LAB Assignment

This is a hands-on lab on knowledge graphs. We will use the GraphDB triplestore for working with a knowledge graph. To prepare for the session, you are recommended to get familiar with the RDF Schema vocabulary specification, SPARQL specification and GraphDB manual.
In the sequel, we provide the environment setup and then exercises to be solved. Each group must upload the solution of the exercises to the Learn-SQL platform (look for the corresponding assignment event). One group member must submit the solutions (check below for a precise enumeration of what you need to submit) and list all group members in such document. Check the assignment deadline and be sure to meet it. It is a strict deadline!

Setup Instructions

For section A

For section A, you need to load the DBpedia ontology TBOX to GraphDB. Follow these steps:

- Download the Graph DB installer: `https://www.ontotext.com/products/graphdb/graphdb-free/`
  Fill in the form that appears on the right. After submitting the form, you will receive an email at the address you provided with various download links depending on your OS (Windows, MacOS or Linux). Please also check your junk / spam folder in case the email went there.

- Install GraphDB: After downloading the installer, you should run normally and you do not require an administrator account to run it. Follow the steps that appear in the executor without changing any of the configurations and wait for the installation process to finish.

- Launch GraphDB: Once the installation is complete, you should start the GraphDB server by launching the `'GraphDB Free'` application from your home / start menu. Once the server is started, a new window in your browser should open auto- matically with the URL `http://localhost:7200/`.

- Configure settings: Go to the GraphDB application and click on the button `'Settings...'` at the top. A new pop-up window should open, and you should introduce the following lines in the textbox called `'JVM settings'`:

  ```
  graphdb.workbench.maxUploadSize=40000000000
  ```

  ```
  Xmx=2000m
  ```

  Afterwards, click on the `'Save and restart'` button at the bottom right. The GraphDB server will restart and will open a new browser window again with the same URL as above.

LAB Assignment

- Load a dump from DBpedia: It includes both the ontology T-Box and some A-Box instances. Download the two files from the following URLs (note that the second file is large and might take few minutes to download):

  `http://downloads.dbpedia.org/2014/dbpedia_2014.owl.bz2`

  `http://downloads.dbpedia.org/2014/en/instance_types_en.nt.bz2`

  Once the download is complete, unzip both downloaded files. This should generate two files, one with the extension `.owl` and another with the extension `.nt`.

- Create a repository: Before importing any data, we need to first create a repository. Go the GraphDB interface and select `'Settings'` and then `'Repositories'` from the left-hand menu. Afterwards, click on the `'Create new repository'` and apply the following:

  Insert a `'Repository ID'` and `'Repository title'` of your own.

  Turn inference off: Under `'Ruleset'` select `'No inference'`.

  Note: GraphDB supports inference out of the box. Inference is the derivation of new knowledge from existing knowledge and axioms. It is used for deducing further knowledge based on existing RDF data and a formal set of inference rules. For more details please check:

  `http://graphdb.ontotext.com/documentation/free/devhub/inference.html#inference-in-graphdb`, and

  `http://graphdb.ontotext.com/documentation/standard/reasoning.html`.

- Load data into the repository: Go the GraphDB interface and select `'Import'` and then `'RDF'` from the left-hand menu. Afterwards, click on the `'Upload RDF files'` button and choose the two files that you just extracted. Wait for uploading the files to be finished, it should take around 3 - 4 minutes to upload the larger file. Once both files are uploaded, select them both and click on the `'Import'` button at the top. A new pop-up will appear and you need to fill in the following configuration parameters:

  Base IRI: enter `http://dbpedia.org/resource/`

  Target graphs: choose `'Named graph'` and enter the following in the text box: `http://localhost:7200/dbpedia/`

  After that, click on the `'Import'` button at the bottom right corner of the pop-up and wait for the import operation to finish. Since the files are large, the import will take around 10 minutes to be done.

- Test: Go to `'SPARQL'` on the left-hand menu and copy the following query into the text box that appears:

```
SELECT DISTINCT ?s WHERE { ?s ?p ?o .  } LIMIT 50
```

Click on the `'Run'` button at the bottom right corner of the text box to execute the query. If the query does not return any results, this means that no data was found in the repository and that the import operation failed. If it does return results, the import was successful and you can proceed with the exercises of the session.

- Turn on autocomplete: Click on `'Setup'` and then `'Autocomplete'` from the left-hand menu. In the page that appears, toggle the button at the top under `'Autocomplete index'` to change it from off to on. A new status that says `'building'` should appear to the right of the toggle. This will activate autocomplete when we are executing queries and will allow us to search for properties or classes that are defined in our ontology. You do not need to wait for the building to finish, you can continue with the exercises of this session and building the autocomplete index will carry on in the background.

What to deliver?

Upload ONE document with your solutions. Name the file as '`[Group]-[MemberSurname]+.pdf`'. The file must contain the three sections that you will find below and, within each section, insert your solution. SPARQL statements must compile and be fully correct (both syntactically and semantically). If you make any assumption not explicit in the statement, add a note before your SPARQL statement.

A Exploring DBpedia

DBpedia

DBpedia[1] *is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. This structured information resembles an open knowledge graph which is available for everyone on the Web. A knowledge graph is a special kind of database which stores knowledge in a machine-readable form and provides a means for information to be collected, organised, shared, searched and utilised. Google uses a similar approach to create those knowledge cards during search.*

Tasks

If you successfully performed the previous steps and started GraphDB, you are now able to open a web tab at the port 3200 to access the GraphDB interface. The advantage of using GraphDB is that it offers an advanced visual interface to explore the classes defined in an ontology. To access this feature, click on `'Explore'` and then `'Class hierarchy'`

[1] https://wiki.dbpedia.org

from the left-hand menu and wait a few seconds for the graphic to load. After that, you should get a visual graphic that depicts the different classes defined in the DBpedia ontology and the hierarchy between them. For instance, you can see that the biggest class is called `'Agent'`, which has many sub-classes. When clicking on the bubble representing each class, a pop-up panel with metadata about that class appears to the right. This panel contains for example the name of the class, its description and some of its instances. Spend some minutes exploring the different classes that you have in the graphic and understanding the different hierarchies that they constitute. Once your curiosity is satisfied, go back to the `'SPARQL'` page and execute the following queries that are related to the classes and properties defined by the DBpedia ontology:

1. Get the classes defined in the ontology.
2. Get the datatype properties defined in the ontology.
3. Get the object properties defined in the ontology. What is the difference between datatype and object properties?
4. Get the labels of all the properties (both datatype and object) defined in the ontology.

   As commented earlier, it is important to notice that the semantics of the knowledge graph are determined by different namespaces, which are defined by the "prefix" keyword at the beginning of the query. Thus, it is important to identify and understand the namespaces used in a dataset.
5. Find the class representing an Actor in the dataset (using filters).
6. Find the super class for the class Actor.
7. Find all the actors in the dataset.
8. Get different classes that are defined as range of the properties that have the class Actor defined as their domain.
9. Find the super property of the `goldenRaspberryAward` property.
10. Return all the properties that have the class Actor as either their range or domain.
11. Return all persons that are not actors.
12. Return the path (in properties and classes) between the Actor and Person classes.

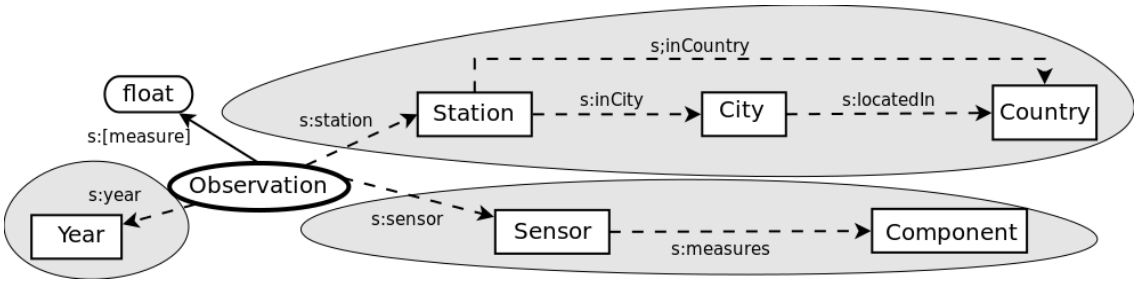

Figure 1: QBOAirbase cube structure

B Analytical queries on top of QBAirbase

European Air Quality Database

The European Air Quality Database[2] *contains data from different countries, integrated into a knowledge graph and thus available as Linked Data (LD). The LD structure of the European Air Quality Database is known as QBOAirbase*[3] *and contains the QB4OLAP vocabulary. QB4OLAP is an extension of QB vocabulary*[4] *to perform Business Intelligence over LD. It enables the user to represent OLAP cubes in RDF and to implement OLAP operators (such as Roll-up, Slice, and Dice) as SPARQL queries directly on the achieved RDF representation. QB4OLAP adds classes and properties to QB that gives datasets the capability of representing dimension levels, level members, rollup relations between levels and level members, and associating aggregate functions to measures.* A visual representation of QBOAirbase's cube structure is given in Figure 1:

Finally a conceptual schema of the QBOAirbase dataset is shown in Figure 2.

Tasks

In this exercise we will query an external integrated dataset of our interest. First, you will have to write some SPARQL queries about specific data aspects of the dataset. Then, once you are familiar with it, we will ask you to explore the data on your own and propose some additional queries.
QBOAirbase can be accessed in 3 different ways:

- Load the dataset into your local GraphDB:

  `'Import'->'RDF'->'Get RDF data from URL'`: `http://130.226.98.179/kashif/qb.zip`

[2] `https://www.eea.europa.eu/data-and-maps`
[3] `http://qweb.cs.aau.dk/qboairbase/`
[4] `https://www.w3.org/TR/vocab-data-cube/`

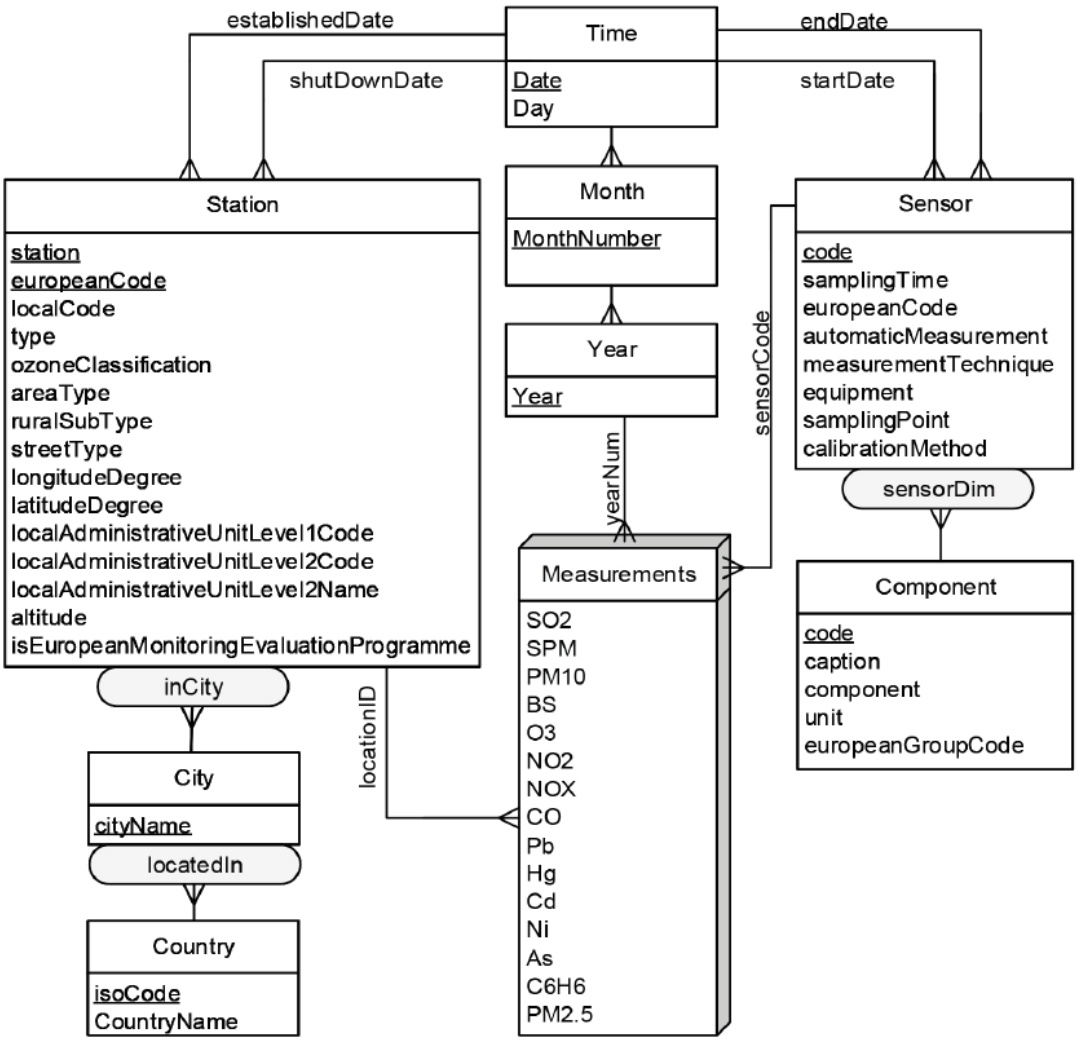

Figure 2: QBOAirbase conceptual schema

- Use the available SPARQL Endpoint: `http://lod.cs.aau.dk:8891/sparql`.
- Use a GraphDB server from the following link: `http://130.226.98.179:7200/`

  username: `user`

  password: `user`

Using any of the methods above to access the database, write the SPARQL queries for the following:

1. List the country, station type, latitude, and longitude details of each station.
   Note: Limit the query to 25 results, and extract only the string values of the required object and not the whole IRIs.
2. List the 10 highest averages of C6H6 emission and the country and the year on which they were recorded.
   Note: A sensor has a property (defined through the prefix: `<http://qweb.cs.aau.dk/airbase/property/>`) `stastisticShortName`, and it can be Mean, Max, etc.
3. For each city and property type, give the yearly average emission for NO2, SO2, PB, and PM10.
4. Define 3 additional SPARQL queries (and their corresponding interpretation) that you think could be interesting for the domain of analyzing air quality/pollution.

C Ontology creation

In order to have a good grasp of the semantic web and the linked data initiative we need to know what ontologies are, how they are used, and how they are created. Note that in the world of knowledge graphs the main point is to reuse existing ontologies. That is, in order to facilitate data crossing and integration, it is worth to link our data with already existing data. In this assignment we will practice how to create your own ontology (i.e., a knowledge graph with a well-defined TBOX and ABOX).

C.1 TBOX definition

Define a TBOX for the research publication domain. Try to define it as complete as possible. The concepts to be used include, but are not limited to: *Author, Reviewer, Paper, Short Paper, Demo Paper, Survey Paper, Full Paper, Conference, Database Conference, Journal, Open Access Journal, Review, etc* along with their properties.

Note: To create the TBOX, you can make use of Protege[5], or VocBench[6], or any other graphical tool. If you are interested in it, you can use OWL instead of RDFS for this exercise.

Tasks:

1. Depending on how you created the TBOX, you need to provide either the SPARQL queries you used for creating the TBOX (in case you used SPARQL), or in case you used another tool, the methodology/method you used and the output generated in a graphical form (the lecturer should not install any additional tool to validate this part).
2. Provide a visual representation of the TBOX (You can use `https://gra.fo/` for instance).

C.2 ABOX definition

In this section we want you to reuse the data you have about the research article domain (from the Assignment of Lab 1) and convert it into an ABOX, that is, define it as an RDF graph.

Note: There are many ways to convert CSV, or relational data, or JSON into RDF. You can use the Jena API[7] and do it programatically or use some existing software, see Any23[8], Open Refine[9] with its RDF extension[10], or any other available tool.

Tasks

1. Explain the method used to define the ABOX.

C.3 Linking ABOX to TBOX

In this section we want to link the ABOX with the TBOX defined. That is, use SPARQL to create new triples that will link the TBOX elements with ABOX elements.

Note: See the `CONSTRUCT` statement in SPARQL.

[5] `https://protege.stanford.edu`
[6] `http://vocbench.uniroma2.it/downloads`
[7] `http://jena.apache.org/index.html`
[8] `https://any23.apache.org`
[9] `http://openrefine.org/`
[10] `https://github.com/stkenny/grefine-rdf-extension/wiki`

Tasks

1. Provide the SPARQL queries required to create the link between the ABOX and TBOX.
2. Provide a summary table with simple statistics about the RDF graph obtained, e.g., the number of classes, the number of properties, the number of instances, etc.

C.4 Queries on top of the Ontology

In this section we want to pose a set of queries on top of our created ontology. We are specifically interested on the benefits of having an explicit TBOX defined. Thus, in this section we want to define queries that exploit the TBOX and queries that ignore the existence of the TBOX. In this case you may want to turn inference on, thus when creating the repository leave the default option for the `'Ruleset'` parameter.

Tasks

Write two versions for each of the following queries (one exploiting the TBOX, and another assuming the TBOX does not exist). Please explicitly state any assumptions you make.

1. Find all the Authors.
2. Find all the properties whose domain is Author.
3. Find all the properties whose domain is either Conference or Journal.
4. Find all the things that Authors have created (either Reviews or Papers).