Report

In this assignment were implemented basic and more complex queries using SPARQL via the GraphDB environment. Regarding the task 1, the option A was chosen and thus the film ontology was used for the implementation. For the task 2, first the DBPedia was explored and then five queries were constructed from the film ontology provided by DBPedia. Finally, in task 3 the results were compared with and without reasoning on different queries. In the following tables are contained the number of the query, the query in natural language and a screenshot showing the SPARQL query and the result obtained.

**Task 1**

In task 1 are presented the queries implemented in film ontology.

|  |
| --- |
| Q2: Return all films with their title |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  select ?film ?title  where {  ?film rdf:type :Film;  rdfs:label ?title  } |
|  |

|  |
| --- |
| Q3: Is there a film named “Dune” |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  ask {  ?film rdf:type :Film;  rdfs:label "Dune"  } |
|  |

|  |
| --- |
| Q5: Give all information about “Dune” movie released |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  describe ?film  where {  ?film rdfs:label "Dune";  :releaseYear 1984.  } |
|  |

|  |
| --- |
| Q7: Return all writers and the film studios for which they have worked |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  Construct{  ?writer :works ?studio  }  where {  ?film :hasScriptWriter ?writer.  ?film :hasFilmStudio ?studio.  } |
|  |

|  |
| --- |
| Q9: Return all actors and the genre of the movie/movies that they played in |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  Construct{  ?actor :preferGenre ?genre  }  where {  ?film :hasGenre ?genre.  ?film :hasActor ?actor.  } |
|  |

|  |
| --- |
| Q10: Select film studios that were established after 1960 |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?studio  where {  ?studio a :FilmStudio.  ?studio :establishedDate ?date.  FILTER (?date > "1960-01-01"^^xsd:date)  } |
|  |

|  |
| --- |
| Q13: Select titles of the movies and genre for which genre is “Action” or “Family” |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?title ?genre\_name  where {  ?film rdf:type :Film;  rdfs:label ?title.  ?film :hasGenre ?genre.  ?genre rdfs:label ?genre\_name.  filter(?genre\_name = "Action" || ?genre\_name = "Family")  } |
|  |

|  |
| --- |
| Q14: Select name of actors who plays in movies ordered by birthdate (ascending) |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?name\_actor  where {  ?actor a :Actor;  :fullName ?name\_actor;  :dateOfBirth ?birthdate  } order BY ASC(?birthdate) |
|  |

|  |
| --- |
| Q16: Select names of the movies and the number of the actors per movies without including the movie “Dune” |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?movie\_name (count(?actor) as ?number\_of\_actors)  where {  ?movie a :Film;  rdfs:label ?movie\_name.  ?movie :hasActor ?actor  filter(?movie\_name != "Dune")  } Group by ?movie\_name |
|  |

|  |
| --- |
| Q17: List all actors and crews together with the title of movies that they are involved in, ordered by their name |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?person ?movie\_title  where {  ?movie a :Film  {?movie :hasActor ?person.  ?person :fullName ?name\_person.  ?movie rdfs:label ?movie\_title}  union  {?movie :hasCrew ?person.  ?person :fullName ?name\_person.  ?movie rdfs:label ?movie\_title}  }order by ?name\_person |
|  |

**Task 2**

In task 2 in order to access the DBpedia SPARQL endpoint, the command: *service <https://dbpedia.org/sparql>{}* was used. The service keyword instructs a federated query processor a portion of SPARQL query to be executed against a remote SPARQL endpoint. Finally, the ontology that was used in order to implement the following queries, is the Film ontology (dbo:Film, where dbo is the prefix <http://dbpedia.org/ontology/>), as mentioned above.

|  |
| --- |
| Q18: How many South Korean movies are listed in DBPedia |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbo: <http://dbpedia.org/ontology/>  PREFIX dbr: <http://dbpedia.org/resource/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  select (count (distinct ?film) as ?count)  where {  service <https://dbpedia.org/sparql>{  ?film rdf:type dbo:Film.  ?film dbp:country dbr:South\_Korea  }  } |
|  |

|  |
| --- |
| Q19: Find all movies released after year 2000 |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbo: <http://dbpedia.org/ontology/>  PREFIX dbr: <http://dbpedia.org/resource/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select distinct ?film ?date  where {  service <https://dbpedia.org/sparql>{  ?film rdf:type dbo:Film .  ?film dbp:releaseDate|dbp:released|dbo:releaseDate|dbo:premiereDate ?date  filter(?date >= "2001-01-01"^^xsd:date)  }  } |
|  |

|  |
| --- |
| Q20: Find all movies directed by Steven Spielberg where Tom Hanks is not playing |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbo: <http://dbpedia.org/ontology/>  PREFIX dbr: <http://dbpedia.org/resource/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  select distinct ?film ?actor ?director  where {  service <https://dbpedia.org/sparql>{  ?film rdf:type dbo:Film .  ?film dbp:director ?director.  ?film dbo:starring ?actor.  filter(?actor != dbr:Tom\_Hanks && ?director = dbr:Steven\_Spielberg)  }  } |
|  |

|  |
| --- |
| Q21: List all movies and the english titles of the movies order by the title (descending) that are directed, produced and acted by Brad Pitt or Bradley Cooper |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbo: <http://dbpedia.org/ontology/>  PREFIX dbr: <http://dbpedia.org/resource/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  select distinct ?film ?name\_movie\_Brad\_Pitt ?name\_movie\_Bradley\_Cooper  where {  service <https://dbpedia.org/sparql>{  {  ?film dbo:director|dbo:starring|dbo:producer dbr:Brad\_Pitt;  rdfs:label ?name\_movie\_Brad\_Pitt.}  union  {  ?film dbo:director|dbo:starring|dbo:producer dbr:Bradley\_Cooper;  rdfs:label ?name\_movie\_Bradley\_Cooper.  }  FILTER (lang(?name\_movie\_Bradley\_Cooper) = "en" || lang(?name\_movie\_Brad\_Pitt) = "en")  }  }order by Desc(?name\_movie\_Brad\_Pitt ) Desc(?name\_movie\_Bradley\_Cooper) |
|  |

|  |
| --- |
| Q22: Select all movies that are directed by Alfred Hitchcock and have more than 3 actors |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX dbo: <http://dbpedia.org/ontology/>  PREFIX dbr: <http://dbpedia.org/resource/>  PREFIX dbp: <http://dbpedia.org/property/>  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  select ?film (count(?ac) as ?actor)  where {  service <https://dbpedia.org/sparql>{  ?film dbo:starring ?ac.  ?film dbo:director dbr:Alfred\_Hitchcock  }  } GROUP BY ?film  HAVING (?actor > "3"^^xsd:integer) |
|  |

**Task 3**

In task 3, 3 queries were implemented that provide different results with and without the inference in the triple store. The execution of each query were influenced by different entailment pattern (subclass, property domain and range, subproperties).

|  |
| --- |
| Q23: Select all the performers from the movies with their names |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX ex: <http://semantics.id/ns/example#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  select ?performer ?name  where {  ?film :hasPerformer ?performer.  ?performer :fullName ?name  } |
| The entailment pattern, which was used in the query 23, is the RDFS pattern related to subclasses. As can be seen from the two screenshots the results are completely different from each other. According to the first screenshot with inference, all the results are printed compared to the screenshot 2, without inference, in which they are not. The reason why this is happening is that the class Actor is subclass to the class Performer thus without inference, the system is not able to see this class-subclass relationship. Therefore, when the inference is on, all performers including the actors are printed in contrast to inference off where only the performers are printed.  Inference ON    Inference OFF |

|  |
| --- |
| Q24: Select all the crew members from the movies with their names |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX ex: <http://semantics.id/ns/example#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  select ?crew ?name  where {  ?film :hasCrew ?crew.  ?crew :fullName ?name  } |
| The entailment pattern, which was used in the query 24, is the RDFS pattern related to sub-properties. As can be seen from the two screenshots the results are completely different from each other. According to the first screenshot with inference, all the results are printed compared to the screenshot 2, without inference, in which they are not. The reason why this is happening is that the property hasCrew has 3 subproperties named hasDirector, hasScriptWriter and hasComposer and thus without inference, the system is not able to see this relationship between the properties.  Inference ON    Inference OFF |

|  |
| --- |
| Q25: Select all people (actors, directors, writers, etc) with their names and date of birth |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX : <http://semantics.id/ns/example/film#>  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  PREFIX ex: <http://semantics.id/ns/example#>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  select ?person ?name ?birthdate  where {  ?person a foaf:Person.  ?person :fullName ?name.  ?person :dateOfBirth ?birthdate  } |
| The entailment pattern, which was used in the query 25, is the RDFS pattern related to property range. As can be seen from the two screenshots the results are completely different from each other. According to the first screenshot with inference, all the results are printed compared to the screenshot 2, without inference, in which they are not. The reason why this is happening is that when the reasoning is disabled, can not be detected the range Person from the properties in order then to provide the information about the actors, directors, writers, etc.  Inference ON    Inference OFF |