

# **SEMANTIC KNOWLEDGE EXTRACTION FROM RELATIONAL DATABASES**

by

Kgotatso Desmond Mogotlane

206030878

Dissertation submitted in fulfilment of the requirement for the degree of

Magister Technologiae: Information Technology

in the

Department of Information and Communications Technology, Faculty of  
Applied and Computer Sciences, Vaal University of Technology

**Supervisor:** Dr J.V. Fonou Dombeu

May, 2014

# DECLARATION

I hereby declare that this dissertation, which I submit for the qualification of

## **Magister Technologiae: Information Technology**

To the Vaal University of Technology, Department of Information and Communications Technology, Faculty of Applied and Computer Sciences, apart from the recognized assistance of my supervisor and provided citations, is my own work and has not previously been submitted to any other institution for any degree.

\_\_\_\_\_ on this \_\_\_\_\_ day of \_\_\_\_\_  
Candidate

\_\_\_\_\_ on this \_\_\_\_\_ day of \_\_\_\_\_  
Supervisor

## **DEDICATION**

I dedicate this work to the Almighty, my mother Rosinah, my father Modise, my two sisters Mpho and Tebogo, my grandmothers Tshaisa, Elizabeth and Norah, and lastly the whole Mogotlane family.

## ACKNOWLEDGEMENTS

I received great support and guidance during the course of the study. I would like to take this moment and express my sincere gratitude to the following individuals who played a vital role in the completion of this dissertation:

- My supervisor Dr J.V. Fonou-Dombeu for his insightful and patient guidance. It would have been completely impossible to complete this work without a supervisor like him. I am proud to say I had a great supervisor for this work.
- To my former colleagues who are also my friends, Bongani Miya and Donald Madibana who were there when this journey began to offer words of encouragement and support.
- My sister Tebogo Mogotlane, even though she was not familiar with the contents of the study, was always there to offer motivation and words of advice during tough times.
- Mogotlane family, for the unconditional love they have and continue to give me.

## **PUBLICATIONS**

The following publications have resulted from this work:

MOGOTLANE, K.D. & FONOU-DOMBEU, J.V. (2014) Development of a Data Model for Semantic Exploitation of Municipality Records in South Africa, In Proceedings of the Information Society Technology of Africa (IST-Africa 2014) Conference, Pointe aux Piments, Mauritius, 5-9 May, pp. 1-7.

MOGOTLANE, K.D. & FONOU-DOMBEU, J.V. (2014) Comparison of Protégé Plug-ins Performance in Automatic Construction of Ontology from Relational Databases, In Proceedings of the 8<sup>th</sup> International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 18-20 December.

MOGOTLANE, K.D. & FONOU-DOMBEU, J.V. Competency Validation of Ontology Constructed from Relational Database, Submitted for Review to 2015 Annual Conference of the South African Institute of Computer Scientists and Information Technologies (SAICSIT 2015).

## ABSTRACT

One of the main research topics in Semantic Web is the semantic extraction of knowledge stored in relational databases through ontologies. This is because ontologies are core components of the Semantic Web. Therefore, several tools, algorithms and frameworks are being developed to enable the automatic conversion of relational databases into ontologies. Ontologies produced with these tools, algorithms and frameworks need to be valid and competent for them to be useful in Semantic Web applications within the target knowledge domains. However, the main challenges are that many existing automatic ontology construction tools, algorithms, and frameworks fail to address the issue of ontology verification and ontology competency evaluation. This study investigates possible solutions to these challenges. The study began with a literature review in the semantic web field. The review led to the conceptualisation of a framework for semantic knowledge extraction to deal with the abovementioned challenges. The proposed framework had to be evaluated in a real life knowledge domain. Therefore, a knowledge domain was chosen as a case study. The data was collected and the business rules of the domain analysed to develop a relational data model. The data model was further implemented into a test relational database using Oracle RDBMS. Thereafter, Protégé plugins were applied to automatically construct ontologies from the relational database. The resulting ontologies are further validated to match their structures against existing conceptual database-to-ontology mapping principles. The matching results show the performance and accuracy of Protégé plugins in automatically converting relational databases into ontologies. Finally, the study evaluated the resulting ontologies against the requirements of the knowledge domain. The requirements of the domain are modelled with competency questions (CQs) and mapped to the ontology using SPARQL queries design, execution and analysis against users' views of CQs answers. Experiments show that, although users have different views of the answers to CQs, the execution of the SPARQL translations of CQs against the ontology does produce outputs instances that satisfy users' expectations. This indicates that Protégé plugins generated ontology from relational database embodies domain and semantic features to be useful in Semantic Web applications.

# TABLE OF CONTENTS

DECLARATION .....	II
DEDICATION .....	III
ACKNOWLEDGEMENTS .....	IV
PUBLICATIONS .....	V
ABSTRACT .....	VI
TABLE OF CONTENTS .....	VII
LIST OF FIGURES .....	IX
LIST OF TABLES .....	X
CHAPTER 1 .....	1
INTRODUCTION .....	1
1.1 BACKGROUND .....	1
1.2 PROBLEM STATEMENT .....	2
1.3 RESEARCH OBJECTIVES .....	3
1.4 METHODOLOGY .....	3
1.5 DISSERTATION OUTLINE .....	3
1.6 ORIGINAL CONTRIBUTIONS .....	4
CHAPTER 2 .....	5
LITERATURE REVIEW .....	5
2.1 INTRODUCTION .....	5
2.2 SEMANTIC WEB .....	5
2.3 ONTOLOGY .....	6
2.4 RELATIONAL DATABASE TO ONTOLOGY MAPPING .....	6
2.5 ONTOLOGY COMPETENCY EVALUATION .....	9
2.5 CONCLUSION .....	10
CHAPTER 3 .....	11
MATERIALS AND METHODS .....	11
3.1 INTRODUCTION .....	11
3.2 RELATIONAL DATABASE TO ONTOLOGY MAPPING .....	11
3.2.1 Relational Database .....	11
3.2.2 Ontology .....	12
3.2.3 Conceptual Relational Database to Ontology Mapping Rule/Principles .....	14
3.2.4 Relational Database to Ontology Conversion Tools and Algorithms .....	18
3.3 COMPETENCY QUESTIONS .....	19
3.4 TROPOS METHODOLOGY .....	20
3.5 COMPETENCY QUESTION TRANSLATION APPROACH .....	20
3.7 CONCLUSION .....	21
CHAPTER 4 .....	22
FRAMEWORK FOR SEMANTIC KNOWLEDGE EXTRACTION FROM RELATIONAL DATABASE .....	22

4.1 INTRODUCTION.....	22
4.2 FRAMEWORK OVERVIEW .....	22
4.3 RELATED WORK .....	25
4.4 CONCLUSION.....	27
CHAPTER 5 .....	28
KNOWLEDGE DOMAIN MODELLING.....	28
5.1 INTRODUCTION.....	28
5.2 PRESENTATION OF THE KNOWLEDGE DOMAIN .....	28
5.3 RELATIONAL DATA MODELLING.....	29
5.4 MODELLING OF COMPETENCY QUESTIONS.....	30
5.4.1 Early Requirements .....	31
5.4.2 Late Requirements .....	31
5.5 SPARQL REPRESENTATION OF COMPETENCY QUESTIONS .....	33
5.6 CONCLUSION.....	35
CHAPTER 6 .....	36
EXPERIMENTS AND DISCUSSIONS .....	36
6.1 INTRODUCTION.....	36
6.2 COMPUTER AND SOFTWARE ENVIRONMENT .....	36
6.3 EXPERIMENTAL RESULTS .....	36
6.4 CONCLUSION.....	49
CHAPTER 7 .....	51
CONCLUSION AND FUTURE WORK .....	51
7.1 SUMMARY OF THE STUDY .....	51
7.2 LIMITATIONS, RECOMMENDATIONS AND FUTURE WORK.....	52
7.3 CONCLUSION.....	53
BIBLIOGRAPHY .....	54
APPENDIX A.....	60
SOUTH AFRICAN MUNICIPALITIES COVERED DURING KNOWLEDGE DOMAIN MODELLING (CASE STUDY).....	60
APPENDIX B .....	62
RELATIONAL DATA MODELLING .....	62
COMPETENCY QUESTIONS AND THEIR SPARQL QUERIES CODE.....	63
SOME OF JAVA AND JENA API SCREENSHOTS .....	68



# LIST OF FIGURES

FIGURE 3.1: SAMPLE RELATIONAL DATABASE SCHEMA.....	14
FIGURE 4.2: KNOWLEDGE EXTRACTION FRAMEWORK.....	23
FIGURE 6.3: SCREENSHOT OF A TEST DATABASE IN ORACLE.....	37
FIGURE 6.4: INHERITANCE STRUCTURE OF ONTOLOGY CONSTRUCTED WITH DATA MASTER PLUG-IN VIA OWLVIZ.....	37
FIGURE 6.5: ONTOLOGY CONSTRUCTED WITH DATA MASTER PLUGIN.....	38
FIGURE 6.6: SCREENSHOT OF ONTOLOGY CONSTRUCTED WITH ONTOBASE PLUGIN .....	39
FIGURE 6.7: PART OF THE INHERITANCE STRUCTURE OF ONTOLOGY CONSTRUCTED WITH ONTOBASE PLUGIN VIA OWLVIZ .....	40
FIGURE 6.8: PART OF ONTOLOGY CONSTRUCTED WITH ONTOBASE .....	41
FIGURE 6.9: (A) DATABASE COMPONENTS (B) DATA MASTER ONTOLOGY COMPONENTS .....	42
FIGURE 6.10: (A) DATABASE COMPONENTS (B) ONTOBASE ONTOLOGY COMPONENTS .....	43
FIGURE 6.11: CHART OF COMPARISON OF PERFORMANCES OF DATA MASTER AND ONTOBASE PLUG-INS .....	44
FIGURE 6.12: PART OF OWL CODE OF THE ONTOLOGY THE SECOND RIGHT BLOCK OF FIGURE 6.9 DEPICTS.....	45
FIGURE 6.13: CQ1 SAMPLE (A) SPARQL QUERY AND (B) OUTPUTS.....	46
FIGURE 6.14: CQ9 SAMPLE (A) SPARQL QUERY AND (B) OUTPUTS.....	47
FIGURE 6.15: CHART OF THE MAPPING OF TERMS OF PARTICIPANT 1 TO SPARQL OUTPUTS INSTANCES .....	48
FIGURE 6.16: CHART OF THE MAPPING OF TERMS OF PARTICIPANT 2 TO SPARQL OUTPUTS INSTANCES .....	48
FIGURE 6.17: CHART OF THE MAPPING OF TERMS OF PARTICIPANT 3 TO SPARQL OUTPUTS INSTANCES .....	49
FIGURE B.1: INPUT RELATIONAL DATABASE SCHEMA.....	62
FIGURE B.2: CQ1 IN JENA.....	68
FIGURE B.3: CQ2 IN JENA.....	68
FIGURE B.4: CQ3 IN JENA.....	69
FIGURE B.5: CQ4 IN JENA.....	69
FIGURE B.6: CQ5 IN JENA.....	70
FIGURE B.7: CQ6 IN JENA.....	70
FIGURE B.8: CQ7 IN JENA.....	71
FIGURE B.9: CQ8 IN JENA.....	71

## LIST OF TABLES

TABLE 5.1 : SUMMARY OF SOUTH AFRICAN MUNICIPALITIES STUDIED .....	29
TABLE 5.2 : LIST OF COMPETENCY QUESTIONS .....	32
TABLE 5.3 : CATEGORISED COMPETENCY QUESTION.....	33
TABLE 5.4 : EXPECTED ANSWERS TO COMPETENCY QUESTIONS .....	34
TABLE 5.5 : ENTITIES EXTRACTED FROM ANSWERS AND COMPETENCY QUESTIONS .....	34
TABLE 5.6 : LIST OF ENTITY TYPES.....	34
TABLE 5.7 : SPARQL QUERIES .....	35
TABLE A.1 : LIST OF SOUTH AFRICAN MUNICIPALITIES .....	60

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

Semantic Web has been a topic of interest since its introduction in 2001 by Tim Berners-Lee, the founder of the World Wide Web (Zhang 2007; Abid & Javed 2013). From a few studies (Sadeh & Walker 2003; Zhang 2007; Khozoie 2012) it can be deduced that Semantic Web is a web of data from different sources that are linked together to create an integrated and structured global data space. This is quite different from the current Web that is composed of linked documents. The Semantic Web is not aimed to replace the current Web; it is an extension that extracts information and structures it in such a way that it can be understood by machines and human beings (Madhu, Govardhan & Rajinikanth 2011).

For Semantic Web to be fully realised, there had to be enabling technologies. Berners-Lee (2001) and the World Wide Web Consortium (W3C) ensured that there are standardised technologies on which semantic web can be built. The initial development was based on Extensible Markup Language (XML) and the Resource Description Framework (RDF) (Zhang 2007). Other technologies have since been added to the list of standardised technologies. Examples of such technologies are Resource Description Framework Schema (RDFS), Web Ontology Language (OWL) and SPARQL amongst others (Madhu et al. 2011; Khozoie 2012; Zemmouchi-Ghomari & Ghomari 2013a). RDF and OWL are Semantic Web languages utilised to represent ontologies (Madhu et al. 2011) and SPARQL is a formal query language used to query ontologies (Zemmouchi-Ghomari & Ghomari 2013a).

Ontology is an explicit specification of a conceptualisation which describes semantics of data, providing a shared and common understanding of a knowledge domain (Cristani & Cuel 2004). Ontology is the most important technology in Semantic Web; it is the backbone of any Semantic Web application (Gali, Chen, Claypool & Uceda-Sosa 2005; Imandi & Rizvi 2012; Madhu et al. 2012; Spanos, Stravrou & Mitrou 2012). On that note, ontology needs to be accurate and competent to create useful Semantic Web applications and be of a great use in the target knowledge domain. A competent ontology is the one that can satisfy the requirements of the knowledge domain meeting end users' expectations. One of the popular

ways to test the competency of any ontology is by using competency questions to ascertain whether the ontological commitments are adequate to support its use and purpose of design (Annamalai & Sanip 2010). Further, it is argued that, the increasing use and growing interest in Semantic Web technologies has escalated the need and demand for competent ontologies (Annamalai & Sanip 2010; Fernandes, Guizzardi & Guizzardi 2011).

Semantic Web technologies are being applied in various domains to build intelligent web applications based on ontologies. One of the main research topics in Semantic Web is the semantic extraction of knowledge stored in relational databases through ontologies. Tirmizi, Sequeda & Miranker (2008) and Sequeda, Marcelo & Miranker (2012) argued that relational databases are very important to the success of Semantic Web because most websites have relational databases as their source of information.

Different attempts have been made in Semantic Web research to propose methods that aim to present data stored in relational databases on the semantic web (Tirmizi et al. 2008; Sequeda et al. 2012; Spanos et al. 2012). In fact, many researchers and domain experts have developed tools, algorithms and frameworks for converting relational databases into well-structured ontologies so they can be presentable and queried on the semantic web (Nyulas, O'Connor & Tu 2007; Cerbah 2008; Zhou, Ling, Han, & Zhang 2010; Pasha & Sattar 2012; Jain & Singh 2013;).

## **1.2 Problem Statement**

Relational databases are the main sources of structured data for government institutions and businesses. This is a reason behind the widespread use of relational database management systems (RDBMS) like DB2, Oracle and Microsoft SQL Server (Lin 2008) and why more websites rely heavily on databases as the source of information (Tirmizi et al. 2008; Sequeda et al. 2012). As mentioned above, different tools, algorithms, and frameworks are being introduced (Cerbah 2008; Zhou et al. 2010; Pasha & Sattar 2012; Jain & Singh 2013) to make sure that data from relational databases can be exploited on the Semantic Web. However, many of these tools, algorithms, and frameworks fail to fully address the issue of ontology verification and ontology competency evaluation. To address these problems, there is a need to investigate solutions that will ascertain that ontology verification and ontology competency evaluation is addressed while practically applying existing Semantic Web technologies, tools

and algorithms to semantically extract knowledge from relational databases. This will ensure that the extracted knowledge is accurate and useful to the knowledge domain.

### **1.3 Research Objectives**

The aim of this study is to investigate solutions for automatically extracting and validating semantic knowledge from relational databases.

The main objectives are:

- 1) To investigate and identify the shortcomings of existing Semantic Web techniques used to semantically represent knowledge from relational databases.
- 2) To develop a framework for the semantic extraction of knowledge from relational databases which will address the identified challenges.
- 3) To conduct experiments to demonstrate the feasibility of the proposed framework.

### **1.4 Methodology**

This study was carried out using a combination of qualitative and quantitative research approaches. The study began with a thorough literature review of semantic web and ontology. The literature review was conducted to obtain information on the existing tools, algorithms and framework used to represent relational databases on the Semantic Web. The literature review then led to the conceptualisation and design of the proposed framework. The proposed framework had to be practically implemented to demonstrate its feasibility. Then a series of experiments were carried out in a chosen knowledge domain. In preparation of the experiments, the Tropos Methodology (Fernandes et al. 2011) and the Competency Questions Translation Approach (Zemmouchi-Ghomari & Ghomari 2013a) were used. The Tropos Methodology was used to get a set of competency questions from the business requirements of the knowledge domain, whereas, the Competency Questions Translation Approach was used to design SPARQL queries from the set of competency questions.

### **1.5 Dissertation Outline**

Chapter 2 covers the literature review of this study. Chapter 3 outlines the materials and methods applied in the study. In chapter 4, the proposed framework for semantic knowledge

extraction from relational database is presented. Chapter 5 presents knowledge domain modelling. In chapter 6, the framework is applied practically, tested and analysed in a series of experiments. Lastly, chapter 7 concludes the study, provides relevant recommendations and outlines future work.

## 1.6 Original Contributions

The original contributions made by this study are as follows:

1. In Chapter 4, a proposed framework for semantic knowledge extraction from relational databases is presented. The framework aims to serve as a practical guideline for the extraction, evaluation and validation of ontology from relational database.
2. In Chapter 5 Subsection 5.2, we present a relational data model for the South African municipality domain with the focus being on the information systems for service delivery. We conducted a case study of the knowledge domain and developed the model *de novo*. The test relational database used in the experiments was based on this model. This original work was published in Mogotlane & Fonou-Dombeu (2014a).
3. In Chapter 6 Subsection 6.3, we present ontology verification through conceptual mapping rules/principles to verify mapping accuracy of Protégé plugins for automatic relational database to ontology construction. This work was published in Mogotlane & Fonou-Dombeu (2014b).
4. In Chapter 6 Subsection 6.3, we apply an approach to evaluate the competency of ontologies that was derived from relational databases through the use of competency questions and users' views of their answers. This work was submitted for review to 2015 Annual Conference of the South African Institute of Computer Scientists and Information Technologies (SAICSIT 2015).

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

This chapter provides more details on semantic web and ontology. Thereafter, the background on relational database on the semantic web which covers existing RDB to ontology mapping principles, and RDB to ontology conversion tools, and algorithms is presented. Lastly, the chapter provides information on ontology competency evaluation using competency questions in ontology engineering.

#### 2.2 Semantic Web

Simply, semantics is associated with “meaning” and has been addressed in different ways by many research areas such as information retrieval, artificial intelligence, and database management (Sheth, Ramakrishnan & Thomas 2005). Semantics play a centre role in Semantic Web as an automated approach to exploit web resources (Sheth et al. 2005). The word “Semantic” in Semantic Web means that the meaning of web resources such as online data cannot only be discovered and interpreted by people, but by computers as well (Madhu et al. 2011). The idea of Semantic Web was conceptualised by Tim Berners-Lee in 2001 (Zhang 2007; Abid & Javed 2013). Berners-Lee et al. (2001) described Semantic Web as an extension of the current web (World Wide Web) where machines will have the capability to fully understand the data that they merely display on web pages. The vision here was to enable machines and humans to understand the content of the web, where intelligent software agents can be created to manipulate numerous web resources to structure meaningful information on behalf of the user (Berners-Lee et al. 2001). Semantic Web has since grown into a very active research field with many authors adding different definitions to support to the original concept.

In (Levshin 2010), Semantic Web is described as a concept that was mainly introduced with the aim of allowing computer agents to intelligently process information from different sources to make it more meaningful. For semantic web to be fully realised, web resources need to be annotated with machine understandable descriptions formally defined with ontology (Lu, Dong & Fotouhi, 2002). Abid & Javed (2013) tipped ontology to be “*one of the*

*pillars of semantic web*". Ontologies form a great part of Semantic Web because they represent formal semantics which are machine readable (Sheth et al. 2005). Ontology aims to express and enable meaningful associations between a set of concepts or entities in a particular knowledge domain (Zhang 2007).

## **2.3 Ontology**

Ontology is a concept from philosophy which deals with the nature of being. The term was adopted by the knowledge engineering, databases and software engineering communities and is given different definitions for different purposes (Corcho, Fernandez-Lopez & Gomez-Perez 2003). These communities use ontology in different areas such as natural language processing, intelligent information integration, the semantic web, etc. (Corcho et al. 2003). A popular definition of ontology is provided by Gruber (1993) who defined it as "*an explicit specification of a conceptualization*", where conceptualisation represents vocabulary of concepts, objects and their definitions in a certain knowledge domain of interest (Gruber 1993). In the context of semantic web, Imandi & Rizvi (2012) described ontology as a conceptualization in which knowledge is structurally represented to facilitate interoperability amongst different web resources.

## **2.4 Relational Database to Ontology Mapping**

Many studies have tackled the issue of converting relational databases into ontology. Sequeda et al. (2012) stated that automatic translation of relational databases to RDF is central to the full realisation of the Semantic Web. Their study presented a non-monotone direct mapping method that deals with the migration of relational databases to the Semantic Web. The non-monotone direct mapping method detects and eliminates any inconsistencies that may exist in the relational database before any migration of data into the RDF/OWL format. In Tirmizi et al. (2008) a method to convert relational data into OWL ontology is discussed. The resulting OWL ontology can be queried semantically and presented on the Semantic Web. This is done via a system that does the automatic transformation of SQL schemas into OWL DL (description logic) ontologies. However, Levshin (2010) argued that the existing methods that deal with the conversion of relational data to RDF/OWL ontologies ignore the constraints that are set on the foreign and primary keys. This may cause loss of meaning during the conversion. Therefore, a technique for mapping relational databases schemas to the Semantic Web without losing meaning is presented (Levshin 2010).



Spanos et al. (2012) took this further by bringing forward problems that relates to relational databases-to-ontology mapping. Their study outlined solutions to specific problems like the creation of ontology from an existing database instance and mapping an existing database instance with an existing ontology. On the problem of creating ontology from existing database instance, the study discussed a method that performs the mapping of relational database schemas into RDF Graphs; the resulting RDF Graphs may be stored in a file and accessed via SPARQL queries and Linked Data access modes (Spanos et al. 2012).

The research by Dou, Qin & Lependu (2010) presents an algorithm for integrating relational databases data into Semantic Web called OntoGrate. OntoGrate aims to integrate Semantic Web and relational databases in a highly automatic manner by combining an ontology mapping system, an inference engine and syntax wrappers. The OntoGrate has five components, namely, ontology matching, rule miner, inference engine, query interface, and syntax wrappers (Dou et al. 2010).

A method for automatic construction of ontology from relational databases, namely, Ontology Acquisition from Relational Database (OARDB) is presented in Meng, Ling & Zhou (2010). OARDB works with normalised relational databases, it utilises reverse engineering methods to extract relational database schema information. The method has the following four main steps: extraction of relational database schema information, analysis of the relational database schema information, retrieval of tuples from the relational database, and mapping of tuples to ontological instances. The study recommends the use of Java API in the first and second steps.

Astrova & Stantic (2004) presented a novel approach to reverse engineering of relational databases to ontologies; the approach extracts the semantics of the relational databases without explicitly analysing their underlying relational schema. According to the study this is done by analysing the HTML forms that acts as a front-end to the underlying relational databases. In this reverse engineering of relational databases to ontologies approach, HTML forms are viewed as an important tool for relational database data entry and display. Some of the reasons why HTML forms are analysed is because they are a popular interface to communicate with a relational database and they are well placed to assist users in understanding the semantics of data from the relational database. HTML forms can also assist to hide badly-designed and de-normalised relational databases during ontology construction. The approach consists of three main steps: extracting Model schema from HTML Forms,

schema transformation which entails application of mapping rules to construct an ontology, and Data migration which deals with the population of the ontology.

A method that enables Semantic Web Applications to query data from relational databases using locally constructed ontology is presented in Sedighi & Javidan (2012). The method is divided into two phases, namely, construction of the local ontology and query of the relational database via RDQL. Laclavik (2006) presented a unique relational database to ontology mapping approach called RDB2Onto. RDB2Onto is based on Java, Jena Framework/Sesame and MySQL Relational Database Management System (RDBMS). However, it can also be applied to other RDBMs using Java Database Connectivity (JDBC) connector. RDB2Onto converts selected relational database data to an RDF/OWL ontology using a SQL query with a defined template.

Jia & Yue (2009) proposed an ontology construction 3-Tuple model which utilised object-relational databases (ORDB) schema. The model consists of the following: ORDB which is the data source, a rule library which contains a set of ontology construction rules, and an ontology representation language (OWL). Jia & Yue (2009) argued that object-relational database offers an easier way to obtain the ontology concepts than constructing them from RDB directly.

An approach called MARSON (Mapping between relational schemas and ontologies) is introduced in Hu & Qu (2007). The approach discovers simple mappings between a relational database schema and an ontology. The mappings can be discovered by using a two-phased paradigm which entails the search of simple mappings between entities in the relational database schema and the ontology, and the development of complex composition based on simple mappings. MARSON was practically implemented using Java SE6. MARSON consists of the following phases: classification of entity types, discovery of simple mappings, validation of mapping consistency, and construction of contextual mappings.

Auer & Ives (2007) proposed a semantic web solution OWLDB developed with a platform called Powl. Powl a semantic web tool based on PHP. The OWLDB solution is an approach for integrating relational databases and description logic based ontologies. OWLDB is similar to OntoGrate from Dou et al. (2010) presented earlier. Like OntoGrate, OWLDB's aim is to integrate ontology and database under one query interface. The design of OWLDB includes:

a serialiser for exporting ontologies from various formats, a RDBMS (preferably MySQL), the subsumption reasoned and instance classifier, an API and SPARQL interface based on PHP, and the Powl's Web front-end. OWLDB is currently available as an Open-Source tool.

## **2.5 Ontology Competency Evaluation**

Several studies have discussed the use of competency questions to evaluate the competency of ontology (Lin & Sakamoto 2009; Annamalai & Sanip 2010; Fernandes et al. 2011; Fonou-Dombeu & Huisman 2011; Nemuraite & Paradauskas, 2012; Bezerra, Freitas & Santana 2013; Zemmouchi-Ghomari & Ghomari 2013b; Porwol, Ojo & Breslin 2014). In Annamalai & Sanip (2010), a tool to support the evaluation of ontology competency during its creation within Protégé is proposed. Competency questions are used to perform the evaluation; the authors advocated the use of formative evaluation which helps to ensure that the right ontology is built from the beginning through progressive validation of the conceptualisation. Another study by Lin & Sakamoto (2009) used competency questions to build ontology in the genetic disease domain.

Competency questions are used to design an e-Participation ontology in Porwol et al. (2014); they served to capture the requirements as well as to test the competency of the final ontology. In Fonou-Dombeu & Huisman (2011), the Uschold and King (1995) methodology was applied to build domain ontology. Competency questions were formulated and used to improve the corpus of concepts of the ontology. An interesting study in Zemmouchi-Ghomari & Ghomari (2013b) applied the NeOn methodology to build an ontology called HERO. The requirements of the domain were captured with eighty one (81) competency questions in the specification phase of HERO ontology development process. Furthermore, the CQT approach (Zemmouchi-Ghomari & Ghomari 2013a) was applied to evaluate the ontology.

In Nemuraite & Paradauskas (2012), competency questions are used to formulate and confirm ontology requirements in an ontology building methodology. Furthermore, the resulting competency questions are translated into SPARQL and tested using Protégé. Another study (Fernandes et al. 2011), discussed the use of competency questions in various ontology engineering methodologies including the Neon Methodology, Uschold and King and Method 101, and proposed the Tropos Methodology (goal modelling) which can be used by ontology engineers to capture and model competency questions. The Tropos Methodology

is an agent-oriented software engineering methodology which is founded on the concepts of actor and goal (Fernandes et al. 2011).

Competency questions are used to evaluate ontology in Bezerra et al. (2013). A tool called CQChecker is proposed to evaluate whether CQs are answered by the ontology. The algorithm of the CQChecker split a CQ into tokens. Thereafter, the tokens are used to retrieve classes and instances from the ontology. These classes and instances are considered as the answer to the CQ. Although the work by Bezerra et al. (2013) is closely related to this study regarding ontology evaluation, the proposed algorithm has not been empirically validated with a rigorously defined set of CQs as well as ontology of a domain of knowledge to justify its feasibility. In fact the proposed algorithm of Bezerra et al. (2013) has been tested with only a small and randomly selected set of CQs and it is unclear which ontologies were used. Furthermore, it was reported that 10 CQs were used to test the proposed algorithm; however, only one CQ is discussed in the study, making the evaluation of the algorithm too simplistic. Moreover, the study did not focus on the particular case of ontology automatically constructed from relational database (Bezerra et al. 2013).

Most of the studies discussed above used CQs while building ontology from scratch. In these studies, CQs are used to capture the requirements of the knowledge domain (e-Participation, Porwol et al. 2014), and government (Fonou-Dombeu & Huisman 2011), etc.) to build the ontology. The literature indicates that there are no studies that has focused on using CQs to evaluate ontology automatically constructed from a relational database as proposed in this study.

## **2.5 Conclusion**

This chapter provided a background of semantic web and ontology. The chapter also provided a discussion of different studies that were carried out to deal with the conversion of relational database into ontology. Finally, a background on the use of competency questions to evaluate the competency of the ontology was provided. The next chapter presents the materials and methods used in the study.

## **CHAPTER 3**

### **MATERIALS AND METHODS**

#### **3.1 Introduction**

This chapter discusses the materials and methods used in the study. They include: relational database, ontology, conceptual database to ontology mapping principles, competency questions, Tropos Methodology, Competency Question Translation approach and Iterative approach. The competency questions (CQs) of the domain are built using the Tropos Methodology, whereas, the Competency Question Translation (CQT) approach is applied to design the SPARQL queries representation of CQs.

#### **3.2 Relational Database to Ontology Mapping**

This section provides the definitions of relational database and ontology as well as the conceptual database-to-ontology mapping rules/principles, tools and algorithms.

##### **3.2.1 Relational Database**

A relational database is a data model which includes sets of relationships, attributes, and basic types (Zhang & Li 2011). A relational database could be represented in the form of a relational database schema (Navathe 1992). The relational database schema defines the structure of the database (Mahmood, Burney & Ahsan 2010) and consists of the following main elements (Li, Du & Wang 2005; Zhou et al. 2010; Telnarova 2010; Zhang & Li 2011; Saleh 2011):

- Relation - database table with a set of columns, rows and constraints.
- Attribute - column of a database table.
- Tuple - record or row of a database table.
- Domain - data type of a column of a database table. This is the type of values that can be present in a column e.g. Integer values etc.
- Primary Key - a constraint placed on a column to maintain entity integrity in the table. A primary key maintains unique rows in the table.

- Foreign Key - a constraint placed on a column to maintain referential integrity. A foreign key maintains relationships among database tables.

A relational database can have different types of relationships between its tables. The relationships are maintained by the use of foreign keys. Consider two related tables T1 and T2 with sets of rows R1 and R2, respectively. The possible relationships between the tables of the relational database are as follows:

- One to One relationship - one row  $r_{1i} \in R1$  ( $1 \leq i \leq n$ ) in T1 corresponds to only one row  $r_{2j} \in R2$  ( $1 \leq j \leq m$ ) in T2, where  $n$  and  $m$  are the numbers of rows in T1 and T2, respectively, i.e., only one row in T1 corresponds to only one row in T2.
- One to Many relationship - each row  $r_{1i} \in R1$  ( $1 \leq i \leq n$ ) in T1 corresponds to  $s_{2j} \in R2$  ( $1 \leq j \leq m$ ) in T2, where  $s_{2j} = \{r_{2k}, 1 \leq k \leq m\}$  is a set of rows in T2,  $n$  and  $m$ , the numbers of rows in T1 and T2, respectively. This means that one row in T1 can have many corresponding rows in T2. In this relationship, a primary key in T1 will be a foreign key in T2.
- Many to Many relationships - a set of rows  $s_{1i} \in R1$  ( $1 \leq i \leq n$ ) in T1 corresponds to a set of rows  $s_{2j} \in R2$  ( $1 \leq j \leq m$ ) in T2, where  $n$  and  $m$  are the number of rows in T1 and T2, respectively, i.e., many rows in T1 corresponds to many rows in T2. These relationships are normally resolved by a use of bridge tables.

### 3.2.2 Ontology

Ontology is a knowledge base system representing the common and shared vocabularies/concepts within a specific domain as well as the relationships between them (Li et al. 2005; Zhou et al. 2010; Telnarova 2010). Typical ontology elements are concepts, relationships/properties, axioms and instances (Zhang & Li 2011; Saleh 2011). A concept is the basic component of ontology. The relationships/properties between concepts define how concepts are semantically related to each other in the ontology. Axioms are the statements in the ontology, i.e., the logical combinations of concepts and properties. The instances are the occurrences/values of concepts or properties in the ontology. The popular languages for the formal representation of ontology are RDF and Web Ontology Language (OWL). However, OWL is preferred over RDF (Li et al. 2005; Jia & Yue 2009) due to the weak expressive

power of the RDF language (Li et al. 2005; Jia & Yue 2009). It is also considered to be the most advanced ontology representation language (Lemaignan, Dantan, & Semenenko 2006). The common keywords of the OWL language for representing ontology elements are defined below (Li et al. 2005; Lemaignan et al. 2006; Telnarova 2010; Zhang & Li 2011; Gherabi, Addakiri & Bahaj 2012; Sedighi & Javidan 2012):

1) *Class*: It represents a concept of ontology in OWL (Li et al. 2005; Jia & Yue 2009).

An example of OWL representation of a class named *PropertyType* is given in the line of code below.

```
<owl:Class rdf:ID = "#PropertyType" />
```

2) *Object Property*: This OWL construct defines relationships between ontology classes (Zhang & Li 2011). Object Properties are defined using domains and ranges which are the classes that are in relation with one another (Zhou et al. 2010). The following code presents an OWL Object Property named *PropertyTypeIDInstance*. The domain of the *PropertyTypeIDInstance* Object Property is the *PropertyService* class and its range the *PropertyType* class, i.e., *PropertyService* and *PropertyType* are in a relation with one another.

```
<owl:ObjectProperty rdf:ID="#PropertyTypeIDInstance"/>
<rdf:type rdf:resource="#functional property"/>
<rdfs:domain rdf:resource="#PropertyService"/>
<rdfs:range rdf:resource="#PropertyType"/>
</owl:ObjectProperty/>
```

3) *Datatype Property*: It represents the attributes of ontology classes in OWL (Gherabi et al. 2012). Datatype Properties are also defined using domains and ranges; here, the domain represents a class that the property belongs to and range represents the type and limit of data that the property can store (Zhou et al. 2010). An example of OWL Datatype Property named *Description* is given in the code below. The domain of the Datatype Property is the *PropertyType* class and the range is *String*. The range indicates that *Description* Datatype Property represents string values.

```
<owl:DatatypeProperty rdf:ID="#Description"/>
<rdfs:domain rdf:resource="#PropertyType"/>
<rdfs:range rdf:resource="XMLSchema#string"/>
</owl:DatatypeProperty/>
```

4) *Individual*: It is an instance of a class or property. An example of an Individual named PropertyTypeInstance is given in the OWL code below. This is an instance of the PropertyType class.

```
<owl:PropertyType rdf:ID="#PropertyTypeInstance"/>
<owl:PropertyTypeID rdf:datatype="&xsd:int">1</owl:PropertyTypeID/>
<owl:Description rdf:datatype="XMLSchema:string">Residential
</owl:Description/>
<owl:Ratable rdf:datatype="XMLSchema:string">Yes </owl:Ratable/>
```

Class, Object Property, and Datatype Property are the main OWL elements as they represent ontology concepts, relationships between the concepts and attributes of the concepts. Classes and properties are components upon which the ontology hierarchy is built (Li et al. 2005). In the OWL hierarchy, owl:Thing is the base class and any other class in the ontology inherits from it (Gherabi et al. 2012). The next Subsection presents existing mapping rules that govern the conversion of a relational database into OWL ontology.

### 3.2.3 Conceptual Relational Database to Ontology Mapping Rule/Principles

The process of converting a relational database into ontology follows certain mapping rules/principles (Li et al. 2005; Cullot et al. 2007; Telnarova 2010; Zhou et al. 2010; Zhang & Li 2011; Gherabi et al. 2012; Pasha & Sattar 2012; Sedighi & Javidan 2012). Mapping rules define how relational database components including Tables, Columns, Foreign Keys, etc., can be converted into ontology components such as Classes, Properties, Instances, etc. In this Subsection, existing mapping rules are discussed using a sample relational database schema in Figure 3.1. The mapping rules used to convert the database tables in Figure 3.1 into OWL ontology constructs are presented below.

Relations	Primary Key	Foreign Key
PropertyType (PropertyTypeID, Description)	PropertyTypeID	None
PropertyService (PropertyServiceID, ServiceID, PropertyTypeID)	PropertyServiceID	ServiceID refers to ServiceID in Service
		PropertyTypeID refers to PropertyTypeID in PropertyType
Service (ServiceID, Description, Type)	ServiceID	None
Customer (CustomerID, Name, ID-RegistrationNo, PhysicalAddress, PostalAddress, Telephone, Cellphone, Email)	CustomerID	None
Query (QueryID, CustomerID, Status, Type, DateEntered, DateClosed, Details, AttendedBy)	QueryID	CustomerID refers to CustomerID in Customer
		AttendedBy refers to EmployeeID in Employee

Figure 3.1: Sample Relational Database Schema



1) *Rule 1 - Mapping of Tables to OWL Classes:* Each table in the relational database is mapped into ontology OWL class with similar name except for bridging bridge tables that are used to resolve many-to-many relationships (Zhou et al. 2010; Zhang & Li 2011; Gherabi et al. 2012; Sedighi & Javidan 2012). On that note, only all the four tables in Figure 3.1 are mapped to OWL classes as in the sample code below.

```
<owl:Class rdf:ID = "#PropertyType" />
<owl:Class rdf:ID = "#Service" />
<owl:Class rdf:ID = "#Customer" />
<owl:Class rdf:ID = "#Query" />
<owl:Class rdf:ID = "PropertyService" />
```

The PropertyService table (Figure 3.1) would not be was also converted to an OWL class because it is simply it couldn't be recognised as a bridge table even though it is used to resolve a many-to-many relationship between PropertyType and Service database tables. This is because it has a separate PropertyServiceID Primary Key in addition to the two foreign keys (ServiceID and PropertyTypeID). Rule 2 underneath elaborates more on handling of bridge tables.

#### 2) *Rule 2 – Handling of Bridge Tables:*

Bridge tables are not mapped into separate OWL classes. This rule applies to properly constructed bridge tables which have foreign keys from the tables participating in a many-to-many relationship as its main primary keys. Even though there is no separate class, many-to-many relationships are still represented by Object Properties in the ontology (Cullot et al. 2007). More on Object Properties is covered in Rule 6 and 7 underneath.

#### 3) *Rule 3 – Mapping of Referential Integrity Relationships to Inheritance Hierarchy:*

OWL Classes are arranged in a hierarchy based on the relationships in the database. In a relationship between two tables, a table that has a foreign key will be mapped into a sub-class of the main class obtained from a table with a corresponding primary key. For example, from the classes created in Rule 1 above, Query will be a sub-class of Customer because of a relationship between Query and Customer tables. Query table has a CustomerID foreign key to symbolise its dependence on the Customer table. An example of OWL code is depicted below:

```
<owl:Class rdf:ID = "#Query">
  <rdfs:subClassOf rdf:resource="#Customer"
<owl:Class />
```

#### *4) Rule 4 - Mapping of Non-Referential Integrity Columns into Datatype Properties:*

All columns in the relational database are mapped into Datatype Properties, except all the foreign keys which maintain referential integrity in the database (Cullot et al. 2007; Zhang & Li 2011; Gherabi et al. 2012; Sedighi & Javidan 2012). For instance, the Query class obtained in Rule 1 will have QueryID, Status, Type, DateEntered, DateClosed, and Details as Datatype Properties. CustomerID and AttendedBy are excluded from Datatype Properties list. The basic OWL code of the Datatype Property named Details in the Query class is provided below.

```
<owl:DatatypeProperty rdf:ID = "#Details" />
</owl:DatatypeProperty/>
```

#### *5) Rule 5 - Representation of Datatype Property host class as Domain and Data Type as Range:*

A Datatype Property includes domain and range which represents the host class and the type of data that will be represented, respectively (Cullot et al. 2007; Zhang & Li 2011; Gherabi et al. 2012; Sedighi & Javidan 2012). The code below shows the Query class as the domain of the Details Datatype Property, whereas, its range is the string datatype, i.e., the Details Datatype Property will represent string values.

```
<owl:DatatypeProperty rdf:ID = "#Details" />
<rdfs:domain rdf:resource = "#Query" />
<rdfs:range rdf:resource = "XMLSchema#string" />
</owl:DatatypeProperty/>
```

#### *6) Rule 6 - Mapping of Relationships represented by referential integrity columns into Object Properties:*

All relationships that are expressed with foreign keys in a relational database are mapped into OWL Object Properties (Li et al. 2005; Zhou et al. 2010; Gherabi et al. 2012; Sedighi & Javidan 2012). Two Object Properties are created for one-to-many or a many-to-many relationship, one for the relationship and one for its inverse. For instance, the Query and Customer classes obtained in Rule 1 would produce two Object Properties which are

represented by a CustomerID Functional Property within the Query class and a CustomerID Inverse Functional Property within the Customer class. This is because the Query class was derived from a Query table with a foreign key that points to a primary key in the Customer table. An OWL code for the Object Properties between the Query and Customer classes is given below:

```
<owl:ObjectProperty rdf:ID = "#CustomerIDInstance"/>
<rdf:type rdf:resource = "#functional property" />
</owl:ObjectProperty />
<owl:ObjectProperty rdf:ID = "#CustomerIDInstance"/>
<rdf:type rdf:resource = "#inversefunctional property" />
</owl:ObjectProperty />
```

#### 7) Rule 7 – Representation of Object Property host classes as domain and range:

An Object Property includes domain and range which represent the two classes in relation with one another. The domain is a class with a functional property while a range is a class with an inverse functional property (Li et al. 2005; Zhou et al. 2010; Gherabi et al. 2012; Sedighi & Javidan 2012). From the code shown below, the domain of the Object Property *CustomerIDInstance* is the Query class, whereas, its range is the Customer class. This Object Property defines the semantic relationship between the Query and Customer classes in the ontology.

```
<owl:ObjectProperty rdf:ID = "#CustomerIDInstance"/>
<rdf:type rdf:resource = "#functional property" />
<rdfs:domain rdf:resource = "#Query" />
<rdfs:range rdf:resource = "#Customer" />
</owl:ObjectProperty />
```

#### 8) Rule 8 – Mapping of Tuples to Individuals:

All database table records are mapped to individuals in ontology (Li et al. 2005; Zhang & Li 2011; Gherabi et al. 2012; Sedighi & Javidan 2012). For instance, if the Service table from Figure 3.1 had two rows of data, those rows will be mapped to OWL individuals as in the code below:

```
<owl:Service rdf:ID="#ServiceInstance"/>
  <owl:ServiceID rdf:datatype="&xsd:int">1
</owl:ServiceID>
```

```

<owl: Description rdf: datatype="XMLSchema#string">Electricity
<owl: Description/>
<owl: Type rdf: datatype="XMLSchema#string">Consumable <owl: Type/>
<owl: Service rdf: ID="#ServiceInstance2"/>
    <owl: ServiceID rdf: datatype="xsd:int">2
</owl:ServiceID>
<owl: Description rdf: datatype="XMLSchema#string">Refuse Removal </owl:
Description>
<owl: Type rdf: datatype="XMLSchema#string">Basic
</owl: Type>

```

#### 9) Rule 9 – Mapping of Column Constraints into Property Cardinalities:

Database column constraints e.g. NULL and NOT NULL are mapped into Ontology Property Cardinalities (Li et al. 2005; Pasha & Sattar 2012). Cardinalities are there to further specify and place restrictions on ontology properties (Li et al. 2005). For example, if the Query table in Figure 1 has a QueryID column which is declared as NOT NULL and a Type column which is NULL, this will lead to the following cardinalities in the ontology:

```

<owl:Restriction>
    <owl:onProperty rdf:resource="#QueryID"/>
    <owl:minCardinality>1< owl:minCardinality/>
<owl:maxCardinality>0< owl:maxCardinality/>
<owl:Restriction/>
<owl:Restriction>
    <owl:onProperty rdf:resource="#Type"/>
    <owl:minCardinality>0< owl:minCardinality/>
<owl:maxCardinality>1< owl:maxCardinality/>
<owl:Restriction/>

```

### 3.2.4 Relational Database to Ontology Conversion Tools and Algorithms

Ontology construction from a relational database used to be a manual and tedious process which relied solely on ontology editors and human experts (Li et al. 2005). Over the years, many tools and algorithms that enabled the automatic conversion of a relational database into ontology have been proposed. Examples of such tools and algorithms include: DB2OWL, R2O, D2RQ, Data Semantic Preservation, DartGrid Semantic, Semantic Bridge, Automapper, XTR-RTO, RTAXON, Learning Ontology from Relational Databases, Ontology Generator (RDB2On), and RDBToOnto amongst others (Zhou et al. 2010; Pasha & Sattar 2012; Jain and Singh 2013). The World Wide Web Consortium (W3C) through their RDB2RDF Working Group is also developing a direct mapping standard that focuses on translating relational database into RDF (Resource Description Framework) ontology (Gherabi et al. 2012). The problem with many of the abovementioned tools is that they are

still at the prototype stage and are not yet available to the public. In fact, some of these tools are still under development and are not yet fully fledged products. Furthermore, these tools have not yet been applied on real world databases to ascertain their performance in the automatic conversion of relational databases into ontologies. Protégé is a widely used ontology editing platform which offers great extensibility and scalability (Alatrish 2012). Its extensibility is due to plugins developed by semantic web experts. A plugin is a separately developed software module that adds more functionality to existing software. Examples of Protégé plugins include OntoLT (Buitelaar, Olejnik, & Sintek 2004), SIM-DLA (Mulligann, Trame & Krzysztof 2011), DataMaster (Nyulas et al. 2007), DataGenie (Gennari, Nguyen, & Silberfein 2007), OntoBase (Yabloko 2009) and RONTO (Papapanagiotou, Katsioulis, Tsetsos, Anagnostopoulos & Hadjiefthymiades 2006). OntoLT enables the extraction of ontology from text within Protégé (Buitelaar et al. 2004). SIM-DLA is a Protégé plugin that enables the comparison of ontology concepts and their meanings through the measurement of semantic similarities (Mulligann et al. 2011). DataMaster, DataGenie, OntoBase and RONTO are Protégé plugins that deal with the conversion of relational databases into ontologies. However, the RONTO plugin is still under development and is not yet available for use in the Semantic Web community (Papapanagiotou et al. 2006). Further, due to technical challenges such as unresolved errors and bugs (Gennari et al. 2007); DataGenie functionalities were improved to create the DataMaster plugin (Nyulas et al. 2007). In light of the above, DataMaster and OntoBase are the only plugins for automatic conversion of relational databases into ontologies that are currently available for use in Protégé. Consequently, they are used in this study to convert relational database to ontology.

### 3.3 Competency Questions

Competency questions (CQs) are natural language questions that are used to determine the scope of the ontology; they can also help to extract the concepts, properties/relations and axioms of the ontology (Lin & Sakamoto 2009). According to Gangemi (2005), accurate and competent ontology should specify all the conceptualisations required to answer a created set of CQs. The formulation of a set of CQs is one of the preliminary exercises that are undertaken at the start of manual ontology development process (Fonou-Dombeu & Huisman 2011; Porwol, Ojo & Breslin 2014). This is a case for numerous ontology engineering methodologies whose main aim is to assist ontology engineers to create ontologies manually *de novo*. The examples of these ontology engineering methodologies are Neon Methodology, Uschold and King, and Method 101 (Fernandes et al. 2011). This study applied CQs to

evaluate ontology generated automatically with Protégé plug-ins from a relational database. The aim is to ascertain whether the resulting ontology can provide answers to CQs of the knowledge domain. To this end, the requirements of the domain are modelled with CQs and mapped to the ontology using SPARQL queries design, execution and analysis against users' views of CQs answers.

### **3.4 Tropos Methodology**

The Tropos Methodology (Fernandes et al. 2011) is made up of three steps, namely:

- Early Requirements,
- Late Requirements and
- Ontology Modelling.

During Early Requirements, organisational actors, goals and their dependencies are identified. Organisational actors are like role players in the target knowledge domain. After the actors are identified, their soft and hard organisational goals are identified and modelled together with resources and organisational plans. This is a way of getting full organisational objectives. Late Requirements focuses on the capturing and modelling of CQs from the information obtained in the Early Requirements phase. Lastly, in the Ontology Modelling phase, concepts and their relationships are extracted from the set of CQs to build the domain ontology. This phase is not applied in this study. Instead, CQs obtained in the Late Requirements phase are used to map an ontology automatically constructed from a relational database to the requirements of the knowledge domain with the aim of evaluating its purpose or intended use (Zemmouchi-Ghomari & Ghomari 2013b).

### **3.5 Competency Question Translation Approach**

Since the Ontology is a machine readable representation of knowledge, end-users should be able to query it using formal language such as SPARQL (Zemmouchi-Ghomari & Ghomari 2013a). The set of competency questions obtained in the Late Requirements phase of the Tropos Methodology (Chapter 3 Section 3.4) are converted into SPARQL queries using the CQT approach (Zemmouchi-Ghomari & Ghomari 2013a). The CQT approach assumes that the user has a working knowledge of ontology languages (RDF or OWL) plus query language (SPARQL) and full understanding of the input ontology and the knowledge domain. The approach starts with the classification of CQs into different categories according to expected answers' types. There are five types of questions including:

- Definition questions ("What is/are?" or "What does mean?" type of questions),

- Boolean questions (question with Yes/No answers),
- Factual questions (those that search precise information),
- List questions (those that query list of entities), and
- Complex Questions (“How” and “Why” type of questions).

After the questions are categorised, their expected answers are then determined. From the answers and questions, entities are extracted and their types (whether it is a class, data property, object property, annotation, axiom, or instance) are identified. With all these information, appropriate SPARQL queries are constructed.

### **3.7 Conclusion**

All the necessary methods and materials needed to semantically extract and validate knowledge from relational databases were described in this chapter. Relational database as an input component and the main source of knowledge was also covered. In the next chapter, all the presented methods and materials are put together to design a framework for semantic knowledge extraction from relational database.

## **CHAPTER 4**

### **FRAMEWORK FOR SEMANTIC KNOWLEDGE EXTRACTION FROM RELATIONAL DATABASE**

#### **4.1 Introduction**

This chapter presents a framework for semantic knowledge extraction from relational database. The framework shows how semantic technologies, tools and algorithms can be applied in a structured manner to semantically extract useful knowledge from relational databases. The framework aims to serve as a guideline starting with the conversion of relational database to ontology, to the extraction of knowledge from the created ontology using a query application, until the verification and validation of the ontology to ensure that the extracted knowledge is valid and useful. The chapter covers the overview of the proposed framework and ends with the related work in the field.

#### **4.2 Framework Overview**

The proposed framework for semantic knowledge extraction from relational database is depicted in Figure 4.1. It has six main components including:

- Relational database,
- Conversion of relational database to ontology,
- Ontology,
- Ontology verification,
- Query application and
- Ontology competency evaluation.

The relational database component of the framework in Figure 4.1 (a) acts as an input. The relational database is then converted to ontology through the next component, namely, the conversion of RDB to ontology (Figure 4.1 (b)).



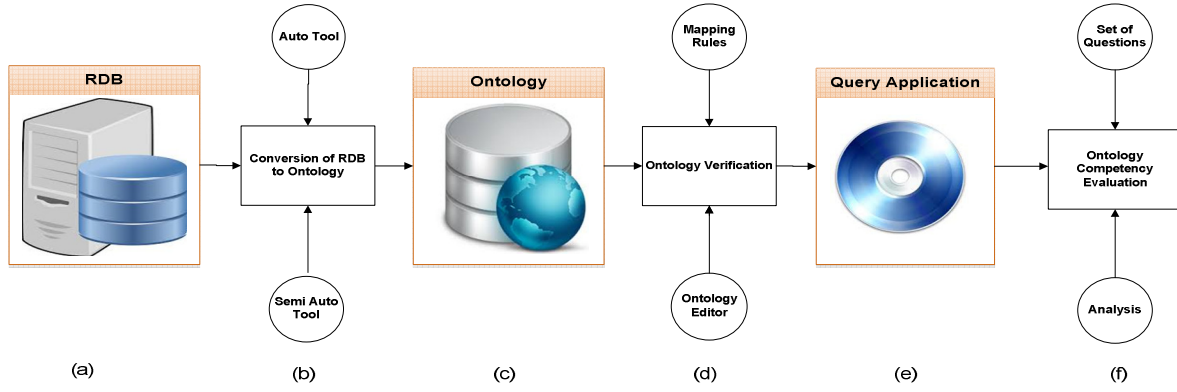


Figure 4.1: Knowledge Extraction Framework

The resulting ontology (Figure 4.1 (c)) is then verified through the ontology verification component in Figure 4.1 (d). The query application component (Figure 4.1 (e)) is then applied to extract knowledge from the ontology. Finally, the ontology is queried in the competency evaluation component in Figure 4.1 (f) to extract useful knowledge for the target domain. Each component of the framework in Figure 4.1 (a to f) is fully described in the subsequent subsections.

#### 4.2.1 Relational Database

The relational database component of the framework in Figure 4.1 (a) can be based on any Relational Database Management System (RDBMS) e.g. Oracle, MySQL, and DB2 amongst others. The study recommends Oracle and MySQL because they enjoy widespread use in many semantic web and ontology studies (Laclavik 2006; Auer & Ives 2007; Cullot et al. 2007; Zhou et al. 2010; Gherabi, Addakiri & Bahaj 2012).

#### 4.2.2 Conversion of Relational Database to Ontology

The Conversion of RDB to ontology component of the framework in Figure 4.1 (b) is one of the main components in the framework where the input relational database is converted to ontology. An automatic or semi-automatic tool can be applied to do the conversion. Automatic conversion of relational databases to ontology is the most preferred form of conversion because manual conversion is time consuming and less reliable (Jain & Singh 2013). The development of automatic or semi-automatic tools to convert relational databases to ontology is a very active research sub-field in the semantic web (Laclavik 2006; Cerbah 2008; Zhou et al. 2010; Pasha & Sattar 2012; Jain and Singh 2013). It was reported in Chapter 3 Section 3.2 that many existing conversion tools are still in the prototype state and are not yet available to the public. Due to this challenge this study applied a stable, extensible

and scalable platform like Protégé (Alatrish 2012) and its plug-ins, namely, DataMaster and OntoBase.

### **4.2.3 Ontology**

The Ontology component of the framework in Figure 4.1 (c) is an output of the Conversion of RDB to ontology component (Figure 4.1 (b)). The ontology should be presented in Web Ontology Language (OWL) or Resource Description Framework (RDF) format. As already mentioned in Chapter 1 Section 1.1, both RDF and OWL are official semantic web languages utilised to represent ontologies (Madhu et al. 2011). However OWL is more recommended over RDF because it produces more machine interpretability of web content and it provides more vocabulary along with formal semantics (Li et al. 2005).

### **4.2.4 Ontology Verification**

The Ontology verification component (Figure 4.1 (d)) validates the output ontology. This is an important quality check process in the proposed framework. Due to the fact that the input ontology was automatically derived from the structure of an existing relational database, it cannot be assumed that the output ontology will fully be valid without proper confirmation. This process validates the output ontology against existing conceptual database-to-ontology mapping principles presented in Chapter 3, Section 3.2. The goal is to ensure that relational database components like Tables, Columns, Primary Keys, Foreign Keys, Tuples, and Constraints etc., are properly converted into their corresponding ontology components such as Classes, Datatype Properties, Object Properties, and Individuals etc. After the ontology has been verified, an Ontology Editor like Protégé can be applied to rectify the ontology in case of gross deviations that might exist after the mapping process.

### **4.2.5 Query Application**

The Query application component of the framework in Figure 4.1 (e) is an interface that allows a user to extract knowledge from the verified ontology through SPARQL queries. SPARQL is a query language originally introduced by World Wide Web Consortium (W3C) to query ontology (Zemmouchi-Ghomari & Ghomari 2013). SPARQL is similar to SQL used to query relational databases.

### **4.2.6 Ontology Competency Evaluation**

The ontology competency evaluation component in Figure 4.1 (f) evaluates the extracted knowledge from the query application based on a set of competency questions to confirm that the output ontology does satisfy the requirements of the knowledge domain. This is to ensure that ontologies and knowledge extracted from them are relevant and useful within the target knowledge domain. This process is another important quality check process in the proposed framework. Ontology competency evaluation is commonly done through the use of competency questions (Lin & Sakamoto 2009; Annamalai & Sanip 2010; Fernandes et al. 2011); but, this study used competency questions to validate ontology automatically constructed from relational database.

### **4.3 Related Work**

The development of frameworks for semantically exploiting data from relational databases is an active research topic in the semantic web field (Li et al. 2005; Laclavik 2006; Saleh 2011; Zhang & Li 2011; Gherabi et al. 2012; Pasha & Sattar 2012; Sedighi & Javidan 2012; Jain & Singh 2013). Gherabi et al. (2012) presented architecture to map relational databases into OWL ontology. The authors developed a prototype to demonstrate the effectiveness of the architecture. This architecture is different from the proposed framework in Figure 4.1 in that it does not perform the competency validation of the resultant ontology. Furthermore, the mapping approach used in this particular study does not cover all the components of relational database and ontology. For instance, the study did not deal with the mapping of bridge tables in the relational database to ontology components.

Another framework for constructing ontology from relational database called OGSRD is presented in Zhang & Li (2011). The framework has three steps, namely, database metadata reading, ontology meta-model construction and goal ontology generation. The first step reads the database, the second step uses mapping rules to create the ontology meta-model and the last step generates ontology from the meta-model. This framework is also different from the proposed framework in Figure 4.1 as its main focus was only to obtain ontology from relational database. In fact the resulting OWL ontology was not further queried and analysed to evaluate its competency.

Sedighi & Javidan (2012) presented an architecture to semantically query data from relational databases using locally constructed ontology. The architecture is a two-phased approach with the first being the construction of a local ontology from a relational database and the second

phase being the querying of the resultant ontology via RDQL. In the first phase, the ontology is constructed in OWL. This is done by evaluating and matching the relational database components to their corresponding ontology components. The second phase utilises the resulting locally constructed ontology to perform semantic web queries in Jena API (Application Programming Interface) with the RDQL semantic query engine. This architecture's main differences to the proposed framework in Figure 4.1 are that the resulting ontology is not verified in any way to check its mapping accuracy and validity. Furthermore, the results from the query engine are also not evaluated to check the competency of the ontology.

Li et al. (2005) also presented a platform-free Ontology Learning Framework applied to develop ontology based applications or semantic web applications. The framework consist of a relational database as an input, a database analyser which extracts schema information from the database, an ontology generator to generate an ontology based on the schema information and rules, and the ontology editor and reasoner. The ontology learning approach in the framework of Li et al. (2005) can acquire ontology together with classes, properties, property characteristics, cardinality and instances using a set of learning rules. Although the framework was practically applied, the study (Li et al. 2005) does not mention how the resultant ontology will be verified to check its validity and queried to evaluate its competency.

The ontology learning framework by Li et al. (2005) was adopted and modified in Pasha & Sattar (2012) to create a new Framework. The framework proposed by Pasha & Sattar (2012) accepts multiple relational databases and relational schemas as input, a database analyser which extracts information from the schemas, an ontology generator to generate an ontology based on the schemas, and the ontology based application to query the ontology. Their study (Pasha & Sattar 2012) also does not indicate how the suggested ontology based application is going to be applied to query the resultant ontology. Further, the resulting ontology is also not verified to check its validity and its competency is also not evaluated.

Another framework to semantically query relational databases using ontology layer is presented in Saleh (2011). Saleh's (2011) framework is similar to that of Sedighi & Javidan (2012) two-phased architecture. The first phase of Saleh's (2011) framework is the offline ontology extraction and the second phase is the online query issuing. In the first phase, ontology is constructed by extracting classes and relations from the relational schema. Saleh's (2011) resultant ontology is verified by a domain expert using only four mapping

rules to check mapping accuracy and validity of the ontology. Mapping rules used in Saleh (2011) are too simplistic and does not cover all the components of the ontology. In the second phase, the user is allowed to run SPARQL queries against the constructed ontology. However the results from the query are not evaluated to check the competency of the ontology.

Jain and Singh (2013) presented another framework to convert relational database to ontology. The framework aimed at addressing problems and deficiencies in existing relational database to ontology conversion tools. Its main components included (1) a dynamic mapping mediator to convert data from multiple databases, (2) a module which facilitates support for different programming languages, (3) a module that enables the output of information in different formats, (4) a mediator class that deals with the translation of SPARQL queries, and (5) a visualisation service for non-programmers to run queries. However, Jain & Singh's (2013) study and the framework it proposed is too theoretical and does not convincingly indicate how the suggested modules are going to be implemented to assist in the process of relational database to ontology conversion. Further, the study (Jain & Singh 2013) does not mention any semantic web tools and platforms on which their framework is going to be based on. Similar to other studies discussed above, Jain & Singh's (2013) approach also focused mainly on the conversion of relational databases to ontology and did not address the issue of ontology verification and evaluation to make sure that the ontology produced by the framework is valid and is fully competent to be useful in a target knowledge domain.

#### **4.4 Conclusion**

This chapter presented the structure of our proposed framework for semantic knowledge extraction from relational database. The framework's main components are relational database, relational database to ontology mapping, ontology, ontology completeness verification, query application and ontology competency evaluation. The chapter also provided related literature that highlights the difference between this proposed framework and other existing frameworks in the semantic web literature. The next chapter presents and models the knowledge domain uses for data collection in this study.

## **CHAPTER 5**

### **KNOWLEDGE DOMAIN MODELLING**

#### **5.1 Introduction**

The task of semantic extraction of knowledge from relational databases has to be applied in a certain knowledge domain for it to be meaningful and useful. A knowledge domain can be anything from e-health, e-learning, e-commerce, e-government, business-to-business, etc. This chapter presents the knowledge domain used in this study in detail. Thereafter, the business rules of the domain are analysed to design a relational data model. Then, the Tropos Methodology (Chapter 3, Section 3.4) is applied to extract competency questions from the knowledge domain. The set of derived competency questions are then modelled into SPARQL queries by applying the Competency Questions Translation approach (Chapter 3, Section 3.5).

#### **5.2 Presentation of the Knowledge domain**

The knowledge domain in this study is the South African municipalities (SAM) information system for service delivery. A study was carried out to understand the SAM domain. The South African (SA) government, through its local and metropolitan municipalities has a constitutional obligation to provide basic services (e.g. potable water, sanitation, refuse removal, property assessments and electricity) to its citizens (RSA 1996; Emfuleni Local Municipality 2014a, 2014b). To achieve the constitutional obligation of effective service delivery, the country is divided into 234 local and metropolitan municipalities (Table 5.1) to ensure that all areas in the country are served (Koma 2010). Municipalities have tariff policies to govern the billing of major services and consumables such as electricity, water, sewerage, and refuse removal. They are also regulated by certain laws such as the Municipal Systems Act of 2000 (RSA 2000) to ensure that they remain constitutional when dealing with the public. The relationship between a municipality and the public can be compared to that of a service provider (municipality) and customer (public).

Table 5.1 shows a summary of municipalities that were studied in all the 9 SA provinces. A total of 9 municipalities were selected and studied per province. Overall 81 (35%) of both local and metropolitan municipalities were studied. The study consisted of a review and

analysis of municipalities' tariff and property rates policies downloaded from the municipalities' public websites.

Table 5.1 : Summary of South African Municipalities Studied

SA Provinces	No of Municipalities	No of Municipalities Covered
Gauteng	10	9 (90%)
Mpumalanga	18	9 (50%)
KwaZulu Natal	51	9 (18%)
Western Cape	25	9 (36%)
Free State	20	9 (45%)
North West	19	9 (47%)
Northern Cape	27	9 (33%)
Limpopo	25	9 (36%)
Eastern Cape	39	9 (23%)
Total	234	81 (35%)

This led to a thorough understanding of the knowledge domain as expressed partially in the following business rules. Municipalities maintain customer accounts to bill services they provide on a monthly basis. A customer can have an account with the municipality by virtue of being a property owner and occupier of a property that receives services. Monthly payments will be made to the account failing of which the account can go into arrears. Customers are allowed to make payment arrangements on accounts that are in arrears. A customer will also be able to lodge a complaint or put forward a general query in case of unhappiness with rendered services. The following services are offered to customers: Water, Electricity, Refuse removal, Basic sewerage and Property assessment. The services listed above are charged using a tariff that is influenced by many factors including: the category of the property, market value of the property determined after municipality property assessments, Consumption, and Peak and non-peak months (this specifically affects consumables like electricity). The municipality is responsible for maintaining a valuation roll that is used to capture all assessed and valued properties according to property category. Property categories are listed as: residential, sectional title, business, commercial, industrial, and farm dwellings.

### 5.3 Relational Data Modelling

An Iterative approach (Basili & Turner, 1975; Green & Ruhle, 2004) was taken during the design of the relational data model. The design had to go through a few iterations. The business rules and scenarios derived above were able to produce the following potential entities after the first iteration: Query, Administrator, Manager, Property Status, Group, Category, Query Type, Query Status, Property, Customer Group, Customer, Account Status,

Account, Account Billing, Property Service, Service, Arrangement, Arrangement Status, Arrears, Payment Method and Tariff. These entities were further searched in the data collected from the other municipalities and showed that these municipalities have compatible entities. Furthermore, it was found that the entities: Services, Property, Property Type/Category, Tariff, and Customer are common to all the municipalities. After data comparison and further analysis, the generic relational data model was drawn as in Figure 5.1.

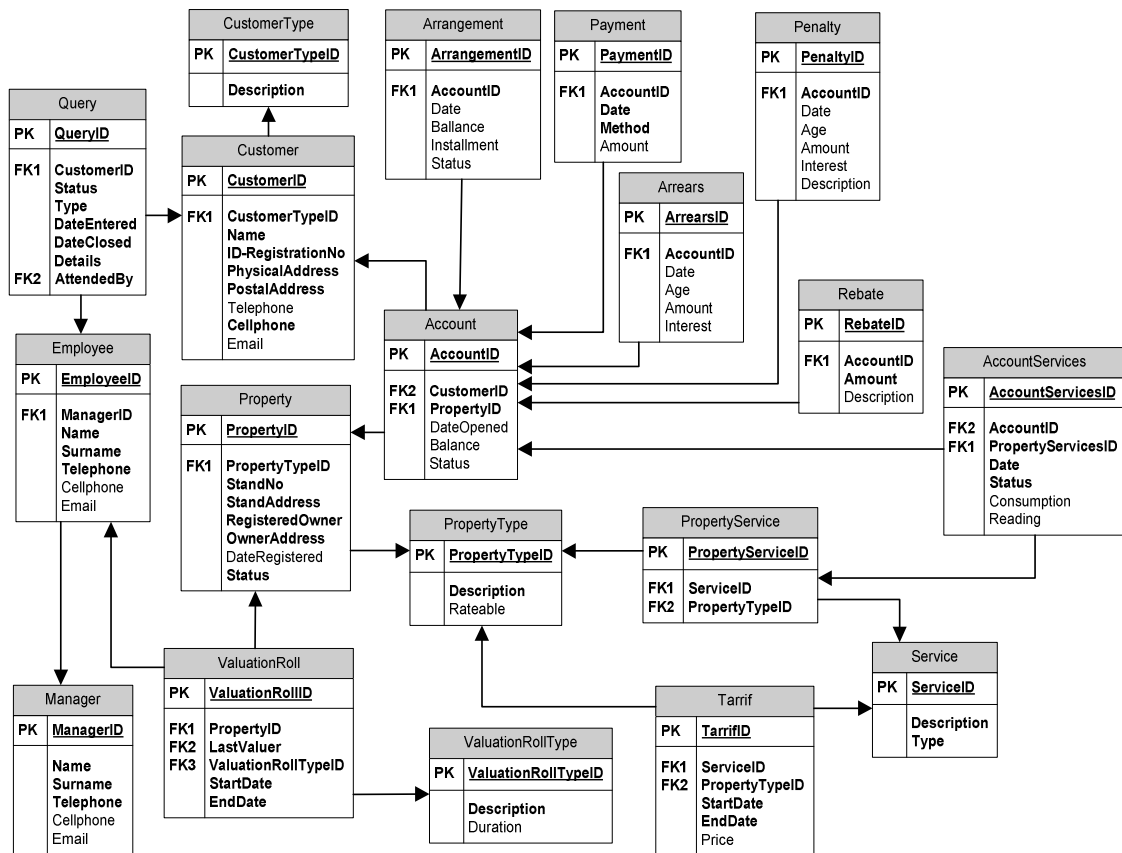


Figure 5.1: Diagram of the Relational Data Model for South African Municipalities

## 5.4 Modelling of Competency Questions

This section presents the application of the early and late requirements phases of the Tropos Methodology (Chapter 3, Section 3.4) to get a set of CQs from the knowledge domain. The knowledge domain in this study is the South African municipalities' information system for service delivery.



### 5.4.1 Early Requirements

From the knowledge domain (Section 5.2), municipality and customer are identified as the main role players/actors. The municipality has a soft goal to provide effective service delivery to customers. The customers are owners of properties that fall under the jurisdiction of the municipality and have a responsibility to log queries whenever they are not satisfied with services offered by the municipality. The soft goal above then leads to three main hard goals of:

- Identifying customers' properties as destinations of services to be rendered,
- Maintaining and managing a roll of all properties under the municipality, and
- Continuously improving services offered to customers.

The three hard goals above are further broken down into four sub-goals: offering services, improving services, capturing properties and managing properties. The resource needed to fulfil the goal of *improving services* is the queries submitted by customers. By addressing customer queries, the municipality will be in a better position to improve service delivery processes. On the other hand, a resource needed to fulfil the goals of *capturing properties* and *managing properties* is a municipal valuation roll. The valuation roll is a list of all properties under the municipality's jurisdiction.

The scenario above provides an overview of the early requirements where information is collected on how the municipalities fulfil their obligations of effective service delivery to customers.

### 5.4.2 Late Requirements

The organisational actors, goals and resources identified above are used in the late requirements phase of the Tropos Methodology (Chapter 3, Section 3.4) to capture and model the competency questions. The CQs obtained are provided in Table 5.2.

Table 5.2 : List of Competency Questions

CQ ID	Competency Question
CQ1	What are the services offered by the municipality?
CQ2	What are the types of services offered by the municipality?
CQ3	Which services are consumables in the municipality?
CQ4	Which services are basic in the municipality?
CQ5	How many customers do we have in our municipality?
CQ6	What are the names of our customers?
CQ7	What types of customers are catered for in our municipality?
CQ8	What are the overall queries in the municipality?
CQ9	What are the details and status of the current customer queries?
CQ10	What are the types of valuation rolls in the municipality?
CQ11	How much is the highly rated property within the municipality?
CQ12	What is the address of the most valued property in the municipality?
CQ13	How many properties do we have in the municipality?
CQ14	How many services are offered for residential properties?
CQ15	What are the ID's of customers who put in queries?
CQ16	What are the closed queries from the customers?
CQ17	What are the current open queries from the customers?

The CQs in Table 5.2 are encoded with identifiers (Fernandes et al. 2011). Seventeen CQs were derived in total with identifiers from CQ1 to CQ17 (Table 5.2). The competency questions CQ1 to CQ4 were derived from the *offering services* goal. In fact, to succeed in offering services to customers, the municipality would be interested in keeping record of service names (CQ1) and their types (CQ2). It will also be necessary for the municipality to specifically know which services are basic (CQ4) and which ones are consumables (CQ3). Consumables are services that are billed according to the customer's usage.

The competency questions CQ5 to CQ7 focus on the *customer* as the second organisational actor and the receiver of services. In this instance the municipality will be interested to know the number of customers, their names and types to gauge the demand for services. The competency questions CQ8 to CQ9 were derived from the *queries* resource. The municipality would need to know the overall queries and their details in order to achieve the *improve services* goal. The competency question CQ10 was derived mainly from the *valuation roll* resource. Here the municipality would need to establish the types of valuation rolls available to achieve the *capturing properties* and *managing properties* goals. The competency questions CQ11 to CQ13 were derived from the *capturing properties* and *managing properties* goals. To achieve these goals, the municipality would have to establish their mostly valued properties (CQ11) and their physical locations (CQ12). The municipality would also be interested to know the number of properties (CQ13) they have in their

jurisdiction. The competency questions CQ14 is derived from *offering services, capturing properties and managing properties* goals; in this case the municipality would be interested in identifying specific services that are offered to residential properties. Lastly, the competency questions CQ15 to CQ17 were derived from customers (organisational actor) and queries (resource). Here, the municipality would be interested in (1) identifying customers (CQ15) who put in queries and (2) the details of closed (CQ16) and open (CQ17) queries from customers. This will assist in the goal to *improving services*.

The next section focuses on the modelling of the CQs (Table 5.2) into SPARQL queries with the Competency Question Translation approach (Chapter 3 Section 3.5).

## 5.5 SPARQL Representation of Competency Questions

Competency questions in Table 5.2 are in the Natural Language (English) format. They need to be translated into a formal query language in order to be executed against the ontology. In this study, the CQT approach (Chapter 3 Section 3.5) is applied to get a set of SPARQL queries to be executed against an ontology automatically constructed from a relational database.

The CQT approach prescribes the classification of CQs into five categories, namely:

- List questions,
- Factual questions,
- Complex questions,
- Boolean questions, and
- Definition questions.

The classification of the competency questions in Table 5.2 is provided in Table 5.3.

Table 5.3 : Categorised Competency Question

CQ Category	CQ ID
List question	CQ1, CQ2, CQ6, CQ8, CQ10
Factual question	CQ3, CQ4, CQ7, CQ9, CQ15, CQ16, CQ17
Complex question	CQ5, CQ11, CQ13, CQ14
Definition question	CQ12
Boolean question	None

The Boolean category in Table 5.3 has no competency question as there was no question that could be simply answered by yes/no in the list of questions in Table 5.2. The competency

questions CQ1, CQ9, CQ11 and CQ12 are used in the remaining discussion of the CQT approach in this study. Expected answers to these questions are listed in Table 5.4.

Table 5.4 : Expected Answers to Competency Questions

CQ ID	Expected Answer
CQ1	Water, Sewerage, Refuse Removal, Basic Sewerage, Basic Electricity and Electricity
CQ9	Any municipality will log specific details and status of their customer queries e.g. <i>“No electricity for 5 days” : Query ‘Open’</i>
CQ11	The highly rated property in the municipality has a value of e.g. <i>“R1 Million”</i>
CQ12	The municipality’s highly valued property is found at following address e.g. <i>“80 Old Pretoria Road, Midrand”</i>

In Table 5.4, the names of services offered by the municipality are the expected answer to CQ1. The specification of customer query details and status constitute the answer to CQ9. For CQ11 and CQ12, expected answers are the value and address of the highly rated property respectively. The CQs together with their expected answers are used to get all relevant terms/entities which are concepts of the knowledge domain. Table 5.5 shows the terms that were manually extracted from the competency questions CQ1, CQ9, CQ11 and CQ12 and their answers in Table 5.4.

Table 5.5 : Entities Extracted from Answers and Competency Questions

CQ ID	Relevant Terms	Answer Relevant Terms
CQ1	Services	Names, Water, Sewerage, Refuse Removal, Basic Sewerage, Basic Electricity and Electricity
CQ9	Details, Status, Customer, Queries	Log Details and Status, Customer, Query
CQ11	Highly rated, Property	Value of Property
CQ12	Address, Highly Valued Property	Highly Valued, Address of Property

The entities in Table 5.5 were further modelled and categorised into entity types such as class, data property, object property, instance, etc. (Zemmouchi-Ghomari & Ghomari 2013a) as in Table 5.6.

Table 5.6 : List of Entity Types

CQ ID	Entity Types
CQ1	Class: Service Datatype Property : Name
CQ9	Class: Customer, Class: Query Datatype Property : Details, Status
CQ11	Class: Property Datatype Property: Value
CQ12	Class: Property Datatype Property : Address, Value

Finally, the CQs (Table 5.2), the answers to CQs (Table 5.4), the entities (Table 5.5) and the entity types (Table 5.6) are used to build SPARQL queries equivalent of CQs. The SPARQL queries for CQ1, CQ9, CQ11 and CQ12 are provided in Table 5.7. In Table 5.7, the acronym SAM stands for South African Municipality.

Table 5.7 : SPARQL Queries

CQ ID	SPARQL Queries
CQ1	SELECT ? Name WHERE { ?service a SAM:Name. ?service SAM:Service.Name ?Name. }
CQ9	SELECT * WHERE { ?query a SAM:Query. ?query SAM:Query.Details ?details. ?query SAM:Query.Status ?status. }
CQ11	SELECT (MAX (?value) AS ?value) WHERE { ?prop a SAM:Property. ?prop SAM:Property.Value ?value }
CQ12	SELECT ?address WHERE { ?prop a SAM:Property. ?prop SAM:Property.Address ?address. ?prop SAM:Property.Value ?value FILTER (?value = Max(value)) }

## 5.6 Conclusion

This chapter presented the study's chosen knowledge domain which is the South African municipalities (SAM) information system for service delivery. The business rules of the domain were studied to develop a relational data model. This relational data model was used to develop a test database in the next chapter. The chapter ended by presenting the domain competency questions and their corresponding SPARQL translations. The competency questions and SPARQL queries are used to evaluate the competency of the automatically constructed ontology in the next chapter. In the next chapter, the Knowledge extraction framework is practically applied, tested and analysed in a series of experiments to demonstrate its feasibility.

## CHAPTER 6

### EXPERIMENTS AND DISCUSSIONS

#### 6.1 Introduction

In this chapter, the experiments conducted in the study are presented. The experiments prove the feasibility of the proposed framework in Chapter 4. Firstly the computer and software environment used are presented. Thereafter, the experimental results are presented in sequence starting with the creation of the relational database to the ontology competency evaluation using competency questions.

#### 6.2 Computer and Software Environment

Experiments were carried out on a Dual Core 32 bit Notebook with 2 GB of RAM and a Windows 7 Operating System. Oracle 11g Express Edition was used as RDBMS. Two plug-ins, namely, DataMaster (Nyulas et al. 2007) and OntoBase (Yabloko 2009) were used to automatically construct ontologies from the Oracle database in Protégé version 4.3. Both plug-ins utilize the Oracle JDBC driver to establish a connection to the Oracle database. The graphical representation of the output ontologies from DataMaster and OntoBase was done using virtualisation plug-ins including OntoGraf (Falconer 2010) and OWLViz (Horridge 2010). A Semantic Web tool that generates a structured documentation of ontology, namely, Parrot (Foundation CTIC 2014) was used to display and analyse the structure of the output ontologies codes from DataMaster and OntoBase. The execution of the SPARQL queries against the OWL codes of the ontologies obtained from Protégé is done using the Jena API Semantic Web library under the Eclipse IDE.

#### 6.3 Experimental Results

The relational data model in Figure 5.1 in Chapter 5 presents the resulting municipality data model of this study. The data model was further implemented into a test database using Oracle 11g Express Edition. Figure 6.1 shows a screenshot of a database in Oracle. The right pane of the screenshot shows the sample data within the database.

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane displays the 'Municipality' database with a list of tables including ACCOUNT, ACCOUNTSERVICE, ARRANGEMENT, ARREARS, CUSTOMER, EMPLOYEE, MANAGER, PAYMENT, PROPERTY, PROPERTYSERVICE, PROPERTYTYPE, QUERY, SERVICE, SYSDIAGRAMS, and TARRIF. The 'SERVICE' table is selected. On the right, the 'Data' tab shows the following data:

ID	DESCRIPTION	TYPE
1	Water Consumption	Consumption
2	Electricity Consumption	Consumption
3	Basic Electricity	Basic
4	Refuse Removal	Basic
5	Basic Sewerage	Basic
6	Sewerage Additional	Basic

Figure 6.1: Screenshot of a Test Database in Oracle

The left pane of Figure 6.1 shows the Municipality database and some of its tables. The Service table was selected to show the sample data on the right pane under the data tab; it has six rows which show the list of municipality services.

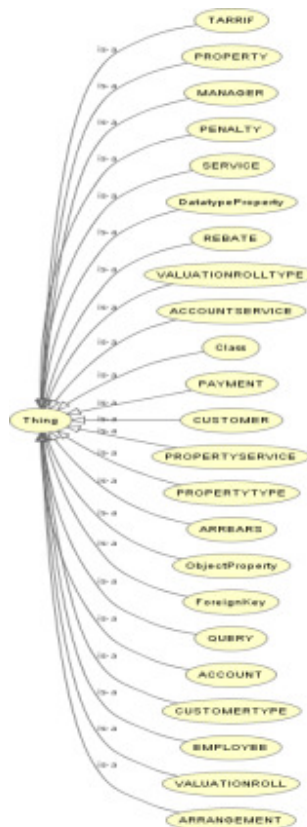


Figure 6.2: Inheritance Structure of Ontology Constructed with DataMaster Plug-in via OWLViz

The relational database in Figure 6.1 was converted to ontology to allow the semantic exploitation of its records. To this end, the DataMaster (Nyulas et al. 2007) and OntoBase (Yabloko 2009) Protégé plug-ins were used to automatically construct ontologies from the Oracle database in Figure 6.1.

Figure 6.2 shows the classes of the OWL ontology constructed from the Oracle database (Figure 6.1) with the DataMaster plug-in. The graphical representation of classes in Figure 6.2 was obtained with the OWLViz virtualisation plugin. The meaning of the graph in Figure 6.2 is that all classes produced, inherit the default OWL class called Thing.

The complete graph of the resulting ontology is shown in Figure 6.3; this graph was generated with the OntoGraf (Falconer 2010) virtualisation plug-in. Figure 6.3 shows all the classes of the ontology constructed with the DataMaster plug-in and the relationships between them.

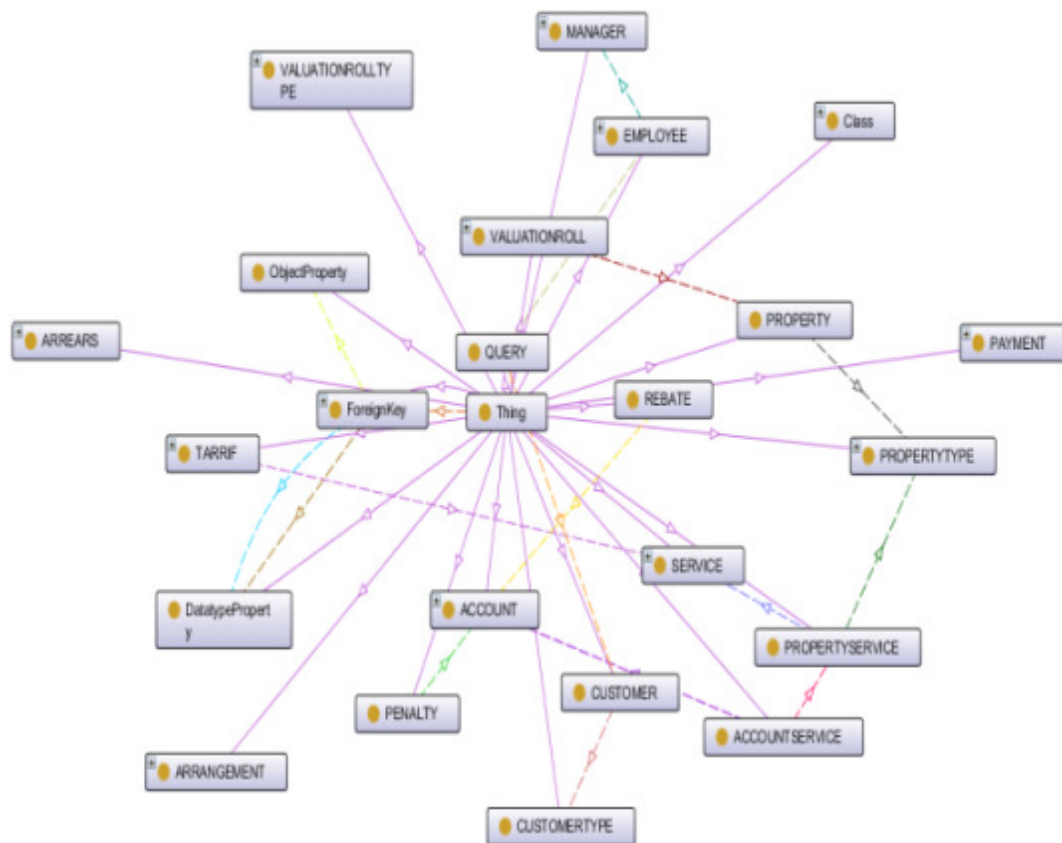


Figure 6.3: Ontology Constructed with DataMaster Plugin



Similarly, OntoBase plugin was used to construct OWL ontology from the Oracle database (Figure 6.1). Figure 6.4 shows a screenshot of the ontology constructed with OntoBase within Protégé. The structure of the ontology (Figure 6.4) is as follows: (1) a top level class, namely, Type, (2) the top level class Type is further divided into four subclasses, namely, Action, Tuple, Stream, Service and Ground; only the Action and Tuple subclasses are displayed in Figure 6.4, (3) the Tuple subclass in turn has a subclass dbo\_Municipality which contains all the classes that reflects the tables from the Oracle database.

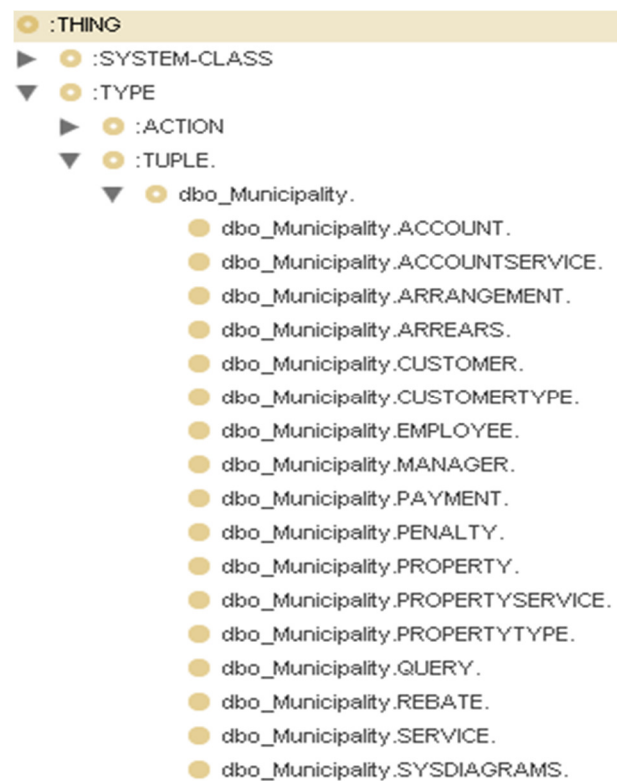


Figure 6.4: Screenshot of Ontology Constructed with OntoBase Plugin

The ontology in Figure 6.4 was further represented graphically in Figure 6.5 with the OWLViz virtualisation plugin. Figure 6.5 shows the inheritance structure of the ontology in Figure 6.4. A complete structure of the ontology in Figure 6.4 obtained with the OntoGraf visualization plugin is depicted in Figure 6.6. In Figure 6.6, all classes of the ontology constructed with OntoBase and the relationships between them are shown.



Figure 6.5: Part of the Inheritance Structure of Ontology Constructed with OntoBase Plugin via OWLViz

As mentioned earlier, the OWL codes of the ontologies constructed with both DataMaster and OntoBase plugins were further analysed using the Parrot (Foundation CTIC 2014) ontology documentation software. Parrot displayed the structure of the resulting OWL ontologies as well as useful comments that explained the OWL constructs (Classes, Datatype Properties, Object Properties, etc.) within the ontologies.

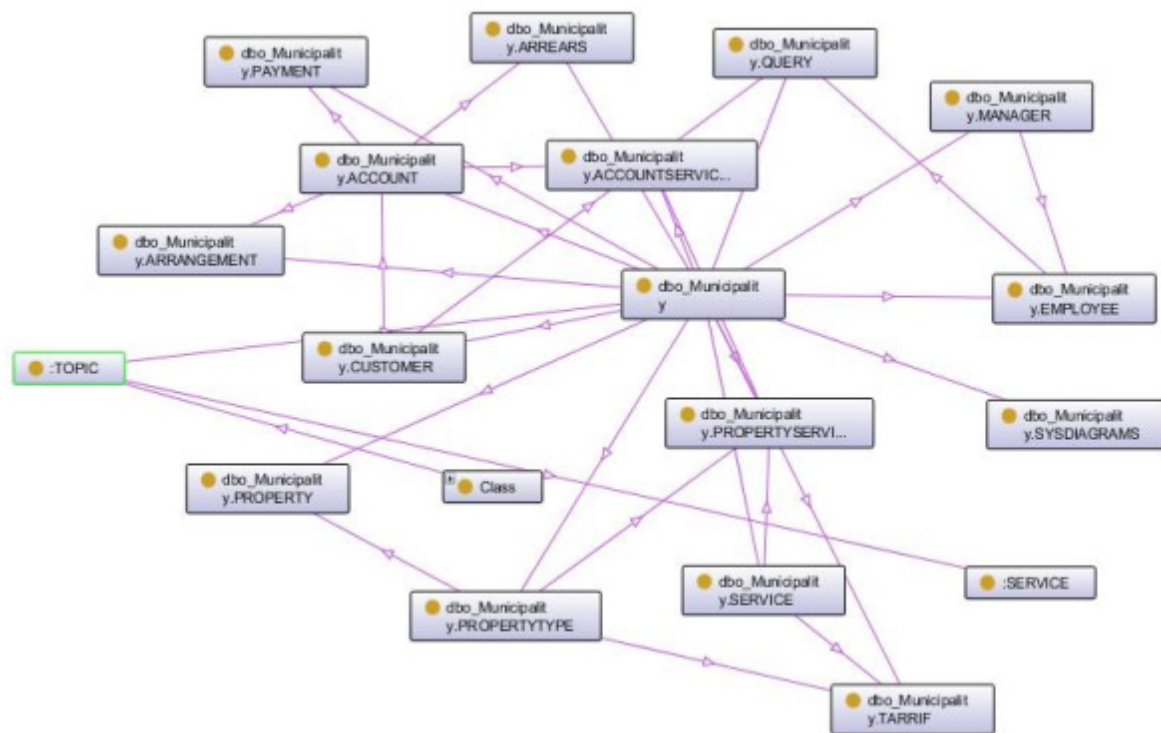


Figure 6.6: Part of Ontology Constructed with OntoBase

Both ontologies constructed with DataMaster and OntoBase (Figure 6.3 and Figure 6.6) were further analysed by applying conceptual relational database to ontology mapping rules/principles presented in Chapter 3 Subsection 3.2.3. Figure 6.7 shows the mapping results of the Oracle database (Figure 6.1) into ontology (Figure 6.2 and Figure 6.3) with the DataMaster plugin. The results in Figure 6.7 (a and b) shows that all tables were successfully mapped to ontology classes (Figure 6.2 and Figure 6.3) including the PropertyService bridge table. This is a deviation from the rules stated in Chapter 3 Subsection 3.2.3 because bridge tables are not supposed to be mapped into classes. In addition to classes mapped from the relational database tables, four other classes were mapped. These extra classes are annotation components added by the DataMaster plugin during the mapping process.

With regards to Datatype Properties, the mapping rules in Chapter 3 Subsection 3.2.3 stated that all columns of the database should be mapped to Datatype Properties except foreign keys; however results in Figure 6.7 (a, b) shows that all 105 columns were mapped into Datatype Properties irrespective of whether they are foreign keys or not. DataMaster also added 7 extra columns for annotation purpose. This finding reveals a slight deviation from the mapping principles in Chapter 3 Subsection 3.2.3. The mapping rules in Chapter 3 Subsection 3.2.3 further stated that relationships represented by foreign keys should be mapped to Object

Properties, with a creation of two object properties for each one-to-many and many-to-many relationship.

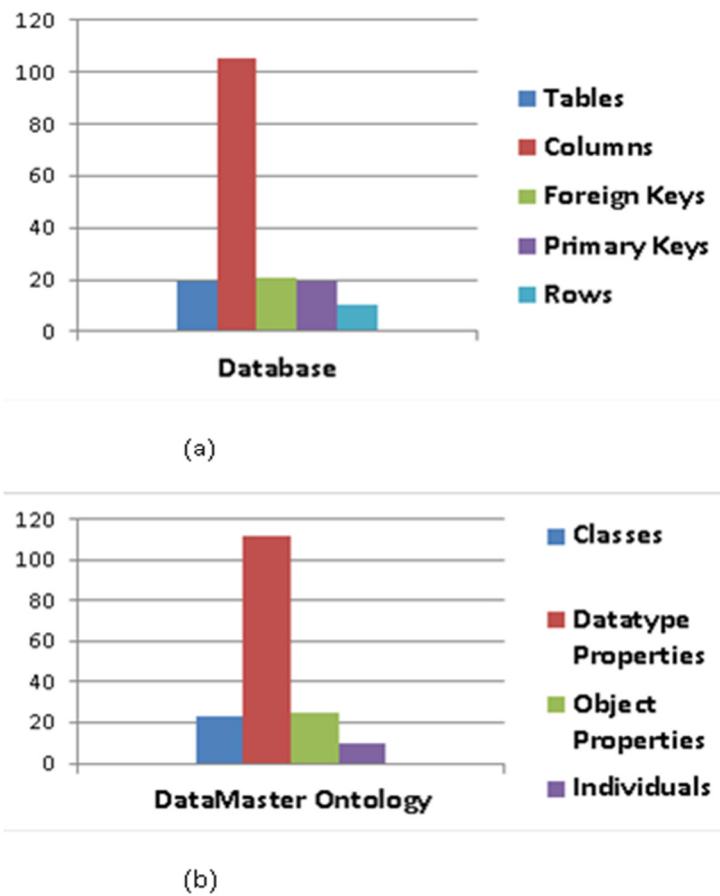


Figure 6.7: (a) Database components (b) DataMaster Ontology Components

The results in Figure 6.7 (b) shows that all 21 foreign keys were successfully mapped to Object Properties with an addition of 4 more Object Properties for annotation purposes as well. A deviation here is that duplicate object properties were not created to represent inverse functional properties as stated in the rules (Chapter 3 Subsection 3.2.3). Lastly, it was stated in Chapter 3 Subsection 3.2.3 that all database table records are mapped to individuals in ontology. The results in Figure 6.7(b) shows that all 10 test rows were successfully mapped to Individuals. Overall, the results in Figure 6.7 show that, according to database to ontology mapping principles, the ontology constructed with the DataMaster plugin has captured most of the features of the input database even though there were slight deviations from the database-to-ontology mapping principles (Chapter 3 Subsection 3.2.3).

Similarly, Figure 6.8 shows the mapping results of the Oracle database (Figure 6.1) into ontology (Figure 6.4, 6.5 and 6.6) with the OntoBase plugin. The structure of the resulting ontology is presented in Figure 6.8 (b) with 20 classes (leftmost bar), 62 Datatype Properties (second bar from the left to the right), 68 Object Properties (third bar from the left to the right) and 10 Individuals (rightmost bar).

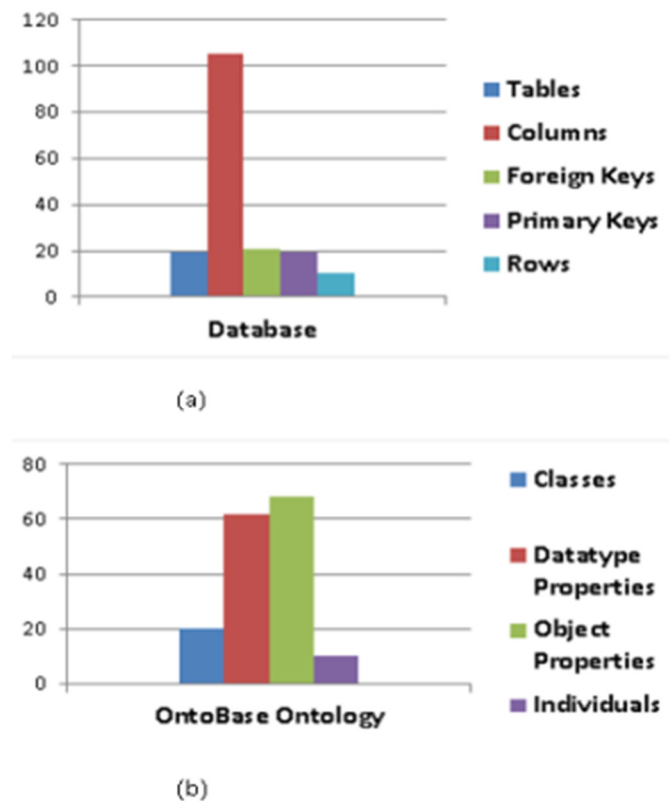


Figure 6.8: (a) Database components (b) OntoBase Ontology Components

The mapping rules in Chapter 3 Subsection 3.2.3 prescribed that all relational database tables should be mapped to ontology classes with similar names except for bridging tables that are used to resolve many-to-many relationships. Figure 6.8 (a and b) shows that all tables in the database were successfully mapped to ontology classes with an addition of 1 class. The PropertyService bridge table was also converted into a standalone class. This is a deviation from the rules stated in Chapter 3 Subsection 3.2.3 because bridge tables are not supposed to be mapped into classes. The mapping rules in Chapter 3 Subsection 3.2.3 also prescribes that all columns of the database be mapped to Datatype Properties except foreign keys. Results in Figure 6.8 (a and b) shows that all columns of the database were mapped into Datatype Properties except for foreign keys. In fact, the OntoBase plugin produced fewer Datatype

Properties; this proves that foreign key columns were indeed excluded. This is a major conformance with the mapping principles in Chapter 3 Subsection 3.2.3.

With regard to Object Properties, the mapping rules in Chapter 3 Subsection 3.2.3 prescribed that relationships represented by foreign keys are to be mapped to Object Properties, with a creation of two object properties for each one-to-many and many-to-many relationship. The results in Figure 6.8 (b) show that all foreign keys were successfully mapped to Object Properties with all the necessary duplicates due to one-to-many and many-to-many relationships. Lastly, it was stated in Chapter 3 Subsection 3.2.3 that all database table records are mapped to individuals in ontology. The results in Figure 6.8(b) shows that all 10 test rows were successfully mapped to Individuals. Overall the results in Figure 6.8 reveal that the structure of the ontology obtained with the OntoBase plugin has few deviations and does capture accurately the features of the input relational database according to the mapping principle in Chapter 3 Subsection 3.2.3.

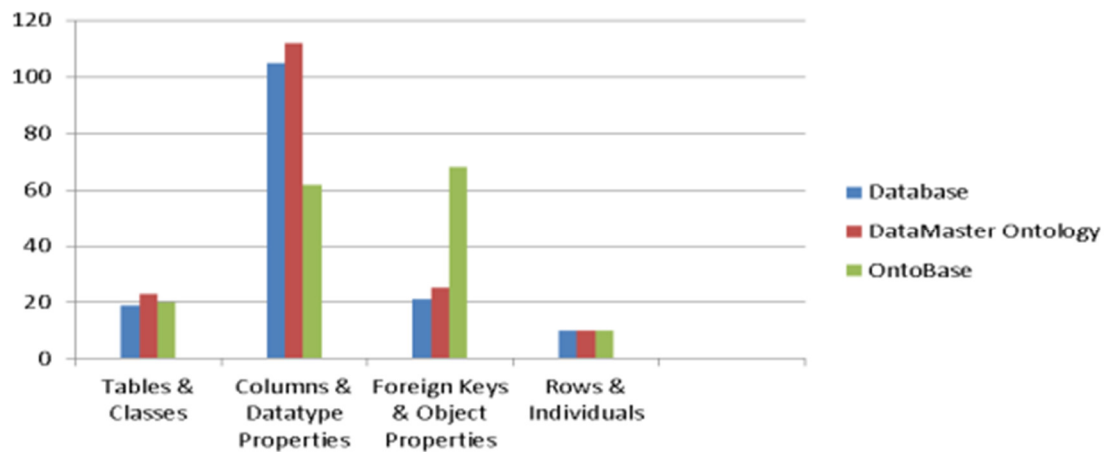


Figure 6.9: Chart of Comparison of Performances of DataMaster and OntoBase Plug-ins

Figures 6.7 and 6.8 presented separate mapping results for DataMaster and OntoBase plugins. In Figure 6.9, all the results are tallied and the mapping performance of both plugins is presented. It is shown in the left block of Figure 6.9 that 19 database tables (left bar) are mapped into 23 ontology classes in DataMaster (middle bar) and 20 ontology classes in OntoBase (right bar). In the second left block of Figure 6.9, 105 database columns (left bar) are mapped into 112 Datatype Properties in DataMaster (middle bar) and 62 Datatype Properties in OntoBase (right bar).

```

<owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#PROPERTYTYPE">
  <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</db:isBridgeTable>
  <db:hasPrimaryKeyFields rdf:datatype="http://www.w3.org/2001/XMLSchema#string">PROPERTYTYPEID</db:hasPrimaryKeyFields>
</owl:Class>
<owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ARRANGEMENT">
  <db:hasPrimaryKeyFields rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ARRANGEMENTID</db:hasPrimaryKeyFields>
  <db:hasForeignKeys>
    <db:ForeignKey rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ForeignKey_Instance_15">
      <db:hasReferenceTable rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ACCOUNT</db:hasReferenceTable>
      <db:hasLocalField rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ACCOUNTID</db:hasLocalField>
      <db:hasReferenceField rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ACCOUNTID</db:hasReferenceField>
      <db:hasRefFieldProperty>
        <owl:FunctionalProperty rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ACCOUNT_ACCOUNTID">
          <db:hasRefFieldProperty>
            <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">FK: ARRANGEMENT.ACCOUNTID ---&gt; ACCOUNT.ACCOUNTID</rdfs:label>
          </owl:FunctionalProperty>
        </db:hasRefFieldProperty>
      </db:hasRefFieldProperty>
      <db:hasRefTableClass>
        <owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ACCOUNT">
          <db:hasRefTableClass>
            <db:hasFKName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fk_ACCOUNTID_ACCOUNT_ACCOUNTID</db:hasFKName>
            <db:hasLocTableClass rdf:resource="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ARRANGEMENT">
              </db:ForeignKey>
            </db:hasForeignKeys>
          </db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</db:isBridgeTable>
        </owl:Class>
      </db:hasRefTableClass>
    </db:ForeignKey>
  </db:hasForeignKeys>
</owl:Class>
<owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#SERVICE">
  <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</db:isBridgeTable>
  <db:hasPrimaryKeyFields rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SERVICEID</db:hasPrimaryKeyFields>
</owl:Class>
<owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#CUSTOMER">
  <db:hasPrimaryKeyFields rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CUSTOMERID</db:hasPrimaryKeyFields>
  <db:hasForeignKeys>
    <db:ForeignKey rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#ForeignKey_Instance_21">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">FK: CUSTOMER.CUSTOMERTYPEID ---&gt; CUSTOMERTYPE.CUSTOMERTYPEID</rdfs:label>
      <db:hasLocTableClass rdf:resource="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#CUSTOMER">
        <db:hasReferenceTable rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CUSTOMERTYPE</db:hasReferenceTable>
        <db:hasLocFieldProperty>
          <owl:FunctionalProperty rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#CUSTOMER.CUSTOMERTYPEID">
            </db:hasLocFieldProperty>
          </db:hasLocFieldProperty>
        </db:hasLocFieldProperty>
        <db:hasRefFieldProperty>
          <owl:FunctionalProperty rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#CUSTOMERTYPE.CUSTOMERTYPEID">
            </db:hasRefFieldProperty>
          </db:hasRefFieldProperty>
        </db:hasLocalField rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CUSTOMERTYPEID</db:hasLocalField>
        <db:hasReferenceField rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CUSTOMERTYPEID</db:hasReferenceField>
        <db:hasFKName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fk_CUSTOMERTYPEID_CUSTOMERTYPE_CUSTOMERTYPEID</db:hasFKName>
        <db:hasRefTableClass>
          <owl:Class rdf:about="http://biostorm.stanford.edu/db_table_classes/DSN_idbc.oracle.thin.@localhost.1521.xe#CUSTOMERTYPE">
            </db:hasRefTableClass>
          </owl:Class>
        </db:ForeignKey>
      </db:hasForeignKeys>
    </db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</db:isBridgeTable>
  </owl:Class>

```

Figure 6.10: Part of OWL Code of the Ontology

The second right block of Figure 6.9 depicts 21 database foreign keys (left bar) that are mapped into 25 Object Properties in DataMaster (middle bar) and 68 Object Properties in OntoBase (right bar). These results reveal that DataMaster has more deviations concerning the production of an accurate Ontology from the relational database. OntoBase on the other hand had major conformance with the mapping principles in Chapter 3 Subsection 3.2.3; this conformance is witnessed in the low number of Datatype Properties (62) in the resulting ontology compared to the number of columns (105) in the input database as well as the high



number of Object Properties (68) compared to the number of foreign keys (21) in the input database.

Finally, SPARQL queries created in Chapter 5 Section 5.5 are run on the OWL code of the ontology (Figure 6.10) and CQs along with users' views of their answers are used to ascertain the ontology meets the requirements of the knowledge domain. Figure 6.11 and 6.12 shows sample SPARQL queries (CQ1 and CQ9 from Table 5.7) execution and outputs.

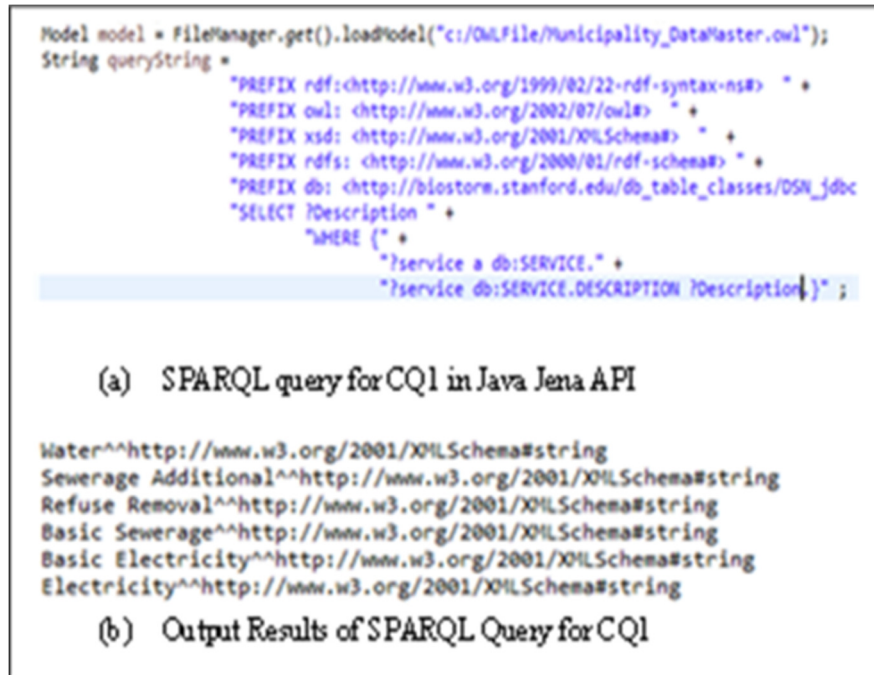


Figure 6.2: CQ1 Sample (a) SPARQL Query and (b) Outputs

The bottom part of Figure 6.11 shows the outputs of the CQ1 SPARQL query which are in this case, the list of all the service instances in the ontology. These are the services offered by the municipality to its customers.



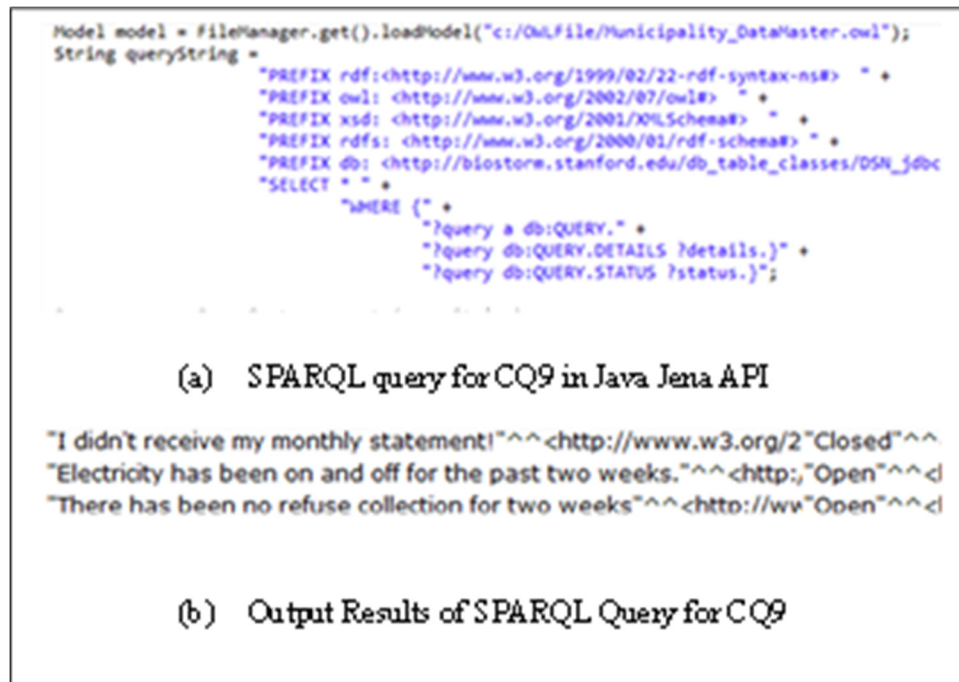


Figure 6.3: CQ9 Sample (a) SPARQL Query and (b) Outputs

The bottom part of Figure 6.12 shows the outputs of the CQ9 SPARQL query. These are the queries made by customers to the municipality.

The outputs of SPARQL queries needed to be further analysed and mapped to the CQs (Chapter 5 Section 5.4). In Bezerra et al. (2013), it was demonstrated that SPARQL queries are assertional in nature and can only output classes or instances. Further, Bezerra et al. (2013) proposed an algorithm which split an input CQ into several tokens. Thereafter, the tokens are used to retrieve concepts or instances from the ontology. The resulting concepts and instances constitute the answer to the CQ. This approach is adopted in this study to map CQs to SPARQL queries outputs.

To differentiate the authors from the end users of the ontology, three individuals were invited to participate in the study. The three participants were all owners of properties in South Africa and had good knowledge of the knowledge domain. Furthermore, the participants hold IT qualifications (MTech IT, BTech IT and BTech Computer Systems) and had good understanding of concepts of entity, class, term, instance, occurrence, etc. However, they had little knowledge of ontology.

The three participants were given the list of competency questions in Table 5.2 and asked to select in each question the terms/entities/classes that they think could be or that the instances could be the answer to the question. The lists of CQs were collected and the selected terms/entities/classes analysed against the SPARQL queries outputs as in Figures 6.13, 6.14 and 6.15.

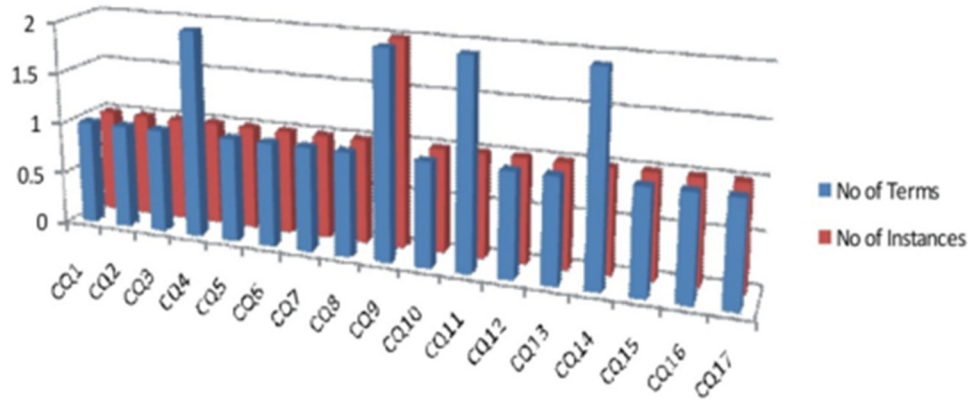


Figure 6.4: Chart of the Mapping of Terms of Participant 1 to SPARQL Outputs Instances

Figure 6.13 shows the results of the mapping of the CQs terms selected by the first participant to the outputs of the SPARQL queries. The chart in Figure 6.13 tells that one term was selected per CQ by the participant for 13 CQs and two terms for four CQs (CQ4, CQ9, CQ11, and CQ14). Further, it showed that all the selected terms had at least one instance in the SPARQL output results; in particular, the CQ9 has two instances in the output results.

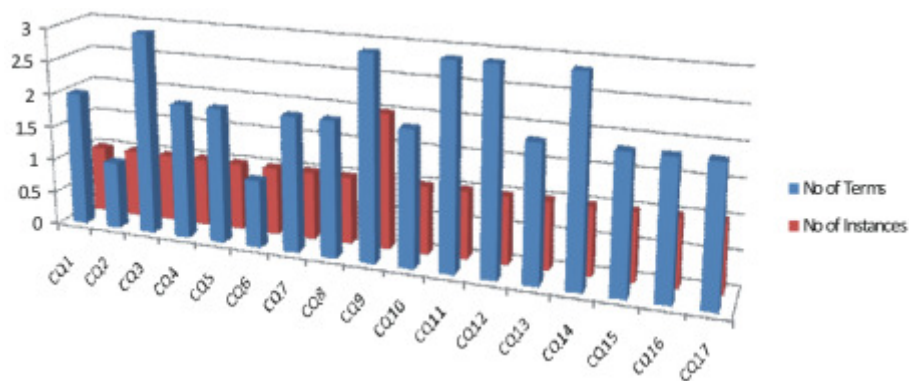


Figure 6.5: Chart of the Mapping of Terms of Participant 2 to SPARQL Outputs Instances

Similarly, Figure 6.14 shows the results of the mapping of the CQs terms selected by the second participant to the outputs of the SPARQL queries. From Figure 6.14 it can be noticed that the participant selected more terms per CQ; one term was selected for the CQs CQ2 and CQ6, two terms for CQs CQ1, CQ3, CQ7, CQ8, CQ10, CQ13, CQ15, CQ1 and CQ17, and three terms for the CQs CQ3, CQ9, CQ11, CQ12 and CQ14. Further, Figure 5 shows that all the selected terms had at least one instance in the SPARQL queries outputs, with CQ3, CQ9, CQ11, CQ12 and CQ14 having more instances in the outputs results.

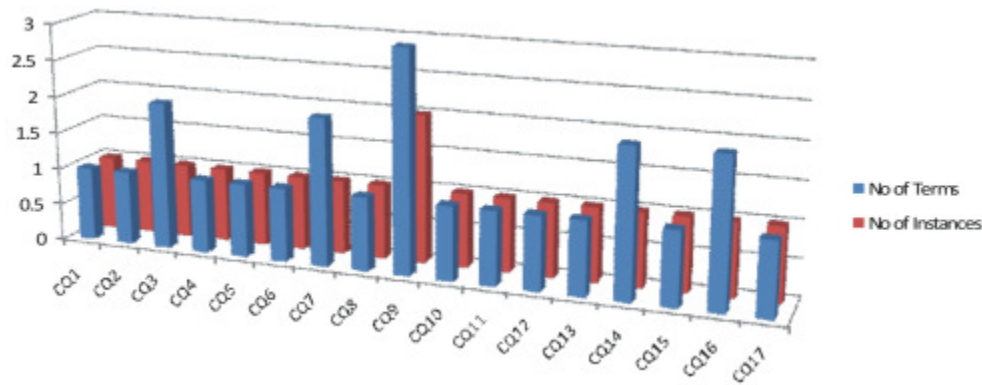


Figure 6.6: Chart of the Mapping of Terms of Participant 3 to SPARQL Outputs Instances

Lastly, the mapping of the terms chosen by the third participant to the SPARQL queries outputs is presented in Figure 6.15. It is shown that the participant selected one term in most of the CQs except for CQ3, CQ7, CQ9, CQ14 and CQ16. In particular, the participant selected three terms in CQ9. Figure 6.15 also depicts that SPARQL output results included at least one instance of each selected term by the third participant with more instances for the CQs CQ3, CQ7, CQ9, CQ14 and CQ16.

In light of the above, the output results of SPARQL queries execution against the ontology provided instances to all selected terms in the CQs. Therefore, it can be concluded that the ontology meets the requirements of the knowledge domain and can be useful in Semantic Web applications.

## 6.4 Conclusion

This chapter presented the experiments conducted during the study. The experiments included the development of a test relational database, automatic construction of ontology from the relational database, ontology verification through conceptual mapping

rules/principles, execution of SPARQL queries against the ontology and ontology competency evaluation through competency questions. Ontology verification through conceptual mapping rules/principles also included a comparison between resultant ontologies constructed with two different Protégé Plug-ins. The next chapter concludes the study by providing the summary of the work, limitations and discussion of future work.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORK**

#### **7.1 Summary of the Study**

In this study, a solution for semantic knowledge extraction and validation from relational database was proposed and practically implemented. The study began with a review of the semantic web literature to get practical ways in which data in relational databases can be exploited on the semantic web. Due to the fact that semantic exploitation of relational database is a very active research field, the review revealed shortcomings in the work that was already done. The identified shortcomings include among others many relational databases to ontology conversion tools still being in prototype state and some tools not being available to the public; and automatic ontology construction frameworks that failed to deal with an issue of ontology verification and ontology competency evaluation.

A framework for semantic knowledge extraction was then conceptualised to deal with some of the highlighted shortcomings. The framework's aim was to serve as a practical guideline from the automatic conversion of relational database to ontology, to the extraction of knowledge from the created ontology using a query application, until the verification and validation of the ontology to ensure that the ontology is valid and knowledge extracted from it is useful to the target knowledge domain.

To practically apply the proposed framework, a real life knowledge domain to base the study on had to be identified. A case study was then conducted in the broad South African municipality domain. The domain was then narrowed specifically to information systems for service delivery in municipalities. The case study produced a relational data model for municipalities in South Africa. The data model was then implemented into a test database using Oracle as RDBMS platform of choice.

The next phase of the study looked at the construction of ontologies from the relational database. The ontologies were automatically constructed from the input Oracle relational database with two Protégé plugins, namely, DataMaster and OntoBase. The semantic structures of the resulting ontologies were analysed by means of two visualization plugins including OntoGraf and OWLViz as well as an ontology documentation software, namely, Parrot. The performances of the plugins were further measured based on the database to

ontology mapping rules/principles. The results revealed that both tools reasonably convert a relational database to ontology with slight deviations from the database-to-ontology mapping principles.

This study then evaluated the verified municipality ontologies which were automatically constructed from a relational database against requirements of the business domain. The requirements of the domain were modelled with competency questions which were executed against the ontology through SPARQL queries. The study applied a combination of Tropos Methodology and Competency Question Translation Approach to get a set of competency questions from the municipality knowledge domain and convert them into executable SPARQL queries. The results of the SPARQL queries were analysed using input from three participants who participated as users with an interest in the input ontology. The results from the analysis revealed that the input ontology does satisfy the requirements of the business domain (Municipality) based on the set of competency questions it was able to successfully answer to the participating users' expectations

## **7.2 Limitations, Recommendations and Future Work**

During the study a few limitations and challenges which could lead to opportunities for further research were noted. The challenges and limitations are as follows:

- To practically apply the proposed semantic knowledge extraction framework, a relational database was developed locally to run the experiments. However it was noted that the relational database is not large enough. Even though the database enables to run small scale experiments as prove of concept; there is an opportunity to investigate possibilities of repeating the experiments with larger relational databases to test flexibility and scalability of some of our introduced approaches.
- The proposed framework applies only one input relational database. This can be expanded by investigating possibilities of applying more than one relational database as input.
- Due to availability and ease of use, the study opted for Protégé Plug-ins (DataMaster and OntoBase) to convert the input relational database into ontology. To address this, the future direction of the research would be to expand the study with other Semantic Web tools that are different from Protégé.

## 7.3 Conclusion

This chapter concluded the study by providing a comprehensive summary of the work which included the main achievements and contributions of the study, amongst others. The chapter also fully highlighted the main limitations and challenges, including suggestions on how the challenges can be tackled through further research. The proposed framework for semantic knowledge extraction contributes directly to the main aim and challenge of semantically exploiting data stored in relational databases. Our ontology verification approach can also be applied in the semantic web community to verify ontologies automatically constructed from relational databases. Lastly, our ontology competency evaluation approach offers an alternative way to confirm that created ontologies do meet requirements and expectations of their target knowledge domains.

## BIBLIOGRAPHY

ABID, A.S. & JAVED, M.Y. (2013) "Moving Towards Semantic Web: Relational Schema to Ontology", *International Journal of Computer Theory and Engineering*, Vol. 5, No.2, pp. 201-203.

ALATRISH, E.S. (2012) "Comparison of Ontology Editors", *eRAF Journal on Computing*, Vol. 4, No 2012, pp. 23-38.

ANNAMALAI, M. & SANIP, Z. (2010) "Natural Language Support for Competency Evaluation of Web-Ontologies", *Journal of IT in Asia*, Vol. 3, No. 1, pp. 37-51.

ASTROVA, I. & STANTIC, B. (2004) "Reverse Engineering of Relational Databases to Ontologies: An Approach Based on an Analysis of HTML Forms", In the Proceedings of 1<sup>st</sup> European Semantic Web Symposium (ESWS), Crete, Greece, 10-12 May, pp. 327-341.

AUER, S. & IVES, Z.G. (2007) "Integrating Ontologies and Relational Data", Technical Reports (CIS), Department of Computer and Information Science, University of Pennsylvania, USA.

BASILI, V.R & TURNER, A.J. (1975) "Iterative Enhancement : A practical technique for Software Development", *IEEE Transactions on Software Engineering*, Vol. 1, No. 4, pp. 390-396.

BERNERS-LEE, T., HENDLER, J. & LISSILA, O. (2001) The Semantic Web. [Online]. Available at < [http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American %20Feature%20Article %20The %20Semantic%20Web %20May%202001.pdf](http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American%20Feature%20Article%20The%20Semantic%20Web%20May%202001.pdf)> Accessed: 31.01.2015.

BEZERRA, C., FREITAS, F. & SANTANA, F. (2013) "Evaluating Ontologies with Competency Questions", In Proceedings of the 2013 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp. 284-285.

BUITELAAR, P., OLEJNIK, D. & SINTEK, M. (2004) "A Protégé Plug-in for Ontology Extraction from Text Based on Linguistic Analysis", In the Proceedings of the 1st European Semantic Web Symposium (ESWS 2004), Heraklion, Greece, May.

CERBAH, F. (2008) "Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies", In the Proceedings IEEE/WIC/ACM International Joint Conference on Web Intelligence (WI'08) and Intelligent Agent Technology (IAT'08), Sydney, Australia, 9-12 December, pp. 553-557.

CHEN, H., WANG, Y., WANG, H., MAO, Y., TANG, J., ZHOU, C., YIN, A. & WU, Z. (2006) "Towards a Semantic Web of Relational Databases : a practical Semantic Toolkit and an In-use case from Traditional Chinese Medicine", In the Proceedings of the 5<sup>th</sup> International Semantic Web Conference, Athens, GA, USA, 5-9 November, pp. 750-763.

CORCHO, O., FERNANDEZ-LOPEZ, M., & GOMEZ-PEREZ, A. (2003) "Methodologies, tools and languages for building ontologies. Where is their meeting point?", *Data & Knowledge Engineering*, Vol. 46, pp. 41-64.



CRISTANI, M. & CUEL, R. (2004) “A Comprehensive Guideline for Building a Domain Ontology from Scratch”, In the Proceedings of WWW 2004 – Session: Ontology Representation and Querying: RDF and SPARQL, Graz, Austria, 30 June - 2 July, pp. 205-212.

CULLOT, N., GHAWI, R., & YTONGNON, K. (2007) “DB2OWL: A Tool for Automatic Database-to-Ontology Mapping”, In: Michelangelo Ceci; Donato Malerba & Letizia Tanca, ed., ‘SEBD’, pp. 491-494.

DOU, D., QIN, H., & LEPENDU, P. (2012) “Ontograte: Towards Automatic Integration for Relational Databases and the Semantic Web through an Ontology-Based Framework”, *International Journal Semantic Computing*, Vol. 4, No.1, pp. 123-151.

EMFULENI LOCAL MUNICIPALITY. (2014a) Tariff Policy 2013/2014 Financial Year, [Online]. Available at:<[http://www.emfuleni.gov.za/images/docs/policies/tariff\\_policy201314.pdf](http://www.emfuleni.gov.za/images/docs/policies/tariff_policy201314.pdf)> .Accessed: 31.01.2015.

EMFULENI LOCAL MUNICIPALITY. (2014b) Property Rates Policy 2013/2014 Financial Year, Available at:<[http://www.emfuleni.gov.za/images/docs/policies/property\\_rates\\_policy201314.pdf](http://www.emfuleni.gov.za/images/docs/policies/property_rates_policy201314.pdf)> .Accessed: 31.01.2015.

FALCONER, S. (2010) OntoGraf. In: ProtegeWiki. [Online]. Available at :<<http://protegewiki.stanford.edu/wiki/OntoGraf>>.Accessed:20.05.2014.

FOUNDATION CTIC (2014) Home page. [Online]. Parrot: A RIF and OWL documentation service. Available at:< <http://ontorule-project.eu/parrot/parrot>>. Accessed: 20.05.2014.

FERNANDES, B.C.B., GUIZZARDI, R.S.S. & GUIZZARDI, G. (2011) “Using Goal Modeling to Capture Competency Questions in Ontology-based Systems”, *Journal of Information and Data Management*, Vol 2, No. 3, pp. 527-540.

FONOU-DOMBEU, J.V. & HUISMAN, M. (2011) “Semantic-Driven e-Government: Application of Uschold and King Ontology Building Methodology for Semantic Ontology Models Development”, *International Journal of Web & Semantic Technology (IJWesT)*, Vol.2, No.4.

GALI, A., CHEN, C.X., CLAYPOOL, K.T. & UCEDA-SOSA, R. (2005) “From Ontology to Relational Database”, *Shan Wang et al. (Eds): Conceptual Modelling for Advanced Application Domains*, Vol. 3289, pp. 278-289.

GANGEMI, A. (2005) “Ontology Design Patterns for Semantic Web Content”, *ISWC 2005*, LNCS 3729, pp. 262-276, 2005.

GENNARI, J., NGUYEN, M. & SILBERFEIN, A. (2007) DataGenie. In: ProtegeWiki. [Online]. Available at:<<http://protege.cim3.net/cgi-bin/wiki.pl?DataGenie>>.Accessed: 10.04.2014.

GHARABI, N., ADDAKIRI, K. & BAHAI, M. (2012) "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.

GREEN, D. & RUHLE, G. (2004) "Software release planning: an evolutionary and iterative approach", *Information and Software Technology*, Vol. 46, No. 2004, pp. 243-253

GRUBER, T.R. (1993) "A translation approach to portable ontology specification", *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220.

HORRIDGE, M. (2010) OWLViz. In: ProtegeWiki. [Online]. Available at: <<http://protegewiki.stanford.edu/wiki/OWLViz>>. Accessed :20.05.2014.

HU, W. & QU, Y. (2007) "Discovering Simple Mappings Between Relational Database Schemas and Ontologies", In Proceedings of the 6th International Semantic Web Conference, Busan, Korea, 11-15 November, pp. 225-238.

IMANDI, N. & RIZVI, S.A.M. (2012) "An Approach to OWL Concept Extraction and Integration across Multiple Ontologies", *International Journal of Web & Semantic Technology (IJWesT)*, Vol. 3, No.3, pp. 33-51.

JAIN, V. & SINGH, M. (2013) "A framework to Convert Relational Database to Ontology For Knowledge Database in Semantic Web", *International Journal of Scientific & Technology Research*, Vol. 2, No. 10, pp. 9-12.

JIA, C. & YUE, W. (2009) "Rules-based object-relational databases ontology construction", *Journal of Systems Engineering and Electronics*, Vol. 20, No.1, pp. 211-215.

KHOZOIE, N. (2011) "Health Information Management on Semantic Web: (Semantic HIM)", *International Journal of Web & Semantic Technology (IJWesT)*, Vol. 3, No.1, pp. 61-68.

KOMA, S.B. (2010) "The state of Local Government in South Africa: Issues, trends and options", *Journal of Public Administration*, Vol. 45, No.1.1, pp. 111-120.

LACLAVIK, M. (2006) "RDB2Onto: Relational Database Data to Ontology Individuals Mapping", In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. Navrat et al. (Eds), pp. 86-89.

LEMAIGNAN, S., SIADAT, A., DANTAN, J. & SEMENENKO, A. (2006) "MASON: A Proposal for an Ontology of Manufacturing Domain", In the Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its Applications, Prague, Czech Republic, June.

LEVSHIN, D.V. (2010) "Mapping Relational Databases to the Semantic Web with Original Meaning", *International Journal of Software Informatics*, Vol. 4, No.1, pp. 23-37.

LI, M., DU, X. & WANG, S. (2005) "Learning Ontology from Relational Database", In the Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18-21 August, pp. 3410-3415.

- LIN, C.Y. (2008) "Migrating to Relational Systems: Problems, Methods, and Strategies", *Contemporary Management Research*, Vol. 4, No. 4, pp. 369-380.
- LIN, Y. & SAKAMOTO, N. (2009) Ontology Driven Modeling for the Knowledge of Genetic Susceptibility to Disease, *Kobe J. Med. Sci.*, Vol. 55, No. 3, pp. E53-E66.
- LU, S., DONG, M., & FOTOUHI, F. (2002) "The Semantic Web: Opportunities and challenges for next-generation Web applications", *Information Research*, Vol. 7, No. 4
- MADHU, G., GOVARDHAN, A., & RAJINIKANTH, T.V. (2011) "Intelligent Semantic Web Search Engines: A Brief Survey", *International Journal of Web & Semantic Technology (IJWesT)*, Vol. 2, No.1, pp. 34-42.
- MAHMOOD, N., BURNEY A., & AHSAN, K. (2010) "A logical Temporal Relational Model", *International Journal of Computer Science Issues*, Vol. 7, No. 1, pp. 1-9.
- MENG, G., LING, H. & ZHOU, S. (2010) "Ontologies Acquisition from Relational Database", *Computer and Information Science*, Vol. 3, No.1, pp. 185-187.
- MOGOTLANE, K.D. & FONOU-DOMBEU, J.V. (2014a) "Development of a Data Model for Semantic Exploitation of Municipality Records in South Africa," In the Proceedings of the Information Society Technology of Africa (ISTAfrica 2014) Conference, Pointe aux Piments, Mauritius, 5-9 May, pp. 1-7.
- MOGOTLANE, K.D. & FONOU-DOMBEU, J.V. (2014b) "Comparison of Protégé Plug-ins Performance in Automatic Construction of Ontology from Relational Databases, In the Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 18-20 December.
- MULLIGANN, C., TRAME, J. & KRZYSZTOF, J. (2011) "Introducing the new SIM-DLA Semantic Similarity Measurement Plug-in for the Protégé Ontology Editor", In the Proceedings of the 1st ACM SIGSPATIAL International Workshop on Spatial Semantics and Ontologies, Chicago, USA, Nov.
- NEMURAITÉ, L. & PARADAUSKAS, B. (2012) "A Methodology for Engineering OWL 2 Ontologies in Practise Considering their Semantic Normalisation and Completeness", *Electronics and Electrical Engineering*, Vol.4, No.120, pp. 89-94.
- Navathe, S.B. (1992) "Evolution of Data Modelling for Databases", *Communications of the ACM*, Vol. 35, No. 9, pp. 112-123.
- NYULAS, C., O'CONNOR, M. & TU, S. (2007) "DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé", In the Proceedings of the 10<sup>th</sup> International Protégé Conference, Budapest, Hungary, 15-18 July.
- Papapanagiotou, P., Katsioulis, P., Tsetsos, V., Anagnostopoulos, C. & Hadjiefthymiades, S. (2006) "RONTOS: Relational to Ontology Schema Matching", *AIS SIGSEMIS Bulletin*, Vol. 3, No. 3, pp. 32-36.

PASHA, M. & SATTAR, A. (2012) “Building Domain Ontologies From Relational Database Using Mapping Rules”, *International Journal of Intelligent Engineering & Systems*, Vol. 5, No.1, pp. 20-27.

PORWOL, L., OJO, A., & J. BRESLIN, J. (2014) “A Semantic Model for e-Participation – Detailed Conceptualisation and Ontology”, In the Proceedings of the 15th Annual International Conference on Digital Government Research, Aguascalientes City, Mexico, 18-21 Jun, pp. 263-272.

REPUBLIC OF SOUTH AFRICA. (1996) Constitution of the Republic of South Africa, [Online]. Available at :<<http://www.thehda.co.za/uploads/images/unpan005172.pdf>> .Accessed: 31-01-2015.

REPUBLIC OF SOUTH AFRICA. (2000) Local Government Municipal Systems Act 32 of 2000, [Online]. Available at: <[http://www.saflii.org.za/za/legis/num\\_act/lgmsa2000384.pdf](http://www.saflii.org.za/za/legis/num_act/lgmsa2000384.pdf)> .Accessed: 31-01-2015.

SADEH, T. & WALKER, J. (2003) “Library Portals: Toward the Semantic Web”, *New Library World*, Vol. 104, No.1184, pp. 11-19.

SEDIGHI, S.M. & JAVIDAN, R. (2012) “Semantic query in a relational database using local ontology construction”, *South African Journal of Science*, Vol. 108, No. 11/12, pp. 1-10.

SALEH, M.E. (2011) “Semantic-Based Query in Relational Database using Ontology”, *Canadian Journal on Data, Information and Knowledge Engineering*, Vol. 2, No. 1, pp. 1-16.

SEQUEDA, J.F., MARCELO, A. & MIRANKER, D.P. (2012) “On Directly Mapping Rational Databases to RDF and OWL”, In the Proceedings of WWW 2012 – Session: Ontology Representation and Querying: RDF and SPARQL, Lyon, France, 16-20 April, pp. 649-658.

SHETH, A., RAMAKRISHNAN, C., & THOMAS, C. (2005) “Semantics for the Semantic Web: The Implicit, the Formal and the Powerful”, *International Journal on Semantic Web & Information Systems*, Vol. 1, No.1, pp. 1-18.

SPANOS, D., STRAVROU, P., & MITROU, N. (2012) “Bringing Relational Databases into the Semantic Web: A Survey”, *Semantic Web Journal*, Vol. 3, No.2, pp. 169-209.

TELNAROVA, Z. (2010) “Relational database as a source of ontology creation”, In the Proceedings of the International Multi-conference on Computer Science and Information Technology, Wisla, Poland, 18-20 October, pp. 135-139.

TIRMIZI, S.H., SEQUEDA, J. & MIRANKER, D. (2008) “Translating SQL Applications to the Semantic Web”, In the Proceedings of the 19<sup>th</sup> International Conference, DEXA 2008, Turin, Italy, 1-5 September, pp. 450-464.

TRISSEL, S. & LESSER, U. (2005) “Querying Ontologies in Relational Database Systems”, In Proceedings of the Second International Workshop, CA, USA, 20-22 July, pp. 63-79.

USCHOLD, M. & KING, M. (1995) "Towards a Methodology for Building Ontologies," In Proceedings of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, pp. 1-13.

VIVIAN, O.M. & HEUSER, C.A. (2002) "Semiautomatic generation of data-extraction ontologies from relational databases", In the Proceedings of the Brazilian Symposium on Databases (SBBDB), Rio Grande do Sul, Brazil, 14-16 October, pp. 252-262.

YABLOKO, N. (2009) OntoBase. In: ProtegeWiki. [Online]. Available at:<<http://protegewiki.stanford.edu/wiki/OntoBase>>.Accessed :10.04.2014.

ZEMMOUCHI-GHOMARI, L. & GHOMARI, A.R. (2013a) "Translating Natural Language Competency Questions into SPARQL Queries: a Case Study", In the Proceedings of The First International Conference on Building and Exploring Web Based Environments, Seville, Spain, 27 January – 1 February, pp. 81-86.

ZEMMOUCHI-GHOMARI, L. & GHOMARI, A.R. (2013b) "Process of Building Reference Ontology for Higher Education", In the Proceedings of World Congress on Engineering, London, UK, 3-5 July.

ZHANG, J. (2007) "Ontology and the Semantic Web", In the Proceedings of the North American Symposium on Knowledge Organisation, Toronto, Canada, 1-5 September, pp. 9-20.

ZHANG, L. & LI, K. (2011) "Automatic Generation of Ontology Based on Database", *Journal of Computational Information Systems*, Vol. 7, No. 4, pp. 1148-1154.

ZHOU, S., LING, H., HAN, M., & ZHANG, H. (2010) "Ontology Generator from Rational Database on Jena", *Computer and Information Science Technology*, Vol. 3, No.2, pp. 263-267.

## APPENDIX A

### South African Municipalities covered during Knowledge Domain Modelling (Case Study)

Table A.1 : List of South African Municipalities

Province	Municipality
Gauteng	Lesedi Local Municipality
Gauteng	Ekurhuleni Metropolitan Municipality
Gauteng	Midvaal Local Municipality
Gauteng	Merafong City Local Municipality
Gauteng	Mogale City Local Municipality
Gauteng	Randfontein Local Municipality
Gauteng	Westonaria Local Municipality
Gauteng	City of Johannesburg Metropolitan Municipality
Gauteng	City of Tshwane Metropolitan Municipality
Mpumalanga	Albert Luthuli Local Municipality
Mpumalanga	eMalahleni Local Municipality
Mpumalanga	Steve Tshwete Local Municipality
Mpumalanga	Mbombela Local Municipality
Mpumalanga	Goven Mbeki Local Municipality
Mpumalanga	Thaba Chweu Local Municipality
Mpumalanga	Victor Khanye Local Municipality
Mpumalanga	Umjindi Local Municipality
Mpumalanga	Msukaligwa Local Municipality
KwaZulu Natal	Ethekwini Metropolitan Municipality
KwaZulu Natal	Msunduzi Local Municipality
KwaZulu Natal	Newcastle Local Municipality
KwaZulu Natal	Richmond Local Municipality
KwaZulu Natal	Umhlathuze Local Municipality
KwaZulu Natal	Hibiscus Coast Local Municipality
KwaZulu Natal	Emnambithi-Ladysmith Local Municipality
KwaZulu Natal	Umtshezi Local Municipality
KwaZulu Natal	Endumeni Local Municipality
Western Cape	George Local Municipality
Western Cape	Knysna Local Municipality
Western Cape	Stellenbosch Local Municipality
Western Cape	Breede Valley Local Municipality
Western Cape	Beaufort West Local Municipality
Western Cape	Drakenstein Local Municipality
Western Cape	Witzenberg Local Municipality
Western Cape	Mossel Bay Local Municipality
Western Cape	City of Cape Town Metropolitan Municipality
Free State	Metsimaholo Local Municipality
Free State	Dihlabeng Local Municipality

Free State	Matjhabeng Local Municipality
Free State	Moqhaka Local Municipality
Free State	Setsoto Local Municipality
Free State	Tswelopele Local Municipality
Free State	Letsemeng Local Municipality
Free State	Mangaung Metropolitan Municipality
Free State	Naledi Local Municipality
North West	Tlokwe Local Municipality
North West	Rustenburg Local Municipality
North West	City of Matlosana Local Municipality
North West	Madibeng Local Municipality
North West	Ditsobotla Local Municipality
North West	Ramotshere Moiloa Local Municipality
North West	Mahikeng Local Municipality
North West	Ventersdorp Local Municipality
North West	Lekwa-Teemane Local Municipality
Northern Cape	Khara Hais Local Municipality
Northern Cape	Ga-Segonyana Local Municipality
Northern Cape	Gamagara Local Municipality
Northern Cape	Emthanjeni Local Municipality
Northern Cape	Umsobomvu Local Municipality
Northern Cape	Tsantsabane Local Municipality
Northern Cape	Richtersveld Local Municipality
Northern Cape	Ubuntu Local Municipality
Northern Cape	Sol Plaatje Local Municipality
Limpopo	Ba-Phalaborwa Local Municipality
Limpopo	Elias Motsoaledi Local Municipality
Limpopo	Ephraim Mogale Local Municipality
Limpopo	Greater Tzaneen Local Municipality
Limpopo	Greater Letaba Local Municipality
Limpopo	Makhado Local Municipality
Limpopo	Lephalale Local Municipality
Limpopo	Mogalakwena Local Municipality
Limpopo	Polokwane Local Municipality
Eastern Cape	Makana Local Municipality
Eastern Cape	Ndlambe Local Municipality
Eastern Cape	Port St John Local Municipality
Eastern Cape	Nelson Mandela Bay Metropolitan Municipality
Eastern Cape	Kouga Local Municipality
Eastern Cape	Maletswai Local Municipality
Eastern Cape	Matatiele Local Municipality
Eastern Cape	Buffalo City Metropolitan Municipality
Eastern Cape	Amahlathi Local Municipality

## APPENDIX B

### Relational Data Modelling

Relations	Primary Key	Foreign Key
Account (AccountID, CustomerID, PropertyID, DateOpened, Balance, Status)	AccountID	CustomerID refers to CustomerID in Customer PropertyID refers to PropertyID in Property
AccountService (AccountServiceID, AccountID, PropertyServiceID, Date, Status, Consumption, Reading)	AccountServiceID	AccountID refers to AccountID in Account PropertyServiceID refers to PropertyServiceID in PropertyService
Arrangement (ArrangementID, AccountID, Date, Balance, Instalment, Status)	ArrangementID	AccountID refers to AccountID in Account
Arrears (ArrearsID, AccountID, Date, Age, Amount)	ArrearsID	AccountID refers to AccountID in Account
CustomerType (CustomerTypeID, Description)	CustomerTypeID	None
Customer (CustomerID, CustomerTypeID, Name, ID-RegistrationNo, PhysicalAddress, PostalAddress, Telephone, Cellphone, Email)	CustomerID	CustomerTypeID refers to CustomerTypeID in CustomerType
Employee (EmployeeID, ManagerID, Name, Surname, Telephone, Cellphone, Email)	EmployeeID	ManagerID refers to ManagerID in Manager
Manager (ManagerID, Name, Surname, Telephone, Cellphone, Email)	ManagerID	None
Payment (PaymentID, AccountID, Date, Method, Amount)	PaymentID	AccountID refers to AccountID in Account
Penalty (PenaltyID, AccountID, Date, Age, Amount, Interest, Description)	PenaltyID	AccountID refers to AccountID in Account
Property (PropertyID, PropertyTypeID, StandNo, StandAddress, RegisteredOwner, OwnerAddress, DateRegistered, MarketValue, Status)	PropertyID	PropertyTypeID refers to PropertyTypeID in PropertyType
PropertyService (PropertyServiceID, ServiceID, PropertyTypeID)	PropertyServiceID	ServiceID refers to ServiceID in Service PropertyTypeID refers to PropertyTypeID in PropertyType
PropertyType (PropertyTypeID, Description)	PropertyTypeID	None
Query (QueryID, CustomerID, Status, Type, DateEntered, DateClosed, Details, AttendedBy)	QueryID	CustomerID refers to CustomerID in Customer AttendedBy refers to EmployeeID in Employee
Rebate (RebateID, AccountID, Amount, Description)	AccountID	AccountID refers to AccountID in Account
Service (ServiceID, Description, Type)	ServiceID	None
Tarif (TarifID, ServiceID, PropertyTypeID, StartDate, EndDate, Price)	TarifID	ServiceID refers to ServiceID in Service PropertyTypeID refers to PropertyTypeID in PropertyType
ValuationRoll (ValuationRollID, ValuationRollTypeID, PropertyID, LastValuer, StartDate, EndDate, MarketValue)	ValuationRollID	ValuationRollTypeID refers to ValuationRollTypeID in ValuationRollType PropertyTypeID refers to PropertyTypeID in PropertyType LastValuer refers to EmployeeID in Employee
ValuationRollType (ValuationRollTypeID, Description, Duration)	ValuationRollTypeID	None

Figure B.1: Input Relational Database Schema



## Competency Questions and their SPARQL Queries Code

### 1. CQ1: What are the Services offered by the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?Description
      WHERE {
          ?service a db:SERVICE.
          ?service db:SERVICE.DESCRPTION ?Description. }
```

### 2. CQ2: What are the types of Services offered by the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT DISTINCT ?Type
      WHERE {
          ?service a db:SERVICE.
          ?service db:SERVICE.TYPE ?Type. }
```

### 3. CQ3: Which Services are consumables in the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?Description
      WHERE {
          ?service a db:SERVICE.
          ?service db:SERVICE.DESCRPTION ?Description.
          ?service db:SERVICE.TYPE ?type FILTER (?type = "Consumption").}
```

### 4. CQ4: Which Services are basic in the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>
```

```

SELECT ?Description
  WHERE {
    ?service a db:SERVICE.
    ?service db:SERVICE.DESCRPTION ?Description.
    ?service db:SERVICE.TYPE ?type FILTER (?type = "Basic").}

```

5. CQ5: How many customers do we have in our municipality?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT (count (?custID) AS ?count)
  WHERE {
    ?cust a db:CUSTOMER.
    ?cust db:CUSTOMER.CUSTOMERID ?custID.}

```

6. CQ6: What are the names of our Customers?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?name
  WHERE {
    ?cust a db:CUSTOMER.
    ?cust db:CUSTOMER.NAME ?name.}

```

7. CQ7: What types of Customers are catered for in the Municipality?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?Description
  WHERE {
    ?ctype a db:CUSTOMERTYPE.
    ?ctype db:CUSTOMERTYPE.DESCRPTION ?Description. }

```

8. CQ8: What are the overall queries in the Municipality?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

```

```

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?details
  WHERE {
    ?query a db:QUERY.
    ?query db:QUERY.DETAILS ?details.}

```

#### 9. CQ9: What are the details and status of the current customer queries?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT *
  WHERE {
    ?query a db:QUERY.
    ?query db:QUERY.DETAILS ?details.
    ?query db:QUERY.STATUS ?status.}

```

#### 10. CQ10: What are the types of valuation rolls in the municipality?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?description
  WHERE {
    ?query a db:VALUATIONROLLTYPE.
    ?query db:VALUATIONROLLTYPE.DESCRPTION ?description.}

```

#### 11. CQ11: How much is the highly rated property within the Municipality?

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT (MAX (?value) AS ?value)
  WHERE {
    ?prop a db:PROPERTY.
    ?prop db:PROPERTY.STANDADDRESS ?address.}

```

?prop db:PROPERTY.MARKETVALUE ?value}

12. CQ12: What is the address of the most valued property in the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>
SELECT ?address WHERE {
    ?prop a db:PROPERTY.
    ?prop db:PROPERTY.STANDADDRESS ?address.
    ?prop db:PROPERTY.MARKETVALUE ?value FILTER (?value =
800000.0) }
```

13. CQ13: How many properties do we have in the Municipality?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT (count (?propID) AS ?count)
WHERE {
    ?prop a db:PROPERTY.
    ?prop db:PROPERTY.PROPERTYID ?propID.}
```

14. CQ14: How many Services are offered for Residential Properties?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT (count (?ptID) AS ?count)
WHERE {
    ?pservice a db:PROPERTYSERVICE.
    ?pservice db:PROPERTYSERVICE.PROPERTYTYPEID ?ptID.
    ?pservice db:PROPERTYSERVICE.PROPERTYTYPEID ?type FILTER (?type =
21.0).}
```

15. CQ15: What are the ID's of Customers who put in queries?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>
```

```
SELECT ?custID
WHERE {
    ?query a db:QUERY.
    ?query db:QUERY.CUSTOMERID ?custID.}
```

16. CQ16: What are the closed queries from the Customers?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

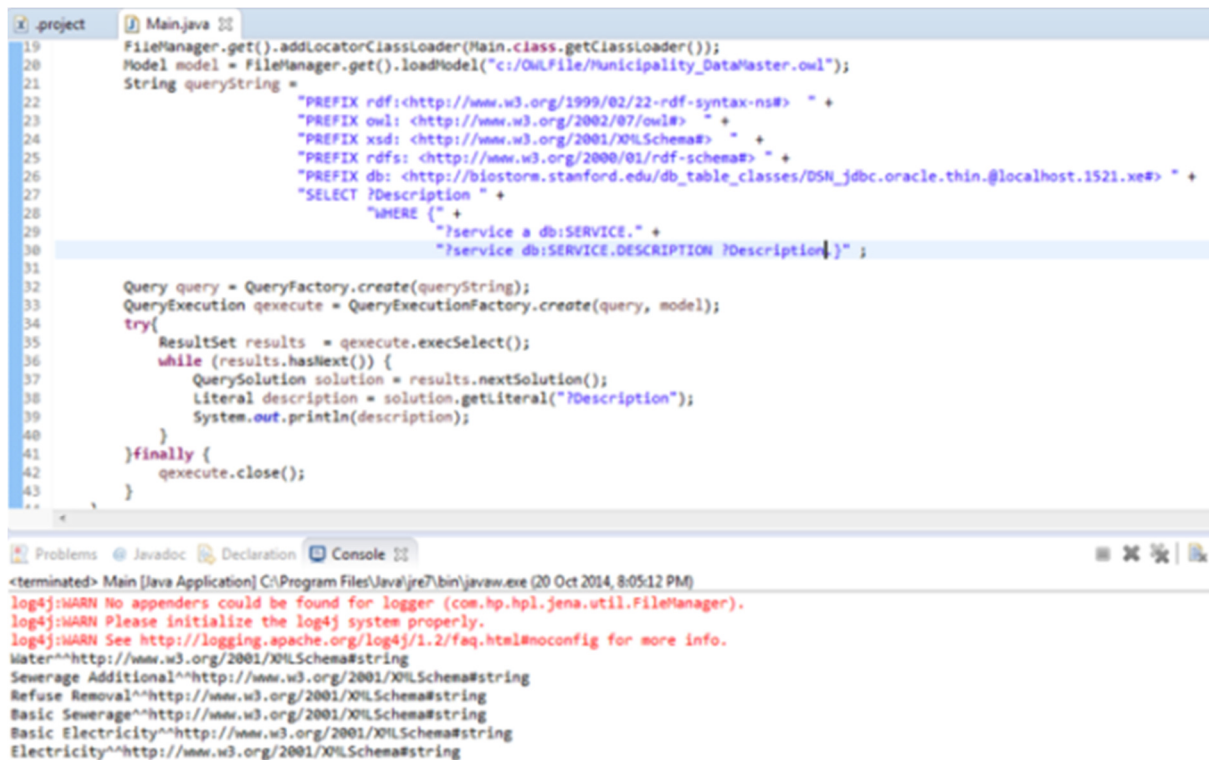
SELECT ?details
WHERE {
    ?query a db:QUERY.
    ?query db:QUERY.DETAILS ?details.
    ?query db:QUERY.STATUS ?status FILTER (?status = "Closed").}
```

17. CQ17: What are the current open queries from the Customers?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db:
<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#>

SELECT ?details
WHERE {
    ?query a db:QUERY.
    ?query db:QUERY.DETAILS ?details.
    ?query db:QUERY.STATUS ?status FILTER (?status = "Open").}
```

## Some of Java and Jena API Screenshots



```

19 FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20 Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21 String queryString =
22     "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23     "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24     "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25     "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26     "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27     "SELECT ?Description " +
28     "WHERE { " +
29     "    ?service a db:SERVICE. " +
30     "    ?service db:SERVICE.DESCRPTION ?Description. }";
31
32 Query query = QueryFactory.create(queryString);
33 QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34 try{
35     ResultSet results = qexecute.execSelect();
36     while (results.hasNext()) {
37         QuerySolution solution = results.nextSolution();
38         Literal description = solution.getLiteral("?Description");
39         System.out.println(description);
40     }
41 }finally {
42     qexecute.close();
43 }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 8:05:12 PM)

log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Water^http://www.w3.org/2001/XMLSchema#string

Sewerage Additional^http://www.w3.org/2001/XMLSchema#string

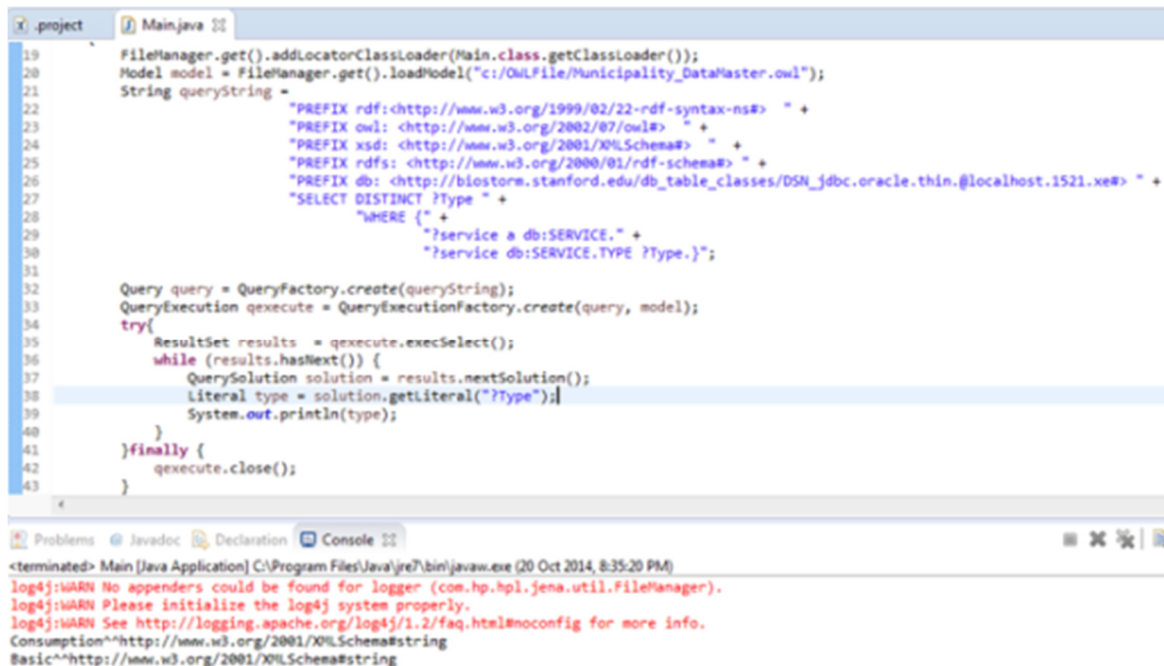
Refuse Removal^http://www.w3.org/2001/XMLSchema#string

Basic Sewerage^http://www.w3.org/2001/XMLSchema#string

Basic Electricity^http://www.w3.org/2001/XMLSchema#string

Electricity^http://www.w3.org/2001/XMLSchema#string

Figure B.2: CQ1 in Jena



```

19 FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20 Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21 String queryString =
22     "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23     "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24     "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25     "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26     "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27     "SELECT DISTINCT ?Type " +
28     "WHERE { " +
29     "    ?service a db:SERVICE. " +
30     "    ?service db:SERVICE.TYPE ?Type. }";
31
32 Query query = QueryFactory.create(queryString);
33 QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34 try{
35     ResultSet results = qexecute.execSelect();
36     while (results.hasNext()) {
37         QuerySolution solution = results.nextSolution();
38         Literal type = solution.getLiteral("?Type");
39         System.out.println(type);
40     }
41 }finally {
42     qexecute.close();
43 }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 8:35:20 PM)

log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Consumption^http://www.w3.org/2001/XMLSchema#string

Basic^http://www.w3.org/2001/XMLSchema#string

Figure B.3: CQ2 in Jena

```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT ?Description " +
28      "WHERE {" +
29          "?service a db:SERVICE." +
30          "?service db:SERVICE.DESCRPTION ?Description." +
31          "?service db:SERVICE.TYPE ?type FILTER( ?type = \"Consumption\").}";
32
33  Query query = QueryFactory.create(queryString);
34  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
35  try{
36      ResultSet results = qexecute.execSelect();
37      while (results.hasNext()) {
38          QuerySolution solution = results.nextSolution();
39          Literal description = solution.getLiteral("?Description");
40          System.out.println(description);
41      }
42  }finally {
43      qexecute.close();
44  }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 9:14:20 PM)

log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Water^^http://www.w3.org/2001/XMLSchema#string

Electricity^^http://www.w3.org/2001/XMLSchema#string

Figure B.4: CQ3 in Jena

```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT ?Description " +
28      "WHERE {" +
29          "?service a db:SERVICE." +
30          "?service db:SERVICE.DESCRPTION ?Description." +
31          "?service db:SERVICE.TYPE ?type FILTER( ?type = \"Basic\").}";
32
33  Query query = QueryFactory.create(queryString);
34  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
35  try{
36      ResultSet results = qexecute.execSelect();
37      while (results.hasNext()) {
38          QuerySolution solution = results.nextSolution();
39          Literal description = solution.getLiteral("?Description");
40          System.out.println(description);
41      }
42  }finally {
43      qexecute.close();
44  }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 9:19:23 PM)

log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Sewerage Additional^^http://www.w3.org/2001/XMLSchema#string

Refuse Removal^^http://www.w3.org/2001/XMLSchema#string

Basic Sewerage^^http://www.w3.org/2001/XMLSchema#string

Basic Electricity^^http://www.w3.org/2001/XMLSchema#string

Figure B.5: CQ4 in Jena

```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT (count (?custID) AS ?count) " +
28      "WHERE {" +
29      "    ?cust a db:CUSTOMER." +
30      "    ?cust db:CUSTOMER.CUSTOMERID ?custID.}";
31
32  Query query = QueryFactory.create(queryString);
33  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34  try{
35      ResultSet results = qexecute.execSelect();
36      while (results.hasNext()) {
37          QuerySolution solution = results.nextSolution();
38          Literal count = solution.getLiteral("?count");
39          System.out.println(count);
40      }
41  }finally {
42      qexecute.close();
43  }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 10:43:04 PM)  
log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).  
log4j:WARN Please initialize the log4j system properly.  
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  
5^http://www.w3.org/2001/XMLSchema#integer

Figure B.6: CQ5 in Jena

```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl:<http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db:<http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT ?name " +
28      "WHERE {" +
29      "    ?cust a db:CUSTOMER." +
30      "    ?cust db:CUSTOMER.NAME ?name.}";
31
32  Query query = QueryFactory.create(queryString);
33  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34  try{
35      ResultSet results = qexecute.execSelect();
36      while (results.hasNext()) {
37          QuerySolution solution = results.nextSolution();
38          Literal name = solution.getLiteral("?name");
39          System.out.println(name);
40      }
41  }finally {
42      qexecute.close();
43  }

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 10:49:29 PM)  
log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).  
log4j:WARN Please initialize the log4j system properly.  
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  
Sonya De Wet^http://www.w3.org/2001/XMLSchema#string  
Malome Gadiki Phasha^http://www.w3.org/2001/XMLSchema#string  
Konyana Badimo^http://www.w3.org/2001/XMLSchema#string  
Salome Mariam Peters^http://www.w3.org/2001/XMLSchema#string  
Mpho Peter Mankwana^http://www.w3.org/2001/XMLSchema#string

Figure B.7: CQ6 in Jena



```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl: <http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db: <http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT ?Description " +
28      "WHERE {" +
29      "    ?ctype a db:CUSTOMERTYPE." +
30      "    ?ctype db:CUSTOMERTYPE.DESRIPTION ?Description.}";
31
32  Query query = QueryFactory.create(queryString);
33  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34  try{
35      ResultSet results = qexecute.execSelect();
36      while (results.hasNext()) {
37          QuerySolution solution = results.nextSolution();
38          Literal description = solution.getLiteral("?Description");
39          System.out.println(description);
40      }
41  }finally {
42      qexecute.close();
43  }

```

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 10:56:30 PM)  
 log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).  
 log4j:WARN Please initialize the log4j system properly.  
 log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  
 Tenant^^http://www.w3.org/2001/XMLSchema#string  
 Owner Over 60^^http://www.w3.org/2001/XMLSchema#string  
 General Owner^^http://www.w3.org/2001/XMLSchema#string  
 Indigency Owner^^http://www.w3.org/2001/XMLSchema#string

Figure B.8: CQ7 in Jena

```

19  FileManager.get().addLocatorClassLoader(Main.class.getClassLoader());
20  Model model = FileManager.get().loadModel("c:/OWLFile/Municipality_DataMaster.owl");
21  String queryString =
22      "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23      "PREFIX owl: <http://www.w3.org/2002/07/owl#> " +
24      "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> " +
25      "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> " +
26      "PREFIX db: <http://biostorm.stanford.edu/db_table_classes/DSN_jdbc.oracle.thin.@localhost.1521.xe#> " +
27      "SELECT ?details " +
28      "WHERE {" +
29      "    ?query a db:QUERY." +
30      "    ?query db:QUERY.DETAILS ?details.}";
31
32  Query query = QueryFactory.create(queryString);
33  QueryExecution qexecute = QueryExecutionFactory.create(query, model);
34  try{
35      ResultSet results = qexecute.execSelect();
36      while (results.hasNext()) {
37          QuerySolution solution = results.nextSolution();
38          Literal details = solution.getLiteral("?details");
39          System.out.println(details);
40      }
41  }finally {
42      qexecute.close();
43  }

```

<terminated> Main [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20 Oct 2014, 11:03:33 PM)  
 log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).  
 log4j:WARN Please initialize the log4j system properly.  
 log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  
 There has been no refuse collection for two weeks^^http://www.w3.org/2001/XMLSchema#string  
 Electricity has been on and off for the past two weeks^^http://www.w3.org/2001/XMLSchema#string  
 I didn't receive my monthly statement!^^http://www.w3.org/2001/XMLSchema#string

Figure B.9: CQ8 in Jena