F20BD - Big Data Management

"Coursework Two: Semantic Web Data Integration"

By David Hobbs, Michael King and Jordan Walker

Undergraduate Group 5

Contents

Question 1- ODBA Data Access

1.1 - Ontology Used

For this project, we have used parts of the "atmonto" ontology vocabulary from https://data.nasa.gov/ontologies/atmonto/. We chose to use NASA's vocabulary as it is the most prominent vocabulary related to the 'Transportation' tag in the Linked Open Vocabularies directory[1]. According According to NASA's description; "*the atmonto ontology makes certain infrastructure components of the NAS concrete by including instances of all major NAS structures (ATCCC, ARTCCs, TRACONs, sectors, fixes, routes, airports, etc.), along with information about the airlines and aircraft manufacturers that utilise the NAS.*[2]"

This is beneficial as it shares a significant amount of relatable vocabulary terms with what we aim to transform from the relational Open Flights database. This means that it contained the best match of relationship names to the relationships needed for literals in the Airports and Airlines tables. The result is that integration is easier and ultimately a better practice as we map the current relational model to an existing ontology. More specifically, we have decided to use the 'nas' and 'gen' sections of the vocabulary as they contain only 1057 axioms in total. This is important as using more than 1500 axioms would cause protege to become dramatically slow and difficult to work with.

1.2 - Cleaning up MySQL Data

Loading materialised triples into Jena Fuseki generated a few error messages relating to bad characters, this prompted that some of the data had to be converted to utf8 using the convert command in MySQL when creating the mappings,[3] as shown in Figure 1.

```
MySQL [movielens]> select  convert(binary convert(name using
latin1)using utf8) from airports;
```

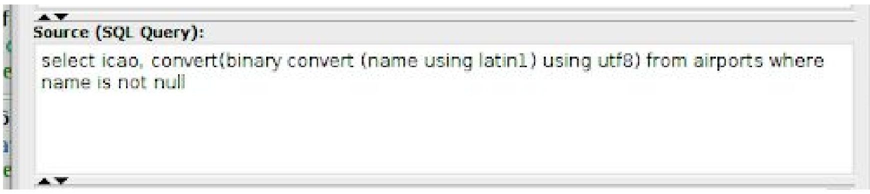


Figure 1: An SQL query that uses the convert function to clean messy data entries.

1.3 - Accounting for null values

A triple store has no support for NULL values and therefore nulls must be accounted for when mapping a relational database to R2RML. To do this, the tables in the MySQL database must be inspected and columns that contain any NULL value must be isolated and dealt with.
A query such as this will show up as NULL values if the column contains them :

```
SELECT iata FROM airports WHERE iata  IS NULL;
```

Classes must be declared before other attributes can be attached to it. Using a non-NULL column as an identifier guarantees that there will be an entry for every airport in the database. The column we decided to use as the identifier was the ICAO code for each airport, as there is a stored value for every airport in the Open Flights database.

```
nas:airport{icao} a nas:Airport .
```

The MySQL query we used to get the respective ICAO codes was:

```
SELECT icao from airports;
```

We then added a separate mapping call which calls the ICAO code, which is used as the Airport's ID, again and then bind IATA codes to ICAO codes where they exist.

```
nas:airport{icao} nas:iataAirportCode {iata} .
```

To handle NULL values, we can use a MySQL query that filters out NULL values. An example of this is given below:

```
SELECT icao, iata FROM airports WHERE iata IS NOT NULL
```

If the mapping comes across an existing entry that there is no IATA code for, an empty space is left. This is why the mappings must be separated out; if IS NOT NULL were to filter out more than one column then rows of data could be lost.

1.4 - Approach to mappings

As the airline table contains a Primary Key ‘ALID’, this was used as the identifier in the mapping to bind all other attributes too. The class `nas:AirCarrier` is declared first and given the label airline, as shown in Figure 2. The label is referred to when appending attributes such as name and country, which is to account for null values.

```
urn:MAPID-cce08439b2bb47bd930e8bd01b0c1697
nas:airline{alid} a nas:AirCarrier .
select alid from airlines where alid >-1
```

Figure 2: AirCarrier Class declaration.

The next stage involves an iterative process attaching single attributes to the Airline label accounting for NULL values. ICAO codes for airlines contained odd characters and numbering schemes which were also accounted for through several steps, as shown in Figure 3.

```
urn:MAPID-1bb102a8a048499881ed62b5f9e7bfbd
nas:airline{alid} nas:airlineCallsign {callsign} .
select alid, callsign from airlines where callsign is not null and callsign != ''
```

```
urn:MAPID-c6ca4eb30afe42128dc77aaf82055829
nas:airline{alid} nas:countryOfRegistry {name} .
SELECT al.alid, cr.name FROM airlines al, countries cr WHERE al.country = cr.code AND al.country is NOT NULL
```

```
urn:MAPID-531c208997c64cf1b45e949a5ae2c5eb
nas:airline{alid} nas:iataCarrierCode {iata} .
select alid, iata from airlines where iata is not null and iata !='' and iata !='-' and alid >-1
```

```
urn:MAPID-3d8bcb8cf6cc4b14bd5178182f48b707
nas:airline{alid} nas:icaoCarrierCode {icao} .
select alid, icao from airlines where icao is not null and alid >-1 and icao!='' and icao !='-';
```

Figure 3: Callsign, country, IATA and ICAO code mappings.

The Airport class is handled in a similar way to Airlines, except there are no NULL values for ICAO codes, so this can be used as an identifier for the airport label. The Airport class is declared first and attributes added later to account for NULL values, as described above. The initial declaration is shown in Figure 4.

```
urn:MAPID-5eef990c26914d289eb50b326d0ddb92
nas:airport{icao} a nas:Airport .
select icao from airports where icao is not null
```

Figure 4: Airport class declaration with airport label and icao identifier

The next stage involves an iterative process attaching single attributes to the airport label to account for null values, with seperate mappings made for these attributes, as shown in Figure 5.

```
urn:MAPID-95b37971a789472f963961aaf22d48b5
nas:airport{icao} nas:iataAirportCode {iata} .
select icao, iata from airports where iata is not null and icao is not null
```

```
nas:airport{icao} rdfs:label {name} .
select icao, name from airports where name is not null
```

```
urn:MAPID-0806a7a96a454dbfafc92a649513d19b
nas:airport{icao} nas:icaoAirportCode {icao} .
select icao from airports where icao is not null
```

```
urn:MAPID-2765c66587ac43a69859df4a859346b9
nas:airport{icao} nas:airportName {name} .
select icao, name from airports where name is not null
```

Figure 5: Airport IATA, ICAO, rdfs label, and name mappings

The final mapping, shown in Figure 6, is different as it uses a class attribute `airportLocation` to link to the Point Location class that contains location data. It still uses the icao number to link the airports together. Location data must be contained in a PointLocation class, it cannot be included with each airport.

```
urn:MAPID-d947d7ba6b7c46f6ada944a408766ebe
nas:airport{icao} nas:airportLocation nas:{icao}coordinates .
select icao from airports where icao is not null
```

Figure 6: AirportLocation mapping

The class declaration for `PointLocation` uses the ICAO identifier to match airports with Airport objects, which is shown in Figure 7. This class must be used to store location data to reflect the structure used by NASA's atmonto ontology.

```
urn:MAPID-a8c3f2bcfeb747c689c3d52860274787
nas:{icao}coordinates a gen:PointLocation .
select icao from airports where icao is not null
```

Figure 7: PointLocation class declaration with icao identifier

As Figure 8 highlights, location data is added as separate mappings to account for null values, types are added to restrict data types allowed.

```
urn:MAPID-15364c1a72704c3a9a9eaa78fd05c672
nas:{icao}coordinates gen:latitude {y}^^xsd:double .
select icao, y from airports where icao is not null and y is not null
```

```
urn:MAPID-e40a53a10f1c4c978bf9457fe47d280b
nas:{icao}coordinates gen:longitude {x}^^xsd:double .
select icao, x from airports where icao is not null and x is not null
```

```
urn:MAPID-05f9db071d5847f99801ff62b9af4b04
nas:{icao}coordinates gen:altitude {elevation}^^xsd:integer .
select icao, elevation from airports where icao is not null and elevation is not null
```

Figure 8: Latitude, longitude, altitude mappings

10 airports with classes - Figure 9

```
PREFIX cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?class
where{
     ?airport cw2:airportName ?name;
          a ?class;
}limit 10
```

name	class
"Goroka Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Madang Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Mount Hagen Kagamuga Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Nadzab Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Port Moresby Jacksons International Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Wewak International Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Narsarsuaq Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Godthaab / Nuuk Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Kangerlussuaq Airport"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>
"Thule Air Base"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#NASfacility>

Figure 9: Query results for 10 airports with classes

10 Airlines with classes - Figure 10

```
PREFIX cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?class
WHERE {
     ?airport cw2:airCarrierName ?name;
          a ?class;

}LIMIT 10
```

name	class
"135 Airways"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"1Time Airline"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"223 Flight Unit State Airline"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"224th Flight Unit"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"247 Jet Ltd"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"3D Aviation"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"40-Mile Air"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"4D Air"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"611897 Alberta Limited"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>
"Ansett Australia"^^xsd:string	<https://data.nasa.gov/ontologies/atmonto/NAS#AirCarrier>

Figure 10: Query results for 10 airlines with classes

Question 2- Querying Our OBDA Source

In this question, we queried our OBDA source that had been developed during Question 1 of this assignment. The queries are executed using SPARQL and are contain the following prefixes for the queries to be executed:

```
PREFIX loc: <https://data.nasa.gov/ontologies/atmonto/general#>
PREFIX cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

Where after the prefixes are declared the query to be executed is described.

2.1- Names of UK Airports

```
SELECT *
WHERE {
     ?airline cw2:airCarrierName ?name;
          cw2:countryOfRegistry ?country.
     FILTER REGEX (?country, 'United Kingdom').
}
```

Sample of answers shown in Figure 11. **Total rows 415.**

airline	name	country
<https://data.nasa.gov/ontologies/atmonto/NAS#airline4>	"2 Sqn No 1 Elementary Flying Training School...	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline8>	"247 Jet Ltd"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline16>	"Army Air Corps"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline52>	"Avcard Services"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline59>	"Air Charter Service"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline77>	"Aero Dynamics"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline105>	"Air Atlantique"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline112>	"Astraeus"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline138>	"Air Partner"^^xsd:string	"United Kingdom"^^xsd:string
<https://data.nasa.gov/ontologies/atmonto/NAS#airline143>	"Air Data"^^xsd:string	"United Kingdom"^^xsd:string

Figure 11: Sample of answers to Question 2.1.

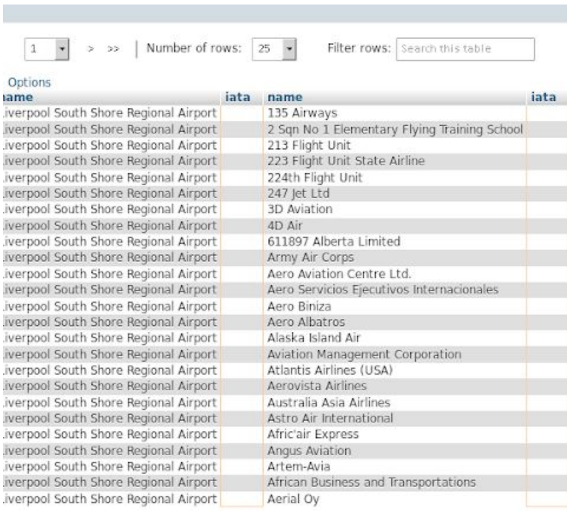
2.2- IATA Codes that Identify Both an Airport and an Airline

This query posed much confusion when trying to create a SPARQL query that would return a meaningful answer.

To test if any such examples of matching data existed in the Open Flights database, we queried the MySQL database directly where we would pull values from the database on where the airline's IATA code is equal to the airport's IATA code. This was done using the following query:

```
SELECT airports.name, airports.iata, airlines.name,
airlines.iata FROM airlines, airports WHERE airports.iata =
airlines.iata
```

This produced the following output, shown in Figure 12, where the lack of IATA code in both airlines and airports resulted in all airlines with no IATA code being connected to Liverpool South Shore Regional Airport. The query contained a total of 4625 results.



1 > >> | Number of rows: 25 Filter rows: Search this table

Options

ame	iata	name	iata
iverpool South Shore Regional Airport		135 Airways	
iverpool South Shore Regional Airport		2 Sqn No 1 Elementary Flying Training School	
iverpool South Shore Regional Airport		213 Flight Unit	
iverpool South Shore Regional Airport		223 Flight Unit State Airline	
iverpool South Shore Regional Airport		224th Flight Unit	
iverpool South Shore Regional Airport		247 Jet Ltd	
iverpool South Shore Regional Airport		3D Aviation	
iverpool South Shore Regional Airport		4D Air	
iverpool South Shore Regional Airport		611897 Alberta Limited	
iverpool South Shore Regional Airport		Army Air Corps	
iverpool South Shore Regional Airport		Aero Aviation Centre Ltd.	
iverpool South Shore Regional Airport		Aero Servicios Ejecutivos Internacionales	
iverpool South Shore Regional Airport		Aero Biniza	
iverpool South Shore Regional Airport		Aero Albatros	
iverpool South Shore Regional Airport		Alaska Island Air	
iverpool South Shore Regional Airport		Aviation Management Corporation	
iverpool South Shore Regional Airport		Atlantis Airlines (USA)	
iverpool South Shore Regional Airport		Aerovista Airlines	
iverpool South Shore Regional Airport		Australia Asia Airlines	
iverpool South Shore Regional Airport		Astro Air International	
iverpool South Shore Regional Airport		Afric'air Express	
iverpool South Shore Regional Airport		Angus Aviation	
iverpool South Shore Regional Airport		Artem-Avia	
iverpool South Shore Regional Airport		African Business and Transportations	
iverpool South Shore Regional Airport		Aerial Oy	

Figure 12: MySQL results of airport and airline shared IATA codes.

Feeling we did not fully understand the question, the definitions of what airline and airport IATA codes were used for was researched.

IATA airline:
IATA codes are using for indication of airports in air schedules, reservation´s systems, coding baggages etc. These codes gives out International Air Transport Association[4].

IATA airport:
The International Air Transport Association's (IATA) Location Identifier is a unique 3-letter code (also commonly known as IATA code) used in aviation and also in logistics to identify an airport. For example, JFK is the IATA code for New York's John F. Kennedy International Airport[5].

The main issue is that an IATA code is given to both an airline and an airport, however, an airlines IATA contains only two characters where the airports IATA contains three[4] [5].

For an airline and airport to share the same code, logically, would mean that there has been a technical error in the storage of a certain value. If there was a shared code in reality then it would lead to significant confusion, as it would lead to ambiguity of what exactly the particular IATA code belongs to. This could result in impossible tasks and routes being made where instructions are given to an airline to fly to another airline or airport.

This SPARQL query tries to find a match between IATA codes of airlines and IATA codes of airports using filter to match both objects. As expected, no results are returned.

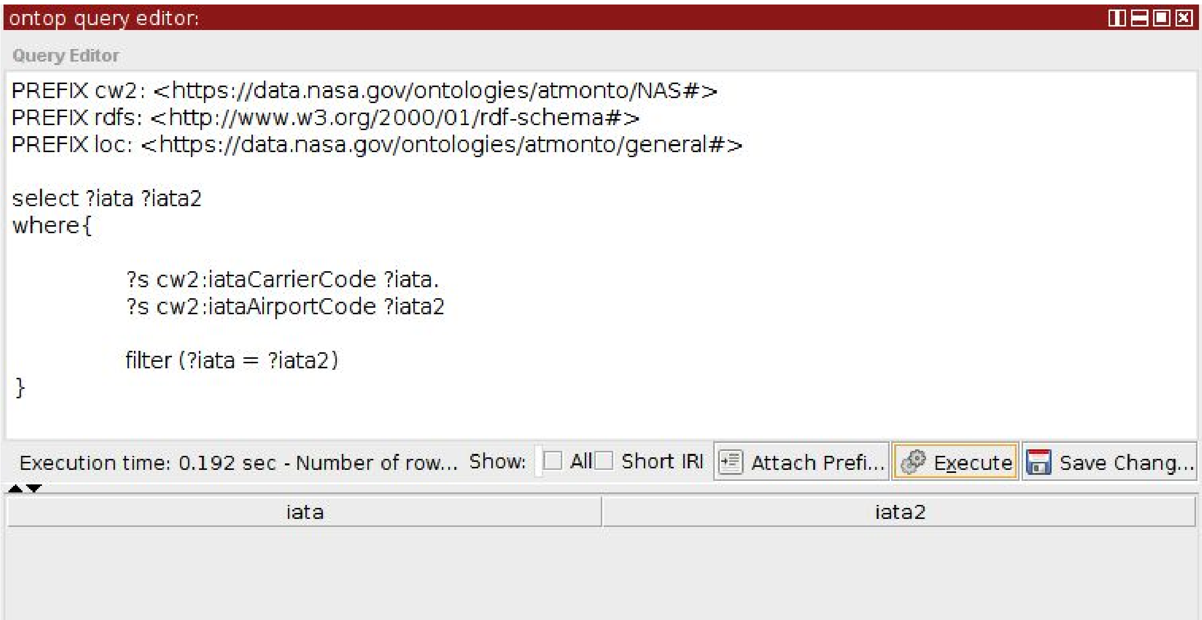


Figure 13: SPARQL query looking for matches of IATA codes.

2.3- The Bermuda Rectangle

To find the occurences of airports in the Bermuda Triangle, we queried all the values between 18ºN and 32ºN and 64ºW and 80ºW. To do this successfully, we had to convert the Western degrees values to Eastern, by making them negative from the middle point of 180ºE and 180ºW, as longitude values are calculated through the Eastern value. A sample value of the answers is given in Figure 13.

```
select name, lat, long
where {

        ?airport cw2:airportName ?name;
        cw2:airportLocation ?loc.
        ?loc loc:latitude ?lat;
        loc:longitude ?long;


        FILTER (?long >= -80.0 && ?long <=-64.00).
        FILTER (?lat <= 32.0 && ?lat >=18.00).

}
```

Total rows 82

Sample :

name	lat	long
"Alberto Delgado Airport"^^xsd:string	"21.788"^^xsd:double	"-79.997"^^xsd:double
"Abel Santamaria Airport"^^xsd:string	"22.492"^^xsd:double	"-79.944"^^xsd:double
"Gerrard Smith International Airport"^^x...	"19.687"^^xsd:double	"-79.883"^^xsd:double
"Sancti Spiritus Airport"^^xsd:string	"21.97"^^xsd:double	"-79.443"^^xsd:double
"South Bimini Airport"^^xsd:string	"25.7"^^xsd:double	"-79.265"^^xsd:double
"Las Brujas Airport"^^xsd:string	"22.621"^^xsd:double	"-79.147"^^xsd:double
"Maximo Gomez Airport"^^xsd:string	"22.027"^^xsd:double	"-78.79"^^xsd:double
"Grand Bahama International Airport"^^...	"26.559"^^xsd:double	"-78.696"^^xsd:double
"Cayo Coco Airport"^^xsd:string	"22.513"^^xsd:double	"-78.511"^^xsd:double
"Abaco I Walker C Airport"^^xsd:string	"27.267"^^xsd:double	"-78.4"^^xsd:double
"Negril Airport"^^xsd:string	"18.343"^^xsd:double	"-78.332"^^xsd:double
"Jardines Del Rey Airport"^^xsd:string	"22.461"^^xsd:double	"-78.328"^^xsd:double
"Florida Airport"^^xsd:string	"21.5"^^xsd:double	"-78.203"^^xsd:double
"San Andros Airport"^^xsd:string	"25.054"^^xsd:double	"-78.049"^^xsd:double
"Sangster International Airport"^^xsd:st...	"18.504"^^xsd:double	"-77.913"^^xsd:double
"Chub Cay Airport"^^xsd:string	"25.417"^^xsd:double	"-77.881"^^xsd:double
"Ignacio Agramonte International Airport...	"21.42"^^xsd:double	"-77.848"^^xsd:double
"Great Harbour Cay Airport"^^xsd:string	"25.738"^^xsd:double	"-77.84"^^xsd:double
"Andros Town Airport"^^xsd:string	"24.698"^^xsd:double	"-77.796"^^xsd:double

Figure 14: Sample results of airports in the Bermuda Triangle.

Question 3- Querying A Related Source

For this question, we used the NASA dataset that correlates information regarding airports in and outside of the US. By use of the triplestore Apache Jena Fuseki, we then queried the dataset through a localhost connection which gave us the ability to use an online SPARQL editor to perform said queries.

For all the following queries, the prefixes given so the SPARQL queries can be correctly executed are;

```
prefix rdfs: <http://rdaregistry.info/termList/fontSize/>
prefix gen:
<https://data.nasa.gov/ontologies/atmonto/general#>
prefix nas: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
```

Where after the prefixes are declared the query to be executed is described.

3.1- Number Of US Airports

For this query, we used the class type to find the total number of airports. This simple query then completes a count of all that hold the class type in the NAS ontology of "`CONUSairport`". To complete the query, however, as initially this only counts airports that are part of mainland US a union is added to count those that have the class type "`NonCONUSairport`". This encompasses all airports that make up the US as this is the NASA datasets all Non-US airports are classed as international.

SPARQL Query-

```
SELECT (count(nas:CONUSairport)as ?count)
WHERE
  {
      {
      ?subject a nas:CONUSairport
      }union{
        ?subject a nas:NonCONUSairport
       }
}
```

Results Of Query-

The result from running this query gave a total of “2585”^^xsd: integer(2585) as a total of the union of the two classes, as shown in Figure 14.

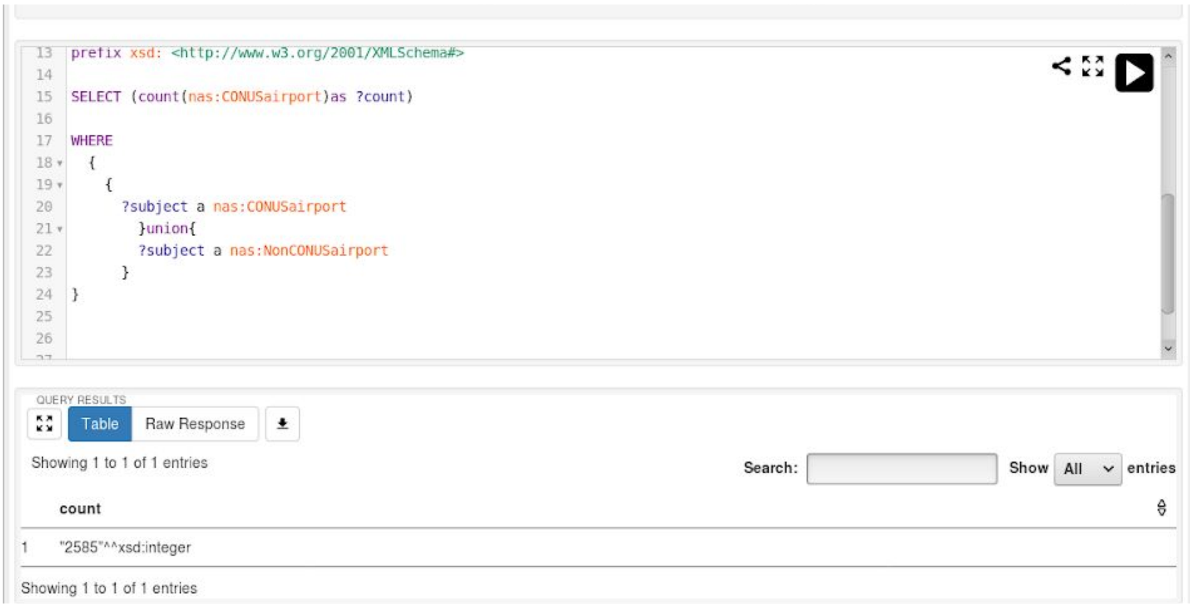


Figure 15: Result for number of US based airports query.

Running the query on a single class by not using the union receives the resulting count;

- 2326 airports of type: “CONUSairport”
- 259 airports of type “NonCONUSairport”

3.2- International Airports Without Time Offset From UTC

This query aimed to return all the airports with there respecting IATA, ICAO codes and the airports' name that did not have a time offset from UTC. To begin this query we set as `?subject nas:airportName` and extended that by returning the IATA and ICAO code associated with that airport. To extend this query further we add in an optional statement which also “grabs” the UTC offset associated with the airport. To complete the final query a Filter statement is used that uses the !bound clause on ?UTC which essentially returns all values where the `hoursOffsetFromUTC` are NULL. By doing this we can then return all values which have no offset whereas if we didn’t use this filter we would get all airports in the NASA dataset with all the requested labels.

SPARQL Query-

```
SELECT ?iata ?icao ?name
WHERE {
     ?subject nas:airportName ?name;
     nas:iataAirportCode ?iata;
     nas:icaoAirportCode ?icao;
     OPTIONAL{?subject nas:hoursOffsetFromUTC ?UTC;}
  FILTER(!bound(?UTC))
}
```

Results Of Query-

Without the filter, the query returns a total of 5344 airports and there name, IATA and ICAO code. By using the filter we return a total of 226 entries. This is shown in Figure 15.

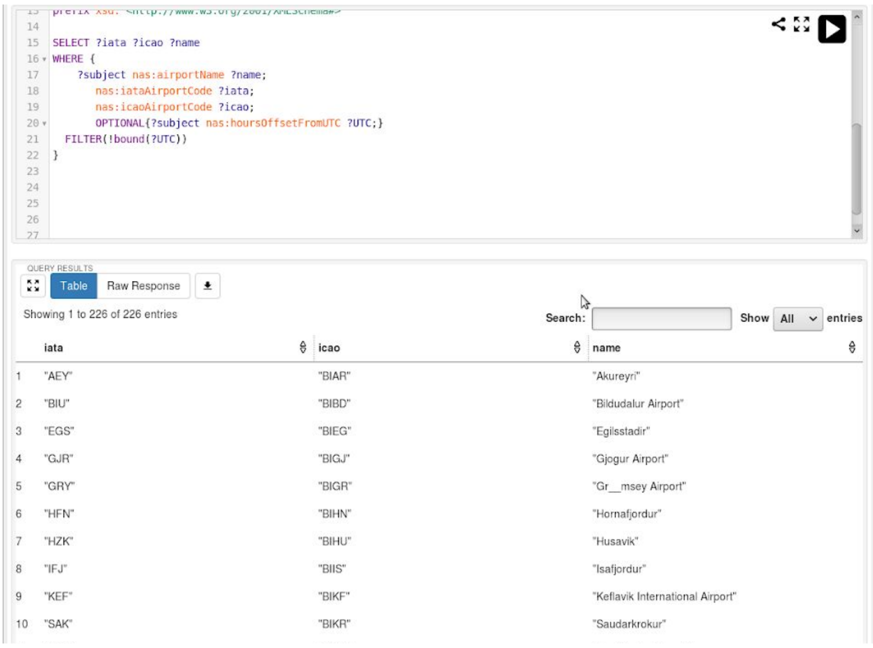


Figure 16: Airports without time offset from UTC..

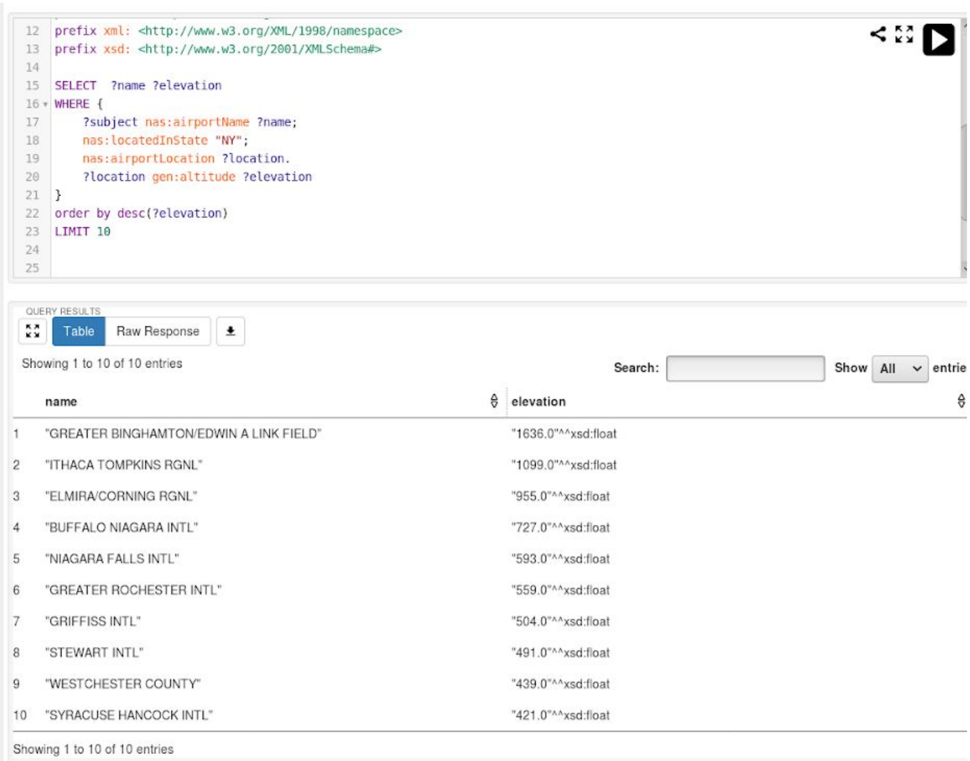
3.3- Highest Airports In New York State

This query aimed to return the highest airports in New York State by their elevation. To complete this query we first had to identify all the airports in the NASA dataset by `nas:airportName`, combined with all the airports that are situated in the state of New York by selecting `nas:locatedInState "NY"`. To find the elevation of the airport then required that we made a connection between data in the nas and gen ttls. This works primarily by getting the location from the nas dataset and then matching that location to the gen on the location where we could then retrieve the altitude of the airport. To return the query results in order and with a limit on ten as asked for in the specification an order by desc was included after the query where clause as well as a limit set to only return the top 10 results.

<u>SPARQL Query-</u>

```
SELECT  ?name ?elevation
WHERE {
      ?subject nas:airportName ?name;
      nas:locatedInState "NY";
      nas:airportLocation ?location.
      ?location gen:altitude ?elevation
}
order by desc(?elevation)
LIMIT 10
```

<u>Results Of Query-</u>



```
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT  ?name ?elevation
WHERE {
    ?subject nas:airportName ?name;
    nas:locatedInState "NY";
    nas:airportLocation ?location.
    ?location gen:altitude ?elevation
}
order by desc(?elevation)
LIMIT 10
```

QUERY RESULTS

Table Raw Response

Showing 1 to 10 of 10 entries

Search: Show All entries

	name	elevation
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

Showing 1 to 10 of 10 entries

Figure 17: The highest airports in New York state, based on elevation..

Question 4- Querying Multiple Sources

4.1 - Name, IATA and ICAO of all Airports

The main aspect of this query was using the SERVICE function inside of SPARQL, which enabled us to query over numerous datasets that we had uploaded to Apache Jena. In the following examples, the converted mappings of the Open Flights relational database is saved in the `/cw2` ontology, whereas the the NASA turtle data is stored in the `/nasa` dataset. The combined values for all airports is shown in Figure 17.

```
prefix cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?icao ?iata
WHERE {
  {SERVICE<http://localhost:3030/cw2/>
     {?airport cw2:airportName ?name.
           optional{?airport cw2:icaoAirportCode ?icao.}
                    optional{?airport cw2:iataAirportCode
?iata.}
           }}
  UNION
  {SERVICE <http://localhost:3030/nasa/>
     {?airport2 cw2:airportName ?name.
               optional{?airport2 cw2:icaoAirportCode
?icao.}
                    optional{?airport2 cw2:iataAirportCode
?iata.}
}               }}
```

Showing 1 to 1,000 of 14,016 entries   Search: Show 1000 entries

	name	icao	iata
1	"RAF Barkston Heath"	"EGYE"	
2	"RAF Leuchars"	"EGQL"	"ADX"
3	"RAF Scampton"	"EGXP"	
4	"Utirik Airport"	"03N"	"UTK"
5	"Ocean Reef Club Airport"	"07FA"	"OCA"
6	"Glasgow Industrial Airport"	"07MT"	
7	"Oconomowoc Airport"	"0WI8"	
8	"Sky Ranch At Carefree Airport"	"18AZ"	
9	"Mobile Airport"	"1AZ0"	
10	"Earl L. Small Jr. Field/Stockmar Airport"	"20GA"	
11	"The Farm Airport"	"24SC"	
12	"Benson Airstrip"	"2XS8"	
13	"Aleknagik / New Airport"	"5A8"	"WKK"

Figure 18: Sample results of the IATA and ICAO values associated with each airport.

4.2 - Number of Airports Where Name and IATA Codes are Common in Both Datasets

To perform this query, we utilised the `HAVING` function. `HAVING` operates over grouped solution sets, in the same way that `FILTER` operates over un-grouped ones in a traditional MySQL query. `HAVING` expressions have the same evaluation rules as projections from grouped queries, as described in the following section[6]. We believed that the `HAVING` clause was the best way to group values that were in common on both datasets under the same vocabulary tag. The results for the queried common airports is shown in Figure 18.

```
prefix cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?iata ?name1 ?iata1
WHERE {
     {?airport cw2:airportName ?name;
          cw2:iataAirportCode ?iata.
     }

  {SERVICE <http://localhost:3030/nasa/>
     {?airport2 cw2:airportName ?name1;
                    cw2:iataAirportCode ?iata1.}
  }
} HAVING  (?iata = ?iata1 && ?name = ?name1)
```

```
prefix cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix loc: <https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?iata ?name1 ?iata1
WHERE {
    {?airport cw2:airportName ?name;
              cw2:iataAirportCode ?iata.
     }

  {SERVICE <http://localhost:3030/nasa/>
    {?airport2 cw2:airportName ?name1;
               cw2:iataAirportCode ?iata1.}
  }
} HAVING  (?iata = ?iata1 && ?name = ?name1)
```

QUERY RESULTS

Table Raw Response

Showing 1 to 1,000 of 1,305 entries

Search: Show 1000 entries

	name	iata	name1	iata1
1	"Utirik Airport"	"UTK"	"Utirik Airport"	"UTK"
2	"Ulawa Airport"	"RNA"	"Ulawa Airport"	"RNA"
3	"Uru Harbour Airport"	"ATD"	"Uru Harbour Airport"	"ATD"
4	"Auki Airport"	"AKS"	"Auki Airport"	"AKS"
5	"Choiseul Bay Airport"	"CHY"	"Choiseul Bay Airport"	"CHY"
6	"Ballalae Airport"	"BAS"	"Ballalae Airport"	"BAS"
7	"Fera/Maringe Airport"	"FRE"	"Fera/Maringe Airport"	"FRE"
8	"Babanakira Airport"	"MBU"	"Babanakira Airport"	"MBU"
9	"Ngorangora Airport"	"IRA"	"Ngorangora Airport"	"IRA"
10	"Santa Cruz/Graciosa Bay/Luova Airport"	"SCZ"	"Santa Cruz/Graciosa Bay/Luova Airport"	"SCZ"
11	"Munda Airport"	"MUA"	"Munda Airport"	"MUA"
12	"Nusatupe Airport"	"GZO"	"Nusatupe Airport"	"GZO"
13	"Mono Airport"	"MNY"	"Mono Airport"	"MNY"
14	"Rennell/Tingoa Airport"	"RNL"	"Rennell/Tingoa Airport"	"RNL"
15	"Sege Airport"	"EGM"	"Sege Airport"	"EGM"
16	"Santa Ana Airport"	"NNB"	"Santa Ana Airport"	"NNB"
17	"Santa Ana Airport"	"CRC"	"Santa Ana Airport"	"CRC"

Figure 19: Sample results of airports recorded in both datasets.

4.3 - Length of the 10 Highest Runways

To record the length of the two longest runways, we produced two queries that ultimately present the same answers. Method 2 follows the same HAVING clause that is familiar from Question 4.2. However, Method 1 makes use of a bind on the IATA code for one of the datasets being queried, which improves the running time of the query significantly. The list of the ten highest runways is shown in Figure 19.

Method 1:

```
prefix cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?iatacode ?length ?alt
WHERE {
  {SERVICE <http://localhost:3030/cw2/>
     {?airport cw2:airportName ?name.
     ?airport cw2:iataAirportCode ?iata;

    cw2:airportLocation ?loc.
    ?loc loc:altitude ?alt.

     bind((?iata) as ?iatacode)

     }}

  {SERVICE <http://localhost:3030/nasa/>
     {
     ?airport2 cw2:iataAirportCode ?iatacode.
                          ?airport2 cw2:hasRunway ?assoc.
           ?assoc cw2:runwayLengthInFeet ?length.

                }}
} order by desc (?alt) limit 10
```

Method 2:

```
prefix cw2: <https://data.nasa.gov/ontologies/atmonto/NAS#>
prefix loc:
<https://data.nasa.gov/ontologies/atmonto/general#>

SELECT ?name ?iatacode ?iata ?length ?alt
WHERE {
  {SERVICE <http://localhost:3030/cw2/>
     {?airport cw2:airportName ?name.
     ?airport cw2:iataAirportCode ?iata;

    cw2:airportLocation ?loc.
    ?loc loc:altitude ?alt.
     }}

  {SERVICE <http://localhost:3030/nasa/>
     {
     ?airport2 cw2:iataAirportCode ?iatacode.
                          ?airport2 cw2:hasRunway ?assoc.
          ?assoc cw2:runwayLengthInFeet ?length.

               }}
} HAVING (?iata = ?iatacode) order by desc (?alt) limit 10
```

	name	iatacode	length	alt
1	"Aspen-Pitkin Co/Sardy Field"	"ASE"	"7006"	"7820"^^xsd:integer
2	"Flagstaff Pulliam Airport"	"FLG"	"8800"	"7014"^^xsd:integer
3	"Grand Canyon National Park Airport"	"GCN"	"8999"	"6609"^^xsd:integer
4	"Lanzhou Zhongchuan Airport"	"LHW"	"5007"	"6388"^^xsd:integer
5	"Lanzhou Zhongchuan Airport"	"LHW"	"2610"	"6388"^^xsd:integer
6	"Lanzhou Zhongchuan Airport"	"LHW"	"5001"	"6388"^^xsd:integer
7	"Lanzhou Zhongchuan Airport"	"LHW"	"2520"	"6388"^^xsd:integer
8	"City of Colorado Springs Municipal Airpor t"	"COS"	"8269"	"6187"^^xsd:integer
9	"City of Colorado Springs Municipal Airpor t"	"COS"	"13501"	"6187"^^xsd:integer
10	"City of Colorado Springs Municipal Airpor t"	"COS"	"11022"	"6187"^^xsd:integer

Showing 1 to 10 of 10 entries

Figure 20: Ten highest runways in combined IATA code airports.

Group Contribution

Throughout this project, a full group contribution has been given to make sure that the tasks stated through the specification have been delivered on time and is to the standards expected of undergraduate students.
As stated in the project description all members should contribute in all areas regarding the coursework and as a group, we took this advice and all worked together when possible to complete the assignment.
In extension to this as everyone has different schedules, it was the responsibility of the group member to contribute, thus if a group member couldn't be present when work was carried out on the assignment they were informed and updated so they could continue contributing as we progressed.
In summary, a full group effort was established for this project and all group members individually aided in the final deliverable of this assignment.

References

[1] Linked Open Vocabularies (2019)
https://lov.linkeddata.es/dataset/lov/terms?&tag=Transport
(Accessed: 12th March 2019)

[2] Keller, R. M. (2018). *The NASA Air Traffic Management Ontology (atmonto),*
Available at: https://data.nasa.gov/ontologies/atmonto/gen
(Accessed: 23rd March 2019).

[3] Salonen, J. (2019). *MySQL Charset/Collate*. [online] Mysql.rjweb.org.
Available at: http://mysql.rjweb.org/doc.php/charcoll#fixes_for_various_cases
(Accessed: 20th March 2019).

[4] Doprava v Praxi. (2009). *IATA codes airlines,*
Available at: http://www.doprava.vpraxi.cz/eng_zkratky_aerolinii.html
(Accessed: 23rd March 2019).

[5] One World - Nations Online. (2011). *IATA 3-Letter Codes of Airports.*
Available at: https://www.nationsonline.org/oneworld/IATA_Codes/airport_code_list.htm
(Accessed: 24th March 2019).

[6] W3.org. (2013). *SPARQL 1.1 Query Language.*
Available at: https://www.w3.org/TR/sparql11-query/#rHavingClause
(Accessed: 24th March 2019).