Ontology Learning from Relational Databases: Transforming Recursive Relationships to OWL2 Components

Mohammed Reda CHBIHI LOUHDI[1]
Research Laboratory on computer science innovation (LRII)
Faculty of Science Aïn Chock Casablanca
Hassan II University, Casablanca, Morocco

Hicham BEHJA[2]
LRI - Laboratory
ENSEM, Hassan II University
Casablanca, Morocco

***Abstract*—Relational databases (RDB) are widely used as a backend for information systems, and contain interesting structured data (schema and data). In the case of ontology learning, RDB can be used as knowledge source. Multiple approaches exist for building ontologies from RDB. They mainly use schema mapping to transform RDB components to ontologies. Most existing approaches do not deal with recursive relationships that can encapsulate good semantics. In this paper, two technics are proposed for transforming recursive relationships to OWL2 components: (1) Transitivity mechanism and (2) Concept Hierarchy. The main objective of this work is to build richer ontologies with deep taxonomies from RDB.**

***Keywords*—Relational databases; ontologies; OWL2; recursive relationship; transitivity; concept hierarchy**

I. Introduction

A relational database is a digital database based on the relational model of data, as proposed by Codd in 1970 [1]. RDB use many components (tables, constraints, etc.) to manage data in a structured way. These databases are usually created on the basis of a conceptual model which is established by designers after a deep analysis of an information system.

However, RDBs are considered “semantically poor” because of the nature of the used components that are structure-oriented. Indeed, the schema of a RDB is composed by a set of tables related by foreign key constraints. This limitation makes the use of RDB for semantic purposes very difficult. Transforming the RDB to an ontology can lift the limitation.

According to Tom Gruber, “an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application” [2].

There are many ways to represent ontologies. The choice of the representation to use depends on the ontology operationalization. The Web Ontology Language is one of the most used languages to represent ontologies on the Web. It is an ontology language for the Semantic Web with formally defined meaning. OWL 2 (latest version of the language) ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents [3].

There are many approaches that transform a RDB to an ontology. Three main techniques are used: (1) Reverse Engineering, to convert the relational model to the conceptual model (which is considered as semantically richer than the relational model) or to retrieve lost information during the transformation of the conceptual model to the relational model, (2) Schema Mapping, to convert relational model components to ontology components, through the use of transformation rules and (3) Data Mining to analyze stored data in order to enrich the ontology.

The majority of the existing techniques for transforming RDBs to ontologies do not deal with the specific case of Recursive Relationships that are simply transformed to a property with domain and range that belongs to the same class.

This work is supported by the project “Knowledge Management for Development in the Context of OCP Group (KM4Dev – OCP Group)” granted by OCP Group – Morocco.

This paper will discuss how we can extract richer semantics from Recursive Relationships by the use of OWL2 components. The rest of this paper is organized as follows: In the 2nd Section, present the previous works of the authors in the case of transforming RDBs to OWL ontologies. The Section 3 gives an overview of related works for transforming Recursive Relationships in RDBs to ontologies. Section 4 discusses Recursive Relationships in RDBs. Section 5 deals with Recursive Properties in OWL2 ontologies. In the 6th Section, a proposal for transforming Recursive Relationships in RDBs to OWL2 components is tackled. Finally, the last Section will include concluding remarks and some topics for further works.

II. Previous Work

In the case of building OWL ontologies from relational databases (RDBs), authors have conducted several researches. In [4], a set of transformation rules was proposed for building OWL ontologies from RDBs. It allows transforming all possible cases in RDBs (one-to-many relations, many-to-many relations, n-ary relations and inheritance) into ontological constructs. After that, a hybrid method for automatic ontology building from a RDB was proposed [5]. That hybrid method combines reverse engineering, schema mapping and data analysis techniques. The extracted ontology is refined by renaming the components whose names do not reflect their real meaning. This method allows (1) recovering lost tables during the mapping of ER-Model components to the relational model, by using reverse engineering technique for the generalization and specialization cases; (2) transforming, in the schema mapping phase, the different constructs and cases such as multiple inheritance, n-ary relations, etc.; (3) analyzing stored data to detect disjointness and totalness (or Completeness) constraints in hierarchies, and calculating the participation level of tables in n-ary relations. This method begins with recovering the database schemata, after that, a set of proposed algorithms are applied to detect generalized and specialized tables in order to enrich the ontology taxonomy. In the next step, a set of transformation rules is applied on the schema to convert the database components to ontology components. At the end, a manual refinement phase is conducted to rename components (classes and properties) having automatic generated names. In [6], an enhancement for the previously proposed algorithms for reverse engineering was presented. The main objective of the enhancement is to detect the lost entities in the multi-level inheritance for the generalization and specialization cases. Indeed, the proposed algorithms in [5] deals with only one-level inheritance.

In the previous works, recursive relationships in relational databases was not correctly transformed. This kind of relations contains semantics that can be used to build richer OWL2 ontologies rather than transforming it to a simple property. In the next section, related works to this case will be discussed.

III. Related Works

In the case of transforming recursive relationships from relational databases to ontologies, there are many works (generally irrelevant to that case). In [7], authors suggest a mapping method to map the two types of recursive relationships to the resource description framework schema (RDFS) [8]. One type of recursive relationship is generated during the integration process with different entities at different levels in a hierarchical structure while the other is generated at the hierarchical structure between the instances of an attribute in an entity. The transformed RDFS, including the class, subclasses, and sub-properties minimizes the loss of data meaning and enables the process of the inference function to be used with RDB data. In paper [9], authors give some proof case studies and propose a model to upgrade the semantics of the relational model, before the ontology learning. They argue that without an explicit model of the domain semantics in the relational model, the automatic learning of ontology risks to infer incorrect semantics. Recursive relationships are transformed into two mutually inverse object properties having domain and range referencing the same class. In [10], the author presents ERD (Entity-Relation Diagram) to OWL-DL ontology transformation rules at a concrete level. These rules facilitate an easy and understandable transformation from ERD to OWL. To transform recursive relationships, the author creates a new class representing the recursive association, resulting on an existential quantification restriction (some Values From restriction in OWL) on the class corresponding to the table having the recursive relationship. In [11], authors propose a method that is consisted of two main phases: building ontology from an RDB schema and the generation of ontology instances from an RDB data automatically. In the first phase, they study different cases of RDB schema to be mapped into the ontology represented in RDF(S)-OWL, while in the second phase, the mapping rules are used to transform RDB data to the ontological instances represented in RDF triples. Rules 15 and 16, are applied to transform recursive relationships into a Transitive or a Symmetric Property. Transitivity can be applied in most cases. But Symmetry can only be applied if and only if each row of the table (having a recursive relationship) references (through a foreign key constraint) another row in the same table and vice versa (one row references the other in both ways).

IV. Recursive Relationships in Relational Databases

The conceptual data model is a structured business view of the data required to support business processes, record business events, and track related performance measures. This model focuses on identifying the data used in the business but not its processing flow or physical characteristics. This model's perspective is independent of any underlying business applications [12]. The conceptual data model represents the overall structure of data required to support the business requirements independent of any software or data storage structure. It is an important phase before building the database.

Entity-Relationship (ER) modeling is a logical design modeling technique. After the business requirements and data requirements are gathered and the business rules understood, we can start developing the logical data model. An ER model is often referred to as a 3NF (third normal form), or sometimes just a normalized model for short. It is also sometimes referred to as a relational model, which is incorrect. Although it is implemented in a relational database, it is not the sole data modeling technique that could be used in a relational database [12].

The ER modeling concepts are sufficient for representing many database schemas for traditional database applications, which include many data-processing applications in business and industry. However, designers of database applications have tried to design more accurate database schemas that reflect more precisely the data properties and constraints. This led to the development of additional semantic data modeling concepts that were incorporated into conceptual data models, such as the ER model. Various semantic data models have been proposed in the literature. Among them, we find the EER Model (Enhanced Entity-Relationship Model) that incorporates a set of new concepts (class/subclass relationships and type inheritance into the ER model, specialization and generalization, various types of constraints on specialization/generalization, etc.) [13].

One of the most difficult relationships to express is a recursive relationship. This is a non-identifying, non-mandatory relationship in which the same entity is both the parent and the child [12]. Each migrating primary key attribute must be given a role name to clarify the attribute's foreign key role. In figure 1, to express the fact that an Employee can supervise another one, a recursive relationship can be used.

The *Emp_SSN* column in relational model represents the supervisor identifier.

However, variations in relationships can be masked in such a model when a dependency exists between relationships or between a relationship and an attribute [14]. For example, in figure 1, some employees manage other employees, while other employees aren't managed (they don't have a supervisor).

Another usage case for recursive relationships is the organization of categories of articles in inventory management systems (each article is belonging to a category). Usually, categories in such a system, are organized in a hierarchical way (a category contains multiple categories which contains others and so on). The figure 2 shows the ER Model representing this case and the corresponding Relational Model. The *Parent_Category_id* column in relational model represents the parent category identifier.

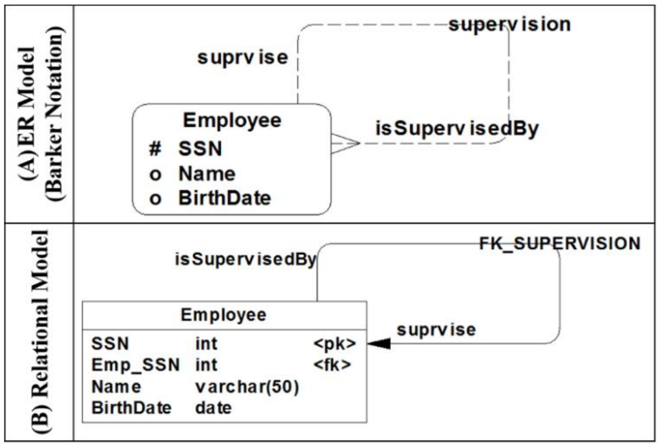


Fig. 1. Example of a Recursive Relationship in ER Model and Relational Model.

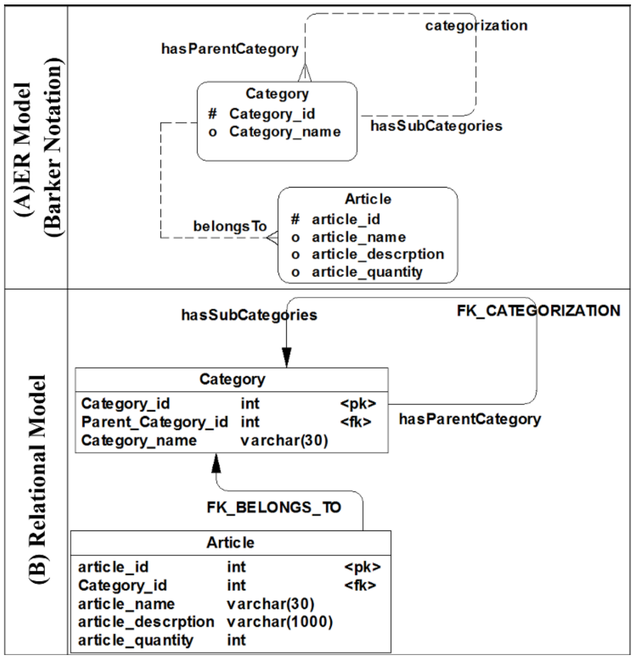


Fig. 2. Using Recursive Relationship to Categorize other Entities.

V. Recursive Relations in OWL2 Ontologies

The Web Ontology Language, informally OWL2, is an ontology language for the Semantic Web with formally defined meaning. OWL2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents [3]. There is no specific way to express recursive properties in OWL2. It is simply represented as an Object Property having domain and range referencing the same class. In the example of figure 3, a Person has as parent another Person. The corresponding code using OWL Functional syntax (which will be used in all following examples) is presented below.

However, the OWL2 recommendation defines multiple characteristics that can be assigned to recursive properties: (A)Symmetry, (IR)Reflexivity and Transitivity.

A. Symmerty

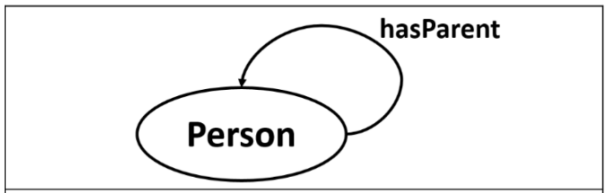
An object property (SOP) is considered as symmetric, if an individual x is connected by SOP to an individual y, then y is also connected by SOP to x. In the example of figure 4, a Woman has as sister another Woman.

B. Asymmerty

An object property (AOP) is considered as asymmetric, if an individual x is connected by AOP to an individual y; then y cannot be connected by AOP to x. In the example of figure 5, a Person has as parent another Person, the property is not valid in both directions.

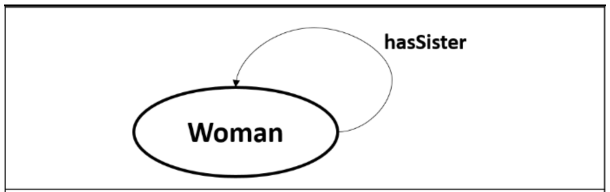
C. Reflexivity

An object property (ROP) is considered as reflexive, if an individual x is connected by ROP to itself. In the example of figure 6, a Person knows himself.



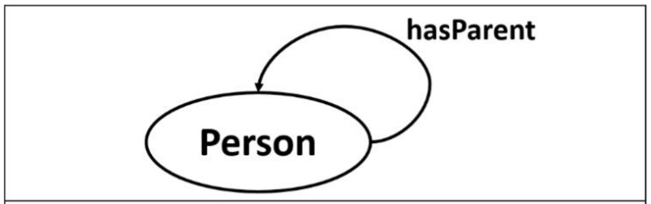
```
Declaration(Class(:Person))
Declaration(ObjectProperty(:hasParent))
ObjectPropertyDomain(:hasParent :Person)
ObjectPropertyRange(:hasParent :Person)
```

Fig. 3. Example of a Recursive Property.



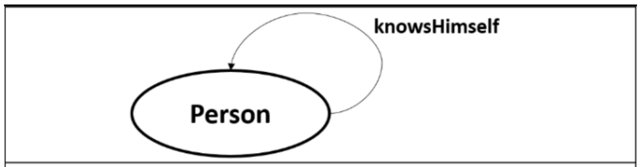
```
Declaration(Class(:Woman))
Declaration(ObjectProperty(:hasSister))
SymmetricObjectProperty(:hasSister)
ObjectPropertyDomain(:hasSister :Woman)
ObjectPropertyRange(:hasSister :Woman)
```

Fig. 4. Example of a Symmetric Property.



```
Declaration(Class(:Person))
Declaration(ObjectProperty(:hasParent))
AsymmetricObjectProperty(:hasParent)
ObjectPropertyDomain(:hasParent :Person)
ObjectPropertyRange(:hasParent :Person)
```

Fig. 5. Example of an Asymmetric Property.



```
Declaration(Class(:Person))
Declaration(ObjectProperty(:knowsHimslef))
ReflexiveObjectProperty(:knowsHimslef)
ObjectPropertyDomain(:knowsHimslef :Person)
ObjectPropertyRange(:knowsHimslef :Person)
```

Fig. 6. Example of a Reflexive Property.

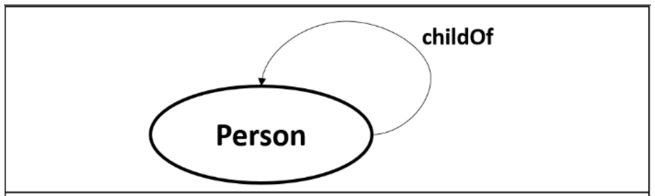
D. *Irreflexivity*

An object property (IOP) is considered as irreflexive, if an individual x cannot be connected by IOP to itself. In the example of figure 7, a Person is the child of another person, but cannot be the child of himself.

E. *Transitivity*

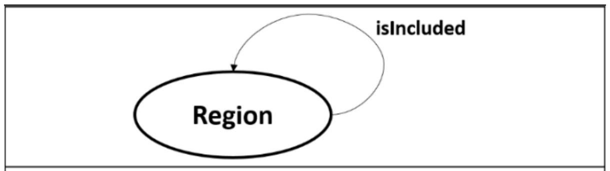
An object property (TOP) is considered as transitive, if an individual x is connected by TOP to an individual y, which is connected to another individual z, then x is also connected to z. In the example of figure 8, a Region is included in another Region (if a Region r1 is included in a Region r2, that is included in a Region r3, then r1 is included in r3).

In the next section, two ways to transform recursive relationships from relational model to OWL2 components will be proposed.



```
Declaration(Class(:Person))
Declaration(ObjectProperty(:childOf))
IrreflexiveObjectProperty(:childOf)
ObjectPropertyDomain(:childOf :Person)
ObjectPropertyRange(:childOf :Person)
```

Fig. 7. Example of an Irreflexive Property.



```
Declaration(Class(:Region))
Declaration(ObjectProperty(:isIncluded))
TransitiveObjectProperty(:isIncluded)
ObjectPropertyDomain(:isIncluded :Region)
ObjectPropertyRange(:isIncluded :Region)
```

Fig. 8. Example of a Transitive Property.

VI. Proposed Transformations for Recursive Relationships

This section discusses and compares two different transformations for recursive relationships in relational model. The first one is to use Transitivity mechanism and the second one through the use of concept hierarchy.

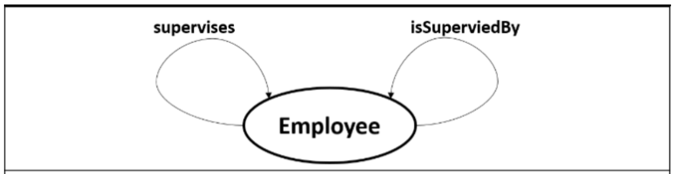
A. *Transitivity Mechanism*

For the first proposal, two mutually inverse object properties are created (each one is the inverse of the other) with domains and ranges referencing the same class (corresponding to the table having the recursive relationship). These properties are also declared as Transitive figure 9.

In the example of figure 1, an Employee is supervised by another Employee. Inversely, an Employee supervises another one. The transformation of this case will produce, a class representing an "Employee", and two mutually inverse Object Properties "*isSupervisedBy*" and "*supervises*" having domains and ranges referencing the "*Employee*" class. The figure below presents the obtained results.

B. *Concept Hierarchy*

In some cases, recursive relationships in the relational model are used to classify, in a hierarchical way, the occurrences, of other entities. In the example of figure 2, Articles are categorized in categories that are organized hierarchically using a recursive relationship.



```
Declaration(Class(:Employee))
Declaration(ObjectProperty(:isSupervisedBy))
Declaration(ObjectProperty(:supervises))
InverseObjectProperties(:isSupervisedBy :supervises)

TransitiveObjectProperty(:isSupervisedBy)
ObjectPropertyDomain(:isSupervisedBy :Employee)
ObjectPropertyRange(:isSupervisedBy :Employee)

TransitiveObjectProperty(:supervises)
ObjectPropertyDomain(:supervises :Employee)
ObjectPropertyRange(:supervises :Employee)
```

Fig. 9. Transformation Example using Transitivity Mechanism.

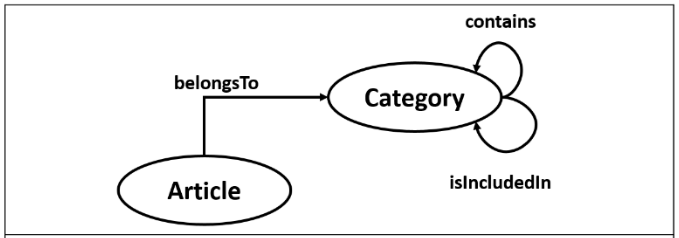
Transforming the example of figure 2 can be achieved using Transitivity. As result, two classes will be obtained, corresponding to the entities "*Article*" and "*Category*", linked with an Object Property "*belongsTo*". Two other mutually inverse Object Properties ("*contains*" and "*isIncludedIn*") will be created having domains and ranges referencing the class "*Category*". These two Properties are also declared as Transitive. The figure 10, shows the obtained transformation.

A better result can be obtained through the use of the Concept Hierarchy, which is a type of background knowledge (an approach for guiding the knowledge discovery process, and for evaluating the patterns found [15]) which expresses the structure of the concept from low-level to a more general concept. The use of the concept hierarchies as background knowledge allows expressing the discovered knowledge in a higher abstraction level, more concise and usually in a more interesting format [16].

The proposed solution is to create a class hierarchy from the occurrences of the entity having the recursive relationship. Each class of the hierarchy will be formed as Disjoint Union of all its subclasses.

As an alternative to the solution proposed in figure 10, a class hierarchy will be created from the occurrences of the table "Category". Each class in the hierarchy correspond to an existing category (occurrence) and is related to other categories by an "is-a" relation.

To explain this proposal, the database of the Inventory Management system of Cadi Ayyad University of Marrakech (named SyGeS : *Système de Gestion de Stock)* will be used. The "*Category*" table is used to categorize articles (each Article belongs to a Category). Figure 11 shows the hierarchical organization of categories in the system.



```
Declaration(Class(:Article))
Declaration(Class(:Category))
Declaration(ObjectProperty(:belongsTo))
Declaration(ObjectProperty(:contains))
Declaration(ObjectProperty(:isIncludedIn))

ObjectPropertyDomain(:belongsTo :Article)
ObjectPropertyRange(:belongsTo :Category)

InverseObjectProperties(:contains :isIncludedIn)
TransitiveObjectProperty(:contains)
ObjectPropertyDomain(:contains :Category)
ObjectPropertyRange(:contains :Category)

TransitiveObjectProperty(:isIncludedIn)
ObjectPropertyDomain(:isIncludedIn :Category)
ObjectPropertyRange(:isIncludedIn :Category)
```

Fig. 10. Transformation of Example of Figure 2 using Transitivity Mechanism.

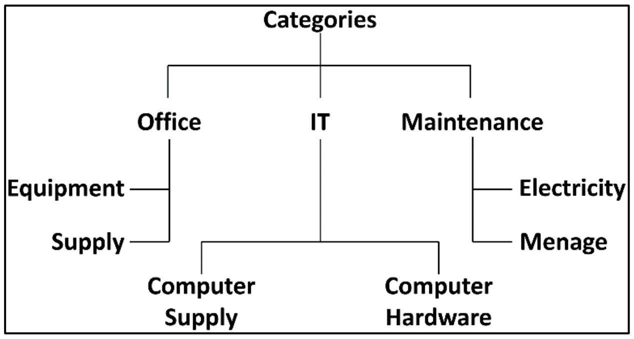
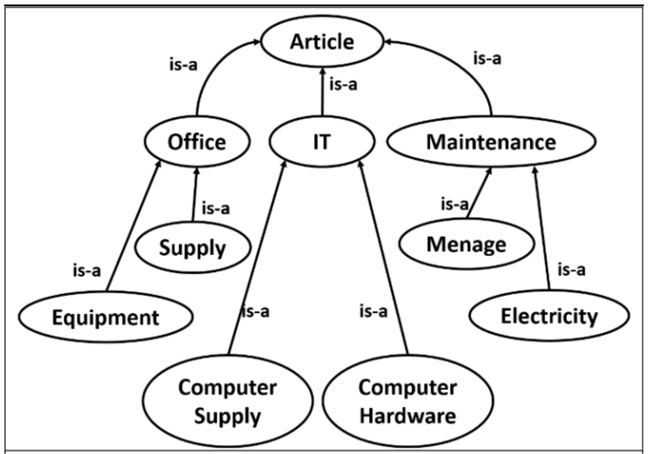


Fig. 11. Article' Categories in SyGeS System.

As result of the transformation (applying Concept Hierarchy) of this case, the categories (occurrences of the table "*Category*") will be declared as classes and organized in a hierarchical way, by detecting the parent of each category (through the recursive relationship). The detected root categories (categories without a parent: null value on foreign key column) will be declared as subclasses of the "Article" class. The figure 12 presents the obtained transformation.



```
Declaration(Class(:Article))
Declaration(Class(:ComputerHardware))
Declaration(Class(:ComputerSupply))
Declaration(Class(:Electricity))
Declaration(Class(:Equipment))
Declaration(Class(:IT))
Declaration(Class(:Maintenance))
Declaration(Class(:Menage))
Declaration(Class(:Office))
Declaration(Class(:Supply))

SubClassOf(:IT :Article)
SubClassOf(:Maintenance :Article)
SubClassOf(:Office :Article)
SubClassOf(:ComputerHardware :IT)
SubClassOf(:ComputerSupply :IT)
SubClassOf(:Electricity :Maintenance)
SubClassOf(:Menage :Maintenance)
SubClassOf(:Equipment :Office)
SubClassOf(:Supply :Office)

DisjointUnion(:Article :IT :Maintenance :Office)
DisjointUnion(:IT :ComputerHardware :ComputerSupply)
DisjointUnion(:Maintenance :Electricity :Menage)
DisjointUnion(:Office :Equipment :Supply)
```

Fig. 12. Transformation of Example of Figure 2 using Concept Hierarchy.

At the end, the occurrences of table "*Article*" are transformed to individuals of the class corresponding to their category in the ontology. For example, an article "*Laptop*" will be declared as an individual of the class "*ComputerHardeware*".

VII. Conclusion and Future Work

This paper proposes two manners for transforming recursive relationships from relational databases to OWL2 ontologies' components. In the first one, Transitivity mechanism is applied as a characteristic for the created object properties representing the recursive relationship. In the second one, Concept Hierarchy is used to build a taxonomy of classes from the occurrences in the table having the recursive relationship. This proposal is an enhancement of the proposed transformation rules in [4][5].

The main objective of this work is learning OWL2 ontology from relational databases to extract richer semantics. As future work, the tables' occurrences will be analyzed in order to extract deeper taxonomies. Some existing approaches tries to do that, like in [17] where the author combines a classical analysis of the database schema with a task specifically dedicated to the identification of categorization patterns in the data. Another improvement clue is to enhance the developed tool in [5] by integrating all recent researches.

Acknowledgment

This work is supported by the project "Knowledge Management for Development in the Context of OCP Group (KM4Dev – OCP Group)" granted by OCP Group – Morocco.

Thanks to Mr. ETTKIRITA Youness, manager of the Stock Management System at the University of Cadi Ayyad– Marrakech, for providing a part of the database of their system to validate our proposal.

References

[1] G. Eason, B. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM. 13 (6): 377–387, 1970.

[2] Gruber, T., "Ontology". The Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009.

[3] W3C, "OWL 2 Web Ontology Language : Structural Specification and Functional-Style Syntax (Second Edition)", W3C Recommendation. Available at: https://www.w3.org/TR/owl2-syntax, 2012.

[4] Chbihi Louhdi, M. R., Behja, H., & Ouatik El Alaoui, S., "Transformation Rules for Building OWL Ontologies from Relational Databases". In the proceeding of the Second Conference of Data Mining & Knowledge Management Process. Dubai, 2013.

[5] Chbihi Louhdi, M.R., Behja, H., & Ouatik El Alaoui, S., "Hybrid Method for Automatic Ontology Building from Relational Database". International Review on Computers and Software, vol. 8(8), pp. 1801-1813, 2013.

[6] Sbai, S., Chbihi Louhdi, M.R., Behja, H., Zemmouri, E., & Chakhmoune, R., "Using Reverse Engineering for Building Ontologies with Deeper Taxonomies from Relational Databases". Journal of Software, Vol. 14(3), pp.138-145, 2019.

[7] Choi, M., Moon, C., Baik, D., Wie, Y., & Park, J., "The RDFS mapping for recursive relationship of relational data model". The IEEE International Conference on Service-Oriented Computing and Applications (pp. 1-6), Perth, WA, 2010.

[8] W3C, "RDF Schema 1.1, W3C Recommendation". Available at: https://www.w3.org/TR/rdf-schema, 2014.

[9] Idrissi, B.E., Baina, S., Baina, K., "Upgrading the semantics of the relational model for rich OWL 2 ontology learning". Journal of Theoretical and Applied Information Technology, vol. 68(1), pp. 138-148, 2014.

[10] Fahad, M., "ER2OWL: Generating OWL Ontology from ER Diagram". In: Shi Z., Mercier-Laurent E., Leake D. (eds) Intelligent Information Processing IV. IIP. IFIP – The International Federation for Information Processing, vol 288. Springer, Boston, MA, 2008.

[11] Hazber, M.A.G., Li, R., Gu, X., & Xu, G., "Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology". International Journal of Applied Mathematics & Information Sciences, vol. 10(3), pp. 881-901, 2016.

[12] Elmasri, R., & Navathe, S.B., "Fundamentals of Database Systems (7th Edition)", Pearson, 2016.

[13] Sherman, R., "Business Intelligence Guidebook From Data Integration to Analytics". In Foundational Data Modeling, pp. 173-195, Elseiver, 2015.

[14] Burton-Jones, A. Lazarenko, K., & Weber, R., "Problems with recursive relationships and relationships with attributes in ER models". In Proceedings of the 10th AIS SIGSAND Symposium, Bloomington, Indiana, USA, 2011.

[15] Han, J., Kamber, M., & Pei, J., "Data Mining: Concept and Techniques, Morgan Kaufmann", Elseiver, 2011.

[16] Di Beneditto M.E.M., & de Barros L.N., "Using Concept Hierarchies in Knowledge Discovery". In: Bazzan A.L.C., Labidi S. (eds) Advances in Artificial Intelligence – SBIA 2004. Lecture Notes in Computer Science, vol 3171. Springer, Berlin, Heidelberg, 2004.

[17] Cerbah, F., "Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies". In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (pp. 553-557). Sydney, NSW, 2008.