7ο Εξάμηνο, 2024 - 2025

ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ ΚΑΙ ΕΥΦΥΕΙΣ ΕΦΑΡΜΟΓΕΣ

Εργασία Β: RDF4J / GraphDB

Ναλμπάντη Παναγιώτα, 4050
Παντελοπούλου Βασιλική, 4153

Περιεχόμενα

Γενικά

Στην παρούσα εργασία κληθήκαμε να πραγματοποιήσουμε τα εξής:

1. Επέκταση του μοντέλου RDFS που αναπτύχθηκε στην πρώτη εργασία, προγραμματιστικά με χρήση της γλώσσας προγραμματισμού Java και της RDF4J.
2. Φόρτωση του μοντέλου στην GraphDB για δημιουργία σημασιολογικής βάσης και εκτέλεσης SPARQL ερωτημάτων.
3. Εξαγωγή στατιστικών στοιχείων από το μοντέλο.

Το μοντέλο μας έχει σχέση με το διάστημα και την αστρονομία, και το αρχείο .ttl που το περιέχει υπάρχει στον φάκελο του GitHub repository ως «astronomy.ttl».

****Υποσημείωση**: *Στο GitHub repository έγιναν τα commits μόνο από ένα άτομο. Ο λόγος είναι διότι λόγω δύσκολων συνθηκών η εργασία έγινε από έναν υπολογιστή, όμως με ομαδική εκπόνηση*.

Επέκταση μοντέλου με RDF4J

Μετασχηματισμός από RDFS σε OWL

Για την μετατροπή του μοντέλου από RDFS σε OWL έγινε χρήση της RDF4J. Αρχικά, μετά την σύνδεση του κώδικα με το «ProjectB» repository στο GraphDB, ανοίγεται με ένα inputStream το αρχείο «astronomy.ttl» με σκοπό την ανάγνωσή του. Επιπλέον, δημιουργείται ένα καινούριο μοντέλο τύπου TreeModel, το owlModel, στο οποίο θα αποθηκευτεί το owl μοντέλο.

Για την διευκόλυνση της ανάγνωσης, ορίστηκαν χώροι ονομάτων για τα IRIs που χρησιμοποιούνται (owl, rdfs, rdf, astronomy, xsd). Στην συνέχεια, προστέθηκαν όλες οι τριπλέτες του μοντέλου μέσα στο owlModel. Οι μετασχηματισμοί που έπρεπε να γίνουν στο συγκεκριμένο μοντέλο ήταν οι εξής:

- RDFS.CLASS → OWL.CLASS
- RDF.PROPERTY → OWL.OBJECTPROPERTY ή OWL.DATATYPEPROPERTY

Έτσι, για την μετατροπή από RDF.CLASS σε OWL.CLASS, με την βοήθεια της getStatements() λάβαμε όλα τα statements που έχουν υποκείμενο τύπου rdf.class, τα αφαιρέσαμε από το owlModel και προσθέσαμε νέα με το ίδιο υποκείμενο και κατηγόρημα, και με αντικείμενο = owl.class.

Παράλληλα, για την μετατροπή της rdf ιδιότητας σε owl ιδιότητα αντικειμένου ή datatype, εφαρμόστηκε η ίδια διαδικασία. Για να εντοπίσουμε τον τύπο που πρέπει να πάρει μια ιδιότητα, με την βοήθεια της getStatements() λάβαμε τις τριπλέτες με κατηγόρημα = rdfs.range. Σε αυτές, έγινε έλεγχος, αν τα αντικείμενα τους ήταν τύπου Literal ή τύπου Object, και έτσι έγινε αλλαγή σε owl.datatypeProperty και owl.objectProperty αντίστοιχα.

Στην περίπτωση όπου κάποια ιδιότητα δεν έχει δηλωμένο range, η default ανάθεση τύπου είναι owl:datatypeProperty.

Μετά από την παραπάνω διαδικασία, το owlModel είναι μετασχηματισμένο σε owl. Για την διατήρησή του, δημιουργείται ένα αρχείο .ttl, «astronomyOWL.ttl», μέσα στο οποίο αποθηκεύεται το owlModel.

Ορισμός περιορισμών

Στο νέο μοντέλο έγινε προσθήκη έξι περιορισμών με την βοήθεια της RDF4J, οι οποίοι περιγράφονται παρακάτω. Για την δήλωση των περιορισμών έγινε χρήση του blank node.

owl:someValuesFrom

Ο περιορισμός someValuesFrom δηλώνει ότι μια ιδιότητα σχετίζεται τουλάχιστον με ένα αντικείμενο μιας συγκεκριμένης κλάσης. Οι δύο σχετικοί περιορισμοί που δηλώθηκαν στο μοντέλο μας είναι οι εξής:

- Κάθε κλάση τύπου Star στην ιδιότητα influences θα λαμβάνει τουλάχιστον μια τιμή από την κλάση Planet. Δηλαδή, αν ένα αστέρι ασκεί επιρροή σε αντικείμενα, τουλάχιστον ένα από αυτά θα είναι πλανήτης.

```
BNode restriction1 = Values.bnode();
owlModel.add(Star, RDFS.SUBCLASSOF, restriction1);
owlModel.add(restriction1, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction1, OWL.ONPROPERTY, influences);
owlModel.add(restriction1, OWL.SOMEVALUESFROM, Planet);
```

```
_:node1ih5dahhfx1 a owl:Restriction;
  owl:onProperty astronomy:influences;
  owl:someValuesFrom astronomy:Planet .
```

- Κάθε κλάση τύπου StellarObject στην ιδιότητα surroundedBy θα λαμβάνει τουλάχιστον μια τιμή από την κλάση Moon. Δηλαδή αν ένα αστρικό αντικείμενο περιτριγυρίζεται από αντικείμενα, τότε τουλάχιστον ένα από αυτά θα είναι φυσικός δορυφόρος (Moon).

```
BNode restriction2 = Values.bnode();
owlModel.add(StellarObject, RDFS.SUBCLASSOF, restriction2);
owlModel.add(restriction2, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction2, OWL.ONPROPERTY, surroundedBy);
owlModel.add(restriction2, OWL.SOMEVALUESFROM, Moon);
```

```
_:node1ih5dahhfx2 a owl:Restriction;
  owl:onProperty astronomy:surroundedBy;
  owl:someValuesFrom astronomy:Moon .
```

owl:allValuesFrom

Ο περιορισμός allValuesFrom δηλώνει ότι όλες οι τιμές μιας ιδιότητας πρέπει να ανήκουν σε μια συγκεκριμένη κλάση. Οι δύο σχετικοί περιορισμοί που δηλώθηκαν στο μοντέλο μας είναι οι εξής:

- Κάθε κλάση τύπου PlanetaryObject στην ιδιότητα orbitsAroundStar θα λαμβάνει τιμή μόνο από την κλάση MainSequenceStar. Δηλαδή, αν ένα πλανητικό αντικείμενο έχει τροχιά γύρω από ένα αστέρι, τότε αυτό το αστέρι θα είναι MainSequenceStar.

```
BNode restriction3 = Values.bnode();
owlModel.add(PlanetaryObject, RDFS.SUBCLASSOF, restriction3);
owlModel.add(restriction3, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction3, OWL.ONPROPERTY, orbitsAroundStar);
owlModel.add(restriction3, OWL.ALLVALUESFROM, MainSequenceStar);
```

```
_:node1ih5dahhfx3 a owl:Restriction;
  owl:allValuesFrom astronomy:MainSequenceStar;
  owl:onProperty astronomy:orbitsAroundStar .
```

- Κάθε κλάση τύπου Planet στην ιδιότητα collidedWith θα λαμβάνει τιμή μόνο από την κλάση Comet. Δηλαδή, αν ένας πλανήτης έχει συγκρουστεί με ένα αντικείμενο, τότε αυτό το αντικείμενο είναι κομήτης.

```
BNode restriction4 = Values.bnode();
owlModel.add(Planet, RDFS.SUBCLASSOF, restriction4);
owlModel.add(restriction4, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction4, OWL.ONPROPERTY, collidedWith);
owlModel.add(restriction4, OWL.ALLVALUESFROM, Comet);
```

```
_:node1ih5dahhfx4 a owl:Restriction;
  owl:allValuesFrom astronomy:Comet;
  owl:onProperty astronomy:collidedWith .
```

owl:hasValue

Ο περιορισμός hasValue δηλώνει ότι μια ιδιότητα έχει συγκεκριμένη τιμή. Οι δύο σχετικοί περιορισμοί που δηλώθηκαν στο μοντέλο μας είναι οι εξής:

- Κάθε κλάση τύπου Planet στην ιδιότητα orbitsAroundStar θα λαμβάνει την τιμή «Sun». Δηλαδή, αν ένας πλανήτης έχει τροχιά γύρω από ένα αντικείμενο, αυτό θα είναι ο ήλιος.

```
BNode restriction5 = Values.bnode();
owlModel.add(Planet, RDFS.SUBCLASSOF, restriction5);
owlModel.add(restriction5, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction5, OWL.ONPROPERTY, orbitsAroundStar);
owlModel.add(restriction5, OWL.HASVALUE, Sun);
```

```
_:node1ih5dahhfx5 a owl:Restriction;
  owl:hasValue astronomy:Sun;
  owl:onProperty astronomy:orbitsAroundStar .
```

- Κάθε κλάση τύπου Moon στην ιδιότητα partOfGalaxy θα λαμβάνει την τιμή «MilkyWay». Δηλαδή, αν ένας φυσικός δορυφόρος είναι μέρος ενός γαλαξία, αυτός ο γαλαξίας θα είναι ο MilkyWay.

```
BNode restriction6 = Values.bnode();
owlModel.add(Moon, RDFS.SUBCLASSOF, restriction6);
owlModel.add(restriction6, RDF.TYPE, OWL.RESTRICTION);
owlModel.add(restriction6, OWL.ONPROPERTY, partOfGalaxy);
owlModel.add(restriction6, OWL.HASVALUE, MilkyWay);
```

```
_:node1ih5dahhfx6 a owl:Restriction;
  owl:hasValue astronomy:MilkyWay;
  owl:onProperty astronomy:partOfGalaxy .
```

Τύποι ιδιοτήτων

Στο νέο μοντέλο ορίσθηκαν έξι τύποι ιδιοτήτων σε ιδιότητες με την βοήθεια της RDF4J, οι οποίοι περιγράφονται παρακάτω.

Συμμετρική ιδιότητα

Μια συμμετρική ιδιότητα P ορίζεται ως owl:SymmetricProperty και δηλώνει ότι αν για δύο αντικείμενα ισχύει [x:a x:P x:b] τότε ισχύει και [x:b x:P x:a]. Παρακάτω δίνονται οι δύο ιδιότητες που ορίσαμε ως συμμετρικές.

```
owlModel.add(connectedTo, RDF.TYPE, OWL.SYMMETRICPROPERTY);
owlModel.add(collidedWith, RDF.TYPE, OWL.SYMMETRICPROPERTY);
```

Επιμεριστική ιδιότητα

Μια επιμεριστική ιδιότητα ορίζεται ως owl:TransitiveProperty και δηλώνει ότι αν για τρία αντικείμενα ισχύει [x:a x:P x:b] και [x:b x:P x:c], τότε ισχύει και [x:a x:P x:c]. Παρακάτω δίνονται οι δύο ιδιότητες που ορίσαμε ως επιμεριστικές.

```
owlModel.add(influences, RDF.TYPE, OWL.TRANSITIVEPROPERTY);
owlModel.add(orbitsAround, RDF.TYPE, OWL.TRANSITIVEPROPERTY);
```

Ανάστροφη ιδιότητα

Οι ανάστροφες ιδιότητες ορίζονται ως owl:inverseOf και δηλώνουν ότι η μια είναι αντίστροφη της άλλης. Δηλαδή, αν για δύο ιδιότητες ισχύει [x:P1 owl:inverseOf x:P2] και [x:a x:P1 x:b] τότε ισχύει και [x:b x:P2 x:a]. Παρακάτω δίνονται οι ιδιότητες που ορίσαμε ως ανάστροφες.

```
owlModel.add(illuminatedBy, OWL.INVERSEOF, illuminates);
owlModel.add(hasArtificialSatellite, OWL.INVERSEOF, artificialSatelliteOf);
```

Προσθήκη νέων αντικειμένων

Για την προσθήκη νέων αντικειμένων στο μοντέλο δημιουργήθηκε η συνάρτηση *objectAddition()*. Μέσα σε αυτή, έγινε προσθήκη 10 αντικειμένων σε 5 κλάσεις. Οι κλάσεις στις οποίες προστέθηκαν τα αντικείμενα παρατίθενται παρακάτω:

1. UnmannedSpacecraft
2. Moon
3. Supernova
4. BlueGiantStar
5. SupermassiveBlackHole

Για την δημιουργία τους, αρχικά, δηλώθηκε το IRI τους. Στην συνέχεια, ορίστηκαν ο τύπος κλάσης του κάθε αντικείμενου, διάφορες ιδιότητες που έχουν και η ετικέτα τους. Όπου κρίθηκε απαραίτητο, προτέθηκαν τριπλέτες όπου τα νέα αντικείμενα λαμβάνουν ρόλο object, έτσι ώστε να «συνδεθούν» και με αυτόν τον τρόπο με άλλα ήδη υπάρχοντα αντικείμενα.

Φόρτωση στην GraphDB

Φόρτωση του μοντέλου

Για την φόρτωση του μοντέλου στην GraphDB χρησιμοποιήθηκε η RDF4J. Πρώτα απ' όλα δημιουργήθηκε ένα repository με όνομα ProjectB στο GraphDB, και μετά την ενημέρωση του μοντέλου (με τα παραπάνω) γίνεται η αποθήκευση του περιεχομένου του owlModel στο repository.

```
String repositoryName = "ProjectB";
HTTPRepository repository = new HTTPRepository( repositoryURL: "http://localhost:7200/repositories/" + repositoryName);

File rdfsFile = new File( pathname: "src/main/resources/astronomy1.ttl");
InputStream inputStream = new FileInputStream(rdfsFile);

try (RepositoryConnection connection = repository.getConnection()) {
```

```
// ΦΟΡΤΩΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΣΤΟ GRAPHDB
connection.clear();
connection.begin();
connection.add(owlModel);
connection.commit();
```

Δημιουργία και εκτέλεση SPARQL ερωτημάτων

Δημιουργήσαμε πέντε ερωτήματα SPARQL, τα οποία αναδεικνύουν τις νέες δυνατότητες που προστέθηκαν, δηλαδή περιορισμούς, τύπους ιδιοτήτων και νέα αντικείμενα. Τα ερωτήματα υλοποιήθηκαν στην συνάρτηση *sparqlQueries()*.

Ερώτημα #1

```
PREFIX astronomy: <http://example.org/astronomy#>
SELECT ?planet (COUNT (?moon) AS ?moonCount) ?star
WHERE {
  ?planet a astronomy:Planet ;
        astronomy:orbitsAroundStar ?star .
  ?moon astronomy:orbitsAroundPlanet ?planet .
}
GROUP BY ?planet ?star
HAVING (?moonCount > 1)
ORDER BY DESC(?moonCount)
```

Κάθε οντότητα του τύπου PlanetaryObject που σχετίζεται μέσω της ιδιότητας orbitsAroundStar μπορεί να περιστρέφεται αποκλειστικά γύρω από οντότητες της κλάσης MainSequenceStar. Ο περιορισμός owl:allValuesFrom διασφαλίζει ότι τα δεδομένα μας είναι συνεπή με το μοντέλο, στο οποίο ορίζουμε ότι ένας πλανήτης μπορεί να περιστρέφεται μόνο γύρω από αστέρια της κλάσης MainSequenceStar. Το ερώτημα που χρησιμοποιούμε επαληθεύει αυτόν τον περιορισμό, εμφανίζοντας μια λίστα πλανητών με

τον αριθμό των φεγγαριών τους και το αστέρι γύρω από το οποίο περιστρέφονται. Τα αποτελέσματα εμφανίζονται σε φθίνουσα σειρά ώστε οι πλανήτες με τα περισσότερα φεγγάρια να εμφανίζονται πρώτοι.

```
QUERY #1
Planet     | Moon Count | Star
-------------------------------
Jupiter    | 4          | Sun
Saturn     | 2          | Sun
```

Το αστέρι Sun είναι τύπου MainSequenceStar.

Ερώτημα #2

```
PREFIX astronomy: <http://example.org/astronomy#>
SELECT ?astrObject ?artSat
WHERE {
  ?artSat a astronomy:ArtificialSatellite.
  ?astrObject a astronomy:AstronomicalObject.
  ?artSat astronomy:artificialSatelliteOf ?astrObject.
}
ORDER BY (?artSat)
```

Η ιδιότητα hasArtificialSatellite ορίζεται ως η αντίστροφη της ιδιότητας artificialSatelliteOf. Αυτό σημαίνει ότι αν ένας τεχνητός δορυφόρος συνδέεται με ένα AstronomicalObject μέσω της ιδιότητας artificialSatelliteOf, τότε το AstronomicalObject συνδέεται με τον τεχνητό δορυφόρο μέσω της ιδιότητας hasArtificialSatellite. Στο μοντέλο, τα αντικείμενα χρησιμοποιούν μόνο την ιδιότητα hasArtificialSatellite. Το ερώτημα που χρησιμοποιούμε εμφανίζει μια λίστα από ζεύγη τεχνητών δορυφόρων και των αστρονομικών αντικειμένων γύρω από τα οποία περιστρέφονται, επιτρέποντας την επαλήθευση ότι όλες οι σχέσεις δηλώνονται σωστά και συμφωνούν με την αντίστροφη ιδιότητα. Τα αποτελέσματα ταξινομούνται αλφαβητικά βάσει του ονόματος των τεχνητών δορυφόρων.

```
QUERY #2
Astronomical Object  | Artificial Satellite
---------------------------------------------
Venus                | Akatsuki
Earth                | GPS
Earth                | Goes
Mars                 | InSight
Earth                | Iridium
Mercury              | MESSENGER
```

Ερώτημα #3

```
PREFIX astronomy: <http://example.org/astronomy#>
SELECT ?object ?type
WHERE {
  ?stellarObj a astronomy:StellarObject;
        astronomy:surroundedBy ?object.
  ?object a ?type.
  FILTER (?type = astronomy:Moon || ?type = astronomy:Planet)
}
LIMIT 10
```

Σύμφωνα με το μοντέλο, κάθε StellarObject πρέπει να περιβάλλεται τουλάχιστον από ένα Moon, όπως καθορίζεται από τον περιορισμό owl:someValuesFrom. Το ερώτημα που χρησιμοποιούμε επιτρέπει την επαλήθευση ότι οι σχέσεις surroundedBy στα δεδομένα συμφωνούν με τον περιορισμό, εφόσον δείχνει ότι υπάρχουν τόσο πλανήτες όσο και φεγγάρια που περιβάλλουν τα αστρικά αντικείμενα. Τελικά, εμφανίζονται τα 10 πρώτα ζεύγη ώστε να διευκολύνεται η ανάλυση.

```
QUERY #3
Object     | Type Of Object
---------------------------
EarthMoon  | Moon
Enceladus  | Moon
Europa     | Moon
Ganymede   | Moon
Phobos     | Moon
Miranda    | Moon
Triton     | Moon
Titan      | Moon
Earth      | Planet
Mars       | Planet
```

Ερώτημα #4

```
PREFIX astronomy: <http://example.org/astronomy#>
SELECT ?subject ?galaxy
WHERE {
  ?subject a astronomy:Moon;
       astronomy:partOfGalaxy ?galaxy;
       astronomy:observedByTelescope ?telescope.
  FILTER(?telescope = astronomy:HubbleSpaceTelescope)
}
ORDER BY DESC(?subject)
```

Σύμφωνα με το μοντέλο, κάθε Moon πρέπει να ανήκει στον γαλαξία MilkyWay, όπως ορίζεται από τον περιορισμό owl:hasValue. Το ερώτημα εντοπίζει φεγγάρια που παρατηρήθηκαν από το τηλεσκόπιο Hubble Space Telescope, καθώς και τον γαλαξία στον οποίο ανήκουν. Με αυτόν τον τρόπο, μπορούμε να επαληθεύσουμε αν τα δεδομένα μας συμφωνούν με τον περιορισμό. Τα αποτελέσματα ταξινομούνται σε φθίνουσα σειρά βάσει του ονόματος των φεγγαριών, ενώ επιστρέφουν πληροφορίες για τον γαλαξία στον οποίο ανήκουν, ο οποίος πρέπει να έχει την τιμή MilkyWay.

```
QUERY #4
Subject    | Galaxy
-----------------------
Titan      | MilkyWay
Phobos     | MilkyWay
Ganymede   | MilkyWay
Europa     | MilkyWay
Enceladus  | MilkyWay
EarthMoon  | MilkyWay
```

Ερώτημα #5

```
PREFIX astronomy: <http://example.org/astronomy#>
SELECT ?subject ?object
WHERE {
  ?subject a ?type ;
        astronomy:influences ?object.
  FILTER(?type = astronomy:Planet || ?type = astronomy:Moon)
}
ORDER BY(?subject)
```

Η ιδιότητα influences έχει οριστεί ως μεταβατική, γεγονός που σημαίνει ότι αν ένα αντικείμενο επηρεάζει ένα δεύτερο και αυτό με τη σειρά του επηρεάζει ένα τρίτο, τότε το πρώτο αντικείμενο θεωρείται ότι επηρεάζει και το τρίτο. Σύμφωνα με το μοντέλο, κάθε αστέρι Star πρέπει να επηρεάζει τουλάχιστον έναν πλανήτη Planet, όπως ορίζεται από τον περιορισμό owl:someValuesFrom. Το ερώτημα εξετάζει δεδομένα σχετικά με πλανήτες και φεγγάρια που επηρεάζουν άλλα αντικείμενα. Τα αποτελέσματα εμφανίζονται ταξινομημένα αλφαβητικά βάσει των υποκειμένων. Παράλληλα, επαληθεύει ότι τα δεδομένα μας συμφωνούν με τον περιορισμό.

Subject	Object
Callisto	Jupiter
Earth	Earth
Earth	EarthMoon
EarthMoon	Earth
EarthMoon	EarthMoon
Enceladus	Saturn
Enceladus	Enceladus
Enceladus	Titan
Europa	Jupiter
Ganymede	Jupiter
Io	Jupiter
Mars	Earth
Mars	Jupiter
Mars	EarthMoon
Miranda	Uranus
Phobos	Earth
Phobos	Mars
Phobos	Jupiter
Phobos	EarthMoon
Saturn	Saturn
Saturn	Enceladus
Saturn	Titan
Titan	Saturn
Titan	Enceladus
Titan	Titan
Triton	Neptune
Venus	Earth
Venus	EarthMoon

Για να διευκολυνθεί η ανάλυση δίνονται κάποια παραδείγματα που υπάρχουν στο αποτέλεσμα που επαληθεύουν την μεταβατική ιδιότητα που τέθηκε ως περιορισμός:

- astronomy:Enceladus astronomy:influences astronomy:Saturn
  astronomy:Saturn astronomy:influences astronomy:Titan
  astronomy:Enceladus astronomy:influences astronomy:Titan

- astronomy:Mars astronomy:influences astronomy:Earth
  astronomy:Earth astronomy:influences astronomy:EarthMoon
  astronomy:Mars astronomy:influences astronomy:EarthMoon

- astronomy:Venus astronomy:influences astronomy:Earth
  astronomy:Earth astronomy:influences astronomy:EarthMoon
  astronomy:Venus astronomy:influences astronomy:EarthMoon

Εξαγωγή στατιστικών δεδομένων με RDF4J API

Για την εξαγωγή στατιστικών δεδομένων από το μοντέλο owl δημιουργήθηκε η συνάρτηση *statistics()*. Για τους περισσότερους υπολογισμούς έγινε χρήση HashMap<String, Integer>, για την διευκόλυνση εξαγωγής αποτελεσμάτων. Οι υπολογισμοί έγιναν ταυτόχρονα σε μια for loop, η οποία ελέγχει κάθε statement που υπάρχει το owlModel. Παρακάτω περιγράφεται αναλυτικά η διαδικασία για την εξαγωγή αποτελέσματος κάθε ζητούμενου.

Συχνότητα εμφάνισης συγκεκριμένων ιδιοτήτων

Για να βρούμε την συχνότητα εμφάνισης ιδιοτήτων, υπολογίσαμε πόσες φορές χρησιμοποιείται κάθε ιδιότητα από αντικείμενα στο μοντέλο. Συγκεκριμένα, για κάθε statement προσθέσαμε το predicate του στο HashMap *propertyFreq*. Κάθε φορά που ένα predicate εισάγεται για πρώτη φορά στο map, η default τιμή του (value) είναι μηδέν, ενώ αν υπάρχει ήδη τότε αυξάνεται κατά ένα. Έτσι, στο τέλος, καταλήγουμε να έχουμε ζεύγη από predicates και συχνότητα εμφάνισής τους. Στα αποτελέσματα υπάρχουν φυσικά και οι ιδιότητες που προέκυψαν μετά από reasoning του GraphDB, όπως η sameAs.

```
// ΣΥΧΝΟΤΗΤΑ ΕΜΦΑΝΙΣΗΣ ΣΥΓΚΕΚΡΙΜΕΝΩΝ ΙΔΙΟΤΗΤΩΝ
propertyFreq.put(predicate, propertyFreq.getOrDefault(predicate, defaultValue: 0) + 1);
```

Αποτελέσματα:

```
Frequency of each property:
Property                    | Frequency
-------------------------------------------
observedBy                  | 28
hasCrewCapacity             | 2
discoveredBy                | 57
hasGravity                  | 36
launchDate                  | 31
type                        | 989
relatedTo                   | 88
launchedBy                  | 8
hasActiveStatus             | 31
interactsWith               | 1
wasObservedOn               | 14
passedNear                  | 5
subClassOf                  | 249
hasOperator                 | 31
imports                     | 1
hasOrbitType                | 5
orbitsAroundStar            | 17
equivalentProperty          | 12
hasDiameter                 | 54
hasMass                     | 54
dockedAt                    | 2
causedBy                    | 4
hasValue                    | 2
versionInfo                 | 1
hasLife                     | 27
hasBrightness               | 9
hasColor                    | 8
visitedBy                   | 24
orbitsAroundPlanet          | 10
hasGalacticCenter           | 4
domain                      | 318
hasOrbitalSpeed             | 2
onProperty                  | 6
monitors                    | 1
observedBySpacecraft        | 4
hasArtificialSatellite      | 6
spectralType                | 15
equivalentClass             | 2
visibleToNakedEye           | 14
collidedWith                | 2
range                       | 248
hasAltitudeFromEarth        | 2
discoveredDate              | 43
illuminatedBy               | 27
hasMaxCoverage              | 4
subPropertyOf               | 86
surroundedBy                | 26
hasDuration                 | 14
influences                  | 35
hasPhysicalProperty         | 96
allValuesFrom               | 2
hasTemperature              | 54
partOfGalaxy                | 48
observedFrom                | 14
inverseOf                   | 2
hasSatellite                | 8
providesServiceTo           | 1
hasOrbitalPeriod            | 27
label                       | 191
someValuesFrom              | 2
orbitsAround                | 17
hasSpin                     | 7
observedByTelescope         | 46
hasFocalLength              | 5
visibleFrom                 | 25
sameAs                      | 301
```

Όπως παρατηρούμε, οι πιο συχνή ιδιότητα είναι η type, ακολουθούμενη από τις domain, sameAs και subClassOf. Οι πιο συχνές ιδιότητες που δημιουργήσαμε εμείς για το μοντέλο είναι οι hasPhysicalProperty και relaterTo.

Κατανομή τύπων αντικειμένων

Για να κάνουμε κατανομή τύπων αντικειμένων, υπολογίσαμε πόσα αντικείμενα ανήκουν σε κάθε κλάση. Συγκεκριμένα, ελέγξαμε αν στο statement που επεξεργαζόμαστε κάθε φορά έχουμε ως κατηγόρημα το RDF.TYPE, το οποίο προδίδει τον ορισμό αντικειμένου. Για κάθε statement που είναι τέτοιας μορφής, πήραμε το object του, το οποίο είναι ουσιαστικά η κλάση στην οποία ανήκει το αντικείμενο. Το προσθέσαμε στο HashMap *numOfClassObjects,* και ως value του map ανανεώναμε κάθε φορά τον αριθμό εμφάνισής της κλάσης. Για να γίνει πιο ξεκάθαρο, μετρήσαμε πόσες φορές εμφανίζεται μια κλάση σε τριπλέτες τύπου [subject, rdf.type, class], διότι μια τέτοια τριπλέτα δηλώνει ένα αντικείμενο που ανήκει στην class. Στα αποτελέσματα υπάρχουν φυσικά και κλάσεις που προέκυψαν μετά από reasoning του GraphDB, όπως η AnnotationProperty και OntologyProperty.

```
// ΚΑΤΑΝΟΜΗ ΤΥΠΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ
if (statement.getPredicate().equals(RDF.TYPE)) {
    String objectIRI = statement.getObject().toString();
    String object = objectIRI.substring( beginIndex: objectIRI.lastIndexOf( ch: '#') + 1);

    numOfClassObjects.put(object, numOfClassObjects.getOrDefault(object, defaultValue: 0) + 1);
}
```

Αποτελέσματα:

```
Number of objects that belong to each class:
Class                          | # of Objects
-----------------------------------------------
Galaxy                         | 4
AstronomicalObject             | 28
Telescope                      | 5
SpaceStation                   | 2
Moon                           | 10
PlanetaryObject                | 17
SpaceArtifact                  | 22
Eclipse                        | 4
Elliptical                     | 2
TransitiveProperty             | 2
StellarBlackHole               | 2
DwarfPlanet                    | 4
NavigationSatellite            | 1
Datatype                       | 34
StellarObject                  | 7
MainSequenceStar               | 2
List                           | 1
Planet                         | 8
Thing                          | 301
DatatypeProperty               | 42
Supernova                      | 10
ShortPeriodComet               | 3
ObjectProperty                 | 128
RedSuperGiantStar              | 1
OntologyProperty               | 5
Resource                       | 55
UnmannedSpacecraft             | 16
SupermassiveBlackHole          | 10
SymmetricProperty              | 2
Comet                          | 5
LunarEclipse                   | 2
Class                          | 112
WeatherSatellite               | 1
SpaceObject                    | 50
Ontology                       | 1
LongPeriodComet                | 2
GiantStar                      | 1
ContainerMembershipProperty    | 1
CommunicationSatellite         | 1
BlueSuperGiantStar             | 1
Star                           | 5
BlueGiantStar                  | 10
CelestialBody                  | 24
SuperGiantStar                 | 2
RedGiantStar                   | 1
Spacecraft                     | 9
MannedSpacecraft               | 2
ArtificialSatellite            | 6
BlackHole                      | 2
Spiral                         | 2
AstronomicalPhenomenon         | 4
Restriction                    | 6
SolarEclipse                   | 2
AnnotationProperty             | 9
```

Όπως παρατηρούμε, τα περισσότερα αντικείμενα είναι τύπου Thing (που προϋπήρχε στο μοντέλο από το TopBraid). Τα περισσότερα αντικείμενα σε κλάση που εμείς έχουμε ορίσει είναι τύπου SpaceObject.

Μέσος αριθμός συνδέσεων ανά αντικείμενο

Για να βρούμε τον μέσο αριθμό συνδέσεων ανά αντικείμενο, υπολογίσαμε τον μέσο αριθμό των ιδιοτήτων που έχει κάθε αντικείμενο. Συγκεκριμένα, κρατήσαμε στο HashMap *subjectConnections* τα αντικείμενα και τον αριθμό εμφάνισής τους σε statements ως subject. Με την βοήθεια του size() βρήκαμε τον συνολικό αριθμό των (distinct) subjects που υπάρχουν στο map, και στην συνέχεια, αθροίσαμε τις συνδέσεις όλων των αντικειμένων. Ο μέσος όρος προκύπτει από τον λόγο του αθροίσματος των συνδέσεων προς τον συνολικό αριθμό αντικειμένων.

```
// ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΣΥΝΔΕΣΕΩΝ ΑΝΑ ΑΝΤΙΚΕΙΜΕΝΟ
String subjectIRI = statement.getSubject().toString();
String subject = subjectIRI.substring( beginIndex: subjectIRI.lastIndexOf( ch: '#') + 1);

subjectConnections.put(subject, subjectConnections.getOrDefault(subject,  defaultValue: 0) + 1);
```

(έξω από την for loop:)

```
double numOfSubjects = subjectConnections.size();
Integer sumOfConnections = 0;
for (Integer connection : subjectConnections.values()) {
    sumOfConnections += connection;
}
double connection_avg = numOfSubjects > 0 ? sumOfConnections / numOfSubjects : 0;
```

Αποτελέσματα:

```
Average number of connections per object: 9,93
```

Συμπεραίνουμε ότι το μοντέλο δεν είναι ιδιαίτερα πολύπλοκο, αφού τα περισσότερα αντικείμενα έχουν περίπου 10 ιδιότητες, αριθμός που δεν θεωρείται πολύ μεγάλος.

Συνολικός αριθμός τριπλετών

Για τον υπολογισμό του συνολικού αριθμού τριπλετών του μοντέλου, κρατήσαμε έναν μετρητή μέσα στην for loop, ο οποίος αυξάνεται κατά 1 κάθε φορά, ώστε στο τέλος να είναι ίσος με τον αριθμό των statements.

```
// ΣΥΝΟΛΙΚΟΣ ΑΡΙΘΜΟΣ ΤΡΙΠΛΕΤΩΝ
numOfTriplets += count;
```

Αποτελέσματα:

```
Total number of triplets: 3505
```

Όπως βλέπουμε, το μοντέλο μας είναι αρκετά μεγάλο σε μέγεθος.

Κατανομή τιμών για ιδιότητες

Για να κάνουμε κατανομή των τιμών για τις ιδιότητες, αναλύσαμε τους τύπους τιμών των ιδιοτήτων και υπολογίσαμε το πλήθος των ιδιοτήτων με τιμές τύπου Literal (ακέραιοι, String, Boolean, κλπ.). Συγκεκριμένα, ελέγξαμε σε κάθε statement αν το object του είναι τύπου Literal.Σε περίπτωση που είναι,

και ο τύπος του είναι έγκυρος (έχει οριστεί σωστά, κλπ), τον αποθηκεύουμε στο HashMap *usesOfDatatypeProperties* κρατώντας την συχνότητα εμφάνισής του. Αυτό ουσιαστικά μας δείχνει πόσες φορές, από πόσες ιδιότητες, χρησιμοποιήθηκε ο τύπος.

```
// ΚΑΤΑΝΟΜΗ ΤΙΜΩΝ ΓΙΑ ΙΔΙΟΤΗΤΕΣ
if (statement.getObject() instanceof Literal) {
    Literal datatypeObjLit = (Literal) statement.getObject();
    IRI datatypeIRI = datatypeObjLit.getDatatype();
    if (datatypeIRI != null) {
        String datatypeObj = datatypeIRI.getLocalName();
        usesOfDatatypeProperties.put(datatypeObj, usesOfDatatypeProperties.getOrDefault(datatypeObj, defaultValue: 0) + 1);
    }
}
```

Αποτελέσματα:

```
Value Distribution for properties:
Value type      | # of Properties that use them
-------------------------------------------------
date            | 88
boolean         | 72
string          | 629
integer         | 6
decimal         | 29
int             | 10
```

Όπως παρατηρούμε, τα περισσότερα literal δεδομένα που υπάρχουν στο μοντέλο είναι τύπου String, ενώ έχουμε ελάχιστα δεδομένα τύπου integer.

Συχνότητα χρήσης συγκεκριμένων κλάσεων

Για να βρούμε την συχνότητα χρήσης συγκεκριμένων κλάσεων, υπολογίσαμε πόσες φορές χρησιμοποιούνται οι κλάσεις στο μοντέλο σε συγκεκριμένες ιδιότητες. Οι ιδιότητες που επιλέξαμε είναι οι πιο γνωστές και απαραίτητες, ιδιότητες που προϋπήρχαν και δεν τις ορίσαμε εμείς (δεν είναι στο χώρο του astronomy δηλαδή, είναι είτε owl είτε rdf/rdfs). Οι ιδιότητες στις οποίες χρησιμοποιούνται οι κλάσεις που μετρήσαμε είναι οι εξής:

- RDF:TYPE
- RDF:SUBCLASSOF
- RDF:RANGE
- RDF:DOMAIN
- OWL:SOMEVALUESFROM
- OWL:ALLVALUESFROM
- OWL:HASVALUE

Συγκεκριμένα, ελέγξαμε αν το object κάθε statement είναι τύπου IRI. Αν είναι IRI, αποκλείουμε με έλεγχο και την πιθανότητα να είναι Literal. Έτσι, συνεχίζουμε ελέγχοντας αν η ιδιότητα που έχει η κλάση κάθε φορά ανήκει σε μια από τις παραπάνω, και ανάλογα προσθέτουμε την κλάση που χρησιμοποιείται με αυτή στο HashMap *usesOfCertainClasses* όπου κρατάμε και την συχνότητα εμφάνισής της (άρα το πόσες φορές χρησιμοποιείται η κλάση με συγκεκριμένη ιδιότητα).

```java
// ΣΥΧΝΟΤΗΤΑ ΧΡΗΣΗΣ ΣΥΓΚΕΚΡΙΜΕΝΩΝ ΚΛΑΣΕΩΝ
IRI pred = statement.getPredicate();
Value obj = statement.getObject();
String class_ = obj.stringValue();
String class_name = class_.substring( beginIndex: class_.lastIndexOf( ch: '#') + 1);

if (obj instanceof IRI) {
    if (!((IRI) obj).getNamespace().equals("http://www.w3.org/2001/XMLSchema#")) {
        if (pred.toString().equals("http://www.w3.org/1999/02/22-rdf-syntax-ns#type")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2000/01/rdf-schema#subClassOf")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2000/01/rdf-schema#range")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2000/01/rdf-schema#domain")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2002/07/owl#someValuesFrom")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2002/07/owl#allValuesFrom")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
        else if (pred.toString().equals("http://www.w3.org/2002/07/owl#hasValue")) {
            usesOfCertainClasses.put(class_name, usesOfCertainClasses.getOrDefault(class_name, defaultValue: 0) + 1);
        }
    }
}
```

Αποτελέσματα:

Usage Frequency of certain Classes			
Class	Frequency	UnmannedSpacecraft	16
Galaxy	8	SupermassiveBlackHole	10
AstronomicalObject	100	SymmetricProperty	2
Telescope	7	Sun	1
SpaceStation	7	Comet	9
Moon	13	LunarEclipse	2
PlanetaryObject	45	NegativePropertyAssertion	4
SpaceArtifact	54	Class	177
Eclipse	8	WeatherSatellite	2
Elliptical	2	SpaceObject	156
TransitiveProperty	2	Ontology	11
StellarBlackHole	2	LongPeriodComet	2
DwarfPlanet	4	GiantStar	3
Nothing	1	ContainerMembershipProperty	1
NavigationSatellite	2	CommunicationSatellite	1
Datatype	41	BlueSuperGiantStar	1
AllDifferent	1	Star	17
Container	3	BlueGiantStar	10
Literal	75	CelestialBody	84
StellarObject	27	Property	16
MainSequenceStar	3	SuperGiantStar	4
List	14	MilkyWay	1
Planet	18	RedGiantStar	1
Thing	316	Statement	3
DatatypeProperty	42	Spacecraft	16
Supernova	11	MannedSpacecraft	4
ShortPeriodComet	3	ArtificialSatellite	17
ObjectProperty	137	BlackHole	6
RedSuperGiantStar	1	Spiral	2
OntologyProperty	5	AstronomicalPhenomenon	15
Resource	196	Restriction	20
		SolarEclipse	2
		AnnotationProperty	9

Όπως παρατηρούμε, σε σχέση με το μέγεθος του μοντέλου, δεν χρησιμοποιούνται πολλές φορές κλάσεις με συγκεκριμένες ιδιότητες. Οι κλάσεις που χρησιμοποιούνται περισσότερο απ' όλες με τις ιδιότητες που προαναφέρθηκαν είναι οι Thing και Resource που ήδη υπήρχαν από το TopBraid, ακολουθεί η Class και από αυτές που εμείς κατασκευάσαμε είναι η SpaceObject.