INTEGRATING SEMWEB WITH NEO4J

Oskar Pawica, Michal Kilian, Marcin Kozak

INTRODUCTION

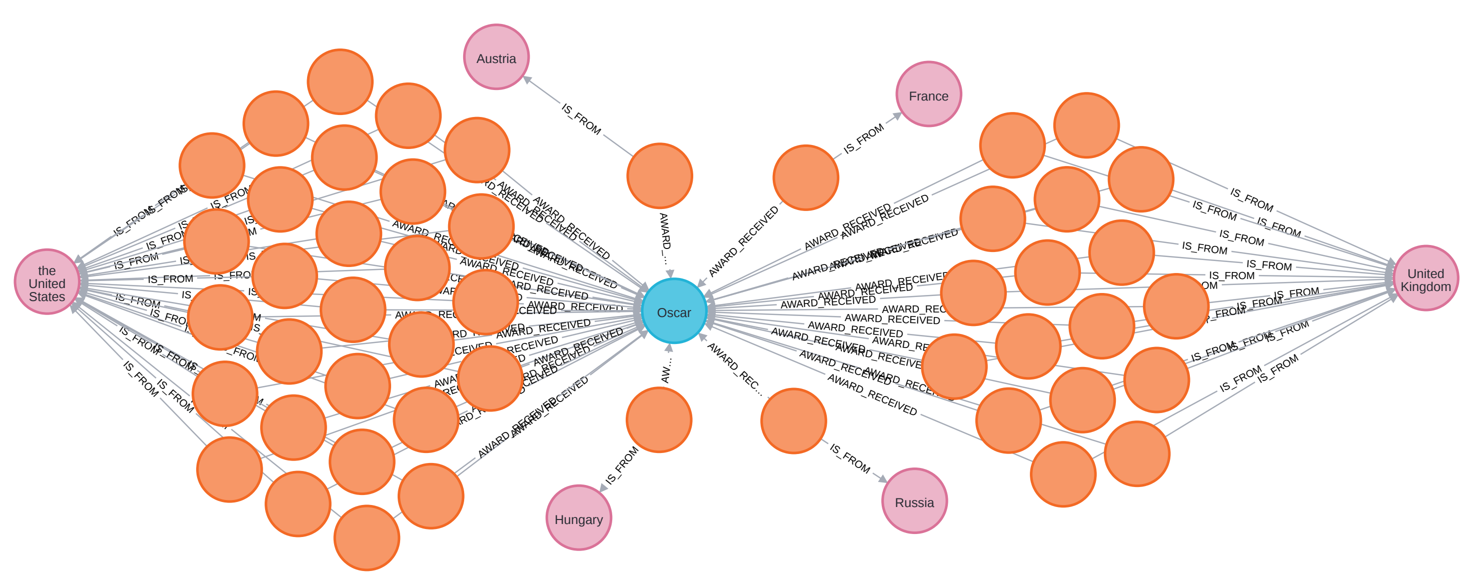
Internet is filled with enormous amout of data. SemanticWeb project tries to structure and unify them to computer-readable form. However, such form is not the most convenient for a human to browse. That's why we've decided to integrate information-rich Semantic Web structure with an eye-pleasing graph database visualization.

NEO4J

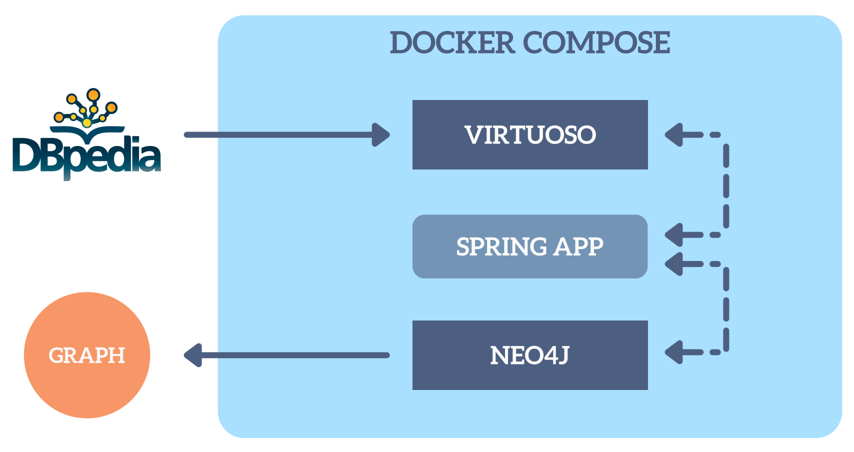
Graph databases are getting more and more attention each year and, thanks to their emphasis on relationships details, they do well with social networks, ontologies or recommendation engines.
They bring together the best characteristics of SQL and NoSQL databases and provide a great scalability potential. Moreover they prove to be convenient to work with, as it is very easy for the developer to present any life situations as a graph model. At the moment **Neo4j is the most popular graph database** on the market and it comes with the most vibrant community of all graph databases.

STRUCTURE

The main node of the graph is an Academy Award Oscar node (blue one). It's connected with Actors' nodes (orange) via the AWARD_RECEIVED arc directed towards Oscar's node. Actor node holds such information as: birth date and actor's name. Every actor node is connected with a Country node (pink) via the IS_FROM arc.
The presented example is rather simple, yet it already shows the idea and its advantages, allowing users to work on SemWeb with ease using graphs.
Adding more nodes such as movies could improve the interconnectivity even more.



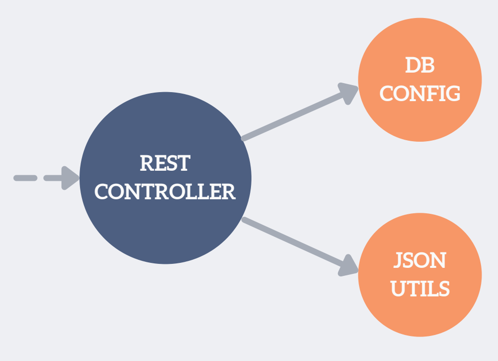
SOLUTION ARCHITECTURE



The diagram on the left presents the architecture of our solution, showing the flow of the integration and its components.
Such an approach allows easy configuration and usage of the bundle.

MIDDLEWARE

The spring application included in the bundle works as a lightweight middleware between Virtuoso and Neo4j.
It is simple and consists of only three classes, which pass Neo4j's requests to Virtuoso and format the responses as JSON.



SPAQRL

SPARQL Protocol And RDF Query Language is a query language able to operate on RDF (Resourse Description Framework) triplets. A triplet is a combination of: subject (source node), predicate (edge name) and object (target node). A simple triple in turtle format may look as follows:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/stuff/1.0/>.
ex:student_102 ex:is_taught_by ex:lecturer_02
```

SPARQL lets us use SQL like syntax to select and access RDF data. Simple SPARQL query may look like this:

```
PREFIX vcard: #data source
<http://www.w3.org/2006/vcard/ns#>
SELECT ?person ?givenName
WHERE
{
 ?person vcard:family-name “Smith”.
  # family-name == "Smith"
 ?person vcard:given-name ?
givenName.
  # given-name != null
}
```

DBPEDIA

DBpedia is a project trying to structurise data available on Wikipedia. It allows to query relationships and properties created in the Wikipedia project.
The initial release was in 2007 and the latest release from 2017 contains 6 milion Wikipedia entities.
To represent information DBpedia uses RDF and to access data SPAQRL is used.

INTEGRATION

The idea behind the project was to integrate Virtuoso SPARQL Query Results into a Neo4j database and get a property graph as an outcome.
Different drivers could have been used to connect Neo4j with Virtuoso. An improper usage of those resulted in a datatype mismatch caused by Virtuoso's extended types for handling RDF (Resource Description Framework) data.
JDBCtoODBC Bridge driver was one of the solutions for this problem, yet we chose a different approach.

The easiest way to handle the issue was to create a simple middleware in Java Spring framework. Neo4j's requests were targetting this lightweight application, which was passing them on to Virtuoso using Type 4 Virtuoso JDBC Driver.
We managed to improve the simplicity of our solution by configuring all three services in a single Docker Compose file. Containerized entities were able to start and communicate with each other, basing on centralized settings and by using one simple command.

DIFFICULTIES

- Not perfectly structured entries in DBpedia
- Not a lot of information about such integration on the Internet
- Mismatched database drivers

Repository: https://github.com/KRaKEn-Research-Group/semweb_neo4j_integration