Research and Development: Geographical Data

Done by Tatchemo Guiafaing Ronald VR512344

Under the Supervision of Professor Alberto Belussi

Department of Computer Science, University of Verona

Monday, December 30, 2024

Contents

1 Introduction

Geographic Information Systems (GIS) and spatial databases are crucial in modern data analysis and visualization. This internship, conducted under the supervision of Professor Alberto Belussi at the Department of Computer Science, University of Verona, focused on exploring the integration capabilities of MongoDB, a leading NoSQL database, with various GIS tools for visualizing and analyzing geographical data.

The project originated from a desire to expand upon knowledge gained in the Advanced Database course, particularly in handling and visualizing spatial data. The main objective was to investigate various methodologies for visualizing GeoJSON data using MongoDB and exploring its integration with established GIS platforms.

2 Background and Objectives

2.1 Technical Background

MongoDB, a document-oriented database, offers robust support for geospatial data through its GeoJSON implementation. This capability, coupled with its flexible schema design, positions MongoDB as an ideal candidate for storing and querying spatial data. This internship aimed to leverage these features while exploring their integration with specialized GIS tools.

2.2 Project Objectives

The key objectives of this internship were:

- Evaluate MongoDB's capabilities in handling and visualizing geospatial data.
- Explore integration methods between MongoDB and QGIS.
- Develop automated workflows for processing and loading spatial data.
- Investigate the potential of GeoServer as a middleware solution.

3 Data Sources and Methodology

3.1 Dataset Selection

The primary dataset utilized was the `us-colleges-and-universities.geojson` file, obtained from OpenData. This comprehensive dataset contains detailed information about educational institutions across the United States, including precise geographical coordinates and institutional attributes.

3.2 Tools and Technologies

The project utilized several key technologies:

- **MongoDB Compass** for database management and initial visualization.
- **QGIS 3.x** for advanced geographical visualization.
- **Python** for data processing and automation.
- **GeoServer** for web-based geospatial data serving.

4 Implementation and Results

4.1 QGIS Integration

Initial visualization efforts focused on QGIS implementation using three distinct shape files:

- Washington DC watersheds for environmental analysis.
- Chicago metropolitan area for urban studies.
- Maryland county boundaries for administrative delineation.

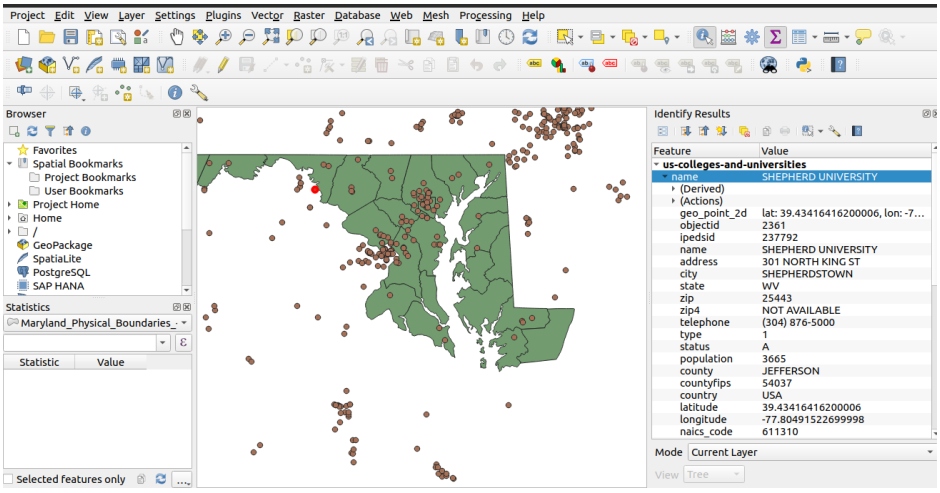

Figure 1: Visualization of MarylandShape file on USColleges in QGIS

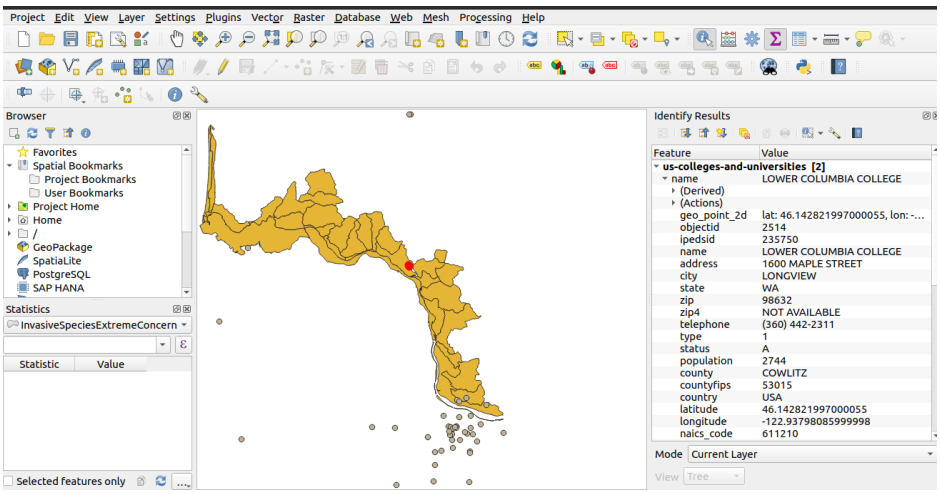

Figure 2: Visualization of WashintonDC Shape file on USColleges in QGIS

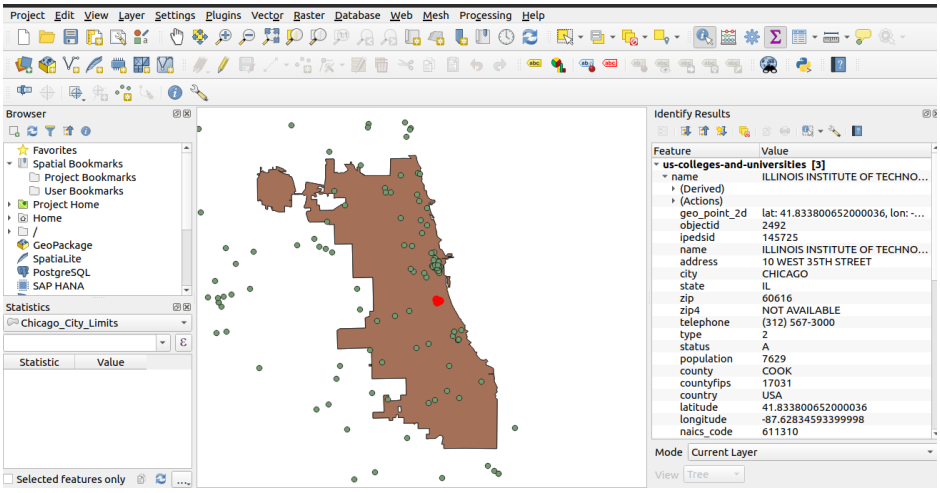

Figure 3: Visualization of Chicago Shape file on USColleges in QGIS

4.2 MongoDB Implementation

4.2.1 Data Processing

The implementation involved sophisticated data processing workflows:

```
import geopandas as gpd
import pymongo
from shapely.geometry import shape

def mongodb_to_qgis(mongodb_uri, database_name,
collection_name, output_shapefile):

        try:
            if 'geometry' in doc:
                geom = shape(doc['geometry'])
                props = doc.get('properties', {})
            elif 'type' in doc and 'coordinates' in doc:
                geom = shape(doc)
                props = {}
```

```
                features.append(geom)
                properties.append(props)
            except Exception as doc_error:
                print(f"Error processing document: {doc_error}")

        # Create GeoDataFrame
        gdf = gpd.GeoDataFrame(properties, geometry=features)

        # Save to shapefile
        gdf.to_file(output_shapefile)

        print(f"Successfully exported {len(gdf)} features to {output_shapefile}")

        return gdf

    except Exception as e:
        print(f"Error connecting to MongoDB or processing data: {e}")
        return None

# Example usage
if __name__ == "__main__":
    # Replace these with your actual MongoDB details
    MONGODB_URI = 'mongodb://localhost:27017'
    DATABASE_NAME = 'USA_COLLEGE_UNIVERSITIES'
    COLLECTION_NAME = 'Universities_Collections'
    OUTPUT_SHAPEFILE = 'mongodb_export.shp'

    # Convert MongoDB to Shapefile
    mongodb_to_qgis(MONGODB_URI, DATABASE_NAME, COLLECTION_NAME, OUTPUT_SHAPEFILE)
```

The Images below represent the practical application of the conversion script whose main goal is to convert each shape file into MongoDB Compass directly.

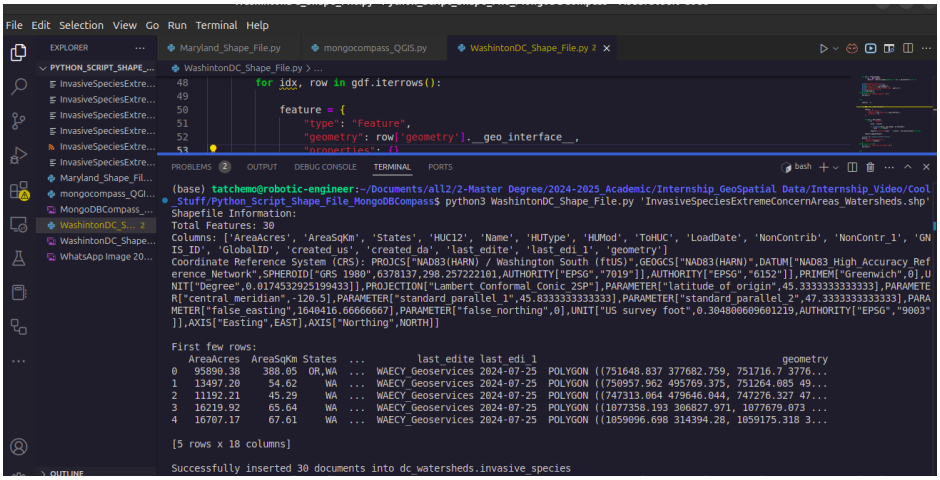

Figure 4: Visualization of Chicago Shape file on USColleges in QGIS

This Python script imports a shapefile containing geographic data and uploads this data to a MongoDB database. The process starts by loading the shapefile using GeoPandas, which reads the file into a GeoDataFrame. It checks for specific date and float columns, converting their values to string and handling any missing data appropriately. It also creates a GeoJSON structure from the GeoDataFrame, converting the geometric shapes and properties of each feature. Next, it connects to MongoDB, determining the target database and collection based on the shapefile name. Any existing documents in the selected collection are deleted to ensure a fresh start. It then inserts the new GeoJSON features into the collection. Finally, the script outputs a summary of the operation, including the number of documents successfully inserted. The run example shows the successful operation of inserting **30 documents** from the shapefile 'Areas$_{W}atersheds.shp$'

4.2.2 Database Structure

Two primary databases were created using modified version of the script previously shown:

- `maryland_counties`: Contains detailed boundary information.

- `dc_watersheds`: Houses environmental and invasive species data.

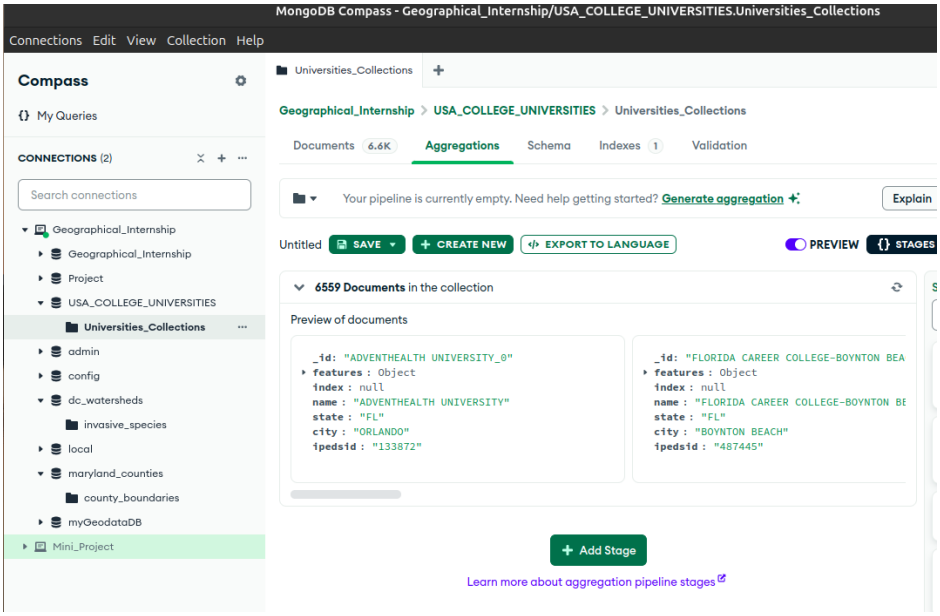

Figure 5: Visualization on MongoDBCompass of the Converted shapefile loaded

5 Integration Approaches

5.1 GeoServer Integration

The first approach utilized GeoServer as a middleware solution:

- Attempted Web Feature Service (WFS) implementation.
- Explored compliance with OGC standards.
- Investigated performance optimization strategies.

5.2 Direct Integration

The second approach focused on direct database integration:

- Developed Python scripts for automated data loading.
- Implemented parameterized processing workflows.
- Created standardized data transformation protocols.

6 Challenges and Solutions

6.1 Technical Challenges

Several challenges were encountered and addressed:

- Variability in source data schema.
- Transformations between coordinate systems.
- Optimization of performance for large datasets.

7 Future Work

Potential areas for future development include:

- Enhanced automation of data processing workflows.
- Implementation of real-time data updates.
- Development of custom visualization tools.
- Integration with additional GIS platforms.

8 Conclusion

This internship provided valuable insights into the integration of MongoDB with GIS technologies. Although some approaches proved more successful than others, the experience highlighted both the potential and limitations of existing geospatial data handling technologies. The knowledge gained will be helpful for future developments in this field.

The project demonstrated the feasibility of using MongoDB for spatial data management while identifying areas for further development to enhance integration with existing GIS tools. The experience gained during this internship at the University of Verona has laid a strong foundation for future work in geospatial data management and visualization.