Consider a very simple `personsExample-01` ontology:

Listing 1: Content of `personsExample-01.ttl`

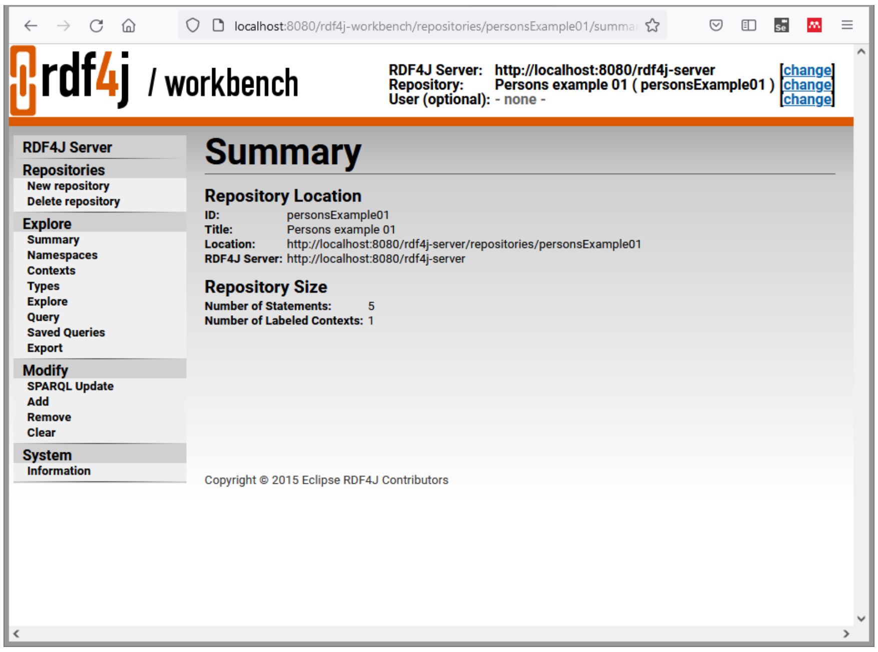
```
@prefix ex: <http://example.org#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:Man rdfs:subClassOf ex:Person .
ex:Woman rdfs:subClassOf ex:Person .
ex:hasWife rdfs:domain ex:Man ;
           rdfs:range ex:Woman .

ex:john ex:hasWife ex:mary .
```

It consist of 5 triples. This can be confirmed after uploading it to the RDF4J in memory store without any reasoning turned on (see Figure 1a). The corresponding explicit graph can be visualized in a simple chart (see Figure 1b).

a)



RDF4J Server: http://localhost:8080/rdf4j-server [change]
Repository: Persons example 01 (personsExample01) [change]
User (optional): - none - [change]

Summary

Repository Location
ID: personsExample01
Title: Persons example 01
Location: http://localhost:8080/rdf4j-server/repositories/personsExample01
RDF4J Server: http://localhost:8080/rdf4j-server

Repository Size
Number of Statements: 5
Number of Labeled Contexts: 1

Copyright © 2015 Eclipse RDF4J Contributors

b)

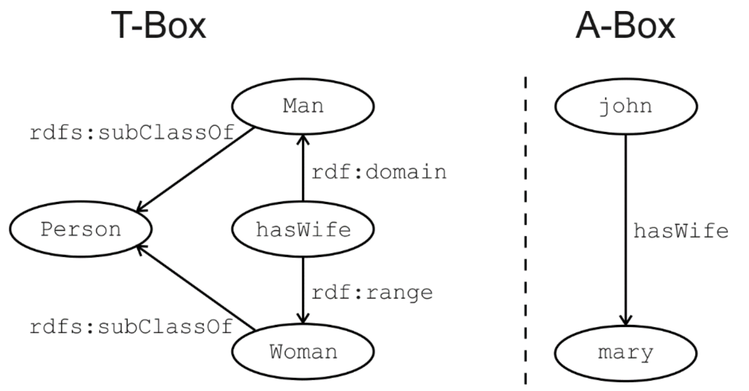


Figure 1: `personsExample-01` ontology: a) summary after uploading to RDF4j triple store, b) explicit graph visualization

However, if the RDFS reasoning is turn on, the number of triples increases. The resulting ontology gets additional, inferred triples. The total number of triples in the considered case will be 170. The triples directly related to the elements of the explicit graph (i.e. to the elements from ex: namespace) are presented In Listing 2. Other triples (having roots in RDF and RDFS model) are omitted.

Listing 2: Content of `personsExample-02.ttl` with inferred triples (the originally declared triples are highlighted in blue, the triples without any direct relation to the explicit graph are omitted).

```
@prefix ex: <http://example.org#> .
@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#> .

ex:Man rdf:type rdfs:Resource, rdfs:Class ;
       rdfs:subClassOf rdfs:Resource, ex:Man, ex:Person, ex:Person .

ex:Woman rdf:type rdfs:Resource, rdfs:Class ;
       rdfs:subClassOf rdfs:Resource, ex:Woman, ex:Person, ex:Person .

ex:hasWife rdf:type rdfs:Resource, rdf:Property , rdf:Property ;
       rdfs:subPropertyOf ex:hasWife ;
       rdfs:domain ex:Man ;
       rdfs:range ex:Woman .

ex:Person rdf:type rdfs:Resource, rdfs:Class ;
       rdfs:subClassOf rdfs:Resource,  ex:Person .

ex:john rdf:type rdfs:Resource, ex:Man, ex:Person ;
       ex:hasWife ex:Mary .

ex:mary rdf:type rdfs:Resource, ex:Woman, ex:Person .
```

The corresponding implicit graph (with omitted triples) can be drawn as follows:

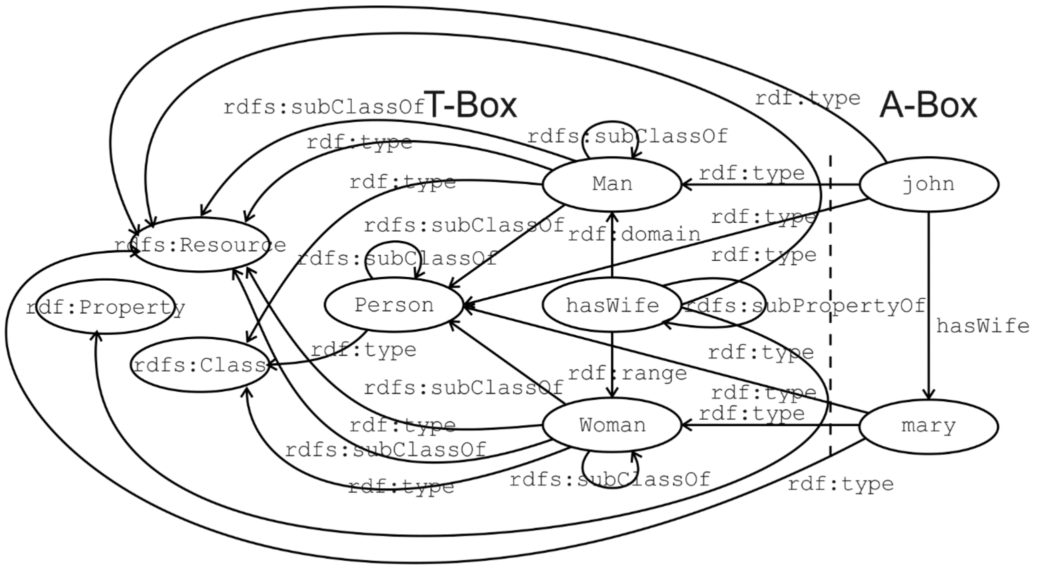


Figure 2: Explicit graph with triples collected in `personsExample-02.ttl`

Please note, that within inferred triples some duplicates might occur. In theory RDF graph is a set of triples, which means that each triple can occur just once. However, the implementation of triple store (and inference algorithms) sometimes causes creation of such duplicates (as in RDF4J implementation).

To make things more clear ontology designers usually declare classes and properties implicitly as in listing 3.

Listing 3: Content of `personsExample-03.ttl`

```
@prefix ex: <http://example.org#> .
@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#> .

ex:Person rdf:type rdfs:Class .
ex:Man rdfs:subClassOf ex:Person .
ex:Woman rdfs:subClassOf ex:Person .
ex:hasWife rdfs:type rdf:Property ;
           rdfs:domain ex:Man ;
           rdfs:range ex:Woman .

ex:john rdf:type ex:Man ;
        ex:hasWife ex:mary .
ex:mary rdf:type ex:Woman .
```

Thus the implicit graph includes declaration of classes and properties as in Figure 3. By convention partial url of resources acting as classes are declares using uppercase, and properties – with lowercase letters at the beginning. Partial urls of instances usually start with lowercase letter.

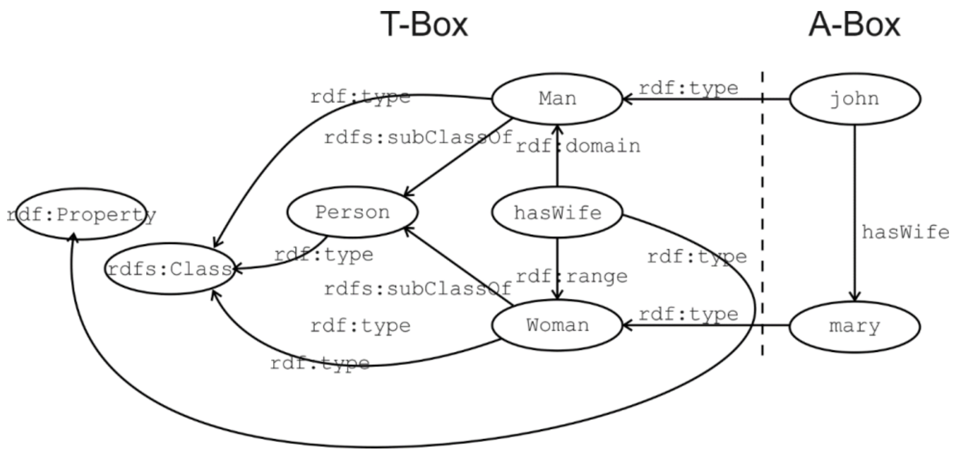


Figure 3: Explicit graph with triples collected in `personsExample-03.ttl`

To check how inferencing works you can try to upload to the triple store the ontology presented in Listing 3. In this example the use of `@base` and custom `@prefix` (according to TURTLE syntax) is demonstrated.

The interesting question is whether the graph is consistent. As we can see, `:susan` has been declared as having `:fatherOf` property. RDF does not deliver any notion which would be used to distinct `:Man` and `:Women` classes. Thus the inference engine will accept such declaration without any warning.

Listing 3: Content of `personsExample-03.ttl`

```
@base <http://example.org/> .
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<Man> rdfs:subClassOf :Person .
:Woman rdfs:subClassOf :Person .
:fatherOf rdfs:domain :Man .

:john a <Man> ;
     :fatherOf :adam .
:robert
     :fatherOf :susan .
:susan
     :fatherOf :john .
:susan a :Woman .
```