Journal of Computer Science and Cybernetics, V.34, N.1 (2018), 77–96
DOI 10.15625/1813-9663/34/1/10110

TRANSFORMING EXTENDED ENTITY-RELATIONSHIP MODEL INTO OWL ONTOLOGY IN TEMPORAL DATABASES

VO HOANG LIEN MINH[1], QUANG HOANG

Hue University of Sciences, Hue University
E-mail: [1]minhvhl@gmail.com

Abstract. The objective of W3C is to transform the current Web into a Semantic Web to reuse the previous system. Many previous systems were built based on the ER model, so the upgradation and transformation of the ER model into ontology in order to reduce cost is really necessary. There are various studies aiming at transforming from ER and EER model into ontology; however, the studies have not still mentioned the transforming from the temporal databases into OWL ontology. Therefore, this paper proposes the rules for transforming the temporal components in TimeER model into OWL.

Keywords. Conceptual data model; ontology; OWL; recursive relationship; Semantic Web; TimeER model.

1. INTRODUCTION

In recent years, ontology has become a well-known term in the field of computer science for helping the machine "understand" and so that they could be able to process the information efficiently. Basically, ontology provides vocabulary to describe data which helps the computer understand their semantic. There are many languages designed to present ontology for semantic web. Among those ones, Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL) are the two popular languages. OWL is considered as an extension of the RDFS to get rid of the disadvantages of RDFS. Nowadays, OWL is being seen as the standard language for building ontology. OWL is a language describing the classes, properties and relationships between these objects in a way that computer can understand web contents.

The objective of World Wide Web Consortium (W3C) is to transform the current web into semantic web with to reuse the previous system [1]. Many previous systems were built based on the Entity - Relationship (ER) model, so upgrading and transforming the ER model into ontology to reuse the previous system and reduce the cost is really necessary.

There are several studies of transformation ER model into ontology. Upadhyaya and Kumar [2] introduced the ERONTO tool with the use of Java and Jena 2.1 for extracting ontology from ER schema. M. Fahad [1] proposed a method of designing the OWL ontology from ER model which based on a set of rules to transform the components of an ER model (entity, attribute and relationship) into the corresponding components in OWL.

Myroshnichenko [3] introduced a solution for automatically transforming ER model into correspondence semantic in OWL Lite Ontology.

In general, the studies have proposed an approach of transforming ER model into OWL ontology. However, these approaches have not formalized the transformation rules with the components of Extended Entity Relationship (EER) model (such as transformation of superclass/subclass relationship...), as well as mentioned the designing of the temporal ontology from the temporal ER model such as the proposed TimeER model by [4]. In addition, previous approaches have not proposed the transformation of the recursive relationship. So, this transformation into OWL ontology will affect preservation of the semantic of the ER model and its extension. Therefore, this paper will propose fully the rules for transforming the EER model into OWL ontology.

Accordingly, this paper is organized as follows: the next section presents a method for transforming EER model into OWL ontology. Section 3 classifies the recursive relationship and proposes the rules for transforming them into OWL ontology. Section 4 proposes a method for transforming TimeER into OWL based on additional rules for transformation of the temporal component of TimeER model into OWL ontology. Section 5 concludes and discusses implications for future works.

2. TRANSFORMATION OF EER MODEL INTO OWL ONTOLOGY

In order to illustrate the transformation of the EER model to OWL ontology and to informally validate our transforming rules, this paper transforms a real-world example of an EER model such as Figure 1 (modeling an employee management system) into an OWL ontology.

Management system is used to store and manage information in a unit. The main components of the EER model are the *Person*, *Employee*, *Department*, *Project* entities with corresponding simple and composite attributes. In it, the entity *Empoloyee* requires that the system manages both the lifespan and transaction time (denoted LT) to record the time that an entity exists in reality and the time that the entity/event is in the database. The entity *Department* is required to manage transaction time. The attribute *Salary* is required to handle the valid time and transaction time (BiTemporal, denoted BT), to record the time that an event is true in reality and the time that the entity/event is in the database. Similarly, the multivalued attribute *Location* is also required to be managed on a valid time. The relationship *Work_for* is required to handle a valid time to record the time that an event is true in reality.

Figure 18 is the result of applying transformation rules. Ontology consists of classes corresponding to the entities of the ER model and object properties corresponding to ER relationship. All of the cardinality constraints in the ER model have been correctly transformed to equivalent OWL restriction.

2.1. Transformation of entities type

In the ER model, entities are used to describe complex constructs such as people, places, things or events of interest. In OWL, the class provides an abstract mechanism for representing objects with similar properties. We have a transformation rule as follows:

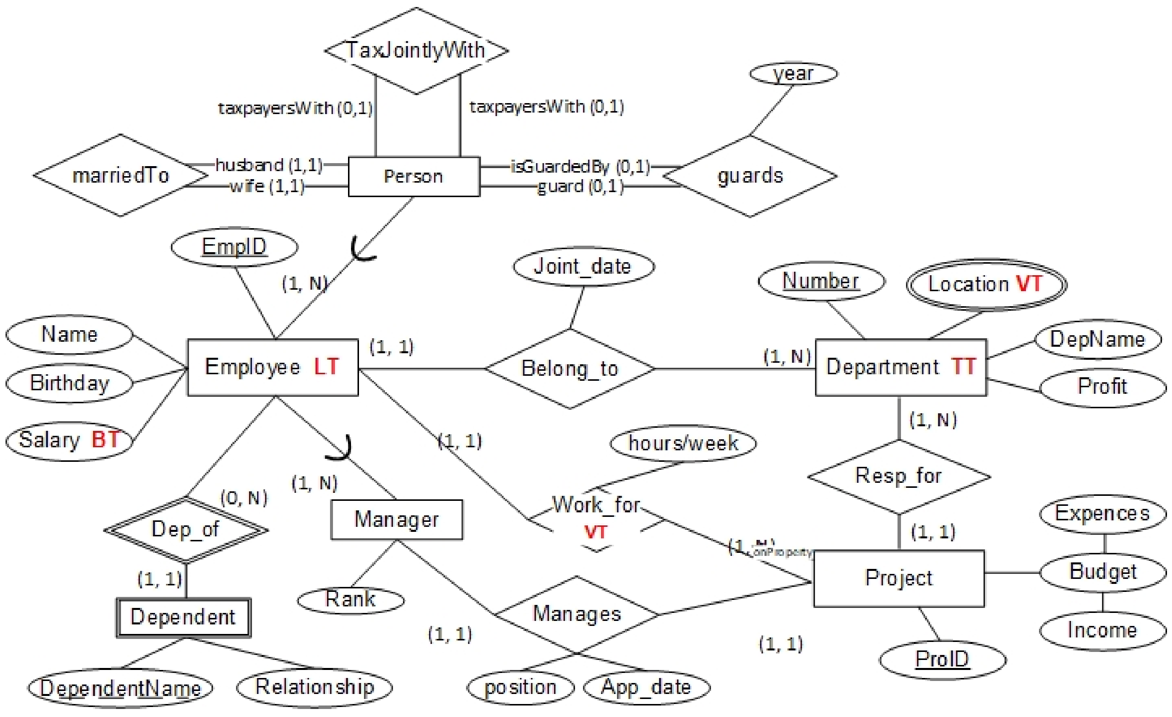


Figure 1. An example of TimeER model

Rule EER1. Transform each entity type E into class $C(E)$ by OWL ontology [5].

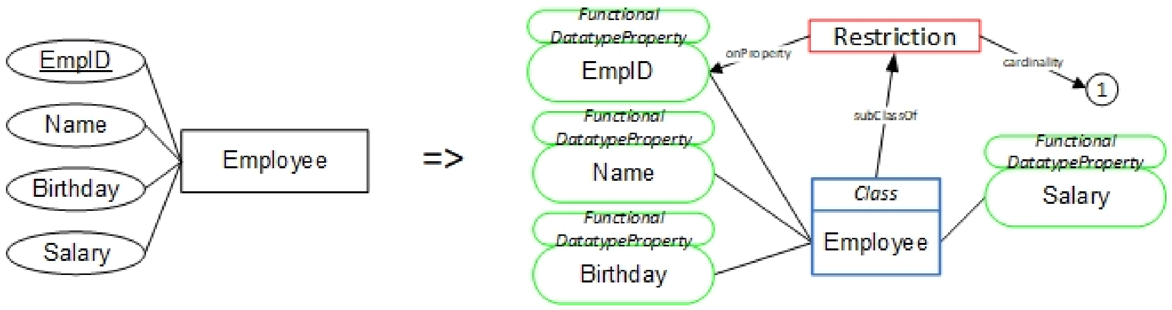


Figure 2. Transformation of entity type and attribute

In Figure 2, the entity *Employee* was transformed into class *Employee* in OWL ontology.

2.2. Transformation of attributes

In OWL, the datatype attribute is a single valued property, so they are equivalent to the single valued attribute in the ER model. And to define a single valued property and have only one value, set it with a cardinality constraint. In the ER model, the primary key is used to uniquely identify an entity. It aims to ensure the condition of a single attribute with a single value, and it is impossible to be a null and must be distinct value. To represent the primary key without getting a null value in OWL, set *minCardinality* of the attribute to 1 and *maxCardinality* to 1, meaning that the *cardinality* is 1. We have a transformation rule as follows:

Rule EER2. The single_valued attribute *attE* of the entity *E* was transformed into the datatype property *attE* with range is corresponding with data type in OWL, and domain is *E*, set function characteristics for datatype property *attE* [4]. If the attribute *attE* is not NULL then set minimum cardinality constraint to 1 for datatype property *attE*. For each key attribute *KE* of entity *E*, transform to datatype property *KE* in the set of key properties of class $C(E)$, set function characteristics and cardinality constraint is 1 [5].

In Figure 2, the attributes *Name*, *Birthday*, *Salary* will be transformed into datatype properties, with range is corresponding with data type in OWL and domain is *Employee* class, set function characteristics for this properties.

The key attribute *EmpID* will be transformed to datatype property *EmpID* in the set of key properties of class *Empolyee*, domain is *Employee* and ranger is *string*. Besides, it is set function characteristics and cardinality constraint is 1 for the datatype properties EmpID by using syntax *owl:cardinality* in Figure 2.

In the ER model, multi-valued attributes can have multiple attribute values. In OWL, by default, the OWL property allows for multiple values.

Rule EER3. Transform multi_valued *attE* of entity *E* similar Rule EER1, but not set function characteristics [5].

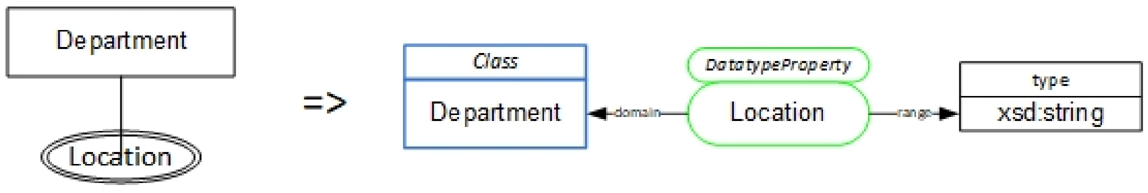


Figure 3. Transformation of multi-valued attributes

In Figure 3, the multi_valued attributes *Location* will be transformed into datatype properties, with range is corresponding with data type in OWL and domain is *Department* class, but not set function characteristics for this properties.

2.3. Transformation of composite attribute

Composite attributes are attribute that are made up of different single attributes. In OWL, it is possible to represent complex properties by setting them as child datatype properties. We have a transformation rule as follows:

Rules EER4. The composite attribute *attE* of the entity *E* has the subattributes *sub_attE* will be transformed to datatype property *attE* of class $C(E)$. The subattributes *sub_attE* will be transformed to datatype properties *sub_attE*, and they are subproperties of datatype properties *attE*, with function characteristics, domain is the datatype properties *atttE* and range is corresponding with data type in OWL [1].

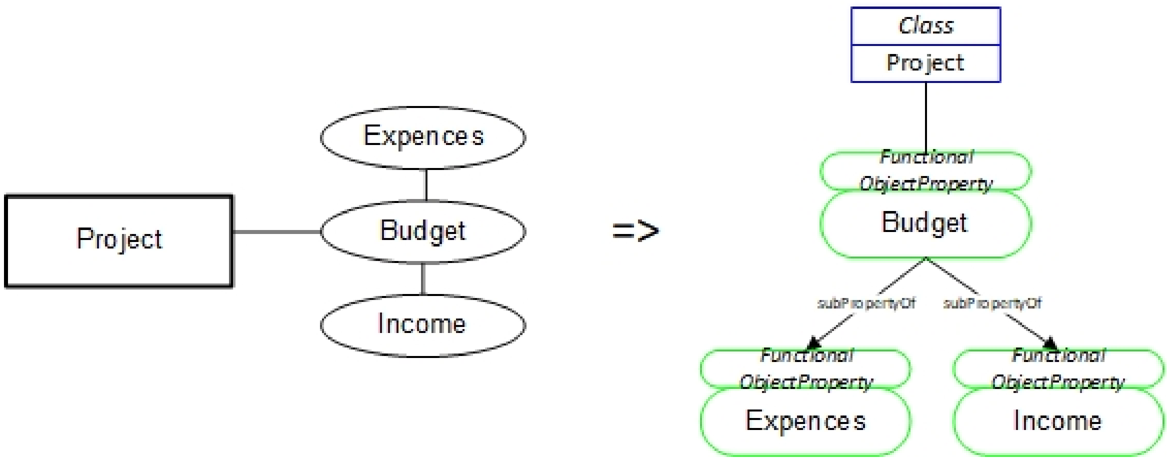


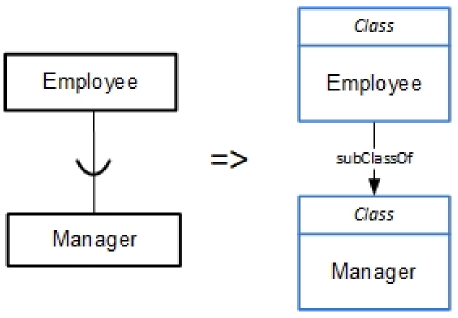
Figure 4. Transformation of composite attribute

The composite attribute *Budget* has two subattributes *Expences* and *Income* in Figure 4 will be transformed into datatype property *Budget*, and two subdatatype properties *Expences* and *Income*.

2.4. Transformation of an inheritance relationship

To represent the disjointness specialization, we can use disjoint constraints between classes that correspond to subentity.

Rule EER5. The subentity *sub_E* inherits from the entity *E* will be transformed into subclass (inheritance) in the corresponding class of the super entity.



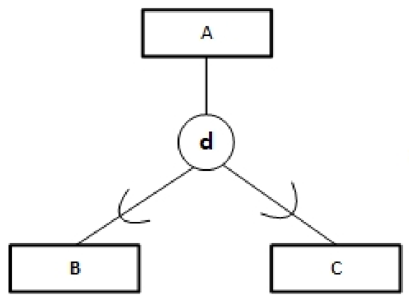
```
<Class rdf:about="Manager">
<rdfs:subClassOf rdf:resource="Employee"/>
</Class>
```

Figure 5. Transformation of an inheritance relationship

In Figure 5, the entity *Manager* was transformed into subclass *Manager* in class *Employee*, and OWL source code as follows.

2.5. Transformation of disjoint and overlap specializations

Rules EER6. Transform each subentity sub_E_i of a subclass in a disjoint specialization E into corresponding subclass $C(sub\_E_i)$ by using syntax *owl:disjointWith*, set subclasses that do not intersect with other subclass inheriting from that disjoint class E [2].



=>

```
<owl:Class rdf:Id="A">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#B" />
    <owl:Class rdf:about="#C" />
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:Id="B">
  <rdfs:subClassOf rdf:resource="#A" />
  <owl:disjointWith rdf:resource="#C"/>
</owl:Class>
```

Figure 6. Transformation of disjoint specialization

In consideration of the entity A, there are two disjoint classes including B and C. We have applied rule **Rule EER6**. in order to transform for two classes B and C by using the syntax *owl:disjointWith* and OWL source code as follows, shown in Figure 6.

In OWL, none of the constraints has the same meaning as the overlapping in the ER model. Therefore, in order to represent this constraint, we can use the union constraint. We have a transformation rule as follows:

Rule EER7. Transform the subclass sub_E_i in the total completeness constraint with super class E to class $C(E)$ and subclass $C(sub\_E_i)$. Using syntax *owl:unionOf* for setting those subclass $C(sub\_E_i)$ [2].

2.6. Transformation of the weak entity and identifying relationship

In OWL, representation of a weak entity is similar to a strong entity, the only difference is that it adds two inverse object properties, adds the owner key properties into the set of key properties of the weak entity class. We have a transformation rule as follows:

Rule EER8. Let us consider that W is a weak type of identifying relationship R and the owner entity type is E. Suppose that W has partial key KW, and KE is a key of E. A weak entity always participates in the identifying relationship with cardinality constraint (1, 1), therefore, depending on the second cardinality constraint of the identifying relationship R, we will construct the transform rules as follows:

- Add the class $C(W)$, the attributes $attW$ of the weak entity type W will transform into datatype properties $attW$ of class $C(W)$ similar to the transformation of the attributes of the strong entity.

- Add two inverse object properties $EhasW$ and $WOfE$ which show relationship between classes $C(E)$ and $C(W)$, their identities, ranges and domains as shown in Table 1. Set function characteristics and minimum constraint is 1 to two properties $WOfE$;

- Depending on the second cardinality constraint of the identifying relationship, add the corresponding min/max to the object properties $EhasW$ (the object properties just added with domain is class corresponding with entity type which has pairs of cardinalities).

- Key of class $C(W)$ was created by combining the partial key KW with key KE of class $C(E)$.

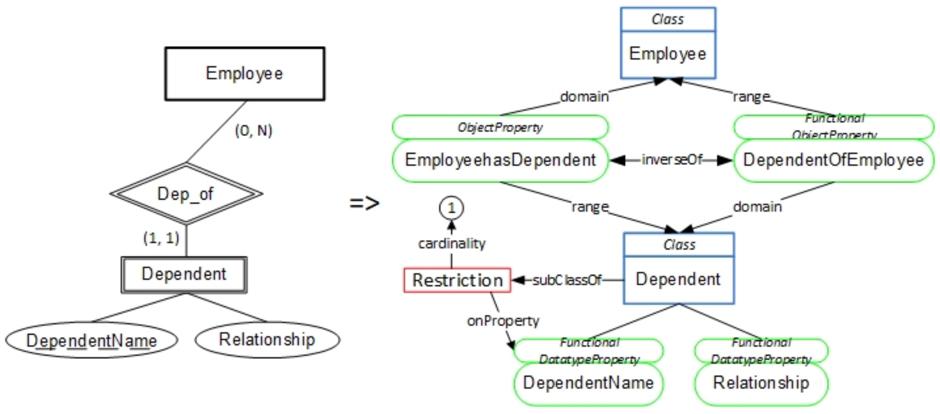


Figure 7. Transformation of the weak entity and identifying relationship

Table 1. Pairs properties added when transformation of the identifying relationship

Identity	**Domain**	**Range**
EhasW	C(E)	C(W)
WOfE	C(W)	C(E)

As an example in Figure 7, the weak entity *Dependent* was transformed into class *Dependent*, the indentifying relationship *Dep_of* was transformed into pairs of inverse object properties *EmployeeHasDependent* and *DependentOfEmployee* for presenting the indentifying relationship. Two attributes *DependentName* and *Relationship* of the weak entity will transfomed into the datatype properties of class *Dependent*.

2.7. Transformation of the relationship

In ER model, the relationship is how the data is shared between entities. The degree of a relationship type is the number of entity types that participate. Next section will consider three types of relationships between entities: the binary relationship, n-ary relationship and recursive relationship.

2.7.1. Transformation of the relationship without attributes

The binary relationship has two cardinality constraint of the form (x, y), where x is a minimum constraint, y is maximum constraint. Typically, x is 0 or 1, and y is 1 or N [6].

Rule EER9. Let us consider the binary relationship R without attributes between two entities E_1 and E_2 with left cardinality constraint is (x_1, y_1) and right cardinality constraint is (x_2, y_2). We have transformation rules as follows:

- Add two inverse object properties for presenting the relationship between classes $C(E_1)$ and $C(E_2)$;

- With each value of cardinality constraint, if min differential 0 or max deferential N on the relationship R, add min/max restriction to the corresponding object properties.

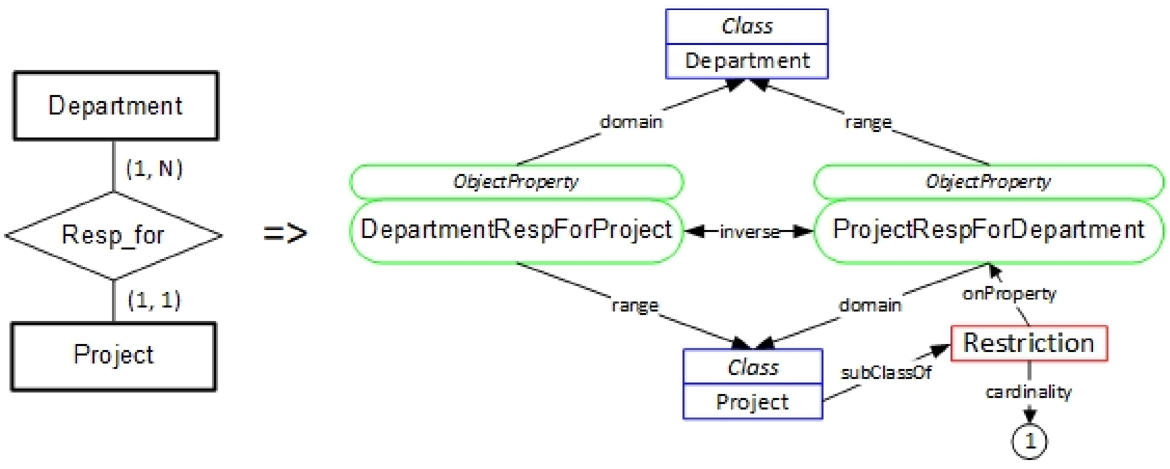


Figure 8. Transformation of the relationship without attributes

In Figure 8, the relationship *Resp_for* between two entities *Department* and *Project* will be transformed into two inverse object properties *DepartmentRespForProject* and *ProjectRespForDepartment*. Because the left cardinality constraint is (1, 1), we set min and max restriction is 1 to object property *ProjectRespForDepartment*.

2.7.2. Transformation of the relationship with attributes

A relationship can also have its own attributes (especially $n - n$ relationships). The attributes of a relationship are single_valued attributes. In this case, where the relationship R has an attribute, if R is a 1-1 relationship, then it can move it into the attribute of one of two entities, and if R is a 1-n relationship, moving this attribute into the attribute of the entity which has maximum constraint.

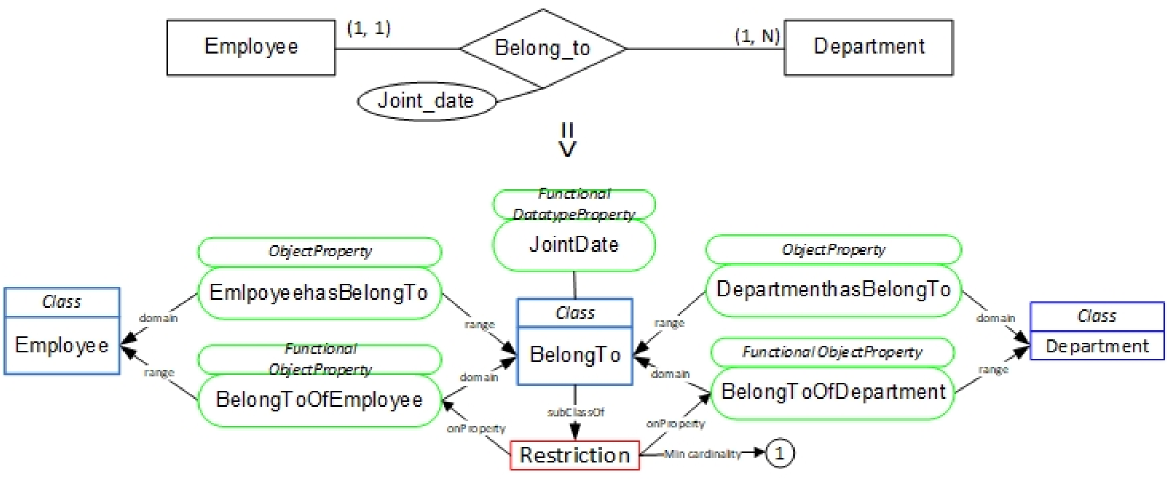


Figure 9. Transformation of the relationship with attributes

However, if the attribute of a relationship is moved into either entity or represented in the OWL, then semantic are not guaranteed. We can treat a relationship as an entity or attribute, the designer can determine the relationship as an entity to ensure semantics during transformation. We have a transformation rule as follows:

Rules EER10. Let us consider the binary relationship R with attributes *attR* between entities E_1, E_2, we have a transformation rule as follows:

- Add class $C(R)$, the attributes *attR* of the relationship R was transformed into the datatype properties of class $C(R)$.

- Add two inverse object properties E_1*HasR* and *ROfE*$_1$ for representing the relationship between classes $C(R)$ and $C(E_1)$, two inverse object properties E_2*HasR* and *ROfE*$_2$ for representing the relationship between classes $C(R)$ and $C(E_2)$. Set function characteristics and minimum constraint is 1 to two properties *ROfE*$_1$, *ROfE*$_2$;

- With each value of min/max constraint, if min differential 0 and max differential N on the relationship R, add corresponding min/max constraint to the object properties E_1*HasR* v E_2*HasR*. If the binary relationship is $N - N$, add two object properties *ROfE*$_1$, *ROfE*$_2$ to the key set of properties of class R [3].

As shown in Figure 9, the relationship *Belong_to* with attribute *Joint_date* will be transformed into class *BelongTo*. Two inverse object properties *EmployeeHasBelongTo*, *Belong-ToOfEmployee* represent the relationship between an entity *Employee* and a relationship *Belong_To*. Likewise, two inverse object properties *DepartmentHasBelongTo*, *BelongToOf-Department* represent the relationship between an entity *Department* and a relationship *Be-long_To*. The attribute *Joint_date* was transformed into the datatype properties *JointDate* of class *BelongTo*.

2.7.3. Transformation of *n*-ary relationship

Thanks to the transforming rules of the binary relationship, we can apply those rules for the transformation of the n-ary relationship R, by considering R as an entity type, and this entity has the binary relationship with the participating entity types E_i. So these binary relationship can only be one-to-one or one-to-many binary relationship (since the cardinality constraint of the entity type R with all these binary relationships is (1, 1)). We have a transformation rule as follows:

Rule EER11. Let us consider the n-ary relationship among the entity types E_i

- Add class $C(R)$, the attributes of the relationship R are transformed into the datatype properties of class $C(R)$;

- Add pairs of the inverse object properties which show the relationship between class $C(R)$ and classes E_i, set the minimum and maximum constraint. Set 1 for min and max constraint of the inverse object properties.

- Add the object properties with domain is $C(R)$ which has just added to set of key properties of class $C(R)$ [3].

Note: If the relationship R has function constraint, remove the object properties which has range corresponding with the entity appeared on the right of the function constraint out of the set of key properties of class $C(R)$.

3. TRANSFORMATION OF THE RECURSIVE RELATIONSHIP

3.1. Classification of the recursive relationship

In a recursive relationship, a set of instances can take on a unique role of two different roles in the same relationship [7]. Checking the role allows us to classify all of the recursive relationships: symmetric or asymmetric.

A recursive relationship is symmetric when all of instance participating in the relationship has a unique role and has the same semantics [8]. Accordingly, if R is a recursive relationship which is symmetric then $role_1 \equiv role_2$, in which $role_1$ and $role_2$ are the name of two roles of the recursive relationship R. Conversely, if R is not symmetric, we call this relationship is asymmetric.

For example, the recursive relationship such as *dancePartnerOf*, *spouseOf*, *siblingOf* and *isFriendOf* are the symmetric relationships.

Formally, we can define a symmetric recursive relationship as follows: There is the recursive relationship R on the entity type E, then R is called the symmetric if:

$\forall$ e_1, $e_2 \in E$: if $(e_1, e_2) \in R$ then $(e_2, e_1) \in R$.

Conversely, R is called the asymmetric, if:

$\exists$ e_1, $e_2 \in E$: $(e_1, e_2) \in R$ and $(e_2, e_1) \notin R$.

3.2. Transformation of the symmetric recursive relationship without attributes

With a symmetric recursive relationship without attributes, because there is only one role, transforming into OWL similar by as transforming the binary relationship, but only add an object property.

Rule EER12. For the symmetric recursive relationship R without attributes on the entity E with role *Role*, we have transformation rules as follows:

- Add the object property with its name is the name of role in the relationship, domain and range is class $C(E)$, set the symmetric characteristics and other possible characteristics may have of the relationship;

- With the relationship 1:1, add the maximum cardinality restriction by 1;

- With the minimum cardinality constraint is 1-1, add minimum cardinality restriction by 1 to this object property.

As in Figure 10, a person may be able to choose whether to pay taxes together or not. This relationship is symmetrical because there will be two cases:

- Two people e_1, e_2 are joined in the relationship: e_1 pay the same as e_2 and e_2 pay the same as e_1.

- Two people e_1, e_2 are not joined in the relationship: e_1, e_2 do not pay taxes with anyone.

Apply these transformation rules of the symmetric recursive relationship 1:1 *TaxJointly-With* of the entity type *Person* as in Figure 10.

3.3. Transformation of the symmetric recursive relationship with attributes

With symmetric recursive relationship with attribute, consider the relationship as an entity, so when transforming into OWL similarly to transforming the binary relationship with attribute, by adding new class and two inverse object properties.

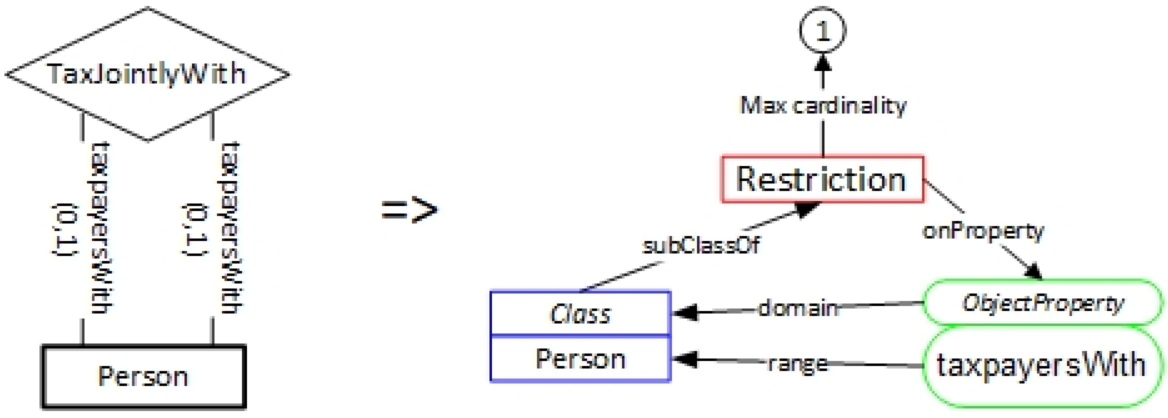


Figure 10. An example of transformation of the symmetric recursive relationship without attributes

Rule EER13. Let us consider the symmetric recursive relationship R with attribute *attR* of the entity type, we have the transformation rules as follows:

- Add the class $C(R)$, the attributes *attR* of the relationship is transformed into the datatype properties *attR* of class $C(R)$;

- Add two inverse object properties *EHasR*, *ROfE* representing the relationship between class $C(R)$ and class $C(E)$. Two properties are reflexive and other characteristics of the relationship R. The object properties *ROfE* has min and max restriction is 1;

- Set min/max restriction on the object properties *EHasR* corresponding with value on role of the min/max cardinality constraint which different from 0 and N. If R is the binary relationship $N : N$, add the object properties *ROfE* to set of key properties of class $C(R)$.

3.4. Transformation of asymmetric recursive relationship without attributes

Rule EER14. Considering the symmetric recursive relationship R without any entity on the entity type E with two roles are $role_1$ and $role_2$, we have the transformation rules as follows:

- Add a pair of inverse object properties with its name are $role_1$ and $role_2$, domain and range is the class E; which has the characteristics of corresponding relationship;

- Each role with min/max cardinality constraint different from 0 and N, add the min/max cardinality restriction corresponding to the object properties.

Considering the entities *Person* is the set of married people and the relationship *marriedTo*. One of the married people is married to only one person. That is, e_1 married e_2, in contrast e_2 married e_1. Two people e_1, e_2 are not married to any other person. Apply these transformation rules of the asymmetric recursive relationship without attribute *marriedTo* of the entity type *Person* as in Figure 11.

3.5. Transformation of the asymmetric recursive relationship with attributes

With an asymmetric recursive relationship with attributes, in consideration of the relationship as an entity, in order to ensure semantics during transformation, it is necessary to

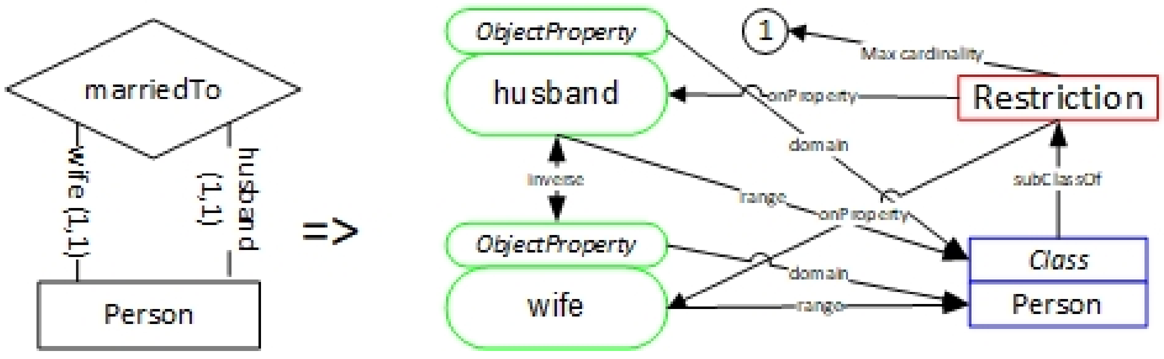


Figure 11. An example of transformation of the asymmetric recursive relationship without attributes

transform the asymmetric recursive relationship into a new class, and the attribute of the asymmetric recursive relationship that transforms into the datatype property of the newly class. Because the relationship has two different roles, it is necessary to create two pairs of two inverse object properties which represent the two roles of the relationship.

Rule EER15. Considering the asymmetric recursive relationship R with attribute *attR* on the entity type E with two roles $role_1$ and $role_2$, we have transformation rules as follows:

- Add the class $C(R)$, attributes *attR* of the relationship R is transformed into the datatype properties *attR* of class $C(R)$;

- Add two inverse object properties $Erole_1R$, $Rrole_1E$ representing a relationship between class $C(R)$ and $C(E)$ with roles $role_1$. Add two inverse object properties $Erole_2R$, $Rrole_2E$ representing relationship between class $C(R)$ and $C(E)$ with roles $role_2$. Set min/max cardinality restriction for two properties $Rrole_1E$ and $Rrole_2E$ is 1;

- Each min/max cardinality different 0 and N on role $role_1$ of the relationship R, add min/max cardinality restriction corresponding to the object properties $Erole_1R$;

- Each min/max cardinality different 0 and N on role $role_2$ of the relationship R, add min/max cardinality restriction corresponding to the object properties $Erole_2R$;

- If the relationship R is $N : N$, add two object properties $Rrole_1E$, $Rrole_2E$ to the set of key properties of class $C(R)$.

Consider the relationship guards on the entity type *Person*. One can guard one person or no one, whereas one can be guarded by one person or without guard. This relationship contains two roles: guard and *isGuardedBy*. Apply these transformation rules of the asymmetric recursive relationship with attribute guards of the entity type *Person* as in Figure 12.

4. TRANSFORMATION OF TimeER INTO OWL ONTOLOGY

TimeER model is an extension of the ER model; the construction of the ontology to represent for the temporal aspects from TimeER model is the problem inspiring many researchers. Thus, the transformation TimeER model into OWL ontology will be as follows: first, transform the components without temporal aspects on TimeER model (including the temporal entity type) into OWL ontology. After that, create the OWL ontology to represent for the temporal aspects in TimeER model. The purpose of this step is to represent the

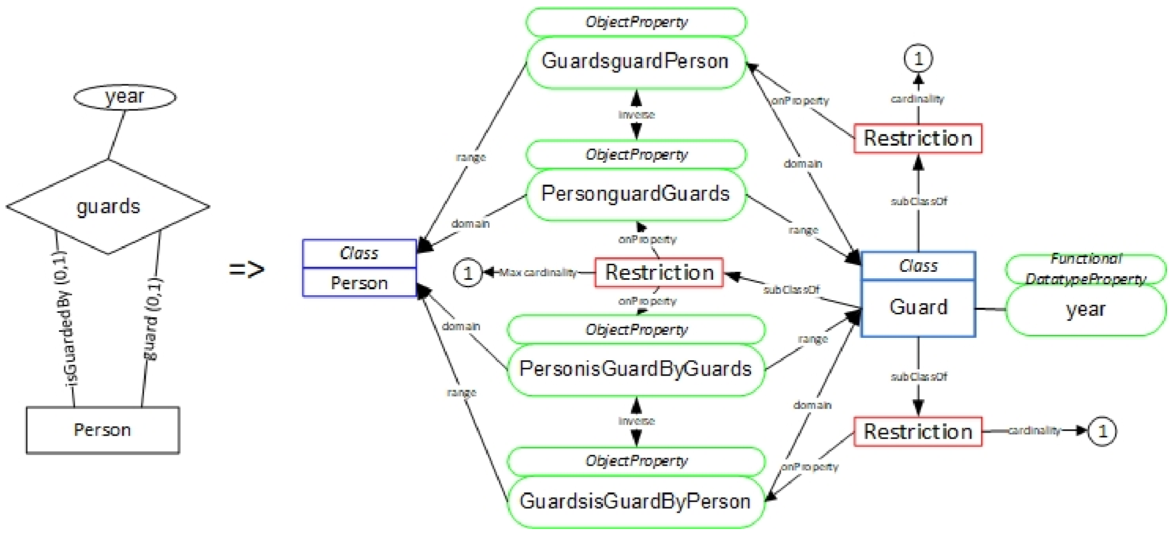


Figure 12. An example of transformation of the asymmetric recursive relationship with attributes

data and temporal aspects constraint on TimeER model. Finally, transform the temporal components on TimeER model into OWL ontology.

4.1. Initially ontology for representing the temporal aspect

In OWL, no component has the same meaning as the temporal components in the TimeER model. To support the presentation of temporal aspects in OWL, we need to create an additional class *InstantDateTime* and object properties that represent the relationship between the class *owl:Thing* class and the class *InstantDateTime*.

Create class *InstantDateTime* representing for a timeline. In this class, create the functional datatype property *hasDateTime* with minimum cardinality restriction set to 1. This propertys range is *xsd:dateTime* and it is the key properties of a class *InstantDateTime*.

Create six functional object properties with minimum cardinality restriction set to one: *hasVTs*, *hasVTe*, *hasLSs*, *hasLSe*, *hasTTs*, *hasTTe*; range set to class *InstantDateTime* and domain set to the class *owl:Thing*. Six properties represent the relationships between class *owl:Thing* and class *InstantDateTime*.

4.2. Transformation of the temporal aspect on TimeER model

4.2.1. Transformation of temporal entity type

To represent the temporal aspects of an entities in the TimeER model, we can add a class, two inverse object properties, and some generated attributes.

Rule TimeER1: Each temporal aspect XX of the entity type E:

- Add the class $C(E\_XX)$;

- Add two inverse object properties: *EHasXX* with domain is class $C(E)$ and range is class C(E_XX); XXOfE with domain is class C(E_XX) and range is class C(E), simultaneously XXOfE has functional characteristics and minimum cardinality restriction is 1;

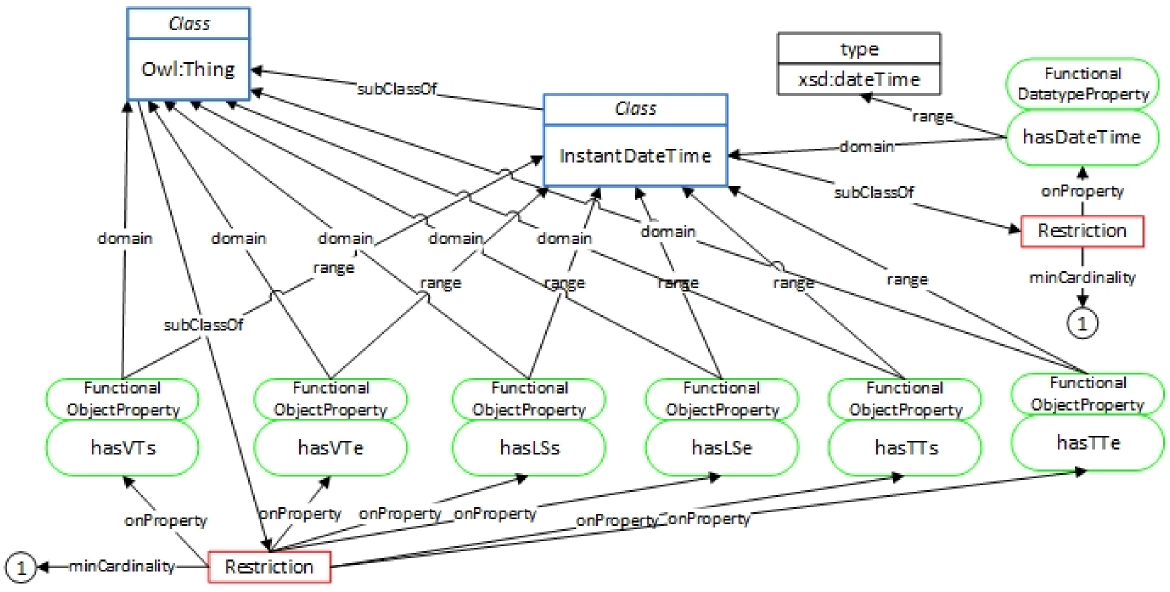


Figure 13. Ontology represents the temporal aspects

- Set of key properties of class C(E_XX) include properties XXOfE and some properties for representing the temporal restriction which depends on temporal aspect *XX* as in Table 2.

Proof:

1. According to [4], relating to the transformation of the temporal entity type of TimeER model into ER model, the entity type E with temporal aspect XX is transformed into the non-temporal entity type E which has a non-temporal attributes XX. Simultaneously, create the weak entity type $W(E\_XX)$ which has the identifying relationship with the entity E, the purpose is to represent for the support temporal aspect XX for each entity type E. The partial key of the weak entity type $W(E\_XX)$ includes some attributes for representing the temporal constraint, depending on type of the temporal aspect XX.

2. According rule EER8, the entity type E is transformed into class $C(E)$, the attributes of the entity type E are transformed into the datatype properties of class $C(E)$. The weak entity type E_XX is transformed into class $C(E\_XX)$, add two inverse object properties *EhasXX* and *XXOfE* representing for the relationship between class $C(E)$ and class $C(E\_XX)$ with domain and range as shown in Table 1.

From (1) and (2), it is proved. ■

To manage the time that an entity exists in reality and the time that the entity/event is in the database, so the temporal aspect of the entity type *Employee* was transformed to class *Employee_LT* and two inverse object properties shown in Figure 14.

We add the class *Employee_LT* and two inverse object properties which describe the relationship between the two classes. In particular, the min cardinality of the object property

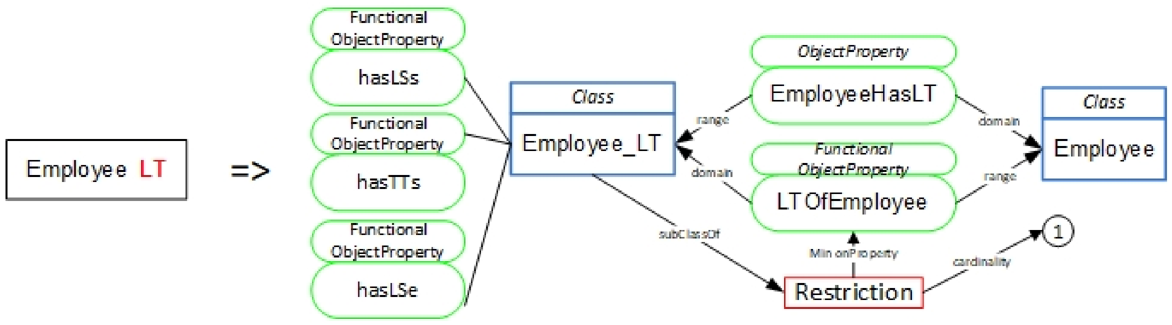


Figure 14. Transformation of the temporal aspect of the entity type

Table 2. The key attributes corresponding with the temporal aspect

Temporal aspect	**The key attributes**
VT	hasVTs
LS	hasLSs
TT	hasTTs
LT	hasLSs, hasLSe, hasTTs
BT	hasVTs, hasVTe, hasTTs

LTOfEmployee is set to 1 to ensure that an individual in the class *Employee* is associated with an individual in the class *Employee_LT*.

4.2.2. Transformation of the temporal attributes of the entity type

In the case of temporal attribute of an entity, in consideration of the attribute as an entity, during the transformation into OWL, we add new class and two inverse object properties.

Rule TimeER2. Each attribute *attA* has the temporal aspect *XX* of the entity type *E*:

- Create class $C(attA\_XX)$, transform attribute *attA* to the datatype properties of class $C(attA\_XX)$;

- Create two inverse object properties: *attAHasXX* with domain is class $C(E)$ and range is class $C(attA\_XX)$; *XXOfattA* with domain is class $C(attA\_XX)$ and range is class $C(E)$. *XXOfattA* has functional characteristics and minimum cardinality restriction is 1;

- Set of the key properties of class $C(attA\_XX)$ includes property *XXOfattA* and some properties which represent the temporal restriction depend on the temporal aspect *XX* as shown in Table 2.

Proof:

3. Accroding [4], relating to the transformation of the temporal entity type of TimeER model into ER model, the attribute *attA* with temporal aspect *XX* is transformed into the weak entity type $W(attA\_XX)$ which has the identifying relationship with the entity *E*, the purpose is to represent the temporal aspect *XX* of attribute *attA* for

each the entity type E. The weak entity type $W(attA\_XX)$ have the attributes $attA$, partial key and some attributes for representing the temporal constraint, depending on the type of the temporal aspect XX.

4. According to rule EER8, the entity type E is transformed into class $C(E)$. The weak $W(attA\_XX)$ is transformed into class $C(attA\_XX)$, add two inverse object properties $attAHasXX$ and $XXOfattA$ which represent the relationship between class $C(E)$ and class $C(attA\_XX)$, their domain and range as shown in Table 1.

From (3) and (4), it is proved. ■

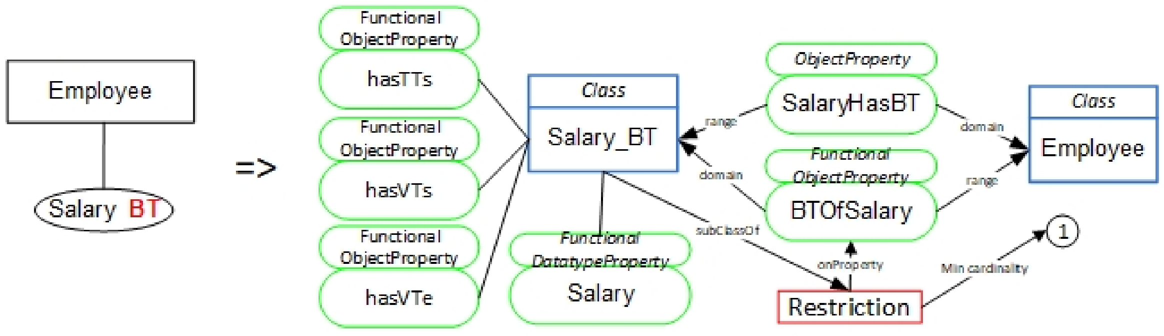


Figure 15. Transformation of the temporal attributes of the entity type

In Figure 15, the attribute *Salary* with temporal aspect *BT* will be transformed to class *Salary_BT* and two inverse object properties, the attribute *Salary* will be transformed to the datatype properties of class *Salary_BT*.

4.2.3. Transformation of the temporal relationship

In the case of transforming of a temporal relationship, the relationship is considered as an entity. To represent a temporal relationship in the TimeER model, we can add new class, two inverse object properties, and some generated attributes.

Rule TimeER3. Each relationship R has the temporal aspect XX among the entities E_i:

- Create class $C(R)$;

- Corresponding to each entity type E_i in the relationship R, create two inverse datatype properties for representing the relationship between class $C(R)$ and class $C(E_i)$: E_iHasR with domain is class $C(E_i)$ and range is class $C(R)$; *ROfEi* with domain is class $C(R)$, range is class $C(E_i)$, set functional characteristics and minimum cardinality restriction is 1. If min/max cardinality constraint of the entity E_i different from 0 and N then add the min/max cardinality restriction corresponding to property $EiHasR$;

- If R is the 1-1 binary or recursive relationship then key of class $C(R)$ consists of one of the two newly added whose range is the class corresponding with the entity in relationship R and some properties representing for the temporal restriction depend on the temporal aspects XX as shown in Table 2.

Proof: Because the relationship R is seen as an entity type $E(R)$ then the temporal relationship becomes the temporal entity type. Therefore, apply Rules TimeER2, Rule TimeER3 is generated. Then, corresponding with instance of the relationship between entity $e \in E_1$ and $f \in E_2$ is an entity of the entity type $E(R)$.

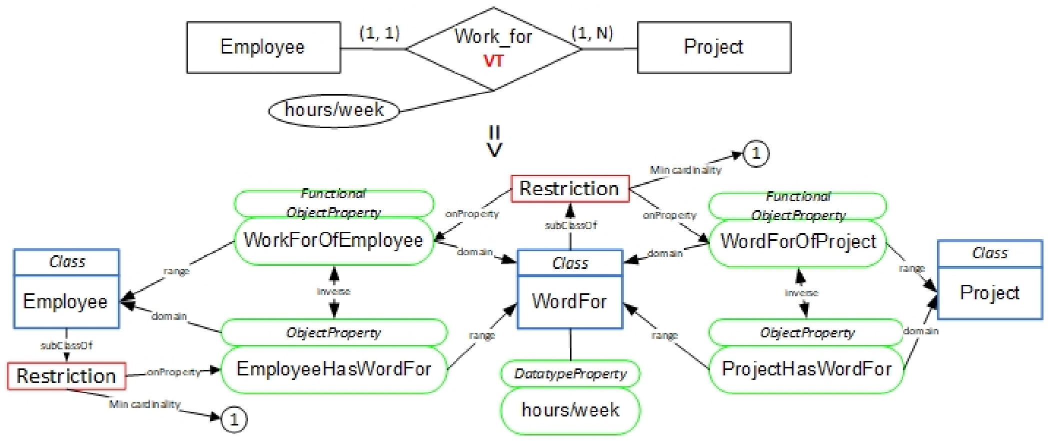


Figure 16. Transformation of the temporal relationship

In the relationship *Worrk_for* with temporal aspect VT to manage a valid time to record the time that an event is true in reality, we transform to class *WorkFor*, and add two pair inverse object properties show in Figure 16. ■

4.2.4. Transformation of the temporal attribute of the relationship

Rules TimeER4. Each attribute *attR* has the temporal aspect XX of the relationship R:

- Add class $C(attR\_XX)$, the attribute *attR* is transformed into the datatype property of class $C(attR\_XX)$ following the rules EER2, EER3 and EER4;
- Add two inverse object properties that represent the relationship between class $C(R)$ and $C(attR\_XX)$: *attRHasXX* with domain is class $C(R)$ and range is class $C(attR\_XX)$; *XXOfattR* with domain is class $C(attR\_XX)$, range is class $C(R)$, and has functional characteristics. Minimum cardinality restriction of two properties set to one;
- Key of class $C(attR\_XX)$ consists of the properties *XXOfattR* and some properties which represent the temporal restrictions depend on the temporal aspect XX as shown in Table 2.

Proof: Because the relationship E is seen as an entity type $E(R)$, so the temporal attribute of the relationship will be the temporal attribute of the entity type $E(R)$. Therefore, apply Rule TimeER2, Rule TimeER4 is generated.

In the relationship in Figure 17, the manager takes over the management positions in the project. Each manager is in a different position. Each person can take many different positions in the projects they participate in. So we transform the temporal aspect VT of the attribute *Position* to class *Position_VT*, and the attribute *Position* was transformed into the datatype property of class *Position_VT*. ■

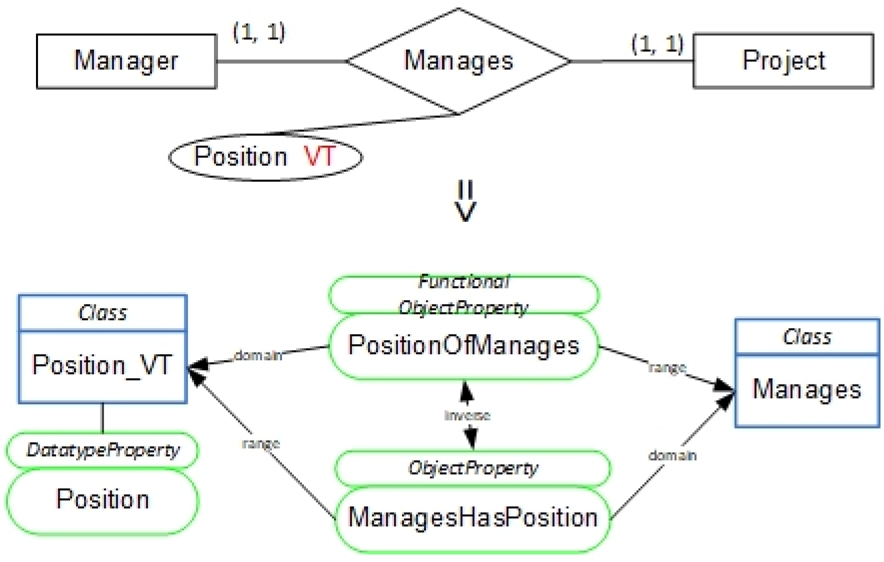


Figure 17. Transformation of the temporal attribute of the relationship

5. REVIEW

Based on the set of rules presented above, we can easily transform any EER model to OWL ontology. It can be seen that in order to transform the entity, the rules EER1, and EER5 - EER7 rules should be applied; and for the properties transformation, the rules EER2-EER4 are applied; for the relationship transformation, the rules EER9 - EER15 are applied.

Suppose that n is the number of entities type, l is the number of attributes of each entity type, m is the number of relationships in the model. We see that if the above-mentioned transformation entity rules are applied, the complexity of algorithm will be O(n), the complexity of the attribute transformation will be O($n \times l$) and the complexity of the transformation of the relationship will be O(m). Since the complexity of each rule is O(1), the complexity of the transformation method will be O(max (m, $n \times l$)).

6. CONCLUSIONS

This paper has proposed an approach to build the OWL ontology from EER model by using the rules to transform EER and TimeER model into OWL ontology. Particularly, this research focuses on clarifying, and categorizing semantics of the recursive relationships on the ER model, thereby put forward the rules for transformation of recursive relationship. Furthermore, the rules for transformation of TimeER model into OWL ontology, such as the temporal entity type, the temporal attributes of the entity type, the temporal relationship, and the temporal attribute of the relationship are also presented clearly. The transformation rules by the virtual model is presented, and the OWL ontology results is verified on Protg.

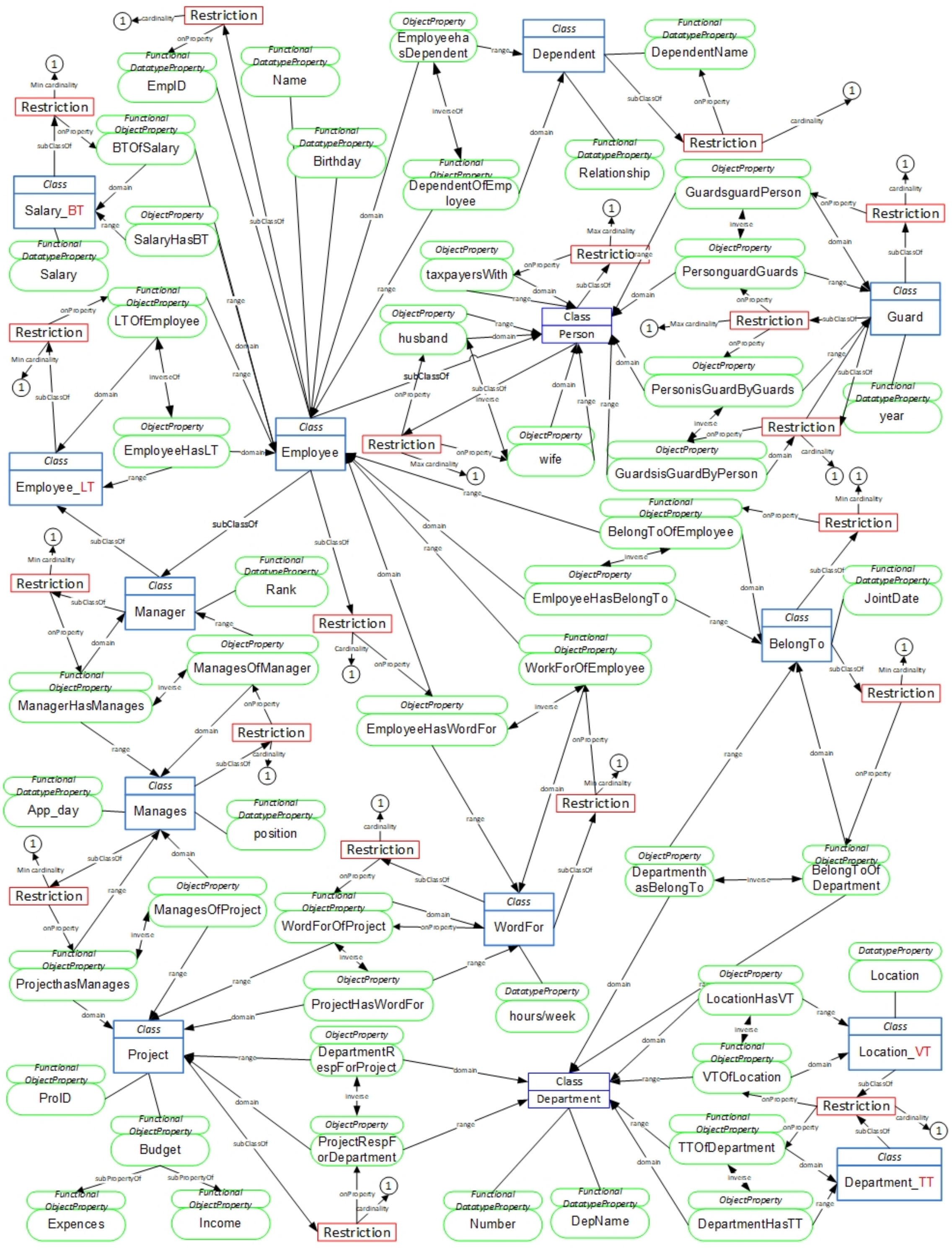

Restriction
Functional DatatypeProperty EmpID
Functional DatatypeProperty Name
ObjectProperty EmployeehasDependent
Class Dependent
Functional DatatypeProperty DependentName
Restriction
Functional ObjectProperty BTOfSalary
Functional DatatypeProperty Birthday
Functional ObjectProperty DependentOfEmployee
Functional DatatypeProperty Relationship
Restriction
ObjectProperty GuardsguardPerson
Restriction
Class Salary_BT
ObjectProperty SalaryHasBT
Functional DatatypeProperty Salary
ObjectProperty taxpayersWith
Restriction
ObjectProperty PersonguardGuards
Class Guard
Functional ObjectProperty LTOfEmployee
Restriction
ObjectProperty husband
Class Person
Restriction
ObjectProperty PersonisGuardByGuards
Functional DatatypeProperty year
ObjectProperty EmployeeHasLT
Class Employee
Restriction
ObjectProperty wife
Restriction
ObjectProperty GuardsisGuardByPerson
Class Employee_LT
Functional ObjectProperty BelongToOfEmployee
Restriction
Restriction
Class Manager
Functional DatatypeProperty Rank
Restriction
ObjectProperty EmlpoyeeHasBelongTo
Functional DatatypeProperty JointDate
Class BelongTo
ObjectProperty ManagesOfManager
Functional ObjectProperty WorkForOfEmployee
Restriction
Functional ObjectProperty ManagerHasManages
ObjectProperty EmployeeHasWordFor
Restriction
Restriction
Functional DatatypeProperty App_day
Class Manages
Functional DatatypeProperty position
Restriction
Restriction
ObjectProperty DepartmenthasBelongTo
Functional ObjectProperty BelongToOfDepartment
Restriction
ObjectProperty ManagesOfProject
Functional ObjectProperty WordForOfProject
Class WordFor
Functional ObjectProperty ProjecthasManages
ObjectProperty ProjectHasWordFor
DatatypeProperty hours/week
DatatypeProperty Location
ObjectProperty LocationHasVT
Class Project
ObjectProperty DepartmentRespForProject
Class Location_VT
Functional ObjectProperty ProID
Class Department
Functional ObjectProperty VTOfLocation
Restriction
Functional ObjectProperty Budget
ObjectProperty ProjectRespForDepartment
Functional ObjectProperty TTOfDepartment
Class Department_TT
Functional ObjectProperty Expences
Functional ObjectProperty Income
Restriction
Functional DatatypeProperty Number
Functional DatatypeProperty DepName
ObjectProperty DepartmentHasTT
cardinality
Min cardinality
Max cardinality
onProperty
subClassOf
domain
range
inverse
inverseOf
subPropertyOf
1

Figure 18. An example of OWL ontology transform from Figure 1

REFERENCES

[1] M. Fahad, "ER2OWL: Generating OWL ontology from ER diagram", *IFIP The International Federation for Information Processing*, Beijing, China, October 19-22, 2008.

[2] Sujatha R Upadhyaya, P Sreenivasa Kumar, "ERONTO: A tool for extracting ontologies from extended E/R diagrams", *ACM Symposium on Applied Computing*, Santa Fe, New Mexico, March 13 - 17, 2005, pp. 666-670.

[3] Igor Myroshnichenko, M.S., Marguerite C. Murphy, Ph.D., "Mapping ER schemas to OWL ontologies", *Semantic Computing. ICSC '09. IEEE International Conference*, Berkeley, CA, USA, Sept. 14-16, 2009, pp. 324-329.

[4] Q. Hoang and P. Thuong, "Expressing a temporal entity-relationship model as a traditional entity-relationship model", in *7th International Conference, ICCCI 2015*, Madrid, Spain, September 21-23, 2015, pp 483–491.

[5] Toan Van Nguyen, Hoang Lien Minh Vo, Quang Hoang, and Hanh Huu Hoang, "A new method for transforming TimeER model-based specification into OWL ontology", in *8th Asian Conference on Intelligent Information and Database Systems ACIIDS*, Da Nang, Viet Nam, February 27, 2016, volume 642, pp 111-121.

[6] Q. Hoang, H.L.M. Vo, and V.H. Vo, "Mapping of nested multi-valued composite attributes: an addition to conceptual design for XML schemas", in *Asian Conference on Information Systems 2014 - ACIS 2014*, Nha Trang, Viet Nam, 2014.

[7] James Dullea, Il-Yeol Song, and Ioanna Lamprou, "An analysis of structural validity in entity-relationship modeling", *Data and Knowledge Engineering - DKE*, Netherlands, vol. 47, no. 2, pp. 167–205, 2003.

[8] Grigoris Antoniou, Frank van Harmelen, "Web Ontology Language: OWL", in *Handbook on Ontologies*, Springer, 2009, pp. 91–110.

[9] M.K. Smith, C. Welty, and D.L. McGuinness, "OWL web ontology language guide", *W3C Recommendation*, 11-12/ 2004. [Online]. Available: https://www.w3.org/TR/owl-guide/. [Accessed 2 2 2017].

[10] B. Motik, P.F. Patel-Schneider, B. Paria, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler , et al., "OWL 2 web ontology language: structural specification and functional-style syntax", 11 -12/2012. [Online]. Available: https://www.w3.org/TR/owl2-syntax/. [Accessed 8 2 2017].

[11] H.C. Cheng, J.L. Cheng, and T.L. Chin, "Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, Transactions on IEEE*, vol. 39, no. 4, pp. 459-473, 2009.

[12] Pasapitch Chujai, Nittaya Kerdprasop, Kittisak Kerdprasop, "On transforming the ER model to ontology using protg OWL tool", *International Journal of Computer Theory and Engineering*, vol. 6, pp 484-489, 2014.

[13] Hoang Huu Hanh, Le Manh Thanh, *Course of Semantic Web*, Vietnam Education Publishing House Limited Company, 2012 (in Vietnamese).

[14] R. Elmasri, S.B. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 7th edn, 2015.

[15] G. Antoniou, and F.V. Harmelen, *A Semantic Web Primer*, Cambridge: The MIT Press, 2004.

Received on June 18, 2017
Revised on June 01, 2018