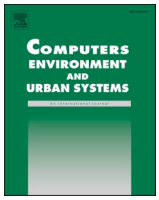
Computers, Environment and Urban Systems 98 (2022) 101884

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/ceus



# Performance benchmark on semantic web repositories for spatially explicit knowledge graph applications

Wenwen Li [a,*], Sizhe Wang [a,b], Sheng Wu [c], Zhining Gu [a], Yuanyuan Tian [a]

[a] School of Geographical Sciences and Urban Planning, Arizona State University, Tempe, AZ 85287-5302, United States of America
[b] School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85287-8809, United States of America
[c] School of Computer and Information Science, Southwest University, Chongqing 400715, China

ARTICLE INFO

*Keywords:*
Triple store
Property graph databases
Ontology
Knowledge graph
Relational database
Ontology-based Data Access (OBDA)

ABSTRACT

Knowledge graph has become a cutting-edge technology for linking and integrating heterogeneous, cross-domain datasets to address critical scientific questions. As big data has become prevalent in today's scientific analysis, semantic data repositories that can store and manage large knowledge graph data have become critical in successfully deploying spatially explicit knowledge graph applications. This paper provides a comprehensive evaluation of the popular semantic data repositories and their computational performance in managing and providing semantic support for spatial queries. There are three types of semantic data repositories: (1) triple store solutions (RDF4j, Fuseki, GraphDB, Virtuoso), (2) property graph databases (Neo4j), and (3) an Ontology-Based Data Access (OBDA) approach (Ontop). Experiments were conducted to compare each repository's efficiency (e.g., query response time) in handling geometric, topological, and spatial-semantic related queries. The results show that Virtuoso achieves the overall best performance in both non-spatial and spatial-semantic queries. The OBDA solution, Ontop, has the second-best query performance in spatial and complex queries and the best storage efficiency, requiring the least data-to-RDF conversion efforts. Other triple store solutions suffer from various issues that cause performance bottlenecks in handling spatial queries, such as inefficient memory management and lack of proper query optimization.

## 1. Introduction

Knowledge graph, a semantic web technology that links massive and cross-domain data, has become a new linked-data way of data organization and a key technology for information retrieval, hidden linkage identification, and knowledge-driven decision support (Li, Song, & Tian, 2019). In the geospatial domain, in which available data are highly heterogeneous in their formats and storage methods, knowledge graphs have been exploited to provide a uniform solution for managing and organizing such data. As geospatial big data have become increasingly prevalent in applications, such as smart cities (Li, Batty, & Goodchild, 2020), earth observations (Usery et al., 2022), and social sensing (Liu et al., 2015), there is an urgent need to investigate different semantic data repositories in their scalability and capabilities to manage large domain knowledge graphs, which often requires the integration and interoperability of diverse datasets in order to address critical environmental and social problems (Janowicz et al., 2022).

In a knowledge graph, the structured or unstructured data will be serialized into the format of triples <subject, predicate, object>, which contain atomic units to express logical relationships between the entities. Leveraging this data structure, a knowledge graph application can more readily perform logic reasoning to infer new knowledge and hidden semantic relationships (Li, Raskin, & Goodchild, 2012). A typical storage solution for such data structures is called a triple store, a type of graph database to perform graph data management and provide semantic query support. Since the notion of the Semantic Web was coined by Berners-Lee, Hendler, and Lassila (2001), there have been many efforts to develop scalable and efficient triple stores leveraging RDFs (Resource Description Frameworks). However, when adopting knowledge graph technology for geospatial applications, a key question has arisen: Are available semantic data repositories sufficient for managing large amounts of spatial data and efficiently handling spatial semantic queries? A semantic data store without proper query optimization may negatively affect the performance of domain applications that rely on knowledge graph technology. Geospatial data, as a special type of datasets, need to be stored and queried efficiently on top of regular

* Corresponding author.
*E-mail address:* wenwen@asu.edu (W. Li).

https://doi.org/10.1016/j.compenvurbsys.2022.101884
Received 18 March 2022; Received in revised form 3 June 2022; Accepted 2 September 2022
Available online 15 September 2022

knowledge graphs to support real-world applications, such as real-time decision making on traffic routing, disaster response, and object tracking that involves critical space and time components (Li, Zhou, & Wu, 2016; Li, 2020; Li, 2022).

Although there are many available open-source and commercial triple store solutions (e.g., RDF4j, AllegroGraph), their support for spatially embedded knowledge graphs is largely understudied. Recent research (Raza, 2019) provides a comparison of triple stores in their support of spatial queries and spatial reasoning, but they evaluate only the availability of certain spatial functions. There is no assessment on the computational efficiency of such solutions to respond to spatial query requests that support de facto geospatial applications. A recent piece by Ioannidis, Garbis, Kyzirakos, Bereta, and Koubarakis (2021) provided a benchmark on geospatial RDF stores from a computational perspective; however, the RDF/triple stores they selected, such as Parliament and Strabon (Kyzirakos, Karpathiotakis, & Koubarakis, 2012), may not be the most active platforms, and the evaluation contains only a single type of semantic data repositories: the triple stores. Jovanovik, Homburg, and Spasić (2021) conducted a comprehensive review on the compatibility of various triple stores with GeoSPARQL (SPARQL with Geospatial functions; SPARQL: Simple Protocol and Rdf Query Language); however, computational performance is not evaluated.

In recent years, there has also been interest in exploiting the use of "property graphs" to model knowledge graphs, which has shed new light on semantically managing large datasets (Alocci et al., 2015). Compared with an RDF triple, a property graph carries more semantic meanings in a single "triple" unit. The "predicate" is not only a connection between entities but it can also have additional properties to semantically enrich that connection. For instance, a typical RDF triple to express a hurricane event is <Hurricane Katrina, landed in, Louisiana>. In a property graph, a property "on August 25, 2015" can be added to the predicate "landed in" to enrich the triple with additional information. Hence, a "triple" in a property graph can carry more semantics than an RDF triple. According to Miller (2013), a very popular property graph database is Neo4j, which stores data in a connected state and delivers deeper context for intelligent analytics.

A third type of semantic data repository is based on relational spatial databases. As geospatial data possess complex geometry information, spatial reasoning and query performance has become a bottleneck to many RDF/triple store solutions, which were originally designed to handle non-spatial data. But spatial databases, such as PostGIS, are capable of providing high-efficiency data storage, indexing, and spatial queries. Hence, there have also been attempts to exploit the use of relational spatial databases to achieve semantic data management through building virtual knowledge graphs. To support a unified semantic-spatial query interface with other RDF stores, a semantic connector is developed on top of the relational databases to receive standard semantic queries in SPARQL format and translate it into the SQL (Structured Query Language) for interacting with the backend spatial database. These solutions are also known as Ontology-Based Data Access (OBDA). Ontop is a flagship OBDA platform that supports standard spatial-semantic queries through a virtualized RDF created from data in a relational database (Bereta, Xiao, & Koubarakis, 2019; Can, Sezer, Bursa, Unalir, & O., 2017). Sparqlify is a similar solution to Ontop, but its performance in handling spatial queries of big data is inferior to Ontop, according to a recent study (Ding, Xiao, Pano, Stadler, & Calvanese, 2021).

Although exciting progress in developing and enhancing semantic data repositories have been made, there is still a lack of systematic studies to compare and evaluate existing solutions on their performance in handling spatial data and spatial queries in a sizable knowledge graph. This void hinders our ability to choose and use proper storage and query platforms when it comes to large-scale knowledge graph applications with spatial data. This paper addresses this problem. We surveyed the most active semantic data repositories, from open source (e.g., RDF4j, Apache Fuseki, Virtuoso) to commercial solutions (e.g., GraphDB), from triple store implementations (e.g., RDF4j, Apache Fuseki, Virtuoso) to property graph database (e.g., Neo4j), as well as an OBDA solution (Ontop), which builds a layer of semantic and spatial queries based on traditional relational databases. We also conducted systematic experiments to evaluate the performance of these semantic data repositories in handling spatial-semantic queries.

The rest of the paper is organized as follows: Section 2 reviews existing semantic data repositories. Section 3 compares their capabilities in supporting spatial queries involving both geometric and topological operations, as well as compares community activeness. Section 4 evaluates the computational efficiency and scalability of the semantic repositories in handling different types of spatial-semantic queries. Section 5 concludes the paper and discusses future research directions.

## 2. Review of semantic data repositories

In this section, we provide a review of popular semantic repositories that fall into three categories: triple stores, property graph database, and relational spatial database with a semantic connector (a type of OBDA).

### 2.1. *Eclipse RDF4j™*

Eclipse RDF4j™ is a Java framework for processing and handling RDF data. This includes creation, storage, parsing, logic reasoning, and semantic querying with RDF and Linked Data (RDF4J, 2022a). RDF4j was formerly known as Sesame and forked from it in 2016. As an open-source RDF data processing framework, RDF4j supports all mainstream RDF file formats and adapts to a wide range of triple store engines, such as GraphDB, Amazon Neptune, Blazegraph, Virtuoso, and Strabon, among others (RDF4J, 2022a). It acts as an integration API endpoint for SPARQL queries. For geospatial extension, RDF4j imports Spatial4J and JTS (Java Topology Suite, a widely used java package for topology calculation) libraries for geospatial reasoning (Batory, Lofaso, & Smaragdakis, 1998). WKT (Well-Known Text) is used for geospatial data representation. RDF4j implements a full set of functions in the OGC (Open Geospatial Consortium) GeoSPARQL specification, which includes common non-topological query, Simple Feature, Egenhofer, and RCC8 (Region connection calculus) functions. The Lucene geospatial index was introduced to improve the query performance on large datasets (RDF4J, 2022b).

### 2.2. *Apache Jena GeoSPARQL Fuseki*

Apache Jena is a popular free and open-source Java framework for building Semantic Web and Linked Data applications (Jena, 2014). It provides rich APIs for manipulating RDF graphs. To enable geospatial model handling, the Apache Jena GeoSPARQL module, once launched, implements the OGC GeoSPARQL 1.0 standard for SPARQL query (Jena, 2022). To establish the GeoSPARQL web service endpoint, Apache introduced GeoSPARQL Fuseki as a web application server, which combines the capabilities of Jena GeoSPARQL and Jena Fuseki to provide a Web accessing endpoint. Its implementation follows the GeoSPARQL standard, and all three spatial relation families are supported: Simple Feature, Egenhofer, and RCC8. In the geospatial layer of Jena GeoSPARQL, the JTS library is imported and provides support for geometry representation, spatial relation calculations, and spatial index. Apart from the WKT format, Jena GeoSPARQL supports GML (Geography Markup Language) by serializing the shape geometry to this standardized GML representation.

### 2.3. *GraphDB*

Ontotext GraphDB, formerly known as Owlim, is a commercial semantic graph database engine and database management system (Güting, 1994). GraphDB implements its own native storage strategy for

managing RDF triples. A query optimizer and reasoner works on top of the data storage for speeding up incoming semantic queries and performing reasoning through forward-chaining of entailment rules. GraphDB can be packaged as a Storage and Inference Layer (SAIL), which is compliant with RDF4j, so that it can be used as the data storage backbone for RDF4j. GraphDB supports multiple query languages (QL), including GraphQL, SPARQL, and SeRQL (Sesame RDF Query Language), as well as RDF serialization formats (e.g., RDF/XML, N3, Turtle). The GraphDB GeoSPARQL plugin provides access and query for geospatial data. GraphDB supports two-dimensional geospatial data that uses the WGS84 (World Geodetic System) as the projection system. A set of topological SPARQL extension functions are implemented to support quantitative reasoning.

### 2.4. Neo4j

Neo4j is a popular graph database management system implemented in Java, which is available in an open-source "community edition" under the GPL3 (General Public License) license (Webber, 2012). Compared with other graph databases, Neo4j is highlighted by its active and vibrant developer communities and the number of use cases. For geospatial data handling, the Neo4j Spatial library enables spatial operations on graph data. Like most Java-based engines, Neo4j Spatial relies on the JTS library to enhance its geospatial capabilities. Developers can create spatial indexes and perform spatial operations on the data, such as searching for data within a specified region or within a specified distance of a point of interest (POI). In addition, Neo4j Spatial also provides plugins for several popular open-source GIS software, such as GeoTools, GeoServer, and uDig (Neo4j, 2017). Unusually, Neo4j uses its own query syntax, the Cypher query language, which does not comply with the OGC GeoSPARQL specification.

### 2.5. Ontop

Ontop is a virtual knowledge graph system. It exposes the content of arbitrary relational databases as knowledge graphs. Other than designing a new native triple store to accommodate graph data, Ontop chooses the full-fledged relational database engine as its datastore infrastructure and provides a SPARQL to SQL syntax transformation layer to convert semantic queries into SQL queries to perform on top of the backend database. These graphs are virtual, which means that data remain in their original storage space and "native formats" instead of being moved to another database or converted to graph data. It takes advantage of lightweight ontologies and maps relational database schemas to RDF schemas (Calvanese et al., 2017). It then translates SPARQL queries expressed over knowledge graphs into SQL queries executed by the relational databases. Ontop-spatial is an extension of the Ontop framework with additional geospatial support. Relying on the powerful capacity of geospatial databases, such as PostgreSQL with PostGIS extension enabled, Ontop-spatial can provide all the topology functions defined in GeoSPARQL with high efficiency. As the GeoSPARQL support has been added in the standard distribution of Ontop since v4.1 (Cogrel et al., 2022), we call the system Ontop throughout the paper for consistency.

### 2.6. Blazegraph

Blazegraph is an open-source triple store and graph database with ultra-high performance. The Blazegraph database is used in the Wikidata SPARQL endpoint and by other commercial customers. It advocates supporting up to 50 billion triples on a single machine and powers the Wikimedia Foundation's Wikidata Query Service. Unfortunately, Blazegraph's geospatial support is limited. It can only handle Point geometry, and the project does not seem to be under active development.

### 2.7. Virtuoso

OpenLink Virtuoso is a well-known universal server consisting of an SQL Object-Relational Database Management System (ORDBMS) and a Web Application Server. In terms of semantic storage and query capability, Virtuoso provides both Web APIs and command line tools to import data in the RDF data model and digests and stores the data as RDF quads (a triple plus an identifier of the parent graph) in its internal column-oriented relational database (Huang, Raza, Mirzov, & Harrie, 2019). Spatial computing-wise, Virtuoso provides efficient GeoSPARQL built-in functions with accelerated query speed based on the support of a spatial index. Virtuoso also provides GeoSPARQL compliant query functionality. However, queries based on GeoSPARQL functions in Virtuoso are not as efficient as its built-in functions, because a spatial index cannot be applied in such cases. Although the Openlink Virtuoso (open-source version) is not as scalable as its commercial counterpart, it maintains the same backend storage and spatial index.

The next section compares the capabilities of these semantic repositories in support of spatial data and queries. We also select a few of the more representative queries for evaluation of their computational performance.

## 3. Comparison of semantic data repositories on their capability to support spatial queries

In this section, we compare the capabilities of semantic repositories from three perspectives: support for geometry operations, support for topological relationship identification, and community activeness. Fig. 1 presents popular geometry operations between two features (intersection, union, difference, and symmetric difference) and for a single feature (convex hull, buffer, envelope, and boundary).

Table 1 provides a comparison of the popular semantic repositories on their support to major geometry operations and some other auxiliary functions, such as returning a geometry as WKT. It can be seen that blazegraph offers very limited to no support for these spatial operations, whereas RDF4j, Fuseki, GraphDB, Neo4j, Virtuoso, and Ontop-spatial provide strong support for spatial operations, such as buffer and distance. Their capabilities for some supporting functions such as returning the geometry asWKT and asGML and getting the reference system information (getSRID) vary quite a bit.

Table 2 further compares the capabilities of different semantic data repositories in their support for topological relations identifications that do not require returning geometries but a True or False answer. The illustration of these topological relationships can be found in Fig. 2. These operations can leverage spatial indexes built as part of the semantic repositories to speed up query performance. Again, RDF4j, Fuseki, GraphDB, Neo4j, Virtuoso, and Ontop provide stronger support for these functions than Blazegraph.

We also compared the community activeness of these projects aiming to provide novel semantic data storage and queries (Table 3). It can be seen that Neo4j is a very active project with an update frequency in days. The same applies to RDF4j, Fuseki, and Ontop. The other projects are not very popular due to being close-sourced (e.g., GraphDB) or lack of community interests (e.g., Blazegraph), measured by number of contributors and update frequencies.

From the above analysis, we selected six semantic web storage solutions which maintain high community activeness and comprehensive support for geospatial and topological operations to evaluate their computational performance in support of spatially enabled semantic applications. These platforms, covering both open-source and commercial solutions, include RDF triple stores (RDF4j, Fuseki, GraphDB, Virtuoso), property graph database (Neo4j), and the semantic connector + relational database solution (Ontop). Virtuoso is also a triple store; the difference between Virtuoso and the other listed triple stores is that instead of using a native storage system, its backend uses a relational database for managing triples.

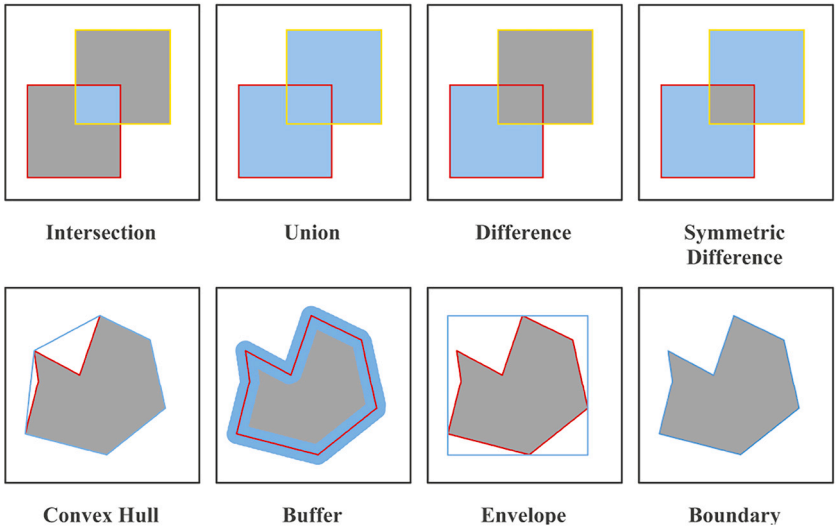



**Fig. 1.** Geometry operations for spatial analysis. Area or line in blue is the expected result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 1**
Comparison among the semantic data repositories on their support for geometry operations. SRID: Spatial Reference Identifier. The following versions of software were used in comparing the capabilities and follow-on experiments: RDF4j 3.6.2, Jena Fuseki 4.0.0, GraphDB 9.10.0, Neo4j 4.2.5, Virtuoso 7.20.3233, and Ontop 4.1.0 (backend with Postgres 13.2 and PostGIS 3.1.1).

| Capabilities | RDF4J | GeoSPARQL Fuseki | GraphDB | Neo4j | Blazegraph | Virtuoso | Ontop |
|---|---|---|---|---|---|---|---|
| buffer | T | T | T | T | | T | T |
| convexHull | T | T | T | T | | T | T |
| envelope | T | T | T | T | | T | T |
| boundary | T | T | T | T | | T | T |
| distance | T | T | T | T | | T | T |
| intersection | T | T | T | T | | T | T |
| union | T | T | T | T | | T | T |
| difference | T | T | T | T | | T | T |
| symDifference | T | T | T | | | T | T |
| asWKT | T | T | T | T | | T | T |
| asGML | T | T | T | T | | T | |
| getSRID | T | T | | | | T | T |
| CRS Support | | T | T | | | T | T |

**Table 2**
Semantic repositories' support for topological relationship identification.

| Capabilities | RDF4J | GeoSPARQL Fuseki | GraphDB | Neo4j | Blazegraph | Virtuoso | Ontop |
|---|---|---|---|---|---|---|---|
| equals | T | T | T | T | | T | T |
| disjoint/intersects | T | T | T | T | | T | T |
| contains | T | T | T | T | T | T | T |
| inside/within | T | T | T | T | T | T | T |
| overlap/crosses | T | T | T | T | | T | T |
| meet/touches | T | T | T | T | | T | T |
| covers | T | T | T | T | | T | T |
| converedBy | T | T | T | T | | T | T |

These semantic repositories will be compared on both simple and complex queries, non-spatial and spatial semantic queries, and queries over increasing size of datasets. The next section describes the experimental data and processing workflow.

## 4. Data and processing workflow

We prepared three major types of geospatial data: point data, polyline data, and polygon data. The point data are from the POI (point of interest) dataset. The polyline data are US highway data. These two datasets are from OpenStreetMap (OSM) downloaded through Mapcruzin (2022). While we derived the OSM data in the Esri shapefile format, the datasets are also available in the RDF format as part of the LinkedGeoData effort (Stadler, Lehmann, Höffner, & Auer, 2012). The polygon dataset is from the Soil Survey Geographic Database (SSURGO) of the USDA (US Department of Agriculture) downloaded through Esri (2022). These data are important data sources to analyze how agriculture and supply chain influence people's food selection and quality on the table. We retrieved data in four US states: Arizona, California, Oregon, and Utah. They each have an attribute table containing non-spatial properties about the spatial features. To understand the scalability and computational efficiency of different semantic repositories, we prepared subsets of these data at different sizes, ranging from 50 k spatial features to 800 k spatial features. Although we use these data for

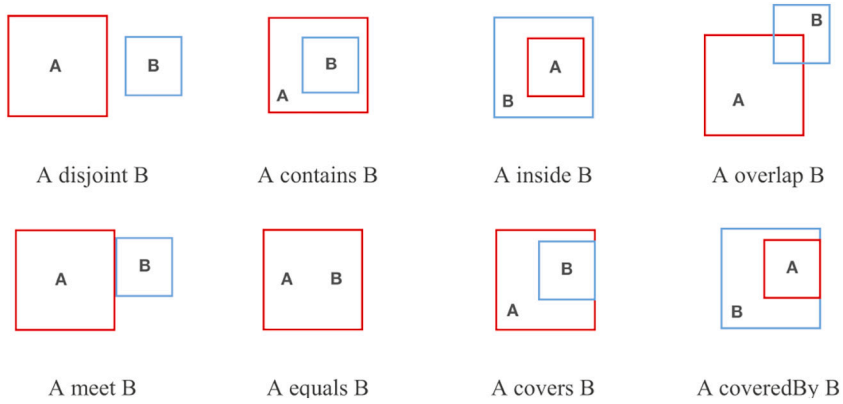



**Fig. 2.** Egenhofer topological relations.

**Table 3**
Community activeness (as of February 1, 2022). Reference to DB-Engines' scores and rankings can be found in DB-Engines (2022).

| Capabilities | RDF4J | GeoSPARQL Fuseki | GraphDB | Neo4j | Blazegraph | Virtuoso | Ontop |
|---|---|---|---|---|---|---|---|
| contributors | 72 | 74 | | 216 | 10 | 17 | 34 |
| stars | 268 | 791 | | 9600 | 649 | 725 | 460 |
| forked | 140 | 543 | | 2100 | 137 | 197 | 136 |
| update frequency | days | days | | days | year | months | days |
| DB-Engines Score | 0.67 | 2.71 | 2.86 | 58.03 | 0.96 | 5.37 | |
| DB-Engines Ranking | 209 | 106 | 103 | 20 | 180 | 73 | |
| Database model | RDF | RDF | Graph, RDF | Neo4j | Graph, RDF | Graph, RDF, R | RDB, RDF |

experiments in this study, our experimental framework is generalizable and applicable to other kinds of non-spatial and spatial datasets.

Fig. 3 demonstrates the data processing framework for setting up a reproducible experimental framework by establishing a docker environment that allows standard and easy (re)deployment of testing environments for each semantic repository, as well as the reproducibility of our results (Goodchild & Li, 2021; Wilson et al., 2021). First, different types of spatial data are made available in the popular Esri shapefile or other formats, that is, WKT (Well-Known Text). Then a knowledge graph auto-generation tool is developed to automatically convert the geospatial data into a knowledge graph-ready format, such as RDF, based on a predefined ontological schema. Here, the raw geospatial data are converted into two formats to be imported into different types of semantic data repositories. A Turtle file (Terse RDF Triple Language) (Beckett, 2008) is created for importing the raw data into RDF triple stores as well as property graph databases. A structured CSV file with the geometry data encoded in WKT format is created for importing the geospatial data into spatial relational databases that serve as the backend data storage for the OBDA platforms, such as Ontop. An Esri shapefile is also an acceptable input format for spatial relational databases (SDB). For Ontop, a relational-data-table to ontological-schema mapping file is also created for the on-the-fly generation of a virtualized knowledge graph using data retrieved from the SDB. The mapping file will also be used to convert a (Geo)SPARQL query from end users to a SQL (Structured Query Language) query in the backend database.

Once the data are converted into knowledge graph-ready format and imported into the semantic data repositories (stores), the next step is to construct spatial-semantic queries for performance evaluation. Table 4 lists the type of queries and the actual GeoSPARQL used to query the semantic stores. Queries 1–8 contain purely spatial queries for both geometry operations and topological relationship identification. We have selected a list of spatial queries that are commonly used in real-world applications. For instance, a question about "How many biodiversity conservation sites (point or polygon) are there in Oregon (polygon)?" could make use of Query 4 or Query 8 in Table 4 to get the answer. Queries 9–10 contain more complex semantic queries involving multiple datasets/data types. Query 9 presents a complex non-spatial semantic query and Query 10 presents a complex query containing both spatial and semantic operations.

Next, to enable a reproducible workflow, we deployed all the semantic data stores into a docker environment, which is a containerization platform that allows packaging applications and its running environment into a virtualized container to share and reuse anywhere. All these semantic stores will expose a semantic search interface to allow spatial queries from one or more remote clients. As a new type of semantic storage solution, Neo4j does not provide a native web interface for spatial queries and uses a different query language, Cypher, that is not compliant with GeoSPARQL. To ensure all the experiments were completed in a similar environment, a Web API based on the REST (Representational State Transfer) standard was developed to enable remote queries.

## 5. Experimental design methods and results

### 5.1. Experimental design

We evaluated the performance of the semantic data repositories in two key areas: storage space and query response time. For the first, we compared the storage taken by the different platforms to store the same amount of information in a knowledge graph. Additional space is used by these stores to build spatial and non-spatial indices, as well as save other auxiliary information. For evaluating the query response time, we designed a set of experiments to measure the response speed of a semantic data repository in handling a single request using queries 1–8 listed above. Two thousand queries of each spatial query type (e.g., line intersection) were randomly generated with different query parameters using the experimental dataset to create a query pool. A hundred queries were then randomly selected and sent to each semantic store sequentially. The averaged response time from these 100 queries was used to measure how fast a semantic repository responds to a specific type of spatial query.

All the experiments were performed on the same machine, with the following configuration: 2 CPUs, each with 20 cores running at 2.1 GHz and 128GB DDR4 memory running at 2933 MT/s. Software wise, every

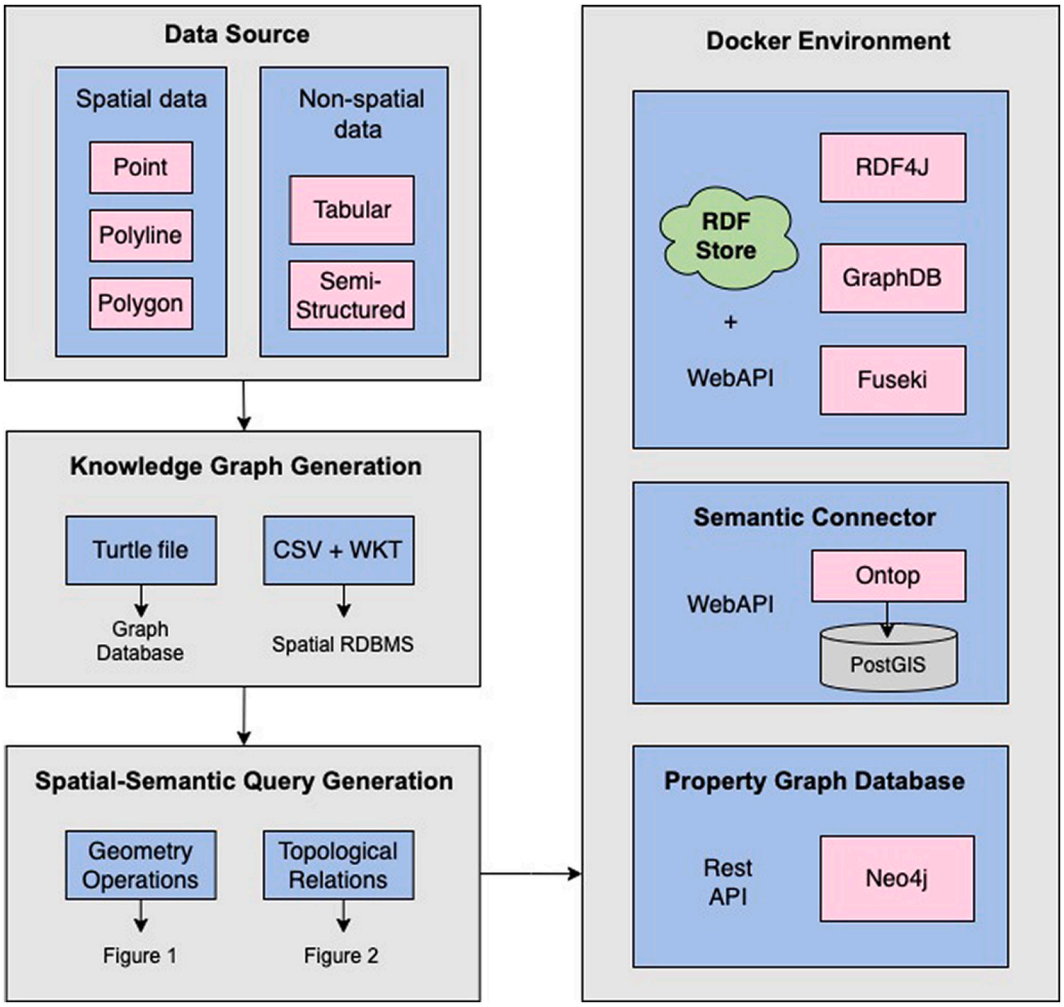



**Fig. 3.** A reproducible, semantic data processing framework for performance evaluation of semantic data repositories. RDBMS: Relational Database Management System.

triple store used in experiments is wrapped in an individual docker container to maintain a clean and isolated experimental environment.

### 5.2. Storage space comparison

Fig. 4 compares the size increase after importing the original spatial data with different sizes (increased from 50,000 spatial features to 800,000 spatial features). The result shows that Ontop was most efficient in terms of storage, since the storage space after data importation is much smaller (less than half) than that of the other semantic repositories. This is because Ontop builds upon a spatial relational database, PostGIS which uses a very efficient generalized search tree as its spatial index, so that it can achieve fast query speed. Another possible reason for the memory advantage of Ontop is that the spatial data, particularly the point, line, and polygon geometries, are stored in binary formats in the spatial databases, hence saving much storage space.

In contrast, the spatial data take up a much larger space in GraphDB and Virtuoso. GraphDB uses R-tree as the spatial index. Its default quadtree encoding provides location accuracy at different levels. The higher the level, the deeper the spatial index tree and thus the larger the index file. We use the default level value 11 in this research. At level 11, the accuracy of the spatial index was at $\pm$2.5 km at the equator. Virtuoso also has a very high storage consumption from its column-wise indexing systems. The storage space for Fuseki, RDF4j, and Neo4j are about the same. RDF4j builds its index based on Lucene SAIL (Storage and Inference Layer), and thus provides both spatial and non-spatial indices. Apache Fuseki uses STR-Tree, an R-tree created using the Sort-Tile-Recursive (STR) algorithm. Similar to GraphDB, RDF4j also uses an R-tree spatial index.

### 5.3. Comparison of semantic data repositories' performance on spatial-semantic queries

We next conducted performance evaluations in terms of query response time (average over 100 queries) for each of the spatial queries (Q1–Q8) in Table 4. Fig. 5 shows the response time for Q1, which is to find the three nearest points of a given point. Virtuoso and Ontop are the top performing platforms. For datasets smaller than 200 k, Virtuoso can return results within 100 ms, and the range for Ontop's response time is between 100 and 350 ms. When the dataset is larger (at 400 k and 800 k spatial features), Virtuoso can return results between 150 ms–350 ms, while Ontop can return results between 500 ms-1 s. Although Virtuoso is a triple store solution, it uses a relational database to store graph data, and it adopts a more deeply coupled query model between the semantic query engine and the database engine. These features make Virtuoso very efficient in handling spatial queries. Ontop stores the structured data in a relational table rather than as an RDF model so that it can leverage the built-in function of a spatial database to perform the queries efficiently. Different from Virtuoso, Ontop develops a SPARQL to SQL translation layer to conduct the query so that all the datasets can be stored in their original native structure in the relational database without the need of any data conversion.

**Table 4**
Spatial-semantic queries.

| Queries | Type | GeoSPARQL |
|---|---|---|
| Q1 | Nearest points using distance function | PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/><br>PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX poi: <http://cici.lab.asu.edu/poi#><br>SELECT?g2<br>WHERE {<br>poi:416935 geo:asWKT?g1,<br>?f geo:asWKT?g2,<br>FILTER (?f! = poi:416935)<br>}<br>ORDER BY ASC (geof:distance(?g1,?g2, uom:meter))<br>LIMIT 3 |
| Q2 | Line intersection | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>PREFIX highway: <http://cici.lab.asu.edu/highway#><br>SELECT?id?p<br>WHERE {<br>highway:16269 geo:asWKT?g1,<br>?f xsd:ID?id,<br>?f geo:asWKT?g2,<br>BIND(geof:intersection(?g1,?g2) AS?p)<br>FILTER(?id! = 16,269 && geof:sfIntersects(?g1,?g2))<br>} |
| Q3 | Polygon boundary | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX ssurgo: <http://cici.lab.asu.edu/ssurgo#><br>SELECT?b<br>WHERE {<br>ssurgo:423338 geo:asWKT?geom,<br>BIND(geof:boundary(?geom) AS?b)<br>} |
| Q4 | Contains (Polygon, Point) | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>SELECT?id<br>WHERE {<br>?f xsd:ID?id,<br>?f geo:asWKT?geom,<br>BIND("POINT (−120.49973620479315 34.805647906537345)"^^geo:wktLiteral AS?p)<br>FILTER (geof:sfContains(?geom,?p))<br>} |
| Q5 | Convex hull (Polygon) | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX ssurgo: <http://cici.lab.asu.edu/ssurgo#><br>SELECT?h<br>WHERE {<br>ssurgo:635480 geo:asWKT?geom,<br>BIND(geof:convexHull(?geom) AS?h)<br>} |
| Q6 | Envelope (Polygon) | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX ssurgo: <http://cici.lab.asu.edu/ssurgo#><br>SELECT?env<br>WHERE {<br>ssurgo:9494 geo:asWKT?geom,<br>BIND(geof:envelope(?geom) AS?env)<br>} |
| Q7 | Intersects (Polygon, Polygon) | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#><br>SELECT?id?label<br>WHERE {<br>?f xsd:ID?id,<br>?f rdfs:label?label,<br>?f geo:asWKT?geom,<br>FILTER (geof:sfIntersects(?geom, "POLYGON ((−118.191746173483 47.5371825726328, −118.191746173483 47.5394688885643, −118.186886287796 47.5394688885643, −118.186886287796 47.5371825726328, −118.191746173483 47.5371825726328))"^^geo:wktLiteral))<br>} |
| Q8 | Within (Point, Polygon) | PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>SELECT?id<br>WHERE {<br>?f xsd:ID?id,<br>?f geo:asWKT?geom,<br>BIND("POINT (−115.61972444331998 33.04264614096238)"^^geo:wktLiteral AS?p)<br>FILTER (geof:sfWithin(?p,?geom))<br>} |
| Q9 | Complex non-spatial query: Return all SSURGO units with area <500 square meters | PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#><br>PREFIX ssurgo: <http://cici.lab.asu.edu/ssurgo#><br>SELECT?ssurgoName<br>WHERE {<br>?z xsd:ID?surgoID.<br>?z rdfs:label?ssurgoName.<br>?z ssurgo:Area?area.<br>FILTER (?area < 500)}<br>GROUP BY?ssurgoName?area<br>ORDER BY DESC(?area)<br>LIMIT 1 |
| Q10 | Complex spatial query: Return the average distance between grocery stores and their closest road networks | PREFIX xsd: <http://www.w3.org/2001/XMLSchema#><br>PREFIX geo: <http://www.opengis.net/ont/geosparql#><br>PREFIX geof: <http://www.opengis.net/def/function/geosparql/><br>PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/><br>PREFIX: <http://cici.lab.asu.edu/ontology/semantic_connector#><br><br>SELECT (AVG(?minValue) as?avgDist){<br>SELECT DISTINCT?x (MIN(?d) AS?minValue)<br>WHERE {<br>?x:pointOfInterestCategory "Grocery store"^^xsd:string.<br>?x geo:hasGeometry?cGeom,<br>?cGeom geo:asWKT?cWKT,<br>?y a:RoadFeature. |

(*continued on next page*)

**Table 4** (*continued*)

| Queries | Type | GeoSPARQL |
|---|---|---|
| | | ?y geo:hasGeometry?fGeom,<br>?fGeom geo:asWKT?fWKT.<br>BIND(geof:distance(?cWKT,?fWKT,uom:<br>metre) as?d)<br>} GROUP BY?x<br>} |

For the other semantic data stores, which run slower than the first-tier solutions, Neo4j, GraphDB, and RDF4j have similar query performance, with GraphDB slightly slower than the other two. Fuseki's performance was relatively good when data size was small; however, as the data grow to a certain size (400 k spatial features and more), the query response time increased exponentially. This may be largely due to the following reasons. First, Fuseki has a very aggressive memory use strategy. With the same amount of graph data loaded, Fuseki's memory consumption is much more than the other semantic repositories. For instance, in our current experimental setting and for loading 800 k SSURGO polygonal records, Fuseki's memory use is twice (43.8GB) as much as Virtuoso (21.84GB), and about five times as much as GraphDB (8.46GB) and about 90 times of Ontop (0.48GB)). Second, substantial CPU time is allocated for conducting Java Garbage Collection (JGC) during the execution of a query. For Q4 run on the 800 k SSURGO data, the total query execution time is around 115 s (Fig. 7a) for Fuseki (which runs on a single CPU core) and the JGC has taken 8.23 s (which runs parallelly on 40 CPU cores). It is expected that the JGC will take up more CPU resources when requests arrive concurrently. Both issues negatively affect its query performance. As the nearest neighbor calculation does not involve the use of spatial index, the resultant query time directly reflects algorithm efficiency in the implementation of different semantic repositories.

Fig. 6 shows the results for spatial query on line intersection. As data sizes increased, Fuseki's performance drops significantly, as reflected by

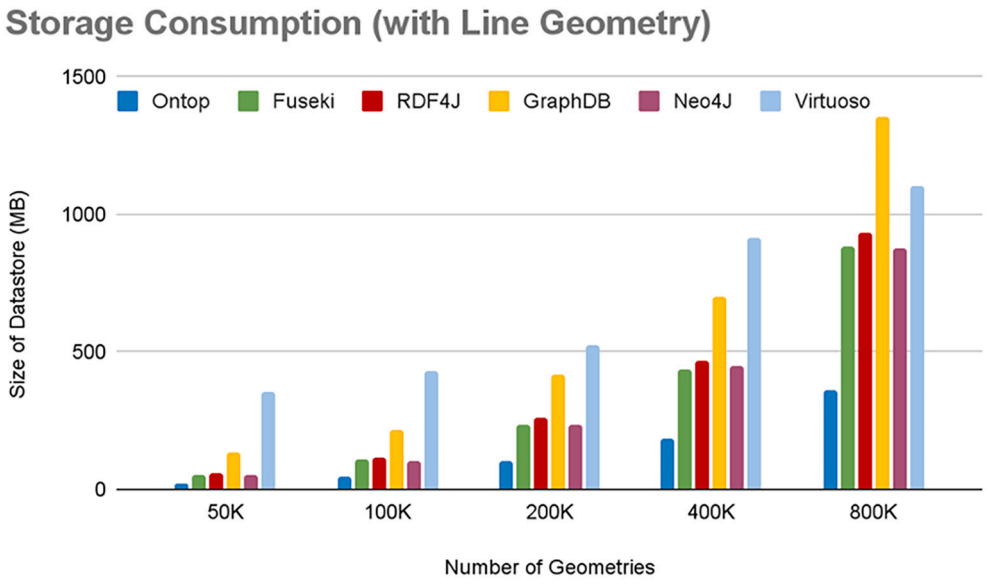



**Fig. 4.** Comparison of storage space consumption among different semantic repositories with an increasing size of line geometries.

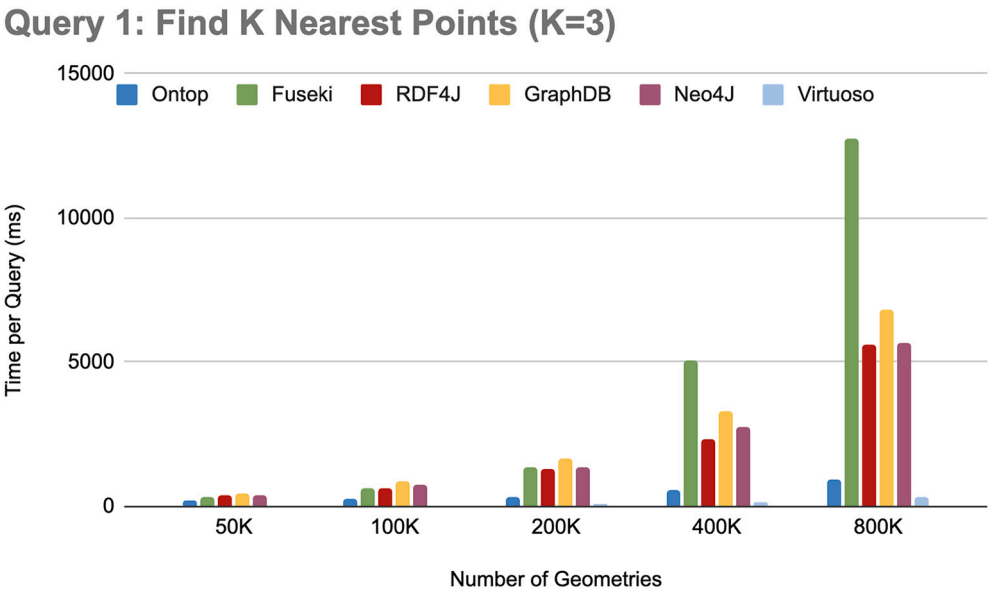



**Fig. 5.** Query performance on k nearest neighbors/points (k = 3).

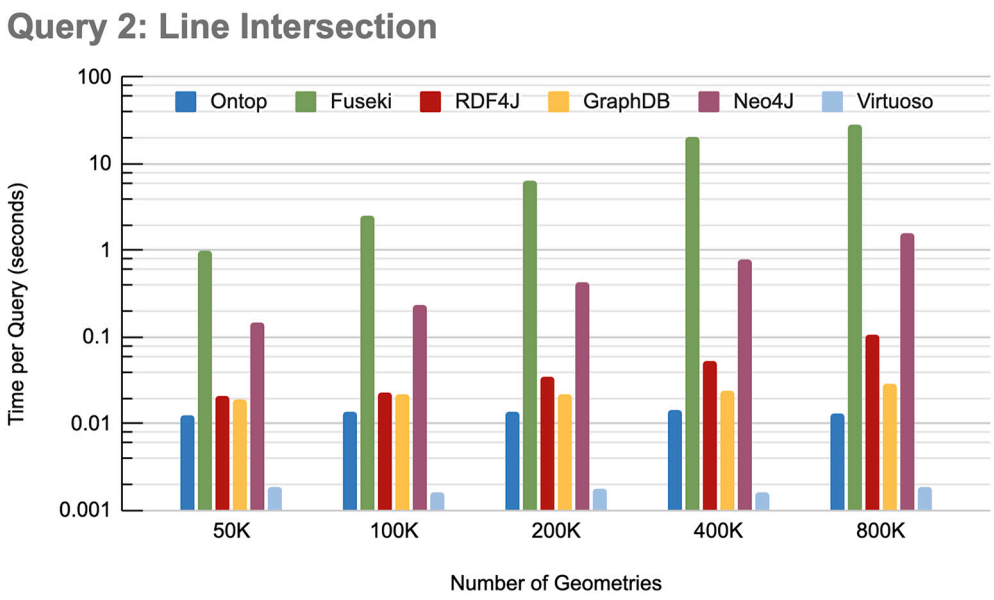



**Fig. 6.** Query performance on line intersection. Note: Y axis uses a base-10 log scale instead of a linear scale to make the very small numbers and large numbers easy to read. The minor ticks between the main data points [x1, x2] represent 2, 4, 6, 8 *(x1). For instance, the four ticks between [0.01,0.1] refer to 0.02, 0.04, 0.06, and 0.08, respectively.

the overly long response time. The overall response time for Fuseki was linearly correlated with the data size, and its response time (at the scale of 1–30 s) was nearly 100 times longer than good performing semantic stores, such as Ontop. (Note that the Y axis uses a base-10 log scale, so the difference between the values of the bars was actually much larger than it appears as the bar gets taller). This is possibly due to suboptimal memory management of Fuseki, as discussed above. Neo4j was the second slowest among all the semantic stores in processing the line intersection query. But it was about 10 times faster than Fuseki. Also, we can still observe a linear correlation between the response time and the data size; this means a spatial index was utilized by the system during this query but not very efficiently.

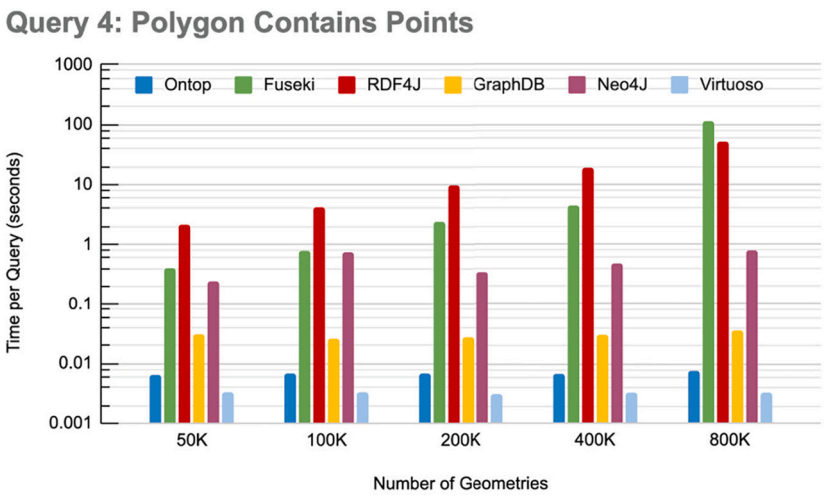
In comparison, Ontop, RDF4j, and GraphDB have demonstrated better performance in this query. The results can be returned within 100 ms for queries of all given data sizes. Of the three repositories, Ontop performed the best, and its response time was consistently low (<15 ms for all datasets) and was almost unchanged with the data size. This means that the spatial index takes effect to process this query efficiently. The second best was GraphDB, and similar to Ontop, the response was almost invariant of the changing data size, indicating the proper applications of the spatial index, but speed-wise, it was about half as fast as Ontop. RDF4j showed a linear increase in response time as the data size increases, although slower than Ontop and GraphDB, it achieved a performance that is one magnitude better than Fuseki and Neo4j.

Virtuoso was overall the best performing semantic store on this query. For all datasets, the results were returned within 2 ms. In comparison, the second fastest platform Ontop returned results within 15 ms in our experiments. This difference is likely due to the more coupled solution between data storage and data query in Virtuoso. In Ontop, however, a translation from (Geo)SPARQL query to a database query is always needed and brings more overhead.

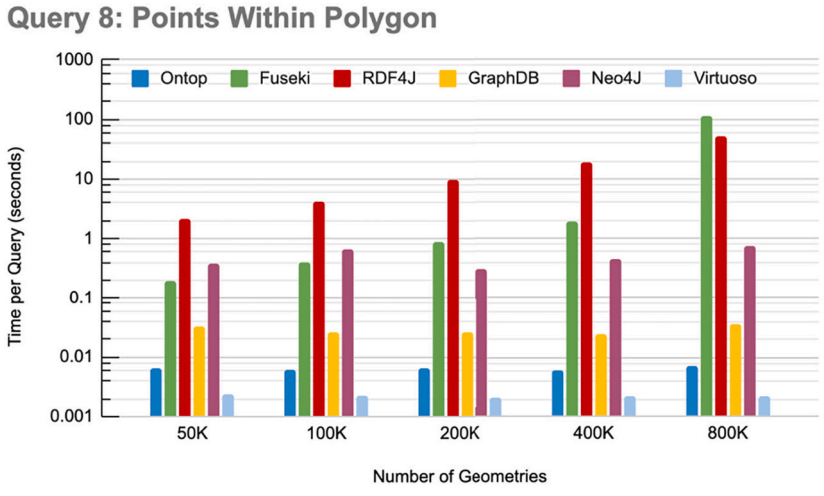
Fig. 7 further demonstrates the query performance for *contains* and *within*, which are two membership relations. If a spatial feature A contains B, we can infer that B is within A. In Fig. 7(a) and (b), A and B are polygon and point features, respectively, and they both are polygons in Fig. 7(c). As these queries do not return any geometry but a "True" or "False" answer, they share similar computational complexity, so we group the results of these three queries together. The results from these queries also show similar trends in the response time.

In this set of results, Virtuoso still worked the best in terms of query speed, followed by Ontop. The difference in response time (about 2 times) was smaller than performing line intersection on these two platforms. GraphDB was the third fastest one, but it was five-times slower than Ontop for point-polygon containment operations (Fig. 7a and b) and three-times slower than Ontop for polygon-polygon containment operations (Fig. 7c). All three solutions showed nearly invariant response time as the data size increased, demonstrating the efficiency of spatial indexing in assisting a quick search and location of spatially related objects in space. The opposite scenarios were found in the results of Fuseki and RDF4j, where their response time increased with the data size. The increase in Fuseki even showed an exponential instead of a linear trend, indicating that besides the issue in using and applying spatial index, Fuseki's performance in handling large datasets is also worrisome. Neo4j, the platform that manages property graphs, can also handle these spatial queries with the support of spatial index. The query response time remained largely unchanged among different sizes of datasets. The longer response time (than Virtuoso, Ontop, and GraphDB) indicates that it has more overhead in processing the query. But these four solutions all responded to these given requests in under 1 s.

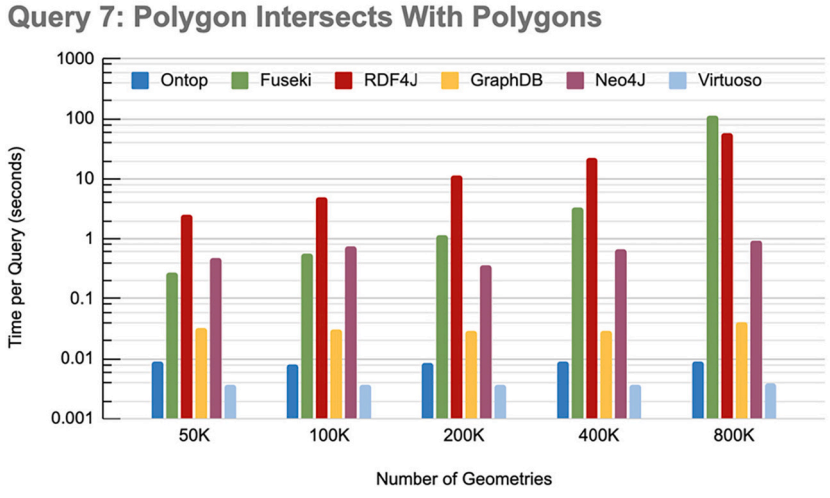
Fig. 8 demonstrates the query performance on three spatial queries that operate on a single feature: (a) Boundary operation returns the combinatorial boundary of a spatial feature; (b) Convex Hull operation calculates the minimal convex polygon for a given spatial feature; and (c) Envelop returns the bounding box (BBOX) of a spatial feature. In this case, because only a single feature is involved in the computation, and no spatial index is needed in the spatial algorithm, almost all the semantic repositories returned results in a timely manner (the scale of Y axis is quite small compared with other figures). Among these solutions, Neo4j had the least satisfying performance, because of the overhead for using its unique query language, Cypher, to query the proper graph data to select a given feature from the data collection. In addition, Neo4j's query transaction requires putting a lock on the data when doing a query and unlocking it when the query is completed. And this overhead becomes substantial in a simple query which runs fast. But overall, all the semantic data repositories returned correct results with good efficiency for queries shown in Fig. 8. The differences among the bars for the platforms are actually very subtle, except for Neo4j.




(a)

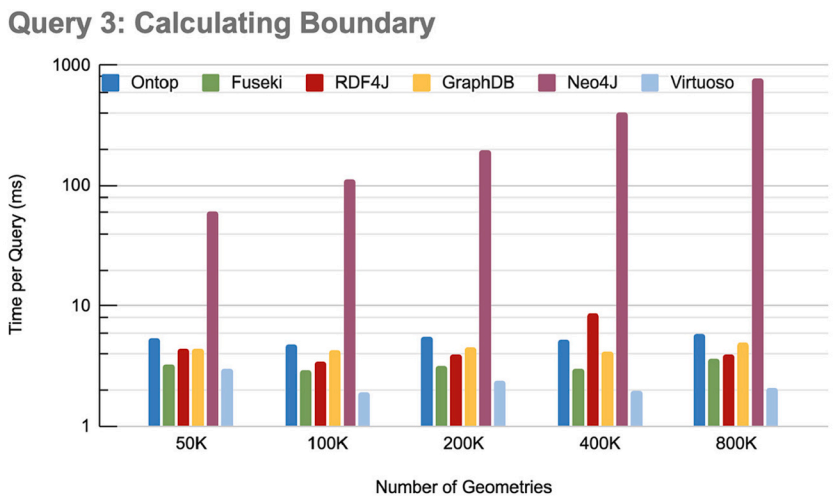



(b)

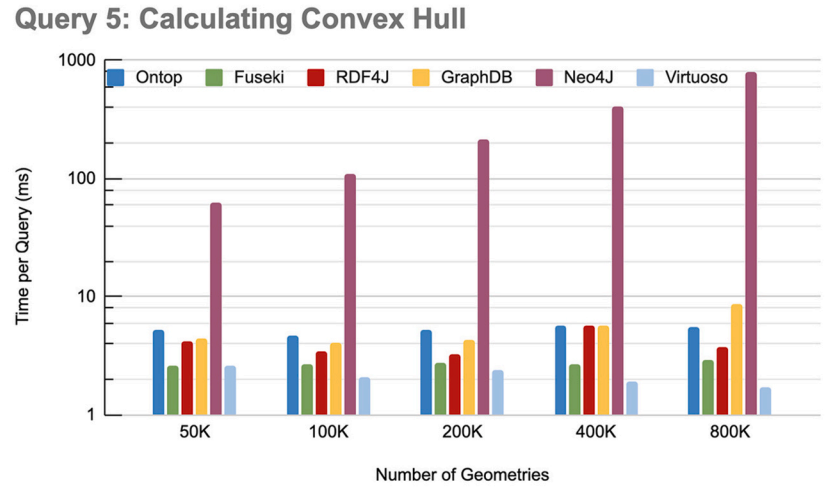



(c)

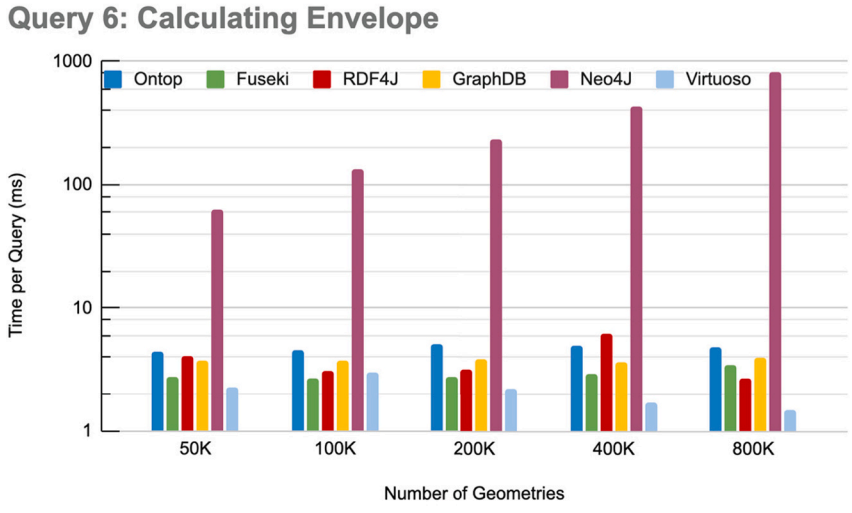
**Fig. 7.** Query performance on (a) contains (Polygon, Points), (b) within (Points, Polygon), and (c) intersects (Polygon, Polygons). Note: the Y axis uses a base-10 log scale, so the difference in the value of the bars gets much higher as the bars get taller. The minor ticks between each main data points [x1, x2] represent 2, 4, 6, 8 * (x1). For instance, the four ticks between [0.01,0.1] refer to 0.02, 0.04, 0.06, and 0.08, respectively.




(a)




(b)




(c)

**Fig. 8.** Query performance on (a) Boundary, (b) Convex Hull, and (c) Envelop calculations. Note: The Y axis uses a base-10 log scale, so the difference among the values of the bars gets much higher as the bars get taller. The minor ticks between the main data points [x1, x2] represent 2, 4, 6, 8 *(x1). For instance, the four ticks between [1,10] refers to 2, 4, 6, and 8, respectively.

### 5.4. Comparison of semantic data repositories' performance on complex semantic queries

The above experiments in Section 5.3 measured the query performance on geometric and topological operations, the results of which provide direct evidence on the efficiency and scalability of different semantic repositories in handling spatial queries in a semantically enabled way. In real-world applications, there are usually more complex queries that combine both spatial and semantic queries. Query 9 and Query 10 (Table 4) are such complex queries involving more filter and sorting operations. Query 9 is a complex non-spatial semantic query; it uses a real-world SSURGO dataset containing 200 k polygon-based soil unit data in California to find all the SSURGO units with areas <500 square meters. Query 10 is a query that combines both spatial and non-spatial semantic filters. It uses a dataset containing 22 k POI data (grocery stores) and 124 k highway lines covering Arizona and Utah for the experiment to return the average distance between grocery stores and their closest highway networks in the two states.

Fig. 9 shows the comparison on response time between the top three performers Ontop, GraphDB, and Virtuoso. Query 9 is a non-spatial semantic query. It can be seen that all can return results at the millisecond level. GraphDB has shown better performance than Ontop in this non-spatial semantic query, because of its powerful capability in handling non-spatial graph data. Virtuoso amazingly returned the results within 2 ms, illustrating very efficient performance for semantic queries. In another complex query (Q10), which contains both spatial and semantic filters, Virtuoso again worked the best, but the difference in response time between Virtuoso and Ontop was smaller than when conducting purely non-spatial queries. GraphDB was much slower than the other two in handling such types of queries.

As observed, although Ontop's performance was quite good, its overall response time remained longer than the triple store solution Virtuoso. The reason for this is two-fold: first, the spatial operation in Query 10 is a call essentially for the "nearest distance" operation, but there is no such function implemented directly in its backend spatial engine, PostGIS. Therefore, it needs the calculation of all the possible distances first and then sorts them to get the smallest one (to answer Query10). Second, the process in Ontop semantic translation (from standard GeoSPARQL query to the SQL query) will add some mandate statements on data type translation and data integrity check in the translated SQL to assure the successful execution of the queries. These extra criteria slow down the database query speed. In the future, more optimization could be applied during the semantic translation to make Ontop more efficient while maintaining the robustness of the system. In comparison, Virtuoso, which takes advantage of the efficient data management and query performance of a relational database and at the same time implements a deeply coupled SPARQL query engine with the database engine, has shown superior performance. GraphDB, which builds its system with an RDF model and a native triple storage, did not show a performance superior to the other two, particularly in handling spatial-semantic queries.

## 6. Discussion and conclusion

While recent literature has evaluated the availability and compatibility of semantic data repositories to support spatial queries, almost all focus on RDF triple stores (Ioannidis et al., 2021; Jovanovik et al., 2021; Raza, 2019) and very few have addressed this question from a computational performance perspective. In this paper, we provide a comprehensive analysis of a variety of semantic repository solutions, including RDF triple stores, property graph databases, and OBDA platforms in terms of their capabilities, community activeness, and computational efficiency for handling spatial-semantic queries. The first category represents traditional semantic data storage solutions in which graph data are modeled upon Semantic Web standards into RDF triples. The second category, property graphs databases, such as Neo4j, allows the use of a complex triple to describe a property linking two entities in order to express more semantically enriched meanings in a statement. On the other hand, there is a third type of approach for accessing spatially enabled knowledge graphs, which is known as OBDA. Instead of building a completely new semantic data store, OBDA approaches, such as Ontop, achieve handling of spatial-semantic queries by developing a semantic connector on top of traditional spatial relational databases, such as PostGIS, resulting in highly efficient in handling geospatial data and queries. Once a new query (in GeoSPARQL) arrives, Ontop-spatial will be able to translate the GeoSPARQL query into a SQL query and conduct the spatial queries using the database engines, and the returned results will then be encapsulated into an RDF format.

Our work performs a systematic analysis of these approaches to provide the knowledge graph community an important reference for selecting semantic storage platforms for knowledge graph applications. We evaluated their performance based on simple and complex spatial-

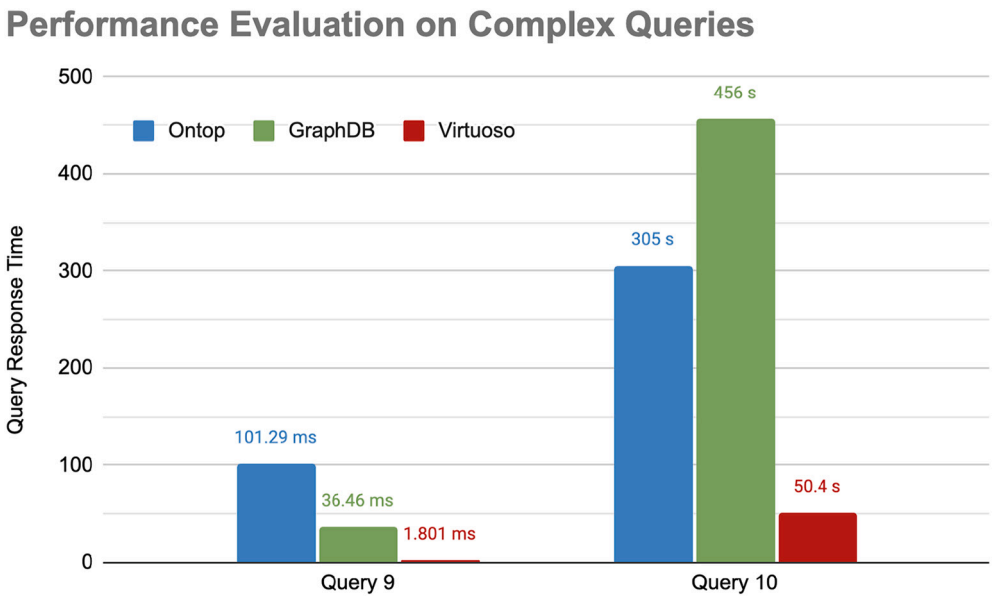



**Fig. 9.** Performance of semantic data repositories on complex queries.

semantic queries to simulate real-world scenarios. Our results show that storage-wise, Ontop is the most efficient solution, because data can be stored in its original and binary formats in relational databases, which are very efficient at managing such structured spatial data. There is no need with Ontop for converting structured data (spatial and non-spatial) into the graph format to fit into the storage requirement of a triple store. Query-wise, Virtuoso performs the best among all solutions in terms of both non-spatial and spatial semantic queries, followed by Ontop. They both perform much better than the other triple store solutions, including the commercial solution GraphDB, whose performance in handling spatial queries is similar to that of RDF4j and Neo4j. Fuseki's query performance is overall the most worrisome, due partially to its suboptimal memory management deficiencies.

Although Virtuoso is a type of an RDF store, it has a storage method sitting in between traditional triple stores and the OBDA approach. Instead of using a native graph storage method, Virtuoso builds its storage based on a relational database. But it has a more coupled solution between the query engine and the database than the OBDA approach, making it a very efficient solution. However, similar to other triple stores, such as RDF4j and GraphDB, in Virtuoso, structured data have to be remodeled into a graph format (such as RDF). In scientific applications, where huge amounts of observation or simulation data are available, this will introduce significant extra effort for preprocessing and data conversion in order to make the data available in a graph format and queryable in a knowledge graph. These converted data will also inevitably consume a huge amount of (duplicated) storage space, as illustrated in our storage consumption experiment (Fig. 4). Additionally, keeping the data in two formats (original and graph) will also substantially increase maintenance cost. In real-world applications where big data are being collected in real-time or where there is a frequent update in the data, the semantic lifting of such data into the graph format may well impede downstream applications with real-time requirements. Also, when there is a schema change to the semantic model of the data, all the datasets will need to go through the data conversion process again.

In comparison, Ontop's semantic model has shown important advantages. First, its implementation respects the heterogeneity of spatial data in terms of formats and their native data structures. Ontop allows geospatial data to remain in their native formats, and the data conversion exists only at the scheme/data model level. This way, the platform benefits from the advanced capabilities of existing spatial databases in efficiently handling spatial queries, and it can also reduce graph maintenance costs by keeping data in their original formats. The updates in semantic models also require the changes to be made only in the mapping table between the database-scheme and the knowledge-graph-scheme without touching the actual data. What is more, keeping data in their well-accepted formats is not only natural for scientists and data analysts, but we can also encourage adoption by the scientific community of knowledge graph technology for building a large, open knowledge network. From this end, more attention could be paid to the OBDA approaches for accelerating the accessibility and availability of geospatial scientific data into a global knowledge graph. Experimental results of the current research show that there is still room for Ontop to be improved in its query translation and efficiency. More optimization could be applied during the semantic translation to make Ontop more efficient while maintaining the robustness of the system.

As a new form of semantic data storage, Neo4j also shows relatively good performance in handling spatial queries. An advantage of Neo4j's design is that it separates the spatial data layer from the non-spatial layer and manages them separately. Its concept of property graph also allows the modeling of complex semantics. In practical applications, however, Neo4j requires the use of a new query language, Cypher, which may introduce additional overhead in query processing. If query optimization can be properly improved, it could also become a good candidate for both spatial and non-spatial knowledge graph applications.

As spatially explicit knowledge graphs are being increasingly investigated, spatial enablement in existing semantic data repositories will become critical and essential in supporting geospatial applications, ranging from environmental to urban science domains. For instance, when a disaster occurs, such as a hurricane or an earthquake, the relief experts need to quickly gather first-hand information in the disaster impacted area, including physical infrastructure and its damage, social-economic conditions, power outage information, health profile as well as information about local and national disaster (response) experts. These datasets are geospatial in nature and they are typically from multiple sources and are encoded in diverse formats. It is therefore a critical and challenging task for collecting such datasets and made them available in an analysis-ready manner. Building a spatially enabled knowledge graph that supports efficient queries and information retrieval from semantic repositories of cross-domain big data is the key for addressing the above challenge (Janowicz et al., 2022; Li, 2020). Our research, through a comprehensive performance evaluation of the functionality, storage and query efficiency of popular semantic data repositories of different types, provides new insights on choosing proper storage solutions to support real-world environmental and urban applications empowered by knowledge graph.

In the future, more work is needed in the spatially enabled knowledge graph research, particularly in building a community for experimenting and promoting new solutions to foster the adoption of knowledge graph technology in spatial data management, interoperability, and accessibility. Instead of following the one-size-fits-all solution that requires the conversion of data of any type to the standard RDF formats, the approach that provides the balanced solution between unity and diversity—such as semantic connector supported by Ontop, which can preserve the native formats and diversity of the data but also utilize the advanced concepts of Semantic Web for heterogeneous data management, integration, and semantic query—may become the optimal solution.

## CRediT authorship contribution statement

**Wenwen Li:** Conceptualization, Methodology, Formal analysis, Visualization, Validation, Writing – original draft, Supervision, Project administration, Funding acquisition. **Sizhe Wang:** Software, Investigation, Formal analysis, Visualization, Validation, Data curation, Writing – original draft. **Sheng Wu:** Methodology, Investigation, Formal analysis, Data curation, Visualization, Writing – original draft. **Zhining Gu**: Experiment. **Yuanyuan Tian**: Experiment.

## Acknowledgement

This work is supported in part by the National Science Foundation under awards 1455349, 2120943, 1853864, and 2230034.

## References

Alocci, D., Mariethoz, J., Horlacher, O., Bolleman, J. T., Campbell, M. P., & Lisacek, F. (2015). Property graph vs RDF triple store: A comparison on glycan substructure search. *PLoS One, 10*(12), Article e0144578. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144578.

Batory, D., Lofaso, B., & Smaragdakis, Y. (1998, June). JTS: Tools for implementing domain-specific languages. In *Proceedings. Fifth International Conference on Software Reuse (Cat. No. 98TB100203)* (pp. 143–153). IEEE.

Beckett, D. (2008). *Turtle-terse RDF triple language. http://www. ilrt. bris. ac. uk/discovery/2004/01/turtle/*.

Bereta, K., Xiao, G., & Koubarakis, M. (2019). Ontop-spatial: Ontop of geospatial databases. *Journal of Web Semantics, 58*, Article 100514.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34–43.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., … Xiao, G. (2017). Ontop: Answering SPARQL queries over relational databases. *Semantic Web, 8*(3), 471–487.

Can, O., Sezer, E., Bursa, O., Unalir, M., & O.. (2017). Comparing relational and ontological triple stores in healthcare domain. *Entropy, 19*(1), 30.

Cogrel, B., Kontchakov, R., Xiao, G., Hardi, J., Muro, M. R., Corman, J., … Bühmann, L. (2022). DB-Engines. In *System Properties Comparison Apache Jena - TDB vs. Blazegraph vs. GraphDB vs. Neo4j vs. RDF4J*. https://db-engines.com/en/system/Apache+Jena+-+TDB%3BBlazegraph%3BGraphDB%3BNeo4j%3BRDF4J (last accessed on May 24, 2022).

Ding, L., Xiao, G., Pano, A., Stadler, C., & Calvanese, D. (2021). Towards the next generation of the LinkedGeoData project using virtual knowledge graphs. *Journal of Web Semantics, 71*, Article 100662.

Esri. (2022). SSURGO downloader. https://www.arcgis.com/home/item.html?id=cdc49bd63ea54dd2977f3f2853e07fff (last accessed on May 24 2022).

Goodchild, M. F., & Li, W. (2021). Replication across space and time must be weak in the social and environmental sciences. *Proceedings of the National Academy of Sciences, 118*(35), Article e2015759118.

Güting, R. H. (1994, September). GraphDB: Modeling and querying graphs in databases. *VLDB, 94*, 12–15.

Huang, W., Raza, S. A., Mirzov, O., & Harrie, L. (2019). Assessment and benchmarking of spatially enabled RDF stores for the next generation of spatial data infrastructure. *ISPRS International Journal of Geo-Information, 8*(7), 310.

Ioannidis, T., Garbis, G., Kyzirakos, K., Bereta, K., & Koubarakis, M. (2021). Evaluating geospatial RDF stores using the benchmark Geographica 2. *Journal on Data Semantics, 10*(3), 189–228.

Janowicz, K., Hitzler, P., Li, W., Rehberger, D., Schildhauer, M., Zhu, R., … Currier, K. (2022). Know, know where, KnowWhereGraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Magazine, 43*(1), 30–39.

Jena, A. (2014). *Apache jena fuseki. 18*. The Apache Software Foundation. https://jena.apache.org/documentation/fuseki2/ (last accessed on March 1, 2022).

Jena, A. (2022). Apache Jena GeoSPARQL. https://jena.apache.org/documentation/geosparql/index.

Jovanovik, M., Homburg, T., & Spasić, M. (2021). A GeoSPARQL compliance benchmark. *ISPRS International Journal of Geo-Information, 10*(7), 487.

Kyzirakos, K., Karpathiotakis, M., & Koubarakis, M. (2012). Strabon: A semantic geospatial DBMS. In *International semantic web conference* (pp. 295–311). Springer.

Li, W. (2020). GeoAI: Where machine learning and big data converge in GIScience. *Journal of Spatial Information Science, 20*, 71–77.

Li, W., Batty, M., & Goodchild, M. F. (2020). Real-time GIS for smart cities. *International Journal of Geographical Information Science, 34*(2), 311–324.

Li, W., Raskin, R., & Goodchild, M. F. (2012). Semantic similarity measurement based on knowledge mining: An artificial neural net approach. *International Journal of Geographical Information Science, 26*(8), 1415–1435.

Li, W., Song, M., & Tian, Y. (2019). An ontology-driven cyberinfrastructure for intelligent spatiotemporal question answering and open knowledge discovery. *ISPRS International Journal of Geo-Information, 8*(11), 496.

Li, W., Zhou, X., & Wu, S. (2016). An integrated software framework to support semantic modeling and reasoning of spatiotemporal change of geographical objects: A use case of land use and land cover change study. *ISPRS International Journal of Geo-Information, 5*(10), 179.

Li, W. (2022). GeoAI in Social Science. In S. J. Rey, & J. Franklin (Eds.), *Handbook of Spatial Analysis in the Social Sciences*. Edward Elgar. In press.

Liu, Y., Liu, X., Gao, S., Gong, L., Kang, C., Zhi, Y., & Shi, L. (2015). Social sensing: A new approach to understanding our socioeconomic environments. *Annals of the Association of American Geographers, 105*(3), 512–530.

Mapcruzin. (2022). OpenStreetMap GIS data layers in the United States. https://mapcruzin.com/free-united-states-shapefiles/ (last accessed on May 24, 2022).

Miller, J. J. (2013, March). Graph database applications and concepts with Neo4j. In *, 2324. Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA* (p. 36).

Neo4j, S. (2017). Neo4j Spatial. http://neo4j-contrib.github.io/spatial/0.24-neo4j-3.1/index.html (last accessed on January 2, 2022).

Raza, A. (2019). *Comparison of geospatial support in RDF stores: Evaluation for ICOS carbon portal metadata. Master Thesis in Geographical Information Science*. Sweden: Lund University.

RDF4J. (2022a). The Eclipse RDF4J Framework. https://rdf4j.org/about/ (last accessed on January 2, 2022).

RDF4J. (2022b). RDF4j GeoSPARQL. https://rdf4j.org/documentation/programming/geosparql/#supported-geosparql-functions (last accessed on January 2, 2022).

Stadler, C., Lehmann, J., Höffner, K., & Auer, S. (2012). Linkedgeodata: A core for a web of spatial open data. *Semantic Web, 3*(4), 333–354.

Usery, E. L., Arundel, S. T., Shavers, E., Stanislawski, L., Thiem, P., & Varanka, D. (2022). GeoAI in the US geological survey for topographic mapping. *Transactions in GIS, 26*(1), 25–40.

Webber, J. (2012, October). A programmatic introduction to Neo4j. In *Proceedings of the 3rd annual conference on systems, programming, and applications* (pp. 217–218). Software for Humanity.

Wilson, J. P., Butler, K., Gao, S., Hu, Y., Li, W., & Wright, D. J. (2021). A five-star guide for achieving replicability and reproducibility when working with GIS software and algorithms. *Annals of the American Association of Geographers, 111*(5), 1311–1317.