CentraleSupélec

université PARIS-SACLAY

Big Data Research Project

Emir NURMATBEKOV
Anant GUPTA

Geology Knowledge Base Construction

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

Schlumberger

BDA

Introduction

Problem Statement

The task at hand was a true research oriented one where the idea was to build a system: a knowledge base in particular, capable of responding to questions about the stratigraphy in the North Sea. The real challenge was to look beyond the building of a knowledge base following the traditional approach of creating triple-store via modelling and loading of TBOX and ABOX respectively.

Bringing us to our first question:

- **What is a Knowledge Graph?**
  According to Ehrlinger et al.[1], *"A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge."*

  A simplified version of understanding a knowledge graph like a programmatic way to model a knowledge domain with the help of subject-matter experts, data interlinking, and machine learning algorithms. The easiest example is probably the box you see in Google's results.

  A knowledge graph is usually built using the existing databases as a base in an attempt to link all data together at web-scale combining both structured information (i.e. the list of students enrolled in a course: BDRP) or unstructured (articles like how to get admit to BDMA).

- **Why are they so important?**
  Knowledge graphs have been developed serving the requirement of executing the task with or act upon the information depending on the context. For examples, they can help in identifying fraud, keeping track of records, writing novels etc. In recent years, knowledge graphs are gaining more attention since their usage can be further extended by machine learning and artificial intelligence techniques.

- **What is the difference between a Knowledge Graph and a Graph Database?**
  Knowledge graphs are nothing but data. Hence, they have to be stored, managed, extended, quality-assured for providing quality answers to queries. This mandates databases and components on top, which are usually implemented in the Semantic Middleware Layer. This 'sits' on the database and at the same time offers service endpoints for integration with third-party systems.

  Thus graph databases form the cornerstone of every knowledge graph. Typically, these are technologies based either on the Resource Description Framework (RDF), a W3C standard, or on Labelled Property Graphs (LPG).

Related Work

Traditional approach

So ‘Knowledge Graph’ is nothing but the aggregation of ontology, usually referred to as ‘TBox’, and the data, usually referred to as ‘ABox’.

Virtual Knowledge Graph

Virtual Knowledge graph approach uses Ontology Based Data Access (OBDA) which is a research focusing on accessing data through ontologies. In our case, we used a system called OnTop which accesses relational data through ontologies.

Benefits:

- Flexible data model
- Flexible query language
- Inference
- Speed, Volume and features

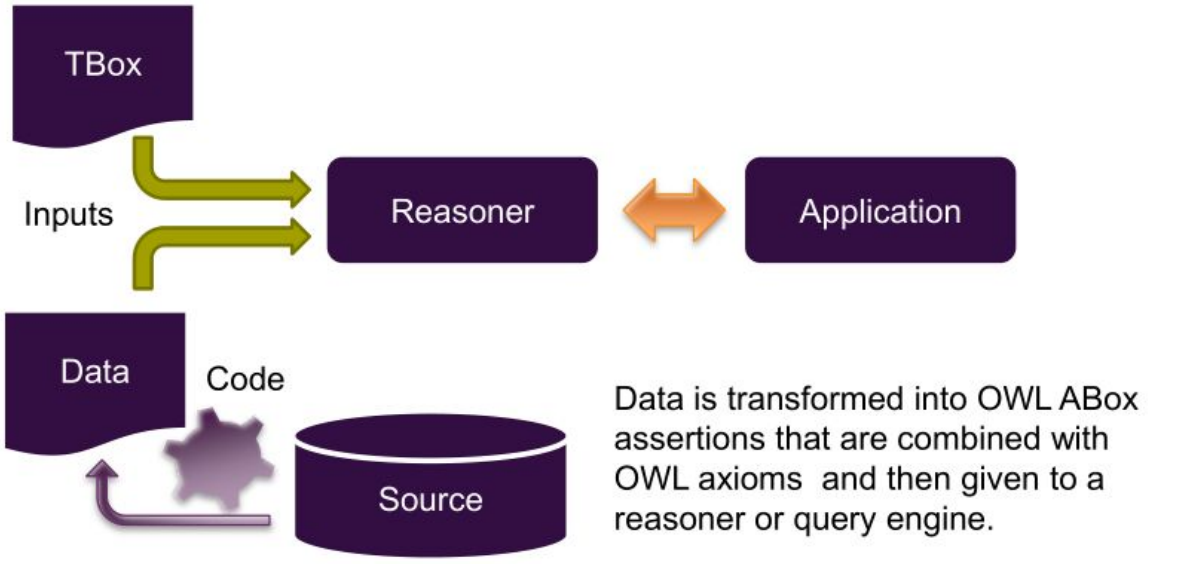
Possible applications:

- Semantic query answering
- Data Integration
- Semantic search

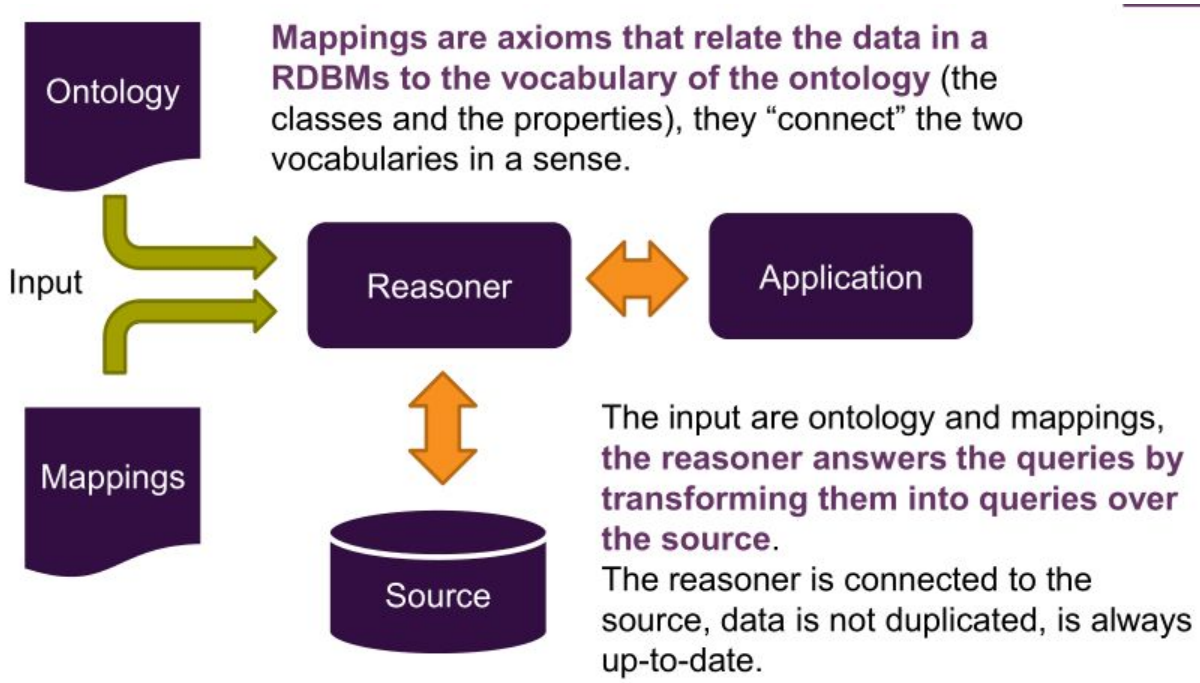
There are two approaches to OBDA:

- ETL
- On-the-fly

The first approach



The second approach

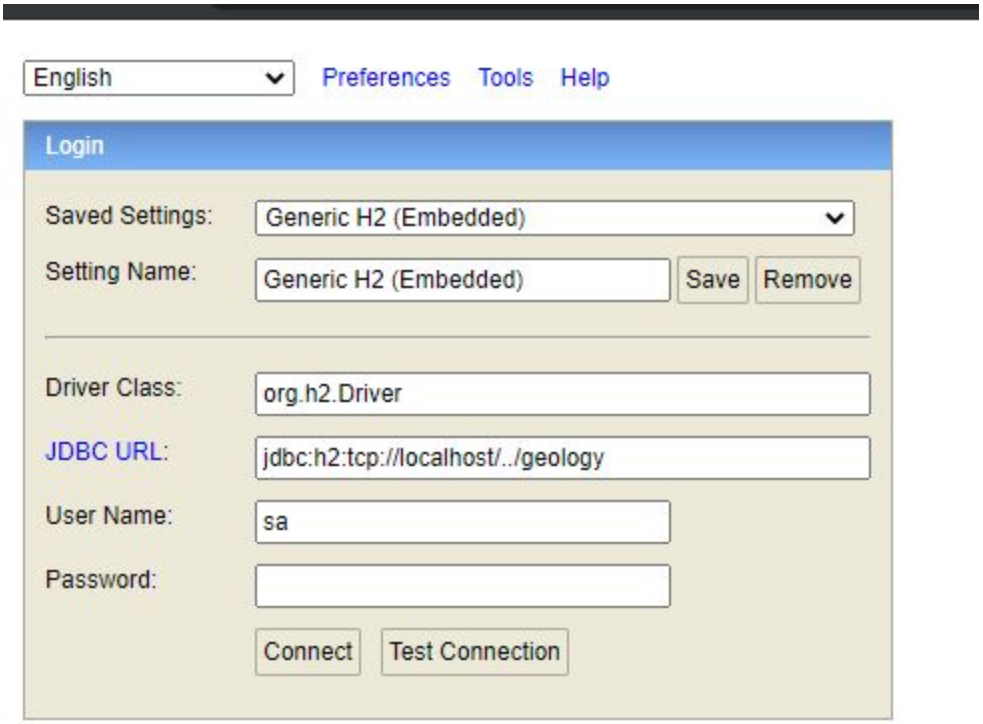


Ontop is a platform for querying relational databases through owl/rdfs ontologies on-the-fly via SPARQL. It has two main components:

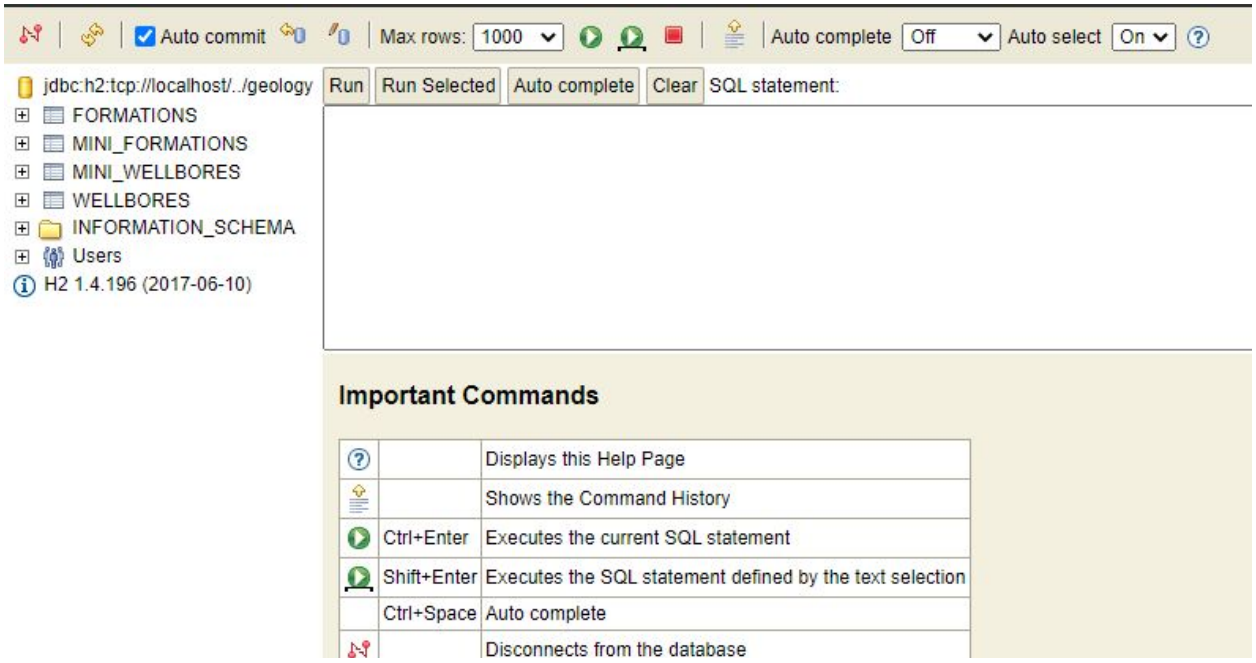
- Quest - a reasoner for answering SPARQL queries, supports OWL 2 QL inference and a powerful mapping language.
- ontopPro - Protege plugin for editing mappings and for using Quest from Protege.

Setting up Ontop

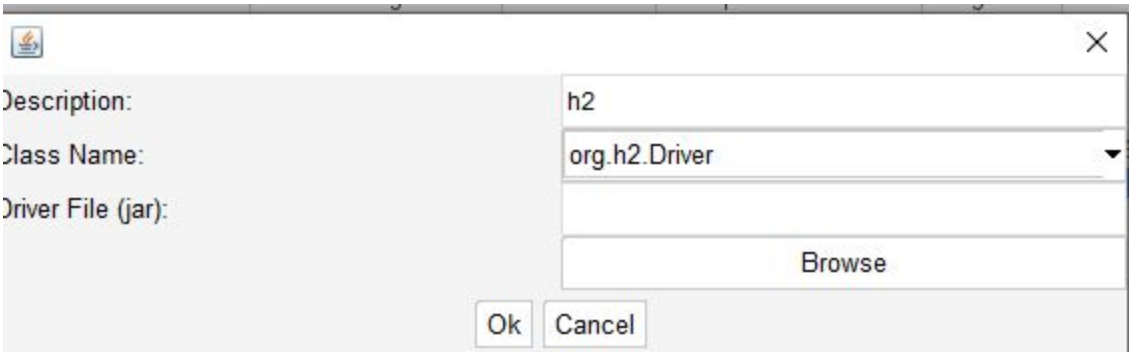
1. First, we need to setup the RDBMS we will be using. We will be using the H2 Database. It is easy to install, we just need to unzip the package, then run the script file h2w.bat. That will redirect us to the main page of the RDBMS.



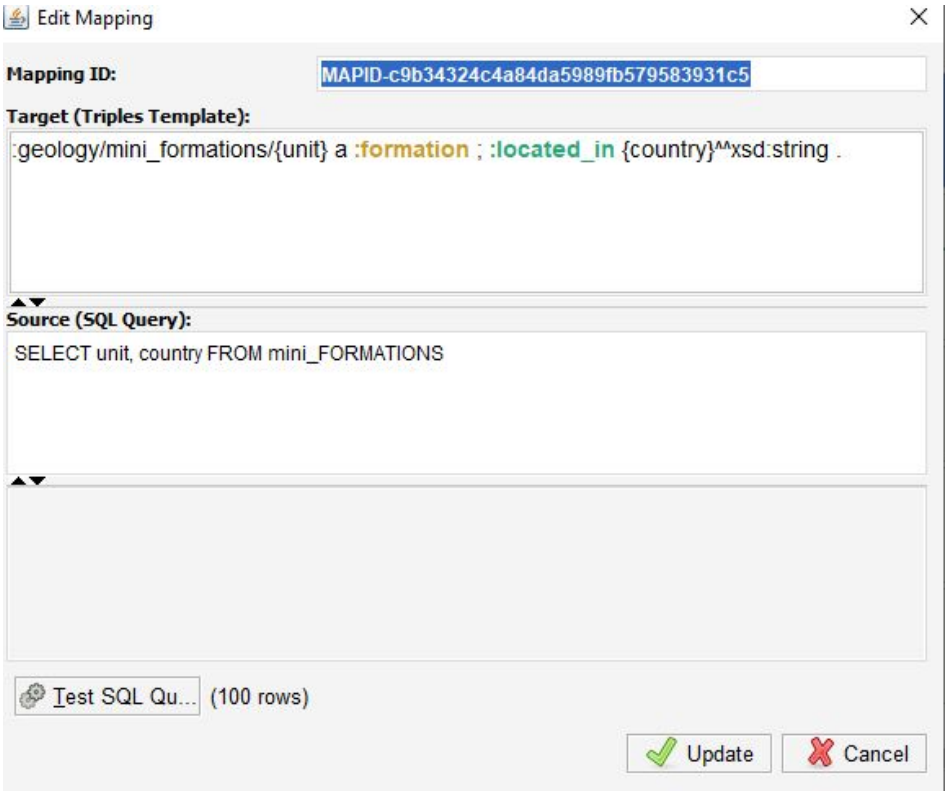
Now, we just need to enter our parameters and we are connected.



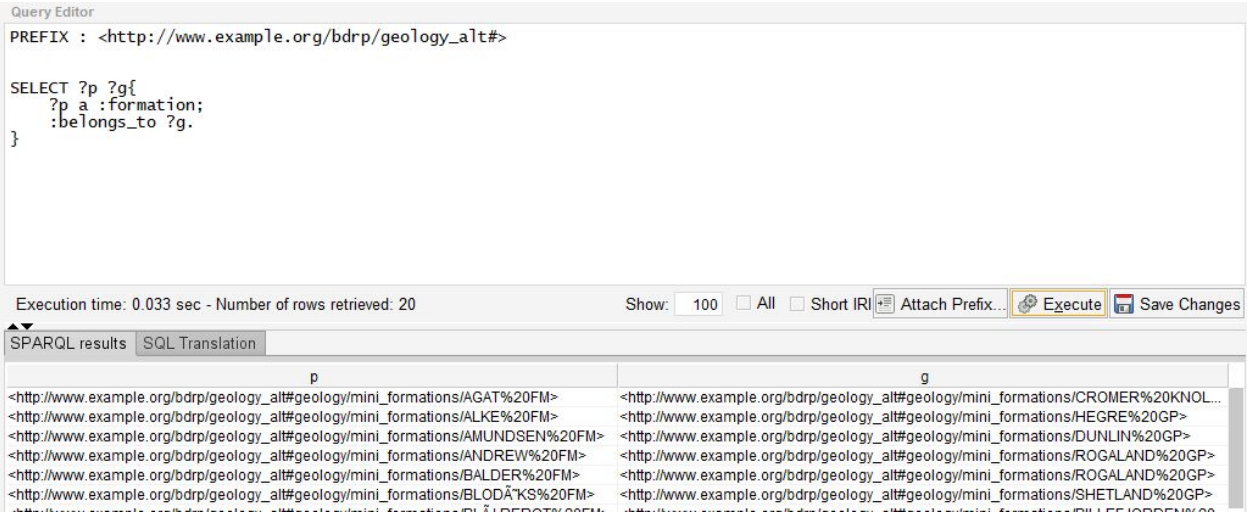
2. Now it is time to setup Protege to work with our RDBMS. For that we need to go to Preferences -> JDBC Drivers -> add and add the parameters of H2.



3. Creating mappings. For that we need to go Ontop Mappings Tab and press Create. In the popped up window we need to specify the target and the source. In the target we specify the assertion in SPARQL syntax, and in the source, we specify the SQL query.



4. Querying. Now that we have created the mappings. We can query the source using SPARQL. For that we need to go to Ontop SPARQL tab and where we can query using usual SPARQL syntax



Research

Description of some methods/ papers

Nowadays, with the data evolving at faster pace(5V's) more and more use cases document the requirement of a more dynamic machine interpretation of input and output data of applications. Currently, applications mostly exchange information on the basis of passing parameters or data,

formatted according to pre-defined strict syntaxes like API calls(POST/ GET). This approach can be understood as the exactness method[1]. Though the error management becomes easier to handle, this leaves no room for data interpretation, thus no scope for the benefits obtained out of reasoning. Ontologies enable a richer knowledge representation which improves machine interpretation of data. This is pivotal reason they have become to be widely used in information systems, like in ours, and applications, and propelling ontology construction being addressed as a hot research topic.

In fact classical development of domain ontology is typically entirely based on strong human participation. It does not adequately fit new applications requirements, because they need a more dynamic ontology and the possibility to manage a considerable quantity of concepts that the human can not achieve alone. The purpose of this research phase was not only to identify existing tools, but also to understand which parts of the generation satisfy our usage and classify each according to their utility to our use case.

There are several states of the art which pertain currently about ontology generation, also referred to as Ontology Learning, but papers focusing on automation for the whole process as defined above are rare and the ones which do the implementation details are hidden to avoid plagiarism. Shamsfard Mehrnoush and Barforoush Abdollahzadeh [2], put forward a complete framework that classifies software and techniques for building ontologies in six main categories (called dimensions). It is a

detailed and interesting classification, but alas it focuses only on the learning method leaving the reader a bit stranded on the background and application. In another approach[3] the authors, provide a comprehensive tutorial on learning ontology from text blocks, which is really seems enchanting and noteworthy to take mention, but the considered corpus source or other data sources do not fit our use case. Castano et al. [4] puts forward an easily comprehensive classification of techniques, making usage of different views of existing tools for ontology matching and coordination, but the limitation was the case of two existing ontologies. Since, the objective to develop a domain specific ontology, this too did fail to satisfy our use case.

Description of our methodology

We follow the essence of agile development methodology, in which we have biweekly sprint meetings and daily updates to understand the state of tasks we have picked up and for calling out blockers and help when required.

Tools

The task at hand was to be research oriented and look for process solutions which could be fascinating. Hence, we decided to proceed with two approaches: one developing the solution following the traditional knowledge base method and second, using an uncharted trail of virtual integration.

To accomplish our idea we make use of the following technologies:

- Python: as our scripting language used for data scraping, munging and generating A-Box.
- Protégé: for the developing and editing of the ontologies
- GraphDB by Ontotext:as the RDF database for our traditional database
- ONTOP as a ‘Virtual Knowledge Base’.

Datasets - process

The process owners eased our transition into the domain by providing some data sources which we were able to build on top off. On investigation, we found the majority of the Oil sources in North Sea, were located in and around Norway. Thus, websites like Norwegian Petroleum Directorate(NPD), Naturhistorisk besides the product owners glossary and Wikipedia acted as able allies to get your understanding of the domain to fair speed.

The data was scrapped through python scripts and the idea was to implement a series of conversions to homogenize the files to make the loading process easier. The first of the many conversions was to get rid of missing values for formations and groups, since the idea was to segregate between all groups and the formations belonging to each group. Having the cleaner version of files and linkage between entities, for example the

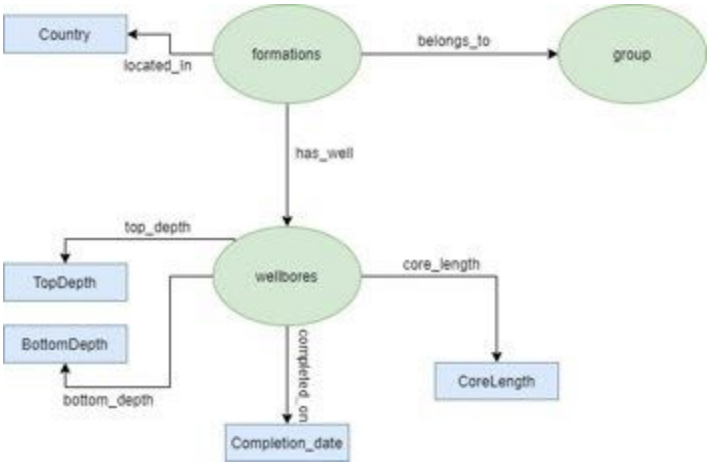
well bores and formations; formations to their country of presence and the lithological unit present along with their age, made the loading of A-Box pretty easy.

Modelling the knowledge graph

Modelling a knowledge starts with identifying the entities and the relationships between them. Therefore, the first step is to sketch the desired knowledge graph using graphical tools.

However, later, we have changed the structure of the graph to be more focused on formations and wellbores which are the central objects of the project. Once we had defined the entities, we started creating classes and literals. For example, *formation* is a class since it has many instances, many relationships with other entities; on the other hand, top depth of a wellbore is not a class, it is just a value, it is only connected to the wellbore. Having this process run through all of the entities we generated a TBox using Protege. Protege generates an .owl file of the ontology created which we later combined with the ABox.

ABox was created using a python library called *rdflib*. Using the library we were able to generate a .ttl file which contained all of the instances. The generation of the file is the process of creating triples subject-predicate-object from the data sources (.csv files, the data obtained through scraping).



Results

The resulting of proof of concept should be able to answer to some basic questions provided by the process owner, which are as follows:

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

Sample queries

As mentioned we created two approaches to potentially answer the question to see any effect on the solution to the queries.

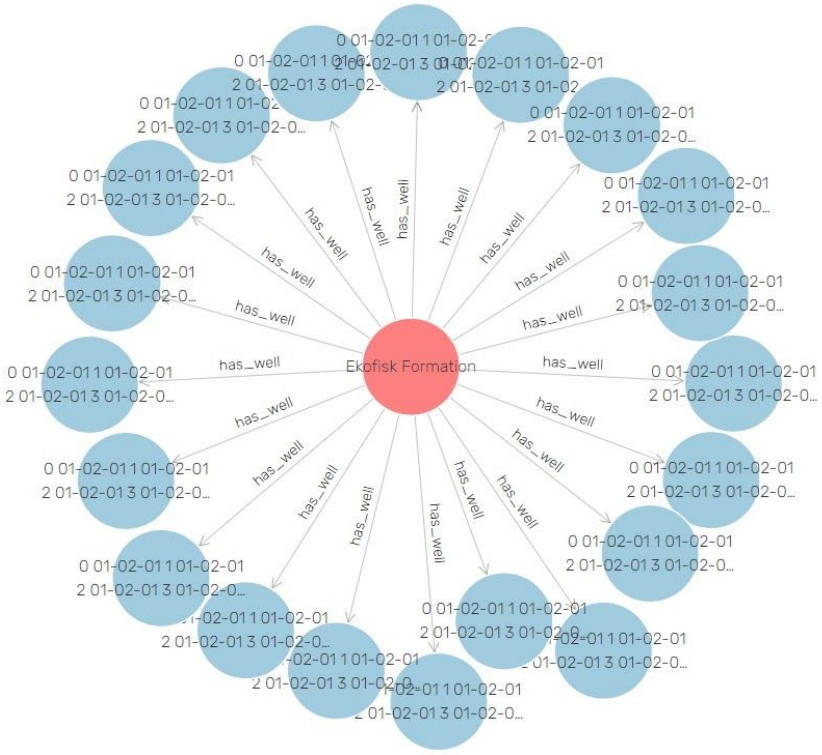
- Query1: Finding the countries where each formation is located?

	formation	country
1	ns1:agatformation	"Norway"
2	ns1:akkarmember	"Norway"
3	ns1:fruholmenformation	"Norway"
4	ns1:algemember	"Norway"
5	ns1:hekkingenformation	"Norway"
6	ns1:amundsenformation	"Norway"
7	ns1:andrewformation	"United Kingdom"
8	ns1:rogalandgroup	"Norway"
9	ns1:balderformation	"United Kingdom"
10	ns1:broomformation	"United Kingdom"
11	ns1:bryggeformation	"United Kingdom"
12	ns1:b√•tgroup	"United Kingdom"
13	ns1:b√∏rglumunit	"United Kingdom"
14	ns1:cookformation	"Norway"
15	ns1:drakeformation	"Norway"
16	ns1:draupneformation	"Norway"

- Query2: Finding to which group does a formation belong to?

	formation	group
1	ns1:agatformation	ns1:cromerknollgroup
2	ns1:akkarmember	ns1:fruholmenformation
3	ns1:fruholmenformation	ns1:realgrunnensubgroup
4	ns1:algemember	ns1:hekkingenformation
5	ns1:hekkingenformation	ns1:adventdalengroup
6	ns1:amundsenformation	ns1:dunlingroup
7	ns1:andrewformation	ns1:rogalandgroup
8	ns1:rogalandgroup	ns1:tempelfjordengroup
9	ns1:balderformation	ns1:rogalandgroup
10	ns1:broomformation	ns1:brentgroup
11	ns1:bryggeformation	ns1:hordalandgroup
12	ns1:b√•tgroup	ns1:rogalandgroup
13	ns1:b√∏rglumunit	ns1:rogalandgroup

Ekofisk Formation



References

http://ceur-ws.org/Vol-1695/paper4.pdf

http://bivan.free.fr/Docs/Automatic_Ontology_Generation_State_of_Art.pdf

Shamsfard Mehrnoush, Barforoush Abdollahzadeh. The State of the Art in Ontology Learning: A Framework for Comparison. The Knowledge Engineering Review, Volume 18, Issue 4. December 2003

Paul Buitelaar, Philipp Cimiano, Marko Grobelnik, Michael Sintek. Ontology Learning from Text Tutorial. ECML/PKDD 2005 Porto, Portugal; 3rd October - 7th October, 2005. In conjunction with the Workshop on Knowledge Discovery and Ontologies (KDO-2005)

S. Castano, A. Ferrar, S. Montanelli, G. N. Hess, S. Bruno. State of the Art on Ontology Coorination and Matching. Deliverable 4.4 Version 1.0 Final, March 2007. BOEMIE Project

Xiao G. et al. (2020) The Virtual Knowledge Graph System Ontop. In: Pan J.Z. et al. (eds) The Semantic Web – ISWC 2020. ISWC 2020. Lecture Notes in Computer Science, vol 12507. Springer, Cham. https://doi.org/10.1007/978-3-030-62466-8_17