Transformation of OWL Ontology Sources into Data Warehouse

M. Gulić
Faculty of Maritime Studies, Rijeka, Croatia
marko.gulic@pfri.hr

Abstract - The Semantic Web, as the extension of the traditional Web, provides the semantic annotations of the information generated by different organizations. Semantic annotations are stored within ontologies. Ontologies are expressed using ontology language. Web Ontology Language (OWL) is one of the most popular ontology languages. As a result of the increasing use of ontologies, large quantity of complex, heterogeneous and semi-structured semantic data sources exists. There is plenty of useful information in these data sources that can be analyzed and used in the decision making process of an organization. In order to facilitate the analysis of semantic data, new data warehouse tools that support semantic data analysis must be made. Data warehouse is used in traditional business analysis and decision making processes. A star schema is the most common design of data warehouse. In this paper a method that transforms OWL structure into the star schema of data warehouse is proposed. After the designer chooses the fact in the ontology, a method transforms OWL into the star schema. At the end, the designer selects which elements in the schema remain while creating physical data warehouse.

I. INTRODUCTION

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [1]. Therefore, the goal of Semantic Web is the creation of standards and technologies that help machines to understand more about the Web data. These standards and technologies will improve user search results, data integration, navigation etc. Semantic annotations for Web data are stored within ontologies.

Ontology is a shared understanding of some domain of interest [2]. Ontology defines a set of entities and relations between them in a way that both humans and machines understand it. There are various data and conceptual models that can be thought of as ontologies (e.g. folksonomies, UML models, XML schemes, formal ontologies, etc.). Ontologies are expressed in an ontology language. OWL [3] is one of the most popular languages that is recommended by W3C organization.

As the Semantic Web is rapidly increasing, a large quantity of heterogeneous, composite and semi-structured semantic data sources exists. In these data, there is a lot of useful information that can be used in the decision making process of some company. For example, the data of the sales that were completed through the Web using a common ontology between two organizations can be used in decision making process. Data warehouse has proved as a good solution in decision making process. According to [4], a data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. The process of creating the data warehouse includes business demands, data design, architecture design, implementation, and deployment [4]. A data modelling is included in the data design. A dimensional fact model is one of the most popular models. The dimensional fact model consists of a fact table and its associated dimension tables [5]. These dimension tables consist of descriptive attributes that define some fact. Fact table also has attributes which are numeric and additive. If the presentation area is based on a relational database, then this dimensional fact model is implemented with a star schema [5]. The star schema is different from typical relational databases that are in the third normal form.

In order to facilitate the analysis of semantic data sources in the data warehouse, new warehouse tools need to be made. The tools [7, 8, 9 and 10] transform OWL ontology into a relational database, but do not deal with the star schema. Therefore, the additional transformation from relational database to the star schema is necessary. Furthermore, there is a risk of losing relevant information during multi-step transformation. Some relations (hierarchy between classes, symmetric relations etc.) within the OWL ontology could be lost after transformation of ontology into a relational database. Therefore it is better to implement a method that transforms OWL ontology into the star schema directly.

Furthermore, these solutions (except the [7]) deal with all subclasses of certain class in a way that the particular table is created for every subclass. This solution is not good for transforming relational database into the star schema because every subclass is managed like specific entity which is not the true. However, we take advantage of their transformation and improve several procedures to transform ontology into the star schema. Only the authors in [11] propose the direct transformation from the OWL ontology to the star schema, but with several drawbacks. The main drawback of their work is that data warehouse does not include the data of instances of certain class and its subclasses that does not have the values for certain attributes in the ontology that are specific for several subclasses.

In this paper a method that transforms OWL Lite ontology into the star schema directly is proposed. The
limitations described in the previous paragraph are also resolved. Our method deals with a class and its subclasses in a way that these classes represent one entity. Therefore the certain class and its subclasses become one dimension in the data warehouse. Our method also stores the data of all individuals of parent class and all its subclasses despite some individuals do not have values for several attributes that are specific only for certain subclasses. In this way, the data warehouse contains total data from the ontology, which is essential if the analyst wants to get the accurate results.

The paper is organized as follows. In Section II the terminology of OWL Lite ontology and star schema is introduced. In Section III related work is discussed. In Section IV the method for transforming OWL structure into the star schema is presented. Finally, the conclusion is given in Section V.

II. TERMINOLOGY

A. OWL Lite ontology

OWL language (figure 1) is used when the information in a document need to be processed by a machine. OWL represents the meaning of terms and the relationships between them. OWL has three sublanguages: OWL Lite, OWL DL, and OWL Full. In this paper, the OWL Lite ontology is used. According to [12] the definitions of the OWL Lite elements that are essential for transformation are given below:

An owl:Class defines a group of individuals that belong together because they share some properties. For example, HP1005 and Canon505 are both members of the class Printer (figure 1). A rdf:subClassOf establishes a hierarchy between one or more class. For example, the class Printer could be stated to be a subclass of the class Product (figure 1). An Individual is instance of the certain class, and properties can relate one individual to another. Individuals carry data that can be used for analysis.

A rdf:Property is used to define relationships between individuals (owl:ObjectProperty) or between individuals and data values (owl:DatatypeProperty). Example of properties can be named as hasProduct, hasPrice etc. A rdfs:domain of a property defines the individuals to which the property can be applied. For example, the property hasProduct can have the domain of Invoice (figure 1). A rdfs:range of a property defines the individuals that the property may have as its value. For example, the property hasProduct can have the range of Product (figure 1). An inverseOf element defines a property that can be the inverse of another property. If the property P1 is the inverse of the property P2, and X is related to Y by the P2, then Y is related to X by the P1. Properties may be stated to be SymmetricProperty. If a property is symmetric, and the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P. If a property is FunctionalProperty, than a unique value is added to each individual that has this property. Transitive and InverseFunctionalProperty also exist in the OWL Lite ontology, but this type of property does not affect the transformation of OWL ontology into a star schema.

OWL Lite restrictions define the rules for using properties by particular instance. There are 5 restrictions: allValuesFrom, someValuesFrom, minCardinality, maxCardinality and cardinality. AllValuesFrom defines a local range restriction of some range class. SomeValuesFrom defines that at least one value of restricted property related to some instance is of the certain type. MinCardinality defines the minimal number of values of certain property that the individual must have. MaxCardinality defines the maximal number of values of certain property that the individual must have. Cardinality defines the exact number of values of certain property that the individual must have.

OWL uses the XML Schema dataTypes for defining the range of owl:XMLSchema dataTypes.

B. Dimensional fact model

In this paper we adopt the dimensional fact model as a conceptual model that presents a data warehouse by a set of fact schemes with facts, measures, dimensions and hierarchies [16]. A fact is the center of interest in the decision making process.

A fact represents an event that occurs dynamically in the business process (e.g. invoice). A fact consists of measures. Measures are attributes that specify a fact. The most useful measures are additive and numeric. For example, every invoice has the total price, total tax etc.

Dimensions consist of discrete attributes that describe the fact event. Dimensions define the grain of the fact. For example, dimensions for every invoice are product, date, time, store etc. In this example, an analyst may request to see the money amount by product, by week, every evening from 7pm to 9pm. An analyst usually uses the OLAP (On-
Line Analytical Processing) tools for analyzing data warehouse. These tools are based on a multidimensional conceptual view of data. In this paper, a transformation from OWL ontology to ROLAP (Relational On-Line Analytical Processing) system is proposed. The ROLAP uses the relational model for representing dimensional fact model. The implementation of dimensional fact model in the relational database is called star schema (figure 2). The star schema consists of a set of dimension tables and the fact table. Each dimension has a set of attributes that describe the dimension. The fact table has a primary key that is a set of foreign keys of dimension tables. As it was stated before, the fact table also has the numeric and additive attributes that are measures of the fact.

III. RELATED WORK

As it was stated before, new data warehouse tools need to be made for more accurate analysis of semantic data sources. The focus of this paper is on the direct transformation of OWL ontology into the star schema because when a multi-step transformation is performed, some relations defined in the OWL ontology could be lost. The authors, who explore the transformation of OWL ontology into the relational database [7, 8, 9 and 10] deal with the transformation in the third normal form, not with the star schema. There are many solutions that transform the relational database in third normal form into the star schema. However, the main problem here is that the OWL ontology has a hierarchy of entities that appear in the ontology. The authors in [8, 9 and 10] propose the transformations that create an extra table in the relational database for every class in class hierarchy. For example, the class Product has subclass Food that has the additional attribute named lifetime. Transformations in [8, 9 and 10] create two tables, one for Product and one for Food. These two tables have all common attributes except for the attribute lifetime. The relation rdfs:subClassOf defined in OWL ontology (Food is a subClassOf Product), between Product and Food is lost after the transformation in the third normal form hence Product and Food will represent two different entities in the third normal form. When the transformation from third normal form into the star schema is made, the star schema has two dimensions for Food and Product instead of one dimension because these two classes represent the same entity, except that the Food class has the additional attribute. Hence, the wrong transformation could happen after losing a class hierarchy in the third normal form. However, some solutions of transformation proposed in [7, 8, 9 and 10] are used and integrated in the method proposed in this paper. A table for the parent class is created in all transformation. If a class has the ObjectProperty and the maxCardinality is 1, a column is created and it is a foreign key for the range class of the ObjectProperty. If a class has the DataTypeProperty and the maxCardinality is 1, a column is created and the type of the column is the most similar type in the relational database comparing XML data types added as a range of the DataTypeProperty. The authors in [10] propose the rules for transforming every XML data type into the certain data type in the relational database. If the ObjectProperty is Functional, the cardinality of the property is 1 and the column in the table is created. If the cardinality of any property is greater than 1, a new table will be created that has the primary key which is a combination of two foreign keys. The first key references to the domain class and the second references to the range class.

Although the authors in [7] present the transformation from OWL ontology in third normal form, they propose an interesting solution. The subclasses of parent class are recognized as the same entity but the ontology is transformed into the object relational database. For example, the class Food inherits all attributes of class Product and has the additional attribute lifetime. This solution of recognizing hierarchical classes as the same entity is used in this paper.

The authors in [11] provide the transformation from OWL ontology to the star schema but with notable limitations. For example, the analyst wants to analyse certain class according to the first attribute which is common for certain class and its subclasses in the ontology, and to the second attribute which is common only for several subclasses. Only the individuals of these several subclasses that have both attributes will be stored in the data warehouse. Therefore, when the analyst wants to see analysis just according to this first common attribute of a class and all of its subclasses, she will get only the individuals of these several subclasses that have the first and the second attribute in the OWL ontology.

IV. THE METHOD FOR AUTOMATICALLY TRANSFORMING OWL STRUCTURE INTO THE STAR SCHEMA

Transformation of OWL ontology into the star schema cannot be fully automated. The designer always has to choose which data she wants to analyze. Hence, the most important topic is that the designer selects what entity will represent the fact. As it was already mentioned, the direct transformation of OWL ontology into the star schema is proposed. The Invoice ontology (figure 4) will be taken as example for transformation of OWL Lite ontology into the star schema. In figure 4, The Invoice ontology is displayed as a directed graph to facilitate the steps of the transformation. That ontology is a combination of two real ontologies. First ontology [13] describes the main parts of the e-invoice while the second ontology [14] is the professional web vocabulary for e-commerce that describes a large number of products and services. These two ontologies are merged to show the real situation that can happen in a company while analyzing sales. For example, an analyst of the company...
wants to analyze sale by certain month, year, product etc. Invoice ontology that is used for example has only the specific parts of the ontologies [13] and [14] through which it can be shown all capabilities of transforming OWL Lite ontologies into the star schema. The names of ontology classes in the example are shorter than real names in the ontologies [13] and [14] to facilitate the transformation steps. When the designer selects the fact, the transformation starts. First, the dependency graph will be created from the OWL ontology. The dependency graph is an intermediate structure used to provide a multidimensional representation of the XML data describing the fact [15, 16]. Here, the graph is used to provide a multidimensional representation of the OWL data. Therefore, in this case, the dependency graph is a directed rooted graph whose vertices are classes or their data attributes in the OWL ontology. Pseudo code of transforming OWL ontology into the dependency graph is shown in the figure 3.

For example, an analyst wants to examine the product sale. The designer selects the object property hasInvoiceLine in the figure 4 to be a fact. The object property hasInvoiceLine becomes the first node (the designer renamed it as Product sales) in the dependency graph. Then, the algorithm gets all dataType and object properties of the domain (Invoice class) and range (InvoiceLine class) classes of the object property hasInvoiceLine and puts them into a list of properties. The Invoice class has one dataType property (hasInvoiceID) and four object properties (hasDelivery, hasTime, hasDate, hasInvoicePrice). All properties have the cardinality equal to 1 and therefore the range classes (Date, Time, Delivery, InvoicePrice) or dataType (xsd:string named InvoiceID) of these five properties are added to the dependency graph (figure 5 a)).

Figure 3  
Pseudo code for transforming OWL into the dependency graph

```java
Create dependency graph starGraph

Select a class or an ObjectProperty (with cardinality greater than 1) in the OWL ontology that will represent the fact;

Create the lists factDataTypeProperty and factObjectProperty;

IF fact is ObjectProperty THEN

Get all objectProperties and dataTypeProperties of the domain class and range class of the fact and put them in the lists factObjectProperty and factDataTypeProperty;
ELSE
Get all objectProperties and dataTypeProperties of the of the fact class and put them in the lists factObjectProperty and factDataTypeProperty;
END IF

ProcessDataTypeProperties(factDataTypeProperty, true);
ProcessObjectProperties(factObjectProperty, true);

FOR every dataTypeProperty in dataTypeProperties

ProcessDataTypeProperty(dataTypeProperty, all);
END FOR

ProcessObjectProperties(objectProperties, all);
FOR every objectProperty in objectProperties

ProcessObjectProperty(objectProperty, all);
END FOR

ProcessDataTypeProperty(dataTypeProperty, all){
Cardinality(dataTypeProperty, all)};
}

Cardinality(property, all){
   IF property cardinality > 1 AND the property represents a many-to-many relationship THEN

      Create node (with name of the property range Class) with cardinality > 1 (sign -> double line) and insert into starGraph; (sign # if all is false)
   ELSE

      Create node (with name of the property range Class) with cardinality = 1 and insert into starGraph; (sign # if all is false)
   END IF
}
```

Figure 3 Pseudo code for transforming OWL into the dependency graph
InvoiceLine class has two dataType properties (hasInvoiceLineID, hasQuantity) and two object properties (hasProduct, hasInvoiceLinePrice). All properties have the cardinality equal to 1 therefore the range classes (Product, InvoiceLinePrice) or dataTypes (xsd:string named InvoiceLineID and xsd:string named Quantity) of these four properties are added to the dependency graph (figure 5a). Thereafter, for every range class inserted into the dependency graph, the algorithm obtains all object and dataType properties.

The properties of the Delivery are hasDate, isDeliveryFor, hasInvoiceID and hasTime. These properties are inserted into the list of properties. An interesting object property is isDeliveryFor because it is an inverseOf property of the hasDelivery property in the Invoice class. The range class (Delivery) of the hasDelivery property is already inserted in the dependency graph through the Invoice node. Hence, the node for the range class (Invoice) of the object property isDeliveryFor will not be created because it is already created. If the algorithm creates the Invoice node again, an infinite recursion will occur. Other properties have the cardinality equal to 1 therefore the range classes (Date, Time) or dataTypes (xsd:string named DeliveryID) of these three properties are added to the dependency graph (figure 5b). Classes Date and Time (added through Delivery node) have one dataType property (hasDataValue and hasTimeValue) therefore one node is added in the Date (DateValue) and Time (TimeValue) node (figure 5b). It can be seen that the dependency graph stops expanding in some direction when the remaining properties are only dataType properties. Hence, the classes Date and Time stop expanding when the DateValue and TimeValue are added into dependency graph.

The properties of the Date and Time in the Invoice node are the same as the properties Date and Time in the Delivery node. Hence, the same nodes are created as earlier (figure 5b). At the end of the algorithm, the designer will merge this Date and Time nodes in the star schema because they represent the same dimensions. The dataType property hasPriceValue is the only property of the class InvoicePrice therefore the node PriceValue is created (figure 5b). The same procedure is made for the hasPriceValue property in the InvoiceLinePrice class (figure 5b).

Furthermore, the Product class needs to be expanded in the dependency graph. The Product class has two subclasses, FoodProduct and ComputerProduct. The ComputerProduct class has one subclass, the Game class. When the class has subclasses, the first step is to create the subClassName node that has a quadratic shape (figure 5b). This shape defines that each value of the subClassName attribute in the dimension table that corresponds with the subClassName node will be one of the names of subclasses (FoodProduct, ComputerProduct and Game in the example) or the root class (Product). The designer can rename the name of the subClassName node after creating the column in the relational database. In this example the column will be named categoryName (figure 6). After creating a node for all subclasses names and their root class, the algorithm gets all common dataType and object properties of the root class (Product) and all child classes (FoodProduct, ComputerProduct, Game). These properties are hasProductPrice and hasProductName. These properties have the cardinality 1 therefore two nodes (ProductName, and ProductPrice) are created in the dependency graph. The node Product is the parent node of these nodes. After creating nodes for properties of the Product class that are common to all subclasses of Product class, the properties that are special just for a subset of all subclasses need to be processed. These properties are isMultiplayered and hasIngredient. The

Figure 4 An example of the Invoice OWL ontology displayed as directed graph

Figure 5 Dependency graph for the transformation of the Invoice OWL ontology
The designer selects measures, dimensions and useful attributes in the final step before implementing the star schema. Let us assume that the designer wants the quantity, invoiceLinePrice and productPrice to be measures and the nodes Date, Time, DeliveryDate, DeliveryTime and Product to be dimensions. Only the nodes that represent the range of dataType property in the dependency graph are actually in the dimension tables because every object property consists of several dataType properties that carry the data of ontology. The designer deleted dataType nodes that the analyst will not need (in our example InvoiceID, DeliveryID and InvoiceLineID). The star schema of the Invoice example can be seen in Figure 6. Every column has its own data type in the relational database that is the most similar to the XML data type defined in the ontology. The method that maps similar data types from XML to SQL is described in [10]. The star schema has one snowflake structure for the ingredients that the product could have because it represents the many-to-many relationship between the product and the ingredients. Furthermore, the designer would probably enrich the date dimension. For example, she would like to have information if the certain date is a weekend (Figure 6), workday, etc. In a similar way the designer will enrich the time dimension.

V. CONCLUSION

In this paper a method that transforms OWL ontology into the star schema is proposed. The method semi automatically transforms all elements of the ontology after the designer selects which class in the ontology will be the fact. Before implementing the star schema, the designer selects which data will be included in the data warehouse in order to drop unnecessary data from the data warehouse.

Our method identifies all subclasses of the certain parent class as the same entity like parent class and thus it differs from other methods that transform OWL ontology into the normalized relational database or star schema. It is important because the certain class and its subclasses could become one dimension in the star schema. In this way, all individuals of parent class and all of its subclasses will be stored together although they have some different attributes that are defined only for some subclasses.

In the future work the attention will be paid on the ranges of the property that are composed of more classes. The solution for the symmetric property should also be improved. Symmetric property is like a recursive structure and it is difficult to define when the algorithm stops while obtaining the data from that property. Also, the implementation of the tool that uses the method described in this paper will be made.

REFERENCES

[7] X. Liu, DataWarehousing Technologies for Large-scale and Right-time Data, dissertation, Faculty of Engineering and Science at Aalborg University, Denmark, 2012.