3. Unit: Java/Jena

This exercise sheet focuses on plain RDF data (no reasoning) handling with Java/Jena.

Exercise 3.1 (Language Tags) A basic exercise: read Mondial into a model and modify the RDF graph according to the results of “simple” SPARQL queries against Wikidata:

Reuse the results of Exercise 2.5 to add the language tags to the local- and other-names: Remove the old non-tagged values and replace them with the new ones. Write the new Mondial version in an n3/turtle file.

Exercise 3.2 (Uninterrupted line-up of heads of state) This exercise exploits procedural programming in Java/Jena do do data-driven exploration of Wikidata and collect results in an RDF graph. It extends Exercise 2.2.

- Collect information about *previous* heads of state and heads of government of states.
- Which country has the longest uninterrupted line-up of heads of state or head of government?
  - It is considered uninterrupted if a head of state/government begins her/his term of office within a year (365 days) after her/his predecessors term ends.
  - Try to include predecessor states of the current states to extend the line of heads of state.

Exercise 3.3 (City-sameAs between Mondial and Wikidata) Use the results from Exercise 2.4 to generate the owl:sameAs triples for provinces and for cities between Mondial and Wikidata.

This process cannot be done using a single query against Wikidata: it would run into Wikidata’s timeout, and refined queries for provinces/cities not found by general queries might be necessary.

So, procedural programming can be used for *partitioning* the problem by solving subsets (e.g. cities in some country) separately. Furthermore, queries that check for instances not yet found can be used to inspect the model.

Exercise 3.4 (Catalonia becomes independent) Consider the case that Catalonia leaves Spain Then, some updates have to be executed on Mondial:

- plan first your strategy, before starting to program,
- consider especially, what kinds of data items must be changed, and how,
- write the programs as generic as possible (such that they can be applied also whenever some other province turns into a new, independent country).
- load Mondial as Default Graph into the model.
- load catdata.rdf (see below) as named graph into the model
- update the default graph accordingly. Compute new values and update relationships where appropriate (e.g., area and population of Spain etc.).
- Catalonia becomes a member of all organizations where Spainis a member.
- All necessary new facts about Catalonia can be found in the [catdata.rdf] file.

Exercise 3.5 (Partitioning Mondial by Continents)

- The goal is to separate the complete mondial.n3 into seven NAMED GRAPHs: one for each continent, the sixth for all seas, and the seventh for all organizations.

  - Do not load the complete mondial.n3 into the system, but extract information on-demand from mondial.n3 (or from querying the LOD service).
  - Use "encompassed" and and "located" to determine for countries and geographical features to which graph they belong. Use Node.getAllStatements (?) and a suitable recursion for handling "subtrees" (like provinces and cities).
  - countries located on two or more continent should be stored for each of them. In each graph, only intra-continent country borders should be stored.
  - Membership information should be stored in the organizations' graph.
- output each graph as a .ttl file.
- keep each of them as a named graph in the data model.
- use this for computing the highest mountain on each of the continents (ignore mountains on islands).
  (Use the GROUP BY ?GRAPH feature, and not again "encompassed")
- Compute the lowest one of these mountains.
- Use the "union of all graphs as Default graph" feature to state the query from Exercise 1.4 unchanged.