

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Συστήματα και Τεχνολογίες Γνώσης**

### **Εξαμηνιαίο Θέμα**

*Έλενα Βεργοπούλου 03116723*

*8ο εξάμηνο*

Πρώτο Μέρος: Κατασκευή οντολογίας

*Στο θέμα αυτό καλούμαστε να κατασκευάσουμε μια σημασιολογική βάση γνώσης που θα περιέχει δεδομένα που σχετίζονται με μετακινήσεις με μέσα μεταφοράς. Θα κατασκευάσουμε μια οντολογία για τα μέσα μεταφοράς η οποία μοντελοποιεί επαρκώς το συγκεκριμένο πεδίο, θα συγκεντρώσουμε κατάλληλα δεδομένα, και θα τα αναπαραστήσουμε ως στιγμιότυπα των εννοιών, ρόλων και ιδιοτήτων τύπων δεδομένων της οντολογίας ώστε να φτιάξουμε μια ολοκληρωμένη βάση γνώσης.*

*Αρχικά παίρνουμε ως βάση κάποιες κλάσεις από το πρότυπο GTFS και τις εμπλουτίζουμε με επιπλέον data properties, ακόμη και αν δεν έχουμε κάποια στοιχεία για αυτά από το dataset μας. Έτσι δημιουργούμε μία πιο πλήρη οντολογία όσον αφορά τις μετακινήσεις με ΜΜΜ σε θεωρητικό επίπεδο για να περιγράψουμε καλύτερα και λεπτομερώς τον κόσμο. Στα επόμενα μέρη θα στηριχθούμε προφανώς στα δεδομένα μας και τις πληροφορίες που μπορούμε να εξάγουμε από αυτά.*

* Agency

Data properties:

* **agency\_id**: range: xsd:integer, αναγνωριστικό του πρακτορείου
* **agency\_name**: range: xsd:string, όνομα πρακτορείου
* **agency\_url**: range: xsd:anyURI, URL της ιστοσελίδας του ταξιδιωτικού πρακτορείου
* **agency\_timezone**: range: [time:TimeZone](https://www.w3.org/TR/owl-time/#time:TimeZone), ζώνη ώρας της τοποθεσίας του πρακτορείου
* **agency\_lang:** range: xsd:string, κύρια γλώσσα που χρησιμοποιείται από το πρακτορείο
* **agency\_phone**: range: xsd:language, τηλέφωνο επικοινωνίας πρακτορείου
* **agency\_fare\_url**: range: xsd:anyURI, URL του ταξιδιωτικού πρακτορείου για την αγορά εισιτηρίων
* **agency\_email:** range: xsd:string, email πρακτορείου
* Vehicle

Data properties:

* **vehicle\_id**: range: xsd:string, το ID αναγνωριστικό κάθε οχήματος πχ αν έχουμε λεωφορείο θα είναι ο αριθμός κυκλοφορίας του, αν είναι πλοίο θα έχουν το κωδικό τους κτλ
* **vehicle\_type:**  range: xsd:string, τύπος μεταφορικού μέσου
* **agency\_id**: range: xsd:integer, αναγνωριστικό του πρακτορείου που ανήκει το όχημα
* **construction\_date**: range: xsd:date, ημερομηνία κατασκευής
* **service\_date:** range: xsd:date, ημερομηνία που έγινε service
* **make**: range: xsd:string, μάρκα οχήματος
* **model**: range: xsd:string, μοντέλο οχήματος
* **colour**: range: xsd:string, χρώμα οχήματος
* Stops

Data properties

* **stop\_id:** range: xsd:integer, το ID αναγνωριστικό μίας στάσης
* **stop\_code:** range: xsd:integer, ένας αναγνωριστικός κωδικός της τοποθεσίας της στάσης που αφορά τους οδηγούς των μέσων
* **stop\_name**: range: xsd:string, το όνομα της στάσης
* **stop\_desc:** range: xsd:string, περαιτέρω περιγραφή της τοποθεσίας της στάσης
* **stop\_point**: range: rdf:point, που περιγράφει τις γεωγραφικές συντεταγμένες της στάσης δηλαδή περιλαμβάνει το Latitude και το Longitude της στάσης (για την σημασιολογική αναπαράσταση θα κάνουμε χρήση του LinkedGeoData)
* **nodes**: range: rdf:boolean, 0 αν δεν είναι κόμβος και 1 αν είναι, δηλαδή αν είναι στάση και λεωφορείου αλλά υπάρχει και μετρό κοντά
* **wheelchair\_boarding:** range: rdf:boolean, 0 αν δεν είναι προσβάσιμο με αναπηρικό αμαξίδιο και 1 αν είναι
* **zone\_id:** range: xsd:integer, το ID αναγνωριστικό της ζώνης εισιτηρίων για αυτή τη στάση
* Stop\_times

Data properties:

* **trip\_id:** range: xsd:integer, αναγνωριστικό της διαδρομής
* **arrival\_time**: range: xsd:dateTime, ώρα αφιξης σε μια συγκεκριμένη στάση
* **departure\_time**: range: xsd:dateTime, ώρα αναχώρησης από μια συγκεκριμένη στάση
* **stop\_sequence:** range: xsd:nonNegativeInteger, που περιγράφει την σειρά της στάσης σε μία συγκεκριμένη διαδρομή
* **stop\_id:** range: xsd:integer, το ID αναγνωριστικό μίας στάσης
* **stop\_headsign**: range: xsd:string, κείμενο σε επιγραφή που θα απεικονίζει τον προορισμό
* **pickup\_type:** range: xsd:integer, τρόπος που προγραμματίζεται η επιβίβαση δηλαδή 0 για συχνή επιβίβαση, 1 για μη διαθέσιμη επιβίβαση, 2 επιβίβαση μετά από επικοινωνία με agency, 3 επιβίβαση μετά από επικοινωνία με τον οδηγό
* **dropoff\_type:** range: xsd:integer, αντίχοιχα με pickup type απλά για αποβίβαση
* **timepoint:** range: xsd:boolean, 0 αν οι χρόνοι στάσεων είναι προσεγγιστικοί και 1 αν είναι ακριβείς
* **stops\_number**:range: xsd:integer, συνολικός αριθμός των στάσεων
* Routes

Data properties:

* **route\_id:** range: xsd:integer, το ID αναγνωριστικό ενός δρομολογίου
* **route\_short\_name**: range: xsd:string, σύντομο όνομα του δρομολογίου
* **route\_long\_name**: range: xsd:string, αναλυτικό όνομα του δρομολογίου
* **route\_desc:** range: xsd:string, περαιτέρω περιγραφή του δρομολογίου
* **route\_type**: range: xsd:integer, τύπος δρομολογίου
* **route\_url**: range: xsd:anyURI, URL η ιστοσελίδα για το συγκεκριμένο δρομολόγιο
* **route\_color:** range: xsd:string, το χρώμα του δρομολογίου
* **route\_text\_ color:** range: xsd:string, το χρώμα γραμματοσειράς του δρομολογίου
* **underground\_route**: range: rdf:boolean, 0 αν δεν είναι υπόγεια διαδρομή και 1 αν είναι
* Trips

Data properties:

* **route\_id:** range: xsd:integer, το ID αναγνωριστικό ενός δρομολογίου
* **service\_id:** range: rfd:setofdates, πότε είναι διαθέσιμο για εξυπηρέτηση
* **trip\_id:** range: xsd:integer, αναγνωριστικό της διαδρομής
* **trip\_headsign**: range: xsd:string, τίτλος του δρομολογίου όπως αναγράφεται στις επιγραφές
* **direction\_id**: range: xsd:boolean, 0 για την μία κατεύθυνση και 1 για την αντίθετη
* **block\_id:** range: xsd:integer, αναγνωριστικό για το μπλοκ στο οποίο ανήκει η διαδρομή
* **shape\_id:** range: xsd:integer, αναγνωριστικό για το γεωχωρικό σχήμα που περιγράφει το μονοπάτι της διαδρομής
* **wheelchair\_accessible**: range: xsd:boolean, 0 αν δεν είναι προσβάσιμο με αναπηρικό αμαξίδιο και 1 αν είναι
* **bikes\_allowed**: range: xsd:boolean, 0 αν δεν επιτρέπονται ποδήλατα και 1 αν επιτρέπονται
* **pets\_allowed:** range: xsd:boolean, 0 αν δεν επιτρέπονται κατοικίδια και 1 αν επιτρέπονται
* **after\_midnight\_trip**: range: xsd:boolean, 0 αν δεν είναι μεταμεσονύχτιο και 1 αν είναι
* Calendar

Data properties:

* **service\_id:** range: xsd:date, πότε είναι διαθέσιμο για εξυπηρέτηση
* **monday:** range: xsd:boolean, 1 αν αν λειτουργεί όλες τις Δευτέρες εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **tuesday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τις Τρίτες εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **wednesday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τις Τετάρτες εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **thursday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τις Πέμπτες εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **friday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τις Παρασκευές εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **saturday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τα Σάββατα εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **sunday:** range:xsd:boolean, 1 αν αν λειτουργεί όλες τις Κυριακές εντός του διαστήματος [start\_date,end\_date] και 0 αν όχι
* **start\_date**: range: xsd:date, ημερομηνία έναρξης των εξυπηρετήσεων
* **end\_date**: range: xsd:date, ημερομηνία λήξης των εξυπηρετήσεων

### Feed\_info

Data properties:

* **feed\_id:** range: xsd:integer, αναγνωριστικό της πηγής που αντλήσαμε τα δεδομένα μας
* **feed\_publisher\_name**: range: xsd:string, όνομα οργανισμού που δημοσίευσε το dataset
* **feed\_publisher\_url:** range: xsd:anyURI, URL της ιστοσελίδας του οργανισμού που αντλήσαμε τα δεδομένα
* **feed\_lang**: range: xsd:language, default γλώσσα των δεδομένων
* **feed\_version:** range: xsd:string, δείχνει την τωρινή εκδοχή του dataset μας
* Fare\_Attributes

Data properties:

* **fare\_id**: range xsd:integer, αναγνωριστικό κομίστρου
* **price:** range xsd:float, τιμή εισιτηρίου
* **currency\_type:** range: xsd:string, τι νόμισμα γίνεται η πληρωμή
* **payment\_time**: range: xsd:boolean, μέθοδος πληρωμής 0 κατά την επιβίβαση και 1 πριν την επιβίβαση
* **transfers**: range xsd:range xsd:nonNegativeInteger, πόσες μετακινήσεις δικαιούται κανείς με το εισιτήριό του
* **transfer\_duration**: range xsd:duration, χρόνος για μετακινήσεις μέχρι τη ληξη του εισιτηρίου
* **payment\_method**: range xsd:integer, μέθοδοι πληρωμής πχ 0 για μετρητά, 1 με πιστωτική, 2 για cryptocurrency κτλ
* Fare\_rules:

Data properties:

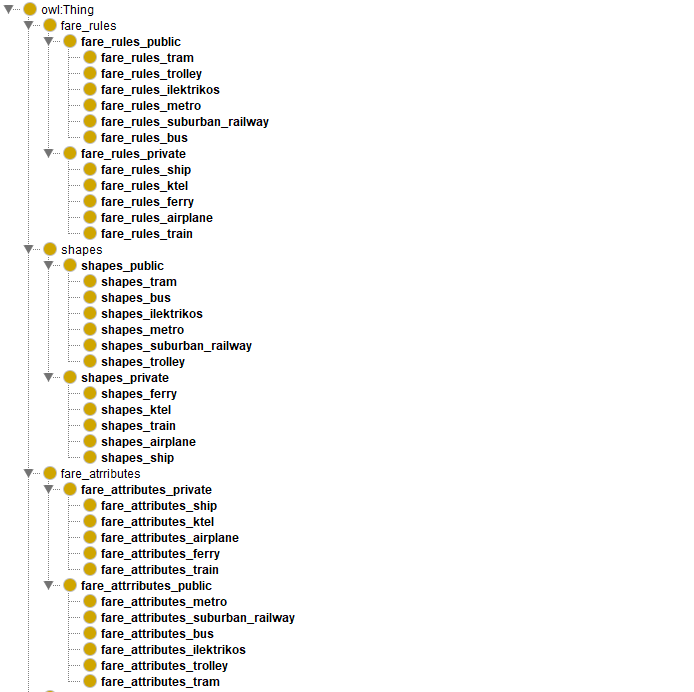
* **fare\_id**: range xsd:integer, αναγνωριστικό κομίστρου
* **route\_id:** range: xsd:integer, το ID αναγνωριστικό ενός δρομολογίου
* **origin\_id:** range: xsd:integer, το ID αναγνωριστικό της ζώνης προέλευσης
* **destination\_id:** range: xsd:integer, το ID αναγνωριστικό της ζώνης προορισμού
* **contains\_id**: range: xsd:integer, το ID αναγνωριστικό των ζωνών που θα εισέλθει κάποιος με το εισιτήριο που έχει αγοράσει
* Shapes:

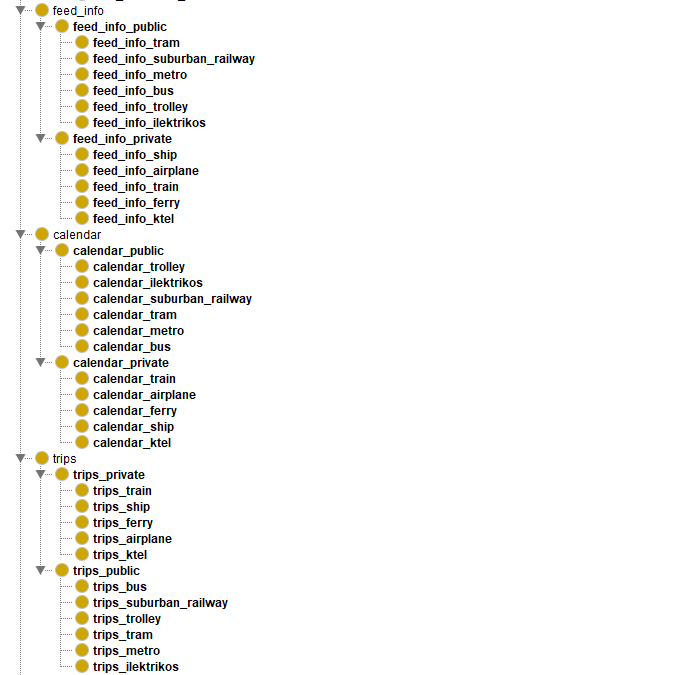
Data properties:

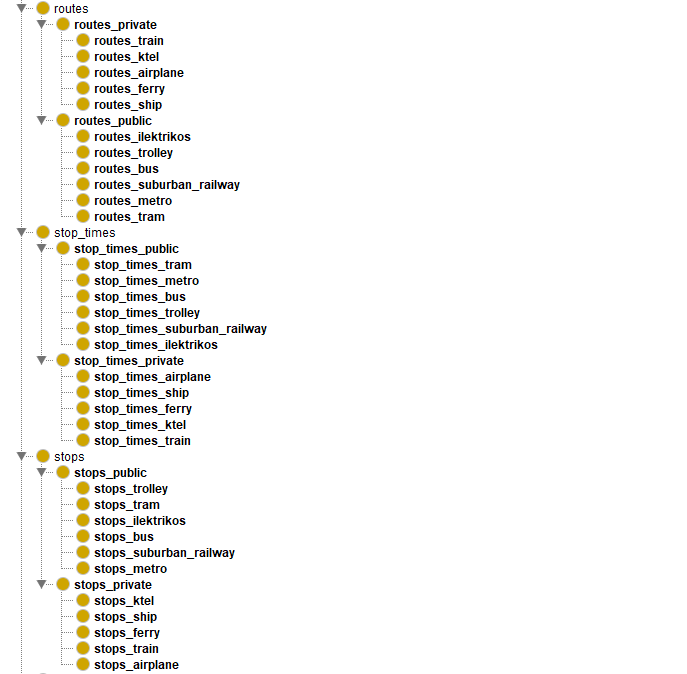
* **shape\_id**: range: xsd:integer, το ID αναγνωριστικό των σημείων του “σχήματος”, δηλαδή της περιγραφής του μονοπατιού που διανύει ένα όχημα κατα μήκος μίας διαδρομής
* **shape\_pt\_point**: range: rdf:point, το γεωγραφικό μήκος και πλάτος ενός σημείου του σχήματος
* **shape\_pt\_sequence:** range: rdf:nonNegativeInteger, σειρά με την οποία συνδέεονται τα σημεία για να δημιουργήσουν το τελικό σχήμα

Η αναπαράσταση της οντολογίας μας θα γίνει στο protege. Κάθε κλάση θα χωριστεί σε δύο υποκλάσεις ανάλογα με το αν το μέσο μεταφοράς είναι δημόσιο ή ιδιωτικής εταιρείας. Έτσι θα έχουμε τα public και private means αντίστοιχα. Στην κατηγορία public εντάσσονται τα λεωφορεία, τρόλλευ, τραμ, μετρό, ηλεκτρικός και προαστιακός, ενώ στα private πλοία, ferries, αεροπλάνα, τρένα, κτελ. Αυτή η διάκριση παράγει μεταγνώση για καλύτερη σημασιολογική κατανόηση και αντιμετώπιση των ταξιδιών από την εφαρμογή. Μετά κάθε μια από αυτές τις δύο κλάσεις χωρίζεται σε υποκλάσεις που περιγράφουν αναλυτικά σε ποιό μέσο μεταφοράς αναφέρονται.

Συγκεκριμένα:

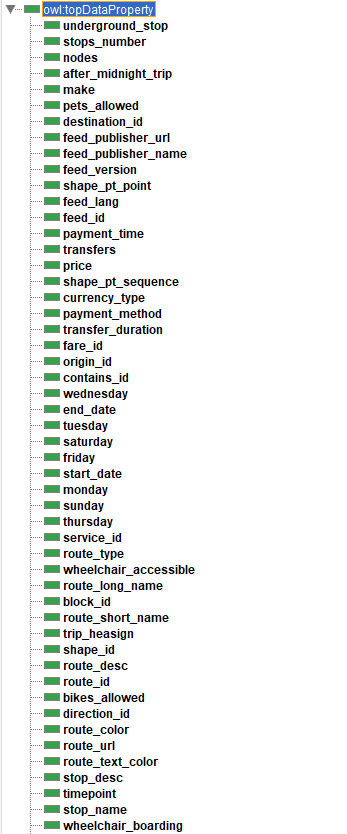


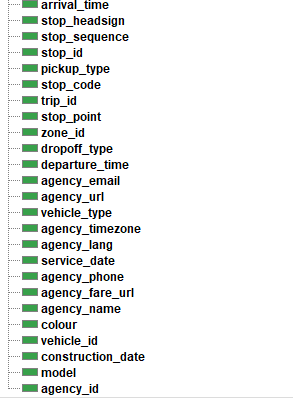






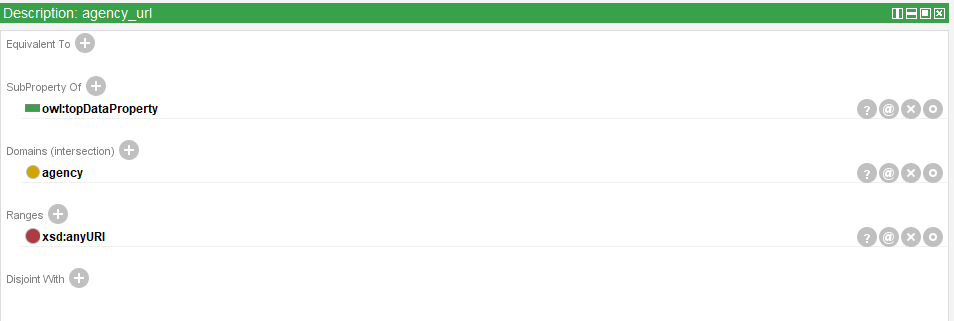
Στη συνέχεια θα προσθέσουμε τα data properties των κλάσεών μας:





Σε κάθε data property ορίσαμε κατάλληλα το domain και το range

Π.χ. για το agency\_url:



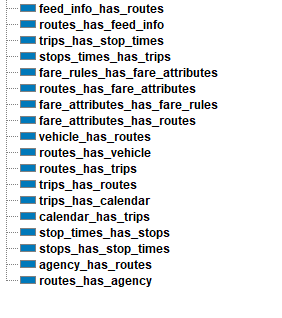
Τώρα θα μελετήσουμε τα object properties που συνδεουν τις κλάσεις μεταξύ τους. Εχοντας ως δεδομένα τις ακμές προς μια κατεύθυνση μπορούμε να δημιουργήσουμε μέσω της οντολογίας και τις ακμές αντίθετης κατεύθυνσης αν ορίσουμε αυτες τις δύο σχέσεις ως inverse.

Έτσι εμείς θα δώσουμε μόνο της σχέσεις προς μια κατεύθυνση και η οντολογία θα παράξει ως μεταγνώση τις σχέσεις της αντιστροφης. Η διαδικασία αυτή γίνεται για λόγους πληρότητας της βάσης μας.

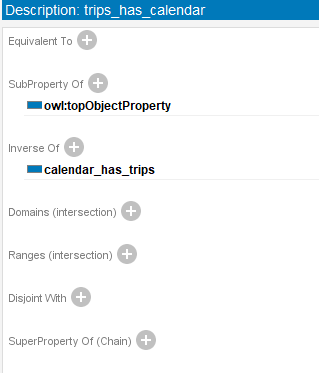
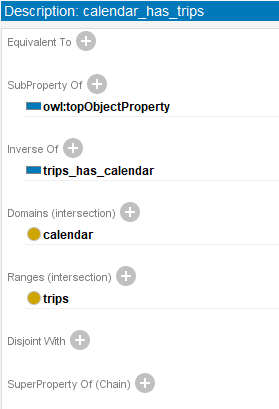
Έτσι έχουμε τις εξής σχέσεις (ορίζουμε το ελάχιστο δυνατό πλήθος σχέσεων για να συνδέσουμε μεταξύ τους τις κλάσεις μας)

* routes\_has\_agency
* stops\_has\_stop\_times
* calendar\_has\_trips
* routes\_has\_trips
* routes\_has\_vehicle
* fare\_attributes\_has\_routes
* fare\_attributes\_has\_fare\_rules
* stop\_times\_has\_trips
* routes\_has\_feed\_info

Στη συνέχεια ορίζουμε κ τις αντίστροφές τους κ έτσι έχουμε:



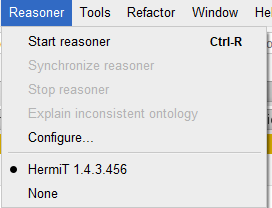
Σε κάθε σχέση από τις ευθείς ορίσαμε domain και range και μετά για τις αντίστροφες ορίσαμε ότι είναι οι αντίστροφες των ευθέων:



Το αρχείο μας από τo protege βρίσκεται στο αρχείο *[stg]protege.owl*

Το επόμενό μας βήμα είναι να δημιουργήσουμε από το protege, κώδικα σε owl.

Για αυτό ενεργοποίουμε το Hermi Τ. Και μετά start reasoner.



Το αντίστοιχο αρχείο είναι το *Myontology*

To συγκεκριμένο μέρος είχε αρκετή ελευθερία και λίγους περιορισμούς καθως μπορούσαμε να επεκτείνουμε την οντολογία μας στο βαθμό που θέλαμε εμείς και να προσθέσουμε όσα στοιχεία επιθυμούσαμε. Το βασικό πρόβλημα που αντιμετωπίσαμε ήταν ότι πολλές ενέργειες στο protege έπρεπε να πραγματοποιηθούν manually. Αυτό ήταν ένα βασικό θέμα όσον αφορά τα data properties που συμπεριλάβαμε αρκετά. Η εισαγωγή τους μπορούσε να γίνει μαζικά ωστόσο η διευκρίνιση των domain και range αποδείχθηκε αρκετά χρονοβόρα.

Δεύτερο Μέρος: Δημουργία RDF τριάδων

Θέλουμε να μετατρέψουμε τα csv αρχεία σε τριάδες RDF. Κάθε csv αρχείο αντιστοιχεί σε μια κλάση. Για να κάνουμε την μετατροπή πρέπει αν προσδιορίσουμε ποιο θα είναι το υποκείμενο, το αντικείμενο και το κατηγόρημα. Σε κάθε περίπτωση το υποκείμενο θα ορίζεται ως ένα uri με την βοήθεια του ID της κλάσης, που βρίσκεται εντός του ίδιου αρχείου με αυτή. Το κατηγόρημα θα είναι είτε το “a” είτε κάποιο URI.

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.

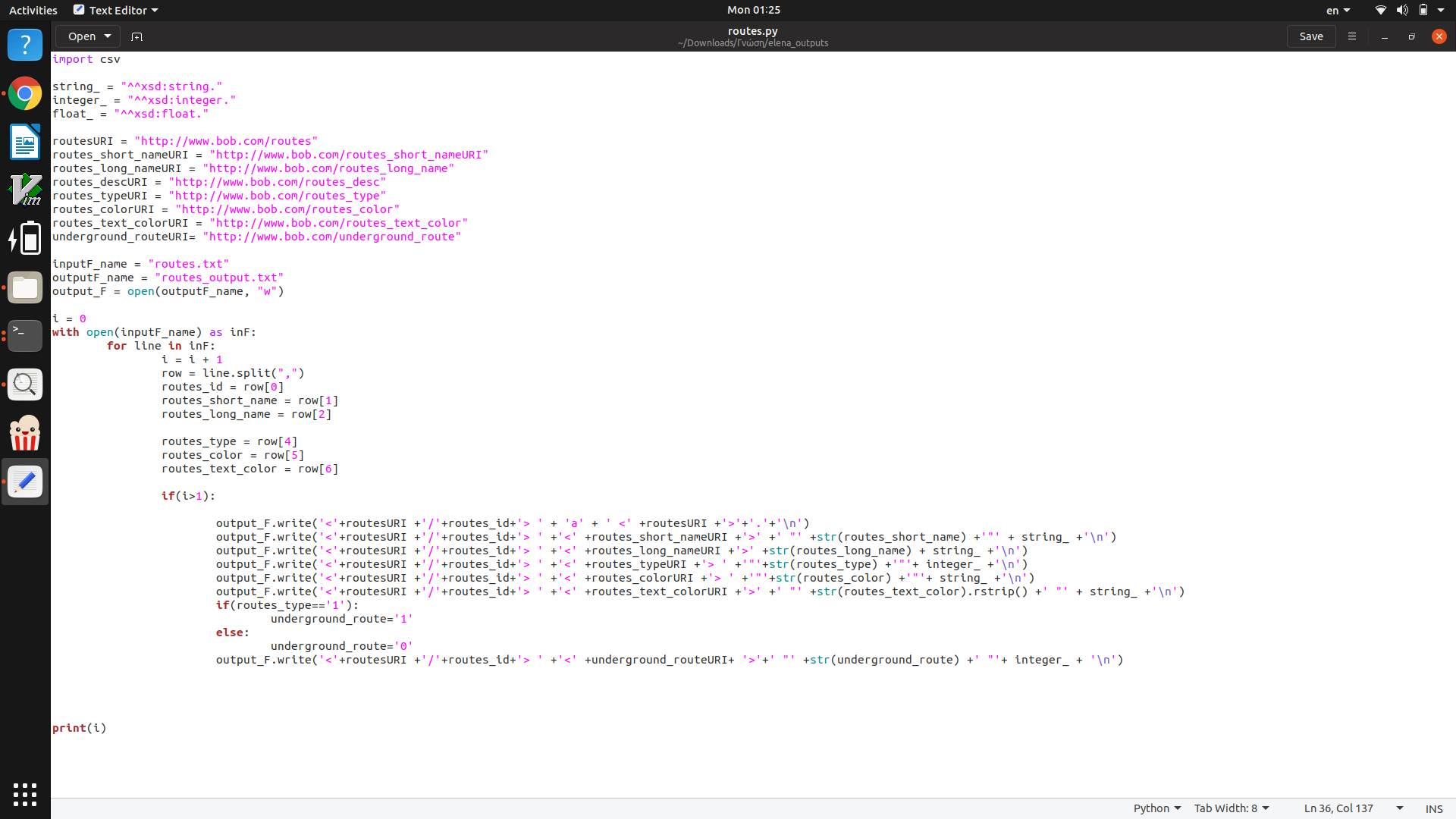
Όσον αφορά τα object properties, αρκεί να σχεδιάσουμε τις ευθείς σχέσεις και η αντίστροφες θα σχεδιαστούν αυτόματα.Γενικά θα χρειαστούμε πληροφορίες για το ID των καταγραφών των κλάσεων που θέλουμε να συσχετίσουμε. (Τα αποτελέσματα αυτά βρίσκονται στα RF αρχεία).

H σειροποίηση επιλέξαμε να γίνει με ​**N-triples.**

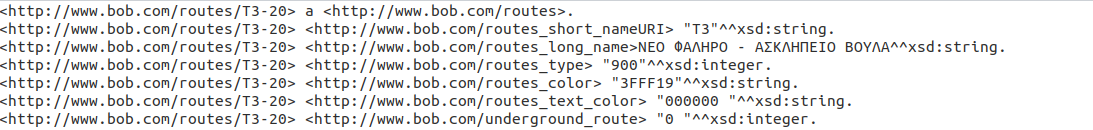
Τέλος, καθώς ένας από τους σκοπούς της άσκησης είναι να παράξουμε **μεταγνώση**, στο routes.py εκτμεταλλετήκαμε το γεγονός ότι όταν μία διαδρομή εκτελείται από μετρό (είναι route type 1) τότε οι σταθμοί του θα είναι υπόγειοι οπότε αυτόματα θα συμπληρωνουμε 1 στο πεδίο underground\_route. Αν η βάση μας ήταν πιο πλούσια θα εξαγάγαμε και θα προσθέταμε αντίστοιχες πληροφορίες και σε άλλες κλάσεις.

Παρουσιάζουμε παρακάτω τους κώδικες σε python με τους οποίους έγινε η μετατροπή και οι οποίοι βρίσκονται στον φάκελο “Κώδικες Python”. Κάθε φορά παρουσιάζουμε και ένα ενδεικτικό αποτέλεσμα. Τα πλήρη txt αρχεία των αποτελεσμάτων βρίσκονται στον φάκελο “outputs”. Από αυτά απουσιάζουν τα αποτελέσματα του stop\_times καθώς τα αρχεία ήταν πολύ μεγάλα (πάνω από 1 GB) ωστόσο έχουμε συμπεριλάβει screenshots.

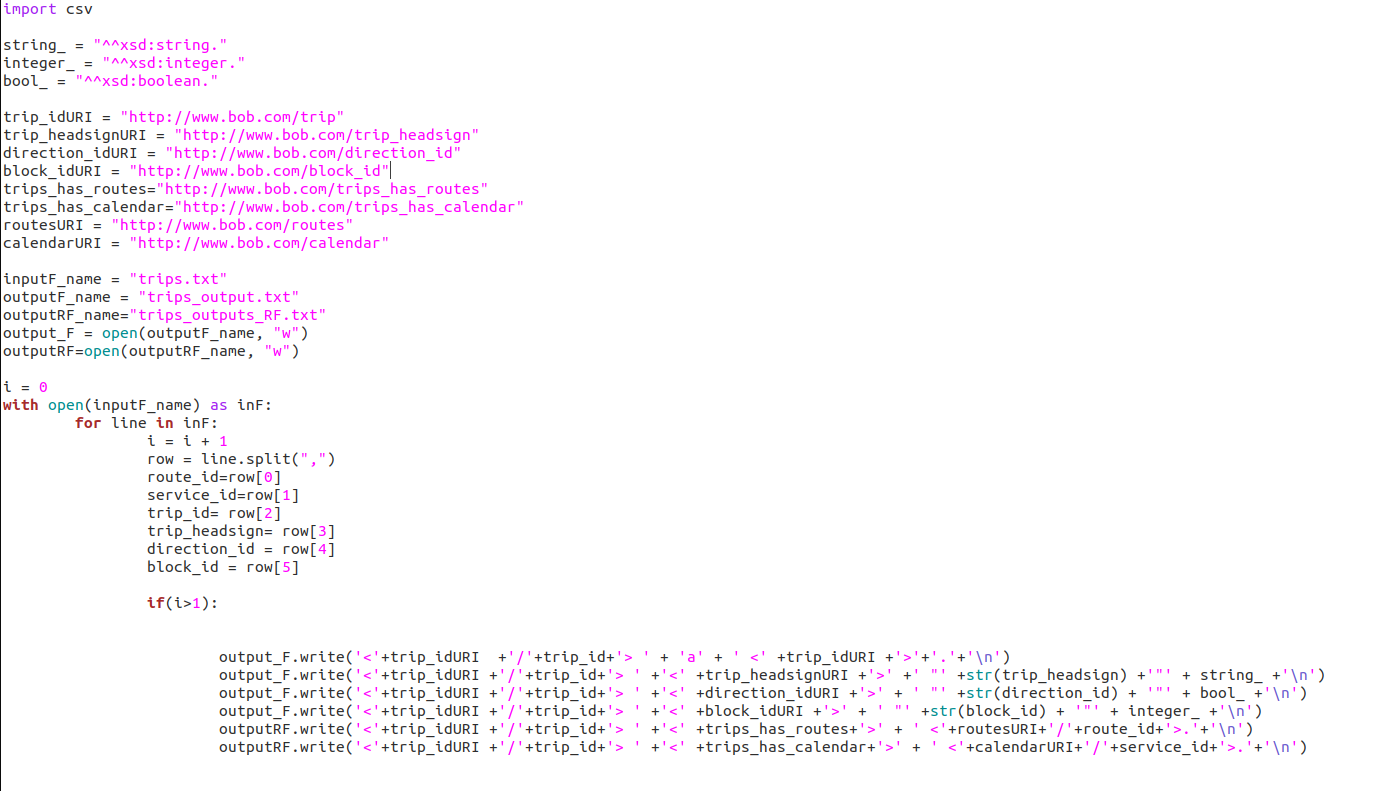
routes.py



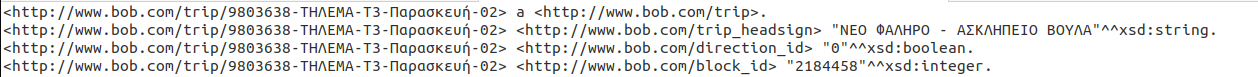
routes\_output.txt



trips.py



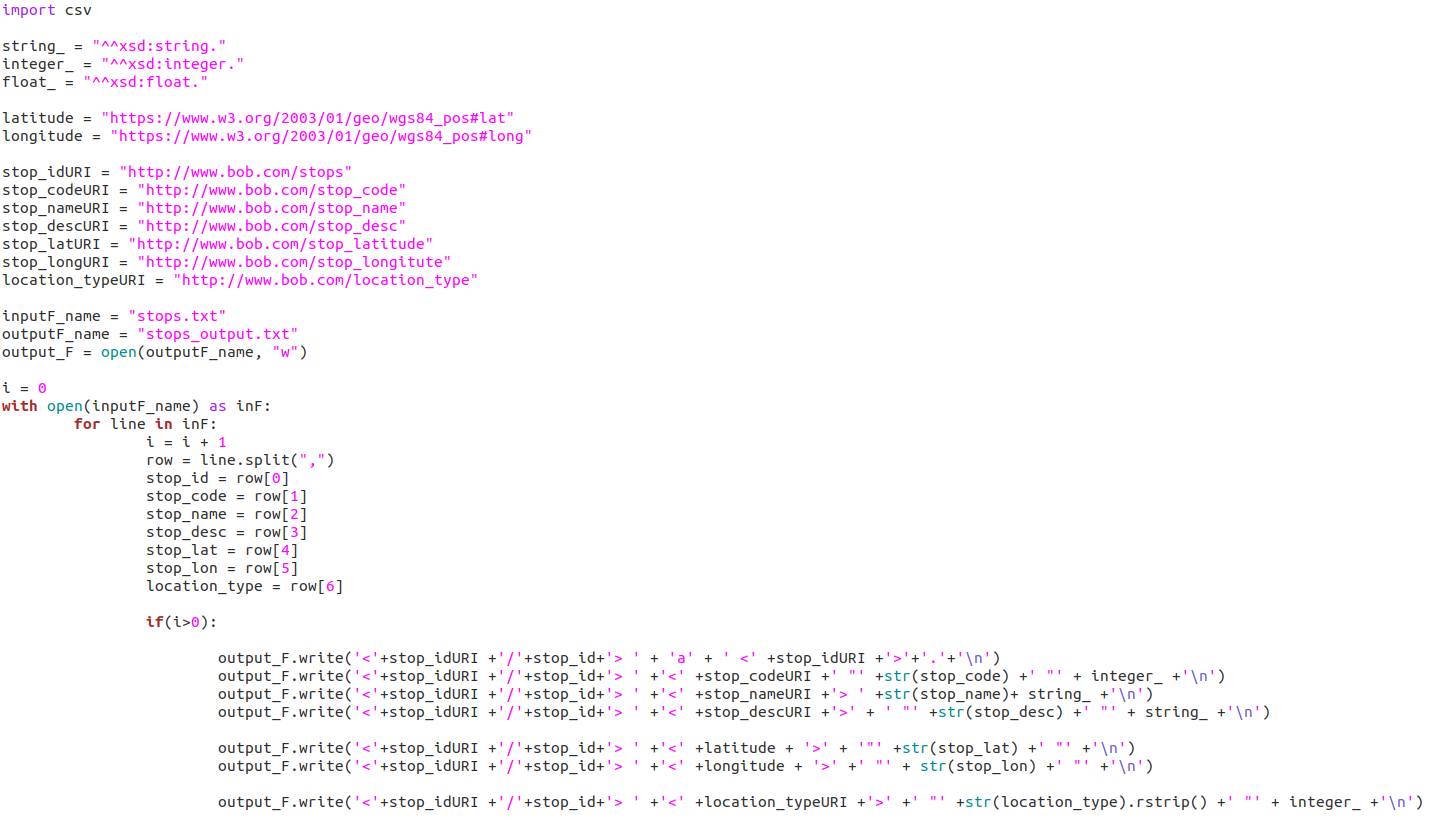
trips\_output.txt



trips\_outputs\_RF.txt



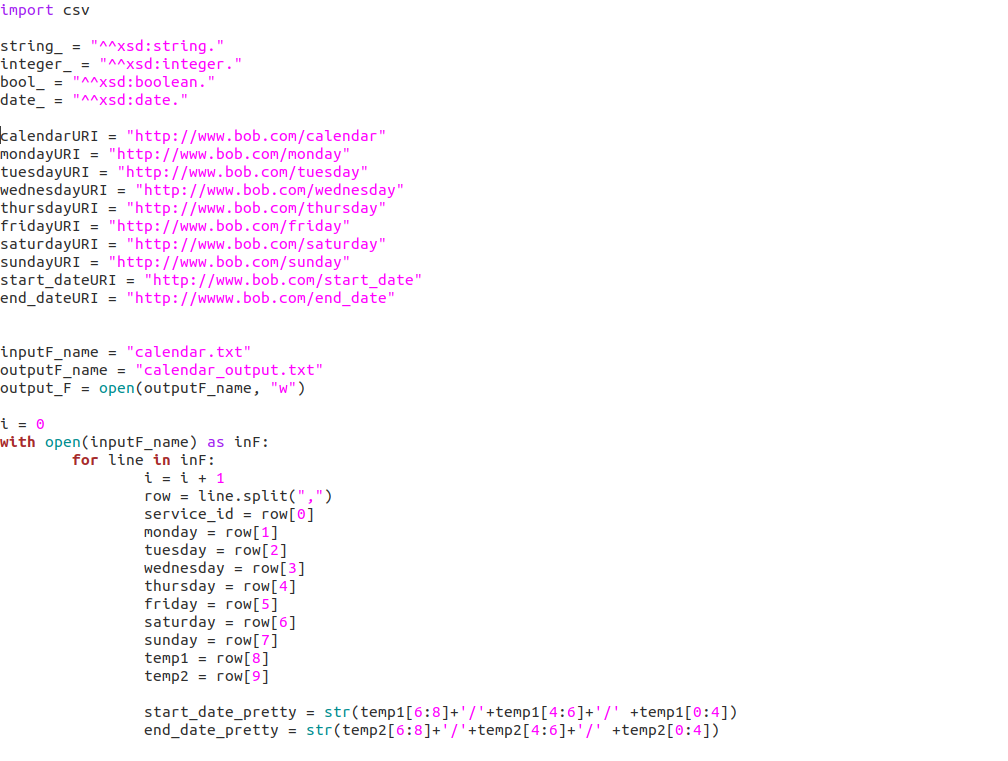
stops.py

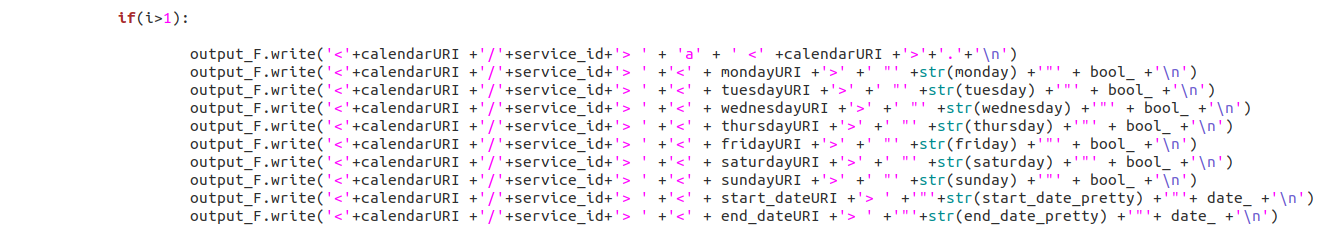


stops\_output.txt



calendar.py

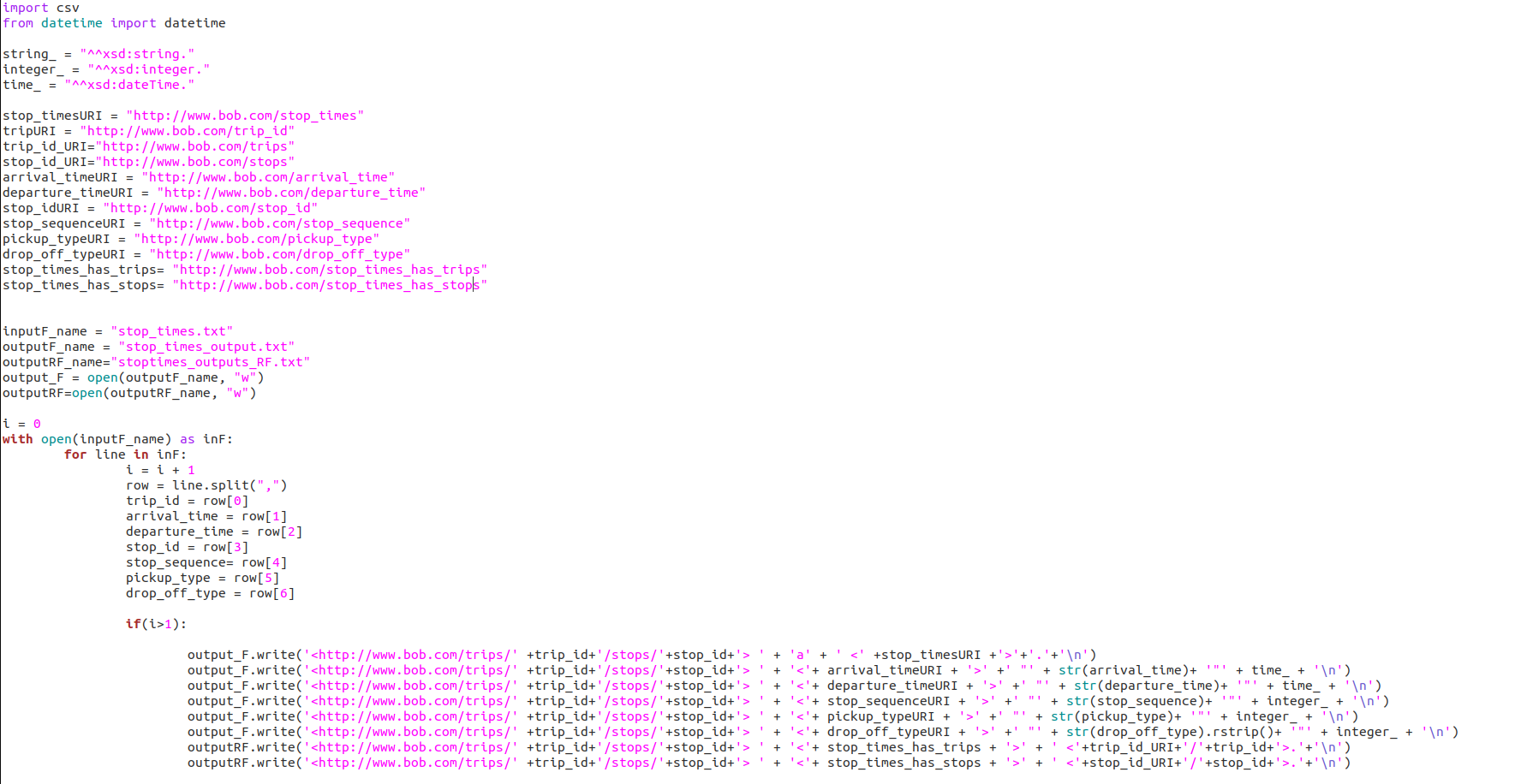




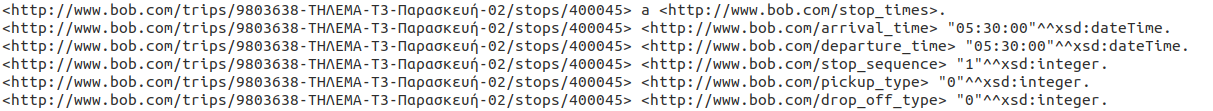
calendar\_output.txt



stoptimes.py



stop\_times\_output.txt



stoptimes\_outputs\_RF.txt



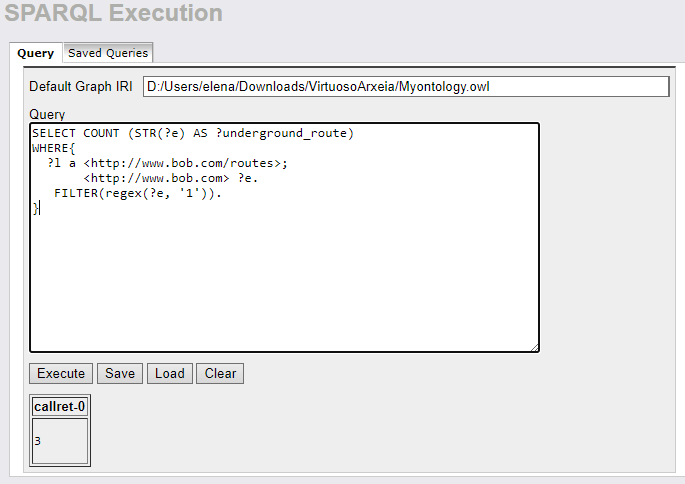
Το βασικό πρόβλημα σε αυτό το ερώτημα ήταν η μορφή των δεδομένων των οποίων η γραφή δεν τηρούσε κάποιο συγκεκριμένο πρωτόκολλο. Έτσι πχ υπήρχαν περιγραφές διαδρομών χωρίς “” όπως ΒΟΥΛΑ ενώ άλλες με “” όπως “ΣΥΝΤΑΓΜΑ”. Αυτό μας δυσκόλεψε αρκετά στο επόμενο στάδιο και πολλές διορθώσεις έγιναν manually. Συνεπώς υπάρχει ενδεχόμενο κάποιο από τα παραδοτέα αρχεία να μην είναι το πλήρως ανανεωμένο για αυτόν τον λόγο.

Μέρος Γ: Ερωτήματα SPARQL

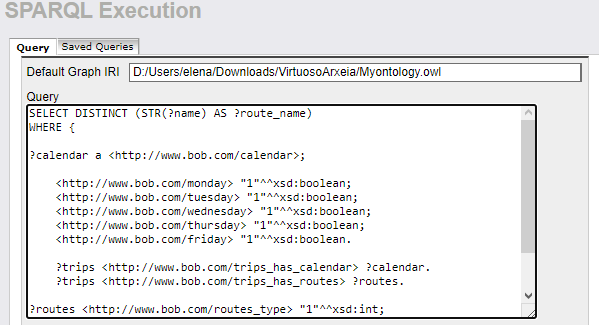
Για αυτό το ερώτημα χρησιμοποιήσαμε το Virtuoso μέσω του οποίου ουσιαστικά ενώσαμε τα προηγούμενα αποτελέσματά μας, τον κώδικα σε owl και τις RDF τριάδες. Έτσι παράγουμε τον γράφο γνώσης μας.

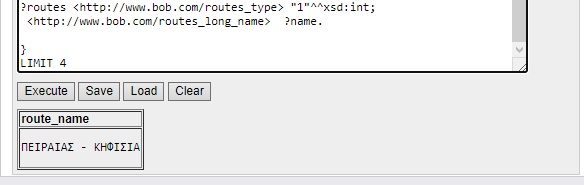
Τώρα είμαστε έτοιμοι να εκτελέσουμε κάποια queries σε SPARQL και παρουσιάζουμε τα αποτελέσματά τους:

* Εκμεταλλευόμενοι την μεταγνώση που παράξαμε θα μετρήσουμε το πλήθος των διαδρομών που είναι υπόγειες:



* Τα δρομολόγια που εκτελούνται κάθε μέρα από μετρό





Ένα πρόβλημα που αντιμετωπίσαμε ήταν αρχικά η εγκατάσταση του virtuoso η οποία προϋπέθετε να ακολουθήσουμε μία πολύ συγκεκριμένη διαδικασία για να μπορέσουμε να χρησιμοποιήσουμε επιτυχώς το πρόγραμμα. Στη συνέχεια κατά την εκτέλεση των SPARQL ερωτημάτων, ήταν αρκετά χρονοβόρο το debugging. Ορισμένα λάθη δεν τα εμφάνιζε καν ή τα παρουσίαζε με ασαφή τρόπο. Επίσης όσα εμφανίζονταν ήταν σειριακά με αποτέλεσμα να προκαλούνται αρκετές καθυστερήσεις. Ωστόσο γενικά όσον αφορά τα queries η διαδικασία του να γραφτούν ήταν αρκετά εύκολη δεδομένης και της δικής μας πρότερης εξοικείωσης με SQL.