

# Exchanging Data from Institutional Repositories to the Semantic Web

by

**Humaira Farid**

(2010-NUST-MS PhD-CSE(E)-20)



Submitted to the Department of Computer Engineering in  
fulfillment of the requirements for the degree of  
**Master of Science in Computer Software Engineering**

Supervisor

**Dr. Muhammad Younus Javed**

College of Electrical & Mechanical Engineering,  
National University of Sciences & Technology, Islamabad, Pakistan

MARCH 2013

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

IN THE NAME OF ALLAH, MOST GRACIOUS, MOST MERCIFUL

## Declaration

I hereby declare that I have implemented this thesis completely on the basis of my personal efforts under the guidance and supervision of Dr. Sharifullah Khan and Dr. Muhammad Younus Javed. All the sources used in this thesis have been cited and the contents of this thesis are not plagiarized. No portion of the work presented in this thesis has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

---

Humaira Farid

## Approval

It is certified that the content and form of the thesis entitled “**Exchanging Data from Institutional Repositories to the Semantic Web**” submitted by **Hu-maira Farid** have been found satisfactory for the requirement of the degree.

Supervisor: **Dr. Muhammad Younus Javed**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member: **Dr. Sharifullah Khan**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member: **Dr. Farooque Azam**

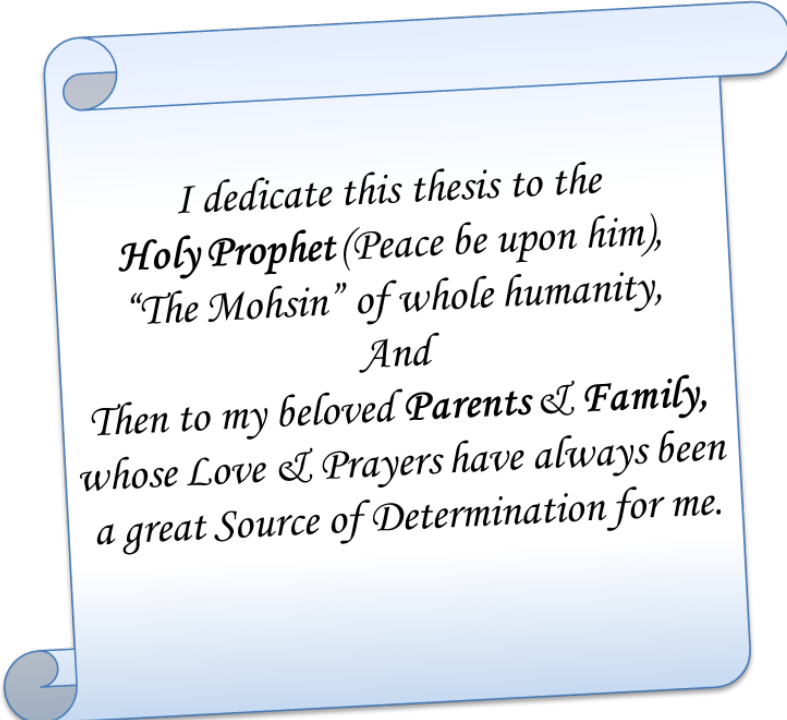
Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member: **Dr. Asia Khanum**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_



*I dedicate this thesis to the  
Holy Prophet (Peace be upon him),  
“The Mohsin” of whole humanity,  
And  
Then to my beloved Parents & Family,  
whose Love & Prayers have always been  
a great Source of Determination for me.*

## Acknowledgment

All praises are for Allah Almighty, the Most Gracious and Most Merciful, who gave me strength to complete this task. Nothing could have been possible without His blessings.

I would like to thank my supervisor **Dr. Muhammad Younus Javed** whose supervision, support and guidance helped me a lot in completing the given task. I would like to pay special thanks to **Dr. Sharifullah Khan** for his continuous technical and intellectual support, guidance and most important his precious time. His cooperation leads me to this success. I would like to appreciate **Dr. Asia Khanum** and **Dr. Farooque Azam** for serving on my committee.

I would like to admit that I owe all my achievements to my truly, sincere and most loving parents, brothers, sisters and friends who mean the most to me, and whose prayers have always been a source of determination for me. They have always supported and encouraged me to do my best in all matters of life.

Humaira Farid

## Abstract

Internet and semantic web technologies have enabled academics to find online research materials with increasing speed and accuracy. They have enabled academics to make connections with each other. Whereas, institutional repositories (IRs) are often built to serve a specific institution's community of users. Mostly existing IRs are using relational database schema for maintaining the metadata of their digital contents. They might need to interact with other information systems that build to manage institutional research activities. Thus, it is crucial to provide interoperability and integration mechanisms to bridge the gap between the semantic web and relational database worlds. To process the data in semantic context, a relational database is transformed into ontology. The use of semantic web technologies in integrating the different IRs metadata enable ontology-facilitated sharing and reuse of learning resources. They provide users access to a web of content which might otherwise require discovering and exploring multiple websites or IRs.

The main promising feature of IRs is their flexible data models that can be customized to arrange the digital documents in a repository according to the organizational structure of an institute. The data model of an organization's IR is not directly converted into IR database schema, but the data model schema is maintained as values in the comprehensive database schema of the IR. The schema of IRs databases is nested schema i.e. a schema is embedded in another schema. In other words, an IR database schema is not a normalized schema with respect to the data model, so, it makes the transformation complicated and different from the typical transformation tasks. A substantial amount of research has already been done to transform a relational database into ontology. However, these systems are only capable to transform a normalized relational database into ontology. They cannot produce accurate results if they are applied on IR databases. After building the ontologies, a key issue is to enable interoperability among different ontologies.

The proposed system first of all identifies the data model of an institute from IR database and builds a normalized relational schema for the data model of the institute. Then metadata of the repository is extracted to populate this produced schema to build an intermediate database. Once we get a normalized relational database, then relational to ontology transformation techniques are applied on this intermediate database to transform it into ontology. After that, the system transforms the instances from the generated ontology into corresponding data or instances expressed in target ontology. The classes from both source and target ontologies are extracted and simple mappings between these classes are generated by the user. Then the individuals of these mapped classes are matched and proper URIs are given to each individual. These individuals are linked with their respective target ontology classes. Finally, an RDF, having individuals of the target ontology, is generated.

The system has mainly three modules: (i) Metadata Extraction; (ii) Relation to Ontology Transformation; (iii) Ontology Alignment and Data Translation. The distinguishing features of the proposed system are (i) identifying the data model of an IR; (ii) extracting metadata of the repository; (iii) creating proper hierarchy of parent and child classes of ontology to preserve the data model hierarchy, (iv) generating mappings between ontologies, and (v) transforming data or instances from source ontology into corresponding data or instances expressed in target ontology. The system has been implemented in Java language and Jena API is used for ontology creation. Experimental results demonstrate that the transformation is correct and the system preserves information capacity.



# Contents

<b>Declaration</b>	<b>iii</b>
<b>Approval</b>	<b>iv</b>
<b>Dedication</b>	<b>v</b>
<b>Acknowledgment</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
<b>List of Figures</b>	<b>xiv</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation	1
1.2 Problem Definition	2
1.3 Proposed Solution	3
1.4 Thesis Outline	4
<b>2 Background Studies</b>	<b>5</b>
2.1 Institutional Repository	5
2.1.1 Institutional Repository Software	5
2.1.2 DSpace	6
2.1.2.1 Data Model of DSpace	7
2.1.2.2 Database Structure of DSpace	7
2.1.2.3 Repository's Metadata	8
2.2 Semantic Web	9
2.2.1 Ontology	9
2.2.2 Resource Description Framework	10
2.2.2.1 The RDF Statement (Triple)	10

2.2.3	RDF Schema	11
2.2.4	Web Ontology Language	11
2.3	Relational database to Ontology Transformation	12
2.4	Semantic Web Application	12
2.4.1	VIVO	13
2.4.1.1	VIVO Ontology	14
2.5	Ontology Interoperability	14
2.5.1	Ontology Interoperability Techniques	15
2.5.1.1	Ontology Mapping/Matching	15
2.5.1.2	Ontology Alignment	15
2.5.1.3	Ontology Merging	15
2.5.1.4	Ontology Translation	15
2.5.1.5	Ontology Transformation	16
2.5.1.6	Ontology Data Translation	16
2.6	Ontology Learning	16
2.6.1	Ontology population	17
2.6.2	Ontology enrichment	17
2.6.3	Inconsistency resolution	17
<b>3</b>	<b>Literature Survey</b>	<b>18</b>
3.1	Related Work	18
3.1.1	Metadata Extraction	19
3.1.2	Relation to Ontology Transformation	20
3.1.3	Ontology Alignment and Data Translation	24
3.2	Critical Analysis	25
<b>4</b>	<b>Proposed Methodology</b>	<b>27</b>
4.1	Introduction	27
4.2	Metadata Extraction	28
4.2.1	Identification of Data Model (IDM):	28
4.2.2	Building Schema:	30
4.2.3	Populating Intermediate Database:	33
4.3	Relation to Ontology Transformation	34
4.4	Ontology Alignment and Transformation	37
4.4.1	Resource Mappings (RM)	37
4.4.2	Instance Matching (IM)	37
4.4.3	Resource Linkage (RL)	38
4.5	Walkthrough Example	39
4.5.1	Metadata Extraction	41
4.5.2	Relation to Ontology Transformation	43

4.5.3	Ontology Alignment and Data Translation	44
<b>5</b>	<b>System Design and Implementation</b>	<b>47</b>
5.1	Introduction	47
5.2	Design and Implementation	47
5.2.1	Metadata Extraction	49
5.2.2	Relation to Ontology Transformation	51
5.2.3	Ontology Alignment and Data Translation	51
<b>6</b>	<b>System Evaluation</b>	<b>56</b>
6.1	Introduction	56
6.2	Transformation Process and its Evaluation	56
6.2.1	Operational Goals	57
6.2.2	Data Sets Specifications	57
6.3	Mathematical Proof	59
6.3.1	Lemmas	60
6.3.1.1	Lemma 1	61
6.3.1.2	Lemma 2	62
6.3.1.3	Lemma 3	64
6.3.1.4	Lemma 4	65
6.3.1.5	Conclusions of Lemma 1-4	67
6.3.1.6	Lemma 5	68
6.4	Experimental Results	69
6.4.1	Data Set 1	70
6.4.1.1	Relation to Ontology Transformation	70
6.4.1.2	Ontology Alignment and Data Translation	72
6.4.2	Data Set 2	74
6.4.2.1	Relation to Ontology Transformation	74
6.4.2.2	Ontology Alignment and Data Translation	75
6.4.3	Data Set 3	76
6.4.3.1	Relation to Ontology Transformation	76
6.4.3.2	Ontology Alignment and Data Translation	77
6.5	Comparison with Gold Standards	78
6.5.1	Lexical Precision & Recall	79
6.5.2	Taxonomic Precision & Recall	80
6.5.3	Evaluation of Relation to Ontology Transformation	80
6.5.3.1	Comparison with Astrova	81
6.5.3.2	Comparison with DataMaster	87
6.5.3.3	Comparison with D2R Server	88
6.5.4	Evaluation of Ontology Data Translation	88

<b>7 Conclusions and Future Work</b>	<b>92</b>
7.1 Discussion	92
7.2 Contribution	94
7.3 Future Work	95
<b>References</b>	<b>96</b>

## List of Figures

2.1	Usage of Open Access Repository Software - Worldwide	6
2.2	Data Model of DSpace	7
2.3	Core DSpace Model	8
2.4	An RDF statement	10
2.5	The simple RDF graph describing the color	10
2.6	Difference between mapping and transformation [1]	12
3.1	Relational.OWL Ontology	20
3.2	An architecture of RDB2ONT tool [2]	21
3.3	Application Architecture	22
3.4	R2O Transformation System	23
4.1	Main Components of the Proposed System	28
4.2	Main components of the proposed system with their key inputs and outputs	29
4.3	Work flow of data model identification	30
4.4	Work flow of intermediate database schema building	32
4.5	Work flow of intermediate database population	33
4.6	Work flow of relation to ontology transformation	35
4.7	Work flow of ontology alignment and data translation	39
4.8	DSpace data model of an institute	40
4.9	Identified Data Model after applying DMI process	41
4.10	Extracted intermediate schema after applying Schema Building rules	42
4.11	Intermediate Database after Schema Population	42
4.12	Generated Relational Database of above DSpace Structure	43
4.13	Graphical representation of the generated ontology	44
4.14	Example of Ontology Mapping	45
4.15	Individual entities for every linked author in RDF	46
4.16	Publication with corresponding linked entities in RDF	46

5.1	Class Diagram	48
5.2	Sequence Diagram for Data Model Identification	49
5.3	Sequence Diagram for Schema Building	50
5.4	Sequence Diagram for Intermediate Database Population	51
5.5	Sequence Diagram for Relation to Ontology Transformation	52
5.6	Sequence Diagram for Resource Mapping	53
5.7	Sequence Diagram for Instance Matching	54
5.8	Sequence Diagram for Resource Linkage	55
6.1	Lemma 1: Classification of Data Model entities	62
6.2	Lemma 2: Intermediate database relations for DSpace entities	63
6.3	A multilevel hierarchy of concepts created for DSpace database	67
6.4	Ontologies for data set 1	83
6.5	Ontologies for data set 2	83
6.6	Ontologies for data set 3	83
6.7	The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and Astrova [1].	86
6.8	The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and DataMaster [3].	87
6.9	The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and D2R Server [4].	88
6.10	Result Evaluation of Ontology Data Translation	90

## List of Tables

4.1	A relevant portion of DSpace Database Schema	40
6.1	The main operational goals	58
6.2	Goals fulfilled by different approaches	59
6.3	Data Set specifications	69
6.4	Data Set 1: Summary of Results - Ontology Creation	71
6.5	Data Set 1: Summary of Results - Data Translation	72
6.6	Data Set 2: Summary of Results	73
6.7	Data Set 2: Summary of Results - Data Translation	75
6.8	Data Set 3: Summary of Results	76
6.9	Data Set 3: Summary of Results - Data Translation	78
6.10	Common semantic cotopies $CSC$ and local taxonomic precision $tp_{CSC}$ for non-leaf concepts of ontologies $\mathcal{O}_{C1A}$ and $\mathcal{O}_{C1B}$	82
6.11	Common semantic cotopies $CSC$ and local taxonomic precision $tp_{CSC}$ for non-leaf concepts of ontologies $\mathcal{O}_{C2A}$ and $\mathcal{O}_{C2B}$	84
6.12	Common semantic cotopies $CSC$ and local taxonomic precision $tp_{CSC}$ for non-leaf concepts of ontologies $\mathcal{O}_{C3A}$ and $\mathcal{O}_{C3B}$	85
6.13	Evaluation of the ontologies (i.e. shown in Figure 6.4, 6.5 and 6.6) with a common semantic cotopy based measure	86
6.14	Data set 1: Evaluation of the data translation process	89
6.15	Data set 2: Evaluation of the data translation process	89
6.16	Data set 3: Evaluation of the data translation process	90

# Chapter 1

## Introduction

### 1.1 Motivation

Internet and semantic web technologies have enabled academics to find online research materials with increasing speed and accuracy. They have enabled academics to make connections with each other. Semantic Web-based network of institutional ontology-driven databases have been created to enable national discovery, networking, and collaboration via information sharing about researchers and their activities [5, 6, 7, 8, 9]. Institutional digital repositories are often built to serve a specific institution's community of users. They might need to interact with other information systems that build to manage institutional research activities to help faculty, researchers, and students for discovering common interests and make connections.

Mostly existing institutional repositories (IRs) e.g., DSpace<sup>1</sup>, EPrints<sup>2</sup>, Invenio<sup>3</sup> and Archimede<sup>4</sup> are using relational database schema for maintaining the metadata of their digital contents. Data management according to the relational data model is expected to be prevalent in the next years because it is still an order of magnitude faster than RDF data management [10]. The semantic web and relational database worlds and their developed infrastructures are based on different data models, semantics and query languages. Thus, it is crucial to provide interoperability and integration mechanisms to bridge the gap between the semantic web and relational database worlds. To process the data in semantic context, a rela-

---

<sup>1</sup><http://www.dspace.org/>

<sup>2</sup><http://www.eprints.org/>

<sup>3</sup><http://invenio-software.org/>

<sup>4</sup><http://www.bibl.ulaval.ca/archimede/index.en.html>



tional database is transformed into ontology. The fundamental goal of ontology development is the creation of an environment that allows for the controlled sharing and exchange of information among autonomous, heterogeneous data sources [11]. Ontology improves the availability of semantically rich content on the Web. The semantic web technologies provide standard representations for meaningful linkage across different sets of data. The Semantic Web Application (SWA) provides self-describing data via shared ontologies which is also readable by machines and does simple reasoning to categorize and find associations. The use of SWA in integrating the different institutional repositories metadata facilitates users' search, access, and retrieval of learning resources. The adoption and implementation of semantic web technologies enable ontology-facilitated sharing and reuse of learning resources. They provide users access to a web of content which might otherwise require discovering and exploring multiple websites or institutional repositories [8, 9].

## 1.2 Problem Definition

The main promising feature of IRs is their flexible data models that can be customized to arrange the digital documents in a repository according to the organizational structure of an institute [12, 13]. The data model of an organization's IR is not directly converted into IR database schema, but the data model schema is maintained as values in the comprehensive database schema of the IR. The schema of IRs databases is nested schema i.e. a schema is embedded in another schema. In other words, an IR database schema is not a normalized schema with respect to the data model. Therefore an IR handles equally the IR data models of various institutes without effecting its schema. As IRs are often built to serve a specific institution's community of users, a key issue is to integrate the metadata of different IRs into the semantic web-based network to help faculty, researchers, and students for discovering common interests and make connections. As mostly existing institutional repositories (IRs) are using relational database schema for maintaining the metadata of their digital contents, it is crucial to provide interoperability and integration mechanisms to bridge the gap between the semantic web and relational database worlds. The adoption and implementation of semantic web technologies enable ontology-facilitated sharing and reuse of learning resources.

A substantial amount of research has already been done to transform a relational database (i.e. schema and its data) into ontology [14, 15, 3, 2, 16, 4, 17, 18, 19]. However the existing transformation systems are only capable to transform a normalized relational database into ontology. They cannot produce accurate result if

they are applied on IR databases because their schema is nested schema not a normalized schema. Since the data model is hidden in the IR database schema, it is required to extract the data model from the IR database schema prior to transforming it into ontology, so, it makes the transformation complicated and different from the typical transformation tasks. Therefore, first of all it is essential to identify the data model of an institute from IR database and to extract its metadata and then to transform it into ontology.

After building the ontologies, a key issue is to enable interoperability among different ontologies and to integrate them into the semantic web-based network. Ontology interoperability is a key factor essential for aligning and integrating distributed ontological resources. It can be achieved by identifying or establishing semantic correspondence between entities (i.e., classes and properties) among multiple ontologies.

### 1.3 Proposed Solution

In recognition of need for integrating the metadata of institutional repositories into the semantic web, the system has been proposed and developed for integrating the metadata of different IRs into the semantic web-based network to help faculty, researchers, and students for discovering common interests and make connections.

The objective of this work is to publish metadata of institutional repositories on the semantic web applications such as VIVO<sup>5</sup> to enable sharing and reusing of existing information. The proposed system first of all identifies the data model of an institute from IR database and builds a normalized relational schema for the data model of the institute. Then metadata of the repository is extracted to populate this produced schema to build an intermediate database. After getting a normalized relational database, relational to ontology transformation techniques are applied on this intermediate database to transform it into ontology. After that instances from the generated ontology are transformed into corresponding data or instances expressed in target ontology. The classes from both source and target ontologies are extracted and simple mappings between these classes are generated by the user. Then the individuals of these mapped classes are matched and proper URIs are given to each individual. These individuals are linked with their respective VIVO classes. Finally, an RDF, having VIVO individuals, is generated. The system has mainly three modules:

---

<sup>5</sup><http://vivoweb.org/>

- a. Metadata Extraction
- b. Relation to Ontology Transformation
- c. Ontology Alignment and Data Translation

The distinguishing features of the proposed system are (i) identifying the data model of an IR; (ii) extracting metadata of the repository; (iii) creating proper hierarchy of parent and child classes of ontology to preserve the data model hierarchy, (iv) generating mappings between ontologies, and (v) transforming data or instances from source ontology into corresponding data or instances expressed in target ontology. The creation of database and ontology has been implemented in Java language and Jena API respectively. Experimental results demonstrates that the transformation is correct and the system preserves information capacity.

## 1.4 Thesis Outline

The rest of the thesis document is organized as follows: Chapter 2 describes the background knowledge of institutional repositories, semantic web and linked data. It describes the importance of Institutional Repositories (IRs), semantic web and linked data. Chapter 3 provides literature review and critical analysis of the research works that have been carried out in the field of metadata extraction, relation to ontology transformation and ontology population in order to publish IRs metadata on the semantic web.

Chapter 4 describes the proposed system in detail by describing the complete architecture of the proposed system including IRs metadata extraction, ontology creation and population of target ontology. Chapter 5 provides details about the proposed system's design, implementation and evaluation. It provides details of the datasets, metrics and measures which are used for system evaluation and results comparison. Chapter 6 finally concludes the work done in this thesis. It describes research contribution and defines future work of the thesis.

## Chapter 2

### Background Studies

#### 2.1 Institutional Repository

An Institutional Repository (IR) is an online locus for harvesting, preserving, and propagating the information in digital form for the intellectual output of an institution. According to Lynch [20], institutional repository provides a set of services for the management, preservation and distribution of digital materials created by the organization and its members.

Repositories provide services to faculty, administrators and researchers to preserve and manage research and other creative research materials. They prevent the universities and research institutions from losing their valuable information by providing new and important information sources. They enable institutions to preserve and maintain their digital material and enable them for interaction and collaboration among users in the organizations. The advantages of institutional repositories are not only for institutions and their members but they are also beneficial to other research community.

##### 2.1.1 Institutional Repository Software

Many research institutions are building repositories and sharing the descriptions of their research activities and contents. Several softwares are available for building institutional repositories. According to the Registry of Open Access Repositories

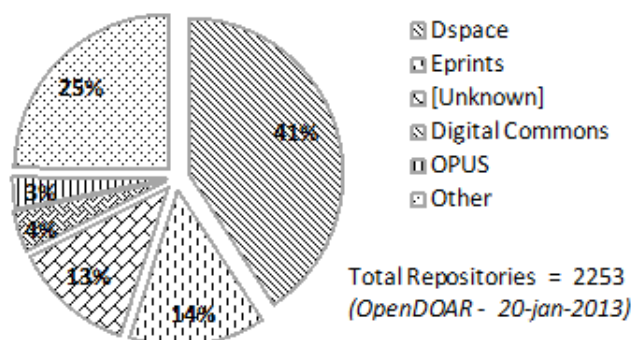


Figure 2.1: Usage of Open Access Repository Software - Worldwide

(ROAR)<sup>1</sup>, there are 3,340 and as per the Directory of Open Access Repositories (OpenDOAR)<sup>2</sup>, there are 2,253 institutional repositories in the world. Most of these repositories are built either by DSpace or Eprints which are free and open source softwares. DSpace has an obvious edge over other available open source institutional repository softwares. The Ranking Web of World Repositories<sup>3</sup> published in January 2013, provided by Webometrics<sup>4</sup>, presents that DSpace is deployed in 46 repositories among the top 100 listed repositories. The OpenDOAR statistics demonstrates that the DSpace is the most widely used open source institutional repository software worldwide, that is shown in Figure 2.1.

### 2.1.2 DSpace

DSpace[21] is an open source software which developed by MIT-Libraries (USA), in collaboration with HP-Labs (USA), in March 2002. It is used to store, index and preserve the research material of an institute in digital format. The research materials and publications are maintained in a repository to give users greater visibility and accessibility over time. Users can also customize DSpace according to the specific needs of an organization. DSpace software provides more permanent and shareable digital archiving and supports a wide variety of artifacts. These artifacts may include books, theses, and digital items, research materials, etc. DSpace was developed as an easy to use, customizable service that could maintain, preserve, and share faculty materials in digital formats. It manages and distributes digital items which are made up of digital files. It provides services for locating and retrieving the items by indexing and searching associated metadata. It supports the long-term preservation of the digital material [13].

<sup>1</sup><http://roar.eprints.org/>

<sup>2</sup><http://www.opendoar.org/>

<sup>3</sup>[http://repositories.webometrics.info/en/top\\_Inst](http://repositories.webometrics.info/en/top_Inst)

<sup>4</sup><http://repositories.webometrics.info>

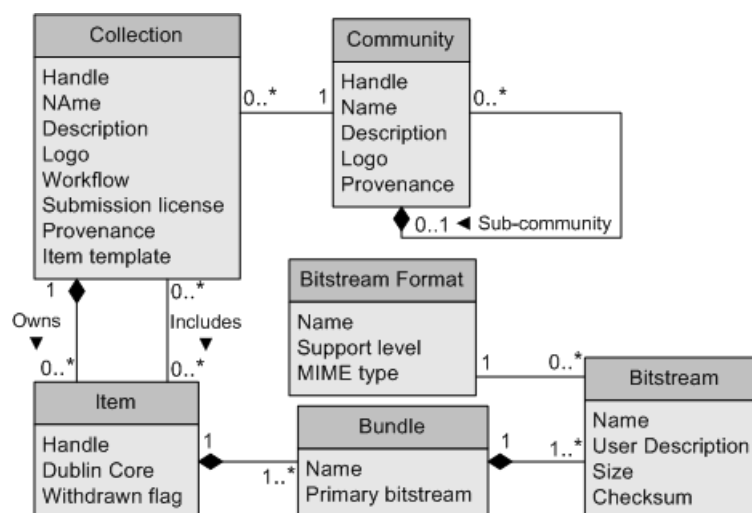


Figure 2.2: Data Model of DSpace

### 2.1.2.1 Data Model of DSpace

The data is organized in DSpace according to the structure of the organization using the DSpace system. As shown in the Figure 2.2, each DSpace site is divided into communities. These communities can be further divided into sub-communities. They reflect the institute structure of college, school, department, laboratory, or research center.

Collections come under the hierarchy of communities. The related content are grouped under these collections. Each collection contains items, which are the basic archival elements of the archive. Each item has only one owning collection. Items are further divided into bitstreams which are organized into bundles. These are ordinary computer files. Every bitstream has one associated Bitstream Format.

### 2.1.2.2 Database Structure of DSpace

DSpace uses a relational database to store the metadata of digital items and other information regarding e-people, authorization, workflows, indices, etc. DSpace has defined a database schema for its flexible data model, as shown in Figure 2.3. Recently, some features specific to PostgreSQL<sup>5</sup> and Oracle<sup>6</sup> are used by the browse indices, but after modifying some code any standard SQL database can be used. After modification of the required code, DSpace would function properly with any

<sup>5</sup><http://www.postgresql.org/>

<sup>6</sup><http://www.oracle.com/database/>

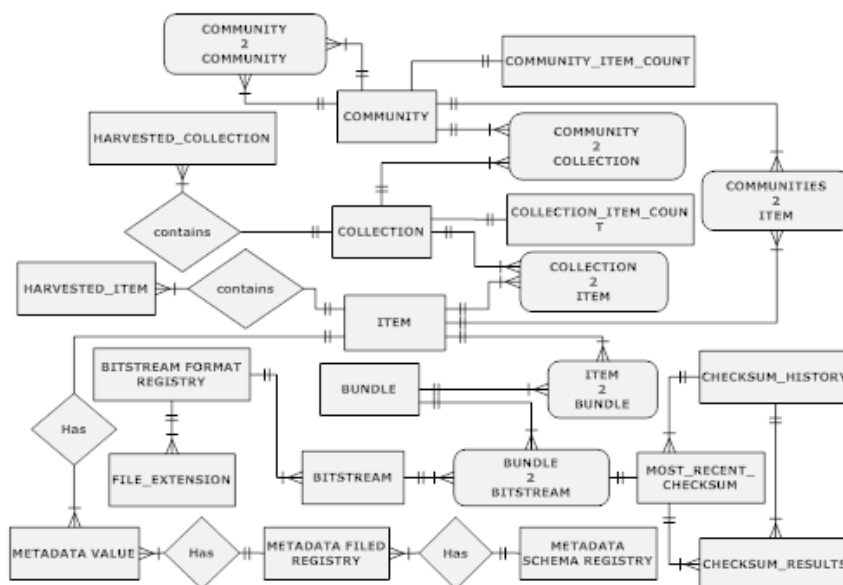


Figure 2.3: Core DSpace Model

other database back-end.

### 2.1.2.3 Repository's Metadata

Metadata provides detailed description about a digital item. Many metadata standards are designed for archival and library domains e.g. Dublin Core (DC), Data Documentation Initiative (DDI), XML Organic Bibliographic Information Schema (XOBIS), Metadata Encoding and Transmission Standard (METS), MACHine Readable Cataloging (MARC), Encoded Archival Description (EAD), Metadata Object Description Schema (MODS), etc.

DC is a most popular metadata standard which provides a set of basic elements to describe a digital item. In qualified DC, these basic elements are further described by their qualifiers. For example, a basic element “dc.contributor” can be a creator, advisor, author, etc., therefore, its qualifiers are “dc.contributor.creator”, “dc.contributor.advisor”, “dc.contributor.author”, etc. The pre-configured metadata standard with the DSpace source code is qualified DC which is used by MIT Libraries<sup>7</sup>. However, multiple metadata schemas, like DC, METS, MODS, MARC, can also be used for describing an item. Communities and collections have some simple metadata for describing their name and some other details. This simple descriptive metadata held within DSpace’s relational schema.

<sup>7</sup><http://dspace.org/technology/metadata.html>

## 2.2 Semantic Web

The Semantic Web (SW) is an extension of the current Web, also referred to as Web 3.0. In SW information is given well-defined meaning which has enabled computers and people to work in cooperation. SW is a web of data which provides associated meaning with data and enables data sharing and reuse across SW applications, enterprises and communities [22]. It allows more advanced processing for facilitating scientific communication.

SW has enabled the best possible use of the material available on the Web by developing a set of interconnecting standards. These standards are used for structuring, encoding, and sharing data. The core technology used in the SW is the Resource Description Framework (RDF). RDF is used to create a linked data system by integrating new and existing data on the web.

SW makes the right information easier to find and share among research community. It facilitates the researchers in finding scholarly material which accelerates the scientific discovery and sharing.

### 2.2.1 Ontology

Ontology is used to resolve the semantic heterogeneity. It is used to represent the data explicitly at a higher level of abstraction. It provides interoperability which is used for interaction between information systems from different sources. Ontology based integration employs ontologies instead of conceptual schemas and therefore, correspondences between source databases and one or more ontologies have to be defined. Ontology creates artifacts which can be shared among different applications. Ontologies can be extended according to the specific domains and applications. These are used with different reasoning engines and semantics of their languages are specified to facilitate this reasoning. Inference and reasoning takes the core stage in ontology integration approaches. However, a database technology is still more powerful than ontology for storing large-scale data sets. Therefore, the use of ontology is not proposed as an alternative for database. Whereas, the idea of the ontology is to support semantic interoperability between programs exchanging data. W3C has standardized several languages to define ontologies to achieve the goal of semantic interoperability. These standards can be used to define common vocabularies and structures for different applications. RDF is a prominent example which is considered as the basis for building the semantic web. In RDF, triples are



```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/2000/02/12-rdf-syntax-ns#"
  xmlns:feature="http://www.linkeddata.com/cloth-feature#">
  <rdf:Description rdf:about="http://www.linkeddata.com/clothe#shirt">
    <feature:color rdf:resource="http://www.linkeddata.com/color#white"/>
  </rdf:Description>
</rdf:RDF>

```

Figure 2.4: An RDF statement

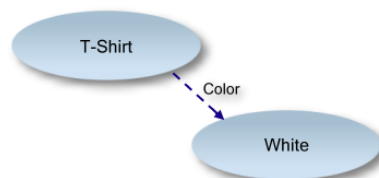


Figure 2.5: The simple RDF graph describing the color

used to represent any kind of information. Each triple states a subject-property-object relationship [23]. Web Ontology Language (OWL) is another prominent example for defining and instantiating web ontologies. The mostly considered variant of OWL is based on a Description Logic [24].

## 2.2.2 Resource Description Framework

Resource Description Framework (RDF) is used for representing the information in SW [22]. RDF uses triples to represent information, it makes **statement** about resource in the form of triple which states a **subject-predicate-object** relationship.

### 2.2.2.1 The RDF Statement (Triple)

The Figure 2.4 shows an RDF statement, also known as RDF triple. The triple breaks the statement into three parts: the subject, predicate and object. The Figure 2.5 illustrates these terms in the form of a simple RDF graph. In this graph:

- T-shirt is a **Subject**
- Color is a **Predicate** (also called property)
- white is an **Object**

### 2.2.3 RDF Schema

RDF Schema (RDFS) is basically used for augmenting the RDF by adding different constructs. These may include classes, properties, class hierarchies, property hierarchies, domain and range. For modeling these constructs the standard vocabulary is used e.g. *rdfs:Class*, *rdfs:Property*, *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:range*, *rdfs:domain*, etc.

### 2.2.4 Web Ontology Language

The Web Ontology Language (OWL) is a language which is to define and instantiate ontologies. It is used for capturing knowledge in a machine understandable way. An OWL may include the description about ontology classes, properties and their instances. OWL is a stronger language which has greater machine interpretability than RDF. An *rdf:RDF* element is a root of an OWL document because all OWL documents are basically RDF documents. The *owl:Ontology* element serves as a container for metadata about the ontology and identifies the current document as an ontology. It gives a better representation for semantics of information by providing richer set of constructs. These construct are used in building ontologies.

The richer set of constructs are provided by OWL for modeling the information. These construct are used in building ontologies which represent semantics in more effective way. Three different OWL variants are used to model the information. These variants are:

- **OWL Full:** It provides maximum expressiveness without any syntactic bounds. OWL Full includes all OWL constructs and their combination with RDF and RDFS. However, OWL Full does not provide the efficient reasoning.
- **OWL Lite:** It is a sub-language of OWL. It provides simple constraints and constructs. These constructs are used for classifying the hierarchy in ontologies. The OWL Lite provides a very restricted and minimum expressivity. However, it is most easier variant for both users and developers.
- **OWL DL:** It is a subset of OWL Full. It is more expressive than OWL Lite but still less expressive than OWL Full. It supports high expressiveness without providing syntactic freedom. OWL DL supports the efficient reasoning as compare to OWL Full.

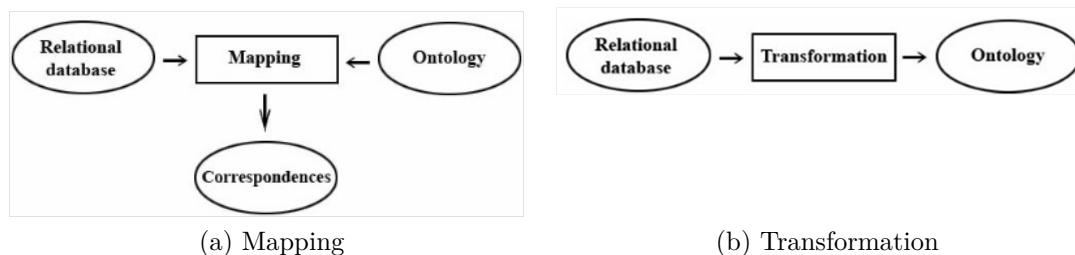


Figure 2.6: Difference between mapping and transformation [1]

## 2.3 Relational database to Ontology Transformation

The difference between transformation and mapping is explained by Astrova in [1]. The mapping process assumes that both a relational database and an ontology exist. It produces a set of correspondences between source relational database and target ontology. Therefore, both relational database and ontology are used as input in mapping process. The output of this process is a set of correspondences which provides links between the constructs of relational database and ontology. Whereas, in case of transformation only a relational database exists. The transformation process produces a new ontology from the relational database. Therefore, the transformation process takes only a relational database as input. The output of this process is an ontology. Figure 2.6 illustrates the difference between transformation and mapping.

A relational model is used for implementing the relational database. This model includes constructs which specifies tables, columns, data types, and other constraints and semantics of the relational database. Whereas, an ontological model is used for implementing an ontology. The constructs of an ontological model are used to specify ontology classes, data types, properties, inheritance, and other semantics. Transformation of relational databases into ontologies is based on a set of rules. These rules are used to specify the mappings between the constructs of the relational model and the ontological model. For example, a table of relational database is mapped to a ontology class, a column of a table is mapped to a data type property of respective ontology class, etc.

## 2.4 Semantic Web Application

The semantic web technologies provide standard representations for meaningful linkage across different sets of data. The Semantic Web Application (SWA) allows data

to be shared and reused across applications by providing a common framework. SWA provides self-describing data via shared ontologies which is also readable by machines and does simple reasoning to categorize and find associations.

The use of SWA in integrating the different institutional repositories meta-data facilitates users' search, access, and retrieval of learning resources. The adoption and implementation of semantic web technologies enable ontology-facilitated sharing and reuse of learning resources.

### 2.4.1 VIVO

VIVO provides a semantic web-based network of institutional ontology-driven databases. It enables networking, national discovery, and collaboration by sharing the information about researchers and their activities. The VIVO project was started at Cornell University and further developed by an National Institutes of Health<sup>8</sup> funded consortium, it is now being established as an open-source project with community participation from around the world.

The results are clustered into different categories e.g. activities, people, organization, events, publications, etc. These categories are used by search engine of VIVO. VIVO is populated with detailed profiles of faculty and researchers including information such as publications, teaching, service, and professional affiliations. Bi-directional hyperlinks are presented by each page in VIVO which provide users access to a web of content which might otherwise require discovering and exploring different websites or institutional repositories [8, 9]. VIVO also provides:

- information about research and researchers - their scholarly works, research interests, and organizational relationships.
- an expressive ontology and tools for managing the ontology.
- a platform for using the ontology to create and manage linked open data for scholarship and discovery.
- a platform for connecting scholars, communities, and campuses using linked open data.
- a support for students to find appropriate research projects and advisors

---

<sup>8</sup><http://www.nih.gov/>

By the end of 2012, over 20 countries and 50 organizations provide information in VIVO format on more than one million researchers and research staff, including publications, research resources, events, funding, courses taught, and other scholarly activity.

#### **2.4.1.1 VIVO Ontology**

VIVO produces Resource Description Framework (RDF) triples formatted according to a published ontology so that information can be exchanged, aggregated and searched by others on the web through standard protocols. An ontology communicates the meaning of these RDF triples by defining types (classes) and the relationships between them (properties); the VIVO ontology is a unified, formal, and explicit specification of information about researchers, organizations, and the activities and relationships that link them together. The VIVO ontology concentrates on modeling scientists in the rich context of their activities, organizations and the products of their research rather than the knowledge in any single domain of science. A key issue in building the VIVO ontology is how to link to external controlled vocabularies in a standard way and enabling interoperability among different ontologies. Since heterogeneous ontologies have been developed in different contexts, ontology interoperability is a key factor essential for aligning and integrating distributed ontological resources. Ontology interoperability can be achieved by identifying or establishing semantic correspondence between entities (i.e., classes and properties) among multiple ontologies [25].

## **2.5 Ontology Interoperability**

In the SW ontologies play a very important role. They are key elements in many applications. These application may use for retrieving information, searching and composing web services. It is required for combining distributed and heterogeneous ontologies which enables people and software agents to work in a more smooth and collaborative way. Therefore, it is important to enable interoperability among different ontologies. If the interoperating applications are used to share the ontologies, they enable the exchange of data both on syntactic and semantic level.

## **2.5.1 Ontology Interoperability Techniques**

Ontologies have emerged as the best means for explicitly describing semantics and contexts of data to be shared among different organizations and information systems. A key challenging issue in building the Semantic Web is to allow the interoperability among these different ontologies. This issue can be handled by ontology mapping, alignment, merging, translation, transformation and data translation processes [26].

### **2.5.1.1 Ontology Mapping/Matching**

Ontology Mapping is the process of identifying correspondences and relationships among entities of different ontologies. The mapping process only produces a set of correspondings without modifying the ontologies.

### **2.5.1.2 Ontology Alignment**

Ontology alignment is the process of describing a semantic bridge between two (or more) ontologies through a set of matches between them. It makes these ontologies consistent and coherent with one and another. In other words, the output of the matching process is alignment.

### **2.5.1.3 Ontology Merging**

Ontology merging takes place once the mappings between the two ontologies are identified, the mapped or aligned concepts are merged into a single one. This process creates a new ontology from source ontologies without modifying existing ontologies.

### **2.5.1.4 Ontology Translation**

It is the process which converts an ontology from one language to another without altering the ontology semantics.

### 2.5.1.5 Ontology Transformation

Ontology transformation is the process which changes the structure of an ontology for presenting the entities of an ontology according to the entities of another ontology. It slightly modifies the semantics of an ontology for making it appropriate for different purposes.

### 2.5.1.6 Ontology Data Translation

It is the process of transforming instances or data from one ontology into corresponding instances or data presented in target ontology.

## 2.6 Ontology Learning

Ontology learning is the process of constructing, evolving or integrating an ontology. This process can be performed by using the following approaches:

- Integration of existing ontologies by identifying the similarities among ontologies. This can be done by:
  - creating a single ontology by merging of different ontologies, and
  - establishing links among ontologies and enabling them to share and reuse their information by using alignment,
- Constructing a new ontology.
- Extending or evolving an existing ontology
- Specialization of a generic ontology according to a specific domain.

The ontology learning involves the population and enrichment of an ontology. The ontology learning requires to resolve inconsistencies introduced by population and enrichment tasks.

### **2.6.1 Ontology population**

It is the process of adding new instances into the concepts of an existing ontology. It does not extend ontology by adding new concepts and properties. Therefore, it does not modify the structure of an ontology so, the non-taxonomic and hierarchical relations remain unaltered.

### **2.6.2 Ontology enrichment**

This process extends an ontology by adding new concepts and properties. Therefore, it modifies the structure of an ontology. Mostly the existing domain knowledge is insufficient to explain the extracted information. Ontology enrichment plays an important role for extending the background and domain knowledge which provides the better explanation of extracted information and its semantics.

### **2.6.3 Inconsistency resolution**

The tasks of ontology population and enrichment introduce some inconsistencies in an ontology. Ontology will be populated with the redundant instances if it is populated without checking whether that instance is already existing in the ontology. Therefore, consistency maintenance is an important process in ontology learning which eliminates the redundant instances. This can be performed automatically by instance matching or domain expert can also perform this task.



## Chapter 3

### Literature Survey

#### 3.1 Related Work

The task of transforming DSpace database to a local ontology and its mapping and translation for populating data in VIVO can be generally divided in three steps.

- a. Extracting Repository's Metadata
- b. Relation to Ontology Transformation
- c. Ontology Alignment and Data Translation

Moreover, the work needs to evaluate the methodology to verify that information is not lost and preserved during the transformation.

Therefore the survey is divided in to three categories.

- a. Metadata Extraction: In this category a literature review of the research works is provided which has been carried out in the field of metadata extraction.
- b. Relation to Ontology Transformation: The papers in this category discuss how to build ontology from relational database. Existing methodologies build ontology from DDL scripts or relational tables (i.e., from metadata).
- c. Ontology Alignment and Data Translation: In this category a literature review of the research works is provided which has been carried out in the field of

ontology alignment and population in order to publish data on the semantic web.

### 3.1.1 Metadata Extraction

There are various techniques exist that extract metadata from deep web, static web and legacy databases using reverse engineering techniques.

The techniques proposed by [27, 19] build a conceptual model (i.e., extended entity-relationship diagram) from database tables. The techniques are suitable for data sources which have little descriptions about the fields in their tables and they have no description for keys. Maatuk et al. [28] present an approach to semantic enrichment for relational DataBase Migration (DBM). In this technique necessary data semantics about a given relational database (RDB) are extracted and enhanced to produce Relational Schema Representation (RSR). The RSR constructs are then classified to develop a Canonical Data Model (CDM), which provides a description of the existing RDB's implicit and explicit semantics. The generated CDM is a sound source of semantics and is a well organized data model, which facilitates an integrated approach to DataBase Migration (DBM). Gherabi et al. [29] also proposed a similar solution for migrating relational database into Web semantic. It takes an existing RDB as input, and extracts its metadata representation (MTRDB). Then a CDM is generated based on this extracted MTRDB. Finally, the structure of the classification scheme in the CDM model is converted into OWL ontology.

Nagy et al. [30] present a method for extraction of data from tables. They transform the tables into a relational database which is accessible by both SQL and SPARQL for relational tables and RDF triple stores respectively. They primarily focused on large statistical information sites. These sites are generated from databases without having direct access to the back-end databases. Kappel et al. [31] proposed a process for lifting metamodels into ontologies for the purpose of integrating modeling languages semantically. In this way, the concepts hidden by these modeling languages are represented explicitly by transforming a metamodel into an ontology.

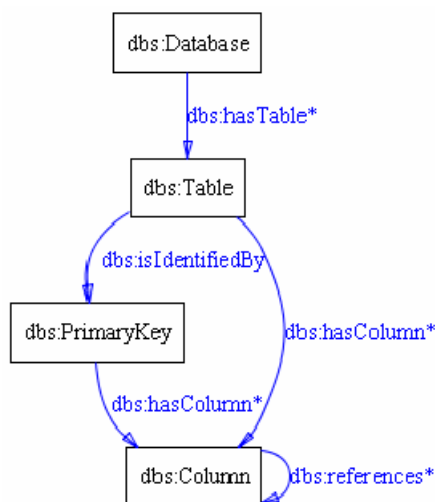


Figure 3.1: Relational.OWL Ontology

### 3.1.2 Relation to Ontology Transformation

A substantial amount of research has already been done in the field of relational to ontology transformation and many approaches are available. Due to the large scope and amount of related work, some significant existing solutions are discussed in this section.

**Relational.OWL** [14] is a most popular and dominant work in the generation of a database schema ontology. The ontology used by Relational.OWL is shown in Figure 3.1. It defines four classes *Database*, *Table*, *Column* and *PrimaryKey*. The instances of these classes and their relationships can represent the schema structure of any relational database. It transforms every relation and attribute in a relational model into corresponding instance of meta-classes *Table* and *Column* respectively. OWL properties are defined to describe the relationship among these classes, such as the *hasColumn* property connecting instances of the *Table* and *Column* classes and the *references* property describing foreign key links. This representation of components and relationships is called Relational.OWL. *owl:equivalentClass* or *owl:equivalentProperty* relationships can be used to link Relational.OWL with the similar representations based on RDF or OWL. Every tuple of a relation is viewed as an instance of a schema representation class, while tuple values are viewed as values of properties-instances of the *Column* class. This lack of separation among classes, properties and individuals makes the produced ontology OWL Full. Relational.OWL is also used in other tools and approaches (e.g. ROSEX [15] and DataMaster[3]).

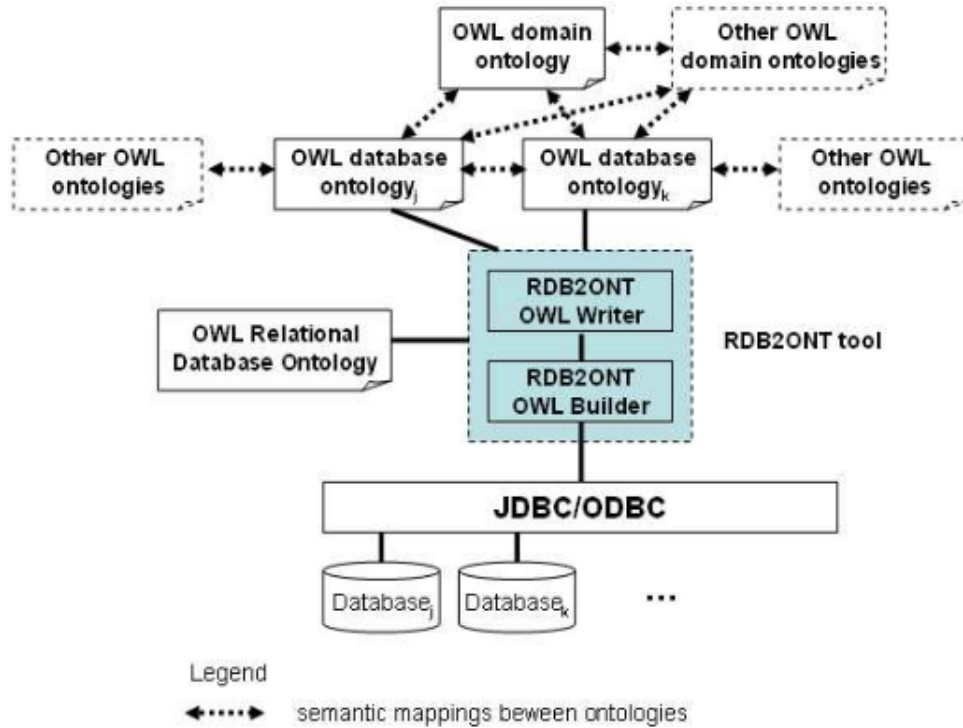


Figure 3.2: An architecture of RDB2ONT tool [2]

**DataMaster** [3] is developed as a plug-in for Protégé<sup>1</sup>. The novelty of DataMaster is to offer two alternative modeling versions of Relational.OWL that manage to stay within the syntactic bounds of OWL DL by providing the separation among classes, properties and individual. One alternative modeling version the schema representation layer of Relational.OWL is completely omitted and by that means database relations and attributes are translated to OWL classes and properties respectively. *hasXSDDType* property is attached to the *Column* class. In the second alternative, data representation layer is missing and *ColumnType* class and *hasColumnType* property is defined that relates *Column* instance to *ColumnType* instance.

**ROSEX** [15] also uses a little modified version of the Relational.OWL ontology to transform the relational schema of a database as an OWL ontology. It extracts the Data Source Ontology from the relational database schema. The generated database schema ontology is mapped to a domain-specific ontology. It is generated automatically by reverse engineering the database schema. This mapping is used to translate SPARQL queries expressed over the domain ontology to SQL queries expressed over the relational database.

Another similar approach to Relational.OWL is **Relational DataBase-**

<sup>1</sup>Protégé is a free and open-source Ontology Editor from Stanford University (<http://protege.stanford.edu/>)

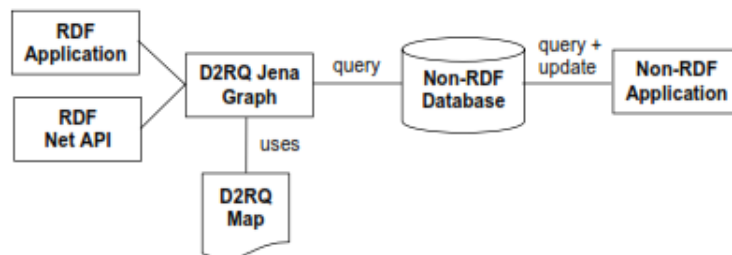


Figure 3.3: Application Architecture

**to-ONTology (RDB2ONT)** [2] that produces a new database schema OWL ontology. It provides a framework for the semantic interoperability between relational databases by generating OWL Database Ontologies. They describe relational database systems in a standardized way in OWL. In the generated ontology the schema of the underlying database systems can be presented at various levels of abstractions. There are two main components, (a) the OWL Builder, (b) the OWL Writer. OWL Builder builds an internal common model by extracting the schema and constraints from the relational database and generates then an OWL ontology. The generated ontology is created as an instance of the OWLRDBO. OWL Writer is in charge for writing these ontologies to the output file. It provides the flexibility to choose namespace URIs for these ontologies. A framework is presented in Figure 3.2. Two databases are used; (a) *Database<sub>j</sub>*, and (b) *Database<sub>k</sub>*, described by database ontologies; *OWL Database Ontology<sub>j</sub>* and *OWL Database Ontology<sub>k</sub>* respectively. These ontologies are automatically generated. The domain experts create domain ontologies, which describe the concepts, properties, and their semantic relationships. The defined classes and properties follow the same meta-modeling paradigm as in Relational.OWL, but the attempt to model relational schema constraints as OWL constraints is erroneous. The contents of a relation represent relational schema constraints and these constraints are translated by applying them on the schema-level OWL classes, however, this effort is unsuccessful to avoid OWL Full.

**D2RQ** [16] is one of the most prominent tools which generates domain-specific RDFS ontology by exporting the contents of a relational database. A declarative custom mapping is specified for describing mappings between relational database schema and RDFS ontology. User can also modify the automatically generated mapping. It considers non-RDF relational databases as virtual RDF graphs. These virtual graphs can be accessed and queried using RDQL. The architecture of D2RQ usage scenario is shown in Figure 3.3. Non-RDF legacy application is used to maintain a relational database. The database content can be accessed via an RDF application by using D2RQ. The engine that uses D2RQ mappings to translate requests from external applications to SQL queries on the relational database is called

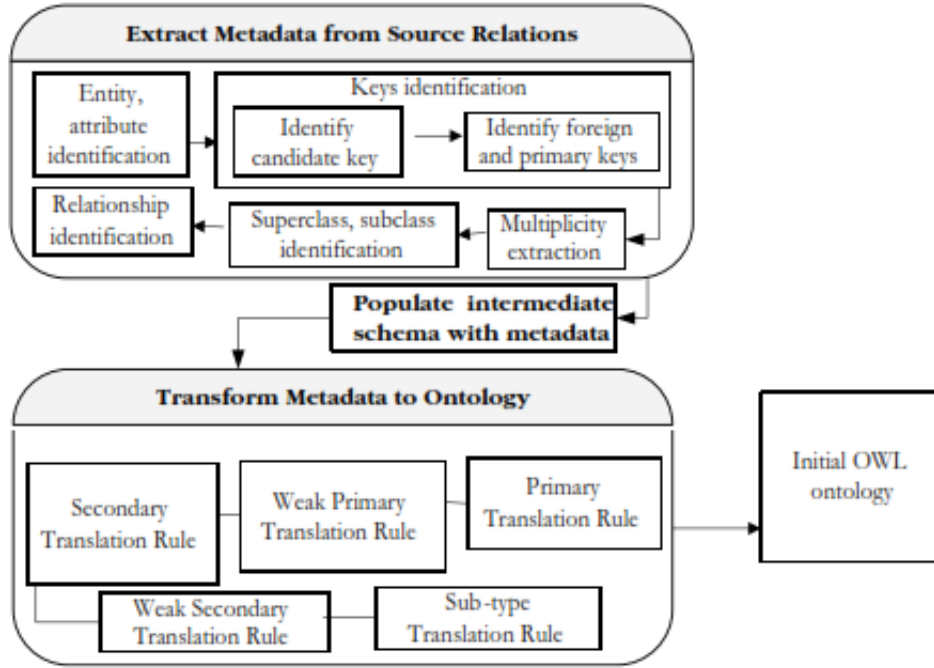


Figure 3.4: R2O Transformation System

D2R Server [4]. D2R Server is in charge for publishing the relational databases content on the SW. A declarative mapping generated by D2RQ is used to map database content to RDF. The mapping specifies the way to identify resources and to generate property values from database content.

**OpenLink Virtuoso Universal Server** is an integration platform that offers an RDF view over a relational database with its RDF Views feature [17]. Virtuoso RDF Views depict existing relational data as virtual RDF graphs. These graphs are created without physically regenerating the relational data as RDF data sets. Virtuoso Server supports both automatic and manual operation modes. In the former, an RDFS ontology is created using the common transformation rules. Latter on a mapping expressed in the proprietary Virtuoso Meta-Schema language is manually defined. Complex mapping cases can also be covered by this mapping. Virtuoso’s mapping language allows assigning any subset of a relation to an RDFS class. The pattern of the generated URIs can also be defined by using this mapping language.

Another prominent work in RDFS extraction from relational schema is **R2O** [19]. R2O transformation system comprises two parts, as depicted in Figure 3.4. In the first part entities, attributes, keys and cardinalities are identified. The cardinalities are then used to identify super-classes and subclasses. In the second part, five transformation rules are designed which cover all types of relations. After analyzing extracted relations, ontology is created by using designed transformation rules.

These rules are designed for primary, secondary, weak secondary and sub-type relations. All attributes, keys, hierarchies and cardinalities are also converted to OWL ontology. But R2O only handles issues in transforming metadata to ontology, it does not transform database contents.

### 3.1.3 Ontology Alignment and Data Translation

COMA++ [32] is a tool for matching schemas to determine similarities between them. It uses different characteristics of schemas for identifying these similarities. These characteristics may include the names of the schema elements and their data types, structural information, etc. Instance-based matching has been proposed for COMA++ in [33]. It supports different sources such as relational schemas, XML and ontologies for importing instance data. Different parsers have been used for parsing schemas and the instance data from different sources. The same generic data representation is used for representing different instance data. Two instance-based matchers are used in this approach. These matchers are content and constraint based matchers which use several linguistic approaches and constraints. They generate similarity matrices which are combined for deriving a mapping between schema elements.

Araujo et al. [34] proposed a tool, SERIMI<sup>2</sup>, which interlinks the datasets published in the Linked Data Cloud. It focuses in the instance matching problem over RDF data and matches instances between a source and a target data sets without prior knowledge of the data, domain or schema of these data sets. This solution is composed of two phases: the selection phase and the disambiguation phase. It uses existing string matching algorithms for solving the selection phase. A new function of similarity is used during the disambiguation phase, which operates even without having any direct ontology alignment between the source and target data sets.

An ontology based information extraction system, BOEMIE (Bootstrapping Ontology Evolution with Multimedia Information Extraction)<sup>3</sup>, has been developed for extracting objects automatically from various media types such as images, video, audio, text, etc [35]. BOEMIE also proposed ontology population methodology. Castano et al. [36] developed an instance matching component HMatch(I) of the HMatch 2.0 ontology matching suite [37]. This component is developed in the framework of the BOEMIE project which provides instance level matching techniques for

<sup>2</sup><https://github.com/samuraraujo/SERIMI-RDF-Interlinking>

<sup>3</sup>Funded by the European Commission, see <http://www.boemie.org>

supporting ontology population tasks. It supports both syntactic and semantic approaches for instance matching. It also has a capability for identifying featuring and non-featuring properties. It applies statistical techniques for learning information about featuring properties. These featuring properties highly contribute for identifying individuals.

Knoblock et al. [38] developed a web application, Karma<sup>4</sup>, which allows users to integrate data from various sources. Data can be extracted from relational databases, XML files or CSV files and this data can be published in different formats i.e. CSV or RDF. Conditional Random Fields (CRF) [39] are used for learning semantic types and the relationships among the source's schema elements are identified by using Steiner tree algorithm. RDF generated by the Karma can be loaded into VIVO by creating proper URIs according to the VIVO ontology. An initial mappings are generated semi-automatically which can be refined and modified by users.

## 3.2 Critical Analysis

The existing metadata extraction techniques [27, 19, 28, 29, 30, 40, 31] are domain and application oriented and apply reverse engineering techniques for extracting data. They cannot be applied in different domains and applications.

The existing relation to ontology transformation systems [1, 41, 42, 19, 43, 15, 14, 3, 4] cannot produce accurate result if they are applied on an IR database because its schema is nested. IR database relation cannot be mapped to RDFS class because RDFS class has to be created for data model entities which are attributes of relations. Therefore, it is required to extract the data model from the IR database schema prior to transforming it into ontology. The existing transformation systems are capable to transform a normalized relational database, however, DataMaster [3] and D2R server [4] are not creating proper hierarchy of parent and child classes so they are not able to preserve the IR data model hierarchy. R2O [19], DM-2-OWL [43] and ROSEX [15] are not populating ontology with instances from the database. Astrova [1], DB2OWL [42], DM-2-OWL [43], RDBToOnto [41], ROSEX [15], DataMaster [3] and D2R server [4] are not handling the multivalued attributes appropriately during transformation which can cause loss of information while integrating into other ontologies.

---

<sup>4</sup><https://github.com/InformationIntegrationGroup/Web-Karma-Public>



Engmann et al.[33] proposed technique for instance matching which does not populate ontology with new instances. Araujo et al. [34] proposed a tool which links instances without populating ontology. However, Castano et al. [36] developed a instance based matching component which is used for ontology population but this is a domain specific approach which cannot be applied in any other domain. The most related technique is proposed by Knoblock et al. [38] but it does not allow to extract data from ontologies. It uses relational databases, XML files and CSV files for data extraction. It requires user intervention for generating mappings which makes it more time consuming and error-prone task.

## Chapter 4

### Proposed Methodology

#### 4.1 Introduction

The proposed system has been designed for integrating the institutional repositories metadata into the semantic web by transforming them into ontology. The proposed approach extracts the metadata, creates a normalized intermediate database and transforms it into ontology. After that instances from the generated ontology are transformed into corresponding data or instances expressed in target ontology. The classes from both source and target ontologies are extracted and simple mappings between these classes are generated by the user. Then the individuals of these mapped classes are matched and proper URIs are given to each individual. These individuals are linked with their respective VIVO classes. Finally, an RDF, having VIVO individuals, is generated. The proposed system is divided into three main modules:

- a. Metadata Extraction,
- b. Relation to Ontology Transformation
- c. Ontology Alignment and Data Translation.

The main components of the system are shown in Figure 4.1. The key inputs and outputs of these components are shown in Figure .

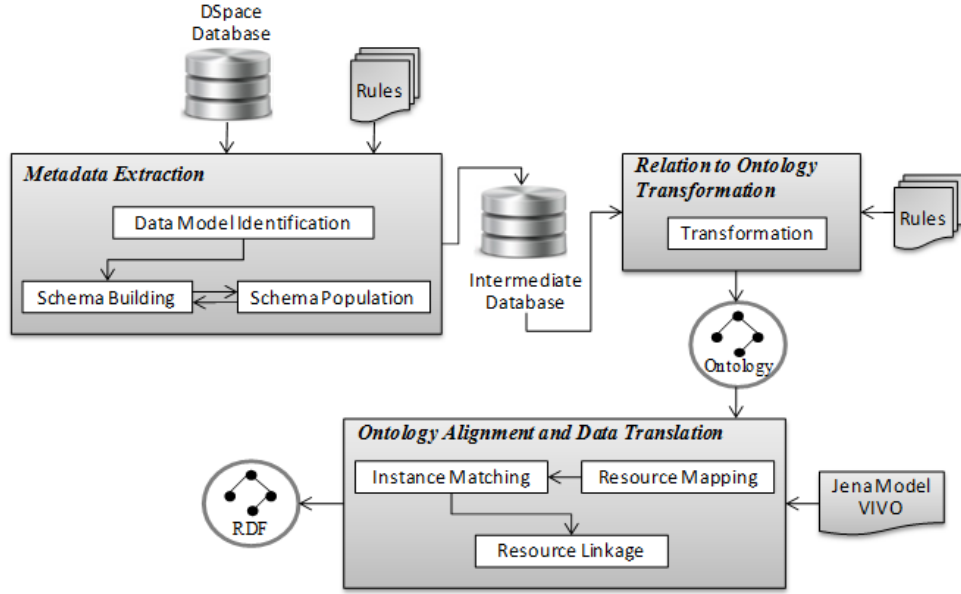


Figure 4.1: Main Components of the Proposed System

## 4.2 Metadata Extraction

In this module a normalized schema of an IR data model and its data is obtained by extracting metadata from the nested schema of an IR database. This module comprises three sub-modules as discussed below.

### 4.2.1 Identification of Data Model (IDM):

DSpace maintains the Data Model ( $DM$ ) information of an institute in its database. In this component, data is extracted from DSpace database to identify the specific  $DM$  of an institute.  $DM$  contains a set of communities  $CM$  and a set of collections  $CL$  and  $CL_i \in CL$ . Every  $CL_i$  is a set of items  $I_j$ .  $I_j$  is a basic archival element of a repository. A community can be a main community  $CM_{m_i}$  or a sub-community  $CM_{s_i}$ . So  $CM$  is (i)  $CM_m$  and  $CM_{m_i} \in CM_m$ ; and (ii)  $CM_s$  and  $CM_{s_i} \in CM_s$ .  $CL_i$  is subsumed by either  $CM_{s_i}$  or  $CM_{m_i}$ . The detailed procedure is described in Algorithm 4.1. The component extracts the identifier of an item (i.e. document) from the *item* table. Then the names of its respective collection and parent communities are identified. The process is repeated for all the items. In this way the names of communities, sub-communities and their respective collections are identified.

Let  $E(DM)$  denotes the set of all entities of some DSpace data model  $DM$  and  $E_i \in E(DM)$ . The category of  $E_i$  is identified, such that  $\forall E_i \in (CM_m \vee CM_s \vee CL)$ .

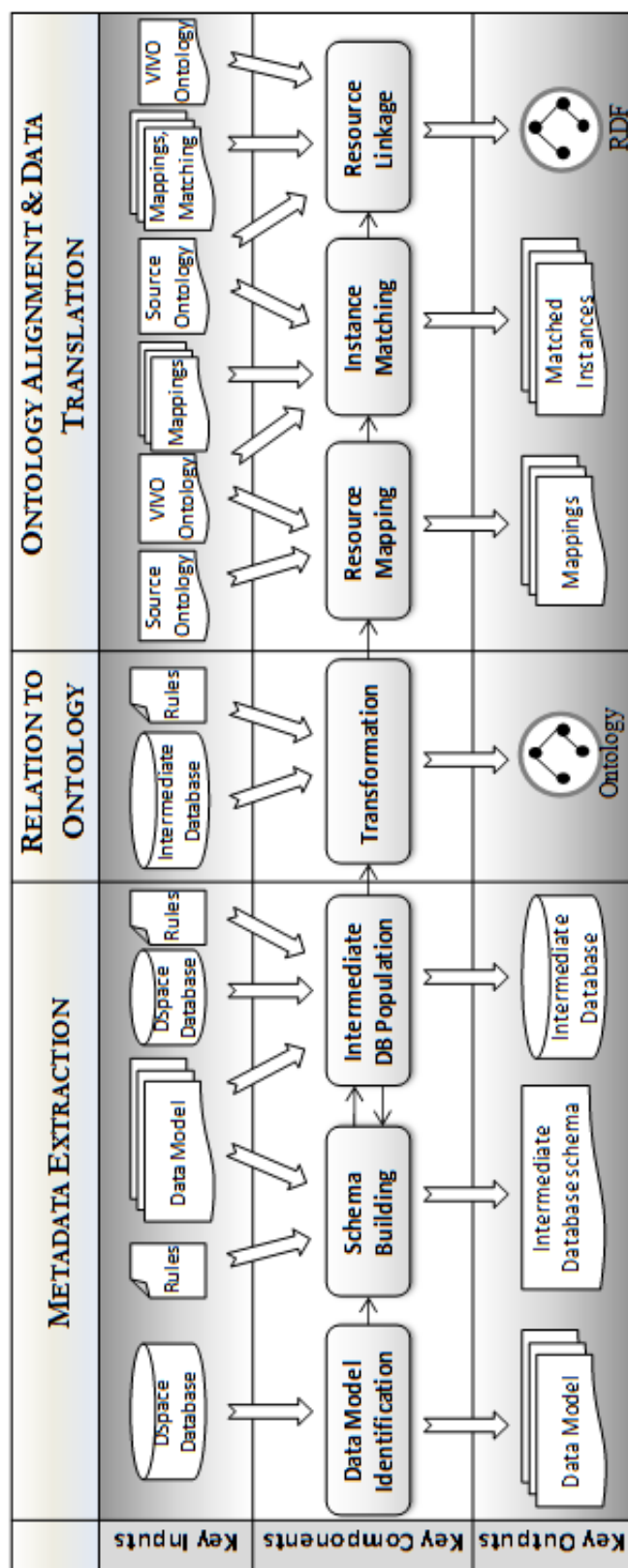


Figure 4.2: Main components of the proposed system with their key inputs and outputs

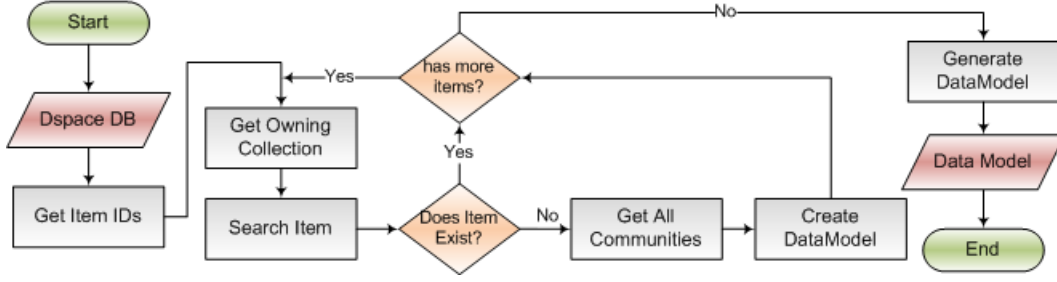


Figure 4.3: Work flow of data model identification

Two properties, *hasChild* and *hasParent*, are defined for explaining conditions. Following rules are used to identify correct category of  $E_i$ :

**Rule 1:**  $E_i$  is a main community  $CM_{m_i}$  iff (i)  $E_i$  is a subsumer and (ii)  $E_i$  is not a subsumee. It means that the entity  $E_i$  has no parent but it has child(ren), such that

$$E_i \in CM_m \iff \neg(E_i \sqcap \forall hasChild.\perp) \wedge (E_i \sqcap \forall hasParent.\perp)$$

**Rule 2:**  $E_i$  is a sub-community  $CM_{s_i}$  iff (i)  $E_i$  is a subsumer and (ii)  $E_i$  is also a subsumee. It means that the entity  $E_i$  has both parent and child(ren), such that

$$E_i \in CM_s \iff \neg(E_i \sqcap \forall hasChild.\perp) \wedge \neg(E_i \sqcap \forall hasParent.\perp)$$

**Rule 3:**  $E_i$  is a collection  $CL_i$  iff (i)  $E_i$  is not a subsumer, (ii)  $E_i$  is a subsumee and (iii)  $E_i$  is not an empty set, it contains all items owned by  $CL_i$ . It means that the entity  $E_i$  has a parent but it has no child(ren), such that

$$E_i \in CL \iff (E_i \sqcap \forall hasChild.\perp) \wedge \neg(E_i \sqcap \forall hasParent.\perp) \wedge (E_i \neq \emptyset)$$

The detailed process of the component is described in Algorithm 4.1 and its work flow is shown in Figure 4.2. These identified entities and their relationship represent the data model of an institute, as shown in Figure 4.8.

#### 4.2.2 Building Schema:

In building of schema (BS), the extracted data model is used for creating the schema of an intermediate database. Attributes of the intermediate database relations are extracted from the DSpace Database. Following rules are used for SB:

---

**Algorithm 4.1** Identifying DSpace Data model
 

---

```

1. START
2. INPUT: DSpace Database
3. VAR : item_ids,parent_communities : ArrayList; exists:Boolean
4. item_ids  $\leftarrow$  Get item IDs from item table;
5. FOR  $i \leftarrow 0$  to  $i < \text{item\_ids.size}$ 
6.   id  $\leftarrow$  item_ids[i], col_id  $\leftarrow$  Get item's owning Collection ID
7.   col_name  $\leftarrow$  Get Collection Name from collection table
8.   exists  $\leftarrow$  search Item in intermediate database
9.   IF item exists in intermediate database THEN
10.     $i \leftarrow i+1$ ; //no need to get its hierarchy, search next item
11.    Go to step 6
12.  END IF
13.  com_id  $\leftarrow$  Get CommunityID from community2collection table
14.  com_name  $\leftarrow$  Get CommunityName from community table
15.  parent_communities  $\leftarrow$  Get all its Parent Communities
16.  Create DataModel using above extracted information
17.END FOR
18.OUTPUT: DSpace Data Model
19.END

```

---

**Rule 4:** A database relation  $R_{CM_{m_i}}$  is created for each  $E_i \in CM_m$ .

**Rule 5:** A child relation  $R_{CM_{s_i}}$  of respective parent relation  $R_{CM_{m_i}}$  is created for  $E_i \in CM_s$ .

**Rule 6:** If any sub-community  $CM_{s_j}$  of  $CM_{s_i}$  exists, a child relation  $R_{CM_{s_j}}$  of  $R_{CM_{s_i}}$  is created for  $CM_{s_j}$ .

**Rule 7:** A database relation  $R_{CL_i}$  is created for each  $E_i \in CL$ .  $R_{CL_i}$  is created as a child relation of its parent entity.

**Rule 8:** Attributes *atts* of the relation  $R_{CL_i}$  are created from metadata of the item  $I_j$  that is extracted from the contents of the *metadata:fieldregistry* table of DSpace Database.

The detailed process of the this component is described in Algorithm 4.2 and its work flow is shown in Figure 4.3.

---

**Algorithm 4.2** Building Schema of Intermediate Database
 

---

```

1. START
2. INPUT: DSpace Database, DSpace Data Model
3. VAR : communities : ArrayList; exists : Boolean;
4. communities  $\leftarrow$  Get Communities from identified data model
5. FOR  $i \leftarrow 0$  to  $i < \text{communities.size}$ 
6.   com  $\leftarrow$  communities[i];
7.   exists  $\leftarrow$  search table in intermediate database
8.   IF table exists in database THEN
9.      $i \leftarrow i+1$ ; //search next community name
10.    Go to step 6
11.  END IF
12.  Create table against that community
13.END FOR
14.exists  $\leftarrow$  search table in intermediate database against collection
15.IF table does not exist THEN
16.  Create table against that collection
17.END IF
18.metadata_fields  $\leftarrow$  Get Metadata Fields
19.Create attributes against extracted Metadata Fields
20.OUTPUT: Intermediate Database Schema
21.END

```

---

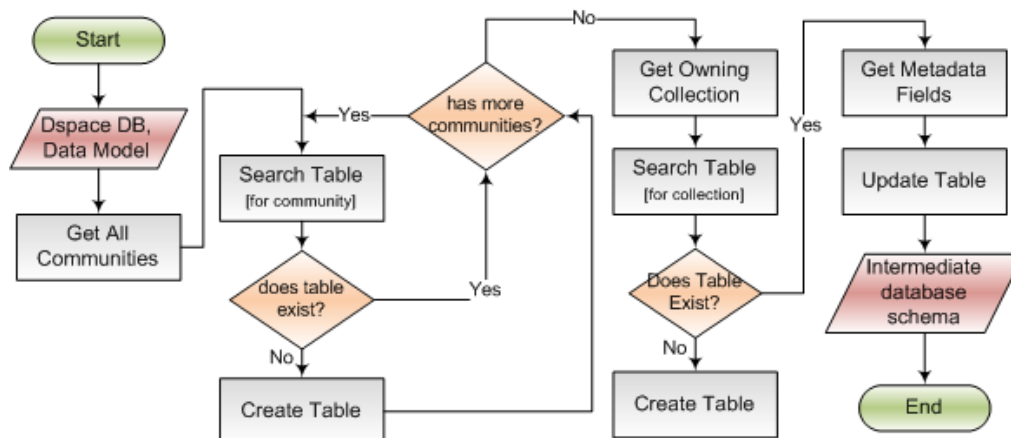


Figure 4.4: Work flow of intermediate database schema building

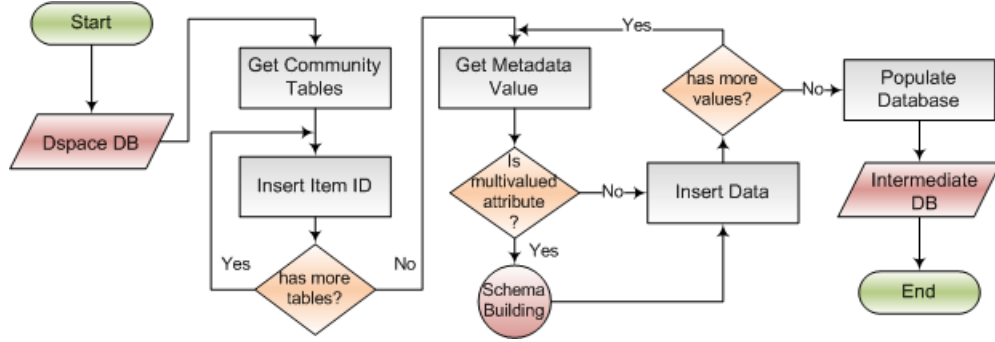


Figure 4.5: Work flow of intermediate database population

### 4.2.3 Populating Intermediate Database:

In populating intermediate database (PID), data is extracted from DSpace database and populated in the intermediate database by using following rules:

**Rule 9:** In relations  $R_{CM_m}$  and  $R_{CM_s}$ , the extracted item identifier is inserted.

**Rule 10:** The values of the attributes *atts* of a relation  $R_{CL_i}$  are extracted from the contents of the table: *metadatabalue* of DSpace database for each item.

**Rule 11:** If an attribute is found multivalued, then a weak primary relation  $R_{MV_i}$  is created as a child relation of  $R_{CM_{m_i}}$  by the BS component in the intermediate database. PID component populates the new relation with appropriate extracted data.

**Rule 12:** If an attribute is found complex attribute<sup>1</sup>, then a weak secondary relation  $R_{CMV_i}$  is created as a child relation of  $R_{CM_{m_i}}$  by the BS component in the intermediate database. PID component populates the new relation with appropriate extracted data.

The detailed process of the component is described in Algorithm 4.3 and its work flow is shown in Figure 4.4.

<sup>1</sup>An attribute that is both composite and multivalued.



---

**Algorithm 4.3** Populating Intermediate Database
 

---

```

1. START
2. INPUT: DSpace Database
3. VAR : tables, values : ArrayList; multivalued : Boolean;
4. tables  $\leftarrow$  Get all tables of intermediate database;
5. FOR i  $\leftarrow$  0 to i < tables.size
6.   table  $\leftarrow$  tables[i];
7.   values  $\leftarrow$  Get MetadataValues of the item;
8.   FOR j  $\leftarrow$  0 to j < values.size
9.     value  $\leftarrow$  values[j];
10.    multivalued  $\leftarrow$  check whether its is Multivalued Attribute
11.    IF it is multivalued attribute THEN
12.      Call SchemaBuilding module to create table
13.    END IF
14.    Insert extracted data in the table
15.  END FOR
16.END FOR
17.OUTPUT: Intermediate Database
18.END

```

---

### 4.3 Relation to Ontology Transformation

In this module, the intermediate database is transformed into ontology by using basic translation rules. Let  $R$  represents the relations where  $R_i \in R$ ,  $X$  represents the total attributes of the relation  $R_i$  and  $Y$  represents the non-key attributes of the relation  $R_i$ . The following rules are used for this transformation:

**Rule 1:** For a relation  $R_i$ , an ontology class  $C_i$  is created, if any of the following conditions satisfied (i)  $R_i$  does not contain any foreign key  $FK_i$ , and (ii) the primary key of  $R_i$  does not contain key of any other relation, such that  $PK_i \cap FK_i = \emptyset$ .

**Rule 2:** If two relations  $R_i$  and  $R_j$  have their primary keys linked with a foreign key constraint ( $PK_i = FK_j$ ), they are mapped to two OWL classes  $C_i$  and  $C_j$  respectively, the one being a super-class of the other. *rdfs:subClassOf* is added to respective class.

**Rule 3:** OWL class  $C_i$  is created for a relation  $R_i$  and *rdfs:subClassOf* is added, if it fulfills the following criteria (i)  $FK_i \subset PK_i$  means the primary key of  $R_i$  is a composite key and contains key(s) of any other relation(s); and (ii)  $\exists Y \in X - PK_i$  means  $R_i$  has non-key attributes.

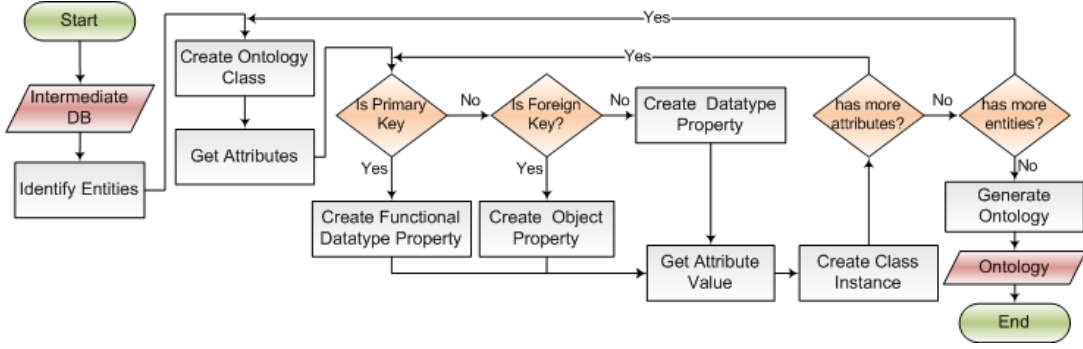


Figure 4.6: Work flow of relation to ontology transformation

**Rule 4:** OWL class is not created for  $R_i$ , if it fulfills the following criteria (i)  $FK_i \subset PK_i$  means the primary key of  $R_i$  is a composite key and contains key(s) of any other relation(s); and (ii)  $|X - PK_i| = 0$ , it means  $R_i$  has not any non-key attribute. The primary key of  $R_i$  which is not a foreign key is mapped to a *DatatypeProperty* and added to the class which has been created for its parent relation. The primary key of  $R_i$  which is also a foreign key is discarded as it has already been mapped to the *functional DatatypeProperty* in the parent relation.

**Rule 5:** An OWL *DatatypeProperty*  $P_i$  is created for non key attribute  $att$  of a relation  $R_i$ .

**Rule 6:** A *functional DatatypeProperty*  $P_i$  with *minCardinality* restriction of 1 is created for primary key attribute  $PK_i$  of a relation  $R_i$ .

**Rule 7:** For each foreign key  $FK_i$  of a relation  $R_i$ , an *ObjectProperty*  $P_i$  is created. Let a relation  $R_i$  (i.e. child relation) refer to a relation  $R_j$  (i.e. parent relation) and  $C_i$  and  $C_j$  are their corresponding classes. The domain of *ObjectProperty* is specified as  $Domain\ of\ P_i = C_i$  and the range is specified as  $Range\ of\ P_i = C_j$  and *allValuesFrom* restriction is applied to the property.

**Rule 8:** For every tuple  $R_i[t_i]$ , the value of an attribute  $att$  maps to a value of the property  $P_i$ .

The detailed process of the module is described in Algorithm 4.4 and main work flow of this transformation is shown in Figure 4.5.

---

**Algorithm 4.4** Relation to Ontology Transformation
 

---

```

1. START
2. INPUT: Intermediate Database
3. VAR : entities, attributes : ArrayList<String>; entity, att, parent_entity : String; i, j : Integer;
4. entities ← IdentifyEntities();
5. FOR i ← 0 to i < entities.size
6.   entity ← entities[i];
7.   attributes ← GetAttributes(entity);
8.   FOR j ← 0 to j < attributes.size
9.     att ← attributes[j];
10.    IF att is Primary Key THEN
11.      Create FunctionalDatatype Property
12.    ELSE IF att is Foreign Key THEN
13.      Create FunctionalObject Property
14.    ELSE
15.      Create Datatype Property
16.    END IF
17.    value ← GetAttributeValue(att);
18.    Create Class Instance
19.  END FOR
20. IF PrimaryKey is linked with ForeignKey constraint THEN
21.   parent_entity ← Get Parent Entity
22.   Add subClassOf <parent_entity> Property
23.END FOR
24.OUTPUT: Ontology
25.END

```

---

## 4.4 Ontology Alignment and Transformation

In this module, an RDF has been generated which contains the individuals of VIVO ontology classes. This module takes two ontologies as its input, one is the source ontology which is generated from the DSpace database and another is VIVO ontology (i.e. target ontology). The classes from both ontologies are extracted and simple mappings between these classes are generated by the user. Then the individuals of these mapped classes are matched and proper URIs are given to each individual. These individuals are linked with their respective VIVO classes. Finally, an RDF, having VIVO individuals, is generated. This module comprises three sub-modules as discussed below.

### 4.4.1 Resource Mappings (RM)

The mapping process only produces a set of correspondings without modifying the ontologies which is then used for ontology alignment and merging.

**Definition 1:** Given two ontologies  $\mathcal{O}1$  (i.e., source ontology) and  $\mathcal{O}2$  (i.e., target ontology). RM generates an alignment  $A$  between the concepts of these two ontologies. Let  $C1$  and  $C2$  be the concepts of  $\mathcal{O}1$  and  $\mathcal{O}2$  respectively such that  $C1_i \in C1$  and  $C2_j \in C2$ .  $A$  is a pair of  $(C1_i, C2_j)$  which means that  $C1_i \equiv C2_j$ .

System extracts all the classes (i.e.  $C1, C2$ ) from both source and target ontologies (i.e.  $\mathcal{O}1$  and  $\mathcal{O}2$  respectively). User selects appropriate classes & system generates simple alignment between the concepts of generated ontology  $\mathcal{O}1$  and the VIVO ontology  $\mathcal{O}2$ .

### 4.4.2 Instance Matching (IM)

In this step, system attempts to match the instances of mapped concepts by using generated alignment  $A$ .

**Definition 2:** Let instance matching (IM) is a triple  $(I1_k, I2_l, R)$ , where  $I1_k \in I1$  and  $I1$  are the instances of  $C1_i$ ,  $I2_l \in I2$  and  $I2$  are the instances of  $C2_j$  and  $R$  is a relation between instances  $I1_k$  and  $I2_l$ .  $R$  is taken from the set  $\{\equiv, \perp, \}$  for

equivalence and disjointness respectively. IM gives  $I1_k$  the same URI as  $I2_l$ , if  $(I1_k, I2_l, \equiv)$ .

System compares the instances  $I1$  of  $C1_i$  with the existing instances  $I2$  of  $C2_j$ . If  $I1_k$  matched with  $I2_l$ , then it means that these two instances are equivalent. Therefore, it creates the triple  $(I1_k, I2_l, \equiv)$  which shows that  $I1_k \equiv I2_l$ . Then  $I1_k$  is given the URI of the  $I2_l$ , which identifies the two as the same instance. For example,  $I1_k$  is an instance of author class (i.e.,  $C1_i$ ) which is mapped to the person class (i.e.,  $C2_j$ ).  $C2_j$  has the same author (i.e.,  $I2_l$ ). Therefore, instead of creating new instance of  $C2_j$ , the URI of  $I2_l$  has been given to  $I1_k$ .

If  $I1_k$  is not matched with any instance of  $C2_j$ , then it means that these two instances are disjoint. Therefore, it creates the triple  $(I1_k, I2_l, \perp)$  which shows that  $I1_k \perp I2_l$ . Then, Resource Linkage (RL) process gives a proper VIVO URI to  $I1_k$ .

#### 4.4.3 Resource Linkage (RL)

In this step, system gives a proper VIVO URI to all unmatched instances and links the new data with existing data in VIVO.

**Definition 3:** Let instance matching is a triple  $(I1_k, I2_l, R)$ , where  $I1_k \in I1$  and  $I1$  are the instances of  $C1_i$ ,  $I2_l \in I2$  and  $I2$  are the instances of  $C2_j$  and  $R$  is a relation between instances  $I1_k$  and  $I2_l$ .  $R$  is taken from the set  $\{\equiv, \perp\}$  for equivalence and disjointness respectively. RL creates new instances of  $C2_j$ ,  $\forall (I1_k, I2_l, \perp)$  and links them with the existing data.

If  $I1_k$  is not matched with any instance of  $C2_j$ , RL process gives a proper VIVO URI to  $I1_k$ . URIs are arbitrarily generated by the system according to VIVO Namespace. System compares these URIs with the URIs already in VIVO and provides surety about their uniqueness. After that system links the new data with existing data in VIVO.

For example,  $I1_k$  is an instance of Publication class (i.e.,  $C1_i$ ) which is mapped to the Conference Paper class (i.e.,  $C2_j$ ).  $I1_k$  does not already exist in  $C2_j$ , RL process gives a proper VIVO URI to  $I1_k$  and checks whether the author of  $I1_k$  already exists in person class (i.e.,  $C2_j$ ). If the author (i.e.  $I2_l$ ) already exists

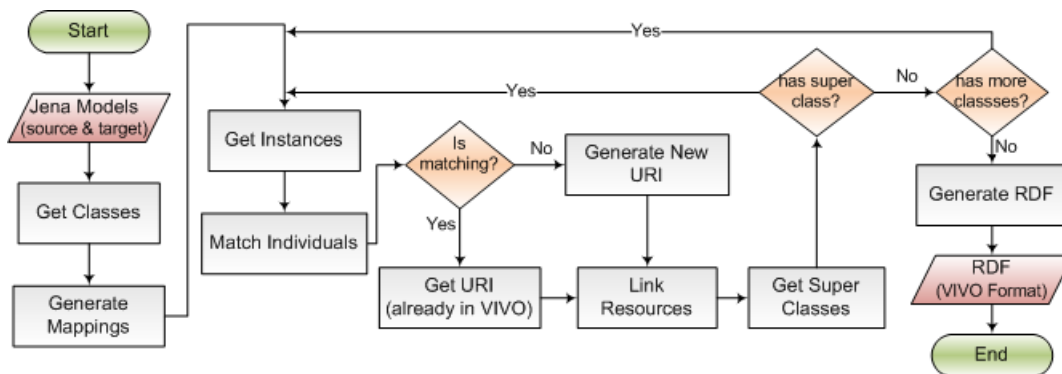


Figure 4.7: Work flow of ontology alignment and data translation

in  $C2_j$ , then RL process links publication  $I1_k$  with author  $I2_l$  by *vivo:linkedAuthor* property. In this way an existing individual in VIVO can have more data added to it. For example, new publication can be linked to existing author or an organization can have more sub or super organizations added to it, etc. The rank of every author has also been maintained properly. Complete class hierarchy has been maintained by identifying and linking super and sub-classes. For example, when system maps a new organization, it extracts full organizational hierarchy from source ontology to sort out its super and sub-organizations. *vivo:subOrganizationWithin* property has been added to respective class.

At the end of this module, system creates an RDF that contains individuals in VIVO format. After this step all data has a proper VIVO URI and is ready for import into VIVO. The main activities and processes of this module is shown in Figure 4.7.

## 4.5 Walkthrough Example

A DSpace data model of an institute, shown in Figure 4.8, is considered as an example to demonstrate the proposed system. The data model contains one main community, eight sub-communities, ten collections and One hundred items. The relevant relations of DSpace database schema is shown in Table 4.1. The schema illustrates that DSpace database maintains the data model information in auxiliary relations. The contents of the relations are used for the creation of the intermediate database of a specific data model of an institute.

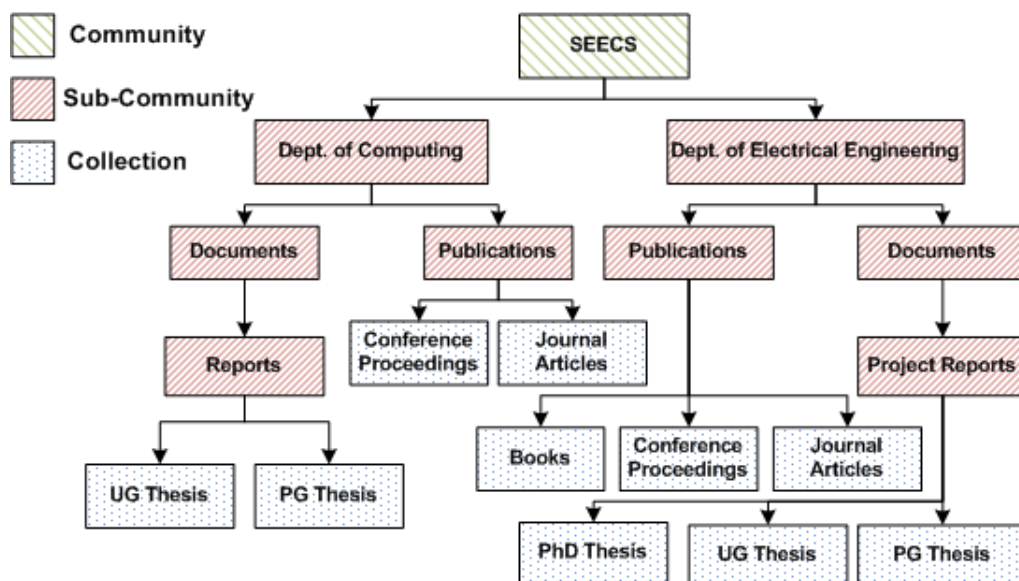


Figure 4.8: DSpace data model of an institute

Table 4.1: A relevant portion of DSpace Database Schema

Relation	Foreign Key
item (item_id, submitter_id, in_archive , withdrawn, owning_collection)	<i>owning_collection</i> referring to <i>collection_id</i> in collection
collection (collection_id, name, short_description, submitter)	Nil
community2collection (id, community_id, collection_id)	<i>community_id</i> referring to <i>community_id</i> in community, <i>collection_id</i> referring to <i>collection_id</i> in collection
community (community_id, name, short_description)	Nil
community2community (id, parent_comm_id, child_comm_id)	<i>parent_comm_id</i> and <i>child_comm_id</i> referring to <i>community_id</i> in community
metadatabfieldregistry (metadata_field_id, element, qualifier, scope_note)	Nil
metadatabvalue (metadata_value_id, item_id, metadata_field_id, text_value)	<i>item_id</i> referring to <i>item_id</i> in item, <i>metadata_field_id</i> referring to <i>metadata_field_id</i> metadatabfieldregistry

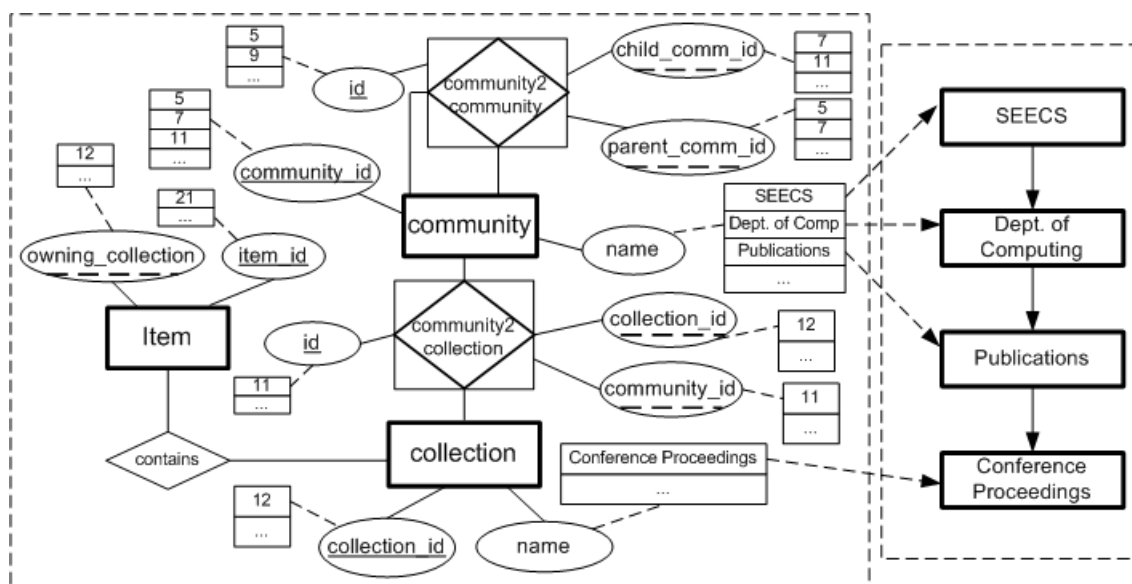


Figure 4.9: Identified Data Model after applying DMI process

#### 4.5.1 Metadata Extraction

In the first module of the system, an intermediate database is created. This work is done in three steps. Figures 4.9, 4.10 and 4.11 illustrate these steps. Each figure is divided into two parts, the left side of a figure depicts the schema of DSpace database and the right side represents corresponding extracted information. In these figures rectangles and ellipses represent relations and their attributes respectively. The values of their respective attributes are represented by arrays.

In the first step, all items are extracted and then for every item its owning collection and parent communities are identified. The step is demonstrated graphically in Figure 4.9. For example, an item having *item\_id* 21 is extracted. After that its corresponding collection having *collection\_id* 12 is extracted, the name of this collection: *Conference Proceedings* is identified. *Conference Proceedings* is appeared in *Publications* community and *Publications* is identified as a sub-community of *Dept. of computing* which is contained in community: *SEECS*. The identified attribute values build a hierarchy of the data model as shown in Figure 4.9. The procedure is repeated for all the items and finally the complete data model is identified as shown earlier in Figure 4.8.

In the second step, an intermediate database schema is created for identified data model using the extracted information. Database relations are created for every collection, community and sub-community; and multivalued attribute if exists. This step is demonstrated graphically in Figure 4.10. A relation, *conference\_proceedings*, has been created for the identified collection name *Conference Proceedings* in the



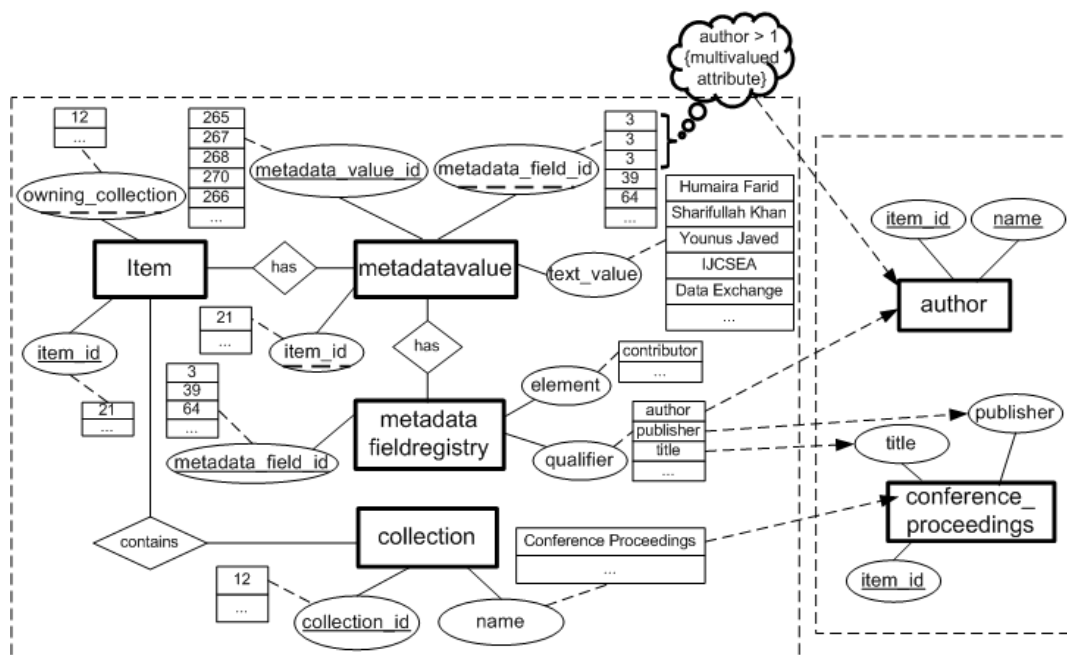


Figure 4.10: Extracted intermediate schema after applying Schema Building rules

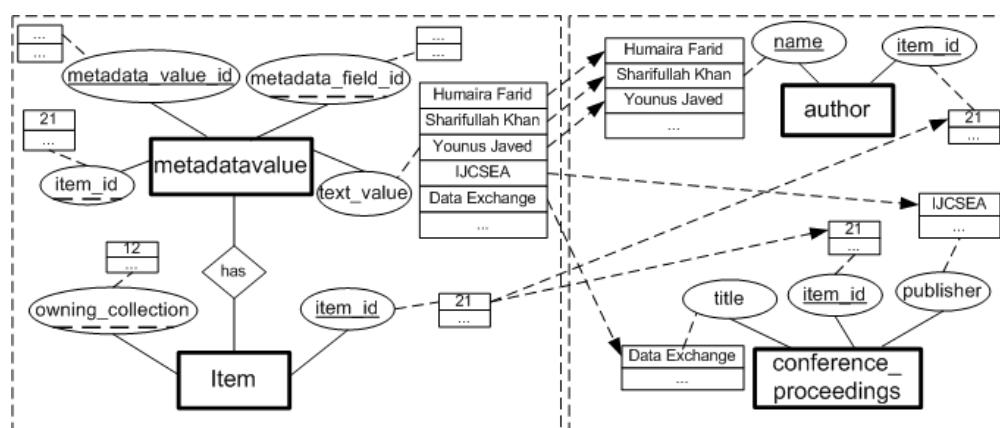


Figure 4.11: Intermediate Database after Schema Population

intermediate database. Attributes of this relation are extracted from the DSpace database table: *metadata fieldregistry*. In order to extract attributes, first of all identifiers of metadata fields, i.e. *metadata\_field\_id* are extracted from the *metadatavalue* table for all items. The values of *metadata\_field\_id*, (i.e. 3, 3, 3, 39, 64) are extracted from the *metadatavalue* table for item having identifier: *item\_id* 21. The corresponding attributes for these identifiers are extracted from table: *metadata fieldregistry*. In the extracted identifier values, 3 is repeating three times. The *qualifier* for *metadata\_field\_id* value 3 is *author*. So *author* is considered as a multivalued attribute and thus a separate relation is created for this attribute. The same process is repeated for all the identifier values. However they are not multivalued, so they are considered as attributes of the created relation: *conference\_proceedings*.

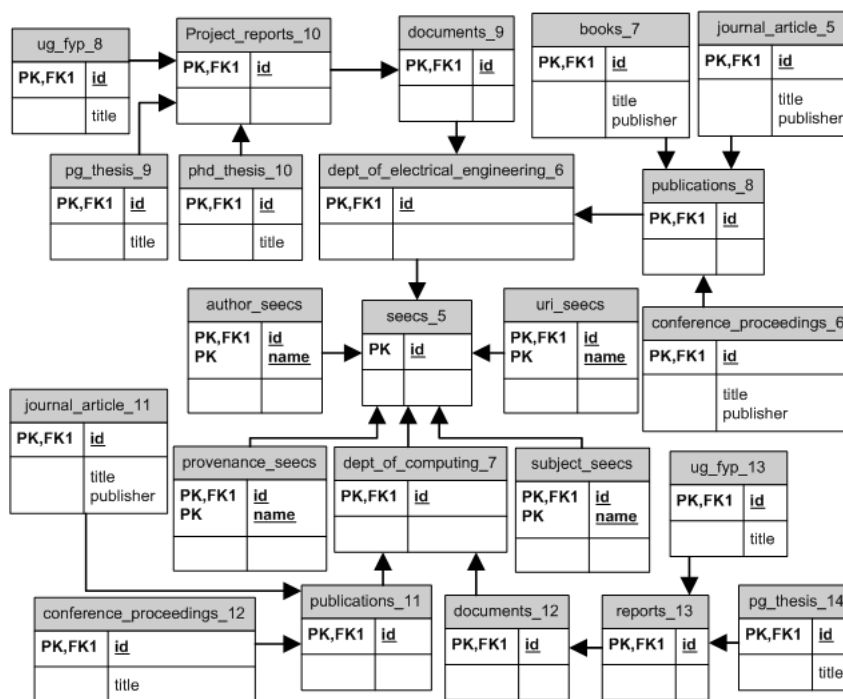


Figure 4.12: Generated Relational Database of above DSpace Structure

In the final step, item metadata is extracted from the DSpace database and populated in the intermediate database. The graphical illustration of the step is shown in Figure 4.11. On the right side of the figure, the created two relations: *author* and *conference\_proceedings* are shown. The values of their attributed are extracted from DSpace database table: *metadatatypevalue*. The corresponding values of the attributes: *name*, *publisher*, and *title* (i.e. Humaira Farid, Sharifullah Khan, Younas Javed, IJCSEA, Data Exchange) are extracted through the identified values of *metadata\_field\_id*, (i.e.3, 3, 3, 39, 64) in table *metadatatypevalue*. After applying the first module of the system on this example, an intermediate database, having 23 relations (tables), is created as shown in Figure 4.12.

#### 4.5.2 Relation to Ontology Transformation

In the second module, an ontology is created against the intermediate database by using transformation rules, defined in Section 4.3. The ontology created for the given DSpace data model is shown in Figure 4.13. In this ontology, OWL class *seecs\_5* has been created for the main community *SEECs*, and sub-classes are created for every sub-community, collection and multivalued attributes, such as *dept\_of\_computing\_7*, *conference\_proceedings\_12* and *author\_seecs* respectively. A functional datatype property has been created for the primary key (i.e., *id*) of the intermediate database relation *seecs\_5*. Similarly functional object property

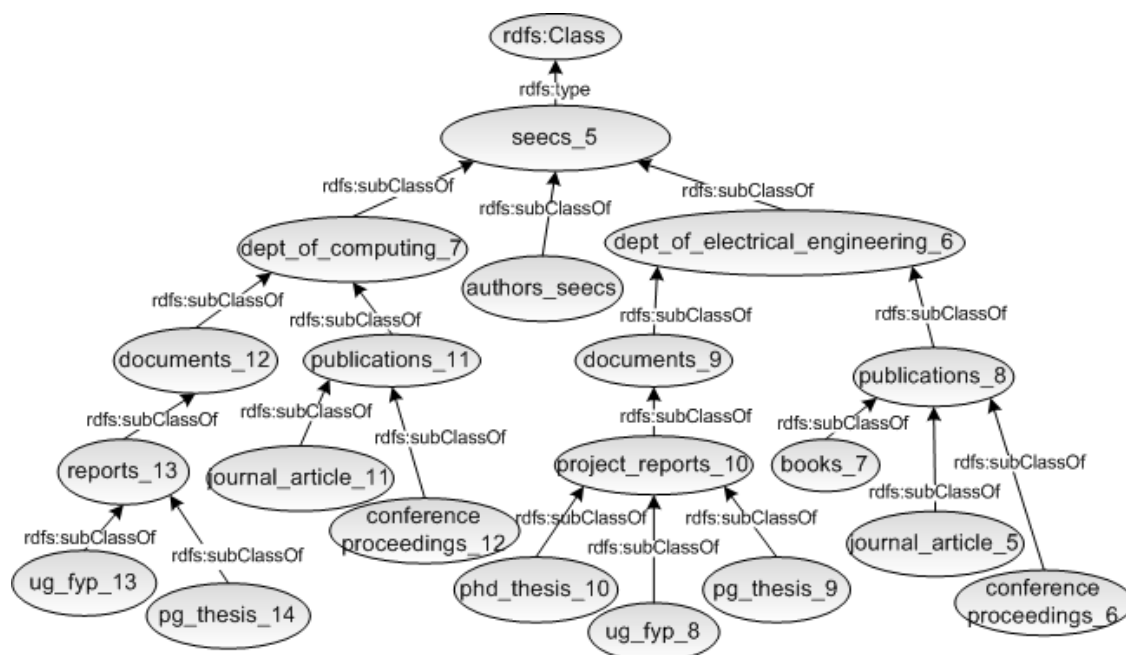


Figure 4.13: Graphical representation of the generated ontology

with cardinality restriction 1 has been created for the foreign key (i.e., id) that is also defined as a primary key in table *dept\_of\_computing\_7*. Since primary key is linked with a foreign key constraint (i.e.,  $PK=FK$ ), so *dept\_of\_computing\_7* has been made as a sub-class of *seecs\_5*.

### 4.5.3 Ontology Alignment and Data Translation

In last module, instances from created ontology is translated into corresponding data or instances expressed in VIVO ontology. The RDF is generated which contains the individuals of VIVO ontology classes. This RDF is then imported into VIVO. This work has been done in four steps. In the first step, all the classes from both source and target ontologies are extracted. Then user selects appropriate classes from both ontologies and system generates simple mappings between these classes. For example, following mappings are generated:

- author  $\rightarrow$  Person
- dept. of computing  $\rightarrow$  AcademicDepartment
- conference proceedings  $\rightarrow$  ConferencePaper

Here, an author class is mapped on the person class. dept. of computing is mapped on the AcademicDepartment class which is sub-class of Organization in VIVO ontol-

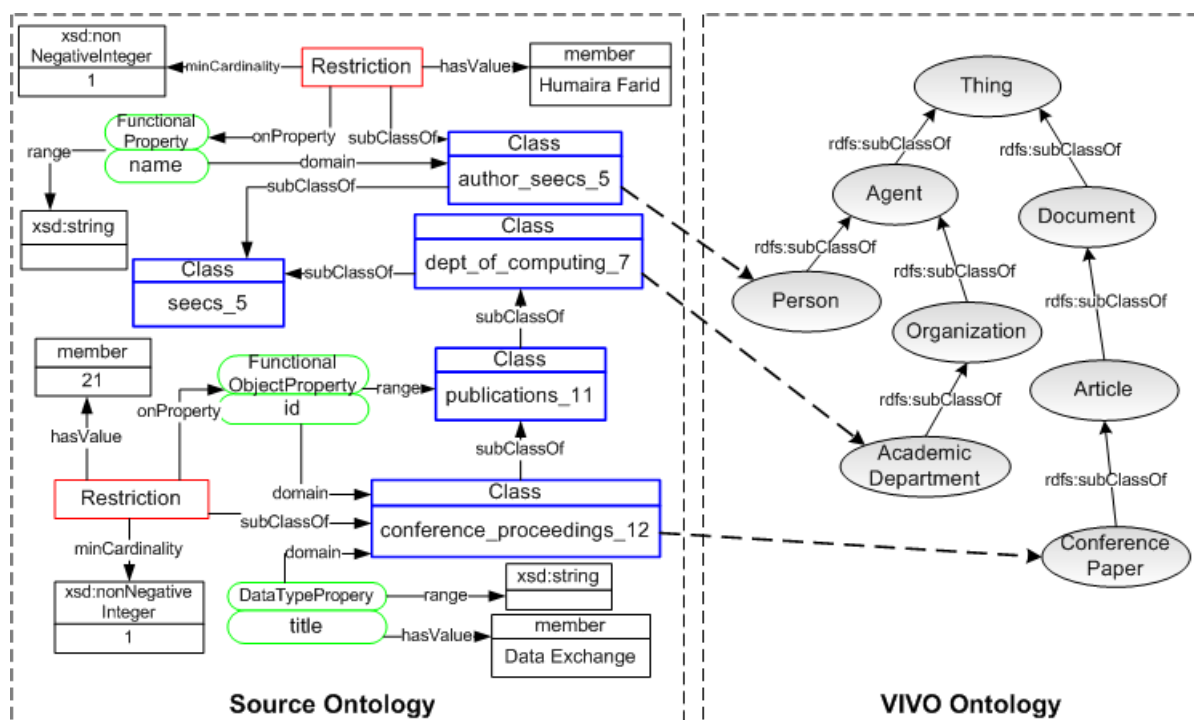


Figure 4.14: Example of Ontology Mapping

ogy. conference proceedings is mapped on ConferencePaper which has three super-classes, Article, Document and InformationResource. Example is shown in Figure 4.14.

In the second step, instances of the mapped classes are matched. If the individual already exists, then the matched input is given the same URI as the individual in VIVO matching it. For example, conference paper has an author: Humaira Farid. This name has been searched in the Person class. VIVO person class has an individual having the same name. System gets URI of that individual which is <http://localhost:8080/vivo/individual/n3119>. That author is given the same URI. and corresponding publication has been added and linked to that author.

In the third step, a proper VIVO URI has been given to all unmatched data. The matched input entities that are compared in Instance Matching process are then linked with the new data. For example, according to the VIVO Namespace, system generated new VIVO URI for conference paper which is <http://localhost:8080/vivo/individual/n174>, dept. of computing is given the URI: <http://localhost:8080/vivo/individual/n16164>, and added as the sub-organization of SEECS. After that, this paper has been linked with its authors and corresponding organization. This paper has three authors: Humaira Farid, Sharifullah Khan and Muhammad Younus Javed, having URIs <http://localhost:8080/vivo/individual/n3119>, <http://localhost:8080/vivo/individual/n6613> and <http://localhost:8080/vivo/individual/n6616> respectively. The corresponding

```

<rdf:Description rdf:about="http://localhost:8080/vivo/individual/n13120">
  <vivo:authorRank rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</vivo:authorRank>
  <vivo:linkedInformationResource rdf:resource="http://localhost:8080/vivo/individual/n17460"/>
  <vivo:linkedAuthor rdf:resource="http://localhost:8080/vivo/individual/n3119"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#Authorship"/>
</rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/vivo/individual/n17352">
  <vivo:authorRank rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</vivo:authorRank>
  <vivo:linkedInformationResource rdf:resource="http://localhost:8080/vivo/individual/n17460"/>
  <vivo:linkedAuthor rdf:resource="http://localhost:8080/vivo/individual/n6613"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#Authorship"/>
</rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/vivo/individual/n13611">
  <vivo:authorRank rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</vivo:authorRank>
  <vivo:linkedInformationResource rdf:resource="http://localhost:8080/vivo/individual/n17460"/>
  <vivo:linkedAuthor rdf:resource="http://localhost:8080/vivo/individual/n6616"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#Authorship"/>
</rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/vivo/individual/n18098">
  <vivo:linkedInformationResource rdf:resource="http://localhost:8080/vivo/individual/n17460"/>
  <vivo:linkedAuthor rdf:resource="http://localhost:8080/vivo/individual/n16164"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#Authorship"/>
</rdf:Description>

```

Figure 4.15: Individual entities for every linked author in RDF

```

<rdf:Description rdf:about="http://localhost:8080/vivo/individual/n17460">
  <rdfs:label>Data Exchange</rdfs:label>
  <vivo:informationResourceInAuthorship rdf:resource="http://localhost:8080/vivo/individual/n13120"/>
  <vivo:informationResourceInAuthorship rdf:resource="http://localhost:8080/vivo/individual/n17352"/>
  <vivo:informationResourceInAuthorship rdf:resource="http://localhost:8080/vivo/individual/n13611"/>
  <vivo:informationResourceInAuthorship rdf:resource="http://localhost:8080/vivo/individual/n18098"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#InformationResource"/>
  <rdf:type rdf:resource="http://purl.org/ontology/bibo/Document"/>
  <rdf:type rdf:resource="http://purl.org/ontology/bibo/Article"/>
  <rdf:type rdf:resource="http://vivoweb.org/ontology/core#ConferencePaper"/>
</rdf:Description>

```

Figure 4.16: Publication with corresponding linked entities in RDF

organization is dept. of computing having URI *http://localhost:8080/vivo/individual/n16164*. For every author separate entity is added in which publication, organization and authors are linked with each other.

In the last step, an RDF having VIVO individuals are generated. For example, Figure 4.15 shows the separate individual entities for every linked author in RDF. New conference paper has been added and linked with corresponding entities as shown in Figure 4.16.

## Chapter 5

# System Design and Implementation

### 5.1 Introduction

In this chapter, the details of design and implementation of the proposed system are discussed. In design, different classes are created. These classes represent the functions and attributes which are used in implementation. The implementation has been started by extracting an IR metadata which is then transformed into ontology. Ontology has been created using Jena-2.6.4 Ontology API. Then instances of that ontology are matched and translated into the VIVO ontology. A front end application has been developed in NetBeans IDE 7.1.2 for the implementation of the system.

### 5.2 Design and Implementation

The design model is the refinement of requirement model with respect to the implementation environment. The most important part of design model is to define all the objects and classes in the system. Class diagram is used to represent the different relationship among different classes as shown in Figure 5.1.

This section presents prototype for the proposed system and underlines following three major components.

- a. Metadata Extraction

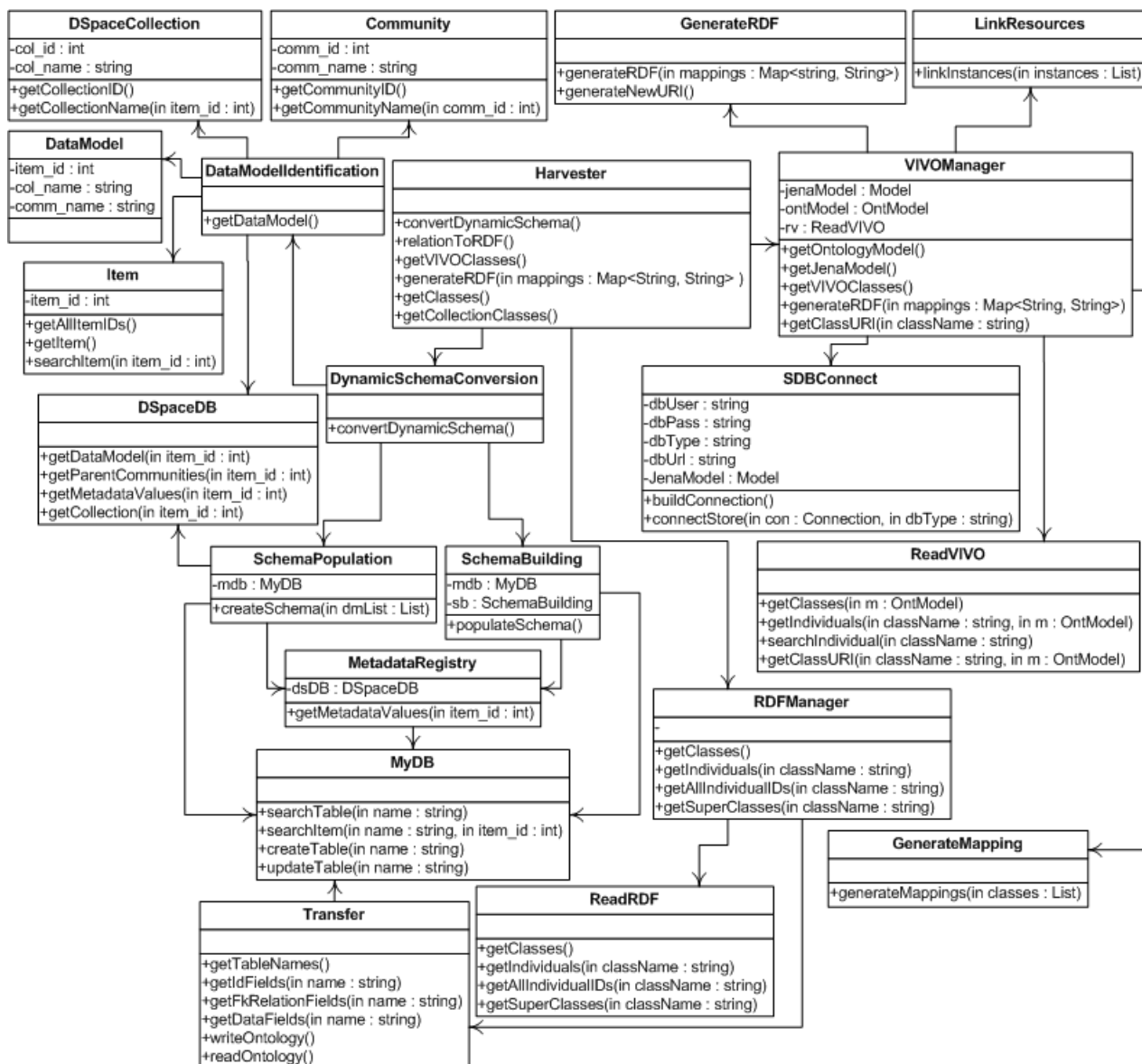


Figure 5.1: Class Diagram

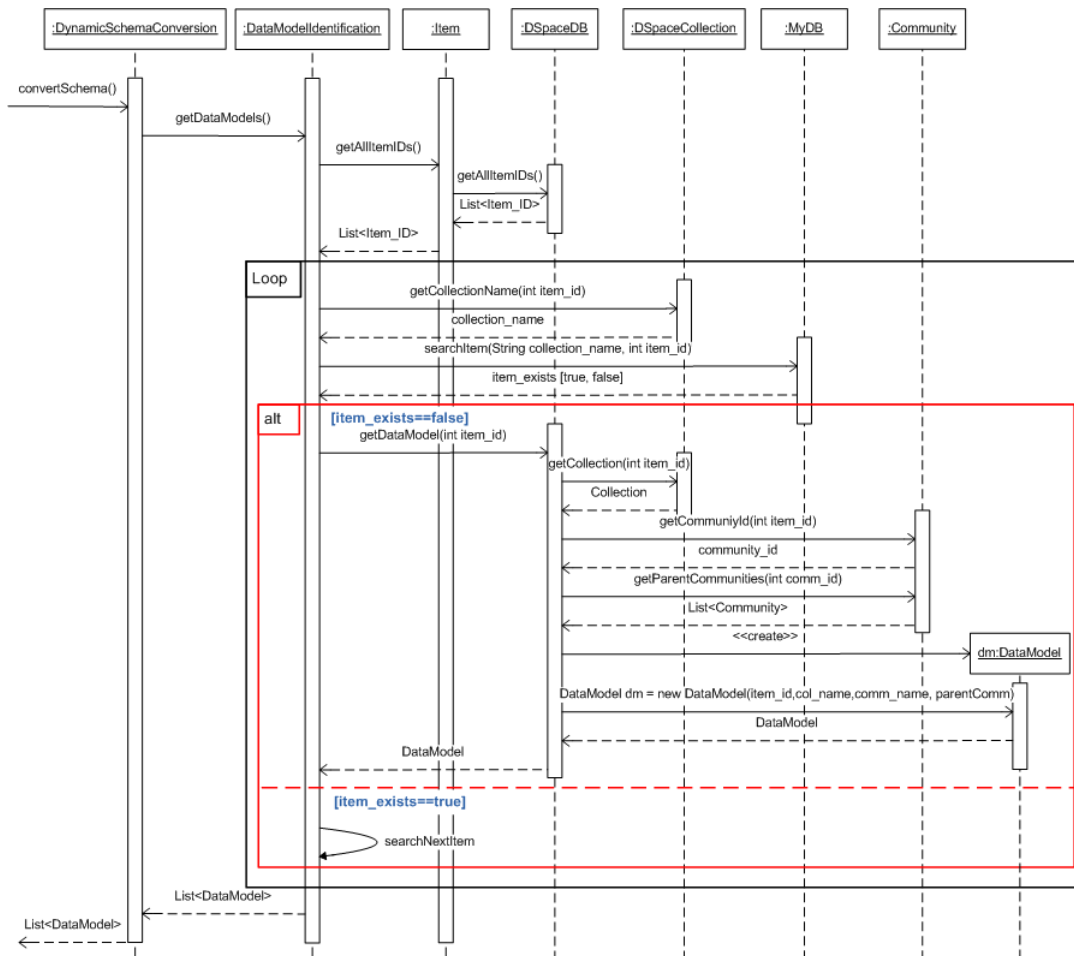


Figure 5.2: Sequence Diagram for Data Model Identification

- b. Relation to Ontology Transformation
- c. Ontology Alignment and Data Translation

Detail of each component and subsequent sub components are given below. The description will cover the challenges faced during the implementation of these modules.

### 5.2.1 Metadata Extraction

In this module a normalized schema of an IR data model and its data is obtained by extracting metadata from the nested schema of an IR database. The System has been implemented using Java language. This component has been divided into three sub components.

- a. Identification Data Model (IDM)



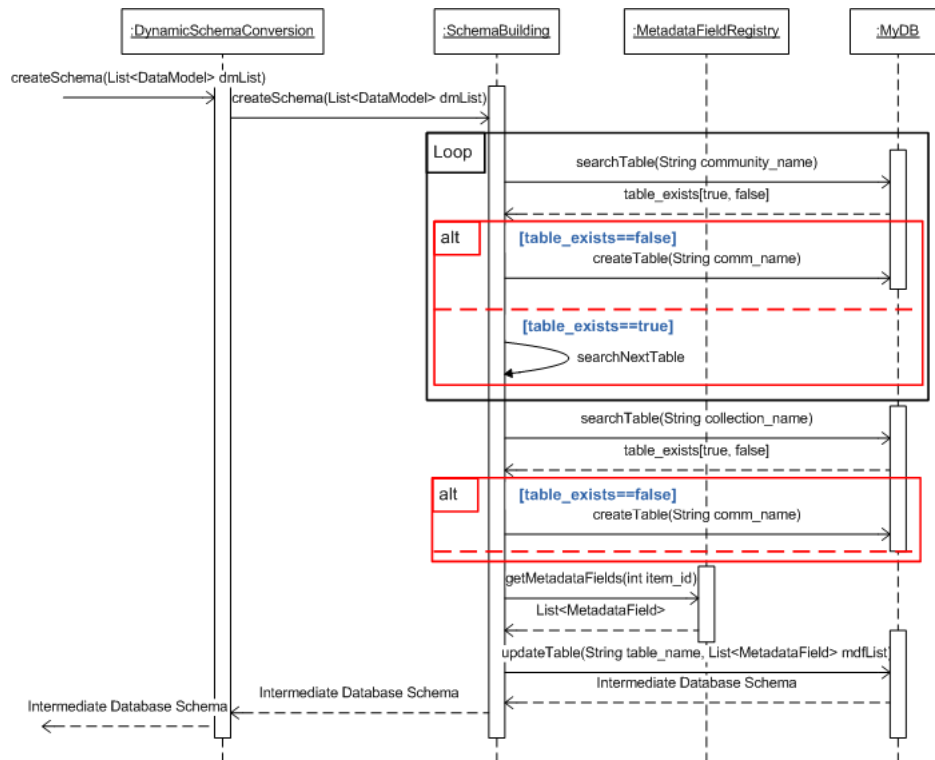


Figure 5.3: Sequence Diagram for Schema Building

- b. Building Schema (BS)
- c. Population of Intermediate Database (PID)

Sequence diagrams are used to represent the sequence of activities of each component. A sequence diagram is a kind of interaction diagram which shows interaction among processes and the order in which they interact with each other.

First of all a data model of an institutional repository (IR) is identified. For that purpose, the identifier of an item is extracted from the *item* table. Then the names of its respective collection and parent communities are identified. The process is repeated for all the items. The sequence diagram shown in Figure 5.2, depicts the sequence of messages exchanged between the objects required to identify the data model of an IR. In this way the names of communities, sub-communities and their respective collections are identified. These identified entities and their relationship represent the data model of an institute.

Secondly, the extracted data model is used for creating the schema of an intermediate database. Attributes of the intermediate database relations are extracted from the DSpace Database. The sequence diagram for building the schema of intermediate database is shown in Figure 5.3.

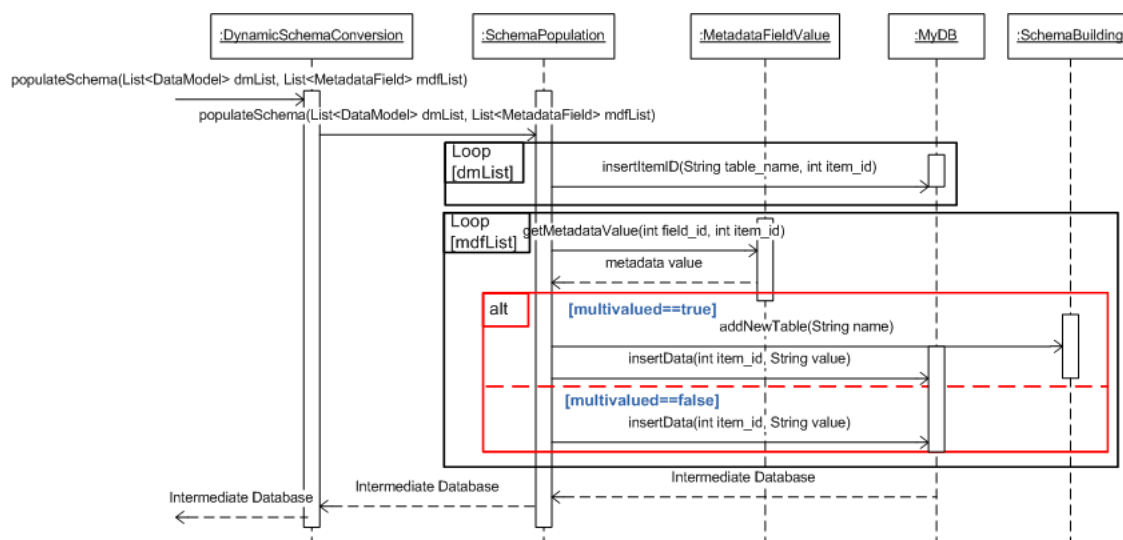


Figure 5.4: Sequence Diagram for Intermediate Database Population

In the final step, data is extracted from DSpace database and populated in the intermediate database, the detailed process is shown in Figure 5.4.

### 5.2.2 Relation to Ontology Transformation

The intermediate database is transformed into ontology. Jena Ontology API is used for creating ontology. Jena is an open source Java API. This API is used to create and manipulate RDF graphs. Object classes are used to represent graphs, properties, resources, and literals. In Jena, a graph is called a model and is represented by the Model interface.

The sequence diagram shown in Figure 5.5, depicts the objects and classes involved in this transformation and the sequence of messages exchanged between the objects required to carry out the functionality of this scenario.

### 5.2.3 Ontology Alignment and Data Translation

Created Ontology is then mapped and merged into the VIVO Ontology. VIVO is storing data in MySQL database by using Jena SDB, but it is not in a relational format so it cannot be easily browsed or queried using SQL tools. SDB is a component of Jena for RDF storage and it also supports SPARQL queries. VIVO stores all of its triples in named graphs. The *quads* table is where it stores these triples, along with a column indicating what named graph each triple is stored in. The values in the *quads* table key into the *nodes* table. Attempting to write to the database directly

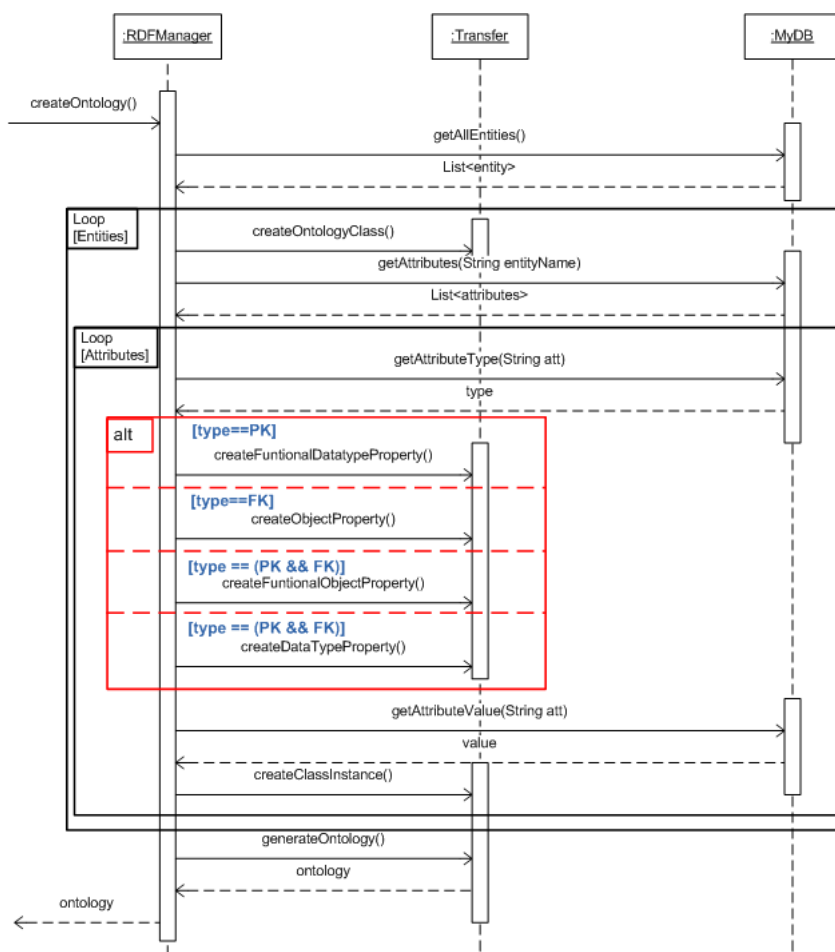


Figure 5.5: Sequence Diagram for Relation to Ontology Transformation

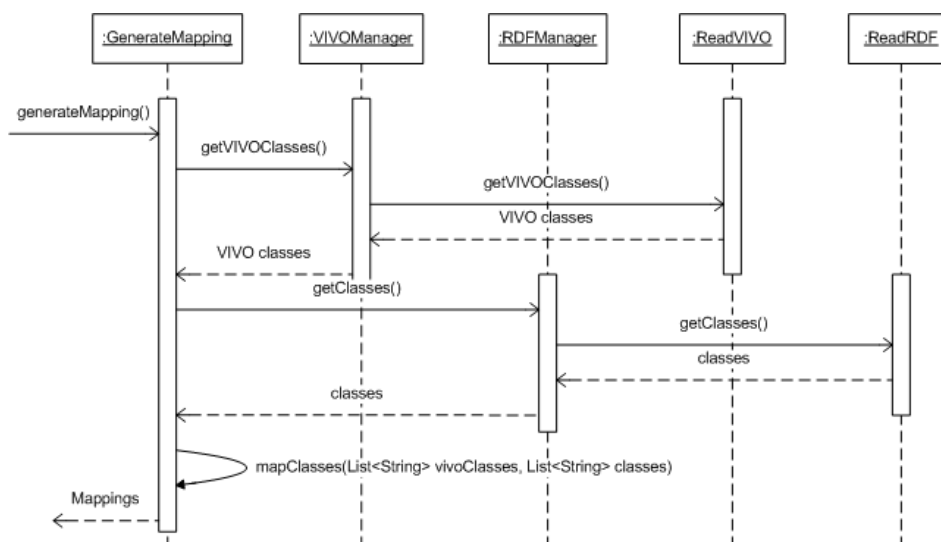


Figure 5.6: Sequence Diagram for Resource Mapping

or querying it is cumbersome and it requires knowledge and expertise in writing SPARQL queries. VIVO has been queried by using the Jena Model API with SDB. SDB 1.3.3 library is used in implementation. SDB loads and queries data based on the unit of a *Store*. The *Store* object has all the information for formatting, loading and accessing an SDB database. One database or table space is one *Store*. *Store* objects are made via the static method of the *StoreFactory* class. *SDBConnection* wraps the underlying database connection, as well as providing logging operations. The *SDBFactory* is used to create Jena Model backed by an SDB store.

For that purpose, classes from both ontologies are extracted and mappings are generated. After that, instances of mapped classes are matched and proper URIs are assigned to every matched and unmatched instances. Instances are linked with each other and RDF in VIVO format is generated which is then imported into VIVO. This component is divided into three sub components:

- a. Resource Mapping
- b. Instance Matching
- c. Resource Linkage

In first step, system extracts all the classes from both source and target jena models. User selects appropriate classes and system generates simple mappings between the generated ontology and the VIVO ontology, the detailed procedure is shown in Figure 5.6.

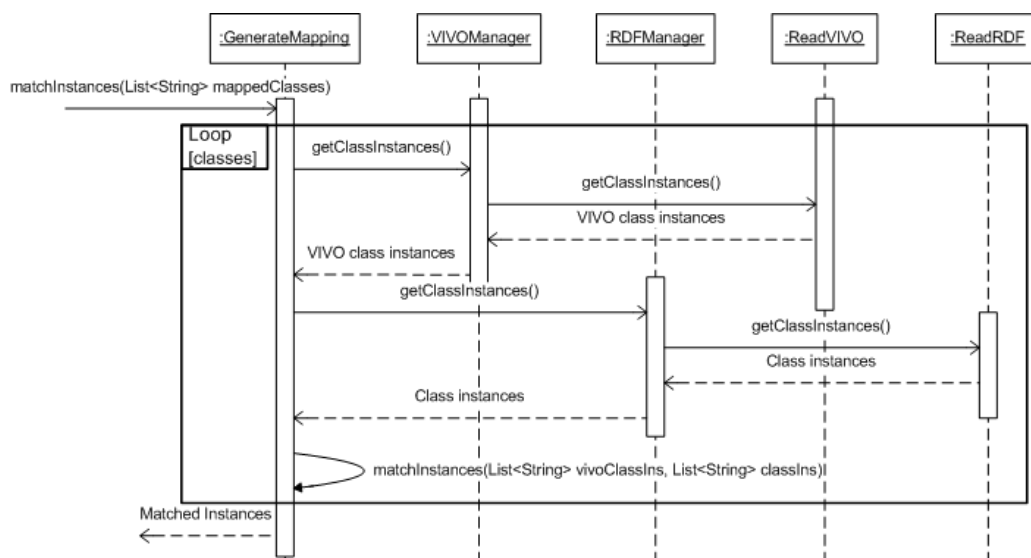


Figure 5.7: Sequence Diagram for Instance Matching

In the second step, system attempts to match incoming data with data already in VIVO. System evaluates the information against the existing VIVO system looking for links and equivalent entities. It compares the individuals of mapped classes. If the individual already exists, then the matched input is given the same URI as the individual in VIVO matching it. The sequence diagram for this step is shown in Figure 5.7.

In the final step, system gives a proper VIVO URI to all unmatched data. URIs are arbitrarily generated by the system. System compares these URIs with the URIs already in VIVO and provides surety about their uniqueness. After that the matched input entities that are compared in Instance Matching process, are linked with the new data. Complete class hierarchy has been maintained by identifying and linking super and sub-classes. In this way an existing individual in VIVO can have more data added to it. For example, new publication can be linked to existing author or an organization can have more sub or super organizations added to it, etc. The sequence diagram for this step is shown in Figure 5.8.

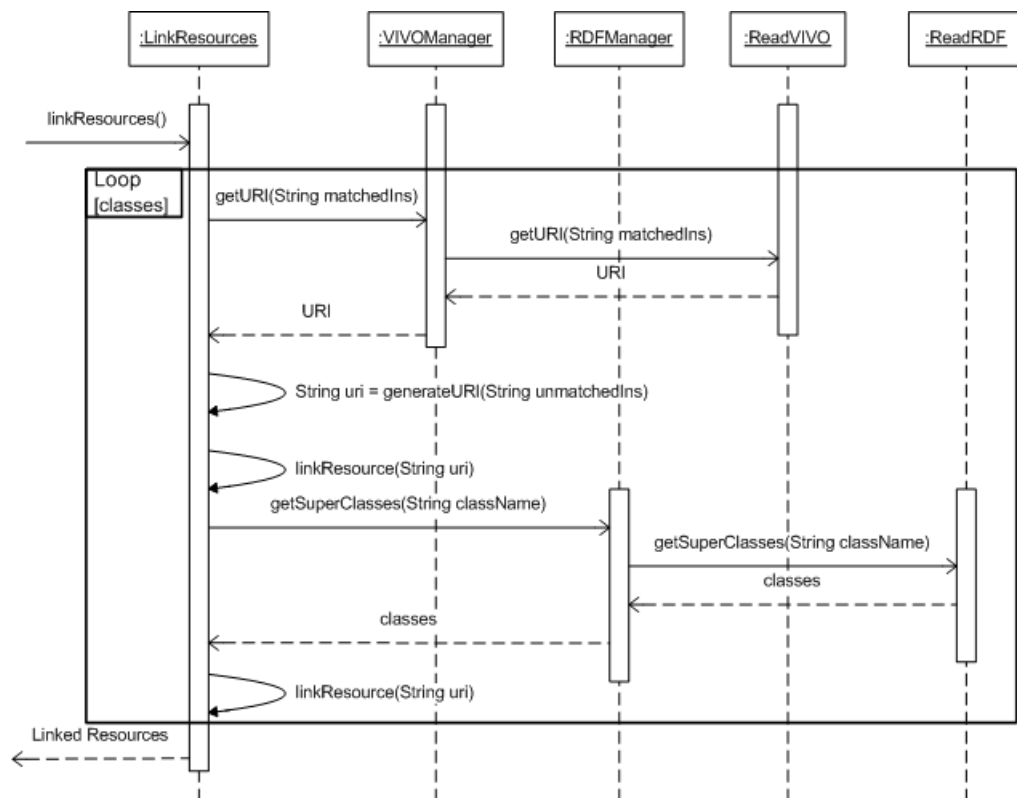


Figure 5.8: Sequence Diagram for Resource Linkage

## Chapter 6

### System Evaluation

#### 6.1 Introduction

In this chapter, the results of the proposed system are evaluated. First the operational goals of the proposed transformation system are explained then the specifications of the data sets are provided. These data sets are used for the evaluation of the proposed system. Later, it is prove that the transformation is information capacity preserving and correct by comparing it with existing techniques. The chapter is important because it verifies the correctness of the proposed system.

#### 6.2 Transformation Process and its Evaluation

In transformation systems, correctness is based on information capacity. A correct and lossless transformation ensures that all data in a source model is represented in the target model. The evaluation criteria for the lossless and correct transformation is defined:

- a. **Preservation of information capacity:** This criterion finds out whether the source model (i.e., relational database) is completely transformed into the target model (i.e., ontology).
- b. **Correct identification and transformation of data:** The correctness of the transformation is evaluated by this criterion. It is significant to find the quality of the built ontology.

### 6.2.1 Operational Goals

Every transformation system has different goals and operational requirements which should be identified explicitly. These goals should be analyzed carefully to check the correctness of the system and to prevent from occasional errors and misconceptions. Therefore, a collection of operational goals are identified for developing correctness criteria. Table 6.1 shows the main operational goals that are defined for measuring the quality and correctness of the generated ontology. G2, G4 and G7 are the most important rules in defined case. G2 is related to the preservation of DSpace data model hierarchy, this goal is important to identify the complete hierarchy of DSpace item. G4 emphasis on the use of multivalued property rules. According to these rules, a relation that has not any non-key attribute and has a composite primary key containing a foreign key referencing a parent relation should be translated to a datatype property with domain the class that corresponds to the parent relation. In the proposed transformation G4 is also important because if it is not achieved then it can cause a loss of information while populating VIVO with generated ontology instances. The generated ontology must be populated with instances from the database, therefore, it also important to achieve G7. The several transformation techniques are examined which have been proposed in the literature. Table 6.2 shows the different techniques and the goals which they identified and achieved. Table 6.2 shows that only proposed system has achieved all the identified operational goals which proved that the correctness criteria has been achieved by the proposed system. In the next sections, the correctness and accuracy of the proposed system has been proved through (i) mathematical proof, (ii) experimental results and (ii) comparison of results with gold standard ontology and also with existing systems.

### 6.2.2 Data Sets Specifications

For the evaluation of of the proposed system, three data sets are used:

- a. **Data Set from DSpace@MIT:** Massachusetts Institute of Technology (MIT) is a private research university located in Cambridge, Massachusetts, United States. Institutional repository is built at MIT for preserving, sharing, and searching digital research materials. Some portion of MIT's DSpace database is harvested for evaluation purpose. This data set includes 3 main communities, 10 sub-communities, 18 collections and 2,242 items. These items contain 37,032 metadata values.



Table 6.1: The main operational goals

Goal	Description
<b>G1:</b> Building standard ontology by using default heuristic rules	Some default rules are used for building ontology (i.e., an OWL class for every relation, OWL datatype properties for non-foreign key attributes, OWL object properties for foreign key attributes and respective OWL class individual for a relation tuple).
<b>G2:</b> Preserving DSpace data model hierarchy	Hierarchy rules are used to preserve the DSpace data model. Such rules identify hierarchies of classes in the relational schema. <i>rdfs:subClassOf</i> is added to respective class.
<b>G3:</b> Handling weak entities properly during transformation	These rules are used to identify weak entities and their corresponding owner entities. Such relations are still mapped to OWL classes.
<b>G4:</b> Handling multivalued attributes properly during transformation	These rules are used to identify relations that act as multivalued attributes. This kind of rules often assume that a relation that has a composite primary key containing a foreign key referencing a parent relation can be translated to a datatype property with domain the class that corresponds to the parent relation.
<b>G5:</b> Exploiting additional schema constraints	These constraints include restrictions on not null and unique attributes and domain constraints. These are usually mapped to appropriate OWL cardinality axioms and global ( <i>rdfs:range</i> axioms) or local universal quantification axioms (e.g. an <i>owl:allValuesFrom</i> axiom).
<b>G6:</b> Defining correspondences between SQL datatypes and XML Schema Types	These rules are used to define correspondences between SQL datatypes and XML Schema Types, which are typically used as datatypes in RDF and OWL.
<b>G7:</b> Populating ontology with instances from the database	Created ontology is populated with instances from the database.

Table 6.2: Goals fulfilled by different approaches

Approach	Goals						
	G1	G2	G3	G4	G5	G6	G7
Astrova (2007) [1]	✓	✓	✓	-	✓	-	✓
DB2OWL [42, 44]	✓	✓	-	-	✓	✓	✓
DM-2-OWL [43]	✓	✓	-	-	✓	✓	-
R2O [19]	✓	✓	✓	✓	✓	-	-
RDBToOnto [41]	✓	✓	-	-	-	-	✓
ROSEX [15]	✓	✓	✓	-	-	-	-
Data Master [3]	✓	-	✓	-	-	✓	✓
D2R Server [4]	✓	-	✓	-	-	-	✓
Proposed System	✓	✓	✓	✓	✓	✓	✓

- b. **Data Set from DSpace@SEECs:** NUST School of Electrical Engineering and Computer Science (SEECs) built institutional repository (SEECs-IR). SEECs-IR is a digital archive of intellectual outputs created by its community (students, faculty members, and research associates) working in academia and different research projects. Initially they have archived reports of final year projects of undergraduate students and theses/dissertations of postgraduate students. The complete DSpace database is used for the evaluation of the proposed system. This data set contains 1 main community, 14 sub-communities, 12 collections and 533 items. These items contain 10,132 metadata values.
- c. **Example Data Set:** DSpace has been installed on local machine for the testing purposes. This data set has been used as a walk through example for explaining the proposed system. This data set contains 1 main community, 8 sub-communities, 10 collections and 100 items. These items contain 1,679 metadata values.

### 6.3 Mathematical Proof

On the basis of information capacity, the correctness of the proposed system is evaluated. The mappings which preserves information capacity should completely transform the source model into the target model. This target model should be reversible so that it can be transformed back to the original source model [45]. The following definitions have been taken from [45]:

**Definition 1:** Let  $A$  and  $B$  be sets. A mapping (binary relation)  $f : A \rightarrow B$  is **functional** if for any  $a \in A$  there exists at most one  $b \in B$  such that  $f(a) = b$ ;

**injective**, if its inverse is functional; **total**, if it is defined on every element of  $A$ ; and **surjective** (onto) if its inverse is total. A functional, injective, total, surjective mapping is a **bijection**.

**Definition 2:** An information capacity preserving mapping between the instances of two schemas  $S_1$  and  $S_2$  is a total, injective function  $f : I(S_1) \rightarrow I(S_2)$ .

**Definition 3:**  $S_2$  dominates  $S_1$  through  $f$ , iff  $f : I(S_1) \rightarrow I(S_2)$  is an information capacity preserving mapping, such that  $S_1 \preceq S_2$ .

In the proposed system, DSpace database is a source model and ontology is the target model. In the context of different transformation and integration tasks, Miller et al. [45] identified a set of operational goals. According to these goals, the operational goal of the proposed system is:

***The target model (i.e., ontology) can be used to query and view the source model (i.e., relational database).***

To achieve this goal, a total function  $f : I(S_1) \rightarrow I(S_2)$  is needed as defined above, but it is also needed that  $f$  must not lose any information. An instance of  $S_1$  should uniquely determine an instance of  $S_2$ , i.e.,  $f$  should also be injective so that its inverse  $f^{-1}$  is well defined. Therefore,  $f$  must be an information capacity preserving mapping for achieving this goal, therefore,  $S_1 \preceq S_2$ .

### 6.3.1 Lemmas

Five lemmas are provided to show that the transformation is information capacity preserving and is correct by adopting the evaluation technique of [45]. Before defining the lemmas, some notations and concepts are defined which are used in evaluation.

- Let  $T$  be a transformation,  $S_1$  be a source schema (i.e., DSpace database) and  $S_2$  be a target schema (i.e., ontology). For transforming DSpace database into Ontology, an intermediate database has been used so  $T$  is divided into two sub transformations,  $T_1$  and  $T_2$ .  $T_1$  is in charge of creating intermediate database  $D$  from  $S_1$ .  $T_2$  is responsible for transforming  $D$  into  $S_2$ .

- Let  $CM_m, CM_s, CL$  be sets which represent main communities, sub-communities and collections of DSpace data model  $DM$  respectively. Let  $CL_i$  be a set of items owned by this collection.
- $CL_i = \{I_j : I_j \text{ is an item owned by } CL_i\}$
- $CL = \{CL_i : CL_i \text{ is a collection}\}$
- $CM_s = \{CM_{s_i} : CM_{s_i} \text{ is a sub - community}\}$
- $CM_m = \{CM_{m_i} : CM_{m_i} \text{ is a main community}\}$
- Only those communities, sub-communities and collections are transformed which have items because if they do not have any item then there is no need to create ontology classes without instances.

### 6.3.1.1 Lemma 1

Let  $E(DM)$  denotes the set of all entities of some DSpace data model  $DM$  and  $E_i \in E(DM)$ . The categories of the entities are identified by the classification imposed on  $E(DM)$  by  $T_1$ . These categories are correct *iff*

$$\forall E_i \in (CM_m \vee CM_s \vee CL)$$

**Proof:**  $T_1$  divides the  $E(DM)$ , such that each entity  $E_i$  falls exactly into one category. Two properties, *hasChild* and *hasParent*, have been defined for explaining the conditions. Following cases are used to identify correct category of  $E_i$ :

$$\text{Case 1: } E_i \in CM_m \iff \neg(E_i \sqcap \forall \text{hasChild}.\perp) \wedge (E_i \sqcap \forall \text{hasParent}.\perp)$$

**Description:**  $E_i$  is a  $CM_{m_i}$  *iff* (i)  $E_i$  is a subsumer and (ii)  $E_i$  is not a subsumee. It means that the entity  $E_i$  has no parent but it has child(ren).

$$\text{Case 2: } E_i \in CM_s \iff \neg(E_i \sqcap \forall \text{hasChild}.\perp) \wedge \neg(E_i \sqcap \forall \text{hasParent}.\perp)$$

**Description:**  $E_i$  is a  $CM_{s_i}$  *iff* (i)  $E_i$  is a subsumer and (ii)  $E_i$  is also a subsumee. It means that the entity  $E_i$  has both parent and child(ren).

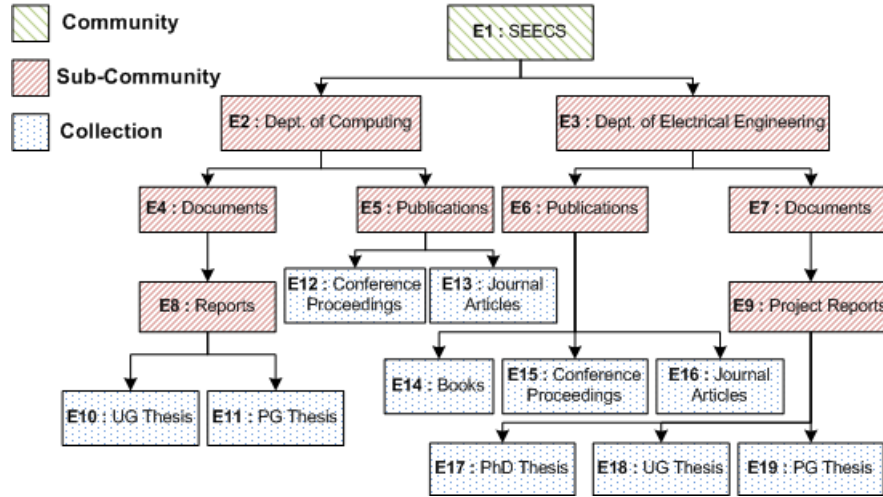


Figure 6.1: Lemma 1: Classification of Data Model entities

### Case 3:

$$E_i \in CL \iff (E_i \sqcap \forall hasChild. \perp) \wedge \neg (E_i \sqcap \forall hasParent. \perp) \wedge (E_i \neq \emptyset)$$

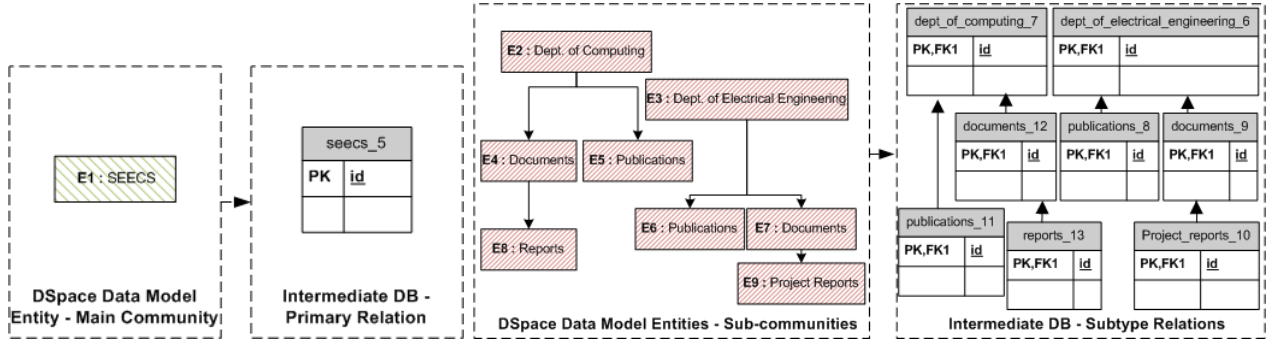
**Description:**  $E_i$  is a collection  $CL_i$  iff (i)  $E_i$  is not a subsumer, (ii)  $E_i$  is a subsumee and (iii)  $E_i$  is not an empty set, it contains all items owned by  $CL_i$ . It means that the entity  $E_i$  has a parent but it has no child(ren). Therefore, this entity is a collection and it contains items.

Figure 6.1 shows the DSpace Data Model  $DM$  of the institute. It contains 19 entities and each entity falls exactly into one category.  $E_1$  has no parent but it has children so it is main community  $CM_m$ . Eight entities  $E_2 - E_9$  have both parents and children so these are sub-communities  $CM_s$ . Remaining ten entities  $E_{10} - E_{19}$  have parents but no children so these are collections  $CL$ . Therefore, the categories are identified by the classification imposed on  $E(DM)$  by  $T_1$ . So  $E_i$  is in one of the categories induced by  $T_1$ .

#### 6.3.1.2 Lemma 2

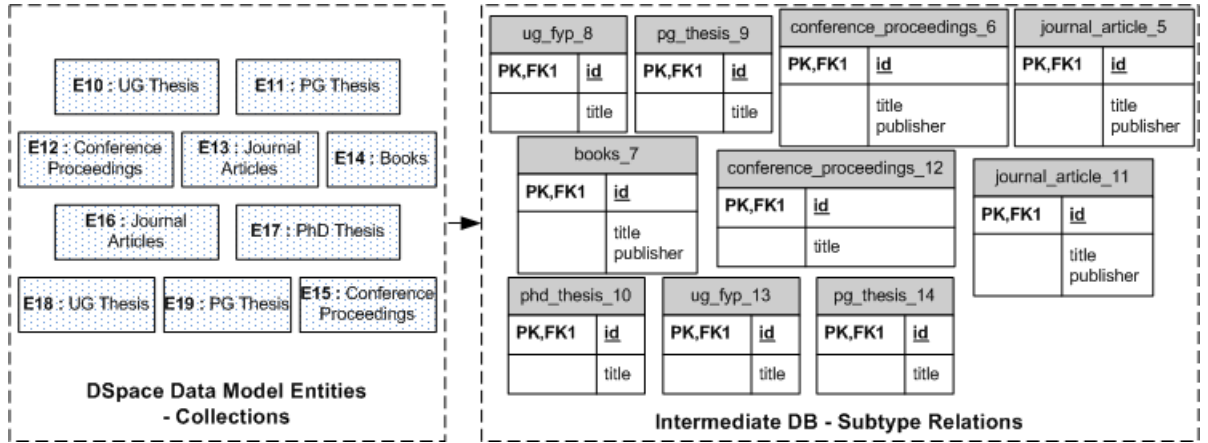
Relations  $R$  of intermediate database  $D$  are created and populated by  $T_1$ .  $T_1$  is correct and information capacity preserving transformation iff

- primary relations  $R_{CM_m}$  of  $D$  are created for  $\forall E_i \in CM_m$ , such that  $R_{CM_m} \subseteq R$
- subtype relations  $R_{CM_s}$  of  $D$  are created for  $\forall E_i \in CM_s$



(a) Lemma 2(a): Primary relation for main community

(b) Lemma 2(b): Subtype relations for sub-communities



(c) Lemma 2(c): Subtype relations for collections

Figure 6.2: Lemma 2: Intermediate database relations for DSpace entities

- c. subtype relations  $R_{CL}$  of  $D$  are created  $\forall E_i \in CL$ , and tuples of  $R_{CL_i}$  are created  $\forall I_i \in CL_i$

Therefore,  $R_{CM_m} \sqcup R_{CM_s} \sqcup R_{CL} = R$

**Proof:** This part of the transformation is total because

- $R_{CM_{m_i}}$  is created as a primary relation of intermediate database for each entity  $E_i$  which comes under the category of main community.  $R_{CM_{m_i}}$  is a primary relation because the primary key of this relation does not contain key of any other relation. Therefore,  $\forall FK_i \subseteq FK, \neg(FK_i \subseteq PK_i)$ .
- $R_{CM_{s_i}}$  is created as a subtype relation of intermediate database for each entity  $E_i$  which comes under the category of sub-community.  $R_{CM_{s_i}}$  is a subtype relation because the primary key of  $R_{CM_{s_i}}$  is a foreign key of parent relation such that  $PK_i = FK_i$ .

- c.  $R_{CL_i}$  is created as a subtype relation of intermediate database for each entity  $E_i$  which comes under the category of collection. The primary key of  $R_{CL_i}$  is a foreign key of parent relation such that  $PK_i = FK_i$  therefore these relations are the subtype relations of the intermediate database. Attributes of the  $R_{CL_i}$  are created for all non-multivalued metadata fields of item  $I_i$ , which is owned by  $CL_i$ . Metadata values of these fields are mapped to the corresponding attributes.

The relations of intermediate database are created for each DSpace entity, the example is shown in Figure 6.2. There is only one main community so one primary relation has been created (i.e., shown in Figure 6.2a). Eight subtype relations are created for sub-communities and ten subtype relations are created for collections (i.e., shown in Figure 6.2b and Figure 6.2c respectively).

The transformation is also injective since its inverse maps:

- all primary relations  $R_{CM_m}$  which do not have any parent relation back to original main communities of  $DM$ ,
- all subtype relations  $R_{CM_s}$  which also have any subtype relation  $R_{CL}$  back to original collections of  $DM$ , and
- all subtype relations  $R_{CL}$  which do not have any subtype relation  $R_{CL}$  back to original collections of  $DM$  and all non-multivalued metadata fields of item  $I_i$  along their values back to their original  $S_1$  tuples because proper hierarchy of collections, and all attributes along their values are maintained by the transformation.

Therefore, this part of transformation is proved to be **total** and **injective**.

### 6.3.1.3 Lemma 3

Let  $A(I_j)$  denotes the set of multivalued attributes of item  $I_j$ .  $A(I_j)$  is further divided into two subsets  $A_C(I_j)$  and  $A_S(I_j)$  which represent the set of complex attributes and simple multivalued attributes respectively such that  $A_C(I_j) \subseteq A(I_j)$  and  $A_S(I_j) \subseteq A(I_j)$ . Here  $A_{C_i} \in A_C(I_j)$  and  $A_{S_i} \in A_S(I_j)$ .  $T_1$  is a correct transformation *iff*

- a. weak primary relations  $R_{cmv}$  are created  $\forall A_{C_i} \in I_j$

- b. weak secondary relations  $R_{smv}$  are created  $\forall A_{S_i} \in I_j$

**Proof:** This part of the transformation is total because

- a.  $R_{cmv_i}$  is created as a weak primary relation of intermediate database for each attribute  $A_{C_i}$  which is a complex attribute of item  $I_j$ . Let  $X$  represents the total attributes of the relation  $R_{cmv_i}$  and  $Y$  represents the non-key attributes.  $R_{cmv_i}$  is a weak primary relation because it fulfills the following criteria (i) the primary key of relation contains key of other relation, and (ii) the relation has non-key attributes. Therefore,  $FK_i \subset PK_i \wedge \exists Y \in X - PK_i$ . Therefore, these relations are the weak primary relations of intermediate database.
- b.  $R_{smv_i}$  is created as a weak secondary relation of intermediate database for each attribute  $A_{S_i}$  which is a simple multivalued attribute of item  $I_j$ . Let  $X$  represents the total attributes of the relation  $R_{smv_i}$  and  $Y$  represents the non-key attributes.  $R_{smv_i}$  is a weak primary relation because it fulfills the following criteria (i) the primary key of relation contains key of other relation, and (ii) the relation has not any non-key attribute. Therefore,  $FK_i \subset PK_i \wedge |X - PK_i| = 0$ . Therefore, these relations are the weak secondary relations of intermediate database.

The transformation is also injective since its inverse, maps all weak primary relations  $R_{cmv}$  and weak secondary relations  $R_{smv}$  back to their original  $S_1$  tuples because all attributes along their values are maintained by the transformation. Therefore, it is concluded that this part of transformation is both **total** and **injective**, thus, it is information capacity preserving transformation.

#### 6.3.1.4 Lemma 4

Given the intermediate database  $D$ , the transformation  $T_2$  is an information capacity preserving transformation *iff*

- a. OWL classes  $C$  are created  $\forall R_{CM_{m_i}}$  and tuples of  $R_{CM_{m_i}}$  are mapped to the instances of  $C_i$ ,
- b. OWL sub-classes  $C_s$  are created  $\forall R_{CM_{s_i}}$  and  $\forall R_{CL_i}$ , and tuples of  $R_{CM_{s_i}}$  and  $R_{CL_i}$  are mapped to the instances of their respective classes.
- c. OWL sub-classes  $C_{cmv}$  are created  $\forall R_{cmv_i}$



- d. Datatype properties are created  $\forall R_{smv_i}$  and added to their respective parent classes.

**Proof:** The transformation is total because

- a. An OWL class  $C_i$  is created for a primary relation  $R_{CM_{m_i}}$ . In a primary relation the attributes can be classified into two categories; (i) non-key attributes, and (ii) primary key attribute. Each non-key attribute is transformed into a *DataTypeProperty*. The primary key attribute of the relation is transformed into a *functional DataTypeProperty* with a *minCardinality* restriction of 1. All the tuples of  $R_{CM_{m_i}}$  are mapped to the instances of  $C_i$ .
- b. An OWL class  $C_{s_i}$  is created for a subtype relation and property *rdf:subClassOf* is added. In a subtype relation the attributes can be classified into two categories; (i) non-key attributes and (ii) primary key attribute (i.e., foreign key). Each non-key attribute is mapped to a *DataTypeProperty*. As the primary key attribute of the relation is also a foreign key, it is transformed into a *functional ObjectProperty* with a *minCardinality* restriction of 1. All the tuples of  $R_{CM_{s_i}}$  and  $R_{CL_i}$  are mapped to the instances of their respective classes.
- c. An OWL class  $C_{cmv}$  is created for a weak primary relation  $R_{cmv_i}$  and property *rdf:subClassOf* is added. In a weak primary relation the attributes can be classified into three categories; (i) non-key attributes, (ii) primary key attribute, and (iii) foreign key (part of primary key) . Each non-key attribute is mapped to a *DataTypeProperty*. The primary key attribute of the relation is transformed into a *functional DataTypeProperty* with a *minCardinality* restriction of 1. As the foreign key attribute of the relation is a part of primary key, it is transformed into a *functional ObjectProperty* with a *minCardinality* restriction of 1. Tuples of the relation are mapped to the class instances.
- d. OWL class is not created for the weak secondary relation  $R_{smv_i}$ . In a weak secondary relation the attributes can be classified into two categories; (i) primary key attribute, and (ii) foreign key (part of primary key) . The primary key of  $R_{smv_i}$  which is not a foreign key is mapped to a *DataTypeProperty* and added to the class which has been created for its parent relation. The primary key of  $R_{smv_i}$  which is also a foreign key is discarded as it has already been mapped to the *functional DatatypeProperty* in the parent relation.

The transformation is also injective since its inverse maps *OWL class* and its properties back to the tuples of source model because information about attribute val-

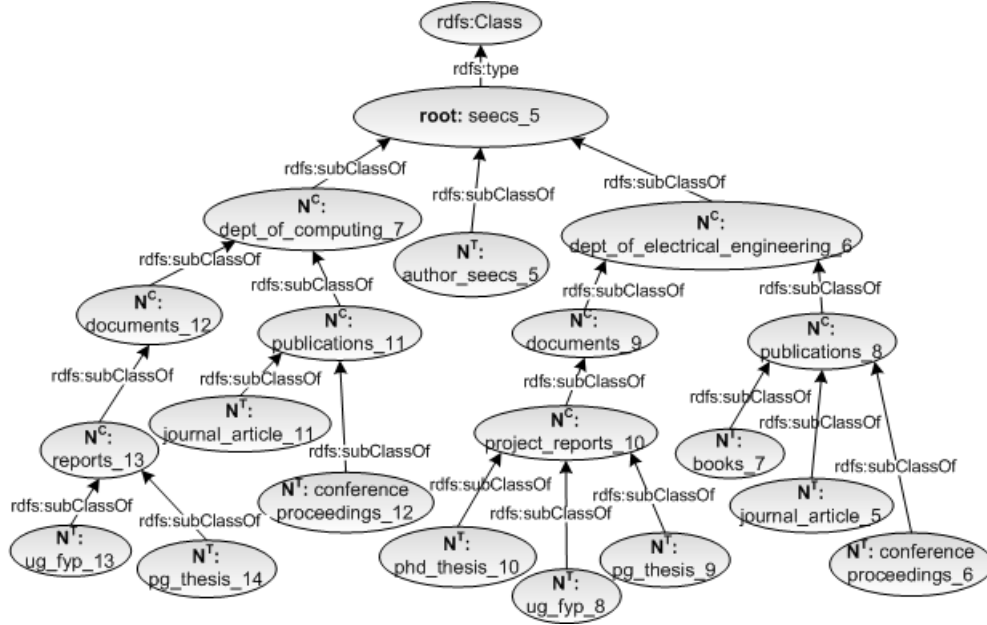


Figure 6.3: A multilevel hierarchy of concepts created for DSpace database

ues and constraint are properly maintained by the transformation. Therefore, it is proved that the transformation is both **total** and **injective** and preserves the information capacity.

### 6.3.1.5 Conclusions of Lemma 1-4

Therefore, these four lemmas proved that transformation  $T$  of relational database  $S_1$  into Ontology  $S_2$  is correct and information capacity preserving transformation. The categories of data model entities  $E(DM)$  are identified such that  $\forall E_i \in (CM_m \vee CM_s \vee CL)$ . A multilevel subsumption hierarchy of concepts  $\mathcal{H}$  is created for these categories.  $\mathcal{H}$  consists of set of nodes:

$$N^H = \{root, N^C, N^T\} \quad (6.1)$$

This hierarchy defines:

- a *root – node*, denoted by *root* which is main parent node,
- a set  $N^C$  of *complex – nodes* which have child(ren) node(s), and
- a set  $N^T$  of *terminal – nodes* which have no child(ren) node(s).

According to DSpace data model these nodes are used to represent following concepts:

- *root* represents  $CM_{m_i}$
- $N^C$  represents  $\forall CM_s \sqsubseteq CM_{m_i}$
- $N^T$  represents  $\left[ \sum_{j=1}^q (CL \sqsubseteq CM_{s_j} \wedge CL \neq \emptyset) \sqsubseteq CM_{m_i} \right] \sqcup [\forall A_C \sqsubseteq CM_{m_i}]$ , here  $q$  is total number of sub-communities of  $CM_{m_i}$  which directly subsume the collection(s) means these sub-communities do not have any sub-community in their child(ren).

The output of transformation  $T$  is target model  $S_2$  (i.e., ontology).  $S_2$  contains a multilevel subsumption hierarchy of concepts  $\mathcal{H}$ , which is a set of nodes  $N^H$ , here;

- total number of  $N^H$  = total number of roots (i.e., *root*) = total number of main communities  $CM_m$
- total number of  $N^C$  = total number of sub-communities  $CM_s$
- total number of  $N^T$  = total number of non-empty collections  $CL$  + total number of complex attributes  $A_C$

In the example scenario, There are one main community, eight sub-communities, ten collections and four multivalued attributes. Therefore,  $\mathcal{H}$  contains only one node  $N^H$  because there is one main community so one multilevel subsumption hierarchy has been created.  $N^H$  contains one *root* node corresponding to main community, 8  $N^C$  nodes corresponding to sub-communities and 11  $N^T$  nodes corresponding to collections and complex attributes. This hierarchy of concepts is shown in Figure 6.3.

### 6.3.1.6 Lemma 5

Given two ontologies  $\mathcal{O}1$  (i.e., source ontology) and  $\mathcal{O}2$  (i.e., target ontology) and an alignment  $A$  between these two ontologies. Let  $C1$  and  $C2$  be the concepts of  $\mathcal{O}1$  and  $\mathcal{O}2$  respectively such that  $C1_i \in C1$  and  $C2_j \in C2$ .  $A$  is a pair of  $(C1_i, C2_j)$  which means that  $C1_i \equiv C2_j$ . Let instance matching is a triple  $(I1_k, I2_l, R)$ , where  $R$  is a relation between instances  $I1_k \in C1_i$  and  $I2_l \in C2_j$ .  $R$  is taken from the set  $\{\equiv, \perp, \}$  for equivalence and disjointness respectively. Data translation is correct *iff*

- $I1_k$  is given the same URI as  $I2_l$ , if  $(I1_k, I2_l, \equiv)$ ,

Table 6.3: Data Set specifications

	Communities			Collections		Items	Metadata Fields			Metadata values
	Main	Sub (Total)	Sub (Empty)	Total	Empty		Non- multivalued	multivalued	Complex	
<b>Data Set 1</b>	3	10	0	18	3	2,242	11	5	3	37,032
<b>Data Set 2</b>	1	14	5	12	5	533	12	2	1	10,132
<b>Data Set 3</b>	1	8	0	10	0	100	9	3	1	1,679

- b. new instances of  $C2_j$  are created  $\forall (I1_k, I2_l, \perp)$  and linked with the existing data

**Proof:** Data translation is functional as:

- a. The triple  $(I1_k, I2_l, \equiv)$  shows that  $I1_k \equiv I2_l$ , therefore  $I1_k$  is given the URI of the  $I2_l$ , which identifies the two as the same instance. For example,  $I1_k$  is an instance of contributor class (i.e.,  $C1_i$ ) which is mapped to the person class (i.e.,  $C2_j$ ).  $C2_j$  has the same author (i.e.,  $I2_l$ ). Therefore, instead of creating new instance of  $C2_j$ , the URI of  $I2_l$  has been given to  $I1_k$ .
- b. The triple  $\forall (I1_k, I2_l, \perp)$  shows that  $I1_k$  is not matching with any instance of  $C2_j$ . A proper VIVO URI is given to  $I1_k$ .  $I1_k$  is finally linked with the existing data. For example, there is a new publication (i.e.,  $I1_k$ ) and its author (i.e.,  $I2_l$ ) is already in  $\mathcal{O}2$ . Therefore,  $I1_k$  is linked with  $I2_l$ .

## 6.4 Experimental Results

For the evaluation of the proposed system, three data sets are used. The specifications of these data sets have been shown in Table 6.3. The results are validated by comparing the generated ontology with (i) the intermediate database, and (ii) the DSpace data model and its database contents. Ontology classes, properties and instances are compared with (i) the intermediate relations, attributes and tuples respectively, and (ii) the DSpace data model entities (i.e., communities, sub-communities and collections), items' metadata fields and items' metadata values respectively. To prove that a multilevel subsumption hierarchy of concepts is complete and correct according to input data sets, following formulas are used which are extracted from the mathematical proof provided in above section:

- total number of  $N^H$  = total number of roots (i.e., *root*) = total number of main communities  $CM_m$ , which have non-empty collection(s).
- total number of  $N^C$  = total number of sub-communities  $CM_s$  =

$$\sum_{i=1}^p [CM_s \sqsubseteq CM_{m_i}] \quad (6.2)$$

- total number of  $N^T$  = total number of non-empty collections  $CL$  + total number of complex attributes  $A_C$  =

$$\sum_{i=1}^p \left[ \sum_{j=1}^q (CL \sqsubseteq CM_{s_j} \wedge CL \neq \emptyset) \sqsubseteq CM_{m_i} \right] \sqcup \sum_{i=1}^m [A_C \sqsubseteq CM_{m_i}] \quad (6.3)$$

here,  $p$  is total number of main communities and  $q$  is total number of sub-communities of  $CM_{m_i}$  which directly subsume the collection(s) means these sub-communities do not have any sub-community in their child(ren).

After evaluating the generated ontology, its instances are compared with the individuals which are translated and expressed with respect to the VIVO ontology.

## 6.4.1 Data Set 1

### 6.4.1.1 Relation to Ontology Transformation

This data has been harvested from MIT's DSpace. 18 collections have been selected for data harvesting. 3 collections are empty as they do not have any item. Total 2,242 items have been harvested in other 15 collections. This data set contains 3 main communities and 10 sub-communities. These items contain 11 non-multivalued, 3 multivalued and 5 complex metadata fields, and these contain 37,032 metadata values. Table 6.4 shows the result obtained at the end of Part 1 (i.e., Intermediate database creation) and Part 2 (i.e., ontology creation) of proposed system. Table shows the schema of intermediate database and ontology, created for the DSpace data model entities and items' metadata fields, and the intermediate database tuples and ontology instances, created for the items' metadata values. 3 main OWL classes have been created for main communities and 28 sub-classes have been created for sub-communities, collections and complex attributes. 19 Datatype Properties are created for non-multivalued, multivalued and non-key attributes. 6 functional Datatype properties with min cardinality 1 are created for primary key

Table 6.4: Data Set 1: Summary of Results - Ontology Creation

DSpace Data Model (Source Model)		Extracted Intermediate Database		OWL Ontology (Target Model)	
Communities, Collections, Items, Metadata Fields & Metadata values		Entities, Attributes & Tuples		OWL Classes, Properties & Instances	
Main Communities	3	Primary Entities	3	Primary Classes	3
Sub-communities (with non-empty collection(s))	10	Sub-entities	10	Sub-classes	10
Non-empty Collections	15	Sub-entities	15	Sub-classes	15
Metadata Fields (complex)	3	Weak Primary entities	3	Sub-classes	3
Metadata Fields (multivalued)	5	Weak Secondary Entities	5	Datatype Property	5
Metadata Fields (not multivalued)	11	Non Key Attributes	14	Datatype Property	14
		Primary Keys (not foreign key)	11	Functional Datatype Property with minCardinality=1	6
		Primary Keys (foreign keys)	33	Functional Object Property with minCardinality=1	28
Total items in 15 collections	2,242	Total tuples of $R_{CL}$ (relations corresponding to collections)	2,242	Total instances of $C_{CL}$ (classes corresponding to $R_{CL}$ )	2,242

attributes which are not foreign keys. 28 functional Object properties with min cardinality 1 are created for those primary key attributes which are also foreign keys. The extracted formulas are applied on the generated ontology which proved that a multilevel subsumption hierarchy of concepts  $\mathcal{H}$  is complete and correct according to input DSpace database. Therefore, it is proved that;

- $\mathcal{H} = \{N_1^H, N_2^H, N_3^H\}$ , Total number of root nodes  $N^H = 3$ , which are equal to the total number of main communities  $CM_m$
- $N_1^H = \{root, N^C, N^T\}$ ,  $N^C = \{N_1^C, N_2^C, N_3^C, \dots, N_8^C\}$ ,  $N^T = \{N_1^T, N_2^T, N_3^T, \dots, N_{12}^T\}$
- $N_2^H = \{root, N^C, N^T\}$ ,  $N^C = \{N_1^C\}$ ,  $N^T = \{N_1^T, N_2^T\}$
- $N_3^H = \{root, N^C, N^T\}$ ,  $N^C = \{N_1^C\}$ ,  $N^T = \{N_1^T, N_2^T, N_3^T, N_4^T\}$
- Total number of nodes  $N^C$  are calculated by using equation 6.2 =

$$\sum_{i=1}^p [CM_s \sqsubseteq CM_{m_i}]$$

Table 6.5: Data Set 1: Summary of Results - Data Translation

Generated Ontology (Source Model)		VIVO Ontology (Target Model)	
<i>Classes &amp; Instances</i>		<i>Identified VIVO individuals</i>	
Total classes	31	AcademicDepartment	3
Total Instances of $C_{CL}$ (classes corresponding to collections)	2,242	Laboratory	5
		Report	272
		Thesis	1,969
		Article	1

here  $p = 3$  therefore,

$$[8 + 1 + 1] = 10$$

which is equal to the total number of sub-communities  $CM_s$

- Total number of nodes  $N^T$  are calculated by using equation 6.3 =

$$\sum_{i=1}^p \left[ \sum_{j=1}^q (CL \sqsubseteq CM_{s_j} \wedge CL \neq \emptyset) \sqsubseteq CM_{m_i} \right] \sqcup \sum_{i=1}^m [A_C \sqsubseteq CM_{m_i}]$$

here  $p = 3$ , for  $i = 1 \rightarrow q = 7$ ,  $i = 2 \rightarrow q = 1$ ,  $i = 3 \rightarrow q = 1$  therefore,

$$[\{(1 + 2 + 1 + 1 + 1 + 1 + 4) + (1) + (3)\} + (1 + 1 + 1)] = 18$$

which is equal to the total number of non-empty collections  $CL$  + total number of complex multivalued attributes  $A_C$

#### 6.4.1.2 Ontology Alignment and Data Translation

Ontology has been created for the data set 1, as discussed above, but this ontology can not be imported into VIVO directly as it is not in the required format. VIVO ontology is needed to be populated with the data of the extracted ontology, for that purpose a semi-automatic approach has been used for translating data into the format of VIVO ontology. Total 31 OWL classes have been created for this data set, as discussed above. An alignment  $A$  has been used for the instance matching,  $A$  is a pair  $(C1_i, C2_j)$  which means that  $C1_i \equiv C2_j$ , for example:

- $contributor\_department\_of\_architecture \equiv Person$
- $department\_of\_architecture \equiv AcademicDepartment$

Table 6.6: Data Set 2: Summary of Results

DSpace Data Model (Source Model)		Extracted Intermediate Database		OWL Ontology (Target Model)	
Communities, Collections, Items, Metadata Fields & Metadata values		Entities, Attributes & Tuples		OWL Classes, Properties and Instances	
Main Communities	1	Primary Entities	1	Primary Classes	1
Sub-communities (with non-empty collections)	9	Sub-entities	9	Sub-classes	9
Non-empty Collections	7	Sub-entities	7	Sub-classes	7
Metadata Fields (complex)	1	Weak Primary Entities	1	Sub-classes	1
Metadata Fields (multivalued)	2	Weak Secondary Entities	2	Datatype Property	2
Metadata Fields (not multivalued)	12	Non Key Attributes	13	Datatype Property	13
		Primary Keys (not foreign key)	4	Functional Datatype Property with min cardinality=1	2
		Primary Keys (foreign keys)	19	Functional Object Property with min cardinality=1	17
Total items in 15 collections	533	Total tuples in the $R_{CL}$ (relations corresponding to collections)	533	Total instances of the $C_{CL}$ (classes corresponding to $R_{CL}$ )	533

- $aerospace\_control\_laboratory \equiv Laboratory$
- $aerospace\_control\_laboratory\_technical\_reports \equiv Report$

Table 6.5 shows the result obtained at the end of Part 3 (i.e., Ontology alignment and data translation) of the proposed system. 3 classes are mapped on AcademicDepartment class and 5 classes are mapped on Laboratory class. Therefore, 3 individuals of Academic department class and 5 individuals of Laboratory class have been created. 8 classes, having 1,969 instances, are mapped on Thesis class, 5 classes, having 272 instances, are mapped on Report class and 1 class having 1 instance, is mapped on Article class. Therefore, 1,969 individuals of Thesis class, 272 individuals of Report class and 1 individual of Article class have been created. The individuals of Person class have been created for their authors.



## 6.4.2 Data Set 2

### 6.4.2.1 Relation to Ontology Transformation

This data set contains the DSpace database of an institute, SEECS. This data set contains 1 main community and 14 sub-communities in which 5 sub-communities contain empty collections. There are total 12 collections in which 5 collections are empty and other 7 collections contain 533 items. These items contain 12 non-multivalued, 1 multivalued and 2 complex metadata fields, and these contain 10,132 metadata values. Table 6.6 shows the results obtained at the end of Part 1 (i.e., Intermediate database creation) and Part 2 (i.e., ontology creation) of proposed system. Table shows the schema of intermediate database and ontology, created for the DSpace data model entities and items' metadata fields, and the intermediate database tuples and ontology instances, created for the items' metadata values. 1 main OWL class has been created for main community and 17 sub-classes have been created for sub-communities, collections and complex attributes. 15 Datatype Properties are created for non-multivalued, multivalued and non-key attributes. 2 functional Datatype properties with min cardinality 1 are created for primary key attributes which are not foreign keys. 17 functional Object properties with min cardinality 1 are created for those primary key attributes which are also foreign keys. By applying the extracted formulas on the generated ontology it is proved that a multilevel subsumption hierarchy of concepts  $\mathcal{H}$  is complete and correct according to input DSpace database. Therefore, it is proved that;

- $\mathcal{H} = \{N_1^H\}$ , Total number of nodes  $N^H = 1$ , which is equal to the total number of main communities  $CM_m$
- $N_1^H = \{root, N^C, N^T\}$ ,  $N^C = \{N_1^C, N_2^C, N_3^C, \dots, N_9^C\}$ ,  $N^T = \{N_1^T, N_2^T, N_3^T, \dots, N_8^T\}$
- Total number of nodes  $N^C$  are calculated by using equation 6.2 =

$$\sum_{i=1}^p [CM_s \sqsubseteq CM_{m_i}]$$

here  $p = 1$  therefore,  $[9] = 9$ , which is equal to the total number of sub-communities  $CM_s$

- Total number of nodes  $N^T$  are calculated by using equation 6.3 =

$$\sum_{i=1}^p \left[ \sum_{j=1}^q (CL \sqsubseteq CM_{s_j} \wedge CL \neq \emptyset) \sqsubseteq CM_{m_i} \right] \sqcup \sum_{i=1}^m [A_C \sqsubseteq CM_{m_i}]$$

Table 6.7: Data Set 2: Summary of Results - Data Translation

Generated Ontology (Source Model)		VIVO Ontology (Target Model)	
<i>Classes &amp; Instances</i>		<i>Identified VIVO individuals</i>	
Total classes	18	Schools	1
Total Instances of $C_{CL}$ (classes corresponding to collections)	533	AcademicDepartments	2
		Reports	70
		Theses	463

here  $p = 1$ , for  $i = 1 \rightarrow q = 7$  therefore,

$$[(1 + 1 + 1 + 1 + 1 + 1 + 1) + (1)] = 8$$

which is equal to the total number of non-empty collections  $CL$  + total number of complex multivalued attributes  $A_C$

#### 6.4.2.2 Ontology Alignment and Data Translation

Total 18 OWL classes have been created for this data set, as discussed above. An alignment  $A$  has been used for the instance matching,  $A$  is a pair  $(C1_i, C2_j)$  which means that  $C1_i \equiv C2_j$ , for example:

- $department\_of\_computing \equiv AcademicDepartment$
- $thesis\_reports \equiv Thesis$
- $fyp\_reports \equiv Report$
- $contributor\_school\_of\_electrical\_engineering\_computer\_science \equiv Person$

Table 6.7 shows the result obtained at the end of Part 3 (i.e., Ontology alignment and data translation) of the proposed system. 1 class is mapped on School class and 2 classes are mapped on AcademicDepartment class. Therefore, 1 individual of School class and 2 individuals of Academic department class have been created. 4 classes are mapped on Thesis class, these classes contain 463 instances. Therefore, 463 individuals of Thesis class have been created. 3 classes are mapped on Report class, having 70 instances. Therefore, 70 individuals of Report class have been created.

Table 6.8: Data Set 3: Summary of Results

DSpace Data Model (Source Model)		Extracted Intermediate Database		OWL Ontology (Target Model)	
Communities, Collections, Items, Metadata Fields & Metadata values		Entities, Attributes & Tuples		OWL Classes, Properties & Instances	
Main Communities	1	Primary Entities	1	Primary Classes	1
Sub-communities (with non-empty collection(s))	8	Sub-entities	8	Sub-classes	8
Non-empty Collections	10	Sub-entities	10	Sub-classes	10
Metadata Fields (complex)	1	Weak Primary Entities	1	Sub-classes	1
Metadata Fields (multivalued)	3	Weak Secondary Entities	3	Datatype Property	3
Metadata Fields (not multivalued)	9	Non Key Attributes	10	Datatype Property	10
		Primary Keys (not foreign key)	5	Functional Datatype Property with min cardinality=1	2
		Primary Keys (foreign keys)	22	Functional Object Property with min cardinality=1	19
Total items in 15 collections	100	Total tuples of $R_{CL}$ (relations corresponding to collections)	100	Total instances of $C_{CL}$ (classes corresponding to $R_{CL}$ )	100

### 6.4.3 Data Set 3

#### 6.4.3.1 Relation to Ontology Transformation

This data set contains an example DSpace database which is used to elaborate the proposed system throughout this thesis. This data set contains 1 main community and 8 sub-communities. There are 10 collections which contain 100 items. These items contain 9 non-multivalued, 1 multivalued and 3 complex metadata fields, and these contain 1,679 metadata values. Table 6.8 shows the results obtained at the end of Part 1 (i.e., Intermediate database creation) and Part 2 (i.e., ontology creation) of proposed system. Table 6.8 shows the schema of intermediate database and ontology, created for the DSpace data model entities and items' metadata fields, and the intermediate database tuples and ontology instances, created for the items' metadata values. 1 main OWL class has been created for main community and 19 sub-classes have been created for sub-communities, collections and complex attributes. 13 Datatype Properties are created for non-multivalued, multivalued and non-key attributes. 2 functional Datatype properties with min cardinality 1 are created for primary key attributes which are not foreign keys. 19 functional Object properties with min cardinality 1 are created for those primary key attributes which

are also foreign keys. The extracted formulas are applied on the generated ontology which proved that a multilevel subsumption hierarchy of concepts  $\mathcal{H}$  is complete and correct according to input DSpace database. Therefore, it has been proved that;

- $\mathcal{H} = \{N_1^H\}$ , Total number of nodes  $N^H = 1$ , which is equal to the total number of main communities  $CM_m$
- $N_1^H = \{root, N^C, N^T\}$ ,  $N^C = \{N_1^C, N_2^C, N_3^C, \dots, N_8^C\}$ ,  $N^T = \{N_1^T, N_2^T, N_3^T, \dots, N_{11}^T\}$
- Total number of nodes  $N^C$  are calculated by using equation 6.2 =

$$\sum_{i=1}^p [CM_s \sqsubseteq CM_{m_i}]$$

here  $p = 1$  therefore,  $[8] = 8$ , which is equal to the total number of sub-communities  $CM_s$

- Total number of nodes  $N^T$  are calculated by using equation 6.3 =

$$\sum_{i=1}^p \left[ \sum_{j=1}^q (CL \sqsubseteq CM_{s_j} \wedge CL \neq \emptyset) \sqsubseteq CM_{m_i} \right] \sqcup \sum_{i=1}^m [A_C \sqsubseteq CM_{m_i}]$$

here  $p = 1$ , for  $i = 1 \rightarrow q = 4$  therefore,

$$[(3 + 3 + 2 + 2) + (1)] = 11$$

which is equal to the total number of non-empty collections  $CL$  + total number of complex multivalued attributes  $A_C$ .

#### 6.4.3.2 Ontology Alignment and Data Translation

Total 20 OWL classes have been created for this data set, as discussed above. An alignment  $A$  has been used for the instance matching,  $A$  is a pair  $(C1_i, C2_j)$  which means that  $C1_i \equiv C2_j$ , for example:

- *department\_of\_electrical\_engineering*  $\equiv$  *AcademicDepartment*
- *pg\_thesis*  $\equiv$  *Thesis*
- *ug\_fyp*  $\equiv$  *Report*
- *journal\_article*  $\equiv$  *Article*

Table 6.9: Data Set 3: Summary of Results - Data Translation

Generated Ontology (Source Model)		VIVO Ontology (Target Model)	
<i>Classes &amp; Instances</i>		<i>Identified VIVO individuals</i>	
Total classes	20	School	1
Total Instances of $C_{CL}$ (classes corresponding to collections)	100	AcademicDepartment	2
		Report	32
		Thesis	23
		Article	12
		Conference Paper	18
		Book	15

- $conference\_proceedings \equiv ConferencePaper$
- $contributor\_seecs \equiv Person$

Table 6.9 shows the result obtained at the end of Part 3 (i.e., Ontology alignment and data translation) of the proposed system. 1 class is mapped on School class and 2 classes are mapped on AcademicDepartment class. Therefore, 1 individual of School class and 2 individuals of Academic department class have been created. 3 classes are mapped on Thesis class, these classes contain 23 instances. Therefore, 23 individuals of Thesis class have been created. 2 classes are mapped on Report class, having 32 instances. Therefore, 32 individuals of Report class have been created. 2 classes are mapped on Article class and 2 classes are mapped on ConferencePaper, these classes contain 12 and 18 instances respectively. Therefore, 12 individuals of Article class and 18 individuals of ConferencePaper class have been created.

## 6.5 Comparison with Gold Standards

An ontology is complex structured, so it should be evaluated separately at different levels. The evaluation of an ontology involves the following levels: (i) lexical term layer, (ii) hierarchy/taxonomy level, (iii) Context/application level, (iv) syntactic level, and (v) Structure/design level. The following approaches can be used for evaluating the ontology construction capability: (1) application based evaluation, (2) human evaluation, (3) data-driven evaluation or (4) comparing with a gold standard [46].

In this section, the measures are used for evaluating the ontology at lexical

and hierarchy levels.

- *Lexical term layer*: This involves the evaluation of concepts, instances, facts, etc. included in the ontology. It involves the string similarity measures and comparison with various data sources.
- *Hierarchy (taxonomy) level*: Evaluation on this level concerns with a hierarchical is-a relation between concepts. The is-a relationship is particularly important for the preservation of DSpace data model hierarchy.

The generated ontology has been compare with gold standard due to frequent evaluations and its feasibility in practice. The precision and recall measures are most widely used to evaluate and compare ontologies [47]. The extended precision and recall measures are used for the gold standard based evaluation of ontologies. The lexical precision and recall measures are used for evaluating the ontology at lexical term layer. and for the concept hierarchy, taxonomic precision, recall and f-measures are used, as defined in [48, 49]. The simplified definition of a core ontology is used which only contains the lexical term layer and the concept hierarchy. Similarly to [50], a core ontology is defined as follows:

**Definition 4:** *The structure  $\mathcal{O} := (C, root, \leq c)$  is a core ontology consisting of a set of concept identifiers,  $C$  and a designated root concept representing top element of the partial order  $\leq c$  on  $C$  such that  $\forall c \in C : c \leq root$ . This partial order is called concept hierarchy or taxonomy.*

### 6.5.1 Lexical Precision & Recall

Let  $\mathcal{O}_C$  is a computed core ontology and  $\mathcal{O}_R$  is a a reference ontology, the lexical precision ( $LP$ ) and lexical recall ( $LR$ ) are defined as follows [48]:

$$LP(\mathcal{O}_C, \mathcal{O}_R) = \frac{|C_C \cap C_R|}{|C_C|} \quad (6.4)$$

$$LR(\mathcal{O}_C, \mathcal{O}_R) = \frac{|C_C \cap C_R|}{|C_R|} \quad (6.5)$$

### 6.5.2 Taxonomic Precision & Recall

Taxonomic Precision ( $TP$ ) and Taxonomic Recall ( $TR$ ) measures can be based on Semantic Cotopy ( $SC$ ) or Common Semantic Cotopy ( $CSC$ ).  $SC$  define all super and sub concept ( $c$ ) relations in ontology ( $\mathcal{O}$ ).  $SC$  can be defined as:

$$SC(c, \mathcal{O}) := \{c_i | c_i \in C \wedge (c_i \leq c \vee c \leq c_i)\} \quad (6.6)$$

The influence of lexical precision in the taxonomic measurement can be avoided by using  $CSC$ .  $CSC$  excludes all concepts which are not also available in the set of concepts of other ontology.  $CSC$  can be defined as:

$$CSC(c, \mathcal{O}_C, \mathcal{O}_R) := \{c_i | c_i \in C_1 \cap C_2 \wedge (c_i <_1 c \vee c <_1 c_i)\} \quad (6.7)$$

The measures  $TP_{SC}$  and  $TR_{SC}$  do not allow a separate evaluation of lexical term layer and concept hierarchy. For evaluation scenarios where a thorough analysis of the learned ontologies is needed the measures  $TP_{CSC}$  and  $TR_{CSC}$  are better suited [48].  $TP_{CSC}$  and  $TR_{CSC}$  can be defined as:

$$TP_{CSC}(\mathcal{O}_C, \mathcal{O}_R) := \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tp_{csc}(c, c, \mathcal{O}_C, \mathcal{O}_R) \quad (6.8)$$

and

$$TR_{CSC}(\mathcal{O}_C, \mathcal{O}_R) := TP_{CSC}(\mathcal{O}_R, \mathcal{O}_C) \quad (6.9)$$

Taxonomic F-measure ( $TF$ ) calculates the harmonic mean of  $TP_{CSC}$  and  $TR_{CSC}$  which used for balancing value of them.  $TF$  can be computed as:

$$TF_{CSC}(\mathcal{O}_C, \mathcal{O}_R) := \frac{2 \cdot TP_{CSC}(\mathcal{O}_C, \mathcal{O}_R) \cdot TR_{CSC}(\mathcal{O}_C, \mathcal{O}_R)}{TP_{CSC}(\mathcal{O}_C, \mathcal{O}_R) + TR_{CSC}(\mathcal{O}_C, \mathcal{O}_R)} \quad (6.10)$$

### 6.5.3 Evaluation of Relation to Ontology Transformation

In this section the measures presented in above sections are used to evaluate the results. The gold standard ontology  $\mathcal{O}_R$  is created by the domain expert for the evaluation of the proposed system. DSpace data model varies from organization to organization, therefore gold standard ontology is created according to source DSpace database. This ontology contains the classes for DSpace data model entities and complex multivalued attributes. It has same multilevel subsumption hierarchy as DSpace data model has. Three different data sets are used for testing the pro-

posed system so separate gold standard ontology has been created for every data set. The gold standard ontologies  $\mathcal{O}_{\mathcal{R}_1}$ ,  $\mathcal{O}_{\mathcal{R}_2}$  and  $\mathcal{O}_{\mathcal{R}_3}$  are used for comparison in evaluation for data set 1, 2 and 3 respectively.  $\mathcal{O}_{C_{1A}}$ ,  $\mathcal{O}_{C_{2A}}$  and  $\mathcal{O}_{C_{3A}}$  are produced ontologies by the proposed system. The non-leaf concept is used to determine *CSC* as used in [49]. The ontologies are evaluated with a common semantic cotopy based measure which is better suited for evaluating a concept hierarchy. The proposed technique, for transforming DSpace database into ontology, has been compared with Astrova [1], DataMaster [3] and D2R server [4]. These techniques are applied on DSpace database but they generated totally incorrect ontologies. They used their basic transformation rules and created OWL classes for DSpace database relation whereas correct classes should be extracted from the contents of these tables. Therefore, lexical and taxonomic precision and recall, and taxonomic f-measure scores of existing techniques are zero percent (0%). Therefore, these techniques are then applied on the intermediate database which is created by the proposed system and then these results are compared with the results of the proposed system.

### 6.5.3.1 Comparison with Astrova

The technique proposed by Astrova [1] has been selected for comparison because it is the most similar technique which is using almost the same transformation rules which are used in the proposed transformation system. But this system is not publicly or commercially available, therefore, its rules are thoroughly studied and then these are applied on the intermediate database and the ontologies  $\mathcal{O}_{C_{1B}}$ ,  $\mathcal{O}_{C_{2B}}$  and  $\mathcal{O}_{C_{3B}}$  are generated for data set 1, 2 and 3 respectively. These ontologies are compared and evaluated with their corresponding gold standard ontologies.

For the data set 1, gold standard ontology  $\mathcal{O}_{\mathcal{R}_1}$  is used for comparison with  $\mathcal{O}_{C_{1A}}$  (generated by the proposed system) and  $\mathcal{O}_{C_{1B}}$  (generated by Astrova). These ontologies are shown in Figure 6.4. Figure 6.4a shows the gold standard ontology which is created by the domain expert for the comparison with implemented system. Ontology generated by the proposed system has exactly the same concept hierarchy as shown in Figure 6.4b, whereas ontology generated by applying the rules of Astrova has 5 extra classes as shown in Figure 6.4c. Astrova is not using multivalued property rules therefore instead of creating datatype properties it created OWL classes for these relations. Similarly ontologies for data set 2 and 3 are shown in Figure 6.5 and 6.6 respectively.

Table 6.10 shows the common semantic cotopies *CSC* and computed taxo-



Table 6.10: Common semantic cotopies  $CSC$  and local taxonomic precision  $tp_{CSC}$  for non-leaf concepts of ontologies  $\mathcal{O}_{C1A}$  and  $\mathcal{O}_{C1B}$

Non-leaf concept	CSC ( $c, \mathcal{O}_{R1}, \mathcal{O}_{C1}$ )	CSC ( $c, \mathcal{O}_{C1A}, \mathcal{O}_{R1}$ )	CSC ( $c, \mathcal{O}_{C1B2}, \mathcal{O}_{R1}$ )	$tp_{csc}$ ( $c, c, \mathcal{O}_{C1A}, \mathcal{O}_{R1}$ )	$tp_{csc}$ ( $c, c, \mathcal{O}_{C1B}, \mathcal{O}_{R1}$ )
$CM_{m1}$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7, CL_8,$ $CL_9, CL_{10}, CL_{11}$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7, CL_8,$ $CL_9, CL_{10}, CL_{11}$	$A_1, A_4, A_5, A_6,$ $CL_1, CL_2, CL_3,$ $CL_4, CL_5, CL_6,$ $CL_7, CL_8, CL_9,$ $CL_{10}, CL_{11}$	12/12=1.00	12/15=0.80
$CM_{m2}$	$A_2, CL_{12}$	$A_2, CL_{12}$	$A_2, A_7, CL_{12}$	2/2=1.00	2/3=0.67
$CM_{m3}$	$A_3, CL_{13}, CL_{14},$ $CL_{15}$	$A_3, CL_{13}, CL_{14},$ $CL_{15}$	$A_3, A_8, CL_{13},$ $CL_{14}, CL_{15}$	4/4=1.00	4/5=0.80
$CM_{s1}$	$CL_1$	$CL_1$	$CL_1$	1/1=1.00	1/1=1.00
$CM_{s2}$	$CL_2, CL_3$	$CL_2, CL_3$	$CL_2, CL_3$	2/2=1.00	2/2=1.00
$CM_{s3}$	$CL_4$	$CL_4$	$CL_4$	1/1=1.00	1/1=1.00
$CM_{s4}$	$CL_4$	$CL_4$	$CL_4$	1/1=1.00	1/1=1.00
$CM_{s5}$	$CL_5$	$CL_5$	$CL_5$	1/1=1.00	1/1=1.00
$CM_{s6}$	$CL_6$	$CL_6$	$CL_6$	4/4=1.00	4/4=1.00
$CM_{s7}$	$CL_7$	$CL_7$	$CL_7$	1/1=1.00	1/1=1.00
$CM_{s8}$	$CL_8, CL_9, CL_{10},$ $CL_{11}$	$CL_8, CL_9, CL_{10},$ $CL_{11}$	$CL_8, CL_9, CL_{10},$ $CL_{11}$	1/1=1.00	1/1=1.00
$CM_{s9}$	$CL_{12}$	$CL_{12}$	$CL_{12}$	1/1=1.00	1/1=1.00
$CM_{s10}$	$CL_{13}, CL_{14}, CL_{15}$	$CL_{13}, CL_{14}, CL_{15}$	$CL_{13}, CL_{14}, CL_{15}$	3/3=1.00	3/3=1.00

$CM_{m1}$ : department\_of\_aeronautics\_and\_astronautics (main community);  
 $CM_{m2}$ : department\_of\_architecture (main community);  $CM_{m3}$ : department\_of\_biology (main community);  $CM_{s1}$ : humans\_and\_automation\_laboratory (sub-community);  $CM_{s2}$ : aerospace\_control\_laboratory (sub-community);  $CM_{s3}$ : faculty\_and\_researchers (sub-community);  $CM_{s4}$ : sheila\_e\_widnall (sub-community);  $CM_{s5}$ : man\_vehicle\_laboratory (sub-community);  $CM_{s6}$ : aerospace\_computational\_design\_laboratory (sub-community);  $CM_{s7}$ : flight\_transportation\_laboratory (sub-community);  $CM_{s8}$ : theses\_aeronautics\_and\_astronautics (sub-community);  $CM_{s9}$ : theses\_department\_of\_architecture (sub-community);  $CM_{s10}$ : theses\_biology (sub-community);  
 $CL_1$ : hal\_reports (collection);  $CL_2$ : aerospace\_control\_laboratory\_technical\_reports (collection);  $CL_3$ : aerospace\_control\_laboratory\_manuscripts (collection);  $CL_4$ : selected\_publications (collection);  $CL_5$ : manuscripts (collection);  $CL_6$ : aerospace\_computational\_design\_laboratory\_technical\_reports (collection);  $CL_7$ : flight\_transportation\_laboratory\_reports (collection);  $CL_8$ : aeronautics\_and\_astronautics\_engineers\_degree (collection);  $CL_9$ : aeronautics\_and\_astronautics\_phd\_scd (collection);  $CL_{10}$ : aeronautics\_and\_astronautics\_bachelors\_degree (collection);  $CL_{11}$ : aeronautics\_and\_astronautics\_masters\_degree (collection);  $CL_{12}$ : architecture\_masters\_degree (collection);  $CL_{13}$ : biology\_phd\_scd (collection);  $CL_{14}$ : biology\_bachelors\_degree (collection);  $CL_{15}$ : biology\_masters\_degree (collection);  $A_1$ : contributor\_department\_of\_aeronautics\_and\_astronautics (complex attribute);  $A_2$ : contributor\_department\_of\_architecture (complex attribute);  $A_3$ : contributor\_department\_of\_biology (complex attribute);

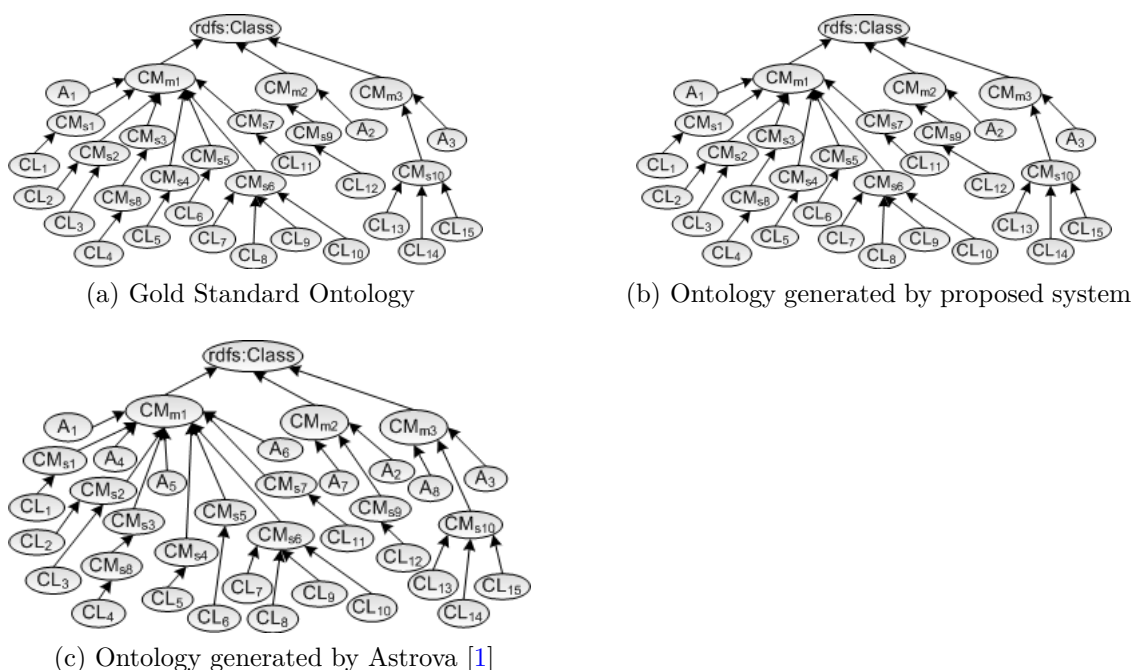


Figure 6.4: Ontologies for data set 1

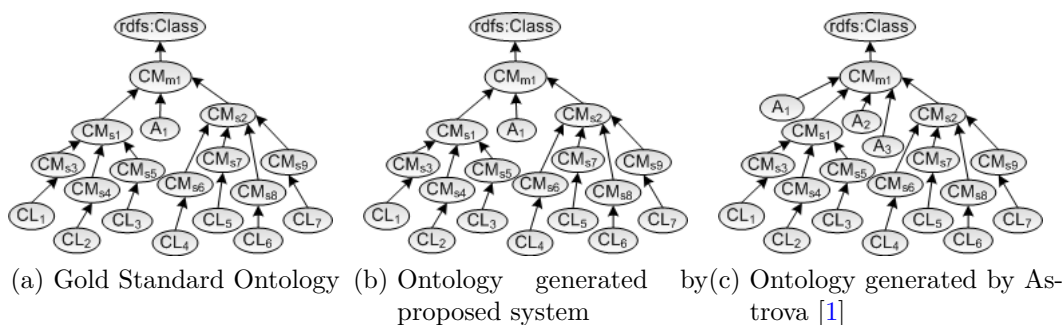


Figure 6.5: Ontologies for data set 2

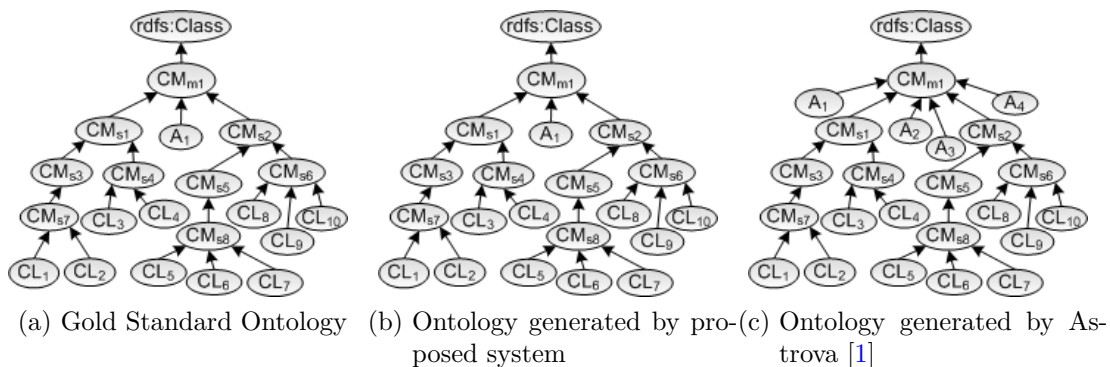


Figure 6.6: Ontologies for data set 3

Table 6.11: Common semantic cotopies  $CSC$  and local taxonomic precision  $tp_{CSC}$  for non-leaf concepts of ontologies  $\mathcal{O}_{C2A}$  and  $\mathcal{O}_{C2B}$

Non-leaf concept	CSC ( $c, \mathcal{O}_{R2}, \mathcal{O}_{C2}$ )	CSC ( $c, \mathcal{O}_{C2A}, \mathcal{O}_{R2}$ )	CSC ( $c, \mathcal{O}_{C2B}, \mathcal{O}_{R2}$ )	$tp_{csc}$ ( $c, c, \mathcal{O}_{C2A}, \mathcal{O}_{R2}$ )	$tp_{csc}$ ( $c, c, \mathcal{O}_{C2B}, \mathcal{O}_{R2}$ )
$CM_{m1}$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7$	$A_1, A_2, A_3, CL_1,$ $CL_2, CL_3, CL_4,$ $CL_5, CL_6, CL_7$	8/8=1.00	8/10=0.80
$CM_{s1}$	$CL_1, CL_2, CL_3$	$CL_1, CL_2, CL_3$	$CL_1, CL_2, CL_3$	3/3=1.00	3/3=1.00
$CM_{s2}$	$CL_1$	$CL_1$	$CL_1$	4/4=1.00	4/4=1.00
$CM_{s3}$	$CL_2$	$CL_2$	$CL_2$	1/1=1.00	1/1=1.00
$CM_{s4}$	$CL_3$	$CL_3$	$CL_3$	1/1=1.00	1/1=1.00
$CM_{s5}$	$CL_4, CL_5, CL_6,$ $CL_7$	$CL_4, CL_5, CL_6,$ $CL_7$	$CL_4, CL_5, CL_6,$ $CL_7$	1/1=1.00	1/1=1.00
$CM_{s6}$	$CL_4$	$CL_4$	$CL_4$	1/1=1.00	1/1=1.00
$CM_{s7}$	$CL_5$	$CL_5$	$CL_5$	1/1=1.00	1/1=1.00
$CM_{s8}$	$CL_6$	$CL_6$	$CL_6$	1/1=1.00	1/1=1.00
$CM_{s9}$	$CL_7$	$CL_7$	$CL_7$	1/1=1.00	1/1=1.00

$CM_{m1}$ : school\_of\_electrical\_engineering\_and\_computer\_science\_seecs (main community);  $CM_{s1}$ : department\_of\_computing (sub-community);  $CM_{s2}$ : ms\_information\_technology (sub-community);  $CM_{s3}$ : bs\_information\_technology (sub-community);  $CM_{s4}$ : pgd\_information\_technology (sub-community);  $CM_{s5}$ : department\_of\_electrical\_engineering (sub-community);  $CM_{s6}$ : ms\_electrical\_engineering (sub-community);  $CM_{s7}$ : ms\_computer\_system\_engineering (sub-community);  $CM_{s8}$ : be\_information\_and\_communication\_systems\_engineering (sub-community);  $CM_{s9}$ : be\_electrical\_engineering (sub-community);  $CL_1$ : thesis\_reports\_5 (collection);  $CL_2$ : fyp\_reports\_6 (collection);  $CL_3$ : thesis\_reports\_15 (collection);  $CL_4$ : thesis\_reports\_24 (collection);  $CL_5$ : thesis\_reports\_25 (collection);  $CL_6$ : fyp\_reports\_13 (collection);  $CL_7$ : fyp\_reports\_12 (collection);  $A_1$ : contributor\_school\_of\_electrical\_engineering\_and\_computer\_science\_seecs (complex attribute);

Table 6.12: Common semantic cotopies  $CSC$  and local taxonomic precision  $tp_{CSC}$  for non-leaf concepts of ontologies  $\mathcal{O}_{C3A}$  and  $\mathcal{O}_{C3B}$

Non-leaf concept	CSC ( $\mathbf{c}, \mathcal{O}_{R3}, \mathcal{O}_{C3}$ )	CSC ( $\mathbf{c}, \mathcal{O}_{C3A}, \mathcal{O}_{R3}$ )	CSC ( $\mathbf{c}, \mathcal{O}_{C3B}, \mathcal{O}_{R3}$ )	$tp_{csc}$ ( $\mathbf{c}, \mathbf{c}, \mathcal{O}_{C3A}, \mathcal{O}_{R3}$ )	$tp_{csc}$ ( $\mathbf{c}, \mathbf{c}, \mathcal{O}_{C3B}, \mathcal{O}_{R3}$ )
$CM_{m1}$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7, CL_8,$ $CL_9, CL_{10}$	$A_1, CL_1, CL_2,$ $CL_3, CL_4, CL_5,$ $CL_6, CL_7, CL_8,$ $CL_9, CL_{10}$	$A_1, A_2, A_3, A_4,$ $CL_1, CL_2, CL_3,$ $CL_4, CL_5, CL_6,$ $CL_7, CL_8, CL_9,$ $CL_{10}$	11/11=1.00	11/14=0.79
$CM_{s1}$	$CL_1, CL_2, CL_3,$ $CL_4$	$CL_1, CL_2, CL_3,$ $CL_4$	$CL_1, CL_2, CL_3,$ $CL_4$	4/4=1.00	4/4=1.00
$CM_{s2}$	$CL_1, CL_2$	$CL_1, CL_2$	$CL_1, CL_2$	6/6=1.00	6/6=1.00
$CM_{s3}$	$CL_3, CL_4$	$CL_3, CL_4$	$CL_3, CL_4$	2/2=1.00	2/2=1.00
$CM_{s4}$	$CL_3, CL_4$	$CL_3, CL_4$	$CL_3, CL_4$	2/2=1.00	2/2=1.00
$CM_{s5}$	$CL_5, CL_6, CL_7,$ $CL_8, CL_9, CL_{10}$	$CL_5, CL_6, CL_7,$ $CL_8, CL_9, CL_{10}$	$CL_5, CL_6, CL_7,$ $CL_8, CL_9, CL_{10}$	3/3=1.00	3/3=1.00
$CM_{s6}$	$CL_5, CL_6, CL_7$	$CL_5, CL_6, CL_7$	$CL_5, CL_6, CL_7$	3/3=1.00	3/3=1.00
$CM_{s7}$	$CL_5, CL_6, CL_7$	$CL_5, CL_6, CL_7$	$CL_5, CL_6, CL_7$	2/2=1.00	2/2=1.00
$CM_{s8}$	$CL_8, CL_9, CL_{10}$	$CL_8, CL_9, CL_{10}$	$CL_8, CL_9, CL_{10}$	2/2=1.00	2/2=1.00

$CM_{m1}$ : seecs (main community);  $CM_{s1}$ : dept\_of\_computing (sub-community);  $CM_{s2}$ : publications\_11 (sub-community);  $CM_{s3}$ : documents\_12 (sub-community);  $CM_{s4}$ : reports\_13 (sub-community);  $CM_{s5}$ : dept\_of\_electrical\_engineering (sub-community);  $CM_{s6}$ : documents\_9 (sub-community);  $CM_{s7}$ : project\_reports\_10 (sub-community);  $CM_{s8}$ : publications\_8 (sub-community);  $CL_1$ : conference\_proceedings\_12 (collection);  $CL_2$ : journal\_article\_11 (collection);  $CL_3$ : pg\_thesis\_14 (collection);  $CL_4$ : ug\_fyp\_13 (collection);  $CL_5$ : phd\_thesis\_10 (collection);  $CL_6$ : ug\_fyp\_8 (collection);  $CL_7$ : pg\_thesis\_9 (collection);  $CL_8$ : books\_7 (collection);  $CL_9$ : journal\_article\_5 (collection);  $CL_{10}$ : conference\_proceedings\_6 (collection);  $A_1$ : contributor\_seecs (complex attribute);

Table 6.13: Evaluation of the ontologies (i.e. shown in Figure 6.4, 6.5 and 6.6) with a common semantic cotopy based measure

	Compare $\mathcal{O}_{R_1}$ with		Compare $\mathcal{O}_{R_2}$ with		Compare $\mathcal{O}_{R_3}$ with	
	$\mathcal{O}_{C_{1A}}$	$\mathcal{O}_{C_{1B}}$	$\mathcal{O}_{C_{2A}}$	$\mathcal{O}_{C_{2B}}$	$\mathcal{O}_{C_{3A}}$	$\mathcal{O}_{C_{3B}}$
$LP$	100%	86.11%	100%	90%	100%	86.96%
$LR$	100%	100%	100%	100%	100%	100%
$TP_{CSC}$	100%	94.38%	100%	98%	100%	97.67%
$TR_{CSC}$	100%	94.38%	100%	98%	100%	97.67%
$TF_{CSC}$	100%	94.38%	100%	98%	100%	97.67%

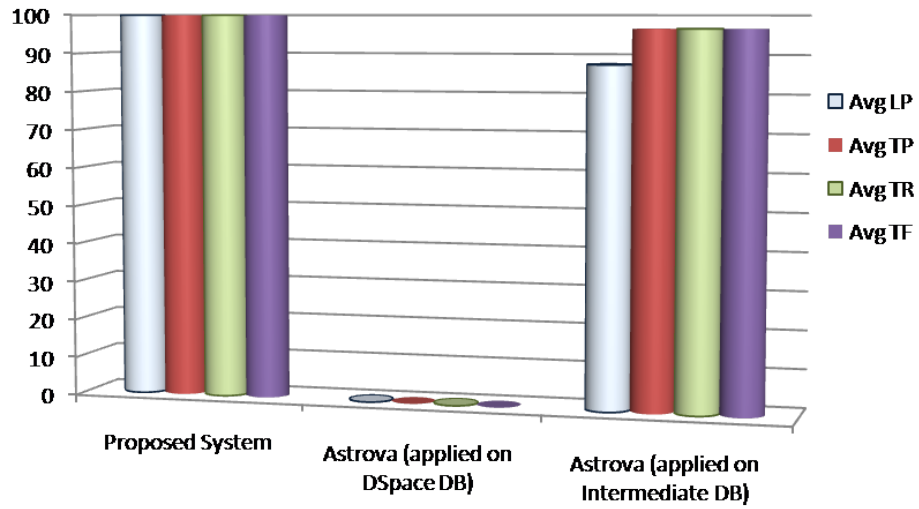


Figure 6.7: The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and Astrova [1].

nomic precision  $TP_{CSC}$  for ontologies  $\mathcal{O}_{C_{1A}}$  and  $\mathcal{O}_{C_{1B}}$ . As the non-leaf concepts in  $\mathcal{O}_{C_{1A}}$  and  $\mathcal{O}_{C_{1B}}$  are perfectly matching with non-leaf concepts in  $\mathcal{O}_{R_1}$ , then the taxonomic recall  $TR_{CSC} = TP_{CSC}$ . In  $\mathcal{O}_{R_1}$  there are total 12 leaf concepts for  $CM_{m_1}$ , the proposed system identified correct concepts whereas Astrova created three extra OWL classes as presented in Table 6.10. Similarly the evaluation of ontologies  $\mathcal{O}_{C_{2A}}$  and  $\mathcal{O}_{C_{2B}}$ , and  $\mathcal{O}_{C_{3A}}$  and  $\mathcal{O}_{C_{3B}}$  are presented in Table 6.11 and 6.12 respectively. Table 6.13 presents the end results of all 3 data sets by the comparison of the average lexical precision  $LP$ , taxonomic precision  $TP_{CSC}$ , recall  $TR_{CSC}$  and f-measure  $TF_{CSC}$  scores between the proposed system and Astrova's proposed technique.  $LP$ ,  $TP_{CSC}$ ,  $TR_{CSC}$  and  $TF_{CSC}$  scores of the proposed technique are 100%. It is evident from the results shown in Table 6.13 that the proposed technique is more accurate as compared to Astrova's. This is because Astrova does not consider multivalued property rule. The results showed that the accuracy of Astrova is decreasing with the increase of number of multivalued attributes in data set. The graph (shown in Figure 6.7) presents the average experimental results of the proposed technique and Astrova's technique. It gave zero percent (0%) precision by applying Astrova's

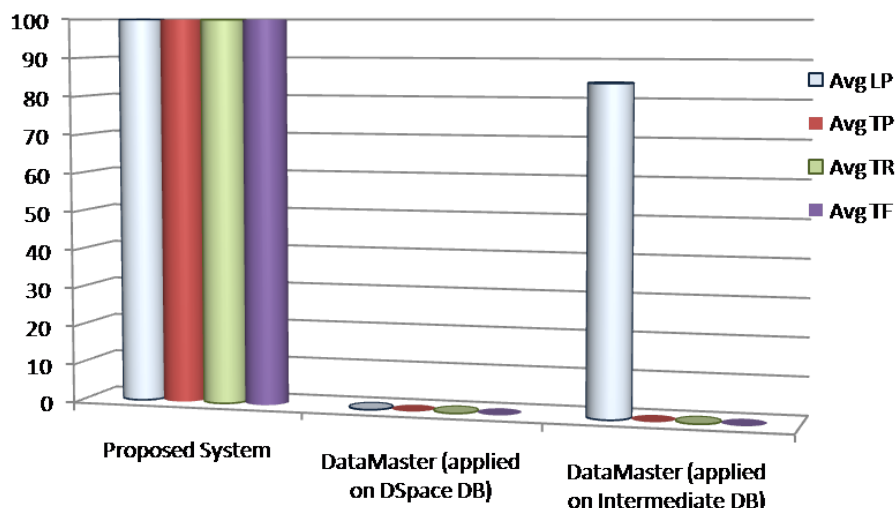


Figure 6.8: The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and DataMaster [3].

technique directly on DSpace database. Therefore, it is then applied on intermediate database which generated much better results with 87.69 % lexical precision and 96.68% taxonomic precision, recall and f-measure scores whereas the proposed system generated 100% score on all three data sets.

### 6.5.3.2 Comparison with DataMaster

The proposed system has also been compared with DataMaster [3]. DataMaster is a Protégé<sup>1</sup> plug-in for importing schema structure and data from relational databases into Protégé. It is the most widely used tool for building ontologies from relational databases. The graph (shown in Figure 6.8) presents the average experimental results of the proposed technique and DataMaster. First this tool is directly applied on DSpace database which generated zero percent (0%) precision. Then it is applied on intermediate database which generated 84.27% lexical precision, still not 100% because DataMaster is not using multivalued property rules so it generated some extra classes. DataMaster is also not creating the concept hierarchy, therefore taxonomic precision, recall and f-measure scores are zero percent (0%) whereas the proposed system generated 100% score on all three data sets.

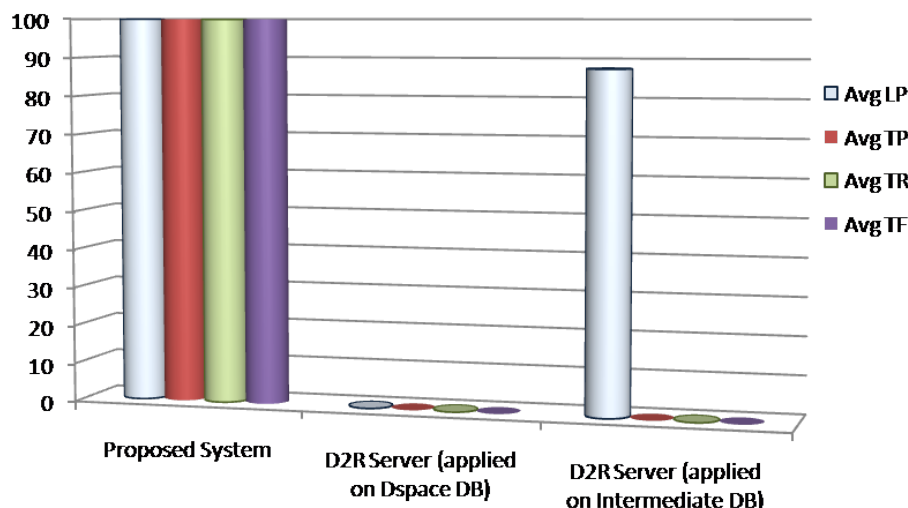


Figure 6.9: The average Lexical and Taxonomic precision, Taxonomic recall and Taxonomic f-measure scores of the proposed system and D2R Server [4].

### 6.5.3.3 Comparison with D2R Server

The results are also compared with D2R server [4]. The graph (shown in Figure ??) presents the average experimental results of the proposed technique and D2R server. First this tool is applied directly on DSpace database which generated zero percent (0%) precision. Then it is applied on intermediate database which generated 87.69% lexical precision, still not 100% because D2R server has also created some extra classes. D2R server is also not creating concept hierarchy, therefore taxonomic precision, recall and f-measure scores are zero percent (0%) whereas the proposed system generated 100% score.

### 6.5.4 Evaluation of Ontology Data Translation

The results of the ontology data translation are evaluated using precision, recall and f-measure measurements. *Precision* is the number of correct results divided by the number of all returned results and *Recall* is the number of correct results divided by the number of results that should have been returned. In this project, they are calculated as follows:

$$Precision = \frac{|T \cap R|}{R} \quad (6.11)$$

<sup>1</sup>A free, open source ontology editor and knowledge-base framework.  
<http://protege.stanford.edu/>

Table 6.14: Data set 1: Evaluation of the data translation process

VIVO classes	Instances					
	<i>E</i>	<i>R</i>	<i>T</i>	<i>Precision</i>	<i>Recall</i>	<i>F-measure</i>
<i>AcademicDepartment</i>	3	3	3	1.00	1.00	1.00
<i>Laboratory</i>	5	5	5	1.00	1.00	1.00
<i>Report</i>	272	272	272	1.00	1.00	1.00
<i>Thesis</i>	1969	1969	1969	1.00	1.00	1.00
<i>Article</i>	1	1	1	1.00	1.00	1.00
	Properties					
<i>AcademicDepartment</i>	12	12	12	1.00	1.00	1.00
<i>Laboratory</i>	20	20	20	1.00	1.00	1.00
<i>Report</i>	1360	1088	816	0.75	0.60	0.67
<i>Thesis</i>	9845	7876	5907	0.75	0.60	0.67
<i>Article</i>	6	5	3	0.60	0.50	0.55

Table 6.15: Data set 2: Evaluation of the data translation process

VIVO classes	Instances					
	<i>E</i>	<i>R</i>	<i>T</i>	<i>Precision</i>	<i>Recall</i>	<i>F-measure</i>
<i>School</i>	1	1	1	1.00	1.00	1.00
<i>AcademicDepartment</i>	2	2	2	1.00	1.00	1.00
<i>Report</i>	70	70	70	1.00	1.00	1.00
<i>Thesis</i>	463	463	463	1.00	1.00	1.00
	Properties					
<i>School</i>	4	4	4	1.00	1.00	1.00
<i>AcademicDepartment</i>	8	8	8	1.00	1.00	1.00
<i>Report</i>	350	350	210	0.60	0.60	0.60
<i>Thesis</i>	2315	2315	1389	0.60	0.60	0.60

$$Recall = \frac{|T \cap R|}{E} \quad (6.12)$$

where  $T$  represents the translated instances/properties,  $R$  represents the retrieved instances/properties and  $E$  represents the expected instances/properties that should have been translated.

The  $F - measure$  is defined as the harmonic mean of precision and recall. It is calculated as follows:

$$F - measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (6.13)$$

The instances and properties of the mapped classes are matched and expressed with respect to the target ontology. The Tables 6.14, 6.15 and 6.16 show



Table 6.16: Data set 3: Evaluation of the data translation process

VIVO classes	Instances					
	<i>E</i>	<i>R</i>	<i>T</i>	<i>Precision</i>	<i>Recall</i>	<i>F-measure</i>
<i>School</i>	1	1	1	1.00	1.00	1.00
<i>Academic Department</i>	2	2	2	1.00	1.00	1.00
<i>Report</i>	32	32	32	1.00	1.00	1.00
<i>Thesis</i>	23	23	23	1.00	1.00	1.00
<i>Article</i>	12	12	12	1.00	1.00	1.00
<i>Conference Paper</i>	18	18	18	1.00	1.00	1.00
<i>Book</i>	15	15	15	1.00	1.00	1.00
	Properties					
<i>School</i>	4	4	4	1.00	1.00	1.00
<i>Academic Department</i>	8	8	8	1.00	1.00	1.00
<i>Report</i>	160	160	96	0.60	0.60	0.60
<i>Thesis</i>	115	115	69	0.60	0.60	0.60
<i>Article</i>	72	72	36	0.50	0.50	0.50
<i>Conference Paper</i>	108	108	54	0.50	0.50	0.50
<i>Book</i>	90	90	45	0.50	0.50	0.50

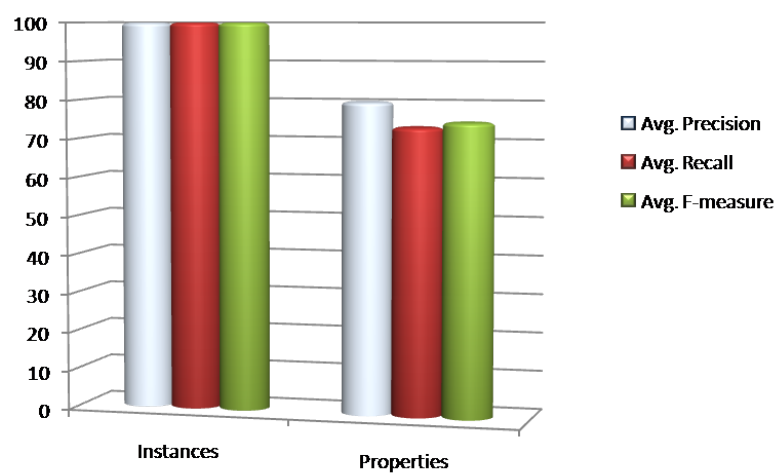


Figure 6.10: Result Evaluation of Ontology Data Translation

the results after evaluating the translated instances and properties of data set 1, 2 and 3 respectively. The tables illustrate that instance translation gives the best precision which is 100 percent, this is due to the reason that system translated all the instances properly and expressed with respect to the target ontology. These instances are transformed into the instances of corresponding mapped classes which are later used to populate the VIVO ontology (i.e., target ontology). Whereas in the translation of properties, it is observed that the translated properties are lesser than the extracted and expected ones meaning it causes some loss of information, which becomes the reason to lower the precision and recall scores. The graph (i.e., shown in Figure 6.10) depicts the average precision, recall and f-measure scores. The graph shows that the precision is 100% in case of instance translation whereas it is 80% in translating properties of these instances.

## Chapter 7

### Conclusions and Future Work

In this chapter the research work carried out under this thesis has been concluded and contribution to the research community is also described. At the end, some directions are suggested where this work can be extended in future.

#### 7.1 Discussion

Internet and semantic web technologies have enabled academics to find online research materials with increasing speed and accuracy. They have enabled academics to make connections with each other. Whereas, institutional digital repositories are often built to serve a specific institution's community of users. Mostly existing institutional repositories (IRs) are using relational database schema for maintaining the metadata of their digital contents. They might need to interact with the many other information systems that exist to manage research activities within the institution or outside. The semantic web and relational database worlds and their developed infrastructures are based on different data models, semantics and query languages. Thus, it is crucial to provide interoperability and integration mechanisms to bridge the gap between the semantic web and relational database worlds. To process the data in semantic context, a relational database is transformed into ontology. Ontology provides a natural way to annotate data and metadata and to capture the domain knowledge and it improves the availability of semantically rich content on the Web. The semantic web technologies provide standard representations for meaningful linkage across different sets of data. The Semantic Web Application (SWA) provides self-describing data via shared ontologies which is also readable by machines

and does simple reasoning to categorize and find associations. The use of SWA in integrating the different institutional repositories metadata facilitates users' search, access, and retrieval of learning resources. The adoption and implementation of semantic web technologies enable ontology-facilitated sharing and reuse of learning resources by giving users access to a web of content that might otherwise necessitate discovering and exploring multiple websites or institutional repositories.

The main promising feature of IRs is their flexible data models that can be customized to arrange the digital documents in a repository according to the organizational structure of an institute. The data model of an organization's IR is not directly converted into IR database schema, but the data model schema is maintained as values in the comprehensive database schema of the IR. The schema of IRs databases is nested schema i.e. a schema is embedded in another schema. In other words, an IR database schema is not a normalized schema with respect to the data model.

A substantial amount of research has already been done to transform a relational database (i.e. schema and its data) into ontology. However the existing transformation systems are only capable to transform a normalized relational database into ontology. They cannot produce accurate result if they are applied on IR databases because their schema is nested schema not a normalized schema. Since the data model is hidden in the IR database schema, it is required to extract the data model from the IR database schema prior to transforming it into ontology, so, it makes the transformation complicated and different from the typical transformation tasks. Therefore, first of all it is essential to identify the data model of an institute from IR database and to extract its metadata and then to transform it into ontology.

After building the ontologies, a key issue is to enable interoperability among different ontologies and to integrate them into the semantic web-based network. Ontology interoperability is a key factor essential for aligning and integrating distributed ontological resources. It can be achieved by identifying or establishing semantic correspondence between entities (i.e., classes and properties) among multiple ontologies.

The proposed system first of all identifies the data model of an institute from IR database and builds a normalized relational schema for the data model of the institute. Then metadata of the repository is extracted to populate this produced schema to build an intermediate database. Once a normalized relational database is obtained, then relational to ontology transformation techniques are applied on this

intermediate database to transform it into ontology. After that instances from the generated ontology are transformed into corresponding data or instances expressed in target ontology. The classes from both source and target ontologies are extracted and simple mappings between these classes are generated by the user. Then the individuals of these mapped classes are matched and proper URIs are given to each individual. These individuals are linked with their respective VIVO classes. Finally, an RDF, having VIVO individuals, is generated. The system has mainly three modules: (i) Metadata Extraction; (ii) Relation to Ontology Transformation; (iii) Ontology Alignment and Data Translation.

## 7.2 Contribution

As IRs are often built to serve a specific institution's community of users, a key issue is to integrate the metadata of different IRs into the semantic web-based network to help faculty, researchers, and students for discovering common interests and make connections. As mostly existing institutional repositories (IRs) are using relational database schema for maintaining the metadata of their digital contents, it is crucial to provide interoperability and integration mechanisms to bridge the gap between the semantic web and relational database worlds. The adoption and implementation of semantic web technologies enable ontology-facilitated sharing and reuse of learning resources. In recognition of need for integrating the metadata of institutional repositories into the semantic web, the system has been proposed and developed which integrates the metadata of different IRs into the semantic web-based network to help faculty, researchers, and students for discovering common interests and make connections.

The main contribution of this work is to integrate metadata of institutional repositories into the semantic web applications such as VIVO to enable sharing and reusing of existing information. The distinguishing features of the proposed system are (i) identifying the data model of an IR; (ii) extracting metadata of the repository; (iii) creating proper hierarchy of parent and child classes of ontology to preserve the data model hierarchy, (iv) generating mappings between ontologies, and (v) transforming data or instances from source ontology into corresponding data or instances expressed in target ontology. The system has been implemented in Java language and Jena API is used for ontology creation. Experimental results demonstrate that the transformation is correct and the system preserves information capacity.

### 7.3 Future Work

Currently the focus is on DSpace database because it is the most widely used open source institutional repository software worldwide. The metadata stored in the DSpace database is transformed, in future this work can be extended for different IR softwares e.g. EPrints, Invenio, Archimede, etc. for integrating more data into the semantic web.

The ontology alignment and transformation features can also be enhanced to maintain more consistency among ontology instances. This work can be extended by integrating WordNet vocabulary to overcome the problem of synonym, homonym, polysemy, etc. during instance matching and ontology population.

## References

- [1] I. Astrova, N. Korda, and A. Kalja, “Rules for mapping sql relational databases to owl ontologies,” in *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, (Corfu Island in Greece), pp. 415–424, October 2007.
- [2] Q. Trinh, K. Barker, and R. Alhajj, “Rdb2ont: A tool for generating owl ontologies from relational database systems,” in *Proceedings of Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW)*, (Guadeloupe, French Caribbean), pp. 170–178, February 2006.
- [3] C. Nyulas, M. OConnor, and S. Tu, “Datamaster—a plug-in for importing schemas and data from relational databases into protege,” in *Proceedings of the 10th International Protege Conference*, (Budapest, Hungary, <http://protege.stanford.edu/conference/2007>), July 2007.
- [4] C. Bizer and R. Cyganiak, “D2r server-publishing relational databases on the semantic web (poster),” in *5th International Semantic Web Conference (ISWC 2006)*, (Athens, USA), p. 26, November 2006.
- [5] J. Futrelle, J. Gaynor, J. Plutchak, *et al.*, “Semantic middleware for e-science knowledge spaces,” *Concurrency and Computation: Practice and Experience*, vol. 23, no. 17, pp. 2107–2117, 2011.
- [6] R. Russell and M. Day, “Institutional repository interaction with research users: a review of current practice,” *New review of academic librarianship*, vol. 16, no. S1, pp. 116–131, 2010.
- [7] T. Tiropanis, H. C. Davis, *et al.*, “Semantic technologies for learning and teach-

- ing in the web 2.0 era,” *IEEE Intelligent Systems*, vol. 24, no. 6, pp. 49–53, 2009.
- [8] D. Krafft, N. Cappadona, B. Caruso, *et al.*, “Vivo: Enabling national networking of scientists,” in *Proceedings of Web Science Conference 2010 (WebSci’10)*, (Raleigh, NC), pp. 1310–1313, April 2010.
- [9] J. McCue, K. Chiang, B. Lowe, *et al.*, “Vivo: connecting people, creating a virtual life sciences community,” *D-Lib Magazine*, vol. 13, pp. 1–16, July/August 2007.
- [10] J. Unbehauen, S. Hellmann, S. Auer, and C. Stadler, “Knowledge extraction from structured sources,” *Search Computing, LNCS*, vol. 7538, pp. 34–52, 2012.
- [11] H. Wache, T. Voegelé, U. Visser, *et al.*, “Ontology-based integration of information—a survey of existing approaches,” in *Proceedings of IJCAI-01 workshop: ontologies and information sharing*, (Washington, USA), pp. 108–117, August 2001.
- [12] R. Tansley, M. Bass, D. Stuve, *et al.*, “The dspace institutional digital repository system: current functionality,” in *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, (Houston, Texas, USA), pp. 87–97, May 2003.
- [13] J. Thakuria, “Building an institutional repository with dspace,” in *Proceedings of 6th Convention Planner*, (Nagaland, India), pp. 102–114, November 2008.
- [14] C. de Laborda and S. Conrad, “Relational owl: a data and schema representation format based on owl,” in *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, (Newcastle, Australia), pp. 89–96, January/February 2005.
- [15] C. Curino, G. Orsi, E. Panigati, and L. Tanca, “Accessing and documenting relational databases through owl ontologies,” in *Proceedings of International Conference Flexible Query Answering Systems (FQAS)*, (Roskilde, Denmark), pp. 431–442, Springer, October 2009.
- [16] C. Bizer and A. Seaborne, “D2rq-treating non-rdf databases as virtual rdf graphs (poster),” in *3rd International Semantic Web Conference (ISWC 2004)*, (Hiroshima, Japan), p. 26, November 2004.



- [17] C. Blakeley, “Virtuoso rdf views getting started guide.” <http://www.openlinksw.co.uk/virtuoso/Whitepapers>, 2007.
- [18] S. Auer, S. Dietzold, J. Lehmann, *et al.*, “Triplify: light-weight linked data publication from relational databases,” in *Proceedings of the 18th international conference on world wide web*, (Madrid, Spain), pp. 621–630, ACM, April 2009.
- [19] S. Khan and K. Sonia, “R2o: Relation to ontology transformation system,” *Journal of Information & Knowledge Management (JIKM)*, vol. 10, no. 01, pp. 71–89, 2011.
- [20] C. A. Lynch, “Institutional repositories: essential infrastructure for scholarship in the digital age,” *portal: Libraries and the Academy*, vol. 3, no. 2, pp. 327–336, 2003.
- [21] M. Smith, M. Barton, Bass, *et al.*, “Dspace: An open source dynamic digital repository,” *D-Lib Magazine*, vol. 9, no. 1, 2003.
- [22] J. J. Carroll, C. Bizer, *et al.*, “Semantic web publishing using named graphs,” in *Proceedings of the 3rd International Semantic Web Conference Workshop on Trust, Security, and Reputation on the Semantic Web*, (Hiroshima, Japan), November 2004.
- [23] G. Lausen, “Relational databases in rdf: Keys and foreign keys,” in *Semantic Web, Ontologies and Databases, VLDB Workshop, SWDB-ODDIS 2007*, (Vienna, Austria), pp. 43–56, September 2007.
- [24] F. Baader and W. Nutt, *Basic Description Logics: The description logic handbook*. Cambridge Univ Press, 2003.
- [25] M. A. Stella Mitchell, Shanshan Chen *et al.*, “The vivo ontology: Enabling networking of scientists,” in *Proceedings of the 3rd International Conference on Web Science, (ACM WebSci’11)*, (Koblenz, Germany), pp. 1–2, June 2011.
- [26] S. Amrouch and S. Mostefai, “Ontology interoperability techniques, the state of the art,” *Journal of Information Organization*, vol. 2, no. 1, pp. 20–27, 2012.
- [27] R. Alhajj, “Extracting the extended entity-relationship model from a legacy relational database,” *Information Systems*, vol. 28, no. 6, pp. 597–618, 2003.
- [28] M. A. Maatuk, M. A. Ali, and B. N. Rossiter, “Semantic enrichment: The first

- phase of relational database migration,” in *Proceedings of 4th International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering, CISSE 2008*, (Bridgeport, USA), pp. 373–378, December 2008.
- [29] N. Gherabi, K. Addakiri, and M. Bahaj, “Mapping relational database into owl structure with data semantic preservation,” *International Journal of Computer Science and Information Security*, vol. 10, no. 1, pp. 42–47, 2012.
- [30] G. Nagy, S. C. Seth, *et al.*, “Data extraction from web tables: The devil is in the details,” in *Proceedings of 11th International Conference on Document Analysis and Recognition (ICDAR 2011)*, (Beijing, China), pp. 242–246, September 2011.
- [31] G. Kappel, E. Kapsammer, H. Kargl, *et al.*, “Lifting metamodels to ontologies: A step to the semantic integration of modeling languages,” in *Proceedings of 9th International Conference on Model Driven Engineering Languages and Systems, MoDELS 2006*, (Genova, Italy), pp. 528–542, October 2006.
- [32] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm, “Schema and ontology matching with coma++,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (Maryland, USA), pp. 906–908, June 2005.
- [33] D. Engmann and S. Maßmann, “Instance matching with coma++,” in *Proceedings of the workshop on Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, (Aachen, Germany), pp. 28–37, March 2007.
- [34] S. Araújo, J. Hidders, *et al.*, “Serimi - resource description similarity, rdf instance matching and interlinking,” in *Proceedings of the 6th International Workshop on Ontology Matching*, (Bonn, Germany), pp. 246–247, October 2011.
- [35] S. Castano, I. S. E. Peraldi, A. Ferrara, *et al.*, “Multimedia interpretation for dynamic ontology evolution,” *Journal of Logic and Computation*, vol. 19, no. 5, pp. 859–897, 2008.
- [36] S. Castano, A. Ferrara, S. Montanelli, and D. Lorusso, “Instance matching for ontology population,” in *Proceedings of the Sixteenth Italian Symposium on Advanced Database Systems, (SEBD 2008)*, (Mondello, PA, Italy), pp. 121–132, June 2008.
- [37] S. Castano, A. Ferrara, and S. Montanelli, “Matching ontologies in open net-

- worked systems: Techniques and applications,” *Journal on Data Semantics (JoDS)*, pp. 25–63, 2006.
- [38] C. A. Knoblock, P. A. Szekely, J. L. Ambite, *et al.*, “Semi-automatically mapping structured sources into the semantic web,” in *Proceedings of the 9th Extended Semantic Web Conference, ESWC 2012*, (Heraklion, Crete, Greece), pp. 375–390, May 2012.
- [39] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, (Williamstown, MA, USA), pp. 282–289, June 2001.
- [40] D. W. Embley, S. W. Liddle, *et al.*, “Kbb: A knowledge-bundle builder for research studies,” in *Proceedings of 2nd International Workshop on Active Conceptual Modeling of Learning (ACM-L 2009)*, (Gramado, Brazil), pp. 148–157, November 2009.
- [41] F. Cerbah, “Mining the content of relational databases to learn ontologies with deeper taxonomies,” in *Proceedings of 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops (WI-IAT 2008)*, (Sydney, NSW, Australia), pp. 553–557, December 2008.
- [42] N. Cullot, R. Ghawi, and K. Yétongnon, “Db2owl : A tool for automatic database-to-ontology mapping,” in *Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007*, (Torre Canne, Fasano, BR, Italy), pp. 491–494, June 2007.
- [43] K. M. Albarrak and E. H. Sibley, “Translating relational & object-relational database models into owl models,” in *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2009*, (Las Vegas, Nevada, USA), pp. 336–341, August 2009.
- [44] R. Ghawi and N. Cullot, “Database-to-ontology mapping generation for semantic interoperability,” in *3rd International Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007*, (Vienna, Austria), September 2007.
- [45] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan, “The use of information capacity in schema integration and translation,” in *Proceedings of the 19th*

- International Conference on Very Large Data Bases (VLDB)*, (Dublin, Ireland), pp. 120–133, August 1993.
- [46] H. Maryam, R. Samhaa, and R. Ahmed, “A survey of ontology learning approaches,” *International Journal of Computer Applications*, vol. 22, pp. 36–43, May 2011.
- [47] T. Zhang, D. Xu, and J. Chen, “Application-oriented purely semantic precision and recall for ontology mapping evaluation,” *Knowledge-based Systems*, vol. 21, pp. 794–799, December 2008.
- [48] K. Dellschaft and S. Staab, “On how to perform a gold standard based evaluation of ontology learning,” in *Proceedings of 5th International Semantic Web Conference, ISWC 2006*, (Athens, GA, USA), pp. 228–241, November 2006.
- [49] H. A. Santoso, S.-C. Haw, and Z. Abdul-Mehdi, “Ontology extraction from relational database: Concept hierarchy as background knowledge,” *Knowledge-Based Systems*, vol. 24, pp. 457–464, April 2011.
- [50] P. Cimiano, A. Hotho, and S. Staab, “Learning concept hierarchies from text corpora using formal concept analysis,” *Journal of Artificial Intelligence Research*, vol. 24, pp. 305–339, August 2005.