Alma Mater Studiorum
Università di Bologna
Sede di Cesena

Elaborato di Ontologia di Trasporto

Fabián Andrés Aspée Encina

Professoressa : **Antonella Carbonaro**
Corso :**Web Semantico 19/20**

Dicembre 2020

Indice

Lista di Figure

1 Introduzione

Ogni giorno in Italia circa 16,3 milioni di persone utilizzano il trasporto pubblico, che equivale a circa 5 miliardi all'anno [1], se pensiamo che L'Italia rappresenta solo lo 0,7% della popolazione a livello mondiale possiamo capire che l'area di trasporto è molto attraente dal punto di vista di sviluppo di nuovi modelli per poter ottenere benefici su di loro.

Attualmente i diversi mezzi di trasporto pubblici esistenti principali sono:

- Autobus: possono essere sia:
  - Urbani: ci permette di muoverci dentro la città.
  - Extraurbano: ci permette di muoverci tra città e città.
- Metro: ci permette di muoverci dentro la città.
- Treno: ci permette di muoverci da città a città o da paese a paese (caso di Europa).
- Aereo: ci permette di muoverci da paese a paese oppure da città a città.

Se consideriamo solo queste tipologie di trasporto e pensiamo alla quantità di persone che fanno uso di questi mezzi, possiamo comunque pensare alla necessità di avere un modello che permetta ai passeggeri di scegliere qual'è il migliore modo per andare da un punto ad un altro tenendo in considerazione diversi aspetti.

Questo lavoro si focalizza principalmente su una ontologia sviluppata per permettere ad un passeggero di poter scegliere il modo più comodo secondo le sue necessità, tenendo in considerazione diversi aspetti, come:

- Modo di trasporto: metro, autobus, treno, ecc.
- Combinazioni: deve cambiare mezzo di trasporto per arrivare a destinazione.
- Servizi presenti nel tragitto: ristoranti, librerie, ecc.

2 Descrizione Dominio e Ontologia Proposta

2.1 Descrizione Dominio

Come è stato inizialmente descritto nella sezione 1, i mezzi di trasporto pubblici sono molto importanti nelle economie dei paesi (sia quelli in via di sviluppo sia quelli sviluppati), in più c'è una forte enfasi nel cambiare il modo in cui questi operano, riferito al tipo di combustibile che utilizzano. Ci sono diversi studi associati a diversi mezzi di trasporto che possono essere utili per capire questi cambi [11].

Se pensiamo ai diversi modi che abbiamo per muoverci (focalizzandosi principalmente ai mezzi per mobilizzarsi da città a città) da un punto ad un altro, possiamo pensarci a quanto può diventare faticoso poter pianificare un viaggio. Perché possiamo avere diverse combinazioni per realizzare lo stesso tragitto, il problema è che in gran parte di queste soluzioni di viaggio, non si tengono in considerazione aspetti che possano essere d'interesse per l'utente.

Pensiamo all'idea di voler viaggiare da Roma a Milano. Se cerchiamo una soluzione per questo viaggio, le opzioni saranno:

- Treno.
- Autobus.
- Aereo.

Pero non ci sono delle soluzioni che tengano in considerazione i servizi che l'utente può trovare oppure il tempo di viaggio o anzi la comodità al momento di viaggiare. Queste cose per banali che siano, possono condizionare una scelta al momento di acquistare un viaggio.

2.2 Ontologia Proposta

La ontologia proposta per Mnasser Houda et. al [6] propone una soluzione alle problematiche presentate nella sezione precedente. Loro costruiscono un'ontologia tenendo in considerazione i punti menzionati nella sezioni 1, dove i servizi presenti, i modi di trasporto e la loro combinazione giocano un ruolo importante al momento di organizzare un viaggio.

Loro per sviluppare questa ontologia si basano nei seguenti steps:

- Scopo: Facilitare il recupero delle informazioni per la pianificazione del viaggio dell'utente.
- Concettualizzazione: permette la definizione dell'ambito dell'ontologia.
- Formalizzazione: consiste nell'esprimere la ontologia in alcun linguaggio e codice in uno specifico tools, in questo caso Protégé.

- Validazione.

L'articolo fatto da loro ha come obiettivo presentare un approccio che si concentra sull'uso dell'ontologia del trasporto pubblico per supportare la pianificazione dei passeggeri dalle diverse opzioni dedotte di un viaggio.

2.2.1 Concettualizzazione

Ci focalizziamo principalmente su questo punto che sarà quello che ci permetterà di comprendere meglio come viene fatta l'ontologia. Questo step risponde alle seguente domande:

- Cos'è la multi-modalità dei trasporti?
- Come si caratterizza un viaggio di trasporto?
- Come sono organizzati i punti di fermata del trasporto pubblico?
- Quali sono i servizi associati a un viaggio?
- Com'è l'infrastruttura del trasporto pubblico?
- Quali tipi di viaggio possono essere offerti a un passeggero?

Cos'è la multi-modalità dei trasporti? : per il loro lavoro vengono identificati 4 modi di trasporto che sono:

- Metro
- Tram
- Treno
- Autobus

In più viene segnalato che un modo di trasporto può avere diversi tipi di veicoli che possiedono diverse proprietà per un viaggio (ad esempio la quantità di posti a sedere).

Come si caratterizza un viaggio di trasporto? : viene segnalato che per una linea di trasporto ci possono essere diversi modelli di viaggi che hanno caratteristiche diverse ad esempio il costo (viaggiare in un autobus è più economico che viaggiare in un treno). Inoltre il modello di viaggio può dipendere anche dal giorno della settimana, cioè se è un feriale o un festivo.

Come sono organizzati i punti di fermata del trasporto pubblico? : in questo punto vengono identificati due tipi di punti di fermata che sono:

- punto di fermata in un modello di viaggio dove il passeggero non ha bisogno di cambiare veicolo per arrivare alla sua destinazione.
- punto di fermata di connessione, chiamato anche connection link, dove il passeggero ha disponibili diverse modalità di trasporto. In questo tipo di fermata possono essere necessari diversi tempi per coprire il collegamento a secondo dei passeggeri.

Quali sono i servizi associati a un viaggio? : vengono presentati diversi elementi geografici che sono localizzati nel punto di fermata di connessione, facendo riferimento al fatto che questi elementi possano fornire diversi servizi che possono essere d'interesse per l'utente, dato che tra un punto di fermata e la continuazione del viaggio può passare un po' di tempo, un punto importante che tengono in considerazione e anche se le fermate forniscono una sorta di protezione dalla pioggia. Gli elementi geografici non devono essere sempre uguali, questi possono essere disgiunti.

Com'è l'infrastruttura del trasporto pubblico? : in questo punto si associa ad ogni punto di sosta con qualche tipo di infrastruttura, per esempio, nel caso dei treni una stazione di treno, nei casi degli autobus una fermata stradale o come chiamano loro un *road junction*,

Quali tipi di viaggio possono essere offerti a un passeggero? : Nella ontologia viene descritto questo come *journey pattern* dove internamente viene classificato come:

- modello di viaggio diretto: quando il *journey pattern* è composto solo di punti di fermata.
- modello di viaggio indiretto quando il *journey pattern* ha qualche punto di fermata di connessione.

Inoltre il modello di viaggio diretto viene diviso in:

- modello di viaggio veloce
- modello di viaggio lento

Nel caso di un modello di viaggio indiretto questo definisce:

- Service journey pattern
- Interesting journey pattern
- Shopping journey pattern
- Leisure journey pattern
- Journey pattern with little walking
- Protected journey pattern

Per definire questo tipo di classificazione appena descritto, hanno considerato fondamentalmente i diversi tipi di elementi geografici, in più il modello di viaggio diretto e indiretto sono disgiunti mentre i tipi di un modello di viaggio indiretto non sono disgiunti, perché questo può essere sia un *Interesting journey pattern* che un *Protected journey pattern*

3 Resource Description Framework (RDF)

RDF è un modello standard per rappresentare conoscenza che viene consigliato da W3C, questo modello ci permette creare descrizione di risorse che sono un qualsiasi oggetto che è identificato univocamente per un Uniform Resource Identifier (URI). RDF viene usualmente indicato come una tripla nel modo:

$$Subject - Predicate - Object \tag{1}$$

3.1 Specifiche RDF

L'idea principale a questo punto è far vedere come vengono descritti alcuni elementi della ontologia usando RDF, si procederà a far vedere un insieme piccolo di elementi con la sua descrizione.

Procedere in questo modo ci permetterà di capire in modo più semplice quale sarà la differenza al momento di utilizzare OWL [5], che si farà vedere nella sezione 4

In seguito nella figura 3.1 abbiamo descritto parte dell'ontologia in modo molto semplice utilizzando turtle [2] che è una sintassi alternativa a *RDF/XML* che permette di serializzare i grafici RDF in una forma di testo più compatta e naturale per rendere più comprensibile il *RDF/XML*.

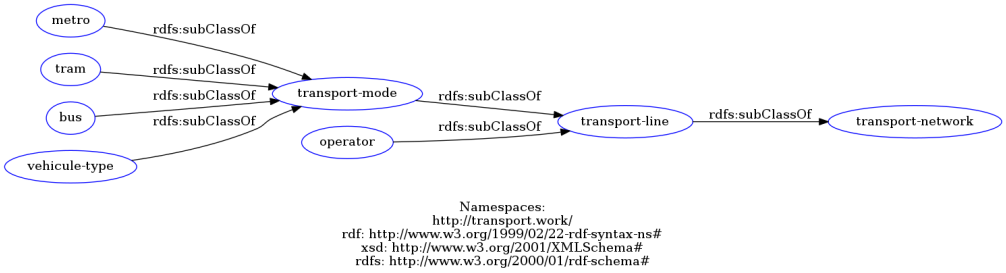


Figura 3.1 Rete di Trasporto

L'immagine 3.2 fa vedere come viene creata la immagine 3.1 utilizzando turtle per diventare così più leggibile. la parola *rdfs:subClassOf* puo essere usata per indicare che una classe è una sottoclasse di un'altra.

```
@prefix : <http://transport.work/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
    :metro rdfs:subClassOf :transport-mode.
    :tram rdfs:subClassOf :transport-mode.
    :bus rdfs:subClassOf :transport-mode.
    :vehicule-type rdfs:subClassOf :transport-mode.
    :transport-mode rdfs:subClassOf :transport-line.
    :operator rdfs:subClassOf :transport-line.
    :transport-line rdfs:subClassOf :transport-network.
```

Figura 3.2 Semantica Turtle

Inoltre se tornassimo alla sezione 2.2 ricorderemo la domanda *Quali sono i servizi associati a un viaggio?*. L'immagine 3.3 è una rappresentazione della descrizione fatta alla domanda precedente.

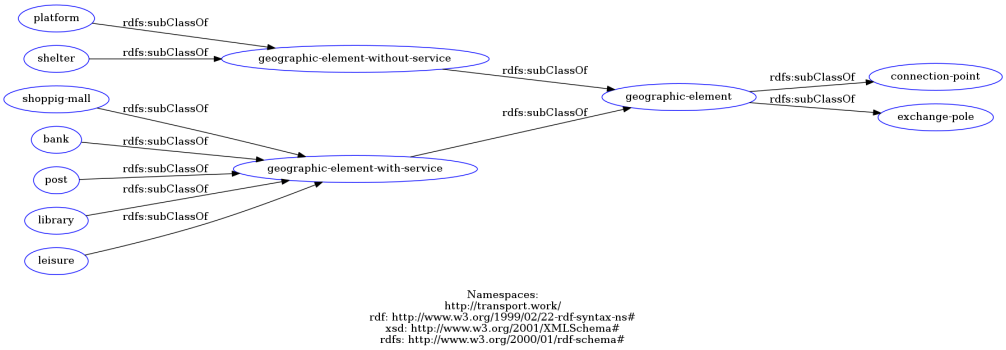


Figura 3.3 Elementi Geografici

Procedendo con la descrizione a livello RDF, si è voluto far vedere altre implementazioni, ognuna con riferimento ad una domanda specifica di quelle descritte nella sezione 2.2. In particolare si sono modellate le seguenti domande:

- *Quali sono i servizi associati a un viaggio?*
- *Com'è l'infrastruttura del trasporto pubblico?*
- *Quali tipi di viaggio possono essere offerti a un passeggero?*

dove si farà vedere anche l'implementazione fatta usando turtle.

Allo stesso modo di come si è presentato per l'immagine 3.1, si procede a far vedere un'implementazione a livello codice.

```
@prefix : <http://transport.work/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

    :shoppig-mall rdfs:subClassOf :geographic-element-with-service.
    :library rdfs:subClassOf :geographic-element-with-service.
    :leisure rdfs:subClassOf :geographic-element-with-service.
    :bank rdfs:subClassOf :geographic-element-with-service.
    :post rdfs:subClassOf :geographic-element-with-service.
    :shelter rdfs:subClassOf :geographic-element-without-service.
    :platform rdfs:subClassOf :geographic-element-without-service.
    :geographic-element-with-service rdfs:subClassOf :geographic-element.
    :geographic-element-without-service rdfs:subClassOf :geographic-element.
    :geographic-element rdfs:subClassOf :connection-point.
    :geographic-element rdfs:subClassOf :exchange-pole.
```

Figura 3.4 Semantica Turtle

L'immagine 3.5 come le altre due immagini in precedenza, è la rappresentazione della domanda fatta nella sezione 2.2 che come si può vedere dall'immagine fa riferimento a *Com'è l'infrastruttura del trasporto pubblico?*

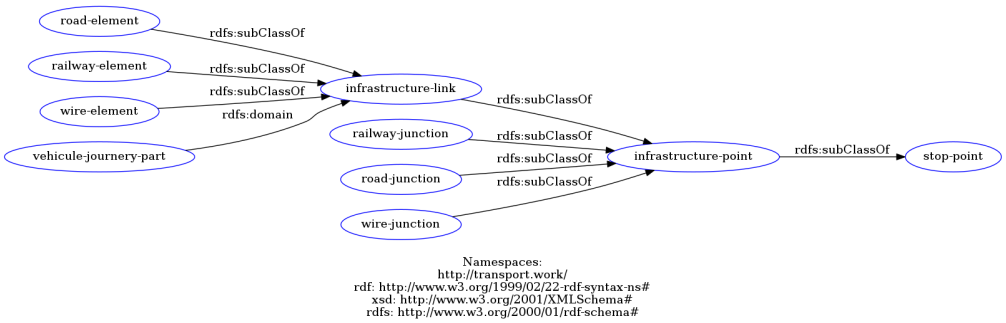


Figura 3.5 Infrastruttura Trasporto

```
@prefix : <http://transport.work/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

    :road-element rdfs:subClassOf :infrastructure-link.
    :railway-element rdfs:subClassOf :infrastructure-link.
    :wire-element rdfs:subClassOf :infrastructure-link.
    :vehicule-journery-part rdfs:domain :infrastructure-link.
    :infrastructure-link rdfs:subClassOf :infrastructure-point.
    :railway-junction rdfs:subClassOf :infrastructure-point.
    :road-junction rdfs:subClassOf :infrastructure-point.
    :wire-junction rdfs:subClassOf :infrastructure-point.
    :infrastructure-point rdfs:subClassOf :stop-point.
```

Figura 3.6 Semantica Turtle

L'immagine 3.7 fa riferimento alla seguente domanda *Quali tipi di viaggio possono essere offerti a un passeggero?*, in quest'immagine possiamo chiaramente vedere che gli elementi sono tutti di tipo *rdf:type* e che il modello di viaggio indiretto è disgiunto dal modello di viaggio diretto, ma che i modelli di viaggi dentro la categoria indiretti tra di loro non sono disgiunti.

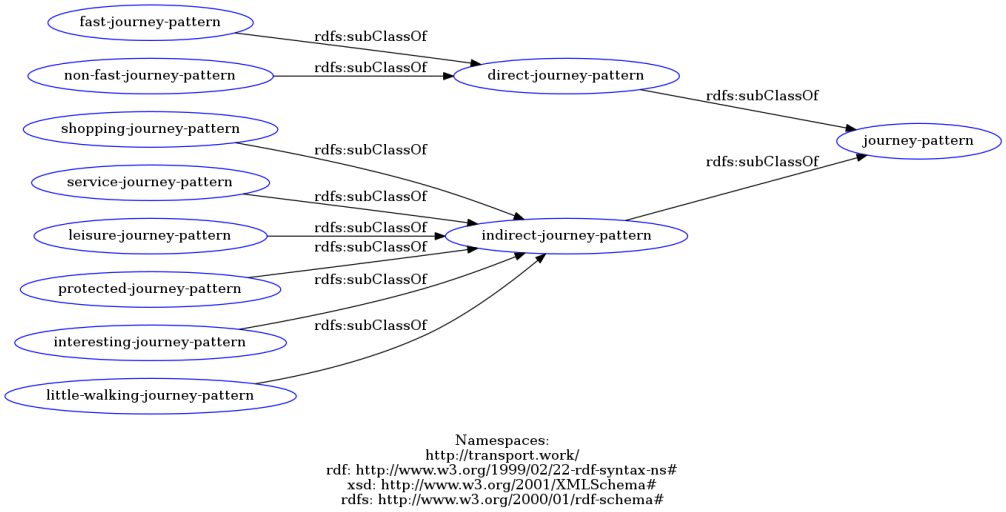


Figura 3.7 Modelli di Viaggio

A questo punto faremo una piccola descrizione del modello appena presentato. Si nota che la decomposizione di un viaggio diretto è:

- non-fast-journey-pattern: quando questo usa solo elementi ferroviari.
- fast-journey-pattern: quando il veicolo di viaggio usa solo elementi su strada.

Mentre un modello di viaggio indiretto è può essere:

- shopping-journey-pattern: quando nel punto di fermata c'è qualche centro commerciale.
- protected-journey-pattern: quando nel punto di fermata c'è qualche struttura per riposarsi.
- interesting-journey-pattern: quando nel punto di fermata c'è qualche struttura d'interesse.
- ecc.

cioè, un viaggio indiretto esclude un viaggio diretto. Un viaggio che includa *shopping - protected - interesting - ecc*, può solo appartenere a un modello di viaggio indiretto ma non a un modello di viaggio diretto. Entrambe queste tipologie di viaggio, insieme, formano quello che viene chiamato modello di viaggio.

```
@prefix : <http://transport.work/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

    :shopping-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :protected-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :interesting-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :little-walking-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :service-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :leisure-journey-pattern rdfs:subClassOf :indirect-journey-pattern.
    :indirect-journey-pattern rdfs:subClassOf :journey-pattern.
    :direct-journey-pattern rdfs:subClassOf :journey-pattern.
    :fast-journey-pattern rdfs:subClassOf :direct-journey-pattern.
    :non-fast-journey-pattern rdfs:subClassOf :direct-journey-pattern..
```

Figura 3.8 Semantica Turtle

4 Web Ontology Language (OWL)

OWL è un linguaggio web semantico disegnato per rappresentare una conoscenza ricca e complessa su cose, gruppi di cose e relazioni tra queste [5]. Uno dei benefici principali è che OWL offre un vocabolario molto più esteso che possiamo usare per rappresentare cose. Inoltre è molto rigido perché non solo ci dice come possiamo usare il vocabolario, ma ci dice anche come non possiamo usarli. In più ci permette poter riutilizzare lavoro già fatto.

4.1 Estensione Ontologia

L'ontologia utilizzata fin ora, ci permette di poter ottenere dei journey pattern in modo tale da poter consentire all'utente di scegliere quello più comodo per se.

L'estensione proposta è quella di poter aggiungere delle review associate alle linee di trasporto, che possono essere in diverse città di diversi paesi. In più elimineremo quello che viene considerato ridondante, in modo tale da far si che l'ontologia sia più chiara, cosa che ci permetterà poterla svolgere con maggior comodità.

Inizialmente l'ontologia presentata aveva la struttura presentata nella sezione 3, in questo punto l'ontologia ha la seguente forma:

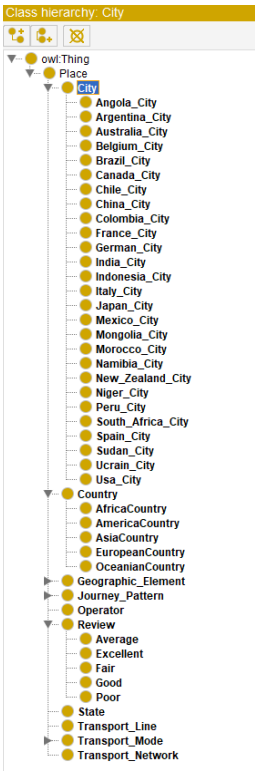


Figura 4.9 Struttura Ontologia Trasporto

Possiamo osservare che nella immagine 4.9 abbiamo sia *Città* che *Country* che *State* che *Review*, nelle sezioni seguenti procederemo a spiegare ognuna di queste classi 4.2, cosi come le object properties 4.3 e data properties 4.4 aggiunte in modo tale da poter raggiungere il nostro obiettivo.

4.2 Classi

Le classi in owl vengono conosciute anche come categorie [4], che ci permettono di poter rappresentare un set di risorse che condividono caratteristiche comuni e sono simili in qualche modo. Tutte le classi in owl devono essere membri di owl:class. In seguito si presentano le classi aggiunte all'ontologia e in più una piccola descrizione.

- Place: incapsula tutte le classi indicando che tutte queste fanno parte di un place, cioè una città è un place e un journey pattern si trova in un place, lo stesso con le country, gli state, gli elementi geografici e le rete di trasporto
- City: riferito alle possibili città che possono esistere, l'idea è di poter ottenere tutte le review delle linee di trasporto per città oppure fare dei confronti tra diversi journey pattern di diverse città, dove la metrica saranno le review.
- State: riferito a un livello più alto della città, uno state può contenere tante città e appartenere a un country, ha lo stesso scopo della città però ad un livello più astratto.
- Country: incapsula i paesi per continente, questo e un livello ancora più alto per poter fare un paragone tra le diverse linee di trasporto basandoci sulle review.
- Review: riferito alle diverse valutazioni che può fare un utente riferito a un servizio, sebbene l'ontologia principale permetta di poter offrire dei servizi di viaggi agli utenti, questa non considera la qualità del servizio, cosa molto importante e che può portare ad un problema maggiore, cioè offrire servizi di scarsa qualità.

Nella classe City sono state aggiunte delle *subClassOf* che permettono di poter fare una restrizione alle cita, cioè indichiamo delle classi per esempio *Italy_City*, che sono *subClassOf* di città pero usano in più la Object Properties *isCityOf*, per indicare che un'Italy_City ha come *value* Italy, questo verrà ripetuto per ogni paese che sia dentro l'ontologia.

Nella classe review come si può vedere nella immagine 4.9, ci sono diversi tipi di review che sono *subClassOf* di review, queste tra di se sono tutte disgiunte, dato che una review non può essere di due tipi diversi, cioè, o è Excellent o è Poor ma non entrambe. Le review non sono di tipo functional perché una transport line può avere diversi tipi di review, che ci permetteranno di sapere quanto è sofisticato il servizio.

4.3 Object Properties

Continuando dalla sezione 4.2, analizziamo le object properties, dove per properties intendiamo le relazioni esistenti tra diversi oggetti, nel momento in cui si definisce una object properties possiamo aggiungere delle proprietà a questo, per esempio come definito a lezione: *se definisco un object properties hasSex posso definire sia il dominio che il range*, se voglio che una persona hasSex un solo sex, allora devo indicare che è *Functional*. In questa ontologia le object properties aggiunte facendo sempre riferimento alle classi descritte prima sono:

- containsTransportLine: indica che un Journey Pattern contiene qualche Transport Line
- hasCity: indica che un Country ha una City
- hasGeographicElement: indica che un Journey Pattern può avere un Geographic Element
- hasOperator: indica che una Transport Line ha un Operator
- hasReview: indica che una Transport Line ha una Review
- hasState: indica che un Country ha uno State
- hasTransportLine: Indica che un Transport Network ha una Transport Line
- hasTransportLineC: indica che una City ha una Transport Line
- isCityOf: indica la proprietà inversa di hasCity
- isOfTransportLine: indica la proprietà inversa di hasReview
- isOfType: indica un Transport Mode per una Transport Line
- isOperatorOf: indica la proprietà inversa di hasOperator
- isPartJourney: indica la proprietà inversa di containsTransportLine
- isPartOf: indica la proprietà inversa di hasCity
- isPartOfCity: indica la proprietà inversa di hasTransportLine
- isStateOf: indica la proprietà inversa di hasState

A differenza della sottosezione 4.2, qua abbiamo elencato tutte le object properties presenti nella ontologia,dato che nella sezione 3 abbiamo già fatto una introduzione delle classi esistenti nella ontologia.

4.4 Data Properties

Una proprietà in OWL è una risorsa utilizzata come predicato nelle istruzioni che descrivono gli individui, dentro delle properties conosciute anche come relazioni abbiamo le data properties che ci permettono di poter indicare dei valori che dopo possono essere usati per gli individuals e che ci tornerà utile anche per la sezione 7 dove si spiega il procedimento per fare delle query usando SPARQL. Quando definiamo una data properties, possiamo indicare il suo dominio ed il suo range, dove per dominio ci facciamo riferimnto alle classi su cui verrà usata questa data property e per range facciamo riferimento ai possibili valori che può prendere. Le data property che abbiamo definito sono le seguenti:

- costOfLiving: questa data property ci permette di poter definire un costo di vita per nazione in modo tale di poter sfruttare il fatto che la qualità di un mezzo di trasporto può essere anche influenzata dal costo di vita del paese. Il suo dominio è Country ed il suo range è Integer.
- name: questa data property ha come dominio Place perché qualsiasi individuo può avere un name, cioè per esempio se definisco i miei individui con identificatore particolare e voglio assegnargli un nome più esplicito, posso usare questa data property.
- population: indica la quantità di popolazione di una nazione, ci viene utile per quantificare se la densità di popolazione di una nazione più il costo della vita influenza la qualità di un mezzo di trasporto.
- review: indica il commento fatto nella review.
- valutazione: indica il punteggio dato a una review.

4.5 Individuals

Gli individual sono qualsiasi risorsa che sia membro di almeno una classe, una risorsa che è un membro di una classe è chiamata individuo e rappresenta un'istanza di quella classe, gli individuals possono diventare membri delle classi sia in modo diretto che indiretto. Proseguendo con l'ontologia, a questo punto possiamo definire gli individuals e anche le loro assertions che possono essere sia *Object Property assertions* che *Data Property assertions*. Si può vedere l'elenco completo degli individuals nel seguente link **Progetto Web Semantico**

4.6 OntoGraf

OntoGraf è un componente di protégé che ci permette di poter visualizzare e navigare sulle relazioni della nostra ontologia in modo interattivo [3], cioè possiamo navigare attraverso il grafo vedendo sia le relazioni che le proprietà definite, in più possiamo vedere gli individuals definiti e le loro associazioni. OntoGraf in più ci permette di poter vedere le informazioni riferite alle URI, Superclasses, ed in generale ci fa vedere la informazioni che descrivono una classe. L'immagine 4.10 è una rappresentazione dell'ontologia descritta fin ora

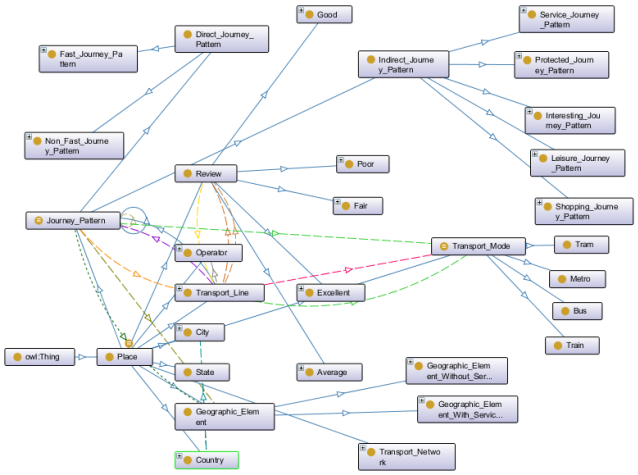


Figura 4.10 Struttura Ontologia Trasporto

4.7 DL query

DL query fornisce una funzione che è potente e facile per cercare in un'ontologia, la sintassi utilizzata si basa sulla sintassi Manchester, prima di usare DL query per interrogare l'ontologia bisogna che questa sia classificata da un reasoner. In seguito si presentano 3 immagini 4.11 4.12 4.14 di diverse query fatte sull'ontologia usando DL query.

Questa Query ha come obiettivo far vedere le linee di trasporto che sono in Brasilia e che hanno delle Review Excellent.

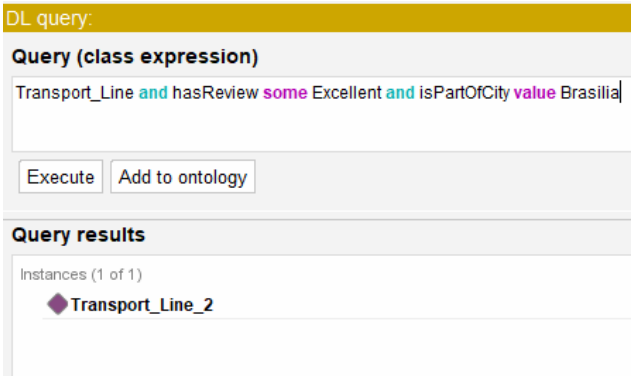


Figura 4.11 Linea di Trasporto con Review

Questa query permette di poter vedere tutte le città europee che hanno delle linee di trasporto e dove queste linee di trasporto hanno delle Review, se volessimo filtrare in modo più specifico, potremmo cambiare Review per il tipo di Review che ci interessa, che può essere *Excellent, Good, Average, Fair, Poor*.

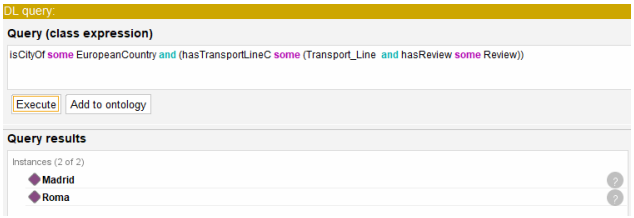


Figura 4.12 Città Europea con Linea di Trasporto

Questa query ci permette di poter ottenere i paesi che hanno delle linee di trasporto che hanno delle Review ma dove il costo di vita è superiore a 17000, Come detto in precedenza, possiamo modificare *hasReview some Review* con uno dei tipi descritti prima, per ottenere in modo più specifico i risultati finali.

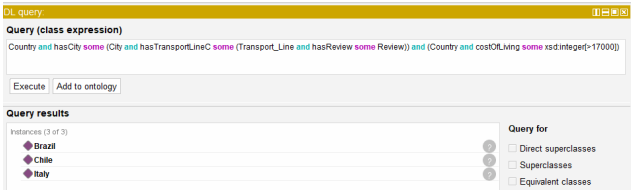


Figura 4.13 Paesi con Linee di Trasporto

4.8 Apache Jena

Apache Jena è un software open source [7] per costruire applicazioni di Web Semantico, questo supporta solo Owl 1.1 ma è possibile usare Jena insieme ad altri framework per costruire applicazioni dove è supportato Owl2. L' immagine seguent presenta una delle capacita di Apache Jena, in questo caso ci permette di esporre le nostre triple come endpoint SPARQL accessibili tramite HTTP, in più Fuseki fornisce un'interazione in stile REST con i dati RDF. Questa API rende semplice e intuitivo fare delle query su SPARQL, l'unica limitazione è che i file che si caricano devono avere un formato RDF, questi non possono essere in formato turtle.

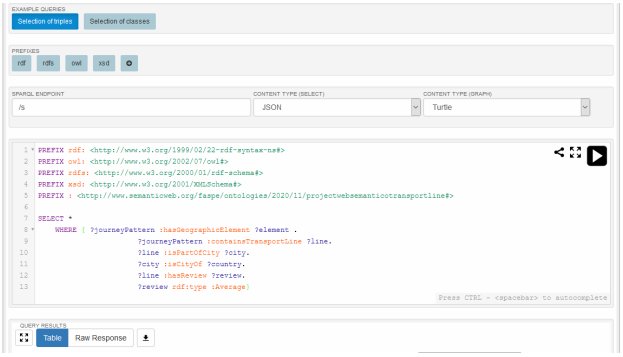


Figura 4.14 Apache Jena, Fuseki EndPoint

4.9 Owl Api 2

OWL Api 2 ci permette di poter lavorare a livello codice con ontologie Owl, questa supporta Owl2 e ci rende facile poter dividere tutti gli individui per classi oppure realizzare delle associazioni, in piu Owl2 ci permette di aggiungere delle relazioni alla nostra ontologia, nella documentazione ci indica che gli Individuals (o classes o properties) non possono essere aggiunti direttamente all' ontologia, prima devono essere usati negli assiomi, e poi gli assiomi vengono aggiunti ad un'ontologia [10]. In seguito un esempio di come possiamo aggiungere una *Review* che è di tipo *Average*.

L'immagine seguente mostra il modo in cui è possibile elencare gli individui appartenenti ad una classe, prima creiamo un'istanza del reasoner per chiedere poi al reasoner di pre-calcolare alcuni tipi di inferenze, dopo quello prendiamo il prefisso della nostra ontologia definita inizialmente e usiamo *getOWLClass* per tornare un'istanza di OWLClass che ha una specifica IRI. Dopo quello facciamo tornare tutti gli individui che sono appartenenti alla specifica classe definita prima e li stampiamo.

```
public void listIndividual(String nameClass){
    OWLReasoner reasoner = ontology.owlReasoner();
    reasoner.precomputeInferences(InferenceType.values());
    OWLClass transporter = dataFactory.getOWLClass(IRI.create(prefixManager.getDefaultPrefix(), nameClass));

    NodeSet<OWLNamedIndividual> individualsNodeSet = reasoner.getInstances(transporter,  direct: false);

    for(Node<OWLNamedIndividual> individual : individualsNodeSet){
        individual.entities().forEach(x-> System.out.println(prefixManager.getShortForm(x)));
    }
}
```

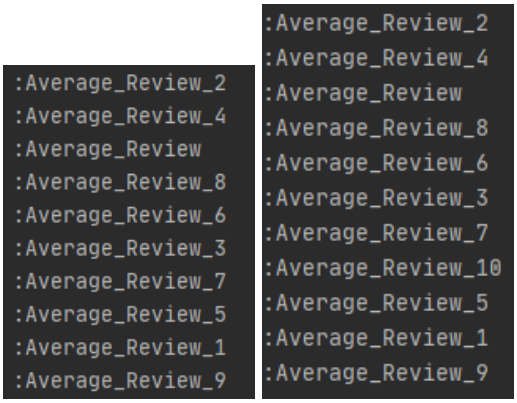
Figura 4.15 Elenco Individui Appartenente alla Class Average

L'immagine successiva mostra il modo in cui possiamo aggiungere un individuo, come detto prima gli Individuals (o classes o properties) non possono essere aggiunti direttamente all'ontologia ed è per questo che bisogna creare prima un assioma che verrà dopo aggiunto all'ontologia. Allora inizialmente creiamo un individual, dopo prendiamo la classe a cui apparterrà e in questo modo creiamo un *OWLClassAssertionAxiom* che indica classe e individual, dopo questo creiamo un'istanza di *AddAxiom* dove questo rappresenta un cambio nell'ontologia, finalmente usiamo *applyChange* che è un metodo che applica solo un cambio all'ontologia per poi salvare i cambi usando *saveOntology*

```
public void createAxiom(String nameIndividual, String nameClass){
    OWLIndividual newAverage = dataFactory.getOWLNamedIndividual(IRI
            .create(prefixManager.getDefaultPrefix() + nameIndividual));
    OWLClass average =dataFactory.getOWLClass( iri: prefixManager.getDefaultPrefix()  + nameClass);
    OWLClassAssertionAxiom axiom = dataFactory.getOWLClassAssertionAxiom(average,newAverage);
    AddAxiom addAxiom1 = new AddAxiom(ontology.getOntology(), axiom);
    ontology.applyChanges(addAxiom1);
    ontology.saveOntology();
}
```

Figura 4.16 Creazione Assioma

A questo punto dopo aver creato un assioma e averlo aggiunto all'ontologia, possiamo vedere i cambi su questa, come si può vedere nell'immagine seguente. A sinistra c'è la nostra ontologia prima di aggiungere un nuovo individual, a destra l'ontologia dopo aver aggiunto un individual.



(a) Prima di aggiungere un assioma (b) Dopo aggiungere un assioma

Figura 4.17 Aggiunta Assioma

Per approfondimenti su OWL API 2 si può guardare nel seguente link **OWLAPI**

5 SPARQL

SPARQL è un linguaggio standardizzato per l'interrogazione di grafici RDF, inizialmente è stato creato solo per il recupero di dichiarazioni RDF. Tuttavia, alcune proposte includono anche operazioni per la manutenzione (creazione, modifica e cancellazione) dei dati [12]. l'interrogazione segue sempre questa struttura:

$$Subject - Predicate - Object \tag{2}$$

Per realizzare delle query in SPARQL, possiamo usare direttamente la scheda SPARQL Query che viene inclusa in protegè oppure possiamo usare Fuseki di Apache Jena per interrogare la nostra ontologia. Per comodità le query che si mostrano in seguito vengono svolte usando Fuseki che secondo me è più completo e intuitivo di quello integrato su protégé.

L'immagine seguente torna tutti i Journey Pattern che hanno elementi geografici di tutte le città di ogni paese e dove la sua Review è di tipo *Average*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>

SELECT *
    WHERE { ?journeyPattern :hasGeographicElement ?element .
                        ?journeyPattern :containsTransportLine ?line.
                        ?line :isPartOfCity ?city.
                        ?city :isCityOf ?country.
                        ?line :hasReview ?review.
                        ?review rdf:type :Average}
```

Figura 5.18 Journey Pattern Con Elementi Geografici

L'immagine seguente torna tutte le Transport Line dove il costo di vita sia maggiore a 17000, dove le Review siano di tipo Excellent e dove la città sia di tipo AmericaCountry

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>

 SELECT ?country ?city ?transportLine ?review ?cost
    WHERE { ?country :hasCity ?city .
                      ?country :costOfLiving ?cost .
                      ?city :hasTransportLineC ?transportLine.
                       ?transportLine :hasReview ?review.
                      ?review rdf:type :Excellent.
                      ?country rdf:type :AmericaCountry.
filter(?cost > "17000"^^xsd:integer)}
```

Figura 5.19 Linea di Trasporto di America

Se volessimo vedere per paese quali sono quelli dove ci sono delle Transport Line che hanno più Review ci basterebbe fare un *GROUP BY* e fare *COUNT(DISTINC(?review))*, in modo che ci venga dato un risultato raggruppando ogni Review e sommandola, l'immagine seguente fa vedere quello appena detto.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>


SELECT ?country (count(distinct(?review)) as ?total)
    WHERE { ?country :hasCity ?city .
                      ?country :costOfLiving ?cost .
                      ?city :hasTransportLineC ?transportLine.
                      ?transportLine :hasReview ?review.
filter(?cost > "17000"^^xsd:integer)}
group by ?country
order by desc(?total)
```

Figura 5.20 Paesi con Transport Line con più Review

La query appena presentata ha dei problemi se un paese non specifica il suo costo di vita, in questo caso quei paesi vengono scartati dalla query precedente. Se vogliamo che tali paesi vengano considerati nella nostra query dovremmo indicare che quel campo ha *OPTIONAL*. L'immagine seguente fa vedere quello appena detto.

```
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>

SELECT ?country ?city (count(distinct(?review)) as ?quantita)
where { ?country :hasCity ?city .
    optional{?country :costOfLiving ?cost .}
    ?city :hasTransportLineC ?transportLine.
    ?transportLine :hasReview ?review.
  optional{
    filter(?cost > "17000"^^xsd:integer)}}
group by ?country ?city
order by desc(?quantita)
```

Figura 5.21 Paesi con Transport Line con più Review 2

Nel casso volessimo sapere quali sono le città con Transport Line con più Review basta indicare nella *GROUP BY* un secondo campo per il quale si deve raggruppare come si fa vedere nella immagine successiva.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>
SELECT ?country  ?city (count(?review) as ?total)
    WHERE { ?country :hasCity ?city .
    optional{
            ?country :costOfLiving ?cost .
            }
            ?city :hasTransportLineC ?transportLine.
            ?transportLine :hasReview ?review.
  optional{
      filter(?cost > "17000"^^xsd:integer)
         }
        }
group by ?country ?city
order by desc(?total)
```

Figura 5.22 Cita per Paesi con Transport Line con più Review

Nelle immagini 5.21 5.22 si può osservare che abbiamo usato la keyword *OPTIONAL* per indicare che un campo può essere presente o meno, però dobbiamo anche indicarlo nel *FILTER* perché nel caso la data properties non fosse stata dichiarata e il filter non fosse optional allora non verrebbe considerato quel paese e tutte le review esistenti non sarebbero visualizzate.

Se volessimo un livello di dettaglio più ampio, cioè, vedere le Transport Line con più Review, basterebbe aggiungere un altro campo e proseguire come nel caso precedente. Nell'appendice 8 si fa vedere la query appena descritta.

6 Semantic Web Rule Language (SWRL) e Semantic Query-Enhanced Web Rule Language (SQWRL)

6.1 Semantic Web Rule Language (SWRL)

Lo SWRL è un linguaggio espressivo basato su OWL che permette agli utenti di scrivere regole che si possono esprimere in termini di concetti di OWL per fornire capacità di ragionamento deduttivo più potente di OWL [8]

Una regola SWRL contiene una parte antecedente che è definita come *body* e una parte conseguente che è definita come *head*, entrambi consistono in unioni di atomi.

$$atom \wedge atom....- > atom \wedge atom... \tag{3}$$

Un atom in SWRL è della forma *p(arg,arg2 ... argn)* dove p è il predicato e arg gli argomenti dell'espressione. I tipi di predicati supportati da SWRL sono i seguenti:

- OWL classes: consiste di una classe o una espressione di classe e un singolo argomento che rappresenta un OWL individual [8].
- OWL properties: consiste di un OWL object properties con due argomenti che rappresentano OWL Individuals [8].
- data types: consiste di un OWL data properties con due argomenti che rappresentano il primo un OWL Individuals mentre che il secondo un data value [8].
- data ranges: consiste di un datatype o un set di letterale e un singolo argomento che rappresenta un data value [8].
- built-ins: è un predicato che prende uno o più argomenti e restituisce true se l'argomento soddisfa il predicato [8]

Mentre gli argomenti possono essere individui oppure data values o variabili che si riferiscono a questi. Allora un atomo può avere la seguente forma usando uno dei predicati descritti in precedenza:

$$Person(?p) \wedge hasSex(?p, ?s) \wedge Male(?s) \rightarrow Man(?p) \tag{4}$$

Dove *Person* è un OWL class e *hasSex* è una OWL properties, allora se la persona ha sesso maschio vuol dire che è un uomo.

6.2 Semantic Query-Enhanced Web Rule Language (SQWRL)

Lo SQWRL e un SWRL-based query language che fornisce operatori come SQL per estrarre informazioni da ontologie OWL [9].

Questo linguaggio offre due sets di operatori per interrogare, che sono:

- Core Operators
- Collection Operators

Una caratteristica importante di SQWRL è che questa non modifica l'ontologia, cioè il risultato di una query non può essere inserito nell'ontologia, dato che tale meccanismo potrebbe invalidare la *open world assumption* di OWL e portare ad una mancanza di monotonia. Però questi tipi di operatori possono operare insieme a quelli di SWRL in modo tale da creare una regola che dopo possiamo interrogare usando SQWRL, seguendo l'esempio fatto in precedenza 4:

$$Man(?p) -> sqwrl : count(?p) \tag{5}$$

La espressione in precedenza conta tutti gli uomini presente nell'ontologia.

6.3 Regole Implementate

Le regole implementate nell'ontologia sono le seguenti:

- Questa regola di interrogazione conta tutte le linee di trasporto che hanno delle *Excellent Review* in più basta cambiare il tipo *Excellent* negli altri tipi di review per poter contare tutte le *Transport Line* con diverse *Review*.

$$Transport\_Line(?t) \wedge hasReview(?t, ?r) \wedge Excellent(?r) \rightarrow sqwrl : count(?t) \tag{6}$$

- Indica che una review con valutazione uguale a 5 viene considerata come *Excellent*, come nel caso precedente, basta cambiare il valore passato in *swrlb:equal(?v, 5)* per creare le regole per gli altri tipi di review.

$$swrlb : equal(?v, 5) \wedge valutazione(?r, ?v) \rightarrow Excellent(?r) \tag{7}$$

- Indica che un *Journey Pattern* che ha un elemento geografico che è sia una *Bank* e anche una *Library* allora verrà classificato sia come *Service Journey Pattern* sia come *Interesting Journey Pattern*, nello stesso modo possiamo dire che altri *Journey Pattern* che abbiano delle caratteristiche in comune possono essere classificati con più di un tipo.

$$\begin{aligned} &Journey\_Pattern(?j) \wedge hasGeographicElement(?j, ?x) \wedge hasGeographicElement(?j, ?l) \\ &\wedge Bank(?x) \wedge Library(?l) \rightarrow Service\_Journey\_Pattern(?j) \wedge Interesting\_Journey\_Pattern(?j) \end{aligned} \tag{8}$$

7 Conclusione

La realizzazione di questa estensione mi ha servito per capire il potere presente in OWL e come questa permette di realizzare dei modelli che possono aiutare l'utente finale. Come si è presentato in questa relazione, l'ontologia presentata ha come obiettivo poter rendere l'uso del trasporto pubblico più semplice. Aggiungendo delle review e dei paesi, permettiamo che questa sia ancora più ampia e che l'utente finale valuti in modo migliore la scelta che sta realizzando, non solo per i servizi presenti nel tragitto ma anche per la valutazione che ha una specifica linea di trasporto. Sebbene altri lavori possano essere applicati in questa area per renderla ancora più utile, è importante considerare che un servizio non è buono solo per quello che si sta offrendo ma anche per la qualità dello stesso.

Come lavoro futuro si propone di poter riempire l'ontologia e anche di aggiungere altre classi per considerare altri aspetti come per esempio:

- Servizi offerti nel viaggio, intesso come se si viaggia la notte, c'è ha disposizione qualche comodità?.
- Altri servizi come bagno, caffetteria, ecc.

Riferimento

[1] AGI. Trasporti: ogni anno 5,8 miliardi di passeggeri. https://www.agi.it/economia/tpl_ogni_anno_5_8_miliardi_di_passeggeri-490297/news/2016-02-05/, Feb. 2016.

[2] E. P. W. G. C. L. M. I. David Beckett, Tim Berners-Lee (W3C). Rdf 1.1 turtle. https://www.w3.org/TR/turtle/, Feb. 2014.

[3] S. Falconer. Ontograf. https://protegewiki.stanford.edu/wiki/OntoGraf, Apr. 2010.

[4] L. D. R. Fulvio Corno, Laura Farinetti. Ontologies. https://elite.polito.it/files/courses/01RRDIU/2019/4-Ontologies-OWL.pdf, Dec. 2016.

[5] O. W. Group. Web ontology language (owl). https://www.w3.org/OWL/, 2012.

[6] M. Houda, M. Khemaja, K. Oliveira, and M. Abed. A public transportation ontology to support user travel planning. In *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*, pages 127–136. IEEE, 2010.

[7] H. Labs. Apache jena. https://jena.apache.org/index.html, May 2020.

[8] M. O'Connor. Swrllanguagefaq. https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ#Does_SWRL_adopt_the_Open_World_Assumption, June 2019.

[9] M. J. O'Connor and A. K. Das. Sqwrl: a query language for owl. In *OWLED*, volume 529, 2009.

[10] N. M. . I. Palmisano. A introduction to the owl api. http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP62342/introduction-owl-api-msc.pdf, Apr. 2016.

[11] UITP. Public transport urban mobility data. https://www.uitp.org/data/, May 2019.

[12] W3C. Sparql. https://it.wikipedia.org/wiki/SPARQL, Jan. 2008.

8 Appendice

Transport Line con più Review ordinate in modo discendente

```
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>

SELECT ?country ?city ?transportLine (count(distinct(?review)) as ?quantita)
where { ?country :hasCity ?city .
    optional{?country :costOfLiving ?cost .}
    ?city :hasTransportLineC ?transportLine.
    ?transportLine :hasReview ?review.
  optional{
    filter(?cost > "17000"^^xsd:integer)}}
group by ?country ?city ?transportLine
order by desc(?quantita)
```

Figura 8.23 Transport Line con più Review

Select delle prime 5 Transport Line con piu Review

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>
SELECT ?country  ?city (count(?review) as ?total)
    WHERE { ?country :hasCity ?city .
    optional{
            ?country :costOfLiving ?cost .
            }
            ?city :hasTransportLineC ?transportLine.
            ?transportLine :hasReview ?review.
  optional{
      filter(?cost > "17000"^^xsd:integer)
          }
        }
group by ?country ?city
order by desc(?total)
limit 5
```

Figura 8.24 Prime 5 Transport Line con più Review

Se vogliamo conoscere su come è costituito un grafo RDF possiamo usare la keyword *DESCRIBE* in modo tale che ci faccia tornare i termini utilizzati nelle triple, un esempio potrebbe essere:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#>
Describe :Italy
```

QUERY RESULTS

Table | Raw Response

```
@prefix :      <http://www.semanticweb.org/faspe/ontologies/2020/11/projectwebsemanticotransportline#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

:Italy  a              :EuropeanCountry , owl:NamedIndividual ;
        :costOfLiving  25000 ;
        :hasCity       :Bologna , :Roma .
```

Figura 8.25 Describe Italy

Lo stesso risultato possiamo ottenerlo usando *SELECT * :Italy ?p ?o*

La seguente immagine è la ontologia vista a un livello più astratto:

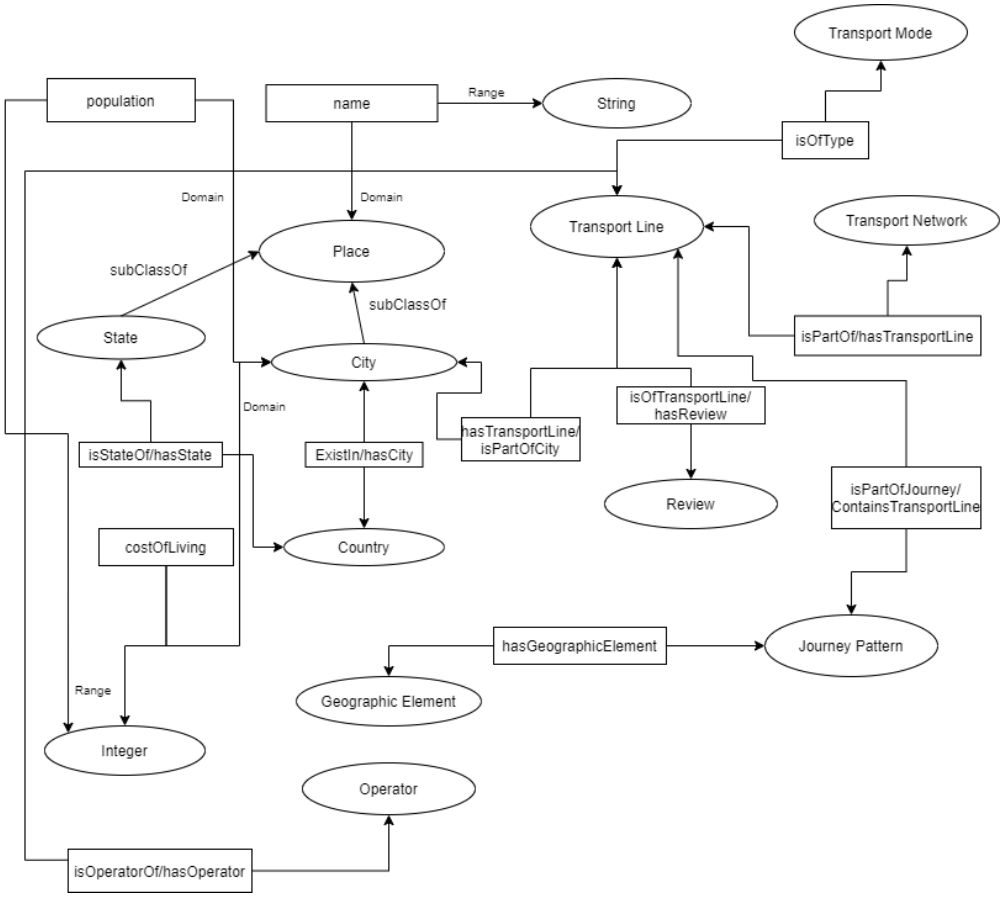


Figura 8.26 Ontology Transport