Cohabitation of Relational Databases and Domain Ontologies in the Semantic Web Context

Leila Zemmouchi-Ghomari
University of Science and Technology, Houari Boumediene, Algeria
leila.ghomari@enst.dz

DOI: 10.20470/jsi.v9i1.334

Abstract: Despite the fact that database technology is mature, it is not yet compatible with Semantic Web requirements, such as Semantic Web language concordance. On the other hand, ontology technology is the most promising solution for concretizing the Semantic Web vision; however, it has not been recognized as the Semantic Web data model thus far. This paper aims to bridge the gap between ontologies and databases in the context of the Semantic Web by highlighting their advantages and disadvantages, their complementarity, and the most popular means of their mutual conversion.

Key words: Semantic Web, Relational Databases, Domain Ontologies

1. Introduction

The Semantic Web addresses some traditional Web issues, such as the management of unstructured and unconnected data described in most Web pages (Berners-Lee et al. 2001; Lausen & Stolberg 2004) that do not allow the recovery of semantically related content. In the Semantic Web, each information source is extended with a semantically organized representation. There are several methods for enhancing semantics in the Web. The most well-known method is the use of ontologies (Finin et al. 2005). On the other hand, relational database technology guarantees the best features for saving, updating, and manipulating information related to a domain (Vysniauskas & Nemuraite 2006). Although database technology is a mature technology, it is not fully compatible with Semantic Web requirements, such as the use of Semantic Web languages.

The Semantic Web language requirements are as follows (Terzi et al. 2003): (1) global expressive and modeling power for high-level information heterogeneity, (2) syntactic interoperability for manipulation of information by programs with included parsers, and (3) semantic interoperability and reasoning support for identifying correspondences between unidentified and known terms. These requirements are satisfied by formal languages such as the Resource Description Framework (RDF) and Web Ontology Language (OWL).

Furthermore, databases are not compatible with the following Semantic Web specifications. (1) Modification abilities (Silva & Rocha 2003): modification of relational databases into ontologies allows for the content of relational databases to be posted on the Web in conformity with Web concepts. On the other hand, modification of ontologies into databases allows for greater use of ontologies in programs with rich databases. (2) Reusability (Zeeshan et al. 2012): database design has to consider the universe of discourse of a particular program, whereas a conceptualization of a domain ontology ought to consider the subjects independently from the problems that are appropriate for them. (3) Performance: a database based on relational design may consist of many connected tables and the overall design may be complicated; therefore, its access time might be slower than that of hierarchical and network models (Brodie & Liu 2011).

Although ontologies contribute toward improving the Semantic Web, they face the following challenges: (1) determining the constraints of the ontology-centered design of a particular domain because the ontology is required to cover the domain as a whole in order to have a realistic bias and be reusable, (2) automatically managing the growing ontology size (automatic maintenance), and (3) guaranteeing ontology quality with regard to correctness and the mapping with the domain that the ontology represents.

In addition, the following questions remain unanswered (Wernher 2005):

• Which criteria can be used to define knowledge that can be expressed by formal ontologies?
• How can the remaining knowledge be encoded to fulfill the requirements of integration, retrieval, and interoperability?

A large number of similarities and differences can be traced between ontologies and databases. Accordingly, the objective of the present study is to provide a deeper understanding of these two artefacts as well as to highlight the cases in which each artefact is more convenient and when they can be complementary in terms of assisting researchers and developers to select the best solution for Semantic Web issues.

The remainder of this paper is organized as follows. Section 2 provides a description of databases and ontologies. Section 3 is intended to describe the similarities and differences between ontologies and databases from the following perspectives: engineering methodology, and role and usage. Section 4 provides a comprehensive overview of the methods available to generate an ontology model from a data model and vice versa for compatibility with one of the most important Semantic Web requirements, i.e., transformation capabilities. Section 5 highlights the cases in which ontologies and databases, respectively, are more suitable and can be complementary in terms of addressing Semantic Web challenges. Section 6 summarizes the mutual fundamental disparities of ontologies and databases as well as the cases in which ontologies and databases are more convenient. Finally, Section 7 concludes the paper by emphasizing the importance of ontologies and databases to the advancement of the Semantic Web vision.

2. Background

This section presents a description of both artefacts (database and ontology) in terms of structure. Ontologies and databases are partial accounts (to varying degrees) of conceptualizations that must consider the structure and rules of the domain to be modeled.

2.1 Database Description

A database is a consistently organized repository of elementary information (usually as a group of linked tables) that allows easy recovery, updating, analysis, and output of data. Most programs (including anti-virus software, excel spreadsheets, and word processors) are databases at their core. Relational database (RDB) technology is a robust technology that has proven abilities to cope with considerable amounts of data (Goodwin et al. 2005) and offers several possibilities for storing data and processing data by user programs.

Conceptual models are concerned with the structure of data in terms of entities, relationships, and a set of integrity constraints. For example, primary keys and functional dependencies play crucial roles within databases. A database includes integrity restriction, which ensures that the stored data are consistent with the defined rules.

Each table in a relational schema is classified into one of three categories: entity, relation, and composite tables (Astrova & Kalja 2006). An entity table represents an entity in an entity-relationship (ER) diagram. A relation table represents a binary many-to-many relationship between two entities in an ER diagram without additional columns attached to the relationship. A composite table represents a binary many-to-many relationship between two entities in an ER diagram with additional columns attached to the relationship.

The semantics of a data model often represent an informal agreement between the designers and the users of the data model.

2.2 Ontology Description

Ontologies are derived from several relevant and precursory technological innovations, e.g., conceptual modeling languages, logics, theorem provers, deductive database, AI, and semantic nets. This makes evaluations with other technological innovations rather complicated, and therefore, ontologies are often preferred in a business setting.

Ontologies consist of interrelated concepts and rules that constrain and specify the intended meaning of the concepts. Ontology axioms represent the means to describe the equivalence of classes, disjunction of classes, or decomposition of classes into subclasses in addition to cardinality/type restrictions or domain/range constraints for classes or properties.

The purpose of an ontology is to clarify meaning or significance and support automated inference. The three most notable value propositions are (a) getting everyone to understand each other, possibly while disagreeing, (b) providing flexibility and agility; and (c) ensuring interoperability and integration.
Ontologies can be represented using RDF, RDFS or OWL. The Resource Description Framework (RDF) is a framework for expressing information about resources (documents, people, physical objects, and abstract concepts). RDF can be used to publish and interlink data on the Web. An RDF statement expresses a relationship between two resources. The subject and the object represent the two resources being related; the predicate represents the nature of their relationship. RDF Schema (RDFS) provides a data-modelling vocabulary for RDF data. OWL is a computational logic-based language such that knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit.

Statements in OWL can be divided in two groups: the TBox, consisting of intensional knowledge (axioms about classes and properties), and the ABox, consisting of extensional knowledge (statements including individuals) (Roussey et al. 2009).

Building ontologies is a challenging task because ontologies are supposed to hold application-independent domain knowledge. Indeed, ontologies include relatively general information that can be reused by different types of applications/tasks (Spyns et al. 2002).

3. Comparison between databases and ontologies

This section compares databases and ontologies on the basis of: engineering methodology, and role and usage.

3.1 Engineering methodology

This is a description of both artefacts (database and ontology) in terms of engineering methodology.

3.1.1 Database design

The database design process consists of the following steps:

- Analysis starts by considering the declaration of specifications and ends by generating a system specification. A specification is a formal representation of what a system should do, expressed independently of how it may be achieved.
- Design begins with a system specification, generates design documents, and provides in-depth information of how a system should be developed.
- Implementation is the development of a program according to the design document, considering the environment in which the system will be exploited. Implementation may be achieved iteratively, usually with prototypes that can be verified and tested before the final system is launched.
- Testing matches the functionalities of the implemented system with the design and specification document. This mapping facilitates the generation of an evaluation report so that the previous steps of the database design can be reviewed.

Maintenance is defined as the process of modifying a software system after delivery in order to correct faults, improve performance, or adapt to a changed environment. The solution over the lifetime of a maintained software system, i.e., the waterfall lifecycle model, will be repeatedly revisited.

Structured Query Language (SQL) is the query language used in most database management systems (DBMSs). Database query languages such as SQL aim to retain the reliability of data sets by using language constructs such as foreign keys.

Constraints in a conceptual schema ensure data integrity. They allow for the specification of semantics of data in the database. The following integrity constraints are rules that enable DBMSs to ensure that the semantics are satisfied by the data.

- Domain integrity: The domain restricts the values of attributes in the relation.
- Entity integrity: Every table has a primary key without NULL values.
- Referential integrity: A foreign key in a table matches with a primary key in another table. This maintains the correspondence between the rows of the two tables.

In addition, databases have useful engineering features such as concurrency control, transaction, crash recovery, and advanced storage techniques.
3.1.2 Ontology design

Reusing conceptual modeling methods and tools can be highly beneficial for ontology modeling. These techniques facilitate ontology adoption and comprehension (Jarrar et al. 2003).

The ontology building process is performed in the following phases (Zemmouchi-Ghomari & Ghomari 2013) (which are inspired by conceptual modeling methods):

- In the ontology specification phase, a collection of requirements that the ontology should fulfill is specified. The output of this activity is a document of requirements that includes the purpose, level of formality, and scope of the ontology, the target group and intended uses of the ontology, and a set of requirements that the ontology to be built should cover.

- In the ontology conceptualization phase, the domain knowledge is structured in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification activity.

- In the ontology formalization and implementation phase, the resulting basic taxonomic structure is enriched by axioms in formal ontology languages such as OWL. The domain rules must express the integrity of the data and the domain conceptualization. Consequently, the domain rules language should include constructs that express taxonomy and support inferencing (such as OWL). However, domain rules that are too specific might reduce the conceptualization genericity.

Simultaneously to the above-mentioned phases, knowledge acquisition, evaluation, and documentation tasks are carried out throughout the lifecycle of the ontology. In fact, unless the ontology developer is an expert in the application domain, most of the acquisition is performed simultaneously to the requirement specification phase and decreases as the ontology development process progresses.

Considering the crucial phase of ontology evaluation, Gomez-Perez (2004) introduced two terms: ontology verification and ontology validation. Ontology verification deals with building the ontology correctly, i.e., checking whether the design model represents the requirements appropriately. Ontology validation deals with checking whether the semantics model the real world for which the ontology was created.

3.2 Role and usage

This is a description of both artefacts (database and ontology) in terms of role and usage.

3.2.1 Database role and usage

Typical reasons for using databases are (German 2005) “efficient data access, data integrity, data security, data administration, concurrent access, crash recovery and reduced application development time.”

Existing DBMSs provide various functions that facilitate the management of a database and its data, which can be classified into four main functional groups:

- Definition: Creation, modification, and removal of definitions that define the organization of the data.
- Update: Insertion, modification, and deletion of the actual data.
- Retrieval: Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in basically the same form as that in which it is stored in the database or in a new form obtained by altering or combining existing data from the database.
- Administration: Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event, such as an unexpected system failure.

3.2.2 Ontology role and usage

Computational ontologies have numerous purposes, including interoperability, search, and software specification (Uschold & Gruninger 2004). Ontologies facilitate the integration of heterogeneous data sources by resolving semantic heterogeneity between them. Furthermore, ontologies are required for the prevention and resolution of communication issues between heterogeneous systems, knowledge
sharing, and information fusion. They facilitate information integration and interoperability between heterogeneous knowledge and information sources while maintaining a high level of abstraction.

Typical reasons for the development and use of ontologies (Noy & McGuinness 2001) are “to share common understanding of the information organization between human or artificial agents; to enable reuse of the domain knowledge; to make the domain assumptions; to separate the domain knowledge from the operational knowledge and to analyze the domain knowledge.”

The process of storing, sorting, and presenting media has many qualities that benefit from the use of Semantic Web technologies, such as ontologies, especially in the case of (a) unstructured information, (b) significant cross-document relationships or annotations that are difficult to manage using traditional relational databases, or (c) constantly changing usage patterns.

3.3 Summary table of main similarities and differences
Ontologies and databases have been compared above in order to establish their degree of similarity along with their advantages and disadvantages. Several aspects, such as efficiency issues and representation capabilities (Martinez-Cruz et al. 2012; Uschold 2010), have been considered. Table 1 summarizes some of the most relevant similarities and differences between ontologies and databases.

Tab 1. Summary of most relevant similarities and differences between ontologies and databases

<table>
<thead>
<tr>
<th>Perspective or Dimension</th>
<th>Database Schema</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differences</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus or core purpose(s)</td>
<td>Data</td>
<td>Meaning, shared understanding</td>
</tr>
<tr>
<td></td>
<td>Structure instances for efficient storage and querying</td>
<td>Human communication, interoperability, search, software engineering</td>
</tr>
<tr>
<td>Representation capabilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notation: syntax</td>
<td>ER diagrams; no standard serialization syntax</td>
<td>Logic; no standard diagram notation syntax</td>
</tr>
<tr>
<td>Notation: semantics</td>
<td>Minimal focus on formal semantics</td>
<td>Strong focus on formal semantics</td>
</tr>
<tr>
<td>Expressivity overlap</td>
<td>Entities</td>
<td>Classes</td>
</tr>
<tr>
<td></td>
<td>Attributes, relations: properties can be defined locally to an entity</td>
<td>Properties: properties are stand-alone entities that can exist without specific classes</td>
</tr>
<tr>
<td></td>
<td>Constraints: Schema behaves as constraints on structure of data; it defines legal database states</td>
<td>Axioms: ontology axioms behave like implications (inference rules); they entail implicit information</td>
</tr>
<tr>
<td></td>
<td>Instances (central): each instance has one entity as its type. Entities cannot share instances</td>
<td>Instances (optional): each individual can belong to multiple entities</td>
</tr>
<tr>
<td>Expressivity differences</td>
<td>No taxonomy</td>
<td>Taxonomy is the backbone (“is a” hierarchy)</td>
</tr>
<tr>
<td></td>
<td>Constraints for integrity, foreign key, delete</td>
<td>Constraints for meaning, consistency, and integrity</td>
</tr>
</tbody>
</table>
### Perspective or Dimension

<table>
<thead>
<tr>
<th>Perspective or Dimension</th>
<th>Database Schema</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlike conceptual modeling languages, knowledge representation languages do not differentiate between the meta-levels. For example, KR languages provide classes that can have instances, and those instances may also be classes. As a result, knowledge representation languages include modeling, meta-modeling, and meta-meta-modeling capabilities at the same time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard rules in natural language, little tool support</td>
<td>No standard rules or guidelines</td>
<td></td>
</tr>
<tr>
<td>Closed-world assumption (CWA): Missing information treated as false</td>
<td>Open-world assumption (OWA): Missing information treated unknown</td>
<td></td>
</tr>
<tr>
<td>Unique name assumption (UNA): Each individual has a single, unique name</td>
<td>No UNA: Individuals may have more than one name</td>
<td></td>
</tr>
</tbody>
</table>

### Efficiency issues

<table>
<thead>
<tr>
<th>Starting point</th>
<th>From scratch, rarely reuse.</th>
<th>Reuse if possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change management, agility, flexibility, evolution</td>
<td>Locked into specific set of queries per DB.</td>
<td>No query lock-in. Queries usable on other systems.</td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Loose coupling</td>
<td></td>
</tr>
<tr>
<td>Lost meaning</td>
<td>Semantics explicit</td>
<td></td>
</tr>
<tr>
<td>Difficult to evolve and maintain</td>
<td>Potentially easier to evolve and maintain</td>
<td></td>
</tr>
<tr>
<td>ETL tools to help</td>
<td>Few tools</td>
<td></td>
</tr>
<tr>
<td>Databases start with a conceptual model at design time, which is then transformed into an executable model</td>
<td>Classes, properties, rules, and axioms as well as instances of an ontology can be managed at run time of an application</td>
<td></td>
</tr>
<tr>
<td>An ontology (based on description logics) is a logical and dynamic model that can deduce new knowledge relations from stored ones, or check for their consistency</td>
<td>Conceptual models are static and explicitly specified at design time, but their semantic implications might be lost at implementation time</td>
<td></td>
</tr>
<tr>
<td>Perspective or Dimension</td>
<td>Database Schema</td>
<td>Ontology</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Processing engines</td>
<td>SQL engines</td>
<td>Theorem provers</td>
</tr>
<tr>
<td></td>
<td>Querying, views, data integrity</td>
<td>Infer new knowledge, ensure consistency</td>
</tr>
<tr>
<td></td>
<td>Eliminate redundancy to accomplish coherence when updating</td>
<td>Reasoning allows inferences (new knowledge), automatic classification of classes in a hierarchy, and data model validation (all definitions of classes are coherent)</td>
</tr>
<tr>
<td></td>
<td>Verify data coherence during insertion and update: integrity constraints</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>Highly tuned for performance and scale</td>
<td>Full inferencing: much smaller scale</td>
</tr>
<tr>
<td></td>
<td>Does not work well with too many joins</td>
<td>Reduced inferencing: attains large scale</td>
</tr>
<tr>
<td></td>
<td>Querying technologies and mechanisms are more mature in the context of relational databases than in the framework of ontologies. SPARQL is not more expressive than SQL.</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Data dictionary: separate artefact</td>
<td>Annotations (such as comments): part of the ontology</td>
</tr>
</tbody>
</table>

### Similarities

- Abstraction of the domain of discourse.
- Describing the structure of and constraints on data; consistency with schema constraints
- Instantiation of schema

#### 4. Conversion from one artefact to the other

Owing to their parallel features, a database can be converted into an ontology and vice versa. This type of transformation is possible when the information stored in the ontology corresponds to the data stored in the database. Vysniauskas and Nemuraite (1994) described three possible options for transferring data between these two domains: (1) using the same formalism to represent the ontology and the database, (2) generating a database schema for the ontology, and (3) obtaining a database from the ontology (with information loss).

This process highlights the correspondences between the elements of databases and ontologies, i.e., matching between relationships, classes and tables, attributes and features, and constraints and axioms (Sir et al. 2015).

#### 4.1 Database to ontology

Ontology development is much more time-consuming and expensive than building databases with regard to ontology complexity and formality. Hence, ontology reuse is recommended by default in current methodologies and guidelines as a key factor for developing cost-effective and high-quality ontologies (Zemmouchi-Ghomari & Ghomari 2013). The underlying principle is that reusing existing and already consensuated databases saves time and money in the ontology development process, and promotes the application of good practices.

One of the challenges of the Semantic Web is to integrate the information already available on the standard Web, usually stored in relational databases (Marzouk 2013).
Currently, the process of transforming databases into ontologies is very popular, and several methods are available to facilitate data conversion from a relational database to an ontology. Astrova (2005) termed this approach as ontology inverse engineering. Different mapping approaches can be used:

- Mapping creation: by an automatic (table-to-class) or manual/semi-automatic mapping representation
- Representation language and access mechanism: mapping implementation
- Static or dynamic: query implementation, SPARQL querying on RDF or SPARQL translation to SQL then querying on RDB
- Application domain: generic or domain-specific and data integration (the ability to integrate data from multiple sources)

Next, we will focus on the first method, i.e., mapping creation, and present an example approach.

Marzouk (2013) proposed a transformation process from an existing relational database schema defined by a set of tables into OWL ontologies using XSLT. A similar approach was proposed by Chujai et al. (2014), who demonstrated a stepwise transformation of an ER model into an OWL ontology using the Protégé editor tool. Marzouk’s approach consists of three steps.

- Mapping tables: Each entity table maps to an OWL class with the same name. Each composite table maps to an OWL class with the same name. Each relation table is not mapped to an OWL class, but to a pair of object properties, op1 and op2, each the inverse of the other.
- Mapping columns: For all tables, columns that are not foreign keys are mapped to data type properties. Foreign keys are ignored for a while, as they represent relationships.
- Mapping constraints: The following constraints specify whether a column in a table is unique, not NULL, a primary key, or a foreign key.
  - Unique column constraint: This constraint specifies that a column in a table is unique, meaning that no two data items in the table have the same value for the column. Therefore, the constraint maps to a functional property.
  - Not null constraint: This constraint specifies that a column in a table is not null, meaning that all the data items in the table contain values for the column. Therefore, the constraint maps to a minimum OWL cardinality of 1.
  - Primary key constraint: Because the value of a primary key can uniquely determine a single row of the table, the primary key constraint maps to OWL inverse functional properties.
  - Foreign key constraint: A foreign key of a table T1 establishes a relationship between T1 and another table T2. Therefore, it is mapped to an object property named T1-T2, with the OWL class corresponding to T1 as the domain class and the OWL class corresponding to T2 as the range class.

4.2 Ontology to database

- Given that Semantic Web queries are expected to increase in terms of number and complexity, the Semantic Web will need to (a) manipulate large amounts of existing data described by ontologies and the applications that operate on them and (b) assume maturity of relational databases to store and query such large amounts of data. Transforming ontologies into databases can provide a solution to this issue.
- Ontologies require a higher level of expressiveness than conceptual models. Several authors (Jean et al. 2006) have considered that conceptual models are not consensual and do not have shared knowledge representation qualities. Hence, ontologies cannot be regarded as conceptual models because they are reusable, whereas conceptual models are reusable at a lower level, at which a database conceptual model is generated from an ontology. The drawback of this approach is that it leads to the loss of important semantics in the translation process (Martinez-Cruz et al. 2012).
- Vysniauskas and Nemuraite (2006) proposed an approach that involves the stepwise transformation of an ontology into a relational database. First, the system transforms the ontology classes. The next step involves transformations of objects, data type properties, and constraints. Finally, the database is filled with instances of the classes.
In the first step, the ontology classes are transformed into relational database tables. The algorithm uses breadth-first search (it guarantees that, when some subclass is being created, its parent class in the hierarchy has already been created). In the relational database, a table is created for every class in the ontology with one-to-one relations between classes and their subclasses. This algorithm also uses the breadth-first search for object properties. First, it parses properties that do not have properties of the higher hierarchical level; then, it parses their sub-properties, and so on. Depending on the local cardinality of some class property, one-to-many or many-to-many relations between tables of classes are created. In the case of a many-to-many relation, an intermediate table is created. Data type properties are relations between instances of classes and RDF literals, and XML Schema data types. Next, a breadth-first search is performed for transforming ontology constraints into relational database metadata tables. In the last stage, i.e., transforming a domain ontology into a relational database, the transformation tool inserts all instances of classes into the created database.

As with many other similar transformation algorithms, this algorithm is restricted to transforming OWL Lite and part of the OWL DL syntax. It is not possible to transform all elements from a domain ontology into a conceptual data model in a straightforward manner because ontologies are semantically richer than data conceptual models (Benslimane et al. 2010).

4.3 Conversion tools

As mentioned earlier, several conversion approaches have been proposed, but few of them have been implemented as tools (Humaira et al. 2015). Most of the existing conversion tools concern the transition from database to ontology based on the mapping creation approach using the logical model of the database. Among the existing tools we can mention the following:

- D2R MAP (Bizer 2003): developed at the Free University of Berlin (Germany), this tool is also based on the method based on the logical model. An XML-based declarative language is used to describe the mapping rules for switching from a relational database to an ontology in RDFS. This language is intended for the enrichment of already existing ontologies from the source database, mapping its content to these ontologies. The main feature of this tool, apart from mapping tables and their attributes, is that it allows flexible mapping of complex relational structures by using SQL statements directly in the mapping rules.

- Relational.OWL (de Laborda & Conrad 2005): developed in 2004 at the German University of Heinrich Heine in Düsseldorf. The tool uses the method based on the logic model, it is also characterized by the use of the meta-modeling capabilities of OWL-Full, 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 different tables in the database are all mapped to instances in the ontology. What characterizes the mapping rules of this tool is the mapping of key attributes (primary and foreign) into data properties specific to the classes corresponding to their tables.

- Vis-A-Vis (Konstantinou et al. 2006): developed as a plug-in of Protégé (ontology editor) that allows to map relational databases to ontologies. The mapping is done manually by selecting from the database the dataset corresponding to a class of the ontology. An SQL query returns the desired dataset to add it to the class as new properties. This tool also performs a set of consistency checks to ensure validation of the mappings.

- DB2OWL (Cullot et al. 2007): It is a prototype that allows to create ontology from a relational database, programmed in java based on the Jena API for the construction of the ontology corresponding to the database source, the mapping is completely automatic as well as the export of database instances.

- OntER (Trinkunas & Vasilecas 2007): developed at the Vilnius Gediminas Technical University in Lithuania. The OntER tool is a Java-based plug-in for building a conceptual model of a relational database from an ontology as an intermediate model. OntER relies on Protege 3.3 and Sybase Power Designer 12.0, which is a design tool for building the conceptual data model of the database that corresponds to the source ontology.
• OntoRel (de Brum Saccol et al. 2011): an ontology is generated from an XML document with the help of the OntoGen semi-automatic tool, and then the OWL ontology is transformed into a relational model by applying a specific set of rules. This tool transforms only the main constituents of the ontology.

• OWLMap (Afzal et al. 2016): this tool is fully automatic 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.

5. Using an artefact in the context of the other artefact

In this section, we explain how and where ontologies are used for databases purposes and vice-versa.

5.1 Using ontologies for database purposes

Ontologies are important for databases. Because an ontology focuses on meaning, it can link applications that use different storage systems. A wide variety of knowledge-based and semantic technologies, including machine learning and natural language processing, depend critically on ontologies (Borgida 1995). Ontologies offer several advantages when used in conjunction with such technologies.

Database design is difficult (Sugumaran & Storey 2006) because “it relies heavily on a database designer understanding the users' database requirements and representing them so they accurately capture the real world application being modeled.” It is preferable to have application domain knowledge in the form of a domain ontology that provides knowledge of the relevant terms of a domain.

Beydoun et al. (2014) proposed the development of an ontology to formally represent IS requirements. Ontologies offer many benefits during the development of information systems, as they provide domain knowledge to requirement engineers. Similarly, they provide support in software extensibility, interoperability, and reuse.

Storey et al. (1997) explained that “the evolution of automated database design tools is to incorporate knowledge and reasoning capabilities.”

According to Spanos et al. (2012), ontologies are used for database purposes in the following cases:

1. Heterogeneous database integration: Ontology-based integration uses ontologies instead of conceptual schemas, and thus, mappings between databases and ontologies have to be established.
2. Ontology-based data access (OBDA): In this scenario, an ontology is linked to a source database and plays the role of an intermediate layer between the user and the data. An OBDA engine hides the data source from the upper levels by rewriting queries against a conceptual schema as queries against the local data source.
3. Mass generation of Semantic Web data: This is achieved by means of automatic extraction of the content of relational databases and its conversion into RDF, which facilitates the development of Semantic Web applications.
4. Ontology learning: Under the supervision of domain experts, extraction of information from relational databases, particularly from their schemas, contents, queries, and stored procedures, can facilitate the construction of rich ontologies (especially in the case of domains with no corresponding ontology).
5. Definition of the intended meaning of a relational schema: Changes are often made to the implemented relational database schema but not extended to the initial conceptual model. An ontology can play the role of an expressive conceptual model in accordance with the implemented relational database for maintenance purposes and integration with other data sources (Guanino & Welty 2002).
6. Integration of database content with other data sources: Conversion of a relational database into an RDF model facilitates easy integration of its content with information already represented in RDF.

Hsu et al. (2015) stated that an ontology has three major effects. (a) As part of the information extraction task, entities in the ontology provide synonyms for the named entity recognition component.
(b) The ontology maps to specific fields in the Web-based data entry form and provides constraints for data validation. (c) The content of the ontology is queried using SPARQL to retrieve information from the underlying data model based on semantic relationships.

5.2 Using databases for ontology purposes

Relational databases are considered as one of the most popular data storage solutions, and they have been generating data for Semantic Web applications. On the other hand, ontologies are considered as the backbone of the Semantic Web, as they are formal Web knowledge representations (Sir et al. 2015).

The role of databases in the Semantic Web has been explored from the viewpoint of gaining the experience and maturity of the database field. Meersman (2001) identified four success factors for the ontologization of an enterprise’s IT:

- Availability of lexical resources with ontological relevance
- Ability to convert the Web into the Semantic Web
- Methodological process to ontologize existing corporate knowledge
- Effective and efficient tools to design, align, and maintain ontological resources

A promising approach to enhance Semantic Web applications is to extract information from a functional relational database and transform it to facilitate its utilization in a Semantic Web perspective. The advantage of this approach is that the focus of efforts shifts from storage and querying of ontologies to issues such as database integration, ontology learning, mass generation of SW data, ontology-based data access, and semantic annotation of dynamic Web pages (Jarrar et al. 2003).

In the classical database architecture, tuples are organized according to a logical schema that has an abstract representation, i.e., the conceptual schema. At run time, developers write SQL queries to query the logical schema, and the conceptual schema is ignored. The Semantic Web community needs to emphasize querying for richer representation, i.e., querying of the conceptual schema (Fig. 1).

Constraints, expressed in formal logic, allow reasoning on the conceptual schema (the ontology) to infer new facts and make the schema richer. In addition, querying facilitates the inference of new facts. However, translating a query to the ontology vocabulary into a query to the data vocabulary is not trivial at all, especially when there is a mismatch between the ontology vocabulary and the logical schema vocabulary (Franconi 2008).

![Fig. 1 Ontology-based application (Franconi 2008)](image-url)
Database querying need to be reformulated in order to consider the logical specifications of RDF queries against plain data specifications. Moreover, constraints on queries add further complexity to the problem. For example, predefined semantics and blank nodes in RDF need to be handled by query processing, such as testing the entailment of databases and query conditions in order to ensure conciseness of RDF databases and query results (Gutiérrez et al. 2004). Curé (2005) presented a framework called DataBase Ontology Mapping (DBOM), which enables the end user to design, enrich, and maintain an ontology from an existing database.

The framework enables controls and maintenance operations from the database to the ontology (ontology updating) and from the ontology to the database (e.g., consistency checking).

This framework aims to synchronize the TBox and Abox states of the knowledge base with the actual domain state, i.e., to synchronize the ontology with the database.

The role of the DBOM framework is to develop an application-based ontology for inference purposes and bind the inference results to the database, thereby providing richer information to the end user.

6. Discussion

Whereas ontologies represent formal knowledge specifying a shared/agreed understanding of a given domain, databases describe specifications for the storage, retrieval, organization, and processing of data in information systems in a way that guarantees data integrity.

In the context of the Semantic Web, the content of databases is only shown when a query is performed in the database; a database tuple cannot be included in the database if it does not satisfy all the semantic constraints of the schema, e.g., the relational model constraints. Consequently, the closed-world assumption is applied to the relational database model, which represents a major semantic loss in the process of modeling information. On the other hand, ontologies use reasoners to solve these problems. These reasoners determine which instances belong to the ontology, depending on whether they fulfill all the constraints, e.g., whether they are coherent with the defined disjoint axioms, inheritance rules, etc. As a result, each time an instance check is required, the reasoner must be executed, in contrast to a database system, in which information integrity is always guaranteed.

Throughout the current survey, we can summarize the fundamental disparities between ontologies and databases as follows:

- The Semantic Web starts with the open-world assumption (OWA). In practice, this means that it is possible to incorporate new facts as needed but not necessarily anticipated.
- The difference is the generality of ontologies because they are usable and sharable at run time, whereas databases are exploited in the offline mode.
- Reasoning provides ontologies with knowledge that has not been explicitly defined in the specification phase. Moreover, other knowledge representation methods, such as those of Borgida (1995) or Reiter (1984), have attempted to enhance the modeling power of databases using Description Logic (DL) or First Order Logic (FOL). All these proposals have complemented the lack of relational databases, especially in semantic representation.

Semantic Web technologies including ontologies are more convenient solutions in the following cases:

- Constant changes in the managed data
- Constant changes in the required views of the data
- Cross-organizational collaboration with large volumes of data
- Schema is large and/or complex and/or used at query time (can use reasoning to structure and check schema and inferred answers and/or intensional queries)
- Not possible/reasonable to assume complete information (e.g., modeling complex structures or activities)

Several well-established organizations use ontologies. For example, the BBC, Times Inc., Elsevier, and the Library of Congress have production systems built using Semantic Web technologies. Similarly, Facebook has developed the Open Graph Protocol (similar to RDF). Microsoft, Google, and Yahoo use Schema.org (RDFa representation). The e-commerce industry uses GoodRelations. These frameworks are actively being used to provide users with a better Web experience.
Nevertheless, computational ontologies should focus on the real world instead of what can be represented. The aim of ontologies should be to represent the domain of interest, focusing on its invariant conditions. Databases can play the role of connectors between ontologies. Using ontologies as a conceptual view over data repositories is becoming increasingly popular, but for it to be standardized in applications, it is essential that the conceptual layer through which the data layer is accessed does not degrade data processing performance. Based on these observations, a family of DLs, called DL-Lite, has been proposed, tailored to capture basic ontologies and conceptual data modeling languages with low complexity of reasoning (w.r.t the size of data) and answering complex queries.

A large volume of Web data is stored in relational databases (Harris et al. 2017). In fact, almost all current websites store data in relational databases, and therefore developing ontologies from such databases is a big opportunity to support the development of the Semantic Web. When the ontologies are populated from the instances of available relational databases, this will enable ontologies to interact with this large volume of the existing data in the relational format.

On the other hand, the enrichment of databases is fundamental to maintain them, as well as the consistency and accuracy of the data (Pokorny 2010). The database becomes useless if it is not up to date. Fortunately, relational databases can evolve thanks to the reasoning abilities on the knowledge provided by ontologies (Zemmouchi-Ghomari et al. 2017). Therefore, reasoning and querying ontologies that are based on relational databases will make the semantic Web more useful. Even if, bridging the gap between ontologies and relational databases is still a research issue. Using relational databases for storing and processing ontologies with large datasets can be considered as a good solution to avoid the problems associated with scaling up.

7. Conclusion

Ontologies and databases are non-trivial conceptual representations that are useful and worth investigating. Databases are used for safely storing large amounts of data. On the other hand, ontologies find application in integration of semantic data or communication between heterogeneous systems, and they share the understanding of the structure of information between people or software. Unlike task-specific and implementation-oriented data models, ontologies, in principle and by definition, should be as generic and task-independent as possible.

These two artefacts are required by the Semantic Web as well as for the advancement of related research. Nonetheless, we believe that computational ontologies represent a step beyond conceptual schemas.

Future work will focus on carrying out an empirical evaluation of the usage of ontologies and databases in semantic web applications in order to have a realistic view of these paradigms usefulness in the context of the semantic web. The methodology could be based on the impact investigations techniques that explore the use of technologies in a specific context.

References


Astrova, I., 2005: Towards the Semantic Web-An Approach to Reverse Engineering of Relational Databases to Ontologies. ADBIS research communications, 152, Tallinn, Estonia


Borgida, A., 1995: Description Logics in Data Management, IEEE Transactions on Knowledge and Data Engineering, 7(5), pp. 671–682

Brodie, M. L., & Liu, J. T., 2001: The power and limits of relational technology in the age of information ecosystems. In Proceedings of Seminar on Digital Enterprise Research Institute, NUI, Galway Ireland


Curé, O., 2005: Mapping Databases to Ontologies to Design and Maintain Data in a Semantic Web Environment. Systemic, Cybernetics and Informatics, 4(4)

De Brum Saccol, D., de Campos Andrade, T. & Pivot, E. K., 2011: Mapping owl ontologies to relational schemata. In Information Reuse and Integration (IRI), Las Vegas, NV, USA


Harris, D. R., Henderson, D. W. & Talbert, J. C., 2017: Using closure tables to enable cross-querying of ontologies in database-driven applications. In proceedings of Biomedical & Health Informatics (BHI), Orlando, Florida, USA


Meersman, R., 2001: Ontologies and Databases: More than a Fleeting Resemblance, OES/SEO Workshop, Rome, Italy


Uschold, M., 2010: Semantic web: down to business, Taxonomy Boot Camp–Washington, USA


Vysniauskas E. & Nemuraite L., 2006: Transforming ontology representation from owl to relational database, information technology and control, 35(3), pp. 333-343

Wernher, B., 2005: Ambient Intelligence Semantic Web or Web 2.0. In: Proceedings of Semantic Content Engineering, Baden-Baden, Germany


**JEL Classification:** L86, M15