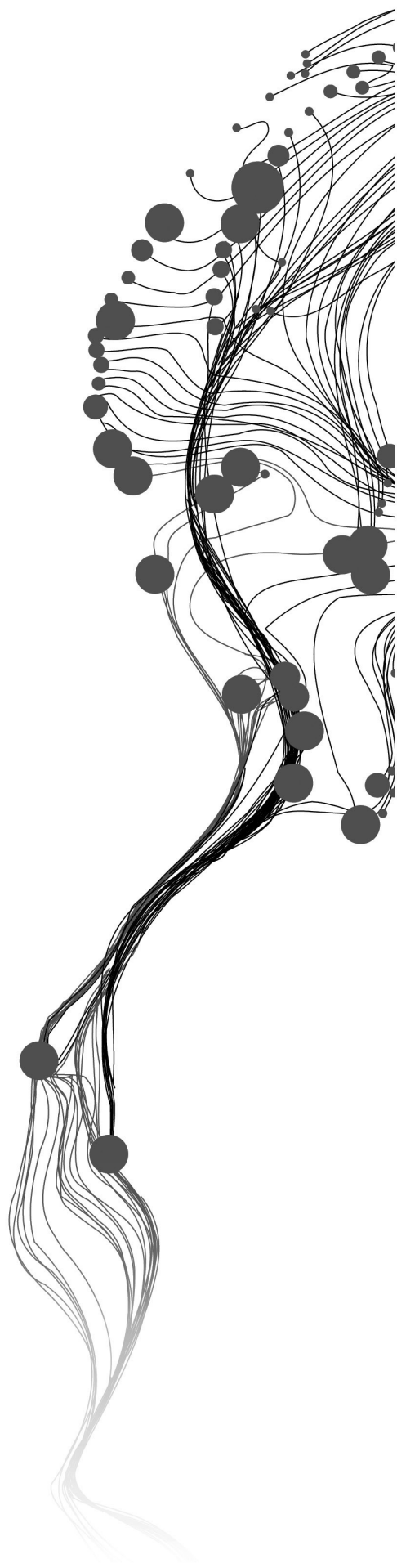


# **ONTOLOGY MATCHING FOR SPATIAL CADASTRAL DATASET INTEGRATION**

SHREYES SHIV  
March, 2011

SUPERVISORS:  
Dr. Sameer Saran  
Dr. Ir. Rolf A. de By



# ONTOLOGY MATCHING FOR SPATIAL CADASTRAL DATASET INTEGRATION

SHREYES SHIV

Enschede, The Netherlands, March, 2011

Thesis submitted to the Faculty of Geo-information Science and Earth  
Observation of the University of Twente in partial fulfilment of the requirements  
for the degree of Master of Science in Geo-information Science and Earth  
Observation.

Specialization: Geoinformatics

## SUPERVISORS:

Dr. Sameer Saran

Dr. Ir. Rolf A. de By

## THESIS ASSESSMENT BOARD:

Prof. Dr. Ir. A. Alfred Stein(chair)

Dr. R. D. Garg

#### Disclaimer

This document describes work undertaken as part of a programme of study at the Faculty of Geo-information Science and Earth Observation of the University of Twente. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the Faculty.

*Dedicated to my mother....*

## ACKNOWLEDGEMENTS

First and foremost, I want to thank my IIRS and ITC supervisors, Dr. Sameer Saran and Dr Ir. Rolf A. de By, for their constant guidance and support in carrying out this research work.

I would like to thank Dr. P.S. Roy (Dean IIRS), Mr. P.L.N. Raju (Head, GID), Mr. Prasun Gupta and Mr. Shashi Kumar for providing such a nice infrastructure and environment to carry out the present research work.

My special thanks to my father Mr. V. S. Shivarudraiah and Suranjana without whom this work would not be possible.

I also thank Vishnu, Ruchi and Priyanka for their support.

I am grateful to my colleagues Deepak, Sourabh, Tanvi, Preethi and Richardson for their support in every possible way.

I thank my parents for giving me constant and unconditional support.

Last but not the least I thank all my friends for their support.

## ABSTRACT

Integration of spatial cadastral datasets is a challenging task in itself that will be useful for many applications in the cadastral domain. By considering the elements in the datasets as concepts, they can be related with each other to create a knowledge base. The concept of ontology comes into picture when concepts, relationships between the concepts and instances of the concepts exist. To create the necessary concepts for a dataset, a good understanding of what is needed should be there for the ontology creator.

Since the datasets are in the form of shape files, it had to be put into a database in order to import it into the ontology environment. Datamaster, a tool in the ontology editor is used to import the database entities as instances of a certain class in the ontology. The ontologies of the two different datasets are imported into a single ontology and they are related so that integration of the datasets takes place. Now, from the resultant ontology the retrieval process is done with the help of a rule engine that takes a rule as an input and retrieves the required output.

This will help many users to create the ontology about their data and relate them in order to build the knowledge base which may be helpful for the community.

### **Keywords**

*Ontology, Datamaster, Rules, SWRL*

# TABLE OF CONTENTS

---

<b>Acknowledgements</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Definition . . . . .	2
1.3 Research Objective . . . . .	4
1.3.1 Main Objective . . . . .	4
1.3.2 Specific Objective . . . . .	4
1.4 Research Questions . . . . .	4
1.5 Related Work . . . . .	4
1.6 Method Adopted . . . . .	5
1.7 Thesis Outline . . . . .	5
<b>2 Ontology Concepts</b>	<b>7</b>
2.1 Ontology Development . . . . .	7
2.2 RDF - Resource Description Framework . . . . .	9
2.2.1 RDFS: A Web Ontological Schema Language . . . . .	9
2.3 OWL - Web Ontology Language . . . . .	10
2.3.1 Limitations of the Expressive Power of RDF Schema over OWL . . . . .	10
2.3.2 Compatibility of OWL with RDF/RDFS . . . . .	11
2.3.3 Species of OWL . . . . .	11
2.4 Description Logics . . . . .	11
2.5 Formalisms . . . . .	13
2.6 Reasoning . . . . .	14
2.7 Ontology Editors . . . . .	14
2.8 Semantic Web Rule Language (SWRL) . . . . .	15
<b>3 The Dataset</b>	<b>17</b>
3.1 DSSLR dataset . . . . .	17
3.2 CM dataset . . . . .	18
3.3 Study area . . . . .	19
<b>4 Constructing the Ontology</b>	<b>21</b>
4.1 Use Case Scenario . . . . .	21
4.2 Determining the Classes . . . . .	22
4.3 Property Definition . . . . .	24
<b>5 Integration of Database and the Ontology</b>	<b>29</b>
5.1 Loading Shape Files into a Database . . . . .	29
5.2 Database and Ontology Editor Connectivity . . . . .	29
5.3 Spatial Analysis . . . . .	30

<b>6</b>	<b>Results and Discussion</b>	<b>33</b>
6.1	The role of SWRL and SQWRL in retrieval of instances . . . . .	33
6.2	Results . . . . .	33
6.2.1	Retrieval of Information using Rules . . . . .	33
6.3	Discussion on the Results . . . . .	41
<b>7</b>	<b>Conclusions and Recommendations</b>	<b>43</b>
7.1	Conclusions . . . . .	43
7.2	Recommendations . . . . .	43
<b>A</b>	<b>Tools for Semantic Modelling</b>	<b>45</b>
A.1	Ontology Editing Tools . . . . .	45
A.2	The IDL program to analyse the plot parcel problem . . . . .	45
<b>B</b>	<b>Terminology</b>	<b>47</b>



## LIST OF FIGURES

---

1.1	A village map which consists of parcels, roads, streams as existed in 1960. . . . .	2
1.2	A sector which contains blocks and plots. . . . .	3
1.3	Flowchart of the methodology. . . . .	6
2.1	Cadastral domain which includes Village, Parcel and Plot and the relationships between them are defined. . . . .	8
3.1	Study area in the Mysore city in Karnataka state which belongs to the Indian subcontinent. Image source Google earth. . . . .	20
4.1	Attribute table of the CM dataset. . . . .	22
4.2	Attribute table of the parcels ontology. . . . .	23
5.1	The SQL file of the DSSLR shape file. . . . .	29
5.2	The SQL file of the CM shape file. . . . .	30
5.3	The Flow chart of the IDL program. . . . .	31
5.4	Attribute table of the intersect shape file. . . . .	32
6.1	The result of Rule 1. . . . .	34
6.2	The result of Rule 2. . . . .	35
6.3	The result of Rule 3. . . . .	35
6.4	The result of Rule 4. . . . .	36
6.5	The result of Rule 5. . . . .	36
6.6	The result of Rule 6. . . . .	37
6.7	The result of Rule 7. . . . .	37
6.8	The result of Rule 8. . . . .	38
6.9	The result of Rule 9. . . . .	38
6.10	The result of Rule 10. . . . .	39
6.11	The Plot S01BAT009 lying in Parcels P1, P2 and P3, and also occupying the area A1, A2 and A3 in the respective parcels. . . . .	40
6.12	The attribute table of the intersect shape file. . . . .	40

## LIST OF TABLES

---

2.1	Letters representing a particular logic. . . . .	12
3.1	Attributes of the parcels table. . . . .	18
3.2	Attributes of the plots table. . . . .	19
4.1	Classes of the Plots ontology . . . . .	23
4.2	Classes of the Parcels ontology . . . . .	23
5.1	Attribute table of the unite table. . . . .	32
A.1	Ontology editors . . . . .	45



## Chapter 1

# Introduction

Integration of datasets is a challenging task in itself. This research pertains to integration of two spatial datasets. Spatial datasets can either be heterogeneous (datasets of different domain having non similar attributes) or homogeneous (datasets of different domain having similar attributes) in nature. Heterogeneity may reside in the concepts, process and the methods used to create the dataset, and hence spatial data integration, sharing and reuse becomes difficult.

There are different processes by which data integration can be done; of which Relational Database Management System (RDBMS) technique is a traditional method. The retrieval of data from a RDBMS is based on schema matching. There are limitations with schema matching in RDBMS as database schema does not describe the semantics of the data explicitly, and the schema cannot be shared or reused. To overcome these limitations ontology technique is used, where the ontologies provide explicit semantics, reusability and sharability for the data. Also, the knowledge representation techniques are complex and plenty.

Ontology is defined by Gruber in [Gru93] as “an explicit specification of a conceptualization.” An ontology can be built on any domain. In this research we are experimenting with the cadastral domain. The existing methods fail to handle semantic issues which arise due to heterogeneous datasets.

### 1.1 MOTIVATION

In India, there is a steady increase in population and thus new families are added every year in large numbers, resulting in fast urban growth and in large proportion. The demand for dwelling houses increases, and hence the agricultural parcels particularly on the outskirts surrounding the existing towns are used for construction of dwelling houses. In India, the trend is that the towns grow horizontally (a single floor building) rather than vertically (multistoreyed buildings). The need for dwelling houses is proportional to the population growth in the towns as well as increase in the population due to migration from villages to urban areas.

The urban growth in cities generally extends into the surrounding agricultural parcels and hence the land use is converted from agriculture to non-agricultural purposes like residential, industrial and commercial. In India, the law proposes that whenever a land use is converted from agricultural to non-agricultural purposes, the user needs to pay a certain fee. For example, if the general public builds dwelling houses in the plots created in agricultural parcels, the land use conversion fee needs to be paid to the Revenue Department.

The Government does not have an elaborate and perspective urban planning system. Unless there is a quick urban plan, the conversion fee cannot be collected by the Revenue Department. Both the planning and conversion fee collection are delayed, due to which the public are not served quickly and are constrained to go without an urban plan and also without paying conversion fee to the Revenue Department. They start developing plots and dwelling areas on their own in the agricultural parcels, sometimes without proper planning from their side. The group housing, co-operative housing and planning is not very intensive in India, and individuals prefer to create their

own plots and dwelling places.

There are different departments that maintain information on cadastral data in India. There could be multiple datasets for the same area of land. The DSSLR (Department of Survey Settlement and Land Records), of Karnataka State in India maintains the information on agricultural parcels, both spatial and textual information. It also maintains information on Government owned lands, roads, rivers, streams, hills, forests and habitation areas. Aggregation of these agricultural parcels and other lands is called village. Average area of a village is 10 km<sup>2</sup>. A sample village map consisting of agricultural parcels, roads, streams is shown in Figure 1.1.

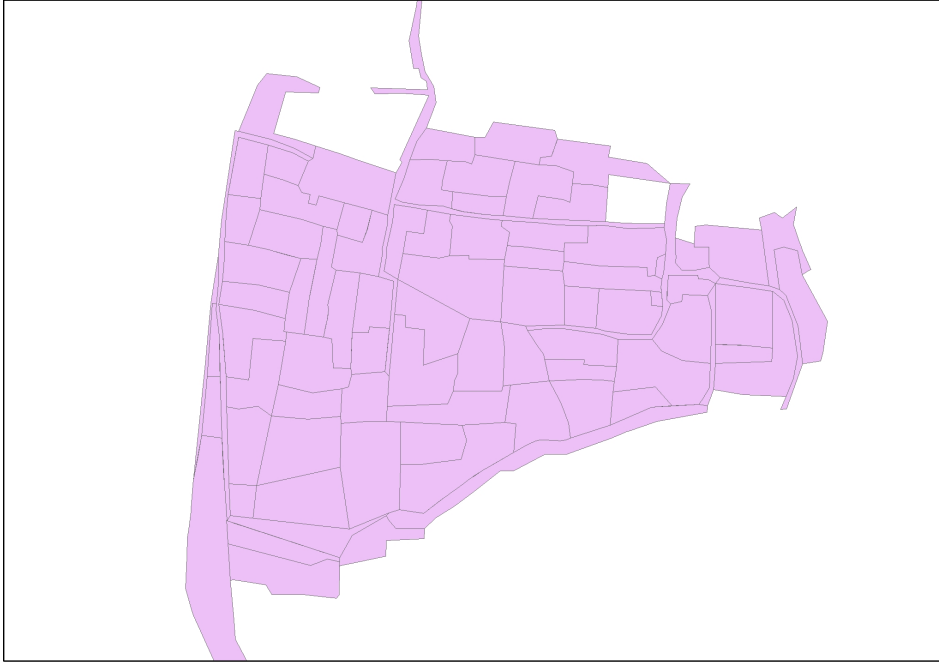


Figure 1.1: A village map which consists of parcels, roads, streams as existed in 1960.

The City Municipalities (CM) in Karnataka are maintaining other types of land related information like plot information (ownership, dimensions and areas), for their respective cities or towns for various purposes like urban facility management, property tax collection, transport management, planning and development etc. A sample map of a sector in a town is shown in Figure 1.2.

The functionality, purpose of data collection, data source type and the data acquisition techniques of the two departments (the DSSLR and the CM) are different. Due to this reason, the terminology, semantics, data schemas and the attributes used by the two departments are different. The two departments DSSLR and CM collect different type of charges and tax on the basis of ownership records. These ownership records have both spatial and non-spatial attributes. The acquisition of data was from different dates due to which it will be difficult to compare and use them in combination. A system had to be developed that helped the departments to use the data in combination with each other and integrate with other data from other organizations.

## 1.2 PROBLEM DEFINITION

Due to the fast growing urban outskirts the agricultural parcel data in the village maps become obsolete and outdated very fast, and the DSSLR has not updated the spatial features in these

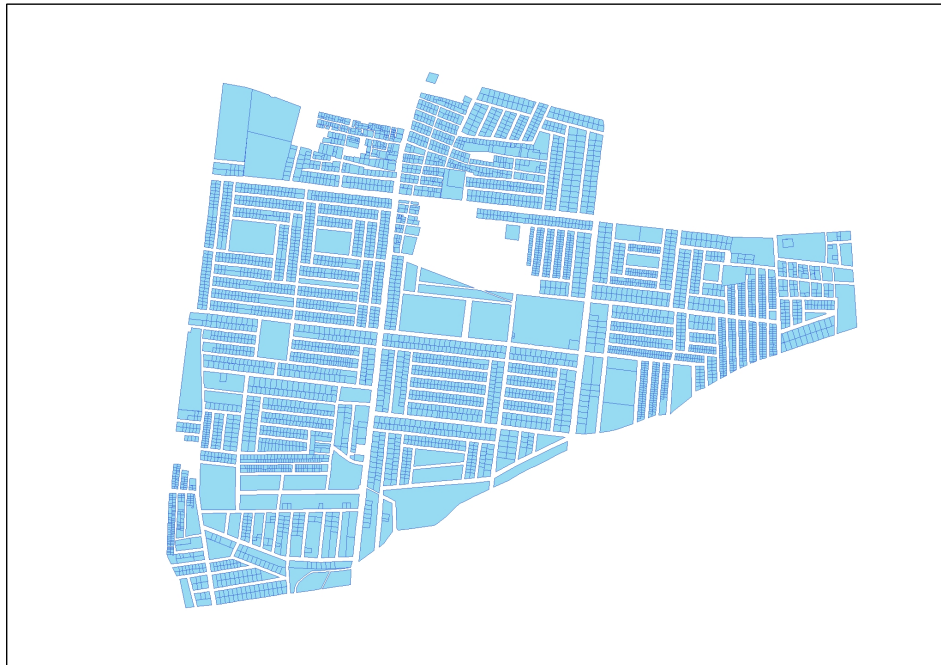


Figure 1.2: A sector which contains blocks and plots.

village maps. It was told that the data in village maps are at least 50 (the data acquisition was done between 1950-1960) years behind, showing the agricultural parcels owned by the private individuals and land parcels owned by the Government and public as existed 50 years ago, before the urbanization took place. The data now is not relevant for agricultural data management in these urbanised extensions. However the spatial data can be used for various other analytical and planning purposes.

Also the land use conversion approval process in the Revenue Department of the Government does not catch up with the demand for dwelling space and urgency of the citizens (which is in turn proportional to the increase in population), and hence the citizens are constrained to go ahead and build their houses and roads in residential layouts without the conversion permission from the Revenue Department.

A new land user, who wants to buy the plot, needs to ensure that the conversion fee has been paid to the Revenue Department for that plot by the earlier owner/land user. But this does not always happen due to land market forces and urgency of need for a dwelling place. Therefore most of the time, the citizens take risk in purchasing a plot even without a land use conversion fee paid for the land, by the earlier owner. They assume they can pay the conversion fee at a later date after purchase, when the Revenue Department actually demands for it or when the owner intends to sell it to others. The other reasons may be lethargy, corruption and delay in approval by the bureaucracy in the Revenue Department. In the present days, cooperative movement for housing is building up, community housing projects and satellite town projects are coming up in agricultural parcels even before the Urban Planning Department plans.

DSSLR also maintains information on agricultural parcels converted for non-agricultural purposes. The Revenue Department along with the assistance of DSSLR are authorized to collect the conversion charges. The DSSLR, now, needs to identify the plots that are lying in the parcels, which have not officially been converted for non-agricultural purposes. CM maintains the information about the plots, owner name, area, land use etc. When the departments have to work with

each other for update and validation, the data does not match because of the variations in schema, semantics and different terminology.

There is a time gap of at least 50 years between the two spatial datasets DSSLR and CM. The sharing of information like new ownership records and bifurcation of land parcels may help fix legal disputes if any.

Spatial ontologies specify the vocabulary in the geospatial domain that captures the “semantic details of geospatial concepts, categories, relationships and processes as well as their interrelations at different levels.” [ZDY<sup>+</sup>07] Ontology can help these departments to automate the process of updating their datasets. Once class relationships are made part of the ontology and the spatial relation rules are established, they can be used to infer the knowledge base, and uncover implicit information stored in it.

### **1.3 RESEARCH OBJECTIVE**

#### **1.3.1 Main Objective**

To develop a method for the integration of two spatial cadastral datasets using ontology, to share and update the datasets.

#### **1.3.2 Specific Objective**

- To integrate cadastral datasets using concepts of ontology.
- To develop a rule set to extract spatial relations from the mapped ontology.

### **1.4 RESEARCH QUESTIONS**

- How to integrate two different datasets, with different spatial domains at the conceptual level?
- How can spatial rules be applied on conceptually mapped datasets for a naive user?
- How can the resultant related ontologies be used in other applications?

### **1.5 RELATED WORK**

Ontology is a very broad area of research. The use of semantic tools such as ontology and reasoning can help integrate data collected by different organizations semantically and unambiguously. The semantic web is a web of information that machines can process without the involvement of humans. The Dutch TD Kadaster had to build a generic generalization model to fulfill the need of a scale-less database. They had to automate the process of generalization due to difference in scales. A generic data model was developed that supported in automatic generalization process to integrate the datasets of different scales. The data model included creating ontologies for semantic integration in the spatial domain. The two knowledge representations RDF (Resource Description Framework) and OWL (Web Ontology Language) were used to build the data models. OWL was specifically used to define hierarchy, relationships between concepts of the same ontology and also between other ontologies [SLKB06].

Geographic dataset integration results in sharing of information among different geographic information sources and also reuse of updates from one geographic dataset to another geographic dataset is possible which saves time and man power. The relationships between domain ontology

and application ontology are defined by abstraction rules at the class level [UvOMM99]. The semantic correspondences between the object classes of different geo-ontologies could be revealed by analyzing spatial and geometric characteristics of instances, of the datasets. The transformation rules were derived by data mining techniques that allowed the semantic connection between datasets, and integrate datasets of different origin and resolution levels [Kie08].

Comparison of the semantics of two or more data sources was possible because of the semantic similarity model which helped in semantic data integration [Hal06]. The conceptual representation of geographical datasets, relationships defined over them, the semantic similarity between the concepts and the instances of heterogeneous data which lead to semantic dataset integration is depicted in [Kip10]. By adopting an ontological framework the existing semantic data integration approaches of geographical ontologies are compared and analyzed [Kok06].

This research shows the mapping of database to the ontology as class instances, relating different ontologies of a similar domain, and finally retrieving the required data from the database according to the query.

## 1.6 METHOD ADOPTED

This section describes the approach followed to address the research problem and attain the research objectives.

Initially the datasets were analysed to identify the possible classes, objects, relations and instances. The ontology was created in the Protégé ontology editor, with the identified classes. Object property was defined where the relationships between the classes are defined and the restrictions were imposed on the classes.

In this research study MySQL database software is used to store the data. The datasets which are in the ESRI shape file format are loaded into a database with the help of shp2mysql tool.

Datamaster a plugin in Protégé which is used to import the database tables into the Protégé environment as instances of a certain class. Now the ontology is connected with the instances. The datatype properties are redefined for all the attributes of the database in the ontology.

A new ontology could be created where both the ontologies are imported. But Protégé instances do not support spatial datatypes due to which the two datasets could not be related spatially instead they can be related non spatially and analysis could be done.

The Semantic Web Rule Language (SWRL) plugin is used to write rules and the rules are executed in the Semantic Query-Enhanced Web Rule Language (SQWRL) query reasoner. The Figure 1.3 illustrates the flowchart of the methodology.

## 1.7 THESIS OUTLINE

**Chapter 1** describes the motivation of this research, the problem statement, research objectives and questions, related work, terminology and the method adopted in this research.

**Chapter 2** gives an overview of the concepts of ontology. A brief description of OWL, RDF, Description Logics, Formalisms, Reasoning, Ontology editors and SWRL are provided in this chapter.

**Chapter 3** describes the dataset, process of acquiring the dataset, specification used etc. The study area is described.



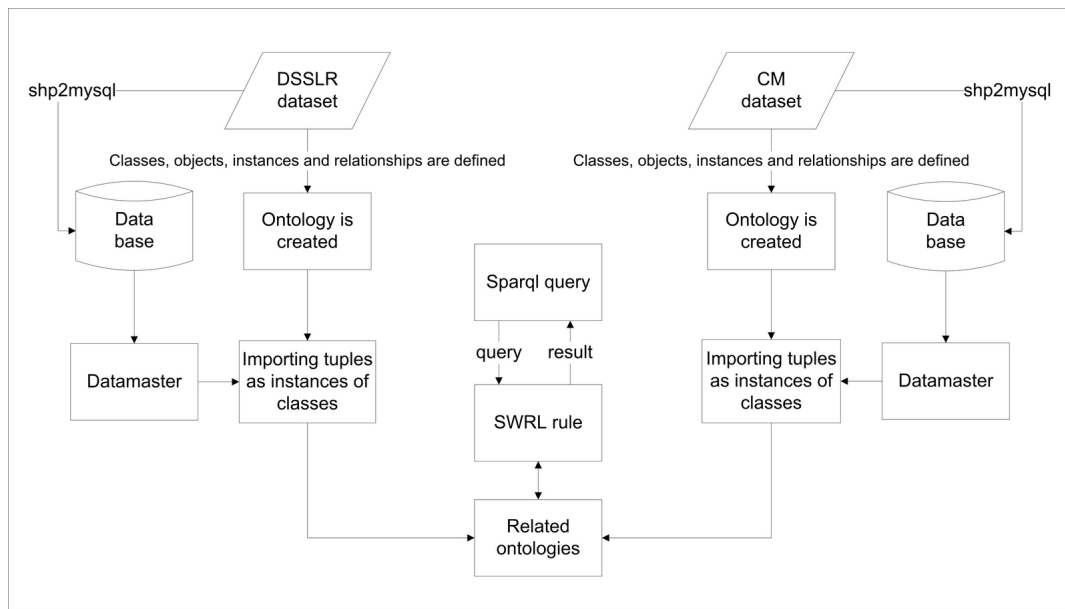


Figure 1.3: Flowchart of the methodology.

**Chapter 4** describes how the ontology was modelled and how the classes and properties were defined in the ontology.

**Chapter 5** describes how the dataset was loaded into a database and how the database was integrated with the ontology.

**Chapter 6** describes how the information was retrieved from the ontology, the role of SWRL and SQWRL in the retrieval of information from the ontolog and discussions.

**Chapter 7** describes the conclusion, answers to the research questions and recommendations.

## Chapter 2

# Ontology Concepts

Ontology is a branch of philosophy that deals with things that exists in reality. The concept of ontology is used in computer science to depict information objects [GOS09]. It is used in Artificial Intelligence (AI) systems, since the concept of knowledge representation in AI systems is to represent the things that exist [Gru93]. Ontology is now gradually moving from AI to several domains [NM01].

There are several reasons stated by Noy and McGuinness in [NM01], to tell why ontologies are developed, of which some are:

- *Sharing the information structure and the common understanding of it* among the people, will help the community to aggregate and share data over the internet, to answer user queries or can be used as input data for other applications.
- *Reuse of domain ontology* leads to efficient use of resources. If one group of people design ontology in detail, others can just use it for their respective domains. If a bigger ontology has to be created then several existing ontologies or newly created ontologies can be integrated, describing parts of the bigger ontology.
- *Explicitly change domain assumptions* as and when the understanding about the domain changes.
- *Exclusivity of domain knowledge from the operational knowledge*, the ontology is separated from its operational knowledge which may be an algorithm. This algorithm can be used with other ontologies of different domain which is altered to do the intended work.
- *Analyzing domain knowledge* can be done when the declarative specification of the terms are available.

The objective of creating ontologies for a specific domain is not the goal, but defining concepts, relationships and their structure for other programs to reuse [NM01].

## 2.1 ONTOLOGY DEVELOPMENT

The knowledge representation of a real world object is done by conceptualization. Genesereth and Nilson in [GN87] explain the notion of conceptualization using a mathematical representation called as an extensional relational structure, here the conceptualization is defined as a tuple  $(D, R)$  where

- $D$  is a set of all elements called the universe of discourse.
- $R$  is a set of all the relations on  $D$ .

The set  $R$  contains mathematical relations, sets of ordered tuples as elements of  $D$ . Every element in  $R$  is an extensional relation, reflecting a specific state of world which includes the elements of  $D$ , as depicted in Figure 2.1, which is explained in Example 2.1 [GOS09].

**Example 2.1:** Similar to example 2.1 in [GOS09] an example related to this research is illustrated here. Consider the cadastral domain where, the set of elements belonging to the universe of discourse  $D$  contains all the pieces of land that have a unique identifier and belong to a certain person or group of persons or a governing body. The set of relations  $R$  contains binary relations like *has\_owner*, *contains* and *has\_landuse*. The extensional relation structure  $(D, R)$  for this example looks like:

$D = \{\text{Village, Parcel, Plot}\}$

$R = \{\text{contains, part\_of, ...}\}$

Relation extensions reflect a specific state of world. In this example we assume that piece of landv(Plots and Parcels) comprises the whole universe  $D$ . The binary relations like *contains*, *part\_of* are the sets of tuples that define every hierarchical relationship and every collaboration in this cadastral domain. Some Village, Parcel and Plot are depicted in Figure 2.1. Here *Plot* S01BSG001 is a *part\_of* *Parcel* 23, *Parcel* 23 is a *part\_of* *Village* 001 and *contains* *Plot* S01BSG001, *Village* 001 *contains* *Parcel* 23.

- $\text{Plot} = \{\text{S01BSG001, S01BAT009, ...}\}$
- $\text{Parcel} = \{23, 45, ...\}$
- $\text{Village} = \{001, 002, ...\}$
- $\text{part\_of} = \{..., (\text{S01BSG001, 23}), (23, 001), ...\}$
- $\text{contains} = \{..., (001, 23), (23, \text{S01BSG001}), ...\}$

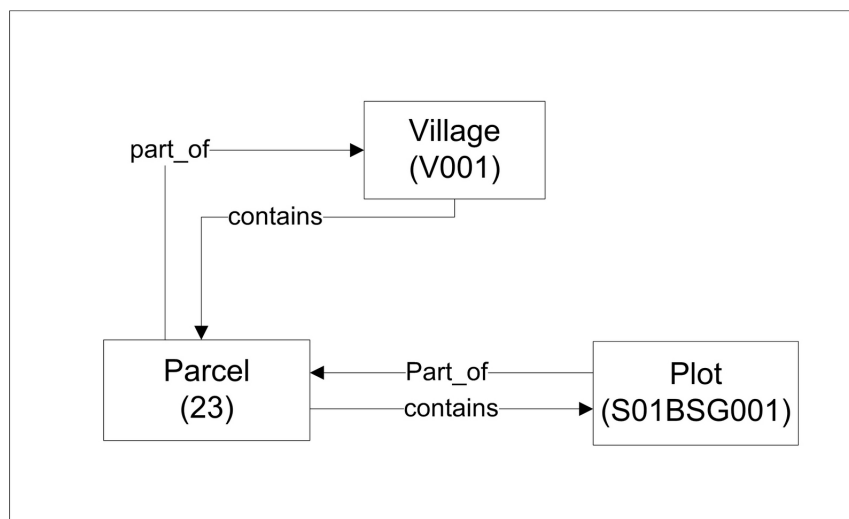


Figure 2.1: Cadastral domain which includes Village, Parcel and Plot and the relationships between them are defined.

Later Gruber in [Gru93] defined conceptualization as “an abstract simplified view of the world that we wish to represent for some purpose,” which was used to define ontology as “explicit specification of a conceptualization.”

## 2.2 RDF - RESOURCE DESCRIPTION FRAMEWORK

RDF provides the semantics to the resources on the web. It provides descriptions (metadata) of the resources on the web by following a standard model specified by RDF Schema (RDFS). `rdf:subject` is used to define the subject of a statement, like a thing or a person who carries out the action. `rdf:predicate` is used to define the predicate of a statement, like a sentence describing about what a thing or a person does. `rdf:object` is used to define the object of a statement, like a thing or a person upon which the action is carried out. RDF represents data in statements (triplets) of subject-predicate-object structure. These triplets can then be shredded in to a relational table [MMP<sup>+</sup>07]. RDFS, however, describe the vocabulary to specify classes, their properties and the relationships they share leading to the creation of light weight ontologies. Gašević et al in [GDD09] has stated that “RDF itself is used to describe instances of ontologies, whereas RDFS encodes ontologies.” This is done using frame-based modelling primitives like `class`, `subClassOf`, `property`, and `subPropertyOf`. But there are certain inherent restrictions defined by `rdfs:domain` and `rdfs:range` which restricts the possible combination of classes and properties.

RDF is a W3C<sup>1</sup> recommendation which provides data model for annotations in the semantic web. A set of RDF statements forms a RDF graph. RDF defines a specific extensible markup language(XML) syntax known as RDF/XML, to represent the statements in a machine understandable language [Pan09].

### 2.2.1 RDFS: A Web Ontological Schema Language

RDFS (RDF Schema) is a W3C recommendation to express simple ontologies with RDF syntax. Classes (concepts), resources and properties (roles) can be defined using the predefined web resources `rdfs:Class`, `rdfs:Resource` and `rdfs:Property` respectively [Pan09].

Information properties are not predefined in RDFS, but a set of meta-properties can be used to represent background assumptions in ontologies as defined in [Pan09] as:

- `rdf:type`: the instance-of relationship.
- `rdfs:subClassOf`: this property models the subsumption hierarchy between classes.
- `rdfs:subPropertyOf`: this property models the subsumption hierarchy between properties.
- `rdfs:domain`: this property constrains all instances of a particular property to describe instances of a particular class.
- `rdfs:range`: this property constrains all instances of a particular property to have values that are instances of a particular class.

RDFS statements are RDF triples, i.e., they provide no syntactic restrictions on RDF triples [Pan09].

---

<sup>1</sup>World Wide Web Consortium <http://www.w3.org/>

## 2.3 OWL - WEB ONTOLOGY LANGUAGE

OWL stands for web ontology language; and is a W3C recommendation for ontology language. OWL language is used to formalize a domain by defining classes and their properties, individuals and their properties, and reasoning on the classes and individuals to the extent specified by the formal semantics of the OWL language [w3c04].

Like RDF and RDFS, Web Ontology Language (OWL) also provides semantics to the resources on the web. OWL vocabulary is built on top of RDFS vocabulary and is much semantically richer language than RDF in connecting relations between classes, properties and instances [w3c04]. This further helps in inferencing over the web resources. Recently W3C has accepted and released the OWL 2.0 language. Some of the new features of OWL 2.0 as stated in [w3c09] include “extra syntactic sugar, additional property and qualified cardinality constructors, extended data type support, simple metamodelling, and extended annotations.”

### 2.3.1 Limitations of the Expressive Power of RDF Schema over OWL

Some ontological knowledge can be represented using the RDF and RDFS. Antoniou and van Harmelen, stated about RDF/RDFS in [AvH09] as “the main modelling of the RDF/RDFS was to organize vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions and instances of classes.” There are many features that are defined in [AvH09], are missing in RDF/RDFS of which some are listed below:

- Local scope of properties: The range of a property is defined by `rdfs:range`. Range restrictions cannot be declared in RDF Schema such that, they apply only to some classes. For example, we cannot say that a piece of land have some *plot\_id*, while other piece of land may have *parcel\_id*.
- Disjointness of classes: Some classes are disjoint which have to be expressed the way they are. For example, *plot\_id* and *parcel\_id* are disjoint. Only subclass relationships can be stated in RDF Schema.
- Boolean combinations of classes: By combining different classes using union, intersection and complement we may wish to build new classes in RDF Schema. For example, we may define the class *id* to be the disjoint union of the classes *plot\_id* and *parcel\_id*. These kind of definitions are not allowed by RDF Schema.
- Cardinality restrictions: The number of values a class property must take, is a restriction that we may wish to define sometimes. For example, we want to say that a landuse of a piece of land can have exactly one type of landuse, and Plot is owned by minimum one owner. Such restrictions cannot be expressed in RDF Schema.
- Special characteristics of properties: Sometimes it is necessary or useful to say that an object property is transitive (like *Plot* is *part\_of* some *Village* and *Village contains* some *Plot*).

An ontology language is needed which is richer than RDF schema that offers the above features and more. There is a tradeoff between expressive power and efficient reasoning support in designing such a language. Hence Antoniou and van Harmelen in [AvH09] stated that “we need a compromise on a language that can be supported by reasonable reasoners, while being sufficiently expressive to express large classes of ontologies and knowledge.”

### 2.3.2 Compatibility of OWL with RDF/RDFS

OWL is an upgrade of RDF schema, it uses the RDF meaning of classes and properties (`rdfs:Class`, `rdfs:subClassOf`, etc.), and it adds additional language primitives to support richer expressiveness specified above.

The trade-off between the expressive power and efficient reasoning simply clashes with the extension of the RDF schema. Some modelling primitives like `rdfs:Class` (the class of all the classes) and `rdfs:Property` (the class of all the properties) are very powerful and expressive. The expressive primitives identified above will lead to uncontrollable computational properties if the logic is extended with them [AvH09].

### 2.3.3 Species of OWL

The requirements like efficient reasoning support and convenience of expression, for a language which is powerful and includes RDF schema with logic, may seem incompatible. The W3C Web Ontology Working Group defined OWL as three different sublanguages by looking into the requirements. Each sublanguage is equipped to fulfil the different aspects of these incompatible full set of requirements:

- **OWL Full:** OWL Full provides maximum expressiveness support than all of its fellow languages and syntactic freedom of RDF with less or no reasoning support [w3c04]. OWL Full uses the entire OWL languages primitives and allows them to be combined in arbitrary ways with RDF and RDF Schema. It is extremely expressive and is upward compatible with RDF, both syntactically and semantically [AvH09]. The disadvantage is that the language has become so powerful that automated reasoners stop processing on OWL full ontologies [Hal06].
- **OWL DL:** OWL DL (Description Logic) is a very expressive sublanguage of OWL Full. In order to regain computational efficiency, OWL DL restricts the way constructors from OWL and RDF are used. Efficient reasoning is supported by OWL DL which is an advantage but, total loss of compatibility with RDF is a disadvantage [AvH09].
- **OWL Lite:** OWL Lite is a sub-set of OWL DL which enforces more restriction on OWL DL to a subset of the language constructors. OWL Lite is easy to understand for all kinds of users and easy to implement for developers. The disadvantage is restricted expressivity [AvH09].

Suitable sublanguage of OWL should be considered by ontology developers according their needs. If the user wants more expressivity then the most would be OWL Full followed by OWL DL and finally OWL Lite. If the users want the meta-modelling facilities of RDF Schema (example: define class of a class, attach properties to a class) then the most supportive would be OWL Full. When the users want reasoning it is better the users go for OWL DL rather than OWL Full since reasoning support is less or impossible [AvH09].

## 2.4 DESCRIPTION LOGICS

Description Logics (DL) is a formal knowledge representation language. DL is mainly used in AI (Artificial Intelligence) for formal reasoning on the concepts of a domain (terminological knowledge). The DL is more expressive and has more efficient decision problems than first-order logic. First-order logic is used to give precise definitions to the concepts and relation of entities in the

real world [Hal06]. In DL, the important idea of the domain is described by the concept descriptions, which Baader et al in [BHS09] described as “expressions built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL.” There are many different sub languages of DL which are formed by the expressivity that a particular sub-language allows. The expressivity is described by Baader and Nutt in [BN03] with the letters representing a particular logic as shown in Table 2.1.

Table 2.1: Letters representing a particular logic.

Letter	Description
$\mathcal{F}$	Functional properties.
$\mathcal{E}$	Full existential qualification (Existential restrictions that have fillers other than owl:thing).
$\mathcal{U}$	Concept union.
$\mathcal{C}$	Complex concept negation.
$\mathcal{S}$	An abbreviation for Attributive Language with complement of any Concept $\mathcal{ALC}$ with transitive roles.
$\mathcal{H}$	Role hierarchy (subproperties - rdfs:subPropertyOf).
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Nominals. (Enumerated classes of object value restrictions - owl:oneof, owl:hasValue).
$\mathcal{I}$	Inverse properties.
$\mathcal{N}$	Cardinality restrictions(owl:Cardinality, owl:maxCardinality).
$\mathcal{Q}$	Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than owl:thing).
$(\mathcal{D})$	Use of datatype properties, data values or data types.

The Protégé ontology editor supports OWL. OWL DL is a sub-language of OWL which exploits its formal semantics from  $\mathcal{SHOIQN}^{(\mathcal{D})}$  [Rag07]. Schmidt-Schaulß and Smolka in [SSS91] defines Attributive Language  $\mathcal{AL}$  as “the minimal sensible attributive concept description language.” For example if A and B are definitions which signify atomic concepts, R be an atomic role and C, D are concept definitions then, the language  $\mathcal{AL}$  is defined by Baader and Nutt in [BN03] as

$C, D \longrightarrow A$		(the atomic concept)
$\top$		(the universal concept)
$\perp$		(the bottom concept)
$\neg A$		(atomic negation)
$C \cap D$		(intersection)
$\forall R.C$		(value restriction)
$\exists R.\top$		(limited existential quantification)

In the above defined language C and D imply the following things respectively,

- They are unique concepts, called as atomic concepts (A).
- Belong to a universal concept ( $\top$ ) that contains all the concepts.
- Part of an empty concept or bottom concept ( $\perp$ ).
- The complement ( $\neg A$ ) of a concept is a concept.
- The intersection ( $C \cap D$ ) of two concepts is a concept.
- A role imposing universal restriction( $\forall$ ) on a concept is a concept.
- A role imposing existential restriction( $\exists$ ) on a concept is a concept.

For example as defined in the above language, Plot and Parcel are atomic concepts. Based on these atomic concepts we define a Village as  $\text{Plot} \cup \text{Parcel}$ . By adding or removing constructors  $\mathcal{AL}$  can be extended or restricted. To increase  $\mathcal{AL}$ 's expressiveness, Hall in [Hal06] describes how the following constructs can be added:

- Union of concepts ( $C \cup D$ ) represented as  $\mathcal{U}$ .
- Full existential quantification ( $\exists R.C$ ) represented as  $\mathcal{E}$ .
- Number restrictions  $\geq n$  and  $\leq n$  meaning that the number of concepts filling a property must be at most or at least n represented as  $\mathcal{N}$ .
- Negation of arbitrary concepts ( $\neg C$ ) written as  $\mathcal{C}$ .

The  $\mathcal{AL}$  family of languages consists of  $\mathcal{AL}$ ,  $\mathcal{U}$ ,  $\mathcal{N}$ ,  $\mathcal{C}$ . All these sub languages of  $\mathcal{AL}$  are not semantically distinct, for example Hall in [Hal06] states that “union and full existential quantification can be described by negation and vice versa.” Hence Hall in [Hal06] states that “all  $\mathcal{AL}$  languages can be written as  $\mathcal{U}$ ,  $\mathcal{E}$ ,  $\mathcal{N}$  and since  $\mathcal{U}$ ,  $\mathcal{E}$  are equivalent to  $\mathcal{C}$  the  $\mathcal{AL}$  family is usually written as  $\mathcal{AL}$ ,  $\mathcal{C}$ ,  $\mathcal{N}$ .”

## 2.5 FORMALISMS

A Formalism is a theory that consists of mathematical and logical statements which are regarded as outcomes of the transformation rules. The TBox is used to define the terminology of the knowledge base and the ABox contains assertions of individuals in a knowledge base. The TBox and ABox are the basic formalisms of Description Logics. A set of terminological axioms in a TBox define how concepts and roles are related to each other. The general terminological axiom inclusion and equality are depicted below,

$$C \subseteq D \text{ and } C \equiv D$$

In the above example the equation  $C \subseteq D$  depicts that the concept C is subsumed by the concept D, in other words a concept of an ontology that represents a specific aspect of the world than the concept in a different ontology, here is C depicts the specific concept. The equation  $C \equiv D$  depicts equivalence between the concepts C and D, where the concepts represent similar aspect of the world. An ABox contains the assertions about the individuals in the knowledge base. For example  $\text{Plot}(\text{S01BSG001})$ , this defines that the S01BSG001 exists and is an instance of the concept Plot [Hal06].



## 2.6 REASONING

Reasoners are used to reason out the knowledge provided in the form of declarative rules. From a set of asserted facts or axioms, logical consequences can be inferred by a reasoner. First-order predicate logic is used to perform reasoning by majority of the reasoners. It helps to check whether a class can subsume another class or not. They are also used to check the consistency of the classes, properties and the overall ontology.

For this research Jess is used as the rule engine. The SQWRL which is based on SWRL is used to query the OWL ontologies. The SQWRL runs with Jess as the rule engine that helps in reasoning out the rules and retrieving knowledge from the OWL ontologies. Jess is a rule engine which also supports reasoning. It is also a powerful and faster rule engine.

Reasoning Axioms of a knowledge base can be translated into first-order logic by using Description logics. Inferences can be made on the axioms to make implicit knowledge contained in the axioms explicit. The main inferences that exist in description logics as stated by Hall in [Hal06] are:

- *Satisfiability*. A concept C is satisfiable with respect to a TBox if there is a model of the TBox so that the set of elements of concept C in the model is not empty.
- *Subsumption*. A concept D subsumes a concept C if the set of elements of C is a subset of the set of elements that D describes with respect to a TBox.
- *Equivalence*. Two concepts C and D are equivalent if the sets of elements are equal with respect to a TBox.
- *Disjointness*. Two concepts C and D are disjoint if the intersection of their sets of elements is the empty set.

Satisfiability and Subsumption are the most used inferences. By updating a knowledge base by adding, removing or changing the concepts, it is vital to know whether the new or changed concept is considered as a valid concept within the TBox and also whether other existing concepts have become invalid through the addition or change.

## 2.7 ONTOLOGY EDITORS

Ontology development is done using ontology editors. There are numerous ontology development tools developed of which the major ones are shown in Table A.1 listed in appendix A. Ontology tools are used to develop and manage ontologies from ground up. The editors are used to define and modify the concepts, properties, relations, restrictions, axioms, rules and many other things. Protégé<sup>2</sup> is one such ontology development tool that is majorly used. Protégé was developed by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine for medical informatics purpose [pro].

Protégé OWL as a software helps in browsing and modifying OWL ontologies. It allows plug-ins to be added to the interface which may help in visualizing, reasoning, and many other things. Users can create domain ontologies, customize them and enter domain knowledge. Protégé helps users to create ontologies even if they do not know the OWL, RDF, RDFS languages. It has a graphical interface which is easy to use. Plug-ins add extra functionality that can be used to create or modify the ontologies in Protégé.

---

<sup>2</sup><http://protege.stanford.edu/>

## 2.8 SEMANTIC WEB RULE LANGUAGE (SWRL)

SWRL is developed by combining two sublanguages of OWL, OWL DL and OWL Lite with the sublanguage of Rule ML, unary/binary Datalog RuleML. It has a highly abstract syntax which allows Horn-like rules to be combined with an OWL knowledge base. The rules are of the form of antecedent (head) and consequent (body). Whenever the antecedent condition holds then the conditions specified in the consequent should hold [HPSB<sup>+</sup>04]. SWRL is used to translate natural language queries to OWL ontology. The variables in SWRL are represented by a identifier that is preceded by a “?”, for example ?a, ?plot.

**Example 2.7:** There are two concepts Plot and Tax. The natural language statement is “if a Plot has an area less than or equal to 27 m<sup>2</sup> then it belongs to tax slab1” the corresponding SWRL rule statement is:

$$\text{Plot}(\text{?p}) \wedge \text{db:Owner\_nm}(\text{?p}, \text{?own}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?pl}) \wedge \text{db:Landuse}(\text{?p}, \text{?land}) \wedge \text{swrlb:lessThanOrEqual}(\text{?a}, 27.0) \wedge \text{db:Area}(\text{?p}, \text{?a}) \longrightarrow \text{Tax\_slab1}(\text{?p})$$

Where ?p, ?area are variables.

SQWRL is an extension of SWRL, it is used to query ontologies and it is semantically compatible with SWRL.



## Chapter 3

# The Dataset

The datasets that I am using in this research are of the cadastral domain. There are two shape files; one is from the DSSLR and the other is from the CM.

### 3.1 DSSLR DATASET

The original village data containing the land parcel boundaries, on paper based map was prepared during 1950-1960 at the scale 1:8000.

The surveying techniques used to prepare the village maps were conventional and old, yet accurate as per the standards set 50 years ago. Survey and Mapping was done using combination of methods like, traverse, triangulation, baseline and offsets (Offsets are perpendicular to the baseline). Mechanical theodolites were used to measure the angles and chains of 20 meter length were used to measure the distances. Each chain consisted of 100 links of 20 cm each.

Initially the village boundary was surveyed using closed traverse method (angles and distances) using theodolite and chains. The village boundary was plotted using the traverse data at 1:8000 scale. Each individual parcel within the village boundary was measured by a combination of triangulation and baseline-offset method using, theodolite (for angle measurement), cross-staff (for right angle measurement) and chain (for distance measurement). The parcel boundaries were integrated with right adjacencies and mosaic using optical projection method to seamlessly stitch all the parcel maps to fit within the boundary of the village.

Though this data was created during the British regime in India, for the sole purpose of collection of land tax, now the data came to be used for the purpose of cadastral land records management (maintaining the record of boundaries of individual land holdings, the area contained in the boundaries of the parcels, the rights on the parcels, and to collect the land tax). Subsequently it has gained importance in protecting the interest of the land holders, maintaining the tenure security, peace and lawfulness in the civil society. The data is being used for various other purposes like, analysis, planning and development, implementation of social security schemes of the Government and agricultural planning. The data is being used as legal data for the ownership on land parcels, to establish the public rights on lands, roads, canals, open grounds and natural features like streams, hills and valleys, forests, etc.

The village map data was however not updated with splits and mergers for various administrative reasons over the past 50 years. The village maps were scanned at one to one contact size to maintain the accuracy of the maps. The data (points, lines and polygons) on the scanned image were later vectorised by digitisation. As explained above, the village data has been prepared for maintaining the cadastral land records and for the overall administration purpose.

This data was not properly georeferenced; instead it has a local adjacency reference system. Now this data is referenced to global reference system in UTM coordinate system, by measuring

the location coordinates using Global Positioning system (GPS) receivers. For each village at least 20 points were taken which could be clearly identified both on the map and on the ground as on today. The spatial accuracy of the dataset is  $\pm 20$  cm. Hence this dataset is well georeferenced and fit for use for present day standards.

The village parcels vector data is as shown in Figure 1.1. This data shows a village which is now urbanised, forming a portion of the city of Mysore in Karnataka, now losing all the characteristics of the village as depicted in the village maps. But still the village map can be super-imposed on the correct land area of the town by taking conspicuous and identifiable points on the ground. The same portion of land as on today is taken for this research study. There are around 90 parcels which belong to this sector. The attributes in this dataset are as shown in Table 3.1.

Table 3.1: Attributes of the parcels table.

Attribute	Description
FID	this is generated when the feature is digitised
Shape	this is generated when the feature is digitised
Id	this is generated when the feature is digitised
village_id	unique village id
village_nm	village name
parcel_id	unique parcel id
land_use	type of land use (agricultural(ag), non-agricultural(nag))
conversion	conversion status of the plot
area	area of the parcel
Shape_Leng	this is generated when the dataset is a feature dataset of a geodatabase
Shape_Area	this is generated when the dataset is a feature dataset of a geodatabase

### 3.2 CM DATASET

The CM data was acquired very recently; about an year ago. The area covered by this shape file as shown in Figure 1.2 is that part of Mysore city, Karnataka, India, which is the same as the DSSLR data.

The data (coordinates of the points) was acquired from the ground using ground-based measuring devices like total station and Geographical Positioning System (GPS) receivers. Initially the control points were established at regular intervals of 500 metres apart spread across the area of interest. The control point monuments were measured for coordinates using GPS receivers in differential mode. A standard control framework was established on ground.

Entire town was divided into number of sectors (each of about 1-2 km<sup>2</sup>), each sector containing many blocks and blocks were surrounded by open space, streams, drains and roads. Each block consisted of many defined plots. The sector boundary was defined to be in the middle line of the roads/streams/drainages and the block boundary was defined to be the edge of the plots.

The coordinates of every corner point on the plot boundaries were captured using total station instruments, the coordinates were derived with reference to the control network already established.

Satellite imagery of the area of interest was used as index map to guide, control and monitor the survey. The plot maps were prepared in digital format at a scale of 1:500. The data collection was done from the field. The spatial accuracy of the dataset is  $\pm 10$  cm. In this research study one such sector in shape file format with about 5000 plots is taken for analysis. The attributes in this dataset are as shown in Table 3.2.

Table 3.2: Attributes of the plots table.

Attribute	Description
FID	this is generated when the feature is digitised
Shape	this is generated when the feature is digitised
Id	this is generated when the feature is digitised
owner_nm	owner name of the plot
land_use	type of land use (commercial(com), residential(res), vacant(vac), recreational(rec), public(pub))
area	area of the plot
plot_id	unique plot id

### 3.3 STUDY AREA

The study area is from southern part of Mysore city, Karnataka, India. This area is chosen since the land here was mostly used for agriculture farming since the last 30 years or more. Since the city is expanding the agricultural lands are now being turned as residential layouts for building dwelling houses, commercial complexes etc. The location of the study area is as shown in the Figure 3.1.



Figure 3.1: Study area in the Mysore city in Karnataka state which belongs to the Indian subcontinent. Image source Google earth.

## Chapter 4

# Constructing the Ontology

This section tells us how the ontology has been constructed from the dataset that is described in the previous chapter. The ontology created for the DSSLR dataset will be referred as parcels ontology and the ontology created for the CM dataset will be referred as plots ontology further on.

### 4.1 USE CASE SCENARIO

The DSSLR maintains the information on agricultural parcels and the conversion status of the parcels converted for non-agricultural purposes. It also includes Government owned lands, roads, rivers, streams, hills, forests and habitation areas.

In India, the law proposes that whenever a land use is converted from agricultural to non-agricultural purposes, the user needs to pay a certain fee. The Revenue department along with the assistance of DSSLR is authorized to collect the conversion charges.

For example, a person owns a plot in the city of Mysore; he was always apprehensive about the legal status of the plot since he purchased it from the earlier owner, without verifying whether it had a conversion approval or not, and whether it falls outside the Government land parcel as there was no way of finding it. Even the Government is not in a position to spatially identify the land parcels on the ground, which has been granted conversions. In spite of a textual approval order for that land parcel, the person is not sure whether his plot falls within that parcel which has been converted and also does not fall inside a Government land parcel.

Also the land use conversion approval process in the Revenue Department of the Government does not catch up with the demand for dwelling space and urgency of the citizens (which is in turn proportional to the increase in population increase), and hence the citizens are constrained to go ahead and build their houses and roads in residential layouts without the conversion permission from the Revenue Department. The DSSLR has the data of land parcels converted for non-agricultural land use. It now needs to identify the plots that are lying in the parcels, which have not officially been converted for non-agricultural purposes. Some of the problems which are faced by the DSSLR are,

- To identify plots and verify whether it falls in the land parcel that has been approved for conversion from agricultural to non agricultural by the DSSLR Department.
- To find the plots that are lying in multiple land parcels and further the plot is assigned the land parcel that contains the maximum amount of area of that plot when compared to other parcels.
- A plot is declared as converted for non-agricultural purpose if, maximum area of that plot is lying in a parcel whose landuse is non-agriculture. Else the plot is declared as not converted for non-agricultural purpose.



The Revenue department of the Government owns certain land parcels with specific parcel identification number in a village. People have encroached into some of these land parcels. Revenue department is authorized to recover these land parcels for its own use or for public use. When it is found that people have already built buildings in the Government land parcels, the Government may decide to either recover the land cost as per the present market value from the encroachers or reclaim the plot encroached if there is no building on the plot. Also, the Government may create proper regulations for future building construction activities using the law.

The CM of Mysore maintains the plot information of the town for various purposes like, urban facility management, property tax collection, transport management, planning and development. The information on encroachment into Government land parcels by the people can be used to recover the land or to estimate the cost of the land to be recovered. In order to know whether the plots fall in the agricultural land parcels or the non-agricultural land parcels both the datasets have to be combined.

Ontology is needed in order to build a huge knowledge base of which, an attempt to build two knowledge bases for two different departments is done in this research. If each and every Department in the Government builds a knowledge base, they can be interrelated with each other. A bigger knowledge base can be built in which, every Department's knowledge base will act as a component. Any number of applications can be built as front ends with a single knowledge base at the back end. Since OWL is based on XML it is easy to deploy on the internet.

## 4.2 DETERMINING THE CLASSES

This research contains two datasets on which ontology has to be created individually. Initially the plots ontology is modelled using the CM dataset

Class is defined as the construct that encapsulates all its data members (objects, instances, functions). A class can also be defined by its instances. By examining the attribute table as shown in Figure 4.1 of the CM dataset, set of classes can be defined which are listed in Table 4.1

FID	Shape *	Id	Owner_nm	Landuse	Area	Plot_id	Shape_Leng	Shape_Area
0	Polygon	0	SAHILA	vac	16387.801807	S01BAB00	566.310894	16387.921779
1	Polygon	0	ABHINANDA	vac	13687.730925	S01BYK00	474.416411	13687.73081
2	Polygon	0	PHANINDRA	res	12897.443433	S01BAD00	455.898838	12897.43855
3	Polygon	0	MUSKAN	vac	225.51905	S01BAC00	60.131504	225.51905
4	Polygon	0	JANITH	res	164.194193	S01BAC00	52.23823	164.194193
5	Polygon	0	SHIPRA	vac	947.329783	S01BAC00	148.309401	947.329783
6	Polygon	0	MISRI	vac	554.443212	S01BAC00	104.016188	554.443212
7	Polygon	0	HANUMAN	res	164.795337	S01BAC00	52.265551	164.795337
8	Polygon	0	KAUMUDI	vac	515.97128	S01BAC00	99.62853	515.97128
9	Polygon	0	LEMAR	vac	176.684974	S01BAD00	55.796862	176.684974
10	Polygon	0	ADIT	vac	455.933628	S01BAD00	86.605003	455.933628
11	Polygon	0	HARSHA	res	221.268465	S01BAD00	60.835183	221.268349
12	Polygon	0	MANASI	res	219.08806	S01BAD00	60.605499	219.087914
13	Polygon	0	RAJIT	res	231.004487	S01BAD00	62.16932	231.004487

Figure 4.1: Attribute table of the CM dataset.

The *Landuse* class depicts the type of land use a plot can possess for example vacant, residential, public, commercial and recreational. The *Owner\_name* class depicts the owner names that a plot can possess. The *Plot* class depicts the plot id that is held by a plot. The *Tax* class contains five sub-classes i.e. *tab\_slab1*, *tax\_slab2*, *tax\_slab3*, *tax\_slab4* and *tax\_slab5* which are types of *Tax* but

Table 4.1: Classes of the Plots ontology

Class
Landuse
Owner_name
Plot
Tax

they are unique in their identity. The classes defined in the plots ontology are disjoint with each other. The creation and deletion of classes is an iterative process until the ontology is completed to do the task it is intended to do. In case of a change in attribute or the domain the classes may change accordingly.

By examining the attribute table as shown in Figure 4.2 of the Parcels ontology a set of classes can be defined which are listed in the Table 4.2.

FID	Shape *	Id	village_id	village_nm	parcel_id	land_use	conversion	Shape_Leng	Shape_Area
0	Polygon	0	001	nachanahalli	59	ag	n	557.239172	17851.592862
1	Polygon	0	001	nachanahalli	60	nag	y	427.673018	11233.488963
2	Polygon	0	001	nachanahalli	91	ag	n	907.947161	41134.463805
3	Polygon	0	001	nachanahalli	28	ag	n	787.428762	24372.544931
4	Polygon	0	001	nachanahalli	58	ag	n	483.374746	14150.108417
5	Polygon	0	001	nachanahalli	29	ag	n	655.877431	25425.610666
6	Polygon	0	001	nachanahalli	27	ag	n	386.967047	9631.184009
7	Polygon	0	001	nachanahalli	31	ag	n	811.901867	20105.785053
8	Polygon	0	001	nachanahalli	47	ag	n	700.974782	27217.215245
9	Polygon	0	001	nachanahalli	30	ag	n	595.395578	14817.296964
10	Polygon	0	001	nachanahalli	46	ag	n	588.671422	17167.678061
11	Polygon	0	001	nachanahalli	26	ag	n	696.082904	21297.853345
12	Polygon	0	001	nachanahalli	32	ag	n	774.786022	23546.553137
13	Polygon	0	001	nachanahalli	33	ag	n	584.392794	19098.262217

Figure 4.2: Attribute table of the parcels ontology.

Table 4.2: Classes of the Parcels ontology

Class
Parcels
Landuse
Conversion
Village

The *Parcels* class contains parcel id that is possessed by a Parcel. The *Landuse* class depicts the type of land use a parcel can possess for example, agriculture or non-agriculture. The *Conversion* class contains the status of a Parcel whether it is converted for non-agriculture purpose or not. The *Village* class contains the village id and village name for the corresponding parcel. The classes defined in the parcels ontology are disjoint with each other.

### 4.3 PROPERTY DEFINITION

To relate the classes in the ontology, properties are defined between them. In the Protégé editor several types of properties can be defined like object property, datatype property and annotation property. A datatype property contains two parts domain and the range. In the domain we define to which class the datatype property belongs, and in the range we specify the range of the domain. Restrictions can be defined on the classes with infix syntax. The description logic symbols, which were used to describe the type of restriction, are replaced with English language keywords like some, only, value, min, exactly and max. These keywords are used to write restrictions and specify range of the object property. The Boolean class constructor symbols are replaced with English language keywords like and, or, not which are mainly used to connect two or more expressions.

The plots ontology contains the properties between the classes as described below. The functional properties *has\_landuse* and *has\_owner* are defined between *Plot* and *Landuse*, and *Plot* and *Owner* respectively. The following OWL code shows how the functional properties are defined.

```
</owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#has_landuse"> # functional property
    <rdfs:range>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#has_landuse"/>
        <owl:someValuesFrom rdf:resource="#Landuse"/> # range restriction
      </owl:Restriction>
    </rdfs:range>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#Plot"/> # domain
  </owl:FunctionalProperty>

<owl:FunctionalProperty rdf:about="#has_owner"> # functional property
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_owner"/>
      <owl:someValuesFrom rdf:resource="#Owner_name"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Plot"/> # domain
</owl:FunctionalProperty>
```

The object property *has\_tax* is defined between the classes *Plot* and *Tax*. The following OWL code shows how the object property is defined.

```
<owl:ObjectProperty rdf:ID="has_tax"> # object property
  <rdfs:domain rdf:resource="#Plot"/> # domain
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_tax"/>
      <owl:someValuesFrom rdf:resource="#Tax"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
```

```

    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>

```

The parcels ontology contains the properties between the classes as described below. The object properties *has\_landuse*, *part\_of*, *contains* and *has\_status* were defined between the classes *Parcel* and *Landuse*, *Parcel* and *Village*, *Village* and *Parcel* and *Parcel* and *Conversion* respectively. The following OWL code shows how the object property is defined.

```

<owl:ObjectProperty rdf:ID="has_status"> # object property
  <rdfs:range>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#Conversion"/>
      <owl:onProperty rdf:resource="#has_status"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Parcel"/> # domain
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="part_of"> # object property
  <rdfs:domain rdf:resource="#Parcel"/> # domain
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="contains"/> # inverse property
  </owl:inverseOf>
  <rdfs:range>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#Village"/>
      <owl:onProperty rdf:resource="#part_of"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:about="#has_landuse"> # object property
<rdfs:domain rdf:resource="#Parcel"/> # domain
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_landuse"/>
      <owl:someValuesFrom rdf:resource="#Landuse"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:about="#contains"> # object property
  <rdfs:domain rdf:resource="#Village"/> # domain
  <rdfs:range>

```

```

    <owl:Restriction>
      <owl:onProperty rdf:resource="#contains"/>
      <owl:someValuesFrom rdf:resource="#Parcel"/> # range restriction
    </owl:Restriction>
  </rdfs:range>
  <owl:inverseOf rdf:resource="#part_of"/> # inverse property
</owl:ObjectProperty>

```

An ontology “unite” is created that imports both plots and parcels ontology. A class called *Unite* is defined in it so that the data as described in Section 5.3 could be imported as instances of it.

The unite ontology contains the properties between the classes as described below. The object property *contains* was defined between the classes *Village* and *Parcel*, *Parcel* and *Plot*. *part\_of* was defined between the classes *Plot* and *Parcel*, and *Parcel* and *Village*. The following OWL code shows how the object property is defined.

```

<owl:ObjectProperty rdf:ID="part_of"> # object property
  <rdfs:domain> # domain
  <rdf:Description
    rdf:about="http://www.owl-ontologies.com/Ontology1300093689.owl#Village">
    <rdfs:subClassOf>
      <owl:Restriction> # restriction
      <owl:onProperty>
<owl:ObjectProperty rdf:ID="contains"/> # object property
    </owl:onProperty>
    <owl:someValuesFrom>
    <rdf:Description
      rdf:about="http://www.owl-ontologies.com/Ontology1300093689.owl#Parcel">
      <rdfs:subClassOf>
      <owl:Restriction> # restriction
      <owl:someValuesFrom
        rdf:resource="http://www.owl-ontologies.com/Ontology1300093689.owl#Village"/>
<owl:onProperty rdf:resource="#part_of"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
    <owl:Restriction> # restriction
    <owl:someValuesFrom>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/plotfinal.owl#Plot">
    <rdfs:subClassOf>
    <owl:Restriction> # restriction
<owl:onProperty rdf:resource="#part_of"/> #object property
    <owl:someValuesFrom
      rdf:resource="http://www.owl-ontologies.com/Ontology1300093689.owl#Parcel"/>
    </owl:Restriction>
    </rdfs:subClassOf>
    </rdf:Description>
    </owl:someValuesFrom>
<owl:onProperty rdf:resource="#contains"/> #object property
    </owl:Restriction>

```

```
</rdfs:subClassOf>
</rdf:Description>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</rdf:Description>
</rdfs:domain>
</owl:ObjectProperty>
```



## Chapter 5

# Integration of Database and the Ontology

### 5.1 LOADING SHAPE FILES INTO A DATABASE

The datasets of both the departments CM and DSSLR are in the ESRI shape file format. These shape files have to be loaded into a database. This research involves the use of MySQL database which is an open source database software. It supports spatial attributes like line, point and polygon with the help of spatial datatypes like geometry, point etc.

The shape files are loaded into the database with the help of shp2mysql tool. The shp2mysql is an open source tool which generates a SQL command file from the attribute table of a shape file. Then the SQL file is loaded into a MySQL database.

In this research two shape files have to be loaded into the database. Initially the DSSLR shape file is considered where the shp2mysql tool is used to generate the SQL file which is as shown in Figure 5.1. Then the SQL file is loaded on to a required MySQL database. The *parcel\_id* attribute is defined as the primary key of the database table parcels.

```

CREATE TABLE parcels (FID INT, Id INT, village_id VARCHAR(255), village_nm VARCHAR(255), parcel_id VARCHAR
(255) NOT NULL PRIMARY KEY, land_use VARCHAR(255), conversion VARCHAR(255), Shape_Leng DOUBLE(19,11),
Shape_Area DOUBLE(19,11));
ALTER TABLE parcels ADD ogc_geom GEOMETRY;
INSERT INTO parcels VALUES
('0','0','001','nachanahalli','59','ag','n','5.57239172036e+002','1.78515928615e+004',GeometryFromText
('MULTIPOLYGON(((677936.404299999587238 1356504.928700000047684 ,677923.996799999848008
1356412.638499999418855 ,677913.662499999627471 1356335.769999999552965 ,677809.569099999964237
1356348.211699999868870 ,677843.313299999572337 1356530.339500000700355 ,677936.404299999587238
1356504.928700000047684 )))',-1) );
INSERT INTO parcels VALUES
('1','0','001','nachanahalli','60','ag','n','4.27673018325e+002','1.12334889630e+004',GeometryFromText
('MULTIPOLYGON(((678063.423100000247359 1356453.848899999633431 ,678032.347699999809265
1356377.118899999186397 ,677986.939899999648333 1356394.440700000151992 ,677923.996799999848008
1356412.638499999418855 ,677936.404299999587238 1356504.928700000047684 ,678015.651499999687076
1356480.594499999657273 ,678063.423100000247359 1356453.848899999633431 )))',-1) );
INSERT INTO parcels VALUES
('2','0','001','nachanahalli','91','ag','n','9.07947161109e+002','4.11344638046e+004',GeometryFromText
('MULTIPOLYGON(((678342.163099999539554 1356417.316099999472499 ,678315.469299999997020
1356282.743699999526143 ,678266.248700000345707 1356299.597899999469519 ,678178.806499999947846
1356322.557700000703335 ,678098.059299999848008 1356344.697300000116229 ,678088.337299999780953
1356314.957900000736117 ,678064.179499999620020 1356321.141300000250340 ,678070.071499999612570
1356348.230900000780821 ,678043.837299999780953 1356354.446299999952316 ,678032.347699999809265

```

Figure 5.1: The SQL file of the DSSLR shape file.

Secondly the CM shape file is considered where the shp2mysql tool is used to generate the SQL file which is as shown in Figure 5.2. Then the SQL file is loaded on to a required MySQL database. The *plot\_id* attribute is defined as the primary key of the database table plots.

### 5.2 DATABASE AND ONTOLOGY EDITOR CONNECTIVITY

The classes of the ontology have been defined. Now the instances have to be created for the classes. In this research the instances of the class, of an ontology, are the database tuples. To import the database tuples from an external database (MySQL) as the instances of a class in an ontology (Protégé) there should be a bridge. Datamaster a plugin in Protégé that helps to import



```

plot.sql - Notepad
File Edit Format View Help
CREATE TABLE plot (FID INT, Id INT, owner_nm VARCHAR(255), land_use VARCHAR(255), area DOUBLE(19,11),
plot_id VARCHAR(255) NOT NULL PRIMARY KEY);
ALTER TABLE plot ADD ogc_geom GEOMETRY;
INSERT INTO plot VALUES('0','0','ramesh','com','7.98613588866e+001','1sg003',GeometryFromText('MULTIPOLYGON
(((678110.416076660156250,1356461.392089843750000,678117.791076660156250,1356439.603088378906250
,678115.931091308593750,1356449.614685058593750,678108.209106445312500,1356451.452087402343750
,678110.216125488281250,1356461.361083984375000,678110.416076660156250,1356461.392089843750000 )))',-1));
INSERT INTO plot VALUES('1','0','municipality','ins','6.22393570635e+001','1sg004',GeometryFromText
('MULTIPOLYGON(((678126.254699707031250,1356458.018676757812500,678127.000671386718750
,1356457.839721679687500,678131.436706542968750,1356456.547485351562500,678131.257080078125000
,1356455.413085937500000,678129.716674804687500,1356448.903686523437500,678129.129699707031250
,1356445.977478027343750,678123.181701660156250,1356447.894714355468750,678126.254699707031250
,1356458.018676757812500 )))',-1));
INSERT INTO plot VALUES('2','0','sandesh','res','8.52664777134e+001','1sg005',GeometryFromText
('MULTIPOLYGON(((678117.899719238281250,1356460.034729003906250,678118.638305664062500
,1356459.864074707031250,678118.785705566406250,1356459.830078125000000,678126.260681152343750
,1356458.012695312500000,678123.187683105468750,1356447.888671875000000,678115.882507324218750
,1356449.616516113281250,678117.899719238281250,1356460.034729003906250 )))',-1));
INSERT INTO plot VALUES('3','0','govt school','res','6.39999045264e+001','1sg006',GeometryFromText
('MULTIPOLYGON(((678131.443115234375000,1356456.541076660156250,678139.518676757812500
,1356454.573303222656250,678136.843078613281250,1356446.603088378906250,678129.723083496093750
,1356448.897094726562500,678131.443115234375000,1356456.541076660156250 )))',-1));
INSERT INTO plot VALUES('4','0','preethi','res','7.11200730558e+001','1sg007',GeometryFromText

```

Figure 5.2: The SQL file of the CM shape file.

schemas and data from Relational Databases into Protégé. The database tables are imported from the database to a certain ontology in Protégé using the Datamaster plugin with the help of JDBC drivers and ODBC drivers.

The database table plots is imported from the MySQL database to the plots ontology in Protégé using the Datamaster plugin with the options import table as class and import table content. The *Plot* class is specified as superclass for the table class which is created when the table is imported into the ontology. The attributes of the table plot i.e. Owner\_nm, Landuse, Area, are now defined as datatype properties. In datatype properties, Domain (the class to which the datatype property belongs), Range (the datatype which is assigned) and the restrictions (allowed values for the datatype) can be defined.

Similarly the database table Parcels is imported from the MySQL database to the parcels ontology in Protégé using the Datamaster plugin with the options import table as class and import table content. The *Parcel* class is specified as superclass for the table class which is created when the table is imported into the ontology. The attributes of the table parcel i.e. village\_id, land\_use, conversion are now defined as datatype properties.

### 5.3 SPATIAL ANALYSIS

Protégé does not support spatial data types like point, geometry, linestring, polygon. So there is no way that spatial analysis could be done on the ontologies which are in turn connected to the database.

An IDL program as shown in Appendix A.2 was written which solves analysis on the spatial problem as stated in the section 4.1. The flowchart is depicted in Figure 5.3.

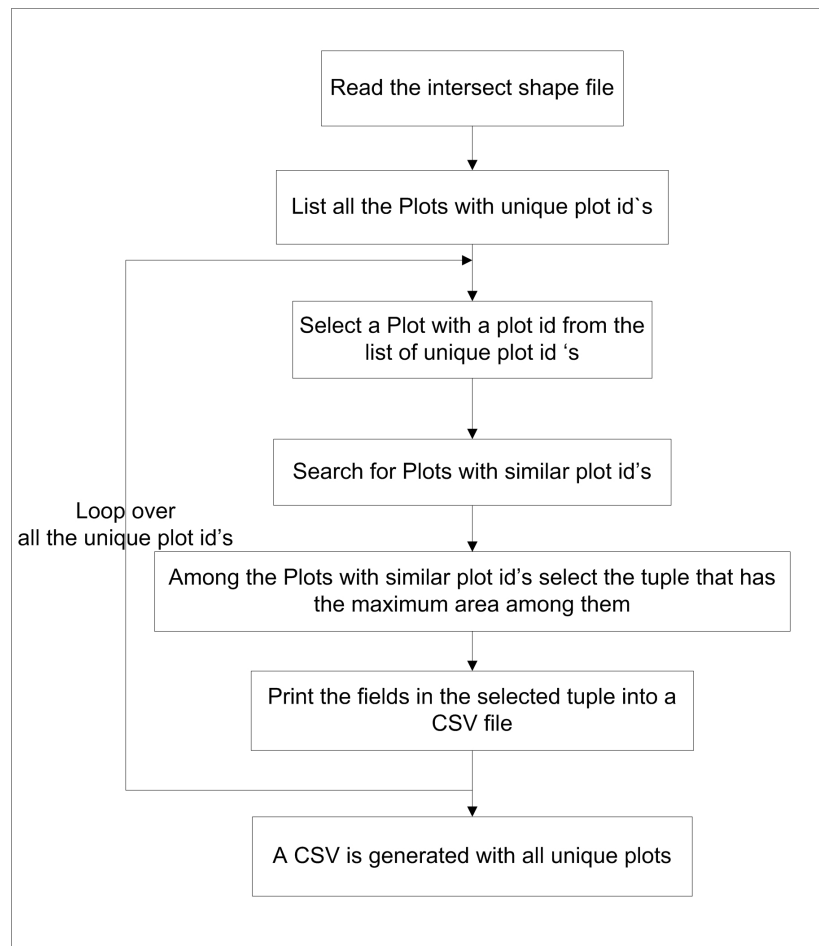


Figure 5.3: The Flow chart of the IDL program.

Initially an intersect tool is run on the two datasets CM and DSSLR in Arc GIS which produces an intersect shape file with the corresponding data in it as shown in Figure 5.4. This intersect shape file is fed as input into the IDL program. The Plots that have unique plot id's are listed. A plot id is selected from the list of unique plot id's. The plots with similar plot id's are selected among which, the tuple with the maximum plot area is selected. The selected tuple is written into a CSV file. The loop continues for all the unique plot id's. Finally a CSV file is generated as the output that has only unique plot id's. The following is an example, that depicts how the IDL program works.

The intersect shape file shows where the plot lies and the number of parcels it lies in. Therefore multiple records are created for the same plot id, with the amount of area that lies in the respective parcels. For example as shown in Figure 5.4 the first three tuples have the same plot id S01BAB001 and amount of area(Shape\_Area) of this plot that lie in parcel id 59 is 14406.62 m<sup>2</sup>, the amount of area(Shape\_Area) of this plot that lie in parcel id 58 is 865.91 m<sup>2</sup> and the amount of area(Shape\_Area) of this plot that lie in parcel id 184 is 1115.38 m<sup>2</sup>. After reading the shape file the program lists all the Plots with unique plot id's. A Plot with a plot id is selected from the list of unique plot id's. Now the whole table is searched for Plots with the same plot id as the selected Plot. Now among the Plots with the same plot id's select the tuple which has the maximum shape area and print the necessary fields in the tuple into a CSV file. Now the next unique plot is taken and the loop goes on until all the plot id's in the attribute table of the shape file are processed

Owner_nm	Landuse	Area	Plot_id	parcel_id	land_use	conversion	Shape_Area
SAHILA	vac	16387.801807	S01BAB001	59	ag	n	14406.627149
SAHILA	vac	16387.801807	S01BAB001	58	ag	n	865.910512
SAHILA	vac	16387.801807	S01BAB001	184	nag	y	1115.386152
ABHINANDA	vac	13687.730925	S01BYK001	59	ag	n	791.399837
ABHINANDA	vac	13687.730925	S01BYK001	60	nag	y	10856.069944
ABHINANDA	vac	13687.730925	S01BYK001	91	ag	n	7.209828
ABHINANDA	vac	13687.730925	S01BYK001	28	ag	n	219.940463
ABHINANDA	vac	13687.730925	S01BYK001	61	ag	n	16.158592
ABHINANDA	vac	13687.730925	S01BYK001	184	nag	y	1796.947066
PHANINDRA	res	12897.443433	S01BAD001	59	ag	n	419.161364
PHANINDRA	res	12897.443433	S01BAD001	60	nag	y	78.744528
PHANINDRA	res	12897.443433	S01BAD001	91	ag	n	11.040638
PHANINDRA	res	12897.443433	S01BAD001	28	ag	n	11615.38598
PHANINDRA	res	12897.443433	S01BAD001	58	ag	n	186.212659
PHANINDRA	res	12897.443433	S01BAD001	29	ag	n	586.89573
MUSKAN	vac	225.51905	S01BAC003	58	ag	n	225.518629

Figure 5.4: Attribute table of the intersect shape file.

through. A CSV file is generated which has all the Plots with unique plot id's.

The table unite is created in the database with the following attributes given in the Table 5.1

Table 5.1: Attribute table of the unite table.

Class	Description
Plot_idun	Unique plot id
Parcel_idun	Unique parcel id
Parcel_landuse	Landuse of the parcel
Conversion_un	The conversion status of the parcel id
Plot_area	Area of the Plot
Village_nameun	Village name
Village_idun	Village id
Plot_landuse	Landuse of the plot
Owner_nameun	owner name of the plot

*Plot\_idun* is defined as the primary key in the unite table. Now the generated CSV file from the IDL program is loaded into the unite table.

An ontology unite is created which imports both the plots and parcels ontology. Now a class called *Unite* is created. Using the datamaster plugin the table called as unite which is generated from the IDL program is imported into the unite ontology.

## Chapter 6

# Results and Discussion

This chapter gives description of the achieved results from this research study, discussion on the results, conclusions drawn from the results and discussion and finally the future recommendations.

### 6.1 THE ROLE OF SWRL AND SQWRL IN RETRIEVAL OF INSTANCES

Semantic Web Rule Language (SWRL) is an OWL-based expressive rule language. The written rules can be expressed in terms of OWL concepts, which provide powerful deductive reasoning capabilities than OWL. The SWRL rules are such that only variables that occur in the antecedent may occur in the consequent. The variable names in an SWRL rule may not be the same as the name of OWL class, property or individual in the same ontology. The variables used in a rule can be used in other rules since the variables are local to the scope of that rule. The variables of SWRL are preceded by a “?” in the SWRL rules for example ?a.

The SWRL Tab is an environment in which the SWRL rules are written in Protégé. With the help of the Jess rule engine the query on the SWRL rules can run in the SQWRL tab which is a sub-tab in SWRL tab.

### 6.2 RESULTS

#### 6.2.1 Retrieval of Information using Rules

The SWRL rule is used to retrieve the information that is present in the ontology in the form of instances. The following rules are written in both the plots ontology and the parcels ontology. When the ontologies are imported into the new ontology unite, all the SWRL rules also get imported.

SWRL Builtins provide flexibility for different implementations. In this research the built-ins for comparisons like `swrlb:equal`, `swrlb:lessThanOrEqual` and `swrlb:greaterThan` are used to infer some instances.

The following rules are written in the SWRL tab and they are run in the SQWRL tab.

These are rules written for the plots ontology.

- To infer the plots whose area is less than 27 m<sup>2</sup> and are assigned to the class *Tax\_slab1*.

**Rule1:** `Plot(?p) ∧ db:Owner_nm(?p, ?own) ∧ db:Plot_id(?p, ?pl) ∧ db:Landuse(?p, ?land) ∧ swrlb:lessThanOrEqual(?a, 27.0) ∧ db:Area(?p, ?a) → Tax_slab1(?p)`

A variable ?p is created for the class Plot. The variable ?p has the datatype property (*db:Owner\_nm*) element ?own. Variable ?p has the datatype property (*db:Plot\_id*) element ?pl. Variable ?p has the datatype property (*db:Landuse*) element ?land. Variable ?p has the datatype property (*db:Area*) element ?a. The SWRL built-ins for comparison

swrlb:lessThanOrEqual is used to compare, whether the value of ?a is less than the value 27. All these statements are conjugated which is implied that the result of the conjugated statements belong to the class *Tax\_slab1*.

To retrieve the instances that belong to the class *Tax\_slab1*, the operator sqwrl:select is used.

$\text{Tax\_slab1}(\text{?p}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?plotid}) \longrightarrow \text{sqwrl:select}(\text{?plotid})$

To retrieve the plot id's of the instances inferred in the class *Tax\_slab1*, the above query is written. Here ?p is a variable of class *Tax\_slab1* and ?p has the datatype property (*db:plot\_id*) element ?plotid which is implied to show all the plot id's of the inferred instances in the class *Tax\_slab1*.

The result as shown in Figure 6.1 is obtained

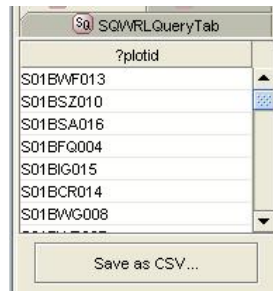


Figure 6.1: The result of Rule 1.

The Rule 2, 3, 4 and 5 infer similar kind of results which are associated to *Tax\_slab2*, *Tax\_slab3*, *Tax\_slab4* and *Tax\_slab5* respectively.

- To infer the plots whose area is more than 27 m<sup>2</sup> and less than 46 m<sup>2</sup> and are assigned to the class *Tax\_slab2*.

**Rule2:**  $\text{Plot}(\text{?p}) \wedge \text{db:Owner\_nm}(\text{?p}, \text{?own}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?pl}) \wedge \text{db:Landuse}(\text{?p}, \text{?land}) \wedge \text{swrlb:lessThanOrEqual}(\text{?a}, 46.0) \wedge \text{swrlb:greaterThan}(\text{?a}, 27) \wedge \text{db:Area}(\text{?p}, \text{?a}) \longrightarrow \text{Tax\_slab2}(\text{?p})$

To retrieve the instances that belong to the class *Tax\_slab2*, the operator sqwrl:select is used.

$\text{Tax\_slab2}(\text{?p}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?plotid}) \longrightarrow \text{sqwrl:select}(\text{?plotid})$

To retrieve the plot id's of the instances inferred in the class *Tax\_slab2*, the above query is written. Here ?p is a variable of class *Tax\_slab2* and ?p has the datatype property (*db:Plot\_id*) element ?plotid which is implied to show all the plot id's of the inferred instances in the class *Tax\_slab2*.

The result as shown in Figure 6.2 is obtained.

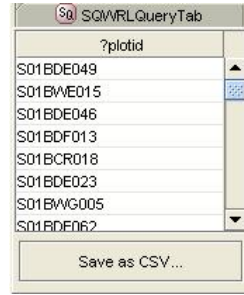


Figure 6.2: The result of Rule 2.

- To infer the plots whose area is more than 46 m<sup>2</sup> and less than 92 m<sup>2</sup> and are assigned to the class *Tax\_slab3*.

**Rule3:**  $\text{Plot}(\text{?p}) \wedge \text{db:Owner\_nm}(\text{?p}, \text{?own}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?pl}) \wedge \text{db:Landuse}(\text{?p}, \text{?land}) \wedge \text{swrlb:lessThanOrEqual}(\text{?a}, 92.0) \wedge \text{swrlb:greaterThan}(\text{?a}, 46) \wedge \text{db:Area}(\text{?p}, \text{?a}) \longrightarrow \text{Tax\_slab3}(\text{?p})$

To retrieve the instances that belong to the class *Tax\_slab3*, the operator `sqwrl:select` is used.

$\text{Tax\_slab3}(\text{?p}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?plotid}) \longrightarrow \text{sqwrl:select}(\text{?plotid})$

To retrieve the plot id's of the instances inferred in the class *Tax\_slab3*, the above query is written. Here *?p* is a variable of class *Tax\_slab3* and *?p* has the datatype property (*db:Plot\_id*) element *?plotid* which is implied to show all the plot id's of the inferred instances in the class *Tax\_slab3*.

The result as shown in Figure 6.3 is obtained.

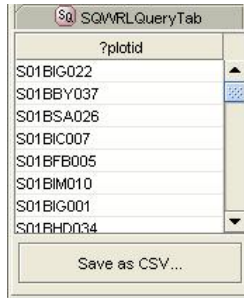


Figure 6.3: The result of Rule 3.

- To infer the plots whose area is more than 92 m<sup>2</sup> and less than 139 m<sup>2</sup> and are assigned to the class *Tax\_slab4*.

**Rule4:**  $\text{Plot}(\text{?p}) \wedge \text{db:Owner\_nm}(\text{?p}, \text{?own}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?pl}) \wedge \text{db:Landuse}(\text{?p}, \text{?land}) \wedge \text{swrlb:lessThanOrEqual}(\text{?a}, 139) \wedge \text{swrlb:greaterThan}(\text{?a}, 92) \wedge \text{db:Area}(\text{?p}, \text{?a}) \longrightarrow \text{Tax\_slab4}(\text{?p})$

To retrieve the instances that belong to the class *Tax\_slab4*, the operator `sqwrl:select` is used.

$\text{Tax\_slab4}(\text{?p}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?plotid}) \longrightarrow \text{sqwrl:select}(\text{?plotid})$

To retrieve the plot id's of the instances inferred in the class *Tax\_slab4*, the above query is written. Here *?p* is a variable of class *Tax\_slab4* and *?p* has the datatype property

(*db:Plot\_id*) element ?plotid which is implied to show all the plot id's of the inferred instances in the class *Tax\_slab4*.

The result as shown in Figure 6.4 is obtained.

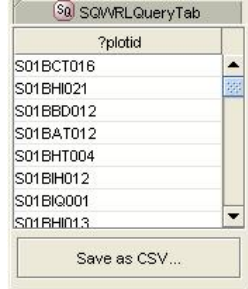


Figure 6.4: The result of Rule 4.

- To infer the plots whose area is more than 139 m<sup>2</sup> and are assigned to the class *Tax\_slab5*.

**Rule5:**  $\text{Plot}(\text{?p}) \wedge \text{db:Owner\_nm}(\text{?p}, \text{?own}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?pl}) \wedge \text{db:Landuse}(\text{?p}, \text{?land}) \wedge \text{swrlb:greaterThan}(\text{?a}, 139) \wedge \text{db:Area}(\text{?p}, \text{?a}) \longrightarrow \text{Tax\_slab5}(\text{?p})$

To retrieve the instances that belong to the *Tax\_slab5*, the operator *sqwrl:select* is used.

$\text{Tax\_slab5}(\text{?p}) \wedge \text{db:Plot\_id}(\text{?p}, \text{?plotid}) \longrightarrow \text{sqwrl:select}(\text{?plotid})$

To retrieve the plot id's of the instances inferred in the class *Tax\_slab5*, the above query is written. Here ?p is a variable of class *Tax\_slab5* and ?p has the datatype property (*db:Plot\_id*) element ?plotid which is implied to show all the plot id's of the inferred instances in the class *Tax\_slab5*.

The result as shown in Figure 6.5 is obtained.

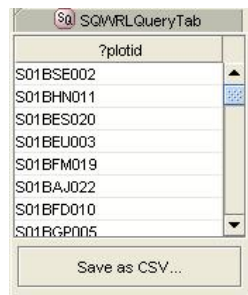


Figure 6.5: The result of Rule 5.

These are the rules written for Parcels ontology .

- To infer the parcels which have landuse use as nag(non agriculture).

**Rule6:**  $\text{Parcel}(\text{?p}) \wedge \text{db:parcel\_id}(\text{?p}, \text{?parcelid}) \wedge \text{db:land\_use}(\text{?p}, \text{?landuse}) \wedge \text{swrlb:equal}(\text{?landuse}, \text{"nag"}) \longrightarrow \text{sqwrl:select}(\text{?landuse}, \text{?parcelid}, \text{?p})$



A variable ?p is created for the class Parcel. The variable ?p has the datatype property (db:parcel\_id) element ?parcelid. Variable ?p has the datatype property (db:land\_use) element ?landuse. The SWRL built-in for comparison, swrlb:equal is used to compare, whether ?landuse is equal to the value “nag.” All these statements are conjugated, which is implied that the result of the conjugated statements is the list of parcel id’s whose corresponding landuse is “nag.”

The result as shown in Figure 6.6 is obtained

?landuse	?parcelid	?p
nag	23	db:parcels_Instance_35
nag	4	db:parcels_Instance_56
nag	24	db:parcels_Instance_36
nag	98	db:parcels_Instance_85
nag	187	db:parcels_Instance_29
nag	8	db:parcels_Instance_74
nag	25	db:parcels_Instance_37
nag	35	db:parcels_Instance_48
nag	85	db:parcels_Instance_79
nag	36	db:parcels_Instance_49
nag	9	db:parcels_Instance_80
nag	178	db:parcels_Instance_25
nag	60	db:parcels_Instance_71

Figure 6.6: The result of Rule 6.

The Rule 7 infer similar type of results with the landuse as ag(agriculture)

- To infer the parcels which have landuse use as ag(agriculture).

**Rule7:**  $\text{Parcel}(\text{?p}) \wedge \text{db:parcel\_id}(\text{?p}, \text{?parcelid}) \wedge \text{db:land\_use}(\text{?p}, \text{?landuse}) \wedge \text{swrlb:equal}(\text{?landuse}, \text{“ag”}) \longrightarrow \text{sqwrl:select}(\text{?landuse}, \text{?parcelid}, \text{?p})$

The result as shown in Figure 6.7 is obtained

?landuse	?parcelid	?p
ag	107	db:parcels_Instance_8
ag	38	db:parcels_Instance_53
ag	26	db:parcels_Instance_38
ag	40	db:parcels_Instance_57
ag	158	db:parcels_Instance_17
ag	1	db:parcels_Instance_1
ag	31	db:parcels_Instance_44
ag	149	db:parcels_Instance_13
ag	47	db:parcels_Instance_64
ag	59	db:parcels_Instance_69
ag	37	db:parcels_Instance_52
ag	39/1A	db:parcels_Instance_55
ag	99	db:parcels_Instance_86

Figure 6.7: The result of Rule 7.

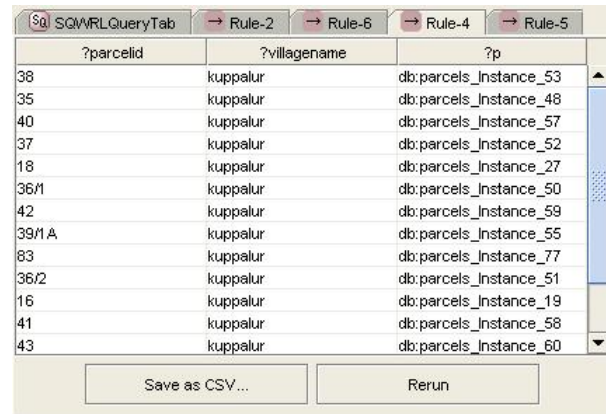
- To infer the parcels which belong to the village “kuppalur.”



**Rule8:**  $\text{Parcel}(\text{?p}) \wedge \text{db:parcel\_id}(\text{?p}, \text{?parcelid}) \wedge \text{db:village\_nm}(\text{?p}, \text{?villagename}) \wedge \text{swrlb:equal}(\text{?villagename}, \text{"kuppalur"}) \longrightarrow \text{sqwrl:select}(\text{?parcelid}, \text{?villagename}, \text{?p})$

A variable ?p is created for the class Parcel. The variable ?p has the datatype property (db:parcel\_id) element ?parcelid. Variable ?p has the datatype property (db:village\_nm) element ?villagename. The SWRL built-in for comparison, swrlb:equal is used to compare, whether ?villagename is equal to the value "kuppalur." All these statements are conjugated, which is implied that the result of the conjugated statements is the list of parcel id's whose corresponding village name is "kuppalur."

The result as shown in Figure 6.8 is obtained



?parcelid	?villagename	?p
38	kuppalur	db:parcels_Instance_53
35	kuppalur	db:parcels_Instance_48
40	kuppalur	db:parcels_Instance_57
37	kuppalur	db:parcels_Instance_52
18	kuppalur	db:parcels_Instance_27
36/1	kuppalur	db:parcels_Instance_50
42	kuppalur	db:parcels_Instance_59
39/1 A	kuppalur	db:parcels_Instance_55
83	kuppalur	db:parcels_Instance_77
36/2	kuppalur	db:parcels_Instance_51
16	kuppalur	db:parcels_Instance_19
41	kuppalur	db:parcels_Instance_58
43	kuppalur	db:parcels_Instance_60

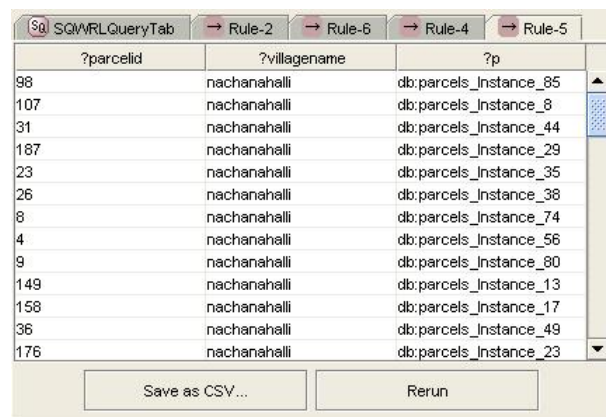
Figure 6.8: The result of Rule 8.

The Rule 9 infer similar type of results with the different village name as nachanahalli.

- To infer the parcels which belong to the village "nachanahalli."

**Rule9:**  $\text{Parcel}(\text{?p}) \wedge \text{db:parcel\_id}(\text{?p}, \text{?parcelid}) \wedge \text{db:village\_nm}(\text{?p}, \text{?villagename}) \wedge \text{swrlb:equal}(\text{?villagename}, \text{"nachanahalli"}) \longrightarrow \text{sqwrl:select}(\text{?parcelid}, \text{?villagename}, \text{?p})$

The result as shown in Figure 6.9 is obtained



?parcelid	?villagename	?p
98	nachanahalli	db:parcels_Instance_85
107	nachanahalli	db:parcels_Instance_8
31	nachanahalli	db:parcels_Instance_44
187	nachanahalli	db:parcels_Instance_29
23	nachanahalli	db:parcels_Instance_35
26	nachanahalli	db:parcels_Instance_38
8	nachanahalli	db:parcels_Instance_74
4	nachanahalli	db:parcels_Instance_56
9	nachanahalli	db:parcels_Instance_80
149	nachanahalli	db:parcels_Instance_13
158	nachanahalli	db:parcels_Instance_17
36	nachanahalli	db:parcels_Instance_49
176	nachanahalli	db:parcels_Instance_23

Figure 6.9: The result of Rule 9.

Analysis with respect to the Unite ontology is described here. The following rule is written in SWRL tab which is run with the help of SQWRL sub-tab.

This rule retrieves the information of Plot (S01BAT009) and identifies to which Parcel id the Plot belongs to, since it lies in multiple land parcels.

**Rule10:** unite(?p)  $\wedge$  j.0:plot\_idun(?p, ?plotid)  $\wedge$  swrlb:equal(?plotid, "S01BAT009")  $\wedge$  j.0:conversion\_un(?p, ?conversion)  $\wedge$  j.0:parcel\_landuse(?p, ?parcellanduse)  $\wedge$  j.0:plot\_landuse(?p, ?plotlanduse)  $\wedge$  j.0:owner\_nameun(?p, ?ownername)  $\wedge$  j.0:village\_idun(?p, ?villageid)  $\wedge$  j.0:village\_nameun(?p, ?villagename)  $\wedge$  j.0:plot\_idun(?p, ?parcelid)  $\wedge$  j.0:plot\_area(?p, ?plotarea)  $\rightarrow$  sqwrl:select(?plotid, ?ownername, ?plotlanduse, ?plotarea, ?parcelid, ?conversion, ?parcellanduse?villageid?villagename)

A variable ?p is created for the class *Unite*. The variable ?p has the datatype property (*j.0:plot\_un*) element ?plotid. The SWRL built-in for comparison, swrlb:equal is used to compare, whether ?plotid is equal to the string "S01BAT009." Variable ?p has the datatype property (*j.0:conversion\_un*) element ?conversion. Variable ?p has the datatype property (*j.0:parcel\_landuse*) element ?parcellanduse. Variable ?p has the datatype property (*j.0:plot\_landuse*) element ?plotlanduse. Variable ?p has the datatype property (*j.0:owner\_nameun*) element ?ownername. Variable ?p has the datatype property (*j.0:village\_idun*) element ?villageid. Variable ?p has the datatype property (*j.0:village\_nameun*) element ?villagename. Variable ?p has the datatype property (*j.0:plot\_idun*) element ?parcelid. Variable ?p has the datatype property (*j.0:plot\_area*) element ?plotarea. All these statements are conjugated, which is implied that the result of the conjugated statements contains the list of plot id, area of the plot, landuse of the plot, owner name of the plot, parcel id, landuse of the parcel, conversion status, village name and the village id of the inferred instance.

The result as shown in Figure 6.10 is obtained

?plotid	?ownername	?plotlanduse	?plotarea	?parcelid	?conversion	?parcelland...	?villageid	?villagename
S01BAT009	PADMAVATI	res	111.71236	S01BAT009	n	ag	001	nachanahalli

Figure 6.10: The result of Rule 10.

The Figure 6.11 depicts the Plot with plot id S01BAT009 which is lying in multiple parcels P1, P2 and P3. The plot occupies the areas A1, A2 and A3 in the parcels P1, P2 and P3 respectively. Since the area A2 is greater than the other areas occupied in other parcels the plot is assigned to the parcel P2 with all its attributes.

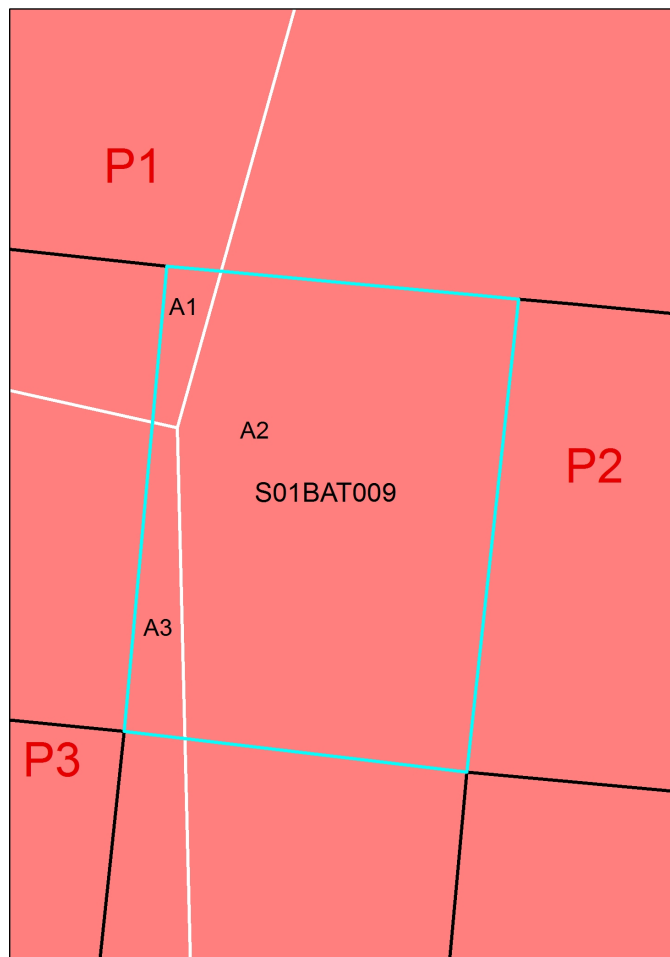


Figure 6.11: The Plot S01BAT009 lying in Parcels P1, P2 and P3, and also occupying the area A1, A2 and A3 in the respective parcels.

The Figure 6.12 shows the intersect shape file attribute table where the attribute Shape\_Area depicts the amount of area which lies in the different parcels with parcel id's 28, 27 and 31. Hence the tuple which has the maximum Shape\_area i.e. 98.348 is retrieved with the corresponding attributes parcel\_id 27, land\_use ag (agricultural), and conversion n (no). This is same as the tuple that is retrieved by running Rule 10 in SQWRL tab.

Owner_nm	Landuse	Area	Plot_id	village_nm	parcel_id	land_use	conversion	Shape_Area
PADMAVATI	res	111.712361	S01BAT009	nachanahalli	28	ag	n	4.313785
PADMAVATI	res	111.712361	S01BAT009	nachanahalli	27	ag	n	98.348065
PADMAVATI	res	111.712361	S01BAT009	nachanahalli	31	ag	n	9.050501

Record: 0 Show: All Selected Records (3 out of 6097 Selected) Options

Figure 6.12: The attribute table of the intersect shape file.

## 6.3 DISCUSSION ON THE RESULTS

### **How to integrate two different datasets, with different spatial domains at the conceptual level?**

The integration of datasets with different domains can be done using many methods. In this research the ontology method is chosen. Initially the data is analyzed i.e. how it is created, the structure, the elements, the concepts that could be formed etc. The ontology mainly deals with concepts, the relationships between them and the properties. Once the data is analyzed the concepts are formed just like the ones mentioned in Chapter 4. The concepts can be related to each other by importing them into a single ontology which is intended to integrate both the ontologies. The relationships and restrictions can also be defined on them as described in Chapter 4.

The Rule 10 in the previous section shows how the integration works even when there is no support of spatial relations, queries by the ontology. The results could be inferred from this rule since the data from the two datasets are integrated as described in Section 5.3.

### **How can spatial rules be applied on conceptually mapped datasets by a naive user?**

The spatial support on ontologies is in its nascent stages. There is a lot of complexity involved in the rules when spatial comes into the picture. There is a subtle difference between what can be expressed and what cannot be expressed in the ontology that has to be understood by the domain users. There is no support of spatial datatypes in the ontology so the spatial polygons in the form of coordinates cannot be stored. Since there are no spatial datatypes there cannot be any spatial relations which can be posed in the ontology. Furthermore for a naive user it is very difficult to write SWRL rules to get the intended results. For a user to write the query he has to know all the datatype properties and classes. Therefore a user interface should be developed so that it is an easy process and get the intended results just by entering elements into it.

### **How can the resultant related ontologies be used in other applications?**

The resultant related ontology could be used in any application. The unite ontology can be used as a back end to an application which wants to provide information about the plot and parcels which gives results like the ones depicted from Rule 1 to Rule 9 in the above section. The spatial problem as depicted in the section 4.1 has a solution as an ontology at the back end which can run Rule 10 that will deliver the output to the user in the front end. The unite ontology that is created could be integrated with ontologies created by other organizations in order to share data and create a complex system which is conceptually related and there is a list of complex relationships between the concepts.



## Chapter 7

# Conclusions and Recommendations

This chapter describes the conclusion on this research work and the future recommendations which can be worked on to make things better.

### 7.1 CONCLUSIONS

The ontology method is best suited for datasets with no spatial datatypes since there is no support for spatial datatypes in the ontology editors. The analysis on the instances that are imported in the respective ontologies are not possible in the unite ontology since there is no support for spatial analysis. The spatial analysis on the datasets could be done in a better way using other techniques which support them for example the databases Postgres, Mysql, Oracle Spatial and the GIS softwares like Arc GIS, QGIS(Quantum GIS) etc. Integration of datasets with non spatial data could be done using ontology since we can relate the real world phenomenon to the classes, relations and restrictions defined on them. A geospatial database cannot explicitly contain the topological relationships like a Parcel “contains” a Plot. Here a SQL spatial containment operator “contains” is used to see whether the Parcel contains the Plot inside it or not. Pre computing the topological relationships between every object in a geospatial database is impractical since the size of the database would be large. Dolbear et al in [DHG06] states that “the existing reasoners are unable to perform topological inference on the majority of geospatial data as they are unable to compute the necessary topology.”

A front end could be developed where the user enters the plot id and gets the intended results with the “unite” ontology running on the back end. Even the SWRL does not have any spatial built ins which can be used to operate on the spatial datatypes. Support should be provided on the development of spatial built-ins for doing analysis on the geospatial ontologies.

The resultant ontologies can be used in many applications. The related ontology could be used as back end for the application which is built as the front end to solve the problem that the departments like DSSLR and CM are facing. In the future if all the departments consider in creating an ontology a knowledge base and data backend by relating among the ontologies from other departments a huge knowledge base could be created which could be used by all the departments of the government where multiple applications may run on the same ontology at the backend.

### 7.2 RECOMMENDATIONS

For a simple ontology Rule ML can be used as alternative reasoners. Support of spatial datatypes in the ontologies should be looked into, in order to support ontologies related to geospatial domain. The reasoning support on the ontologies related to geospatial domain should also be improved. It is better to use alternative techniques for spatial analysis on the spatial datasets rather than ontology.

Once there is a support of spatial datatypes the tools like JTS (Java Topology Suite) could be looked into which helps in spatial analysis, new tools which are yet to come like GeoSWRL which will have spatial built-in capabilities, Rule ML (markup language) can be used to write rules which provides users to create their own built-ins, along with this OO jDREW is used as reasoners for the Rule ML which will provide the reasoning capability for it. Other appropriate reasoners should be looked into which supports spatial datatypes and spatial analysis.

## Appendix A

# Tools for Semantic Modelling

### A.1 ONTOLOGY EDITING TOOLS

There are many Ontology Editing Tools of which some are listed here.

Table A.1: Ontology editors

Ontology editors	Protégé	OIL Ed	OntoEdit	WebODE	Ontolingua
Developers	SMI	University of Manchester	Ontoprise	UPM	KSL
Pricing policy	Open source	Free ware	Free ware and licensed	Free web access license	Free web access
Extensibility	Plugins	No	Plugins	Plugins	none
Inference services	FaCT, Pellet, Racer Pro	FaCT	OntoBroker	Prolog	none
Usability graphical taxonomy	Yes	No	No	Yes	Yes

### A.2 THE IDL PROGRAM TO ANALYSE THE PLOT PARCEL PROBLEM

```
pro shr_parce_plot
```

```
file = 'D:\idl\final dataset\intersect.shp'
OUTPUT = 'D:\idl\result\re.csv'
myShape=OBJ_NEW('IDLffShape', file)
attrNew = myshape ->GetAttributes(/all)
nEntities = n_elements(attrNew)
help, attrNew, /str
```

```
myShape->GetProperty, ATTRIBUTE_NAMES=attr_names
print, attr_names ; this is to print all the attributes in the shape file
```

```
OBJ_DESTROY, myShape
```



```

PLOTS = attrNew.ATTRIBUTE_5 ; attribute assigned to the variable
PARCELS = attrNew.ATTRIBUTE_11 ; attribute assigned to the variable
LANDUSE_PARCEL = attrNew.ATTRIBUTE_12 ; attribute assigned to the variable
AREAS = attrNew.ATTRIBUTE_17 ;this is a variable used in the program
CONVERSION = attrNew.ATTRIBUTE_13 ; attribute assigned to the variable
AREA = attrNew.ATTRIBUTE_4 ; attribute assigned to the variable
VILLAGE_NAME = attrNew.ATTRIBUTE_10 ; attribute assigned to the variable
VILLAGE_ID = attrNew.ATTRIBUTE_9 ; attribute assigned to the variable
LANDUSE_PLOT = attrNew.ATTRIBUTE_3 ; attribute assigned to the variable
OWNER_NAME = attrNew.ATTRIBUTE_2 ; attribute assigned to the variable

UNIQ_PLOTS = PLOTS[UNIQ(PLOTS, SORT(PLOTS))]
nUNIQ_PLOTS = n_elements(UNIQ_PLOTS)

OPENW, UNIT, OUTPUT, /GET_LUN
PRINTF, UNIT, 'PLOT ID,PARCEL ID,LANDUSE_PARCEL,CONVERSION,AREA_PLOT,
      VILLAGE_NAME,VILLAGE_ID,LANDUSE_PLOT,OWNER_NAME' ;first line csv file

for i = 0, n_elements(UNIQ_PLOTS)-1 do begin      ; begin for loop
    index = where(PLOTS eq UNIQ_PLOTS[i], count)
    if count lt 1 then begin                      ; begin if
        print, 'I did not find this plot ID', UNIQ_PLOTS[i], '. Exiting.'
        return
    endif                                         ; end if
    areas_for_comparison = AREAS[index]
    temp = MAX( areas_for_comparison, Max_Subscript);tuple with the maximum area
        PRINTF,UNIT,UNIQ_PLOTS[i],',',',',PARCELS[index[Max_Subscript]],',',',
LANDUSE_PARCEL[index[Max_Subscript]],',',',CONVERSION[index[Max_Subscript]],',',',
AREA[index[Max_Subscript]],',',',VILLAGE_NAME[index[Max_Subscript]],',',',
VILLAGE_ID[index[Max_Subscript]],',',',LANDUSE_PLOT[index[Max_Subscript]],',',',
OWNER_NAME[index[Max_Subscript]] ; print all the attributes in the csv file
endfor
CLOSE, UNIT & FREE_LUN, UNIT
PRINT, 'COMPLETE. CHECK OUTPUT'
end

```

## Appendix B

# Terminology

Administrative boundaries of geographical land areas in the Indian subcontinent consist of 28 States and seven union territories. Each state consists of a number of districts, each district having a number of taluks and each taluk having a number of villages. Town, sector, block are also administrative boundaries within a town area. The Administrative Head of the taluk maintains the cadastral land records of both village (agricultural parcels and habitation areas) and urban areas (towns and bigger cities).

**Village:** A village is the smaller administrative unit, it is the spatial aggregation of various types of land parcels like dwelling areas, forest, streams and rivers, hills and valleys, government owned land parcels, public land parcels, community land parcels and privately owned land parcels. There are about 600,000 villages in India. Generally total village area varies from 1 km<sup>2</sup> to 20 km<sup>2</sup>. In a normal village, majority of the area is used for cultivation purposes and they are called agricultural parcels. Almost all agricultural parcels are privately owned. These agricultural parcels along with other lands adjoining them, will get converted to urban areas as and when there is a pressure due to increase in population. Village loses its characteristics and the village area becomes included in the urban town limits.

**Parcels:** Many parcels aggregate to form a village. It is represented by its identity number called as “parcel-id” which is unique in a village. Maximum number of land parcels in any village is owned by the private individuals and are mainly used for agricultural purpose. Individual parcel boundaries of a particular village are measured and demarcated inside the bigger boundary of the village. The number of land parcels in a village varies from 100 to 2,000 for a village. Cadastral land records are maintained for these parcels, but they are not up-to-date. Hence the government is facing problems in collecting the conversion charges.

**Town:** A town consists of many sectors, each sector having many blocks and road networks. Each block consist plots of various land use.

**Sector:** Many sectors aggregate to form a town. The entire town is conveniently divided into many sectors. Each sector consists of many blocks and also includes road network, water lines, streams, rivers, drainages and other public land parcels.

**Block:** A block consists of the plots with different varieties of land use. It is usually surrounded by the roads and drainages.

**Plot:** A plot is a piece of land with a specific area and generally related to a person through certain rights. A plot can be used for any land use including residential. Most of the residential plots are owned by private individuals. Some plots are public lands like parks, community lands, playgrounds, forests, lakes, water bodies, etc. Most of the residential plots are of some standard sizes like, 40\*30 feet, 50\*80 feet, and bigger. Majority of the plots are used for residential purposes.

## LIST OF REFERENCES

---

- [AvH09] G. Antoniou and F. van Harmelen. Web ontology language: Owl. In *Handbook on Ontologies*. Springer - Verlag, Berlin, Heidelberg, 2009.
- [BHS09] F. Baader, I. Horrocks, and U. Sattler. Description logics. In *Handbook on Ontologies*. Springer - Verlag, Berlin, Heidelberg, 2009.
- [BN03] F. Baader and W. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors, *The description logic handbook : theory, implementation, and applications*. Cambridge University Press, Newyork, 2 edition, 2003.
- [DHG06] C. Dolbear, G. Hart, and J. Goodwin. What owl has done for geography and why we don't need it to map read. page 4, 2006.
- [GDD09] D. Gašević, D. Djurić, and V. Devedžić. *Model Driven Engineering and Ontology Development*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [GN87] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
- [GOS09] N. Guarino, D. Oberle, and S. Staab. What is an ontology? In *Handbook on Ontologies*. Springer - Verlag, Berlin, Heidelberg, 2009.
- [Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5:199–220, 1993.
- [Hal06] M. Hall. A semantic similarity measure for formal ontologies(with an application to ontologies of a geographic kind). Master's thesis, Alpen-Adria Universitat Klagenfurt, 2006.
- [HPSB<sup>+</sup>04] I. Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosz, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml, May 2004.
- [Kie08] B. Kieler. Semantic data integration across different scales: Automatic learning of generalization rules. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII. Part B2:685–690, 2008.
- [Kip10] K. W. Kipngénoh. Ontology mapping for geoinformation integration. Master's thesis, ITC, 2010.
- [Kok06] M. Kokla. Guidelines on geographic ontology integration. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 36:67–72, July 2006.
- [MMP<sup>+</sup>07] L. Ma, J. Mei, Y. Pan, K. Kulkarni, A. Fokoue, and A. Ranganathan. Semantic web technologies and data management, 2007.

- [NM01] N. F. Noy and D. L. McGuinness. *Ontology development 101: A guide to creating your first ontology*, 2001.
- [Pan09] J. Z. Pan. Resource description framework. In *Handbook on Ontologies*. Springer - Verlag, Berlin, Heidelberg, 2009.
- [pro] Protégé.
- [Rag07] A. Ragone. Owl-dl as a power tool to model negotiation mechanisms with incomplete information. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Richiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 941–945. Springer Berlin / Heidelberg, 2007.
- [SLKB06] J.E. Stoter, R.L.G. Lemmens, B.J. Kobben, and N. Bakker. Semantic data integration in a multiple representation environment. In: *Proceedings of Joint ISPRS working groups II/3 and II/6 workshop on multiple representation and interoperability of spatial data, Hannover*, page 8, February 2006.
- [SSS91] M. Schmidt-Schaulß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1 – 26, 1991.
- [UvOMM99] H. Uitermark, P. van Oosterom, N. Mars, and M. Molenaar. Ontology-based geographic data set integration. In *Spatio-temporal database management*, pages 60–78. Springer, 1999.
- [w3c04] Owl web ontology language guide, February 2004.
- [w3c09] Owl 2 web ontology language document overview, October 2009.
- [ZDY<sup>+</sup>07] P. Zhao, L. Di, W. Yang, G. Yu, and P. Yue. Geospatial semantic web: critical issues. *Encyclopedia of Geoinformatics*. Idea Group Publishing, Hershey, pages 178–187, 2007.