F20BD - Big Data Management

Assignment 2 - Semantic Web

By Group 1

Mamta Sofat (H00236213) BSc Computer Systems

Mobeen Aftab (H00220767) BSc Computer Science

Wai Teng Chong (H00187212) BSc Computer Science

William Russell (H00221650) BSc Computer Systems

Github: https://github.com/MobeenAftab/F20BD
Google Drive: https://drive.google.com/open?id=1I3Lf54Xtgn7TlWamoR_V-uYh_-QG2QR6

1 Task: OBDA Data Access

1.1 Developing R2RML Ontology Mapping

Given the two MySQL tables ‘Airlines’ and ‘Airport’ to be mapped to the selected NASA ontology [1], two classes have been identified from the ontology whose properties semantically describe the same concept captured in the MySQL tables. These classes were discovered using the official NASA docs [2] which visually display the schema relationships between classes, their names and descriptions of the meaning of the captured data.

1.1.1 ETL Process

This task required an ETL process consisting of the following steps:

1. Identifying an existing ontology to map the database. In this case the existing ontology used is ‘The NASA Air Traffic Management Ontology (atmonto)’ [1].
2. Extracting the data from the given MySQL source and converting it into RDF format.
3. Writing queries to map tables and columns to classes and properties to the ontology.
4. Writing SPARQL queries over the developed ontology.

These four steps are a high level overview of the ETL process from discovering an existing ontology that semantically describes the same concepts as the original source dataset, mapping the two properties from different schemas and then querying over this newly mapped ontology using SPARQL. Each individual process is further described in more detail in their respective sections noted above.

1.1.2 Identifying an Existing Ontology

As described in section ‘1.1 Developing R2RML Ontology Mapping’ the NASA ontology [1] contains two classes whose properties can be mapped to the MySQL tables. This section breaks down the similarities between the two schemas and justifies the mapping of the new ontology. The following figures from the NASA ontology are taken from the official NASA documents [2].

Airports

Mapping the Airports table required two classes from the NASA ontology called nas:Airport and gen:PointLocation which is a subclass of nas:Airport.

MySQL Table: Airports	**Ontology Class: nas:Airport, gen:PointLocation**
apid	n/a
name	nas:airportName
city	n/a
iata	nas:iataAirportCode
icao	nas:icaoAirportCode
x	gen:longitude
y	gen:latitude
elevation	gen:altitude

Airline

MySQL Table: Airlines	Ontology Class: nas:AirCarrier
alid	n/a
name	nas:airCarrierName
iata	nas:iataCarrierCode
icao	nas:icaoCarrierCode
callsign	nas:airlineCallsign
country	nas:countryOfRegistry
alias	nas:airCarrierAlias
mode	n/a
active	n/a

1.1.3 Converting MySQL data to RDF (Ontop Mappings)

This section describes the process of mapping between the MySQL table, column to the ontology classes and properties described in section 1.1 Developing R2RML Ontology Mapping.

Nulls and blank values have been accounted for from the source database by having our MySQL queries only return values that are valid. This is shown in the following figures for each class.

A base URN is created to represent each class identified in section 1.1.2 Identifying an Existing Ontology. Every property of that class derives from the base class URN along with the primary key field from the MySQL database. This results in a URN for Airport ‘nas:airport{apid} a nas:Airport . ’and ‘AirCarrier nas:aircarrier{alid} a nas:AirCarrier .’ .

The full list of Ontop mappings is available from the nasa.owl file.

1.1.4 SPARQL Queries over Ontologies

Testing the Ontop mappings by querying the new ontology using SPARQL. These SPARQL queries are designed to return every mapped ontology for each class to test the correctness of the Ontop mapping.

The result would have been higher if these queries had not pattern matched for every valid RDF triple and return objects if they are missing a value.

Airports

The following SPARQL query returned 5,539 results

```
SELECT ?name ?iata ?icao ?long ?lat ?elevation

WHERE{
   ?random nas:airportName  ?name;
           nas:iataAirportCode ?iata;
           nas:icaoAirportCode ?icao;
           gen:longitude ?long;
           gen:latitude ?lat;
           gen:altitude ?elevation .
}
```

Results

	name	iata	icao	long	lat	elevation
1	"RAF Leuchars"	"ADX"	"EGQL"	"-2.8684"^^xsd:double	"56.373"^^xsd:double	"38.0"^^xsd:double
2	"Faya Largeau Airport"	"FYT"	"FTTY"	"19.111"^^xsd:double	"17.917"^^xsd:double	"771.0"^^xsd:double
3	"El Monte Airport"	"EMT"	"KEMT"	"-118.04"^^xsd:double	"34.086"^^xsd:double	"296.0"^^xsd:double
4	"Joshua Mqabuko Nkomo International Airport"	"BUQ"	"FVBU"	"28.618"^^xsd:double	"-20.017"^^xsd:double	"4359.0"^^xsd:double
5	"Buffalo Range Airport"	"BFO"	"FVCZ"	"31.579"^^xsd:double	"-21.008"^^xsd:double	"1421.0"^^xsd:double
6	"Victoria Falls International Airport"	"VFA"	"FVFA"	"25.839"^^xsd:double	"-18.096"^^xsd:double	"3490.0"^^xsd:double
7	"Farah Airport"	"FAH"	"OAFR"	"62.183"^^xsd:double	"32.367"^^xsd:double	"3083.0"^^xsd:double
8	"Harare International Airport"	"HRE"	"FVHA"	"31.093"^^xsd:double	"-17.932"^^xsd:double	"4887.0"^^xsd:double
9	"Pasighat Airport"	"IXT"	"VEPG"	"95.336"^^xsd:double	"28.066"^^xsd:double	"477.0"^^xsd:double
10	"Kee Field"	"I16"	"KI16"	"-81.559"^^xsd:double	"37.6"^^xsd:double	"1783.0"^^xsd:double

Airlines

This query resulted in a total of 84 Airlines that matched this pattern.

```
SELECT ?name ?iata ?icao ?callsign ?country ?alias

WHERE{
  ?random nas:airCarrierName ?name;
          nas:iataCarrierCode ?iata;
          nas:icaoCarrierCode ?icao;
          nas:airlineCallsign ?callsign;
          nas:countryOfRegistry ?country;
          nas:airCarrierAlias ?alias .
}
limit 10
```

Result

	name	iata	icao	callsign	country	alias
1	"Dennis Sky"	"DH"	"DSY"	"DSY"	"IS"	"Dennis Sky Holding"
2	"World Experience Airline"	"W1"	"WE1"	"WEA"	"CA"	"WEA"
3	"BRAZIL AIR"	"GB"	"BZE"	"BRAZIL AIR"	"BR"	"BRAZIL AIR"
4	"KSY"	"KY"	"KSY"	"KSY"	"GR"	"Kreta Sky"
5	"SOCHI AIR"	"CQ"	"KOL"	"SLOW FROG"	"RS"	"SOCHI"
6	"Tom\'s & co airliners"	"&T"	"T&O"	"T&"	"FR"	"Tom\'s air"
7	"LSM Airlines"	"PQ"	"LOO"	"slowbird"	"RS"	"slowbird"
8	"LionXpress"	"C4"	"LIX"	"LIX"	"CM"	"lionXpress"
9	"bmi"	"BD"	"BMA"	"MIDLAND"	"UK"	"bmi British Midland"
10	"Domenican Airlines"	"D1"	"MDO"	"Domenican"	"DR"	"Domenican"

2 Task: Querying OBDA Source

2.1 Query 1: Provide the Names of All UK Airlines

Solution:

```
8  SELECT ?name (str(?country) as ?label)
9
10 WHERE{
11   ?x nas:airCarrierName  ?name;
12      nas:countryOfRegistry ?country .
13 FILTER regex(?country, "^UK")
14 }
```

Result: Total returned 415

	name	label
1	"Air UK"	"UK"
2	"Suckling Airways"	"UK"
3	"Air Atlantique"	"UK"
4	"Air Mercia"	"UK"
5	"Air Medical"	"UK"
6	"Jc royal.britannica"	"UK"
7	"Birmingham European"	"UK"
8	"Air Montegomery"	"UK"
9	"Atlantic Airlines"	"UK"
10	"Astraeus"	"UK"

2.2 Query 2: List the IATA codes that identify both an Airline and an Airport

Solution:

```
SELECT ?iataAirport ?iataAirline

WHERE{
  ?x nas:iataAirportCode  ?iataAirport;
     nas:iataCarrierCode ?iataAirline .
}
```

Result:
The total number of Airports and Airlines that share the same iata code is zero (because they cannot be the same).

Showing 0 to 0 of 0 entries Search: Show 50 entries

iataAirport	iataAirline
No data available in table	

Showing 0 to 0 of 0 entries

2.3 Bermuda Triangle

Query 3: List the names and coordinates of airports located in the Bermuda Triangle. (For simplicity we will model the Bermuda Triangle as a rectangle between 18ºN and 32ºN and 64ºW and 80ºW.)

Solution:

```
SELECT ?name ?long ?lat

WHERE{
  ?x nas:airportName ?name;
     gen:longitude ?long;
     gen:latitude ?lat .
  FILTER(?lat >18 && ?lat <32 && ?long >64 && ?long <80)
}
```

Result:

Given the above values for the Bermuda Triangle there are a total of 78 Airports within the given area.

	name	long	lat
1	"Sahiwal Airport"	"72.392"^^xsd:double	"31.889"^^xsd:double
2	"Mirpur Khas Air Base"	"69.073"^^xsd:double	"25.683"^^xsd:double
3	"Rafiqui Air Base"	"72.283"^^xsd:double	"30.758"^^xsd:double
4	"Faisal Air Base"	"67.118"^^xsd:double	"24.874"^^xsd:double
5	"Kandahar Airport"	"65.848"^^xsd:double	"31.506"^^xsd:double
6	"Faisalabad International Airport"	"72.995"^^xsd:double	"31.365"^^xsd:double
7	"Shahbaz Air Base"	"68.45"^^xsd:double	"28.284"^^xsd:double
8	"Jinnah International Airport"	"67.161"^^xsd:double	"24.907"^^xsd:double
9	"Alama Iqbal International Airport"	"74.404"^^xsd:double	"31.522"^^xsd:double
10	"Walton Airport"	"74.346"^^xsd:double	"31.495"^^xsd:double

3 Task: Querying a related source

The data used for this section was downloaded from (https://data.nasa.gov/ontologies/atmonto/airportInst.ttl) and then imported into Apache Jena Fuseki. Throughout the task, the official document available at https://data.nasa.gov/ontologies/atmontoCore/doc/ was used for assistance.

3.1 Count the number of US airports (include both continental and non-continental airports).

ANSWER: The total number is **2585** including both continental and non-continental airports.

The following Query was used to get this result:

```
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
SELECT   (count( ?subject) as ?count)
WHERE{ {?subject a nas:CONUSairport;}
        UNION
        {?subject a nas:NonCONUSairport;}
}
```

Result:

	count
1	"2585"^^xsd:integer

Problems faced:

At first the query was written to return the name of the continental and non-continental airports and count the number of results returned. The result was 1,385, but after reviewing the query used, an amendment was made by removing “airportName” which then returned 2,585. This result is likely to be more accurate as there may be entries in the database that don’t have a value for “airportName”, as a result of which it wouldn’t be returned using the first query. To ensure this didn’t happen, the query now returned all airports with a value for “CONUSairport” and “NonCONUSairport”.

3.2 Provide the IATA code, ICAO code, and name of all International Airports without a time offset from UTC.

Query:

```
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
SELECT   ?name ?iata ?icao
WHERE{ ?air a nas:InternationalAirport;
        nas:airportName ?name;
        nas:iataAirportCode ?iata;
        nas:icaoAirportCode ?icao.
        FILTER NOT EXISTS { ?air nas:hoursOffsetFromUTC ?code}
}
limit 10
```

3.1.1 Result:

There are **226** entries returned, with the screenshot below showing the result of the above query which limits the results to the first 10 results.

	name	iata	icao
1	"Akureyri"	"AEY"	"BIAR"
2	"Bildudalur Airport"	"BIU"	"BIBD"
3	"Egilsstadir"	"EGS"	"BIEG"
4	"Gjogur Airport"	"GJR"	"BIGJ"
5	"Gr__msey Airport"	"GRY"	"BIGR"
6	"Hornafjordur"	"HFN"	"BIHN"
7	"Husavik"	"HZK"	"BIHU"
8	"Isafjordur"	"IFJ"	"BIIS"
9	"Keflavik International Airport"	"KEF"	"BIKF"
10	"Saudarkrokur"	"SAK"	"BIKR"

Problems faced:

At first, a filter was used to answer the question, with the filter returning all entries with an offset of UTC 0 but this always returned 0 results. Then a trial was done to return the “hoursOffsetFromUTC” just to observe the returned data. The results contained no entries with an offset of 0, hence, it was decided to filter out all values where there were no values for “hoursOffsetFromUTC”.

3.3 Return the name and elevation of the 10 highest airports in New York State.

Query:

```
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix gen: <https://data.nasa.gov/ontologies/atmonto/general#>
SELECT ?name  ?loc

WHERE{
  ?random nas:airportName  ?name;
          nas:locatedInState "NY";
           nas:airportLocation ?elevation.
 ?elevation gen:altitude  ?loc.
}
ORDER BY desc(?loc)
limit 10
```

Result:

Without the “limit” boundary the query returned 19 entries. With the result shown in the screenshot below showing only the top 10 results coming from the query I the screenshot above with the limit in place.

Showing 1 to 10 of 10 entries **Search:**

	name	loc
1	"GREATER BINGHAMTON/EDWIN A LINK FIELD"	"1636.0"^^xsd:float
2	"ITHACA TOMPKINS RGNL"	"1099.0"^^xsd:float
3	"ELMIRA/CORNING RGNL"	"955.0"^^xsd:float
4	"BUFFALO NIAGARA INTL"	"727.0"^^xsd:float
5	"NIAGARA FALLS INTL"	"593.0"^^xsd:float
6	"GREATER ROCHESTER INTL"	"559.0"^^xsd:float
7	"GRIFFISS INTL"	"504.0"^^xsd:float
8	"STEWART INTL"	"491.0"^^xsd:float
9	"WESTCHESTER COUNTY"	"439.0"^^xsd:float
10	"SYRACUSE HANCOCK INTL"	"421.0"^^xsd:float

Problems faced:

To get the altitude of the airports from the data, a check on the NASA official website helped find that each airport has an “altitude” property which is contained in the location of all airports.

In order to only return the airports in New York state, (locatedInState “NY”) was used to ensure that only values that had this value were returned. The results were then ordered by altitude with the highest value at the top, and limited the results to 10 to get the 10 highest airports

4 Task: Querying multiple sources

In this section, it is required to use federated queries to query across two datasets. For this, the NASA dataset (https://data.nasa.gov/ontologies/atmontoCore) and the ontology created during task 1 were used. The data was then exported using the N3 format and while mapping the data all the “Null” and empty data values were removed from the database.

This may not be the best approach as it may remove some data, but it will result in a clean and full dataset. The data was then imported into Apache Jena Fuseki to be queried.

4.1 Return the name, IATA code, and ICAO code for all the airports in the two datasets.

Query:

```
prefix :<https:://example.com/nasa#>
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
SELECT ?nameone ?iata ?icao ?icaotwo ?nametwo ?iatatwo
WHERE{
{ ?x nas:airportName ?nameone;
     nas:iataAirportCode ?iata;
     nas:icaoAirportCode ?icao.
}
  UNION{
  SERVICE <http://localhost:3030/NASA> {
?j nas:airportName ?nametwo ;
     nas:iataAirportCode ?iatatwo;
     nas:icaoAirportCode ?icaotwo.}}
}
```

Result:

Without the “limit” boundary the query returned 10,997 entries. When the limit was used, the 10 results shown in the screenshot below were returned.

Showing 1 to 10 of 10 entries Search: Show 50 entrie

	nameone	iata	icao	icaotwo	nametwo	iatatwo
1	"RAF Leuchars"	"ADX"	"EGQL"			
2	"Faya Largeau Airport"	"FYT"	"FTTY"			
3	"El Monte Airport"	"EMT"	"KEMT"			
4	"Joshua Mq-abuko Nkomo International Airport"	"BUQ"	"FVBU"			
5	"Buffalo Range Airport"	"BFO"	"FVCZ"			
6	"Victoria Falls International Airport"	"VFA"	"FVFA"			
7	"Farah Airport"	"FAH"	"OAFR"			
8	"Harare Inter-national Airport"	"HRE"	"FVHA"			
9	"Pasighat Airport"	"IXT"	"VEPG"			
10	"Kee Field"	"I16"	"KI16"			

Problems faced:

First the new database from task 1 and the given NASA database were queried separately to see how many results would each one give, then these number of results were added together to get 10,997(5,653 and 5,344). This concluded that when both databases are queried together, the result should be same as 10,997.

A trial was conducted one using UNION and one without it to find the result of merging the two databases. When UNION was used to merge two databases the data would return the new dataset first before the NASA data as shown in the result screenshot above. Without the UNION, the system would match each data from the first dataset into the second dataset which will take forever and is not correct.

4.2 Find the number of airports where the name and IATA code are in common across the two datasets.

Query:

```
prefix :<https:://example.com/nasa#>
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
SELECT ?nameone ?iata
WHERE{
{ ?x nas:airportName ?nameone;
     nas:iataAirportCode ?iata;
}
  SERVICE <http://localhost:3030/NASA> {
?j nas:airportName ?nameone ;
     nas:iataAirportCode ?iata;
}
}
```

Result:

Without the “limit” boundary it returned 1,305 entries. When used limit 10 it will only return the top ten results as:

Showing 1 to 10 of 10 entries **Search:**

	nameone	iata
1	"Prince Albert Glass Field"	"YPA"
2	"Thule Air Base"	"THU"
3	"Flinders Island Airport"	"FLS"
4	"Dawadmi Domestic Airport"	"DWD"
5	"Santa Maria Airport"	"RIA"
6	"Keflavik International Airport"	"KEF"
7	"General Santos International Airport"	"GES"
8	"San Fernando Airport"	"SFE"
9	"Santa Rosa Airport"	"SRA"
10	"Linden Airport"	"LDJ"

Problem facing:

After the problem solved from question one, this question was easier to modify from it. For this question it wasn't required to use UNION to separate the data from both databases, instead, the same "object" name was used so when loading the data it knows that it is the same.

4.3 What are the lengths of the 10 highest runways, where the height is taken from the open flights data and the datasets are joined on their IATA code.

Query:

```
prefix gen: <https://data.nasa.gov/ontologies/atmonto/general#>
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
SELECT  ?loc ?len
WHERE{
  {
   ?random nas:iataAirportCode ?iata;
            nas:airportLocation ?elevation;
            nas:hasRunway ?runway.
    ?runway nas:runwayLengthInFeet ?len.
 ?elevation gen:altitude  ?loc.}
    SERVICE <http://localhost:3030/four> {
?j nas:iataAirportCode ?iata;}}ORDER BY desc(?loc)
limit 10
```

Result:

Without the “limit” boundary it returned **1,080** entries. When used limit 10 it will only return the top ten results as:

Showing 1 to 10 of 10 entries Search:

	loc	len
1	"7838.0"^^xsd:float	"7006"
2	"7014.0"^^xsd:float	"8800"
3	"6609.0"^^xsd:float	"8999"
4	"6187.0"^^xsd:float	"11022"
5	"6187.0"^^xsd:float	"13501"
6	"6187.0"^^xsd:float	"8269"
7	"5885.0"^^xsd:float	"10001"
8	"5885.0"^^xsd:float	"4800"
9	"5885.0"^^xsd:float	"7000"
10	"5838.0"^^xsd:float	"4572"

Problem facing:

This question is similar to question 4.2 but the difference here is that here it is required to create a another “object” called “?runway” to load up the “runwayLengthInFeet” from the NASA database and then match with “iataAirportCode” to the database created in task 1.

5 References

[1] M. Keller, R. (2018). *The NASA Air Traffic Management Ontology*. [online] Data.nasa.gov. Available at: https://data.nasa.gov/ontologies/atmonto/?fbclid=IwAR2G1eBAHG3PXiUSOLHq0RUB5R1YzyJSf9WUnNFj-1APf39QcWfm3piOpWA [Accessed 25 Mar. 2019].

[2] M. Keller, R. (2018). *Ontology Documentation*. [online] Data.nasa.gov. Available at: https://data.nasa.gov/ontologies/atmontoCore/doc/ [Accessed 25 Mar. 2019].

6 Bibliography

Bruce, L. (2017). Converting The Guide To Pharmacology Relational Database to RDF Data.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M. and Xiao, G. (2016). Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3), pp.471-487.

GitHub. (2019). *ontop/ontop*. [online] Available at: https://github.com/ontop/ontop/wiki/ObdalibFirstSteps [Accessed 27 Mar. 2019].

Oudani, A., Bahaj, M. and Cherti, I. (2015). Creating an RDF Graph from a Relational Database Using SPARQL. *Journal of Software*, 10(4), pp.384-391.

Protegewiki.stanford.edu. (2019). *Protege Wiki*. [online] Available at: https://protegewiki.stanford.edu/wiki/Main_Page [Accessed 27 Mar. 2019].

Sequeda, J., Arenas, M. and Miranker, D. (2012). On Directly Mapping Relational Databases to RDF and OWL. [online] Available at: https://www2012.universite-lyon.fr/proceedings/proceedings/p649.pdf [Accessed 27 Mar. 2019].

Sequeda, J., Priyatna, F. and Villaz´on-Terrazas, B. (n.d.). Relational Database to RDF Mapping Patterns. *Department of Computer Science, The University of Texas at Austin, OEG-DIA, FI, Universidad Polit´ecnica de Madrid, Spain*. [online] Available at: http://ceur-ws.org/Vol-929/paper9.pdf [Accessed 27 Mar. 2019].

W3.org. (2019). *SPARQL Working Group*. [online] Available at: https://www.w3.org/2009/sparql/wiki/Main_Page [Accessed 27 Mar. 2019].

7 work sharing

Mobeen Aftab and Wai Teng Chong worked on task 1 - 4 equally.
Mamta Sofat helped refactor the document for tasks 3 and 4.