Journal of Big Data

RESEARCH Open Access

A novel approach for learning ontology from relational database: from the construction to the evaluation

Bilal Ben Mahria*, Ilham Chaker and Azeddine Zahi

*Correspondence:
bilal.benmahria@usmba.ac.ma
Faculty of Science
and Technology, 2202 Fez,
Morocco

Abstract

The aim of converting relational database into Ontology is to provide applications that are based on the semantic representation of the data. Whereas, representing the data using ontologies has shown to be a useful mechanism for managing and exchanging data. This is the reason why bridging the gap between relational databases and ontologies has attracted the interest of the ontology community from early years, and it is commonly referred to as the database-to-ontology mapping problem. In this paper, we: (1) propose a new life cycle for ontology learning from RDBs based on the software engineering requirements; (2) describe a new method for building ontology from Relational database based on the predefined life cycle; (3) add three new semantics that can be extracted from RDB; (4) we suggest an evaluation process based on two categories of metrics: (i) conceptual ontology (TBox) evaluation metrics; (ii) factual ontology (ABox) evaluation metrics.

Keywords: Ontology, Relational database, Tbox, Abox, Conceptual ontology, Factual ontology

Introduction

The benefits of using ontologies have been empirically grounded in several studies, among the most recent being the ones by [1–3]. According to Cardoso [3], for instance, ontologies are mostly used to make domain assumptions explicit (70%), to enable reuse of domain knowledge (56%), or to share a common understanding of the structure of information among people or software agents (37%) [4]. In other words, ontologies have gained tremendous momentum due to their great potential for providing a new approach for managing, searching, retrieving, maintaining, sharing and viewing information. They offer a best solution for resolving the heterogeneity problem that occurs between two or more information systems, by providing a generic knowledge that can be shared and reused by different kind of domains such as artificial intelligence, semantic web services, knowledge engineering and computer science [5]. As ontologies tend to evolve rapidly over time and between different applications, there is an increasing need in recent years towards their construction approaches.

Generally stated, building ontology is an engineering activity and there are two main approaches for its construction—either from scratch, or by using ontology learning approaches. Building ontology from scratch or manually [6–11] is a very complicated and expensive task that usually requires a combination of the knowledge of domain experts and skills of ontology engineers. This task is difficult due to the unbelievable rate of knowledge development in the real world, which requires the ontology engineers to constantly update and revise the resulting ontologies with new concepts, terms and lexicons. Consequently, building ontology from scratch is non-intuitive, time-consuming, error-prone, and can be costly [12]. Due to these limitations, the term "ontology learning" has appeared, which captures an approach to discover ontological knowledge automatically or semi-automatically from various resources [13]. Ontology learning can solve the problems of knowledge acquisition and greatly facilitates the building of ontologies compared with the scratching methods.

Formally, using learning approaches, ontologies can be constructed from various sources of information including structured sources, such as a relational database, semi-structured sources, such as dictionaries, or unstructured sources, such as web pages [14]. The majority of the studies in the literature focus on relational database as a source of information for several reasons. Firstly, around 70% of data on the web is stored in relational databases [15]. Secondly, relational databases present full conceptual models [16]. Thirdly, they provide a full information resource [16]. Finally, they offer one of the best techniques for storing and manipulating data. However, relational databases suffer from the absence of semantic meaning, which is hinders the ability to achieve interoperability among information systems [17].

Despite the significant progress made during the last few years and the wide number of proposed approaches [18–30], there are still many issues that have not been sufficiently addressed. First, all the existing works [18–30] focus only on generating A-Box or T-Box [31] and ignore the integration process between these two components. Second, the majority of these studies [18–30] mainly focused on the process of building ontologies from relational database without covering the maximum semantics resided in the database [32]. Furthermore, all these studies focus only on describing the process of generating ontology from RDB, while they did not define a life cycle for describing the most common scenarios that arise during the creation of the ontology from RDB [33]. Broadly stated, there is a difference between a lifecycle and process. Indeed, the need to the life cycles increases dramatically with the need to resolve the data integration problems and evaluation constraints [33].

Finally, the availability of ontology for different domains on the web is gradually increasing. Therefore, the resulting ontology from RDBs must be evaluated from different perspectives to determine its quality before use or reuse. All the existing works in this topic did not take into consideration the measurement of the quality of the resulting ontology [34].

In this paper, we: (1) propose a new life cycle for ontology learning from RDBs based on the software engineering requirements; (2) describe a new process for building ontology from Relational database based on the predefined life cycle; (3) add three new semantics that can be extracted from RDB; (4) we suggest an evaluation process

based on two categories of metrics: (i) Conceptual Ontology (T-Box) Evaluation metrics; (ii) Factual ontology(A-Box) evaluation metrics.

The rest of this paper is organized as follows. In "Related works" section, we present the related works, which describes the most popular studies about relational database into ontology conversion. In "Learning ontology from relational database (LOFRDB): life cycle" section, we introduce, the life cycle for learning ontologies from relational database. In "Proposed method" section, we introduce the proposed processes for generating ontologies from relational database. "Results and discussion" section is devoted to present the experimental results and discussions. Finally, we conclude the paper and suggests directions for future works.

Related works

Considerable amount of studies [18–30] have been conducted on building ontologies from RDBs using SQL-DDL [35]. While these studies share the common objective of converting RDB into Ontology, they differ in the process used as well as the metadata extracted and the mapping rules proposed. In fact, these studies fall roughly into one of the two categories. Firstly, approaches based on an analysis of relational schema. Secondly, approaches based on analysis relational data.

On one hand, all methods described in [18–30] take into account the mapping of: tables, columns, primary keys and foreign keys. However, the binary relationship is missed in [24, 25], and the ternary relationship is not manipulated in [21, 24, 25, 28, 30]. Only [22, 26, 29] covered the check constraint, Not Null constraint, and unique, while added the cardinality constraint. Moreover, Astrova [22] represents the only work that can handle the transitive and symmetric property, while [29] handled just the transitive property. In addition [29], presents the most reference work in the literature, because it consists of combining the existing studies and adds new rules for building ontology from RDB. Besides, Sequeda [29] covers all possible combinations of primary key and foreign keys as depicted in Table 2. Clearly, the two studies provided by Astrova [22] and Sequeda [29] represent the most relevant ones because they proposed many requirement that can act as best practices for building ontologies from RDBs. On other hand, building an ontology based on an analysis of relational data (Migration of the instances) is addressed in [21, 22, 28, 29].

However, all these studies ignore constraints that capture additional semantics in order to improve the quality of the resulting T-Box [31], such as owl: hasvalue constraint, data range restriction, and owl: all values from constraint [36]. In addition, all these works did not take into consideration the phase of integrating the A-Box with T-Box. In fact, the combination of TBox and ABox has two main benefits; (a) it facilitates the Semantic integration problem; (b) it allows to use a reasoning services for checking the consistency and *satisfiablility* of the resulting ontology [37]. To the extent of our knowledge, this is the first work that integrates the A-Box with T-Box in addition to use the reasoning capabilities for checking the consistency and *satisfiablility* [37]. These approaches allowed a mapping of RDB models into Ontology, they [18–30] focused only on describing the process of building the ontology, whilst they did not describe the life cycle. From software engineering perspective, the ontology development process identifies which activities are to be performed.

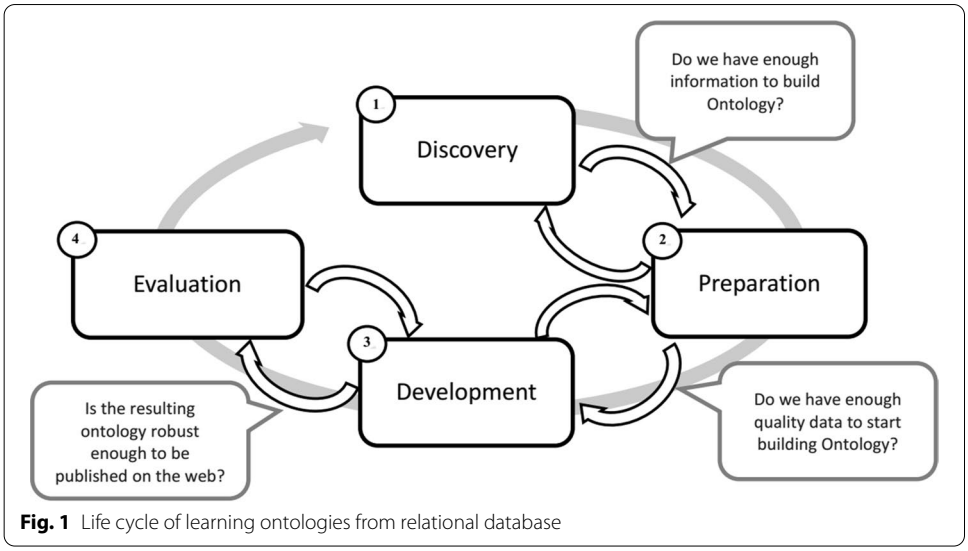

Fig. 1 Life cycle of learning ontologies from relational database

However, it does not identify the order in which the activities should be performed. Whereas the life cycle identifies when the activities should be carried out, it determined the global stages through which the ontology moves during its life time and it describes what activities are to be performed in each stage and how the stages are related.

Eventually, the ontology evaluation becomes extremely important for developers to determine the fundamental characteristics of ontologies in order to improve the quality, estimate cost and reduce future maintenance [38]. To the best of our knowledge, there are only a few papers [22, 29] have appeared with the concern of evaluating not the resulting ontology but the mapping process. Astrova [22] proposed a method for measuring the quality of the mapping process RDB to Ontology based on retransforming the resulting ontology to a relational database and testing if the transformation is reversible using the lexical overlap measure. Sequeda [29] introduced an effective approach for validating the mapping process with regard to four properties. Nevertheless, those studies mainly focused only on the validation of the mapping process and not on the quality of the resulting ontology. In fact, learning ontologies from relational databases without considering the evaluation phase means that the resulting ontology does not cover the user or the domain needs [39].

Learning ontology from relational database (LOFRDB): life cycle

In this section, we present LOFRB lifecycle, which refers to the activities or phases that have to be performed for learning ontologies from relational databases. As depicted in Fig. 1, our proposed life cycle is based on four phases: Discovery, Preparation, Development, and evaluation. For most phases in the life cycle, the movement can be either forward or backward. This iterative depiction is intended to more closely portray a real project [40], in which aspects of the project move forward and may return to earlier stages as new information is uncovered and ontologist learns more about the domain of interest [5].

Discovery

In this stage, the ontologist must define clearly the domain and scope of the ontology by answering the following questions [10]:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology (Application)?
- For what types of questions the information in the ontology should provide answers?
- What are the ontology intended uses and who are the end-users (Stakeholders)?
- What are the sources of RDBs used to build ontology?
- Is it necessary to interviewing the domain expert?

The answers to these questions may change during the ontology development, but at any given time they help to limit the scope of the model. In this stage also the ontologist formulates some competency questions (CQ) that the ontology should be able to answer and that can be tested later [41]. The aim of the CQ is to check if the ontology includes sufficient information to answer these questions and if the answers require a particular level of detail or representation of a particular area. These CQs are just a sketch and do not need to be exhaustive [41].

As part of the discovery phase, the ontologist needs to assess the resources available to support the ontology development process. In this context, resources contain technology, tools, data, and people [40]. In addition, the ontologist can remove or add the data sources from this phase [40].

Preparation

The second phase of the LOFRDB involves data preparation, which includes the steps to explore and preprocess (conditioning) data prior. The data exploration consists of checking if the data sources contain enough semantics for generating ontology by checking if the RDB contains the complete space of relations and the maximum possible combinations of the primary keys and foreign keys [29]. The second sub-phase is data conditioning, which refers to the process of cleaning data and normalizing datasets. We can consider the RDB normalization as a part of the data conditioning phase.

Development

The Development (the building of the ontology) is tackled in two phases: the pre-development and post-development. The pre-development starts by the Data acquisition (ABox), which consists of extracting the instances from the relational database [42], and represent them based on the RDF triple form [43]. After the data acquisition, the schema acquisition (TBox) [22] will be started in order to generate the definition and the meaning of the extracting instances. Therefore, it is necessary to build a vocabulary of these terms for simplifying the development phase. The Development does not only include the data and schema acquisition, but provides also the phase for integrating these two components. The post-development encompasses several other tasks such as alignment, merging and integration, etc. [5].

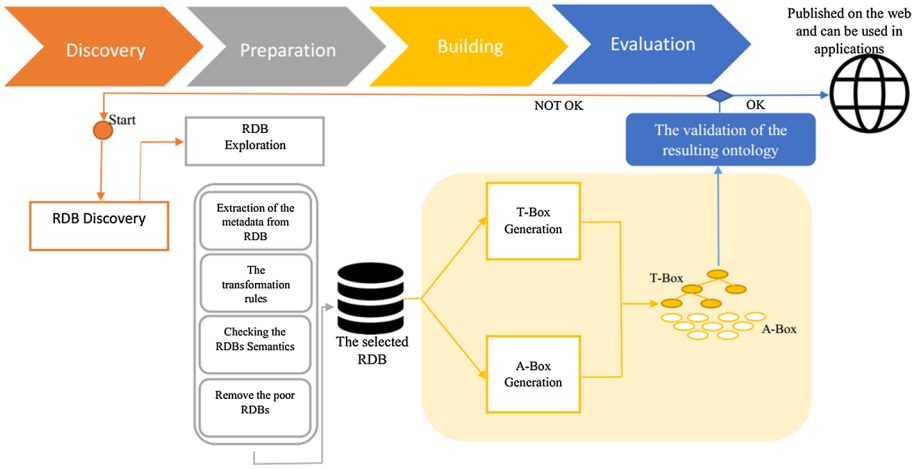

Fig. 2 The method of building the ontology from RDB

Evaluation

After having built ontology from RDB, metrics for evaluating the resulting ontology must be presented [44]. Generally, the process of evaluation can be defined as the process of deciding on the quality of the ontology with respect to particular metrics [44]. For this purpose, two orthogonal dimensions to evaluate the quality of the resulting ontology are defined; (i) the first dimension is T-Box evaluation; (ii) the second dimension is A-Box evaluation. T-Box Evaluation postulates the design of the constructed T-Box. Although we cannot definitely know if the T-Box design correctly models the domain knowledge, metrics such as the richness, and inheritance indicate the quality of the T-Box created. The most significant metrics in this category are described in [45].

Proposed method

From the proposed lifecycle, many processes or models can be extracted and this is depends on the needs of the ontologist and the objectives of the project. In this work, we propose a method for ontology learning from RDB based on our proposed lifecycle. In this method, we consider that the data is already cleaned and conditioned. In addition, the resulting ontology needs neither alignment nor fusion with other ontologies.

As depicted in Fig. 2, after the discovery phase, which aims to identify the domain and scope of the ontology as well as take a first look at the data sources, the next phase is the data preparation. In this phase, some semantic characteristics are extracted and we use a novel metric to choose the RDB the most relevant. From this last one, we generate the ABox and the TBox [37] and then after we integrate the two component to get the final ontology. The last phase of our process is the validation of the resulting ontology that consists of evaluating the ABox and the TBox components by using some metrics and a reference ontology, and finally verify if the resulting ontology can response to the Competency Questions (CQ) [41]. If the validation [46] is failed, this means that the resulting ontology cannot be published on the web or used inside applications. In this case, it is necessary to return to the discovery phase.

Table 1 Summary of patterns to calculate NS

Patterns	Acronym
Table without FKs	NTDFK
Table with one FK	NTFK
Table with more than 2 FKs	NTMTWOFK
Tables that contain exactly 2 foreign keys with presence of independent attributes	NTEXTWOFK
Attributes that are FK + NULL + Not UNIQUE	NAFKNNU
Attributes that are FK + NOT NULL + NOT UNIQUE	NAFKNNNU
Attributes that are FK + NOT NULL + UNIQUE	NAFKNNU
Attributes that are FK + NOT NULL + UNIQUE + NOT PK (FK is not equal to the PK)	NANNUNPK
Attribute (neither PK nor FK)	NA
Attribute + NOT NULL	NAN
Attribute NOT FK + UNIQUE	NANFKU
PK	NPK
Attribute with constraint with an integer greater than 0	NACHECK
CHECK constraint with enumeration	
CHECK constraint	
Attribute with Default Value constraint	NADef
Tables with tables share the same primary key	NTSAMEPK
FK is a reference to the same table	NUnaryRel
FK that is a reference to the same table, but it is accompanied by a trigger ON DELETE CASCADE	NTrRel

RDB exploration

The exploration phase consists in verifying if the input relational databases contain the complete space of metadata and semantic characteristics for generating ontology. In this context, some information can be extracted from the input RDBs like the number of: tables, columns, primary keys, foreign keys and instances. On the other hand, we consider the semantic characteristics summarized in Table 1 to choose the most relevant RDB.

In this context, we suggest the number of semantics (NS) metric that represents the number of semantic characteristics of each input RDB. The range of this metric is from 0 to 17. Values close to 0 reflects a relational database that semantically poor, while large values, that are close to 17, represent a rich RDB. The NS metric is calculated by giving the value "1" to each characteristic existing in the RDB and "0" otherwise:

$$\begin{aligned} NS = {} & NTDFK + NTFK + NTMTWOFK + NTEXTWOFK + \text{NAFKNNU} \\ & + NAFKNNNU + NAFKNNU + NANNUNPK \\ & + NA + NAN + NANFKU + NPK + NACHECK \\ & + NADEF + NTSAMEPK + NUnaryRel + NtrRel. \end{aligned}$$

The RDB exploration needs also the human intervention for selecting the relevant relational database because the database that have high total number of semantics does not mean that it covers all the possible semantics [47].

Table 2 The applied rules for generating conceptual ontology (TBox)

Patterns	Kind of patterns	OWL corresponding element
Table patterns	Table without FK	OWL: class
	Table with one FK	
	Table with more than 2 FKs	
	Tables that contain exactly 2 foreign keys with presence of independent attributes	
Binary Relationship table	Tables that contain exactly 2 foreign keys without presence of independent attributes	We create two object properties (owl:objectProperty) The latter is an inverse of the former
Tables with one FK	Attributes that are FK + NULL + Not UNIQUE	Object Property + Functional Property + Min Cardinality of the inverse property = 1
	Attributes that are FK + NOT NULL + NOT UNIQUE	Object Property + Card = 1 + Min Cardinality of the inverse property = 1
	Attributes that are FK + NOT NULL + UNIQUE	Object Property + Functional Property + Functional Property for the inverse Property
	Attributes that are FK + NOT NULL + UNIQUE + NOT PK (FK is not equal to the PK)	Object Property + Functional Property + Card = 1 + Functional Property for the inverse Property
Attributes	Attribute (neither PK nor FK)	DatatypeProperty
	Attribute + NOT NULL	DatatypeProperty + MinCardinality = 1
	Attributes NOT FK + UNIQUE	DatatypeProperty + MaxCardinality = 1
	Primary Key	MinCardinality + MaxCardinality = 1 (Cardinality = 1)
Check Constraint	Attribute with constraint with an integer greater than 0	xsd:positiveInteger
	CHECK with enumeration	xsd:positiveInteger
	CHECK constraint as Value Restriction	Xsd:minInclusive, Xsd:maxInclusive, Xsd:minExclusive, Xsd:maxExclusive
Default constraint	Attribute with Default Value	Owl:hasValue
Inheritance relationship	Two tables share the same primary key	rdfs:subClassOf
Symmetric Relationship	FK is a reference to the same table	owl: SymmetricProperty
Transitive Relationship	FK is a reference to the same table, but now it is accompanied by a trigger ON DELETE CASCADE	OWL:TransitiveProperty
Inheritance relationship improvement	The range of the foreign Key attribute	Owl:AllValuesFrom

Building the TBox (conceptual ontology)

The TBox introduces the vocabulary of an application domain. It represents the repository that contains the declarations of concept axioms or roles [48]. To generate the TBox from RDB, we use some transformation patterns that are defined in Table 2. Concisely, the main steps for generating conceptual ontology is depicted in Algorithm 1.

In this step, we propose 3 new transformation rules which allow to transform: the check constraint, the default constraint, and the constraint for improving inheritance relationship.

SQL DDL	OWL
CREATE TABLE Persons (Age int, CHECK (Age>=18));	<rdfs:Datatype rdf:about="&example;PersonAge"> <owl:withRestrictions rdf:parseType="Collection"> <rdf:Description> <xsd:minInclusive rdf:datatype="&xsd;integer">18</xsd:minInclusive> </rdf:Description> </owl:withRestrictions> </rdfs:Datatype>

Fig. 3 Check constraint for data range restriction

SQL DDL	OWL
CREATE TABLE Persons (City varchar(255) DEFAULT 'FEZ');	<owl:Restriction> <owl:onProperty rdf:resource="#hasCity"/> <owl:hasValue rdf:datatype="&xsd;String">FEZ </owl:hasValue> </owl:Restriction>

Fig. 4 Default Value constraint transformatio

SQL DDL	OWL
CREATE TABLE Person (Pid integer primary key, Name varchar(15) Not Null); CREATE TABLE Professor (id integer primary key, Title varchar(15), CONSTRAINT FK_Person FOREIGN KEY (id) REFERENCES Employee (pid));); CREATE TABLE Student (id integer primary key, Degree varchar(15), CONSTRAINT FK_Person FOREIGN KEY (id) REFERENCES Employee (pid)););	<owl:ObjectProperty rdf:ID="hasProfessor"> <rdfs:domain rdf:resource="#Student"/> </owl:ObjectProperty> <owl:class rdf:about="#Student"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="hasProfessor"/> <owl:AllValuesFrom rdf:resource="#Professor"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

Fig. 5 Improvement of the inheritance relationship example

Transformation of the check constraint

As mentioned in [49], Check constraints are conditions that validates the data in a table. In this work, we propose a rule for transforming the CHECK constraint as data range restriction. For resolving this problem, we used the bounds facets, which are: xsd:minInclusive, xsd:minExclusive, xsd:maxInclusive, and xsd:maxExclusive [36] (see Fig. 3).

Transformation of the default constraint.

The DEFAULT constraint in RDB [50] is used to provide a default value for a column. In this respect, the owl: hasValue constraint describes a class of all individuals for which the property concerned has at least one value semantically equal to the default value. Consequently, owl: hasValue says regardless of how many values a class has for a particular property, at least one of them must be equal to the default value [36]. Figure 4 depicts the transformation of the default constraint to OWL.

Improvement of the inheritance relationship

It is important to realize that in OWL domains and ranges should not be viewed as constraints to be checked [36]. They are used as 'axioms' in reasoning. For instance, if the property hasProfessor has the range set as Professor and the domain set as Student, then we applied the hasProfessor property to Student (instances that are members of the class Student), this would generally not result in an error. Knowing that Student and Professor are subclasses of Person. In this context, it would infer that Student and Professor Classes can have instances in common. More precisely, it can be found that "Student hasProfessor Student". As a result, we will use the owl: AllValuesFrom constraint [36] for avoiding such problem as depicted in Fig. 5.

The generation of the TBox

The TBox introduces the terminology and the vocabulary of application domain. It represents the repository that contains the declaration of concept axioms or roles. A naïve approach would consider that the TBox corresponds to the schema of the Relational Database [31]. In this phase, we implement the rules that are identified in Table 2. Concisely, the main steps for generating the TBox is depicted in Algorithm 1.

```
Algorithm 1: a Sketch of the T-Box Generation
Input: SQL DDL File
Output: OWL File
Var:
       manager: OWLOntologyManager
       arrTables: ArrayList<String>
       cp: ArrayList<TableComponent>
Begin
1: arrTables←getTables()
2: for each i<arrTables.size()start
3:   cp←getColumnsOfSpecificFieldTable(arrTables.get(i))
4:      if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()="YES"&&
NotUnique)
5:         createOWLDataProperty();
6:      end if
7:      if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()="NO"&&
NotUnique)
8:         createOWLDataPropertyWithMinCadrinalityEqualOne();
9:       end if
10:     if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()="YES"&&
Unique)
11:        createOWLDataPropertyWithMaxCadrinalityEqualOne();
12:     end if
13:     if(isNotCompositePrimaryKey(arrTables.get(i)))
14:        if(isNotPrimaryKey(cp.getColName(), arrTables.get(j))
15:           createOWLDataPropertyWithExactCadrinalityEqualOne();
16:        end if
17:     end if
18:     if(isNotCompositePrimaryKey(arrTables.get(i)))
19:        if(isNotPrimaryKey(cp.getColName())&&isFkeys(cp.getColName()))
20:           objectProperty←createObjectProperty(parentClass, childClass)
21:        end if
22:     end if
23:     if(isPrimaryKeyComposite(arrTables.get(i))&&getNumberOfPKeys()>getNumberOfFKs())
24:        p←listAllPrimaryKeysOfSpecifiedTable(arrTables.get(i))
25:        for each k<p.size() start
26:           if(isForeignKey(arrTables.get(i)),cp.getColName())
27:              objectProperty←createObjectProperty(parentClass, childClass)
28:           else
29:              createOWLDataPropertyWithMinCadrinalityEqualOne()
30:        end for
31:     end if
32: end
```

The automated process of our algorithm receives as input the SQL DLL file [51] that contained the definition of the RDB and generates the OWL file as output. More

precisely, the Algorithm 1 gets all RDB patterns depicted in Table 2 then it matches each RDB element with its corresponded element in OWL. It is important to mention that our algorithm is completely automatic. The implementation if this algorithm is uploaded into our GitHub repository.

The generation of the ABOX

The process of generating the A-Box is conducted using the R2RML language [52] that plays an important role for completing the data acquisition phase. Generally, the algorithm receives a SQL file that includes statement represented by SQL DDL. We then use the Database Metadata Extraction Engine (DMEE) that analyzes the SQL file and extracts automatically the metadata from it. The extracted metadata includes tables, columns, primary keys (PKs), and Foreign Keys (FKs). Thirdly, Mapping Generator Engine (MGE) exploits the extracted metadata and build a mapping file (R2RML file). Lastly, R2RML engine takes as input, the database model (Schema + Instances) and the generated mapping document that contains a set of rules representing the database schema, then provides an output represents the RDF dataset (triples) using r2rml-kit-master.[1] Concisely, the main steps for generating the A-Box is depicted in the following algorithm.[2] For convenience to the readers, the algorithms of generating the A-Box are deeply explained in [53]

Algorithm 2: The sketch algorithm of generating the A-Box

```
Input  : metadata;
output: R2RML Mapping file;
var:
      tmap: TriplesMap;
      collectionMap: Collection<TriplesMap>;
      metadata: DButils;
      table_names: ArrayList<String>;
      colInfo: Map<String, List<TableComponent>>
  Begin
1:    table_names←metadata.getTablesInfo();
2:    for each i<table_names.size();
3:       logicalTable←mf.createTableOrView()
4:       colInfo←data.getColumnInfo(tbl_names.get(i))
5:       for each key in colInfo.KeySet() do
6:          for each k < colInfo.get(key).size()
7:            if(metadata.isPrimarykey()) then
8:               templet←mf.createTemplate()
9:               subjetcMap←mf.createSubjectMap(template)
10:           else
11:             pm←mf.createPredicateMap("ex:"+colInfo.getColName);
12:             objectMap←mf.createObjectMap(colInfo.getColumnName());
13:             prdicateObjectMap←mf.createPredicObjectMap(pm,oMap);
14:             triplesMap←mf.createTriplesMap(ltable,sMap,pOMap);
15:           if(metadata.isForeignKey())
16:             tmap←getRefObjetMap(getPtb, getPtC,getCht(),getChC());
17:    triplesMap.setResource(createResource(url));
18:    collectionMap.add(triplesMap);
19:    R2RMLFileMapping←write(collectionMap)
20:  End
```

.

The evaluation

The last step of our process involves validation of the resulting ontology. For this purpose, we propose to evaluate the ABox component and the TBox component separately by using some metrics. In this context, we have choose, the attributer richness,

[1] https://github.com/d2rq/r2rml-kit.

[2] https://github.com/bilalbenma.

Inheritance Richness and Relationship Richness to evaluate the TBox component, and Class Richness as well as Average Population to evaluate the ABox [54].

The evaluation of TBox

Although we cannot really know whether the design of the T-Box correctly models the domain knowledge, metrics such as wealth, width, depth and heritage indicate the quality of the T-Box created. Therefore, the most important measures in this category are described below.

Attribute richness (AR) AR represents the average number of attributes (slots) per class. Generally, we assume that more the attributes are generated from RDB more the knowledge conveys to the ontology [44].

Definition The attribute richness is defined as the average number of attributes per class. It is calculated as the number of attributes for all classes (ATT) divided by the number of classes (C).

$$AR = \frac{|ATT|}{|C|}$$

Inheritance richness (IR) This metric represents the distribution of information across different levels of T-BOX and serves as an indicator of how well knowledge is grouped into different categories and subcategories in TBox. A TBox with a low IR indicates that the T-Box covers a specific domain in a detailed manner, while a T-Box with a high IR represent a general knowledge [44].

Definition IR is defined as the average number of subclasses per class, where H is the sum of the number of inheritance relationships, and C is the total number of classes.

$$IR = \frac{|H|}{|C|}$$

Relationship richness (RR) This metric reflects the diversity of the types of relations in the TBox such as. A TBox that contains only inheritance relationship usually conveys less information than a T-Box that contains a diverse set of relationships such as Transitive, symmetric, and reflexive relationship [45].

Definition The RR of a T-Box is defined as the ratio of the number of non-inheritance relationships (P), divided by the sum of inheritance relationships (H) and non-inheritance relationships (P).

$$RR = \frac{|P|}{|H| + |P|}$$

Table 3 A list of metadata extracted from RDB

RDB	Tables	Columns	PKs	FKs	Instances
North	30	150	30	25	5029
Iscommerce	5	20	5	6	200
Ecommerce	25	100	25	17	5000
EcommerceDB	3	20	4	2	1000
Sakila	16	90	18	22	47,237
Northwind	13	89	16	13	2110

A-Box validation

A-Box evaluation metrics can be used to check how the data is placed inside the ontology. More specifically, A-Box evaluation refers to the instances metrics. In this respect, we used two predefined metrics: class richness and average population.

Class richness (CR) CR is related to how instances are distributed across classes. The number of classes that have instances in the KB is compared with the total number of classes, giving a general idea of how well the KB utilizes the knowledge modeled by the T-Box. A-Box with low CR indicates that the A-Box does not have data that exemplifies all the class knowledge exist in the T-Box. On the other hand, A-Box with high CR proves that the data in A-Box covers most of the knowledge [44].

Definition *CR*is defined as the ratio between the total number of classes that have instances c' divided by the total number of classes (C).

$$CR = \frac{|c^{'}|}{|C|}$$

Average population (AP) This measure is an indication of the number of instances compared to the number of classes. It can be useful if the ontology developer is not sure if enough instances were extracted compared to the number of classes [44].

Definition *AP*is defined as the number of instances in the A-Box (I) divided by the number of classes defined in the ontology schema (C).

$$AP = \frac{|I|}{|C|}$$

Results and discussion

To evaluate the efficiency and the solidity of the proposed process, we have started from 6 relational databases of the e-commerce domain. These databases cover several metadata used in the process of learning ontologies from relational database, such as tables, columns, foreign keys (FKs) and primary keys (PKs). The detailed information of these databases is summarized in Table 3. As proof of concept, our experimental simulations were conducted on a personal computer under windows 10, with Intel core i7 2.70 GHZ processor and 16 GB RAM.

Table 4 The list of questions the ontologist must answer before start building ontology

What is the domain that the ontology will cover?	E-Commerce domain
For what we are going to use the ontology (Application)?	Describing businesses, offering, prices, features, payments options, opening hours, and so on
For what types of questions the information in the ontology should provide answers	Table 5 depicts some competency questions for validating the resulting ontology
What are the ontology intended uses and who are the end-users (Stakeholders)?	Customers, Employee, web Master, Accounting, etc
What are the characteristics of the selected RDBs	Table 3 presents all the necessary metadata that we need to figure out the characteristics of the selected RDBs
Is It necessary to interview the domain expert?	No

Table 5 A list of competency questions

Query 1: find movie for a given set of generic features such as name and duration, etc
Query 2: retrieve basic information about a specific movie for display purposes
Query 3: find movie having a label that contains specific words
Query 4: get information about a reviewer
Query 5: find movies having a label that contains specific words
Query6: find Text description of a given movie's title
Query 7: find movies that are similar to a given movies

The discovery phase

In the discovery phase, we have to answer the following question: do we have enough information background to start building ontology? Table 1 shows the most relevant questions that we have covered. It may be possible to refer to an expert in the studied domain to resolve some problems concerning the gathered data such as the database conceptualization problems [47].

Unlike many traditional stage-gate processes, in which the process of building ontology from relational databases start without checking if some specific criteria are met. Therefore, the proposed lifecycle is intended to accommodate more ambiguity. As depicted in the Table 4, it is recommended to pass certain checkpoints as a way of gauging whether we are ready to move to the next phase of the LOFRDB lifecycle. Creating the perfect plan for learning ontology from RDB requires a clear understanding of the domain area, the problem to be solved, and scoping of the data sources to be used. Answering these questions clarify the problem definition and help us to select the appropriate database that can be used in later phases. The Table 5 exhibits a list of competency questions (CQs) that represent informal questions that the ontology must be able to answer [41]. We consider these to be natural language sentences that express patterns for types of question people want to be able to answer with the ontology.

As we know, ontology authors are usually domain experts but not necessarily proficient in ontology technologies, especially their logic underpinnings [41]. As a consequence, on the one hand it is difficult for human authors to express their requirements for the axiomatization of an ontology and, on the other hand, it is also difficult to know whether the requirements are fulfilled as a result of their ontology authoring actions. To address this issue, we introduce the methodology of Competency Question in order to help the authors of the ontology to check if the resulting ontology embedded all the

Table 6 The set of patterns

Rules	North	Ecommerce	Sakila	Northwind
Tables Without FKs	√	√	√	√
Tables With one FK	√	√	√	√
Tables with more than 2 FKs	√	√	√	√
Tables that contain exactly 2 FKs with presence of independent attribute	√	√	√	√
Many-to-many relationship: a table that contains exactly two FKs	√	√	√	√
FK + NULL + Not UNIQUE	√	√	√	√
FK + NOT NULL + NOT UNIQUE	√	√	√	√
FK + NOT NULL + UNIQUE			√	√
FK + NOT NULL + UNIQUE + NOT PK			√	√
Attribute (neither PK nor FK	√	√	√	√
Attribute + NOT NULL	√	√	√	√
NOT FK + UNIQUE	√	√	√	√
PK	√	√	√	√
Attribute with constraint with an integer greater than 0			√	√
CHECK with enumeration			√	√
CHECK constraint as DataTypeRestriction			√	
The range of the foreign Key	√	√	√	√
Default Value			√	√
Two tables share the same primary key			√	
FK is a reference to the same table			√	
FK is a reference to the same table, but now it is accompanied by a trigger ON DELETE CASCADE			√	

necessary information. In fact, it is important to list these questions in the discovery phase in order to allow to the ontologist to take them into consideration during the process of development.

Additionally, in the discovery phase, we can build an initial look at the list of data that we have chosen in order to determine whether it contains a large number of necessary metadata. It can be clearly seen from Table 3, that, the relational database EcommerceDB and Iscommerce did not contain sufficient semantics to start building ontology. For instance, EcommerceDB database contains 3 tables, 20 columns, 4 PKs, 2 FKs, and 100 instances. Based on these measures, we can decide that the EcommerceDB database is semantically poor. In the same context, the Iscommerce database also provides a poor semantics. As a result, in the discovery phase, we can remove the EcommerceDB and Iscommerce databases. We eliminate these two databases based on the rule: RDB poor semantically implies ontology poor semantically [31].

The RDB exploration

Now, to choose the most relevant RDB among the remaining ones, we have to calculate the NS measure from the patterns depicted in Table 6.

As stated previously, the NS metric represents the number of semantic characteristics present in the relational database. Table 6 shows that the Sakila database covers all possible semantics that can be used to build a rich ontology from RDB. In this context, we compare also the total number of semantics per RDB as shown in Table 7. The first interesting observation is that the database having a high total number of semantics does not

Table 7 The NS and the total number of semantics for each database

	North	E-commerce	Sakila	Northwind
NTDFK	5	12	5	4
NTFK	10	6	10	3
NTMTWOFK	8	7	6	3
NTEXTWOFK	7	4	7	3
NAFKNNU	15	10	7	5
NAFKNNNU	10	7	7	4
NAFKNNU	0	0	6	0
DefVal	0	0	3	0
NANNUNPK	0	0	2	0
NA	50	23	24	42
NAN	30	25	8	15
NANFKU	15	10	10	4
NPK	30	25	18	16
NACHECK	0	0	5	0
NADef	0	0	3	3
NTSAMEPK	0	0	2	0
NUnaryRel	0	0	2	0
NTrRel	0	0	1	0
Number of semantics (NS)	10	10	17	13
Total number of semantics	180	129	126	102

mean that it covers all the possible semantics as depicted in Table 7. For instance, the total number of semantics for the North database is 180, but the number of semantics is 10 (less than 17). Consequently, if we decided to build ontology from the North and e-commerce databases, the resulting ontology will not address the following semantics: inheritance, transitive, symmetric, value restriction, data range restriction, Functional and inverse Functional property. This leads to predict that the resulting ontology based on the North and E-commerce databases will be very poor semantically.

For the Northwind and Sakila database, the NS and the total number of semantics are (13,102) and (17,126) is 13 and its total number of semantics is 102. We can notice that, the number of instance of each database are respectively 2120 and 47,237 for Northwind and Sakila. As a result, the most appropriate relational database for building ontology is Sakila, because it covers the most important semantics and the large number of instances.

The ontology building evaluation

For the ontology building evaluation, we typically compared our resulting ontology against a gold-standard which is suitably designed for the domain of discourse [54]. This may in fact be an ontology considered to be well-constructed to serve as reference. As we aforementioned, the domain of discourse that we treat is E-commerce [55]. The ontology reference that represents the E-commerce domain is GoodRelations ontology [56]. It is a standardized vocabulary for product, price, and company data that can be (i) embedded into existing and dynamic web pages and (ii) processed by other computer. Generally, GoodRelations is used to facilitate creation of formal descriptions of product offering for electronic commerce. Table 8 shows the basic metrics of the GoodRelations Ontology versus the resulting ontology.

Table 8 The basic metrics of the GoodRelations versus the resulting ontology

Ontology	TCC	Axioms	LAC	TOP	TDP	TINDV
GoodRelations	38	1141	450	53	49	46
Resulting ontology	23	28,816	28,816	60	72	47,803

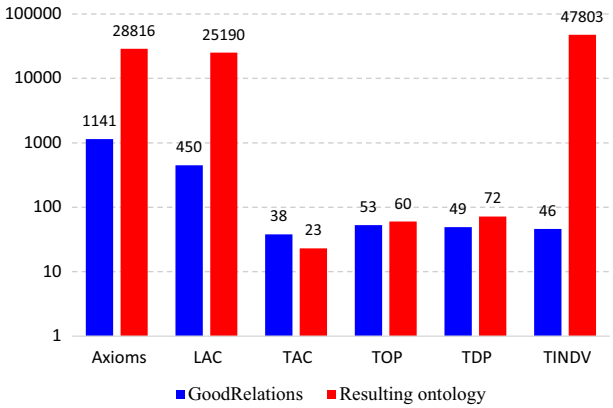

Fig. 6 Discriminative effect of the knowledge coverage

Table 9 The list of metrics for evaluating the resulting ontology

Ontology	A-Box evaluation		T-Box evaluation		
	AP	CR	AR	IR	RR
GoodRelations	1.21	0.236	1.89	0.5	0.9082
Resulting ontology	2078.39	0.7142	5.142	2.357	0.3125

The basic metrics of ontology provide the count number of classes, objects, axioms, properties and instances used in the ontology. Considering the result presented in Fig. 6, it is clearly seen that our resulting ontology covers more basic knowledge than the reference ontology with regard to the total number of classes, the total number of datatype property (TDP) and object Properties (TOP), the number of logical axioms (LAC), the number of axioms, and the number of instances (TINDV). For instance, the TINDV are 47,803 and 46 for the resulting ontology and reference ontology respectively. However, we cannot discuss the quality of the resulting ontology based on these metrics, because these metrics represent just the discriminative effect of the knowledge coverage [54] as shown in Fig. 6. In this respect, the two following subsections are well explained the metrics that we used to measure the quality of our ontology.

The TBox evaluation

IR values close to zero indicate flat or horizontal ontology representing perhaps more general knowledge while large values represent vertical ontologies describing detailed knowledge of a domain. As depicted in Table 9, the IR for our ontology is 2357 while for GoodRelations is 0.5. This indicates that our resulting ontology describes the

Table 10 Ontological output of each mapping approach

	[18]	[19]	[20]	[21]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[22]	[30]	Our Approach
Owl:class	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Rdfs:class														
Owl:AllValuesFrom														√
owl:someValuesFrom														
owl:hasValue														√
owl:maxCardinality	√		√	√	√	√		√	√		√	√	√	√
owl:minCardinality	√		√	√	√	√		√	√		√	√	√	√
owl:cardinality			√	√	√			√	√		√	√		√
owl:intersectionOf														
owl:unionOf														
owl:complementOf														
rdfs:subClassOf		√	√		√	√			√		√	√	√	√
owl:equivalentClass														
owl:disjointWith														
owl:oneOf		√	√		√						√	√		√
owl:DatatypeProperty	√	√	√	√	√	√	√	√	√	√	√	√	√	√
owl:ObjectProperty	√	√	√	√	√	√	√	√	√	√	√	√	√	√
rdfs:subPropertyOf														
rdfs:domain	√		√	√	√	√	√	√	√	√	√	√	√	√
rdfs:range	√		√	√	√	√	√	√	√	√	√	√	√	√
owl:equivalentProperty														
owl:inverseOf	√				√	√	√		√		√	√	√	√
owl:FunctionalProperty	√			√	√	√		√	√		√		√	√
owl:InverseFunctionalProperty					√	√		√	√		√	√	√	√
owl:TransitiveProperty					√							√		√
owl:SymmetricProperty					√							√		√
Data Migration				√						√	√	√	√	√

Table 11 A list of competency questions with answers

Queries	Negative (N)/ positive (P) answers
Query 1: find movie for a given set of generic features such as name and duration, etc	P
Query 2: retrieve basic information about a specific movie for display purposes	P
Query 3: retrieve basic information about a specific movie for display purposes	P
Query 4: find movie having a label that contains specific words	P
Query 5: find movies that are similar to a given movies	P
Query 6: find the category of movies	P
Query7: find Text description of a given movie's title	P

E-commerce domain better than the reference ontology. However, the relationships richness for the ontology reference is greater than the resulting ontology, which indicate that the reference ontology contains many relationships other than class-subclass relations, where our ontology is richer than a taxonomy with only class-subclass relationships. On other hand, the attribute richness for our resulting ontology is significantly greater than the AR of the reference ontology, which indicates that our ontology defined more knowledge than the reference ontology.

According to the result depicted in Table 10, we presented the TBox output of each surveyed approach using a specific OWL elements. In addition, the last column shows our mapping result. It is evident from this table that our ontology is greatly contained a high number of semantics compared to the other approaches.

The ABox evaluation

The first group of measures that we have considered for this validation is related to the knowledge distribution in the ontology. As we can see in Table 7, the average population (AP) for the resulting ontology is better than the ontology reference. Compared to the reference ontology, the value of the AP of our ontology, which is 2078.39, involves that our ontology offers a sufficient number of instances for describing the e-commerce

domain. According the authors in [57], this metric is proposed to be used in conjunction with the class richness metric (CR). In this respect, we calculated the CR metric. The value of this metric confirms that our ontology's classes are populated with a high number of instances with regard to GoodRelations ontology, and this is reflects the diversity of knowledge embedded in our A-Box.

The competency questions (CQs)

Now to validate the resulting ontology in its totality we have checked if it is able to answer the competency questions established previously. As depicted in Table 11, the positive Answer means that our ontology can provide the correct answer to the query, while Negative answer means that the ontology cannot answer the query. Therefore, our resulting ontology answered all the formulated queries with a positive feedback. These queries are formulated in SPARQL Query Language [58]. For a high-level description of each query, we refer the reader to our GitHub Link.[3]

Eventually, we can conclude that our proposed life cycle shows sufficient exactitude to be used for selecting an appropriate database for building ontology and it is able to exhibit very accurate result. Note that the life cycle phases represents formal stages-gates; they save as criteria to help ontologist for answering a very important question: how to select a Relational database that provides a sharp and clear boundary between the relational model and ontological model. From this experiment, we can notice that, we start our experimental study with 6 databases, and during each phase in life cycle, we evaluated the outcome of this phase in order to check if we made enough progress to move to the next phase. As a result, instead of converting the six databases directly into ontology, we early removed some RDBs that are not contained the sufficient semantics for representing the ontological model.

Conclusion

To sum up, in this paper, we tried to gather the most important and contributing approaches in the subject of the mapping of the relational database to ontology. We attempted to provide the reader with concise overview of these approaches in terms of identifying the main drawbacks that the researchers in this field are faced as well as suggesting solutions. In addition, the biggest contributions within this paper are the following: (1) We propose a new life cycle for ontology learning from RDBs based on the software engineering requirements; (2) We describe a new method for building ontology from Relational database based on the predefined life cycle; (3) We add three new semantics that can be extracted from RDB; (4) we suggest an evaluation process based on two categories of metrics: (i) Conceptual Ontology (T-Box) Evaluation metrics; (ii) Factual ontology(A-Box) evaluation metrics. In future works, we aim to focus on the cleaning and conditioning the data embedded in the relational database in order to improve the quality of the resulting ontology. Also, we plane to focus on different structured sources of information such as Excel spreadsheet, comma-separated value (CSV),

[3] https://github.com/bilalbenma.

and SQL DDL files in order to integrate these diverse data Format. Finally, we plan to move toward the unstructured data sources for constructing ontologies.

Abbreviations
ABox: Assertional box; TBox: Terminological box; RDB: Relational database; SQL: Structured query language; DDL: Data definition language; PK: Primary key; FK: Foreign key; CQ: Competency question; OWL: Ontology web language; AR: Attribute richness; IR: Inheritance richness; RR: Relationship richness; CR: Class richness; AP: Average population; TCC : Total number of classes; LAC: Logical axioms; TOP: Total number of object properties; TDP: Total Number of datatypes property; TINDV: Total number of instances (or individuals); R2RML: Relational to RDF mapping language; MGE: Mapping generator engine.

Acknowledgements
Not applicable.

Authors' contributions
BBM, IC and AZ conceived of the presented idea. BBM developed the theory, performed the algorithms in addition to writing the manuscript with support from IC and AZ. IC and AZ verified the analytical methods and they helped supervise the project. Finally, all authors discussed the results and contributed to the final manuscript. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The data that support the findings of this study are available from the corresponding author (Bilal Ben Mahria), upon reasonable request. The Data contains the Relational databases in addition to the RDF files generated.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
Not applicable.

Received: 19 October 2020 Accepted: 16 January 2021
Published online: 28 January 2021

References

1. Dermeval D, Vilela J, Bittencourt II, Castro J, Isotani S, Brito P, Silva A. Applications of ontologies in requirements engineering: a systematic review of the literature. Requirements Eng. 2016;21:405–37.
2. Simperl EPB, Tempich C. Ontology engineering: a reality check. In: OTM confederated international conferences "On the Move to Meaningful Internet Systems". Berlin: Springer; 2006. p. 836–54.
3. Cardoso J. The semantic web vision: where are we? IEEE Intell Syst. 2007;22:84–8.
4. Bürger T, Simperl E. Measuring the benefits of ontologies. In: OTM confederated international conferences "On the Move to Meaningful Internet Systems". Berlin: Springer; 2008. p. 584–94.
5. Calero C, Ruiz F, Piattini M. Ontologies for software engineering and software technology. Berlin: Springer Science & Business Media; 2006.
6. Sure Y, Staab S, Studer R. On-to-knowledge methodology (OTKM). In: Handbook on ontologies. Berlin: Springer; 2004. p. 117–32.
7. Grüninger M, Fox MS. The role of competency questions in enterprise engineering. In: Benchmarking—theory and practice. Berlin: Springer; 1995. p. 22–31.
8. Fernández-López M, Gómez-Pérez A, Juristo N. METHONTOLOGY: from ontological art towards ontological engineering. In: AAAI-97 Spring symposium series, Stanford University, EEUU, 24–26 March 1997.
9. Uschold M, King M. Towards a methodology for building ontologies. Citeseer. Edinburgh: Artificial Intelligence Applications Institute, University of Edinburgh; 1995.
10. Noy NF, McGuinness DL. Ontology development 101: a guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01–05 and …2001.
11. Al-Arfaj A, Al-Salman A. Ontology construction from text: challenges and trends. Int J Artif Intell Expert Syst IJAE. 2015;6:15–26.
12. Antoniou G, Van Harmelen F. A semantic web primer. Cambridge: MIT press; 2004.
13. Maedche A, Staab S. Ontology learning. In: Handbook on ontologies. Berlin: Springer; 2004. p. 173–90.
14. Santoso HA, Haw S-C, Abdul-Mehdi ZT. Ontology extraction from relational database: concept hierarchy as background knowledge. Knowl-Based Syst. 2011;24:457–64.
15. He B, Patel M, Zhang Z, Chang KC-C. Accessing the deep web. Commun ACM. 2007;50:94–101.
16. Martinez-Cruz C, Blanco IJ, Vila MA. Ontologies versus relational databases: are they so different? A comparison. Artif Intell Rev. 2012;38:271–90.
17. Meersman R. Ontologies and databases: more than a fleeting resemblance. STAR. 03; 2001.

18. Telnarova Z. Relational database as a source of ontology creation. In: Proceedings of the international multiconference on computer science and information technology. New York: IEEE; 2010. p. 135–9.
19. Zhang H, Diao X, Yuan Z, Chun J, Huang Y. EVis: a system for extracting and visualizing ontologies from databases with web interfaces. In: 2012 fourth international symposium on information science and engineering. New York: IEEE; 2012. p. 408–411.
20. Li M, Du XY, Wang S. Learning ontology from relational database. In: 2005 international conference on machine learning and cybernetics. New York: IEEE; 2005. p. 3410–5.
21. Ghawi R, Cullot N. Database-to-ontology mapping generation for semantic interoperability. In: Third international workshop on database interoperability (InterDB 2007); 2007.
22. Astrova I, Korda N, Kalja A. Rule-based transformation of SQL relational databases to OWL ontologies. In: Proceedings of the 2nd international conference on metadata & semantics research. Citeseer; 2007. p. 415–24.
23. Tirmizi SH, Sequeda J, Miranker D. Translating sql applications to the semantic web. In: International conference on database and expert systems applications. Berlin: Springer; 2008. p. 450–64.
24. Zhang L, Li J. Automatic generation of ontology based on database. J Comput Inf Syst. 2011;7:1148–54.
25. Yiqing L, Lu L, Chen L. Automatic learning ontology from relational schema. In: 2012 IEEE symposium on robotics and applications (ISRA). New York: IEEE; 2012. p. 592–5.
26. Buccella A, Penabad MR, Rodriguez FJ, Farina A, Cechich A. From relational databases to OWL ontologies. In: Proceedings of the 6th national russian research conference; 2004.
27. Sedighi SM, Javidan R. A novel method for improving the efficiency of automatic construction of ontology from a relational database. Int J Phys Sci. 2012;7:2085–92.
28. Bakkas J, Bahaj M, Marzouk A. Direct migration method of rdb to ontology while keeping semantics. Int J Comput Appl. 2013;65:6–10.
29. Sequeda JF, Tirmizi SH, Corcho O, Miranker DP. Survey of directly mapping sql databases to the semantic web. Knowl Eng Rev. 2011;26:445–86.
30. Tissot H, Huve CAG, Peres LM, Del Fabro MD. Exploring logical and hierarchical information to map relational databases into ontologies. Int J Metadata Semant Ontol. 2019;13:191–208.
31. Konstantinou N, Spanos DE. Materializing the web of linked data. Berlin: Springer; 2015.
32. Press R. Ontology and database mapping: a survey of current implementations and future directions. J Web Eng. 2008;7:001–24.
33. Gomez-Perez A, Fernández-López M, Corcho O. Ontological engineering: with examples from the areas of knowledge management, e-commerce and the semantic web. Berlin: Springer Science & Business Media; 2006.
34. Khan ZC. Applying evaluation criteria to ontology modules. (2018)
35. Sequeda JF, Tirmizi SH, Miranker DP. SQL databases are a moving target. In: Position paper for W3C workshop on RDF access to relational databases; 2007.
36. Yu L. A developer's guide to the semantic Web. Berlin: Springer Science & Business Media; 2011.
37. Domingue J, Fensel D, Hendler JA. Handbook of semantic web technologies. Berlin: Springer Science & Business Media; 2011.
38. Zhe Y, Zhang D, Chuan YE. Evaluation metrics for ontology complexity and evolution analysis. In: 2006 IEEE international conference on e-business engineering (ICEBE'06). New York: IEEE; 2006. p. 162–70.
39. Vrandečić D. Ontology evaluation. In: Handbook on ontologies. Berlin: Springer; 2009. p. 293–313.
40. Services, E.E. Data science and big data analytics: discovering, analyzing, visualizing and presenting data. New York: Wiley; 2015.
41. Pan JZ, Vetere G, Gomez-Perez JM, Wu H. Exploiting linked data and knowledge graphs in large organisations. Berlin: Springer; 2017.
42. de Medeiros LF, Priyatna F, Corcho O. MIRROR: Automatic R2RML mapping generation from relational databases. In: International conference on web engineering. Berlin: Springer; 2015. p. 326–43.
43. Gutierrez C, Hurtado CA, Mendelzon AO, Pérez J. Foundations of semantic web databases. J Comput Syst Sci. 2011;77:520–41.
44. Lourdusamy R, John A. A review on metrics for ontology evaluation. In: 2018 2nd international conference on inventive systems and control (ICISC). New York: IEEE; 2018. p. 1415–21.
45. Tartir S, Arpinar IB, Moore M, Sheth AP, Aleman-Meza B. OntoQA: Metric-based ontology quality analysis; 2005.
46. Fernández M, Overbeeke C, Sabou M, Motta E. What makes a good ontology? A case-study in fine-grained knowledge reuse. In: Asian Semantic Web Conference. Berlin: Springer; 2009. p. 61–75.
47. Spanos D-E, Stavrou P, Mitrou N. Bringing relational databases into the semantic web: a survey. Semantic Web. 2012;3:169–209.
48. Jimborean I, Groza A. Ranking ontologies in the ontology building competition boc 2014. In: 2014 IEEE 10th international conference on intelligent computer communication and processing (ICCP). New York: IEEE; 2014. p. 75–82.
49. Obrenović N, Luković I. An approach to consolidation of database check constraints. ICIST 2014; 2014.
50. El Alami A, Bahaj M. The migration of a conceptual object model COM (conceptual data model CDM, unified modeling language UML class diagram...) to the Object Relational Database ORDB. MAGNT Research Report (ISSN. 1444–8939). 2:318–32.
51. Din AI. Structured query language (SQL) A practical Introduction; 2014.
52. Vidal VMP, Casanova MA, Neto LET, Monteiro JM. A semi-automatic approach for generating customized R2RML mappings. In: Proceedings of the 29th annual ACM symposium on applied computing; 2014. p. 316–22.
53. Benmahria B, Chaker I, Zahi A. Validation and evaluation of the mapping process for generating ontologies from relational databases. In: World conference on information systems and technologies. Berlin: Springer; 2019. p. 337–50.
54. Hlomani H, Stacey D. Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: a survey. Semantic Web J. 2014;1:1–11.

55. Ordysiski T. Ontology of E-commerce solution. Studia i Materialy Polskiego Stowarzyszenia Zarzadzania Wiedza/ studies & proceedings polish association for knowledge management; 2011. p. 384–95.
56. Hepp M. Goodrelations: An ontology for describing products and services offers on the web. In: International conference on knowledge engineering and knowledge management. Berlin: Springer. p. 329–46.
57. Sicilia M-Á, Rodríguez D, García-Barriocanal E, Sánchez-Alonso S. Empirical findings on ontology metrics. Expert Syst Appl. 2012;39:6706–11.
58. Schmidt M, Meier M, Lausen G. Foundations of SPARQL query optimization. In: Proceedings of the 13th international conference on database theory; 2010. p. 4–33.

Publisher's Note