**An Open Source Alternative of**

**Geospatially Enabled Semantic Web Framework**

# Objectives

* Code clean-ups. This process will restructure the general code framework and optimise the hard-coded part for a better performance.
* Further demonstrate implementation challenges during the entire development stage.
* Build a benchmark framework which is compatible with the Parliament and Strabon triple store. In specific, the benchmark framework should not only consider whether the test has yielded results or not, but also take the result accuracy into consideration.
* Suggestion of future works. This practice will outline the future development possibilities of this implementation, such as spatial database integration and the SPARQL server and endpoint implementation.

# GeoSPARQL Implementation

In order to provide a consistent representation and manipulation of geospatial semantic data, we implement the GeoSPARQL standard upon Apache Jena as an alternative open source geospatially enabled semantic web framework.

## Components of GeoSPARQL

The GeoSPARQL standard follows a modular design approach which allows implementations to specify the extent of conformance with it. These layered components give implementations flexibility to achieve the GeoSPARQL functionalities. For example, for application only wants to conduct qualitative spatial reasoning is suggested to support the core and topological vocabulary requirement classes. All these components are represented by a requirement class from GeoSPARQL: core, topology vocabulary extension, geometry extension, geometry topology extension, RDFS entailment extension, and query rewrite extension.

**Core**. This requirement class simply indicates the implementation should follow the SPARQL syntax and return a correct result with a proper format. And the RDFS class geo:SpatialObject and geo:Feature should be both used in the GeoSPARQL queries and datasets.

**Topology vocabulary extension**. The compliance of this requirement class need to implement all the topological functions from the relation family as SPARQL property functions. In addition to the relation family, GeoSPARQL specifies three relation families: The Simple Features relation family, the Egenhofer relation family, and the RCC8 relation family.

**Geometry extension**. This requirement class is quite similar to the core class, which requires a set of geometry vocabularies should be used in the GeoSPARQL queries and datasets. These vocabularies not only facilitate the serialisation and versioning of the geometry, but also provides several properties for describing geometry metadata. Furthermore, a set of non-topological functions is suggested to be implemented as the SPARQL filter function.

**RDFS entailment extension**. This requirement class suggests the implementation to inherit the RDFS entailment regime defined by W3C. In simple words, this class requires the implementation should be able to query the implicit relations defined in the dataset based on the RDF and RDFS semantics. Moreover, for the serialisation language specified by the implementation, i.e., the WKT, GML, or both, implementation should also be able to access the implicit relations based on the class hierarchy of the serialisation language.

**Query rewrite extension**. This requirement class provides a flexible query pattern for the implementation. For example, the comparison between geometry and geometry can be achieved via a comparison between feature to feature by implementation the query rewrite extension.

## Implementation Challenges

* The coordinate reference system support
* The geometry property support
* Geometry wrapper around JTS
* DE-9IM intersection matrix difference between JTS and GeoSPARQL

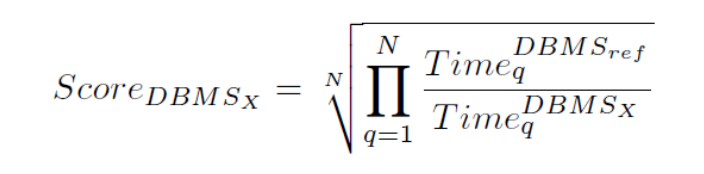
# Benchmark Framework Design

Recently, due to the nonexistence of the spatial semantic web benchmark, many researchers and institutions have extended their new spatial SPARQL benchmarks which are based on the existing semantic web benchmark or the spatial relational database benchmark. This section will mainly review these state of the are benchmarks and present our customised GeoSPARQL benchmark which is based on two most popular benchmark: The benchmark Geographica and the benchmark designed by GeoKnow project. The decision making during this benchmark development will also be explained as well.

## Review of the spatial SPARQL benchmarks

For the purpose of the consistence of name, we will name the spatial (or geospatial) semantic web benchmark as spatial SPARQL benchmark. We will mainly focus on the two prominent spatial SPARQL benchmarks which are intended to evaluate the performance of the spatial triple stores: The benchmark Geographica and the benchmark designed by GeoKnow project. Specially, we will also review another spatial relational database benchmark Jackpine since these two spatial SPARQL benchmarks are both based on this benchmark. At the end of this section, a brief enumeration of other related benchmarks will be constructed.

**Jackpine**. This benchmark is initially designed for assessing the performance of relational spatial database via the JDBC driver implementation. it has a “micro benchmark” for the primitive spatial operations and a “macro benchmark” for evaluating the queries under some real-world scenarios. In addition to the datasets, Jackpine uses the shapefiles of Texas which is based on the TIGER (Topologically Integrated Geographic Encoding and Referencing) system. Finally, the performance of the test database is assessed through the Equation 1. A higher database score indicates a better overall performance.



Equation 1: overall database score calculation.

**GeoKnow spatial benchmark**. Although GeoKnow project have not specified a name for this benchmark, the name “GeoKnow spatial benchmark” is assigned for the convenience of content delivery. Generally, GeoKnow spatial benchmark has extended the Jackpine benchmark to make it compatible with the spatial SPARQL queries. The purpose of this benchmark is to evaluate the performance of the spatial triple stores through the location queries, range queries, spatial joins, nearest neighbour queries, and spatial aggregates. It uses the ESRI shapefile of Great Britain from OpenStreetMap. Ultimately, the measurement covers the data loading amount, bulk loading time, spatial indexing cost, and the evaluation time.

**Geographica**. Similar to Jackpine, this benchmark consists of two sub-benchmarks as well, a “micro benchmark” for testing the basic spatial operations and a “macro benchmark” for assessing the performance via three typical scenarios: reverse geocoding, map search and browsing, and the rapid mapping for wild fire monitoring. It utilises several sources of data, real-world data from LinkedGeoData and OpenStreetMap, a customised synthetic data is also generated for a controlled test environment. Finally, the elapsed time is calculated for assessing the performance.

## Design of the GeoSPARQL benchmark

As being suggested by the benchmark Geographica and GeoKnow project, the development of the GeoSPARQL standard compliance test will greatly facilitate the implementation of this standard and enhance the interoperability of spatial semantic data. Therefore, the emphasise of designing a generic GeoSPARQL compliance test need to be hereby made. Additionally, the result precision and recall will also be considered in order to evaluate the accuracy of the query result since these two parameters have not been considered in the previously introduced benchmarks.

Generally, this GeoSPARQL benchmark consists of two subset benchmarks: the micro benchmark for testing the compliance of the GeoSPARQL; the macro benchmark for testing standard the primitive spatial functions and their corresponding performance.

### Datasets

### Micro benchmark

### Macro benchmark

### Precision and recall

## Experimental Setup

## Evaluation Result

# Possible Enhancements

## Spatial database integration

The result of the performance test is promising because the performance of the initial implementation is overwhelmingly better than the performance of Parliament. Therefore, it would be reasonable to assume the integration of the spatial database will improve the performance by using the spatial index.

The advantages of spatial index extension can be illustrated in three aspects: with the utilisation of spatial index, this project can release the heavy dependency on the JTS since some text search engine library like Apache Lucene has already enabled the geospatial information search. Another benefit is that spatial index extension will improve the query speed. This is because with the spatial index pre-installed in the database, the topological calculation of two spatial objects can be firstly conducted by comparing the bound box of these two objects rather than compare the actual geometries. This improvement will more significant when comparing two polygon spatial objects. Finally, the spatial index can optionally save some basic figures of a spatial object such as distance, envelop, or the SRID, etc. Therefore, queries that involve such information retrieval will be transferred to queries that directly ask these values.

Future work of this topic is to conduct research study on the spatial index extension. This process will involve but not limit to:

* Investigating different search engines such as Apache Lucene, Apache Solr, etc.
* Developing a framework that link the spatial search engine to the SPARQL query engine.
* Enable the creation of spatial index when loading new RDF datasets.
* The development of a complete

Furthermore, as mentioned in the evaluation of the benchmark test, a deeper survey need to be conducted for the purpose of investigating the correlation between geometry instance distribution in the dataset and the test performance.