Semantic Web application for a higher education institution

Martiros Yeghiazaryan

Report for Web of Data

1. Data

This is an RDF graph that represents a university system. It contains information about people (students and teachers), courses, programs, and promotions/scholarships in the Université Côte d'Azur. The graph uses several prefixes to define namespaces for common RDF vocabularies, such as rdf, rdfs, and owl.

The graph defines four main classes: Person, Teacher, Student, Educational Entity, Program, Course, and Promotion. A person is the superclass of Teacher and Student, and Educational Entity is the superclass of Program, Course, and Promotion.

There are several object properties defined in the graph, including teaches, taughtBy, enrolledIn, partOfProgram, and awardedTo. There are also several datatype properties defined in the graph, such as emailAddress, dateOfBirth, hasEnrollmentStatus, courseTitle, courseDescription, teacherName, and studentName.

The graph contains information about several individuals, including Ava, Ethan, Liam, Isabella, Amelia, Harper, Daniel, Elijah, Emily, Samuel, Abigail, Alexander, William, Olivia, DrSmith, and DrJohnson. These individuals are instances of either Teacher or Student.

1.1 Students

There are 10 students in the dataset with various ages, genders, and enrollment statuses. The students are enrolled in a range of courses, including Psychology, Mathematics, Computer Programming, Literature, History, Physics, and Chemistry. Some students receive scholarships based on their merits or academic excellence.

1. Ethan: 22-year-old male, enrolled in Psychology 101, Math 102, and Computer Programming.
2. Liam: 21-year-old male, enrolled in Math 101, Math 102, Computer Programming, and Modern History.
3. Amelia: 23-year-old female, enrolled in Literature 101 and World Literature, receives a Merit Scholarship.
4. Daniel: 24-year-old male, enrolled in History 101, Modern History, and World Literature.
5. Emily: 20-year-old female, enrolled in Chemistry 101 and Math 102.
6. Samuel: 24-year-old male, enrolled in Physics 101, Modern History, and Physics 102, receives a Merit Scholarship.
7. Abigail: 22-year-old female, enrolled in Psychology 101, Modern History, and Chemistry 102.

8. Alexander: 23-year-old male, enrolled in Computer Programming, Math 101, Math 102, and Physics 102, receives an Academic Excellence Scholarship.
9. William: 21-year-old male, enrolled in Literature 101, World Literature, and Chemistry 102.
10. Olivia: 20-year-old female, enrolled in History 101, World Literature, and Physics 102.

1.2 Teachers

There are six teachers in the dataset with varying experience, subject expertise, and popularity ratings.

1. Isabella: 32-year-old female, teaches Literature, has 8 years of experience, and a popularity rating of 4.5.
2. Harper: 39-year-old female, teaches History, has 12 years of experience, and a popularity rating of 3.8.
3. Elijah: 45-year-old male, teaches Computer Science, has 20 years of experience, and a popularity rating of 4.0.
4. Ava: 35-year-old female, teaches Mathematics, has 10 years of experience, and a popularity rating of 4.2.
5. Dr. John Smith: 45-year-old male, teaches Physics, has 17 years of experience, and a popularity rating of 4.1.
6. Dr. Jane Johnson: 40-year-old female, teaches Psychology, has 15 years of experience, and a popularity rating of 3.9.

1.3 Courses

There are several courses in the dataset, including Math 101 (Calculus I), Psychology 101 (Introduction to Psychology), Physics 101 (Introduction to Physics), Chemistry 101 (Introduction to Chemistry), Literature 101 (Introduction to Literature), and History 101 (Introduction to History). Each course is part of a specific program and is taught by a designated teacher.

1.4 Exams and Scores

1. Two instances of the Exam class are created, representing two types of exams: Midterm and Final.
2. The **prp:examName** property is used to provide the name for each exam.

1.5 Programs

1. Six instances of the Program class are created, representing different academic programs: Computer Science, History, Literature, Chemistry, Psychology, and Physics.
2. The **prp:programName** property is used to provide the name for each program.
3. The **prp:programDuration** property is used to specify the duration of each program, which is set to 4 years in this case.
4. Two instances of the Exam class are created, representing two types of exams: Midterm and Final.

1.6 Scholarships

1. Two instances of the Promotion class are created, representing different scholarships: Academic Excellence Scholarship and Merit-based Scholarship.

2. The **prp:promotionName** property is used to provide the name for each scholarship.
3. The **prp:awardedTo** property is used to specify the students who are awarded each scholarship. The Academic Excellence Scholarship is awarded to the individual Alexander, and the Merit-based Scholarship is awarded to individuals Amelia and Samuel.

2. Onthologies

This ontology file describes an educational domain, including classes for various entities like students, teachers, courses, programs, promotions, and exams. It also defines the relationships between these entities using object properties.

The file starts by defining prefixes for easier reference throughout the ontology:

1. cls: Represents classes in the ontology, such as Person, Teacher, Student, etc.
2. prp: Represents properties in the ontology, such as emailAddress, dateOfBirth, teaches, etc.
3. ind: Represents individuals in the ontology, which are instances of the defined classes, such as specific students or teachers.
4. crs: Represents courses in ontology, which are instances of the Course class.
5. prg: Represents academic programs in the ontology, which are instances of the Program class.
6. sch: Represents scholarships or promotions in the ontology, which are instances of the Promotion class.
7. exm: Represents exams in the ontology, which are instances of the Exam class.

2.1 Classes

Next, it defines the main classes in the ontology:

1. Person: Represents a general person.
2. Teacher: Represents a teacher and is a subclass of Person.
3. Student: Represents a student and is a subclass of Person.
4. EducationalEntity: Represents a general educational entity.
5. Program: Represents an academic program and is a subclass of EducationalEntity.
6. Course: Represents an academic course and is a subclass of EducationalEntity.
7. Promotion: Represents a promotion or scholarship awarded to a student and is a subclass of EducationalEntity.
8. Exam: Represents an exam taken by a student.

2.2 Object Properties

After defining the main classes, the ontology defines object properties to describe the relationships between these entities:

1. emailAddress, dateOfBirth: These properties describe email addresses and dates of birth for Person entities.
2. hasEnrollmentStatus: Describes the enrollment status of a Student in a Course or Program.
3. teaches, taughtBy: These properties describe the relationship between Teacher and Course entities.
4. enrolledIn: Describes the relationship between Student and Course entities.
5. partOfProgram: Describes the relationship between Course and Program entities.
6. awardedTo: Describes the relationship between Promotion and Student entities.
7. courseTitle, courseDescription: These properties describe the title and description of a Course.
8. teacherName, studentName, promotionName: These properties describe the names of Teacher, Student, and Promotion entities.
9. teacherAge, teacherExperience, teacherPopularity, teacherGender, teachingSubject: These properties describe various characteristics of Teacher entities.
10. hasExamScore, score: These properties describe exam scores for Student entities.
11. examName: Describes the name of an Exam entity.

Here is some statistics on the data

Classes:	*9*
Object properties:	*5*
Datatype properties:	*16*
Individuals:	*21*
Nodes:	*25*
Edges:	*26*

3. Queries

For retrieving all the queries first we need to include all these following prefixes.

```
prefix cls: <http://www.universite-cote-dazur.org/classes#>
prefix prp: <http://www.universite-cote-dazur.org/properties#>
prefix ind: <http://www.universite-cote-dazur.org/individuals#>
prefix crs: <http://www.universite-cote-dazur.org/courses#>
prefix prg: <http://www.universite-cote-dazur.org/programs#>
prefix sch: <http://www.universite-cote-dazur.org/scholarships#>
prefix exm: <http://www.universite-cote-dazur.org/exams#>
```

1. Select all the triples:

```
SELECT * WHERE {
  ?x ?p ?y
```

```
}
```

2. Returning only the subclasses of the Person class only:

```
SELECT ?subclass
WHERE {
  ?subclass rdfs:subClassOf cls:Person .
  FILTER ( ?subclass != cls:Person )
}
```

3. Select all the students whose age is bigger than 21:

```
SELECT ?student
WHERE {
  ?student a cls:Student ;
           prp:studentAge ?age .
  FILTER(?age >21)
}
```

4. Construct all the properties of Alexander:

```
CONSTRUCT {
  ind:Alexander ?property ?value
}
WHERE {
  ind:Alexander ?property ?value
}
```

5. All the courses taught by a certain teacher (Isabella in this case):

```
SELECT ?teacher ?course
WHERE {
  ?teacher prp:teacherName "Isabella" .
  ?teacher prp:teaches ?course .
}
```

6. Retrieve the emails:

```
SELECT ?name ?email
WHERE {
  ?teacher rdf:type cls:Teacher ;
           prp:teacherName ?name ;
           prp:emailAddress ?email .
}
```

7. Retrieve the names and dates of birth of all the students who were born after January 1st, 2000:

```
SELECT ?name ?dob
WHERE {
  ?student rdf:type cls:Student ;
           prp:studentName ?name ;
```

```
          prp:dateOfBirth ?dob .
  FILTER (?dob > "2000-01-01"^^xsd:date)
}
```

8.Retrieve the names and popularity of all the teachers who specialize in teaching Mathematics:

```
SELECT ?name ?popularity
WHERE {
  ?teacher rdf:type cls:Teacher ;
           prp:teacherName ?name ;
           prp:teachingSubject "Mathematics" ;
           prp:teacherPopularity ?popularity .
}
```

9. Return the students who are interested in Mathematics (either in Math101 or Math102):

```
SELECT DISTINCT ?student
WHERE {
  ?student rdf:type cls:Student .
  { ?student prp:enrolledIn crs:Math101 } UNION { ?student prp:enrolledIn
crs:Math102 }
}
```

10. Retrieve the names and experiences to know the most experienced teachers:

```
SELECT ?teacher ?teacherName ?experience
WHERE {
  ?teacher rdf:type cls:Teacher ;
           prp:teacherName ?teacherName ;
           prp:teacherExperience ?experience .
}
ORDER BY DESC(?experience)
```

11. Retrieve the courses with most enrolled students in descending order:

```
SELECT ?course (COUNT(?student) AS ?numOfStudents)
WHERE {
  ?student rdf:type cls:Student .
  ?student prp:enrolledIn ?course .
}
GROUP BY ?course
ORDER BY DESC(?numOfStudents)
```

12. Retrieve the names and scholarship types of all the students who received a scholarship:

```
SELECT ?name ?promotion
WHERE {
  ?student rdf:type cls:Student ;
           prp:studentName ?name .
  OPTIONAL {?student prp:receivesPromotion ?promotion}
```

```
}
```

13. Retrieve the names and genders of all the female students:

```
SELECT ?name ?gender
WHERE {
  ?student rdf:type cls:Student ;
           prp:studentName ?name ;
           prp:studentGender ?gender .
  FILTER (?gender = "Female")
}
```

14. Now I want to know the ratio of the male students to female students:

Approach: To calculate the ratio of male students to female students, you can first count the number of male and female students separately, and then divide the number of male students by the number of female students. Here's the query to achieve that:

```
SELECT (COUNT(?maleStudent) AS ?maleCount) (COUNT(?femaleStudent) AS
?femaleCount) ((COUNT(?maleStudent) / COUNT(?femaleStudent)) AS ?ratio)
WHERE {
  {
    SELECT ?maleStudent
    WHERE {
      ?maleStudent rdf:type cls:Student ;
                   prp:studentGender "Male" .
    }
  }
  UNION
  {
    SELECT ?femaleStudent
    WHERE {
      ?femaleStudent rdf:type cls:Student ;
                     prp:studentGender "Female" .
    }
  }
}
```

15. In comparison looking at the ratio of genders between teachers is 0.5 which means there are two times more female teachers than male:

```
SELECT (COUNT(?maleTeacher) AS ?maleCount) (COUNT(?femaleTeacher) AS
?femaleCount) ((COUNT(?maleTeacher) / COUNT(?femaleTeacher)) AS ?ratio)
WHERE {
  {
    SELECT ?maleTeacher
    WHERE {
      ?maleTeacher rdf:type cls:Teacher ;
                   prp:teacherGender "Male" .
```

```
    }
  }
  UNION
  {
    SELECT ?femaleTeacher
    WHERE {
      ?femaleTeacher rdf:type cls:Teacher ;
                     prp:teacherGender "Female" .
    }
  }
}
```

16. Let's look at the percentage of students who are on scholarship:

```
SELECT (COUNT(DISTINCT ?student) AS ?totalStudents)
       (COUNT(DISTINCT ?studentWithScholarship) AS ?studentsWithScholarships)
       ((COUNT(DISTINCT ?studentWithScholarship) / COUNT(DISTINCT ?student)) *
100 AS ?scholarshipPercentage)
WHERE {
  ?student rdf:type cls:Student .

  OPTIONAL {
    ?studentWithScholarship rdf:type cls:Student ;
                            prp:receivesPromotion ?scholarshipType .
  }
}
```

17. Lets see all the averages of the score for all the students:

```
SELECT ?studentName (AVG(?grade) AS ?averageGrade)
WHERE {
  ?student rdf:type cls:Student .
  ?student prp:studentName ?studentName .
  ?student prp:hasExamScore ?examScore .
  ?examScore prp:score ?grade .
}
GROUP BY ?studentName
ORDER BY DESC (?averageGrade)
```

18. Find all students with an average exam score of 90 or higher, along with their names, email addresses, and the course they are enrolled in:

```
SELECT ?student ?studentName ?emailAddress ?course WHERE {
    ?student rdf:type cls:Student ;
             prp:studentName ?studentName ;
             prp:emailAddress ?emailAddress ;
             prp:enrolledIn ?course ;
             prp:hasExamScore [ rdf:type exm:Midterm ; prp:score ?midtermScore
```

```
] ;
            prp:hasExamScore [ rdf:type exm:Final ; prp:score ?finalScore ] .
    BIND((?midtermScore + ?finalScore) / 2 AS ?averageScore)
    FILTER(?averageScore >= 90)
}
```

19. Find all students with an average exam score of 40 or higher, along with their names, email addresses, and the course they are enrolled in:

```
SELECT ?student ?studentName ?emailAddress ?course WHERE {
    ?student rdf:type cls:Student ;
             prp:studentName ?studentName ;
             prp:emailAddress ?emailAddress ;
             prp:enrolledIn ?course ;
             prp:hasExamScore [ rdf:type exm:Midterm ; prp:score ?midtermScore
] ;
             prp:hasExamScore [ rdf:type exm:Final ; prp:score ?finalScore ] .
    BIND((?midtermScore + ?finalScore) / 2 AS ?averageScore)
    FILTER(?averageScore >= 40)
}
```

We can see that all the 10 students passed as the passing point was 40.

4. Conclusion

To sum up, in this report I explored the creation and deployment of a semantic web application for a higher education institution, specifically Université Côte d'Azur. Using RDF graphs and ontologies, the application represents the university's organizational structure, encompassing elements like students, teachers, courses, programs, scholarships, and exams. An extensive range of queries has been devised to retrieve data from the dataset, including aspects like student demographics, faculty expertise, course demand, and student achievement.

By employing semantic web technologies, the higher education institution can effectively handle, investigate, and assess its data. This approach can help decision-making processes and provide insights to enrich the educational experience for both students and staff. The semantic web application can be further developed to include supplementary data sources, such as scholarly publications, alumni connections, or administrative records, thereby augmenting its significance and practicality within the university environment.