνά και γι αυτό τα περισσότερα URI είναι URN.

Έτσι μπορούμε να σκεφτούμε ότι τα URI που προσδιορίζουν μοναδικά τους πόρους του RDF σαν μοναδικά ID σε πίνακες μιας σχεσιακής βάσης δεδομένων, με την διαφορά όμως ότι είναι μοναδικά σε παγκόσμιο επίπεδο. Αυτό σου επιτρέπει να διασυνδέσεις δεδομένα από διαφορετικές πηγές από όλο τον κόσμο αντίθετα με τις σχεσιακές βάσεις δεδομένων που σου επιτρέπουν να διασυνδέσεις δεδομένα από διαφορετικούς πίνακες του ίδιου database.

### Resource Description Framework (RDF)

Σύμφωνα με το W3C το RDF είναι ένα data model στο οποίο η μονάδα πληροφορίας ονομάζεται τριπλέτα. Μια τριπλέτα έχει ένα subject ένα predicate και ένα object, μπορείτε να φανταστείτε αυτά ως ένα αναγνωριστικό πηγής (resource identifier), ένα χαρακτηριστικό ή ονομα ιδιότητας και ένα χαρακτηριστικό ή τιμή ιδιότητας. Τέλος για να αποφευχθεί οποιαδήποτε ασάφεια από την πληροφορία, σε κάθε τριπλέτα το subject και το predicate πρέπει να είναι URI.

### Resource Description Framework Schema (RDFs) και Web Ontology Language (OWL)

Το RDFs και η OWL είναι W3C standard λεξιλόγια που σου επιτρέπουν να ορίσεις και να περιγράψεις κλασεις και ιδιότητες που μπορεί να χρησιμοποιήσει ένα dataset από τριπλέτες. Είναι σημαντικό να τονιστεί ότι δεν λειτουργούν ως template στα οποία τα δεδομένα πρέπει να προσαρμοστούν, καθώς λειτουργούν ως επιπρόσθετα **μετα-δεδομένα** που σε βοηθάνε να εξάγεις περισσότερη γνώση από τα δεδομένα σου.

Ας εξετάσουμε ένα παράδειγμα παραγωγής νέας πληροφορίας από το dataset. Έστω ότι έχουμε μια τριπλέτα που περιγραφει το a:spouse property σαν μια owl:SymmetricProperty κι έχουμε και μια άλλη τριπλέτα που λέει ότι ο Γιάννης έχει σύζυγο την Μαρία. Τότε μία εφαρμογή που αντιλαμβάνεται το owl:SymmetricProperty θα αντιληθεί ότι η Μαρία έχει σύζυγο τον Γιάννη.

Ένα ακόμη παράδειγμα με χρήση SPARQL:

Έστω a ένας ονοματόχωρος και έχουμε τα εξης δεδομενα:

```
ab:i0432 ab:firstName      "Richard" ;
ab:lastName        "Mutt" ;
ab:postalCode      "49345" ;
ab:city            "Springfield" ;
ab:homeTel         "(229) 276-5135" ;
ab:streetAddress   "32 Main St." ;
ab:region          "Connecticut" ;
ab:email           "richard49@hotmail.com" ;
ab:playsInstrument ab:vacuumCleaner .
```

Τώρα έστω μια τριπλέτα που αναλύσει το ab:playsInstrument και ab, rdf, rdfs γνωστοί ονοματοχώροι.

```
ab:playsInstrument
  rdf:type rdf:Property ;
  rdfs:comment "Identifies the instrument that someone plays" ;
  rdfs:label "plays instrument" ;
  rdfs:domain ab:Musician ;
  rdfs:range ab:MusicalInstrument .
```

Και εστω το εξής ερώτημα:

```
SELECT ?firstName ?lastName
WHERE
{
  ?person a ab:Musician ;
  ab:firstName ?firstName ;
  ab:lastName ?lastName .
}
```

Κάνοντας το παραπάνω ερώτημα σε έναν επεξεργαστή που δεν χρησιμοποιεί το δεύτερο αρχείο για να αυξήσει τα δεδομένα τότε η απάντηση θα είναι ότι δεν υπάρχει τέτοια τριπλέτα.

Αλλά αν χρησιμοποιήσουμε έναν επεξεργαστή που κατανόηση το rdfs:domain και rdfs:range τότε θα βρεθεί η αναμενόμενη απάντηση: Richard Mutt καθώς θα προστεθούν οι απαιτούμενες τριπλέτες.

### SPARQL Protocol and RDF Query Language (SPARQL)

O Tim Berners-Lee είχε πεί την εξής φράση σχετικά με την SPARQL:

*To να προσπαθείς να χρησιμοποιήσεις το Σημασιολογικό Διαδίκτυο χωρίς την χρήση της SPARQL είναι σαν να προσπαθείς να χρησιμοποιήσεις μια σημασιολογική βάση δεδομένων χωρίς της SQL.*

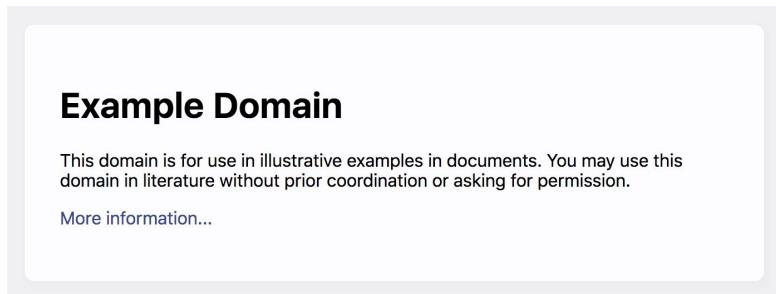
Η SPARQL είναι γλώσσα ερωτημάτων που χρησιμοποιείται στο σημασιολογικό διαδίκτυο όπου τα δεδομένα έχουν RDF μορφή.

## Μέρος Α: Σχεδιασμός Οντολογίας

### Domain Εφαρμογής:

Θα χρησιμοποιήσουμε το <http://www.example.com/> που είναι ένα domain εκπαιδευτικής χρήσης και προσποιείται ότι δέχεται κάθε path μεσα σε αυτό το domain παραδείγματος χάριν το <http://www.example.com/kanoMiaDokimi/> προσποιείται ότι υπάρχει.

Σαφώς σε μια πραγματική εφαρμογή θα έπρεπε να χρησιμοποιήσουμε υπαρκτά domains.



### Classes - Data properties - Object properties

Επιλέγουμε να ορίσουμε ως κλάση μια **σημασιολογικά** θεμελιώδης έννοια που μπορεί να περιέχει ορισμένες ιδιότητες -Data properties- και μπορεί να σχετίζεται με άλλες κλάσεις μέσο ορισμένων Object Properties. Συνήθως θα θέλαμε να είναι κάτι σημασιολογικά σημαντικό στην βάση μας που θα διακρίνει καταγραφές μεταξύ τους. Αυτη η έννοια της σημασιολογικής διαφοροποίησης εξαρτάται από την εφαρμογή, πχ αν φτιάχναμε μια σημασιολογική βάση για εταιρία χρωμάτων θα μπορούσαμε να είχαμε ως κλάση το χρώμα, όμως το να είχαμε ως κλάση το χρώμα του μέσου σε αυτή την εφαρμογή φαίνεται άσκοπο.

Πολλές φορές μπορεί να προκληθεί σύγχυση με το αν πρέπει να ορίσουμε κάτι ως Data Property ή ως Κλάση. Παραδείγματος χάρη το είδος του μέσου μεταφοράς, πχ αν είναι μετρό, τραμ, τρόλεϊ κλπ μπορεί να οριστεί ως κλάση ή ως ιδιότητα του route, με όνομα route-type και με μια σύμβαση αναγνωριστικών που για παράδειγμα αν το route\_type είναι 0 τότε αναφερόμαστε σε τραμ αν είναι 1 αναφερόμαστε σε μετρό κλπ.

Η επιλογή εξαρτάται από την εφαρμογή και το κατα πόσο θέλουμε να δώσουμε έμφαση στο είδος του μέσου, πχ αν θέλαμε να δώσουμε επιπρόσθετα χαρακτηριστικά στο μέσο τότε θα έπρεπε να το ορίσουμε ως κλαση.

Σε αυτό το σημείο αξίζει να αναφέρουμε το General Transit Feed Specification (gtfs) που είναι ένα πρότυπο που μπορεί να περιγράψει ικανοποιητικά τα μέσα μεταφοράς. Το πρότυπο περιγράφεται πλήρως [εδώ](#). Από αυτό το πρότυπο θα αντλησουμε πολλές πληροφορίες για το σχεδιασμό της οντολογίας μας, αλλά δεν θα σταθούμε μόνο σε αυτό αλλά θα δώσουμε μια πιο λεπτομερής οντολογία. Έτσι, κάθε txt αρχείο όπως περιγράφεται από αυτό το λήμμα θα μπορούσε να ήταν μια κλάση στο σύστημά μας, και τα attributes που έχει κάθε .txt αρχείο εκτός από αυτά που συνθέτουν το primary key της αντίστοιχης κλάσης, θα μπορούσαν να οριστούν ως Data properties της κλάσης. Τα object properties δεν μπορούν να εξαχθούν άμεσα από το gtfs ωστόσο μπορεί να εξαχθούν άμεσα και θα το αναλύσουμε στην συνέχεια.

## Classes

Με βάση την προηγούμενη ανάλυση επιλέγουμε να ορίσουμε τις εξείς κλάσεις:

- **Κλάση stops:** Περιγράφει τις στάσεις όποι τα μέσα επιβιβάζουν οι αποβιβάζουν επιβάτες.
- **Κλάση routes:** Περιγράφει δρομολόγια που εκτελούν τα μέσα, π.χ. το 608 είναι ένα δρομολόγιο.
- **Κλάση trips:** Περιγράφει μια συγκεκριμένη διαδρομή ενός δρομολογίου που συμβαίνει σε συγκεκριμένη χρονική περίοδο.
- **Κλάση stop\_times:** Περιγράφει τις χρονικές στιγμές που φτάνει ένα μέσο στις στάσεις του για κάθε trip του.
- **Κλάση calendar:** Περιγράφει τις ημερες της εβδομάδας που λειτουργεί το δρομολόγιο και ορίζει την χρονική περίοδο που ισχύει το περιγραφόμενο σύστημα.
- **Κλάση fares:** Περιγράφει τις απαραίτητες πληροφορίες για την εκπόνηση της συναλλαγής για την αγορά των απαραίτητων εισιτηρίων.
- **Κλάση agency:** Περιγράφει τις απαραίτητες πληροφορίες για την εταιρία που είναι υπεύθυνη για ένα δρομολόγιο
- **Κλάση transit\_means:** Διακρίνει τα μέσα μεταφοράς μεταξύ τους, πχ αν είναι λεωφορείο, αεροπλάνο
- **Κλάση oxima:** Περιγράφει τις απαραίτητες πληροφορίες για το όχημα μεταφοράς.

Κάθε κλάση χωρίζεται σε δύο υπο κλάσεις ανάλογα με το αν το ταξίδι είναι μικρής χρονικής διάρκειας ή μεγάλης χρονικής διάρκειας. Αυτή η διάκριση παράγει μεταγνώση για καλύτερη σημασιολογική κατανόηση κι αντιμετώπιση των ταξιδιών από την εφαρμογή.

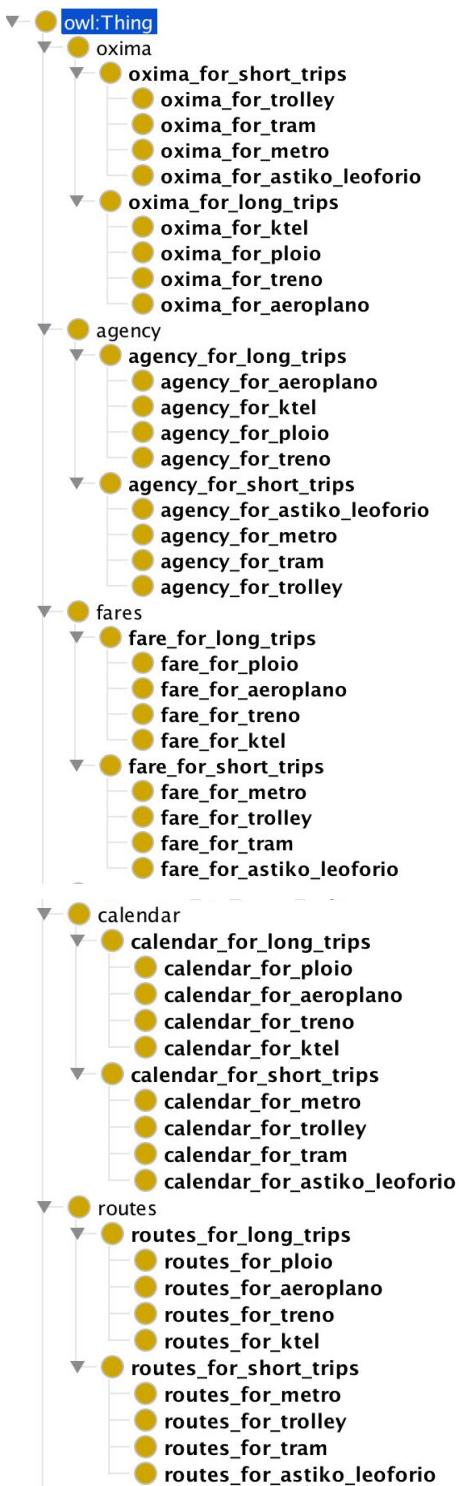
Μετά κάθε μια από αυτές τις δύο κλάσεις χωρίζεται σε υποκλάσεις που περιγράφουν αναλυτικά σε ποιο μέσο μεταφοράς αναφέρονται, π.χ. βλέπουμε ότι η κλαση stops έχει υποκλάση stops\_for\_short\_trips που περιλαμβάνει την κλάση stops\_for\_tram που αντιστοιχεί στην κλάση αντικειμένων που αναφέρονται στις στάσεις του tram που βλέπουμε ότι ανηκουν στην ευρύτερη κλάση των στάσεων για ταξίδια μικρής διάρκειας και στην ακόμη πιο ευρυτερη κλάση που περιλαμβάνει όλες τις στάσεις.

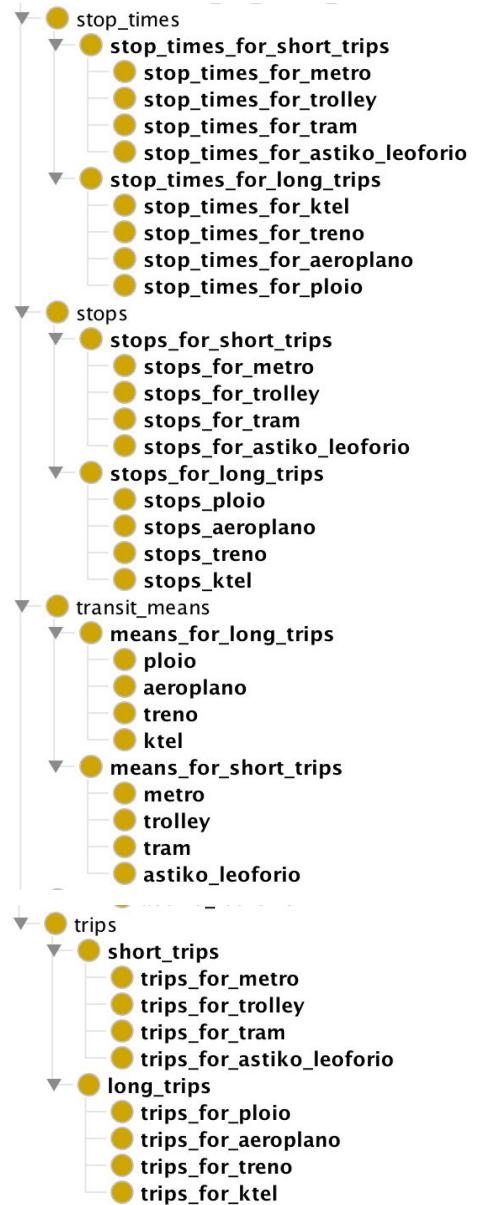
Ένα εύλογο ερώτημα θα ήταν το εξής: “Γιατί όμως να έχουμε και κλάση που διακρίνει τα μέσα με όνομα transit\_means και σε κάθε κλάση να λέμε σε ποιο μέσο αναφερόμαστε;”

Η απάντηση είναι ότι θέλουμε να κατασκευάσουμε μια όσο πιο γενικευμένη οντολογία μπορούμε και ο κάθε προγραμματιστής θα έχει την ευχέρεια να την αξιοποιήσει όπως επιθυμεί, καθώς η owl και η RDFs δεν λειτουργούν ως template στα οποία τα δεδομένα πρέπει να προσαρμοστούν, καθώς λειτουργούν ως επιπρόσθετα **μετα-δεδομένα** που σε βοηθάνε να εξάγεις περισσότερη γνώση από τα δεδομένα σου.

Επομένως ένας προγραμματιστής θα μπορούσε απλά να ορίσει ένα subject ως λεωφορείο και μετά να μην εκμεταλλευτεί την υποκλάση που δηλώνουν ότι αυτό το δρομολόγιο αναφέρεται σε λεωφορείο και να εκμεταλλεύεται όμως την υποκλάση που δηλώνει ότι αυτή η στάση αναφέρεται σε λεωφορείο.

Παρακάτω φαίνονται οι **κλάσεις** όπως εισηχθησαν στο Protege.





Παρακάτω φαίνεται τμήμα του κώδικα σε owl όπως αυτός δημιουργήθηκε από το Protege:

```

<!-- http://www.example.com/agency -->
<owl:Class rdf:about="http://www.example.com/agency"/>

<!-- http://www.example.com/agency_for_aeroplano -->
<owl:Class rdf:about="http://www.example.com/agency_for_aeroplano">
    <rdfs:subClassOf rdf:resource="http://www.example.com/agency_for_long_trips"/>
</owl:Class>

<!-- http://www.example.com/agency_for_astiko_leoforio -->
<owl:Class rdf:about="http://www.example.com/agency_for_astiko_leoforio">
    <rdfs:subClassOf rdf:resource="http://www.example.com/agency_for_short_trips"/>
</owl:Class>

<!-- http://www.example.com/agency_for_ktel -->
<owl:Class rdf:about="http://www.example.com/agency_for_ktel">
    <rdfs:subClassOf rdf:resource="http://www.example.com/agency_for_long_trips"/>
</owl:Class>

```

## Μοναδικό Αναγνωριστικό κάθε κλάσης

Τώρα θα πρέπει να δούμε πως θα μπορέσουμε να διακρίνουμε τις διάφορες στάσεις μεταξύ του, τα διαφορα δρομολόγια μεταξύ τους κλπ.

Θέλουμε να βρούμε ένα μοναδικό αναγνωριστικό για να περιγράψουμε την κάθε οντότητα όταν λέμε ότι η συγκεκριμένη οντότητα ανήκει στην κλάση των στάσεων για τραμ.

Για να αντιμετωπίσουμε αυτό το πρόβλημα ας σκεφτούμε πως λειτουργούν οι σχεσιακές βάσεις δεδομένων. Σε κάθε σχεσιακή βάση οι καταγραφές αποθηκεύονται σε πίνακες και σε κάθε πίνακα οι διάφορες καταγραφές διακρίνονται μεταξύ τους από το Primary Key.

Έτσι η κάθε οντότητα θα έχει ένα μοναδικό ID που είναι το μοναδικό αναγνωριστικό αυτής της οντότητας.

Το ID δεν αποτελεί Data property των εγγραφών της κλάσης, αφού δεν παρέχει κάποια πληροφορία για την καταγραφή αλλά είναι technicality για να διακρίνουμε τις διαφορετικές καταγραφές μεταξύ τους, σαφώς μέσω κατάλληλου URI.

Έτσι, θα έχουμε τις εξεις διακρίσεις:

- Η κλάση routes θα έχει ID το route\_id
- Η κλάση calendar θα έχει ID το service\_id
- Η κλάση stops θα έχει ID το stop\_id
- Η κλάση trips θα έχει ID το trip\_id
- Η κλάση stop\_times θα έχει ID το {trip\_id, stop\_id}
- Η κλάση agency θα έχει ID το agency\_id, που στα δεδομένα δεν υπάρχει γιατί υπάρχει μόνο μια εταιρείας παροχής σε αυτό το dataset, αλλα σε μια εκτεταμένη βάση με πολλές εταιρίες θα έπρεπε να υπάρχει για να διακρίνονται μεταξύ τους
- Η κλάση fares (που δεν υπάρχει σαν δεδομένο αλλά την ορίζουμε εμείς) θα αρκούσε να είχε ένα ID με όνομα fares\_id
- Η κλάση oxima (που δεν υπάρχει σαν δεδομένο αλλά την ορίζουμε εμείς) θα αρκούσε να έχει ID το arithmos\_kikloforias, που είναι ο αριθμός κυκλοφορίας του οχήματος δηλαδή οι πινακίδες του
- Όσον αφορά την κλάση transit\_mean δεν θα έχει κάποιο συγκεκριμένο ID για να διακρίνονται οι διαφορετικές κλάσεις μεταξύ τους, επειδή χρησιμεύει μόνο για διάκριση των οντοτήτων σε κλάσεις και δεν παρέχει κάποια άλλη πληροφορία (data property). Έτσι θα λέμε ότι ένα oxima με arithmos\_kikloforias ανήκει στην κλάση των μέσων μεταφοράς που ονομάζονται τραμ. Αυτό θα το λέγαμε κάπως έτσι:

<http://www.example.com/oxima/T2712> a <http://www.example.com/transit\_means/tram>

Παραδείγματος χάρη με βάση αυτό το σκεπτικό θα έχουμε τις εξής RDF τριπλέτες:

<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02> a <http://www.example.com/trips>.

<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> a <http://www.example.com/stop\_times>.

## Data properties και το ranges τους

Η κλάση routes θα έχει τις εξής Data properties:

- route\_short\_name: που περιγράφει το σύντομο όνομα του δρομολογίου με range το xsd:string
- route\_long\_name: που περιγράφει το αναλυτικό όνομα του δρομολογίου με range το xsd:string
- route\_type: που περιγράφει το είδος του μέσου που χρησιμοποιεί το δρομολόγιο με range το xsd:int
- route\_color: που περιγράφει το χρώμα βάσης του δρομολογίου με range το xsd:string
- route\_text\_color: που περιγράφει το χρωμα της γραμματοσειρας του δρομολογίου με range το xsd:string
- metra\_link: που περιέχει το link που παραπέμπει σε κυβερνητική ιστοσελίδα με τα μέτρα που πρέπει να ληφθούν για το συγκεκριμένο δρομολόγιο για την αντιμετώπιση της κρίσης του SARS-CoV-2.

Η κλάση stops θα έχει τις εξής Data properties:

- stop\_name: που περιγράφει το όνομα της στάσης με range το xsd:string
- stop\_desc: που περιγράφει την περιοχή της στάσεις με range το xsd:string
- stop\_code: που περιγράφει την στάση για τους οδηγούς με μια αριθμητική τιμή μεμε range το xsd:int
- stop\_point: που περιγράφει τις γεωγραφικές συντεταγμένες της στάσης δηλαδή περιλαμβάνει το Latitude και το Longitude της στάσης. Για την αναπαράσταση των γεωγραφικών σημείων γίνεται κατάλληλη σημασιολογική αναπαράσταση με χρήση του LinkedGeoData και έτσι μπορούμε να εκφράσουμε έννοιες όπως “βρες μου όλες τις στάσεις σε ακτίνα 500 μέτρων από την οικία μου”.

Η κλάση trips θα έχει τις εξής Data properties:

- block\_id: που περιγράφει το block που ανήκει το δρομολόγιο με range το xsd:int
- direction\_id: που περιγράφει την κατευθυνση του δρομολογίου 0 για την μια κατευθυνση 1 για την αντιθετη με range το xsd:int
- trip\_headsign: που περιγράφει τον τίτλο του δρομολογίου όπως φαίνεται στις επιγραφές με range το xsd:string

Η κλάση stop\_times θα έχει τις εξής Data properties:

- arrival\_time: που περιγράφει για ένα trip την ώρα που φτάνει σε μια συγκεκριμένη στάση με range το xsd:dateTime
- departure\_time: που περιγράφει για ένα trip την ώρα που φεύγει από μια συγκεκριμένη στάση με range το xsd:dateTime
- drop\_off\_type: που περιγράφει με ποιον τρόπο προγραμματίζεται η αποβίβαση με μια αριθμητική τιμή με range το xsd:int
- pickup\_type: που περιγράφει με ποιον τρόπο προγραμματίζεται η επιβίβαση με μια αριθμητική τιμή με range το xsd:int
- stop\_sequence: που περιγράφει την σειρά της στάσης μέσα σε ένα συγκεκριμένο trip με range το xsd:int

Η κλάση calendar θα έχει τις εξης Data properties:

- start\_date: που περιγράφει την ημερομηνία που από την οποια αρχίζει να ισχύει το dataset με range το xsd:date
- end\_date: που περιγράφει την ημερομηνία που από την οποια σταματάει να ισχύει το dataset με range το xsd:date
- monday: που περιγράφει αν λειτουργεί όλες τις Δευτέρες εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- tuesday: που περιγράφει αν λειτουργεί όλες τις Τρίτες εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- wednesday: που περιγράφει αν λειτουργεί όλες τις Τετάρτες εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- thursday: που περιγράφει αν λειτουργεί όλες τις Πέμπτες εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- friday: που περιγράφει αν λειτουργεί όλες τις Παρασκευές εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- saturday: που περιγράφει αν λειτουργεί όλα τα Σάββατα εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean
- sunday: που περιγράφει αν λειτουργεί όλες τις Κυριακές εντός του οριου που ορίζεται από τις τιμές start\_date και end\_date, λαμβάνει τιμή 1 αν λειτουργεί και 0 αν δεν λειτουργεί με range το xsd:boolean

Η κλάση fares θα έχει τις εξης Data properties:

- price: που περιγράφει το κόστος του εισιτηρίου με range xsd:float
- currency\_type: που περιγράφει με ποιο νόμισμα μπορεί να πληρωθεί το εισιτήριο με range xsd:string
- payment\_method: που περιγράφει που μπορεί να πληρωθεί το εισιτήριο και λαμβάνει την τιμή 0 αν πρέπει να πληρωθεί εντός του μέσου ή 1 αν πρέπει να πληρωθεί πριν την επιβίβαση στο μέσο με range xsd:boolean

Η κλάση agency θα έχει τις εξης Data properties:

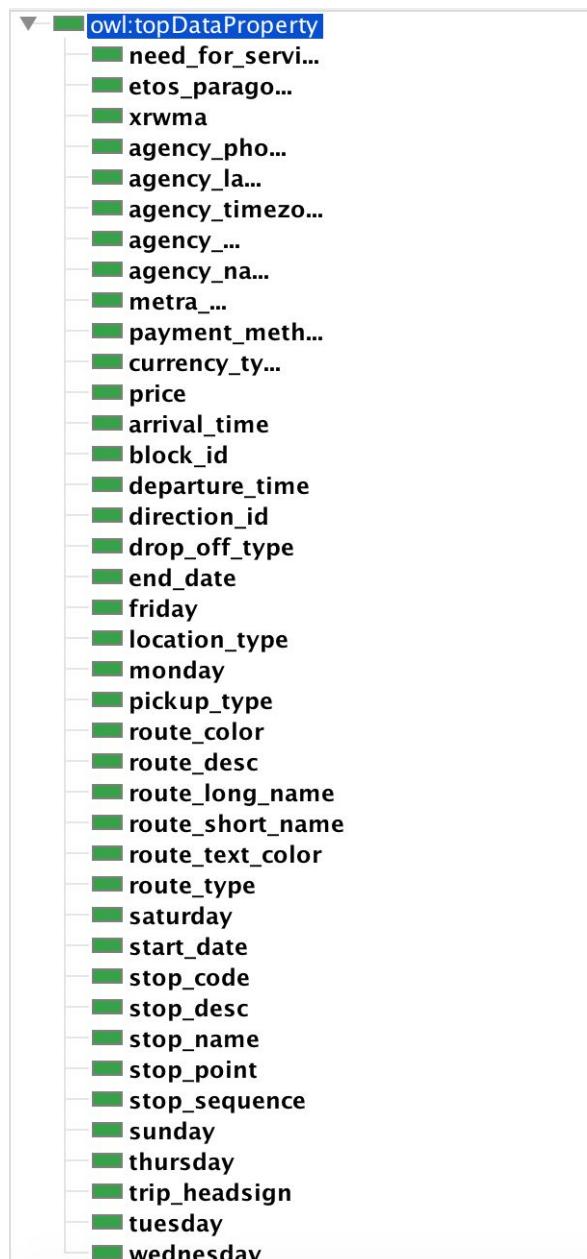
- agency\_name: που περιγράφει το όνομα της εταιρείας για την μεταφορά με range xsd:string
- agency\_url: το url του website της εταιρείας για την μεταφορά με range xsd:string
- agency\_timezone: που περιγράφει το timezone με το οποίο η εταιρεία εργάζεται με range xsd:string, πχ αναγράφει ώρα Αθήνας
- agency\_lang: αναφέρει ποια είναι η πρωτευούσα γλώσσα που χρησιμοποιεί η εταιρεία με range xsd:string
- agency\_phone: αναφέρει ένα τηλέφωνο επικοινωνίας με την εταιρεία με range xsd:int

Η κλάση transit\_means δεν έχει κάποιο data property, και χρησιμεύει για την διάκριση των κατηγοριών μέσων μεταφοράς πχ τραμ αεροπλάνο κλπ

Η κλάση οχίμα θα έχει τις εξής Data properties:

- xrwma: που περιγράφει το χρώμα του οχήματος σε δεκαεξαδικό σύστημα με range xsd:string
- etos\_paragogis: που αναφέρει το έτος παραγωγής του οχήματος με range xsd:int
- need\_for\_service: που αναφέρει λεπτομέρειες για τις ανάγκες επισκευής που πρέπει να γίνουν στο οχήμα με range xsd:string

Παρακάτω φαίνονται τα **Data properties** όπως εισηχθησαν στο Protege:



Παρακάτω φαίνεται τμήμα του κώδικα σε owl όπως αυτός δημιουργήθηκε από το Protege:

```
<!-- http://www.example.com/agency_lang -->
<owl:DatatypeProperty rdf:about="http://www.example.com/agency_lang">
  <rdfs:domain rdf:resource="http://www.example.com/agency"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<!-- http://www.example.com/agency_name -->
<owl:DatatypeProperty rdf:about="http://www.example.com/agency_name">
  <rdfs:domain rdf:resource="http://www.example.com/agency"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<!-- http://www.example.com/agency_phone -->
<owl:DatatypeProperty rdf:about="http://www.example.com/agency_phone">
  <rdfs:domain rdf:resource="http://www.example.com/agency"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<!-- http://www.example.com/agency_timezone -->
<owl:DatatypeProperty rdf:about="http://www.example.com/agency_timezone">
  <rdfs:domain rdf:resource="http://www.example.com/agency"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

## Object Properties

Τώρα θα μελετήσουμε το πλήθος των object properties που διασυνδέουν τις κλάσεις μεταξύ τους και ας σκεφτούμε το object property ως μια κατεύθυνμενη ακμή. Αν έχουμε n κλάσεις θα πρέπει να έχουμε τουλάχιστον (n-1) object properties για να μπορούμε από μια κλάση να μεταβούμε σε οποιαδήποτε άλλη και τις (n-1) αντίστροφές τους. Επίσης αν έχουμε n κλάσεις θα μπορούμε να έχουμε το πολύ n(n-1)/2 διαφορετικές object properties και τις n(n-1)/2 αντίστροφές τους.

Ωστόσο έχοντας ως δεδομένα μόνο της ακμές προς μια κατεύθυνσης μπορούμε να δημιουργήσουμε μέσω τις οντολογίας και τις ακμές αντίθετης κατεύθυνσης αν ορίσουμε αυτες τις δύο σχέσεις ως inverse. Έτσι εμείς θα δώσουμε μόνο της σχέσεις προς μια κατεύθυνση και η οντολογία θα παράξει ως μεταγνώση τις σχέσεις της αντιστροφής κατεύθυνσης.

Επίσης αξίζει να σημειωθεί ότι αυτό γίνεται για την πληρότητα της βάσης και ότι πρακτικά στην εφαρμογή μας που έχουμε να εξετάσουμε SPARQL ερωτήματα μπορούμε να κάνουμε το εξής:  
Έχοντας της σχέσεις προς μια κατεύθυνσης μπορούμε να βρούμε την απάντηση για σχέση αντίθετης κατεύθυνσης ας μελετήσουμε ένα παράδειγμα:

Αν έχουμε μόνο την σχέση a:hasChild μπορούμε να βρούμε τον πατέρα ενός παιδιου ως εξής:

SELECT ?parent

WHERE {

?parent a:hasChild "Nick"^^xsd:string.

}

Δηλαδή δεν χρειάζεται να έχουμε την σχέση a:isFatherOf για να βρούμε τον πατέρα, αλλά μπορούμε να το βρούμε άμεσα από τη σχέση a:hasChild.

Έτσι, επιλέγουμε να ορίσουμε τις ελάχιστες δυνατές σχέσεις μεταξύ των κλάσεων:

- stop\_times\_has\_stops: που συνδέει τη κλάση stop\_times με τη κλάση stops
- stop\_times\_has\_trips: που συνδέει τη κλάση stop\_times με τη κλάση trips
- trips\_has\_calendar: που συνδέει τη κλάση trips με τη κλάση calendar
- trips\_has\_routes: που συνδέει τη κλάση trips με τη κλάση routes
- routes\_has\_fares: που συνδέει τη κλάση routes με τη κλάση fares
- routes\_has\_oxima: που συνδέει τη κλάση routes με τη κλάση oxima
- routes\_has\_transit\_means: που συνδέει τη κλάση routes με τη κλάση transit\_means
- routes\_has\_agency: που συνδέει τη κλάση routes με τη κλάση agency

Και τις αντίστροφες τους κατα αντιστοιχία

- stops\_has\_stop\_times: που συνδέει τη κλάση stops με τη κλάση stop\_times
- trips\_has\_stop\_times: που συνδέει τη κλάση trips με τη κλάση stop\_times
- calendar\_has\_trips: που συνδέει τη κλάση calendar με τη κλάση trips
- routes\_has\_trips: που συνδέει τη κλάση routes με τη κλάση trips
- fares\_has\_routes: που συνδέει τη κλάση fares με τη κλάση routes
- oxima\_has\_routes: που συνδέει τη κλάση oxima με τη κλάση routes
- transit\_means\_has\_routes: που συνδέει τη κλάση transit\_means με τη κλάση routes
- agency\_has\_routes: που συνδέει τη κλάση agency με τη κλάση routes

Παρακάτω φαίνονται τα **Object Properties** όπως εισηγθησαν στο Protege.



Παρακάτω φαίνεται τμήμα του κώδικα σε owl όπως αυτός δημιουργήθηκε από το Protege:

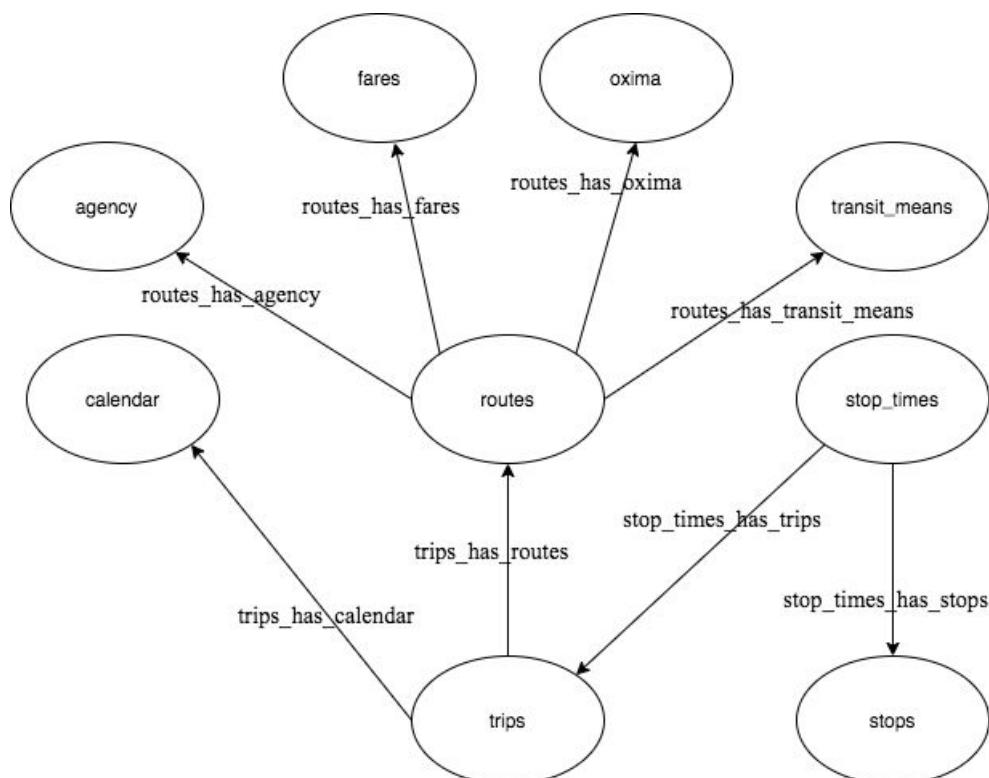
```
<!-- http://www.example.com/calendar_has_trips -->
<owl:ObjectProperty rdf:about="http://www.example.com/calendar_has_trips">
  <owl:inverseOf rdf:resource="http://www.example.com/trips_has_calendar"/>
</owl:ObjectProperty>

<!-- http://www.example.com/routes_has_fares -->
<owl:ObjectProperty rdf:about="http://www.example.com/routes_has_fares">
  <rdfs:domain rdf:resource="http://www.example.com/routes"/>
  <rdfs:range rdf:resource="http://www.example.com/fares"/>
</owl:ObjectProperty>

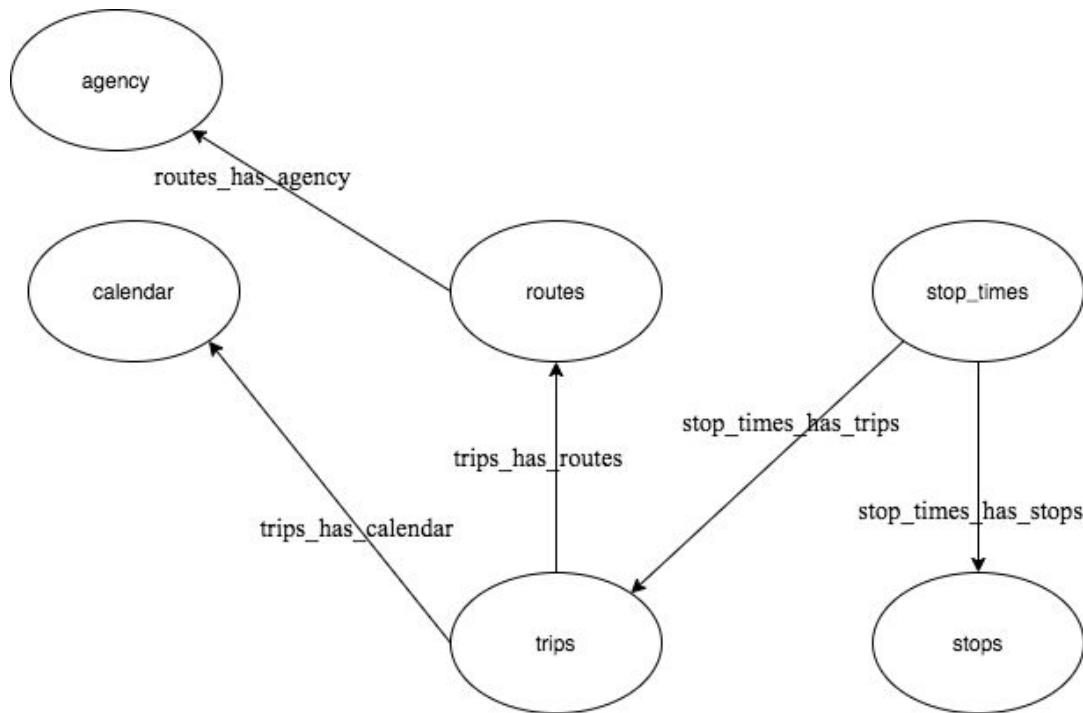
<!-- http://www.example.com/routes_has_trips -->
<owl:ObjectProperty rdf:about="http://www.example.com/routes_has_trips">
  <owl:inverseOf rdf:resource="http://www.example.com/trips_has_routes"/>
</owl:ObjectProperty>

<!-- http://www.example.com/stop_times_has_stops -->
<owl:ObjectProperty rdf:about="http://www.example.com/stop_times_has_stops">
  <owl:inverseOf rdf:resource="http://www.example.com/stops_has_stop_times"/>
  <rdfs:domain rdf:resource="http://www.example.com/stop_times"/>
  <rdfs:range rdf:resource="http://www.example.com/stops"/>
</owl:ObjectProperty>
```

Ας δουμε πως φαίνονται σχηματικά οι σχέσεις (παρατίθενται μόνο οι ευθείες σχέσεις και όχι οι αντιστροφες για την απλότητα των σχημάτων)



Ωστόσο δεδομένου ότι δεν υπάρχει πληροφόρηση στο δοσμένο dataset για τα fares το αντίστοιχο σχήμα που θα μας βοηθήσει για τα SPARQL είναι το εξής:



Θα μπορούσαμε να δημιουγούσαμε παραπάνω σχέσεις έτσι ώστε στα SPARQL ερωτήματα να κάνουμε πιο γρήγορη την διάσχιση του γράφου, ωστόσο σχεδιαστικά επιλέχθηκε να έχουμε το λιγότερο δυνατό πλήθος σχέσεων.

Παρατηρούμε ότι στο σύστημα εισάγουμε μόνο τις σχέσεις κατά μια φορά και έχοντας ορίσει στην OWL τις αντίστροφες σχέσεις ως inverse of τον “ορθών” βλέπουμε ότι παράγεται ως μεταγνώση η αντίστροφες σχέσεις και μπορούμε να τις χρησιμοποιήσουμε στην SPARQL.

## Μέρος Β: Δημιουργία τριάδων RDF

### Δημιουργία τριάδων RDF για Data Properties (Σχολιασμό κώδικα στο τέλος)

Παρατηρώντας την μορφή των δεδομένων που ακολουθούν το πρότυπο gtfs με μικρές αποκλίσεις. Θέλουμε να μετατρέψουμε τα csv αρχεία σε τριάδες RDF.

Επειδή κάθε csv αρχείο αντιστοιχεί σε μια κλάση όπως την ορίζουμε και στην οντολογία πρέπει να δούμε τα εξεις: Ποιο θα είναι το subject; Ποιο το predicate; Ποιο το object;

Σε κάθε περίπτωση το subject θα ορίζεται ως ένα uri με την βοήθεια του ID της κλάσης, που βρίσκεται εντός του ίδιου αρχείου με την κλάση. Το predicate θα είναι είναι το “a” είτε κάποιο uri που δηλώνει το data property και το object θα είναι η τιμή που αναγράφεται στο αντίστοιχο κελί.

Παραδειγματως χαριν έχουμε την εξής εγγραφή:

route_id	route_short_name	route_long_name	route_type	route_color	route_text_color
T3-20	T3	ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ	900	3FFF19	000000

Η αντίστοιχη εγγραφή σε τριάδες RDF είναι η εξής:

```
<http://www.example.com/routes/T3-20> a <http://www.example.com/routes>.  
<http://www.example.com/routes/T3-20> <http://www.example.com/route_short_name> "T3"^^xsd:string.  
<http://www.example.com/routes/T3-20> <http://www.example.com/route_long_name> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.  
<http://www.example.com/routes/T3-20> <http://www.example.com/route_type> "900"^^xsd:int.  
<http://www.example.com/routes/T3-20> <http://www.example.com/route_color> "3FFF19"^^xsd:string.  
<http://www.example.com/routes/T3-20> <http://www.example.com/route_text_color> "000000"^^xsd:string.
```

Οπου παρατηρούμε ότι το subject όλων των τριάδων που αφορούν αυτή την εγγραφή εκφράζεται μέσω του route\_id και τα predicates είναι είτε το “a” που δηλώνει ότι το subject αυτό ανήκει στη κλάση αντικειμένων routes, είτε τα domain spaces των data property με object την αντίστοιχη τιμή που έχουν στο κελί.

Παρατηρούμε ότι η κάθε κλάση αντιστοιχεί σε κάποιο αρχείο txt όπως έχουν δοθεί από το geodata.gov.gr. Σε κάθε τέτοιο αρχείο βρίσκουμε τα απαραίτητα στοιχεία για το ID που θα διακρίνει τις διάφορες οντότητες μεταξύ τους, όπως και όλα τα data properties που σχετίζονται με την δεδομένη κλάση. Επίσης για την δημιουργία των object properties παρατηρούμε ότι χρειαζόμαστε τις εξεις πληροφορίες: το ID της οντότητας της κλάσης που θέλουμε να συσχετίσουμε με μια άλλη κλάση και το ID της άλλης κλάσης. Επομένως μπορούμε να παράγουμε τόσο τον ορισμό τις εκάστοτε οντότητας όσο και τις σχέσεις για όλες τις ιδιότητες της διαβάζοντας το αρχείο με τον τίτλο της κλάσης.

Επίσης επιλέγουμε να μην χρησιμοποιήσουμε την κλάση transit\_means επειδή δεν έχουμε πληροφορίες για τον αριθμό κυκλοφορίας του εκαστοτε οχήματος ενός δρομολογίου όπως αναφέραμε και προηγμένος. Επίσης για να δείξουμε την ευχρηστιά της OWL επιλέγουμε να μην χρησιμοποιήσουμε τις υποκλάσεις αλλά μονο την μητρική κλάση, πχ δεν χρησιμοποιούμε την κλάση stops\_for\_tram αλλά την κλάση stops. Επίσης, η data property metra, που αναφέρεται σε μέτρα προστασίας απέναντι από τον SARS-CoV-2 δεν χρησιμοποιείται καθώς δεν έχουμε κάποια τέτοια πληροφορία από τα δεδομένα. Τέλος, ακολουθούμε το πρότυπο gtfs για να μάθουμε αν ένα όχημα είναι τραμ ή τρολευ κλπ βλέποντας το data property route\_type

Σχετικά με τις πληροφορίες για τον τύπο μέσω του πεδίου route\_type έχουμε την εξής αντιστοίχιση:

3 -> Λεωφορείο 900 -> Τραμ 1 -> Μετρό 800 -> Τρόλεϊ π.χ. από την εξής καταγραφή:

608-20,608,"ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - NEKP. ΖΩΓΡΑΦΟΥ",,3,153CE0,FFFFFF μαθαίνουμε ότι η γραμμή 608 κάνει την διαδρομή ΓΑΛΑΤΣΙ - ΑΚΑΔΗΜΙΑ - NEKP. ΖΩΓΡΑΦΟΥ και είναι τυπου 3, δηλαδή είναι λεωφορείο.

Επιλέχθηκε σειροποίηση με N-triples που έχουν πιο απλή σύνταξη συγκριτικά με άλλους τρόπους σύνταξης, για την αναπαράσταση των RDF τριπλετών όπως φαίνεται και στην συνέχεια.

## Δημιουργία τριάδων RDF για Κλάσεις και Data Properties

Σχετικά με το αρχείο stops.txt

Αρχικό αρχείο stops.txt:

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_lon,location_type
010001,010001,ΣΤΡΟΦΗ,Επί της ΕΛ.ΒΕΝΙΖΕΑΟΥ,37.9986082641367,23.6649846246733,0
010002,010002,ΚΟΛΩΝΕΣ,Επί της ΛΕΩΦ.ΕΑ.ΒΕΝΙΖΕΑΟΥ,37.996719436997,23.6631823242898,0
010003,010003,ΑΓ.ΒΑΡΒΑΡΑ,Επί της ΛΕΩΦ.ΕΑ.ΒΕΝΙΖΕΑΟΥ,37.9946947354979,23.6611529618238,0
010004,010004,ΔΗΜΑΡΧΕΙΟ,Επί της ΗΠΠΕΙΡΟΥ,37.9928326565131,23.6592710105915,0
010005,010005,ΚΡΗΤΗΣ,Επί της ΠΑΛΑΙΩΝ ΠΟΛΕΜΙΣΤΩΝ,37.9904261625749,23.6561503633519,0
010006,010006,ΚΟΥΝΤΟΥΡΙΟΤΟΥ,Επί της ΠΑΛΑΙΩΝ ΠΟΛΕΜΙΣΤΩΝ,37.9880007445562,23.6558200457878,0
010007,010007,ΔΗΜΑΡΧΕΙΟ ΑΓ.ΒΑΡΒΑΡΑΣ,Επί της ΛΕΩΦ.ΕΑ.ΒΕΝΙΖΕΑΟΥ,37.9899667799348,23.6593412065609,0
010008,010008,1Η ΑΓ. ΜΑΡΙΝΑΣ,Επί της ΑΓ.ΜΑΡΙΝΑΣ,37.9959735237703,23.6671036000934,0
010011,010011,ΘΕΜΙΣΤΟΚΛΕΟΥΣ,Επί της ΠΑΛ.ΠΑΤΡΩΝ ΓΕΡΜΑΝΟΥ,37.9884921024649,23.6543373126933,0
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/stops/010001> a <http://www.example.com/stops>.
<http://www.example.com/stops/010001> <http://www.example.com/stop_code> "010001"^^xsd:int.
<http://www.example.com/stops/010001> <http://www.example.com/stop_name> "ΣΤΡΟΦΗ"^^xsd:string.
<http://www.example.com/stops/010001> <http://www.example.com/stop_desc> "Επί της ΕΛ.ΒΕΝΙΖΕΑΟΥ"^^xsd:string.
<http://www.example.com/stops/010001> <http://www.example.com/stop_point> "POINT(37.9986082641367 23.6649846246733)"^^<http://www.openlinksw.com/schemas/virttrdf#Geometry>.
<http://www.example.com/stops/010002> <http://www.example.com/location_type> "0"^^xsd:int.
<http://www.example.com/stops/010002> a <http://www.example.com/stops>.
<http://www.example.com/stops/010002> <http://www.example.com/stop_code> "010002"^^xsd:int.
<http://www.example.com/stops/010002> <http://www.example.com/stop_name> "ΚΟΛΩΝΕΣ"^^xsd:string.
<http://www.example.com/stops/010002> <http://www.example.com/stop_desc> "Επί της ΛΕΩΦ.ΕΑ.ΒΕΝΙΖΕΑΟΥ"^^xsd:string.
<http://www.example.com/stops/010002> <http://www.example.com/stop_point> "POINT(37.996719436997 23.6631823242898)"^^<http://www.openlinksw.com/schemas/virttrdf#Geometry>.
<http://www.example.com/stops/010002> <http://www.example.com/location_type> "0"^^xsd:int.
<http://www.example.com/stops/010003> a <http://www.example.com/stops>.
<http://www.example.com/stops/010003> <http://www.example.com/stop_code> "010003"^^xsd:int.
<http://www.example.com/stops/010003> <http://www.example.com/stop_name> "ΑΓ.ΒΑΡΒΑΡΑ"^^xsd:string.
<http://www.example.com/stops/010003> <http://www.example.com/stop_desc> "Επί της ΛΕΩΦ.ΕΑ.ΒΕΝΙΖΕΑΟΥ"^^xsd:string.
<http://www.example.com/stops/010003> <http://www.example.com/stop_point> "POINT(37.9946947354979 23.6611529618238)"^^<http://www.openlinksw.com/schemas/virttrdf#Geometry>.
<http://www.example.com/stops/010003> <http://www.example.com/location_type> "0"^^xsd:int.
```

Η μετατροπή έγινε με χρήση του εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

stop_idURI = "http://www.example.com/stops"
stop_codeURI = "http://www.example.com/stop_code"
stop_nameURI = "http://www.example.com/stop_name"
stop_descURI = "http://www.example.com/stop_desc"
stop_latURI = "http://www.example.com/stop_lat"
stop_lonURI = "http://www.example.com/stop_lon"
location_typeURI = "http://www.example.com/location_type"

lat = "http://www.w3.org/2003/01/geo/wgs84_pos#lat"
lon ="http://www.w3.org/2003/01/geo/wgs84_pos#long"

string_ = "^^xsd:string."
integer_ = "^^xsd:int."
float_ = "^^xsd:float."

outputFileName = "stopsOUT.txt"
outputFile = open(outputFileName, "w")

inputFileName = "stops.txt"

cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")

        stop_id = row[0]
        stop_code = row[1]
        stop_name = row[2]
        stop_desc = row[3]
        stop_lat = row[4]
        stop_lon = row[5]
        location_type = row[6]

        if cnt>1:
            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> ' + 'a' + ' <' + stop_idURI+ '> ' + ' ' + '\n')
            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + stop_codeURI + '> ' + ' ' + str(stop_code)+ ' ' + integer_ + '\n')
            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + stop_nameURI + '> ' + str(stop_name)+ ' ' + string_ + '\n')
            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + stop_descURI + '> ' + ' ' + str(stop_desc)+ ' ' + string_ + '\n')

            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + lat + '> ' + ' ' + str(stop_lat)+ ' ' + '\n')
            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + lon + '> ' + ' ' + str(stop_lon)+ ' ' + '\n')

            outputFile.write('<' + stop_idURI + '/'+stop_id+ '> '+ '<' + location_typeURI + '> ' + ' ' + str(location_type).rstrip() + ' ' + integer_ + '\n')

print(cnt)
```

Σχετικά με το αρχείο routes.txt

Αρχικό αρχείο routes.txt:

```
route_id,route_short_name,route_long_name,route_desc,route_type,route_color, route_text_color
T3-20,T3,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,,900,3FFF19,000000
T4-20,T4,"ΣΥΝΤΑΓΜΑ - ΝΕΟ ΦΑΛΗΡΟ",,900,3FFF19,000000
T5-20,T5,"ΣΥΝΤΑΓΜΑ - ΒΟΥΛΑ",,900,3FFF19,000000
M2-20,M2,"ΑΝΘΟΥΠΟΛΗ - ΣΤ.ΕΛΛΗΝΙΚΟ",,1,D81A13,000000
M3-20,M3,"ΑΓΙΑ ΜΑΡΙΝΑ - ΔΟΥΚ. ΠΙΑΚΕΝΤΙΑΣ - ΑΕΡΟΔΡΟΜΙΟ",,1,FFE4CA,000000
M1-20,M1,"ΠΕΙΡΑΙΑΣ - ΚΗΦΙΣΙΑ",,1,1C6F0B,ffffff
1-20,1,"ΠΛΑΤ. ΑΤΤΙΚΗΣ - ΤΖΙΤΖΙΦΙΕΣ - ΜΟΣΧΑΤΟ",,800,F19A1E,000000
2-20,2,"ΑΝΩ ΚΥΨΕΑΗ - ΠΑΓΚΡΑΤΙ - ΚΑΙΣΑΡΙΑΝΗ",,800,F19A1E,000000
3-20,3,"Ν. ΦΙΛΑΔΕΛΦΕΙΑ - ΑΝΩ ΠΑΤΗΣΙΑ - ΝΕΟ ΨΥΧΙΚΟ",,800,F19A1E,000000
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/routes/T3-20> a <http://www.example.com/routes>.
<http://www.example.com/routes/T3-20> <http://www.example.com/route_short_name> "T3"^^xsd:string.
<http://www.example.com/routes/T3-20> <http://www.example.com/route_long_name> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/routes/T3-20> <http://www.example.com/route_type> "900"^^xsd:int.
<http://www.example.com/routes/T3-20> <http://www.example.com/route_color> "3FFF19"^^xsd:string.
<http://www.example.com/routes/T3-20> <http://www.example.com/route_text_color> "000000"^^xsd:string.
<http://www.example.com/routes/T4-20> a <http://www.example.com/routes>.
<http://www.example.com/routes/T4-20> <http://www.example.com/route_short_name> "T4"^^xsd:string.
<http://www.example.com/routes/T4-20> <http://www.example.com/route_long_name> "ΣΥΝΤΑΓΜΑ - ΝΕΟ ΦΑΛΗΡΟ"^^xsd:string.
<http://www.example.com/routes/T4-20> <http://www.example.com/route_type> "900"^^xsd:int.
<http://www.example.com/routes/T4-20> <http://www.example.com/route_color> "3FFF19"^^xsd:string.
<http://www.example.com/routes/T4-20> <http://www.example.com/route_text_color> "000000"^^xsd:string.
<http://www.example.com/routes/T5-20> a <http://www.example.com/routes>.
<http://www.example.com/routes/T5-20> <http://www.example.com/route_short_name> "T5"^^xsd:string.
<http://www.example.com/routes/T5-20> <http://www.example.com/route_long_name> "ΣΥΝΤΑΓΜΑ - ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/routes/T5-20> <http://www.example.com/route_type> "900"^^xsd:int.
<http://www.example.com/routes/T5-20> <http://www.example.com/route_color> "3FFF19"^^xsd:string.
<http://www.example.com/routes/T5-20> <http://www.example.com/route_text_color> "000000"^^xsd:string.
```

Η μετατροπή έγινε με χρήση του εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

routesURI = "http://www.example.com/routes"
route_short_nameURI = "http://www.example.com/route_short_name"
route_long_nameURI = "http://www.example.com/route_long_name"
route_descURI = "http://www.example.com/route_desc"
route_typeURI = "http://www.example.com/route_type"
route_colorURI = "http://www.example.com/route_color"
route_text_colorURI = "http://www.example.com/route_text_color"

string_ = "^^xsd:string."
integer_ = "^^xsd:int."

outputFileName = "routesOUT.txt"
outputFile = open(outputFileName, "w")

inputFileName = "routes.txt"

cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")
        route_id = row[0]
        route_short_name = row[1]
        route_long_name = row[2]
        #route_desc = "EMPTY"
        route_type = row[4]
        route_color = row[5]
        route_text_color = row[6]
        if cnt>1:
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + 'a' + ' <' + routesURI + '>' + string_ + '\n')
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_short_nameURI + '>' + string_ + str(route_short_name) + string_ + string_ + '\n')
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_long_nameURI + '>' + string_ + str(route_long_name) + string_ + '\n')
            #outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_descURI + '>' + string_ + str(route_desc) + string_ + string_ + '\n')
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_typeURI + '>' + string_ + str(route_type) + string_ + integer_ + '\n')
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_colorURI + '>' + string_ + str(route_color) + string_ + string_ + '\n')
            outputFile.write('<' + routesURI + '/' + route_id + '> ' + '<' + route_text_colorURI + '>' + string_ + str(route_text_color).rstrip() + string_ + string_ + '\n')

print(cnt)
```

Ας δούμε αναλυτικά τα αποτελέσματα

Σχετικά με το αρχείο trips.txt

Αρχικό αρχείο trips.txt:

```
route_id,service_id,trip_id,trip_headsign,direction_id,block_id,shape_id
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803638-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184458,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803639-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184460,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184459,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803641-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184461,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803642-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184462,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803643-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184462,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803644-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184458,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803645-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184460,
T3-20,ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,9803646-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184459,
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> a <http://www.example.com/trips>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/trip_headsign> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/direction_id> "0"^^xsd:boolean.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/block_id> "2184458"^^xsd:int.
<http://www.example.com/trips/9803639-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> a <http://www.example.com/trips>.
<http://www.example.com/trips/9803639-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/trip_headsign> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/trips/9803639-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/direction_id> "0"^^xsd:boolean.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/block_id> "2184460"^^xsd:int.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> a <http://www.example.com/trips>.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/trip_headsign> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/direction_id> "0"^^xsd:boolean.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/block_id> "2184459"^^xsd:int.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> a <http://www.example.com/trips>.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/trip_headsign> "ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ"^^xsd:string.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/direction_id> "0"^^xsd:boolean.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-Τ3-Παρασκε υή-02> <http://www.example.com/block_id> "2184461"^^xsd:int.
```

Η μετατροπή έγινε με χρήση του εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

trip_idURI = "http://www.example.com/trips"
trip_headsignURI = "http://www.example.com/trip_headsign"
direction_idURI = "http://www.example.com/direction_id"
block_idURI = "http://www.example.com/block_id"

string_ = "^^xsd:string."
integer_ = "^^xsd:int."
boolean_ = "^^xsd:boolean."


outputFileName = "tripsOUTNEW.txt"
outputFile = open(outputFileName, "w")

inputFileName = "trips.txt"

cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")

        trip_id = row[2]
        trip_headsign = row[3]
        direction_id = row[4]
        block_id = row[5]

        if cnt>1:
            outputFile.write('<' + trip_idURI + '/'+trip_id+ '> ' + 'a' + ' <' +trip_idURI+ '>'+string_+'\n')
            outputFile.write('<' + trip_headsignURI + '/'+trip_headsign+ '> ' + 'a' + ' <' + trip_headsignURI+ '>'+string_+'\n')
            outputFile.write('<' + direction_idURI + '/'+direction_id+ '> ' + 'a' + ' <' + direction_idURI+ '>'+integer_+'\n')
            outputFile.write('<' + block_idURI + '/'+block_id+ '> ' + 'a' + ' <' + block_idURI+ '>'+integer_+'\n')

print(cnt)
```

Ας δούμε αναλυτικά τα αποτελέσματα

Σχετικά με το αρχείο stop\_times.txt

Αρχικό αρχείο stop\_times.txt:

```
trip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,drop_off_type
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:30:00,05:30:00,400045,1,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:31:00,05:31:00,400044,2,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:33:00,05:33:00,320025,3,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:35:00,05:35:00,240119,4,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:36:00,05:36:00,240038,5,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:38:00,05:38:00,380117,6,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:39:00,05:39:00,380116,7,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:40:00,05:40:00,380115,8,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:42:00,05:42:00,380113,9,0,0
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> a <http://www.example.com/stop_times>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/arrival_time> "05:30:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/departure_time> "05:30:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/stop_sequence> "1"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/pickup_type> "0"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/drop_off_type> "0"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> a <http://www.example.com/stop_times>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/arrival_time> "05:31:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/departure_time> "05:31:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/stop_sequence> "2"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/pickup_type> "0"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/drop_off_type> "0"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> a <http://www.example.com/stop_times>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/arrival_time> "05:33:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/departure_time> "05:33:00"^^xsd:dateTime.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/stop_sequence> "3"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/pickup_type> "0"^^xsd:int.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/drop_off_type> "0"^^xsd:int.
```

Η μετατροπή έγινε με χρήση του εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

stop_timesURI = "http://www.example.com/stop_times"
tripURI = "http://www.example.com/trip_id"
arrival_timeURI = "http://www.example.com/arrival_time"
departure_timeURI = "http://www.example.com/departure_time"
stop_idURI = "http://www.example.com/stop_id"
stop_sequenceURI = "http://www.example.com/stop_sequence"
pickup_typeURI = "http://www.example.com/pickup_type"
drop_off_typeURI = "http://www.example.com/drop_off_type"

string_ = "^^xsd:string."
integer_ = "^^xsd:int."
time_ = "^^xsd:dateTime."

outputFileName = "stop_timesOUT.txt"
outputFile = open(outputFileName, "w")

inputFileName = "stop_times.txt"

cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")
        trip_id = row[0]
        arrival_time = row[1]
        departure_time = row[2]
        stop_id = row[3]
        stop_sequence = row[4]
        pickup_type = row[5]
        drop_off_type = row[6]
        if cnt>1:
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + 'a' + ' ' + '<' + stop_timesURI + '>' + '\n')
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + '<' + arrival_timeURI + '>' + ' ' + str(arrival_time)+ ' ' + time_ + '\n')
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + '<' + departure_timeURI + '>' + ' ' + str(departure_time)+ ' ' + time_ + '\n')
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + '<' + stop_sequenceURI + '>' + ' ' + str(stop_sequence)+ ' ' + integer_ + '\n')
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + '<' + pickup_typeURI + '>' + ' ' + str(pickup_type)+ ' ' + integer_ + '\n')
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+stop_id+'>' + ' ' + '<' + drop_off_typeURI + '>' + ' ' + str(drop_off_type).rstrip() + ' ' + integer_ + '\n')
print(cnt)
```

Ας δούμε αναλυτικά τα αποτελέσματα

Σχετικά με το αρχείο calendar.txt

Αρχικό αρχείο calendar.txt:

```
service_id,monday,tuesday,wednesday,thursday,friday,saturday,sunday,start_date,end_date
CALEND-021-Καθημερινή-06,1,1,1,1,1,0,0,20161201,20170630
CALEND-021-Κυριακή-05,0,0,0,0,0,1,0,20161204,20170625
CALEND-021-Σάββατο-06,0,0,0,0,1,0,20161203,20170701
CALEND-022-Καθημερινή-06,1,1,1,1,1,0,0,20161201,20170630
CALEND-022-Κυριακή-05,0,0,0,0,0,1,0,20161204,20170625
CALEND-022-Σάββατο-05,0,0,0,0,1,0,20161203,20170701
CALEND-024-Καθημερινή-11,1,1,1,1,1,0,0,20161201,20170630
CALEND-024-Κυριακή-10,0,0,0,0,0,1,0,20161204,20170625
CALEND-024-Σάββατο-11,0,0,0,0,1,0,20161203,20170701
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> a <http://www.example.com/calendar>.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/monday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/tuesday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/wednesday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/thursday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/friday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/saturday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/sunday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/start_date> "2016-12-01"^^xsd:date.
<http://www.example.com/calendar/CALEND-021-Καθημερινή-06> <http://www.example.com/end_date> "2017-06-30"^^xsd:date.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> a <http://www.example.com/calendar>.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/monday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/tuesday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/wednesday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/thursday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/friday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/saturday> "0"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/sunday> "1"^^xsd:boolean.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/start_date> "2016-12-04"^^xsd:date.
<http://www.example.com/calendar/CALEND-021-Κυριακή-05> <http://www.example.com/end_date> "2017-06-25"^^xsd:date.
```

Η μετατροπή έγινε με χρήση των εξής κώδικα σε python3:

```
#!/usr/bin/env python3
import csv
calendarURI = "http://www.example.com/calendar"
mondayURI = "http://www.example.com/monday"
tuesdayURI = "http://www.example.com/tuesday"
wednesdayURI = "http://www.example.com/wednesday"
thursdayURI = "http://www.example.com/thursday"
fridayURI = "http://www.example.com/friday"
saturdayURI = "http://www.example.com/saturday"
sundayURI = "http://www.example.com/sunday"
start_dateURI = "http://www.example.com/start_date"
end_dateURI = "http://www.example.com/end_date"
string_ = "^^xsd:string."
integer_ = "^^xsd:int."
boolean_ = "^^xsd:boolean."
date_ = "^^xsd:date."
outputFileName = "calendarOUTNEW.txt"
outputFile = open(outputFileName, "w")
inputFileName = "calendar.txt"
cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")
        service_id = row[0]
        monday = row[1]
        tuesday = row[2]
        wednesday = row[3]
        thursday = row[4]
        friday = row[5]
        saturday = row[6]
        sunday = row[7]
        s = row[8]
        f = row[9]
        start_day_final = str(s[0:4]+ '-' + s[4:6]+ '-' + s[6:8])
        end_date_final = str(f[0:4]+ '-' + f[4:6]+ '-' + f[6:8])
        if cnt>1:
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + 'a' + ' ' + '<' + calendarURI+ '> ' + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + mondayURI + '> ' + ' ' + str(monday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + tuesdayURI + '> ' + ' ' + str(tuesday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + wednesdayURI + '> ' + ' ' + str(wednesday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + thursdayURI + '> ' + ' ' + str(thursday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + fridayURI + '> ' + ' ' + str(friday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + saturdayURI + '> ' + ' ' + str(saturday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + sundayURI + '> ' + ' ' + str(sunday)+ ' ' + boolean_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + start_dateURI + '> ' + ' ' + str(start_day_final)+ ' ' + date_ + '\n')
            outputFile.write('<' + calendarURI + '/'+service_id+ '> ' + '1' + '<' + end_dateURI + '> ' + ' ' + str(end_date_final)+ ' ' + date_ + '\n')
```

Ας δούμε αναλυτικά τα αποτελέσματα

Σχετικά με το αρχείο agency.txt

Αρχικό αρχείο agency.txt:

```
agency_name,agency_url,agency_timezone,agency_lang,agency_phone  
"Οργανισμός Αστικών Συγκοινωνιών ΟΑΣΑ - Athens Urban Transport Organisation",http://www.oasa.gr,Europe/Athens/el,+302108200999
```

Το αρχείο είναι πολύ ευσύνοπτο επομένως δεν θα γραφεί κώδικας για μετατροπή του σε RDF

τριάδες, αλλά η μετατροπή θα γίνει με το χέρι για λογους εξοικείωσης.

Αξίζει να τονιστεί ότι κανονικά θα έπρεπε να υπάρχει και η κολώνα agency\_id, αλλά ο σχεδιαστής της βάσης έχει επιλέξει επειδή υπάρχει μόνο μια καταγραφή να μην δώσει τέτοια πληροφορία. Εμεις θα ορίσουμε ως ID για τον ΟΑΣΑ το 1.

Έτσι έχουμε τα εξής αποτελέσματα σε RDF τριάδες:

```
<http://www.example.com/agency/1> a <http://www.example.com/agency>.  
<http://www.example.com/agency/1> <http://www.example.com/agency_name> "Οργανισμός Αστικών Συγκοινωνιών ΟΑΣΑ - Athens Urban Transport Organisation"^^xsd:string.  
<http://www.example.com/agency/1> <http://www.example.com/agency_url> "http://www.oasa.gr"^^xsd:string.  
<http://www.example.com/agency/1> <http://www.example.com/agency_timezone> "Europe/Athens"^^xsd:string.  
<http://www.example.com/agency/1> <http://www.example.com/agency_lang> "el"^^xsd:string.  
<http://www.example.com/agency/1> <http://www.example.com/agency_phone> "302108200999"^^xsd:int.
```

## Δημιουργία τριάδων RDF για Object Properties

Αρχικά αρκεί να σχεδιάσουμε μόνο τις ευθείς σχέσεις και η αντίστροφες θα σχεδιαστούν μόνες τους με εφαρμογή των ιδιοτήτων της οντολογίας.

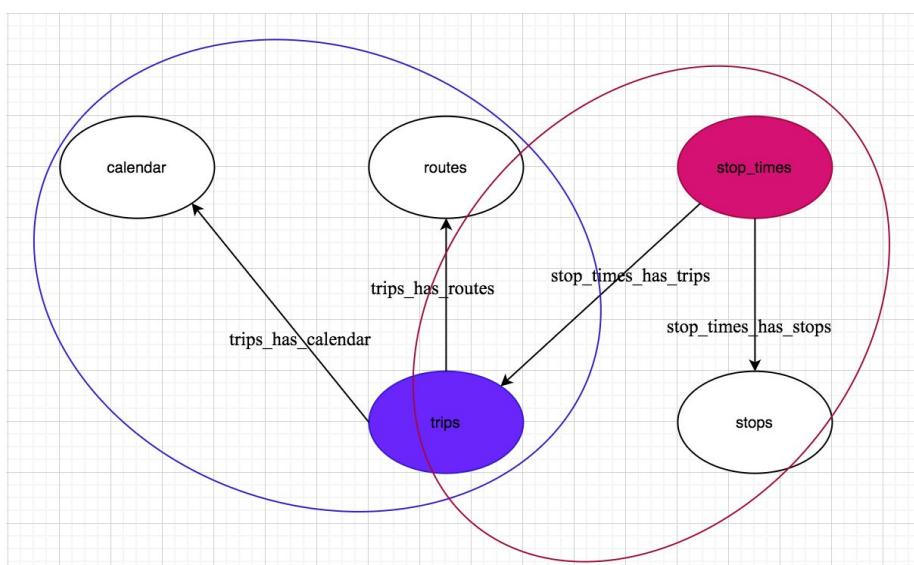
Γενικά για τον σχεδιασμό των object properties για μια κλάση χρειαζόμαστε πληροφορίες για το ID των καταγραφών της κλάσεις και το ID των καταγραφών των κλάσεων που θέλουμε να το συσχετίσουμε.

Παρατηρούμε το εξής:

Στο αρχείο trips.txt οπου αναφέρεται στη κλάση trips και έχουμε σαν ID το trips\_id έχουμε και foreing key το service\_id, που είναι το Primary Key της κλάσης calendar, και το routes\_id που είναι το Primary Key της κλάσης routes. Επομένως έτσι διαβάζοντας το αρχείο trips.txt μπορούμε να ορίσουμε τις σχέσεις trips\_has\_calendar και trips\_has\_routes.

Στο αρχείο stop\_times.txt οπου αναφέρεται στη κλάση stop\_times και έχουμε το {trip\_id, stop\_id} έχουμε και foreing key το stops\_id, που είναι το Primary Key της κλάσης stops, και το trips\_id που είναι το Primary Key της κλάσης trips. Επομένως έτσι διαβάζοντας το αρχείο stop\_times.txt μπορούμε να ορίσουμε τις σχέσεις stop\_times\_has\_stops και stop\_times\_has\_stops.

Αυτό φαίνεται και γραφικά στο εξής σχήμα:

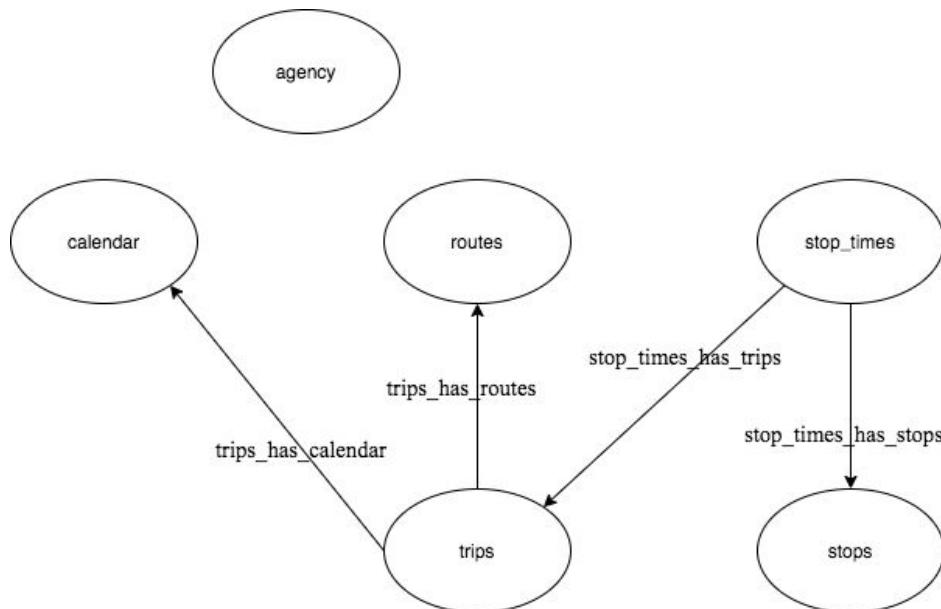


Σχετικά με την σχέση routes\_has\_agency, δεδομένου ότι έχουμε μόνο μια agency σε αυτό το σύστημα θεωρήθηκε **υπολογιστική σπατάλη** να συσχετίσουμε όλα τα δρομολόγια με την ίδια agency.

Επομένως θεωρείται πιο αποδοτικό για το σύστημα να μην προσθέσουμε τόσες RDF τριάδες που λένε ακριβώς το ίδιο πράγμα, δηλαδή ότι ένα δρομολόγιο έχει την ίδια agency με όλα τα υπόλοιπα.

Έτσι στο σύστημα θα υπάρχει η πληροφορία για το ποια είναι η agency αλλά δεν αξίζει να υπάρχει η διασύνδεση της agency με την routes για οικονομια και έτσι α κερδίσουμε και ταχύτητα στο σύστημα.

Επομένως το τελικό Σύστημα θα μοιάζει ως εξής (παραλείπονται οι data properties για να είναι το σχήμα πιο ευανάγνωστο).



### Γενικός Σχολιασμός Κώδικα

Αρχικά ορίζουμε τα κατάλληλα τμήματα uri που θέλουμε να χρησιμοποιήσουμε. Έπειτα δημιουργούμε ένα αρχείο εξόδου με δικαίωμα εγγραφής και ανοίγουμε το αρχείο εισόδου.

Διαβάζουμε γραμμή γραμμή το αρχείο εισόδου και σπάμε την κάθε γραμμή σε έναν πίνακα όπου κάθε κελί του πίνακα περιέχει οτιδήποτε υπάρχει ανάμεσα σε κόμματα. Εντοπίζουμε τις χρήσιμες πληροφορίες από την κάθε γραμμή και γράφουμε στο άλλο αρχείο αυτά που επιθυμούμε με βάση τις πληροφορίες που διαβάσαμε και τα uri που γράψαμε στην αρχή.

Η python3 σε Ubuntu 18.04 κατάφερε να διαβάσει όλα τα αρχεία χωρίς να δημιουργήσει κάποιο πρόβλημα ή ξεχνόντας γραμμές (αυτός ο έλεγχος γίνεται με την μεταβλητή cnt)

Ας δούμε αναλυτικά τα αποτελέσματα

Σχετικά με το αρχείο trips.txt

Αρχικό αρχείο trips.txt:

```
route_id,service_id,trip_id,trip_headsign,direction_id,block_id,shape_id
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184458,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803639-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184460,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803640-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184459,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803641-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184461,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803642-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184462,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803643-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184462,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803644-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184458,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803645-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184460,
T3-20,ΤΗΛΕΜΑ-T3-Παρασκευή-02,9803646-ΤΗΛΕΜΑ-T3-Παρασκευή-02,ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ,0,2184459,
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_routes> <http://www.example.com/routes/T3-20>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_calendar> <http://www.example.com/calendar/ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803639-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_routes> <http://www.example.com/routes/T3-20>.
<http://www.example.com/trips/9803639-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_calendar> <http://www.example.com/calendar/ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_routes> <http://www.example.com/routes/T3-20>.
<http://www.example.com/trips/9803640-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_calendar> <http://www.example.com/calendar/ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_routes> <http://www.example.com/routes/T3-20>.
<http://www.example.com/trips/9803641-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_calendar> <http://www.example.com/calendar/ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803642-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_routes> <http://www.example.com/routes/T3-20>.
<http://www.example.com/trips/9803642-ΤΗΛΕΜΑ-T3-Παρασκευή-02> <http://www.example.com/trips_has_calendar> <http://www.example.com/calendar/ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
```

Η μετατροπή έγινε με χρήση των εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

trip_idURI = "http://www.example.com/trips"

trips_has_routes = "http://www.example.com/trips_has_routes"
trips_has_calendar = "http://www.example.com/trips_has_calendar"

routesURI = "http://www.example.com/routes"
calendarURI = "http://www.example.com/calendar"

outputFileName = "trips_RN_OUT.txt"
outputFile = open(outputFileName, "w")

inputFileName = "trips.txt"

cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")

        rout_id = row[0]
        service_id = row[1]
        trip_id = row[2]

        if cnt>1:
            outputFile.write('<' + trip_idURI + '>' + trip_id + '>' + '<' + trips_has_routes + '>' + '<' + routesURI + '>' + rout_id + '>' + '\n')
            outputFile.write('<' + trip_idURI + '>' + trip_id + '>' + '<' + trips_has_calendar + '>' + '<' + calendarURI + '>' + service_id + '>' + '\n')

print(cnt)
```

Σχετικά με το αρχείο stop\_times.txt

Αρχικό αρχείο stop\_times.txt:

```
trip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,drop_off_type
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:30:00,05:30:00,400045,1,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:31:00,05:31:00,400044,2,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:33:00,05:33:00,320025,3,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:35:00,05:35:00,240119,4,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:36:00,05:36:00,240038,5,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:38:00,05:38:00,380117,6,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:39:00,05:39:00,380116,7,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:40:00,05:40:00,380115,8,0,0
9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02,05:42:00,05:42:00,380113,9,0,0
```

Αποτέλεσμα σε RDF:

```
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/stop_times_has_trips> <http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400045> <http://www.example.com/stop_times_has_stops> <http://www.example.com/stops/400045>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/stop_times_has_trips> <http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/400044> <http://www.example.com/stop_times_has_stops> <http://www.example.com/stops/400044>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/stop_times_has_trips> <http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/320025> <http://www.example.com/stop_times_has_stops> <http://www.example.com/stops/320025>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/240119> <http://www.example.com/stop_times_has_trips> <http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/240119> <http://www.example.com/stop_times_has_stops> <http://www.example.com/stops/240119>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/240038> <http://www.example.com/stop_times_has_trips> <http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02>.
<http://www.example.com/trips/9803638-ΤΗΛΕΜΑ-T3-Παρασκευή-02/stops/240038> <http://www.example.com/stop_times_has_stops> <http://www.example.com/stops/240038>.
```

Η μετατροπή έγινε με χρήση των εξής κώδικα σε python3:

```
#!/usr/bin/env python3

import csv

stop_timesURI = "http://www.example.com/stop_times"
stop_times_has_trips = "http://www.example.com/stop_times_has_trips"
stop_times_has_stops = "http://www.example.com/stop_times_has_stops"

trip_idURI = "http://www.example.com/trips"
stop_idURI = "http://www.example.com/stops"

outputFileName = "stop_times_RN_OUT.txt"
outputFile = open(outputFileName, "w")

inputFileName = "stop_times.txt"

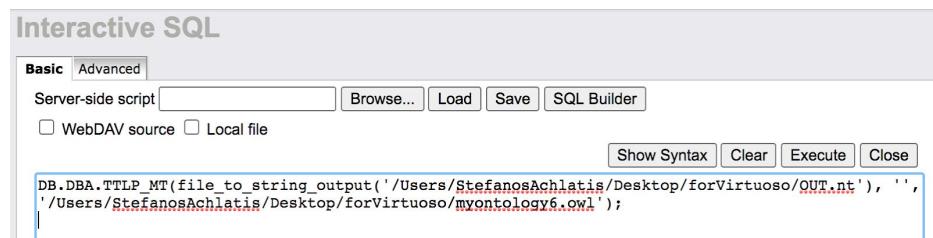
cnt = 0
with open(inputFileName) as infile:
    for line in infile:
        cnt = cnt + 1
        row = line.split(",")
        trip_id = row[0]
        stop_id = row[3]
        if cnt>1:
            outputFile.write('<http://www.example.com/trips/'+trip_id+'/stops/'+ stop_id +'> ' + '<' + stop_times_has_trips + '>' + ' ' + '<' + trip_idURI + '/' + trip_id + '>.' + '\n')
            outputFile.write('<' + http://www.example.com/trips/' + trip_id + '/stops/' + stop_id + '> ' + '<' + stop_times_has_stops + '>' + ' ' + '<' + stop_idURI + '/' + stop_id + '>.' + '\n')

print(cnt)
```

## Εισαγωγή των δεδομένων στο virtuoso

1. Αρχικά δημιουργούμε ένα directory στο Desktop με όνομα forVirtuoso.
2. Πηγαίνουμε στο directory: “/Applications/Virtuoso Open Source Edition v7.2.app/Contents/virtuoso-opensource/database”
3. Ανοίγουμε το αρχείο virtuoso.ini όπου δίνουμε “άδεια” στο Virtuoso να διαβάζει από το αρχείο forVirtuoso γράφοντας το εξής:

```
AllowOSCalls          = 0
SchedulerInterval     = 10
DirsAllowed           = ., ../vad, /Users/StefanosAchlatis/Desktop/forVirtuoso
ThreadCleanupInterval = 0
ThreadThreshold        = 10
```
4. Ενεργοποιούμε το Virtuoso κάνοντας log in
5. Ανοίγουμε το iSQL του Virtuoso και πληκτρολογούμε την εξής εντολή και πατάμε Execute: Όπου OUT.nt είναι το αρχείο με όλες τις RDF τριαδες που δημιουργήσαμε πριν και myontology6.owl είναι το αρχείο με την οντολογία όπως δημιουργήθηκε από το Protege.



6. Αν υπάρξει κάποιο σφάλμα θα υπάρχει σχετική ενημέρωση εναλλακτικά η διαδικασία ολοκληρώθηκε.

Μπορούμε να δούμε τον γράφο γνώσης εδώ:

The screenshot shows the 'RDF Document (Named Graph) Management' interface. It has tabs for 'Home', 'System Admin', 'Database', 'Replication', 'Web Application Server', 'XML', 'Web Services', 'Linked Data', and 'NNTP'. Below these are sub-tabs: 'SPARQL', 'Sponger', 'Statistics', 'Graphs', 'Schemas', 'Namespaces', 'Views', 'Quad Store Upload'. The 'Graphs' tab is selected. A table lists various named graphs with their URLs. One URL, '/Users/StefanosAchlatis/Desktop/dataStamou/myontology6.owl', is highlighted with a yellow background.

Graph
http://www.openlinksw.com/schemas/virtrdf#
http://www.w3.org/ns/ldp#
http://localhost:8890/sparql
http://localhost:8890/DAV/
http://www.w3.org/2002/07/owl#
http://localhost:8890/DAV/
/Users/StefanosAchlatis/Desktop/dataStamou/teliko4.owl
/Users/StefanosAchlatis/Desktop/dataStamou/onotologyFinal.owl
http://www.w3.org/ns/auth/cert#
/Users/StefanosAchlatis/Desktop/dataStamou/teliko2.owl
http://vocab.gtfs.org/gtfs
/Users/StefanosAchlatis/Desktop/dataStamou/teliko.owl
/Users/StefanosAchlatis/Desktop/dataStamou/myontology6.owl

## Μέρος Γ: Ερωτήματα SPARQL

### Ερώτημα 1

**Σκοπός:** Εδώ θα δείξουμε διαδρομές πάνω στο γράφο γνώσης και χρήση του LinkedGeoData για να βγάλουμε σημασιολογικά αποτελέσματα με βάση την γεωγραφική θέση.

**Ερώτηση:** Εύρεση όλων των στάσεων του ΟΑΣΑ σε ακτίνα 300 μέτρων από ένα συγκεκριμένο γεωγραφικό σημείο

**SPARQL και Αποτελέσματα:**

Default Graph IRI    /Users/StefanosAchlatidis/Desktop/dataStamou/ontologyFinal.owl

Query

```
SELECT DISTINCT STR(?stop_name) AS ?stop_name,STR(?stop_point) AS ?stop_point
WHERE{
  ?stop_id
    a <http://www.example.com/stops>;
    <http://www.example.com/stop_name> ?stop_name;
    <http://www.example.com/stop_point> ?stop_point.

    Filter(bif:st_intersects (?stop_point, bif:st_point (37.933483,23.710630), 0.3)).
}
```

Execute Save Load Clear

stop_name	stop_point
ΣΤΑΔΙΟΥ	POINT(37.93387789232 23.712731368075)
ΕΥΑΓΓΕΛΙΚΗ ΣΧΟΛΗ	POINT(37.933098049599 23.710799896671)
ΠΑΛ. ΤΕΡΜΑ	POINT(37.935001044616 23.711315909516)

## Ερώτημα 2

**Σκοπός:** Εδώ θα δείξουμε βαθιές διαδρομές πάνω στο γράφο γνώσης και χρήση του LinkedGeoData για να βγάλουμε σημασιολογικά αποτελέσματα με βάση την γεωγραφική θέση. Επίσης χρησιμοποιείται και ο χρόνος και τα χρονικά διαστήματα.

**Ερώτηση:** Είναι Παρασκευή στις 5:30 και ο χρήστης βρίσκεται σε γνωστή τοποθεσία και θέλει να βρει όλες τις στάσεις εντός ακτίνας 500 μέτρων από την οποία φεύγει οποιοδήποτε μέσο τα επόμενα 15 λεπτά.

**SPARQL και Αποτελέσματα:**

Default Graph IRI    /Users/StefanosAchlatis/Desktop/dataStamou/ontologyFinal.owl

Query

```
SELECT DISTINCT (STR(?stop_name) AS ?stop_name)
WHERE{
?stop_times <http://www.example.com/arrival_time> ?time;
<http://www.example.com/stop_times_has_stops> ?stop;
<http://www.example.com/stop_times_has_trips> ?trip.
?stop <http://www.example.com/stop_point> ?stop_point;
<http://www.example.com/stop_name> ?stop_name.
?trip <http://www.example.com/trips_has_calendar> ?calendar.
?calendar <http://www.example.com/friday> "1"^^xsd:boolean.

Filter(bif:st_intersects (?stop_point, bif:st_point (37.9444968100465, 23.6687927494187), 0.5)).
Filter(?time < "05:45:00").
Filter(?time > "05:30:00").
}
```

Execute Save Load Clear

stop_name
ΝΕΟ ΦΑΛΗΡΟ
ΗΣΑΠ Ν.ΦΑΛΗΡΟΥ
ΑΚΤΑΙΟΝ
ΣΚΑΙ

### Ερώτημα 3

**Σκοπός:** Εδώ θέλουμε να δείξουμε βασικές διαδρομές στο γράφο και να μελετήσουμε έτοιμες συναρτησεις της SPARQL, πιο αναλυτικά την συνάρτηση regex(), που κάνει έλεγχο αν η συμβολοσειρά που της έχει δοθεί ως όρισμα υπάρχει ως υποσύνολο στην συμβολοσειρά που κάνει έλεγχο.

**Ερώτηση:** Θέλουμε να βρούμε όλες τις στάσεις που έχουν ως σημείο αναφοράς την Βούλα SPARQL και Αποτελέσματα:

Default Graph IRI /Users/StefanosAchlatitis/Desktop/dataStamou/ontologyFinal.owl

Query

```
SELECT DISTINCT (STR(?o) AS ?route_long_name)
WHERE{
  ?s a <http://www.example.com/routes>;
    <http://www.example.com/route_long_name> ?o.
  FILTER(regex(?o, "ΒΟΥΛΑ")).
}
```

Execute Save Load Clear

route_long_name
ΝΕΟ ΦΑΛΗΡΟ - ΑΣΚΛΗΠΕΙΟ ΒΟΥΛΑ
ΣΤ. ΕΛΛΗΝΙΚΟ - ΑΝΩ ΓΛΥΦΑΔΑ (ΠΑΝΟΡΑΜΑ ΒΟΥΛΑΣ)
ΒΟΥΛΑ - ΠΑΝΟΡΑΜΑ (ΚΥΚΛΙΚΗ)
ΠΕΙΡΑΙΑΣ - ΒΟΥΛΑ
ΑΚΑΔΗΜΙΑ - ΒΟΥΛΑ (Μέσω Λ. ΑΜΦΙΘΕΑΣ)
ΣΥΝΤΑΓΜΑ - ΒΟΥΛΑ

#### Ερώτημα 4

**Σκοπός:** Εδώ θέλουμε να δείξουμε βαθίες διαδρομές στο γράφο και να μελετήσουμε έτοιμες συναρτησεις της SPARQL, πιο αναλυτικά την συνάρτηση `regex()`, που κάνει έλεγχο αν η συμβολοσειρά που της έχει δοθεί ως όρισμα υπάρχει ως υποσύνολο στην συμβολοσειρά που κάνει έλεγχο. Θα μελετήσουμε και τον χρόνο και την ημέρα.

Ταυτόχρονα αυτο το query αποσκοπεί στο να δείξει βαθύτερη κατανόηση του dataset.

**Ερώτηση:** Θέλουμε να βρούμε όλα τα νυκτά δρομολόγια λεωφορείων (μόνο λεωφορείων) μεταξύ των ωρών 00:00:00 - 02:30:00 τα έχουν Τερματικό Σταθμό το Σύνταγμα

**Προσοχή:** Στο dataset η ώρες μετά τις 12 το βράδυ αναγράφονται ως την ώρα + 24, δηλαδή η μια το βράδυ αναγράφεται ως 25:00:00.

**SPARQL και Αποτελέσματα:**

Default Graph IRI    /Users/StefanosAchlatis/Desktop/dataStamou/ontologyFinal.owl

Query

```
SELECT DISTINCT STR(?name) AS ?route_name
WHERE{
    ?route a <http://www.example.com/routes>;
        <http://www.example.com/route_type> "3"^^xsd:int;
        <http://www.example.com/route_long_name> ?name.
    ?trip <http://www.example.com/trips_has_routes> ?route.
    ?stop_times <http://www.example.com/stop_times_has_trips> ?trip.
    ?trip <http://www.example.com/trips_has_calendar> ?calendar.
    ?calendar <http://www.example.com/friday> "1"^^xsd:boolean.
    ?stop_times <http://www.example.com/arrival_time> ?time.

    Filter(?time > "24:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>).
    Filter(?time < "26:30:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>).
    FILTER(regex(?name,"ΣΥΝΤΑΓΜΑ"))
}
```

Execute Save Load Clear

route_name
ΠΕΙΡΑΙΑΣ - ΣΥΝΤΑΓΜΑ
ΣΥΝΤΑΓΜΑ - ΑΕΡΟΛ. ΑΘΗΝΩΝ (EXPRESS)
ΣΥΝΤΑΓΜΑ - ΚΗΦΙΣΙΑ

## Ερώτημα 5

**Σκοπός:** Εδώ θέλουμε να δείξουμε μεσαίου μεγέθους διαδρομές στο γράφο και να μελετήσουμε έτοιμες συναρτησεις της SPARQL όπως την LIMIT.

**Ερώτηση:** Θέλουμε να βρούμε όλα τα ονόματα 5 στάσεων που είναι στάσεις αφετηρίας SPARQL και Αποτελέσματα:

Default Graph IRI    /Users/StefanosAchlatis/Desktop/dataStamou/ontologyFinal.owl

Query

```
SELECT DISTINCT (STR(?stop_name) AS ?stop_name)
WHERE{
    ?stops a <http://www.example.com/stops>;
        <http://www.example.com/stop_name> ?stop_name.
    ?stop_times <http://www.example.com/stop_times_has_stops> ?stops;
        <http://www.example.com/stop_sequence> ?stop_sequence.

    FILTER(?stop_sequence = "1"^^xsd:int).

}
LIMIT 5
```

Execute Save Load Clear

stop_name
ΣΥΝΤΑΓΜΑ
ΣΤΑΔΙΟ ΕΙΡΗΝΗΣ-ΦΙΛΙΑΣ ΣΕΦ
ΑΣΚΛΗΠΙΕΙΟ ΒΟΥΛΑΣ
2η ΑΓ.ΚΟΣΜΑ
ΑΓΙΟΣ ΑΛΕΞΑΝΔΡΟΣ

## Ερώτημα 6

**Σκοπός:** Εδώ θέλουμε να δείξουμε βαθιές διαδρομές στο γράφο και να μελετήσουμε την συνάρτηση UNION,LIMIT,OFFSET όπως επίσης να προ-επεξεργαστούμε τα δεδομένα άλλων ερωτημάτων.

**Ερώτηση:** Τα ονοματα 10 δρομολογίων που κάνουν στάση είτε στο Σύνταγμα είτε στο Ζάππειο.

Θέλουμε 5 που κάνουν στάση στο Σύνταγμα και άλλα 5 που κάνουν στάση στο Ζάππειο , μπορεί κάποια δρομολόγια να κάνουν στάση και στις 2.

**Προσοχή:** Θα χρησιμοποιήσουμε το UNION για να της δυο στάσεις ανεξάρτητα. Τώρα θα αναλύσουμε το πως βρίσκουμε τα 5 και 5. Σκεφτόμαστε το εξής, αρχικά κάνουμε query μόνο για τα δρομολόγια που κάνουν στάση στο Ζαππειο και βλέπουμε ότι είναι συνολικά 8 δρομολόγια. Η σειρά που γίνεται το UNION έχει σημασία, οπότε ρωτάμε πρώτα για το Ζάππειο και κανουμε offset 3 δηλαδή πετάμε τις 3 πρώτες απαντήσεις και άρα έχουμε 5 απαντησεις για το Ζάππειο, και μετά παίρνουμε τις υπόλοιπες απαντήσεις από το Σύνταγμα, αλλά κρατάμε συνολικά 10, άρα από το συνταγμα θα κρατήσουμε 5.

**SPARQL και Αποτελέσματα:**

```
SELECT DISTINCT STR(?route_name) AS ?route_name
WHERE{
{
?rout a <http://www.example.com/routes>;
<http://www.example.com/route_long_name> ?route_name.
?trips <http://www.example.com/trips_has_routes> ?rout.
?stop_times <http://www.example.com/stop_times_has_trips> ?trips;
<http://www.example.com/stop_times_has_stops> ?stops.
?stops <http://www.example.com/stop_name> ?stop_name.
FILTER(regex(?stop_name, "ΖΑΠΠΕΙΟ"))
}
UNION
{
?rout a <http://www.example.com/routes>.
?rout <http://www.example.com/route_long_name> ?route_name.
?trips <http://www.example.com/trips_has_routes> ?rout.
?stop_times <http://www.example.com/stop_times_has_trips> ?trips.
?stop_times <http://www.example.com/stop_times_has_stops> ?stops.
?stops <http://www.example.com/stop_name> ?stop_name.
FILTER(regex(?stop_name, "ΣΥΝΤΑΓΜΑ"))
}
}
OFFSET 3
LIMIT 10
```

Execute Save Load Clear

route_name
ΖΑΠΠΕΙΟ - ΠΕΡΙΣΤΕΡΙ
ΛΑΜΠΡΙΝΗ - ΠΛ. ΣΥΝΤΑΓΜΑΤΟΣ - ΤΖΙΤΖΙΦΙΕΣ
ΠΕΙΡΑΙΑΣ - ΣΥΝΤΑΓΜΑ
ΤΕΡΨΙΘΕΑ - ΑΡΓΥΡΟΥΠΟΛΗ - ΣΥΝΤΑΓΜΑ (ΚΥΚΛΙΚΗ)
ΜΕΤΑΜΟΡΦΩΣΗ - ΠΛΑΤΕΙΑ ΣΥΝΤΑΓΜΑΤΟΣ (ΚΥΚΛΙΚΗ)
ΑΝΘΟΥΠΟΛΗ - ΣΤ. ΕΛΛΗΝΙΚΟ
ΑΓΙΑ ΜΑΡΙΝΑ - ΔΟΥΚ. ΠΛΑΚΕΝΤΙΑΣ - ΑΕΡΟΔΡΟΜΙΟ
Α. ΠΑΤΗΣΙΑ - Ν. ΠΑΓΚΡΑΤΙ - Ν. ΕΛΒΕΤΙΑ
ΠΕΤΡΑΛΩΝΑ - ΔΙΚΑΣΤΗΡΙΑ - ΕΛ. ΒΕΝΙΖΕΛΟΥ
ΑΝΩ ΚΥΨΕΛΗ - ΠΑΓΚΡΑΤΙ - ΚΑΙΣΑΡΙΑΝΗ

### Ερώτημα 7:

Σκοπός: Βαθιές διαδρομές σε γράφο γνώσης και χρήση πολλών data properties

Ερώτηση: Βρές 5 δρομολόγια τρόλεϊ που δουλεύουν όλες τις καθημερινές

SPARQL και Αποτελέσματα:

Default Graph IRI /Users/StefanosAchlatis/Desktop/dataStamou/onotologyFinal.owl

#### Query

```
SELECT DISTINCT STR(?name) AS ?route_name
WHERE {
    ?calendar a <http://www.example.com/calendar>;
        <http://www.example.com/monday> "1"^^xsd:boolean;
        <http://www.example.com/tuesday> "1"^^xsd:boolean;
        <http://www.example.com/wednesday> "1"^^xsd:boolean;
        <http://www.example.com/thursday> "1"^^xsd:boolean;
        <http://www.example.com/friday> "1"^^xsd:boolean.
    ?trips <http://www.example.com/trips_has_calendar> ?calendar.
    ?trips <http://www.example.com/trips_has_routes> ?routes.
    ?routes <http://www.example.com/route_type> "800"^^xsd:int;
        <http://www.example.com/route_long_name> ?name.
}
LIMIT 5
```

Execute Save Load Clear

route_name
ΠΛΑΤ. ΑΤΤΙΚΗΣ - ΤΖΙΤΖΙΦΙΕΣ - ΜΟΣΧΑΤΟ
ΤΖΙΤΖΙΦΙΕΣ - ΧΑΛΑΝΔΡΙ
Α. ΠΑΤΗΣΙΑ - Ν. ΠΑΓΚΡΑΤΙ - Ν. ΕΛΒΕΤΙΑ
ΖΑΠΠΕΙΟ - ΠΕΡΙΣΤΕΡΙ
ΠΕΤΡΑΛΩΝΑ - ΔΙΚΑΣΤΗΡΙΑ - ΕΛ. ΒΕΝΙΖΕΛΟΥ

### Ερώτημα 8:

**Σκοπός:** Δεδομένου ότι τα δεδομένα είναι λίγα, θα κάνουμε ερωτήματα σε δεδομένα που δεν υπάρχουν για να δείξουμε και άλλα τμήματα της βάσης μας.

**Ερώτηση:** Δείξε μου όλες τις στάσεις και τα κόμιστρα για δρομολόγια που περνάνε κοντά από μια συγκεκριμένη περιοχή.

**SPARQL και Αποτελέσματα:**

```
Query
SELECT DISTINCT STR(?price) AS ?price, STR(?stop_name) AS ?stop_name
WHERE{
{
    ?rout a <http://www.example.com/routes>;
           <http://www.example.com/route_long_name> ?route_name.
    ?trips <http://www.example.com/trips_has_routes> ?rout.
    ?stop_times <http://www.example.com/stop_times_has_trips> ?trips;
           <http://www.example.com/stop_times_has_stops> ?stops.
    ?stops <http://www.example.com/stop_name> ?stop_name.
    ?stops <http://www.example.com/stop_point> ?stop_point.
    ?rout a <http://www.example.com/routes_has_fares> ?fares.
    ?fares <http://www.example.com/price> ?price.

Filter(bif:st_intersects (?stop_point, bif:st_point (37.9444968100465, 23.6687927494187), 0.5)).
}
```

### Ερώτημα 9:

**Σκοπός:** Δεδομένου ότι τα δεδομένα είναι λίγα, θα κάνουμε ερωτήματα σε δεδομένα που δεν υπάρχουν για να δείξουμε και άλλα τμήματα της βάσης μας.

**Ερώτηση:** Δείξε μου όλα τα δρομολόγια που τα μέσα που εκτελούν τα δρομολόγια είναι φτιαγμένα μετα το 2014.

**SPARQL και Αποτελέσματα:**

```
Query
SELECT DISTINCT STR(?route_name) AS ?route_name,
WHERE{
{
    ?routes a <http://www.example.com/routes>;
           <http://www.example.com/route_long_name> ?route_name.
    ?routes <http://www.example.com/routes_has_oxima> ?oxima.
    ?oxima <http://www.example.com/etos_paragogis> ?etos_paragogis

Filter(?etos_paragogis > "2014"^^xsd:int).
}
```

## Ερώτημα 10:

**Σκοπός:** Θα γίνει ένα query σε μια Public Data Source και πιο συγκεκριμένα στη DBpedia σχετικά με τα μέσα μεταφοράς. Η DBpedia χρησιμοποιεί ένα πρόγραμμα που ονομάζεται SNORQL που δέχεται SPARQL ερωτήματα και επιστρέφει απαντήσεις.

**Ερώτηση:** Θέλουμε να βρούμε τα ονόματα των αεροπορικών που έχουν βάση τον Διεθνής Αερολιμένα Αθηνών.

**Σχόλια:** Αρχικά ψάχνουμε το link στην wikipedia για το αεροδρόμιο

[https://en.wikipedia.org/wiki/Athens\\_International\\_Airport](https://en.wikipedia.org/wiki/Athens_International_Airport)

επομένως αυτό θα υπάρχει στην dbpedia ως:

[http://dbpedia.org/resource/Athens\\_International\\_Airport](http://dbpedia.org/resource/Athens_International_Airport)

πηγαίνοντας να το επισκεφτούμε από τον browser βλέπουμε ότι γίνεται αμέσως redirect στο

[http://dbpedia.org/page/Athens\\_International\\_Airport](http://dbpedia.org/page/Athens_International_Airport) που είναι η html έκδοση αυτού του λήματος.

Επομένως χρησιμοποιούντας την οντολογία της dbpedia <http://dbpedia.org/ontology/> που έχει ως οντότητα το Airline και target Airport, ψάχνουμε πρώτα όλα τα subjects που είναι Airlines και από αυτά επιλέγουμε όσα έχουν target Airport το Διεθνής Αερολιμένα Αθηνών, δηλαδή το [http://dbpedia.org/resource/Athens\\_International\\_Airport](http://dbpedia.org/resource/Athens_International_Airport) που με βάση το default PREFIX της σελίδας το γραφουμε ως :Athens\_International\_Airport.

## SPARQL και Αποτελέσματα:

```
SPARQL:  
PREFIX owl: <http://www.w3.org/2002/07/owl#>  
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
PREFIX : <http://dbpedia.org/resource/>  
PREFIX dbpedia2: <http://dbpedia.org/property/>  
PREFIX dbpedia: <http://dbpedia.org/>  
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>  
PREFIX d: <http://dbpedia.org/ontology/>
```

```
SELECT *  
WHERE {  
?s a d:Airline.  
?s d:targetAirport :Athens_International_Airport.  
}
```

Results: [Browse](#) [Go!](#) [Reset](#)

### SPARQL results:

s
<a href="#">:Viking_Airlines</a>
<a href="#">:Fly_Hellas</a>
<a href="#">:SkyGreece_Airlines</a>
<a href="#">:Ellinair</a>
<a href="#">:Astra_Airlines</a>
<a href="#">:Hellas_Jet</a>
<a href="#">:Hellenic_Imperial_Airways</a>

**Σχολιασμός:** Εντυπωσιακό ότι δεν βρίσκουμε την Aegean (:Aegean\_Airlines), θα την βρούμε όμως κάνοντας την ίδια ερώτηση για το αεροδρόμιο Ηρακλείου. Και όμως η Aegean δεν έχει βάση στην Αθήνα.

## **Προβλήματα που αντιμετωπίστηκαν και προτάσεις διόρθωσης:**

### **Σχετικά με τα Δεδομένα**

Γενικότερα όσον αφορά το dataset παρατηρήθηκαν ορισμένα προβλήματα, όπου ο δημιουργός του δεν διατηρεί σταθερό το μοτίβο γραφής σε ορισμένα τμήματα. Παραδείγματος χάριν σε ορισμένα σημεία όταν περιέκλει strings από τα κατάλληλα συμβολα “”, σε κάποιες καταγραφές το διατηρούσε και σε άλλες δεν το διατηρούσε, όπως φαίνεται και στο routes\_long\_name όπου σε ορισμένες καταγραφές ξεχνούσε να τις πειρκλίσει με αυτά τα συμβολα.

Επίσης, μας έκανε εντύπωση το γεγονός ότι στην καταγραφή των ωρών μετά της 12 το βράδυ συνεχίζει μετά την ώρα 24:00 και δεν μηδενίζει, πχ η ώρα μια το βράδυ θα γραφόταν ως 25:00. Επίσης σαν λεπτομέρεια, δεν διατηρεί ομοιομορφο τον τρόπο συγγραφής των calendar\_id, όπου ορισμένες φορές μιλάει για calendar\_id αναφερόμενος σε μια συγκεκριμένη μέρα της εβδομάδας ενώ άλλες φορές σε όλη την εβδομάδα. Βέβαια αυτό δεν δημιουργεί κάποιο πρόβλημα.

### **Σχετικά με το Protege**

Θα ήταν πιο βολικό να υπάρχει μια συστηματική προγραμματιστική διαδικασία για την εισαγωγή των ζητούμενων στο Protege. Επίσης θα ήταν βολικό να παράγει και μια σχηματική αναπαράσταση της οντολογίας αυτόματα, έτσι ώστε να γίνεται πιο εύκολα κατανοητή και να επιβληθεί η διαδικασία του debugging.

### **Σχετικά με το OpenLink Virtuoso**

Η διαδικασία εμφάνισης των σφαλμάτων ήταν σειριακή πράγμα που δημιουργούν μεγάλα προβλήματα. Αυτο συμβαίνει καθώς υποβάλλοντας το ένα αρχείο με τις τριάδες χρειάζεται αρκετή ώρα για να το φορτώσει και ταυτόχρονα να ελέγχει τα λάθη και η διαδικασία σταμάταγε στο πρώτο λάθος, επομένως για να δεις το επόμενο θα έπρεπε να περιμένεις να ξεφορτωθεί το αρχείο (αφότου διόρθωσε το πρώτο λάθος) και αυτη η διαδικασία ήταν χρονοβόρα.

Επίσης ο σχολιασμός τυχόν λαθών στην SPARQL δεν ήταν τόσο κατανοητος και πάλι έβγαλε μήνυμα λάθους με το πρώτο σφάλμα που έβρισκε.

## **Επίλογος**

Το σημασιολογικό Διαδίκτυο χωρίς να έχει αναπτυχθεί ακόμη πλήρως δίνει σεβαστά και αξιόλογα αποτελέσματα κάνοντας την πληροφορία πιο κατανοητή από της μηχανές. Σε αυτό το project η διαδικασία αφορούσε μόνο την κατασκευή της σημασιολογικής βάσης δεδομένων, ωστόσο το πιο ενδιαφέρον κομμάτι είναι η αλληλεπίδραση της βάσης μας με δεδομένα από όλο το web, ή άντληση πληροφοριών από την βάση μας από άλλους χρήστες κλπ. Τα δεδομένα αποκτούν μια πιο ουσιαστική μορφή και όχι μια απλή γραφή για αναπαράσταση στο ανθρώπινο μάτι και ο προγραμματιστής μπορεί να κάνει ευκολότερα πολλές διαδικασίες και online συγκρίσεις και αναζητήσεις.

Επίσης η μοναδικότητα των urī εγγυάται από όλο το web αυτό σημαίνει ότι ο εμπλουτισμός της βάσης μας από άλλα δεδομένα στο web θα είναι πολύ πιο εύκολος αφού θα έχουν εγκιεμένο διαφορετικό ID, αφού δεν γίνεται να υπάρχει ίδιο uri.

Τέλος η χρήση της σημασιολογίας πάνω στις γεωγραφικές συντεταγμένες ήταν κάτι πολύ χρήσιμο και πολύ απλό σε σύγκριση με άλλους τρόπους αντιμετώπισης τέτοιων θεμάτων (πχ μέσω javascript)

## **Βιβλιογραφία**

1. Αναπαράσταση οντολογικής γνώσης και συλλογιστική, Γ. Στάμου, διαθέσιμο στο Αποθετήριο «Κάλλιπος» [<http://repository.kallipos.gr/handle/11419/4225>]
2. Εισαγωγή στο Σημασιολογικό Ιστό, G. Antoniou, F. van Harmelen, Κλειδάριθμος, 2008
3. Learning SPARQL, Bob DuCharme
4. OpenLink Virtuoso Universal Server Documentation
5. <https://protege.stanford.edu/>
6. Τεχνητή Νοημοσύνη - Μια Σύγχρονη Προσέγγιση, Russell and Norvig, Κλειδάριθμος, 2004
7. Knowledge Representation and Reasoning, Ronald Brachman, Hector Levesque, Elsevier, 2004