Using Assertion Capabilities of an OWL-Based Ontology for Query Formulation

Kamran Munir, Mohammed Odeh, Peter Bloodsworth, and Richard McClatchey
Centre for Complex Cooperative Systems (CCCS),
University of the West of England (UWE), Frenchay Campus
Bristol, United Kingdom
kamran.munir@cern.ch

Abstract—This paper reports on the development of a framework to assist users in formulating relational queries without requiring a complete knowledge of the information structure and access mechanisms to underlying data sources. The emphasis here is on exploiting the semantic relationships and assertion capabilities of OWL ontologies to assist clinicians in writing complex queries. This has been achieved using both a bottom-up and top-down approaches to build an ontology model to be the repository for complex end user queries. Relational database schemas are mapped into the newly generated ontology schema to reinforce the current domain ontology being developed. One of the key merits of this approach is that it does not require storing data interpretation, with the added advantage of even not storing database instances as part of the domain ontology, especially for systems with huge volume of data.

Keywords—Ontology assertions, description logic, relational query formulation, and metadata.

I. INTRODUCTION

Over the past few years, there has been a tremendous increase in the number of data providers, in the volume, and the heterogeneity of generated data. For the foreseeable future, it is likely that most data will continue to be stored in relational databases. As the number of data sources increases, more intelligent data retrieval techniques, focusing on the information content and semantics, are required if users are to make the most effective use of these sources. Currently, ontologies are widely being used to manage such semantic information. In order to work with this information in ontology-driven data analysis applications, some methods and associated tools are required to bridge these two relational and ontological models. To achieve this, intelligent query formulation techniques need to be developed that can harness the semantic knowledge defined within the ontology to formulate relational queries. This paper focuses on exploiting semantic relationships and assertion capabilities of a DL based OWL ontology for information querying. The aim is to help users in formulating queries without requiring a complete knowledge of the information structure and access mechanisms to the underlying data sources. One approach to achieve this goal is the use of source metadata information defined as ontological concepts within the internal structure of the respective ontology.

Currently there are several methods and tools available for transforming relational databases to ontologies [1, 2, 3] and domain ontologies to conceptual schema [4]. In this research some existing transformation rules are extended to represent the source metadata information in an ontology for the purpose of query formulation. Here the ontology also stores the domain knowledge as concept assertions. This ontological information, is then used for situation-based information querying. Users are assisted in describing the search criteria by an ontology stored in an ontology server. Once the new criteria have been described, the users do not need to manually write the complex source query languages and the query is executed transparently. Query formulation techniques ‘understand’ the conceptual relationships between ontological concepts and the associated knowledge required to formulate a query. This approach has been applied to the integrated schema generated for the EU funded Health-e-Child1 (HeC) project with the aim of providing ontology assisted query formulation techniques. The aim being to provide simplified access methods to clinical researchers working with millions of medical records across Europe.

The remainder of this paper is organized as follows. Section II presents the related work in the area. Section III introduces the system architecture with the main focus on the ontological representation for formulating queries. This section also discusses a query formulation approach that utilises domain ontologies to assist in providing database query services. The final Section IV concludes the paper and considers the current status of the research and some directions for future work.

II. RELATED WORK

Currently there are several tools available that transform relational databases into ontologies. DataGenie [1] is a plug-in for Protégé [11] that imports data from a relational database to an ontology. Similarly, related work has been done by [2], [3] and [7] on the transformation between relational databases and ontologies. These mappings are fairly trivial where each table row to an instance. In addition, the foreign key columns are used to link an instance of a class to instances of another class. In this research, while using domain ontologies to assist the user in creating relational queries, some of the basic rules to generate domain ontology from relational schema remain the same. However, this relational schema to ontology
transformation is different in the sense that relational data are not transformed and then stored as ontology instances. In addition, ontologies are used to assist in the provision of database querying services to the end users and their applications.

On the one hand a database relational schema provides a logical map of the information content of the database along with related semantic data control assertions following the relational model of databases. On the other hand, although ontology schemas share the data elements’ conceptualisation aspects of relational schema, ontology model specifications and in particular OWL ontologies (utilised in this research) are based on Description Logic (DL) theory [5] and thus referred to as OWL DL. But, in order to represent a relational data model in OWL DL, respective transformations of the relational model to DL remain a critical requirement to seek consistency, completeness of these transformations. In relation to this, work has been carried out in [8] which describes the relationship between entities in the ER model and DL theory. In this research some of the basic ontology to DL transformation rules are employed and extended to handle the requirements for formulating database queries.

Furthermore, work has been carried out in [4] to map domain ontologies to conceptual data models, where several mapping rules were proposed that guide the transformation from domain ontology to a respective conceptual schema. One of these mapping rules describes the transformation of ontology properties to an entity-attribute in the conceptual model. This rule has been extended in our approach to define mappings between a DL based ontology and a relational database.

Most existing query formulation approaches are based on a RDF [6, 9, 10] structure which, although yielding schema information, provides insufficient knowledge for query formulation. Some of these approaches use only direct relationships between ontology concepts such as is-a and part-of to access data. Recent developments in modern DL-based ontology languages such as OWL DL, have enabled ontologies to become more expressive than RDF alone. Such ontologies can be utilised to formulate user queries without manually navigating through a large ontology. Unlike previous approaches, the system presented in this paper does not store data elements’ instances as part of the ontology or even link them directly to ontology concepts. Often it is not practically feasible to store all data as part of a certain domain ontology and especially for systems with large amounts of data. The data that are stored as part of the ontology need to be loaded in memory to perform the “select” query operation. In addition, it may become both a complex and time consuming activity to directly link all data with associated ontological concepts. However, in our new approach, OWL ontology concepts are specified with related assertions that drive the process of query reformulation.

III. THE QUERY FORMULATION SYSTEM

This system (as depicted in Figure 1) has two major subsystems: (1) the Ontology Assisted Query Formulation and (2) the Query Reformulation Engine. The Ontology Assisted Query Formulation subsystem is composed of three components: (1) ontology creation to assist in formulating queries, (2) the ontology server, and (3) the ontology assisted query formulation process. This paper focuses mainly on the ontology assisted query formulation subsystem and its components.

A. Ontology Creation to Assist in Formulating Queries

As a first step towards ontology assisted query formulation, an ontology is created which stores database metadata information within the basic ontology structure. A relational database includes tables, columns, data types, constraints, cardinality relationships and others. Similarly, the ontological model includes the classes, data types, assertions, properties, and other semantics. However, for the purpose of query formulation this approach does not require domain ontology to include all constructs of the relational model. The domain knowledge is expressed in terms of ontology property assertions that need to be consistent with the basic ontology structure. It is also possible to include the domain knowledge from widely available domain ontologies. The following mapping rules are briefly outlined below that explain what needs to be included in the ontology for the purpose of query formulation:

1) An entity type E in a relational schema R(E) is mapped to a class in the ontological model. This does not include
mapping of the relational schema that is based on the cardinality-based relationships between entity types into classes. For example if R is a relationship in a database D that links an entity type E1 in D to the entity type E2 in D, with P1 being the primary key of E1 and P2 being the primary key of E2; here R could be many-to-many, a many-to-one or a one-to-many relationship type, then the relationship type R is mapped to a schema relation denoted by R(R) = P1 U P1. In such situations only the entity type E1 and E2 are represented into the ontology as classes. In practice the day to day transactions are stored with such R(R) relations. We do not store such relations into the ontology as classes because we do not aim to store transactional data in the ontology rather the entity related data is used to query such transactional data. This also allows us to keep the ontology simple and light weight.

2) Each entity of the entity type is converted into the subclass of the entity-type class. Furthermore, entities are also defined as instances of the corresponding subclass. This is required for the definition of Domain and Range classes for the object properties. During the query formulation process, in order to link the instances of one class with the instances of another class, object properties are utilised. However, class instances cannot be directly linked as Domain and Range with object properties, therefore each entity is converted into instances and stored under the corresponding class. This class is then defined as a Domain or Range for the related object properties. In addition, similar entities could be further defined under one generalized parent class, if needed.

3) An object property links a class (as Domain) to another class (as Range). Other than classes which directly store instances, each class (entity) is associated to an object property pointing to itself as property Range. The property is defined both symmetrically and functionally. This rule is similar to rules used in the majority of database to ontology mapping approaches [2, 3]. However, this approach does not require data to be stored as ontology instances. Therefore, unlike previous approaches it does not require the definition of properties for each instance of the entity table as a property Range.

4) The column values are represented in OWL by instances. Therefore, each column corresponds to an object property. Moreover, each foreign key is mapped to an object property, in this case it will link the primary key row instance(s) (whose corresponding entity is defined as Domain class) with a foreign key row instance(s) (whose corresponding entity is defined as Range class).

5) In a relational model, the general schema restrictions (e.g. not-null, unique etc.) are used to restrict data entry i.e. Insert or Update operations. In this research we are dealing with “select” query operation exclusively. Therefore, it does not require the definition within the ontology of general schema datatype ranges and cardinality restrictions or specific constraints expressed in the schema as axioms.

These rules are applied on a subset of the HeC patients database. Due to space limitation, it is not practical to describe the complete HeC database schema [12, 13]. Therefore, as a running example throughout the paper, we shall use a small subset of the database relations from our Patients’ database:

1) Patients_Data (patient_id, clinical_test, clinical_test_value, ..)
2) Clinical_Test (clinical_test_id, clinical_test_name, …)
3) Clinical_Test_Values (id, clinical_test_id, clinical_test_value)
4) Patient (id, description, …)

Rows in the table clinical_test (entity E1) (see figure 2) store all the possible clinical tests that can be taken for a particular patient. The clinical test values table (entity E2) stores all the possible clinical test results for any particular clinical test. The patients data table R(R) references the patients, their clinical tests and results of medical tests.

This section explains how the patient’s database metadata is represented in an OWL ontology by applying the above specified conversion rules. As a first step the conversion rules 1 and 2 are applied to the entity types. A clinical_tests class is added into the ontology that contains all of the clinical tests. These would include for example, headaches, double_vision, thrombosis_sequlae and orthopedic_sequelea as subclass.
individuals. Similarly, the second class namely clinical_test_values has been defined to hold all possible clinical test values for each clinical test. We define clinical_tests and clinical_test_values as disjoint, so that an individual (or object) cannot be an instance of more than one of these two classes. Due to the fact that patients clinical tests can hold any type of result set value for each clinical test e.g. Boolean, String, Number, Float etc, further subclasses of clinical_test_values i.e. headaches_values, double_vision_values, thrombosis_sequelea_values and orthopedic_sequelea_values are created.

Once the basic structural elements of the domain ontology have been defined that are further enriched with domain knowledge. The domain knowledge is expressed in terms of OWL property assertions that need to be consistent with the basic ontology structure. We store this domain knowledge as ontology concepts. In this way the consistency of the domain knowledge with ontology concepts can be verified using an Ontology Reasoner. Users are able to select from concepts that define domain knowledge. Concepts restrictions are used to describe conditions for the selection of records that match some given criteria. These restrictions could be either singular or complex involving many conditions. For example, all conditions (as in figure 4) must match for a patient record to be selected as a member of a particular brian_tumor_disease type.

B. The Ontology Server

Once the ontology is defined it is then processed and stored in a database. At this stage the ontology to database mapping information is also stored in a separate table called mappings. The mappings table is created automatically during the ontology processing phase and it stores the information about ontology property links, database name, table name, column name, primary and foreign keys. Initially the mapping table only contains the information about ontology properties, which is then updated with the database metadata information. The ontological knowledge interface (as shown in figure 1) interacts with the ontology server to retrieve the ontological information. This information is then used to assist the users in defining their search criteria and to generate database queries. The query formulation engine utilises both database mappings and domain knowledge to formulate queries.

C. The Ontology Assisted Query Formulation Process

Users are guided throughout the query formulation process to select domain concepts and corresponding individuals or values to define the search criteria. Ontology properties are used to select the next appropriate concept, sub-concept or instance. This is achieved by following the semantic links between the concepts. Such links have been defined as the Domain and a Range of a property. Figure 5 shows this ontology assisted query formulation process.

Figure 3. Class Hierarchy and Corresponding Individuals

Figure 4. An example of property assertions that drive the selection of relevant records
Using the above conceptual schema to ontology mapping rules, each class (entity) is associated to an object property as a property Range (OWL construct). Furthermore, consider the entities in the relational models that are not associated with any other entity as when mapped into the ontological model they do not have any Domain class defined for the associated class property. Therefore, the query formulation process starts by presenting those properties to the user that have their domain defined as null and range as not-null. For example, in the fragment of the ontology shown in figure 3, the class property “hasClinicalTestName” is presented to the user and once the user selects the “hasClinicalTestName” property then the request is passed to “ListPropertyRangeClassData”. This takes a property value as input and lists its corresponding Range class instances.

At this stage, if the Range class has any instances they are presented to the user in a list. The user selects from the list the conditions that are to be added to the search criteria. However, if there are no instances defined for the Range class then the list of all subclasses is presented for further selection. For example, in case of the object property “hasClinicalTestName” the subclass values headaches, double_vision, thrombosis_sequelea and orthopedic_sequelea are retrieved. Now the user has two options either (1) to select the subclass value and add this condition to the overall search criteria or (2) to follow the link towards other classes where the class/subclass of the selected value is stored as the Domain of an object property or properties. For example: the subclass thrombosis_sequelea has an object property “hasThrombosisSequeleaValue” which stores thrombosis_sequelea as a domain class. In this case the Range class “cvt_Thrombosis_Sequela” has instances. Therefore, a list that contains the instances Asymptomatic, Moderate_Symptomatic, Sever_Symptomatic is retrieved. The user selects from this list of instances which search conditions to include in the overall search criteria. However, if for example there were no instances found for the Range class then the list of all subclasses is presented for further selection and the whole selection process begins again. In this way this approach allows multiple search conditions to be defined using different levels of the ontology hierarchy. The whole search criteria once defined can be saved as a new ontological concept that might be used by other users at a later time.

It is possible to select a concept from the predefined domain knowledge. The domain knowledge concept which stores the conditions as assertions that conform to the basic ontology structure, are retrieved from the ontology server. However, in order to differentiate between domain knowledge concepts and those that store the entity related data the following rules are applied:

1) Find all the classes (concepts) which are not defined as Domain or Range for any of the object properties.

2) A class selected in step-1 should only continue to be selected if the following condition holds: Any of its subclasses, is not defined as Domain or Range for any of the object properties. This condition also applies to the subclasses of a subclass e.g. If B is a subclass of A and C is the subclass of B, this implies that C is also a subclass of A.

Once the overall search criteria is described (or selected from the pre-defined domain knowledge) it is then passed to the query formulation engine which translates these conditions (defined as property assertions) into SQL. For the above example (in figure 4), A domain concept Brain_Tumor_Disease_X is described as OWL property assertions by using an ontology property namely “hasClinicalTestName”. For this example, it is assumed that the thrombosis_sequelea, headaches and orthopedic_sequelea are the clinical tests names related to brain tumor disease x. This information is described as follows by using the OWL hasValue property restriction:

\[
\text{hasClinicalTestName has THROMBOSIS_SEQUELEA UNION}
\]
In the above statements, the “union” operation implies that there is an “or” condition within each test condition. Here database mappings are used to convert the ontology property information into database table/column information (described in section III.A). Finally, the formulated query for this particular example will retrieve the data from the Patients table (as shown in Figure 2) where the values in Clinical_Test column matches any of the following values {thrombosis_sequelea, headaches, orthopedic_sequelea}. Further details about the query formulation engine are beyond the scope of this paper and will be presented in due course.

This approach has been applied on a selected subset of the integrated HeC patients’ database schema to perform query formulation tasks. For initial testing, this subset of patients’ was used to evaluate the complete ontology assisted query formulation process described in this paper. A prototype system has been developed and presented to domain experts for evaluation to inform its significance in assisting end user formulating complex queries.

IV. CONCLUSION AND FUTURE WORK

This paper has presented our initial framework for mapping relational database schema to a respective ontology schema in addition to strengthening the resultant ontology model further with domain knowledge. And, in particular the concentration has been on mapping database metadata to OWL ontology representations and on query formulation techniques that use ontological knowledge to assist users in querying complex databases during their clinical research. One of the key merits of this approach is that no interpretation of data needs to be carried out to be stored as ontology instances. This is clearly beneficial since the interpretation of data in existing data source(s) may cause some serious scalability issues with existing legacy applications. This approach is also useful where it is not practical to store all data as part of the respective domain ontology, especially for systems with a huge volume of data. Further work is being carried out to in relation to the Query Formulation Engine to handle more complex situations that involve multiple ontology assertions with a mixture of union and intersection operations in each property restriction.

REFERENCES


Authorized licensed use limited to: University of the West of England. Downloaded on October 23, 2009 at 13:10 from IEEE Xplore. Restrictions apply.