I.J. Modern Education and Computer Science, **2018, 7, 13-28**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.07.02

Proposal for a Mutual Conversion Relational Database-Ontology Approach

Leila Zemmouchi-Ghomari
Ecole Nationale Supérieure de Technologie, ENST, Algiers, Algeria
Email: leila.ghomari@enst.dz

Abdelaali Djouambi and Cherifa Chabane
Université des Sciences et des Technologies Houari Boumediene, USTHB, Algiers, Algeria
Email: {abdelalidjouambi, cherifa.chabane25}@gmail.com

Received: 11 May 2018; Accepted: 04 June 2018; Published: 08 July 2018

***Abstract*—**Whereas ontologies are formal knowledge representations, conveying a shared understanding of a given domain, databases are a mature technology that describes specifications for the storage, retrieval, organization, and processing of data in information systems to ensure data integrity. Ontologies offer the functionality of conceptual modeling while complying with the web constraints regarding publication, querying and annotation, as well as the capacity of formality and reasoning to enable data consistency and checking. Ontologies converted to databases could exploit the maturity of database technologies, and databases converted to ontologies could utilize ontology technologies to be more used in the context of the semantic web. This work aims to propose a generic approach that enables converting a relational database into an ontology and vice versa. A tool based on this approach has been implemented as a proof of a concept.

***Index Terms*—**Relational database, ontology, conversion approach, conversion tool.

I. Introduction

Relational databases (RDB) remain the central support of data around the world, the most successful architecture for storing and manipulating data in the workplace. On the other side, ontology is a technology that offers the possibility to represent, share and reuse knowledge formally and hence enable reasoning capabilities and enhance querying possibilities.

Despite the differences that separate ontologies from databases, several studies have attempted to exploit the benefits of each in favor of the other [1]. Ontologies can be used in the context of databases in several cases, such as integration of heterogeneous data storage systems, and database design as ontology is a consensual and affluent representation of knowledge related to a specific domain. On the other side, databases can be useful in the semantic web context, for example, in the case of the information extraction from existing corporate knowledge and online data embedded in web pages to be used in semantic web applications, so new knowledge can be inferred and querying richer representations will be possible.

Many research works have been dedicated to proposing approaches and tools to transform databases into ontologies and vice versa, however, to the best of our knowledge no work has proposed a two-way conversion approach. Besides, we address conversion of all main OWL constructs to enable considering the most critical elements of any ontology. The proposed approach is supported by an implementation that facilitates its experimentation and use.

This research work focuses on providing a consistent set of mapping rules describing two-way conversion (RDB-Ontology and Ontology-RDB). Then, to implement a mutual conversion tool based on the proposed approach as a proof of a concept.

This paper is organized as follows: section 2 is a background section in which we present brief definitions of the two artifacts then we highlight the main differences between relational databases and ontologies, we end this section by describing the conversion approaches categories. Section 3 describes the related works or the existing conversion approaches as well as a comparison between related works and the current work. Section 4 presents the motivation of this work in which we explain the reasons to convert one artifact into another. Section 5 explains in detail the two-way generic conversion approach. Section 6 describes the implementation of the conversion tool and tests by converting a sample database and an online ontology to illustrate its efficiency. Section 7 concludes the work by highlighting its contributions and discussing its limitations and perspectives.

II. Background

A. Relational Database Description

The Database concept emerged in the 1960s to meet the management and sharing needs of businesses facing the growing volume of information. A database can be defined as a structured set of data stored on computer-accessible media, representing real-world information

that can be queried and updated by a community of users. Access to and management of a database is provided by a set of programs called the database management system. As part of our work, we are interested in relational databases, given their popularity and extensive use [2, 3]. The concept of the relational database was invented by E. F Codd at IBM in 1970. A relational database is a collection of data organized in the form of tables formally defined by the relational algebra on which they are based. The relational model owes its success in the world of computing to a set of characteristics, such as ease of use, availability of languages for the definition, manipulation, and control of data, such as SQL (Structured Query Language) and the independence of the logical and physical levels.

B. Ontology Description

To facilitate the sharing and reuse of knowledge formally represented in artificial intelligence systems, it is beneficial to define a standard vocabulary in which shared knowledge is represented. The specification of this vocabulary is commonly called an ontology. As a result, ontologies define structured vocabularies, grouping relevant concepts of a domain and their relationships, which serve to organize and exchange information in an unambiguous way. Thus, the knowledge used in the semantic web is based on ontologies to be shared and equipped with operational interpretations. Gruber's definition of ontology is undoubtedly the most related in the literature, which states that ontology is a "formal specification of a shared conceptualization" [4]. Conceptualization refers to an abstract model specific to any phenomenon, which identifies the concepts related to that phenomenon. Explanation means that the concepts used and the constraints of their use are explicitly defined. Its formal character refers to the interpretability of ontologies by machines. Ontology must also be shared and should reflect a community consensus on the represented knowledge.

C. Database Versus Ontology

Like relational databases, ontologies conceptualize a set of entities using classes associated with properties and hierarchies using subsumption relationships. Although used terminologies to name their respective elements are different, the basic principles of modeling are quite similar. Actually, there are several differences between the two structures, which we will summarize in the following table (Table 1).

Table 1. Summary of the differences between relational databases and ontologies

Criterion	Database	Ontology
Purpose	Storage and manipulation of raw data set without semantics.	Representing and sharing knowledge on a domain by explaining what is implicit in the universe of discourse, thus ontology provides the semantics of the domain.
Consensus-building	Built by information gathering techniques: documents, interviews, observation.	Being by definition consensual, the concepts represented are usually the result of a consensus of a community of experts and users.
Concepts' definition rules	Normalization rules require unique identification of database concepts. This constraint is specific to the information system for which the database was developed. Thus, the same concept can be defined in different ways according to the contexts in which they evolve, which generates heterogeneity despite identical semantics.	The non-unique name assumption property specifies that two classes can have different names but their definitions correspond to the same concept. This is considered as flexibility in the definition of concepts due to the open and dynamic environment in which ontologies evolve. The mapping between the different concepts is carried out using artifacts offered by knowledge representation languages, such as owl: equivalentClass and owl: sameAs.
Closed vs. Open world assumption	In databases context, the assumption of the Closed World Assumption prevails that: what is not known to be true is necessarily false. Databases are systems that contain complete information about their field of application; Closed World Assumption can provide either positive or negative response to user requests.	In the semantic web, it is assumed that what is not known as true is merely unknown. This theory is called Open World Assumption and applies when a system has incomplete information. For example, consider a patient's clinical history system. If the patient's clinical history does not include a particular allergy, it would be incorrect to say that the patient does not have this allergy. It is not known if the patient suffers from this allergy unless other information is provided to refute this hypothesis.
Dependence or independence	Presented as a solution to a specific problem, databases depend on the context of the problem.	Generally independent of a specific application or problem (except in the case of application ontologies)

D. Conversion Approaches

Conversion approaches can be classified into three categories:

1) **Logic model approaches**: Existing approaches and methods in this category are based on translating the physical implementation of the database or ontology. Information is retrieved for each entity belonging to the RDB or ontology to be converted, to create the corresponding RDB/ontology. Tools such as D2R MAP [5], VisAVis [6], Relational.OWL [7] have been designed to translate the database logic schema. Some of these tools, such as D2R MAP require a manual definition of mapping rules, others, such as Relational. OWL automatically maps the RDB in ontology according to predefined rules.
2) **Conceptual model approaches**: For this conversion approach, some work has been proposed to move from a conceptual model of a relational database to an ontology. For approaches such as the one presented in [8], the primary motivation to exploit the conceptual model is that the latter (expressed as an ER Relation Entity Diagram) is richer in semantics compared to the relational schema. For the opposite direction of conversion, the primary motivation is to take advantage of the consensual knowledge of domain ontologies and use it for the modeling and design of relational databases. The work done by [9], describes a set of transformation rules that allows translation of the OWL script into UML entities.
3) **Intermediate conceptual model approaches**: The intermediate model is a graph inspired by graph theory [10], it is generated from the entities of an RDB or ontology, by a process that translates each entity into a graph element. The motivation to use this intermediate model is twofold: the first being to make a conversion independent of the evolution of the structure to be converted because RDB and ontologies are often subject to updates. The second point and perform an independent conversion of the physical implementation of the data structure, so the resulting RDB or ontology will be independent of the management system of the structure converts.

We can notice that approaches based on the conceptual model or the intermediate model, do not favor the conversion of the instances contrary to the approaches based on the logical model. Logical model approaches are based on the physical implementation of the structure being converted, and can thus have access to the instances of the relational database or the ontology to be converted.

III. Related Works

Several works dealing with the conversion between relational database and ontology have been proposed, but not all approaches have been implemented as conversion tools [11]. Most of the proposed work describes only the conversion approach and mapping rules. In this section, we focus on approaches wholly or partially implemented. Existing tools (Table 2) include the following:

D2R MAP [5]: mapping rules are described using a declarative language based on XML. This language supported by a tool has been proposed to enrich an already existing ontology from the source database by mapping its contents to this ontology. The mapping process has four phases: For each class, a recordset is selected from the database. Second, the record set is grouped according to the groupBy columns of the specific ClassMap. Next, the class instances are created and assigned a URI or a blank node identifier. Finally, the instance properties are created using datatype and object property bridges. The main feature of this language is that it allows flexible mapping of complex relational structures by using SQL statements directly in mapping rules. In this language, the information contained in the rows of the database is not massively extracted from the database because it is assumed that we are not interested in the contents of the rows of the different tables in ontology but only in the critical information that makes the subject of a query. A D2R processor prototype is publicly available under GNU LGPL license. The processor is implemented in Java and is based on the Jena API.

Ontology Modeler/Document & Resource Manager [12]: it is a mapping system containing three parts: Ontology Modeler, Document Manager, and Ontology Resource Manager. The ontology Modeler creates an ontology model from the OWL document. All constraints will be identified and recorded. An appropriate OWL reasoner is selected depending on the OWL version of the input. Document Manager based on Jena is dedicated to manipulating OWL documents. It builds the union of the imported documents and creates upon them a new ontology model. Ontology Reasoner provides methods for listing, getting and setting the RDF types of a resource.

Relational.OWL [7]: is a fully automatic tool that allows having an ontological representation of a relational database schema. It is characterized by the use of OWL-Full meta-modeling capabilities, which limits the decidability of the resulting ontology. Relational. OWL performs a massive and automatic data migration, which means that the information contained in the rows of the various tables in the database is all mapped to instances in the ontology. It is also characterized by the fact that it maps key attributes (primary and foreign) in data properties specific to the classes corresponding to their tables.

KAON2 [13]: is a platform equipped with a correspondence between databases and ontologies such as R2O. From this perspective, it is not a matter of automating the construction of the classes and properties. The objective is to provide declarative means to describe instantiation processes based on relational bases of predefined ontologies manually. KAON2 makes it possible to provide a "view" under ontology form (what its designers call it a "virtual ontology"), fueled on-the-fly by "virtual ontology" and instances extracted from a database.

Vis-à-vis [6]: is a Protégé plug-in that allows mapping relational databases to existing Protégé ontologies. Mapping is done manually by selecting the dataset for an ontology class from the database. An SQL query will be executed and returns the desired dataset, adding it as new properties to the class. This tool also performs a set of consistency checks to validate the mappings.

DBOM [14]: is a Protégé plug-in enabling end-users to design and instantiate an integration of multiple existing relational databases into an OWL knowledge. The DBOM system includes a migration system composed of a set of formulas linking a set of source schemas of DBs and the target ontology schema formalized in OWL DL.

OWL2DB based approach [15]: the algorithm is based on OWL2DB approach, implemented as a protégé plugin. Mapping rules can be summarized as follows: classes are mapped to tables, properties to relations and attributes and constraints are stored in the metadata table.

DataMaster [16]: is a Protégé plug-in that imports both schema and data from relational databases into Protégé. The particularity of this tool is that supports both OWL and frames-based ontologies. The user can select the tables to be imported into Protégé and can get a preview of the selected tables. The superclass selector enables users to select superclass(es) for the imported table classes, and all the imported classes will be created as subclasses of that class.

DB2OWL [17]: is a prototype that allows creating an ontology from an RDB, programmed in Java and based on the Jena API for the construction of the ontology corresponding to the source database, the mapping performed by DB2OWL is fully automatic just like the export of database instances. A mapping process identifies templates for conceptual elements (based on R2O document) in the database and therefore converts database elements to the corresponding ontology components.

OntER [18]: is a Java plug-in for constructing a conceptual model of an RDB from an ontology through an intermediate model. OntER relies on Protege 3.3, which is an authoring system for ontology creation, for manual ontology construction, and Sybase Power Designer 12.0, which is a design tool, for constructing the conceptual data model of the RDB corresponding to the source ontology.

RDB2ONTO [19]: is an approach based on mapping SQL query to RDF/OWL XML template. Hence, OWL data are stored in an ontology model. The SQL query is executed, and for each row of the query results, it fills in the XML-based OWL 2 template.

OBDA [20]: is a protégé plugin for ontology editing, data mapping and querying enabling users to query the database through the mediating ontology. This tool provides the user with functionality that is not available in RDBMSs with SQL queries, which is an inference of subsumption queries. Another advantage is the possibility to query the data source through the ontological domain model.

RDBToOnto [21]: is a tool that eases the design and implementation of methods for ontology learning from relational databases. It supports an iterative approach for refinement of the learning process through the definition of constraints by users. RDBToOnto's development is oriented towards the recognition of categorization structures by jointly analyzing the database schema and stored data. Thus, the RTAXON converter, central in this tool, implements a generic method of recognition of categorization attributes, based on the one hand on the name of the attribute and the other hand on the redundancy in the extensions of the attributes using an entropy-based method

OWL to ER and ER to OWL [22]: is an approach based on conceptual graphs. The first step is the transformation of OWL ontology to ER and the second step is the transformation from ER to a relational database. The tool that supports this approach is not entirely automatic, and only central OWL constructs are covered.

OntoRel [23]: is specified as a tool providing a mechanism to transform the OWL ontology into a relational database. In this approach, ontology is first generated from an XML document using the OntoGen tool, which is a semi-automatic tool for ontology generation, and then the OWL ontology is transformed into a relational model by implementing the OntoRel tool and applying specific transformation rules. The disadvantage of this tool is that it only transforms the main components of the ontology.

Hybrid approach [24]: this tool transforms OWL 2 ontologies into relational databases using a hybrid approach. A part of ontology constructs is directly represented by relational database structures, the other part with no direct correspondences in a relational database is stored in metadata tables. It combines the direct representation of ontology classes, properties, and instances in database tables with representing axioms and restrictions in metatables. According to the authors. The correctness of the transformation means that for every construct of OWL 2 ontology metamodel the direct and reverse transformation exist and they are related.

OWLMAP [25]: is a fully automatic tool that serves in mapping ontology (OWL) to relational database format. Information is extracted according to each ontology construct. Next, proposed mapping rules are applied automatically to ensure lossless transformation. Proposed mapping rules are as follows: ontology classes are transformed into tables, object type properties are mapped into columns or tables, data type properties

should be mapped into columns or tables according to mapping rules, and restrictions are stored into metadata tables.

According to the study of similar works, the works dedicated to each conversion way are about the same number, however, the literature shows that the works of databases conversion to ontologies based on the logical model are more frequent. As indicated by [11], mostly, research work has been dedicated to direct transformation from databases to ontologies, but little effort has been made to develop research on the conversion of ontologies into databases. We think that the loss of semantics when converting an ontology into a database due to the difference between the two artifacts regarding language representation expressivity is the most important reason for this observation.

Approaches transforming ontologies to relational databases should tackle the issue of structure, and data loss, i.e., commonly constraints are ignored. Most of the proposed conversion tools are incomplete and miss essential OWL constructs. Besides, most of them are not fully implemented.

Table 2. Comparative table of works dealing with conversion between RDBs and ontologies

Related works	Way of conversion		Mapping rules Definition	Approach Category	Automaticity	limitations
	RDB to Ontology	Ontology to RDB				
D2R MAP [5]	✓		predefined rules	Conceptual Model	Fully automatic	No support for Named Graphs and inference
Ontology Modeler/Document & Resource Manager [12]		✓	predefined rules	Logical model	Fully automatic	Some object Properties are lost during the transformation process, and property restrictions are not considered
Relational.OWL [7]	✓		predefined rules	the logical model	Fully automatic	Use of OWL Full which limits decidability of the resultant ontology
KAON2 [13]	✓		Manual definition of rules	Logical model	Fully automatic	Inheritance relationships are ignored
Vis- à-vis [6]	✓		manual definition of mapping rules	Logical model	Semi-automatic	Performs mapping, not conversion and the user is heavily involved
DBOM [14]	✓		predefined rules +possible configuration by the user	Logical model	Semi-automatic	the user is involved and must master SQL
OWL2DB based Approach [15]		✓	predefined rules	Logical model	Fully automatic	Only Part of OWL DL syntax is covered
DATAMASTER [16]	✓		predefined rules	Logical model	Fully automatic	Database Instances and inheritance relationships are ignored
DB2OWL [17]	✓		predefined rules	Logical model	Fully automatic	Only some fundamental mapping rules are applied
OntER [18]		✓	predefined rules	Intermediate model	Fully automatic	Oriented design, the result of ontology conversion is a conceptual schema of an RDB
RDB2ONTO [19]	✓		predefined rules + possible user configuration	Conceptual model	Fully automatic	Multiple inheritances is excluded, two levels of maximum depth obtained
OBDA [20]	✓		predefined rules	Logical model	Fully automatic	the limited expressiveness of DL-Lite
RDBToOnto [21]	✓		predefined rules	Logical model	Fully automatic	No recommendations on how to select data to be transformed
OWL to ER and ER to OWL [22]		✓	predefined rules	Intermediate model	Not fully automatic	Only main OWL constructs are covered
OntoRel [23]		✓	predefined rules	Logical model	Semi-automatic	Only transforms the main components of the ontology
Hybrif Approach [24]		✓	predefined rules	Logical model	Fully automatic	Query capabilities are limited
OWLMAP [25]		✓	predefined rules	Logical model	Fully automatic	Some ontological constructs are not taken into accounts, such as Class complements, intersection classes and Reflexive and Irreflexive properties
Our proposal	✓	✓	predefined rules	Logical model	Fully automatic	OWL 2 constructs are not taken into account

To the best of our knowledge, no method performs mutual conversion between relational database and ontology, each of the methods presented in this section deals with a single direction of transformation, from the RDB to the ontology, or from the ontology to the RDB and which is fully automatic. The notion of inheritance in both artifacts is supported by our approach and tool concerning classes, attributes, and relationships. Besides,

the most commonly used constraints are supported such as uniqueness, not null, a single value or multiple value constraints.

IV. Motivation

The conversion can be defined as the process of changing or causing something to switch from one form to another. We aim to establish a conversion approach that allows for a crossover transformation between relational databases and ontologies, as well as the implementation of a tool to perform the task of conversion automatically. The primary motivations behind this proposal are:

A. Motivation for Conversion from Relational Database to Ontology

Ontologies are increasingly used because they allow automatic manipulation and smarter access to data by taking into account the semantic dimension of data (through the formal languages of ontologies) [26]. Whereas, relational databases store data expressed as terms and values that cannot be interpreted by machines. In this context, an ontology can act as an intermediary for the integration of heterogeneous databases provided that the ontology correctly covers the concepts and relationships related to the knowledge domain of the databases to be integrated [27, 28]. In what follows, we explain some other reasons that lead to the conversion: relational databases-ontologies.

- Exploitation of the semantic reasoner: The semantic reasoner applied on an ontology makes it possible to infer new rules from the rules already described in the ontology. So relational databases would benefit from being converted into ontologies to take advantage of this characteristic of ontology and continually enrich their conceptual schema [29, 30, 31, 32].
- Complex construction process: The development of an ontology is a time-consuming and challenging process. As for databases, they are defined for a specific field of application, and database development methods are now mature and well mastered by a large number of developers. So it is easier to design and implement a database then convert it into ontology than to develop an ontology from scratch.
- Databases can exploit web-based capabilities offered by ontologies. Availability of web applications such as OntoQuery or AmiGO, enabling the exploitation of ontologies by providing interfaces for the interrogation of ontologies (using the SPARQL language which is a query language for RDF data) and the display of results according to different formats as well as various functionalities such as syntax assistance and validation of SPARQL queries entered. Several web applications, such as Ontology Lookup Service or WebVOWL, provide an interactive visualization and ontology discovery service. They allow display of the elements of ontologies and also have interaction techniques to allow an in-depth exploration of ontologies.

B. Motivation for Conversion from Ontology to Relational Database

A large number of ontologies on various domains are available on the web which encourages and enhances their reuse. The ontology is used for the modeling of the conceptual data primarily for reasons of sharing and reuse. So ontology is a representation of consensual knowledge and can be reused in whole or in part. Ontologies have been widely used to automate the process of integrating heterogeneous databases. The advantages of using ontologies for the development of relational databases can be summarized as follows:

- Reusability of available knowledge representations: If an ontology representing the domain of a database is available, and has been recognized by the community of experts and users as valid, the design and implementation of this database will be more comfortable and less time-consuming [33].
- Avoid the non-decidability of ontologies: The OWL FULL language is the most complex version of an OWL, it allows the highest level of expressiveness and is intended for situations where it is essential to have a high level of representative capacity but without guaranteeing the decidability and completeness of the calculations made on the ontology. So to take advantage of the content of FULL ontologies and avoid incompleteness and non-decidability when queried, converting the OWL FULL ontology into a relational database can be considered as a solution to this problem because databases are structures that ensure decidability. However, a considerable loss of semantics remains inevitable, because it is a passage from a very expressive language to a language much less expressive.
- Exploit the maturity of RDB technologies: Relational databases have been on the market since the 1970s. Large firms such as Oracle, MySQL AB, Microsoft have ensured this dominance through the creation and the continuous evolution of their database management systems (Oracle DataBase, Microsoft SQL Server, MySQL). This evolution can be summed up in several features added to these systems, such as interoperability and portability, Programmable Logical Structured Query Language (PL / SQL), for writing functions and procedures, and so on. DBMS providers are continuously improving the speed of processing and the reduction of storage space as well as adapting their products to the needs of companies [34]. As a result, ontologies could benefit from this maturity of RDB technologies by being converted into relational databases.

V. Proposed Conversion Approach

Our application is based on the OWL language for the manipulation and description of ontologies, and the SQL language for databases. The conversion process, in both directions, goes through two main steps, acquiring and classifying metadata from the ontology or source RDB, and then building the RDB or the corresponding ontology according to a set of mapping rules that we will specify for each direction of conversion.

A. Conversion Approach from RDB to Ontology

The process of converting an RDB into an ontology begins by collecting the set of information from the RDB schema that will be used to build the corresponding ontology. The set of information to be collected is summarized in the name of each table, all of its primary and foreign keys, their types, tables providing each foreign key and the rest of the attributes and their respective types.

The broad outlines of the process of transforming a database into an ontology are as follows:

1. Metadata of each table is extracted and stored so that it can be mapped into ontology components.
2. Schema tables are transformed into classes according to mapping rules.
3. Referential integrity constraints, or foreign keys, are mapped to object properties in the ontology according to mapping rules.
4. Attributes are transformed into data properties in the ontology.
5. Tuples are converted into instances in the ontology.

Our approach is based on a classification of database tables that will be taken into consideration during the construction process of the corresponding ontology.

Notation

Let T be a table of a relational database DB, $Col\ (T)$, the set of columns of the table T. $PK\ (T)$ the set of primary keys of the table T and $FK\ (T)$ the set of foreign keys of the table T. We also note $PFK\ (T)$, the set of keys that are both primary and foreign for the table T and $A\ (T)$ the columns that are simple attributes in the table T (they are neither primary keys nor foreign keys). We also put the *RIC* notation as the set of referential integrity constraints to specify the foreign key containing table and the table providing the foreign key. This relationship will be represented by the *RIC* triple $(T1, A, T2)$ where $T1$ is the table containing the foreign key, $T2$ is the table that provides the foreign key. A is the foreign key that belongs to $T1$ and referenced from a column of $T2$.

- *Tables Classification*

Our conversion approach from RDB to Ontology classifies the tables in the database into four categories. We will rely on the following sample database to provide examples on the tables' categories. Primary keys are in bold and foreign keys are underlined.

CLIENT (**NumClient**, *NumEnterprise*, *NameClient*, *NSS*)
VEHICLE (**NumVehicle**, *NumBrand*, *Year*)
LOCATION (*NumClient*, *NumVehicle*, *Amount*)
CAR (*NumVehicle*, **NumCar**, *NumBrand*, *Color*, *Type*)
BRAND (**NumBrand**, *NameBrand*)
SUPPLIER (**NumSupplier**, *Address*)
BRANDSUPPLY (*NumSupplier*, *NumBrand*)
ENTERPRISE (**NumEnterprise**, *Address*, *NameEnterprise*)

Category 1 Definition: a table T is classified in this category, if it is linked to a table $T1$ by a referential integrity constraint. The common attribute between the two tables is a foreign key for the table T and is at the same time a primary key in the table T. This common attribute also corresponds to the primary key of the table $T1$. This case corresponds in RDB to an inheritance relationship between two tables (table T inherits from table $T1$), and this can be formulated as follows:

$$RIC\ (T, A, T1) = PFK\ (T) \wedge PK\ (T1) \qquad (1)$$

Example: Consider the CAR table in the sample database. *NumVehicle* is part of the primary key of the CAR table $PK\ (CAR) = \{NumVehicle, NumCar\}$, it is also in the referential integrity constraint $RIC\ (CAR, \{NumVehicle\}, VEHICLE)$, which makes the *NumVehicle* column a primary and foreign key in the *CAR* table, referring to the primary key in the *VEHICLE* table. So the *CAR* table inherits from the *VEHICLE* table and therefore ranks in this first category.

Category 2 Definition: When a table T is used to link two other tables $T1$, $T2$ in a many-to-many relationship, it can be divided into two disjoint columns $A1$, $A2$, each participating in a referential constraint with $T1$ and $T2$ respectively:

$$\begin{aligned} &RIC\ (T) = \{ric1 \wedge ric2\}: \\ &ric1\ (T, A1, T1)\ and\ ric2\ (T, A2, T2) \end{aligned} \qquad (2)$$

All the columns of T are foreign keys and primary keys: $Col\ (T) = FK\ (T) = PK\ (T)$, therefore: $Col\ (T) = PFK\ (T)$. Thus, the T table is classified in the category 2.

Example: Consider the table "BRANDSUPPLY" which is composed of the two columns $\{NumSupplier, NumBrand\}$.

$PK\ (BRANDSUPPLY) = \{NumSupplier, NumBrand\}$
and $FK\ (BRANDSUPPLY) = \{NumSupplier, NumBrand\}$,
so,
$PFK\ (BRANDSUPPLY) = Col\ (BRANDSUPPLY)$.

Besides, $RIC\ (BRANDSUPPLY) = \{ric1 \wedge ric2\}$
where:

$ric1 = (BRANDSUPPLY, \{NumSupplier\}, SUPPLIER)$,
and $ric2 = (BRANDSUPPLY, \{NumBrand\}, BRAND)$,
so, the BRANDSUPPLY table belongs to the second category.

Category 3 Definition: The tables in this category are similar to tables belonging to category 2, in addition they include at least one non-key attribute, which can be noted as follows:

$$FK\ (T) = PK\ (T) = PFK\ (T) \neq Col\ (T) \quad (3)$$

Example: let consider the LOCATION table, which can be described as follows:

$PK\ (LOCATION) = \{NumClient, NumVehicle\}$,
and $FK\ (LOCATION) = \{NumClient, NumVehicle\}$,
so, $PFK\ (LOCATION) = \{NumClient, NumVehicle\}$,
and, $Col\ (LOCATION) = \{NumClient, NumVehicle, Amount\}$,
so, $PFK\ (LOCATION) \neq Col\ (LOCATION)$.
LOCATION table belongs to category 3.

Category 4 Definition: In this category are ranked tables that do not meet the requirements of the previous categories. Example: tables that belong to this category in our sample database are: The ENTERPRISE table, described by:

$Col\ (ENTERPRISE) = \{NumEnterprise, Address, NameEnterprise\}$,
and, $PK\ (ENTERPRISE) = \{NumEnterprise\}$
and $FK\ (ENTERPRISE) = \varnothing$,
and the CLIENT table, described by:
$Col\ (CLIENT) = \{NumClient, NumEnterprise, NameClient\}$
where $PK\ (CLIENT) = \{NumClient\}$
and $FK\ (CLIENT) = \{NumEnterprise\}$.

- *Rules for Mapping RDB Elements into Ontology Elements*

In the mapping process, we use obtained information from the source database, to build an ontology. We explain in the following the mapping rules of each category.

Category 1 Mapping Rules: Category 1 tables will be mapped into subclasses of classes corresponding to their parent tables. If T is in category 1, then there is a referential integrity constraint

$RIC\ (T, PFK\ (T), T1)$, *where* $PFK\ (T) = PK\ (T1)$.

So T is mapped to the subclass of the class corresponding to $T1$. Example: The *CAR* table is transformed into a subclass of the class corresponding to the *VEHICLE* table.

Category 2 Mapping Rules: Category 2 tables will not be mapped into classes, as these tables are used to link two other tables (in the case of many-to-many relationships), without containing any additional information (attribute). These tables will be transformed into object properties linking the two corresponding classes to the two tables associated with a many-to-many relationship. Two object properties will be added, one for each class. In other words, when a table T is in the second category, there are two referential constraints, moreover, if we consider that $C1, C2$ the two corresponding classes to $T1, T2$ respectively, so we assign to $C1$ *an object property* $Op1$ *whose range is* $C2$, and to $C2$ *the object property* $Op2$ *whose range is* $C1$ while specifying these two properties $Op1, Op2$ being inverse of each other. Example: in our sample database, the *SUPPLYBRAND* table falls into this category, it links two other tables: *SUPPLIER and BRAND*, so it is mapped to a first objet property: *BRAND.SUPPLIER* property with *SUPPLIER* class as domain, and its range is the *BRAND* class, and we define *SUPPLIER.BRAND* inverse object property whose range is the *SUPPLIER* class and domain the *BRAND* class.

Category 3 Mapping Rules: Unlike category 2 tables, category 3 tables will be mapped into classes because they stand for connecting two tables with a many-to-many relationship while carrying additional information (attributes that are not keys). For that, they will be mapped into classes, to which we will assign two functional object properties, each linking this class to one of the classes corresponding to the two tables linked with the many-to-many relationship. Moreover, for each of the two properties created, we will specify an inverse property. In other words, we have two referential constraints:

$ric1\ (T, A, T1)$ *and* $ric2\ (T, A, T2)$ *and*
$C, C1, C2$ *be the classes corresponding to* $T, T1, T2$ *respectively*

So we attribute to C a functional object property $Op1$ whose range is $C1$, and a second functional property $Op2$ whose range is the class $C2$, then the inverse properties will be created for each of the properties.

Example: in our sample database, *LOCATION* table falls into category 3, since it links two tables: *CLIENT* and VEHICLE while having "Amount" attribute. So, it will be mapped to a *LOCATION* class. A functional *LOCATION.CLIENT* object property will have as a domain the class *LOCATION*, and as a range: the *CLIENT* class (it will be specified as being functional to say that each instance of the *LOCATION* class

corresponds to a single instance of the *CLIENT* class), its inverse *CLIENT.LOCATION* will be created and will thus have as domain the *CLIENT* class and as range the *LOCATION* class.

Category 4 Mapping Rules: If a table T is in category 4 and has a referential integrity constraint $ric(T, A, T1)$, and we assume that C, $C1$ are the classes corresponding to T, $T1$ respectively. We assign to C an object property Op whose range is $C1$, and we assign to $C1$ an object property $Op1$ whose range is C to preserve the original direction of the referential constraint from T to $T1$. The object property Op will be specified as functional, so an instance of class $C1$ will correspond at most to a single instance of class C.

Example: *CLIENT* table belongs to category 4. This table has a referential integrity constraint with the *COMPANY* table: $RIC(CLIENT, NumEnterprise, ENTERPRISE)$, we will assign to its corresponding class a functional object property *CLIENT.ENTERPRISE*, whose range is the class corresponding to the *ENTERPRISE* table and we will assign to the corresponding class in the *ENTERPRISE* table an *ENTERPRISE.CLIENT* object property whose range is the *CLIENT* corresponding class.

Mapping of Non-Referential Columns: For all tables, we complete the conversion by adding non-referential columns (neither primary nor foreign keys), to their corresponding classes in ontology as data properties. Each data property will have the same name as its corresponding column in RDB, and the type will be assigned to it according to a table of types' correspondences (Table 3). Example: in the *BRAND* table, the *NameBrand* attribute will be mapped to a data property *BRAND.NameBrand* in the *BRAND* class.

Mapping of the "Unique" Constraint: A column with the unique constraint means that its value must be unique to each tuple of the table. Following the definition of a functional property, the unique constraint will be mapped to a specified data property as functional. Because by definition, *if P is a property marked as functional*

Then *for all* $P(x, y)$ *and* $P(x, z)$ *we have* $y = z$.

Example: In the *CLIENT* table, there is the NSS column, this attribute must be unique for each client in the *CLIENT* table, so it must be specified as Unique. This attribute will be mapped to a *CLIENT.NSS* data property of the corresponding *CLIENT* class in ontology and marked as functional. The type of the property will be assigned according to the table of correspondences between SQL and XSD types.

Mapping of Not Null Constraint: A column that has the Not Null constraint means that for any row in the table, this attribute must have a value. This constraint will be mapped into a restriction on the data property, i.e., the minimum cardinality 1:

$< OWL: minCardinality\ rdf > 1 < / OWL: minCardinality >$

Example: The NSS attribute in the client table will be mapped to a data property with the restriction of minimum cardinality equals to 1.

Mapping of Table Tuples: The tuples contained in the database tables will be mapped to individuals in the ontology such that, for each tuple, an individual (instance) of the corresponding class is created, and the attributes values will be mapped to data properties values.

Mapping of Types or Cross-Type Mapping Table: In Table 3, we will present the correspondence between XSD types and SQL data types.

At the end of this subsection, we summarize the set of correspondences between relational database elements and ontology elements in Table 4.

Table 3. Correspondence between SQL and XSD data types, transition from the RDB to the ontology

SQL Type	XSD Type
TINYINT	INTEGER
SMALLINT	INTEGER
MEDIUMINT	INTEGER
INT	INTEGER
BIGINT	LONG
NUMERIC	DECIMAL
FLOAT	FLOAT
DECIMAL	DECIMAL
REAL	DOUBLE
DOUBLE	DOUBLE
CHAR	STRING
VARCHAR	STRING
TINYTEXT	STRING
TEXT	STRING
MEDIUMTEXT	STRING
LONGTEXT	STRING
DATE	DATE
DATETIME	DATETIME
TIMESTAMP	INTEGER
YEAR	INTEGER
TIME	TIME

B. Conversion Approach from Ontology to RDB

The process of converting an ontology into an RDB begins by loading the OWL file containing the source ontology using the Jena API. Using the different methods provided by Jena, information about the source ontology components is extracted. The extracted information consists in: classes, object properties linking these classes, data properties, sub-properties (object and data properties), and properties' types. Based on the extracted information, the construction of the RDB corresponding to the initial ontology is carried out according to the different mapping rules that describe the correspondence between the ontology elements and the RDB elements.

The broad outlines of the process of transforming an ontology into a database are as follows:

1. The ontology contained in an OWL file is loaded, necessary information for conversion is extracted using the Jena API.
2. Connection to MySQL for building the RDB using JDBC.
3. Classes and subclasses are transformed into separate tables, and relationships between classes are created according to defined mapping rules.

4. Object properties are transformed into relationships between schema tables according to mapping rules.

5. Datatype properties are mapped into attributes in the tables corresponding to their classes. Data properties with sub-properties are transformed into tables.

6. The process of switching from ontology to relational database will be completed by transforming the tuples of each database table into instances of the corresponding class ontology.

Table 4. Summary of mapping rules for the transition from RDB to ontology

RDB Element		Equivalent Element in the OWL ontology	Contribution
Nonassociative Table		Class	Existing Rule
Inheritance between two tables		Inheritance between two classes (<owl: SubClassOf)	Existing Rule
Primary key		URI	Existing Rule
Foreign Key: the case of relation 1 to many (n)		Functional object property	Existing Rule
Foreign Key: the case of many to many relation	Associative table without attributes	Two object properties, one inverse to the other, linking the two corresponding classes to the tables linked by the many to many relationships.	Proposed as part of our work
	Associative table with attributes	A class linked by two functional object properties to classes corresponding to tables linked by the many to many relations. The attributes of the associative table are mapped to data properties in the class.	Proposed as part of our work
Table column		Data property	Existing Rule
Not Null Attribute Constraint		Restriction <owl :minCardinality rdf>1</owl :minCardinality> on the data property corresponding to the column	Existing Rule
Unique Attribute Constraint		Make data property functional	Proposed as part of our work

- *Mapping Rules*

Classes: Each class will be transformed into a table in the relational database, it will have the same name of this class and a primary key written as ID_TableName will be assigned to it. A table that corresponds to a subclass will have as primary key, the foreign key that refers to its parent table.

Object Properties: are mapped according to their characteristics as follows:

- **A functional object property** will be mapped to a foreign key in a table. As, the object property domain will be the table containing the foreign key, the range of the object property will be the table referenced by the foreign key. The name of the foreign key will have the name of the object property. Example: the following OWL functional object property

```
- <owl:ObjectProperty rdf:about="http://www.NewOnto1.org/ontology1#is_wife_of">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.NewOnto1.org/ontology1#Man"/>
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Woman"/>
  </owl:ObjectProperty>
```

This property is translated into natural language by the fact that a woman can have one and only one husband. In a relational database, this axiom is translated by creating a foreign key in the table "Woman" corresponding to the domain class of the property, which will be named "is_wife_of", and which will refer to the primary key of the table Man corresponding to the range of the object property.

- **An inverse functional object property** will be mapped to a foreign key in the table that corresponds to the range of the object property. This foreign key refers to the primary key of the table corresponding to the domain of the object property. The foreign key will have the same name as the object property, preceded by the prefix "Inverse. ». Example: In the same context of the previous example, the following inverse functional object property:

```
- <owl:ObjectProperty rdf:about="http://www.NewOnto1.org/ontology1#is_husband_of">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Man"/>
    <rdfs:range rdf:resource="http://www.NewOnto1.org/ontology1#Woman"/>
  </owl:ObjectProperty>
```

According to the definition of inverse functional object property that we have already discussed, and which indicates that if a subject is linked to an object by an inverse functional predicate, that subject is the only subject for that object. For this example, the natural language translation is that a woman can only be married to one man. This inverse functional property is mapped in the RDB, into a foreign key in the Woman table corresponding to the range of the property, referring to the key of the Man table, corresponding to the domain class.

- **An object property that is not specified as functional or inverse functional** will be mapped to a table (Associative table), its primary key will be the combination of two foreign keys. One of the

two foreign keys referring to the primary key of the corresponding table to the domain and the other to the corresponding table to the range of the property. The name of the object property will be assigned to this associative table. Example of the following object property:

```
- <owl:ObjectProperty rdf:about="http://www.NewOnto1.org/ontology1#attends">
    <rdfs:range rdf:resource="http://www.NewOnto1.org/ontology1#Course"/>
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Student"/>
  </owl:ObjectProperty>
```

This object property represents the relationship between students and courses, several students can attend the same course, and a student can attend several courses. This relation is of many to many type. It will be mapped in a table having the name of the object property "Assists" and its primary key will be the combination of two foreign keys. One referring to the primary key of the table "Student" corresponding to the domain, it will, therefore, have the same name as this primary key. The second one referring to the primary key of the table "Course" corresponding to the range and will also have the same name as this last one.

Object sub-properties are converted in the same way as object properties since hierarchy between relationships is a concept that cannot be represented in an RDB. Example: The object sub-property "hasSoldOn", according to the mapping rules that have already been defined, will be mapped to an associative table.

```
- <owl:ObjectProperty rdf:about="http://www.NewOnto1.org/ontology1#CarriesOutTransactionsOn">
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Person"/>
    <rdfs:range rdf:resource="http://www.NewOnto1.org/ontology1#SiteOfSale&amp;Purchase"/>
  </owl:ObjectProperty>

- <owl:ObjectProperty rdf:about="http://www.NewOnto1.org/ontology1#hasSoldOn">
    <rdfs:subPropertyOf rdf:resource="http://www.NewOnto1.org/ontology1#CarriesOutTransactionsOn"/>
  </owl:ObjectProperty>
```

Data Properties: Like object properties, data properties are mapped according to their characteristics, and their mapping rules are as follows:

- **A simple data property (single value)** will be mapped into a column in the table that corresponds to the domain class of the property. The name of the data property will be assiged to this property, and the property type will be specified according to the OWL and SQL type mapping table that we will present later. Example of the data property “Age” of integer type with a single value:

```
- <owl:DatatypeProperty rdf:about="http://www.NewOnto1.org/ontology1#Age">
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Employee"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  </owl:DatatypeProperty>
```

The Age data property will be mapped into a column in the "employee" table corresponding to the domain class "Employee", this column will have the same name as the data property, and its type will be specified according to the matching table between types.

- **A multiple value data property** will be mapped to a table because in RDB an attribute cannot accept multiple values for the same instance, for this purpose a table is created, and the same name of the multiple value data property will be assigned to it. A primary key, named ID_Value, will be assigned to this table, as well as a foreign key referring to the primary key of the table corresponding to the domain of the data property, it will have the same name of the primary key of this table. Since the multiplicity of values is at the origin of this table, an attribute named Value will be assigned to this table, and its type will be specified according to the table of correspondences between types. In this way, we can link a set of values to the same tuple of the corresponding table to the domain of the multi-value property.
- **A data property divided into sub-properties** will also be mapped to a table. A primary key: ID_PropertyName will be assigned to it. A Value_PropertyName attribute of the type corresponding to the type of data property to store its value. A foreign key referring to the primary key of the table corresponding to the property class having sub-properties and each of the data sub-properties will be mapped to a column in the table. The table will take the name of the data property, and each column will have the same name as its corresponding sub-property. Example: a data property “Weight” for a canned mixed salad, the “salad” containing “green salad” and “tomatoes”, and their weights are specified as sub-data properties of the weight data property.

```
- <owl:DatatypeProperty rdf:about="http://www.NewOnto1.org/ontology1#SaladWeight">
    <rdfs:subPropertyOf rdf:resource="http://www.NewOnto1.org/ontology1#Weight"/>
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="http://www.NewOnto1.org/ontology1#TomatoWeight">
    <rdfs:subPropertyOf rdf:resource="http://www.NewOnto1.org/ontology1#Weight"/>
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="http://www.NewOnto1.org/ontology1#Weight">
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Food"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  </owl:DatatypeProperty>
```

The weight property will be mapped into a table, with an ID_Weight key, a column for the Value_Weight, a foreign key ID_VariateSaladBox referring to the VariedSaladBox table, and two columns Tomato and GreenSalad corresponding to the two data sub-properties Tomato and GreenSalad.

- **A data property specified as being functional** will be mapped to a column. The equivalent of the functional characteristic in RDB is the Unique attribute constraint.

For example, the NumSS data property, which represents the social security number and has Employee class as its domain, will be mapped to a column with the Unique attribute constraint.

```
- <owl:DatatypeProperty rdf:about="http://www.NewOnto1.org/ontology1#NumSS">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="http://www.NewOnto1.org/ontology1#Employee"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
  </owl:DatatypeProperty>
```

Individuals of the ontology: Each individual in the ontology will be mapped into a tuple in the database. The individual's class will become the table where the tuple will be inserted, and values in data properties will be stored in the attributes values of tuples.

Table of correspondences between types

In Table 5, we present the correspondence between XSD and SQL types when switching from OWL ontology to RDB.

Table 5.Correspondence between XSD and SQL data types, transition from ontology to RDB

XSD Type	SQL Type
STRING	VARCHAR
NORMALIZED STRING	VARCHAR
TOKEN	VARCHAR
NMTOKEN	VARCHAR
NAME	VARCHAR
NC NAME	VARCHAR
LANGUAGE	INTEGER
BYTE	INTEGER
SHORT	INTEGER
INTEGER	
LONG	BIGINT
UNSIGNED INTEGER	INTEGER
NEGATIVE INTEGER	INTEGER
NONNEGATIVE INTEGER	INTEGER
UNSIGNED INT	INTEGER
UNSIGNED LONG	INTEGER
DECIMAL	DECIMAL
FLOAT	FLOAT
DOUBLE	DOUBLE PRECISION
HEXBINARY	VARCHAR
TIME	TIME
DATE	DATE
DATETIME	TIMESTAMP
GYEARMONTH	DATE
GDAYMONTH	DATE
GMONTH	DATE
GDAY	DATE
BOOLEAN	BIT
ANYURI	VARCHAR

At the end of this subsection, we summarize the set of mapping rules for converting an OWL ontology to an RDB in Table 6.

Table 6. Summary of the mapping rules for switching from ontology to RDB

Ontology Element		Equivalent Element in the RDB	Origin or Contribution
Class		Table	Inversely to the rule of transition from RDB to ontology.
SubClassOf		Inheritance relationship between two tables.	Inversely to the rule of transition from RDB to ontology.
Object Property	Non-functional	Associative Table is linking tables corresponding to the domain and the range.	Inversely to the rule of transition from RDB to ontology.
	Functional	A foreign key in the corresponding table to the domain, referring to the primary key of the table corresponding to the range.	Inversely to the rule of transition from RDB to ontology.
	Reverse functional	A foreign key in the table corresponding to the range referring to the primary key of the table corresponding to the domain.	proposed as part of our work
Data properties	Has one value	Column	Inversely to the rule of transition from RDB to ontology.
	Has multiple values	Table linked to the table corresponding to the domain of the property.	proposed as part of our work
	With data sub-properties	Table and sub-properties as columns of the table. Linked to the table corresponding to the domain.	proposed as part of our work
	Functional	Column with a Unique constraint	Inversely to the rule of transition from RDB to ontology.
	With cardinality restriction : Min Cardinality =1	Column with a Not Null constraint	Inversely to the rule of transition from RDB to ontology.

VI. Implementation and Tests

The tool (Fig. 1) is implemented in Java using Jena API library for ontology manipulation and MySQL database as a relational database using JDBC connector. We will present in what follows, some results obtained after executing our conversion tool on a source file: RDB then ontology.

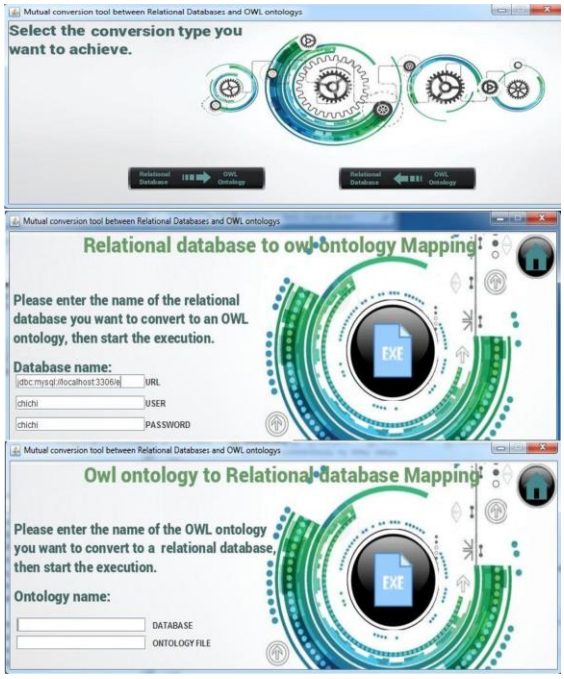

Fig.1. Conversion tool interface

A. *Testing Automatic Conversion of a Sample RDB into Ontology*

For the conversion of a database to ontology, we applied our tool on a sample database which has the following schema (primary keys are in bold and foreign keys are underlined):

PERSON (**idPerson**, *name, first_name*)
STUDENT (*idPerson*,**idStudent**,*idUniversity*, *name, first_name, ssNum*)
UNIVERSITY(**idUniversity**, *nameUniv*, *addressUniv*)
COURSE (**idCourse**, *domainCourse*)
EXAMMARK (*idCourse, idStudent, mark*)

The property ssNum has constraint NOT NULL and UNIQUE. Some results obtained after the conversion of this sample database into ontology are presented through the following OWL code portions:

Conversion of the inheritance relationship between Person and Student tables into an inheritance relationship between the two classes Person and Student, and conversion of the Not Null constraint and ssNum attribute to the minimum cardinality constraint assigned to this property.

```
- <owl:Class rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#Student">
  - <owl:equivalentClass>
    - <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.ConvertedOntology.org
        /OntologyFromRDB#ssNum"/>
        <owl:minQualifiedCardinality rdf:datatype="http://www.w3.org
        /2001/XMLSchema#nonNegativeInteger">1</owl:minQualifiedCardinality>
        <owl:onDataRange rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://www.ConvertedOntology.org
    /OntologyFromRDB#Person"/>
  </owl:Class>
```

Conversion of the relationship represented by the foreign key idUniversity between the tables University and Student into a functional object property between the two classes: University and Student.

```
- <owl:ObjectProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#Student_University">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Student"/>
    <rdfs:range rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#University"/>
  </owl:ObjectProperty>
```

Conversion of a set of attributes from University table into data properties.

```
- <owl:DatatypeProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#adressUniv">
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#University"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#idUniversity">
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#University"/>
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#nameUniv">
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#University"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
```

Conversion of an ExamMark **associative table with attributes** to a class.


```
- <owl:ObjectProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#courseMark">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Course"/>
    <rdfs:range rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Mark"/>
  </owl:ObjectProperty>
- <owl:ObjectProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#markCourse">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Course"/>
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Mark"/>
    <owl:inverseOf rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#courseMark"/>
  </owl:ObjectProperty>
- <owl:ObjectProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#markStudent">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Mark"/>
    <rdfs:range rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Student"/>
  </owl:ObjectProperty>
- <owl:ObjectProperty rdf:about="http://www.ConvertedOntology.org/OntologyFromRDB#studentMark">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Mark"/>
    <rdfs:domain rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#Student"/>
    <owl:inverseOf rdf:resource="http://www.ConvertedOntology.org/OntologyFromRDB#markStudent"/>
  </owl:ObjectProperty>
```

B. Testing Automatic Conversion of Movie Ontology into RDB

We applied the conversion tool on Movie ontology (www.movieontology.org). The result can be summarized as follows: all Movie ontology classes are transformed into tables in the database (Fig. 2). For example, Movie class is transformed into a table in the database (Fig. 3), with a primary key: ID_movie and mapping of movie class data properties into attributes in movie table.

```
movie
musical_artist
musical_entertainment
nominatedfor
northern_africa
northern_america
northern_europe
oceania
old_action
online_retailer
partofterritory
person
place
polynesia
porn
presentation
produced
producer
production_company
scifi_and_fantasy
sensible
sensible_thrilling
socialactive
```

Fig.2. List of tables created from Movie Ontology

```
mysql> describe movie;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| ID_Movie    | int(11)     | NO   | PRI | NULL    | auto_increment |
| title       | varchar(40) | YES  |     | NULL    |                |
| runtime     | int(11)     | YES  | UNI | NULL    |                |
| releasedate | date        | YES  | UNI | NULL    |                |
| imdbrating  | double      | YES  | UNI | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Fig.3. Movie table with its attributes

BelongsToGenre object property, between Movie and Genre classes is transformed into an associative table between Movie and Genre tables in the database (Fig. 4).

```
mysql> describe belongsToGenre;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| ID_Movie  | int(11) | NO   | PRI | NULL    |       |
| ID_Genre  | int(11) | NO   | PRI | NULL    |       |
+-----------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig. 4. Screenshot of an associative table

Inheritance between the two classes Person and Writer transformed into inheritance relationship (Fig. 5) between the table Person (mother table) and the table Writer (daughter table).

```
mysql> describe writer;
+-----------+---------+------+-----+---------+----------------+
| Field     | Type    | Null | Key | Default | Extra          |
+-----------+---------+------+-----+---------+----------------+
| ID_Writer | int(11) | NO   | PRI | NULL    | auto_increment |
| ID_Person | int(11) | NO   | PRI | NULL    |                |
+-----------+---------+------+-----+---------+----------------+
```

Fig.5. Screenshot of an inheritance relation

VII. Conclusion

We believe that relational databases and ontologies can benefit from each other, by exploiting their respective content in the process of building databases and ontologies. Though, the conversion between relational databases and ontologies remains a research issue because no standardized mapping rules have been adopted by a large community of users. Proposed

solutions differ from each other and have significant drawbacks, such as the loss of semantics when converting an ontology to a database and lack of support to inheritance relationships and instances in the process of turning a database into an ontology which profoundly limits reasoning capabilities.

The goal of this work is to propose a complete set of mapping rules between ontologies and relational databases throughout an approach supported by a tool that allows an entirely automatic mutual conversion between the two artifacts. This feature is exclusive to our tool because related works handle one conversion direction, and all of the mapping rules we have defined and on which our tool is based are, in our view, a consistent set of rules which ensure a usable and operational result.

Nevertheless, our work has several limitations, explicitly considering the set of novelties provided by the OWL 2 version that is not supported by our conversion proposal. This limitation implies losses in the passage from ontology to RDB.

Hence, several improvements are possible, including:

- Support the specifics of version 2 of OWL, such as intersection, union, complementarity, property symmetry, and irreflexivity
- Pre-selection of the subset of the artifact to be converted, this allows more flexibility in which the user selects the set of entities he wants to transform
- Comparison of our conversion tool with related works
- Ensure compatibility of or tool with other popular DBMS, such as Oracle and SQL server
- Conduct more experimentations to test scalability.

REFERENCES

[1] L. Zemmouchi-Ghomari, "Cohabitation of Relational Databases and Domain Ontologies in the Semantic Web Context", *Journal of Systems Integration*, Vol.9, No.1, pp. 42-57, 2018.

[2] C. Coronel and S. Morris, *Database Systems: Design, Implementation, and Management*, 1978.

[3] M. Dadjoo and E. Kheirkhah, "An approach for transforming of relational databases to OWL ontology". arXiv preprint arXiv:1502.05844, 2015.

[4] T. R Gruber, "A translation approach to portable ontology specifications", *Knowledge acquisition*, Vol.5, No.2, pp.199-220, 1993.

[5] C. Bizer, "D2R MAP - a database to RDF mapping language", in Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary, 2003.

[6] N. Konstantinou, D. Spanos, M. Chalas, E. Solidakis and N. Mitrou, "VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents", in Proceedings of Workshops and Doctoral Consortium, the 18th International Conference on Advanced Information Systems Engineering - Trusted Information Systems, Luxembourg, Luxembourg, 2006.

[7] C. P. De Laborda and S. Conrad, "Relational.OWL: a data and schema representation format based on OWL", in Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling. Newcastle, Australia, 2005.

[8] M. R. C. Louhdi, H. Behja and S. O. El Alaoui, "Transformation rules for building owl ontologies from relational databases", in Second International Conference on Advanced Information Technologies and Applications, pp. 271-283, 2013.

[9] S. M. Benslimane, M. Malki and D. Bouchiha, "Deriving Conceptual Schema from Domain Ontology: A Web Application Reverse Engineering Approach", *International Arab Journal of Information Technology*, Vol.7, No.2, pp.167-176, 2010.

[10] H. El-Ghalayini, M. Odeh, R. McClatchey and T. Solomonides, "Reverse Engineering Domain Ontology to Conceptual Data Models", in Proceedings of the 23rd IASTED International Conference on Databases and Applications (DBA), Innsbruck, Austria, pp. 222-227, 2005.

[11] A. Humaira, N. Tabbasum, S. Ayesha, "A Survey on Automatic Mapping of Ontology to Relational Database Schema", *Research Journal of Recent Sciences*, Vol.4, pp.66-70, 2015.

[12] A. Gali, C. X. Chen, K. T. Claypool and R. Uceda-Sosa, "From ontology to relational databases", in International Conference on Conceptual Modeling. Springer, Berlin, Heidelberg, pp.278-289, 2004.

[13] B. Motik and R. Studer, "KAON2–a scalable reasoning tool for the Semantic Web", in Proceedings of the 2nd European Semantic Web Conference, ESWC'05, Heraklion, Greece, 2005.

[14] O. Curé and R. Squelbut, "Integrating data into an OWL Knowledge Base via the DBOM Protégé plug-in", in Proceedings of the 9th International Protégé conference, Stanford, California, US, 2006.

[15] E. Vyšniauskas and L. Nemuraite, "Transforming Ontology Representation from OWL to Relational Database", *Information Technology and Control*, Vol.35, No.3, pp. 333-343, 2006.

[16] C. Nyulas, M. O'Connor and S. Tu, "DataMaster a Plug-in for Importing Schemas and Data from Relational Databases into Protégé", in Proceedings of 10th International Protégé Conference, Budapest, Hungary, 2007.

[17] N. Cullot, R. Ghawi and K. Yétongnon, "DB2OWL: A Tool for Automatic Database-to-Ontology Mapping", in proceedings of the 15th Italian Symposium on Advanced Database Systems, Ginosa, Italy, 2007.

[18] J. Trinkunas and O. Vasilecas, "Building ontologies from relational databases using reverse engineering methods", in Proceedings of the international conference on Computer systems and technologies, Rousse, Bulgaria, 2007.

[19] M. Laclavık, "RDB2Onto: Relational database data to ontology individuals mapping", in: Tools for Acquisition, Organisation and Presenting of Information and Knowledge, pp.86–89, 2006.

[20] A. Poggi, M. Rodriguez and M. Ruzzi, "Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé, in proceedings of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC), Vol.496, Karlsruhe, Germany, 2008.

[21] F. Cerbah, "Learning highly structured semantic repositories from relational databases", in European Semantic Web Conference, pp. 777-781, San Jose, California, USA, 2008.

[22] S. H. Tirmizi, J. Sequeda and D. Miranker, "Translating SQL applications to the semantic web", in International Conference on Database and Expert Systems Applications,

pp. 450-464, Turin, Italy, 2008.
[23] D. De Brum Saccol, T. de Campos Andrade and E. K. Piveta, “Mapping owl ontologies to relational schemas”, in Information Reuse and Integration (IRI), Las Vegas, NV, USA, 2011.
[24] E. Vyšniauskas, L. Nemuraitė, R. Butleris and B. Paradauskas, “Reversible lossless transformation from OWL 2 ontologies into relational databases”, *Information Technology and Control*, Vol. 40, No.4, pp. 293-306, 2011.
[25] H. Afzal, M. Waqasa and T. Naz, “OWLMap: Fully Automatic Mapping of Ontology into Relational Database Schema”, *International journal of advanced computer science and applications*, Vol.7, No.11, pp. 7-15, 2016.
[26] V. Jain and M. Singh,"Ontology Development and Query Retrieval using Protégé Tool", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.5, No.9, pp .67-75, 2013.
[27] H. Abbes and F. Gargouri, "MongoDB-Based Modular Ontology Building for Big Data Integration", *Journal on Data Semantics*, Vol.7, No.1, pp1-27, 2018.
[28] S. Zhao, Q. Qian, "Ontology based heterogeneous materials database integration and semantic query", *AIP Advances*, Vol.7,No.10, 105325, 2017.
[29] L. Zemmouchi-Ghomari, A. R. Ghomari, L. Adjir and L. Belaala, “Integrating an ontology into a software system”, *Journal of Systems Integration*, Vol.8, No.3, pp.27-39, 2017.
[30] N. S. Ougouti, H. Belbachir and Y. Amghar, “A new owl2 based approach for relational database description”, *International Journal of Information Technology and Computer Science*, Vol.7, No.1, pp. pp.48-53, 2015.
[31] G. Yang and J. Feng, “Database Semantic Interoperability based on Information Flow Theory and Formal Concept Analysis”, *International journal of information technology and computer science*, Vol.7, pp.33-42, 2012.
[32] K. Zarour and N. Zarour, "Data Center Strategy to Increase Medical Information Sharing in Hospital Information Systems", *International Journal of Information Engineering and Electronic Business*, Vol. 1, pp. 33-39, 2013.
[33] E. Ong, Z. Xiang, Z., Zheng, J. Smith and Y. He, "Ontobull and BFOConvert: Web-based Programs to Support Automatic Ontology Conversion", *In International Conference on Biological Ontology and BioCreative, Corvallis*, OR, USA, 2016.
[34] M. Mohamed Hamri and S. M. Benslimane, "Building an Ontology for the Metamodel ISO/IEC24744 using MDA Process", *International Journal of Modern Education and Computer Science*, Vol.7, No.8, pp.48-60, 2015.

Authors’ Profiles

Leila Zemmouchi-Ghomari is currently a Lecturer at ENST: Ecole Nationale Supérieure de Technologie, Algiers, Algeria.

She received her PhD in Computer Science from ESI, Ecole Nationale Supérieure d’Informatique, Algiers, Algeria, in January 2014.

Her research interests focus on Ontology Engineering, Web of Data and Linked Data.

Abdelaali Djouambi received his Research Master in Software Engineering (Computer Science) from USTHB (Université des Sciences et des Technologies Houari Boumediene), Algiers, Algeria in June 2017.

His research interests include Ontology Engineering and Databases Design.

Cherifa Chabane received her Research Master in Software Engineering (Computer Science) from USTHB (Université des Sciences et des Technologies Houari Boumediene), Algiers, Algeria in June 2017.

Her research interests include Software and Web Development

How to cite this paper: Leila Zemmouchi-Ghomari, Abdelaali Djouambi, Cherifa Chabane, " Proposal for a Mutual Conversion Relational Database-Ontology Approach", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.7, pp. 13-28, 2018.DOI: 10.5815/ijmecs.2018.07.02