IIC3432 — Advanced topics in databases

SPARQL

1. A quick intro to SPARQL

The objective of this class is to familiarize ourselves with the SPARQL[1] query language and how to use SPARQL to query RDF databases served through endpoints.
If you do not know how to write a basic SPARQL query here is a very brief tutorial that will get you up to speed:

http://www.w3.org/TR/sparql11-query/#WritingSimpleQueries

If you have any doubts what a SPARQL query should do, the language standard is at:

http://www.w3.org/TR/sparql11-query/

2. The DBpedia endpoint

The first endpoint we will explore is the SPARQL endpoint of DBpedia. The DBpedia database contains the information extracted from Wikipedia infoboxes (the short summary of Wikipedia you get on every page of some importance) stored in RDF. The access point for DBpedia is its SPARQL endpoint available at:

http://dbpedia.org/sparql

2.1. Complex graph patterns in SPARQL

The query we want to answer here is "who is the oldest Chilean woman in Wikipedia". How would we look for this information manually? (No, it's not on Google!) We'll see how to answer this query very quickly using the DBpedia endpoint.

1. The first thing we need is the IRI corresponding to Chile in DBpedia. One way to get this is using the following query.

   ```
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT DISTINCT ?s WHERE {?s rdfs:label "Chile"@en }
   ```

   What does each thing in this query do? You are looking for all the triples of the form `(?s, rdfs:label, "Chile")`; in the query `@en` is just a convention, the above tutorial has more info on this.

[1] You can search the internet to find out what the acronym SPARQL stands for. It's quite nerdy.

2. Now that you have the IRI corresponding to Chile, you should find the *property* that corresponds to the "place of birth" in DBpedia (hint: in the query above you should replace one element with the IRI of Chile, and another one with a variable). Note that DBpedia has two properties corresponding to "birth place". Find both.
3. With the birth place IRIs you can now find all the resources/IRIs that were born in Chile.
4. Finally, order these RDF objects by their birth date. Who is the oldest? Is it all good? Are there any errors?

2.2. Path queries in SPARQL

Here we will play with path queries in SPARQL. These are known as property paths and are equivalent to the RPQs we introduced in the first lecture. They also use the same semantics as RPQs. Do you see how this could potentially lead to some small issues with respect to the semantics. Which semantics do BGPs in SPARQL use? Do they count repetitions?
The objective here is to find people with a finite Bacon number.

1. First, locate the IRI of Kevin Bacon in DBpedia. Use the same trick as with Chile above.
2. Next, find a predicate that tells you that Kevin Bacon acts in some movie.
3. With the two things above write a simple property path returning people with Bacon number 1.
4. Using Kleene star find people with any Bacon number. What happens? Did you manage to get any results?
5. To remedy the above situation, the SPARQL implemntation behind DBpedia (Virtuoso engine) allows you to repeat a pattern a fixed number of times using the `{,n}` operator in place of the Kleene star. Replace the Kleene star in your query with this operator for different values of n. What happens when n is large?
6. For $n = 4$ try writing the property path manually. What happens now? What does this tell you about SPARQL and the way it processes property paths.

Can you come up with any other "natural" property path query using Kleene star?

2.3. Data modeling exercise

You have now played with two different graph database models: property graphs and RDF/edge-labeled graphs. While RDF seems conceptually simple, think a bit about RDF data in DBpedia and what it actually represents. The best way to do this is to look at DBpedia page of some entity; e.g. `http://dbpedia.org/page/Berlin`. How can you view this RDF data as a property graph? What are the literal values here and their associated properties. What are the actual edges?

3. The Wikidata endpoint

Stolen from wikidata.org: "Wikidata is a free and open knowledge base that can be read and edited by both humans and machines. Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wikisource, and others."
Technically Wikidata is not a native RDF database, but since RDF is so awesome, Wikidata is also served as RDF. The easiest way to work with Wikidata is to use its SPARQL endpoint:

https://query.wikidata.org/

3.1. Wikidata and RDF

As stated above, the data in Wikidata is a bit more general than RDF. Consider for example the entity Q298 (Chile) in the Wikidata display page:

https://www.wikidata.org/wiki/Q298

Look at the property population (P1082).

1. How would you represent the fact that Chile had different population at different times using property graphs?
2. Explain why this can not be done *directly* using RDF.
3. You can however code any sort of a relation in RDF. Using the SPARQL endpoint for Wikidata explore how these sort of relations are represented in RDF in the case of Wikidata.

3.2. Pattern queries in Wikidata

Do the following queries in the Wikidata's SPARQL endpoint:

1. All the countries in Wikidata, together with their label in Spanish (when available).
2. The country that has most neighbors (the countries it has a border with).
3. All the countries whose president or head of state is a woman.
4. The 20 highest mountain peaks.
5. All Chilean writers that are currently alive.
6. All the female Chilean writers, together with the prizes they won (in case they won any prizes).
7. All the Chileans working in Computer Science.

3.3. Path queries in Wikidata

Do the Kevin Bacon query in Wikidata. For this locate Kevin Bacon, and locate some movie he starred in using the Wikidata search option. How does this endpoint respond to the Kleene star query? What about a fixed number of iterations? Do them manually as well.