USING RELATIONAL MODEL TO STORE OWL ONTOLOGIES AND FACTS

Tarek Bourbia and Mahmoud Boufaida

LIRE Laboratory, University Constantine 2, Algeria
{bourbia_tarek, boufaida_mahmoud}@yahoo.fr

ABSTRACT

The storing and the processing of OWL instances are important subjects in database modeling. Many research works have focused on the way of managing OWL instances efficiently. Some systems store and manage OWL instances using relational models to ensure their persistence. Nevertheless, several approaches keep only RDF triplets as instances in relational tables explicitly, and the manner of structuring instances as graph and keeping links between concepts is not taken into account. In this paper, we propose an architecture that permits relational tables behave as an OWL model by adapting relational tables to OWL instances and an OWL hierarchy structure. Therefore, two kinds of tables are used: facts or instances relational tables. The tables hold instances and the OWL table holds a specification of how the concepts are structured. Instances tables should conform to OWLtable to be valid. A mechanism of construction of OWLtable and instances tables is defined in order to enable and enhance inference and semantic querying of OWL in relational model context.

KEYWORDS

Relational Model, Database, OWL, Instance, Fact, Ontology

1. INTRODUCTION

A database model defines the logical structure of database and determines in which manner data can be stored, organized and manipulated. It is still evolving and new models are being considered, especially in the semantic aspects. Despite this evolution, the relational model is still the most used and none other model even made the end of the dominance of relational databases. Its simplicity and its performances motivate research to design new models on the relational engine by operating, mapping and transformation mechanisms, as is achieved -for instance- by object-relational, XML[2], RDF models [6] [7].

Besides, the field of knowledge representation [20] defines the syntax of ordered symbols that are readable and interpretable by the machine. In the same context the semantic web [9] aims to ensure for computer to analyze data contents and their relationships. It enables structuring and storing the web knowledge. One of the motivations considered by semantic web, through its models, is the semantic search. In this context, ontological languages are proposed to formulate knowledge bases to improve the retrieval in documents repositories.

The field of database model offers the best way to store data efficiently and guarantees its persistence; the knowledge representation languages provide a rich vocabulary and a power of

Dhinaharan Nagamalai et al. (Eds) : CSE, DBDM, CCNET, AIFL, SCOM, CICS, CSIP - 2014
pp. 83–98, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4408

expressiveness. The combination of those two fields allows providing solutions to manage knowledge semantically and efficiently with great performances.

The Web Ontology Language (OWL) [1] supplies for developers more than eXtensible Markup Language (XML), Resource Description Framework (RDF) [6]. With a wide formal vocabulary, OWL [5] is a formalism to build a knowledge base. However, if the quantity of the knowledge is very important and the size of the OWL domain is voluminous, it will be complex to treat them by managing a document in a simple file format. Thus, it would be beneficial to embed these ontologies on a database management system and to manage them as databases. The proposed approach, allows the storage of metadata and ontology instances into a persistent and optimal way, while preserving semantics and constraints at an abstract level.

It is important in engineering to implement database based ontology onto an existing database system. It is also interesting to integrate ontology features with an existing database model. Using an existing infrastructure does not deprive the ontologies-based database to be native.

OWL contains three sublanguages [1]: OWL Lite, characterized by a hierarchy and a simple mechanism of constraints; OWL DL, based on the description logic with offering a high degree of expressiveness and ensures computational completeness and decidability; OWL Full, has a big capacity of expressiveness, without ensuring completeness and decidability. OWL DL language has been chosen here to express the knowledge base. It contains all the structures of OWL with some limitations such as the separation of types (a class cannot be at the same time an individual and a property), an important thing is to not confuse the data and the metadata.

A semantic web application requires storing and manipulating enormous data. Storing an ontology and its instances and processing them by a user application need some mechanisms to cope, on the one hand, with huge numbers and relations between concepts and, on the other hand, with large amounts of instances. Relational database is a good support as repository to ensure the persistence of ontology instances, due to its experience, performance and features. For this reason, it has been chosen in this paper to embed OWL DL knowledge bases.

To take benefit of the structure expressed in OWL and at the same time of the database system of which embeds the ontology instances, this paper provides a solution to explicit a graph structured document as OWL into relational tables with preserving constraints and relationships. It aims at ensuring the persistence of a huge number of OWL concepts and a large amount of instances in a native way without using the classical way of triples embedded into relational table of three columns [17]. The objective of this work is to present a mechanism of how to store OWL concepts and instances in relational tables while preserving the semantic and without losing data. The main advantage of this solution is to have a specification and conception under the OWL-DL language, a logical modelling under relational model, and a physical storage into the relational database. Therewith the distance is close between the conceptual model and the logical model in matters of preserving data, link and constraint. For that purpose, it could be said that this work enters in the field of the Ontology-Based Database.

This paper is organized as follows: The following section presents the works carried out on this subject, section (3) shows the principles of the proposed approach of how storing OWL ontology into relational tables and section (4) details the mechanism of storing OWL ontology concepts and facts, followed by a conclusion and some perspectives.

2. SOME RELATED WORKS

In order to manage ontologies efficiently and to ensure a robust persistence, several knowledge management systems have been proposed. Each work in this context uses its own strategies to embed ontologies in a persistent data model.

Pierra et al. have proposed a database architecture model called "Ontology-Based Database" (OntoDB) [10]. The latter defines separately the implementation of the ontology and those of data. The solution consists of two parts: representing the primitive of the ontology model in a meta-schema and defining, once and for all, the physical storage of ontologies from RDF triplets by using the relational model and the object-relational one. ONTOMS [3] developed by Myung et al. is another OWL database management system architecture, whereby the data is physically stored in classes modeled into relational tables. It also performs complex operation as inverseOf, symmetric, transitive and reasoning for instances. ONTOMS also evaluates the requests expressed by OWL-QL [4]. M. Shoaib has developed ERMOS [11], a solution that provides an efficient transformation of the OWL concepts to relational tables to store ontology and to allow easy and fast knowledge retrieval with semantic query. JENA [18] is a framework for building Semantic Web applications. It allows processing RDF and OWL ontology by providing a container for collections of RDF triples. The implementation of SPARQL [8] is used in JENA to query RDF triples. Although JENA provides interaction with OWL ontology, its OWL instances, as well as for RDF, are based on RDF triples.

The two architectures ONTOMS and ERMOS adopt a representation in OWL, which are richer in concepts and semantics by report of what is chosen by other work presented above. The challenge in this kind of work is to make it possible to store ontologies in a right, coherent, scalable and efficient way in order to retrieve knowledge by semantic inference. In the literature, the near total of work aiming at the management of ontologies as a database use the relational model as logical and physical model to shelter the ontology-based data.

All the precedent systems are founded on an external middleware layer that is added on top of a database management system engine. These systems keep users far from ontology instances and do not provide ability to interact with stored ontology facts directly.

The triplet-based approach has been used by several systems, which aim to ensure persistence of OWL ontologies. It solves, somehow, the issue of the integration of semantics in the field of databases, and improves the way of storing and dealing with big amounts of OWL instances. However, the triplet-based approach presents a number of drawbacks such as:

- Losing the power of expressiveness of OWL ontology.
- Storing all triples extract from OWL ontology in the same table. This make self-join complex and with less performance when retrieving instances
- Scanning all table triples to reach the needed triples.

Several works based on NoSQL [15] models use the structure offered by the XML language to store data and to locate them [16]. XML represents a significant evolution of the concept of database to store large volumes of data or documents. An XML database defines a logical model in a XML document, and stores and retrieves the documents according to that model. The structure in graph of the ontology language, inspired from XML, is far from the tables of the relational model. For this purpose, there exists some works that deal with the mapping or the correspondence of ontology model with XML tree [2]. It seems more relevant than its

correspondence with relational table, by the fact that XML is the first pillar to build assertional languages and ontology. Nevertheless, XML databases are built into relational engines and applying mapping under "mapping" could lead to degrade the performances and cause data loss.

Besides, there exist other works, in the opposite direction, that aim to ensure the extraction of ontology from relational database schema [12]. These works construct a local ontology from data that already exist in a relational database. Their goal is not to solve problems of huge ontologies and instances, but to deal with data stored in relational model semantically.

Summarizing, storing ontology instances using a file system makes querying those instances very difficult and with poor performances. To cope with huge ontology, most of the works in the literature use mapping mechanisms from the ontology into relational or objects databases. And that would not be fluent and work perfectly in terms of performance due to large ontology vocabulary and complexity of graph. The proposed approach in this paper allows storing OWL concepts and instances in relational tables without losing semantic, nor data, and enables users to interact with data directly. Compared with existing methods [13] [14], the proposed approach keeps faithfully a class hierarchy and relationship between concepts. In addition, it provides capabilities to interact directly with instances, unlike other existing methods.

3. OVERVIEW OF THE ARCHITECTURE

The architecture of storing OWL ontology into relational tables resulted after combining some aspects of the field of databases and those of the semantic web and knowledge representation. This architecture is designed to allow both the efficient management of the ontology concepts and instances, and to ensure semantic search and update regardless of the physical data structure.

The essential functions of the proposed system are (see Figure 1):

- Storing and manipulating OWL Ontology in relational model. A meta-concept table named “OWLtable” represents the classes and relationships between classes and the properties ensure predicates between classes and individuals. In function of this table, the OWL ontology will be constructed and represented by a graph or an OWL expression. In fact, the unique OWL ontology domain that exists is that stored in relational table into “OWLtable”. This way allows approaching the semantic conceptual model and the logical model, and avoiding losing concepts that suffer the architectures based on transformation mechanism and mapping solution.

- Storing and manipulating ontology instances and facts. A set of relational tables represents classes and some properties and holds individuals of the OWL ontology as tuples. The meaning of tuples is preserved, and each instance is associated with an ontological concept stored into “OWLtable” and referenced by both table name and attribute. Instances tables and instances have to be conforming to the structure of OWL concepts and match with constraints. In other words, any operation in the set of instances tables should be checked and validated with concepts structure and constraints stored in “OWLtable”.

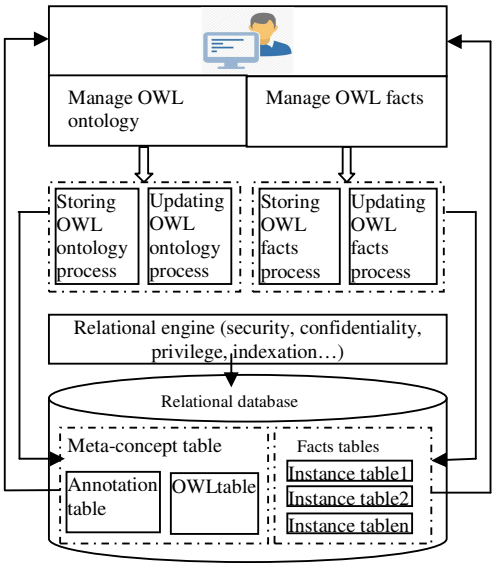

Figure 1:Storing ontology architecture.

The proposed architecture consists of a native OWL database engine built on the system of a relational database and integrated with the relational engine (definition and manipulation language). This approach allows storing, accessing and searching on the OWL using relational tables to ensure the persistence of data linked with OWL concepts and properties.

In fact, neither OWL document, nor OWL instances exist; what exists is their equivalent in relational database that are “OWLtable” and instances tables. The purpose of “OWLtable” is to define concepts of OWL ontology, and so instances tables will be an instance document to describe an OWL individuals and facts that conforms to “OWLtable”.

The “OWLtable” contains kinds of elements like: class, object property, data type property and constraints which give the content nature of instances tables and the hierarchy of element.

This approach allows manipulating OWL ontology in relational tables. Furthermore, accessing, reasoning and querying OWL ontology are performed in persistence layer.

The next section details the part of how using relational tables to store ontology classes, properties and individuals to make them ready for any actions later on.

4. STORING OWL ONTOLOGY INTO RELATIONAL TABLES

The mechanism adopted to reach the objective of managing OWL ontology in relational engine, contains a number of actions, as is shown in Figure 2. Those actions allow having: a meta-concept table named “OWLtable” that acts like OWL ontology; and a set of relational tables resulted from ontological concept which embed the facts and instances of OWL ontology. For this purpose, an ordered steps process is conceived here to create a set of relational tables that represent OWL concepts and facts without losing hierarchy of concept, links, constraints and facts. In other words, this set of ordered steps process ensures the correspondence between the semantic structure of the ontology as a graph with relationships, and the syntactical logical structure in the form of relational tables to ensure persistence of instances with the best performance possible.

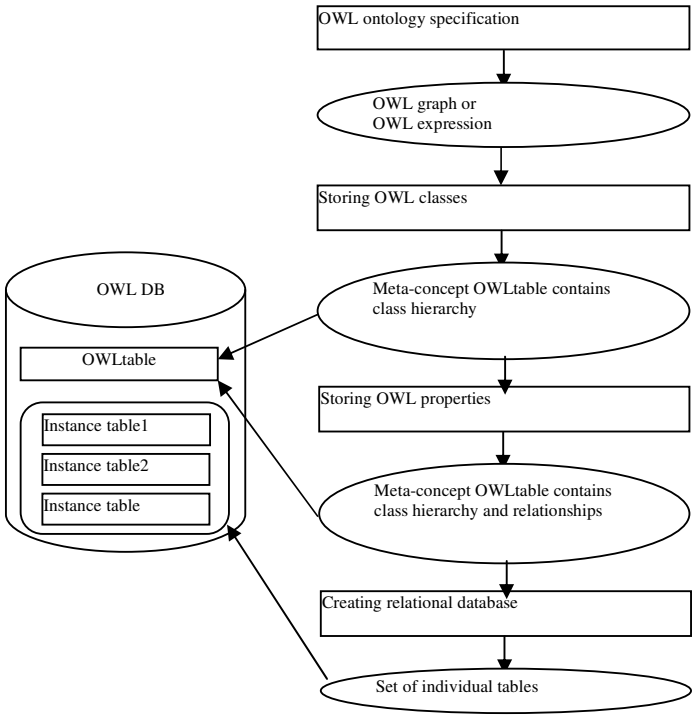

Figure 2:A Mechanism of storing OWL ontology into relational tables.

As shown in Figure (2), the creation of the meta-concept table "OWLtable" is the first step, then filling it with class hierarchy and properties, the next step is to create the database that is a set of instance tables obtained by parsing classes, data type properties and object properties. The set of instances tables hold facts.

In the following subsections, a number of processes are developed with rules to detail how storing classes and properties into a meta-concept table that keep the hierarchy and the relationships between concepts; and how storing facts and instances into relational tables that hold the database of OWL knowledge base. To illustrate that, an example of OWL ontology domain is used.

4.1. Storing OWL Class into Relational Table

The basic concept of ontology is the class. So, the most important step and rule is to perform the correspondent relational table to embed ontology classes. It is around class that any object property or data type property is defined. So the first correspondence applies to the classes to support the other concepts of OWL later on.

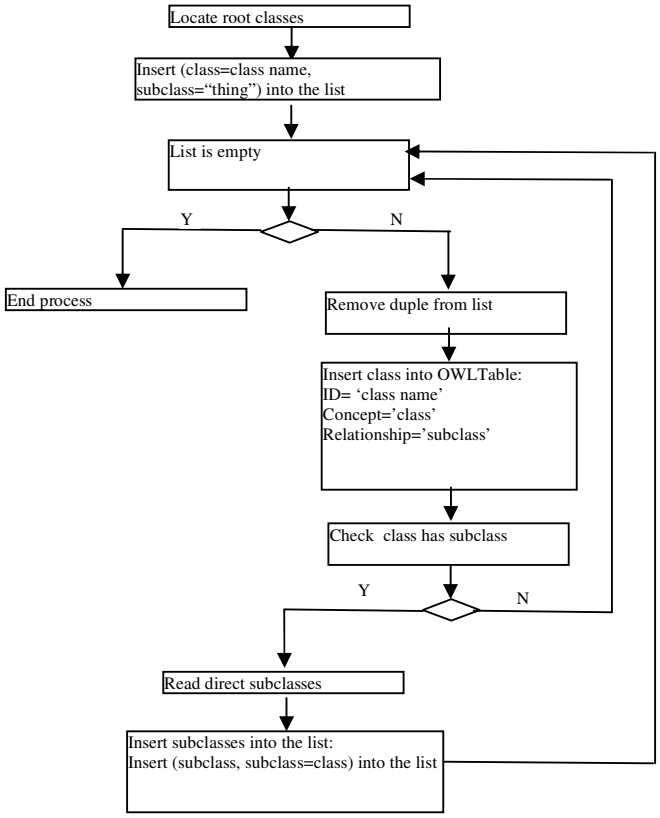

Figure3:The process of storing OWL classes into relational table.

The process of storing OWL classes into relational table that under the title "OWLtable" focuses on locating every OWL class, regardless of its nature and its position in OWL graph and on saving it with some information in order to keep hierarchical structure and type of every relationship with another class, as is shown in Figure (3). This process is based on the following rule:

RULE 1: for each class in OWL ontology, a record in "OWLtable" is inserted. This record defines: that is a class concept and the super class in relationship with this class. The super class of root classes is considered in this process as "thing", the super class of all classes in OWL specification [1].

EXAMPLE: the following classes presented in OWL syntax:

<owl:Class rdf:ID="Human"> …</owl:Class>
<owl:Class rdf:ID="Man"><rdfs:subClassOf rdf:resource="#Human"/></owl:Class>
<owl:Class rdf:ID="Woman"><rdfs:subClassOf rdf:resource="#Human"/></owl:Class>
<owl:Class rdf:ID="Country"> …</owl:Class>
<owl:Class rdf:ID="Town">…</owl:Class>

are inserted in OWLtable as shown in Table(1).

Table 1:OWLtable contains hierarchical of classes.

ID	Concept	Relationship
Human	Class	Thing
Man	Class	Human
Woman	Class	Human
Country	Class	Thing
Town	Class	Thing

At the end of the process of storing OWL classes into relational table, a hierarchy of classes is obtained from records of "OWLtable", and it will be considered as a meta-concept table for knowledge base. The next section is devoted to enrich the meta-concept "OWLtable" by another important ontological concept which is "OWL property".

4.2. Storing OWL Property into Relational Table

OWL property gives a way of describing a kind of relationship of individuals to individuals and of individuals to data values [1]. To complete OWL graph in relational field, on the other hand, to enrich class hierarchy already stored in the meta-concept "OWLtable", linked classes by predicates are treated by the process shown in Figure (4) to have a complete OWL graph in the meta-concept table. The process deals with a number of ontological concepts: object property, data type property, cardinality constraint.

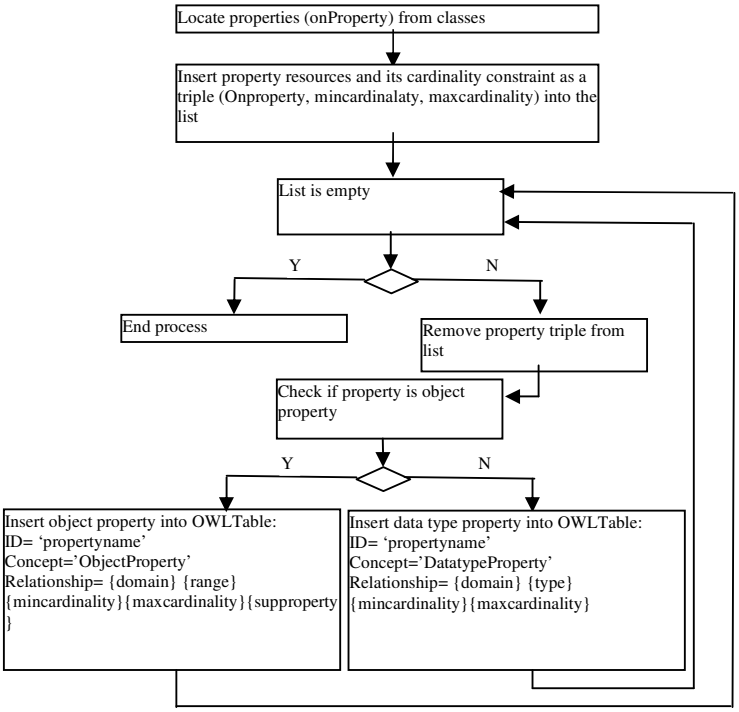

Figure 4:The process of storing OWL property into relational table.

The process of storing OWL predicates into "OWLtable" affects the two types of relationship of individuals: Object Property and DatatypeProperty. A number of restrictions that define property are preserved in "OWLtable", that are: domain, range and cardinalities. In the case of

ObjectProperty the generalization is preserved by sup property in meta-concept table. The property definition is formulated by the expression:

Relationship= {domain} {range} {mincardinality} {maxcardinality} [{supproperty}], as is shown in Figure (4). This process is based on these three rules:

RULE 1: for each onProperty in OWL class, a record in "OWLtable" is inserted. This record defines: that is an onProperty concept; the relationship that details predicates by a set a restriction: domain, range, cardinalities.

RULE 2: the type of onProperty is procured from property statement by checking each ID. There are two types: ObjectProperty and DatatypeProperty.

RULE 3: The super property of ObjectProperty is considered in this process and is added at the end of relationship expression:

Relationship= {domain} {range} {mincardinality} {maxcardinality}{supproperty}.

As an example, consider the following set of statements about properties presented in OWL syntax:

<!--**classes definition**-->

<owl:Classrdf:ID=" Human ">
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource="#HasFriend">
<owl:mincardinality rdf:datatype="&xsd:int">0< /rdfs:mincardinality>
</owl:Restriction>< /rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource="#HasFather">
<owl:cardinality rdf:datatype="&xsd:int">1< /rdfs:cardinality>
</owl:Restriction>< /rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource="#HasSpouse">
<owl:maxcardinality rdf:datatype="&xsd:int">1< /rdfs:maxcardinality>
</owl:Restriction>< /rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource="#name">
<owl:cardinality rdf:datatype="&xsd:int">1< /rdfs:cardinality>
</owl:Restriction>< /rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource="#Mail">
<owl:mincardinality rdf:datatype="&xsd:int">0< /rdfs:mincardinality>
</owl:Restriction>< /rdfs:subClassOf></owl:Class>……

<!--**object properties definition**-->

<owl:ObjectProperty rdf:ID=" HasFriend ">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/></owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="HasFather">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Man"/></owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="HasSpouse">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/></owl:ObjectProperty>……

<!--**data type properties definition**-->

<owl:DatatypeProperty rdf:ID="name">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="&xsd:string"/></owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="mail">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="&xsd:string"/></owl:DatatypeProperty>…….

Those properties are inserted in OWLtable as shown in Table(2).

Table 2:OWLtable contains hierarchical of classes.

ID	Concept	Relationship
Human	Class	Thing
Man	Class	Human
Woman	Class	Human
Country	Class	Thing
Town	Class	Thing
HasFriend	ObjectProperty	{Human}{Human}{0}{unlimited} {does not exist}
HasFather	ObjectProperty	{Human}{Man}{1}{1} {does not exist}
HasSpouse	ObjectProperty	{Human}{Human}{0}{1} {does not exist}
Name	DatatypeProperty	{Human}{String}{1}{1}
Mail	DatatypeProperty	{Human}{String}{0} {unlimited}

At the end of the process of storing OWL property into meta-concept “OWLtable”, a complete OWL graph is embedded in relational database with its class hierarchy and all predicates. The next section presents the way of storing facts and defines instances relational tables that are conform to “OWLtable”.

4.3. Storing OWL Facts into Relational Tables

OWL data type properties give “facts” that link individuals to data values [1]. Facts typically are statements indicating class membership of individuals and property values of individuals. In relational model [19], a table is a set of tuple (individual) that have the same attributes (table membership). A tuple usually represents an object and information about that object. All the data referenced by an attribute are in the same domain and conform to the same constraints. So by analogy, for each class that has individuals and property values of individuals, a relational table, given the same name of the class with attributes which are no longer than OWL data type properties, is the suitable container to ensure persistence of OWL instances in relational models, as is shown in Figure (5). Besides, to keep relationships and predicates between classes a relationship table, given the same name of the property and foreign keys attribute will be necessary to ensure the relationship between relational tables obtained from parsing OWL data type properties, as is shown in Figure (6).

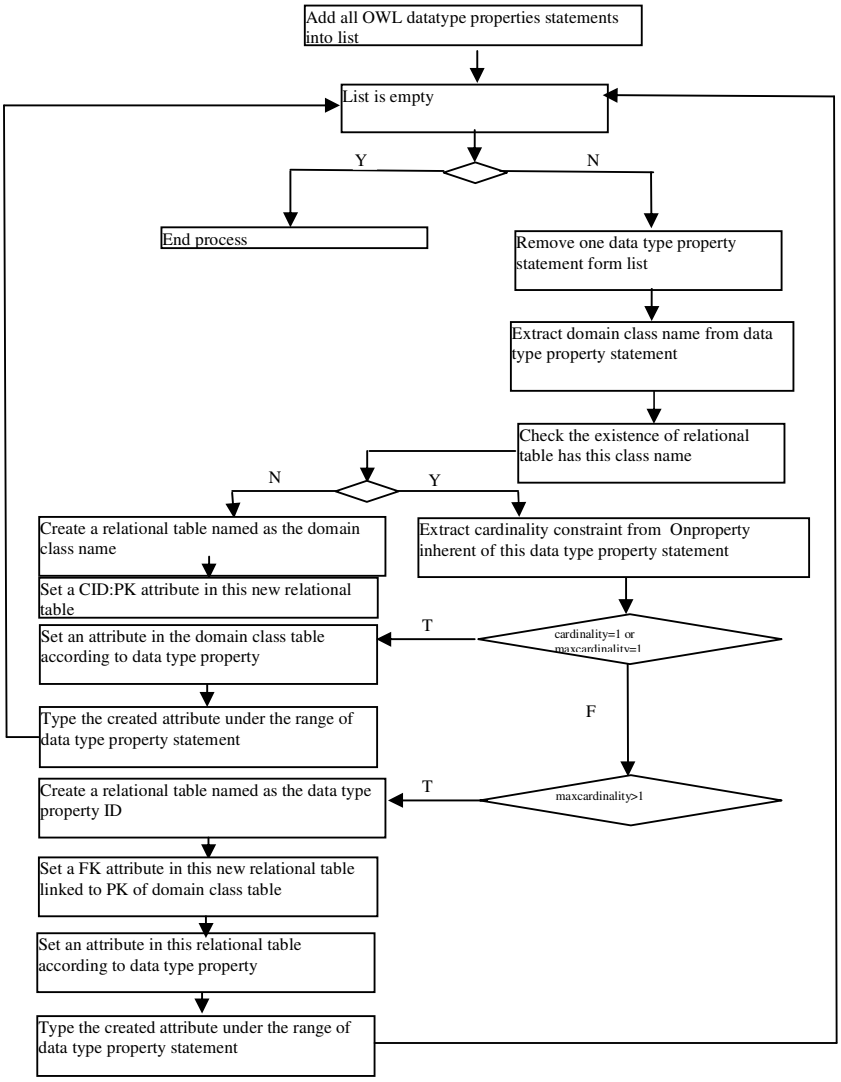

Figure 5:Process of creating relational tables to embed data type properties.

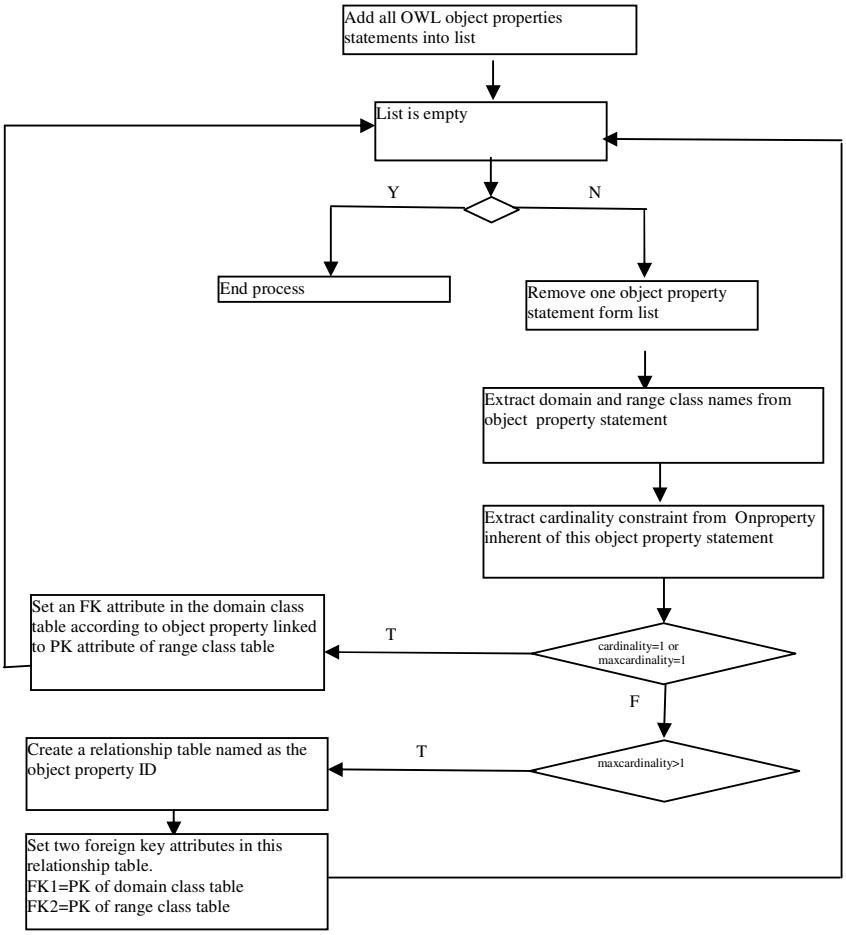

Figure 6:Process of creating relational tables to embed object properties.

After building the meta-concept table "OWLtable" that describe the ontology, the process of storing facts into relational table is founded on how OWL data type properties are grouped then embedded in relational table as attributes (Figure 5); and how OWL object properties are embedded in relational table as attributes with foreign key feature (Figure 6). The outcome relational tables are the repository of each part of fact. The semantic of each part is preserved in "OWLtable" and its correspondent attribute in tables. The mechanism, proposed here, to guarantee to have the appropriate environment to store and preserve OWL instances and facts in relational domain is based on six rules:

RULE 1: for each data type property statement, a relational table is created according to the domain class name if it has not been created by another data type property.

RULE 2: OWL classes must be unambiguously identifiable when using relational tables to store their facts. A class identifier CID for each OWL class is added as a primary key in its correspondent relational table.

The relational table obtained is shown in Table3:

Table 3:Domain class table.

Humain (table name)	
Attribute	Data type
CID	PK
.....	.....

RULE 3: If a cardinality constraint restricts an instance of a class to have at most one semantically value for a data type property, an atomic elementary attribute is set in the relational table that corresponds a domain class of this data type property statement.

EXAMPLE:

<owl:Classrdf:ID="Human">
<rdfs:subClassOf><owl:Restriction>
<owl:OnProperty rdf:resource="#name"/>
<owl:cardinality rdf:datatype="&xsd:int">1<rdfs:cardinality />
<owl:Restriction/><rdfs:subClassOf />……..
<owl:DatatypeProperty rdf:ID="name">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="&xsd:string"/></owl:DatatypeProperty>…..

The obtained relational table is shown in Table (4).

Table 4:Domain class table.

Human (table name)	
Attribute	Data type
CID	PK
Name	String
.....	.....

RULE 4: If a cardinality constraint restricts an instance of a class to have semantically more than one value for a data type property, a relational table is created and named as the data type property ID. A Foreign Key attribute is added in this new relational table linked to Primary Key (CID) of domain class table.

EXAMPLE:

<owl:Classrdf:ID="Human">
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource=”#Mail”>
<owl:mincardinalityr df:datatype="&xsd:int">0< /rdfs:mincardinality>
</owl:Restriction>< /rdfs:subClassOf></owl:Class>……
<owl:DatatypeProperty rdf:ID="mail">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="&xsd:string"/></owl:DatatypeProperty>…..

The relational tables obtained are shown in Figure (7).

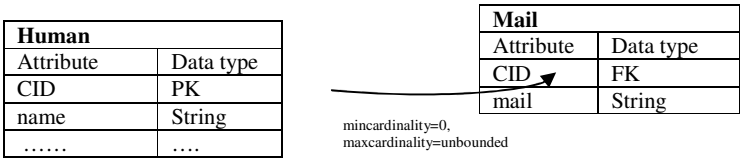

Figure 7:Multi value data type property

RULE 5: If a cardinality constraint restricts individuals of a class to have semantically more than individual for an object property, a relationship table named as the object property ID, is created. Two foreign key attributes are added in this relationship table to link individuals of domain class table to ones of range class table.

EXAMPLE:

<owl:Classrdf:ID="Human">
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource=”#HasFriend”>
<owl:mincardinalityrdf:datatype="&xsd:int">0< /rdfs:mincardinality>
</owl:Restriction>< /rdfs:subClassOf></owl:Class>……
<owl:ObjectProperty rdf:ID=" HasFriend ">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/></owl:ObjectProperty>
The obtained relational tables are shown in Figure (8).

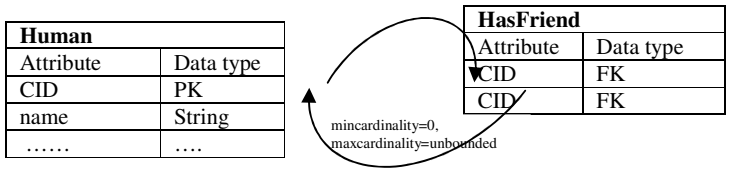

Figure 8:Relationship table

RULE 6: If a cardinality constraint restricts the individuals of a class to have at most one semantically individual for an object property, a foreign key attribute is added in the domain class table according to object property linked to the primary key attribute of range class table.

EXAMPLE:

<owl:Classrdf:ID="Human">
<rdfs:subClassOf><owl:Restriction><owl:onPropertyrdf:resource=”#live”>
<owl:cardinality rdf:datatype="&xsd:int">1< /rdfs:cardinality>
</owl:Restriction>< /rdfs:subClassOf></owl:Class>……
<owl:ObjectProperty rdf:ID="live">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Town"/></owl:ObjectProperty>

The obtained relational tables are shown in Figure (9).

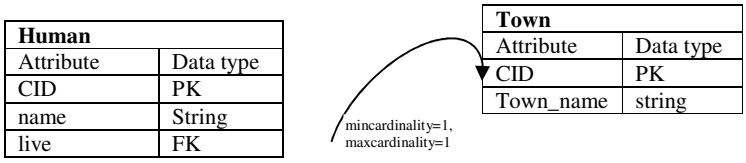

Figure 9:Object property foreign key

5. CONCLUSIONS

In this paper, we have presented an approach that aims at storing a huge OWL ontology and big amounts of instances in the relational engine efficiently without losing concepts neither instances. This approach allows the use of the expressiveness power of the knowledge representation language "OWL" as well as the facilities, and the efficiency that offer relational model. We have defined the meta-concept table "OWLtable" that embeds faithfully class hierarchy and OWL concepts with their relationships and constraints. A set of relational tables are created according to class and property features in order to store OWL instances. In fact, only "OWLtable" and a set of instances table are effective to represent natively the whole ontology. That is why users are able to access, manage and manipulate instances directly. More complex OWL concepts such as subproperty, value constraints and other class and property description [1] could be added into meta-concept table to enrich the OWL specification in relational model.

In order to enable querying ontology instances that are stored in a relational database, our future work will be focused on the refinement of the proposed architecture to integrate the mechanism of searching both OWL instances and concepts with the mechanism of updating. The principle of these mechanisms is based on parsing the meta-concept table "OWLtable" before any query run on OWL instances tables. This leads to check on the one hand, the well-formed and the validity of OWL instances tables according to "OWLtable", and on the other hand, to check constraints and locate properties that give a semantic dimension to the query.

REFERENCES

[1] S.Bechhofer, F.V.Harmelen, J.Hendler, I.Horrocks, D.L.McGuinness, P.F.Patel-Schneider & L.A.Stein, (2004) "OWL Web Ontology Language Reference, W3C Recommendation", http://www.w3.org/TR/owl-ref.

[2] T.Bray, J.Paoli, C.M.Sperberg-McQueen, E.Maler & F.Yergeau, (2008) "Extensible Markup Language (XML) Version 1.0. (fifth edition), W3C Recommendation", http://www.w3.org/TR/REC-xml.

[3] P.Myung, L.Ji-Hyun, L.Chun-Hee, L.Jiexi, O.Serres & C.Chin-Wan, (2007)"ONTOMS : An Efficient and Scalable Ontology Management System", Springer Advances in Databases: Concepts, Systems and Applications, Vol. 4443/2007, pp 975-980.

[4] R.Fikes, P.Hayes & I.Horrocks, (2004) "OWL-QL-A Language for Deductive Query Answering on the Semantic Web", Journal of Web Semantics 2(1), pp 19–29.

[5] F.Gandon, C.F.Zucker & O.Corby, (2012) Le web sémantique: Comment lier les données et les schémas sur le web?, Dunod, France, pp 83-131.

[6] G.Klyne, J.J.C aroll & B.McBride, (2004) "RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation", http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[7] D.Brickley & B.McBride, (2004) "RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation", http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[8] E.Prud'hommeaux & A.Seaborne, (2008) "SPARQL Query Language for RDF, W3C Recommendation", http://www.w3.org/TR/rdf-sparql-query.

[9] T.Berners-Lee, J.Hendler & O.Lassila, (2001) "The Semantic Web", Scientific American, 284 (5), pp 34-44.

[10] H.Dehainsala, G.Pierra & L.Bellatreche, (2007) "Ontodb: An ontology-based database for data intensive applications". InProceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07), Lecture Notes in Computer ScienceSpringer, pp 497-508.
[11] M.Shoaib & A.Basharat, (2010) "ERMOS: An Efficient Relational Mapping for Ontology Storage". 2010 IEEE International Conference on Advanced Management Science (ICAMS 2010), Chengdu, China, pp 399 – 403.
[12] H. A.Santoso, S.C.Haw and Z.T.Abdul-Mehdi, (2011) "Ontology extraction from relational database: Concept hierarchy as background knowledge". Knowledge-Based System 24 Elseiver , pp 457-464.
[13] E.Vysniauskas & L.Nemuraite, (2006) "Transforming Ontology Representation from OWL to Relational Database". Information Technology And Control, Kaunas, Technologija, Vol. 35A, No. 3, pp 333 - 343.
[14] E.Vyšniauskas, L.Nemuraitė & A.Šukys, (2010) "A hybrid approach for relating OWL 2 ontologies and relationaldatabases". In: P. Forbrig, H. Gunther (Eds.): Perspektives in Business Informatics Research. Proceedings of the 9th international conference, BIR 2010, Rostock, Germany, September 29 - October 1. Berlin-Heidelberg-New York, Springer, 2010, pp 86–101.
[15] J.Han, E.Haihong, G.Le & J.Du, (2011) "Survey on nosql database," in Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on. IEEE, pp 363–366.
[16] T.Bourbia & M.Boufaida, (2013) "Extension of Databases by semantics: XML Schema for embedding OWL-DL Ontology", KMIKS' 13 International Conference on Knowledge Management, Information and Knowledge Systems, 18-20 Avril 2013, Hammamet, Tunisia, pp 219-232.
[17] K.Wilkinson, C.Sayers, H.Kuno & D.Reynolds, (2003) "Efficient rdf storage and retrieval in jena2". HP Laboratories Technical Report HPL-2003-266, pp 131–150.
[18] J.J.Carroll, I.Dickinson, C.Dollin, D.Reynolds, A.Seaborne & K.Wilkinson, (2004) "Jena: implementing the semantic web recommendations". In WWW Alt. '04: Proceedings of the 3th international World Wide Web conference on Alternate track papers & posters, NY, USA. ACM Press, pp 74–83.
[19] E.F.Codd, (1970) "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6), pp 377–387.
[20] F. S.John, (2000) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, USA.

Authors

Tarek Bourbia is affiliated to LIRE laboratory of the university Constantine 2 in Algeria. He received his magister degree in 2009 in database and web semantic domain. He is preparing his PhD thesis in the field of ontological database

Mahmoud Boufaïda is a full professor in the Computer Science department of the university Constantine 2 in Algeria. He heads the research group 'Information Systems and Knowledge Bases'. He has published several papers in international conferences and journals. He has managed and initiated multiple national and international level projects including interoperability of information systems and integration of applications in organizations. He has been program committee member of several conferences. His research interests include cooperative information systems, web databases and software engineering