DATABASE USAGE IN SEMANTIC ONTOLOGIES
DATUBĀZES IZMANTOŠANA SEMANTISKAJĀS ONTOLOĢIJĀS

Author: **Daniēls ZEPS**, e-mail: dz22036@edu.rta.lv
Scientific supervisors: **Imants ZAREMBO**, Dr.sc.ing**.**, e-mail: imants.zarembo@rta.lv
Sergejs KODORS, Dr.sc.ing**.**, e-mail: sergejs.kodors@rta.lv
Rezekne Academy of Technologies
Atbrīvošanas aleja 115, Rēzekne

Abstract. *Semantic ontology languages are a way for experts to write down their knowledge in a commonly accepted way, it allows information to be understood by humans and machines. However, ontology tools do not provide the ability to use this data in a database for geodata-based analysis easily. The project is focused on ontology web language (OWL) usage for web content generation for business-to-business communication using geodata analysis. This specific paper is focused on selecting a database that could support the ontology tools. In the research scope, different databases were checked (relationship, document, and graph databases). In conclusion, graph databases were the most similar to the structure of the ontology, so it was chosen to use Neo4j as the database.*

Keywords: *databases, graph databases, ontology, semantic, web content generation.*

Introduction

Semantic ontology languages (SOL) are a good way to represent data in a visual format, that is both understandable to humans and computers. It allows you to see the connections between different data points. It allows communication between human and computer languages with almost no restrictions (Understanding Semantic Web and Ontologies: Theory and Applications, 2010). While good for data depiction, SOL is not good for data storage and is not easily accessible, as a result, an additional database needs to be used to store data.

The project is focused on ontology web language (OWL) usage for web content generation for business-to-business communication using geodata analysis. The project will focus on three parts – server code writing, database selection, and OWL conversion to a database. To enable web content viewing and server hosting it was chosen to use Django, due to developers' team preference.

This paper is focused on determining which database will support SOL the best. The purpose of comparing databases is due to possible complications that could be found in the later stages of the project due to the lack of databases flexibility or limitations of data structure and its storage as a result it might be very hard to take data from Web Protégé and convert it.

Research goal: find the best-fitting database for storing information from OWL.

Tasks:

1) Study types of databases;
2) Select databases that are compatible with Django;
3) Test out selected databases on a sample ontology;
4) Evaluate the results.

Materials and methods

Primary criteria for a good database tool

Since the main project uses Web Protégé and has to use Django to host web-generated content the requirements will be different and that needs to be considered before picking a database for the project. For purposes of this paper, it was decided that testing out different types of databases would give a better insight into which database to use. It was decided to test out relational databases (*PostgreSQL – Django built-in database*), document-oriented databases (*MongoDB*), and graph databases (*GraphDB*, *Neo4j*).

https://doi.org/10.17770/het2023.27.7378

To select the best-fitting database the following criteria were selected:
1) Supports Python Django.
2) Is open source or free to use;
3) Supports unstructured data;
4) Supports data relationships;
5) Has simple Python request syntax;
6) Can process geo data.

What are graph databases?

Graph databases (*see Fig.1*) are a way of storing information based on *Graph theory* in the field of mathematics. They are mostly used in social media, for example, Facebook and Instagram (Wikipedia, 2023). Data in a graph database is stored as relationships, which means that it can be accessed faster, due to data connections. Graph databases perform fewer data requests because data is directly connected, so there is no repeated indexing. Graph databases are better used if there are many joins between data points (Neo4j, 2023).

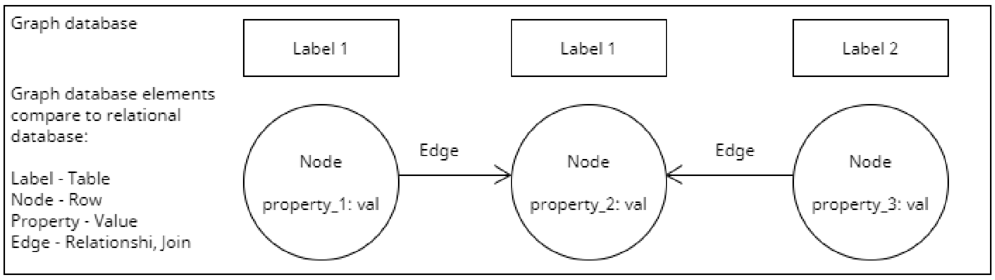


Fig.1. **Visual depiction of a graph database**

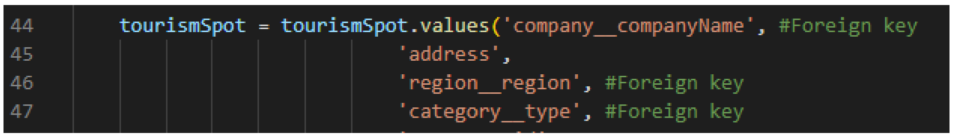
Results and discussion

After having researched and tested all of the databases the following data was gathered (*see Table 1*). The databases were tested using a sample OWL data that was converted into a database and by using the database tools.

Table.1. **Comparison of databases based on previously mentioned criteria**

	PostgreSQL	MongoDB	GraphDB	Neo4j
Supports Python Django	X	X		X
It is open source or free to use	X	X	X	X
Supports unstructured data		X		X
Supports data relationships	X		X	X
Has a simple Python request syntax	X			X
Can process geo data		X		
Total	4	4	2	5

PostgreSQL (PostgreSQL, 2023) (*Fig.2*) was the Django built-in database which was good in testing with sample ontology. It is based on SQL, so it has a lot of extensions. This is a relational database so information is more secure, but this also means that it does not support unstructured data like others. All the fields would have to be defined for all entries which would make it take up a lot of unused space and make the scheme very complex.

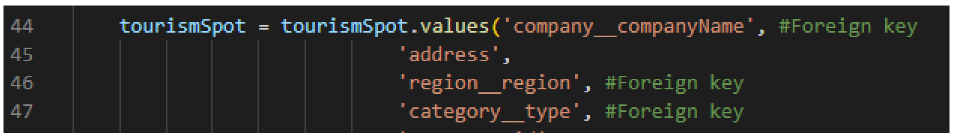


```
tourismSpot = tourismSpot.values('company__companyName', #Foreign key
                                 'address',
                                 'region__region', #Foreign key
                                 'category__type', #Foreign key
```

Fig.2. **PostgreSQL code showing foreign key query requests**

MongoDB (MongoDB, 2023) (*Fig.3*) has the advantages of having built-in GeoJSON and no schema, which means that direct data transfer is simpler, but it still was the least useful due to it not supporting relationships and requiring the use of additional code to enable this functionality, which would make code generation very difficult.

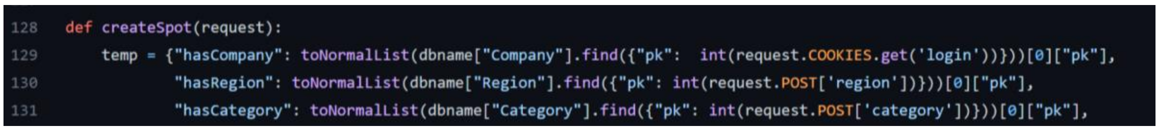
PostgreSQL (PostgreSQL, 2023) (*Fig.2*) was the Django built-in database which was good in testing with sample ontology. It is based on SQL, so it has a lot of extensions. This is a relational database so information is more secure, but this also means that it does not support unstructured data like others. All the fields would have to be defined for all entries which would make it take up a lot of unused space and make the scheme very complex.



```
tourismSpot = tourismSpot.values('company__companyName', #Foreign key
                                 'address',
                                 'region__region', #Foreign key
                                 'category__type', #Foreign key
```

Fig.2. **PostgreSQL code showing foreign key query requests**

MongoDB (MongoDB, 2023) (*Fig.3*) has the advantages of having built-in GeoJSON and no schema, which means that direct data transfer is simpler, but it still was the least useful due to it not supporting relationships and requiring the use of additional code to enable this functionality, which would make code generation very difficult.

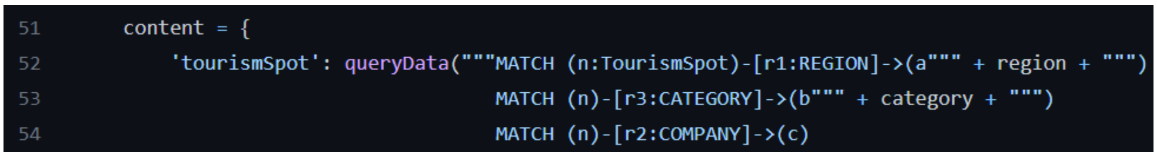


```
def createSpot(request):
    temp = {"hasCompany": toNormalList(dbname["Company"].find({"pk":  int(request.COOKIES.get('login'))}))[0]["pk"],
            "hasRegion": toNormalList(dbname["Region"].find({"pk": int(request.POST['region'])}))[0]["pk"],
            "hasCategory": toNormalList(dbname["Category"].find({"pk": int(request.POST['category'])}))[0]["pk"],
```

Fig.3. **MongoDB code showing foreign key query requests**

GraphDB (ontotext, 2023) stores data in RDF format which is one of the formats that OWL allows data to be stored in, which makes it similar to WebProtégé, but it was not usable in this project due to not having a Python library (*one could not be found during research*).

Neo4j (Neo4j, 2023) (*Fig.4*) was deemed the best option for this project because it supports unstructured data and has easier syntax for user interface and web generation, making Python files and Django backend programming simpler. While not storing data in a similar format it displays data in a similar format to OWL, because it is a graph database and represent data connections that are present in OWL in a similar way. The only meaningful downside to Neo4j is data being less secure, due to it not having a schema.



```
    content = {
        'tourismSpot': queryData("""MATCH (n:TourismSpot)-[r1:REGION]->(a""" + region + """)
                                    MATCH (n)-[r3:CATEGORY]->(b""" + category + """)
                                    MATCH (n)-[r2:COMPANY]->(c)
```

Fig.4. **Neo4j code showing foreign key query requests**

Conclusions

For purposes of the main project graph databases would be more useful because they focus on data relationships and data relationships which makes them much more similar to OWL, plus the added flexibility of unstructured data is going to make OWL conversion less strained by database requirement. Additionally, Neo4j built-in tools provide the ability to verify that the SOL data has been translated correctly and make testing faster.

Summary

SOL is a powerful tool for data representation. It allows users to represent data and see relationships between data points. The goal of this paper is to find a database that can support SOL. A database that could be used should support:

1) Supports Python Django;
2) Is open source or free to use;
3) Supports unstructured data;
4) Supports data relationships;
5) Has simple Python request syntax;
6) Can process geo data.

It was decided to test out relational databases (*PostgreSQL – Django built-in database*), document-oriented databases (*MongoDB*), and graph databases (*Neo4j*, *Graph DB*). Out of these graph databases, Neo4j is the best-fitted one due to its focus on relationships and built-in tools for data analysis.

Acknowledgement

This research is funded by the Latvian Council of Science, project “Digital Twin to Promote the Development of Tourism Competitiveness and Complementarity: the Use Case in Latgale region”, project No. lzp-2022/1-0350.

Bibliography

1. MongoDB. (2023, February 17). *MongoDB*. Retrieved from MongoDB: https://www.mongodb.com/
2. Neo4j. (2023, February 6). *Neo4j Developer*. Retrieved from Neo4j: https://neo4j.com/developer/graph-database/ontotext. (2023, February 17). *GraphDB*. Retrieved from GraphDB: https://graphdb.ontotext.com/
3. PostgreSQL. (2023, February 9). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Retrieved from PostgreSQL: https://www.postgresql.org/
4. Understanding Semantic Web and Ontologies: Theory and Applications. (2010). *Journal of Computing*, Volume 2, Issue 6. Retrieved from https://arxiv.org/ftp/arxiv/papers/1006/1006.4567.pdf
5. Wikipedia. (2023, February 6). *Graph database*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Graph_database.