Indonesian Journal of Electrical Engineering and Computer Science
Vol. 33, No. 2, February 2024, pp. 1116~1125
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v33.i2.pp1116-1125

Ontology learning from object-relational mapping metadata and relational database

Agus Sutejo[1,2], Rachmat Gernowo[1], Michael Andreas Purwoadi[2]
[1]School of Postgraduate Studies, Universitas Diponegoro (UNDIP), Semarang, Indonesia
[2]National Research and Innovation Agency Republic of Indonesia (BRIN), Jakarta, Indonesia

Article Info

Article history:

Received Oct 4, 2023
Revised Nov 27, 2023
Accepted Dec 13, 2023

Keywords:

Ontology
Ontology learning
ORM metadata
RDF
Relational database
Web ontology language

ABSTRACT

Ontologies play an important role in representing the semantics of data sources. Building an ontology as a representation of domain knowledge from available data sources is not a simple process, particularly when dealing with relational data, which remains prevalent in existing knowledge systems. In this study, we create an ontology from a relational database using object-relational mapping (ORM) metadata as additional rules for mapping. Our method comprises two main phases: ontology schema construction using ORM metadata and the generation of ontology instances from the relational database. During the initial phase, we analyzed the ORM metadata to map it to an resource description framework schema (RDF(S))-OWL representation of the ontology. In the subsequent phase, we applied mapping rules to convert the relational database (RDB) data into ontological instances, which are then represented as RDF triples. Using ORM metadata, we enhance the accuracy of the resulting ontology, particularly in terms of extracting concepts and hierarchical relationships. This study contributes to the field of ontology learning by showcasing a novel approach that leverages ORM metadata to create ontologies from relational databases.

Corresponding Author:

Agus Sutejo
School of Postgraduate Studies, Universitas Diponegoro (UNDIP)
50241 Semarang, Indonesia
Email: agussutejo@students.undip.ac.id

1. INTRODUCTION

Today, a widely used knowledge source is dynamic web content, where the information displayed originates from relational databases. Information systems within organizations also rely on relational databases to store data. Despite their inherent shortcomings, the enduring popularity of relational database management systems (RDBMS) can be attributed to extensive investments in applications built on these systems, their consistency and reliability for business data, and the existing skill sets of employees [1].

Building an ontology as a representation of domain knowledge from available data sources is not a simple process; however, most data in existing knowledge systems are still stored in relational form. The ontology development process based on the available data sources can be categorized into manual and automatic/semi-automatic methods. Manual methods involve manually performing each phase of the development, which is time-consuming and prone to errors [2]. On the other hand, semi-automatic or automatic methods perform almost all steps automatically, a process often referred to as ontology learning. A relational database (RDB) is structured on the basis of a relational model, employing tables to depict data and its relationships. The relational model is a conceptual framework for organizing and manipulating data. It is based on the principle of representing data as relations or tables. Each table in an RDB consists of rows and columns, with rows representing individual records or instances and columns representing attributes or properties [3]. Relational databases are a common way to structure domain data, and their schemas reflect domain characteristics [4]. Generating an ontology from such a database offers an advantage by effectively expressing domain characteristics, as database tables align with domain data concepts. This process becomes crucial in the context of the semantic web, a vital field in recent research enabling computers to process web information and transform it into a medium for data sharing, understanding, and automation [5].

Ontology, a key technology in the semantic web, provides a structured framework for defining and representing concepts, relationships, and categories of entities. This structured representation serves as the backbone for enhanced comprehension and processing of information across diverse domains. The world wide web consortium (W3C) recommends formats such as resource description framework (RDF), RDF schema (RDFS), and web ontology language (OWL) to formally describe concepts, terms, and relationships, facilitating automated reasoning within specific domains.

In various sectors such as healthcare, education, and tourism, ontology plays a crucial role as a foundational element supporting custom solutions. Numerous studies have underscored the significant impact of ontology on improving predictive models and personalized suggestions. For instance, in healthcare, ontology enhances the accuracy of predicting Covid-19 symptoms, surpassing the capabilities of machine learning algorithms [6]. In education, ontology-powered adaptive learning systems can effectively customize educational materials to suit individual learner conditions, encompassing mental states and social contexts [7]. Similarly, in the tourism domain, integrating ontological knowledge bases with supervised learning models enhances the performance of recommendation systems, enabling seamless cooperation between domain expertise and machine learning for more efficient suggestions [8].

Ontology learning from relational databases can be categorized into three main methods: reverse engineering, mapping, and machine learning. Reverse engineering involves transforming the logical model of an RDB into a richer conceptual model. Mapping methods include rule-based, graph-based, and similarity-based approaches. Machine learning for ontology learning is a more recent development with various algorithms and tools, although it has been predominantly used for text-based ontology learning [9].

Several studies have investigated methodologies for mapping relational databases to semantic web ontologies using various approaches. Hazber *et al.* [10] proposed a two-phase method involving the construction of an ontology schema based on the RDB schema and the extraction of ontology instances from RDB data using mapping rules. An and Park [11] introduced an approach to generate an ontology model from database metadata, translating it into database tuples, particularly handling multiple individuals within the ontology.

Fabro *et al.* [12] emphasized the use of logical database metadata and schema mapping rules to enhance ontology readability and naming conventions. Lakzaei and Shmasfard [2] expanded schema mapping rules by incorporating extraction from stored procedures, user-defined functions, and views in their ontology generation process. Mahria *et al.* [13] proposed a comprehensive lifecycle for ontology learning with stages such as discovery, preparation, and development, introducing transformation rules such as check constraints and inheritance increment constraints.

Kaulins and Borisov [14] introduced a methodology based on mapping rules for ontology construction, not constrained by a particular database system, as it adheres to internationally recognized standards for data management, specifically SQL99. Lin *et al.* [15] employed reverse engineering techniques to automate OWL ontology creation from relational databases, focusing on conceptual correspondences between the RDB schema and the OWL-description logic (OWL-DL) ontology. Bouougada *et al.* [16] presented a model-driven engineering approach to transform traditional web applications into semantic ones, involving phases such as generating input models from structured query language (SQL) databases and converting them into ontologies.

Aggoune [17] outlined an automated process for ontology learning and evolution from relational databases, encompassing key steps such as generating classes, datatype properties, and object properties based on table relationships. Louhdi and Behja [18] proposed methods to convert recursive relationships in relational databases to OWL2 ontology components using transitivity for object properties or creating a class hierarchy based on table occurrences. Dadjoo and Kheirkhah [19] introduced a method that employs a transition system and graph theory to transform relational databases into OWL-based ontology models, ensuring semantic richness and independence from the physical structure of the database.

The studies mentioned primarily rely on pure RDB sources, overlooking the prevalent adoption of object-relational mapping (ORM) in modern business applications. RDB structures often result from ORM tool generation, leading to a loss of certain semantic details. ORM tools abstract the database structure, potentially altering or simplifying the underlying relational schema, which could impact the fidelity of information and the representation of relationships between entities in the database.

In contemporary computer science, two prominent paradigms coexist: object-oriented programming (OOP) and RDBMS. OOP is based on principles such as encapsulation, abstraction, inheritance, and polymorphism, which are not fully supported by the tabular and relational nature of DBMSs. While association and aggregation can be directly mapped to relational concepts, the mapping of inheritance is more complex. Various strategies have been documented for implementing the semantics of inheritance relationships in relational databases [20]–[23]. This fundamental difference leads to challenges when using both approaches together. To bridge this gap, modern applications often employ ORM layers, which automate the mapping process and abstract technical details.

ORM is a comprehensive term that encompasses the principles and procedures employed to establish connections between object-oriented programming language classes and relational database tables [24]. The use of an ORM provides convenience and speeds up application development without having to worry about how objects are persistently stored in the database. It has become popular in system applications. Enterprise applications typically use object-oriented technologies with relational databases for persistence. ORM layers or middleware automate the mapping between objects and tables, simplifying data type and relationship mapping [25].

This study focuses on the central theme of ontology learning from relational databases, with a particular focus on harnessing ORM metadata to enhance the accuracy and comprehensiveness of the resulting ontologies. Our goal is to bridge the gap between the OOP paradigm and relational databases by leveraging ORM metadata to generate RDF graphs. This approach not only seeks to overcome the challenges arising from the differing data paradigms but also aims to facilitate a more seamless integration between object-oriented structures and the RDB model through the enriched representation provided by ORM metadata. We analyze and design schema mapping rules using ORM metadata and develop software to generate RDF graphs. To measure accuracy, the resulting ontology is compared to a reference ontology created in consultation with domain experts. This method evaluates the generated ontology by measuring its accuracy, recall, and F-measure against the reference ontology.

In the subsequent sections, we will delve into the methodology of our study, which includes schema mapping rules using ORM metadata. We will also detail the software development process for generating RDF graphs. Importantly, we will measure the accuracy of the resulting ontology and compare it to a reference ontology, providing a quantitative assessment of the effectiveness of our approach.

2. METHOD

We present an approach for the automated creation of an ontology from an RDB, relying on ORM metadata. Figure 1 illustrates this approach using ORM definition files and a relational database as inputs. We leverage ORM metadata as a source for ontology schema extraction to mitigate the semantic loss caused by databases which are generated outputs from ORM tools. Our methodology comprises three major steps: construction and analysis of the abstract syntax tree (AST), extraction of OWL-ontology components, and extraction of instances from the RDB.

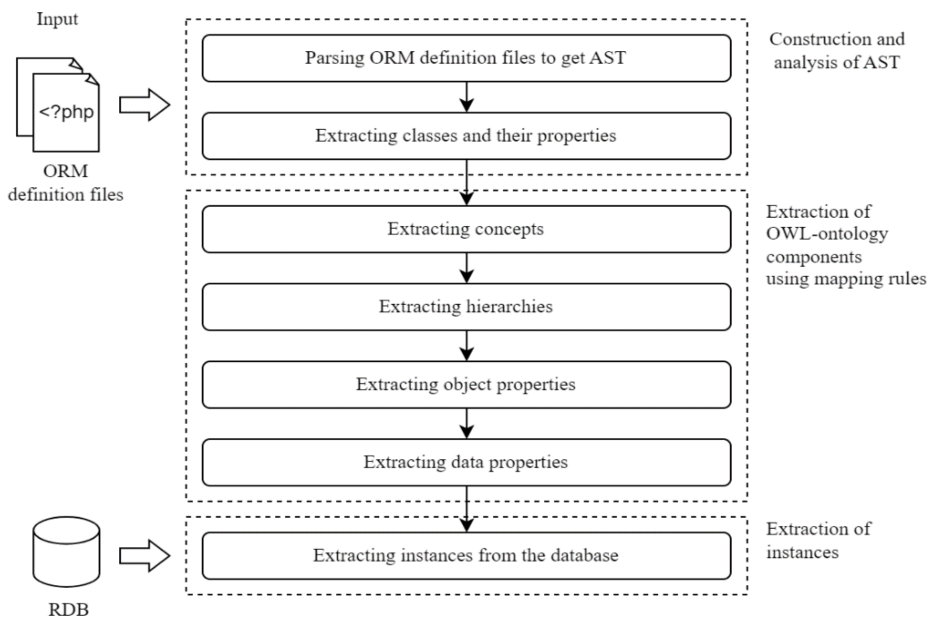

Input
<?php
ORM definition files
Parsing ORM definition files to get AST
Extracting classes and their properties
Construction and analysis of AST
Extracting concepts
Extracting hierarchies
Extracting object properties
Extracting data properties
Extraction of OWL-ontology components using mapping rules
RDB
Extracting instances from the database
Extraction of instances

Figure 1. Stages of ontology learning using ORM metadata and RDB

For the experimentation, we used the source code of the SIMRAL Application written in PHP. Our approach was implemented using the programming language PHP and a MySQL database. We use Protégé, a free and open-source ontology editor developed by Stanford University, to view and edit the generated ontology.

2.1. Construction and analysis of the AST

The initial step involves constructing an AST from the provided ORM definition files. A parser dissects these files to form a structured AST, capturing essential information such as class names, attributes, and associated annotations. Annotations provide basic information about the mapping of classes and attributes to database tables. The source code of the application, which is available for experimentation, is written in the PHP programming language and utilizes doctrine ORM. We employed nikic/PHP-Parser [26] to parse the source code into an AST, represented in the form of an array or JSON. This parser offers the ability to traverse the AST, simplifying the analysis of its structure and content.

After the formation of the AST, a content analysis is performed using the traverse method of the parser. In the context of ontological schema creation, we focus on nodes categorized as 'Stmt_Class' and their corresponding sub-nodes labeled as 'Stmt_Property'. To streamline subsequent stages, we extract and store information from these specific node types into two tables. From the 'Stmt_Class' nodes, relevant data, such as class names, parent classes, and comments in the form of docblock annotations, are captured and stored. Similarly, the information extracted from 'Stmt_Property' nodes contains property names along with docblock annotations. This approach serves as a foundational step for our next stages.

2.2. Extraction of the OWL-ontology elements

In contrast to other methods that use database schemes as their mapping source, our approach involves the development of mapping from ORM metadata. This metadata can be retrieved from the AST to map to ontology components. Doctrine ORM offers various methods for specifying metadata, including attributes, XML, PHP code, docblock annotations, and YAML [27]. Our source code for experimentation using docblock annotations will serve as the foundation for exploring the mapping rules in the following subsection. At this stage, docblock annotations associated with class and property statements are analyzed. We parse the docblock annotations to extract the metadata required for mapping to the OWL ontology using the data stored in the tables from the previous phase. In the following subsection, the mapping rules for this process are elaborated.

2.2.1. Extracting concepts

Every class definition in the ORM will be mapped as an OWL-Class. Doctrine ORM defines each object that will be stored in the database as an ‘Entity’. As depicted in Table 1, PHP class definitions marked as Entity will be mapped as OWL-classes. Additionally, in Table 2, we illustrate the extraction of ORM Entity to the OWL representation.

Table 1. Mapping doctrine annotation to OWL elements

Doctrine ORM annotation	OWL Element
Entity	OWL class
Column	DataType property
OneToOne	Object property
OneToMany	Object property
ManyToOne	Object property
ManyToMany	Object property

Table 2. Extraction of the ORM entity to OWL

Doctrine ORM annotation	OWL
Entity	`<owl:Class rdf:about="http://www.simral.id/rapbd#RapbdRapbd"> <rdfs:label>RapbdRapbd</rdfs:label> </owl:Class>`

2.2.2. Extracting hierarchies

Information regarding the hierarchy among classes can be obtained by combining two methods: analyzing the AST and leveraging doctrine ORM metadata within the class definition. The first method provides insights into the structure and organization of classes within the codebase. Furthermore, doctrine ORM metadata, comprising annotations such as InheritanceType reveals the employed inheritance mapping strategy, whether single-table or class-table inheritance. Table 3 illustrates the hierarchy extraction process mapped to OWL representation.

Table 3. Hirarchie extraction to OWL

AST token	OWL
Extends	`<owl:Class rdf:about="http://www.simral.id/rapbd#RkaSkpd">` `<rdfs:label>RkaSkpd</rdfs:label>` `<rdfs:subClassOf rdf:resource="http://www.simral.id/rapbd#RkaRka"/>` `</owl:Class>`

2.2.3. Extracting the object properties

The OWL ontology has two types of properties: object properties and data type properties. Object properties represent relationships between individuals (objects), whereas datatype properties are used to assign literal values with specific data types, such as numbers or strings, to individuals. This distinction is fundamental for accurately modeling ontology concepts and their relationships.

The doctrine annotations employed for mapping object properties include OneToOne, OneToMany, ManyToOne, and ManyToMany, as outlined in Table 1. These annotations are mapped to the OWL ontology to define the cardinality of the relationship between concepts. Table 4 illustrates the extraction of ORM relations to OWL, exemplifying how specific relationships, like 'OneToMany,' are mapped as OWL Object Properties. As seen here, utilizing ORM metadata makes it easier to identify many-to-many relationships compared to having to detect intermediary tables within such relationships.

Table 4. Extraction object properties from the ORM to OWL

Doctrine ORM annotation	OWL
Column, OneToMany	`<owl:ObjectProperty rdf:about="http://www.simral.id/has.RkaMataAnggaran">` `<rdfs:label>has.RkaMataAnggaran</rdfs:label>` `<rdfs:range rdf:resource="http://www.simral.id/rapbd#RkaMataAnggaran"/>` `<rdfs:domain rdf:resource="http://www.simral.id/rapbd#RkaRka"/>` `</owl:ObjectProperty>` `<owl:Restriction>` `<owl:onProperty rdf:resource="http://www.simral.id/has.RkaMataAnggaran"/>` `<owl:someValuesFrom rdf:resource="http://www.simral.id/rapbd#RkaMataAnggaran"/>` `</owl:Restriction>`

2.2.4. Extracting the datatype properties

Each doctrine annotation, represented as a 'Column' as shown in Table 1, is mapped as a datatype property in the OWL ontology. Datatype mapping involves translating doctrine datatypes into XML schema definition (XSD) datatypes, as illustrated in Table 5. The mapping shows the conversion of Doctrine datatypes such as string to String, integer to Integer, and others to their respective XSD counterparts. Table 6 illustrates the process of transferring ORM columns into the OWL ontology, showcasing the mapping of individual columns as OWL datatype properties.

Table 5. Mapping doctrine datatypes to XSD datatypes

Doctrine datatype	XSD datatype
String	String
Integer	Integer
Smallint	Short
Bigint	Long
Boolean	Boolean
Decimal	Decimal
Date	Date
Time	Time
Datetime	DateTime
Text	String

Table 6. Extraction of ORM column to OWL

Doctrine ORM annotation	OWL

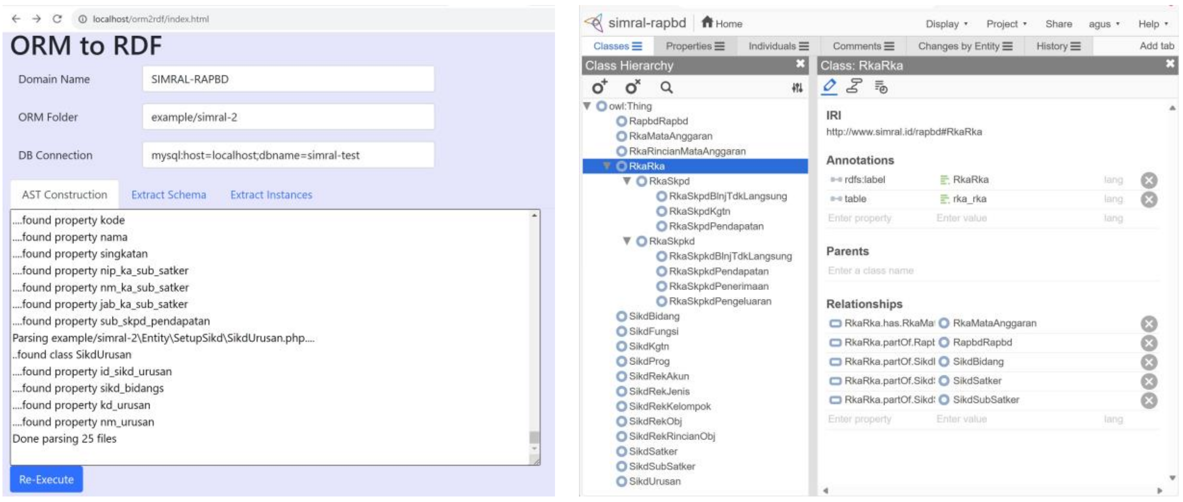
Column	`<owl:DatatypeProperty rdf:about="http://www.simral.id/RkaRka.no_rka">`
	    `<rdfs:label>RkaRka.no_rka</rdfs:label>`
	    `<rdfs:domain rdf:resource="http://www.simral.id/rapbd#RkaRka"/>`
	    `<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>`
	    `<orm:column>no_rka</orm:column>`
	`</owl:DatatypeProperty>`

2.3. Extracting instances from the database

After the ontology schema is extracted, the next step is to transform the data from the relational database according to the above-mentioned mapping rules. The aim is to extract instances from rows in the relational database tables. Based on the ORM metadata, a mapping is derived that identifies where the data of a class persists within the database table and which columns will be extracted as the values of data properties.

Subsequently, RDF triples are formed for each class and properties derived from the database tables to represent the data in a structured format. These triples consist of subject-predicate-object components, where the subject denotes the class or entity, the predicate signifies the property, and the object represents the specific value in the database. This process aligns with the conversion of the relational database content into a linked data format, ensuring that each piece of information is organized and represented according to the RDF data model.

Figure 2 is the application used in the experiment, as depicted in Figure 2(a), we developed an application to execute the aforementioned three major processes. The main input of this application is the folder containing the ORM definition files and the database connection string. The resulting ontology output can be exported to a file using the XML/RDF syntax. To visualize and validate the generated ontology, we imported this file into the Protégé application, as shown in Figure 2(b).



Domain Name: SIMRAL-RAPBD
ORM Folder: example/simral-2
DB Connection: mysql:host=localhost;dbname=simral-test

```
....found property kode
....found property nama
....found property singkatan
....found property nip_ka_sub_satker
....found property nm_ka_sub_satker
....found property jab_ka_sub_satker
....found property sub_skpd_pendapatan
Parsing example/simral-2\Entity\SetupSikd\SikdUrusan.php....
..found class SikdUrusan
....found property id_sikd_urusan
....found property sikd_bidangs
....found property kd_urusan
....found property nm_urusan
Done parsing 25 files
```

```
owl:Thing
    RapbdRapbd
    RkaMataAnggaran
    RkaRincianMataAnggaran
    RkaRka
        RkaSkpd
            RkaSkpdBlnjTdkLangsung
            RkaSkpdKgtn
            RkaSkpdPendapatan
        RkaSkpkd
            RkaSkpkdBlnjTdkLangsung
            RkaSkpkdPendapatan
            RkaSkpkdPenerimaan
            RkaSkpkdPengeluaran
    SikdBidang
    SikdFungsi
    SikdKgtn
    SikdProg
    SikdRekAkun
    SikdRekJenis
    SikdRekKelompok
    SikdRekObj
    SikdRekRincianObj
    SikdSatker
    SikdSubSatker
    SikdUrusan
```

IRI
http://www.simral.id/rapbd#RkaRka

Annotations

rdfs:label	RkaRka	lang
table	rka_rka	lang

Parents

Relationships

RkaRka.has.RkaMa	RkaMataAnggaran
RkaRka.partOf.Rapt	RapbdRapbd
RkaRka.partOf.Sikdl	SikdBidang
RkaRka.partOf.Sikd:	SikdSatker
RkaRka.partOf.Sikd:	SikdSubSatker

(a) (b)

Figure 2. Application used in the experimentation to (a) extract ontology and (b) visualize ontology

3. RESULTS AND DISCUSSION

We have implemented our methodology to extract an ontology from the SIMRAL application, which supports regional budgeting and financial processes in Indonesia. The development of this application followed an object-oriented design approach and used the doctrine ORM. The database structure was deployed using doctrine's migration tools. To enhance performance, the application employs single-table inheritance for mapping. However, this choice can present challenges when attempting to extract class properties in the absence of ORM metadata assistance.

Within the SIMRAL application, there are over 100 ORM definition files; however, for our experimentation, we specifically focused on extracting data from the budgeting module. We chose to concentrate solely on this module because of the extensive resources required for creating the reference ontology. For our analysis, we narrowed down our scope to 25 ORM definition files, each representing a distinct class within the budgeting module. The database tables used in this module are shown in Figure 3.

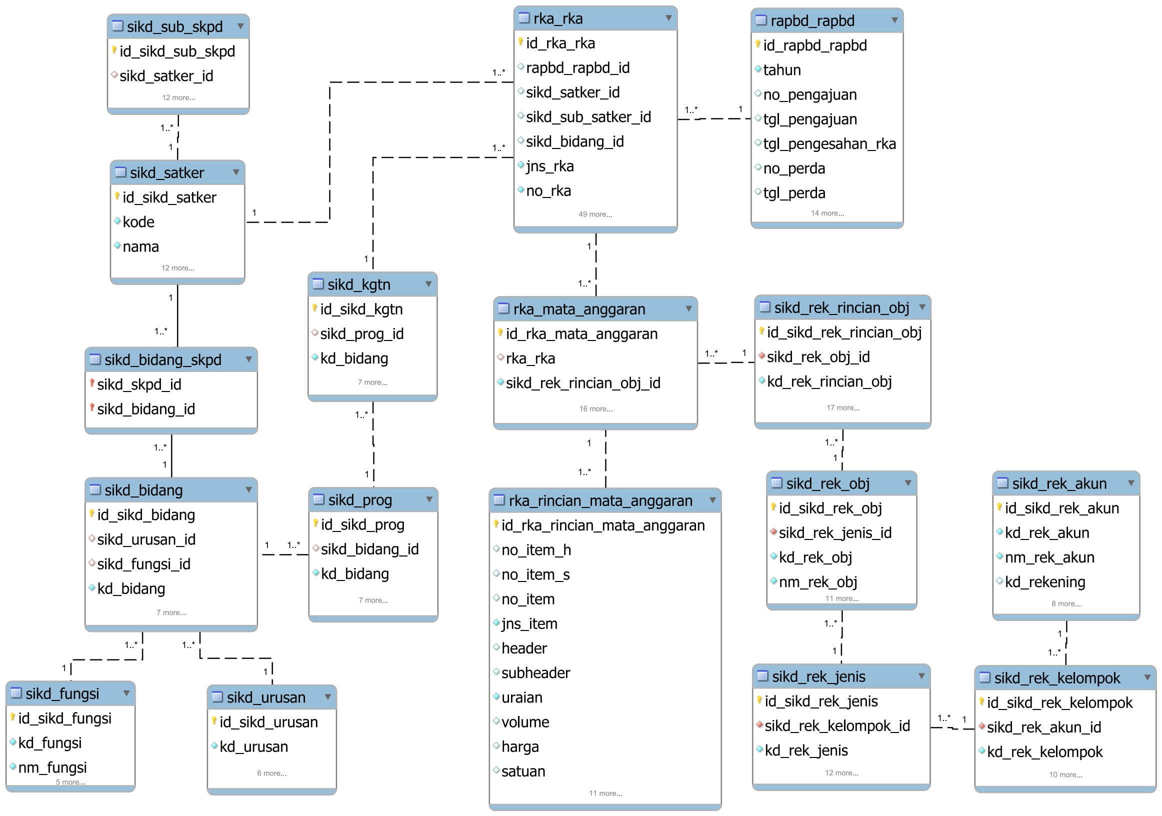


Figure 3. Relational database schema used in the experiments

The resulting ontology contains 25 concepts/classes, 29 object properties, 162 datatype properties, and 193 axioms. Visualization of this ontology is shown in Figure 4. The visualization highlighted a clear taxonomy of classes and relationships, providing a visual confirmation of the accuracy and interlinkages derived from the ORM metadata. For instance, it visually showcased the inheritance hierarchies that would have been otherwise challenging to infer solely from the data comparison.

3.1. Comparison

We compared our method with existing methods by assessing ontology elements such as concepts, hierarchical and non-hierarchical relationships, axioms, and instances. Additionally, we compared the input data utilized and the syntax of the resulting ontology, whether it is in RDF or OWL format [2]. Table 7 lists the results of this comparison.

3.2. Evaluation

To evaluate the generated ontologies, we employ a gold standard, which is a reference ontology crafted by a knowledge engineer. The resulting ontology was measured against the reference ontology using three metrics: precision, recall, and the F-measure [28]. Precision is quantified as the ratio of the number of true positives (|R∩A|) to the total number of correspondences retrieved (|A|).

$$Precision(A,R) = \frac{|R \cap A|}{|A|} \quad (1)$$

Recall is specified as the ratio of the number of true positives (|R∩A|) and those to be retrieved (|R|).

$$Recall(A,R) = \frac{|R \cap A|}{|R|} \quad (2)$$

The F-measure is a metric that strikes a balance between precision and recall and provides a single value that summarizes the overall performance of a classification or matching system.

$$F\ Measure(A,R) = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Compared with the reference ontology, we obtained the results depicted in Figure 5. It is evident that the generated ontology exhibits an F-measure of 0.91 for concept extraction and 0.9 for hierarchy extraction, despite the database storing data for object inheritance using a one-table strategy, which would pose challenges for other methods.

Our methodology encountered challenges due to the application's use of single-table inheritance for mapping within the doctrine ORM. While this strategy optimizes performance within the SIMRAL application, it presents hurdles during the ontology extraction process. Specifically, when attempting to extract class properties from the database without direct assistance from ORM metadata, the process became intricate and required meticulous analysis.

For instance, consider the class inheritance structure, where ORM's single-table inheritance leads to multifaceted relationships between entities. In these cases, the extraction process necessitated additional scrutiny to accurately define the hierarchy and relationships among various classes. This complexity highlights the importance of leveraging ORM metadata for a more streamlined and precise ontology generation process.

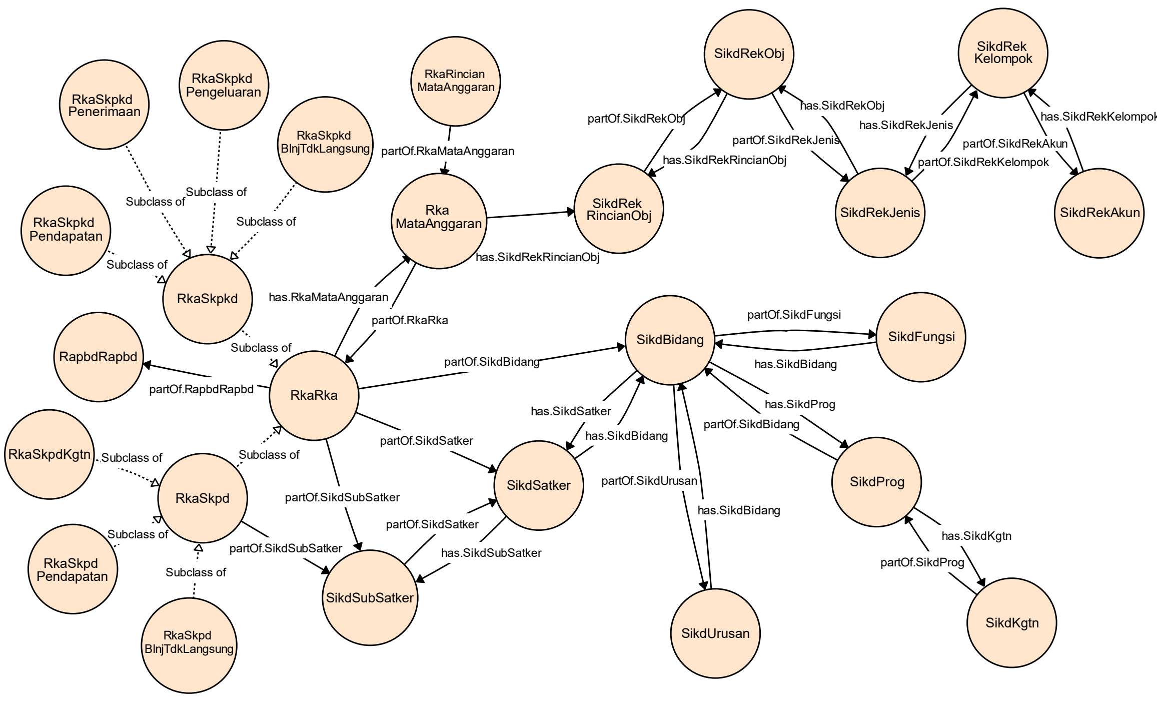


Figure 4. Visualization of the generated ontology using WebVOWL

Table 7. Comparative assessment of our approach and other existing approaches

Approaches	Ontology language	Data source	Learning elements				
			Concepts	Hierarchical relations	Non-hierarchical relations	Axioms	Instances
Hazber *et al.* [10]	RDF	RDB	√	-	√	√	√
Fabro *et al.* [12]	OWL	RDB	√	√	√	√	√
Kaulins and Borisov [14]	OWL	RDB	√	√	√	√	√
Lin *et al.* [15]	OWL	RDB	√	√	√	√	-
Our approach	OWL	RDB and ORM source code	√	√	√	√	√

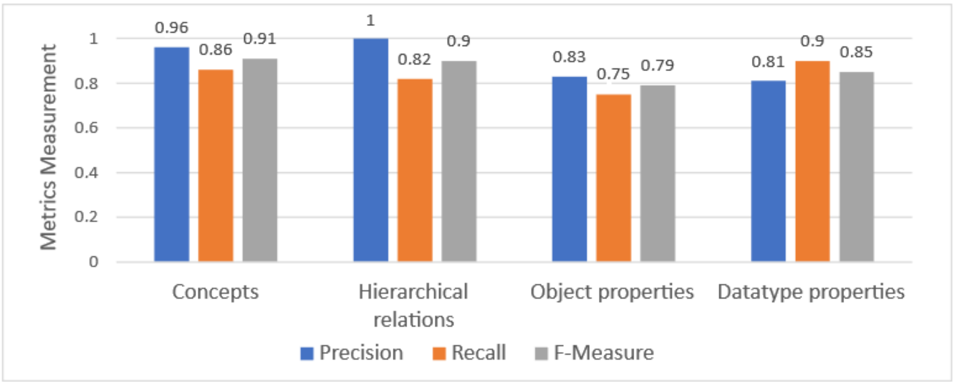


Figure 5. Ontology learning performance measurement

4. CONCLUSION

This research aims to propose a new method for ontology learning by leveraging ORM metadata as a complement alongside RDB. Due to the gap between object-based and relational data paradigms, the extraction of concepts or classes and their inheritance poses a huge challenge without the assistance of an ORM. This synergy between ORM metadata and RDB structures not only resolves the complexities inherent in class extraction but also enhances the accuracy of the resultant ontology, crucial for robust knowledge representation. Although this method has the limitation of requiring access to the application's source code, it holds promise for enhancing the accuracy of ontology learning from RDB. Moreover, this method can be applied to extract ontologies from in-house developed applications or open-source applications. Future research directions could focus on exploring the adaptability of this method to various application architectures and ORM tools would broaden its applicability. Practically, our proposed methodology holds immense promise in domains requiring highly accurate ontologies, such as healthcare or finance, where precise representation of domain knowledge is critical. For instance, in finance applications such as SIMRAL, where the accuracy of financial processes relies on well-defined concepts and relationships, our methodology proves its strength. Similarly, in healthcare or scientific domains requiring meticulous ontological representations, leveraging ORM metadata could significantly enhance the accuracy and comprehensiveness of extracted knowledge.

REFERENCES

[1] P. Atzeni, C. S. Jensen, G. Orsi, S. Ram, L. Tanca, and R. Torlone, "The relational model is dead, SQL is dead, and i don't feel so good myself," *ACM SIGMOD Record*, vol. 42, no. 2, pp. 64–68, Jun. 2013, doi: 10.1145/2503792.2503808.

[2] B. Lakzaei and M. Shamsfard, "Ontology learning from relational databases," *Information Sciences*, vol. 577, pp. 280–297, Oct. 2021, doi: 10.1016/j.ins.2021.06.074.

[3] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970, doi: 10.1145/362384.362685.

[4] S. L. Osborn and T. E. Heaven, "The design of a relational database system with abstract data types for domains," *ACM Transactions on Database Systems*, vol. 11, no. 3, pp. 357–373, Aug. 1986, doi: 10.1145/6314.6461.

[5] J. G. Breslin, D. O'Sullivan, A. Passant, and L. Vasiliu, "Semantic web computing in industry," *Computers in Industry*, vol. 61, no. 8, pp. 729–741, Oct. 2010, doi: 10.1016/j.compind.2010.05.002.

[6] H. El Massari, N. Gherabi, S. Mhammedi, H. Ghandi, F. Qanouni, and M. Bahaj, "Integration of ontology with machine learning to predict the presence of covid-19 based on symptoms," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2805–2816, Oct. 2022, doi: 10.11591/eei.v11i5.4392.

[7] A. Ouatiq, K. ElGuemmat, K. Mansouri, and M. Qbadou, "A design of a multi-agent recommendation system using ontologies and rule-based reasoning: pandemic context," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 515–523, Feb. 2022, doi: 10.11591/ijece.v12i1.pp515-523.

[8] H. Q. Dung, L. T. Quynh Le, N. H. H. Tho, T. Q. Truong, and C. H. Nguyen-Dinh, "A novel ontology framework supporting model-based tourism recommender," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 1060–1068, Dec. 2021, doi: 10.11591/ijai.v10.i4.pp1060-1068.

[9] C. Ma and B. Molnár, "Ontology learning from relational database: opportunities for semantic information integration," *Vietnam Journal of Computer Science*, vol. 09, no. 01, pp. 31–57, Feb. 2022, doi: 10.1142/S219688882150024X.

[10] M. A. G. Hazber, R. Li, X. Gu, and G. Xu, "Integration mapping rules: transforming relational database to semantic web ontology," *Applied Mathematics & Information Sciences*, vol. 10, no. 3, pp. 881–901, May 2016, doi: 10.18576/amis/100307.

[11] J. An and Y. B. Park, "Methodology for automatic ontology generation using database schema information," *Mobile Information Systems*, vol. 2018, pp. 1–13, 2018, doi: 10.1155/2018/1359174.

[12] M. D. D. Fabro, L. M. Peres, H. Tissot, and C. A. G. Huve, "Exploring logical and hierarchical information to map relational databases into ontologies," *International Journal of Metadata, Semantics and Ontologies*, vol. 13, no. 3, p. 191, 2019, doi: 10.1504/IJMSO.2019.10021447.

[13] B. Ben Mahria, I. Chaker, and A. Zahi, "A novel approach for learning ontology from relational database: from the construction to the evaluation," *Journal of Big Data*, vol. 8, no. 1, p. 25, Dec. 2021, doi: 10.1186/s40537-021-00412-2.
[14] A. Kaulins and A. Borisov, "Building ontology from relational database/ ontoloģiju izveide no relāciju datubāzes/ Построение онтологии по реляционной базе данных," *Information Technology and Management Science*, vol. 17, no. 1, Jan. 2014, doi: 10.1515/itms-2014-0006.
[15] L. Lin, Z. Xu, and Y. Ding, "OWL ontology extraction from relational databases via database reverse engineering," *Journal of Software*, vol. 8, no. 11, pp. 2749–2760, Nov. 2013, doi: 10.4304/jsw.8.11.2749-2760.
[16] B. Bouougada *et al.*, "Mapping relational database to OWL ontology based on MDE Settings," *Revue d'Intelligence Artificielle*, vol. 35, no. 3, pp. 217–222, Jun. 2021, doi: 10.18280/ria.350305.
[17] A. Aggoune, "Automatic Ontology learning from heterogeneous relational databases: application in alimentation risks field," 2018, pp. 199–210.
[18] M. R. C. Louhdi and H. Behja, "Ontology Learning from relational databases: transforming recursive relationships to OWL2 components," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 10, 2019, doi: 10.14569/IJACSA.2019.0101037.
[19] M. Dadjoo and E. Kheirkhah, "An approach for transforming of relational databases to OWL Ontology," *International journal of Web & Semantic Technology*, vol. 6, no. 1, pp. 19–28, Jan. 2015, doi: 10.5121/ijwest.2015.6102.
[20] S. Holder, J. Buchan, and S. G. MacDonell, "Towards a metrics suite for object-relational mappings," in *Model-Based Software and Data Integration*, 2008, pp. 43–54, doi: 10.1007/978-3-540-78999-4_6.
[21] L. Cabibbo and A. Carosi, "Managing inheritance hierarchies in object/relational mapping tools," in *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 2005, vol. 3520, pp. 135–150, doi: 10.1007/11431855_11.
[22] M. Fowler, *Patterns of enterprise application architecture*. MA, USA: Addison-Wesley Professional, 2002.
[23] A. Torres, R. Galante, M. S. Pimenta, and A. J. B. Martins, "Twenty years of object-relational mapping: a survey on patterns, solutions, and their implications on application design," *Information and Software Technology*, vol. 82, pp. 1–18, Feb. 2017, doi: 10.1016/j.infsof.2016.09.009.
[24] M. Lorenz, G. Hesse, and J.-P. Rudolph, "Object-relational mapping revised - a guideline review and consolidation," in *Proceedings of the 11th International Joint Conference on Software Technologies*, 2016, pp. 157–168, doi: 10.5220/0005974201570168.
[25] M. Lorenz, J.-P. Rudolph, G. Hesse, M. Uflacker, and H. Plattner, "Object-relational mapping revisited - a quantitative study on the impact of database technology on O/R mapping strategies," 2017, doi: 10.24251/HICSS.2017.592.
[26] N. Popov, "PHP-parser," *GitHub*, 2023. https://github.com/nikic/PHP-Parser (accessed Sep. 14, 2023).
[27] "Welcome to doctrine 2 ORM's documentation!," *doctrine-project.org*. https://www.doctrine-project.org/projects/doctrine-orm/en/2.16/index.html (accessed Sep. 14, 2023).
[28] M. Ehrig and J. Euzenat, "Relaxed precision and recall for ontology matching," *CEUR Workshop Proceedings*, vol. 156, pp. 25–32, 2005.

BIOGRAPHIES OF AUTHORS

Agus Sutejo is a postgraduate student at Diponegoro University, Indonesia and researcher at the Research Center for Artificial Intelligence and Cyber Security at the National Research and Innovation Agency in Indonesia. He has research experience in regional government financial information systems, cloud computing, semantic web, and knowledge management systems. His research interests include database management, semantic web, blockchain, and IoT. He can be contacted at email: agussutejo@students.undip.ac.id.

Rahmat Gernowo is student in doctoral education at the Bandung Institute Technology, Indonesia, majoring in Geo Physics in 2009. Research area in the field of geo physics and atmosfer science, modelling hazard information system. He can be contacted at email: gernowo@yahoo.com.

Michael Andreas Purwoadi is a principal engineer at the Research Center for Electronics at the National Research and Innovation Agency in Indonesia. He has experience on Information Systems for National Transportation, Information Systems for National General Election, and on producing regulations on National Digital Government and National Artificial Intelligent Strategy. His research interests include operational research, industrial automation, power electronics and cable-based underwater information network. He can be contacted at: michael.andreas.purwoadi@brin.go.id.