MSc. Data Science & Artificial Intelligence

Web of Data

Dr. Catherine FARON

Final project

Author: Joris LIMONIER

joris.limonier@gmail.com

Due: April 3, 2022

Contents

1 Model presentation

The goal of this project is to model an Electronic Medical Records (EMRs) using the XML syntax. We use a mix of foaf, schema and custom namespaces. The structure revolves around an EMR object, which contains informations including:

- $ns : belongsTo$
- $foaf : name$
- $foaf : age$
- $schema : weight$
- $schema : height$
- $ns : hasAllergy$
- $ns : reimbursement$
- $ns : surgery$
- $ns : consultation$

where the $ns : belongsTo$ predicate indicates the owner of the EMR. The other predicates give medical or personal information and should be self-explanatory.
The $ns : consultation$ predicate has range $ns : Consultation$, which is the generic class for consultation instances. Consultations then hold pieces of information about a consultation. That includes:

- $ns : prescription$
- $ns : hasPhysician$
- $ns : diagnosis$
- $ns : price$
- $ns : date$

2 Queries

In this section, we present a number of queries to study our RDF and its schema All queries in this section should be performed with the following prefices defined:

```
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix schema: <https://schema.org/>
prefix ns: <http://www.erm.fr/2022/01/01/ns.rdfs#>
prefix inst: <http://www.erm.fr/2022/01/01/inst.rdfs#>
```

2.1 Show all triples in the namespace *ns*

We know that a triple is composed of Subject, Predicate and Object. This query allows to see all defined triples, which have an object in the ns namespace.

```
select * where { ?subject a ?object . filter(strstarts(?object, ns:)) }
```

2.2 Find people whose age is even

Let $\lfloor x \rceil$ be defined as the rounding of x to the nearest integer. If x is even, then we get:

$$\frac{\lfloor x \rceil}{2} = \left\lfloor \frac{x}{2} \right\rceil \tag{1}$$

we check equation (1) with the following SPARQL query:

```
select ?person ?ageEven
    where{
    ?emr ns:belongsTo ?person
    ?emr foaf:age ?age .
    bind (xsd:integer(?age/2) = xsd:integer(?age)/2 as ?ageEven)
}
```

2.3 Find people with age bigger than 24 or age less than 15

```
select * where {
    ?emr a ns:EMR
{ ?emr foaf:age ?age .
filter (?age > 24) }
union
{ ?emr foaf:age ?age .
filter (?age < 15) }
}
```

2.4 Construct graph with youngerThan relationship

Construct a graph of People, with relationships *youngerThan* if their age is less than the person they are being compared to.

```
construct {?person1 h:youngerThan ?person2}
where {
    ?emr1 ns:belongsTo ?person1 .
    ?emr2 ns:belongsTo ?person2 .
    ?emr1 foaf:age ?age1
    ?emr2 foaf:age ?age2
    filter (?age1 < ?age2)
}
```

2.5 Use OWL to infer ***Person***'s

Get all people in the data. Let "⊆" denotes "is a subclass of". The OWL syntax allows us to deduce the following:

$$ns : Infectiologist \subseteq ns : Physician \subseteq foaf : Person \tag{2}$$

Thus the following query also returns $ns : Raoult$, which is a $ns : Infectiologist$.

```
select * where {
    ?person a foaf:Person
}
```

2.6 Link people to their physicians

Get a list of pairs with people and physicians they had at least one consultation with.

```
select ?person ?physician where {
    ?emr ns:belongsTo ?person .
    ?emr a ns:EMR .
    ?emr ns:consultation ?consultation .
    ?consultation ns:hasPhysician ?physician
}
```

2.7 Find minors who took aspirin

Let's say some medication (*e.g.* aspirin) is found to be dangerous for minors (people under the age of 18). In this case, we would be interested in getting a list of people who took aspirin, then filter only those who are less than 18 years old.

```
select * where {
    ?emr foaf:age ?age
    ?emr ns:consultation ?consultation .
    ?consultation ns:prescription ?prescription .
    ?prescription ns:medication inst:Aspirin .
    filter(?age <= 18)
}
```

2.8 Compute BMI

The Body Mass Index (BMI) is given by:

$$BMI = \frac{weight\ (kg^2)}{height^2\ (m^2)} \tag{3}$$

but since our height is given in centimeters, equation (3) becomes:

$$BMI = 10^4 \times \frac{weight\ (kg^2)}{height^2\ (cm^2)} \tag{4}$$

which we compute using the following SPARQL query:

```
select ?person ?bmi where {
    ?emr ns:belongsTo ?person
    ?emr schema:height ?height
    ?emr schema:weight ?weight
    bind (10000*?weight/(?height*?height) as ?bmi)
}
```

2.9 Compute *isObese*

Starting from the previous example (BMI), determine whether someone is obese. Note that we computed the condition for obesity based on the $bmi > 22$, but this is only for demonstration purposes, since the actual criterion for obesity is $bmi > 25$.

```
insert { ?person ns:isObese ?obese } where {
    ?emr ns:belongsTo ?person
    ?emr schema:height ?height
    ?emr schema:weight ?weight
    bind (10000*?weight/(?height*?height) as ?bmi)
    bind (?bmi > 22 as ?obese)
}
```