Crop Disease Analysis Using Ontology and SPARQL

Submitted by:
Rehnuma Shakil
ID: 2015-2-96-001
Dept. of Computer Science & Engineering
East West University
Dhaka.

Supervised by:
Dr. Mohammad Rezwanul Huq
Assistant Professor
Dept. of Computer Science & Engineering
East West Engineering
Dhaka.

This project has been submitted to the Dept. of Computer Science & Engineering, East West University in the partial fulfillment of the requirement for the degree of Masters of Science in Computer Science & Engineering.

East West University
Dept. of Computer Science & Engineering
Spring 2017

Declaration

This project has been submitted to the Dept. of Computer Science & Engineering, East West University in accordance with the partial fulfilment of the requirement for the degree Masters of Science in Computer Science & Engineering, performed by me, Rehnuma Shakil, ID: 2015-2-96-001, under supervision of Dr. Mohammad Rezwanul Huq, Assistant Professor, Dept. of Computer Science & Engineering, East West University. This project work has been completed under the course work Master’s Project, CSE 597.

I, hereby declare that, this project has not been submitted elsewhere for the requirement of any other degrees or diploma or any other purpose.

Signature of the candidate

Rehnuma Shakil

Letter of Acceptance

This project is entitled "Crop Disease Analysis Using Ontology and SPARQL" submitted by Rehnuma Shakil, ID: 2015-2-96-001, to the Dept. of Computer Science & Engineering, East West University, Dhaka-1212, Bangladesh is accepted by the department in order to the partial fulfilment of requirement for the degree of Masters of Science in Computer Science & Engineering, April 2017.

Board of Examiners:

Supervisor:

Dr. Mohammad Rezwanul Huq
Assistant Professor
Dept. of Computer Science & Engineering
East West University
Dhaka, Bangladesh.

Chairperson:

Dr. Ahmed Wasif Reza
Associate Professor & Chairperson (Acting)
Dept. of Computer Science & Engineering
East West University
Dhaka, Bangladesh.

Acknowledgement

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this project. First and foremost, my thanks and gratitude to the Almighty, for His immeasurable grace and profound kindness, I have been successful in completing all my works.

I would like to express my deepest gratitude to my academic advisor, Dr. Mohammad Rezwanul Huq, Assistant Professor, Dept. of Computer Science & Engineering, East West University, for his excellent guidance, patience and providing me with all the necessary support for the successful completion of this project work.

I would never have been able to finish this work without the constant guidance and encouragement from my parents. Successful completion of any task would be incomplete without them. Their ceaseless cooperation and inspiration crowns all efforts with success.

Finally, a very special thanks to my friends, classmates and colleagues, who have inspired me and supported me along the way.

Thanks for all your encouragement!

Abstract

Due to the growth of population, it is essential to improve farm productivity to meet the rapidly growing demand for food across the world. Technologies have been used widely to increase the yielding of crops for the highly growing population. Bangladesh has an agriculture based economy. Though majority population of the country lives on agriculture, but sadly, very few of them are aided with the blessings of technology. The knowledge of different factors like weather condition, soil type and particular attributes of the crops to plant is essential to improve the productivity of agricultural crops. The proper management and easy accessibility of this knowledge can increase the production of crops and decrease the loss of crops due to these natural factors. Formally represented knowledge is based on a conceptualization. Ontology is an explicit specification of a conceptualization. It defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. Ontology driven applications face the challenge of integrating multiple formats with the information stored in ontologies.

In this project, we have proposed an ontology-based knowledge management methodology for the diagnosis of soybean diseases using Protégé and SPARQL. A set of new ontology concepts and properties has been defined to build the conceptual model for the attributes that describes the soybean plant, the factors that defines the soybean diseases and the relation between them. Finally, we develop a web application for soybean disease analysis using a SPARQL endpoint at the backend.

Contents

Chapter 1

Introduction

1.1 Background Study

Agriculture is the backbone of Bangladesh. Majority of the population lives on agriculture and agricultural products. To keep up with the increasing demand of food supply for the highly growing population, technology must be used to increase the production of crops. Traditional method of farming is used in most part of the country. These methods are having drawbacks like fewer yields of crops and crop loss due to diseases.

Information and communication technologies (ICT) have significant importance in our lives across several domains. Agriculture is no exception to that. The key characteristics of ICT are the dissemination, access and exchange of information. ICT could play a vital role to increase the production of agricultural products by providing relevant information to agronomists. Appropriate management of information could help them benefit from proper planning and decision making.

The agriculture domain is huge and vast. Various aspects have to be integrated to build a fully functioning system with all the information related to agriculture. Taking the soybean crop as example, not only using pesticides alone can prevent its diseases; there are other factors which can cause diseases such as soil status, weather conditions, condition of different plant attributes during different times of a year, etc. Due to the lack of integration between such heterogeneous data, extraction of single information like which time of the year is likely to be attacked by a particular disease is difficult, which results into reduced production of soybean.

Building ontology is a complex task. An expert opinion is required to determine the dataset on a particular domain. This project is based on the existing dataset of soybean disease analysis mentioned on "**Learning on being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis**" by **R. S. Michalsky** and **R. L. Chilausky**, January 30, 1980. In our work, we have developed ontology using Protégé for defining the attributes and factors that describes a soybean disease and define the relation between them. H2 database is used as the source of the individual instances. This data source has been mapped to the ontology using an open-source Ontology Based Data Access (OBDA) system, *Ontop* that allows querying relational data sources through a conceptual representation of the domain of interest, provided in terms of ontology, to which data sources are mapped. We prototype a web-based application to take the factors and attributes as user inputs and make a disease analysis based on the inputs provided.

1.2 Objective

The main objective of this project is to develop a knowledge based system using ontology that consists of a user friendly platform to diagnose crop diseases, provided not only the conditions of the crop plants but also the weather conditions, by the users.

In this project, we have developed a system for the diagnosis of soybean plant disease. A question may rise, why choose soybean plant which is not much popular in Bangladesh and there are other more important crops in our country which faces loss in yields every year due to diseases. We have taken the soybean plant as an example; an expert derived set of rules and a training data set is available for soybean disease (R.S.Michalski & R.L.Chilauski). We have developed a system for soybean disease analysis based on these rules, trained it with the available training data set; working with these actual data set has provided a certain level of validity to the project. And also if real data set for other important crops of our country is provided, we can reuse our ontology to develop a similar system for the diagnosis of those crops.

1.3 Relational Database vs. Ontology Vocabulary

Question may rise why data is stored in ontology rather than in relational database. Relational database is a computer database in which all data is stored in relations which are tables with rows and columns. Each table is composed of records called tuples and each record is identified by a field called attribute containing a unique value. Every table shares at least one field with another table in one-to-one, one-to-many or many-to-many relationships. These relationships allow the database user to access the data in almost an unlimited number of ways and to combine the tables as building blocks to create complex and very large databases.

Ontology, in the other hand, is a set of concepts and categories in a subject area or domain that shows their properties and the relationship between them. Ontologies have become common on the World Wide Web (WWW). The ontologies on the web range from large taxonomies categorizing websites to categorization of products for sale and their features. The WWW consortium (W3C) has developed the Resource Description Framework (RDF), a language for encoding knowledge on web pages to make it understandable to electronic agents searching for information. RDF is a simple language for expressing an RDF statement. RDF/XML is syntax to express an RDF graph as an XML document. RDFS (RDF Schema) extends the vocabulary of simple RDF. An RDF or triple is a statement with three constituent parts – a subject, a predicate and an object. Example:

The sky is blue.

"The sky" is the subject, "is" the predicate or link, "blue" is the object or property. In RDF/XML, each part is defined by a URI (Uniform Resource Identifier).

Ontologies are considered one of the pillars of the semantic web. The semantic web is an extension of the web through standards by W3C consortium. The standards promote common data formats and exchange protocols on the web, most fundamentally the RDF. On the semantic web, vocabularies define the concepts and the relationships used to describe and represent a domain. The role of vocabularies in the semantic web is to help data integration when ambiguities may exist on the terms used in the different data sets, or when a bit of extra knowledge may lead to the discovery of new relationships. Semantic data is assembled in a graph database that is unlike the more common relational and hierarchical databases that have some elements of data that are more important than others. A graph database uses arbitrary object relations with no intrinsic importance.

SPARQL (Search Protocol and RDF Query Language) is a query language for semantic data similar to SQL. Without semantic data, the owners of separate SQL sites would have to decide on a common data format to share information that they could both understand. Unfortunately this requires human intervention. In the semantic data model, the same base ontology for expressing the meaning behind the data is used by different sites that makes it available for SPARQL searching. SPARQL queries RDF datasets, rather like SQL queries in a relational database, except SELECT identifies the triples values selected on the RDF graph that will be returned in the results and where identifies the triple patterns in the RDF graph to be matched against the RDF dataset.

OWL (Web Ontology Language) classifies things in terms of semantics, or meaning, where a class is a group of individuals that share common characteristics. A machine reader knows if an

individual is a member of a class and its object properties or data properties related to that OWL class.
Ontology can provide more meaningful querying interface than database and ontology can use reasoner to find hidden information to get better results. The main advantage of using ontologies is the formalized semantics. In this way, a reasoner can automatically infer new statements without writing specific codes.
In short, we can say:

- Databases have closed world assumption, but ontologies have open world assumption.
- In databases, each individual has a single unique name, but in ontologies, individuals might have more than one name.
- Implicit information can be inferred from ontology, but databases cannot infer.
- The schema in ontology is large and complex, but databases have simple and smaller schema. This implies that, the focus on formal semantics is much stronger in ontologies than in databases. Because the aim of ontologies is to represent meaning rather than data.

In the upcoming chapters, a detailed description of the project work is provided. In chapter 2, we define the problem statement of our project and how we have tried to solve it using ontology vocabulary. A detail description of the soybean ontology is given here. In chapter 3, we propose the system architecture and discuss the different levels in details. In chapter 4, we deal with the implementation process. Finally we conclude in chapter 5 by discussing the limitations of our work and what can be done to improve the system.

Chapter 2

System Specification

2.1 Problem Statement

In this project, we have used the existing large soybean data set for soybean disease analysis (R.S.Michalski and R.L.Chilausky). Our objective is to build an ontology based on the data set mentioned, use it as knowledge base for soybean plant, query the data set, and compare the query results to some expert generated decision rules that could classify a soybean disease.

As mentioned earlier, this task can also be done by storing the data into a relational database. A relational database is capable enough to handle a huge and complex data set. But the semantic content in a web is increasing steadily day by day. Now-a-days, a greater number of web applications are used to produce or consume semantic web content. It brings sentences to the web which enhances the usability and the usefulness of the web and its interconnected resources.

The soybean disease data set that we have used for this project is huge. Diagnosis of soybean disease was selected as the representative of the problems one faces in the diagnosis of the plant diseases in general. The knowledge base was developed with sufficient information to diagnose the following 15 diseases:

1. Diaporthe Stem Canker
2. Charcoal Rot
3. Rhizoctonia Root Rot
4. Phytophthora Root Rot
5. Brown Stem Rot
6. Powdery Mildew
7. Downy mildew
8. Brown Spot
9. Bacterial Blight
10. Bacterial Pustule
11. Purple Seed Stain
12. Anthracnose
13. Phyllostica Leaf Spot
14. Alternaria Leaf Spot
15. Frog Eye Leaf Spot

A description space for diagnosing the selected soybean diseases was developed in collaboration with an expert in soybean pathology. The variables used were 35 plant and environmental descriptors and one decision variable that specifies the diagnosis. Each of these descriptors can have more than one value. They may contain structured domains and maintain hierarchies (parent-child nodes). Relationship among the variables may also exist. All these characteristics make the soybean database a really big and complex one. With all the hierarchies and relationships among the variables, using the relational database to store all these information becomes quite troublesome. A database would require multiple tables, quite a number of one-to-many, many-to-one and many-to-many relations. Accessing data gets even more complicated. The larger the database grows, the more complicated it gets to maintain and query this huge database. Instead, we build an ontology for these 35 plant and environmental descriptors, define the relationship among them in the form of RDF triples. We store the values of the descriptor variables in a simple, flat table in a database. We map these values to the corresponding descriptors defined in the ontology. This makes our data set simpler to understand and easily accessible.

2.2 Soybean Ontology

In this section, we propose the soybean ontology, which consists of the plant and environmental descriptors, the hierarchy of the attributes that describes the plant and environmental descriptors and relationship between them.

The ontology contains two main concepts (classes) – *PlantDescriptor* and *Attribute*.

PlantDescriptor – The concept "PlantDescriptor" defines the different types of plant and environmental descriptors for soybean plant. There are three types of descriptors – Environmental Descriptor, Plant Global Descriptor and Plant Local Descriptor. The hierarchy for this concept is as follows:

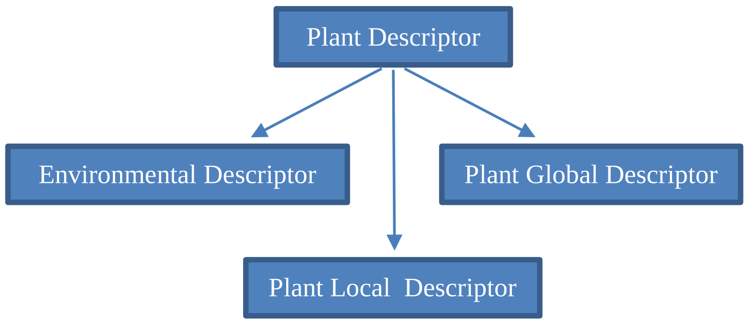


Fig 2.1: Hierarchy of the concept "PlantDescriptor"

Implementation of this hierarchy in ontology is quite simple. We can define the concept "PlantDescriptor" in triples (a subject, a predicate and an object) format with the following OWL axioms:

```
:PlantDescriptor :hasType 'Environmental_Descriptor'.
:PlantDescriptor :hasType 'Plant_Global_Descriptor'.
:PlantDescriptor :hasType 'Plant_Local_Descriptor'.
:hasType rdf:type owl:DatatypeProperty.
:hasType rdfs:domain :PlantDescriptor.
```

Here, "hasType" is the data property for the concept "PlantDescriptor" whose domain is restricted to the concept. We relate this concept to its types by defining this data property. The types for "PlantDescriptor" are stored in the data source, which are mapped later as the individual instances of the mentioned concept to the ontology using data mapping techniques.

Attribute **–** The concept "Attribute" defines different types of attributes that describes the different descriptors. There are 35 attributes that describes the 3 types of plant and environmental descriptors. The attributes that have been used in this project are:

1. Time of Occurrence (TOC)
2. Plant stand (Plant_Stand)
3. Precipitation (Precip)

4. Temperature (Temp)
5. Occurrence of hail (Hail)
6. Number of years crop repeated (Crop_hist)
7. Damaged area (Area_damaged)
8. Severity (Severity)
9. Seed treatment (Seed_tmt)
10. Seed germination (Germination)
11. Plant growth (Plant_Growth)
12. Condition of leaves (COL)
    a. Leafspots-halos
    b. Leafspots-margin
    c. Leafspot size
    d. Leaf shredding
    e. Leaf malformation
    f. Leaf mildew growth
13. Condition of stem (COS)
    a. Presence of lodging
    b. Stem cankers
    c. Canker lesion color
    d. Fruiting pod on stem
    e. External decay
    f. Mycelium on stem
    g. Internal discoloration
    h. Sclerotia
14. Condition of fruits-pods (COF)
    a. Fruit-spots
15. Condition of seed (COSeed)
    a. Mold growth
    b. Seed discoloration
    c. Seed size
    d. Seed shriveling
16. Condition of roots (COR)

We can define these attributes as the subclasses of the concept “Attribute” in triples format with the following OWL axioms:

```
:TOC rdfs:subClassOf :Attribute.
:Plant_Stand rdfs:subClassOf :Attribute.
:Precip rdfs:subClassOf :Attribute.
:Temp rdfs:subClassOf :Attribute.
:Hail rdfs:subClassOf :Attribute.
```

```
:Crop_hist rdfs:subClassOf :Attribute.
:Area_damaged rdfs:subClassOf :Attribute.
:Severity rdfs:subClassOf :Attribute.
:Seed_tmt rdfs:subClassOf :Attribute.
:Germination rdfs:subClassOf :Attribute.
:Plant_Growth rdfs:subClassOf :Attribute.
:COL rdfs:subClassOf :Attribute.
:COS rdfs:subClassOf :Attribute.
:COF rdfs:subClassOf :Attribute.
:COSeed rdfs:subClassOf :Attribute.
:COR rdfs:subClassOf :Attribute.
```

These attributes have multiple values that are stored in our data source. Moreover, some of them may have structured domain. Domains of structured descriptors can be represented as tree hierarchies in which the parent nodes represent more generalized concepts than concepts represented by their descendant nodes. For example, structured domain were used for the subclass "Area_damaged":

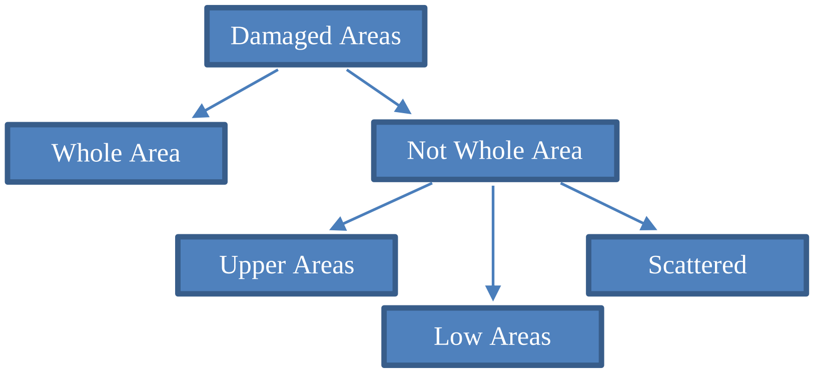


Fig 2.2: Hierarchy of the subclass "Damage Areas" of concept "Attribute"

We define this hierarchy by assigning a data property "hasAreaDamaged" to "Area_damaged". The data property is further extended by two sub-properties – "hasWholeAreaDamaged" and "hasNotWholeAreaDamaged". The OWL axioms for this complex hierarchy is actually quite straightforward:

```
:Area_damaged :hasWholeAreaDamaged 'Whole_field'.
:Area_damaged :hasNotWholeAreaDamaged 'Scattered'.
:Area_damaged :hasNotWholeAreaDamaged 'Upper_areas'.
:Area_damaged :hasNotWholeAreaDamaged 'Low_areas'.
:hasWholeAreaDamaged rdfs:subPropertyOf :hasAreaDamaged.
:hasNotWholeAreaDamaged rdfs:subPropertyOf :hasAreaDamaged.
:hasAreaDamaged rdf:type owl:DatatypeProperty.
```

```
:hasAreaDamaged rdfs:domain :Area_damaged.
```

The values for "Area_damaged" are stored in the data source, which are mapped later as the individual instances of the mentioned concept to the ontology using data mapping techniques.

Similarly, more complex hierarchies can be implemented in ontology which is easily understandable and also data access is more flexible.

Now comes the matter of relating the concept "PlantDescriptor" to the concept "Attribute". A plant descriptor is described by multiple attributes. The relationship between two concepts can be defined by assigning an object property.

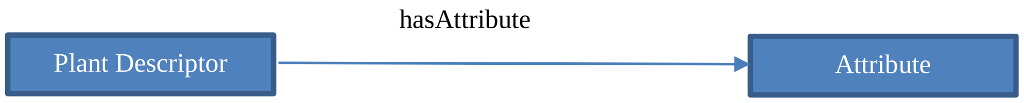


Fig 2.3: Relationship between two concepts

For example, we assign an attribute value "July" for the attribute "TOC" for the concept "PlantDescriptor". This can be easily represented using OWL axioms as follows:

```
:PlantDescriptor :hasType 'Environmental_Descriptor'.
:PlantDescriptor :hasAttribute :TOC.
:PlantDescriptor :hasTOC 'July'.
```

2.3 Soybean Data Source

Each of the concepts are described by multiple values. It is not practical to store all these values to the ontology. Rather, we store all the values for all the concepts and their sub-concepts to a database and map that values to the concepts in ontology. This is more feasible, because, we can reuse both the ontology and the data source. A concept may have multiple values, even same value may be mapped to multiple concepts. If we store both the concepts and their values either in a database or in ontology, in both cases, the idea is not very effective; if we use a separate database for the values and use it as a data source, we can reuse the data for multiple concepts very easily.

For example, the concept "Hail" has two possible values – "Yes" and "No". Another concept "Lodging" also has the same values. If we store the whole thing in database, we need to define two different tables for storing the two different concepts with same values; if we store the whole thing in ontology the same thing happens – we need to define the values separately for the two different concepts. To avoid these redundancies, we have created a table in the database for storing all the attribute values, which is served as the data source, store these values – "Yes" and "No", assign them with a particular type. Now we can map these values no matter how many times we need them in the ontology. We define once, and we use them multiple times without any redundancies.

Chapter 3

Methodology

3.1 Proposed System Architecture

The system we proposed in this project is built maintaining four different levels:

3.1.1 Ontology and Data Source

Ontology is created which stores database metadata information within the basic ontology structure. Ontology is a formal explicit description of concepts in a particular domain. Ontological model includes classes, data types, object properties, data properties, assertions, individual instances of a class and other semantics. An ontology together with a set of individual instances of classes constitutes a knowledge base. We propose a knowledge base for the purpose to analyze soybean disease. The reasons for soybean disease depends on not only environmental conditions, but also on the condition of different plant attributes. We have built an ontology which consists of the hierarchy of descriptors that describe both the environmental conditions and the soybean plant conditions that causes diseases, attributes that describes both kind of descriptors and the relation between them.

We have used Protégé as the ontology editor. It stores the ontology in both OWL and RDF/XML formats; we prefer the RDF/XML formats, as it is known to have bugs in OWL format in Protégé. Using Protégé as the ontology editor has provided lots of advantages. Protégé's plug-in architecture can be adapted to build complex ontology-based applications. We have integrated the output of Protégé to a rule engine which helped us to query the data from any platform we choose. We have discussed the soybean knowledge base earlier. It is a large ontology and has complex hierarchies. But using Protégé and ontology, we have created the knowledge base in a systematic way.

We have a database that is used as a data source for those descriptor and attribute values that cause soybean diseases. The database contains two tables - one for storing the descriptor types that defines the plant and weather conditions, and the other one for storing the attribute values describing these descriptor types.

We have used H2 database engine. H2 database is a free SQL database that is written in Java. We connect Protégé to H2 database via a JDBC connection. We use the data base as a data source to map the values to its corresponding concepts. A mapping tool is used to access the data from the data source by querying with SQL.

3.1.2 Data Mapping

The descriptors and the attributes built in the ontology are now mapped to their corresponding values stored in the data source which resides in the database. The database-to-ontology mapping is generally regarded as a case of data integration. The goal of data integration system is to provide a common interface to various data sources to enable users to focus on specifying what they want. In our proposed system, the database is the source schema and the ontology model is the target

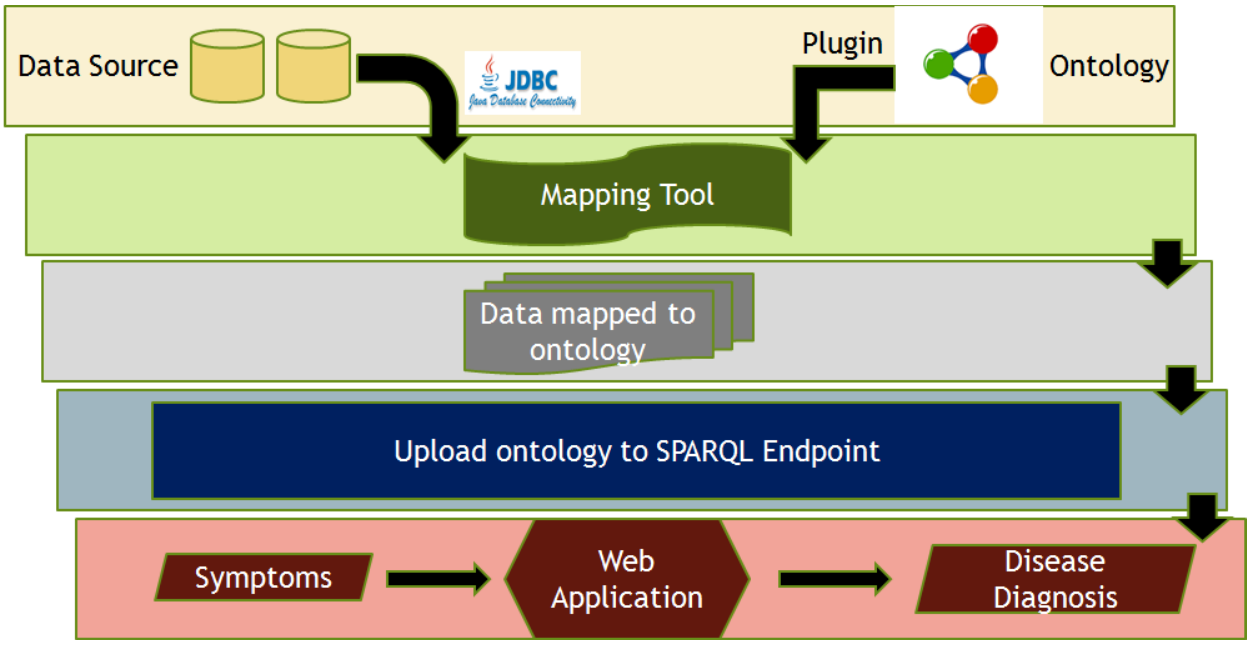


Fig 3.1: Proposed System Architecture

schema. Therefore, data integration or data mapping can be described as the problem of creating correspondences between sets of relational and ontological data. To define the problem of database and ontology mapping, taking our proposed system into account, we are given:

- An ontology expressed in RDF; the ontology contents is viewed as a set of triples.
- A relational database instance, whose contents are stored into tuples.

The objective of data mapping is to find mappings and create a set of correspondences relating ontological data – predicates in the ontology, and the relational data tuples.
For example, the concept "Temp" has the following hierarchy stored in the ontology:

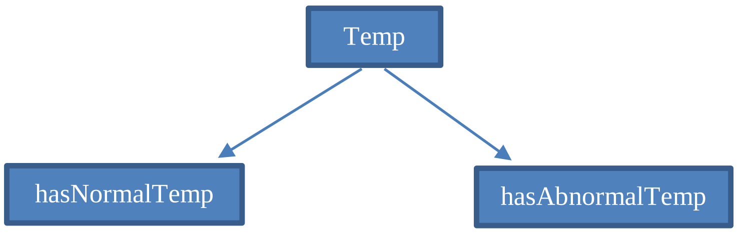


Fig 3.2: Hierarchy for concept "Temp" stored in ontology

The data source contains three possible values for the concept – Normal, Greater_than_normal, Less_than_normal. After mapping, the ontology will look like:

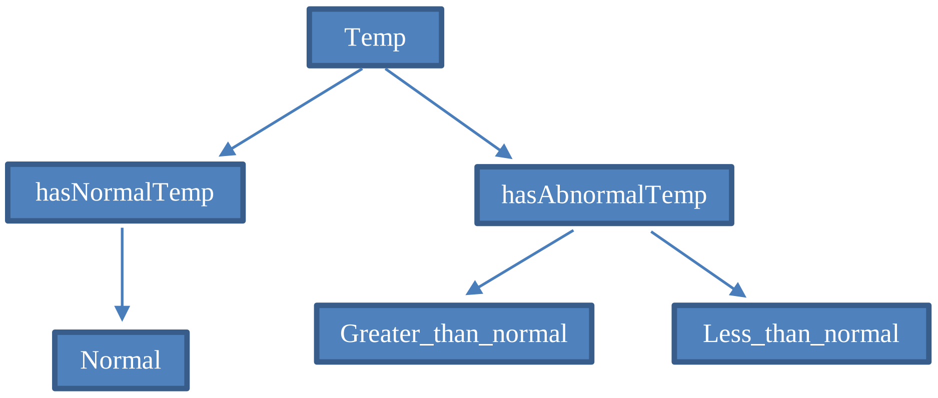


Fig 3.3: Concept "Temp" after data mapping

Therefore, the triples are now complete, and it has now the following vocabulary:

```
:Temp :hasNormalTemp 'Normal' .
:Temp :hasAbnormalTemp 'Greater_than_normal'.
:Temp :hasAbnormalTemp 'Less_than_normal'
```

3.1.3 Making the Ontology Available "Online"

The ontology is now uploaded to Fuseki server. Fuseki is a SPARQL endpoint that allows to query and alter a graph. SPARQL is an RDF (Resource Description Framework) query language that is a semantic query language for databases able to retrieve and manipulate data stored in triple format. Fuseki is the part of Apache Jena project, but is distributed as a separate package. Jena is a java based framework for building semantic applications. Fuseki is an HTTP server with all the Jena modules together. SPARQL queries and updates can be sent to Fuseki using simple HTTP requests and get responses in various formats such as JSON, XML, and CSV etc. The data can be stored either in memory or in a TDB file.

3.1.4 The Web Application

Although Apache Jena Fuseki is based on Java framework, it's just HTTP calls, so any platform can be used on the client side if a client-server architecture is required. In our project, we have used PHP as the client side, connected Fuseki server to the client side using a SPARQL library file for PHP, and built a user input interface using PHP and HTML. It is a very simple and straightforward interface. What the user needs to do is provide the inputs by selecting different options from a drop down selection menu, submit the input form and wait for the result. The system then responses with a result showing whether the combination of provided inputs result into a soybean disease or not.

Chapter 4

Implementation

This project has been implemented in a Windows environment. Protégé version 5.1.0 has been used to build the ontology. H2 database engine is used as the data source for storing all the attribute values. We have used Fuseki-server as the SPARQL endpoint. Web application has been developed using PHP, HTML. The application is hosted in XAMPP.

4.1 Create the Data Source in H2 Database Engine

We have used H2 database engine as the data source for storing the descriptor types of the soybean plant disease and the attribute values that causes them. For the plant descriptor types, we have created the table TBL_PLANTDESCRIPTOR and for the attribute values we have table TBL_ATTRIBUTE.

I. Code for creating the tables:

- **TBL_PLANTDESCRIPTOR**

```
CREATE TABLE tbl_plantdescriptor(
    PD_ID int(10) NOT NULL,
    PD_NAME varchar(40),
PRIMARY KEY PD_ID
);
```

- **TBL_ATTRIBUTE**

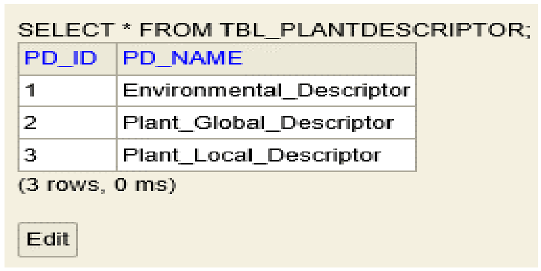
```
CREATE TABLE tbl_attribute(
    ATT_ID int(10) NOT NULL,
    ATT_NAME varchar(40),
    ATT_TYPE varchar(10),
PRIMARY KEY ATT_ID
);
```

II. Code for inserting the data into the tables:

- **TBL_PLANTDESCRIPTOR**

```
INSERT INTO TBL_PLANTDESCRIPTOR
VALUES
(1, ‘Environmental_Descriptor’),
(2, ‘Plant_Global_Descriptor’),
(3, ‘Plant_Local_Descriptor’);
```

The table in H2 database engine looks like:



```
SELECT * FROM TBL_PLANTDESCRIPTOR;
```

PD_ID	PD_NAME
1	Environmental_Descriptor
2	Plant_Global_Descriptor
3	Plant_Local_Descriptor

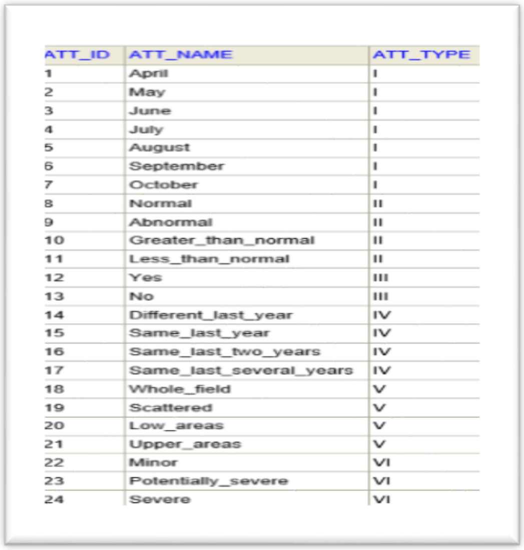
(3 rows, 0 ms)

Fig 4.1: TBL_PLANTDESCRIPTOR

- **TBL_ATTRIBUTE**

```
INSERT INTO TBL_ATTRIBUTE
VALUES
(1, ‘April’, ‘I’),
(2, ‘May’, ‘I’),
(3, ‘June’, ‘I’),
(4, ‘July’, ‘I’),
(5, ‘August’, ‘I’),
(6, ‘September’, ‘I’),
(7, ‘October’, ‘I’),
(8, ‘Normal’, ‘II’),
(9, ‘Abnormal’, ‘II’),
(10, ‘Greater_than_normal’, ‘II’),
---------------------------------
---------------------------------
---------------------------------
(56, ‘Galls_cysts’, ‘XIX’),
(57, ‘DNA’, ‘XX’);
```

The table in H2 database engine looks like:

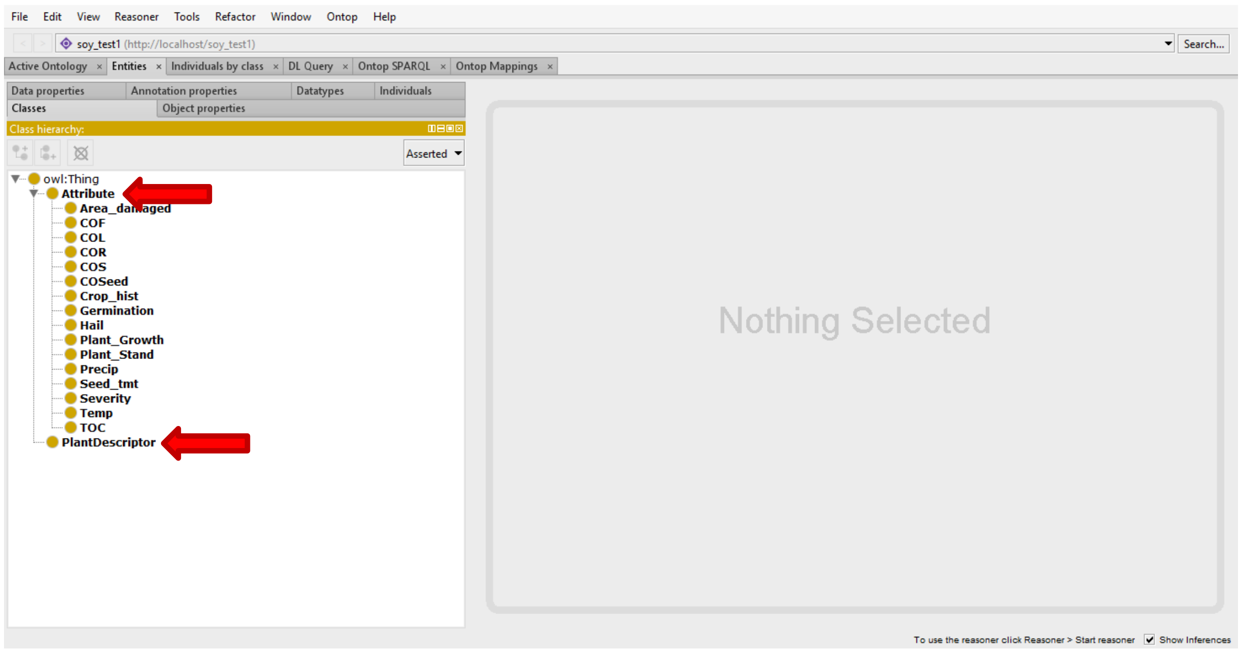


ATT_ID	ATT_NAME	ATT_TYPE
1	April	I
2	May	I
3	June	I
4	July	I
5	August	I
6	September	I
7	October	I
8	Normal	II
9	Abnormal	II
10	Greater_than_normal	II
11	Less_than_normal	II
12	Yes	III
13	No	III
14	Different_last_year	IV
15	Same_last_year	IV
16	Same_last_two_years	IV
17	Same_last_several_years	IV
18	Whole_field	V
19	Scattered	V
20	Low_areas	V
21	Upper_areas	V
22	Minor	VI
23	Potentially_severe	VI
24	Severe	VI

Fig 4.2: TBL_ATTRIBUTE

4.2 Create the ontology in Protégé

In this section, we propose a system for developing the knowledge base for soybean disease analysis. We have built an ontology to show the hierarchy of the soybean plant and environmental descriptors and the attributes that have impact on the soybean plant diseases and the relationship between them. Protégé has been used as the ontology editor.



- owl:Thing
  - Attribute
    - Area_damaged
    - COF
    - COL
    - COR
    - COS
    - COSeed
    - Crop_hist
    - Germination
    - Hail
    - Plant_Growth
    - Plant_Stand
    - Precip
    - Seed_tmt
    - Severity
    - Temp
    - TOC
  - PlantDescriptor

Fig 4.3: Concepts in Soybean Ontology

We have two new concepts in the Soybean ontology. One is the PlantDescriptor, which defines the different descriptor types which describes a soybean plant and also the environmental factors that could cause diseases and another one is the Attribute that describes these different descriptors. The relationship between the PlantDescriptor and the Attribute is defined by an object property "**hasAttribute**",

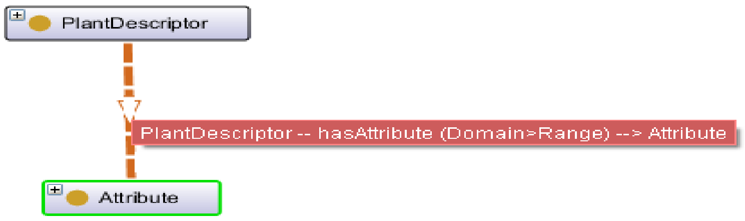

PlantDescriptor
PlantDescriptor -- hasAttribute (Domain>Range) --> Attribute
Attribute

Fig 4.4: Relationship between PlantDescriptor and Attribute Concepts

The concept Attribute is further extended into several subclasses – temp, TOC, area_damaged, leaves etc. to describe each of the PlantDescriptor.

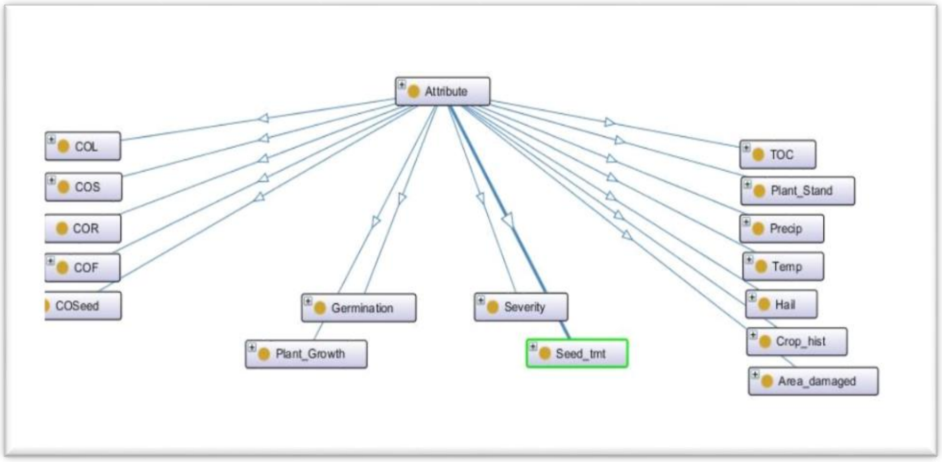


Fig 4.5: Sub-classes of Attribute concept

These attirbutes are assigned corresponding data properties to describe them. These data properties are further classified into sub-properties to describe the attributes' structured charactersitics. For example, the sub-attribute "Precip" is created to define the Precipitation of type Attribute. It is assigned the data property "hasPrecip" to describe the corresponding attribute values. It could have three possible values – Normal, Greater_than_normal, Less_than_normal. We can present this attribute and along with its corresponding values in a structured way:

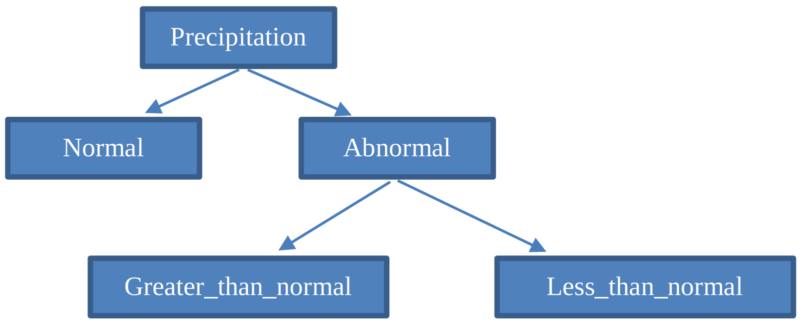


Fig 4.6: Structured domain for "Precipitation"

To implement this structured domain, we define sub-properties "hasNormalPrecip" and "hasAbnormalPrecip" to the data property "hasPrecip".

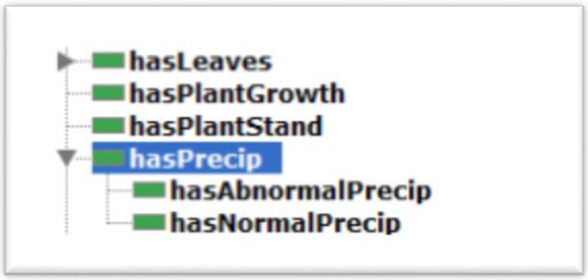


Fig 4.7: Data property "hasPrecip" and their sub-properties

These data properties are assigned to their corresponding values – Normal, Greater_than_normal, Less_than_normal, which are stored in the database, using data mapping technique.

Similarly, these sub-properties can be further classified into another set of sub-properties if needed. For example, the sub-attribute "COSeed" is created to define the "Condition of Seeds" of type Attribute. It is assigned the data property "hasSeed" to describe the corresponding attribute values. It could have two possible values – Normal and Abnormal. Abnormality of seeds can be further classified based on some other seed characterstics. We can present these values in a structured way:

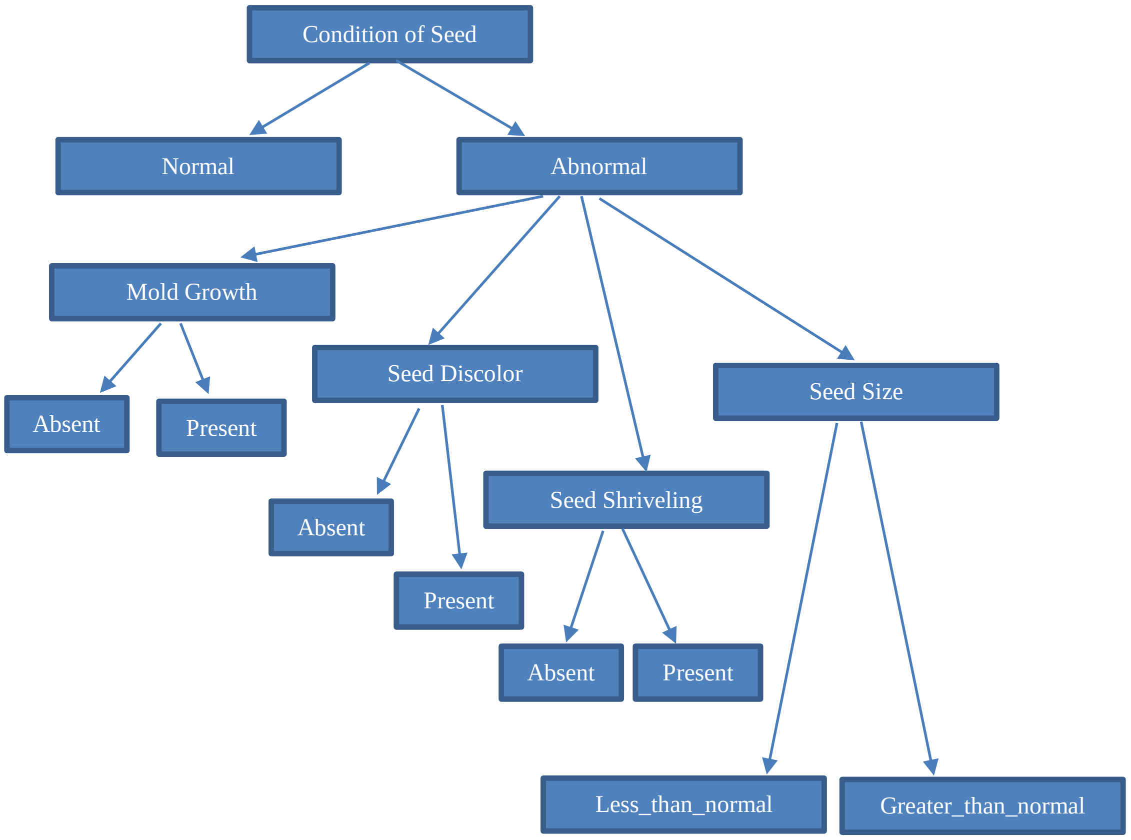


Fig 4.8: Structured domain for "Condition of Seed"

To implement this structured domain, we define sub-properties "hasNormalSeed" and "hasAbnormalSeed" to the data property "hasSeed". The sub-property "hasAbnormalSeed" seed is extended further to classify four other characteristics of seed – "hasMoldGrowth", "hasSeedDiscolor", "hasSeedShriveling", "hasSeedSize".

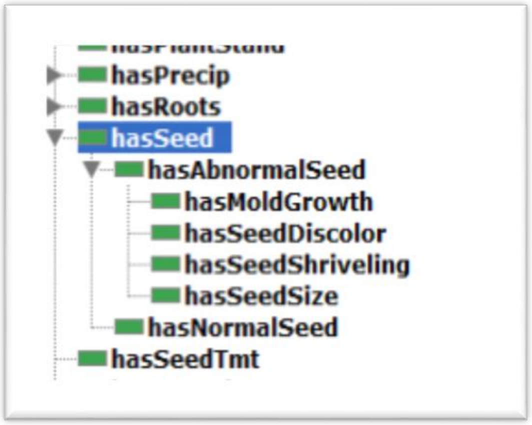


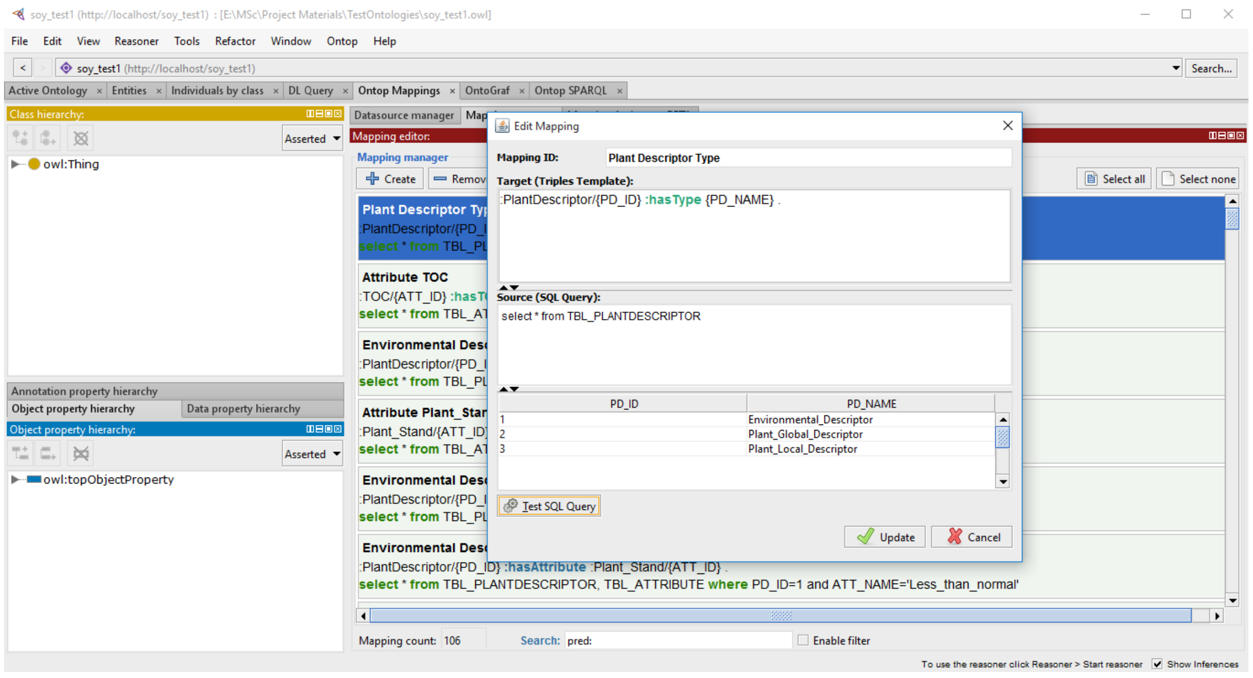
Fig 4.9: Data property "hasSeed" and its sub-properties

In this way, we can built more complex hierarchies in a very simple way. If we had to build this hierarchy in a relational database it would have been more complicated, it would require more tables and one-to-many and many-to-one relations. Using Protégé as the ontology editor have helped us to avoid such complexities.

4.3 Data mapping using *Ontop* Tab in Protégé

Now we map the data from the data source in H2 database engine to our ontology in Protégé. All the data are mapped to their corresponding classes and subclasses as "individual instances" according to the hierarchy we have defined. The data mapped as individual instances are the terminating point of the hierarchy. Mapping is done in two ways – data property mapping and object property mapping.

- **Data Property Mapping:** Mapping procedure is very easy and simple. For example, we map the concept "PlantDescriptor" with its corresponding values stored in the database through data property mapping. What we have to do is, just define a name for the individuals to which the values from the database are to map as instances of the mentioned concept. We assign this individual to the corresponding data property as RDF triples vocabulary. Then we make an SQL query to get the data into the corresponding individuals of the ontology. The whole process is done using the *Ontop* plugin for Protégé.


soy_test1 (http://localhost/soy_test1) : [E:\MSc\Project Materials\TestOntologies\soy_test1.owl]
File Edit View Reasoner Tools Refactor Window Ontop Help
soy_test1 (http://localhost/soy_test1)
Search...
Active Ontology | Entities | Individuals by class | DL Query | Ontop Mappings | OntoGraf | Ontop SPARQL
Class hierarchy:
Asserted
owl:Thing
Annotation property hierarchy
Object property hierarchy
Data property hierarchy
Object property hierarchy:
Asserted
owl:topObjectProperty
Datasource manager
Mapping editor:
Mapping manager
Create
Select all
Select none
Edit Mapping
Test SQL Query
Update
Cancel
Enable filter
To use the reasoner click Reasoner > Start reasoner
Show Inferences

Mapping ID: Plant Descriptor Type

Target (Triples Template):

```
:PlantDescriptor/{PD_ID} :hasType {PD_NAME} .
```

Source (SQL Query):

```
select * from TBL_PLANTDESCRIPTOR
```

PD_ID	PD_NAME
1	Environmental_Descriptor
2	Plant_Global_Descriptor
3	Plant_Local_Descriptor

Environmental Desc

```
:PlantDescriptor/{PD_ID} :hasAttribute :Plant_Stand/{ATT_ID} .
select * from TBL_PLANTDESCRIPTOR, TBL_ATTRIBUTE where PD_ID=1 and ATT_NAME='Less_than_normal'
```

Mapping count: 106 Search: pred:

Fig 4.10: Mapping "PlantDescriptor" Individuals using *Ontop*

Now the data is mapped to the concept “PlantDescriptor”. It has now three individuals – PlantDescriptor/1, PlantDescriptor/2, PlantDescriptor/3, and they have three corresponding instances – Environmental_Descriptor, Plant_Global_Descriptor, Plant_Local_Descriptor.

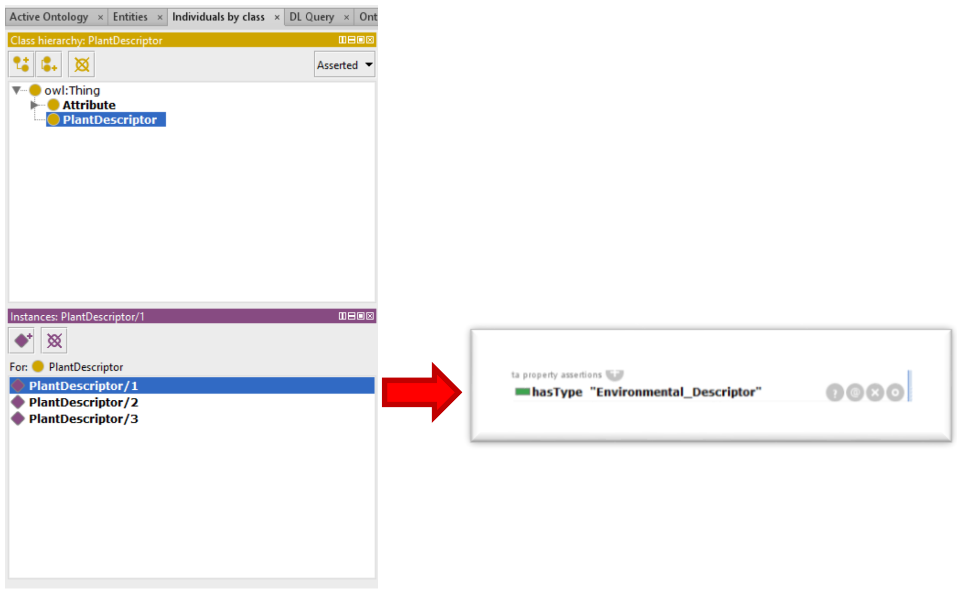
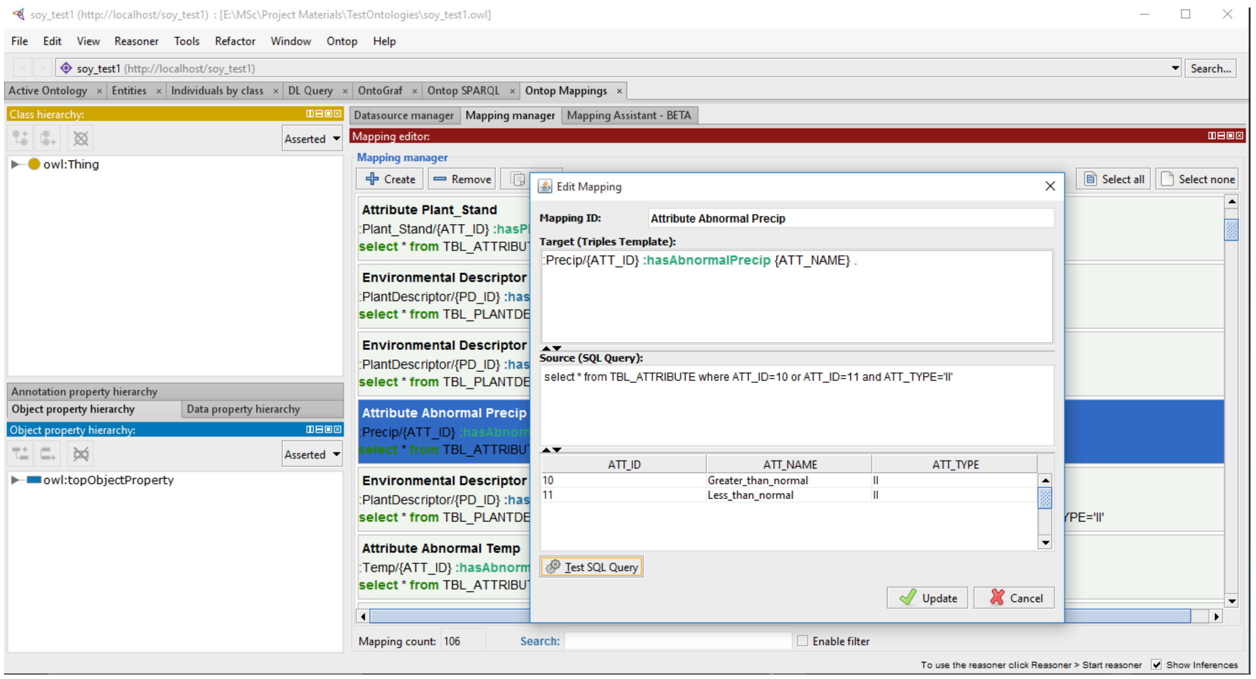


Fig 4.11: Individuals and instances for “PlantDescriptor”

- **Object Property Mapping:** Now we assign an object property to our concept “PlantDescriptor”. An object property is used to connect two different concepts. In our example ontology, we establish the relationship between the concepts “PlantDescriptor” and “Attribute” via the object property “hasAttribute”. The concept “Attribute” has several subclasses, so what really happens here is that, an individual instance of “PlantDescriptor” is connected to one or more individual instances of the subclasses of the concept “Attributes”.

  For example, we map the individual instance “Environmental_Descriptor” of the concept “PlantDescriptor” with the individual instance “Greater_than_normal” of the subclass “Precip” of the concept “Attribute” with the object property “hasAttribute”. First we need to map data property for “Precip” with its individual instance “Greater_than_normal” and “Less_than_normal” via data property “hasAbnormalPrecip”. Then we do the object property mapping in the same way we did the data property mapping, only difference is, we provide the object property instead of data property.


soy_test1 (http://localhost/soy_test1) : [E:\MSc\Project Materials\TestOntologies\soy_test1.owl]
File Edit View Reasoner Tools Refactor Window Ontop Help
Active Ontology | Entities | Individuals by class | DL Query | OntoGraf | Ontop SPARQL | Ontop Mappings
Class hierarchy:
owl:Thing
Object property hierarchy:
owl:topObjectProperty
Datasource manager | Mapping manager | Mapping Assistant - BETA
Mapping editor:
Create | Remove | Select all | Select none
Edit Mapping
Test SQL Query
Update | Cancel
Mapping count: 106
Search:
Enable filter
To use the reasoner click Reasoner > Start reasoner
Show Inferences

Mapping ID: Attribute Abnormal Precip

Target (Triples Template):

```
:Precip/{ATT_ID} :hasAbnormalPrecip {ATT_NAME} .
```

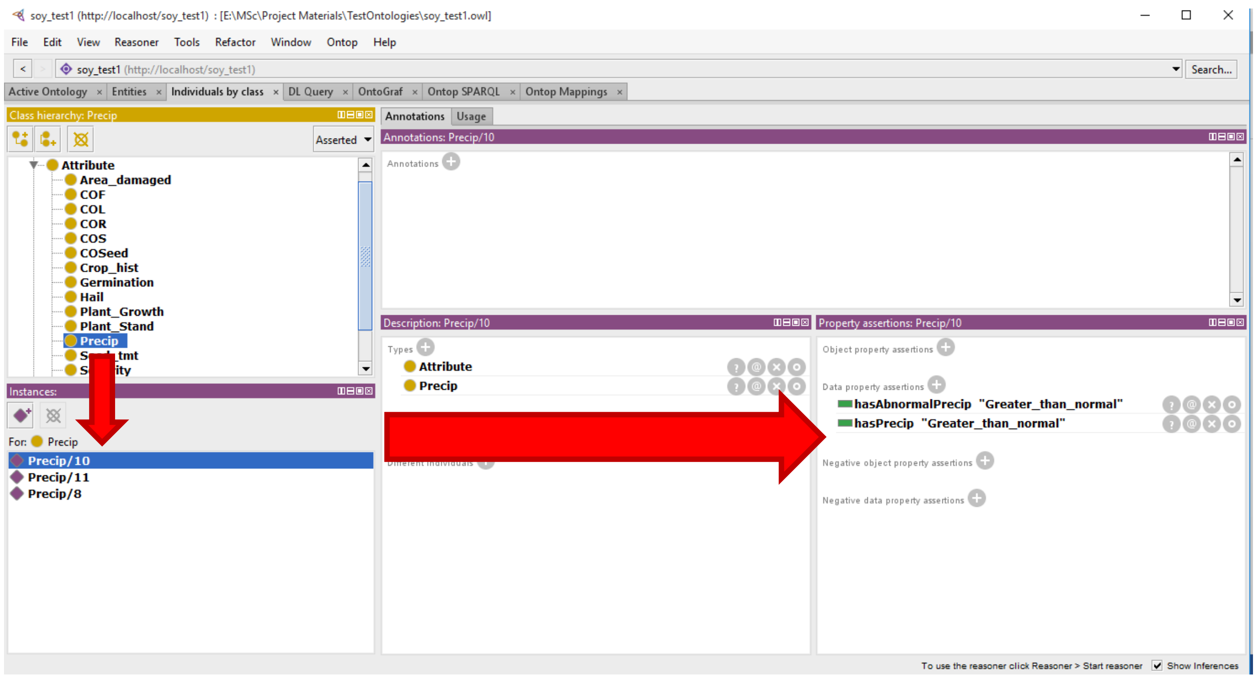
Source (SQL Query):

```
select * from TBL_ATTRIBUTE where ATT_ID=10 or ATT_ID=11 and ATT_TYPE='II'
```

ATT_ID	ATT_NAME	ATT_TYPE
10	Greater_than_normal	II
11	Less_than_normal	II

Fig 4.12: Mapping "PlantDescriptor" Individuals using *Ontop*

Now the data is mapped to the subclass "Precip" of the concept "Attribute". It has now two individuals for the data property "hasAbnormalPrecip" – Precip/10 and Precip/11, and they have two corresponding instances – Greater_than_normal and Less_than_normal.


soy_test1 (http://localhost/soy_test1) : [E:\MSc\Project Materials\TestOntologies\soy_test1.owl]
File Edit View Reasoner Tools Refactor Window Ontop Help
Active Ontology | Entities | Individuals by class | DL Query | OntoGraf | Ontop SPARQL | Ontop Mappings
Class hierarchy: Precip
Asserted
Attribute
Area_damaged
COF
COL
COR
COS
COSeed
Crop_hist
Germination
Hail
Plant_Growth
Plant_Stand
Precip
Instances:
For: Precip
Precip/10
Precip/11
Precip/8
Annotations | Usage
Annotations: Precip/10
Description: Precip/10
Types
Attribute
Precip
Property assertions: Precip/10
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions
To use the reasoner click Reasoner > Start reasoner
Show Inferences

hasAbnormalPrecip "Greater_than_normal"
hasPrecip "Greater_than_normal"

Fig 4.13: Individuals and instances for "Precip"

We have completed all other mappings following the same procedure. In this way, we can use ontology to store large and complex databases easily.

4.4 Upload the Ontology to Fuseki Server

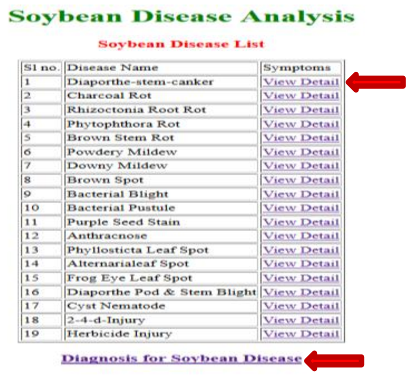
To access the data stored in ontology, we need to upload it somewhere in the localhost. We upload our ontology to the SPARQL endpoint, Fuseki server. The upload procedure is quite simple. We saved our ontology to a local drive, envoke the Fuseki server from the command prompt using the follow command:

```
java -Xmx1200M -jar fuseki-server.jar --update --loc=Data
/myDataset
```

Then the Fuseki server is started. We upload the ontology and now our ontology is ready for access from any platform we want.

4.5 Developing a Web-based Application

Our main objective is to provide a user platform for soybean disease diagnosis where the user will have the privilege to provide symptoms for the soybean plant and get a response from the system developed whether the symptoms result in a disease or not. To achieve this, we have developed a web application using PHP, where a user can find the expert defined symptoms for a particular soybean disease, as well as provide the real symptoms and check for the presence of a disease.
There is no direct method to query the uploaded ontology from the PHP client side program. We have used a SPARQL library file for PHP, developed in the University of Southampton. This library file is used to connect the Fuseki server to PHP, which enables querying the ontology using SPARQL.
The home page of our application contains the list of 15 soybean diseases we have diagnosed.



Soybean Disease Analysis

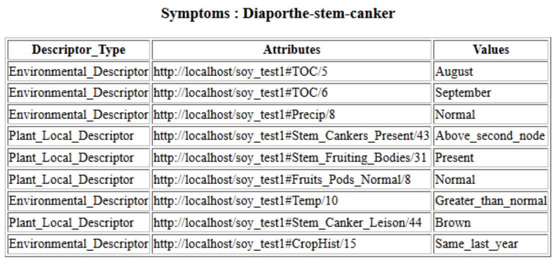
Soybean Disease List

Sl no.	Disease Name	Symptoms
1	Diaporthe-stem-canker	View Detail
2	Charcoal Rot	View Detail
3	Rhizoctonia Root Rot	View Detail
4	Phytophthora Rot	View Detail
5	Brown Stem Rot	View Detail
6	Powdery Mildew	View Detail
7	Downy Mildew	View Detail
8	Brown Spot	View Detail
9	Bacterial Blight	View Detail
10	Bacterial Pustule	View Detail
11	Purple Seed Stain	View Detail
12	Anthracnose	View Detail
13	Phyllosticta Leaf Spot	View Detail
14	Alternarialeaf Spot	View Detail
15	Frog Eye Leaf Spot	View Detail
16	Diaporthe Pod & Stem Blight	View Detail
17	Cyst Nematode	View Detail
18	2-4-d-Injury	View Detail
19	Herbicide Injury	View Detail

Diagnosis for Soybean Disease

Fig 4.14: Home page of the web application

If the user wants to see the expert defined symptoms for any particular disease, one has to click on "View Details", the link will take the user to the page for that particular disease.

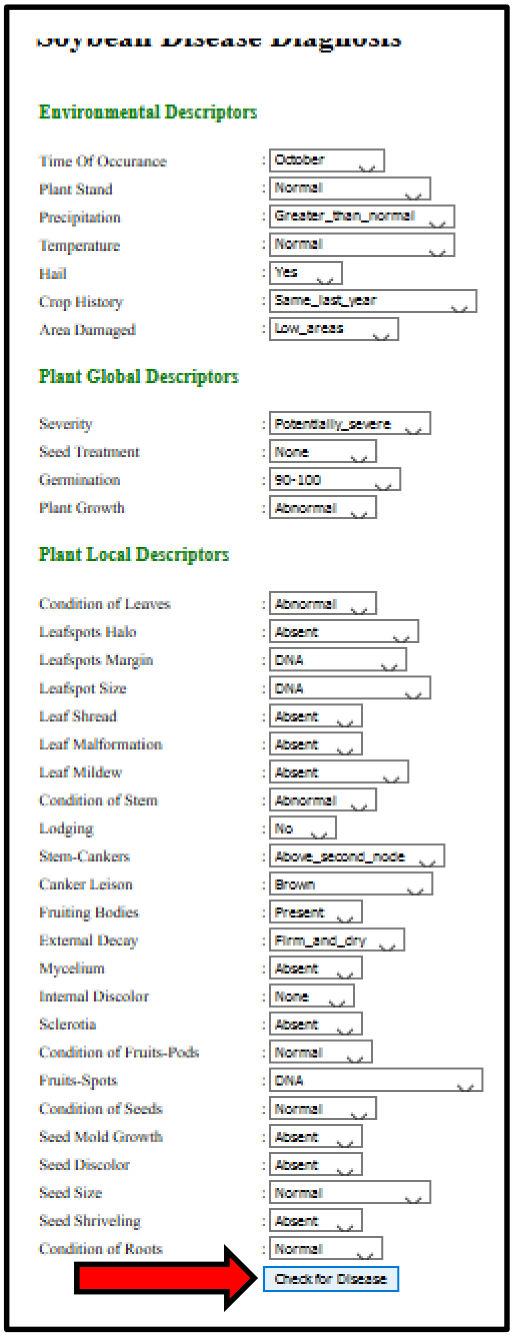


Symptoms : Diaporthe-stem-canker

Descriptor_Type	Attributes	Values
Environmental_Descriptor	http://localhost/soy_test1#TOC/5	August
Environmental_Descriptor	http://localhost/soy_test1#TOC/6	September
Environmental_Descriptor	http://localhost/soy_test1#Precip/8	Normal
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Cankers_Present/43	Above_second_node
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Fruiting_Bodies/31	Present
Plant_Local_Descriptor	http://localhost/soy_test1#Fruits_Pods_Normal/8	Normal
Environmental_Descriptor	http://localhost/soy_test1#Temp/10	Greater_than_normal
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Canker_Leison/44	Brown
Environmental_Descriptor	http://localhost/soy_test1#CropHist/15	Same_last_year

Fig 4.15: Symptoms for disease "Diaporthe-Stem-Canker"

The user can also provide symptoms as inputs and check for the presence of a disease. There is a user input form, where the user just needs to select the symptoms from a drop-down selection menu, and the system will perform diagnosis for disease. Clicking on the link "Diagnosis for soybean disease" will take the user to the input form.
After providing all the required inputs, the system matches the symptoms and replies with the result whether a disease has occurred or not.



Soybean Disease Diagnosis

Environmental Descriptors

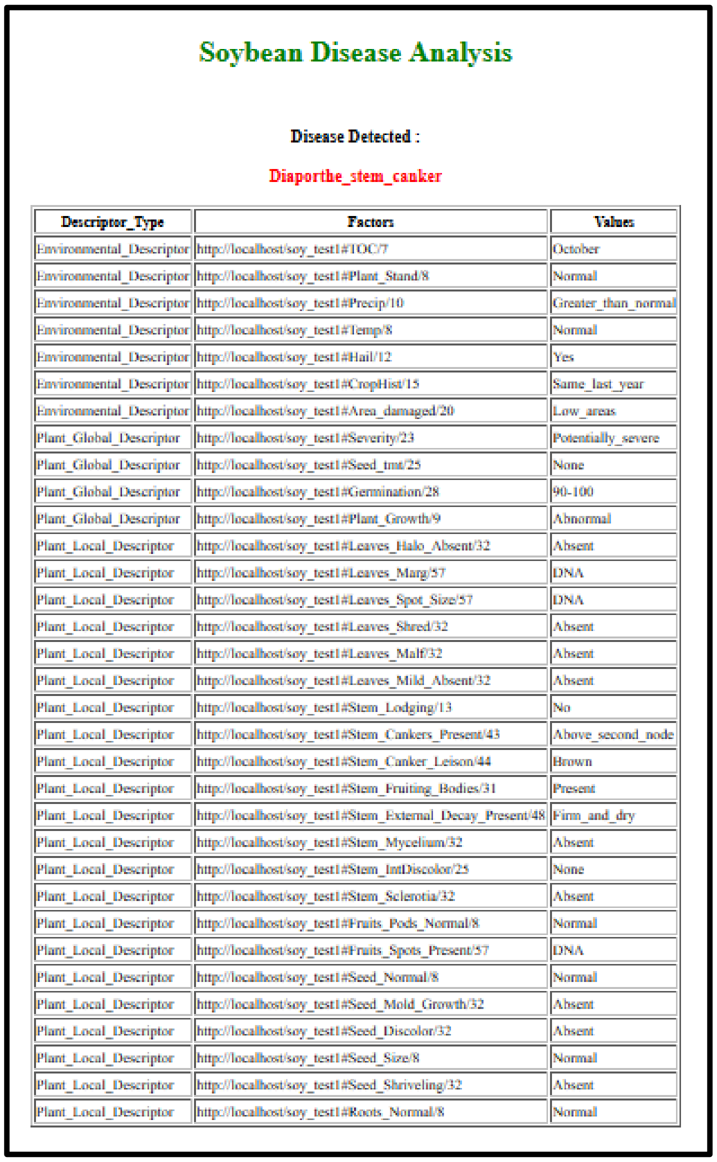
Time Of Occurance	October
Plant Stand	Normal
Precipitation	Greater_than_normal
Temperature	Normal
Hail	Yes
Crop History	Same_last_year
Area Damaged	Low_areas

Plant Global Descriptors

Severity	Potentially_severe
Seed Treatment	None
Germination	90-100
Plant Growth	Abnormal

Plant Local Descriptors

Condition of Leaves	Abnormal
Leafspots Halo	Absent
Leafspots Margin	DNA
Leafspot Size	DNA
Leaf Shread	Absent
Leaf Malformation	Absent
Leaf Mildew	Absent
Condition of Stem	Abnormal
Lodging	No
Stem-Cankers	Above_second_node
Canker Leison	Brown
Fruiting Bodies	Present
External Decay	Firm_and_dry
Mycelium	Absent
Internal Discolor	None
Sclerotia	Absent
Condition of Fruits-Pods	Normal
Fruits-Spots	DNA
Condition of Seeds	Normal
Seed Mold Growth	Absent
Seed Discolor	Absent
Seed Size	Normal
Seed Shriveling	Absent
Condition of Roots	Normal



Soybean Disease Analysis

Disease Detected :

Diaporthe_stem_canker

Descriptor_Type	Factors	Values
Environmental_Descriptor	http://localhost/soy_test1#TOC/7	October
Environmental_Descriptor	http://localhost/soy_test1#Plant_Stand/8	Normal
Environmental_Descriptor	http://localhost/soy_test1#Precip/10	Greater_than_normal
Environmental_Descriptor	http://localhost/soy_test1#Temp/8	Normal
Environmental_Descriptor	http://localhost/soy_test1#Hail/12	Yes
Environmental_Descriptor	http://localhost/soy_test1#CropHist/15	Same_last_year
Environmental_Descriptor	http://localhost/soy_test1#Area_damaged/20	Low_areas
Plant_Global_Descriptor	http://localhost/soy_test1#Severity/23	Potentially_severe
Plant_Global_Descriptor	http://localhost/soy_test1#Seed_tmt/25	None
Plant_Global_Descriptor	http://localhost/soy_test1#Germination/28	90-100
Plant_Global_Descriptor	http://localhost/soy_test1#Plant_Growth/9	Abnormal
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Halo_Absent/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Marg/57	DNA
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Spot_Size/57	DNA
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Shred/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Malf/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Leaves_Mild_Absent/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Lodging/13	No
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Cankers_Present/43	Above_second_node
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Canker_Leison/44	Brown
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Fruiting_Bodies/31	Present
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_External_Decay_Present/48	Firm_and_dry
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Mycelium/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_IntDiscolor/25	None
Plant_Local_Descriptor	http://localhost/soy_test1#Stem_Sclerotia/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Fruits_Pods_Normal/8	Normal
Plant_Local_Descriptor	http://localhost/soy_test1#Fruits_Spots_Present/57	DNA
Plant_Local_Descriptor	http://localhost/soy_test1#Seed_Normal/8	Normal
Plant_Local_Descriptor	http://localhost/soy_test1#Seed_Mold_Growth/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Seed_Discolor/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Seed_Size/8	Normal
Plant_Local_Descriptor	http://localhost/soy_test1#Seed_Shriveling/32	Absent
Plant_Local_Descriptor	http://localhost/soy_test1#Roots_Normal/8	Normal

Fig 4.16: User input form (left) and disease diagnosed in response to the provided input (right)

Chapter 5

Conclusion

5.1 Summary

In this project, we have presented our experience working with ontology based tools to build a very simple web application that exposes the capability of Protégé to create an ontology in RDF format and use it as the vocabulary for a PHP application. We have used an open source OBDA system, *Ontop* that allows querying relational data sources through a domain provided in terms of ontology and mapping the data sources to the ontology. The key advantage of this mapping procedure is that no interpretation of data needs to be carried out to be stored as ontology instances. This has reduced a lot of work, because interpretation of data in existing data sources may cause some scalability issues with existing legacy applications. This approach is also beneficial where it is not practical to store all data as a part of the respective domain ontology, especially for systems with a huge volume of data. Even in relational database, storing data with structured domain is also very complex. Mapping technique has helped us to get rid of all these tedious work.
The use of Fuseki-server as the SPARQL endpoint has provided us with the advantage to use it as a server for any client-side programming for a client-server based application. The Fuseki-server has served as the triple store for the ontology vocabulary. A PHP library file has been used to connect Fuseki-server to the PHP application that has been developed as a prototype system for the users for the diagnosis of soybean disease.

5.2 Limitations

Due to the limitation of time, many important features that could have made the application more demanding, are absent. There are scopes for immense improvement to enhance this project.

- Any application is all about attracting users with flexible, user-friendly interface with attractive and useful features and utilities. Our application is quite flat and simple; it lacks attractiveness. It is user-friendly and quite flexible to use, but it still needs some improvement in the GUI.
- We have not exercised the advantage of SPARQL at its best; further work must be done to handle more complex situations that may involve multiple ontology assertions.
- Protégé has some issues with performance; the larger the ontology gets, the more time it takes to map the data from data source. This becomes a major issue for applications with huge ontology.
- The manual establishment of mappings between database schemas and ontologies is considered as an additional burden in the development process. It is time consuming and error-prone task.
- At present, the application contains some defined set of rules for soybean disease diagnosis, based on which the disease analysis is done. There is no provision for a domain expert to alter or generate new rules for the respective domain.

The application is developed as a prototype system for soybean disease analysis. This can be made more generic. The same ontology can be reused, even multiple ontologies can be used to diagnose multiple diseases in the same platform.

5.3 Future Work

This project has been developed with the vision of future possibilities that could bring change in the sector of agriculture. Further studies and research work could add diversity to this system which could bring a change to our society for good.

- For end-users, the GUI interfaces can be improved to add more utilities and make the application more user-friendly.
- At present, our project work support all major databases, but later on we could further study to make it possible to support other kind of data sources like graph databases and document oriented databases, etc.
- Domain experts must have the privilege to alter the existing rules and to insert new rules for the respective domain. Regular updating information would make the best use of this application.
- Fuzzy logic can also be applied for real predictions. For example, based on some symptoms provided by the user, the system would reply with the possibility of the occurrence of a disease.

Appendix A: Source Code

1. **SPARQL Query for viewing disease symptoms**

```
<html>
      <head>
            <title>Soybean Disease Analysis</title>
      </head>



<body>
      <?php
      require_once( "sparqllib.php" );

      $db = sparql_connect( "http://localhost:3030/myDataset/sparql" );
      if( !$db ) { print sparql_errno() . ": " . sparql_error(). "\n";
exit; }
      sparql_ns( "foaf","http://xmlns.com/foaf/0.1/" );
      sparql_ns("rdf", "http://www.w3.org/1999/02/22-rdf-syntax-ns#");
      sparql_ns("rdfs", "http://www.w3.org/2000/01/rdf-schema#");

      $sparql = "PREFIX : <http://localhost/soy_test1#>
                        select   distinct   ?Descriptor_Type   ?Attributes
?Values
                        where{
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                    ?PlantDescriptor       :hasAttribute
?Attributes .
                                    ?Attributes :hasTOC 'August' .
                                    ?Attributes :hasTOC ?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                    ?PlantDescriptor       :hasAttribute
?Attributes .
                                    ?Attributes :hasTOC 'September' .
                                    ?Attributes :hasTOC ?Values .}

                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                    ?PlantDescriptor       :hasAttribute
?Attributes .
                                    ?Attributes         :hasNormalPrecip
'Normal' .
                                    ?Attributes :hasNormalPrecip ?Values
.}


                        union
```

```
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
                                     ?Attributes     :StemCankersPresent
'Above_second_node' .
                                     ?Attributes     :StemCankersPresent
?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
                                     ?Attributes  :hasStemFruitingBodies
'Present' .
                                     ?Attributes  :hasStemFruitingBodies
?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
                                     ?Attributes   :hasNormalFruits_Pods
'Normal' .
                                     ?Attributes   :hasNormalFruits_Pods
?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
                                     ?Attributes        :hasAbnormalTemp
'Greater_than_normal' .
                                     ?Attributes        :hasAbnormalTemp
?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
                                     ?Attributes    :hasStemCankerLeison
'Brown' .
                                     ?Attributes    :hasStemCankerLeison
?Values .}
                        union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type .
                                     ?PlantDescriptor      :hasAttribute
?Attributes .
```

```
                                    ?Attributes        :hasSameCropHist
'Same_last_year' .
                                    ?Attributes        :hasSameCropHist
?Values .}


                             }";

      $result = sparql_query( $sparql );
      if( !$result ) { print sparql_errno() . ": " . sparql_error(). "\n";
exit; }

      $fields = sparql_field_array( $result );

      print "<h3 style=\"text-align:center\">Symptoms : Diaporthe-stem-
canker</h3>";
      //print "<p>Number of rows: ".sparql_num_rows( $result )."
results.</p>";
      print "<table border=\"1\"; align=\"center\">";
      print "<tr>";
      foreach( $fields as $field )
      {
            print "<th>$field</th>";
      }
      print "</tr>";
      while( $row = sparql_fetch_array( $result ) )
      {
            print "<tr>";
            foreach( $fields as $field )
            {
                  print "<td>$row[$field]</td>";
            }
            print "</tr>";
      }
      print "</table>";
      ?>

</body>
</html>
```

2. SPARQL Query for disease diagnosis

```
<?php

session_start();
require_once( "sparqllib.php" );

$db = sparql_connect( "http://localhost:3030/myDataset/sparql" );
if( !$db ) { print sparql_errno() . ": " . sparql_error(). "\n"; exit; }
sparql_ns( "foaf","http://xmlns.com/foaf/0.1/" );
sparql_ns("rdf", "http://www.w3.org/1999/02/22-rdf-syntax-ns#");
sparql_ns("rdfs", "http://www.w3.org/2000/01/rdf-schema#");


$sparql1 = "PREFIX : <http://localhost/soy_test1#>
                  select distinct ?Descriptor_Type ?Factors ?Values where
{
                                                 {?PlantDescriptor
:hasType  ?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .
?Factors :hasTOC '" .$_POST['TOC']. "' . ?Factors :hasTOC ?Values .}
                                           union
                                                 {?PlantDescriptor
:hasType  ?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .
?Factors  :hasPlantStand  '"  .$_POST['PS'].  "'  .  ?Factors  :hasPlantStand
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasAbnormalPrecip '" .$_POST['Precip']. "' . ?Factors :hasAbnormalPrecip
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasNormalTemp  '"  .$_POST['Temp'].  "'  .  ?Factors  :hasNormalTemp  ?Values
.}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasAbnormalTemp   '"   .$_POST['Temp'].   "'   .   ?Factors   :hasAbnormalTemp
?Values .}

                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasHail '" .$_POST['Hail']. "' . ?Factors :hasHail ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasSameCropHist '" .$_POST['Crop_hist']. "' . ?Factors :hasSameCropHist
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasNotWholeAreaDamaged   '"   .$_POST['Area_damaged'].   "'   .   ?Factors
:hasNotWholeAreaDamaged ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
```

```
:hasAreaDamaged '" .$_POST['Area_damaged']. "' . ?Factors :hasAreaDamaged
?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasSeverity '" .$_POST['Severity']. "' . ?Factors :hasSeverity ?Values
.}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasSeedTmt '" .$_POST['Seed_tmt']. "' . ?Factors :hasSeedTmt ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasGermination '" .$_POST['Germination']. "' . ?Factors :hasGermination
?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasPlantGrowth '" .$_POST['Plant_growth']. "' . ?Factors :hasPlantGrowth
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:LeafSpotAbsent    '"    .$_POST['Leafspots-halo'].    "'    .    ?Factors
:LeafSpotAbsent ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasLeafSpotsMarg   '"   .$_POST['Leafspots-margin'].   "'   .   ?Factors
:hasLeafSpotsMarg ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasLeafSpotsSize    '"    .$_POST['Leafspot-size'].    "'    .    ?Factors
:hasLeafSpotsSize ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasLeafShred  '"  .$_POST['Leaf-Shread'].  "'  .  ?Factors  :hasLeafShred
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasLeafMalf '" .$_POST['Leaf-malf']. "' . ?Factors :hasLeafMalf ?Values
.}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
```

```
:LeafMildAbsent '" .$_POST['Leaf_mild']. "' . ?Factors :LeafMildAbsent
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemLodging '" .$_POST['lodging']. "' . ?Factors :hasStemLodging
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:StemCankersPresent   '"   .$_POST['stem_cankers'].   "'   .   ?Factors
:StemCankersPresent ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemCankerLeison   '"   .$_POST['canker-leison'].   "'   .   ?Factors
:hasStemCankerLeison ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemFruitingBodies  '"  .$_POST['fruiting_bodies'].  "'  .  ?Factors
:hasStemFruitingBodies ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:ExternalDecayPresent  '"  .$_POST['external_decay'].  "'  .  ?Factors
:ExternalDecayPresent ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemMycelium '" .$_POST['mycelium']. "' . ?Factors :hasStemMycelium
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemIntDiscolor   '"   .$_POST['intdiscolor'].   "'   .   ?Factors
:hasStemIntDiscolor ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemSclerotia    '"    .$_POST['sclerotia'].    "'    .    ?Factors
:hasStemSclerotia ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalFruits_Pods   '"   .$_POST['fruit_pods'].   "'   .   ?Factors
:hasNormalFruits_Pods ?Values .}
                              union
```

```
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasDiseasedFruits_Pods  '" .$_POST['fruit_spots']. "' . ?Factors
:hasDiseasedFruits_Pods ?Values .}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalSeed '" .$_POST['COSeed']. "' . ?Factors :hasNormalSeed ?Values
.}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasMoldGrowth '" .$_POST['mold_growth']. "' . ?Factors :hasMoldGrowth
?Values .}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedDiscolor  '"  .$_POST['seed_discolor'].  "'  .  ?Factors
:hasSeedDiscolor ?Values .}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedSize '" .$_POST['seed_size']. "' . ?Factors :hasSeedSize ?Values
.}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedShriveling  '" .$_POST['seed_shriveling'].  "'  .  ?Factors
:hasSeedShriveling ?Values .}
                             union
                                {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalRoots '" .$_POST['COR']. "' . ?Factors :hasNormalRoots ?Values
.}



                             filter  (regex('"  .$_POST['TOC'].  "',
'July')  ||  regex('"  .$_POST['TOC'].  "',  'August')  ||  regex('"
.$_POST['TOC'].  "',  'September')  ||  regex('"  .$_POST['TOC'].  "',
'October')) .
                             filter  (regex('"  .$_POST['PS'].  "',
'Normal') || regex('" .$_POST['PS']. "', 'DNA')) .
                             filter  regex('"  .$_POST['Precip']."',
'Greater_than_normal') .
                             filter  (regex('"  .$_POST['Temp'].  "',
'Normal') || regex('" .$_POST['Temp']."', 'Greater_than_normal')) .
```

```
                              filter  (regex('"  .$_POST['Hail'].  "',
'Yes')  ||  regex('"  .$_POST['Hail'].  "',  'No')  ||  regex('"
.$_POST['Hail']. "', 'DNA')) .
                              filter (regex('" .$_POST['Crop_hist']. "',
'Same_last_year')    ||    regex('"    .$_POST['Crop_hist'].    "',
'Same_last_two_years')) .
                              filter (regex('" .$_POST['Area_damaged'].
"', 'Scattered') || regex('" .$_POST['Area_damaged']. "', 'Low_areas') ||
regex('" .$_POST['Area_damaged']. "', 'DNA')) .
                              filter (regex('" .$_POST['Severity']. "',
'Severe') || regex('" .$_POST['Severity']. "', 'Potentially_severe') ||
regex('" .$_POST['Severity']. "', 'DNA')) .
                              filter (regex('" .$_POST['Seed_tmt']. "',
'Fungicide') || regex('" .$_POST['Seed_tmt']. "', 'None') || regex('"
.$_POST['Seed_tmt']. "', 'DNA')) .
                              filter  (regex('"  .$_POST['Germination'].
"',  '90-100')  ||  regex('"  .$_POST['Germination'].  "',  '80-89')  ||
regex('"  .$_POST['Germination'].  "',  'Less_than_80')  ||  regex('"
.$_POST['Germination']. "', 'DNA')) .
                              filter (regex('" .$_POST['Plant_growth'].
"', 'Abnormal') || regex('" .$_POST['Plant_growth']. "', 'DNA')) .
                              filter    regex('"    .$_POST['COL']."',
'Abnormal') .
                              filter    regex('"    .$_POST['Leafspots-
halo']."', 'Absent') .
                              filter    regex('"    .$_POST['Leafspots-
margin']."', 'DNA') .
                              filter    regex('"    .$_POST['Leafspot-
size']."', 'DNA') .
                              filter regex('" .$_POST['Leaf-Shread']."',
'Absent') .
                              filter  regex('"  .$_POST['Leaf-malf']."',
'Absent') .
                              filter  regex('"  .$_POST['Leaf_mild']."',
'Absent') .
                              filter    regex('"    .$_POST['COS']."',
'Abnormal') .
                              filter  (regex('"  .$_POST['lodging']."',
'Yes') || regex('" .$_POST['lodging']."', 'No')) .
                              filter                          regex('"
.$_POST['stem_cankers']."', 'Above_second_node') .
                              filter    (regex('"    .$_POST['canker-
leison']."', 'Brown') || regex('" .$_POST['canker-leison']."', 'DNA')) .
                              filter                          regex('"
.$_POST['fruiting_bodies']."', 'Present') .
                              filter                          regex('"
.$_POST['external_decay']."', 'Firm_and_dry') .
                              filter  regex('"  .$_POST['mycelium']."',
'Absent') .
```

```
                              filter regex('" .$_POST['intdiscolor']."',
'None') .
                              filter  regex('"  .$_POST['sclerotia']."',
'Absent') .
                              filter  regex('"  .$_POST['fruit_pods']."',
'Normal') .
                              filter regex('" .$_POST['fruit_spots']."',
'DNA') .
                              filter    regex('"    .$_POST['COSeed']."',
'Normal') .
                              filter regex('" .$_POST['mold_growth']."',
'Absent') .
                              filter                            regex('"
.$_POST['seed_discolor']."', 'Absent') .
                              filter  regex('"  .$_POST['seed_size']."',
'Normal') .
                              filter                            regex('"
.$_POST['seed_shriveling']."', 'Absent') .
                              filter     regex('"     .$_POST['COR']."',
'Normal')}";


$sparql2 = "PREFIX : <http://localhost/soy_test1#>
                  select ?Descriptor_Type ?Factors ?Values where {
                                                {?PlantDescriptor
:hasType  ?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .
?Factors :hasTOC '" .$_POST['TOC']. "' . ?Factors :hasTOC ?Values .}
                                          union
                                                {?PlantDescriptor
:hasType  ?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .
?Factors  :hasPlantStand  '"  .$_POST['PS'].  "'  .  ?Factors  :hasPlantStand
?Values .}
                                          union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasNormalPrecip  '"  .$_POST['Precip'].  "'  .  ?Factors  :hasNormalPrecip
?Values .}
                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasAbnormalPrecip '" .$_POST['Precip']. "' . ?Factors :hasAbnormalPrecip
?Values .}


                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type  .  ?PlantDescriptor  :hasAttribute  ?Factors  .  ?Factors
:hasNormalTemp '" .$_POST['Temp']. "' . ?Factors :hasNormalTemp ?Values
.}
                              union
```

```
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasAbnormalTemp  '" .$_POST['Temp']. "'  .  ?Factors  :hasAbnormalTemp
?Values .}
                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasHail '" .$_POST['Hail']. "' . ?Factors :hasHail ?Values .}
                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSameCropHist '" .$_POST['Crop_hist']. "' . ?Factors :hasSameCropHist
?Values .}

                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNotWholeAreaDamaged  '"  .$_POST['Area_damaged'].  "'  .  ?Factors
:hasNotWholeAreaDamaged ?Values .}

                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasWholeAreaDamaged   '"   .$_POST['Area_damaged'].   "'   .   ?Factors
:hasWholeAreaDamaged ?Values .}

                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeverity '" .$_POST['Severity']. "' . ?Factors :hasSeverity ?Values
.}
                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedTmt '" .$_POST['Seed_tmt']. "' . ?Factors :hasSeedTmt ?Values .}

                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasGermination '" .$_POST['Germination']. "' . ?Factors :hasGermination
?Values .}

                              union
                                    {?PlantDescriptor           :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasPlantGrowth '" .$_POST['Plant_growth']. "' . ?Factors :hasPlantGrowth
?Values .}
                              union
```

```
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:LeafSpotAbsent   '"   .$_POST['Leafspots-halo'].   "'   .   ?Factors
:LeafSpotAbsent ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasLeafSpotsMarg  '"  .$_POST['Leafspots-margin'].  "'  .  ?Factors
:hasLeafSpotsMarg ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasLeafSpotsSize   '"   .$_POST['Leafspot-size'].   "'   .   ?Factors
:hasLeafSpotsSize ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasLeafShred '" .$_POST['Leaf-Shread']. "' . ?Factors :hasLeafShred
?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasLeafMalf '" .$_POST['Leaf-malf']. "' . ?Factors :hasLeafMalf ?Values
.}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:LeafMildAbsent '" .$_POST['Leaf_mild']. "' . ?Factors :LeafMildAbsent
?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemLodging '" .$_POST['lodging']. "' . ?Factors :hasStemLodging
?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:StemCankersAbsent   '"   .$_POST['stem_cankers'].   "'   .   ?Factors
:StemCankersAbsent ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemCankerLeison   '"   .$_POST['canker-leison'].   "'   .   ?Factors
:hasStemCankerLeison ?Values .}
                              union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemFruitingBodies  '"  .$_POST['fruiting_bodies'].  "'  .  ?Factors
:hasStemFruitingBodies ?Values .}
```

```
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:ExternalDecayAbsent  '"  .$_POST['external_decay'].  "'  .  ?Factors
:ExternalDecayAbsent ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemMycelium '" .$_POST['mycelium']. "' . ?Factors :hasStemMycelium
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemIntDiscolor   '"   .$_POST['intdiscolor'].   "'   .   ?Factors
:hasStemIntDiscolor ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasStemSclerotia    '"    .$_POST['sclerotia'].    "'    .    ?Factors
:hasStemSclerotia ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalFruits_Pods   '"   .$_POST['fruit_pods'].   "'   .   ?Factors
:hasNormalFruits_Pods ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasDiseasedFruits_Pods  '"  .$_POST['fruit_spots'].  "'  .  ?Factors
:hasDiseasedFruits_Pods ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalSeed '" .$_POST['COSeed']. "' . ?Factors :hasNormalSeed ?Values
.}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasMoldGrowth '" .$_POST['mold_growth']. "' . ?Factors :hasMoldGrowth
?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedDiscolor   '"   .$_POST['seed_discolor'].   "'   .   ?Factors
:hasSeedDiscolor ?Values .}
                              union
                                   {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedSize '" .$_POST['seed_size']. "' . ?Factors :hasSeedSize ?Values
.}
                                union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasSeedShriveling  '"  .$_POST['seed_shriveling'].  "'  .  ?Factors
:hasSeedShriveling ?Values .}
                                union
                                    {?PlantDescriptor            :hasType
?Descriptor_Type . ?PlantDescriptor :hasAttribute ?Factors . ?Factors
:hasNormalRoots '" .$_POST['COR']. "' . ?Factors :hasNormalRoots ?Values
.}




                                filter  (regex('"  .$_POST['TOC'].  "',
'July')  ||  regex('"  .$_POST['TOC'].  "',  'August')  ||  regex('"
.$_POST['TOC']. "', 'October')) .
                                filter   (regex('"   .$_POST['PS'].   "',
'Normal') || regex('" .$_POST['PS']. "', 'DNA')) .
                                filter  (regex('" .$_POST['Precip'].  "',
'Normal') || regex('" .$_POST['Precip']."', 'Less_than_normal')) .
                                filter   (regex('"   .$_POST['Temp'].   "',
'Normal') || regex('" .$_POST['Temp']."', 'Greater_than_normal')) .
                                filter   (regex('"   .$_POST['Hail'].   "',
'Yes')   ||   regex('"   .$_POST['Hail'].   "',   'No')   ||   regex('"
.$_POST['Hail']. "', 'DNA')) .
                                filter (regex('" .$_POST['Crop_hist']. "',
'Same_last_year')     ||     regex('"     .$_POST['Crop_hist'].     "',
'Same_last_two_years')) .
                                filter (regex('" .$_POST['Area_damaged'].
"',   'Upper_areas')   ||   regex('"   .$_POST['Area_damaged'].   "',
'Whole_field')) .
                                filter regex('" .$_POST['Severity']. "',
'Severe') .
                                filter (regex('" .$_POST['Seed_tmt']. "',
'Fungicide') || regex('" .$_POST['Seed_tmt']. "', 'None') || regex('"
.$_POST['Seed_tmt']. "', 'DNA')) .
                                filter  (regex('"  .$_POST['Germination'].
"',  '90-100')  ||  regex('"  .$_POST['Germination'].  "',  '80-89')  ||
regex('"  .$_POST['Germination'].  "',  'Less_than_80')  ||  regex('"
.$_POST['Germination']. "', 'DNA')) .
                                filter  regex('"  .$_POST['Plant_growth'].
"', 'Abnormal') .
                                filter     regex('"     .$_POST['COL']."',
'Abnormal') .
                                filter     regex('"    .$_POST['Leafspots-
halo']."', 'Absent') .
```

```
                              filter    regex('"    .$_POST['Leafspots-
margin']."', 'DNA') .
                              filter     regex('"     .$_POST['Leafspot-
size']."', 'DNA') .
                              filter regex('" .$_POST['Leaf-Shread']."',
'Absent') .
                              filter  regex('"  .$_POST['Leaf-malf']."',
'Absent') .
                              filter  regex('"  .$_POST['Leaf_mild']."',
'Absent') .
                              filter     regex('"     .$_POST['COS']."',
'Abnormal') .
                              filter   (regex('"   .$_POST['lodging']."',
'Yes') || regex('" .$_POST['lodging']."', 'No')) .
                              filter                            regex('"
.$_POST['stem_cankers']."', 'Absent') .
                              filter      regex('"      .$_POST['canker-
leison']."', 'Tan') .
                              filter                            regex('"
.$_POST['fruiting_bodies']."', 'Absent') .
                              filter                            regex('"
.$_POST['external_decay']."', 'Absent') .
                              filter   regex('"   .$_POST['mycelium']."',
'Absent') .
                              filter regex('" .$_POST['intdiscolor']."',
'Black') .
                              filter  regex('"  .$_POST['sclerotia']."',
'Present') .
                              filter  regex('"  .$_POST['fruit_pods']."',
'Normal') .
                              filter regex('" .$_POST['fruit_spots']."',
'DNA') .
                              filter    regex('"    .$_POST['COSeed']."',
'Normal') .
                              filter regex('" .$_POST['mold_growth']."',
'Absent') .
                              filter                            regex('"
.$_POST['seed_discolor']."', 'Absent') .
                              filter  regex('"  .$_POST['seed_size']."',
'Normal') .
                              filter                            regex('"
.$_POST['seed_shriveling']."', 'Absent') .
                              filter     regex('"     .$_POST['COR']."',
'Normal')}";
```

```
$result1 = sparql_query( $sparql1 );
$result2 = sparql_query( $sparql2 );


if(  !$result1  ||  !$result2  )  {  print  sparql_errno()  .  ":  "  .
sparql_error(). "\n"; exit; }

$fields1 = sparql_field_array( $result1 );
$fields2 = sparql_field_array( $result2 );


$num_of_rows1 = sparql_num_rows( $result1 );
$num_of_rows2 = sparql_num_rows( $result2 );

//print "</p>$num_of_rows1</p>";
//print "</p>$num_of_rows2</p>";

//print "<p>Number of rows: ".sparql_num_rows( $result )." results.</p>";


if($num_of_rows1 != 0){
      print "<h1 style='text-align:center; color:Green'>Soybean Disease
Analysis</h1>";
      print "<br>";
      print  "<h3  style='text-align:center;'>Disease  Detected  :</h3><h3
style='text-align:center; color:Red'>Diaporthe_stem_canker</h3>";

      print "<table border=\'1\'; align=\"center\">";
      print "<tr>";
      foreach( $fields1 as $field )
      {
            print "<th>$field</th>";
      }
      print "</tr>";
      while( $row = sparql_fetch_array( $result1 ) )
      {
            print "<tr>";
            foreach( $fields1 as $field )
            {
                  print "<td>$row[$field]</td>";
            }
            print "</tr>";
      }
      print "</table>";
      }

else if($num_of_rows2 != 0){
      print "<h1 style='text-align:center; color:Green'>Soybean Disease
Analysis</h1>";
```

```
      print "<br>";
      print  "<h3  style='text-align:center;'>Disease  Detected  :</h3><h3
style='text-align:center; color:Red'>Charcoal_rot</h3>";

      print "<table border=\'1\'; align=\"center\">";
      print "<tr>";
      foreach( $fields2 as $field )
      {
            print "<th>$field</th>";
      }
      print "</tr>";
      while( $row = sparql_fetch_array( $result2 ) )
      {
            print "<tr>";
            foreach( $fields2 as $field )
            {
                  print "<td>$row[$field]</td>";
            }
            print "</tr>";
      }
      print "</table>";
      }

else
      print "No match";


      //print $_POST['Leaf-Shread'];
?>
```

Appendix B: Expert Derived Rules for Soybean Diseases

D1:Diaporthe-Stem-Canker
[TOC='August'^Precip>'Normal'^Stem_cankers='Above_second_node'^Fruiting_bodies='Present'^Fruits_pods='Normal'^Temp>='Normal'^Canker_leison_color='Brown'^Crop_hist='Same_last_year']

D2:Charcoal-Rot
[TOC='July…August'^Precip<='Normal'^Temp>='Normal'^Plant_Growth='Abnormal'^COL='Abnormal'^COS='Abnormal'^Scelorotia='Present'^Roots='Rotted'^IntDiscolor='Black'^Area_damaged='Upper_areas'^Severity='Severe'^Seed_size>'Normal'^Crop_hist='Same_last_two_years']

Appendix C: Data Sets Used for Disease Diagnosis

1. Data Set for D1: Diaporthe-stem-canker

- Data set 1: (October, Normal, Greater_than_normal, Normal, Yes, Same_last_year, Low_areas, Potentially_severe, None, 90-100, Abnormal, Abnormal, Absent, DNA, DNA, Absent, Absent, Absent, Abnormal, No, Above_second_node, Brown, Present, Firm_and_dry, Absent, None, Absent, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)
- Data set 2: (August, Normal, Greater_than_normal, Normal, Yes, Same_last_two_years, Scattered, Severe, Fungicide, 80-89, Abnormal, Abnormal, Present, DNA, DNA, Absent, Absent, Absent, Abnormal, Yes, Above_second_node, Brown, Present, Firm_and_dry, Absent, None, Absent, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)
- Data set 3: (July, Normal, Greater_than_normal, Normal, Yes, Same_last_year, Scattered, Severe, Fungicide, Less_than_80, Abnormal, Abnormal, Absent, DNA, DNA, Absent, Absent, Absent, Abnormal, Yes, Above_second_node, DNA, Present, Firm_and_dry, Absent, None, Absent, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)

2. Data Set for D2: Charcoal_rot

- Data set 1: (October, Normal, Less_than_normal, Greater_than_normal, Yes, Same_last_year, Whole_field, Severe, Fungicide, 90-100, Abnormal, Abnormal, Absent, DNA, DNA, Absent, Absent, Absent, Abnormal, Yes, Absent, Tan, Absent, Absent, Absent, Black, Present, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)
- Data set 2: (August, Normal, Less_than_normal, Normal, No, Same_last_year, Whole_field, Severe, Fungicide, 80-89, Abnormal, Abnormal, Absent, DNA, DNA, Absent, Absent, Absent, Abnormal, No, Absent, Tan, Absent, Absent, Absent, Black, Present, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)
- Data set 3: (July, Normal, Less_than_normal, Normal, Yes, Same_last_year, Upper_areas, Severe, None, 90-100, Abnormal, Abnormal, Absent, DNA, DNA, Absent, Absent, Absent, Abnormal, Yes, Absent, Tan, Absent, Absent, Absent, Black, Present, Normal, DNA, Normal, Absent, Absent, Normal, Absent, Normal)

References:

1. R.S.Michalski, R.L.Chilauski, International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, 1980, "**Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition**"
2. Kamran Munir, Mohammed Odeh, Peter Bloodsworth, and Richard McClatchey, Center for Complex Cooperative Systems (CCCS), University of the West England (UWE), October 23, 2009 "**Using Assertion Capabilities of an OWL-based Ontology for Query Formulation**"
3. Hai Dong, RMIT University, Farookh Khadir Hussain, University of Technology, Sydney, Elizabeth Chang, Curtin University, Conference Paper, January 2007, "**Application of Protégé and SPARQL in the Field of Project Knowledge Management**"
4. Timea Bagosi, Diego Calvanese, Sarah Komla-Ebri, Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, Free University of Bozen-Bolzano, Bolzano, Italy, Joseph Hardi, Obidea Technology, Jakarta, Indonesia, Mariano, Rodriguez-Muro, IBM T.J. Watson Research Center, NY, USA, "**The *Ontop* Framework for Ontology Based Data Access**"
5. Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, Free University of Bozen-Bolzano, Bolzano, Italy, Joseph Hardi, Obidea Technology, Jakarta, Indonesia, Mariano, Rodriguez-Muro, IBM T.J. Watson Research Center, NY, USA, Roman Kontchakov, Brickbeck, University of London, "**Ontop: Answering SPARQL Queries over Relational Databases**"
6. https://www.w3.org/standards/semanticweb/ Last visited: 04-04-2017
7. https://www.w3.org/TR/owl-features/ Last visited: 02-04-2017
8. http://www.h2database.com/html/tutorial.html Last visited: 17-03-2017
9. https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes Last visited: 23-03-2017
10. https://www.w3.org/TR/sparql11-query/ Last visited: 28-03-2017
11. https://github.com/ontop/ontop/wiki/ontopProMappingsTab Last visited: 12-03-2017
12. https://github.com/ontop/ontop/wiki/Easy-Tutorial:-Using-Ontop-from-Protege Last visited: 15-03-2017
13. https://github.com/cgutteridge/PHP-SPARQL-Lib/blob/master/sparqllib.php Last visited: 24-03-2017
14. https://stackoverflow.com Last visited: 30-03-2017