NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
"INFORMATION AND DATA MANAGEMENT"

MASTER'S THESIS

Proposing a Methodology for Designing an Enterprise Knowledge Graph to Ensure Interoperability Between Heterogeneous Data Sources

Panagiotis I.

SUPERVISORS: **Manolis Koubarakis**, Professor
Sarra Ben Abbes, PhD
Lynda Temal, PhD

ATHENS

SEPTEMBER 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
“ΔΙΑΧΕΙΡΙΣΗ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΔΕΔΟΜΕΝΩΝ”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προτείνοντας μια Μεθοδολογία για το Σχεδιασμό Ενός Γνωσιακού Γράφου Επιχειρήσεων για να Διασφαλιστεί η Διαλειτουργικότητα Μεταξύ Ετερογενών Πηγών Δεδομένων

Παναγιώτης Η. Σούρσος

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανόλης Κουμπαράκης**, Καθηγητής
Sarra Ben Abbes, Διδάκτωρ
Lynda Temal, Διδάκτωρ

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2021

MASTER’S THESIS

Proposing a Methodology for Designing an Enterprise Knowledge Graph to Ensure Interoperability Between Heterogeneous Data Sources

Panagiotis I.
ID: M1558

September 2021

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προτείνοντας μια Μεθοδολογία για το Σχεδιασμό Ενός Γνωσιακού Γράφου Επιχειρήσεων για να Διασφαλιστεί η Διαλειτουργικότητα Μεταξύ Ετερογενών Πηγών Δεδομένων

Παναγιώτης Η. Σούρσος
Α.Μ.: Μ1558

Σεπτέμβριος 2021

ABSTRACT

The main research goal of this thesis is to propose an approach for designing an Enterprise Knowledge Graph (EKG) to ensure interoperability between different heterogeneous sources, taking into account the already existing efforts and automation processes developed by ENGIE in their attempt to build a domain-based EKG. Reaching this goal, demands a deep understanding of the already existing state-of-the-art on EKG approaches, their technologies with a focus on data-transformation and query methods and finally a comparative presentation of any new findings in this new challenge of defining an end-to-end formula for EKG construction. The criteria of evaluating the different works have been decided in a way to cover the following questions. (i) Which are the Implications and practical expectations of different design strategies to realize an EKG? (ii) How do those strategies affect semantic complexity and decrease or increase performance? (iii) Is it possible to maintain low latency and permanent updates?

Furthermore, our work was limited to one use case defined by ENGIE to explore the open data of accident and the data of road map as a starting point experience in EKG construction. We shall experiment with data transformation from heterogenous data sources into a final unified RDF datastore ready to be used as the foundation of an EKG. After, we are going to present the technical challenges, the vocabulary and the methods used to achieve a solution to the EKG definition problem.

Finally, a side goal of the thesis is to practically test and compare technical methods for data integration, enrichment and transformation. Most importantly we are going to test the ability to query Geospatial information which is a key element for this domain-based EKG. In this work, we are presenting the different implementations of RDF stores which support a Geographic Query Language for RDF Data (GeoSPARQL), a W3C standard for geo-related and Semantic representation of geographical data. Furthermore, we have formed our test data as a subset coming from ENGIE's big data. Our test data have been put and benchmarked against the knowledge transformation and linkage phases, using state-of-the-art Semantic tools.

SUBJECT AREA: Knowledge Graphs, Semantic Web, Knowledge Representation

KEYWORDS: Enterprise Knowledge Graphs, Geospatial Data, Knowledge Bases, Ontologies

ΠΕΡΙΛΗΨΗ

Ο κύριος ερευνητικός στόχος αυτής της διπλωματικής εργασίας είναι να προτείνει μια προσέγγιση για το σχεδιασμό ενός Γνωσιακού Γράφου Επιχειρήσεων (EKG) για τη διασφάλιση της διαλειτουργικότητας μεταξύ διαφορετικών ετερογενών πηγών, λαμβάνοντας υπόψη τις ήδη υπάρχουσες προσπάθειες και διαδικασίες αυτοματισμού που αναπτύχθηκαν από την ENGIE στην προσπάθειά τους να δημιουργήσουν ένα Γνωσιακό Γράφο ειδικού σκοπού. Για την επίτευξη αυτού του στόχου, απαιτείται η βαθιά κατανόηση των ήδη υπαρχόντων σύγχρονων προσεγγίσεων EKG, των τεχνολογιών τους με έμφαση στη μετατροπή δεδομένων και στις μεθόδους επερώτησης και τέλος η συγκριτική παρουσίαση τυχόν νέων ευρημάτων σε αυτή τη νέα πρόκληση καθορισμού ενός EKG. Τα κριτήρια αξιολόγησης των διαφόρων διαδικασιών έχουν αποφασιστεί με τρόπο που να καλύπτει τις ακόλουθες ερωτήσεις. (i) Ποιες είναι οι επιπτώσεις και τα πρακτικά αποτελέσματα των διαφορετικών στρατηγικών σχεδιασμού για τον ορισμό ενός EKG; (ii) Πώς αυτές οι στρατηγικές επηρεάζουν τη σημασιολογική πολυπλοκότητα και μειώνουν ή αυξάνουν την απόδοση; (iii) Είναι δυνατόν να διατηρηθεί χαμηλά η καθυστέρηση και να έχουμε μόνιμες ενημερώσεις;

Επιπλέον, η εργασία μας περιορίστηκε σε ένα σενάριο χρήσης που ορίστηκε από την ENGIE για να διερευνήσει τα ανοιχτά δεδομένα ατυχημάτων και τα δεδομένα του οδικού χάρτη ως μια αφετηρία στην κατασκευή ενός EKG. Θα πειραματιστούμε με τον μετασχηματισμό δεδομένων από ετερογενείς πηγές δεδομένων σε μία τελική ενιαία συλλογή δεδομένων (RDF) έτοιμη να χρησιμοποιηθεί ως τη βάση του Γνωσιακού Γράφου. Στη συνέχεια, θα παρουσιάσουμε τις τεχνικές προκλήσεις, το λεξιλόγιο και τις μεθόδους που χρησιμοποιούνται για την επίλυση του προβλήματος ορισμού ενός EKG.

Τέλος, ένας παράλληλος στόχος της διπλωματικής εργασίας είναι η πρακτική δοκιμή και σύγκριση τεχνικών μεθόδων για την ενσωμάτωση, τον εμπλουτισμό και τον μετασχηματισμό δεδομένων. Το σημαντικότερο για εμάς, είναι ότι θα δοκιμάσουμε την ικανότητα επερωτήσεων γεωχωρικών πληροφοριών που αποτελούν βασικό στοιχείο για αυτό το Γνωσιακό Γράφο ειδικού σκοπού. Σε αυτήν την εργασία, παρουσιάζουμε τις διαφορετικές υλοποιήσεις των RDF stores που υποστηρίζουν μια γλώσσα γεωγραφικών επερωτήσεων για δεδομένα RDF (GeoSPARQL), ένα πρότυπο του W3C για γεωγραφική και σημασιολογική αναπαράσταση γεωγραφικών δεδομένων. Επιπλέον, δημιουργήθηκαν δοκιμαστικά δεδομένα, τα οποία συγκρίνουν τον μετασχηματισμό και τη σύνδεση διαφορετικών δεδομένων, αξιοποιώντας τις αρχές του Σημασιολογικού Ιστού.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γνωσιακοί Γράφοι, Σημασιολογικός Ιστός, Αναπαράσταση Γνώσης

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Γνωσιακοί Γράφοι Επιχειρήσεων, Γεωχωρικά Δεδομένα, Γνωσιακές Βάσεις, Οντολογίες

“Data is a precious thing and will last longer than the systems themselves.” — Tim Berners-Lee, A Framework for Web Science (Foundations and Trends)

ACKNOWLEDGEMENTS

First of all, I would like to acknowledge my supervisor Prof. Manolis Koubarakis. His course Knowledge Technologies has directly contributed in discovering my personal interest in Artificial Intelligence, Semantic Web technologies and more specifically assisted me in selecting this topic. Furthermore, our long-lasting partnership during my undergraduate degree has been a great influence and an invaluable source of knowledge. Secondly, I would also like to acknowledge my PhD supervisors over ENGIE, Dr. Sarra Ben Abbes and Dr. Lynda Temal, whose research and previous studies shaped the background and the goal of this project. They have assisted and guided me with patience during the whole process of this thesis. I hope we join our forces again in the future and participate in their admirable work. Finally, I would like to thank Strabon's research team for their technical assistance and feedback.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

PREFACE

The present thesis is part of the requirements for the acquisition of a Master’s degree “Information and Data Management” in the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens.

1 CONTEXT AND CHALLENGES

World Wide Web has been designed in order to offer information understood by human beings. It consists by billions of documents, mainly HTML pages, interlinked via “lousy” hyperlinks [2]. Current architecture offers very limited opportunities for querying and retrieving information tasks by machines. Semantic Web provides a state-of-the-art framework that allows data to be shared and reused across application, enterprise, and community boundaries with great respect in machine automated processing. This framework has three crucial elements: the Resource Description Framework (RDF) data model, the Ontology Web Language (OWL) and SPARQL, the query language for RDF data sets.

The above-mentioned elements, are forming a framework which can help us with resource representation and at the same time, with the description of all possible connections between those same resources. Each resource could be assimilated to a node, and relations between these resources could be assimilated to an arc. These multi node data graphs alongside with properties for each node are known as Knowledge Graphs (KG). The user is enabled then to traverse the KG in order to gather information on resources represented by nodes and their properties. The construction of a Knowledge Graph is based on the reuse of some existing ontologies or on the definition of new concepts, if necessary, but designing a KG from scratch underlies many challenges.

As knowledge representation is gaining added value in the World Wide Web, enterprises have constantly shared an interest into creating their own KGs and benefit from them. Just having Big Data comes into nothing if those data are not efficient and enriched with metadata. An automated framework for transforming heterogenous data streams and databases into a unified data model, capable of extracting relationships, links and context has become the holy grail of the Knowledge Graph integration to modern enterprises. The systematic approach of automating end-to-end architectures having as input heterogenous data sources and as output well defined interlinked and interoperable KGs has actively become a domain of its own that we may call an Enterprise Knowledge Graph (EKG).

Building an EKG underlies many challenges. While a Knowledge Graph represents a collection of interlinked descriptions of entities, objects, events or concepts, an Enterprise Knowledge Graph uses a KG to identify all information that lies in disparate data sources throughout an organization on a domain-specific basis. First of all, a strong foundation with a complete and correct KG should exist in order to talk about other modules and processes that will transform any KG into an EKG. Knowledge coverage, architecture and entity extraction are some of the problems arousing when trying to create a KG in the scale of an enterprise. Regardless of the architectural differences between the different approaches, there appears to be common ground on the most important challenges in building an EKG. Those challenges include problems related to (i) the availability and scale of the KG, (ii) the knowledge extraction from heterogenous and unstructured sources (iii) the evaluation and evolution of the newly created representations. KG scaling has been consistent via manual approaches but as the demand for semantic representations increases inside big enterprises, we have to consider either semi-supervised or unsupervised knowledge extraction. This work aims to study the state-of-the-art both conceptual and practical approaches coming from both academia and industry worlds, learn about the challenges laying ahead and try to tackle down their biggest obstacles.

The rest of the thesis is structured in four chapters. Chapter 2 in its first part is dedicated into giving a general overview of the Semantic Web technologies and the tools that the reader will come across. The second part of the chapter is presenting the state-of-the-art on Enterprise Knowledge Graph approaches and finally, tries to classify them and at the same time distinguish the best practices in order to form a proposal for an EKG building framework. Across Chapter 3, we are addressing the challenges of ENGIE's heterogenous data sources and the effort to build ENGIE's EKG following our proposed EKG framework. Furthermore, experiments and tools are being investigated in technical terms for their performance in Chapter 4. The problem of transforming ENGIE datasets from heterogenous sources into a unified KG is presented as well. By the end of this chapter the reader has also the chance to guide through some use-cases arising from our transformed datasets. Finally, in Chapter 5, some general conclusions on the state-of-the-art and the creation of modern EKGs can be found alongside with our summary on the technologies used for this work and some possible matters that can be dealt with in future work.

2 STATE OF THE ART

2.1 Introduction

Having addressed the context and the challenges of this work above, this section will present the existing approaches, tools and the main foundation elements of Semantic Web technologies. Technical solutions are briefly presented for the readers to familiarize with, so they can follow the technical work of the latter chapters. Later on, we are focusing on the state-of-the-art on EKG approaches by classifying and comparing them in order to expose their strengths and weaknesses. Lastly, we attempt to propose a final well-defined methodology for EKG realization.

2.2 Background

2.2.1 RDF Data Models

A data model is defined as "the data items of a certain part of the perceived reality (business domain) relevant for a specific application or a specific user in a structured way. This model includes the data relationships" [3]. Several data models have been implemented for managing various data streams viable to big enterprises. The Resource Description Framework (RDF) is a framework for representing information in the web. The core structure of the abstract syntax is a set of triples, each consisting of a subject, a predicate and an object. A set of such triples could be assimilated to an RDF graph. An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link as depicted in Figure 1. RDF triples comply on RDF Schema, a semantic extension which supports classification and describes the interactions between resources. RDF is a data model that can express the "meaning" of individual pieces of information in a shared way. By dropping all heterogenous sources of information from our data models we are immediately dropping major problems. Problems like querying the same semantics represented in schemas or files or the need of converting of all possible representations of a fact into one statement. RDF provides us with a standard way of writing statements. Finally, we end up with a unifying model not only to itself but open to other data models as well. The model is immediate expandable by adding simple statements without re-designing the whole model and fast in terms of querying and mining information.

2.2.2 OWL Ontologies

The World Wide Web Consortium (W3C) Web Ontology Language (OWL) [1] is a Semantic Web language designed to represent rich and complex knowledge about things, groups of

[1] https://www.w3.org/OWL/.

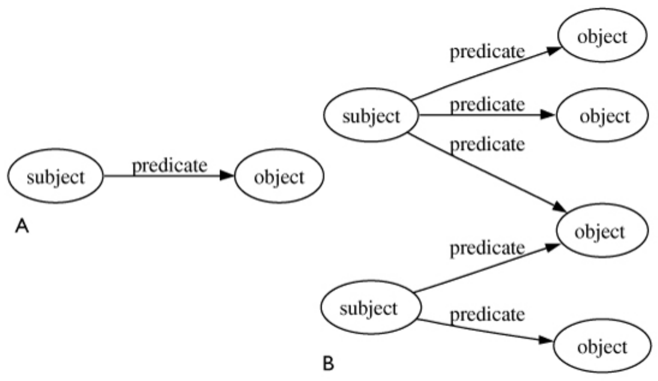


Figure 1: A. RDF Triple. B. Triples combine to form an RDF graph.

things, and relations between things. “OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit” [4]. World Wide Web is home OWL documents also known as ontologies. The same documents may be referred by or refer different ontologies. A group of technologies such as RDF, RDFS and SPARQL can be acknowledged as the building blocks of W3C’s Semantic Web technology stack along with OWL. The main role of OWL is to describe various data within the paradigm of the ontology, in that sense a set of individuals and a set of property assertions are deployed in order to relate all these elements together. An ontology is consisting of “classes”. This term, is used to describe a set of axioms placing constraints on sets of individuals and the permitted relationships between them. As an outcome, axioms are shaping the semantics of ontologies and enable systems to infer additional information by the provided data. In this work two OWL ontologies are utilized to establish a strong and efficient data model given all the advantages described above. Having established a machine understandable vocabulary, we may now introduce a way of forming queries against it.

2.2.3 SPARQL and GeoSPARQL

SPARQL [2] is an SQL-like query language having application in semantic databases. This RDF query language, aims to handle, fetch and manipulate data in the Resource Description Framework format. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middle-ware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. It also supports a broad set of constraining and value testing queries by source RDF graph.

The Open Geospatial Consortium (OGC) GeoSPARQL [3] standard supports representing

[2]https://en.wikipedia.org/wiki/SPARQL.
[3]https://www.ogc.org/standards/geosparql.

and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data. In addition, GeoSPARQL is designed to accommodate systems based on qualitative spatial reasoning and systems based on quantitative spatial computations. GeoSPARQL is consisting of (i) a set of SPARQL extension functions for spatial computations, (ii) its core vocabulary (RDFS/OWL) and (iii) a set of query rewrite rules. To fully perceive GeoSPARQL we fist need to discuss some basic geospatial concepts.

Features and geometries are two fundamental concepts of GeoSPARQL. A feature is simply any entity in the real world with some spatial location [5]. The location can either be precisely defined as a geometry or abstract. A geometry is any geometric shape, such as a point, polygon, or line, and is used as a representation of a feature's spatial location. Geometries vary in Coordinate Reference Systems (CRS) [5]. There are four parts that make up a CRS: a coordinate system, an ellipsoid, a datum, and a projection. The above mentioned concepts, are bound together forming the GeoSPARQL vocabulary, following the subject-predicate-object paradigm of RDF. The following (Figure 2) is a simplified diagram of the GeoSPARQL classes *Feature* and *Geometry* , as well as some of their properties.

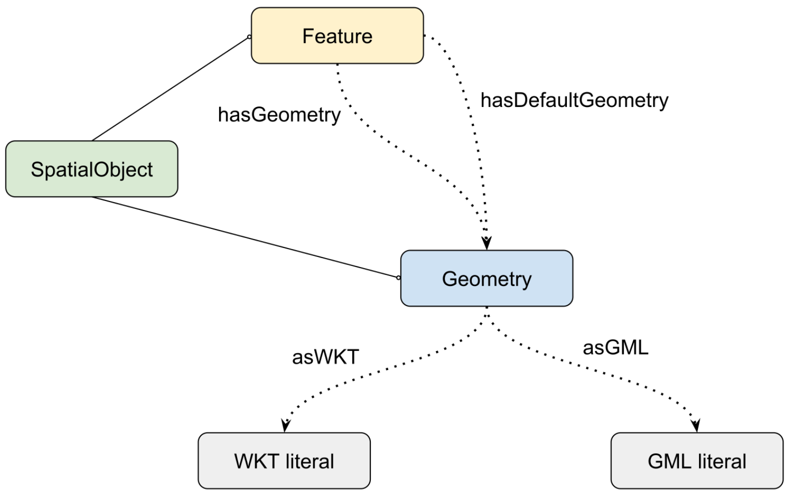


Figure 2: Diagram of the GeoSPARQL classes Feature and Geometry, as well as some of their properties.

2.2.4 Enterprise Knowledge Graphs

"A KG represents a collection of interlinked descriptions of entities, real-world objects and events, or abstract concepts (e.g., documents)" [6]. Descriptions have formal semantics that enable both humans and machines to process them efficiently and uniquely. Putting entities together, ultimately forms a network in which each entity is a fraction of a complete description of a concept and provides a solid and common context for its interpretation.

In recent years enterprises have been eager to manage different types of knowledge, hence the development of new Knowledge Management (KM) technologies. As the volume of incoming data is increasing in the scope of an enterprise, the importance of KM increases as well. Knowledge Management is developing new ways of reducing costs, increasing the performance and supporting an additional added value to company's products across different enterprise domains. Novel KM approaches often suggest new data organization architectures and foster their implementation in enterprises. One of such an architecture leverages semantic technologies, i.e., the technologies the Semantic Web is based on, in order to allow machines to understand the meaning of the data they work with. The concept of Linked Enterprise Data (LED) describes a framework to incorporate benefits of Semantic Web technologies into enterprise IT environments [7]. The initiative of modeling the embodiment of LED is known as an EKG, an EKG is consisting of all the properties, links, semantic network of concepts and is also accurately representing all core and fundamental knowledge relevant for an enterprise. EKGs can be portrayed as the gold standard between enterprise information and the next generation of data integration, offering a large scale data processing with robust Semantic technologies. Although EKGs have recently been seen an increase in interest, a formal conceptual framework for their design remains undeveloped. In order to address this deficiency, in the context of this work we revisit the concept and analyze the requirements for their realization.

2.3 Semantic and Data Transformation Tools

This section describes all the necessary tools that have been used and evaluated during the course of this work. Semantic Web has started to be enriched with geospatial data and geospatial extensions of SPARQL, like GeoSPARQL and stSPARQL [4].

GeoSPARQL extensions can be used in RDF graphs and SPARQL queries to represent and query geographic features with vector geometries. While those enhancements found a huge support by the semantic community, at the same time researchers needed to implement geospatial RDF stores that support these SPARQL extensions. Existing RDF stores like Sesame, RDF4J etc. have now became the foundation on which geospatial RDF stores are built upon [8]. By relying in state-of-the-art spatially-enabled RDBMS (e.g., PostGIS) for the storage and querying of geometries, GeoSPARQL RDF stores like Strabon, GraphDB and Apache Jena are presented below. Also, two data transformation tools (GeoTriples, SPARQL-Generate) which have an active contribution towards the implementation of our EKG are described as well.

[4] http://www.strabon.di.uoa.gr/files/stSPARQL_tutorial.pdf.

2.3.1 Strabon

Strabon [5] is a spatiotemporal RDF store. It is used to store linked geospatial data that change over time and pose queries using two popular extensions of SPARQL. Strabon supports spatial data types enabling the serialization of geometric objects in OGC standards Well-Known Text (WKT) and Geography Markup Language (GML). It also offers spatial and temporal selections, spatial and temporal joins, a rich set of spatial functions similar to those offered by geospatial relational database systems and support for multiple Coordinate Reference Systems. Strabon can be used to model temporal domains and concepts such as events, facts that change over time etc. through its support for valid time of triples, and a rich set of temporal functions. Strabon is built by extending the well-known RDF store Sesame (now called RDF4J) and extends RDF4J's components to manage thematic, spatial and temporal data that is stored in the backend RDBMS.

The first query language supported by Strabon is stSPARQL. stSPARQL can be used to query data represented in an extension of RDF called stRDF. stRDF and stSPARQL have been designed for representing and querying geospatial data that changes over time, e.g., the growth of a city over the years due to new developments can be represented and queried using the valid time dimension of stRDF and stSPARQL respectively. The expressive power of stSPARQL makes Strabon the only fully implemented RDF store with rich spatial and temporal functionalities available today.

Strabon also supports the querying of static geospatial data expressed in RDF using a subset of the recent OGC standard GeoSPARQL which consists of the core, geometry extension and geometry topology extension. Finally, Strabon is exposed using the RDF4J HTTP Server, users can manage the repository through the embedded Workbench, the RDF4J Workbench, or other tools integrated with RDF4J.

2.3.2 Ontotext GraphDB

Ontotext GraphDB [6] is a family of highly efficient, robust, and scalable RDF databases. It comes under both a free and an enterprise version by Ontotext. It streamlines the load and use of linked data cloud datasets, as well as your own resources. For easy use and compatibility with the industry standards, GraphDB implements the RDF4J framework interfaces, the W3C SPARQL Protocol specification, and supports all RDF serialization formats. The database version is the preferred choice of both small independent developers and big enterprise organizations because of its community and commercial support, as well as excellent enterprise features such as cluster support and integration with external high-performance search applications - Lucene, Solr, and Elasticsearch.

With GraphDB users can extract new semantic facts deriving from already existing ones. This is happening due to semantic inference enclosed within. GraphDB is taking advantage of the features coming along with RDF4J such as RDF model, query engines and

[5] http://strabon.di.uoa.gr/.
[6] https://graphdb.ontotext.com/documentation/standard/index.html.

RDF parsers. For that reason, the tool is packaged as a SAIL (Storage and Inference Layer). Inference is carried out by the Reasoner (TRREE engine), with explicit and derived instructions being stored in highly optimized data structures that are kept in-memory for query evaluation and subsequent inferences.

Implementing the Sail API interface, enables GraphDB to be integrated with the rest of the RDF4J framework, e.g., the query engines and the web UI. A user application can be designed to use GraphDB directly through the RDF4J SAIL API or via the higher-level functional interfaces. GraphDB like Strabon can also be exposed using the RDF4J HTTP Server and a Workbench.

2.3.3 Apache Jena

Apache Jena [7] is an open-source java framework for building semantic web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine along with a variety of storage strategies for RDF stores. On top of Jena there is the Apache Fuseki component. Fuseki [8] is a SPARQL server component that can use TDB as underlying persistent storage and also enable the access from multiple applications. TDB is a component of Jena for RDF storage and query. It supports the full range of Jena APIs. TDB can be used as a high-performance RDF store on a single machine. Fuseki can be launched as a standalone server, a web application or an operating system service and it exposes management interface for server monitoring and administration. The work package also contains Ontology API as well as Inference API to add custom semantics as well as inference and reasoning on the RDF data.

In Jena APIs, the Model class denotes an RDF graph and it contains the collection of RDF triples. It is an abstraction over different ways to store the graph like memory structures, disk-based persistent stores and inference engines etc. At lower levels, Jena uses another interface Graph for simpler abstraction and lower-level interaction. The required methods and interfaces to manage the RDF data can be acquired from the Model object for processing an RDF graph. Jena comes with a GeoSPARQL implementation as pure Java and does not require any set-up or configuration of any third-party relational databases and geospatial extensions while the WKT and GML serializations are supported.

2.3.4 GeoTriples

GeoTriples [9] is a tool for transforming geospatial data from their original formats (e.g., shapefiles or spatially-enabled relational databases) into RDF. The following input formats are supported: spatially-enabled relational databases (PostGIS and MonetDB), ESRI shapefiles and XML, GML, KML, JSON, GeoJSON and CSV documents. GeoTriples sup-

[7] https://jena.apache.org/documentation/geosparql/.
[8] https://jena.apache.org/documentation/fuseki2/.
[9] http://geotriples.di.uoa.gr/.

ports the mapping languages R2RML and RML and extends them for modeling the transformation of geospatial data into RDF graphs.

2.3.5 SPARQL-Generate

SPARQL-Generate [10] is an expressive template-based language to generate RDF streams or text streams from RDF datasets and document streams in arbitrary formats. It extends SPARQL, while it can be extended to support new data sources and formats. It also integrates seamlessly with existing standards for consuming Semantic Web data, such as SPARQL or Semantic Web programming frameworks. SPARQL-Generate can generate RDF streams or text streams from RDF, SQL, XML, JSON, CSV, GeoJSON, HTML, CBOR, plain text with regular expressions, large CSV documents, MQTT or WebSocket streams, repeated HTTP GET operations.

2.4 Comparative Analysis of Existing Approaches

Below we present a selection of the state-of-the-art approaches on EKGs. Knowledge graphs are extensively used for consistently representing real-world data. Large-scale, general purpose knowledge graphs, having millions of facts, have been constructed through automated techniques from publicly available datasets or sensitive in-house enterprise data. It is common for the KGs, to lack in completeness and often fail to correctly capture the semantics of the data [9]. This holds true particularly for domain-specific data, where the generic techniques for automated knowledge graph creation often fail due to several challenges, such as lack of training data, semantic ambiguities and absence of representative ontologies.

Additionally, we discuss methods, initiatives or even complete EKG solutions, which might be suitable as a backbone when designing and building and EKG from scratch. The fundamental particles of EKGs are tightly coupled with several heterogenous linked data sources instead of manually creating new ones from scratch. Automatic dataset fusion and interlinking thus becomes an important task. Several tools which are addressing the creation of an EKG are presented below, in an attempt to classify them based on their openness and availability. Three classes are emerging through this kind of classification, *Open-Source* EKG approaches, *Proprietary* EKG approaches and *In-House* ones. This distinction is very important to us, because as we can observe below, enterprises in their vast majority are in need of flexible technology stacks to integrate easily with existing routines, tools and pipelines an enterprise might already be utilizing. For any enterprise that joins the EKG paradigm it is certain that an existing architecture prior to semantic enforcement will be in place, therefore Open-Source adaptable approaches are more preferable over black-box Proprietary approaches. On the other hand, we can learn a lot by studying the In-House approaches developed by huge enterprises which are leading their domains.

[10] https://ci.mines-stetienne.fr/sparql-generate/.

2.4.1 Open-Source Approaches

Classifying an approach as Open-Source means that the proposed EKG solution has taken into consideration already existing open-source and free semantic tools. Such solutions, have directly dealt with real world problems and enterprise environments, enabling them to consider the importance of flexibility of their technology stacks and strategies for realizing EKGs. The whole concept of Semantic Web relies on the openness and sharing of Linked Data. It has also been the driving force for developing open and free tools that embody semantic concepts. The following works are offering coherent steps and open-source solutions for their technical discussion.

`Heaven Ape (HAPE)` [10] is an integrated big Knowledge Graph platform supporting the construction, management, and operation of large to massive scale KGs. In this approach, a major part of the DBpedia [11] knowledge base has been transformed and reconstructed it as a big Knowledge Graph. The architecture of HAPE is a platform consisting of three parts: the client side, which provides various kinds of services to the visitors and interacts directly with them, the server side, which provides all kinds of knowledge management and processing, which directly affect HAPE's third part the KG's knowledge base. More precisely, HAPE's client side is a browser operating on KG's knowledge base, while its server side is HAPE's operating system (OS), operating on the whole HAPE platform. Both of the browser and the OS are running according to their own scripts. The project is aiming to implement a general-purpose EKG platform which is built on existing open-source tools such as: PostgreSQL [12], Neo4j [13] and Apache Lucene [14].

`LDIF` [11] translates heterogeneous Linked Data from the Web into a clean, local target representation while keeping track of data provenance. It consists of a run-time environment and a set of pluggable modules. The run-time environment manages the data flows between the modules. The pluggable modules are organized in data access modules, data transformation modules and data output modules. LDIF provides access modules for replicating Web data locally via file download, crawling or SPARQL. These different types of import jobs all generate provenance metadata, which is passed on throughout the complete integration process. Import jobs are managed by a scheduler that can be configured to refresh the local cache hourly, daily or weekly for each source. LDIF also employs the R2R Framework to translate Web data that is represented using terms from different vocabularies into a single target vocabulary. Vocabulary mappings are expressed using the R2R Mapping Language. After those steps, the approach includes the *Sieve Data Quality Assessment* and *Data Fusion Framework* steps. At the end of the integration queue, LDIF outputs the cleansed data together with the provenance information in the form of a single N-Quads file. This file contains the translated versions of all graphs that have been gathered from the Web, the content of the provenance graph as well as the quality scores

[11] https://www.dbpedia.org/about/.
[12] https://www.postgresql.org/.
[13] https://neo4j.com/.
[14] https://lucene.apache.org/.

for all graphs. LDIF is distributed as a free [15] application providing both a single-node and a Hadoop version.

`MindLab` [12] is an industrial research project for building Knowledge Graphs to be consumed by conversational agents in domains like tourism. The approach starts with knowledge creation which describes the process of extracting information from different sources, structuring it, and managing established knowledge. This step is analyzed into three stages, (i) preparation for modeling, (ii) domain specification modeling and (iii) application of models. New knowledge is RDF-natured, stored in a graph database with respect to provenance, historical data and data duplication. After the data reside inside the *Knowledge Repository*, a special formalization is used to describe the newly created KG. *Knowledge Refinement* is consisting of three sub-categories that will ensure the knowledge enrichment, assessment and finally the cleanings services. A stable and version of the EKG can be finally made public to serve APIs, training ML models, SPARQL endpoints, internal services etc. MindLab [16] is built on-top of Semantify.it [17]. A free software as a service platform that helps in creating, validating, updating and publishing semantic metadata.

`Ontologies for Enterprise Knowledge Management (OKMS)` [13] is integrated enterprise knowledge management architecture for implementing an ontology-based knowledge management system. This EU-funded research project focuses on distributed ontology base knowledge management applications and is investigating how ontologies can improve traditional knowledge management systems. The work is introducing the terms *Global-As-View* (GAV) and *Local-As-View* (LAV). GAV, creates the integrated ontology as a view over individual sources. The drawback is that, for n sources, the integrated ontology might need to express n^2 interactions between source ontologies. LAV specifies each source as a query into the integrated ontology. In this case, writing only one query for each source ontology is sufficient. The first step (phase 1) of the approach, addresses bringing existing information to the ontology level. It extends the ontology-mapping problem somewhat to the problem of integrating existing information sources that are not ontology based. To help the user create more accurate ontology-mapping rules, the similarity extraction phase applies heuristic algorithms and machine learning techniques. A second phase (phase 2), is taking place dealing mostly with managing and evolving the created ontologies in terms of representation, propagation validation and discovery. The final and third phase (phase 3), refers to the framework including a comprehensive tool suite allowing easy ontology management and application. Key elements of the last phase are the evolution strategy and the query answering. OKMS is built on-top of the Karlsruhe ontology and Semantic Web framework [18] (KAON) which is an open-source ontology-management infrastructure targeted for semantics-driven business applications.

[15] http://ldif.wbsg.de/.
[16] https://mindlab.ai/en/.
[17] https://semantify.it/.
[18] https://www.aifb.kit.edu/web/Web_Science/en.

2.4.2 Proprietary Approaches

Big enterprises are complex environments, they are consisting of many departments and their data are synthesized as the combination of many heterogenous data sources (either internal or external). A successful EKG framework must tackle down the openness problem. The cost would skyrocket if an enterprise had to rethink its whole infrastructure in order to develop an EKG. In our opinion the openness of modern semantic technologies, has been the main reason in the past that big enterprises did not follow the semantic paradigm and unfortunately all proprietary approaches we studied in the context of this thesis are offering black-box solutions that are expensive to maintain and opposite to the Semantic Web's vision.

`Capsenta Ultrawrap` [14] is an Open Database Access (ODBA) platform that ties together the ontology, mappings and queries of an enterprise. R2RML mappings are being used in order to link the meaning of the business conceptualizations with data. Capsenta is naming their approach as a "Pay-as-you-go" method. There are three actors involved throughout the process: (i) *Business Users* are subject matter experts who can identify the list of prioritized business questions, understand the business rules associated with the data and validate the integrity of the created data. (ii) *IT Developers* understand database schemas, including how the data are interconnected. (iii) *Knowledge Scientists* serve as the communication bridge between Business Users and IT Developers. The Knowledge Scientist works with the Business User to understand the business questions, define an "whiteboard" version of the ontology and work with the IT Developer to determine which data is needed. This is documented in a knowledge report. Users must understand and clarify the business questions, and Users must identify the data necessary to answer those business questions. Later on, the Knowledge Scientist implements the ontology, mappings and queries based on the content from the knowledge report. That is why the goal of the knowledge implementation phase is to formalize the content of the knowledge report into an OWL ontology, R2RML mappings, SPARQL queries and subsequently validate the data. At the very end, the Business User is exposed to the data in a simplified and easy to understand view enabling straightforward data access with common Business Intelligence (BI) tools. Ultrawrap has been acquired by data.world[19] and it is offed as a paid EKG solution.

`Semantic Web Company PoolParty` [20] is a software that supports enterprises in knowledge management, data analytics and content organization. This approach in order to outline the architecture of the Knowledge Graph does not start with the description of the technical infrastructure for an EKG. The infrastructure for the KG is provided as a semantic middleware for the system architecture of the company. Structured data like relational databases, Excel and other spreadsheets, XML, etc. can be transformed into RDF as outlined in the chapter "RDFization: Transforming Structured Data into RDF" [15]. Standards like R2RML are used to connect relational databases, but traditional methodologies like XSLT are also of use here. Again, the key is to make it easy to set up those connections and provide

[19]https://data.world/.

[20]https://www.poolparty.biz/.

services that allow us to do so. Unstructured data in file systems etc., has to be made available in the simplest case to be sent for tagging (enrichment) or to be broken down into structured data by making unstructured data structured using the document structure as an outline. Some of the main KG enrichment services included in the process of EKG realization form this work include, relation and fact extraction, concept-based tagging and entity linking. This paid approach is service oriented with different services in the layers of consuming data and orchestrating existing Extract, Transform, Load (ETL) frameworks.

2.4.3 In-House Approaches

Many technology leaders like Google, Uber and LinkedIn have developed their KGs in order to enhance their search engine's results with information gathered from a variety of sources. Although some of those solutions like Google's Knowledge Graph [21] have been around for a few years and inspired new EKG approaches, at the same time their implementation details are not open. These approaches are kept within the premises of their enterprises and we are only offered high-level technical details on their implementations. Studying the big industry's In-House EKG solutions, and combining them with open-source semantic tools is something we are trying to achieve by presenting them within this work.

`Microsoft's Satori Graph` [22] is a graph-based repository that comes out of Microsoft Research's Trinity graph database and computing platform. It uses the Resource Description Framework and the SPARQL query language, and it was designed to handle billions of RDF triples (or entities). Its methodology towards the realization of an EKG, is consisting of four novel steps, (i) *Data Ingestion*, referring to the selection of data sources, data preparation and targeted fact extraction by NLP, (ii) *Match and Merge*, consisting of matching entity contents, detection of matched entities and scaling, (iii) *Knowledge Refinement*, for knowledge fusion, error detection and fact inference, and finally (iv) *Publish and Serve*, including all the modules and APIs for the final user.

`Thomson Reuters' Knowledge Graph` [16]. In this paragraph we are going to discuss the initiative of building and querying Thomson Reuters' knowledge graph. Data in heterogeneous formats is first acquired from various sources. Named entity recognition, relation extraction and entity linking techniques for mining information from the data and integrating the mined data across different sources. Modeling and storing data in RDF triples aim to create a unified KG named "TR-Discover" that enables users to search for information with natural language questions. Data acquisition on this framework also allows for manual data entry, as well as web scraping, feed consumption, bulk upload and OCR. Named Entity Recognition (NER) is also offered by extracting only known entities, without discovering unfamiliar ones. The core of relation extraction is a machine learning classifier that predicts the probability of a possible relationship for a given pair of identified entities in a given sentence. Finally, entity linking service links entities to nodes in the KG, primarily based on matching the attribute values of the nodes in the graph and that of a new entity.

[21] https://en.wikipedia.org/wiki/Google_Knowledge_Graph.
[22] https://kdd2018tutorialt39.azurewebsites.net/.

`Magic Mirror` [17], is an approach aiming to build KGs by exploiting semantic technologies to reconcile the data from diverse sources incrementally. A national-wide enterprise KG which incorporates information about 40.000.000 enterprises in China. First step is the to manually design or extend the schema of the EKG since the schema is subject change when new data sources are added. This is achieved through a D2R transformation, namely writing a customized mapping file in D2RQ to map fields related to atomic entity tables and atomic relation tables into RDF format. Information extractions adopt a multi-strategy learning method to extract multiple types of data from various data sources, until all information from different sources is fused into the EKG. Finally, usage scenarios are being cross validated to evaluate the completeness of the EKG.

2.5 Discussion

In this section we will evaluate the approaches presented in section 2.4 and we will focus on the most important and efficient elements of those initiatives. For that purpose, two comparative tables have been created gathering all the distinct characteristics and similarities across all solutions, in order to help us distinguish based on the nature of one's enterprise which solution fits better their purpose.

In Table 1 we have isolated a few important characteristics each EKG framework should be examined for. For each work (column 1) we are presenting a list of characteristics and how the presented approaches scored for each one of them. The Class (column 2) describes the aforementioned classification under which our approaches have been grouped in the previous section. This classification is quite important in understanding the limitations of each work and their compliance with the Semantic Web vision. Open-Source (column 3) is a binary field stating yes for open-source solutions and no for those who are not. Heterogenous Data (column 4) is again a binary field describing whether each work can support the fusion of heterogenous data sources, a significant characteristic for a modern EKG. Number of Sources (column 5) can have the values High, for a high number of incoming sources for the framework to process and Small, for a small number of incoming sources, usually less than 3. Flexibility (column 6) refers to the ability of each work to present us all the tools upon which they are built. A High value for flexibility means that a solution can be implemented with many alternative tools other than the ones already used. A Medium value for flexibility means that the alternative tools to address a problem are existing but over a few choices. Finally, a Small value in the flexibility field means that the presented work can only be implemented with its initial technology stack. Type of Data (column 7) with values Internal, External and Both, describes the type of data that each work is described to handle. Internal data are the data which already exist and produced by an enterprise's internal needs and operations. External data are the data which enter an enterprise via Web or files. Many enterprises are making the distinction between Internal and External data as Internal ones are subject to privacy restrictions, thus this feature is important.

Throughout all the presented works, we came across some numerous and different steps

Table 1: Comparison between the main characteristics of EKG approaches

Work	Class	Open-Source	Heterogenous Data	No. of Sources	Flexibility	Type of Data
HAPE	Open-Source	YES	YES (URL, XML/CSV/Text, RDF, KB)	High	Small	Both
LDIF	Open-Source	YES	NO	Small	Small	External
MindLab	Open-Source	YES	YES (JSON, URL, RDF, DB)	High	High	Both
OKMS	Open-Source	YES	NO	Small	Small	Internal
Ultrawrap	Proprietary	NO	NO	Small	High	Internal
PoolParty	Proprietary	NO	YES (JSON, URL, RDF, DB, XML/CSV/Text)	High	Small	Both
Satori Graph	In-House	NO	YES (JSON, URL, RDF, DB, XML/CSV/Text, Image, PDF)	High	Small	Both
T.Reuteurs KG	In-House	NO	YES (XML/CSV/Text, RDF, PDF, DB)	High	High	Both
Magic Mirror	In-House	NO	NO	Small	Medium	Both

in the creation process of an EKG. In our attempt to evaluate those steps we have created Table 2, a comparative table listing only a vital sub-set of those. Each approach had its own naming for similar routines, thus we tried to categorize and distinguish them into the following fields. Design Method (column 2) describes the driving force each enterprise is designing their EKG for. Some enterprises are likely to design a KG driven by their data, while others by the business questions of their departments or their clients. Mapping Method (column 3) lists the main mapping tools for each work. Mapping is one of the first services to be implemented when an enterprise is trying to transform data from heterogenous sources. There are several mapping methods in order to transform heterogenous data into RDF triples. The rest of the table consists of binary fields (yes for implementing and no for not), each describing a milestone step an EKG framework should implement. Knowledge Enrichment (column 4) refers to all services e.g., NER, NLP which are enhancing data transformation and which are adding value in the later produced RDF triples. Knowledge Cleaning (column 5) tries to prevent our triples from having duplicate entries and/or references to entities which overload the performance of the systems when joining data. Error Detection (column 6) is responsible for the final integrity or the RDF triples. It tries to solve problems like syntactic errors or events when there are semantically wrong assertions etc. Iterative (column 7) field describes whether each work can be developed in iterations or not. Many enterprises wish having iterative thus easily repairable frameworks. Data Security (column 8) gives us information about the security of data within an enterprise. Each enterprise can have sensitive client data that need to be processed, so a successful EKG framework should be able to operate with different data security levels. Finally, Machine Learning Methods (column 9) describes whether each work is supported by machine learning methods and algorithms to automate or semi-automate several routines from each step.

Table 2: Comparison between the main steps in the creation process of an EKG framework

Work	Design Method	Mapping Method	K. Enrichment	K. Cleaning	Error Detection	Iterative	Data Security	ML Methods
HAPE	Data Driven	Custom	YES	YES	YES	NO	YES	NO
LDIF	Data Driven	R2R	NO	YES	NO	NO	NO	NO
MindLab	Any	Any	YES	YES	YES	YES	YES	YES
OKMS	Ontologies	APIs	NO	YES	YES	YES	NO	YES
Ultrawrap	Business Questions	R2RML	NO	NO	NO	YES	NO	NO
PoolParty	Business Questions	ETL/Visual tools	YES	YES	YES	YES	YES	YES
Satori Graph	Data Driven	Declarative Mapping	YES	YES	YES	NO	NO	YES
T.Reuteurs KG	Data Driven	NLP	YES	YES	YES	YES	NO	YES
Magic Mirror	Data Driven	D2RQ	NO	YES	NO	YES	NO	YES

After consulting both tables above, let us now discuss and encapsulate in a few sentences what each presented work stood out for.

A first drawback of HAPE would be the fact that introduces a new vocabulary for the subject-predicate-object standard drifting away from the standards of W3C. As we have already mentioned, this approach is classified within the open-source approaches making it approachable. At the same time, HAPE requires a vast number of changes into an existing Enterprise KG workflow and the development of new custom JavaScript and Python scripts with limited support.

LDIF has an evident flaw, it has very limited mapping abilities over Relational Databases. As we know, all enterprises are heavily relying on their RDBMS systems and have been storing their data into relational databases for decades.

OKMS approach is found to be very ontology oriented, drifting away from the big picture of creating an EKG. Again, in this work as in the other works of the same class, we are having some proposed open-source tools upon which we can build our EKG.

As a critique to Capsenta's effort, we can acknowledge that the ”Pay-as-you-go” Methodology depends on having a Knowledge Scientist(s). It can also sometimes take a while to get the ball rolling until business users can see the first understandable results. A great issue with this work is the amount of manual work and iterations could be seen as inefficient and expensive. On the up side, Although Capsenta's product is proprietary, the approach mostly includes concepts rather than specific tools, so each enterprise is free to use their own technology stack following the same concepts.

MindLap is a really flexible and modular architecture. For each step presented, there are various technologies and tools proposed, as a result it can be immediately integrated into an existing workflow. Also, each step of the methodology has a distinct purpose and a broad distribution over different sub-routines for data ingestion, enrichment and correction.

Poolparty follows, with well-defined steps are the bottom line of this work. As Poolparty is a paid solution, the offered services are like a black-box to end-users. Those services would have to be rethinked and mapped to existing open-source or in-house solutions. For those who are willing pay the price of a paid solution, Poolparty offers a great plug-and-play highly flexible approach.

We continue with Microsoft's Satori Graph and the In-House classified approaches, and here we can distinguish the lack of implementation details which is a bottleneck for most enterprises but we can keep the clean and distinct steps of the approach.

The great advantage of Thomson Reuter's KG, is the in-depth analysis of the implementation and the technical details of the services provided over real-world datasets for evaluation. State-of-the-art machine learning methods are enforced as well, leading to a semi-supervised framework. Unfortunately, the lack of some concrete steps in comparison to other approaches must be mentioned, e.g., lack of EKG cleaning, updating and correction services.

Lastly, Magic Mirror is a service-oriented approach, therefore easily adapted by any enterprise as it is offering flexibility. Throughout this approach, each technical step has plenty of information on the technology stack used and makes a great example on how we can benefit from a solution that is not publicly available.

2.6 Conclusions

If the creation process of modern EKGs can educate us about one thing, this would be that each enterprise and each knowledge-domain is different, what can be a vital step for one enterprise can decrease the performance or increase the cost of another. Overall, a concrete recipe cannot be followed across all enterprise. Concluding we are trying to gather all the important information highlighted by the comparison in section 2.5 and break down the similarities and differences of each proposed methodology. Having in mind our classification for the state-of-art between Open-Source, Proprietary and In-House approaches, we are attempting to propose a general approach that respects the adaptability and technical flexibility and at the same time encapsulates the most important steps in a concrete EKG creation framework.

At first, we have to define which are the building blocks of our EKG framework. We cannot consider an EKG without addressing all the services evolving around data of course. As "EKG Framework" we define the technology stack, services, ontologies, taxonomies, data sources etc. from which the final EKG will be created.

Data alone cannot realize a KG, the first step is to transform them. The majority of the state-of-the-art have dedicated steps for data handling. Whether it is heterogenous sources, internal or external data, they all have to be transformed into unified RDF triples. We shall name this step *Data Ingestion*. This step can be implemented either by ETL tools, or even Deep Learning methods like NER, but it is always recommended to be supervised by a knowledge scientist. We can further analyze and breakdown the sub-steps consisting the ingestion process. It can be split in three stages as follows:

Data Ingestion: describes the process of extracting information from different sources, structuring it, and managing established knowledge.

First Stage (Preparation for Modeling):

1. Analysis of a domain's entities and their (online) representation
2. Defining a vocabulary (potentially by restricting and/or extending an already existing vocabulary)
3. Mapping to semantic vocabularies
4. Annotation of data
5. Evaluation and analysis of annotations

Second Stage (Domain Specification Modeling): Reflects the results of Stage 1 and formalizes the findings in a (i) unified (ii) exchangeable (iii) machine-readable and understandable way.

Third Stage: Applies models for further data Ingestion, either by creating mapping according to the domain specifications or by annotating services. These services can be supervised or semi-supervised.

After acquiring data from heterogenous sources, transforming them and having created our mappings, it is time to store the new shape of those data. Semantic technologies are drifting away from traditional RDBMS systems, utilizing their own triple-based graph stores as we already have seen in the introductory chapters. Many open-source solution exist here that can play the role of an RDF-store, well known in the community of Semantic Web, e.g. RDF4J [23] , Neo4j, hence the third step in our proposal is the following:

Knowledge Repository: RDF-nature, storage in graph database with respect to:

- Provenance
- Historical data

After the data reside inside the Knowledge Repository, we may decide to refine them, correct them and run sanity tests against them using SPARQL queries. *Knowledge Refinement* will be the lead-step, while it will be consisting of three sub-categories that will ensure the: *Knowledge Evaluation*, *Knowledge Cleaning* and finally, *Knowledge Enrichment*. Knowledge Evaluation as the word states will be responsible for the correctness and completeness of our KG. Some evaluation indicators would be the following: (i) Accessibility, (ii) Accuracy, (iii) Correctness, (iv) Completeness, (v) Cost-effectiveness, (vi) Interoperability, (vii) Relevancy and (viii) Variety. Knowledge Cleaning can breakdown into three sub-routines as well:

Knowledge Cleaning: All the actions taken to improve the accuracy of KGs. Main sources and types or Errors:

1. Instance Errors
   i) Syntactic errors in the instance identifiers
   ii) Type does not exist in the vocabulary
2. Property Value Errors
   i) Property does not exist in the vocabulary
   ii) Domain and range violations
   iii) Semantically wrong assertion
3. Equality Assertion Errors
   i) Syntactic errors
   ii) Semantically wrong assertion
4. Class Violations
   i) When an individual can be mapped to more than one different classes

[23] https://rdf4j.org/.

ii) When individuals a, b, c are mapped in R(a,b), R(a,c) where R is functional.

First Stage (Error Detection): Identify errors from different error sources

Second Stage (Error Correction): Correct the identified errors manually or semi-automatically

Knowledge Enrichment: Provides the processes for improving the completeness of a knowledge graph by adding new statements. It is consisting of four main sub-services:

1. Knowledge source detection
    i) Open source - may be automated to some extent
    ii) Proprietary sources - usually very hard to automate
2. Knowledge source integration (Integrations should be done via standard service interfaces so that all your developers and data engineers can understand and easily work with it within the automation loop of the knowledge graph life cycle)
3. Duplicate Detection (Having duplicate statements can dangerously increase the complexity space of calculations as well as the Cartesian products while joining facts from different domains)
4. Property-Value-Statements correction
    i) Addition of missing instance assertions
    ii) Addition of missing equality assertions
    iii) Addition or deletion of property value assertions

Finally, ***Knowledge Deployment*** will be a stable version of the EKG can be made public to serve APIs, training ML models, SPARQL endpoints, internal services etc.

Concluding, we believe that we have successfully proposed a subset of key-steps and functions for the realization of a modern EKG. Inspired by the state-of-the-art we have presented some vital steps in the EKG construction process. These steps are a product of our evaluation criteria and readers are encouraged to further investigate similar approaches and solutions. In the next chapter we are going to investigate the problems and challenges of implementing some of these steps in ENGIE’s effort to create its own Enterprise Knowledge Graph.

3 BUILDING ENGIE'S ENTERPRISE KNOWLEDGE GRAPH

3.1 Introduction

Every enterprise is organizing information, as a network of clearly identifiable entities with valuable connections between them. An EKG aims to replicate this network with most of its relevant entities into a graph database. In Chapter 2, we tried to evaluate and discuss about the state-of-the-art approaches and we also attempted to propose a general framework consisting of key steps for realizing an EKG. Moreover, the EKG construction process might use data analysis technologies, machine learning and semantic methods to enhance the data quality and transform unstructured data into knowledge. Therefore, this chapter focuses on analyzing the problems and challenges emerging mainly during the Data Ingestion and Knowledge Evaluation steps while building ENGIE's EKG. More specifically we are addressing the data curation and correction services as an attempt to put to the test some of the steps defined in the previous chapter. At the end of this chapter we are also discussing about the criteria of Knowledge Repository step.

3.2 Problems and Challenges of Heterogenous Sources During Data Ingestion Step

In this section we are trying to present the problems and challenges as well as our effort towards combining two different heterogenous data sources. First, let us talk about our domain. ENGIE is trying to create an EKG consisting of various traffic accident information connected with OpenStreetMap shapefiles of Metropolitan France. Traffic accident data are coming in the form of CSV files while OpenStreetMap files in the form of shapefiles (shp). For both datasets we are presenting and benchmarking the methods we have used in order to transform them into semantic enabled turtle (ttl) format and unify them under two main ontologies developed by ENGIE. The first ontology (Accident Ontology) encapsulates all the metadata and semantic information for the accidents while the road ontology (City Ontology) those referring to the routes, street characteristics etc. The connecting link between those to ontologies is the class “Thoroughfare”. A thoroughfare is a transportation route connecting one location to another. Parts of Accident and City ontologies can be seen in Figure 3 and Figure 4 respectively.

3.2.1 Description of ENGIE's Heterogenous Data Sources

During this section we are testing the steps of Data Ingestion and more specifically the preparation for modeling phase and the Mapping to semantic vocabularies, such us our ontologies. For that purpose, we are presenting more details on our two data sources from OpenStreetMap and the French Road Safety Observatory.

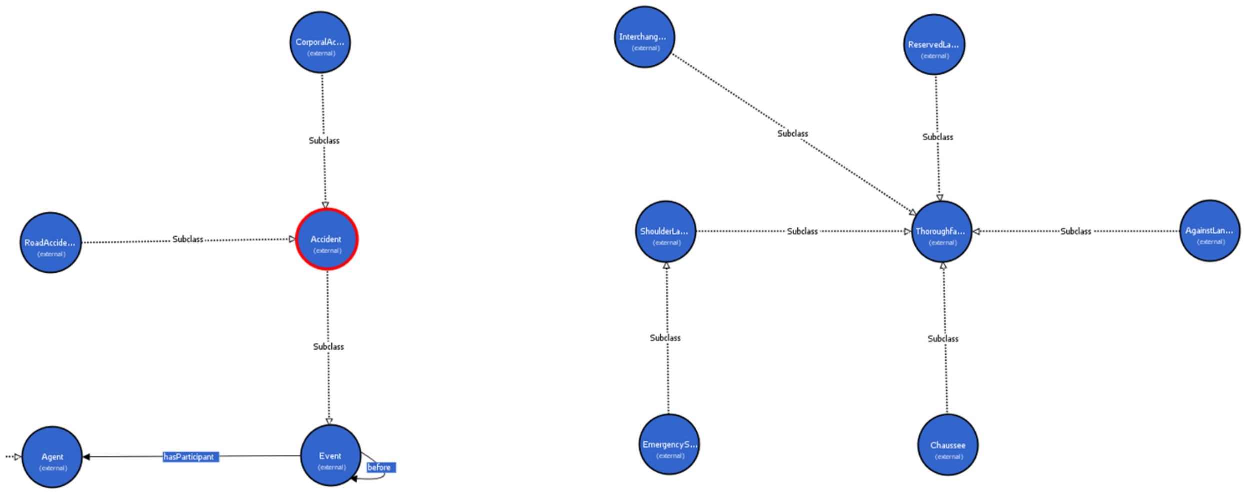


Figure 3: Visualization of Accident Ontology's main classes

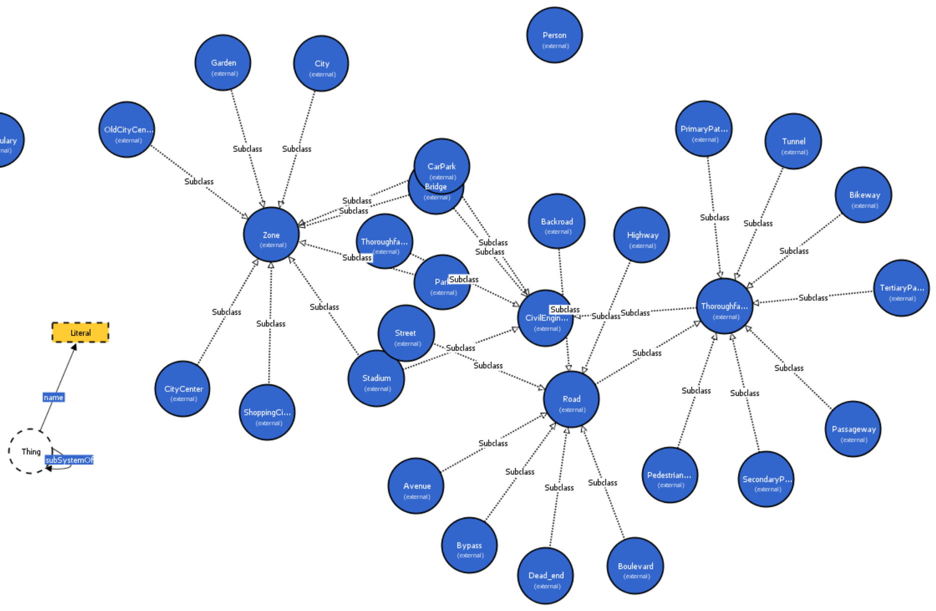


Figure 4: Visualization of City Ontology

OpenStreetMap (OSM) [1] is a collaborative project to create a free editable map of the world. The geodata underlying the map is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability

[1] https://en.wikipedia.org/wiki/OpenStreetMap.

of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.

Our road data, are coming directly from the French branch of OSM [2]. We have used shapefile data from 21 regions of France. SPARQL's CONSTRUCT queries were later used to transform OSM data into turtle files although the same task can be successfully achieved by other methods and tools such as Geotriples.

The French Road Safety Observatory (ONISR) [3] leads the collection and consolidation of accident data and publishes road safety analyses. For each bodily injury (i.e., an accident on a road open to public traffic, involving at least one vehicle and having caused at least one victim requiring treatment), information is entered describing the accident by the unity of the police (police, gendarmerie, etc.) who intervened at the scene of the accident. These incidents are gathered in a sheet entitled analysis bulletin of bodily accidents. All of these files constitute the national file of traffic accidents known as the "BAAC file" administered by ONISR.

The Etalab database of traffic accident data for a given year is divided into 4 headings in the form for each of them of a file in CSV format.

1. The section CHARACTERISTICS which describes the general circumstances of the accident.
2. The section PLACES which describes the main location of the accident even if it took place at a intersection.
3. The section of VEHICLES involved.
4. The section of USERS involved.

Each of the variables contained in an item must be able to be linked to the variables of the other items. The accident identifier number (Cf. "Num_Acc") present in these 4 sections makes it possible to establish a link between all the variables which describe an accident. When an accident involves several vehicles, it is also necessary to be able to link each vehicle to its occupants. This link is made by the id_vehicule variable.

Now that we have familiarized with out data sources, we can proceed with the description of the problems residing within the data. Data Ingestion does not stop in the process of extracting information from different sources, but also structuring it and managing the established knowledge. Furthermore, mapping the newly fetched data as a first step before their transformation is also a goal of this specific step and of the EKG build process in general.

[2] https://www.openstreetmap.fr/donnees/.

[3] https://www.data.gouv.fr/fr/datasets/bases-de-donnees-annuelles-des-accidents-corporels-de-la-circulation-routiere-annees-de-2005-a-2019/.

3.2.2 Challenges During Preparation for Modeling Phase

Data Transformation (DT) from our two main heterogenous sources (CSV, shapefile) is part of the Data Ingestion step and it has been the main technical challenge of this work. Because of the huge importance of DT services, we have dedicated Chapter 4 for that purpose. For the rest of this section, we are addressing to problems occurring after their transformation in RDF triples. The preparation for modeling phase along with knowledge evaluation, is examined below, while we are presenting the methods used to solve two different challenges.

The main challenge of our research data has been the different representations on geospatial information. On one side we had a 2-dimensional geospatial representation for accident locations using the WGS84 Geo Positioning RDF Vocabulary [4], which is a vocabulary for representing latitude, longitude and altitude information in the WGS84 geodetic reference datum. And on the other side we had each region's Line String route in order to be mapped in our ontology's class "Thoroughfare". On land a thoroughfare may refer to anything from a multi-lane highway with grade separated junctions, to a rough trail. Thoroughfares used by a variety of traffic, such as cars on roads and highways. Line string data represent the spatial coverage of the associated data object. The spatial coverage of the data object is the trajectory consisting of many points enclosed together forming a line on a map. For each data object having a spatial coverage, there is an RDF statement that connects it to the spatial coverage object with the predicate ogc:hasGeometry. The spatial coverage object is then associated to its geographic literal with the predicate ogc:asWKT (see Figure 2). Below, the different architectures and some sample data extracted from our datasets are being presented. Listing 1 gives us a sample of the transformed accident location points, originally in CSV format . Listing 2 gives us a sample of the transformed road data, originally in shapefile format.

```
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

<http://engie.com/accident/201600007071/location/point>
    a           geo:Point ;
    geo:lat     "41.9420066"^^xsd:double ;
    geo:long    "8.7570108"^^xsd:double .
```

Listing 1: Accident's location point in WGS84 CRS

```
@prefix ogc: <http://www.opengis.net/ont/geosparql#> .

<http://www.engie.fr/roads/gis_osm_roads_free_1/corse/thoroughfaresection
   /1916/geometry>
    a           ogc:Geometry ;
    ogc:asWKT   "<http://www.opengis.net/def/crs/EPSG/0/4326> LINESTRING
   (9.1258871 42.5615069, 9.1267367 42.5613208, 9.1268807 42.5612707,
   9.1269598 42.5611714)"^^ogc:wktLiteral;
```

Listing 2: Road's location point in OGC CRS

[4] https://www.w3.org/2003/01/geo/wgs84_pos.

But let us consider the problems arising from having two different vocabularies to describe geo-locations. To fully take advantage of GeoSPARQL and perform in depth queries combining accident points with road Line Strings, we had to make use of the topological query functions [5] provided such as *geof:sfContains* and *geof:sfInterects*. Those functions only accept as variables WKT or GML literals and OGC defined geometries. Our first attempt was to transform the WGS84 coordinates into OGC Points dynamically while performing the query. This query is part of the Knowledge Evaluation step while building an EKG. An example of such a query can be seen in Listing 3. Such a solution was not found to be functional because the query engine evaluates the newly created OGC literal as a plain string literal. Unable to be given as input into a topological function.

```
SELECT *
WHERE {
    ?thorouhfare a seas:ThoroughfareSection.
    ?thorouhfare geo:hasGeometry ?geo.
    ?geo geo:asWKT ?asWKT .

    ?accident a eao:RoadAccident.
    ?accident seas:location ?location.
    ?location lgdgeo:point ?point.
    ?point lgdgeo:lat ?lat.
    ?point lgdgeo:long ?long.

    BIND(CONCAT("\"POINT(",STR(?lat)," ",STR(?long),")\"^^geo:wktLiteral") AS ?label)
    FILTER (geof:sfIntersects(?label, ?asWKT))
}
```

Listing 3: SPARQL query to dynamically transform accident's point from WGS84 CRS to OGC CRS

After that attempt, we decided to modify our accident data and create on top of the WGS84 representations an additional OGC point representation. These additions have been made upon the transformed data that produced turtle files with automated Java scripts. In Listing 4 we can see both representations residing in our accident dataset after the Java script has run.

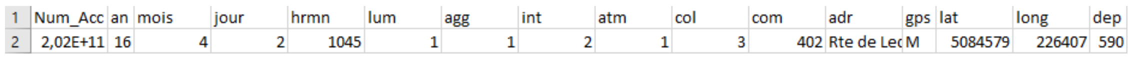
```
<http://engie.com/accident/201600007071/location/point>
    a            geo:Point ;
    geo:lat      "41.9420066"^^xsd:double ;
    geo:long     "8.7570108"^^xsd:double .

<http://engie.com/accident/201600007071/location/geometry> a ogc:Geometry ;
    ogc:asWKT "<http://www.opengis.net/def/crs/EPSG/0/4326> POINT (8.7570108 41.9420066)"^^ogc:wktLiteral
```

Listing 4: Final form of accident dataset having borth CRS representations

Before we even came across the different CRS representation problem, we had to find a solution for a smaller but yet fundamental one. Our CSV data had a quality problem. The coordinates given in latitude and longitude fields within the files we are missing the

[5] http://defs.opengis.net/vocprez/object?uri=http://www.opengis.net/def/function/geosparql/.

precision represented by the dot symbol (.). In this case the segments describing lat and long coordinates that we met above, had an initial form of plain numbers e.g. 41.9420066 came in the form of 419420066. This problem not only affects the quality of our knowledge but it is a blocking issue since any geospatial query cannot be functional for the given coordinates. In Figure 5 we can see a row of our input CSV file which provides all the essential characteristics of an accident along with lat and long information. These files have been our input for the transformation.



1	Num_Acc	an	mois	jour	hrmn	lum	agg	int	atm	col	com	adr	gps	lat	long	dep
2	2,02E+11	16	4	2	1045	1	1	2	1	3	402	Rte de Lec	M	5084579	226407	590

Figure 5: Sample row from accident characteristics CSV input file

We addressed this issue by using a semantic tool. SPARQL-Generate which is a basic tool utilized by ENGIE for various data transformations was selected for this task because of its added value to the company. Below in Listing 5 we are able to see a sample SPARQL-Generate query that produces a Point triple with corrected lat and long information.

```
BASE <http://engie.com/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
PREFIX fun: <http://w3id.org/sparql-generate/fn/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX seas: <https://w3id.org/seas/>
PREFIX sch: <https://schema.org>
PREFIX fibo: <https://spec.edmcouncil.org/fibo/ontology/FBC/>
PREFIX ocds: <http://purl.org/onto-ocds/ocds#>
PREFIX engdisc: <https://nglab.fr/>
PREFIX geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX seas:  <https://w3id.org/seas/>
PREFIX eao:   <http://www.engie.fr/ontologies/accidentontology/>

GENERATE {
    <http://engie.com/accident/{?account}/location/point>
        a          geo:Point ;
        geo:lat    ?lat ;
        geo:long   ?long .
    }

    ITERATOR iter:CSV(<http://engie.com/caracteristiques_2016.csv>, true, "\""
    , ";", "\n", "Num_Acc", "lat", "long")
    AS ?account ?latstr ?longstr
    WHERE {
        BIND( if(STRLEN(?latstr) >= 7, CONCAT (substr(?latstr, 1,2),".",substr
    (?latstr, 3),"^^xsd:double"), ?empty) as ?lat)
        BIND( if(STRLEN(?longstr) >= 3, "{?longstr}"^^xsd:double, ?empty) as ?
    long)
    }
}
```

Listing 5: Sample of SPARQL-Generate query for lat and long precision correction

Having addressed the two major problems with our heterogenous data sources, we have managed to test both programming and semantic methods to overcome them. The next step is the Knowledge Repository (KR) and refers to the knowledge storing and availability through an RDF-store. The next session addresses the challenges of KR.

3.3 Problems and Challenges During Knowledge Repository Step

As we have established, the Knowledge Repository step is part of our general EKG build method. Among other operations, it includes the process of storing the unified data in a graph database. In order to store our research data into a unified RDF graph, we had to think about which RDF-stores are suitable for the task. The RDF stores to be evaluated for suitability of this data are required to be capable to handle large datasets and also offer the geospatial support conforming to an established standard (GeoSPARQL). There are quite many RDF stores that could have been potential candidates for this study. Therefore, we have created a list of must-haves for the selected RDF stores that we have tried and compared. (i) The RDF store should be actively supported. (ii) The RDF store should support W3C standards like SPARQL 1.1, and also semantic reasoning, including ontological and rule-based reasoning. (iii) The RDF store should have advanced geospatial capacity and support GeoSPARQL. (iv) The RDF store should preferably be open-source. (Garbis G., et al. 2013). Based upon the criteria listed above, following RDF stores were selected for this evaluation:

- Strabon
- Ontotext GraphDB
- Apache Jena

Strabon and Jena are selected because these open-source products are commonly used as libraries or underlying framework component by a wide range of other RDF stores. A number of RDF databases and stores support Java APIs conforming to RDF4J (Strabon) and Jena interfaces. GraphDB offers a free edition and it provides a strong GeoSPARQL support.

3.4 Problems and Challenges During Knowledge Deployment Step

Knowledge Deployment refers to a stable version of the EKG, that can be made public to serve APIs, training ML models and SPARQL endpoints. It is the final step in the process of building an EKG and it comes after the step of Knowledge Repository. At the same time, it is the main source of feedback on whether the business questions and enterprise needs are met by our EKG. This particular step can lead the engineers add or remove services, automated processes and even re-design whole ontologies. In the case of ENGIE, the attempt to answer important business questions about accidents in the roads of Metropolitan France has brought up certain challenges.

Transformed shapefiles have provided us with a dataset consisting of road information and routes in the form of Linestring for 21 regions of France. On the other side, our transformed CSV files, lead to a dataset of accidents in the form of Points. The main business question to be answered by ENGIE's KG is the safety of the roads in Metropolitan France. A road is marked safe based on the number of accidents that we can count while traveling from point A to point B yet not all of our accidents happen exactly within our route. Some accidents which have effect on our arrival time, might have taken place on pavements near our route so we have to count them as well.

In order to achieve this, two main GeoSPARQL functions have been utilized. The first is *geof:distance* to measure the distance between an accident Point and our Linestring route. The second is *ogc:sfIntersects* which enables us to match accidents that occurred directly onto our route. *ogc:sfIntersects* matches two given geometries and returns true if exact matches of the whole geometries or parts of them intersect one another. The main difference and the need of using both functions is that *ogc:sfIntersects* is being used for precision that can answer different business questions and more direct, while *ogc:distance* allows us to be less precise and count accidents on a certain radius. The main topological relations between OGC spatial objects are visualized in Figure 6.

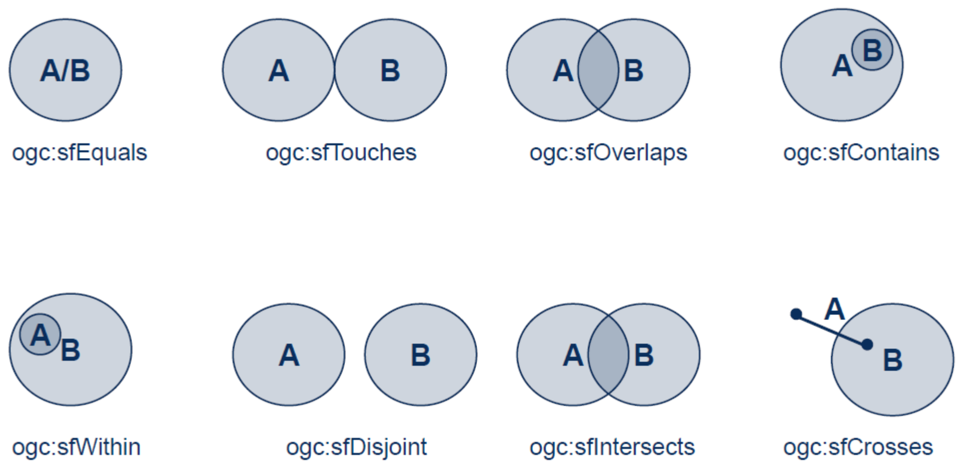


Figure 6: Topological Relations between ogc:SpatialObject [1]

3.5 Conslusions

In this chapter we have tested certain steps of our proposed EKG framework presented in section 2.6. Data Ingestion for the preparation for modeling phase and the evaluation of our data though SPARQL queries (Knowledge Evaluation) have been presented as well. We have also opened the discussion of RDF-stores and their selection criteria, to test them against our Knowledge Repository step. Furthermore, we have presented our approach on challenges that occurred during Knowledge Deployment and before we post sanity queries against our KG. What about the missing vital steps? Our effort to build ENGIE's EKG based on our proposed EKG framework resumes in Chapter 4. Steps such as knowledge transformation and knowledge mapping are going to be experimentally tested. For these tasks, we have selected our RDF-stores and other semantic tools for their parallel role as transformation and mapping tools.

4 EXPERIMENTAL RESULTS

4.1 Introduction

In the previous chapter, we have seen how to tackle specific problems occurring in the first phase of Data Ingestion. We have prepared our dataset and evaluated it so as to be stored into an RDF-store (Knowledge Repository step). But how did we end up having RDF triples from either CSV or shapefile input data? This process is part of Data Ingestion as well and more specifically, the transformation and mapping activities. In this chapter we are testing the aforementioned activities, presenting experimental results on their challenges and outcomes. Three RDf-stores and one semantic tool, are going to be tested and compared with each other. The comparison includes the transformation times as well as the storage times for our RDF-ready data. We are also presenting the outcome of a mapping method using Geotriples as an alternative to traditional CONSTRUCT queries.

4.2 Comparing Semantic Tools in the Context of Transformation and Mapping Activities

The software components and their underlying modules for each of the RDF-stores to manage the geospatial linked data are referenced and implemented from the official documentation of each of the RDF stores. After we studied all technical information for each tool, we tried to install and deploy a functional local environment for all of them. Both Strabon and GraphD are offering a web interface in order for the users to upload their datasets and query them. Apache Jena on the other hand, offers a command line environment in Java for the same purpose. Finally, Geotriples comes as a docker installation with a command line tool, or as a Java executable (JAR file). A few basic SPARQL queries were run after deployment, for testing each solution. Geospatial support/indexing was also enabled for validating the documented features.

In order to enable spatial indexes and search for Apache Jena you just have to reference in Jena's class configuration as follows: `GeoSPARQLConfig.setupSpatialIndex();`. For GraphDB a spatial index is provided by default but you need the statement depicted in Listing 6 to enable GeoSPARQL functions. Finally, for Strabon everything is in place out-of-the-box.

```
PREFIX : <http://www.ontotext.com/plugins/geosparql#>

INSERT DATA {
  _:s :enabled "true" .
}
```

Listing 6: GraphDB geospatial index activation query

4.2.1 Practical Experiments

All practical experiments conducted in the study, are focusing on performance metrics against storing the biggest stand-alone region source file available in our dataset (provence-alpes-cote-d-azur.ttl) of 1GB and afterwards, a CONSTRUCT query that will transform our road shapefile data into turtle formatted triples embedded in the vocabulary of our City Ontology. The tests have been hosted into a commodity hardware with a first-gen i7 Intel® 64 processor and 16GB of DDR3 RAM.

Strabon (v.3.3.2) was installed along with PostgreSQL (v.9.6) database and the bundled PostGIS (v.2.5) plugin to support geospatial indexes and GeoSPARQL, while for GraphDB (v.9.8.0) and Apache Jena (v.4.1.0) we used the out-of-the-box configuration of the local file system and memory respectively. A footnote for Strabon would be the PostgreSQL configuration which was consulted by Strabon's GitHub page [1]. For Geotriples (v.1.1.6) you may find installation and usage instructions in the tool's GitHub page [2]. The KR-suite version was used for our experiments.

Because the CONSTRUCT query was creating an infinite amount of Cartesian products, the split method was enforced, splitting the query into two sub-queries and then the two different output files were merged programmatically. For Strabon and GrapBD the queries run within the web interfaces provided by both tools. For Apache Jena a Java script had to be used in order to load the dataset and execute the two queries. In Listing 7 we can see one of the split CONSTRUCT queries used to transform our data.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX seas: <https://w3id.org/seas/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX iter: <http://w3id.org/sparql-generate/iter/>
PREFIX fn: <http://w3id.org/sparql-generate/fn/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX eao: <http://www.engie.fr/ontologies/accidentontology/>
PREFIX vocab:  <http://example.com/ontology#>
PREFIX cdt: <http://w3id.org/lindt/custom_datatypes#>

BASE <http://www.engie.fr/roads/gis_osm_roads_free_1/>
# You must update manually the variable ?region to the corresponding name of
    region

CONSTRUCT {

  ?engieGeoURI a ogc:Geometry ;
          ogc:asWKT ?wktLiteral ;
          ogc:coordinateDimension ?coordinateDimension ;
          ogc:dimension ?dimension;
          ogc:is3D ?is3D ;
```

[1] https://github.com/esarbanis/strabon.

[2] https://github.com/GiorgosMandi/KR-Suite-docker.

```
          ogc:isEmpty ?isEmpty ;
          ogc:isSimple ?isSimple ;
          ogc:spatialDimension ?spatialDimension .
}

WHERE {

  ?geometryURI ogc:asWKT ?wktLiteral ;
         ogc:coordinateDimension ?coordinateDimension ;
         ogc:dimension ?dimension;
         ogc:is3D ?is3D ;
         ogc:isEmpty ?isEmpty ;
         ogc:isSimple ?isSimple ;
         ogc:spatialDimension ?spatialDimension .

  BIND ( "provence-alpes-cote-d-azur" AS ?region)
  BIND (STRAfter(STR(?geometryURI), "/Geometry/") AS ?geoTripleGeometryID)
  BIND (URI(CONCAT(?region,"/thoroughfaresection/",STR(?geoTripleGeometryID),"
    /geometry")) AS ?engieGeoURI)

}
```

Listing 7: CONSTRUCT query for data transformation

In the case of Geotriples we simply followed the three steps mentioned in Listing 8 inside our Docker container hosting the service. Note that the single node version has been used (there is also a Spark version available for Big Data), and that we have selected the RAW file approach as a data source. The tools can also take input from data stored in a spatially enabled database.

```
// 1. Generate Mapping files (will create mapping.ttl inside /inout directory)
    .
./geotriples-mapping -o /inout/mapping.ttl -b http://engie.com /inout/rhone-
    alpes-latest-free.shp

// 2. Manually edit the mapping.ttl to fit your needs (sample follows)

@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix  rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@base <http://geotriples.eu/base> .
@prefix rrx: <http://www.w3.org/ns/r2rml-ext#>.
@prefix rrxf: <http://www.w3.org/ns/r2rml-ext/functions/def/>.
@prefix ogc: <http://www.opengis.net/ont/geosparql#>.
@prefix schema: <http://schema.org/>.
@prefix onto: <http://engie.com/ontology#>.

<#gis_osm_roads_free_1>
rml:logicalSource [
  rml:source "/inout/provence-alpes-cote-d-azur-shp/gis_osm_roads_free_1.shp";
  rml:referenceFormulation ql:SHP;
  rml:iterator "gis_osm_roads_free_1";
];
```

```
rr:subjectMap [
  rr:template "http://www.engie.fr/roads/gis_osm_roads_free_1/id/{GeoTriplesID
    }";
  rr:class onto:gis_osm_roads_free_1;
];
rr:predicateObjectMap [
  rr:predicateMap [ rr:constant onto:hasOsm_id ];
  rr:objectMap [
    rr:datatype  xsd:string;
    rml:reference "osm_id";
  ];
];
rr:predicateObjectMap [
  rr:predicateMap [ rr:constant onto:hasCode ];
  rr:objectMap [
    rr:datatype  xsd:integer;
    rml:reference "code";
  ];
];

...

].

// 3. Transform file into RDF (creates out.ttl transformed turtle dataset from
     the shapefile and rules described in the mapping.ttl).
./geotriples-rdf -o /inout/out.ttl -f TURTLE /inout/mapping.ttl

// Sample of generated triples

<http://www.engie.fr/roads/gis_osm_roads_free_1/id/208682>
    <http://www.engie.fr/ontology#hasBridge>   "F" ;
    <http://www.engie.fr/ontology#hasCode>     5142 ;
    <http://www.engie.fr/ontology#hasFclass>   "track" ;
    <http://www.engie.fr/ontology#hasLayer>    "0"^^xsd:long ;
    <http://www.engie.fr/ontology#hasMaxspeed> "0"^^xsd:integer ;
    <http://www.engie.fr/ontology#hasName>     "{name}" ;
    <http://www.engie.fr/ontology#hasOneway>   "B" ;
    <http://www.engie.fr/ontology#hasOsm_id>   "223566406 " ;
    <http://www.engie.fr/ontology#hasRef>      "" ;
    <http://www.engie.fr/ontology#hasTunnel>   "F" ;
    <http://www.opengis.net/ont/geosparql#hasGeometry>
        <http://www.engie.fr/roads/gis_osm_roads_free_1/Geometry/208682> .
```

Listing 8: Geotriples steps for generating a turtle file from a shapefile

Combining both methods for transforming our shapefile data, we are ready to present Table 3. Within this table we have attempted to measure the loading times (column 2) of loading provence-alpes-cote-d-azur.ttl (size of 1GB) for the semantic tools in question. After that, CONSTRUCT Query part1 field (columns 3) and CONSTRUCT Query part2 (column 4) are presenting the execution times of our CONSTRUCT queries using two split queries. Total Time (columns 5) includes the total loading and execution time for both query parts. Finally, Storage Type (column 6) lists the types of different storage methods our tools are using as each storage type has a direct effect on performance.

Table 3: Comparison between semantic tools for the knowledge transformation step

Tool	Loading Time	CONSTRUCT Query part1	CONSTRUCT Query part2	Total Time	Storage Type
Strabon	30 min.	1 hour	30 min.	**2 hours**	RDBMS
Ontotext GraphDB	6 min.	10 sec.	10 sec.	**6.20 min.**	Filesystem
Apache Jena	2 min.	2 min.	1 min.	**5 min.**	Memory
Geotriples	N/A	N/A	N/A	**15 sec.**	N/A

4.2.2 Discussing the Practical Experiments

The time has come to compare and discuss the results of our experiments. As it is noticeable, Strabon is by a huge margin the slowest solution for that kind of transformation. The results can be explained due to the relational nature of the data storage (PostgreSQL) and the evaluation of each statement. At the same time, GraphDB and Apache Jena are not bound to a relational database so that would explain the fast-loading times of our dataset, especially Apache Jena for this experiment stored the data into the memory. For Geotriples Loading Time is not applicable because the tool is streaming the shapefile dataset and working in real time with it.

On the query part again, Strabon is by far the slowest. GraphDB has exciting results, as it is the only solution to achieve the transformation in under a minute. This in our opinion has to do with the lack of evaluation and treating the statements as literals. Our need to transform our heterogenous data into turtle formatted triples, suggests a robust solution that will be functional no matter the size of the input files. Apache Jena's total time might seem ideal for the job, but we have to consider that if we had parallel incoming files or just huge big data inputs no commodity memory could handle the work-load. At this point one might say that the next best-performing solution is the right-one because it stores the file into the filesystem having no bottleneck on the memory side, while commodity hard drive space is cheap. GraphDB is indeed a great set of tools with complete GeoSPARQL support. Geotriples performance is ideal as its architecture allows it to open a connector to the data source and process the input data directly. This will only produce the R2RML mapping file which contains all the rules for the final transformation into an RDF format. The mapping is also enriched with subject and predicate object maps so that the RDF graph that will be produced follows the GeoSPARQL vocabulary [18]. The final transformation takes place in a Java efficient way and it is bound to the user's hardware only.

Strabon might seem to be one's last option for a similar transformation task but bear in mind that having a relational database storing your data, makes them highly available (you can connect to remote databases), safe and evaluated at all times. PostgreSQL along with PostGIS offer an out-of-the-box query optimizer, evaluator and great structure to the data. So, if the needs of a task are to just transform an amount of data, then Apache Jena would be the best choice, if the task is part of a bigger picture like in our case, as you will see in the following chapter, where the transformed data later will be queried making use of GeoSPARQL and inference a more robust solution should be considered.

Finally, as for using Geotriples, the only disadvantage would be the manual labor added to an expert's time, editing the automated generated mapping file, in order to shape the final RDF desired graph. One might argue that the same case exists in drafting the CONSTRUCT query for the transformation, but R2RML is still very limited in aggregate functions and string handling operations that come at ease with SPARQL.

4.3 Trying to Answer Business Questions During Knowledge Deployment

After transforming our research data into a unified RDF-dataset under the same RDF-store, it is time to put them to the test. Use-Cases as in queries were used in order to validate the connections between our data and embody the relationships described in our ontologies. This section aims to present some samples of how we translated business questions and business needs into practical queries. For these tests the queries were validated using Strabon because in our empirical experience has the best geospatial support compared to GraphDB and Jena in the contexts of this study.

We have realized that the ability of answering Business Questions is one of the main purposes of an Enterprise Knowledge Graph. Whole frameworks from Chapter 2, have been developed and built in order to answer enterprises questions. This is also the purpose of Semantic Web as a whole, an environment capable of capturing a representation of the real world, emphasizing in objects and their relations. After that, it is made possible by inference to deduct conclusions on hidden relationships the human eye and brain would fail to spot.

But put the capitalization of new knowledge aside, enterprises need to answer and answer quickly, questions regarding their daily business scenarios of their domains. Like in our case, ENGIE's most critical question is to find the safest route to travel from point A. POINT(8.7569314 41.9418509) to point B. POINT(8.762151 41.9433067).

Below we are demonstrating a use-case where two points have been selected as a starting point and ending point of our desired travel route. Our demo road dataset, consists of three routes (thoroughfare sections) in the form of Linsestring OGC geometries. Our demo accident dataset, consists of nine accidents in the form point OGC geometries. All geometries are located in the island of Corsica. The relation between those points follows and can also be depicted in Figure 7:

- For thoroughfare section with id 9072 we have one accident that is located exactly on the Linestring, one that differs only in precision of a few meters from the Linestring and one relatively close.
- For thoroughfare section with id 1916 we have one accident that is located exactly on the Linestring, one relatively close and one that is far away.
- For thoroughfare section with id 30216 we have all three accidents located far away from the Linsestring.

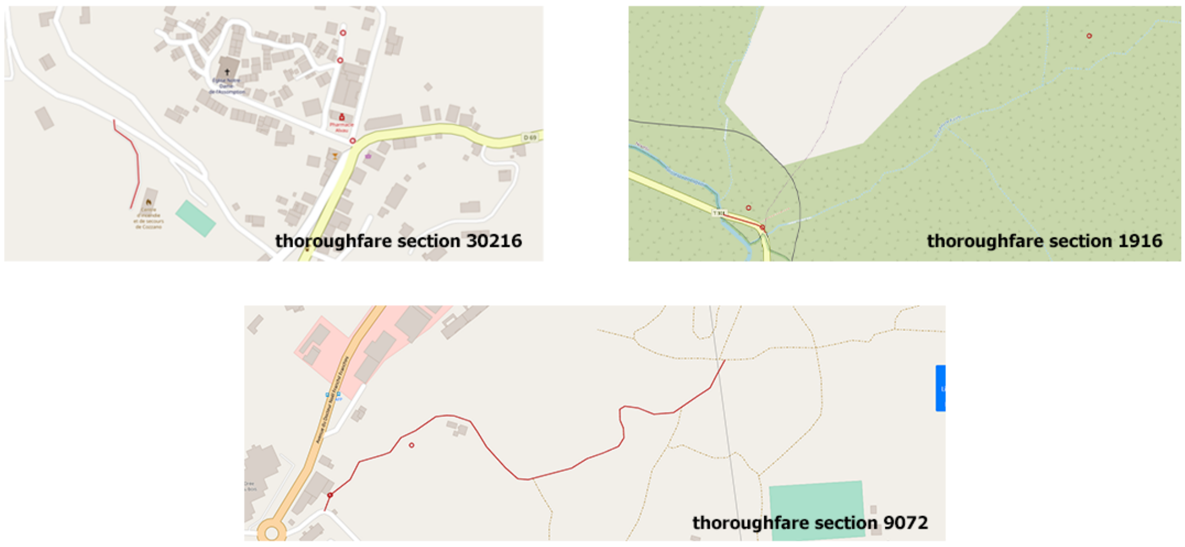
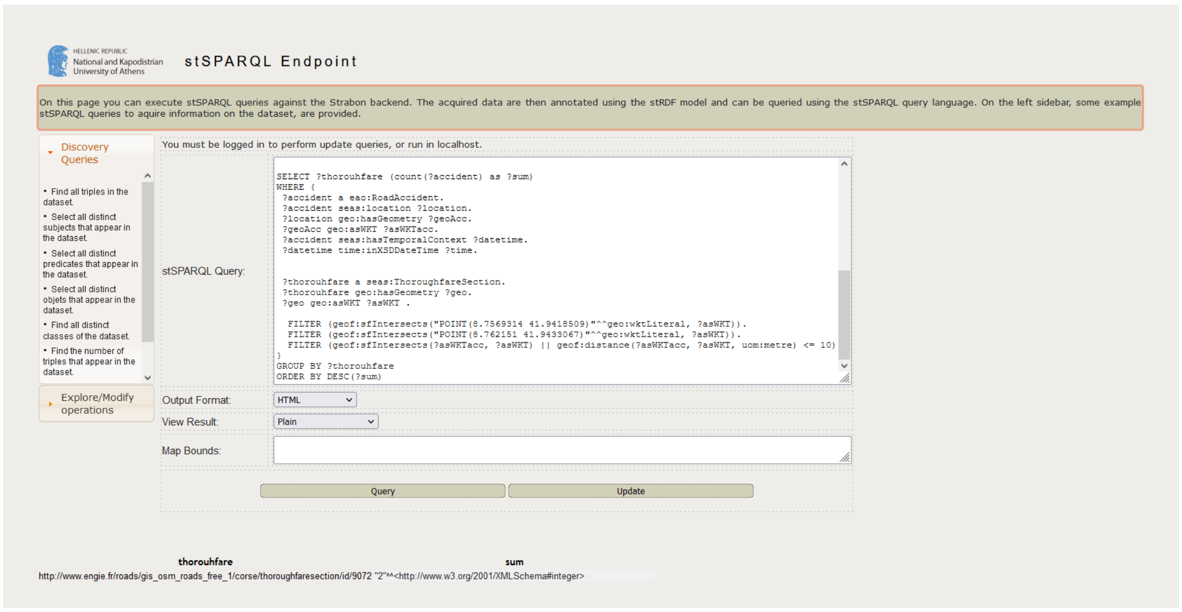

thoroughfare section 30216
thoroughfare section 1916
thoroughfare section 9072

Figure 7: All three thoroughfare sections and their association with accidents of demo dataset

Depending on the radius we are interested in, the criteria of the safest route from our point A to point B can change. In the example to follow, we have constructed a SPARQL query that returns the safest route to follow between our two starting and ending points. Our thoroughfare sections have to intersect with either of our points in order to be candidates for the safest road to follow. In our demo dataset only one thoroughfare section intersects with our point and this is the section with id 9072. As we can observe in Figure 8 the query results to thoroughfare section with id 9072 and a count of 2 accidents in that route. We are also noticing that the radius to count accidents has been set to 10 meters. That means that if there is an accident within a distance of 10 meters from our thoroughfare, we consider it to have an effect on our journey. During another test we have increased the distance to 100 meters and then, the safest route had 3 accidents occurring within our journey.

The main business question has been successfully answered during the Knowledge Deployment step. It seems that the actions of data transformation, mapping and evaluation during the Data Ingestion step described in Chapter 3, have been effective. Those steps, have resulted into a unified RDF-dataset stored in our RDF-store, completing that way the Knowledge Repository step as well. Our KG has been published in Strabon and it is now ready to answer ENGIE's business questions. But how can our EKG respond to bigger questions for data across all Metropolitan France?

Below, we are presenting the results of yet another business question posted across a full dataset for all 21 regions of France. The question concerns the top 10 Departments of France with the most accidents and their counts. The results can be found in Figure 9.


HELLENIC REPUBLIC
National and Kapodistrian University of Athens
stSPARQL Endpoint
Discovery Queries
Find all triples in the dataset.
Select all distinct subjects that appear in the dataset.
Select all distinct predicates that appear in the dataset.
Select all distinct objets that appear in the dataset.
Find all distinct classes of the dataset.
Find the number of triples that appear in the dataset.
Explore/Modify operations
Query
Update

On this page you can execute stSPARQL queries against the Strabon backend. The acquired data are then annotated using the stRDF model and can be queried using the stSPARQL query language. On the left sidebar, some example stSPARQL queries to aquire information on the dataset, are provided.

You must be logged in to perform update queries, or run in localhost.

stSPARQL Query:

```
SELECT ?thorouhfare (count(?accident) as ?sum)
WHERE {
 ?accident a eao:RoadAccident.
 ?accident seas:location ?location.
 ?location geo:hasGeometry ?geoAcc.
 ?geoAcc geo:asWKT ?asWKTacc.
 ?accident seas:hasTemporalContext ?datetime.
 ?datetime time:inXSDDateTime ?time.


 ?thorouhfare a seas:ThoroughfareSection.
 ?thorouhfare geo:hasGeometry ?geo.
 ?geo geo:asWKT ?asWKT .

  FILTER (geof:sfIntersects("POINT(8.7569314 41.9418509)"^^geo:wktLiteral, ?asWKT)).
  FILTER (geof:sfIntersects("POINT(8.762151 41.9433067)"^^geo:wktLiteral, ?asWKT)).
  FILTER (geof:sfIntersects(?asWKTacc, ?asWKT) || geof:distance(?asWKTacc, ?asWKT, uom:metre) <= 10)
}
GROUP BY ?thorouhfare
ORDER BY DESC(?sum)
```

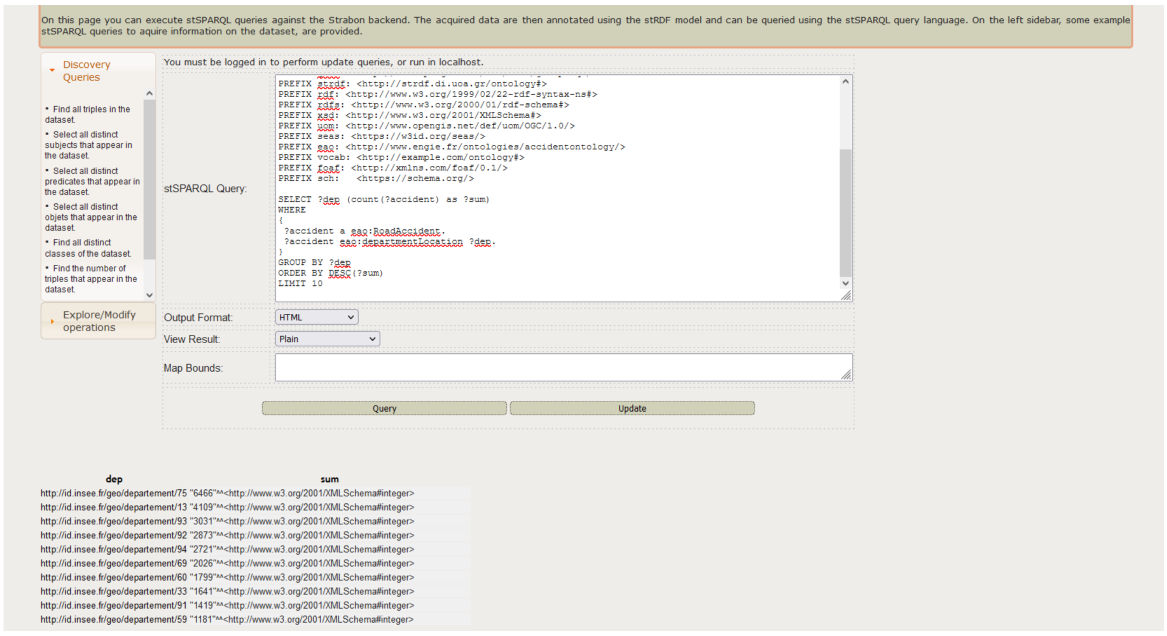
Output Format: HTML

View Result: Plain

Map Bounds:

thorouhfare	sum
http://www.engie.fr/roads/gis_osm_roads_free_1/corse/thoroughfaresection/id/9072	"2"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 8: Example of SPARQL query against Strabon workbench for the safest route from point A to point B, along with the count of accidents


Discovery Queries
Find all triples in the dataset.
Select all distinct subjects that appear in the dataset.
Select all distinct predicates that appear in the dataset.
Select all distinct objets that appear in the dataset.
Find all distinct classes of the dataset.
Find the number of triples that appear in the dataset.
Explore/Modify operations
Query
Update

On this page you can execute stSPARQL queries against the Strabon backend. The acquired data are then annotated using the stRDF model and can be queried using the stSPARQL query language. On the left sidebar, some example stSPARQL queries to aquire information on the dataset, are provided.

You must be logged in to perform update queries, or run in localhost.

stSPARQL Query:

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
PREFIX seas: <https://w3id.org/seas/>
PREFIX eao: <http://www.engie.fr/ontologies/accidentontology/>
PREFIX vocab: <http://example.com/ontology#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX sch:   <https://schema.org/>

SELECT ?dep (count(?accident) as ?sum)
WHERE
{
 ?accident a eao:RoadAccident.
 ?accident eao:departmentLocation ?dep.
}
GROUP BY ?dep
ORDER BY DESC(?sum)
LIMIT 10
```

Output Format: HTML

View Result: Plain

Map Bounds:

dep	sum
http://id.insee.fr/geo/departement/75	"6466"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/13	"4109"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/93	"3031"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/92	"2873"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/94	"2721"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/69	"2026"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/60	"1799"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/33	"1641"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/91	"1419"^^<http://www.w3.org/2001/XMLSchema#integer>
http://id.insee.fr/geo/departement/59	"1181"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 9: Example of SPARQL query against Strabon workbench to list with the top 10 the safest Departments of France and their accident counts

4.4 Conclusions

Across this chapter we have tested and presented the results of our transformation experiments. Four different semantic tools have been tested against the tasks of storing our data and transforming them into their final RDF-enabled format. We have listed the challenges laying on each solution, and how we have finally achieved to solve them. Also, we have seen in practice how we managed to unify our distinct datastets from two heterogenous sources by using GeoSPARQL's topological functions. The technical specifications and details shared within this chapter, can enable us to further develop services for transforming and mapping heterogenous data. Then, we have put to the test the steps of knowledge Repository and Knowledge Deployment by storing and publishing all of our data in Strabon. Lastly, we have tested the sanity of our newly created EKG by validating some of ENGIE's business questions in the form of SPARQL queries.

The data quality of heterogenous data sources is an essential factor. In particular, the EKG hugely benefits by non-redundant, up-to-date and complete data sets [19]. As a whole, a flexible architecture that enables aggregation, storage, querying, processing, and analyzing of Big Data in a graph structure is the core prerequisite here. The data store has to be optimized for graph operations and should use a flexible and extensible data schema for organizing the information. The main task of the data transformation step, is to connect to multiple internal and external data sources for extracting relevant structured and unstructured data from different data formats and by finishing this chapter we have presented a proof-of-concept for our proposed EKG framework.

5 GENERAL CONCLUSIONS AND FUTURE WORK

Before we close this work, in this chapter we are going to share our thoughts on each element that embodies this thesis. To recapitulate, we started by enabling the readers to understand the context and the challenges that lie ahead of EKG realization. The majority of enterprises nowadays, are eager to capture more knowledge and make good use of it. Knowledge Graphs for the last decades have gathered the attention of the industry for their alternative and powerful knowledge representation, but it is only in the very short past that organized attempts are taking place in the research world or realizing a KG for the enterprise, or as we tend to call it an Enterprise Knowledge Graph.

Although the demand has skyrocketed for EKGs, the vast majority of enterprises are still using dedicated KGs per domain of their business, usually not connected between them. Then, the always increasing heterogenous sources of information and the dependence in traditional RDBMS systems is further slowing down the development of a concrete EKG framework. In this thesis, we have attempted to study and understand the state-of-the-art approaches on EKG creation with a vision to comprehend the bottlenecks, the similarities and the differences in order to propose what makes in our opinion a coherent sub-set of steps in forming an EKG. A serious number of approaches presented here, lack what we defined as the must-have characteristic of openness. Enterprises are businesses driven by profit for their organizations and it is pretty difficult for them to migrate or adapt to a framework that is not flexible and does not re-use their existing automation and workflows. EKG creation has to always be inspired by flexibility in the technology stack. We also have seen some open-source EKG approaches, which were mainly developed and tested in real enterprises and have been successful integrating the semantic concepts in heterogenous data sources. For all approaches, we have compared their methodologies by structuring two comparable tables so the readers can highlight the main similarities, architecture and differences between them. Finally, for the state-of-art we also attempted to give a proposed EKG framework consisting of the most important steps and features in our perspective.

In our attempt to build ENGIE's Enterprise Knowledge Graph, we came across the challenges and the problems of several steps as described by our proposed EKG framework. By solving problems first occurring during Data Ingestion we have been able to form a unified semantically enabled dataset ready to be stored in RDF-stores. That would be the Knowledge Repository step which also has its own challenges before the published EKG is put to the test by the Knowledge Deployment step.

After EKG realization this thesis tried to present the challenges of heterogenous data transformation within a specific organization (ENGIE) and a study case of the accidents in the roads of Metropolitan France. For this part, driven by the semantic paradigm and EKGs we attempted to solve the transformation problem with the use of semantic tools. We have tested five different tools (Strabon, Ontotext GraphDB, Apache Jena, Geotriples and SPARQL-Generate) for transforming two heterogenous sources (CSV, RAW shapefiles) into a unified RDF dataset. Comparing the tools showed us that different approaches within the same technology domain exist. A relational database enabled solution, a file system solution or even an in-memory solution can handle data, store them and transform them. The times between those solutions differ in a great manner, but in our view each tool can be a great fit for different enterprise needs. We have to understand that semantic technologies are efficient when they are able to fit a domain perfectly. Each domain has different needs and can fit with different tools. Having presented the transformation problem, we resumed on the effort to realize ENGIE's EKG by testing whether our unified dataset is capable of answering some important business questions of the domain. Our queries displayed once again the challenges behind combining two different ontologies but finally the task has been successful.

As a future work, we would encourage a deeper look into automated or semi-automated frameworks so EKG creation becomes more efficient. Deep Neural Networks could solve many adaptation and transformation problems, while Named Entity Recognition could eliminate manual work in Knowledge evaluation and refinement. In general, we believe that combining Semantic Web with Machine Learning methods can give a new dimension to EKG construction and that in a few years' time more complete approaches with make their appearance. Another interesting development of the work would be the identification of the characteristics of the roads with the most accidents (turn, intersection, speed limits).

We hope that this work has been a step forwards in understanding what is missing from most of the state-of-art and it will be a reference for other teams and for ENGIE in particular to start developing their own EKGs.

ABBREVIATIONS - ACRONYMS

W3C	World Wide Web Consortium
OGC	Open Geospatial Consortium
KG	Knowledge Graph
EKG	Enterprise Knowledge Graph
AI	Artificial Intelligence
ML	Machine Learning
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
OWL	Web Ontology Language
RML	RDF Mapping Language
WKT	Well-Known Text
GML	Geography Markup Language
CRS	Coordinate Reference Systems
BI	Business Intelligence
ETL	Extract, Transform, Load
NER	Named Entity Recognition
LED	Linked Enterprise Data
NLP	Natural Language Processing
URI	Uniform Resource Identifier
RDBMS	Relational Database Management System
API	Application Programming Interface

BIBLIOGRAPHY

[1] M. Perry, "Ogc geo. sparql: Standardizing spatial query on the semantic web," 2011. [Online; accessed August 10, 2021]. 11, 41

[2] S. Panagiotis, "Nomothesi@ api - reengineering the electronic platform," 2015. 15

[3] G. Gottlob, A. Benczúr, and J. Demetrovics, "Advances in databases and information systems, 8th east european conference, adbis 2004, budapest, hungary, september 22-25, 2004, proceesing," 01 2004. 17

[4] O. W. Group, "Web ontology language (owl)," 2012. [Online; accessed August 10, 2021]. 18

[5] R. Battle and D. Kolas, "Geosparql : Enabling a geospatial semantic web," 2011. 19

[6] O. Group, "What is a knowledge graph?," 2021. [Online; accessed August 10, 2021]. 19

[7] M. Galkin, S. Auer, M.-E. Vidal, and S. Scerri, "Enterprise knowledge graphs: A semantic approach for knowledge management in the next generation of enterprise information systems," 04 2017. 20

[8] G. Garbis, K. Kyzirakos, and M. Koubarakis, "Geographica: A benchmark for geospatial rdf stores (long version)," in *International Semantic Web Conference*, 2013. 20

[9] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Comput. Surv.*, vol. 54, July 2021. 23

[10] R. Lu, C. Fei, C. Wang, S. Gao, H. Qiu, S. Zhang, and gen Cao, "Hape: A programmable big knowledge graph platform," *Inf. Sci.*, vol. 509, pp. 87–103, 2020. 24

[11] C. Bizer, C. Becker, P. N. Mendes, R. Isele, A. Matteini, and Schultz, "Ldif - a framework for large-scale linked data integration," 2012. 24

[12] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler, *How to Build a Knowledge Graph*, pp. 11–68. 02 2020. 25

[13] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, "Ontologies for enterprise knowledge management," *IEEE Intelligent Systems*, vol. 18, no. 2, pp. 26–33, 2003. 25

[14] J. Sequeda, W. Briggs, D. Miranker, and W. Heideman, *A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases*, pp. 526–545. 10 2019. 26

[15] A. Blumauer and H. Nagy, "The knowledge graph cookbook: Recipes that work." 26

[16] D. Song, F. Schilder, S. Hertz, G. Saltini, C. Smiley, P. Nivarthi, O. Hazai, D. Landau, M. Zaharkin, T. Zielund, H. Molina-Salgado, C. Brew, and D. Bennett, "Building and querying an enterprise knowledge graph," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 356–369, 2019. 27

[17] T. Ruan, L. Xue, H. Wang, F. Hu, L. Zhao, and J. Ding, "Building and exploring an enterprise knowledge graph for investment analysis," pp. 418–436, 10 2016. 28

[18] K. Kyzirakos, I. Vlachopoulos, D. Savva, S. Manegold, and M. Koubarakis, "Geotriples: A tool for publishing geospatial data as rdf graphs using r2rml mappings," *CEUR Workshop Proceedings*, vol. 1401, pp. 33–44, 01 2014. 47

[19] L. Masuch, "Enterprise knowledge graph - one graph to connect them all," 2014. [Online; accessed August 10, 2021]. 51