**PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA**
**MINISTERY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH**
**UNIVERSITY OFMOHAMED BOUDIAF - M'SILA**

**FACULTY: Mathematics and Computer Science**
**DEPARTMENT: Computer Science**
**N°:……**

**DOMAIN: Mathematics and Computer Science**
**Branch: Computer Science**
**option: SIGL**

**Dissertation submitted in partial fulfillment of the requirements for The degree of MASTER**

**By**: TAKIA HADJIRA

## SUBJECT

# Ontology Based Relational Database Access

**Publicly defended before the jury composed of:**

| | | |
|---|---|---|
| ..... | University of M'sila | Chair |
| Mr. MEFETAH LAKHAL | University of M'sila | Supervisor |
| ..... | University of M'sila | Examiner |

**Academic Year: 2017/2018**

# Acknowledgements

This work could not have reached fruition without the unflagging assistance and participation of so many people whom I would never thank enough for the huge contribution that made this work what it is now.

First and foremost, heartfelt gratitude and praises go to the Almighty Allah who guided me through and through.

I would like to thank profoundly Ms M.Lakhal for his scientific guidance and corrections, suggestions and advice, pertinent criticism and pragmatism, and particularly for his hard work and patience. I am very grateful to him, for without his help an important part of this work would have been missed.

My sincere appreciation needs to be addressed to the honorable board of examiners, whose insightful remarks during the viva will certainly enrich this work.

# Plan

# Figures Table:



# Table list

# GENERAL INTRODUCTION

In this thesis we will teak a new paradigm called Ontology-based data access (OBDA) which facilitates access to relational data, realized by linking data sources to ontology by means of declarative mappings. In the first chapter we talk about semantic web which is actually an extension of the current one in that it represents information more meaningfully for humans and computers alike. It enables the description of contents and services in machine-readable form, and enables annotating, discovering, publishing, advertising and composing services to be automated. It was developed based on Ontology, which is considered as the backbone of the Semantic Web. In other words, the current Web is transformed from being machine-readable to machine-understandable. In fact, Ontology is a key technique with which to annotate semantics and provide a common, comprehensible foundation for resources on the Semantic Web. Moreover, Ontology can provide a common vocabulary, a grammar for publishing data, and can supply a semantic description of data which can be used to preserve the Ontologies and keep them ready for inference. This chapter provides Architecture of semantic web and Components of Ontology,The Web Ontology Language Owl and defines the importance of ontologies . In the second chapter we talk about description logic , a family of logic-based knowledge representation languages that can be used to represent the terminological knowledge of an application domain in a structured way.

It first it gives a short introduction of the ideas underlying Description Logics. Then it introduces syntax and semantics, covering the basic constructors can be used to build knowledge bases and I talked about a Big Family of description Logics. Finally, we touched DLLite which is forms the basis of OWL 2 QL that is latter is to be the language of choice for applications that use very large amounts of data and where query answering is the most important reasoning task.

In the last chapter we will talk about OBDA which as we previously knew it that is In recent years, ontology-based data access (OBDA) has emerged as a promising and challenging application of ontologies. The idea is to enrich instance data with a ‘semantic layer’ in the form of an ontology, used as an interface for querying and to derive additional answers. A central research problem in this area is to design query answering engines that can deal with sufficiently expressive ontology languages yet scale to large data sets. The most

popular ontology languages that have been considered for OBDA include the three profiles OWL2 RL, OWL2 QL, and OWL2 EL, as well as various description logics and data-log variants related to these profiles where OBDA system rewrites ontology and query to new data-log query for answering this new query with it we talk about rewriting query and algorithm the rewriting and the finally I talked about answering query .

# Chapter 1:

# The Semantic Web

# 1. Introduction

The Semantic Web is an intelligent incarnation and advancement in World Wide Web to collect, manipulate and annotate the information by providing categorization[1]on a web-page and reprocesses it so that other machines including computers can understand the information. Semantic Web was part of Berners-Lee's vision for the World Wide Web from the beginning (Berners-Lee ., 2001, Berners-Lee, 2003).

where he sees it as being an extension of the current World-Wide Web that will bring a common structure to the content of Web pages, thereby providing such content with meaning which will allow external software agents to carry out sophisticated tasks on behalf of the reader or user and, as such, promote a greater degree of cooperation between humans and computers. In so doing, a new age of computing will be ushered in where machines are better able to process and "understand" the data.

This vision of a Semantic Web can therefore be viewed from three different perspectives:

- ✓ a type of universal library which can readily be accessed and used by humans in their day-to-day information acquisition;
- ✓ the backbone for software or computational agents to use autonomously in order to perform particular activities on behalf of their human counterparts;
- ✓ a method for federating particular knowledge bases and databases to perform anticipated tasks for humans and their agents (Marshall and Shipman ). [2]

By using the descriptive technologies Resource Description Framework (RDF) and Web Ontology Language(OWL), and the data-centric, customizable Extensible Mark-up Language (XML).These technologies are combined in order to provide descriptions that supplement or replace the content of Web documents. Thus, content may manifest as descriptive data stored in Web-accessible databases, or as mark-up within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout/rendering cues stored separately). The machine-readable descriptions enable content managers to add meaning to the content, i.e. to describe the structure of the knowledge we have about that content. In this way, a machine can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more

meaningful results and facilitating automated information gathering and research by computers.[3]

## 2. Architecture of semantic web:

Figure 1 illustrates the architecture of the Semantic Web:

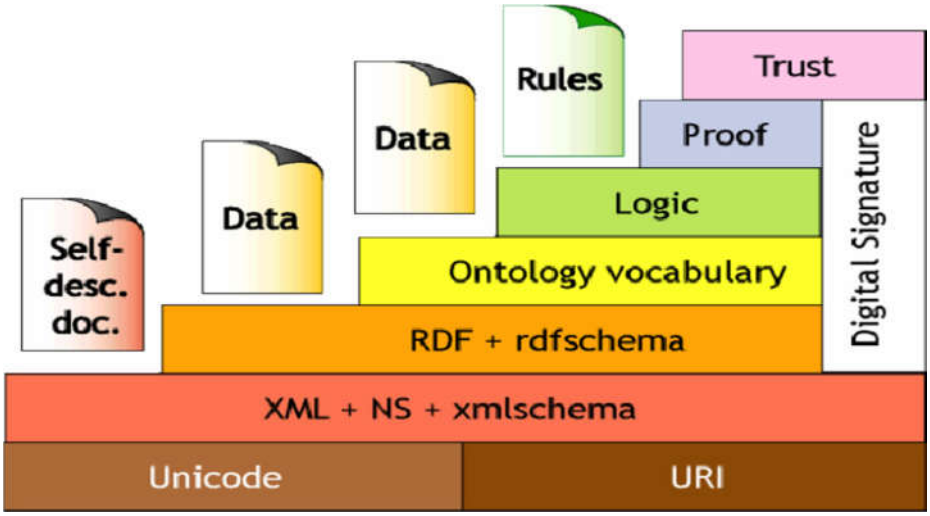


**Figure 1:**Semantic Web architecture.

### 2.1. Unicode:

Unicode provides a unique number for every character, independently of the underlying platform, program, or language. Before the creation of Unicode, there were various different encoding systems. The diverse encoding made the manipulation of data complex. Any given computer needed to support many different encodings. There was always the risk of encoding conflict, since two encodings could use the same number for two different characters, or use different numbers for the same character. Examples of older and well known encoding systems include ASCII and EBCDIC.[4]

• ASCII – 7 bit, 128 characters (a-z, A-Z, 0-9, punctuation)

• Extension code pages – 128 chars (ß, Ä, ñ, ø, Š, etc.)[4]

### 2.2. URI:

A universal resource identifier (URI) is a formatted string that serves as a means of identifying abstract or physical resource. A URI can be further classified as a locator, a name,

or both. Uniform resource locator (URL) refers to the subset of URI that identifies resources via a representation of their primary access mechanism. An uniform resource name (URN) refers to the subset of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable. For example:[4]

–http://www.ietf.org/rfc/rfc3986.txt

–mailto:John.Doe@example.com

–news:comp.infosystems.www.servers.unix

– telnet://melvyl.ucop.edu/

The URL http://dme.uma.pt/jcardoso/index.htm identifies the location from where a Web page can be retrieved.

• The generic form of any URI is scheme: [//authority] [/path] [?query] [#fragid]

– The scheme distinguishes different kinds of URIs, The scheme lays out the concrete syntax and any associated protocols for the URI. Schemes are case-insensitive and are followed by a colon. Ideally, URI schemes should be registered with the Internet Assigned Numbers Authority (IANA)Examples of popular schemes include http, https, ftp ,mailto, file ,data and irc, although nonregistered schemes can also be used.

– Authority normally identifies a server , An authority component is made up of multiple parts: an optional authentication section, a host -- consisting of either a registered name or an IP address -- and an optional port number. The authentication section contains the username and password, which are separated by a colon and followed by the symbol for at (@). After the @ comes the hostname, which is in turn followed by a colon and then a port number.

– Path normally identifies a directory and a file, The path contains data, is notated by a sequence of segments separated by slashes. The path must begin with a single slash if an authority part was present. It may also begin with a single slash even if there is no authority part, but it cannot begin with a double slash. Keep in mind that while this part of the syntax may closely resemble a particular file path, it does not always imply a relation to that file system path.

– Query adds extra parameters ,it contains a string of nonhierarchical data. Although the syntax is not well-defined, it is most often a sequence of attribute value pairs separated by a

delimiter, such as an ampersand or a semicolon. The query is separated from the preceding part by a question mark.

– Fragment ID identifies a secondary resource, it contains a fragment identifier that provides direction to a secondary resource. For example, if the primary resource is an HTML document, the fragment is often an ID attribute of a specific element of that document. If the fragment identifies a certain section of an article identified by the rest of the URI, a Web browser will scroll this particular element into view. The fragment is separated from the preceding part by a hash (#).[51]

### 2.3. Extensible Markup Language (XML):

Is a markup language, which means that it is machine-readable and has its own format. It is widely known in the WWW community because it has a flexible text format and was designed to describe data and to meet the challenges of large-scale e-business and electronic publishing; it plays an important role in exchanging different types of data on the Web. In fact, it is the basis of a rapidly growing number of software development activities. Each document starts with a namespace declaration using XML Namespace.[4]

**Syntax:**

<name>contents</name>

–<name> : is called the opening tag

–</name> : is called the closing tag

• **Examples for xml:**

<gender>Female</gender>

<story>Once upon a time there was…. </story>

• Element names case-sensitive

Name/value pairs, part of element contents

• Syntax <name attribute_name="attribute_value">contents</name>

• Values surrounded by single or double quotes

• **Example**

<temperature unit="F">64</temperature>

<swearword language='fr'>con</swearword>

Empty element: <name ></name>

• This can be shortened:<name/>

• Empty elements may have attributes

• Example<grad value='A'/>

**XML namespace and XML schema** definitions makes sure that there is a common syntax used in the semantic Web. XML namespaces allow specifying different markup vocabularies in one XML document. XML schema serves for expressing schema definition of a particular XML document.[6]

Example for xmlns:

Use "xmlns" attribute

**Named prefix :**

<a:foo xmlns:a='http://example.com/NS'/>

**Default prefix :**

<foo xmlns='http://example.com/NS'/>

### 2.4.The Resource Description Framework (RDF) and RDF Schema:

On top of XML is RDF that is a framework for using and representing metadata and describing the semantics of information about resources on the Web in a machine-accessible way. It uses URIs to identify Web resources and to describe the relations between these resources, using a graph model. While describing classes of resources and the properties between them, using RDF Schema (which is a simple modeling language), it also provides a simple reasoning framework for inferring types of resources.

RDF Schema (RDFS) defines the vocabulary of RDF model. It provides a mechanism to describe domain-specific properties and classes of resources to which those properties can be

applied, using a set of basic modeling primitives (class, subclass-of, property, subproperty-of, domain, range, type). However, RDFS is rather simple and it still does not provide exact semantics of a domain.[8]

Information is represented by triples subject-predicate-object in RDF. An example of a triple is shown in the figure below. It says that "Joe Smith has homepage http://www.example.org/~joe". All elements of this triple are resources defined by URI. The first resource http://www.example.org/~joe/contact.rdf#joesmith(subject) is intended to identify Joe Smith. Note that it precisely defines how to get to a RDF document as well as how to get the joesmith RDF node in it. The second resource http://xmlns.com/foaf/0.1/homepage (predicate) is the predicate homepage from a FOAF (Friend-of-a-friend) vocabulary. The last resource (object) is Joe's homepage http://www.example.org/~joe/.

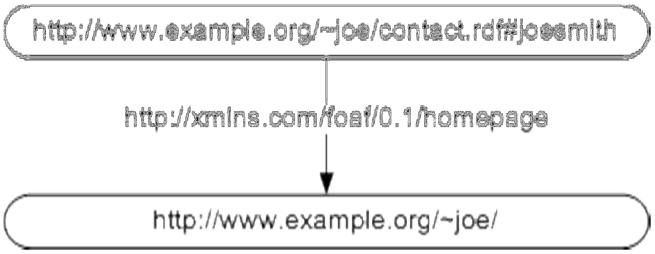


**Figure 2**:RDF triple (in graph representation) describing Joe Smith - "Joe has homepage identified by URI http://www.example.org/~joe/"

All of the elements of the triple are resources with the exception of the last element, object, that can be also a literal. Literal in the RDF sense is a constant string value such as string or number. Literals can be either plain literals (without type) or typed literals typed using XML Datatypes. An example of literal usage is illustrated in the triple shown in the figure below.

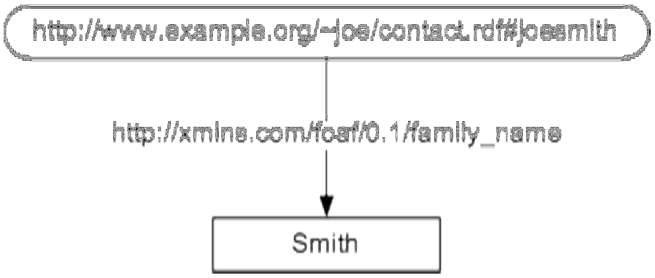


**Figure 3**:RDF triple (in graph representation) describing Joe Smith - "Joe has family name Smith"[9].

## 2.5.Ontology Vocabulary:

Ontology comprises a set of knowledge terms, including the vocabulary, the semantic interconnections, simple rules of inference and logic for some particular topic. Ontologies applied to the Web are creating the semantic Web. Ontologies facilitate knowledge sharing and provide reusable Web contents, Web services, and applications. Few of the ontology languages are DAML (DARPA Agent Markup Language), OIL (Ontology Interference Layer) and OWL (Web Ontology Language). OWL is developed starting from description logic and DAML+OIL. OWL is a set of XML elements and attributes, with well-defined meaning, that are used to define terms and their relationships (e.g. Class, equivalentProperty, intersectionOF, unionOF, etc.). OWL elements extend the set of RDF and RDFS elements, and the OWL namespace is used to denote OWL encoding. OWL comes in three species – OWL Lite for taxonomies and simple constraints, OWL DL for full description logic support and OWL Full for maximum expressiveness and syntactic freedom of RDF. OWL DL is widely used for ontology representation. In practice, ontologies are often developed using integrated, graphical, ontology authoring tools, such as Protégé, OILed and OntoEdit. Protégé facilitates extensible infrastructure and allows an easy construction of knowledge rich domain ontologies[4].

Ex :

Ontology: <http://example.org/tea.owl>

Class: Tea

## 2.6. Logic and Proof:

The logic layer consists of logic-based rules that can serve as extensions of, or alternatives to, description logic based ontology languages; and can be used to develop declarative systems on top of (using) ontologies. The logic layer is accompanied by the proof layer which contains all the necessary inference mechanisms that manipulate the rules of the logic layer and, furthermore, explicitly expose the proofs of inferences in order to generate explanations between proof systems, agents and humans. [52]

## 2.7. Trust:

Is the final layer of the Semantic Web. This component concerns the trustworthiness of the information on the Web in order to provide an assurance of its quality. [5]

## 3.The goal of semantic web:

The goal of semantic web research is to allow the vast range of web-accessible information and services to be more effectively exploited by both humans and automated tools. To facilitate this process, RDF and OWL have been developed as standard formats for the sharing and integration of data and knowledge the latter in the form of rich conceptual schemas called ontologies. These languages, and the tools developed to support them, have rapidly become standards for ontology development and deployment; they are increasingly used, not only in research labs, but in large scale IT(Information Technologies) projects. Although many research and development challenges still remain, these "semantic web technologies" are already starting to exert a major influence on the development of information technology.[6]

## 4.Ontologies:

Ontology : all the objects recognized as existing in the domain. To build an ontology, it is also to decide on the way of being and to exist objects. To continue towards a definition of ontology, it seems to us essential to remind that the works on the ontologies are developed in an IT context that is the case, for instance, for Engineering of knowledge, Artificial intelligence or, more specifically here, the context of Semantics Web where the final goal is to specify an IT artifact. In this context, the ontology becomes then a model of the existing objects which makes a reference to it through concepts of the domain.

The ontology is a theory on the representation of the knowledge. As indicated in 2000, the ontology "defines the kinds of things that exist in the application domain". It is by this theory that it unifies in the domain of the computing. The ontology "is a formal, explicit specification of a shared conceptualization" (Gruber, 1993). For the IT specialist of semantic web, the ontology is a consensual model, because the conceptualization is shared and brings then to build a linguistic specification with the vocabulary RDF/RDFS and the language OWL. In the semiotic perspective, conceptualization according to Gruber relates to the domain of the speech because it is the abstraction. The domain of the speech takes place of the referent. In semiotics, we shall say that the ontology symbolizes the conceptualization, the terms, the notions and the relations which are conceptualized. [7]

Ontologies : play an important role in achieving interoperability across organizations and on the Semantic Web , because they aim to capture domain knowledge and their rule is to create semantics explicitly in a generic way, providing the basis for agreement within a domain. Ontology is used to enable interoperation between Web applications from different areas or from different views on one area. For that reason, it is necessary to establish mappings among concepts of different ontologies to capture the semantic correspondence between them.

An ontology is a specification of a conceptualization .The ontology defines a set of representational are typically classes (concepts), attributes (or properties), and relationships (or relations among class members). [7]

A concept can be an object of any sort: person, car, can describe an activity or state: swimming, Work ,Exam. an represent abstract concepts like time or value. There is no strict restriction what can express as a concept in our ontology.

A relationship in an ontology represents a way in which two concepts, two things, can be connected to each other :Train needs Rails .characteristics of objects: Fatma is Woman.

## 5.Components of Ontology:

Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. Most ontology describes individuals (instances), classes (concepts), attributes, and relations. Common components of ontologies include:

**Individuals:** Elements represented by individuals or instances are terms not included in the class. 'Person' is a class, but 'John Smith' is an instance.

**Classes**: Sets, collections, concepts, classes in programming, types of objects, or kinds of things.

**Relations**: Elements represented by relationships or attributes are used to describe the relationship between two terms.

**Function terms:** A function is a special type of relationship, meaning that a term has a restriction on its number in relation to other terms. For example, 'mother' has a relationship with the concept of 'woman', one of the concepts of 'people'. A function can be referenced to a slot if it is applicable to only two terms.

**Restrictions**: Formally stated descriptions of what must be true in order for some assertion to be accepted as input.

**Axioms:** The axiom is a sentence composed of First Order Logic, which is found to be "true" and is not proven. Structurally, you can refer to a statement that can not be expressed as a slot or value. Axiom must use prefix notation. '=>' Is a logical implementation, '<=>' means logically equivalent, and there are notations that can express connection, separation, negation, Variables must begin with '?' And free variables are assumed to be quantified. [1]

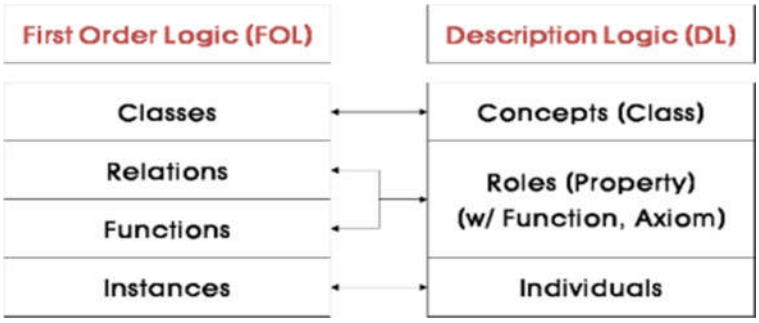


**Figure 4**: Components of Ontology.

## 6. Use of ontologies:

In A knowledge in computer systems is thought of as something that is explicitly represented and operated on by inference processes .Any software that does anything useful cannot be written without a commitment to a model of the relevant world—to entities, properties, and relations in that world. Data structures and procedures implicitly or explicitly make commitments to a domain ontology.[8]

Information-retrieval systems, digital libraries, integration of heterogeneous information sources, and Internet search engines need domain ontologies to organize information and direct the search processes.[8]

For example, a search engine has categories and subcategories that help organize the search. The search-engine community commonly refers to these categories and subcategories as ontologies. Object-oriented design of software systems similarly depends on an appropriate domain ontology. Object systems representing a useful analysis of a domain can often be reused for a different application program. Object systems and ontologies emphasize different aspects.[8]

As information systems model large knowledge domains, domain ontologies will become as important in general software systems as in many areas of AI. In AI, while knowledge representation pervades the entire field. [8]

## 7. The Web Ontology Language Owl:

OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. Although already recognizable as an ontology language, the capabilities of RDF and RDF Schema (RDF-S) are rather limited: they do not, for example, include the ability to describe cardinality constraints, a feature found in most conceptual modeling languages, or to describe even a simple conjunction of classes. OWL facilitates greater machine interpretability of Web content by providing additional vocabulary along with a formal semantics OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full as shown below(figure 5).This layering is motivated by different requirements that different users and developers have for a Web ontology language:


**Figure5**:Classification according to expression area of OWL. [50]

### 7.1.RDF/XML:

Is the most common and recommended syntax for OWL 2 documents, which are intended for those users primarily needing a classification hierarchy with typing of properties and meta-modeling facilities; The most elementary building block of RDF(S) is a class, which defines a group of individuals that belong together because they share some properties. The following states that an instance e belongs to a class c:

$$\boldsymbol{Individual(etype(c))}\ (\boldsymbol{“eisoftypec”})$$

The second elementary statement of RDF(S) is the subsumption relation between classes subClassOf:

$$\boldsymbol{subClassOf}(\boldsymbol{cicj})$$

In RDF, instances are related to other instances through properties:

$$\mathbf{In}\boldsymbol{dividual}(\boldsymbol{eivalue}(\boldsymbol{pej}))$$

Properties are characterized by their domain and range:

$$\boldsymbol{ObjectProperty}(\boldsymbol{pdomain}(\boldsymbol{ci})\boldsymbol{range}(\boldsymbol{cj}))$$

just as with classes, properties are organised in a subsumption hierarchy:

$$\boldsymbol{SubPropertyOf}(\boldsymbol{o1}:\boldsymbol{pio2}:\boldsymbol{pj})$$

RDF and RDFS allow the representation of some ontological knowledge. The main modeling primitives of RDF/RDFS concern the organization of vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classes.

### 7.2.OWL Full:

The entire language is called OWL Full, and uses all the OWL languages primitives . It also allows to combine these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) to change the meaning of the pre-defined (RDF or OWL) primitives, by applying the language primitives to each other. For example, in OWL Full we could impose a cardinality constraint on the class of all classes, essentially limiting the number of classes that can be described in any ontology. The advantage of OWL Full is that it is fully upward compatible with RDF, both syntactically and semantically: any legal RDF document is also a legal OWL Full document, and any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion.

The disadvantage of OWL Full is the language has become so powerful as to be undecided, dashing any hope of complete (let alone efficient) reasoning support.

### 7.3.OWL DL:

In order to regain computational efficiency, OWL DL (short for: Description Logic) is a sublanguage of OWL Full which restricts the way in which the constructors from OWL and

RDF can be used. but roughly this amounts to disallowing application of OWL's constructor's to each other, and thus ensuring that the language corresponds to a well studied description logic. The advantage of this is that it permits efficient reasoning support. The disadvantage is that we lose full compatibility with RDF: an RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Conversely, every legal OWL DL document is still a legal RDF document.

It is often useful to say that two classes are disjoint:

$$\boldsymbol{DisjointClasses}(\boldsymbol{cicj})$$

OWL DL allows arbitrary Boolean algebraic expressions on either side of an equality of subsumption relation:

$$\boldsymbol{SubClassOf}(\boldsymbol{ciunionOf}(\boldsymbol{cjck}))$$

ci is not subsumed by either cj or ck, but is subsumed by their union. Similarly

$$\boldsymbol{EquivalentClasses}(\boldsymbol{ciintersectionOf}(\boldsymbol{cjck}))$$

OWL DL completes the Boolean algebra by providing a construct for negation:

$$\boldsymbol{complementOf}(\boldsymbol{cicj})$$

**7.4.OWL Lite:**

An ever further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointed statements and arbitrary cardinality (among others). The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is of course a restricted expressivity.

equality are possible in OWL Lite:

$$\boldsymbol{SameIndividual}(\boldsymbol{eiej})$$

through a pair of mutual Subclassof or SubPropertyOf statements), this can be done directly in OWL Lite:

$$\boldsymbol{EquivalentClasses}(\boldsymbol{c1cj})\ \boldsymbol{EquivalentProperties}(\boldsymbol{p1pj})$$

inequality must be explicitly stated, as:

$$\boldsymbol{DifferentIndividuals(eiej)}$$

OWL Lite provides an abbreviated form:

$$\boldsymbol{DifferentIndividuals(e1 \ldots e4)}$$

The relationship between such pairs of properties is established by stating:

$$\boldsymbol{ObjectProperty(piinverseOf(pj))}$$

Other vocabulary in OWL Lite (TransitiveProperty and SymmetricProperty are modifying a single property, rather then establishing a relation between two properties: [9]

$$\boldsymbol{ObjectProperty(o1 : piTransitive)}$$

$$\boldsymbol{ObjectProperty(o1 : piSymmetric)}$$

**7.4.1.OWL 2 EL:**

Is based on the description logic EL. It enables polynomial time algorithms for all the standard reasoning tasks; it is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees.[10]

**7.4.2. OWL 2 QL:**

Is based on description logics similar to DL-Lite . It is one of three profiles of the Web Ontology Language OWL 2, was designed with the aim of supporting ontology-based data access (OBDA). The key idea is that data, 'stored in a standard relational database management system (RDBMS), can be queried through an OWL 2 QL ontology via a simple rewriting mechanism, i.e., by rewriting the query into an SQL query that is then answered by the RDBMS, without any changes to the data' (www.w3. org/TR/owl2-profiles). The rewritability property ensures, in particular, that the data complexity of answering queries over OWL 2 QL ontologies matches the complexity of database query answering, which is in AC0.[10]

**7.4.3.OWL 2 RL:**

Enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples. [10]

## 8.Why are ontologies important? :

Without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge. Thus, the first step in devising an effective knowledge representation system, and vocabulary, is to perform an effective ontological analysis of the field, or domain. Weak analyses lead to incoherent knowledge bases.

Clarifying the terminology enables the ontology to work for coherent and cohesive reasoning purposes

Ontologies enable knowledge sharing. Suppose we perform an analysis and arrive at a satisfactory set of conceptualizations, and their representative terms, for some area of knowledge for example, the electronic-devices domain. The resulting ontology would likely include domain-specific terms such as transistors and diodes; general terms such as functions, causal processes, and modes; and terms that describe behavior such as voltage. The ontology captures the intrinsic conceptual structure of the domain.

In order to build a knowledge representation language based on the analysis, we need to associate terms with the concepts and relations in the ontology and devise a syntax for encoding knowledge in terms of the concepts and relations.[11]

We can share this knowledge representation language with others who have similar needs for knowledge representation in that domain, thereby eliminating the need for replicating the knowledge-analysis process. Shared ontologies can thus form the basis for domain-specific knowledge-representation languages. Shared ontologies let us build specific knowledge bases that describe specific situations. For example, different electronic devices manufacturers can use a common vocabulary and syntax to build catalogs that describe their products. Then the manufacturers could share the catalogs and use them in automated design systems. This kind of sharing vastly increases the potential for knowledge reuse.[11]

## 9. CONCLUSION:

The goal of semantic web research is to enable the vast range of web-accessible information and services are given a well defined meaning. The semantic Web is a Web for machines, but the process of creating and maintaining it is a social one. To make possible the creation of the semantic Web the W3C has been actively working on the definition of open standards, such as the RDF and OWL. Although machines are helpful in manipulating symbols according to pre-defined rules, only the users of the semantic Web have the necessary interpretative and

associative capability for creating and maintaining ontologies. The principal benefit of semantics is that it provides a formal foundation for reasoning about the properties of systems that do automated knowledge translation based on sharing of ontology. Developers are vigorously building semantic Web services.

The goal of an ontology is to achieve a common and shared knowledge that can be transmitted between people and between application systems. Thus, ontologies play an important role in achieving interoperability across organizations and on the Semantic Web, because they aim to capture domain knowledge and their role is to create semantics explicitly in a generic way, providing the basis for agreement within a domain. Thus, ontologies have become a popular research topic in many communities. In fact, ontology is a main component of this research; therefore, the definition, structure and the main operations and applications of ontology are provided.

# Chapter 2:

# DESCRIPTION LOGICS

# 1. INTRODUCTION:

**Description logics (DLs):** are a family of formal knowledge representation (KR) languages[20],which is among the most important formalisms for ontological modeling today, which is also due to their central role for the semantics of the Web Ontology Language OWL. that can be used to represent the terminological knowledge of an application domain in a structured way. [12]

**DLs** are built from atomic concepts and atomic roles . Such description can be use in axioms and assertions of DL Knowledge bases and can be reasoned about write DL knowledge based by DL Systems.

Description Logics support inference patterns that occur in many applications of intelligent information processing systems, and which are also used by humans to structure and understand the world: classification of concepts and individuals.

- ✓ Classification of concepts determines subconcept/superconcept relationships (called subsumption relationships in DL) between the concepts of a given terminology, and thus allows one to structure the terminology in the form of a subsumption hierarchy.
- ✓ Classification of individuals (or objects) determines whether a given individual is always an instance of a certain concept. [13]

# 2. Definition of the basic formalism:

A knowledge base (KB) comprises two components, the TBox and the ABox:

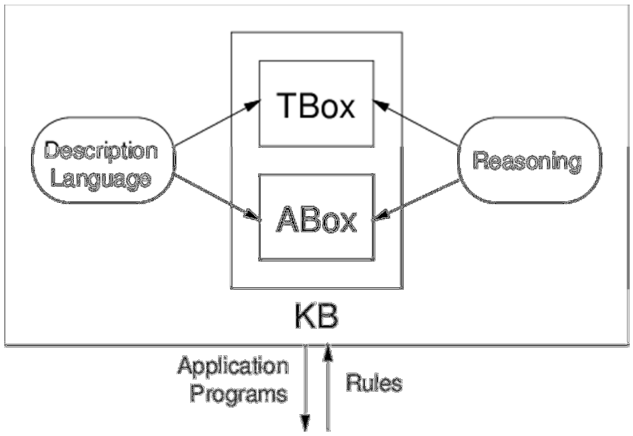


**Figure 6:**Architecture of a knowledge representation system based on Description Logics. [23]

- The TBox introduces t²he terminology, i.e., the vocabulary of an application domain.
- The ABox contains assertions about name individuals in terms of this vocabulary.

**2.1.The TBox :**

One key element of a DL knowledge base is given by the operations used to build the terminology. Such operations are directly related to the forms and the meaning of the declarations allowed in the TBox.

The TBox can be used to assign names to complex descriptions. The language for building descriptions is a characteristic of each DL system, and different systems are distinguished by their description languages. [14]

The basic form of declaration in a TBox is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts. For example, a woman can be defined as a female person by writing this declaration:

$$\mathbf{Woman \equiv Person \cap Female}$$

That declaration is usually interpreted as a logical equivalence which amounts to providing both sufficient and necessary conditions for classifying an individual as a Woman. This form of definition is strength of this kind of declaration is usually considered a characteristic feature of DL knowledge bases.[14]

the below figure (figure 7 ) illustrates how convert natural language into TBox inference

| | |
|---|---|
| All Women are Persons | $Woman \subseteq Person$ |
| Any Married Person has some spouse | $MarriedPerson \subseteq \exists spouse.MarriedPerson$ |
| Men are not Women | $Man \subseteq \neg Woman$ |
| Men and Women are disjoint | $Man \cap Woman \subseteq \bot$ |
| People who have grandchildren are grandparents | $Person \cap \exists hasChild.(\exists hasChild.Person) \subseteq Grandparent$ |

**Figure 7**: DL-based TBOX inference.

**DL terminologies:**

_ Only one definition for a concept name is allowed.

_ Definitions are acyclic in the sense that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them.

**TBox Axiomes:**

A name can be assigned to a concept description by a concept definition. For instance, we can write Tutorial ≡ $Event \sqcap \exists\, givenby.Lecturer \sqcap \exists\, hastopic.\top$ to supply a concept definition for the concept Tutorial. Let A be a concept name and C, D be (possibly) complex concept description:

• A concept definition is a statement of the form equivalent A ≡ C.

• A general concept inclusion (GCI for short) is a statement of the form C ⊑ D.

It is easy to see that every concept definition A ≡ C can be expressed by two GCIs: A ⊑ C and C ⊑ A. The terminological information expressed by GCIs is collected in the so-called TBox, which is simply a finite set of GCIs. [15]

In the figure 8 and figure 9 DL Syntax and Example for illustrates the Tbox axioms.

| Axiom | DL Syntax | Example |
|---|---|---|
| subClassOf | $C_1 \sqsubseteq C_2$ | Human ⊑ Animal ⊓ Biped |
| equivalentClass | $C_1 \equiv C_2$ | Man ≡ Human ⊓ Male |
| disjointWith | $C_1 \sqsubseteq \neg C_2$ | Male ⊑ ¬Female |
| sameIndividualAs | $\{x_1\} \equiv \{x_2\}$ | {President_Bush} ≡ {G_W_Bush} |
| differentFrom | $\{x_1\} \sqsubseteq \neg\{x_2\}$ | {john} ⊑ ¬{peter} |
| subPropertyOf | $P_1 \sqsubseteq P_2$ | hasDaughter ⊑ hasChild |
| equivalentProperty | $P_1 \equiv P_2$ | cost ≡ price |
| inverseOf | $P_1 \equiv P_2^-$ | hasChild ≡ hasParent$^-$ |
| transitiveProperty | $P^+ \sqsubseteq P$ | ancestor$^+$ ⊑ ancestor |
| functionalProperty | $\top \sqsubseteq \leqslant 1P$ | ⊤ ⊑ ⩽1hasMother |
| inverseFunctionalProperty | $\top \sqsubseteq \leqslant 1P^-$ | ⊤ ⊑ ⩽1hasSSN$^-$ |

**Figure8**: Axiom of OWL

| Constructor | DL Syntax | Example | FOL Syntax |
|---|---|---|---|
| intersectionOf | $C_1 \sqcap \ldots \sqcap C_n$ | Human ⊓ Male | $C_1(x) \wedge \ldots \wedge C_n(x)$ |
| unionOf | $C_1 \sqcup \ldots \sqcup C_n$ | Doctor ⊔ Lawyer | $C_1(x) \vee \ldots \vee C_n(x)$ |
| complementOf | $\neg C$ | ¬Male | $\neg C(x)$ |
| oneOf | $\{x_1\} \sqcup \ldots \sqcup \{x_n\}$ | {john} ⊔ {mary} | $x = x_1 \vee \ldots \vee x = x_n$ |
| allValuesFrom | $\forall P.C$ | ∀hasChild.Doctor | $\forall y.P(x,y) \rightarrow C(y)$ |
| someValuesFrom | $\exists P.C$ | ∃hasChild.Lawyer | $\exists y.P(x,y) \wedge C(y)$ |
| maxCardinality | $\leqslant nP$ | ⩽1hasChild | $\exists^{\leq n} y.P(x,y)$ |
| minCardinality | $\geqslant nP$ | ⩾2hasChild | $\exists^{\geq n} y.P(x,y)$ |

**Figure 9:**OWL CLASS CONSTRUCTOR.

**2.2.The ABox:**

The ABox contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called membership assertions. For example:

**Female ∩ Person(fatma)**

states that the individual **fatma** is a female person. Given the above definition of woman, one can derive from this assertion that **fatma** is an instance of the concept Woman.

$$\boldsymbol{hasChild(fatma, mohamed)}$$

specifies that fatma has mohamed as a child. Assertions of the first kind are also called concept assertions, while assertions of the second kind are also called role assertions.

In the ABox, one describes a specific state of affairs of an application domain in terms of concepts and roles.

the figure 10 illustrates how the convert the natural language into ABox inference :

| | |
|---|---|
| Mary is a Woman | Woman(Mary) |
| Mary has Bill for spouse | spouse(Mary,Bill) |
| Mary has at least two male sons | ≥ 2 hasChild.Man(Mary) |
| Mary and Jane are different people | Mary ≠ Jane |

**Figure 10**: DL-based ABOX inference.

Some of the concept and role atoms in the ABox may be defined names of the TBox. [16]

**ABox assertions**:

An individual assertion can have any of the following forms:

• C(a), called concept assertion,

• R(a, b), called role assertion,

• ¬R(a, b), called negated role assertion,

• a = b, called equality statement

• $a \neq b$, called inequality statement [17]

## 3. The basic description formalism :

A: Atomic Concept

⊤ : Universal Concept

$\perp$ : Bottom concept

¬C : Negation

$C \sqcup D$ : Union

$C \sqcap D$ : Intersection

∃R.C : existential restriction

∀R.C : Universal restriction

where A is an atomic concept, C and D are concepts, and R is a role. [18]

## 4. Semantics :

The semantics of description logics is defined in a model-theoretic way. Thereby, one central notion is that of an interpretation. Interpretations might be conceived as potential "realities" or "worlds." In particular, interpretations need in no way comply with the actual reality. [29]

### 4.1. Interpretation:

An interpretation $I = (\Delta^I, \cdot^I)$ consists of anon-empty set $\Delta^I$, the domain of I, and a valuation function ·I that maps:

1. Each individual $a \in N_I$ to an element $a^I \in \Delta^I$,
2. each concept name $A \in N_C$ to a set $A^I \subseteq \Delta^I$,
3. each role name $p \in N_R$ to a binary relation $p^I \subseteq \Delta^I \times \Delta^I$
4. for the special concepts names, if present, we have $\top^I = \Delta^I$and $\perp^I = \emptyset$.

A DL vocabulary is in fact a FOL signature, that contains no function symbols and no variables, but only constants ($N_I$) and predicates of arities one ($N_C$) and two ($N_R$). Interpretations are just standard Tarski-style interpretations as in FOL[17]

Example 1:

-The role names hasParent, hasFather, hasMother, . . .

– The concept names Parent, Mortal, Male, Female, . . .

– The individual names fatma, mohamed, , …

| DL | Interpretations | DL name |
|---|---|---|
| **A** | $I(A) \subseteq I(\triangle)$ | Primitive concept |
| **R** | $I(R) \subseteq I(\triangle) * I(\triangle)$ | Primitive role |
| ⊤ | $I(\triangle)$ | Top |
| **⊥** | $\phi$ | Bottom |
| **¬C** | $I(\triangle)/I(C)$ | Complement |
| $\boldsymbol{C \sqcup D}$ | $I(C) \cup I(D)$ | Conjunctive |
| $\boldsymbol{C \sqcap D}$ | $I(C) \cap I(D)$ | Disjunctive |
| **∃R.C** | $\{x \mid \exists y. I(x,y) \in I(R) \wedge I(y) \in I(C)\}$ | Existential quantitative |
| **∀R.C** | {x \| ∀y. I(x, y) ∈ I(R) → y∈ I(C)} | Universal quantitative |

**Table 1:**Translation between interpretation and DL.

### 4.1.1. The semantics of TBox Axiomes:

Is defined as one would expect. An interpretation I satisfies an inclusion $C \sqsubseteq D$ if$I(C) \subseteq I(D)$, and it satisfies an equality C ≡ D if $(C) = I(D)$.

If $T$ is a set of axioms, then $I$ satisfies $T$ iff $I$ satisfies each element of $T$. If $I$ satisfies an axiom , then we say that it is a model of this axiom . Two axioms or two sets of axioms are equivalent if they have the same models.

### 4.1.2. Semantics of ABox assertion:

The interpretation$I$ satisfies the concept assertion C(a) if $I(a) \in I(C)$ , and it satisfies the role assertion R(a, b) if $I(a,b) \in I(R)$. An interpretation satisfies the ABox A if it satisfies each assertion in A. In this case we say that I is a model of the assertion or of the ABox.

## 5. DESCRIPTION LOGICS, A BIG FAMILY OF LOGICS :

We will give a general introduction to Big Family Description Logics:

## 5.1.The Basic Description Logics ALC and ALCH:

The semantics of description logics is defined in a model-theoretic way. Thereby, one central notion is that of an interpretation. Interpretations might be conceived as potential "realities" or "worlds." In particular, interpretations need in no way comply with the actual reality.

**Concepts and Roles.** We start with the syntax of *concepts* and *roles*. *ALC* and *ALCH* do not support any role constructors, that is, only role names *p* are roles. On the other hand, they provide the five 'basic' concept constructors: negation $\neg C$, conjunction $C1 \sqcap C2$, disjunction $C2 \sqcup C2$, and existential and universalrestrictions which are expressions of the form $\exists p.C$ and$\forall p.C$, respectively.

**Definition 2 (*ALCH* concepts and roles)**. Each role name $p \in N_R$is a role.

Concepts C obey the following grammar, where $A \in N_C$ and p is a role:

$C, C1, C2 ::= A \mid \neg C \mid C1 \sqcap C2 \mid C2 \sqcup C2 \mid \exists p.C \mid \forall p.C$

In *ALC* and all its extensions, the special names and $\bot$ can be simulated usinga tautological concept of the form $A \sqcup \neg A$and a contradictory concept of theform $A \sqcup \neg A$, respectively, so it makes no difference whether we assume thatthey are present in the signature or not. Concepts of the form $\exists p.\top$are usually called *unqualified* existential restrictions, and written $\exists p$.

**Assertions and Axioms** Using these concepts and role expressions, we can write different kinds of statements. These may also vary in different DLs, but in general, they can be classified into two different kinds:

– At the *extensional level*, we can state that a certain individual participate in some concept, or that some role holds between a pair of individuals; wecall this kind of statement *ABox assertions*. A finite set of this assertions is called an *ABox*.

At the *intentional level*, we can specify general properties of concepts and roles, constraining the way they are interpreted and defining new concepts and roles in terms of other ones. We call these kinds of statements *TBox axioms*, and a *TBox* is a finite set of them. TBox are also called *terminologies*.

ABox assertions and TBox axioms together form a *knowledge base* (KB).

**Ontologies** The term *ontology* is used frequently, but it does not have a fixed, formally defined meaning. It is used both as a synonym for TBox, or as a synonym for KB. We adopt the former use, i.e., an *ontology* is just a terminology.

This meaning is perhaps more frequent, particularly in the context of *ontology based data access* that we will discuss in the next chapter.

We now define the assertions and axioms of the basic DL *ALCH*.

**Definition 3 (*ALCH* ABox assertions and TBox axioms).** *For ALCH,* assertions and axioms are defined as follows.

**ABox assertions:**

– If C is a concept and $a \in N_I$ is an individual, then $C(a)$ is a concept membership assertion.

– If p is a role and $a, b \in N_I$ are individuals, then $p(a, b)$ is a role membership assertion.

– If $a, b \in N_I$ are individuals, then $a \not\approx b$ is an inequality assertion.

**TBox axioms:**

– If C1 and C2 are concepts, then $C1 \sqsubseteq C2$ is a general concept inclusion axiom (GCI).

– If p1 and p2 are roles, then $p1 \sqsubseteq p2$ is a role inclusion axiom (RIA).

Assertions and axioms for ALC are defined analogously, but RIAs $p1 \sqsubseteq p2$ are disallowed.

**Knowledge Bases.** Now we can define *knowledge bases*, which are composed by a set of ABox assertions, the ABox, and a set of TBox axioms, the TBox.

The definition of these components is the same for all DLs.

**Definition 4 (ABoxes, TBoxes, Knowledge bases).** For every DL L, we define:

– An ABox in L is a finite set of ABox assertions in L.

– A TBox in L is a finite set of TBox axioms in L.

– An knowledge base (KB) in L is a pair $K = < A, T >$, where A is an ABox in L and T is a TBox in L.

**TBox:**

$Animal \sqsubseteq LivingThing$

$Donkey \equiv Animal \sqcap \forall hasParent.Donkey$

$Horse \equiv Animal \sqcap \forall hasParent.Horse$

$Mule \equiv Animal \sqcap \exists hasParent.Horse \sqcap \exists hasParent.Donkey$

$\exists hasParent.Mule \sqsubseteq \bot$

**ABox:**

Horse(Mary)

Mule(Peter)

Donkey(Sven)

hasParent(Peter, Mary)

hasParent(Peter, Carl)

hasParent(Sven, Hannah)

hasParent(Sven, Carl) [30]

**Definition 5 (semantics of** ALCH **concepts).** Let I = (ΔI, ·I) be an interpretation. The function ·I is inductively extended to all ALCH concepts as *follows:*

$(\neg C)^I = \Delta^I \setminus C^I$

$C1 \sqcap C2)^I = C1^I \cap C2^I$

$(C1 \sqcup C2)^I = C1^I \cup C2^I$

$(\exists p.C)^I = \{d \mid \exists d'.(d,d')^I \in p^I \wedge (d')^I \in C^I\}$

$(\forall p.C)^I = \{d \mid \forall d'.(d,d')^I \in p^I \rightarrow (d')^I \in C^I\}$

Now that we have fixed the semantics of concepts and roles, we can define the satisfaction of assertions and axioms. This is done in a natural way. The symbol ⊑in the TBox axioms is understood as an relation. That is, a concept inclusion $C1 \sqsubseteq C2$ indicates that every object that is $C1$ is also $C2$, or to be more precise, that every object that participates in the interpretation of concept $C1$ also participates in the interpretation of concept $C2$. Similarly, a role inclusion $p1 \sqsubseteq p2$ indicates that every pair of objects that participates in $p1$ also participates in $p2$. Concept and role membership assertions in the ABox simply state that (the interpretation of) an individual participates in (the interpretation of) a concept, and that a pair of individuals participates in a role, respectively.

An assertion of the form $a \not\approx b$states that the individuals $a$ and $b$ cannot be interpreted as the same domain element. This is closely related to the *unique name assumption (UNA)*, sometimes made in related formalisms. Under the UNA, each interpretation $I$ must be such that $a^I = b^I$only if $a = b$, that is, one domain element cannot be the interpretation of two different individuals. In DLs the common practice is not to make the UNA. This setting is more general and, if desired, the UNA can be enforced by adding assertions $a \not\approx b$for eachrelevant pair of individuals.

### 5.2.Expressive and Lightweight DLs:

The term lightweight DLs refers to logics that are based on fragments of ALC and restrict its expressivity to achieve lower complexity, enabling the realization of efficient and scalable algorithms. The most prominent lightweight DLs are the DL-Lite and EL families underlying the OWL QL and RL profiles, respectively.

**5.3.The SH Family:**

*SHOIQ* is a very expressive DL that is closely related to the Web Ontology Language standard known as OWL-DL . *SHOIQ* supports the vast majority of the common DL constructors, and hence most popular DLs can be defined assublogics of it.

**Definition 8 (**SHOIQ **concepts and roles).** Atomic concepts B, concepts C and (atomic) roles P, S obey the following grammar, where $a \in N_I, A \in N_C, p \in N_R, and\ n \geq 0$:

$B ::= A \mid \{a\}$

$C, C1, C2 ::= B \mid \neg C \mid C1 \sqcap C2 \mid C1 \sqcup C2 \mid \exists P.C \mid \forall P.C \mid \geq n\, S.C \mid \leq n\, S.C$

$P, S ::= p \mid p^-$

The inverse of $p \in N_R$ is $p^-$, and the inverse of $p^- is\ p$. To avoid expressions such as$(p^-)^-$, we denote by $Inv(P)$ the inverse of the role P. Concepts of the form {a} are called nominal, while concepts $\geq n\, S.C$ and $\leq n\, S.C$ are called(qualified) number restrictions (NRs).

If a number restriction is of the form $\geq$ n S. or $\leq$ n S., it is called unqualified and can be written simply $\geq$ n S. or $\leq$ n S.

In addition to the new role constructor $p^-$and the new concept constructors$\{a\}$, $\geq$ n S. and $\leq$ n S, *SHOIQ* extends *ALCH* with another kind of axioms.

**Definition 9 (**SHOIQ **ABox assertions and TBox axioms).** ABox assertions, GCIs and RIAs in SHOIQ are defined analogously to ALCH, but allowing for SHOIQ concepts and roles where applicable. In addition to GCIs and RIAs, SHOIQ TBox allow for transitivity axioms (TAs), which are expressions trans(P) where P is a role.

Knowledge bases in SHOIQ are defined essentially as for ALCH, but must satisfy an additional constraint: the roles S that occur in the number restrictions $\geq n\, S.C\ and\ \leq n\, S.C$ must be simple, which means that they can not be impliedby roles occurring in transitivity axioms. Intuitively, this allows us to count only the direct neighbors of a node, but not nodes that are further away inane interpretation. It is well known that dropping this restriction results in an undecidable logic.

To formalize the notion of simple roles, we use the relation $\sqsubseteq^T$, which relates each pair of roles $P1$, $P2$ such that $P1^I \subseteq P2^I$holds in every interpretation thatsatisfies $T$ .

**Definition 10 (simple roles,** SHOIQ **knowledge bases).** For a TBox T , we denote by$\sqsubseteq^T$the reflexive transitive closure of $\{(P1, P2) | P1 \sqsubseteq P2\ or Inv(P1) \sqsubseteq Inv(P2)$ is in T $\}$; we usually write $\sqsubseteq^T$ in infix notation. A role S is simplew.r.t. T , if there is no P such that P $\sqsubseteq^T$S and trans(P) ∈ T .

A knowledge base in SHOIQ is a pair $K = \prec A, T \succ$consisting of an ABox A and a TBox T, such that all roles S occurring in a number restriction $\geq nS.Cor \leq nS.C$ are simple w.r.t. T.

**Example about TBox**:

$FirstYearCourse \sqsubseteq \forall isTaughtBy.Professor$
$MathCourse \sqsubseteq \exists isTaughtBy.\{949352\}$
$AcademicStaffMember \sqsubseteq \exists teaches.UndergraduateCourse$
$course\ v \sqsubseteq \geq 1\ isTaughtBy\ Department \sqcap \geq 10\ hasMemberu \sqcap \leq 30\ hasMember$
$course \sqsubseteq \neg staffMember$
$peopleAtUni \equiv staffMember\ t\ student$
$facultyInCS \equiv faculty \sqcap \exists belongsTo.\{CSDepartment\}$
$adminStaff \equiv staffMember \sqcap \neg(faculty\ t\ techSupportStaff\ )$[31]

**Semantics of SHOIQ.** To give semantics to SHOIQ knowledge bases, we need to define the semantics of the new concept and role constructors.

**Definition 11 (semantics of concepts and roles in** SHOIQ**).** For every interpretation I, we define:

$(p^-)^I = \{(d', d) \mid (d, d') \in p^I\}$
$\{a\}^I = \{a^I\}$
$(\geq n\ S.C)^I = \{d \mid \{d' \mid (d, d') \in S^I \wedge d' \in C^I\}| \geq n\}$
$(\leq n\ S.C)^I = \{d \mid \{d' \mid (d, d') \in S^I \wedge d' \in C^I\}| \leq n\}$

We also need to define the semantics of assertions and axioms, on which the semantics of knowledge bases depends. Sublogics of SHOIQ. There are many well known DLs that contain ALC, and extend it with some of the features of SHOIQ. The logic S is the extension of ALC with transitivity axioms. Both ALC and S can be extended with the additional features as follows: the presence of the letter H indicates that RIAs are allowed, and the additional letters I, O and Q respectively denote the presence of inverses as a role constructor,

of nominals, and of number restrictions. Some of these extensions, are listed in Table 2. The best known of them is SHIQ, which is closely related to the OWL-Lite standard.

| DL | Tas | RIAs | Inverses | nominals | NRs |
|---|---|---|---|---|---|
| ALC | | | ✓ | | |
| ALCI | | | | | ✓ |
| ALCHQ | ✓ | ✓ | | | |
| SH | ✓ | ✓ | ✓ | | ✓ |
| SHIQ | ✓ | ✓ | | ✓ | ✓ |
| SHIQ | ✓ | ✓ | ✓ | ✓ | |
| SHOI | | ✓ | ✓ | ✓ | ✓ |
| ALCHOIQ | | | | | |
| SHOIQ | ✓ | ✓ | ✓ | ✓ | ✓ |

**Table 2:**Some expressive DLs between ALC and SHOIQ.

### 5.4. The SR Family:

SROIQ is a rather well known extension of SHOIQ, which was proposed as the basis for the Web Ontology Language standard OWL 2 . Its sublogics SRIQ, SROQ and SROI are analogous to SHIQ, SHOQ and SHIO.

The most prominent feature of the logics in the SR family are complex role inclusion axioms of the form P1 ° ····· °$P \subset P$. It is also possible to explicitlystate certain properties of roles like transitivity, (ir)reflexivity and disjointness.Some of these additions increase the expressivity of the logic, while others are just 'syntactic sugar' and are intended to be useful for ontology engineering. Were call the definition of SROIQ from , borrowing some notation from .As usual, we start by defining concepts and roles.

**Definition 13 (**SROIQ **concepts and roles).** In SROIQ, we assume that the signature contains a special role name U, called the universal role. Atomic concepts B, concepts C, atomic roles P, S, and roles R, obey the follow in grammar, where $a \in N_I, A \in N_C, p \in N_R$:

$B ::= A \mid \{a\}$

$C, C1, C2 ::= B \mid \neg C \mid C1 \sqcap C2 \mid C1 \sqcup C2 \mid \forall P.C \mid \exists P.C \mid \forall P.C \mid \exists U.C \mid \forall U.C \mid$

$\geq nS.C \mid \leq nS.C \mid \exists S.Self$

$P, S ::= p \mid p^-$

$R, R1, R2 ::= S \mid R1 \circ R2$

We denote by NR the set of all atomic roles $\{p, p^- \mid p \in N_R\}$. Non-atomic rolesof the form $P1 \circ \cdots \circ Pn$ may be called role chains.Note that U may only occur in universal and existential restrictions. SROIQ supports some assertions and axioms that were not present in the other

logics so far. In particular, the rich role axioms are its main distinguishing feature.

**Definition 14 (**SROIQ **ABox assertions, TBox axioms).** In SROIQ, ABox assertions are as follows:

– If C is a concept and $a \in N_I$ an individual, then C(a) is a concept membership assertion.

– If P is an atomic role and $a, b \in N_I$ are individuals, then $P(a, b)$ is a (positive) role membership assertion.

– If S is an atomic role and $a, b \in N_I$ are individuals, then $\neg S(a, b)$ is a (negative) role membership assertion.

– If a, $b \in N_I$ are individuals, then $a \not\approx b$ is an inequality assertion.

TBox axioms are GCIs, defined as usual, as well as:

– If R is a role chain and P is an atomic role, then $R \sqsubseteq P$ is a complex role inclusion axiom (CRIA).

To define SROIQ knowledge bases, we need some additional conditions that were designed to ensure decidability. In particular, we need a notion called regularity and, similarly to SHOIQ, we must define simple roles, and restrict the roles occurring in certain positions to be simple. As for SHOIQ, we define a relation$\sqsubseteq^T$that contains the pairs R, P of roles such that $R^I \subseteq P^I$for each model I of T , but the definition is more involved due to the presence of role chains in the role inclusion axioms.

## 6. Reasoning Services:

The basic reasoning services in DL systems is to test for the satisfiability of a concept or a TBox, to test whether the information specified in it contains logical contradictions or not.

In case the TBox contains a contradiction, any consequence can follow logically from the TBox. Moreover, if a TBox is not satisfiable, the specified information can hardly capture the intended meaning from an application domain. To test for satisfiability is often a first step for a user to check whether a TBox models something “meaningful”.

The concept description C is satisfiable iff it has a model, iff there exists an interpretation$I$ such that $I(C) \neq \emptyset$. A TBox $T$ is satisfiable iff it has a model, an interpretation that satisfies all GCIs in $T$ .

If a concept or a TBox is not satisfiable, it is called unsatisfiable. [19]

## 6.1.Reasoning services of TBox Axiomes:

C, D two concept descriptions and $T$ is TBox. The concept description C is subsumed by the concept description D w.r.t. $(C \sqsubseteq D)$:

**iff $I(C) \subseteq I(D)$ holds in every model$I$ of $T$ .**

Two concepts C, D are equivalent w.r.t. T $(C \equiv D)$:

**iff $I(C) = I(D)$ holds for every model $I$ of $T$ .**

## 6.2.Reasoning services of ABox assertion:

We can test for the absence of contradictions in ABoxes. An ABox $A$ is consistent w.r.t. a TBox$T$ , iff it has a model that is also a model for . The individual$i$ is an instance of the concept description C w.r.t. an ABox A we write $C(i)$ :

iff $I(i) \in I(C)$for all models $I$of A.

## 7. Reasoning Techniques:

There are three main reasoning approaches for the DLs that underlie OWL. For the expressive DLs, which offer all Boolean concept constructors, most reasoning services can be reduced to consistence of an ABox w.r.t. a TBox in polynomial time. In presence of full negation we can devise the following polynomial time reductions.

• Equivalence can be reduced to subsumption: C ≡ D iff C ⊑ D and D ⊑C.

• Subsumption can be reduced to (un)satisfiability: C ⊑ D iff C ⊓ ¬D is unsatisfiable w.r.t. T .

• Satisfiability can be reduced to consistency: C is satisfiable w.r.t. T iff the ABox {C(a)} is consistent w.r.t. T .

• The instance problem can be reduced to (in)consistency: $T \cup A \Vdash C(a)\ iff\ A \cup \{\neg C(a)\}$ is inconsistent w.r.t. T . [19]

## 8. FOL:

First-order logic (FOL) is provides a way of representing information like the : Mary is a person. Whereas propositional logic assumes world contains facts, first-order logic (like natural language) assumes the world contains: Constants and Predicates as jhon is person and function as Italian(fatherOf (Mario)).

FOL is the basis of any query language for relational databases, to the best of our knowledge, the most expressive class of queries that go beyond instance checking, and for which decidability of query answering has been proved in DLs, is the class of union of conjunctive queries (UCQ) . This restriction on the query language may constitute a serious limitation to the adoption of DLs technology in information management tasks, such as those required in Semantic Web applications.

### 8.1. The table for FOL:

Many description logics are decidable fragments of first-order logic (FOL$\Phi$), also known as first-order predicate calculus (FOPC), and many of two-variable logic or guarded logic, however, some description logics have more features than first-order logic.

| **FOL** | **DL** |
|---|---|
| $\Phi(A,x)$ | $A(x)$ |
| $\Phi(T,x)$ | $T$ |
| $\Phi(\bot,x)$ | $\bot$ |
| $\Phi(\neg\mathbf{C},x)$ | $\neg\Phi(C,x)$ |
| $\Phi(C \sqcap D,x)$ | $\Phi(C,x) \wedge \Phi(D,x)$ |
| $\Phi(C \sqcup D,x)$ | $\Phi(C,x) \vee \Phi(D,x)$ |
| $\Phi(\forall R.C,x)$ | $\forall y\,(R(x,y) \rightarrow \Phi(C,y))$ |
| $\Phi(\exists R.C,x)$ | $\exists y R(x,y) \wedge \Phi(C,y)$ |

**Table3:**Translation between DL and FOL.

Despite of the feasibility of direct translation between FOL and DL, guaranteeing complete and terminating reasoning requires a different transformation, such as the structural transformation. The structural transformation is based on a conjunction normal form, which replaces FOL sub formulae with new predicates, for which it also provides definitions. A major advantage of the structural transformation is that it avoids the exponential size growth of the clauses. [20]

## 9. DL-Lite:

The DL-Lite family is testified by the fact that it forms the basis of OWL 2 QL, one of the three profiles of OWL. The OWL 2 profiles are fragments of the full OWL 2 language that have been designed and standardized for specific application requirements. According to the official W3C profiles document, the purpose of OWL 2 QL is to be the language of choice for applications that use very large amounts of data and where query answering is the most important reasoning task.

DL-Lite family of description logics was proposed with the aim of capturing typical conceptual modeling formalisms, such as UML class diagrams and ER models ,while maintaining good computational properties of standard DL reasoning tasks . [21]

## 10.DAML+OIL and Description Logics:

DAML+OIL is a language developed as part of the US DARPA Agent Markup Language (DAML) programme and OIL (the Ontology Inference Layer) , developed by a group of (mostly) European researchers. This language has a syntax based on RDF Schema (and thus is Web compatible), and it is based on common ontological primitives from Frame Languages (which supports human understandability). Its semantics can be defined via by a translation into the expressive DL SHIQ.

DAML+OIL is designed to describe the structure of a domain; it takes an object oriented approach, describing the structure in terms of classes and properties. An ontology consists of a set of axioms that assert, e.g., subsumption relationships between classes or properties. Asserting that resources (pairs of resources) are instances of DAML+OIL classes (properties) is left to RDF, a task for which it is well suited. When a resource res is an instance of a class C we say that rec has type C. From a formal point of view, DAML+OIL can be seen to be equivalent to a very expressive description logic (DL), with a DAML+OIL ontology corresponding to a DL terminology (Tbox). As in a DL, DAML+OIL classes can be names (URIs) or expressions, and a variety of constructors are provided for building class expressions. The expressive power of the language is determined by the class (and property) constructors supported, and by the kinds of axiom supported. [22]

## 11. Conclusion:

The emphasis in DL research on a formal, logic-based semantics and a thorough investigation of the basic reasoning problems, together with the availability of highly optimized systems for very expressive DLs, makes this family of knowledge representation formalisms an ideal starting point for defining ontology languages for the Semantic Web. The reasoning services required to support the construction, integration, and evolution of high quality ontologies are provided by state-of-the-art DL systems for very expressive languages.

To be used in practice, these languages will, however, also need DL-based tools that further support knowledge acquisition (i.e., building ontologies), maintenance (i.e., evolution of ontologies), and integration and inter-operation of ontologies. First steps in this direction have already been taken. For example, OilEd is a tool that supports the development of OIL and DAML+OIL ontologies, and ICom is a tool that supports the design and integration of entity-relationship and UML diagrams. On a more fundamental level, so-called non-standard inferences that support building and maintaining knowledge bases (like computing least common subsumers, unification, and matching) are now an important topic of DL research . All these efforts aim at supporting users that are not DL-experts in building and maintaining DL knowledge bases.

# Chapter 3:
# OBDA

## 1. Introduction:

**Ontology-based data access (OBDA)** is a recent paradigm for accessing and integrating data sources through an ontology that acts as a conceptual, integrated view of the data, and declarative mappings that connect the ontology to the data sources.[37] an ontology defines a high-level global schema of (already existing) data sources and provides a vocabulary for user queries. An OBDA system rewrites such queries and ontologies into the vocabulary of the data sources and then delegates the actual query evaluation to a suitable query answering system such as a relational database management system. [23]

The first experiences in the application of the OBDA framework in real-world scenarios have shown that the semantic distance between the conceptual and the data layer is often very large, because data sources are mostly application-oriented: this makes the definition, debugging, and maintenance of mappings a hard and complex task. Such experiences have clearly shown the need of tools for supporting the management of mappings. [24]

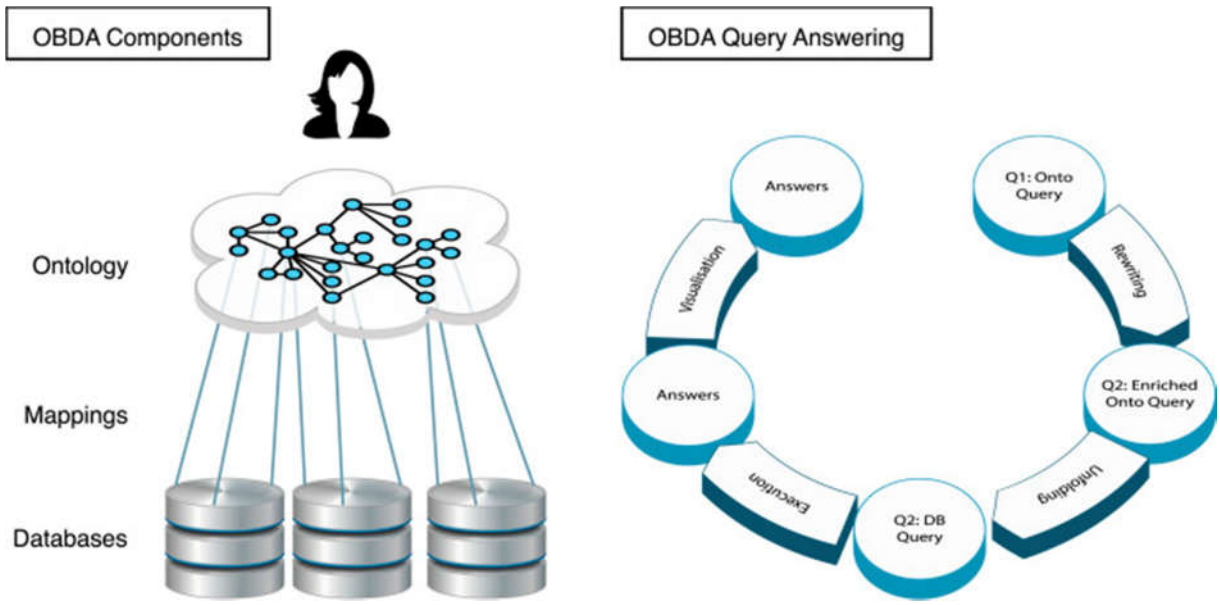


**Figure 11:**Ontology Based Data Access (OBDA).

## 2. Ontology-based data access framework:

In last decade the terms related to Semantic Web become significant elements in the efficient way of information retrieval, processing and supporting availability of machine readable data. An ontology offers a wide spectrum of its application for data access. Ontology-Based Data Access is regarded as a key ingredient for the new generation of information systems, especially for Semantic Web applications that involve large amounts of

data. OBDA system uses an ontology as a conceptual schema of the subject domain, and as a basis of the user interface for SQL database systems. we will present overview of OBDA [40]wich has gained attention in recent years for providing access to large volumes of data by using ontologies as a conceptual layer and exploring their ability to describe domains and deal with data incompleteness. This is done through mappings that connect the data in the database to the vocabulary of the ontology. [25]

## 3. Mapping between the data and the ontology:

One important aspect in OBDA concerns the construction of a system specification, defining the ontology and the mappings over an existing set of data sources. Mappings are indeed the most complex part of an OBDA specification, since they have to capture the semantics of the data sources and express such semantics in terms of the ontology. More precisely, a mapping is a set of assertions, each one associating a query φ(x) over the source schema with a query ψ(x) over the ontology. The intuitive meaning of a mapping assertion is that all the tuples satisfying the query φ(x) also satisfy the query ψ(x). We write a mapping assertion as $\varphi(x) \rightsquigarrow \psi(x)$.

As an example, consider $tabP(x, y, z) \rightsquigarrow person(x)$, name(x, y), which maps the ontology predicates person and name to the database relation tabP, thus indicating how ontology instances can be constructed from the data retrieved at the sources.

In the following, we assume to have three pair wise disjoint, countably infinite alphabets: an alphabet$\Gamma T$ of ontology predicates, an alphabet $\Gamma S$of source schema predicates, and an alphabet $\Gamma C$ of constants.

**Data sources(S):** A data source S consists of a data schema and a number of corresponding data instances. which are external and independent (possibly multiple and heterogeneous). A typical example for a data source is a relational or semi-structured database.

A source schema S is a relational schema containing relations in ΓS, possibly equipped with integrity constraints (ICs). A legal instance D for S is a database for S that satisfies the $IC$s of S. Const(D) are the set of constants occurring in D. Given a first-order sentence α, we write S |= α if for each database D legal for S, the interpretation $I\ D \Vdash \alpha$, where $I\ D$ is the interpretation induced by D.

We consider simple schemas, i.e., relational schemas without $ICs$, and FD schemas, i.e., simple schemas with functional dependencies (FDs) . and by a CQ over a source schema S that mean a CQ over the alphabet of S. With $\varphi(\sim x)$ which denote a CQ with free variables $\sim x$. The number of variables in $\sim x$ is the arity of the query. A Boolean CQ is a CQ without free variables. Given a CQ q over S and a legal instance D for S, $eval(q, D)$ denotes the evaluation of q over D.

**Ontology**: In the context of OBDA, it is usual to consider the ontology to be a Description Logic (DL) ontology ( equivalently, an OWL ontology). A DL ontology consists of a finite set of axioms that are usually in the form of set inclusions between two (possibly complexly defined) concepts that represent classes of objects. The ontology captures general knowledge about the domain of interest, such as generalizations, relational links, etc.Ontology provides a unified, conceptual view of the managed information.

In particular, a DL ontology $O$ is pair $hT$ , $Ai$, where T is the TBox and A is the ABox. O, T. we do not interpret ontologies under the Unique Name Assumption. $Mod(O)$ are the set of models of O, and with $O \models \alpha$ the fact that O entails a sentence α. Also, by ontology inconsistency we mean the task of deciding whether Mod(O) = ∅, and by instance checking the task of deciding whether $O \models \beta$, where β is a ground atom. By CQs over O we mean CQs over the alphabet of the TBox of O, and by CQ entailment the task of checking whether $O \models q$, where q is a Boolean CQ. we consider DLs that are the logical basis of the W3C standard OWL and of its profiles, SROIQ , which underpins OWL, DL-LiteR , which is the basis of OWL 2 QL, RL .

**Mappings :** Mappings associate data from the data sources with concepts in the ontology. A mapping m has the form:

$$\boldsymbol{m}: \boldsymbol{\varphi}(\mathbf{x}) \rightsquigarrow \boldsymbol{\psi}(\mathbf{x})$$

**$\boldsymbol{\varphi}(\mathbf{x})$:** is a query over the data sources. called the body of m.**$\boldsymbol{\psi}(\mathbf{x})$** is an element of the ontological vocabulary. called the head of m. The number of variables in x is the arity of the mapping assertion. Given a mapping assertion m, we also use FR(m) do denote the frontier variables x, head(m) to denote the query ψ(x), and body(m) to denote the query φ(x). We also remark that queries used in our mappings, besides variables, may contain constants from $\Gamma C$. A mapping M from S to T is a finite set of mapping assertions from S to Thereinafter M will always denote a mapping. In principle, φ(x) and ψ(x) can be specified in generic query

languages. The literature on data integration and OBDA has mainly considered φ(x) expressed in first-order logic (FOL), and ψ(x) expressed as a CQ.[26]

## 4. Query rewriting:

Query rewriting is an important technique for answering queries over data described using ontologies. In query rewriting the input, a conjunctive query (CQ) q and an ontology O, is transformed into a new data-log query that captures all answers of q over O and *any* dataset D. This process can be time-consuming as it is of high computational complexity. In many real-world applications, this can be particularly problematic as they involve frequent and relatively small modifications on quite large ontologies. Hence, a drawback of most of modern query rewriting systems is that every time the initial ontology is modified, e.g. when new axioms are added or existing ones removed, they compute a new rewriting from scratch. In this paper, we study the problem of computing a rewriting for a CQ over an ontology that has been modified. We do this by reusing the information obtained by the extraction of some previous rewriting with the goal of performing the least possible computations. We study the problem theoretically, present detailed algorithms for both ontology revision and ontology contraction and finally, present an extensive experimental evaluation using the well-known query rewriting systems Requiem. [27]

### 4.1. Query Rewriting Algorithms:

In this section we describe the query rewriting algorithms, We use the well-known notions of constants, variables, function symbols, terms, and atoms of first-order logic. A Horn clause C is an expression of the form $D_0 \leftarrow D_1 \wedge \ldots \wedge D_n$, where each $D_i$ is an atom. The atom $D_0$ is called the head, and the set $\{D_1, \ldots, D_n\}$ is called the body. We require that all the variables occurring in the head of C occur at least in one ofits body atoms. For instance, the expression teaches$(x, f(x)) \leftarrow Professor(x)$ is a Horn clause.

A conjunctive query (CQ) Q posed over an ontology O is a Horn clause whose head predicate does not occur in O, and whose body predicates are class and property names occurring in O. A union of conjunctive queries (UCQ) over O is a set of conjunctive queries over O with the same head up to variable renaming . A tuple of constants $a$ is a certain answer to a UCQ Q over O and a set of instance data $A$ $iff$ $O \cup A \cup Q \vDash Q_P(a)$,where $Q_P$ is the head predicate of Q, and Q is considered to be a set of universally quantified implications

with the usual first-order semantics. The set of all answers to Q over O and A is denoted by $ans(Q, O \cup A)$. Given a conjunctive query Q and an ontology O, a query $Q_O$is said to be a rewriting of Q w.r.t. O if $ans(Q, O \cup A) = ans(Q_O, A)$ for every A. Both algorithms compute the rewriting $Q_O$ of a given query Q w.r.t. a DL-LiteR ontology O. DL-LiteR is the basis for OWL 2 QL. [28]

**4.1.1. The algorithm 1:**

The algorithm computes $Q_o$by using the axioms of *O* as rewrite rules and applying them to the body atoms of *Q*. The algorithm is shown in Algorithm 1. The partial function ref takes as input an axiom *α*and an atom *D*, and returns an atom ref(*D; α*) as follows.

$If D = A(x) then we have that\ (i)\ if \alpha = B \sqsubseteq A, then ref(D;\ \alpha) = B(x);$
$(ii) if \alpha = \exists P \sqsubseteq A, then ref(D;\ \alpha) = P\ (x\ ;-);\ and (iii) if \alpha = \exists P^{-} \sqsubseteq A, then$
$ref(D;\ \alpha) = P\ (-\ ;\ x).$
$If D = P\ (x;-), then we have that\ (i)\ if \alpha = A \sqsubseteq \exists P\ , then ref(D;\ \alpha) = A(x);$
$(ii)\ if \alpha = \exists S \sqsubseteq \exists P\ , then ref(D;\ \alpha) = S(x;);\ and\ (iii)\ if \alpha = \exists S^{-} \sqsubseteq \exists P\ ,$
$then ref(D;\ \alpha) = S(-;\ x).$
$If D = P\ (-\ ;\ x), then we have that\ (i)\ if \alpha = A \sqsubseteq \exists P^{-}, then ref(D;\ \alpha) = A(x);$
$(ii)\ if \alpha = \exists S \sqsubseteq \exists P^{-}, then ref(D\ ;\ \alpha) = S(x;-);\ and\ (iii)\ if \alpha = \exists S - \sqsubseteq \exists P^{-}\ ,$
$then ref(D;\ \alpha) = S(\ -;\ x).$
$If D = P\ (x;\ y), then we have that\ (i)\ if either \alpha = S \sqsubseteq P or \alpha = S^{-} v P^{-}\ ,$
$then ref(D;\ \alpha) = S(x;\ y);\ and\ (ii)\ if either \alpha = S v P - or \alpha = S - v P\ , then$
$ref(D;\ \alpha) = S(y;\ x).$[29]

**4.1.2.The algorithm 2:**

This algorithm is based on query rewriting every time the initial ontology is modified when new axioms are added By usingConjunctive query :

**Input**: Conjunctive query $Q$, $\mathrm{DL-Lite_R}$ontology $O$
$Q_O = \{Q\}$;
**repeat**
 **foreach** *query* $Q' \in Q_O$**do**
  (reformulation) **foreach** *atom D in Q'***do**
   **foreach** *axiom* $\alpha \in O$**do**
    **if** *α is applicable to D* **then**

$$Q_O = Q_O \cup \{Q'[D/ref(D;\ \alpha)]\};$$

**end**

**end**

**end**

(reduction) **forall** *atoms* $D_1$*;* $D_2$*in Q'* **do**

**if** *D*1 *and D*2 *unify* **then**

$\sigma$= MGU($D_1$*;* $D_2$);

$Q_O = Q_O \cup \{\lambda(Q'\sigma)\};$

**end**

**end**

**end**

**until** *no query unique up to variable renaming can be added to* $Q_O$;

**return** $Q_O$; [30]

Example:

$Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$

$\text{Doctor} \sqsubseteq \exists \text{treats.Patient}$

$\text{Consultant} \sqsubseteq \text{Doctor}$

D= treats(x,f(x))

D= Patient (f(x))

$\text{treats}(x, f(x)) \leftarrow \text{Doctor}(x)$

$\text{Patient}(f(x)) \leftarrow \text{Doctor}(x)$

$\text{Doctor}(x) \leftarrow \text{Consultant}(x)$

$Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x))$

$Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x)$

$Q(x) \leftarrow \text{Doctor}(x)$

$Q(x) \leftarrow \text{Consultant}(x)$

For DL-Lite, result is a union of conjunctive queries :

$$Q(x) \leftarrow (\text{treats}(x, y) \wedge \text{Patient}(y)) \vee \text{Doctor}(x) \vee \text{Consultant}(x)$$

## Mapping:

- Data can be stored/left in RDBMS
- Relationship between ontology and DB defined by mappings, e.g.:

$$\begin{aligned} \text{Doctor} &\mapsto \text{SELECT Name FROM Doctor} \\ \text{Patient} &\mapsto \text{SELECT Name FROM Patient} \\ \text{treats} &\mapsto \text{SELECT DName, PName FROM Treats} \end{aligned}$$

UCQ translated into **SQL query**:

$$Q(x) \leftarrow (\text{treats}(x, y) \wedge \text{Patient}(y)) \vee \text{Doctor}(x) \vee \text{Consultant}(x)$$

```
SELECT Name FROM Doctor UNION
SELECT DName FROM Treats, Patient WHERE PName=Name
```

### 4.1.3. Most General Unifier:

A substitution, $\sigma$ is a *most general unifier* (mgu) of a set of expressions $E$ if it unifies $E$ , and for any unifier, $\omega$ of $E$, there is a unifier, $\lambda$ such that $\omega = \sigma \circ \lambda$.

The idea is that $\sigma$ is less specific than (technically, no more specific than) $\omega$, that is, we can substitute for some of the variables of $\sigma$ and get $\omega$. Note that there can be more than one most general unifier, but such substitutions are the same except for variable renaming.

In the above example, $\sigma 2$ is the mgu of the set of expressions. We can see that : $\sigma = \sigma_2 \circ [a/x]$There is a simple algorithm for finding the most general unifier of a set of expressions. First, we need to define the *disagreement set* of a set of expressions. This is found by (textually) finding the first symbol starting from the left that is not the same in every expression and extracting the sub expressions that begin with that symbol at that position in each expression of the set. The resulting set of sub expressions is the disagreement set. For example, the disagreement set $for\ \{P(x, y, a), P(x, f(g), b), P(x, f(g), x) is\ \{, f(g)\}$ .

If ref($D;\ \alpha$) is defined for $\alpha$and $D$, we say that $\alpha$is *applicable* to $D$. The expression $Q[D/D']$ denotes the CQ obtained from $Q$ by replacing the body atom $D$ with a new atom $D'$ The function MGU takes as input two atoms and returns their most general unifier. The function $\lambda$takes as input a CQ $Q$ and returnsa new CQ obtained by replacing each variable that occurs only once in $Q$ withthe symbol " $-$ ".

Starting with the original query $Q$, Algorithm continues to produce queries until no new queries can be produced. In the *reformulation* step the algorithm rewrites the body atoms of a given query $Q'$by using applicable ontology axiomsas rewriting rules, generating a new query for every atom reformulation. Then,in the *reduction* step the algorithm produces a new query $\lambda(Q'\sigma)$ for each pairof body atoms of $Q'$that unify. [31]

## 5.Query answering:

Ontological databases extend traditional databases with ontological constraints. This technology is crucial for many applications such as semantic data publishing and integration as well as model-driven database design. For many classes of ontological constraints, query answering can be solved via query rewriting. In particular, given a conjunctive query and a set of ontological constraints, the query is compiled into a first-order query, called the perfect rewriting, that encodes the intentional knowledge implied by the constraints. Then, for every database D, the answer is obtained by directly evaluating the perfect rewriting over D. Since first-order queries can be easily translated into SQL.

## 6.Conception:

We use the conception for presenting application diagrams which illustrate the relationship between application classes

## 6.1. Class diagram in the Unified Modeling Language (UML):

**Class diagram** is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.[33]

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data

modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- ✓ The top compartment contains the name of the class.
- ✓ The middle compartment contains the attributes of the class.
- ✓ The bottom compartment contains the operations the class can execute.[35]

## 6.2.Class diagram for application:

On this page I used the class diagram to illustrate the work of application classes and the relationship between classes:

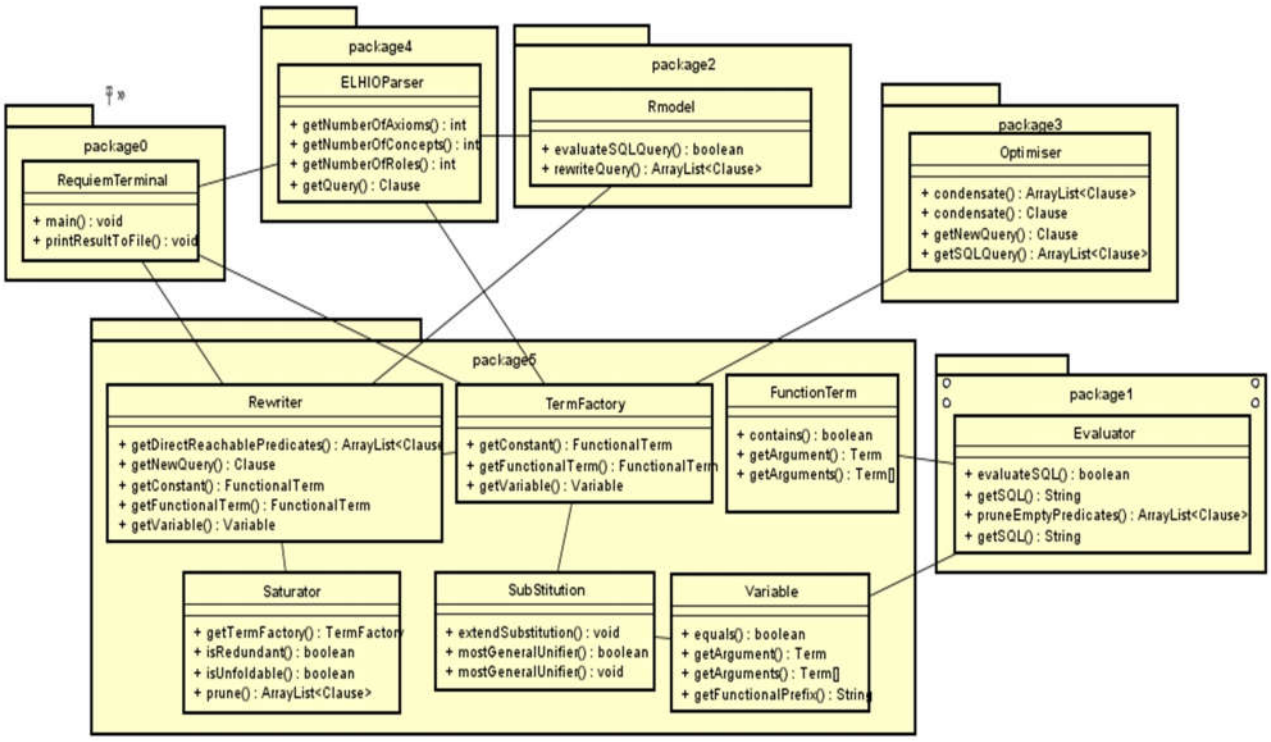


**Figure 12:** Class diagram for OBDA application.

## 7.Use case for application:

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors and represent:

- ✓ Scenarios in which your system or application interacts with people, organizations, or external systems.
- ✓ Goals that your system or application helps those entities (known as actors) achieve.
- ✓ The scope of your system.[37]

## 7.1.Use case diagram components:

To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

• Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

• System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

• Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.[38]

## 7.2.Use case for application:

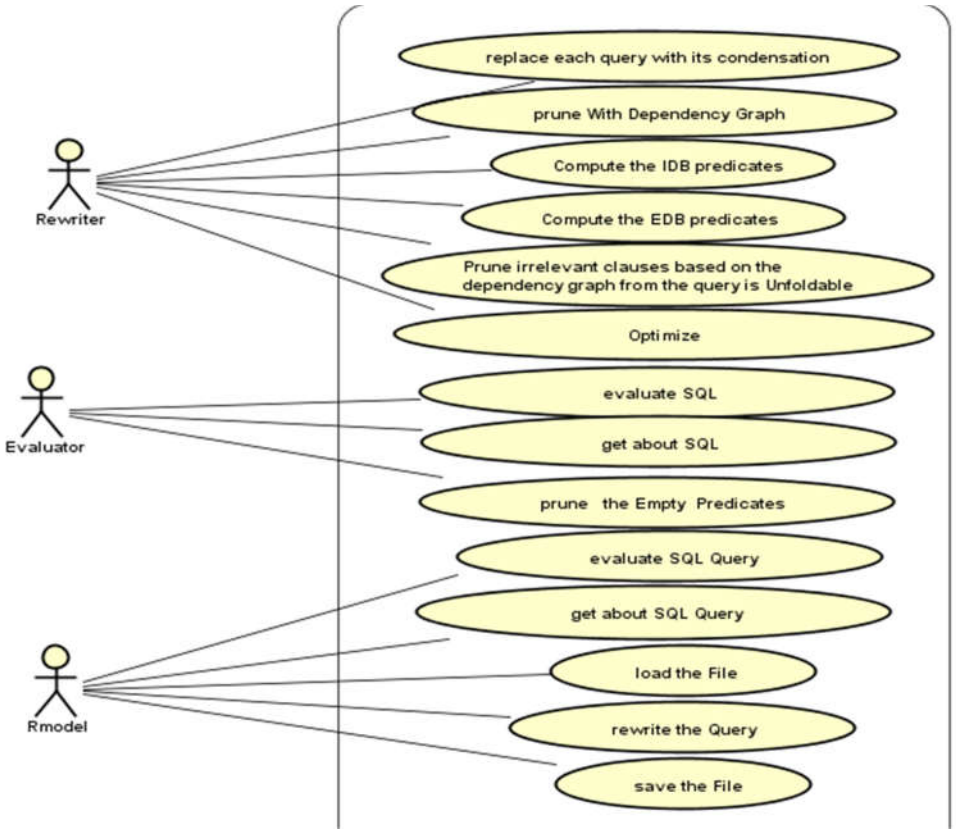

**Figure13:** Use case for OBDA application.

## 8.Diagrame sequence:

Sequence diagram: an “interaction diagram” that models a single scenario executing in a system

• 2nd most used UML diagram (behind class diagram)

• Shows what messages are sent and when[38]

## 8.1.Diagrame sequence for application:

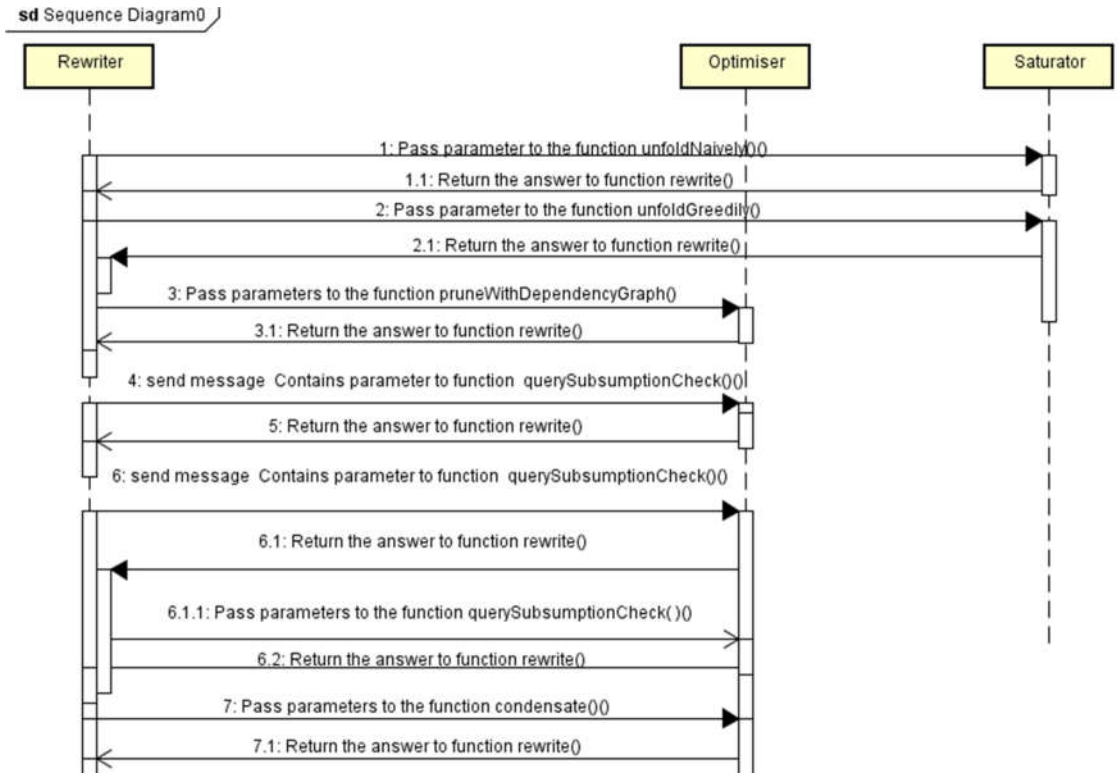


**Figure14:**Diagrame sequence for OBDA application.(Rewriter)

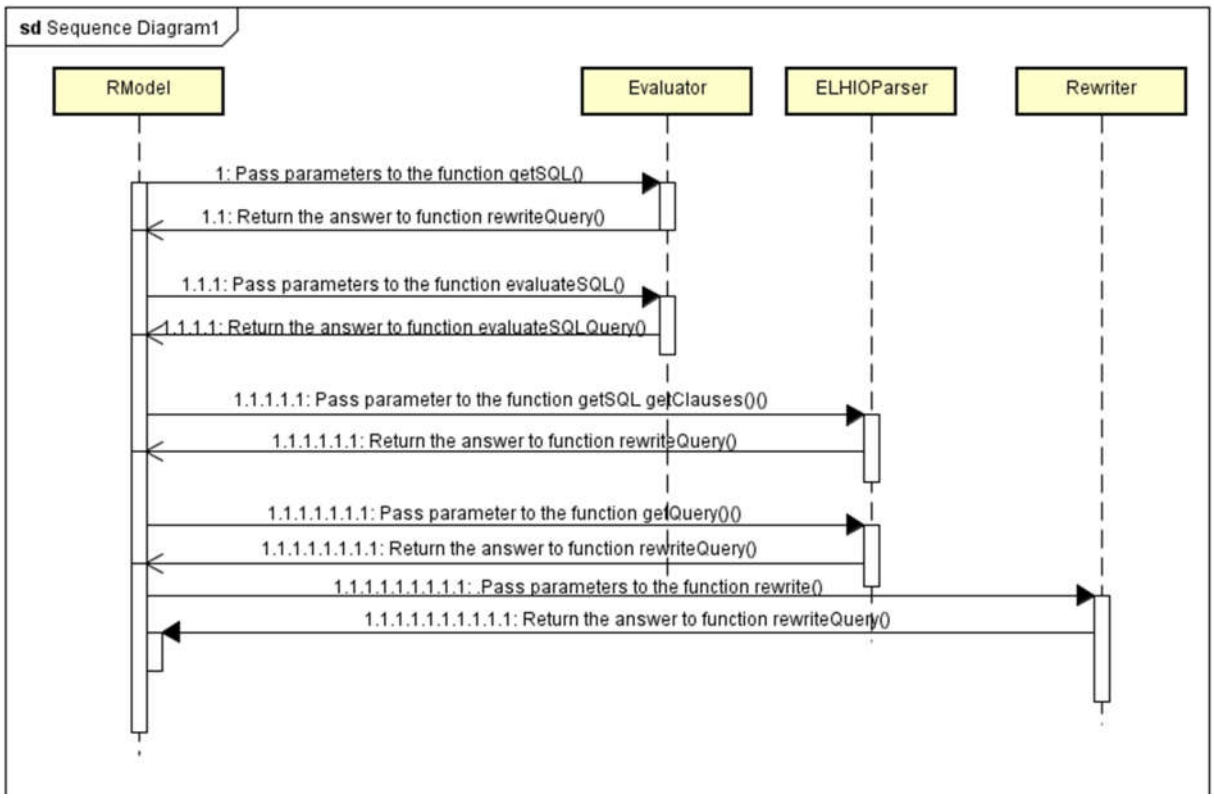


**Figure15:** Diagram sequence for OBDA application.(RModel)

## 9. Example about University diagram class:

This application can be implemented on any of the Ontology we want, where I took an example about the university wish applied about it ontology, where from the classes of this university the person ,student ,professor, university....., where applied about it Logic description, which facilitated the establishment of Diagram Class in order to clarify the relationship between these categories.

### 9.1.logic description for University:

$Person \sqsubseteq \exists affiliatedOf.Organization$

$Organization \sqsubseteq \exists affiliatedOrganizationOf.Organization$

$Organization \sqsubseteq \exists affiliatedOrganizationOf.Organization$

$University \sqsubseteq \exists degreeFrom.Person$

$University \sqsubseteq \exists hasAlumnus.Person$

$University \sqsubseteq \exists doctoratDegreeFrom.Person$

$\exists doctoratDegreeFrom \sqsubseteq \exists degreeFrom$

$ExamRecord \sqsubseteq \exists hasExamRecord.Student$

$University \sqsubseteq \exists hasFaculty.Faculty$

$University \sqsubseteq \exists isPartOfUniversity.Faculty$

$\exists headOf \sqsubseteq \exists workFor$

$University \sqsubseteq \exists masterDegreeFrom.Person$

$\exists masterDegreeFrom \sqsubseteq \exists DegreeFrom$

$Person \sqsubseteq \exists member.Organization$

$Organization \sqsubseteq \exists memberOf.Person$

$Course \sqsubseteq \exists teacherOf.FacultyStaff$

$University \sqsubseteq \exists UndegraduateDegreeFrom.Person$

$UndegraduateDegreeFrom \sqsubseteq \exists DegreeFrom$

$AdministrativeStaff \sqsubseteq Employee$

$AssistantProfessor \sqsubseteq Professor$

$AssiciateProfessor \sqsubseteq Professor$

$BachelorExam \sqsubseteq Exam$

$Career \sqsubseteq Work$

$Chair \sqsubseteq UndegraduateStudent$

$ClericalStaff \sqsubseteq AdministrativeStaff$

$College \sqsubseteq Organization$

$Course \sqsubseteq Work$

$Dean \sqsubseteq Professor \sqcap \exists headOf.College$

$Degree \sqsubseteq Work$

$Director \sqsubseteq Person \sqcap \exists headOf.Program$

$Employee \sqsubseteq Person \sqcap \exists workFor.Organization$

$ExDean \sqsubseteq Professor$

$Exam \sqsubseteq Work$

$ExamRecord \sqsubseteq Work$

$Faculty \sqsubseteq Organization$

$FacultyStaff \sqsubseteq Employee$

$FullProfessor \sqsubseteq Pr$ofessor

$GaduateCourse \sqsubseteq Course$

$GraduateStudent \sqsubseteq Person \sqcap \exists takesCourse.GraduateCourse$

$PostDoc \sqsubseteq FacultyStaff$

$Professor \sqsubseteq FacultyStaff$

$Program \sqsubseteq Organization$

$ResearchAssistant \sqsubseteq Student$

$Student \sqsubseteq Person \sqcap \exists takesCourse.Course$

$SystemStaff \sqsubseteq AdministrativeStaff$

$UndergraduateStudent \sqsubseteq Student$

$University \sqsubseteq Organization$

$VisitingProfessor \sqsubseteq Professor$

## 9.2.Class diagram for university:

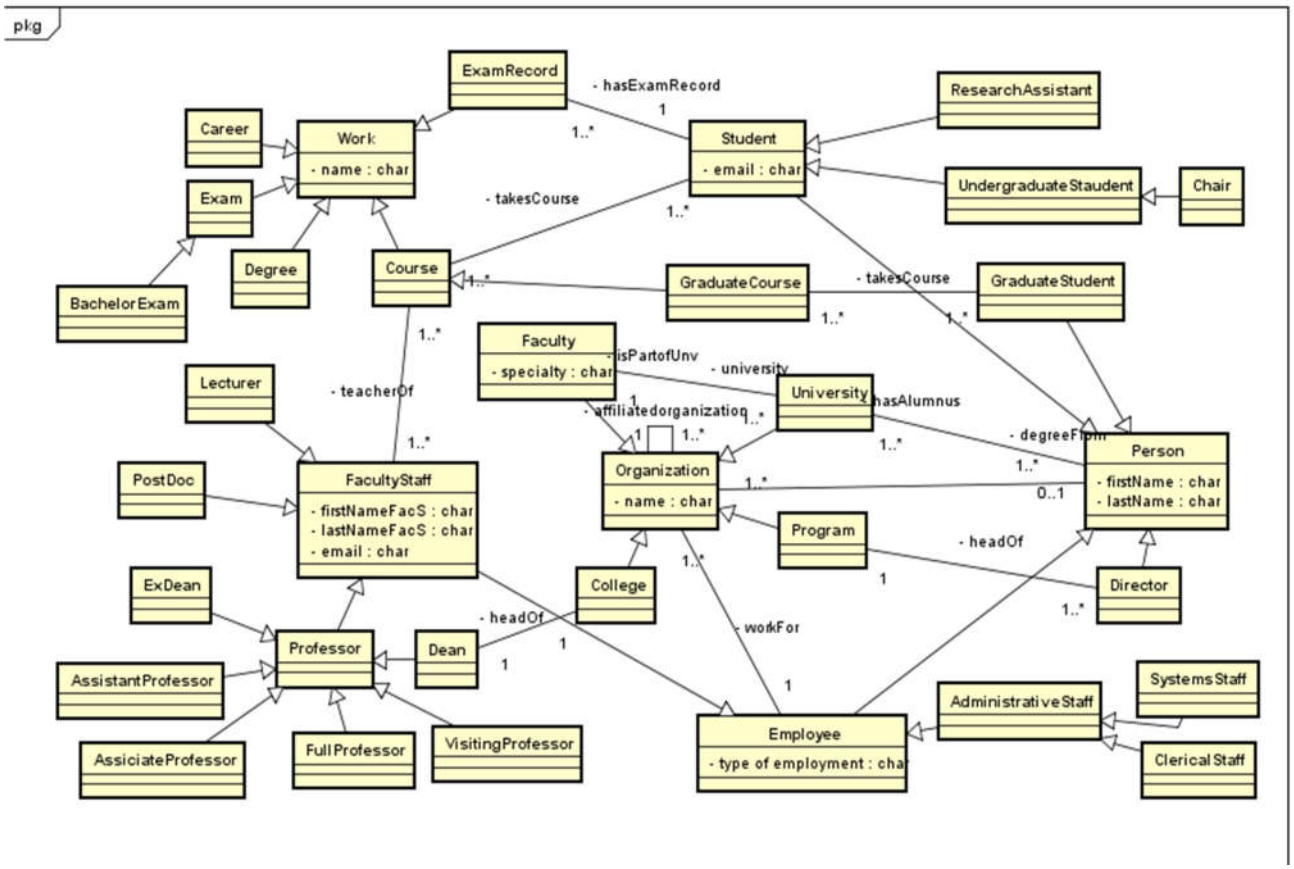


**Figure16:**Class diagram for university applied in the OBDA application.

## 10.Development environment:

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-in system for customizing the environment. Eclipse is

written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins,

including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua,NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematical. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.[39]

## 11.Implementation:

The application procedure has been explained . The developed application was performed in order to expand the process of connecting to the database .it rewrites an ontology query then it convert it into SQL ,the latter is connected to the database to answer the query by pressing au the button answer .

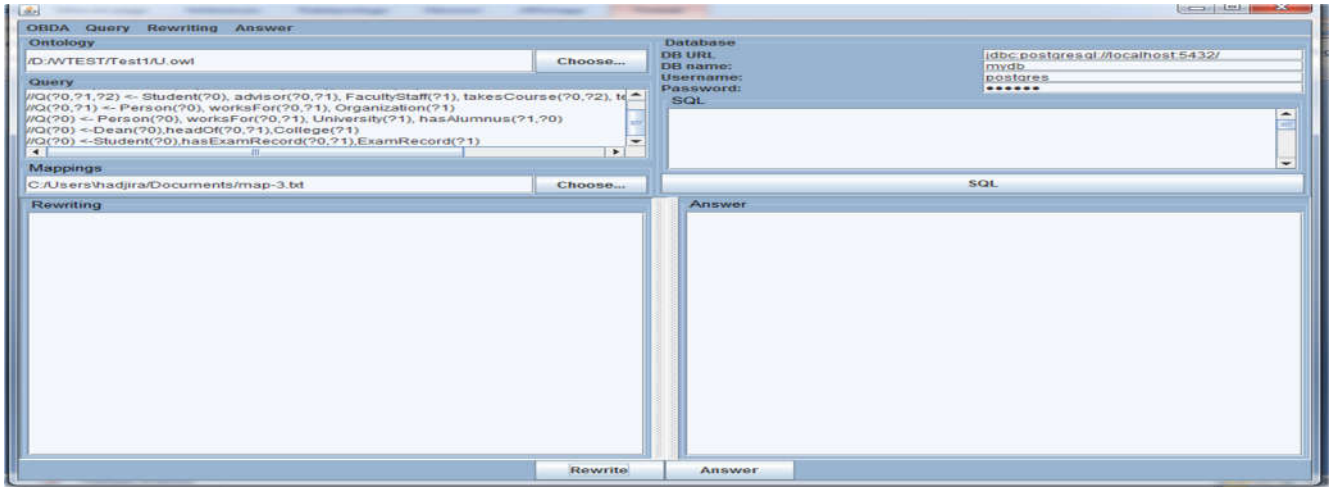


**Figure17**:OBDA application before rewriting and answering query.

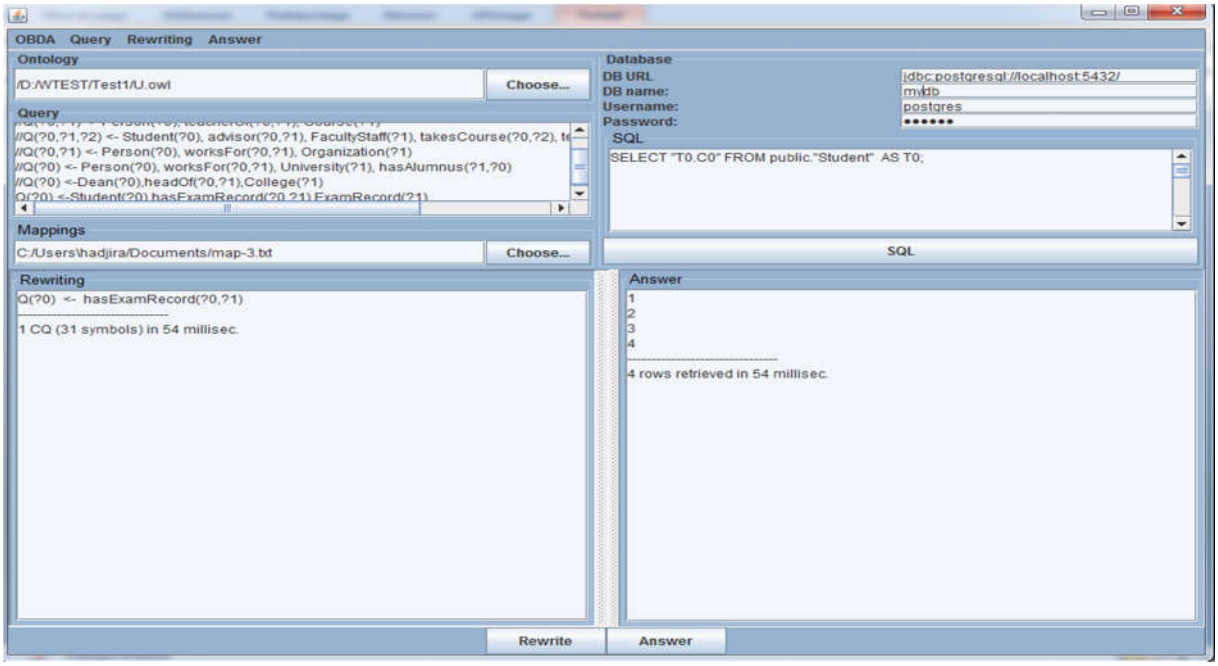


**Figure18:** OBDA application after rewriting and answering query.

## 12.Conclusion:

In this chapter, OBDA is defined .It uses the system of ontology as conceptual Diagram of the user interface of the database system SQL .

We talked about Query rewriting and the Algorithm that has been used to wake the query rewriting process easy as well as the answer query which comes after converting every rewriting query to SQL.

# GENERAL CONCLUSION

The Ontology-Based Data Access (OBDA) paradigm consists in exposing, for the purpose of querying, a conceptual view of the domain of interest, given in the form of an ontology that hides the structure of the data sources. Queries can then be posed over this high-level conceptual view, and end users no longer need an understanding of the data sources, the relation between them, or the encoding of the data. User queries are translated by the OBDA system into queries over one or multiple data sources The framework of OBDA has received a lot of attention in the last years: many theoretical studies have paved the way for the construction of OBDA systems and the development of OBDA projects for enterprise data management in various domains . The formalism of choice for representing ontologies in OBDA is the description logic DL-LiteR , which underpins OWL 2 QL . DL-LiteR was designed to ensure that queries against the ontology are first order rewritable; that is, they can be reformulated as a set of relational queries over the sources.

The purpose of this topic that SQL is giving the data exist only in the database reverse OBDA, which increases the aspects of semantic existing tic. When the query is entered and the ontology ,the OBDA system rewrites this query and ontology into many of the new query. where each query transforms into SQL , that we can extract data indirect relationships.

Improvements that need to be made are:

- Automatically convert from database to ontology.

- Create an automatic mapping that converts each query to SQL.

- Give An automatic procedure that converts a query from a natural language into an language formal where the user only shows the natural language and the result

## Referances:


[1].*Zeeshan Ahmed and Detlef Gerhard," Role of Ontology in Semantic Web Development",* March 2011 , DESIDOC Journal of Library & Information Technology

[2].*G Kück, "Tim Berners-Lee's Semantic Web",March 2004*

[3].*TAPAS KUMAR MISHRA, "SEMANTIC WEB ",*ACM New York, NY, USA ©2003

[4]. https://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

[5].*https://www.obitko.com/tutorials/ontologies-semantic-web/rdf-schema-rdfs.html*

[6].*Mohammad Mustafa Taye, Understanding Semantic Web and Ontologies: Theory and Applications, June 2010*

[7].*Ian Horrocks," Ontologies and the Semantic Web", 2009, Oxford University Computing Laboratory Oxford, UK*

[8].*Elodie Marie Gontier, "Web Semantic and Ontology",2015*

*Mechanical Engineering Information and Virtual Product Development (MIVP), Vienna University of Technology, 1060 Getreidemarkt 9/307 Vienna, Austria*

[9].*Chandrasekaran and John R and Richard Benjamins, "What Are Ontologies, and Why Do We Need Them?", FEBRUARY 1999*

[10*]. Grigoris Antoniou, Enrico Franconi, and Frank van Harmelen, "Introduction to Semantic Web Ontology Languages", Springer-Verlag Berlin Heidelberg 2005*

[11].*"Web Ontology Language OWL"*

[12].*https://fr.wikipedia.org/wiki/Logique_de_description*

[13]. *Anni-Yasmin Turhan,"Description Logic reasoning for Semantic web ontologies",* Copyright 2011 ACM Copyright c 2011 ACM 978-1-4503-0148-0/11/05

[14]. *Franz Baader and Werner Nutt, "Basic Description Logics",* Edited by F. van Harmelen, V. Lifschitz and B. Porter © 2008 Elsevier B.V. All rights reserved

[15].*https://www.researchgate.net/figure/Architecture-of-a-knowledge-representation-system-based-on-Description-Logics_fig1_2885808*

[16]. *Franz Baader and Werner Nutt, "Basic Description Logics"*

[17]. *Anni-Yasmin Turhan,"Description Logic reasoning for Semantic web ontologies"*

[18].*Daniele Nardi, Ronald J. Brachman," An Introduction to Description Logics"*

[19].*Leif Harald Karlsen," Description Logic 1: Syntax and Semantics",2015*

[20].*Sebastian Rudolph, "Foundations of Description Logics",* c *Springer-Verlag Berlin Heidelberg 2011*

[21].*Leif Harald Karlsen, Description Logic 1: Syntax and Semantics,2015*

[22]. *Anni-Yasmin Turhan,"Description Logic reasoning for Semantic web ontologies"*

[23].*http://www.lesliesikos.com/description-logics-and-first-order-logic/*

24].*Alessandro Artale and Diego Calvanese and Roman Kontchakov and Michael Zakharyaschev, "The DL-Lite Family and Relations",2009*

[25].*Ian Horrocks, DAML+OIL: a Description Logic for the Semantic Web*

[26*]. Domenico Lembo , Jose Mora , Riccardo Rosati1 , Domenico Fabio Savo1 and Evgenij Thorstensen, "Mapping Analysis in Ontology-based Data Access: Algorithms and Complexity (Extended Abstract)"*

[27].*Roman Kontchakov , Mariano Rodr´ıguez-Muro and Michael Zakharyaschev, "Ontology-Based Data Access with Databases", Faculty of Computer Science, Free University of Bozen-Bolzano, Italy*

[28].*" web Reasoning and Rule Systems"*

[29].https://link.springer.com/chapter/10.1007/978-3-319-45246-3_38

[30].*Thiago H. D. Araujo , Barbara T. Agena , Kelly R. Braghetto , Renata Wassermann,* "*OntoMongo - Ontology-Based Data Access for NoSQL",* Instituto de Matem´atica e Estat´ıstica – Universidade de S˜ao Paulo

[31].*Domenico Lembo , Jose Mora , Riccardo Rosati , Domenico Fabio Savo , Evgenij Thorstensen, "Mapping Analysis in Ontology-based Data Access: Algorithms and Complexity"*

[32].*https://academic.oup.com/comjnl/articleab stract/60/3/389/2609373?redirectedFrom=fulltex*

[33]. H*´ector P´erez-Urbina, Ian Horrocks, and Boris Motik, Efficient Query Answering for OWL 2*

[34]. *http://www.mathcs.duq.edu/simon/Fall04/notes-7-4/node6.html*

[35].*https://en.wikipedia.org/wiki/Class_diagram*

[36].*https://www.lucidchart.com/pages/uml-use-case-diagram*

[37].*Emina Torlak, UML Sequence Diagrams,2015*

[38].*https://en.wikipedia.org/wiki/Eclipse_(software)*

[39].*http://blog.saltlux.com/info center/%EC%98%A8%ED%86%A8%EB%A1%9C%EC%A7%80%EC%99%80-%EC%B6%94%EB%A1%A0*

ملخص :

يعد نهج الوصول إلى قاعدة البيانات العلائقية المستندة إلى الانتولوجي (OBDA) جزءًا أساسيًا من الويب الدلالي. في هذا الأسلوب ، يحدد المستخدم استعلامًا ، مستخدمًا TBox. تتم كتابة هذا الاستعلام في مجموعة من الاستعلامات. يتم الرد على الاستعلامات التي تمت إعادة كتابتها باستخدام ontology ABox. في حالتنا ، يتم تمثيل ABox بواسطة قاعدة بيانات علائقية وعملية ربط(mapping) بين الأنتولوجي وقاعدة البيانات المقترحة.

ميزة OBDA هو استعلام قاعدة بيانات يجب أن تعود ليس فقط على البيانات التي يتم تخزينها في قاعدة البيانات بشكل واضح، ولكن أيضا الحقائق الإضافية التي يمكن استنتاجها من المعلومات الانتولوجي(ontological information) من قبل عملية اعادة الكتابة الاستعلامات.

هناك العديد من الخوارزميات لحل مشاكل إعادة كتابة الاستعلام في هذا العمل نستخدم الخوارزمية D. Calvanese التي تقوم على (DL). لهذا السبب ، نستكشف عائلات مختلفة من هذه DL. نناقش أيضًا اللغات الفرعية للغة الأنطولوجية على الإنترنت (OWL) وشرح علاقتها بمشكلتنا.

**الكلمات المفتاحية :**

OWL، OBDA، DL ، قواعد البيانات

## Abstract:

Ontology based relational data bases access approach (OBDA) is a crucial main of semantic web. In that approach, the user specifies a query, using the TBox of the ontology. This query is rewritten into a set of queries. The rewritten queries are answered using the ABox of the ontology only. In our case the ABox is represented by a relational data base and a mapping process between the ontology and a data base is proposed.

The advantage of the OBDA is that a database query then should return not only the data that is stored explicitly in the database, but also additional facts that can be inferred from it using the ontological information by rewriting process.

There are many algorithms for rewriting problem, in this work we use the algorithm of D. Calvanese which is based on description logics (DL). For this reason, we explore different family of such DL. We discuss also, sublanguages of ontology web languages (OWL) and explain their relationship with our problem.

**Key words:**

Ontology, OWL, OBDA, DL, data bases.

## Résumé :

L'approche d'accès aux bases de données relationnelles basées sur l'ontologie (OBDA) est un élément essentiel du web sémantique. Dans cette approche, l'utilisateur spécifie une requête, en utilisant le TBox de l'ontologie. Cette requête est réécrite dans un ensemble de requêtes. Les requêtes réécrites sont répondues en utilisant l'ABox de l'ontologie seulement. Dans notre cas, l'ABox est représenté par une base de données relationnelle et un processus de mappage entre l'ontologie et une base de données est proposé.

L'avantage de l'OBDA est qu'une requête de base de données doit retourner non seulement les données qui sont stockées explicitement dans la base de données, mais aussi des faits supplémentaires qui peuvent être déduits à partir de l'information ontologique par le processus de réécriture.

Il existe de nombreux algorithmes pour résoudre les problèmes de réécriture, dans ce travail nous utilisons l'algorithme de D. Calvanese qui est basé sur des logiques de description (DL). Pour cette raison, nous explorons différentes familles de ces DL. Nous discutons aussi, des sous-langages des langages web d'ontologie (OWL) et expliquons leur relation avec notre problème.

**Mots clés:**
Ontologie, OWL, OBDA, DL, bases de données.