Journal of Ambient Intelligence and Humanized Computing (2023) 14:16841–16851
https://doi.org/10.1007/s12652-023-04693-8

ORIGINAL RESEARCH

Ontology learning from relational database: a review

Rosalba Mosca[1] · **Massimo De Santo**[1] · **Rosario Gaeta**[1]

Received: 22 February 2023 / Accepted: 28 August 2023 / Published online: 19 September 2023

Abstract
A relational database (RDB) is a digital database that uses components (such as constraints, tables, keys, etc.) to manage data in a structured manner. Because of these components, RDBs are considered 'poor' from a semantic point of view, precisely because of the structure-oriented nature of the components used. One way to eliminate this limitation is to transform the RDB into an ontology. The purpose of this article is to review the different approaches existing in the literature to extract data from an RDB and convert it into ontology instances. Two approaches are used to integrate the mapping between RDBs and ontologies. The first allows ontologies to be extracted from an RDB, the second consists of a mapping of the relational database to an existing ontology. Our proposed review focuses on methods for creating a specific ontology from an RDB. The proposed review examines this field, classifying the methods that will be analyzed according to their inputs and outputs. Such classification may be useful for understanding the usability of methods. The aim is to critically review existing studies to help outline this research topic's progress and identify methods' gaps and functionalities.

Keywords Ontology · Ontology learning · Relational databases

1 Introduction

The construction of an ontology is an engineering activity. Two main approaches to ontology construction can be found in the literature: the manual approach and ontology learning (Cimiano et al. 2009). Constructing an ontology from scratch or manually (Sure et al. 2004; Grüninger and Fox 1995; Mariano Fernández and Juristo 1997; Uschold and King 1995; Noy and McGuinness 2001) is resource-intensive because it involves collaborative work that requires the skills of ontology engineers. Building an ontology manually is also difficult because knowledge is dynamic: it develops very quickly in all real-world domains. Consequently, ontology engineers must continuously revise and update the resulting ontologies (adding new terms and concepts). Building an ontology from scratch is not intuitive, time-consuming, error-prone and expensive (Al-Arfaj and Al-Salman 2015); moreover, it is this approach is prone to a loss of semantics (Ma and Molnár 2020). Ontology learning was created to overcome the limitations of the manual approach. Ontology learning (OL) is an approach to automatically or semi-automatically construct an ontology from different knowledge domains. OL provides potential opportunities for semantic integration and, at the same time, outlines new challenges related to its use (Ma and Molnár 2022). 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 (Maedche and Staab 2004). For several reasons, most studies focus on the relational database as an information source. Firstly, about 70 (Santoso et al. 2011) Relational databases also provide a comprehensive information resource (He et al. 2007). Finally, they constitute one of the most useful and widespread data storage and manipulation techniques. However, relational databases need more semantic meaning so as not to hinder the ability to achieve interoperability between information systems (Martinez-Cruz et al. 2012). The work of d'Avila Garcez and Lamb (2020) constitutes an important reflection on current advances in the field of Artificial Intelligence and Machine Learning. In this paper, we propose

✉ Rosalba Mosca
rmosca@unisa.it

Massimo De Santo
desanto@unisa.it

Rosario Gaeta
rgaeta@unisa.it

[1] Department of Industrial Engineering, University of Salerno, Via Giovanni Paolo II 132, Fisciano 84084, Italy

a review of different approaches to extract the data model from an RDB schema and then convert them into instances of an ontology. This conversion enables the interoperability of information that otherwise risks remaining 'an island of data' stored in individual databases. Many approaches exist to integrate the mapping between RDBs and ontologies. The first is used for extracting ontologies from an RDB and the second is for mapping between relational databases and an existing ontology (Hazber et al. 2016). In the second method, such as the one proposed by Vavliakis et al. (2013), both ontology and relational databases are inputs. These methods start from an RDB to create instances for inclusion in the ontology concepts. Conversely, the former techniques take the RDB as input and generate an ontology as output. In this paper, we have focused on the latter methods.

2 Methodology

Multiple searches of open scientific literature were conducted on Scopus, Google Scholar and the IEEE Xplore digital library.

The main search strategy was the hunt for articles relevant to the topic "From RDB to Ontology". From a detailed state-of-the-art analysis, a total of 13 relevant articles with the relative proposed methods were identified, mainly based on the number of citations made to these articles and the quality of the bibliography reported. In addition, two tools for conversion from RDB to Ontology released by Protégé Stanford (2023) as plugins were analyzed. These two plugins are still available in 3.x versions of Protégé but, as of today, are no longer supported.

2.1 From RDB to ontology

A relational database is a digital database based on the relational data model proposed by Codd (1970). To manage data in a structured form, RDBs use several components (such as tables, constraints, keys, etc.). After thoroughly analyzing the information system, designers establish a conceptual model used to create databases. However, RDBs are 'poor' from a semantic point of view precisely because of the structure-oriented nature of their components. Transforming an RDB into an ontology allows it to be used for semantic purposes. The limitation of the RDB lies precisely in its structure, which consists of tables linked to foreign key constraints. Tom Gruber states, "An ontology defines a set of representation primitives with which to model a knowledge or discourse domain. Representation primitives are typically classes (or sets), attributes (or properties), and relations (or relationships between class members). The definitions of representation primitives include information on their meaning and constraints on their logically consistent application" (Mogotlane and Fonou-Dombeu 2016). The choice of representation of an ontology derives from the use and purpose for which it is modeled. There are many ways to represent ontologies. One of the most widely used languages for this purpose is Web Ontology Language (OWL). This language has a formally defined meaning and is an ontology language for the Semantic Web. OWL 2 ontologies (the latest language version) provide classes, properties, individuals, and data values and are stored as Semantic Web documents (W3C 2023).

2.2 Classification of the proposed approaches

Various classifications can be identified starting from the analysis of the selected papers on ontology learning from relational databases. We notice that all the methods apply mapping rules to translate certain RDB elements patterns into OWL elements, producing an OWL ontology file. Each work applies and describes such rules at a different level of detail. Some authors apply only the rules for translating the main and most typical pattern. In contrast, others perform a more detailed translation, converting, for example, the symmetrical relationships, queries, views, and triggers. The level of detail of the mapping rules for each method will be understandable from Table 2 and the following description of each work. We propose classifying the selected papers based on the input and output of the proposed approaches because we find such a way of classification extremely useful to understand the pros and cons of each proposal deeply.

2.2.1 Classification based on the input of the method

The methods proposed in the Review paragraph can be divided in:

- Approaches that can connect RDBs directly;
- Methods that perform parsing of a SQL-File, which can contain information written in DDL and DML.

Both methods take the information from the data sources to extract meta-data and data from them, applying a translation method to obtain the final owl ontology.

2.2.2 Classification based on the output of the method

All the approaches proposed in this review generate an OWL-File in output. Each of these methods produces Owl concepts from the database schema. Still, not all of them make import the data-producing instances. Note that for Owl concepts, we refer to classes, properties, axioms, etc. but not instances In particular, the methods proposed in our Review can be divided in:

- Methods that make a translation only based on meta-data but not on the data and that for this reason generate an Ontology with only Concepts and Properties but not Instances.
- Methods that make a translation of meta-data and an import of the data in the database, producing OWL Concepts, Properties, and Instances.
- Methods that create an OWL-Ontology with concepts and properties and that allow the user to access the data present in the database, for example converting SPARQL queries into SQL queries, but which do not directly import instances into the ontology.

2.3 Review

In the following, we describe the papers subject of this review. For each article, we will briefly indicate the type of input and output of the method proposed in the papers, according to the previously described classification:

Li et al. (2005) propose an approach for learning OWL ontology from data in relational databases using a set of mapping rules. The framework can connect to a database through a database analyzer in order to extract meta-data and data from it and produce an OWL-File containing Owl concepts and instances. This approach has been used in the Semantic Web project of Renmin University of China. In this method, a relational scheme in the third normal form (3NF) is required. It is a method that applies 12 mapping rules, divided into: class learning rules, property learning rules, hierarchy learning rules, cardinality learning rules and instance learning rules. Once the information is obtained, it is transferred to an ontology generator. The latter generates an ontology based on the information schema and rules. The user is allowed to modify the resulting ontology through the ontology editor and ontology reasoner. The proposed model is not bound to specific domains and applications. According to Sequeda et al. (2011) some of the rules proposed by this paper can be considered not optimized or ambiguous: in particular, the rule which creates the has-part and is-part object properties can also be used to represent a hierarchy. Moreover, creating $N * (N - 1)$ relation can be redundant since the n-ary relation can be modeled successfully with a class. Moreover, the authors declare they test the method on an existing digital library of Renmin University, but the paper does not clearly describe the evaluation phase.

In Ghawi and Cullot (2007), the authors propose a general interoperability architecture that uses ontologies to explicitly describe the semantics of information sources and web services to help the communication between the different components of the architecture. We are interested in the paper's main focus: a tool, part of the general architecture, called DB2OWL, which connects to a database, extracts meta-data from it and uses them to generate an owl ontology file automatically. The tool also generates a mapping document that preserves the set of transformations made. The resulting ontology contains only classes and properties while the instances are not loaded, but they remain in the databases in such a way that they can be retrieved in response to a user query. The paper, starting from 3 table cases with its 6 mapping rules, describes useful rules for converting database elements into classes, data properties, object properties, and hierarchical class structures. However, it does not define any conversion rule capable of building axioms starting from RDB base constraints (like a not null, unique, and primary key), which are typically required for a complete mapping of real relational databases. These shortcomings may limit the usability of the systems. Moreover, the paper does not propose any experimentation on an RDB database.

In the Zhang and Li (2011), the authors present a system that connects to a database and extracts the meta-data model of such RDB; then RDB meta-data model is converted into ontology concepts using a series of mapping rules (6 are explicitly described while the rules for the construction of the axioms are not explicitly described); finally, the ontology instances are mapped, and the complete owl-document is generated. The authors design and implement an automatic ontology generation system based on relational databases (OGSRD). This system has a three-part structure: reading the database metadata, building the ontology meta-model and generating the target ontology. Although the paper proposes architecture and mapping rules for performing ontology learning starting from a database, the article is unclear if and how the conversion of a table with 2 foreign keys and other attributes and the conversion of the table with more of the N foreign key must be mapped. Moreover, also the construction rules of ontology axioms appear vague.

A short paper by Yiqing et al. (2012), proposes a system that connects to a database via JDBC, obtaining useful schema information and using 11 mapping rules to convert such information into an ontology. The paper does not specify if there is any rule for importing also instances in the such ontology. Most tables are converted into owl classes using these rules, generating the related class axioms. When a class is generated, the corresponding datatype property is generated for each attribute that is not a foreign key (FK). The domain and range of each datatype property are also automatically derived. In particular, the range of a datatype property is obtained by mapping each of SQL datatypes of the database into an xml schema datatype of the ontology (with a mapping logic reported in the paper). Each of the FK attributes of a table is converted into an object property with the related domain and range. In this way, the method can map the one-to-many relationships between two entities into properties between two owl classes. The attributes conversion also considers the constraints applied to the original attributes in the relational database tables

(not null, unique, and so on). Obviously, suppose a table has no other properties except two foreign keys. In that case, it is mapped into two object properties between the 2 owl classes resulting from converting the original RDB tables. Indeed, this table can be considered a bridge table that realizes a many-to-many relationship. A detail worth noting is that the resulting owl ontology will be written in owl-full (indeed, a datatype property can be inverse functional only in owl-full language). Typically, such an OWL-Full ontology can be less useful in terms of possible inference on the data because OWL-Full is not completely processable by a reasoner. However, the authors of the paper do not report any performance evaluation of their method and do not show any application of such method in a real case of study.

In Zhang et al. (2012) the EVis system is presented, which supports mapping RDB into ontologies, editing the extracted ontology, and mapping it into the database. The EVis system consists of three modules: database-to-ontology module, ontology editing module, and ontology visualization module. The database-to-ontology module extracts the ontology from the relational database (which is assumed to be in 3NF) using OWL as the language for the resulting ontology. This module connects to the target database to automatically extract the schema data and produce an owl ontology. It does not import any data as ontology instances from the database nor provide any way to retrieve data inside it with user Sparql query. The ontology extraction procedure is based on a set of rules for mapping the SQL schema. The module provides editing functions to modify the SQL schema. The rules are provided in the form of first-order logic. The ontology-editing module is used for editing the ontologies extracted from the databases in the previous step. The ontology visualization is used to visualize the extracted ontology and map between ontologies and databases. This tool works on different web browsers, such as IE and Firefox, and can be published as a web service while more famous ontology editors, like Stanford (2023), run on desktops. The advantage of such EVis system is that it proposes a system to create, edit and visualize the ontology, but the main shortcoming of this paper is that it does not describe in detail the rule it proposes, but it only lists some of them. In addition, as written before, only a mapping of the owl concepts is provided without porting the instances. The prototype developed by the authors is shown with its interface in the paper, but they do not report an evaluation of the system's performance.

Bakkas et al. (2013) provide a method that extracts the RDB schema directly from the database to convert the RDB meta-data and data into an owl ontology file with owl concepts and instances. This method operates on two levels, applying two different algorithms: the first algorithm is based on reverse engineering; it then extracts the RDB schema (database schema is assumed to be normalized 3NF) and converts it into the ontology model (TBOX). Before starting the conversion, the procedure creates two classes, Entity and Association, which are superclasses of all the classes that will be created. The tables representing entity inherit from the class Entity, while the tables representing many-to-many relationships inherit from the class Association. The algorithm creates various classes with its data and object properties, considering various patterns of elements in the RDB schema. The second algorithm attempts to construct the ontology of individuals (ABOX) using data from different RDBs records based on the ontological model. The proposed method can do the conversion in OWL of all the types of tables together with their columns in an original way because it keeps a trace of the original nature of the relational database tables by creating the two superclasses Entity and Association. The paper provides no rules for mapping not null or unique constraints in the case of foreign key attributes, which can be an important shortcoming. In addition, when a primary key must be translated an inverse functional data property is created: this means that the generated ontology will be in OWL-Full Language. Unfortunately, as written before in the [24] comment, OWL-Full typically is not fully complained with reasoning activity, and for this reason, an ontology modeled in OWL-Full may be less useful than an OWL-DL Ontology in terms of possible inference on the data. The authors evaluate their prototype application on a case study based on a small dataset.

Thuy et al. (2014) introduce a method called RDB2RDF, which connects to a relational database and, using select queries, can extract meta-data and data from it to generate an ontology file reporting both owl concepts and instances. The paper considers that many ontology learning works proposed at that time performed a conversion from RDB to ontology without considering that usually, in such databases, some relational columns are also similar to others in the name: duplicate columns, representing the same information can lead to data redundancy. The authors study a method that measures the similarity between duplicate columns in an RDB and also proposes a transformation strategy for the detected levels of similarity. This is, therefore, a new method for solving the problem of duplicates. The method automatically produces an OWL ontology transforming relational database tables into ontology elements. The transformation allows RDF ontology to be inverted into relational tables. The approach consists of 5 small steps, which extract the necessary information to obtain the final owl ontology, calculate the similarity between the columns, and produce an owl schema file. Then the instances are stored in an XML file. The two files are then used to generate the final owl file. To evaluate the transformation method, a comparison is made between a relational database and an OWL file to establish the actual matches, comparing the results with other related methods (in particular, the analysis showed

better results for the proposed method with respect to the other). The metrics used are precision and recall. With respect to other papers, the evaluation has been described in more detail. While clearly describing the innovation presented, the paper does not describe how each pattern of an element in the RDB is translated into owl elements or axioms.

Dadjoo and Kheirkhah (2015) presented a method for transforming a relational database into an ontology in which the input of the system is a relational database, written in SQL Data Definition, and its output is an ontology model in the OWL structure that will be produced from the knowledge extracted from a relational database. Even if the authors do not specify anything, from the input and the evaluation presented in this work, we can deduce that the method imports only the concepts but not the instances of the owl ontology. The method is based on three stages: in the first, the relational database receives the desired information (metadata) as input and the database tables are classified according to it, then an intermediate conceptual model is created using the information from the first stage. In the third stage, the final ontology is generated from the intermediate graph provided as output from the previous phase. A series of rules are used for the middle graph and final ontology creation. Different from the other papers, this article proposes rules to translate triggers. When there is a trigger, a class (Event) and two DatatypeProperties (Time and Agent) are created within the ontology using the class domain. The two properties respectively indicate the time when the event occurs (Time) and the class on which the event depends (Agent). Each trigger type will be translated into an instance of a subclass of the "Event" class (insert, delete, and update). More, an ObjectProperty will be created and its domain will be the event class and as a range the subclass of the event class. The capability to translate RDB trigger in an ontology form is an optimal feature for an ontology because domain knowledge reflects the world, as it is constantly evolving. Although the fact that the data inside the database are not translated is a significant shortcoming.

In Hazber et al. (2016), the authors describe a method that applies 25 rules to build an ontology from an RDB schema and to convert RDB data into ontology instances, represented by RDF triples, which populates the previously built ontology. The proposed approach, connecting to an RDB using a JDBC driver, extracts metadata and data and uses it to construct an RDFS-OWL ontology. The approach is implemented using Apache Jena in Java and MYSQL and validated with examples using an ontology validator. The paper is very exhaustive in explaining the rules, which can cover the translation of various RDB patterns into Owl. It also proposes and shows several examples and explains the evaluation protocol used. Also, the authors compare with the other methods, highlighting the results and the mapping rules that are implemented by their approach and not taken into consideration by the other approaches. However, from the rules applied can be deduced that the owl language used is the Owl-Full, (because the columns with a unique constraint are converted in DataProperty that are InverseFunctional, this is possible only in owl-full). Unfortunately, as written before in the (Yiqing et al. 2012) comment, owl-full typically is not fully complained with reasoning activity, and for this reason, an ontology modeled in OWL-Full may be less useful than an OWL-DL Ontology in terms of possible inference on the data.

Liu and Gao (2018), proposes a method for learning ontology from an RDB called WN_Graph. Since this method is based on Dadjoo and Kheirkhah (2015) and the authors do not specify anything about the input and output of their model, we can deduce that, like (Dadjoo and Kheirkhah 2015), the input of the system is a relational database written in SQL Data Definition and that its output is an ontology model in the OWL structure in which only the concepts are imported but without any instances resulting from a mapping of the database data. To provide a better hierarchical relationship between concepts, the proposed method uses the intermediate conceptual graph model combined with WordNet so that the approach can extract more rich semantic relationships from the RDB. This method focuses on the identification of hierarchy relationships inferable by the RDB. In particular, the more two tables have identical columns the greater the conformity between these two tables. By mapping these tables into classes, a common top class may result. When the number of overlapping properties of the two classes exceeds a threshold, WordNet is used to help identify the word common to the two classes. At this point, a 'parent' class is constructed with a common word to name the two classes. In addition, a hierarchical relationship can also be deducible from the name of the tables since the table which belongs to a hierarchical relationship can have the same last word as a name. After segmenting a word, the method does a comparison, and if there is a common parent word, then the two classes will have a common parent class. The method uses these concepts, implemented through a series of algorithms, together with those introduced by Dadjoo and Kheirkhah (2015) to construct the graph model and then the owl ontology. The authors tested their method against an electronic medical record database of a hospital in Wuhan, using a manually built medical ontology as ground truth and comparing the result with the ontology created by Dadjoo and Kheirkhah (2015). The result highlights an improvement in the performance with respect to the original method of Dadjoo. Indeed, the great innovation and advantage of the proposed method are that it attempts to infer the semantic knowledge implicit in the RDB, unlike other methods.

In Ben Mahria et al. (2021), propose a method that, starting from an SQL file, extracts both concepts (T-BOX) and instances (A-BOX) to obtain the final OWL ontology. The method described extracts the metadata from RDB and generates the concept ontology following a series of transformation levels. Three new transformation rules are also proposed for converting the control and default constraints and improving class inheritance. The R2RML language is used to generate the A-BOX. The Database Metadata Extraction Engine (DMEE) extracts metadata from the SQL file (including DDL statements). Tables, primary and foreign keys, and columns constitute the metadata. Thirdly, the Mapping Generator Engine (MGE) generates a file containing the mapping (R2RML file). Finally, this R2RML model inputs the complete schema of the instances and the file containing a set of rules, then, using r2rml-kit-master, produces the data in RDF triples. Six databases were used for validation. However, some of these were removed because they were considered semantically poor (and thus, the resulting ontology will also be semantically poor). The constructed ontologies are evaluated against manually constructed ontologies (used as ground truths) recognized as good. The authors emphasize that the ontology constructed with their method adds semantic elements compared to other approaches. The method implements more rules than most other articles (it leaves only triggers, queries, and some types of control and hierarchical structure unmapped). However, it does not specify how FKs characterized only by the uniqueness constraint are mapped. In addition, the method applies a rule which translates each FK with reference to the table itself into a SymmetricProperty. This translation is not always correct because there are some cases in which a property is not symmetric, even if the properties have the same domain and range. Such problems affect also Transitive Property mapping. Similar considerations in terms of Symmetric and Transitive Property translation can be done also for the work (Hazber et al. 2016) previously cited.

In Benamar et al. (2021), the authors use a two-phase strategy based on MDE Settings which takes in input from an SQL file and builds an OWL Ontology with both owl concepts and instances. The scope of this method is to use the knowledge in a specific field owned by the database and use it to create an ontology. The strategy proposed by MDE consists of two steps: the first (pre-processing) structures the data in a database; from this structure obtained, the system generates an SQL file containing the tables and their extensions. This ascertains the synchronization between the input model and the RDB metamodel. The resulting input model is used in the mapping phase. This phase uses a set of rules in the Atlas Transformation Language (ATL) to transform the product model into an OWL ontology. The two meta-models used for this purpose are the relational meta-model, the OWL ontology meta-model, and a mapping between the elements of these meta-models. Mapping is described by six rules to produce both concepts and instances. The input model, the two meta-models, and the mapping are then sent to the transformation engine to generate an ontology. The solution was tested using a number of specially created databases and was built in the Eclipse environment. The authors highlight that by analyzing the value of the metrics used in the evaluation, the tool provides very encouraging results. Although, the paper proposes a small number of rules with respect to other work of the same period.

The approach proposed by Lakzaei and Shamsfard (2021), takes a relational database (normalized in a 3NF form) as input and converts it into an ontology, producing both concepts and instances. A number of techniques are employed to transform the relational schema into database components. Subsequently, the different components of the ontology are created through the mapping rules. These are divided into rule classes: rules for concepts and their properties, rules for data types, rules for hierarchical relations, instances, and axioms. Through these, all database components are converted into the corresponding ontology components to generate the final ontology. The approach was implemented using C#.NET and SQL Server. The authors evaluated the method by comparing it to all the approaches represented. The performances used for the comparison are the following: the creation of ontological elements, capacity to examine RDB elements, and calculation and comparison of precision, recall, and measure F. This work is the most completed from the point of view of the mapping rule implemented, introducing a series of rule for translating all the type of hierarchical patterns inferable by an original RDB, and also query and view, into owl concepts. Although, also this work, like the Li et al. (2005), proposes a not-optimal translation of the n-ary relationship table, which is converted in $N * (N - 1)$ relationship, which may produce unnecessary relationship, while this type of relationship is representable by simple owl-classes (Sequeda et al. 2011).

DataMaster (2023) is a Protege plug-in for importing schema structure and data from relational databases into Protege. It takes data and meta-data from a database connecting using JDBC/ODBC drivers and gives in output an owl ontology composed of owl concepts and instances imported by the data inside the database.

OntoBase (2023) automatically generates ontology grounded in the database (by means of Protégé API using frames and facets), and it is, therefore, possible for a user to compose new classes starting from the columns of one or more tables. It utilizes reverse engineering to create ontology from a relational database schema. The tool connects through JDBC/ODBC drivers to a database and converts its schema into owl ontology concepts. Then the database can be queried in real-time.

The study by Mogotlane and Fonou-Dombeu (2016) proposes a series of mapping principles and then examines the two plugins, comparing them based on a real case of study. The paper highlights that DataMaster has a slight deviation from the proposed mapping principles in the creation of Data Properties and Object Properties. It does not produce any cardinality in response to not null and null columns. Also, the inheritance hierarchies produced are not compliant with the mapping principles. From the proposed comparative analysis, the study highlights that OntoBase outperforms DataMaster in the creation of data, object properties, and hierarchy structures. Although, like DataMaster, it does not produce any cardinality in response to not null and null columns. In general, the comparison between the two methods reveals that the structure of the ontology obtained with the OntoBase plug-in has fewer deviations than the one obtained with DataMaster, and that is more compliant with the mapping principles established by the authors of the paper. Another shortcoming of DataMaster is that it translates the database's foreign key into a class and each foreign key into instances of that class. This is a way of managing foreign keys, which is quite useless for generating a real owl ontology.

2.4 Comparison

Table 2 summarizes, for each discussed/selected paper, the translation of the RDB element patterns into Owl elements and axioms, together with the Owl language of the ontology given in the output. Generally, the owl language of the paper is deduced starting from the rules used in the translation: if the rules generate owl constructs that are typically not allowed by owl-dl, then it can be deduced that the language used is owl-full. Since Zhang et al. (2012), Thuy et al. (2014) and Benamar et al. (2021) do not precisely describe the conversion rules applied by their methods, a direct comparison with the other articles appears difficult. For this reason, these works are not reported in Table 2. For space reasons, also Liu and Gao (2018) is not included, but since it is based on the work of Dadjoo and Kheirkhah (2015), the column will be almost equal with the addition of a more detailed management of hierarchies translation. Each cell expresses the mapping between the RDB pattern and the corresponding Owl element, and when a cell is empty, it means that the corresponding mapping is not considered by the paper. Table 1 is the legend of the symbols which can be ambiguous in the reading of Table 2. Table 2 has been compiled to quickly and easily provide a view of how the various patterns in an SQL database are mapped into patterns in an OWL ontology, focusing, in this case only on the TBOX generation. From such table, it is possible to observe that the work by Li et al. (2005) and Ghawi and Cullot (2007) express fewer rules than the others and, in general, are vaguer than others about the possible rules to be used, especially in translating the various constraints related to table attributes. Compared to this work, the method of Yiqing et al. (2012) and Bakkas et al. (2013) propose a more detailed translation of the constraints that may be present in a database, but they completely leave out the possible translation of hierarchical structures, which are fundamental in the construction of complex, well-structured ontologies. Moreover, as can be seen from Table 1, such works, together with the Hazber et al. (2016), give in output an OWL-Full Ontology (because of the inverse functional data property translation of the unique constraint), which, since cannot be used with a reasoner, may be less useful than ontologies written in OWL-DL. The method proposed by Ben Mahria et al. (2021) tries to generate an OWL ontology complete of various OWL Patterns (also SymmetricProperty and TransitiveProperty, which was considered only by Hatzber Et al.) but maintaining an OWL-DL syntax. However, the translation of the symmetric and transitive property turns out to be forced, as expressed before, if not mediated by human operators. The work by Lakzaei and Shamsfard (2021) is very detailed in the translation of SQL to OWL patterns, but it has a non-optimal management of the n-ary relationship table, as for the Li et al. (2005) work. In conclusion, from the analysis of the table, it is possible to notice that many of such works propose valid rules for the translation of certain SQL patterns into OWL patterns: in particular, the works of Ben Mahria et al. (2021), which are the newest, are those which try to propose a more detailed translation between SQL and OWL patterns. However, from our analysis can be seen that each method presents some shortcomings in its ontology generation and that, after about 20 years of work in the field of Ontology learning starting from RDBs, no convergence has yet been found on the best way to convert the data source into a valid ontology. For this reason, a merging of the various rules and advantages of the approaches can be desirable in order to be able to carry out an automatic generation of an ontology that is as useful and faithful as possible to the semantics implicit in the original data source.

Table 1 Legend of the symbols

Type of symbol	Symbol	Meaning
Key	FK	Foreign key
	PK	Primary key
Property	DataProp	Data property
	ObjProp	Object property
	Func Prop	Functional property
	Inv Func Prop	Inverse functional property
Cardinality	MaxCard	Maximum cardinality
	MinCard	Minimum cardinality
	Card	Cardinality

Table 2 Conversion table between RDB pattern and OWL

Pattern	Type	Li et al.	Ghawi and Cullot	Zhang et al.	Yiqing et al.	Bakkas et al.	Dadjoo and Kheirkhah	Hatzber et al.	Ben Mahria et al.	Lakzaei and Shamsfard
Lang		Owl-DL	Owl-DL	Owl-DL	Owl-Full	Owl-Full	Owl-DL	Owl-Full	Owl-DL	Owl-DL
Table	No FK	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class
	1 FK	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class	owl:class
	Only 2 FK	2 owl: objprop	2 owl: objprop	2 owl: objprop	2 owl: objprop	owl:class	2 owl: objprop	2 owl: objprop	2 owl: objprop	2 owl: objprop
	2 FK and other no FK columns		owl:class		owl:class	owl:class	owl:class	owl:class	owl:class	owl:class
	Only N FK	N*(N-1) objprop	owl:class		owl:class	owl:class	owl:class	owl:class	owl:class	N*(N-1) objprop
	N FK and other no FK columns		owl:class		owl:class	owl:class	owl:class	owl:class	owl:class	owl:class
Column	!FK without constraint	Dataprop	Dataprop	Dataprop	Dataprop	Dataprop+ FuncProp	Dataprop	Dataprop	Dataprop	Dataprop+ FuncProp+ Maxcard = 1
	!FK+!null	Dataprop+ Mincard=1			Dataprop+ Mincard=1	Dataprop+ FuncProp	Dataprop+ Mincard=1	Dataprop+ Mincard=1	Dataprop+ Mincard=1	Dataprop+ Funcprop+ Card=1
	!FK+unique	Dataprop+ Maxcard=1			Dataprop+ Inverse Func Prop	Dataprop+ FuncProp+ InvFunc	Dataprop+ Funcprop+ MaxCard=1	Dataprop+ Inverse Func Prop	Dataprop+ Maxcard=1	Dataprop+ FuncProp+ Maxcard = 1
	!FK+PK	Dataprop+ Card=1			Dataprop+ Mincard=1+ InverseFunc Prop	Dataprop+ FuncProp+ InvFunc	Dataprop+ Funcprop+ Card=1	Dataprop+ Inverse func prop+ Min-card=1	Dataprop+ Card=1	Dataprop+ Funcprop+ Card=1
	FK without constraint	1 or 2 ObjProp	2 Objprop+ First is functional	Objprop	Objprop	ObjProp+ FuncProp	Objprop	Objprop	Objprop+ Func prop +Min-card =1 for Inverse prop	2 objprop
	FK+!null	1 or 2 ObjProp+ First with Mincard=1			Objprop+ Min-card=1	ObjProp+ FuncProp	Objprop+ Min-card=1	Objprop+ Min-card=1	Objprop+ Card=1+ Mincard =1 for Inverse prop	2 objprop First Is Func Prop and Card = 1
	FK+unique	1 or 2 ObjProp+ First with Maxcard=1			Objprop+ Inverse FuncProp	ObjProp+ FuncProp	Objprop+ InverseFunc Prop+ Max-card=1	Objprop+ Inverse Func Prop		2 objprop+ First is Inverse Funcprop

Table 2 (continued)

Pattern	Type	Li et al.	Ghawi and Cullot	Zhang et al.	Yiqing et al.	Bakkas et al.	Dadjoo and Kheirkhah	Hatzber et al.	Ben Mahria et al.	Lakzaei and Shamsfard
	FK+!null +unique	1 or 2 ObjProp+ First with Card=1			Objprop+ Inverse FuncProp+ Mincard=1	ObjProp+ FuncProp+ The class of the FK inherits from Association class if $FK \subseteq PK$	Objprop+ Func Prop+ Card=1	Objprop+ Inverse-Func Prop+ Mincard=1 Or Card=1 for one-to-one where $FK \in PK$	Objprop+ FuncProp+ FuncProp for Inverse Prop (+Card=1 if not PK)	2 objprop+ First Is Funcprop and InverseFunc Prop+ Card = 1
Hierarchy	Column as type of sub-entities									Constraints on a column become subclasses
	Only sub-entities are represented									Subclass of new father class
	Tables share PK	rdfs: subclass	rdfs: subclass	rdfs: subclass			rdfs: subclass	rdfs: subclass	rdfs: subclass	rdfs: subclass
Symmetric	FK to the table itself							Owl: Symmetric Property	Owl: Symmetric Property	Owl: Symmetric Property
Transitive	Symmetric and trigger on delete cascade							Owl: Transitive Property	Owl: Transitive Property	

3 Conclusion

This article aims to provide a review of different approaches to learning ontology from a relational database and extracting data from an RDB schema model, converting them into ontology instances. Two different problems can be faced in the context of the mapping between RDB and ontologies. The first is to extract an ontology from an RDB, and the second is to map a relational database to an ontology that already exists. Within this work, we have chosen an overview of the first method for creating a new and specific ontology from an RDB. Our aim was to review existing studies on this topic critically. This will help to outline progress in this research topic, so as to identify gaps and the functionalities of methods. The problem with many of the methods mentioned is that they have remained at the prototype stage and are not available for use by the community. In fact, some of these methods have stopped at a development stage and are not yet fully-fledged products. In other cases, these approaches have not yet been applied to real-world databases to verify their performance in automatically converting relational databases into ontologies. It is amply demonstrated in the literature that an ontology layer is an important added value to the data as it makes it linked and interrogable and thus enriches any information assets (Fensel 2001; Uschold and Gruninger 2004).

Funding Open access funding provided by Università degli Studi di Salerno within the CRUI-CARE Agreement.

Data availability For this paper no data are available.

References

Al-Arfaj A, Al-Salman A (2015) Ontology construction from text: challenges and trends. Int J Arti Intell Expert Syst 6(2):15–26

Bakkas J, Bahaj M, Abderrahim M (2013) Direct migration method of RDB to ontology while keeping semantics. Int J Comput Appl 65:975–8887

Ben Mahria B, Chaker I, Zahi A (2021) A novel approach for learning ontology from relational database: from the construction to the evaluation. J Big Data 8(1):25. https://doi.org/10.1186/s40537-021-00412-2

Benamar B, Bouchiha D, Redha R et al (2021) Mapping relational database to owl ontology based on MDE settings. Revue D Intell Artif 35:217–222. https://doi.org/10.18280/ria.350305

Cimiano P, Maedche A, Staab S et al (2009) Ontology learning. Springer, Berlin, Heidelberg, pp 245–267. https://doi.org/10.1007/978-3-540-92673-3_11

Codd EF (1970) A relational model of data for large shared data banks. Commun ACM 13(6):377–387. https://doi.org/10.1145/362384.362685

Dadjoo M, Kheirkhah E (2015) An approach for transforming of relational databases to owl ontology. International journal of Web & Semantic Technology 6. https://doi.org/10.5121/ijwest.2015.6102

DataMaster (2023) DataMaster-Protege Wiki. https://protegewiki.stanford.edu/wiki/DataMaster

d'Avila Garcez A, Lamb LC (2020) Neurosymbolic AI: the 3rd wave. arXiv: 2012.05876

Fensel D (2001) Ontologies. Ontologies: a silver bullet for knowledge management and electronic commerce. Springer, Berlin, pp 11–18. https://doi.org/10.1007/978-3-662-04396-7_2

Ghawi R, Cullot N (2007) Database-to-ontology mapping generation for semantic interoperability. In: Third international workshop on database interoperability (InterDB 2007)

Grüninger M, Fox MS (1995) The role of competency questions in enterprise engineering. Springer, Boston, pp 22–31. https://doi.org/10.1007/978-0-387-34847-6_3

Hazber M, Li R, Gu X et al (2016) Integration mapping rules: transforming relational database to semantic web ontology. Appl Math Inf Sci 10:881–901. https://doi.org/10.18576/amis/100307

He B, Patel M, Zhang Z et al (2007) Accessing the deep web. Commun ACM 50(5):94–101. https://doi.org/10.1145/1230819.1241670

Lakzaei B, Shamsfard M (2021) Ontology learning from relational databases. Inf Sci 577:280–297. https://doi.org/10.1016/j.ins.2021.06.074

Li M, Du XY, Wang S (2005) Learning ontology from relational database. In: 2005 International conference on machine learning and cybernetics, pp 3410–3415, vol 6. https://doi.org/10.1109/ICMLC.2005.1527531

Liu X, Gao F (2018) An approach for learning ontology from relational database. In: Proceedings of the 2018 international conference on algorithms, computing and artificial intelligence. Association for Computing Machinery, New York, NY, USA, ACAI'18. https://doi.org/10.1145/3302425.3302495

Ma C, Molnár B (2020) Use of ontology learning in information system integration: a literature survey. In: Sitek P, Pietranik M, Krótkiewicz M et al (eds) Intelligent information and database systems. Springer, Singapore, pp 342–353

Ma C, Molnár B (2022) Ontology learning from relational database: opportunities for semantic information integration. Vietnam J Comput Sci 9:31–57. https://doi.org/10.1142/S2196888821500024X

Maedche A, Staab S (2004) Ontology learning. Springer, Berlin, pp 173–190. https://doi.org/10.1007/978-3-540-24750-0_9

Mariano Fernández AGP, Juristo N (1997) Methontology: from ontological art towards ontological engineering. In: Papers from the 1997 AAAI spring symposium

Martinez-Cruz C, Blanco IJ, Vila MA (2012) Ontologies versus relational databases: are they so different? A comparison. Artif Intell Rev 38(4):271–290. https://doi.org/10.1007/s10462-011-9251-9

Mogotlane KD, Fonou-Dombeu JV (2016) Automatic conversion of relational databases into ontologies : a comparative analysis of protege plug-ins performances. Int J Web Seman Technol 7(3/4):21–40. https://doi.org/10.5121/ijwest.2016.7403

Noy NF, McGuinness DL (2001) Ontology development 101: A guide to creating your first ontology. Tech. rep., Stanford Knowledge

Systems Laboratory, http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html
OntoBase (2023) OntoBase-Protege Wiki. https://protegewiki.stanford.edu/wiki/OntoBase
Santoso HA, Haw SC, Abdul-Mehdi Z (2011) Ontology extraction from relational database: Concept hierarchy as background knowledge. Knowledge-Based Systems 24(3):457–464. https://doi.org/10.1016/j.knosys.2010.11.003
Sequeda JF, Tirmizi SH, Corcho O et al (2011) Survey of directly mapping SQL databases to the semantic web. Knowl Eng Rev 26(4):445–486. https://doi.org/10.1017/S0269888911000208
Stanford (2023) Protégé. https://protege.stanford.edu/
Sure Y, Staab S, Studer R (2004) On-to-knowledge methodology (OTKM). Springer, Berlin, pp 117–132. https://doi.org/10.1007/978-3-540-24750-0_6
Thuy PTT, Thuan ND, Han Y et al (2014) RDB2RDF: completed transformation from relational database into RDF ontology. In: Proceedings of the 8th international conference on ubiquitous information management and communication. Association for Computing Machinery, New York, NY, USA, ICUIMC'14. https://doi.org/10.1145/2557977.2558083
Uschold M, Gruninger M (2004) Ontologies and semantics for seamless connectivity. SIGMOD Rec 33(4):58–64. https://doi.org/10.1145/1041410.1041420
Uschold M, King M (1995) Towards a methodology for building ontologies. In: Workshop on basic ontological issues in knowledge sharing, held in conjunction with IJCAI-95
Vavliakis KN, Grollios TK, Mitkas PA (2013) RDOTE—publishing relational databases into the semantic web. J Syst Softw 86(1):89–99. https://doi.org/10.1016/j.jss.2012.07.018
W3C (2023) OWL-semantic web standards. https://www.w3.org/OWL/
Yiqing L, Lu L, Chen L (2012) Automatic learning ontology from relational schema. Proceedings, IEEE symposium on robotics and applications. ISRA 2012. https://doi.org/10.1109/ISRA.2012.6219258
Zhang L, Li J (2011) Automatic generation of ontology based on database. J Comput Inf Syst 7(4):1148–1154
Zhang H, Diao X, Yuan Z et al (2012) EVIS: a system for extracting and visualizing ontologies from databases with web interfaces. In: 2012 Fourth iternational symposium on information science and engineering, pp 408–411. https://doi.org/10.1109/ISISE.2012.98