Practical session : Triple Stores

Objective

The objective of this practical session is to:

- use the Jena API to create and access RDF data using a Java program
- use Jena TDB as a triple store store that is needed when the datasets are very big and cannot fit in memory
- use GraphDB to create and query RDF data and an OWL ontology
- use OntoRefine to transform tabular data into RDF and query them using SPARQL.

Some needed materials are available at e-campus (course 5/Lab-5):

Link 1 : https://ecampus.paris-saclay.fr/course/view.php?id=32757&section=6#section-6

Part 1 : Jena API and Jena TDB triple store

For the ones that do not know any thing about Java and prefer use Python, please do the same exercises using RDFLib (https://github.com/RDFLib)

From https://jena.apache.org/tutorials/rdf_api.html, download the suitable file apache-jena-3.16.zip and unzip it in your java workspace directory.

In the sub-directory ./lib you will find all the jar files that can add to your java application that uses Jena and TDB.

In the sub-directory ./src-examples you will find a series of examples in Java for the Jena API including the use of TDB.

Open your Java IDE (Eclipse, Netbeans, …) and add the .jar to the class-path

The Java imports that you may need are:

```
import java.io.InputStream;
import java.util.*;
import org.apache.jena.graph.Graph;
import org.apache.jena.graph.GraphListener;
import org.apache.jena.graph.Node;
import org.apache.jena.graph.NodeFactory;
import org.apache.jena.graph.Triple;
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.ResourceFactory;
import org.apache.jena.sparql.core.Quad;
import org.apache.jena.sparql.util.graph.GraphListenerBase;
import org.apache.jena.tdb.TDBFactory;
import org.apache.jena.tdb.TDBLoader;
import org.apache.jena.tdb.store.DatasetGraphTDB;
import org.apache.jena.tdb.sys.TDBInternal;
import org.apache.jena.util.FileManager;
import org.apache.jena.vocabulary.RDFS;
import org.apache.jena.rdf.model.*;
import org.apache.jena.query.Dataset;
import org.apache.jena.query.ReadWrite;
import org.apache.jena.tdb.base.file.Location;
```

For querying part :

```
import org.apache.jena.query.Dataset;
import org.apache.jena.query.ReadWrite;
import org.apache.jena.query.Query ;
import org.apache.jena.query.QueryExecution ;
import org.apache.jena.query.QueryExecutionFactory ;
import org.apache.jena.query.QueryFactory ;
import org.apache.jena.query.QuerySolution ;
import org.apache.jena.query.ResultSet ;
```

Exercice 1: Jena API

Consider the java examples of given in course 5 (slides 25-29) and create and read the parent-children RDF data graph. You can find parts of the code at e-campus Link 1.

Exercice 2: Jena TDB for RDF

TDB store can be used through Jena API in command line or in a Java application. Here, we will use TDB via a Java application and Eclipse.

1) This function allows you create an instance of the dataset given in `file` and put it in the `directory`. This function is executed only one time.

Consider the RDF file "restaurant1.rdf"

```
public Dataset createDataset(String file, String directory)
{
	try{
	Dataset dataset = TDBFactory.createDataset(directory);
	dataset.begin(ReadWrite.WRITE) ;
	Model model = dataset.getDefaultModel();
	TDBLoader.loadModel(model, file);
	dataset.commit();
	dataset.end();
	return dataset;
	}catch(Exception ex)
	{
		System.out.println("##### Error Fonction: createDataset #####");
		System.out.println(ex.getMessage());
		return null;
	}

}
```

---

2) One can use the two following instructions to create a Jena model from the dataset created using TDB and stored in `directory`.

```
Dataset d = TDBFactory.createDataset(directory);

Model model = d.getDefaultModel();
```

3) Then, to read the content of the model, you can for example display all the named graphs of the dataset as follows:

```
d.begin(ReadWrite.READ);
try {
	Iterator<Quad> iter = d.asDatasetGraph().find();
	int i=0;
```

```
System.out.println("begin ");
while (iter.hasNext() && i < 20) {
      Quad quad = iter.next();
      System.out.println("iteration "+i);
      System.out.println(quad);
      i++;
  }
} finally { d.end(); }
d.close();
System.out.println("finish ...");
```

---

4) Query the dataset using SPARQL queries:

a- a simple counting query

```
String sparqlQueryString = "SELECT (count(*) AS ?count) { ?s ?p ?o }" ;
// See http://incubator.apache.org/jena/documentation/query/app_api.html
Query query = QueryFactory.create(sparqlQueryString) ;
QueryExecution qexec = QueryExecutionFactory.create(query, d) ;
try {
      ResultSet results = qexec.execSelect() ;
      for ( ; results.hasNext() ; )
      {
            QuerySolution soln = results.nextSolution() ;
            int count = soln.getLiteral("count").getInt() ;
            System.out.println("count = "+count) ;
      }
      } finally { qexec.close() ; }
      // Close the dataset.
      d.close();
```

---

b - a more complex query with a set of results

```
String sparqlQueryString = "… "

// See http://incubator.apache.org/jena/documentation/query/app_api.html
```

```
Query query = QueryFactory.create(sparqlQueryString) ;
QueryExecution qexec = QueryExecutionFactory.create(query, d) ;
try {
      ResultSet results = qexec.execSelect() ;
      while (results.hasNext()) {
      QuerySolution sol = results.next();
      System.out.println("Solution := "+sol);
      for (Iterator<String> names = sol.varNames(); names.hasNext(); ) {
            final String name = names.next();
            System.out.println("\t"+name+" := "+sol.get(name));
         }
       }
      } finally { qexec.close() ; }

      // Close the dataset.
      d.close();
```

Part 2 : GraphDB and OntoRefine

- Download installation file and run it (https://graphdb.ontotext.com/)
- Access workbench via http://localhost:7200/
- More information: https://graphdb.ontotext.com/documentation/free/

1- In the work bench create a dataset from an RDF file (take `human_2007_09_11.rdf`).

Try the offered functionalities of GraphDB:
- *Explore*, with different modes
- *SPARQL* querying, you can use the queries given in file human-sparql-queries.zip

2- You can also use the *ontoRefine* functionality for importing tabular data (take movies.xls file), transforming it to RDF and the querying it using SPARQL