A SEMANTIC WEB FRAMEWORK FOR REPRESENTING, LINKING AND ANALYZING MEDICAL DATA FOR OPTIMIZING LABORATORY UTILIZATION

by

Suria Kala Subbu

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2016

To the Almighty God for His blessings,

To my Dad Mr. Subbu and Mom Mrs. Saroja for their endless love,

To my siblings Suresh, Latha and Chinu for their encouragement,

To my Friends Praveen and Ankit for their love and moral support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this thesis, we investigate semantic web based methods for representing, linking and analyzing medical data. The main challenge addressed in this work is the transformation of data stored in a relational database to an ontological model that allows to represent as RDF triples and to link the data with external data sources using linked data principles. We have implemented a semantic analytics framework that comprises the following elements: (a) Domain-specific ontology to represent the data model and data inference. (b) RDMS data extraction using a domain-specific ontology (TBOX) based on the relational database schema; (c) Ontology instantiation (ABOX) that involves converting the relational data in terms of RDF triples. A key feature of our approach is the data is not physically migrated from the RDBS to RDF, rather we dynamically materialize the RDF triples thus avoiding the creation of a large RDF dump; (d) Linking the RDF data with available open data in RDF format using ontology-based concept alignments; and (e) Semantic analytics using SPARQL to identify semantically-salient patterns within the data. We have applied our semantic analytics data to analyze pathology lab data (over 5 years of pathology order data), where we were able to identify prevalent order-sets inherent within the data, and we also evaluated the change in the frequent order-sets over a five year time period.

LIST OF ABBREVIATIONS USED

AACC	American Association of Clinical Chemistry
API	Application Program Interface
COPD	Chronic Obstructive Pulmonary Disease
ETL	Extract Transform and Load
EMR	Electronic Medical Records
EHR	Electronic Health Records
IMWF	International Waldenstrom's Macroglobulinemia Foundation
LOD	Linked Open Data
NCBO	National center for Biomedical ontology
NSHA	Nova Scotia Health Authority
OBDA	Ontology based Data Access
OBO	Open Biomedical Ontologies
OSM	Open Street Map
OWL	Ontology web language
RDF	Resource Description Framework
RDFS	RDF Schema
RDMS	Relational Database Management System
SPARQL	SPARQL Protocol and RDF query Language
URI	Uniform Resource Identifier
WWW	World Wide Web
XML	Extensible Markup Language

ACKNOWLEDGEMENT

I would like to thank my supervisor ***Dr.Raza Abidi*** for taking me in as a graduate student in his lab and giving me the opportunity to learn semantic web technology even though I did not have a semantic web background. I would like to thank him for his time, patience and guidance throughout my thesis.

I would also like to thank ***NICHE research group*** for their constructive suggestions and guidance during my thesis.

I would also like to thank ***my family and friends*** who supported me throughout my theses, who were always there to listen to me, and who often provided me with moral support when I needed it most.

CHAPTER 1 INTRODUCTION

The amount of information that can be extracted from a data can be increased to multiple folds if the data can be published on the Web and linked to other associated or relevant data. Data publishing can be made in various methods. The data can be published by simply placing flat files such as PDF, Excel files on a website (Hausenblas, 2008). The drawback here is that it can only be retrieved, cached and indexed, but cannot be analyzed or linked with other data to discover new information. Other alternative is to publish the data as structured files (spreadsheets). This technique allows data analysis but the information in one spreadsheet is not connected to the ones in other spreadsheets resulting in static data silos. Both these techniques have published data as raw dump sacrificing much of its structure and semantics. Classic Web has reduced the difficulty of publishing and accessing documents with the help of hypertext links and also helped in keeping them connected (C. Bizer T. H.-L., 2009). In Classical Web, the relationship between two-linked documents is not explicitly available because the data format (HTML) is not adequately expressive to allow individual entities defined in a particular document to be connected by typed links to relate other entities. The Web has a lot of information available as HTML documents but the raw data itself is not available. Linked Data (Gabriel, 2015) proposes to turn the existing Web of hyperlinked documents into a Web of interlinked data by applying the principles of the Semantic Web (URI, RDF, SPARQL) to share data. The basic idea here is to enable computers to automatically read the data from different sources by connecting and querying the data sources.

1.1 Research Objectives

Our research objective is to investigate Semantic Web methods for representing and linking data to perform semantic analytics. In this research we developed several questions in order to reach the research objectives such as,

- What are Linked Data and Semantic Web? And how Semantic Web methods address the challenges of Linked Data?
- What are the steps involved in transforming a (legacy) dataset into Linked Data? And what are the steps involved in interlinking Linked Data?

1.2 Research Challenges

The task of representation and linking data brings forth the following two challenges:

1.2.1 Technical Challenge

How to model, integrate and transform the data from multiple resources into a format that is feasible for semantic analysis?

1.2.2 Modeling Challenges

How to describe the knowledge model with computer interpretable semantic rich formalism, to facilitate data linking and analysis?

1.3 Solution Approach

We aim to attain our research objectives by exercising Semantic Web features. Our solution approach comprises of conceptual modeling, ontology instantiation, data linking and data analytics using inference. Specifically, this thesis explores the possibility of developing a knowledge-driven, semantic model that can effectively perform data analytics. The scope of this work will be limited to demonstrating the applicability of this Semantic Web approach in the context of efficient data modeling, linking and analysis. Figure 1.1 illustrates the proposed solution approach.

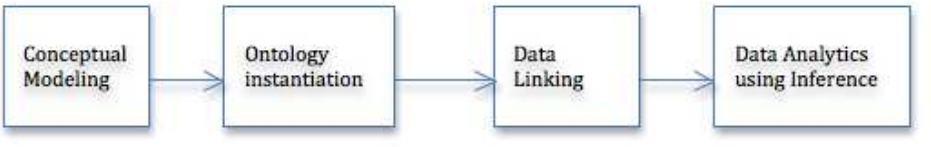

Figure 1.1 Operationalization of our Solution Approach for data analysis

1.4 Contributions

As mentioned in the research objective, it mainly focuses on Semantic Web methods for data linking and representations. We have developed a Linked Data Analytical framework encompassing following elements:

1. Conceptual Modeling of data: Conceptual Modeling involves depiction of universe of discourse represented by modeling languages such as *Ontology Web Language (OWL)*, *Resource Description Framework Schema (RDFS)* etc. We have used ontology for conceptual modeling, as it complies with comprehensible modeling constructs. Noticeable feature of semantic technology is its descriptive nature, which facilitates reuse. Global ontology and Standard vocabulary reuse are taken into consideration while building ontology model.

2. RDB to RDF conversion: Ontology based data access technique is used to obtain the *RDB* data as *RDF* data. Ontology based data access does not involve transformation of RDB data into RDF dump. Instead it involves direct equivalence mapping, which means the source does not change but the output is obtained in desired format as virtual RDF view.

3. Creation of mapping axioms for data linking: This involves construction of mapping axioms that facilitates linking of data with other semantic data. We have constructed mapping axioms that link the relational database and the ontology. In addition, we have also formulated the mapping rules to link the data with other *RDF* data.

4. Semantic Querying: As our process involves data access through ontology and Linked Data, SPARQL queries are written to fetch the required information. SPARQL queries can yield meaningful outputs only if the appropriate reasoner is involved. So we have used a *SPARQL* endpoint that supports the data linking mappings axioms.

5. Analysis of Semantic data: The *RDF* data obtained through *SPARQL* queries are

further investigated to mine patterns that would be helpful.

To achieve the stated objective, we have researched the following criteria's that helps in building the comprehensive system.

1. Study of existing *RDB* to *RDF* mapping tools and approaches to identify the most suitable approach and tool.
2. Study of direct equivalence data mapping and data transformation.
3. Identification of methods for mapping the *RDF* data to the global ontologies.
4. Identification of suitable data retrieval techniques for Linked Data such as SPARQL endpoint.

1.5 Organization of Thesis

Chapter 2 provides details about Linked Data and Semantic Web techniques along with the literature review on core components involved

Chapter 3 defines our research methodology and the tools used.

Chapter 4 explains the in depth details of implementation of developing semantic model for data analyses.

Chapter 5 presents the RDF analysis on developed linked data.

Chapter 6 outlines the evaluation of implemented Semantic Web approach.

Chapter 7 concludes this thesis by highlighting the findings and drawbacks of the proposed system and possible future works.

CHAPTER 2 BACKGROUND

This chapter provides an overview of Linked Data and challenges relevant to the data representation and linking. Semantic Web technologies are at the forefront of Linked Data applications and we will review the specific role of Semantic Web methods for handling Linked Data. In addition, it also provides extensive review of methods and approaches for implementing Linked Data.

2.1 Linked Data

Linked Data is about creating typed links between the data in one data source to data in another data source by publishing and connecting structured data on the Web (C. Bizer T. H.-L., 2009). Technically, Linked Data refers to data published on the Web in such a way that it is machine readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets.

The presence of data, such as Open Data (figure 2.1), on the Web has resulted in a Web of Data. But these are just collection of datasets without explicitly specifying the relationships among data and making it difficult to perform querying and also drawing inferences from the Web of Data. The purpose of Linked Data is to annotate data by ascribing semantics (or meaning) to data so that it can be well understood and in turn be utilized in a systematic and unambiguous manner. Linked data requires the data on the web to be in a standard format so that it can be accessed, linked and analyzed. *Tim Berners Lee* summarized a set of following rules as Linked Data principles to ensure that all published data turns out to be a part of single global data space (Berners-Lee T. , 2006):

1. *Use of URIs to name things.*
2. *Use of HTTP URIs to search those names (aka. dereferencing).*
3. *Use of standards such as RDF and SPARQL to provide helpful information when someone searches a URI.*
4. *Inclusion of links to other URIs to discover more things.*

2.1.1 Naming Things with URI

Uniform Resource Identifiers (URIs) is a globally unique identifier for Web documents, digital contents, real world objects and abstract concepts. In other words, it identifies anything of interest in the data, including entities, classes or concepts and the properties. *URI* supports many schemes such as Uniform Resource Namespace (*URN*), *Digital Object Identifier's (DOI*) and *Hyper Text Transfer Protocols* (*HTTP*) But Linked data uses only *HTTP URIs* as it provides a universal access mechanism describing the identified entity. Example *URI* to represent a person: *http://www.example.com/about#sam.*

2.1.2 Making URIs Dereferenceable

Dereferencing *HTTP URIs* indicates the process of *HTTP* clients looking up the *URI* using *HTTP* protocol and retrieving a description of the resource being identified by the *URI*. The description of resources is represented in the form of Web documents. The Web is considered to be an information space to retrieve resource representations by humans and machines. This is achieved using an *HTTP* mechanism called content negotiation. The process involves *HTTP* clients sending *HTTP* headers and servers inspecting these headers and selecting an appropriate response.

2.1.3 Providing Helpful RDF Information

Linked Data employs Resource Description Framework (*RDF*) as a standardized data format to process Web content. Resources are represented in the form of triples. Each triple has three parts: subject, predicate and object. *SPARQL* is used to query the *RDF* data. The *RDF* data model is explained in more detail in section 2.3.5.

2.1.4 Including Links to Other Things

External *RDF* links are fundamental to Web of Data since it connects data silos into a global, interconnected data space and facilitates applications to find additional data sources. External *RDF* link is technically an *RDF* triple in which subject is a *URI*

reference in the namespace of one data set, while the predicate and/or object are *URI* references pointing into the namespaces of other data sets. Descriptions of the linked resources are obtained by dereferencing the *URI*s, which usually include additional *RDF* links pointing to other *URI*s which in turn can also be dereferenced and so on.

The result can be conceptualized as a Web of Data, where *URI*s identify things, dereferencing *URI*s through *HTTP* returns structured data (RDF) about those things, and the structure's information is comprised of related *URI*s, which constitutes other sources enabling further discovery.

2.1.5 Benefits of Linked Data Principles

Linked Data principles give us a number of benefits:

- The data is placed in context, i.e. each entity has a Web address that can be annotated and referenced, acknowledging explanations and implications to be linked back directly to the data. Thereby, increasing the data quality.
- The data is linked to its information model and to related data, enabling information to be combined across silos and deducing new knowledge out of existing facts. Data integration and browsing through complex data was also made easier because of the linkage.
- The data is accessible at fine grain through Web, so that downstream applications can run from the live data ensuring it is up to date. It also facilitates easy updates and additions of the data models. Static data dumps can also be used if preferred.

2.1.6 Linked Open Data (LOD)

The most obvious exemplar of adoption and application of the Linked Data principles has been the Linking Open Data project. It was started on February 2007 by Chris Bizer and Richard Cyganiak and supported by the *W3C SWEO* (Semantic Web Education and Outreach Group). It was put forth to get real data to work with. It involves taking existing open data sets, converting them to RDF, publishing them on the Web and

linking them together. It has vastly grown and comprises datasets from varied organizations and data providers encompassing media, governments and user generated content. In order to visualize key *Linked Open Data (LOD)* providers and their linkages, Figure 2.1 illustrates the Open Data cloud (Max Schmachtenberg).

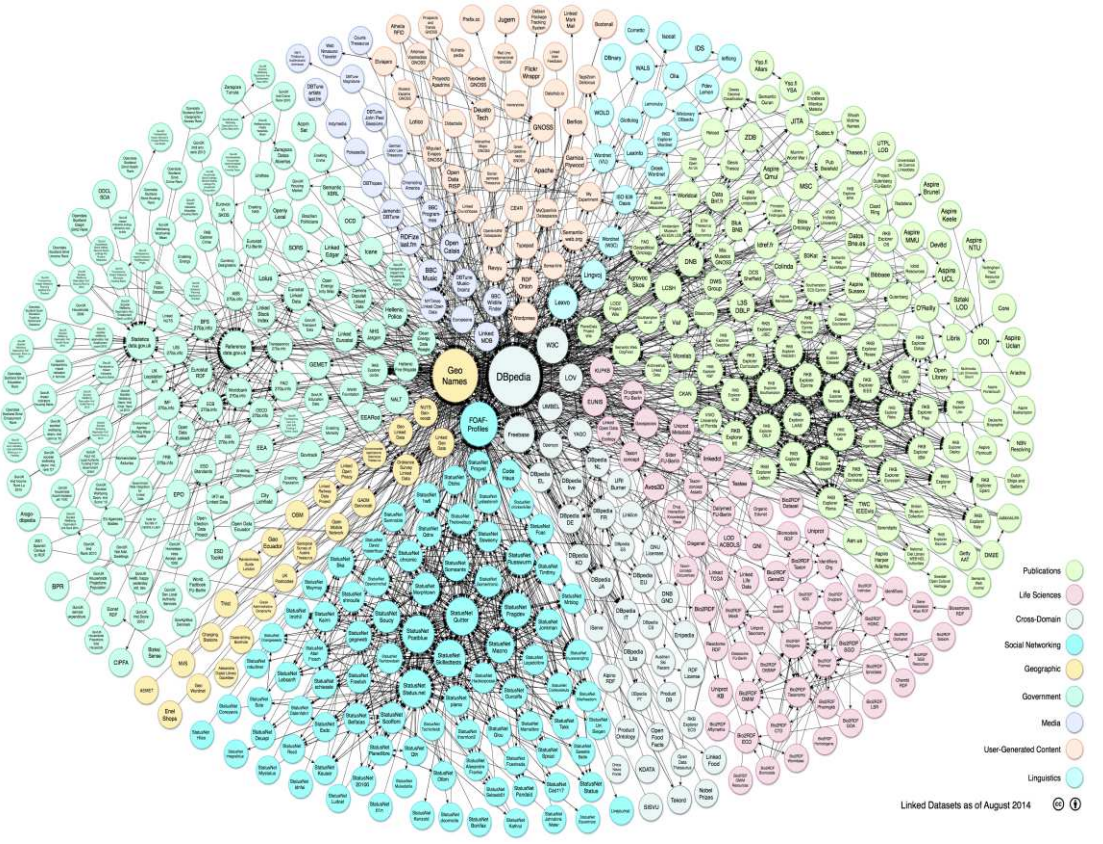

Figure 2.1 Linked Open Data Cloud (Jentzsch, 2014)

2.2 Challenges in Consuming Linked Data

In this section, we explain the core challenges associated with accessing the wealth of information provided in the Web of Data, focusing particularly on the problem of schema mapping, reasoning and data retrieval.

2.2.1 Schema and Ontology Mapping

Schema and Ontology mapping aims at identifying semantic correspondences between metadata structures or models, such as database schemas, XML message formats and ontologies. Most of the Linked Data applications display data from different sources that uses different vocabularies to represent information. This results in multitude of ontologies to represent the knowledge within Linked Data sources. The source and the ontologies must be integrated before it is being displayed to the user. The integration is facilitated by mapping of terms from distinct vocabularies to the applications target schema (C. Bizer T. H.-L., 2009). W3C recommended terminologies like owl:equivalentClass, owl:equivalentProperty, rdfs:subClassOf, rdfs:subPropertyOf to establish basic correspondences in the mappings. These correspondences are too coarse-grained to properly transform data between schemata. Hence, there is a need for fine-grained schema mapping language for combining partial mappings in some cases where data sources mix terminology from different vocabularies. There are many semantic web tools available to address ontology mapping like *MapOnto* (J. Yuan An, 2006), *Silk* (A. Jentzch, 2010), *Karma* (S. Gupta, 2015) etc.

2.2.2 Reasoning over Linked Data

Implicit Knowledge obscured in Linked Data could be captured with semantics of RDFS and OWL alone. However, it does not offer means to express mathematical conversions in RDFS or OWL (J. Pan, 2010). Semantic Web provides a solution in terms of Semantic Web Rule Language (SWRL). SWRL has defined mathematical built-ins for various numeric types such as *swrlb:add, swrlb:subtract, swrlb:multiply, swrlb:divide, swrlb:mod, swrlb:pow*, and so on.

2.2.3 Data Retrieval from Multiple Sources

Linked Data has information from a wide variety of sources in different formats like relational database, *CSV, XML, HTML*, images etc. Data retrieval can be problematic and challenging when an application has to fetch data from different sources. The challenges are due to the differences in formats, structure, semantics and concept labels with

different sources. This can be addressed by using techniques like *crawling*, *caching*, or *on-the-fly link traversal* and *federated SPARQL queries* (S. Shekarpour, 2016).

2.3 Semantic Web Technologies for Data Linking

Linked Data use Semantic Web technologies to perform data linking. Semantic Web technology stack along with data linking challenges that can be addressed by applying Semantic Web is explained.

2.3.1 The Semantic Web

The *World Wide Web* (WWW) is a hypertext system that provides text, audio, video and graphics on the Internet. The key technology that makes World Wide Web powerful is hyperlink. Yet it lacks the ability of machine intelligence (Ahmud-Boodoo, 2016). The World Wide Web has been remarkably successful but it had various shortcomings such as lack of ability to subsequent processing by machine itself and lack of ability to cross-reference consistently. *Tim Berners-Lee* proposed the Semantic Web as a variation to World Wide Web by enabling machine processing. The Semantic Web relates meaning with content that allows machines to share and exploit the knowledge. This would facilitate computerized agents, sophisticated search engines and interoperable service to empower more intelligent applications and higher degree of automation. Figure 2.2 illustrates the semantic web pyramid.

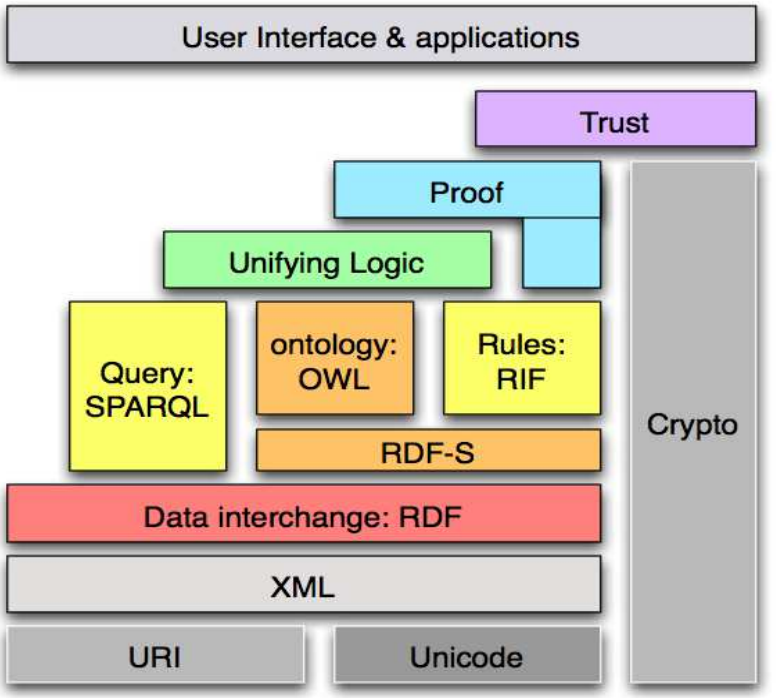

Figure 2.2 Semantic Web pyramid

2.3.2 URI

Linked Data relies on URI that are fundamental to Semantic Web to name resources. Uniform Resource Identifier is a formatted string to identify the abstract or physical resource. An URI can be additionally categorized as a locator, a name or both. A subset of URI, which identifies resources through representation of their primary access mechanism, is referred to as Uniform Resource Locator (URL). A subset of URI that remains globally unique and persistent even when the resource becomes unavailable is referred as Uniform Resource Name (URN).

2.3.3 Unicode

Unicode is a character encoding system, like ASCII, designed to help developers to create software applications that work in any language in the world. Unicode provides a unique number for every character, independent of the underlying platform, program, or language.

2.3.4 XML

Extensible Markup Language (XML) with XML namespace and XML schema definition ensures that there is a common syntax used in the Semantic Web. XML namespaces specifies diverse markup vocabularies in one XML document. XML schema states schema definition of a particular XML document. However, XML has disadvantages when it comes to semantic interoperability.

2.3.5 Resource Description Framework

The first major step towards accomplishing the machine-understandable Web to link Data is *RDF*, recommended by *W3C*. It provides a standardized means of structured and semi-structured data modeling and sharing such that it can be interchanged between *RDF*-aware agents without loss of meaning. It works based on two major assumptions: (i) *Open World Assumption* (OWA) and (ii) *no Unique Name Assumptions* (UNA). First assumption indicates that the data are naturally incomplete and the second one indicates that no centralized naming service i.e. same entity can be identified using diverse identifiers. The RDF data is usually represented in the form of *triples*, which acts as elementary units of the Semantic Web. Each statement states a fact and it consists of three sections such as *subject*, *predicate* and *object*, hence the name triple. The example triple format of tuples is, (Sandra, type, student). The subject of the triple is usually the resource. The object of the triple may contain resource, blank node or a literal value that relates to the subject. The predicate specifies a relationship between subject and object. A set of triples is called as RDF graph. A resource defined by user is named as *URI* and *blank-nodes* represents unnamed resources. It is basically used as a platform for Semantic Web data modeling. Simple ontologies can be built using *RDF*. All information pertaining to Semantic Web is represented and stored using *RDF* (RDF Working Group, 2014). The heaps of the Semantic Web pyramid atop *RDF* assist to build even more expressive complex ontologies. Hence, RDF provides a generic graph based model to structure and link data that describes things in the world.

2.3.6 RDF Schema

RDF has greatly helped in machine understandable web service but it lacked semantic expressivity. *RDF Schema* extends the semantic enrichment of *RDF* by allowing it to attach well-defined relationships between classes and properties by providing additional *RDF Schema* properties, resources and labels. The predefined property rdfs:subclassOf is used as a predicate in a statement to declare that a class is a specialization of another more general class. RDFS provides vocabulary such as rdfs:domain, rdfs:range for describing how properties and classes are intended to be used together in RDF data. The rdfs:domain predicate can be used to indicate that a particular property applies to instances of a designated class. The rdfs:range predicate is used to indicate that the values of a particular property are instances of a designated class. For a property, we can have zero, one or more than one domain or range statements. RDF schema provides a way to specialize relationship between two properties, described by rdfs:subPropertyOf. The group of resources that are RDF Schema classes is itself a class called rdfs:Class. All classes are instances of this class. All things described by RDF are called resources and are instances of the class rdfs:Resource. Rdfs:resource is the class of everything. In addition, it also provides utility properties such as *rdfs:Literals, rdfs:label, rdfs:comment, rdfs:isDefinedBy* and *rdfs:seeAlso.* Rdfs:literals denotes a simple, untyped string. Rdfs:label is used to provide human readable version of the property or class. Rdfs:comment is used to provide human readable description of the property/class. Rdfs:seeAlso relates a resource to another resource that explains it. Rdfs:isDefinedBy relates a resource to the place where its definition is found.

2.3.7 OWL and Ontology

"An ontology is a formal, explicit specification of a shared conceptualization" as defined by Thomas Gruber (T.R, 1995).

Ontology describes a domain using concepts, relationships and properties that are represented by a collection of triples. *Individual* represents the object that is mapped into ontology. Individuals of the same type can be defined as *instances of concepts*.

Classes or concepts represent a group of different individuals that share a common characteristic. Both the classes and individuals have *attributes* to specify their characteristics and properties. A *relation* connects two arbitrary classes or individuals to each other. In addition, ontologies also contain function terms, restrictions, rules, axioms and events. *Function terms* are the structures formed from relations, which is used in terms in each statement. *Restrictions* describe what is true for additional knowledge so that it can be accepted. *Rules* are if-then statement notation to describe and draw logical inferences. *Event* indicates the changes to attributes or relations.

Figure 2.3 illustrates the example ontology model. *Human*, *Male*, and *Female* are concepts. *Jack, Dorothy* and *Rose* are individuals. *Subclass of, is a, has parent and DOB* are properties. '*Subclass of'* defines one concept to be a sub-concept of another concept. '*has parent'* is a relation which associates two individuals to each other. '*DOB'* is a property of the individuals. Explicit facts from above ontology models are: Male and Female are humans. Jack is male while Dorothy and Rose are female. Dorothy has two parents, Jack and Rose.

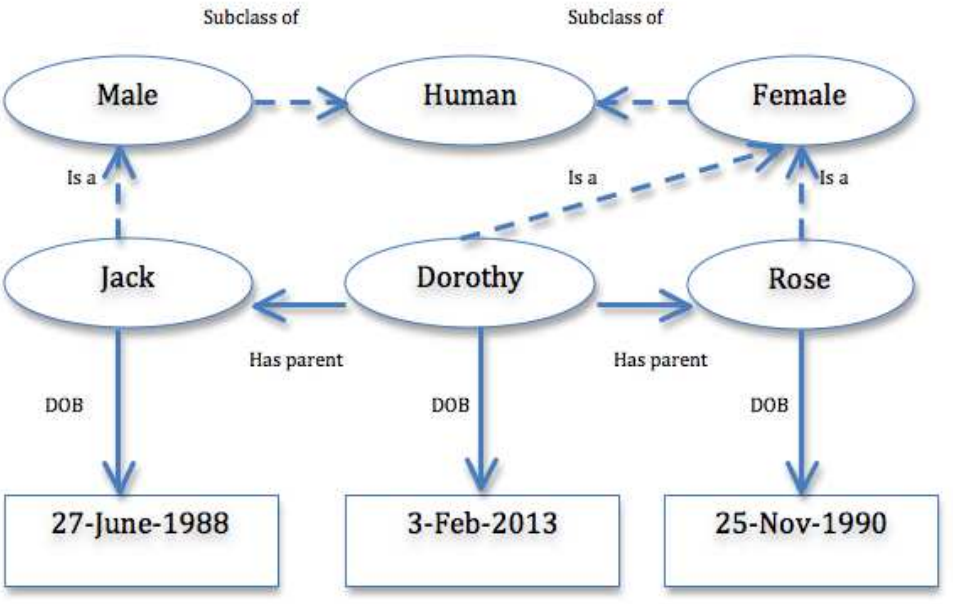

Figure 2.3 Example of Ontology model

Ontology has a special feature named automatic reasoning to infer facts that are implicitly stated in the data model. The *Male* and *Female* are sub-concepts of *Human* and *Jack*, *Dorothy* and *Rose* are instances of *Human*. We could define '*has child'* as an inverse property to '*has parent'*. i.e. the statements *Jack* has child *Dorothy* and *Rose* has child *Dorothy* can be inferred. If a concept *Father* is defined as a *Male* who has at least one child, then *Jack* can be deduced to be an instance of this concept. Similarly, properties such as '*has daughter,* '*has mother', 'has father'* etc. can also be defined.

Ontologies are expressed using OWL (Web Ontology Language). It is an extension upon *RDFS* and is more expressive compared to RDFS (Heflin, 2007). Similar to *RDF* and *RDFS*, *OWL* also abides by the *OWA* and no *UNA* assumptions. *OWL* enables us to define concepts in a way that it can be mixed and matched with other concepts so that it can be linked and reused for diverse applications and functions. Semantic expressivity is improved by *OWL* properties as follows: The property *owl:sameAs* explicitly states that two individuals are the same individual. The property o*wl:differentFrom* specifies that two individuals can never be the same individual. owl:equivalentClass property states that two classes are equivalent. owl:allDisjointClass property states there is no individual that is an instance of more than one class from the defined set of classes. The properties have various characteristics such as owl:inverseOf, owl:propertyDisjointWith, owl:ReflexiveProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty. owl:inverseOf, which indicates that a property is a inverse of another property. owl:propertyDisjointwith indicates that the two properties can be disjoint. owl:Reflexive Property indicates that the property relates everything to itself and owl:FunctionalProperty specifies that every individual can be linked to at most one other individual. owl:InverseFunctionalProperty specifies that the inverse property is functional. Additionally, it also provides extended set for metadata such *as owl:versionInfo* (string providing information about the version*), owl:priorVersion* (statement containing a reference to another ontology indicating that it is a prior version*), owl:backwardCompatibleWith (*indicates that the prior version of ontology is backward compatible*), owl:incompatibleWith (*indicates that the prior version of ontology is not backward compatible) . Ontologies are usually designed using semantic

editors like Protégé (U. Prot, 2011). The reasoners for OWL include Pellet (E. Sirin B. P., 2007), RacerPro (V. Haarslev, 2012), FaCT++ (D. Tsarkov, 2006), Hermit (R. Shearer, 2008) and Ontop Quest (M. Rodriguez- Muro, 2012).

Hence, Ontology provides a uniform, well-structured syntax for domain knowledge representation, whereas relational database traditionally stores and exemplifies relationships in the data. Sometimes, it is extremely advantageous to have benefits of both the high-level depiction of the collected information domain and the traditional stores. Constructing new, highly expressive, knowledge-driven applications by interpreting the stored data will presumably lead to innovations in many areas. Ontologies pave the way for integrating heterogeneous resources. They also play a major role in reusing the existing knowledge with the ability to *reason* on the data. It is also one of the widely used knowledge management tools (R. McKerlich, 2013).

2.3.8 SPARQL

Queries on RDF, RDFS and OWL models are often executed using SPARQL, which abbreviates to *SPARQL Protocol and RDF Query Language*. It is proposed especially to query *RDF* databases across various systems (DuCharme, 2011). The functionality of *SPARQL* is similar to the function of *SQL* in the relational databases. The queries can involve one or more triples where the subject, predicate and/or object can be variables. Figure 2.4 shows the example *SPARQL* query to retrieve all the predicates and objects associated with disease DOID_11786.

```
SELECT ?p ?o
WHERE
{
<http://purl.obolibrary.org/obo/DOID_11786> ?p ?o .
}
```

Figure 2.4 Sample SPARQL Query

?p ?o denotes SPARQL variables that matches any node in the RDF dataset. We have used a single triple pattern *(<http://purl.obolibrary.org/obo/DOID_1178> ?p ?o*) to

retrieve multiple properties about a particular resource. The SELECT result clause returns variables that satisfy the query.

2.3.9 Logic, Proof and Trust

The Logic layer enhances the ontology language and allows writing application-specific declarative knowledge. For the semantic web to become more expressive enough to help in a wide range of situations, it is necessary to construct a powerful logic language for making inferences. The proof layer involves the actual deductive process as well as the representation of proofs in Web languages. Finally, the Trust layer involves digital signatures and other kinds of knowledge, based on recommendations by trusted agents, certification agencies and consumer bodies.

2.4 Data Linking Techniques

Data Linking refers to the process of linking two object descriptions to signify the point that both the objects refer to the same real-world object in a given domain or the point that some kind of relation holds between them. It usually takes one or more collections of datasets as input and outputs a collection of mapping set. *Ferrara* and *Nikolov* has classified data linking technique based on dimensions of *granualrity*, *type of evidence* and *source of evidence* (A. Ferrara, 2011). Granularity dimension can be categorized into 3 types. (i) *Value matching* involving identification of equivalence between property value of instance. (ii) *Individual matching* involves identification of two individuals representing the same real-world object. (iii) *Dataset matching* involves construction of optimal alignment by considering all individuals in the datasets into account. Second dimension, type of evidence categorize the data linking into 2 types. (i) *Data-level* that utilize the information defined at individual level / A-Box. (ii) *Knowledge-level* that utilize the knowledge defined at ontological schema / T-Box as well as knowldege from external resources. Third dimension, source of evidence is classified into *internal* and *external* based on the information utilization. Below we review recent works that employs semantic mapping for linking data.

[1] The steps involved in data linking task along with the data linking classification was explained in this paper (A. Ferrara, 2011). In addition, Survey is done on eleven data linking tools such as *LN2R* (N. Pernelle, 2007), *Objectcoref* (W. Hu, 2011), *Okkam* (E. Ioannou, 2010), *RKB-CRS* (H. Glaser, 2008), *LD-Mapper* (Y. Raimond, 2008), *Silk* (J. Volz, 2009), *Limes* (Auer, 2011), *Knofuss* (A. Nikolov, 2008), *RDF-AI* (F. Scharffe, 2009), *Serimi* (S. Araujo) and *Zhishi.links* (X. Niu, 2011). Based on the survey conducted, it suggests three possible improvements for data linking at the dataset level of granularity: (i) Sharing schema alignments using server (R2R) to enable reuse for every data linking task with aligned schemas. (ii) Attaching schemas with information about the techniques, thus eliminating the need to specify it in each data linking task.(iii) Use of statistical analysis to automatically determine the properties needed to compare the identity of two instances. This helps in identifying data linking tasks along with appropriate data linking classification suitable for the application.

[2] Various use cases demonstrating the mapping and linking of life science data and health data with other linked data using semantic methods is discussed in this paper (M. Marshall, 2012) which has directed to the recognition of *general data workflow* for mapping healthcare and life science data to RDF and linking it with other Linked Data sources. Set of practices and principles for authoring Linked Data publications are proposed such as (i) Use of mapping instead of migration or data conversion for creation of RDF view (ii) Use of standard vocabularies. (iii) Publish RDF so that it can be discovered. (iv) Assigning a graph URI and addition of provenance and metadata about the graph which makes it possible for crawlers and visitors to know about the graph. These recommendations are useful for identifying the tools, techniques for Linked Data.

[3] A new tool named *Karma* that engages semantic approach for extraction, linking and integration of geospatial data from varied sources have been implemented (Y. Zhang, 2013). It proposes semi-automatic approach for building mappings to translate data in structured sources to RDF. This system automatically infers the mappings and provides a user interface to select the mappings, which is done by domain experts. These mapping rules are used to produce a semantically rich RDF data. This work mainly focuses on the ability to interactively model sources with a chosen vocabulary and to publish it in RDF.

Semantic matching algorithms are used for data linking and *SPARQL* queries for integration.

[4] *Muriela* and *Gerald* has emphasized the importance of Linked Data in the future internet and also suggests the new possibilities to support data linkage by embedding linkage behaviours as properties of Future Internet resources (M. Foulonneau,2010). Data linking is done through proactive linking mechanisms based on the Content Centric infrastructure. It involves addition of semantics at the data packets level so that basic semantic content retrieval can be done at network level and addition of interactivity at the content level. Datasets are interlinked either by using an URI or through identification of similar concepts in other datasets.

2.5 Architecture of Linked Data Application

A Linked Data application usually contains three-tier architecture: presentation layer, data layer and logic layer. The presentation layer provides user interface, which accepts user queries as inputs and outputs the resulting data. In addition, it also has an Application Program Interface. The Logic layer implements the business logic and analytical computation of the application. It retrieves data from different data source and performs analyses and conversion of non-RDF data sources to RDF data. In addition, it also converts the user queries in desired format and aggregates the RDF results.

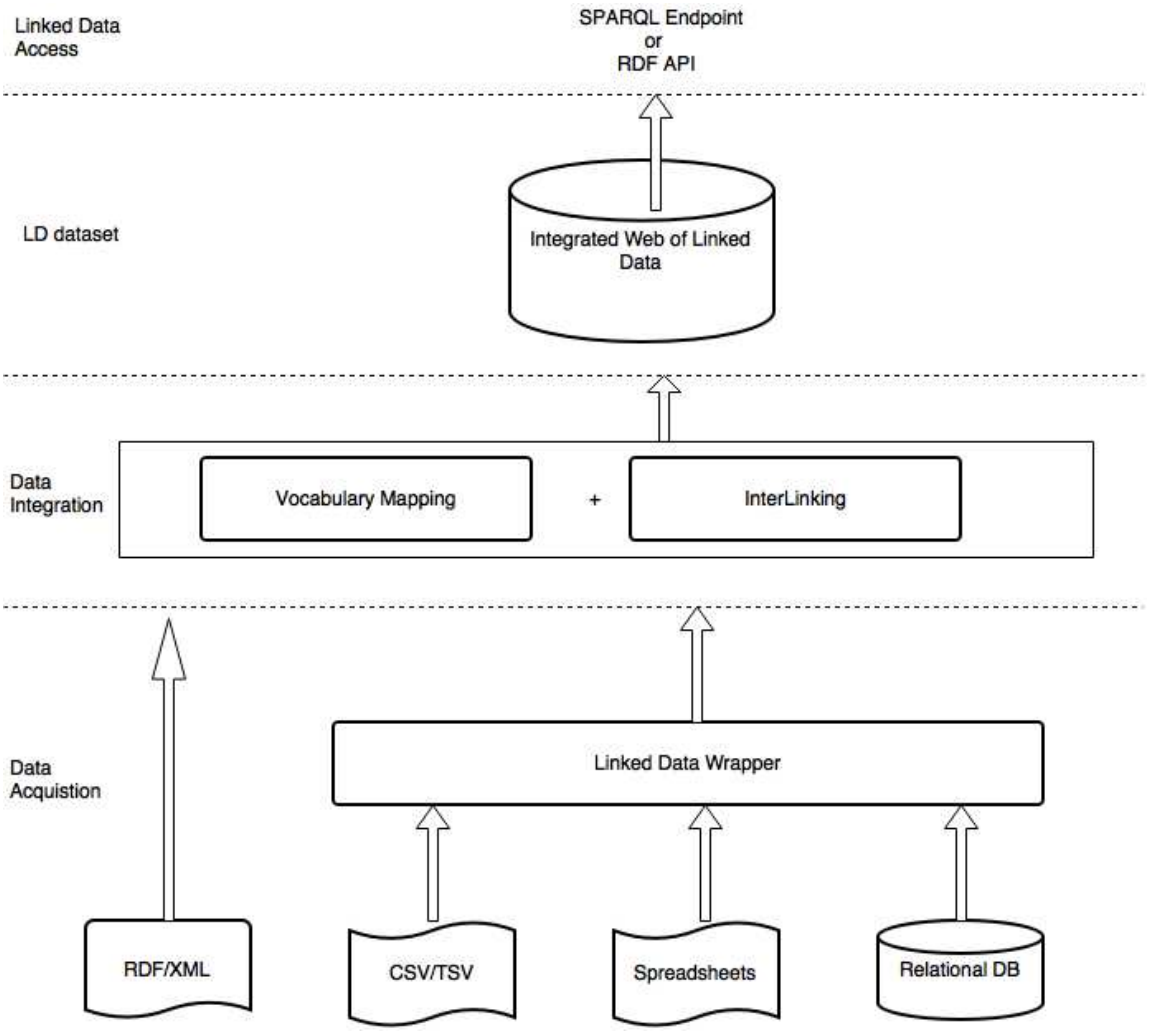

Figure 2.5 General Architecture of Linked Data Application

The data layer represents the underlying data sources. It is mostly responsible for storing data (Triple stores). The data stored is accessed from SPARQL endpoints or RDF dumps. Data layer has wrappers to translate data in desired format. If the data source is relational database, R2RML (S. Das, 2012) can be used to convert the relational data to RDF format. It also supports data cleansing, vocabulary mapping and interlinking. If there exists ambiguities in the dataset, data cleansing is done. Vocabulary mapping and data linking is carried out when more than one resources or ontologies are involved. Figure 2.5 shows the general architecture for linked data applications (Education Curriculum for the usage of Linked Data, 2012).

2.6 Architectural Patterns of Linked Data Applications

The architectural pattern refers to the way the data is consumed and integrated within the application. This patterns depends largely on the specific use case. However, it follows one of the three following architectural patterns(i) *The Crawling Pattern,* (ii) *The On-The-Fly Dereferencing Pattern* and (iii) *The Query Federation Pattern* (Bizer, 2011).

2.6.1 The Crawling Pattern

The architecture of applications that implement this pattern imitates the architecture of common search enginess like *Yahoo* (J. Iturrioz, 2015) and *Google* (Steiner, 2010). This involves traversing RDF links by crawling the Web of data in advance and after that, the integration and cleansing of discovered data happens (Schultz. A, 2014). It can be used in applications that are built on top of open and growing sources. And it has the disdavantage of data replication. *DBpedia* (S. Auer, 2007), *Open Government archives* (Kew) application implements the crawling pattern.

2.6.2 The On-The-Fly Dereferencing Pattern

This Crawling pattern is suitable for application that has data currency and a very high degree of completeness, as the *URI*s are de-referenced and links are followed the instant the application needs data. The disadvantage is that it involves more complex operations (Hasan, 2014). Many Linked Data Browsers follows this architecture including browsers such as *Disco hyperdata browser* (Chris Bizer, 2007), *Tabulator browser* (Berners-Lee et. al., 2006), *Marbles* (Bizer, 2011), *LOD Browser Switch* and Linked Data Search Engines such as *Sig.ma* (G. Tummarello, 2010), *Falcons* (G. Cheng, 2009) and *SWSE* (A. Hogan, 2011).

2.6.3 The Query Federation Pattern

This is mainly used in applications with data sources with both the *SPARQL* endpoints and de-referenceable *URI*s (M. Saleem, 2015). Complex set of queries is sent directly to

data sources. Disadvantage of this pattern is finding query execution plans for complex join queries.

2.7 Using Linked Data

Most of the available data is in flat data files or relational databases. In order to link data from multiple sources, the available data needs to be transformed into semantically rich data representation formalism (such as RDF) so that it can be linked with other data sources. There are wide ranges of Semantic Web tools to perform these data linking tasks.

2.7.1 RDB to RDF Conversion

One of the features of Semantic Web is the power of conceptual modeling (Cambridge Semantics, 2016). By conceptual modeling we mean abstraction of domain, which is feasible, effective, credible and of use. Semantic Web supports various languages such as *OWL, RDF, Turtle* and *N3* that help in constructing a conceptual modeling. In Semantic Web, the conceptual models are termed as ontology. The process of ontology development involves two approaches. (i) It can be created from the scratch. (ii) By *re-using existing ontologies*. Noted feature of Semantic Web is the ability to reuse existing vocabularies and standard ontologies. This saves time and easies the linking and sharing of conceptual model (E.Bontos, 2005). To reuse, contemporarily available ontologies need to be researched for their possible reuse. Also, they can be reused entirely or partially, augmented to the current schemas. Nevertheless, the possibility of a single, available ontology that matches all our requisites may be little. Fresh ontology can be developed in two ways. One technique is building domain specific ontology using the *protégé* tool (Daniel L. Rubin, 2007), with the help of domain knowledge experts. Another method is by generating database schema ontology using *metadata* of the given database (N. Deshpande, 2011). Once the ontology construction technique is chosen, we have to select the mapping description, as it is crucial for data transformation. Generally, mapping description explains how the database is mapped to the RDF representation. It

will be either *direct mapping* or *domain semantics-driven mapping* (P. Heyvaert, 2015). The *domain semantics-driven mapping* reuses pre-existing domain ontologies (McLeod, 2006). It is the same as an ontology population technique where the transformed data are instances of the concepts defined in the ontology schema. This method can be applied only when there are existing ontologies available as per the specification and domain of the RDB. The second type of mapping description is *direct mapping* that makes the semantics coded in the relational schema explicit (M. Arenas, 2012). The created ontology is referred to as database schema ontology. In *direct mapping,* the RDB contents are translated as follows, the tables into ontological classes, columns into properties, rows into resource and cell into literal values (Berners-Lee, 1998). Review of the works related to ontology development based on relational database schema is discussed, as our research mainly focus on semantic representation and linking of relational database.

[1] *DB2OWL* was designed for automatic generation of ontologies from relational database. Mapping is done by detection of particular cases for database conceptual elements and then conversion of database components to the corresponding ontology components (N.Cullot, 2007).

[2] *Shufeng* has created an ontology generator from *RDB* by extracting metadata information from *RDB* with reverse engineering and have analyzed corresponding relationship between *RDB* and ontology, then the ontology generation (S.Zhou, 2010).

[3] *Gherabi* and *Addakiri* proposed an approach for mapping relational database RDB to web semantic. It involves (i) extraction of *MTRDB* (metadata of relational database) from *RDB*, (ii) generation of *CDM* (Canonical Meta Model) based on *MTRDB* and (iii) conversion of *CDM* model to *OWL* ontology based on classification schema structure. This also stores dataset into *OWL* document (Noreddine GHERABI, 2012).

[4] *AGOFRD* implementation was introduced to build ontologies by generation of automatic construction rules using relational database. This method involves (i) Database

metadata reading by employing a tool named "*Data master*", (ii) Ontology meta-model construction with the help of construction rules built by analyzing the relations between primary key and attributes and Goal ontology generation by mapping the database data into ontology instances (L.Ravi, 2012).

[5] Methodology developed by *Marzouk* was based on *XSLT*. The approach categorizes the relational schema tables into Entity, Relational and Composite table. Then the categorized tables are represented as a file *XML* documents validate by a suitable *XML schema*. The transformation of relational database schema to *OWL* is applied as a set of *XSLT* style sheets, which takes *XML* description of relational schema as input and an *OWL* output document comprising the *OWL* classes and their properties (Marzouk, 2013).

[6] Similarly *Ramathilagam* and *Valarmathi* proposed a framework for generating *OWL DL* ontology from the Banking domain *RDB* based on the mapping rules for direct mapping and integrity constraints mapping (C.Ramathilagam, 2013).

[7] A *RDB* to *RDF* approach using graph as middle layer was introduced to increase the semantic richness. Triggers / Events are also taken into consideration to enhance the conversion process (Mona Dadjoo, 2015).

All the above-mentioned literature is studied to extract the consolidated rules that fit the research objective and a Jena program is written to generate the OWL file.

2.7.2 Data Storage and Querying

Ontology Based Data Access is a methodology to query databases via an ontology. It provides direct access to the underlying database that contains dynamic and non-redundant data. Mapping rules facilitate the link between the data and the ontology. It also performs automatic query translation of schema level queries into data level queries to be executed by underlying database. Mapping rules indicates mapping implementation. It signifies to the manner database tuples are translated into individuals

(ontological instances). There are two methods of mapping implementation available, *data materialization* and *on-demand mapping* (F. Michel, 2014). *Data materialization* is an extract, transform and load approach. The data is extracted and converted to desired format and stored and then queried. The drawback of this approach is the dedicated storage space and also that the data is obsolete (LargeTripleStores) (Blin). It increases intricacy of the application as it involves Extract Transform Load (ETL) approach and also the development cost and maintenance. In addition, it generates replicas that consume software resources. Also in this case, the query-answering engine does not have any knowledge about the provenience of the data and the data extraction method, which is a major optimization factor in inference and query answering. For these reasons, this method is not preferred for our approach. However, in *on-demand mapping*, the queries are evaluated in run time against the database. The data remains in the legacy database and so fits well for large dynamic data sets (Borodaenko, 2009). Here the queries to the *RDF* need to be rewritten into *SQL* and this must be done at query evaluation time. On-demand mapping allows implementation of access control policies of *RDMS* and also not creating *RDF* copy, makes sure that the data is up to date. It eradicates the necessity for data duplication and harmonization. In addition to this, it exploits sophisticated algorithms for efficient and faster query answering. Here we are reviewing different OBDA based tools like *D2RQ, Mastro, Ultrawrap, ODEMapster, Virtuoso RDF views* and *Ontop*.

[1] *D2RQ* is the most widely used tool for generating *RDF view* over relational data (*D2RQ*). It uses native *D2RQ* mapping to define mappings between the ontology and the underlying database. *R2RML* mapping is only supported partially. Main problem with the D2RQ system is that it does not have reasoning capabilities and does not support federation. In addition to this, the query re-writing techniques are not efficient, as it results in excessive number of joins. Both the Transient and Persistent Views are supported but the Transient Views are considered to be slow with *D2RQ* (C. Bizer, 2004).

[2] *MASTRO* is a Java based commercial tool for ontology based data access tool in which the ontology is specified in *DL-lite* (D. Calvanese G. D., 2011). It is an extension

of *QuOnto* tool hence it has exceptional query rewriting algorithms in place as well as *OWL 2 QL* reasoning capabilities. *Protégé 4* plugin is also available (G. De Giacomo, 2012).

[3] *Ultrawrap* is a commercial OBDA system, which is a hybridization of materialization and query rewriting. *Ultrawrap* is built based on the works of *Ontop* tool. It uses the same technique as *Ontop* when it comes to query rewriting (J. Sequeda D. M., 2013). Saturated mappings are used for the creation of regular and materialized views. This is the only system that supports ontologies with *transitivity* (J. Sequeda, 2014).

[4] *Morph-RDB* is previously known as *ODEMapster*. It is a Scala based relational database to *RDF* engine that conforms to the *R2RML* specification. Query translator component of *Morph-RDB* has very good query optimizations capabilities that eliminates self-join, left outer join and sub query but it has no support to *inference* (F. Priyatna).

[5] *Virtuoso RDF view* is one of the most powerful commercially available toolkit for Triple stores. It enables *RBD* to Virtual *RDF* graph conversion without the need for materialization of *RDF* datasets. Quad map patterns represent the *RDF* view. *Virtuoso* meta-schema mapping is used to define mappings between *RDB* to triples. No reasoning abilities supported by virtuoso in *OBDA* mode. Hence it is mostly used as a Linked Data publishing solution or as a triple stores. Both the Transient and Persistent Views are supported (O. Software, 2007).

[6] *Ontop* is one of the best open-source java libraries available for *OBDA* systems that have a free plugin for *protégé 5*. Key aspects of *Ontop* are (i) virtual approach to *OBDA* which eliminates materialization, (ii) best *SPARQL to SQL* query rewriting techniques which facilitates answering conjunctive queries in Log space, (iii) *OWL 2 QL* profile that provides open-world reasoning required for data integration and better computational properties. Ontop provides *SPARQL 2 SQL* rewriting by means of both Virtual *RDF* graphs and materialization of triples. (iv) Multiple Database federation is also supported

via *Teiid*. (v) Ontop has its own powerful reasoner named *Quest*, which offers rapid query processing. It is 10x to 500x times quicker than all other available engines (Optique, 2013).

Table 2.1 shows the comparisons between these tools and we have chosen Ontop for the proposed solution. This is mainly because Ontop is available as Protégé plug-in. And also, the features in Ontop like support for virtual and classic RDFs, easy-to-use native mapping language, Java API and SPARQL end-point for querying makes it suitable for the proposed approach.

Feature	**D2RQ**	**Mastro**	**Ultra-wrap**	**ODEMapster/ Morph-RDB**	**Virtuoso RDF view**	**Ontop**
Reasoning	No	OWL 2 QL	RDFS-Plus	No	RDFS OWL inference	Quest engine RDFS, OWL 2 QL, SWRL*
Mapping	Direct mapping, D2RQ, R2RML*	R2RML*[1]	Direct mapping, R2RML, Native	R2RML	R2RML*, Meta schema Mapping	Direct Mapping, Ontop, R2RML
SPARQL-SQL Query Rewriting	Yes	Yes	-	Yes	Yes	Yes
Data Federation	Yes	Yes	No	No	No	Yes
Supported Database	MySQL, Oracle, SQL Server, PostgreSQL, HSQLDB, Interbase	Any JDBC	PostgreSQL, DB2, SQL Server, Oracle	PostgreSQL, MySQL, Monet	MySQL, Oracle, DB2, SQL Server, PostgreSQL, HSQLDB, Interbase. Any SQL-92, SQL-99 ODBC, JDBC	MySQL, PostgreSQL, H2, DB2 Oracle, SQL Server, Teiid
Open Source	Yes	No	No	Yes	No	Yes

1 * indicates Limited support

Table 2.1 Comparison of tools based on its semantic features

2.8 Linked Data Applications

Linked Data is used extensively in various places such as in libraries, Biomedicine and government organizations. Linked Data in libraries has the potential to produce globally interlinked library data (Duraspace). In addition, it also facilitates the process of exchanging and maintaining a global cultural graph of dependable and persistent information. Linked Data in biomedicine signifies a set of principles for vocabulary development or ontology that helps in construction of coordinated family of logical and interoperable ontologies to mitigate the explosive propagation of data in the biomedical domain. On the other hand, linked public data reuses public sector information thereby interlinking government and non-government information. Best practice examples to showcase the power of Linked Data are discussed including data providers (DBpedia & Geonames) and pioneering applications (National Archives & OpenEI).

2.8.1 DBpedia

DBpedia is one of the largest Linked Data hub and data source for web and enterprise applications (J. Lehmann, 2014). It was built with the aim to extract structured information from Wikipedia and serve the extracted information as Linked Data on the web. Wikipedia typically comprises of free text as well as structured information (images, category information, hyperlinks, infobox template, disambiguation information, etc.), which are extracted and embedded into DBpedia as N-Triples dump stored in Virtuoso triple store that enables users to execute sophisticated queries through SPARQL endpoint. Figure 2.6 illustrates the architecture of DBpedia.

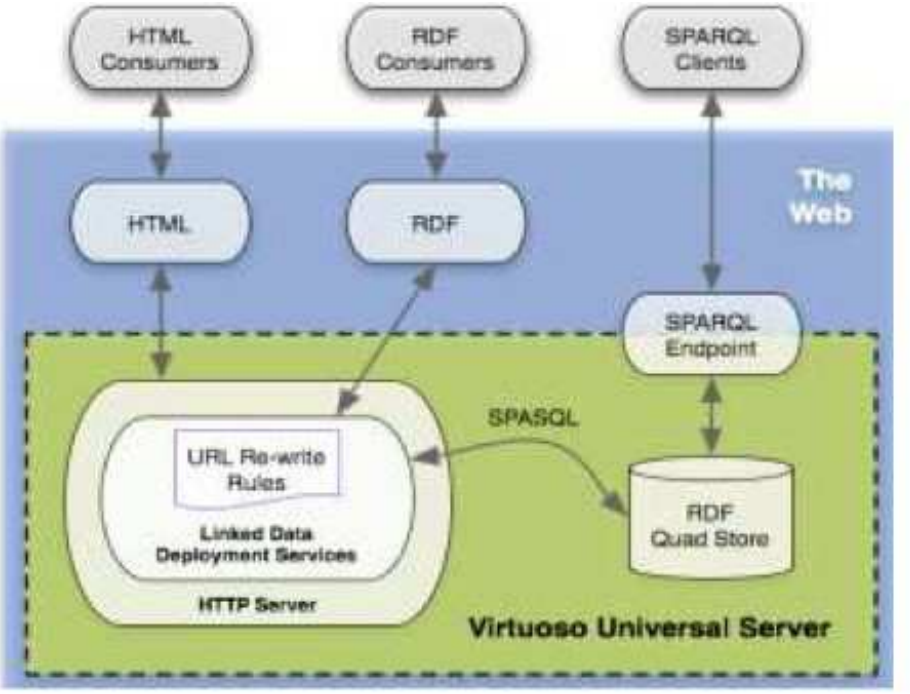

Figure 2.6 DBpedia Architecture (A. Ismail, 2015)

Over the past few years, there is an increase in the number of data publishers in setting data links to DBpedia concepts, which has made the DBpedia a fundamental interlinking hub for the emerging web of data (G. Kobilarov, 2009). DBpedia acts as a linking hub connecting various domains by means of semantic relations. Media like BBC employs DBpedia as main source to link data to other public data.

2.8.2 Linked Geo Data

Linked Geo Data (C. Stadler, 2011) is another geospatial source that extracts the data from Open Street Map (OSM) project and convert it into Linked Data. It provides information about more than 350 million spatial features. DBpedia, Geonames and Linked Geo Data are linked to each other (S. Hahmann, 2010). Geonames serves as a hub for the datasets that have some geographical component (3Kbo). It provides RDF descriptions of millions of geographical locations worldwide. Geonames database is an open-license geographical database that publishes Linked Data about 8 million locations. Figure 2.7 illustrates the architecture of Linked Geo Data .

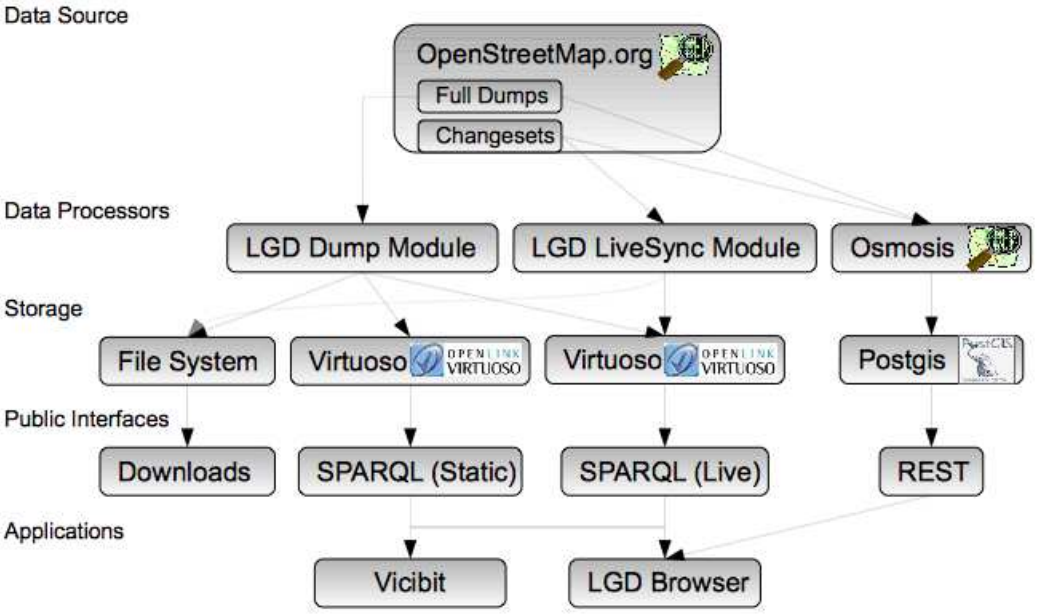

Figure 2.7 LinkedGeo Data Architecture

(C. Stadler, LinkedGeoData: A core for a web of Spatial Open Data, 2011)

LinkedGeoData stores the mapping information and OSM data in a relational database. It uses Sparqlify, a SPARQL-to-SQL rewriter to generate RDF. The following Mapping Tables are used to relate the OSM tags to corresponding RDF resources:

- lgd_map_datatype(k, data type): Values of the given keys should be interpreted as members of a certain data type (boolean, int or float).
- lgd_map_literal (k, property, language): Values of the given keys are treated as plain literals having the specified language tag.
- lgd_map_resource_k (k, property, object): The presence of a key yields triples with the given property and object.
- lgd_map_resource_kv (k, v, property, object): Similar to above, except that the value is also taken into account.
- lgd_map_property (k, property): Relates keys to corresponding properties.

The LinkedGeoData RDB2RDF Mapping is then used to map the relational OSM database to RDF. The transformation helps in retrieval and interlinking of spatial data with greater level of granularity.

2.8.3 Open Energy Information (OpenEI)

It is a combined knowledge-sharing program, which provides free and open access to energy related data, models, tools and information (NREL). OpenEI mission is to share data by adopting Linked Data principles. All the information available on OpenEI is available to public in a variety of machine-readable formats (API, RDF and SPARQL) so that it can be used instantly by other data savvy enterprises. It relates terms and definitions from other sources like DBpedia and Reegle through RDF or SPARQL endpoints in real time to always get only the current information. By linking OpenEI definition to Wikipedia, other Wikipedia linked definitions by DBpedia and Reegle can also be accessed using OpenEI. This project demonstrates that linking multiple definitions to a central concept can improve the human and machine understandability of the concept. Further, this allows outsourcing selected contents in the site to the experts.

2.8.4 The National Archives (Linked Government Data)

United kingdom has come with three projects that are best practice example of semantic web and linked data (Sheridan).

(i) ***data.gov.uk*** : It uses Linked Data to promote the transparency agenda of the government. The UK government uses this project to publish government data using Linked Data standards.
(ii) ***legislation.gov.uk*** : It is United Kingdom's official government archive that unites legislation from across the UK in one place. Also it is a part of initiative towards a Linked Data statute book, which is a step change in transparency, providing full access to the statute book as open data.
(iii) ***nationalarchived.gov.uk*** : semantic knowledge base for the web archive.

2.9 Concluding Remarks

We have presented the Linked Data principles and how the Semantic Web technologies help in addressing the challenges of data linking. We have reviewed literature associated with data modeling and data linking for data analysis purpose based on the Semantic Web technology. We have also highlighted some studies related to ontology engineering approaches. Based on the reviews done here from different data linking techniques to the survey of tools available to work with Linked Data, informed choices were made in the implementation of the proposed approach.

CHAPTER 3 RESEARCH METHODOLOGY

This chapter presents the series of steps undertaken to address the problem statement. Our focus will be on the data representation and linking framework for analysis with firm grounding in Semantic Web. Subsequent chapters will present the implementation details and evaluation strategy. Figure 3.1 shows the steps involved in our research methodology.

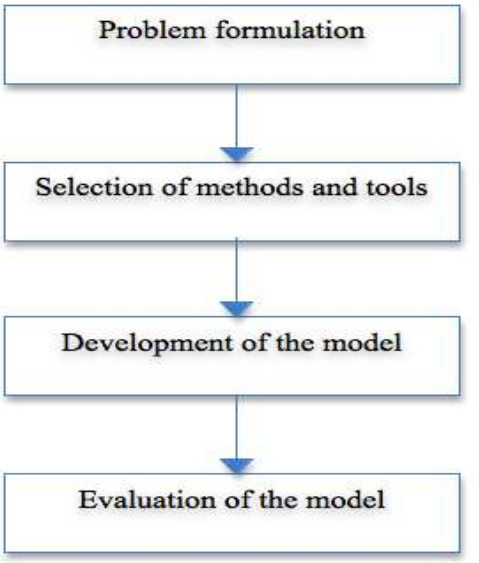

Figure 3.1 Steps in research methodology

3.1 Problem Formulation

First and foremost step in any research work is formulating the problem statement. Our problem statement is formulated in an attempt to identify the efficient data representation and linking techniques. Our problem statement is, "*To Investigate Semantic Web methods to implement Linked Data framework for efficient data representation, linking and analysis*".

3.2 Selection of Linked Data Methods and Tools

Detailed review was done (section 2.7) to select the methods and tools that are appropriate for achieving stated goal. Tools are selected based on the semantic functionality supported. Since the data source we are using is relational database, we have restricted our focus on relational database related tools. The RDB to RDF conversion techniques listed in section 2.7.1 is based on database schema. Most of the techniques does not provide an implementation or tool and just described the method. The tools available convert both the schema and the data. We aim for an ODBA method for data access, which requires converting only the data from RDB to RDF and not the schema. So, in the solution design the RDB to RDF conversion is done programmatically using the consolidated rules to generate OWL file. *Protégé* is used to load the generated OWL file and manage the ontology. In our approach, the data linking is done at the knowledge-level. This is because, we are using OBDA for data access and so the data will not be available at linking stage. The linking is done based on the ontology or database schema. For reasoning and data access, different Sematic Web development tools available for OBDA systems were reviewed with respect to their functionalities and *Ontop Quest* is chosen.

3.2.1 Protégé

Many ontology construction tools are available, such as *OntoEdit* (K. Vasconcelos), *TopBraid Composer*, *WebODE* (C. Arpirez, 2001), *Protégé*, *Neon Toolkit* (P. Haase, 2008) and *WebOnto*. In our system, we have used *Protégé* as it has many competitive advantage compared to all other tools. For many years, *protégé* has been used for knowledge acquisition of domain apart from building domain ontology. It also supports integration and alignment of existing ontologies by means of robust plug-in architecture, which makes it extremely extensible. Output file format can be personalized to accommodate any formal language. These qualities make it a meta-tool for conceptual domain ontology building.

3.2.2 Ontop Quest

A system that can support dynamic materialization using *Ontology Based Data Access*

needs to be identified. It should have the capability to query *SPARQL* under *OWL 2 QL* or *RDFS inference rules*. Our dataset is immense so the system should perform efficiently in the presence of huge volumes of data and large expressive ontologies. All this features are supported by tool called *Quest.* It employs extremely enhanced query rewriting techniques. It relies on database for query execution and *ABOX* storage thereby avoiding redundant inferences. It benefits all major relational databases. The Ontop *Quest* generates the *SQL* queries that are well tuned with respect to the potentials of *SQL* engines. By using *OBDA*, It offers the capability to handle with huge volumes of data from varied resources. It involves 2 phases during Query answering which make its competent. First step involves *Perfect query rewriting* and the next step is *evaluating the query*. It also involves the *Query optimization* procedure (D. Calvanese B. C.).

3.3 Developing the Semantic Model

We have considered a knowledge-based approach for the design and implementation of solution approach for data analysis. The step-by-step detail of the modeling approach is the topic of the next chapter. The next subsection encompasses the definition of the terms and a gist of the development strategy, without exploring too much into the implementation details.

3.3.1 Definitions

Description and definition of important terms used are present in this section just to ensure readers to interpret as it is intended.

- *Conceptual Model* is a combination of concepts and relationship that represents the domain knowledge.
- *Data Linking* is a technique of publishing structured data to facilitate the process of interlinking.
- *Reasoning* is a logical process to form conclusions or inferences.
- *Semantic mapping* refers to the conversion of data elements from one namespace into another on the semantic web.

3.3.2 Strategy for Implementing the Solution Approach

We have taken Linked Data approach to implement the solution. Our strategy utilizes conceptual modeling and data linking techniques. A schematic diagram of operationalization of our solution approach to data analysis using Linked Data methods is shown in figure 3.2.

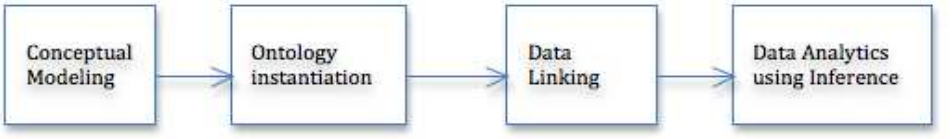

Figure 3.2 Linked Data Analytical Framework

It mainly consists of Conceptual modeling and data linking. The modeling is based on the source database. Major key tasks that are performed in our approach are:

- Conceptual modeling of database schema - Ontology engineering.
- Ontology instantiation of database tuples.
- Data Linking using semantic mapping - Semantic Rules construction
- Ontology based data access- Reasoning over *SPARQL* query.
- *RDF* Data analysis.

3.4 Evaluating the Model

Our evaluation strategy consists of technical evaluation of the developed conceptual models and the translation engine. We have used scenario-based evaluation to validate the developed Linked data analytical model and Precision is used as a measure.

3.5 Concluding Remarks

We have presented the steps in our research methodology to address the problem statement along with the definition of important terms used. We have highlighted

reasons for selecting the methods and tools used in our solution approach. Upcoming chapters will have further discussion by giving an in depth account of the implementation details.

CHAPTER 4 SEMANTIC WEB MODEL FOR PATHOLOGY LAB DATA ANALYSIS

This chapter provides a detailed account of the methods used to develop a Semantic Web model for pathology lab data representation and analysis. Two key tasks were pursed in this regard:

1. Transformation of a relational database (model) to RDF.
2. Determination of disease for a lab order using Linked Data methods.

4.1 Relational Database (Source)

This section describes the dataset that is being used to construct the ontology. The dataset is a real life health data of pathology lab test requisition orders by primary care physicians within *Nova Scotia Health Authority (NSHA)*. The provincial pathology laboratory conducts on average 15 million laboratory tests on 150,000 Patients. Our dataset comprises data over five-year period. The data is stored in *MySQL relational database*.

4.1.1 Pathology Test

Pathology test deals with the laboratory study of bodily fluid such as *blood*, *urine* and *tissue* specimens to examine and diagnose disease (pathology, n.d). Usually, laboratories process samples to provide results regarding blood counts, urine electrolytes or blood clotting ability.

4.1.2 Significance

Laboratory testing acts as an essential component of modern health care as it is critical for assessing medical conditions of patients. These tests are usually the least expensive element of the health care systems, yet they persuade more than 70 percent when it

comes to health care decisions (NCOPP, 2016). The knowledge derived from these tests provides insights into how to inhibit, diagnose, treat and control the disease.

4.1.3 Database Schema

The database consists of 5 tables in total, such as *Patient, Physician, Encounter, Order* and *Tests*. The *Encounter* table contains details coupling physician and patient with the encounter id, including the specific date and timestamp of the encounter. The order related information, like the encounter (*encounter_id*) that placed the particular order and also the date and status of the order, are all stored in the *Order* table. It will also record test fails in an order as abnormalities. All the test details, like the normal and actual result values are recorded in the *Tests* table. The *Patient* table captures the basic patient information like gender, DOB etc. Similarly, the physician details are stored in the *Physician* table. Figure 4.1 illustrates the database schema of pathology lab data.

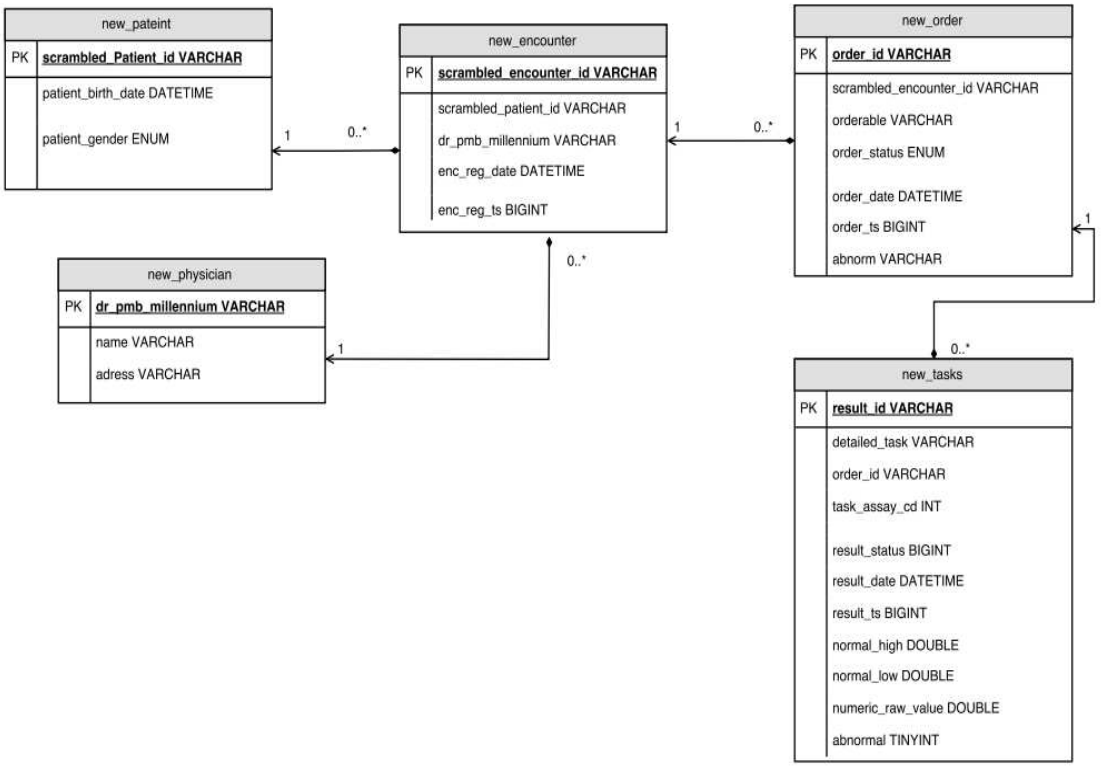

Figure 4.1 Pathology lab data - Database schema

4.1.4 Work Flow

The base unit of study is a *test*, or a single lab value. An *order* is comprised of one or more *tests*. *Physicians* select 1 or more *orders* for a *patient* from a list of 15 tests to create an *encounter*. *Physicians* can have multiple *encounters* with the *patients* and place orders for each *encounter*. Figure 4.2 illustrates the workflow of pathology lab data.

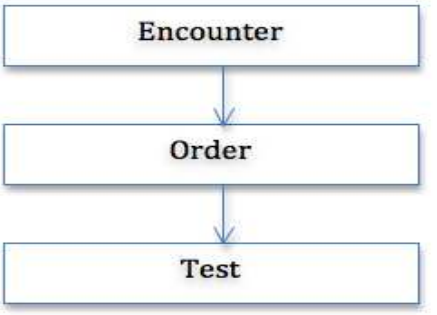

Figure 4.2 Pathology lab data – Work flow

4.2 Transformation of RDB to RDF

To obtain semantic information from a relational database, deduce inferences and acquire valuable information by data linking, we need to represent the data in RDF. Information representation of RDB in RDF (machine accessible format) generates the semantic web content for data publishing and linking. The main motivation of RDB to RDF conversion is to capture data semantics to achieve data integration and reasoning. The major challenge in this process is to decide how to represent the database schema in terms of RDF class and properties. This can be addressed by using mapping rules, which form basis for RDF triple generation. The transformation of RDB to RDF includes two major steps,

(i) Converting the Relational Database schema into ontology (Conceptual Modeling).

(ii) Converting the Relational Database instances into RDF (Ontology instantiation).

4.2.1 Conceptual Modeling to Develop a Pathology Data Ontology

Conceptual Modeling provides an abstraction of the domain in terms of a formal representation, such as Ontology. Conceptual modeling helps to understand the terminologies in the underlying database and also to capture the knowledge from the data. The motivation behind conceptual modeling is to represent knowledge in a sharable and re-usable manner and to reduce the data volume overhead by encoding the structure of particular domain. This also enables access to the relational database contents from the developed conceptual model (ontology). But developing a conceptual model from scratch is challenging process and so using the schema information can reduce the complexity of the process.

In order to represent the pathology lab data in terms of a semantic model, we first needed to develop an ontological model of the pathology lab data. The pathology lab data ontology captures the pathology lab metadata in terms of semantically defined concepts and relationships. The ontology engineering process involved two steps: (1) obtain the metadata of the RDB and transform it into ontology model (TBOX). The conversion process takes into account the conversion of the RDB sematic values presented by integrity constraints, primary keys, and foreign keys, and (2) Formulating Schema mapping rules to map extracted metadata to ontological structures – i.e concepts and relationships. Figure 4.3 illustrates the conceptual modeling framework.

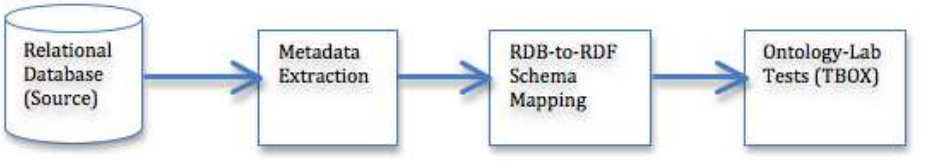

Figure 4.3 Conceptual Modeling framework

To develop the pathology lab data ontology, we considered the following three requirements: (i) Use of *DL* (Description Logic) formalism for ontology model

specification. (ii) Ontology model should be complete in terms of the vocabularies of data sources to be confederated. This means that the ontology model should comprise a union of all terms contained in all vocabularies referred by the sources being federated and (iii) Ontology model should not contain expressions that contradict with each other or with the logical consequences that can be inferred out of expressions it contains (K. Siegemund, 2011). These requirements are formulated for the ontology *TBOX* only, but were not applied to the ABOX because of two reasons:

i. The *ABOX* changes dynamically; these changes result from real time processes and can hardly be predicted or restricted.
ii. Real data as a rule is both incomplete and inconsistent. Therefore the formulation of requirements listed above would crucially reduce the number of sources that can potentially be integrated.

4.2.1.1 Metadata Extraction

Schema (metadata) extraction helps in automating the ontology generation process. This includes mining and obtaining database components like primary keys, foreign keys, triggers, events etc., that helps to obtain knowledge with high semantic power and more expressiveness (Mona Dadjoo, 2015). The extraction of metadata is also important from the ontology development perspective. The approaches involved in metadata extraction are: (i) By directly querying through the schema tables of relational database using SQL. This approach allows us to access and extract all the metadata of the relational database. (ii) By using JDBC to access and extract the metadata of a relational database. This can be achieved through a single interface like *DatabaseMetaData*. But using this method, only the metadata of the relational database that are provided through JDBC can be accessed and extracted. In this work, we have used SQL method since it supports extraction of all the metadata from the relational database.

The Pathology lab data stored in MySQL database allows a range of SQL methods to extract database metadata such as,

- Using SHOW statements.

- Using Command line programs.
- Using tables in INFORMATION_SCHEMA database.

We first considered SHOW statements to fetch the metadata using metadata extraction statements such as *SHOW DATABASE, SHOW TABLE, SHOW COLUMNS, SHOW INDEX* and *SHOW TABLE STATUS*. This metadata that extracted was quite limited and was helpful only to track the database contents in terms of a set of columns and did not give the tables. Therefore, we decided not to use this method.

Next, we investigated the MYSQLSHOW command but the results were no different then what information we got from the SHOW statements.

Finally, we pursued metadata extraction by using tables in *INFORMATION_SCHEMA* database. This approach used SELECT statements to access *INFORMATION_SCHEMA*, and we were able to name specific output columns, select information using any specific expression and save the retrieved results in another table. The following information tables were extracted from the *INFORMATION SCHEMA:* CHARACTER*_SETS, COLLATIONS, COLLATION_CHARACTER_SET_APPLICABILITY, COLUMNS, COLUMN_PREVILEGES, ENGINES, EVENTS, FILES, GLOBAL_STATUS, GLOBAL_VARIABLES, KEY_COLUMN_USAGE, PARAMETERS, PARTITIONS, PROFILING, REFERENTIAL_CONSTRAINTS, SCHEMATA, TABLES, TABLE_CONSTRAINT, TRIGGERS and VIEWS*. The table 4.1 shows the database tables along with the information that are being used for schema extraction.

INFORMATION_SCHEMA.Tables	Information about the tables in databases.
INFORMATION_SCHEMA.Key_column_usage	Information such as Constraint_name, table_schema, table_name, column_name, referenced_table_schema,

	referenced_table_name, referenced_column_name.
INFORMATION_SCHEMA.Columns	Information about the columns such as column_name, table_name, is_nullable, data_type, column_key, column_type.

Table 4.1 List of Information schema tables

The figure 4.4 shows sample queries for metadata extraction using INFORMATION_SCHEMA.

```
SELECT  * FROM INFORMATION_SCHEMA.table_name
```

```
SELECT COUNT (*) FROM INFORMATION_SCHEMA.TABLES
```

Figure 4.4 SQL queries - metadata extraction

4.2.1.2 Schema Mapping Rules

The extracted *RDMS* schema information needs to be mapped to OWL in order to develop the pathology lab data ontology. Schema mapping rules need to be formulated to map metadata to ontological structures – i.e. concepts and relationships. RDB to RDF schema mapping enables the creation of conceptual model of the RDB data at the RDB schema level. Further, it also allows accessing the underlying RDB data from the created semantic/conceptual model perspective. Tim Berners-Lee has proposed the basic mapping principles for mapping using RDB schema (Berners-Lee T. , 2006):

1. A record is an RDF node.

2. The field (column) name is RDF property type.

3. The record field (table cell) is a value.

It is evident from the above principles that the Semantic Web data model is directly connected to the schema of the RDB. In order to make RDB more easily available to the

Sematic Web, we need reliable strategies to map data from RDB to the RDF format. The conversion was done using mapping rules. It works by representing relational database components such as tables, columns and constraints in equivalent owl semantics whenever the mapping rule gets satisfied. The following mapping rules were derived.

Mapping Tables

- ***Tables are mapped to an OWL class.***

The Tables in the pathology lab database schema (fig 4.1) were mapped to OWL classes as shown in table 4.2. Each ontology class represents corresponding table in the database. Whilst mapping the tables to classes, we renamed the tables to more meaningful class labels. But here, the classes are represented with more meaningful names than the database table names. For instance, table *New_tasks* in relational database contains information related to various pathology tests and its results. Hence, we renamed it as *Tests*, as it was more meaningful compared to *New_tasks*.

Table	**Class**
New_patient	Patient
New_encounter	Encounter
New_order	Order
New_Physician	Physician
New_Tasks	Test

Table 4.2 List of classes in the ontology

- ***For a Table, if there exists a column that acts as both primary and foreign key, then the table is mapped to RDFS subclass.***

It is obvious from pathology lab database schema (fig 4.1) that there exists no column, which acts as both primary key and foreign key. Hence our pathology lab data ontology does not have any subclasses in it. In future, if the database is updated with column

denoting both the primary and foreign key, the ontology will be updated with the subclass.

Mapping Columns

- ***For all tables, map each column to data type properties, whose name is the column name, and the class that represents table is its domain and the type of the field is its range.***

Usually, conceptual modeling works by the principle of using single data type property to represent similar columns from different tables. For example, if there exists *'id'* column (*encounter_id, result_id, physician_id*, etc.) in different tables, one data type property is used to represent the *'id'* for all tables. But, we did not adhere to this principle. Rather, different data types were created for different *'id'* instead of using a common property, because in our dataset all these ids represent different levels. In addition, the data type properties are represented by more meaningful names but not by the database column names. Each data type property should have a domain and range. And if the same column appears in two tables, then that particular data type property would have those two tables as their domains.

DB Columns	**Data Property**	**Domain**	**Range**
Abnorm	Abnormal_orders	Order	String
Abnormal	Abnormal_test	Test	Integer
Detailed_task	Detailed_test	Test	String
Dr_pmb_millennium	Dr_pmb_millennium	Physician, Encounter	String
Enc_reg_date	Enc_reg_date	Encounter	Date Time
Ence_reg_ts	Enc_reg_ts	Encounter	Integer
Scrambled_encounter_id	Encounter_id	Encounter, Order	String
Order_date	Order_date	Order	Date Time

Order_id	Order_id	Test, Order	String
Order_status	Order_status	Order	{Completed, Inprocess}
Order_ts	Order_ts	Order	Integer
Orderable	Orderable	Order	String
Patient_birth_date	Patient_birth_date	Patient	Date Time
Patient_gender	Patient_gender	Patient	{Female, Male}
Scrambled_patient_id	Patient_id	Encounter	Integer
Adress	Physician_adress	Physician	String
Name	Physician_name	Physician	String
Task_assay_id	Test_assay_cd	Test	Integer
Normal_high	Test_normal_high	Test	Double
Normal_low	Test_normal_low	Test	Double
Numeric_raw_value	Test_numeric_rawvalue	Test	Double
Result_date	Test_result_date	Test	Date Time
Result_id	Test_result_id	Test	String
Result_status	Test_result_status	Test	{Autoverified, Corrected, verified}
Result_ts	Test_result_ts	Test	Integer

Table 4.3 DataTypeProperties list in the pathology lab data ontology

The domain of the property denotes the set of all things to which that property applies. Example of domain representation in rdfs is shown below,

:physician_name rdfs:domain :physician

The range of the property denotes the set of values that the property can accept. Example of range representation in rdfs is shown below,

:scrambled_patient_id rdfs:range xsd:int

Range of data type property usually signifies the data type. The underlying relational database and the ontology has their own data type representation. Hence, there is a challenge for data type mapping between them. For instance, the attribute *scrambled_patient_id* of patient table is represented by *varchar* value in the database, but in the OWL 2 QL, it is represented by *xsd:string*. The table 4.4 shows the type mapping between the logical data model and OWL.

Logical data model data types	***OWL data types***
Varchar	Xsd: string
Double	Xsd: double
Datetime	Xsd: datetime
Integer	Xsd: integer
Tinyint	Xsd: integer
Bigint	Xsd: integer

Table 4.4 Logical data model to OWL data type mapping list

Enumerations are also used to denote the range of data type property. Table 4.3 illustrates the data type properties such as *Test_result_status*, *Order_status* and *Patient_gender* that have enumerations as their ranges. There are two ways to represent enumerations in the ontology. One method is to use class – subclass constructs to represent these enumerations and other method involves by restricting the data type property to specific values. We have used data type property restriction to enumerate, as we did not want the data types represented as sub classes in the ontology. Data type restriction is done using *OWL:OneOf*. Figure 4.5 shows the structure of *test_result_status* attribute enumeration using data type restriction from pathology lab data ontology.

```
DataPropertyRange
(
 :test_result_status DataOneOf
      (
      “AutoVerified”^^xsd:string
      “Corrected”^^xsd:string
      “Verified”^^xsd:string
      )
)
```

Figure 4.5 Enumeration in OWL Functional Syntax

Figure 4.6 shows the screenshot of resulting data type properties obtained by applying the above mapping rule.

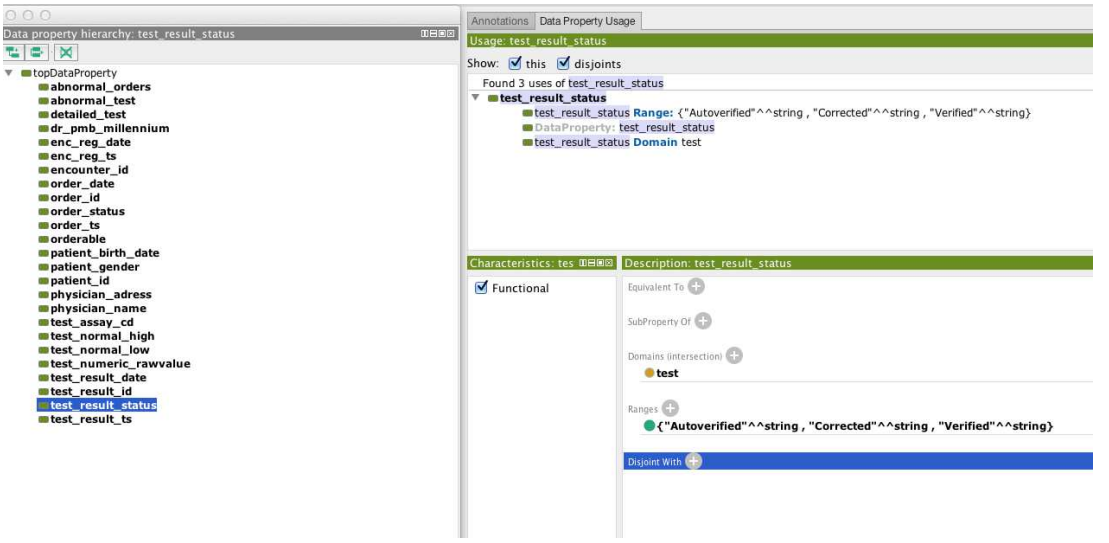

Figure 4.6 Pathology data- Data Type properties

Mapping Unique Constraints

- ***Column with unique constraint is mapped to a functional property***

Unique constraint indicates that no two data in the table have the same value for the column. Hence it is more appropriate to map to functional property as it ensures

atomicity of the attributes. Pathology lab test data has unique constraints only for the primary key columns. Hence, the data type properties such as *patient_id, encounter_id, order_id, dr_pmb_millennium,* and *result_id* are mapped to functional property.

Mapping Not Null Constraints

- ***Columns with not null constraint are mapped to a minimum OWL cardinality of one.***

This constraint specifies that a column in a table is not null, meaning that all data in the table contains values for the column. In our dataset, only the primary key columns are set as not null. Hence only the data type properties such as *patient_id, encounter_id, order_id, dr_pmb_millennium* and *result_id* are set to minimum cardinality of one.

Mapping Primary Key Constraints

- ***Columns with primary key constraint are mapped to OWL Inverse Functional properties. The constraint also maps to maximum OWL cardinality of one.***

The value of primary key uniquely determines a single row of table; hence it is set to minimum cardinality of one, as it should have at least one entry. Properties such as *patient_id, encounter_id, order_id, dr_pmb_millennium* and *result_id* are set to minimum cardinality of one.

Mapping Foreign Key Constraints

- ***Foreign key constraints are mapped to an object property with an OWL class corresponding to T1 as Domain class and an OWL class corresponding to T2 as Range class.***

A foreign key establishes a relationship between the class that represents the table T1 of field to the class that represents the table T2 referenced by the foreign key. Figure 4.7 shows foreign keys of tables corresponding to pathology lab test data.

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
new_encounter	scrambled_encounter_id	PRIMARY	NULL	NULL
new_encounter	scrambled_patient_id	new_encounter_ibfk_1	new_patient	scrambled_patient_id
new_encounter	dr_pmb_millennium	new_encounter_ibfk_2	new_physician	dr_pmb_millennium
new_order	order_id	PRIMARY	NULL	NULL
new_order	scrambled_encounter_id	new_order_ibfk_1	new_encounter	scrambled_encounter_id
new_patient	scrambled_patient_id	PRIMARY	NULL	NULL
new_physician	dr_pmb_millennium	PRIMARY	NULL	NULL
new_tasks	result_id	PRIMARY	NULL	NULL
new_tasks	order_id	new_tasks_ibfk_1	new_order	order_id

Figure 4.7 Foreign key column usage of pathology lab data in RDMS

Initially, the object properties were created with the naming convention *"has_ + Referenced_column_name*", which resulted in object properties like *has_scrambled_patient_id, has_dr_pmb_millennium, has_scrambled_encounter_id* and *has_order_id.* Later, the names were changed by analyzing reference table names, referenced column name and actual table name to make it more meaningful and understandable as it establishes the relations between the instances of the classes in the ontology. Object type property obtained from pathology lab test data along with its corresponding domain and range is listed in table 4.5.

Object Property	***Domain***	***Range***
HasEncounter_By	Physician	Encounter
HasEncounter_For	Patient	Encounter
HasOrders	Encounter	Order
HasTest	Order	Test

Table 4.5 Object type properties list in the pathology lab data ontology

The object properties are inverse functional, which means it can take more than one value. Like data type property, the object type property should also have domain and range associated with it. The domain representation for an object type property in rdfs for pathology lab data is specified as "*:HasEncounter_By rdfs:domain :Physician"* and the range is specified as "*:HasEncounter_By rdfs:range :Encounter*."

Figure 4.8 shows the screenshot of resulting object type properties obtained by applying above mapping rules using protégé ontology editor.

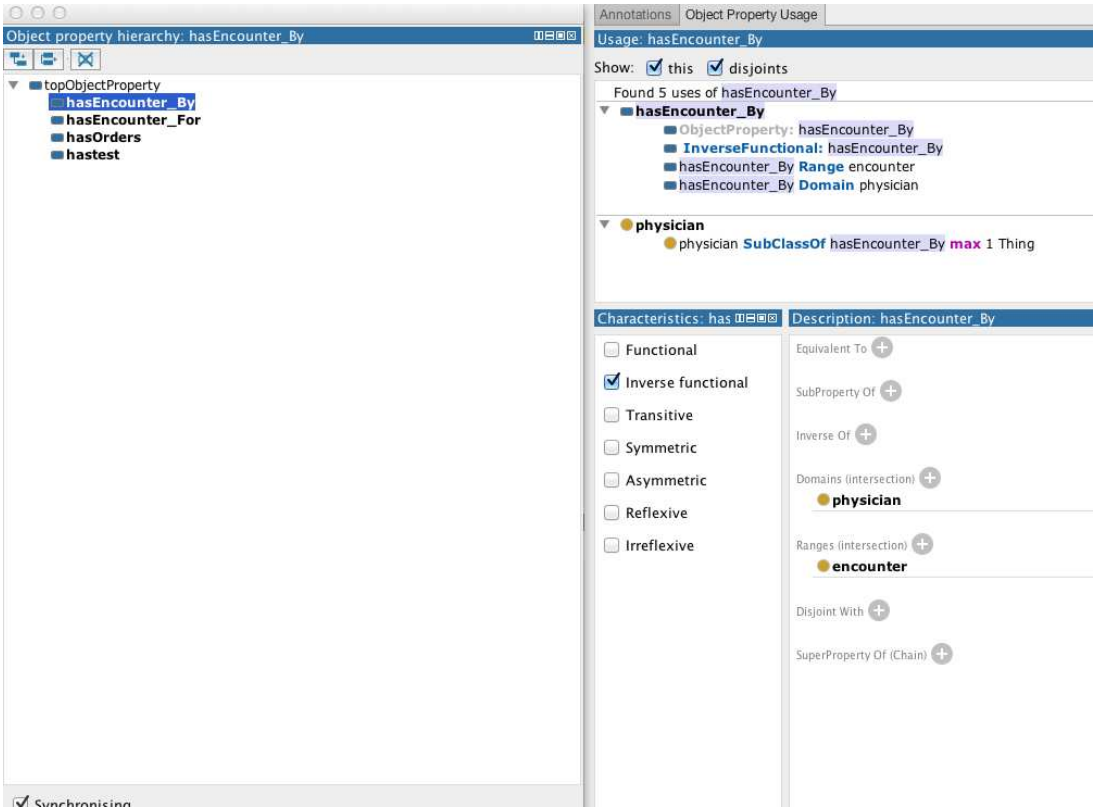

Figure 4.8 Pathology lab test data- Object Type properties

The figure 4.9 illustrates the final pathology data ontology obtained from applying all the schema-mapping rules mentioned in 4.2.1.2.

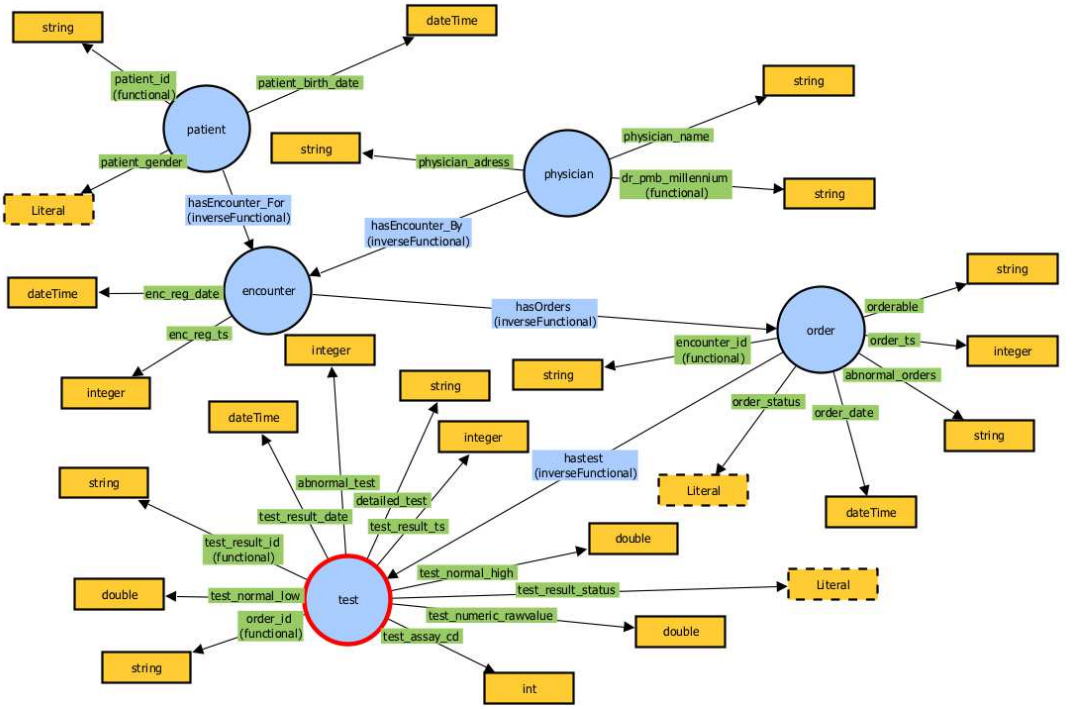

Figure 4.9 Pathology lab data ontology

4.2.2 Ontology Instantiation

Once an ontology is defined on the underlying RDB schema, it needs to be populated to create a knowledge base. This process is defined as ontology instantiation and can be performed using Ontology Based Data Access. In our work, the ontology instantiation was done using *dynamic materialization method* after extensive review described in section 2.7.2. Dynamic materialization involves on-the-fly conversion of RDB instances into RDF format with the help of mapping axioms. The ontology is instantiated by individuals (*ABOX*) using the data from *RDB* and the model of the ontology. We have used *Ontop Quest* tool to dynamically materialize the RDF triples. The *OBDA* model consists of two major tasks, data source definition and mapping axioms.

4.2.2.1 Data Source Definition

The connection between the ontology and the underlying database is established using the data source definition. It defines the data source that is used. The data source

definition provides the information that a system requires to access the data source. *Quest* and *Ontop* use JDBC connections to connect to data sources and so they require JDBC parameters. The connection parameters such as connection URL, the Driver class, the database name, and the database password needs to be assigned. Connection URL is a URI that determines the type, location and name of the databases. Additionally, it is possible to pass connection parameters to the JDBC driver by means of JDBC URL. The driver class is a string that indicates which JDBC driver to use when establishing a JDBC connection. JDBC drivers are software implemented by third parties that handle interaction with the DB in their own proprietary protocols. For MySQL, we use the following driver class,

Database	Driver name	Version	Class
MySQL	Connector/J	5.1.35	Com.mysql.jdbc.Driver

Using the OBDA model feature in protégé, we defined the connection parameters (Figure 4.10) to the pathology lab test database and established connection to the data source.

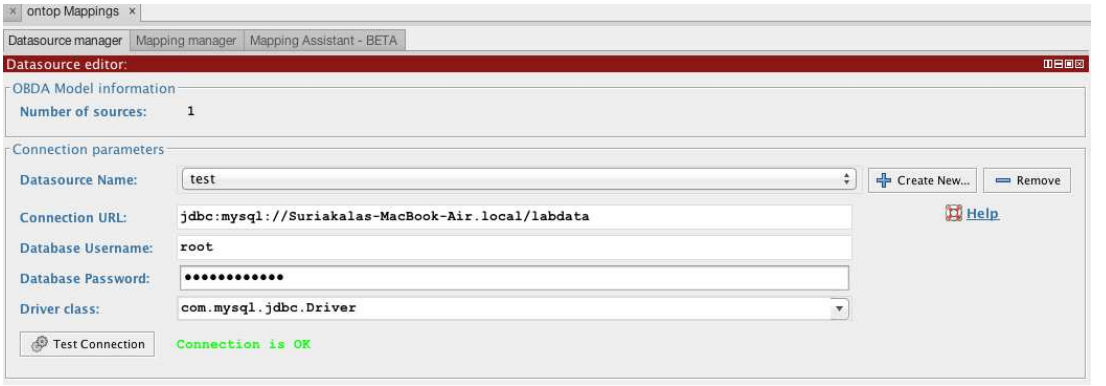

Figure 4.10 Setting up data source in Ontop

4.2.2.2 Creation of Mapping Axioms

Mapping axioms specify the relationship between the data in the data source and the vocabulary of OWL ontology. The mapping rules represent the manner the database

tuples are outputted as semantic data. The data resides in the database and it is linked to the ontology through mappings. The mappings are created using the mapping manager in Ontop mappings and by providing the appropriate source and assertion template. Mapping rules need to be defined as per the Ontop mapping language to access the database using *SPARQL*. General rules to create assertion template for Ontop mapping are explained in detail:

i. A base *URI* is chosen arbitrarily and used as default prefix. For example:
   @prefix: <http://www.example.org#>.

ii. The base *URI* is concatenated with entity name to create an *URI* template for each entity. For example:
   < http://www.example.org#encounter_id/{scrambled_encounter_id}>.

iii. Create one rule per entity. The rule consists of three parts.
   - *Rdf:type* to map the entity to the class.
   - *Owl:datatypeProperty* to map the attributes.
   - *Owl:objectProperty* to map the relations.

Mapping rules creation using *Ontop quest* involves the following steps:
(i) First step is to load the table names from data sources into the mapping assistant so that the standard SQL query can be automatically generated with the table entries. This facilitates the column selection to create the subject or object in assertion triple template. This also helps in checking the correctness of the data linking by reviewing the data fetched by the SQL query.
(ii) Next step involves linking the database columns to the ontology descriptions (classes and properties) to facilitate ontology based data access.

Class Linking:

Class linking is done by selecting the primary key column of the table as subject and mapping it to equivalent ontology class. It involves populating '*Focus on URI'* field with

primary key column i.e. '*scrambled_encounter_id'* in the data preview table and then selecting the class '*encounter*' from the '*Mapping for class'* field. Figure 4.11 shows the class linking using Ontop mappings.

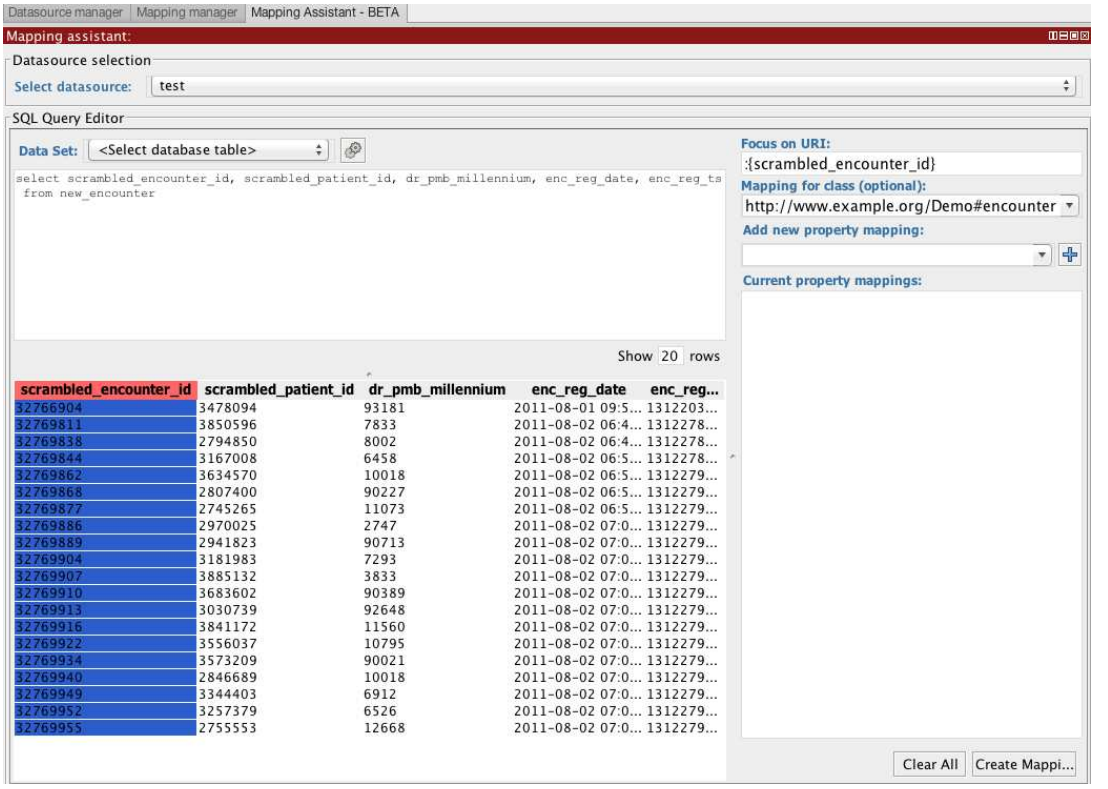

Figure 4.11 Class Linking using Ontop

This will result in the following mapping,

```
MAPID-67b657e86b8948a0b8ad4e1158edcfa6
:{scrambled_encounter_id} a <http://www.example.org/Demo#encounter> .
select scrambled_encounter_id, scrambled_patient_id, dr_pmb_millennium, enc_reg_date, enc_reg_ts from new_encounter
```

It can be seen that the mappings are created in triple format. The above mapping represent that the subject '*scrambled_encounter_id'* value belongs to the object class '*encounter*'. Predicate in this mapping is "*rdf:type"* aka "a", which is implicitly created when the class selection is made.

Data Type Property Linking:

Data type property linking is done by selecting the primary key column of the table as subject, data type property from ontology as predicate and corresponding table column as object. It involves populating '*Focus on URI'* field with primary key column i.e. '*scrambled_encounter_id'* in the data preview table and then selecting the data type property that needs to be mapped i.e. '*enc_reg_data*' from the '*Add new property mapping'* field as predicate and finally populating '*Current property mappings'* with '*enc_reg_date'* column. In case if we need to add the data type for the attribute, it can be selected in the data type selector. This will override the type of data provided both by the database schema and ontology. Figure 4.12 shows the data type property linking using Ontop.

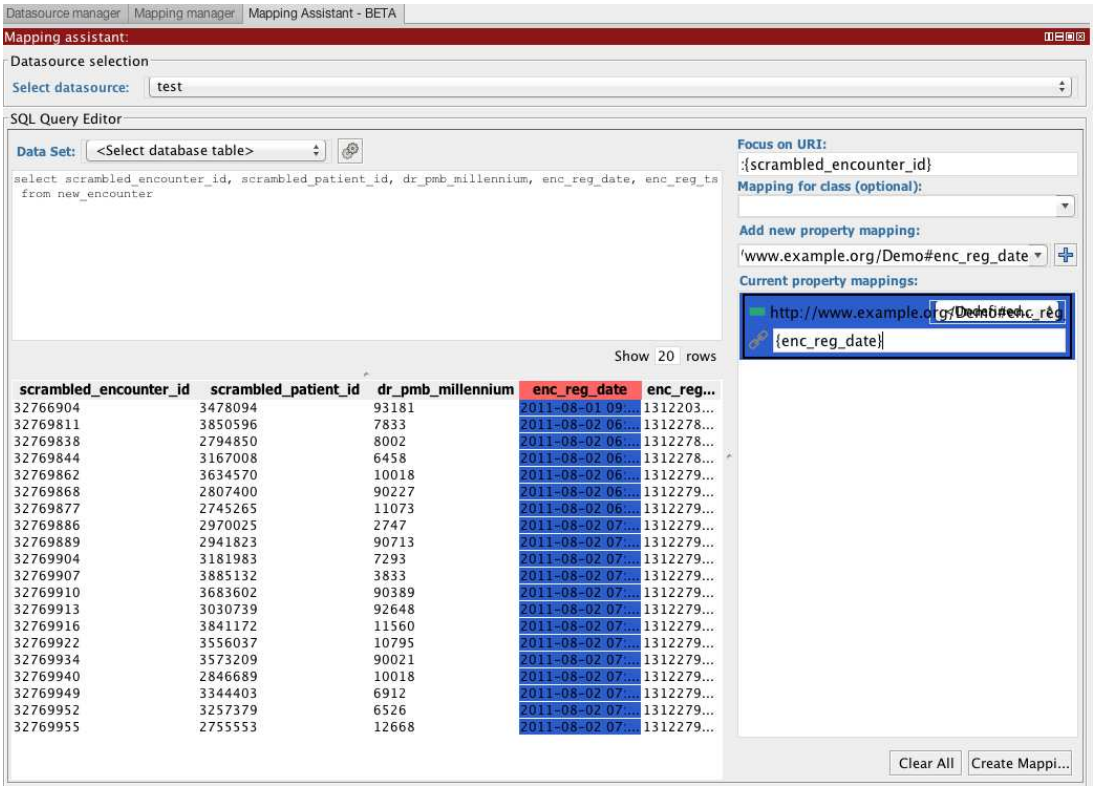

Figure 4.12 Data type property linking using Ontop

The linking will result in following mapping,

```
MAPID-7f8b231203724778b49138653ec41483
:{scrambled_encounter_id} <http://www.example.org/Demo#enc_reg_date> {enc_reg_date}.
select scrambled_encounter_id, scrambled_patient_id, dr_pmb_millennium, enc_reg_date, enc_reg_ts from new_encounter
```

The above mapping represent that the subject 'scrambled_encounter_id' value has an object with 'enc_reg_date' value which is linked using predicate 'demo#enc_reg_date'.

Object Type Property Linking:

Linking object type property is similar to the data property linking. The only difference is that the object type property is chosen as predicate. Figure 4.13 shows object type property linking using Ontop.

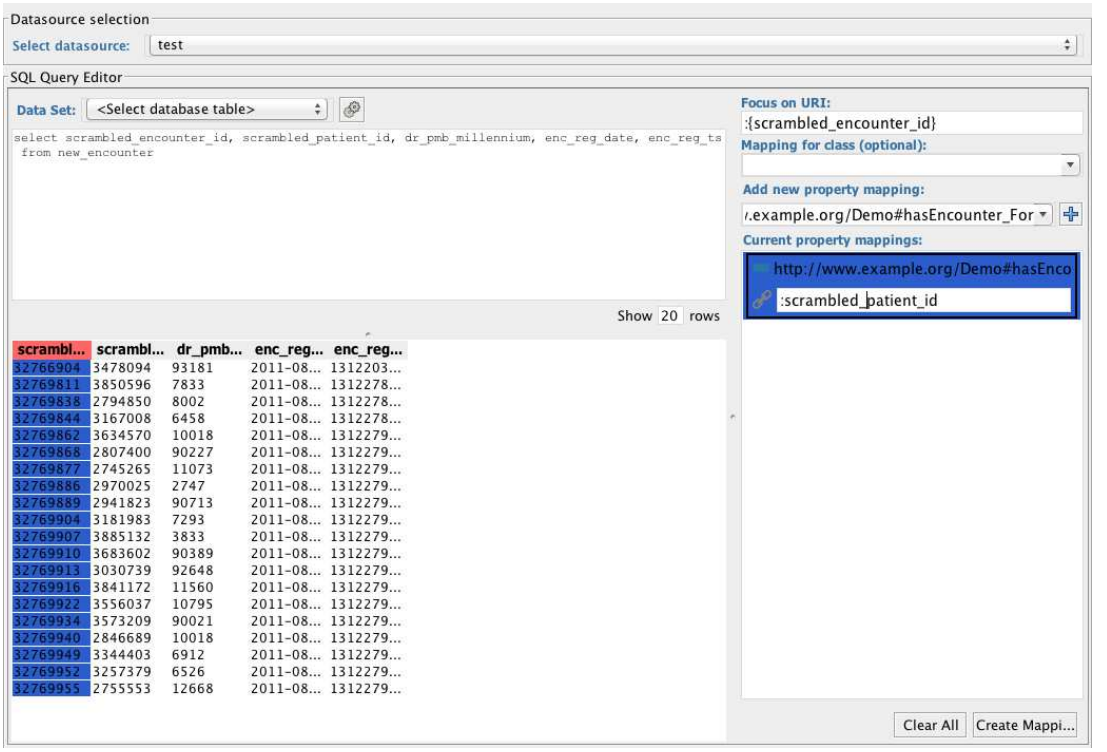

Figure 4.13 Object type property linking using Ontop

The above will result in following mapping,

```
MAPID-1115d34162cb40e9a3d9d3f67d434473
  :{scrambled_encounter_id} <http://www.example.org/Demo#hasEncounter_For> :scrambled_patient_id .
  select scrambled_encounter_id, scrambled_patient_id, dr_pmb_millennium, enc_reg_date, enc_reg_ts from new_encounter
```

Mapping indicates that the subject '*scrambled_encounter_id'* value has an object '*scrambled_patient_id'* value, which is connected by '*has_encounter_For'* predicate.

Using the above steps, mapping rules for all the class present in the pathology data is created as shown in figure 4.14.

```
Encounter
  :encounter_id/{scrambled_encounter_id} a <http://www.example.org/Demo#encounter> ;
  <http://www.example.org/Demo#hasEncounter_For> :patient_id/{scrambled_patient_id} ;
  <http://www.example.org/Demo#hasEncounter_By> :physician_id/{dr_pmb_millennium} .
  select scrambled_encounter_id, scrambled_patient_id, dr_pmb_millennium  from new_encounter limit 100
```

```
Order
  :orderid/{order_id} a <http://www.example.org/Demo#order> ; <http://www.example.org/Demo#hasOrders>
  :encounter_id/{scrambled_encounter_id} ; <http://www.example.org/Demo#orderable> {orderable} ;
  <http://www.example.org/Demo#order_status> {order_status} ; <http://www.example.org/Demo#abnormal_orders> {abnorm} ;
  <http://www.example.org/Demo#order_date> {order_date} .
  select order_id, scrambled_encounter_id, orderable, order_status, order_date, order_ts, abnorm from new_order limit 10
```

```
Patient
  :patient_id/{scrambled_patient_id} a <http://www.example.org/Demo#patient> ; <http://www.example.org/Demo#patient_gender>
  {patient_gender} .
  select scrambled_patient_id, patient_birth_date, patient_gender from new_patient limit 10
```

```
Physician
  :{dr_pmb_millennium} a <http://www.example.org/Demo#physician> ; <http://www.example.org/Demo#physician_name> {name} .
  select dr_pmb_millennium, name, adress from new_physician limit 100
```

```
Tests
  :{result_id} a <http://www.example.org/Demo#test> ; <http://www.example.org/Demo#test_numeric_rawvalue> {numeric_raw_value} ;
  <http://www.example.org/Demo#test_normal_high> {normal_high} ; <http://www.example.org/Demo#detailed_test> {detailed_task} ;
  <http://www.example.org/Demo#order_id> :orderid/{order_id} .
  select  result_id, normal_high, numeric_raw_value, detailed_task, order_id from new_tasks limit 100
```

Figure 4.14 Pathology lab data - Mapping axioms

As the data is accessed directly from database using *OBDA*, it is efficient to create the mapping rules based on the primary keys as they are indexed by default, which improves data retrieval. It can be noticed from above mapping rules that only primary key attribute is used as the subject of the triple in assertion template.

4.2.3 RDF Data Retrieval

RDF data is retrieved using Ontology bases data access mechanism. In OBDA system, the data resides in the source and the *SPARQL* query is written on top of ontology model developed which is RDF representation of DB schema. The mapping rules specify relation between the database and the ontology model. To establish a connection between these two, we need to translate the SPARQL queries to SQL. This translation of SPARQL to SQL is performed using *Ontop Quest* reasoner. With the help of ontology, data sources and the mappings, the reasoner processes the request. It translates the *SPARQL* query along with the mappings into Datalog program. Database metadata is used to optimize the Datalog program, which is then converted into relational algebra and then finally into SQL queries. The generated *SQL* queries are executed against the *Relational DB*. The SQL query results are then extracted in the target triples format (*RDF triples*) specified in the mapping rules. Instead of converting the entire data stored in underlying database *into RDF*, we use *virtual RDF graphs* (D. Calvanese, 2015). Figure 4.15 illustrates the ontology based data access using Ontop Quest reasoner.

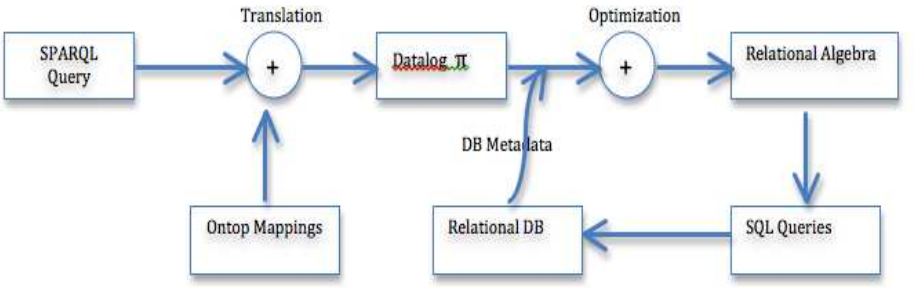

Figure 4.15 Ontology based data access using reasoner

From the pathology lab dataset, we retrieved the following information using the developed pathology lab data ontology,

- **Details of patients under each physician**

Consider the scenario where we have to fetch the details of the patients who visited a particular physician (say physician id #93181). The patient details are present in the table '*New_patient'* and the physician details are stored in '*New_Physician'* table, these data that links these two entities are present in the '*New_encounter'* table. In the pathology data ontology we have created, these tables are represented using the classes '*patient', 'physician'* and '*encounter'* respectively. So, if we have to fetch the patient details under each physician, the SQL query has to join all the three tables to retrieve the data. But with our developed pathology data ontology the query can as simple as shown in figure 4.16, which can simply ask for all the patients details for a given physician id. This is achieved by the object property mapping axioms we have created. Since we do not have access to the patient details except for the patient id, the SPARQL query we retrieve has only the patient ids under a given physician.

Figure 4.16 SPARQL query to retrieve patient details under a physician

The SPARQL query in Figure 4.16 consists of two triple patterns. The pathology data ontology we have created has object properties that link the patient and physician classes via the encounter class. To fetch the *patient_id*, we need two triple patterns, one for the *patient_id* linking the *encounter_id* and other for the *physician_id* linking the *encounter_id*. Further, the triples in the SPARQL should match the triples in the mapping rules. Otherwise the query will simply returns null set. This is because the triples in the mapping rules define how the fetched data is outputted. The 1st triple pattern *'?encounter_id <http://www.example.org/Demo#hasEncounter_For> ?patient_id'* selects the *patient_id* corresponding to the *encounter_id*. The next triple narrow downs the encounter_id selected based on the physician_id #93181.

- **Order details for particular encounter**

The next scenario is to fetch all the order details for a given encounter. This is slightly different from the previous scenario, where we had to fetch data from two different classes. Here, all the details reside in the same table *new_order.* In our pathology data ontology *order* class represents this table. The SPARQL query in Figure 4.17 consists of four triple patterns, for fetching four different attribute data from the *new_order* table. For the *order* class, we have created four data type properties and an object property for the *encounter_id* in order to represent the pathology data hierarchy.

```
Query Editor
select ?order_id ?orderable ?order_status ?order_date
where
{
?order_id <http://www.example.org/Demo#hasOrders> <http://www.example.org#encounter_id/32770015> ;
<http://www.example.org/Demo#orderable> ?orderable  ; <http://www.example.org/Demo#order_status> ?order_status ;
<http://www.example.org/Demo#abnormal_orders> ?orderabnorm ;
<http://www.example.org/Demo#order_date> ?order_date  .
}
```

order_id	orderable	order_status	order_date
<http://www.example.org#orderid/170969911>	"TSH"^^xsd:string	"Completed"	"2011-08-02T07:09:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969914>	"Creatinine"^^xsd:string	"Completed"	"2011-08-02T07:09:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969917>	"Urea"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969920>	"CBC"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969923>	"Alkaline Phosphatase"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969926>	"GGT"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969929>	"ALT"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969932>	"AST"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime
<http://www.example.org#orderid/170969935>	"PT"^^xsd:string	"Completed"	"2011-08-02T07:10:00.0"^^xsd:dateTime

Figure 4.17 SPARQL query to retrieve order details for an encounter

The query in fig 4.17 fetches the details of *order_id, orderable, Order_status, Order_date* corresponding to *encounter_id* #32770015.

- **Tests taken in a particular order**

The last SPARQL query we have analyzed here is for retrieving all the tests taken under a particular order. Like the previous scenario, all the required data are present in the same table *New_tasks*, which is represented by the class *tests* in our pathology lab data ontology. The SPARQL query in Figure 4.18 is similar to the previous query, with four triple patterns, each one for a *tests* class attribute. For a given order_id, the data from four different fields for each test in the *New_tasks* table are fetched.

```
Query Editor
select ?result_id ?numeric_raw_value ?normal_high ?detailed_task
where {
?result_id a <http://www.example.org/Demo#test> ;
<http://www.example.org/Demo#test_numeric_rawvalue> ?numeric_raw_value ;
<http://www.example.org/Demo#test_normal_high> ?normal_high ;
<http://www.example.org/Demo#detailed_test> ?detailed_task ;
 <http://www.example.org/Demo#order_id> <http://www.example.org#orderid/170968820>.
}
```

result_id	numeric_raw_value	normal_high	detailed_task
<http://www.example.org#169995648>	"101.0"^^xsd:double	"97.0"^^xsd:double	"MCV"^^xsd:string
<http://www.example.org#169995649>	"34.8"^^xsd:double	"32.0"^^xsd:double	"MCH"^^xsd:string
<http://www.example.org#169995651>	"16.4"^^xsd:double	"14.5"^^xsd:double	"RDW"^^xsd:string
<http://www.example.org#169995654>	"75.3"^^xsd:double	"70.0"^^xsd:double	"Neutro percent Auto"^^xsd:string
<http://www.example.org#169995656>	"10.8"^^xsd:double	"10.0"^^xsd:double	"Mono percent Auto"^^xsd:string

Figure 4.18 SPARQL query to retrieve test details in an order

4.2.4 Section Summary

Representing the RDB data in RDF format helps us to add semantics to the data, making it machine understandable and allowing automatic linking with other RDF data. One of the main challenges in converting the legacy data into RDF is the volume of data. When it comes to conceptual modeling of data, most of the tools present does not support huge volumes of data and the data retrieval is also computationally expensive. Hence, we have used a technique that does schema conversion first and then performs data conversion using OBDA separately. This lets the data to reside in the database source not in the conceptual model and accessed on the fly when queries are executed. The proposed method described above is programmatically implemented and the program automatically populates the owl ontology, which includes the classes, properties, properties characteristics and cardinality using above-mentioned mapping rules. Further, the method used to convert RDB to RDF can be applied to any RDB datasets and across any domain.

4.3 Determination of Diseases Using Linked Data

The dataset just provided information about the test results for each patient encounter and did not have any information relating the test results to the diseases. Our aim was to determine the diseases from the pathology lab test results. Our approach was to infer the disease based on the ordered pathology test. This required creating a mapping between the disease and a set of pathology tests that may be ordered to diagnose and manage the disease. Our approach was to refer to established medical resources that provide a correspondence between disease and pathology tests, and then develop a linked data solution that (a) links external information sources to our Pathology data ontology and

the associated RDF triples; (b) links the ordered test to a set of corresponding diseases; and (c) determine the disease that may have mitigated the ordered test by analyzing the linked data results.

4.3.1 Examination of External Resources for Lab Tests Association with Diseases

Various government organizations have developed lab tests charts that associate lab tests with potential diseases. In this thesis, we examined the below mentioned resources to establish a semantic mapping between the pathology tests and diseases presented in Table 4.6. The mapping take into account the value of the test result and depending on the result value range the corresponding disease is assigned.

I. *American association of clinical chemistry* (*AACC*) is a global scientific and medical professional organization dedicated for clinical laboratory science and its application to healthcare. It has devised an award-winning health information web resource called as LabTestsOnline to help patients and caregivers understand the lab tests. LabTestsOnline provides the usage of multiple related lab tests and the indications of test results in one lab test web page. Disease related to lab test results is listed in the section 'what does the test result mean'. This includes details about the lab tests, conditions/diseases, screening and the lab tests news.

II. *IWMF* is another non-profit organization, which supports the innovation of knowledge in the pathophysiology. It provides particulars about the blood tests and disease related to the results of those tests.

III. *APS Healthcare* is an associate of the Universal American Family of companies, which also provides similar kind of information.

IV. *Medline Plus* provides well-organized disease related information in a semi-structured XML files. It has different sections to describe the disease such as Symptoms & Diagnosis, Treatment, etc. Lab tests are indicated by means of information box in Medical Encyclopedia section.

Name of the test	*High results mean*	*Low results mean*
Sodium	Cushing's syndrome, kidney disease.	Addison's disease, diarrhea, adrenal insufficiency.
Potassium	Acute or chronic Kidney failure, Addison's disease, diabetes.	Cushing's syndrome.
Chloride	Cushing's syndrome, kidney disease.	Emphysema, Lung disease.
CO2	Lung disease, COPD.	Diarrhea, kidney disease.
Creatinine	Kidney disease.	Low muscle mass.
Urea	Kidney disease.	Liver disease, nephritic (kidney) syndrome.
Triglycerides	Risk of heart disease.	None.
Cholesterol	Risk of heart disease.	None.
HDL	Risk of heart disease.	None.
Alkaline Phosphatase	Primary cirrhosis, Rheumatoid arthritis, myocardial infarction, liver cancer.	Scurvy, Pernicious anemia, hypophosphatemia, - malnutrition, milk-alkali syndrome.
GGT	Diabetes, Cushing's syndrome, hyperthyroidism and acute stress.	Adrenal insufficiency, hypothyroidism or insulin overdose.
ALT	Hepatitis, hepatic necrosis, cirrhosis.	No low results available.
AST	Heart disease, liver disease, skeletal muscle disease, anemia, pancreatitis, hepatitis muscle injury.	Acute renal disease, diabetic ketoacidosis, Beriberi

PT	Longer time for blood clot.	Not a concern.
WBC	Infection, leukemia, corticosteroid, Inflammation, intense exercise, stress.	Drug toxicity, bone marrow failure, viral infections, and autoimmune disease.
RBC	COPD, Congenital heart disease, polycythemia vera, pulmonary disease, renal problems, dehydration.	Folate deficiency, iron deficiency, vitaminB12 deficiency, bone marrow failure, leukemia, lymphoma, Hemolysis, Anemia, hemorrhage, cirrhosis of the liver.
Hgb	COPD, Congenital heart disease, Dehydration, polycythemia vera, pulmonary disease, renal problems.	Anemia, hemorrhage, cirrhosis of the liver, leukemia, lymphoma, red blood cell hemolysis, Folate deficiency, iron deficiency, vitamin B12 deficiency, Bone marrow failure.
Hct	COPD, Congenital heart disease, Dehydration, Polycythemia vera, pulmonary disease, renal problems.	Anemia, hemorrhage, cirrhosis, Iron, vitamin or Folate deficiency, bone marrow damage, leukemia, lymphoma, Blood loss, hemolysis.
MCV	Vitamin B12 or Folate deficiency.	Iron deficiency.
MCH		

	Vitamin B12 or Folate deficiency.	Iron deficiency.
MCHC	Sickle cell disease, hereditary spherocytosis.	Iron deficiency.
RDW	Iron deficiency, vitamin 12 or Folate deficiency and blood loss.	Not a concern.
Platelet	Leukemia, myeloproliferative disorders, inflammatory conditions, rheumatoid arthritis, anemia.	Hemorrhage, Bone marrow failure, chemotherapy, viral infection, lupus, pernicious anemia, leukemia or lymphoma, sequestration in the spleen.
MPV	Inherited disorders.	Aplastic anemia, thrombocytopenia.
Neutro absolute or percent auto	Infection, Inflammation, Leukemia, stress, corticosteroids.	Bone marrow failure, chemotherapy.
Lymph absolute or percent auto	Viral infections, leukemia, lymphoma.	Bone marrow failure, chemotherapy
Mono absolute or percent auto	Chronic infections, autoimmune disease, leukemia.	Bone marrow failure, chemotherapy.
Eos absolute or percent auto	Parasitic infections.	Not a concern.
Baso absolute or percent auto	Active allergy response.	Not a concern.
TSH	Hypothyroidism, Thyroiditis	Secondary hypothyroidism, Hyperthyroidism.
Glucose Random	Diabetes, pancreatic cancer, Cushing's syndrome.	Liver disease, Kidney disease.

Glucose AC	Diabetes.	None.
INR	Longer time for blood clot.	None.

Table 4.6 Tests and possible disease chart

The mapping rules, used by the linked data solution, to link lab test results with possible diseases were derived from Table 4.6.

4.3.1.1 Disease Ontology

In order to study the feasibility of inferring diseases from pathology tests we needed a resource that defines diseases. The *Human Disease Ontology* (DO) provided by *Bio-portal* is an open standard ontology representing a broad knowledge base of congenital, developmental and acquired disease concepts captured across biomedical resources. It was created with the aim of providing the biomedical community with consistent, reusable and sustainable descriptions of human disease terms, phenotype characteristics and related medical vocabulary disease concepts through collaborative efforts of researchers at Northwestern University, center for Genetic Medicine and the University of Maryland School of Medicine, Institute of Genome sciences. It is also integrated semantically with *SNOMED-CT*, *ICD-9, UMLS, MeSH, OMIM* and the *NCI* through disease concepts and medical vocabularies. Currently, Disease Ontology contains 9247 classes of diseases.

To map the pathology tests to diseases, we examined the diseases noted in table 4.6 and selected a subset of the disease ontology that comprises the disease noted in the table 4.6. The figure 4.19 shows the partial disease ontology needed for our analysis loaded into protégé.

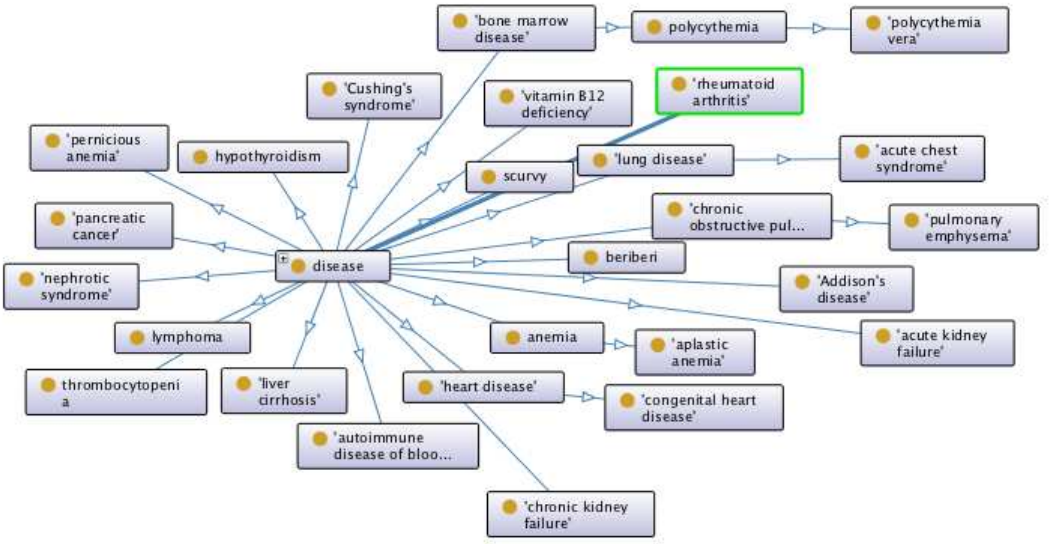

Figure 4.19 Human Disease Ontology

4.3.2 Linking Pathology Lab Data with Disease Ontology using Linked Data

One of the most significant challenges of the semantic web is to comprehend how to represent and reason with different, multiple, separately built but linked ontologies. In our case, we linked the generated pathology lab test ontology with the Human Disease ontology. All the diseases that appear in table 4.6 are imported from Human disease ontology, however in future, we can also extend the ontology with as many diseases as possible. Based on our review on efficient data linking techniques (section 2.4), we have employed a Linked Data approach.

4.3.2.1 Data Linking Framework

The data linking process is illustrated in Figure 4.20 and comprises 3 steps:

Step 1: Configuration of the system settings for data matching, including importing relevant ontologies, installation of appropriate reasoners, mapping managers and deciding on the similarity criteria for matching.

Step 2: Selection of data linking predicates. Data Linking predicates refers to the‘predicate relationships’ which is used to link the two different source, in our case the

pathology tests and the diseases. Predicate selection is crucial because these predicates indicates the meaning of the linking relation.

Step 3: Mapping of instances based on Lab tests and disease association chart using the data linking predicates. The data linking process outputs the mapping sets, which is the collection of binary relations between the objects in the input datasets. The mapping rules comprises of target triples and the *SQL* query. Linking predicate is the part of the target triples in mapping rules.

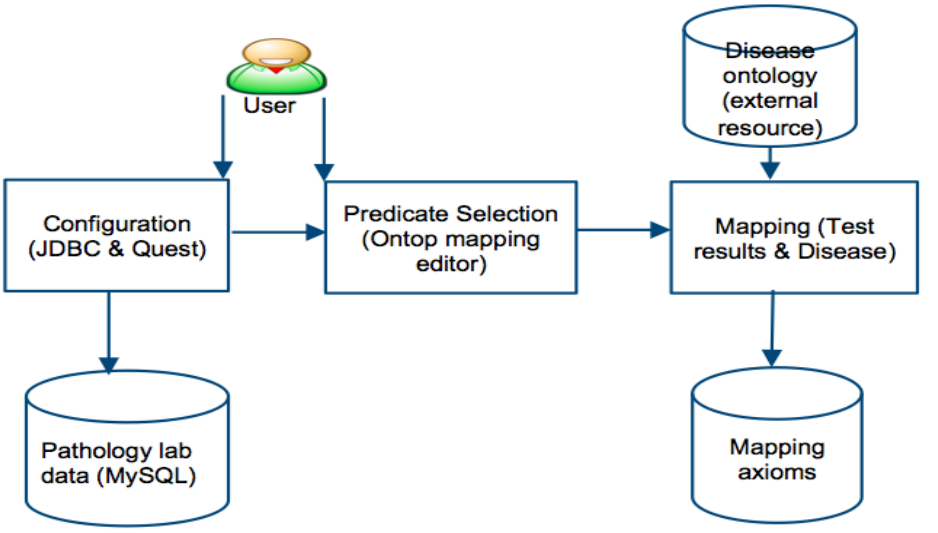

Figure 4.20 Data linking framework for pathology data and disease ontology

Configuration of the System Settings

Configuring the system i.e. *Protégé,* involves setting appropriate values for different parameters. The first step in the configuration is to load the ontology file in to *Protégé* in order to edit the vocabulary or the ontology and also to check the inconsistencies in the created ontology. Hence, the generated pathology lab data *OWL* file is first loaded into *Protégé*. The next step is setting up the ontology headers such as ontology *IRI* (International Resource Identifier) and *IRI version*, for identifying the ontology and its elements. We have used the *OBDA* approach to access the relational database. The

reason for choosing *ODBA* approach is described in section 2.7.2. For using the *OBDA* approach, we need to set up the JDBC driver parameters for relational database i.e. MySQL, because in our work the pathology lab data is stored in a MySQL relational database. To infer logical consequences from a set of asserted facts or axioms, a reasoner plugin is also need to be added to the *Protégé*. The *Ontop Quest* can act as both an OBDA system as well as a reasoning engine. Hence, we have plugged in the *Ontop Quest* into *Protégé*. The data configuration mode tells the system whether to access the data in virtual or classical mode. In virtual mode, the data is not really transformed from RDB to RDF, instead the SPARQL query is translated into SQL. Whereas, in the classical mode, the RDF dump is created and stored as triple stores. So we have selected the virtual mode in the data configuration. In addition to these basic configurations, we have to import all the relevant ontologies i.e., the Human disease ontology is imported.

Selection of Data Linking Predicate

The goal of Linked Data is to establish meaningful links between different data sources. We have established the link between the pathology lab data ontology and the Human disease ontology using the data-linking predicate. Since the goal of data linking depends on the assumptions and user goals, the predicate selection varies according to user goals. Predicates can be classes, object properties and data properties. In our case, we have used the class predicate *rdf:type*. The predicate *rdf:type* denotes that resource is an instance of a class, i.e., from figure 4.21 disease is an instance of rdfs:Class and *result_id* is an instance of disease. We could not create any object property, linking the pathology data ontology and the Human disease ontology. This is because, when an object property is used as a predicate, we need to select an object from the underlying database. But, in our case, the Human disease data is not available in the database. Since the Human disease data is available only as ontology classes, we can use only *rdf:type* as predicate.

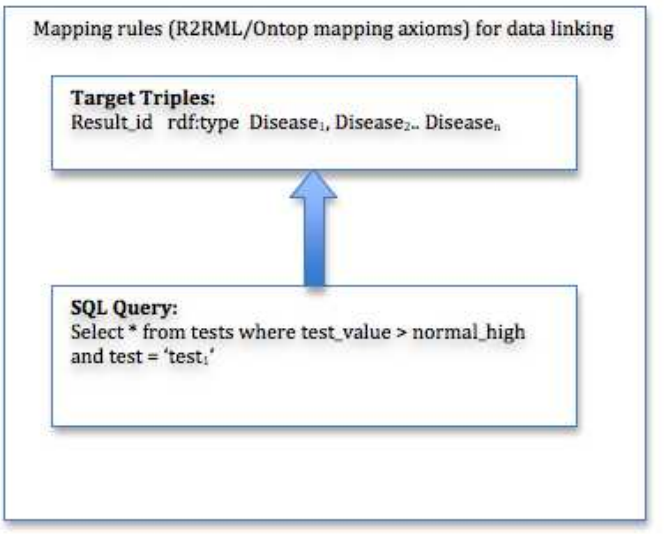

Figure 4.21 Mapping between test results and diseases

Mapping of Pathology Data Instances with Diseases

Different resources often use different vocabularies to represent data about the same entity. In order to generate integrated view on data, mapping has to be done linking the vocabularies. The mapping relies on links such as *rdfs:subclassOf, rdfs:subPropertyOf , owl:equivalentClass and owl:equivalentProperty*. We have used *Ontop* mappings along with *protégé* to perform data linking. OBDA information such as connection URL, database username and password and appropriate driver class needs to input to establish connection to underlying data. Once the connection is established, mappings are created in mapping manager. The Ontop also provides assistance in mapping creation via mapping assistant-BETA.

Mapping rules were created for linking the pathology lab tests data with Human disease ontology using external resources (table 4.6 Lab test association with disease chart). These mappings were created using *Ontop mapping editor* available in *Protégé*, which facilitates the data retrieval in *RDF* format using *Ontop SPARQL* editor and *Quest* reasoner. The pathology lab dataset has the test related details stored in the *tests* table. The table records the normal_high, normal_low and actual test result value for each test.

This information was used to identify abnormalities in the test results. Further, comparing the actual test result value with the normal high and normal low, we inferred possible diseases based on whether the abnormal value is high or low. The *result_id* of the test with abnormal high or low values is fetched and then compared with the possible diseases chart to assign it to the disease classes. For example, for the '*MCH*' test, the abnormal high value is associated with *Folate* and *Vitamin B12* deficiencies. So, the result ids with abnormal high values for this test are assigned to the *Folate_deficiency* and *Vitamin_B12_deficiency* classes.

To create mapping rules to link the lab tests to disease, it involves following steps:

(i) The mapping axioms for class *'Tests'* and related data type properties need to be created. We are going to use the same mapping axioms created in section 4.2.2.2.

(ii) Then the class 'Tests' is linked with disease by selecting the primary key column of the table as subject and mapping it to equivalent disease ontology class. It involves populating '*Focus on URI'* field with primary key column i.e. '*result_id'* in the data preview table and then selecting the disease class 'Vitamin_B12_Efficiency' from the '*Mapping for class'* field. This will result in the mapping, "*:result_id a Vitamin_B12_deficiency "*. Here the subject '*result_id'* value is linked to object class '*Vitamin_B12_deficiency'* with the predicate '*rdf:type'*. Only the result id's that satisfy the SQL query condition are mapped to the corresponding diseases.

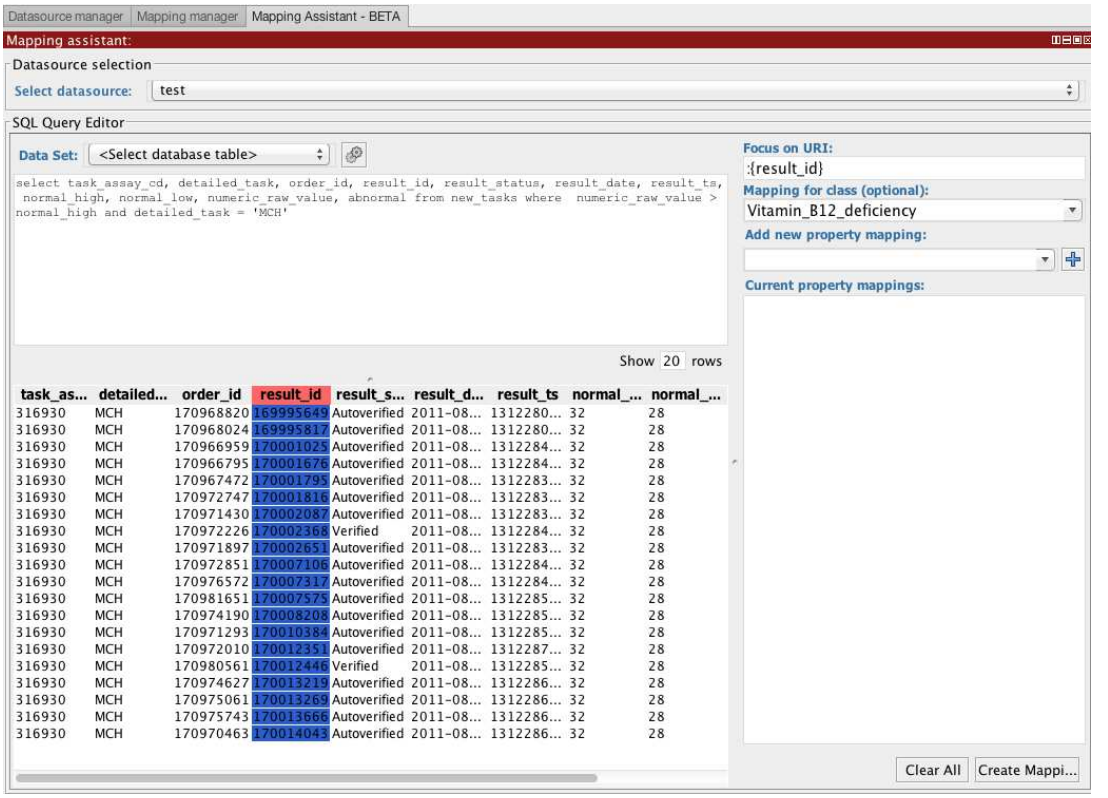

Figure 4.22 Disease class mapping with Tests

.

Figure 4.23 and 4.24 shows the final mappings of MCH-abnormal tests with diseases obtained using above steps,

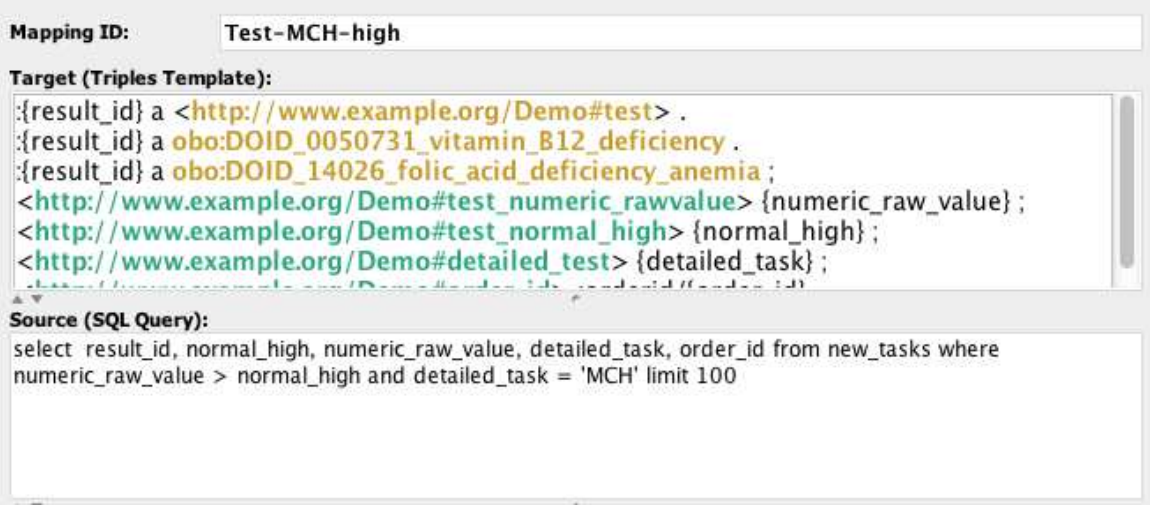

Figure 4.23 MCH-high mapping axiom

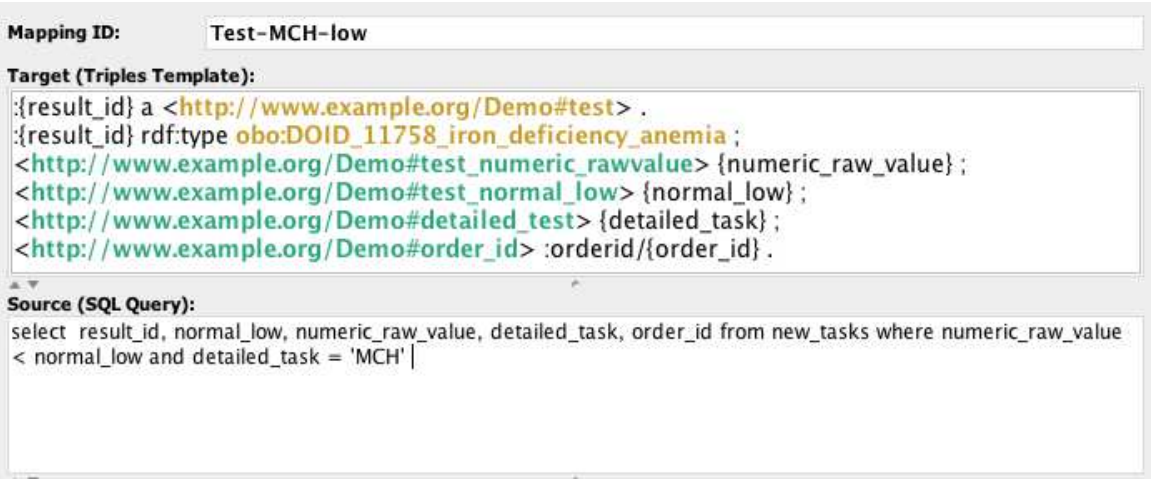

Figure 4.24 MCH-low mapping axiom

As the '*result_id'* is mapped to the disease class, all the other data type properties that are mapped to that 'r*esult_id'* are also get mapped to the disease class. Similarly, the mapping rules were created for all the tests available in the pathology lab tests dataset.

4.3.3 Disease Inference from the Linked Ontologies Using SPARQL

Once the system configuration, data linking between the pathology data ontology and Human disease ontology using predicate and the data mapping is done, the final step is to infer the diseases using *SPARQL* queries. The *Ontop Quest* reasoner along with the *SPARQL* query performs disease inference. Figure 4.25 shows the SPARQL query that can be applied to the system built to fetch possible disease(s) for each patient under a given physician id.

```
select  ?disease ?patient_id ?detailed_test
where
 {
?encounter_id a <http://www.example.org/Demo#encounter> ; <http://www.example.org/Demo#hasEncounter_For> ?patient_id ;
<http://www.example.org/Demo#hasEncounter_By> <http://www.example.org#physician_id/10008>.

?order_id <http://www.example.org/Demo#hasOrders> ?encounter_id ; <http://www.example.org/Demo#orderable> ?orderable  ;
<http://www.example.org/Demo#order_status> ?order_status ; <http://www.example.org/Demo#abnormal_orders> ?orderabnorm ;
<http://www.example.org/Demo#order_date> ?order_date  .

?testresultid a ?disease ; <http://www.example.org/Demo#order_id> ?order_id ;
  <http://www.example.org/Demo#detailed_test>  ?detailed_test .
}
```

Figure 4.25 SPARQL query to retrieve disease from lab test

The query consists of 11 triple patterns, which can fetch data from 9 attributes from five different tables (*new_encounter, new_order, new_physician, new patient* and *new_tasks*) represented by the classes *encounter, order, physician, patient* and *tests* respectively. 1st triple represents the '*encounter*' class relation. 2nd triple represents the object type relation between the *encounter* and *patient*. Similarly, 3rd triple represents the object type relation between the *encounter* and the *physician_id* #10008. 4th triple represents the object type relation between the *order* and *encounter* class. 5th to 8th triple represents data type relation of *order* class with its properties. The 9th triple represents the *disease* class relation with '*test_result_id'*. Rest of the triples represents the data type relation of *test* class with its properties. The query is executed on the pathology data ontology and the Human disease ontology linked in the data-linking step explained above. Other than disease inference, the query can also successfully fetch additional details related to the patients like the *encounter_id, patient_id, order_id, orderable, order_status, order_abnormal, Order_date, test_result_id and detailed_test*. These details can be more informative with respect to the diseases inferred and the patient history. Further, the query can also provide information about the ailments of all the patients consulted by a physician. This knowledge can also be helpful to the physician as well.

Table 4.7 shows the sample output of disease and lab test data linking retrieved for few patients.

Patient_id	Tests	Diseases
5322786	MCH, MCHC, RDW	Iron deficiency, anemia, Folic acid deficiency, Vitamin B12 deficiency
3811339	MCH	Folic acid deficiency, anemia, Vitamin_B12 deficiency
3293888	HCT	Anemia, hemorrhage, cirrhosis, Iron, vitamin or Folate deficiency, bone marrow damage, leukemia, lymphoma, Blood loss, hemolysis.
1053805	MCH, MCV, Neutro percent auto, Lymph percent auto, Eos percent auto	Leukemia, Chemotherapy, Lymphoma, Bone marrow failure, Vitamin B12 deficiency, Parasitic infection, viral infection, Folate deficiency

Table 4.7 Sample output of disease and Lab data linking

The pathology data ontology and the Human disease ontology were successfully linked. The data linking process using '*rdf:type'* as predicate helped us to link an ontology built from RDB schema to an ontology of only classes with no underlying relational database. This type of data linking is particularly useful in the case of using already existing ontologies and external resources to increase the amount of information or knowledge extracted from the ontology built on a relational database. The data linking process can be easily customized to accommodate these requirements. In our work, we aimed at efficient data retrieval and data analytics on the pathology lab tests data by creating ontology. But, linking it with the Human disease ontology extended its potential to infer

possible diseases from the test results for a patient. With the pathology data ontology, we could only get details about the patient and the results of the tests (i.e., normal or abnormal). Now, since the Human disease ontology has been linked to the pathology data ontology, based on the test result and the type of abnormality (i.e., low or high from normal value) the appropriate possible diseases can be inferred. Thus demonstrating the value added to the pathology data ontology by the data linking process.

4.3.4 Section Summary

Using Linked data techniques for inferring possible diseases from pathology lab data was challenging, as the data source did not have disease related data. This challenge was addressed by studying extensive online resources related to lab test data and disease association. Information like the normal and abnormal test result values were gathered for tests to infer the possible diseases. Linked data techniques were used to link the pathology lab data ontology and disease ontology and the diseases relating to patients were retrieved using SPARQL.

4.4 Concluding Remarks

This chapter presented the implementation details of the semantic model for data analytics in two distinct phases. The first section discussed the use of semantic methods for conceptual modeling that was used for ontology construction from relational database. In addition, it also discussed the formulation of mapping rules for ontology instantiation along with the process of ontology-based data. The second section illustrated the data linking framework utilizing the previously developed semantic models to link the disease with Pathology lab data to retrieve possible disease(s) for patients.

CHAPTER 5 RDF DATA ANALYSIS

5.1 RDF Analysis Using Ontop API

The interlinked nature of RDF data along with its clearly defined semantics form a great basis to perform data linking and analysis which helps in distilling valuable insights from the data. Currently available Linked Data technology such as RDF, SPARQL provides rich capabilities to perform slice and dice operation by supporting features including aggregation, nested and distributed queries. But it lacks the support for programming such as recursion and iteration to perform more complex data analysis. To address this challenge, we perform the programming tasks on the Semantic Web model developed in previous chapter thereby gaining advantage of both semantics as well as data analysis.

For the pathology lab dataset, conceptual modeling and appropriate linking was done, so that the data is available as RDF triples. The pathology lab data contains details about the patients, tests and the test results. With this dataset, we can analyze information like patient details under each physician, test results for all orders of a particular patient, status of all orders in an encounter of a particular patient, tests taken in a particular order and so on using normal *SPARQL* queries that supports aggregate functions. But with the help of programming we were able to perform more complex analysis such as clustering and classification involving deriving order patterns.

In our approach, we have employed *Ontop SPARQL* to perform data fetching and Java Jena to perform specific tasks on the resulting RDF dump. The Ontop API enables the use of custom java programs to demonstrate knowledge querying and analysis using the mappings created. But to use the *Ontop Quest* in a Java-based system, it needs to be integrated as a dependency. The *Ontop* is published on the central *Maven* repository, so it can be easily integrated as a dependency. *Maven* is project management and comprehension tool by Apache. It allows users to declare the dependencies in a *XML* file. The suitable dependency needs to be added to the pom.xml file. Dependencies are

added and identified using artifact IDs. Figure 5.1 shows the pom.xml entry for Quest DB API. The dependencies that are required for working with Ontop Quest are *ontop-obdalib-core, ontop-obdalib-owlapi3, ontop-obdalib-protege4, ontop-obdalib-sesame, ontop-obdalib-r2rml, ontop-quest-db, ontop-quest-owlapi3, ontop-quest-sesame, and ontop-reformulation-core*.

```
<dependency>
<groupId>it.unibz.inf.ontop</groupId>
<artifactId> ontop-quest-db </artifactId>
<version>1.12.0</version>
</dependency>
```

Figure 5.1 Dependency declaration in pom.xml

Further, in MySQL the quotes and double quotes are used in a non-standard way leading to "*table not defined*" error, when accessed from an API. This error can be avoided by specifying ANSI mode in the connection string like, *jdbc:mysql://myserver:port/mydatabase?sessionVariables=sql_mode='ANSI'*.

5.2 Mining Frequent Order Sets in Pathology Lab Data

Frequent pattern refers to how frequent a pattern occurs in a data set. A pattern is a conjunction of items. It defines a set of items, subsequences, sub-graphs or a group of instances. It was first proposed in the context of frequent item sets and association rule mining for market basket analysis. Frequent mining follows *Apriori* property, which states that any subset of a frequent pattern must be frequent. It was developed with the aim to find inherent regularities in the data.

5.2.1 Significance of Order Sets

Order sets assists in increasing physician practice efficiency by studying pathology lab test ordering pattern. It acts as a key to identify the possible reasons behind placing a particular order for a subject (patient). It proposes the feasibility of an effective method with no memory requirement restriction for order set generation from the massive volumes of data. The order sets will assist in answering below mentioned questions,

- To find the percentage of orders placed that are similar.
- To find the tests those are most often ordered together.
- To look at the peaks and valleys of ordering patterns.

We have derived all the possible order sets from n=1 to n = 15 where 'n' denotes the size of the sets.

5.2.2 Method Used for Generating Frequent Order Sets

The massive volume of pathology lab data limits the mining process, as it requires excessive storage and computational requirements. To overcome this disadvantage, we have used the linked data analytical framework to analyze the orders that are placed simultaneously. It involves an approach that converts the data into *RDF* dumps virtually, which can be further retrieved by means of querying and the results obtained can be mined to derive the order sets. If 'n' denotes the number of distinct orders in the dataset, then the orders can be grouped into sets starting from 1 order per set to 'n' orders per set.

For n=1: $\{O_1\}$, $\{O_2\}$, $\{O_3\}\ldots\{O_n\}$;
For n=2: $\{O_1, O_2\}$, $\{O_2, O_3\}$, $\{O_1, O_3\}\ldots\{O_3, O_n\}$, Where, O_n is the order 'n'.

The order sets for each level or different values of 'n' were generated using a recursive function. The recursive function can iterate through the orders retrieved from the virtual RDF dumps and generate the simultaneously placed orders for each set at a given level.

The levels were automatically incremented till the maximum value that is the number of distinct orders in the dataset. Initially, the variable '*n*' is set to 1 and incremented in every loop. The number of orders in our dataset was 15, so the iteration was done till value of 'n' reached 15. For each '*n*' value, a set of combination is created based on '*n*' value along with the frequency counter set to zero initially. For every combination, the database is scanned to check for combination set entry and the counter is increased by 1 whenever the entry is found. The counter is increased only when the combination set belongs to the same '*encounter_id'* and same '*Order_date'*.

5.2.3 Pathology Lab Order Sets Results

Using the pathology lab test data,

(i) We are able to successfully identify the concurrently placed orders sets for two different time durations such as (01-Aug-2011 to 20-Jun-2014) and (21-Jun-2014 to 02-May-2016).

Table 5.1 shows the concurrently ordered tests that are most often placed together during Aug 2011 to Jun 2014 in same encounter. The red color order sets indicates that it's the first highest occurrence and blue represents the second highest occurrence. The least occurring sets are ignored because there are multiple entries of order sets available. To indicate the order sets are from first interval and second interval we have used the word 'F' and 'S' respectively and to indicate the order size we have suffixed it with the size and the subscript '1' indicates its first highest occurrence and subscript '2' indicates its second highest occurrence, e.g.'F141' indicates that it is the first highest 14 set order from the first time interval. It can be seen that $F1_1$ contributes to the highest order sets, $F1_2$ is the second highest and $F13_1$ is the third highest order sets during 2011-2014.

Order set size		Concurrently ordered tests (order set)	Order set volume
14	$F14_1$	ALT_AST_CBC_Cholesterol_Creatinine_ElectrolytePanel_GGT_GlucoseAC_HDLCholesterol_PT_TSH_Triglycerides_Urea	1814
	$F14_2$	ALT_AST_CBC_Cholesterol_Creatinine_Electrolyte Panel _GGT_Glucose Random_HDL Cholesterol_PT_TSH_ Triglycerides_Urea	32
13	$F13_1$	ALT_Alkaline Phosphatase_CBC_Cholesterol _Creatinine _ Electrolyte Panel_GGT_Glucose AC_HDL Cholesterol_TSH_ Triglycerides_Urea	24357
	$F13_2$	ALT_CBC_Cholesterol_Creatinine_Electrolyte Panel_GGT_ Glucose AC_HDL Cholesterol_PT_TSH_Triglycerides_Urea	864
12	$F12_1$	ALT_AST_CBC_Cholesterol_Creatinine_Electrolyte Panel_ GGT_Glucose AC_HDL Cholesterol_TSH_Triglycerides _Urea	13546
	$F12_2$	ALT_AST_Alkaline Phosphatase_CBC_Cholesterol_Creatinine _Electrolyte Panel_Glucose AC_HDL Cholesterol_TSH_ Triglycerides_Urea	10396
11	$F11_1$	ALT_AST_CBC_Cholesterol_Creatinine_Electrolyte Panel_Glucose AC_HDL Cholesterol_TSH_Triglycerides_Urea	20431
	$F11_2$	ALT_Alkaline Phosphatase_CBC_Cholesterol_ Creatinine_ Electrolyte Panel_Glucose AC_HDL Cholesterol_TSH_ Triglycerides_Urea	13952
10	$F10_1$	ALT_CBC_Cholesterol_Creatinine_Electrolyte Panel_Glucose AC_HDL Cholesterol_TSH_Triglycerides_Urea	16310
	$F10_2$	ALT_AST_CBC_Cholesterol_Creatinine_Electrolyte Panel_Glucose AC_HDL Cholesterol_TSH_Triglycerides	9262
7	$F7_1$	ALT_AST_Alkaline Phosphatase_CBC_Creatinine_Electrolyte Panel_Urea	4724
	$F7_2$	CBC_Cholesterol_Creatinine_GlucoseAC_HDCholesterol_TSH_Triglycerides	1992

5	$F5_1$	CBC_Creatinine_Electrolyte Panel_Glucose Random_Urea	6944
	$F5_2$	CBC_Creatinine_Electrolyte Panel_PT_Urea	4099
3	$F3_1$	Creatinine_Electrolyte Panel_Urea	7896
	$F3_2$	CBC_Creatinine_Electrolyte Panel	4873
2	$F2_1$	CBC_TSH	15349
	$F2_2$	CBC_PT	11596
1	$F1_1$	PT	82410
	$F1_2$	CBC	63648

Table 5.1 Order sets placed during 2011-2014

Table 5.2. shows the concurrently ordered tests that are most often placed together during Jun 2014 to May 2016 in the same encounter. It can be seen that $S1_1$ contributes to the highest order sets, $S1_2$ is the second highest and $S2_1$ is the third highest order sets during 2011-2014.

Order set size		**Concurrently ordered tests (order set)**	**Order set volume**
14	$S14_1$	ALT_AST_CBC_Cholesterol_Creatinine_ElectrolytePanel_GGT _Glucose AC_HDLCholesterol_PT_TSH_Triglycerides_Urea	392
	$S14_2$	Urea_TSH_PT_GlucoseRandom_GGT_ElectrolytePanel_Creatini ne_AlkalinePhosphatase_AST_ALT_Triglycerides_HDL Cholesterol_Cholesterol_CBC	11
13	$S13_1$	Urea_TSH_GlucoseAC_GGT_ElectrolytePanel_Creatinine_ AlkalinePhosphatase_AST_ALT_Triglycerides_HDL Cholesterol _Cholesterol_CBC	4187
	$S13_2$	Urea_TSH_PT_Glucose AC_GGT_Creatinine_Alkaline Phosphatase _AST_ALT_Triglycerides_HDLCholesterol_ Cholesterol_CBC	679
12	$S12_1$	Urea_TSH_GlucoseAC_GGT_Creatinine_AlkalinePhosphatase_	7346

		AST_ALT_Triglycerides_HDL Cholesterol_Cholesterol_CBC	
	$S12_2$	Urea_TSH_GlucoseAC_GGT_ElectrolytePanel_Creatinine_AST _ALT_ Triglycerides_HDL Cholesterol_Cholesterol_CBC	2777
11	$S11_1$	Urea_TSH_GlucoseAC_GGT_Creatinine_AST_ALT_Triglycerides_HDL Cholesterol_Cholesterol_CBC	5759
	$S11_2$	Urea_TSH_GlucoseAC_ElectrolytePanel_Creatinine_Alkaline Phosphatase_ALT_Triglycerides_HDL Cholesterol_Cholesterol_CBC	4893
10	$S10_1$	Urea_TSH_GlucoseAC_Creatinine_AlkalinePhosphatase_ ALT_ Triglycerides _HDL Cholesterol_Cholesterol_CBC	8734
	$S10_2$	Urea_TSH_GlucoseAC_Creatinine_AST_ALT_Triglycerides_ HDL Cholesterol_Cholesterol_CBC	6888
7	$S7_1$	TSH_GlucoseAC_Creatinine_Triglycerides_HDLCholesterol_ Cholesterol_CBC	3751
	$S7_2$	Urea_GlucoseRandom_Creatinine_AlkalinePhosphatase_AST_ ALT_CBC	2058
5	$S5_1$	Creatinine_Alkaline Phosphatase_AST_ALT_CBC	2457
	$S5_2$	Urea_GlucoseRandom_ElectrolytePanel_Creatinine_CBC	1861
3	$S3_1$	Urea_Creatinine_CBC	6017
	$S3_2$	Triglycerides_HDL Cholesterol_Cholesterol	2357
2	$S2_1$	TSH_CBC	10911
	$S2_2$	PT_CBC	5070
1	$S1_1$	PT	46093
	$S1_2$	CBC	39092

Table 5.2 Order sets placed during 2014-2016

Table 5.3 shows the count of distinct concurrently ordered sets during two different intervals such as 2011 to 2014 and 2014 to 2016.

Order Set Size	Count of distinct order sets during 2011 to 2014	Count of distinct order sets during 2014 to 2016
14	6	4
13	23	19
12	84	64
11	192	155
10	317	269
9	446	363
8	548	445
7	608	501
6	541	486
5	453	394
4	332	281
3	197	178
2	82	74
1	15	15

Table 5.3 Number of distinct order sets during 2011-14 and 2014-16

(ii) Compared and identified the differences between the order sets placed during two different time durations such as (01-Aug-2011 to 20-Jun-2014) and (21-Jun-2014 to 02-May-2016).

Order sets during first interval (2011-14)	Change in abundance rank compared to Order sets in second interval (2014-16)	
	From	To
$F14_1$	No Change	
$F14_2$	No Change	
$F13_1$	1	8
$F13_2$	Not available	
$F12_1$	1	2
$F12_2$	2	3
$F11_1$	1	4

$F11_2$	No change	
$F10_1$	1	3
$F10_2$	2	4
$F7_1$	1	3
$F7_2$	2	1
$F5_1$	1	2
$F5_2$	2	10
$F3_1$	1	3
$F3_2$	2	7
$F2_1$	No change	
$F2_2$	No change	
$F1_1$	No change	
$F1_2$	No change	

Table 5.4 Order sets comparison (2011-14) Vs. (2014-16)

Comparing the order sets over the two year time period, the trend of highest placed order sets does not change. Showing, patients took same kind of tests more often in the two time periods. Interestingly, the order set $F13_2$, which occurred 864 in the first time period did not occur even one time in the second time period. Indicating that the particular combination of tests where not prescribed to any patient in the second time period. Our analysis may be not able to determine the exact reason behind this observation, but definitely a pointer towards further analysis. Out of the 20 order sets (Table 5.3), 7 of them did not change in their abundance ranking between the two time periods and 12 order sets slipped from their first time period abundance ranking. Suggesting that even though the combinations of tests taken in the two periods are same, for many order sets the number of times a particular combination of tests taken has reduced. Except the order set $F7_2$, which actually moved up from rank 2 to rank 1, indicating an increase in this tests combination. This kind of intuitive analysis from the dataset can help the physicians and lab technicians with well-informed decision-making. The trends between the time

periods and within each time period can also be analyzed to extract more detailed and unexplored information from the dataset.

5.3 Concluding Remarks

This chapter presented the RDF data analysis challenges and a solution showing how the challenges can be addressed programmatically using our Data Linking framework. Further, the chapter also explained in detail the API setup required for enabling programming with the *Ontop Quest* for accessing the RDF data. We have also discussed the methods involved in mining frequent sets in pathology lab dataset along with the significance of frequent set generation. Order patterns were generated from pathology lab data in RDF format for two different time periods (2011-14) and (2014-16). The order sets generated is further studied in detail to obtain the difference in ordering sets over the time period. In the next chapter, we will discuss in detail about the evaluation techniques used for our approach.

CHAPTER 6 EVALUATION

This chapter presents the final step in our research methodology, which is to ensure the validity of the knowledge model. We assess the construction of our pathology lab data ontology on the grounds whether it correctly implements the ontology requirements. We devised two evaluation strategies to ensure that our ontology has adequate representational adequacy and it functions as intended. Following a brief discussion on this ontology evaluation, we will present the scenario-based evaluation of developed Linked Data framework with details in subsequent section.

6.1 Evaluation of Pathology Data Ontology

The evaluation process ensures the legitimacy of the knowledge model. There are numerous criteria available for ontology evaluation (Casellas, 2009). Evaluation approach can be either *Quantitative* or *Qualitative*. The criteria selection for evaluation is based on the purpose and scope of the ontology. And it should ensure that the correctness of the knowledge by building it in the appropriate manner. We have chosen the following criteria for evaluating our pathology lab data knowledge model:

1. Conciseness
2. Consistency
3. Completeness

6.1.1 Conciseness

Conciseness is the most important factor to be considered during the process of ontology construction. It denotes that the ontology should be relevant and non-redundant. In other words, it should not contain irrelevant or redundant definitions, as these redundancies cannot be deduced from stated definitions. To ensure the conciseness, we used *Hermit* and *Ontop Quest reasoners* results to show the absence of irrelevant and redundant

definitions. In addition to the conciseness of the definitions, we also made sure that there are no redundant rules or concepts.

6.1.2 Completeness

Completeness of ontology signifies its ability to fulfill all the competencies that are put forth during requirement gathering process. It means the ontology developed should answer all the questions that are captured before the design phase. In our case, we have used schema-mapping rules to build an ontology. The rules are carefully built by reviewing extensive literature to satisfy the ontology requirements. The knowledge expected from ontology should be asserted explicitly. In some cases, this knowledge is inferred through reasoner. The rules, relationships and concepts required to attain the research objective is present in our developed pathology data ontology. Hence our model is complete with respect to the functional requirement of the application.

6.1.3 Consistency

Consistency refers to non-existence of asserted and inferred contradictory statements in the ontology. Logical inconsistency arises due to contradictory rules, is identified by the reasoner. We have used Pellet (E. Sirin, 2007) and *Ontop Quest reasoner* to check the consistency of pathology data ontology.

6.2 Evaluation of Linked Data Analytical Framework

The Linked Data Analytical framework is evaluated based on the performance of the SPARQL query and scenario-based evaluation for checking the validity of the developed model.

6.2.1 Performance Evaluation

The design time and run time required for the linked data analytical framework is calculated. The design time mainly involves the conceptual modeling, ontology

instantiation and data linking. The conceptual modeling was done programmatically and generates the ontology file. The program is very dynamic and can generate correct ontology even if the underlying RDMS schema changes and eliminating the need to redesign. But, the generation of mapping axioms for ontology instantiation and data linking was done manually and so the changes in the RDMS schema and external resources will not be automatically reflected in the mappings. Hence, every time the external resources or the database schema changes, the mappings need to be regenerated manually. This is the only additional design cost in the analytical framework with respect to changes in the resources.

Further, the run time of the Linked data analytical framework was analyzed calculating the time required to run Ontop SPARQL queries on the pathology lab data in relational database (22 Million records). The total time taken for generation of Virtual RDF views with linked external resources and fetching the entire 22 million records of pathology lab data via SPARQL to SQL query conversion is 8 minutes 47 seconds. The chart (Figure 6.1) shows the time required for different range of DB size (in number of records).

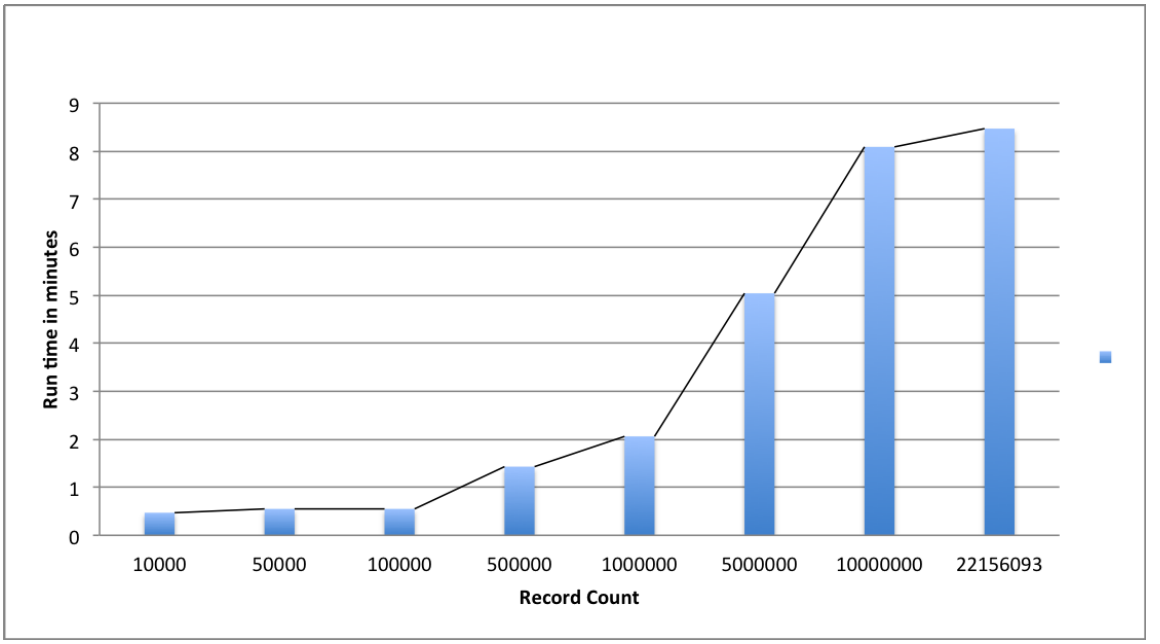

Figure 6.1 Performance – Runtime

6.2.2 Scenario-Based Evaluation

This section presents scenario-based evaluation of the developed Linked Data framework using semantic method to ensure the validity of developed model. The validity is the only way to ensure correctness of the developed knowledge model. Evaluation based on specific scenarios is one of the ways to assess the model. The intent of scenario-based evaluation is to access whether the semantic web model produces the desired output or not. This process involves setting up result expectations and reporting the results. Expectations define the desired output and meaningful conclusions we can draw from our developed model.

These are ontology-wide expectations that assess the inherent connections of the pathology lab data ontology and should always be fulfilled. Table 6.1 lists all our reasoner-specific expectations related to ontology instantiation that we expect to meet.

O1	Each encounter should infer only one patient.
O2	Each encounter should infer only one physician
O3	For a given patient, each order should infer only one encounter.
O4	For given patient, each test should infer only one order.

Table 6.1 Reasoner specific expectations related to Ontology instantiation

Table 6.2 lists some of our reasoner-specific expectations related to data linking that we expect to meet.

L1	Cushing's syndrome, Kidney diseases should be inferred for Sodium test with high result value.
L2	Addison's disease, Diarrhea, adrenal insufficiency should be inferred for Sodium test with low result value.
L3	Chronic kidney failure, Addison's diseases, diabetes should be inferred for a

	Potassium test with high result value.
L4	Cushing's disease should be inferred for Potassium test with low result value.
L5	Kidney disease, Cushing's diseases should be inferred for Chloride test with high result value.
L6	Lung disease, Emphysema should be inferred for Chloride test with low result value
L7	Lung disease, COPD should be inferred for CO2 test with high result value.
L8	Diarrhea, Kidney disease should be inferred for CO2 test with low result value
L9	Kidney disease should be inferred for Creatinine test with high result value
L10	None should be inferred for Creatinine test with low result value
L11	Kidney disease should be inferred for Urea test with high result value
L12	Liver disease, kidney disease should be inferred for Urea test with low result value.
L13	Heart disease should be inferred for Triglycerides test with high result value
L14	Heart disease should be inferred for Cholesterol test with high result value
L16	Heart disease should be inferred for HDL test with high result value

Table 6.2 Reasoner specific expectations related to data linking

The actual outcomes of evaluation against expected outcomes comes in table 6.3.

Expected outcome	**Actual Outcome**
O1	✓
O2	✓
O3	✓
O4	✓
L1	✓
L2	✓
L3	✓

L4	✓
L5	✓
L6	✓
L7	✓
L8	✓
L9	✓
L10	✓
L11	✓
L12	✓
L13	✓
L14	✓
L15	✓
L16	✓

Table 6.3 Actual outcomes Vs. Expected outcomes

Precision can be defined as the fraction of actual outcome that meet our specified expectations and is given by the formula:

$$\text{Precision} = \frac{(\text{Actual\_Outcomes} \cap \text{Expected\_Outcomes})}{(\text{Actual\_Outcomes} \cup \text{Expected\_Outcomes})}$$

After evaluation, precision is calculated to be,

Precision = (20/ 20) = 100%

This is because the system is deterministic in nature.

6.3 Concluding Remarks

In this chapter we have presented an account for the pathology data ontology evaluation. We discussed qualitative analysis of developed model where in focus was concentrated on completeness, consistency and conciseness. In addition, we have also evaluated the Linked Data framework and found that it has 100 percent precision with respect to correctness.

CHAPTER 7 CONCLUSION

It is significant to link the data with other related data to make it more informative and develop methods for deducing intuitive information from the Linked Data. But the process is not always easily accomplishable, depending on the varied structure of datasets. The pathology lab dataset used in this work has enormous data stored as relational database, which are vital to learn more about trends and patterns if it represented in Linked Data format. Hence there is a need for linking the pathology lab data to other related data. To address this, extensive studies and reviews were made to choose the most appropriate techniques for building a knowledge representation model for the pathology lab data.

We have built OWL ontology for the pathology lab dataset to represent the data model. The domain-specific ontology developed can be used for RDMS data extraction and data inference. The extracted relational data were converted into RDF triples by dynamic materialization of the RDF triples and avoiding the creation of a large RDF dump. The RDMS data was not physically migrated to RDF, eliminating the time and storage resources required. Data linking was performed with the developed ontology. Semantic analytics were performed on the developed ontology using SPARQL to infer intrinsic information from the dataset, which is otherwise difficult to infer from RDMS. We have also exhibited the data management capabilities such as efficient data retrieval and data transformation of developed system in addition to data analytics by means of generation of order sets over a period of 5 years. Further, the advantage of developing ontology for the pathology lab data was extended by linking it to disease ontology for inferring possible diseases from the pathology lab test results. Rules connecting pathology data ontology and Human disease ontology are constructed to achieve this.

Advantages of the proposed Linked Data analytical framework

1) The main advantage of our proposed linked data analytical framework is that the data to be analyzed is available in RDF. Hence, it can provide new insights by evaluating relationships and connections, instead of just statistics.

2) We can query both the schema and the data. For e.g.: we can retrieve all the relationships linking particular disease to other entities.

3) Our framework involves reasoner to draw inference. Hence, analytics can be done not only over data but over the schema as well.

Limitations of the proposed Linked Data analytical framework

1) Conceptual modeling process that involves generating RDF using schema conversion rules is constructed only for the relational data source. For other formats such as unstructured formats, CSV, Excel, etc. we have not implemented conversion to RDF.

2) SWRL rules are not supported hence the reasoning of mathematical expressions not possible.

3) Our disease deduction approach correctness depends entirely on the disease and lab test chart. Hence there are chances to produce incorrect results.

Possible Future Works

In future, we can extend the work by linking the developed model with the symptoms, which has high probability of deducing cent percent accurate results. Also, data analytics can be done on these Linked Data to generate disease related patterns for patients based on the tests ordered. We can also generate test sets for a patient during encounter to

study the testing patterns. Not all the tests in the order are placed together all the time. Hence, analyzing tests sets would give us better insights at test level.

The ontology developed for the pathology data definitely enabled us with extracting the implicit information in the data. The ability to extend the ontology by linking it with related ontologies makes it more useful and can easily makes deductions from the Linked data. Further, the dynamic materialization technique for data instantiation used in our approach seemed more advantageous from the obtained results in retrieving Linked data.

WORKS CITIED

(2016). Retrieved from wiki.dbpedia.org.

3Kbo. (2015). Retrieved from Strategies for Building Semantic Web Applications: http://notes.3kbo.com/node/30

A. Ferrara, A. N. (2011). Data linking for the semantic web. *International Journal on Semantic Web and Information Systems* (3), 46-76.

A. Hogan, A. H. (2011). Semantic Search- Reading Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web* .

A. Ismail, H. A.-f. (2015). Querying DBpedia Using HIVE-QL.

A. Jentzch, R. I. (2010). Silk - Generating RDF links while publishing or consuming Linked data. *CEUR Workshop Proceedings , 658*, 53-56.

A. Khandelwal, I. J. (2011). Linked rules: Principles for rule reuse on the web. 108-123.

A. Nikolov, V. U. (2008). Integration of semantically annotated data by the KnoFuss architecture. *In 16th International Conference on Knowledge Enginerring and Knowledge Management (EKAW 2008)* .

A. Rodriguez - Gonzalez, A. G.-C. (2011). Automated Diagnosis Through Ontologies and Logical Descriptions. *International Journal of Decision Support System Technology , 3* (1), 21-39.

AACC. (2015). Retrieved Jan 28, 2016, from https://www.aacc.org.

Ahmud-Boodoo, R. (2016). E-Learning and the semantic Web: Descriptive literature review. *Mobile Computing and wireless Networks: Concepts, Methodology, Tools and Applications* , 11-40.

Amit Sheth, C. T. (2014). *Continuous Semantics to Analyze Real- Time Data.* Retrieved from http://knoesis.wright.edu/sites/default/files/STM10-IC-Continuous-Semantics.pdf
APS Healthcare. (n.d.). Retrieved Jan 17, 2016, from http://www.apshealthcare.com

Auer, A. N. (2011). LIMES - a time efficient approach for large-scale link discovery on the web of data. *In Proc. of the International Joint Conference on Artifical Intelligence (IJCAI 2011)* .

Berners Lee, T. (2006). *Relational Databases on the Semantic Web.* Retrieved from http://www.w3.org/DesignIssues/RDB-RDF.html

Berners-Lee et. al., B. (2006). Tabulator: Exploring and Analyzing linked data on the Semantic Web. *The 3rd international semantic web user interaction workshop.*

Berners-Lee, T. (2006). *Linked Data-Design Issues*. Retrieved from http://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T. (1998, September). *Relational Databases on the Semantic Web*. Retrieved from http://www.w3.org/DesignIssues/RDB-RDF.html

Bizer, T. H. (2011). In *Linked Data : Evolving the web into a global data space.* (1st edition, 2011 ed.). Morgan & Claypool.

Blin (2014), O. C. *RDF database Systems: Triples Storage and SPARQL Query Processing.*

Borodaenko, D. (2009). On-demand RDF to relational query translation in Samizdat RDF store. *2009 IEEE International Conference on Intelligent Computing and Intelligent System* , 413-417.

C. Arpirez, O. C.-L. (2001). WebODE: A Scalable Workbench for Ontological Engineering. *Proceedings of the 1st International Conference on Knowledge Capture: Victoria, Canada:2001* , 6-13.

C. Bizer, A. S. (2004). D2RQ- Treating Non-RDF Databases as Virtual RDF Graphs. *World Wide Web Internet and Web Information Systems* (26).

C. Bizer, T. H.-L. (2009). Linked data- the story so far. *International journal on Semantic Web and Information Systems , 5* (3), 1-22.

C. Stadler, J. L. (2011). LinkedGeoData: A core for a Web of Spatial Open Data.

C.Ramathilagam, M. (2013). A Framework for OWL DL based Ontology Construction from Relational Database using Mapping and Semantic Rules. *International Journal of Computer Applications (0975-8887) , 76.*

Cambridge Semantics. (2016). Retrieved Feb 6, 2016, from http://www.cambridgesemantics.com/resources/blog/ontologies-conceptual-models

Casellas, N. (2009). Ontology Evaluation through usability measures: An experiment with the SUS scale in the legal document. 549-603.

Cerbah, F. (2008). Learning highly structured semantic repositories from relational databases: The RDBToOnto tool. *In Proceedings of the 5th European Semantic web conference(ESWC)* , 777-781.

Chiky, M.-A. A. (2014). From Business Intelligence to Semantic Data Stream Management. *Advances in Conceptual Modeling* , 85-93.

Chris Bizer, G. T. (2007). *Disco- Hyperdata Browser*. Retrieved from http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/

Cory Henson, A. S. (2012). "Semantic Percepttion: Converting Sensory Observations to Abstractions". *IEEE Internet Computing , 16*.

D. Calvanese, B. C. (2012). OBDA with the Ontop Framework. 1-8.

D. Calvanese, B. C.-e. (2015). Ontop: Answering SPARQL Queries over Relational Databases. *Semantic Web Journal* .

D. Calvanese, G. D. (2011). The MASTRO system for ontology-based data access. *Semantic Web , 2* (1), 43-53.

D. Calvanese, G. D. (2011). The MASTRO system for ontology-based data access. *Semantic Web , 2* (1), 43-53.

D. Tsarkov, I. H. (2006). Fact++ Description Logic Reasoner System Description. *Proceedings of the Third International Joint Conference* .

D.Fin, F. (2014). Concepts and Trends in Healthcare Information Systems. *Annals of Information Systems* , 79-101.

D2RQ. (2011). *D2RQ Accessing Relational Databases as Virtual RDF Graphs*. Retrieved Nov 2015, from http://d2rq.org

Daniel L. Rubin, N. F. (2007, Nov). Protege: A tool for Managing and Using Terminology in Radiology Applications. *J Digital Imaging* , 34-46.

David Beckett, T. B. (2011). *Turtle - Terse RDF Triple Language*. Retrieved from W3C:

http://www.w3.org/TeamSubmission/turtle/

David Beckett, T. B.-L. (2014, Feb 24). Retrieved March 23, 2016, from W3C: www.w3.org/TR/Turtle

Divanshu Gupta, A. S. (2014). Graphical Analysis and Visualization of Big Data in Business Domains. *Big Data Analytics: Third International Conference* , 53-57.

DuCharme, B. (2011). *Learning SPARQL.* O'Reilly Media.

Duraspace. (2014). *Linked data for libraries (LD4L)*. Retrieved from https://wiki.duraspace.org/

E. Ioannou, C. N. (2010). Intelligent entity matching and ranking :OKKAM project.

E. Sirin, B. P. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics , 5* (2), 51-53.

E.Bontos, M. (2005). Case Studies on Ontology Reuse. *Processings of I-kNOW'05* , 345-353.

EUCLID. (2012). *Building Linked Data Applications*. Retrieved from Education Curriculum for the usage of Linked Data: http://www.euclid-project.eu/modules/chapter5

F. Michel, J. M.-Z. (2014). A survey of RDB to RDF translation approaches and tools. (May), 23.

F. Priyatna, R. C.-m. R2RML-based access and querying to relational clinical data with morph-RDB. *1*, 1-10.

F. Scharffe, Y. L. (2009). RDF-AI : an architecture for RDF datasets matching, fusion and interlink. *In Workshop on Identity and Reference in Knowledge Representation, IJCAI 2009.*

G. Cheng, Y. Q. (2009). Searching Linked objects with Falcon. *International Journal on Semantic Web and Information Systems* .

G. De Giacomo, D. L. (2012). Mastro: Ontology-based data access at work (extended abstract). *7567 LNCS*, 667-668.

G. Kobilarov, C. B. (2009). DBpedia- A linked Data Hub and Data Source for Web and Enterprose Applications. *International Worls Wide Web Conference , 18*, 1-3.

G. Tummarello, R. C. (2010). Sig.ma: Live views on the web of Data. *Web Semantics: Science,Services and Agents on the World Wide Web , 8*.

Gabriel, K. .. (2015). *From the Web of Documents to the Linked Data.*

H. Glaser, I. M. (2008). Rkbexplorer.com : aknowledge driven infrastructure for linked data providers. *In Proceedings of the 5th European semantic web conference on the semantic web : research and applications* .

H.-M., H., & P., K. (2014). Semantic Data Interoperability: The Key problem of Big Data. *Big Data Computing* , 245-269.

Hasan, R. (2014). Docteur en Sciences Predicting Query Performance and Explaining Results to Assist Linked Data Consumption.

Hausenblas, M. (2008). Quick Linked Data Introduction.

Heflin, J. (2007). An Introduction to the OWL Web Ontology Language. *Lehigh University* .

I. Spasic, S. A. (2005). Text mining and ontologies in biomedicine:Making sense of raw data. *Briefings in Bioinformatics* , 239-251.

IMWF. (n.d.). Retrieved Jan 6 29, 2016, from http://www.iwmf.com

J. Iturrioz, I. A. (2015). Linked Data Wrappers atop Yahoo's YQL. *Services and Applications over Linked APIs and Data – SALAD2015* , 20-21.

J. Lehmann, R. I. (2014). DBpedia - A Large scale Multilingual Knowledge Base Extraxted from Wikipedia. *Semantic Web Journal , 1*, 1-5.

J. Pan, A. H. (2010). *Scalable OWL reasonong for Linked Data.*

J. Sequeda, D. M. (2013). Ultrawrap: SPARQL execution on relational data . *Journal of Web Semantics , 22*, 19-39.

J. Sequeda, M. A. (2014). OBDA: Query rewriting or materilaization? In practice, both! *8796*, 535-551.

J. Volz, C. B. (2009). Discovering and maintaining links on the web of data. *In 8th International Semantic Web conference (ISWC 2009)* .

J. Yuan An, A. B. (2006). Discovering the Semantics Of Relational Tables Through Mappings. *Journal on Data Semantic VII* .

Jaun F. Sequeda, H. S. (n.d.). Direct Mapping SQL Databases to the semantic Web: A survey.

Jentzsch, R. C. (2014, August). *Linking Open Data cloud diagram*. Retrieved from http://lod-cloud.net

K. Siegemund, E. T. (2011). Towards ontology-driven requirements enginerring. *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference* .

K. Vasconcelos, C. L. OntoEditor: a Web Tool for Manipulating Ontologies stored in Database servers. *Av. Aprigio Veloso , 882*, 58109-970.

Kew, R. (n.d.). *The National Archives*. Retrieved March 28, 2016, from Web archiving and web continuity guidance: http://www.nationalarchives.gov.uk/webarchive/guidance.htm#how-to-link-to-web-archive

L. Ma, J. M. (2007). Semantic web technologies and data management. *Proc. of W3C Workshop on RDF Access to Relational Databases* .

L.Ravi, N. (2012, Dec). Towards Ontology Development Based on Relational Database. *International Journal of Web Technology* .

LargeTripleStores. (n.d.). Retrieved from W3C wiki: https://www.w3.org/wiki/LargeTripleStores

Linked data. (n.d.). *Linked Data- Connect Distributed Data across the web*. Retrieved Jan 10, 2016, from http://linkeddata.org/

M. Arenas, A. B. (2012, Sept 27). *A direct mapping of relational data to RDF*. Retrieved from http://www.w3.org/TR/rdb-direct-mapping/

M. Foulonneau, G. A. (2010). *Linking Data on the Future Internet Semantic networks and intelligent objects*. Tudor Research Center.

M. Marshall, R. B. (2012). Emerging practices for mapping and linking life sciences data using RDF- A case series. *Journal of Web Semantics , 14*, 2-13.

M. Najafabadi, F. V. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data , 2* (1), 1.

M. Rodriguez- Muro, D. C. (2012). Quest, an OWL 2 QL reasoner for ontology- based data access.

M. Rodriguez-muro, D. C. (n.d.). Quest, a System for Ontology Based Data Access.

M. Saleem, M. A. (2015). Federated Query Processing over Linked Data. *ISWC* .

M.Poulymenopoulou, D. P. (2015). A health analytics semantic ETL service for obesity surveillance. *Studies in health technology and informatics* , 840-4.

Maryam Panahiazar, V. T. (2014, October). Empowering Personalized Medicine with Big Data and Semantic Web Technology: Promises, Challenges and Use Cases. *Proc IEEE Int Conf Big Data* , 790-795.

Marzouk, A. (2013, June). Mapping Process of Relational Schema to OWL Ontologies using XSLT. *International Journal of Computer Applications (0975-8887)* .

Max Schmachtenberg, C. B. (n.d.). *Linking Open Data Cloud diagram 2014*. Retrieved from http://lod-cloud.net/

McLeod, D. (2006). Ontology- Driven Semantics Matches between Databases Schemas. *22nd International Conference on Data Engineering Workshops (ICDEW'06)* , 6-6.

Mona Dadjoo, E. K. (2015). An Approach For Tranformating of Relational Databases to OWL Ontology. *International Journal of Web & Semantic Technology(IJWesT) , 6* (1).

N. Deshpande, R. K. (2011). Construction and Applications of ontology: Recent Trends. *DESIDOC Journal of Library & Information Technology , 31* (2), 84-89.

N. Noy, M. D. (2008). BioPortal: A web repository for biomedical ontologies and data resources. *CEUR Workshop Proceesings* , 5-6.

N. Pernelle, M. R. (2007). L2R: A Logical Method for Reference Reconciliation.

N.Cullot, R. G. (2007). DB2OWL : A tool for automatic database-to-ontology mapping. *In Proceedings of the 15th Italian Symposium on Advanced Database Systems* , 491-494.

N.Cullot, R. K. (2007). DB2OWL: A tool for automatic database-to-ontology mapping. *Proceedings of the 15th Italian Symposium on Advanced Database Systems* , 491-494.

NCOPP. (2016). Retrieved March 3, 2016, from National Coalition of Public Pathology: http://www.ncopp.org.au/site/ImportanceofPathology_.php

Noreddine GHERABI, K. A. (2012). Mapping relational database into OWL Structure with data semantic preservation. *International Journal of Computer Science and Information Security , 10*.

NREL. (n.d.). *National Renewable Energy Laboratory*. Retrieved from http://www.nrel.gov/docs/fy11osti/48798.pdf

O. Software, D. T. (2007). OpenLink Virtuoso Universal Server: Doumentation.

Optique. (2013). Scalable end user access to BIg data.

P. Haase, H. L. (2008). The neon ontology engineering toolkit. *World Wide Web* .

P. Heyvaert, A. D. (2015). Towards Approaches for Generating RDF Mapping Definitions. 1-4.

pathology. (n.d.). *medical dictionary*. Retrieved Jan 18, 2016, from Miller-Keane Encylopedia and Dictionary of Medicine, Nursing and Allied Health, Seventh Edition: http://medical-dictionary.thefreedictionary.com/pathology

R. McKerlich, C. I. (2013). Measuring use and creation of open educational resources in higher education. *International Review of Research in Open and Distance Learning* , 90-103.

R. Shearer, B. M. (2008). HermiT: A Highly - Efficient OWL Reasoner.

RDF Working Group. (2014, Feb 25). Retrieved March 5, 2016, from W3C Semantic Web: https://www.w3.org/RDF/

S. Araujo, A. V. (n.d.). Serimi results for OAEI 2011. *ISWC 2011* .

S. Auer, C. B. (2007). DBpedia: A nucleus for a Web of open data.

S. Das, S. S. (2012). R2RML: RDB to RDF Mapping Language. *W3C Recommendation*.

S. Gupta, P. S. (2015). Karma: A system for mapping structured sources into the semantic web. 430-434.

S. Hahmann, D. B. (2010). Connecting linkedgeodata and geonames in the spatial semantic web. *Proc. of GIScience 2010* .

S. Shekarpour, D. L. (2016). Question Answering on Linked Data: Challenges and Future Directions. 693-698.

S.Zhou, H. M. (2010). Ontology Generator from Relational Database Based on Jena. *Computer and Information Science , 3* (2), 263-267.

Schultz. A, M. A. (2014). Linked Data Integeration Framework : A framework for builiding Linked data Applications.

Sheridan, J. (n.d.). *Slideshare*. Retrieved from Linking UK Government Data: http://www.slideshare.net/semwebcompany/linking-uk-government-data-john-sheridan

Sheth, A. (2014, Dec). *Data Semantics and Semantic Web - 2014 year- end reflections and prognosis*. Retrieved from http://amitsheth.blogspot.ca

Sheth, A. (2013). Transforming Big Data into Smart Data: Deriving Values via harnessing Volume, Variety and Velocity using semantics and Semantic Web. *Keynote at the 21st Italian Symposium on Advanced Database Systems* .

Sheth, K. T. *Semantics-Empowered Approaches to Big Data Processing for Physical-Cyber-Social Applications*. Wright State University, Ohio Center of Excellence in Knowledge-enabled Computing, Dayton.

Shields, A. (2014, Jul 25). *Must-Know : An overview of Big Data*. Retrieved Feb 17, 2015, from Market Realist: http://marketrealist.com/2014/07/overview-big-data/

Steiner, T. (2010). How Google is using Linked Data Today and Vision For Tomorrow. *Proceedings of Linked Data in the Future Internet at the Future Internet Assembly (FIA 2010)* .

T. Bagosi, D. C. (2014). The ONTOP framework for Ontology Based Data Access. *The Semantci Web and Web science , 480*, 67-77.

T.Hasaan, R. P. (2015). Semantic HMC for big data analysis. *Proceedings- 2014 IEEE International Conference on Big Data, IEEE Big Data 2014* , 26-28.

T.R, G. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies* , 5-6.

U. Prot, M. H. (2011). A practical Guide To Building OWL Ontologies Uisng Protege 4 and CO- ODE Tools Edition 1.3.

UBM TechWeb. (2012, September). *Managing the Explosion of Medical Data*. Retrieved from http://docs.media.bitpipe.com/io_10x/io_107199/item_599973/32-EMR%20Data%20Explosion%20Whitepaper_Final.pdf

V. Haarslev, K. H. (2012). The RacerPro knowledge representation and reasonong system. *Semantic Web Journal* .

W. Hu, J. C. (2011). A self-training approach for re-solving object coreference on the Semantic Web. *In Proceedings of the 20th International World Wide Web Conference* .

W3C Recommendation. (2014, Feb 25). Retrieved Jan 15, 2016, from https://www.w3.org/TR/rdf-schema/

W3C. (n.d.). *Semantic Web*. Retrieved March 2016, from http://www.w3.org/standards/semanticweb/data

Wikipedia. (n.d.). Retrieved 2016, from https://en.wikipedia.org/wiki/Linked_data

X. Niu, S. R. (2011). Zhishi-links results for oaei 2011. *ISWC* .

Y. Chen, W. G. (2010, April). Expression of XML Implicit Semantics. *Proceedings of the Second International Symposium on Networking and Network Security (ISNNS '10)* , 15-18.

Y. Raimond, C. S. (2008). Automatic interlinking of music datasets on the semantic wen. *In proceedings of the linked Data On the Web workshop at WWW 2008* .

Y. Zhang, Y. C. (2013). A semantic approach to retrieving linking, and integrating heterogenous geospatial data. *Joint Proceedings of the workshop on AI Problems and Approaches for Intelligent Environments and Workshop on Semantic Cities - AIIP'13* , 31-37.

Yesha Mehta, D. S. (2015). Big Data Mining and Semantic Technologies: Challenges and Opportunities. *International Journal on Recent and Innovation Trends in Computing and Communication , 3* (7).