ISSN No. 0976-5697

Volume 5, No. 3, March-April 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

"Comparison of Rule based Semantic Web Reasoners for Ontologies in OWL2 Profile"

Deepika Chaudhary*
Associate Professor MCA,
Chitkara Univeristy Punjab,India

Dr. Archana Mantri
Pro VC,
Chitkara University, Punjab, India

Dr. D.P.Kothari
FNAE,FNASc, Fellow-IEEE
Director - Research
MVSR Engineering College , Hyderabad, India

Abstract: The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web[1]. In semantic web, Knowledge can be encoded using a formal language termed as OWL (Ontology Web Language). To keep this language alive Progression of OWL2 from OWL is one such major step. OWL2 is a strong Modeling language and extends the W3C OWL with a small but useful set of features. The objective of this study is an attempt to highlight those features and to provide and indepth analysis on:

a. Introduction to three profiles of OWL2 with a particular focus on how these profiles differ and why the differences are important.
b. Comparison of OWL Profiles on the basis of Usage, Reasoning Time, Complexity and Algorithms.
c. Reasoning in OWL RL Profile and comparison of Reasoners in OWL RL Profiles.

Keywords: OWL – Ontology Web Language, OWL axioms – Inference rules

I. INTRODUCTION

OWL is a Ontology ('schema') language for semantic web which refers to a description of knowledge about a particular domain, basically referred for two main purposes i.e data modeling and automated reasoning. *Ontology* is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related. Now the question here arises why develop an Ontology? The possible answers to this question can be: To share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, make domain assumptions explicit, separate domain knowledge from the operational knowledge, analyze domain knowledge. Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies (Musen 1992; Gruber 1993). For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites.

The agents can use this aggregated information to answer user queries or as input data to other applications. An ontology is a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions))[2]. An ontology together with a set of individual instances of classes constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins. Classes are the focus of most ontologies. Classes describe concepts in the domain. For example, a class of wines represents all wines. In practical terms, developing an ontology includes: defining classes in the ontology, arranging the classes in a taxonomic (subclass–superclass) hierarchy, defining slots and describing allowed values for these slots, filling in the values for slots for instances. We can then create a knowledge base by defining individual instances of these classes filling in specific slot value information and additional slot restrictions An *OWL ontology* may include descriptions of classes, properties and their instances[3]. It also refers to a formal way of writing machine readable content in the form of OWL axioms(rules of inference). The OWL axioms allow us to state a sub class relation which can be understood by an OWL reasoner i.e a subclass axiom and equivalent Class axiom.

The W3C standards specifies some characteristics for defining an Ontology.

(a). Ontologies have a well defined syntax
(b). A formal Semantics
(c). Convenience of expression
(d). Efficient reasoning support
(e). Sufficient expressive power.

Out of which a formal semantics is of utmost importance. A formal Semantics describes the meaning of knowledge precisely and is a prerequisite for effective reasoning. There are different form for specifying semantics

a) ***Direct Semantics***- The meaning of OWL axioms is derived by directly relating them to Description Logic. Description logics (DL) are logics serving primarily for formal description of concepts and roles (relations). Semantically they are found on predicate logic, but their language is formed so that it would be enough for practical modeling purposes and also so that the logic would have good computational properties such as decidability. Knowledge representation system based on DLs consists of two components - TBox and ABox. The TBox describes *terminology*, i.e., the ontology in the form of concepts and roles definitions, while the ABox contains *assertions* about individuals using the terms from the ontology. Concepts describe sets of individuals, roles describe relations between individuals[4].

b) ***RDF Based Semantics***– In this the axioms are translated first in the form of Directed Graphs(RDF) and then reasoned upon. Resource Description Framework (RDF) is a framework for representing information about resources in a graph form. Since it was primarily intended for representing metadata about WWW resources, it is built around resources with URI. Information is represented by triples *subject-predicate-object* in RDF.

A. Why OWL Profiles?:

Profiles in context of OWL can be defined as a sub-language (syntactic subsets) that can offer important advantages in particular application scenarios. The first version of OWL created a profile called OWL Lite which tried to restrict the features of OWL in order to make reasoning easier. However, the goal was not achieved. OWL 2 defines three new profiles or sub-languages that offer important advantages depending on your application scenario: OWL 2 EL, OWL 2 QL and OWL 2 RL. Given that reasoning in OWL 2 is so hard, each of these profiles tries to find a sweet spot for particular application scenarios by trading off the expressiveness (what you can express in the ontology) in order to gain the possibility of creating efficient algorithms for reasoning. Three different profiles are defined: OWL 2 EL, OWL 2 QL, and OWL 2 RL. Each profile is defined as a *syntactic restriction* of the OWL 2 Structural Specification, i.e., as a subset of the structural elements that can be used in a conforming ontology, and each is more restrictive than OWL DL.

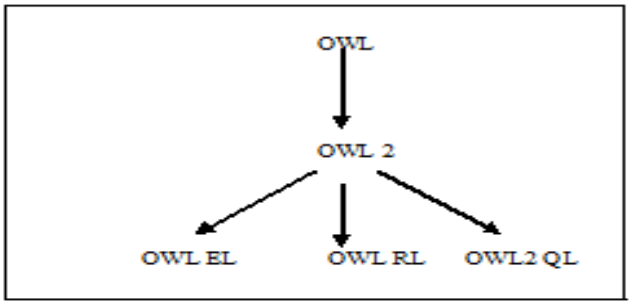
B. OWL2 Profiles:

a. ***OWL2 EL*** – This profile is particularly useful in applications employing ontologies that contain very large number of properties and classes for eg in biomedical ontologies.Gene Ontology is an ontology that describes genes and gene properties with more than 25k classes while SNOMED-CT is an ontology of clinical terms with over 500k classes. With this profile, classes can be defined with complex descriptions such as defining a class in terms of the existence of something else[5].

b. ***OWL 2 RL***- is tailored for applications that want to describe rules in ontologies. This profile is ideal if you already have RDF data and you want to implement your business logic in rules (if/then). OWL 2 RL runs efficiently on business rule engines, such as Drools. Therefore, OWL 2 RL is basically a rule language (hence the RL).This profile is used for those applications where scalable reasoning is required but without sacrificing for expressive power for eg. While reasoning with web data. It is geared towards running efficiently on traditional business rule engine.

c. ***OWL 2 QL***- is tailored for applications that want to reason on top of very large volumes of data. The motivation for this profile was to be able to keep data in a relational database and allow reasoning to be translated into queries on the database (hence the QL). In order for reasoning to be translated into a query, the expressivity of QL is a bit restricted. This profile can express conceptual models such as UML class diagrams and ER diagrams and also define hierarchies between classes and properties and inverse properties[6].

Any OWL 2 EL, QL or RL ontology is, of course, also an OWL 2 ontology and can be interpreted using either the Direct or RDF-Based Semantics. When using OWL 2 RL, a rule-based implementation can operate directly on RDF triples and so can be applied to an arbitrary RDF graph, i.e., to any OWL 2 ontology. In this case, reasoning will always be *sound* (that is, only correct answers to queries will be computed), but it may not be *complete* (that is, it is not guaranteed that all correct answers to queries will be computed). Theorem PR1 of the Profiles document states, however, that (in general) when the ontology is consistent with the structural definition of OWL 2 RL, a suitable rule-based implementation performing ground atomic queries will be both sound and complete[4].Fig 1 shows the relation between OWL1 and OWL2 which commonly is a fragment or a sub language of OWL, that trades expressive power for efficiency in reasoning and has three different profiles, each of which achieves efficiency in a different way and is useful in different application scenario.



e

Figure 1 Transition from OWL1 To OWL2 Profile

Below we present the comparison of 3 Profiles of OWL language on the basis of their usage and various complexity parameters.

Table: 1

Language	Application/Usage	Reasoners Required	Reasoning Complexity			
			Reasoning Problems	Taxanomic	Data	Query
OWL2 EL	Ontologies that contain large number of properties and classes.	Dedicated and highly scalable reasoners. Elk,pellet, CEL	Ontolology Consistency, Class expression, Satifiability, Subsumption, Intsance Checking	PTime Complete	PTime Complete	Not Applicable
			Conjuctive Query Answering	PTime Complete	PTime Complete	NP-Complete
OWL2 RL	Ontologies that require scalable reasoning without sacrificing too much expressive power	Reasoner based on first order implication rules. OWLIM,OWLRM,Jena.An attractive feature of OWL2 RL is that reasoning is polynomial with respect to size of ontology.	Ontolology Consistency, Class Expression, Satifiability, Subsumption, Intsance Checking	PTime Complete	PTime Complete	Not Applicable
			Conjuctive Query Answering	PTime Complete	NP-Complete	NP-Complete
OWL2 QL	The ontologies that uses very large number of instance data and where query answering is the most important reasoning task.	Reasoners efficient in conjuctive query answering OWLgres,Quest.	Ontolology Consistency, Class expression, Satifiability, Subsumption, Intsance Checking	IN AC°	Not Applicable	NLOGSPACE Complete
			Conjuctive Query Answering	IN AC°	NP-Complete	NP-Complete

II. OWL2 RL PROFILE LANGUAGE AND REASONING

The OWL2 profile specification defines OWL2 RL as "aimed at applications that require scalable reasoning without sacrificing too much expressive power"[6]. The profile is designed to work on rule engines and for this the specifications are provided in RDF Based semantics which can be applied on to RDF Graphs.OWL2 RL can therefore be divided in two ways-:

a) As restrictions placed on OWL 2 Full in the use and position of certain OWL2 language features
b) As set of entailment rules to be applied to the RDF serialization of an OWL Ontology[6].

Below we present the syntax of entailment rules

A rule is generally of the form

$\mathbf{A_1 \wedge A_2 \wedge \ldots\ldots\ldots\ldots\ldots\ldots \wedge A_n \rightarrow A,}$

Where A_1……..A_n are expression of the form C(x) concept expressions or R(x,y) role names, x and y are the variables or individual names where y can also be a data type if allowed and the following conditions are satisfied

a) The pattern of variables in rule body forms a tree.
b) The first argument of A is the root of the just mentioned tree.

These rules are defined by W3C in the following Style

Table: 2

Name	*If*	*Then*
Eq-sym	*T(?x, owl:SameAs, ?y)*	*T(?y,owl : SameAs, ?x)*

Every rule has a name some if condition and a then part. It is also possible to have no if condition that means this rule will be executed at the program start. The rules mentioned above can be divided into six categories

a) Semantics of Equality
b) Semantics of axioms and properties
c) Semantics of classes
d) Semantics of class axiom
e) Semantics of Data types
f) Semantics of Schema Vocabulary

Figure 2 gives example of certain rules:

Man(x) ^ hasBrother(x,y) ^ hasChild(y,z) → Uncle(x)

Cake(x) → ∃ contains.Egg(x)

worksAt(x,y) ^ → University(y) Employeeof(x,y)

supervises(x,z) ^ PhdStudent(z) → Guideof(x,z)

These rules take a variety of forms-:

a) Triple pattern rules- The rule body and head are made up of atomic formulae representing triples in RDF graph.

b) ***Assertional rules-*** The rule body is empty, in which case these can be considered as being always applicable.

c) ***Consistency check-*** The head of these rules contain false only, in which case the input RDF graph should be considered inconsistent when the premises of the rule body hold.

d) ***List rules –*** These rules make shorthand notations for processing RDF collections.

e) ***Data type rules-*** These rules require special processing for data types, eg rule dt-eq that asserts lt1 owl:sameas lt2[7].

One advantage of encoding knowledge using OWL is that there are many tools available now a days that can reason ontologies in a simpler and easier manner. In OWL2 RL Profile the reasoning is done using a rule based reasoner. Many software packages are now a days available for creating ontologies and then reason upon them. i.e Protégé 3.5 and Protégé version 4.2 In Protégé 3.5 PROTÉGÉ OWL's SWRLTAB supports an OWL RL Reasoner. The SWRLTab provides control of the OWL 2 RL inference process by allowing the selective enabling and disabling of these rules. This control is provided at the Java API level via the SWRL Rule Engine API, the SQWRLQueryTab, the SWRLDroolsTab, and the SWRLJessTab. This interface provides a control tab to indicate if rule tables are active or inactive. The following is an example of this interface as displayed in the SWRLJessTab.[7].

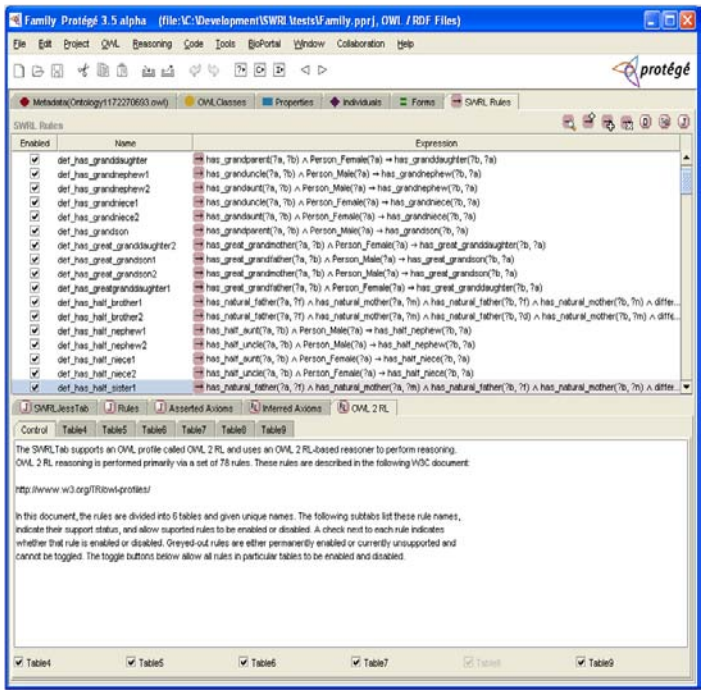


Figure 2 SWRL Jess Tab

In protégé 4.2 however the rules are visible under Window -> Views -> Ontology Views -> Rules and SWRL Tab is visible under Window -> Tabs -> SWRL.

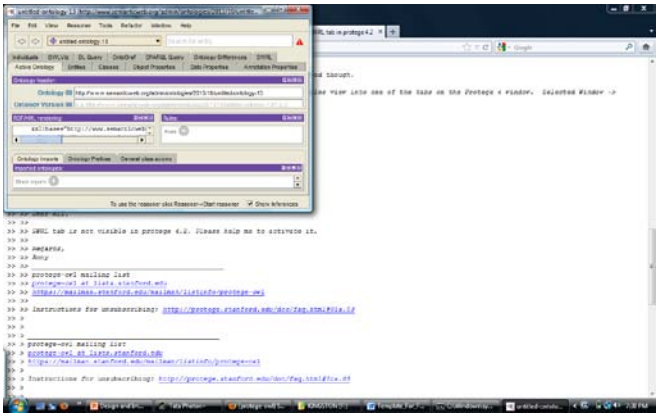


Figure 3 SWRL Tab in Protege 4.2

III. COMPARISON OF OWL2 RL PROFILE BASED REASONERS

This section of the study focuses the characteristics of OWL2 RL profile reasoners i.e Jena , OWLIM, Owl2 RL based inference engin integrated in Oracle 11g etc.

A. Jena2 Overview:

Jena2 offers a simple abstraction of RDF Graph and was first released in August 2003 which is the second generation toolkit In addition to providing an API for RDF, RDFS, OWL and SPARQL, it includes a rule-based inference engine; the inference engine can use both forward and backward chaining, and it supports the most common OWL constructs. Additionally it allows users to define their own custom rules, however it does not natively support any constructs introduced in OWL 2. Jena also supports incremental maintenance (when the forward-chaining RETE-based engine is used). This is used uniformly for graph implementations, including in-memory, database-backed, and inferred graphs. The main contribution of Jena is a rich API for manipulating RDF graphs. Around this, Jena provides various tools, e.g., an RDF/XML parser, a query language, I/O modules for N3, N-triple and RDF/XML output. Underneath the API the user can choose to store RDF graphs in memory or in databases. Jena provides additional functionality to support RDFS and OWL[8].

The two key architectural goals of Jena2 are:

a. Multiple, flexible presentations of RDF graphs to the application programmer. This allows easy access to, and manipulation of, data in graphs enabling the application programmer to navigate the triple structure.

b. A simple minimalist view of the RDF graph to the system programmer wishing to expose data as triples.

B. Storage Schema and Architecture of Jena2:

Jena2 uses a denormalized schema in which URIs and simple literal values are stored directly in the statement table and a separate table is only used to store literal values and

separate resource table is used to store long URIs only, the advantage of which is that many operations are possible without a Join operation.Jena2 persistent architecture is implemented using specialized Graph interface as shown in the diagram below

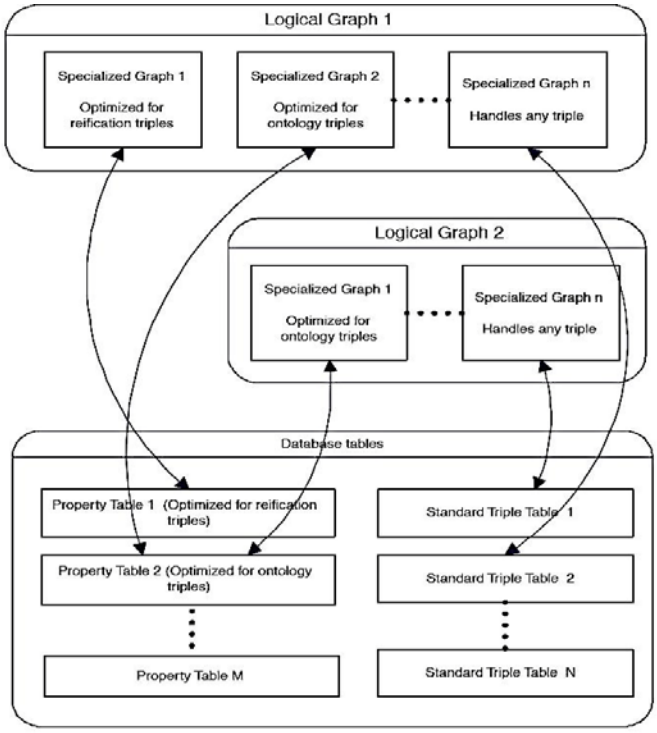


Figure 4 Jena2 Persistence Architecture an Overview

Jena2 also provides a general facility for clustering properties that are commonly accessed together. A Jena2 property table is a separate table that stores the subject value pairs related by a particular property[8]. The schema's for various tables are as

Statement Table

Subject	Predicate	Object
mylib.doc1	dc:title	Jena2
mylib.doc1	dc:description	101

Literals Table

Id	Value
101	A very long description that might be stored using blob.

Resource Table

Id	URI
201	hp: aResource with an extremely long URI

DC Properties Table

Subject	Title	Publisher	Description
Mylib.doc1	Jena2	-	101

C. Jena2 Persistence Architecture:

An overview of Jena2 architecture was presented in Figure1. In this section we describe an implementation of that architecture.

a. Specialized Graph Interface:

The Jena2 persistence layer presents a Graph interface to the higher levels of Jena, supporting the graph operations of add, delete and find.

b. Database Driver:

The database driver provides an abstract storage interface that insulates the specialized graphs from differences in how database engines support blobs,nulls,expressions,table and index creations.

c. Configuration and Meta Graphs:

Jena2 provide default graphs containing the default configuration parameters for all supported databases.Associated with each Jena2 persistent store is a meta-graph, a separate auciliary RDF graph containing metadata about each logical graph.

D. Inference Support :

In Jena2, inference engines are structured as Graph combinators called Reasoners. An instance of Reasoners combine one or more RDF Graphs and exposes the entailments from them as another RDF Graph.

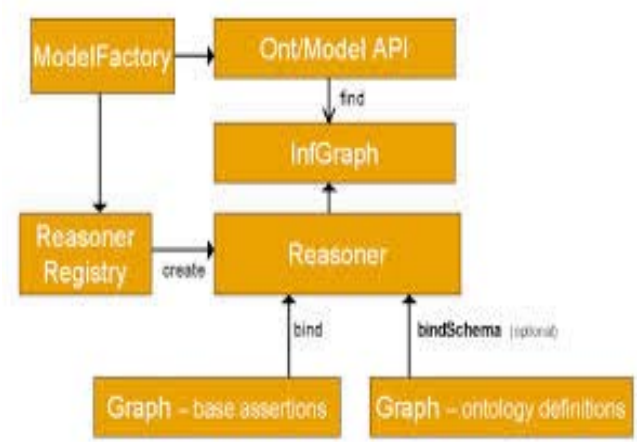


Figure 5 The inference API Layering

E. Built in Reasoners:

As a part of default distribution we include a selection of inference engines which includes

a) Transitive Reasoner- This reasoners provides the transitive closure of the rdfs:subPropertyOf relationships contained in the source graphs.
b) RDFS Reasoner- This provides an implementation of the RDFS closurerules.
c) Rubrik Reasoner- This reasoner supports rule based RDF inferences. Rule clauses are either extended triple or procedural callouts to primitives defined in Java. Both forward chaining and backward chaining rule engines are provided with some hybridization in that forward rules are able to create and install new backward rules.

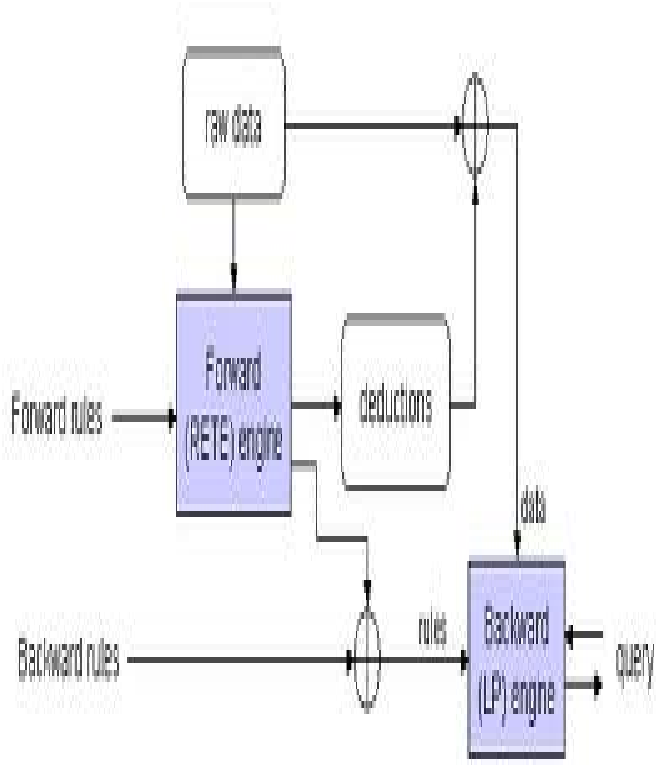


Figure 6 Forward and Backward engine : Rubrik Reasoner

To summarize we can conclude as

a) Supports a denormalized schema used for storing generic triple statements.
b) Property tables are used to store subject value pairs.
c) Provides an efficient implementation for reification

IV. OWLIM A SEMANTIC REPOSITORY

The OWLIM Repository is implemented in Java which consist of a native RDF Store, a reasoner and a query answering engine. It is packaged as SAIL (Storage and Inference Layer) and distributed as scalable, resilient platform[9].

The reasoner is based on R-entailment defined by ter Horst, where inference rules are applied directly to RDF triples. Each rule is made up of a number of premises and conclusions, each of which is an RDF triple pattern with variables allowed at any position. The rule language is based on R-entailment defined by ter Horst and the reasoner uses forward chaining to apply selected inference rules directly to RDF Statements. There are two editions of OWLIM: SwiftOWLIM and BigOWLIM, The SwiftOWLIM is an entirely in memory system and BigOWLIM uses a file based storage layer. Typically SwiftOWLIM can manage millions of explicit statements on desktop hardware whereas BigOWLIM can manage billion of statements and multiple concurrent user sessions[10,11].

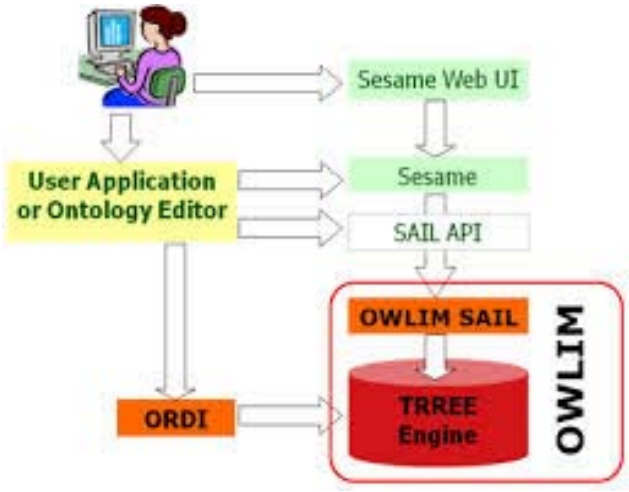


Figure 7 OWLIM Architecture (owlim.ontotext.com)

A. Reasoning Capabilities:

The inferencing strategy in OWLIM is materialization of all inferred statements at load time which are based on R-entailment like rules. Total materialization involves computing all the entailed statements at load time which somewhat increases the reasoning cost but can be balanced as query processing can proceed extremely fast. In all edition of OWLIM several standard rule set are included i.e 'empty' (no inference), OWL-Horst,RDFS and OWL2 RL. In addition to the standard semantics, user-defined rule set can also be used.

V. CONCLUSION

This is a review paper in which an attempt is made to highlight the three profiles of OWL moreover it also gives a brief introduction to OWL RL profile language and the working of few reasoners under this profile. The reasoners Jena2 and OWLIM were evaluated on the basis of their working and the Performance measurements of Jena2 indicated that the denormalized schema of Jena2 is twice as fast as that of Jena1,which can further be optimized when native SQL types will be used instead of string literals. On the other hand OWLIM is both sound and complete when working on OWL RL/RDF rules except for the missing support for data type reasoning.

VI. REFERENCES

[1] Jeff Heflin , Raphael Volz , Raphael Volz , Requirements for a *Web* Ontology Language, *www.w3.org/TR/2002/WD-webont-req-20020307/*

[2] Natalya F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

[3] Michael K. Smith, Chris Welty, Deborah L. McGuinness, OWL Web Ontology Language Guide, http://www.w3.org/TR/owl-guide/

[4] Frank W. Hartela, Sherri de Coronadoa,Robert Dionneb, Gilberto Fragosoa, Jennifer Golbeckc, Modeling a description logic vocabulary for cancer research, http://www.sciencedirect.com/science/ article/pii/ S1532046404000917

[5] Juan Sequeda, Introduction to OWL Profiles, http://semanticweb.com/introduction-to-owl-profiles_b35607

[6] Boris Motik Bernardo Cuenca Grau, OWL 2 Web Ontology Language Profiles http:// www.w3.org // TR /owl2-profiles/

[7] Jartin J.O'Connor, Amar Das,"A Pair of OWL2 RL Reasoners", Standford Centre of Biomedical Informatics Research Standford,CA 94305,U.S.A

[8] Kevin Wilkinson, Craig Sayers, Harumi Kuno, Dave Reynolds, Efficient RDF Storage and Retrieval in Jena2,http://www.hpl.hp.com/techreports/2003/HPL- 2003-266.pdf

[9] Barry Bishop, Spas Bojanov,"Implementing OWL 2 RL and OWL 2 QL rule-sets for OWLIM". Proc. of 8th International", Workshop on OWL: Experiences and Directions (OWLED2011), San Francisco, USA, June 5-6, 2011, CEUR-WS.org, ISSN 1613-0073.

[10] [10] OWLIM Primer *nlpainter.googlecode.com /svn.../owlim_docs/OWLIM_primer_v4.3.pdf*

[11] [11] Atanas Kiryakov, Barry Bishop, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev,Ruslan Velkov Ontotext AD, 135 Tsarigradsko Chaussee, "The Features of BigOWLIM that Enabled the BBC's World Cup Website" Proceedings of VLDB 2010, International Conference on Very Large Databases Singapore.

.