Thesis in Geographical Information Technics No 23

Data transformation using linked data ontologies

Gunnar Rolander

2017
Civilingenjörsutbildningen i Lantmäteri
Lunds Tekniska Högskola

Institutionen för Naturgeografi och Ekosystemvetenskap
Lunds Universitet

Gunnar Rolander (2017).
Data transformation using linked data ontologies
Master degree thesis, 30 credits in *Program in Surveying and Land Management*
Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *September* 2016 until *January* 2017

Disclaimer

This document describes work undertaken as part of a program of study at the University of Lund. All views and opinions expressed herein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Data transformation using linked data ontologies

Gunnar Rolander

Master thesis, 30 credits, in *Program in Surveying and Land Management*

Lars Harrie
Department of Physical Geography and Ecosystem Science

Ulf Månsson
Sweco Position

Examiner: Ali Mansourian
Opponents: Måns Andersson and Moa Eklöf

Preface

First of all I would like to thank my supervisors: Lars Harrie for his invaluable feedback and support and Ulf Månsson for coming up with the idea for the project and help with the technical part.

I would also like to thank everyone at the Sweco Position office in Malmö for welcoming me into the group and for the warm atmosphere. I've had a blast working with you.

Abstract

Interoperability is a sought after quality in most software. One way to increase interoperability is to make transformation of data between formats easier. Several transformation tools exists and many utilizes the Extract, Transform, Load-process to perform the transformations.

The transformation part of the process can be done in several different ways. Using a lookup table to arrange the underlying structure of data is one way which is commonly used.

This study tests a method of transforming geodata using the semantic web technologies to create the underlying data structure. The Semantic Web is a concept of an internet which is readable by both man and machine, by giving data semantic meanings that computers can understand. A test is made where these technologies are applied in unison with a tool using the Extract, Transform, Load (ETL) process to transform geodata. The result is compared to a transformation done using the same ETL tool with a lookup table approach.

The test of the method was successful. It is possible to use the semantic web technologies when transforming geodata in ETL tools.

While the method tested works, it's quite unwieldy and thus unlikely to replace current methods used in transformations. It could possibly be extended to utilize computer reasoning to a larger degree and used to determine the transformation parameters by itself. It could also be of use as a link between regular geodata and linked web geodata.

Sammanfattning

Interoperabilitet är en eftertraktad kvalitet hos de flesta typer av mjukvara. Ett sätt att öka interoperabiliteten är att göra det lättare att transformera data mellan olika format. Det finns många verktyg utformade för att transformera data och många använder Extract, Transform, Load-processen för att utföra själva transformeringen.

Transformeringsdelen kan utföras på olika sätt. Ett sätt är att använda lookup-tabeller för att strukturera datan korrekt.

Den här studien testar en metod för att transformera geodata som tar hjälp av semantisk webb-teknologi för att skapa den underliggande datastrukturen. Den Semantiska Webben är ett koncept där information på internet får en innebörd som är läsbar och förståbar för både människa och maskin. Genom att ge data en innebörd datorer kan förstå kan datorerna även resonera och dra slutsatser utifrån datan. I studien görs ett test där semantisk webb-teknologi integreras i ett tranformeringsverktyg för att transformera geodata.

Testet av metoden var lyckat. Det är fullt möjligt att använda semantisk webb-teknologi vid tranformering av geodata i ett ETL-verktyg.

Metoden som testades fungerade, men är något otymplig och omständig och därför är det osannolikt att den skulle ersätta metoder som används i nuläget. Om man däremot byggde på metoden så att den i större utsträckning utnyttjade dator-logik skulle eventuellt transformations-parametrarna kunna resoneras fram av datorn. Om metoden byggdes ut borde den också kunna vara till hjälp transformationer mellan vanlig geodata och länkad webb-geodata.

Abbreviations

DAGI	Danmarks Administrative Geografiske Inddelning
DB	Database
ETL	Extract, Transform, Load
GIS	Geographic Information Systems
GML	Geography Markup Language
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
KML	Keyhole Markup Language
OGC	Open Geospatial Consortium
OSM	Open StreetMap
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language
W3C	World Wide Web Consortium

Contents

1 Introduction

1.1 Background

Interoperability has been an issue for as long as there's been different file standards. It is defined by the Open GIS Consortium as the "capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units" [8]. Defining standards and transforming data are two solutions to increase interoperability.

Interoperability is a sought-after quality for many reasons: fear of vendor lock-in, the ability to work with several different applications and technologies seamlessly, and the ability to co-operate with other organizations or companies without too much hassle.

Throughout the years different solutions to achieve higher interoperability have been developed. One solution is defining an open standard, as was done in the Geographic Information System (GIS) world during the 90's to increase interoperability in the GIS field [29]. Standards have their own problems though. They require more or less everyone to follow/implement them to reach their full potential of effectiveness and interoperability.

A key issue in interoperability is data transformation. Transforming data to a format or model with higher compatibility can lead to better interoperability. Data transformation is generally divided into two main categories: syntactic and semantic. The function of syntactic transformations is the alignment of data from one format into another. Semantic transformations handles alignment of the meaning of the data. One process used in data transformations is the Extract, Transform, Load (ETL) process, which several softwares utilize.

Data transformation can be done using semantic web technologies. The semantic web (sometimes referred to as Web 3.0[33]) technologies are a collection of technologies and formats originally designed to create an extension of the web readable by machines using linked data. Central to this ambition is the Resource Description Framework (RDF), a framework which describes resources, their properties and relations to each other using statements called triples. Even though the semantic web so far has not lived up to the hype, the technology still has a lot of merits.

1.2 Problem statement

Working with many formats in a project can often prove both time-consuming and annoying. Transforming data between GIS formats often requires a custom-made solution using either an ETL tool or a chain of different software which can open and save in interim formats. The quality of ETL-solutions are dependant on the creator of the workflow applied as well as the implementation of the tool. Using a chain of different software and formats can often lead to data loss, depending on the intermediate formats.

Harmonization of data by connecting different data models in a federated model using ontologies is possible, as shown by Métral et al.[30]. A question is whether it would be possible to transform data from one format to another using a linking rule set, containing metadata for the formats and their attributes as well as how they relate to each other.

1.3 Aim

The overall aim of this project is to study the transformation problem of geodata and determine whether using semantic web technology could be a possible solution.

The aim of the theoretical part is to study the semantic web technologies, commonly used formats and standards. A brief study of formats in GIS relevant to the use cases is also included.

The aim of the practical part of the project is to verify the method of transforming data using ontologies and RDFs as well as developing a method to integrate it into an ETL workflow. The quality of this method is also evaluated.

1.4 Method

To fulfill the aim of this project the work is done in several distinct steps. (1) Current transformation methods are described. (2) The method using ontologies are described. (3) A practical test is done where the methods are compared. (4) The results of the test are evaluated.

The theoretical part consists of studying relevant technologies and formats, i.e. the semantic web technologies and its connected formats (RDF, SPARQL, OWL etc). Since ontologies are a key component it and its' geospatial variants are studied in detail. Common GIS formats and general data transformation methods are also investigated.

The practical test uses an ontology, more specifically a linking rule set, to link two formats. The linking rule set are used to create the transformation rules. The visualized concept of a linking rule set can be seen in figure 1.

To determine the feasibility of transforming data using ontologies a comparison with the transformation using lookup tables is done. A main criteria for evaluation is the amount of human input needed to perform the transformation. The amount of lost data is another criterion.

1.5 Delimitations

Due to time restraints only syntactic transformations is studied in this study.

1.6 Disposition

Following the introductory chapter (chapter 1) in this report is a chapter defining the use cases (chapter 2). In chapter 3 and 4 general data transformation methods and the semantic web technologies are described respectively. Chapter 5 describes the usage of semantic web technology in transformations and in geospatial applications and chapter 6 describes the case study, the process and results of developing the linking rule set and the comparison with the solution using lookup tables. This is followed by a discussion in chapter 7 and conclusions in chapter 8.

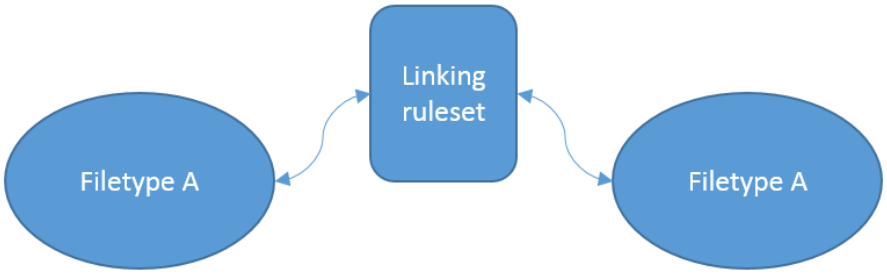

Figure 1: Two data models linked via linked ruleset

2 Use cases

2.1 Use case - Transforming data between formats

The distribution of geodata in Denmark is performed by the Agency for Data Supply and Efficiency (in Danish: Styrelsen for Dataforsyning og Effektivisering (SDFE)). They recieve data from multiple sources and supply it for free to both private and public sectors and users [13]. To do this they have developed a self-serve web-portal called Kortforsyningen/Download using ETL workflows to serve data in several formats.

Kortforsyningen/Download is the SDFE's self serve solution for distribution of geodata. The structure of Kortforsyningen/Download is displayed in figure 2. The SDFE transforms geodata it receives from several private and governmental contractors and stores it in an PostGIS Database. When someone requests data from the web portal they specify a format, projection and sometimes an area and then get a download link to retrieve the data. Kortforsyningen/Download supplies over 120 different datasets and each dataset is available in at least one but generally three to four formats, commonly Shape (.shp) and GML (.gml) as well as MapInfo Tab (.tab).

To generate the requested data an ETL workflow is run on a server. This workflow is specifically designed to transform a dataset to a specific format, which means that several hundred unique workflows has to be created. Each of these workflows need a lookup table defining what attributes and features should map to in the destination format. Since there's no established standard procedure regarding this it all need to be done manually.

To decrease the number of workflows needed ontologies could be used. When using ontologies only a single workflow is needed while the linking rule set differs depending on what destination format is requested. The linking rule sets still need to be created manually but could ideally be re-used between datasets stored and available in the same formats.

One specific dataset which the Kortforsyning supplies is DAGI (Danmarks Administrative Geografiske Inddelning/Denmarks Administrative Geographical Division) 1:10.000, which is a map of the borders between administrative counties in Denmark. It's available in ESRI Shape, MapInfo Tab and Geomedia database (.mdb).

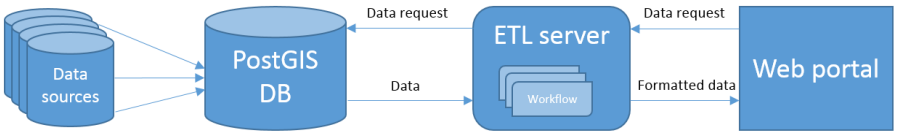

Figure 2: Structure of Kortforsyningen/Download.

3 Transformation methods and tools

3.1 Syntactic and semantic transformation

Data transformation converts data in one format or structure into a destination format or structure. When transforming data two issues must be addressed, semantic tranformation and syntactic transformation. The syntactic transformation handles the syntax of the data, how it looks or is represented digitally [27]. The semantic transformation handles the semantics, the meaning, of the data. Syntactic transformations are generally easier and requires less manual input than their semantic counterparts.

In figure 3 we can see the source structure of some data and the structure of the destination data. During the transformation a syntactic transformation were made. All the text had whitespace removed and the number had two zeroes added, thus the structure of the data, the syntax, was changed.

The transformation done in figure 3, changing the name of a column and adding zeroes to another, are just a few basic operations that can be performed. When transforming data there's a multitude of possible operations, such as aggregating data, transposing columns to rows, encrypting, encoding and sorting data, and many more.

A semantic transformation changes the meaning of the data. An example of this is how classification of land can differ depending on the purpose of the map. In figure 4 two maps of the same land area, classified by land use, can be seen. In the left map only two classifications exist: *forest* and *open fields*. Forest is defined as all land with trees and open fields is defined as all land without trees, which includes meadows and cultivated fields. In the right map three classifications exists: forest, open fields as well as *cultivated fields*. In this case open fields are defined as all land without trees, not including cultivated fields. The forest area is the same in both but the definition of open fields differ, and thus the meaning and semantics differ.

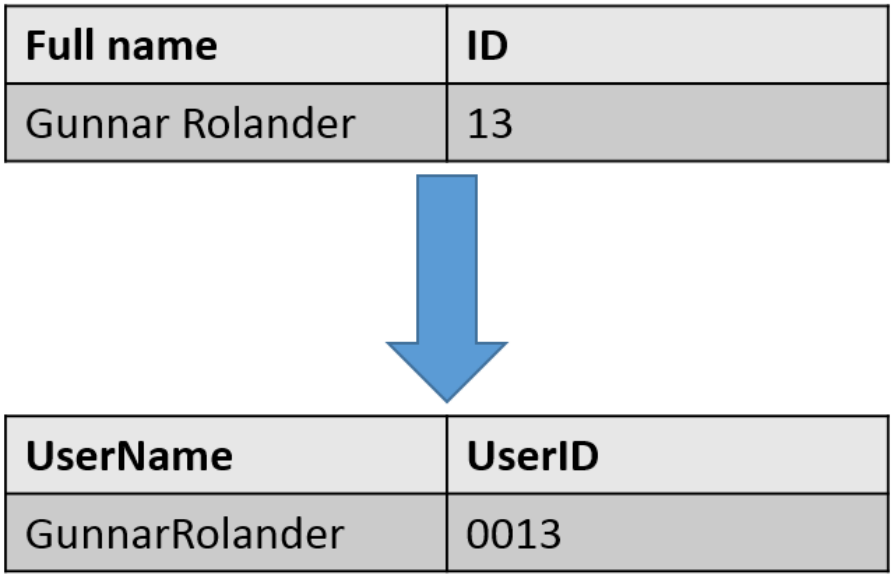

Figure 3: Example of syntactic data transformation.

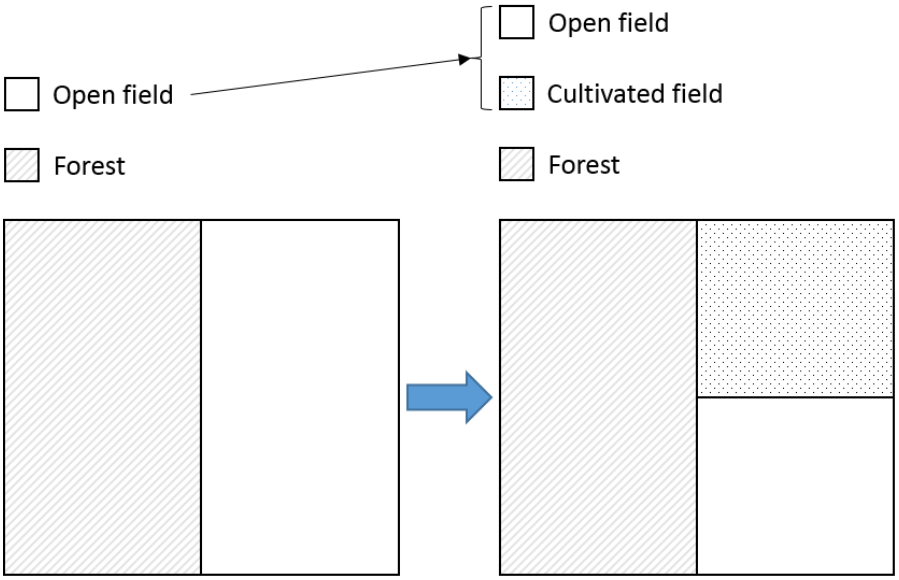

Figure 4: Example of semantic data transformation.

3.2 Extract, Transform, Load

Extract, transform, load (ETL) is a process used in data transformation. In the extract phase, data is extracted from the source file(s) and converted to a common intermediate format. The transformation phase involves the transformation of the data to the structure of the destination format. The load phase then loads the data into the destination file.

The ETL process is utilized in many data transformation tools. ETL tools are diverse and used in many applications. A few ETL tools are specialized to work with spatial formats: Feature Manipulation Engine (FME), the open-source tool GeoKettle and the Spatial Data Integrator add-on to the ETL tool Talend, to mention a few.

3.3 Feature Manipulation Engine

The Feature Manipulation Engine (FME) is an ETL tool suite developed by Safe Software. FME uses the ETL process mentioned above to transform data. FME is specialized in transforming geospatial data but can handle some other common formats too. Besides transforming data FME can also perform some GIS operations such as selecting data in a bounding box, adding buffers and merging layers.

To transform data in FME a workflow is created. Using a graphic user interface *readers*, which reads data from a source, *writers*, which writes data to a destination, and *transformers*, which performs a wide variety of operations on the data, are added and connected. The way the readers, writers an transformers are connected determine the flow of the data and in what order operations are carried out. An example of a very basic workflow can be seen in figure 5.

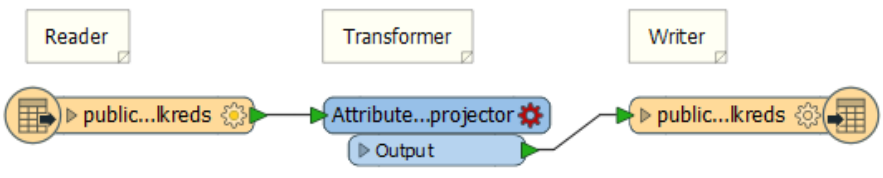

Figure 5: Basic FME flow with a writer, transformer and reader.

Even though the graphic interface is what is commonly used to create workflows it can also be programmed by writing code, which is how it was done before the graphic interface was introduced. Users can copy a part of a graphic workflow and paste in a text editor, which will show the code of that part of the workflow. It can also be pasted in another workflow. The following code is an example of a transformer creating an attribute named *_prio* with the value 1.

```
DEFAULT_MACRO WB_CURRENT_CONTEXT

# ----------------------------------------------------------------

FACTORY_DEF * TeeFactory                                        \
  FACTORY_NAME "$(WB_CURRENT_CONTEXT)_CREATOR_BRANCH_TARGET"\
  INPUT FEATURE_TYPE *                                          \
 OUTPUT FEATURE_TYPE *

# ----------------------------------------------------------------

FACTORY_DEF * AttrSetFactory                                    \
FACTORY_NAME AttributeCreator_3                                 \
ATTRSET_CREATE_DIRECTIVES _PROPAGATE_MISSING_FDIV               \
INPUT  FEATURE_TYPE Sampler_NOT_SAMPLED                         \
ATTR _prio 1                                                    \
   OUTPUT OUTPUT FEATURE_TYPE AttributeCreator_3_OUTPUT
```

When transforming data FME uses an intermediate format, often referred to simply as "FME native format". FME comes preconfigured with default lookup tables mapping every supported format to the native FME format.

3.4 GIS formats

In the GIS field the need for transforming data is big. One reason for this is the myriad of file formats that exists. At the moment there's almost 400 different spatial file formats (based on the number of formats supported by FME). One possible explanation for this diversity is illustrated in figure 6.

In the 90's an effort to establish open GIS standards by creating the Open Geospatial Consortium (OGC) were made [18]. The effort was successful and these days the thirty-five or so OGC abstract standards are implemented in most GIS file formats and technologies [29]. One of the results of the success is that many ventures have extended the open standards and created their own flavour of it. An example of this is the Geography Markup Language (GML). Other notable GIS formats are the vector Shapefiles (.shp) and the rasterformat GeoTIFF.

3.4.1 Geographic Markup Language

The Geography Markup Language (GML), according to [12], "...is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features.". The XML grammar GML was defined and released by the OGC as an open standard in 2000 [23]. It is designed to be a standardized encoding that is open while still being useful to both governmental, academic and private sectors.

To achieve the storage and transportation of two-dimensional geographic data in GML Simple Features, which is an OGC and International Organization for Standardization (ISO) standard, is utilized. The key geometries used in GML is point, linestring and polygon.

The release of GML 3.0 in 2003 introduced more complex geometries and features, such as 3D coordinates and topology, to GML. GML 3.0 is

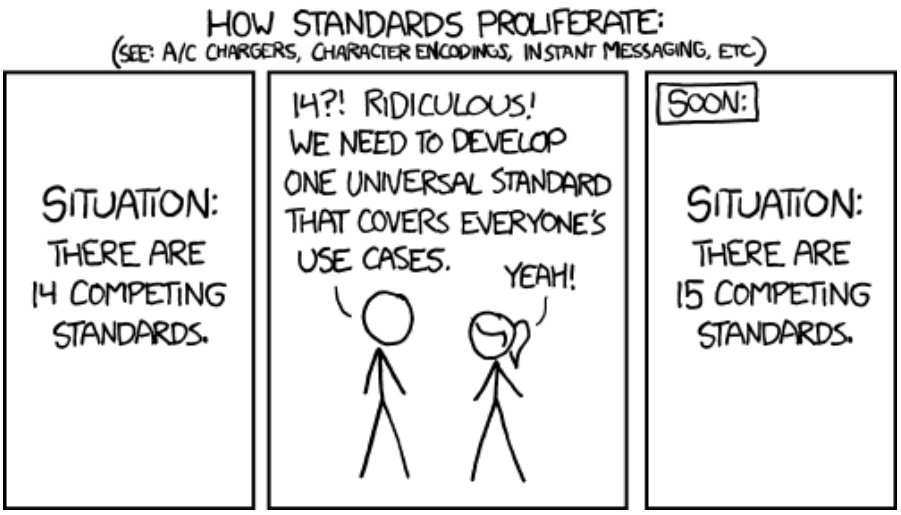

Figure 6: The webcomic XKCD regarding standards (xkcd.com/927/).

backwards compatible, which was important since GML 2.0 was used in several commercial software products at the time of its' release [35].

Due to the openness and design of GML anyone may create an application-schema which fits a special need or niche, which has led to a great variety of schemas [9]. Some GML schemas have in turn become OGC standards, such as CityGML and IndoorGML. CityGML is an application schema that is intended to be used to model representations of sets of 3D urban objects [17]. IndoorGML is used to model indoor spatial information.

4 Semantic web technologies

The semantic web technologies are a stack of technologies associated with the semantic web (sometimes referred to as Web 3.0). The semantic web is a concept of an internet consisting not only of human-readable pages and documents, but of both that and data which are machine-readable. The idea was that site owners should add ”invisible” data to their sites, which linked to other data and created a huge network of linked data readable by computers and other machines. This concept was originally created by the inventor of the World Wide Web, Tim Berners-Lee [5].

The semantic web was a very hyped up concept during the early 21st century, thought to bring a revolution of artificial intelligence and a smarter internet. While this never really happened and the semantic web never lived up to the extreme hype it did still garner some successes and is now used on about 3 million websites. The semantic web technologies have also found uses in fields as diverse as medicine, finance and library catalogues.

Three semantic web technologies are the Resource Description Framework, a framework which stores the semantic data, the SPARQL query language which is used to query the data and the Web Ontology Language which introduces inference and enables reasoning in the queries.

4.1 Resource Description Framework

Resource Description Framework is a model framework commonly used when working with linked data. It's a W3C standard.

An RDF dataset (or graph) consists of one or more triples. A triple is a statement which consists of three things: a subject, an object and a predicate. The subject is the thing that is described, the predicate is the property of the thing that is described and the object is the value of that property. This structure of subject, predicate and object is sometimes known as the SPO-structure. The subject, predicate and object can be more or less anything, depending on what the purpose of the dataset is. An example could be *Lund* (subject) *is a* (predicate) *city* (object).

A node in a triple (subject, predicate or object) can be one of three things: An IRI, an RDF literal or a blank node. Internationalized Resource Identifiers (IRI) can point to webpages or local filepaths, which usually contains descriptions of the resource in case and often links to other resources. The IRI is one of the main features of the RDF, since it enables the linking of data. We can modify our previous example to use the definition of the relationship *is a* as defined in RDF Schema (RDFS) by using the URI to the definition instead of just text, which would look like the following: *Lund http://www.w3.org/2000/01/rdf-schema#isA city*. URIs that are frequently used are often shortened (prefixed), in our example we could associate the URI *http://www.w3.org/2000/01/rdf-schema#* with the prefix *rdfs:* as is conventionally done [11]. Our example would then take the more readable form of *Lund rdfs:isA city*. You can find more info about RDFS in the next section.

An RDF literal is used to store a value, such as a number, name or date. A literal consists of two elements: a Unicode string which is the actual value and a datatype IRI which defines the datatype of the value. One exception exists where a literal consists of three elements: if the datatype is a langString, i.e. a string of a specific language, a language tag is added as a third element.

A blank node is defined as the disjoint of IRIs and literals. It's not a identifier and not a value, instead it represents an anonymous resource. While IRIs and literals can have any role in a triple, a blank node can only be the subject or object according to the RDF standard.

By using IRIs several different datasets and ontologies can be linked, which is called *linked data*. Several serializations of RDF exists, a few of the most popular are Turtle, RDF/XML, JSON-LD, N-Triples and Notation3 (N3) [33].

Here's an example, with the added info of Lunds relation to Sweden, written in standard Turtle syntax.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .

Lund rdfs:isA city .
Lund geo:sfWithin Sweden .
```

A visualization of the data model described above can be seen in figure 7.

4.2 Ontologies and inference

Ontologies (or vocabularies, the terms are used somewhat interchangeably) as defined by W3C [10] are "... used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. In practice, ontologies can be very complex (with several thousands of terms) or very simple (describing one or two concepts only)". Ontologies can be stored in the RDF format. A visualized ontology can be seen in figure 8. The ontology in question is a part of the linking rule set used in the case study, describing the links between FME's native data types and MapInfo Tab's data types.

One of the most common vocabularies is the RDF Schema, which is an ontology that "...provides mechanisms for describing groups of related resources and the relationships between these resources." [11].

Ontologies can be used to discover new relationships in linked data by following the logical rules in the ontology. This process is called inference. Using ontologies and inference, combined with a reasoning engine makes it possible to retrieve information or data that were originally not in the model. Loosely speaking one might say that new data is derived from or discovered in the original data.

Continuing on the example with Lund above (*Lund rdfs:isA city; geo:sfWithin Sweden*) we could add the assertion that *Sweden geo:sfWithin Europe*. With this info the inference that *Lund is within Europe* can be made.

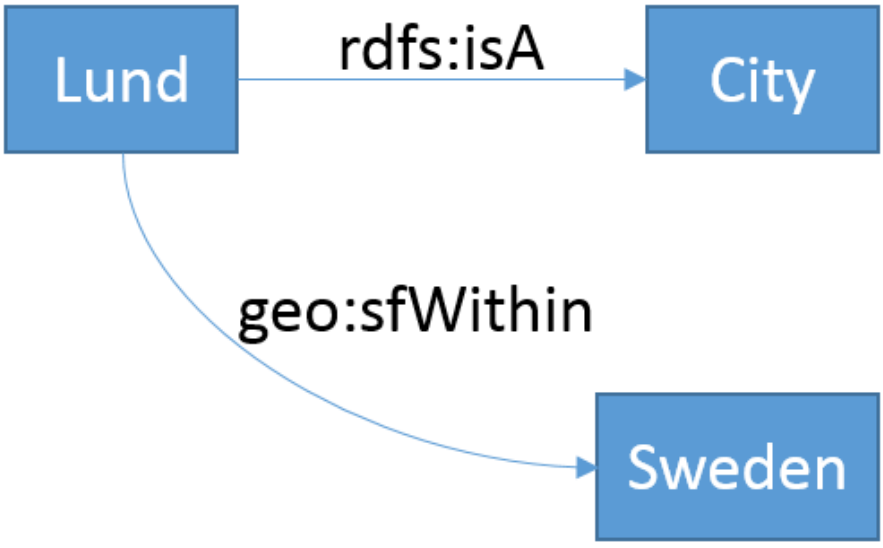

Figure 7: A graphical representation of Lunds properties and relations.

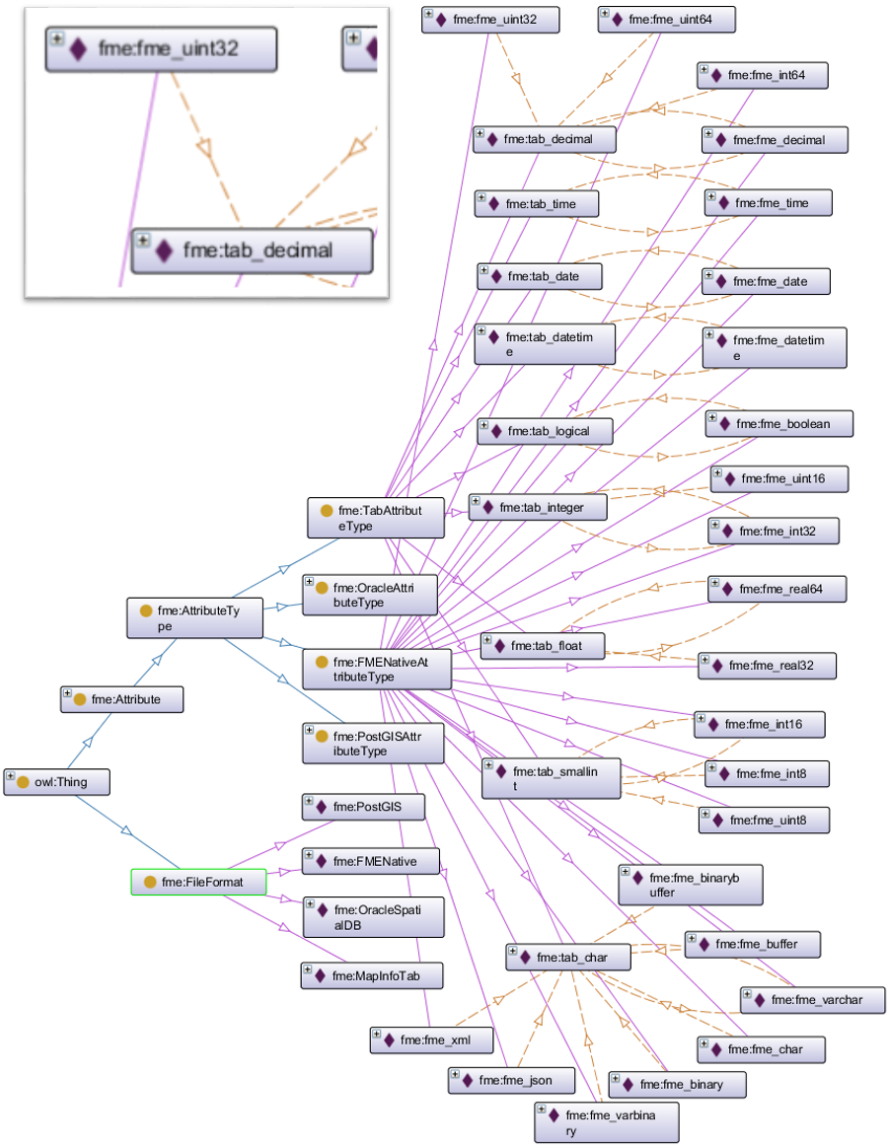

Figure 8: An ontology visualized in Protégé using the plugin OntoGraf.

4.3 Web Ontology Language

The Web Ontology Language (OWL) is a family of languages used to describe the semantics in ontologies. The OWL languages are based on RDF.

The OWL family of languages is defined by the W3C in three variants: OWL Lite, OWL DL and OWL Full. The simplest one is OWL Lite. OWL DL is an extension of OWL Lite and OWL Full is in turn an extension of OWL DL [28].

OWL 2 is a definition by the W3C released in 2009 which extends the original OWL family and introduces three new OWL species: EL, RL and QL. These species works the same way as OWL Lite, DL and Full, each species being capable of more advanced reasoning than the former [20].

Since OWL is based on RDF they share many things, such as serializations and syntaxes which means that OWL ontologies can be written using Turtle, N3 etc, and stored using RDF.

4.4 Geo ontologies

A number of ontologies is designed to give the tools to describe geospatial information.

One of the most basic is the *Basic Geo (WGS84 lat/long) Vocabulary* which is a minimalistic ontology for describing points with latitude, longitude and altitude in WGS84. It can also be used to connect a point to an object. In 2007 an updated and extended version, *Geo OWL*, was released. Geo OWL uses the GeoRSS model, a model used to add positional data in an RSS feed, to describe geometries and features [26].

Another early geo ontology is *GeoOnion*, a vocabulary designed to describe relations between points of interest using a number of concentric circles centred on one of the points. These layers of circles vaguely resembling the layers of an onion, hence the name. The radius of the circles is 3 power the number of the circle. So the first circle is $3^0 = 1$, the second $3^1 = 3$, and so on. This approach lacked input from geographic information science and is as of yet incomplete [18].

NeoGeo is yet another proposed vocabulary to describe geodata in RDF. It is based on the Simple Features standard but also describes composite geometries as aggregations of simple geometries. NeoGeo was developed by the NeoGeo community and proposed as a common representation for GeoData in 2009. It unfortunately never became more than a draft vocabulary. Work on it has since been taken over by GeoVocab.org [36].

One of the more advanced and widely used geo ontologies is the ontology used in GeoSPARQL, which is discussed later.

4.5 SPARQL Protocol and RDF Query Language

The SPARQL Protocol and RDF Query Language (SPARQL in short, it's a recursive acronym) is a query language used to retrieve and insert data stored in an RDF dataset. SPARQLs syntax resembles regular SQL with a few distinctive differences: a where-clause is always needed and all variables are defined with a prefixed *?* (e.g. *?name*).

The following example query retrieves the subject of all triples where the object is *"city"*.

```
SELECT ?a
WHERE
        {
                ?a ?b "city"
        }
```

The basis of a SPARQL query is an algebra expression that is built from different *graph patterns*. A basic graph pattern is a logic for extracting subsets in an RDF graph [15].

The simplest graph pattern is the *triple pattern*, which works like an RDF triple with the exception that it also can utilize the usage of variables as subject, predicate and object [34]. According to the W3C definition a triple pattern is member of the set:

$$(RDF - T \cup V) \times (I \cup V) \times (RDF - T \cup V) \tag{1}$$

where *RDF-T* is a all RDF Literals and blank nodes, I is a set of all IRIs and V a set of all variables [22].

A basic graph patterns is a set of triple patterns which matches a subgraph if all the triple patterns in the set matches. The basic graph pattern can be combined with value constraints and other basic graph patterns to create a SPARQL query.

4.5.1 GeoSPARQL

GeoSPARQL is an extension of SPARQL which enables geospatial queries and reasoning. This is done by introducing the concepts of features and geometries through a vocabulary describing them as well as a set of spatial functions for use in SPARQL queries [3]. GeoSPARQL also supports topological relationships between geometries and the usage of topological queries [32].

GeoSPARQL is an OGC standard. Since GeoSPARQL is just a standard exactly how it works may vary depending on how it is implemented. A number of triple stores or semantic graph databases implement some or all of the requirements and conventions outlined in the specification of GeoSPARQL.

To model topological relationships an implementation of GeoSPARQL should support the use of different but equivalent topological relation families such as Simple Features, RCC8 (Region connection calculus) and Egenhofer. This allows GoeSPARQL to do geometrical and spatial reasoning. GeoSPARQL is intended to be able to do both quantitative and qualitative spatial reasoning. Quantitative spatial reasoning involves defined geometries for features which can be used in calculating topological relationships or distances. Qualitative spatial reasoning has geometries which aren't well defined or even unknown, but the features can still have topological relationships which allows inference.

To find all parks in Lund using GeoSPARQL spatial reasoning can be utilized. First off we define the variable *?g1* as the geometry of Lund, then we define the variable *?g2* as any geometry of any park. Finally we select all parks, *?park*, where the geometry of the park, *?g2*, is within the geometry of Lund, *?g1*. An example of this query written in TURTLE syntax:

```
SELECT ?park
WHERE
  {
    ex:Lund geo:hasGeometry ?g1 .
    ?park geo:hasGeometry ?g2 .
    ?g2 geo:within ?g1
  }
```

4.6 Semantic web tools

4.6.1 StarDog

StarDog is a commercial semantic graph database which supports both RDF and OWL2. It uses SPARQL as query language. StarDog also provides rea-

soning capabilities since it supports OWL2. It can be run on Windows, Linux and OSX and can be interacted with through the terminal, a graphic browser interface and the Stardog Native API for the RDF Language (SNARL). The HTTP support allows a number of programming languages to connect to the database.

4.6.2 Protégé

Protégé is an open-source ontology editor. It provides the user with a user interface to create, edit and manage ontologies as well as run SPARQL queries on it. It was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. There are several plugins with a range of purposes. In this report the plugin OntoGraf is used to visualize an ontology [31].

5 Usages of semantic web technologies

5.1 Semantic web technologies in transformations

As the name hints the semantic web technologies excels mainly at the semantic parts of transforming, specifically semantic mapping. Syntactic mapping is commonly used in ETL software to define which attributes of the source file should be mapped to which attributes in the destination file. This mapping can be done by manually creating a look-up table stored as a file or in a database. An alternative to this is using ontologies and create a linked ruleset which describes the relationships between the input and output files and their attributes and let the reasoning engine determine the mappings.

Using ontologies in ETL processes is studied by Zhang and Wang[40]. A framework model for an ontology-driven ETL process is proposed. The process consists of four main phases: metadata abstract, ontology map, rule reason, ETL direction. In the first phase metadata is extracted as ontologies from the data sources. These ontologies are mapped to an existing domain ontology in the ontology mapping phase. In the rule reason phase a reasoning engine runs reasoning tests to find new relationships and conflicts in the ontology. In the final phase the transformation rules are provided to the ETL process which uses them to transform the data.

An early attempt to apply the semantic web technologies was done by Bowers and Ludäscher in 2004 [7]. They propose a framework using ontologies to do semantic mapping in the context of a scientific workflow. A case study was done on the Ptolemy system, an open source system used to design and execute data-flow process networks. Ptolemy works by connecting different services, which perform tasks on input data and serves the output through a port. What Bowers and Ludäscher wanted to do was an extension which transformed output data from one service to the correct structure to use as input in another service, since these tended to differ, based on semantic information extracted from the datasets.

Spencer and Lieu [37] suggests a process for transforming data sent between web services by using OWL-S and WSDL. OWL-S is an ontology for describing Semantic Web Services which is built upon OWL. WSDL (Web Services Description Language) is an XML-based language used to describe Web Services and their functionality. By running a rule compiler with the WSDL descriptions, OWL-S descriptions, auxiliary ontologies and specifications of data type converters as input transformation rules are created. These rules are then used to transform the data sent from one service to another.

5.2 Use of semantic web technologies in geospatial applications

Semantic web technologies can be used in geospatial applications, as shown by Métral *et al.* [30]. They make the argument that CityGML is insufficient to represent several aspects of a 3D city, such as complex transportation or mobility aspects. By describing urban information in ontologies and connecting them to CityGML-models a semantically enriched 3D city model is acquired, which can be used for various applications. A case study where an ontology describing soft mobility enriched a CityGML model of Geneva was used as a basis for the argument.

Zhao *et al.* [41] presents an approach to geospatial knowledge transformation which uses a web service chain throughout the whole life-cycle of the geospatial data. They propose that further standardization and classification

of geospatial data and processes be made, in the form of standards and ontologies on multiple levels. They argue that by having a solid foundation of geospatial standards and domain, data and process ontologies combined with the interoperability of web services geospatial domain knowledge can be precisely captured and represented. One of the web services proposed is called Geospatial Fusion Service and would transform maps between different formats, coordinate systems or reference systems.

A framework for sharing geospatial data is proposed by Zhang *et al.* [39]. The basic idea is to use geospatial semantic web technologies to create an ontology-based catalogue service which stored links to geospatial data sources as well as semantic information and meta data about the sources. They argue that this approach would increase interoperability and lower duplicate data being produced.

Ligouris *et al.* [25] argue that current implementations of spatial extensions to RDF stores focus mainly on supporting GeoSPARQL and less on the performance. They propose a number of extensions to RDF to increase the efficiency of indexing and supporting spatial queries. These extensions are *index support for spatial queries*, *spatial encoding of entities*, *spatial join algorithms* and *spatial query optimization*. A detailed system which implements these extensions is presented. They compare their extended version of a triple store (RDF-3X) with an unmodified version, as well as two commercial triple stores with spatial support. The conclusion that their approach minimizes the evaluation cost connected to spatial components in all RDF queries is made.

The Linked Geodata project is a project integrating data from a number of open data sources to create a large spatial knowledge base. This is accomplished by linking data from three open data sources with Open StreetMap data. The sources in question is DBpedia, GeoNames and the Food and Agriculture Organization of the United Nations (FAO). DBpedia is a knowledge base of structured information extracted from Wikipedia. The data of DBpedia is structured so that sophisticated SPARQL queries can be executed or the data itself be downloaded as RDF. GeoNames is a geographical database with over 10 millions geographical names linked to WGS84 coordinates which collects and aggregates data from over a hundred sources. The FOA provides information about the official and short names of countries, as well as the name of their nationality and currency in many languages. All this linked together data can either be queried via SPARQL endpoints or browsed as an enriched OSM map [38]. In total the Linked Geodata dataset contains over 200 million triples. A screenshot of the Linked Geodata browser can be viewed in figure 9.

With the exception of the Linked Geodata project the amount of geodata published in RDF is not massive. Out of 126 000 geospatial datasets published on data.gov, the US open data portal, about 30 could be downloaded as RDF. On the UK's counterpart, the numbers were higher: 224 out of 40 000 datasets. The federal data published by the US is free to use in any way, since it's produced by the government and released without any restrictions or copyright.

One governmental agency that have adopted the semantic web technologies is the United States Geological Survey (USGS). The USGS has published a SPARQL endpoint to a dataset derived from The National Map. This dataset includes very specific and detailed information regarding geographic names, land cover, structures, hydrography and muh more. It also stores the topological relations of entities [2]. The Ordnance Survey, Great Britain's national mapping agency, have also published five of their products as linked data as part of the OS OpenData initiative launched in 2010. In total they have published over 115 million triples spread over the five products [1]. The National Land Survey of Finland have also done a test in publishing linked geospatial data [19].

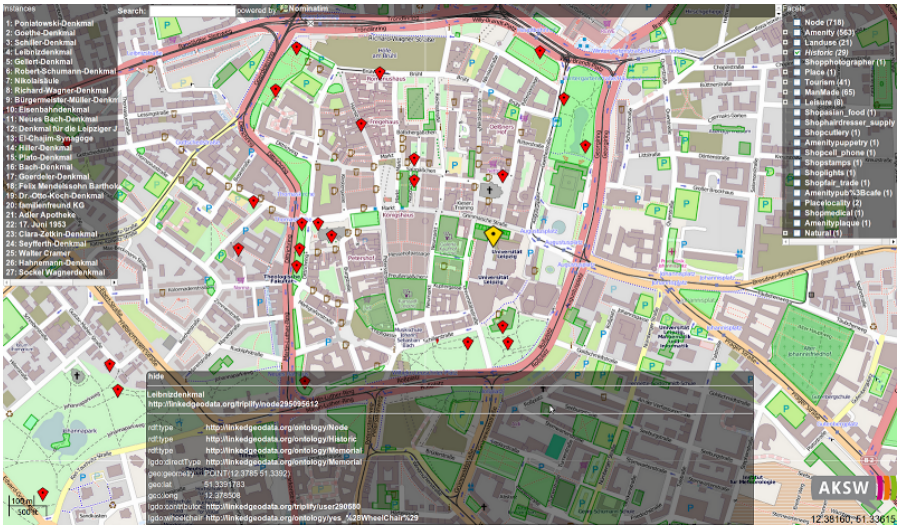

Figure 9: A screenshot of the Linked Geodata browser. Picture courtesy of http://aksw.org/

One project which aims to increase the amount of published geospatial linked data is the GeoKnow project. It intends to do this by creating a tool set known as the GeoKnow Generator framework. The GeoKnow Generator framework gives the user tools to convert a multitude of data types to geospatial linked data. The project is funded by the European Commission. Two tools that is included in the GeoKnow toolbox is Sparqlify and TripleGeo-Service. Sparqlify is a tool that enables creating RDF views on relational databases which makes them query-able with SPARQL. TripleGeo-Service is a tool that transforms geodata from some of the common geospatial formats such as GML, KML and ESRI Shapefile as well as DBMSs with geospatial support such as PostGIS and Oracle Spatial into triples. It also includes Virtuoso, which is a hybrid DBMS that can host relational databases, triple stores and document, file and web servers. The GeoKnow project is driven by the same group that are responsible for the Linked GeoData project, AKSW [24].

Another AKSW project is the GEISER platform. The GEISER platform is an integration software designed to handle the integration of multiple heterogenous data sources, among them social networks, satellite navigation systems and open geodata such as OSM. The basic idea is to combine a large amount of data streams, both geospatial and non-geospatial, into one big stream with linked data. GEISER uses semantic web technologies both to retrieve data from certain sources and to do semantic filtering. It is capable of extracting geospatial data by utilizing machine learning and graph algorithms.[16]

One interesting application using the semantic web technologies is DBpedia Mobile. DBpedia Mobile is a faceted browser made for mobile use which shows you information about entities with DBpedia entries in your vicinity, such as parks, rivers or shops. It uses Google Maps as a background map, and bases your position on IP and or GPS. DBpedia Mobile is a client which retrieves its' data from DBpedia and GeoNames, and presents it to the user on a map. Each entity shown also presents what it is linked to, which lets the user read related entries [4].

Dolbear and Hart [14] discusses solutions to ontology merging, which can be useful when extracting data from two or more sources described with different domain ontologies. The idea is to create a new ontology from the two source ontologies. They test three different approaches: Referencing, referencing with extension and referencing with restriction. Referencing is merely creating a new ontology which references classes in the source ontologies. Referencing with extension creates a copy of the class in the

source ontology which also is equivalent with it and further extends it with axioms. Referencing with restriction is the most advanced. It creates a new ontology and only incorporates info which is absolutely required.

6 Case study

The case study of this report center around the development of a workflow in an ETL tool (FME) using ontologies to map attributes. This workflow is compared with a workflow that is currently used in the Danish geodata distribution service Kortforsyningen/Download which uses the lookup approach.

6.1 Data

The data used in this case study is a dataset known as DAGI (Danmarks Administrative Geografiske Inddelning/Denmarks Administrative Geographical Division). The dataset is a standardized reference dataset which describes and shows Denmarks administrative geographic division.

Apart from the obvious geographical information and related geometries it also contains administrative data such as the name and code of a county, postal numbers, which agency that submitted the data and at what date the data was collected. The dataset is raw geodata, meant to be processed before being published as a finished map or web based map service.

Since DAGI encompasses quite a lot of data this study will only be using the dataset in scale 1:10.000 and the table containing data about the counties of Denmark. The data used in the test is stored on a PostGIS server. This data does not correspond to the latest data that can be found on Kortforsyningen/Download, but is data that was used in the development of the currently used workflow. A screenshot of the latest data can be seen in figure 10.

The wanted configuration and schema of the finished data is defined in an XML file which is used in the inception of both workflows. This file, just as the test data, does not correspond to the configuration of the data currently available from Kortforsyningen. The data used in the comparison is data

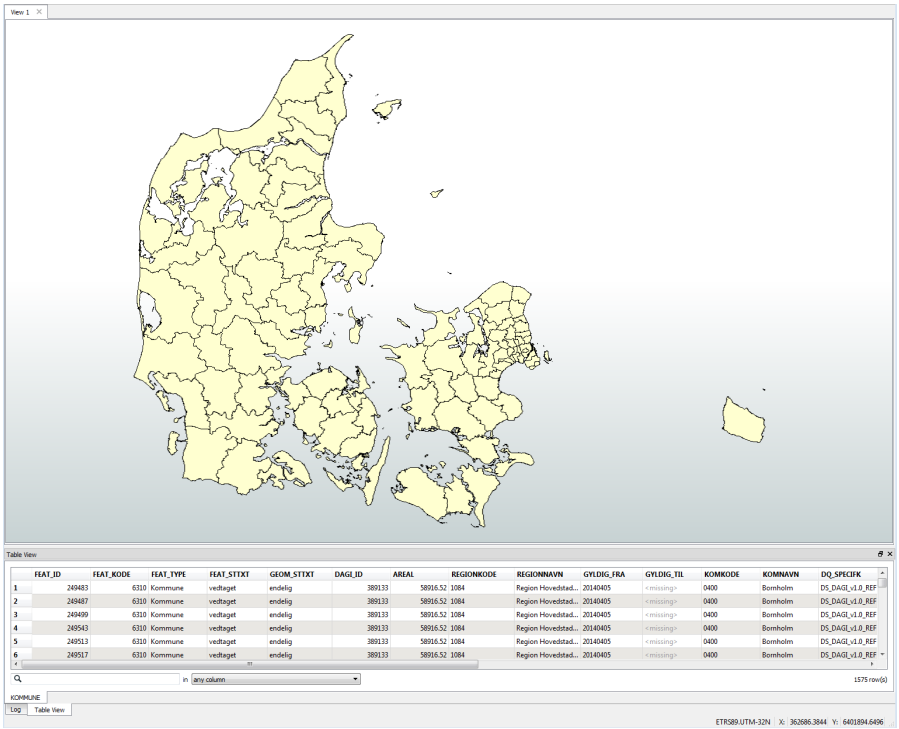

Figure 10: DAGI county division.

created using the lookup workflow which corresponds to the configuration in the XML file.

6.2 Data transformation using lookup tables in an ETL workflow

The workflow that is used at Kortforsyningen/Download to transform the DAGI data is an FME workflow relying heavily on a lookup table configured from an XML file. The file was created as part of a transition from a Python-based transformation routine to the FME routine used today, to save the correct transformation parameters. A simplified visualization of the lookup workflow can be seen in figure 11.

At the setup of the transformation routine the lookup table was created from the XML file and uploaded to a Postgres database. This is done every time the configuration of the dataset changes, not every time data is requested. There's a specific workflow designed to create the lookup tables from the specification in XML.

When a request for data is received the FME workflow is initiated. Using SQL it retrieves the base of an SQL query designed to get all the data relevant to a specific dataset, in this case DAGI. The query is then customized to get the data inside the bounding box and in the scale supplied by the user in the request. The generated query is then used to retrieve the data to be transformed from the Postgres DB .

Once the data is retrieved all the attribute types in the schema get hard-coded, both for FME's native file format as well as the destination format, in this case Mapinfo. This is accomplished using the FME transformer *SchemaMapper*, which uses the lookup table stored in the Postgres table to establish the mappings. This is done once for each format (FME and Mapinfo).

Once the attribute type mappings are correct the values of the attributes are run through the SchemaMapper. Some formats need to sort data into different attributes depending on their value, which is done in this step. For example one format may keep all poles in one attribute, while another format keeps telephone poles in one attribute and electricity poles in another. Just a with the attributes, this is done once for the FME native format and once for MapInfo.

In the final phase before writing the data a schema feature needs to be created. The writer uses the schema feature to properly structure the data when writing to the destination. This is done by reading the schema from the Postgres DB, processing it in FME to create a schema feature, and aggregating it to the data sent to the writer.

As a last step the data is written to the destination, in this case a file which is then supplied to the user.

6.3 Data transformation using ontologies in an ETL workflow

6.3.1 Method and terminology

Detailed method

A transformation of data from one schema to another in FME can be summarized in a few processes:

a The FME Workspace is created.

b The destination schema is defined/generated.

c The source schema is defined/generated.

d Rules for how the mapping between the schemas is created.

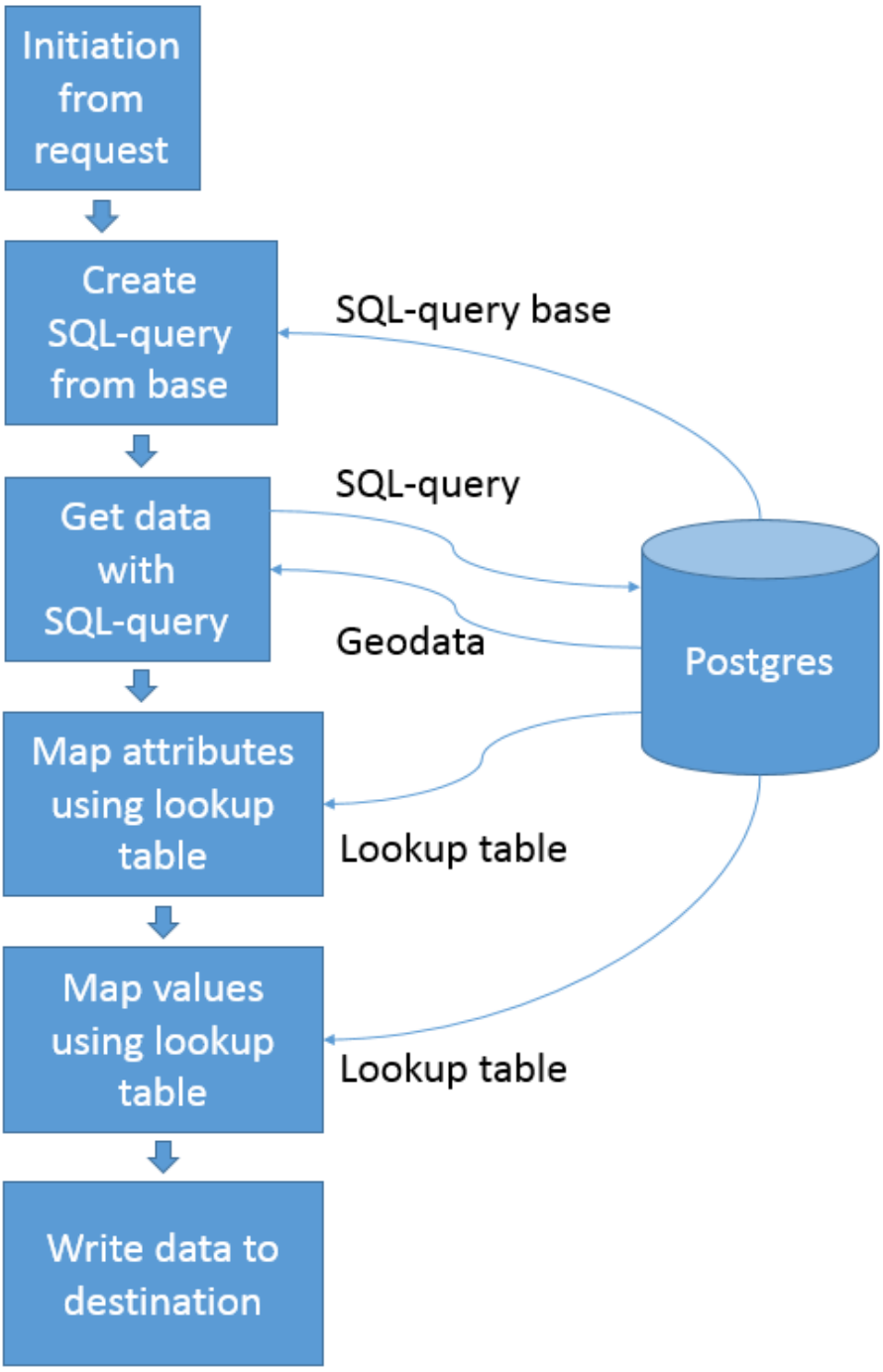

Figure 11: Simplified visualization of the lookup workflow.

e The workspace is executed and the result verified.

To achieve the above the following approach is used: the platforms used are FME and StarDog. FME to handle the actual transforming and communication and StarDog to store the ontologies. Ontologies describing the formats and and how they are linked are created. Once mapping rules have been generated using the ontologies they are loaded into FME and executed. The testdata used are DAGI from Kortforsyningen/Download by SDFE/GST in Denmark.

Linking rule set

There is no generally recognized definition of what a linking rule set is. In this report the term is used as defined in the V-Con projects Modelling and Linking Guide [6]. The definition being that a specification linking two ontologies is a linking rule set. Depending on the specific use of the rule set it is sometimes referred to as an *alignment ontology* or a *conversion rule set*.

A similiar term, *linked rules*, exists and is used in the work of Khandelwal *et al.* [21], but with the meaning of linking several logic rule sets which is used for other intelligent reasoning tasks together.

6.3.2 System architecture

The main part of the process is run in an FME workflow with the linking rule set hosted on a StarDog semantic graph database.

To start off this specific process, which can be seen in a simplified version in figure 12, the schema of the source file, in this case a PostGIS database table, is read by a FeatureReader transformer. The FeatureReader transformer retrieves the name, native data type and FME data type of each attribute. In the same transformer the actual data is also retrieved. The data and the schema are then sent on two different but parallel tracks.

The schema is sent to a custom Python script run by a transformer. The Python script iterates over the attributes and creates triples describing them, which is stored in a long string using the Turtle syntax. The string is written to a Turtle-file (.ttl) using the TempPathnameCreator to create a unique temporary file name and the AttributeFileWriter to write the actual file. The file is then loaded into a StarDog server using SystemCaller, a transformer which sends commands to the terminal. The command sent to the terminal is created in an AttributeCreator transformer.

A SPARQL query sent to to the database from another SystemCaller transformer retrieves the data needed to create the attributes suitable to the destination format, which is the name, native data type and FME data type. The result of the query is stored in a temporary text file, where the path once again is created by a TempPathnameCreator.

Using the result of the SPARQL-query another Python-script renames the attributes of the data to match the new schema. The new schema and the data are then sent to a transformer called FeatureMerger, which adds the schema to the data. The FeatureMerger transformer is also where our two parallel paths merge. The data with the new schema is sent to a writer and written to the destination file.

6.3.3 Implementation

The implementation consists of three main parts which have been developed: The FME workflow, the linking rule set and the Python scripts. The linking rule set and the Python scripts will be described in detail in this section. A detailed screenshot of the FME workflow can be seen in figure 15.

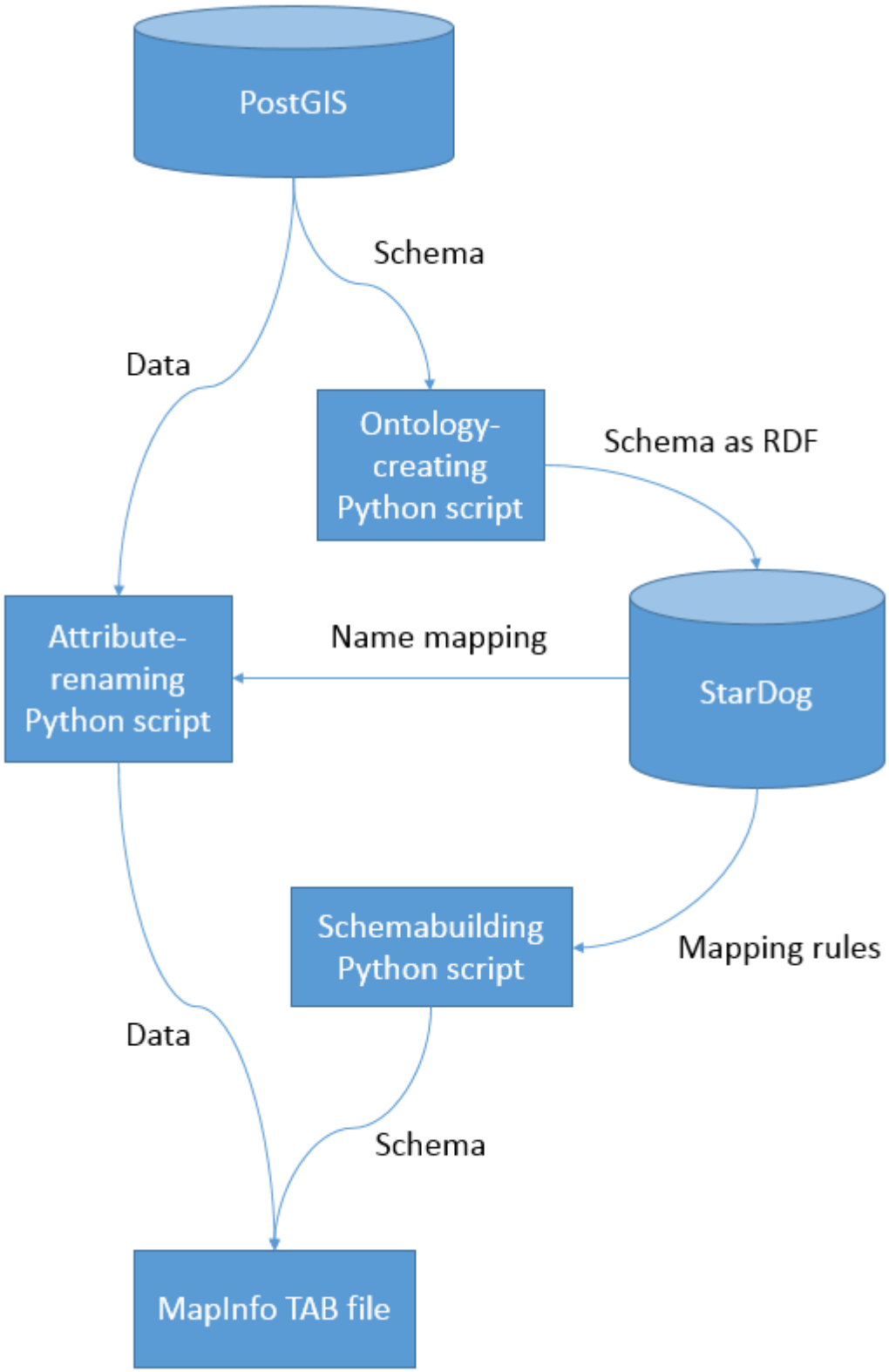

Figure 12: A simplification of the workflow.

Linking rule set

A Linking rule set is used to describe the connection between two data models, in this case PostGIS and MapInfo Tab. For a Linking rule set to work in this system, it should describe the attribute types of their respective formats. The info that should be included is the native datatype and what FME data type it maps to.

In this project the ontology creation tool Protégé was used to design the Linking rule set. To describe each format and its attribute types a number of classes were created: Attribute, AttributeType and FileFormat. The general structure of these classes can be seen in figure 13.

To link the data models each attribute has the property mapsTo, which indicates which FME native data type it corresponds to, e.g. the MapInfo integer is mapped to the FME int32 data type.

A number of named attributes were also created to describe the attributes present in the source dataset. These contained info about what data type they are and what their name in the source data was as well as what their name in the destination format should be according to the specification. Those properties are stored using the object property assertions (properties where the object is an IRI) dataType and belongsToFormat and the data property assertions (properties where the object is an RDF Literal) isNamed and mappingName. An example of a named attribute and its properties can be viewed in figure 14.

Attribute

This is an abstract class describing an attribute.

AttributeType

This subclass of Attribute is an abstract class describing an AttributeType. Its subclasses in turn are attribute types belonging to specific formats, such as PostGISAttributeType or FMENativeAttributeType. The subclasses has individuals describing its native data types, such as *post_char* which is the PostGIS char data type.

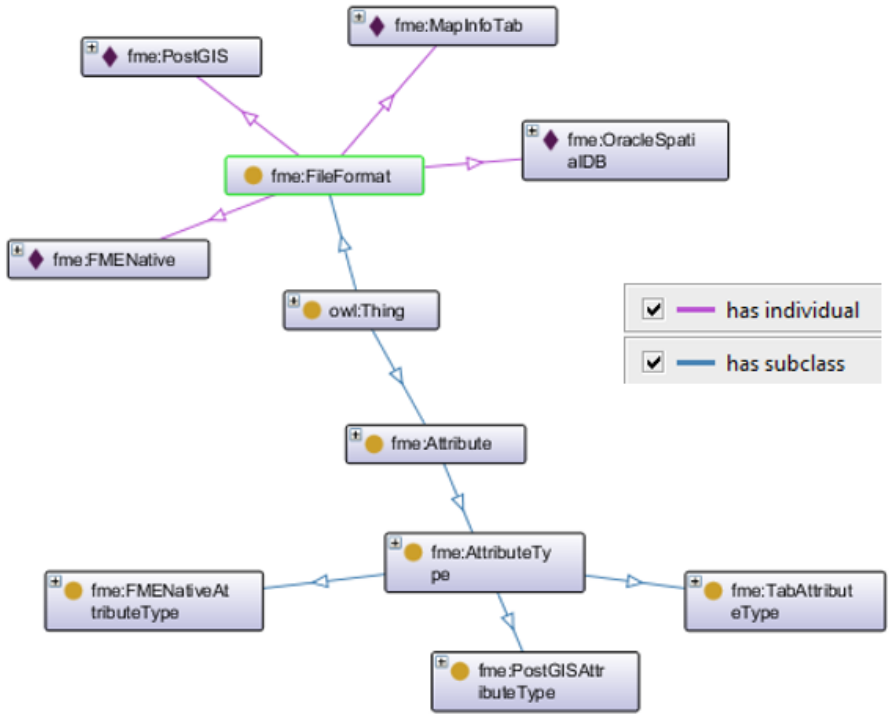

Figure 13: Class structure of ontology describing file formats.

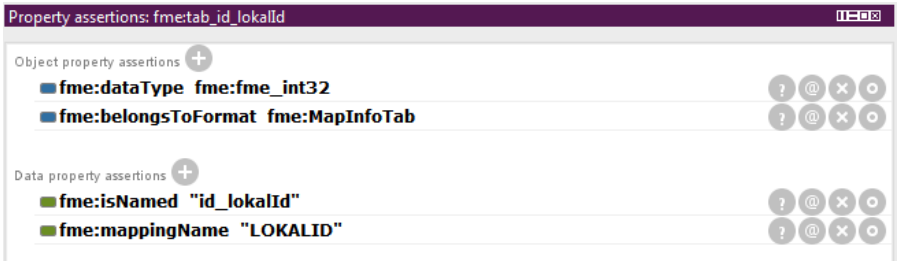

Figure 14: Example of a named attribute and its properties. Screen shot from Protégé.

FileFormat

This is an abstract class describing file formats. Its subclasses are all different file formats such as MapInfoTab or PostGIS.

MapsTo

MapsTo is an object property which describes a relation where the subject data type maps to an object which is a data type that belongs to another format.

BelongsToFormat

BelongsToFormat is an object property which describes a relation where the subject, a data type, belongs to the object, a file format.

DataType

DataType is an object property which describes the data type of a named attribute. The object should be a subclass of AttributeType, i.e. a data type belonging to a certain file format.

IsNamed

IsNamed is a data property which describes the name of an attribute in the source schema. This property is only used when describing named attributes. The object is an RDF Literal.

MappingName

MappingName is a data property which describes the name of an attribute in the destination schema. This property is only used when describing named attributes. The object is an RDF Literal.

HasValue

HasValue is a data property which describes the value of an attribute. This property is only used when describing format specific metadata attributes, such as *mapinfo_brush_background*. The object is an RDF Literal.

SPARQL, Python and commandline scripts

Using the transformers PythonCaller and SystemCaller in FME enables the execution of Python code and calls to the command terminal. By importing the FMEObjects Python package in the Python scripts features and attributes can be handled just like any other object.

The Python and commandline scripts described below handles the transfer of data to and from StarDog as well as creating the schema for the destination format.

Schema to RDF Python script

This custom Python script iterates through a list of FME attributes, filters out the ones that is part of the schema and constructs triples describing them. The information stored in the triples is the name of the attribute, its native data type and its FME data type. The triples is exported from the script as one long string, contained in the attribute ttl_out. It is then written to an RDF file.

Commandline scripts

Two command line scripts are run in the process. The first will insert the RDF with the schema triples into the StarDog database and the second one will send a query to get the information needed to create a schema for the destination format. The result will be stored in a temporary file.

SPARQL query

The SPARQL query is used to get the metadata required to build the schema for the destination file from the triple store. It looks like this:

```
PREFIX fme: <file:///C:/Users/SEGURO/Desktop/fmeobjects
    .ttl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns
    #>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?name (SAMPLE(?mappingName) AS ?mappingName) (
    SAMPLE(?native_type) AS ?native_type) (SAMPLE(?
    fme_type) AS ?fme_type) (SAMPLE(?value) AS ?value)
WHERE {
        ?attribute fme:isNamed ?name .
        ?attribute fme:mappingName ?mappingName .
        ?attribute fme:dataType ?fme_type .
        ?attribute fme:belongsToFormat fme:MapInfoTab .
        OPTIONAL { ?attribute fme:hasValue ?value } .
        ?fme_type fme:mapsTo ?native_type .
        ?native_type fme:belongsToFormat fme:MapInfoTab
} GROUP BY ?name
```

The first part of the query is the prefixes. These point to the commonly used ontologies such as RDFS, OWL and XSD, as well as to itself (fme). The select clause selects the name, new name (called mappingName in the query), destination format data type (native_type), FME format data type (fme_type) and value. By using the OPTIONAL pattern, ?value is only included if the attribute actually has a value, which only is the case if the attribute is a format attribute. Since the query groups by name, all the other selected variables need the use of the SAMPLE function to work properly. The correct destination format type is retrieved by querying for the FME data type of the attribute, then selecting the destination format data type linked to that FME data type.

SPARQL query result to schema Python script

This Python script takes the result of a SPARQL query and creates a schema for the destination file. This also utilizes the FMEObjects Python package, in this case to handle the creation of the destination schema.

```
feature.setAttribute('attribute{'+str(index)+'}.name',
    name)
```

```
feature.setAttribute('attribute{'+str(index)+'}.
    fme_data_type', fme_type)
feature.setAttribute('attribute{'+str(index)+'}.
    native_data_type', native_type)
```

In the code excerpt above the critical part of the Python script can be seen; the part that builds the new schema. Name, fme_type and native_type are retrieved from the SPARQL result using regex.

Data attribute renamer Python script

This Python script creates a table mapping the old attribute names to the new attribute names using the SPARQL query results. It then renames the attributes of the data to match the new schema.

```
for index, name in enumerate(attr_names):
  attr_value = feature.getAttribute(name)
  new_name = mapping_dictionary.get(name, False)
  if not new_name:
    continue

  feature.removeAttribute(name)
  feature.setAttribute(new_name, attr_value)
```

In the code excerpt above is the part that do the renaming. This is done by retrieving the current value of the attribute, getting the new name from the mapping dictionary, removing the attribute with the old name and then create a new attribute with the new name but the same value as the old one. If no new name is found the old attribute won't be removed and no new attribute created.

The complete workflow in FME can be seen in figure 15.

6.4 Comparison

The requirement used in the development of the ontology-approach is that the resulting data should be identical to the data produced by the lookup workflow. What is evaluated is:

1. Attributes are to be named correctly.
2. Attributes are to be given correct data types.
3. Correct values are to be produced.
4. Correct geometries are to be produced.
5. A correct amount of data are to be produced.

The data produced by the two workflows are compared. The primary format used in the comparison is Mapinfo TAB since it's a GIS format which can handle a number of complex aspects of data storage such as: Having multiple tables/layers in one file, storing drawing styles (symbology used, colour of polygon etc) as well as storing different data types (integers, strings, floats etc).

In the evaluation the main aspects that the data are compared by are:

1. Attribute names, or user attributes. Commonly referred to as columns.
2. Format attributes, i.e. attributes that are metadata with info about color fills, coordinate systems or geometry types.
3. Objects, that the actual data with features and geometries is correct and correctly formatted.

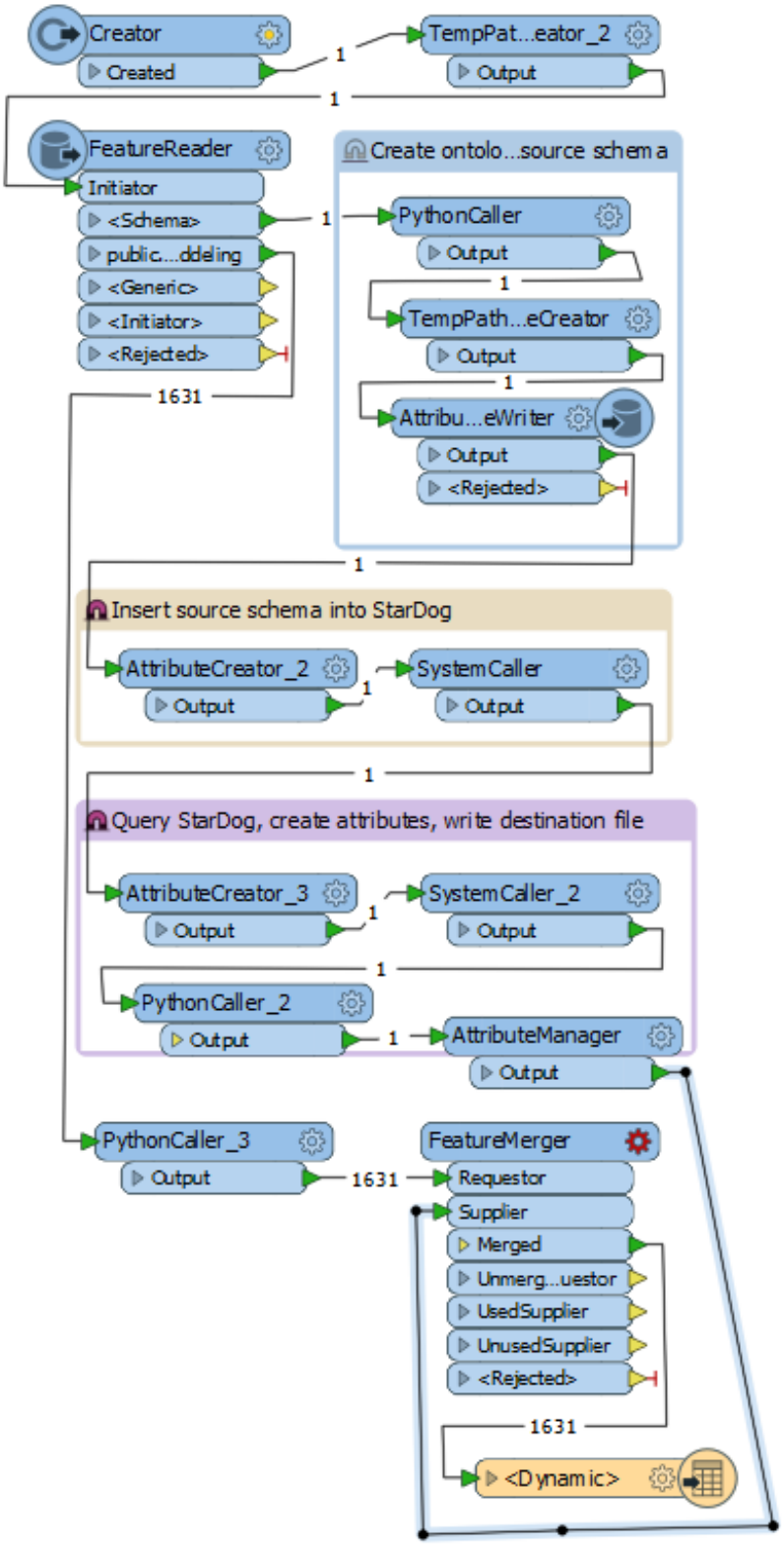

Figure 15: The ontology approach workflow in FME.

6.4.1 Comparison of user attributes

The proper structure, number and data type of attributes are correct if the mapping of the attributes in the schema has been done properly.

In figure 16 the attributes of the data transformed by the lookup workflow and ontology workflow respectively can be seen. The two tables are almost identical with the exception of the ontology workflow having an extra attribute, *sys_deleted*. The extra attribute is an attribute related to FME which proved hard to remove. In both cases the attribute names and data types are correct.

6.4.2 Comparison of format attributes

The format attributes, in this case all the attributes whose name start with *fme_* or *mapinfo_*, determines the looks of the graphics in FME and the destination format. These attributes can be set in the schema just as the user attributes, but it's not always necessary since FME has some default values for different formats.

As can be seen in 16 these format attributes can all be accounted for. The data types of the attributes match and are correct.

6.4.3 Comparison of objects

To compare the objects a sample object is examined closer. The values and properties of this object can be seen in figure 18. The geometries of the object can be seen in figure 17.

The geometries all look correct compared to the source data. The number of vertices also match between the geometries. They also have the same coordinate system.

Result using lookup tables

Attributes (36)
DAGIID (string)
DATA_SPEC (encoded: cp1252)
fme_color (string)
fme_fill_color (string)
fme_geometry (string)
fme_type (string)
FOR_HAEND (encoded: cp1252)
FOR_OMR (encoded: cp1252)
FOR_PROCES (encoded: cp1252)
GEOMSTATUS (encoded: cp1252)
KOMMUEKODE (encoded: cp1252)
LANDEKODE (encoded: cp1252)
LAU1VAERDI (encoded: cp1252)
LOKALID (string)
mapinfo_brush_background (32 bit unsigned integer)
mapinfo_brush_foreground (32 bit unsigned integer)
mapinfo_brush_pattern (32 bit integer)
mapinfo_brush_transparent (string)
mapinfo_centroid_x (64 bit real)
mapinfo_centroid_y (64 bit real)
mapinfo_pen_color (32 bit unsigned integer)
mapinfo_pen_pattern (32 bit integer)
mapinfo_pen_width (32 bit integer)
mapinfo_type (string)
NAMESPACE (encoded: cp1252)
NAVN (encoded: cp1252)
OBJECTID (string)
REDIGGSRET (encoded: cp1252)
REG_LOKID (string)
REGIST_AKT (encoded: cp1252)
REGIST_FRA (string)
SKALA (encoded: cp1252)
STATUS (encoded: cp1252)
U_KOM_IND (encoded: cp1252)
VIRK_AKT (encoded: cp1252)
VIRK_FRA (string)

Result using ontologies

Attributes (37)
DAGIID (string)
DATA_SPEC (encoded: cp1252)
fme_color (string)
fme_fill_color (string)
fme_geometry (string)
fme_type (string)
FOR_HAEND (encoded: cp1252)
FOR_OMR (encoded: cp1252)
FOR_PROCES (encoded: cp1252)
GEOMSTATUS (encoded: cp1252)
KOMMUEKODE (encoded: cp1252)
LANDEKODE (encoded: cp1252)
LAU1VAERDI (encoded: cp1252)
LOKALID (string)
mapinfo_brush_background (32 bit unsigned integer)
mapinfo_brush_foreground (32 bit unsigned integer)
mapinfo_brush_pattern (32 bit integer)
mapinfo_brush_transparent (string)
mapinfo_centroid_x (64 bit real)
mapinfo_centroid_y (64 bit real)
mapinfo_pen_color (32 bit unsigned integer)
mapinfo_pen_pattern (32 bit integer)
mapinfo_pen_width (32 bit integer)
mapinfo_type (string)
NAMESPACE (encoded: cp1252)
NAVN (encoded: cp1252)
OBJECTID (string)
REDIGGSRET (encoded: cp1252)
REG_LOKID (string)
REGIST_AKT (encoded: cp1252)
REGIST_FRA (string)
SKALA (encoded: cp1252)
STATUS (encoded: cp1252)
sys_deleted (string)
U_KOM_IND (encoded: cp1252)
VIRK_AKT (encoded: cp1252)
VIRK_FRA (string)

Figure 16: Attributes of the transformed datasets.

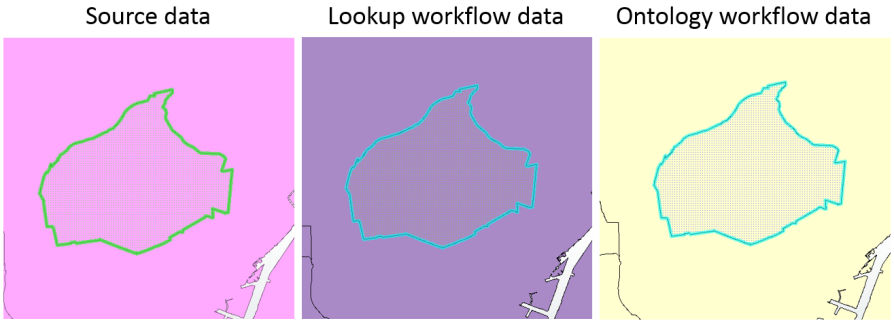

Figure 17: The sample geometry in the different datasets.

When examining the data of the object, most values look OK with the exception of a few differences. The values of the attributes DAGIID, LOKALID, fme_fill_color and Feature Type differ between the datasets. When comparing these to the source data of each attribute we can see that something is amiss. The source value of DAGIID is 248811, which is the same as the result of the ontology approach but not the lookup approach, as can be seen in 19.

The source value of LOKALID is a string of letters and numbers, possibly hexadecimal, base64 or something similar. The fact that the source data is a string and the destination data should be an integer forces it to become -9999 in the ontology approach. In the lookup approach the value for LOKALID is 389104, possibly by translating the string, or by using the value of dagiid instead.

It is also noted that the source data contains 1631 objects (i.e. geometries with attributes) but the data transformed in the ontology-driven process

Property	Value	Value
Feature Type	KOMMUNE	public.pdb_rc00001_kommune...
Coordinate System	ETRS89.UTM-32N	ETRS89.UTM-32N
Dimension	2D	2D
Number of Vertices	871	871
Min Extents	719497.14355528925, 6174592.3...	719497.14355528925, 6174592.3...
Max Extents	723630.96670947154, 6178106.5...	723630.96670947154, 6178106.5...
Attributes (37)		
DAGIID (string)	389104	248811
DATA_SPEC (encoded: cp1252)	DAGI Objektspecifikation versi...	DAGI Objektspecifikation versi...
fme_color (string)	0,0,0	0,0,0
fme_fill_color (string)	0.662745098039216,0.5411...	1,1,0.815686274509804
fme_geometry (string)	fme_polygon	fme_polygon
fme_type (string)	fme_area	fme_area
FOR_HAEND (encoded: cp1252)	matrikulærAjourføring	matrikulærAjourføring
FOR_OMR (encoded: cp1252)	DAGI	DAGI
FOR_PROCES (encoded: cp1252)	DAGISYSeditor	DAGISYSeditor
GEOMSTATUS (encoded: cp1252)	endelig	endelig
KOMMUEKODE (encoded: cp1252)	0147	0147
LANDEKODE (encoded: cp1252)	DK	DK
LAU1VAERDI (encoded: cp1252)	147	147
LOKALID (string)	248811	-9999
mapinfo_brush_background (32 bit unsigned integer)	16777214	16777215
mapinfo_brush_foreground (32 bit unsigned integer)	11111111	16777168
mapinfo_brush_pattern (32 bit integer)	2	2
mapinfo_brush_transparent (string)	false	false
mapinfo_centroid_x (64 bit real)	721564.05513238045	721564.05513238045
mapinfo_centroid_y (64 bit real)	6176349.4519338896	6176349.4519338896
mapinfo_pen_color (32 bit unsigned integer)	0	0
mapinfo_pen_pattern (32 bit integer)	2	2
mapinfo_pen_width (32 bit integer)	1	1
mapinfo_type (string)	mapinfo_region	mapinfo_region
NAMESPACE (encoded: cp1252)	http://data.gov.dk/dagi	http://data.gov.dk/dagi
NAVN (encoded: cp1252)	Frederiksberg	Frederiksberg
OBJECTID (string)	295353	295353
REDIGGSRET (encoded: cp1252)	GST	GST
REG_LOKID (string)	-9999	-9999
REGIST_AKT (encoded: cp1252)	DAGI-operatør	DAGI-operatør
REGIST_FRA (string)	20140904085341.000	20140904085341.000
SKALA (encoded: cp1252)	1:10.000	1:10.000
STATUS (encoded: cp1252)	vedtaget	vedtaget
sys_deleted (string)		false
U_KOM_IND (encoded: cp1252)	false	false
VIRK_AKT (encoded: cp1252)	Geodatastyrelsen	Geodatastyrelsen
VIRK_FRA (string)	20140904085334.000	20140904085334.000

Figure 18: A sample value of the transformed datasets.

Attribute name (destination)	Lookup	Ontology
LOKALID	248811	-9999
DAGIID	389104	248811
Attribute name (source)	Source Data	
id_lokalid	ede4a799-63a7-24a4-e044-00144f3ead67	
dagiid	248811	

Figure 19: Table of values differing between datasets.

contains 1630 objects. For an unknown reason one object is missing. The object in question has objectid 248784, is an isolated polygon and seems to be some form of breakwater in a harbour in Copenhagen. No suspicious or odd qualities or values can be found upon closer examination of the object. The missing geometry can be seen in figure 20, alongside the source data with the geometry intact.

To investigate where exactly the problem that causes object 248784 to disappear in the transformation process is situated the following method can be used. Create a workflow in FME which converts the data from PostGIS to MapInfo TAB without doing any specific mappings or similar. If the object is still missing in the resulting data the ontology-driven part of the process can't be the cause of the problem, it's obvious that it's caused by FME. If however the object can be found in the converted data, the ontology-driven part of the process needs to be examined closer to determine what's causing the problem.

It turned out to be the mechanics of the writer that caused the demise of object 248784. A specific setting for the writer, which is a new addition in the latest beta of FME, didn't work exactly as expected. Instead of reading the schema feature and writing the first object (in this case object 24874) it read the schema and discarded the first object. This was hotfixed by sending the first object twice.

6.4.4 Amount of manual interaction

The amount of manual interaction involved in creating the two workflows is not easily measurable, especially since the lookup workflow was created before the study was started. Due to this the parts probable to require the biggest amount of manual interaction in each approach, assuming the required infrastructure is in place, will be listed.

Lookup approach

1. Customize the SQL query to fetch the correct data.
2. Create or modify the XML file containing the configuration, which in turn will create the lookup table.

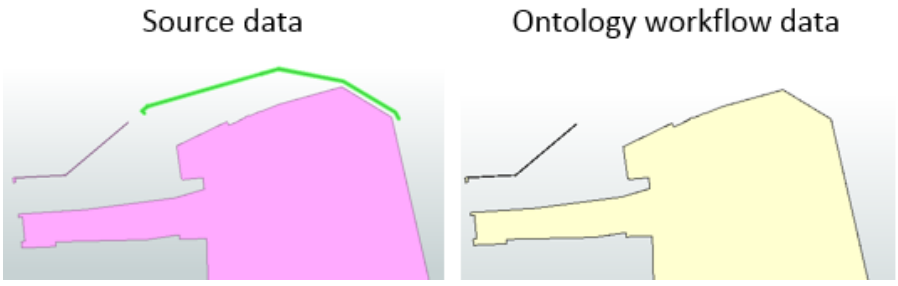

Figure 20: The mystery of the missing breakwater geometry.

Ontology approach

1. Creating ontologies describing the file formats and their data types.
2. Building the linking rule set.

7 Discussion

The main aim of this project is to study the transformation problem of geodata and determine whether using semantic web technology could be a possible solution. This has been achieved and the aim is considered met. The practical test of using an ontology-driven ETL process was successful. The process developed in the test is quite basic and not polished in any way, but it is an important and valuable step towards developing a proper ontology-driven ETL process. The successful test is proof that such a process is not only possible, but also plausible.

The hardest part during the development of the test was designing and building the models describing the different formats, their attribute types and data types. Determining what was relevant and essential information that should be included in the model was at first quite hard, but acquiring more knowledge regarding how the intermediate format in FME is utilized during the transformation made it easier.

Creating the new schema feature and making FME use it properly together with the data to transform was also a hard part to figure out, but quite easy once solved. It was a problem that could be easily solved with some experience of working with FME.

The ontology-driven process used in this project bears a heavy resemblance to the framework process proposed by Zhang and Wang [40] and Ludäscher and Bowers [7]. The basic ideas in those frameworks as well as in this project is the same. Metadata about the structure of the source and destination models are extracted as ontologies, fed to a reasoning engine which establishes the transformation rules which is returned to the transformation process. While this project was developed independently from Zhang and Wangs proposal, the process tested in this project is more or less a test of Zhangs proposed framework process, if a simple one due to the lack of reasoning and in a geospatial context.

One of the main strengths of using an ontology-driven process is the potential for utilizing computer reasoning. By using reasoning the queries could probably become less complex or at least more generic. In this project no reasoning is used, so the query used was customized to fit the models and linking rule set. For reasoning to work properly, both the models and the linking rule set need to be more complex and descriptive.

StarDog was chosen to be the triple store used in this project. This is mostly due to a mentors previous experience with it, but also due to the fact that StarDog is compatible with multiple platforms, can be interacted with in a multitude of ways and also can be interfaced to many different code languages. In an effort to continue on this project, in the form of building plug-ins or using it in a commercial solution for data transformation, a versatile triple store is an advantage. The one drawback of StarDog is that it isn't open source, so any larger applications with it would need a license.

Protégé was chosen as the ontology editor used in this project. It has a good reputation online as a complete ontology editor with plenty of documentation and tutorials. There's also a multitude of plugins which extends the functionality of Protégé in a variety of ways. Another ontology editor that was briefly considered was OntoEdit.

FME was chosen as the ETL tool to implement the ontology approach in. The amount of experience with FME possessed by a mentor and colleagues made this choice easy. The fact that FME supports a large amount of formats makes any future commercial endeavours involving the process able to reach a wider audience. FME is also the ETL tool used in the lookup-approach, which makes comparison easier since the potential discrepancies caused by using different ETL software is non-existent.

When comparing the lookup approach to the ontology approach some things need to be mentioned. The lookup process studied in this project is probably more complex than needed, since it's a transition or translation from an earlier process using homebuilt Python solutions instead of an ETL tool. Also, what the ontology approach in the end resulted in was more or less a lookup table. The ontology approach also didn't do any attempts at implementing a bounding box selection or something similar, since it's not needed in the requirements or is a matter which is connected to the transformation problem studied.

The fact that the data on the test server didn't match the specification XML also caused some trouble. All the integer fields that became -9999 was an ID-string which got typecast to integer and thus became -9999. One could argue that this is a problem in the transformation process, but on the other hand this typecasting also shows that the schema got the correct data types. The mis-match between server and specification is probably also the cause for the differing LOKALIDs.

Why object 248784 went missing during the transformation process is somewhat of a mystery. It's not an odd geometry, or at least not odder than some similar ones in its close proximity. It has reasonable values and nothing that stands out about it at all.

As mentioned above the lookup process could be made simpler, which also is one of the strengths of using the lookup approach: it's simple. It's easy to understand and simple to set up; a lookup table can be created and kept in an excel sheet or CSV-file and read by the SchemaMapper transformer. To set up a lookup table for each dataset and format in the Kortforsyningen/Download case would probably be quite time consuming though, but not particularly complex.

Setting up the ontology approach is somewhat more complex, since you need to create the models of the formats as well as the linking rule set, and set up the triple store/graph database to store it all and run queries against. It also takes some time getting used to working and thinking with triples, since the concept is quite different from the usual database tables.

One way the ontology approach could be improved is by creating a default linking rule set between formats. That way minor changes could be made to the rule set to suit new datasets when setting up the workflow, without requiring a completely new linking rule set each time. A default linking ruleset could even describe links between all the relevant formats, there's actually no reason for it to just describe the relationship between two formats. This could possibly even allow one workflow to read data from multiple sources with different formats.

Another possibility with a default linking rule set is to set up the reasoning to prioritize small custom made case-specific linking rule sets and only use the default mappings to map parts not covered by the case-specific set.

The default linking rule set could also be published online, which would make it linkable to from anywhere. This would be useful since the workflow in its current state feeds the triple store with an RDF that describes the source schema and it is assumed that the triple store already contains the linking rule set. Once used, the triple store needs to be deleted if another datasets is to be transformed. Hosting the default rule linking set online would remove the step of uploading it to the triple store each time a new database is set up and would also allow a centralized control of it.

Another way to simplify the usage of the ontology approach could be replacing the Python scripts and system callers with proper FME plugins. A prime example would be the part of the workflow where the schema is converted to RDF, written to a file and that file is uploaded to the triple store. All of that could be done in a plugin instead, which would make the workflow more intuitive to use and easier to understand. Sending SPARQL-queries to a triple store would also be a useful plug-in. It could also be used in other workflows which intend to utilize the semantic web technologies.

The ontology approach developed and tested in this project uses the semantic web technologies to find the mappings between the attributes of different formats and uses it to create a data schema usable in FME. This is one way to use the technologies, another is to also transform the data into RDF, and that way create a federated data model. Storing geodata as RDF also makes it possible to query it using regular SPARQL, or preferably GeoSPARQL which allows spatial functions in queries. RDF could also be used as an intermediate format in a transformation process. GeoSPARQL could then be used to filter the data or select specific data before transforming to the destination format.

Storing geodata as RDF also means that it can be linked to the open data available on the internet. The probably largest and most well known open geodata is the Linked Geodata project, as mentioned in 5.2. One could argue that linking to open geodata could give a small dataset some context.

While more and more data is released by governments around the world (e.g. Datafordeleren in Denmark, data.gov and data.gov.uk) it seems that publishing it as RDF hasn't quite caught on just yet. Some governmental agencies and smaller institutions (such as FAO) may publish their data as RDF, but on the large scale it's not a common phenomenon. In that aspect the community-sourced knowledge bases such as Wikipedia and OpenStreetMap is ahead, with DBPedia and the Linked Geodata project being their Semantic Web counterparts. The fact that the open source communities is ahead could be because governmental institutions tend to be large organizations which require both time and effort to adapt to new technologies. Some governmental agencies are catching up though, which the USGS is an excellent example of, but then again publishing geographic data is what they do. Since the federal US data is published without any restrictions or copyright there's nothing that stops the community to convert it to RDF and republish it, if need be.

One reason that the publishing of geodata which is compatible with the semantic web has taken so long is the fact that there's been no consensus or standard for describing geospatial information until GeoSPARQL was released in 2011. Even then it was just the standard that was released and it took another while before any kind of working implementation was released to the public. Before GeoSPARQL the proposed vocabularies all had flaws. OGC Geo Basic was very simple and unable to model advanced geometries, GeoOnion had a very unorthodox take on distances with it's onion-like distance layers, and NeoGeo represented each point in a geometry as an object which led to big datasets and made comparing geometries unnecessarily inefficient [2]. GeoSPARQL on the other hand is based on already well established standards such as Simple Features and SPARQL which makes transitioning to it smoother.

One project that might become very valuable in the creation and publishing of open geodata is GeoKnow. The tools included in the GeoKnow toolbox, especially Sparqlify and TripleGeoService, lowers the amount of time, effort and knowledge needed to create and publish geodata. Since the toolbox also includes a triple store, RDF verifying software as well as data explorers for smartphones it more or less covers the whole life cycle of geospatial data. Since all the tools are developed to work together, it's a complete package for publishing geodata. One may argue that if it gets easier to publish geodata, more people and organization will consider doing it.

If more geodata is stored and published as RDF, it might take a while for GIS software to catch up and offer compatibility. The process developed in this project might be a base for a solution to that problem. While it doesn't currently actually use RDF as intermediate format, the general process of extracting schema, generating transformation rules and doing the transformation could probably be applicable in that scenario too. If the process is extended, as discussed earlier, to use RDF as a intermediate

format, it could be a solution in making GIS software compatible with linked geodata.

The GEISER project is taking the next step in linked geodata, combining linked geodata from a multitude of sources to create some sort of amalgamation between linked data and Big Data [16]. The ambition to automatically sort and filter data, and assign it semantics is very complex. If they manage to live up to their ambitions it would be very impressive and probably beneficial to both fields of data science.

One could argue that projects and systems such as GEISER will become more common in the future. Enormous amounts of data and geodata is being produced worldwide on a daily basis. It would be unwieldy and unthinkable to sort and classify all this data manually. This is where GEISER and the likes could come in and bring some form of order and structure to the massive amount of data.

8 Conclusions

One aim of this project was verifying the process of using ontologies and RDFs in the transformation project. That aim was achieved and the result was an FME workflow utilizing semantic web technologies to map attributes which successfully transformed data to the specified format. A comparison between two approaches to using ETL workflows was done and resultwise they both work well, with some minor oddities turning up. To conclude: using semantic web technologies when transforming data in an ETL process is possible.

The next step in developing the ontology-approach would be to create more complex models representing the formats and a more complex linking rule set which would allow reasoning to a larger degree. Creating a default linking rule set is another way to further advance the process.

Creating readers and writers for the formats related to the semantic web technologies for ETL software would probably increase the interoperability between the geospatial semantic web and common GIS software.

Compared to the lookup method the ontology process is more complex to set up, delivers similar results (with a few exceptions) and requires considerable knowledge on the users part. Even though the lookup solution studied was quite complex the ontology approach is at a serious disadvantage.

While the ontology approach might become more viable with further development and research this study does not provide sufficient evidence that ontology-driven ETL solutions decreases the amount of manual interaction needed compared to lookup-driven ETL solutions.

Hopefully the tools developed and studied in this project can be used in further development and projects.

Bibliography

[1] Rob Andrews. *New Ordnance Survey Linked Data service proving popular with developers*. July 17, 2013. URL: http://www.directionsmag.com/pressreleases/new-ordnance-survey-linked-data-service-proving-popular-with-developers/340085.

[2] Robert Battle and Dave Kolas. "Enabling the geospatial semantic web with parliament and geosparql". In: *Semantic Web* 3.4 (2012), pp. 355–370.

[3] Robert Battle and Dave Kolas. "Geosparql: enabling a geospatial semantic web". In: *Semantic Web Journal* 3.4 (2011), pp. 355–370.

[4] Christian Becker and Christian Bizer. "Exploring the geospatial semantic web with dbpedia mobile". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7.4 (2009), pp. 278–286.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (2001), pp. 28–37.

[6] Michel Böhms, Lars Wikström, Olle Bergman, and Bart Luiten. *V-Con Modelling and Linking Guide for Semantically-enhanced Linked Data (SeLD)*. 2016.

[7] Shawn Bowers and Bertram Ludäscher. "An ontology-driven framework for data transformation in scientific workflows". In: *International Workshop on Data Integration in the Life Sciences*. Springer. 2004, pp. 1–16.

[8] Kurt Buehler and Lance McKee. *The OpenGIS Guide: Introduction to Interoperable Geoprocessing: Part I of the Open Geodata Interoperability Specification (OGIS)*. Open GIS Consortium, Incorporated, 1996.

[9] Open Geospatial Consortium et al. *The Importance of Going "Open", OGC White Paper, 8p*. 2005.

[10] World Wide Web Consortium. *Ontologies - W3C*. Sept. 8, 2016. URL: https://www.w3.org/standards/semanticweb/ontology.html.

[11] World Wide Web Consortium. *RDF Schema 1.1*. Sept. 19, 2016. URL: https://www.w3.org/TR/rdf-schema/.

[12] Simon Cox, Adrian Cuthbert, Paul Daisey, John Davidson, Sandra Johnson, Edric Keighan, Ron Lake, Marwa Mabrouk, Serge Margoulies, Richard Martell, et al. "OpenGIS® Geography Markup Language (GML) Implementation Specification, version". In: (2002).

[13] Styrelsen for Dataforsyning og Effektivisering. *SDFE - Om os*. Oct. 4, 2016. URL: http://sdfe.dk/om-os/.

[14] Catherine Dolbear and Glen Hart. "Ontological Bridge Building-Using Ontologies to Merge Spatial Datasets." In: *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*. 2008, pp. 15–20.

[15] George HL Fletcher. "An algebra for basic graph patterns". In: *workshop on Logic in Databases, Rome, Italy*. 2008.

[16] GEISER. *GEISER Platform - GEISER*. Dec. 12, 2016. URL: http://www.projekt-geiser.de/en/geiser-platform/.

[17] Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, and Karl-Heinz Häfele. "OGC city geography markup language (CityGML) encoding standard V2. 0". In: *OGC Doc* 12-019 (2012).

[18] Glen Hart and Catherine Dolbear. *Linked Data: A Geographic Perspective*. 1st ed. Boca Raton, Florida: Taylor & Francis Group, 2013. ISBN: 9781439869956.

[19] E Hietanen, L Lehto, and P Latvala. "Providing Geographic Datasets as Linked Data in Sdi". In: *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2016), pp. 583–586.

[20] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. "OWL 2 web ontology language primer". In: *W3C recommendation* 27.1 (2009), p. 123.

[21] Ankesh Khandelwal, Ian Jacobi, and Lalana Kagal. "Linked rules: Principles for rule reuse on the web". In: *International Conference on Web Reasoning and Rule Systems*. Springer. 2011, pp. 108–123.

[22] Graham Klyne, Jeremy J Carroll, and Brian McBride. "RDF 1.1 concepts and abstract syntax". In: *W3C Recommendation* 25 (2014).

[23] Wolfgang Kresse and David M Danko. *Springer handbook of geographic information*. Springer Science & Business Media, 2012.

[24] Jens LEHMANN, Spiros ATHANASIOU, Andreas BOTH, Alejandra GARCIA ROJAS, Giorgos GIANNOPOULOS, Daniel HLADKY, Jon Jay LE GRANGE, Axel-Cyrille NGONGA NGOMO, Mohamed Ahmed SHERIF, Claus STADLER, et al. "The GeoKnow Handbook". In: (2015).

[25] John Liagouris, Nikos Mamoulis, Panagiotis Bouros, and Manolis Terrovitis. "An effective encoding scheme for spatial RDF data". In: *Proceedings of the VLDB Endowment* 7.12 (2014), pp. 1271–1282.

[26] Joshua Lieberman, Raj Singh, and Chris Goad. *W3C Geospatial Vocabulary, W3C Incubator Group Report 23 October 2007*.

[27] Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Science & Systems*. 4th ed. John Wiley & Sons, Inc., 2015. ISBN: 0471735450.

[28] Deborah L McGuinness, Frank Van Harmelen, et al. "OWL web ontology language overview". In: *W3C recommendation* 10.10 (2004), p. 2004.

[29] Lance McKee. *OGC History (detailed)*. Sept. 23, 2016. URL: `http://www.opengeospatial.org/ogc/historylong`.

[30] Claudine Métral, Roland Billen, Anne-Francoise Cutting-Decelle, and Muriel van Ruymbeke. "Ontology-based approaches for improving the interoperability between 3D urban models". In: *FormaMente* 1.2 (2010), pp. 85–111.

[31] Mark A Musen. "The Protégé project: a look back and a look forward". In: *AI matters* 1.4 (2015), pp. 4–12.

[32] Matthew Perry and John Herring. "OGC GeoSPARQL-A geographic query language for RDF data". In: *OGC Implementation Standard. Sept* (2012).

[33] Jeffrey Pollock. *Semantic Web for Dummies*. Indianapolis, Indiana: Wiley Publishing Inc., 2009.

[34] Bastian Quilitz and Ulf Leser. "Querying distributed RDF data sources with SPARQL". In: *European Semantic Web Conference*. Springer. 2008, pp. 524–538.

[35] OGCGML RWG. "ISO/TC 211/WG 4/PT 19136 Geographic information–Geography Markup Language (GML)". In: (2004).

[36] Juan Martín Salas, Andreas Harth, Barry Norton, Luis M Vilches, Alexander De León, John Goodwin, Claus Stadler, Suchtith Anand, and Dominic Harries. *NeoGeo vocabulary: defining a shared RDF representation for GeoData*. Dec. 6, 2016. URL: `http://geovocab.org/doc/neogeo.html`.

[37] Bruce Spencer and Sandy Liu. "Inferring data transformation rules to integrate semantic web services". In: (2004).

[38] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. "Linkedgeodata: A core for a web of spatial open data". In: *Semantic Web* 3.4 (2012), pp. 333–354.

[39] Chuanrong Zhang, Weidong Li, and Tian Zhao. "Geospatial data sharing based on geospatial semantic web technologies". In: *Journal of Spatial Science* 52.2 (2007), pp. 35–49.

[40] Zhuolun Zhang and Sufen Wang. "A Framework Model Study for Ontology-Driven ETL Processes". In: *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2008, pp. 1–4.

[41] Peisheng Zhao, Liping Di, Genong Yu, Peng Yue, Yaxing Wei, and Wenli Yang. "Semantic Web-based geospatial knowledge transformation". In: *Computers & Geosciences* 35.4 (2009), pp. 798–808.

Institutionen av naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser) i geografisk informationsteknik. Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 2010. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers och via Geobiblioteket (www.geobib.lu.se)

Serie examensarbete i geografisk informationsteknik

1. *Patrik Carlsson och Ulrik Nilsson* (2010) Tredimensionella GIS vid fastighetsförvaltning
2. *Karin Ekman och Anna Felleson* (2010) Att välja grundläggande karttjänst – Utveckling av jämförelsemodell och testverktyg för utvärdering
3. *Jakob Mattsson* (2011) Synkronisering av vägdatabaser med KML och GeoRSS - En fallstudie i Trafikverkets verksamhet
4. *Patrik Andersson and Anders Jarisoo* (2011) Effective use of open source GIS in rural planning in South Africa
5. *Nariman Emamian och Martin Fredriksson* (2012) Visualiserig av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation
6. *Gustav Ekstedt and Torkel Endoff* (2012) Design and Development of a Mobile GIS Application for Municipal Field Work
7. *Karl Söderberg* (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management
8. *Viktoria Strömberg* (2012) Volymberäkning i samhällsbyggnadsprojekt
9. *Daniel Persson* (2013) Lagring och webbaserad visualisering av 3D-stadsmodeller - En pilotstudie i Kristianstad kommun
10. *Lisette Danebjer och Magdalena Nyberg* (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog
11. *Alexander Quist* (2013) Undersökning och utveckling av ett mobilt GIS-system för kommunal verksamhet
12. *Nariman Emamian* (2014) Visning av geotekniska provborrningar i en webbmiljö
13. *Martin Fredriksson* (2014) Integrering av BIM och GIS med spatiala databaser – En prestandaanalys
14. *Niklas Krave*(2014) Utveckling av en visualiseringsapplikation för solinstrålningsdata
15. *Magdalena Nyberg* (2015) Designing a generic user interface for distribution of open geodata: based on FME server technology
16. *Anna Larsson* (2015) Samredovisning av BIM- och GIS-data
17. *Anton Lundkvist* (2015) Development of a WEB GI System for Disaster Management

18. *Ellen Walleij* (2015) mapping in Agricultural Development – Introducing GIS at a smallholders farmers' cooperative in Malawi

19. *Frida Christiansson* (2016) Lagring av 3D - geodata - en fallstudie i Malmö Stad

20. *Lisette Danebjer* (2016) Methodology for creating and modifying distributed topologically structured geographical datasets

21. *Jeanette Dunn Ekelund* (2016) En jämförelse av algoritmer och resultat för flödesberäkning i QGIS/GRASS och ArcGIS.

22. *Ebba Gröndahl och Frida Thorman* (2016) Verksamheters optimala läge i staden och hur de är lokaliserade idag

23. *Gunnar Rolander* (2017) Datatransformation using linked data ontologies