Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 8 Issue 6, June - 2022

Learning Ontology from Relational Database

(Symmetry and Transitive Characteristics)

Paramita Mayadewi[1]
Institut Teknologi Bandung, School of Electrical Engineering and Informatics
Telkom University
Bandung, Indonesia
paramita@telkomuniversity.ac.id

Benhard Sitohang[2]
Institut Teknologi Bandung, School of Electrical Engineering and Informatics
Bandung, Indonesia
benhard@stei.itb.ac.id

Fazat N. Azizah[3]
Institut Teknologi Bandung, School of Electrical Engineering and Informatics
Bandung, Indonesia
fazat@informatika.org

***Abstract*—In ontology learning, relational databases can be used as a source of knowledge. The are several approaches to building ontologies from relational databases. Most of them use schema analysis to transform database components into ontology components. Few of the existing approaches deal with symmetric and transitive relationships in databases. This paper proposes an approach using primary and foreign key patterns to identify symmetric and transitive relationships. This work aims to infer the facts stated in the knowledge base and enrich the ontology generated. Tests were carried out using the Pellet reasoner on the Protégé application and showed significant results.**

Keywords—symmetric; transitive; primary key; foreign key; ontology; relational databases

I. INTRODUCTION

Ontology is a conceptualization tool at the semantic and knowledge level, providing explicit descriptions and methods for information and knowledge [1]. Ontology development is an engineering activity, and there are two main approaches to building, namely building from scratch (manually) or using an ontology learning approach. The term ontology learning describes an approach to finding ontological knowledge automatically or semi-automatically from various sources [2]. [3] distinguishes different ontology learning approaches based on resource types as follows: ontology learning from unstructured data (web pages), ontology learning from semi-structured data (XML document) and ontology learning from structured data (databases). There are many ways to represent an ontology. Web Ontology Language (OWL) is one of the most widely used languages for representing ontologies. OWL is an ontology language for the semantic web with formally defined meanings. OWL provides class, property, individual, and data values and is stored as a semantic web document [4].

Regardless of how ontologies are represented, learning ontologies from relational databases is not a new research problem. Several approaches and tools have been developed to build ontologies of relational databases [5][6][7][8][9][10][11][12][13][14][15]. There are three main techniques used: (1) reverse engineering, converting the relational model to a conceptual model (which is considered to be semantically richer than the relational model) or retrieving information lost during the transformation of the conceptual model to the relational model; (2) schema mapping, converting relational components into ontology components, through the use of transformation rules and (3) data mining to analyze stored data to enrich the ontology.

However, not many studies discuss the identification of symmetric and transitive relationships. Identifying symmetric and transitive relationships can help infer the facts to be stated in the knowledge base and enrich the resulting ontology. The approach taken by [15][16][17] establishes rules for identifying symmetric and transitive relationships. The rules for symmetric relationships are defined in a unary relationship table where the foreign key refers to a primary key in the same table. Transitive rules are applied to unary relationships with the On Delete Cascade constraint. This constraint describes the whole and part relationship, where the part cannot exist without the whole (i.e. if the parent data is deleted, all child data that refers to it will also be deleted) [18][19]. Based on this, foreign keys are mapped into transitive relationships.

Not all database designs apply On Delete Cascade constraints on unary relationships, so it will be hard to identify symmetric and transitive relationships if only based on the presence or absence of On Delete Cascade constraints on unary relationships. This paper proposes an approach to identifying symmetric and transitive relationships based on the patterns formed between primary and foreign keys.

The remainder of this paper is organized as follows. Section 2 discusses the proposed approach as well as a brief explanation of the symmetry and transitive relationships. We describe the experiments carried out and the evaluation of the proposed approach in section

3, followed by a discussion of our work. The final section includes concluding comments and some topics for further work.

II. PROPOSED APPROACH

Symmetric and transitive relationships cannot be described explicitly in a relational database, whereas OWL allows the meaning of properties to be enriched by using characteristic properties. Symmetric and transitive relationships in relational databases will be translated into the object properties characteristics in OWL. Some characteristics are functional, inverse functional, transitive, symmetric and others [20]. In order to understand the symmetric and transitive relationship, we start with the following definition. [21][22].

Definition 1: A relation R on a set A is symmetric if every aRb then bRa, that is, if every (a,b) ∈ R then (b, a) ∈ R. Thus, R is not symmetric if there is a, b ∈ A such that (a, b) ∈ R but (b, a) ∉ R.

Symmetric is the opposite of itself. If a property P is symmetric, and the property relates individual a to individual b, then individual b is also related to individual a via property P. The hasSibling or hasSpouse property is an example of a symmetric property. If Hansel has a spouse Ailee, then Ailee has a spouse Hansel.

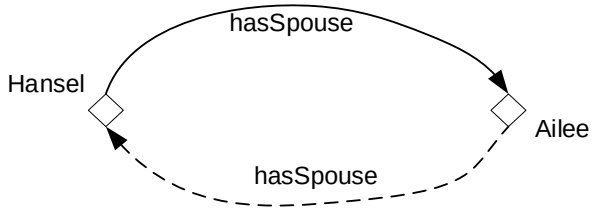


Fig 1. *Example of Symmetric Property* : hasSpouse

Definition 2: A relation R on a set A is transitive if every aRb and bRc then aRc, that is, if every (a, b), (b, c) ∈ R then (a, c) ∈ R. Thus, R is not transitive if there is a, b, c ∈ R such that (a, b), (b, c) ∈ R but (a, c) ∉ R.

Suppose property P is transitive, and P relates individual a to individual b and individual b to individual c. In that case, it is concluded that individual a is related to individual c through property P. For example, a transitive relationship has an ancestor. Individual Diana has Freya ancestors, and Freya has Havana ancestors, so it can be concluded that Diana has Havana ancestors.

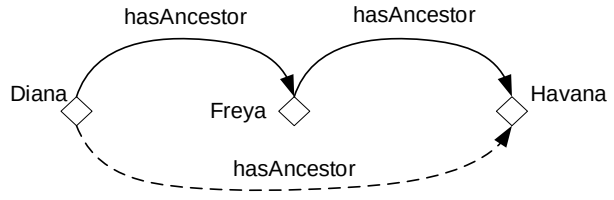


Fig 2. *Example of Transitive Property* : hasAncestor

As previously explained, we limit our approach to tables which are unary relations. A unary relationship, also called recursively, is a relationship in which there is a relationship between occurrences and the same entity set. In this relationship, the primary and foreign keys are the same but they represent two entities with different roles. Here's the definition.

Definition 3: Given a relaton schema R, where r is an instance of the relation. The primay keys pk in R (i.e. pk = pkey(R)) and fk are foreign keys in R that refer to pk (i.e. fk ∈ fkey(R) and refpk(fk) = pk).

The problem is that there are many possible relational database designs exist where the relationships between tables can be symmetric, transitive or both. The relationship between tables that reflect the symmetric and transitive relationships can be seen based on characteristics of the primary and foreign key relationships. In this paper, we limit the identification of symmetric and transitive relationships to unary relationships. The following is an example of an illustration of symmetric and transitive relationship data that may be contained in a unary relationship.

1. **Person**{id (PK), name, spouse (FK reference Person(id))}

Id	Name	Spouse
10	Ali	30
20	Ayten	50
30	Ayse	10
40	Bedriye	
50	Ismail	20
60	Mediha	

The primary and foreign key relationship characteristics form a symmetric pattern in the **Person** table. For example, Spouse(Ali, Ayse) → Spouse(Ayse, Ali)

2. **Employee**{id (PK), name, manager_id (FK reference Employee(id))}

Id	Name	Manager_id
100	Steven	
101	Neena	100
102	Lex	100
103	Alexander	102
104	Bruce	103
:	:	:
114	Den	100
115	Alexander	114
116	Shelli	114

The Employee table has a transitive pattern. Lex is Alexander's manager, and Alexander is Bruce's manager, so indirectly, Lex is Bruce's manager or Bruce's superior. Manager(Lex, Alexander) ∧ Manager(Alexander, Bruce) → Manager (Lex, Bruce).

Our approach identifies symmetric/transitive relationships based on the pattern formed between the primary and foreign keys. After the pattern is detected, data analysis is carried out to determine the characteristics of the pattern formed, whether it is a symmetric or transitive relationship. The proposed identification method considers the number of data instances that support the findings and only processes those that pass a certain threshold. Here is the algorithm.

```
Input: Query result (pk & fk)
Output: Transitive/Symmetric Characteristic

Check_Transitive()
  num = len(list)
  for rec in list
   b=0
   while (b < num)
    if (fk[rec]=pk[b]) and (pk[rec] <>
        fk[b])
       add to list fk and pk
    b=b+1
  jum=0
  for k in list
     jum=jum+1

  if (jum = num)
    return "Transitive"
```

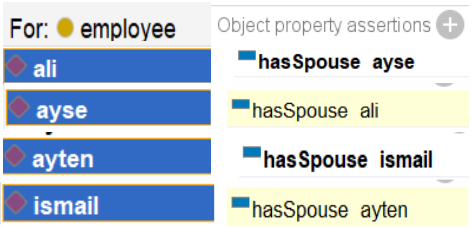
```
check_Symmetry()
  num=len(list)
  for c in list
   b=0
   while (b<num) and ((pk[c],fk[c]) <>
         (fk[b], pk[b]))
     b=b+1

   if ((pk[c], fk[c]) = (fk[b], pk[b])
     add to list fk and pk
   else
     add to list2 fk and pk

 if list2 null
   return "Symmetry"
```

III. Experiments and Evaluation

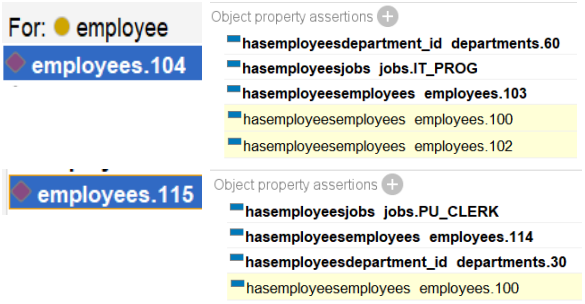
Symmetric and transitive relationships in relational databases are translated into Object Properties characteristics in OWL. The resulting OWL document from the proposed algorithm was tested in the free, open-source ontology editor, Protégé. We use the Pellet reasoner to check the inference correctness of the resulting ontology. Figure 3 shows the symmetric relationship inference results and the corresponding code using the Manchester OWL syntax from the symmetric relationship example shown in section 2.




```
ObjectProperty: hasSpouse
   Characteristic: Symmetric
   Domain: employee
   Range: employee
```

Fig 3. *Example of Symmetric Property Results*

Figure 4 shows transitive relationships inference results and the coresponding code using the Manchester OWL syntax from the example of the transitive relationship shown in section 2. The results of the inference show the indirect manager of an employee. For example, employee104 has employee103 as a direct manager. While employee103 has an employee102 as direct managers and employee102 has an employee100 as direct manager. The inference results for employee104 indicate that employee102 and employee100 are indirect managers od employee104. The same applies to employee115. The inference results show that the indirect manager of employee115 is employee100, and the direct manager is employee114.




```
ObjectProperty: hasemployeesemployees
   Characteristic: Transitive
   Domain: employees
   Range: employees
```

Fig 4. *Example of Transitive Property Results*

IV. Discussion

Relational schemas can capture some cardinality constraints on relationships between entities by defining constraints on foreign keys. However, relational schemas do not have the expressive power to define relationships with logical characteristics such as symmetric and transitive. We have proposed an approach to identify symmetric and transitive relationships in unary relationship tables by finding the characteristic pattern between primary and foreign keys. Although the characteristic pattern of symmetry and transitive relationships can be distinguished, some relationships may not have the same characteristics, although these relationships are expressed in a unary relationships. Example: Employees(IdEmp, NmEmp, MgrId), where IdEmp is the primary key, and MgrId is a

foreign key to the Employee table itself, which doubles as Id Manager. The Employees table does not have a symmetric relationship because if Alister is Chris's manager, it is impossible for the same Chris to be Alister's manager. Depending on domain semantics, a transitive relationship may or may not be a transitive relationship. If an employee's manager means another employee higher up in the organization, the manager is a transitive relationship. However, it is not a transitive relationship if it means only the direct supervisor.

The example clearly shows that it is inherently difficult to identify the relationship's logical characteristics in a relational schema without using domain knowledge. While our proposed approach can assist in identifying symmetric and transitive relationships in the unary relationships table, expert assistance in the knowledge domain can assist in establishing the actual semantics implied in these relationships.

V. Conclusion

Relational databases do not have explicit support for symmetric and transitive support. Database designers depend on data modelling patterns to be able to identify them. Meanwhile, ontology languages (e.g. OWL) provide grammar to explain symmetric and transitive relationships. Based on this, we investigated the research conducted to identify such relationships in relational databases.

We have proposed a way to identify symmetric and transitive relationships in unary relationships and transform them into Object Property components in OWL. The results of the transformation evaluation carried out using the Pellet reasoner show the expected conclusions from the symmetric and transitive relationship.

The main aim of this work is to study the OWL from relational databases to extract richer semantics. In future work, the tables in the database will be further analyzed to extract new relationships or concepts that may be implicit in the relational database.

References

[1] Y. Luo and C. Yu, "Development method of domain ontology based on reverse engineering," *2007 IEEE Int. Conf. Serv. Oper. Logist. Informatics, SOLI*, 2007.

[2] R. (eds. . Staab, S.; Studer, *Handbooks on Ontologies*. 2007.

[3] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 72–79, 2001.

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

[5] M. Li and X. Y. Du, "Learning Ontology From Relational Database," *Proceeding Fourth Int. Conf. Mach. Learn. Cybern.*, no. August, pp. 18–21, 2005.

[6] R. Ghawi and N. Cullot, "Database-to-Ontology Mapping Generation for Semantic Interoperability," *VDBL'07 Conf. VLDB Endow. ACM*, no. September 2007, pp. 1--8, 2007.

[7] J. Sequeda, "Translating SQL Application to the Semantic Web," in *In S.S. Bhowmick, J. Küng, and R. Wagner (Eds.) DEXA*, 2008, vol. 5181, no. May, pp. 450 – 464.

[8] Saeed M. Sedighi, "A novel method for improving the efficiency of automatic construction of ontology from a relational database," *Int. J. Phys. Sci.*, vol. 7, no. 13, pp. 2085–2092, 2012.

[9] J. F. Sequeda, S. H. Tirmizi, O. Corcho, and D. P. Miranker, *Survey of directly mapping SQL databases to the Semantic Web*, vol. 26, no. 4. 2011.

[10] N. Gherabi, K. Addakiri, and M. Bahaj, "Mapping relational database into OWL Structure with data semantic preservation," *Int. J. Comput. Sci. Inf. Secur.*, vol. 10, no. 1, pp. 42–47, 2012.

[11] I. Astrova, "Reverse engineering of relational databases to ontologies," *Proceeding 1st Eur. Semant. Web Symp. Grete, Greece*, pp. 327–341, 2004.

[12] N. Cullot, R. Ghawi, and K. Yétongnon, "DB2OWL: A tool for automatic database-to-ontology mapping," *SEBD 2007 - Proc. 15th Ital. Symp. Adv. Database Syst.*, pp. 491–494, 2007.

[13] Z. Lei and L. Jing, "Automatic Generation of Ontology Based on Database," *J. Comput. Inf. Syst.*, no. April 2011, 2011.

[14] L. Yiqing, L. Lu, and L. Chen, "Automatic Learning Ontology from Relational Schema," *IEEE Symp. Robot. Appl.*, pp. 592–595, 2012.

[15] I. Astrova, N. Korda, and A. Kalja, "Rule-Based Transformation of SQL Relational Databases to OWL Ontologies," *2nd Int. Conf. Metadata Semant. Res.*, pp. 415–424, 2007.

[16] M. A. G. Hazber, R. Li, X. Gu, and G. Xu, "Integration mapping rules: Transforming relational database to semantic web ontology," *Appl. Math. Inf. Sci.*, vol. 10, no. 3, pp. 881–901, 2016.

[17] B. Ben Mahria, I. Chaker, and A. Zahi, "A novel approach for learning ontology from relational database: from the construction to the evaluation," *J. Big Data*, vol. 8, no. 1, 2021.

[18] A. K. Silberschatz, F. Henry, and S. Sudarshan, *Database System Concepts*, no. c. McGraw-Hill, 2006.

[19] Ramez Elmasri and S. Navathe, *Fundamentals of*

Database Systems. Addison Wesley, 2011.

[20] U. Prot *et al.*, “A Practical Guide To Building OWL Ontologies Using Protege,” *Matrix*, no. April, pp. 0–107, 2011.

[21] S. Lipschutz and M. Lipson, *Schaum’s outline of theory and problems of discrete mathematics*. 1976.

[22] Rozen, “Discrete Mathematics and its Applications - Rosen - 0-07-288008-2.pdf.” .