Geological Knowledge Base Construction

Big Data Research Project

Gayane VARDANYAN
Yuhsuan CHEN

—

prepared at CentraleSupelec

Defended on September 2020

Advisor: 1st name LAST NAME - host institution email

Contents

Chapter 1

Introduction

1.1 Project Definition

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.

1.2 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.

[Lettmayr 2011]

Chapter 2

Modelling the Knowledge Base

2.1 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. The entities are 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 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
- Wells have top and bottom depths (numerical values)

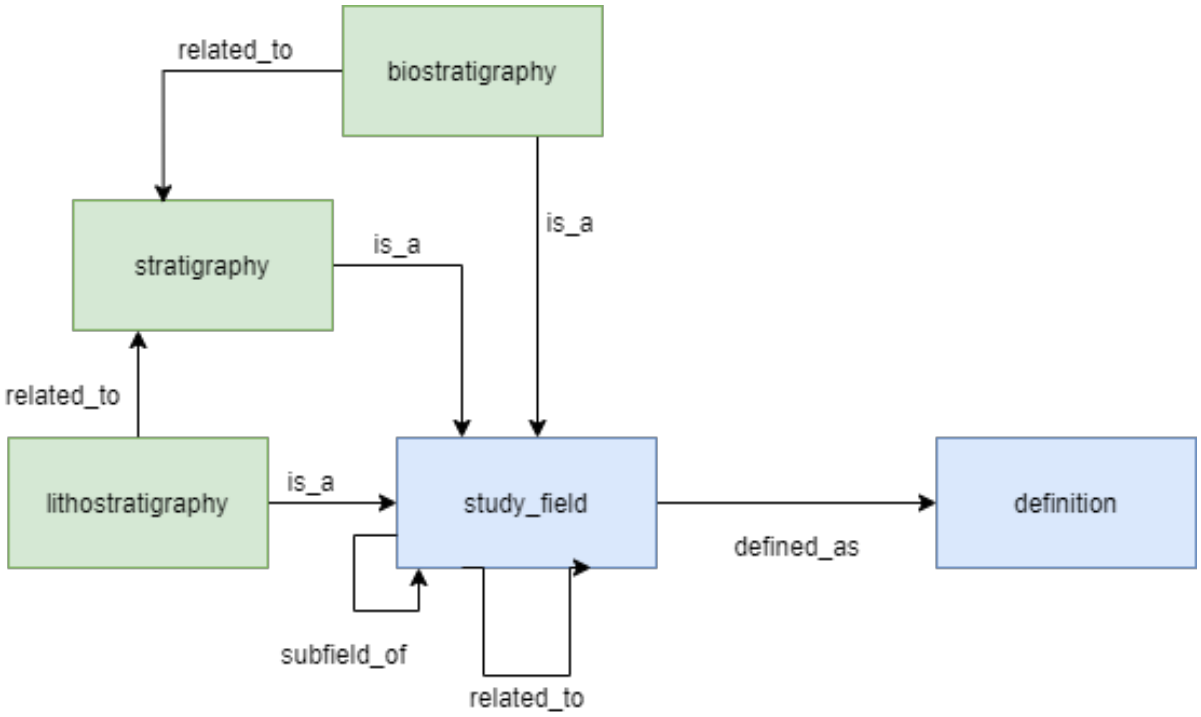


Figure 2.1: Knowledge base for questions about stratigraphy

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

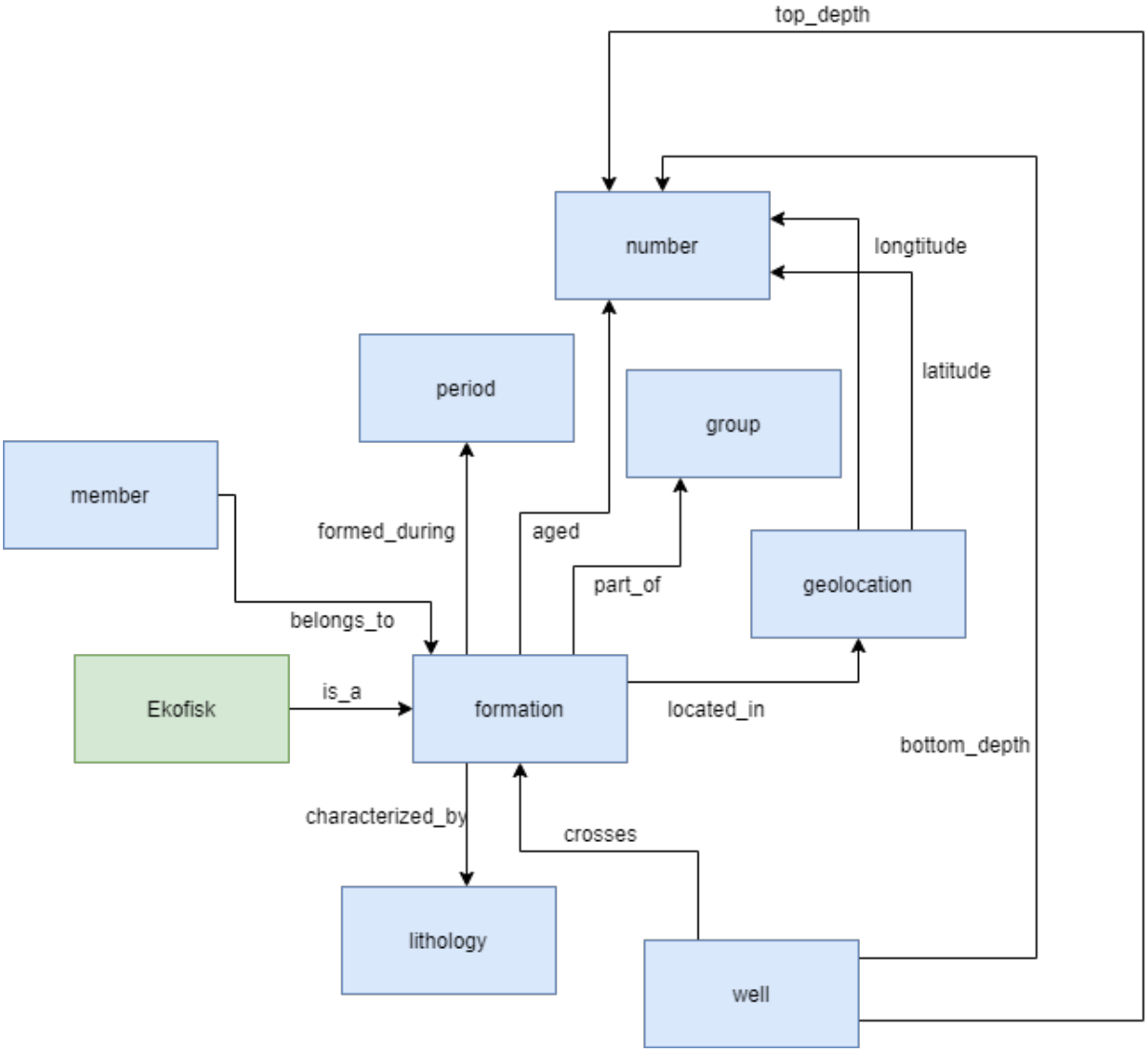


Figure 2.2: Knowledge base for questions about formations

Chapter 3

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.

3.1 Data sources and usage

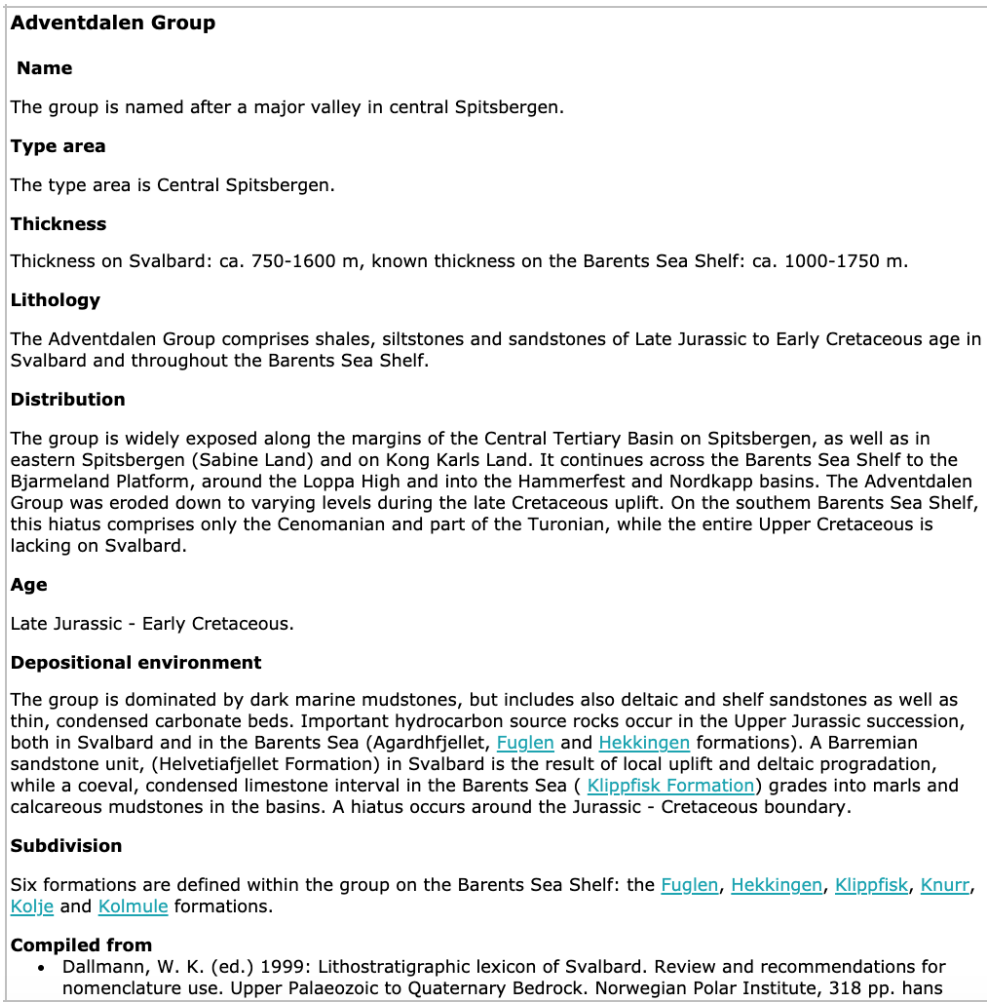
Our data sources include fact sheets on the Norwegian Petroleum Directorate (NPD) websites, Wikipedia, Schlumberger Oilfield Glossary. We can find the most stratigraphy-related answers from chapter one on NPD websites, including stratigraphy and oilfield aspects. As for Wikipedia and Schlumberger Oilfield Glossary, like a dictionary, used to answer definition questions like What is stratigraphy?

3.2 Data preprocessing

We downloaded the csv file from the NPD website that contains the whole group, formation, and members' detailed information. In this case, we do not need to iterate scraping every website. However, the csv file is not the format we can use later. The first problem we have is all the information is store in the same cell as the picture below. We use a python script to split the data 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, and so on. The content will then be the rest of the paragraph. We realized that terminology various from different lithology units during the first categorization, and each cell carries different amounts of information. After acknowledging the problem, another question is, how are we dealing with un-unify terminology? As the picture below shows, some terminologies are interchangeable, which happened a lot in our case. So the solution is based on their similarity; we cluster words and unify them into the most common form. Then we store the raw term and the unified term into the dictionary. Thus, when categorizing, we will use the dictionary as a reference.

3.3 Potential Need for Text Mining

Analyzing the initial model created and contrasting it to the data that was collected, one thing becomes clear- 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.



Adventdalen Group

Name

The group is named after a major valley in central Spitsbergen.

Type area

The type area is Central Spitsbergen.

Thickness

Thickness on Svalbard: ca. 750-1600 m, known thickness on the Barents Sea Shelf: ca. 1000-1750 m.

Lithology

The Adventdalen Group comprises shales, siltstones and sandstones of Late Jurassic to Early Cretaceous age in Svalbard and throughout the Barents Sea Shelf.

Distribution

The group is widely exposed along the margins of the Central Tertiary Basin on Spitsbergen, as well as in eastern Spitsbergen (Sabine Land) and on Kong Karls Land. It continues across the Barents Sea Shelf to the Bjarmeland Platform, around the Loppa High and into the Hammerfest and Nordkapp basins. The Adventdalen Group was eroded down to varying levels during the late Cretaceous uplift. On the southem Barents Sea Shelf, this hiatus comprises only the Cenomanian and part of the Turonian, while the entire Upper Cretaceous is lacking on Svalbard.

Age

Late Jurassic - Early Cretaceous.

Depositional environment

The group is dominated by dark marine mudstones, but includes also deltaic and shelf sandstones as well as thin, condensed carbonate beds. Important hydrocarbon source rocks occur in the Upper Jurassic succession, both in Svalbard and in the Barents Sea (Agardhfjellet, Fuglen and Hekkingen formations). A Barremian sandstone unit, (Helvetiafjellet Formation) in Svalbard is the result of local uplift and deltaic progradation, while a coeval, condensed limestone interval in the Barents Sea (Klippfisk Formation) grades into marls and calcareous mudstones in the basins. A hiatus occurs around the Jurassic - Cretaceous boundary.

Subdivision

Six formations are defined within the group on the Barents Sea Shelf: the Fuglen, Hekkingen, Klippfisk, Knurr, Kolje and Kolmule formations.

Compiled from

- Dallmann, W. K. (ed.) 1999: Lithostratigraphic lexicon of Svalbard. Review and recommendations for nomenclature use. Upper Palaeozoic to Quaternary Bedrock. Norwegian Polar Institute, 318 pp. hans

Figure 3.1: Example of a single cell

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	146	• Depositional environment (130 rows) • Depositional environments (15 rows) • Depositional envitronment (1 rows)	☐	Depositional environment
2	10	• Reference sections (9 rows) • Reference section (1 rows)	☐	Reference sections
2	114	• Well reference section (80 rows) • Well reference sections (34 rows)	☐	Well reference section
2	78	• Basal Stratotype (77 rows) • Basal stratotype 1) (1 rows)	☐	Basal Stratotype
2	2	• UPPER MEMBER OF THE EKOFISK FORMATION (1 rows) • Upper member of the Tor Formation (1 rows)	☐	Upper Member
2	39	• Characteristics of the upper boundary (38 rows) • Characteristics of the tipper boundary (1 rows)	☐	Characteristics of the upper
2	2	• LOWER MEMBER OF THE EKOFISK	☐	Lower Member

Figure 3.2: Example of various terminologies

	ID	title	content
0	Adventdalen Group	Name	The group is named after a major valley in central Spitsbergen.
1	Adventdalen Group	Well type section	The type area is Central Spitsbergen.
2	Adventdalen Group	Thickness	Thickness on Svalbard: ca. 750-1600 m, known thickness on the Barents Sea Shelf: ca. 1000-1750 m.
3	Adventdalen Group	Lithology	The Adventdalen Group comprises shales, siltstones and sandstones of Late Jurassic to Early Cretaceous age
4	Adventdalen Group	Distribution	The group is widely exposed along the margins of the Central Tertiary Basin on Spitsbergen, as well as in easte
5	Adventdalen Group	Age	Late Jurassic - Early Cretaceous.
6	Adventdalen Group	Depositional environment	The group is dominated by dark marine mudstones, but includes also deltaic and shelf sandstones as well as t
7	Adventdalen Group	Subdivision	Six formations are defined within the group on the Barents Sea Shelf: the Fuglen, Hekkingen, Klippfisk, Knurr,
8	Adventdalen Group	Compiled from	Dallmann, W. K. (ed.) 1999: Lithostratigraphic lexicon of Svalbard. Review and recommendations for nomencla
0	Agat Formation	Name	Named after the gas-condensate Agat Discovery in Norwegian block 35/3.
1	Agat Formation	Well type section	Norwegian well 35/3-4 from 3589 m to 3345 m, coordinates N 61°51'54.54", E 03°52'26.99" (Fig 5.19), 95 m o
2	Agat Formation	Well reference section	Norwegian well 35/3-5 from 3620 m to 3219 m, coordinates N 61°47'46.71",E 03°54'44.01" (Fig 5.20). 65 m of
3	Agat Formation	Thickness	In the type well the gross thickness of the formation is 244 m, and in the reference well 401 m. The gross thick
4	Agat Formation	Lithology	In the type well the formation consists of white to light grey, fine- to medium-grained, moderately to well-sorte
5	Agat Formation	Lower member	The lower boundary is defined where sandstones become the dominant lithology and is placed at the base of t
6	Agat Formation	Upper member	The upper boundary is placed at the top of the upper sandstone layer. This boundary is especially distinct on t
7	Agat Formation	Distribution	The formation is encountered in the area around the Måløy Fault Blocks in Norwegian blocks 35/3-36/1 (Fig 5.2
8	Agat Formation	Age	Aptian-Albian (possibly Early Cenomanian).
9	Agat Formation	Depositional environment	Marine environment influenced by gravity flows of sediment.
10	Agat Formation	Source	Isaksen, D. and Tonstad, K. (eds.) 1989: A revised Cretaceous and Tertiary lithostratigraphic nomenclature for t

Figure 3.3: Data set after cleaning

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 4

TBox Modelling

4.1 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
- **NeON Toolkit** - An editor largely suitable for projects dealing with immense amounts of data. There are a number of plugins available for this editor
- **SWOOP** - A small, compact and relatively very simple ontology editor
- **Neologism** - An online vocabulary editor and publishing platform.

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
- 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
- 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

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

4.2 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.

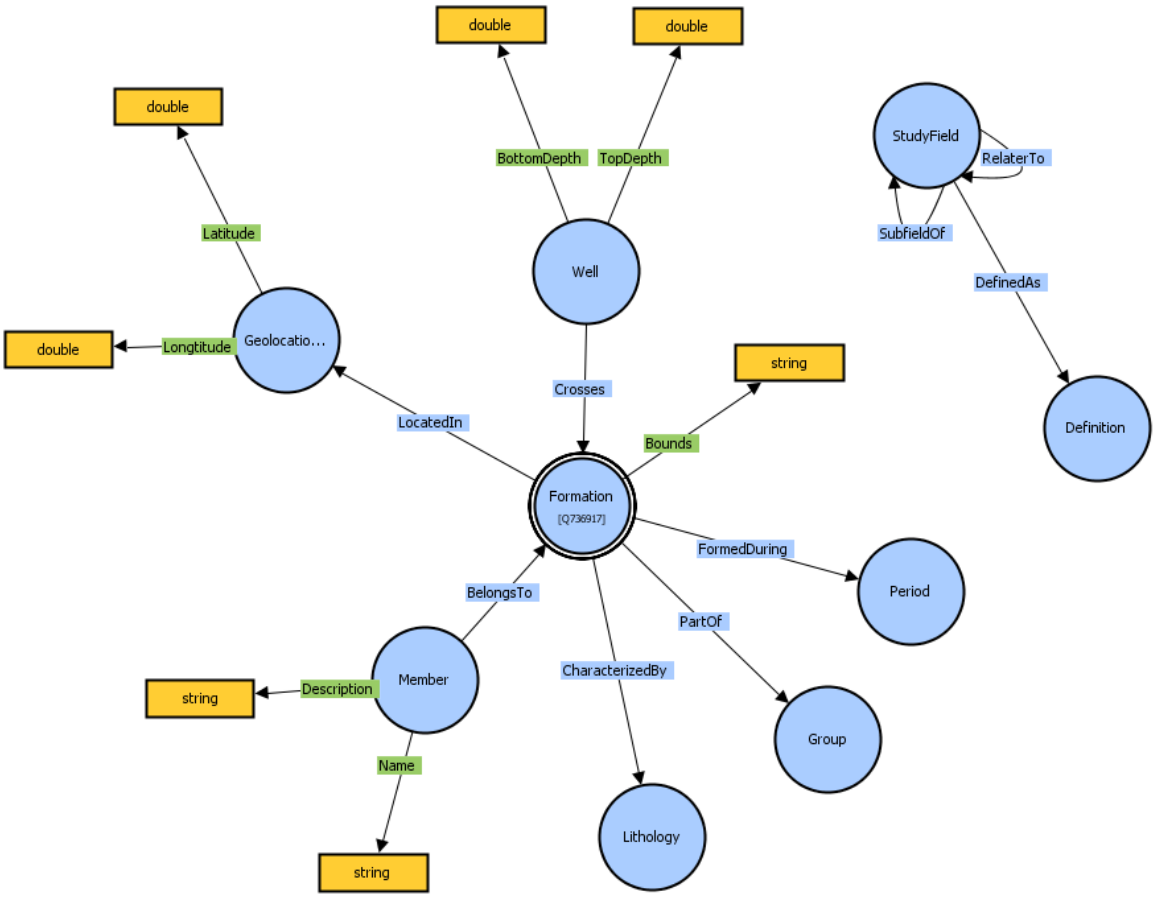


Figure 4.1: The TBox graphical representation

Chapter 5

Implement TBox and Populate ABox using Jena

5.1 Jena Introduction

The knowledge base is equal to TBox (ontology) plus ABox (data). 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 ensuring a range of inference engines or reasoners is implemented.

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

5.2 Create TBox

Apache Jena is a Java framework for constructing Semantic Web and to ensure a range of inference engines or reasoners is implement. In the following code, we create the domain-specific terminologies. The data properties, conceptual elements, and constraints describe each resource' relationships.

courier listings, xcolor

```
package com.company.define;

import org.apache.jena.ontology.*;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.vocabulary.XSD;

public class DefineTBox {
   static OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec
       .OWL_DL_MEM);

   public static void run() {
```

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

OntClass Definition = ontModel.createClass(ns + "Definition");
OntClass Formation = ontModel.createClass(ns + "Formation");
OntClass Geolocation = ontModel.createClass(ns + "Geolocation");
OntClass Group = ontModel.createClass(ns + "Group");
OntClass Lithology = ontModel.createClass(ns + "Lithology");
OntClass Member = ontModel.createClass(ns + "Member");
OntClass Period = ontModel.createClass(ns + "Period");
OntClass StudyField = ontModel.createClass(ns + "StudyField");
OntClass Well = ontModel.createClass(ns + "Well");

/*Defining the object properties*/
ObjectProperty CharacterizedBy = ontModel.createObjectProperty(ns + "
    CharacterizedBy"); //done
CharacterizedBy.addDomain(Formation);
CharacterizedBy.setRange(Lithology);

ObjectProperty FormedDuring = ontModel.createObjectProperty(ns + "
    FormedDuring"); //done
FormedDuring.addDomain(Formation);
FormedDuring.addRange(Period);

ObjectProperty PartOf = ontModel.createObjectProperty(ns + "PartOf");
     //done
PartOf.addDomain(Formation);
PartOf.addRange(Group);

ObjectProperty LocatedIn = ontModel.createObjectProperty(ns + "
    LocatedIn");
LocatedIn.addDomain(Formation);
LocatedIn.addRange(Geolocation);



ObjectProperty Crosses = ontModel.createObjectProperty(ns + "Crosses"
    );//done
Crosses.addDomain(Well);
Crosses.addRange(Formation);


ObjectProperty DefinedAs = ontModel.createObjectProperty(ns + "
    DefinedAs");
DefinedAs.addDomain(StudyField);
```

```
DefinedAs.addRange(Definition);
ObjectProperty SubfieldOf = ontModel.createObjectProperty(ns + "
    SubfieldOf");
SubfieldOf.addDomain(StudyField);
SubfieldOf.addRange(StudyField);
ObjectProperty RelaterTo = ontModel.createObjectProperty(ns + "
    RelaterTo");
RelaterTo.addDomain(StudyField);
RelaterTo.addRange(StudyField);


ObjectProperty BelongsTo = ontModel.createObjectProperty(ns + "
    BelongsTo");
BelongsTo.addDomain(Member);
BelongsTo.addRange(Formation);


/*Defining the data properties*/

DatatypeProperty BottomDepth = ontModel.createDatatypeProperty(ns + "
    BottomDepth"); //done
BottomDepth.setDomain(Well);
BottomDepth.setRange(XSD.xdouble);

DatatypeProperty Description = ontModel.createDatatypeProperty(ns + "
    Description");
Description.addDomain(Member);
Description.addDomain(Lithology);
Description.addDomain(Definition);
Description.setRange(XSD.xstring);

DatatypeProperty Latitude = ontModel.createDatatypeProperty(ns + "
    Latitude");
Latitude.setDomain(Geolocation);
Latitude.setDomain(Well);
Latitude.setRange(XSD.xstring);

DatatypeProperty Longitude = ontModel.createDatatypeProperty(ns + "
    Longitude");
Longitude.setDomain(Geolocation);
Latitude.setDomain(Well);
Longitude.setRange(XSD.xstring);

DatatypeProperty Name = ontModel.createDatatypeProperty(ns + "Name");
     //done
```

```
        Name.addDomain(Formation);
        Name.addDomain(Group);
        Name.addDomain(Member);
        Name.addDomain(Period);
        Name.addDomain(StudyField);
        Name.addDomain(Well);
        Name.setRange(XSD.xstring);

        DatatypeProperty TopDepth = ontModel.createDatatypeProperty(ns + "
            TopDepth");//done
        TopDepth.setDomain(Well);
        TopDepth.setRange(XSD.xdouble);

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

    }


}
```

5.3 Populate ABox

The ABox (assertional box) contains facts about particular individuals and specifies a set of instances of their properties and relationships.

In the following code, we read 2 different formatted csv files. One contains the general information of the formation, group, and member.

We will use Java and Jena to fulfill the following functions:

- Part1 -includes Formation's information

    1. ID
    2. Formation
    3. Name
    4. Lithology
    5. Period
    6. Group
    7. Member

- Part2 -includes the wells' information and location

  1. Well
  2. Name (well's number)
  3. Longitude
  4. Latitude

```
import com.company.utils.CSV;
import com.company.utils.Utils;
import org.apache.jena.datatypes.xsd.XSDDatatype;
import org.apache.jena.ontology.DatatypeProperty;
import org.apache.jena.ontology.Individual;
import org.apache.jena.ontology.ObjectProperty;
import org.apache.jena.ontology.OntClass;
import org.apache.jena.rdf.model.Literal;
import org.apache.jena.riot.Lang;
import org.apache.jena.riot.RDFDataMgr;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.company.define.DefineTBox.ontModel;


public class DefineABox {
   final static String ns = "http://www.semanticweb.org/user/ontologies
       /2020/11/Stratigraphy_in_North_Sea#";

   OntClass geolocationClass = ontModel.getOntClass(ns + "Geolocation");
   ObjectProperty LocatedIn_prop = ontModel.getObjectProperty(ns + "
       LocatedIn");

   public static void run() {

      try {
         import_Part1(); //Formation, Group, Lithology, Period
         import_Part2();
         //OutputStream out = new FileOutputStream("output-test.ttl");
```

```
        //RDFDataMgr.write(out, ontModel, Lang.TURTLE);
        OutputStream out = new FileOutputStream("Abox_output.rdf"); //
            output the ABox
        RDFDataMgr.write(out, ontModel, Lang.RDFXML); //output the ABox
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
}

private static void import_Part1() throws IOException {
    String formation_basic = "src/Test_Abox.csv";
    OntClass lithologyClass = ontModel.getOntClass(ns + "Lithology");
    OntClass periodClass = ontModel.getOntClass(ns + "Period");
    OntClass groupClass = ontModel.getOntClass(ns + "Group");
    OntClass memberClass = ontModel.getOntClass(ns + "Member");
    OntClass formationClass = ontModel.getOntClass(ns + "Formation");
    OntClass DefinitionClass = ontModel.getOntClass(ns + "Definition");
    OntClass StudyFieldClass = ontModel.getOntClass(ns + "StudyField");
    DatatypeProperty Name_prop = ontModel.getDatatypeProperty(ns + "Name"
        );
    ObjectProperty CharacterizedBy_prop = ontModel.getObjectProperty(ns +
         "CharacterizedBy");
    ObjectProperty FormedDuring_prop = ontModel.getObjectProperty(ns + "
        FormedDuring");
    ObjectProperty PartOf_prop = ontModel.getObjectProperty(ns + "PartOf"
        );
    ObjectProperty BelongsTo_prop = ontModel.getObjectProperty(ns + "
        BelongsTo");
    ObjectProperty DefinedAs_prop = ontModel.getObjectProperty(ns + "
        DefinedAs");
    List<String[]> lines = CSV.read(formation_basic, ",/t,");

    for (String[] line : lines) {

        String formation_name = line[3];
        String group_content = line[2];
        String lithology_content = line[4];
        String period_content = line[5];
        Utils.addFormation(Utils.cleanURI(formation_name));

        // Inserting the formation, its group and related info
        if (formation_name.contains("formation") || formation_name.
            contains("Formation")) {
           Individual formation = formationClass.createIndividual(ns + "
```

```
            Formation/" + Utils.cleanURI(formation_name));
        Literal formation_Name_string = ontModel.createTypedLiteral(
            formation_name, XSDDatatype.XSDstring);
        ontModel.add(formation, Name_prop, formation_Name_string);
        if (group_content.contains("group")) {
           Individual group = groupClass.createIndividual(ns + "Group/"
                + Utils.cleanURI(formation_name));
           Literal group_content_string = ontModel.createTypedLiteral(
               group_content, XSDDatatype.XSDstring);
           ontModel.add(group, Name_prop, group_content_string);
           ontModel.add(formation, PartOf_prop, group);
        }
        //Inserting the lithology
        Individual lithology = lithologyClass.createIndividual(ns + "
            Lithology/" + Utils.cleanURI(formation_name));
        Literal lithology_content_string = ontModel.createTypedLiteral(
            lithology_content, XSDDatatype.XSDstring);
        ontModel.add(lithology, Name_prop, lithology_content_string);

        //Inserting the period
        Individual period = periodClass.createIndividual(ns + "Period/"
             + Utils.cleanURI(formation_name));
        Literal period_content_string = ontModel.createTypedLiteral(
            period_content, XSDDatatype.XSDstring);
        ontModel.add(period, Name_prop, period_content_string);

        ontModel.add(formation, CharacterizedBy_prop, lithology);
        ontModel.add(formation, FormedDuring_prop, period);

      } else if (formation_name.contains("group") || formation_name.
          contains("Group")) {
        Individual group = groupClass.createIndividual(ns + "Group/" +
            Utils.cleanURI(formation_name));
        Literal group_content_string = ontModel.createTypedLiteral(
            group_content, XSDDatatype.XSDstring);
        ontModel.add(group, Name_prop, group_content_string);
        //Inserting the lithology
        Individual lithology = lithologyClass.createIndividual(ns + "
            Lithology/" + Utils.cleanURI(formation_name));
        Literal lithology_content_string = ontModel.createTypedLiteral(
            lithology_content, XSDDatatype.XSDstring);
        ontModel.add(lithology, Name_prop, lithology_content_string);

        //Inserting the period
        Individual period = periodClass.createIndividual(ns + "Period/"
```

```
            + Utils.cleanURI(formation_name));
        Literal period_content_string = ontModel.createTypedLiteral(
            period_content, XSDDatatype.XSDstring);
        ontModel.add(period, Name_prop, period_content_string);

        ontModel.add(group, CharacterizedBy_prop, lithology);
        ontModel.add(group, FormedDuring_prop, period);

    } else if(formation_name.contains("member")||formation_name.
        contains("Member")){
        Individual member = memberClass.createIndividual(ns + "Member/"
            + Utils.cleanURI(formation_name));
        Literal member_content_string = ontModel.createTypedLiteral(
            formation_name, XSDDatatype.XSDstring);
        ontModel.add(member, Name_prop, member_content_string);
        if(group_content.contains("formation")){
            Individual formation = formationClass.createIndividual(ns +
                "Formation/" + Utils.cleanURI(group_content));
            Literal formation_Name_string = ontModel.createTypedLiteral(
                group_content, XSDDatatype.XSDstring);
            ontModel.add(formation, Name_prop, formation_Name_string);
            ontModel.add(member, BelongsTo_prop, formation);
        }

    }


    //Inserting the studyfield
    Individual studyfield = StudyFieldClass.createIndividual(ns + "
        StudyField/" + Utils.cleanURI(formation_name));
    Literal study_field_string = ontModel.createTypedLiteral("
        Stratigraphy", XSDDatatype.XSDstring);
    ontModel.add(studyfield, Name_prop, study_field_string);


    //Inserting the definition
    Individual definition = DefinitionClass.createIndividual(ns + "
        Definition/" + Utils.cleanURI(formation_name));
    Literal definition_string = ontModel.createTypedLiteral("
        Stratigraphy␣is␣a␣geology␣study␣involved␣the␣study␣of␣the␣rock
        ␣layer(strata).␣It␣includes␣three␣main␣subfields,␣
        lithostratigraphy,␣biostratigraphy␣and␣chronostratigraphy.",
        XSDDatatype.XSDstring);
    ontModel.add(definition, Name_prop, definition_string);
    ontModel.add(studyfield, DefinedAs_prop, definition);
```

```
    }
    System.out.println("Part1 done");
}

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;
}

//Wells information
private static void import_Part2() throws IOException {
    String part2 = "src/wells_location_formation.csv";
    OntClass WellClass = ontModel.getOntClass(ns + "Well");
    ObjectProperty Crosses = ontModel.getObjectProperty(ns + "Crosses");
    DatatypeProperty Name_prop = ontModel.getDatatypeProperty(ns + "Name"
        );
    DatatypeProperty TopDepth_prop = ontModel.getDatatypeProperty(ns + "
        TopDepth");
    DatatypeProperty BottomDepth_prop = ontModel.getDatatypeProperty(ns +
         "BottomDepth");
    DatatypeProperty Longitude_prop = ontModel.getDatatypeProperty(ns + "
        Longitude");
    DatatypeProperty Latitude_prop = ontModel.getDatatypeProperty(ns + "
```

```
         Latitude");
      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];


         // Inserting the well
         String cleanedFormationName = Utils.cleanURI(formation_name);
         Individual formation = ontModel.getIndividual(ns + "Formation/" +
            cleanedFormationName);
         Individual well = WellClass.createIndividual(ns + "Well/" + Utils.
            cleanURI(formation_name + "/" + well_number));
         Literal well_string = ontModel.createTypedLiteral(well_number,
            XSDDatatype.XSDstring);
         Literal latitude_string = ontModel.createTypedLiteral(latitude,
            XSDDatatype.XSDstring);
         Literal longitude_string = ontModel.createTypedLiteral(longitude,
            XSDDatatype.XSDstring);

         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);

     });
     System.out.println("Part2 done");

  }
}
```

Chapter 6

Geolocation of Formations

6.1 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, first we needed to collect additional data regarding the well geolocations, since in the earlier stages we had not done that. Once the data was present we had to reconstruct the ABox to accommodate the bounding box structure. In the next subsections, we will talk about the details of achieving this.

6.2 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)
```

6.3 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];
```

To correctly insert the well 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 6.1: 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 6.1: Geolocation of Ekofisk formation

Chapter 7

Connecting the Abox and TBox

7.1 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.

7.2 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 7.1: 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 7.2: 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 7.3: 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 7.4: 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 7.5: 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 7.6: 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 7.7: SPARQL query

- Where is Ekofisk Formation ?

As for this last question, since the geolocation data is more complicated and has specificity, unlike the others above, before being able to answer it, we had to perform a number of steps, which are described in the next chapter.

Chapter 8

Parsing Question to Query Using Jena

8.1 Question Parsing Process

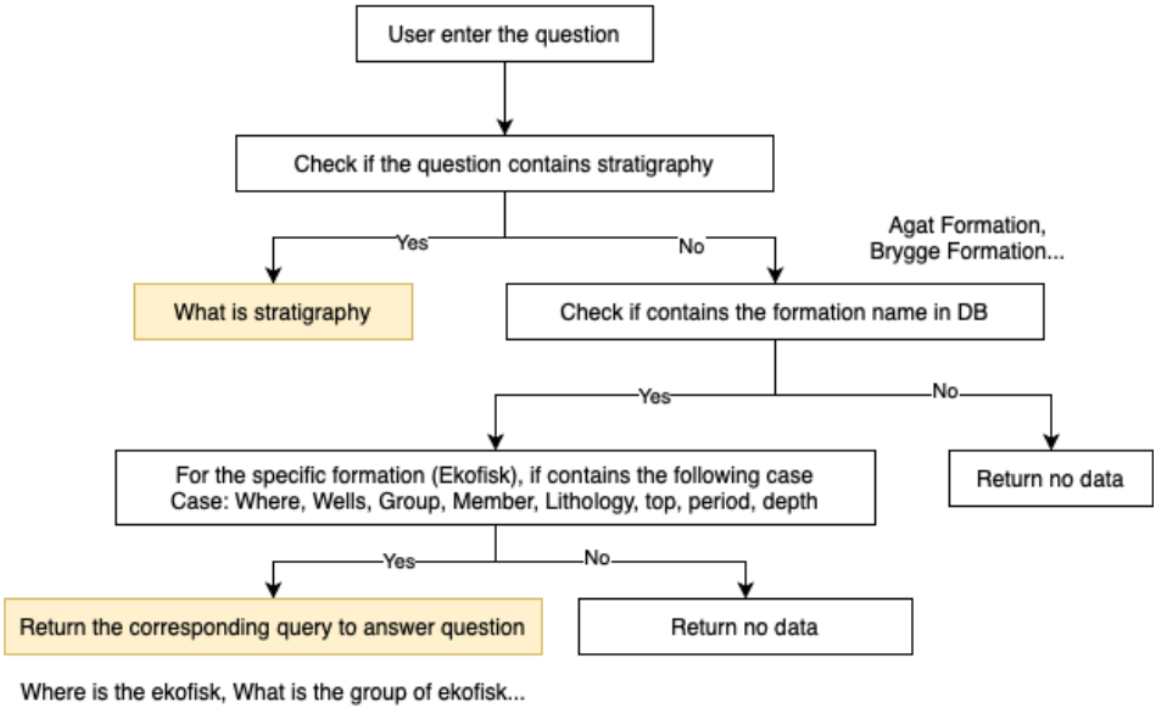


Figure 8.1: The process of parsing query

8.2 Initial Setup and queries

8.2.1 Question answer without Query

As the workflow chart above, there is no need query to find the answer, as 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.

"Stratigraphy is a geology study involved the study of the rock layer(strata). It includes three main subfields, lithostratigraphy, biostratigraphy and chronostratigraphy."

8.2.2 Question answer with Query

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.

- 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␣
          .␣}";

}
```

- 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␣.␣}";
}
```

8.2.3 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

}
```

Bibliography

[Lettmayr 2011] Christian F Lettmayr and Tarja Riihimäki. The benefits of vocational education and training. Luxembourg: Publications Office of the European Union, 2011.