CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

OWL2 benchmarking for the evaluation of the knowledge based systemś platforms

by

Sher Afgun Khan

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Computing
Department of Computer Science

July 2018

OWL2 benchmarking for the evaluation of the knowledge based systemś platforms

By
Sher Afgun Khan
PC093009

Foreign Evaluator 1
Dr. Jose Valente De Oliveria
Department of Computer Science, Universidade do Algarve,
Portugal

Foreign Evaluator 2
Dr. Mohamed Amin Embi
Universiti Kebangsaan, Malaysia (National University of Malaysia)

Supervisor Name
Dr. Muhammad Abdul Qadir

Dr. Nayyer Masood
Head, Department of Computer Science

Dr. Muhammad Abdul Qadir
Dean, Faculty of Computing

DEPARTMENT OF COMPUTER SCIENCE
CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY
ISLAMABAD
2018

CAPITAL UNIVERSITY OF SCIENCE & TECHNOLOGY ISLAMABAD

Expressway, Kahuta Road, Zone-V, Islamabad
Phone:+92-51-111-555-666 Fax: +92-51-4486705
Email: info@cust.edu.pk Website: https://www.cust.edu.pk

CERTIFICATE OF APPROVAL

This is to certify that the research work presented in the thesis, entitled "**OWL2 Benchmarking for the Evaluation of the Knowledge Based Systems' Platforms**" was conducted under the supervision of **Dr. Muhammad Abdul Qadir**. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the **Department of Computer Science, Capital University of Science and Technology** in partial fulfillment of the requirements for the degree of Doctor in Philosophy in the field of **Computer Science.** The open defence of the thesis was conducted on **06 July, 2018.**

Student Name : Mr. Sher Afgun Khan (PC093009)

The Examination Committee unanimously agrees to award PhD degree in the mentioned field.

Examination Committee :

(a) External Examiner 1: Dr. Javed Ferzund, Associate Professor COMSATS, Sahiwal Campus

(b) External Examiner 2: Dr. Amanullah Yasin, Assistant Professor CASE, Islamabad

(c) Internal Examiner : Dr. Nayyer Masood, Professor, CUST, Islamabad

Supervisor Name : Dr. Muhammad Abdul Qadir, Professor, CUST, Islamabad

Name of HoD : Dr. Nayyer Masood, Professor, CUST, Islamabad

Name of Dean : Dr. Muhammad Abdul Qadir, Professor, CUST, Islamabad

AUTHOR'S DECLARATION

I, **Mr. Sher Afgun Khan (Registration No. PC093009)**, hereby state that my PhD thesis titled, '**OWL2 Benchmarking for the Evaluation of the Knowledge Based Systems' Platforms**' is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/ world.

At any time, if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my PhD Degree.

(**Mr. Sher Afgun Khan**)

Dated: July, 2018 Registration No : PC093009

PLAGIARISM UNDERTAKING

I solemnly declare that research work presented in the thesis titled "**OWL2 Benchmarking for the Evaluation of the Knowledge Based Systems' Platforms**" is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/ cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of PhD Degree, the University reserves the right to withdraw/ revoke my PhD degree and that HEC and the University have the right to publish my name on the HEC/ University Website on which names of students are placed who submitted plagiarized thesis.

(Mr. Sher Afgun Khan)
Registration No. PC093009

Dated: July, 2018

Acknowledgements

In the name of Allah, the Most Gracious, and the Most Merciful Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis. This thesis has been kept on track and been seen through to completion with the support and encouragement of numerous people including my well-wishers, my friends, colleagues and various institutions. At the end of my thesis, I would like to thank all those people who made this thesis possible and an unforgettable experience for me. To express my thanks to all those who contributed in many ways to the success of this study and made it an unforgettable experience for me. First of all, I pay my gratitude to my supervisor,*Prof. Dr. Muhammad Abdul Qadir* for his continued guidance, commitment, support, and encouragement on pursuing my thesis on a priority basis. In fact, I choose Ontology Engineering as a research area because of his lectures during my PhD program. Due to his guidance, I got the right direction in the ongoing research work in pursuit of a Ph.D. degree in computer science. Despite his busy schedule, he used to review my thesis progress, give his valuable suggestions and made my corrections. His unflinching courage and conviction always inspired me. I also want to pay my gratitude to Dr. Muhammad Tanvir Afzal for encouragement,guidance, and write-ups to accomplish my research work. I would also like to thanks to Higher Education Commission, Islamabad, Pakistan, for providing financial assistance in the form of Fellowship which buttressed me to perform my work comfortably. I am really grateful to Dr. Muhammad Azeem Abbas of CDSC research group of this university, who's contributions in my research work are considerably impactful. I am also grateful to Dr. Abdul Shahid (Ex-CUST fellow), who played an important role in keeping me live and motivated in the studies. Now, I think of my parents whose selfless sacrificial life and their great efforts with pain and tears and unceasing prayers have enabled me to reach the present position in life. Finally yet importantly, I want to express my deepest gratitude to my wife, sons (Muhammad Ibrahim, Muhammad Baqir), and daughter(Zainab Muhammad) for their love, understanding and continuous support. Finally, I thank all those who have helped me directly or indirectly in the successful completion of my thesis.

Sher Afgun Khan

PC093009

List of Publications

Journal Papers

1. Khan, S. A., Qadir, M. A., Abbas, M. A., Afzal, M. T. (2017). OWL2 benchmarking for the evaluation of knowledge based systems. PloS one, 12(6), e0179578.

Conference Papers

1. Khan, S. A., Qadir, M. A. (2010, June). A genaric framework for evaluation of ontology to relational database transformation process. In Information and emerging technologies (ICIET), 2010 international conference on. IEEE, pp. 1-5.

Abstract

Data modelling using OWL semantics for the development of a knowledge based system (KBS) has recently attracted the attention of many applications in various domains like Business, Biosciences, Health, and Digital Libraries etc. Well-known knowledge base systems platforms (KBSPs) are used for storing and querying the ontology-based applications using different storage formats (memory, graph, file, and database). Choosing an appropriate KBSP is considered as an important task to help domain experts to select suitable KBSP. In this research, a problem with the current state of the art evaluation benchmarks has been identified; the existing state of the art evaluation benchmarks are not designed to support complete OWL semantics (OWL1.1 and OWL2). The objectives of the research include; inspection of the existing evaluation benchmarks for the missing OWL semantics, construction of benchmark with complete OWL covergae, analysis of the proposed benchmark and evaluation of the KBSPs using proposed benchmark. In this research, the proposed OWL2 benchmark (OEB2) for the evaluation of the KBSPs is constructed using the foundational building blocks of the evaluation benchmark: data schema, dataset, and queryset with performance evaluation matric. The proposed work uses university ontology as case study in the construction of OEB2. The complete OWL semantics are added in the data schema of the proposed benchmark through survey of relevant ontologies, usage of WordNet senses, and addition of property characteristics through patterned queries. The dataset of the proposed benchmark is enriched with all the assertion level OWL semantics and coverage of all the property characterisitics. The coverage of OWL semantics in the queries set are covered by classifying the queries and also making them more generic. Finally, the proposed benchmark has been tested on the memory based, file based, graph based, and relational database KBSPs for the performance and scalability measures. The results show that OEB2 is able to evaluate the behaviour of different KBSPs with complete OWL semantics (OWL1.1 and OWL2). The reported results provides an evidence that different knowledge base system are suitable for different domains. The present work assist domain experts to choose a relevant knowledge base system based on the nature of their domain. This research also concludes with multiple directions for future research in this domain.

Contents

List of Figures

List of Tables

Abbreviations

OWL	Web Ontology Language
OWL2	Web ontology Language Version 2
EnR	Edge to Node Ratio
NoE	Number of Edges
RDF	Resource Description Framework
KBSP	Knowledge Based System Platform
KBS	Knowledge Based System
TBox	Terminological Axiom
ABox	Assertion Axiom
LUBM	Lehigh University Benchmark
UOBM	University Ontology Benchmark
BSBM	Berlin Sparql Benchmark
LBBM	Lehigh Bibtex Benchmark
SQ	Simple Queries
CQ	Complex Queries
OPQ	Object Properties Characteristics Pattern Queries
MIS	Management Information System
API	Application Program Interface

Chapter 1

Introduction

1.1 Overview

In recent years ontologies have emerged as a significant data model to represent domain knowledge for various scientific domains or applications. Domain ontology and its instances can be referred as a knowledge-based system (KBS). A system used to store and query the ontology and instances is termed as a knowledge-based systems platform (KBSP). Some examples of KBSPs are Sesame [1], Blazegraph [2], Jena SDB [3], and OntRel [4]. Well known storage formats used by these platforms include memory-based, file-based, graph-based and database.

Every domain carries its own complexity based on its schema, number of axioms, the semantics used, the nature of data and types of expected queries. For this reason domain users are keen to select an appropriate knowledge based platform which suits their applications requirements. For instance, a memory based KBSP might be a suitable choice to represent the Agriculture ontology [5]. In contrast, Gene ontology might require a persistent storage model because of its high schematic complexity that exists in the form of multiple parents [6]. To choose an appropriate KBSP for domain specific requirements, there is a need of a suitable evaluation benchmark for KBSP.

In literature, benchmark evaluation is basically testing the strength of a benchmark to rigorously test the KBSP. Strength of a benchmark can be defined based upon

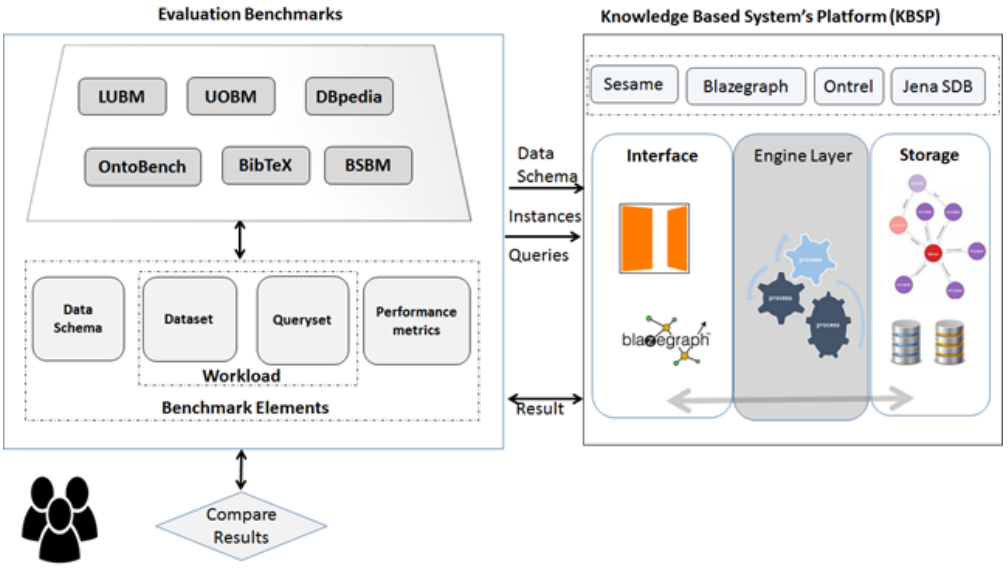

FIGURE 1.1: Evaluation Benchmark and the KBSP.

the need of the applications which are going to use these KBSPs. In the context of knowledge based systems platform, the expectations are going to be at-least followings:

- The KBSP must preserve the semantics of the knowledge (ontology and its instances) being stored in it so that all the domain level queries can be answered.
- The KBSP must be scalable to store the ever growing knowledge and answer queries.
- The load time for a knowledge based system must be minimum possible.
- The query response time must be minimum possible.

The interaction between an evaluation benchmark and a knowledge based systems platform is shown in Figure 1.1. A benchmark is going to evaluate different KBSPs, therefore the values against four points criteria to evaluate KBSP will be measured as relative for different systems at a time.

To explain Figure 1.1, first an appropriate benchmark is selected to evaluate the KBS platform. The KBSP benchmark which contains a knowledge schema (ontology), its instances (data) and a set of queries (query) will be loaded to the KBSP.

The results from KBSP along with other parameters (load time and response time) will be evaluated to judge the overall strength of the platform to store knowledge based systems application and to query it. This would help the vendors and users to choose an appropriate evaluated platform for their applications.

There are three major components of a KBSP evaluation benchmark. These include; knowledge schema, knowledge instances (Data Set) and a set of queries with performance metrics.

1.2 Problem Description

The nature of the present research exhibits problems, which are identified and addressed in the present research include:

1.2.1 Expressivity issue in the knowledge schema of evaluation benchmarks

Web Ontology Language OWL was developed with the emergence of sematic web to fulfil the need of developing ontology tools including editors and applications. Therefore, the scientific community like Biopex, NASA, started using ontologies for representing their domain knowledge and data exchange services [7]. The community realized that the language lacks in modelling complex domains due to expressivity issues with the web ontology language OWL [7]. The problem of lack of expressiveness in OWL language is addressed by extending the web ontology language i.e. OWL2. Knowledge or Data schema is an important component of evaluation benchmarks. Existing benchmarks to evaluate the performance of KBSPs are tailored on the expressiveness of OWL or OWL1.1. Their data schemas are based on the OWL language constructs. The evaluation benchmarks include; Lehigh University Benchmark (LUBM)[8] , University Ontology Benchmark (UOBM) [9], Berlin SPARQL Benchmark (BSBM) [10], Dbpedia benchmark [11], SP2B [12],OntoBench [13]. The data schemas of the benchmark lacks in covering all the expressivity of OWL language. The data schemas also lacks in covering

the extended expressivity offered in OWL2 language. Due to lack of coverage of OWL semantics in the data schema, the performance of the KBSPs cannot be accurately measured.

1.2.2 Expressivity issue in the knowledge instances (dataset) of evaluation benchmarks

Knowledge instances or Dataset is the important part of evaluation benchmark building block. As data schemas of the evaluation benchmarks does not provide the coverage of OWL and OWL2 semantics. Accordingly, the datasets provided by the evaluation benchmarks does not provide coverage of OWL and OWL2 language constructs. Even the usage of OWL constructs in the benchmark datasets is less than that of their schemas. This scenario may leads to unsound performance of the KBSPs against the varied patterns of instances using extended OWL constructs.

1.2.3 Expressivity issue in the queryset of evaluation benchmarks

In order to check the performance of KBSPs, the evaluation benchmarks uses different querysets. These querysets are tailored for high selectivity, input size, output size, inference but not sufficient to capture the language semantics (i.e. language constructs) embodied in the knowledge schemas [8],[9],[10],[11],[12],[13].One of the reason is the missing of generic queries that emphasis on the language constructs. As a result it becomes challenging if not impossible to check the behavior of KBSPs against different language constructs especially the OWL2 constructs. Like other building blocks, the existing evaluation benchmarks does not provide coverage of OWL2 semantics in their querysets.

The literature survey shows that current state of the art benchmarks do not cover all the constructs of OWL in their benchmarks components and hence do not provide complete coverage of OWL or OWL 1.1 semantics. In order to resolve the expressivity issues [7], OWL2 [14] has also been released and being used in

applications. The available benchmarks do not support OWL2 extensions in their components, too. Therefore, the usability of these benchmarks to evaluate the performance of KBSPs for OWL1.1 semantics is limited, and for OWL2 extensions, it is not available.

1.3 Statement of Problem

The research problem is summarized as follows:

- Data Schema, one of the major components of the existing KBSP benchmark lacks OWL2 constructs [8],[9],[10],[11],[12].
- Dataset, an important component of the existing KBSP benchmark lacks instances against OWL2 assertions [8],[9],[10],[11],[12],[13].
- Queryset, a third important component of the existing KBSP benchmark is not sufficient to check the semantic preservation of OWL2 [8],[9],[10],[11],[12].
- All the above three components (data schema, data set, query set) are missing in the existing benchmarks for the complete coverage of OWL1.1 semantics, too [8],[9],[10],[11],[12].

1.4 Research Questions

In order to solve the problem stated in section 1.2 we have to address the following research questions.

1. How can a KBSP benchmark be evaluated against the four qualities of a KBSP?
2. What are the missing OWL1.1 and OWL2 semantics in the data schema/ontology of the existing KBSP benchmarks?
3. What are the missing OWL1.1 and OWL2 constructs in the dataset and its generator in the existing KBSP benchmarks?

4. What are the missing OWL1.1 and OWL2 constructs in the query set of existing KBSP benchmark?

5. How to integrate the missing OWL1.1 and OWL2 semantics in all the components of proposed KBSP benchmark?

6. How proposed OWL2 KBSP benchmark shall be evaluated for the incorporation of the performance metric?

7. Does the proposed evaluation benchmark implemented on the theoretical foundations of the KBSP reflects the result against the required features of a KBSP (load time, query response time, scalability)?

1.5 Objectives

1. To inspect the existing evaluation benchmarks for missing OWL1.1 and OWL2 semantics using W3C standard list of OWL and OWL2 construct.

2. To propose an OWL2 evaluation benchmark covering all the constructs of OWL and OWL2 semantics.

3. To evaluate the proposed OWL2 KBSP benchmark.

4. To evaluate the Knowledge based platforms by using the proposed benchmark against the performance evaluation metric.

1.6 Research Scope

The incorporation of the missing OWL1.1 and OWL2 semantics in each component of the standard evaluation benchmark will lead to a proposed evaluation benchmark. Scope of the thesis is to propose a benchmark for the evaluation of OWL2 knowledge based systems platform against the performance evaluation metric. The proposed benchmark consist of all the standard components i.e. data schema, dataset, queryset and performance evaluation metric. The performance

is evaluated for load time, query response time, scalability and semantic coverage. Scalability of the platform will be evaluated with the increase in the dataset. Evaluation of the data schema, dataset and queryset is going to ensure the evaluation of the platform for complete semantic coverage (preservation). As a case study, the proposed benchmark will be implemented on well-known KBSPs. The research will provide a basis for further enhancement of the performance metric to check additional capabilities of the platform for inference and reasoning with automated tools.

1.7 Methodology

The methodology consists of analysis of the existing benchmarks for the evaluation of KBSP, finding gaps in the benchmark to support the evaluation of the latest KBSP to store OWL2 semantics, the construction of benchmark with all the components (data schema and workload, i.e. dataset generation process and queryset), and formulation of a performance evaluation metric.

The existing evaluation benchmarks will be checked and compared with the standard list of OWL1.1 and OWL2 semantics provided by OWL Working Group initiated by Semantic Web activity of World Wide Web consortium . A selection of data schema (ontology) is required as a case study. Keeping in view the concepts and properties of the selected data schema, the incorporation of OWL2 semantics is required which models the real world domain. Considering this factor, different ontologies are required to be surveyed for inclusion of OWL2 semantics. In order to demonstrate the working with a case study, the university ontology [9] will be used as a knowledge schema for the benchmark. Finally, evaluation of seven well known KBSP on three different sized datasets against the constructed queries will be performed to demonstrate their behavior against OWL2 semantics. The structure of the thesis is organized as follows. Chapter 2 describes the literature review and analysis of the existing benchmarks. Chapter 3 describes OEB2, the proposed evaluation benchmark and its components. Chapter 4 evaluates the proposed benchmark. Chapter 5 reports the results and discussion of the evaluation

of the KBSP using proposed benchmark. Chapter 6 describes the conclusion and future work.

1.8 Definitions

1.8.1 Knowledge Based System

The term has been frequently used in different sub-fields of computer science like AI, knowledge representation, software engineering, etc. In the context of different sub-fields, KBS employ different definitions. In this thesis, KBS is defined as a term comprises of three elements i.e. data schema with rich semantics, instances, and application, where data schema is the ontology to model a domain, instances are the assertions against the ontology and application is the usage of the ontology and its instances in that specific domain as shown in Figure 1.2. Querying ontology and its instances for explicit and implicit semantics is the primary application.

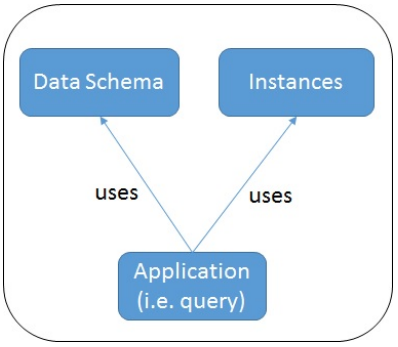

FIGURE 1.2: Elements of Knowledge Based System

1.8.2 Knowledge Based System Platform(KBSP)

It is software based system, which provides suitable tools over which domain specific knowledge based applications (i.e.KBS) can be constructed, hosted, and managed.

1.8.3 Ontology

Ontology is defined as a common and explicit specification of knowledge representation of a domain. More specifically, ontology comprises of structure represented in the form of graph consisting of hierarchy of concepts, and relations termed as predicates. The axioms to define constructs are classified as terminological axioms (TBox) and assertion axioms (ABox). TBox contains schema relevant information and ABox contains instances.

1.8.4 OWL

OWL OWL is a web ontology language which is used to represent the domain knowledge with the help of modelling constructs. OWL language became W3C recommendation in 2004. Its first version is known as OWL1.1. The current version of OWL language is OWL2, which provides more semantics and expressiveness to model the complex domains.

1.8.5 Benchmark

It is defined as an organized and continuous procedure to create, measure, compare and improve a given target system [9].

1.8.6 Semantic Coverage

Semantic Coverage The term is defined as the OWL2 coverage (means all the semantics of web ontology language) in the data schema, dataset and benchmark queries that can be checked in the KBSP.

1.8.7 Edge to Node Ratio (EnR)

Edge to Node ratio is defined as the complexity measure of the ontology structure by computing ratio between the number of edges and its respective number of

nodes (classes) [15], [16]. Formally, given a data schema i.e. ontology (O) containing number of edges (E) and number of nodes (N), the EnR is the ratio computed as |E|/|N |, where E is a set of edges and N is a set of nodes in the ontology and the symbol ||gives the size of the set.

1.8.8 Number of Edges (NoE)

The number of edges (NoE) is the number of occurrences of distinct ontology constructs (e) involving all the edges in the graph.

1.8.9 Load Time

Load time refers to the time required for loading a dataset into memory or persistence storage.

1.8.10 Response Time

The response time is the time required for issuing a query, obtaining and traversing the results sequentially.

Chapter 2

Literature Review

2.1 Overview

This chapter reviews and evaluates the benchmarks for knowledge based systems platform (KBSP) and then concludes towards the shortcomings in the available benchmarks. These shortcomings are termed as a research gap in the formulation of a benchmark.

2.2 The Benchmarking Activity

Benchmarking is an organized and continuous procedure of creating, measuring, comparing and improving a target system [17]. The procedure initiates with the construction of a benchmark. The activity normally results in the set of software tools with specifications to execute the benchmark on the target system to measure its performance. The findings or results related with the performance of the target systems would help the end users to select appropriate target system. These findings help the vendors to improvise their products, too.

2.3 Knowledge Based Systems Platform

Knowledge Based Systems Platform (KBSP) provides a facility of hosting and managing knowledge based applications (to develop, store and query ontology and its instances). The applications hold a domain knowledge, which is formally modelled by using ontology. Ontology is defined as a common and explicit specification of knowledge representation of a domain. More specifically, it comprises of structure represented in the form of a graph consisting of hierarchy of concepts, and relations between these concepts are termed as predicates. The axioms to define concepts are classified as terminological axioms (TBox) and assertion axioms (ABox). TBox contains schema relevant information to define concepts & properties and ABox contains instances against these concepts.

The common storage formats employed by KBSPs are memory based, file based, graph based and relational based. Relational storage formats is further divided into four types. These are vertical partitioned (uses two column relation for every property in the ontology document), triple store (relation with three columns subject, predicate and object), class decomposition (transform each class as a relation where each property is represented as column of that class), and metadata approach (uses fixed number of relations for ontology storage). Some well-known KBSP are Blazegraph , Sesame, Jena SDB and OntRel [18]. Blazegraph is an open source knowledge base system platform to store the ontology documents using the graph database [2]. The design of Blazegraph supports RDF, OWL, SPARQL 1.1 and graph database APIs. Sesame is also an open source KBSP for storing the OWL documents and querying by using SPARQL. It provides different storage formats. Three of the storage formats have been selected for evaluation. These include Native storage for file based format, Database for database format and in-memory for memory based storage format. Jena SDB is an open source component of the Jena framework [3]. Jena SDB stores OWL documents in relational databases and allows querying these documents through SPARQL query servers. The SPARQL query will be translated into SQL query by the platform automatically. OntRel KBSP is based on the metadata approach with relations for each OWL constructs. Instances of the ontology documents are stored in the OWL

constructs relations [18]. OWL2ToRDB [19] is based on the class decomposition approach for storage model, where classes have separate relations and properties are represented as columns.

2.4 KBSP Benchmarks

To evaluate a KBSP for its suitability to host and manage a knowledge based application, there are few evaluation benchmarks proposed. Among the well-known contributions, Lehigh University Benchmark (LUBM) is the most influencing KBSP evaluation benchmark, used to measure the capabilities of KBSPs [8]. It comprises of three major components; benchmark ontology, dataset generator, and benchmark queries. The benchmark ontology belongs to the university domain and built with OWL-Lite language constructs. The benchmark provides flexible ABox dataset generation tool to generate user specified datasets. To evaluate a selected KBSPs against different performance metrics LUBM uses fourteen queries. These benchmark queries cover the properties, input size, high selectivity, complexity, inference of hierarchy, and logical inference. The benchmark uses data loading, repository size, query response time, query completeness, query soundness as performance metrics.

Lehigh BibTeX Benchmark (LBBM) is proposed to overcome the difficulty of using insufficient real world data in LUBM [20]. For this purpose, Monte Carlo algorithm is used to generate the synthetic data on the relevant extracted properties from real domain documents. The Berlin SPARQL

Belin Sparql Benchmark (BSBM) is developed for comparing the performance between native RDF stores and systems featuring SPARQL-to-SQL rewriters [10]. The benchmark is built around an e-commerce use case. It performs evaluation among four RDF stores. The benchmark comprises of two components: data set generator and test driver. Dataset generator is responsible of generating large amount of data for the e-commerce structure. The test driver is responsible of executing the SPARQL queries on the systems. In order to make the benchmark

queries more realistic, the queries are developed around the use case scenarios of e-commerce [10]. The benchmark queries are influenced by search and navigational patterns of the customers for different products.

University Ontology Benchmark (UOBM) [9] is an extension to LUBM [8] in terms of testing of inference and scalability. UOBM covers most of the OWL Lite and OWL DL constructs in their data schema. Major contributions of the UOBM are generation of the benchmark ontology using OWL Lite and OWL DL inference, construction of a single connected RDF graph using property connects and evaluation of several popular KBSPs for inference and scalability.

OntoDBench [21] evaluates the scalability and query performance of the relational storage systems under different storage representations (vertical, horizontal and binary). The benchmark addresses the storage models and query rewriting modules to perform evaluation of real world characteristics in LUBM datasets.

Smart City RDF benchmark [22] is developed to check the performance of two RDF stores including Virtuoso and GraphDB. The purpose of the benchmark is to find the suitability of RDF store for integrating smart cities data, modelling and application. The data schema used by the benchmark has ALCIQ DL expressivity. The data schema comprises of 605 class counts, 629 SubClassOf axioms, 100 object properties, 08 disjoint classes, 23 inverse object properties and zero object property characteristics. The inspection of the benchmark shows the limited coverage of OWL1.1 and OWL2 constructs. Due to high number of SubClassOf relation the data schema exhibits hierarchal nature. Using of one object property characteristics alongwith limited use of OWL constructs makes the data schema simple. Instead of using synthetic dataset generator, the benchmark use three datasets expanding with temporal horizon i.e. 1 month, 2 months and 3 months of real time data [22]. The dataset is not available publically, but even then, it guarantees the missing of all the semantics which are not entailed by the benchmark data schema. The benchmark queries comprises of 25 queries with focus on finding the geo spatial data. This concludes that the benchmark lacks in providing coverage of OWL2 and most of the OWL1.1 semantics.

OntoBench [13] provides testing of the ontology tools in terms of ontology supported features and OWL2 semantics. This is done by providing user to select the elements from the ontology features like OWL Lite, OWL DL, OWL2 EL, OWL2 QL and OWL2 RL. Then the benchmark generates the ontology version based upon the user selected elements. The benchmark contributes in addressing the inflexibility and overhead issue of static benchmark ontology. The benchmark supports testing of TBox only (without testing ABox). Under the benchmark evaluation, OntoBench deals only with the data schema and testing of ontology visualization tools like WebVOWL.

The Norwegian Petroleum Directorate (NPD) benchmark evaluates the ontology based data access systems (OBDA) using the real world dataset from the oil industry [23]. The benchmark proposed by Butt et al. consists of Barton library dataset, six benchmark queries, test cases based upon CRUD operations and evaluation metrics such as resource utilization, success ratio and cumulative query performance [24]. The benchmark is tailored to evaluate the performance and scalability of the semantic web databases.

The Dbpedia benchmark [11] is a SPARQL benchmark, which consists of Dbpedia ontology, benchmark queries, SPARQL endpoint and RDF synthetic dataset generator. The benchmarks queries are a selected set of user queries posed over the Dbpedia Knowledge base. The benchmark uses the query-log mining, clustering and SPARQL feature analysis to compare the RDF triple stores.

In general, KBSPs evaluation benchmark consists of three elements as, 1) data schema, 2) workload (data set and query set) and performance metrics [8] , [9], [11], [25]. Data Schema (Knowledge Schema) describes the structure of concepts and the relations between the concepts of a domain knowledge (Ontology) being used in the application. Workload comprises of dataset generation process and a set of queries. Dataset generation process generates different size datasets/instances against the given data schema. The performance metrics is a method to describe the performance of any system in a quantitative manner. There are number of basic performance metrics being used in the benchmarking for evaluating

the performance of KBSP system. Some of the common metrics include response time, load time, throughput, and scalability.

Benchmark evaluation is an important task to check the appropriateness (efficiency and scalability) of KBSPs for applications of knowledge based systems for different domains. In order to evaluate a benchmark for KBSP the following steps are adopted:

1. Examine the data schemas for the complete OWL semantics with the standard OWL reference list [26], [27].

2. Evaluate the complexity of the data schemas being used in the benchmarks. The complexity of the existing benchmarks data schemas is determined by computing the edge to node ratio (EnR) of each OWL construct and checking the usage of Object property characteristics.

3. Inspect the datasets for a sufficient set of OWL semantics. For this purpose, the dataset generator program of the benchmark is manually inspected for the classes, object properties, and data properties used for generating the dataset. The distinct usage of OWL constructs in the classes, object, and data properties is collected and combined. This combined list is then compared with the standard OWL constructs list.

4. Inspect the benchmark queries for a sufficient set of OWL semantics. For example, the following query (as shown in Figure 2.1) use OWL construct rdf:type in the query. A class GraduateStudent is also used in the query. This class has three uses in the data schema as shown in the Figure 2.2. Similarly the uses of OWL constructs for the object property takesCourse are also noted. After combining all the distinct OWL constructs used directly in the queryset or indirectly obtained as explained above are compared with the OWL constructs standard list to find out the missing constructs in the benchmark queries.

```
SELECT ?X
WHERE
        {
        ?X rdf:type ub:GraduateStudent .   ?X ub:takesCourse
        http://www.Department0.University0.edu/GraduateCourse0
        }
```

FIGURE 2.1: SPARQL query showing the usage of OWL constructs

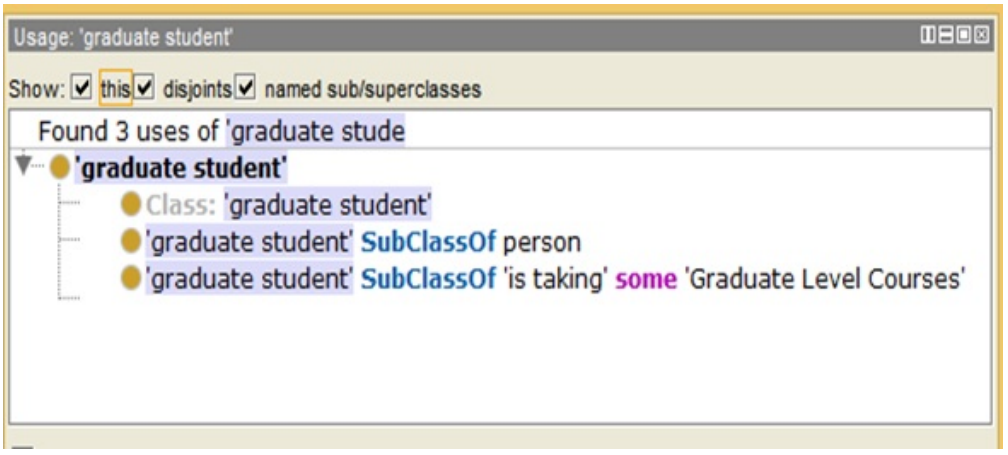

FIGURE 2.2: OWL constructs uses for the class GraduateStudent

Evaluation of the benchmarks against these four points is going to converge towards the gaps in these benchmarks, and will guide us towards the formulation of a new benchmark.

2.5 Evaluation of KBSP Benchmarks - LUBM

Inspection of data schema of LUBM shows that it does not provide complete coverage of OWL (OWL1.1 & OWL2) semantics. Some of the important OWL1.1 constructs missed in the data schema are ComplementOf, oneof, UnionOf, DisjointWith, hasSelf, irreflexive and assertion constructs [9]. OWL2 constructs are not used in the data schema. All the constructs missing in the data schema of LUBM are shown in the Appendix-B. To check the complexity of the data schema EnR is computed. Table 2.1 shows that for most of the OWL constructs EnR is zero and only subClassOf relation carries higher EnR i.e. 0.85. The combined EnR of user defined properties and OWL constructs is shown in Table 2.2. This table

shows that the overall EnR with OWL constructs is 3.06 and without OWL constructs, it is 0.58. The low EnR without OWL constructs shows that the minimum number of object properties is connected to the classes. In LUBM only 14 classes are directly related to the properties. This reflects the data schema is of simple nature. The inspection of the LUBM dataset shows that only 19 out of 43 classes, 13 out of 25 object properties, and 3 out of 7 data properties used in the dataset. The snippet of the dataset generator in Figure 2.3 shows the limited use of classes. LUBM uses only one object property i.e. transitive property in the data schema. This concludes the data schema is not of complex nature. The dataset files are generated as separate and isolated graphs without carrying schema information as shown in the Figure 2.4 (generated through WebVOWL 1.0.4). The Figure shows that the instances of fourteen (14) classes numbered in class circle reflect that data schema and dataset are separate in the LUBM. All the OWL constructs missing in the data schema are also missing in the dataset. The list of missing OWL1.1 & OWL2 constructs is shown in the Appendix B.

```
case CS_C_UNIV:
  break;
case CS_C_DEPT:
  break;
case CS_C_FULLPROF:
  instances_[i].num = _getRandomFromRange(FULLPROF_MIN, FULLPROF_MAX);
  break;
case CS_C_ASSOPROF:
  instances_[i].num = _getRandomFromRange(ASSOPROF_MIN, ASSOPROF_MAX);
  break;
case CS_C_ASSTPROF:
  instances_[i].num = _getRandomFromRange(ASSTPROF_MIN, ASSTPROF_MAX);
  break;
case CS_C_LECTURER:
  instances_[i].num = _getRandomFromRange(LEC_MIN, LEC_MAX);
  break;
case CS_C_UNDERSTUD:
  instances_[i].num = _getRandomFromRange(R_UNDERSTUD_FACULTY_MIN *
                                          instances_[CS_C_FACULTY].total,
                                          R_UNDERSTUD_FACULTY_MAX *
                                          instances_[CS_C_FACULTY].total);
```

FIGURE 2.3: LUBM dataset generator code snippet

LUBM provides fourteen queries for checking the performance of KBSP. The investigation of OWL semantics in the benchmark queries is manually performed against the classes, object properties and data properties used in each query. As

FIGURE 2.4: LUBM dataset file visualization using WebVOWL

mentioned previously that the schema lacks some of the important semantics of OWL1.1 and all the semantics of OWL2, we cannot expect these semantics in the query set. The finding shows that except transitive property, all the other object property characteristics are missing in the benchmark queries like functional, inverse functional, symmetric, asymmetric, reflexive and irreflexive properties. As LUBM lacks OWL1.1 and OWL2 coverage in data schema, dataset and queryset, which conclude that all the missing semantics or constructs cannot checked or verified through LUBM benchmark.

2.6 Evaluation of the KBSP Benchmark-UOBM

The benchmark extends the LUBM by adding the missing OWL1.1 constructs in the data schema. To provide OWL constructs coverage, two separate data schema files are provided for OWL-Lite and OWL-DL languages However, the data schema misses important OWL1.1 constructs including oneOf, Nothing, sameAs, Asymmetric, reflexive, irreflexive object properties, qualified cardinality restrictions for object and data properties, data property axioms (i.e. equivalent, functional properties) and assertion axioms (Appendix B). The benchmark does not provide OWL2 coverage in the data schema.

The EnR column of Table 2.1 for OWL-DL and OWL-Lite data schema shows that both data schemas have higher (i.e. 0.75 for OWL-DL and 1.36 for OWL-Lite) EnR in subClassOf relation. While, EnR of other OWL constructs is very minimal. If OWL constructs are removed from the data schemas, the EnR for the data schema become very low i.e. 0.309, 0.487 for both ontologies of the UOBM. These values are even lower than the value of LUBM data schema. The overall EnR of OWL-DL and OWL-Lite is 2.567 and 3.292 respectively as shown in Table 2.2. This indicates that the small sized clusters of instances and classes are sparsely connected. The analysis reveals the fact that 97 classes (nodes) in OWL-DL and 89 classes of OWL-Lite are not connected with any object properties. UOBM uses only four object property (out of nine) characteristics: functional, inverse functional, transitive and symmetric property in the data schema. This shows the lack of semantics in the benchmark data schema. This conclude that the data schemas of UOBM is of simple structure. The manual inspection of the dataset generator shows that only 28.32

There are 16 benchmark queries in the UOBM. Out of these three queries are written for DL. Like the LUBM, the benchmark queries are used to check the performance of the KBSP. The inspection of the benchmark queries for OWL semantics shows better coverage of OWL constructs as compare to LUBM queries. These OWL constructs include Disjointwith, symmetric, transitive, inverseOf, and functional properties. However, the benchmark queries misses out the use of OWL constructs like equivalent class, equivalent property, SameAs, differentFrom, alldifferentfrom, inverseFunctional, etc. Like LUBM, the benchmark queries does not provide coverage of OWL2 constructs [28]. The behavior of the benchmark queries of UOBM is similar to that of LUBM queries except in the cases where some additional constructs of OWL have been added into the benchmark queries set as shown in the Appendix B. As the queries are not sufficient to provide complete coverage of OWL semantics so the benchmark does not provide semantic preservation.

2.7 Evaluation of KBSP Benchmarks Dbpedia Benchmark (DPSBM)

The data schema of DPSBM [29] provides the OWL semantics coverage of subClassOf, EquivalentClass, DisjointClass, subObjectProperty, subDataProperty, functionalDataProperty, annotation Assertions, EquivalentObjectProperty and Domain, Range for both object and data properties. The remaining OWL1.1 constructs and all OWL2 constructs are missing in the data schema.

The EnR of DPSBM is relatively high for the subClassOf, Domain, Range, and subProperty. There are two main reasons for the EnR of the above mentioned OWL constructs. One is the large number of occurrences of the OWL constructs as shown in the Table 2.1. The other reason is the usage of large number of classes, object and, data properties in the data schema. Due to these factors, the overall EnR with and without OWL constructs is high (i.e. 7.18 & 1.14 respectively). However, from the structural complexity point of view, it is still lesser than Vehicle ontology shown in Table 2.2. The benchmark adopts two approaches to generate datasets from the main Dbpedia repository. Due to the non-availability of the dataset generator, the inspection of the dataset provided at the benchmark website is inspected. Which shows only instances without the schema information. The dataset misses OWL assertion constructs.

There are twenty six queries used in the benchmark. Checking all the queries shows that the focus of the queries is to ensure the coverage of most of the SPARQL language constructs. Although the focus of OWL semantics is limited to the available semantics of the data schema like rdf: type, subClassOf, equivelentProperty, rdfs:domain, rdfs:range, etc. However, the remaining OWL1.1 and OWL2 constructs are not covered in the benchmark queries.

2.8 Evaluation of KBSP Benchmarks NPD

Inspecting the data schema of NPD benchmark [23], shows 343 classes, 142 object properties, and 238 data properties. Although the data schema is based on OWL2 QL fragment [23], but it lacks in covering OWL1.1 and OWL2 constructs including IntersectionOf, ComplementOf, unionOf, equivalent class, etc. Except symmetric object property, all the remaining object properties are missing in the data schema.

EnR of NPD data schema is high for the subClassOf (i.e. 2.51), Domain (i.e. 0.66), Range (0.9) and DisjointWith (i.e. 0.14). For most of the OWL constructs, EnR is either zero or less than 0.1. This shows that schema is of simple nature with impression of hierarchal nature. Due to the high usage of subClassOf relation, the overall EnR with and without OWL constructs is high (i.e. 7.41 & 0.41 in Table 2.2) but lesser than Vehicle ontology. For dataset, no dataset generation is used rather than it is selected from the FactPages [30]. The benchmark ontology is mapped on the FactPages [30] and stored in the database. To increase the size of dataset, the Virtual Instance Generator [VIG] has been introduced, which is implemented using mappings of RDF statements to relational queries. As RDF graphs are virtual and generated from relational database, so this has no relevance with the standard dataset generators [8] , [9], [11], [25] used in the KBSPs. Inspecting the relational database and mapping files of the benchmark does not capture the OWL constructs. The queryset comprises of 25 queries, which are based on the interviews of the users of NPD dataset. These queries have been categorized in different level of complexity. These queries include maximum number of SubClassOf for a concept, application of filters, using of aggregating functions, etc. By checking the classes, object properties, and data properties used directly or indirectly in the benchmark queries, the findings shows that except subClassOf, disjointWith and few other OWL constructs, all the remaining OWL1.1 and OWL2 semantics are missing in the queries.

2.9 Evaluation of KBSP Benchmarks OntoBench

OntoBench [13] generates the data schemas with already defined structure along with the options to select OWL1.1 and OWL2 elements. The benchmark provides coverage of all the OWL1.1 and OWL2 semantics in the data schema.

The EnR of DPSBM is relatively high for the subClassOf, Domain, Range, and subProperty. There are two main reasons for the EnR of the above mentioned OWL constructs. One is the large number of occurrences of the OWL constructs as shown in the Table 2.1. The other reason is the usage of large number of classes, object and, data properties in the data schema. Due to these factors, the overall EnR with and without OWL constructs is high (i.e. 7.18 & 1.14 respectively). However, from the structural complexity point of view, it is still lesser than Vehicle ontology shown in Table 2.2.

The benchmark does not provide the dataset generator for generating the instances against the given data schema. So there is no coverage of OWL2 semantics for dataset element.

As the benchmark is not tailored for the evaluation of the KBSP, so no benchmark queryset is provided with the OntoBench.

2.10 Evaluation of Vehicle ontology for the suitability of benchmark

The vehicle data schema is used as a case study to demonstrate the semantic preservation of OWL2 ontologies in the relational databases [19]. The data schema provides OWL2 coverage.

Due to this coverage, each OWL2 construct has an EnR. Although, EnR of most of the OWL2 constructs ranges from 0.1 to 0.4. These constructs include Symmetric, Transitive, InverseFunctional, etc. The maximum EnR is 0.6 shown for dataPropertyDomain. The overall EnR of the data schema (with and without OWL2 constructs) is 11.84 and 1.473 respectively. These values are higher than

TABLE 2.1: Major OWL constructs usage in the benchmark and non-benchmark ontology

OWL Constructs	LUBM		UOBM				Vehicle		OntoBench		DPSBM	
			DL		Lite							
	NoE	EnR	NoE	EnR	NoE	EnR	NoE	EnR	NoE	EnR	NoE	EnR
Subclassof	36	0.84	85	0.75	154	1.36	52	2.74	23	0.24	744	0.64
Domain	21	0.49	27	0.24	27	0.24	17	0.89	39	0.41	920	0.79
Range	18	0.42	25	0.22	43	0.38	18	0.95	40	0.42	1015	0.87
Subproperty	5	0.12	9	0.08	29	0.26	1	0.05	1	0.01	933	0.8
Equivalent class	6	0.14	22	0.19	0	0	2	0.11	6	0.06	404	0.35
IntersectionOf	6	0.14	20	0.18	20	0.18	0	0	2	0.02	0	0
SomeValuesFrom	8	0.19	36	0.32	22	0.19	3	0.16	11	0.12	0	0
Allvaluesfrom	0	0	6	0.05	2	0.02	5	0.26	5	0.05	0	0
ComplementOf	0	0	4	0.04	0	0	0	0	1	0.01	0	0
Unionof	0	0	1	0.01	0	0	8	0.42	3	0.03	0	0
Inverseof	2	0.05	4	0.04	5	0.04	9	0.47	1	0.01	0	0
Disjointwith	0	0	1	0.01	0	0	13	0.68	2	0.02	24	0.02
Equivalent Prop.	0	0	1	0.01	1	0.01	1	0.05	6	0.06	3	0
Functional Prop.	0	0	1	0.01	1	0.01	12	0.63	1	0.01	0	0
Inver. Func. Prop.	0	0	1	0.01	1	0.01	3	0.16	1	0.01	0	0
Transitive Prop.	1	0.02	2	0.02	2	0.02	2	0.11	1	0.01	0	0
Symmetric Prop.	0	0	2	0.02	2	0.02	1	0.05	1	0.01	0	0
Data Prop.domain	4	0.09	6	0.05	6	0.05	24	1.26	57	0.6	1464	1.76

the EnRs of the surveyed benchmark data schemas. This concludes that the structure of the vehicle ontology [19] is more complex and uses more OWL constructs.

There is no separate dataset used by the approach [19]. Only 43 individuals are added with the vehicle data schema. However, all the assertions related OWL2 constructs including SameIndividual, DifferentIndividual, ObjectProperty assertions are used. However, these assertions lead towards inconsistency in the data schema as shown in Figure 2.4.

There are thirteen queries described by the OWL2ToRDB approach [19]. Five of the queries are specific to the OWL constructs, which include rdf:type, subClassOf, rdfs:comment, rdfs:label. The OWL coverage (direct and indirect) in the queries is high and include equivalent property, disjointWith, Domain, Range, someValuesFrom, rdfs:only, dataTypePropertyDomain, dataTypePropertyRange, Functional, subProperty, PropertyChain, inverseOf, Symmetric, Transitive, etc. Yet, all of the queries are of simple nature.

TABLE 2.2: Edges, Object Properties, and EnR of the benchmark and non-benchmark ontology

Ontology	Total No. of Classes(N)	Total no. of OWL constructs drawn at Edges (E)	Total No. of user defined object properties (OP)	Overall EnR with OWL constructs (E + OP) / N	Overall EnR without OWL constructs (OP / N)
LUBM	43	107	25	3.06	0.580
OWL-Lite	113	255	35	2.567	0.309
OWL-DL	113	317	55	3.292	0.487
Vehicle	19	197	28	11.842	1.473
BSBM	8	0	10	1.25	1.25
BSBM4	11	10	7	1.55	0.64
NPD	343	2403	142	7.41	0.41
DPSBM	1165	7038	1334	7.18	1.14

2.11 Research Gap

1. The existing evaluation benchmarks does not provide complete coverage of OWL1.1 semantics in their data schemas and workloads (i.e. dataset and queryset).

2. The computed complexity of the data schemas (based upon the EnR of OWL constructs and usage of object property characteristics) shows that the benchmark data schemas are of simple nature and thus insufficient to be used to evaluate the performance of KBSPs.

3. The existing evaluation benchmarks does not provide coverage of OWL2 semantics in the benchmarks data schema, dataset, and queryset.

Analysis in Table 2.3, shows that most of the benchmarks [10],[31] compared the performance of RDF stores and focuses on the SPARQL benchmarking. Whilst, the benchmarks [8], [9], [23] have a commonality of ontological benchmark features. The benchmark [13] is about ontologies generator with OWL2 coverage but having no scope of KBSP evaluation on OWL2 semantics, which is the scope of our work. Thus it can be concluded that all of the performance evaluation benchmarks lacks support for OWL2 semantics.

Table 2.3: Evaluation benchmarks and their KBSPs

Evaluation benchmark	Ontology expressiveness	Evaluated ontology storage systems(KBSPs)	OWL constructs	
			Benchmark ontology	Benchmark Dataset
LUBM [1]	SROIN(D), OWL Lite	File based, Memory based, RDMS, RDF store	Limited use of OWL constructs	Limited use of OWL constructs
BSBM [15]	RDFS,AL	RDBMS, RDF store	-	-
UOBM [2]	SHIN(D), OWL Lite, OWL DL	File based, Memory based, RDBMS, RDF store	OWL Lite and OWL-DL complete No support for OWL2	Limited use of OWL constructs
OntoDBench	SROIN(D), OWL Lite	RDBMS with three database representation	Limited use of OWL constructs No support for OWL2	Limited use of OWL constructs
Dbpedia [4]	ALCHF(D)	RDF store	Limited use of RDFS /OWL constructs No support for OWL2	-
Butt,2014 [18]	RDFS	RDF store	-	-
OntoBench [6]	SROIQ(D)	Ontology visualization tools	Support OWL & OWL2 constructs	Lack of support for ABox, No dataset is provided
RdfStore Benchmarking	RDFS	RDF store	-	-

2.12 Summary of the Chapter

This chapter described the review of relevant state of the art. It starts with the broader domains namely the benchmarking and KBSP. Later on, it provides a detailed literature review of the evaluation benchmarks using the defined evaluation steps and conclude the research gap.

Chapter 3

OWL2 EVALUATION BENCHMARK -OEB2

3.1 Overview

This chapter presents a proposed evaluation benchmark named as OWL2 evaluation benchmark (OEB2). The proposed benchmark follows the university ontology as a case study to enrich the elements of the benchmarks (i.e. data schema, dataset, queryset) with OWL1.1 and OWL2 coverage with the incorporation of the features to satisfy the four conditions for the evaluation of the benchmark as mentioned in chapter 2.

3.2 Data Schema

As a first step the OEB2 enriches the data schema with all the OWL constructs. In this regard, the present work have adopted the following methodology:

- Survey of Domain Ontologies: A survey of the available domain ontologies like Dbpedia [11] which has multi-domain scope with large instances, Vehicle [19] ontology for full coverage of OWL2 constructs and People ontology for common classes and properties which are used in our schema, is performed

to prepare the list of OWL2 constructs for the inclusion in the schema, as shown in Table 3.1.

- Use of WordNet Senses: WordNet senses [32] are used to add new classes and properties in the data schema. For instance, using hypernym sense 1 has_as_a_graduate object property is obtained against _Has_as_an_Alumnus object property under SameAs construct. Similarly, Association class is obtained for the already existed Institute class.

- Inclusion of Pattern Queries on existing ontologies in order to discover the object property characteristics: In order to add object property characteristics like symmetric, irreflexive, reflexive, transitive, asymmetric, inverse of, and disjoint in data schema domain experts are required. For this purpose we adopted a different approach. Object properties pattern queries are formulated and executed on the university ontology with more than 2.3 million triples data set and the results set were studied for the properties to be included in OEB2 data schema. For example, the following query in Figure 3.1 was submitted to obtain irreflexive object property characteristics. It returns three object properties hasSameHomeTownWith, isFriendOf and subOrganizationOf, which are then added in the data schema.

With the above three activities, the OEB2 data schema (ontology) contains all the semantics of OWL2. The benchmark ontology contains 132 concepts, 61 object properties, and 11 data properties. The detail is available in Appendix-D.

3.3 Dataset Generator

A number of axioms are supported by OWL2 to describe assertions (i.e. facts about the individuals). These assertions include class, object property and data property assertions. The dataset generator of OEB2 incorporates all of these OWL2 assertions in its dataset as shown in Table 3.2. The OEB2 dataset generator adopts a dynamic approach as compared to the static approach followed by UOBM.In dataset generation process, dynamic approach refers selection of classes

TABLE 3.1: OWL2 constructs in proposed data schema

OWL2 constructs	**Data schema axioms**
All Disjoint Classes	AllDisjointClasses(:ConferencePaper :JournalArticle :TechnicalReport)
Disjoint Union	disjointunion(:ConferencePaper :JournalArticle :TechnicalReport)
Property Chain	SubObjectPropertyOf(ObjectPropertyChain (:subOrganizationOf :subOrganizationOf)
Self-Restriction	:Person ObjectHasSelf(:belivesIn)
Reflexivity property	ReflexiveObjectProperty (:likes)
Irreflexivity property	IrreflexiveObjectProperty(:fatherOf)
Asymmetry property	AsymmetricObjectProperty(:fatherOf)
Disjoint object properties	DisjointObjectProperties(:isFriendOf :isOpponentOf)
Disjoint data properties	DisjointDataProperties(:FirstName :LastName)
Keys	HasKey (:Student :hasRegistrationNo)

```
SELECT distinct ?p
Where {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        ?p rdfs:domain ?k.
        ?p rdfs:range ?k
        Filter not Exist {
                        ?a ?p ?a
                        }
        }
```

FIGURE 3.1: SPARQL query for irreflexive object property pattern

and number of instances against the selected classes is obtained on the basis of the usage of classes, their depth, mean value and random number. This approach provides basis to generate dataset for any given domain. The algorithm for the dataset generation is mentioned in Figure 3.2.

The above figure shows that in the first step, we take the mean value of each class as a measure to find out to which degree a class is used as subclass of relations, restriction, domain and range in the data schema. Later, classes are ranked according to their mean value and only top quadrant is selected. In our

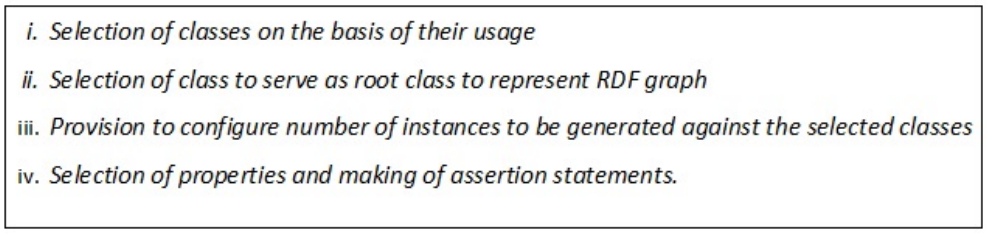

FIGURE 3.2: Algorithm for the dataset generation

TABLE 3.2: OWL2 class assertions in the proposed dataset generator

OWL2 Assertions	Proposed Dataset generator
SameIndividual	SameIndividual(:AP10 :AssistantProfessor10)
DifferentIndividual	DifferentIndividuals(:UnderGraduateStudent1 :GraduateStudent22 a:AssociateProfessor13)
ObjectPropertyAssertion	ObjectPropertyAssertion(:fatherOf :AssociateProfessor13 : UnderGraduateStudent1)
Negative Data property assertion	Negativedatapropertyassertion(:Assistantprofessor1 telephone number 0323434334)
Negative Object property assertion	NegativeObjectPropertyAssertion(:hasFather :UniveristyGraduate1 :AssociateProfessor10)

case 32 classes out of 131 classes are chosen. The selected classes are then mapped to their original hierarchy to form clusters for determination of the candidate classes that are suitable for serving as a root class. As a result, among the five formulated clusters (organization, person, publication, course and research) two main clusters are chosen, led by Person and Organization class respectively. In the second step, a root class is dynamically selected (i.e. Organization class) from the main clusters on the basis of superiority of domain and range associations. In the third step, the configuration for generating instances for each selected class is based on the depth of the class, its mean value and a random number (between 10 to 100). In the last step, properties are selected whose range and domain belong to the selected classes. These selected properties contain both OWL1.1 and OWL2 property characteristics while assertion statements generation follows UOBM [9].

3.4 QuerySet

The benchmark queries characterize high selectivity, input size, output size, inference and query language constructs. With the purpose of checking the performance of KBSP on OWL2 semantics, the proposed benchmark queries have been divided into simple queries (SQ), complex queries (CQ) and object properties characteristics pattern queries (OPQ).

3.4.1 Simple Queries

Simple Queries (SQ) are about checking OWL2 constructs from structure point of view. This is just to confirm that OWL2 semantics are answered by the KBSP or not. There are four queries in the simple queryset to help users to check the OWL2 constructs.

Query1 (SQ1): The following query defines two triple patterns interconnected with

```
SELECT distinct ?Class,?Proptype
        WHERE
                { ?Class rdf:type Owl:ObjectProperty.
                ?Class rdf:type ?Proptype
                }
```

FIGURE 3.3: Simple query for the distinct object properties

each other via AND. The first pattern describes the variable ?Class is bound to the node with first edge rdf:type pointing to owl:ObjectProperty and second edge pointing to ?Proptype. Then joining both individual set mappings on variable ?Class. The result would hold all the mappings from ?Class to the nodes which satisfy the two patterns. Then the Select would project the result for the variables ?Class and ?Proptype. The result of the query enlist the distinct object property characteristics.

Query2 (SQ2):

SQ2 comprises of only one triple pattern. The purpose of this query is to check the support of KBSP for the OWL2 restriction. This query simply retrieves all the object properties using property chain axiom.

```
SELECT ?ObjectProperty
WHERE
        {
        ?ObjectProperty Owl:PropertyChainAxiom ?node
        }
```

FIGURE 3.4: Simple query for the property chain axiom

Query3 (SQ3):

```
SELECT  ?domain ?p ?range
WHERE
        {
        ?p rdfs:domain ?domain.
        ?p rdfs:range ?range
        }
```

FIGURE 3.5: SPARQL query for obtaining domain and range

The purpose of the above query is to obtain the domain and range of all the object and data properties.

Query4 (SQ4):

```
SELECT ?subject ?object
WHERE
        {
        ?subject owl:disjointUnionOf ?object
        }
```

FIGURE 3.6: Simple query to find classes which are disjointunion

This above query is used to find all the classes which are associated with OWL2 construct DisjointUnion.

3.4.2 Complex Queries

The main objective of the benchmark queries has been checking the performance of the KBSPs. For this purpose queries are written in a way to check the behavior of KBSP on complex graph patterns. Doing so, one common way is to write long chains queries where nodes are linked to each other through a long path [24]. In this way query is written for the node having the longest path. The other way is the bushy pattern query where single nodes having connected to large number of other nodes [24]. Similarly, the queries involving high selectivity are also common performance determinant. In the proposed benchmark, we have categories these type of queries as complex queries (CQ). These queries include bushy patterns, long chains, large size, and to select all the instances of all the disjoint object properties to check the responses of KBSP systems.

Query5 (CQ1):

```
SELECT ?Publication ?Pub_Author ?friend_name ?university_degree_from
?teaching_course ?ResearchArea
WHERE
      {
      ?Publication oeb2:publicationAuthor ?PubAuthor.
      ?PubAuthor oeb2:isFriendOf ?friendname.
      ?PubAuthor  oeb2:hasMasterDegreeFrom
      ?university_degree_from.
      ?PubAuthor oeb2:teacherOf ?teaching_course.
      ?PubAuthor oeb2:researchInterest ? ResearchArea
      }
```

FIGURE 3.7: Complex query for the bushy pattern

The above query in Figure 3.7 describes the bushy pattern i.e. variables Publication, Pub_Author, friend_name, university_degree_from, teaching_course , ResearchArea are linked with the publication authors who having publication, friendship, holds master degree from the university, teaching course and also have research interest. This query is used to find all the authors, their publications,

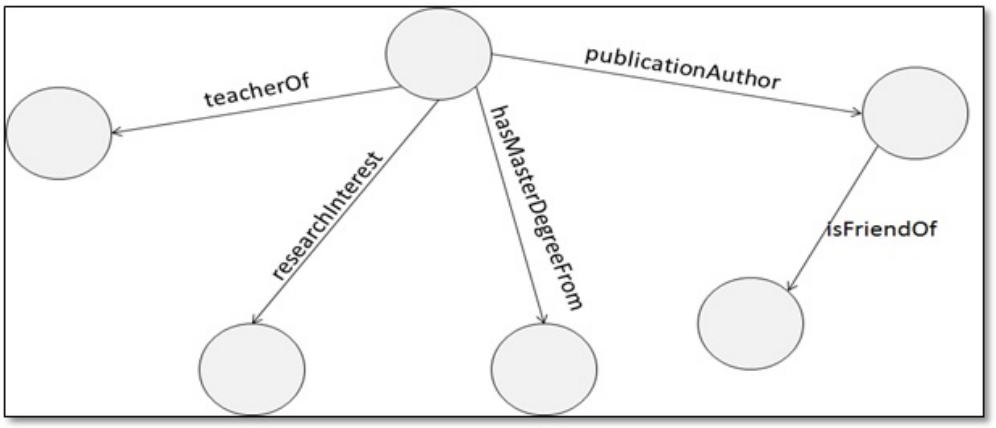

FIGURE 3.8: Bushy pattern query explained through figure

friends, universities from where they obtained masters degree, teaching courses and research interests. The Bushy Pattern query is also explained through figure in Figure 3.8.

Query6 (CQ2):

```
SELECT ?Publication ?Pub_Author ?supervisor_name ?membership_institute ?organization
WHERE
        {
        ?Publication oeb2:publicationAuthor ?PubAuthor.
        ?PubAuthor oeb2:isAdvisedBy ?supervisor_name.
        ?supervisor_name oeb2:isMemberOf ?membership_institute.
        ?membership_institute oeb2:subOrganization ?organization
        }
```

FIGURE 3.9: Complex query to find the long chain pattern

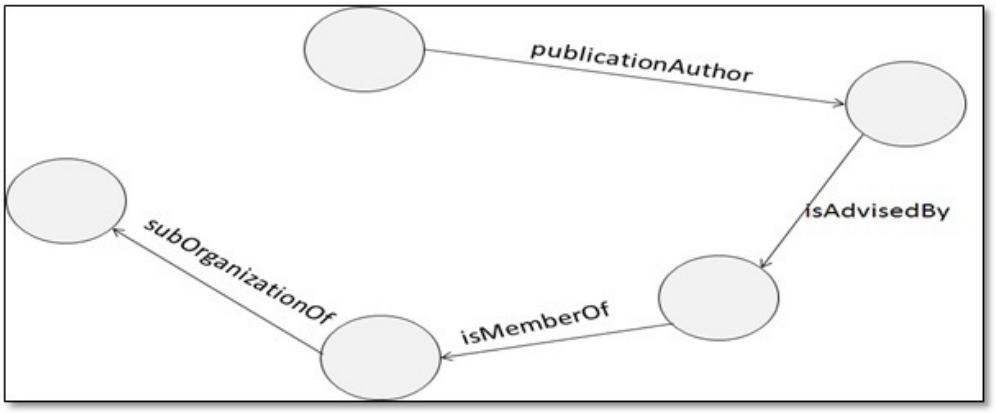

FIGURE 3.10: Long chain pattern query explained through figure

Query6, as Figure 3.9 describes the long chain patterns i.e. variables Publication, Pub_Author, supervisor_name, membership_institute, organization are linked to four triple patterns with conditions of author having publication under some supervisor who is the member of some institute and that institute is also a sub organization. Figure 3.10 explains, how long chain query is visualized as figure.

Query 7(CQ3):

```
SELECT ?student
WHERE
        {
        ?student oeb2:takesCourse  ?course
        }
```

FIGURE 3.11: Complex query for the high selectivity

The above query implements the high selectivity pattern and enlist all the students of all the classes who takesCourse.

Query 8(CQ4):

```
SELECT  ?property1 ?property2 ?instanceX1 ?instanceY1 ?instanceX2 ?instanceY2
WHERE
        {
        ?property1 owl:propertyDisjointWith ?property2.
        ?instanceX1 ?property1 ?instanceY1.
        ?instanceX2 ?property2 ?instanceY2
        }
```

FIGURE 3.12: Complex query for the instances with disjoint object properties

Query8 is used to select all the instances of all those classes having disjoint object properties. The variables in this query are linked to three triple patterns.

3.4.3 Object Properties characteristics pattern queries

The object property characteristics pattern based queries (OPQ) are used for obtaining the patterns from the assertion data [25]. There are nine queries in

the OPQ to cover inverse of, inverse functional, Asymmetric, Functional, Transitive, Disjoint, Symmetric, Irreflexive and Reflexive object properties (OPQ9 to OPQ17). Although Disjoint is not an object property characteristics but it is included in this group due to pattern queries.

Query 9(OPQ9):

```
SELECT ?a ?p1 ?b ?p2
WHERE
        {
        ?a ?p1 ?b.
        ?p1 rdf:type owl:ObjectProperty.
        ?b ?p2 ?a
        ?p2 rdf:type owl:ObjectProperty.
        filter(?p1!=?p2)
        }
```

FIGURE 3.13: Pattern query for the inverse-of-relationship

Inverse object property states that the object property expression P1 is an inverse of the object property expression P2 when an individual x is connected to an individual y by P1 and y is also connected to x by P2, and vice versa.

The Figure 3.13 describes Query 9, exhibiting the pattern of inverse of property characteristics. When this pattern query is run on the data schema, the instances alongwith properties matching the inverse of relationship are retrieved.For example, isAssurerRoleOf and playsAssurerRole properties alongwith instances of Assurer and Person Classes are retrieved. This is shown in Figure 3.14.

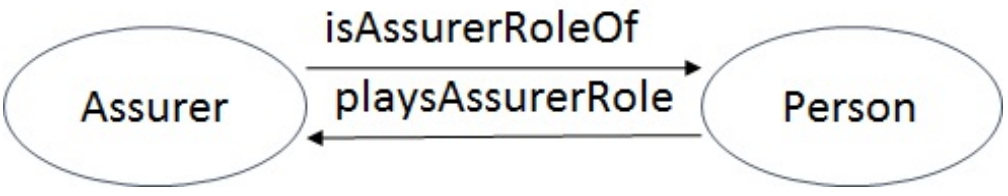

FIGURE 3.14: The retrieved inverse-of-relationship

Query 10 (OPQ10):

```
SELECT ?a ?p ?b ?c
WHERE
        {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        ?c ?p ?b
        filter(?a!=?c)
        }
```

FIGURE 3.15: Pattern query for the inverse functional characteristics

Inverse Functional object property pattern is basically the inverse of the Functional property. According to OWL2 Web ontology language structural specifications [26], Inverse Functional object property pattern OPP may be described as, for every individual X, there is at most one individual Y where X is connected with Y through OPP. The query in Figure 3.15 retrieves the instances against the inverse functional pattern over the dataset. When the pattern query of Inverse Functional is run on the schema, the results shows the object properties enrollIn, hasDoctoralDegreeFrom, is_advisedBy, etc with instance from various classes, as shown in the figure 3.16.

FIGURE 3.16: The retrieved inverse functional relationship

Query 11 (OPQ11):

Asymmetric property pattern is defined as, if an individual X is connected to the individual Y through an object property pattern OPP then individual Y cannot

```
SELECT ?a ?p ?b
WHERE
        {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        Filter not Exist {?b ?p ?a}
        }
```

FIGURE 3.17: Pattern query for the asymmetric pattern

be connected to individual X through OPP[26]. The query shown in the figure 3.17 retrieves the instances for the asymmetric property pattern.

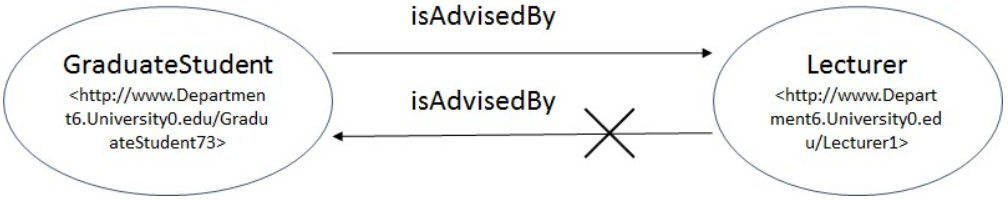

FIGURE 3.18: The retrieved asymmetric relationship

When the pattern query is run, the results show isAdvisedBy, publicationAuthor, takesCourse, etc alongwith instances. Figure 3.18 explain the asymmetric property pattern.

Query 12 (OPQ12):

Functional property pattern is defined as, if there is at most one individual Y for individual X through the object property pattern OPP. Figure 3.19 describe the pattern query for the functional property characteristics. By running the query pattern, the obtained results shows the object properties hasUndergraduateDegreeFrom, isAdvisedBy, teachingAssistantOf, etc with individuals from classes having at most one individuals from other classes. Figure 3.20 shows that there can be only one university for the http://www. Department10.University0.edu /GraduateStudent0 from where he received his under graduate degree i.e. http://www. University85.edu

```
SELECT ?a ?p ?b
WHERE
        {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        Filter not Exist
                        {
                        ?a ?p ?c
                        Filter(?b!=?c)
                        }
        }
```

FIGURE 3.19: Pattern query for the functional property pattern

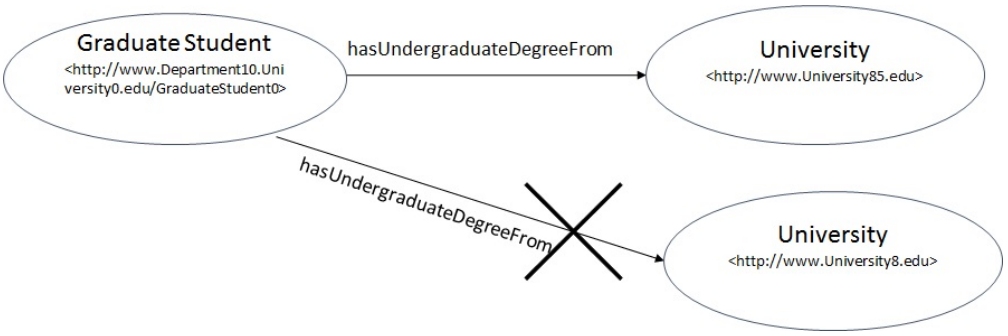

FIGURE 3.20: The retrieved functional property characteristics relationship

Query 13 (OPQ13): Transitive property pattern is defined as a relationship such that object property pattern exhibiting OPP(x, y) and OPP(y,z) implies OPP(x,z). Figure 3.21 describe the pattern query of transitive relationship. By running the pattern query, the obtained results shows isFriendOf relation with instances matching the pattern criteria. The transitive behavior is shown in Figure 3.22.

Query 14(OPQ14): Disjointness of classes is defined as a relationship such that two classes are not sharing the common elements in them. When the patterned query as shown in Figure 3.23 is run, the results show the list of pair of classes not sharing common instances. For example, the pairs (Aeronautical_Engineering_Class,

```
SELECT ?a ?p ?b ?c
WHERE
        {
        ?a ?p ?b.
        ?b ?p ?c.
        ?a ?p ?c.
        ?p rdf:type owl:ObjectProperty
        }
```

FIGURE 3.21: Pattern query for the transitive characteristics

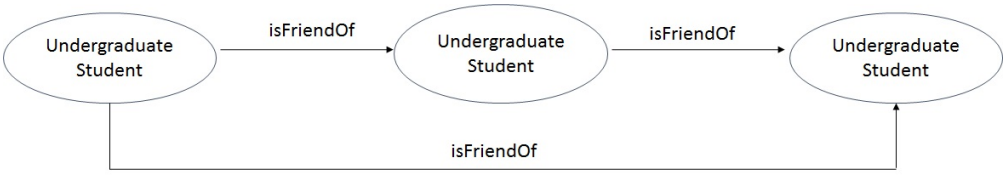

FIGURE 3.22: The retrieved transitive property characteristics relationship

```
SELECT distinct ?s ?cls ?t ?cls2
WHERE
        {
        ?s a ?cls.
        ?t a ?cls2.
        Filter (?s!=?t && ?cls!=?cls2)
        }
```

FIGURE 3.23: Pattern query for the disjoint characteristics

Humanities Class), (DramaClass, Car), (Assurer, Automobile), etc are disjoint classes.

Query 15 (OPQ15):

```
SELECT ?a ?p ?b
WHERE
        {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        ?b ?p ?a
        }
```

FIGURE 3.24: Pattern query for the symmetric characteristics

A Symmetric property pattern is described as, if an individual X is connected to the individual Y through an object property pattern OPP1 and individual Y is connected to individual X through OPP1. Figure 3.24 describe the symmetric relationship query.

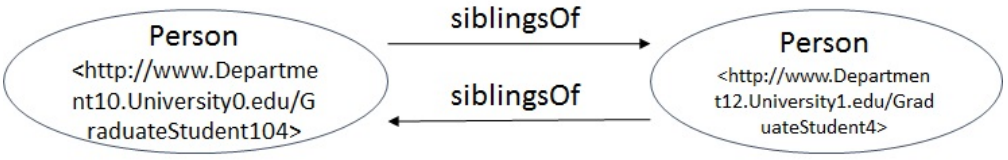

FIGURE 3.25: The retrieved symmetric property characteristics relationship

When the query pattern of symmetric relation is run, the results show isFriendOf, siblingsOf, etc alongwith instances as shown in figure 3.25.

Query 16 (OPQ16): Irreflexive object property pattern is described as, if an object

```
SELECT ?a ?p ?b
WHERE
        {
        ?a ?p ?b.
        ?p rdf:type owl:ObjectProperty.
        Filter not Exist
                        {
                        ?a ?p ?a
                        }
        }
```

FIGURE 3.26: Pattern query for the irreflexive characteristics

property pattern OPP relates an individual X with individual Y, but the individual X is not related to itself through OPP. The query describe the irreflexive relationship is shown in figure 3.26.

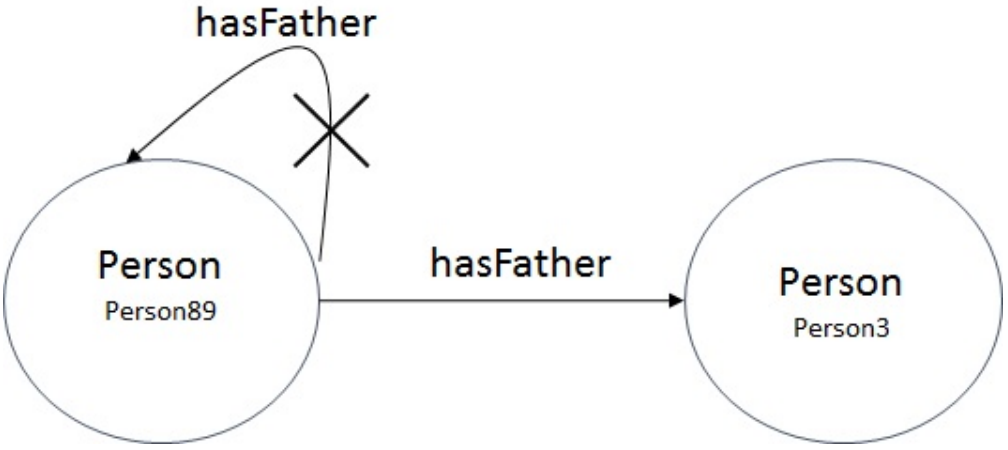

FIGURE 3.27: The retrieved irreflexive property characteristics relationship

When the pattern query is run, the patterned properties obtained include hasFather, enrollIn, isAdvisedBy, isFriendOf, etc. The figure 3.27 shows the irreflexive relation.

Query 17 (OPQ17): An object property reflexivity axiom states that the object property expression OPE is reflexive that is, each individual is connected by OPE to itself. The query to describe the reflexive relationship is shown in figure

```
SELECT ?a ?p ?a
WHERE
        {
        ?a ?p ?a. ?p rdf:type owl:ObjectProperty.
        }
```

FIGURE 3.28: Pattern query for the reflexive characteristics

3.28. The patterned properties showing reflexive relationships include like, loves,

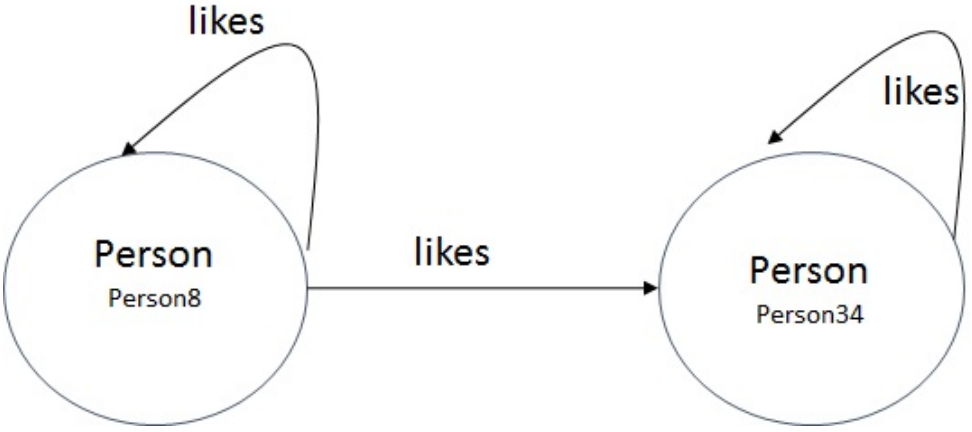

FIGURE 3.29: The retrieved reflexive property characteristics relationship

isFriendOf, etc. One such relationship is shown in figure 3.29.

3.5 Summary of the Chapter

This chapter described the selection of data schema as a case study and explain the methods adopted in building up the proposed evaluation benchmark for the enrichment of the data schema, dataset (alongwith dataset generator), and queries set with complete OWL semantics coverage.

Chapter 4

ANALYSIS OF THE OEB2 BENCHMARK

4.1 Overview

This chapter describe the analysis of the proposed benchmark (OEB2) against the four activities used in the chapter 2 for the evaluation. These activities comprises of examining the data schema for the coverage of OWL semantics, evaluate the data schema complexity, inspection of the dataset and benchmark queries for OWL coverage. The analysis of the proposed benchmark in this chapter is concluded with comparing the results of the proposed benchmark with the exsiting evaluation benchmarks.

4.2 Examine the data schema for the complete OWL semantics

The data schema of OEB2 is built using the three activities i.e. survey of domain ontologies, use of WordNet senses, and execution of pattern queries [detail is mentioned in chapter 3]. This enabled in adding the missing OWL1.1 and OWL2

semantics in the university ontology. As a result, OEB2 provides complete coverage of all the OWL1.1 and OWL2 semantics in the data schema. The usage of few important OWL constructs in the data schema by the benchmarks is shown in Figure 4.1& 4.2. Dbpedia benchmark shows high use of domain and range constructs as compared to the other benchmarks. Both LUBM [8] and UOBM [9] have high use of SubClassOf relation constructs. The coverage of subClassOf is also high in OEB2. Figure 4.2 shows that OEB2 overtake other benchmarks in using all the object properties in the data schema. The coverage of SubClassOf relation and object properties characteristics indicates the data schema of OEB2 as a balance between concept organization and property associations.

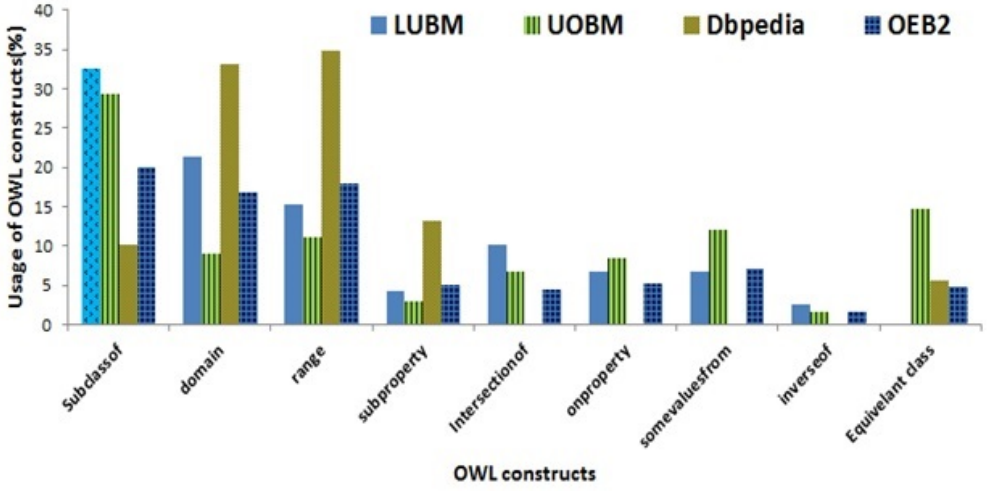

FIGURE 4.1: Comparison of OWL & OWL2 constructs usage in the OEB2 and existing benchmarks ontologies

4.3 Examine the complexity of the data schema

As mentioned in Chapter 2 that the complexity of the data schemas is based on the EnR of OWL constructs and usage of object property characteristics. Construct wise EnR of OEB2 and other benchmarks is shown in figures 4.3, and 4.4. Overall EnR for the OWL constructs in OEB2 as shown in Figure 4.5. is higher as compared to the other benchmarks. Just focusing on the KBSPs evaluation benchmarks (i.e. LUBM, UOBM, and OEB2), we see that Figure 4.6 gives a picture of overall EnR with and without OWL constructs, which are not drawn at edges.

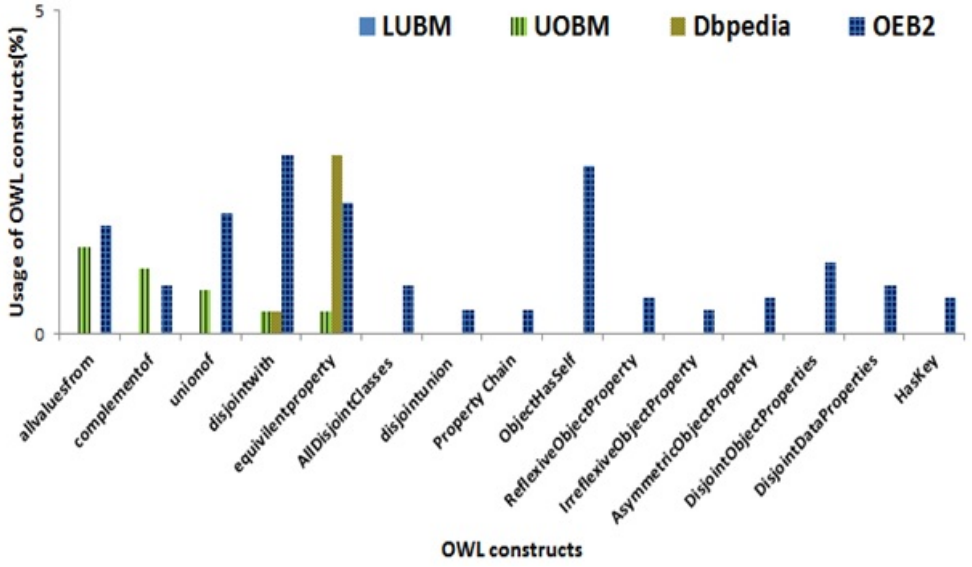

FIGURE 4.2: Comparison of OWL & OWL2 constructs usage in the OEB2 and existing benchmarks ontologies

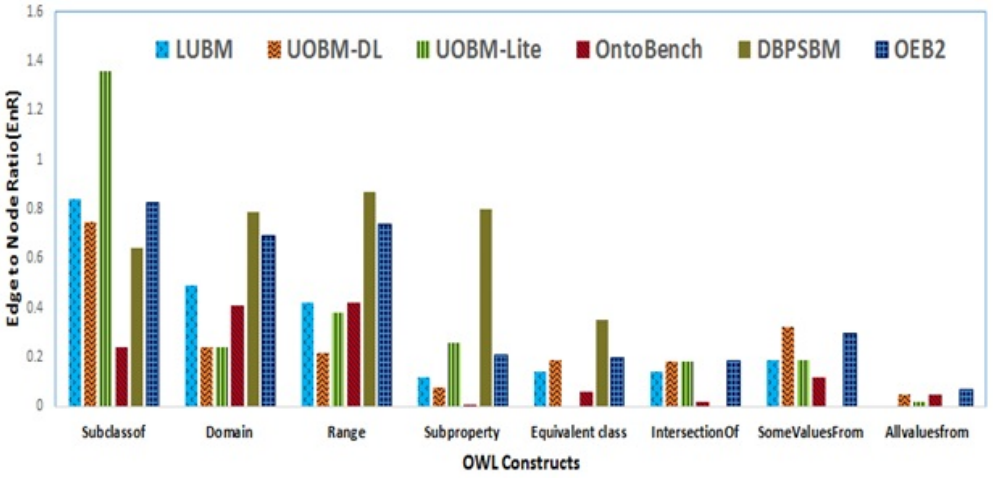

FIGURE 4.3: Comparison of EnR between OEB2 and existing benchmarks ontologies

The figure shows that the proposed benchmark has an upper edge over LUBM and UOBM data schema complexity in both combined ENR with and without owl constructs. By checking the usage of object property characteristics in the data schemas as shown in Figure 4.7, it is seen that OEB2 outperforms other evaluation benchmarks.

And finally, Figure 4.8 shows the overall schema complexity,which concludes that OEB2 is semantically more complex as compared to KBSP evaluation benchmarks.

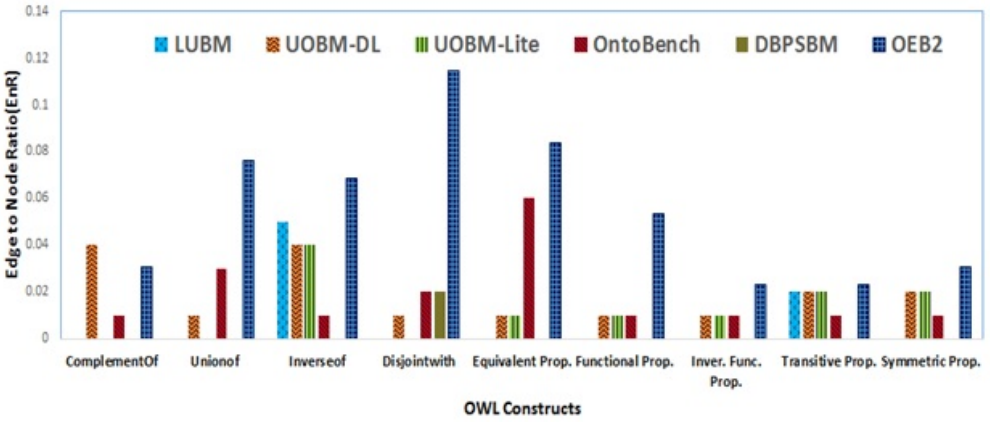

FIGURE 4.4: Comparison of EnR between OEB2 and existing benchmarks ontologies

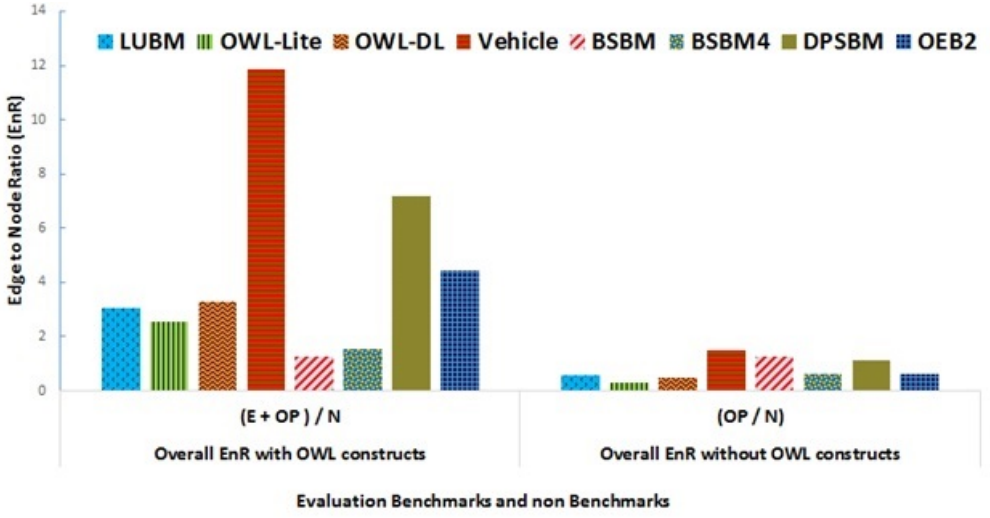

FIGURE 4.5: EnR with and without OWL constructs

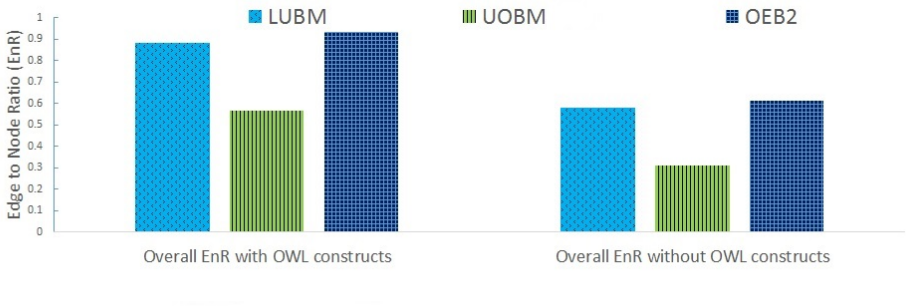

FIGURE 4.6: Overall EnR with & without OWL constructs not drawn at edges

The result also shows that OEB2 is association centric as compared to the hierarchical centric behavior of the other evaluation benchmarks(i.e. LUBM, UOBM).

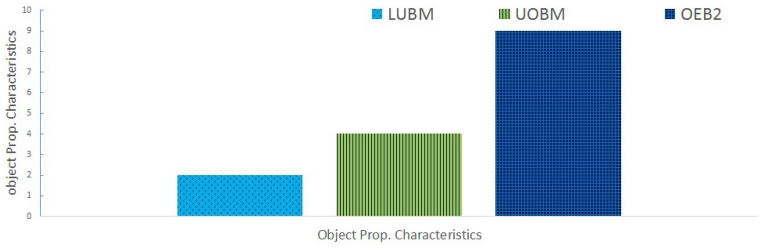

FIGURE 4.7: Usage of Object Property Characteristics

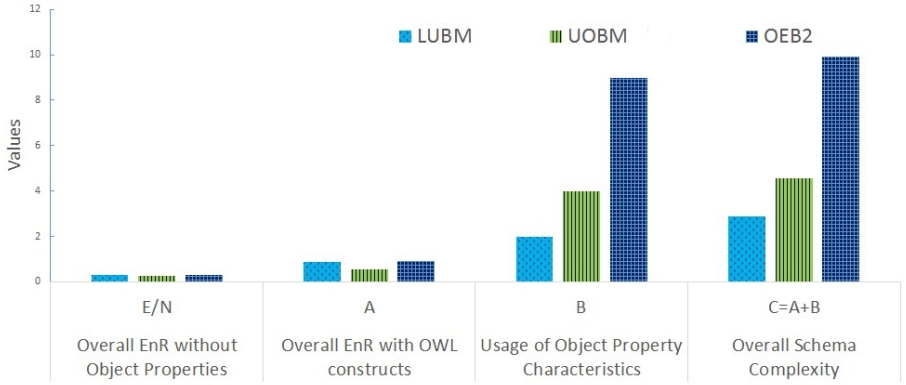

FIGURE 4.8: Overall schema complexity

TABLE 4.1: Percentage usage of the ontology constructs by the benchmark dataset generators

Benchmark	Classes			Property Characteristics			Object Properties			Data Properties		
	Total	Used	% Usage	Total	Used	% Usage	Total	Used	% Usage	Total	Used	% Usage
LUBM	43	18	41.86	9	1	11.11	25	13	52.00	7	4	57.14
UOBM	113	32	28.32	9	4	44.44	35	21	60.00	9	6	66.67
OEB2	132	61	46.21	9	9	100	41	24	58.54	11	3	27.27

4.4 Inspection of the dataset for a sufficient set of OWL semantics

Due to sufficient use of OWL constructs, using of all the property characteristics, increased number of object properties make the data schema of OEB2 more complex than LUBM and UOBM. OEB2 provides higher coverage of OWL constructs in the dataset generation process. Table 4.1 shows the comparison of ontology constructs usage in the dataset generators. For users defined classes and object properties OEB2 has higher percentage of usage.

OEB2 provides hundred percent use of the object property characteristics as compare to LUBM and UOBM. The reason is shown in Table 4.2, which states that

Table 4.2: Usage of Object Property Characteristics by the benchmark dataset generators

	Object Property Characteristics								
Benchmark	Functional	Inv. Function	Transitive	Symmetric	Disjoint	Reflexive	Irreflexive	Asymmetric	InverseOf
LUBM	Yes	No	No	No	No	No	No	No	No
UOBM	Yes	Yes	Yes	Yes	No	No	No	No	No
OEB2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

OEB2 uses all the nine (09) object property characteristics while LUBM and UOBM uses only one (01) and (04) object property characteristics respectively. The usage of Classes and Object properties in OEB2 is similar to the other benchmarks (Table 4.1). Moreover, OEB2 dynamically selects Classes and Properties as compared to the static approach of existing benchmarks. For generation of instances, OEB2 adopts a concise approach in contrast to the exhaustive approach followed by existing benchmarks. For example, isFriendOf is a reflexive property; then only one statement would express both directional relationship between two distinct individuals as followed by OEB2 (i.e. A isFriendOf B inherently covers B isFriendOf A).

4.5 Inspection of the benchmark queries for a sufficient set of OWL semantics

OEB2 benchmark queries are evaluated by comparing the coverage of OWL constructs in OEB2 and existing benchmark queries. The proposed queries have a higher coverage of ontology constructs as compared to other existing benchmarks. For example, the query (OPQ8) retrieves twenty one interconnected object properties that reflects the true coverage of the OEB2 queries. The reason is that, most of the OEB2 queries are generic in nature, which means a query does not look for a specific class instance or a property, rather instances are retrieved against the object property characteristics or a queried pattern e.g. bushy patterns and long chain queries. Moreover, the classification of OEB2 queries supports evaluation of different domains ranging from simple to high complexity. This would help the researchers to exploit the structural complexities of the KBSP.

4.6 Summary of the chapter

This chapter has presented the analysis of the proposed benchmark (OEB2) against the four evaluation activities which are used to evaluate the surveyed evaluation benchmarks. The data schema of OEB2 provides complete coverage of OWL (1.1 & 2) semantics as compared to the existing benchmarks. Further, the proposed benchmark make use of all the object properties characteristics offered in OWL language and user defined object properties. The proposed data schema exhibits more schematic complexity as compared to other evaluation benchmarks. This is shown in the EnR with and without OWL constructs. Although Figure 4.5 illustrates high EnRs for Vehicle data schema and Dbpedia data schema. Yet, both of these data schemas are not a part of the KBSPs evaluation benchmarks. Similarly, EnR without OWL construct is also significant for the proposed benchmark. This finding shows that much emphasis has been given to ensure the coverage of OWL semantics in the data schema. OEB2 adopts concise approach in dataset generation as compared to the static approach followed by the other evaluation benchmarks. This raise the potential of the proposed dataset generator usable for other domains. Further, OEB2 makes uses of OWL2 assertions in the dataset, which are missing in the existing evaluation benchmarks. Due to the generic nature of queries, the proposed benchmark provides higher coverage of OWL semantics as compared to the existing evaluation benchmarks. Moreover, different type of OEB2 queries support in evaluation of KBSPs against different domains ranging from simple to complex nature. The outcome of this chapter encompasses that the analysis of OEB2 and its comparison with existing benchmarks concludes that OEB2 outperforms others in the coverage of OWL semantics.

Chapter 5

EVALUATION OF KBSPs USING OEB2

5.1 Overview

In this chapter, well known KBSPs are evaluated using OEB2 benchmark to demonstrate that whether the proposed benchmark works in the real environment or not. For this purpose three synthetic datasets of size 24K, 240K and 2400K triples are generated for the said evaluation. The semantic tools used for conducting the experimentation include Jena API, OpenRDFWorkbench, Protege, MySQL and SQL Server on Intel Core i5-4200M CPU @ 2.5 GHz with 6 GB RAM. The KBSP used for the evaluation includes Sesame (in memory), Sesame (DB), Jena SDB, Blazegraph, RDF Native storage system, OWL2ToRDB, and OntRel. The evaluation metrics comprises of load time and response time.

5.2 Load Time Behavior of the KBSPs

The load time of different KBSPs is shown in the Figure 5.1, where x-axis represents KBSPs and y-axis shows the load time in milliseconds. The benchmark shows that Sesame (in memory), Native Store and Blazegraph loads 24K triples in less than 20 seconds as compared to the other KBSPs. The reason is that:

1. Memory based KBSPs does not require insert/update routines and therefore, perform better on small size datasets [33].

2. The Native store reduce the load time of the datasets due to the physical organization and proper indexes [34].

3. KBSPs with database storage systems involve schema dependency (i.e. transformation from one schema model to another model i.e. relational model), while loading the dataset [33].

For example, The KBSP with database systems such as OntRel uses a fixed relational schema for the storage of the constructs. OntRel models each individual constructs to a separate predefined relation. Similarly, another KBSP with database storage system used in the evaluation is OWL2ToRDB. The KBSP uses class decomposition approach, where each class and property of the data schema is stored in a separate relation. OWL2ToRDB also maintains meta-tables and record tables. In contrast, Sesame (in-memory) is memory based, native store is file based and Blazegraph is graph based that do not require such schema dependency. For larger datasets (240K and 2400K), the benchmark reveals that non-database KBSPs perform better than the KBSPs with database storage systems. The benchmark demonstrates that KBSP with in memory system performs better in loading the datasets than its counterpart i.e. persistent storage. However, in memory systems are resource dependent and their performance may vary for larger datasets (i.e. over 100 million triples) with less memory size [35]. In the present work, the benchmark reports Sesame (in memory) behavior on the average computing environment for upto 2.5 million triples. The benchmark also reports the consistent behavior of Native store on all the datasets. As Native store uses file system with SPOC (subject, predicate, object and context: the natural sorting order) and POSC (predicate, object, subject and context: the inverse sorting order) triple indexing schemes [34]. Similarly, graph based KBSP also demonstrate better load time like Native store and in memory system. The reason may be that Blazegraph relies on the file system cache to improve the disk access [36].The proposed benchmark highlights that KBSPs with less schema dependency performs

FIGURE 5.1: Load time of KBSPs on different size datasets

better in loading the datasets. As different KBSPs have been selected on the basis of different storage layouts/representations for checking the performance of KBSPs at different scales. The results of KBSPs for data loading at different scales reflect their expected behaviors, which were obtained during the selection and analysis of the KBSPs. So, the benchmark (OEB2) reflects the true behavior for load time of underlying KBSPs because of high data schema complexity, full coverage of OWL constructs, and dynamic generation of dataset.However, the details of causes or factors lead towards different behavior of KBSPs in loading different size datasets is a subject of future work.

5.3 Response Time Behavior of the KBSPs

As the proposed evaluation approach (OEB2) is a type of black box evaluation where KBSPs are evaluated to see how they are suitable for different domains (i.e. simple or complex). This is done by means of checking the performance of KBSPs on different type of queries. The benchmark demonstrate the query response time of different KBSPs. There are three type of queries used by the benchmark. Simple queries are about retrieving the OWL constructs and their

result set is also limited. The results reported by the benchmark shows that the response time of KBSPs is nominal (maximum upto 0.7 milliseconds) and remains unchanged on different size datasets as shown in Figures 5.2, 5.3, & 5.4. Simple queries are build keeping in view of the simple data schemas like agriculture [5] and simple nature of queries like querying the OWL constructs. So scaling of data becomes insignificant in this case. The results of KBSPs on simple queries matches the expected results of the KBSPs and validates the benchmark.

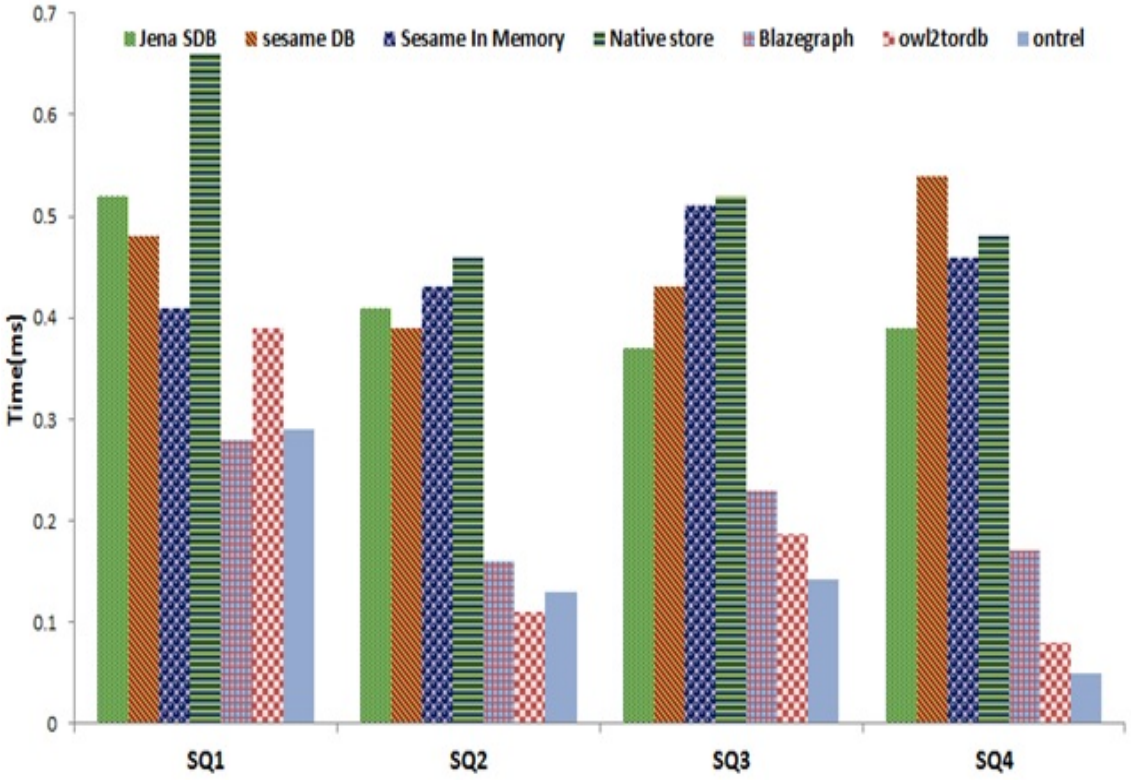

FIGURE 5.2: Simple Query Response time of KBSP on 24K triples

The benchmark reports that Blazegraph, Owl2ToRDB and OntRel performs better than the other KBSPs. The reason is that both OntRel and OWL2ToRDB stores OWL constructs, classes, and properties as individual relations. This factor supports in quick retrieval of results against the simple queries. The consistent behavior of these KBSPs (i.e. OntRel and OWL2ToRDB) on all size datasets reflects that the size factor becomes insignificant for these type of queries, where the result set is of small size and may remain constant in numbers [33]. On the other hand there is a tradeoff between the load time and query response time of such KBSPs for simple queries. The results of the benchmark also shows that graph

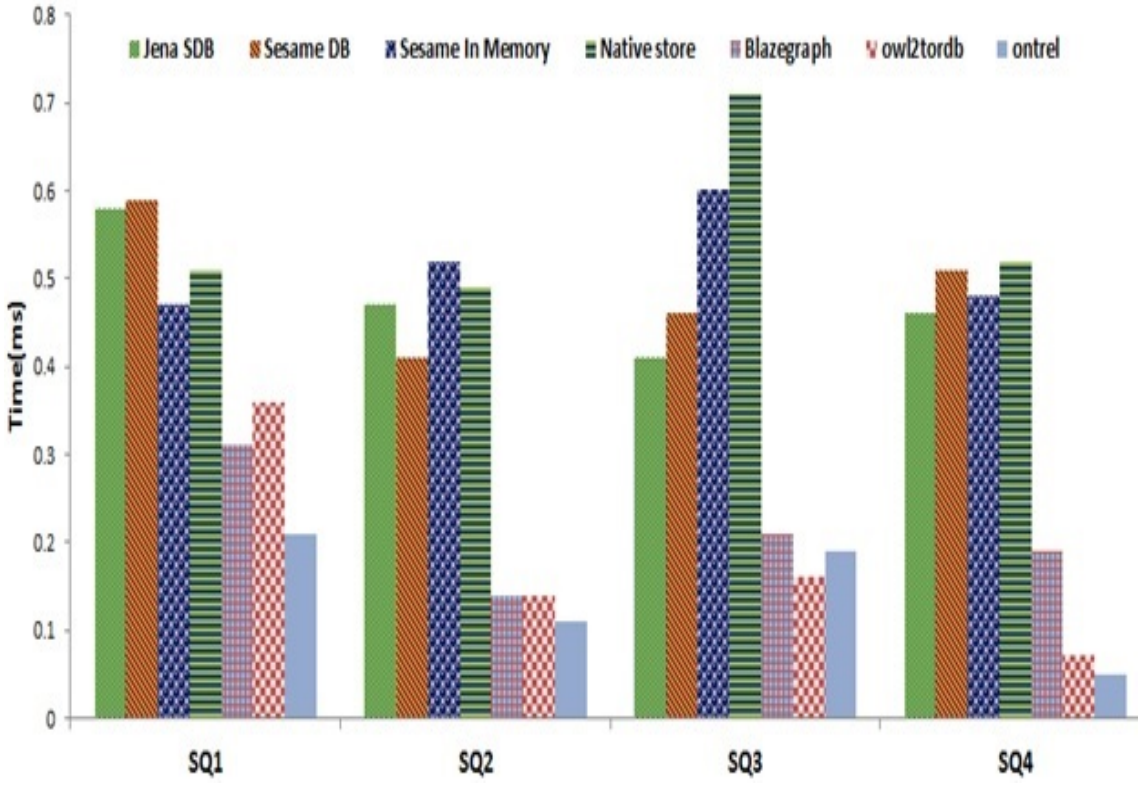

FIGURE 5.3: Simple Query Response time of KBSP on 240K triples

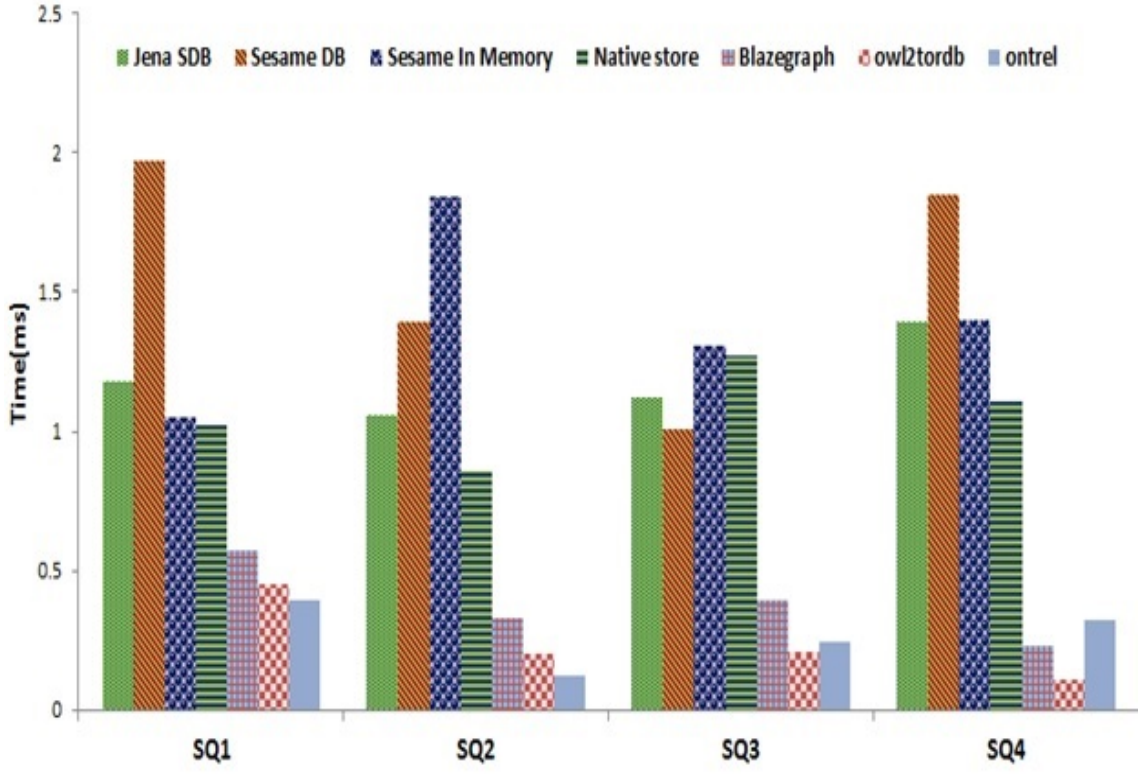

FIGURE 5.4: Simple Query Response time of KBSP on 2400K triples

based KBSP (i.e. Blazegraph) performs equally well for simple queries (basic triple pattern). The reason is that Blazegraph is a schema free graph storage model that supports triplestore and quadstore. Other than the Blazegraph, the response time behavior of the non-database systems is proportional to the memory, result size and computational resources for the simple queries.

The purpose of complex queries is to expose the efficiency of underneath storage representations and query answering mechanism. The first query in complex queries is about Bushy pattern (CQ1). The benchmark reports that graph based KBSP outperforms the other KBSPs on all three datasets. For the long chain query on 240K and 2400K datasets, the benchmark shows that KBSPs with database storage system (i.e. Jena SDB and Sesame DB) have better query response time. As Jena SDB and Sesame DB support inference and reasoning capabilities that subsequently retrieve long chain queries efficiently. The benchmark reports that the database supported KBSPs (i.e. OntRel and OWL2ToRDB) have high response time for bushy pattern and long chain queries as shown in Figure 5.5, 5.6, 5.7. The query (CQ3) is about the high selectivity i.e. retrieving instances without any join and filter applied. All the KBSPs shows similar response time for the high selectivity query against 24K dataset. The benchmark shows that the response time of in memory KBSP (i.e. Sesame) is affected by the size of the result set as shown in Figures 5.6 & 5.7. Overall, the benchmark reports variation in the behavior of KBSPs on different size datasets. This expected variation in the response time of KBSPs on complex queries validates OEB2. However,the similar behaviour of Blazegraph, Jena SDB, OntRel, and Owl2toRDB against CQ3 and CQ4 on different size datasets is not expected.

The third type of queries used by the benchmark are about object property characterisitics pattern queries. The first query (OPQ1) is about inverseOf relationship. The benchmark reports that all the KBSPs have similar response time for OPQ1. The reason is the use of single filter in the pattern query (i.e. the properties used in the inverseOf relations must be different). The behavior of all the KBSPs remains same on different datasets for the query (OPQ1). The query (OPQ2) is about inverse Functional property characteristic. The benchmark shows that KBSPs with

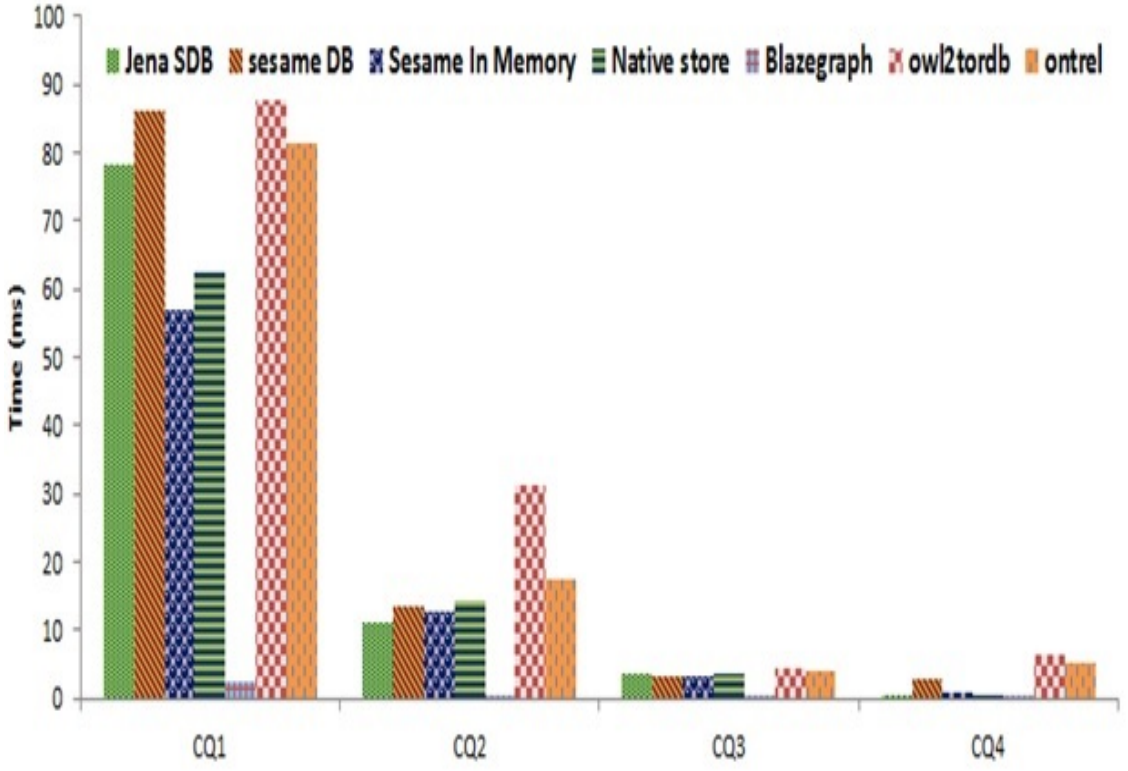

FIGURE 5.5: Complex Query Response time of KBSP on 24K triples

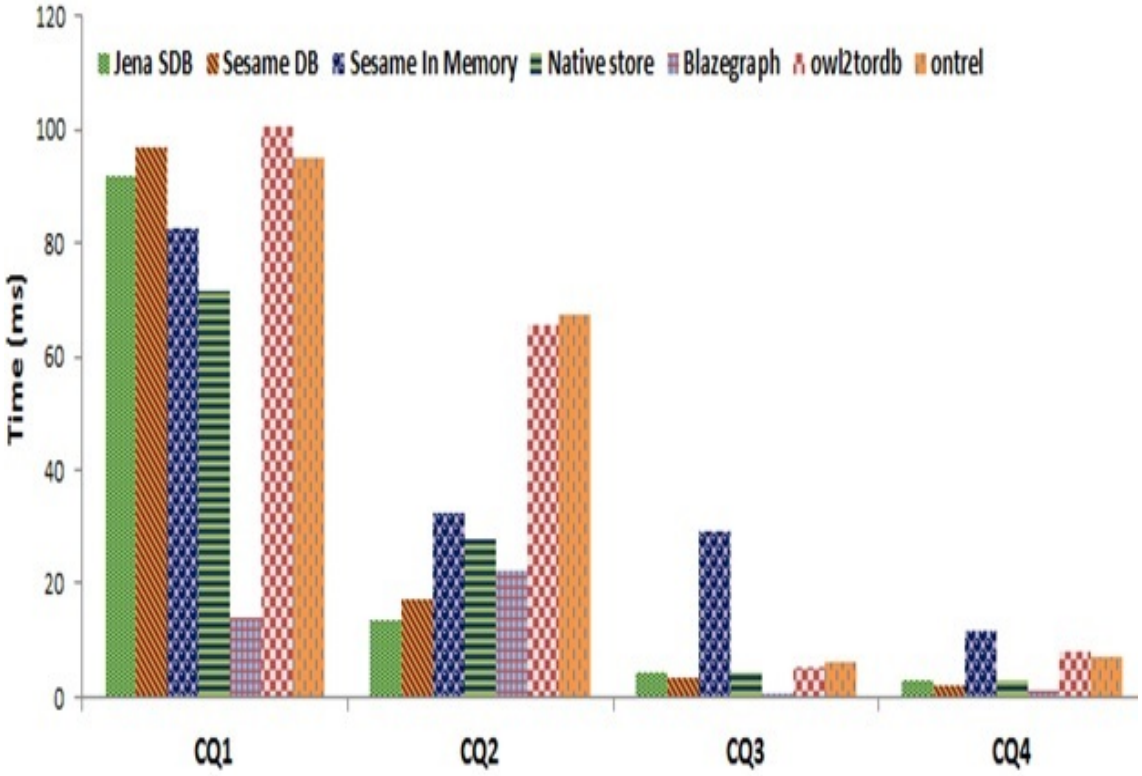

FIGURE 5.6: Complex Query Response time of KBSP on 240K triples

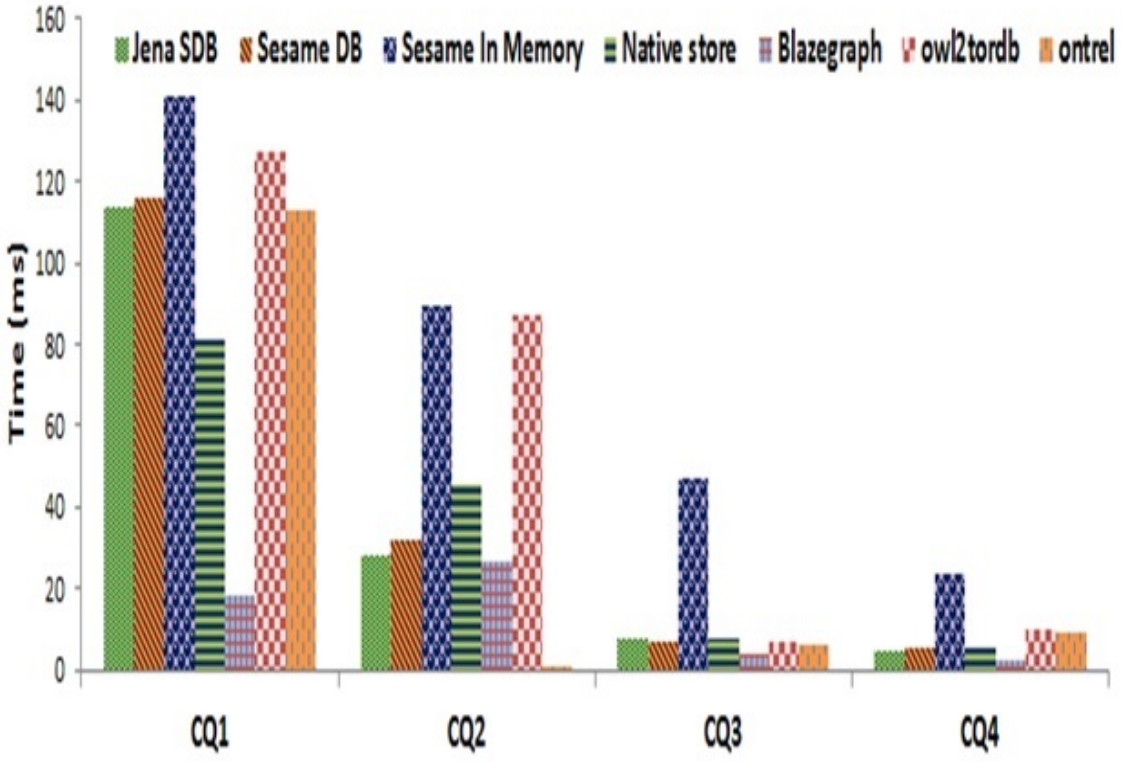

FIGURE 5.7: Complex Query Response time of KBSP on 2400K triples

database systems (i.e. Jena SDB and other database systems) have high spikes for all datasets. Figures 5.8, 5.9,& 5.10 shows that on 24K dataset, the behavior of in-memory, graph and native based KBSPs is far better than the KBSP with database systems (i.e. Ontrel and Owl2ToRDB). Blazegraph demonstrate consistent behavior on all the object property characteristics pattern queries. Sesame (in-memory) shows minimum response time for 24K and 240K datasets but its response time rises on the larger dataset i.e. 2400K. The results of KBSPs have shown their expected behavior on different type of pattern queries. All those KBSPs, which employ high schema dependency takes more time to answer such type of queries. Similarly the behaviour of memory based KBSP is affected by the size of the data.

5.4 Summary of the Chapter

As a summary of the evaluation of the KBSPs, the proposed benchmark concludes that Blazegraph, Owl2ToRDB and OntRel are efficient in simple queries

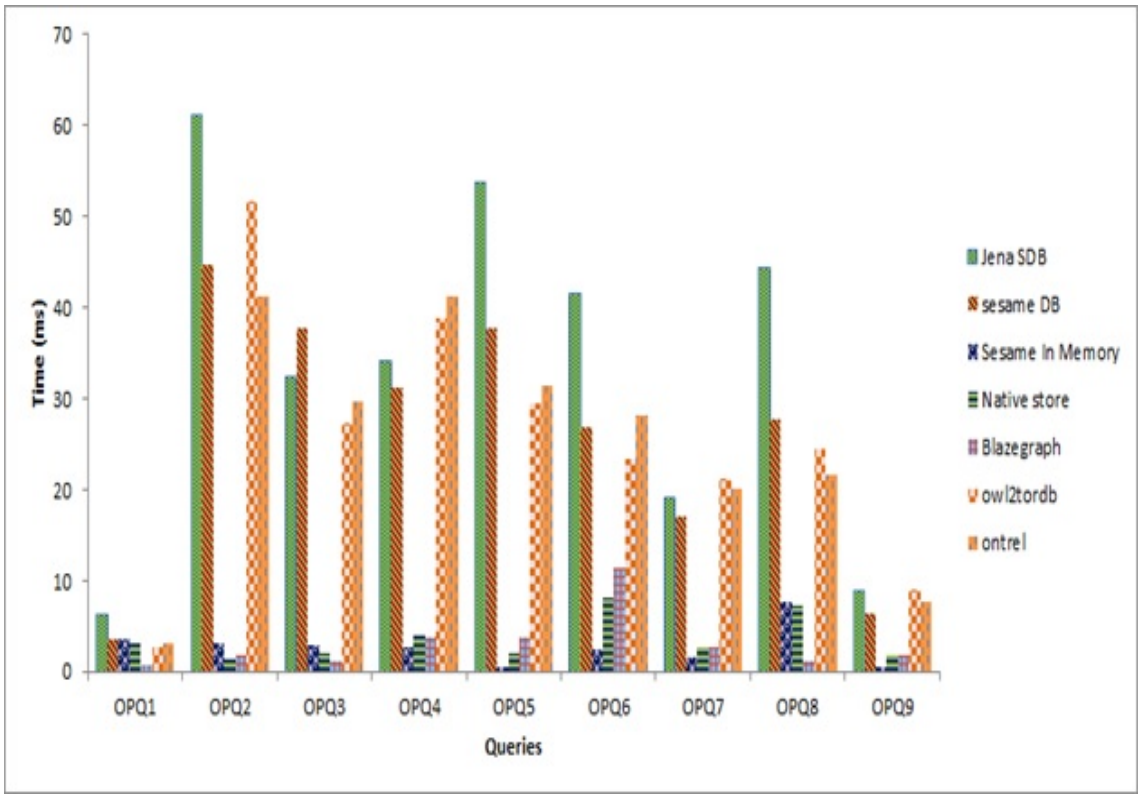

FIGURE 5.8: Property pattern query response time of KBSP on 24K triples

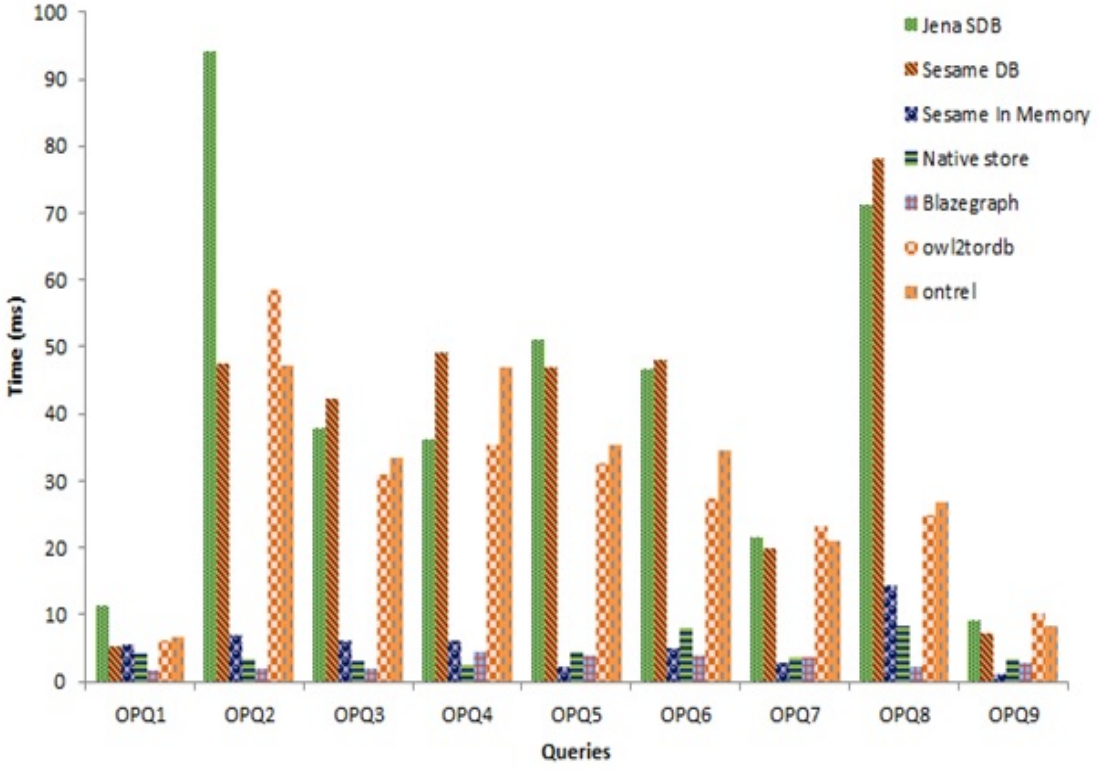

FIGURE 5.9: Property pattern query response time of KBSP on 240K triples

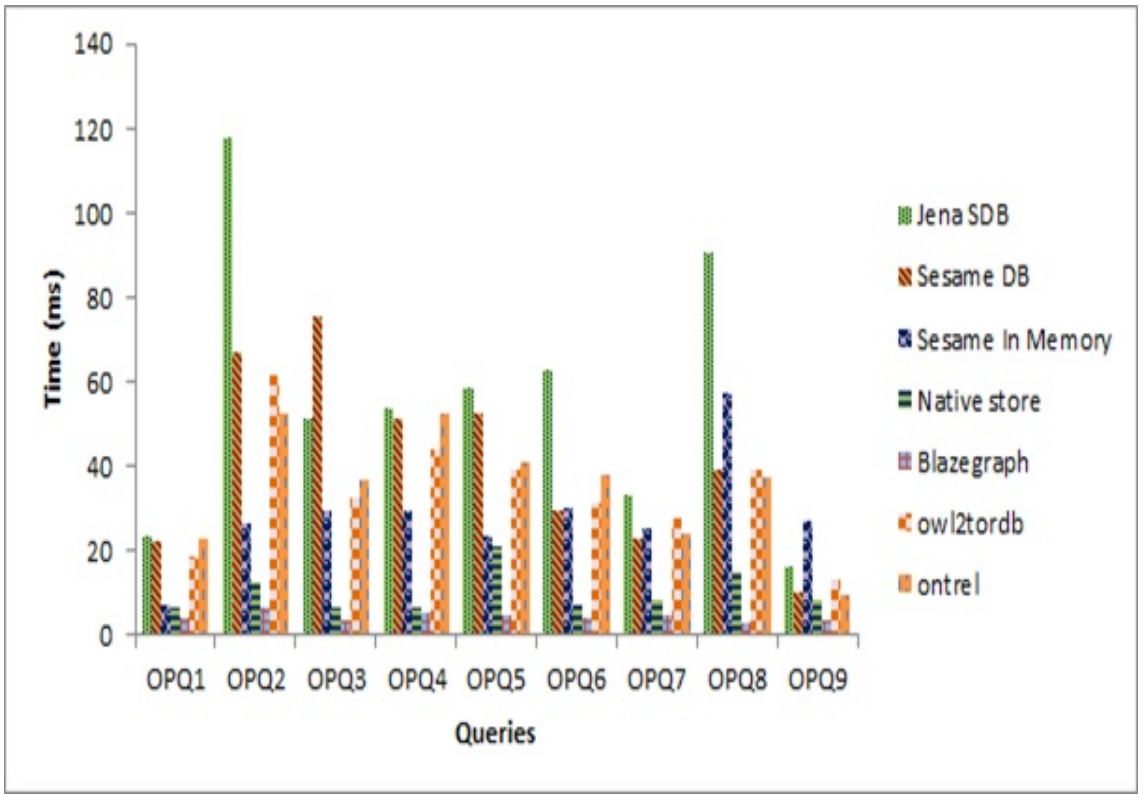

FIGURE 5.10: Property pattern query response time of KBSP on 2400K triples

like searching for distinct properties, domain and range, subject and predicates, etc. as shown in Figure 5.11.

Therefore, the benchmark is able to conclude that the KBSPs (i.e. Blazegraph, OWL2ToRDB, and OntRel) may be suitable choice for domains using simple kind of queries i.e. Agriculture, and education domain. Figure 5.12 shows that Blazegraph out performs all other KBSPs in query response time of complex queries. While Jena SDB, Sesame DB and Native Store perform similar behavior on complex queries. Due to complex nature of queries involving bushy patterns. long chains, irregular patterns, the performance of Sesame in-memory is poor. So the proposed benchmark reports Blazegraph as an appropriate choice for domains like. medical and scientific domains.

The proposed benchmark reports that the performance of the Blazegraph, Sesame (in memory) and RDF native store (Sesame) is better as compared to others in object property characteristics queries (OPQs) as shown in figure 5.13. The performance of KBSPs with relational database KBSP (i.e. Sesame DB, Jena

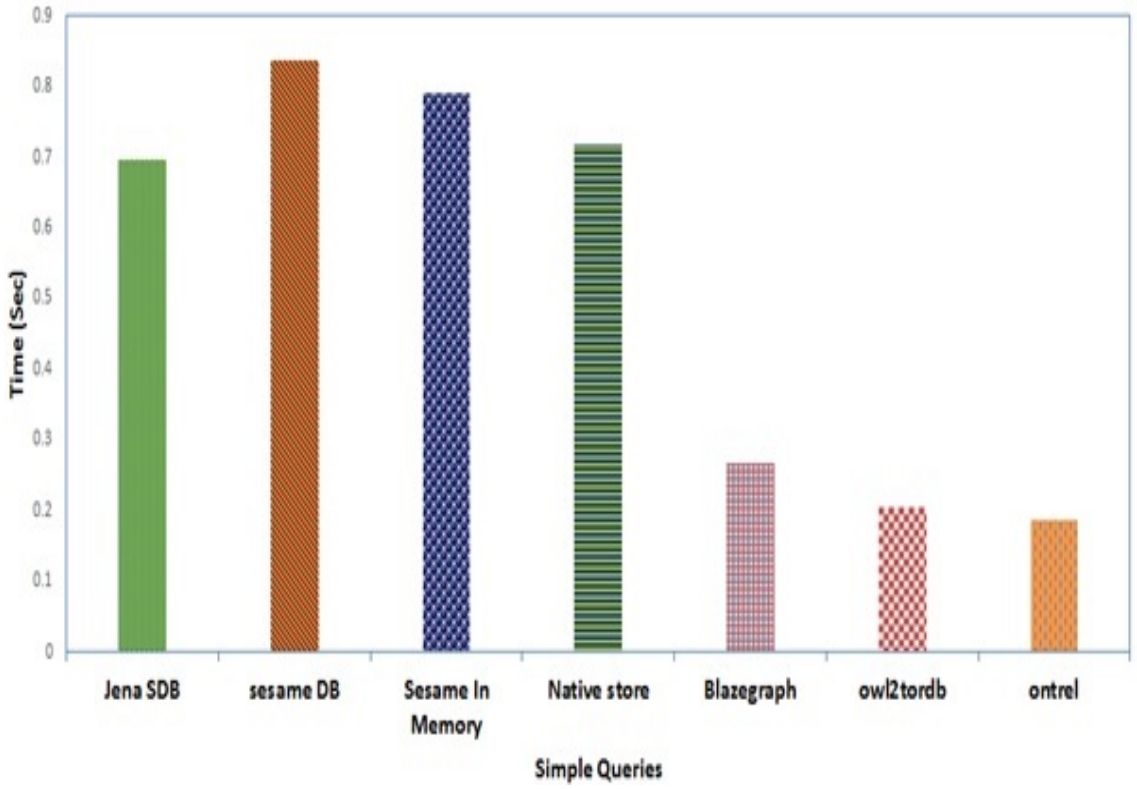

FIGURE 5.11: Average query response time of KBSP against Simple Queries on all the used datasets

SDB, OntRel and OWL2ToRDB) for OPQs is very poor over large size datasets. On the basis of the results, it is concluded that in memory systems are suitable for the domains where queries generate relatively small result size. Overall Blazegraph has shown stable performance except slight performance degradation on bushy pattern (CQ1) and long chain pattern (CQ2) queries.

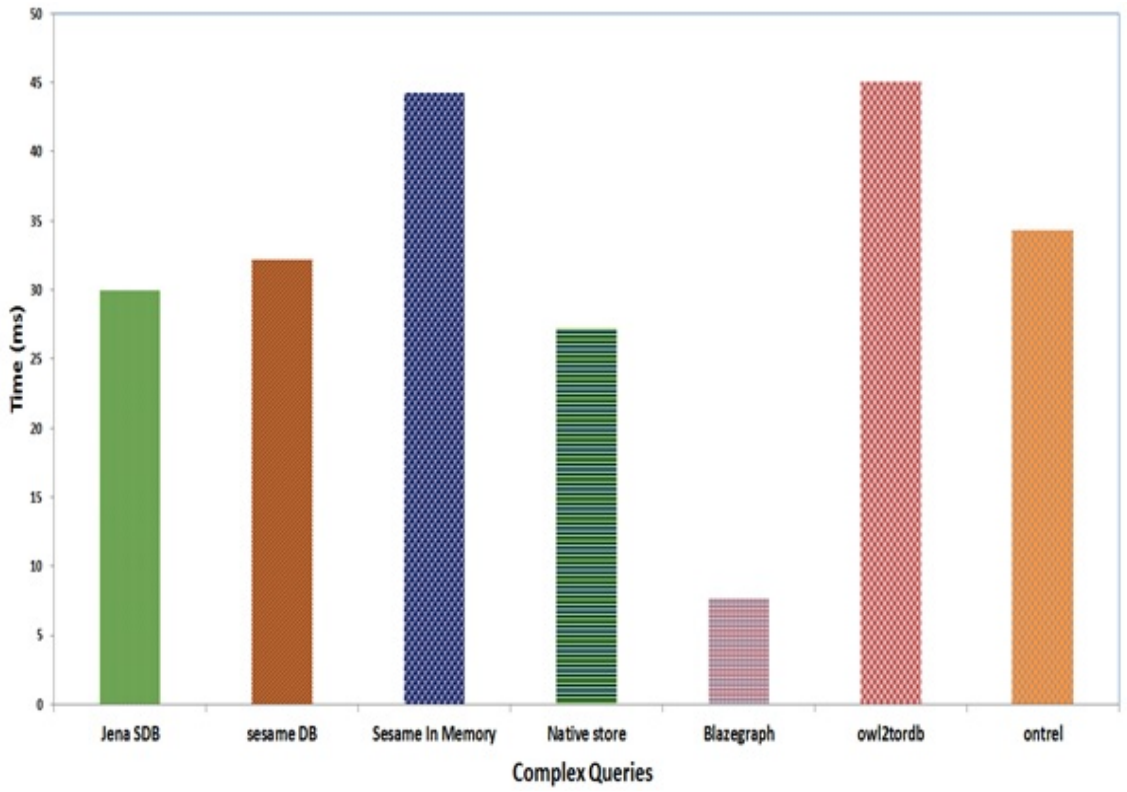

FIGURE 5.12: Average query response time of KBSP against Complex Queries on all the used datasets

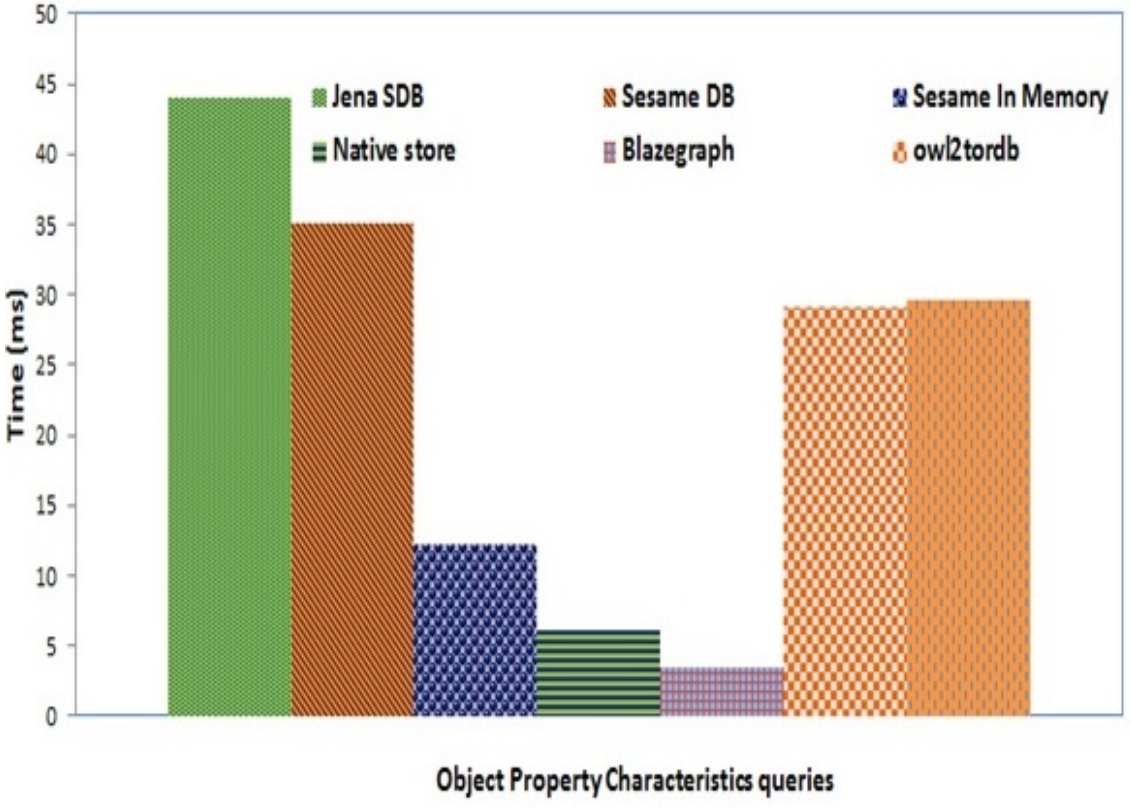

FIGURE 5.13: Average query response time of KBSP against Object property pattern Queries on all the used datasets

Chapter 6

CONCLUSION AND FUTURE WORK

This chapter summarizes our research work, and concludes the results and findings obtained from the present research. The research findings, which are drawn from the objectives, have been concluded and discussed. The present research, contributes in the creation of an evaluation benchmark that is capable to evaluate KBSPs against OWL semantics, especially OWL2 semantics. This proposal of creating the evaluation benchmark allows domain users to select suitable KBSPs based upon the performance on OWL2 semantics. This chapter comprises of the summary of research findings which fulfilled our objectives and answered the research questions raised in Chapter 1. This chapter also presents the limitations of this research and laydown the future research directions related to this research.

6.1 Summary of Research Findings

This section provides discussion of the research findings. The discussion draws the earlier identified research objectives. Based on these objectives, a number of research findings are summarized as follows:

- Objective 1: To inspect the existing evaluation benchmarks for missing OWL1.1 and OWL2 semantics.

- Findings: In order to check the provision of all the OWL (OWL1.1 and OWL2) semantics in the evaluation benchmarks, a literature review of the existing evaluation benchmark was performed. Upon the selection of relevant evaluation benchmarks for the KBPS, the inspection of the OWL constructs in the data schemas was performed. This task was done through manual inspection of the OWL constructs in the data schemas and compared with standard OWL list available at W3C. The limitations in the existing evaluation benchmarks was found that all the benchmarks data schemas lacks complete OWL2 semantics and few OWL1.1 semantics [Reference Chapter2: section 2.1-2.7]. The list of all the missing OWL constructs is available in Appendix-B. The complexity of data schema is important for determining the realistic performance of the KBSPs. For this purpose complexity of data schemas of all the evaluation benchmarks is computed. Edge to Node Ratio (EnR) is used as a method to compute the complexity of the data schema. The EnR is computed with and without OWL constructs used in the data schemas of the evaluation benchmarks. The findings show that the data schemas of the existing benchmarks are of simple structure and more inclined to hierarchal nature [Reference Chapter2: section 2.1-2.7].

  The datasets of the evaluation benchmarks are inspected for the coverage of OWL semantics. The methodology followed manual inspection of classes, object properties, and data properties in the dataset generators of the evaluation benchmarks. Against these classes and properties the usage of OWL constructs is collected and a list of distinct OWL constructs is prepared. This list is then compared with standard OWL list to determine the sufficient usage of OWL semantics in the datasets. The findings show that in addition to the inhabited flaws of the dataset generators [refer section 2.2 and 2.3], there is minimum usage of classes, object properties, and data properties in the dataset generators of the surveyed benchmarks. Accordingly, the coverage of OWL semantics in the produced datasets is also minimum. The non-availability of OWL2 constructs in the data schema are also missing in the datasets of the benchmarks [Reference Chapter2: section 2.1-2.7]. The

list of missing OWL constructs is given in the Appendix-B.

The benchmark queries are inspected for the sufficient set of OWL semantics for testing the KBSPs performance against those OWL semantics. For this purpose, queries set of the evaluation benchmarks are manually inspected to prepare a list of OWL constructs directly or indirectly used in the benchmark queries [Reference Chapter2: section 2.1-2.7]. The findings shows that majority of the benchmark queries involve subClassOf relations. The generic queries are limited for the class instances only. The generic level queries for the properties are missing in the benchmark queries. Like, data schemas and datasets, the benchmark queries misses OWL2 coverage. [Details of missing OWL constructs is given in Appendix-B].

- Objective 2: To enrich the components of the standard evaluation benchmark with the OWL and OWL2 semantics

- Findings: On the basis of identified research gap, an evaluation benchmark (OEB2) is proposed in the present work to provide complete coverage of OWL semantics in the data schema, dataset generator, and queries set. As a case study data schema of UOBM is used [Reference: chapter-3].

  First of all data schema of the proposed benchmark is enriched by adding the missing OWL constructs. This activity is performed in three steps. In the first step survey of domain ontologies is performed. In the second step WordNet senses are implemented on the existing classes and object properties of the proposed benchmark. In the third step pattern based queries are executed on the large dataset of the UOBM. With these steps data schema is able to provide complete coverage of OWL semantics [Reference chapter-3: section 3.1].

  The missing OWL semantics in the dataset generator are covered by adding all the axioms, which describe OWL assertions (i.e. facts about the individuals). By doing this, the dataset generator covers all the OWL2 assertions in the dataset. Further, the dataset generator is built with a dynamic approach to allow dataset generation for any domain [Reference chapter 3: section 3.2].

In order to provide more coverage of OWL semantics in the proposed benchmark queries, the queries set has been divided into three categories. The simple queries provides basis for queries which are meant to check the semantic preservation (i.e. OWL semantics). For example, checking all the subClassOf pairs, all the object properties, propertyChain axioms, etc. These type of queries explicitly provides the full coverage of OWL semantics. The two other type of queries include Complex queries and object property characterisitics pattern based queries. Both type of queries are generic in nature so indirect involvement of OWL semantics also increases the usage of OWL semantics in the benchmark queries [Reference chapter 3: section 3.3].

- Objective 3: To evaluate the proposed OWL2 KBSP benchmark.
- Findings: In order to validate the proposed benchmark for proof of concept of OWL2 usage in an evaluation benchmark, the proposed benchmark is evaluated against all the steps used for evaluating the existing benchmarks.

  The coverage of all the OWL (OWL1.1 & OWL2) semantics in the data schema of the proposed benchmark is checked and compared with standard OWL list and then compared with the data schemas of the existing evaluation benchmarks. The findings of some of the OWL constructs is shown in Figure 4.1 (a & b) and reports that the proposed benchmark has complete coverage of OWL semantics in the data schema. The finding also reports that instead of hierarchal organization the data schema is more association centric. [Reference Chapter 4].

  With reference to the complexity of the data schema, the complexity in the data schema is added by; a. Giving uniform coverage of all OWL constructs in the data schema which raise the EnR of each OWL constructs. b. More object properties are added in the data schema. c. Usage of all the object properties characteristics in the data schema. The findings shows that the proposed benchmark has higher EnR with and without OWL constructs as compared to the existing benchmarks. The finding concludes that the proposed data schema is more complex than LUBM, UOBM, and BSBM [Reference Chapter 4].

The dataset evaluation of the proposed benchmark is performed by checking the coverage of OWL2 semantics, property characteristics, usage of classes, object properties, and data properties in the dataset. The findings show that the proposed dataset covers all the assertion constructs like SameIndividual, DifferentIndividual, ObjectPropertyAssertion, Negative Data property assertion, Negative Object property assertion in the dataset. The usage of classes, object properties, and data properties is high in the proposed dataset. Further, all the object property characteristics are covered in the dataset generation process. This concludes the proposed benchmark dataset provides complete coverage of OWL2 semantics [Reference Chapter 4]. To evaluate the benchmark queries set, the direct and indirect usage of OWL constructs is checked in the proposed queries set. Due to generic nature of queries, and simple queries (specifically designed for checking the OWL construct for semantic preservation) the usage of OWL2 constructs in the proposed benchmark is high. [Reference Chapter 4].

- Objective 4: To evaluate the Knowledge based systems platforms [KBSPs] by using the proposed benchmark against the performance evaluation metric.

- Findings: For the evaluation of the proposed benchmark seven KBSPs were selected. The selection was made on the criteria to cover memory based systems, file based systems, graph based system, and relational database systems. Within the relational database, systems were selected on the basis of different relational schemas. These KBSPs include Sesame (in memory), Sesame (DB), Jena SDB, Blazegraph, RDF Native storage system, OntRel, and OWL2ToRDB. The performance metrics used for evaluation the KBSPs include load time and query response time. Three type of queries set [Reference Chapter 3] are run over three different size dataset. For load time, the findings of the proposed benchmark are:

  - In memory, graph based, and file based systems perform better in loading the datasets of 2.5 million triples (i.e. 2400K).

- The behavior of the KBSPs with relational database storage systems varies on different size datasets. These KBSPs takes longer time to load the datasets.
- The KBSPs with less schema dependency takes less time to transform the datasets into the destination storage model [Reference Chapter 5: section 5.1].

For query response time, the findings of the proposed benchmark include:

- For simple queries, the behavior of query response time of all the KBSPs remains similar on all the datasets because of the limited result set and queries type.
- For complex queries, the behavior of the KBSPs varies on different datasets.
  - In bushy pattern query, the response time of graph based KBSP is better than the other KBSPs against all the datasets.
  - The KBSPs with relational database storage systems (i.e. triple store) shows better response time on long chain query.
  - The relational database storage systems using class decomposition schema or metadata schema have high query response time.
  - The query response time for high selectivity query is high in the memory based systems against large datasets (i.e. 240 K, 2400K).
- For object properties pattern queries, the findings shows that;
- The query response time of all the KBSPs for InverseOf query remains similar for all the datasets.
- The KBSPs using class decomposition schema or metadata schema have high query response time on InverseFunctional query against different datasets.
- The graph based KBSP have minimum response time against all type of pattern queries on all the datasets.

  - In memory KBSP response time increases on all type of patterned queries as the dataset size grows.

6.2 Research Contribution

The present research work provides contribution as follows;

- This research provides mechanism for the evaluation of the KBSP benchmark. The mechanism comprises of four steps upon which the evaluation benchmarks are tested. The results concludes the research gap for the need of a new benchmark.
- The present work proposed an evaluation benchmark for evaluating KBSPs with respect to use of OWL2 semantics. The use of OWL2 semantics in the proposed benchmark (OEB2 - OWL2 Evaluation Benchmark) is deemed to be the novelty of the present work. The benchmark follows the standard building blocks of an evaluation benchmark.
- A dynamic approach is adopted to generate scalable datasets for any given domain.
- State of the art KBSPs (belonging to memory, persistence storage, relational database and graph based) are evaluated against a set of performance metrics i.e. load time, response time and scalability.

6.3 Limitations

The focus of overall research is the evaluation of KBSP benchmarks for checking the coverage of OWL semantics in the benchmark elements. On the basis of final outcome of evaluation of the existing benchmarks,construction of the proposed benchmark and its analysis, the KBSPs were evaluated by the proposed benchmark. In performing the research, the present work has exhibits some limitations, which include:

- OEB2 uses three different size datasets (i.e. 24K, 240K, 2400K triples). The maximum number of triples size is 2457600. During the evaluation of KBSPs it was observed that the performance of different KBSPs like in-memory may behave poor on the larger size dataset.But in present research, the dataset more than 2400K has not been used.

- In the present research, EnR has been computed for OWL construct by assigning each OWL semantics used in the data schema an equal weight. Assigning different weights to OWL semantics based upon their impact on the data schema may revise the EnR values for OWL constructs. which may lead towards variation in the semantic complexity of the data schema.

- Object property pattern queries have been used to obtain the OWL object property characteristics for the semantic enrichment of the data schema and coverage of OWL semantics. However, it is not necessary that the object property characteristics obtained represent the similar kind of characteristics in the real domain.

6.4 Future Direction

The future directions to this work includes the following areas.

- Evaluate the KBSPs against multiple domains ontologies containing OWL and OWL2 semantics. This will provide a more exhaustive evaluation of KBSPs for different real world domains.

- Detailed evaluation of the KBSPs using OEB2 is required to validate the soundness, completeness and semantic preservation. Another important evaluation that evaluates the reasoning and inference capabilities of the KBSPs with OWL2 semantics is a demanding area of research.

- The construction of a recommender system for the enrichment and enhancement of the input data schema with OWL semantics.

- Using dynamic approach, datasets for different domains may be generated for checking the behaviour KBSPs on different domains.
- Further study of object property characteristics may lead towards identifying more semantic relationships.
- The adoption of a systematic approach in the present research for the evaluation of KBSP benchmarks provides basis of an automated solution with single point of interface. Where user may select evaluation benchmarks,KBSPs, performance metrics, data schema, user specified dataset.

Bibliography

[1] RDF4J, "Eclipse rdf4j- a java framework for rdf," *rdf4j.org, Accessed on: June 08,2014*, 2014.

[2] Blazegraph, "Blazegrpah download- graph database & application dowload," *https://www.blazegraph.com/, Accessed on: April 13,2014*, 2014.

[3] J. SDB, "Apache jena - sdb - persistent triple stores using relational databases," *https://jena.apache.org/documentation/sdb/index.html, Accessed on: January 8,2014*, 2014.

[4] A. Khalid, S. A. H. Shah, and M. A. Qadir, "OntRel: An Ontology Indexer to store OWL-DL Ontologies and its Instances," in *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*, 2009, pp. 478–483.

[5] B. Lauser, M. Sini, A. Liang, and J. Keizer, "From AGROVOC to the Agricultural Ontology Service/Concept Server. An OWL model for creating ontologies in the agricultural domain," *Dublin Core Conference*, 2006. [Online]. Available: http://eprints.rclis.org/21109/

[6] GeneOntology, "Gene ontology consortium," *http://www.geneontology.org/, Accessed on: January 8,2017*, 2017.

[7] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "Owl 2: The next step for owl," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 4, pp. 309–322, 2008.

[8] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2, pp. 158–182, 2005.

[9] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu, "Towards a complete OWL ontology benchmark," in *European Semantic Web Conference*, 2006, pp. 125–139.

[10] B. Christian and S. Andreas, "The Berlin SPARQL Benchmark," *International Journal on Semantic Web & Information Systems*, vol. 5, no. 2, pp. 1–24, 2009.

[11] M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo, "DBpedia SPARQL benchmark–performance assessment with real queries on real data," in *International Semantic Web Conference*, 2011, pp. 454–469.

[12] M. Schmidt, T. Hornung, and G. Lausen, "SP2Bench: a SPARQL performance benchmark," *International Conference ...*, 2009. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/4812405/

[13] V. Link, S. Lohmann, and F. Haag, "OntoBench: Generating Custom OWL 2 Benchmark Ontologies," *International Semantic Web Conference*, 2016. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-46547-0{_}13

[14] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "OWL 2 web ontology language primer," *W3C recommendation*, vol. 27, no. 1, p. 123, 2009.

[15] M. Jazayeri, A. Ran, and F. V. D. Linden, "Software architecture for product families: principles and practice," 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=337672

[16] M. Jones, *Big data in cognitive science*, 2016. [Online]. Available: https://books.google.com.pk/books?hl=en{&}lr={&}id=qDclDwAAQBAJ{&}oi=fnd{&}pg=PP1{&}dq={%}5B16{%}5D.{%}09Jones,+M.+N.+(2016)

.+Big+Data+in+Cognitive+Science.+Psychology{&}ots=a7NTiZyCnq{&}sig=MICtjWijiDafDCvTV2uuLX{_}j{_}Qo

[17] P. Boncz, I. Fundulaki, A. Gubichev, and J. Larriba-Pey, "The linked data benchmark council project," *Datenbank-*, 2013. [Online]. Available: http://link.springer.com/article/10.1007/s13222-013-0125-y

[18] A. Khalid, S. Shah, and M. Qadir, "OntRel: An ontology indexer to store owl-dl ontologies and its instances," *Soft Computing and Pattern*, 2009. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/5368643/

[19] E. Vyšniauskas, L. Nemurait, and B. Paradauskas, "Preserving semantics of OWL 2 ontologies in relational databases using hybrid approach," *information Technology and*, 2012. [Online]. Available: http://www.itc.ktu.lt/index.php/ITC/article/view/833

[20] S. Wang, Y. Guo, A. Qasem, and J. Heflin, "Rapid benchmarking for semantic web knowledge base systems," *The Semantic WebISWC 2005*, 2005. [Online]. Available: http://link.springer.com/content/pdf/10.1007/11574620.pdf{#}page=780

[21] S. Jean, L. Bellatreche, G. Fokou, M. Baron, and S. Khouri, "Ontodbench: Ontology-based database benchmark," *28e journ{é}es Bases de Donn{é}es Avanc{é}es (BDA)*, 2012. [Online]. Available: http://bda2012.isima.fr/files/paper64.pdf

[22] P. Bellini, P. Nesi, and G. Pantaleo, "Benchmarking rdf stores for smart city services," in *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. IEEE, 2015, pp. 46–49.

[23] D. Lanti, M. Rezk, M. Slusnys, G. Xiao, and D. Calvanese, "The NPD benchmark for OBDA systems," in *Proc. of the 10th Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2014)*, vol. 1261, 2014, pp. 3–18.

[24] A. S. Butt and S. Khan, "Scalability and performance evaluation of semantic web databases," *Arabian Journal for Science and Engineering*, vol. 39, no. 3, pp. 1805–1823, 2014.

[25] L. Bühmann and J. Lehmann, "Universal OWL axiom enrichment for large knowledge bases," *Knowledge Engineering and Knowledge*, 2012. [Online]. Available: http://link.springer.com/content/pdf/10.1007/978-3-642-33876-2.pdf{#}page=74

[26] P. Hitzler, M. Krötzsch, B. Parsia, and P. Patel-Schneider, "OWL 2 web ontology language primer," *recommendation*, 2009. [Online]. Available: https://www.w3.org/TR/2009/PR-owl2-primer-20090922/all.pdf

[27] C. Golbreich, E. Wallace, and P. Patel-Schneider, "OWL 2 Web Ontology Language new features and rationale," *2009) http://www. w3. org/TR ...*, 2009. [Online]. Available: http://www.w3.org/TR/2008/WD-owl2-new-features-20081202/all.pdf

[28] B. Motik, Y. Nenov, R. Piro, I. Horrocks, and D. Olteanu, "Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems." *AAAI*, 2014. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/download/8505/8410

[29] C. Bizer, J. Lehmann, G. Kobilarov, and S. Auer, "DBpedia-A crystallization point for the Web of Data," *Web Semantics: science*, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570826809000225

[30] FactPages, "Fact pages norweigien petroleum department," *http://factpages.npd.no/factpages/, Accessed on: March 28,2018.*

[31] M. Morsey, J. Lehmann, and S. Auer, "DBpedia SPARQL benchmarkperformance assessment with real queries on real data," *The Semantic Web*, 2011. [Online]. Available: http://www.springerlink.com/index/CV8812100X36J527.pdf

[32] R. Mihalcea and D. Moldovan, "Semantic indexing using WordNet senses," *language processing and information retrieval ...*, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1117760

[33] H. Kitagawa, Y. Ishikawa, W. Li, and C. Watanabe, *Database Systems for Advanced Applications: 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1-4, 2010, Proceedings.* Springer, 2010, vol. 5981.

[34] D. C. Faye, O. Cure, and G. Blin, "A survey of rdf storage approaches," *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, vol. 15, pp. 11–35, 2012.

[35] L. Fopa, F. Jouanot, A. Termier, M. Tchuente, and O. Iegorov, "Benchmarking of triple stores scalability for mpsoc trace analysis," in *2ndt International workshop on Benchmarking RDF Systems (BeRSys 2014)*, 2014.

[36] J. Frey, K. Müller, S. Hellmann, E. Rahm, and M.-E. Vidal, "Evaluation of metadata representations in rdf stores."

APPENDICES

APPENDIX A

Lehigh University Benchmark queries [8]

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

PREFIX ub: <http:/www.lehigh.edu/2004/0401/univ-bench.owl >

Query1

SELECT ?X

WHERE

{?X rdf:type ub:GraduateStudent . ?X ub:takesCourse

http://www.Department0.University0.edu/GraduateCourse0}

Query2

SELECT ?X, ?Y, ?Z

WHERE

{?X rdf:type ub:GraduateStudent . ?Y rdf:type ub:University .

?Z rdf:type ub:Department . ?X ub:memberOf ?Z .

?Z ub:subOrganizationOf ?Y . ?X ub:undergraduateDegreeFrom ?Y}

Query3

SELECT ?X

WHERE

{?X rdf:type ub:Publication . ?X ub:publicationAuthor

http://www.Department0.University0.edu/AssistantProfessor0}

```
# Query4
SELECT ?X, ?Y1, ?Y2, ?Y3
WHERE
{?X rdf:type ub:Professor .
?X ub:worksFor <http:/www.Department0.University0.edu >.
?X ub:name ?Y1 . ?X ub:emailAddress ?Y2 . ?X ub:telephone ?Y3}

# Query5
PREFIX ub: <http:/www.lehigh.edu/2004/0401/univ-bench.owl >
SELECT ?X
WHERE
{?X rdf:type ub:Person .
?X ub:memberOf <http:/www.Department0.University0.edu >}

# Query6
SELECT ?X WHERE {?X rdf:type ub:Student}

# Query7
SELECT ?X, ?Y
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Course .
?X ub:takesCourse ?Y .
<http:/www.Department0.University0.edu/AssociateProfessor0 >,
ub:teacherOf, ?Y}
# Query8
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student . ?Y rdf:type ub:Department .
?X ub:memberOf ?Y . ?Y ub:subOrganizationOf
```

<http:/www.University0.edu >.
?X ub:emailAddress ?Z}

Query9
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student . ?Y rdf:type ub:Faculty .
?Z rdf:type ub:Course . ?X ub:advisor ?Y .
?Y ub:teacherOf ?Z . ?X ub:takesCourse ?Z}

Query10
SELECT ?X
WHERE
{?X rdf:type ub:Student . ?X ub:takesCourse
<http:/www.Department0.University0.edu/GraduateCourse0 >}

Query11
SELECT ?X
WHERE
{?X rdf:type ub:ResearchGroup . ?X ub:subOrganizationOf
<http:/www.University0.edu>}
Query12
SELECT ?X, ?Y
WHERE {?X rdf:type ub:Chair . ?Y rdf:type ub:Department .
?X ub:worksFor ?Y . ?Y ub:subOrganizationOf <http:/www.University0.edu>}
Query13
SELECT ?X
WHERE
{?X rdf:type ub:Person . <http:/www.University0.edu>ub:hasAlumnus ?X}
Query14
SELECT ?X

WHERE {?X rdf:type ub:UndergraduateStudent}

University Ontology Benchmark queries [9]

Prefix benchmark-dl: <http:/semantics.crl.ibm.com/univ-bench-dl.owl >

#Query1

SELECT DISTINCT X

from

{X} rdf:type {benchmark-dl:UndergraduateStudent}

benchmark-dl:takesCourse {<http:/www.Department0.University0.edu/Course0>}

#Query2

SELECT DISTINCT X

from

{X} rdf:type {benchmark-dl:Employee}

#Query3

SELECT DISTINCT X

from

{X} rdf:type {benchmark-dl:Student}

benchmark-dl:isMemberOf {<http:/www.Department0.University0.edu>}

#Query4

SELECT DISTINCT X

from

{Y} rdf:type {benchmark-dl:Faculty}

benchmark-dl:isMemberOf {<http:/www.Department0.University0.edu>},

{X} rdf:type {benchmark-dl:Publication}

benchmark-dl:publicationAuthor {Y}

#Query5
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:ResearchGroup}
benchmark-dl:subOrganizationOf {<http:/www.University0.edu>}

#Query6
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:Person},
{<http:/www.University0.edu>} benchmark-dl:hasAlumnus {X}
#Query7
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:Person}
benchmark-dl:hasSameHomeTownWith {<http:/www.Department0
.University0.edu/FullProfessor0>}

#Query8
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:SportsLover},
{<http:/www.Department0.University0.edu>} benchmark-dl:hasMember {X}

#Query9
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:GraduateCourse}
benchmark-dl:isTaughtBy {Y},

{Y} benchmark-dl:isMemberOf {Z},
{Z} benchmark-dl:subOrganizationOf {<http:/www.University0.edu>}

#Query10
SELECT DISTINCT X
from
{X} benchmark-dl:isFriendOf {<http:/www.Department0.
University0.edu/FullProfessor0>}

#Query11
http://www.Department0.University0.edu
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:Person}
benchmark-dl:like {Y},
{Z} rdf:type {benchmark-dl:Chair}
benchmark-dl:isHeadOf {<http:/www.Department0.University0.edu>}
benchmark-dl:like {Y}

#Query12
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:Student}
benchmark-dl:takesCourse {Y},
{Y} benchmark-dl:isTaughtBy {<http:/www.Department0.
University0.edu/FullProfessor0>}

#Query13
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:PeopleWithHobby}

benchmark-dl:isMemberOf {<http:/www.Department0.University0.edu>}

#Query14
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:Woman}
rdf:type {benchmark-dl:Student}
benchmark-dl:isMemberOf {Y},
{Y} benchmark-dl:subOrganizationOf {<http:/www.University0.edu>}

#Query15
SELECT DISTINCT X
from
{X} rdf:type {benchmark-dl:PeopleWithManyHobbies}
benchmark-dl:isMemberOf {<http:/www.Department0.University0.edu>}

#Query16
SELECT DISTINCT X
from
{X} rdf:type{benchmark-dl:NonScienceStudent}
benchmark-dl:isMemberOf {<http:/www.University0.edu>}

APPENDIX B

OWL1.1 and OWL2 constructs used by the benchmarks

Benchmarks			LUBM			UOBM			OntoBench		
			Benchmark elements								
OWL1.1 & OWL2	Constructs & Axioms	OWL Language	DSH	DS	QS	DSH	DS	QS	DSH	DS	QS
Classes and enumerations	owl:Class	OWL1.1	Y	Y	Y	Y	Y	Y	Y		
	owl:ComplementOf	OWL1.1	N	N	N	Y	Y	Y	Y		
	owl:IntersectionOf	OWL1.1	Y	Y	N	Y	Y	N	Y		
	owl:Nothing	OWL1.1	N	N	N	N	N	N	Y		
	owl:one of	OWL1.1	N	N	N	N	N	N	Y		
	owl:Thing	OWL1.1	Y	Y	N	Y	N	Y	Y		
	owl:unionOf	OWL1.1	N	N	N	Y	N	N	Y		
Class Axioms	owl:AllDisjointClasses	OWL2.0	N	N	N	N	N	N	Y		
	owl:disjointUnionOf	OWL2.0	N	N	N	N	N	N	Y		
	owl:disjointWith	OWL1.1	N	N	N	Y	Y	Y	Y		
	owl:EquivalentClass	OWL1.1	N	N	N	Y	Y	N	Y		
	rdfs:subClassOf	OWL1.1	Y	Y	Y	Y	Y	Y	Y		

Table 6.3 continued from previous page

Object property characteristics	owl:AllDisjointProperties	OWL2.0	N	N	N	N	N	N	Y
	owl:AsymmetricProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:bottomObjectProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:equivalentProperty	OWL1.1	N	N	N	Y	Y	Y	Y
	owl:FunctionalProperty	OWL1.1	N	N	N	Y	Y	Y	Y
	owl:InverseFunctionalProperty	OWL1.1	N	N	N	Y	Y	Y	Y
	owl:inverseOf	OWL1.1	Y	Y	Y	Y	Y	Y	Y
	owl:IrreflexiveProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:ObjectProperty	OWL1.1	Y	Y	Y	Y	N	Y	Y
	owl:PropertyChainAxiom	OWL2.0	N	N	N	N	N	N	Y
	owl:propertyDisjointWith	OWL2.0	N	N	N	N	N	N	Y
	owl:ReflexiveProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:SymmetricProperty	OWL1.1	N	N	N	Y	Y	Y	Y
	owl:topObjectProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:TransitiveProperty	OWL1.1	Y	Y	Y	Y	Y	Y	Y
	rdfs:domain	OWL1.1	Y	Y	Y	Y	Y	Y	Y
	rdfs:range	OWL1.1	Y	Y	Y	Y	Y	Y	Y

Table 6.3 continued from previous page

	rdfs:subPropertyOf	OWL1.1	Y	Y	Y	Y	Y	Y	Y
Restrictions (object properties)	owl:allValuesFrom	OWL1.1	N	N	N	Y	N	Y	Y
	owl:cardinality	OWL1.1	N	N	N	Y	N	N	Y
	owl:hasSelf	OWL1.1	N	N	N	Y	N	Y	Y
	owl:hasValue	OWL1.1	N	N	N	N	N	N	Y
	owl:maxCardinality	OWL1.1	N	N	N	Y	N	Y	Y
	owl:maxQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:minCardinality	OWL1.1	N	N	N	Y	Y	Y	Y
	owl:minQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:qualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:someValuesFrom	OWL1.1	Y	Y	Y	Y	Y	Y	Y
Restrictions Data property	owl:allValuesFrom	OWL2.0	N	N	N	N	N	N	Y
	owl:cardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:hasValue	OWL2.0	N	N	N	N	N	N	Y
	owl:maxCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:maxQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:minCardinality	OWL2.0	N	N	N	N	N	N	Y

Table 6.3 continued from previous page

	owl:minQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:qualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:someValuesFrom	OWL2.0	N	N	N	N	N	N	Y
	owl:AllDisjointProperties	OWL2.0	N	N	N	N	N	N	Y
	owl:bottomDataProperty	OWL2.0	N	N	N	N	N	N	Y
	owl:DatatypeProperty	OWL1.1	Y	Y	Y	Y	Y	N	Y
	owl:equivalentProperty	OWL1.1	N	N	N	N	N	N	Y
	owl:FunctionalProperty	OWL1.1	N	N	N	N	N	N	Y
Data	owl:propertyDisjointWith	OWL1.1	N	N	N	N	N	N	Y
properties	owl:topDataProperty	OWL2.0	N	N	N	N	N	N	Y
and axioms	rdfs:domain	OWL1.1	Y	Y	Y	Y	Y	Y	Y
	rdfs:range	OWL1.1	Y	Y	N	Y	Y	Y	Y
	rdfs:subPropertyOf	OWL1.1	N	N	N	N	N	N	Y
	owl:allValuesFrom	OWL2.0	N	N	N	N	N	N	Y
	owl:cardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:hasValue	OWL1.1	N	N	N	Y	Y	N	Y
Restrictions(Data	owl:maxCardinality	OWL2.0	N	N	N	N	N	N	Y
Property)									

Table 6.3 continued from previous page

	owl:maxQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:minCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:minQualifiedCardinality	OWL2.0	N	N	N	N	N	N	Y
	owl:someValuesFrom	OWL2.0	N	N	N	N	N	N	Y
Assertions	owl:AllDifferent	OWL1.1	N	N	N	N	N	N	Y
	owl:hasKey	OWL2.0	N	N	N	N	N	N	Y
	owl:NegativePropertyAssertion	OWL2.0	N	N	N	N	N	N	Y
	owl:sameAs	OWL1.1	N	N	N	N	N	N	Y
OWL individual	Owl:anonymous Individual	OWL2.0	N	N	N	N	N	N	N
	Owl:NamedIndividual	OWL2.0	Y	Y	N	Y	N	N	Y

APPENDIX C

The details of datasets, technical report is given at:
https:/github.com/azeemabbas/oeb2Benchmark

APPENDIX D

Evaluation Dataset and Graphs

The objective of this research is the performance evaluation of Knowledge Based Systems Platform (KBSP) on OWL2 semantics. This requires dataset of large size with OWL2 semantics for measuring the performance of KBSP. Therefore, we are interested in using a large and OWL2 semantic enabled dataset. There are quite number of dataset available for the testing purpose like LUBM dataset [8], DBpedia [31], SP2B [12] and UOBM [9]. These datasets fulfils the scalability factor but lacks OWL2 semantics. The UOBM dataset [9] is chosen in our benchmark because its dataset covers most of the OWL semantics in comparison of the surveyed evaluation benchmarks. The dataset is provided with the UOBM benchmark [9]. With the help of UOBM dataset generator, user specified dataset is created. The structure of the dataset belongs to university domain. The missing OWL2 semantics have been added in the dataset.

Benchmark Dataset

The benchmark dataset is analyzed using Protg plug-in.

Statistics of the proposed university ontology dataset

A detail analysis of the dataset provides the characteristics of evaluated data that are presented in the Table 7.4.

Namespaces

The namespaces used the dataset are mentioned in the Table 7.5.

Dataset Scaling

We perform our evaluation on three different sizes of university dataset. These are Dataset 1, Dataset 2, and Dataset3 contains 24K, 240K and 2400K triples respectively.

TABLE 6.4: Summary Statistics of proposed university ontology dataset

Characteristics	Total Number
Total Number of Triples	2727936
Total Nodes	26878
Total Number of Classes	131
Total Number of Object Properties	63
Total Properties characteristics used	09

TABLE 6.5: Prefix to URI mapping

Prefix	URI
OEB2	http:/semantics.crl.ibm.com/univBench.owl
rdf	http:/www.w3.org/1999/02/22-rdf-syntax-ns
owl	http:/www.w3.org/2002/07/owl
rdfs	http:/www.w3.org/2000/01/rdf-schema

Dataset Cleaner

In the proposed dataset generator, the redundant triples are identified and eliminated by identifying all the triples, which are associated with object properties characteristics. So that single copy of the triple is retained as shown in the following example.

<owl:NamedIndividual rdf:about=http:www.Department10.University0.edu
/AssistantProfessor0 >
<isFriendOf rdf:resource=http://www.Department10.University0.edu
/AssistantProfessor8 >
<owl:NamedIndividual rdf:about=http:www.Department10.University0.edu
/AssistantProfessor8 >
<isFriendOf rdf:resource=http://www.Department10.University0.edu
/AssistantProfessor0 >