



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Συστήματα & Τεχνολογίες Γνώσης

Project

Γεωργίου Δημήτριος (03115106)

<el15106@central.ntua.gr>

Βασιλείου Βασιλική (03115033)

<el15033@central.ntua.gr>

Ιούλιος 2019

1 Σκοπός της Άσκησης

1.1 Γενική Περιγραφή

- Στο παρόν project θα δημιουργήσουμε μια **σημασιολογική βάση γνώσης** με σκοπό την διαχείριση και εξυπηρέτηση ερωτημάτων χρηστών (**queries**) που σχετίζονται με οπτικοακουστικές παραγωγές (κινηματογραφικές ταινίες τηλεοπτικές σειρές, ταινίες μικρού μήκους).
- **Πιο συγκεκριμένα** θα δημιουργήσουμε μια **οντολογία**, στην οποία:
 - Αρχικά θα μοντελοποιηθούν δεδομένα της πηγής δεδομένων IMDB, ενώ θα γίνουν και σημασιολογικές επεκτάσεις στο τελευταίο βήμα. Επομένως εξασφαλίζεται η **επάρκεια** της βάσης ως προς τα μοντελοποιημένα δεδομένα σε πλήθος και διαφορετικότητα
 - Συγκεκριμένα, θα αναλυθεί και παρακάτω, τα δεδομένα που αφορούν την πηγή IMDB μετατράπηκαν από μορφή `.tsv` σε `.csv` και τελικά σε τριάδες *N-Triples* (*n3*) *RDF*, οπότε και ολοκλήρωθηκε επιτυχώς η αναπαράσταση εννοιών, ρόλων και ιδιοτήτων τύπων δεδομένων ως στιγμιότυπα της σημασιολογικής βάσης¹
 - Τέλος μέσω κατάλληλων **queries** έγινε η διασύνδεση της τοπικής βάσης με τρίτες πηγές Σημασιολογικού Ιστού **Semantic Web**²

1.2 Βήματα που ακολουθήθηκαν

Παρακάτω θα γίνει μια ενδεικτική ανασκόπηση των βημάτων που ακολουθήθηκαν, που έχουν δομή παρόμοια με αυτή της εκφώνησης, με μικρές τροποποιήσεις

1. Έγινε download των κατάλληλων αρχείων `.tsv` των ταινιών και των ατόμων, που εμπλέκονται με οπτικοακουστικές παραγωγές, και τα οποία βρίσκονται στα `/interfaces` της IMDB. Βάσει αυτών και ορισμένων υποθέσεων, έγινε σχεδιασμός της αντίστοιχης οντολογίας *OWL2* με χρήση του εργαλείου **Protégé**
2. Έγινε run 2 script `.py` τα οποία κάνουν parsing όλα τα αντίστοιχα `.csv` αρχεία και παράγουν σε buckets/chunks μεγέθους 10000, *RDFs*³ τύπου:
 - **Data Properties** μεταξύ των individuals movie/person και των αντίστοιχων fields και για τις αναπαράσταση των ιδιοτήτων⁴
 - **Object Properties** μεταξύ individuals για αναπαράσταση των κατάλληλων ρόλων/
3. Για όλα τα παραπάνω *RDFs*, περνάμε τους αντίστοιχους φακέλους **data & object properties** στο content της εφαρμογής **Virtuoso**⁵, δημιουργώντας έτσι την μεγάλη σε όγκο (περίπου ~ 8GB δεδομένα), τοπική, Web Semantic Database, αφού έγιναν τα κατάλληλα configurations πρώτα (πχ. loading, checkpoint κτλ.)
4. Το βήμα αυτό, που για εμάς είχε την μεγαλύτερη σημαντικότητα σε επίπεδο web databases, αφορούσε την υποβολή ενδιαφερόντων ερωτήματων SPARQL⁶. Η κατασκευή των παραπάνω ερωτήματων βασίστηκε στην ταυτοποίηση των πόρων μας (πχ. σχεδόν όλες τις φορές βάσει του **primary key** : `imdb id`) με τους αντίστοιχους πόρους των τρίτων Web Semantic Databases των **Wikidata** και **Dbpedia**, ενώ έπρεπε να γίνει κατάλληλη χρήση συναρτήσεων της SPARQL για κατάλληλη διαχείριση των αποτελεσμάτων/*strings* που πρέπει να επιστραφούν.
5. Τέλος, έγινε εμπλουτισμός της τοπικής Web Semantic Database με υποβολή extra ερωτημάτων σε SPARQL, που κατα κύριο λόγο αφορούν εκ νέου την διασύνδεση με τις παραπάνω τρίτες πηγές για εξαγωγή επιπλέον πληροφοριών για τα movies/persons.

¹state = 2 για όλα τα αρχεία `rdf` που δημιουργήσαμε και ανέβηκαν στην βάση μέσω του **virtuoso**

²με κατάλληληση εφαρμογή των **services** στα αντίστοιχα **queries**

³Ο κατακερματισμός αυτός των `rdf` γίνεται για σκοπούς εύκολης διαχείρισης των τόσο μεγάλου πλήθους δεδομένων **big data**.

Αυτό συνοεί και την λειτουργία του **Virtuoso**

⁴Θα αναφερόμαστε στις αντίστοιχες αγγλικές ορολογίες από εδώ και πέρα για καλύτερη κατανόηση, ειδικά όταν γίνει και επεξήγηση των **queries**

⁵Η διαχείριση των προβλημάτων που αντιμετωπίσαμε θα γίνει στην επόμενη ενότητα

⁶Το αντίστοιχο end-point για ερωτήματα στην τοπική βάση δεν απαιτούσε διασύνδεση στο διαδίκτυο, ενώ αυτά της **Wikidata** και της **Dbpedia** απαιτούσαν

2 Κύριο Μέρος

2.1 Δημιουργία Οντολογίας σε Protege

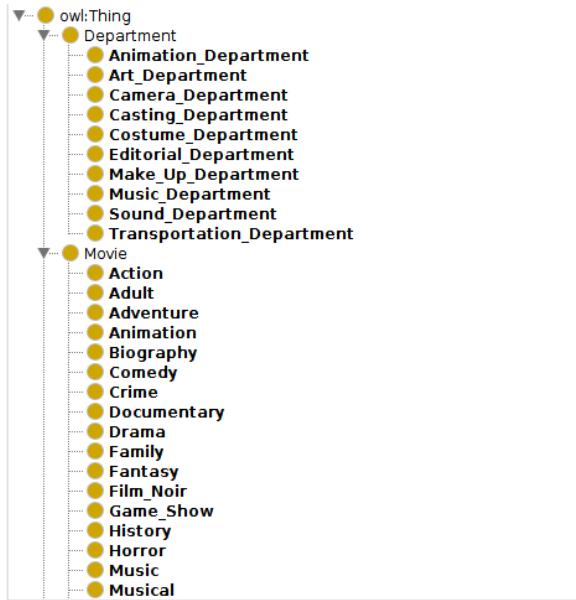
- Αρχικά έγινε download των κατάλληλων αρχείων .tsv του /interfaces της IMDB.

- name.basics.tsv
- title.basics.tsv
- title.principals.tsv
- title.ratings.tsv
- title.crew.tsv

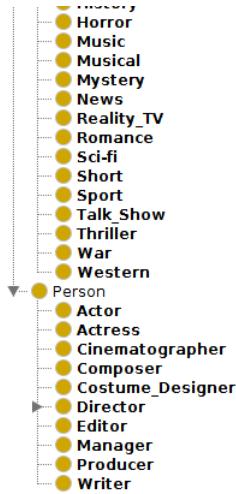
- Αφού μελετήσαμε λίγο την δομή τους διαπιστώσαμε τα εξής, σε επίπεδο σχεδιαστικό

- **Σχετικά με τα classes**

- * Τα πεδία **nconst** και **tconst** χρησιμοποιούνται ως primary keys για την ταυτοποίηση των πόρων των ταινιών και των συσχετιζόμενων με οπτικοακουστικές παραγωγές ατόμων. Έτσι αποφασίσαμε να διαχωρίσουμε τα δεδομένα σε δύο μεγάλες κλάσεις/classes **Movie**⁷ και **Person**
- * Πολύ εύκολα διαπιστώσαμε ότι υπάρχουν πολλά διαφορετικά είδη εμπλεκόμενων, πχ director, composer, το οποίο ενώ θα μπορούσαμε να το μοντελοποιήσουμε ως ιδιότητα του Person, επιλέχθηκε να υλοποιηθούνε ως **Subclasses** της Person για λόγους πληρότητας, και σωστής διεκπαρέωσης ερωτημάτων SPARQL.
- * Στο ίδιο καταλήξαμε και για τα διαφορετικά είδη/genres Movie που υπάρχουν, καθώς κάθε Movie μπορεί να ανήκει σε 1 ή περισσότερα είδη.
- * Μάλιστα αξίζει να αναφερθεί ότι με κατάλληλο script .py (με εντοές μορφής `set(list(fieldi))`, με `fieldi = {job, genre}`) εκτυπώσαμε τα διαφορετικά είδη Movie και Person που υπάρχουν και μετά ορίσαμε τις παραπάνω subcalsses. Συγκεκριμένα γιά το Person κάποιες που θεωρήσαμε ασήμαντες δεν τις ορίσαμε.
- * *Ta classes φαίνονται παρακάτω*

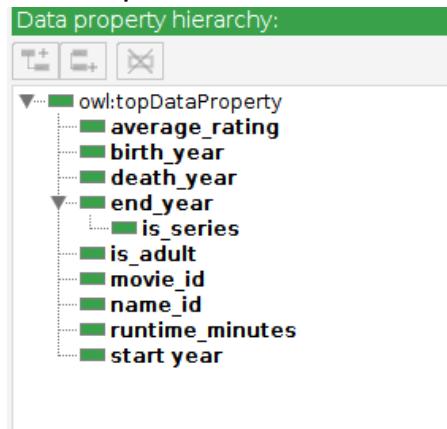


⁷ Είναι πολύ σημαντικό να αναφερθεί ότι η έννοια Movie αντιπροσωπεύει ταινίες αλλά και τηλεοπτικές σειρές



– Σχετικά με τα data properties - ιδιότητες

- * Σε μια απόπειρα μοντελοποίησης των fields *type*, *primary title*, *original title*, *isAdult*, *start year*, *end year*, *rating*, *voting* Τα αρχεία *title.basics* και *title.ratings* λήφθηκαν υπόψιν συνδυαστικά για την εξαγωγή της πληροφορίας σχετικά με τις ιδιότητες των Movies για τα Movies, ορίστηκαν οι αντίστοιχες ιδιότητες (και με τα ίδια ονόματα) *type*, *primary title*, *original title*, etc...
- * Αντίστοιχα για τα Person ορίστηκαν αποκλειστικά οι ιδιότητες *name*, *birth year*, *death year* και μόνο.
- * *Ta data properties φαίνονται παρακάτω*



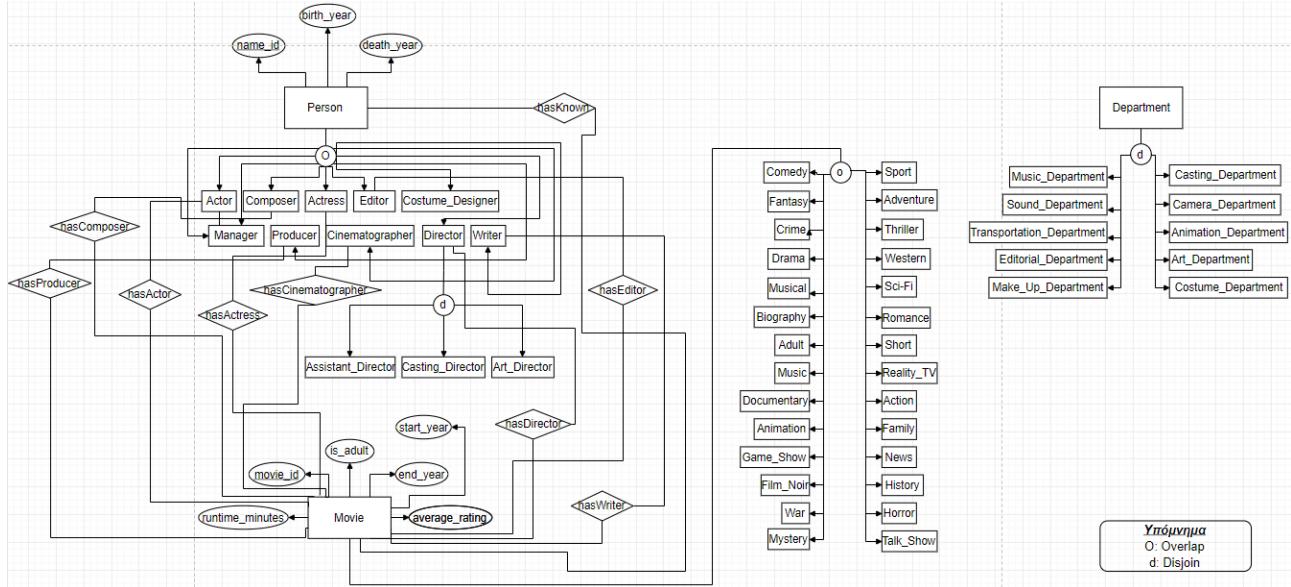
– Σχετικά με τα object properties - ρόλοι

- * Γενικά είναι λογικό να αναφερθεί ότι κάθε instance του Movie (ανεξαρτήτως Genre) έχει X εμπλεκόμενους Person, ή καλύτερα X εμπλεκόμενα άτομα για τα οποία γνωρίζουμε το job τους. Άρα θα ορίσουμε ρόλους μεταξύ της γενικής class Movie και των υποκλάσεων Director, Editor, Composer, etc... της Person ΚΑΙ τις αντίστροφες/inverse (με τα κάταλλη λα ονόματα φυσικά και τα καταλληλα configuration στο Protege)
- * *Ta object properties φαίνονται παρακάτω*



- Αν και το εργαλείο της Protege απαιτείται για τον καλό σχεδιασμό της αντίστοιχης οντολογίας της IMDB, η μη κατάλληλη ονοματοδότηση των **classes**, **data properties** και **object properties** δεν θα προκαλούσε χάπια δυσλειτουργία για τα επόμενα βήματα και τελικά την υποβολή ερωτημάτων SPARQL. **Παρ' όλα αυτά**, για λόγους πληρότητας λάβαμε υπόψιν την παραμετροποίηση των ονοματισμένων **classes**, **data properties** και **object properties** της οντολογίας του αρχείου **IMDB-Ontology.owl**, και βάσει αυτής ορίστηκαν και οι 3άδες rdfs του επόμενου βήματος.

- To **EER (Enhanced Entity Relational) Model** φαίνεται παρακάτω:



2.2 Δημιουργία 3άδων N-triples

- Γενικά, για την 1-1 αντιστοίχιση της παραπάνω οντολογίας σε N-triples RDFs έγινε εκτέλεση των *name-triples.py*, *title-triples.py* στα οποία με αντικειμενοστραφή τρόπο ορίζουμε της κλάσεις Person και Movie, και τις συναρτήσεις του *data_prop()*, *object_prop()* αντίστοιχα.
- Εφόσον τα RDFs έχουν την μορφή

URI₁ *URI₂* *URI₃* (and *URI₄* σε περίπτωση N-quads).

αποφασίσαμε να ορίσουμε στους constructor *_init_()*, τα λεξικά **prefix**, **infix**, **postfix** τα οποία θα χρησιμοποιηθούν αργότερα για την αντιστοίχιση των πληροφοριών των .csv στα κατάλληλα *URI_i* s.

- Επιλέξαμε την **σειριοποίηση με N-triples**, που έχουν πιο απλή σύνταξη συγχριτικά με άλλους τρόπους σύνταξης, για την αναπαράσταση των RDFs μας. Εφόσον οι δηλώσεις του γράφου, που τελικά θα δημιουργηθεί μετά το loading των RDFs στο επόμενο βήμα, θεωρείται ότι αποτελούν ισχυρισμούς για την κατάσταση του κόσμου, και συνδυαστικά με το γεγονός ότι αποσκοπούμε σε εύκολο Linking με τρίτες βάσεις δεδομένων, επιλέξαμε τον συγκεκριμένο τρόπο σύνταξης.
- Για την κατασκευή των RDF triples χρησιμοποίησηκε το domain <http://www.example.com/> καθώς έχει καθιερωθεί για να χρησιμοποιείται για επεξηγηματικά παραδείγματα σε έγγραφα. με αυτόν τον τρόπο μπορούμε να το χρησιμοποιούμε χωρίς να ζητήσουμε άδεια.

• Μορφή RDFs

– DATA PROPERTIES

```
< https://www.imdb.com/title/tt0020297 >< http://www.example.com/adult >
"0"^^xsd:z.
```

– OBJECT PROPERTIES

```
< https://www.imdb.com/title/tt0000001 >< https://www.example.com/has-self ><
https://www.imdb.com/name/nm1588970 >
```

• Διαδικασία Μετατροπής

– Για Movie data properties [~ 900 MB]

1. Μετατροπή του **titles.ratings** σε 1 Dataframe (λόγω του μικρού πλήθους αυτών, άρα διαπιστώνουμε ότι για λίγες movies της βάσης της IMDB υπάρχει η αντίστοιχη πληροφορία για το review της) των ~ 10000 rows **συνολικά**
2. Καταχερματισμός του **titles.basics** σε Dataframes που περιλαμβάνει όσα rows απαιτούνται για να κάνει include όλα τα απαραίτητα RDFs για 10000 ταινίες, και επιλέξαμε τον καταχερματισμό αυτόν γιατί είχαμε να κάνουμε με > 590.000 instances movies. Το κάθε ένα από τα 596 basics Dataframe, το κάνουμε merge inner join με το παραπάνω review Dataframe.
3. Προσπελαύνουμε το merged Dataframe γραμμή γραμμή, ενώ για κάθε title imdb id δημιουργούμε τα κατάλληλα *prefix*, *infix*, *postfix*. Συγκεκριμένα για το **prefix** χρησιμοποιήσαμε το URI με βάση την IMDB, για το **infix** -και κατ'επέκταση τον ορισμό της ιδιότητας- το URI με βάση την example⁸⁹, και για το **postfix** την αντίστοιχη τιμή του πεδίου ως λεκτικό με συγκεκριμένο τύπο δεδομένων.
4. Στην συνέχεια γίνεται join των *prefix*, *infix*, *postfix* σε ένα μοναδικό string το οποίο γράφουμε στο κατάλληλο .nt αρχείο¹⁰ **Παράδειγμα φαίνεται παρακάτω**

```

1 <https://www.imdb.com/title/tt0260620> <http://www.example.com/type> "tvSeries"^^xsd:string.
2 <https://www.imdb.com/title/tt0260620> <http://www.example.com/prim_title> "Gourmet Express"^^xsd:string.
3 <https://www.imdb.com/title/tt0260620> <http://www.example.com/orig_title> "Gourmet Express"^^xsd:string.
4 <https://www.imdb.com/title/tt0260620> <http://www.example.com/adult> "0"^^xsd:boolean.
5 <https://www.imdb.com/title/tt0260620> <http://www.example.com/start_year> "2000"^^xsd:string.
6 <https://www.imdb.com/title/tt0260620> <http://www.example.com/rating> "8.0"^^xsd:double.
7 <https://www.imdb.com/title/tt0260620> <http://www.example.com/voting> "6"^^xsd:int.
8 <https://www.imdb.com/title/tt0260621> <http://www.example.com/type> "tvSeries"^^xsd:string.
9 <https://www.imdb.com/title/tt0260621> <http://www.example.com/prim_title> "Hodge Podge Lodge"^^xsd:string.
10 <https://www.imdb.com/title/tt0260621> <http://www.example.com/orig_title> "Hodge Podge Lodge"^^xsd:string.
11 <https://www.imdb.com/title/tt0260621> <http://www.example.com/adult> "0"^^xsd:boolean.
12 <https://www.imdb.com/title/tt0260621> <http://www.example.com/start_year> "1970"^^xsd:string.
13 <https://www.imdb.com/title/tt0260621> <http://www.example.com/end_year> "1977"^^xsd:string.
14 <https://www.imdb.com/title/tt0260621> <http://www.example.com/is_genre> <http://www.example.com/Family>.
15 <https://www.imdb.com/title/tt0260621> <http://www.example.com/rating> "8.7"^^xsd:double.
16 <https://www.imdb.com/title/tt0260621> <http://www.example.com/voting> "33"^^xsd:int.
17 <https://www.imdb.com/title/tt0260622> <http://www.example.com/type> "tvSeries"^^xsd:string.
18 <https://www.imdb.com/title/tt0260622> <http://www.example.com/prim_title> "Angel"^^xsd:string.
19 <https://www.imdb.com/title/tt0260622> <http://www.example.com/orig_title> "Hana no ko Lun Lun"^^xsd:string.

```

– Για Person data properties [~ 980 MB]

1. Καταχερματισμός του **name.basics** σε Dataframes που περιλαμβάνει όσα rows απαιτούνται για να κάνει include όλα τα απαραίτητα RDFs για 10000 ταινίες, και επιλέξαμε τον καταχερματισμό αυτόν γιατί είχαμε να κάνουμε με >941.000 instances movies.
2. Προσπελαύνουμε το κάθε ένα από τα 941 Dataframes γραμμή γραμμή, ενώ για κάθε name imdb id δημιουργούμε κατάλληλα *prefix*, *infix*, *postfix* για τα fields *name*, *birth_year*, *death_year* (αφορώντας για τα *primaryProfession*, *knownForTitles* που θα χρησιμοποιηθούν για τον ορισμό ρόλων)
3. Με την ίδια διαδικασία, τα κάνουμε join σε ένα μοναδικό string το οποίο γράφουμε στο κατάλληλο .nt αρχείο **Παράδειγμα φαίνεται παρακάτω**

⁸⁹Για την ονοματολογία : <<http://www.example.com/x>> όπου *x* = *adult*, *start_year*, ... τα πεδία σε πεζά χωρισμένα με κάτω παύλες, ακολουθούμε πιστά την ονοματολογία της οντολογίας OWL που ορίσαμε

⁹⁰TODΟ ΓΙΑΤΙ;

¹⁰Να σημειωθεί ότι με αφορμή του parsing του παρόντος .tsv αρχείου, ορίσαμε και τον ρόλο μεταξύ της κλάσης Movie και της κατάλληλης υποκλάσης Genre στην οποία ανήκει η Movie

```

1 <https://www.imdb.com/name/nm0000001> <http://www.example.com/name> "Fred Astaire"^^xsd:string.
2 <https://www.imdb.com/name/nm0000001> <http://www.example.com/birth_year> "1899"^^xsd:string.
3 <https://www.imdb.com/name/nm0000001> <http://www.example.com/death_year> "1987"^^xsd:string.
4 <https://www.imdb.com/name/nm0000002> <http://www.example.com/name> "Lauren Bacall"^^xsd:string.
5 <https://www.imdb.com/name/nm0000002> <http://www.example.com/birth_year> "1924"^^xsd:string.
6 <https://www.imdb.com/name/nm0000002> <http://www.example.com/death_year> "2014"^^xsd:string.
7 <https://www.imdb.com/name/nm0000003> <http://www.example.com/name> "Brigitte Bardot"^^xsd:string.
8 <https://www.imdb.com/name/nm0000003> <http://www.example.com/birth_year> "1934"^^xsd:string.
9 <https://www.imdb.com/name/nm0000004> <http://www.example.com/name> "John Belushi"^^xsd:string.
10 <https://www.imdb.com/name/nm0000004> <http://www.example.com/birth_year> "1949"^^xsd:string.
11 <https://www.imdb.com/name/nm0000004> <http://www.example.com/death_year> "1982"^^xsd:string.
12 <https://www.imdb.com/name/nm0000005> <http://www.example.com/name> "Ingmar Bergman"^^xsd:string.
13 <https://www.imdb.com/name/nm0000005> <http://www.example.com/birth_year> "1918"^^xsd:string.
14 <https://www.imdb.com/name/nm0000005> <http://www.example.com/death_year> "2007"^^xsd:string.
15 <https://www.imdb.com/name/nm0000006> <http://www.example.com/name> "Ingrid Bergman"^^xsd:string.
16 <https://www.imdb.com/name/nm0000006> <http://www.example.com/birth_year> "1915"^^xsd:string.
17 <https://www.imdb.com/name/nm0000006> <http://www.example.com/death_year> "1982"^^xsd:string.
18 <https://www.imdb.com/name/nm0000007> <http://www.example.com/name> "Humphrey Bogart"^^xsd:string.
19 <https://www.imdb.com/name/nm0000007> <http://www.example.com/birth_year> "1899"^^xsd:string.
20 <https://www.imdb.com/name/nm0000007> <http://www.example.com/death_year> "1957"^^xsd:string.
21 <https://www.imdb.com/name/nm0000008> <http://www.example.com/name> "Marlon Brando"^^xsd:string.
22 <https://www.imdb.com/name/nm0000008> <http://www.example.com/birth_year> "1924"^^xsd:string.
23 <https://www.imdb.com/name/nm0000008> <http://www.example.com/death_year> "2004"^^xsd:string.
24 <https://www.imdb.com/name/nm0000009> <http://www.example.com/name> "Richard Burton"^^xsd:string.
25 <https://www.imdb.com/name/nm0000009> <http://www.example.com/birth_year> "1925"^^xsd:string.
26 <https://www.imdb.com/name/nm0000009> <http://www.example.com/death_year> "1984"^^xsd:string.
27 <https://www.imdb.com/name/nm0000010> <http://www.example.com/name> "James Cagney"^^xsd:string.
28 <https://www.imdb.com/name/nm0000010> <http://www.example.com/birth_year> "1899"^^xsd:string.
29 <https://www.imdb.com/name/nm0000010> <http://www.example.com/death_year> "1986"^^xsd:string.
30 <https://www.imdb.com/name/nm0000011> <http://www.example.com/name> "Gary Cooper"^^xsd:string.
31 <https://www.imdb.com/name/nm0000011> <http://www.example.com/birth_year> "1901"^^xsd:string.
32 <https://www.imdb.com/name/nm0000011> <http://www.example.com/death_year> "1961"^^xsd:string.
33 <https://www.imdb.com/name/nm0000012> <http://www.example.com/name> "Bette Davis"^^xsd:string.
34 <https://www.imdb.com/name/nm0000012> <http://www.example.com/birth_year> "1908"^^xsd:string.
35 <https://www.imdb.com/name/nm0000012> <http://www.example.com/death_year> "1989"^^xsd:string.

```

– Για Movie/Person object properties [~ 2x 3.9 G B]

- Για κάθε Movie υπάρχουν από 1 έως X Person που εμπλέκονται με αυτήν με διάφορους ρόλους τους οποίους πρέπει να ορίσουμε. Για τους ρόλους ακολουθούμε την ίδια ακριβώς διαδικασία με προηγουμένως, συγκεκριμένα καταχερματίζοντας το **title.principals**.
- Αίζει να σημειωθεί ότι επειδή κάθε Movie εμπλέκεται με αρκετά Person (κατά μό 10), τα RDFs που θα προκύψουν είναι πολύ περισσότερα από τα Movies δηλαδή πολύ περισσότερα από 941000 instances. Μάλιστα, για τα RDF

`< Movie > < has_role > < Subclass of Person >`

προκύπτουν .nt αρχεία συνολικού μεγέθους 3.9 GB, ενώ με κατάλληλα queries SPARQL μπορούν να προκύψει πολύ έυκολα η αντίστροφη σχέση. Παρόλα αυτά για λόγους πληρότητας, δημουργήσαμε όλα τα RDF

`< Subclass of Person > < is_role_of > < Movie >`

και τα φορτώσαμε στο Semantic Web Base **Παράδειγμα φαίνεται παρακάτω**

δεδομένη στιγμή. Γενικά, θεωρούμε ότι ο κατακερματισμός γίνεται για λόγους ασφάλειας ειδικά σε περιπτώσεις που το running ενός script .py μπορεί να τερμάτιζε λόγω ανεπιθυμήτων παραγόντων

3 Virtuoso RDF Loading

Βήματα:

1. Ρυθμίσαμε στο αρχείο `database/virtuoso.ini` και αλλάξαμε την μεταβητή **DirsAllowed** ώστε να επιτρέπει το ανέβασμα RDFs από τους φακέλους `movie_data_properties`, `name_data_properties`, `name_object_properties`, `data_object_properties`, `data_object_properties_extra`, `name_object_properties_extra`, που περιέχουν τα DATA PROPERTIES και τα OBJECT PROPERTIES.
2. Για το ανέβασμα του πχ. `movie_data_properties`
 - (a) `ld_dir('../vad/movie_data_properties', '*.nt', 'movie_data')`
 - (b) `select * from DB.DBA.load_list;` Φαίνονται όλα τα αρχεία σε NULL και state = 0
 - (c) `rdf_loader_run();`
 - (d) `checkpoint;`
 - (e) `select * from DB.DBA.load_list;` Φαίνονται όλα τα αρχεία σε NULL και state = 2
3. Επανάληψη της διαδικασίας και για τους υπόλοιπους 5 φακέλους

Παράδειγμα φαίνεται παραχώτω

```
.../vad/name_object_properties_extra/name_rdf3418.nt          name_object_extra           2      2019.7.8 4:7.59 759722000 2019.7.8 4:8.0 63615000 0
.../vad/name_object_properties_extra/name_rdf3419.nt          name_object_extra           2      2019.7.8 4:7.59 759722000 2019.7.8 4:8.0 383739000 0
.../vad/name_object_properties_extra/name_rdf3420.nt          name_object_extra           2      2019.7.8 4:8.0 384421000 2019.7.8 4:8.0 881449000 0
.../vad/name_object_properties_extra/name_rdf3421.nt          name_object_extra           2      2019.7.8 4:8.0 384421000 2019.7.8 4:8.1 133209000 0
.../vad/name_object_properties_extra/name_rdf3422.nt          name_object_extra           2      2019.7.8 4:8.1 133673000 2019.7.8 4:8.1 466019000 0
.../vad/name_object_properties_extra/name_rdf3423.nt          name_object_extra           2      2019.7.8 4:8.1 133673000 2019.7.8 4:8.1 616279000 0
.../vad/name_object_properties_extra/name_rdf3424.nt          name_object_extra           2      2019.7.8 4:8.1 619148000 2019.7.8 4:8.1 945838000 0
.../vad/name_object_properties_extra/name_rdf3425.nt          name_object_extra           2      2019.7.8 4:8.1 619148000 2019.7.8 4:8.2 201620000 0
.../vad/name_object_properties_extra/name_rdf3426.nt          name_object_extra           2      2019.7.8 4:8.2 202387000 2019.7.8 4:8.2 472222000 0
.../vad/name_object_properties_extra/name_rdf3427.nt          name_object_extra           2      2019.7.8 4:8.2 202387000 2019.7.8 4:8.2 695757000 0
.../vad/name_object_properties_extra/name_rdf3428.nt          name_object_extra           2      2019.7.8 4:8.2 696376000 2019.7.8 4:8.2 988812000 0
.../vad/name_object_properties_extra/name_rdf3429.nt          name_object_extra           2      2019.7.8 4:8.2 696376000 2019.7.8 4:8.3 276505000 0
.../vad/name_object_properties_extra/name_rdf3430.nt          name_object_extra           2      2019.7.8 4:8.3 277894000 2019.7.8 4:8.3 479820000 0
NULL
```

4 SPARQL querries

1. 1o Ερώτημα

Έναν πίνακα που να περιέχει για κάθε είδος (genre) κινηματογραφικής ταινίας, το πλήθος των ταινιών αυτού του είδους που περιέχει η βάση του IMDB. Ο πίνακας θα πρέπει να είναι ταξινομημένος κατά φθίνουσα σειρά πλήθους ταινιών.

Αρχικά επιλέγουμε από όλες τις Movies εκείνες για τις οποίες ορίζεται ο ρόλος `is_genre` (για όλες ορίζεται τουλάχιστον μία φορά αφού κάθε ταινία ή σειρά ανήκει σε κάποιο είδος, ενώ πολλές έχουν παραπάνω από έναν). Στην συνέχεια, ομαδοποιούμε τις Movies αυτές με κριτήριο το είδος που ανήκουν κάνοντας χρήση της εντολής `GROUP BY`, και εντός κάθε ομάδας μετράμε πόσες υπάρχουν στην συγκεκριμένη ομάδα κάνοντας χρήση της εντολής `COUNT()`

QUERY 1	
PREFIX ex: <http://www.example.com/>	
SELECT DISTINCT ?film_genre , (COUNT(?movie_id) AS ?number_of_movies)	
WHERE{	
?movie_id ex:is_genre ?genre.	
BIND (STRAFTER(STR(?genre), "http://www.example.com/") AS ?film_genre)	
}	
GROUP BY ?film_genre	
ORDER BY DESC(?number_of_movies)	

film_genre	number_of_movies
Drama	1602779
Comedy	1272889
Short	796776
Documentary	583472
Talk-Show	560879
Romance	551313
Family	420269
News	392331
Animation	303115
Reality-TV	293672
Music	277412
Crime	260373
Action	256002
Adventure	213961
Game-Show	196978
Adult	172646
Fantasy	126471
Sport	125062
Mystery	122867
Horror	109706
Thriller	106056
Sci-Fi	83704
History	82978
Biography	71035
Musical	47156
Western	26154
War	24461
Film-Noir	769

2. 2o Ερώτημα

Έναν πίνακα που να περιέχει τις εταιρείες διανομής (dbo:distributor) των τίτλων της IMDB που έχουν βαθμολογηθεί από περισσότερους από 10000 χρήστες με μέση βαθμολογία τουλάχιστον 7 και για τους οποίους υπάρχει η σχετική πληροφορία στην DBpedia. Κάθε εταιρεία διανομής θα πρέπει να συνοδεύεται από το πλήθος των ταινιών που διένειμε και ο πίνακας θα πρέπει να είναι ταξινομημένος κατά φιλίουσα σειρά πλήθους ταινιών.

Στο ερώτημα αυτό, αρχικά επιλέγουμε όλες τις Movies εκείνες για τις οποίες ορίζονται οι ρόλοι **rating**, **voting** (για λίγες μόνο Movies σε σχέση με το σύνολο υπάρχει πληροφορία και για τους 2 ρόλους), και από αυτές με κατάλληλη χρήση της εντολής **FILTER()** κρατάμε μόνο όσες έχουν rating τουλάχιστον 7 και αριθμό ψήφων τουλάχιστον 100000. Τέλος, τελικά αφού για αυτές τις ταινίες κρατήσουμε το από όλο το URI του imdb id μόνο το αριθμητικό μέρος - με κατάλληλη χρήση της **SUBSTR()** - , δηλαδή πχ. από <<https://www.imdb.com/title/tt9916578>> κρατάμε μόνο το 9916578, για ταυτοποίηση των ταινιών αυτών (αν υπάρχουν) στην βάση δεδομένων της **DBpedia**. Ετσι λοιπόν καλώντας την υπηρεσία της **DBpedia**, βρίσκουμε μέσω του ρόλου **imdbId** το αντίστοιχο Movie resource (αν υπάρχει όπως είπαμε) και μέσω αυτού και του ρόλου **distributor**, βρίσκουμε τους distributors των Movies αυτών, ενώ με χρήση της **SUBSTR()** κρατάμε το μέρος του distributor resource που μας ενδιαφέρει, δηλαδή το όνομα του. Τέλος,

ομαδοποιούμε τις Movies αυτές με κριτήριο τον διανομέα της ταινίας κάνοντας χρήση της εντολής *GROUP BY*, και εντός κάθε ομάδας μετράμε πόσες ταινίες έχουν διανεμηθεί από τον συγκεκριμένο διανομέα κάνοντας χρήση της εντολής *COUNT()*

QUERY 2
<pre> PREFIX ex:<http://www.example.com/> PREFIX dbo: <http://dbpedia.org/ontology/> SELECT ?distributor_name, (COUNT(?imdb_id) AS ?number_of_distributor_movies) WHERE{ ?movie_id ex:rating ?rating. ?movie_id ex:voting ?voting. BIND(STR(?voting) AS ?new_voting) FILTER(?rating >= 7.0) FILTER(?new_voting >= 100000) BIND (SUBSTR(STR(?movie_id),30,8) AS ?tmp) BIND (SUBSTR(STR(?tmp),1,8) AS ?imdb_id) SERVICE <http://dbpedia.org/sparql> { ?movie dbo:imdbId ?imdb_id; dbo:distributor ?distributor. BIND(STRAFTER(STR(?distributor),"resource/") AS ?distributor_name) } } GROUP BY (?distributor_name) ORDER BY DESC(?number_of_distributor_movies) </pre>

distributor_name	number_of_distributor_movies
"Warner_Bros."	8
"Columbia_Pictures"	7
"Paramount_Pictures"	5
"Universal_Studios"	4
"Walt_Disney_Studios_Motion_Pictures"	4
"20th_Century_Fox"	4
"Warner_Bros._Pictures"	3
"Universal_Pictures"	3
"Metro-Goldwyn-Mayer"	2
"DreamWorks"	2
"Warner_Bros._Television_Distribution"	2
"20th_Television"	2
"The_Weinstein_Company"	2
"NBCUniversal_Television_Distribution"	2
"Canal+"	1
"Entertainment_Film_Distributors"	1
"Sonar_Entertainment"	1
"Netflix"	1
"Touchstone_Pictures"	1
"Savoy_Pictures"	1
"New_Line_Cinema"	1
"Gramercy_Pictures"	1
"Screen_Gems"	1
"Roadshow_Entertainment"	1
"Joseph_Burstyn"	1
"Momentum_Pictures"	1
"Toei_Company"	1
"FremantleMedia"	1
"Endemol_UK"	1
"Metro-Goldwyn_Mayer"	1
"Sony_Pictures_Classics"	1
"Orion_Pictures"	1
"Bryanston_Distributing_Company"	1
"Fox_Searchlight_Pictures"	1
"United_Artists"	1
"Buena_Vista_International"	1
"Warner_Independent_Pictures"	1
"Miramax_Films"	1
"Toho"	1

3. 3ο Ερώτημα

Έναν πίνακα που να περιέχει τα ονόματα των χωρών γέννησης των σκηνοθετών των τίτλων της IMDB που έχουν βαθμολογήθει από περισσότερους από 200000 χρήστες με μέση βαθμολογία τουλάχιστον 8,5 και για τους οποίους υπάρχει η σχετική πληροφορία στην Wikidata. Κάθε όνομα χώρας θα πρέπει να συνοδεύεται από το πλήθος των αντίστοιχων ταινιών και ο πίνακας θα πρέπει να ειναι ταξινομημένος κατά φυλίουσα σειρά πλήθους ταινιών.

Με παρόμοια λογική όπως στο Ερώτημα 2ο, κάνουμε ένα πρώτο φιλτράρισμα και κρατάμε τα imdb id των Movies για τις οποίες υπάρχει η πληροφορίας του rating και του voting και ξεπερνάει το κατώφλι 8.5 και 200000 αντίστοιχα. Για την ταυτοποίηση εδώ πέρα των πόρων με την Wikidata δεν απαιτείται μόνο το αριθμητικό μέρος του id αλλά ολόχληρο, δηλαδή πχ. από <<https://www.imdb.com/title/tt9916578>> κρατάμε μόνο το tt9916578. Έτσι λοιπόν καλώντας την υπηρεσία της **Wikidata**, βρίσκουμε μέσω του ρόλου **P31**, **P:279**, **P:345** βρίσκουμε όσα Movies υπάρχουν στην βάση δεδομένων της **Wikidata** και τα οποία έχουν την ιδιότητα **Q539826**, **Q11424** δηλαδή είναι ταινίες ή σειρές (προφανώς τελικά όλες όσες υπάρχουν πράγματι στην βάση). Για όλες αυτές μέσω του ρόλου **P:57**, βρίσκουμε τους σκηνοθέτες της ταινίας, μέσω του ρόλου **P:19** για τον κάθε σκηνοθέτη την πόλη που γεννήθηκε, ενώ μέσω του ρόλου **P:17** για την κάθε πολή την χώρα που αυτή ανήκει. Τώρα για την κάθε χώρα - και εφόσον η localhost εξαγωγή δεδομένων δεν επιτρέπει την χρήση της wikibase - εκμεταλλευόμαστε την γενική βάση δεδομένων w3, και μέσω του ρόλου **label** και συνοδευτικό φιλτράρισμα για αγγλική γλώσσα μόνο (για αποφυγή διπλότυπων) μέσω της χρήση της **FILTER()** και **LANG()**, βρίσκουμε την κατάλλη ετικέτα για το resource της χώρας που διαθέταμε. Τέλος, ομαδοποιούμε τις Movies αυτές με χρήσιμο το label αυτό της χώρας προέλευσης του σκηνοθέτη κάνοντας χρήση της εντολής **GROUP BY**, και εντός κάθε ομάδας μετράμε πόσες ταινίες έχουν σκηνοθετήσει οι σκηνοθέτες της συγκεκριμένης χώρας κάνοντας χρήση της εντολής **COUNT()**.

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ex:<http://www.example.com/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX p: <http://www.wikidata.org/prop/>

SELECT DISTINCT (STR(?label) AS ?country_of_directors), (COUNT(?imdb_id) AS ?number_of_director_movies)
WHERE{
?movie_id ex:rating ?rating.
?movie_id ex:voting ?voting.

BIND(STR(?voting) AS ?new_voting)

FILTER( ?rating >= 8.5 )
FILTER( ?new_voting >= 200000 )

BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id)

SERVICE <https://query.wikidata.org/sparql> {
VALUES ?type {wd:Q5398426 wd:Q11424}
?movie wdt:P31/wdt:P279* ?type;
       wdt:P345 ?imdb_id;
       wdt:P57 ?director.

?director wdt:P106 wd:Q2526255;
          wdt:P19 ?city.

?city wdt:P17 ?country .

?country rdfs:label ?label .
FILTER (LANG(?label) = "en")

}

GROUP BY (?label)
ORDER BY DESC(?number_of_director_movies)

```

country_of_directors	number_of_director_movies
United States of America	96
United Kingdom	35
Canada	12
Italy	5
France	5
Hungary	3
Japan	3
Brazil	3
Sweden	3
New Zealand	2
Australia	2
Belgium	2
Denmark	2
Argentina	1
Czech Republic	1

4. *4o Ερώτημα*

Έναν πίνακα όπου να φαίνονται ποιοι τίτλοι της IMDB που έχουν βαθμολογηθεί από περισσότερους από 10000 χρήστες δεν περιλαμβάνονται στην βάση ούτε της DBpedia ούτε της Wikidata.

5 Σημασιολογικός εμπλουτισμός βάσης

1. 1o Ερώτημα

To query **black and white movies** χρησιμοποιείται για την εύρεση των ασπρόμαυρων ταινιών. Συγχειριμένα, αρχικά κρατάμε τις ταινίες που έχουν πληροφορία χρώματος και στην συνέχεια μέσω ενός του rdfs πάρουμε το καταλληλότερο αντίστοιχο label.

```
BLACK AND WHITE MOVIES

PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX ex:<http://www.example.com/>

SELECT DISTINCT ?movie_id, (STR(?label) AS ?movieColor)
WHERE{
    ?movie_id ex:rating ?rating.
    ?movie_id ex:voting ?voting.
    FILTER( ?rating >= 8.0 and ?voting > 100000)
    BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id)

    SERVICE <https://query.wikidata.org/sparql>{
        VALUES ?type {wd:Q5398426 wd:Q11424}
        ?movie wdt:P345 ?imdb_id;
               wdt:P462 ?color.

        ?color rdfs:label ?label.
        FILTER (LANG(?label) = "en")
        FILTER (STR(?label) = 'black-and-white')

    }
}
```

Επειδή πρόκεται για μεγάλο πλήθους ταινιών, για τους σχοπούς του παρόντος project και για την εξαγωγή κάποιων ενδεικτικών αποτελεσμάτων, αντί για `?movie_id ?x ?y`, κρατάμε μόνο όσες ταινίες έχουν `rating > 8.0` και `rating > 100000`, ενώ τα αποτελέσματα φαίνονται παρακάτω:

movie_id	movieColor
https://www.imdb.com/title/tt0113247	black-and-white
https://www.imdb.com/title/tt0032553	black-and-white
https://www.imdb.com/title/tt0032976	black-and-white
https://www.imdb.com/title/tt0033870	black-and-white
https://www.imdb.com/title/tt0036775	black-and-white
https://www.imdb.com/title/tt0040522	black-and-white
https://www.imdb.com/title/tt0338564	black-and-white
https://www.imdb.com/title/tt0079522	black-and-white
https://www.imdb.com/title/tt0080678	black-and-white
https://www.imdb.com/title/tt0017136	black-and-white
https://www.imdb.com/title/tt0050976	black-and-white
https://www.imdb.com/title/tt0053604	black-and-white
https://www.imdb.com/title/tt0021749	black-and-white
https://www.imdb.com/title/tt0022100	black-and-white
https://www.imdb.com/title/tt0027977	black-and-white
https://www.imdb.com/title/tt0310793	black-and-white
https://www.imdb.com/title/tt0040897	black-and-white
https://www.imdb.com/title/tt0041959	black-and-white
https://www.imdb.com/title/tt0042192	black-and-white
https://www.imdb.com/title/tt0042876	black-and-white
https://www.imdb.com/title/tt0043014	black-and-white
https://www.imdb.com/title/tt0044079	black-and-white
https://www.imdb.com/title/tt0046250	black-and-white
https://www.imdb.com/title/tt0047296	black-and-white
https://www.imdb.com/title/tt0050825	black-and-white
https://www.imdb.com/title/tt0033467	black-and-white
https://www.imdb.com/title/tt0034583	black-and-white
https://www.imdb.com/title/tt0038650	black-and-white
https://www.imdb.com/title/tt0081398	black-and-white
https://www.imdb.com/title/tt0053291	black-and-white
https://www.imdb.com/title/tt0054215	black-and-white
https://www.imdb.com/title/tt0057012	black-and-white
https://www.imdb.com/title/tt0108052	black-and-white
https://www.imdb.com/title/tt0378194	black-and-white
https://www.imdb.com/title/tt0381061	black-and-white
https://www.imdb.com/title/tt0401792	black-and-white
https://www.imdb.com/title/tt0047478	black-and-white
https://www.imdb.com/title/tt0050083	black-and-white

2o Ερώτημα

To query **just comedy** χρησιμοποιείται για την εύρεση των ταινιών που ανήκουν μόνο στην κατηγορία των κωμωδιών και σε καμία άλλη κατηγορία. Συνεπώς μετράμε το πλήθος των κατηγοριών στις οποίες ανήκουν και κρατάμε τις ταινίες με μοναδικό πλήθος. Στην συνέχεια από αυτές διατηρούμε μόνο όσες ανήκουν στο είδος κωμωδία.

JUST COMEDY
<pre> PREFIX ex:<http://www.example.com/> PREFIX wd: <http://www.wikidata.org/entity/> SELECT DISTINCT ?movie_id WHERE{ ?movie_id ex:is_genre ?gernes. FILTER((COUNT(?gernes)) == 1) BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id) SERVICE <https://query.wikidata.org/sparql>{ VALUES ?type {wd:Q40831} ?imdb_id ?y ?type. } } </pre>

3. 3ο Ερώτημα

To query **comedy writers** χρησιμοποιείται για την εύρεση των συγγραφέων κωμωδιών. Συγκεκριμένα διατηρούμε όσες ταινίες είναι κωμωδίες και από αυτές όταν διαθέτουν την πληροφορία του συγγραφέα τότε επιστρέφουμε το id του.

COMEDY WRITERS
<pre> PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/> PREFIX ex:<http://www.example.com/> SELECT DISTINCT ?movie_id, (STR(?label) AS ?movieWriter) WHERE{ ?movie_id ex:rating ?rating. ?movie_id ex:voting ?voting. FILTER(?rating >= 9.0 and ?voting > 100000) BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id) SERVICE <https://query.wikidata.org/sparql>{ VALUES ?type {wd:Q5398426 wd:Q11424} ?movie wdt:P345 ?imdb_id; wdt:P58 ?writer. ?writer rdfs:label ?label. FILTER (LANG(?label) = "en") } } </pre>

Επειδή πρόκεται για μεγάλο πλήθους ταινιών, για τους σκοπούς του παρόντος project και για την εξαγωγή κάποιων ενδεικτικών αποτελεσμάτων, αντί για ?movie_id ?x ?y , κρατάμε μόνο όσες ταινίες έχουν rating > 9.0 και rating > 100000, ενώ τα αποτελέσματα φαίνονται παρακάτω:

movie_id	movieWriter
https://www.imdb.com/title/tt0417299	Bryan Konietzko
https://www.imdb.com/title/tt0417299	Michael Dante DiMartino
https://www.imdb.com/title/tt0417299	Aaron Ehasz
https://www.imdb.com/title/tt0417299	Tim Hedrick
https://www.imdb.com/title/tt2301451	Moira Walley-Beckett
https://www.imdb.com/title/tt4283088	D. B. Weiss
https://www.imdb.com/title/tt4283088	David Benioff
https://www.imdb.com/title/tt4283094	D. B. Weiss
https://www.imdb.com/title/tt4283094	David Benioff
https://www.imdb.com/title/tt0795176	David Attenborough
https://www.imdb.com/title/tt2356777	Nic Pizzolatto
https://www.imdb.com/title/tt0468569	Christopher Nolan
https://www.imdb.com/title/tt0468569	Bob Kane
https://www.imdb.com/title/tt0468569	Jonathan Nolan
https://www.imdb.com/title/tt0468569	David S. Goyer
https://www.imdb.com/title/tt0468569	Bill Finger
https://www.imdb.com/title/tt7366338	Craig Mazin
https://www.imdb.com/title/tt0068646	Francis Ford Coppola
https://www.imdb.com/title/tt0068646	Mario Puzo
https://www.imdb.com/title/tt0903747	Vince Gilligan
https://www.imdb.com/title/tt0903747	George Mastras
https://www.imdb.com/title/tt0903747	Peter Gould
https://www.imdb.com/title/tt0903747	Sam Catlin
https://www.imdb.com/title/tt0903747	John Shiban
https://www.imdb.com/title/tt0903747	Thomas Schnauz
https://www.imdb.com/title/tt0944947	George R. R. Martin
https://www.imdb.com/title/tt0944947	Jane Espenson
https://www.imdb.com/title/tt0944947	David Benioff
https://www.imdb.com/title/tt0944947	Dave Hill
https://www.imdb.com/title/tt0944947	D. B. Weiss
https://www.imdb.com/title/tt0944947	Bryan Cogman
https://www.imdb.com/title/tt0944947	Vanessa Taylor
https://www.imdb.com/title/tt0071562	Francis Ford Coppola
https://www.imdb.com/title/tt0071562	Mario Puzo
https://www.imdb.com/title/tt0111161	Stephen King
https://www.imdb.com/title/tt0111161	Frank Darabont
https://www.imdb.com/title/tt0185906	Tom Hanks
https://www.imdb.com/title/tt0185906	Stephen E. Ambrose
https://www.imdb.com/title/tt0303461	Jane Espenson
https://www.imdb.com/title/tt0303461	Joss Whedon
https://www.imdb.com/title/tt0303461	Tim Minear
https://www.imdb.com/title/tt0303461	Jose Molina
https://www.imdb.com/title/tt0303461	Drew Z. Greenberg
https://www.imdb.com/title/tt0303461	Cheryl Cain
https://www.imdb.com/title/tt0303461	Ben Edlund
https://www.imdb.com/title/tt0303461	Brett Matthews

4. 4ο Ερώτημα

To query **dramatic actors** χρησιμοποιείται για την εύρεση των ηθοποιών που παίζουν σε δραματικές ταινίες. Συνεπώς αν μία ταινία είναι δραματική και περιέχει πληροφορία για τους ηθοποιούς τους, τότε επιστρέφουμε το id του ηθοποιού

DRAMATIC ACTORS
<pre> PREFIX wd: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/> PREFIX ex:<http://www.example.com/> SELECT DISTINCT ?movie_id,(STR(?label) AS ?dramaticActor) WHERE{ ?movie_id ex:rating ?rating. ?movie_id ex:voting ?voting. FILTER(?rating >= 9.0 and ?voting > 100000) BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id) SERVICE <https://query.wikidata.org/sparql>{ VALUES ?type {wd:Q5398426 wd:Q11424} ?movie wd:P345 ?imdb_id; wd:P161 ?actor. ?actor wd:P106 wd:Q33999; wd:P19 ?dramatic_actors. ?dramatic_actors rdfs:label ?label. FILTER (LANG(?label) = "en") } } </pre>

Επειδή πρόκεται για μεγάλο πλήθους ταινιών, για τους σκοπούς του παρόντος project και για την εξαγωγή κάποιων ενδεικτικών αποτελεσμάτων, αντί για ?movie_id ?x ?y , χρατάμε μόνο όσες ταινίες έχουν rating > 9.0 και rating > 100000, ενώ τα αποτελέσματα φαίνονται παρακάτω:

movie_id	dramaticActor
https://www.imdb.com/title/tt5813916	Karşıyaka
https://www.imdb.com/title/tt1475582	London
https://www.imdb.com/title/tt1475582	Dublin
https://www.imdb.com/title/tt1475582	Bradford
https://www.imdb.com/title/tt1475582	Norwich
https://www.imdb.com/title/tt1475582	Southend-on-Sea
https://www.imdb.com/title/tt1475582	Hammersmith
https://www.imdb.com/title/tt1475582	Aldershot
https://www.imdb.com/title/tt1475582	Bozeat
https://www.imdb.com/title/tt1475582	Weston-super-Mare
https://www.imdb.com/title/tt1475582	Sedgefield
https://www.imdb.com/title/tt0468569	England
https://www.imdb.com/title/tt0468569	New York City
https://www.imdb.com/title/tt0468569	London
https://www.imdb.com/title/tt0468569	Singapore
https://www.imdb.com/title/tt0468569	Perth
https://www.imdb.com/title/tt0468569	Nicosia
https://www.imdb.com/title/tt0468569	Memphis
https://www.imdb.com/title/tt0468569	Vancouver
https://www.imdb.com/title/tt0468569	Cupertino
https://www.imdb.com/title/tt0468569	Huntington
https://www.imdb.com/title/tt0468569	Compton
https://www.imdb.com/title/tt0468569	Biloxi
https://www.imdb.com/title/tt0468569	Framingham
https://www.imdb.com/title/tt0468569	Haverfordwest
https://www.imdb.com/title/tt0468569	Leigh
https://www.imdb.com/title/tt0468569	Douglas
https://www.imdb.com/title/tt0468569	Edmonton
https://www.imdb.com/title/tt0468569	West Roxbury
https://www.imdb.com/title/tt0468569	Mitchel Air Force Base
https://www.imdb.com/title/tt0468569	Upper Clapton
https://www.imdb.com/title/tt7366338	London
https://www.imdb.com/title/tt7366338	United Kingdom
https://www.imdb.com/title/tt7366338	Stockholm

Το query **europen movies** χρησιμοποιείται για την εύρεση των ταινιών που έχουν γυριστεί εντός Ελλάδας. Συγκεκριμένα, διατηρούμε τις ταινίες που διαθέτουν πληροφορία για την ήπειρο στην οποία έχουν γυριστεί και στην συνέχεια τροποποιούν την ήπειρο αυτή με την Ευρώπη. Έτσι το query επιστρέφει τα id των ανίστοιχων ταινιών.

```
PREFIX wd: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

SELECT DISTINCT (STR(?label) AS ?european_movies)
WHERE{

    ?movie_id ?y >z
    BIND ( SUBSTR(STR(?movie_id), 28, 9) AS ?imdb_id)

    SERVICE <https://query.wikidata.org/sparql>{
        VALUES ?type {wd: Q5107}
        ?imdb_id wdt:P31/wdt:P279* ?type;
                  wdt: P345 ?continent.

        ?continent wdt:P106 wd:046;
                   wdt:P19 ?european_movies.

        ?european_movies refs:label ?label
    }
}
```

6. *6o Ερώτημα*

Το query **english language** χρησιμοποιείται για την εύρεση των αγγλόφωνων ταινιών. Οπότε αρχικά διατηρούμε τις ταινίες που διατίθεται στην πληροφορία για την γλώσσα και στην συνέχεια από αυτές επιλέγουμε αυτές που η γλώσσα τους είναι η αγγλική.

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX ex:<http://www.example.com/>

SELECT DISTINCT ?movie_id,(STR(?label) AS ?englishLanguage)
WHERE{
    ?movie_id ex:rating ?rating.
    ?movie_id ex:voting ?voting.
    FILTER( ?rating >= 9.0 and ?voting > 100000)
    BIND (SUBSTR(STR(?movie_id),28,9) AS ?imdb_id)

    SERVICE <https://query.wikidata.org/sparql>{
        VALUES ?type {wd:Q5398426 wd:Q11424}
        ?movie wdt:P345 ?imdb_id;
               wdt:P364 ?language.

        ?language wdt:P106 wd:Q1860.

        ?language rdfs:label ?label
        FILTER (LANG(?label) = "en")
    }
}
```

7. *7o Ερώτημα*

Το query **movies of 90's** χρησιμοποιείται για την εύρεση των πρωτοπροβλήματων στην δεκαετία του 90. Συγκεκριμένα αρχικά επιλέγουμε τις ταινίες για τις οποίες διανέτουμε πληροφορία για το έτος κυκλοφορίας του. Και στην συνέχεια επιλέγουμε αυτές για τις οποίες το έτος ανήκει στο range 1990 - 2000

```

MOVIES OF '90s

PREFIX wdt: <http://www.wikidata.org/prop/direct/>

SELECT DISTINCT ?movie_id
WHERE{
    ?movie_id ?y ?z
    BIND (SUBSTR(STR(?movie_id), 28, 9) AS ?imdb_id)

    SERVICE <https://query.wikidata.org/sparql>{
        ?imdb_id wdt:P577 ?year.
        FILTER(?year >= 1990)
        FILTER(?year <= 2000)
    }
}

```

8. 8ο Ερώτημα

Το query **greek subtitles** χρησιμοποιείται για την εύρεση των ταινιών που διαθέτουν ελληνικούς υπότιτλους. Οπότε όπως και στις προηγούμενες περιπτώσεις αρχικά διατηρούμε τις ταινίες που διαθέτουν πληροφορία για τους υπότιτλους και στην συνέχεια όσες από αυτές έχουν υπότιτλους στα ελληνικά.

```

GREEK SUBTITLES

PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>

SELECT DISTINCT (STR(?label) AS ?greek_lang)
WHERE{
    ?movie_id ?y ?z
    BIND (SUBSTR(STR(?movie_id), 28, 9) AS ?imdb_id)

    SERVICE <https://query.wikidata.org/sparql>{
        VALUES ?type {wd: Q204028}
        ?imdb_id wdt:P31/wdt:P279* ?type;
                  wdt: P345 ?language.

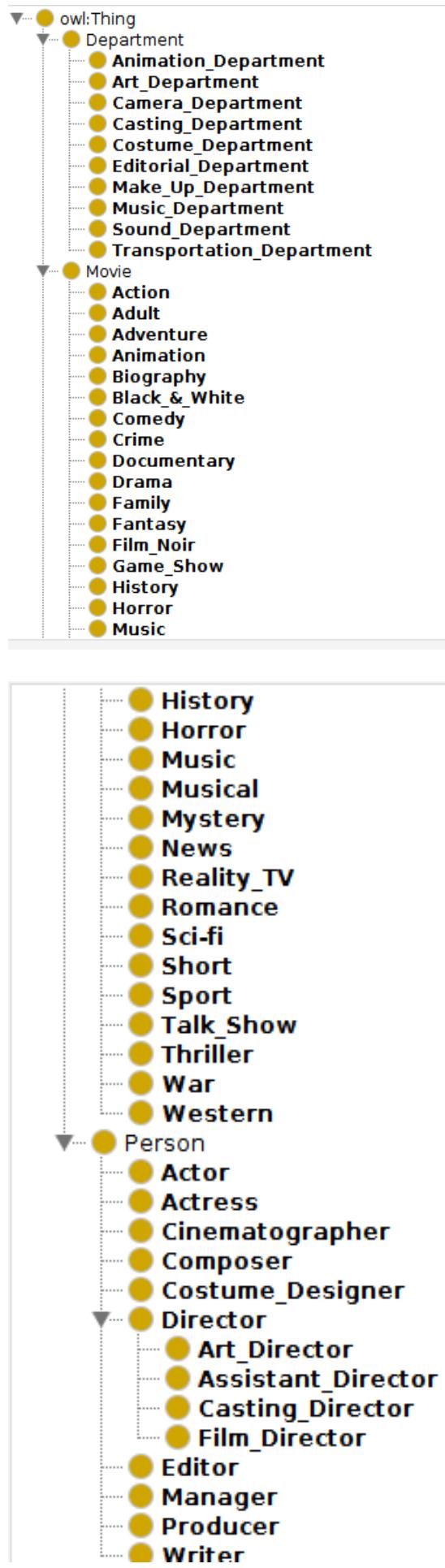
        ?language wdt:P106 wd:Q9129;
                  wdt:P19 ?greek_lang.

        ?greek_lang refs:label ?label
    }
}

```

Συμπεράσματα:

- Όμως μπορούμε να συμπεράνουμε από τα παραπάνω queries που κατασκευάσαμε όπως από πολλά ακόμα που θα μπορούσαμε να κατασκευάσουμε, η επέκταση μιας τοπικής βάσης δεδομένων με άλλες μεγάλεις βάσεις δεδομένων είναι ιδιαίτερα σημαντική.
- Συγκεκριμένα μέσα την κατασκευή των RDFs καταλάβαμε πόσο κοστίζει η αποθήκευση ακόμα και μιας μικρής σχετικά semantic web base. **Συνεπώς** η διασύνδεση της με άλλες τεράστιες βάσεις του ιστού είναι ιδιαίτερα σημαντική για την αντιμετώπιση ερωτημάτων σχετικά με λεπτομέρειες που δυσκολευόμαστε να προσθέσουμε στις τοπικές βάσεις.
- Από τα προηγούμενα διαφαίνονται πόσο αναγκαία είναι η διασύνδεση μιας τοπικής βάσης που κατασκευάζουμε με τις μεγαλύτερες βάσεις δεδομένων όπως στην περίπτωση μας είναι οι **wikidata** και **dbpedia**.
- Τα αντίστοιχα αξιώματα στην Protege μετά την επέκταση της βάσης θα είναι:





6 Προβλήματα

- **Για 1ο Βήμα - Οντολογία**

- Για την κατασκευή της οντολογίας στο Protege δεν υπάρχει σχεδιαστικά κάποιος τρόπος για την εισαγωγή δεδομένων μέσω κάποιου Script με αποτέλεσμα η εισαγωγή *individuals* και η απονομή ρόλων και ιδιοτήτων σε αυτά να αποτελεί ιδιαίτερα χρονοβόρα διαδικασία, για αυτό απλά προστέθηκαν κάποια ενδεικτικά.
- Θέλοντας να δώσουμε μια πιο απτή εικόνα της οντολογίας που κατασκευάσαμε προσθέσαμε κάποια plug-ins στο Virtuoso προκειμένου να μπορέσουμε αυτόματα να εξάγουμε ένα EER (Enhanced Entity Relational) Model όχι όμως επιτυχώς, με αποτέλεσμα να χρειαστεί να σχεδιάσουμε το EER μόνοι μας σε ένα σχεδιαστικό πρόγραμμα το draw.io

- **Για 2ο Βήμα - Virutoso & RDFs**

- Στην ονοματοδοσία των πεδίων *primaryTitle*, *originalTitle* των Movies σε ορισμένες περιπτώσεις εμπειριέχονταν διπλά "", και άρα η δήλωση της ιδιότητας τους οδηγούσε σε RDF της μορφής πχ "*The good "parent"*" :xsd:string., το οποίο προκαλούσε σε error όταν καλούσαμε την loader στο virtuoso για φόρτωση των αχείων .nt στην Semantic Web Base. Γενικά έπρεπε να γίνει σωστή διαχείριση όλων των syntax errors
- Αρχικά πειραματιστήκαμε αρκετά με την εισαγωγή των RFDs στην αποθήκη με αποτέλεσμα να έχουμε αποθηκεύσει σε αυτήν αχρείαστους γράφους τους οποίους έπρεπε στην συνέχεια να σβήσουμε ώστε να μην αντιμετωπίσουμε πρόβλημα με το πλήθος των δεδομένων της βάσης, με κατάλληλη χρήση της εντολής SPARQL CLEAR GRAPH <graph-name>;

- **Για 3ο Βήμα - SPARQL queries**

- Ο τρόπος που είναι αποθηκευμένα τα δεδομένα στα **Wikidata**, **Dbpedia** δημιουργησε αρχετά προβλήματα κατά την διασύνδεση της τοπικής μας Semantic Web Base με τις δύο αυτές βάσεις.
- Η έλλειψη δεδομένων της DBpedia για πολλές ταινίες είχε εγείρει πολλές απορίες , με αποτέλεσμα να αναφωτιόμαστε για την προβληματικότητα των queries μας

7 Αναφορές

1. Αναπαράσταση οντολογογικής γνώσης και συλλογιστικής, Γ. Στάμου, διαθέσιμο στο αποθετήριο "Κάλλιπος" [<https://repository.kallipos.gr/handle/11419/4225>]
2. Εισαγωγή στον Σημασιολογικό Ιστό, G. Antoniou, F. van Harmelen, Κλειδάριθμος, 2008
3. A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3, Matthew Horridge, The University Of Manchester, March 24, 2011
4. Mapping OWL to the Entity Relationship and Extended Entity Relationship models, Shikha Bagui, Int. J. Knowledge and Web Intelligence, Vol. 1, Nos. 1/2, 2009
5. OpenLink Virtuoso Universal Server Documentation, 1999-2019 OpenLink Software
6. https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial
7. <http://sites.linkeddata.center/help/devop/examples/sparql-examples>
8. <https://www.w3.org/TR/rdf-schema/>