Ontology to relational database transformation for web application development and maintenance

Kamal Mahmudi, M M Ingriani Liem, Saiful Akbar
School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132, Indonesia
23514050@std.stei.itb.ac.id, inge@informatika.org, saiful@informatika.org

Abstract. Ontology is used as knowledge representation while database is used as facts recorder in a KMS (Knowledge Management System). In most applications, data are managed in a database system and updated through the application and then they are transformed to knowledge as needed. Once a domain conceptor defines the knowledge in the ontology, application and database can be generated from the ontology. Most existing frameworks generate application from its database. In this research, ontology is used for generating the application. As the data are updated through the application, a mechanism is designed to trigger an update to the ontology so that the application can be rebuilt based on the newest ontology. By this approach, a knowledge engineer has a full flexibility to renew the application based on the latest ontology without dependency to a software developer. In many cases, the concept needs to be updated when the data changed. The framework is built and tested in a spring java environment. A case study was conducted to proof the concepts. Keywords—ontology to database transformation, converter, application generator.

1. Introduction
The multilayered architecture approach is one of best practices in application development. Separating domain, infrastructure and user interface layers give us a cleaner and easier application to be maintained. Each layer has its responsibility and responds to change with different need and speed. Separating domain layer from other layers also helps a reengineering process in the future development [1].

Ontology as explicit specification of conceptualization [2] can be used as domain layer representation. Ontology has been used to solve many problems with incomplete, complex or evolved knowledge representation. Some researches mapped knowledge domain represented in an ontology to an object model represented as a database. But, methods, procedures and tools for application developers to develop and maintain ontology-based application are still limited [3]. Ontology as a basis of a domain model is important in the application development process, and it can be combined with relational database [4]. As another component of application, ontology is also prone to evolution. Especially as the backbone of the application, change in the conceptual model will lead to change in the requirement, and then to change the model domain [5].

This research harvests methods and tools for data transformation to ontology and code generators in order to create an updated system that shall generate a web application from the ontology. The system shall be able to adapt to a new conceptual model as well as the last updated data.
2. Problem Statement and Proposed Solution
Database to ontology transformation and ontology to database transformation have been reported in many researches. Some of them used a migration rule and mapping [4][8][9][10][11]. However, the concept evolves and the development of ontology can still occur even after it’s used to build the application. Hence, there is a need to implement a system that is not only able to transform ontology to relational database, but also be able to generate the application from the ontology. And when a change happened to the database or ontology, the system shall synchronize the change and regenerate the application.

This research proposes a set of tools to transform ontology component for database and vice versa, generate application code, build and run the application. All processes are being done automatically within the framework, by pushing a button. The main function of the generated application is to manage the data that ontology has. But since the generated source code is editable, developer can modify it to add his sophisticated functional that he needs.

3. Related Works

3.1. Ontology
Ontology can be defined as explicit specification of conceptualization [2]. Ontology in a program can be illustrated as the definition of all representation of that program including classes and relationships. Ontology has no standard format, it can have different structures depend on the language it used.

Generally, an ontology consists of classes, attributes, individual/instances, relations, rules, and axioms [6]. These components can be coded in XML based ontology language, such as DARPA Agent Mark-up Language + Ontology Interchange Language (DAML+OIL), Resource Description Framework (RDF), or Web Ontology Language (OWL). Classes can be defined and organized in a hierarchical structure using OWL. OWL consists of individuals, properties, and classes [7].

The individual represents object from the domain of discourse. The individual is also known as instance, or instance of a class. The properties represent the relationship that an individual has. Properties can be object or datatype properties. Object properties represent the relationship between individuals while datatype properties represent the relationship between individual and XML datatype (integer, boolean, string, etc.)

The properties can be either functional, transitive, or symmetric. Functional property means that each object has only one value. An example of this property is HasBirthMother which every human only has one birth mother.

A property can be a transitive property, that is if property P connects individual A and B, and connects individual B and C, then P also connects individual A and C. HasAncestor is one of many examples of this type of property. A symmetric property connects individual A to B, the same way it connects individual B to A. A perfect example of this relationship is HasSibling relationship.

Some researches have proposed algorithms, mappings, and rules for transforming an ontology to a relational database [4][8] and from relational databases to ontology [9][10][11]. Some studies combine both technologies, for example, using a relational database for storing ontology [12], using a relational database as data source in reasoning [13].

3.2. Relational Database
A database is a set of symbols which organized and machine readable [14]. A relational database organizes the symbols in a collection of relations. Its consists of relation, attribute, tuple, and key [15].

A relational database uses a two-dimensional table to represent entities of application domain, such as Customer or Order, or to represent the relationship between entities. This two-dimensional table is called a relation. A tuple in a relational database represents an instance of an entity. Each row (tuple) of the Customer table represents a customer.

An entity has properties representing the attributes or fields of the table. For example, a Customer table may have Name and Address attributes. The domain of an attribute is a range of possible values.
that the attribute may have. Each table in a relational database has one or more fields to identify each of rows on that table called key. The key may be used on another table as a foreign key.

To be a self-described entity, database must contain metadata, a description of the data held in the database. This description is also called schema. With this schema, a program can determine the format, the size and the type of the data. For example, the schema would show that the first 20 bytes of the stored records are the name of the customer, and the next 30 bytes of the records are his address.

### 3.3. Web Based Application

A web based application is a client-server computer program in which the client runs in a web browser. The multilayered architecture approach appropriates to be used in a web application development. By separating the domain layer from the infrastructure layer and the user interface layer, the application development is easier to maintain and the codes are cleaner. Each layer responds to change independently from others. Separating domain layer from other layers also helps the re-engineering process in the future development [1].

A three-tier architecture separates presentation layers, business layer and data layer [16]. This three-tier architecture, then evolved to various architectures depends on the needs of the application. Figure 1 is an example of a multilayered architecture where the business layer is split into an application layer and a domain layer. The application layer handles tasks which triggered by the presentation layer. The domain layer handles business logic which can be used in different presentation layers or even different applications.

![Figure 1. An example of multilayered architecture [16].](image)

Ontology can be used in the application development phase or during the application usage [17]. Zviedries used ontology in the development phase to generate an application for information system [18]. In that research, he created generic component for information system, then the information system will load the ontology and adjust the information system to the ontology specification. There is no editable source code for a developer to edit, because all specifications of his application were coded in the ontology, so a developer needs to have knowledge in ontology to modify the way information interacts.

On the other hand, Kremen used ontology as the persistence layer in his multilayered application [3]. He builds all application components to process data based on Open World Assumption where the data and the knowledge may not complete and will be considered as unknown information rather than invalid information.

In our previous research [19], we were able to generate a knowledge repository (an application) from a given ontology. In that research, the ontology was used once to generate repository components, such as relational database, data editor, and data API. Every tool was used one-by-one manually to perform certain function such as repository component generation, database
transformation, etc. The repository was then composed of those components, and operated based on its database. After the repository was being built, it was completely independent from the origin ontology. Changes in the repository will not be synchronized to the ontology and vice versa.

3.4. Application Generator

A large-scale application may consist of a set of similar modules. Developing an application with similar modules may take time and prone to error since the programmer may reckless in copy-paste certain codes. With an application generator, a programmer only needs to specify the definition of each module.

An application generator at least does one of these processes [20]: (a.) Receives DSL code, parse, and generate code in the target language. (b.) Receives the templates and the content of each template, and then generate the code by filling the template using the given content. (c.) Generates the code algorithmically.

3.5. Spring Framework

Spring Framework is a Java platform with some ready-to-use, comprehensive infrastructures to develop a Java application [21]. With Spring Framework infrastructure, a developer can focus on the development of the core of the application. Spring Framework can be combined with several extensions to build a web based application on the Java EE Platform.

Spring MVC Web Framework consists of dispatcher servlet, controller, model, and view [22]. An HTTP request is processed by a dispatcher servlet to determine which tasks will be executed by determining the URL pattern, parameters, HTTP headers, and other factors. Servlet maintains lifecycle of HTTP request, determines and executes controller class, receives and returns the model and renders the suitable view.

Spring Roo is created to increase the productivity of programmers in a Spring environment. The Spring Roo generates the code based on the given commands from a terminal. It can also generate a model and the controller classes based on class diagram that have been designed on Spring Roo notation. Spring Roo is also able to reverse-engineer an existed database, to generate codes and modules. The results are Java code and not limited to Spring (Spring Security and Spring Web Flow). The codes are also compatible with other Java technology like Java Persistence API (JPA), Java Server Pages, Apache Maven, and AspectJ.

Spring Roo uses a “Convention over Configuration” paradigm: every class in the class diagram will be associated with a model class, a controller class, and a table in a database [23]. However, Spring Roo only generates the code and the database scheme. It cannot be used to populate data in the database.

4. Analysis and Solution

The following section starts with a brief discussion on database to ontology migration and vise-versa, and the generation of web based application.

4.1. Metadata and Data

In this research, a mapping rule between the ontology components and the relational database components is defined as shown in table 1. This mapping is used for the ontology to relational database transformation. We combine the Vysnauskas’s rules [4] and the Spring Framework characteristic.Vysnauskas’s rules are used to generate the tables from the ontology. The Spring Roo generates the models and controller classes for the Spring application.

In Description Logics (DL) community, ontology components are categorized into Terminological Box (TBox)and Assertional Box (ABox). TBox contains assertions about individuals, i.e. OWL facts such as type, property-value, equality or inequality assertions. ABox contains axioms about classes, i.e. OWL axioms such as subclass, equivalent class or disjointness axioms [24]. TBox component is
used to generate the tables in the database and generating the model and controller classes from ontology.

| No | Ontology                  | Relational Database         | Type          |
|----|---------------------------|-----------------------------|---------------|
| 1  | Class                     | Table                       | TBox          |
| 2  | Subclass                  | Table                       | TBox          |
| 3  | Object properties         | Relationship table          | TBox+ABox     |
| 4  | Data type properties      | Field/column                | TBox+ABox     |
| 5  | Constraint                | Datatype, constraint        | TBox          |
| 6  | Individual                | Row                         | ABox          |

The classes and the subclass in an ontology are similar to tables in a database. Object property in the ontology connects an individual to another individual which is similar to the relationship table in a relational database. A functional object property is transformed into a field in the relational database (TBox) and defined as a foreign key. The value of functional object properties is transformed as a row of data in relationship table (ABox). The following equation is used to check the result of the transformation of classes in the ontology to tables in the database:

\[ \text{OKL} = \text{BTB} - \text{BTR} \]  

where:

- \( \text{OKL} \) : the number of classes in the ontology
- \( \text{BTB} \) : the number of tables in the relational database
- \( \text{BTR} \) : the number of relationship tables in the relational database

An individual in ontology becomes a row of data in a relational database. But a relationship table in a relational database may also have row(s) of data. The following equation is used to check the transformation of individuals in the ontology to rows of data in the relational database:

\[ \text{OIN} = \text{BBD} - \text{BBR} \]  

where:

- \( \text{OIN} \) : the number of individuals in the ontology
- \( \text{BBD} \) : the number of rows of data in the relational database
- \( \text{BBR} \) : the number of rows of data in the relationship tables in the relational database

Each knowledge engineer may design a different ontology for the same concept. This may happen because of different requirement in reasoning or other specific requirement that engineer faces. For example, in the pizza ontology [25], names of countries are being represented as individuals while names of pizzas are being represented as subclasses of Pizza class. The system should support a variation from default rules. In our tools, a developer can configure the transformation by ignoring some classes, considering a subclass as a row of a table, or transforms the subclasses to enumerations or constants.

A modified version of the class transformation algorithm from Vysnauskas’s algorithm can be seen in figure 2. In our algorithm, each class, is queued, flagged according to the configuration (if there’s any) while the transformation is done.
The subclasses of that class have the same flag as their superclass’s flag. This modified algorithm changes the equation (1) and the equation (2). Since some classes in an ontology maybe ignored, or may be transformed as row(s) of data in a relational database, the equation (1) is then modified into

\[
OKL - OIG - ODA - OKE - OSE = BTB - BTR
\]

where:
- \( OKL \): the number of classes in the ontology
- \( OIG \): the number of classes in the ontology which will be ignored during transformation
- \( ODA \): the number of subclasses in the ontology which will be considered as rows of data in the relational database
- \( OKE \): the number of classes in the ontology which will be considered as enumeration during transformation
- \( OKT \): the number of classes in the ontology that are not part of \( OIG \), \( ODA \), and \( OKE \)
- \( OSE \): the number of subclasses of \( OKT \)

and the equation (2) is modified into:

\[
ODA + OIN = BBD - BBR
\]
4.2. Change's Impact

Change can happen in the rows of data or the domain model itself. The usage of web application leads to relational database’s change. In this research, we restrict to handle the change in the data, not in the database schema.

In the beginning of the life cycle of the application, the knowledge engineer represents the domain concept in the ontology. The MVC components and relational database will be generated by the system from this ontology considering the configuration rule. The application is ready to be used.

When the knowledge engineer needs to update the ontology, the application and the database must be changed accordingly. On the other side, the data in the database is changed while the application is used. This change must be detected and added to ontology after the data are sent to the ontology. TBox component will be adjusted according to the change, as ABox component from old application will be deleted from the database and repopulated.

4.3. Code Generation

The codes (MVC components) are generated by Spring Roo upon accepting domain model’s description in DSL. The forms and data viewers have similar themes and interfaces. The displays of tables’ fields are generated conformed to the description in the DSL. This DSL is generated by TBox and ABox extractor using transformation mapping (table 1) and algorithm (figure 2).

4.4. System Automation

The automation mechanism generates a web application following a workflow as illustrated in figure 3. At first, a knowledge engineer creates or provides an ontology. Then this ontology with an appropriate configuration file are loaded into the system. At the next step, TBox components from the ontology are extracted. Then the extracted components will be used to generate the code and to build the web application. After the code and database schema generation, ABox components from the ontology are extracted and imported to the database.

![Figure 3. Application Generation Workflow](image)

At this point, the system is a runnable web application with data in RDBMS. As the application is running and being used, the data in the RDBMS are updated. Our framework checks the RDBMS
periodically, and synchronize automatically the ontology from the latest data if changes is detected. In this case, the application does not need to re-generated.

At the last step, when the knowledge engineer changes the ontology, he triggers the process for importing the new ontology that will trigger the code and database re-generation.

The framework has two constraints in the recent version. The domain engineer can’t modify the ontology, while web application is running. The system does not consider the scalability, such as data volume and transaction volume will not be considered in generating this ontology-based application. However, the system does not prevent a developer to modify the generated code.

4.5. Implementation
This section described tools harvested through the open source repositories or implemented by authors. The framework architecture is illustrated in figure 4. The tools are:

1. Apache Maven, a comprehensive tool which is used to make software development project building process easier.
2. Spring Framework, a Java platform that provides comprehensive infrastructure support to develop Java application.
3. SpringKR, a Spring Framework extension to accommodate the Spring application to have knowledge repository functionalities.
4. Spring Roo, a code generator using the infrastructures provided by Spring Framework.
5. RooKR, an add-on for Spring Roo to automatically add SpringKR characteristic into the MVC code.
6. OWL2Roo, an ontology converter from OWL representation to Spring Roo notation which is used in generating web based application.
7. OWLUpdater, a tool to detect change(s) in relational database and add it into the ontology.

Apache Maven, Spring Framework, and Spring Roo are harvested through the open source repositories. SpringKR, RooKR, OWL2Roo, and OWLUpdater are implemented by authors.

5. Experiment and Proof of Concept
To test our system, pizza ontology [25] and a configuration file is loaded to the framework to generate MVC component and pizza database. The configuration file in this testing specifies that: DomainConcept, ValuePartition, Food and IceCream are ignored, subclasses of Pizza, PizzaBase, and PizzaTopping are flagged as data, and Spiciness is flagged as enumeration.
Equation (3) is used to check the result of class transformation, and equation (4) is used to check the result of data transformation. Both equations use the number of component in pizza ontology and pizza database (table 2). Generated MVC code is shown at figure 5.

| No | Code | Component | Location | Total |
|----|------|-----------|----------|-------|
| 1  | OKL  | The number of classes | Ontology | 100   |
| 2  | OIG  | The number of classes that will be ignored (classIgnored) | Ontology | 4     |
| 3  | ODA  | The number of subclasses of classes which flag as data (classAsData) | Ontology | 88    |
| 4  | OKE  | The number of classes which flag as data enumeration (classAsEnum) | Ontology | 1     |
| 5  | OSE  | The number of subclasses of classes which flag as data enumeration (classAsEnum) | Ontology | 3     |
| 6  | OIN  | The number of individuals | Ontology | 5     |
| 7  | BTB  | The number of tables | Database | 6     |
| 8  | BTR  | The number of relational tables | Database | 2     |
| 9  | BBD  | The number of rows of data | Database | 259   |
| 10 | BBR  | The number of rows of data in the relationship tables | Database | 166   |

Figure 5. Generated code

6. Result and Discussion
In this research, we successfully mapped and transformed an ontology into a relational database by a default transformation rules and configuration file. This transformation can be used to generate a web application from the ontology, detect changes in the databases, and synchronize the changes to the ontology. A knowledge engineer can use this as a cycle to maintain ontology and relational database used by the application.
The transformation can be configured as needed by the application developer. Because of the different viewpoint and style among domain engineer in defining a domain, not all ontology elements can be mapped into a certain element in relational database. For example, domain engineer can define a concept as a subclass or an entity from an existing class.

The change in the ontology doesn’t require application developer to change the code manually. The developer can restart the application generator, and the framework adjusts the code and the database accordingly. Also, the change in the database doesn’t require the developer to adjust the change to ontology, because data synchronization can be scheduled by the framework automatically. At this time of writing, synchronization tool is still being developed.

In this research, ontology is chosen as base for generating web based application. We hope in the future, the framework can maximize the role of ontology as knowledge base which can be used in Knowledge Management System.

Our framework generates web application to maintain data (CRUD operations). Axioms in the ontology are transformed into database constraint only. Further development of the framework is planned for generating more complex business rules.

References
[1] E. Evans, “Domain-Driven Design: Tackling Complexity in the Heart of Software”, 2003.
[2] T. R. Gruber, “Towards Principles for the Design of Ontologies Used for Knowledge Sharing,” International Journal of Human-Computer Studies, 43, pp. 907-928, 1995.
[3] P. Kremen, “Building Ontology-Based Information System,” Doctoral Thesis Faculty of Electrical Engineering Department of Cybernetics, 2012.
[4] E. Vysnauskas and L. Nemuraite, “Transforming Ontology Representation from OWL to Relational Database,” Information Technology and Control, Vol. 35, No. 3 A, pp. 333-343, 2006.
[5] N. Stojanovic, et al., “Evolution in The Ontology-Based Knowledge Management System”, 2002.
[6] J. Bormejo, “A Simplified Guide to Create an Ontology,” 2007.
[7] M. Horridge, “A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.3”, The University of Manchester, 2011.
[8] M. M. Roldan & J. A. Montes, “A Protege Plugin for Storing OWL ontologies in Relational Databases”, 2015.
[9] K. J. Lee & T. K. Whangbo, “Semantic Mapping between RDBMS and Domain Ontology,” Computers and Communications, 2007. ISCC 2007.
[10] G. Bumans, “Mapping between Relational Databases and OWL Ontologies: an example”, Scientific Papers, University Of Latvia, 2010.
[11] N. Cullot, et al., “DB2OWL: A tool for automatic database-to-ontology mapping”, Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007.
[12] M. D. M. Roldan & J. A. Montes, “A Protege Plugin for Storing OWL ontologies in Relational Database”, 2015.
[13] Y. Liu & P. McBrien, “SQOWL2: Transactional Type Inference for OWL 2 DL in an RDBMS”, British International Conference on Databases 2015, pp. 206-219.
[14] H. Darwen, “An Introduction to Relational Database Theory”, Ventus Publishing ApS, 2010.
[15] C. Ritchie, “Database Principles and Design”, 3rd edition, Delmar Learning, 2008.
[16] D. Esposito, “Modern Web Development: Understanding Domains, Technologies, and User Experiences”, Microsoft Press, 2016.
[17] N. Guarino, “Formal Ontology and Information System,” Proceedings of FOIS’98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15, 1998.
[18] M. Zviedris, et al., “Ontology-Based Information System,” JIST 2013, pp.33-47.
[19] K. Mahmudi & I. Liem, “Ontology-Based Knowledge Repository: using Java Spring Framework,” KNIF 2015.
[20] I. Liem & Y. Nugroho, “An Application Generator Framelet,” Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. DOI: 10.1109/SNPD.2008.113

[21] Pivotal Software, “Spring Framework Reference Documentation”, 4.2.1.RELEASE.

[22] K. Rimple & S. Penchikala, “Spring Roo in Action”, Manning Publications Co, 2012.

[23] Pivotal Software, “Spring Roo – Reference Documentation”, 1.3.1.RELEASE. https://github.com/rabarro/spring-roo-1.3.1.RELEASE/blob/master/docs/pdf/spring-roo-docs.pdf

[24] E. Sirin, et al., “Pellet: A Practical OWL-DL Reasoner”, Web Semantics:Science, Services and Agents on the World Wide Web, Volume 5, Issue 2, pp. 51-53, 2007.

[25] Stanford University, “Pizza Ontology”, last access at October 2nd, 2015, http://protege.stanford.edu/ontologies/pizza/pizza.owl