CentraleSupélec

université PARIS-SACLAY

Geological Knowledge Base Construction

Big Data Research Project

Gayane VARDANYAN
Yuhsuan CHEN

—

prepared at CentraleSupelec

Defended on February 2021

Advisor: Nacéra SEGHOUANI - CentraleSupelec nacera.seghouani@centralesupelec.fr
Advisor: Francesca BUGIOTTI - CentraleSupelec francesca.bugiotti@centralesupelec.fr
Advisor: Sylvain WLODARCZYK - Schlumberger SWlodarczyk@slb.com

CentraleSupélec

BDMA

Contents

Chapter 1

Introduction

This project focuses on the construction of a geological knowledge base that would help internal stakeholders answer questions about stratigraphy in north-sea, serving as a search engine for researching on domain-related questions and obtaining relevant answers in minimum amounts of time.

The main contribution of this work is a ready and functional knowledge base, which we created from scratch- obtaining and processing relevant data, creating the model, populating the latter, parsing typical relevant questions into queries that would run on the knowledge base.

Chapter 1 will focuses on the definition of our problem statement and the main objectives, followed by Chapter 2, where we perform domain-related literature review, aimed at understating the background information for the knowledge base; as well as proceed with modelling the knowledge base. Chapter 3 focuses on the preliminary data gathering and pre-processing, based on the model obtained in the previous chapter. Chapter 4 talks about the tools and decision made regarding the TBox model creation, as well as describes the process of generating the TBox, logically followed by Chapter 5, which contains information about the physical implementation of the TBox as well as its populating with data using Apache Jena. Chapter 6 deals with a specific particularly difficult question that defines the geolocation for a formation. Finally in Chapter 7 we connect the TBox and Abox created, translate queries into SPRQL for testing purposes. Later on in Chapter 8 we implement a model for parsing natural language questions into SPARQL-like queries, that would help answer any questions using the knowledge base.

Further information on each step and the methodologies that we used are presented in this in-detail report, in the corresponding chapters.

Chapter 2

Project Definition

2.1 Introductory Remarks

The aim of this project is the construction of a geological knowledge base capable of answering questions about stratigraphy in north-sea. The knowledge based is aimed at being used by the internal stakeholders as a search engine for researching on domain-related questions and obtaining relevant answers in minimum amounts of time.

The overall objective is to construct a performance-efficient knowledge base that would work closely with a natural language processing engine for parsing natural language questions into database queries and providing answers for those.

Typical domain-related questions that present particular interest would be:

- What is stratigraphy ?
- Where is Ekofisk Formation ?
- What are the wells crossing Ekofisk Formation ?
- What is the group of the Ekofisk Formation ?
- What is the lithology of Ekofisk ?
- Describe at Best Ekofisk Formation
- What is the top of the Ekofisk Formation for the well 1/3-1 ?
- What are the members of Ekofisk ?
- What is the period and age of Ekofisk ?

The overall project implementation process could be deconstructed into of a number of modules, each being an integrated part of the final result. Namely, the main steps to be implemented are:

- Research into the domain with the purpose of understanding the specifies and suggesting appropriate solutions.
- Research into related works of knowledge-base creation, to compare different alternatives and choose the best fits for the architecture of the product. In particular we will:
  - Compare and contrast different NLP tools, such as BERT or NLTK to choose the best one in the context of the project.

  - Compare several triple-store creation tools, such as GraphDB, Apache Jena, with a number of property graph based knowledge base creation tools like Neo4J, poolparty.biz, etc, to decide on the most appropriate ones to be exploited in the frameworks of the project.

- Data gathering process, particularly through scraping the websites containing answers of the main domain-related questions.
- The creation of the knowledge base itself.
- Utilizing the knowledge base for answering domain-related questions about the stratigraphy in north-sea.
- Parsing natural language questions into database-readable queries to interact with the knowledge base created.
- Overall evaluation of the performance of the created systems.

While the project entails a large amount of work we decided to spread it out evenly to fit in the limited time-frame of 13 weeks, using 2 week-long sprints, each focusing on a larger chunk of work dealing with on specific aspect of the project. At the end of every sprint a meeting is intended with the coordinators of the project for reporting the results as well as discussing any challenges and advancements possible. At the same time day-to-day communication between the pair members ensures discussing intermediate tasks and efficiently dealing with any major or minor complications.

In the next section of the report we present the first and preliminary step results. In particular we researched a number of domain-related websites guided by the list of questions mentioned above and found the answers for each. We indicate below the researched questions alongside the answers as well as the respective sources where-from the answer can be obtained.

Chapter 3

Modelling the Knowledge Base

In this chapter we describe the process of logical modelling of the knowledge base. First we focus on the literature review that was performed with the purpose of gaining preliminary knowledge regarding the domain area. Then we introduce the reasoning behind the model, which we intend to create in the further chapters, specifically illustrating the main entities and the relationships between those entities, as well as providing examples of typical instances.

3.1 Domain-related Literature Research

The primary step is to collect text from multiple websites and rephrase them into a paragraph to answer the domain-related questions we listed before. By doing this, we allow getting a more in-depth understanding of stratigraphy and Ekofisk formation.

1. What is stratigraphy ?
   Stratigraphy is a geology study involved the study of the rock layer(strata). It includes three main sub fields, lithostratigraphy, biostratigraphy and chronostratigraphy. To better understand the topic, in lithostratigraphy, it studies the wells log, and physic characteristic of the rocks, including texture, mineral content and color. For biostratigraphy and chronostratigraphy, they studies fossils to determine the absolute or relative age of the formation or other types of rocks.

2. Where is Ekofisk Formation ?
   There is no documentation clearly state where is the Ekofisk formation. However, we can know from the previous studies that it locate in the North Sea continues throughout the basinal areas of the Central Graben(Central Trough), Outer Moray Firth and South Viking Graben and the Southern North Sea.
   Even though no actual distribution of Ekofisk formation, we can answer the exact location of the Ekofisk oil field that it is in block 2/4 of the Norwegian sector of the North Sea approximately 320 km southwest of Stavanger. The Greater Ekofisk Area contains the following oil fields: Cod, Ekofisk, West Ekofisk, Tor, Albuskjell, Eldfisk, Edda and Embla.

3. What are the wells crossing Ekofisk Formation

   - Norwegian well 1/3-1 from 3354 m to 3258 m, coordinates N 56°51’21.00", E 02°51’05.00". No cores.
   - UK well 22/1-2A from 2982.5 m to 2935 m, coordinates N 57°56’12.20", E 01°02’55.80". No cores.

- Norwegian well 2/5-1 from 3132 m to 3041 m, coordinates N 56°38'19.95", E 03°21'07.94". Cored through the upper 78 m.

4. What is the group of the Ekofisk Formation ?
   In most resources, they stated that Ekofisk formation belongs to the Chalk group. However, in the fact sheet of Norwegian Petroleum Directorate, it says that it belongs to Shetland Group as it contains the former Chalk Group.

5. What is the lithology of Ekofisk ?
   The Ekofisk formation comprises white, chalky limestones containing white and grey crystalline and bedded dark layers and thin grey to green calcareous. There are also brownish-grey cherts can be seen rarely to richly throughout the formation.

6. Describe at Best Ekofisk Formation
   Ekofisk formation is named after the Ekofisk Oil Field. The formation is widespread in the southern and central North Sea. Its thickness is up to 140 m and mainly form by white and chalky limestones, but some glauconite can also be seen in the base.

7. What is the top of the Ekofisk Formation for the well 1/3-1 ?
   According to Norwegian Petroleum Directorate, the depth is 3258.

8. What are the members of Ekofisk ?
   The larger part of the lower member (lower part) of Ekofisk formation consists of the informal Ekofisk reworked zone with mainly reworked Maastrichtian chalks (Tor Formation) deposited as various mass flows and peridotite-facies chalks. The lowest part is called tight zone, which consists of a low porosity to tight zone with higher terrigenous clay content.
   The upper member is mainly homogenous chalks with low clay content, reworked Danian chalks and minor turbidites. A lower tight to low porosity zone (Tommeliten tight zone) is present in parts of the Central Graben.

9. What is the period and age of Ekofisk ?
   Danian, the Danian is the oldest age of the Paleocene period. The Danian age started from the Cretaceous–Paleogene extinction event 66 Ma. to 61.6 Ma, being followed by the Selandian age.

In this stage, we visited several websites from dictionary terms to the fact sheet of Norway, the United Kingdom and Netherlands. It was not easy to summarize them for human because most of them are not daily using words, and we need to understand them first. However, the situation becomes more tricky for the computer. Our next challenge would be to implement the natural language processing techniques to enable the program to find the pattern and extract the essential insight from them.

3.2 Knowledge Base Modelling

After exploring the questions mentioned in the previous chapter and coming up with the answers for those using all the sources, the next step was structurally designing the knowledge base that would be able to provide the answers for all those questions.

The modelling process was performed bearing in mind triple-store structure.We decided on using graphs as a model for representing the triple-store-based knowledge base, given that graphs best represent entities with various relationships with each other. The nodes of our graph represent the entities, which the directed edges stand for the relationships between two entities.

First the main concepts were identified, and then the relationships between those concepts were named accordingly to portray the real life relationships in the best way possible. A few instances per entity were also identified for a clearer picture of what the final knowledge base would look like. To easier differentiate between the two- the entities are visualized in blue, while the instances are in green.

In figure 2.1 a snippet from the knowledge base covering the theoretical questions regarding stratigraphy is presented.

- We decided on generalizing `study_field` and definition as entities and taking stratigraphy, biostratigraphy and lithostratigraphy as instances of `study_field` for a number of reasons. Firstly this format allows for easy modifications of information regarding each instance.
- At the same time the structure above makes the model generally more flexible, as it can get expanded over other study areas at any point in time, if the necessity arises.
- Between different study-fields we have defined 2 relationships- two areas can be related, or one can be the sub-field of the other. For each `study_field` we also have a triple as follows: a `study_field` –> `is_defined` –> as a definition.

The figure 2.2 further expands the structure of the knowledge base. The main reasoning is as following:

- The concepts formation, group, lithology, well are used as central ones, representing the central geographical terms around which the overall knowledge base is designed.
- At the same time we decided to make geolocation as a separate entity referencing the class of numbers, for longitude and latitude, since a question in the primary list of our guiding questions is about locating the given well.

The relationships between the entities above correspond to the following domain-based relationships:

- A formation is located in a geolocation
- A formation is a part of a group
- A formation has an age (that being a numerical value)
- A formation was formed during a period
- A formation is characterized by a lithology
- Members belong to formations
- Wells cross formations

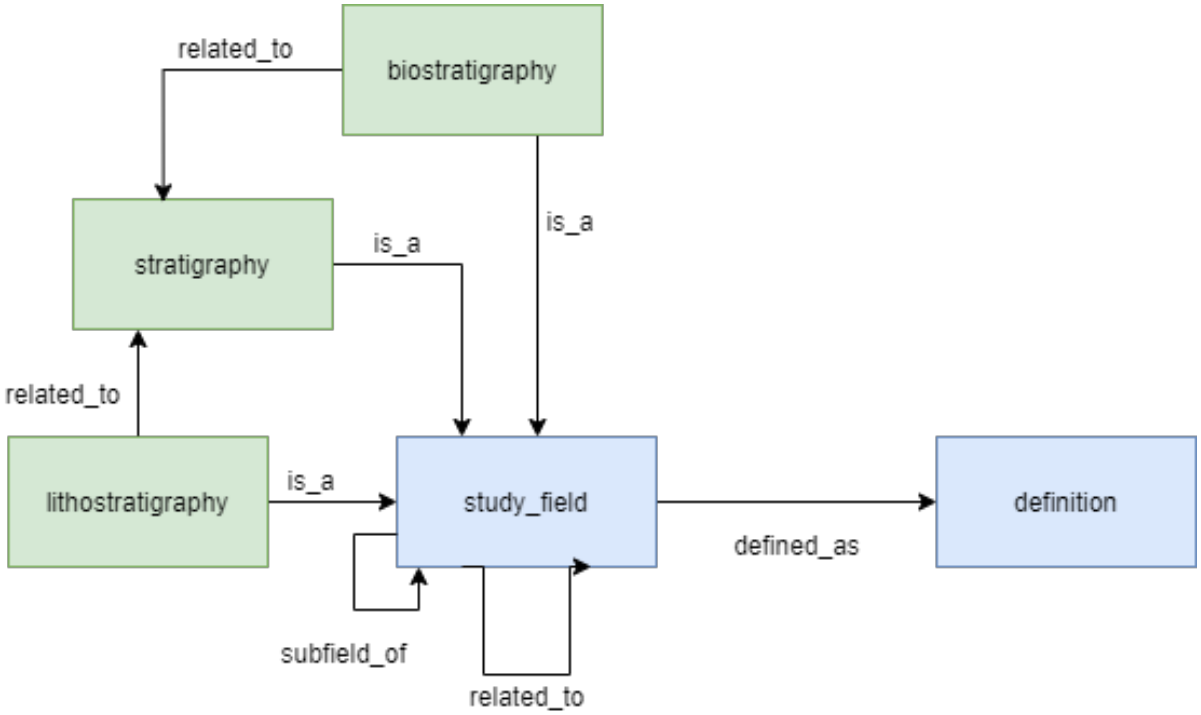

Figure 3.1: Knowledge base for questions about stratigraphy

- Wells have top and bottom depths (numerical values)

The structure again, apart from showcasing the given concepts, allows for later expansion with not only new instances but concepts.

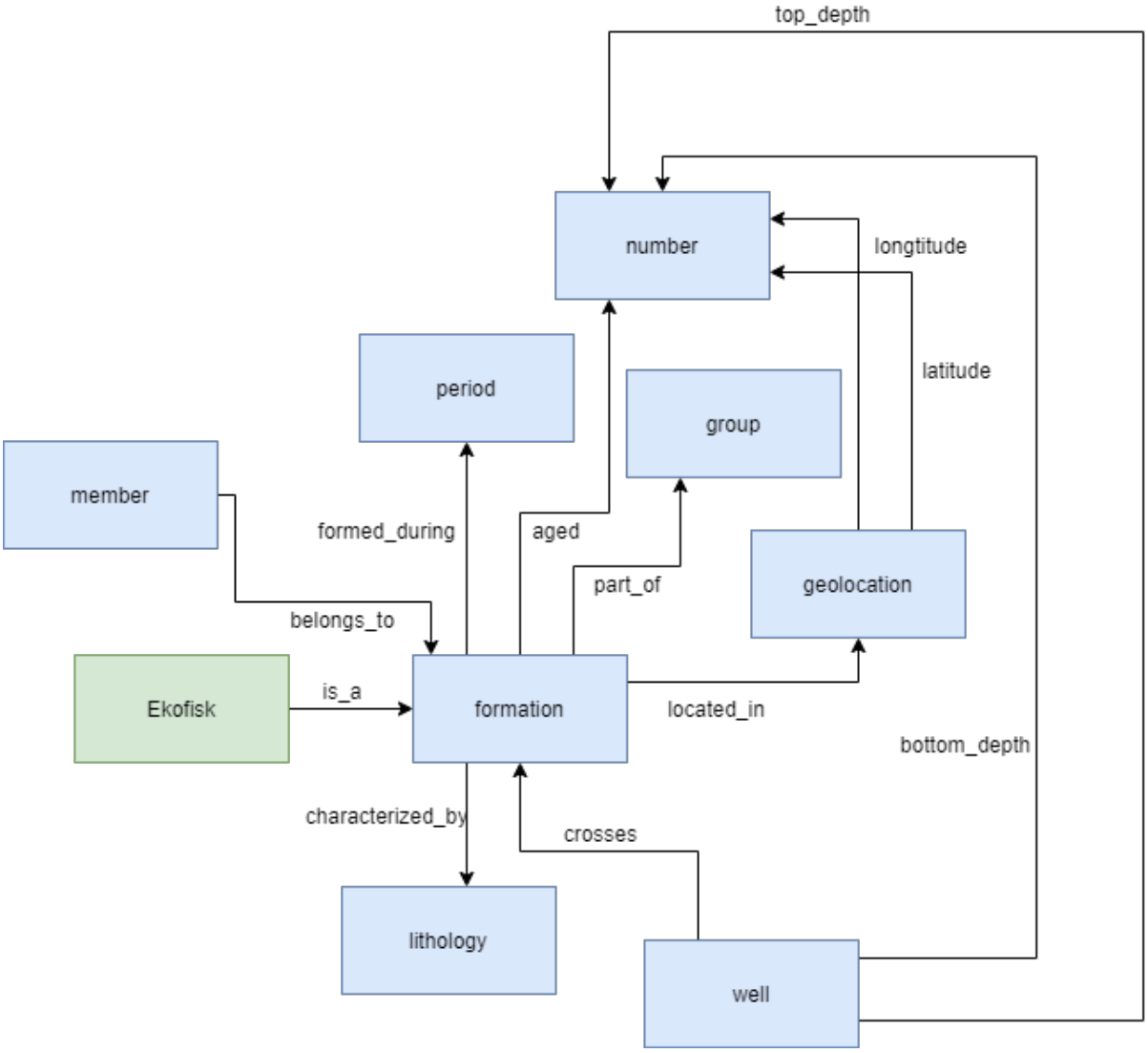

Figure 3.2: Knowledge base for questions about formations

Chapter 4

Data Collection and Preprocessing

This chapter will introduce the data sources and usage for populating the instance (TBox) of the knowledge graph. After, we will discuss the data preprocessing task that we did to obtain the structural data that easier for future use.

4.1 Data sources and usage

So far, our data sources include the text from Schlumberger Oilfield Glossary and the Norwegian Petroleum Directorate (NPD). We use Schlumberger's website to answer questions like: What is stratigraphy/ formation/ lithology. It served as a dictionary to help the user understand some particular domain-used terms. We answer most questions about an individual member, formation, and group, such as Ekofisk formation, Rogaland group using NPD. Consider it will also be other different data sources to be integrated in the future. We might not only consider just using the Norwegian website; the official page from the United Kingdom or the Netherlands might also be used because they collect the stratigraphy information from the North sea. However, it would not happen periodically, so manually integrating them into one format is our current solution.

Nevertheless, not NPD provides us rich data to gather knowledge from the north sea. In the beginning, we thought that a web page scraper is needed to collect the data; eventually, we found a data set provided on the website that can be downloaded directly in different formats includes CSV, excel, and XML. To answer the question we aim to answer, the data provided is enough. The data we download contains 179 lithology units, including member, formation, and group. Each contains their name and the description. However, the description box's information is still messy, and it contains a name, age, lithology, disposition environment, distribution, thickness, sources, etc. The next section will discuss how we clean the data in the description box and make it a well-structured file.

4.2 Data preprocessing with Python and OpenRefine

As mentioned before, we need to decompose the description box's information to extract the data we need. In each box, the description carries different amounts of information. We first used a python script to split the text using the delimiters and categorize them into either title or content. Here title indicates the entity name, and the content means the instance. The title includes Name, Type area, age, and the content will then be the rest of the paragraph that belongs to the title.

4.2.1 Identify problems in the dataset

However, we realized the terminology varies from different lithology units during the classification of the title and content, so we need to deal with un-unify terminologies, as the picture below shows. The common problems are:

- Interchangeable terminologies (A lot in our case)
- Capital letters and lower case letters
- Typo
- Special cases

We decide to use OpenRefine to cluster words based on their similarity and unify them into the most common form to have a unified term in the database.

4.2.2 Functions in OpenRefine

OpenRefine is a java-based web browser application that allows us to clean and reconcile data, even integrate with new incoming web data. OpenRefine provides a variety of powerful text clustering algorithms that we need.Including the following:

- key collision: fingerprint, ngram-fingerprint, metaphone3, cologne-phonetic
- nearest neighbor: levenshtein, PPM

Nearest neighbor algorithm works really well for our data set and able to detect the majority of the problem we identify in the previous step. It is more computationally expensive than key collision method, but it can detect more than just simple spelling error.

4.3 Potential Need for Text Mining

Analyzing the initial model created and contrasting it to the data that was collected, one thing becomes clearer a number of the questions can be answered using purely the data we managed to obtain, however some of the questions demand text mining with the purpose of obtaining the answers to the questions. Below we list all the questions that can be answered using the information that we have followed by the questions that assume natural language processing or text mining.

The questions that can be answered with the existing data:

- What is stratigraphy ?
- What are the wells crossing Ekofisk Formation ?
- What is the group of the Ekofisk Formation ?
- What is the top of the Ekofisk Formation for the well 1/3-1 ?
- What is the lithology of Ekofisk ?

- What are the members of Ekofisk ?
- What is the period and age of Ekofisk ?

The questions that demand text mining:

- Where is Ekofisk Formation ?
- Describe at Best Ekofisk Formation

Chapter 5

TBox Modelling

This chapter is dedicated to the modelling of the TBox model. In particular, we provide technical research results, regarding the existing tools for creating the TBox, as well as reasons why we selected the Protege tool in specific. Finally, we also illustrate the knowledge base that was created, listing the advanatges of the model against other alternatives.

5.1 Choosing the Ontology Editor

Choosing the right tool for creating the ontology for our knowledge base is one of the decisions of colossal importance and for that reason we performed a thorough research over different tools and mechanism in practice to choose the most corresponding one for our geological knowledge base.

We carefully studied the pros and cons of the following ontology editors:

- **Protégé** - A very popular and pluggable ontology editor. Protégé documentation could be found here.
- **NeON Toolkit** - An editor largely suitable for projects dealing with immense amounts of data. There are a number of plugins available for this editor. More information regarding the tool is available in the official documentation.
- **SWOOP** - A small, compact and relatively very simple ontology editor. The following documentation offers more information regarding the tool.
- **Neologism** - An online vocabulary editor and publishing platform. Details regarding the tool are available here.

After researching over each of the alternatives as well as trying them on practice we concluded that Protégé would be the most suitable tool for creating an OWL language-based knowledge base and exploiting the capabilities of description logic classifiers to maintain consistency of the ontology. The reasons for such a choice include but are not limited to the following advantages of Protégé editor:

- Protégé runs on a broad range of hardware platforms, hence not limiting the freedom of the user in any form.
- Protégé has an extremely active user community, which on one hand is a proof of the software's user-friendliness and comfort, on the other hand allows for larger opportunities for finding support in case of any intermediate questions of issues. This wouldn't be the case if we decided to use SWOOP or Neologism.

- Protégé contains a graphical editor for Logical OWL Expressions hence allowing for quick manipulation of logical expressions. The graphical object-oriented display of primitive and defined classes also assists the user freeing from a lot of bothers.
- Protégé contains a GUI and API which ease the work with classes, slots and instances.
- Has a direct access to reasoners, ensuring consistency checks, classification and instance classification. This reason particularly makes Protégé stand out against the other alternatives.
- Provides multi-user support, which makes it easier to work on a common project with a team
- Supports multiple storage formats, namely Clips, XML, RDF, and OWL
- Speaking of the alternatives, NeON Toolkit was the closest to our needs with the flexibility it could offer, however it would have made more sense to use that tool, if the project at hand was several times the size of what we worked with. In this particular case, we preferred the ease of work with Protégé, given it covered all the functionality that we needed, as mentioned above.
- As for the other alternatives, SWOOP was simple, compared to NeON Toolkit, however was not flexible enough to allow for the flexibility we wanted; and Neologism was more focused on vocabulary editing, which as well was not what we wanted.

In the next section we describe the process of creating the ontology with Protégé.

5.2 TBox Creation with Protégé

In this section we discuss the main logic behind the TBox that we created with the help of Protégé. Figure 4.1 graphically demonstrates the initial simplified form of the TBox that we obtained. The main idea is to replicate the structure we had obtained in the previous chapter, for storing the data in the knowledge base. The predicates in the image are defined as rectangles, while the circles represent the classes. Here is the list of the predicates, with their respective domains and ranges:

Object Properties:

- Crosses: The domain is Well, the range is Formation
- BelongsTo: The domain is Member, the range is Formation
- CharacterizedBy: The domain is Formation, the range is Lithology
- PartOf: The domain is Formation, the range is Group
- FormedDuring: The domain is Formation, the range is Period
- DefinedAs: The domain is StudyField, the range is Definition

- SubfieldOf: The domain is StudyField, the range is StudyField
- RelatedTo: The domain is StudyField, the range is StudyField

Datatype Properties:

- BottomDepth: The domain is Well, the range is double
- TopDepth: The domain is Well, the range is double
- Bounds: The domain is Formation, the range is String
- Name: The domain is Member, the range is Strnig
- Description: The domain is Member, the range is String
- DefinedAs: The domain is StudyField, the range is Definition
- SubfieldOf: The domain is StudyField, the range is StudyField
- RelatedTo: The domain is StudyField, the range is StudyField

Some changes were of course made, out of the specificity of the project. In particular, for Class **Formation** we have defined a datatype property called **Bounds**, the reasoning behind which we will explain in later chapters.

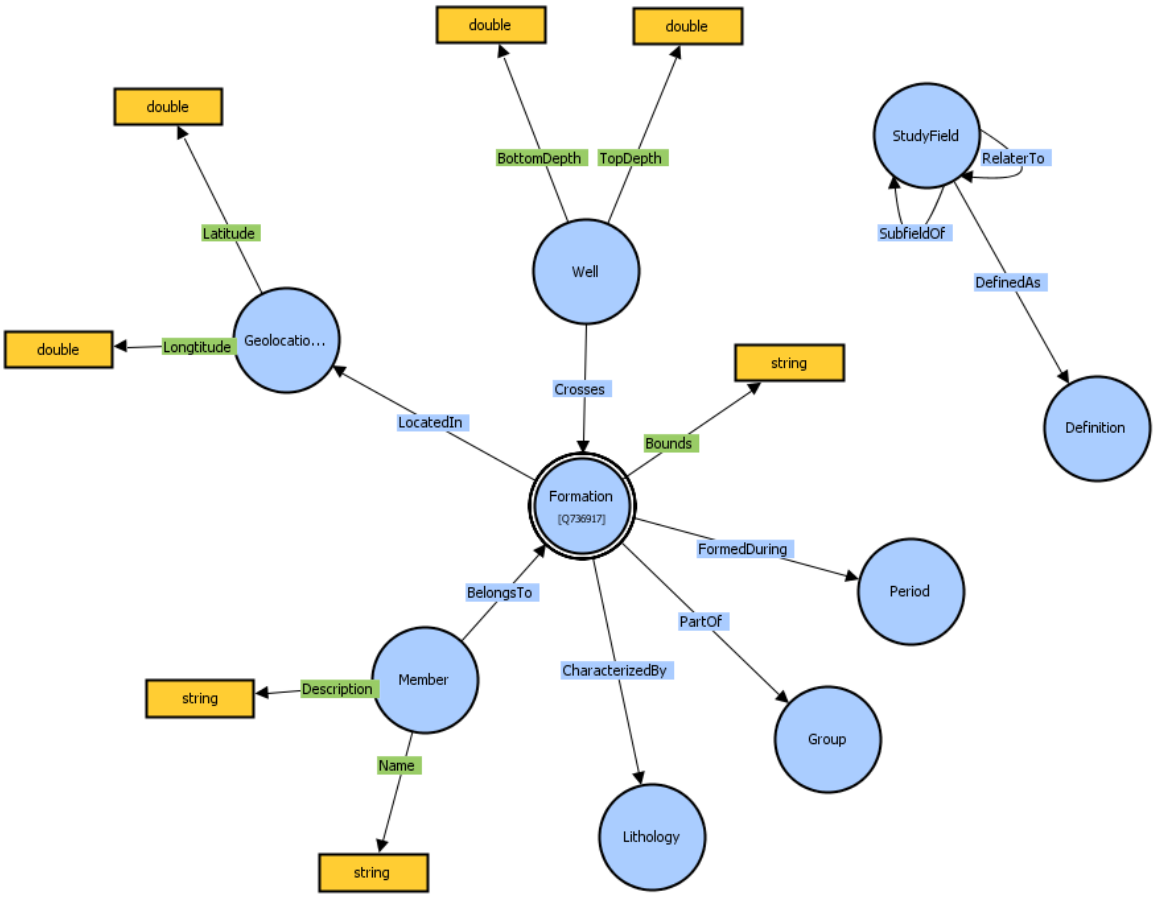

Figure 5.1: The TBox graphical representation

Chapter 6

Implement TBox and Populate ABox using Jena

In this chapter we will build a knowledge base according to the TBox model we illustrated in the last chapter and the data we collected to populate the knowledge base. We will mainly use Apache Jena to transform CSV files to our data instance.

6.1 Apache Jena Introduction

The TBox created with Protégé can not be used directly in the application. So we need an API Apache Jena, a Java framework for constructing semantic web and to ensure a range of inference engines or reasoners is implement. It support several reasoning choices with respect to their characteristics. For example, the transitive reasoner can be used for inference. We will use Java and Jena to fulfill the following functions:

- **Create TBox**
- **Populate ABox** - read the data files (csv) and then according to TBox model generate an rdf file to represent the knowledge base
- **Query Knowledge Graph** - Using Jena to interact with knowledge base directly, without using GraphDB

This chapter will focus on the first two bullet points only. In a later chapter, we will explain the querying process in detail.

6.2 Create TBox

For creating the TBox, we create the domain-specific terminologies. Include the ontology, data properties, conceptual elements, and constraints describe each resource' relationships like the last chapter but in Java code. Like an extract of the piece of code below.

```
static OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.
    OWL_DL_MEM);

final String ns = "http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#";

OntClass Formation = ontModel.createClass(ns + "Formation");
OntClass Period = ontModel.createClass(ns + "Period");
```

```
/*Defining the object properties*/
FormedDuring.addDomain(Formation);
FormedDuring.addRange(Period);

DatatypeProperty Name = ontModel.createDatatypeProperty(ns + "Name");
Name.addDomain(Formation);
Name.addDomain(Period);
}
```

6.3 Populate ABox

The ABox, also called the assertion box, contains facts about particular individuals and specifies a set of instances of their properties and relationships.

We read two different formatted CSV files. One file contains the general information of the formation, group, and member. Another contains list of each formation's wells number and their corresponding latitude,longitude. After planing the csv column and the field for them based on the TBox model. We will use Java and Jena to fulfill the following functions:

- Part1 -includes Formation's information

    1. ID, Formation, Name, Lithology, Period, Group, Member

- Part2 -includes the wells' information and location

    1. Well, Name (well's number), Longitude, Latitude

For ABox, we assign a unique URI to each entity with the unified namespace and the lower case formation name after removing non-English words. Jena also supports another standard format of file such as turtle, RDF, etc.; after all the instances are added to the ABox, we write it into an RDF file.

Chapter 7

Geolocation of Formations

In this chapter we deal solely with the question of finding the geolocation of a given formation. This question is more intricate compared to the others mentioned in the previous chapters, as it intends additional data collection and processing, as well as certain changes to be made in the initial TBox and ABox structures. All of those steps are described and explained below in further details.

7.1 Interpretation of the Question

The questions of locating a formation is more complicated than the others for a number of reasons. First of all, it's important to understand that each formation is crossed by a number of wells, and each of those wells has a different geolocation, defined by a unique pair of latitude and longitude. However, it wouldn't be a good practice to just provide information about all the wells crossing the formation.

Instead what we decided to do was to provide a bounding box for each formation, defined by comparing the latitude-longitude pairs of all the wells crossing the given geolocation and returning the box bounded by the north-most, south-most, east-most and west-most geolocations.

In order to get this done, the following steps were performed:

- first we needed to collect additional data regarding the well geolocations, since in the earlier stages we had not done that. This part is covered in section 7.2.
- Process the collected data into a usable form. The details are covered in section 7.3.
- Once the data was present we had to reconstruct the TBox to accommodate the bounding box structure. Section 7.3 covers the logic and reasoning behind what was done.
- Make necessary changes to the ABox structure. The details are present in section 7.3.

In the next subsections, we will talk about the details of achieving this.

7.2 Collecting Additional Geolocation Data

We found data about the wellbore geolocation in Wellbore Factpages. There we have a list of links for each of the wells, leading to a large table that contains information about the well, including the latitude of longitude of the well. The first step was creating a python script to scrape all the data we needed about the wells.

The main URL of the page was concatenated with the nid of each well to dynamically contract the URL per well, and table id was used for reaching the table:

```
def fetch_npi_id(np_id):
   with open(f"/home/Projects/gayane/bdrp/well_info/html_dump/{np_id}.html"
       ) as f:
      soup = BeautifulSoup(f.read(), 'html.parser')
      table = soup.find(id="8iT0S0T0")
      rows = table.find_all("tr")
      detail = {}
      for row in rows:
         cols = row.find_all("td")
         col_name = cols[0].text
         value = cols[1].text
         detail[col_name] = value
      results.append(detail)
```

The function below demonstrates how we obtained the nid for each well URL:

```
def get_nids():
   np_ids = []
   master_page = requests.get("https://factpages.npd.no/en/wellbore/
       pageview/exploration/all")
   master_list_page = BeautifulSoup(master_page.content, 'html.parser')
   tables = master_list_page.find(id="tvCarriers").find_all("table")
   for table in tqdm(tables):
      atag = table.find("a")
      np_ids.append(atag["href"].split("/")[-1])
   return np_ids
```

Since the number of wells was very large, we decided to implement parallelism for the loop in order to fasten the process of scraping, using the following function:

```
def download_to_local(ids):
   with parallel_backend('threading'):
      Parallel(n_jobs=50)(delayed(download_page)(npi_id) for npi_id in tqdm
          (ids))
   files_present = [path.split(".")[0] for path in
                os.listdir("/home/Projects/gayane/bdrp/well_info/html_dump"
                    )]
   for np_id in ids:
      if np_id not in files_present:
         download_page(np_id)
```

7.3 Adding Geolocation to the ABox and Querying

The idea here is creating another datatype property in our TBox- **Bounds**, which has **Formation** as a domain and a string as a range. To do this we added the following snippet into our *DefineTBox.java* file:

```
DatatypeProperty bounds = ontModel.createDatatypeProperty(ns + "Bounds");
bounds.setDomain(Formation);
bounds.setRange(XSD.xstring);
```

The trickier part was modifying the data injection process in order to process the raw data obtained and turn it into a bounding box. The data that was scraped from the website contained the original latitude and longitude values as strings, for example: 56° 59' 32" N or 2° 29' 47.66" E. As already mentioned above, the idea was to be able to find the north-, south-, west- and east- most points, since geolocation's latitude increases as you go north, whereas the longitude increases as you go East, so we could basically treat latitude as a north-south axis, and longitude as an east-west axis. In order to achieve that we need to compare those latitude and longitude values. The following code snippet shows what data processing was performed to obtain the hour, minute and second parts of the latitude/longitude to be able to compare them in an incremental order and return the biggest one:

```
public static String getCoordinate(String one, String two, boolean max)
    {
   if (one.equals(two)) {
      return one;
   }
   List<String> delimier = Arrays.asList("", "'", "''");
   String oneMutation = one;
   String twoMutation = two;
   for (String l : delimier) {
      String[] oneSplit = oneMutation.split(l);
      String[] twoSplit = twoMutation.split(l);
      if (Double.parseDouble(oneSplit[0].trim()) > Double.parseDouble(
          twoSplit[0].trim())) {
         return max ? one : two;
      }
      if (Double.parseDouble(oneSplit[0].trim()) < Double.parseDouble(
          twoSplit[0].trim())) {
         return max ? two : one;
      }
      oneMutation = oneSplit[1];
      twoMutation = twoSplit[1];
   }
   return null;
}
```

Then it was necessary to ensure the Bounds information was added to the well data, for which we added the following code snippet to the data importing function.

```
DatatypeProperty formationBound = ontModel.getDatatypeProperty(ns + "Bounds
    ");
// Formation name: <W: value, E: value, N: value, S: value>
      Map<String, Map<String, String>> boundedBox = new HashMap<>();
      List<String[]> lines = CSV.read(part2, ",");
      for (String[] line : lines) {
         String well_number = line[0];
         String top = line[1];
         String bottom = line[2];
         String formation_name = line[3];
         String latitude = line[4];
         String longitude = line[5];
```

COnsidering that in the initial model we lacked latitude and longitude information, at this point, in order to correctly insert the well geolocation data, the following two lines were added:

```
Literal latitude_string = ontModel.createTypedLiteral(latitude,XSDDatatype.
    XSDstring);
Literal longitude_string = ontModel.createTypedLiteral(longitude,
    XSDDatatype.XSDstring);
```

Finally using the following code snippet, the bounding box was created and a textual (string type) information stating the formation is bounded by certain latitude and longitude values, was injected into the knowledge base:

```
if (formation != null) {
            ontModel.add(well, Crosses, formation);
            ontModel.add(well, Name_prop, well_string);
            ontModel.add(well, Longitude_prop, longitude_string);
            ontModel.add(well, Latitude_prop, latitude_string);
            Map<String, String> existingValue = boundedBox.getOrDefault(
                cleanedFormationName, new HashMap<>());
            existingValue.put("N", getCoordinate(existingValue.getOrDefault
                ("N", latitude), latitude, true));
            existingValue.put("S", getCoordinate(existingValue.getOrDefault
                ("S", latitude), latitude, false));
            existingValue.put("E", getCoordinate(existingValue.getOrDefault
                ("E", longitude), longitude, true));
            existingValue.put("W", getCoordinate(existingValue.getOrDefault
                ("W", longitude), longitude, false));
            boundedBox.put(cleanedFormationName, existingValue);
         }
```

```
        //add top and bottom value
        Literal top_value = ontModel.createTypedLiteral(top, XSDDatatype.
            XSDdouble);
        ontModel.add(well, TopDepth_prop, top_value);
        Literal bottom_value = ontModel.createTypedLiteral(bottom,
            XSDDatatype.XSDdouble);
        ontModel.add(well, BottomDepth_prop, bottom_value);
     }
     boundedBox.forEach((key, value) -> {
        Individual formation = ontModel.getIndividual(ns + "Formation/" +
            key);
        String boundedValue = "";
        boundedValue = boundedValue + "Formation is bounded in NS by " +
            value.get("N") + " to " + value.get("S") + ".";
        boundedValue = boundedValue + "Formation is bounded in EW by " +
            value.get("E") + " to " + value.get("W") + ".";
        Literal boundedValueLiteral = ontModel.createTypedLiteral(
            boundedValue, XSDDatatype.XSDstring);
        ontModel.add(formation, formationBound, boundedValueLiteral);
```

With the new structure of the ABox already, the geolocation of the formation is easy query using the following simple SPARQL query:

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#>

SELECT ?boundary
WHERE
  { ?formation stratig:Name "Ekofisk Formation" ;
           stratig:Bounds ?boundary
  }
```

Listing 7.1: SPARQL query

The query above returns the bounding box of the Ekofisk formation as described in Figure 7.1.

	boundary
1	"Formation is bounded in NS by60° 47' 38.94'' N to 56° 7' 32.15'' N.Formation is bounded in EW by6° 10' 4.7'' E to 1° 32' 49.9'' E."

Figure 7.1: Geolocation of Ekofisk formation

Chapter 8

Connecting the Abox and TBox

This chapter is dedicated to the process of finally bringing together the TBox and Abox created in the previous chapters, connecting them and running a number of sample queries to confirm that the model is working. We took advanatge of the initially defined domain-relevant questions, translating them into SPARQL queries to run on top of the created knowledge base.

8.1 GraphDB

Once the ABox and TBox were created, the next step was to connect them and ensure that the knowledge base is sufficient to answer the main questions that we're interested in. Apache Jena was used for connecting the ABox and TBox and then we decided to use the GraphDB graph database for loading the knowledge base and querying it using the standard SPARQL language.GraphDB is a very efficient and popular graph database, owning a very comprehensible GraphDB Documentation that covers every aspect of use of the tool.

8.2 Translating Questions Into SPARQL Queries

In order to test whether the created knowledge base was working properly to obtain the answers to our questions, the initial list of our questions of interest was translated into SPARLQL queries and run over the database. All the results that we obtained were later confirmed using the official website. The questions were translated as follows:

- What is stratigraphy ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#>

SELECT ?def
WHERE
  { ?s stratig:DefinedAs ?definition .
    ?definition stratig:Description ?def .
  }
```

Listing 8.1: SPARQL query

- What are the wells crossing Ekofisk Formation ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#>

SELECT ?well
WHERE
 { ?well stratig:Crosses ?formation .
   ?formation stratig:Name <http://www.semanticweb.org/user/ontologies
       /2020/11/Stratigraphy_in_North_Sea#Formation/ekofisk_formation>
 }
```

Listing 8.2: SPARQL query

- What is the group of the Ekofisk Formation ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
   Stratigraphy_in_North_Sea#>

SELECT ?name
WHERE
 { <http://www.semanticweb.org/user/ontologies/2020/11/
     Stratigraphy_in_North_Sea#Formation/ekofisk_formation>
          stratig:PartOf ?group .
   ?group stratig:Name ?name
 }
```

Listing 8.3: SPARQL query

- What is the top of the Ekofisk Formation for the well 1/3-1 ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
   Stratigraphy_in_North_Sea#>

SELECT ?top
WHERE
 { <http://www.semanticweb.org/user/ontologies/2020/11/
     Stratigraphy_in_North_Sea#Well/ekofisk_formation/1/3-1>
          stratig:TopDepth ?top
 }
```

Listing 8.4: SPARQL query

- What is the lithology of Ekofisk ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
   Stratigraphy_in_North_Sea#>

SELECT ?name
```

```
WHERE
 { <http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#Formation/ekofisk_formation>
         stratig:CharacterizedBy ?lithology .
  ?lithology stratig:Name ?name
 }
```

Listing 8.5: SPARQL query

- What are the members of Ekofisk ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
   Stratigraphy_in_North_Sea#>

SELECT ?member
WHERE
 { ?member stratig:BelongsTo <http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#Formation/
    ekofisk_formation> }
```

Listing 8.6: SPARQL query

- What is the period and age of Ekofisk ?

```
PREFIX stratig: <http://www.semanticweb.org/user/ontologies/2020/11/
   Stratigraphy_in_North_Sea#>

SELECT ?name
WHERE
 { <http://www.semanticweb.org/user/ontologies/2020/11/
    Stratigraphy_in_North_Sea#Formation/ekofisk_formation>
         stratig:FormedDuring ?period .
  ?period stratig:Name ?name
 }
```

Listing 8.7: SPARQL query

- Give the best summary for Ekofisk formation ?

  This question is slightly trickier than the previous ones. The idea is to collect information, such as the period and age of the formation, the group of the formation, the lithology description and characteristics of the formation, as well as the geolocation of the formation, and form an answer for the user in a string format, that in a human language presents a general summary containing all the information about a formation. In order to integrate the query in a human language way we use BIND operation of SPARQL, which allows for string and variable concatenations to provide the best possible summarization of the available information for each formation.

```
SELECT ?res
WHERE
  {
?formation stratig:Name "Ekofisk␣Formation" ;
              stratig:FormedDuring ?period .
?period stratig:Name ?period_name .
?formation stratig:Bounds ?boundary ;
              stratig:PartOf ?group .
?group stratig:Name ?group_name .
?formation stratig:CharacterizedBy ?lithology .
?lithology stratig:Name ?lithology_name
BIND(concat("The␣formation␣was␣originated␣in␣", concat(?period_name,
    concat("␣period␣and␣belongs␣to␣", concat(?group_name, concat(".␣",
     concat(?boundary, concat("␣and␣lithology␣is", ?lithology_name)))))
    )) AS ?res)
  }
```

Listing 8.8: SPARQL query

A sample answer for this query is given in the figure 8.1. It contians the answer GraphDB provides after running the query for Ekofisk formation.

	res
1	"The formation was originated in Danian. period and belongs to shetland group. Formation is bounded in NS by60° 47' 38.94'' N to 56° 7' 32.15'' N.Formation is bounded in EW by6° 10' 4.7'' E to 1° 32' 49.9'' E. and lithology is"In the type well, the formation comprises white, tan or beige, hard, dense, sometimes finely crystalline limestones, although softer chalky textures are also present. The formation usually consists of white to light grey, beige to brownish, mudstones or wackestones with occasional packstones/grainstones and pisolitic horizons, often alternating with argillaceous chalks, chalky limestones or limestones. Thin beds of grey, calcareous, often pyritic shales or clays are most common in the lower part while brownish-grey cherts occur rarely to abundantly throughout the formation. ""

Figure 8.1: The process of parsing query

Chapter 9

Question answering with Jena

The chapter will introduce how the question is parsing and answering by interacting with the database using the SPARQL query.

9.1 Question parsing

As the diagram below shows, when the user is entering a question, the process is triggered. It will first check if the question contains the keyword "stratigraphy" because it is independent of all the other questions. If "stratigraphy" is not part of the user's input, we continually check if the question contains any string in our formation databases, such as Agat formation and Brygge formation. If it does not exist in our database, either the user has a typo or does not answer the user's question. However, if the question contains the formation name, we will be further checking which keywords are in the question. For example, it can be group, lithology, top, period, etc. In this case, we can recognize what question the user is asking, then construct a query based on the user's input and query the database.

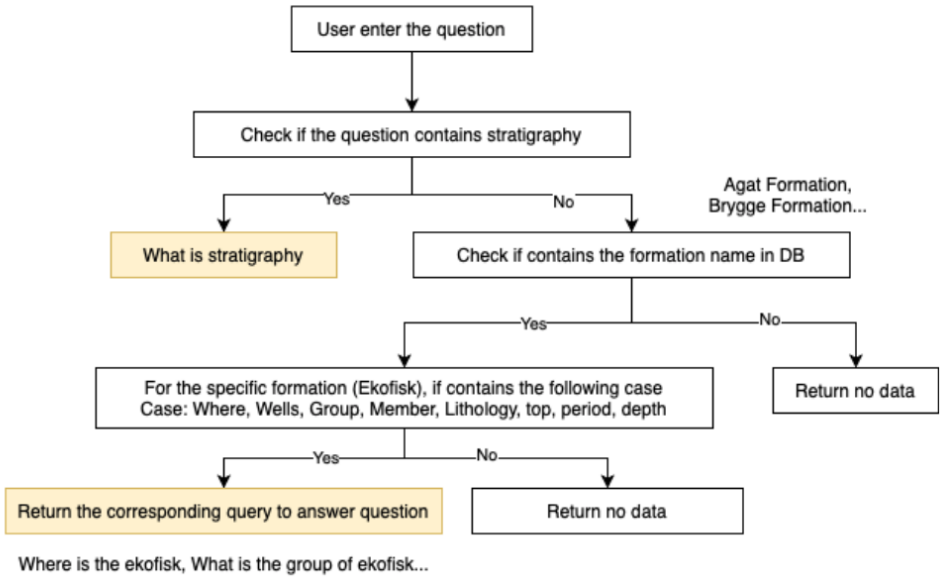

Figure 9.1: The process of parsing query

9.1.1 Question answering

For questions that are formation specific, we tokenize input question and check if the user's input includes our database's formation name. If so, we then check what topic the user wants to know.In order to answer the following question, the query need to be perform. Later, the question and its query construction will be preset. The following list contains the question and the corresponding query.

- What is stratigraphy?
  It's independent from the rest of classes and not formation specific. So we can directly use the following paragraph as answer without waiting for the query to respond.

- What are the wells crossing XXX Formation ?

  If the user's input question contain the string "well" and also a formation name that in our database. We can return the query that can answer this question.

```
if(input_question.contains("well")){
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString ="PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>"+
   "SELECT ?wellnum "+
   "WHERE { ?well stratig:Crosses "+formation_uri+" . "+
   "?well stratig:Name ?wellnum . }";
}
```

- What is the group of the XXX Formation ?

  If the user's input question contain the string "group" and also a formation name that in our database. We can return the query that can answer this question.

```
if(input_question.contains("group")){
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString ="PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>"+
   "SELECT ?name "+
   "WHERE {"+formation_uri+" stratig:PartOf ?group .\n" +
   "?group stratig:Name ?name . }";

}
```

- What are the members of XXX ?

  If the user's input question contain the string "member" and also a formation name that in our database. We can return the query that can answer this question.

```
if(input_question.contains("member")) {
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString = "PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>" +
"SELECT ?name " +
"WHERE { ?member stratig:BelongsTo "+formation_uri+" ."+
"?member stratig:Name ?name . }";
}
```

- What is the lithology of XXX ?

  If the user's input question contain the string "lithology" and also a formation name that in our database. We can return the query that can answer this question.

```
if(input_question.contains("lithology")){
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Lithology/"+subject+">";
QueryString ="PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>"+
   "SELECT ?name "+
   "WHERE {"+formation_uri+" stratig:Name ?name . }";

}
```

- What is the period and age of XXX ?

  If the user's input question contain the string "age" or "period" and also a formation name that in our database. We can return the query that can answer this question.

```
if(input_question.contains("age")||input_question.contains("period")){
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString ="PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>"+
      "SELECT ?name "+
      "WHERE {"+formation_uri+" stratig:FormedDuring ?period ." +
      "?period stratig:Name ?name . }";

}
```

- What is the top of the XXX Formation for the well XXX ?

  If the user's input question contain the string "well" and "/" and also a formation name that in our database. We can return the query that can answer this question because for every well, the name format is like "1/3-1"

```
if(input_question.contains("/") && input_question.contains("well")){
String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString ="PREFIX stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>"+
      "SELECT ?top "+
      "WHERE { "+formation_uri+" stratig:PartOf stratig:TopDepth ?top
          . }";

}
```

- Give the best summary for XXX formation ?

  If the user's input question contains the strings "summary" and/or "best", we return the following query that contains the summary of a formation as an end result:

```
if (input_question.contains("summary") || input_question.contains("
    best")) {
            String formation_uri = "<http://www.semanticweb.org/user/
                ontologies/2020/11/Stratigraphy_in_North_Sea#Formation/"
                 + subject + ">";
            QueryString = "PREFIX stratig:<http://www.semanticweb.org/
                user/ontologies/2020/11/Stratigraphy_in_North_Sea#>" +
                  "SELECT ?res " +
                  "WHERE {" + formation_uri + " stratig:FormedDuring ?
                      period ." +
                  "?period stratig:Name ?period_name ." +
                  formation_uri + "stratig:Bounds ?boundary ." +
                  formation_uri + "stratig:PartOf ?group ." +
                  "?group stratig:Name ?group_name ." +
                  formation_uri + "stratig:CharacterizedBy ?lithology .
                      " +
                  "?lithology stratig:Name ?lithology_name . " +
                  "BIND( concat(\"The formation was originated in \"," "
                       +
                  "concat(?period_name, concat(\" period and belongs to
                      \"," " +
                  "concat(?group_name, concat(\". \", concat(?boundary,
                      " +
                  "concat(\" and lithology is\", ?lithology_name)))))))
                      AS ?res )}";
```

- Where is XXX Formation ?

  If the user's input question contain the string "where" or "location" and also a formation name that in our database. We can return the query that can answer

this question. Consider to include "location" because the question can be raised by asking what is the location of XXX formation.

```
if(input_question.contains("where")||input_question.contains("location
    ")) {

String formation_uri="<http://www.semanticweb.org/user/ontologies
    /2020/11/Stratigraphy_in_North_Sea#Formation/"+subject+">";
QueryString = "PREFIX␣stratig:<http://www.semanticweb.org/user/
    ontologies/2020/11/Stratigraphy_in_North_Sea#>" +
   "SELECT␣?boundary␣" +
   "WHERE␣{␣"+formation_uri+"␣stratig:Bounds␣?boundary␣.␣}";
}
```

9.1.2 Execute the query

After parsing the question and constructing the query. This step here is to execute the query that the system build and return the output. Here, the output will be exactly the same as using Sparql query in GraphDB. If we want a more human-like response, further modification is needed.

```
//Initialize the connection for querying the graph
InputStream in = new FileInputStream(new File("Abox_output.rdf"));

// Create an empty inmemory model and populate it from the graph
Model model = ModelFactory.createMemModelMaker().createModel("model");
model.read(in,null); // null base URI, since model URIs are absolute
in.close();

Query query = QueryFactory.create(Query_String); //execute the query
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();
ResultSetFormatter.out(System.out, results, query); // Output results
qe.close(); // Free up resources

}
```

Chapter 10

Interface realization

We aim to create a chatbot environment for users to interact with our application for the realization. This chapter will introduce the tools we use to integrate the question answering system with websites to create a web application as the user interface and their techniques.

10.1 Tools introduction

This section will introduce and briefly explain the tools that help us deploy our application on the website.

10.1.1 Servlet

Servlet is a server-side Java program module that manages client's requests and performs the servlet interface. It allows the Java program to be executed within a web container; in our case, we use tomcat, which allows handling client requests and server responses straightforwardly. Servlet using the client's data input could be a text input or button click, any data that we can typically obtain through GET and POST request. We use another application layer to manage the cookies and session parameters. Also, we connect it back to our question-answering codes to querying the knowledge base and acquire to answer to sends the output in string form back to the client in text formats by using a Java Server Pages (JSP) file.

10.1.2 Apache Tomcat Server

Apache Tomcat is an open source implementation, it's use as the container of the Java Servlet, JSP and Java Web Socket technologies. In order to run our servlet application, we need to install and correctly configured Apache Tomcat web server.

10.2 Processes

We will first introduce the mandatory files for configuration, then we explain how we use JSP file to return the message to the clients.

10.2.1 Configuration files to setup the application

For every servlet application, we need 3 following files:

- web.xml -To define deployment descriptor (DD) is our first thing to do. We specifies how an application should be deployed in a web environment in this document.

- index.html file - include all the CSS style sheets and possibly JavaScript code, the HTML file is the first interface when user open the website.
- Servlet file - It is used to handle the client's request. We need two inputs to the process, the request object and the response object. The idea is simple; the former informs the servlet about the client's request, and the latter responds to the client. As the code below, we using getParameter to get the user's input, then we set the answer using setAttribute then display it on the result page.

  Its definition is as follows:

```
@WebServlet(
      name = "askingQuestion",
      urlPatterns = "/AskQuestion"
)
public class Servlet extends HttpServlet {

   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse
       resp) throws ServletException, IOException {

      String input_question = req.getParameter("LIST");

      input_question=input_question.toLowerCase();
      String answering = Question_Parsing.question_parsing(
          input_question);

      req.setAttribute("answer", answering);
      req.setAttribute("question", input_question);
      RequestDispatcher view = req.getRequestDispatcher("result.jsp");
      view.forward(req, resp);

   }
}
```

10.2.2 Connect to knowledge base and querying

We can see we pass the user input to the query parsing function here from the code above. We will do the exact technique explained in the last chapter to answer user's questions.

10.2.3 Rendering Page to respond the question

After created configuration files, we need to define the JSP files to rendering the form and displaying answer once the user submit the question. As the code from last subsection, we redirect to result.jsp with attributes for both answering and question. It's because we want the result page display the content of these two parameter, later you will see that

JSP file is really similar to HTML file but with some extra expression from JSP technology as the following example of screenshot and the corresponding JSP code.

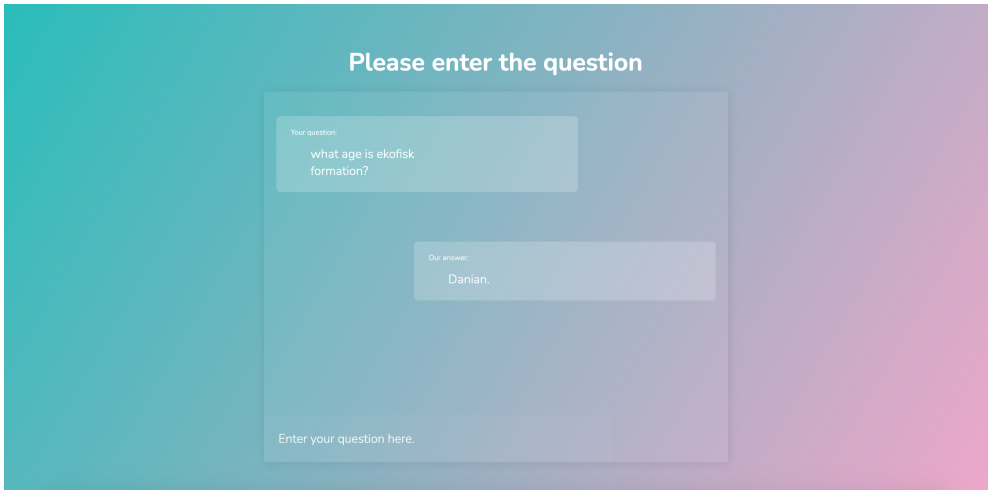

Figure 10.1: The JSP page screenshot

```
...
<div>
   <div>
      <p class="name">{{"Your␣question:␣"}}</p>
      <p class="message"><% String question =(String)request.getAttribute("
          question");
       out.println(question);%></p>
   </div>
</div>
<div>
   <div>
      <p class="name">{{"Our␣answer:␣"}}</p>
      <p class="message"><% String answer =(String)request.getAttribute("
          answer");
         out.println(answer);%></p>
   </div>
</div>

<form method="post" action="AskQuestion">
   <input name="LIST" type="text" placeholder="Enter␣your␣question␣here.">
...
```

Bibliography

[1] Kamran Munir, M. Sheraz Anjum. *The use of ontologies for effective knowledge modelling and information retrieval.* Applied Computing and Informatics Volume 14, Issue 2, July 2018, Pages 116-126

[2] Factpages - Norwegian Petroleum Directorate. (n.d.). Factpages - NPD. https://factpages.npd.no/en

[3] P. (2016, July 2). Servlet JSP Tutorial. JournalDev. https://www.journaldev.com/2114/servlet-jsp-tutorial

[4] Apache Jena - Jena Ontology API. (2020). Apache Jena. https://jena.apache.org/documentation/ontology/

[5] Protege Documentation, http://protegeproject.github.io/protege/

[6] SPARQL Documentation, https://www.w3.org/TR/rdf-sparql-query/